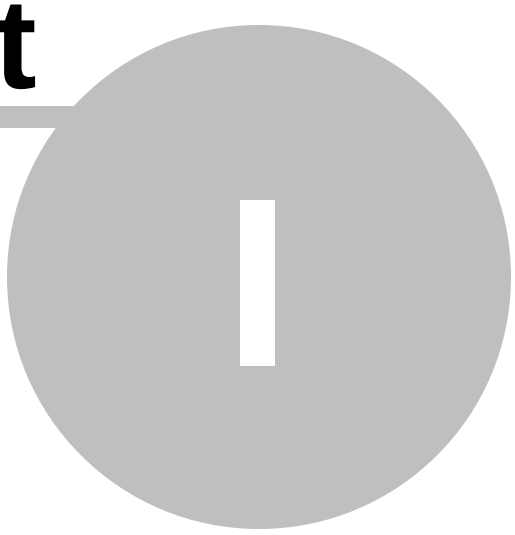
Kentico CMS 7.0 Developer's Guide

Kentico
unlimited website possibilities

www.kentico.com

Table of Contents

Part



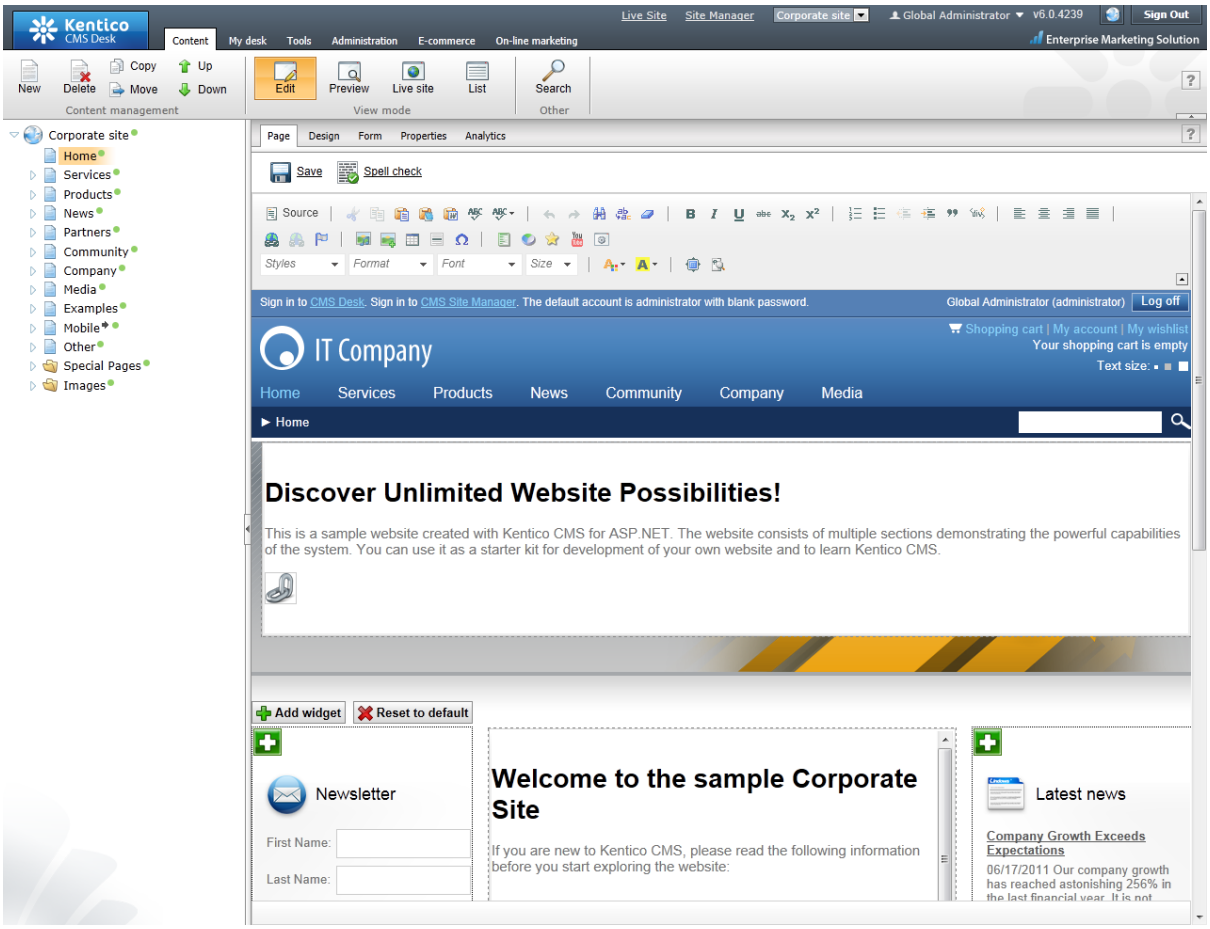
Introduction

1 Introduction

1.1 Introduction

Thank you for using Kentico CMS, the content management system for ASP.NET developers. Kentico CMS 7.0 empowers developers in creation of dynamic websites. It provides rich functionality out of the box:

- Flexible content repository for storing structured content with versioning and workflow.
- Powerful portal engine for assembling web pages from web parts.
- A wide range of built-in web parts and web controls.
- Export and import for easy deployment.
- Granular security model.
- Support for multiple-site configuration.
- Support for multiple-languages on a single website.
- Many built-in modules:
  - E-commerce
  - Forms
  - Forums
  - Image gallery
  - Media libraries
  - Staging
  - Content rating
  - Blogs
  - Polls
  - Reporting
  - Web Analytics
  - Time zones
  - Events
  - User contributions (Wiki)
  - and others



Kentico CMS enables a **high level of flexibility**, **customization** and **integration** – you can create:

- custom page templates
- custom web parts
- custom document types
- custom workflow schemas
- custom modules

Kentico CMS object model and **open, well-documented application programming interface (API)** allow you to enhance the system and script any system feature by your .NET code. You can also write custom handlers to add your own actions when a system event (such as creation of a new document) occurs.

More product information

You can download Kentico CMS, find additional information, make the purchase and get support at http://www.kentico.com

1.2 About this guide

Who is this guide for?

Kentico CMS Developer's Guide contains reference information for Kentico CMS developers and system administrators.

It is expected that you are familiar with the basics of Kentico CMS explained in Kentico CMS Tutorial (or Kentico CMS Tutorial ASPX if you are developing in the ASPX page templates mode). It is also expected that you have a certain degree of experience in ASP.NET web development and C# programming to be able to fully understand all articles in this guide.

What information can I find in the guide?

This guide should provide comprehensive reference for website development and administration in Kentico CMS. It provides description of the system's functionality and step-by-step examples of typical development and administration tasks.

How is the guide structured?

The guide is divided into several logical chapters:

- Kentico CMS overview - basic information about the system intended for a first-time user.
- Installation and deployment - covers the installation process, additional configuration tasks that might be needed after installation and a troubleshooting section with solutions to usual installation issues.
- Content management - information on how to manage the content of your website. It explains how pages can be edited, how the WYSIWYG editor can be used, how files can be managed, etc.
- Managing sites - management of sites, i.e. creation of a new site, running and stopping sites, import and export of sites and objects, license management, etc.
- Website settings - overview chapter with links to other sections of the guide where various website settings are explained in different contexts.
- Development - provides information on various topics related to website development and system administration.
- Modules - a reference for the out-of-the-box modules and other types of functionality in Kentico CMS.
- API programming and Kentico CMS internals - examples of the usage of Kentico CMS API and an explanation of the internals of the system.

Valuable references can also be found in the appendices:

- Appendix A - Path expressions - explains how path expressions can be entered in various parts of the system
- Appendix B - Web.config paramenters - list of keys which can be added to the *web.config* file of your site to perform additional low-level settings

Are there different versions of the guide?

This guide is available in three different formats. You can always find the most up to date version of the guides in the locations listed below:

- An **HTML** version of the guide can be found here.
- A **PDF** version of the guide can be found here.

- A **CHM** version of the guide can be downloaded here.

1.3 Where do I get further help?

This guide is not the only reference available. All documentation can be found in various formats at devnet.kentico.com/documentation.aspx.

If you can't find the information that you are looking for, please visit devnet.kentico.com. This is the official portal for Kentico CMS developers where you can find:

- Blogs
- Discussion forums
- Knowledge base
- FAQs

Another useful source of information is the Deliver Now! Methodology. It outlines all important aspects of the website development and delivery process with Kentico CMS. Apart from the base methodology text, it also contains a set of checklists, and templates usable for creation of project specification documents and website structure sheets.

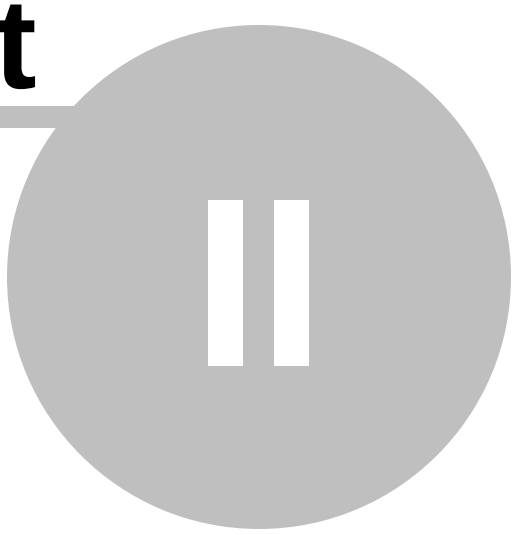
Paid license owners can also contact our highly-responsive support team, please visit www.kentico.com/support.aspx to learn more details.

You are also welcome to send us feedback on particular topics in this guide. You can do it by clicking the **Mail us feedback on this topic!** button at the top right corner of the HTML and CHM versions.

1.4 Version history

Listing of all new features, minor improvements and bug fixes in each released version of Kentico CMS can be found at http://www.kentico.com/Support/Support-files/Version_History.

Part



Kentico CMS overview

2 Kentico CMS overview

2.1 What are the benefits of Kentico CMS?

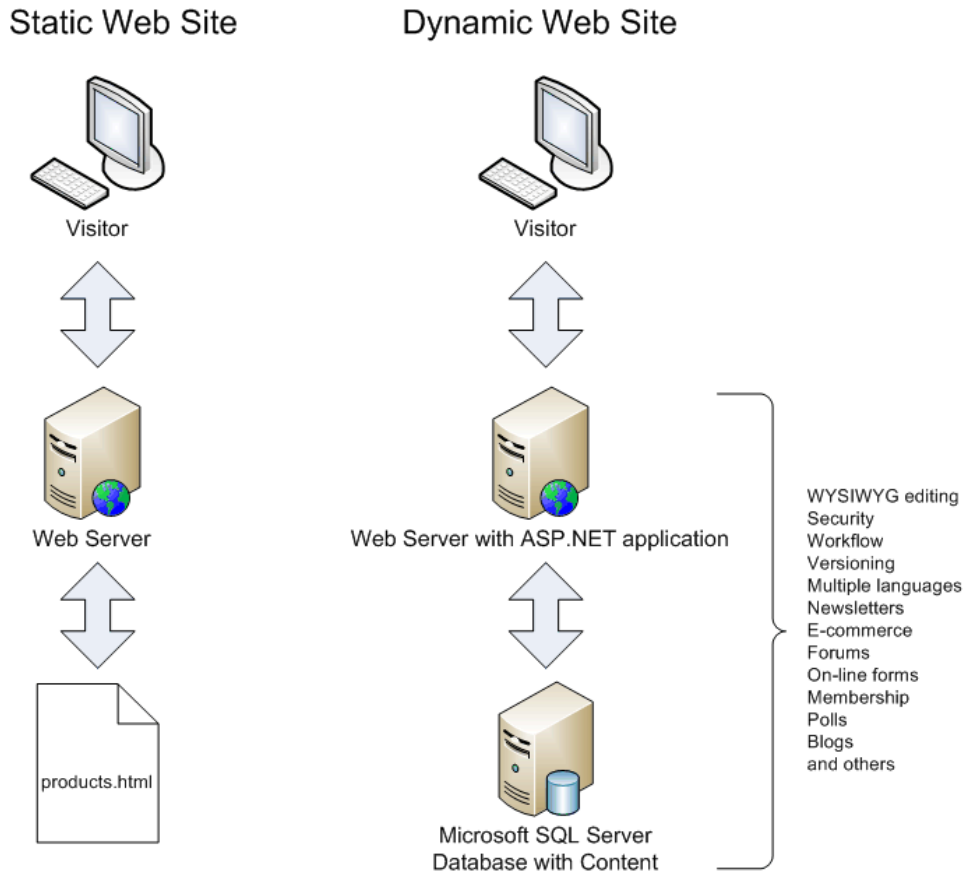
Before you dive into the details, it may be useful to understand the big picture. This chapter explains how the system works, describes its architecture and answers the most common questions you may have.

Is Kentico CMS a standard ASP.NET application?

Yes, Kentico CMS is a standard ASP.NET 3.5 SP1 application written in C#. It uses only standard functionality of .NET managed code which means you can use it on basically any ASP.NET 3.5 SP1-enabled web server without complicated configuration.

How does the CMS work?

The CMS allows you to manage content of dynamic websites. Unlike static websites that use static HTML files stored on a disk, a dynamic website displays content from the database. Kentico CMS provides both content storage and all surrounding infrastructure to manage the content and display it on the website. Kentico CMS doesn't pre-render static HTML pages; instead, it renders the content in real time, when it's requested by the visitor.



The main advantages of a dynamic website with a content management system include:

- **Easy content editing** through a WYSIWYG, browser-based interface for non-technical users
- **Content re-use** - you can display the same structured content in various ways while managing the data in one place
- **Multi-user environment** - website management is not limited to a single web developer
- **Additional functionality and applications**, such as Newsletters, E-commerce, Forums, etc.

Why should I use Kentico CMS and what benefits does it bring to me?

Kentico CMS simplifies the development of dynamic websites. Instead of developing the whole infrastructure for editing, you can utilize the flexible content management framework of Kentico CMS and focus on the site-specific functionality and design. If you consider how much time you would spend only by developing the security system, there's no doubt you should use an existing framework.

With Kentico CMS, you:

- save time and money by developing the dynamic website faster
- focus on the client's business needs instead of core infrastructure
- provide your client with additional functionality, such as Newsletters, Forums, and others that would be difficult and expensive to develop

Is it flexible enough for my needs?

Well, now you may think "If I develop the website from scratch I can create the system and enhance it at any time as my client requests." Yes, you're right, but you can do the same with Kentico CMS. Kentico CMS has been used for thousands of websites worldwide and it was designed to fit various needs of web developers and their clients. Besides, if you need to add extra functionality, you can:

- create your own modules
- add your own code to the pages
- modify default system behavior using custom handlers and providers
- customize the core engine of the system (if you purchase the source code version)

Kentico CMS was designed for the needs of web developers and their clients. You can be confident that you can **implement basically any website structure, navigation, graphic design and integrate custom functionality**.

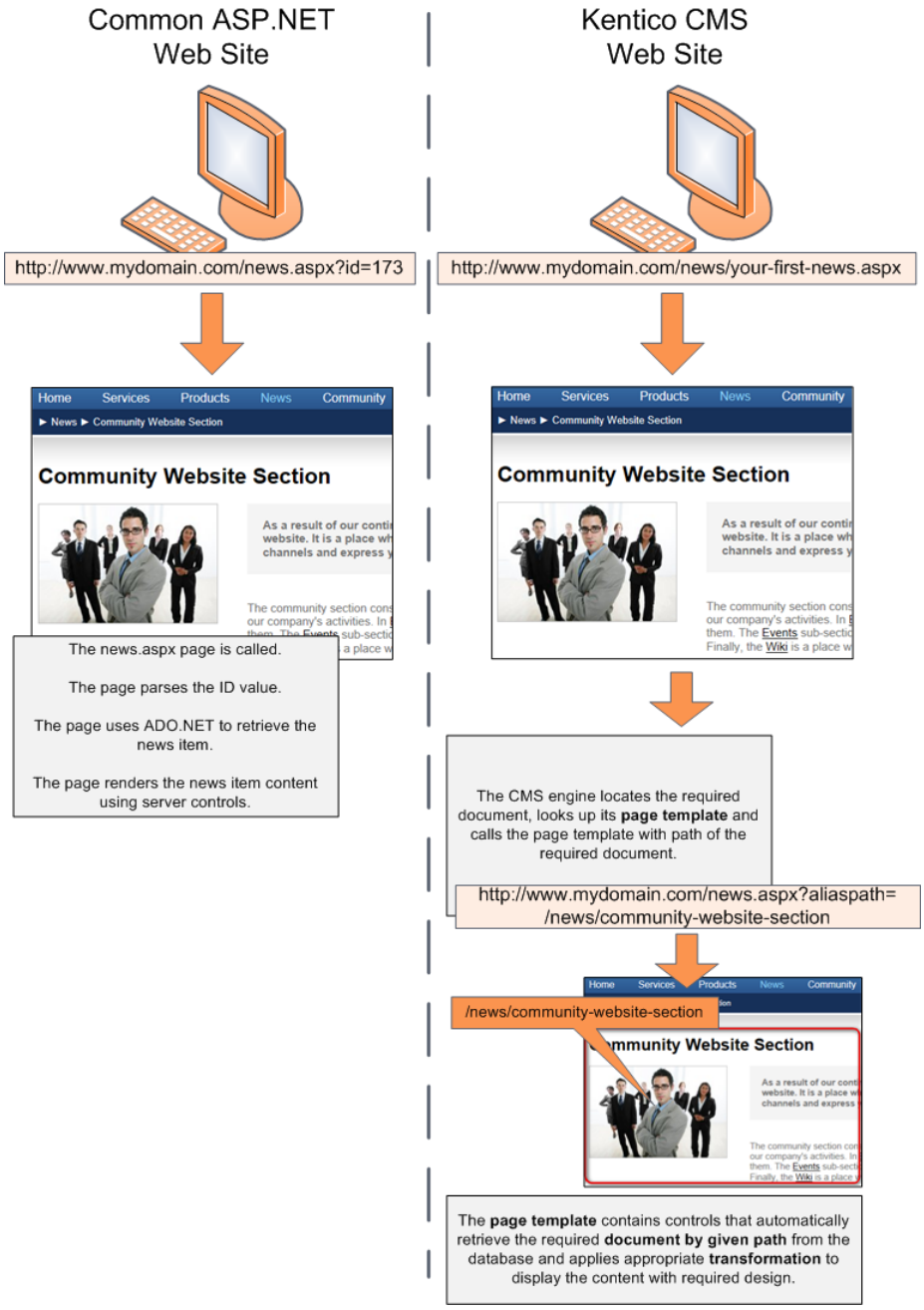
Don't take our word for it

Please visit the Kentico CMS Showcase at http://www.kentico.com/Showcase.aspx for reference websites, clients and testimonials.

2.2 How does it really work?

If you're familiar with dynamic websites, you may want to know what the difference between a CMS system and a common ASP.NET website is. Technically, Kentico CMS is just another ASP.NET web application. Its advantage is that it provides a ready-to use framework for all common tasks.

Here's a comparison of page processing in a typical ASP.NET and Kentico CMS:



But it looks more complex!

Yes, the CMS system is more complex to make your job easier. In Kentico CMS, you do not need to develop complex pages, write ADO.NET code or SQL queries.

The CMS does much of that for you and you usually only drag-and-drop controls or web parts and set their properties. You write .NET code only if you need to add additional business logic or functionality that is not supported out-of-the-box.

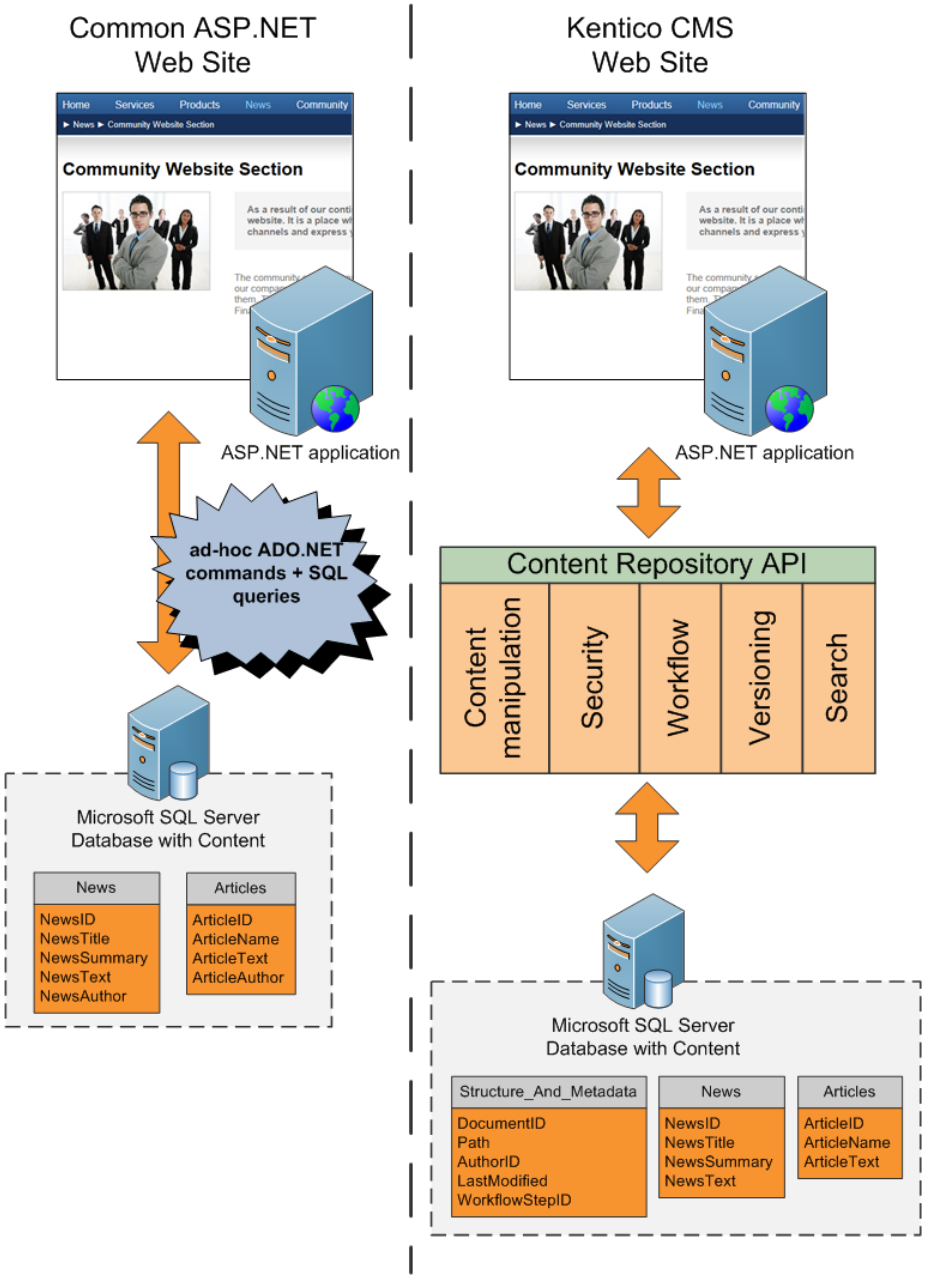
What else does the CMS do for me?

Kentico CMS provides a powerful content repository for your web content. Read the next chapter to learn more.

2.3 Where is the content stored?

Kentico CMS provides a content management sub-system, also known as **content repository**, that allows you to organize website structure and content. Moreover, it provides **a layer of security, workflow, versioning, search** and other services. All types of content can be retrieved and modified through a single **Application Programming Interface (API)**.

The following figure explains the difference between common data access approach and a content repository of Kentico CMS:



As you can see, a common approach to building dynamic websites is to write code for every page and every content type. It means that you need to write similar ADO.NET commands and SQL queries over and over again. With unified content repository, you use a complete set of API methods that allow you to save and retrieve any content type using methods someone wrote for you.

It greatly simplifies the management and retrieval of content since:

- you do not need to write your own methods, **you only call API or use built-in controls**
- **the same rules and mechanisms can be applied to any content type**, without writing additional code for every new type

An important part of the content repository is **metadata**. The metadata includes:

- content organization into a tree hierarchy that also represents the site map
- information about content authors and modifications
- workflow-related details, such as the current workflow step
- content expiration
- permissions for the given document

In the example above, you can see that with the classic approach, both News and Articles tables contain the attribute Author. In the content repository, the author is stored in shared metadata for all documents, regardless of their type.

Please note

Metafiles and attachments are stored either in the database, or the server's file system under a folder specified in the **Site Manager -> Settings -> System -> Files -> Files folder** setting. If this path is not set:

- metafiles of objects not connected with a particular site (global objects) are stored under <*web root*>/*CMSFiles*
- metafiles of objects connected with a particular site are stored under <*web root*>/<*site name*>/*metafiles*
- attachments (always connected with a particular site) are stored under <*web root*>/*site name*>/*files*

As you can see, the **content repository represents a systematic approach to content storage, manipulation and security**.

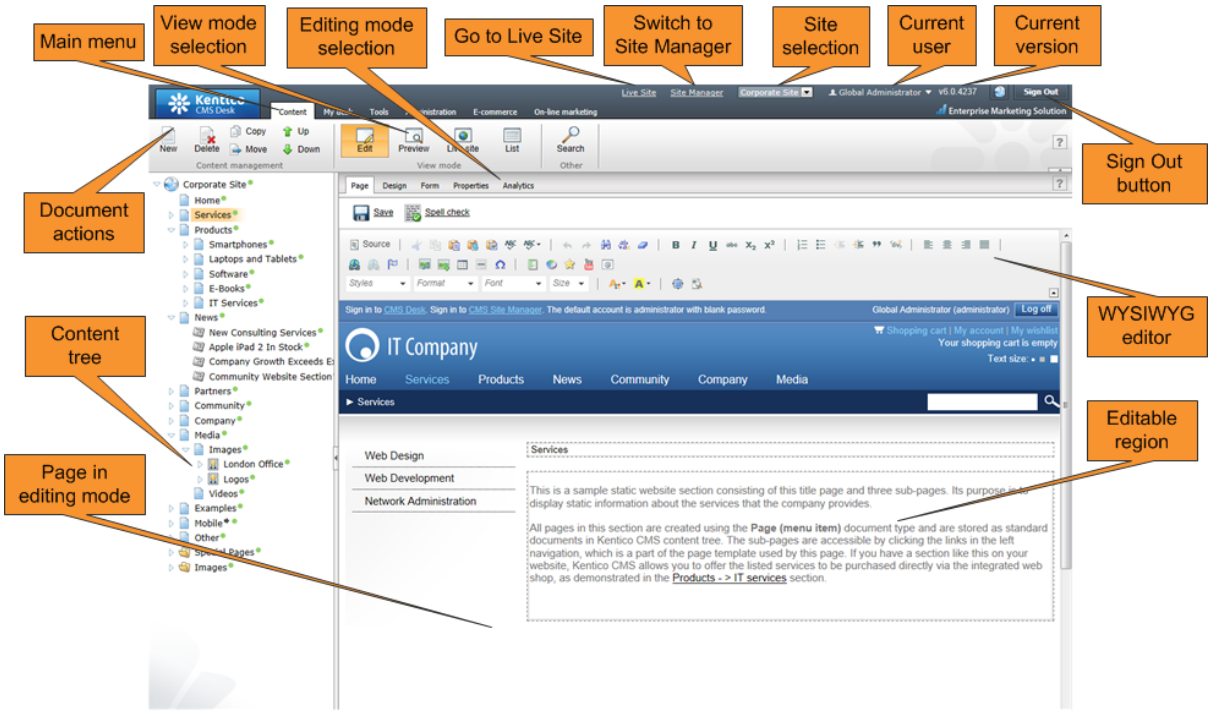
2.4 How do I edit content?

Until now, we have discussed how the system works and its architecture. But how do I edit the content? After all, it's the most important feature of a content management system.

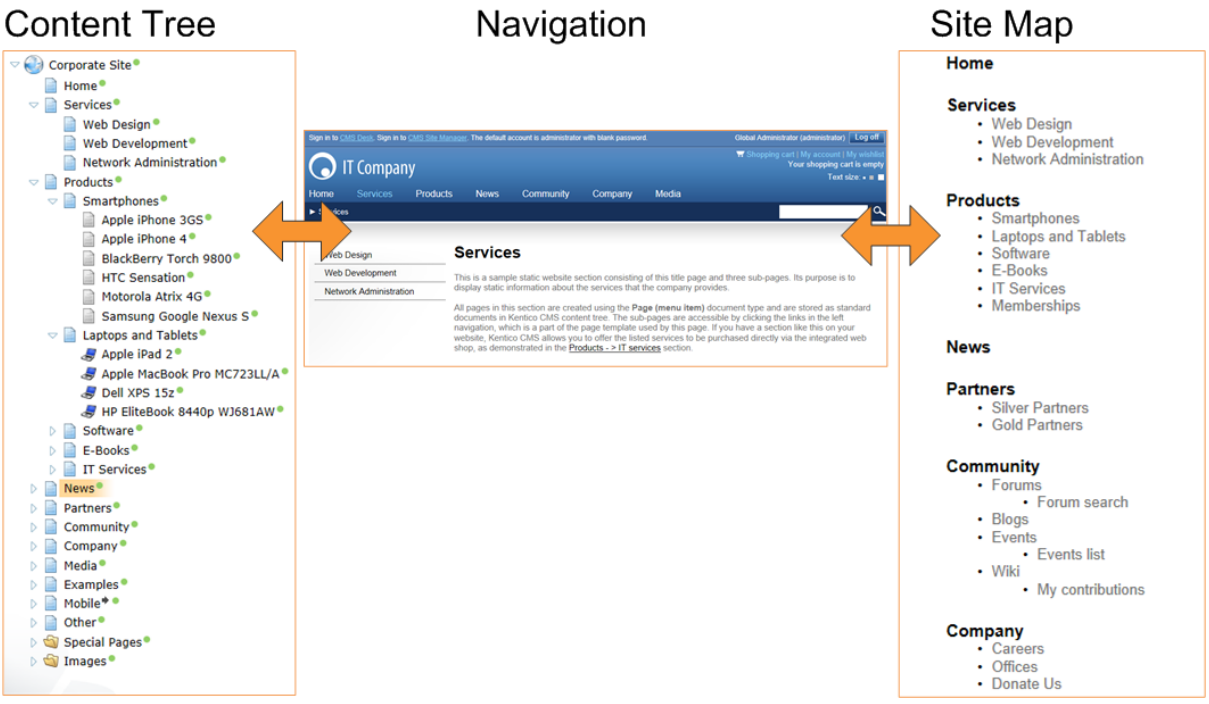
Kentico CMS comes with a browser-based interface, which is divided into several parts:

- **CMS Desk** (*http://www.example.com/cmsdesk*) - the main user interface for the management of a specific website, intended for content editors, website designers and marketers.
- **On-site editing mode** (*http://www.example.com/cmsedit*) - a convenient editing interface that may be used to make content adjustments directly while viewing the live site.
- **CMS Site Manager** (*http://www.example.com/cmssitemanager*) - a global system administration interface, intended for top-level administrators and web developers.

The following figure shows an overview of the CMS Desk user interface.



The **content tree** on the left allows you to browse the content and choose the document you want to edit. The content tree also represents a site map of the site and it's used for rendering navigation. In the following figure, you can see how the content tree, navigation and site map fit together:



The **action menu** allows you to create, delete, copy, move and change the order of documents.

The **view mode** allows you to switch between the following views:

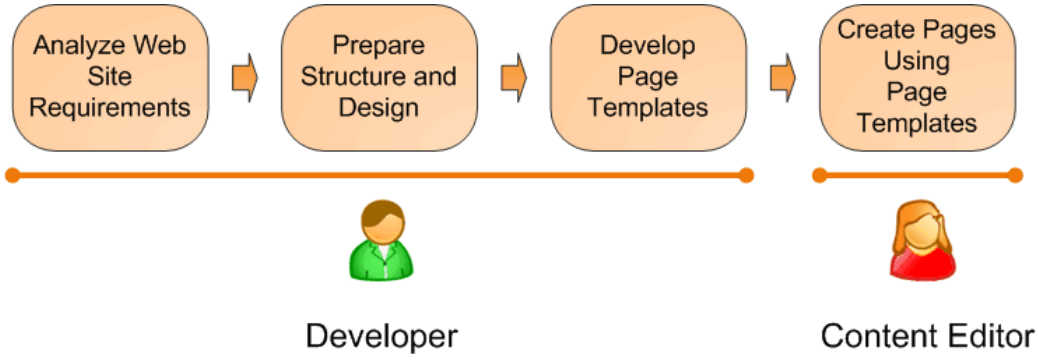
- **Edit** - editing mode.
- **Preview** - a preview mode that displays the current version of the page, even before it is published to the live site. It also shows content without using caching, which allows you to preview the content even if the live website displays the cached version.
- **Live site** - allows you to view exactly how the page appears on the live site while still remaining in CMS Desk.
- **List** - displays a list of child documents under the currently selected document. It may be used to perform mass actions for multiple documents simultaneously and can also be useful if there are too many documents in a given section and cannot be browsed comfortably in the content tree.

The **editing area** allows you to edit the content and metadata of the document selected in the content tree. You can choose from the following tabs/editing modes:

- **Page** - here you can edit the content of editable regions that are used for unstructured content. Additionally, any document can also have structured content that may be edited in the editing form on the *Form* tab.
- **Design** - used to modify the main layout of the page, as well as fixed content and functionality added through web parts (this applies to the portal engine development model that will be described later in this guide). This tab is only available for designated developers and global administrators.
- **Form** - here you can modify the structured data of the document, such as a news title, news summary, release date, etc.
- **Product** - here you can modify the product specification of documents representing products that can be added to the shopping cart (when using the e-commerce module).
- **Properties** - here you can modify various settings, permissions, metadata and design options of the document.
- **Analytics** - may be used to view statistical data measured for the document when using the built-in web analytics, and set up advanced on-line marketing functionality.

2.5 How do I develop a website?

Now that you know how to edit the content, you may want to know how to develop the website and manage the design. Although these topics will be described later in this guide, let's take a quick look at the general development process:



This figure shows how you develop the website and how the roles are split between developers and editors. The typical development process consists of the following steps:

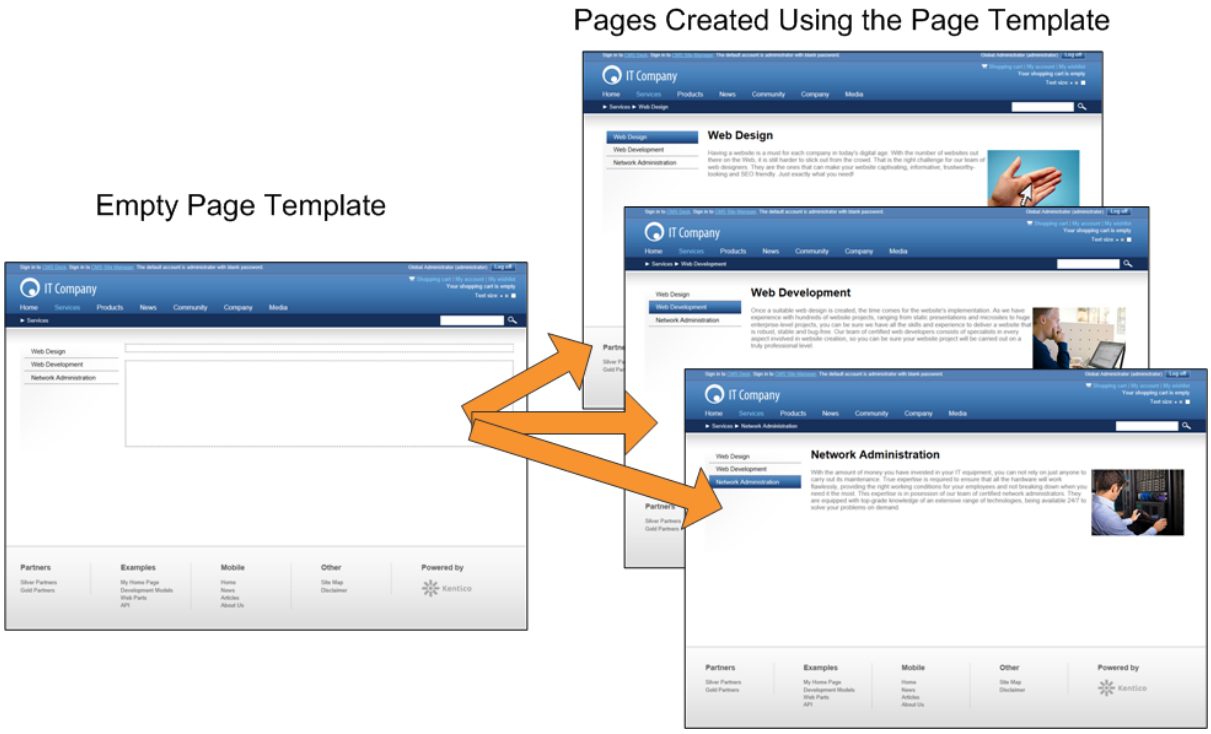
1. The developer analyzes the client requirements.
2. The developer prepares the site structure (site map) and web design.

3. The developer creates **page templates** for every type of the page (home page, solutions, products, news, etc.)
4. The content editor creates new pages - they enter text and images into the page templates defined by the developer.

What is a page template?

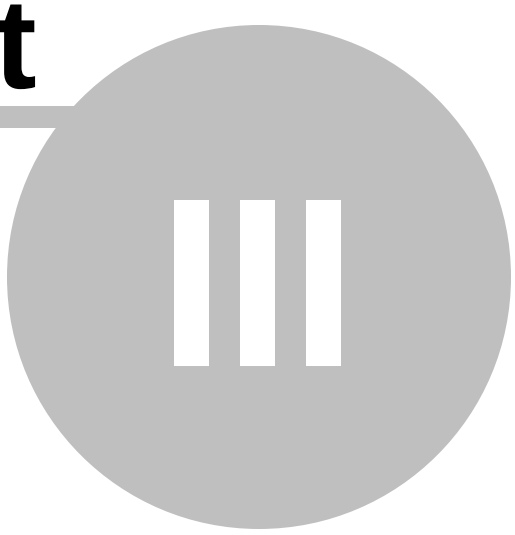
A page template is a predefined look for pages that allows content editors to enter the content. A single template can be re-used for multiple pages with the same structure and design, but with different content.

Templates allow content editors to focus on content editing, without taking care of the page formatting. They also help keep the web design consistent throughout the website. The following figure shows how a single page template can be used for multiple pages:



The details on page template development are described later in this guide.

Part



Installation and deployment

3 Installation and deployment

3.1 Overview

Before you begin with the installation procedure, please check whether your machine meets the System requirements.

New Kentico Installer

This section describes the installation of Kentico released before 5/2013. If you are looking for the new installation type documentation, see the Kentico CMS 7 Installer Documentation.

Old Kentico Installer

The old version of the installer that this documentation describes has been deprecated. However, if you need to use the old installer, download it here.

Kentico CMS installation consists of several steps depending on the installation type:

Installation on the local development machine

1. Setup (KenticoCMS.exe) - installs only the program files. No changes to the system configuration (registry, IIS or SQL Server) are made. You do not need to run the setup on your production server - it's intended primarily for development machines.
2. Web Installer - creates a new website project and optionally configures IIS or uploads the project to the server using FTP. Again, you do not need to run the Web Installer on your production server - you can run it locally and deploy the files over FTP.
3. Database setup - runs in the web user interface. It creates a new database on your SQL Server with system tables and basic data.
4. New site wizard - runs in the web user interface after you create a new database. It allows you to create a new site managed by Kentico CMS.

Installation on a remote (production) server

On the remote (production) server, it's not necessary to run any executable or register DLL libraries. Unless you have full administrative access to the server, you will typically follow these steps:

1. Install Kentico CMS on your development machine using Setup (KenticoCMS.exe).
2. Run the Web Installer on your development machine. Choose to install Kentico CMS on a Remote server and choose a temporary folder on your local disk.
3. Copy the files from the temporary folder to the production server (e.g., over FTP). If the files are not copied directly to the root of the website, you will need to create a virtual directory. See Additional configuration tasks -> Creating a virtual directory for details.
4. Open a web browser and navigate to the root URL of the copied files on your web server.

The rest of the installation is the same as on the local machine (Database setup and New site wizard).

If you encounter any problems during the installation, please see the Troubleshooting installation issues chapter or contact our support at http://www.kentico.com/Support.aspx

Tip: There's no magic behind it!

Kentico CMS is a **standard ASP.NET application**. Since it doesn't make any modifications to the system, you can move it to another system as you would with any other ASP.NET project. You can also open the project in Visual Studio and debug it or compile it.

The database is a standard MSSQL database, so you can move it to another server using a typical backup/restore procedure. The connection string is stored in the *web.config* file, in the *ConnectionStrings/CMSConnectionString* element.

Silent Install

If you want to install Kentico CMS in the background without having to interact with the installation interface, navigate directly to the Silent Install topic.

3.2 System requirements

The following configurations are supported by Kentico CMS. Other configurations have not been tested, but may be functional.

Server-side Requirements

- Windows Vista Home Premium/Business/Enterprise/Ultimate, Windows 7 (both 32bit and 64bit) or Windows 8 (both 32bit and 64bit), or Windows Server 2008, 2008 R2, 2012.
- Microsoft .NET Framework 3.5 SP1 or 4.0 or higher.
- Microsoft Internet Information Services (see the table below) or Visual Studio/Visual Web Developer 2008/2010/2012 built-in web server.
- Microsoft SQL Server 2005, 2008, 2008 R2, 2012 (including free SQL Server Express Edition 2005/2008/2012).

Internet Information Services overview

Internet Information Services version	Operating system	Details and installation instructions
IIS 5.1	Windows XP Professional	not supported
IIS 6.0	Windows Server 2003	not supported
IIS 7.0	Windows Vista	IIS 7 Installation and Deployment
	Windows Server 2008	
IIS 7.5	Windows 7	

	Windows Server 2008 R2	
IIS 8.0	Windows 8	Installing IIS on Windows Server 2012
	Windows Server 2012	

It's recommended to install IIS with all its available features.

Hosting Requirements

- ASP.NET 3.5 SP1 (or higher) and Microsoft SQL Server 2005/2008/2012 support.
- Medium-trust or full-trust permissions for the ASP.NET application.
- If the server uses medium trust, ASP.NET AJAX 1.0 must be installed on the server.
- If the application uses .NET Framework 3.5 SP1 and is hosted in a medium trust environment, it is necessary to have Microsoft Chart Controls installed on the server.
- It's recommended that your hosting plan comes with 250 MB or more memory and 100+ MB database.

You can use your favorite hosting provider or choose from our hosting partners.

Windows Azure

Kentico CMS fully supports the Microsoft Windows Azure platform, including SQL Azure, Azure Storage and other services. Windows Azure SDK 1.7 is required.

Development Tools

If you want to create custom web parts or integrate custom code, you need **Visual Studio 2008/2010/2012** or free **Visual Web Developer 2008/2010/2012 Express Edition**.

External utilities

Kentico AD Import Utility, Kentico Import Toolkit, Kentico Installation Manager, Kentico Hotfix and Upgrade utility and Kentico Service Manager require **.NET Framework 3.5** to run.

Supported Client Browsers for Content Editors

- Internet Explorer 8, 9
- Firefox 4.0+
- Chrome 12
- Safari 4.0+ or Firefox 4.0+ on Mac OS

Supported Client Browsers for Site Visitors

- Internet Explorer 6.0+
- Firefox 1.0.5+
- Chrome 12+
- Mozilla 1.7.1+
- Netscape 7.1+

- Opera 7.52+
- Safari or Firefox on Mac OS
- Mobile browsers, such as Safari on iPhone, are supported as well, but some features may be limited by browser capabilities.

(the visitor browser requirements also depend on the functionality used on the website)

3.3 Installation procedure

3.3.1 Setup (KenticoCMS.exe)

New Kentico Installer

This section describes the installation of Kentico released before 5/2013. If you are looking for the new installation type documentation, see the Kentico CMS 7 Installer Documentation.

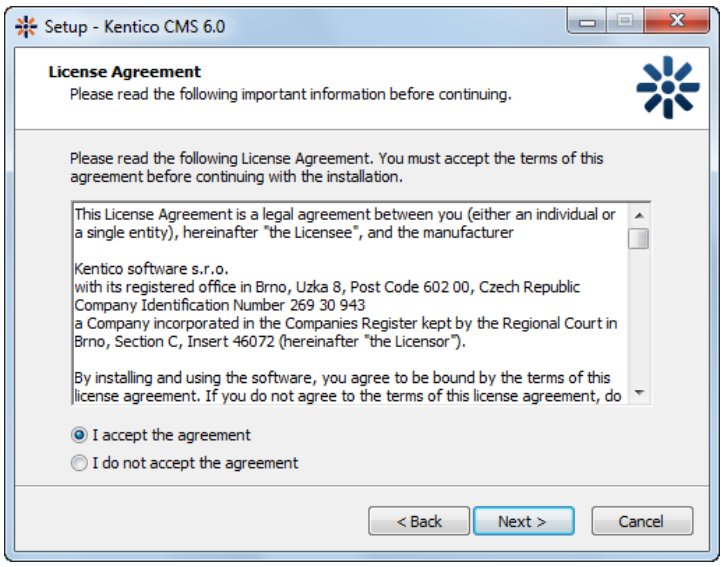
Use this file to install the program files on your development machine.

Installation on a shared hosting server

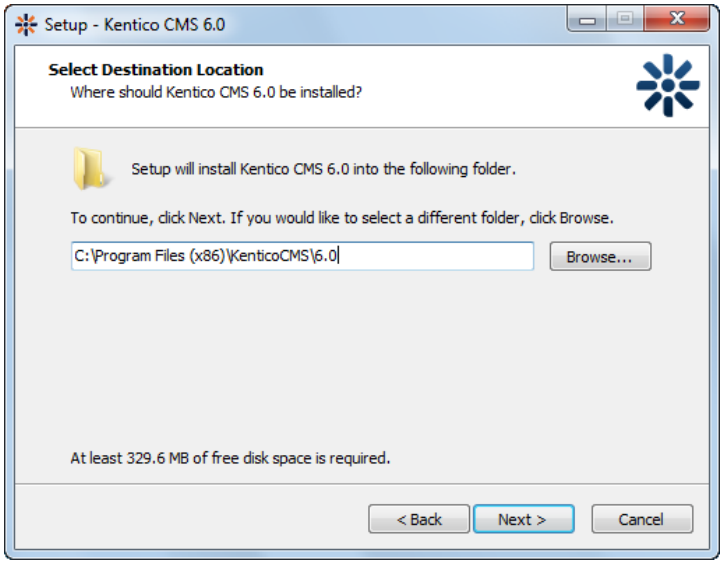
If you're going to run Kentico CMS on a shared hosting server, you do not need to run any EXE file or register any DLL file on the server (you're usually not allowed to do that anyway). Please read Additional configuration tasks -> Installation on shared hosting server to find out how to solve this.

1. Run *KenticoCMS_<version>.exe*. You will see the welcome screen.

2. Click **Next**.
3. Read and accept the license agreement and click **Next**.



4. Choose the location where the Kentico Web Installer and documentation will be deployed.
5. Click **Next** and **Install**.
   - Please note: this is not the folder where your website will be placed, it's only a place for Kentico CMS program files and help files.



6. After the installation is finished, enable the **Launch Kentico Web Installer** option and click **Finish**.
7. Continue to the Web Installer chapter.

3.3.2 Web installer

3.3.2.1 Overview

New Kentico Installer

This section describes the installation of Kentico released before 5/2013. If you are looking for the new installation type documentation, see the Kentico CMS 7 Installer Documentation.

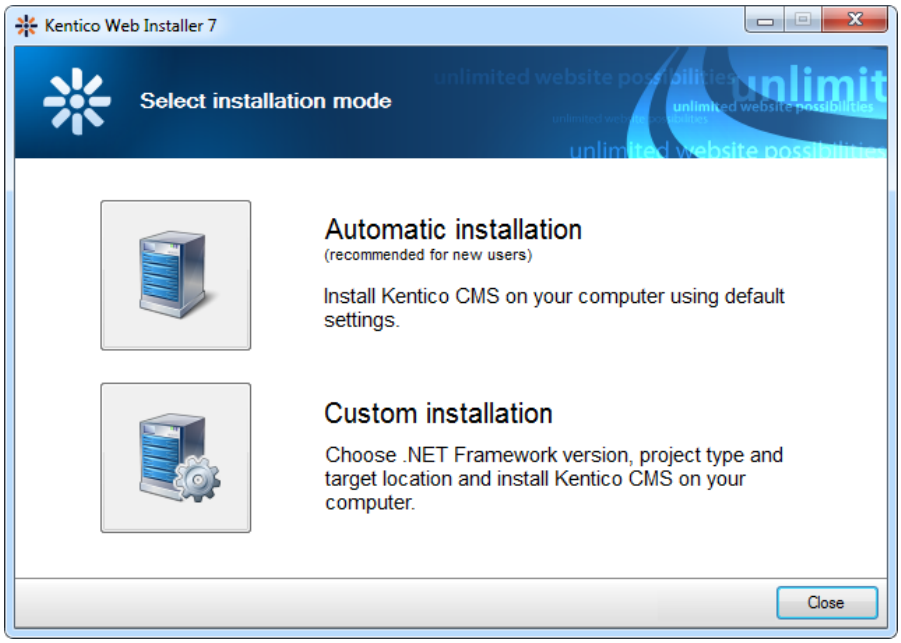
Kentico Web Installer allows you to create a new project and (optionally) configure the Microsoft IIS web server.

Selecting installation mode

You have two ways to proceed with the installation:

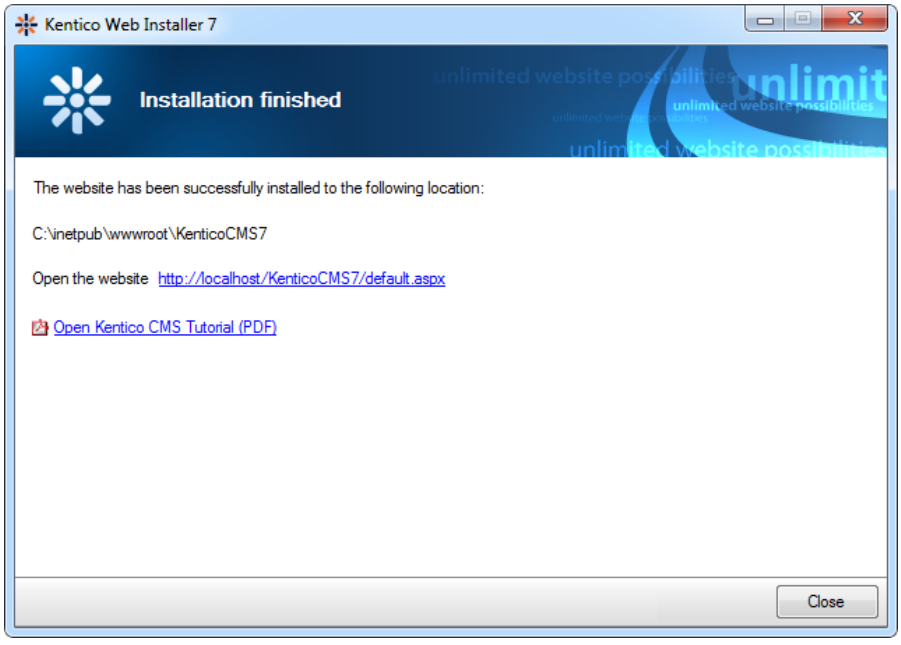
- Automatic installation - uses default settings. Recommended if you are new to Kentico and want to, e.g., evaluate the functionality.
- Custom installation - allows you to choose the .NET framework, as well as the type of the project and the installation location folder.

Note that **Automatic installation** chooses .NET framework based on the framework that is available in your system, first checking for .NET 4.0, .NET 4.5 and finally 3.5. *Web site project* and default IIS *inetpub\wwwroot* installation folder are selected.



Automatic installation

If you clicked Automatic installation, the installer copies the necessary files to the default folder and displays a link to view your website.



Continue to the Database setup topic.

Custom installation

If you clicked **Custom installation**, the wizard continues through the following steps.

Step 1 - Select .NET Framework version

First, you need to choose whether you use:

- **.NET Framework 3.5 and Visual Studio 2008**
- **.NET Framework 4.0 and Visual Studio 2010**
- **.NET Framework 4.5 and Visual Studio 2012**

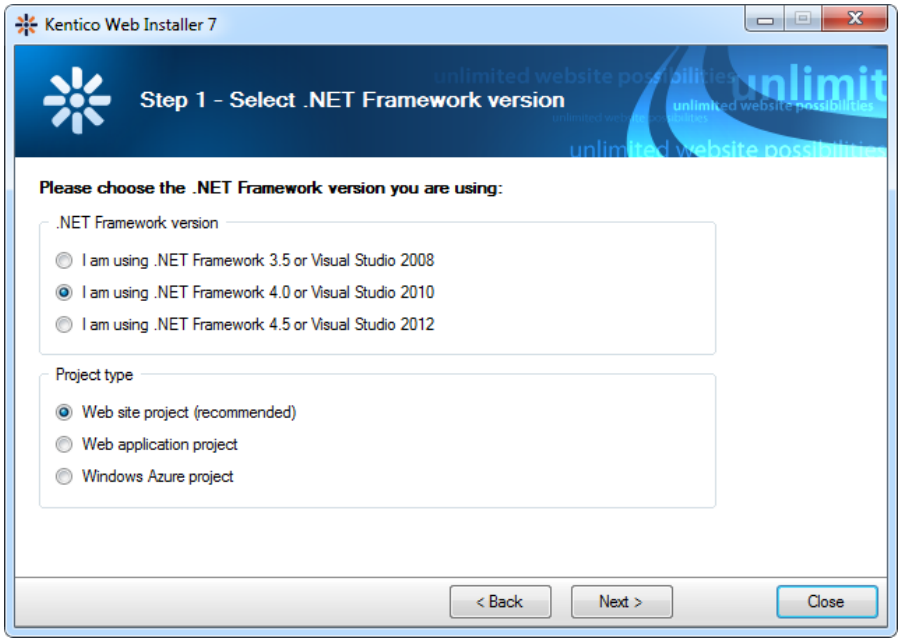
Depending on your choice, the installer will use appropriate *web.config* and *webproject.sln* files. The binaries and code are the same for both options; they are compiled for .NET 3.5 SP1 and can be used with .NET 4.0 as well.

Then, you need to select from the following options:

- **Web site project (recommended)**
- **Web application project**
- **Windows Azure project**

This determines which Visual Studio project type the web installer will create. The name of the solution file will be *WebProject.sln* for a web site project or *WebApp.sln* for a web application.

The last option creates a web application suitable for deployment to the Windows Azure Platform. For more information about this type of installation, please refer to the Windows Azure Deployment Guide.

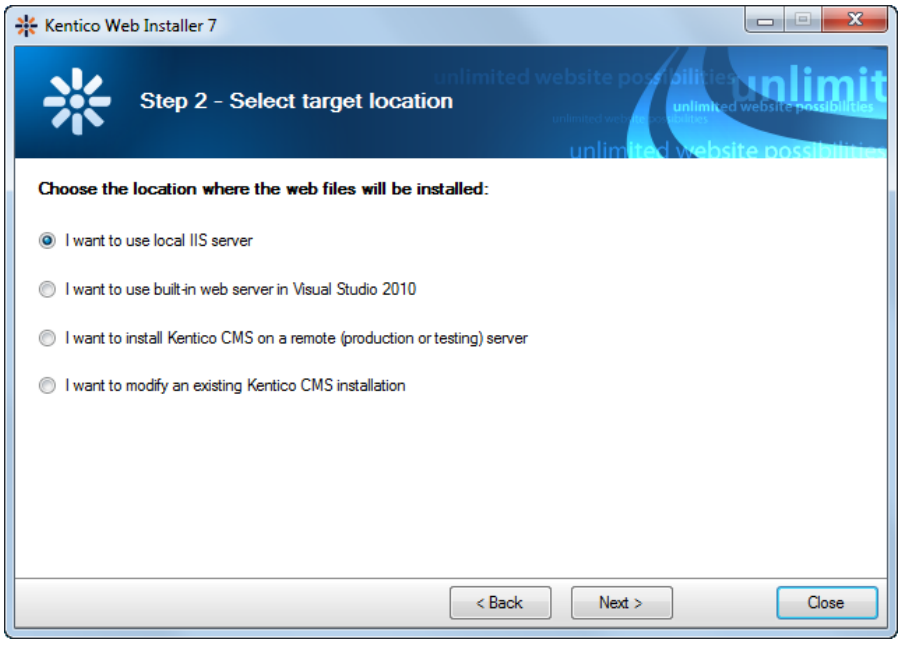


Step 2 - Select target location

Choose one of the options. Click the link to view instructions for the selected option.

- I want to use local IIS server - you must have local IIS server installed, running and configured for the version of ASP.NET which you chose in the previous step

- I want to use built-in web server in Visual Studio - you must have Visual Studio or Visual Web Developer Express Edition installed on your local machine
- I want to install Kentico CMS on a remote (production or testing) server - this option only copies the project files to a temporary folder on your disk and you need to copy the files to your production server manually (e.g., over FTP)
- I want to modify existing Kentico CMS installation - this option modifies (adds or removes components) an existing installation on a local machine



3.3.2.2 Local IIS server

If your machine is running the IIS server locally, choose **I want to use local IIS server** in Step 2 of the Web installer and click **Next**.

Step 3 - Local IIS

Specify the following information:

- **Choose website**: choose one of the websites configured on your IIS; please make sure that the website you choose is running
- **Type of application pool**: type of application pool which will be used for the website (*Integrated* by default, read this article for more information)
- **Choose target folder**: choose the disk folder where the web project files will be placed

If you're installing Kentico CMS into the root of your website (such as *http://www.example.com*) and do not wish to create another virtual directory (such as *http://www.example.com/cms*), check the **This is an installation to the root** check-box.

Installation to the root

Check this option (use the website's root virtual directory for the application):

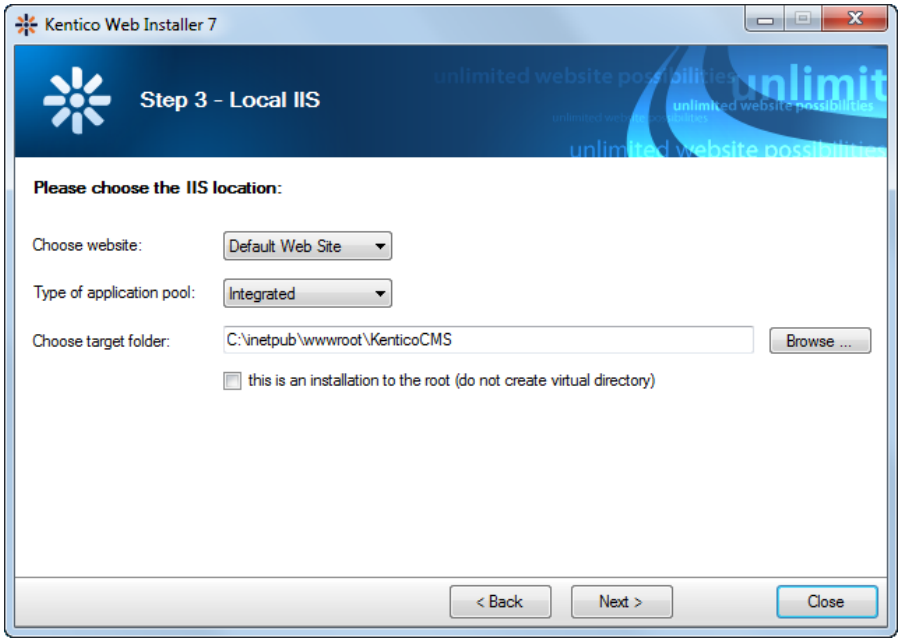
- If you want to have only one application under the website.
- If you want to set different bindings for your applications (to have your applications under different domains and IPs). You can set bindings only for websites, so you will have to use multiple websites.
- When you want to deploy your web application to a hosting server, as it might be easier to export the IIS settings this way.

Uncheck this option (create a virtual directory):

- If you want to install more applications under one website (best for developing purposes).

Concerning Kentico, there are not any differences between these two options.

You can find more information about virtual directories in this article: Understanding Sites, Applications, and Virtual Directories on IIS 7.



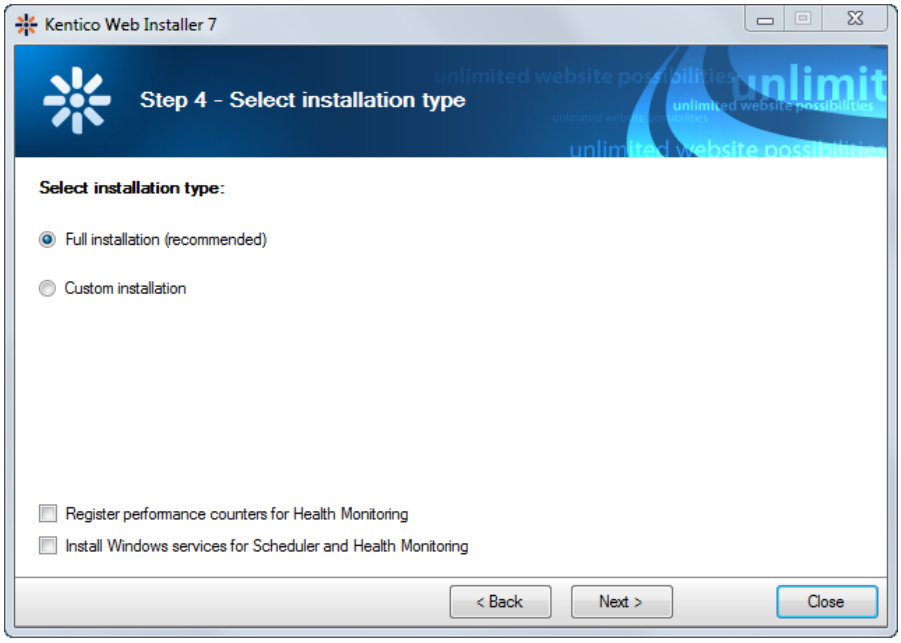
Step 4 - Select installation type

Decide between the following options:

- **Full installation** - performs full installation using all optional components
- **Custom installation** - lets you choose which components will be installed

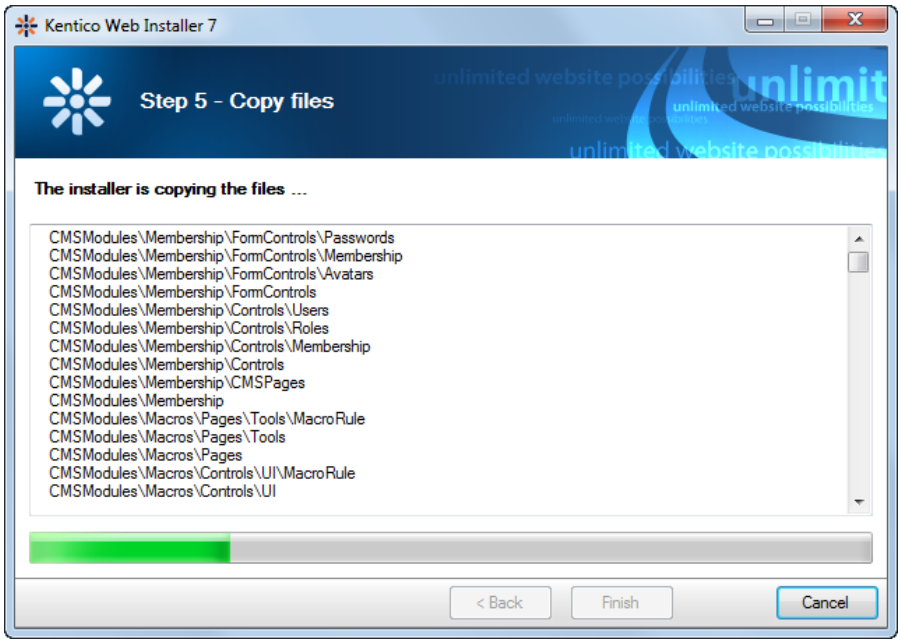
Enabling the **Register performance counters for Health Monitoring** option registers performance counters for the currently installed instance of Kentico CMS in Windows. The counters can be used for monitoring of system load and performance in an external application.

Checking the **Install Windows services for Scheduler and Health Monitoring** option installs the **Kentico CMS Scheduler** and **Kentico CMS Health Monitor** services.
Please refer to Modules -> Health monitoring for more details on these features.



Step 5 - Copy files

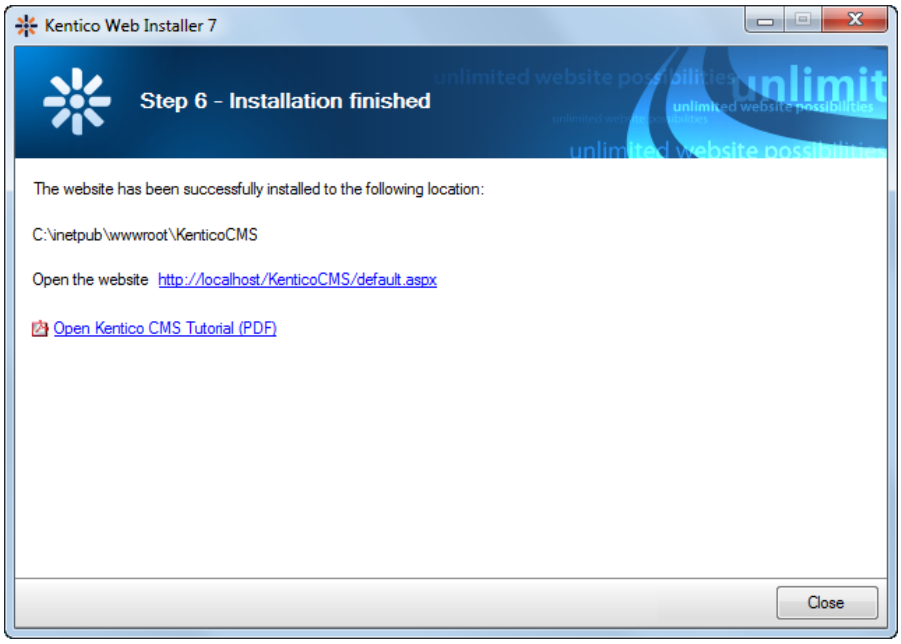
The installer copies the project files to the specified folder:



Step 6 - Installation finished

On the final page of the installer you can click on the displayed link to open the website in your default

browser.



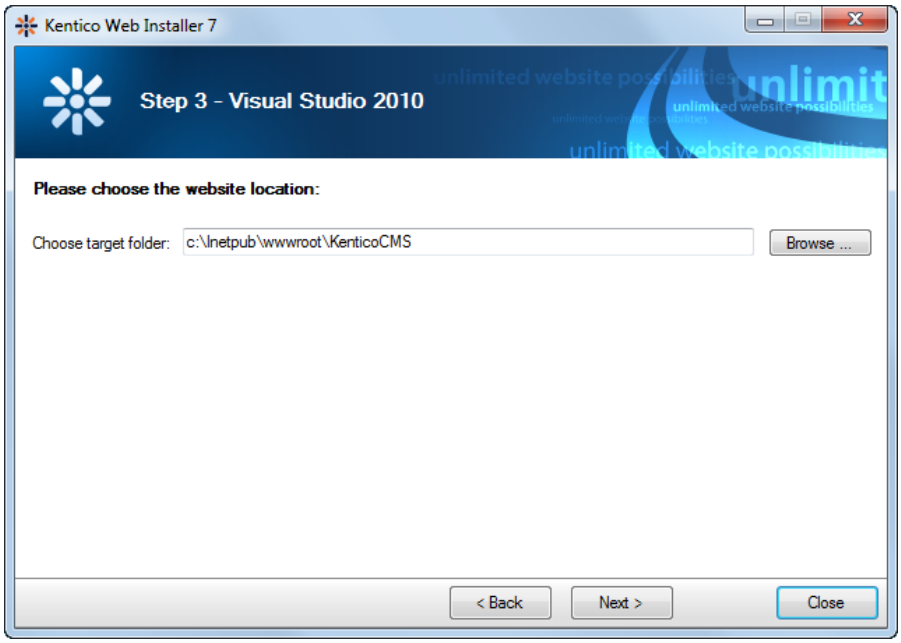
Continue to the Database setup topic.

3.3.2.3 Built-in web server in Visual Studio

If you want to use the built-in server which is included in Visual Studio, choose **I want to use built-in web server in Visual Studio** in Step 2 of the Web installer and click **Next**.

Step 3 - Visual Studio 2010

Choose the target location of the files on your disk and proceed by clicking **Next**.

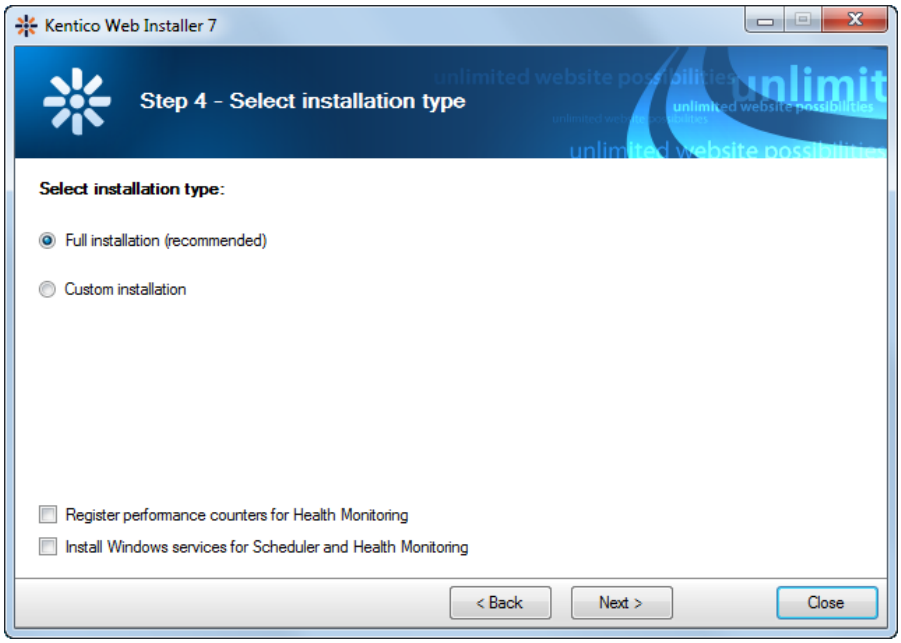


Step 4 - Select installation type

Decide between the following options:

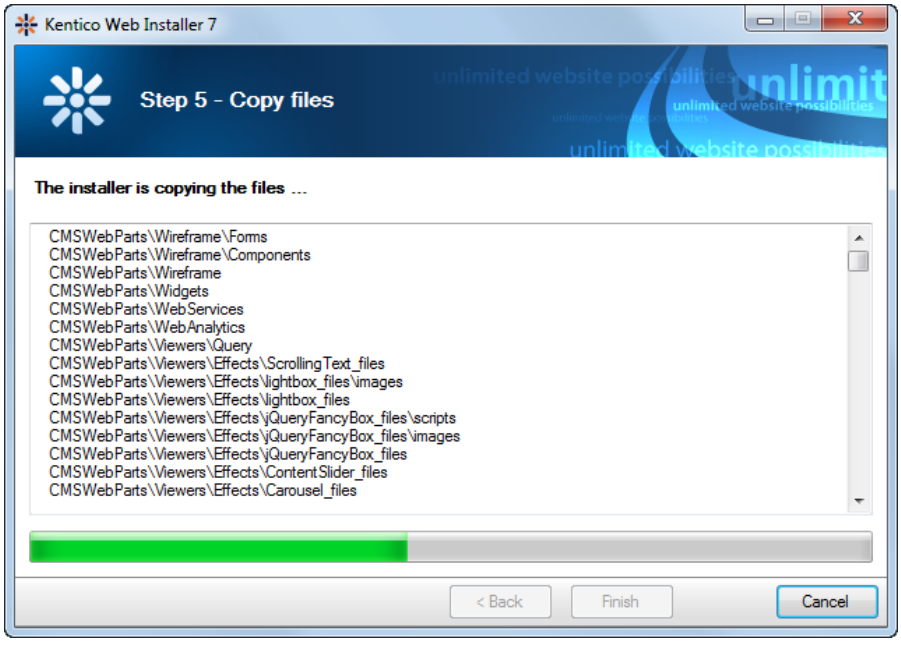
- **Full installation** - performs full installation using all optional components
- **Custom installation** - lets you choose which components will be installed

Enabling the **Register performance counters for Health Monitoring** option registers performance counters for the currently installed instance of Kentico CMS in Windows. The counters can be used for monitoring of system load and performance in an external application.
Checking the **Install Windows services for Scheduler and Health Monitoring** option installs the **Kentico CMS Scheduler** and **Kentico CMS Health Monitor** services.
Please refer to Modules -> Health monitoring for more details on these features.



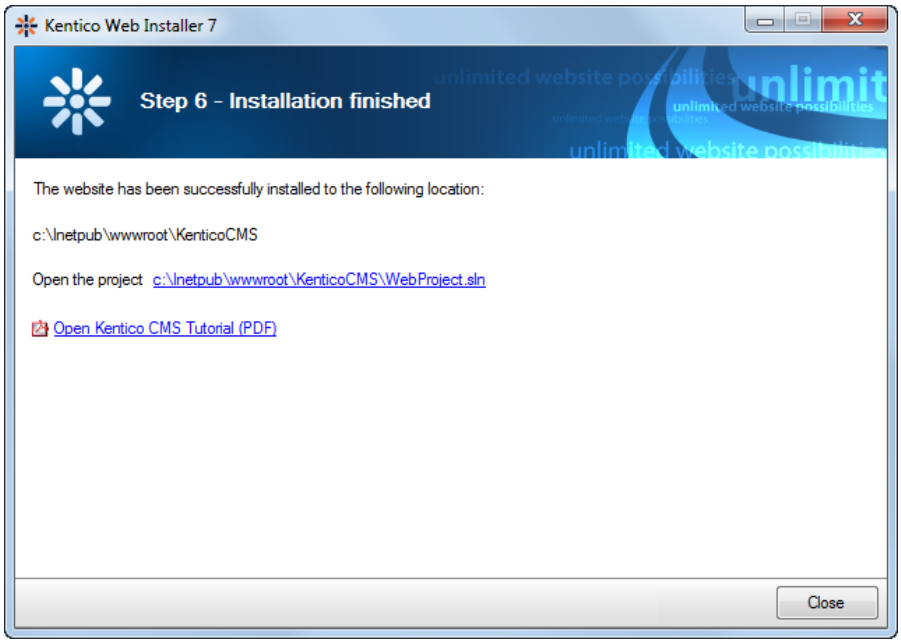
Step 5 - Copy files

The installer copies the project files to the specified folder:

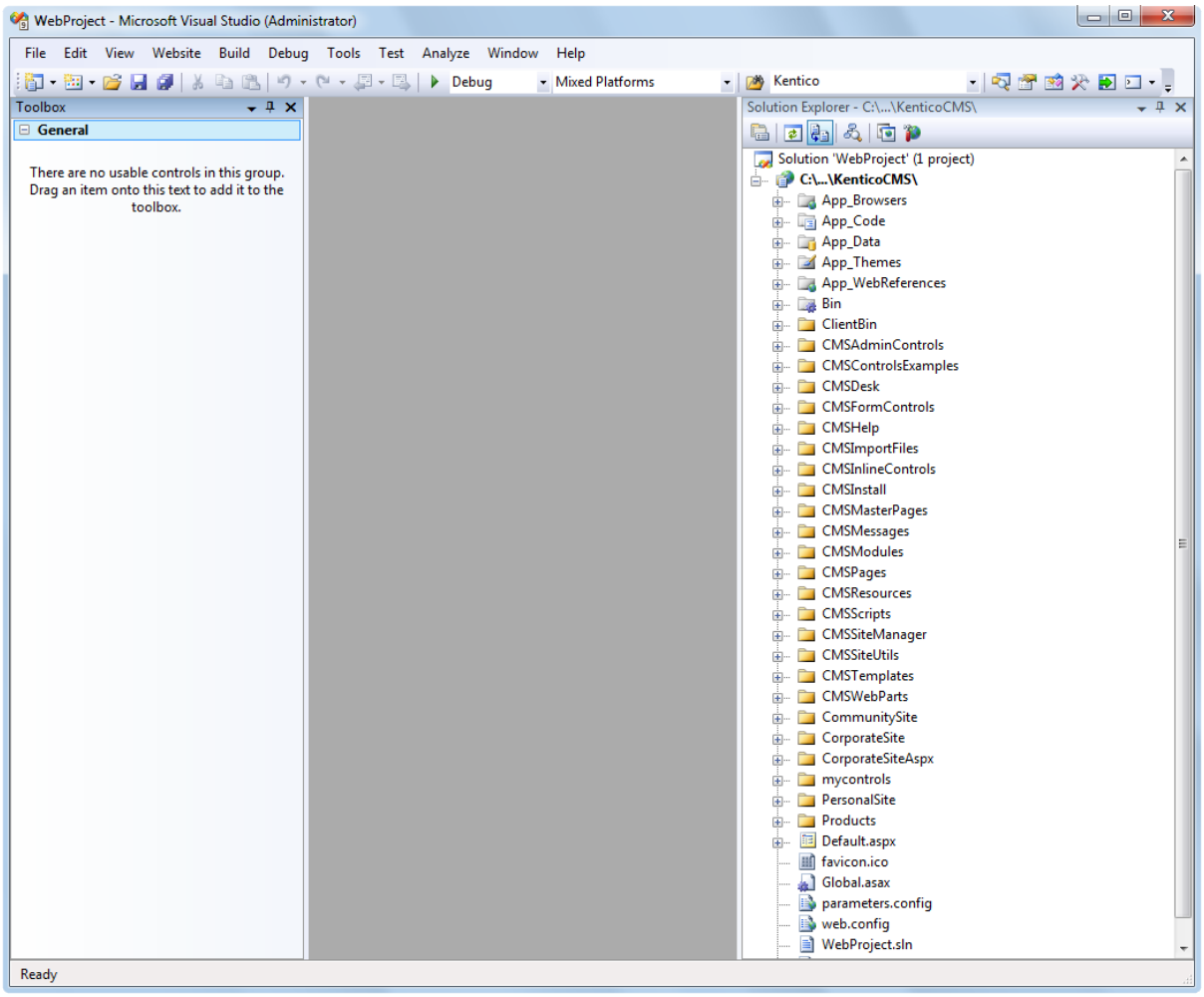


Step 6 - Installation finished

Once the files have been copied to the folder, you can open the solution in Visual Studio by clicking the link:



The project opens in Visual Studio:



Choose **Debug -> Start without debugging** from the main menu. The site is displayed in the new browser window, using the built-in VS web server.

When you cannot open the website in Visual Studio

If the link for opening the project in Visual Studio doesn't work, you may need to start Visual Studio manually. Then choose **File -> Open -> website...** from the main menu and locate the project folder on your disk manually.

Continue to the Database setup topic.

3.3.2.4 Remote (production or testing) server

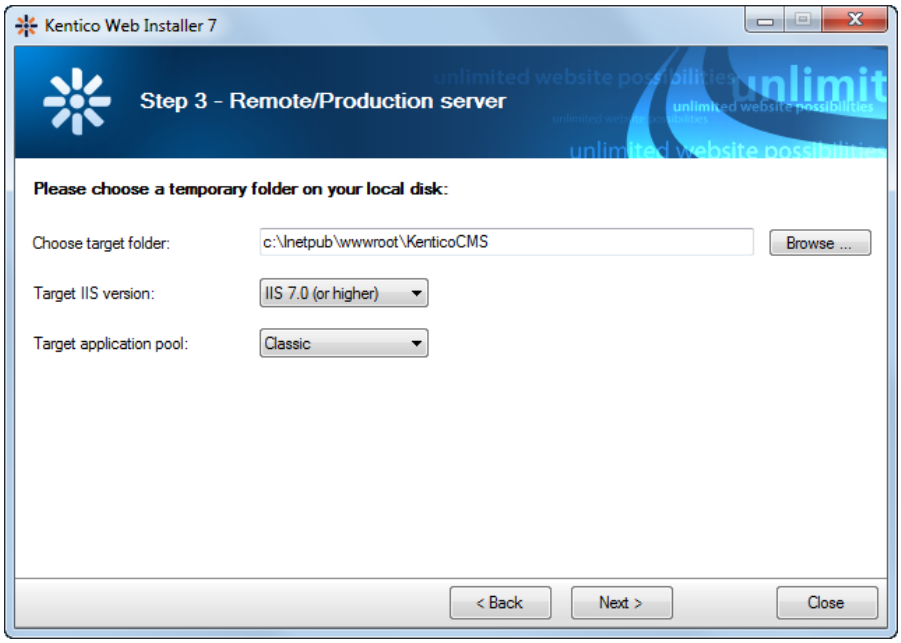
If you need to install Kentico CMS on a remote web server where you cannot run the setup directly (e.g., a shared hosting server), you need to choose the **I want to install Kentico CMS on a remote (production or testing) server** option in Step 2 of the Web installer.

Step 3 - Remote/Production server

Specify the following:

- **Choose target folder** - a temporary folder on your local disk where the web project will be created
- **Target IIS version** - version of IIS installed on the remote server where you want to deploy the CMS
- **Target application pool** - type of the target application pool on the remote server where the CMS should be running

Click **Next** to proceed through the rest of the wizard.

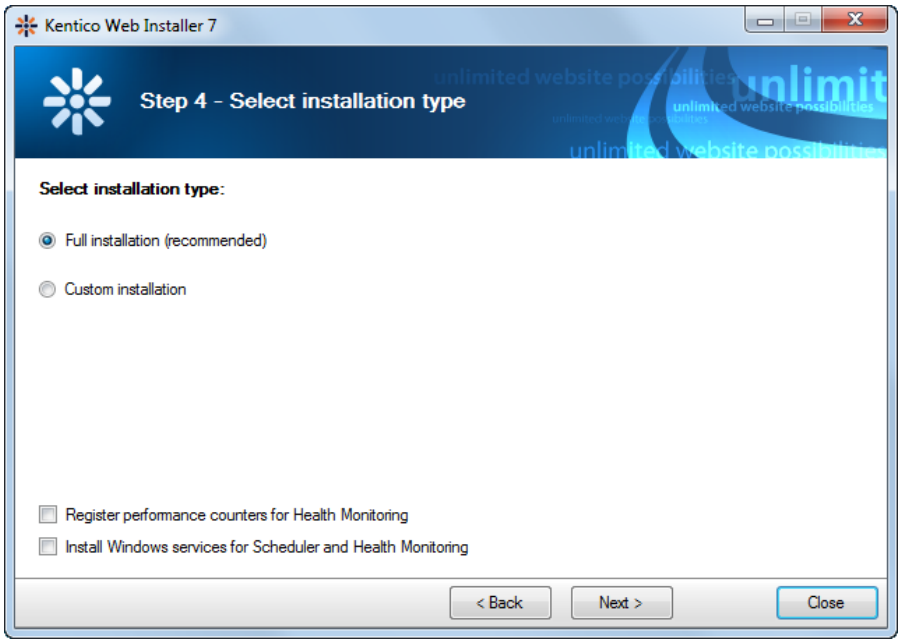


Step 4 - Select installation type

Decide between the following options:

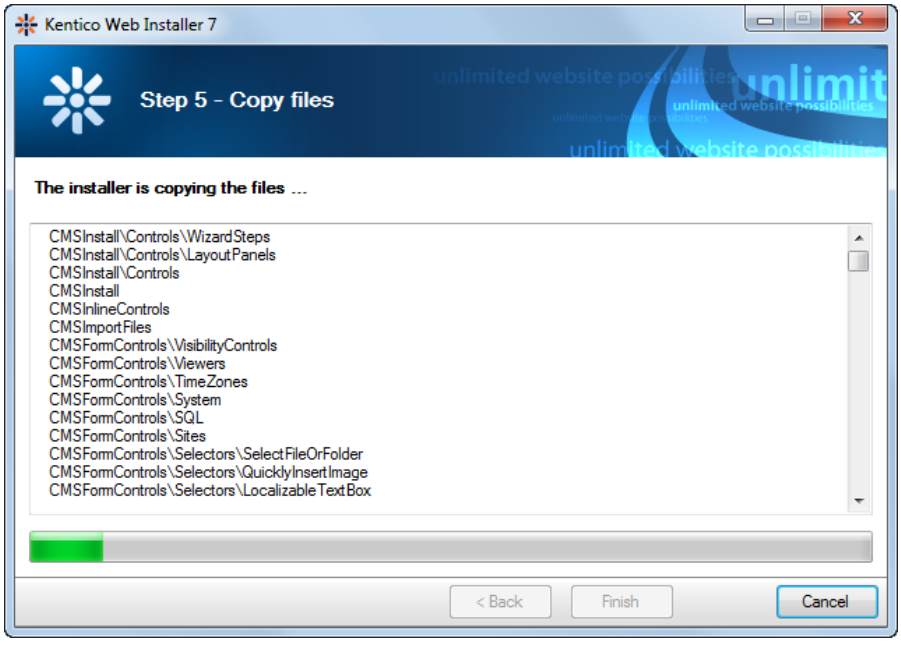
- **Full installation** - performs full installation using all optional components
- **Custom installation** - lets you choose which components will be installed

Enabling the **Register performance counters for Health Monitoring** option registers performance counters for the currently installed instance of Kentico CMS in Windows. The counters can be used for monitoring of system load and performance in an external application. Please note that by enabling this option in installation to a remote server, the counters will be registered on the current machine, not the remote one.
Checking the **Install Windows services for Scheduler and Health Monitoring** option installs the **Kentico CMS Scheduler** and **Kentico CMS Health Monitor** services.
More details on these features can be found in Modules -> Health monitoring.



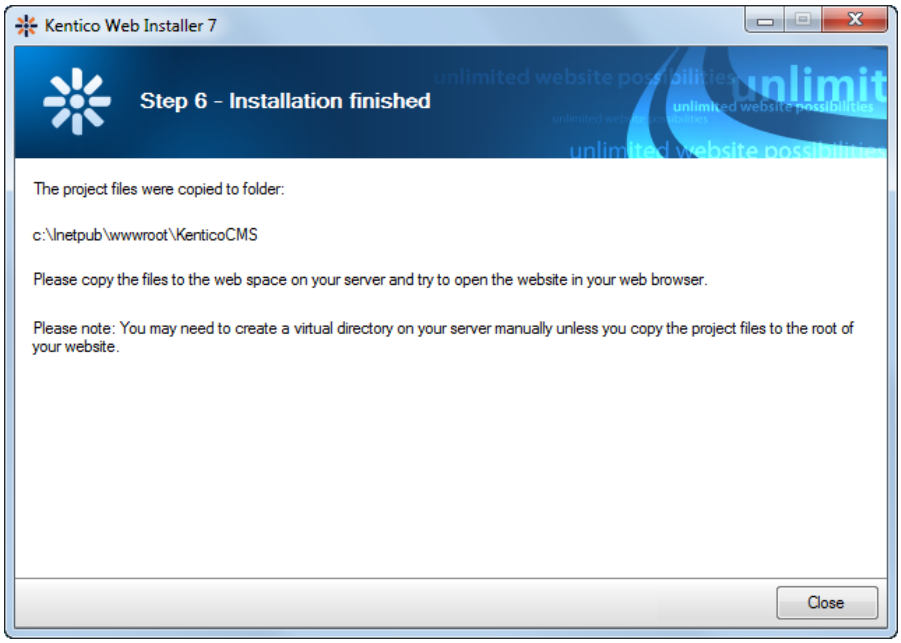
Step 5 - Copy files

The installer copies the project files to the specified folder:



Step 6 - Installation finished

The system creates the web project on your disk and displays a confirmation message:



Now you need to copy the website to your server (e.g., over FTP). If your web project isn't placed in the root of the remote website, you may need to create a virtual directory as described in Creating a virtual directory.

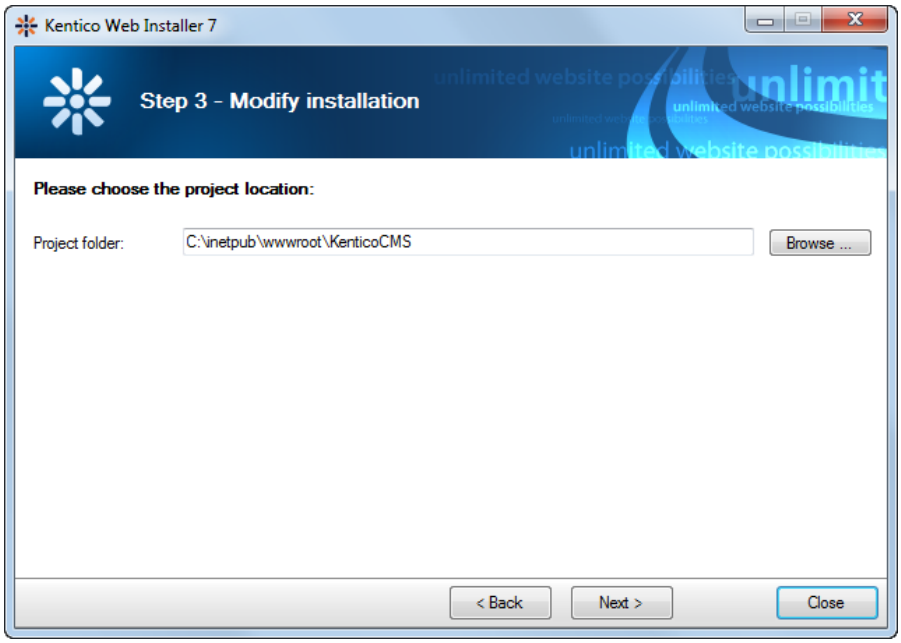
Continue to the Database setup topic.

3.3.2.5 Existing Kentico CMS installation

The **I want to modify existing Kentico CMS installation** option in Step 2 of the Web installer enables you to add or remove components in an existing Kentico CMS web project.

Step 3 - Modify installation

Specify the root folder of the project that you want to modify and click **Next**.



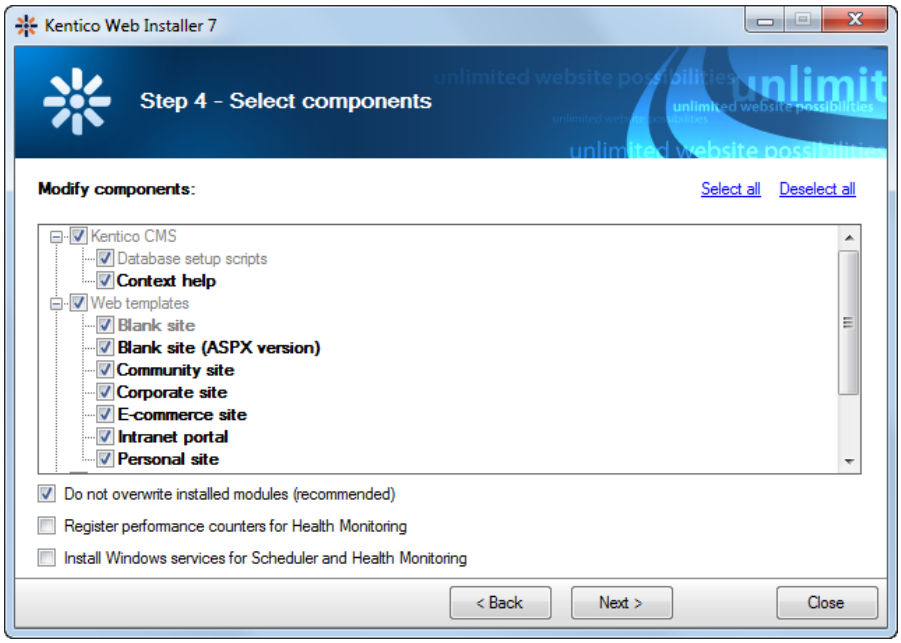
Step 4 - Select components

The installer displays a component tree. Gray components are a native part of the installation and cannot be removed. Black components are optional. The check-boxes indicate which components are currently installed and you can modify the installation by selecting or unselecting them.

If you enable the **Do not overwrite installed modules (recommended)**, only unselected components will be removed and additionally selected components will be added. If the option is disabled, all enabled components (including *web.config*) will be overwritten by the installation, which means that your modifications to these components may get lost!

Enabling the **Register performance counters for Health Monitoring** option registers performance counters for the currently installed instance of Kentico CMS in Windows. The counters can be used for monitoring of system load and performance in an external application.
Checking the **Install Windows services for Scheduler and Health Monitoring** option installs the **Kentico CMS Scheduler** and **Kentico CMS Health Monitor** services.
Please refer to Modules -> Health monitoring for more details on these features.

Click **Next**.

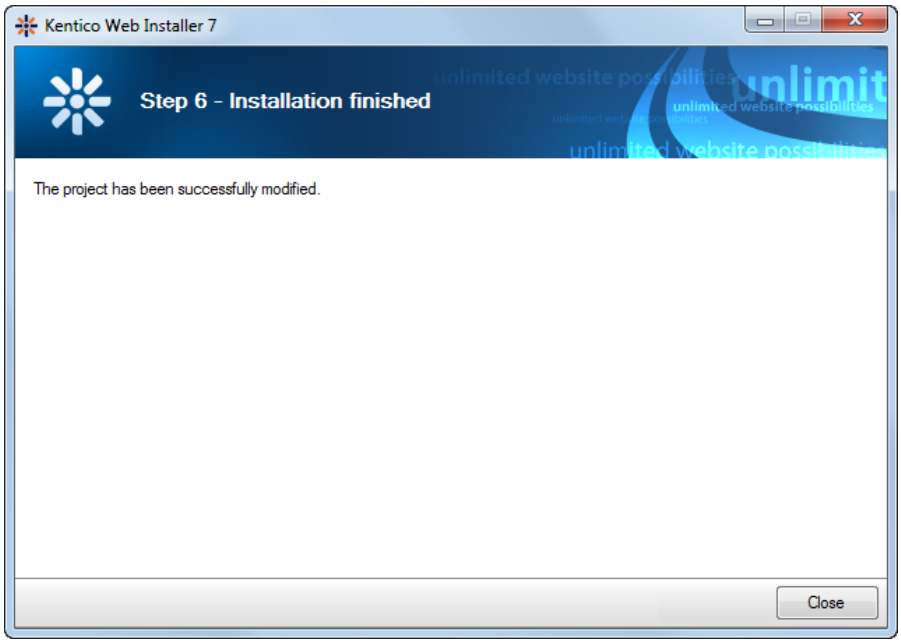


Step 5 - Copy files

The installer modifies the requested components.

Step 6 - Installation finished

When you reach the following screen, it indicates that your project has been successfully modified.



3.3.3 Database setup

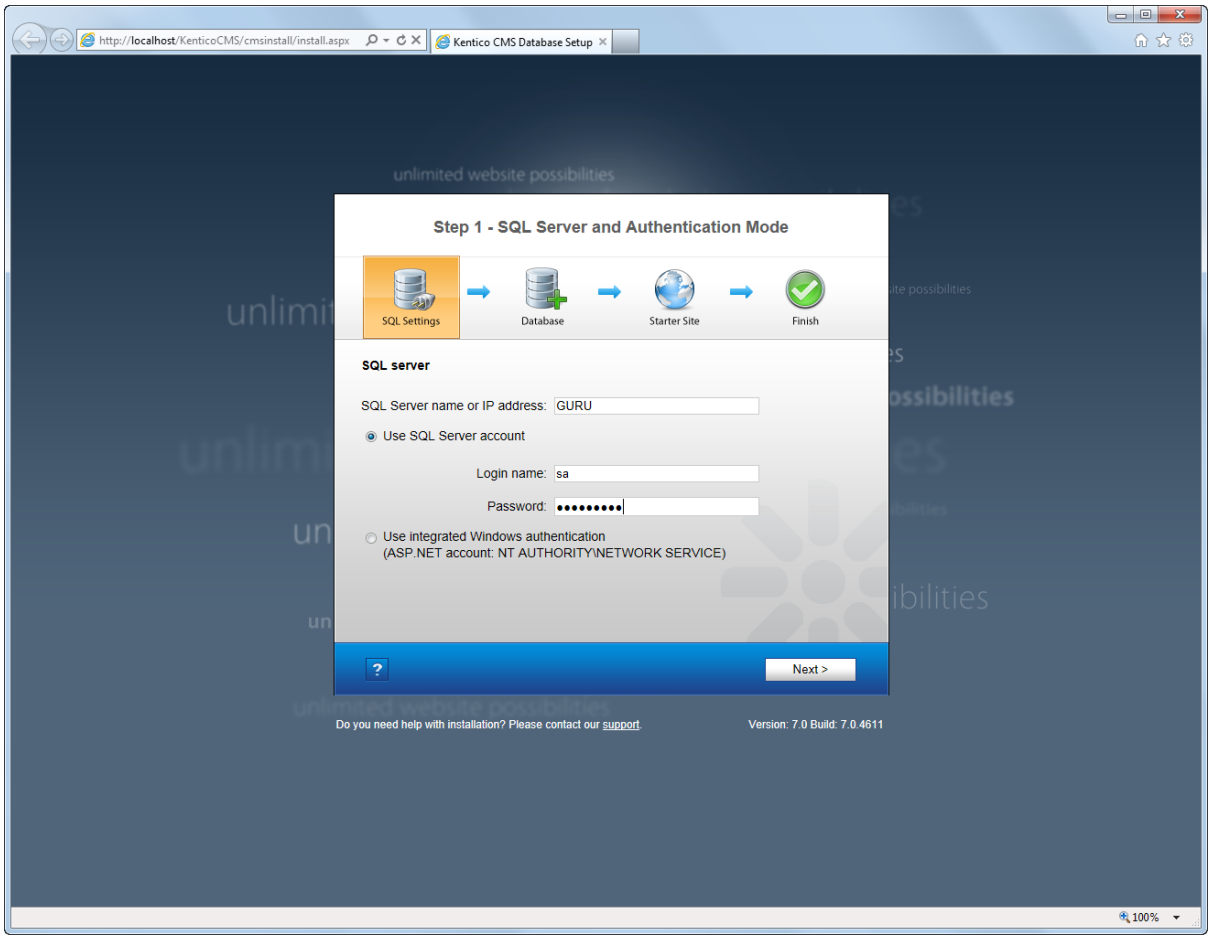
After you successfully finish the Web Installer, open a web browser with the Database Setup. It will create database tables for Kentico CMS in a specified SQL Server database.

Step 1 - SQL Server and Authentication Mode

Specify the SQL Server and authentication mode used to access the server:

- **SQL Server name or IP address** - enter the name of the server. You will typically use one of these:
  - the name of the server (such as DBSERVER1) or
  - the IP address of the server (such as 192.168.1.105) or (local)
  - *<SERVERNAME>\sqlexpress* (if you're using Microsoft SQL Server 2005 Express Edition)
- **Use SQL Server account** - use this option if your server is configured for **mixed mode** authentication with SQL logins
- **Use integrated Windows authentication (ASP.NET account)** - use this option if your server is configured for Windows integrated authentication. In this case, you need to use SQL Server 2005/2008 Management Studio to create a new login for user account under which you currently run the web application (ASPNET for Windows XP and Network Service for 2003 - the actual ASP.NET account name is displayed on the screen).

Click the **Next** button.



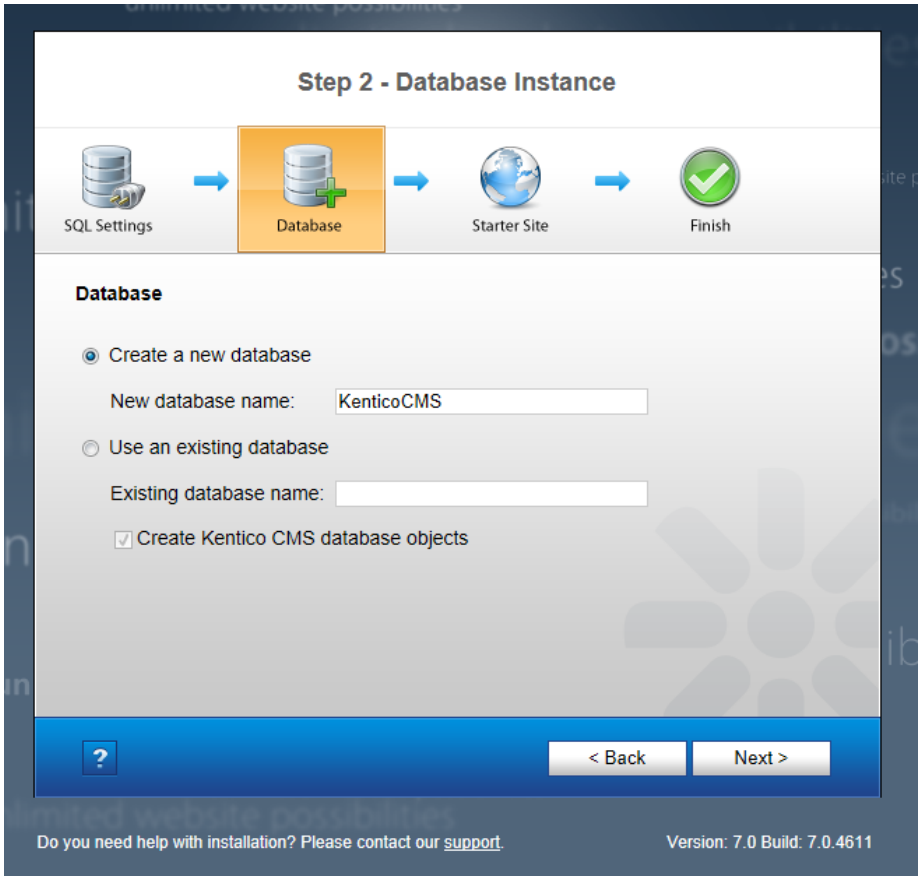
Step 2 - Database Instance

Now you can decide if you want to use an **existing database** or **create a new database**. In both cases, you need to enter the name of the database into the appropriate field.

In case you are using an existing database, you can choose if you want to **Create Kentico CMS database objects**.

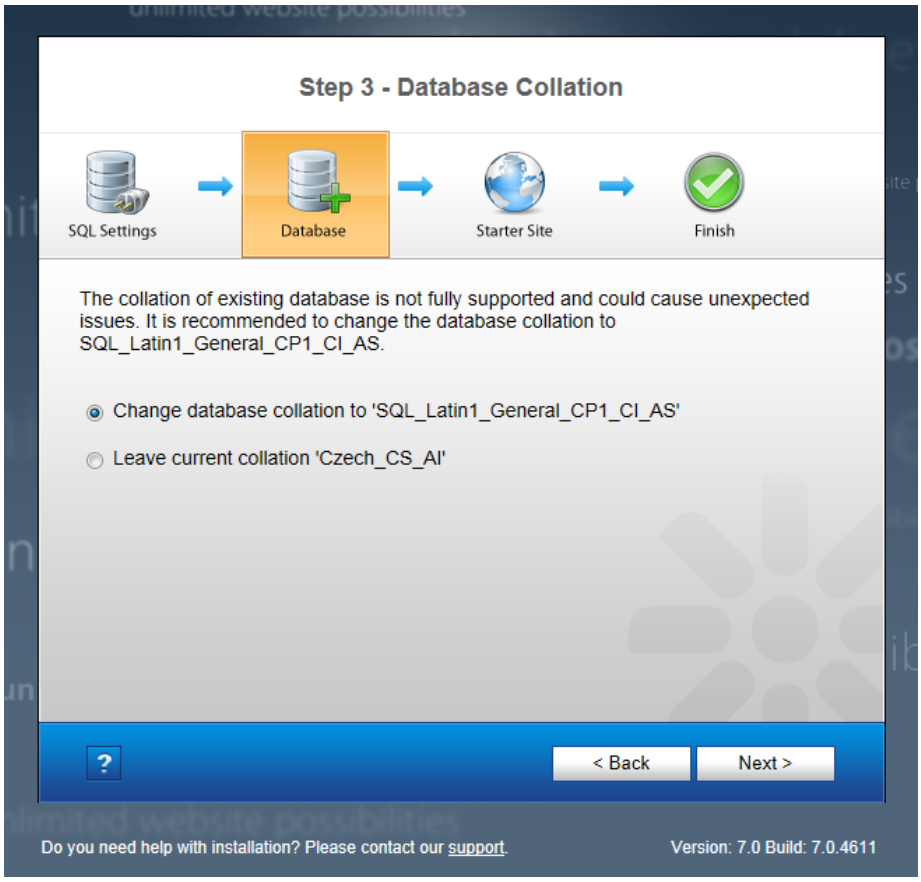
- If the existing database already contains Kentico CMS objects (tables, stored procedures, views), then uncheck the box.
- If the database does not contain these objects (typically when you are installing into an empty database), leave the option enabled.

Click **Next.**



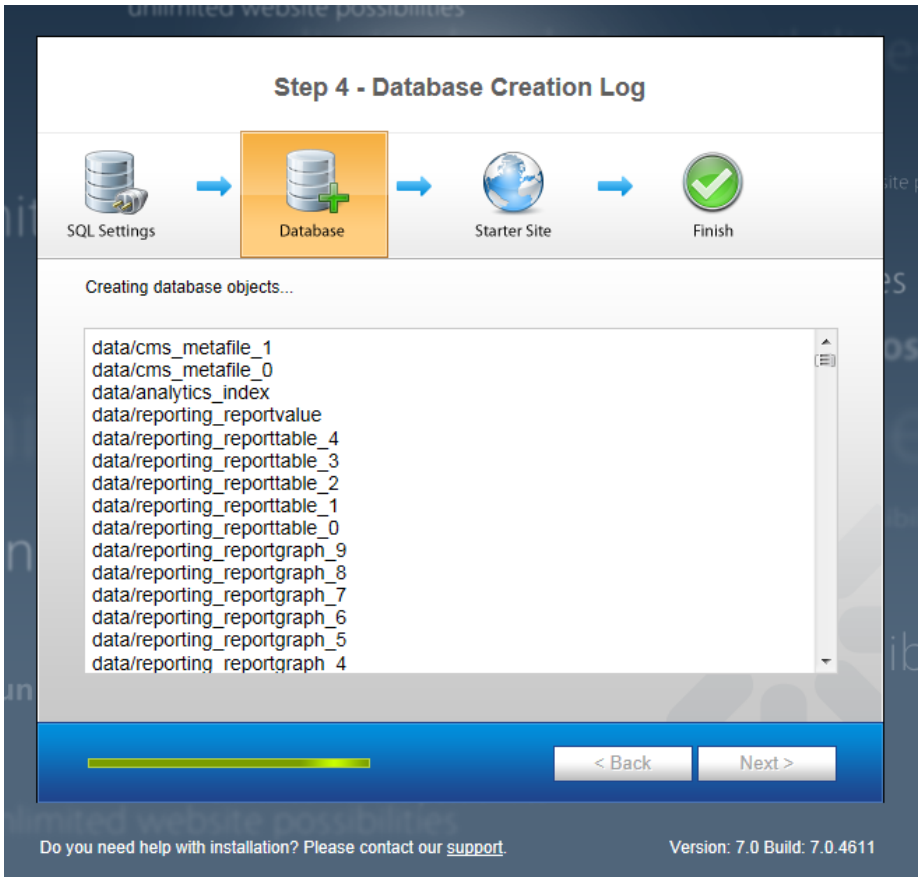
Step 3 - Database Collation

When using an existing database, you may also come across the following dialog. It is displayed if the database collation is different from *SQL_Latin1_General_CP1_CI_AS*. The dialog lets you choose if you want to change the collation or leave the original one.
For correct functionality, it is highly recommended to change it to the recommended value.



Step 4 - Database Creation Log

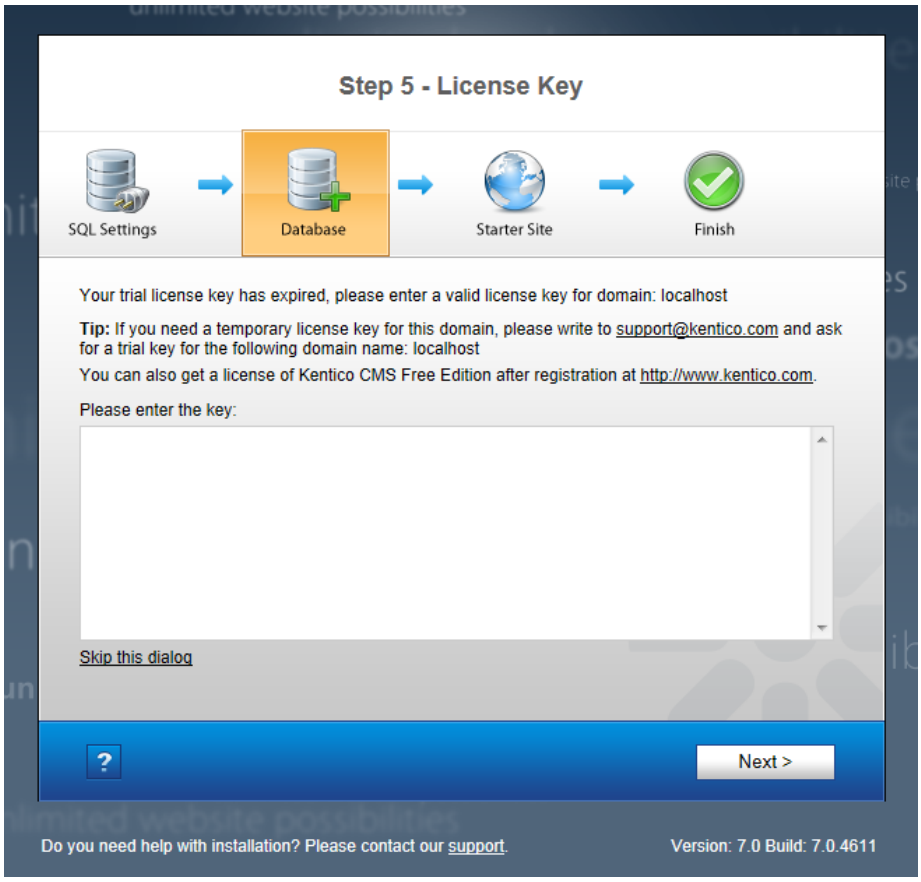
A log will be displayed, showing the progress of database creation. After it finishes, a message *Database has been successfully created.* appears at the top of the log and you are moved forward to the next step.



Step 5 - License Key

If you run Kentico CMS on a domain other than *localhost* or *127.0.0.x* (where *x* is between 1 and 255), you will be asked to insert a license key. Since the trial version works only with *http://localhost* and *http://127.0.0.x* (where *x* is between 1 and 255), the same dialog is displayed if your trial period has expired.

Enter a valid license key and click the **Next** button.
Alternatively, you can skip this dialog by clicking **Skip this dialog** link at the bottom left part of the dialog and continue to the New site wizard. You can enter a license key later under Site Manager -> Licenses**.**

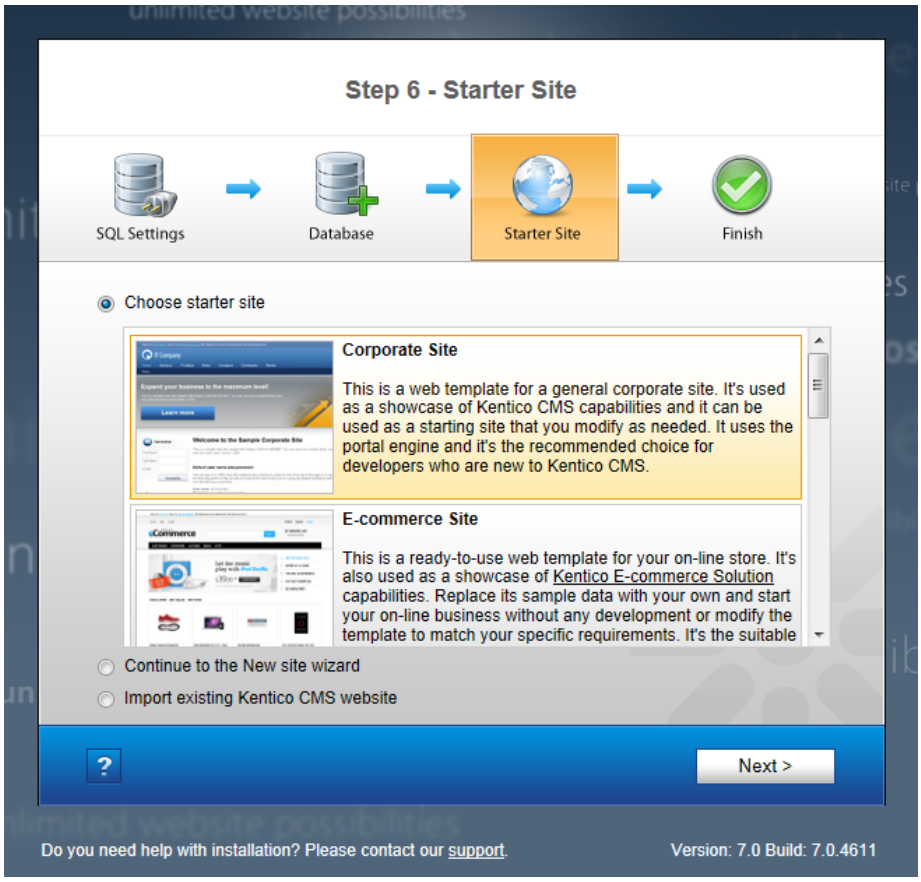


Step 6 - Starter Site

The installer offers you the following options:

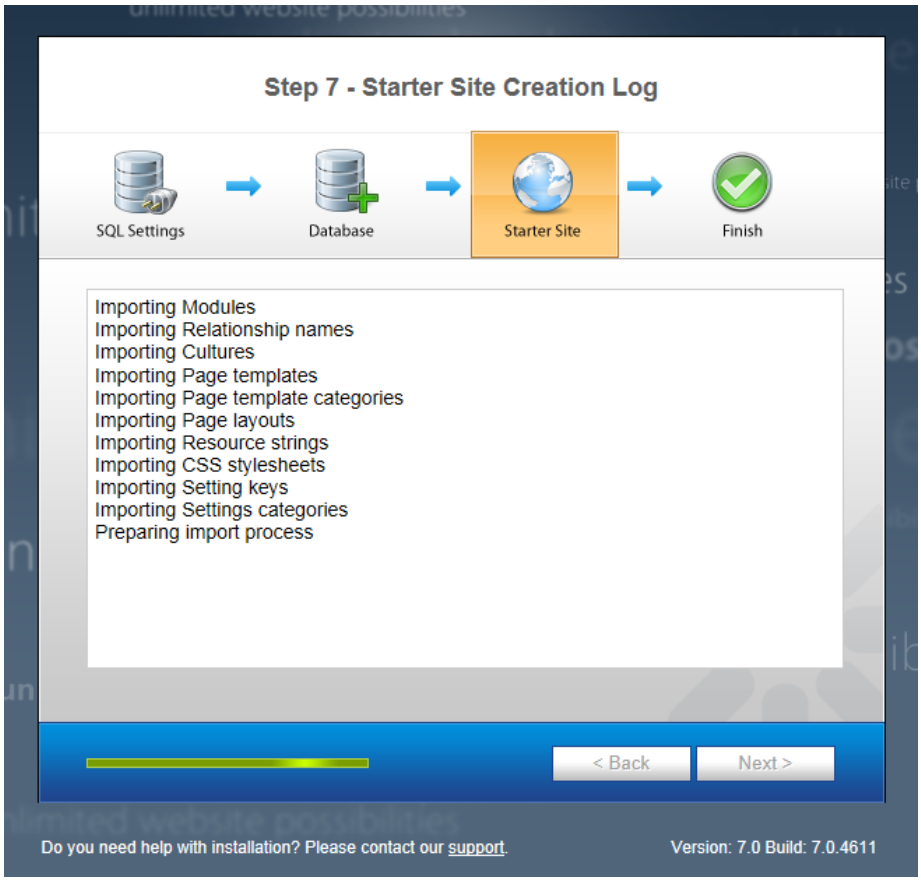
- **Choose a starter site:**
  - **Corporate Site** - this option installs the sample corporate site - it is recommended for most users, especially for evaluators
  - **Blank site** - this is a blank site without any content; you will use it to create a new site from scratch
  - **Blank site ASPX** - the same as above, but for ASPX page templates
  - and others
- **Continue to the New site wizard** - this option is recommended if you're starting a new site from scratch; learn more in the New site wizard chapter
- **Import existing Kentico CMS website** - use this option if you have already created a website with Kentico CMS and you need to import it into a new installation (e.g., on the production server)

If you're new to Kentico CMS, it's highly recommended that you start with the sample Corporate Site (portal engine). If you decide to run the New site wizard, you can find more details in the following chapter.
Select an option and click the **Next** button.



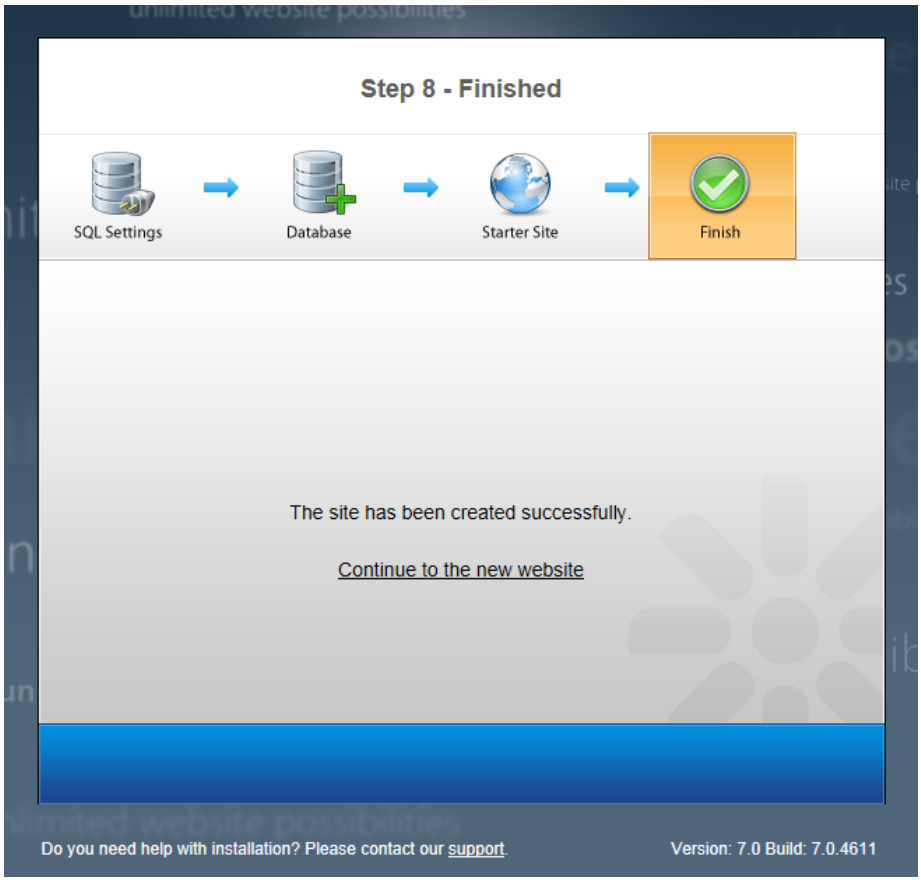
Step 7 - Starter Site Creation Log

A log is displayed showing you the progress of website creation.



Step 8 - Finished

Once the website is created, the installer displays a confirmation and you can follow the link to access the new live website.
This is the final step of the necessary installation procedure. You can now begin with managing the website content.



Default user name and password

The default user name is **administrator**, the default password is blank.

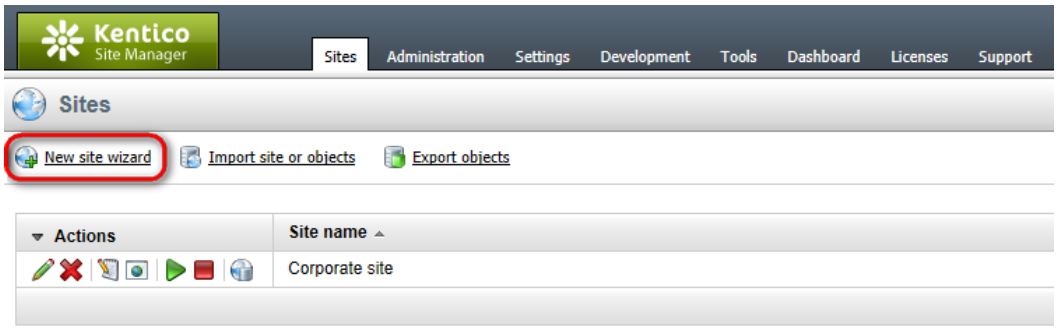
It's highly recommended that you change the password after you finish the installation.

The default URL of your site is *http://localhost/KenticoCMS*.
The default URL of CMS Desk is *http://localhost/KenticoCMS/CMSDesk*.
The default URL of Site Manager is *http://localhost/KenticoCMS/CMSSiteManager*.

3.3.4 New site wizard

3.3.4.1 Overview

The New site wizard guides you through the process of adding a new website to the system. It is accessible from the *Starter Site* step of the Database setup. You can also launch the wizard from **Site Manager -> Sites**.

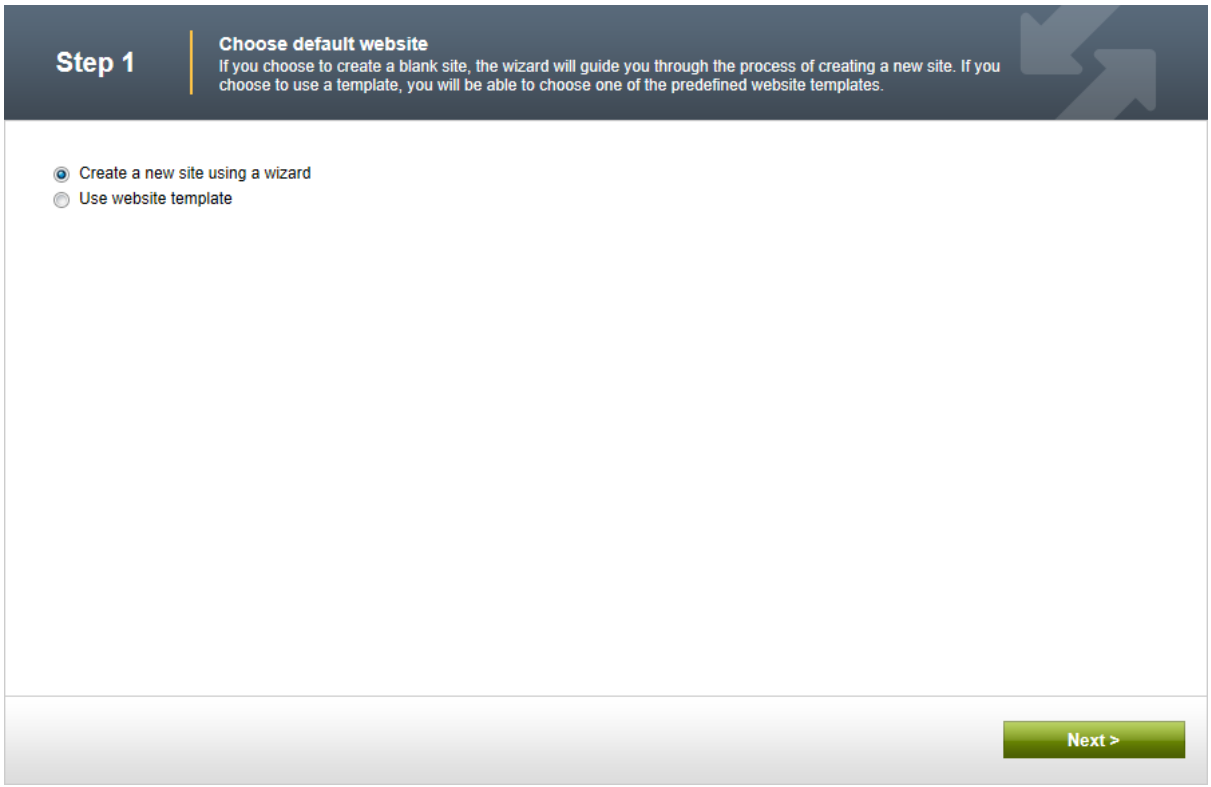


Step 1 - Choose default website

In the first step of the wizard, you can select if you want to create the new site using a wizard or use a website template.

- Create a new site using a wizard - creates a new blank site and allows you to configure the basic structure of the site.
- Use website template - creates a new site based on a template chosen in the next step.

Select one of the options and click the **Next** button. Click one of the links above to learn about the following steps of the wizard for the chosen option.



3.3.4.2 New site

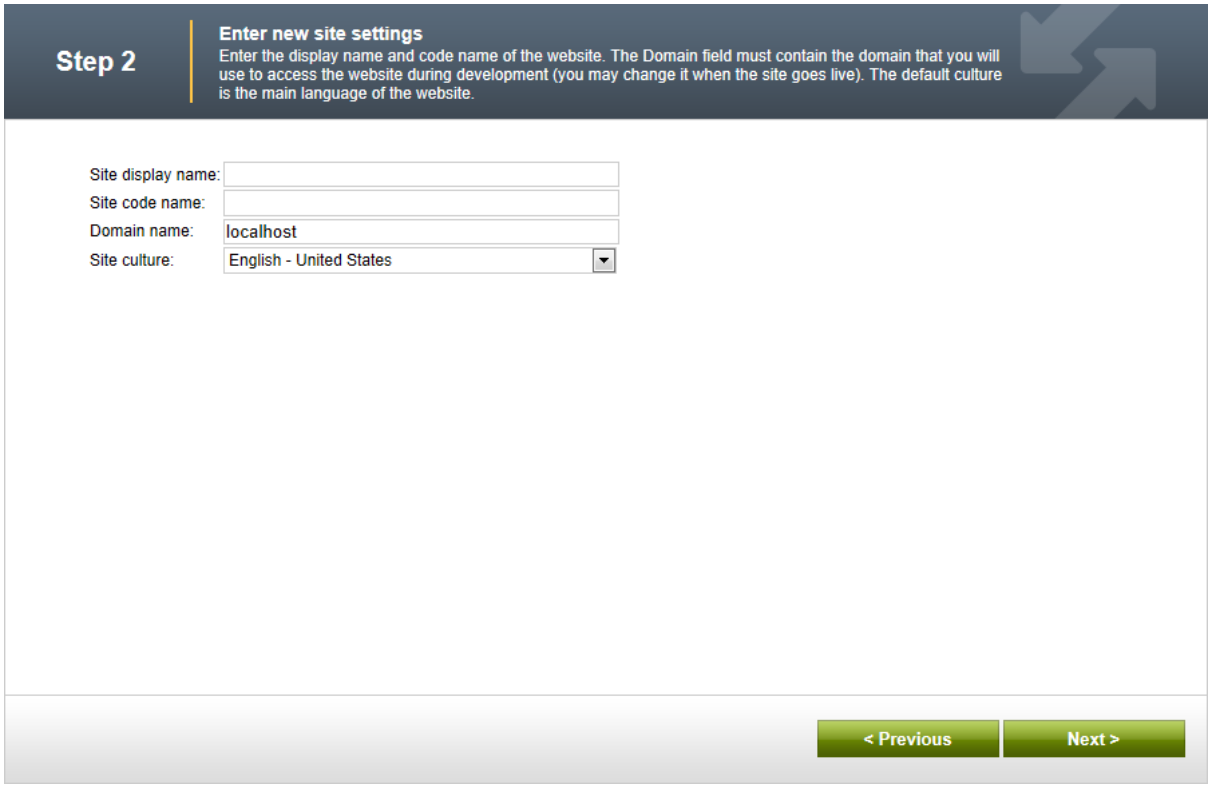
This topic describes adding of a new website to the system using the **New site wizard** when the **Create a new site using a wizard** option is chosen in the first step of the wizard.

Step 2 - Enter new site settings

Enter the following basic site properties:

- **Site display name** - name of the new site displayed in the administration interface.
- **Site code name -** name of the new site used in website code.
- **Domain name** - domain name on which the new site will be running. The domain must be unique for each website running in the system. If you enter the same domain for two websites, they cannot be running at the same time.
- **Site culture** - the default culture of the site.

Click **Next** to continue.



Step 3 - Objects selection

In this step, you can select which objects will be imported along with your new site. You can make this selection by choosing one of the categories displayed in the tree view on the left side of the screen. By selecting a category, a set of check boxes appears in the right part of the screen, letting you select the exported objects.

If you select the root of the tree, the following options will be offered:

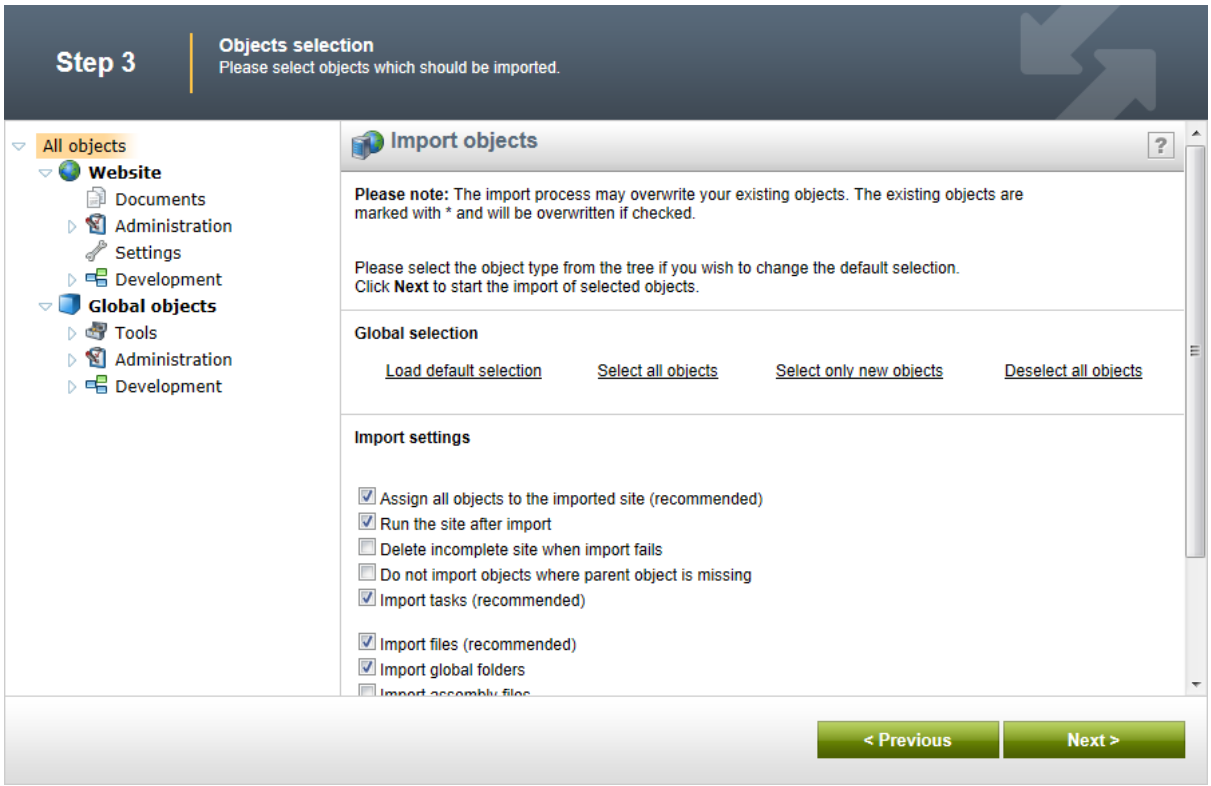
Global selection

- **Load default selection -** if clicked, object pre-selection will be done based on choice in Step 1.
- **Select all objects -** if clicked, all objects will be preselected.
- **Select only new objects** - if clicked, only objects not existing in the database will be preselected.
- **Deselect all objects** - if clicked, all objects will be de-selected.

Import settings

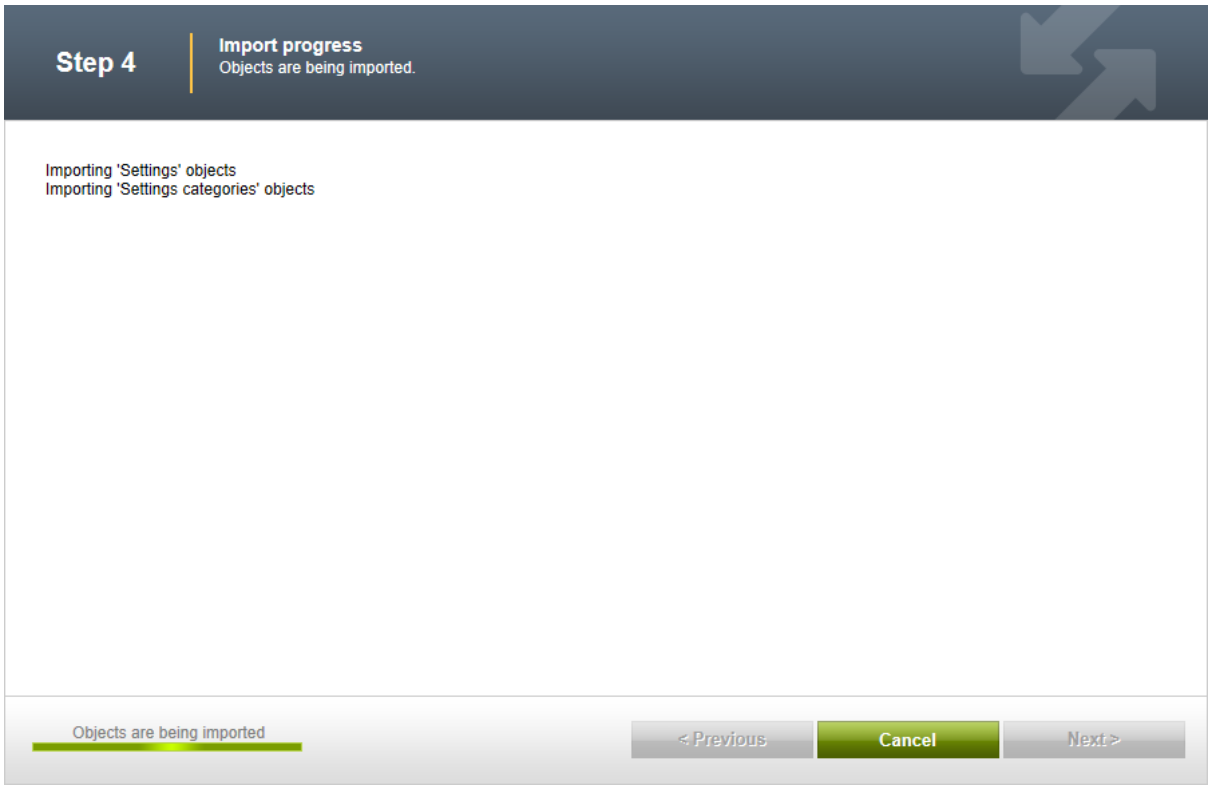
- **Assign all objects to the imported site (recommended)** - if checked, all imported site related objects will be assigned to the imported site.
- **Run the site after import** - if checked, the updated site will be run after the import is finished.
- **Delete incomplete site when import fails** - if checked, incomplete site will be deleted when import fails.
- **Import files (recommended)** - if checked, files will be imported.
- **Do not import objects where parent object is missing** - if checked, objects that are in the package but whose parent object is not present in the target instance will not be imported
- **Import tasks (recommended)** - if checked, delete tasks (incremental deployment) included in the package will be imported

You may leave the default settings and click **Next** to continue.



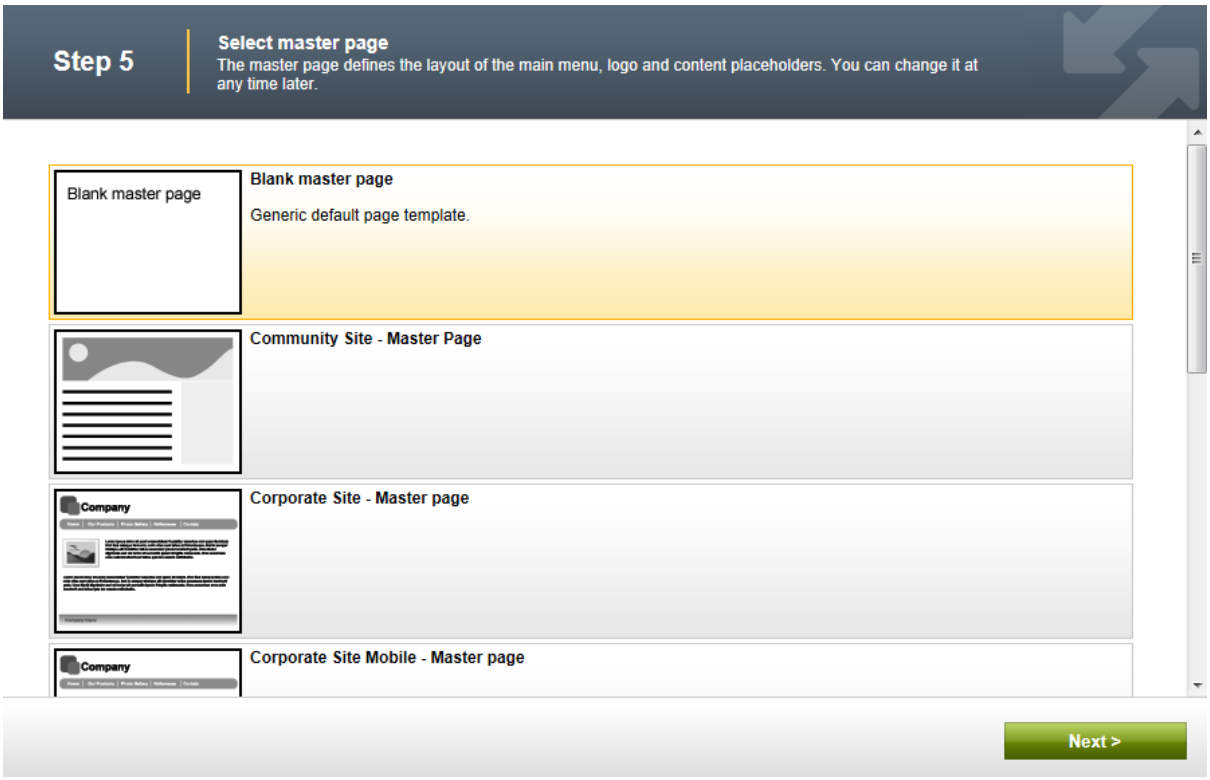
Step 4 - Import progress

A log will be displayed, showing you the progress of the site import. When the process finishes, click the **Next** button.



Step 5 - Select master page

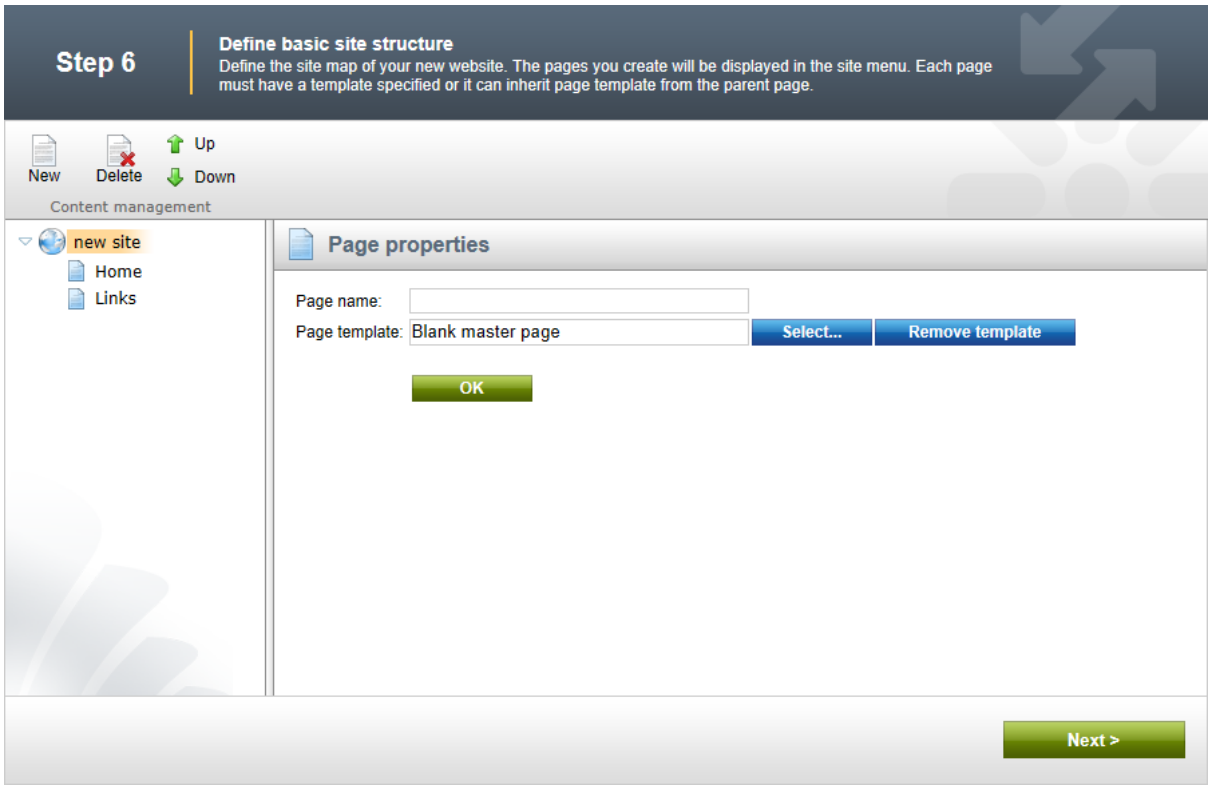
Here you can select the master page layout, which defines the basic visual structure of the website. These settings can be altered any time later, no matter which layout you have selected. Select one of the layouts and click **Next**.



Step 6 - Define basic site structure

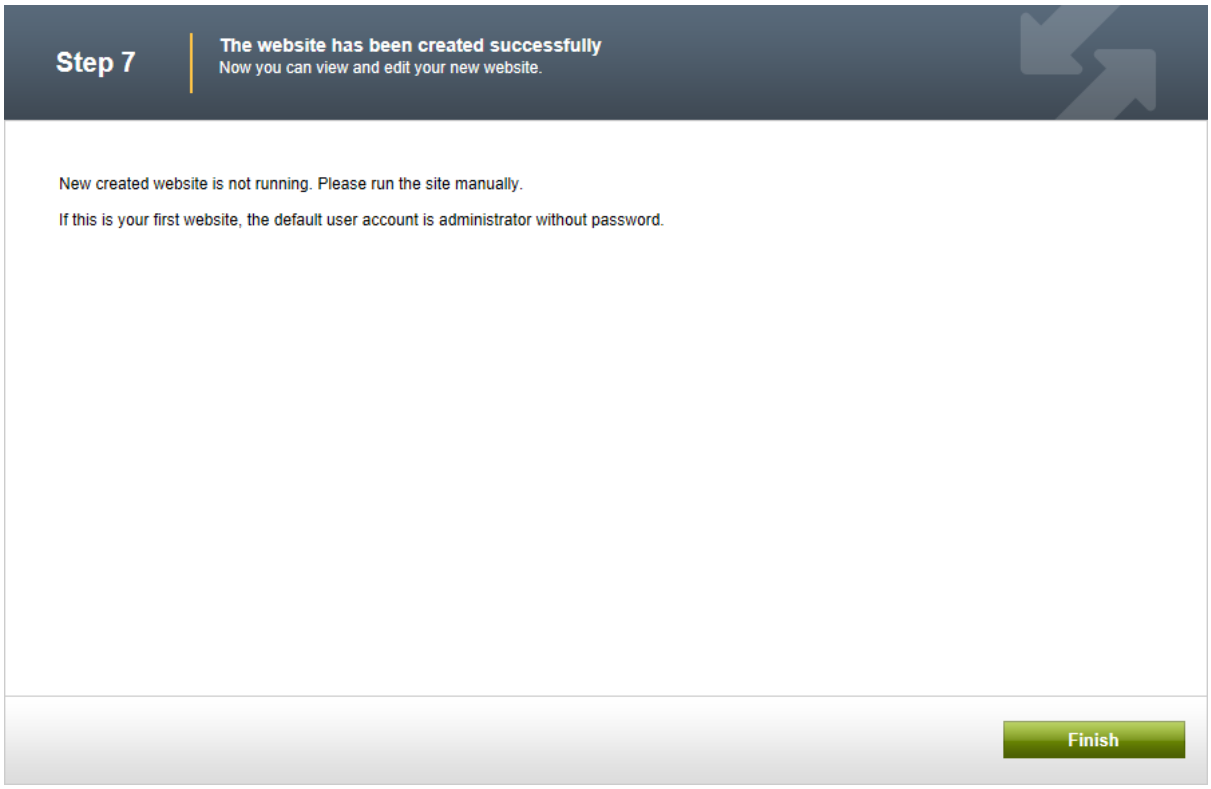
Now you can define the site map of your new website. Select a node of the tree under which you want to place a new page and click **New**. Enter the name of the new page and select a page template which will be used on the page. Alternatively, you can inherit the page template from the parent page by clicking the **Inherit template** button. Click **OK**.

The newly created page will appear in the tree view. Repeat this procedure until you have defined the desired site structure, then click **Next**. The defined site structure can be modified later.



Step 7 - The website has been created successfully

You have successfully created the new website. Click the **Edit your new website** link to switch to CMSDesk and start editing the site immediately. Alternatively, click the **Finish** button to get redirected to **Site manager -> Sites**.



This is the final step of the necessary installation procedure. You can now begin with managing the website content.

3.3.4.3 Website template

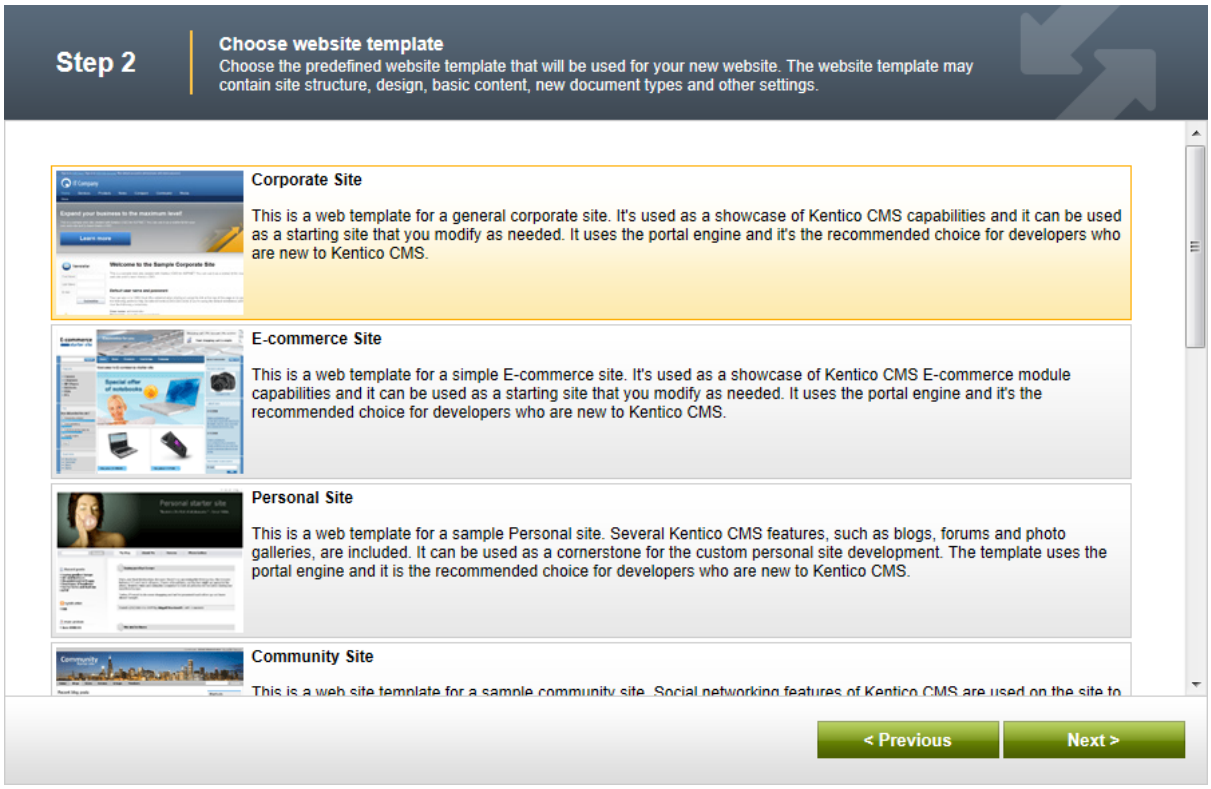
This topic describes creation of a new website using the **New site wizard** when the **Use website template** is chosen in the first step of the wizard.

Step 2 - Choose website template

You can choose from a number of website templates:

- **Corporate site** is a typical web presentation of a company.
- **E-commerce site** is a typical e-shop showing the possibilities of the E-commerce module.
- **Community site** is a sample community website demonstrating social networking capabilities of Kentico CMS.
- **Blank site** is a blank template used for creating websites from scratch.
- and others.

Some of the templates are available in two versions, one using the portal engine and the other using ASPX page templates.
Choose one of the listed website templates and click the **Next** button.

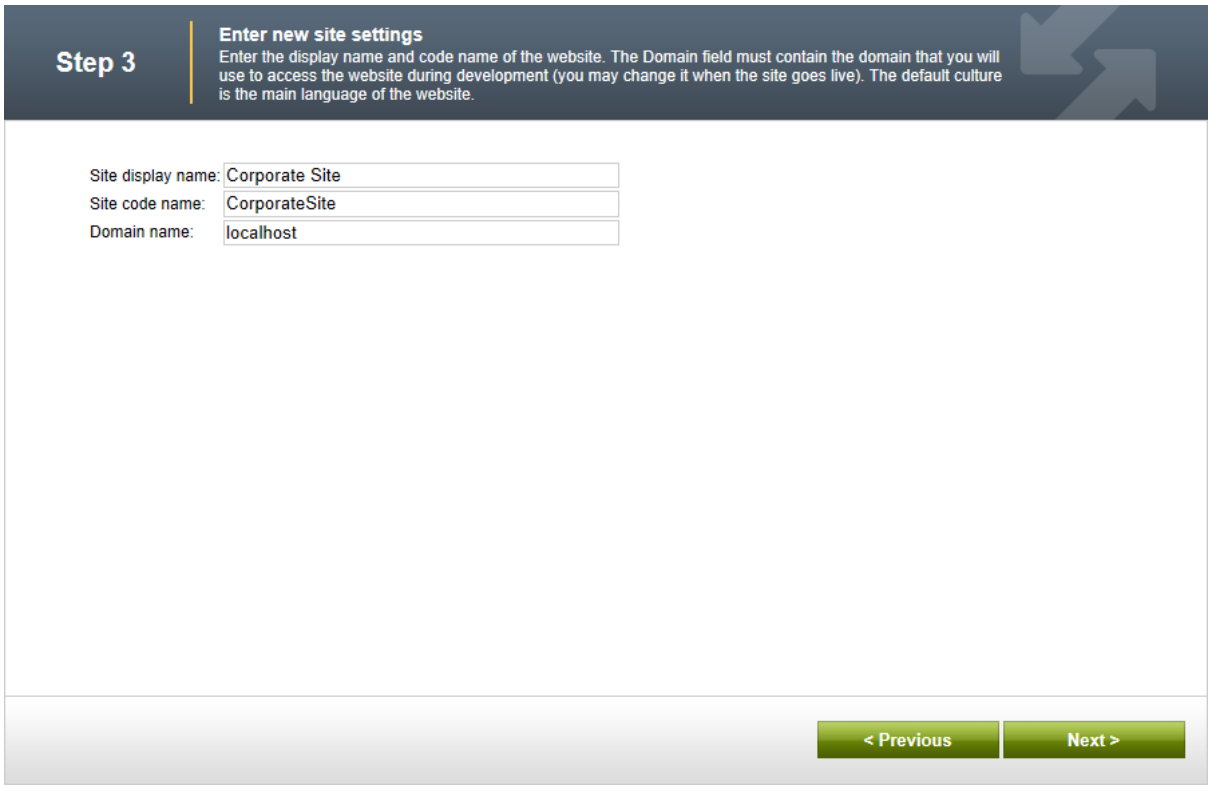


Step 3 - Enter new site settings

Enter the following basic site properties:

- **Site display name** - display name of the new site.
- **Site code name -** code name of the new site.
- **Domain name** - domain name on that the new site will be running. The domain must be unique for each website running in the system. If you enter the same domain for two websites, they can't be running at the same time.

Click **Next** to continue.



Step 4 - Objects selection

In this step, you can select which of the objects from the site package will be imported. You can make this selection by choosing one of the categories displayed in the tree on the left side of the screen. By selecting a category, a set of check boxes appears in the right part of the screen, letting you select which objects will be imported.

If you select the root of the tree, the following options will be offered:

Global selection

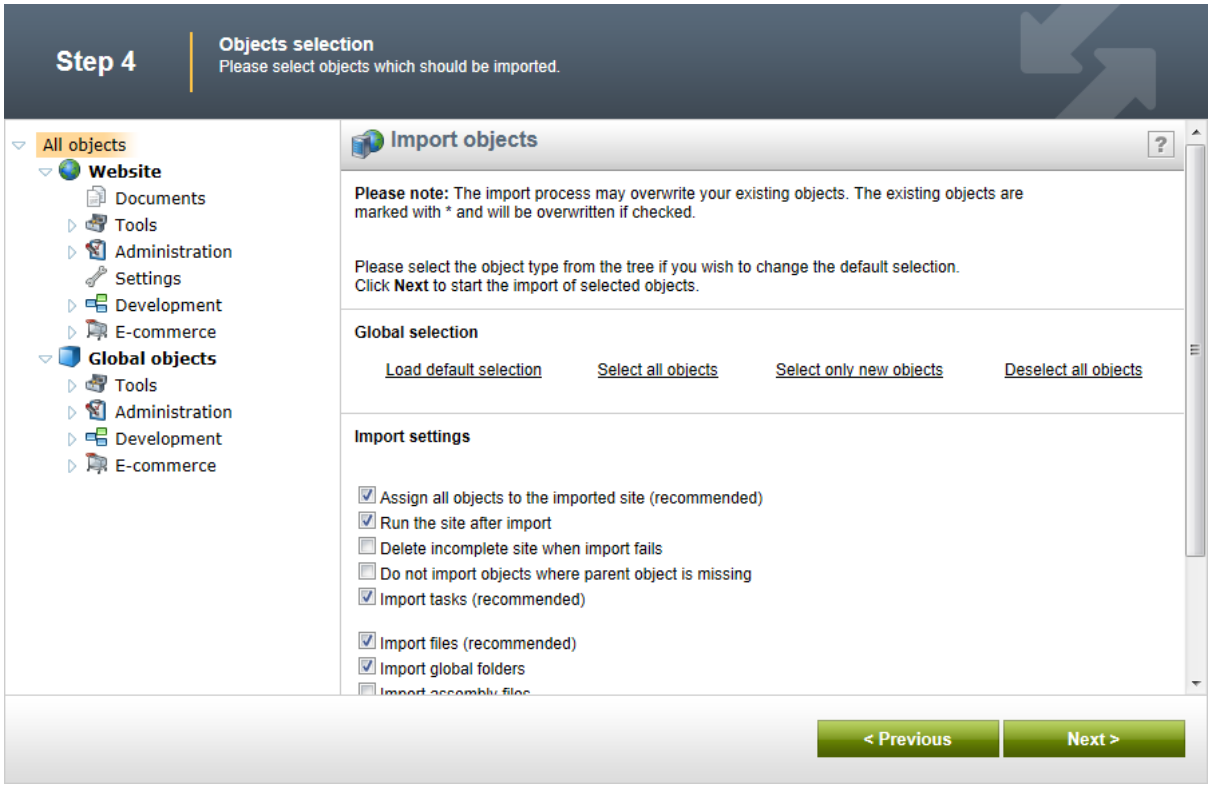
- **Load default selection -** if clicked, object preselection will be done based on choice in Step 1.
- **Select all objects -** if clicked, all objects will be preselected.
- **Select only new objects** - if clicked, only objects not existing in the database will be preselected.
- **Deselect all objects** - if clicked, all objects will be deselected.

Import settings

- **Assign all objects to the imported site (recommended)** - if checked, all imported site related objects will be assigned to the imported site.
- **Run the site after import** - if checked, the updated site will be run after the import is finished.
- **Delete incomplete site when import fails** - if checked, incomplete site will be deleted when import fails.
- **Import files (recommended)** - if checked, files will be imported.
- **Do not import objects where parent object is missing** - if checked, objects that are in the package but whose parent object is not present in the target instance will not be imported
- **Import tasks (recommended)** - if checked, delete tasks (incremental deployment) included in the

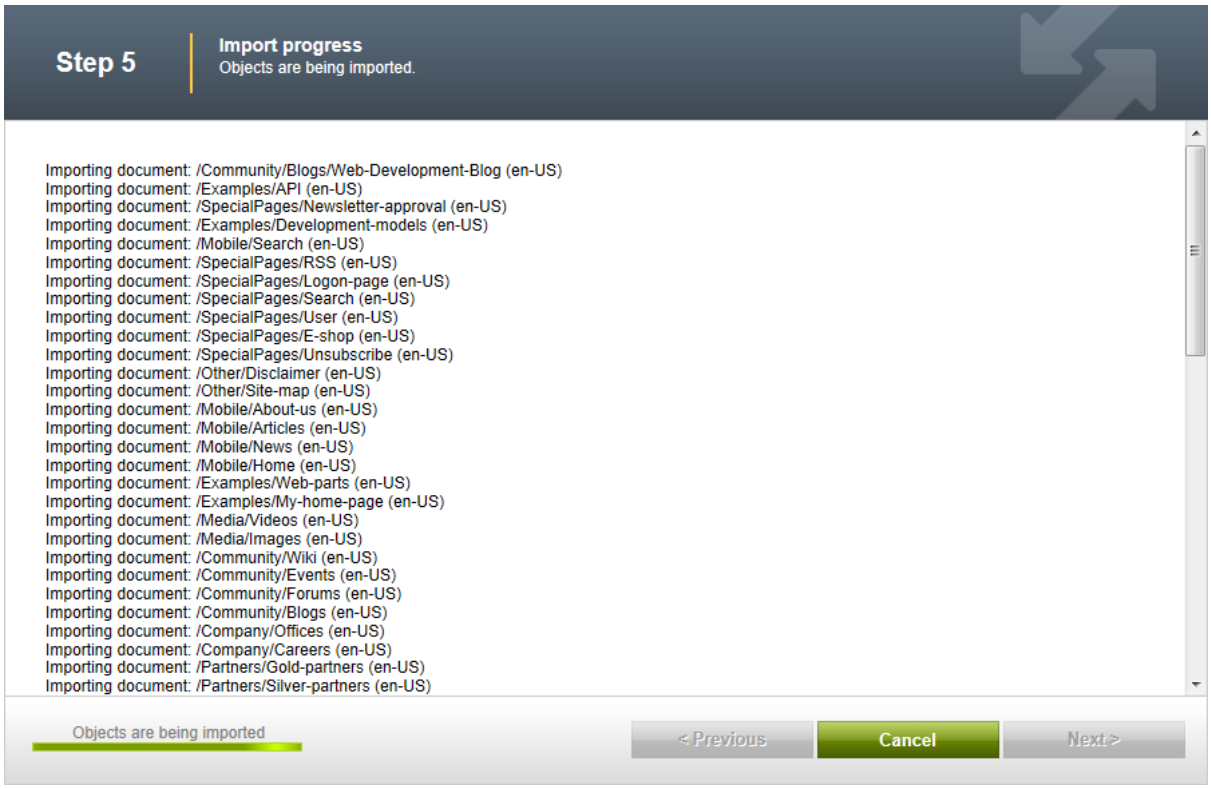
package will be imported

You may leave the default settings and click **Next** to continue.



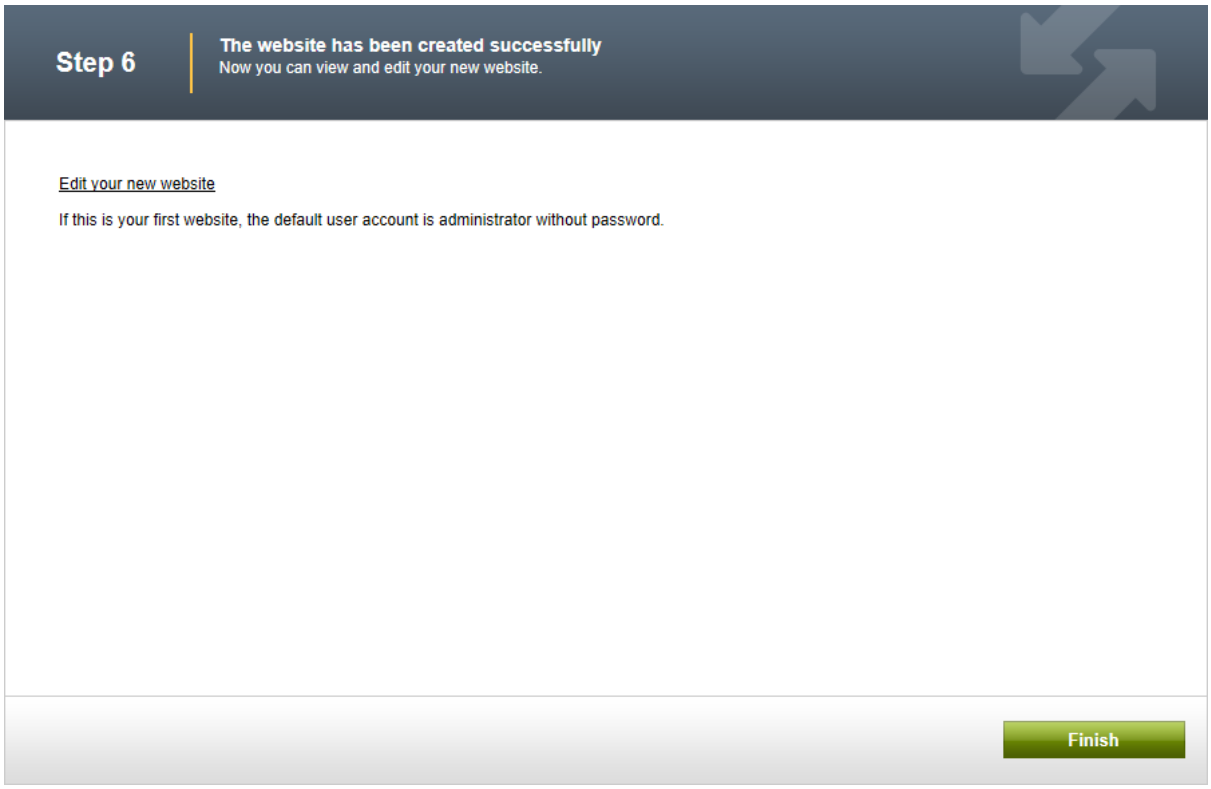
Step 5 - Import progress

A log will be displayed, showing you the progress of the website import. When the process finishes, click the **Next** button.



Step 6 - The website has been created successfully

You have successfully created the new website. Click the **Edit your new website** link to switch to CMSDesk and start editing the site immediately. Alternatively, click the **Finish** button to get back to **Site manager -> Sites**.



This is the final step of the necessary installation procedure. You can now begin with managing the website content.

3.4 Deployment

3.4.1 Deployment to the live server

With Kentico CMS, you can easily develop a website on your local machine. When the website is ready to go live, you need to export it into an export package. You will need to make sure that the ASP.NET account has the **Modify permission** for the disk, specifically for the CMS*SiteUtils**Export* folder. See the Troubleshooting installation issues -> Disk permissions problems chapter for more information on how to set up these permissions.

Go to **Site Manager -> Sites** and click the **Export site** icon next to the site to be exported. Enter the name of the export package and click **OK**. The site will be exported to the **<web project>\CMSSiteUtils\Export** folder.

Now you need to install Kentico CMS on the live server. If you're using a remote server without desktop access, follow these steps:

1. Run Kentico Web Installer on your **local development computer**.
2. Choose to create a new website on a **remote (production) server** and deploy the website into some local disk folder.
3. **Copy all deployed files** to the root of your web server (i.e., the web.config file must be placed in the root of the server). If you need to run the website in a sub-folder, you need to create a virtual directory manually as described in Additional configuration tasks -> Creating a virtual directory.
4. Copy your exported package into the **<website root>\CMSSiteUtils\Import** folder.

5. Make sure your live server is **configured for ASP.NET 3.5 SP1**. It's also highly recommended that the ASP.NET account has Modify permission on the server disk for easy import (however, it's possible to complete the import without Modify permission as well).
6. Open the page /default.aspx of your live web server in the web browser.
7. The Database setup wizard will be displayed. Create the database for Kentico CMS.
8. You may be asked for the license key for your production domain.
9. In step 3, choose to **Import existing Kentico CMS website**.
10. Choose to import the previously exported package and overwrite all duplicates. The import wizard is described in Managing sites -> Export and import -> Importing a site.

Now that you have imported the site, you may need to adjust its configuration:

1. Go to the **Sites** section. Edit website properties and make sure the website domain and domain aliases are configured correctly for the production domain(s).
2. Go to the **Site Manager -> Settings** and make sure your site settings contain correct values, especially the **SMTP server** value in the **System -> E-mails** section.
3. Go to the **Sites** section. Click the **Open live site** icon next to your new site and make sure the website is displayed correctly.

Alternative Approach

If you, for some reason, cannot deploy the website using the procedure described above, you can use a classic backup/restore approach:

1. Backup your development database and restore it on the production server.
2. Copy the web project folder into the root of your production website.
3. Update the CMSConnectionString value in the web.config file.

Shared Hosting Environment and Medium Trust Level

If you're deploying Kentico CMS on a shared hosting server that requires medium trust level, you may need to make additional configuration changes as described in Additional configuration tasks -> Configuration for Medium Trust environment.

Related topics: Visual Studio integration -> Pre-compilation (Publish function)

3.4.2 Deployment to Windows Azure

Kentico CMS is compatible with the Windows Azure Platform. To learn how to install, configure and deploy a website to this type of environment, please refer to the Windows Azure Deployment Guide.

3.5 Additional configuration tasks

3.5.1 Overview

The following chapters contain descriptions of various configuration tasks that require changes to your system. You needn't perform them for standard installation, they are required only in specific situations.

3.5.2 Creating a virtual directory

If you need to install Kentico CMS manually on a remote server or restore it from backup and, at the same time, you run Kentico CMS in a sub-folder (in contrast to running Kentico CMS in the root of the website), you need to create a new virtual directory for the folder where you have the web project files.

Application root

The root of the website or the virtual directory must be the same as the folder that contains the *web.config* file of Kentico CMS. This folder is called **application root**.

The creation of a virtual directory depends on the installed IIS version:

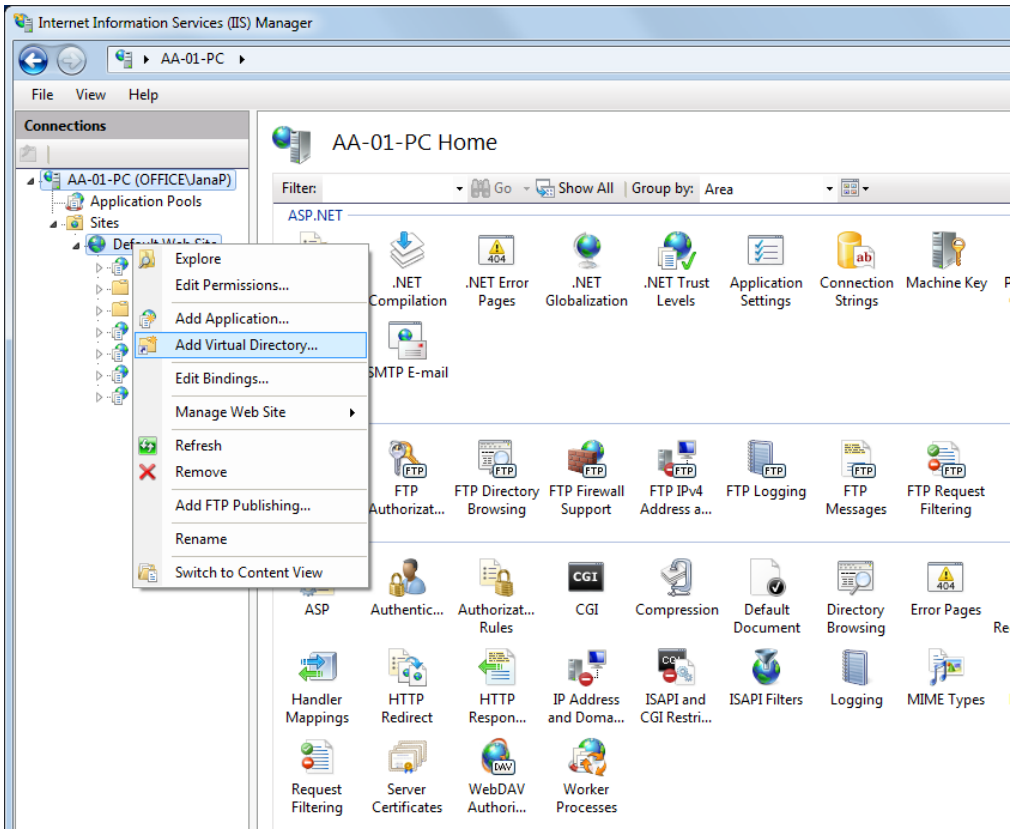
Creating a virtual directory using IIS 7.5 and 7.0

Since IIS 7, there exists a separate concept for a *virtual directory* and an *application*.

- You can map a virtual directory to a physical directory that is located on a local or a remote computer. The physical directory (under the specified name of the virtual directory) then becomes part of the application's URL.
- An application, on the other hand, is a group of files that provides services over protocols. So if you want to run your website as http://localhost/KenticoCMSDemo, you have to **use an application instead of a virtual directory** and specify KenticoCMSDemo as the application's alias.

1. Open IIS Manager
   - On Windows 7: Open **Start -> Control Panel -> System and Security** category **-> Administrative Tools -> Internet Information Services (IIS) Manager**.
   - On Windows Server 2008 or Windows Vista: Open **Start -> Control Panel -> System and Maintenance** category **-> Administrative Tools -> Internet Information Services (IIS) Manager**.

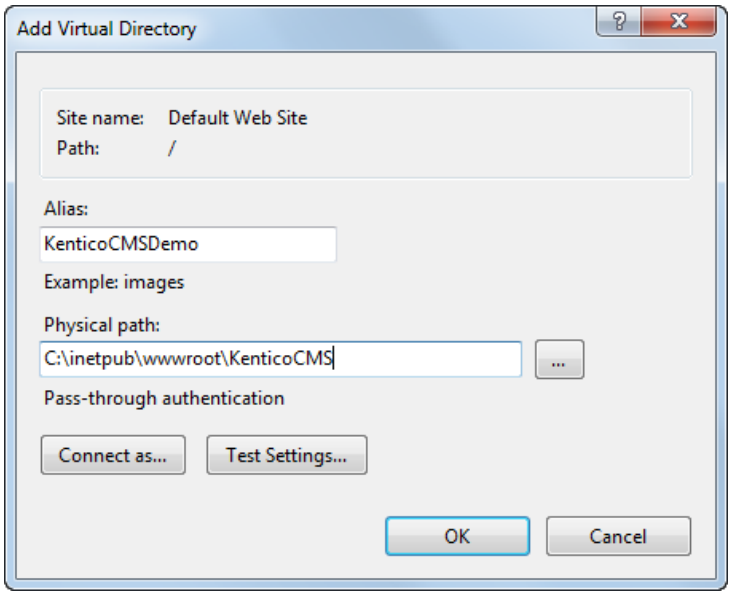
2. Expand **local computer -> Sites ->** right-click on **Default Web Site** (or other website if you're running multiple websites on the same computer) and choose **Add Virtual Directory...**



- An **Add Virtual Directory** dialog appears.

3. Enter the Virtual Directory **Alias**. If you want the website to run as http://localhost/KenticoCMSDemo, enter alias **KenticoCMSDemo**.

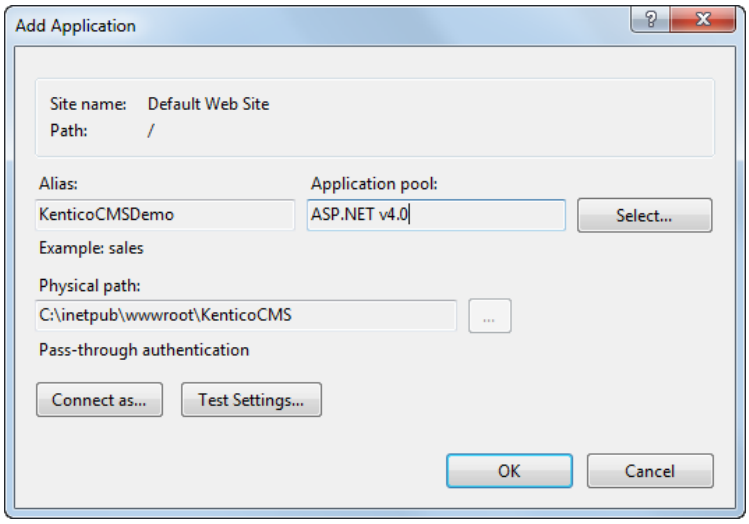
4. Type a path or browse to the physical directory that contains the chosen directory.



5. Click **OK**.
   - The system creates a new virtual directory.

6. Right-click the virtual directory and choose **Convert to Application**.
   - An **Add Application dialog** appears.

7. Click **Select...** and choose the **ASP.NET v4.0** application pool from the drop-down menu.



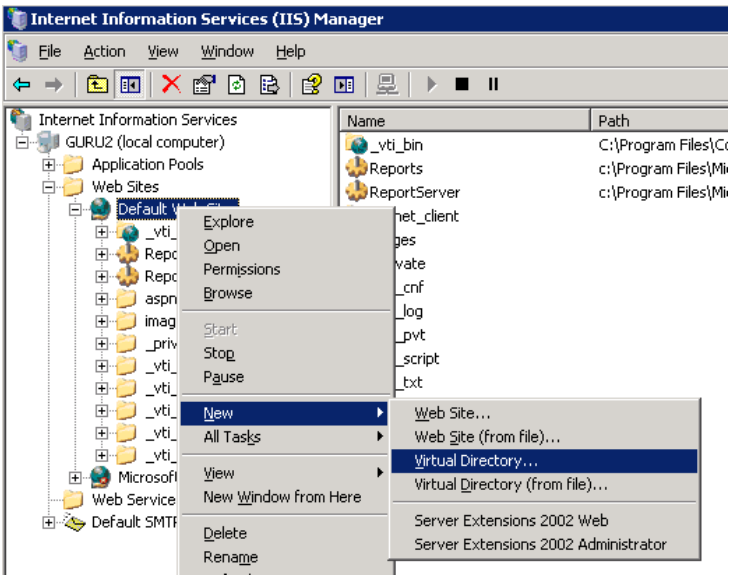
8. Click **OK**.
   - The system converts the virtual directory to an application.

Alternatively, you can create an application in one step by right-clicking a web site and choosing **Add Application...** and filling the required information in the **Add Application** dialog as mentioned above.

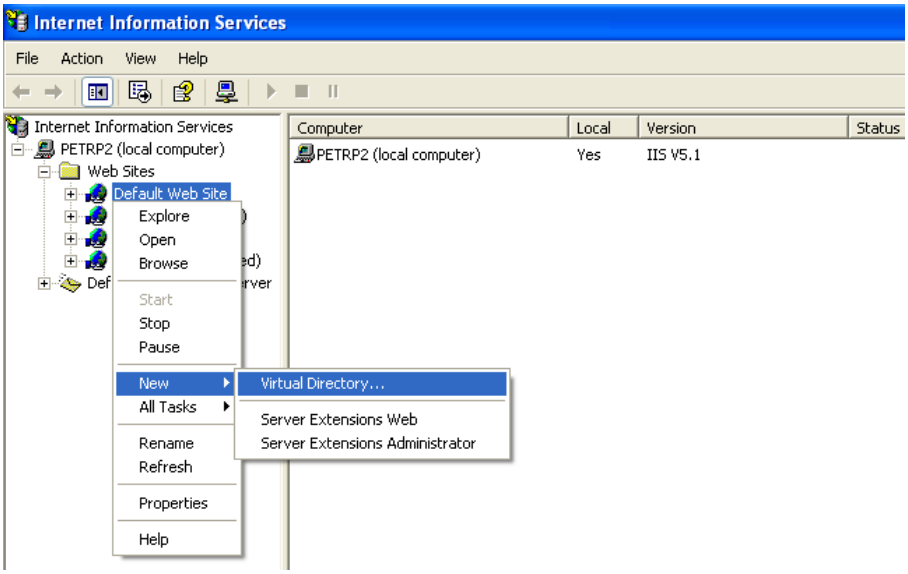
Creating a virtual directory using IIS 6.0 and 5.1

In IIS version 6.0 and earlier, the concept of a virtual directory and an application overlaps, so creating a virtual directory is all you need to do.

1. Open IIS Manager
   - On Windows Server 2003: Open **Start -> Administrative Tools -> Internet Information Services (IIS) Manager**.



- On Windows XP: Open **Start -> Control Panel -> Administrative Tools -> Internet Information Services**.

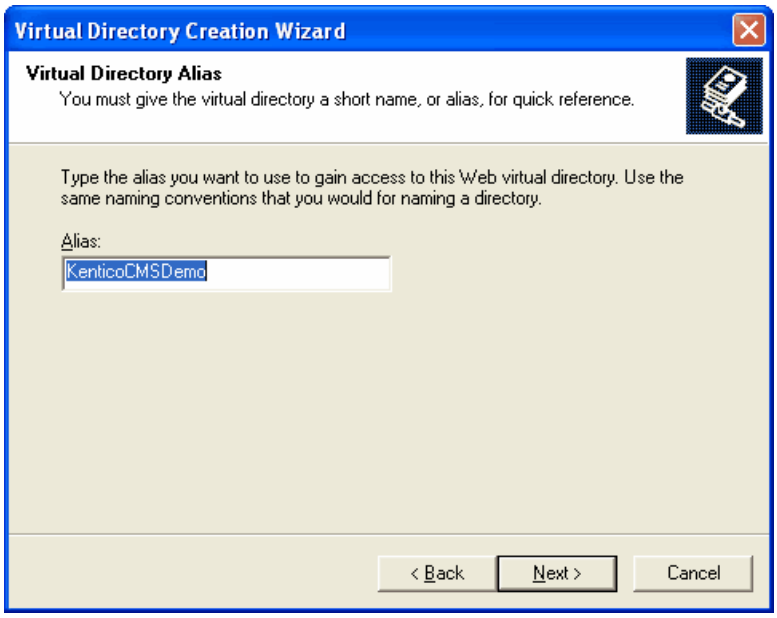


2. Expand **local computer -> Web Sites ->** right-click on **Default Web Site** (or other website if you're running multiple websites on the same computer) and choose **New -> Virtual Directory...**.
   - The **Virtual Directory Creation Wizard** appears. It looks the same for both operating systems.

3. Click **Next** on the first screen.

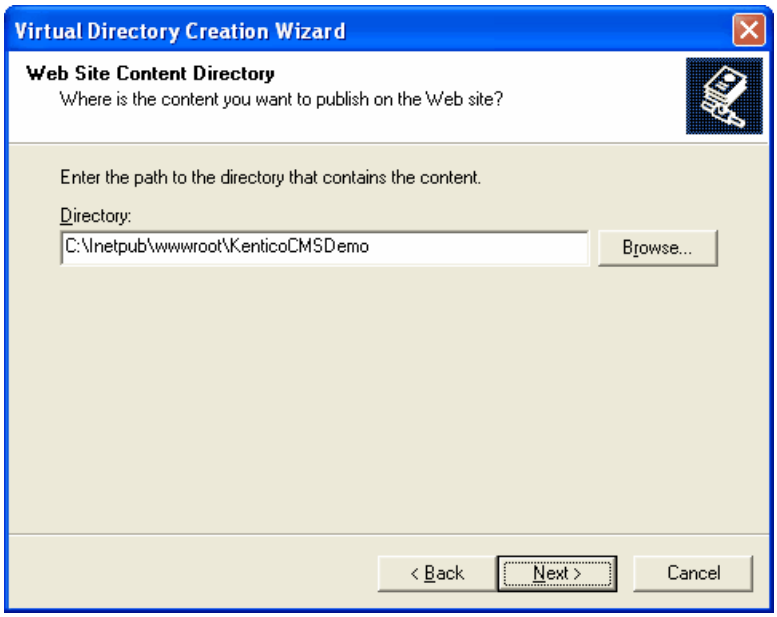
4. Enter the Virtual Directory **Alias**. If you want the website to run as http://localhost/KenticoCMSDemo, enter alias **KenticoCMSDemo**.

5. Click **Next**.



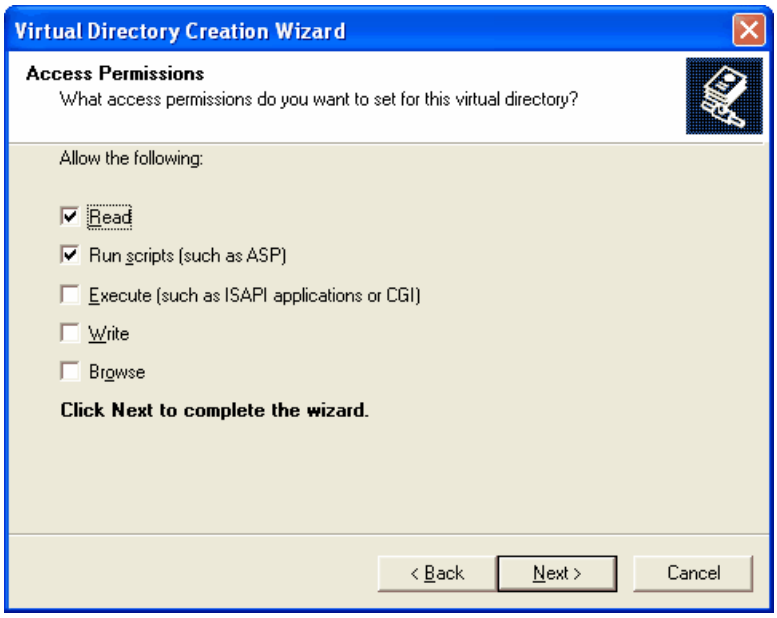
6. Type a path or browse to the physical directory where the Kentico CMS web project files are placed.
   - Please note that you cannot use a remote (shared) disk. The web project files are NOT the files in the C:\Program Files folder - you need to create the web project using the Kentico Web Installer first - see chapter Web Installer for details.

7. Click **Next**.



8. Specify the access permissions for the new virtual directory. Select at least **Read** and **Run scripts** check boxes, and it's recommended that you do NOT allow any other options.

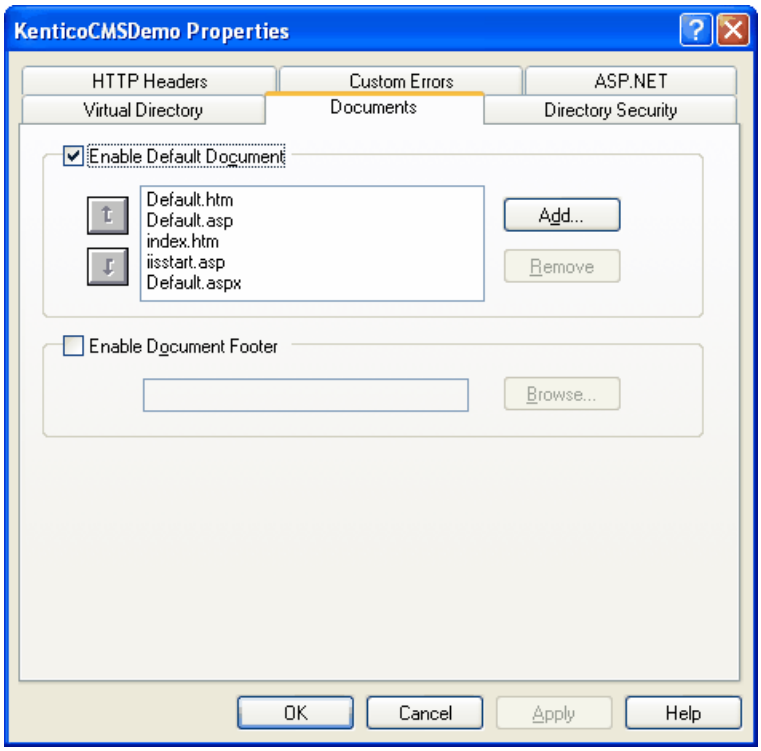
9. Click **Next** and click **Finish**.



The system creates a new virtual directory. Now you need to specify its details.

1. Right-click the virtual directory and choose **Properties**.

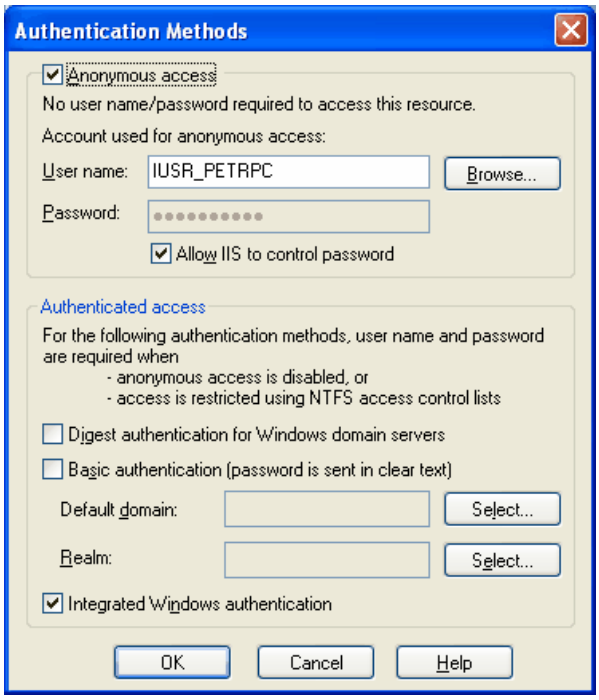
2. Choose the **Documents** tab and make sure the **Enable Default Document** box is checked and the list contains the **Default.aspx** file.



3. Choose the **ASP.NET** tab and make sure the **ASP.NET version is 2.0.50727** (the last number may be different) is selected. If you cannot see the **ASP.NET** tab, you may not have **ASP.NET 3.5 SP1**

installed or registered correctly. IIS 6 may display the **ASP.NET version as 2.0.50727** even if you have **ASP.NET 3.5 SP1** or higher installed and registered.

4. Choose the **Directory Security** tab, click **Edit** in the **Anonymous access and authentication control** section and make sure that both **Anonymous access** and **Integrated Windows authentication** are chosen.

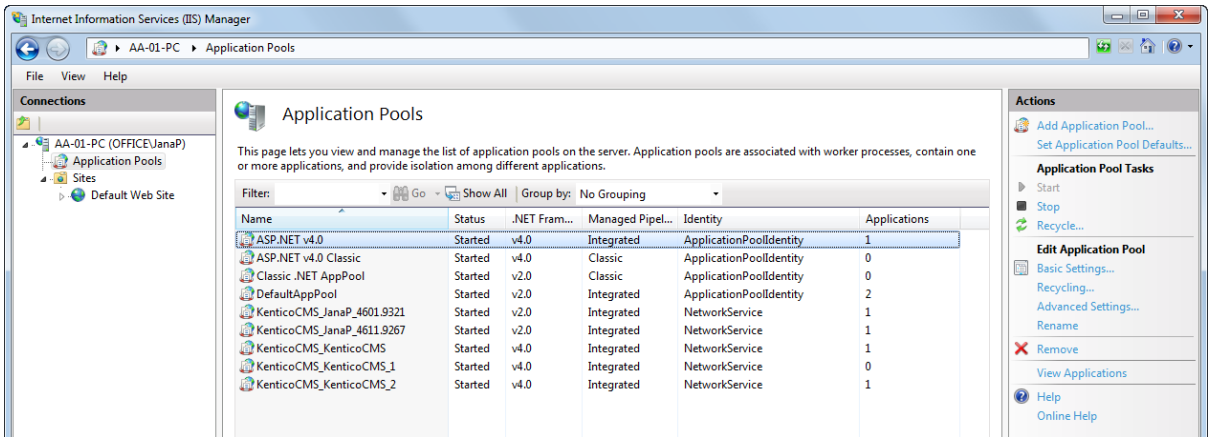


5. Click **OK** on all dialogs to save the changes.

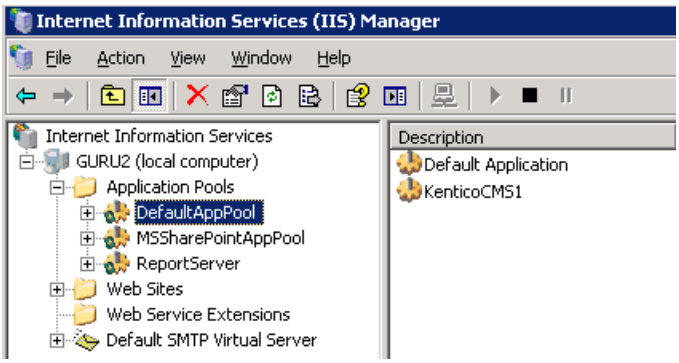
3.5.3 Configuring Application Pools

Application Pools provide you with additional level of website management. They are supported only by Windows Server 2003/2008/Vista/7. You can configure them in the **Internet Information Services (IIS) Manager** under **local computer -> Application Pools.**

IIS 7.0:



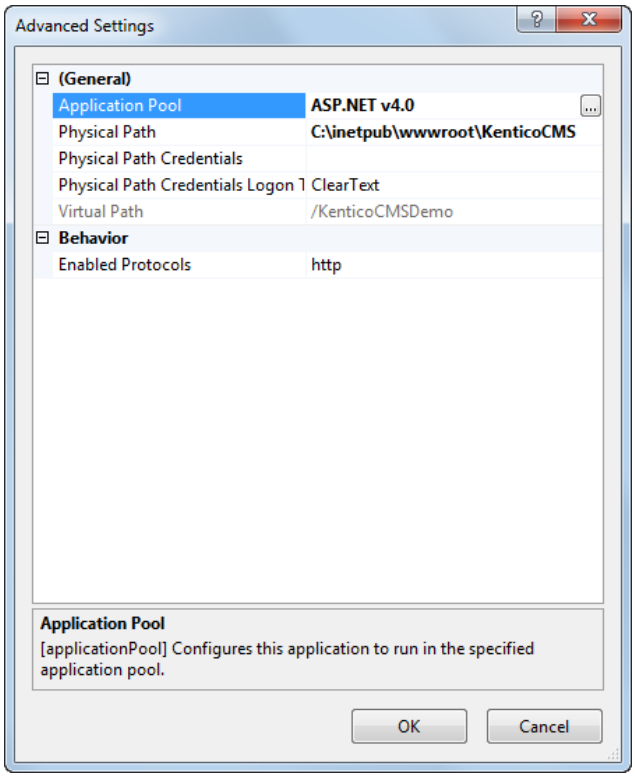
IIS 6.0:



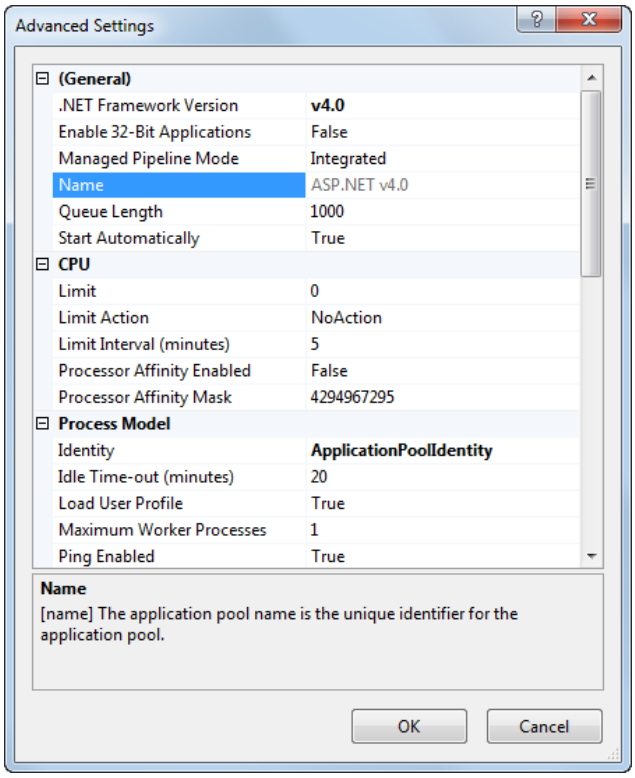
The application pools allow you to group applications (websites) into pools that share server resources using the same rules. This chapter contains recommendations on how to configure the website application pools for Kentico CMS.

Application pools in IIS 7.0 and higher

You can check and change the assigned application pool by right-clicking an application under **local computer -> Sites -> Default Web Site** (or other web site) and selecting **Manage Application -> Advanced Settings...**

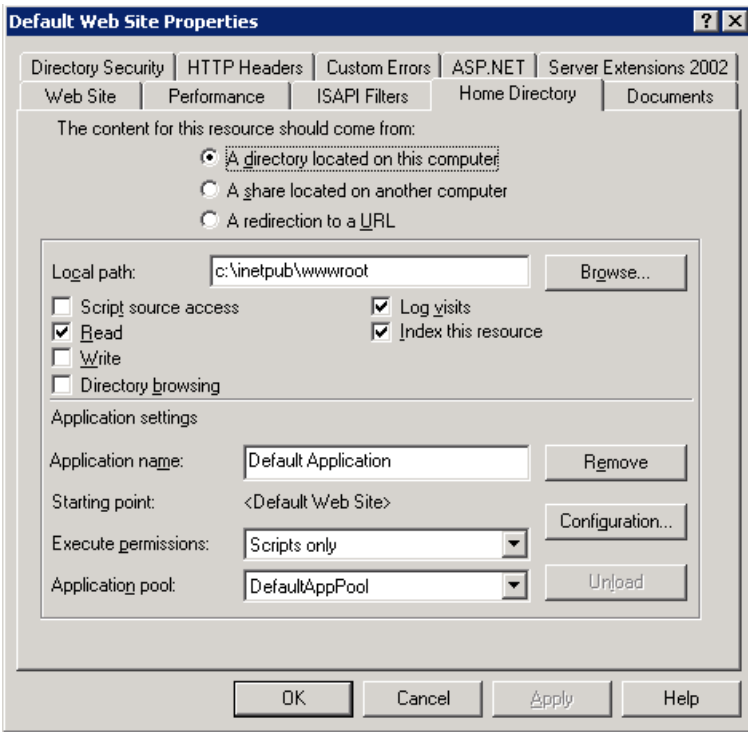


You can set the Recycling and other options by right-clicking on an application pool under **local computer -> Application Pools** and selecting **Advanced Settings...**.

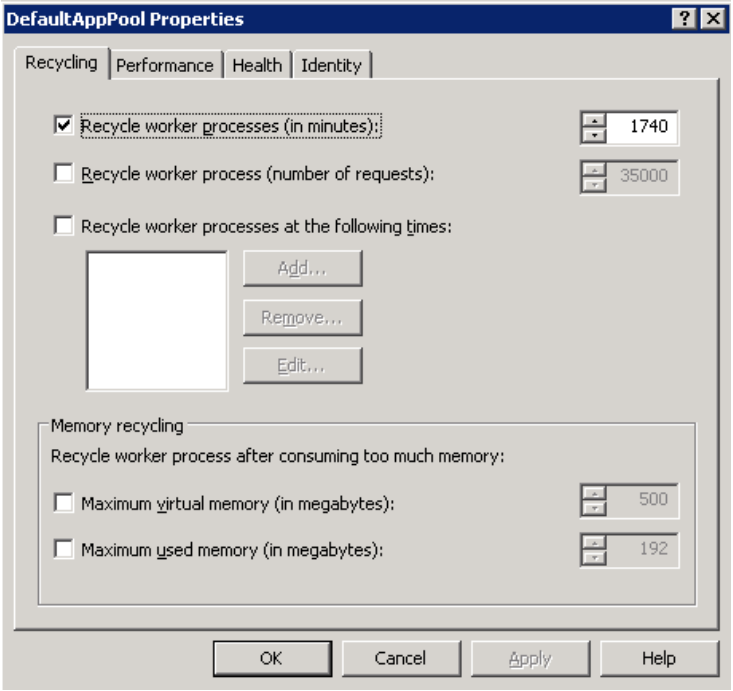


Application pools in IIS 6.0 and earlier

You can check the name of the application pool in the website Properties dialog, on the Home Directory tab. If you're running Kentico CMS in a virtual directory, you will find it in the Virtual Directory Properties dialog, on the Virtual Directory tab.



You can see the **Application Pool Properties** dialog by right-clicking the appropriate application pool and selecting Properties from the context menu. You will see dialog like this:



Recommended Application Pool Configuration

- It's highly recommended that you run Kentico CMS in a separate application pool. If you share the pool with other websites, the system may behave unpredictably.
- If you need to run multiple websites in a single pool, be sure to run only ASP.NET 3.5 applications in the same pool.
- It's recommended that you specify some value in the **Recycle worker** processes on the **Recycling** tab. This value shouldn't be too short (less than **60 minutes**) or too long (more than **1440 minutes**/1 day). Setting this value ensures that the **memory is recycled and the application is automatically recovered from failures** by regular restart. **If your website freezes** time to time, you can temporarily set the value to 30 minutes to ensure that the website is restarted automatically. **Short intervals** may lead to high server load and slow response since after each recycling, the application needs to be restarted and data reloaded to the cache.
- It's recommended that you **do not limit the maximum virtual or used memory**. If you need to use some value, use **at least 100 MB**. If your website is being **restarted too often**, it may be caused by low maximum memory limit. You can check the frequency of application restarts in Kentico CMS Event Log (Site Manager -> Administration -> Event log).
- The **Maximum number of worker processes** on the **Performance** tab must be set to 1. If you set a higher value, the worker processes will not be synchronized and Kentico CMS website will not work correctly. This may lead to **unexpected caching of content and system objects**.
- You can configure the **user account** under which the application runs on the **Identity** tab. This information is useful if you need to troubleshoot issues with **permissions**, such as disk write permissions.
- Kentico CMS **does not support Web garden**. Therefore, the **Maximum number of worker**

processes has to be set to **1**.

3.5.4 Configuration for Medium Trust environment

This topic describes how to run a Kentico CMS website in a Medium Trust Level environment.

Medium trust level

The .NET Framework provides a batch of predefined code access security policies, categorized into several trust levels, which determine the permissions available for applications running on the given machine.

The medium trust level is often used by web hosts on shared servers to prevent applications from accessing certain resources that could be harmful to other websites running on the server. Kentico CMS can be used with the default medium trust policy. To run the system under medium trust, you need to follow certain rules. There are three main components that require higher than medium trust and must be considered in this situation:

- **VirtualPathProvider for .NET 3.5** - retrieves resources (ASCX layouts and transformations) from the database and provides them through a virtual file system.
- Staging module - ensures synchronization of content between production and live site servers.
- Bounced e-mail monitoring - this feature of the newsletters module tracks the amount of unsuccessfully delivered e-mails.

Note

It is recommended to develop websites under *Full trust*. Medium trust environments should only be used for hosting live production websites.

Virtual path provider when running on .NET 3.5

This library handles virtual objects stored in the database that need to be compiled, such as document transformations and page layouts. The system references the files through a virtual path, and the Virtual path provider provides the control code to the compiler.

The .NET 3.5 Virtual path provider cannot run under medium trust (requires AspNetHostingPermission with "high" trust level), so you need to store the physical files on the file system before deploying the website to the production environment.

1. Go to **Site Manager -> Administration -> System -> Virtual objects**.

2. Click **Store all virtual objects in file system**. This saves all virtual objects into files under the following folder:

- **~/CMSVirtualFiles**

The system loads the data from the files on the disk instead of the database. You can still make changes to the objects through the administration interface (they are carried over to the files immediately).

3. When you are ready to deploy the website, copy the web project over FTP to the server (including the content of the *CMSVirtualFiles* folder).

You cannot edit the content of the virtual objects directly on the production website, because the CMS application will usually not be allowed to access the file system on a medium trust server. You need to redeploy the website if you wish to modify the virtual objects.

When using .NET 4.0 or newer, the Virtual path provider runs under medium trust, so the issues described in this section do not occur (you can store virtual objects in the database even on medium trust servers).

Staging (Microsoft Web Services Extensions 3.0)

This section applies only if you're using the Staging module.

The **Microsoft.Web.Services3.dll** library from the Web Services Extensions 3 (WSE) package which is used by the Staging module requires Full trust permissions because of the low level operations related to the communication protocols. To ensure the proper functionality, the library needs to be registered in the Global Assembly Cache (GAC) of the server. The library is provided by Microsoft and most hosting providers pre-install it on their shared servers.

If you manage the server, please follow these steps:

1. Go to **Control panel -> Administrative tools -> Microsoft .NET Framework 2.0 Configuration**.
2. Select the **Assembly cache**, click on **Add an Assembly to the Assembly Cache** and select the *bin\Microsoft.Web.Services3.dll* library file from your web project.
3. Delete the *bin\Microsoft.Web.Services3.dll* file from your web project if it's present.
4. Make sure that your project *web.config* file contains the following item:

```
<system.web>
  ...
  <compilation debug="false" numRecompilesBeforeAppRestart="100">
    <assemblies>
      ...
<add assembly="Microsoft.Web.Services3, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35" />
...
    </assemblies>
  </compilation>
  ...
</system.web>
```

If you are not able to install the library to the GAC or convince your web host to do so, you may still run Kentico CMS under medium trust, but you will not be able to use the Staging module. If this is the case, you will need to manually remove some of the system components by deleting the **bin/Microsoft.Web.Services3.dll** file from your web project if it is present.

After these changes, your system will work correctly in a medium trust environment but you will not be able to use content staging operations.

Bounced e-mail monitoring

This section is only relevant if you wish to use the bounced e-mails feature of the newsletters module.

To be able to check bounced e-mails, the newsletters module makes use of a component that creates outgoing network connections using POP3, a standard e-mail protocol for receiving maildrops from an e-mail server. This component requires the **SocketPermission** for its operation, otherwise it fails when attempting to connect to the server. This permission is denied for applications under medium trust.

If you cannot raise the trust level or create a custom security policy that includes this permission, the only solution is to attempt to convince the hosting administrators to grant the SocketPermission to your application. If you are unable to do so, the bounced e-mail monitoring feature will unfortunately not be functional in a medium trust environment.

Reporting graphs (MS Charts)

The Reporting module utilizes Microsoft Chart Controls to generate its graphs. If your application uses the 3.5 SP1 version of the .NET Framework and is hosted in a medium trust environment, it is necessary to have the MS Chart package installed on the server to ensure that the appropriate library is available in the Global Assembly Cache. The installation executable can be downloaded from the Microsoft website: http://www.microsoft.com/download/en/details.aspx?id=14422

The required assemblies are already included in .NET Framework 4.0, so the installation is not necessary if your application uses this version.

Running the website

Now the system should work under a medium trust level properly. Restart your IIS in order for the configuration changes to take effect and run the website.

If your website uses any third-party components that do not support a medium trust level by default, you may need to configure the system for them. In this case, please contact their author to get the information how to perform the configuration required to run in a medium trust environment.

3.5.5 Configuring Windows Communication Foundation

This topic describes the installation and configuration process of **Windows Communication Foundation (WCF)**. You have to configure WCF for Workflow designer and Chat module to become usable.

WCF overview

WCF is a component of the .NET Framework that provides a programming model for building service-oriented multi-platform applications that communicate across the web. For more information on the technology, visit http://msdn.microsoft.com/en-us/netframework/aa663324.

Installing WCF

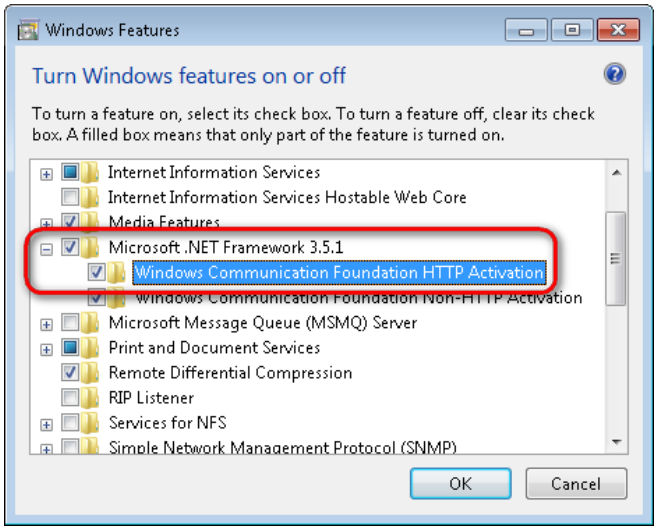
WCF is automatically installed with .NET 3.0 and any higher version. However, you still need to install the WCF HTTP Activation feature yourself:

1. Open the **Start menu**.

2. Navigate to **Settings -> Control Panel -> Programs -> Programs and Features**.

3. Click on **Turn Windows features on or off.**

4. Under the **Microsoft .NET Framework 3.5** node, turn the **Windows Communication Foundation HTTP Activation** checkbox on.



5. Click **OK** to start the installation.

Activating WCF on IIS

If you installed IIS after installing WCF and are running **Windows XP** or **Windows Server 2003,** you have to register script maps in IIS. This will enable IIS to recognize WCF Services (.svc files).

If you are using **.NET framework 3.5**, run the following command from the command line to register script maps in IIS:

```
"%WINDIR%\Microsoft.Net\Framework\v3.5\WFServicesReg.exe" /c
```

Run the following command if you are using **.NET framework 4.0**:

```
"%WINDIR%\Microsoft.Net\Framework\v4.0.30319\ServiceModelReg.exe" -r
```

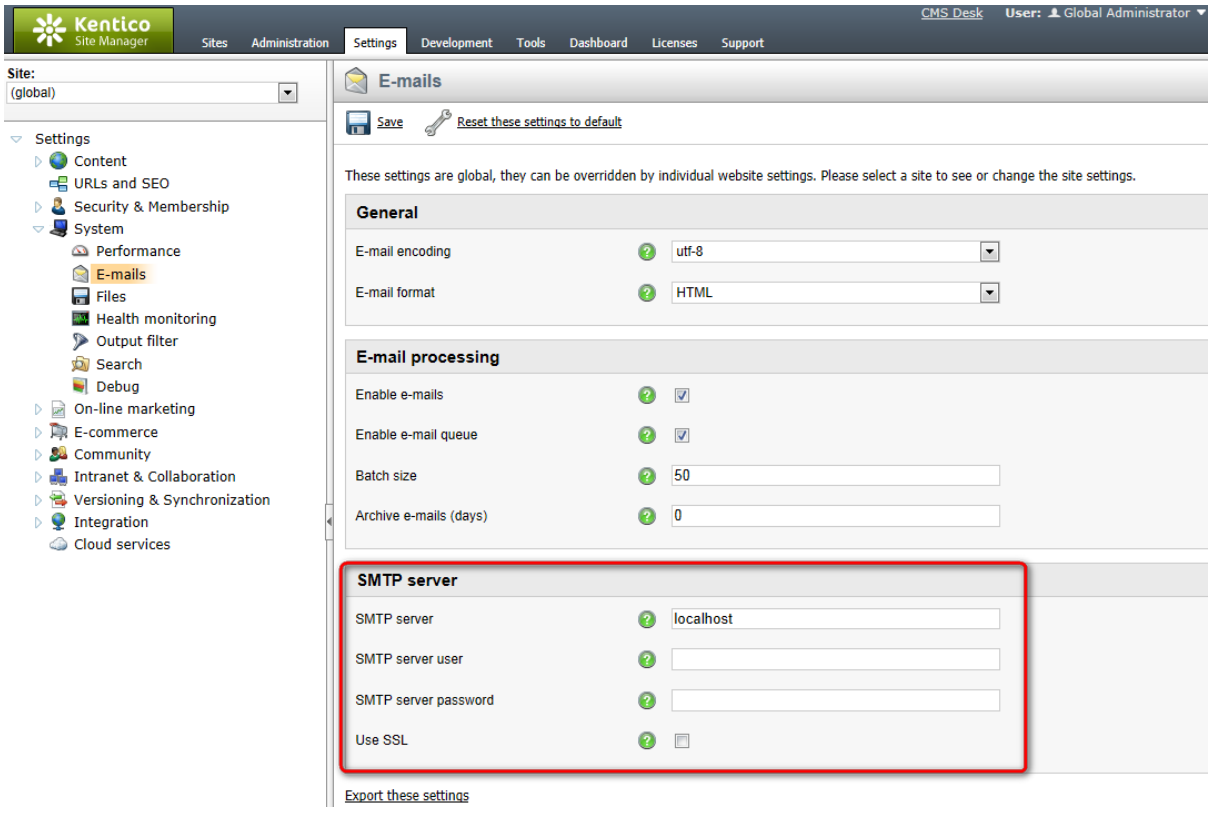
To find more information about troubleshooting problems related to WCF Services hosted by IIS, refer to the following MSDN document: http://msdn.microsoft.com/en-us/library/ms752252.aspx.

3.5.6 SMTP server configuration

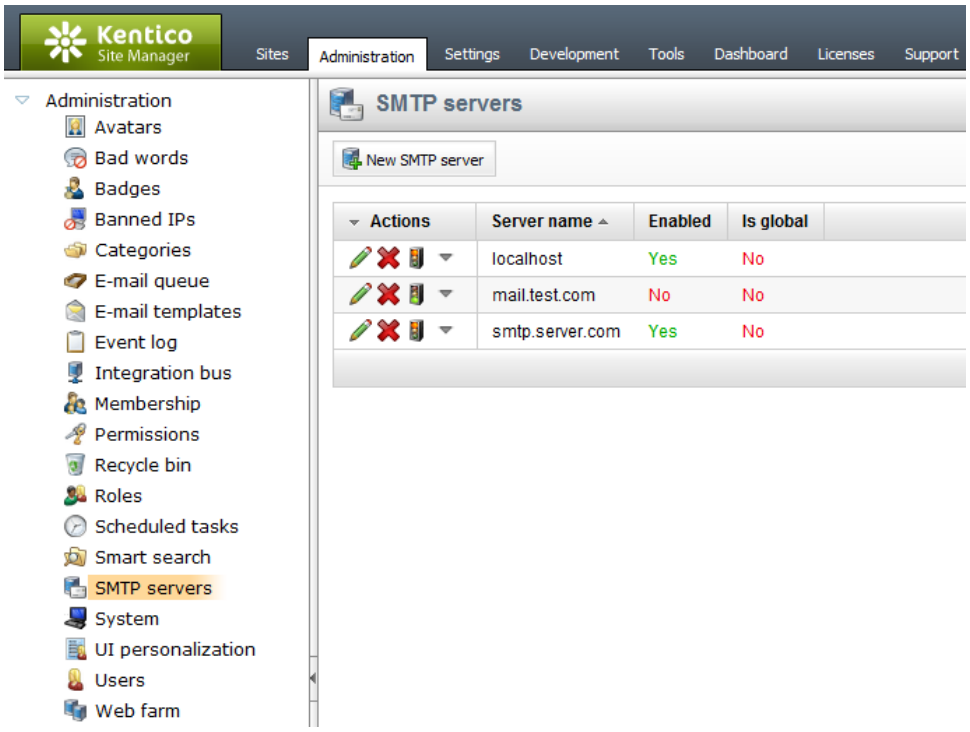
Kentico CMS includes many features and modules that utilize e-mail messages as part of their functionality, including automatic notifications, various types of confirmation messages, subscriptions, newsletters and more. To allow the application to send out e-mails, you need to register and configure SMTP servers for it.

After you finish the New site wizard, you can set up the SMTP servers directly through the administration interface. The website's primary server can be specified in **Site Manager -> Settings -> System -> E-mails**. This server will be used as the default option when sending e-mails. Depending on the value selected in the **Site** drop-down list, the server will either be dedicated to a single website, or designated as a global server (i.e. it will accept e-mails from all sites in the system and also handle mails that are not related to any specific website). You can enter the following values:

- **SMTP server** - must contain the domain name or IP address of the server, including the port number if it is not *25*. Enter *localhost* if you wish to use the server provided by your local machine.
- **SMTP server user** - if the server requires authentication, you can enter the user name here.
- **SMTP server password** - if the server requires authentication, you can enter the password here.
- **Use SSL** - indicates if the SMTP connection to the server should be secured by SSL. The server must be configured to use SSL in order for this to work.

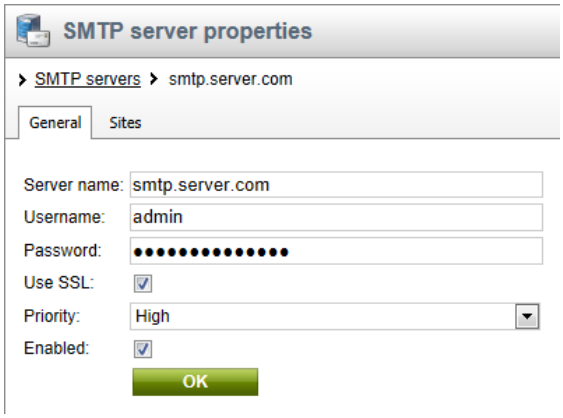


In addition to the default server, you may add any number of servers in **Site Manager -> Administration -> SMTP servers**. Having multiple SMTP servers available for a website allows it to send out a greater volume of e-mail traffic, since the extra servers will be used whenever the default one is busy.



Servers can be defined by clicking **New SMTP server**. Once some servers are added, you can manage them through the appropriate icons in the actions column, i.e. delete () them or edit () their configuration. Servers may also be **Disabled** () or **Enabled** () at any time (within the context of Kentico CMS).

When creating a new SMTP server or editing an existing one on the **General** tab, you can fill in the same options as described above for the default server. In addition, you may also select a **Priority**. Further information about server priority can be found in the section below.



If there are multiple sites running under your application, you may wish to assign different servers to individual websites. You can do so by switching to the **Sites** tab of a server's editing interface, where the given server can either be set as global or limited to one or more specific websites.

E-mail processing

There are several other settings available in **Site Manager -> Settings -> System -> E-mails** that affect how e-mails are delivered to the SMTP servers.

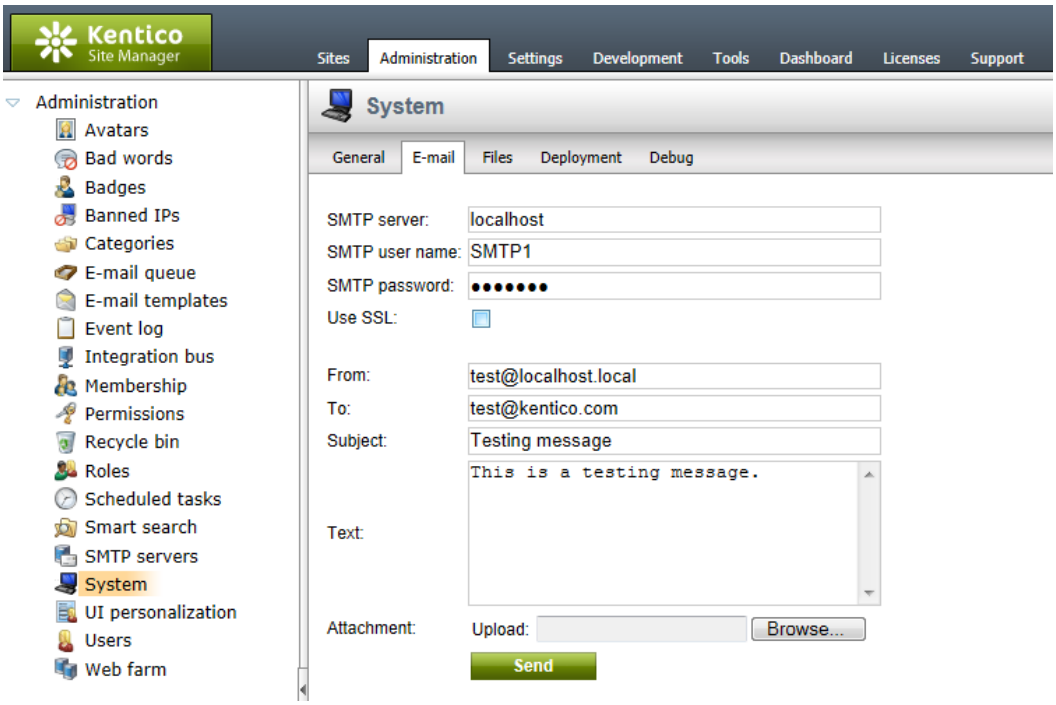
If **Enable e-mail queue** is checked, e-mails generated by the system are temporarily stored in the database instead of being sent directly to the servers. This allows advanced e-mail processing (as described below) and automatic resending of mails that are lost due to errors. Please see the Modules -> E-mail queue chapter of this guide to learn more. Using the e-mail queue is highly recommended when sending out large amounts of e-mails over a short amount of time.

This queue is processed periodically (every minute by default) and the stored e-mails are asynchronously distributed to the appropriate SMTP servers. First, a predefined amount of e-mails is loaded from the queue as a batch. The system then goes through the available SMTP servers in the order of their priority and assigns the batch to the first server that is not busy. The default SMTP server always has the highest priority and other servers are ordered according to the value of their **Priority** property (i.e. batches are first assigned to *High* priority servers, then *Normal* and finally *Low*). When a server receives a batch of e-mails, it is flagged as busy until the sending is complete. A new batch is then loaded from the queue and assigned to the next available server. This process is repeated until all e-mails in the queue are mailed out.

The maximum number of e-mails that can be transferred from the queue to an SMTP server in a single batch is determined by the value of the **Batch size** setting. This setting affects all sites in the system, so it is only available if the *(global)* option is selected from the **Site** drop-down list on the top left of the settings page. Entering an appropriate batch size can significantly affect performance. The ideal batch size depends on the amount of available SMTP servers and their capacity. If it is too large, a server may not be able to process all of the e-mails at once or its send limit may be reached. Using a smaller batch size requires more operations, but it ensures that the load is spread more evenly across multiple servers.

Testing an SMTP server

If you wish to confirm that the values entered for an SMTP server are valid and that it is available, you can send a testing e-mail using the **Site Manager -> Administration -> System -> E-mail** dialog.



3.5.7 Installation on shared hosting server

If you want to deploy Kentico CMS directly to the live server (or generally to a remote server), you need to follow these steps:

1. Install Kentico CMS on your **local machine** if you haven't done so yet.

2. Run the Kentico Web Installer (you can find it in *Start -> All Programs -> Kentico CMS*).

3. Select **I want to install Kentico CMS on a remote (production or testing) server** and enter the path *C:\tempfiles*. Finish the wizard - it only copies the files to the given folder. You do not need to have Visual Studio installed.

4. **Copy the files** from your local folder *C:\tempfiles* to the root of the website using FTP. If you want to use a sub-folder, create a new virtual directory as described in Creating a virtual directory.

5. **Open a web browser** and navigate to your website. The Database Setup wizard starts up.

6. Go through the wizard and create a **new Kentico CMS database** on your live server.

7. At the end of the Database Setup Wizard, choose to either create a new website or import your existing Kentico CMS website.

> **Installation in Medium Trust Level**
>
> If you're installing Kentico CMS in medium-trust environment, please read chapters Installation in medium-trust environment and Configuration for Medium Trust

environment.

Related topic: Deployment to the live server

3.5.8 Installation in medium-trust environment

If you want to deploy Kentico CMS directly to the live server (or generally to a remote server) that uses a **medium trust security level**, you need to follow these steps:

1. Install Kentico CMS on your **local machine** if you haven't done so yet.

2. Run the Kentico Web Installer (you can find it in *Start -> All Programs -> Kentico CMS*).

3. Select **I want to install Kentico CMS on a remote (production or testing) server** and enter the path *C:\tempsite*. Finish the wizard - it only copies the files to the given folder. You do not need to have Visual Studio installed.

4. **Copy the files** from your local folder *C:\tempfiles* to the root of the website using FTP. If you want to use a sub-folder, create a new virtual directory as described in Creating a virtual directory.

5. Make sure the web.config file on your server contains the following value in the **appSettings** section (it specifies that the CMS should use the managed provider):

```
<add key="CMSDirectoryProviderAssembly" value="CMS.DirectoryProviderDotNet" />
```

6. **Open a web browser** and navigate to your website. The Database Setup wizard starts up.

7. Go through the wizard and create a **new Kentico CMS database** on your live server. At the end of the process, you will be asked to update your web.config file manually - please follow the instructions on the screen.

8. Choose to either create a **new website** or **import your existing** Kentico CMS website.

9. If you are using .NET version **3.5**, you need to save all virtual objects to the disk after creating the new website:
   a. Go to **Site Manager -> Administration -> System -> Virtual objects**.
   b. Click **Store all virtual objects in file system**.
   c. If the medium trust server does not allow your application to write on the disk, you first need to create the web project on a local machine using the Web installer, save all virtual objects to the disk and then copy the web project over FTP.

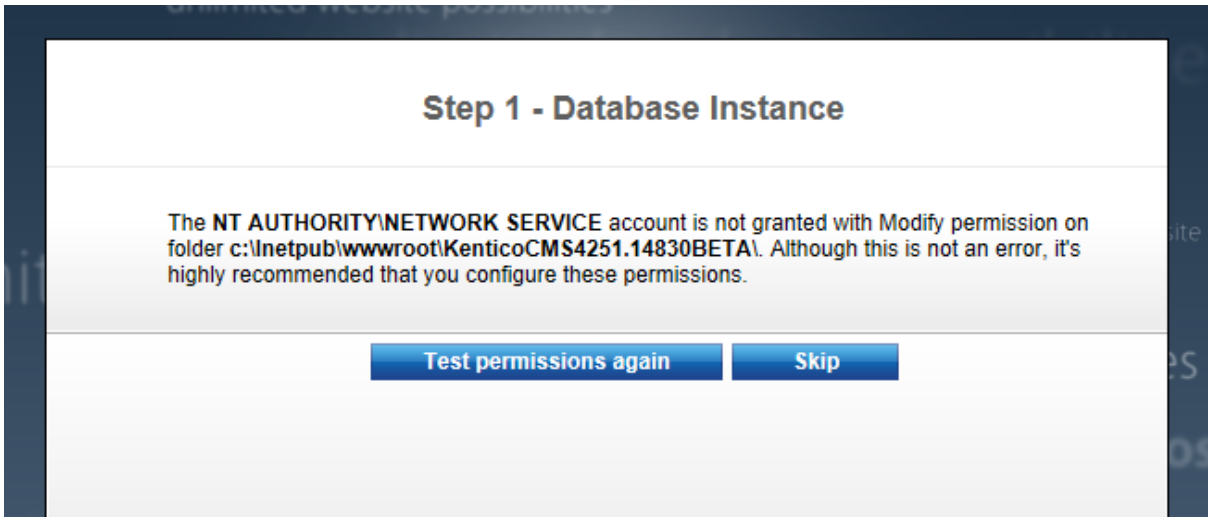
Medium trust environment specifics

For more information about running under the medium trust level, please read the Configuration for Medium Trust environment topic.

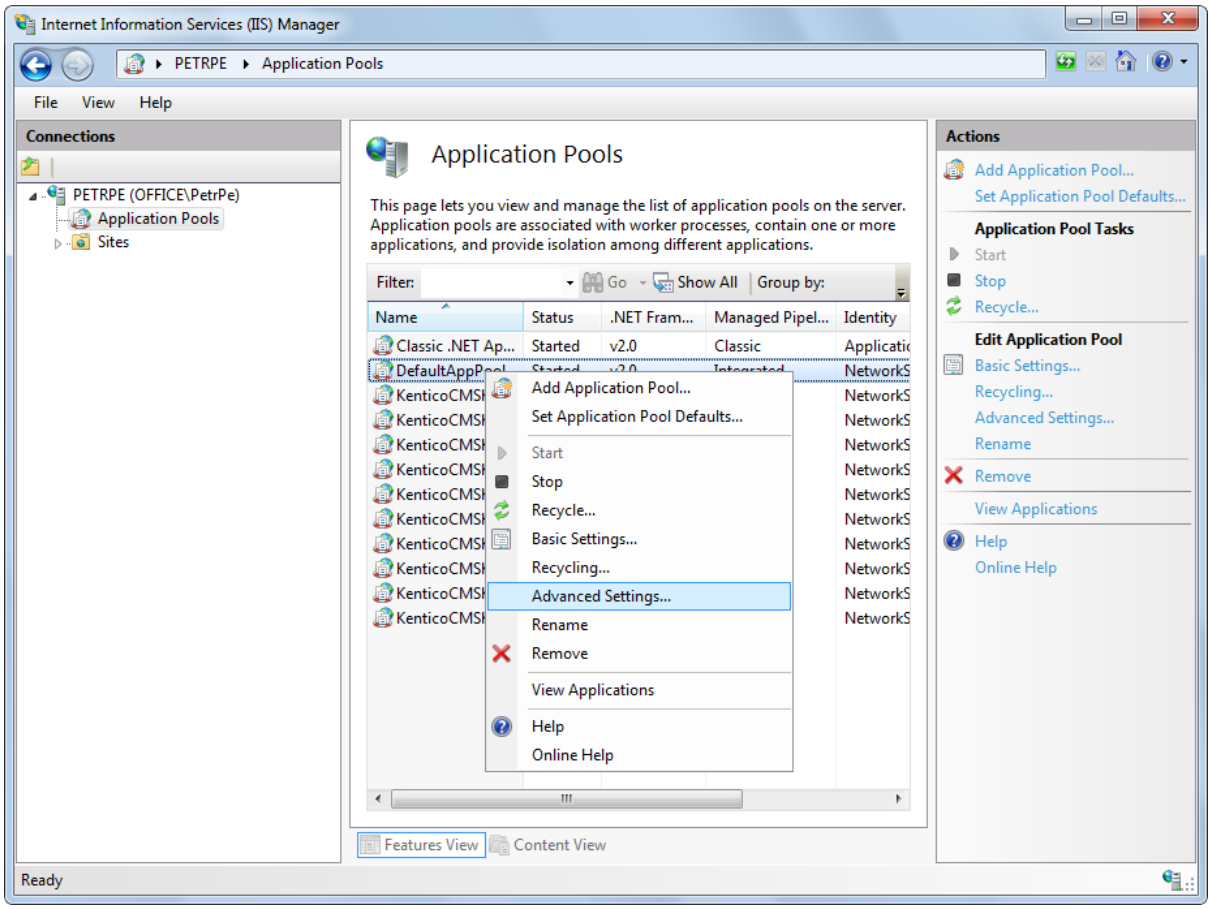
3.5.9 AppPool permissions on Windows 7 or Windows Server 2008 R2

Kentico CMS runs on both **Windows 7** and **Windows Server 2008 R2**. On both systems, you need to add the **Write** permission to the website's folder as described in Troubleshooting installation issues -> Disk permissions problems.

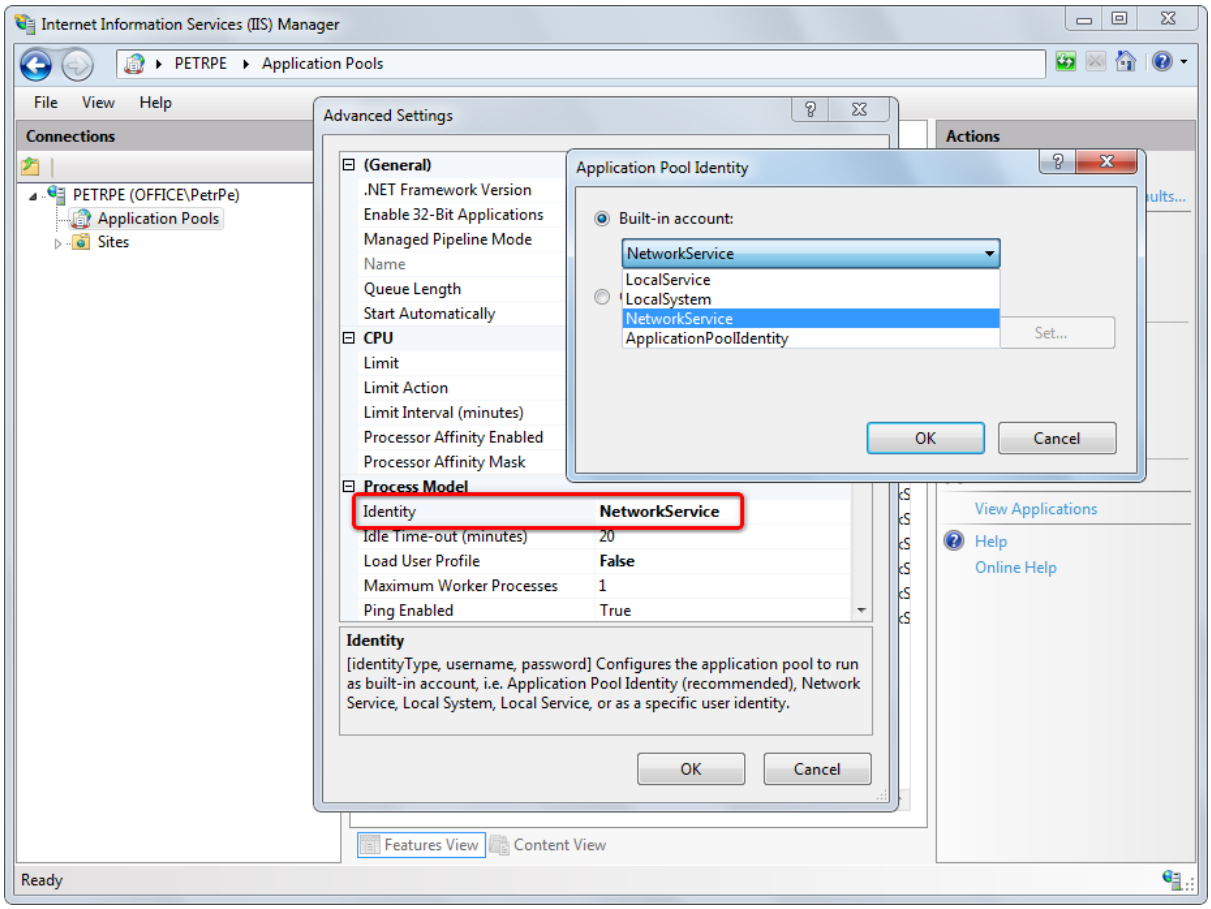
If you can still see the following screen after the end of the Database setup in either of the systems, it's necessary to make some further settings in your IIS.



1. Select the **Application Pools** node in the **Connections** tree in the IIS console. All application pools will be listed. Right-click the **DefaultAppPool** and chose **Advanced settings** from the context menu.

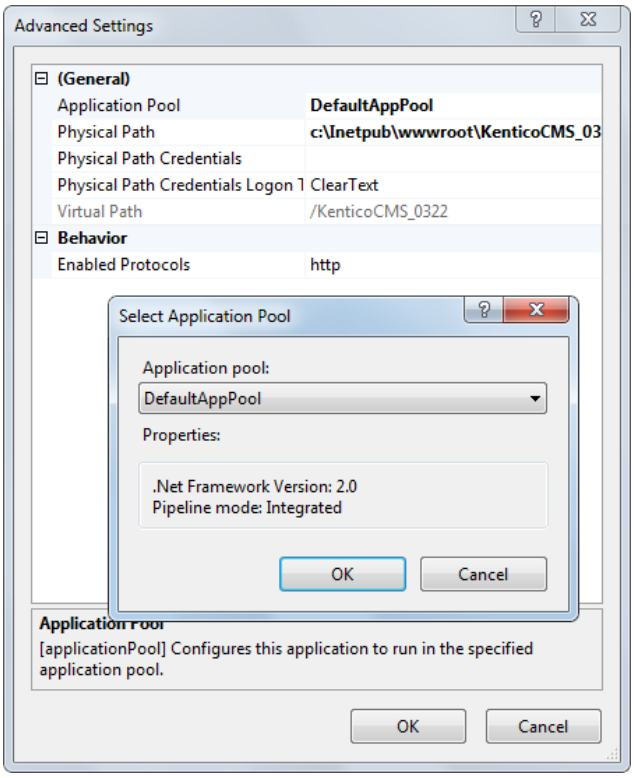


2. In the following dialog, switch the **Identity** value in the **Process Model** category to **Network Service**.



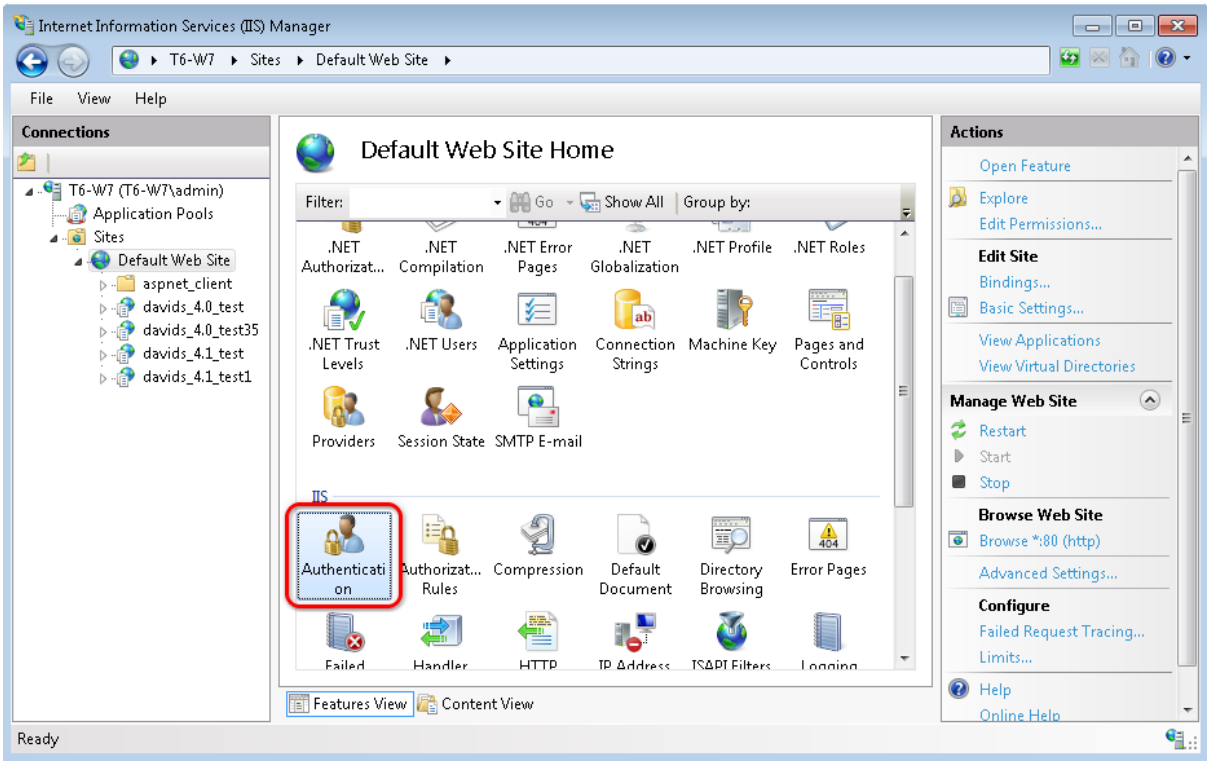
3. Now go to your website's **Advanced settings** ...

4. ... and set the **Application Pool** property to **DefaultAppPool**.

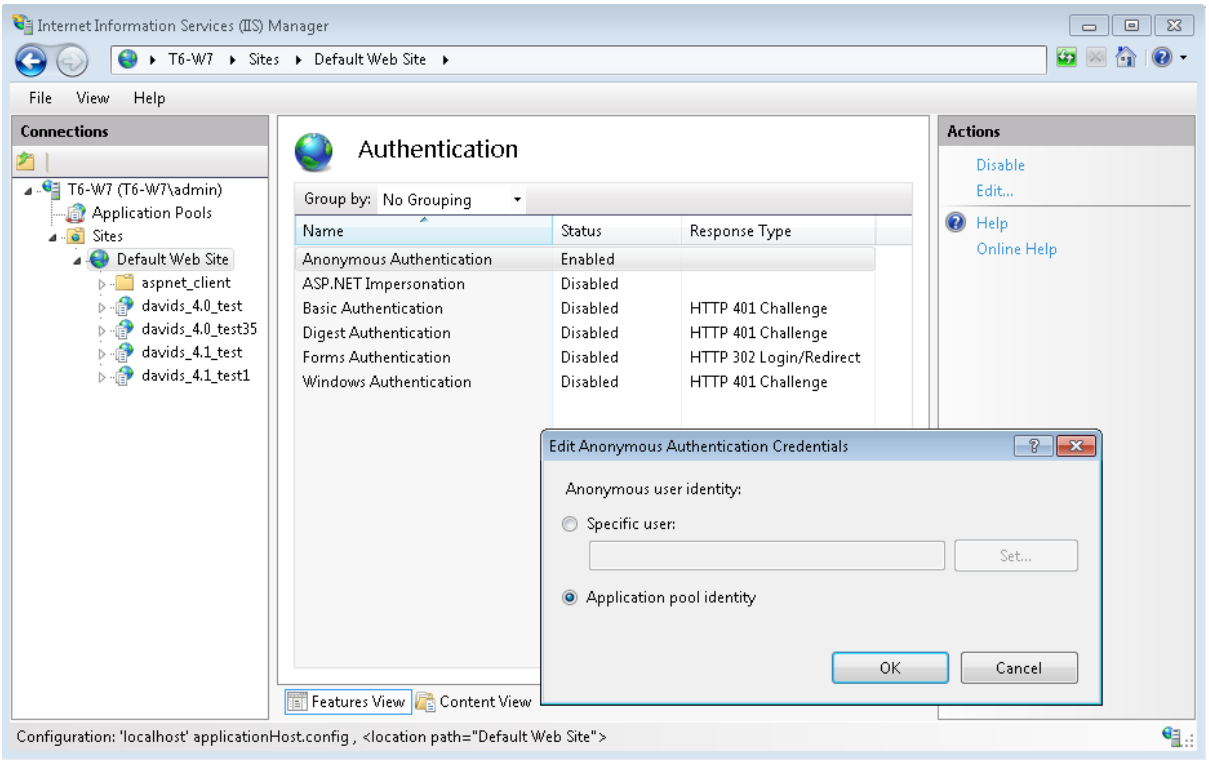


On **Windows 7**, you also need to change the Anonymous authentication settings as described below:

1. Open **IIS manager**, select your site and double-click the **IIS -> Authentication** icon.



2. Select **Anonymous authentication** from the list and click **Edit**. In the pop-up dialog, choose **Application pool identity** and click **OK**.



3.5.10 Configuring custom error pages

There are several possible reasons why you might wish to configure the system to display custom pages instead of standard error messages. This can help reduce the inconvenience caused to visitors if they run into an error while browsing your website, and also improves the security of the site by hiding potentially sensitive internal data (e.g. code in stack traces). You can create custom pages for this purpose with any kind of content, such as an apology or additional instructions, and then configure the system to ensure that they are displayed in the appropriate situations.

Custom Page not found error pages

Because the *Page not found* error (404 HTTP status code) is one of the most common problems encountered by visitors, there are several features that provide a convenient way to set up a custom page that will be displayed as a response. For this type of error, the page can either be a physical *.aspx* file placed under the web project or a dedicated document created in a specific website's content tree.

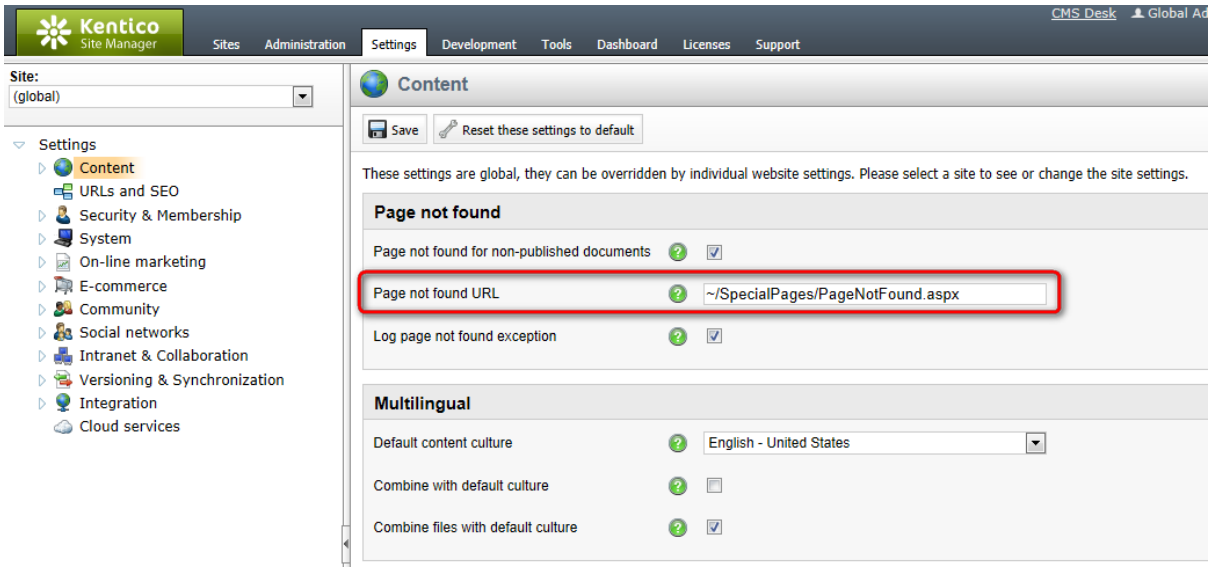
To assign your custom page to a particular website (or globally), simply go to **Site Manager-> Settings -> Content** and enter the URL of the given page as the value of the **Page not found URL** setting, for example:

~/SpecialPages/PageNotFound.aspx

Since there are two possible types of error pages, this value could be interpreted in two different ways. It

could be used to specify the URL of a physical page named *PageNotFound.aspx* located in a folder called *SpecialPages*. If such a file does not exist, the system would try to select a document under the current website with an alias path equal to */SpecialPages/PageNotFound*.

It is recommended to use documents for this purpose. With this approach, you can define the error page's content using the portal engine and leverage all of its features just like you would with any other page. For instance, you can translate the page not found document on a multilingual website and the CMS will automatically display the culture version that matches the currently selected language.



There is no need to manually handle the HTTP response code of the page specified by the setting. It will automatically return a 404 code when accessed (applies to both documents and physical pages). This way, applications, services and web crawlers can identify it as the appropriate type of error page.

Handling 404 errors for general content

To ensure that the system returns your custom *Page not found* error page for invalid requests that target all types of site content, not just the pages processed by the Kentico CMS engine, you need to fulfill several additional requirements.

- The site must be hosted on IIS version 7 or newer.
- The *Managed pipeline mode* of the application pool needs to be set to *Integrated*.

If your environment meets the conditions above, edit your application's **web.config** file, find the **system.webServer** section directly under the root (i.e. not under a specific *<location>* element) and add the following attribute to the **<modules>** element:

```
<modules runAllManagedModulesForAllRequests="true">
```

General error pages

Since Kentico CMS is a standard ASP.NET application, it is possible to configure the handling of all types of errors and exceptions by adding the **<httpErrors>** element into the <*system.webServer*> section of your web.config file as shown below:

```
<system.webServer>
...
  <httpErrors existingResponse="Auto" errorMode="Custom">
    <clear />
    <error statusCode="500" path="/<Virtual_Directory>/CMSMessages/
CustomError.aspx" responseMode="ExecuteURL" />
  </httpErrors>
...
</system.webServer>
```

The **errorMode** attribute sets the basic error page behavior. You may use one of the following possible options:

- **Detailed** - in this mode, the default error pages display details of the encountered error to all users.
- **DetailedLocalOnly** - with this option, detailed error information is only shown to the local host. Simplified error pages are displayed to everyone else, i.e. users who access the site remotely.
- **Custom** - this mode should be used if you wish to replace the default error pages with a custom alternative.

When specifying completely custom error pages, set the **errorMode** to *Custom* and add any number of child **<error>** elements representing individual types of HTTP errors that you wish to handle. It is necessary to enter the HTTP response code of the given error into the **statusCode** attribute and the URL of the appropriate error page as the **path** value. If your Kentico CMS application is running in a virtual directory, replace the <*Virtual_Directory*> string in the sample path value above with the directory's name.

It is recommended to add general error pages directly into your web project as *.aspx* files, e.g. under the **CMSMessages** folder. In this case, the **responseMode** attribute of the corresponding **<error>** elements should be set to *ExecuteURL*. The actual content may be defined to match your specific requirements, but keep in mind that an error page should always return the appropriate *HTTPResponse* status code.

The approach described above works for applications running on IIS 7 or newer with an application pool using *Integrated* Managed pipeline mode. On older versions, you can instead edit the **<customErrors>** element under the <*system.web*> section of the web.config to achieve similar results.

3.5.11 Securing the CMSHelp folder

Kentico CMS comes with an on-line help reference that is available in the database installer and most parts of the administration interface. Users can view it to access context-specific information about the current section of the application's interface. By default, any users (including public) can open the HTML content of the on-line help by entering the appropriate URL, which may not be desirable in certain scenarios, e.g. in the case of high-security websites or if you are creating a rebranded solution.

There are several ways to solve this issue. The simplest is to delete the **~/CMSHelp** folder from the

project of your production website. This removes the possibility of public users opening the help files, but the on-line help in the Kentico CMS administration interface will no longer be available.

If you wish to keep the on-line help on your live website, you can limit access to the content of the help folder so that only users with the appropriate authorization are allowed to view it. Follow the steps below to perform the required configuration:

1. Edit your application's **web.config** file.

2. Find the <*system.webServer*> section directly under the web.config root (i.e. not under a specific <*location*> element).

3. Configure the application to handle the requests for the HTML help files:

a. One option is to add the **runAllManagedModulesForAllRequests** attribute to the <*modules*> element:

```
<system.webServer>
   ...
  <modules runAllManagedModulesForAllRequests="true">
    ...
  </modules>
  ...
</system.webServer>
```

Setting this attribute to *true* ensures that the CMS application processes all types of requests and requires authentication if needed.

b. If you do not want the application to process all additional request types, only **.html** and **.htm**, add the following two handlers into the <*handlers*> element:

```
<handlers>

    ...

    <add name="HTMLRequestHandler" path="*.html" verb="*" modules="IsapiModule"
scriptProcessor="C:\Windows\Microsoft.NET\Framework64\v4.0.30319
\aspnet_isapi.dll" resourceType="Unspecified" preCondition="" />
    <add name="HTMRequestHandler" path="*.htm" verb="*" modules="IsapiModule"
scriptProcessor="C:\Windows\Microsoft.NET\Framework64\v4.0.30319
\aspnet_isapi.dll" resourceType="Unspecified" preCondition="" />

    ...

</handlers>
```

Adjust the path in the **scriptProcessor** attribute as necessary according to your specific .NET environment.

4. Define the authorization rules applied to the content of the **CMSHelp** folder by adding the following section into your **web.config** file:

```
<location path="CMSHelp">
    <system.web>
        <authorization>
            <deny users="?"/>
        </authorization>
    </system.web>
</location>
```

The example above only allows authenticated users to access the on-line help files. Public users cannot reach the files through a direct URL without being prompted to log in. To further increase the security, you can restrict access only for a specific set of roles by editing the <*authorization*> section as shown below:

```
<authorization>
    <allow roles="GlobalAdmin, CMSDeskAdmin"/>
    <deny users="*"/>
</authorization>
```

This ensures that only users who belong to the given roles (specified by their code names) have access.

3.5.12 Installing and configuring WebClient service

The WebClient service is a client prerequisite for connecting to a WebDav server. The service comes pre-installed if you are using desktop versions of Microsoft Windows. However, you need to install it as part of the Desktop Experience feature when using Windows Server 2008 and 2012.

Installing WebClient service on Windows Server 2008 and 2012

1. Start the **Windows Server Manager**.
2. In the tree view, click on the **Features** node.
3. In the details pane, click **Add Features**.
4. In the **Add Features Wizard**, check the **Desktop Experience** box, and then click **Next**.
5. Click **Install**.
6. When the **Add Features Wizard** has finished, click **Close**.
7. Click **Yes** when promoted to restart the computer.

Running WebClient service

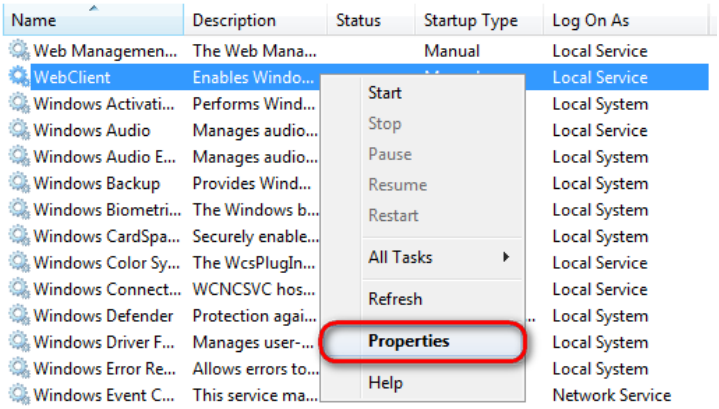
1. Press WinKey + R or type into the search dialog:

services.msc

2. Press Enter. The Services window opens.

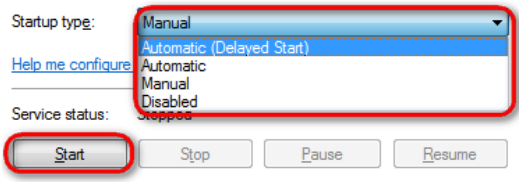
3. Right-click the **WebClient** service.

4. Click on **Properties**. The WebClient properties window opens.



5. If the service is stopped, click on **Start**.

6. Change the **Startup type** to **Automatic**.



7. Click **OK** to confirm the changes.

3.5.13 Database replication

3.5.13.1 Overview

Replication is a set of technologies for copying and distributing data and database objects from one database to another and then synchronizing between databases to maintain consistency. The databases can be running either on separate servers or even on the same server. You will typically use it for back-up purposes (you have two databases with the same content) or in case that you need to spread the load among more databases.

Kentico CMS database can be replicated using the **merge replication**. It can be configured in MS SQL Server Management Studio. The configuration process consists of creating a publication and one or more subscriptions:

- Publication - the database which has the data to offer to the other server
- Subscription - the database which receives updates from the publisher when the data is modified

Detailed information about replication in MS SQL can be found at: http://msdn.microsoft.com/en-us/library/ms151198.aspx.

Limitations caused by DB replication!

Using database replication with Kentico CMS results in the following limitations:

- importing is not functional
- upgrade from previous versions of Kentico CMS to a higher one is not possible
- newly created document types, forms and custom tables can't be synchronized to the subscribers
- field changes in document types and system tables can't be synchronized to the subscribers
- any other additional changes to the database structure are not guaranteed to be synchronized to the subscribers

It is therefore recommended to go through the following procedure when your website is complete, i.e. when you won't need to use any of the limited features.

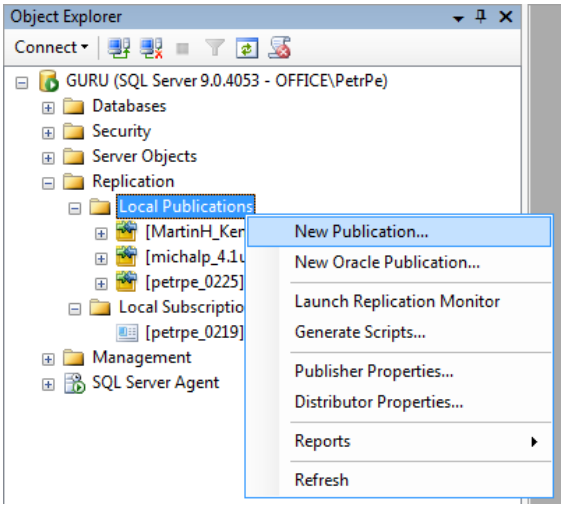
In case that you needed to use any of the limited features listed above, you would need to disestablish your existing replication, make the required changes and set up new database replication from scratch.

If you want to achieve this manually without disestablishing the existing replication, you can follow the instructions on this page.

3.5.13.2 Creating a publication

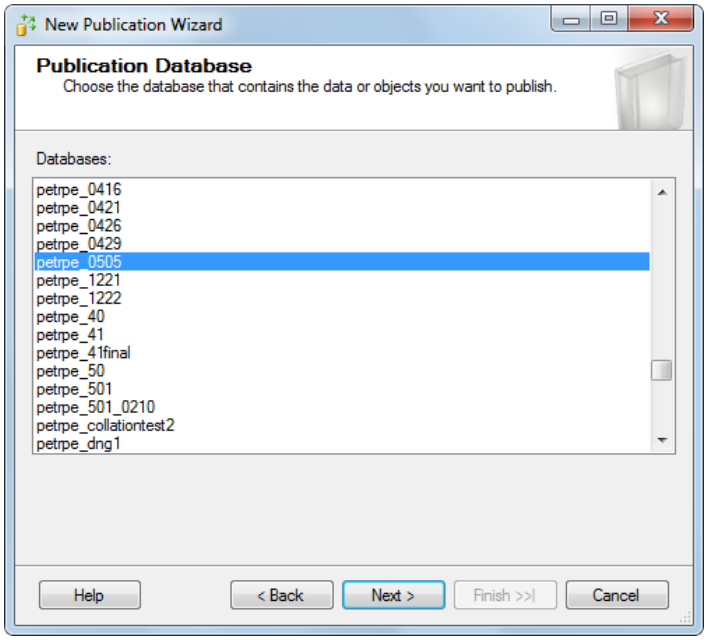
The first logical step in setting up database replication is to create the publication.

1. Open MS SQL Server Management Studio. Expand the **Replication** node, right-click the **Local Publications** folder and select **New Publication** from the context menu.

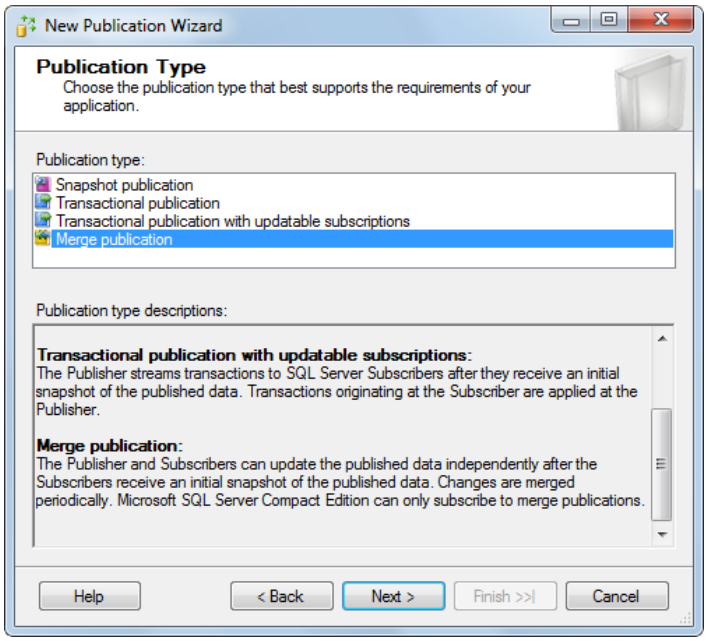


2. The **New Publication Wizard** starts. Click **Next** in the first step.

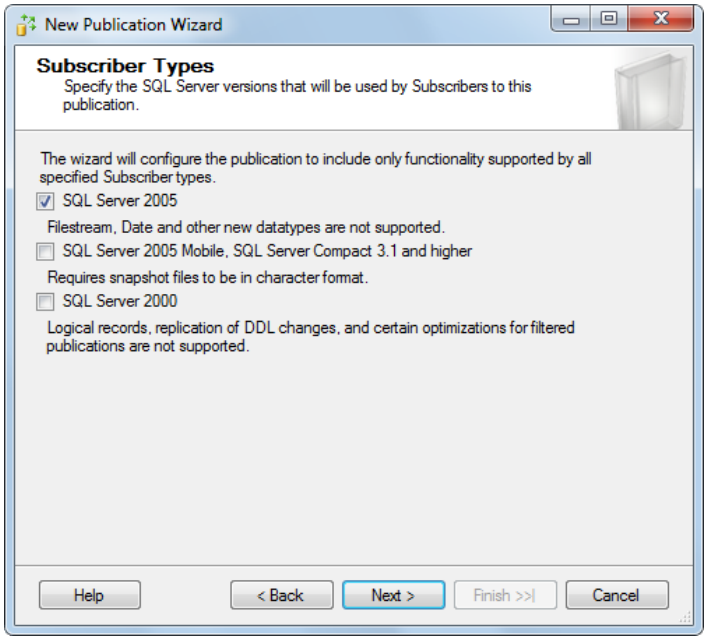
3. Now you need to select the publication database. All databases on the server will be listed, so choose the one which you want to be the publisher and click **Next**.



4. The **Publication Type** step lets you choose the type of replication which you are going to use. Kentico CMS supports merge replication only, so choose **Merge publication** and click **Next**.

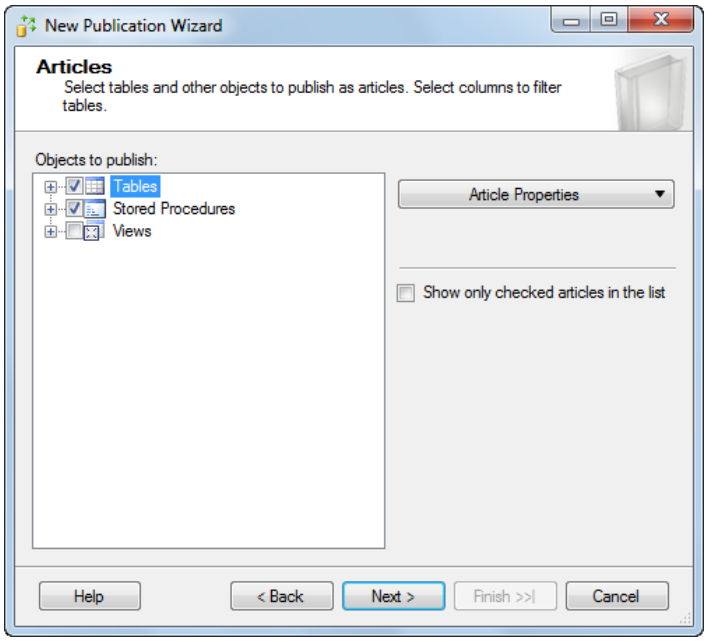


5. This step lets you specify which versions of SQL Server are supported for the subscribers. Choose the versions that your subscribers are running on and click **Next**.



6. This step lets you configure which **Tables** and **Stored Procedures** will be synchronized between the servers. You should not select any **Views** as because changes in metadata which cause changes in any of the Views can't be synchronized automatically (see the note in Overview for more details).

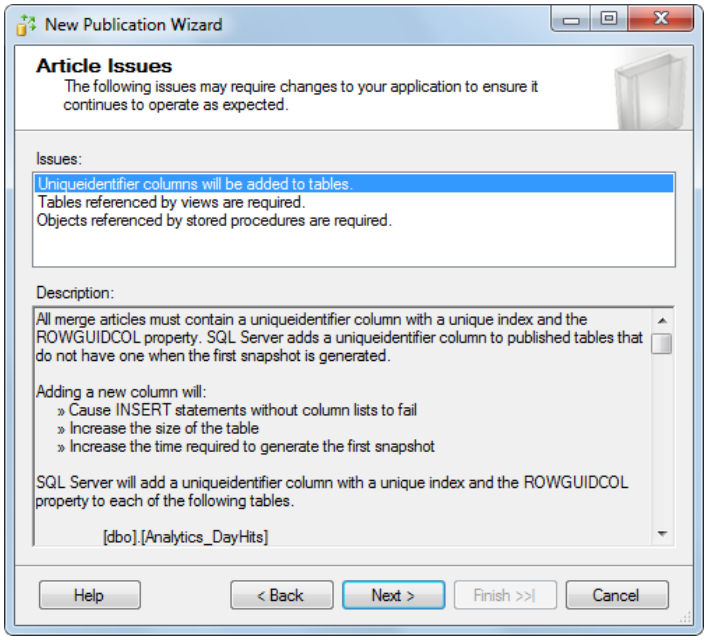
Select the tables and stored procedures that you wish to synchronize (unless you have some specific needs, it makes the most sense to select all) and click **Next**.



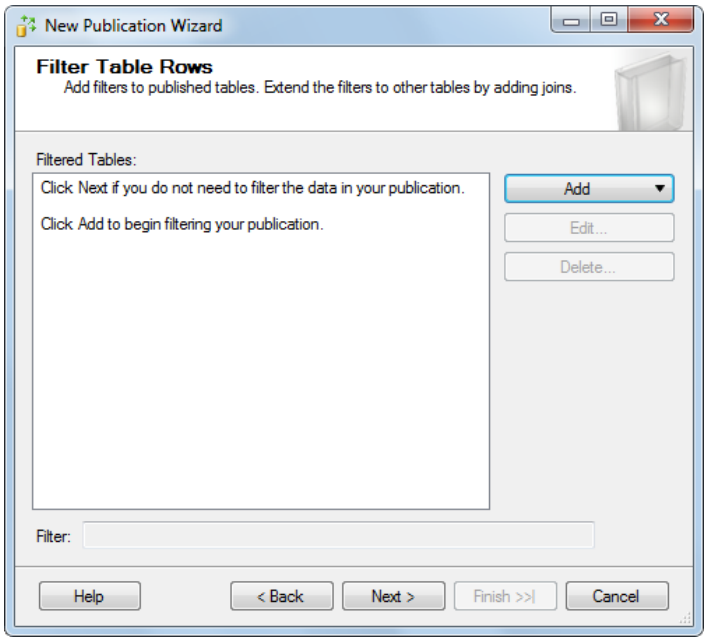
7. The **Arcile Issues** step is only informational - it notifies you about modifications to your database which are required for the application to work as expected with the replication.

The most important modification is that a column named **rowguid** (uniqueidentifier column with a unique index) and the **ROWGUIDCOL** property are added to each table.

Read through the info and click **Next**.



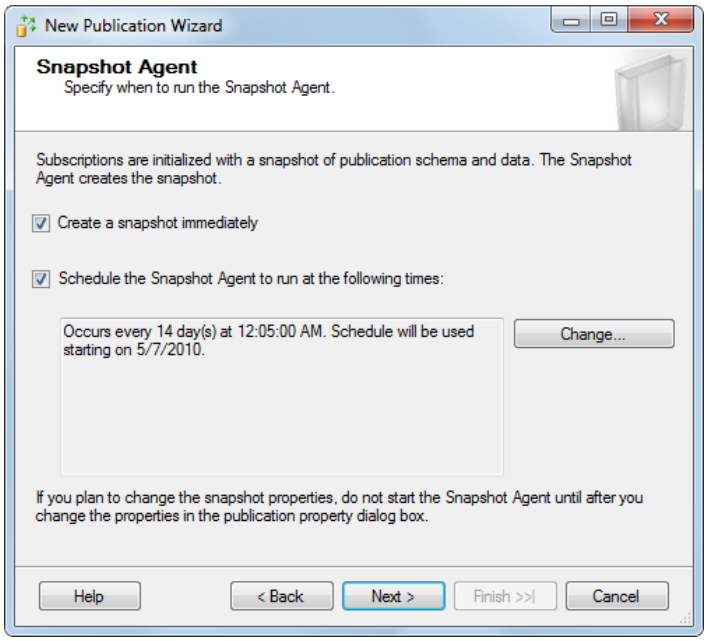
8. In the **Filter Table Rows** step, you can add filters the published tables. This depends on your specific needs, so add filters as needed (or do not add any) and click **Next**.



9. Now you need to configure the **Snapshot Agent** to take a snapshot of the publisher database. The snapshot will be used for the initial content of the subscriber database, which can't be created without it. You have the following two options:

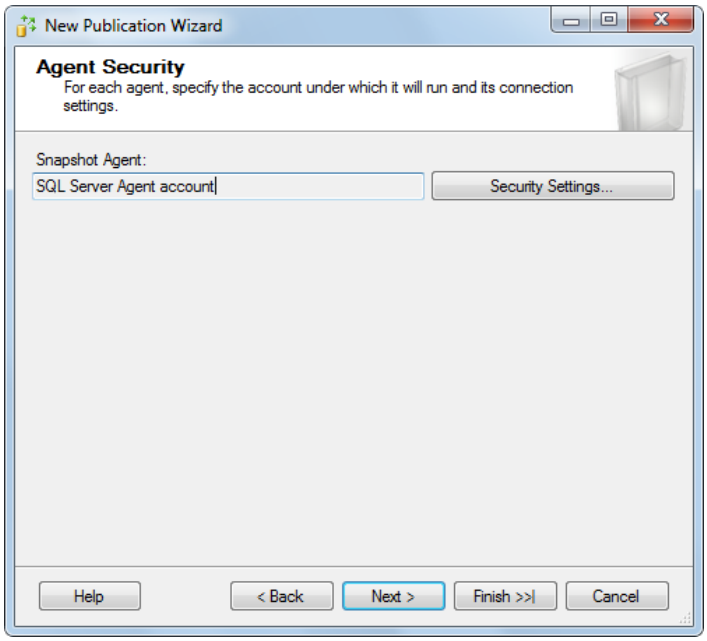
- **Create a snapshot immediately** - if enabled, the Snapshot Agent will take the snapshot immediately after this wizard is completed
- **Schedule the Snapshot Agent to run at the following times** - if enabled, the Snapshot Agent will take the snapshot on a regular basis, which can be specified if you click the *Change* button

Make the selection and click **Next**.



10. In this step, you need to configure the Snapshot Agent's security settings, i.e. how the Snapshot Agent will get authenticated when accessing the database to create the snapshot. The default **SQL Server Agent account** works fine in most cases, but you may need to specify different account in specific cases.

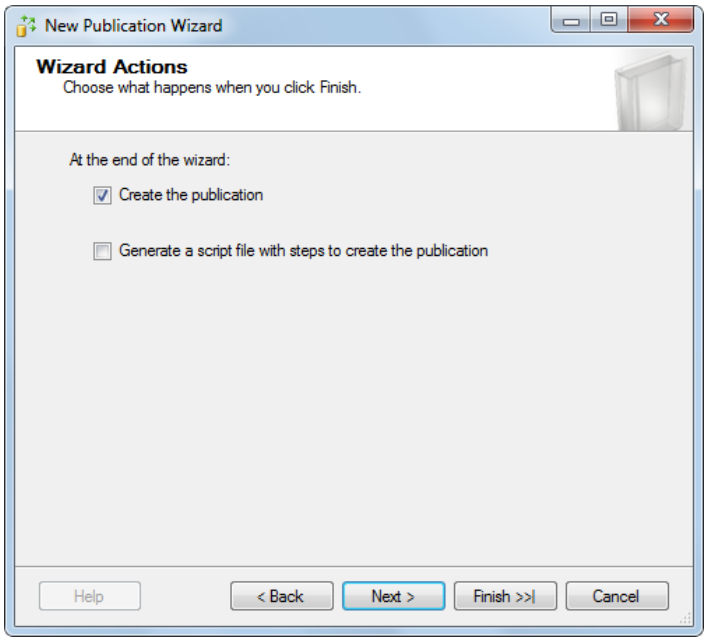
Make a configuration suitable for your environment and click **Next**.



11. The **Wizard Actions** step lets you decide what happens when you finish the wizard. You have the following two options:

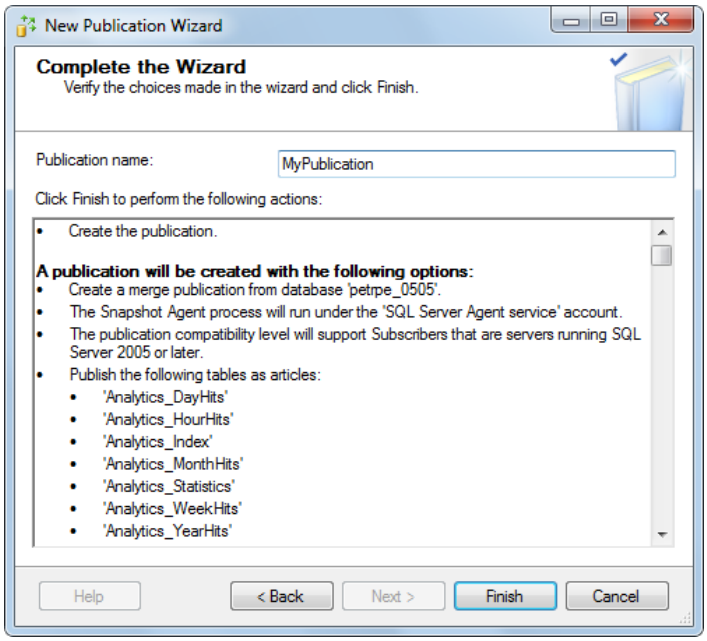
- **Create the publication** - if enabled, the publication will be created as defined throughout the wizard
- **Generate a script file with steps to create the publication** - if enabled, the wizard generates a script which, when executed, creates the publication as defined throughout the wizard

Click **Next**.



12. In the final step, you need to enter the name of your new publication into the **Publication name** field. The section below gives an overview of options which you defined throughout the wizard.

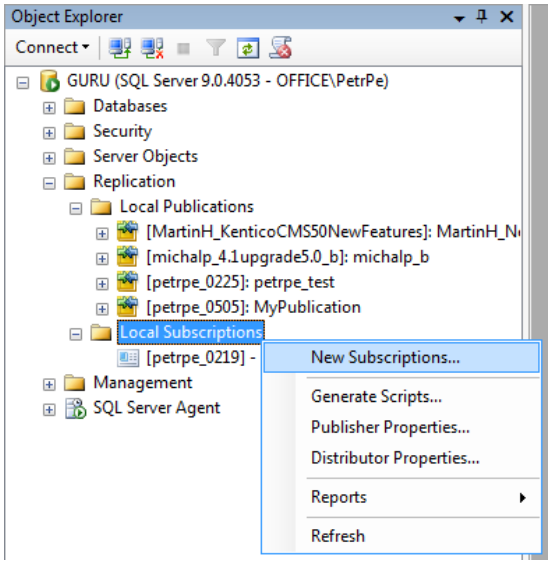
Click **Finish** to create the publication based on the listed options.



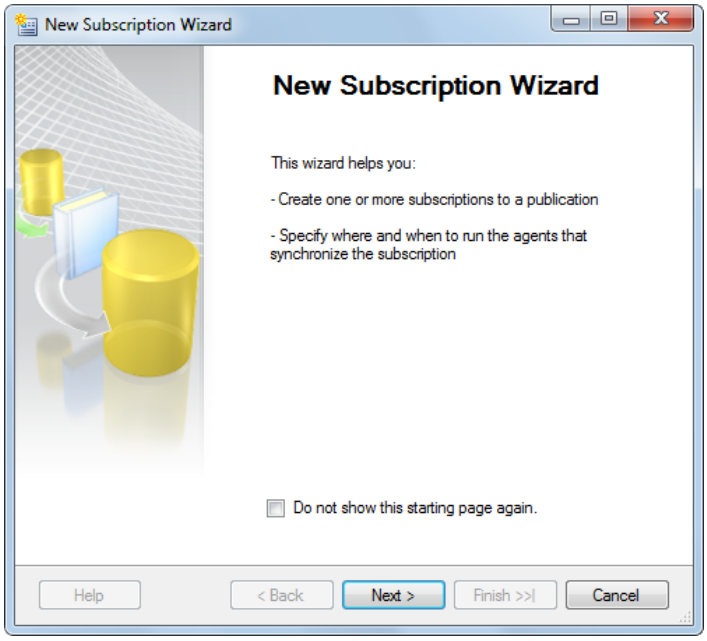
3.5.13.3 Creating a subscription

When you have a publication configured, you can proceed to creating the subscriptions. There can be one or more subscriptions, while you will need a dedicated database for each subscription. Each subscription database needs to be an **exact copy of the publication database**. The copy can be created using backup and restore in SQL Server Management Studio.

1. In **SQL Server Management Studio**, expand the **Replication** node and right-click the **Local Subscriptions** folder. Choose **New Subscriptions** from the context menu.

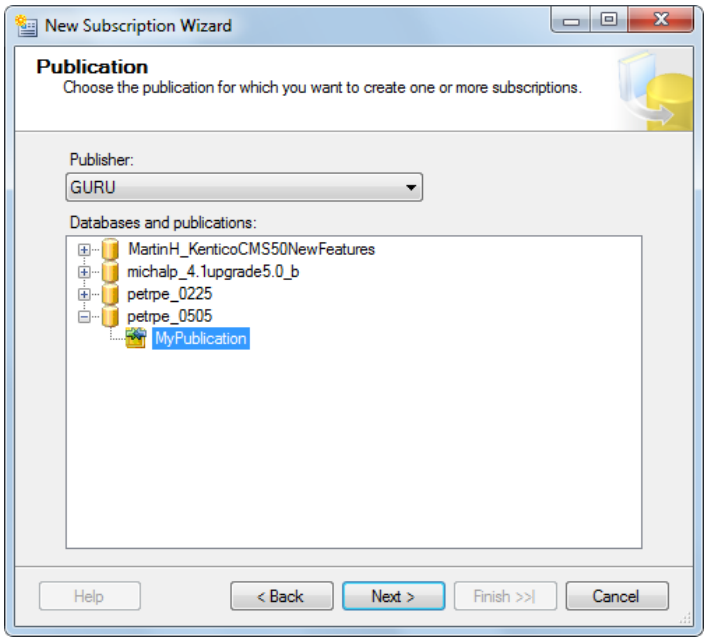


2. The **New Subscription Wizard** starts. Click **Next** in the first step.



3. In the **Publication** step, you need to select the publication to which you want to subscribe. Select the server where the publication is located from the **Publisher** drop-down list. All publications on the selected server will be listed in the **Databases and publications** section below.

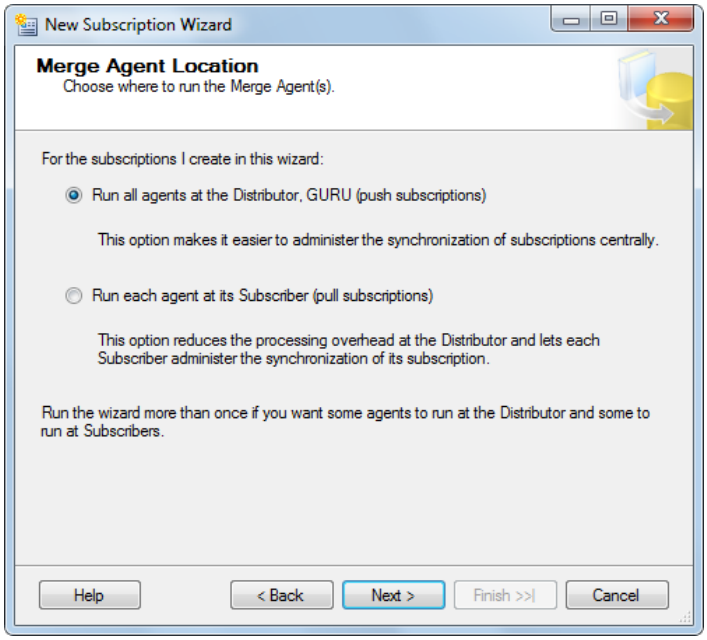
Select the required publication and click **Next**.



4. In this step, you need to decide where the Merge Agent will be run. You have the following two options:

- **Run all agents at the Distributor (push subscription)** - if selected, the Merge Agent will run at the distributor (the server where the publication is)
- **Run each agent at its Subscriber (pull subscription)** - if selected, the Merge Agent will run at the subscriber (the server where the subscription is)

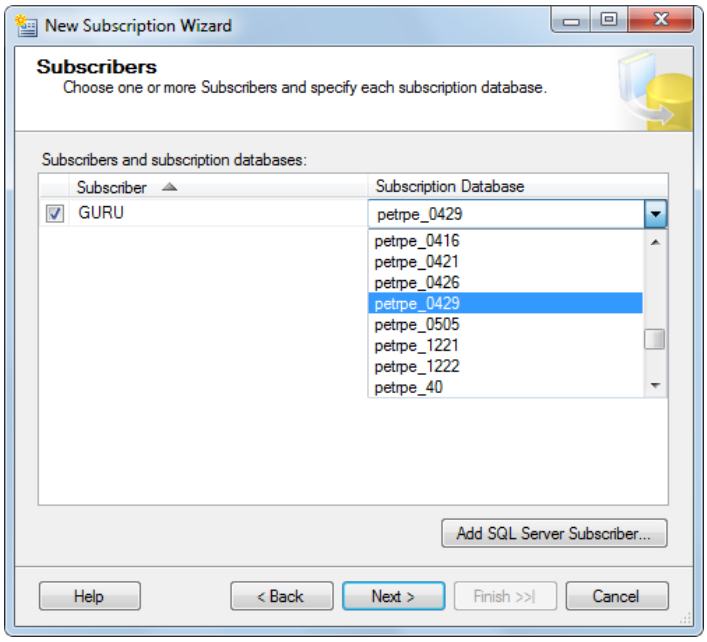
Make you choice (running the agent at the Distributor is recommended for most cases) and click **Next**.



5. Now you need to select the subscription database(s).

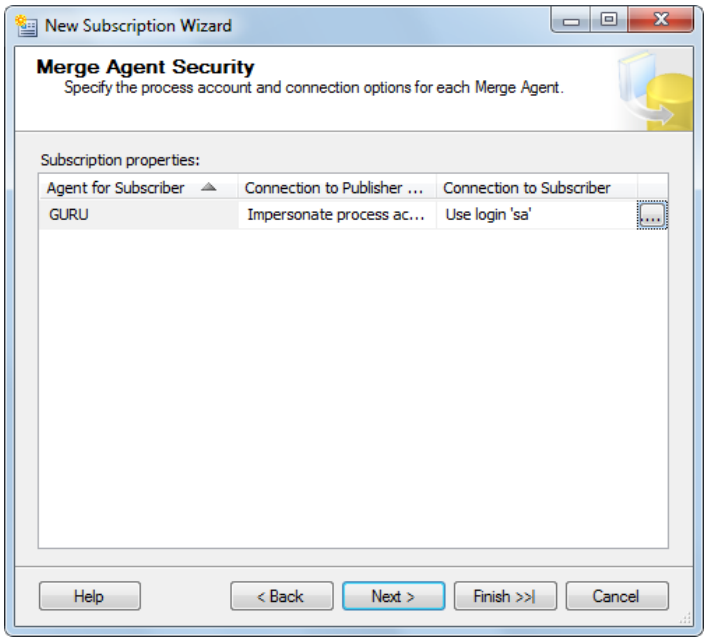
The currently managed database server will be offered in the **Subscriber** column. Using the **Add SQL Server Subscriber** button, you can add other servers to the list, which enables you to create subscriptions on a different server than the currently managed one. Next to each server, you can find a drop-down list containing all databases on the server.

Choose the subscription database(s) that you prepared before starting this wizard (as described at the beginning of this page) and click **Next**.



6. In the **Merge Agent Security** step, you are asked to specify the process account and connection options for the Merge Agent. This can be specified for each server by clicking the appropriate "**...**" button.

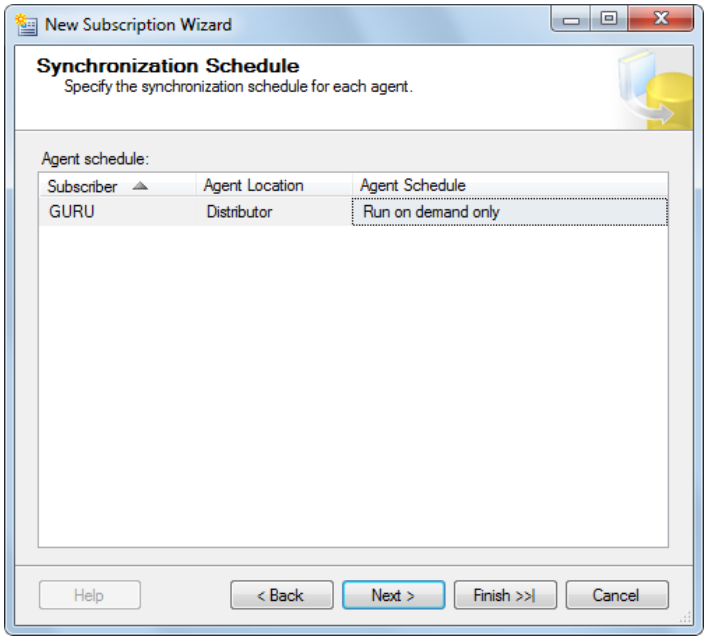
Specify the required information according to your environment and click **Next**.



7. In the **Synchronization Schedule** step, you need to specify when will synchronization between the subscriber and the publisher be performed. You have the following options:

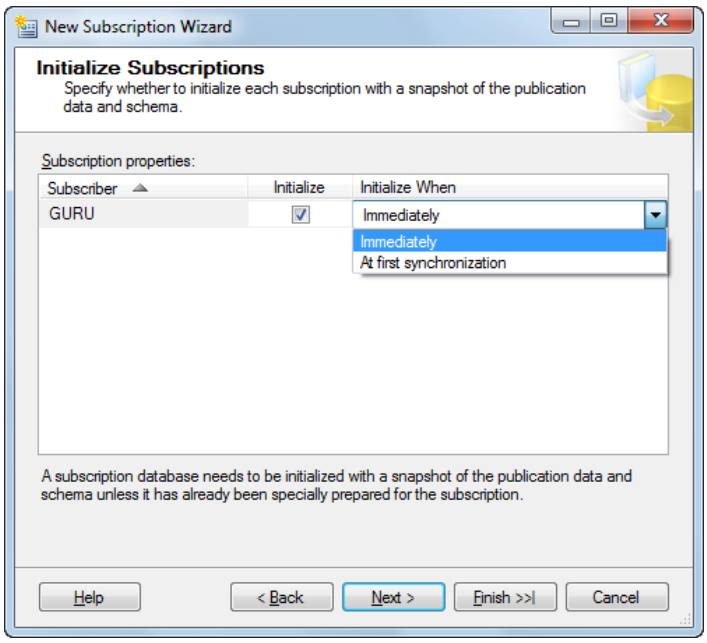
- **Run continuously** - synchronizes immediately whenever a change is made
- **Run on demand** - synchronization is performed only when executed manually from **Replication Monitor** in **SQL Server Management Studio**
- **Run on schedule** - synchronization is performed on a regular basis after a set interval

Select the required schedule and click **Next**.

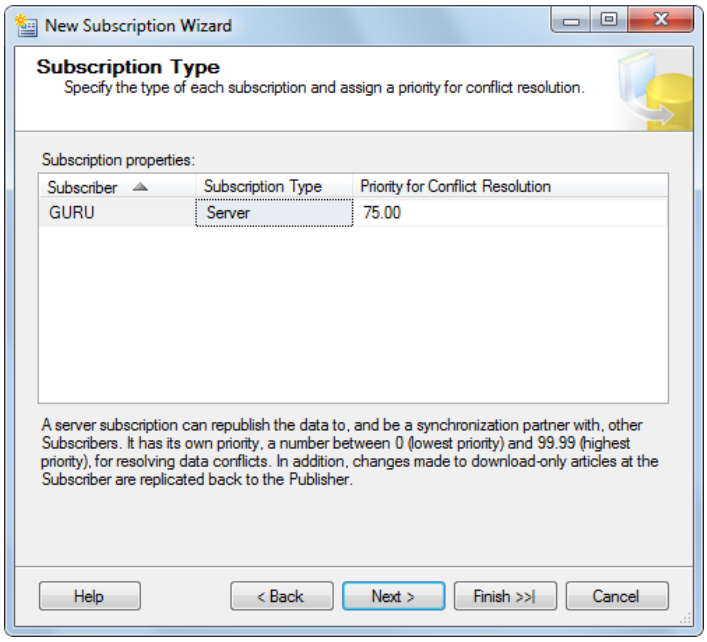


8. This step lets you initialize the subscription, i.e. fill the subscription database with data from the

publisher database snapshot. To perform this, the database snapshot needs to be already created. You can select from the following two options defining when will the synchronization be performed:



9. The **Subscription Type** step lets you decide if you want the subscription to be of the **Server** or **Client** type. For the purposes of our example, leave the default values (*Server* subscription with *75.00* priority) and click **Next**.



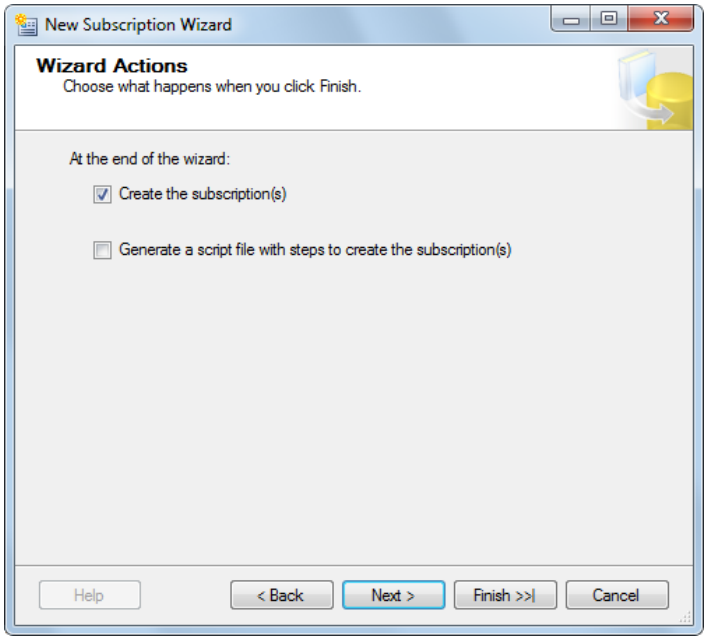
10. The **Wizard Actions** step lets you decide what happens when you finish the wizard. You have the following two options:

- **Create the subscription(s)** - if enabled, the subscription(s) will be created as defined throughout the

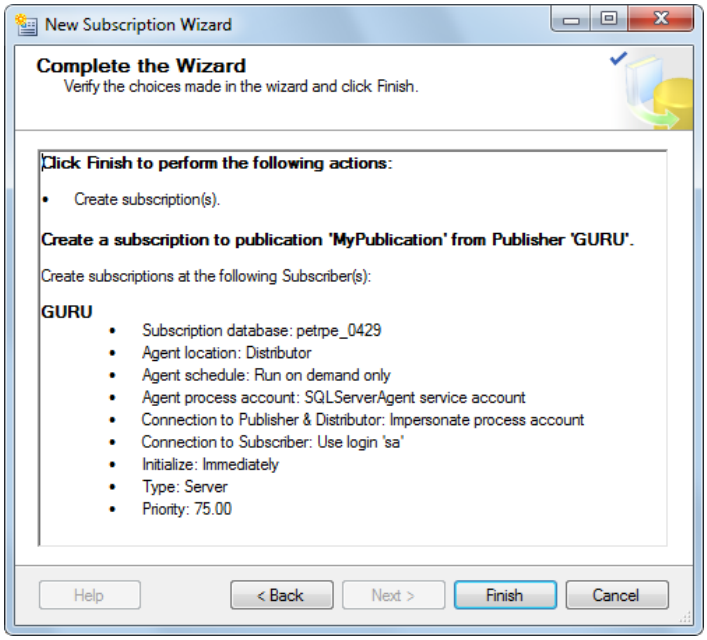
wizard

- **Generate a script file with steps to create the subscription(s)** - if enabled, the wizard generates a script which, when executed, creates the subscription(s) as defined throughout the wizard

Click **Next**.



11. The final step is only informational. It gives you an overview of the options that you selected throughout the wizard. Click **Finish** to create the publication based on the listed options.



3.5.13.4 Modifying structure of a replicated DB

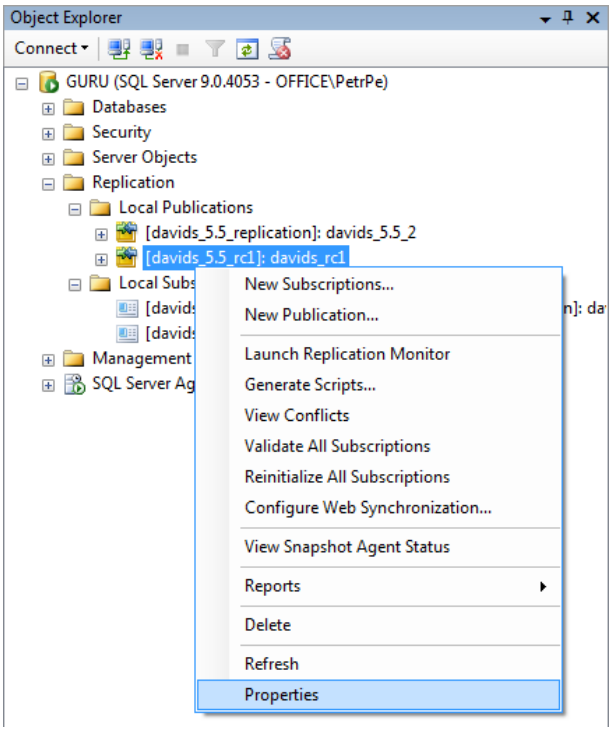
It is possible to make some changes to the DB structure even without disestablishing the existing replication.

Adding forms, custom tables or document types

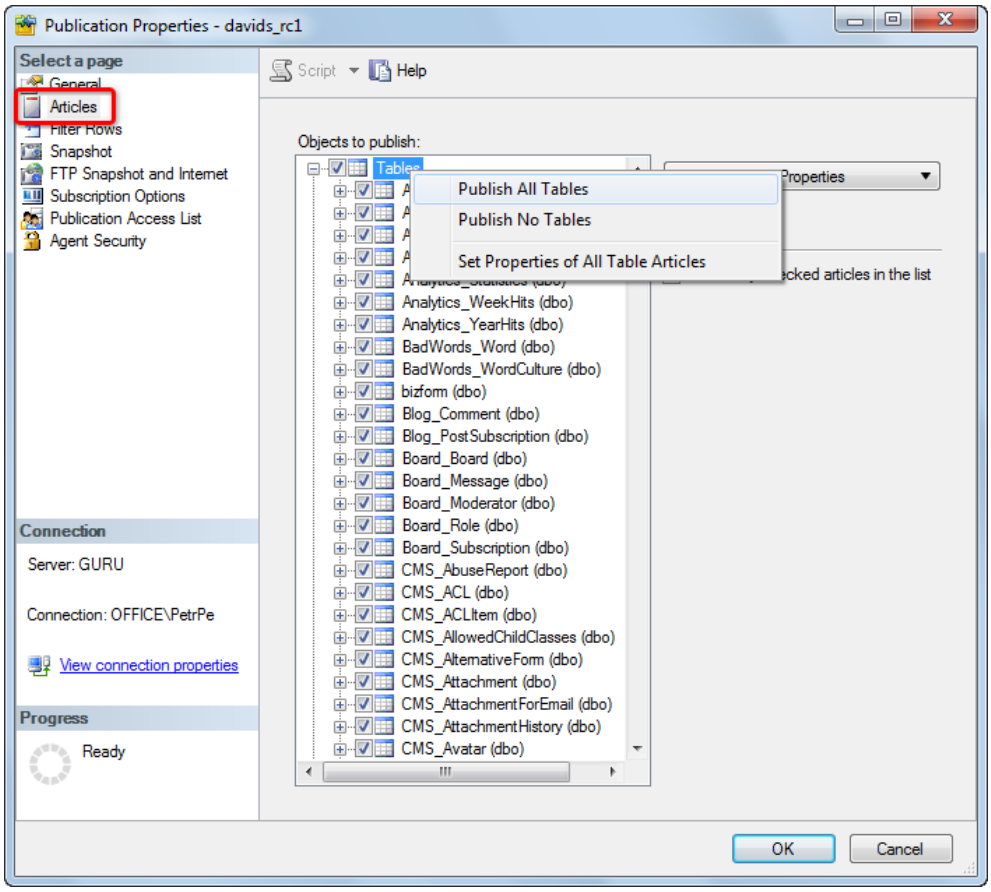
Forms, custom tables and document types can be created even when database replication is established. You only need to follow the steps below to

1. Create the form, custom table or document type from within the publisher's Kentico CMS user interface.

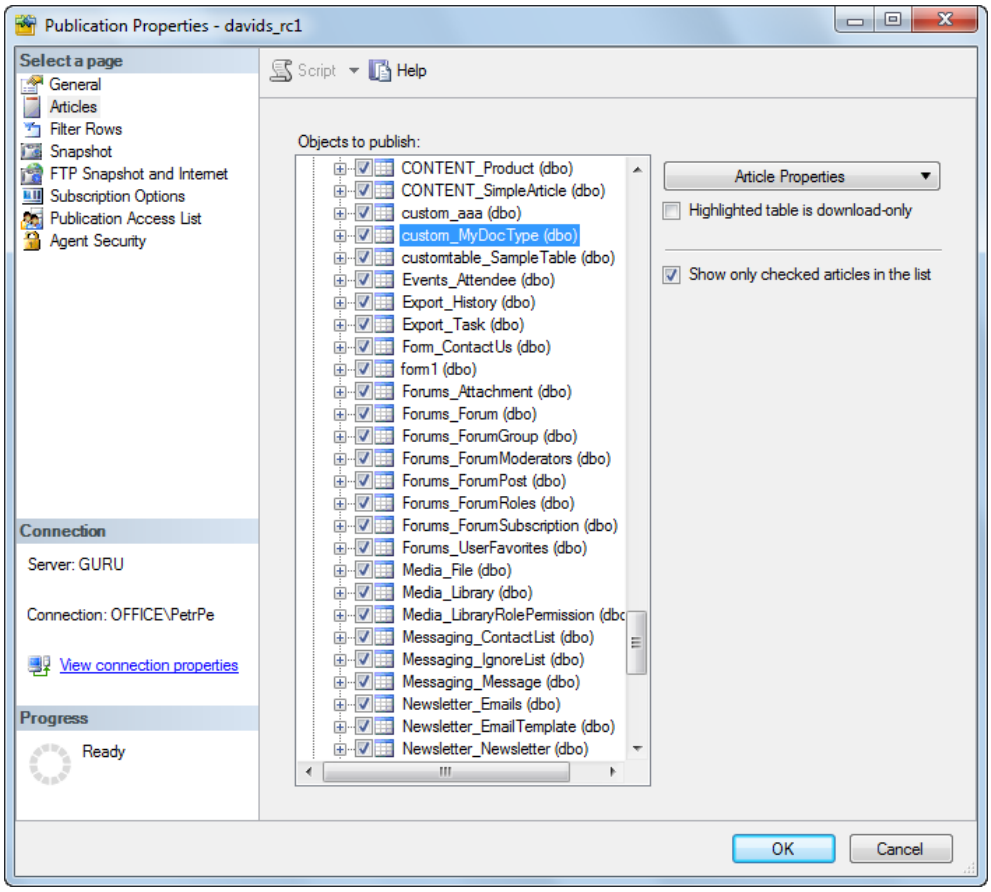
2. Open **SQL Server Management Studio**. In **Object Explorer**, expand the **Replication -> Local publications** node, right-click your publication and choose **Properties** from the context menu.



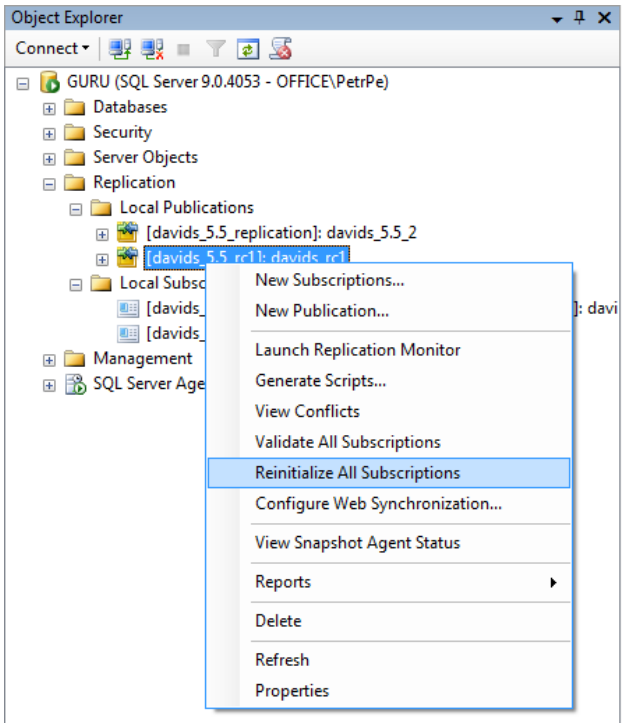
3. In **Publication Properties**, select **Articles** from the right menu. You will see a tree of database tables which are synchronized to the subscribers. You can verify that the new table created in step 1 is missing, i.e. it is not synchronized. Right-click **Tables** (the root of the tree) and click **Publish All Tables**.



4. In case that a new table is found, you are displayed with a confirmation window where you need to confirm that you really want to add the table (article) to the publication. Click **Yes**. The table will be added and you should be able to find it in the tree. Click **OK**.



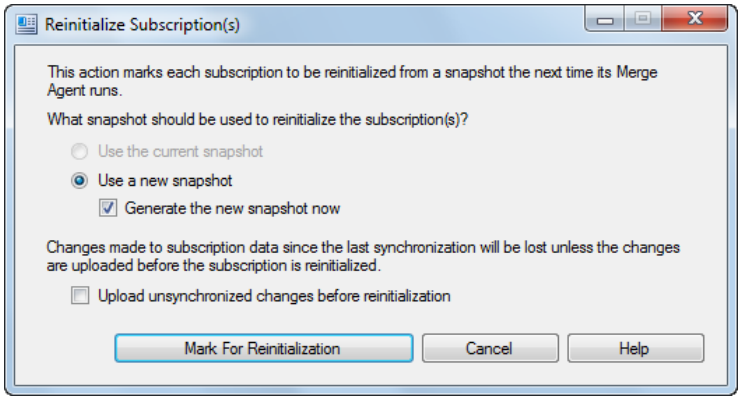
5. Now back in **Object Explorer**, right-click the publication again and choose **Reinitialize All Subscriptions** from the context menu.



6. A dialog window will be raised, telling you that all subscriptions need to be reinitialized from a snapshot the next time its Merge Agent runs. For the changes to be reflected, you need to reinitialize the subscriptions from a snapshot of the new database structure, i.e. from a snapshot which has been taken after steps 1-4 have been performed.

If you haven't created the snapshot since you finished step 4, choose **Use a new snapshot** and **Generate the new snapshot now**. If you already have the snapshot, choose **Use the current snapshot**.

Click **Mark For Reinitialization**. Next time the subscriptions' Merge Agent runs, the subscriptions will be reinitialized with the new database structure.



Field changes in system tables

It is also possible to make changes to the fields of System tables without disestablishing an existing replication. In this case, you need to replace all views on the subscription databases with views from the

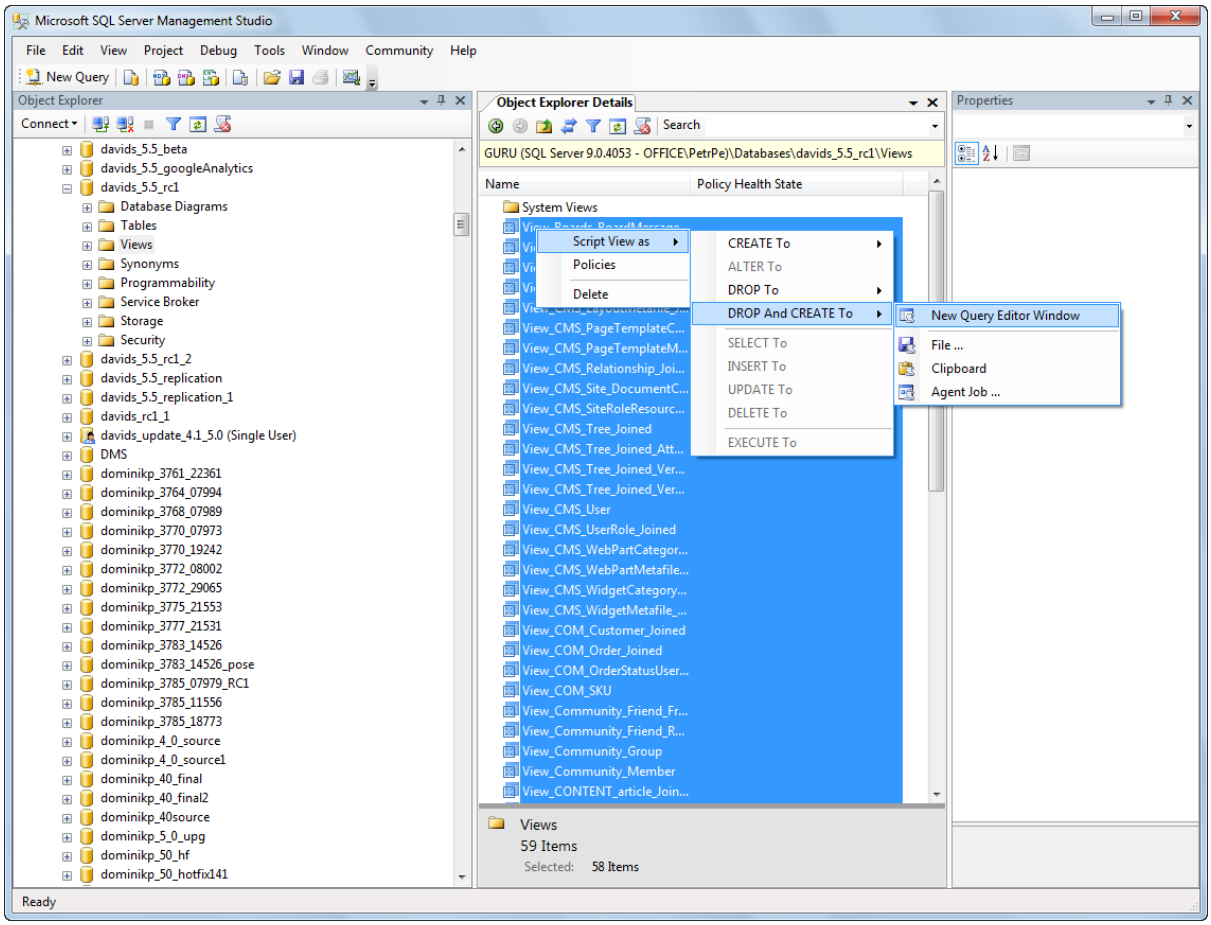
modified publisher database.

The easiest way to do this is to generate a DROP And CREATE script on the publication database and run it on the subscription databases.

1. Click **F7** to open **Object Explorer Details** view and select the **Views** folder of the publication database.

2. Select all listed views, right-click and select **Script View as -> DROP And CREATE To -> New Query Editor Window** from the context menu.

3. Execute the generated script on the subscription databases.



Field changes in document types

Changes to document type fields can also be made without disestablishing an existing replication. The solution in this case is almost identical to the solution for system tables (described above). The only difference is that you don't need to DROP And CREATE all views, but only the view related to the particular document type.

3.5.14 Configuration of full-text search in files

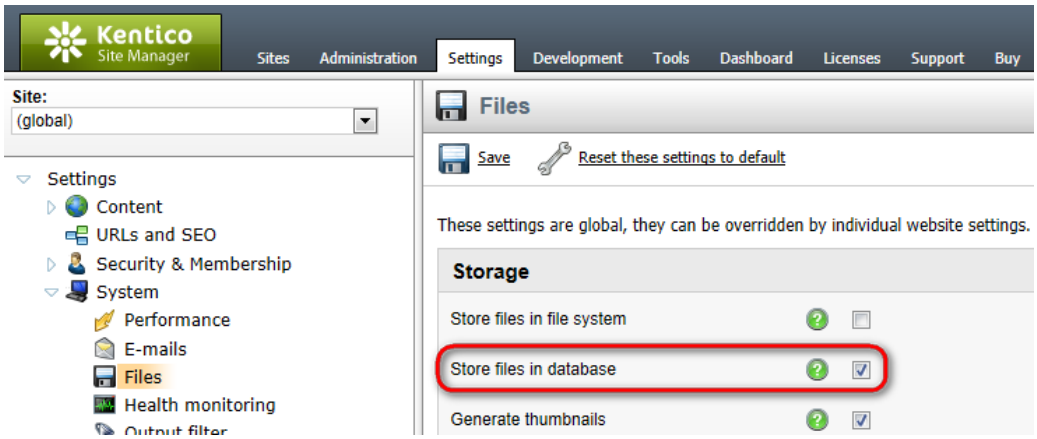
3.5.14.1 Overview

Kentico CMS allows you to search the content of files (document attachments) uploaded into the database. The attachment search uses the standard Microsoft SQL Server full-text search engine, which supports a wide variety of files. The search is available for all supported versions of SQL Server:

- SQL Server 2005
- SQL Server 2008
- SQL Server 2008 R2
- SQL Server 2012

Prerequisites

- Full-text search support must be installed on your SQL Server. The full-text search is available for all supported editions of Microsoft SQL Server, including the Express Edition with Advanced Services.
- Your Kentico CMS website must be configured for storing files in the database (**Site Manager -> Settings -> System -> Files -> Store files in database**).



Use one of the following guides to configure your installation of Kentico CMS for search in files:

- Manually configuring full-text search on MSSQL Server
- Enabling full-text search on MSSQL Server (Script)

Supported file types

The standard full-text search engine delivered with Microsoft SQL Server can search TXT, HTML, DOC, XLS and PPT files.

- To search in PDF files, install a free driver from Adobe: Searching PDF files
- To search Microsoft Office Open XML documents, such as DOCX or XLSX, install a free IFilter pack: Searching Microsoft Office documents

If you want to search other types of documents, you need to install appropriate IFilter

libraries. You can download or purchase IFilter libraries from third-party vendors.

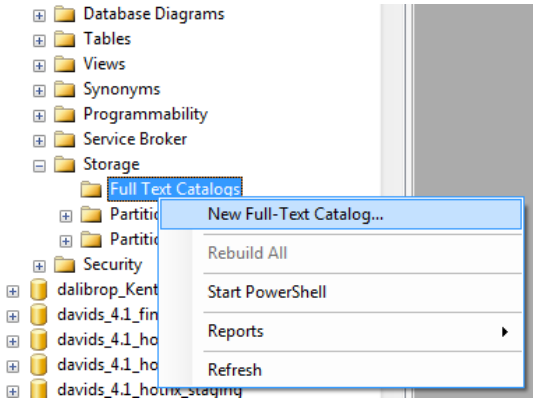
3.5.14.2 Manually configuring full-text search on MSSQL Server

Use the following steps to configure your Kentico CMS database for full-text search in file attachments:

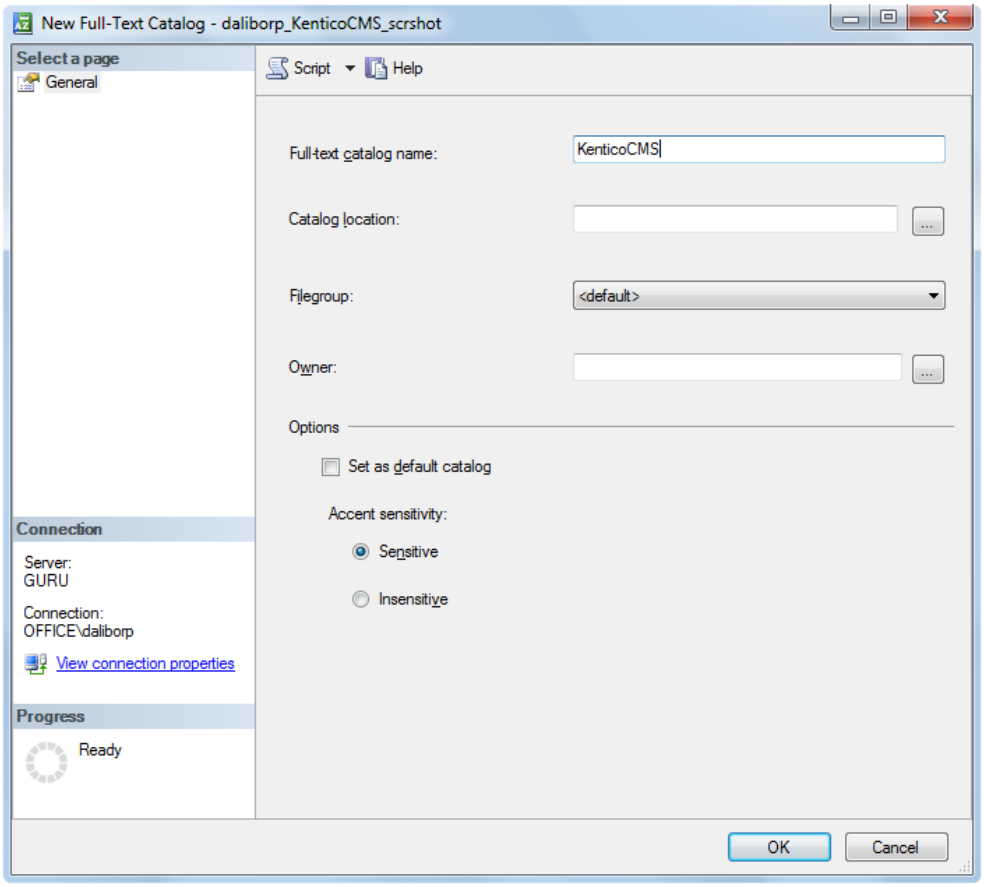
1. Start **Microsoft SQL Server Management Studio**.
   - If you cannot use SQL Server Management Studio on your database server, you can configure the full-text search through a script instead.

2. Locate your Kentico CMS database.

3. Unfold the **Storage** sub-folder, right-click **Full Text Catalogs** and click **New Full-Text Catalog**.



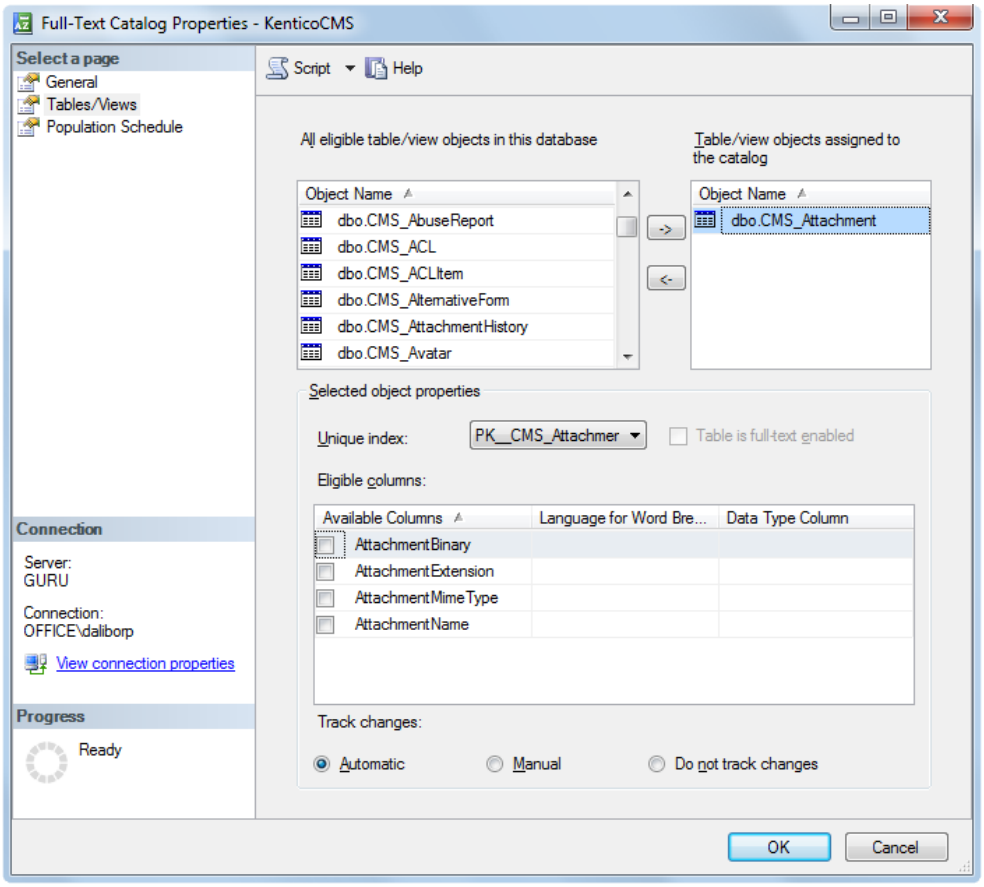
4. Type a **Full-text catalog name** and click **OK**.



5. Right-click the new full-text catalog and choose **Properties**.

6. In the Full-Text Catalog Properties dialog, click the **Tables/Views** tab.

7. Assign the *CMS_Attachment* table to the catalog.
   a. Check the box next to the *AttachmentBinary* column
   b. Set the **Language for Word Breaker** to *English* or another value
   c. Set the **Data Type Column** to *AttachmentExtension*



8. Click **OK**.

You can now search through the content of basic files types stored in the database. See also:

- Searching PDF files
- Searching Microsoft Office documents

Enabling attachment search for the SQL search

Perform the following steps if you wish to search attachments using the SQL search:

Note

The SQL search engine is obsolete. It is recommended to use the Smart search instead. In that case you only need to enable the **Search in attachments** property of the relevant smart search results web parts.

1. Sign in to **Site Manager** and go to **Development -> Document types**.
2. Edit the **Root** document type.
3. Open the **Queries** tab.

4. Edit the **searchattachments** query and uncomment the following part of the code:

```
SELECT View_CMS_Tree_Joined.*, View_CMS_Tree_Joined.NodeName AS SearchResultName
FROM CMS_Attachment INNER JOIN View_CMS_Tree_Joined
ON View_CMS_Tree_Joined.DocumentID = CMS_Attachment.AttachmentDocumentID
WHERE (##WHERE##) AND
(([AttachmentName] Like N'%'+ @Expression + N'%') OR ([AttachmentTitle] Like N'%'+
 @Expression + N'%') OR ([AttachmentDescription] Like N'%'+ @Expression + N'%'))
OR (FREETEXT(AttachmentBinary, @expression))
ORDER BY ##ORDERBY##
```

3.5.14.3 Enabling full-text search on MSSQL Server (Script)

If you cannot use SQL Server Management Studio to configure the full-text search, run the following script against your Kentico CMS database:

```
-- Allows IFilter library loading
exec sp_fulltext_service 'verify_signature', 0
exec sp_fulltext_service 'load_os_resources', 1

-- Creates the Full Text Catalog
exec sp_fulltext_catalog 'KenticoCMSCatalog','create'

-- Adds the CMS_Attachment table to the catalog
exec sp_fulltext_table
'CMS_Attachment','create','KenticoCMSCatalog','PK_CMS_Attachment'

-- Sets the data column of the CMS_Attachment table in the catalog
exec sp_fulltext_column 'CMS_Attachment','AttachmentBinary','add',
NULL,'AttachmentExtension'

-- Populates the catalog
exec sp_fulltext_table 'CMS_Attachment','start_full'
```

You can now search through the content of basic files types stored in the database. See also:

- Searching PDF files
- Searching Microsoft Office documents

Enabling attachment search for the SQL search

Perform the following steps if you wish to search attachments using the SQL search:

Note

The SQL search engine is obsolete. It is recommended to use the Smart search instead. In that case you only need to enable the **Search in attachments** property of the relevant smart search results web parts.

1. Sign in to **Site Manager** and go to **Development -> Document types**.
2. Edit the **Root** document type.
3. Open the **Queries** tab.
4. Edit the **searchattachments** query and uncomment the following part of the code:

```
SELECT View_CMS_Tree_Joined.*, View_CMS_Tree_Joined.NodeName AS SearchResultName
FROM CMS_Attachment INNER JOIN View_CMS_Tree_Joined
ON View_CMS_Tree_Joined.DocumentID = CMS_Attachment.AttachmentDocumentID
WHERE (##WHERE##) AND
(([AttachmentName] Like N'%'+ @Expression + N'%') OR ([AttachmentTitle] Like N'%'+
 @Expression + N'%') OR ([AttachmentDescription] Like N'%'+ @Expression + N'%'))
OR (FREETEXT(AttachmentBinary, @expression))
ORDER BY ##ORDERBY##
```

3.5.14.4 Searching PDF files

Kentico CMS does not support searching in PDF files by default. To enable it, you need to install the free **Adobe PDF IFilter** on your machine. The installation process is different for 32-bit and 64-bit systems.

Note: Adobe PDF IFilter is not a product of Kentico Software and we cannot guarantee its functionality.

Configuration on 32-bit platforms

Use the following procedure to enable searching in PDF files on 32-bit platforms:

1. Download the free Adobe PDF IFilter 6.0 from the Adobe website: http://www.adobe.com/support/downloads/detail.jsp?ftpID=2611
2. Run the installer and follow the instructions.
3. Rebuild the full-text search catalog that you created for Kentico CMS.
4. If you are using MS SQL Server 2005, run the following SQL commands to ensure that the server can load the PDF IFilter libraries:

```
sp_fulltext_service 'verify_signature', 0
GO
sp_fulltext_service 'load_os_resources', 1
```

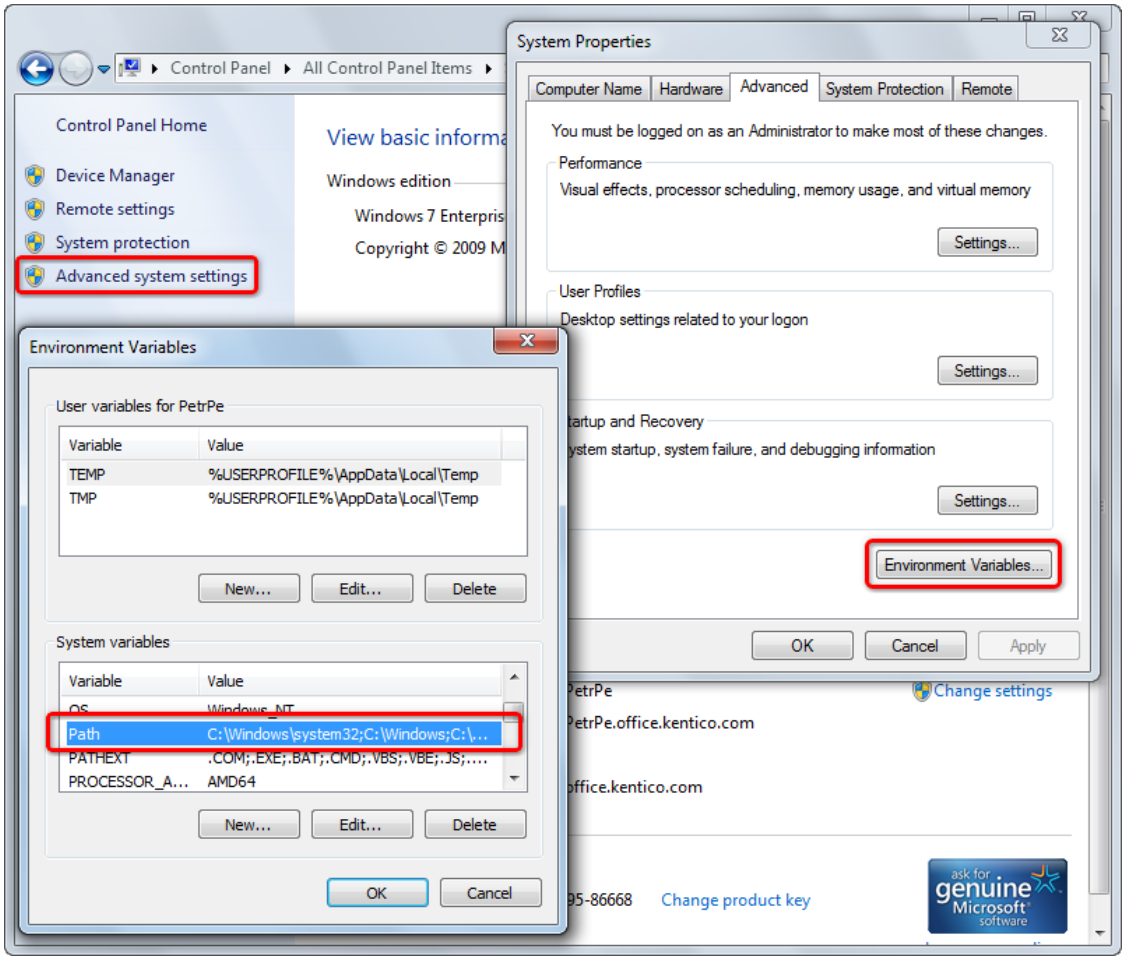
You can now search for PDF files stored in the database.

Configuration on 64-bit platforms

Use the following procedure to enable searching in PDF files on 64-bit platforms:

1. Download Adobe PDF IFilter 9 for 64-bit platforms from the following location: http://www.adobe.com/support/downloads/detail.jsp?ftpID=4025
2. Make sure that no other version of IFilter is installed on your machine.
3. Run the installer and follow the instructions.
4. After installing the PDF IFilter, it is recommended to add the **bin** folder of the IFilter installation directory to your system **Path** environment variable:

a. Go to **Control Panel -> System -> Advanced system settings -> Environment Variables**.
b. Append *C:\Program Files\Adobe\Adobe PDF IFilter 9 for 64-bit platforms\bin* to the value of the ***Path*** variable.



5. Restart the computer.
6. Check whether Adobe IFilter is registered with the server — run the following query against your server:

```
SELECT * from sys.fulltext_document_types
```

You should see a list of supported file types that have filters installed. Verify that you can see an entry for **.pdf**.

7. If you do not see an entry for **.pdf**, execute the following two lines:

```
EXEC sp_fulltext_service 'load_os_resources', 1
EXEC sp_fulltext_service 'verify_signature', 0
GO
```

8. Restart the SQL Server.
9. Run the query from step 6 again and verify that you can see the entry for **.pdf**.

You can now search for PDF files stored in the database.

3.5.14.5 Searching Microsoft Office documents

Kentico CMS does not support searching in Microsoft Office Open XML files by default (documents created in Office 2007 or newer — .docx, .xlsx, .pptx etc.). To enable search in these files, you need to install and configure **Microsoft Filter Pack** IFilters on your machine.

Note: Microsoft Filter Pack is not a product of Kentico Software and we cannot guarantee its functionality.

Perform the following steps to enable searching in Microsoft Office Open XML files:

1. Download the Microsoft Filter Pack installer that matches the architecture of your operating system from the Microsoft website:

- Microsoft Office 2007 Filter Pack
- Microsoft Office 2010 Filter Pack

2. Run the installer and follow the instructions.

3. After you finish the installation, start SQL Server Management Studio, access your Kentico CMS database and run the following command in that instance:

```
sp_fulltext_service 'load_os_resources', 1
```

This allows the server to load the IFilter libraries.

4. Check whether the IFilters are registered with the server. Type and execute the following in a new query window:

```
SELECT * from sys.fulltext_document_types
```

You should see a list of supported file types that have filters installed. Verify that the list contains entries for all desired file extensions.

5. Restart the SQL Server service.

6. **Rebuild** the full-text search catalog that you created for Kentico CMS.

You can now search for Microsoft Office Open XML files stored in the database.

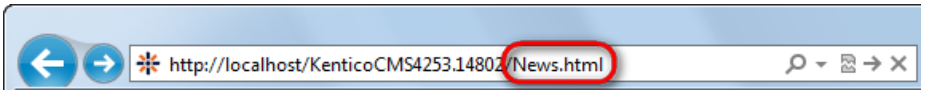
3.5.15 Custom URL extensions and extensionless URLs

3.5.15.1 Overview

This chapter describes how to configure the system for using custom URL extensions and extension-less URLs. Custom URL extensions let you have the extensions rendered with a different extension than the default *.aspx*, like *.html*, *.php*, *.xxx* or any other.

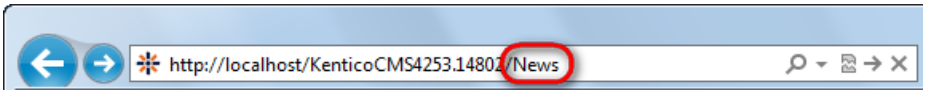
So for example, instead of the default *http://www.example.com/news**.aspx***, you can have URLs ending with **.html**:

*http://www.example.com/news**.html***



you can also configure an empty extension, which results in an **extension-less URL**:

http://www.example.com/news



To achieve this result you have to take the following two steps:

1. Adjust your web.config or IIS to handle the custom extensions. This step is different for IIS 7 or higher and IIS 6.
2. Configure custom URL extensions in Kentico CMS. It is described in this chapter.

3.5.15.2 IIS 7 and higher

If your instance of Kentico CMS is running on a server with IIS 7 or higher installed, you only need to add or modify several keys in your project's **web.config** file.

Note: This procedure only works for IIS 7 sites that use an **Application Pool** with **Managed Pipeline Mode** set to *Integrated*.

Required configuration

1. Edit your application's **web.config** file.

2. Find the **<modules>** element inside the **system.webServer** section directly under the web.config root (i.e. not under a specific <*location*> element).

3. Modify the the opening <*modules*> tag to match the following code:

```
<system.webServer>
    ...
<modules runAllManagedModulesForAllRequests="true">
```

```
    ...
</system.webServer>
```

4. With this modification in your web.config, proceed to configuring the extensions in the administration interface of Kentico CMS.

You can also perform additional configuration as described below.

Optional configuration (recommended)

5. It is recommended to configure a custom *Page-not-found* (404) error page when using custom URL extensions, otherwise a blank page may be displayed in some browsers if a user attempts to access a non-existing resource. This can be achieved by entering the URL of the appropriate document or aspx page as the value of the **Page not found URL** setting in **Site Manager -> Settings -> Content**, for example: *~/CMSMessages/PageNotFound.aspx*. Please read the Configuring custom error pages topic to learn more.

Alternatively, you can add a **<httpErrors>** element into the *system.webServer* section of your **web.config** file according to the following:

```
<httpErrors existingResponse="Auto" errorMode="Custom">
  <clear/>
  <error statusCode="404" responseMode="ExecuteURL" path="/<Virtual_Directory>/
CMSMessages/PageNotFound.aspx" />
</httpErrors>
```

Replace the <*Virtual_Directory*> string in the value of the **path** attribute of the **<error>** element with the name of the virtual directory where you run Kentico CMS.

6. You can also add the following key to the *AppSettings* section, which ensures that URLs remain the same even after postback.

```
<add key="CMSUseExtensionOnPostback" value="false" />
```

7. If you are using trailing slashes (enabled by the **Use URLs with trailing slash** option in **Site Manager -> Settings -> URLs and SEO**), you can use the following extra key to have only extensionless URLs ending with the trailing slash. URLs ending with an extension are rendered without the slash when the key is used.

```
<add key="CMSUseTrailingSlashOnlyForExtensionLess" value="true" />
```

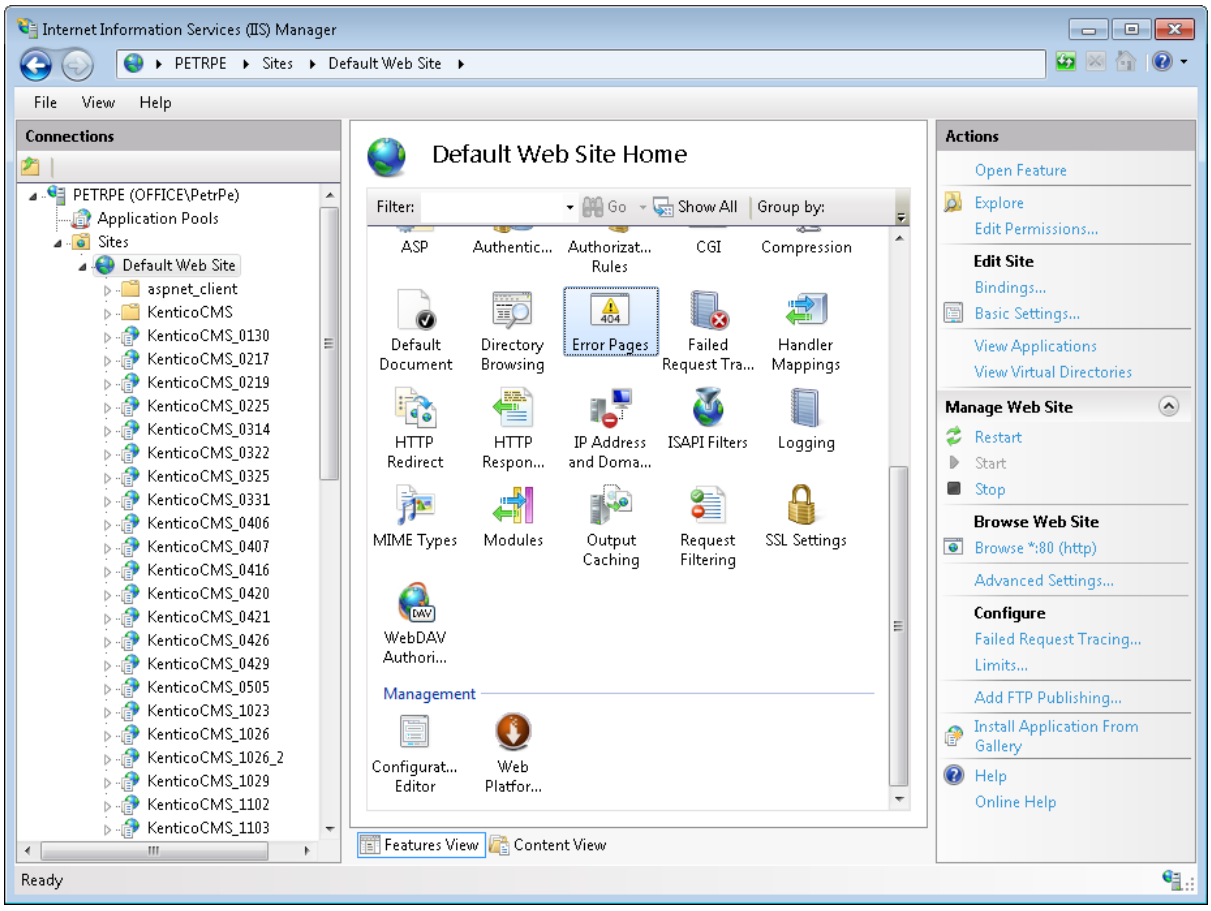
Using the cmspages/handler404.aspx special page (obsolete)

Due to backward compatibility, you can also enable custom URLs by configuring your IIS manually. It is not recommended now as it is obsolete. If you receive the **Lock violation error** during the procedure, try the solution described here.

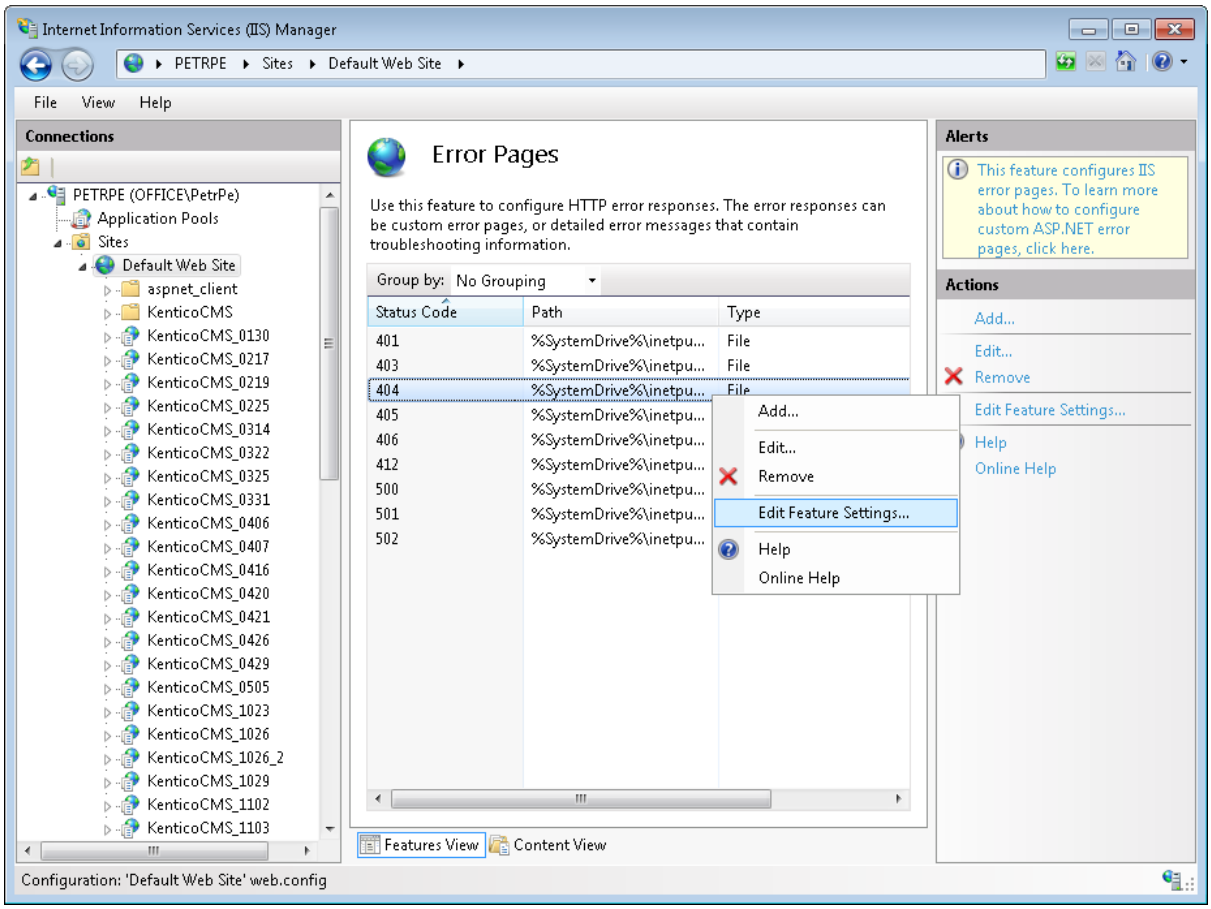
The setup procedure on IIS 7 is the following:

1. Open **Start -> Control Panel -> Administrative Tools -> Internet Information Services (IIS) Manager**.

2. Select your website from the tree on the left and open the **Error pages** section.



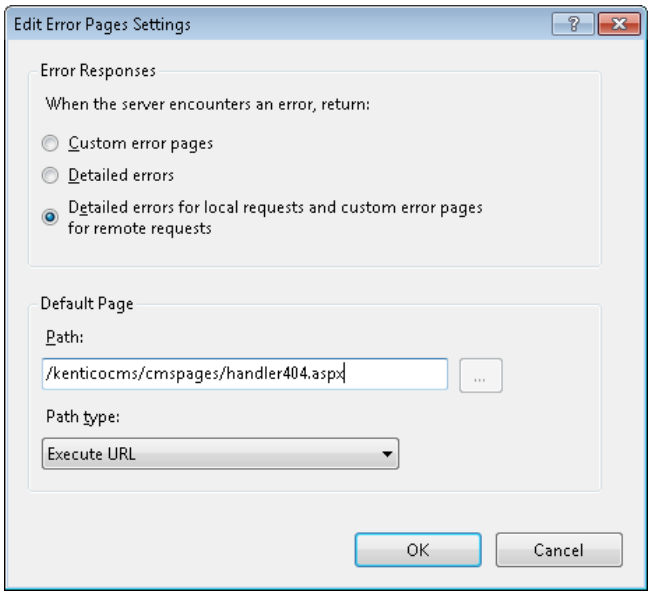
3. Right-click the **404 error** and choose **Edit Feature Settings**.



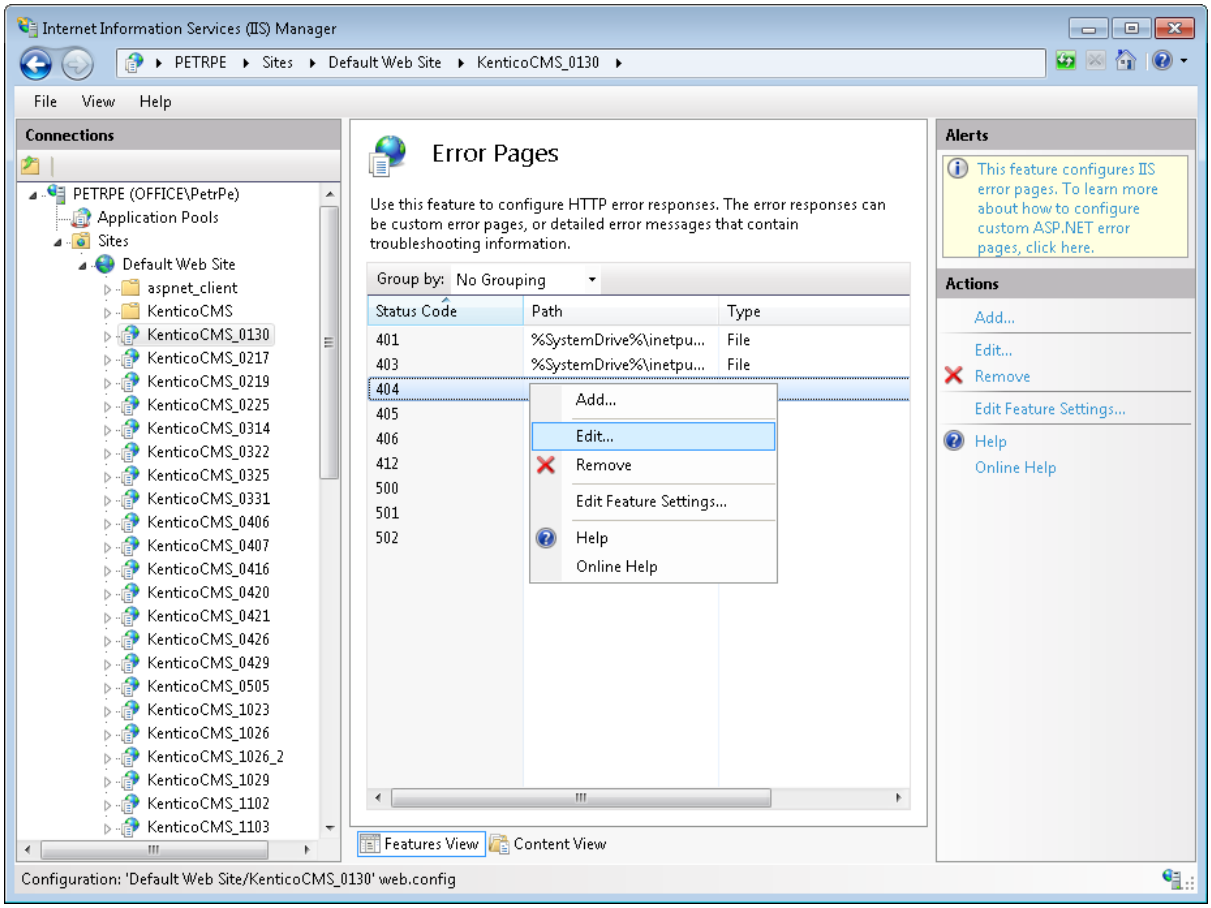
4. Enter the following values:

- **Path:** enter the URL of the **cmspages/handler404.aspx** page according to your application's URL.
  Example: if you run your web project in virtual directory `/kenticocms`, you need to enter `/kenticocms/cmspages/handler404.aspx`
- **Path type:** Execute URL

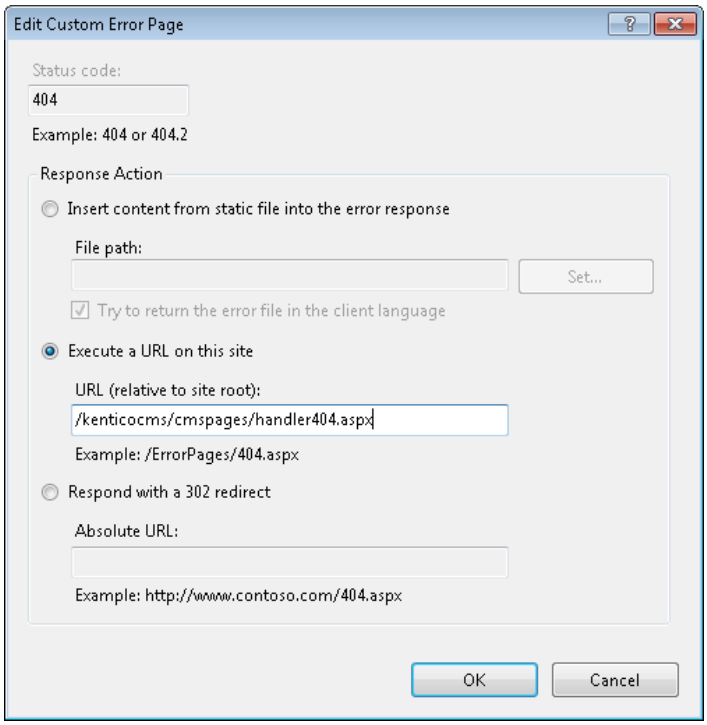
Click **OK**.



5. Now right-click the **404 error** again and choose **Edit** from the context menu.



6. Select **Execute a URL on this site** and enter the same URL that you entered in step 4. Click **OK**.



7. Go back to step 3 and repeat the same procedure for the **405 error**.

8. Click **OK** on all dialogs to save the changes. It's not necessary to restart the application.

Setting up the .html extension on IIS manually (obsolete)

The following approach is also possible, but not recommended and considered obsolete now. To use the **.html** extension, go through the following steps:

1. Run **IIS Manager** .

2. Select your web.

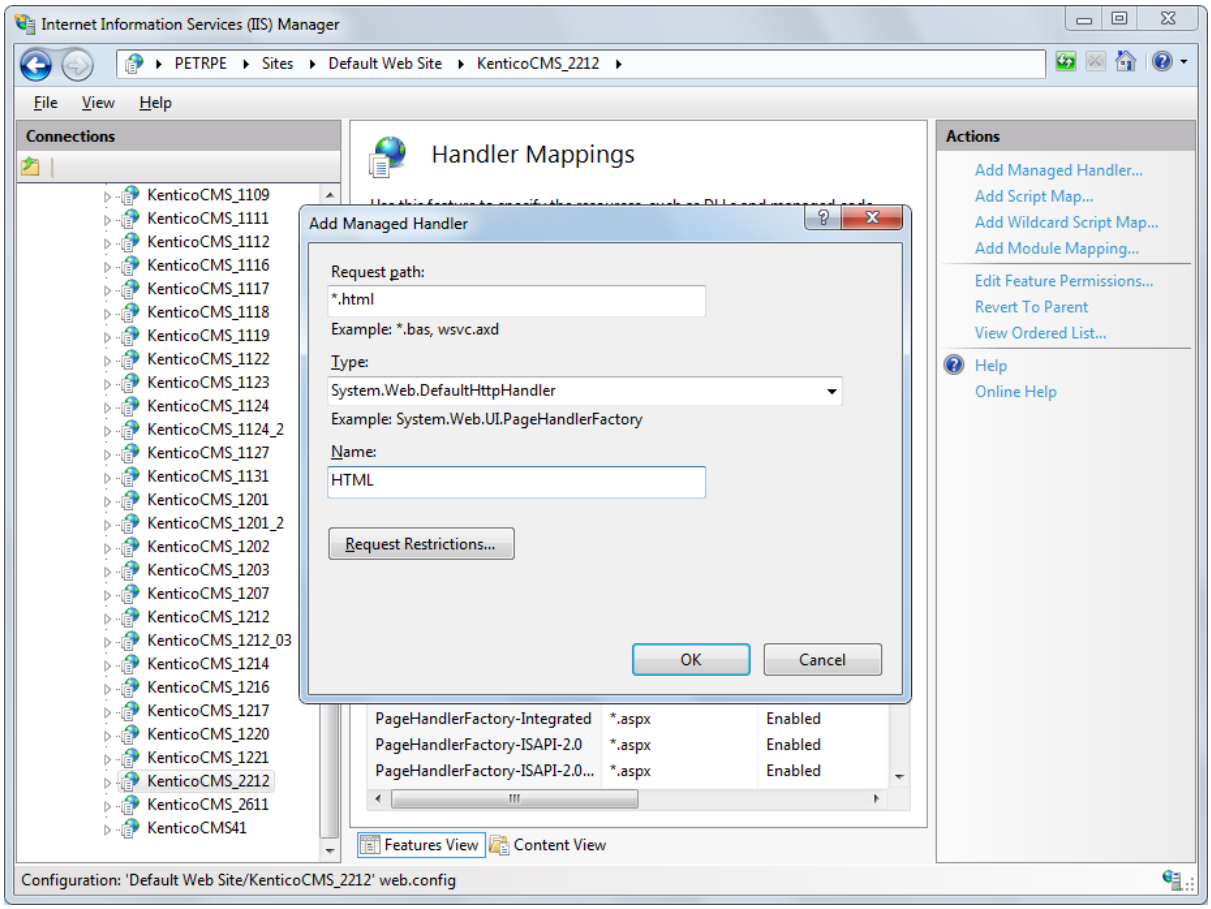
3. Open **Handler mappings**.

4. Click to **Add managed handler...** .

5. Enter the following values:

- **Request path**: *.html
- **Type**: System.Web.DefaultHttpHandler
- **Name**: HTML

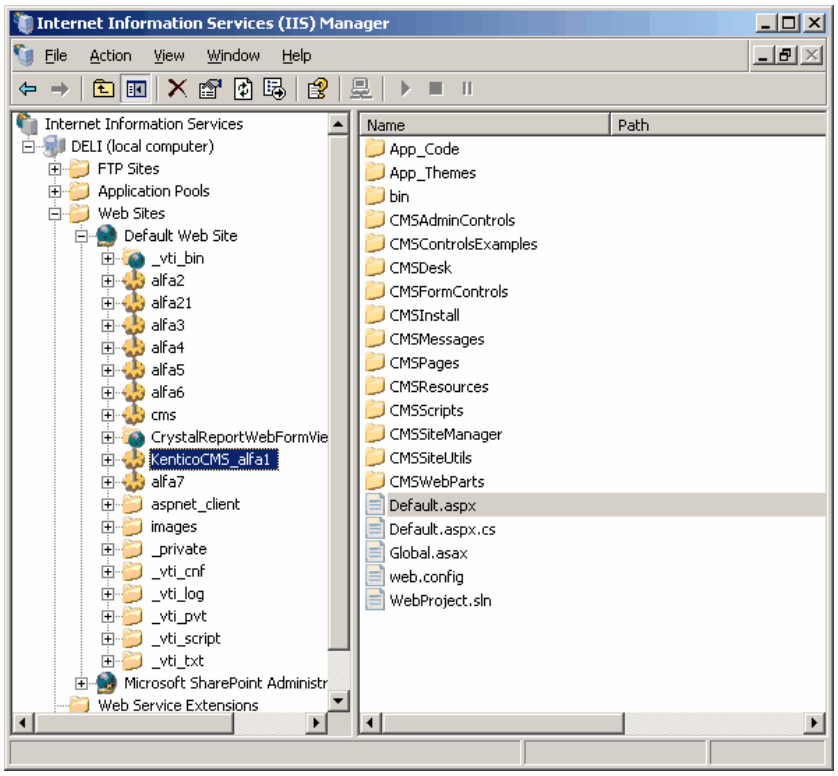
Click **OK**.



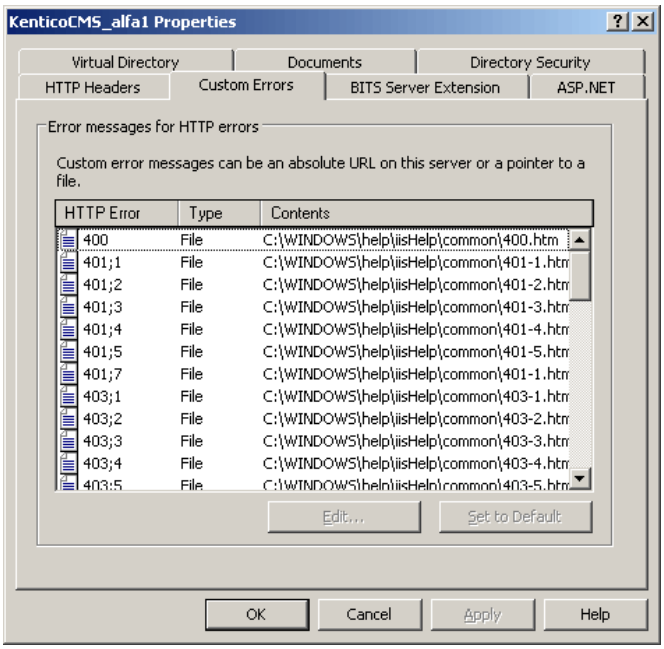
3.5.15.3 IIS 6

This procedure needs to be performed to configure IIS 6 to handle Kentico CMS's custom URL extensions. You can use this configuration on **Windows XP** and **Windows Server 2003** with Internet **Information Services (IIS) 6** installed. It's not possible to use it with Visual Studio's built-in web server.

1. Go to **Start -> Control Panel -> Administrative Tools** and launch the **Internet Information Services (IIS) Manager**. Locate the appropriate website and virtual directory (if you installed Kentico CMS into the root, you will make this change on the website level only).



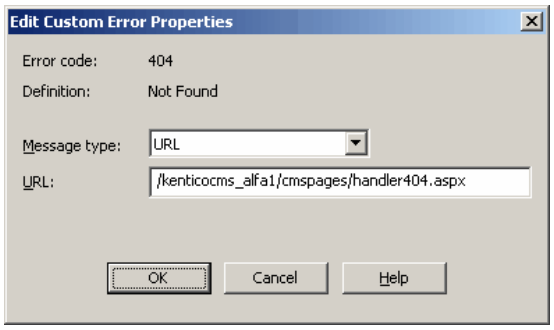
2. Right-click the directory (or website) and choose **Properties** and then click the **Custom Errors** tab:



3. Edit the 404 error and enter the following values:

- **Message type:** URL
- **URL:** enter the URL of the cmspages/handler404.aspx page according to your application's URL. E.g. if you run your web project in virtual directory `/kenticocms`, you need to enter `/kenticocms/`

`cmspages/handler404.aspx`



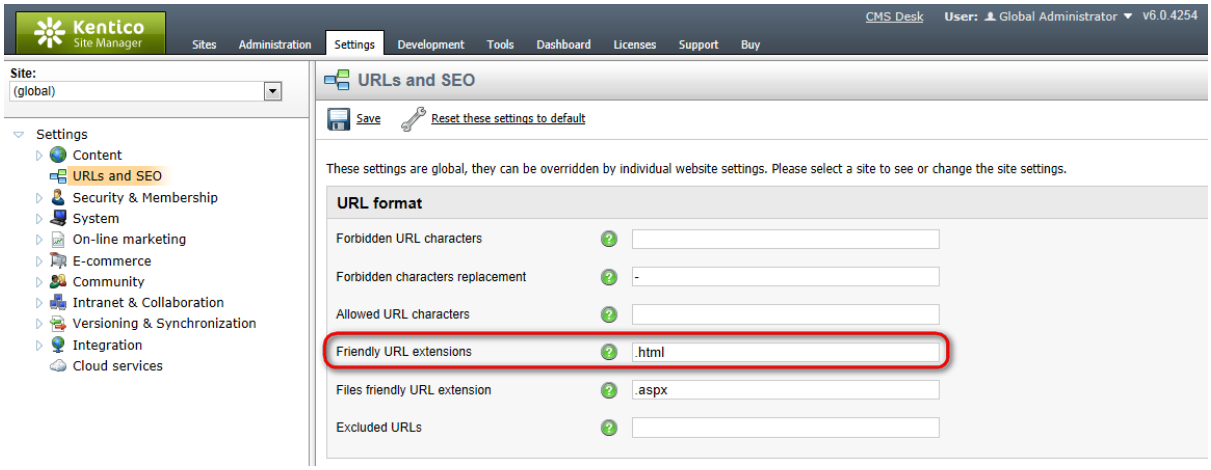
4. Now repeat the same for the 405 error, using the same custom URL: `/kenticocms/cmspages/handler404.aspx`

Click **OK** on all dialogs to save the changes. It's not necessary to restart the application.

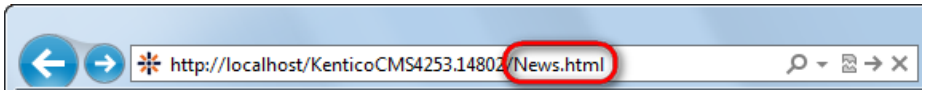
3.5.15.4 Configuration of custom URL extensions (.html or other)

When you have performed the required configuration for your version of IIS, you can proceed with entering the actual extensions.

This can be done in **Site Manager -> Settings -> URLs and SEO**. For example, try to set the **Friendly URL extensions** value to *.html* and click **Save**.



Now when you go to the live website, you will see that all URLs in menus and listings are rendered with **.html** extensions. In some cases, you may need to update some static links which were created with the default **.aspx** extension.



Using multiple extensions

You can enter multiple extensions into the **Friendly URL Extension** field mentioned above. The

following format should be used:

.html;.htm;;.xxx;.abc

- The first extension is used as the default one and the links will be generated with it in the browser.
- Other extensions are entered divided by semicolons (;). Pages can be accessed through URLs ending with all entered extensions.
- If you enter a semicolon without any extension in front of it, like in the middle of the sample entry above, extension-less URLs can be achieved.

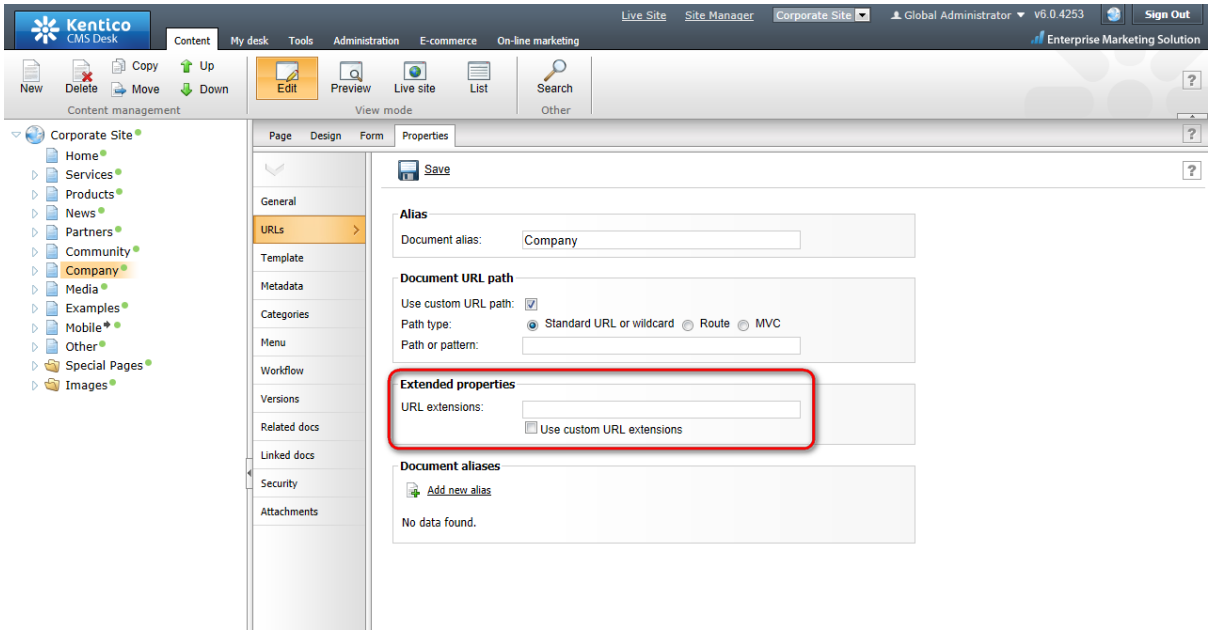
The **Redirect documents to main extension** setting, which can also be found at **Site Manager -> Settings -> URLs and SEO**, causes URLs with non-default extensions to be redirected to corresponding URLs with the current default extension. This can be useful for SEO purposes. For example, if you want to change the extensions of your documents and want them to be used when the documents are accessed from a search engine that has your website indexed with the old extension.

Document-level extensions settings

Apart from the global settings described above, document extensions under which the document can be accessed can also be set separately for each document. The default extension with which the pages are **rendered** in the browser is always taken from the **global** settings.

1. Select the document from the content tree.
2. Switch to its **Properties -> URLs** tab.
3. Enable the **Use custom URL extensions** check-box.
4. Enter the required extension(s) using the same rules as described above.

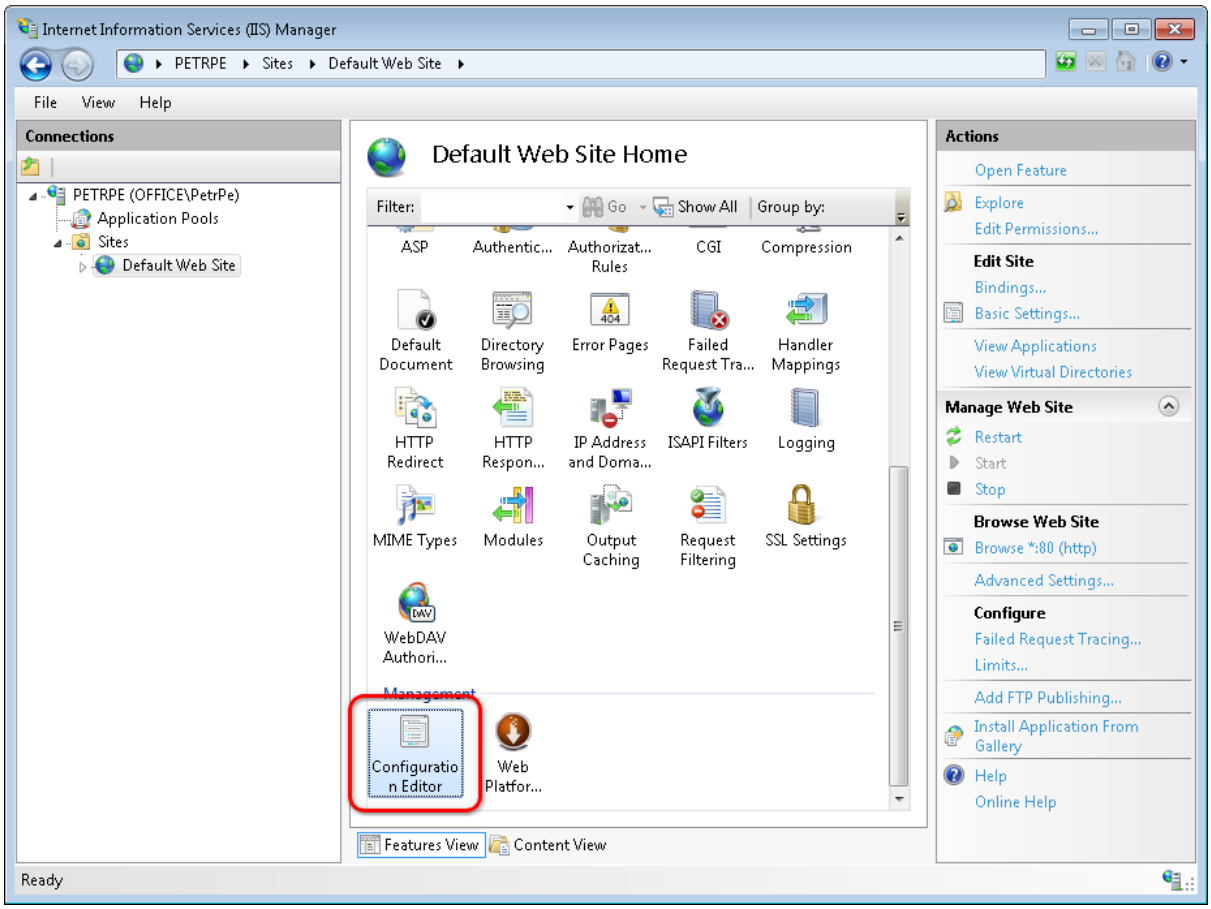
Please note: Even if the **Use custom URL extensions** option is disabled, files (**cms.file** documents) can be accessed under their physical extensions.



3.5.15.5 Lock violation on IIS7

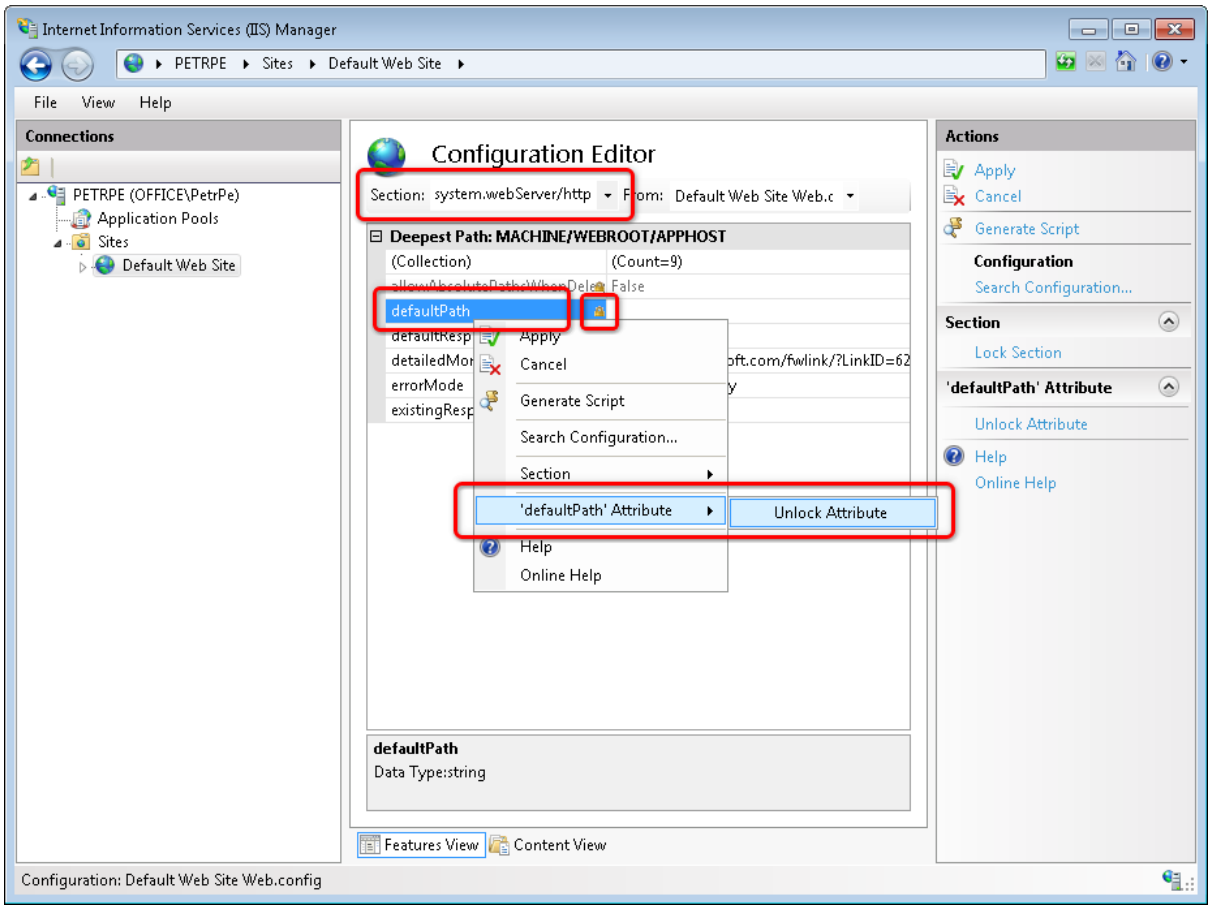
When configuring your IIS 7 (or later) to allow custom extensions or extension-less URLs, especially if you're running Kentico CMS in a virtual directory, you may receive the **Lock violation** error message. This typically doesn't allow you to specify the path settings in step 4 of the instructions for using cmspages/handler404.aspx described on this page.

The reason is most probably a locked **defaultPath** attribute in the **httpErrors** section. You can check and unlock it in **IIS Manager**. Select your site (IIS site) and open **Configuration Editor** (it is included in the *Management* section in the standard installation of IIS 7.5; if you are using IIS 7, you can download and install it as a part of the IIS7 Administration Pack).



In Configuration Editor, choose **webServer/httpErrors** from the **Section** drop-down list.

If there is a **lock icon** next to the **defaultPath** attribute name, right-click on the attribute name and select the **'defaultPath' Attribute -> Unlock Attribute** action from the context menu. If the option is missing in the context menu, you will probably have to unlock it on a higher level in the IIS tree, i.e. on the parent site or on the root of the server.

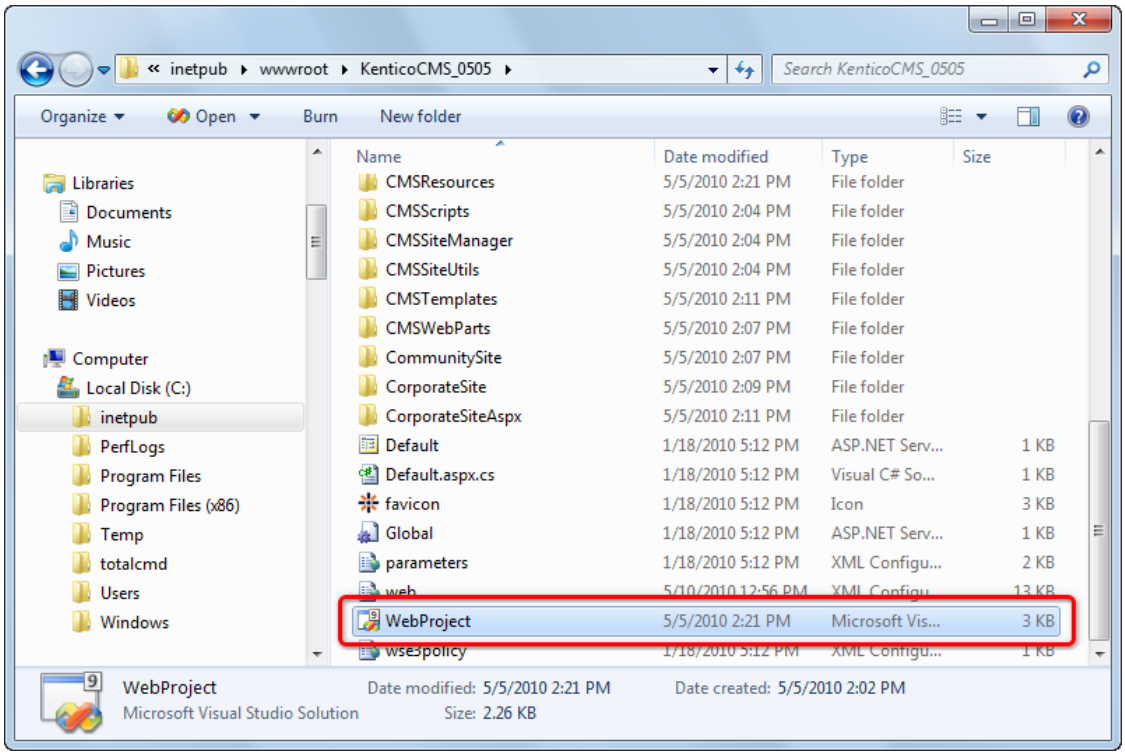


Click **Apply** to save the changes and from now, the *Lock violation* errors shouldn't appear.

3.6 Visual Studio integration

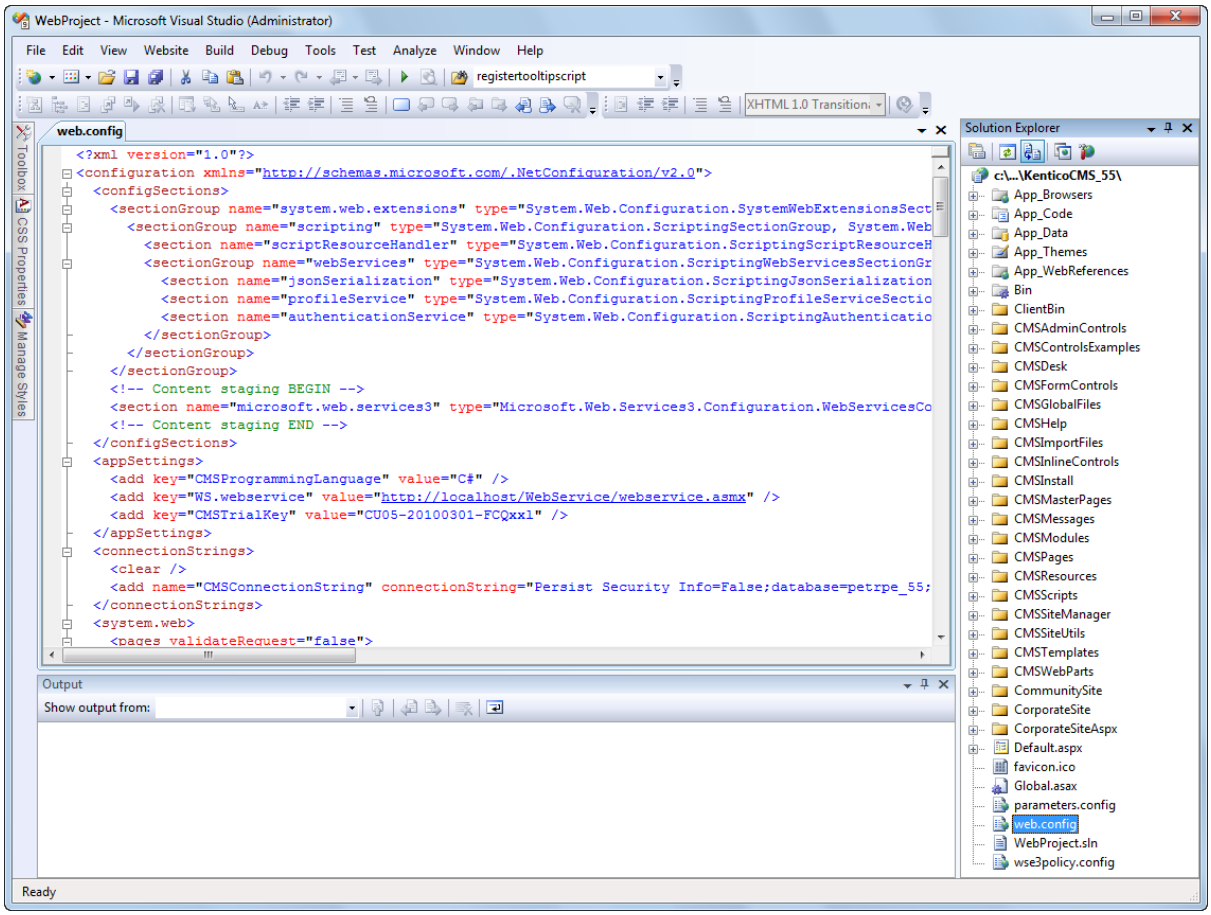
3.6.1 Opening the project

After you create a new Kentico CMS web project, you can open it in Visual Studio by double-clicking the WebProject.sln file in Windows Explorer:



If this option doesn't work, you can start **Visual Studio** and choose **File -> Open -> website** in the main menu and navigate to the folder which contains the *WebProject.sln* file.

The project looks like this:



Making modification to the standard code

Although Kentico CMS is delivered with source code of the administration interface (CMS Desk, CMS Site Manager), it's recommended that you do NOT modify the default files to avoid problems when upgrading to a higher version (your changes may be overwritten by new code).

If you need to modify some dialog, note down its name and merge your modifications with the new version during the upgrade to a higher version of Kentico CMS.

If you need to modify some web part, create its copy and then modify it.

Source Code Options

There are two levels of source code:

- the **source code of the website**, administration interface and web parts - this source code is delivered with every license (even in the trial version)
- the **full source code** of all libraries, including data layer, business layer and Kentico CMS Controls - this source code is only available as a part of the 1 website Ultimate or 1 Server Ultimate licence with Source Code.

You're allowed to modify the source code you receive (in both options) and deploy the modified version into production environment, provided you meet all other licensing conditions.

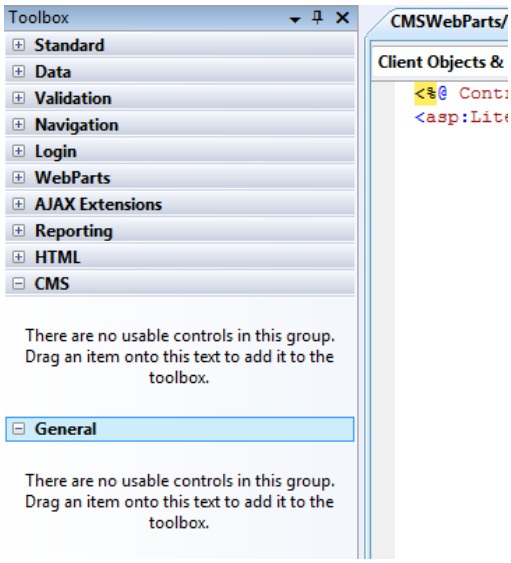
3.6.2 Adding Kentico CMS Controls to the Toolbox

Before you start using Kentico CMS Controls in your ASP.NET project, it is recommended to add the controls to the **Toolbox**:

1. Open the web project in Visual Studio and edit any ASPX web form or ASCX user control file, for example *Default.aspx* under the project root. This is necessary, because the controls will only be offered in the toolbox when working with ASPX markup.

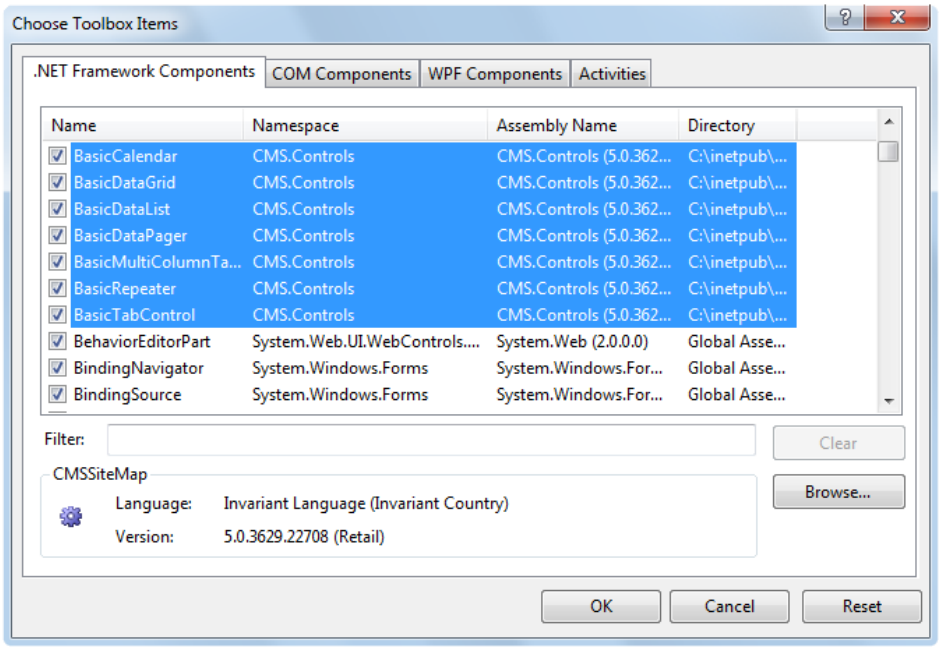
2. Right-click the **Toolbox** and choose **Add tab** from the context menu.

3. Type the name of the new tab (e.g. *CMS*) and press Enter:

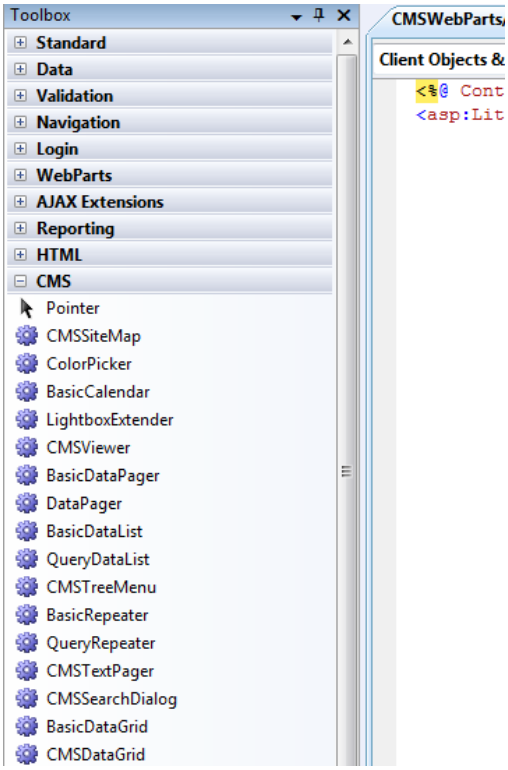


4. Right-click the new tab and choose **Choose items...** from the context menu.

5. In the **Choose Toolbox Items** dialog, click **Browse** and locate the **CMS.Controls.DLL** library in the **bin** folder under your website. Click **Open** and then click **OK**.

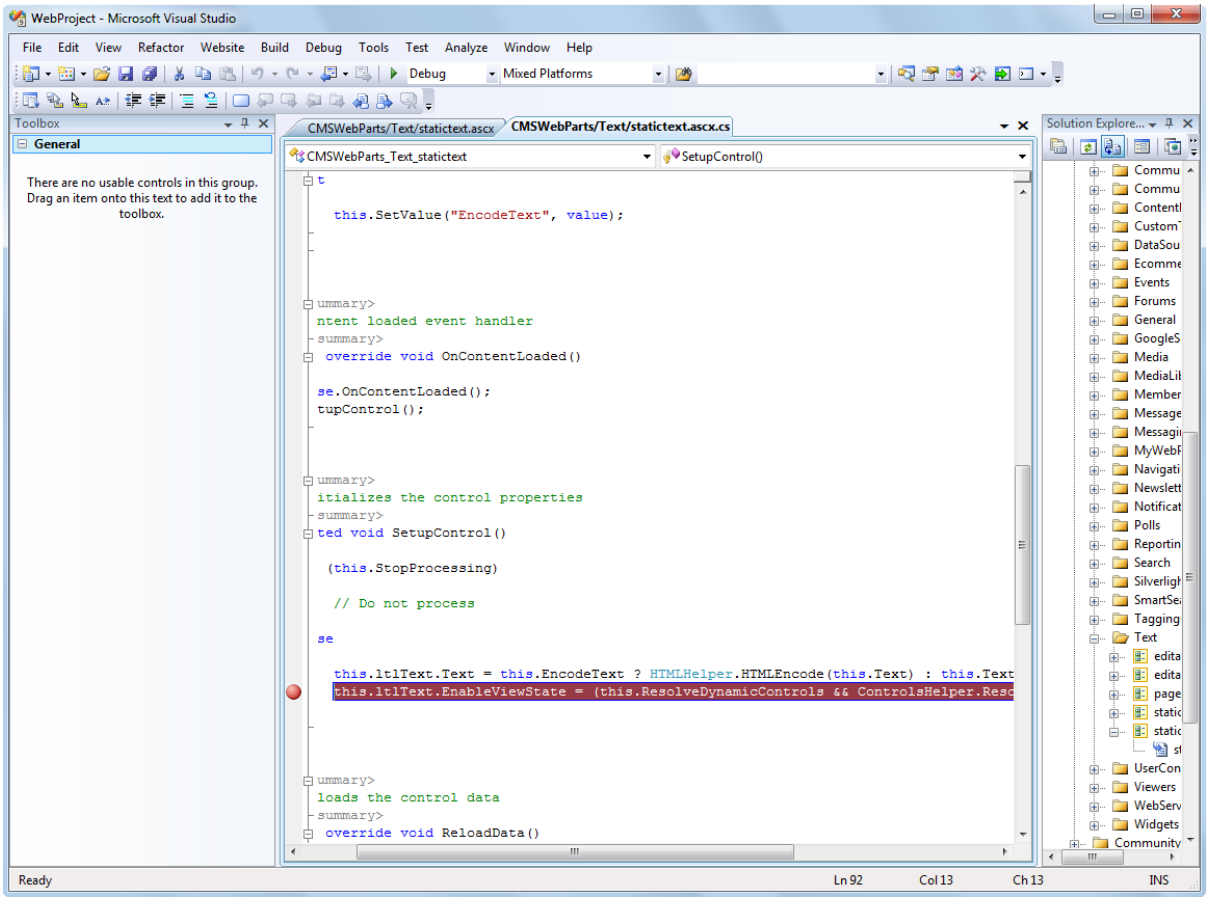


6. The controls are now added to the Toolbox, as you can see in the following screenshot. Now you can easily drag and drop the controls into the content of your web forms or user controls.

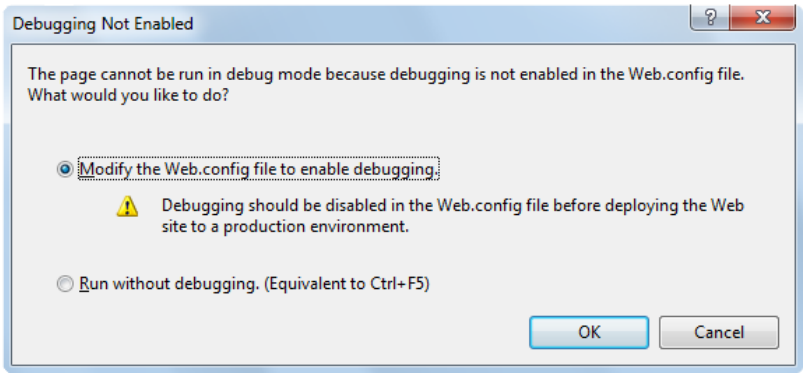


3.6.3 Debugging

If you're adding custom code using Visual Studio into Kentico CMS, you can easily debug it in Visual Studio as you're used to since Kentico CMS is a standard Visual Studio application. Simply click on the left next to your method or command and it will create a red breakpoint:



Now choose **Debug -> Start debugging** in the main menu or press **F5**. The website starts and you can track the code flow. You may get a message like this:



You will need to choose to **Modify the web.config file to enable debugging** and click **OK**. It's recommended that you disable debugging before deploying the website to a production environment by

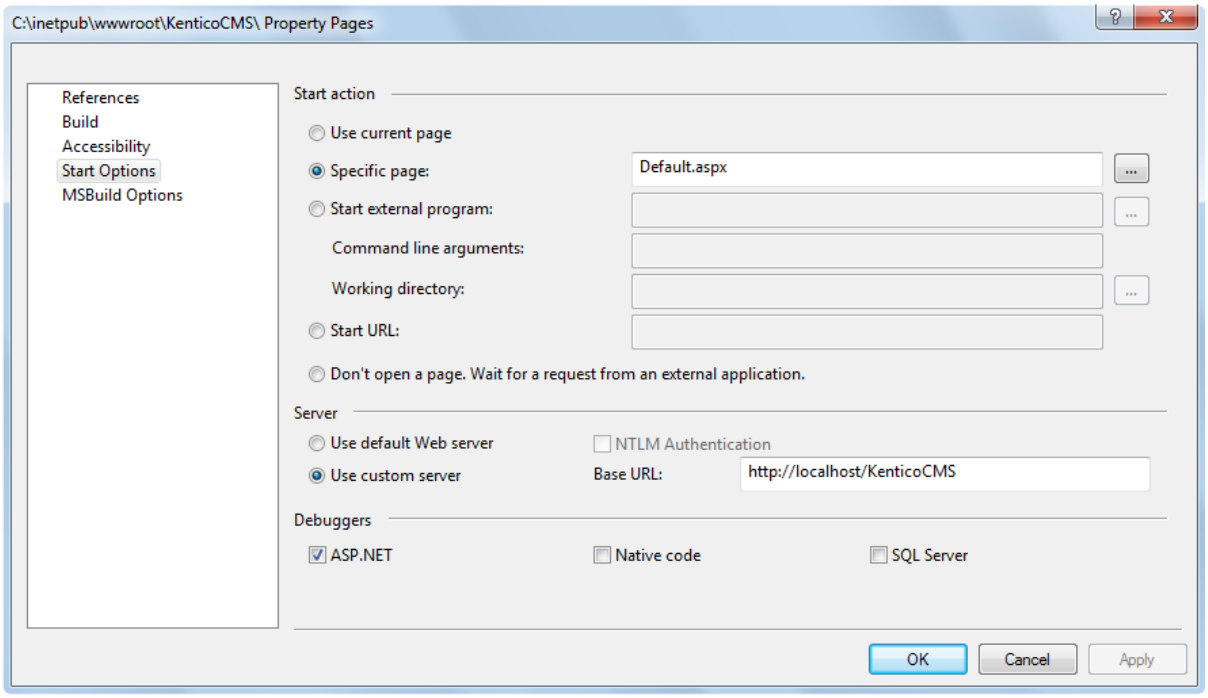
setting the value

```
<compilation debug="true" .... />
```

to **false** in your *web.config* file for better performance.

Debugging using IIS website

The debug mode starts in built-in web server by default. You can change this in the **website -> Start** options dialog by choosing the Use custom server option like this:



Please note that you need to be local administration and the website or virtual directory must be configured for both anonymous and Windows authentication (see Additional configuration tasks -> Creating a virtual directory for details).

Debugging from within Kentico CMS UI

Kentico CMS provides certain debugging possibilities directly from within its user interface in **Site Manager -> Administration -> System -> Debug**. Further information on this topic can be found in the Development -> Debugging chapter of this guide.

3.6.4 Precompilation (Publish function)

When deploying a website, you have the option of precompiling the web project before you place it on a server. This compiles the source code into assemblies, which provides several advantages:

- Faster initial response times - resources do not need to be compiled dynamically when they are requested for the first time.

- Source code protection - you can create a compiled version of the website without any accessible source code.
- Efficient deployment - using the **Publish** function, you can compile and deploy the website at the same time.

Limitations of precompiled websites

Only use precompilation to deploy completed websites. Precompiled instances of Kentico CMS have limited website development features.

You cannot edit, create or import virtual objects that contain ASP.NET markup and require compilation:

- Page layouts (ASCX type)
- Transformations (ASCX type)
- Web part layouts

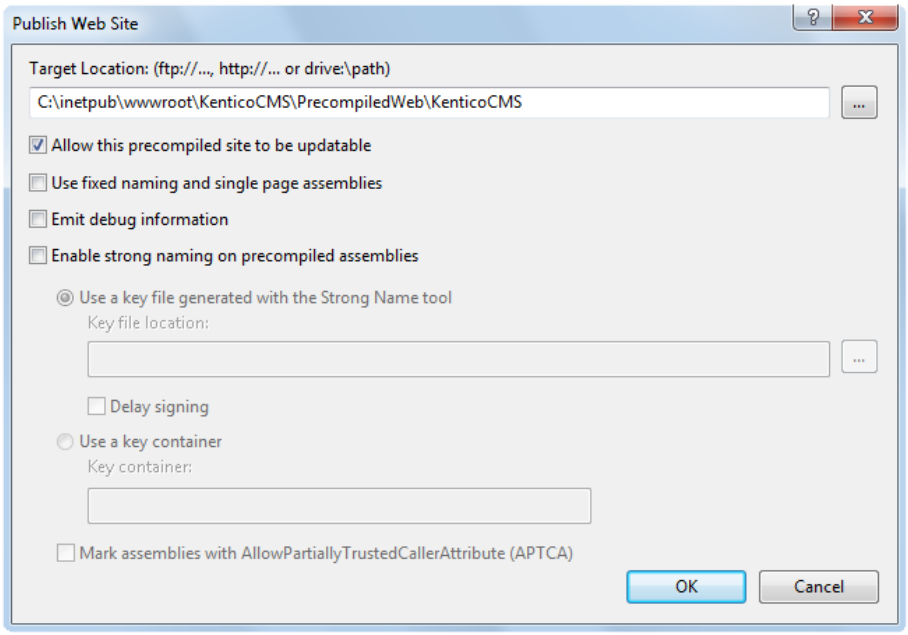
If you wish to import physical files that require compilation (such as *cs*, *vb*, *aspx*, *ascx*), you need to run the import process on the original project, and then create the precompiled site again.

Pre-compiling a web project

1. Rename the **Microsoft.Web.Services3.dll.rename** file in the ~*Bin* directory to **Microsoft.Web.Services3.dll**.
   - This allows you to use content staging on the precompiled site.

2. Go to **Site Manager -> Administration -> System -> Virtual objects** and click **Store all virtual objects in file system**.
   - This saves all virtual objects into the web project as physical files, so they can be included in the precompilation.
   - The system cannot load virtual objects from the database in a precompiled environment (the Virtual path provider is not available).

3. Right-click your web site project in the Solution Explorer in Visual Studio and select **Publish Web Site**.
   - This function is not present in the Express versions of Visual Studio, but you can use the command line ASP.NET Compilation Tool.
   - The following dialog box opens:



4. Enter a path outside of your current web project where the precompiled version will be placed.

5. Choose between two compilation modes using the **Allow this precompiled site to be updatable** checkbox:

- checked - compiles the code behind of all web forms and controls, but stores the markup files (.aspx, .ascx) in their original code form.
- unchecked - the compilation also includes markup files. The website contains only empty "stub" .aspx/.ascx files without any code.

We generally recommend using non-updatable sites for optimal performance.

6. Click **OK**.

Once the process is complete, you can deploy the precompiled site to your server.

Moving the database

You may need to copy the database to the server using the standard **backup/restore** procedure, since the compiled website cannot be used to install an SQL Server database. Other options are:

- Install a non-compiled website on the server first, go through the setup wizard and then replace the non-compiled files with compiled ones, while keeping the web.config file as is.
- Install a Kentico CMS website locally and run the database setup against the remote SQL Server on the live server.

Testing the website before compilation

To check if your website runs without using the **Virtual path provider** (to simulate a precompiled environment), you can disable the provider by adding the following key into the *appSettings* section of your web.config file:

```
<add key="CMSUseVirtualPathProvider" value="false" />
```

If your application runs correctly with the virtual path provider disabled, you should be able to run it after the precompilation.

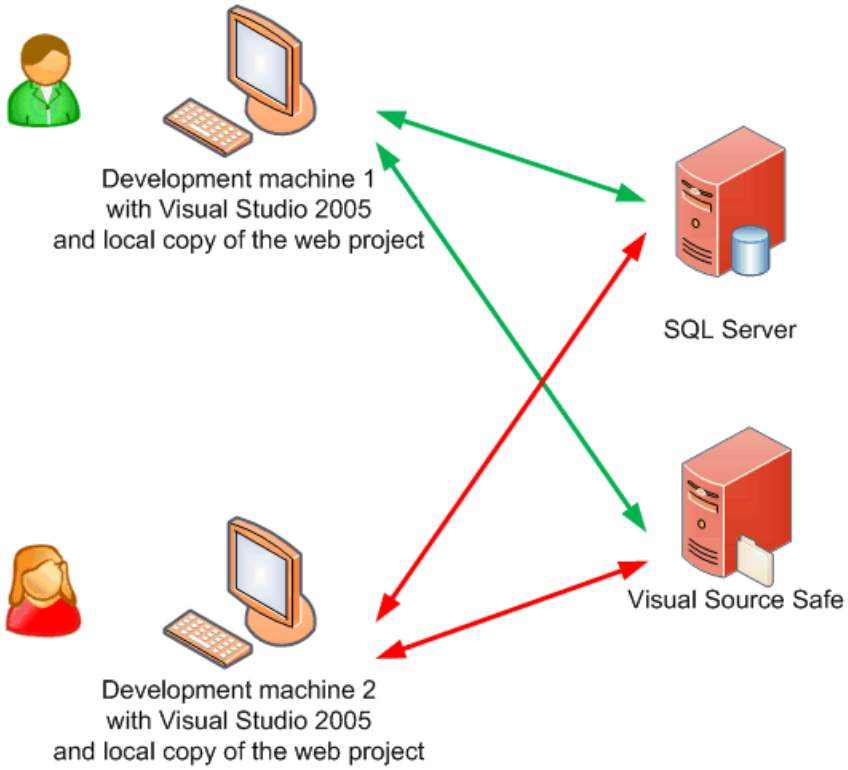
Related topics:

- Additional configuration tasks -> Configuration for Medium Trust environment
- Installation procedure -> Deployment to the live server
- Additional configuration tasks -> Installation on shared hosting server

3.6.5 Visual Source Safe and Team Development

Kentico CMS can be used in team development environment as any other website project in Visual Studio. You can also use Microsoft Visual Source Safe (VSS) as you're used to: you simply add the website to VSS and then you need to check out/check in the files you want to modify.

In this case, all developers have their local copy of Kentico CMS installed, but they use the same VSS code and the same database as shown on the following picture:



Synchronization of memory objects between development machines

Kentico CMS caches some system objects (such as transformations, templates, etc.) in memory. It means that the memory on multiple development machines may not be synchronized and the developers may not see the latest version and they may even overwrite the work of other developers.

We recommend you to synchronize the memory objects between development machines using the Web farm synchronization module.

Team Development without Visual Studio

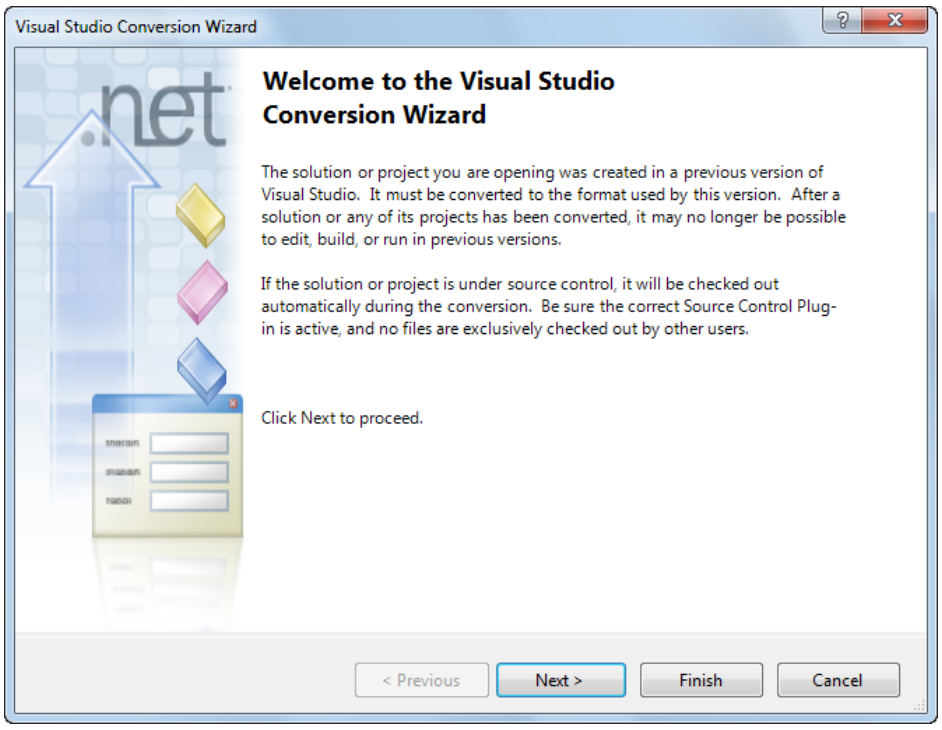
If the developers do not modify the source code and use the portal engine development model, they do not even need to have the local copies of the web project and they do not need to use VSS. In this case, they can install a single instance of Kentico CMS on their web server and develop the website through the browser-based interface.

3.6.6 Opening a VS2005 project in VS2008

If you were using Visual Studio 2005 for your web project and wish to convert to Visual Studio 2008, all you need to do is:

1. Start **Visual Studio 2008**.
2. Click **File -> Open website...** .
3. Choose the folder with your web project on the disk and click **OK**.

4. If you're asked if you wish to upgrade the project to a newer version of .NET Framework and Visual Studio, click **Next** and go through the wizard.



Compilation error

You may receive a compilation error saying there are different versions of the **System.Web.Extensions.dll** library in the Global Assembly Cache (GAC) and a temporary folder. In this case, you need to locate the file **bin/System.Web.Extensions.dll** in your web project and delete it.

3.7 Troubleshooting installation issues

3.7.1 Overview

You may encounter various issues during the installation. The following chapters may help you sort them out.

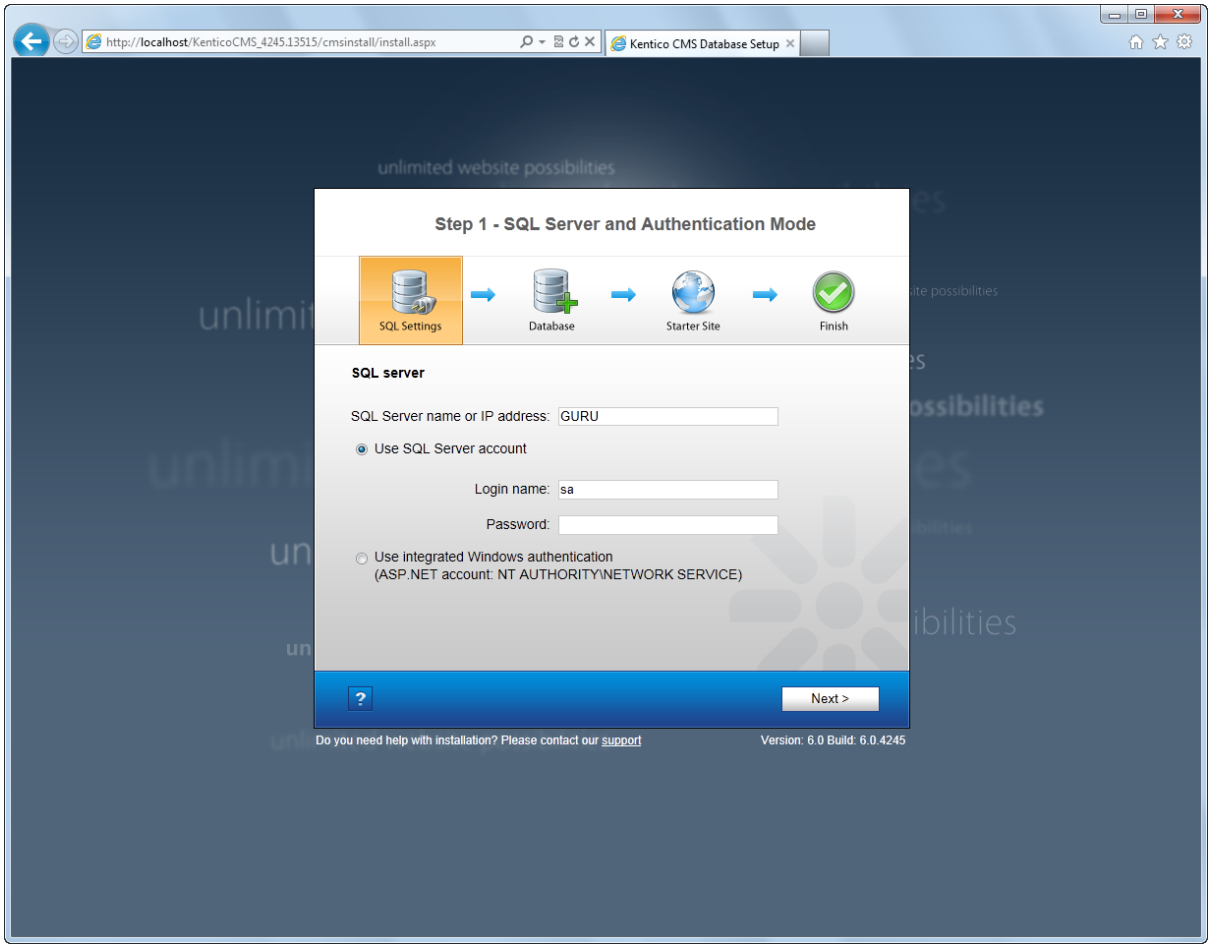
You may encounter problems in following areas:

- SQL Server connection problems
- ASP.NET not working on Windows Server 2003
- Disk permissions problems

Click one of the links to learn how to solve the issues.

3.7.2 SQL Server connection problems

You may encounter problems when entering the database connection details in the first step of database setup:



Error 1: Establishing connection to the server

Error message:

> An error has occurred while establishing a connection to the server. When connecting to SQL Server 2005, this failure may be caused by the fact that under the default settings SQL Server does not allow remote connections. (provider: Named Pipes Provider, error: 40 - Could not open a connection to SQL Server)

Troubleshooting:

1. Make sure the SQL Server name or IP address is correct. In some cases, using one of the following values may help:

- your computer name
- localhost

- 127.0.0.1
- (local)

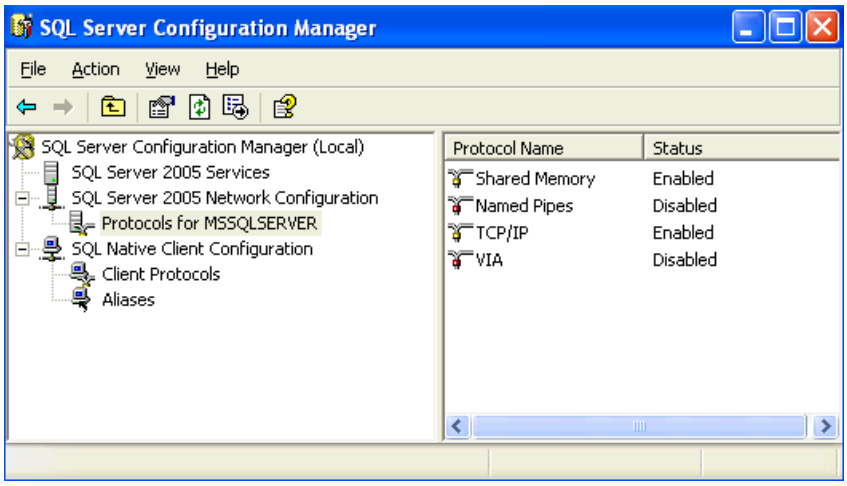
2. Make sure the server has Microsoft SQL Server 2005 or 2008 installed and running.

3. Make sure you are using the appropriate instance of the SQL Server in case you are using different instances of SQL Server. The instance name must be entered as myserver\myinstance (please note there's a backslash ****).

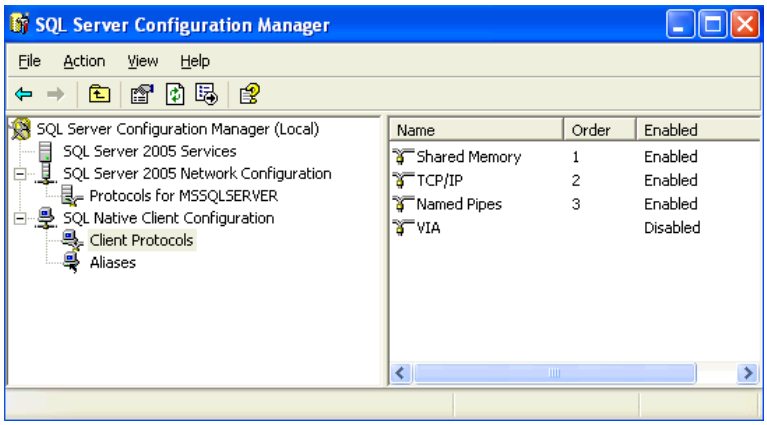
4. If you're using Microsoft SQL Server Express 2005 with default installation settings, the correct server name is **.\sqlexpress** or **computername\sqlexpress** .

5. Make sure the access to the database server is not blocked by some firewall (the default port number for TCP/IP protocol is 1433).

6. If you're using **SQL Server 2005** (especially the Express Edition), some protocols are disabled by default. You may need to go to **Start menu -> All Programs -> Microsoft SQL Server 2005 -> Configuration Tools** on the computer where the SQL Server is installed and start **SQL Server Configuration Manager**. Then, go to SQL Server 2005 Network Configuration and enable the TCP/IP protocol:



7. You may also need to enable the TCP/IP protocol in the **SQL Native Client Configuration -> Client Protocols** section:



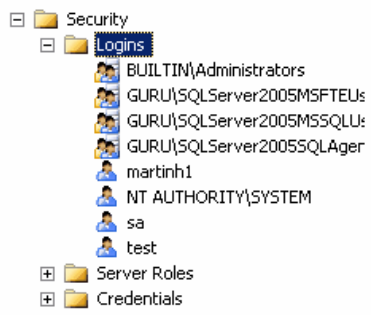
Error 2: Login failed for user 'xy'

Error message:

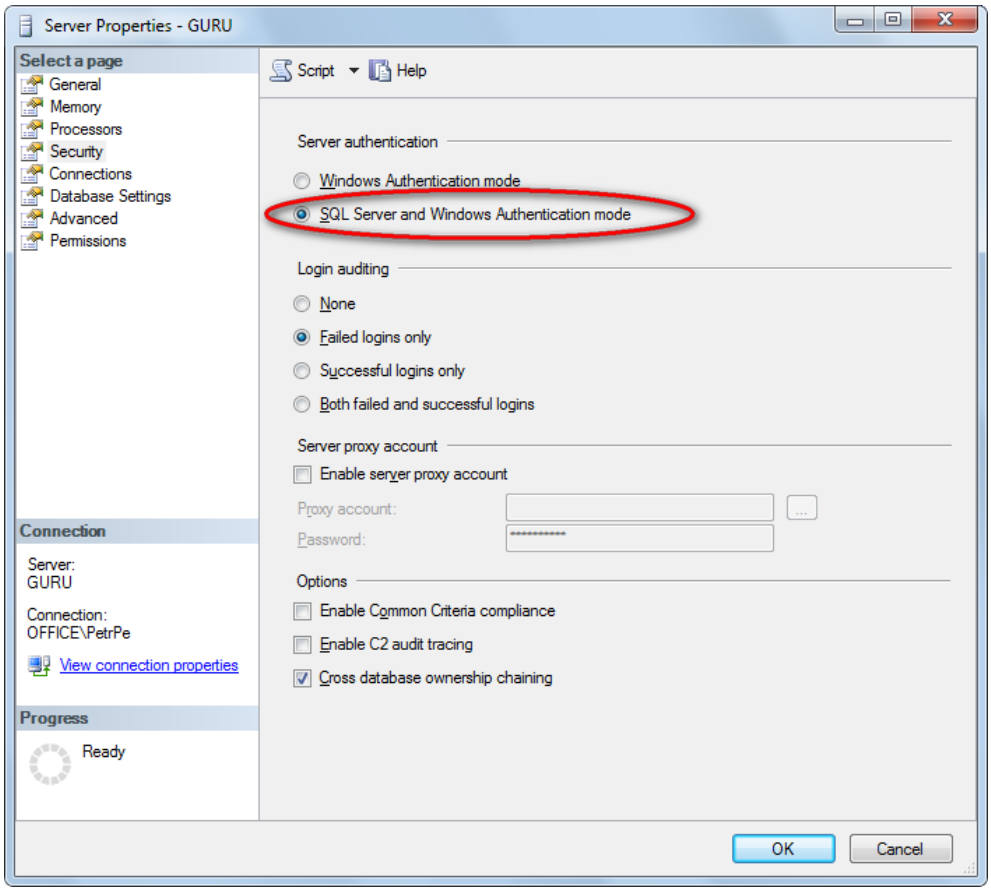
Login failed for user 'xy'

Troubleshooting for SQL Server account

If you're using SQL Server account with password, make sure you are using a valid user name and password. The login must be created on the server, it must be enabled and permissions to connect to the server must be granted to it. You can check the user account in Enterprise Manager/SQL Server Management Studio -> Server -> Security -> Logins:



Also, check the **Server Properties -> Security** dialog in Enterprise Manager/SQL Server Management Studio and make sure your server supports **SQL Server and Windows Authentication mode**:



Troubleshooting for Windows Authentication account

If you're using Windows Authentication account, the situation may be a little more complex and may require you to contact your network administrator. ASP.NET applications run under some particular local or domain account. This current account is displayed on the screen:

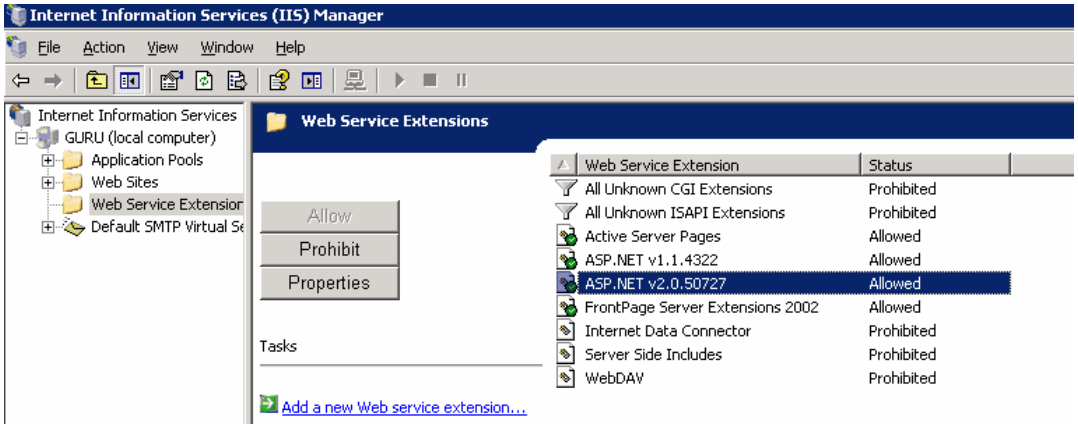
This account must have its own login with Windows authentication in the SQL Server. You can create the login in **Enterprise Manager\SQL Server Management Studio -> Security -> Logins** and grant appropriate permissions on the server to it. If you SQL Server is located on a different machine than your web server, you may need to configure your web application so that it runs under some domain account, rather than local account so that you can the login in the remote SQL Server.

If you do not succeed to configure Windows authentication, you may want to enable Windows and SQL Server Authentication on your SQL Server and use SQL Server account instead. You can learn more about SQL Server authentication in the **Troubleshooting for SQL Server account** section earlier in this chapter.

3.7.3 ASP.NET not working on Windows Server 2003

If you get the *404: Page not found* error or a similar error every time you request some ASPX page on your server and this is the first ASP.NET application installed and running on your server, it may be caused by configuration of Web Services Extensions on Windows Server 2003.

Go to **Control Panel -> Administrative tools -> Internet Information Services (IIS) Manager**, click **Web Services Extensions** and make sure that **ASP.NET 3.5** (or higher depending on the version you use) is **Allowed** (IIS 6 may display the **ASP.NET version as 2.0.50727** even if you have a higher one installed and registered) as shown in the following screenshot:

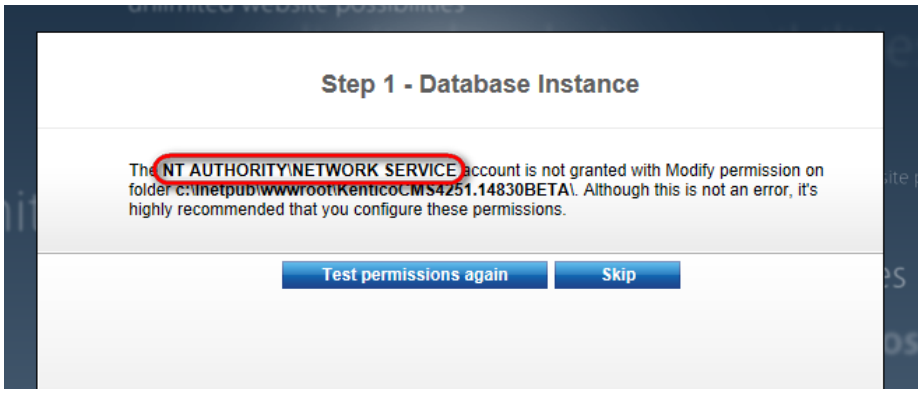


3.7.4 Disk permissions problems

3.7.4.1 Disk permissions problems

Kentico CMS is able to perform most operations without writing to disk. However, there are situations when the web application needs to write to the disk for optimal operations or performance, such as importing/exporting a site or storing uploaded files in the file system (which is optional).

If you receive the error message depicted below, saying that the web application cannot write to disk, you need to grant the **Modify** permissions on the whole website folder to the appropriate user account.



User account of the web application

The web application runs under a user account that depends on your environment. Please note that the accounts listed below are just the default ones, they may be different in your environment. However, the

name of the account is always displayed with the error message, as highlighted in the screenshot above.

- On Windows 7, the user account is the local **IIS_IUSRS** account by default.
- On Windows Vista or Server 2008, the user account is the local **NETWORK SERVICE** account by default.
- If you're using **Visual Studio's** built-in web server, it is running under your account.

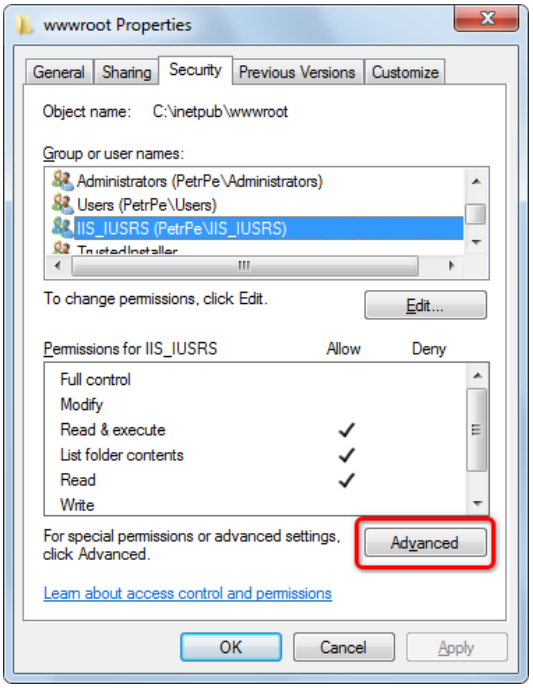
You can see the name of the user account under which the application runs in **Site Manager -> Administration -> System** dialog.

3.7.4.2 Solution on Windows 7

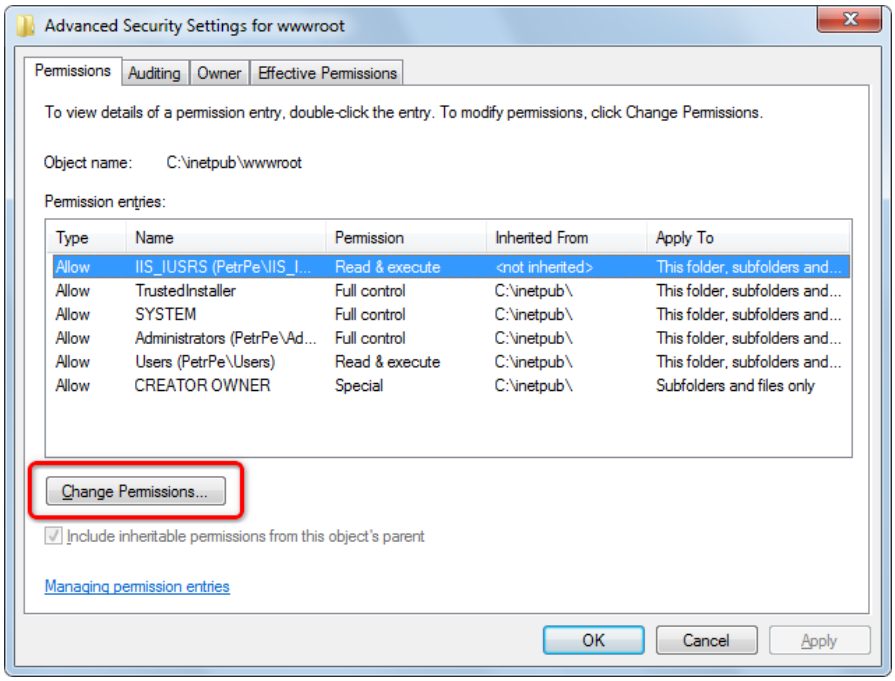
1. Open Windows Explorer, locate the folder with your website, right-click the folder and display its **Properties**. Choose the **Security** tab.

2. Verify that the account that you need to grant the permissions to (the name was displayed with the error message) is present in the **Group or user names** list. If not, click **Edit**, click **Add** in the pop-up dialog and add the required account to the list. Close the pop-up dialog when finished.

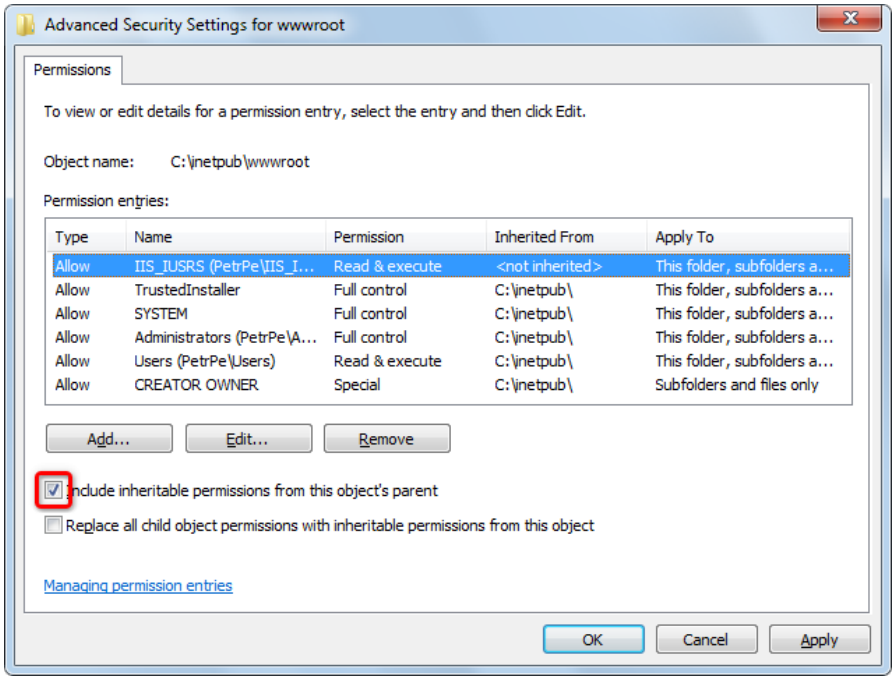
3. Back on the **Security** tab of the folder properties, select the appropriate account and click **Advanced**.



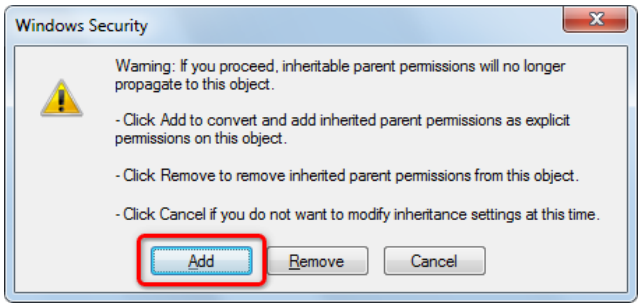
4. In the pop-up dialog, select the account again and click **Change Permissions**.



5. Disable the **Include inheritable permissions from this object's parent** option.

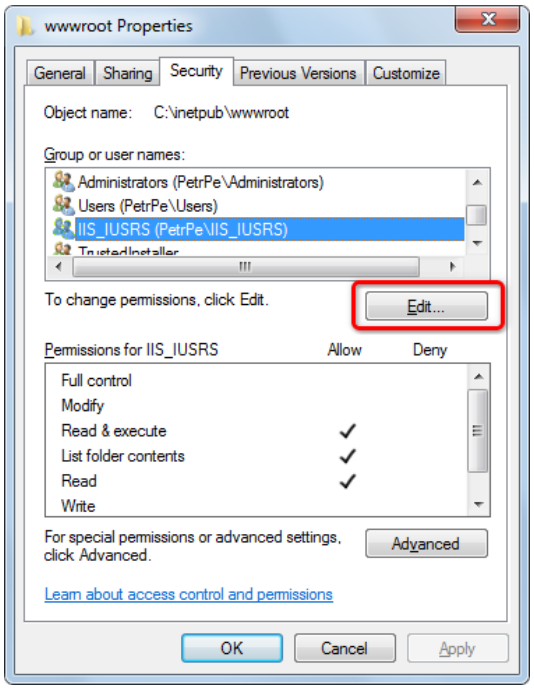


6. In the pop-up dialog, click **Add**.

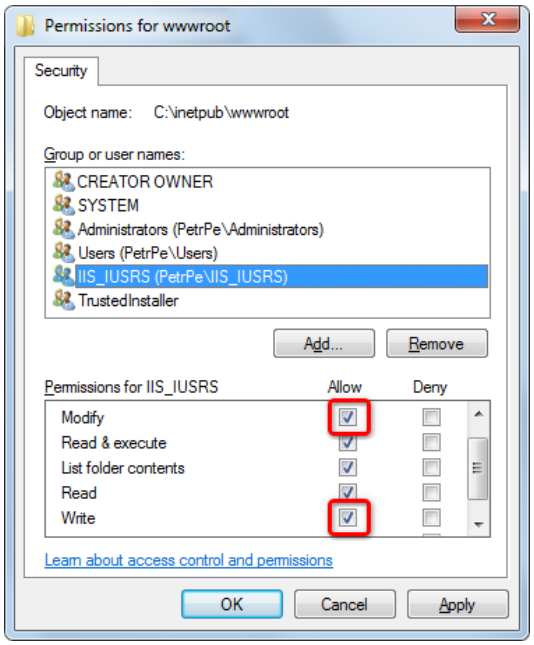


7. When the dialog closes, click **OK** to close the dialog under it. Click **OK** again to close the dialog under the previous one.

You are back in the folder properties dialog now. Select the account and click **Edit**.



8. Check the **Allow** check-box for the **Write** and **Modify** permissions and click **OK**.



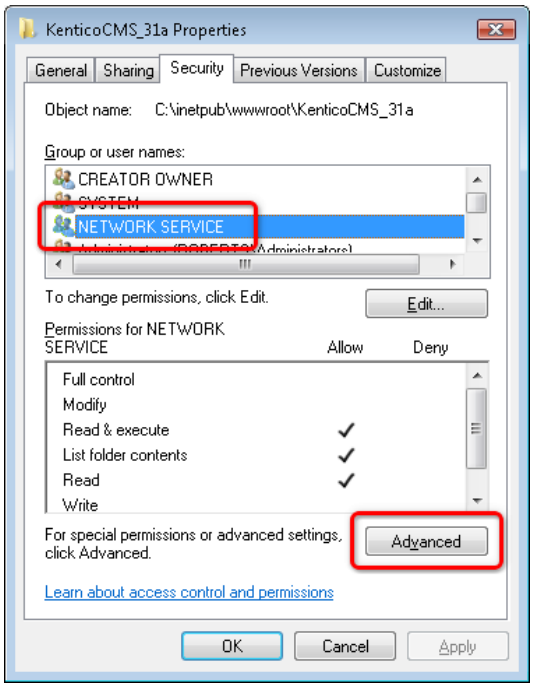
9. You have assigned the account with the required permissions. Kentico CMS should now be able to perform all disk write operations and therefore work correctly.

3.7.4.3 Solution on Windows Vista or Server 2008

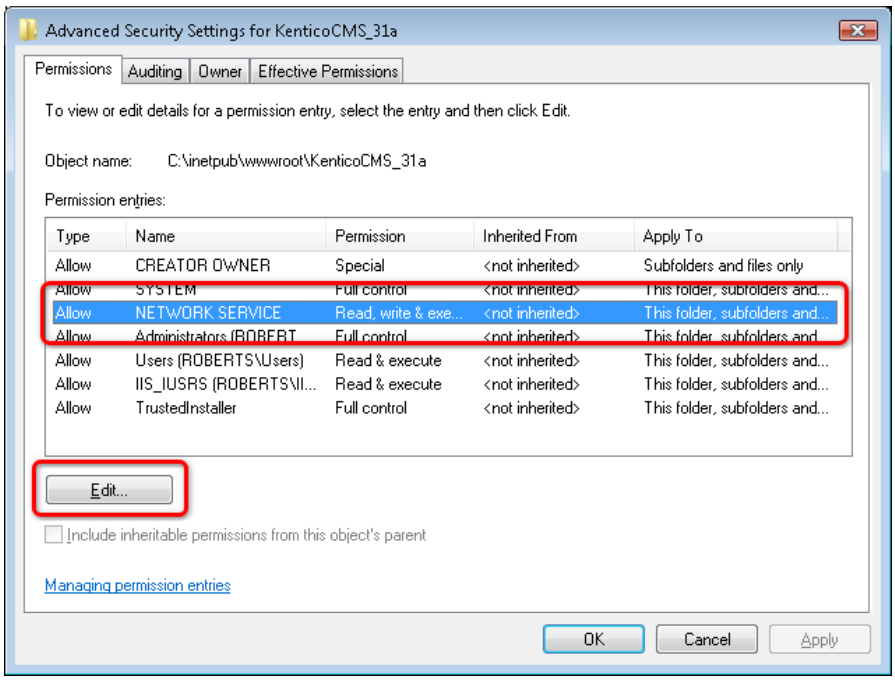
1. Open Windows Explorer, locate the folder with your website, right-click the folder and display its **Properties**. Choose the **Security** tab.

2. Verify that the account that you need to grant the permissions to (the name was displayed with the error message) is present in the **Group or user names** list. If not, click **Edit**. In the pop-up dialog, click **Add** and add the required account to the list. Close the pop-up dialog when finished.

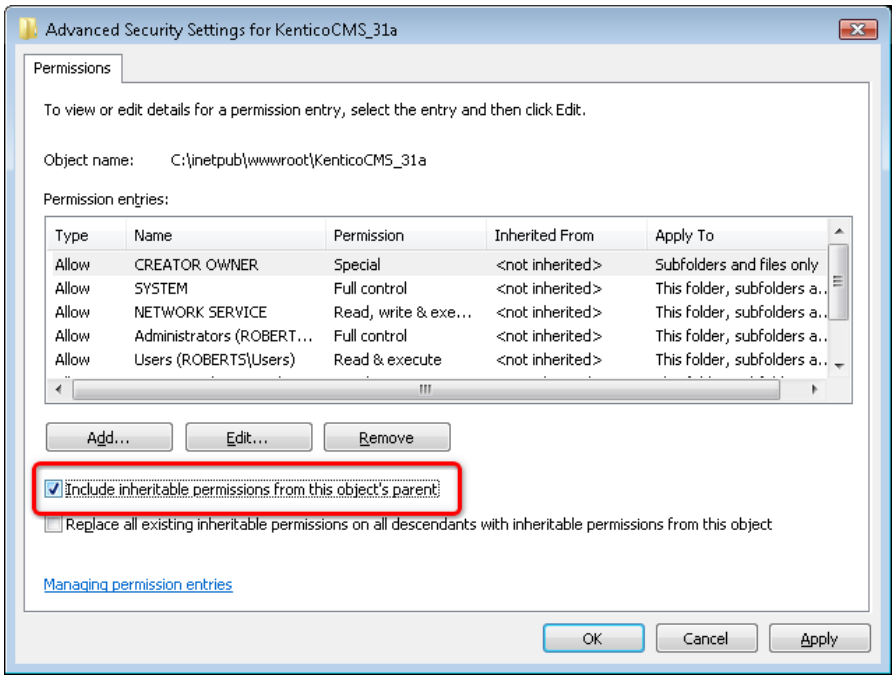
3. Select the required account and click **Advanced**.



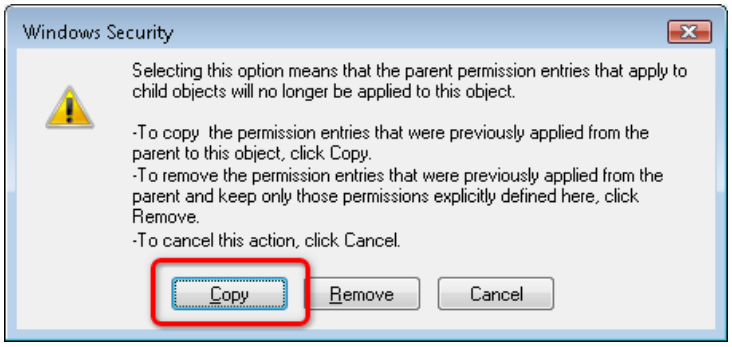
4. In the pop-up dialog, select the account again and click **Edit**.



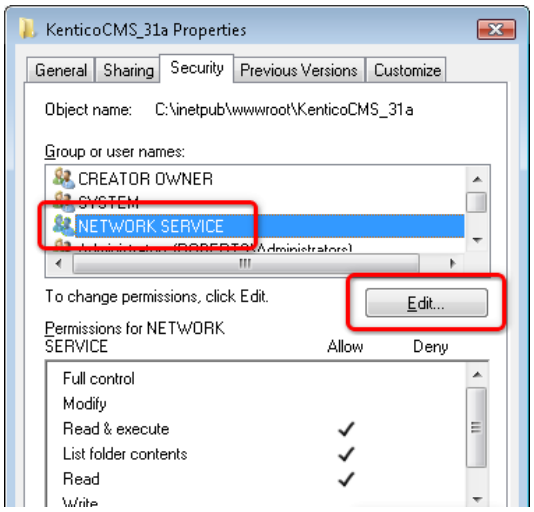
5. Disable the **Include inheritable permissions from this object's parent** option.



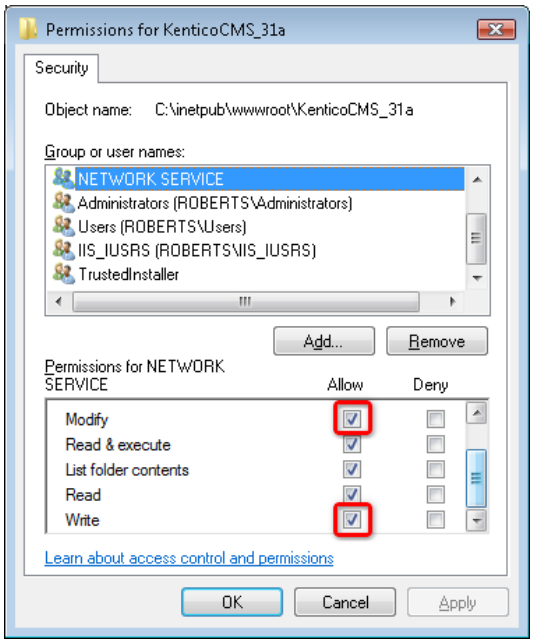
6. On the pop-up dialog, click **Copy**.



7. When the dialog closes, click **OK** to close the dialog under it. Click **OK** to close the dialog under the previous one. You are back in the folder properties dialog now. Select the account again and click **Edit**.



8. Check the **Allow** check-box for the **Write** and **Modify** permission and click **OK**.



9. You have assigned the account with the required permissions. Kentico CMS should now be able to perform all disk write operations and therefore work correctly.

3.8 System backup and recovery

Backup

To backup the data of Kentico CMS instance, you need to:

1. Backup the Kentico CMS database.
2. Backup the files of the Web application.

Recovery

To recover the system, you need to:

1. Recover the Kentico CMS database.
2. Recover the website files and create a virtual directory for it.

3.9 Silent Install

3.9.1 Overview

Silent Install is a tool which enables you to install Kentico CMS directly from a command line, without any user interface and user interaction involved. It also allows you to modify an existing installation and add or remove components contained in this installation. You only need to prepare a configuration XML file and execute a command from the command line with this XML file as a parameter.
The main purpose of the tool is the possibility of automated installation and modification of Kentico CMS.

Silent Install is capable of substituting the whole installation procedure including:

- Setup (KenticoCMS.exe) - Silent Install installs the program files to a specified location on your hard drive.
- Web Installer - Silent Install creates a new web project and optionally configures IIS.
- Database setup - Silent Install creates a new database with system tables and basic data on your SQL server.
- New site wizard - Silent Install allows you to install websites based on available web templates.

You can download a ZIP package containing the Silent Install tool and sample XML files from the following location:

- http://www.kentico.com/downloads/SilentInstall.zip.

Silent installation procedure

1. Create a configuration XML file using a text editor.
   - You can use any of the sample XML files stored in the *Examples* folder within the *SilentInstall.zip* package and edit it to suit your needs.
   - The *Examples* folder also contains the *SilentInstall.xsd* file, which is an XML schema file defining the configuration XML file format. You can use it to validate the XML configuration file you have created.
   - You can find a reference on the required format of the configuration XML file in the XML configuration topic.

2. Copy the XML file to the folder containing Kentico CMS installation files.

3. Execute a command with the **SilentInstall.exe <configuration XML file name>.xml** syntax in the folder.

```
SilentInstall.exe CorporateSite.xml
```

The installation will be executed and carried out according to the configuration in the XML file.

3.9.2 XML configuration

This topic provides reference on the format of the XML file used as a source of the Silent install configuration.

Instead of creating one from scratch, it is highly recommended to use one of the sample XML files located in the *Examples* folder within the *SilentInstall.zip* package and customize it according to your specific needs.

The *Examples* folder also contains the *SilentInstall.xsd* file, which is an XML schema file defining the configuration XML file format. You can use it to validate the XML configuration file you have created.

Boolean attribute values

You can enter the boolean attribute values in these ways:

True value	False value
TRUE	FALSE
Yes	No
1	0

Supported macros in attribute values

You can use macros in the following XML attributes to get the respective values dynamically:

- SilentInstall -> LogFile
- Setup -> SetupFolder
- IIS -> TargetFolder
- SQL -> Database

Macro expression	Description	Sample resolved value
{%shortversion%}	Build number of the installed Kentico CMS version.	4709.166
{%longversion%}	Full version number of the installed Kentico CMS version.	7.0.4709.166
{%programfiles%}	Path to the *Program Files* folder on your local drive.	c:\Program Files (x86)\
{%machine%}	Name of the current machine.	PC-01
{%username%}	User name of the currently used account.	Andy
{%date%}	Current date in format *yyyy-mm-dd*.	2012-12-21

{%time%}	Current time in format *hh:mm:ss*.	14:53:23

Note that these macros are completely independent of standard Kentico CMS macros that can be used in the system's user interface.

SilentInstall XML element

The root element of the XML file is *SilentInstall*. Using the attributes of this element, you can configure general options of the installation:

Attribute name	Description	Accepted values
ShowProgress	Determines if the Silent Install tool displays the progress of the installation or not. • **COMMAND_PROMPT** - the progress is shown in the command line. • **NO** - the progress is not shown anywhere.	NO | COMMAND_PROMPT
Log	Enables logging of import progress to a file.	TRUE | FALSE
OnError	Determines the action the tool performs when an error occurs during the installation.	STOP | CONTINUE
LogFile	File where the log is saved. The default value is *<installation path>\setup.log*.	full path to a file
CheckRequirements	Indicates, if the tool checks whether the correct .NET version is installed on the target machine before executing the installation. If the installed .NET version is not matching, the installation is aborted.	TRUE | FALSE

Setup XML element

The first sub-element that should be present under *SilentInstall* is *Setup*. This element allows you to configure the general options by means of attributes:

Attribute name	Description	Accepted values
NET	.NET version installed on the target machine.	3.5 | 4.0 | 4.5
SetupFolder	Folder where the tool installs the Kentico CMS setup files (not the web project).	full path to a folder
WebProject	Determines if the project is installed as an ASP.NET web site, ASP.NET web application, or as a Windows Azure project.	WebSite | WebApplication | WindowsAzure
InstallContextHelp	Indicates, if Context help (built-in documentation accessible by clicking the icon throughout the user interface) is installed with the project.	TRUE | FALSE

OpenAfterInstall	Indicates, if the tool opens the newly installed website in a new browser window when the installation finishes.	TRUE | FALSE
RegisterCounters	Indicates, if performance counters for Health Monitoring should be registered. This option has the same functionality as the **Register performance counters for Health Monitoring** option in Step 4 of Kentico CMS Web Installer.	TRUE | FALSE
InstallWinServices	Indicates, if the tool installs Kentico CMS Windows services in Windows.	TRUE | FALSE
DeleteExisting	Indicates, if the the tool deletes the existing setup files in a folder with the same name as set in the *SetupFolder* attribute.	TRUE | FALSE

IIS XML element

The *IIS* sub-element is also located directly under *SilentInstall*. Its attributes allow you to adjust IIS-related settings of the installation:

Attribute name	Description	Accepted values
AppPool	Type of IIS application pool which the new website will use. • **Native** - the application pool uses the mode that is default in the installed IIS version (*Classic* for IIS versions prior to 7 and *Integrated* for IIS 7 and higher).	Native | Classic | Integrated
WebSite	Name of the website in IIS.	string value
TargetFolder	Path to the folder where IIS websites are stored (typically *C:\Inetpub\wwwroot*).	full path to a folder
RootFolder	Indicates, if the tool installs the website into the IIS root. • **TRUE** - the website is installed into the IIS root and no virtual directory is created. • **FALSE** - the website is installed into a new virtual directory.	TRUE | FALSE
DeleteExisting	Indicates, if the the tool deletes an existing website with the same name as set in the *WebSite* attribute.	TRUE | FALSE
Location	Determines where will the tool install the web files: • **Local** - on a local IIS server. • **VisualStudio** - on a build-in server which is	Local | VisualStudio | Remote | Modify

	included in Visual Studio or Visual Web Developer Express Edition. • **Remote** - copies the files into a temporary folder on your disk. You will need to copy these files to your production server. • **Modify** - modifies an existing installation (adds or removes components) on a local machine. See Kentico CMS Developer's Guide -> Installation procedure -> Web installer for more details.	

SQL XML element

The *SQL* sub-element of the *SilentInstall* element allows you to configure database-related settings of the installation. It covers the configuration that is normally performed in Step 1 and Step 2 of the **Database setup**.

Attribute name	Description	Accepted values
Database	Name of the target database.	string value
SqlName	User name of the account used for access to the SQL server.	string value
SqlPswd	Password for the SQL server account.	string value
Server	Name of the target SQL server.	string value
Operation	Determines whether the tool creates a new database modifies an existing one. The default value is **New**.	New | Modify
Authentication	Type of the authentication used to access the SQL server.	SQL | WINDOWS | WIN
DeleteExisting	Indicates, if the tool deletes an existing database with the same name as set in the *Database* attribute.	TRUE | FALSE

Notification XML element

The *Notification* sub-element allows you to configure the automatic notification e-mail messages. The system sends them to the specified e-mail address when an error occurs during the installation.

Attribute name	Description	Accepted values
Enabled	Indicates, if the tool sends notification e-mails when an error occurs during the installation.	TRUE | FALSE

From	E-mail address of the sender of the notification e-mail. This value is required in the *Notification* element.	e-mail address
To	E-mail address of the recipient of the notification e-mail. This value is required in the *Notification* element.	e-mail address (or multiple addresses separated by a semicolon (;))
Server	Name of the SMTP server used to send out the notification e-mail.	string value
Subject	Subject of the notification e-mail message.	string value
UserName	User name of the SMTP server account used to send out the notification e-mail.	string value
Password	Password for the SMTP server account used to send out the notification e-mail.	string value
SSL	Indicates, if SSL is used when sending the notification e-mail.	TRUE | FALSE
AttachLogFile	Indicates, if the tool includes a compressed installation log file as an attachment.	TRUE | FALSE

WebTemplates, UICultures, Modules and Dictionaries XML elements

The *WebTemplates*, *UICultures*, *Modules* and *Dictionaries* elements allow you to configure which web templates, UI cultures, modules and dictionaries will be installed by the Silent Install tool. If you chose to modify an existing Kentico CMS installation (by specifying the *Modify* value for the *Location* attribute), you can also use these elements to add new components to the existing installation or to remove unwanted components.
Each of these elements only has a single attribute:

Attribute name	Description	Accepted values
type	• **InstallAll** - the tool adds all components of the type to the installation. However, you can limit added components in the sub-elements by specifying *No* or *Remove* in their *operation* attribute. • **RemoveAll** - the tool removes all components of the type from the installation. However, you can limit removed components in the sub-elements by specifying *No* or *Add* in their *operation* attribute. • **Mix** - only configuration in the *operation* attribute of the sub-elements will be taken into account.	Mix | InstallAll | RemoveAll

Each of the *WebTemplates*, *UICultures*, *Modules* and *Dictionaries* elements can have any number of sub-elements named *WebTemplate*, *UICulture*, *Module* or *Dictionary*. These sub-elements represent particular web templates, UI cultures, modules or dictionaries and have the following attributes:

Attribute name	Description	Accepted values
name	Name of the web template, UI culture, module or dictionary.	string value
operation	The operation that should be performed with the component: • **Add** - the component is added to the installation. • **Remove** - the component is removed from the installation. • **No** - no action is taken.	Add | Remove | No

WebSites XML element

The *WebSites* element lets you configure which websites will be installed based on available web templates. It has no attributes itself — it only contains *WebSite* sub-elements representing particular websites to be installed. Each *WebSite* sub-element has the following attributes:

Attribute name	Description	Accepted values
domain	Domain used by the website.	string value
displayname	Name of the website used in Kentico CMS user interface.	string value
codename	Name of the website used in code.	string value
webtemplatename	Code name of the chosen web template.	available web template code name
runsite	Indicates, if the tool should run the website after it is imported to the system.	TRUE | FALSE

Licenses XML element

You can add any number of *License* sub-elements under the *Licenses* element. These sub-elements represent particular licenses for separate domains. You should state the domain in the *domain* attribute and add the actual key as a sub-element wrapped in a CDATA enclosure.

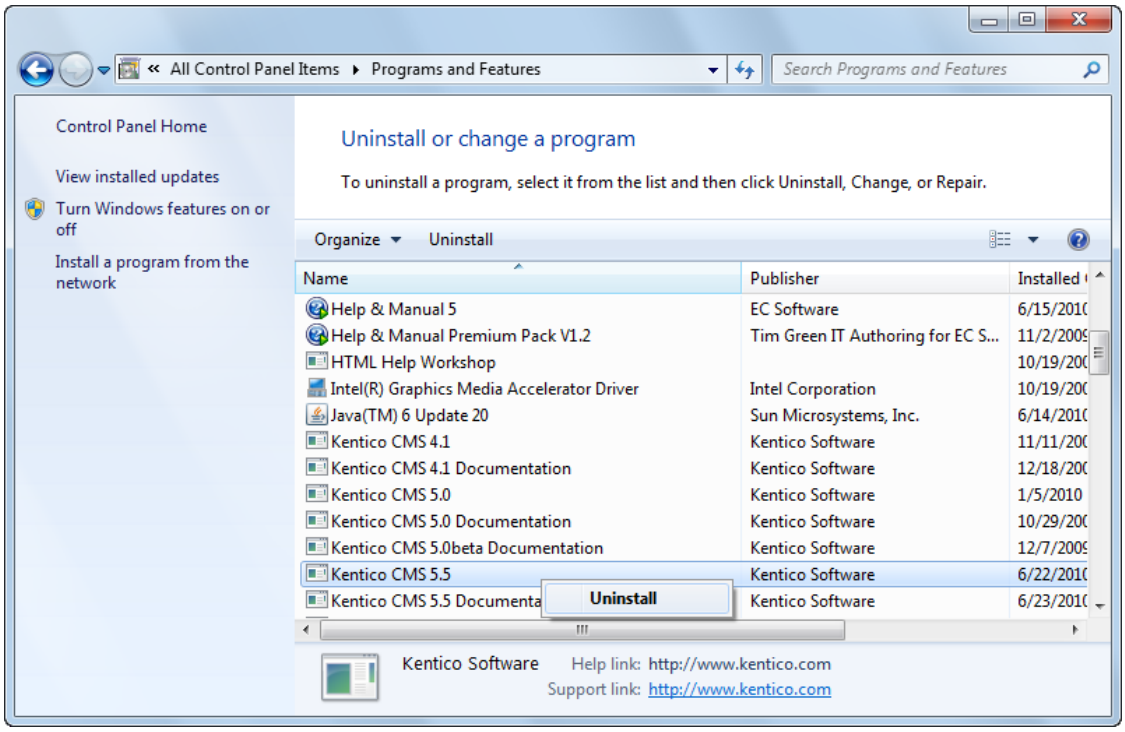
Example

Here you can find an example of the configuration file. It installs Kentico CMS 7.0 with the sample Corporate site website and all the available web templates, UI cultures, modules and dictionaries.

```
<SilentInstall ShowProgress="COMMAND_PROMPT" Log="TRUE"
OnError="STOP" LogFile="{%programfiles%}\KenticoCMS\7.0\setup.log"
CheckRequirements="TRUE">
  <Setup NET="4.5" SetupFolder="{%programfiles%}\KenticoCMS\7.0"
InstallContextHelp="TRUE" />
  <IIS AppPool="Native" Website="Default Web Site" TargetFolder="c:
\Inetpub\wwwroot\KenticoCMSCorporateSite70" RootFolder="FALSE"
Location="Local" />
  <Sql SqlPswd="password" Database="KenticoCMSCorporateSite70"
Operation="New" SqlName="login" Server="Artemis"
Authentication="SQL" />
  <WebTemplates type="InstallAll">
  </WebTemplates>
  <UICultures type="InstallAll">
  </UICultures>
  <Modules type="InstallAll">
  </Modules>
  <Dictionaries type="InstallAll">
  </Dictionaries>
  <WebSites>
    <WebSite domain="localhost" displayname="Corporate site"
codename="CorporateSite" webtemplatename="CorporateSite" />
  </WebSites>
  <Licenses>
    <License domain="localhost"><![CDATA[DOMAIN:localhost
PRODUCT:CF07
EXPIRATION:00000000
PACKAGES:
SERVERS:1
p8NrcXDSRiiEdH6Paef6MFISFY4Mihhwz9E+75fDKp1srPgxhTxEoLt0P2XXMkmCRSwh
Qk85/
zjp017iCUIpwHhfgNQv/83ILVx3bIAEIZReY2Grs4Lah5jHSLlq3RUCX6d5ZL2Q2lxhK
ckPxMWjVhBlvDKLMttek+56QZmMp8oQlEMlqGYCIV+HMgD66Ob5ukdKYKvCw0Zcd2nhi
+7W2KqJcWCRtRVxIY/Xi69ZgpT/
Mae/8cxEfxZ+xzfw0Tn81Qf5vxVUkfG5UwVdmBQ1NFMqA6OTvx60kkRjGkUFNbsJVogs
J+WdMXr/MNhHx+qFAuMLdCOL13h4WMr/y8M+yA==]]></License>
  </Licenses>
</SilentInstall>
```

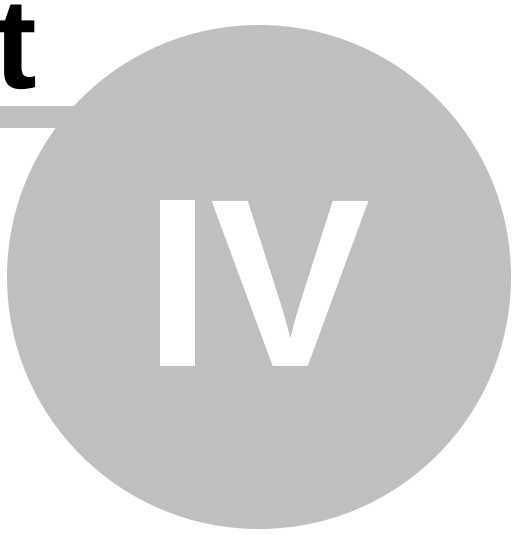
3.10 Uninstallation

To uninstall Kentico CMS from your computer, open **Start -> Control Panel -> Programs and Features**. Right-click **Kentico CMS** and choose **Uninstall** from the context menu.



The uninstaller will delete files created on your computer except the deployed instances of Kentico CMS - it means that the database instances and the website folders will not be removed and you have to delete them manually if needed.

Part



Content management

4 Content management

4.1 Overview

The content of a website's pages can be managed through a browser-based, WYSIWYG user interface. Editors need to have the appropriate permissions and must first authenticate themselves by signing in with a user name and password.

Links for accessing the user interface

The Kentico CMS user interface, i.e. **CMS Desk**, **On-site editing mode** and **Site Manager**, can be accessed via the links described below, where the first part of the link URL is dependent on the site's domain name and the virtual directory specified during the installation. If the default URL of your website is *http://localhost/KenticoCMS*, then the following URLs may be used:

- **CMS Desk**: *http://localhost/KenticoCMS/**CMSDesk***
- **On-site editing**: *http://localhost/KenticoCMS/**CMSEdit***
- **Site Manager**: *http://localhost/KenticoCMS/**CMSSiteManager***

Default user name and password

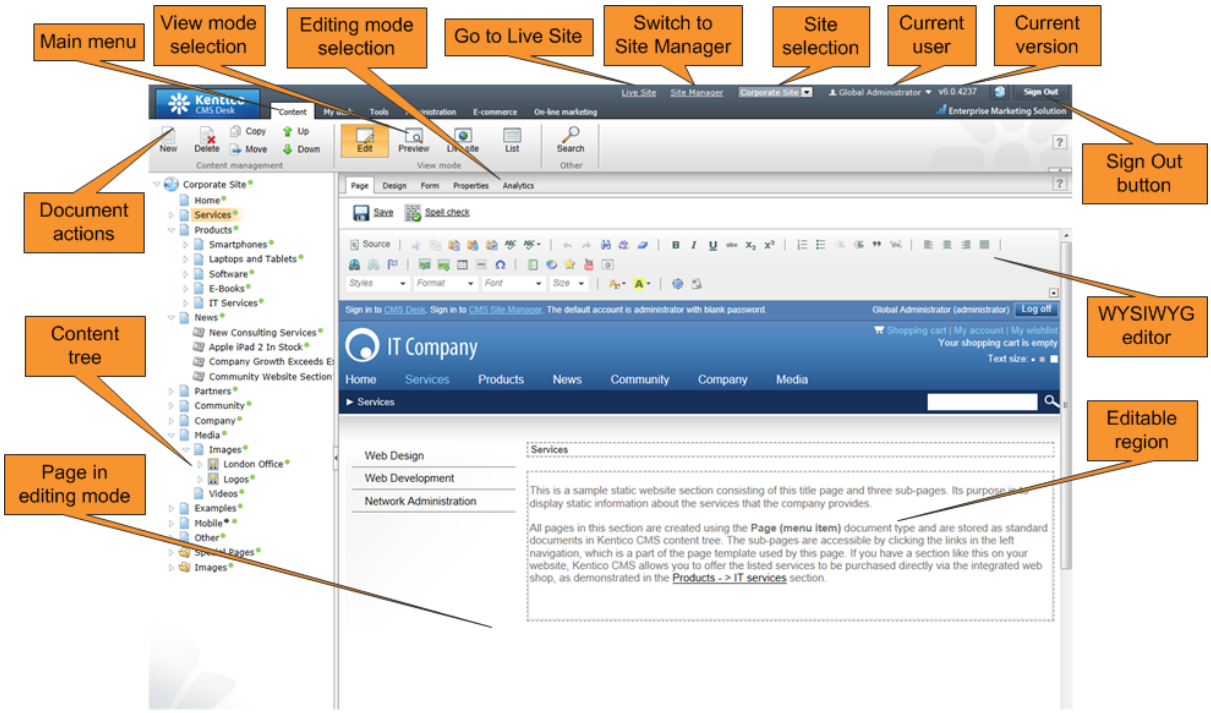
To log in to a website's Kentico CMS interface for the first time, use the default logon credentials:

- The default user name is **administrator**.
- The default password is **blank (no password)**.

It is highly recommended to change the password after you finish the installation.

User interface overview

The following figure shows the CMS Desk user interface.



The user interface consists of the following main sections and features:

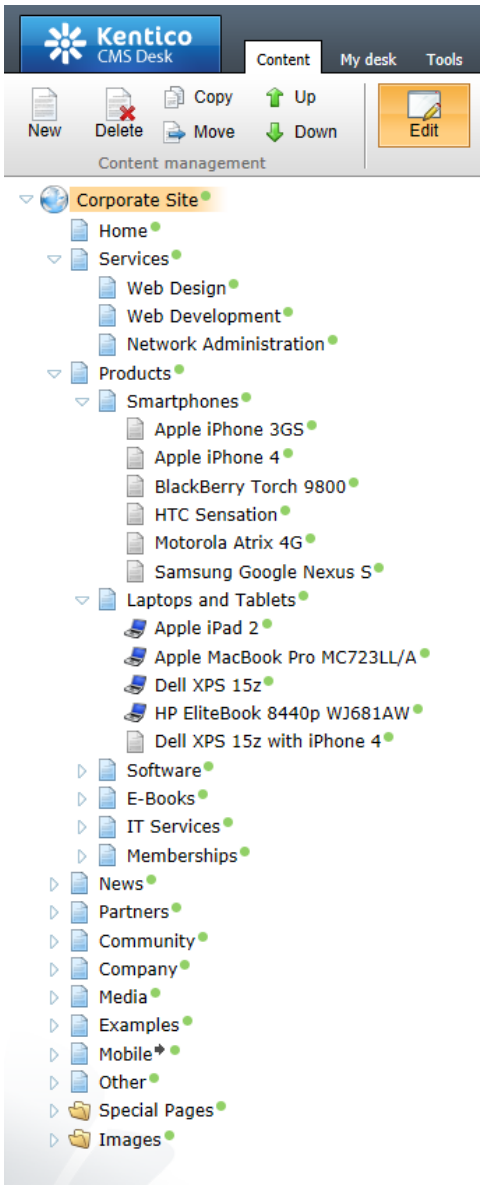
- **Main menu** with Content, My Desk, Tools, Administration, E-commerce and On-line marketing tabs.
- **Content tree** that represents the site map of the website. It allows you to organize the site's structure or select a specific document for editing in the content section.
- **Document actions** toolbar with buttons for creating new documents, deleting, copying, moving and sorting documents.
- **View mode selection** - allows you to choose between editing, preview, live site view and list view.
- **Editing mode selection** - you can choose to edit page content, design the page template, edit the document's fields, product properties or document properties.
- **Live Site** - this action redirects you to the title page of the currently edited website, logged under the same user account that you used to log into CMS Desk. This is a more convenient way than using the **Sign out** button and logging in on the live site afterwards.
- **Site Manager** - redirects you to Site Manager, the other part of the system's administration interface. This option is only available for global administrators.
- **Site selection** - this drop-down list is used to select the currently edited website. Only those websites that the current user can edit are available in the list.
- **Current user** - displays the name of the current user.
- **Current version** - version number of the Kentico CMS installation.
- **Sign Out button** - clicking this button logs you out of the user interface and redirects you to the title page of the live site. This button is only displayed if *Forms authentication* is used. When using *Windows authentication*, the link is not displayed.
- **WYSIWYG editor** - allows you to edit text, change its formatting and insert graphics or other items. It is available for **Editable regions** on the *Page* tab, as well as when editing document fields on the *Form* tab.
- **Page in editing mode** - this is where you can view and edit the document selected in the content tree, in the mode selected in the view mode and editing mode toolbar.

See also: Kentico CMS overview -> Where is the content stored?, Kentico CMS overview -> How do I

edit content?

4.2 Organizing pages, files and documents

All content in Kentico CMS is stored in a **tree hierarchy**. You can see the content tree on the left in **CMS Desk -> Content**:

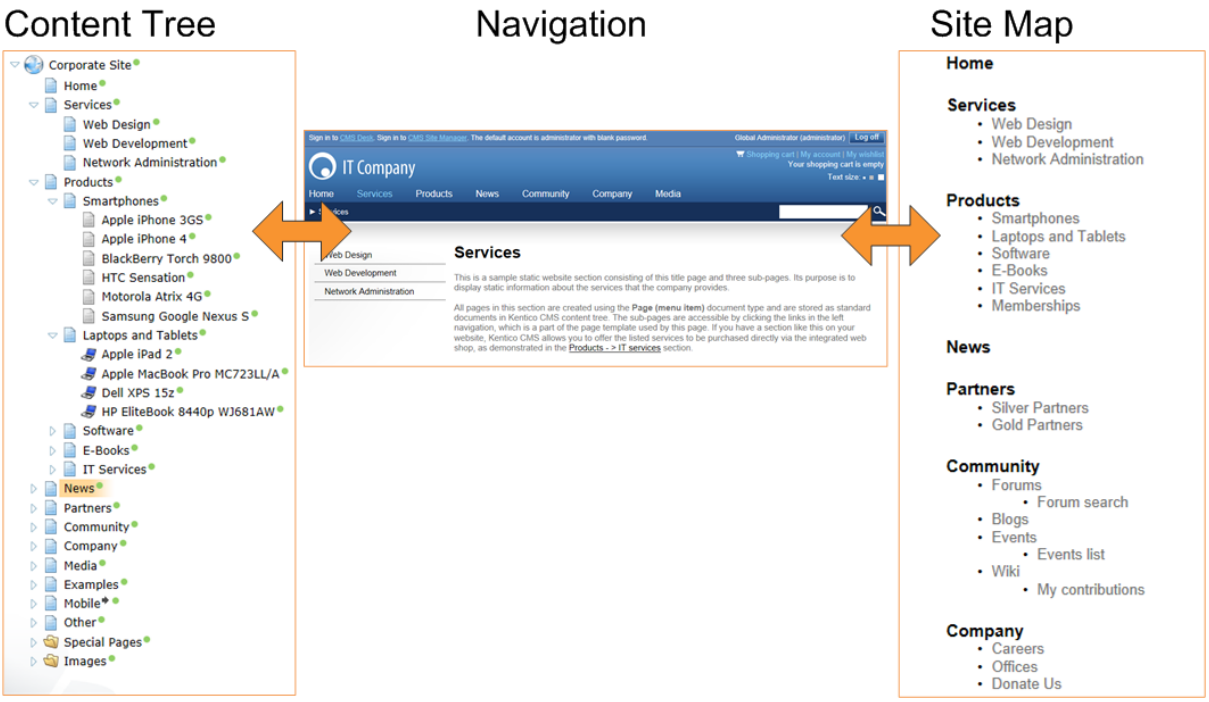


The tree hierarchy provides many advantages:

- It organizes the page in a logical structure that represents the (dynamic) site map.
- It ensures easy-to-navigate information architecture.
- It provides a logical categorization of pages and documents.
- The content of sub-pages can be nested inside the parent pages.
- The position of the document is reflected in its URL, which consists of the document path in the tree

hierarchy,
such as */products/lcd-displays/nec-52vm.aspx*.
- The structure allows you to define permissions (and other types of settings) for a particular site section and have underlying items inherit them.

The following figure shows how the content tree defines the navigation and site map of the website:



Document types

Each document is of some type - it can be a page, a news item, an article, a product specification, etc. Each document type has its own:

- fields (data structure)
- editing form layout
- transformations (appearance and design)
- queries

and other settings.

Document types are fully customizable - you can add, modify and delete custom fields. The advantage of using custom document types is that you can define custom structure of documents and store content (data) separated from design. This can be done in **Site Manager -> Development -> Document types**.

More details can be found in the Document types chapter.

Pages and documents

All items in the content tree are in fact documents. However, there is a special type of documents

called **pages**. The pages (such as */Home*, */Products*, */Services*) display the content and they are displayed as menu items by default (this can be also customized).

Unlike pages, structured documents (such as the news items under the */News* section of the sample Corporate Site) contain structured data that can be displayed on the pages.

While pages usually contain unstructured content in the form of editable regions that can be edited on the **Page** tab, the structured documents contain structured and typed data stored in document type-specific database tables and edited on the **Form** tab.

You will typically use **structured documents** when you need to display a **list of items**, such as list of news, list of products, etc.

Page versus Form

There are two types of content: content stored in editable regions on the page and content stored in forms. The following table compares both approaches:

	Editable regions on the page	Form
Content structure	Simple content structure, only text-based content.	Complex content structures, typed data, such as text, date-time, numbers, etc.
Validation	Only basic validation rules for minimum and maximum length.	Complex validation rules, including regular expressions and custom form controls with custom validation code.
Display	The content is displayed in the context of the page providing truly WYSIWYG editing.	The content is displayed using XSLT or ASCX transformations using special controls or web parts.
Storage	The content is stored in a single XML document in the document properties.	The content is stored in a separate database table. Each field has its own column. The data can be easily modified using SQL queries or API.
Examples of use	Home page, contact page. Generally: pages with simply structured or unstructured, text-only content. The editable regions are usually used only in connection with documents of type **Page (menu item)**.	News, product specification, event details, job opening, etc. Generally: pages with structured content where you need to separate content from design and keep the content in its original data type. The form-based content is usually used in connection with documents of type **News**, **Product, Article,** etc.

Organizing media files

There are two types of files you need to manage on the website:

- **Website design files** - images and flash files that are a part of the website's design, such as logo, background or menu images, etc. These should be stored in the file system as a part of the application theme as explained in the App themes topic.
- **Media files and document files** - images, flash movies, word documents, PDFs, etc. that are published on the website and are part of the content editable by editors. These should be uploaded to the content tree as documents so that they can be managed by the content editors and so that you can apply all content-related features (permissions, workflow, versioning, multilingual support) to files as well.

You can find more details on files in the File management chapter.

See also: Where is the content stored?

4.3 Storing data effectively

When storing data in Kentico, you should consider whether you want to store it in **Documents**, **Custom tables** or **Media libraries**. Each of the options was created with different use in mind and thus provides various advantages over the others. Refer to the following text to see a comparison of the approaches:

Documents

Documents offer a hierarchical tree data structure for storing data. You can use them in a workflow or to hold multilingual content without having to use localization macros.

Custom tables

Custom tables provide you with better performance than Documents when storing data in a flat structure. Consider using custom tables when you need to store a large amount of structured items. To localize custom table data, you need to use localization macros.

Learn more about custom tables.

Media libraries

Media libraries are designed to store large files — not exclusively of media character — such as videos, high-res images or packaged files. You can access media library files without overhead as there is no need to query a database in order to access them.

Learn more about Media libraries.

Comparing documents with custom tables

	Documents	Custom tables
Can hold traditional data types (int, float, ...)	✓	✓
Can hold binary stream data (files)	✓	✗
Can hold one-to-many data (radio buttons, drop-down lists)	✓	✓

Data can be formatted using transformations	✓	✓
Can be used as an E-commerce product	✓	✗
Can be displayed by 'Listings' web parts	✓	✓
Can be displayed by 'Navigation' web parts	✓	✗
Workflow	✓	✗
Versioning	✓	✗
Multilingual content	✓	✓[1]
Hierarchical data structure	✓	✗
Each record has its own URL	✓	✗[2]
Data can be accessed using API	✓	✓
Importable / Exportable	✓	✓
Performance		
Recommended for large data sets when using flat data structure	✗	✓
Number of database tables that store the data	3+	1
Recommended for large binary data	✗	✗
Binary data can be stored in file system	✓	✗

1) You can translate custom table data using localization macros only.
2) You can access custom table records via listing web parts and query string URL parameters.

Comparing documents with media libraries

	Documents	Media libraries
Recommended for large binary data	✗	✓
Direct access to the data without querying database	✗	✓
Number of database tables that store the data	3+	2
Binary data can be stored in file system	✓	✓

4.4 Editing content (CMS Desk)

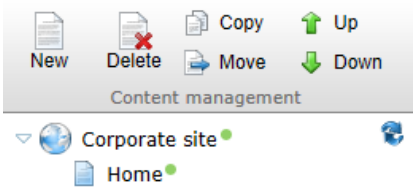
4.4.1 Basic content tree actions

This topic gives an overview of actions that can be performed with documents in the content tree. The actions can be executed either from the ribbon above the content tree or from a **context menu** displayed by right clicking a document.

The ribbon offers the following actions:

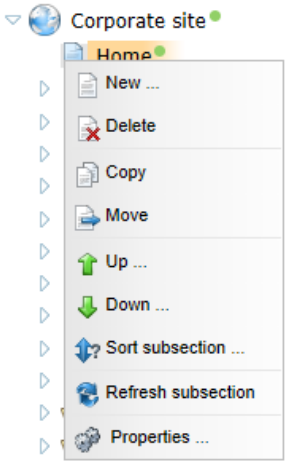
- **New** - creates a new document under the currently selected one
- **Delete** - deletes the currently selected document
- **Copy** - creates a copy of the currently selected document in a location specified in a pop-up dialog

- **Move** - moves the currently selected document to a location specified in a pop-up dialog
- **Up** - moves the currently selected document above the one which is above it at the same level
- **Down** - moves the currently selected document below the one which is below it at the same level
- **Refresh content tree** - displayed only on mouse over of the content tree's top right corner; refreshes the content tree so that it shows the current content (useful e.g. for multi-user editing or when blog posts are published using MetaWeblog API)



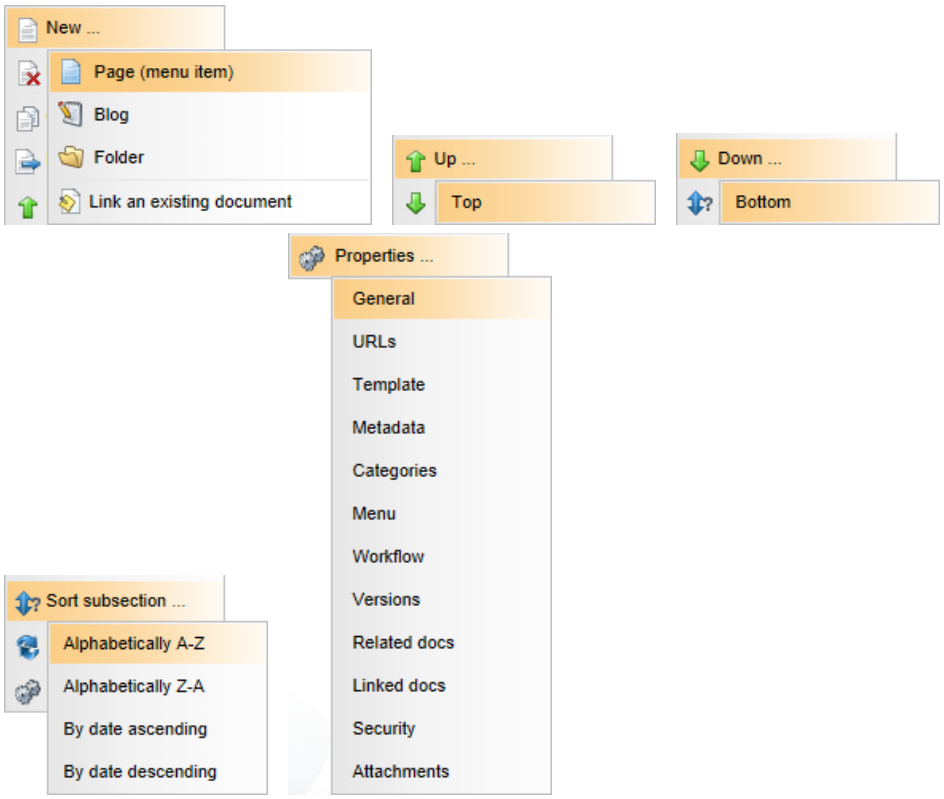
If you right-click a document in the content tree, a **context menu** appears. The menu offers the same actions as listed above, while the following extra options are available on top of the ones from the ribbon:

- **Sort subsection** - sorts documents under the clicked one, while both ascending and descending order by date or alphabet is achievable
- **Refresh subsection** - refreshes the content tree subsection under the clicked document (useful e.g. for multi-user editing or when blog posts are published using MetaWeblog API)
- **Properties** - opens the document's **Properties -> General** tab



If you right-click some of the items in the context menu or hold the mouse pointer above it, a **sub-menu with additional options** appears:

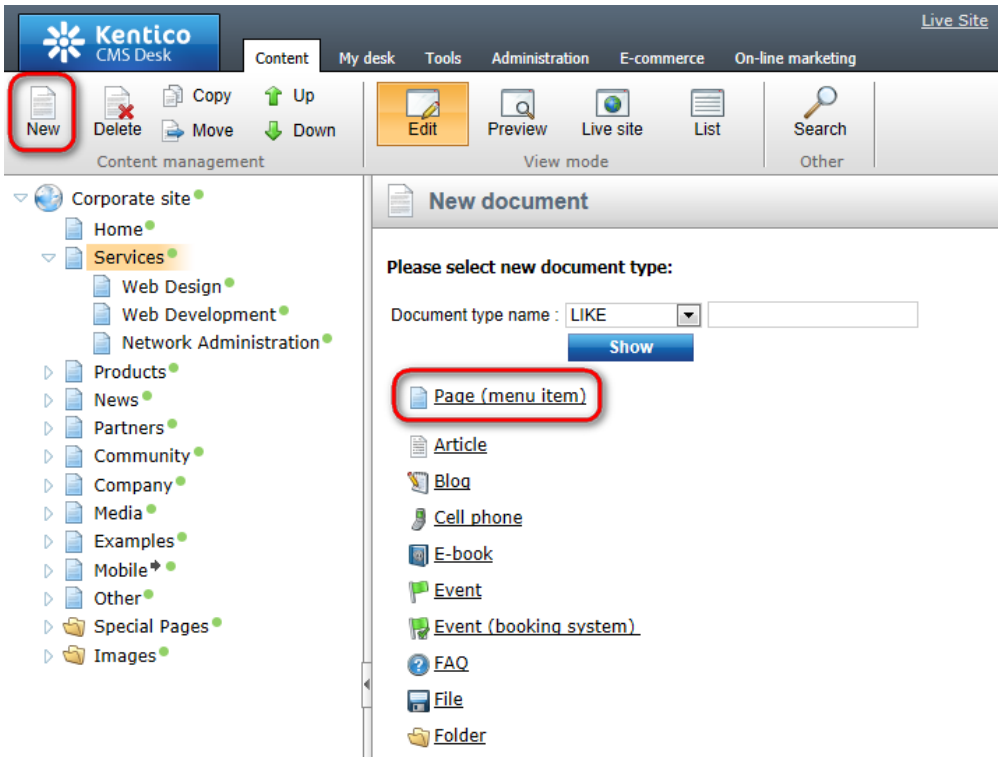
- **New** - the sub-menu offers you the document type of the new document, while only allowed child document types are offered
- **Up** - the sub-menu offers only the **Top** option, which moves the document to the first position in the current tree level
- **Down** - the sub-menu offers only the **Bottom** action, which moves the document to the last position in the current tree level
- **Sort subsection** - the sub-menu offers alphabetical sorting from A to Z and Z to A, and both ascending and descending sorting by date
- **Properties** - the sub-menu offers shortcuts to open the document on particular sub-tabs of the **Properties** tab



4.4.2 Creating a new page

To create a new page, you need to go through the following steps:

1. In the content tree, select the document under which you want to place the new one. Click the **New** button in the main toolbar and choose the **Page (menu item)** document type.

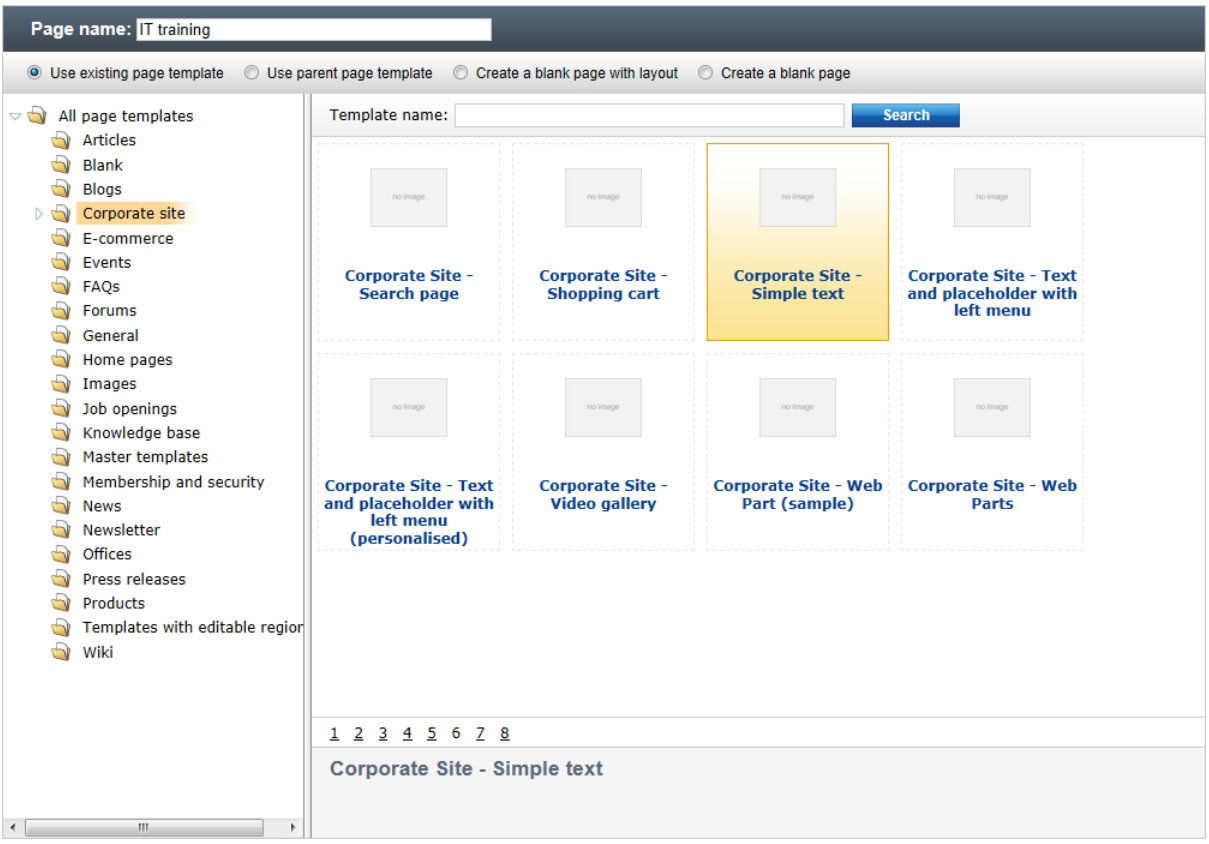


2. Enter the new page's name into the **Page name** field and select the page template that will be used by the page.

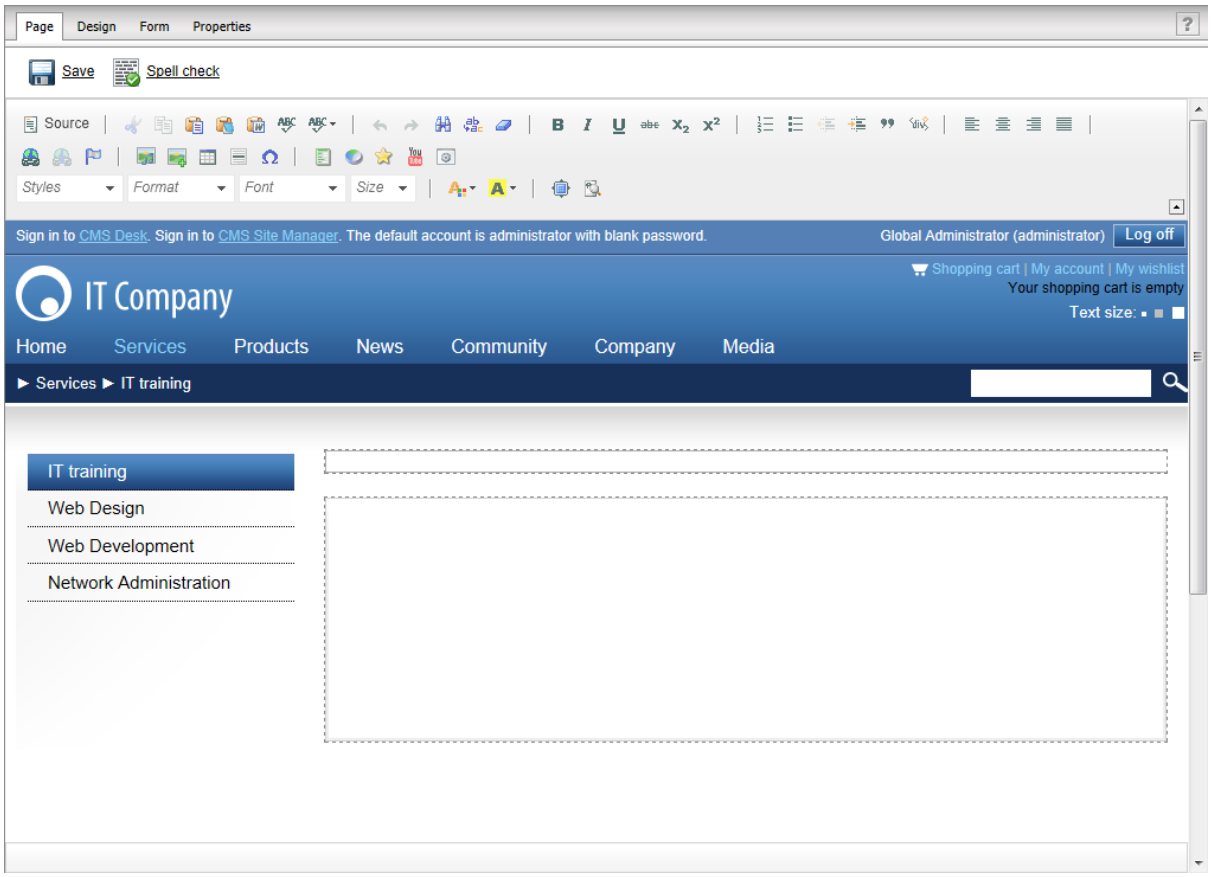
The page template defines the layout and design of the new page. When creating a new page, you can choose from the following options using the check-boxes below the **Page name** field:

- **Use existing page template** - displays the catalog of reusable page templates and allows one to be chosen for the new page.
- **Use parent page template** - if selected, the new page will inherit the template used by its parent document.
- **Create a blank page with layout** - creates a new ad-hoc portal page template using the page layout selected below, and assigns it to the new page. If the **Copy this layout to my page template** box below the layout list is checked, a unique copy of the chosen layout will be generated for the template. Otherwise the template will use the specified shared layout. Any changes made to a shared layout affect all page templates that use it.
- **Create a blank page** - creates a new ad-hoc portal page template for the new page with a single web part zone and no other formatting (the same as the *Simple* page layout). You can learn more in Content tree and page templates.

Choose **Use existing page template**, select the **Corporate Site - Simple text** template from the **Corporate Site** category, enter **IT training** into the **Page name** field and click **Save**.



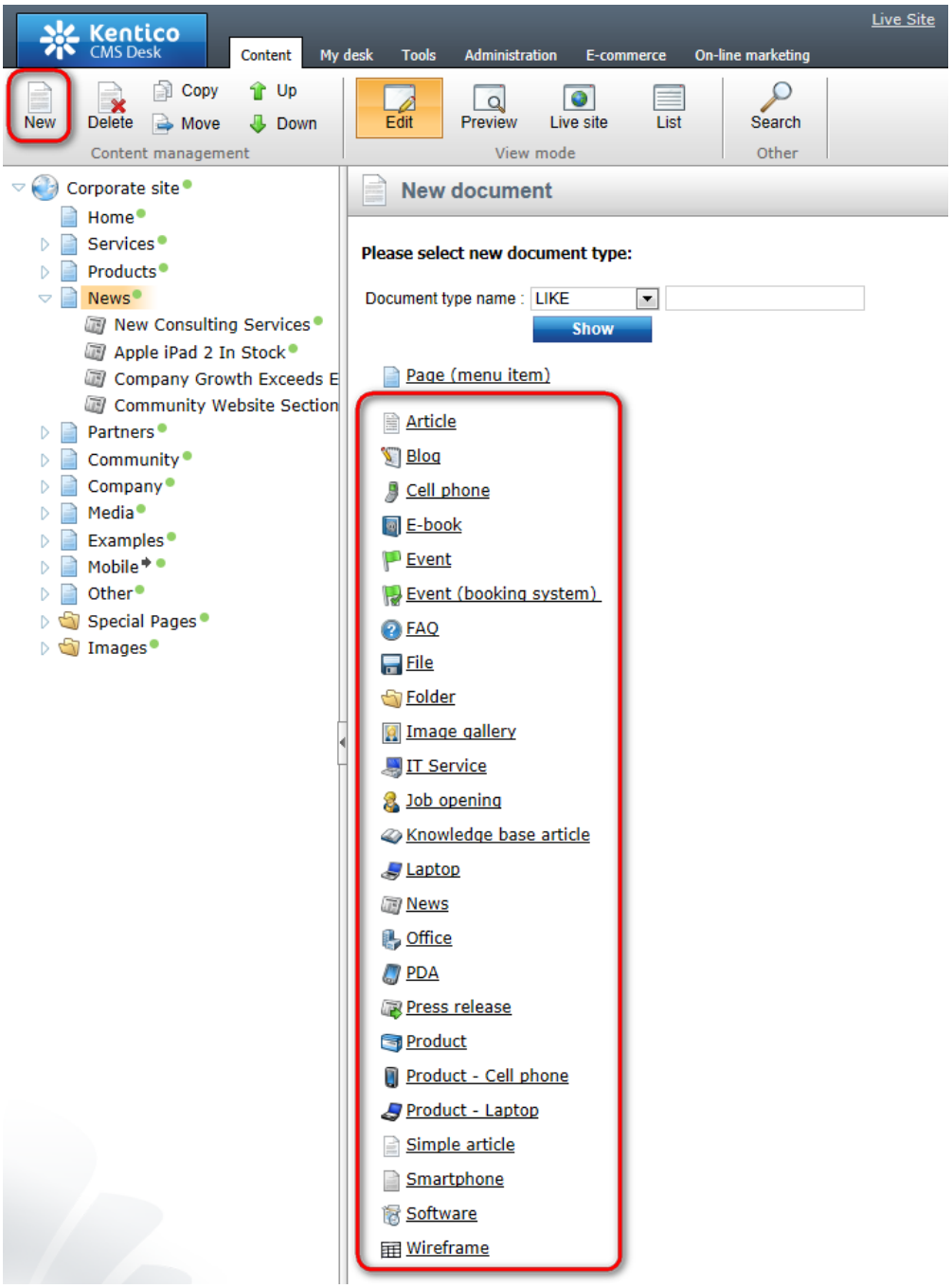
3. The **Page** tab opens and you can enter some content. Click **Save** to confirm the changes.



4.4.3 Creating a new structured document

You can create new documents in **CMS Desk -> Content**.

1. Click the document under which the new item should be placed, click **New** and choose the required document type.



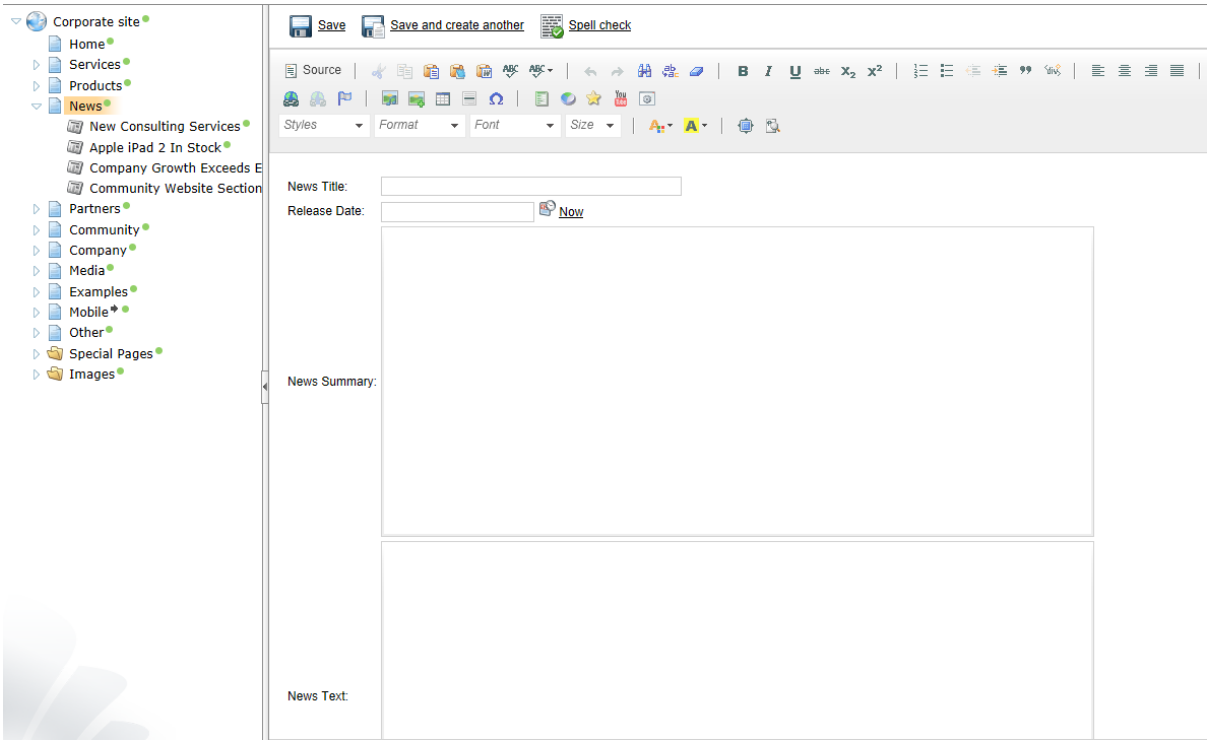
Available document types

The types of documents that can be created under the selected document depend on the type of the selected document. If the required document type is not available, the site administrator needs to add it in Site Manager -> Development -> Document types -> ... edit parent document type ... -> Child types.

2. You are then redirected to the appropriate editing form. For example, if you chose to create a new **News** document in the previous step, you would get an editing form like in the screenshot below. Enter some sample values like the following:

- **News title:** My testing news
- **Released date:** 8/15/2011 (the release date displayed on the website)
- **News summary:** Some news summary.
- **News text:** Some news text.
- **Publish from:** <leave empty for now, it can be used to specify when the document goes live>
- **Publish to:** <leave empty for now, it can be used to specify when the document expires>

Now you can click **Save** to save the document and continue editing. You can also click **Save and create another** to save the news document and immediately create another news document in the same location. The latter option can help save time if you need to create multiple documents of the same type under the same parent.

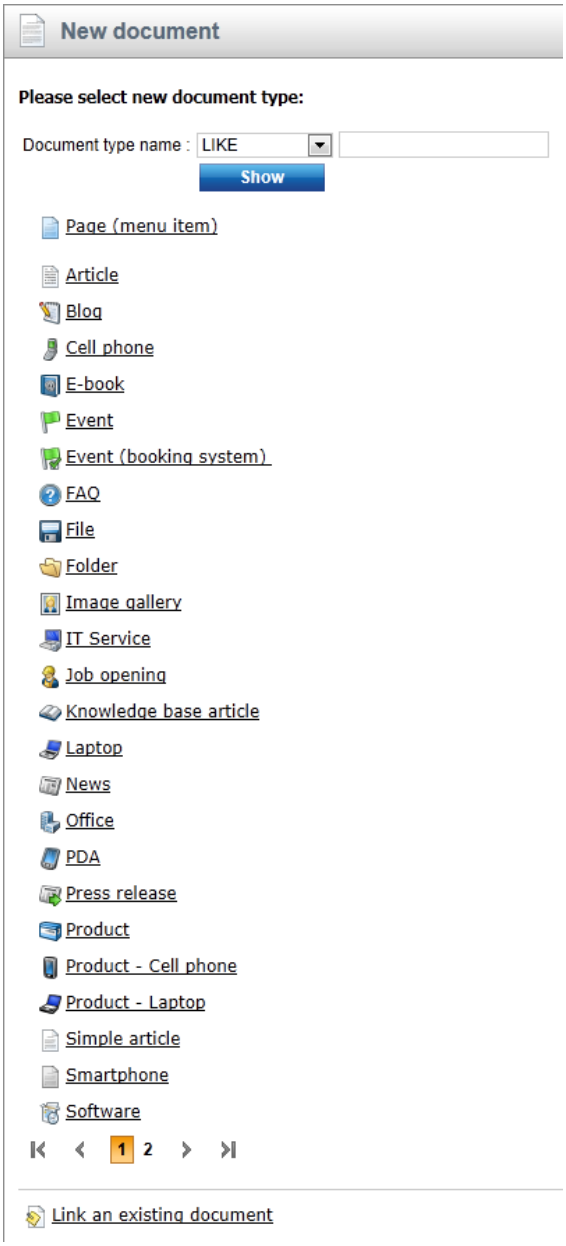


4.4.4 Creating a linked document

A linked document is a "stub" or "shortcut" of some existing document. It allows you to place a single document to multiple places in the content tree instead of creating its copies. Such a document is then displayed in the given part of the website, but when you edit it, you actually update the original document.

This feature is useful if you need to include a document (product) in multiple site sections (product categories), but keep only one instance of the document.

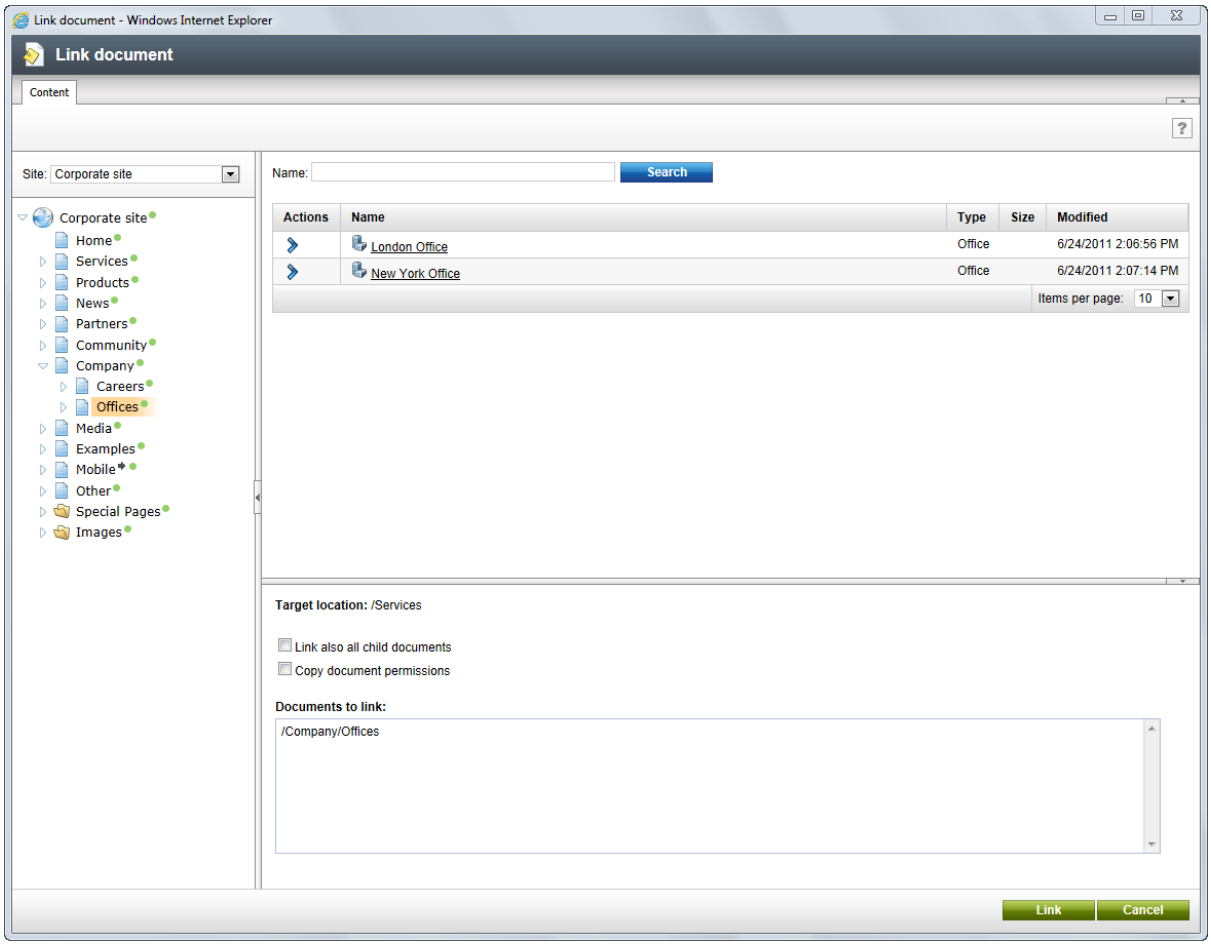
In order to create a linked document, select the document under which you want the linked one to be placed, click **New** and choose **Link an existing document** in the bottom of the screen:



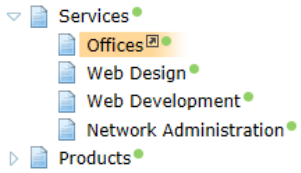
Then, choose the document that should be linked. You have two additional options to configure:

- **Link also all child documents** - this option is displayed only when the selected document has some child documents stored under itself; if enabled, all child documents will be linked along with the selected document
- **Copy document permissions** - if enabled, document-level permissions will be preserved on the linked document; if disabled, the document will inherit permissions from its parent in the target location

Click **Link** to create the new linked document.



Unless the **Site Manager -> Settings -> Content -> Content management -> Display linked icon** option is disabled, the linked document will appear with the icon in the content tree:



You can see a list of all linked documents for the currently selected document in the Properties -> Linked docs dialog.

4.4.5 Drag-and-drop operations with documents

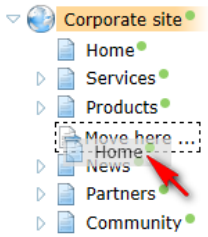
You can perform the following operations with documents quickly by dragging and dropping in the content tree.

- **Move** - just drag and drop the document
- **Copy** - drag and drop while holding CTRL
- **Create linked document** - drag and drop while holding down CTRL+SHIFT

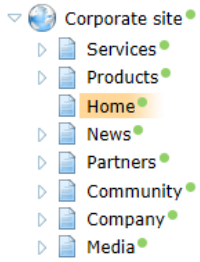
The following steps describe the process of dragging and dropping a document:

1. Select the document that you want to move.

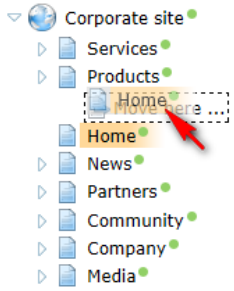
2. Move the mouse to the target location while still holding down the mouse button.



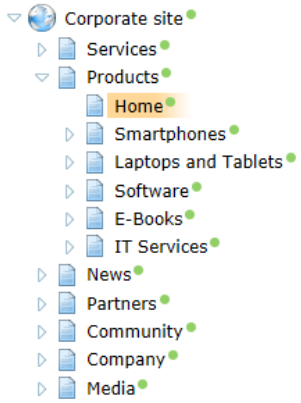
3. Release the mouse button. The document will be moved to the location where you dropped it.



4. If you drag the document a bit to the right ...

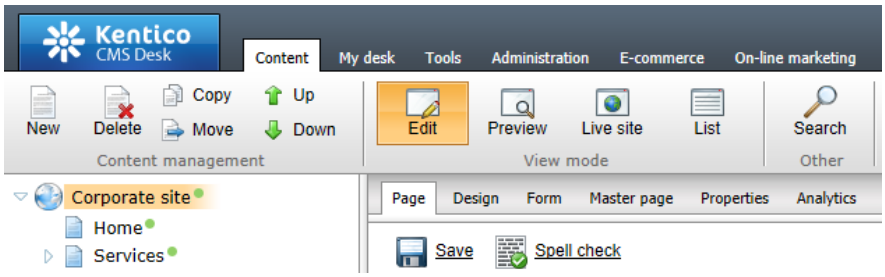


5. ... it will be placed under the document above it.



4.4.6 Previewing documents

When you open the page in **CMS Desk -> Content** section, the default mode is **Edit**. This mode allows you to edit the page's content, design and properties.



The other view modes are:

- **Live site** - this mode shows the page as it currently appears to visitors on the live site.
- **Preview** - this mode also displays the current version of the page, but it may be used even for documents that are not actually published on the live site yet.
- **List** - this mode shows a list of all documents under the currently selected document. This can be useful if there is a large number of documents under a single parent, or if you wish to perform an action for multiple documents simultaneously.

The following points summarize the difference between the **Live site** and **Preview** view modes:

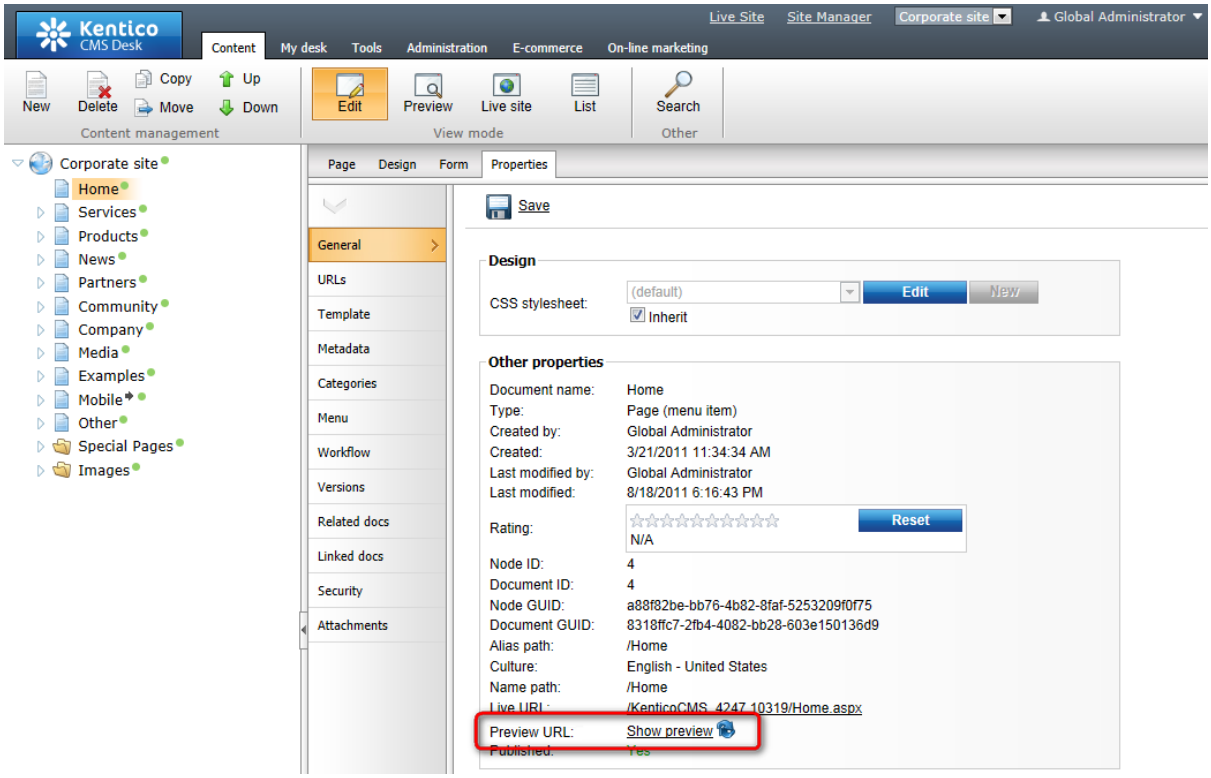
- The **Live site** and **Preview** modes display the same content when a document is published and no

further changes have been made to it since it was published.

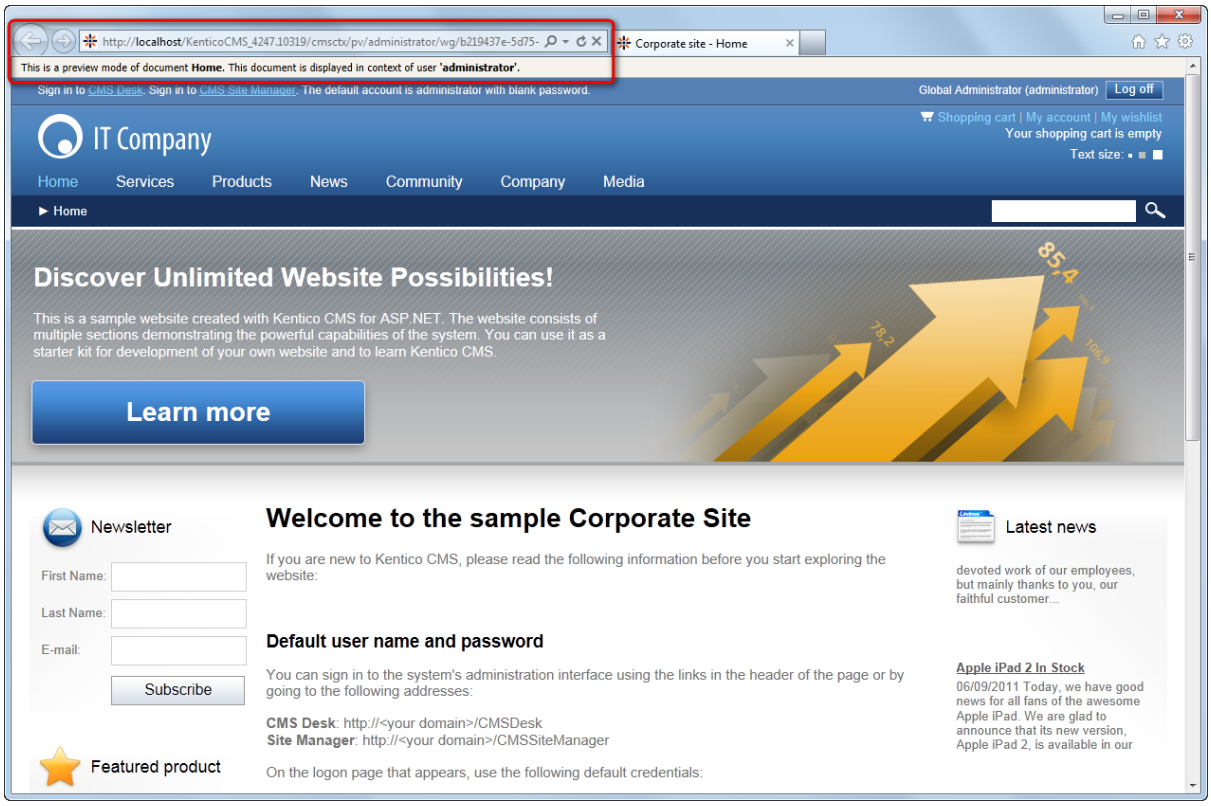
- The **Preview** mode does not use caching, so it may display published changes that are not visible in **Live site** mode yet due to caching.
- If a document does not use workflow and its **Publish from** property is set to a future date and time, the **Live site** mode does not display any content, while the **Preview** mode displays the content that will be published after the specified date and time.
- If the document uses workflow and has not yet reached the **Published** step, the **Live site** mode doesn't display any content, while the **Preview** mode displays the content created in the current workflow step.
- If the document uses workflow, is already in the **Published** workflow step and its workflow cycle has been restarted (i.e. it was switched from the **Published/Archived** workflow step back to the **Edit** step and is going through the workflow cycle again), the **Live site** mode displays the last published version, while the **Preview** mode displays content from the current workflow step.

Preview URLs

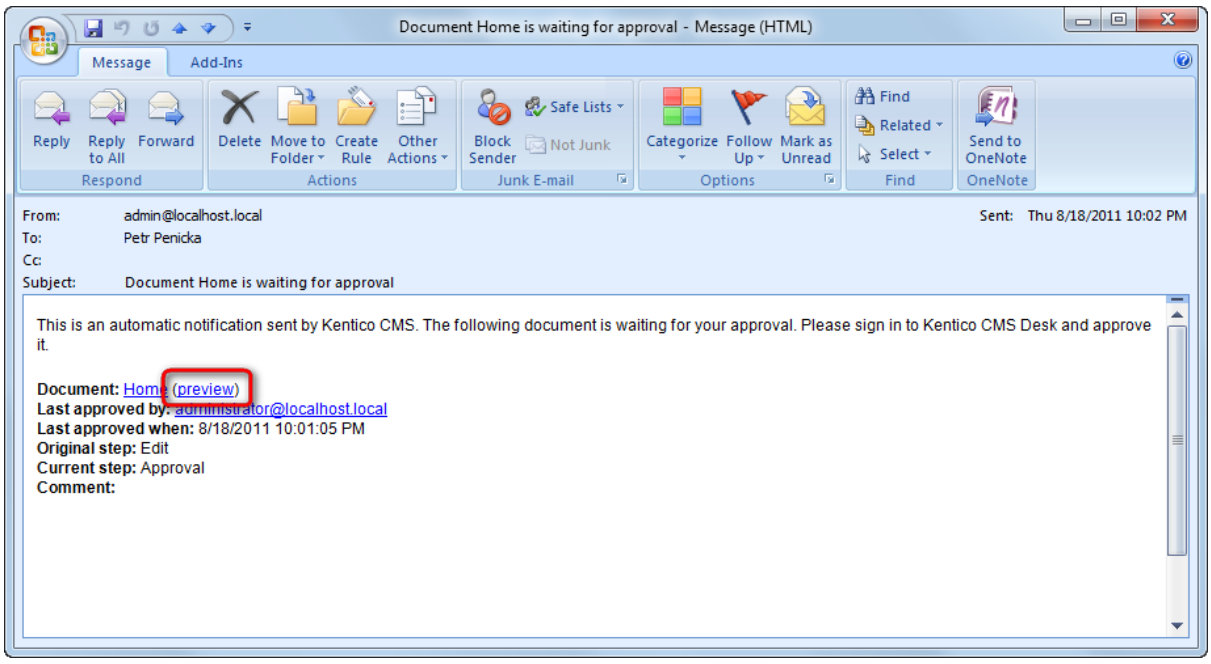
On the **Properties -> General** tab of an edited document, you can find the **Preview URL** property. If you click the **Show preview** link next to it, a new window will be opened, displaying the current document in **Preview** mode outside Kentico CMS user interface.



The document will be displayed on a special page with a dedicated URL. The URL can be sent to a person without access to Kentico CMS user interface to let them view content that is not published yet. At the top of the page, an info line is displayed, informing that the page is displaying a document preview in context of a specific user (the **Show preview** link is always leading to preview in context of the current user). Links on the page are intentionally not working, so the person who accesses the preview URL can only see the preview of the particular document.

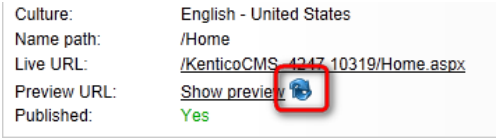


Preview links are also included in workflow notification e-mail templates. In this case, the preview is displayed in context of the user who performed the workflow action about which the e-mail is notifying.



Generating new preview URLs

If you do not want the document's preview to be accessible under a link that you already sent to someone, you can generate a new preview URL for the document. This can be done by clicking on **Generate preview** ( ) next to the **Show preview** link on the document's **Properties -> General** tab. By doing so, the document gets a new preview URL and the original one is no longer functional.



Depending on the **Allow permanent preview links** setting in **Site Manager -> Settings -> Content -> Content management**, new preview URLs can also be generated automatically for documents under workflow when their workflow cycle is restarted (i.e. when they are switched from the **Published/ Archived** workflow step to the **Edit** step). If disabled, new preview URLs are generated in this situation and the original ones are no longer usable. If enabled, original preview links are preserved when the workflow cycle is restarted.

Additional configuration for long URLs in .NET 3.5

URLs in ASP.NET can have a maximal length of 260 characters. This may cause problems with links in previewed documents (e.g. links to attachments of documents which are nested deep in the content tree). If these URLs are long on their own, they may potentially exceed the length limit because the preview URL prefix is pre-pended to in document preview.

If you are running Kentico CMS on .NET 4.0, handling of such URLs is pre-configured in the default *web.config* file, so you should not encounter any problems. If you are running Kentico CMS on .NET 3.5, you need to perform the following steps in order to avoid problems with such URLs:

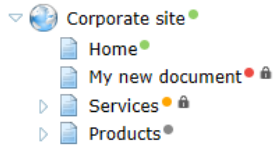
1. You need to install the URL Rewrite Module into the IIS. It can be downloaded from the following location: http://www.iis.net/download/urlrewrite.

2. Add the following rewriting rule into the <*system.webServer*> section of the project's *web.config* file:

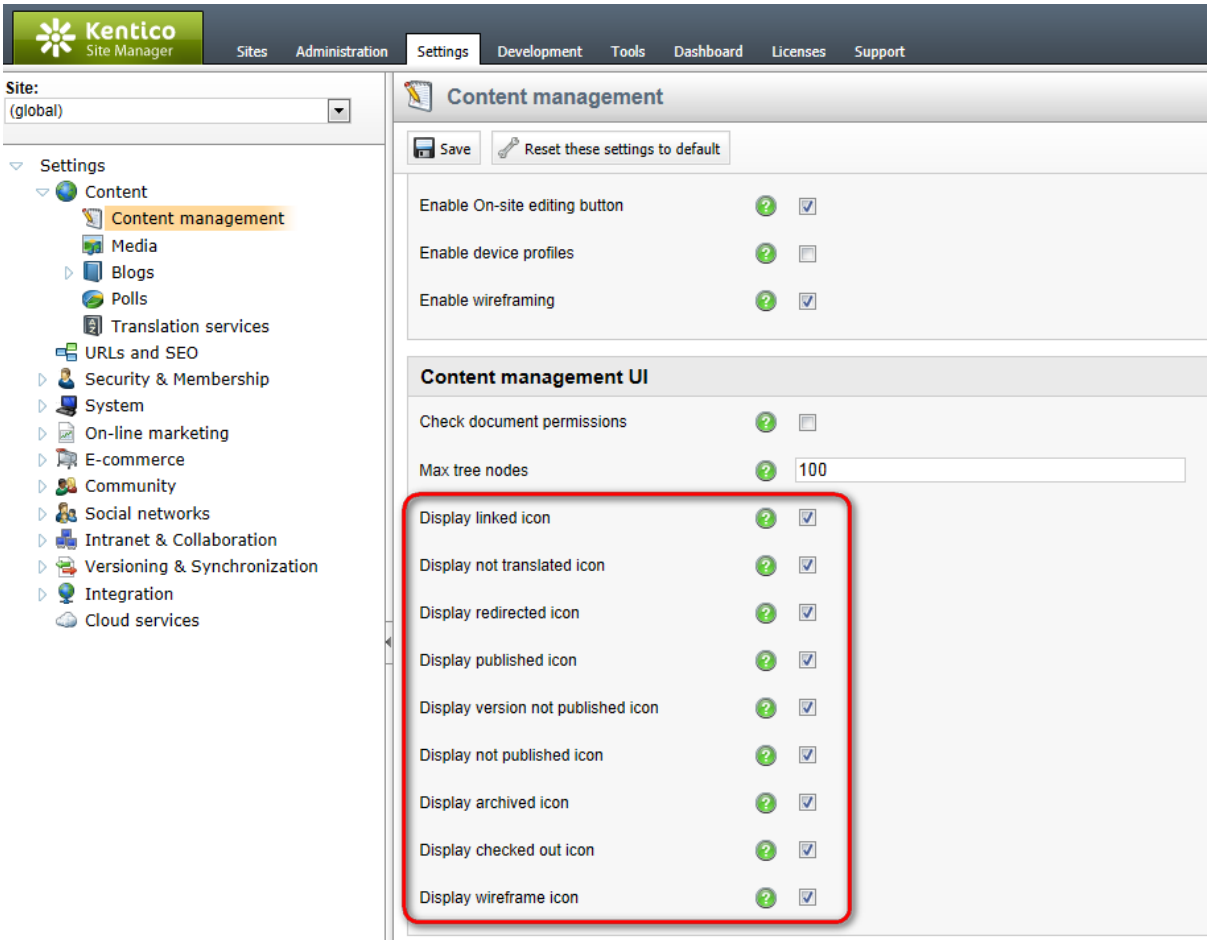
```
<system.webServer>
  <rewrite>
    <rules>
      <rule name="Remove virtual context prefix" stopProcessing="false">
        <match url="^cmsctx/(.+)/-/(.+)$" />
        <action type="Rewrite" url="cmsctx/-/{R:2}" />
      </rule>
    </rules>
  </rewrite>
  ...
</system.webServer>
```

4.4.7 Document status icons

It is possible to have status icons displayed next to documents in the content tree, as you can see in the screenshot below.



These icons give additional information about the document and its current status. You can have them displayed by enabling a set of settings in **Site Manager -> Settings -> Content -> Content management**.



The following table gives information on what each icon indicates and which setting needs to be enabled in order for the particular icon to be displayed.

Icon	Description	Required settings
	This icon is displayed next to linked documents, i.e. documents that only represent a link pointing to another document in the content tree. See the Creating a linked document topic for more details.	Display linked icon
	This icon appears next to documents that are not available in the currently edited culture.	Display not translated icon

	This icon appears next to documents that have a redirection configured in **Properties -> Navigation -> URL redirection**.	Display redirected icon
	This icon indicates that the document is not published on the live site. When a document is not under workflow, it indicates that the **Publish to** property is set to a past date and time. Under workflow, it indicates that the document has not yet been published, i.e. that it has no previously published version.	Display not published icon
	This icon appears next to documents that are scheduled to be published in the future. Without workflow, this happens when the **Publish from** value on their **Form** tab is set to a future date and time. Under workflow, the same applies, while a document must also not have a previously published version (if it has one, the icon is displayed instead).	Display not published icon or Display published icon
	This icon indicates that the document is currently published on the live site. by means of configuration of the **Publish from** and **Publish to** properties on the document's **Form** tab. If the document is under workflow, it also needs to be in the **Published** workflow step for this icon to appear next to it.	Display published icon
	This icon only appears next to documents under workflow that already have a published version and a new version of the document is being created, but is not published yet. In other words, it is displayed next to documents next to documents in any workflow step before the **Published** step.	Display version not published icon
	This icon only appears next to documents under workflow that already have a published version and another version is scheduled to be published. This happens when the new version already is in the **Published** workflow step and the **Publish from** value on its **Form** tab is scheduled to some future date and time.	Display published icon or Display version not published icon
	This icon appears next to documents that have been submitted to a Translation service and are waiting for the translation to be completed.	N/A (this icon is always enabled)
	This icon appears next to documents that are archived. Archived documents are no longer visible on the live site, but are still present in the content tree and can be restored when needed. You can archive a document by clicking the **Archive** button on the **Properties -> Workflow** tab.	Display archived icon
	This icon indicates that the document is currently checked-out, i.e. that it is being edited by another user. You can't edit a document while it is checked out, you have to wait until the user finishes editing and checks the document back in. See Development -> Workflow and versioning -> Content locking for more details.	Display checked out icon
	This icon is displayed next to documents (of any type) that contain a wireframe definition. It is not added to dedicated wireframing documents of the *CMS.Wireframe* type, which can already be identified by their main document type icon. Please see the Development -> Wireframing chapter to learn more.	Display wireframe icon

4.5 On-site editing

4.5.1 On-site editing overview

Kentico CMS also provides a way to work with the content of websites through a convenient on-site editing interface. This allows editors to make changes to page content and manage documents directly while viewing the website from the live site perspective.

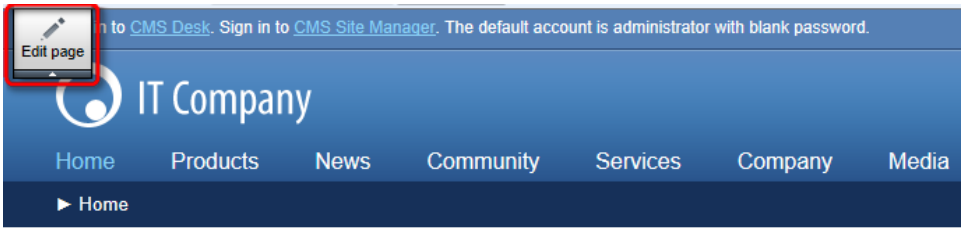
All actions that can be done via on-site editing are also available in CMS Desk. It simply offers a different way to edit the website that may be quicker and easier in many cases (particularly for users who prefer a minimal interface). It is recommended to use CMS Desk for creating the core structure of new sites or when performing complex changes. On-site editing is most suitable for adding or adjusting the content of an existing website that is already running.

To open the title page of a website in on-site editing mode, access it through the following URL:

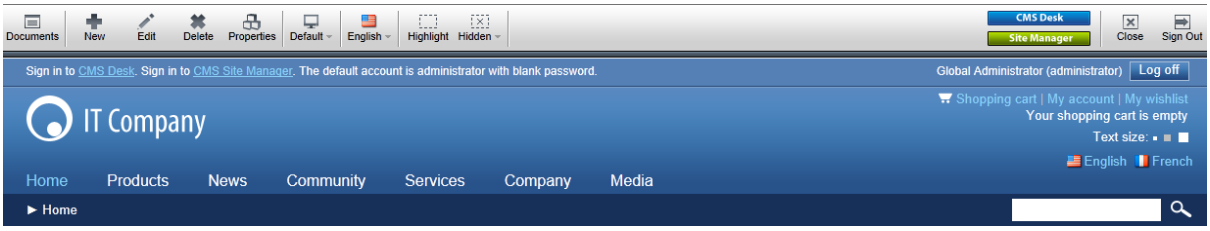
*http://<domain>/<virtual directory>/**cmsedit***

Just like CMS Desk, on-site editing mode requires authentication and a logon screen opens if you are not signed in yet.

Alternatively, editors who are already logged in on the live site may switch to on-site editing mode simply by clicking the **Edit page** button displayed in the corner of the page. This opens the on-site editing interface for the page that is currently being viewed.



On-site editing mode works just like the standard live site, with the addition of a toolbar shown above the page. The content and configuration of pages can be modified through the buttons on this toolbar, along with a special interface displayed when hovering over specific editable sections of the currently viewed page (as described in the Editing content topic).



To move between pages, simply use the site's standard navigation. All other functionality of the website is also available in on-site editing mode. However, there are several important differences from the regular live site that should be considered:

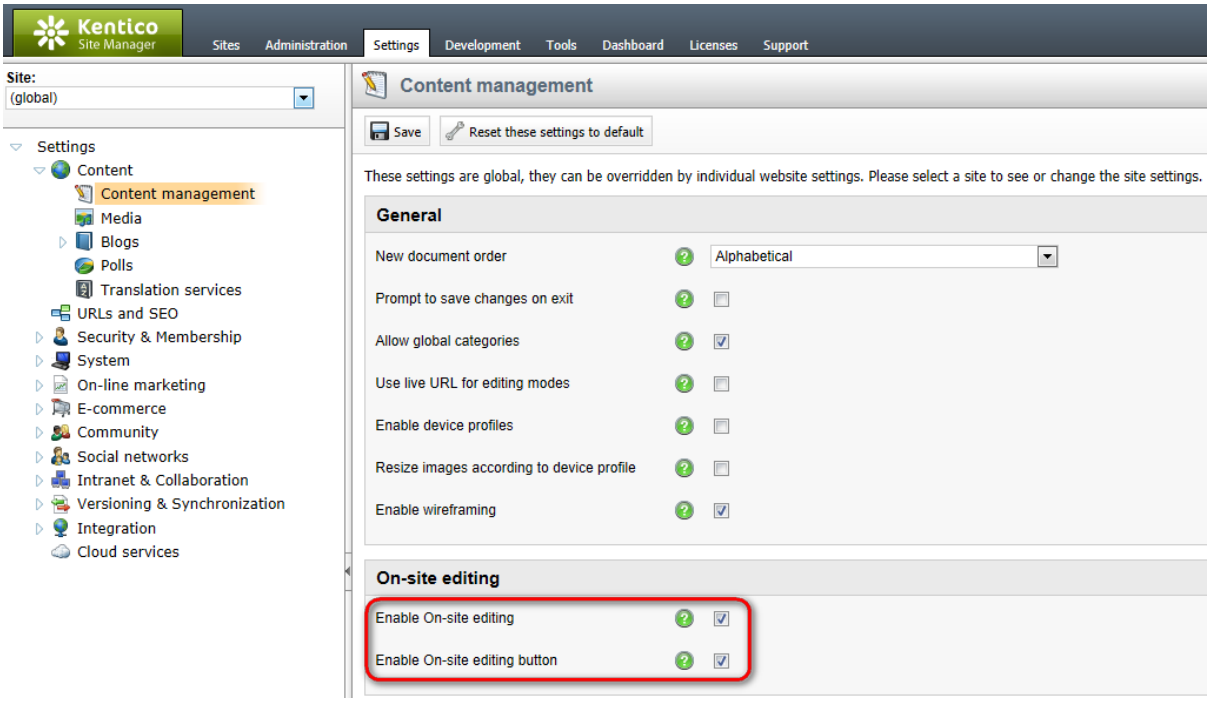
- Pages are always displayed in their current editing state, even if they are not published on the actual live site yet (like in the Preview and Edit view modes of CMS Desk). This makes it possible to view and modify pages before they are presented to the website's regular visitors.

- All forms of page redirection are disabled. This allows editing of pages that would normally be forwarded to a different address on the live site. Pages that are configured for redirection can be identified by the presence of a message box, which also provides a link to the target page.
- If a *Page not found* error occurs while browsing in on-site editing mode, it is handled by a special system page that contains a link to the root of the website.

On-site editing settings

The on-site editing interface is not a required part of the system and can be disabled globally or for specific sites. System administrators may do so by configuring the appropriate settings in **Site Manager -> Settings -> Content -> Content management**:

- **Enable On-site editing** - on-site editing is only available for websites that have this field checked.
- **Enable On-site editing button** - determines whether the *Edit page* button should be displayed in the corner of pages on the live site for users who are authorized as editors. If you do not wish to use this button, users can alternatively access on-site editing mode through its dedicated URL, or by following a link provided by the Edit document link or Admin actions web part.

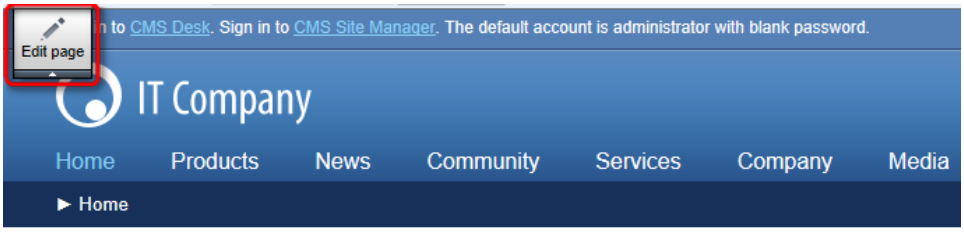


4.5.2 Editing content

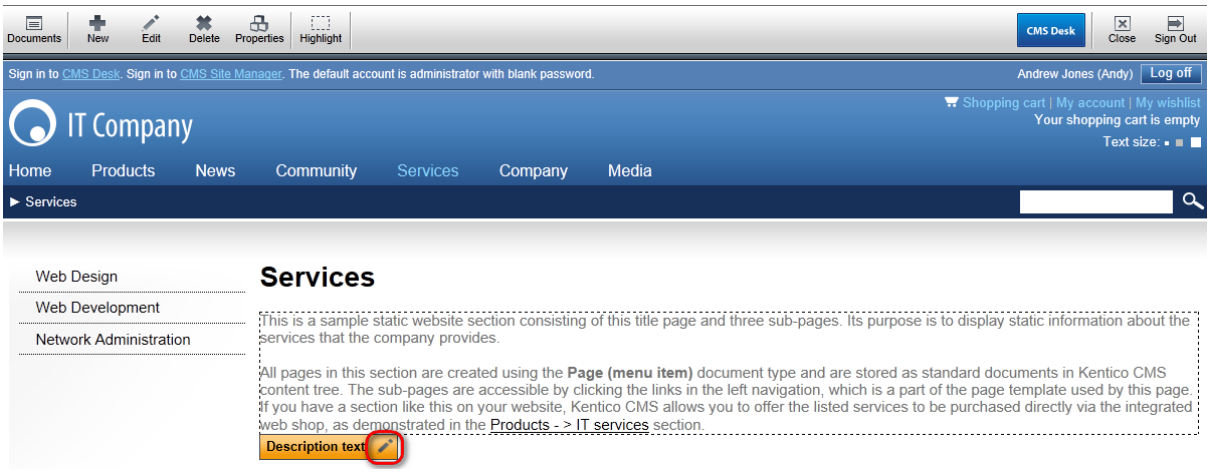
The following steps demonstrate how to edit various types of page content through the on-site editing interface:

1. Open the sample Corporate site and log on to the live website as a user who is an editor. For example, you can use the default editor account — user name *Andy* with a blank password. Do not sign in to CMS Desk, simply enter the credentials into the logon form at the top right of the site's header.

2. Click the **Edit page** button in the corner of the page to open on-site editing mode. The system displays the button because the current user is authorized as an editor.



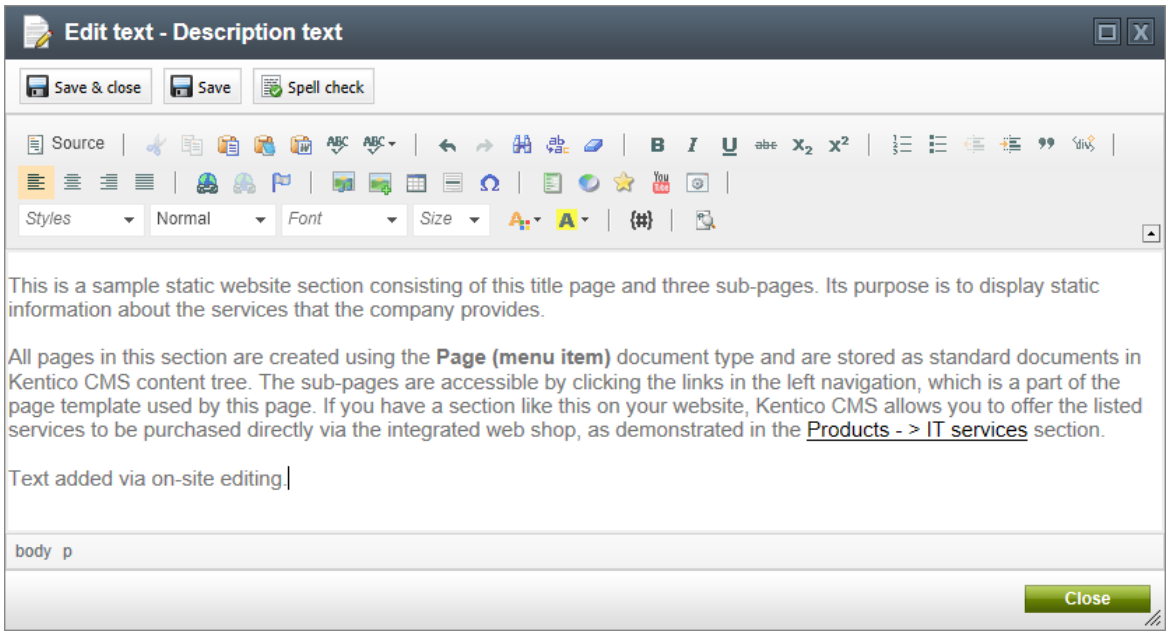
3. Navigate to the **Services** page through the website's main menu. Try moving the mouse over the sections of text displayed on the page. In on-site editing mode, hovering over editable text regions causes them to be highlighted by a dotted outline, with a box showing the title of the given region. Do this for the main text area and then click edit (✎) in the title box.



Tip

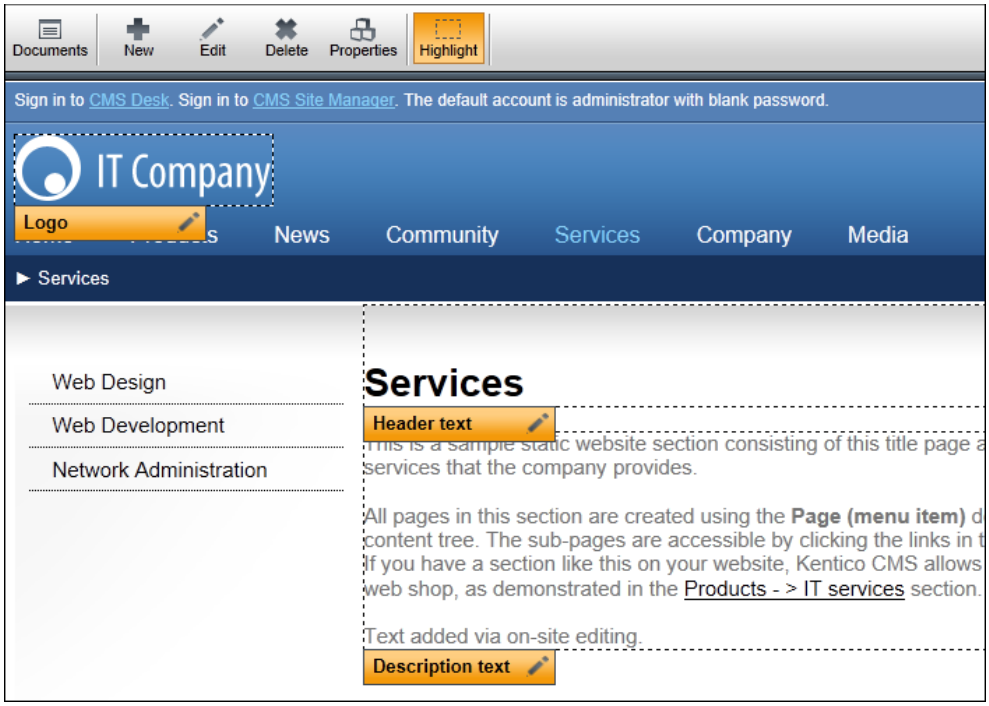
If you encounter a situation where the highlight feature is blocking a part of the page that you wish to click, you can bypass it by holding down the **CTRL** key while moving your mouse. This way, you can reach the required location without triggering the highlight.

4. A new dialog opens, where the content of the region can be edited as needed using the WYSIWYG editor. Add some new text below last paragraph and confirm the change by clicking **Save & close**.



This automatically refreshes the page and displays the new content.

5. Continue by clicking the **Highlight** action on the on-site editing toolbar above the page. When this button is toggled on, all editable items placed on the page are highlighted at the same time. This way you can easily find all editable text regions and editable images and modify them as necessary.



To change an editable image, click the edit (✎) action next to its title and then select a different image in the displayed dialog. Please note that the company logo on the Corporate site is not actually a part of the *Services* page. It is defined in the website's root document, which provides the header content

shared by most pages. As a result, changing the image here also changes it for the entire website.

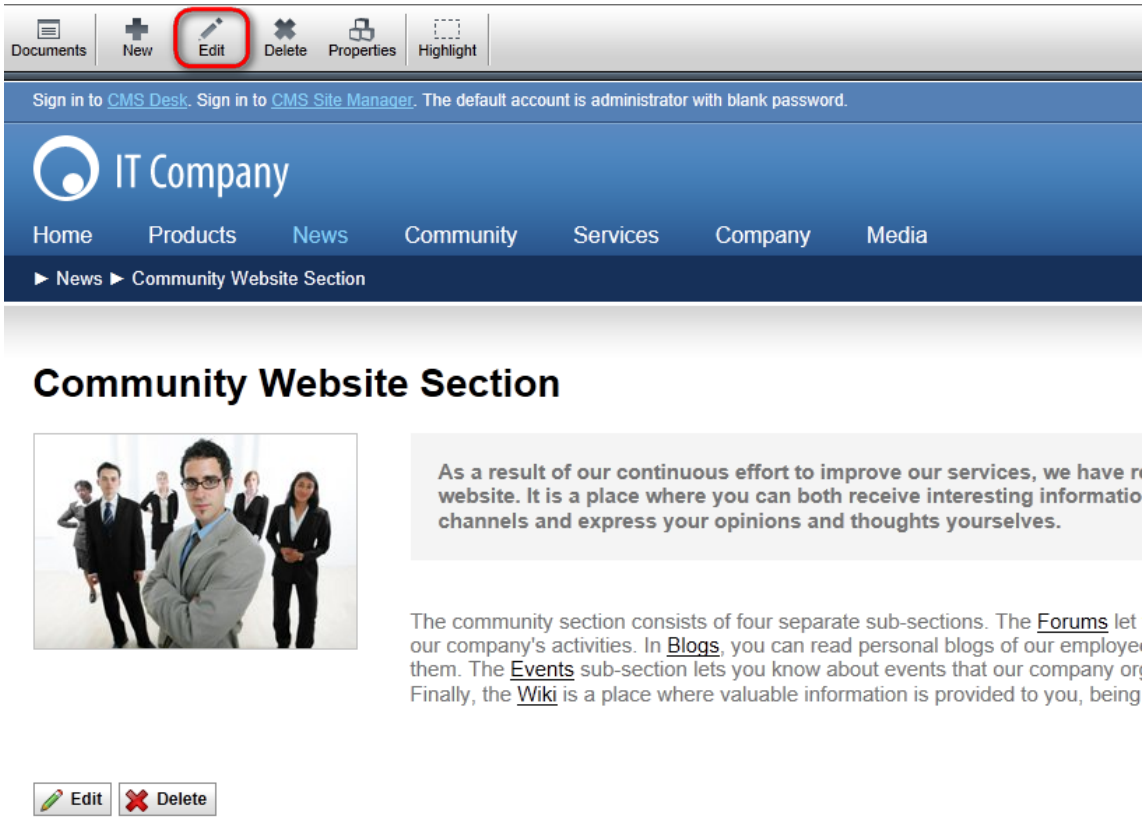
On-site editing with Design permissions

If the current user is also authorized to design the website, the system additionally highlights individual Web parts placed on the page. In this case, on-site editing also offers the option to configure (⚙) the properties of these web parts.

Editing structured documents

Now we will edit a page that loads its content from the fields of a structured document.

1. Go to the **News** section of the website and view the first news article in the list, titled **Community Website Section**. Once the page is loaded, click the **Edit** action on the on-site editing toolbar.

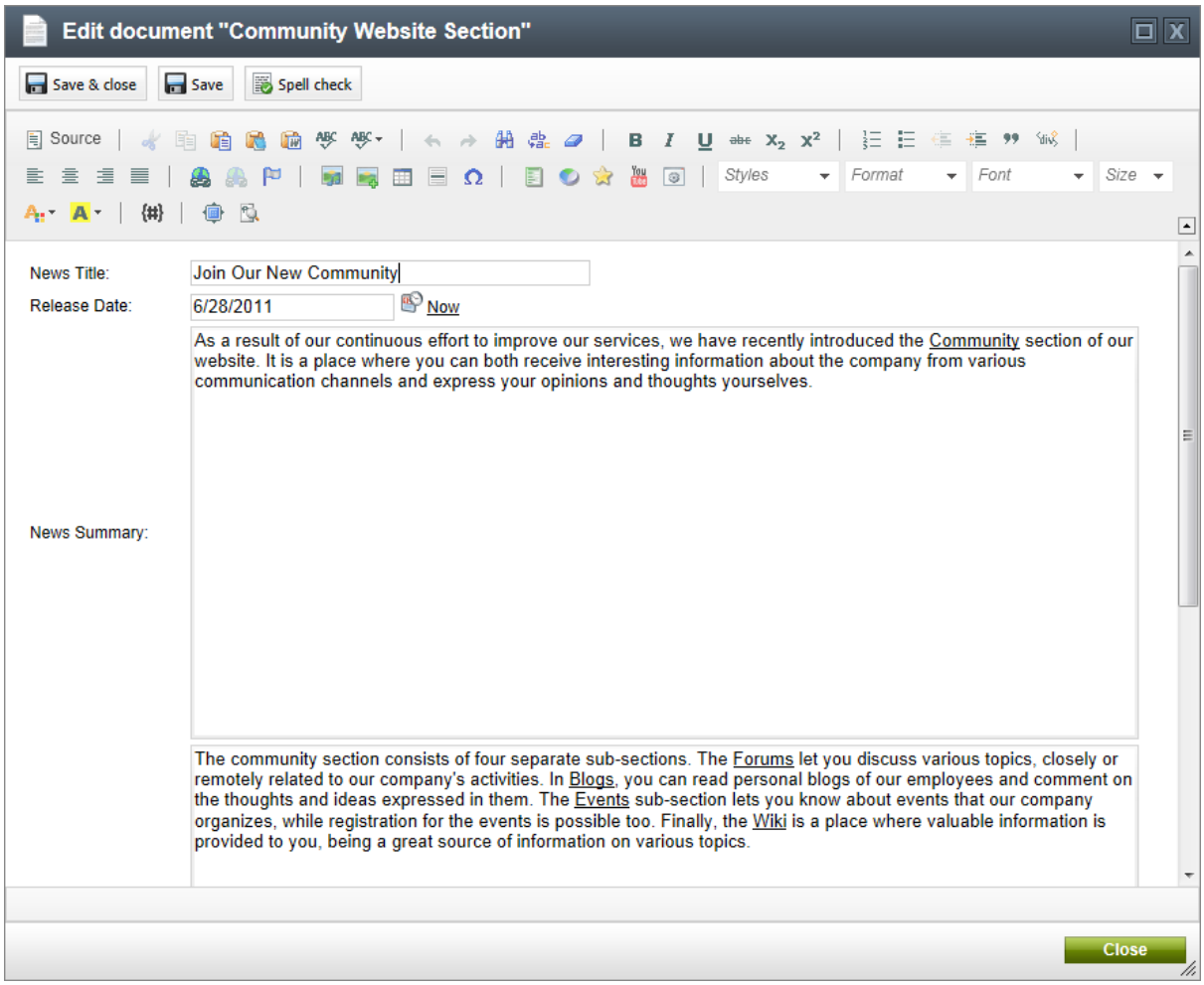


Edit mode buttons

In many cases, document lists are configured to provide edit mode buttons, which are also visible in on-site editing mode. Such buttons are shown both on the list page next to individual items and when viewing the details of a specific document.

> The **Edit** button works the same way as the corresponding action on the main toolbar.

2. An editing dialog opens, where you can work with the values of the given document's fields. This is the equivalent of the *Form* tab in CMS Desk. You can make any required changes to the news document. For example, set the **News Title** field to *Join Our New Community*. Confirm the change by clicking **Save & close**.



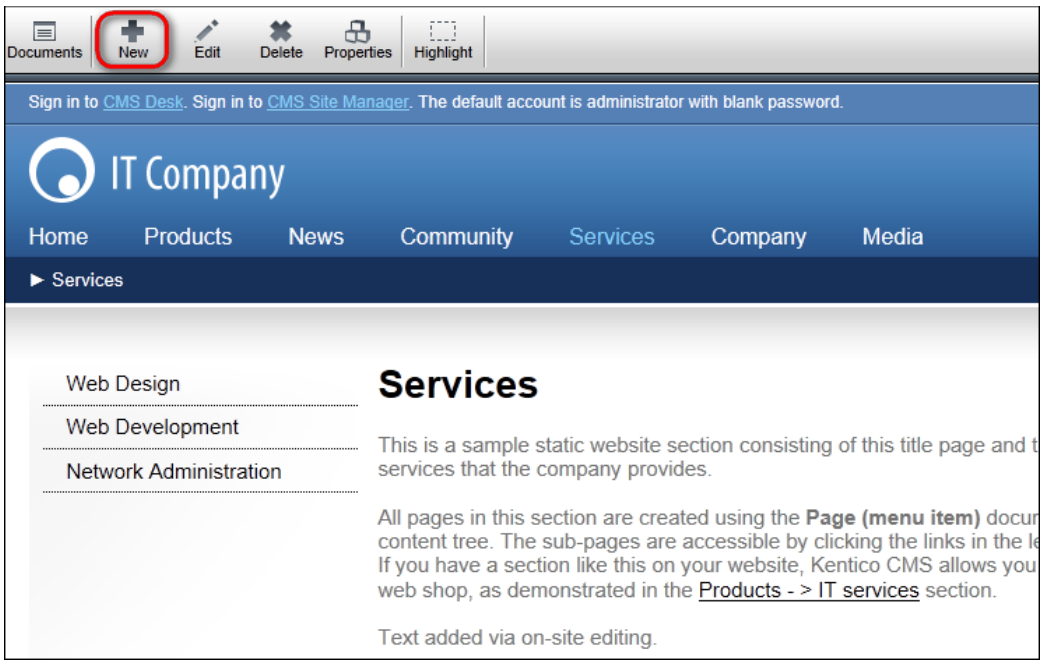
The new title is shown on the detail page of the news article, as well as in the main list of items on the **News** page.

4.5.3 Adding new pages

This topic contains an example describing the typical steps that need to be performed when creating new pages (documents) through the on-site editing interface.

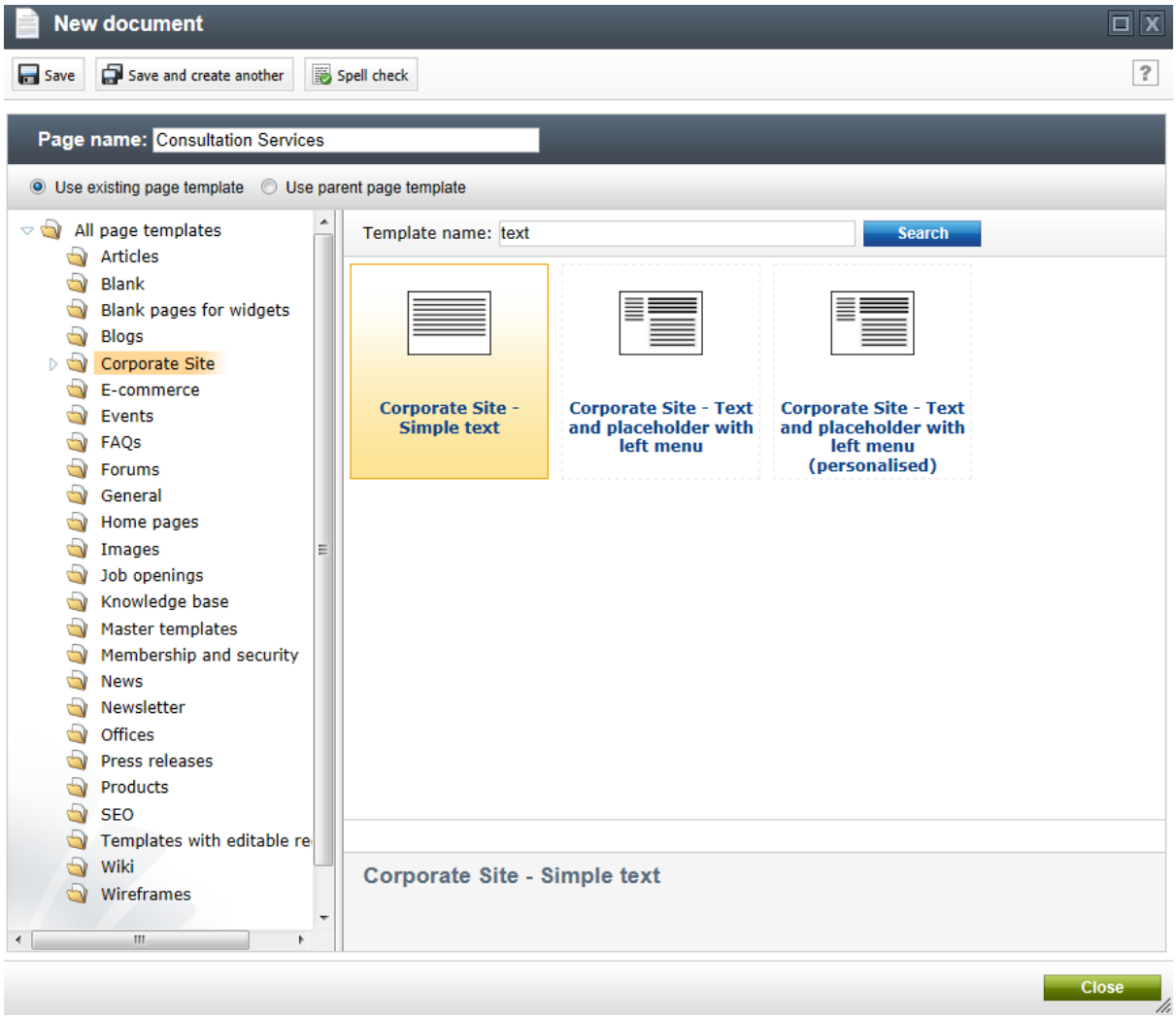
1. Open the sample Corporate site, log on to the live website as an editor (for example user name *Andy* with a blank password) and switch to on-site editing mode.

2. Select the **Services** page in the website's main navigation menu and then click **New** on the on-site editing toolbar. This action creates a child document under the page that is currently being viewed.



3. In the displayed document creation dialog, choose the **Page (menu item)** document type. You must then select a template for the new the page. Click the **Corporate Site** category in the tree, enter the word *text* into the **Template name** filter and then choose the **Corporate Site - Simple text** template.

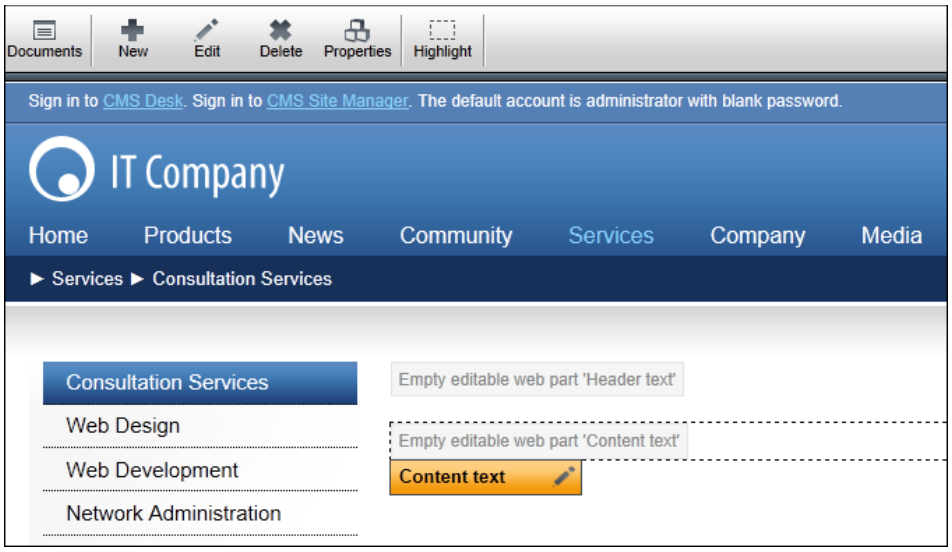
4. Type *Consultation Services* into the **Page name** field and click **Save**.



The new page is added under the *Services* section of the website. Visitors can now access it through the site's navigation menus.

5. The content of new pages is often empty. In this example, the page template contains two editable regions without any text. Empty editable regions are represented by placeholder boxes in on-site editing mode, so you can easily find them on the page and edit their content.

Hover over the placeholders and use the edit action to enter some text into the page's header and content areas.



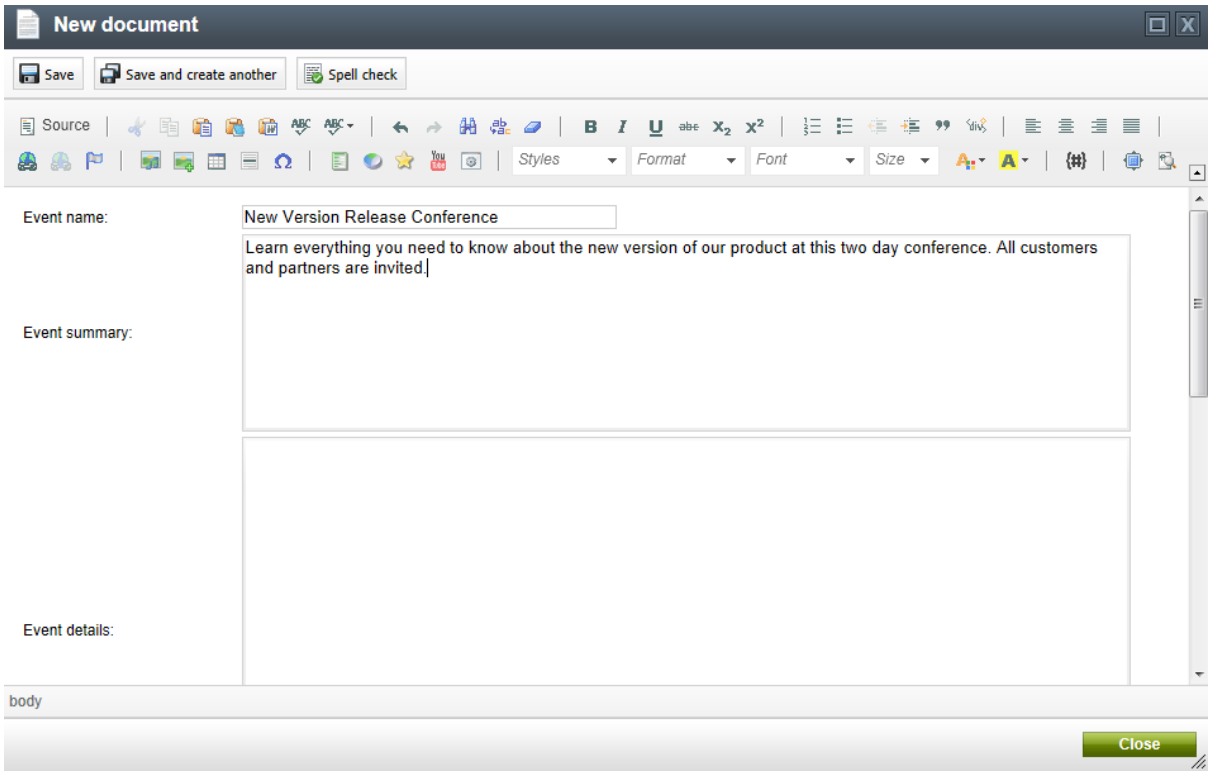
Creating structured documents

1. Navigate to the **Community -> Events** page through the main menu. You can add structured documents to the website in the same way as standard pages.

2. Click the **New** action on the toolbar and select the **Event (booking system)** document type. Alternatively, the same could be achieved using the **New event** button provided in the list of upcoming events on the right side of the page.

3. Instead of selecting a page template, you need to specify values for the document's fields in the creation dialog. For example, you can use the following:

- **Event name**: New Version Release Conference
- **Event summary**: Learn everything you need to know about the new version of our product at this two day conference. All customers and partners are invited.
- **Start date** / **End date**: Use the calendar () to enter a date in the future



Confirm the creation of the document by clicking **Save**. The data displayed on the *Events* page is loaded from the child event documents. As a result, you can look up the specified date in the calendar and the new event is shown. It is also displayed in the upcoming event list.

Events in Calendar

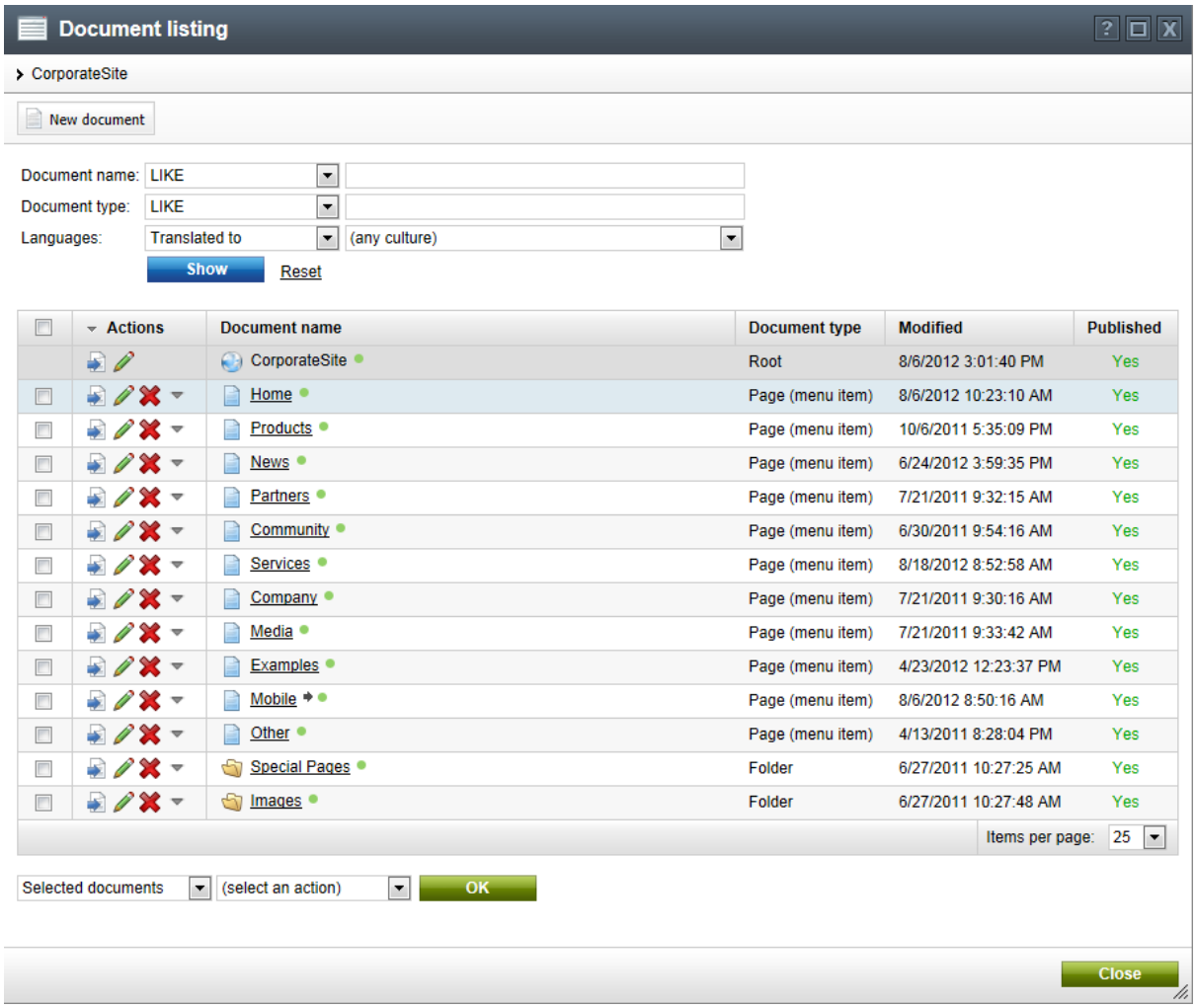
June 2012

Sun	Mon	Tue	Wed	Thu	Fri	Sat
27 No event	28 No event	29 No event	30 No event	31 No event	1 No event	2 No event
3 No event	4 No event	5 No event	6 No event	7 No event	8 No event	9 No event
10 No event	11 No event	12 No event	13 No event	14 No event	15 No event	16 No event
17 No event	18 No event	19 No event	20 No event	21 Partners Training	22 Partners Training	23 Open Day
24 No event	25 No event	26 No event	27 No event	28 New Version Release Conference	29 New Version Release Conference	30 No event
1 No event	2 No event	3 No event	4 No event	5 No event	6 No event	7 No event

4.5.4 Document management

In addition to basic editing of page content, on-site editing may also be used to manage the document structure of the website. This is done via a separate dialog that can be accessed by clicking the **Documents** button on the on-site editing toolbar. The overall structure of the website may not always be obvious from the live site point of view and it can be difficult or even impossible to access some documents using only the natural on-site navigation. For these reasons, the dialog offers an alternative way to explore the website based on document lists.

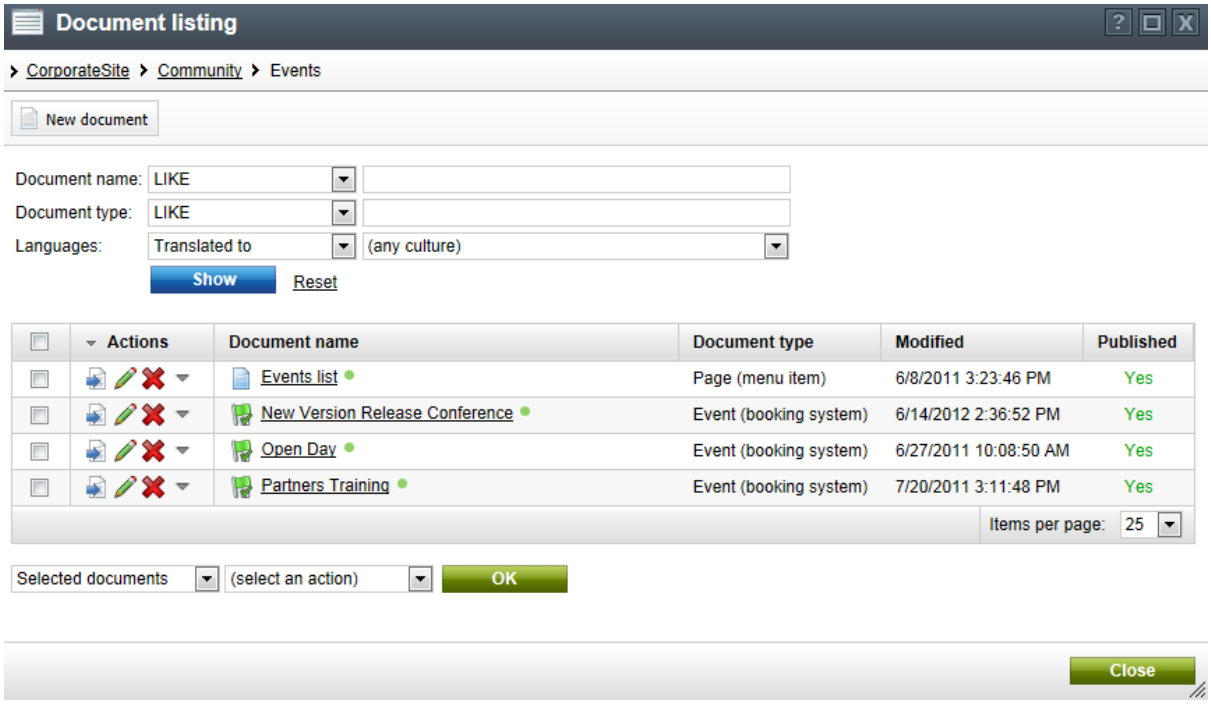
When opened, the list initially displays all sibling documents of the page that is currently being viewed in on-site editing mode (i.e. the documents on the same content level that share the same parent).



Editing the website root

The **Document listing** dialog also provides an easy way to access the *root document* of the website. You can find it as a special item at the top of the list on the first level of the site's content (it is marked by the document type icon).

You can navigate deeper into the site structure by clicking on the name of any listed document, which displays the child documents of the given item. To return to a previous content level, click on one of the parent documents shown in the breadcrumb navigation at the top of the dialog. This way, you can quickly access any section of the website.



The following actions are available for individual documents in the list:

- **View** - navigates to the page represented by the given document on the live site (in on-site editing mode).
- **Edit** - clicking this action opens a dialog where the fields of the given document can be edited.
- **Delete** - removes the given document from the website.
- The action opens a context menu with further options that may be used to change the position of the document on the given level of content:
  - **Top**
  - **Up**
  - **Down**
  - **Bottom**

Additionally, the dialog may be used to perform various types of document management operations via the two drop-downs below the list. First, you need to select one of the following options:

- **Selected documents** - the action is performed only for the documents selected through the checkboxes ( ).
- **All documents** - the action is done for all documents listed on the given content level.

Then you can choose one of the available actions:

- **Move** - moves the documents to a new location specified through a separate dialog.
- **Copy** - creates copies of the selected documents at the specified location.
- **Link** - creates linked versions of the selected documents at the specified location.

- **Delete** - removes the documents from the site.
- **Translate** - submits the selected documents to a Translation service specified via a separate dialog.
- **Publish/Archive** - may be used to publish or archive documents under Workflow.

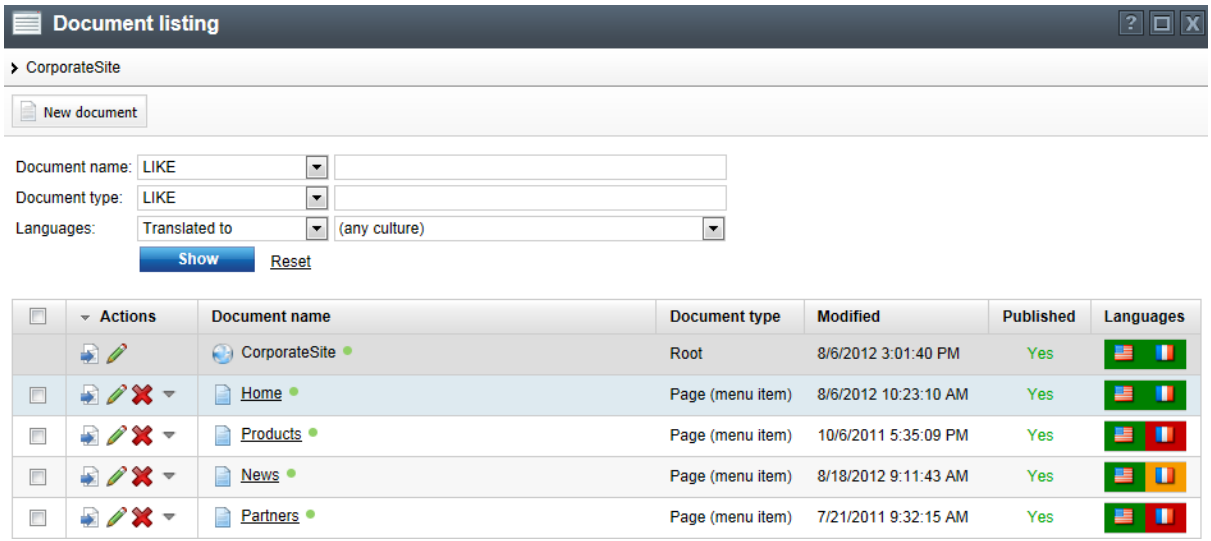
Click **OK** to perform the selected action.

You can also add a new document to the currently selected location by clicking the **New document** button above the list. This opens a creation dialog that works the same way as described in the Adding new pages topic.

Managing multilingual websites

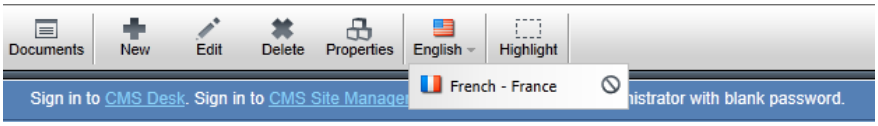
If the website's content is available in more than one language, the document list also provides a **Languages** column with additional data. It displays the status of the documents in all languages assigned to the website. The colors surrounding the flags of individual languages indicate the following possible language statuses:

- **Green - Translated** - the document is available in the given language and up-to-date. The actual language of the document's content has no effect on the status, the system only checks whether the language version exists.
- **Orange - Outdated** - the language version exists for the document, but is outdated. The system considers language versions to be outdated if the default language version of the document has been modified (or published when using workflow) more recently.
- **Red - Not available** - the document does not exist in the given language.



Clicking a **Translated** or **Outdated** flag redirects you to the matching language version of the document on the live site (in on-site editing mode). If you click a **Not available** flag, a dialog opens instead where you can create a new version of the document in the given language.

You can also switch between languages and create translated page versions directly while viewing the page content using the selector on the main toolbar.



Refer to the Multilingual content chapter to learn more about creating websites in multiple languages.

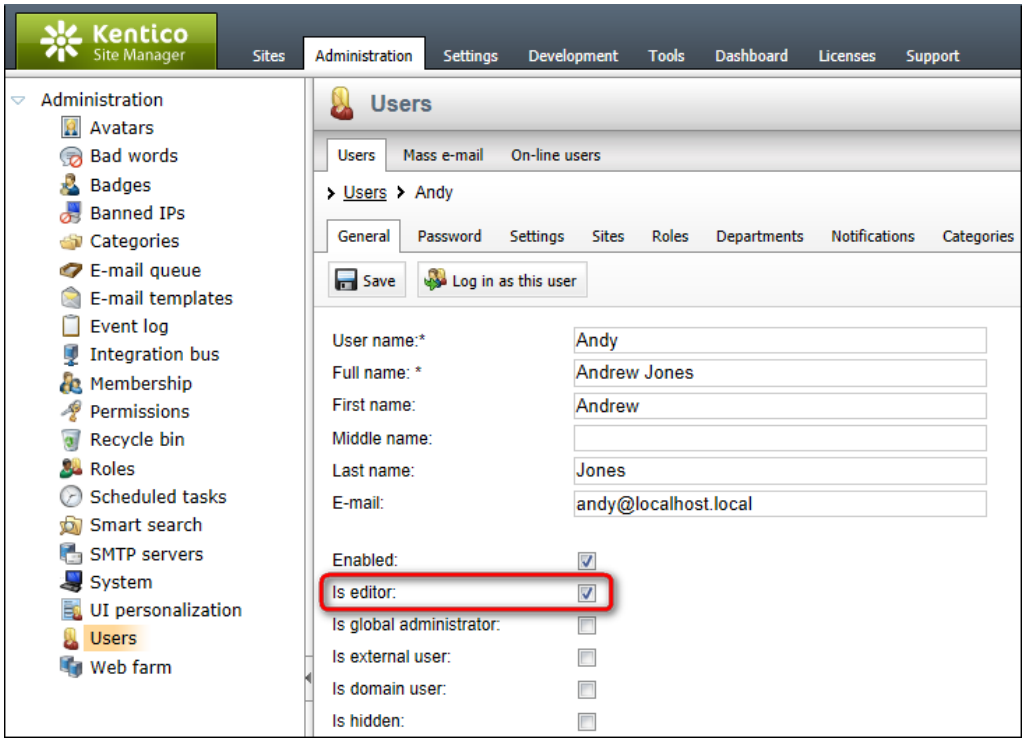
4.5.5 Security

This topic describes the permissions and other security requirements related to the on-site editing interface. The information here is primarily intended to help system administrators correctly set up user accounts for on-site editing.

Editor prerequisite

Like with CMS Desk, users need to be designated as editors to access on-site editing mode.

1. Go to **Site Manager** (or **CMS Desk**) **-> Administration -> Users**.
2. Edit a user on the **General** tab.
3. Check the **Is editor** box.
4. Click **Save**.

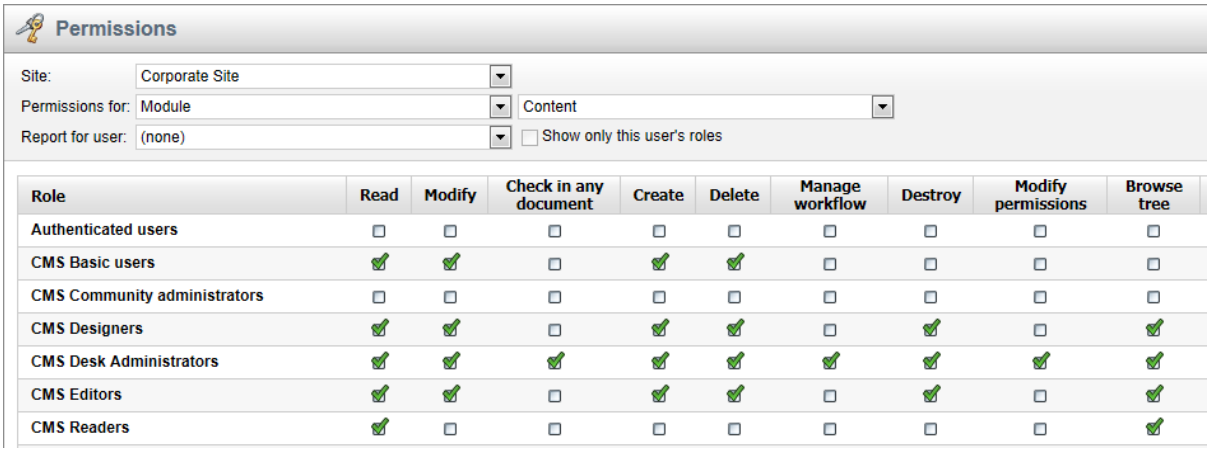


The system now recognizes the given user as an editor.

Permissions

There are several module permissions that affect on-site editing. You can configure these permissions for specific roles in **Site Manager / CMS Desk -> Administration -> Permissions**.

Module	Permission	Description
Content	Read	Only users who belong to roles that have this permission are able to enter on-site editing mode.
	Modify	Allows users to edit document fields and the content of editable regions.
	Create	Allows users to create new pages (documents).
	Delete	Allows users to delete existing pages from the website.
	Browse tree	Users need this permission to perform document management actions, such as creating and deleting pages or viewing the document list. It is not required for basic work with editable region content.
Design	Design website	Users need this permission to configure the properties of web parts through the on-site editing interface. The system only highlights editable regions and images for users without this permission.



Additionally, all permissions configured for document types or individual documents also apply in on-site editing mode. You can find more information in the Development -> Membership, permissions and security -> Permissions chapter.

UI personalization

You can also customize the visibility of individual elements on the on-site editing toolbar and in the related dialogs for particular roles via UI personalization.

1. Go to **Site Manager** / **CMS Desk -> Administration -> UI personalization**.
2. Select the appropriate **Site** and **Role**.
3. Choose the *CMS On-site editing* **Module**.
4. Configure the checkboxes as required. The available options match individual buttons on the main on-site editing toolbar.

Members of the selected role can now see only the specified UI elements.

Note

All of the UI personalization options set for the *Content* **Module** also apply (e.g. various document property tabs, the web part properties dialog etc).

You can find additional details in the Development -> Membership, permissions and security -> UI personalization chapter.

4.6 WYSIWYG editor

4.6.1 Overview

Kentico CMS comes with a built-in WYSIWYG editor. It is based on CKeditor, which is one of the best browser-based editors available on the market.

Where the editor is available

You can come across the WYSIWYG editor in many parts of the system. However, its main use is related to the **Editable text** web part, which enables content editors to enter page content via the **Page** tab.

Another example of the use of the WYSIWYG editor is the **HTML area** form control that can be displayed e.g. on the **Form** tab of a document, as a part of **Form** or in web part properties dialogs of certain web parts (e.g. **Static HTML** or **Silverlight application**).

WYSIWYG editor toolbar

The default toolbar looks like this:

As you can see, it provides functionality similar to Microsoft Word. Still, there are several icons that may require additional explanation:

	Paste as plain text - this command pastes the content of your clipboard while cleaning out all formatting.
	Paste from Word - this command allows you to clean-up content pasted from Microsoft Word. It cleans up the HTML code so that it does not contain extra code and fits your website design. You can find more details in the Copy & Paste from Microsoft Word topic.
	Insert/Edit link - creates a link from the selected text or inserts a link into the text. Please refer to the WYSIWYG editor -> Insert link subchapter for more details.
	Insert/Edit image or media - inserts an image or other media into the text. Please refer to the Insert image or media topic for more details.
	Quickly insert image or media - inserts an image, video, document or any other type of file

	from a disk in a quick way, without any additional settings when inserting. Please refer to the Quickly insert image or media topic for more details.
	Insert Form - inserts an on-line form into the text. You can find more details on forms in the Modules -> Forms chapter.
	Insert Poll - inserts a poll into the text. You can find more details on polls in the Modules -> Polls chapter.
	Insert/Edit YouTube video - inserts a YouTube video. Please refer to the Insert YouTube video topic for more details.
	Insert/Edit widget - inserts an inline widget into the text. You can find more details about widgets in the Development -> Widgets chapter.

4.6.2 Insert image or media

4.6.2.1 Overview

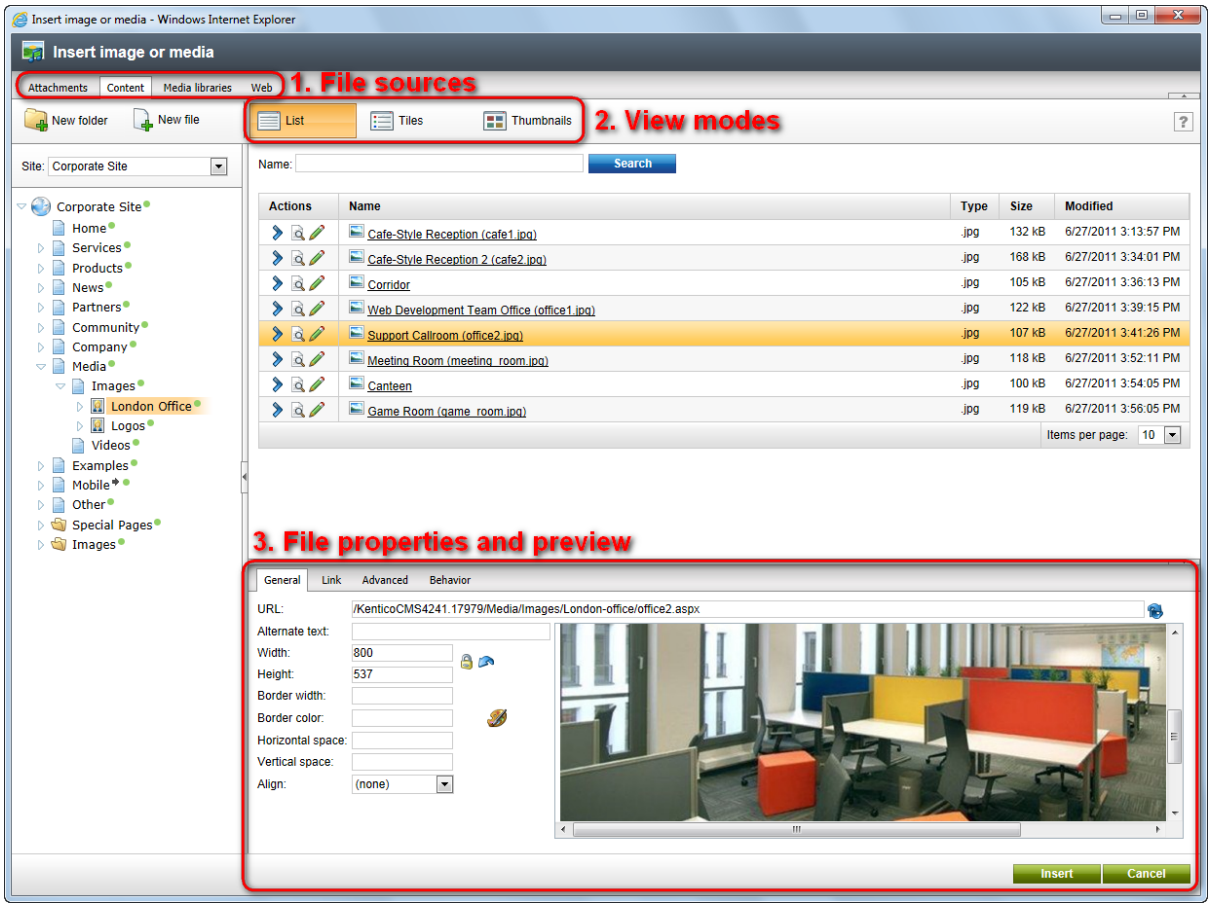
The **Insert image or media** dialog can be used to insert the following types of files:

- **Images** - bmp, gif, ico, png, wmf, jpg, jpeg, tiff, tif.
- **Audio** - wav, wma, mp2, mp3, mid, midi, mpga.
- **Video** - avi, mp4, mpg, mpeg, wmv, qt, mov, rm.
- **Flash** - swf.

Custom types can be added as described in the Configuring custom file types topic of the **Modules -> Media libraries** chapter.

The dialog can be opened by clicking the **Insert/edit image or media** ( ) icon, as highlighted in the screenshot below.

After clicking the icon, the following dialog will be displayed:



The highlighted parts of the dialog have the following functions. Their position may vary based on the selected file source:

1. **File sources** - using these four tabs, you can select from where the inserted files should be taken. Please refer to the File sources topic for more details.
2. **View modes** - using these buttons, you can switch between different modes of viewing files listed in the dialog. Please refer to the View modes topic for more details.
3. **File properties** - in this section, you can define properties of a file inserted into the text; the properties are different for images, audio/video and flash.

General process of inserting an image or media

1. Place the cursor in the appropriate position in the WYSIWYG editor.
2. Click the **Insert/edit image or media** ( ) icon.
3. The dialog window opens. Select the appropriate file source tab according to from where you want to add the file.
4. Locate and select the file on the tab.
5. Appropriate properties according to the file type are displayed. Specify the required properties and click the **Insert** button.
6. The required code is inserted into the WYSIWYG editor.

4.6.2.2 File sources

Using the four tabs at the top of the dialog window, you can choose from where the image or media should be inserted. The following tabs are available:

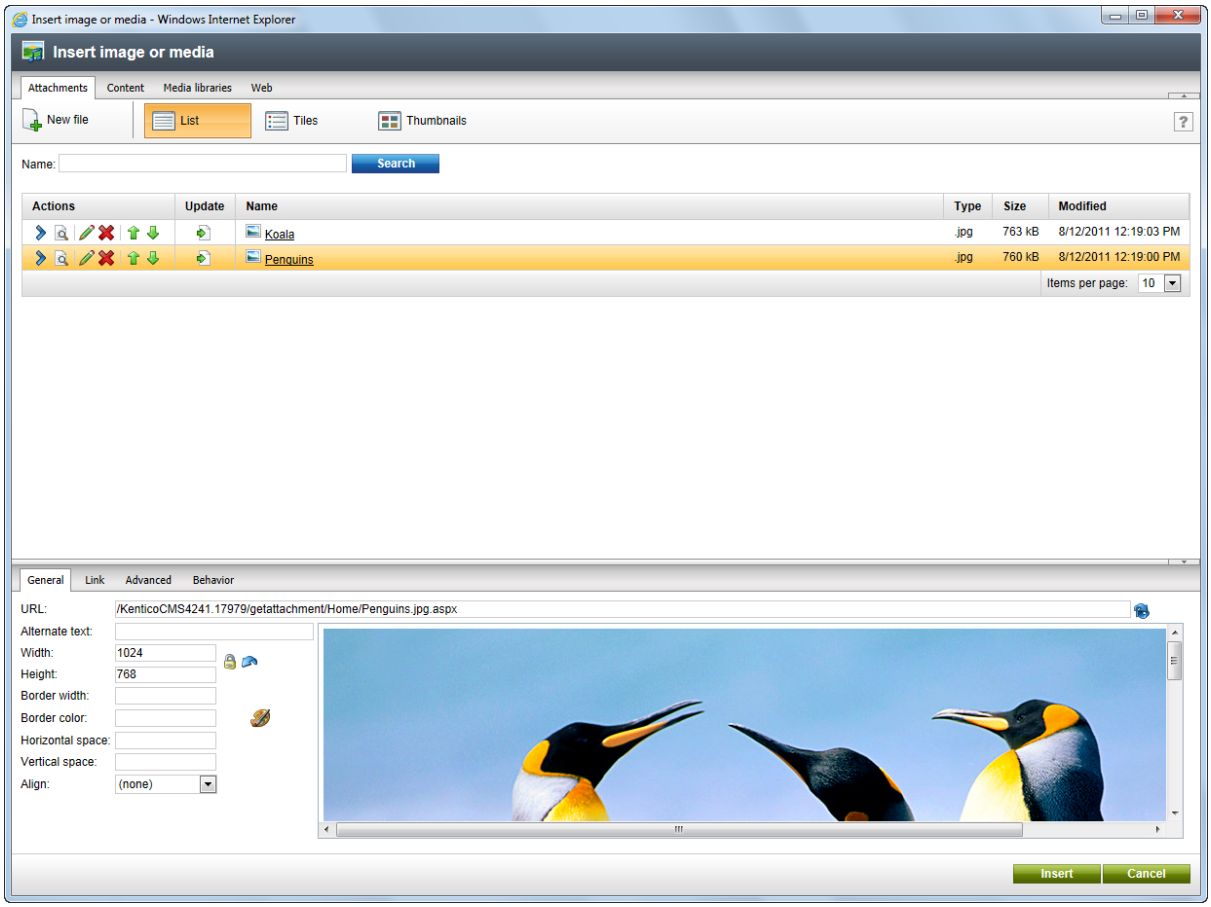
Attachments

The selection of files available on this tab depends on the location where the WYSIWYG editor is being used. When editing documents (e.g. on the **Form** tab, through user contributions etc.), the unsorted file attachments of the current document are offered. For more information about document attachments, please refer to the Document attachments chapter of this guide. When editing the content of other objects in the CMS (various types of layouts, e-mail templates etc.), this tab provides a way to insert the given object's metafile attachments.

From this tab, you can insert image attachments of the document into the text. You can select an attachment using the **Select** ( ) icon or by simply clicking the appropriate line (or tile/thumbnail in the other view modes).

You can upload new attachments using the **New file** button. The following actions can be performed with the listed attachments:

- After clicking the **View** ( ) icon, the attachment will be opened in a new window.
- Using the **Delete** ( ) icon, you can remove the attachment from the document.
- Using the **Move up** ( ) and **Move down** ( ) icons, you can change the order of document attachments. This option is available only in the **List** view mode. The order is stored in the **AttachmentOrder** property of each attachment. You can enter *AttachmentOrder* into the **ORDER BY expression** property of a displaying web part to have the attachments ordered accordingly.
- Using the **Edit** ( ) icon, you can edit the metadata of the file. However, clicking this icon placed beside an image opens the image in the built-in image editor.
- Using the **Update** ( ) icon, you can replace the original attachment with a new one.



Content

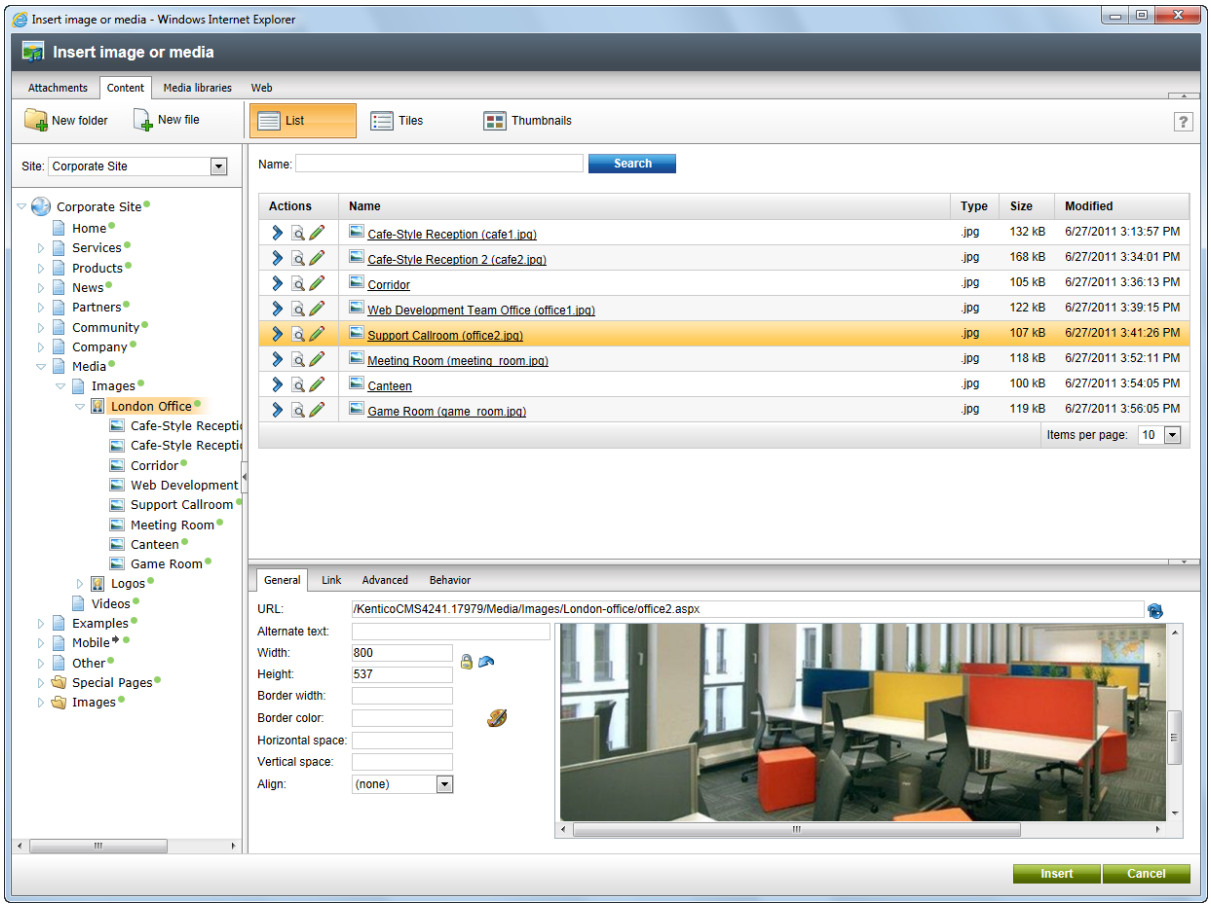
From this tab, you can insert files stored in the content tree of any of the sites running in the system. The site to insert from can be selected using the **Site** drop-down list, while its content tree will be displayed below. You can define which sites will be available. You can also define the starting alias path of the displayed content tree when defining a field in the field editor, as described in the Dialog configuration topic.

If you select a node of the content tree, all insertable files stored under it will be listed in the main area. You can filter the listed files by **Name**, **Type**, **Size** and the time when the files were **Modified** using the filter above the list.

You can select a file using the **Select** ( ) icon or by simply clicking the appropriate line (or tile/ thumbnail in the other view modes).

New folders (*cms.folder* documents) can be created in the content tree via this dialog using the **New folder** ( ) button. You can also upload new files (*cms.file* documents) into the content tree using the **New file** ( ) button. The following actions can be performed with the listed files:

- After clicking the **View** ( ) icon, the file will be opened in a new window.
- Using the **Edit** ( ) icon, you can edit metadata of the file. However, clicking this icon placed beside an image opens the image in the built-in image editor.



Media libraries

From this tab, you can insert files stored in a media library. Depending on the settings described in the Dialogs configuration topic, you can select a library using the set of three drop-down lists - **Site**, **Group** and **Library** - in the top right part of the dialog.

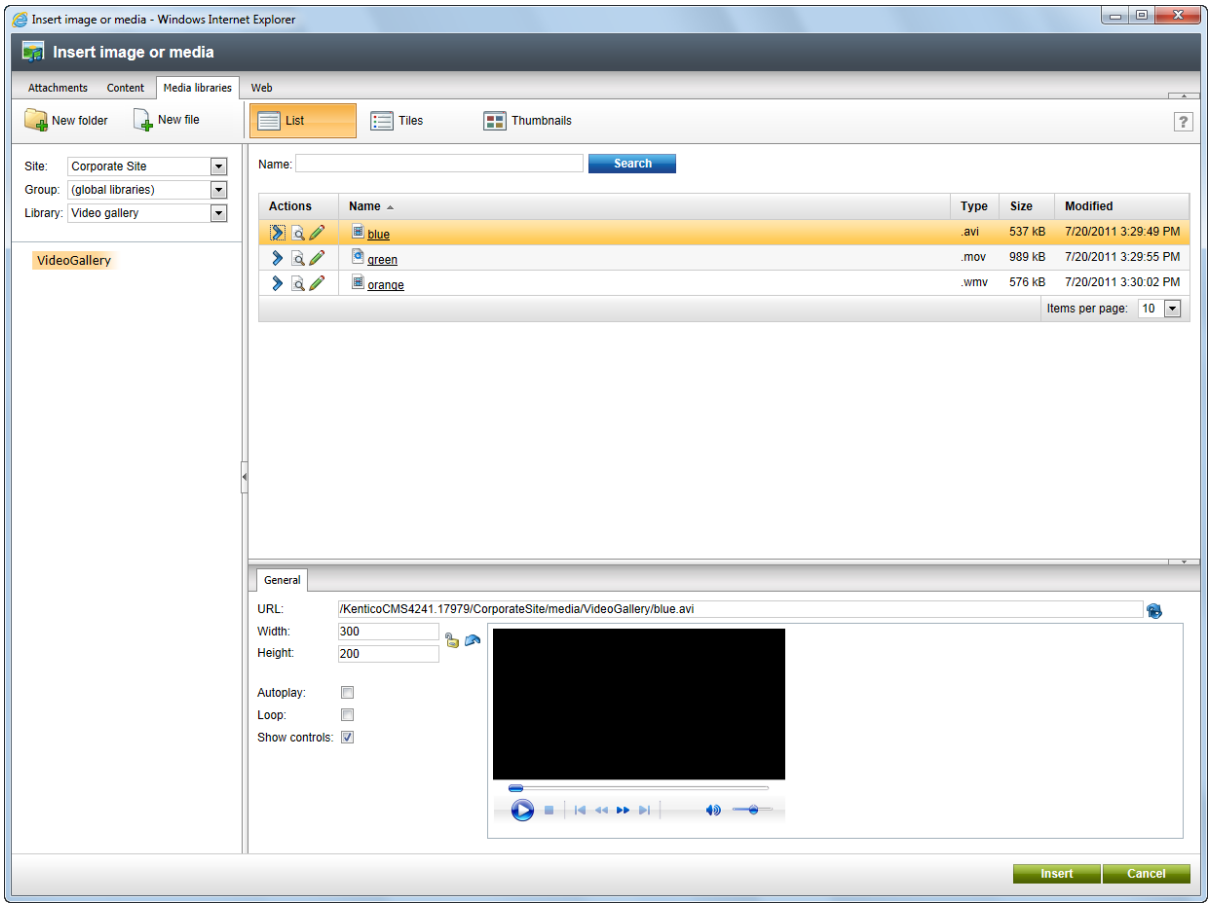
More information about Kentico CMS Media libraries can be found in the Modules -> Media libraries chapter.

When a library is selected, its folder structure is displayed below the drop-down lists. After selecting a folder, its content will be offered in the main area. You can filter the listed files by **Name**, **Type**, **Size** and the time when the files were **Modified** using the filter above the list.

You can select a file using the **Select** ( ) icon or by simply clicking the appropriate line (or tile/ thumbnail in the other view modes).

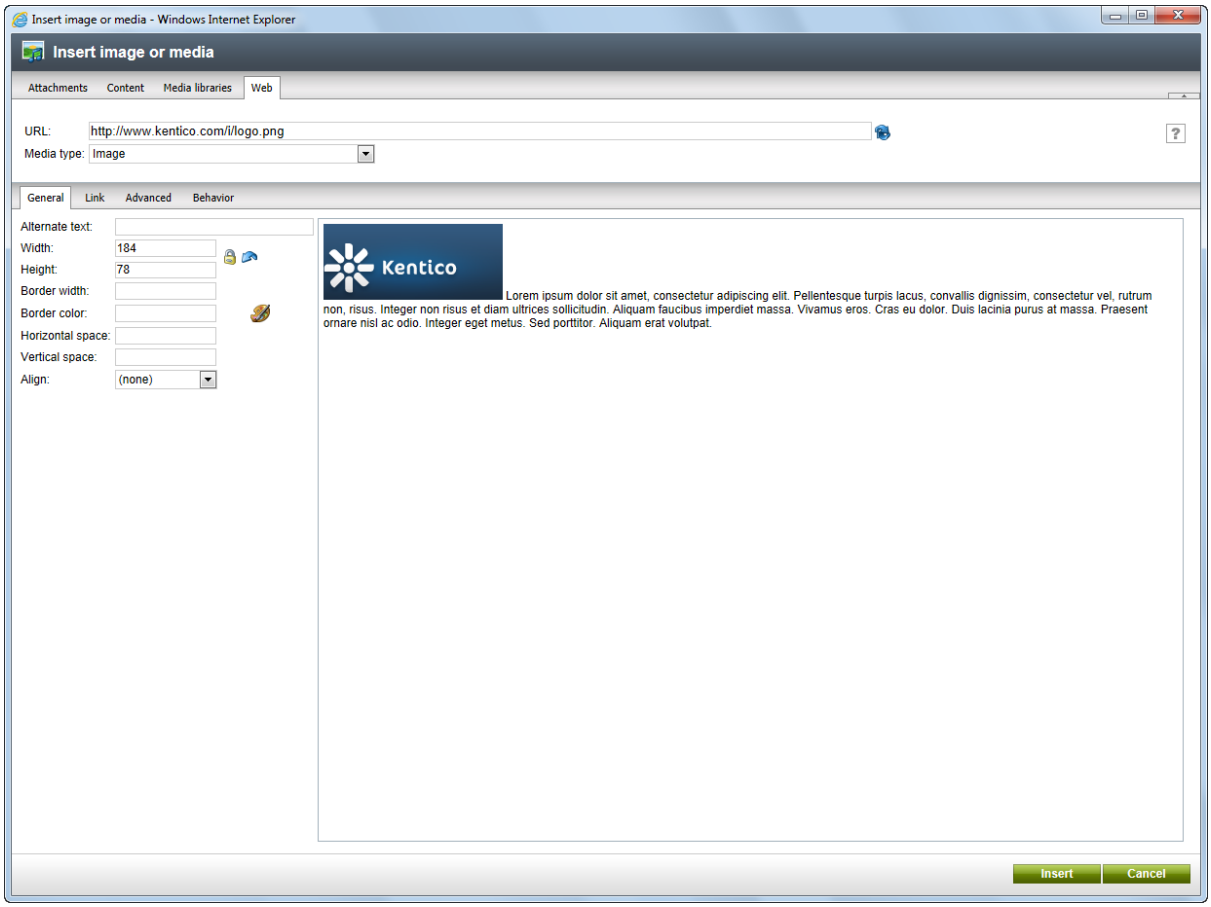
You can also perform the following actions with the listed files:

- After clicking the **View** ( ) icon, the file will be opened in a new window.
- Using the **Edit** ( ) icon, you can edit metadata of the file. However, clicking this icon placed beside an image opens the image in the built-in image editor.



Web

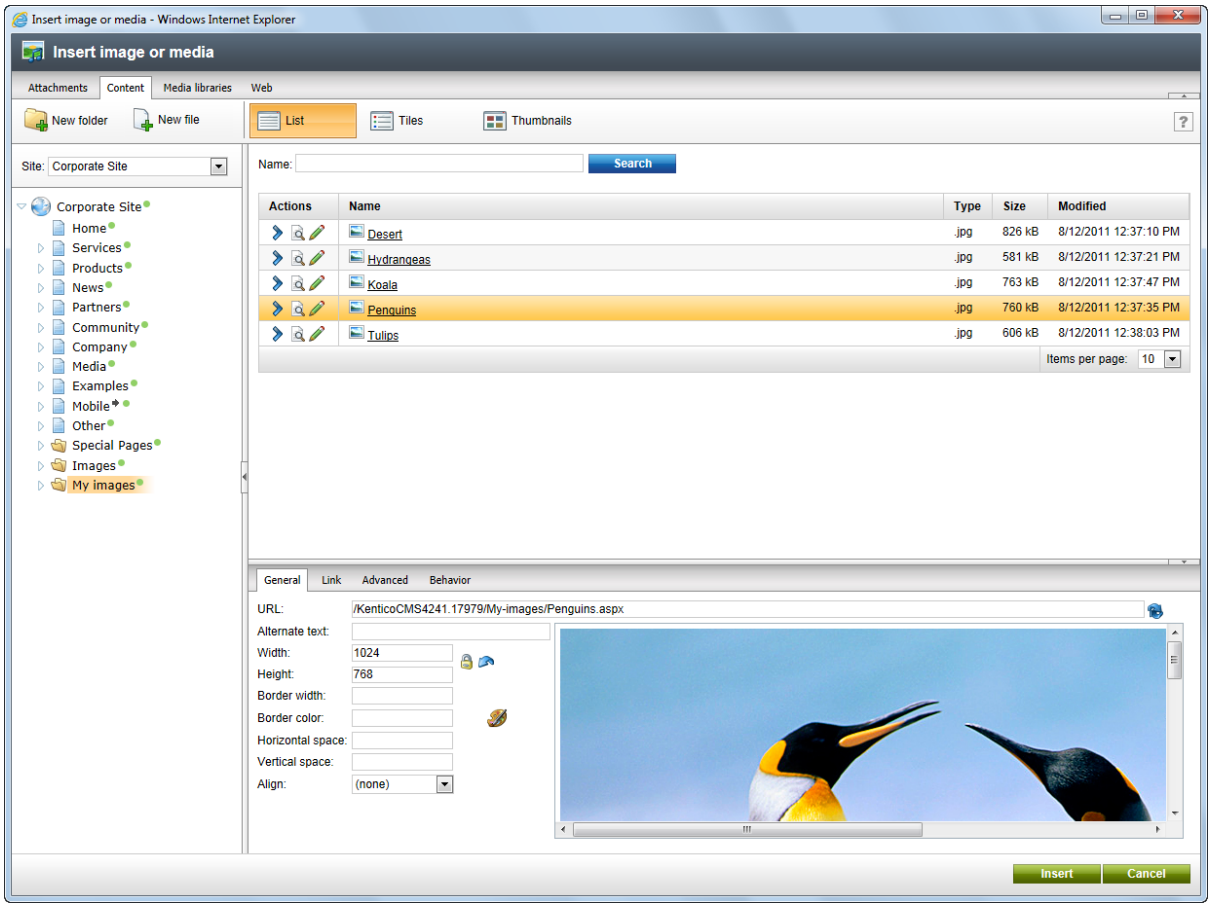
From this tab, you can enter an image, audio, video or flash from the web by entering its URL. More information on how to use this tab can be found in the Inserting images or media from the Web topic.



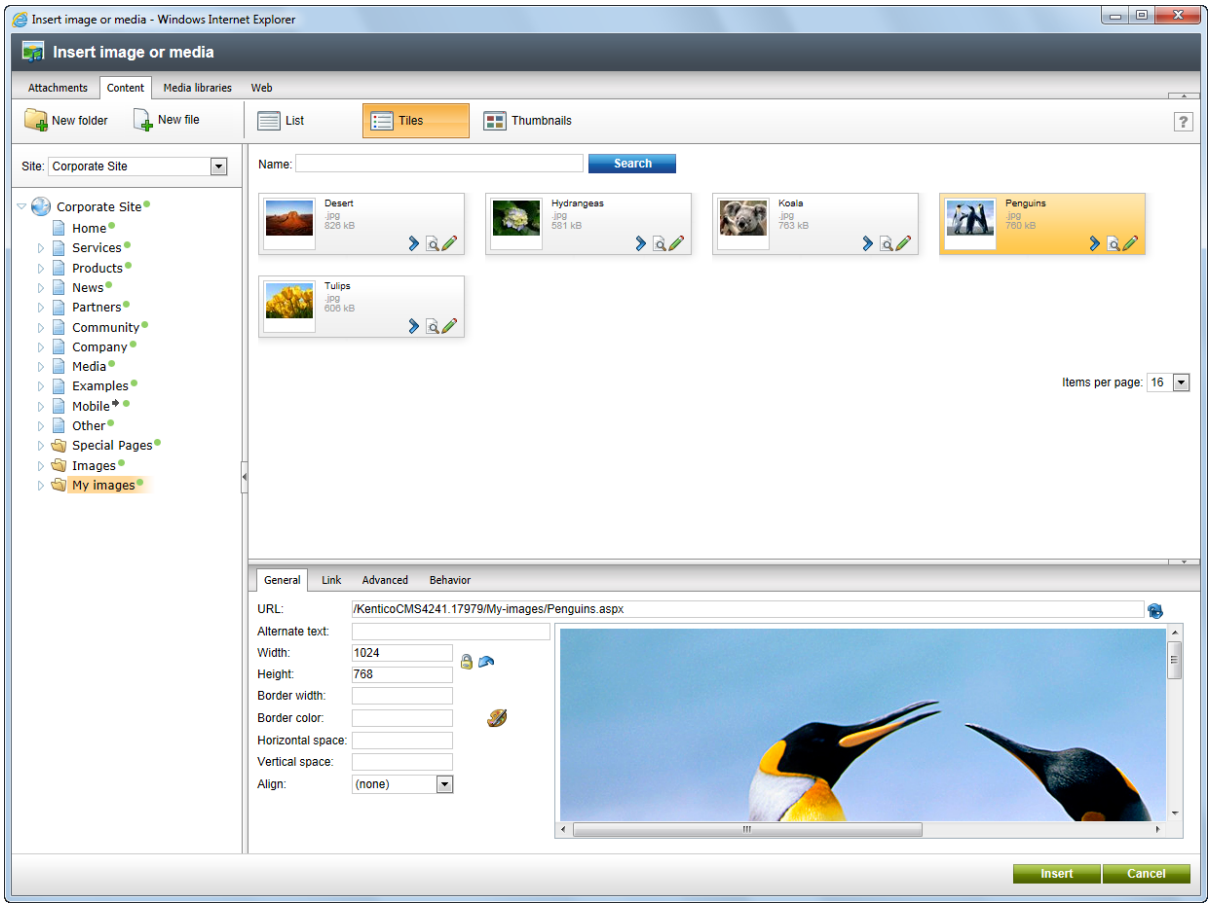
4.6.2.3 View modes

The following three view modes are available on the **Attachments**, **Content** and **Media libraries** tabs:

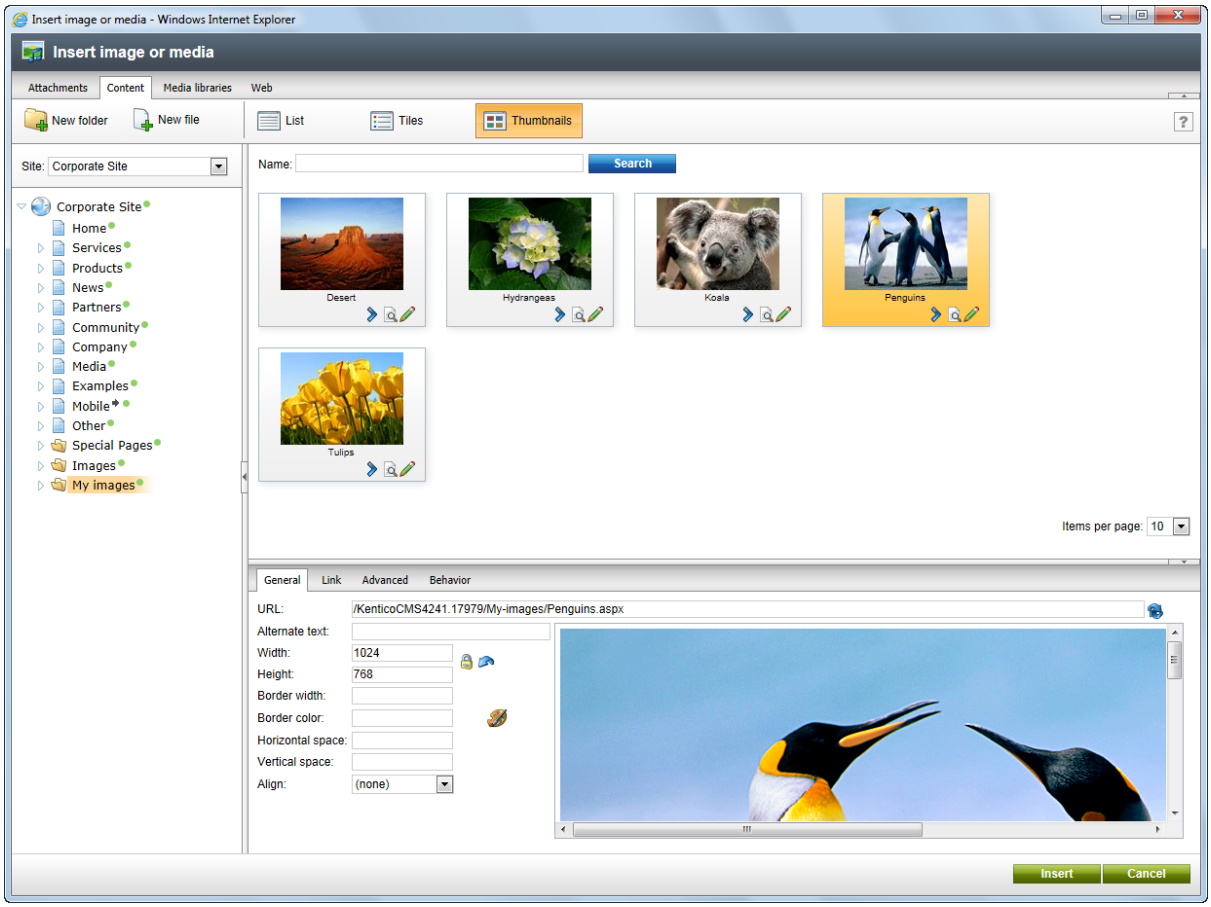
List



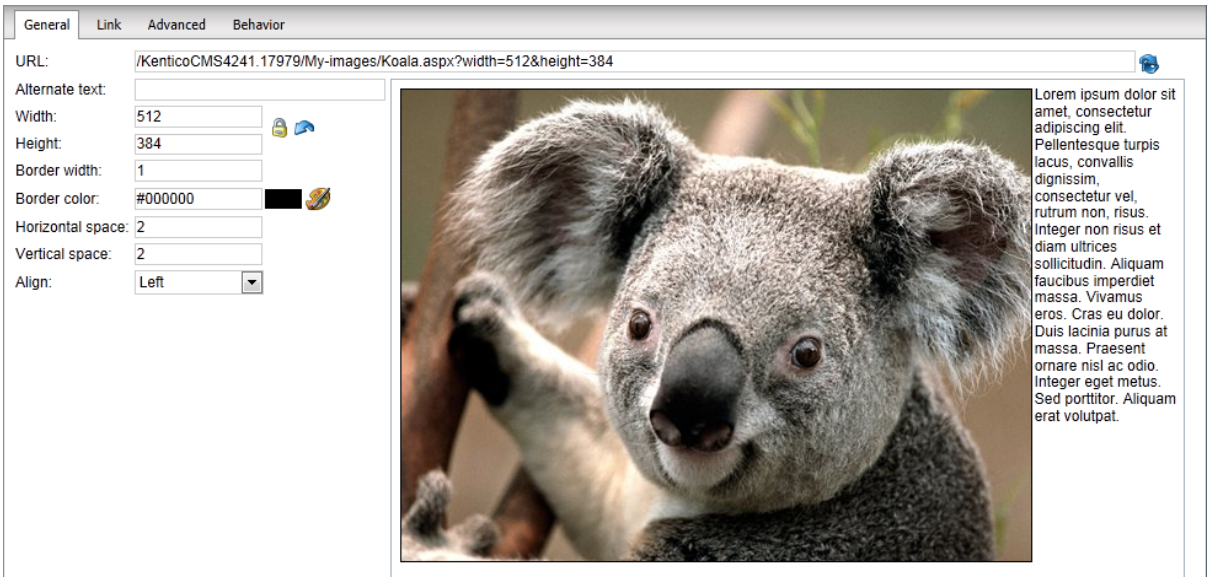
Tiles


Insert image or media - Windows Internet Explorer
Insert image or media
Attachments
Content
Media libraries
Web
New folder
New file
List
Tiles
Thumbnails
Site: Corporate Site
Name:
Search
Corporate Site
Home
Services
Products
News
Partners
Community
Company
Media
Examples
Mobile
Other
Special Pages
Images
My images
Desert
jpg
826 kB
Hydrangeas
jpg
581 kB
Koala
jpg
763 kB
Penguins
jpg
760 kB
Tulips
jpg
606 kB
Items per page: 16
General
Link
Advanced
Behavior
URL: /KenticoCMS4241.17979/My-images/Penguins.aspx
Alternate text:
Width: 1024
Height: 768
Border width:
Border color:
Horizontal space:
Vertical space:
Align: (none)
Insert
Cancel

Thumbnails



4.6.2.4 Inserting images



Inserting images via this dialog enables you to set a number of properties of the image. If you want to insert an image from the disk in a quick way, without specifying any properties, you can use the **Quickly insert image or media** ( ) icon as described in the Quickly insert image or media topic.

When inserting an image, its properties can be set on the following four tabs:

General

- **URL** - URL of the attached image.
- **Alternate text** - the text displayed when the image cannot loaded correctly. Using alternative text improves the accessibility of your website for visitors with screen readers or browsers that do not support images or have them disabled. It is also a good SEO practice.
- **Width**, **Height** - the width and height of the displayed image; in pixels.
  - Aspect ratio can be locked (), which makes the second dimension recalculated automatically when you change one dimension, while the ratio between the two dimensions is kept.
  - If unlocked (), dimensions can be entered manually into both fields, without the ratio being kept.
  - You can also reload the default dimensions using the **Reset size** () icon.
- **Border width** - the width of the border around the displayed image.
- **Border color** - the color of the border around the image; has no effect when border width is not set.
- **Horizontal space** - horizontal space between the image and the surrounding text.
- **Vertical space** - vertical space between the image and the surrounding text.
- **Align** - image alignment.

Link

- **URL** - if set, the image will become a link to the resource defined by the entered URL. Settings on the **Behavior** tab are overridden.
- **Target** - the destination where the linked resource should be displayed when the image is clicked.

Advanced

- **ID** - the identifier of the image HTML element.
- **Tooltip** - the text displayed when the mouse cursor is placed over the image.
- **Class** - the image element CSS class.
- **Style** - image element additional styles.

Behavior

- ***None*** - the image is inserted as a standard image. When the image is clicked, no action is performed.
- ***Open full size in the same window*** - the image will become a link. When the image is clicked, its full size is displayed in the same window.
- ***Open full size in a new window*** - the image will become a link. When the image is clicked, its full size is displayed in a new window.
- ***Show larger size on mouse-over*** - when the mouse cursor is placed over the image, the image is displayed in a "floating window" in the defined size. It will be inserted as the **Image** inline control.
  - **Width** - the width of the displayed image; in pixels.
  - **Height** - the height of the displayed image; in pixels.
    - Aspect ratio can be locked (), which makes the second dimension recalculated automatically when you change one dimension, while the ratio between the two dimensions is kept.
    - If unlocked (), dimensions can be entered manually into both fields, without the ratio being kept.
    - You can also reload the original dimensions using the **Reset size** () icon.

6 ways how images can be inserted

1. Standard image

- **Behavior tab** - *None*.
- **Link tab -> URL** - an empty field.

The image is inserted as a standard image with no special behavior. No action is performed when the image is clicked or mouse-overed.

The output code looks like the following code sample:

```
<img style="width: 320px; height: 240px" alt="" src="/KenticoCMS41/MyImages/
Waterfall.aspx" />
```

2. Image with link

- **Behavior tab** - *None*.
- **Link tab -> URL** - a URL is specified.

The image functions as a link to the specified URL. When the image is clicked, the user is redirected to the URL in the same browser window.

The output code looks like the following code sample:

```
<a href="www.kentico.com"><img style="width: 320px; height: 240px" alt="" src="/
KenticoCMS41/MyImages/Waterfall.aspx" /></a>
```

3. Image with special behavior - full size in the same window

- **Behavior tab** - *Open full size in the same window.*
- **Link tab -> URL** - an empty field.

When the image is clicked, it is displayed in full size in the same browser window.

The output code looks like the following code sample:

```
<a target="_self" href="/KenticoCMS41/MyImages/Waterfall.aspx"><img style="width:
320px; height: 240px" alt="" src="/KenticoCMS41/MyImages/Waterfall.aspx" /></a>
```

4. Image with special behavior - full size in a new window

- **Behavior tab** - *Open full size in a new window.*
- **Link tab -> URL** - an empty field.

When the image is clicked, it is displayed in full size in a new browser window.

The output code looks like the following code sample:

```
<a target="_blank" href="/KenticoCMS41/MyImages/Waterfall.aspx"><img style="width:
320px; height: 240px" alt="" src="/KenticoCMS41/MyImages/Waterfall.aspx" /></a>
```

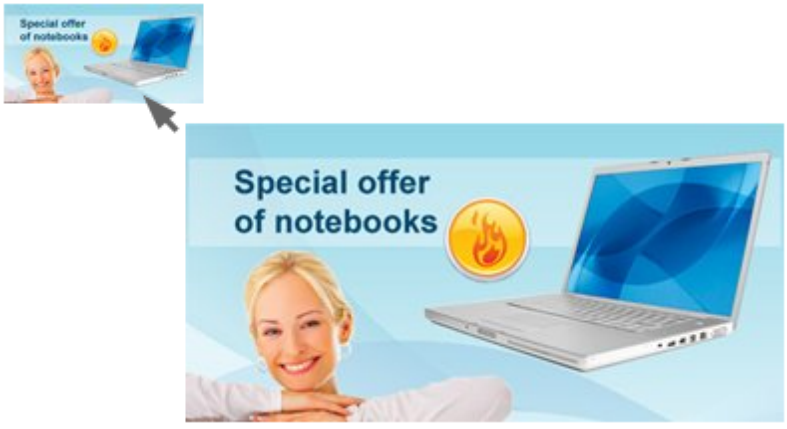
5. Image with special behavior - larger size on mouse-over

- **Behavior tab** - *Show larger size on mouse-over*; **Height** = *xx*; **Width** = *yy*.
- **Link tab -> URL** - an empty field.

When the image is mouse-overed, it is displayed in a new layer in size defined on the **Behavior** tab as shown in the screenshot below.

In this case, the image is inserted as the **Image** inline control. The output code looks like the following code sample:

```
<object codetype="CMSInlineControl" height="265" type="Image" width="400">
<param name="mouseoverwidth" value="400" />
<param name="ext" value=".jpg" />
<param name="mouseoverheight" value="265" />
<param name="behavior" value="hover" />
<param name="url" value="~/Images-(1)/Services_webdevelop.aspx" />
<param name="cms_type" value="image" />
</object>
```



6. Image with special behavior - larger size on mouse-over with link

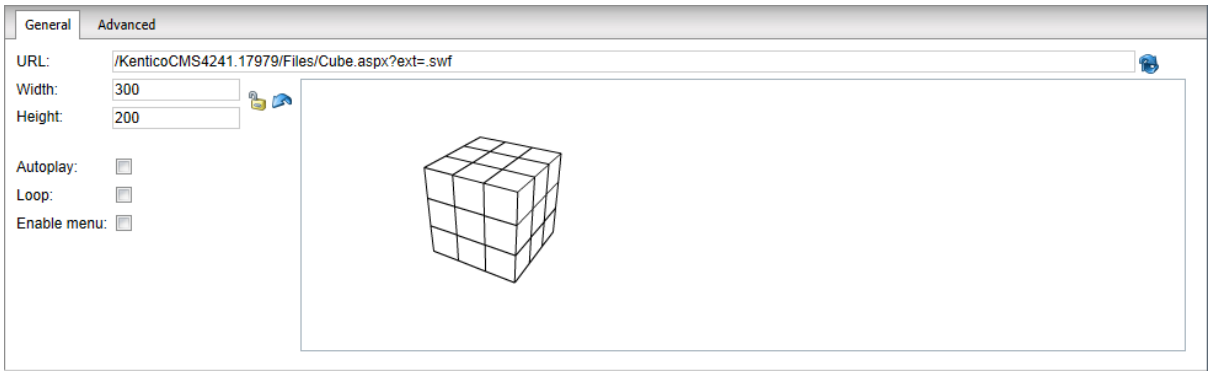
- **Behavior tab** - *Show larger size on mouse-over*; **Height** = *xx*; **Width** = *yy*.
- **Link tab -> URL** - a URL is specified.

When the image is mouse-overed, it is displayed in a new layer in size defined on the **Behavior** tab as shown in the screenshot above. The image is clickable and when clicked, the user is redirected to the specified URL in the same browser window.

In this case, the image is inserted as the **Image** inline control. The output code looks like the following code sample:

```
<a href="www.kentico.com">{^Image|(behavior)hover|(url)~/MyImages/Waterfall.aspx?
width=200&amp;height=150|(width)200|(height)150|(mouseoverwidth)400|
(mouseoverheight)300^}</a>
```

4.6.2.5 Inserting flash



When inserting flash, its properties can be specified on the following two tabs:

General

- **URL** - URL of the attached flash file.
- **Width**, **Height** - the width and height of the flash player; 300x200px is used by default.
  - Aspect ratio can be locked (), which makes the second dimension recalculated automatically when you change one dimension, while the ratio between the two dimensions is kept.
  - If unlocked (), dimensions can be entered manually into both fields, without the ratio being kept.
  - You can also reload the default dimensions using the **Reset size** () icon.
- **Autoplay** - indicates if the video plays automatically when the player loads.
- **Loop** - indicates if the player plays the video repeatedly in a loop.
- **Enables menu** - indicates if flash options are available in the flash context menu. The flash context menu is displayed on right click of the flash player.

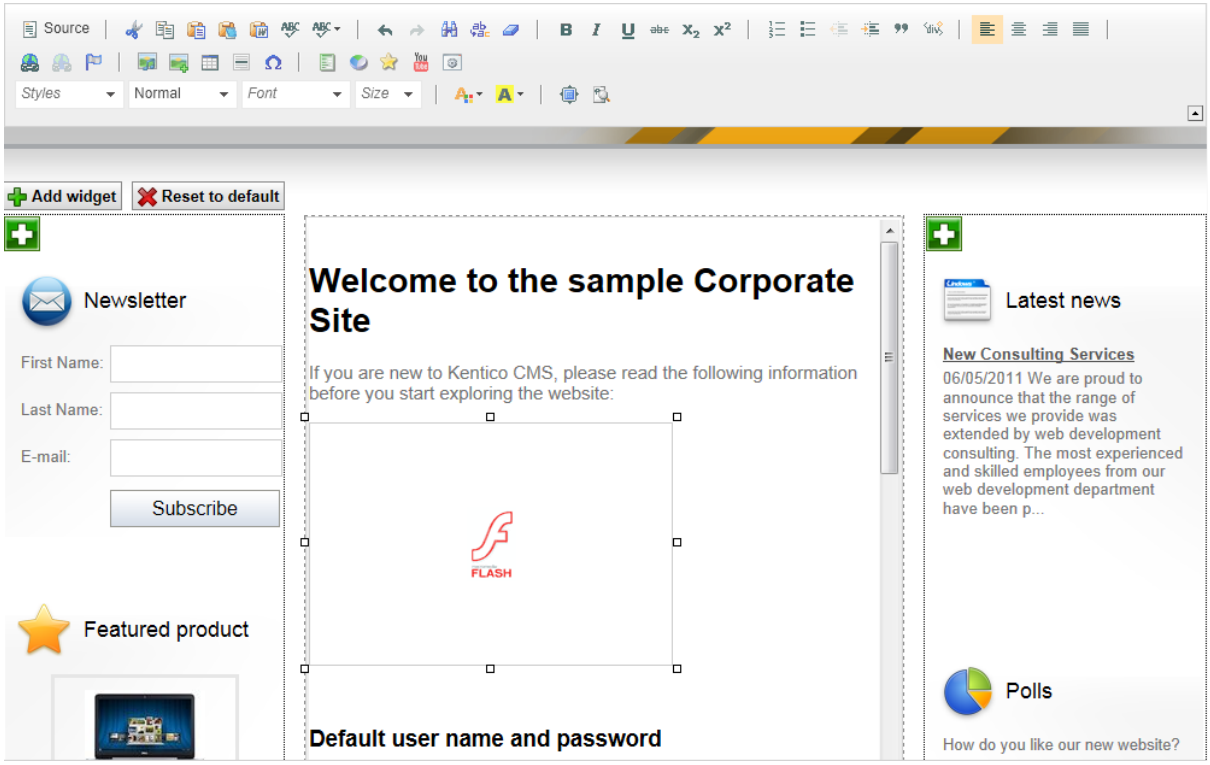
Advanced

- **Scale** - defines how the flash player stretches, shrinks and resizes when the browser window is resized.
- **ID** - the identifier of the flash HTML object.
- **Advisory title** - the text displayed when mouse cursor is placed over the flash player.
- **Class** - the flash element CSS class.
- **Style** - flash element additional styles.

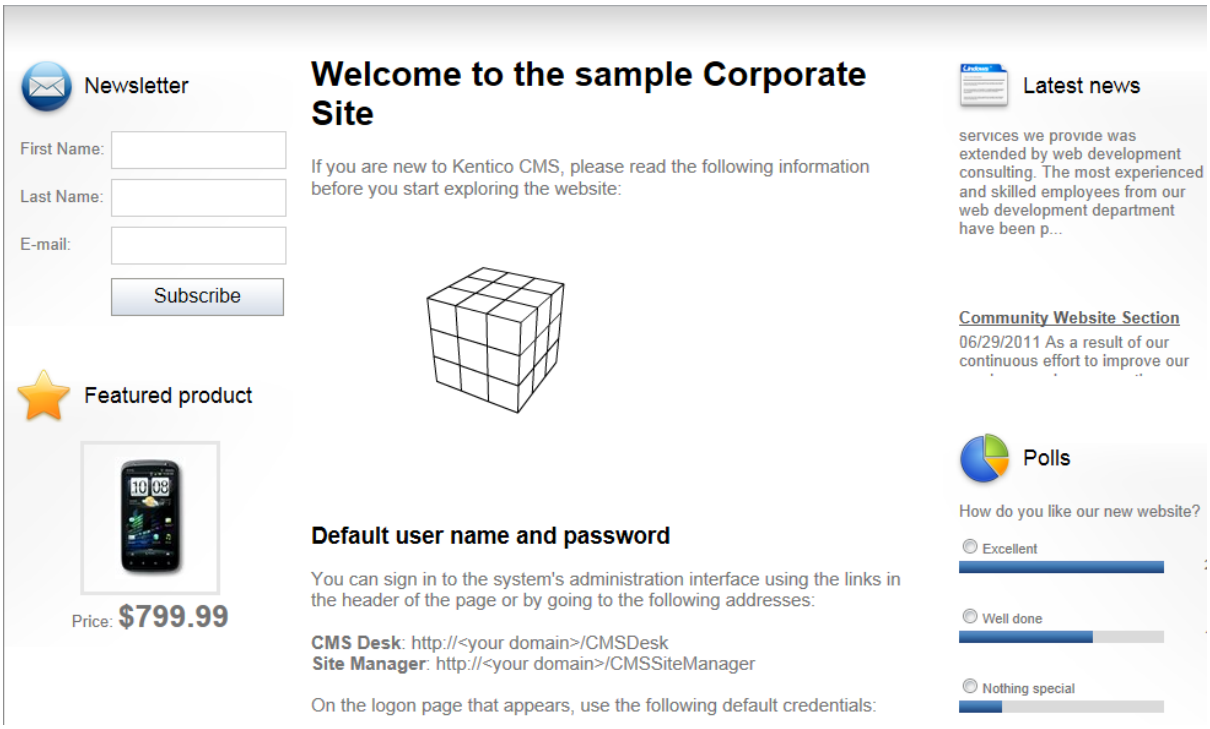
Flash is inserted into the output code as the **Media** inline control. The following code sample shows what the output code looks like:

```
<object codetype="CMSInlineControl" height="200" type="Media" width="300">
<param name="url" value="http://127.0.0.1/KenticoCMS/Files/Cube.aspx?ext=.swf" />
<param name="ext" value=".swf" />
<param name="cms_type" value="flash" />
</object>
```

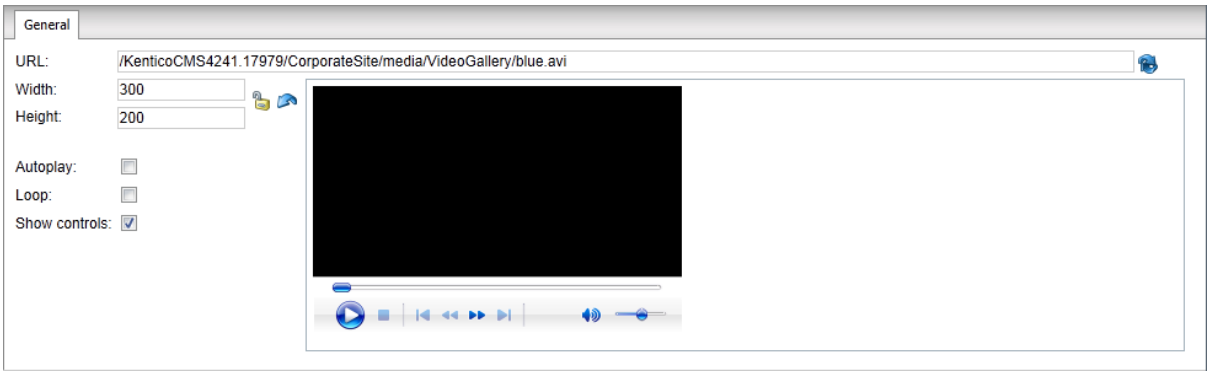
In the WYSIWYG editor, the flash is displayed only in the form of a box with the Flash logo, giving information about the size of the player:



And this is the result on the live site:



4.6.2.6 Inserting audio/video



When inserting both audio and video, the following properties can be set:

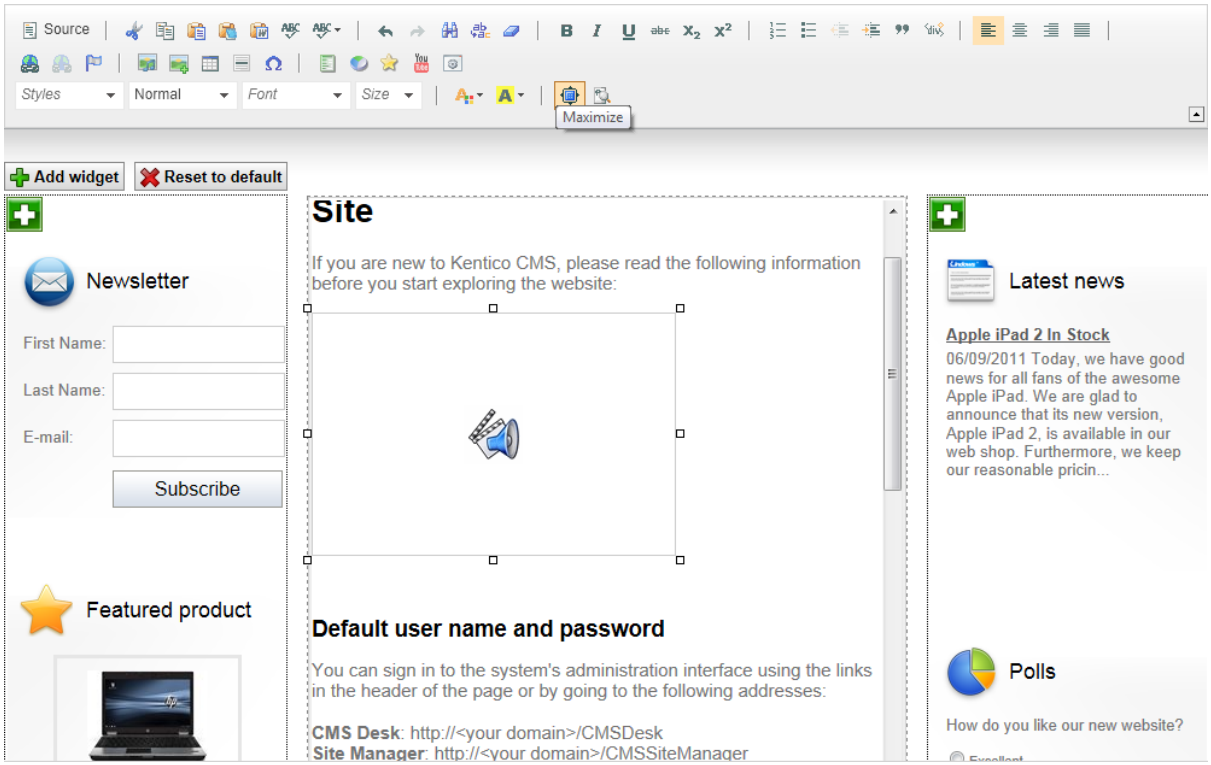
- **URL** - URL of the attached audio/video file.
- **Width**, **Height** - the width and height of the media player; 300x200px is used by default.
  - Aspect ratio can be locked (), which makes the second dimension recalculated automatically when you change one dimension, while the ratio between the two dimensions is kept.
  - If unlocked (), dimensions can be entered manually into both fields, without the ratio being kept.
  - You can also reload the default dimensions using the **Reset size** () icon.
- **Autoplay** - indicates if playback starts automatically when the page is loaded.
- **Loop** - indicates if playback is performed repeatedly in a loop.
- **Show controls** - indicates if playback controls (play, stop, fast forward, ...) are displayed. In some browsers, the controls may not be displayed if the player size is too small even if this option is enabled.

Audio and video is inserted into the output code as the **Media** inline control. The output code looks like

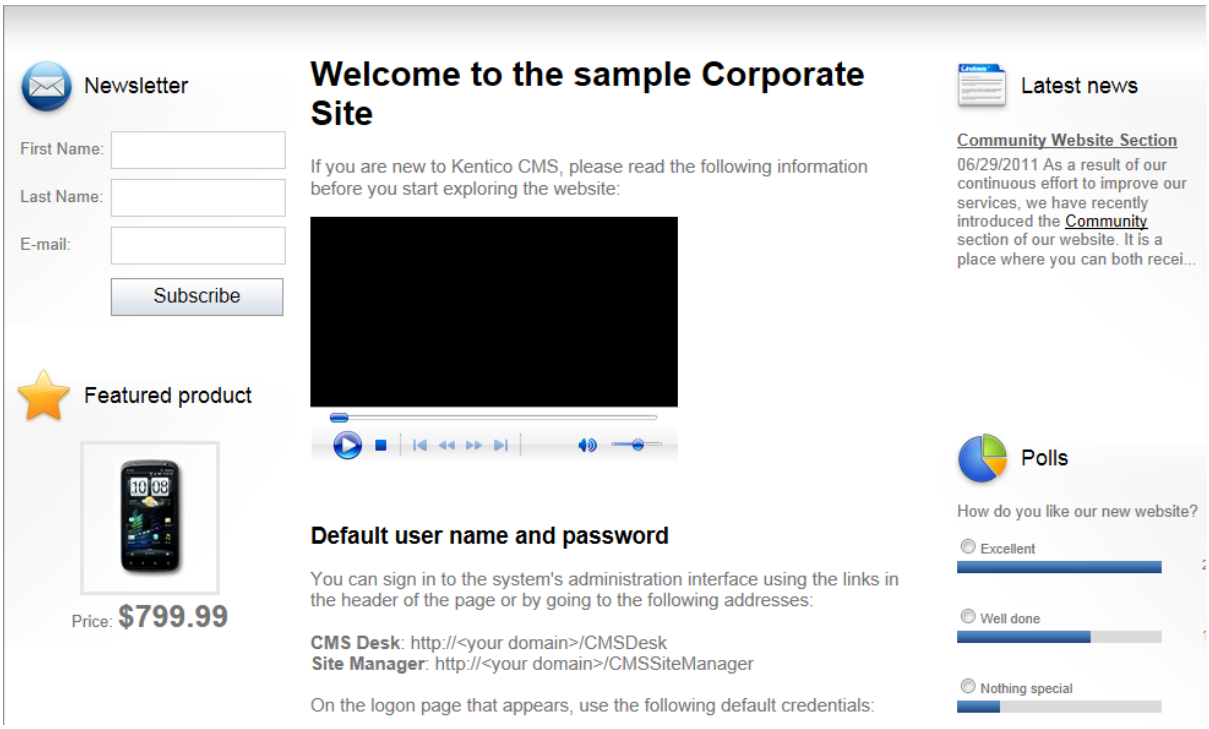
the following code sample:

```
<object codetype="CMSInlineControl" height="200" type="Media" width="300">
<param name="ext" value=".avi" />
<param name="cms_type" value="audiovideo" />
<param name="url" value="~/getattachment/Sample.avi.aspx" />
</object>
```

In the WYSIWYG editor, the player is not displayed. Instead, only a box with the audio/video icon can be seen, giving information about the size of the player on the live site:



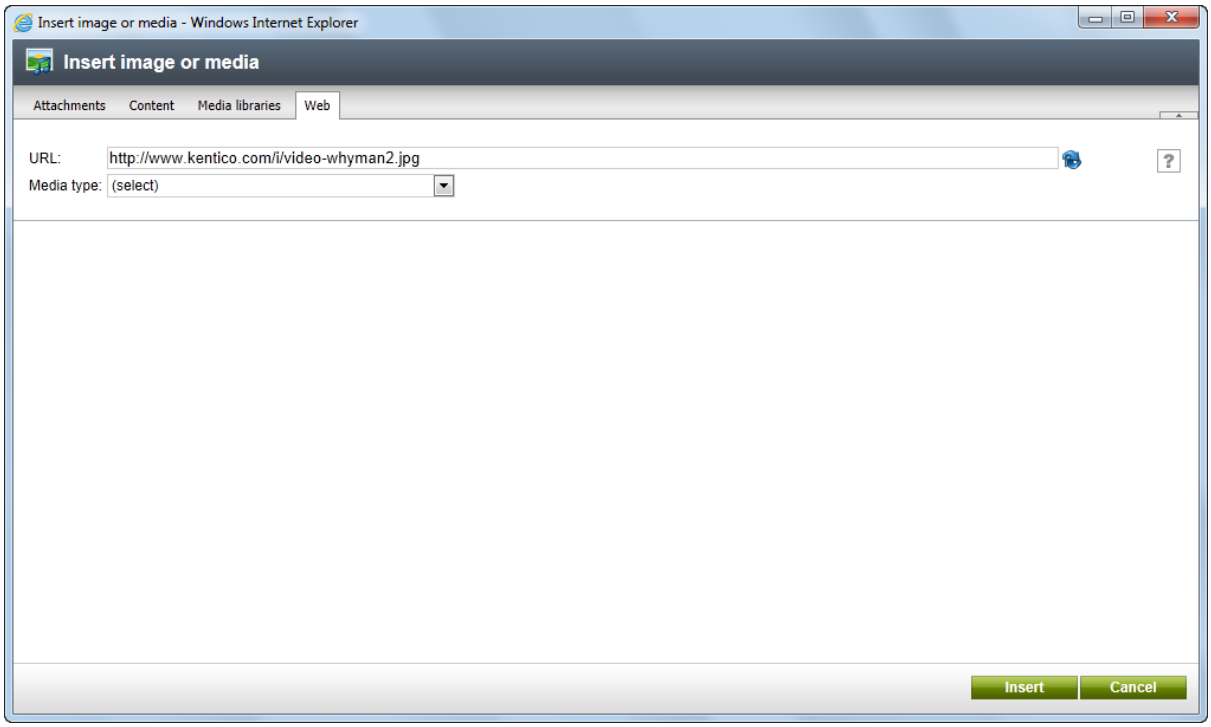
And this is how it looks on the live site:



4.6.2.7 Inserting images or media from the Web

Via the **Web** tab you can insert from the web any of the file types enumerated in the Insert image or media -> Overview topic, just by entering the respective URL. The generated code depends on the type of linked media and looks as the code samples mentioned in the previous three chapters.

The dialog initially looks like this on the tab:



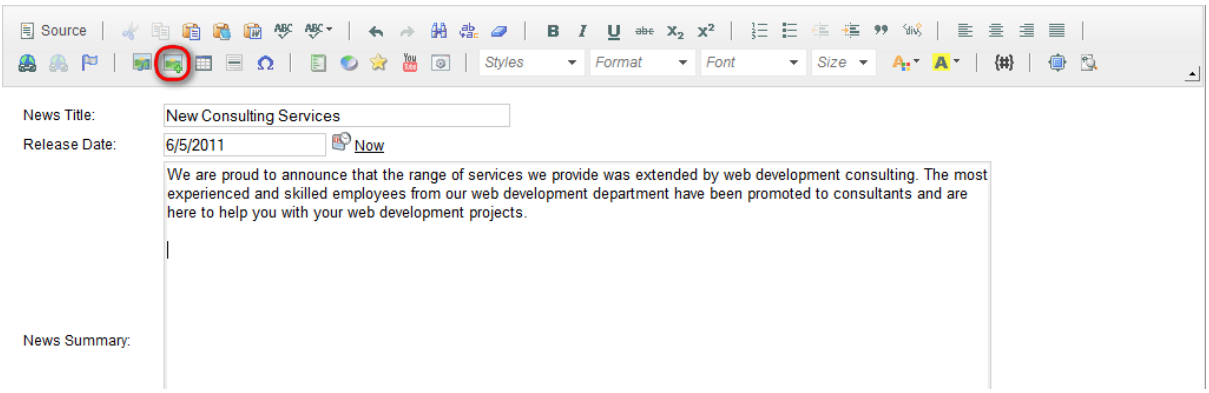
The general process of adding images or media from the web is as follows:

1. Enter the URL of the resource into the **URL** field.
2. Try automatic file type detection using the **Refresh** ( ) icon. In case that the file type cannot be detected, you can still choose it manually from the **Media type** drop-down list.
3. Based on the file type, its properties will be loaded into the main area. The properties of individual file types are described in the Inserting images, Inserting flash and Inserting audio/video topics.
4. Enter the properties and click the **Insert** button. The image or media file is inserted into the text.

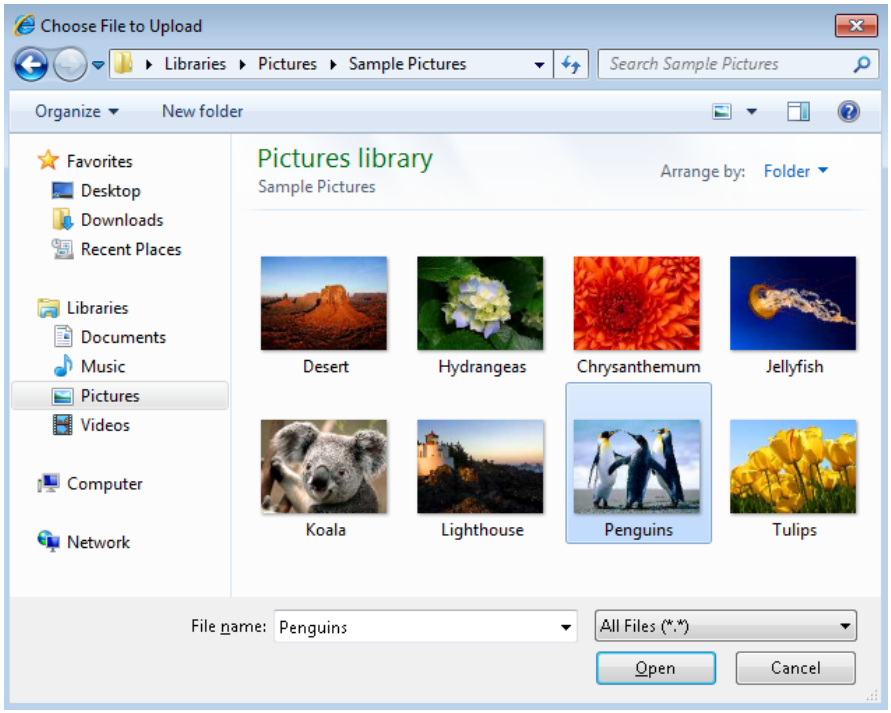
4.6.3 Quickly insert image or media

The Quickly insert image or media action can be used to insert an image, video, flash video or document from a disk in a quick way. The following three steps need to be taken to insert media this quick way:

1. Place the cursor in the appropriate position and click the **Quickly insert image or media** ( ) icon on the WYSIWYG editor toolbar.



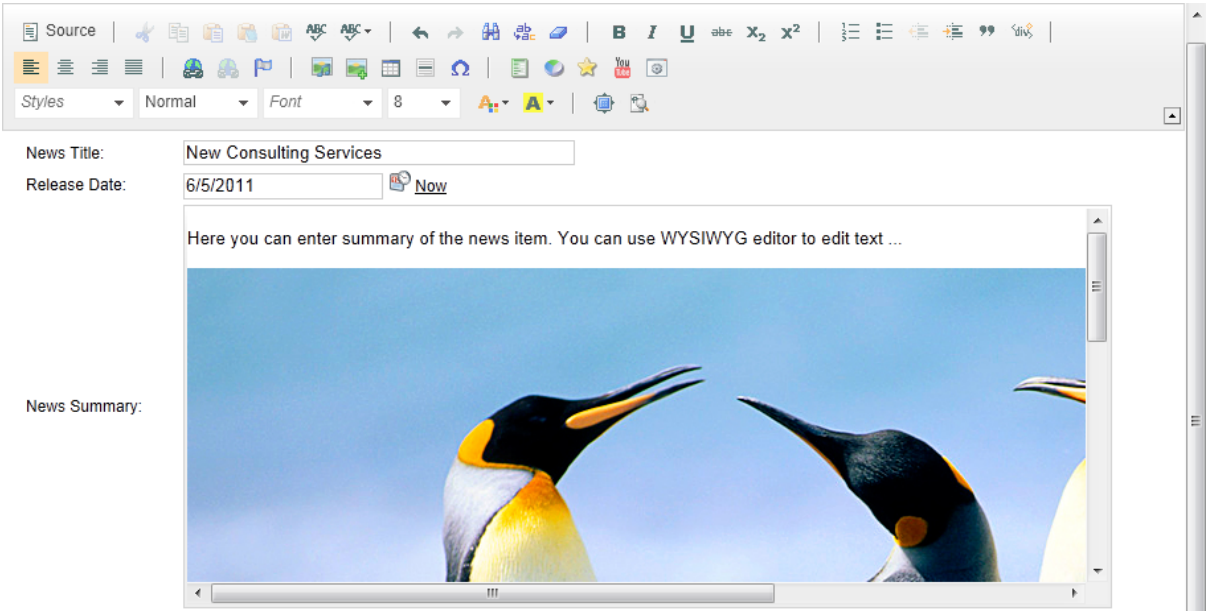
2. The **Choose file** dialog of your browser opens. Locate the file on the disk and click **Open**.



3. Result

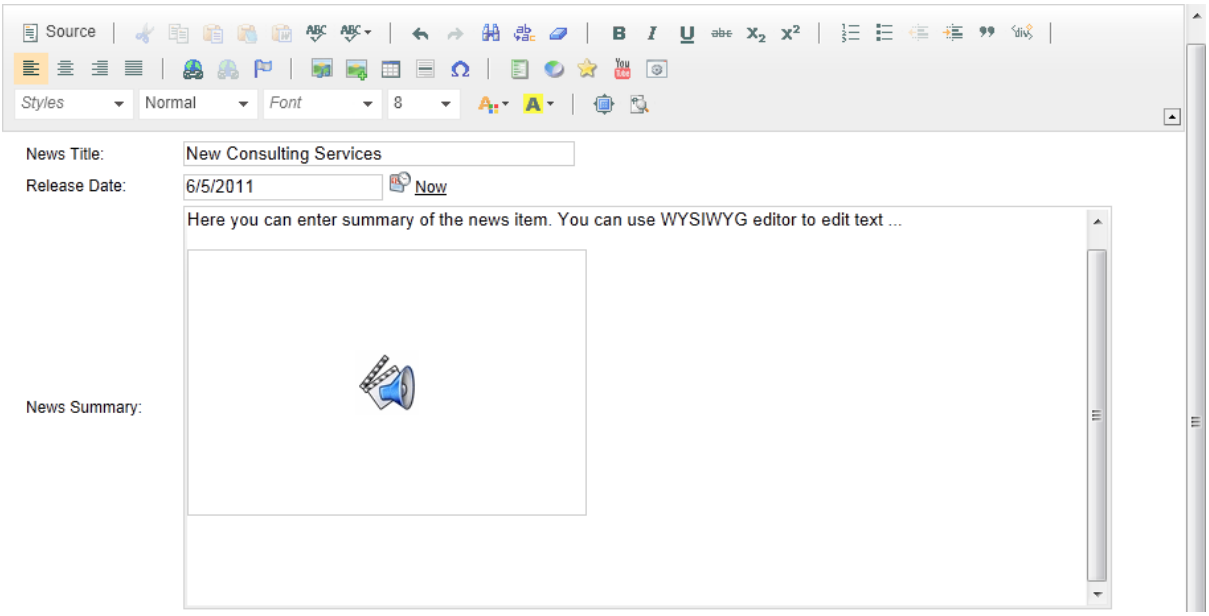
Image

An image is inserted into the text in its original size; to learn how to change this setting, please refer here. At the same time, it is uploaded to the document as its attachment. However, you can edit the image just as images inserted via the **Insert image or media** dialog. For more information, please refer to the Editing inserted items topic.



Video and flash video

Same as images, videos are inserted into the text and uploaded to the document as its attachments. The size of an inserted video is 300x200px by default; to learn how to change this setting, please refer here. Videos inserted this way can be edited just as videos inserted via the **Insert image or media** dialog. For more information, please refer to the Editing inserted items topic.

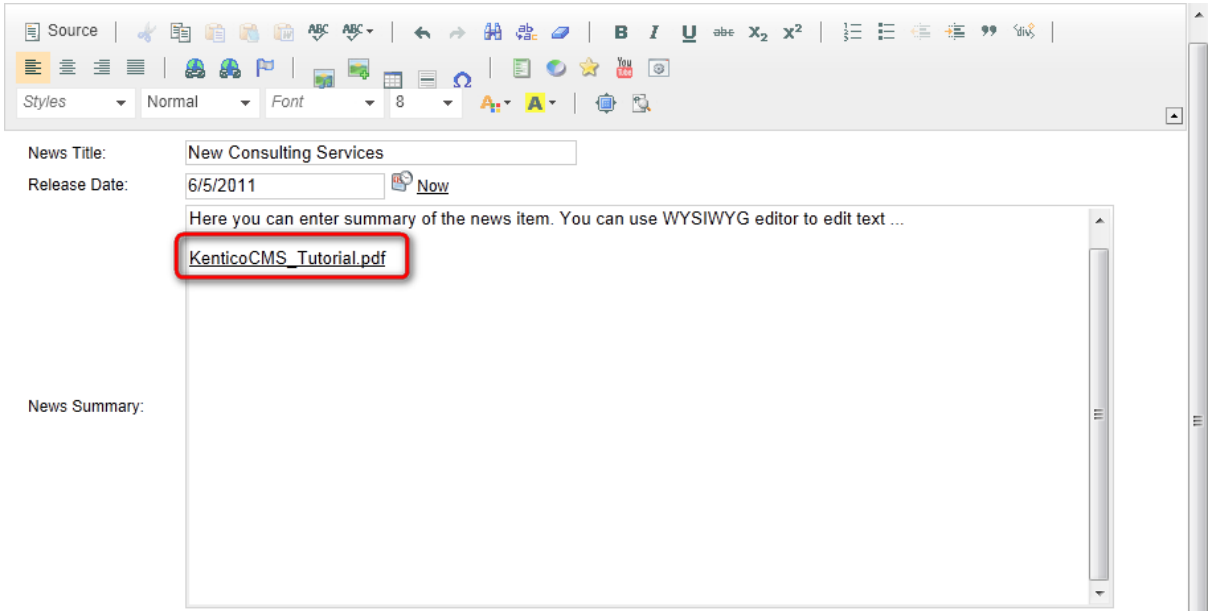


Please note

To change the default size of the inserted audio/video player, you need to modify the *DEFAULT_OBJECT_WIDTH* and *DEFAULT_OBJECT_HEIGHT* constants in the ***DirectFileUploaderControl.ascx.cs*** control located in ***<your web project folder>/CMSModules/Content/Controls/Attachments/DirectFileUploader***.

Document

A document file or any other type of file except for an image, video or flash video is inserted into the text as a link tag.



Please note

The link to a document file has the following format: ***<attachment name.file extension>***.

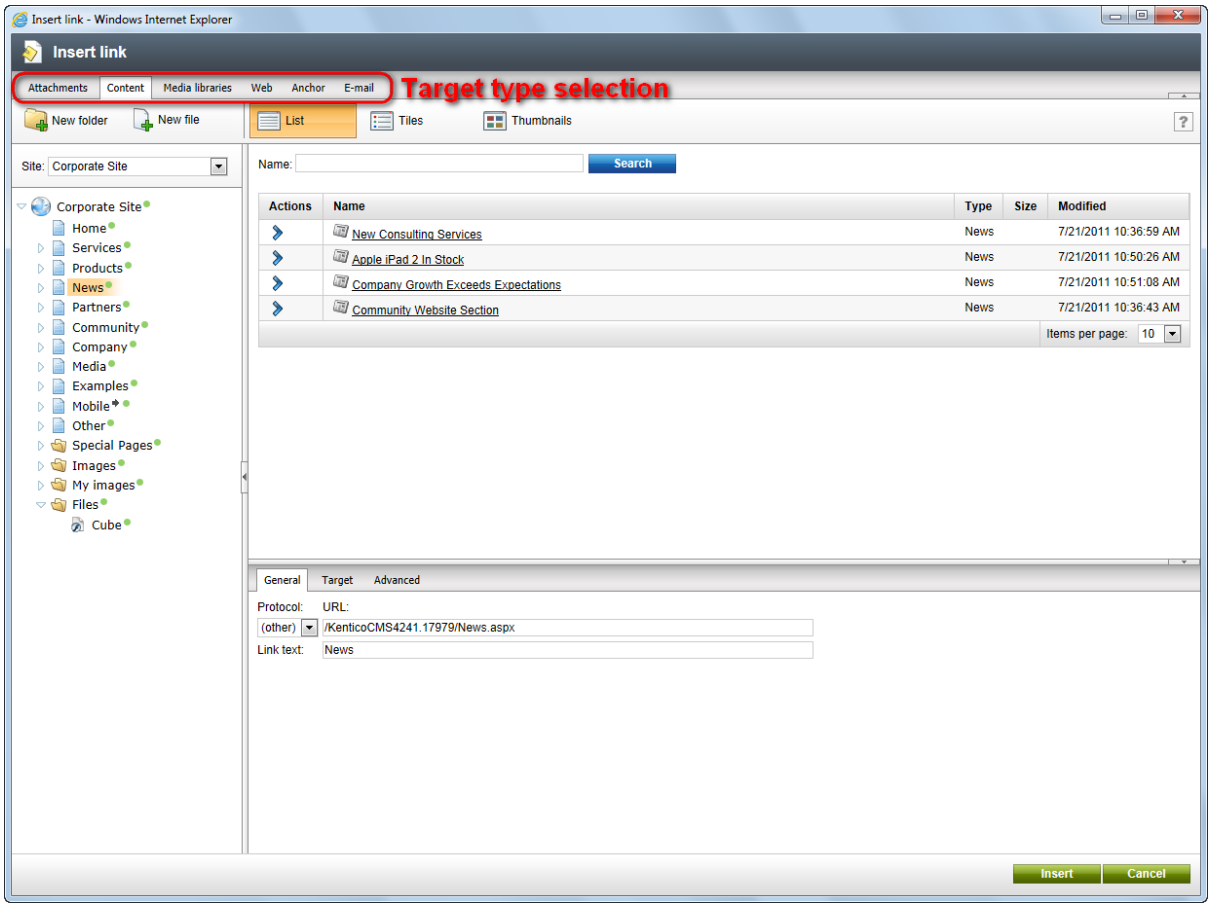
4.6.4 Insert link

4.6.4.1 Overview

The **Insert link** dialog is accessible by clicking the **Insert link** ( ) icon on the WYSIWYG editor toolbar:

All links are inserted using **<a>** tags and the following types of targets can be linked:

- Content within the CMS - via the **Attachments**, **Content** and **Media libraries** tabs.
- Content anywhere on the Web - via the **Web** tab.
- Anchors in documents - via the **Anchor** tab.
- Mailto links - via the **E-mail** tab.

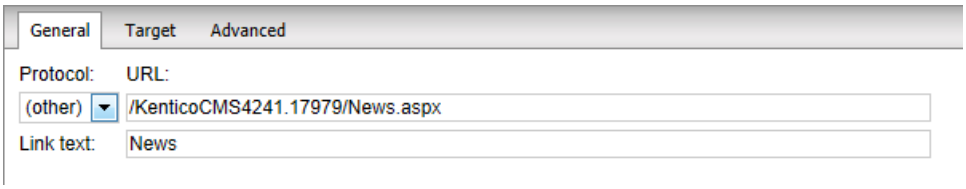


General process of inserting a link

1. Select the text that should become a link or place the cursor to the position where the link text should be inserted.
2. Click the **Insert link** ( ) icon on the WYSIWYG editor toolbar.
3. Choose the appropriate tab and specify where the link should be leading.
4. Specify link properties and click the **Insert** button.
5. The link is inserted into the text.

4.6.4.2 Link properties

On the **Attachments**, **Content**, **Media libraries** and **Web** tabs, you can see the following section for setting up link properties:



You can specify the following properties on the respective tabs:

General

- **Protocol + URL** - the address of the linked resource.
- **Link text** - the text of the link that will appear in the text. This field is visible only when inserting a link into an empty space in the text area, i.e. when no text or object is selected.

Please note

On the **Web** tab, only the **Target** and **Advanced** tabs are displayed and the general properties are displayed above them.

Target

- **Target** - using this drop-down list, you can define where the link will be opened.
- **Target frame name** - the name of the frame where the target should be opened. This option is displayed only if the **Target** property is set to *(frame)*.

Advanced

- **ID** - the identifier of the link HTML element.
- **Name** - the name of the link HTML object.
- **Tooltip** - the text displayed when the mouse cursor is placed over the link.
- **Class** - the link element CSS class.
- **Style** - additional link element styles.

4.6.4.3 Links to content within the CMS

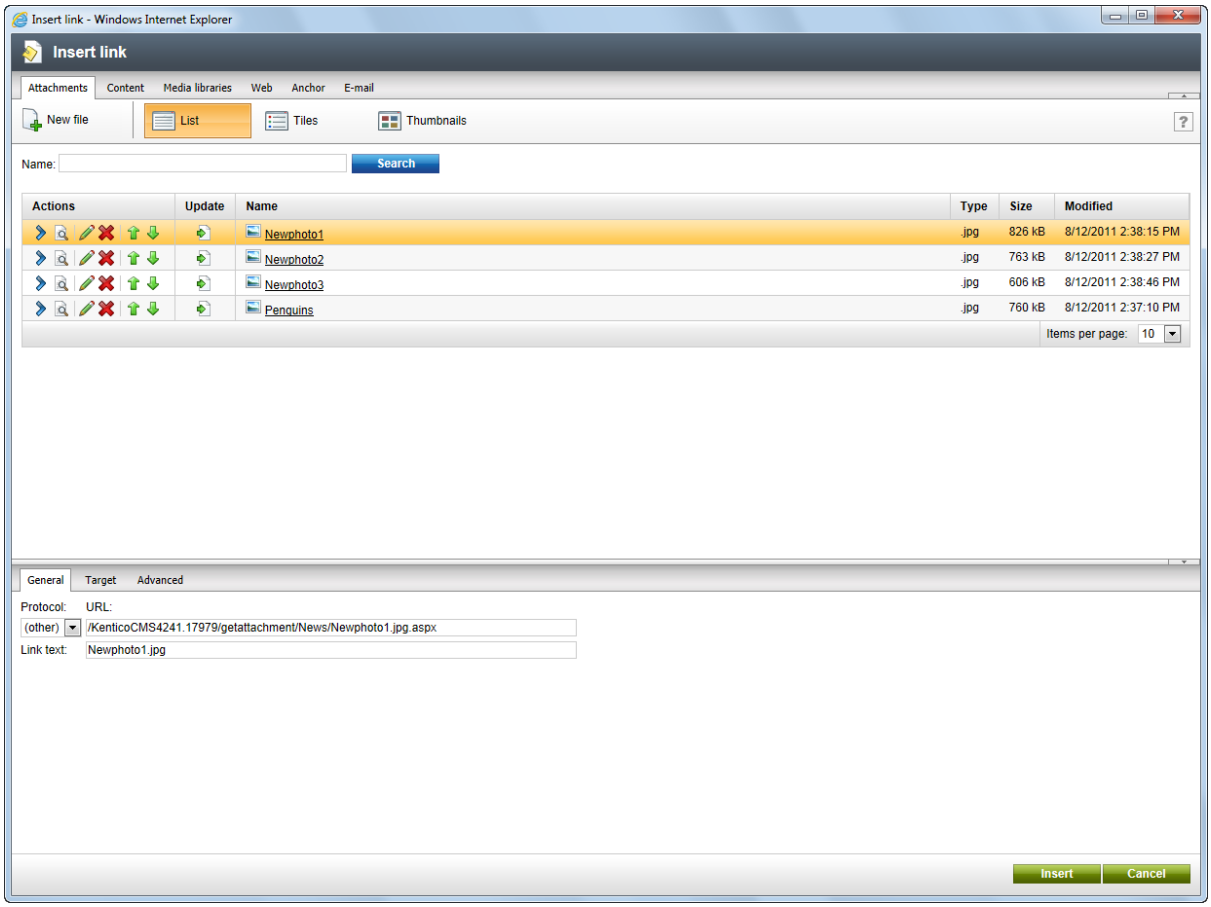
Links to content within the CMS can be inserted via the **Attachments**, **Content** and **Media libraries** tabs. Detailed descriptions of all possible actions that can be performed on the tabs can be found in the File sources topic of the **WYSIWYG editor -> Insert image or media** subchapter.

Attachments

Via this tab, you can insert links to attachments of the current document. For more information about document attachments, please refer to the Document attachments chapter of this guide.

The following code sample shows what the output code looks like:

```
<a href="/3644_9682/getattachment/News/Your-first-news/
NewsPhoto1.JPG.aspx">NewsPhoto1.JPG</a>
```

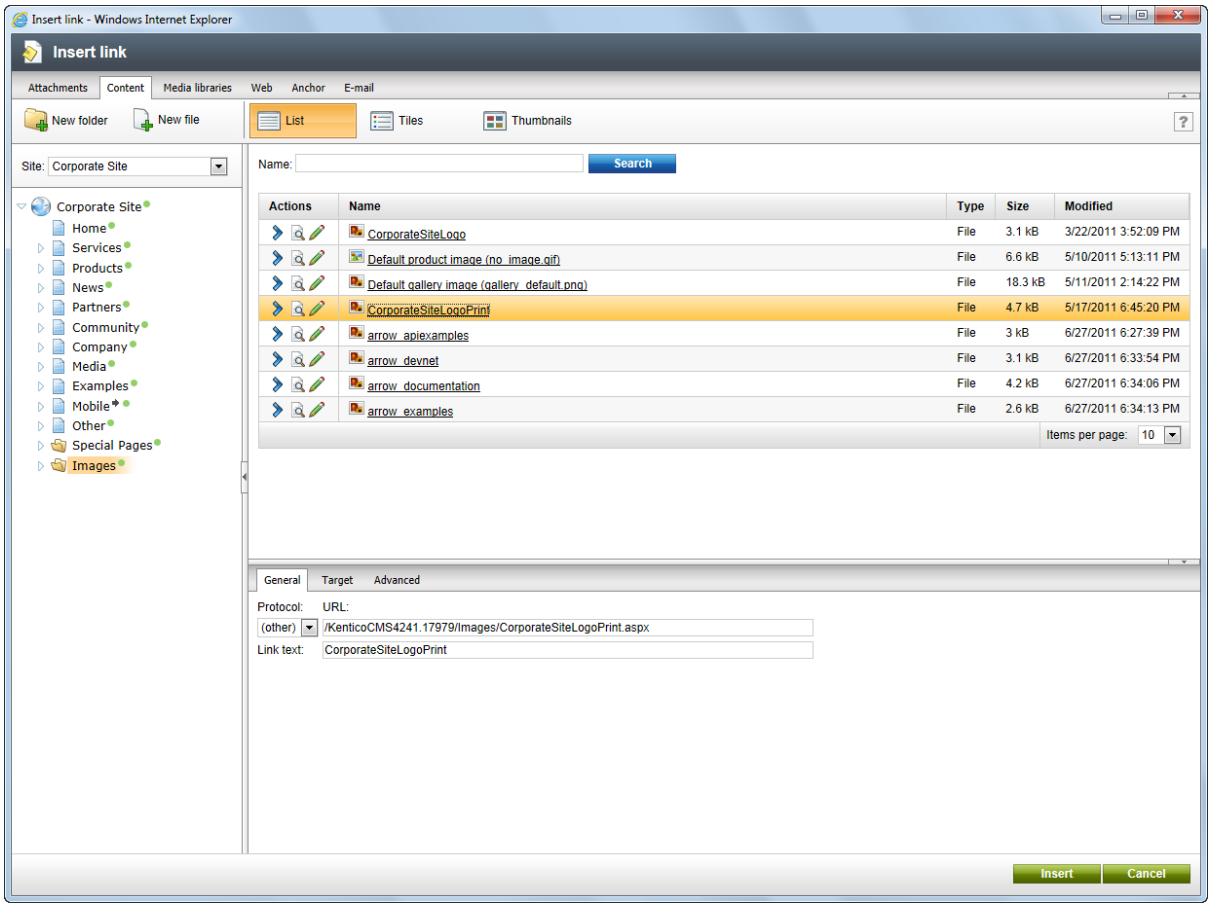


Content

Via this tab, you can insert links to any pages, documents or files within the content tree of a site. The site can be selected using the **Site** drop-down list, while its content tree will be displayed below. You can define which sites will be available. You can also define the starting alias path of the displayed content tree when defining a field in the field editor, as described in the Dialog configuration topic.

The following code sample shows what the output code looks like:

```
<a href="/3644_9682/Images-(1)/Services_webdesign.aspx">Services webdesign</a>
```

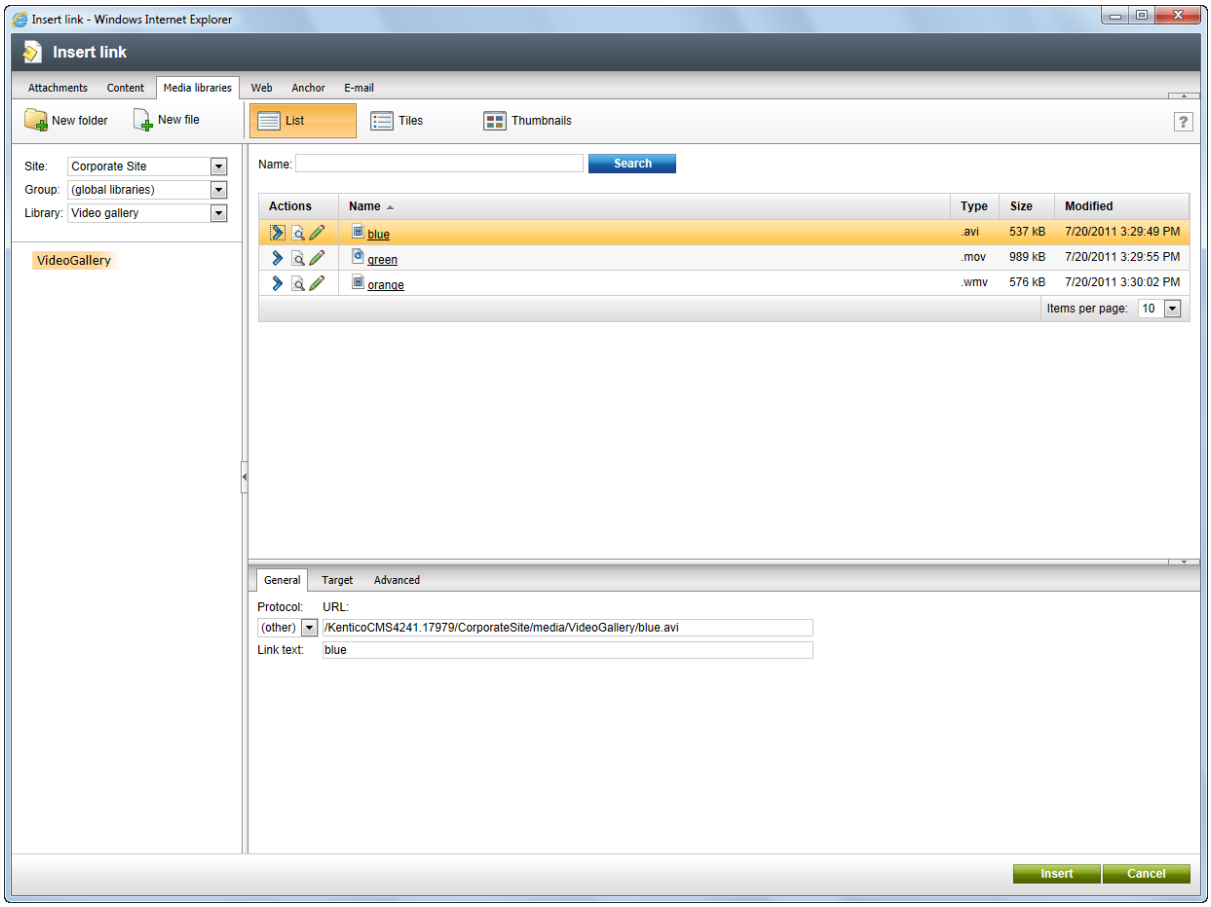


Media libraries

Via this tab, you can insert links to files stored within media libraries. Depending on the settings described in the Dialogs configuration topic, you can select a library using three drop-down lists - **Site**, **Group** and **Library** - in the top right part of the dialog.

The following code sample shows what the output code looks like:

```
<a href="/3644_9682/CorporateSite/media/CzechCities/IM002595.JPG">IM002595</a>
```

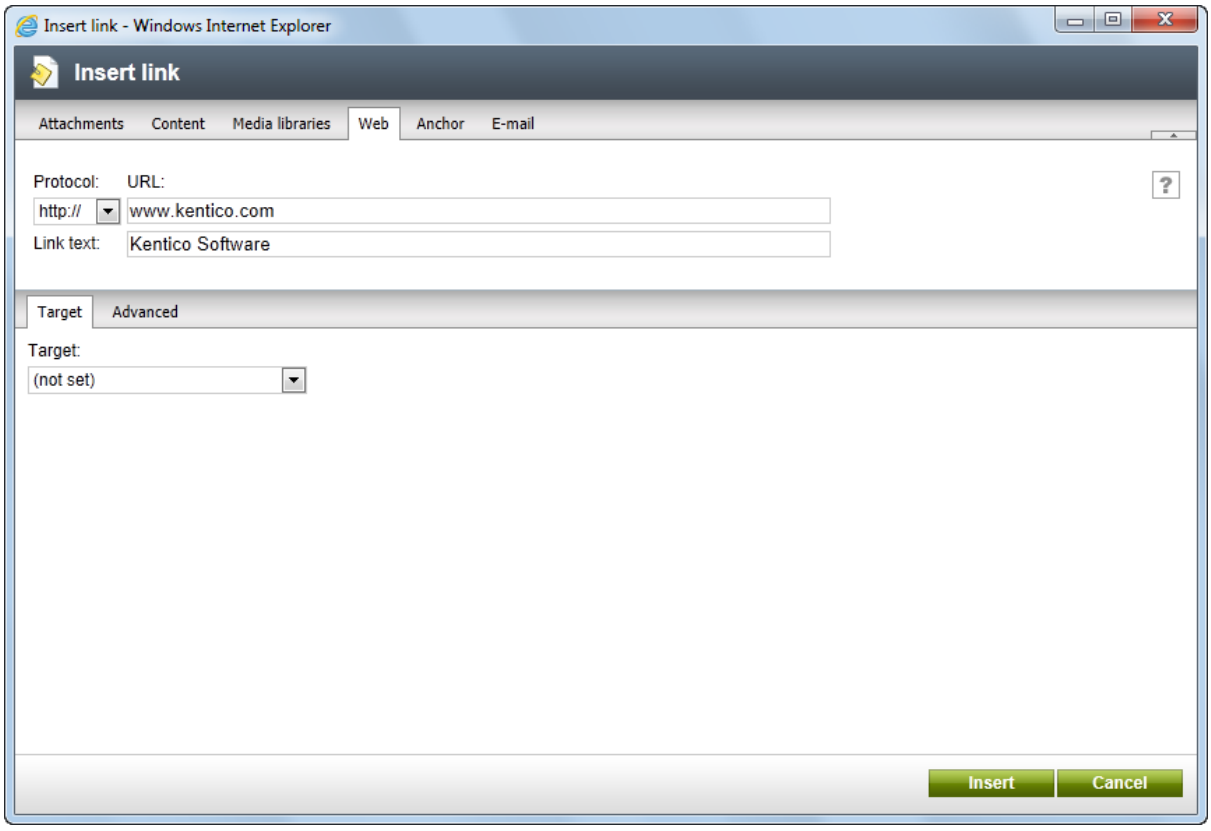


4.6.4.4 Links to the Web

On the **Web** tab, you can link any resource on the web by entering its URL. The output code is generated as an **<a>** HTML element:

```
<a href="http://www.kentico.com">Kentico Software</a>
```

On this tab, you can specify the same properties as described in the Link properties topic, with the difference that the content of the **General** tab is displayed above the two other tabs.



4.6.4.5 Links to anchors

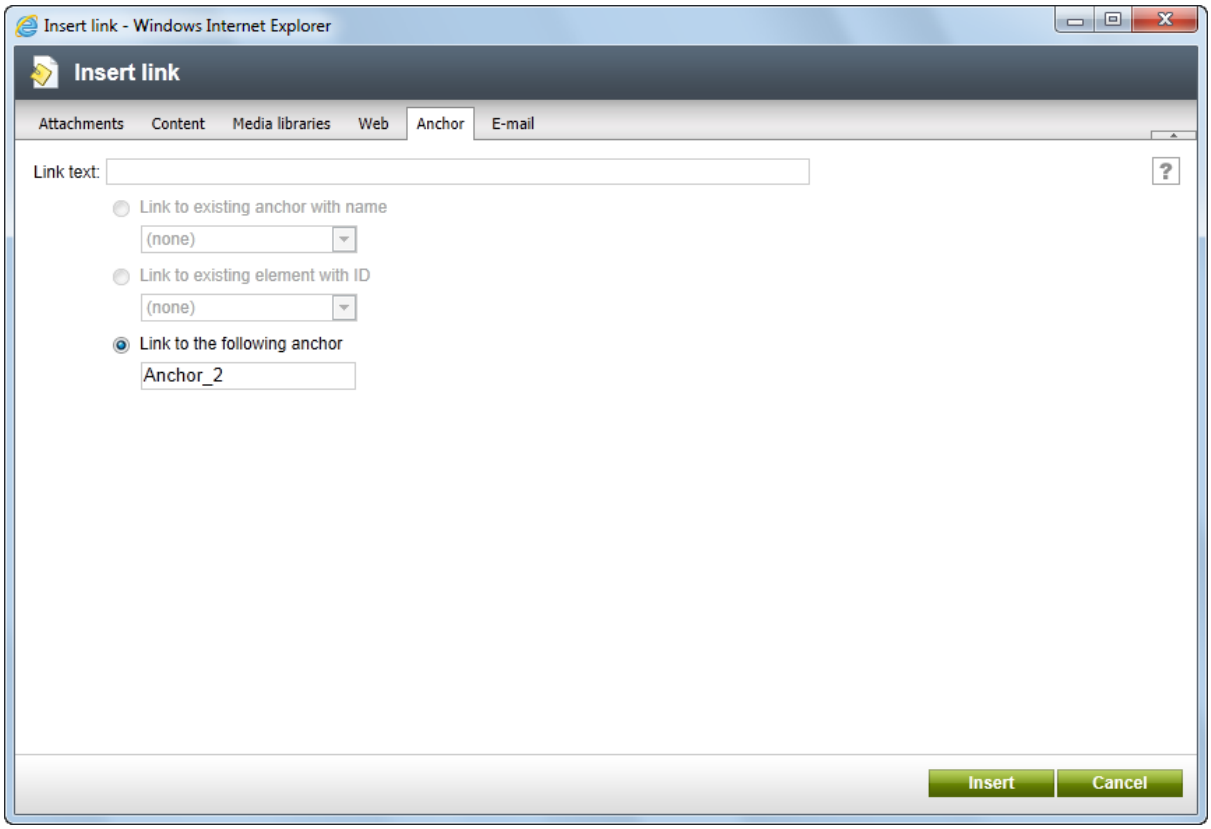
Via the **Anchor** tab, users can insert links to any anchor or any HTML element with a specified ID attribute on the current page. Anchors are <a> html elements with the **Name** attribute specified. They can be inserted using the **Anchor** ( ) icon on the WYSIWYG editor toolbar. If you link to an anchor, the page will scroll to it after clicking the link.

You have the following options on this tab:

- **Link text** - the text of the link that will appear in the text. This field is visible only if inserting a link into an empty space in the text area, i.e. if no text or object is selected.
- **Link to existing anchor with name** - if selected, you can choose an anchor from the drop-down list below as the target.
- **Link to existing element with ID** - if selected, you can choose an HTML element from the drop-down list below as the target.
- **Link to the following anchor** - if selected, you can type in the name of the target anchor or ID of the target element manually.

The output code looks like the following code sample, while the text after # is the name of the anchor or the value of the ID attribute:

```
<a href="#Anchor_2">Second chapter</a>
```



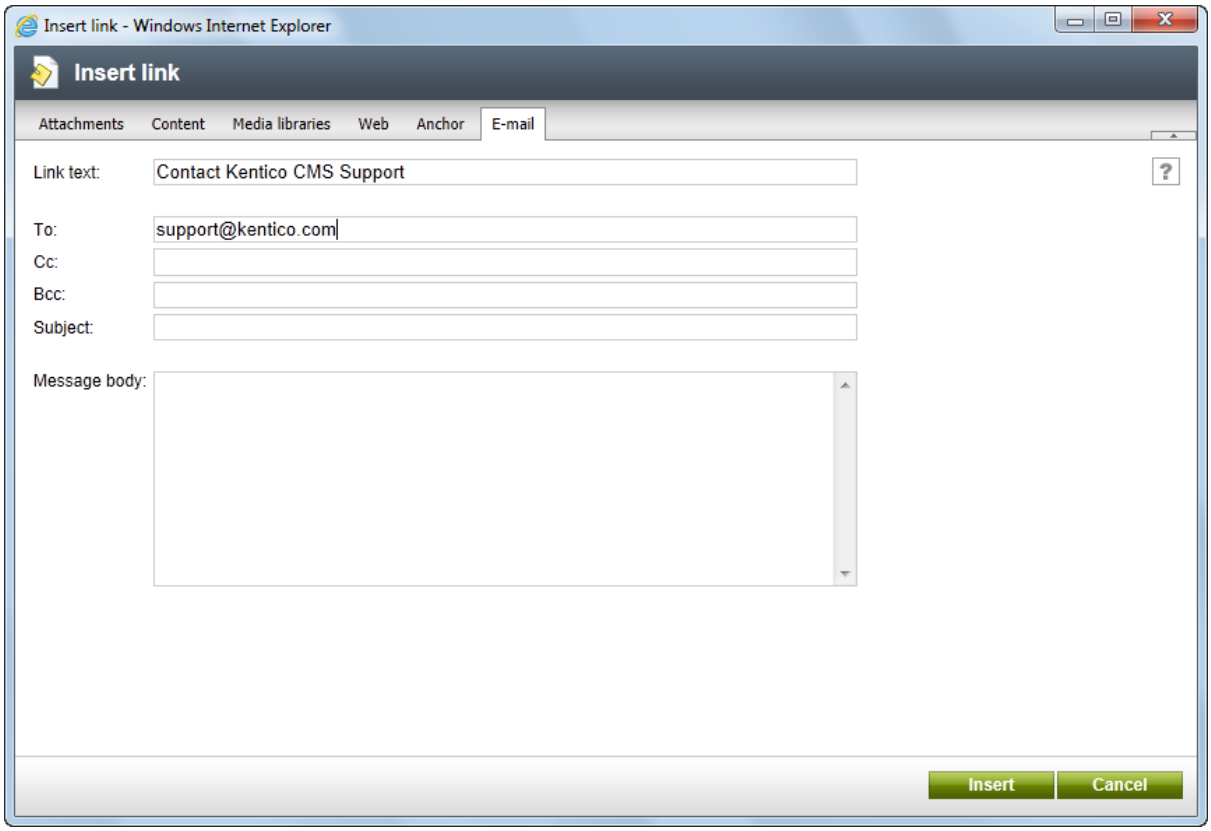
4.6.4.6 Mailto links

Via the **E-mail** tab, you can insert standard "mailto" links. After clicking such a link, a new message window of the user's e-mail program (e.g. Outlook) is opened, while some details may be pre-filled based on what is specified in the following properties:

- **Link text** - the text of the link that will appear in the text. This field is visible only when inserting a link into an empty space in the text area, i.e. when no text or object is selected.
- **To** - e-mail recipient's address; a required field. Multiple addresses can be entered divided by semicolons.
- **Cc** - e-mail copy recipient's address. Multiple addresses can be entered divided by semicolons.
- **Bcc** - e-mail blind carbon copy recipient's address. Multiple addresses can be entered divided by semicolons.
- **Subject** - the subject of the e-mail.
- **Message body** - the text of the e-mail.

The output code looks like the following code sample:

```
<a href="mailto:support@kentico.com">Contact Kentico CMS Support</a>
```



4.6.5 Insert YouTube video

Using this dialog, a video from YouTube can be easily added to a page. The dialog can be opened using the **Insert YouTube video** ( ) icon on the WYSIWYG editor toolbar, as you can see in the screenshot below:

Inserting YouTube video

The general process of inserting a YouTube video is as follows:

1. Place the cursor at the appropriate position in the text area and click the **Insert YouTube video** ( ) icon.
2. The dialog opens.
3. Insert the URL of the YouTube video into the **URL** field and click the **Refresh** ( ) icon.
4. The entered URL is checked and if it is valid, default properties are loaded and the preview displayed.
5. Specify the properties of the video according to your needs. The changes you make are reflected in the preview in the right part of the dialog.
6. When you are finished with the properties, click the **Insert** button.
7. The video is inserted into the text area.

If you click on **Go to YouTube** ( ) in the dialog, you will be redirected to the YouTube home page. This

home page will be opened in a new browser tab.

YouTube video properties

The following properties can be specified in this dialog:

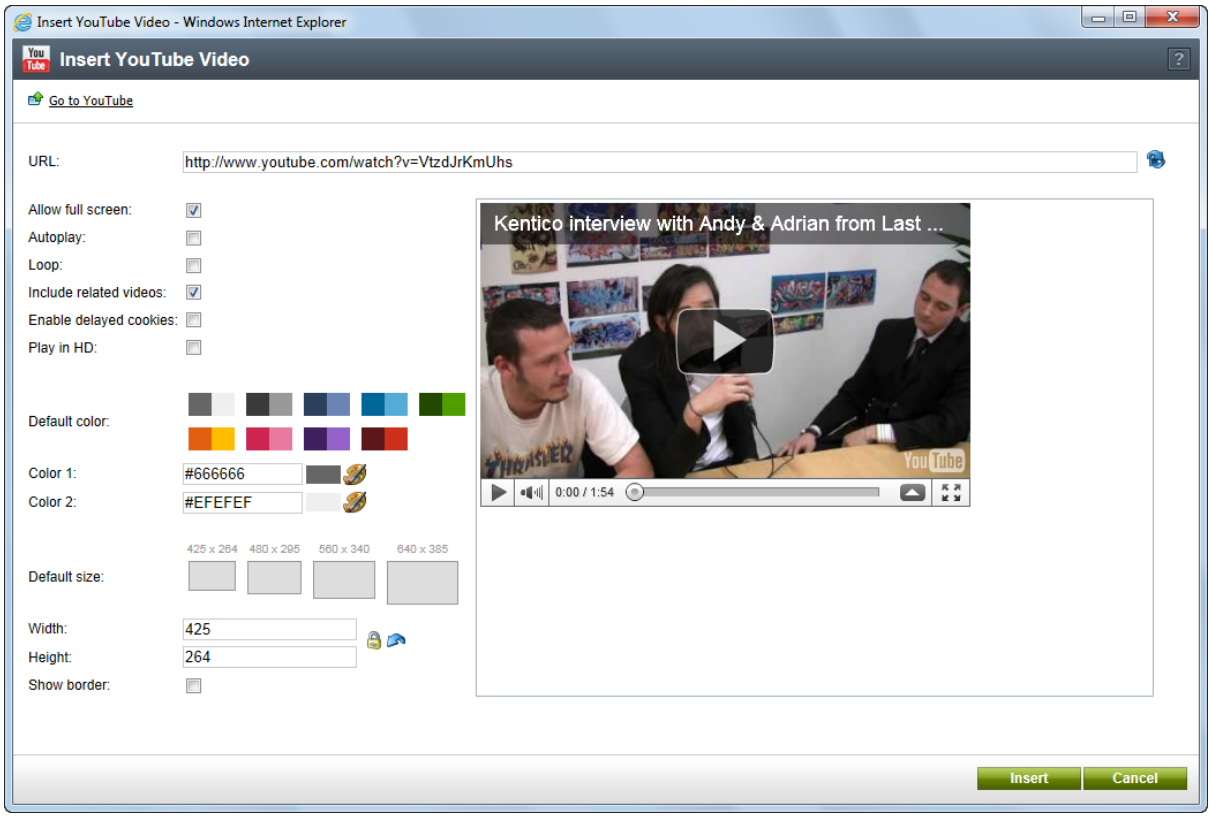
- **URL** - URL of the YouTube video. You can copy&paste it from the address line of your browser or from the **URL** field on the video page.

- **Allow full screen** - indicates if the control to switch playback to full screen will be available in the video toolbar.
- **Autoplay** - indicates if playback starts automatically when the page is loaded.
- **Loop** - indicates if playback is continuously repeated in a loop.
- **Include related videos** - indicates if related videos will be displayed when playback finishes.
- **Enable delayed cookies** - indicates if delayed cookies should be used.
- **Play in HD** - indicates if the video will be played in HD by default. The user can switch back to normal quality by pressing the **HD** button while playing the video.

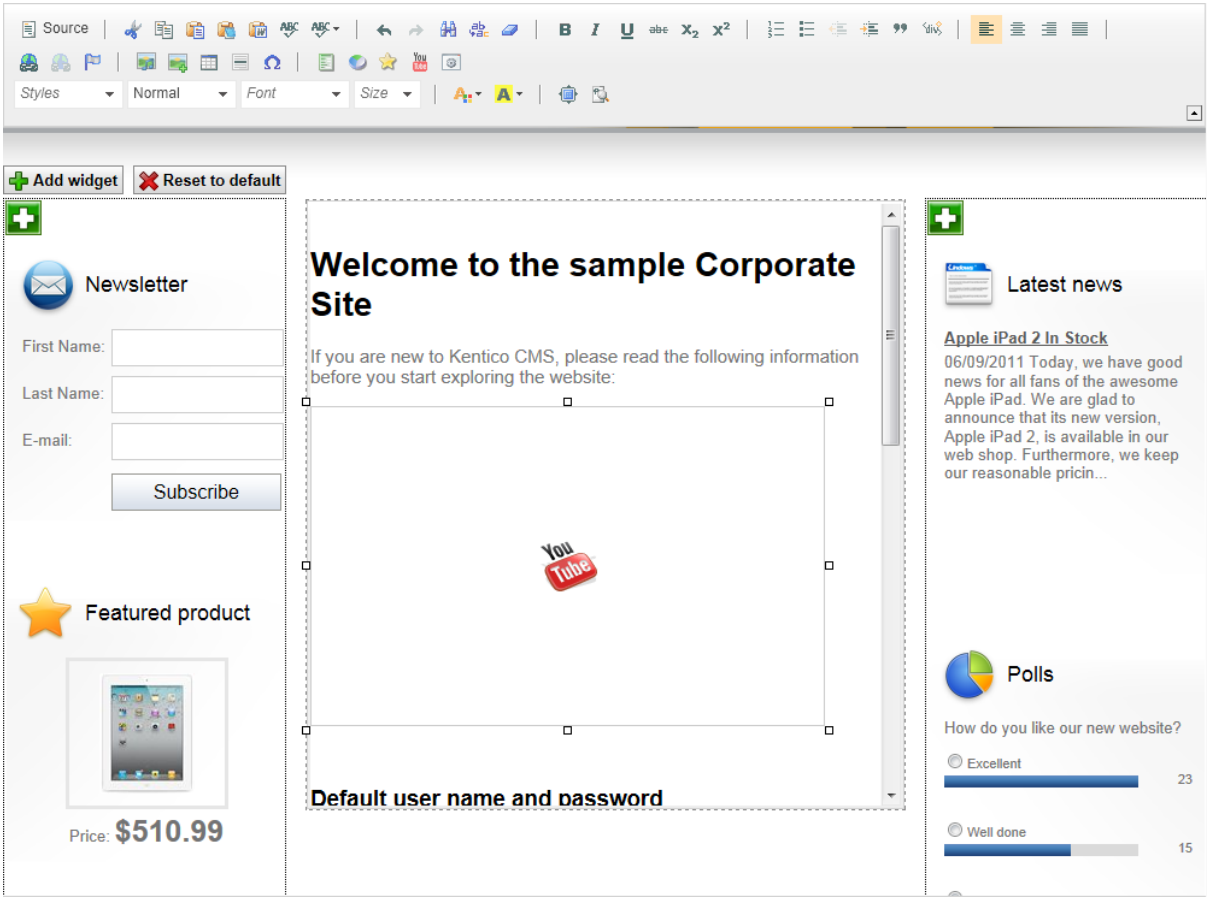
- **Default color** - you can choose one of the default color combinations, which will be used in the **Color 1** and **Color 2** properties.
- **Color 1** - the color of the border around the player, relevant only if the **Show border** property is enabled.
- **Color 2** - the color of the player toolbar.

- **Default size** - you can choose one of the default sizes of the video player, which will be used in the **Width** and **Height** properties.
- **Width** - the width of the video player.
- **Height** - the height of the video player.
- **Show border** - indicates if border should be shown around the player. Enabling this option adds 20px to both the width and height of the player.

This is what the dialog window looks like when a video is loaded:



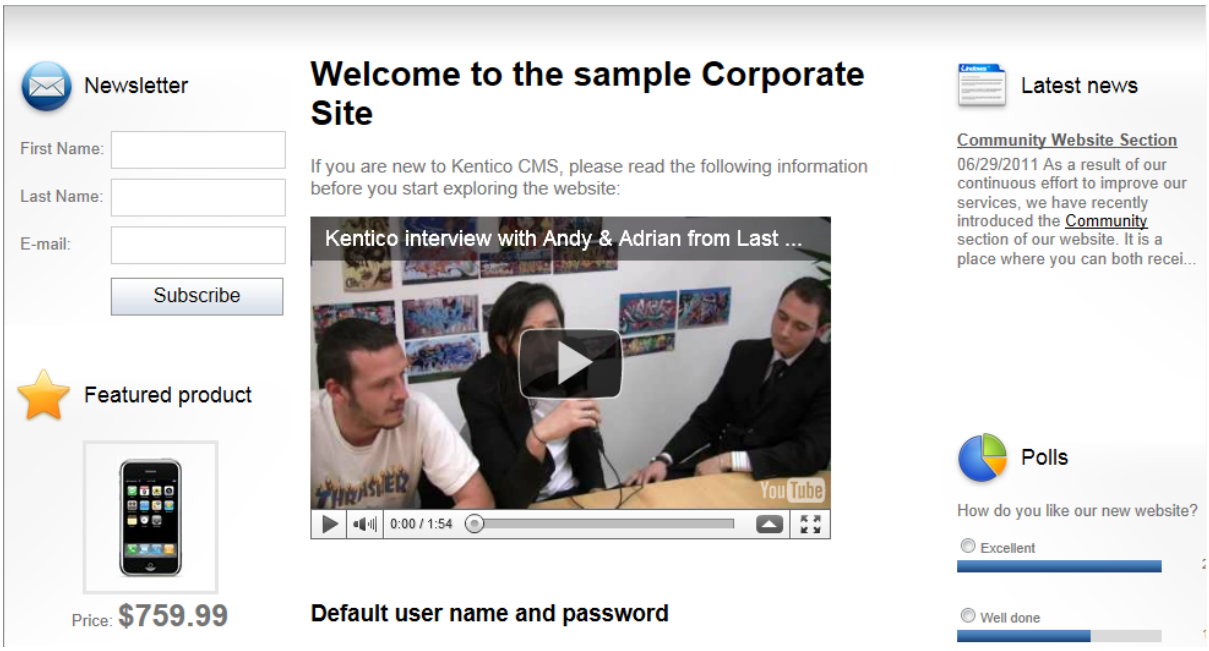
In the WYSIWYG editor, the actual video is not displayed. You can only see a box with the YouTube logo in the middle to give you information about the player size:



The YouTube video is generated as the **YouTubeVideo inline control**, as you can see in the code sample below:

```
{^YouTubeVideo|(url)http://www.youtube.com/watch?v=VtzdJrKmUhs|(width)320|(height)
198|(fs)true|(autoplay)false|(loop)false|(rel)true|(cookies)false|(border)false|
(color1)#3A3A3A|(color2)#999999^}
```

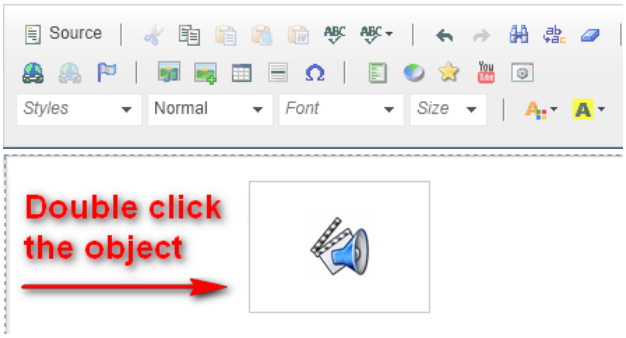
On the live site, the video is displayed in the player:



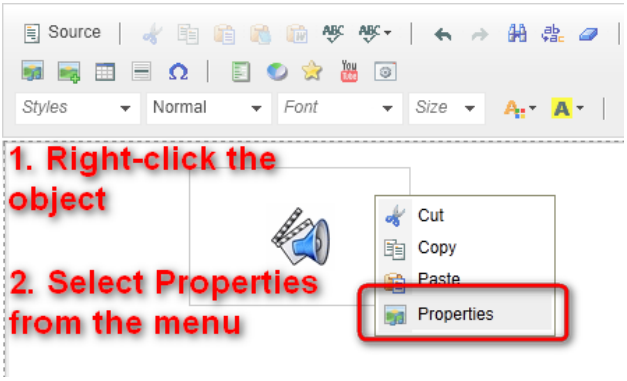
4.6.6 Editing inserted items

Properties of images, audios, videos, flash videos, YouTube videos and links that are already inserted in the WYSIWYG editor can be edited. The same dialog that was displayed when you inserted the items can be opened again the following three ways:

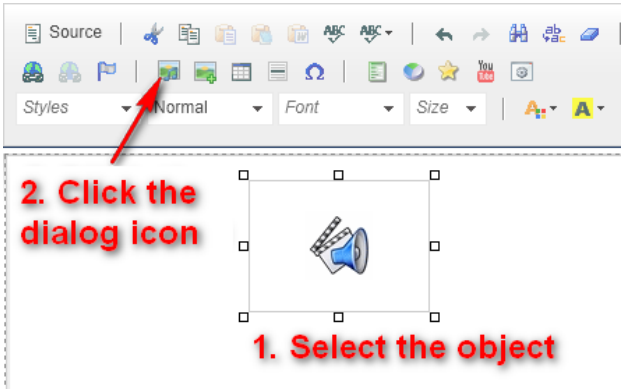
1. Double-click the item.



2. Right-click the item and choose Properties from the context menu.



3. Select the item and click the appropriate button on the WYSIWYG editor toolbar.



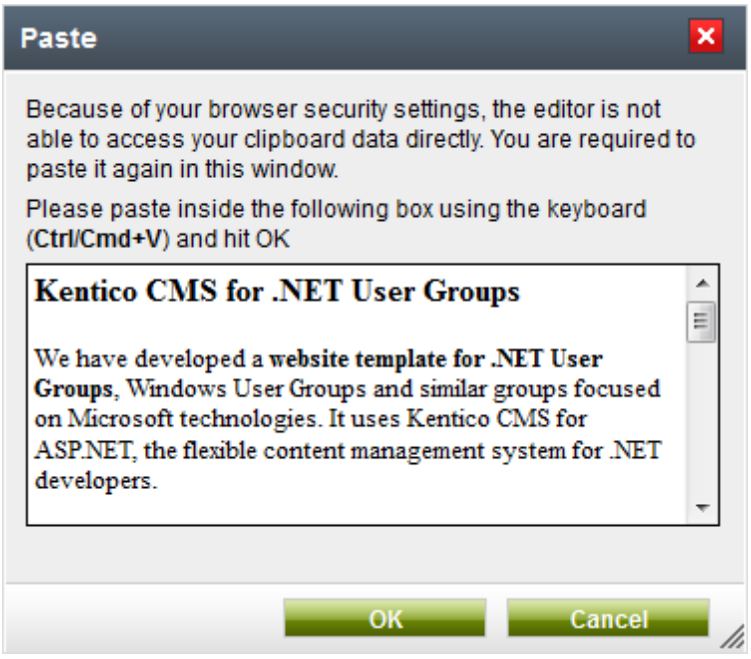
4.6.7 Copy & Paste from Microsoft Word

When copying a text from Microsoft Word, the text is encapsulated with many unnecessary tags that may break your web page design. That is why the built-in WYSIWYG editor allows you to clean the pasted text so that it contains only basic formatting.

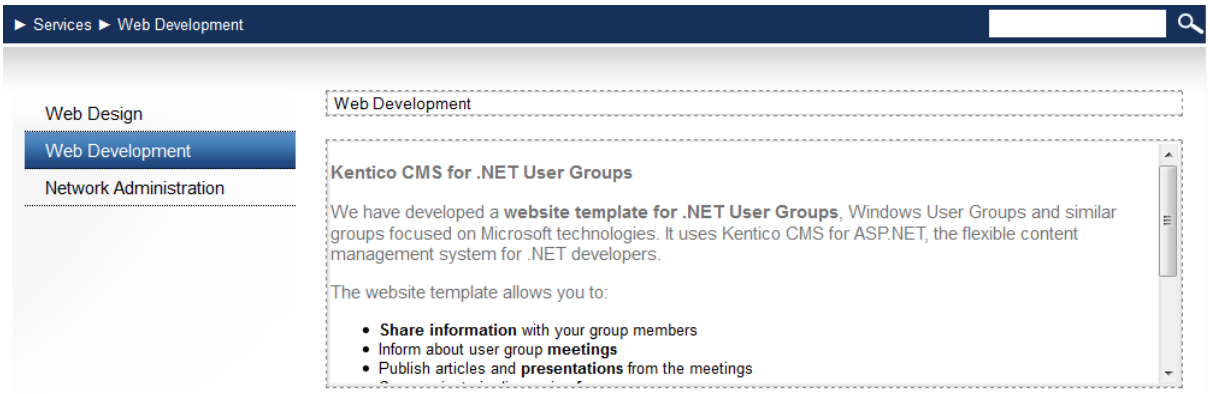
1. Select the text in a Microsoft Word document and copy it to clipboard (**CTRL+C**):

2. Now place the cursor into a Kentico CMS text area and click the **Paste from Word** ( ) icon on the WYSIWYG editor toolbar. The behaviour depends on the browser that you are using:

- If you are using Microsoft Internet Explorer, the text will be pasted into the text area automatically.
- If you are using a different browser, the **Paste from Word** dialog opens. Paste the text into the box using **CTRL+V** and check both the check boxes and click **OK**.



3. The text gets pasted into the text are and looks like this:



As you can see, the style follows your website design. However, since Word does not provide appropriate tagging information, some formatting may not be preserved and you may need to apply the design manually - e.g. the header in the sample text.

4. Finally, click **Save** ( ) to save the changes.

4.6.8 Defining custom toolbars

The WYSIWYG editor toolbar is customizable so that content editors can be prevented from using certain formatting features. This helps to keep the website design consistent and clean.

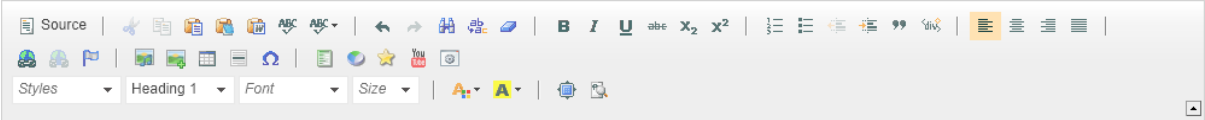
1. Defining toolbar sets

You can define toolbar sets in **<web project>\CMSAdminControls\CKeditor\config.js**.

The following code sample shows a definition of the default toolbar set containing all available icons. However, you can create your custom toolbar sets by modifying it or by using icon names contained in it.

```
config.toolbar_Full = config.toolbar_Default =
[
    ['Source', '-', 'Preview'],
    ['Cut', 'Copy', 'Paste', 'PasteText', 'PasteFromWord', '-', 'SpellChecker',
'Scayt'],
    ['Undo', 'Redo', '-', 'Find', 'Replace', '-', 'SelectAll', 'RemoveFormat'],
    ['Bold', 'Italic', 'Underline', 'Strike', '-', 'Subscript', 'Superscript'],
    ['NumberedList', 'BulletedList', '-', 'Outdent', 'Indent', 'Blockquote',
'CreateDiv'],
    ['JustifyLeft', 'JustifyCenter', 'JustifyRight', 'JustifyBlock'],
    ['InsertLink', 'Unlink', 'Anchor'],
    ['InsertImageOrMedia', 'QuicklyInsertImage', 'Table', 'HorizontalRule',
'SpecialChar', 'PageBreak'],
    ['InsertBizForms', '-', 'InsertPolls', '-', 'InsertRating', '-',
'InsertYouTubeVideo', '-', 'InsertWidget'],
    '/',
    ['Styles', 'Format', 'Font', 'FontSize'],
    ['TextColor', 'BGColor'],
    ['Maximize', 'ShowBlocks']
```

```
];
```



As you can see, the toolbar set definition consists of several arrays, e.g. *['Source', '-', 'Preview]*. This array displays a group of three icons: *Source* and *Preview*, with the first icon separated by a vertical line (defined by the *'-'* string).

If you need to insert a line break between the icon groups, use the *'/'* string.

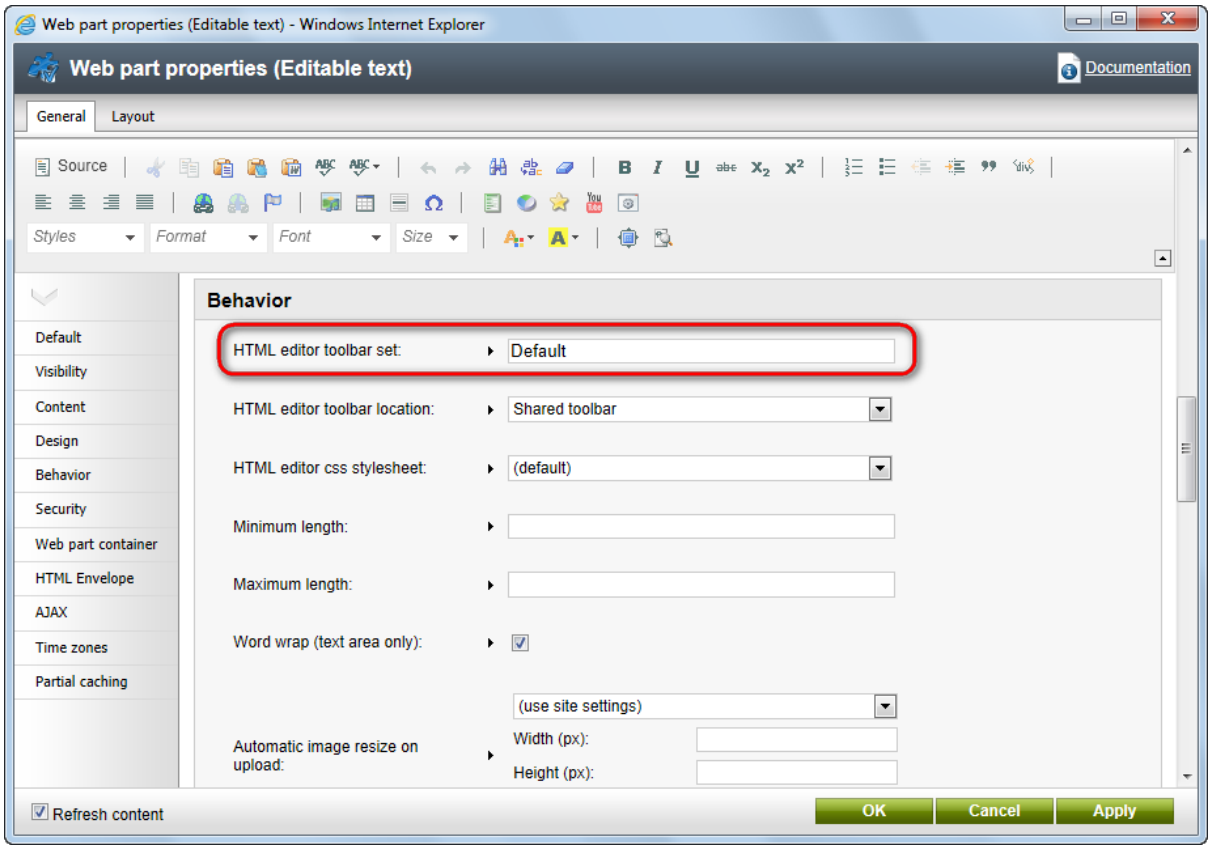
If you want to define your own toolbar set, add a command in the ***config.toolbar_ToolbarName*** format to the **config.js** file. When you have done this, save the changes you made to the file.

Every time you modify the *config.js* file (or any other file that is used for the WYSIWYG editor), you need to clear the cache of your browser so that the changes are applied.

Now you need to configure a page or document type in order for it to use your new toolbar set.

2. Assigning the toolbar set to a page

To assign the toolbar set to a page with editable regions (edited on the **Page** tab), you need to configure web parts of the *Editable region* type on the page template. Specifically, you need to set their **Toolbar set** property values to the name of your toolbar set (in the above example, it is *Default*).



Toolbar set used for structured documents

If you want to modify a toolbar set used for **structured documents** (edited on the **Form** tab), you need to set the **Toolbar set** property value of the custom field. If you share the toolbar between multiple fields, the first field toolbar set will be used.

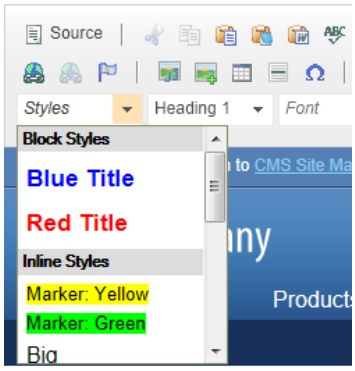
Standard toolbar sets

Kentico CMS comes shipped with several standard toolbar sets that are used by the Kentico CMS modules:

- **Full/Default** - used for structured documents.
- **Basic** - the simplest toolbar set; should not be used as default.
- **ProjectManagement** - used for project management.
- **BizForm** - used for Form module forms.
- **Forum** - used for the WYSIWYG editor in forums (if enabled).
- **Newsletter** - used for newsletters.
- **Reporting** - used for reporting.
- **SimpleEdit** - used for simple editing.
- **Invoice** - used for invoicing.
- **Group** - used for groups.
- **Widgets** - used for widgets.

4.6.9 Defining styles

The WYSIWYG editor allows you to apply a selected style to the text using the **Style** drop-down list. You can use either built-in styles or your custom styles:



The styles offered in the list are defined in the minified **<web project>\CMSAdminControls\CKeditor\plugins\styles\styles\default.js** file. For illustration purposes, the structure of the unminified file looks like this:

```
CKEDITOR.stylesSet.add( 'default',
[
{ name : 'Blue Title'      , element : 'h3', styles : { 'color' : 'Blue' } },
{ name : 'Red Title'       , element : 'h3', styles : { 'color' : 'Red' } },
{ name : 'Marker: Yellow'  , element : 'span', styles : { 'background-color' :
'Yellow' } },
{ name : 'Marker: Green'   , element : 'span', styles : { 'background-color' :
'Lime' } },
{ name : 'Big'             , element : 'big' },
{ name : 'Small'           , element : 'small' },
{ name : 'Typewriter'      , element : 'tt' },
{ name : 'Computer Code'   , element : 'code' },
{ name : 'Keyboard Phrase' , element : 'kbd' },
{ name : 'Sample Text'     , element : 'samp' },
{ name : 'Variable'        , element : 'var' },
{ name : 'Deleted Text'    , element : 'del' },
{ name : 'Inserted Text'   , element : 'ins' },
{ name : 'Cited Work'      , element : 'cite' },
{ name : 'Inline Quotation', element : 'q' },
{ name : 'Language: RTL'   , element : 'span', attributes : { 'dir' : 'rtl' } },
{ name : 'Language: LTR'   , element : 'span', attributes : { 'dir' : 'ltr' } },
{
       name : 'Image on Left',
       element : 'img',
       attributes :
       {
              'style' : 'padding: 5px; margin-right: 5px',
              'border' : '2',
              'align' : 'left'
       }
```

```
},
{
       name : 'Image on Right',
       element : 'img',
       attributes :
       {
              'style' : 'padding: 5px; margin-left: 5px',
              'border' : '2',
              'align' : 'right'
       }
},
{ name : 'Borderless Table', element : 'table', styles: { 'border-style':
'hidden', 'background-color' : '#E6E6FA' } },
{ name : 'Square Bulleted List', element : 'ul', styles : { 'list-style-type' :
'square' } }
]);
```

If you need to set more styles for one object, separate individual style definitions with a comma:

```
{ name : 'Object', element : 'element', styles : { 'style 1' : 'value' , 'style
2' : 'value' } }
```

Every style has its name and element for which it is used. The styles are offered in the drop-down list based on the current position of the cursor. If you select some image, the styles for the ***img*** element will be offered. If you select some text, the styles for the ***h3*** or ***span*** element will be offered.

If you choose to apply e.g. the ***Red Title*** style to the following HTML code:

```
We provide web development services.
```

the result will be as follows:

```
<h3 style="color: red">We provide web development services.</h3>
```

As you can see, the text was encapsulated with the ***h3*** element with the **style** attribute set to ***color: red***.

However, you may want to apply CSS class names instead of hard-coded styles. If you want to add a new style, the definition of the style needs to be inserted into the *default.js* file in front of the last square bracket. In this case, your style definition will look like this:

```
name : 'Green text',
element : 'div',
attributes :
{
       'class' : 'GreenText'
```

```
}
```

and the result will be:

```
<div class="GreenText">We provide web development services.</div>
```

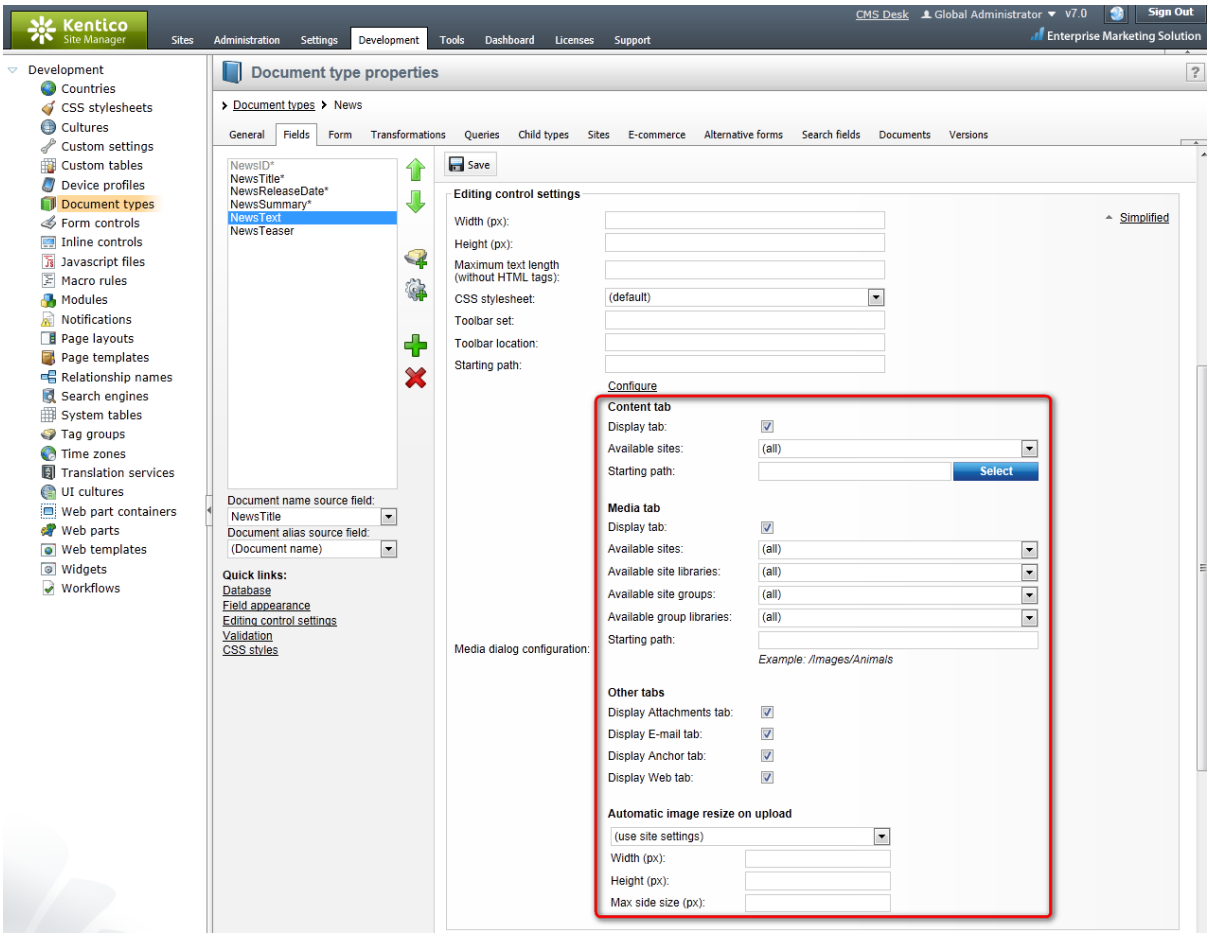
In your CSS stylesheet, you need to define the *GreenText* class name like this:

```
.GreenText { color: green; }
```

Every time you modify the *default.js* file (or any other file that is used for the WYSIWYG editor), you need to clear the cache of your browser so that the changes are applied.

4.6.10 Dialog configuration

When defining a new field that uses the **HTML area (Formatted Text)** form control, you can access advanced dialog settings by clicking the **Configure** link next to the **Media dialog configuration** parameter. Please note that it is only displayed if the **Editing control settings** of the form control are viewed in **Advanced** mode.



The following properties can be set. They are organized in categories related to the respective tabs in the dialogs:

Content tab

- **Display tab** - indicates if the **Content** tab should be displayed in the **Insert image or media** and **Insert link** dialogs.
- **Available sites** - defines which sites will be used and therefore also which content trees will be available in the dialogs.
- **Starting path** - the alias path from where the content tree should be displayed. If not relevant for the selected site, its whole content tree will be displayed.

Media tab

- **Display tab** - indicates if the **Media libraries** tab should be displayed in the **Insert image or media** and **Insert link** dialogs.
- **Available sites** - defines from which sites the media libraries can be selected.
- **Available site libraries** - defines which media libraries from the above selected site can be used.
- **Available site groups** - defines from which groups the media libraries can be selected.
- **Available group libraries** - defines which group libraries can be selected.
- **Starting path** - the path within the library from where the content should be offered. If not relevant for the selected library, the whole library will be offered.

Other tabs

- **Display Attachments tab** - indicates if the **Attachments** tab should be displayed in the **Insert image or media** and **Insert link** dialogs.
- **Display E-mail tab** - indicates if the **E-mail** tab should be displayed in the **Insert link** dialog.
- **Display Anchor tab** - indicates if the **Anchor** tab should be displayed in the **Insert link** dialog.
- **Display Web tab** - indicates if the **Web** tab should be displayed in the **Insert image or media** and **Insert link** dialogs.

Automatic image resize on upload

- ***(do not resize)*** - if selected, images will not be resized and will be pasted in full size.
- ***(use site settings)*** - if selected, images will be resized according to settings made in **Site Manager -> Settings -> System -> Files**.
- ***(use custom settings)*** - if selected, the settings specified in the three properties below will be used. The settings have the same effects as described in the Resizing images on upload topic in the **Content management system** section of this guide.
- **Width (px)** - the width of the image on upload; in pixels.
- **Height (px)** - the height of the image on upload; in pixels.
- **Max side size (px)** - if one of the sides of the image is larger than this value, the image will be resized so that its larger side dimension matches the entered value. The aspect ratio is kept and the width and height settings are not applied.

Some of the properties have the **(current)** values (e.g. current site, current library etc.). This means that the current value from the page context will be taken - e.g. if you are on a page belonging to a group and have the **Available group libraries** set to **(current group)**, the group to which the page belongs will be used.

Setup on last insertion

The dialog remembers the selection setup on last insertion for each user. This means that when a particular user opens the **Insert image or media** or **Insert link** dialog, all of the following properties are in the same state as on last insertion:

- Selected tab.
- Selected view mode.

Content tab only:

- Selected site.
- Selected path in the content tree.

Media libraries tab only

- Selected site.
- Selected group.
- Selected media library.
- Selected path within the library.

Dialogs-related settings

The following settings in the categories under **Site Manager -> Settings** are related to WYSIWYG

editor dialogs:

- **URLs and SEO -> Use permanent URLs** - if enabled, URLs of documents and document attachments will be generated in a permanent format. If disabled, friendly URLs will be used.
- **Content -> Media -> Use permanent URLs** - indicates if links in a permanent format should be used for files stored in media libraries.
- **System -> Files -> Automatic image resize on upload** - for more information on image resizing, please refer to the Resizing images on upload topic.

4.6.11 Dialog security

The following tables show which permissions are required to perform the listed actions on the particular tabs.

Media libraries tab

In this table, you can see that three types of permissions may enable users to perform the required action:

Media library module permissions and groups module permissions are those permissions that can be set in the permissions matrices in **Site Manager -> Administration -> Permissions**.

Media library permissions are those permissions that can be set on the **Security** tab when editing a particular media library in **CMS Desk -> Tools -> Media**.

Action/Permission	Media library module permissions		Groups module permissions			Media library permissions						
						File			Folder			See content
	Read	Manage	Read	Manage		Create	Delete	Modify	Create	Delete	Modify	
Global administrator in **group/global** media library												
Creating new media file	No permissions are required											
Creating new media folder												
See media library content in dialogs												
Group admin in **group** media library												
Creating new media file	No permissions are required											
Creating new media folder												
See media library content in dialogs												
Any other user in **global** media library												
Creating new media file		X			or	X						
Creating new media folder		X			or				X			
See media library content	X											X

in dialogs												
Any other user in **group** media library												
Creating new media file				X	or	X						
Creating new media folder				X	or				X			
See media library content in dialogs			X		or							X

Attachments tab - the document already exists

To perform the following actions on the **Attachments** tab, the user needs to have the appropriate document permissions for the particular document.

Action/Permission	Document persmissions			
	Read	Create / Manage	Modify	Delete
Creating new attachment (New file)			X	
Replacing source file of the attachment (Update)			X	
Deleting attachment (Delete)			X	
Moving attachment up or down (Move up/Move down)			X	
See attachments of the specified document	X			

Attachments tab - the document does not exist yet

To perform the following actions on the **Attachments** tab, the user needs to have the appropriate document permissions for the particular document.

Action/Permission	Document persmissions			
	Read	Create / Manage	Modify	Delete
Creating new attachment		X		
Replacing attachment		X		
Deleting attachment		X		
Moving attachment up or down		X		
See attachments of the specified document		X		

Content tab

To perform the following actions on the **Content** tab, the user needs to have the appropriate document permissions and the listed document types (*CMS.File* and *CMS.Folder*) need to be allowed to be created in the location.

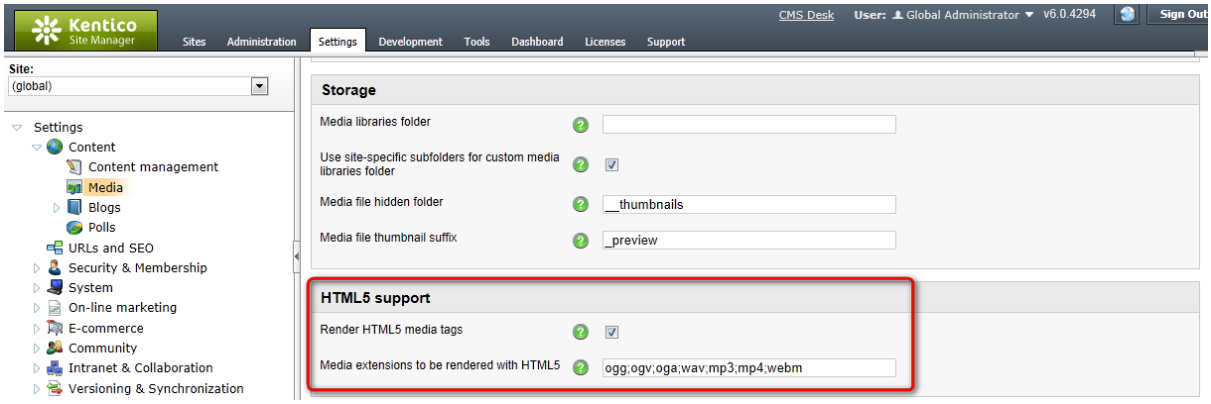
The last two columns indicate if the ***cms.file*** and ***cms.folder*** document types are **allowed child types** of the document under which they should be created. This can be set separately for each document type in **Site Manager -> Development -> Document types -> Edit ( ) the particular document type** on the **Child types** tab.

Action/Permission	Document persmissions					
	Browse tree	Create	Modify	Delete	CMS.File is allowed	CMS.Folder is allowed
		Manage				
Creating new cms.file		X			X	
Creating new folder		X				X
See child documents of the specified document	X					

4.6.12 HTML5 media tags

In **Site Manager -> Settings -> Content -> Media**, you can find the following two settings related to rendering of HTML 5 media tags:

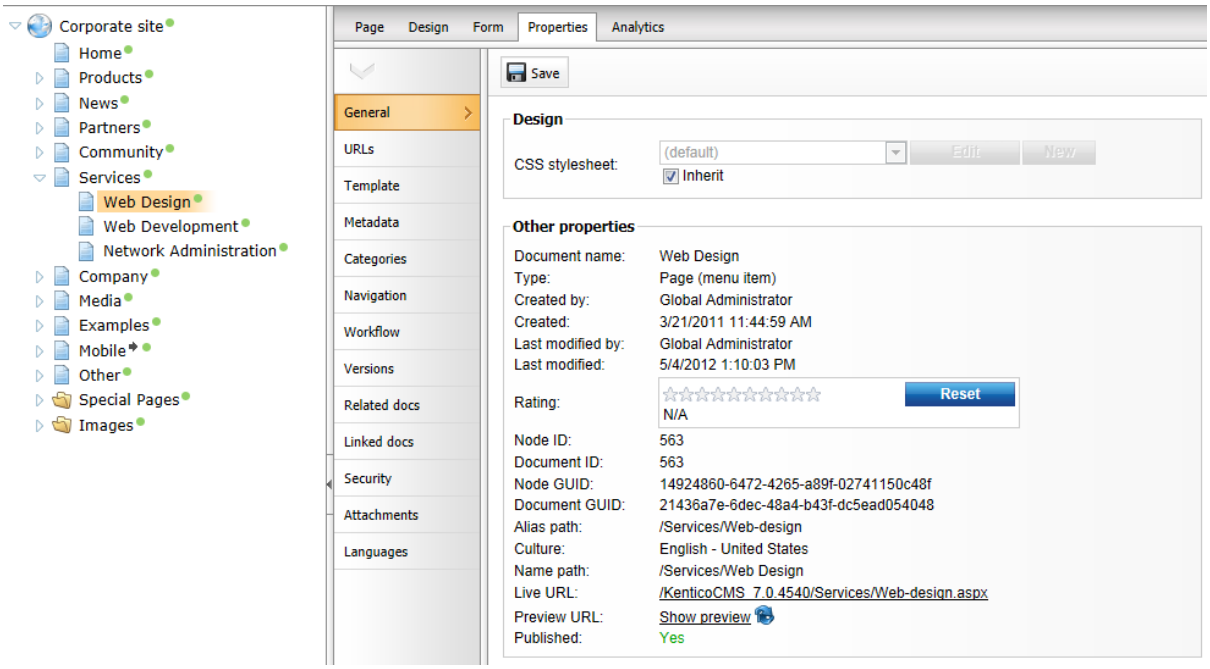
- **Render HTML5 media tags** - if enabled, the system renders HTML5 <*audio*> and <*video*> tags for media content inserted using WYSIWYG editor dialogs.
- **Media extensions to be rendered with HTML5** - specifies a list of media extensions to be rendered with HTML5 media tags. Enter a list of extensions separated by a semicolon, e.g. *.mp4; .ogg*.



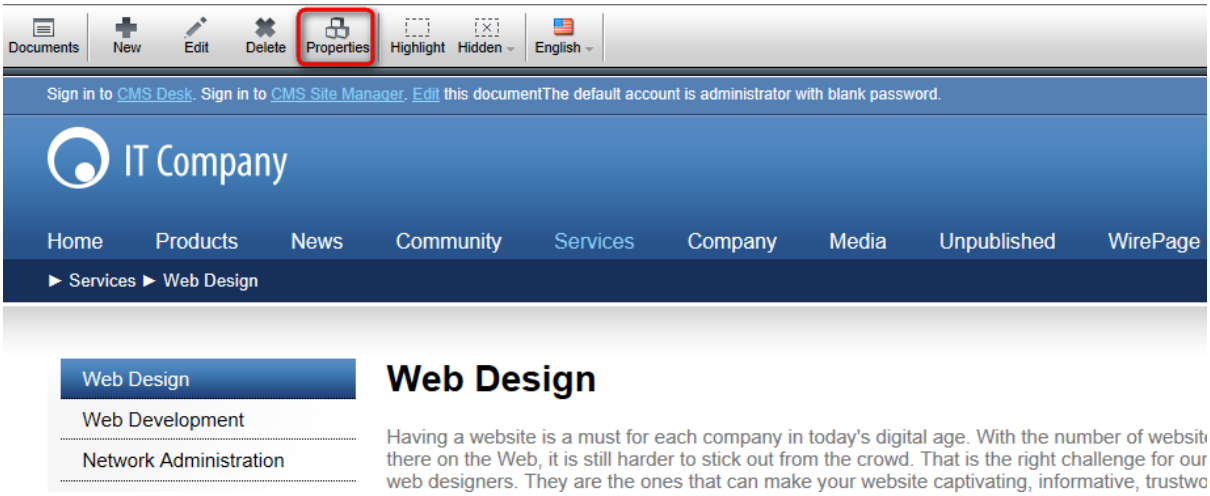
4.7 Document properties

4.7.1 Overview

You can edit document properties in **CMS Desk -> Content** after selecting a document in the content tree and opening the **Properties** tab in **Edit** mode.



The same options may also be accessed for a specific document when viewing the corresponding page in On-site editing mode. In this case, the properties can be configured by clicking the **Properties** button on the toolbar.



4.7.2 General

On this tab, you can see general information about the currently edited document:

General	
CSS Stylesheet	The CSS stylesheet used for this particular page. You can choose some particular stylesheet or use the default site stylesheet. You can also choose to inherit the stylesheet from the parent document.

	By clicking the **New** button or selecting the **(new stylesheet...)** option from the drop-down list, you can create a new CSS stylesheet. The created stylesheet will be added to the list of stylesheets in **Site Manager -> Development -> CSS stylesheets**.
Other properties	
Document name	Name of the document.
Type	Document type on which the document is based.
Created by	User who created the document.
Created	Date and time when the document was created.
Last modified by	User who last edited the document.
Last modified	Date and time when the document was last edited.
Rating	Rating of the document's content posted evaluated by site visitors. You can reset the value using the **Reset** button. See Modules -> Content rating for more details.
Node ID	Identifier of the document's node in the content tree (common for all language versions).
Document ID	Identifier of the document in the currently edited language version.
Node GUID	Globally unique identifier of the document's node in the content tree (common for all language versions).
Document GUID	Globally unique identifier of the document in the currently edited language version.
Alias path	Unique path to the document in the content tree structure.
Culture	Culture (language) of the currently edited language version of the document.
Name path	Friendly version of the document's Alias path (without blank spaces replaced with dashes, etc.).
Live URL	URL under which the document is accessible on the live site.
Preview URL	If you click the **Show preview** link, a new window will be opened, displaying the current document in **Preview** mode outside Kentico CMS user interface. The document will be displayed on a special page with a dedicated URL. The URL can be sent to a person without access to Kentico CMS user interface to let them view content that is not published yet. If you do not want the document's preview to be accessible under a link that you already sent to someone, you can generate a new preview URL for the document by clicking the **Generate preview link** ( ) button. See Editing content -> Previewing documents for more details.
Published	Indicates if the document is currently published.

Owner	
Owner	Document owner is the user responsible for its editing. This feature doesn't imply any special permissions for the owner, but it allows for easier orientation in documents. The owner can see all their documents in the **My Desk -> My Documents** section. The owner is by default set to the user who created the document. The owner can be changed only by users with the *Modify permissions* permission.
Owned by group	This field shows which group is the owner of the document (this is particularly the case of group pages). This field is mainly informative, but in some special situations, you might find it useful to change the group that the document belongs to by using the Change button.
Output cache	
Use output cache	Indicates if the system caches the full HTML output of the page. Output caching can greatly improve the performance of the page, but is not suitable for pages with dynamic content. You can inherit the output cache settings from the parent document. **Important**: Output caching must also be allowed in the website's settings. Administrators can enable output caching in **Site Manager -> Settings -> System -> Performance**.
Cache minutes	Determines how long the system keeps the output code of the page in the cache (if output caching is enabled). The page's output cache is automatically cleared if someone modifies the page. You can manually remove the page from the output cache by clicking **Clear output cache**.
Allow file system cache	Indicates if the system stores the page's output cache on the server's file system. This provides persistent caching in case of application restarts. If disabled, the application only caches the page output in its memory. If enabled, the system checks both types of cache. You can inherit the setting from the parent document. The file system cache is stored for the number of minutes specified in the **Cache output in file system (minutes)** setting in Site Manager -> Settings -> System -> Performance.
On-line marketing	
Log on-line marketing activity	Indicates if page-related on-line marketing activities (e.g. Page visit, Landing page or Content rating activity types) should be logged for this document.

	If the **Inherit** field is checked, the document loads the value of this setting from the parent document.
Advanced	
Edit regions & web parts	This button opens a window that displays a list of all **Editable text** and **Editable image web parts** (or **Editable regions** for pages based on ASPX templates) placed on the current document and allows them to be edited.
Add wireframe to this page	Inserts a wireframe schematic into the document. This can be used to plan out the page's structure and design. The document's wireframe can be defined on the **Wireframe** tab, which will become available in the main **Edit** mode menu.
Remove wireframe	If the document has a wireframe definition, this action may be used to delete it.

4.7.3 URLs

URLs tab fields:

Document alias	The unique name of the document in the given section of the website. By default, this alias is not changed when you modify the document name. The alias is used for the following purposes: 1. To define the **alias path**, which is the unique path to the document. Developers typically use the alias path in the **Path** property of certain web parts and controls. 2. For the URL of the document. The URL of the document is like: *www.kentico.com**/products/cms**.aspx* Where */products/cms* is the alias path of the document. **Forbidden characters** Some characters are forbidden in URLs and thus they are replaced by a safe character (by default, it's a dash -). You can specify the forbidden characters and the replacement character in **Site Manager -> Settings -> URLs and SEO** through the **Forbidden URL characters** and **Forbidden characters replacement** properties.
Document URL path	If you want to use a specific URL for a document that doesn't depend on its Alias path, you need to check the **Use custom URL path** box and enter a different value. You can choose from the following **Path types** for the custom URL: • **Standard URL or wildcard** - the URL will be handled by the standard Kentico CMS rewriting engine. • **Route** - the URL will be processed as an ASP.NET Route pattern. • **MVC** - requests to the URL will be handled as requests for an MVC

	page. Please see the MVC development model chapter for more information. The selection made here determines how the value of the **Path or pattern** field will be processed. For *MVC* or *Route* patterns, it is always necessary to include the URL extension at the end of the path. If you leave the path without an extension, the document will automatically be available under an extensionless URL. If MVC is selected, you must also specify the following values: • **Default controller** - the name of the MVC controller containing the action that should be performed when the alias URL is accessed, without the *Controller* part at the end. For example, if the class is called *NewsMVCController*, enter *NewsMVC*. The system first searches for the specified class in the *CMS.Controllers.<current site code name>* namespace. If it is not found there, the *CMS.Controllers.Global* namespace is searched. • **Default action** - specifies the action defined within the controller that should be performed when the document is accessed under this URL. The URL path must always start with /. When entered, the document will be accessible under two URLs. This can be useful if you need to define a short URL for a page or need to optimize the URL for search engines. The URL path is culture-specific, unlike the alias path. This means that if you use URL paths: /product (for the English version) /produkt (for the German version) Then when the user comes to /produkt.aspx, the language of the website will automatically be switched to German. The URL path may contain wildcard URLs or MVC or Route patterns. Wildcards use the following format: *{wildcard}* or *{wildcard;defaultvalue}*, where *wildcard* is the name of the wildcard and *defaultvalue* its value if the URL is used without a wildcard being entered. More information can be found in the Wildcard URLs topic.
URL extensions	Default URL extensions are defined in **Site Manager -> Settings -> URLs and SEO -> Friendly URL extensions**. If you check the **Use custom URL extensions** check box, you can define other extensions under which the document can be accessed. In case that the option is disabled, physical file extensions can also be used for **cms.file** documents. Multiple extensions can be entered divided by a semicolon. If you enter a semicolon without any extension, extensionless URLs will be allowed. Example: entering *.htm;;* for a document with a URL path set to */test*

	will allow the document to be accessed through both of the URLs below: • *<site path>/test.htm* • *<site path>/test* For this to work, you also need to to set up your IIS for handling 404 and 405 errors as described in the Installation and deployment -> Additional configuration tasks -> Custom URL extensions chapter.
Document aliases	This section displays a list of other document aliases under which the document can be accessed. One document can have an unlimited number of aliases. You can edit (✎) and delete (✖) aliases in the list or **Add new alias**. If the **Remember original URLs when moving documents setting** is enabled in **Site Manager -> Settings -> URLS and SEO**, new document aliases containing the original URL properties will automatically be created whenever the **Document URL path** or **URL extensions** properties are changed. More information about document aliases can be found in the Multiple document aliases topic in the **Development -> Page processing and URLs** chapter of this guide.

4.7.4 Template

In the Template dialog, you can specify the page template that will be used for the selected document and configure its content inheritance.

Template

The textbox in this section displays the name of the page template currently assigned to the document. The options above determine what type of template will be used:

- **Inherit from parent**
- **All culture versions use the same page template**
- **Each culture version uses its own page template**

If the first option is selected, the page template set for the parent document will automatically be loaded.

With the other two options, you can assign a specific template to the document via the **Select** button. If the document is available in multiple languages, you can choose whether all versions should share the same template or if it should be possible to specify a different one for each language. By selecting the third option, you can configure the template settings on this tab separately for each document version by switching between cultures using the language toolbar in the main menu of the **CMS Desk -> Content** tab.

Changes need to be confirmed by clicking **Save**.

Setting the template of the website's root document

The website's root document has no parent from which the template could be inherited, so the first radio button option (**No master page template**) may instead be used if you wish to have an empty root page.

In most scenarios, it is highly recommended to assign an appropriate template to the root document and use content inheritance.

Additionally, the following actions may be used to manage the document's template:

- **Save as new template** - saves the current template as a new re-usable page template. This can be useful if you wish to use an ad-hoc template for other pages.
- **Clone template as ad-hoc** - creates a copy of the current page template as a new ad-hoc template. This allows you to modify the design of the page without affecting other documents that use the original template.
- **Edit template properties** - opens a new window where you can configure the properties of the currently assigned page template.

Inherit content

In this section, you can configure how the document should inherit content from its parent pages (if it is not a master page). Select one of the following options:

Use page template settings	The inheritance is determined by the settings of the assigned page template. To manage the template's configuration, click the **Edit template properties** button and set the **Inherit content** property on the **General** tab.
Do not inherit any content	The document does not inherit any content from parent documents in the content tree.
Inherit only master page	The document inherits content from the first master page above the document in the content tree. If there are multiple master pages, the document only inherits from the closest one in the hierarchy.
Select inherited levels	You can select exactly from which parent documents the page should inherit via the checkboxes below.

4.7.5 Metadata

Here your can manage the information describing the page. You can choose to inherit the values from the parent document or enter document-specific values.

Page settings	
Page title	The page title is displayed to users in the header of their browser (or tab) when the page is viewed. It is also an important factor for Search engine optimization. Many search engines use this title for the page in their

	search result lists. The content of the field is added into the <*title*> element in the HEAD section of the page's output.
Page description	A brief description of the page and its purpose that can be used for SEO purposes and when performing searches on the website. The content of this property is added as a description meta element in the HEAD section of the page. This tag may be indexed by some search engines.
Page keywords	Adds meta keywords to the page, which may be used when searching the content of the website (for example by the built-in Smart search). Multiple keywords need to be separated by commas.
Tags	
Page tag group	Tag group that will be used for tagging this document using the Page tags parameter below.
Page tags	Enter the Tags that you want to tag the document with. When entering more than one tag into the **Page tags** field, the tags should be separated with a comma or a blank space. A combination of both in a single entry is also valid. The following examples are all valid entries for adding three tags - *tag1*, *tag2* and *tag3*: *tag1, tag2, tag3* *tag1 tag2 tag3* *tag1, tag2 tag3* In case that you are entering a tag consisting of more than one word, you should enclose it within quotation marks. Multiple long tags can also be entered and can be also divided by both blank spaces and commas: *"long tag1", tag, "long tag2"* *"long tag1" tag "long tag2"* Quotation marks can also be used for tags containing special characters that couldn't be used otherwise: *"tag@1", "tag#2", "long, strange: tag@#$"* The page tags field also has an autocompletion function implemented. This functions offers you tags from the selected tag group while you are writing.

Global Settings

You can configure a prefix for the page title, description and keywords for all documents on the website using the corresponding settings in **Site Manager -> Settings ->**

Content. Here you can set the prefixes and also the standard format of the page title.

The default page title format is: {%prefix%} - {%pagetitle_orelse_name%}

It means that the format consists of the prefix followed by the page title value. If the page title value is not set, the document name is used.

Macro expressions in metadata

You can use macros in format {%ColumnName%} to insert the values of the current document's fields into the title or other metadata properties. See Development -> Macro expressions for more information about macro expressions.

4.7.6 Categories

Categories are topic-related groups of documents. On this page, you can assign the selected document to an unlimited number of categories. There are two lists of categories:

- **My categories -** this list shows the current user's custom categories. Each of the categories can be edited () or deleted (). By clicking on **New category**, you can create your own category. The following details will be required:

Display name	Display name of the category.
Description	Text describing the category.

- **Global categories** - this list shows global categories that can be defined in **Site Manager -> Development -> Categories**. These categories can only be edited () from this page.

In case that you want to assign the selected document to some of the categories, check the category's **Select** check-box in the list and click **Save**.

See the Modules -> Categories chapter for more details.

4.7.7 Navigation

This dialog allows you to specify how the current document should be displayed in navigation elements.

Basic properties

Menu caption	The name of the document as it's displayed in navigation. It may be different compared to the document name. If no value is entered, the document name is used.
Show in navigation	Indicates if the document should be displayed by navigation web parts (in the menus). **Please note**: the document is displayed in the navigation if all of the following conditions are met:

	1. The **Show in navigation** box is checked. 2. The document is published. 3. The type of the document matches the document types configured in the appropriate navigation control (web part) - by default, only **Page (menu item)** documents are displayed in navigation. 4. If you turn on the **Check permissions** property of the menu control, the current user must be allowed to read the given document or it will not appear in the navigation controls/web parts.
Show in site map	Indicates if the document should be included in the website's Google sitemap and displayed by the Site map web part.

Menu actions

You can choose from the following menu item behavior options:

Standard behavior	The menu item redirects the user to the page as expected.
Inactive menu item	Clicking the menu item doesn't cause any action - the item is disabled. This option is useful if you need to create a menu item that cannot be clicked, but has sub-items that can be clicked. If selected, the **Redirect to URL** field appears, where you can enter the URL of a page to which users will automatically be redirected if they access the given page, e.g. through a link.
Javascript command	If you enter some JavaScript command, it will be run when this menu item is clicked. **Example**: *alert('hello');return false;*
URL redirection	The user is redirected to the target location when they try to access the given page. Redirected documents are marked with the icon in the content tree. **Example**: *http://www.domain.com* or *~/products.aspx*

Search & SEO

Exclude from search	If enabled, the current document and its content (including attachments) will be ignored by all forms of search. This affects both the Smart search module and the SQL search. Additionally, it instructs web crawlers (robots) not to index the page by adding the following meta tag into the HEAD section, which should exclude it from search engine listings: `<meta name="robots" content="noindex,nofollow" />` Enabling this setting also ensures that the page is excluded from Google sitemaps by default, but this can be overridden by individual Google Sitemap (XML Sitemap) web parts.

	Please note: Objects or documents displayed by web parts will not be excluded by enabling this property for the page that contains them.
Sitemap change frequency	Determines the value of the document's <*changefreq*> element in the website's Google sitemap, which provides a suggestion to search engines about how often the page should be indexed. This value should reflect the frequency of the page's content modifications.
Sitemap priority	Allows you to inform web crawlers which pages you consider to be the most important. The selected priority is converted to a decimal number between 0 and 1 and added as the value of the document's <*priority*> element in the website's Google sitemap. The priority is only measured in relation to other pages on the website.

Menu design

The menu item design properties are available in three alternatives:

- standard design
- mouse-over design - style used when a user hovers their mouse over the menu item,
- highlighted design - style applied if the page represented by the menu item is currently selected.

These values override the settings of individual navigation web parts (or controls) as long as their **Apply menu design** property is enabled. The CSS styles defined in the CSS stylesheet are overridden as well.

Please note: some of the following properties may not be applied to some navigation web parts or controls.

Menu item style	Style definition of the menu item. Values can be entered the same way as when defining a CSS class in a stylesheet. Sample value: *color: orange; font-size: 140%*
Menu item CSS class	CSS class defined in the website's stylesheet. Sample value: *h1*
Menu item left image	Image that will be displayed next to the menu caption on the left side. Sample values as below.
Menu item image	Image that will be displayed in the menu instead of the menu caption. You can enter either an absolute URL or a relative path in the content tree. Sample values: *http://www.domain.com/image.gif* *~/Images-(1)/icon.aspx*
Menu item right image	Image that will be displayed next to the menu caption on the right side. Sample values as above.

4.7.8 Workflow

On this tab, you can approve or reject document if it uses **workflow** and you're authorized to approve/ reject given document in the given workflow step. See the Workflow and versioning chapter for more details.

Approve section

- **Comment** - you can add a comment which will be added to the e-mail message to the editor whose work you are approving
- **Send e-mail** - if the box is checked, a message can be sent to the editor whose work is currently being approved; if not, no message is sent to the editor

Workflow steps

This table displays the workflow steps of the current document's workflow, while pointing out the current step

Workflow history

This list displays the current document's workflow steps history.

4.7.9 Versions

On this tab, you can view the previous versions of the document if it uses a workflow. You can also roll back to a previous version of the document by clicking the icon, delete ( ) the older versions or destroy the whole document history.

When you click on the icon, you can preview the particular document version.

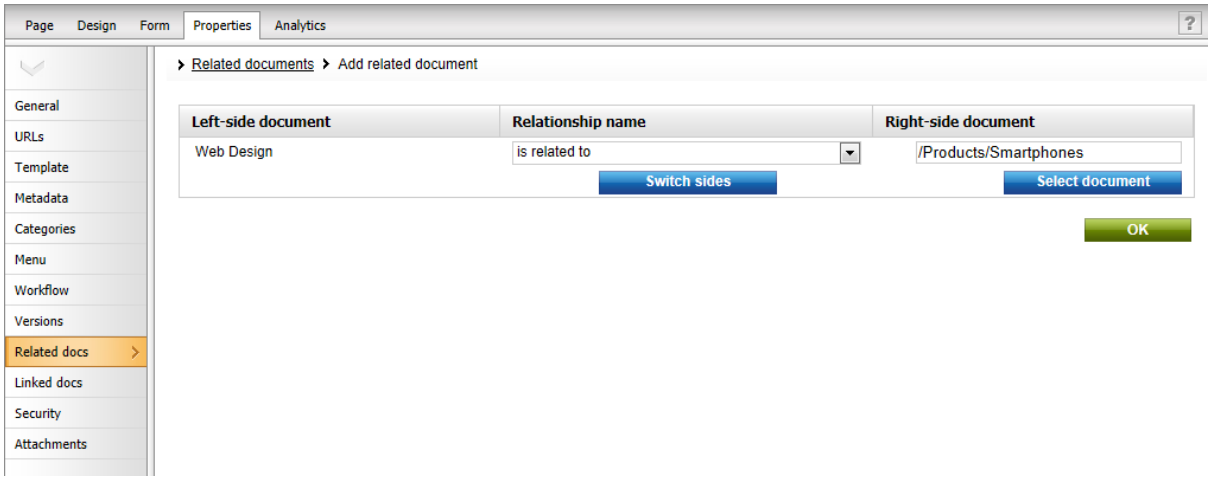
See chapter Workflow and versioning for more details.

4.7.10 Related docs

This dialog allows you to specify documents that are related to the selected document. Click ( ) **Add related document**.

Now you can:

- select the related document using the Select button
- switch the sides of the relationship
- select the type of the relationship



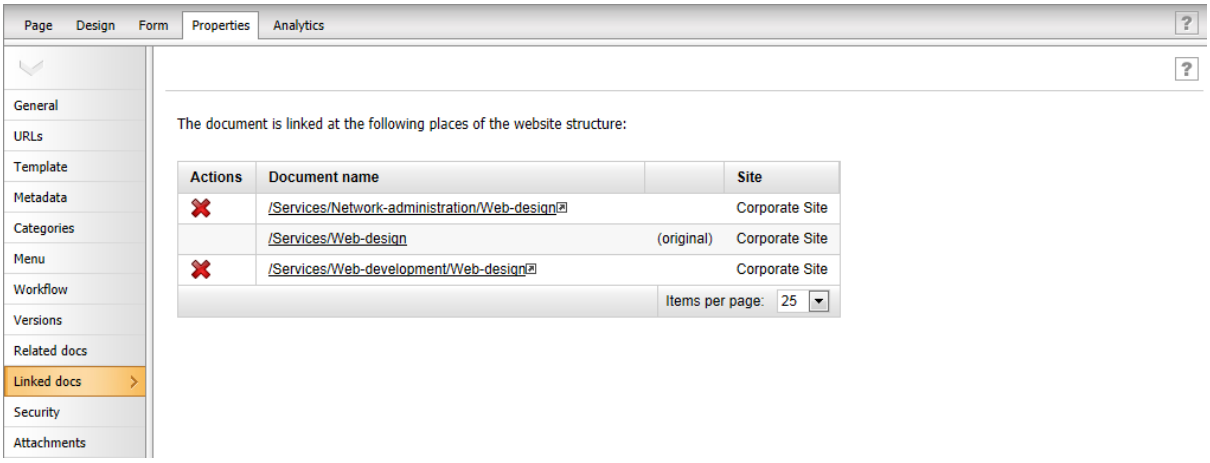
You can view related documents using controls (web parts). Refer to Kentico CMS Web parts and Controls reference -> Displaying related documents for more information.

You need to define the relationship names in **Site Manager -> Development -> Relationship names**. Only relationship names enabled for the current website are available in the drop-down list.

4.7.11 Linked docs

This dialog displays the linked documents (see Creating a linked document for explanation of the concept of linked documents).

You can delete the linked documents using the **Delete** (✖) icon or navigate to other linked documents or to the original document by clicking the link:



4.7.12 Security

This dialog is divided into two independent sections - **Permissions** and **Access**

Permissions

Here you can specify the document-level permissions for the given document. Please see the Authorization (permissions) chapter for more details.

> **Local and global permissions**
>
> Please remember that the local permissions are combined with global permissions defined for roles in the **CMS Desk -> Administration -> Permissions** dialog as described in the Authorization (permissions) chapter.

Access

Here you can configure if the page is accessible by public (anonymous) visitors or if it's only available to users who sign in with their user name and password. If you choose the option **Inherits**, the value is

inherited from the parent document.

Please read the Secured website areas chapter for more details.

You can also specify if the give page or website section **requires SSL** (HTTPS) protocol. If you set it to Yes, the CMS automatically redirects the user to the URL with https:// protocol.

Requires authentication

- **Yes** - authentication is required to access the page
- **No** - authentication is not required to access the page
- **Inherits** - value of the setting is required from the parent page

Requires SSL

This setting indicates if users who access the page should automatically be redirected to the version of the page that is secured through the SSL protocol (i.e. the target URL will use the **https** scheme). You can choose one of the following options:

- **Yes** - users attempting to access the page will always be redirected to the HTTPS version of the page's URL.
- **No** - users will not be explicitly redirected to a secured URL when they access the page, but the protocol used in the current URL will remain unchanged (i.e. if a user navigates to the page from another page that is secured, this page will also use SSL).
- **Inherits** - the settings defined for the parent document will be used.
- **Never** - users trying to access the page through a secured URL will be explicitly redirected to the unsecured version of the page.

Please note: Kentico CMS does not configure your website for SSL/HTTPS. It may only be used to automatically redirect users to the appropriate URL. You need to perform the configuration itself manually using the standard IIS settings, as described in IIS documentation or in this article: http://msdn.microsoft.com/cs-cz/magazine/cc301946(en-us).aspx

4.7.13 Attachments

On this tab, files can be attached to the document.

On this page, you can see a list of the currently selected document's unsorted attachments. Grouped attachments can be added to the document on the Form tab in case that the appropriate field(s) are defined for the particular document type.

You can upload **New attachment** ( ). You can also perform the following actions with the attachments in the list:

- Using the **Delete** ( ) icon, you can remove the attachment from the document.
- Using the **Move up** ( ) and **Move down** ( ) icons, you can re-order the attachments. The order is stored in the **AttachmentOrder** property of each attachment. You can enter AttachmentOrder into the ORDER BY expression property of a displaying web part to have the attachments ordered accordingly.

Please note

The order of attachments is **not versioned** with documents' workflow. This means that if you change the order of attachments in one version of a document, the order is changed in all other versions too.

- Images have also the **Edit** () icon available. This icon opens the image in the built-in image editor.
- Using the **Update** () icon, you can replace the original attachment with a new one.
- After clicking an attachment's name, the attachment will be opened.

For more information on the Attachments module, please refer to the Document attachments chapter of this guide.

4.7.14 Languages

This tab only appears on multilingual sites (i.e. those that have more than one content culture assigned). See the Multilingual content chapter for more details.

Here you can view the language versions of the selected document. A document can have one of the following statuses for each language:

- **Translated** - the document is translated and up-to-date. The actual language of the document's content has no effect on the status, the system only checks whether the language version exists.
- **Outdated** - the language version exists for the document, but is outdated. The system considers language versions to be outdated if the default language version of the document has been modified (or published when using workflow) more recently.
- **Not available** - the document does not exist in the given language.
- **Waiting for translation** - the document exists in the given language, but its content has not been translated yet. It is currently submitted to a translation service. You can manage the translation submission in *CMS Desk -> Tools -> Translations*.

You can use the following actions for individual language versions:

- **Edit culture version** - allows you to edit the fields for the given language version of the document (i.e. opens the **Edit -> Form** tab in CMS Desk).
- **Create new culture version** - shown for languages in which the document is not available. Clicking the action creates a new version of the document in the given language.
- **Translate document** - opens a dialog where you can submit the document for translation to the given language, using a specified Translation service.

4.8 File management

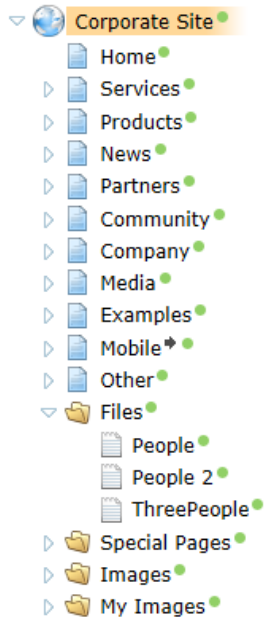
4.8.1 File management overview

Kentico CMS allows you to upload files (such as GIF, JPG, SWF, PDF, XLS, DOC, etc.) to the Kentico CMS database or file system and manage them as any other content.

File management approaches

There are four types of files from the file management perspective:

- **CMS.File documents** - these files are uploaded by the content editors as new *CMS.File* documents into the content tree. You will typically use this type for files that are used as part of unstructured documents, such as document links or images inserted into editable regions of pages. It is advisable to have files stored within folders (*CMS.Folder* document type). You can also use the File import module when uploading multiple files. The Document library module allows convenient management of CMS.File documents on the live site.



- **Document attachments** - these files are stored as a part of a structured document and their life cycle is also bound with the document (including workflow and versioning). You can have an unlimited number of files attached to a document. A detailed description of the whole concept and examples of typical usage can be found in the Document attachments sub-chapter.

- **Media Libraries** - the Media Libraries module allows storage of large amounts of files, while large file sizes are supported; the whole module and its typical usage is described in Modules -> Media libraries.

- **Unmanaged files** - these files are part of the website theme and they should be stored in the <*web project*>*app_themes*\<*theme*>*images* folder on the disk. They usually include images and Flash animations used throughout the site. These files are not managed by the CMS system.

For performance optimization tips related to file management, please refer to the Development -> Caching and performance -> File management and performance chapter of this guide.

4.8.2 Document attachments

4.8.2.1 Overview

Kentico CMS allows users to attach files to the website's documents. This concept greatly simplifies the way you work with files. Each document can have any number of attached files, which is a great step forward compared to the previous use of the File field, which allowed only one uploaded file per one

defined File field.

Attachments are directly bound to a document's life cycle. So if a document gets published, its attachments get published too. If you delete a document, its attachments will also be deleted.

There are two types of attachments:

- **Unsorted -** added via the **Properties -> Attachments** tab; using this approach, you can easily add attachments to any document; all attachments added this way are taken as **one group** and displayed together using the web parts
- **Grouped** - added by defining a 'Document attachments' field for a document type and then uploading a file on a document's **Form** tab; for each Document attachments field, you can have an unlimited number of files attached; you can also define more fields of this type to have several groups of attachments which can be displayed separately on the live site
- **File field** - another type of field that can be used for file upload via the **Form** tab; the difference from Grouped attachments is that only one file can be uploaded into one File field

The attachments can be displayed with the document using one of the following web parts:

- **Attachment image gallery** - using the default transformation, the web part displays thumbnails of the document's image attachments. After a thumbnail is clicked, a lightbox will display the attachment in full size. For non-image attachments, a file type icon will be displayed instead. If clicked, the file will be offered for download. Of course, this behaviour can be modified by defining a custom transformation.
- **Document attachments** - displays a list of document's attachments. After an attachment is clicked, it is opened in a new window or offered for download in case that it is not an image.
- **Attachments data source** - data source web part for providing attachments to a connected displaying web part; see also Using Data source web parts.

More information about the web parts can be found in the Available web parts topic.

You can also display a document's unsorted attachments using the following two inline widgets that can be added via the **Editable text** web part:

- **Attachment image gallery** - displays thumbnails of the document's unsorted attachments in a gallery. After a thumbnail is clicked, the attachment will be displayed in a lightbox.
- **Document attachments** - displays the document's unsorted attachments. After an attachment is clicked, it will be displayed on a new page if it is an image and non-image attachments will be offered for download.

More information about the inline widgets can be found in the Available inline widgets topic.

Settings which can be configured in **Site Manager -> Settings -> System -> Files** are applied to document attachments. You can learn about available settings in the Files-related settings chapter.

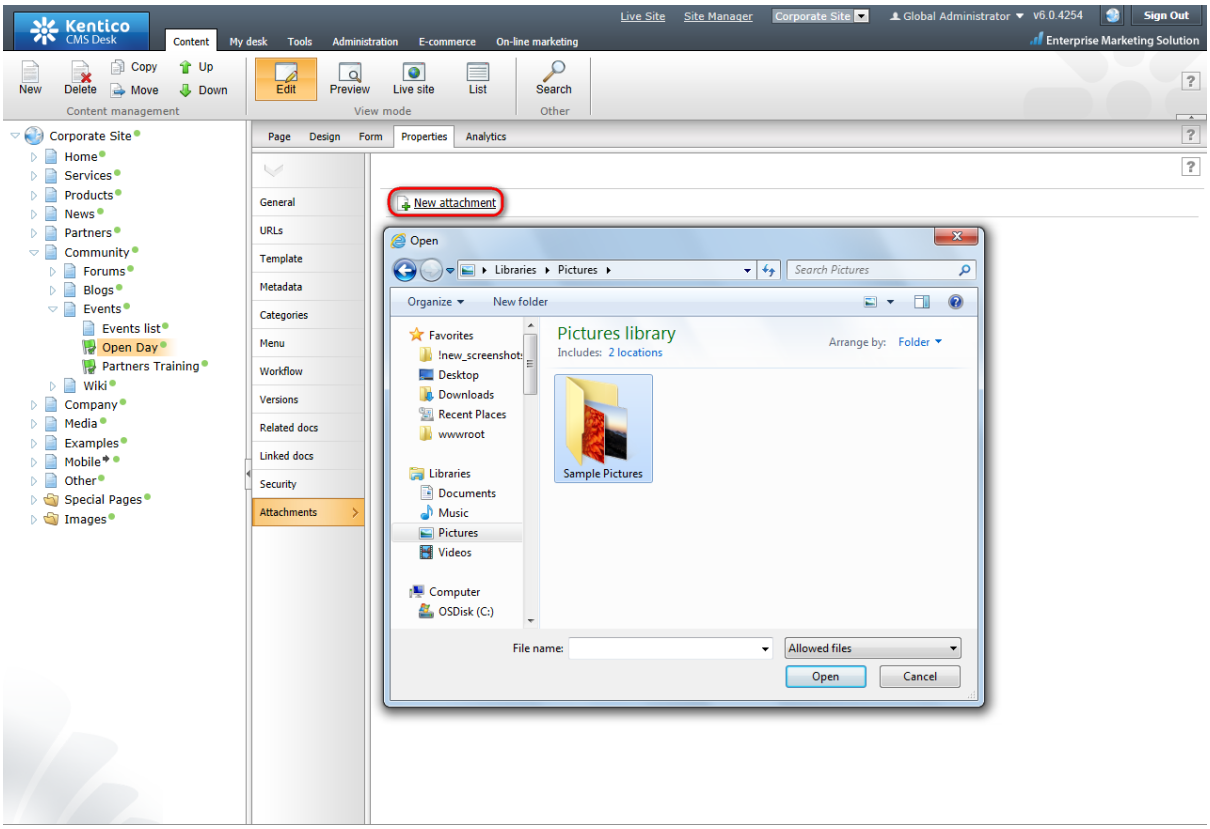
4.8.2.2 Example: Unsorted attachments

In the following example, you will learn how to add attachments to a document via the **Properties -> Attachments** tab and display them on the live site. We will use the **Events** section of the sample **Corporate Site**. First, we will upload the attachments to some of the events in the section. Then we will add the **Attachments image gallery** web part to the **Events** page, which will display the attachments for each displayed event.

1. Adding attachments to a document this way is quite simple. First, you need to select a document from the content tree and switch to its **Properties -> Attachments** tab.

2. On the tab, click the **New attachment** ( ). The familiar **Choose file** dialog will be displayed. Choose any file from your local drive and click the **Open** button.

Please note: The mouse pointer doesn't change when you hover the **New attachment** ( ). This is by design and you needn't be worried about it.



3. Repeat the same procedure so that you have a few files attached to at least one document. Preferably, include some images among them. The document's attachments tab should look similarly to the screenshot below after the attachments' upload. Note that if you hover an image in the list, its thumbnail is displayed as in the screenshot.

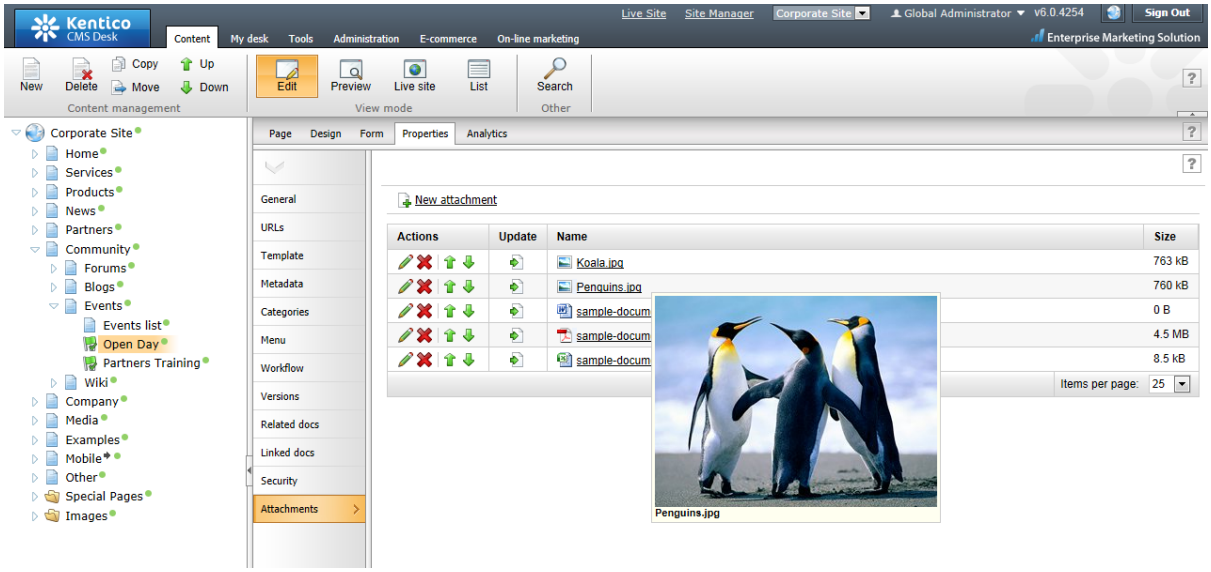
In the list of attachments, you can do several things with each attachment:

- **Delete** ( ) - removes the attachment from the document.
- **Move up** ( ) and **Move down** ( ) - re-orders the attachments. The order is stored in the **AttachmentOrder** property of each attachment. You can enter *AttachmentOrder* into the **ORDER BY expression** property of a displaying web part to have the attachments ordered accordingly.

> **Please note**
>
> The order of attachments is **not versioned** with documents' workflow. This means

that if you change the order of attachments in one version of a document, the order is changed in all other versions too.

- **Edit** ( ) - if the attachment is an image, clicking the icon opens it in the built-in image editor. If the attachment is not an image, the metadata editor is opened after clicking the icon.
- **Edit in client application ( )** - is displayed only when WebDAV integration is enabled and only next to file types supported by WebDAV. Clicking the icon opens the document for editing in an external application installed on the client computer (the application must support WebDAV for this to work). Saving the document in the client application updates the attachment with the new version automatically.
- **Update** ( ) - enables you to replace the original attachment with a new one.
- After clicking an attachment's name, the attachment is opened.

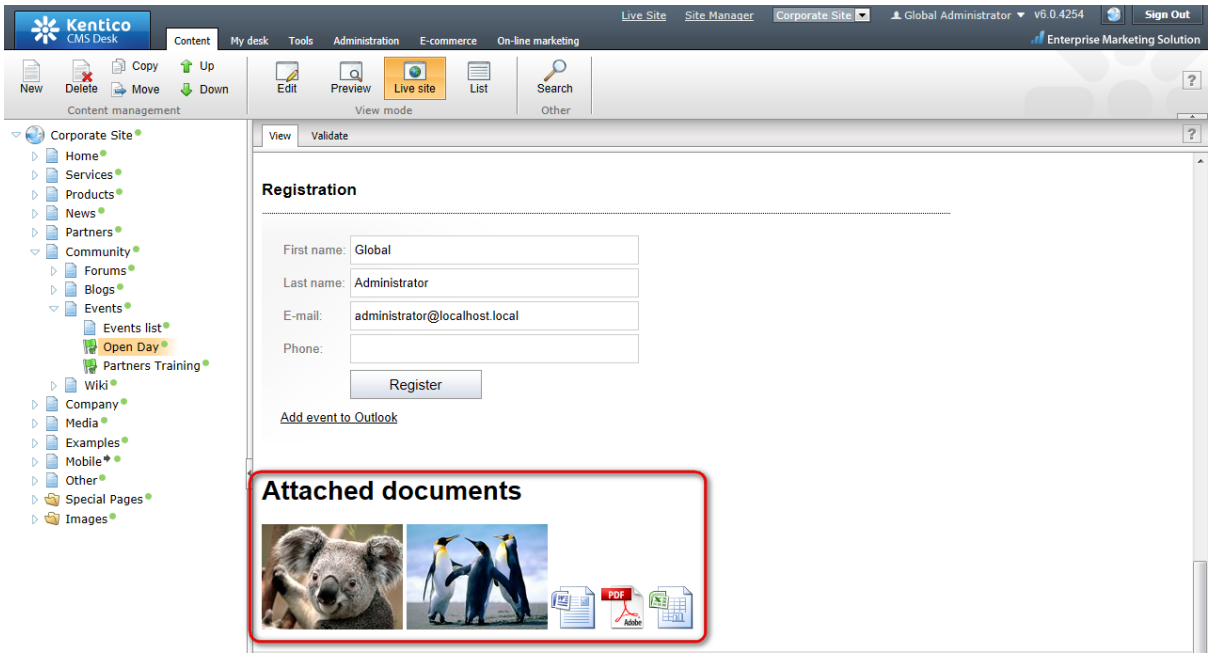


4. Now that we have the attachments uploaded, we can display them on the live site. The events are displayed by a repeater on the **Events** page, so we will have to add the **Document attachments** web part to this page.

Select the **Events** page from the content tree and switch to the **Design** tab. Click the **Add web part** ( ) icon of the **Main zone** web part zone and choose the **Attachments -> Attachments image gallery** web part. You do not need to set any web part properties in order for the web part to display the attachments. However, you can add some heading via the **Content before** property:

- **Content before**: <h1>Attached documents</h1>

Click **OK**. If you switch to the live site now, you should see its attachments' thumbnails as in the screenshot below.

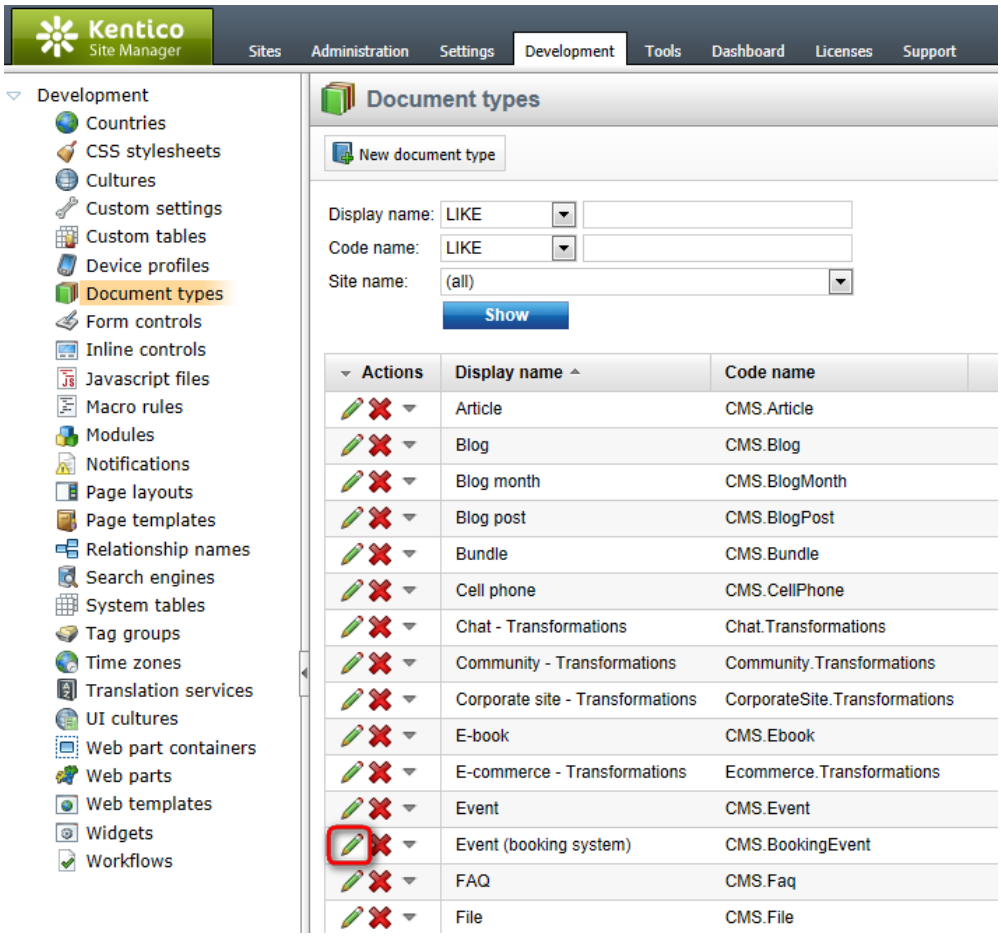


4.8.2.3 Example: Grouped attachments

In the previous example, you learned how to add attachments to a document via the **Properties -> Attachments** tab. This is the easier approach, however, in some situations, you may want to have more groups of attachments.

In this example, you will learn how to add two **Document attachments** fields to a document type and how to use them on the live site. Again, we will use the **Events** section of the sample **Corporate Site**. So if you completed the previous example right before, please delete the **Document image gallery** web part from **Main zone** in order to get the page to the original appearance.

1. The events' document type is **CMS.BookingEvent**. We will need to add the fields to it. Go to **Site Manager -> Development -> Document types** and choose to **Edit** (✎) the **Event (booking system)** document type.

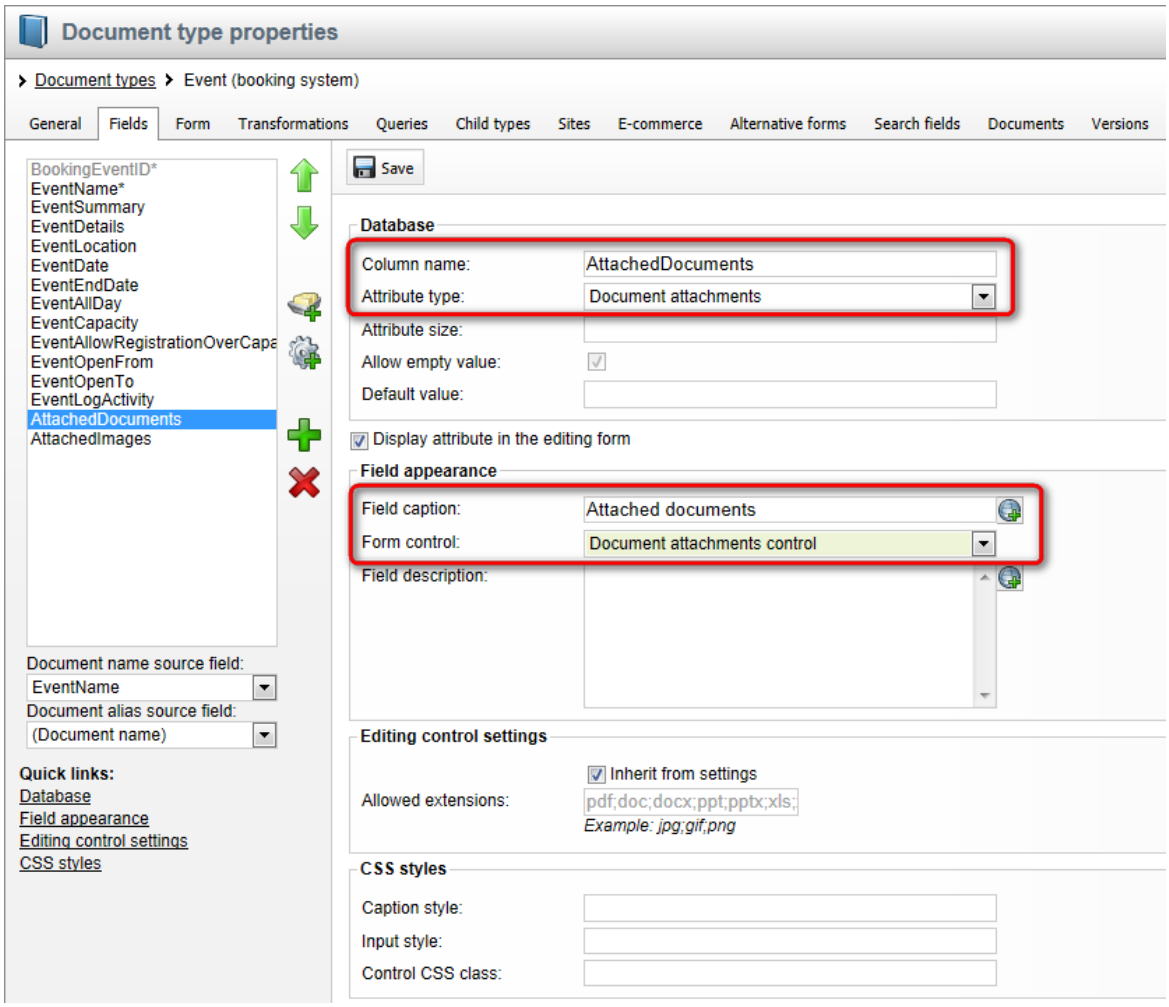


2. Switch to the **Fields** tab. Use the **New attribute** (+) icon to add the following two fields. For each attribute, set only the following properties and leave the default values for the rest. Click the **Save** button to confirm the creation of each field.

- **Column name**: AttachedImages
- **Attribute type**: Document attachments
- **Field caption**: Attached images
- **Form control**: Document attachments control

- **Column name**: AttachedDocuments
- **Attribute type**: Document attachments
- **Field caption**: Attached documents
- **Form control**: Document attachments control

Finally, use the **Move up** (↑) and **Move down** (↓) arrows to move the two attributes to the bottom of the list so that they will be displayed at the bottom of the **Form** tab.



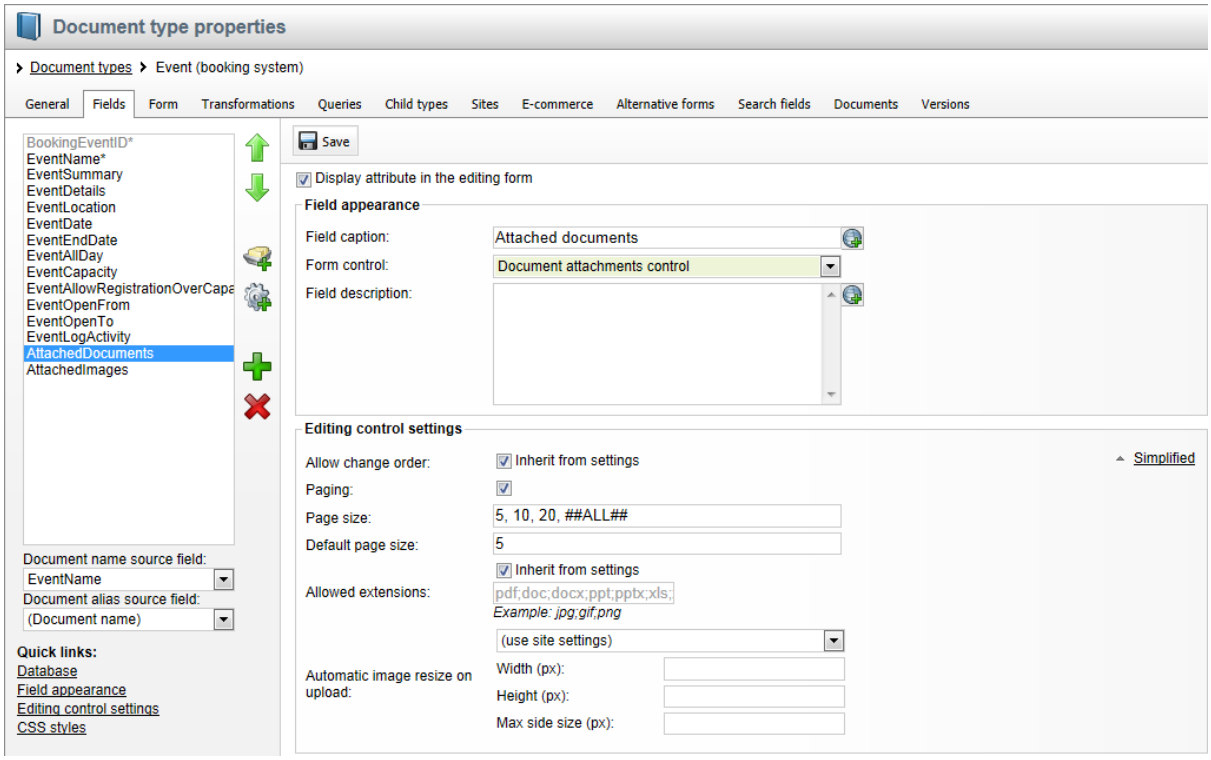
3. Optionally, you can set the following additional parameters for the fields in their **Editing control settings** section (most of the parameters are only available if you switch to **Advanced** mode using the link on the side of the section):

- **Allow change order** - indicates if the order of attachments can be changed.
- **Paging** - indicates if paging should be used for the list of attachments.
- **Page size** - defines the page size options that will be selectable in the attachments pager. Values must be separated by commas, e.g. *25,50,100,##ALL##*.
- **Default page size** - sets the amount of attachments displayed per page by default (if paging is allowed).
- **Allowed extensions** - sets which file types can be uploaded as attachments. Uploading files that do not match the specified extensions will not be permitted. Check *Inherit from settings* to use the values specified in *Site Manager -> Settings -> System -> Files -> Upload extensions*

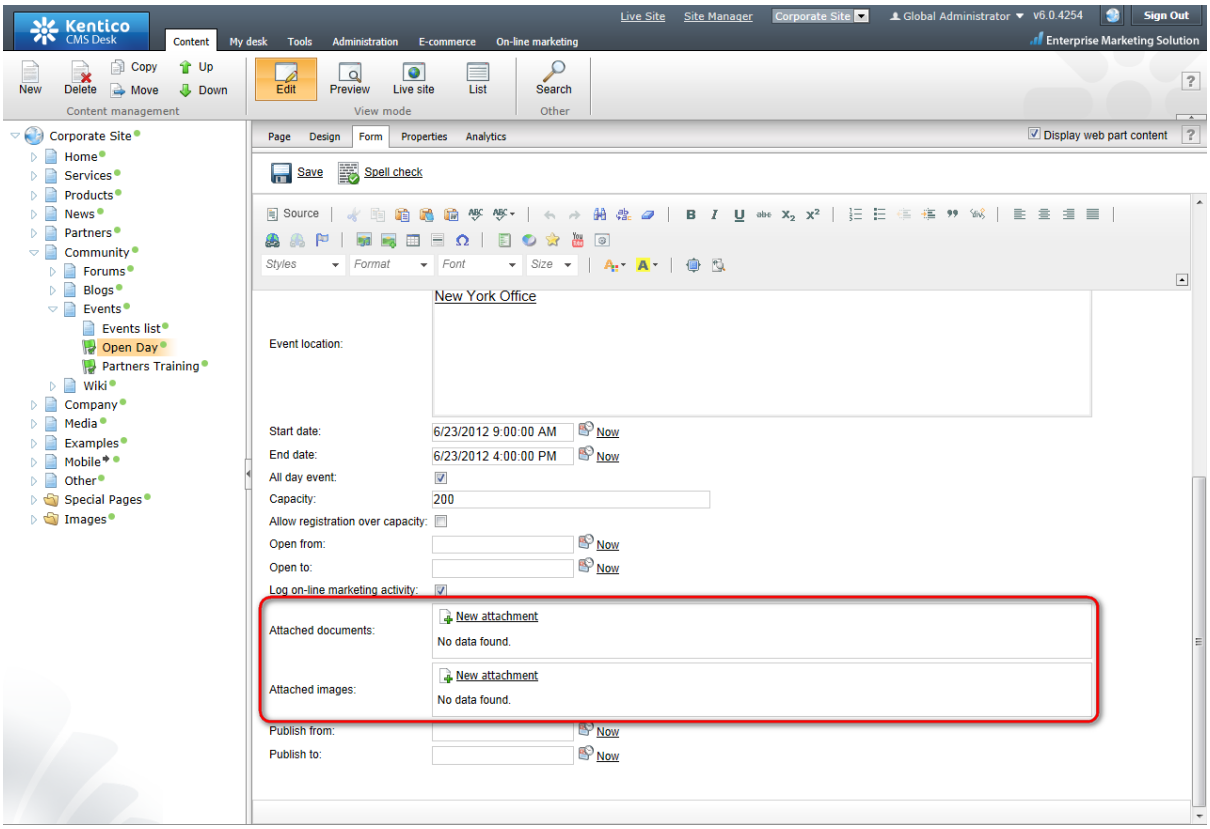
- **Automatic image resize on upload**:
  - **(do not resize)** - uploaded images will not be resized.
  - **(use site settings)** - uploaded images will be resized according to site settings in *Site Manager -> Settings -> System -> Files -> Automatic image resize on upload*.
  - **(use custom settings)** - uploaded images will be resized according to the Width, Height and Max side size values set below.

Depending on which values you fill in the **Width**, **Height** and **Max side size** fields, the functionality is the following:

- **Only width or only height** - images will be resized so that the width/height matches the entered value. The other dimension is also resized so that the aspect ratio is kept.
- **Both width and height** - images will be resized so that both dimensions match the entered values. The aspect ratio will not be kept in this case
- **Max side size** - if one of the image's sides is larger than this value, the image will be resized so that the its larger side matches the entered value. The aspect ratio is kept and width and height settings are not applied.



4. Now that we have the two fields defined, we can upload some attachments to the existing events on the site. Switch to **CMS Desk** and select the **Events -> Open day** from the content tree. Switch to its **Form** tab. If you scroll down the page, you should see the two fields defined in the previous step.



5. Use **New attachment** ( ) to add attachments into both fields. According to the name of the field, upload some documents into the first one and some images into the second one.

Just as on the **Properties -> Attachments** tab, you can perform the following actions with the attachments in the lists:

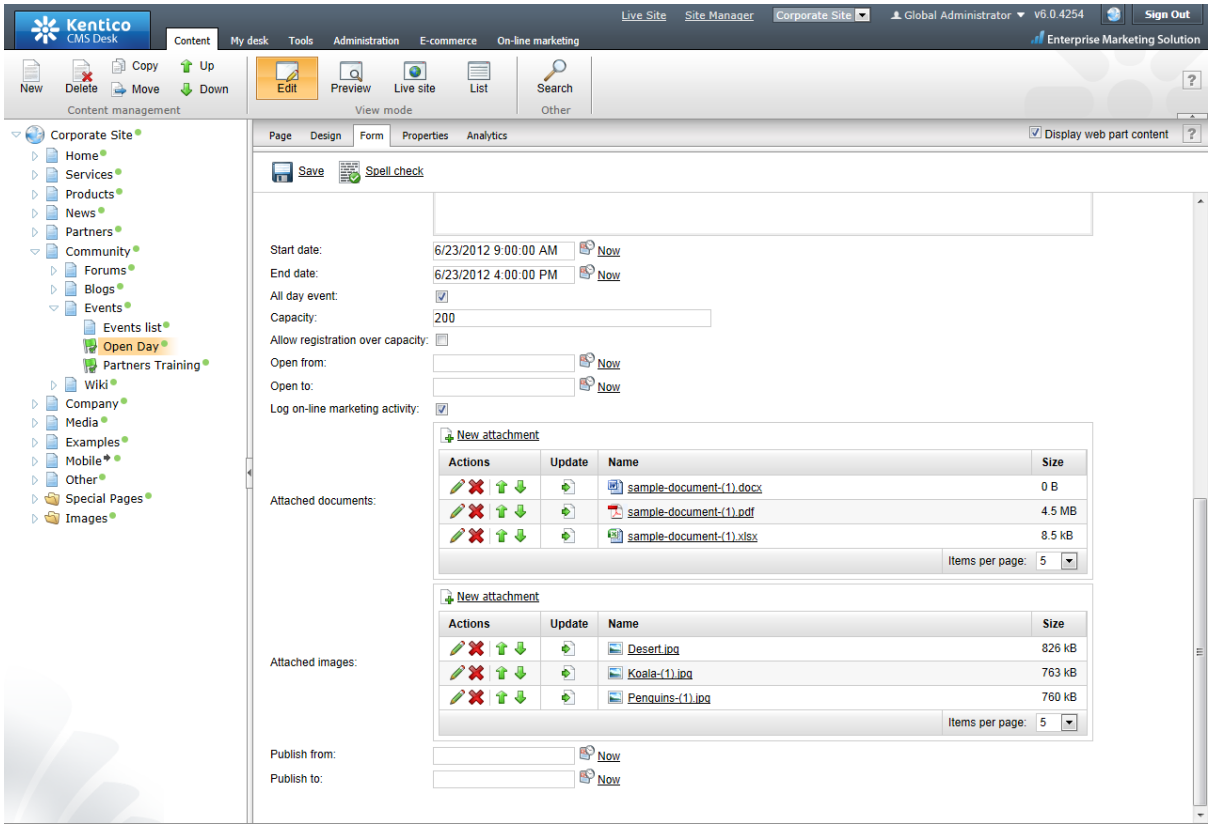
- **Delete** ( ) - removes the attachment from the document.
- **Move up** ( ) and **Move down** ( ) - re-orders the attachments. The order is stored in the **AttachmentOrder** property of each attachment. You can enter *AttachmentOrder* into the **ORDER BY expression** property of a displaying web part to have the attachments ordered accordingly.

> **Please note**
>
> The order of attachments is **not versioned** with documents' workflow. This means that if you change the order of attachments in one version of a document, the order is changed in all other versions too.

- **Edit** ( ) - if the attachment is an image, clicking the icon opens it in the built-in image editor. If the attachment is not an image, the metadata editor is opened after clicking the icon.
- **Edit in client application ( )** - is displayed only when WebDAV integration is enabled and only next to file types supported by WebDAV. Clicking the icon opens the document for editing in an external application installed on the client computer (the application must support WebDAV for this to work). Saving the document in the client application updates the attachment with the new version

automatically.

- **Update** ( ) - enables you to replace the original attachment with a new one.
- After clicking an attachment's name, the attachment is opened.



6. Now that we have the attachments uploaded, it's time to have them displayed on the live site. The events are displayed by a repeater on the Events page, so we will have to add the displaying web parts to this page. First, we will add the **Document attachments** web part. We will configure it so that it will display attachments from **Attached documents** field.

Select the **Events** page and switch to the **Design** tab. Click the **Add web part** ( ) icon of the **Main zone** web part zone and choose the **Attachments -> Document attachments** web part. Set the following properties of the web part, leave the rest at the default values.

- **Show for document types**: CMS.BookingEvent
- **Attachment group**: Event (Booking system) -> Attached documents
- **Content before**: <h2>Attached documents</h2>

Click **OK**.

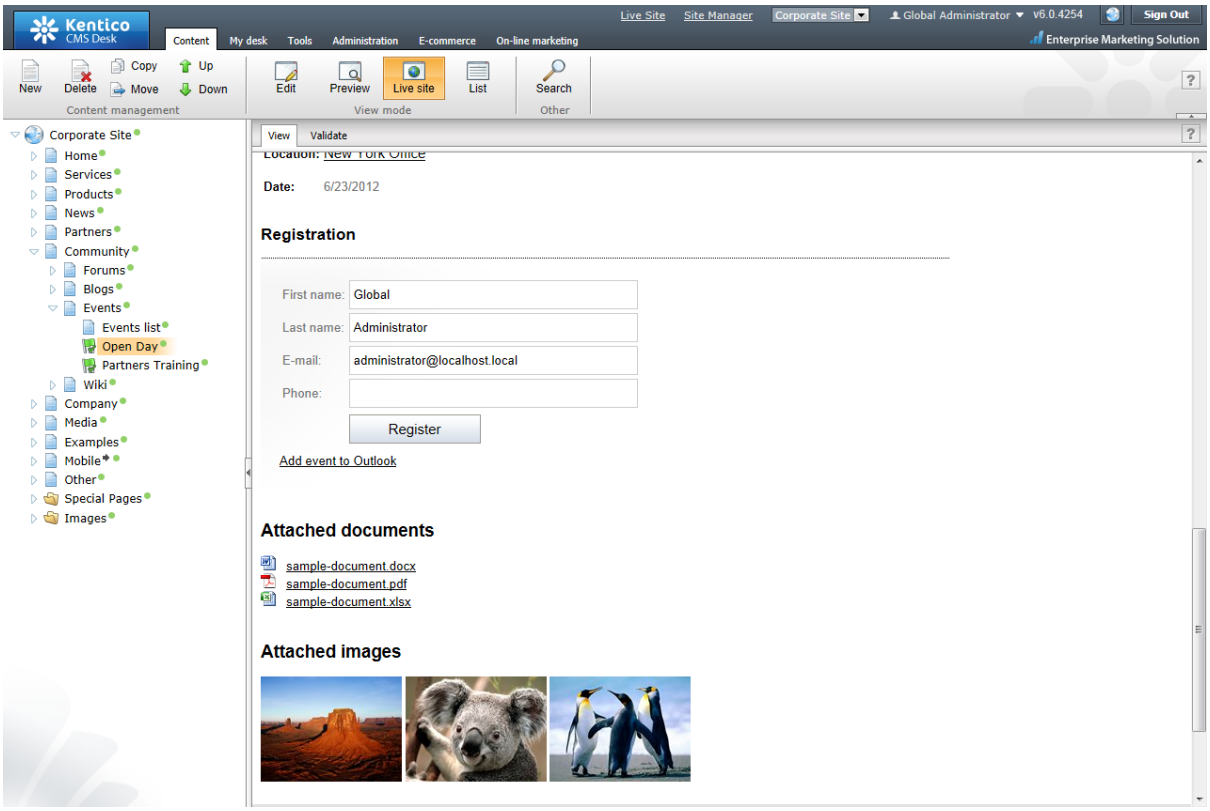
7. Below the documents, we will want the attached images to be displayed. For the images, it will be better to use the **Attachment image gallery** web part. Click the **Add web part** ( ) icon of the **Main zone** web part zone and choose the **Attachments -> Attachment image gallery** web part. Set the following properties of the web part, leave the rest at the default values.

- **Show for document types**: CMS.BookingEvent
- **Attachment group**: Event (Booking system) -> Attached images

- **Content before**: <h2>Attached images</h2>

Click **OK**.

8. If you switch to the live site now, you should see the two web parts displaying the attachments, each web part one group defined earlier in this example. If you add attachments to any other event, they would be displayed with the event too.



4.8.2.4 Available web parts

In this chapter, you can see an overview of the web parts that can be used to display document attachments. Only the most important web part properties are explained here. For a complete list of web part properties, please refer to Kentico CMS Web Parts reference guide or click the **Documentation** link at the top right corner of the web part properties window.

Document attachments

Sample-document-(1).docx
Sample-document-(1).pdf
Sample-document-(1).xlsx

The web part is located in the **Attachments** web part category. This web part displays a list of files attached to a document. After clicking an attachment, it is opened in a new window.

The following properties of the web part are the most important:

- **Path**: alias path of the document whose attachments should be displayed; if blank, the currently displayed document's attachments will be displayed
- **Attachment group**: specifies the field from which the attachments should be displayed; if blank, ungrouped attachments from the Properties -> Attachments tab will be used
- **Transformation**: transformation used for displaying the attachments; the default transformation is *CMS.Root.AttachmentList*

Attachments image gallery

The web part is located the **Attachments** web part category. The web part is particularly useful for image attachments. It displays thumbnails of images attached to the document. After clicking a thumbnail, the clicked image is displayed in its full size by a **lightbox**. For non-image attachments, its file type icon is displayed. After clicking such an icon, the file is offered for download.

The following properties are the most important:

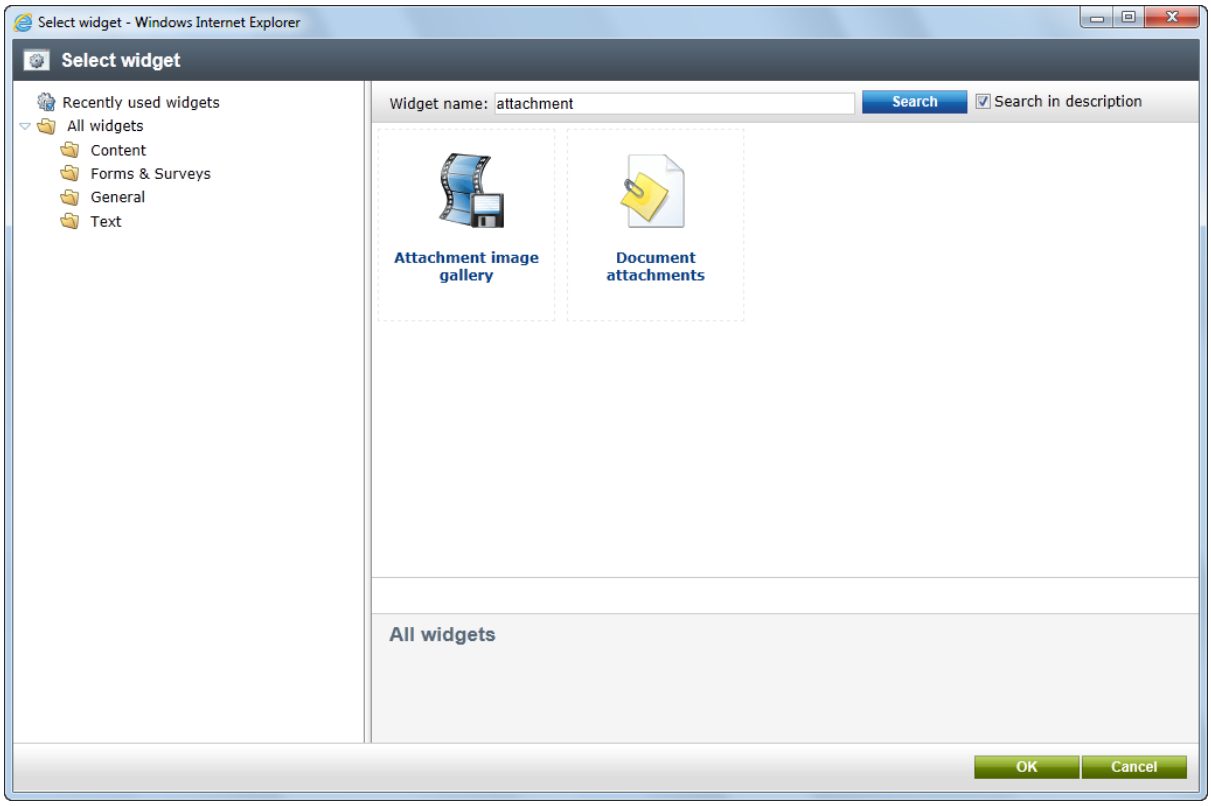
- **Path**: alias path of the document whose attachments should be displayed; if blank, the currently displayed document's attachments will be displayed
- **Attachment group**: specifies the field from which the attachments should be displayed; if blank, ungrouped attachments from the Properties -> Attachments tab will be used
- **Transformation**: transformation used for displaying the attachments; the default transformation is *CMS.Root.AttachmentLightbox*
- **Selected item transformation**: transformation used to display the selected item; the default transformation is *CMS.Root.AttachmentLightboxDetail*
- **LightBox Configuration**: the properties in this section can be used to customize the used LightBox

Attachments Data Source

This is a special web part - a data source. It is not visible on the live site. It only provides data to another connected web part (e.g. Basic Repeater), which will display the provided data. More information on data source web parts can be found in this chapter.

4.8.2.5 Available inline widget

There are two Inline widgets that can be used for displaying unsorted attachments of documents. The inline widgets can be added to a page via the **Editable text** web part, using the WYSIWYG editor on the **Page** tab. To add a widget to the page, just place the cursor in the appropriate position and click the **Insert/Edit widget** ( ) button. You can choose one of the following two widgets from the **Content** category for displaying attachments, which are based on the document attachment web parts:



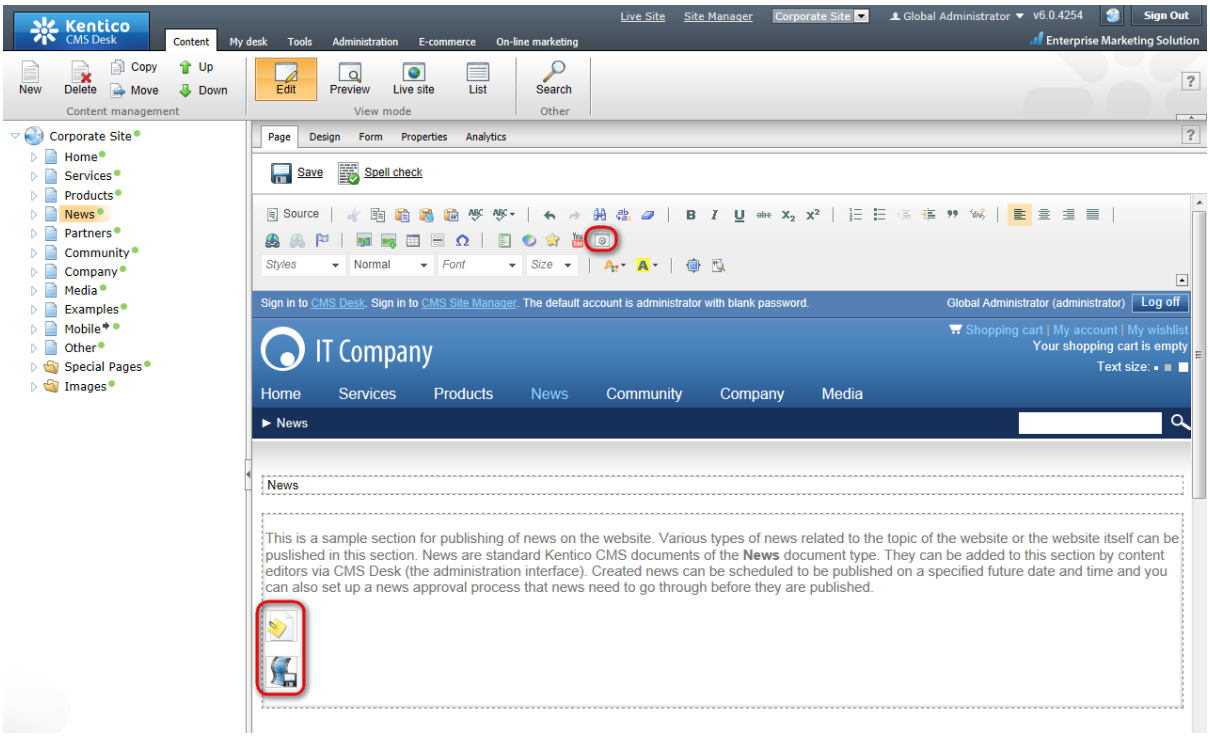
Attachment image gallery

Displays thumbnails of the document's unsorted attachments in a gallery. After a thumbnail is clicked, the attachment will be displayed in a lightbox. By default, the widget uses the **cms.root.attachmentLightbox** transformation for the gallery and **cms.root.attachmentLightboxDetail** for the lightbox.

Document attachments

Displays the document's unsorted attachments. After an attachment is clicked, it will be displayed on a new page if it is an image and non-image attachments will be offered for download. By default, images are displayed based on the **cms.root.attachment** transformation.

In the Editable text's text area, the widgets are represented by matching placeholder images.



4.8.2.6 Handling attachments in transformations

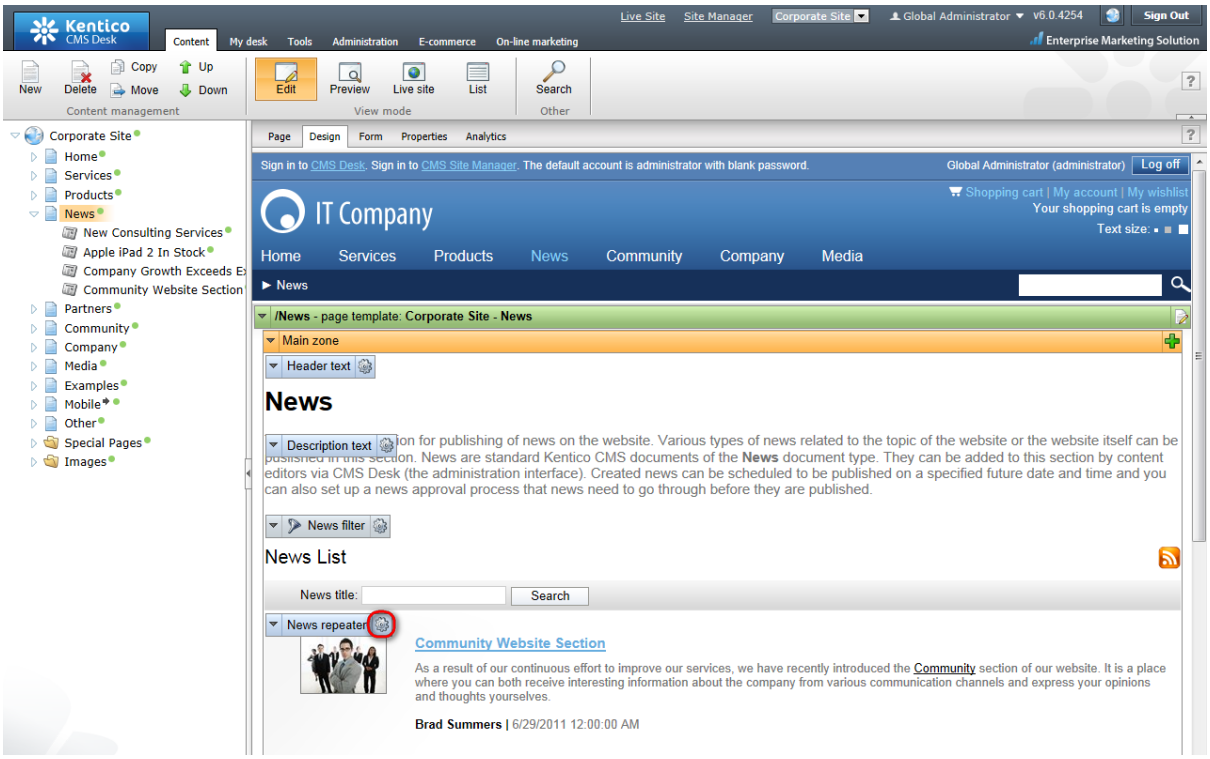
You can display document's unsorted attachments by adding one of the following two controls directly into the transformations:

- **~/CMSAdminControls/Attachments/DocumentAttachments.ascx**
- **~/CMSAdminControls/Attachments/AttachmentLightboxGallery.ascx**

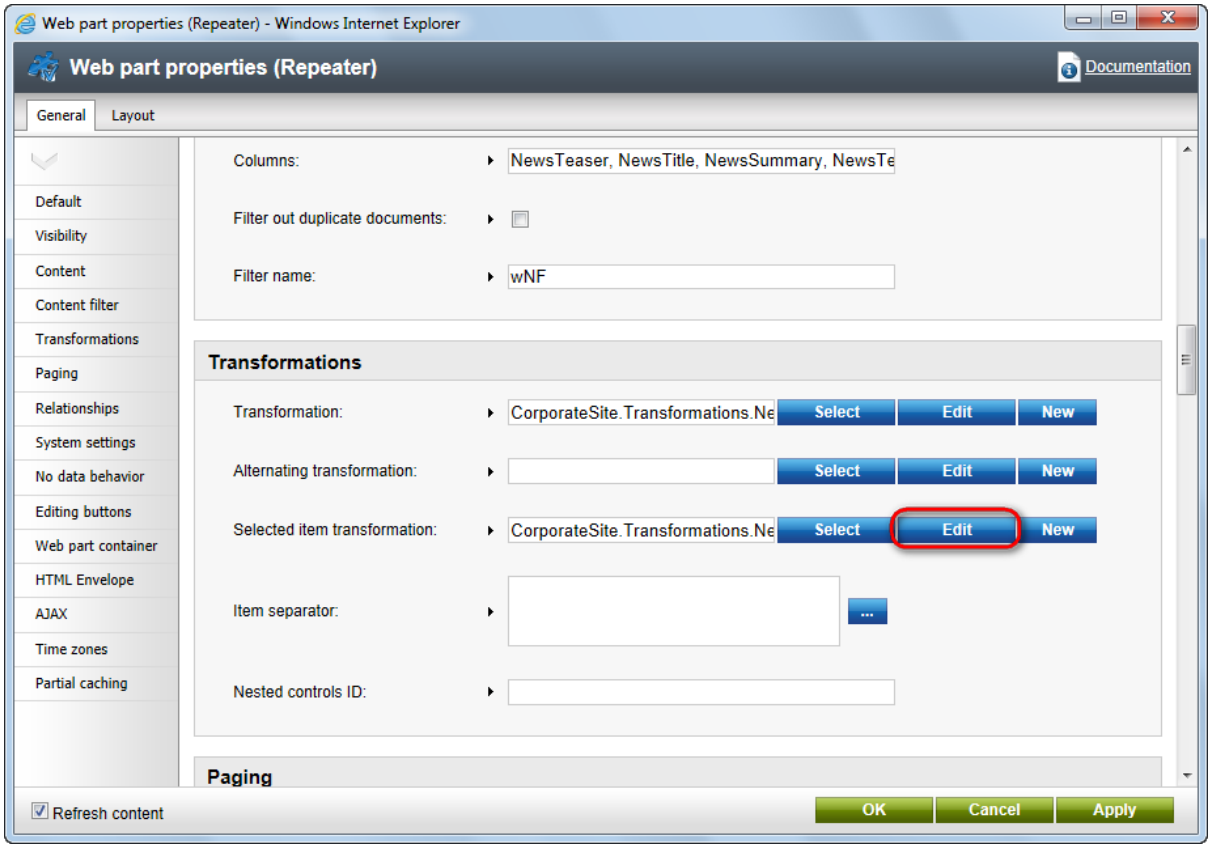
In the following example, you will learn how to use these controls in transformations. The news in the **News** section of the sample **Corporate Site** have some images attached by default. In this example, you have learned how to display these attachments using a web part. Another way of displaying the attachments is by modifying the particular transformation for the document type.

The news are displayed on the News page using a **Repeater**. Detail view of each news item is displayed using the **CMS.News.NewsDetail** transformation. We will modify this transformation so that it displays the attachments along with the news summary and text.

1. Go to **CMS Desk** and select the **News** page from the content tree. Switch to **Design** tab and choose to **Configure** ( ) the **News repeater** web part.



2. In the web part properties window, choose to **Edit** the **Selected item transformation**.

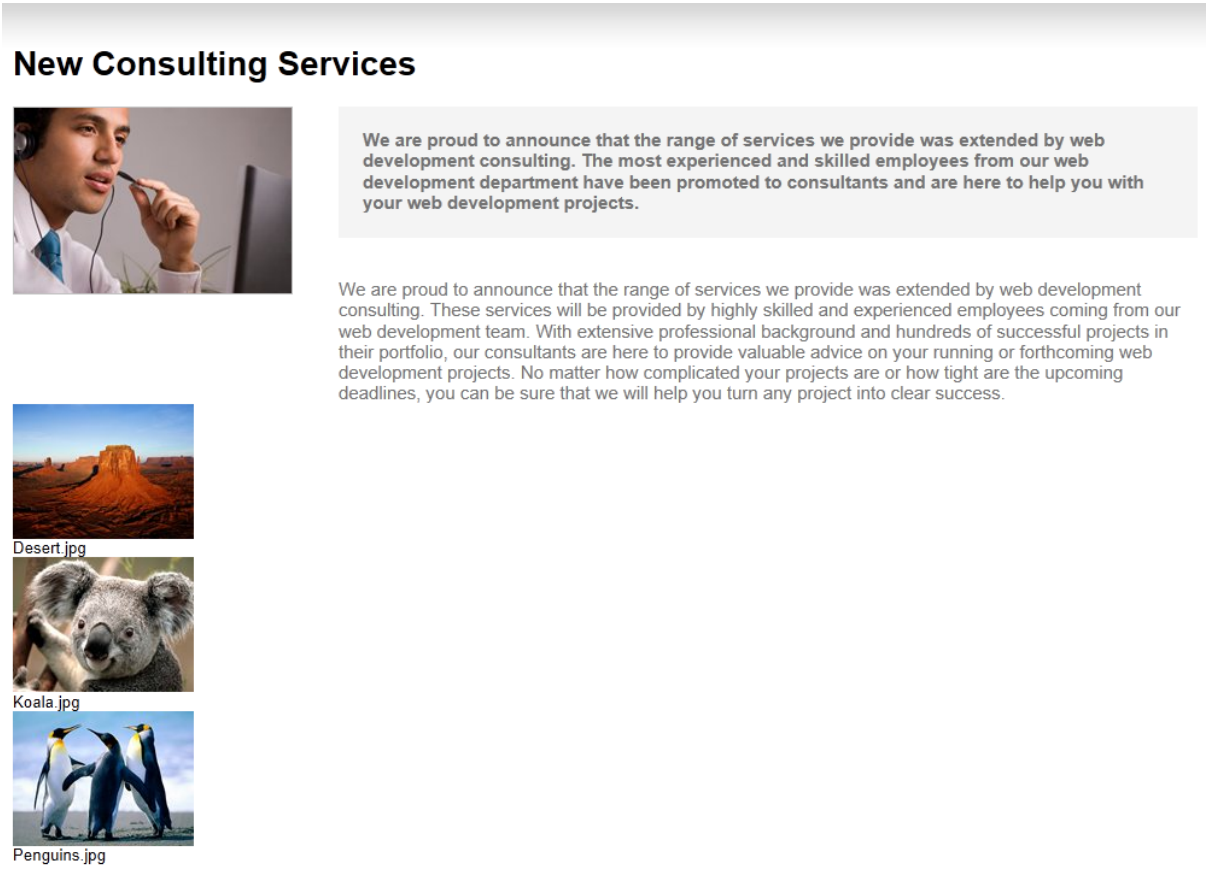


3. Replace the original transformation with the following code, which is the original transformation with the highlighted parts added.

```
<%@ Register Src="~/CMSInlineControls/DocumentAttachments.ascx"
TagName="DocumentAttachments" TagPrefix="cms" %>

<div class="listDetail">
<h1 class="newsTitle"><%# Eval("NewsTitle", true) %></h1>
<div class="teaser"><%# IfEmpty(Eval("NewsTeaser"),"<img src=\"~/App_Themes/
CorporateSite/Images/no_image_news.png\" alt=\"Default news teaser image\" />",
GetImage("NewsTeaser", 230, 230, 500, Eval("NewsTitle"))) %></div>
<div class="contentText contentTextTeaser">
<div class="summary">
<p><%# Eval("NewsSummary") %></p>
</div>
<div class="text">
<p><%# Eval("NewsText") %></p>
</div>
</div>
<div class="clear"></div>

<div class="NewsBody">
<cms:DocumentAttachments ID="ucDocAttachments" runat="server"
TransformationName="cms.root.attachment" Path='<%# Eval("NodeAliasPath") %>' />
</div>

</div>
```

Click **Save**. Click **OK** in the web part properties window.

4. If you go to the live site now and view some of the news, you should see the attachments displayed below the news text, just as in the screenshot below.



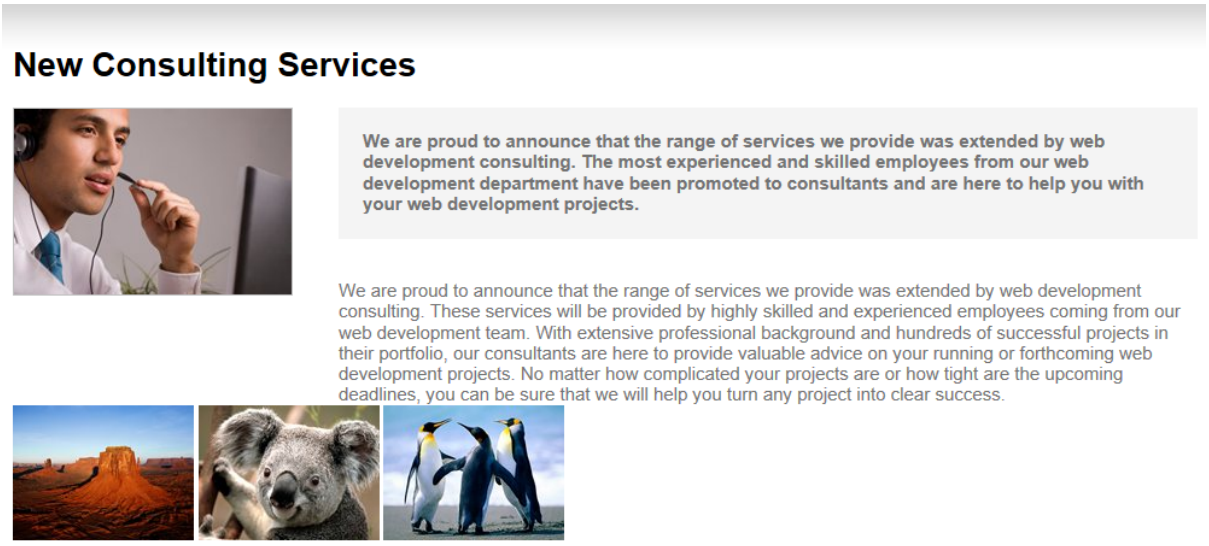
5. Let's try the other control now. Choose to **Configure** ( ) the **NewsRepeater** again and choose to **Edit** its **Selected item transformation**. Replace the transformation with the code below. Again, it is the original transformation with the highlighted parts added.

```
<%@ Register Src="~/CMSInlineControls/AttachmentLightboxGallery.ascx"
TagName="AttachmentLightboxGallery" TagPrefix="cms" %>

<div class="listDetail">
<h1 class="newsTitle"><%# Eval("NewsTitle", true) %></h1>
<div class="teaser"><%# IfEmpty(Eval("NewsTeaser"),"<img src=\"~/App_Themes/
CorporateSite/Images/no_image_news.png\" alt=\"Default news teaser image\" />",
GetImage("NewsTeaser", 230, 230, 500, Eval("NewsTitle"))) %></div>
<div class="contentText contentTextTeaser">
<div class="summary">
<p><%# Eval("NewsSummary") %></p>
</div>
<div class="text">
<p><%# Eval("NewsText") %></p>
</div>
</div>
<div class="clear"></div>

<div class="NewsBody">
<cms:AttachmentLightboxGallery ID="ucDocAttachments" runat="server"
TransformationName="cms.root.attachmentLightbox"
```

```
SelectedItemTransformationName="cms.root.attachmentLightboxDetail" Path='<%# Eval
("NodeAliasPath") %>' />
</div>

</div>
```

Click **Save**. Click **OK** in the web part properties window.

6. If you go to the live site now, you should see the result as in the screenshot below. After clicking one of the images, it will be displayed in the lightbox.

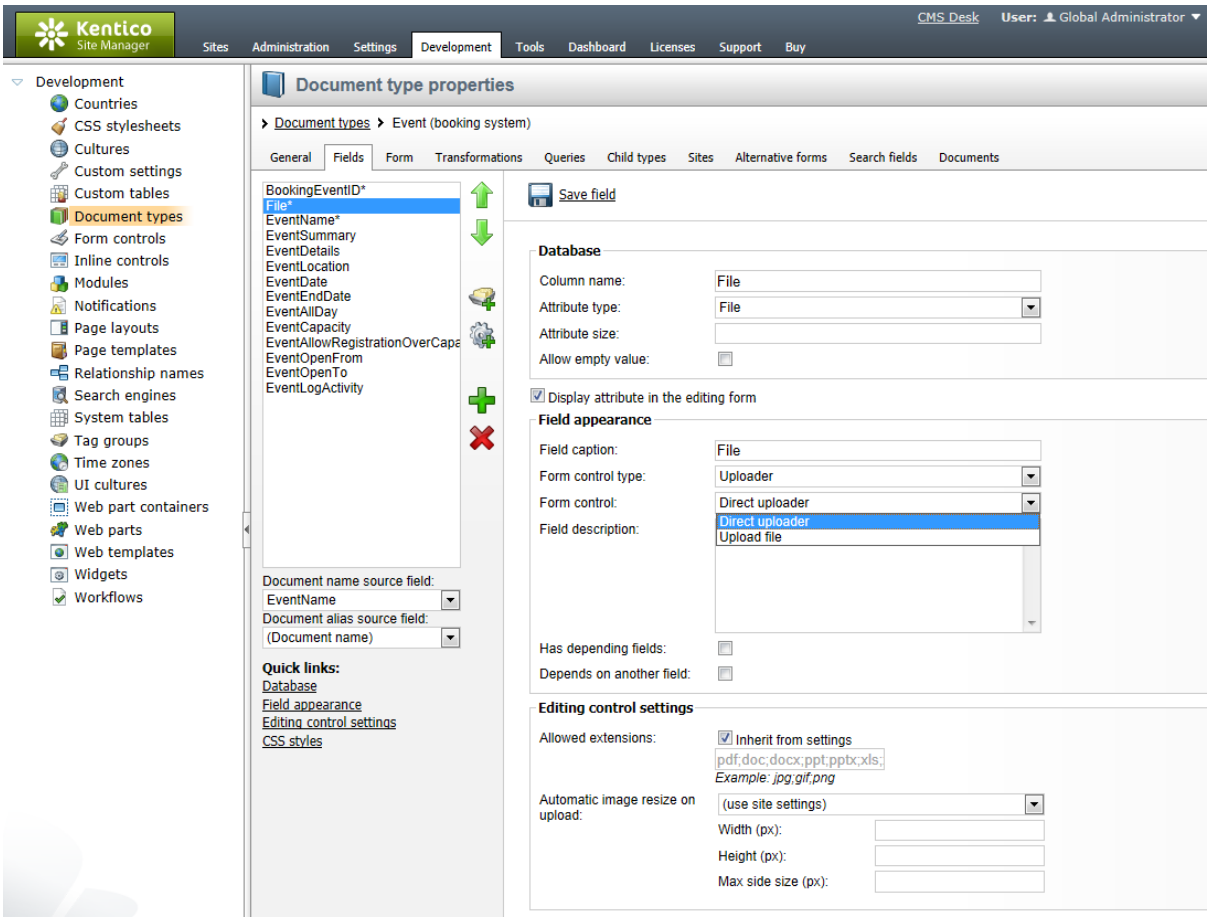


Please note: the appearance shown in this example is based on the controls' default transformations. You can fully customize the appearance by modifying the default transformations or creating your own transformations and specifying them in the **TransformationName** and **SelectedItemTransformationName** properties of the controls.

4.8.2.7 Using the File field

In the overview, we mentioned the **File** field used in the previous versions. Use of this field is still possible and could be particularly useful if you want to have a field where only one file can be uploaded. The default **CMS.File** document type is based on the use of this type of field.

If you want to define a new **File** field, choose **File** from the **Attribute type** drop-down list when defining a new field.



As you can see in the screenshot above, the **Form control** drop-down list can be used to choose between two available controls for file uploading which will be displayed on the **Form** tab:

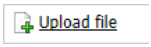
- **Upload file**

  This is the original control which was used in the previous versions.

- **Direct uploader**

  This is the same control that is used for uploading attachments via the **Form** tab. The only difference is that it allows only one file to be uploaded.

  This is how it looks like when no file is uploaded:



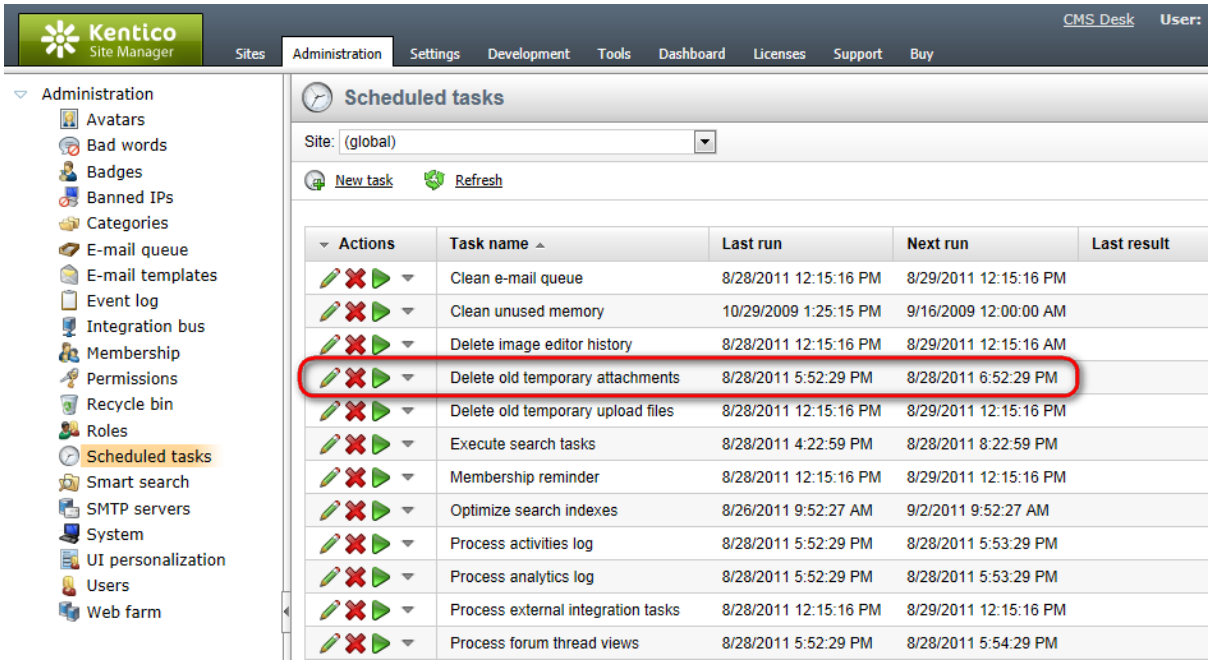
  And here is the control after file upload:

4.8.2.8 Temporary attachments handling

If you create a new document and start attaching files to it, temporary attachments are created. When the document is saved, the temporary attachment becomes a real attachment. If the document is not saved, the temporary attachments remain on the disk or in the database. To handle the unused files, there is a scheduled task pre-defined.

In **Site Manager -> Administration -> Scheduled tasks**, you can find the **Delete old temporary attachments** scheduled task. By default, this task is performed on a daily basis and deletes all temporary attachments older than 24 hours.

You can customize the interval by adding the <add key="CMSDeleteTemporaryAttachmentsOlderThan" value="1"/> key into the <appSettings> section of your site's *web.config* file. The value specifies the interval in hours.



4.8.2.9 Attachment names

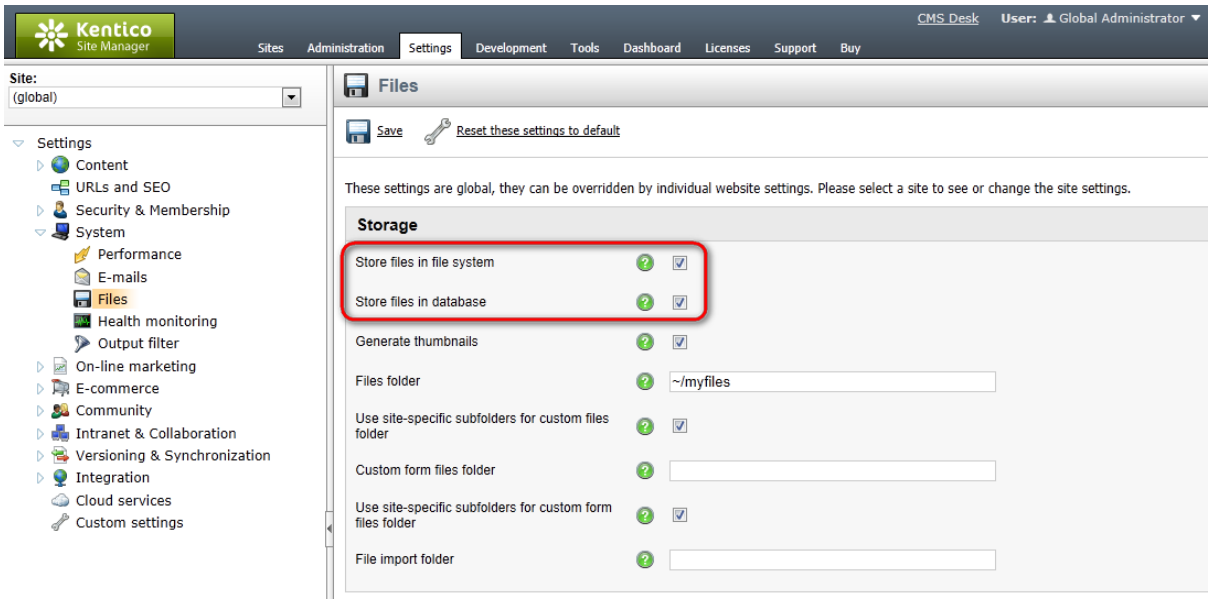
Attachment names are unique for a document (or its version). If you try to upload an attachment that has the same name as some already existing attachment of the particular document, the new attachment gets a number in the **-(1)** format attached to its name.

This happens for the attachments to be accessible via friendly URLs in the following format:

- **~/getattachment/<node_alias_path>/<safe_filename>.<extension>**

4.8.3 Where the files are stored

Document attachments and CMS.File documents can be physically stored in the file system, in the database, or in both. You can define this in **Site Manager -> Settings -> System -> Files**, using the **Store files in file system** and **Store files in database** options.



The following three combinations can be achieved using the settings:

- **File system** - the files are stored in the configured folder on your disk. This option provides the best performance. However, the **Modify** permissions on the disk must be granted to the ASP.NET account on your machine, which is not always possible. The process of granting the Modify permission is described in this chapter.
- **Database** - the files are stored in the database. This option provides worse performance, but it allows you to use full-text search in uploaded files. It also doesn't require the Modify permission on the disk and it allows you to easily backup the uploaded files as a part of your database backup.
- **Both - Database and file system** - this option combines the advantages of both options. It provides the same performance as the file system-only option since the files are stored in the file system. At the same time, you can use the full-text search because the database is also available.

Document attachments and metafiles location

Document attachments are stored in the file system under *~/<site code name>/files*, metafiles are stored under *~/<site code name>/metafiles*. Both of these folders can be located in a different location, which can be configured using the **Site Manager -> Settings -> System -> Files -> Custom files folder** settings option. Lear more in File-related settings.

Form files location

Files uploaded by site users into forms are always stored in the file system. The default location is *~/<site code name>/BizFormFiles*. You can customize the location in **Site Manager -> Settings ->**

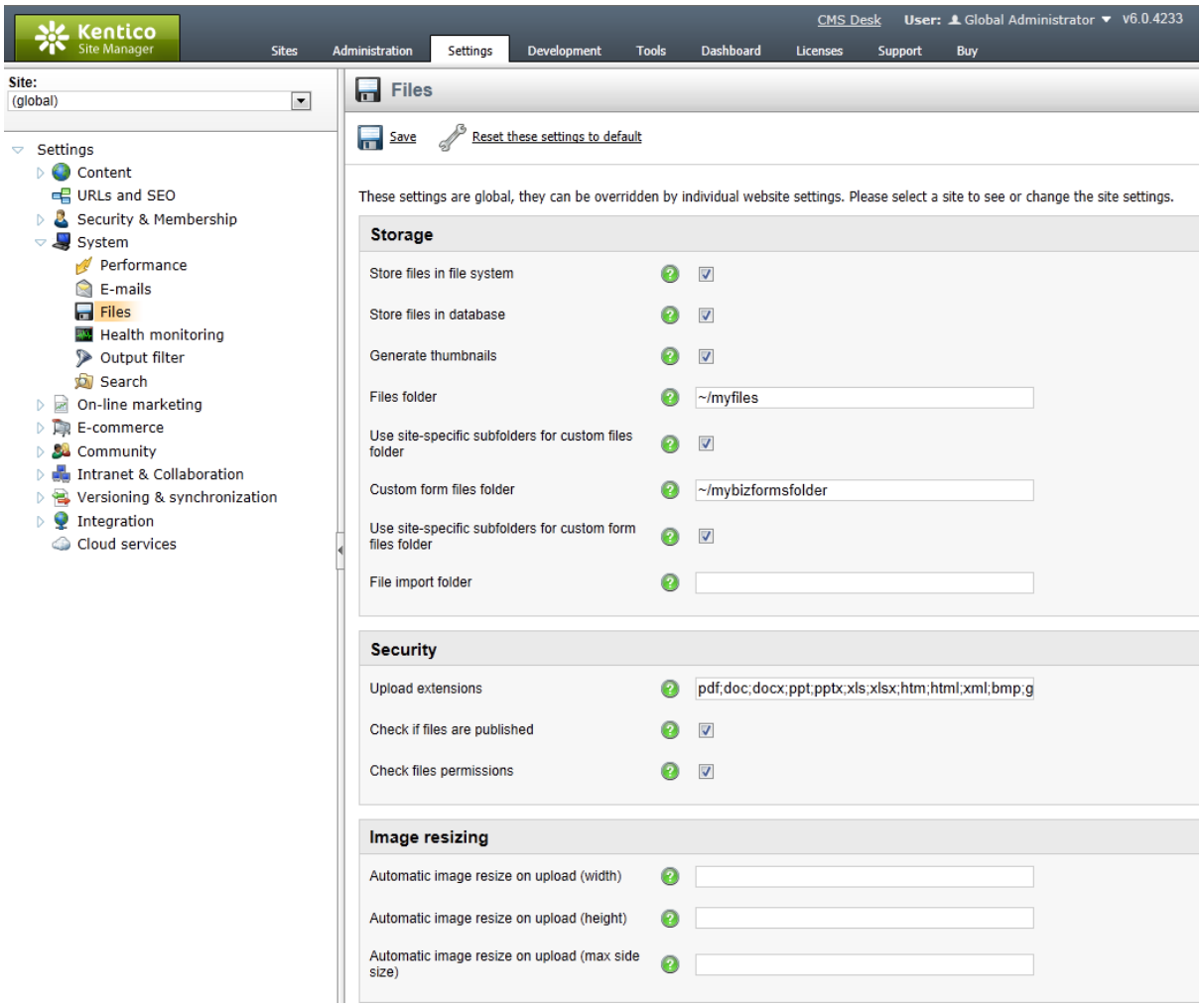
System -> Files -> Form files folder.

Media library files location

Files stored in media libraries are always stored in the file system. The default location is *~/<site code name>/media*, while the location of the folder can be customized in **Site Manager -> Settings -> Content -> Media -> Custom media libraries folder**, as described in Modules -> Media libraries -> Media libraries settings.

4.8.4 File-related settings

You can configure file storage options in **Site Manager -> Settings -> System -> Files**.



The following options can be set on the page:

Storage	
Store files in file system	Indicates if files should be stored in the file system.

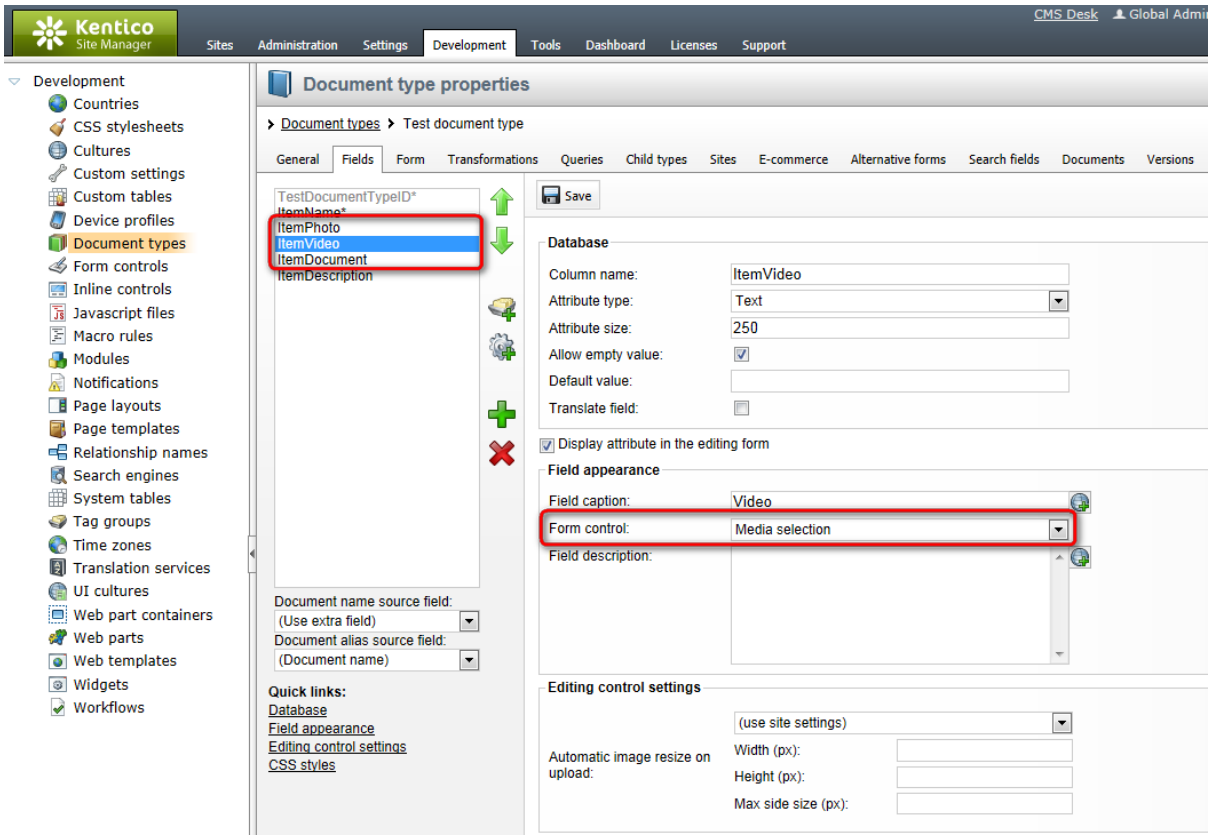
Store files in database	Indicates if files should be stored in the database.
Generate thumbnails	Indicates if the CMS should generate image thumbnails on the disk when a **resized version** of the image is displayed. This option only applies if files are stored in the file system. It **improves site performance**.
Files folder	The folder on the disk where document attachments and metafiles are stored. You can use: • **physical disk path**: e.g. *c:\myfiles* • **virtual path**: *~/UploadedFiles* • **UNC path**: *\\server\folder* If you do not specify any value, the files are stored in folder *~/<site code name>/files*.
Use site-specific subfolders for custom files folder	This setting is only applied when a **Custom form files folder** is configured. If enabled, attachment files will be stored in a sub-folder named according to the code name of the site where the form is placed, i.e. *<custom form files folder>/<site code name>*. It is useful for better orientation in files when multiple sites are running in the system.
Custom form files folder	Folder where files uploaded via Forms are stored. You can use: • **physical disk path**: e.g. *c:\myfiles\mysite* • **virtual path**: *~/UploadedFiles* • **UNC path**: *\\server\folder* If no value is entered, the files are stored in *~/<site code name>/BizFormFiles*.
Use site-specific subfolders for custom form files folder	This setting is only applied when a **Custom form files folder** is configured. If enabled, attachment files will be stored in a sub-folder named as the site code name under the custom files folder, i.e. *<custom BizForm files folder>/<site code name>*. It is useful for better orientation in files when multiple sites are running in the system.
File import folder	Path to the source folder where files to be imported by the File import should be uploaded before import. If no value is entered, *~/CMSImportFiles* is used by default.
Security	
Upload extensions	Specifies which extensions are allowed for uploaded files. You can restrict the types of uploaded files by entering a limited list of extensions separated by semicolons, for example: *gif;jpg;doc;pdf* This allows you to block users from uploading potentially dangerous files, such as ASPX scripts. If no value is specified, uploading will be allowed for all file types.
Check if files are published	If checked, only files that are in the **Published** workflow step can be accessed from the live site when a workflow is applied to the document.
Check files permissions	If checked, document permissions are applied to the files.

Image resizing	
Automatic image resize on upload (width, height, max side size)	Depending on which values you fill in, the functionality is the following when uploading images: • **No values are entered** - images are not resized. • **Only width or only height** - images are resized so that their width or height matches the entered value. The other dimension is also resized so that aspect ratio is preserved. • **Both width and height** - images are resized so that the larger dimension matches the respective entered value. Aspect ratio is not preserved. • **Max side size** - if one of the image's sides is larger than this value, the image is resized so that its larger side's dimension matches the entered value. Aspect ratio is preserved and width and height settings are not applied. More info can be found in the Resizing images chapter of Kentico CMS Developer's Guide.
Watermark	
Watermark image	Image name or path that will be used for watermarking the images. User either a full path (~/..) or just a file name from the *~/App_Themes/Default/Images/Watermarks* folder.
Watermark position	The position where the image watermark is placed on the watermarked images.
Minimum image width for watermark	Minimum width of an image to be watermarked.
Minimum image height for watermark	Minimum height of an image to be watermarked.
Use watermark for document images	If checked, the watermark is used for document attachments.
Use watermark for media files	If checked, the watermark is used for media files.
Use watermark for object attachments	If checked, the watermark is used for object attachments.

4.8.5 Using the Media selection control

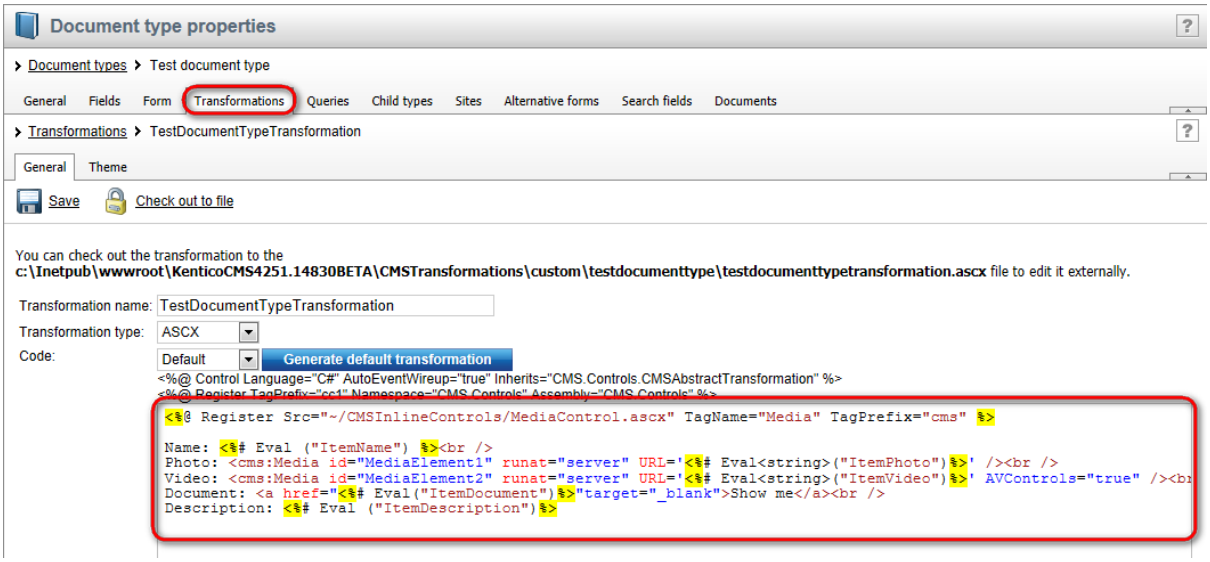
The **Media selection** form control can be used to enable users to select files on the **Form** tab of a document. It allows to use any file from any source (from Document attachments, Content and Media libraries) in your documents. The following example demonstrates how you can achieve this:

1. Navigate to **Site Manager -> Development -> Document types** and create a custom document type with 3 fields, which will enable content editors to select images, videos and documents. Name the document type *Test document type* and the three fields *ItemPhoto*, *ItemVideo* and *ItemDocument.* Make sure that all three fields use the *Media selection* **Form control**:

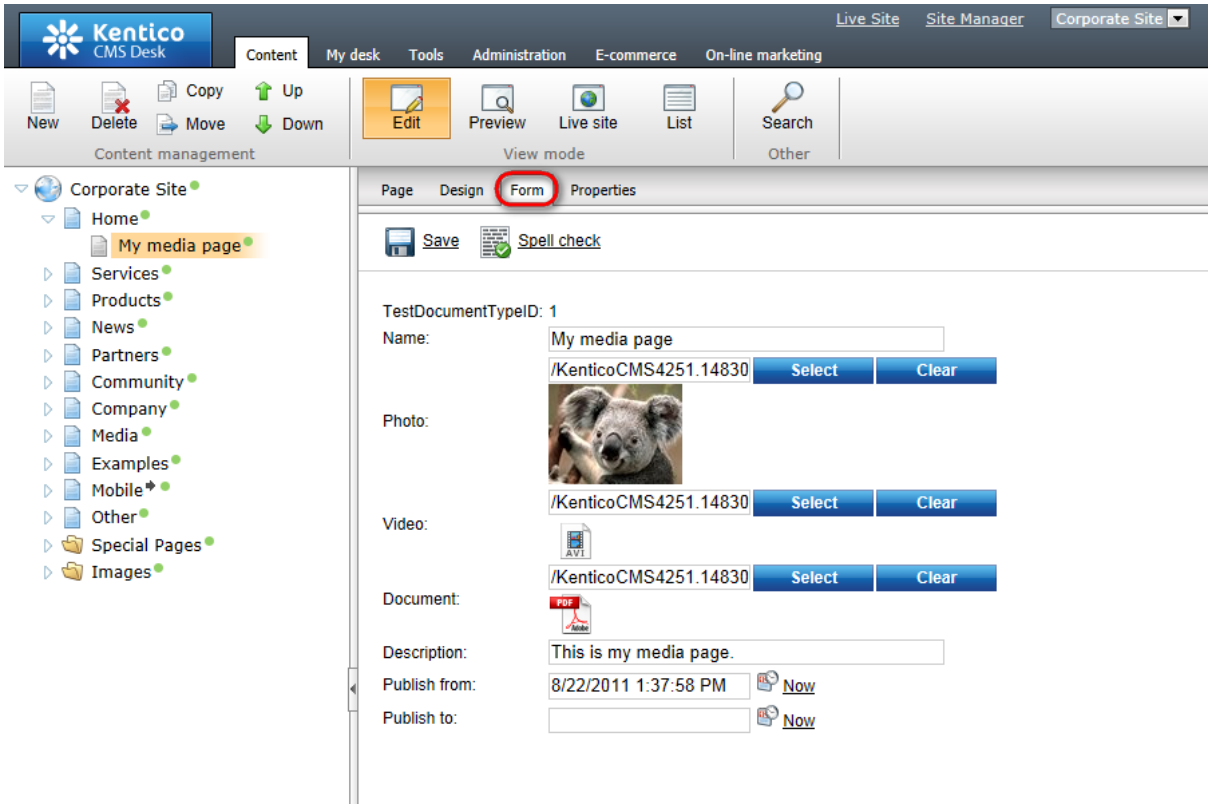


2. In the **Document type properties** user interface, switch to the **Transformations** tab and specify the new transformation as follows:

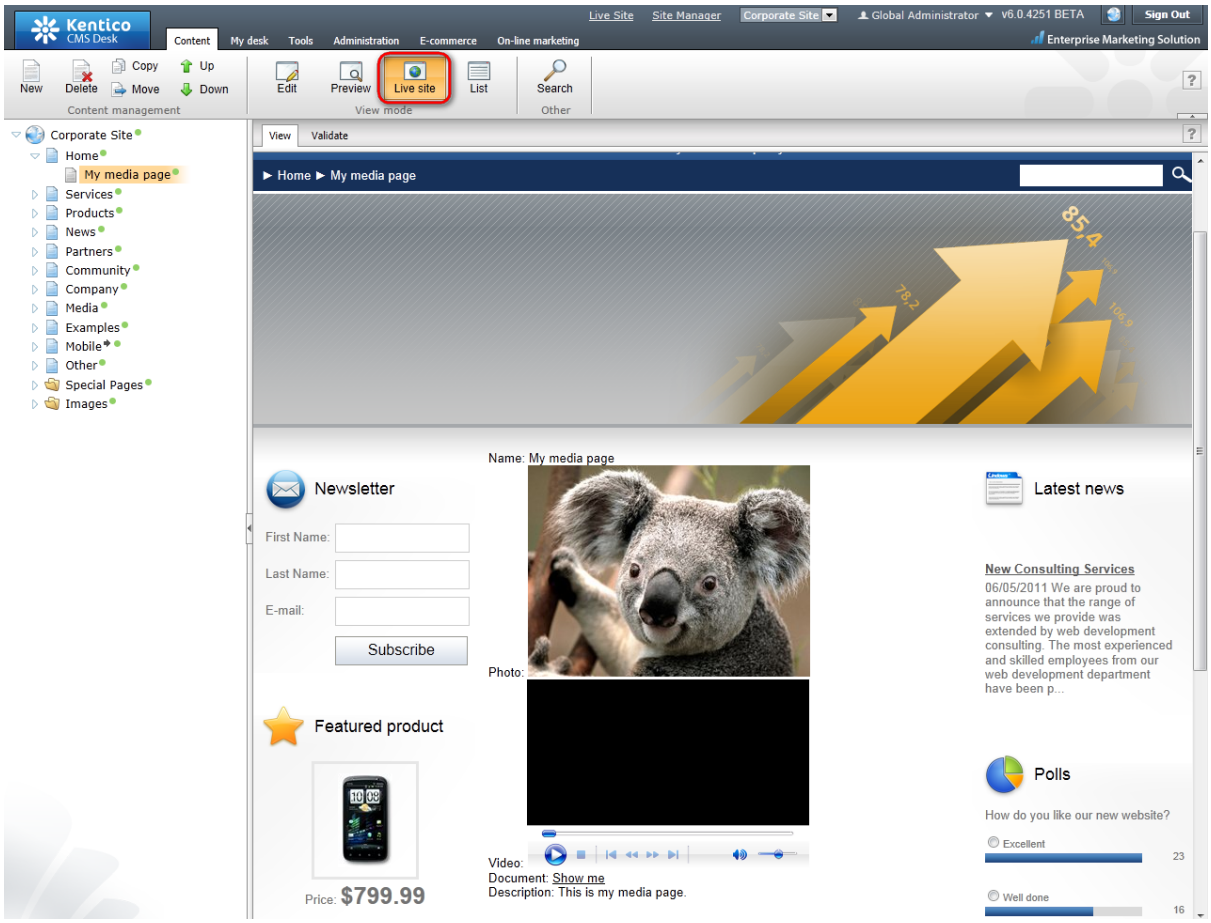
```
<%@ Register Src="~/CMSInlineControls/MediaControl.ascx" TagName="Media"
TagPrefix="cms" %>

Name: <%# Eval ("ItemName") %><br />
Photo: <cms:Media id="MediaElement1" runat="server" URL='<%# Eval<string>
("ItemPhoto")%>' /><br />
Video: <cms:Media id="MediaElement2" runat="server" URL='<%# Eval<string>
("ItemVideo")%>' AVControls="true" /><br />
Document: <a href="<%# Eval("ItemDocument")%>"target="_blank">Show me</a><br />
Description: <%# Eval ("ItemDescription")%>
```



3. Now go to **CMS Desk -> Content**, create a new document of the *Test document type* type and specify its data on the **Form** tab:



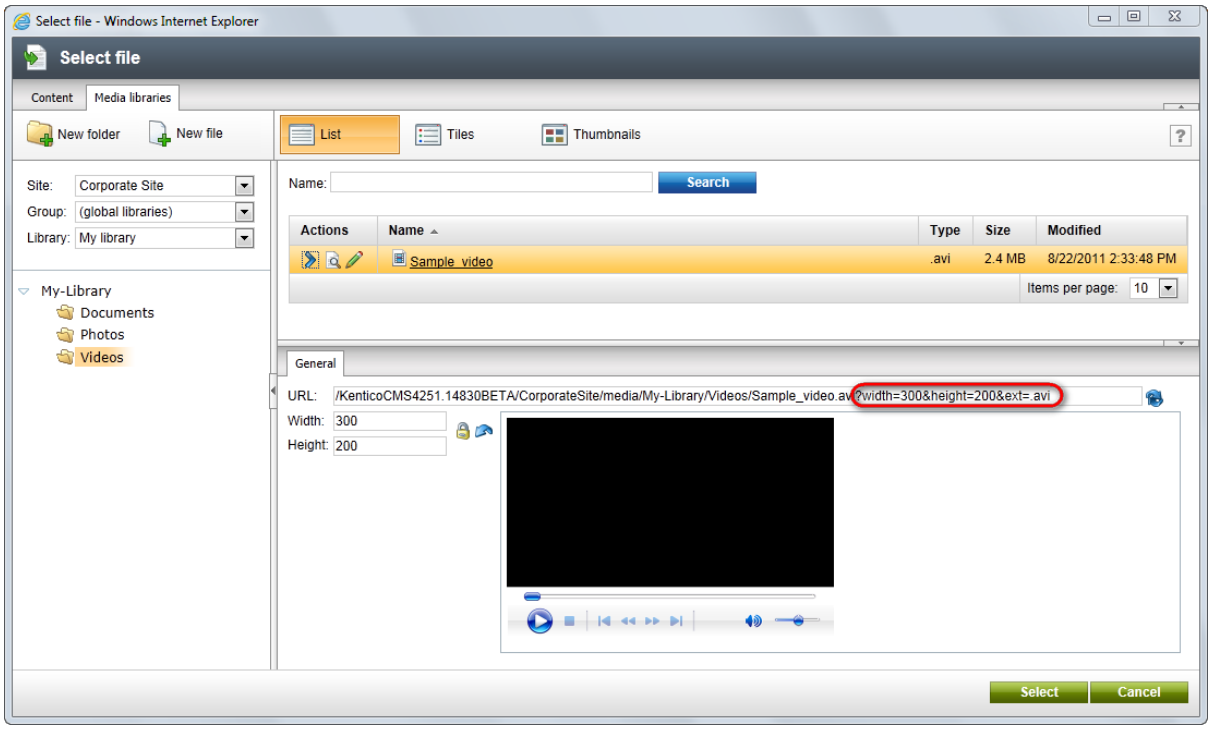
4. Finally, switch to the **Live site** mode to see the results. You will need to use some of the viewers (e.g. the **Repeater** web part) to display data of your new document using your defined transformation:



As you can see, the **Media selection** control is able to identify the type of content from its URL property automatically. It is recognized according to the following URL parameters:

- **Width** - the width of the image or media player.
- **Height** - the height of the image or media player.
- **Ext** - the extension of the file (i.e. the type of media).

These parameters are automatically attached to the selected file URL when selecting the media using the **Media selection** field:



4.8.6 Image editor

4.8.6.1 Overview

Kentico CMS comes shipped with the integrated Image editor that enables you to edit images of four image formats: **.bmp**, **.gif**, **.png** and **.jpg**. The editor can be accessed throughout the whole system, everywhere the **Edit** (🖉) icon is available for a listed image, just like in:

- **Media libraries**
- **Document attachments**
- ***CMS.File* documents Form tab**
- **Product images**
- **Form files**
- **etc.**

After the **Edit** (🖉) icon is clicked, the following Image editor dialog is displayed:

- Image editing tools - enable you to perform actions such as **Resize**, **Rotation**, **Convert**, **Crop** and **Color** adjustment; you can view also the **Properties** of the image.
- **Image preview** - previews the current change to the image.
- **Undo & Redo changes** - you can undo and redo the changes made to the image.
- **Save changes & Close** - if you want to keep the changes you have made, you can save the image; or you can leave the editor and keep the original image.

For more information on how to perform the **Undo & Redo** changes and how to use the **Save changes & Close** buttons, please refer to the Saving changes topic.

4.8.6.2 Image editing

Here you will learn how to edit the selected image. Besides, you will learn how to view the image properties:

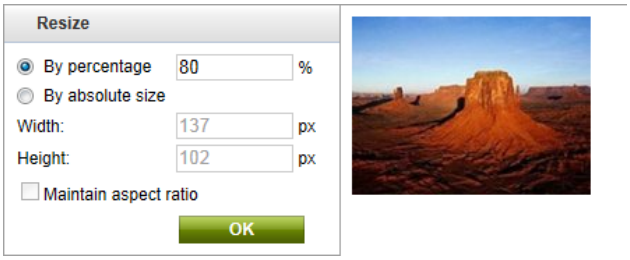
- Resize
- Rotation
- Convert
- Crop
- Color
- Properties

Each action is performed after clicking the **OK** button. Please refer to the Saving changes topic for more

details.

Resize

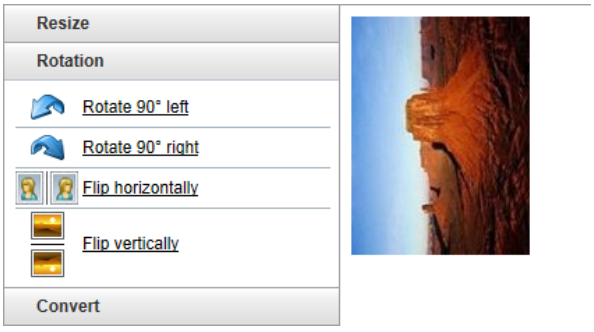
The **Resize** action lets you change the size of the image. The image is either enlarged or reduced to the size you specify. You may specify either a relative percentage to resize the image by, or an absolute pixel size. If the **Maintain aspect ratio** option is checked, then typing a new **Width** value will maintain a proportional **Height** value, and vice versa.



- **By percentage** - resizes the image to the entered percentage of side size.
- **By absolute size** - resizes the image to the size specified in the **Width** and **Height** fields.
  - **Width** - the width size of the image; in pixels.
  - **Height** - the height size of the image; in pixels.
  - **Maintain aspect ratio** - if checked, proportional values of the image dimensions will be maintained.

Rotation

The **Rotation** action commands enable you to rotate the image by 90 degrees (in either the clockwise or counter-clockwise directions) and to flip the image, both horizontally and vertically.

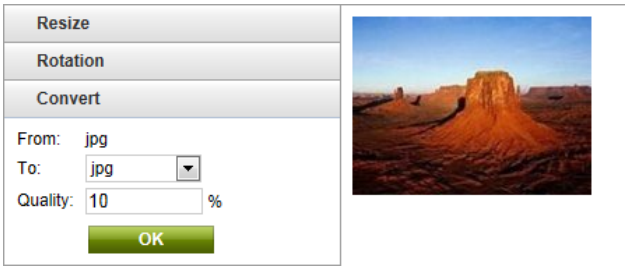


- **Rotate 90° left** - rotates the image to the left by 90 degrees.
- **Rotate 90° right** - rotates the image to the right by 90 degrees.
- **Flip horizontally** - flips the image horizontally.
- **Flip vertically** - flips the image vertically.

Convert

The **Convert** action enables you to transform the image file from one format to another. The **From** field of the action dialog tab is always filled in automatically according to the current format of the edited image. Please note that for the **.jpeg** image file type, you can also define image quality by entering the

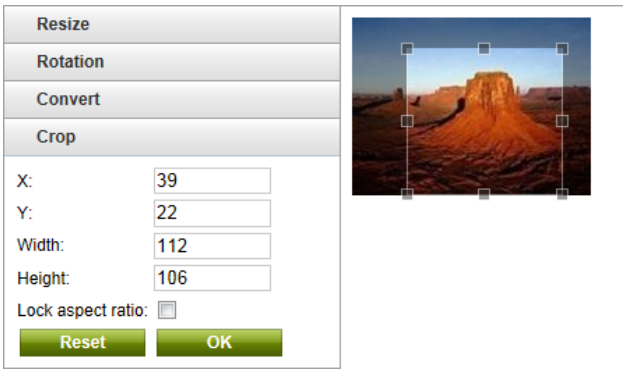
image quality percentage into the **Quality** field.



- **From** - the source format of the image.
- **To** - the target format of the image; **.bmp**, **.gif**, **.png** and **.jpg** image formats are supported.
- **Quality** - the quality of compression; applicable only for **.jpg** conversion.

Crop

The **Crop** action enables you to remove portions of the image to create focus or strengthen the composition. Image size is reduced without changing image resolution, so that the edited image is not distorted. Using Kentico CMS Image editor, you can crop images either manually by entering the coordinates of the upper-left point of the crop area rectangle and its width and height or you can move the mouse over the area of focus. The crop area rectangle can be resized in any direction unless the **Lock aspect ratio** option is checked. If checked, proportions of the currently selected crop area rectangle are maintained while resizing.



- **X** - the X coordinate of the upper-left point of the crop area rectangle.
- **Y** - the Y coordinate of the upper-left point of the crop area rectangle.
- **Width** - the width of the crop area rectangle; in pixels.
- **Height** - the height of the crop area rectangle; in pixels.
- **Lock aspect ratio** - if checked, the crop area rectangle will keep its current proportions while resizing it in any direction.
  - The same functionality can be achieved by holding the SHIFT key while moving the mouse. If the **Lock aspect ratio** option is checked while performing this operation, the result is inverse.
  - Pressing the CTRL key will result in returning the square of the currently selected rectangle area.

Please note that the crop area rectangle can be moved within the crop area by moving the mouse inside the rectangle.

Click **OK** to confirm the immediate change or click **Reset** or anywhere within the edited image outside the crop area rectangle to clear all the **Crop** action input fields.

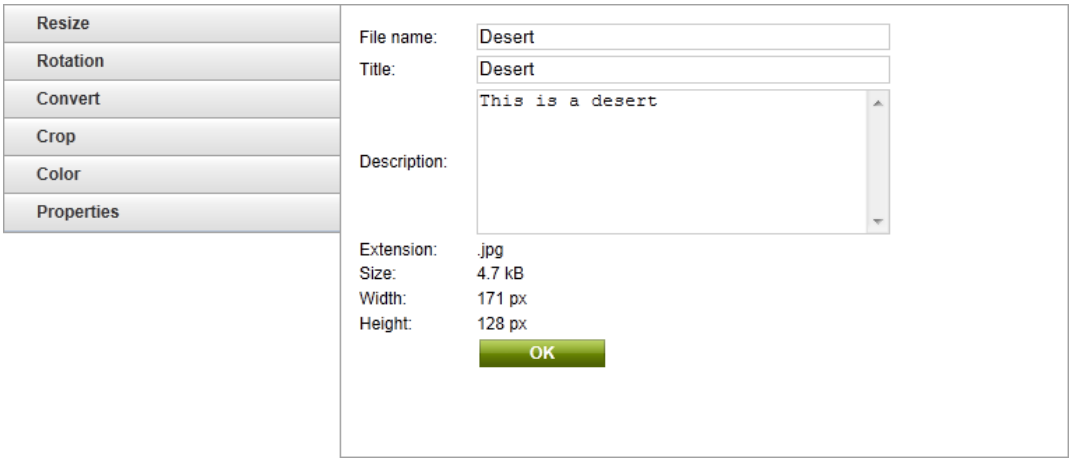
Color

The **Color** action lets you easily convert the image from color to grayscale.

Convert to grayscale - clicking this link results in converting the image from color to grayscale.

Properties

The **Properties** tab displays important information about the image such as its file name, title, description, extension, size, width and height. The **File name**, **Title** and **Description** fields are editable, enabling you to change the respective properties of the image.



- **File name** - the file name of the image.
- **Title** - the title of the image; image metadata that can be used e.g. in transformations.
- **Description** - the description of the image; image metadata that can be used e.g. in transformations.
- **Extension** - the file type extension of the image.
- **Size** - the size of the image; in kB.
- **Width** - the width of the image; in pixels.
- **Height** - the height of the image; in pixels.

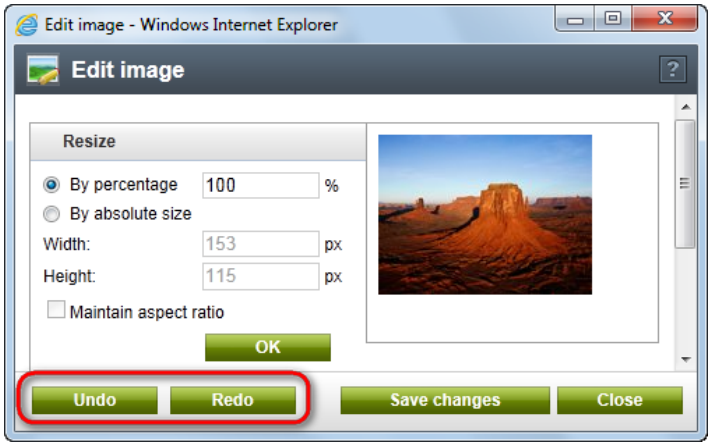
Please note

If you decide to leave the Image editor without saving the changes you have made (by clicking the **Close** button), a message window will pop up informing you that all changes will be lost.

4.8.6.3 Saving changes

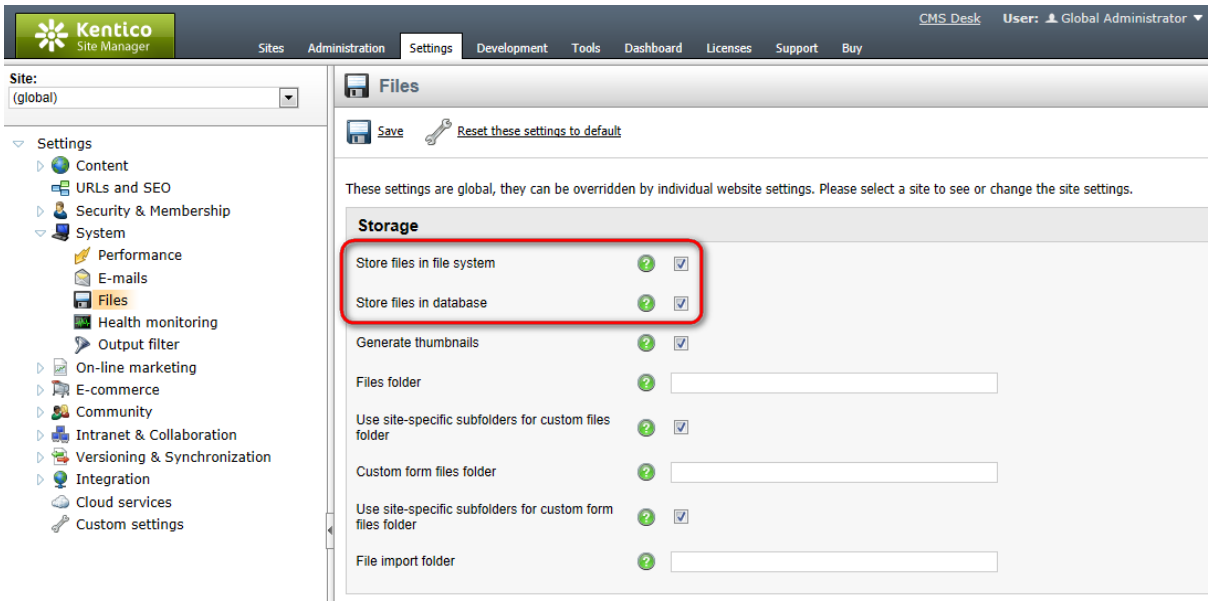
Undo and Redo

The **Undo** and **Redo** actions allow you to undo and redo up to 1000 changes made while editing the image. After each partial change, a temporary version of the image is created and stored in *Application URL/App_Data/CMSTemp/ImageEditor/First two letters of image editor instance GUID/Image editor instance GUID*, the database or in both; see Storing image versions. By clicking the **Undo** and **Redo** buttons, you can view the stored separate versions of the image. By clicking the **Save changes** button, the original image is replaced by the currently edited version of the image.

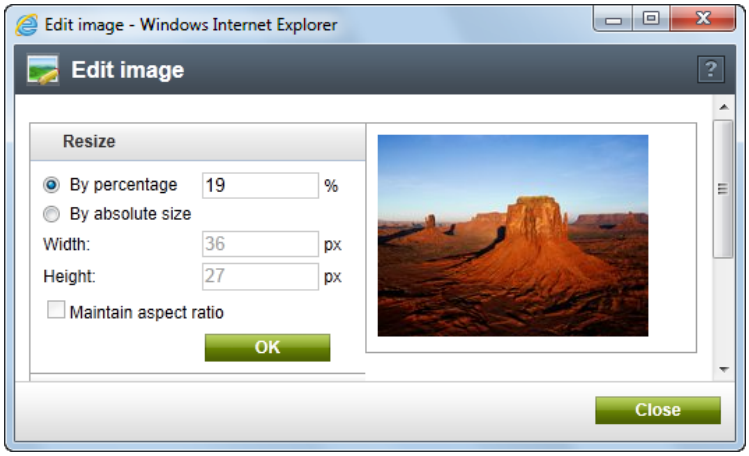


Storing image versions

Separate image versions containing changes of the image are stored either in the database or on the disk, as set in **Site Manager -> Settings -> System -> Files**.

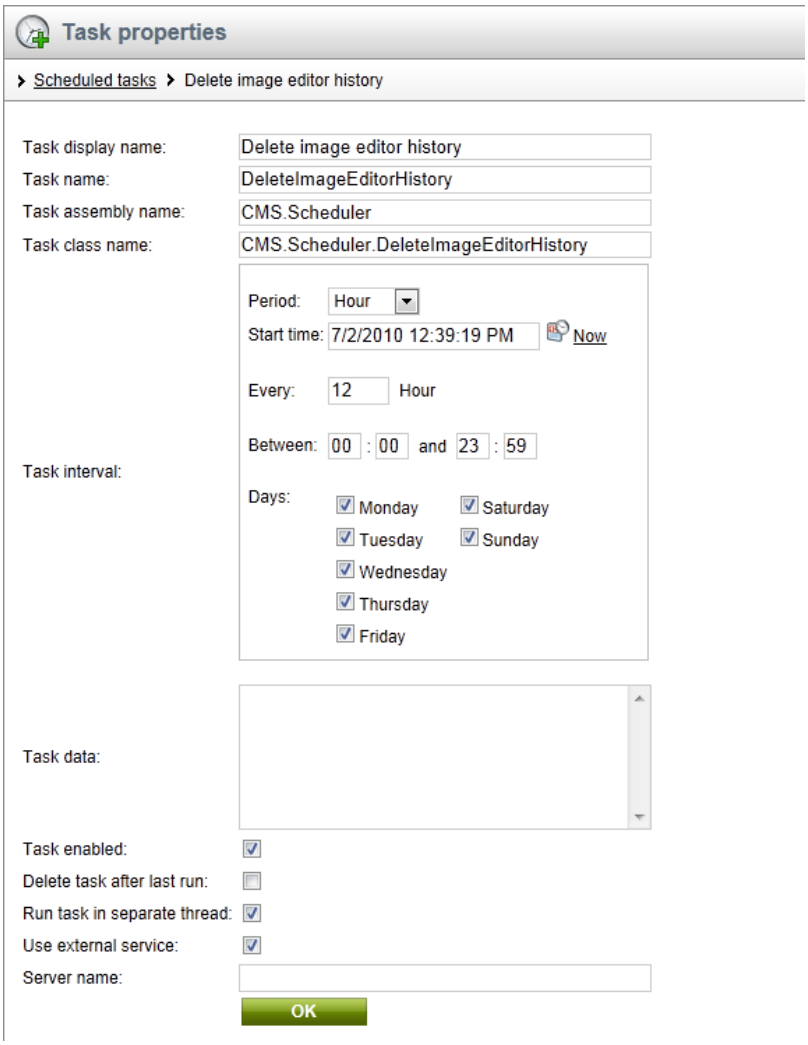


Please note that if only the storing of files in the file system is enabled and permission to write to the local disk is missing, the **Undo**, **Redo** and **Save changes** buttons are hidden and the changes are applied immediately to the original image.



Deleting image versions

If you decide to leave the Image editor by clicking the **Close** button, all image versions files from the current editing session will be deleted, regardless of their storing location. However, if you happen to finish editing in a non-standard way, e.g. by quitting the browser, the files will remain stored (both in the file system and in the database) until a global scheduled task performs their deletion.
You can change the settings of this global task in **Site Manager -> Administration -> Scheduled tasks**. Select **(global)** from the **Site** drop-down list and choose to **Edit** (✎) the **Delete image editor history** scheduled task. Change the settings of the schedule as required and click **OK**.



By default, this task deletes all image versions older than 24 hours. If you would like to change this, you need to alter the value in the settings key of the global task. This key is contained in the **web.config** file placed in your Kentico CMS installation folder.

- ***<add key="CMSDeleteImageEditorHistoryOlderThan" value="24"/>*** - the value is in hours, the default value is 24.

For more details on how to configure scheduled tasks, please refer to the Scheduler chapter in the **Development** section of this guide.

4.8.7 Metadata editor

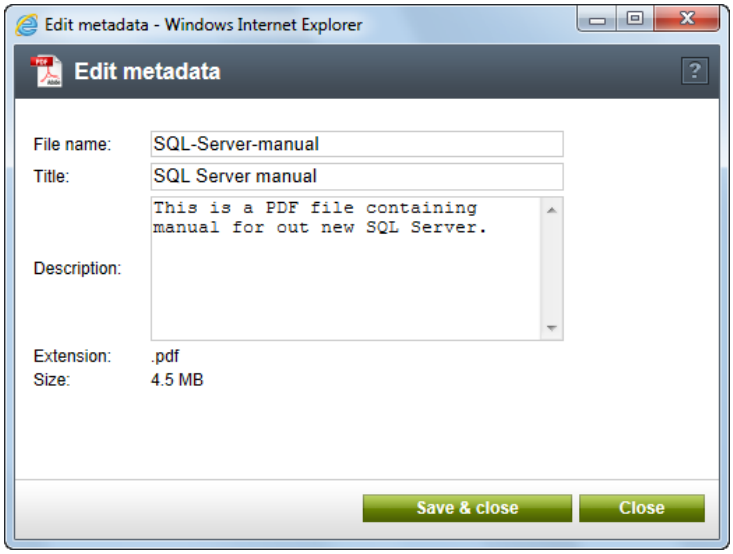
The Metadata editor can be used to edit metadata (i.e. descriptions) of non-image files stored in Media libraries, as document attachments or as object metafiles. The dialog is always accessible by clicking the **Edit** (✎) icon next to the file listed in the respective part of the UI.

The following metadata can be entered:

- **File name** - name of the file (without the trailing dot and extension). The current file name is always

pre-filled when the dialog is opened.

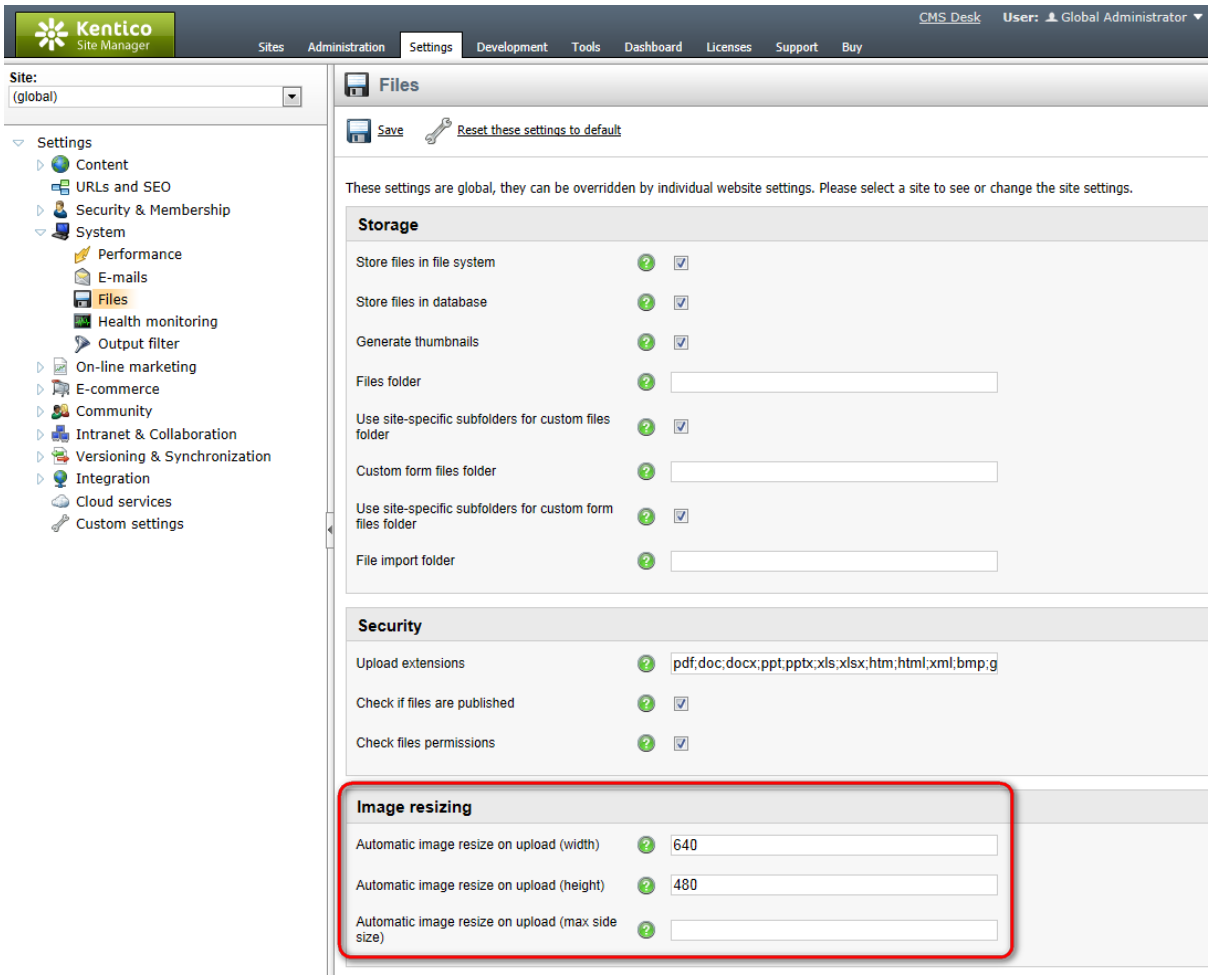
- **Title** - title of the file (may be different from the actual file name).
- **Description** - text describing the content of the file.



4.8.8 Resizing images on upload

You can set global image resizing parameters in **Site Manager -> Settings -> System -> Files -> Image resizing** by entering the following values:

- **Automatic image resize on upload (width, height, max side size)** - depending on which values you fill in, the functionality is the following when uploading images:
  - **No values are entered** - images are not resized.
  - **Only width or only height** - images are resized so that their width or height matches the entered value. The other dimension is also resized so that aspect ratio is preserved.
  - **Both width and height** - images are resized so that the larger dimension matches the respective entered value. Aspect ratio is preserved.
  - **Max side size** - if one of the image's sides is larger than this value, the image is resized so that its larger side's dimension matches the entered value. Aspect ratio is preserved and width and height settings are not applied.



These settings are applied by default when uploading images as:

- **media library files**
- **document attachments**
- **CMS.File documents**
- **Editable text web part** - uploading images using the WYSIWYG editor dialogs
- **Editable image web part** - uploading images using the **Select image** dialog
- **Field editor fields** - uploading images using the following field types (form controls), e.g. in forms, Document type field editor, etc.
  - **File** attribute type -> **Upload file** and **Direct uploader** form controls
  - **Text** attribute type -> **HTML Area (Formatted Fext)**, **BBcode editor**, **Image selection**, **Media selection** and **File selection** form controls

The default settings defined here can be overridden by local settings in the particular parts of the user interface (e.g. web part properties, field editor, WYSIWYG editor dialogs, etc.).

4.8.9 Uploading multiple files at once

Files in Kentico CMS can be uploaded either one by one or as multiple files. This can be useful if you need to upload a greater number of files at once or if you need to perform the task in the quickest possible way. Upload of multiple files at once is available for:

- **media files** - files in a media library (e.g. CMS Desk -> Tools -> Media -> Edit (✎) *media library* -> Files tab).
- **meta files** - files related to a specified object (e.g. e-mail template attachments in CMS Desk -> Administration -> E-mail templates -> Edit (✎) *e-mail template* -> Attachments).
- document attachments - files attached to a document (e.g. CMS Desk -> Content -> *document* -> Properties tab -> Attachments tab).

Enabling upload of multiple files

To enable upload of multiple files in the whole CMS the following conditions must be met concurrently:

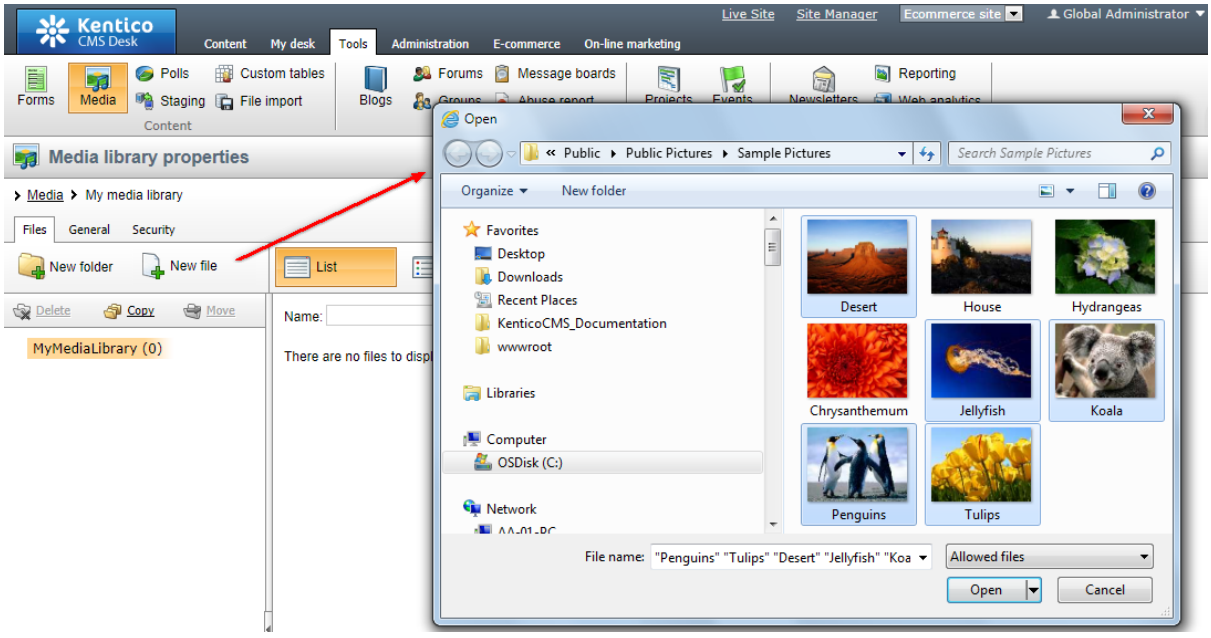
1. The value of the ***CMSUseSilverlightUploader*** key is set to TRUE or the key is not added in the *appSettings* section of the *web.config* file (because the upload of multiple files is the default option).

2. The CMS application has the write-to-disk permission for the temporary files repository, i.e. for **web project\App_Data\CMSTemp\MultiFileUploader**.

3. Microsoft Silverlight is installed on your client computer.

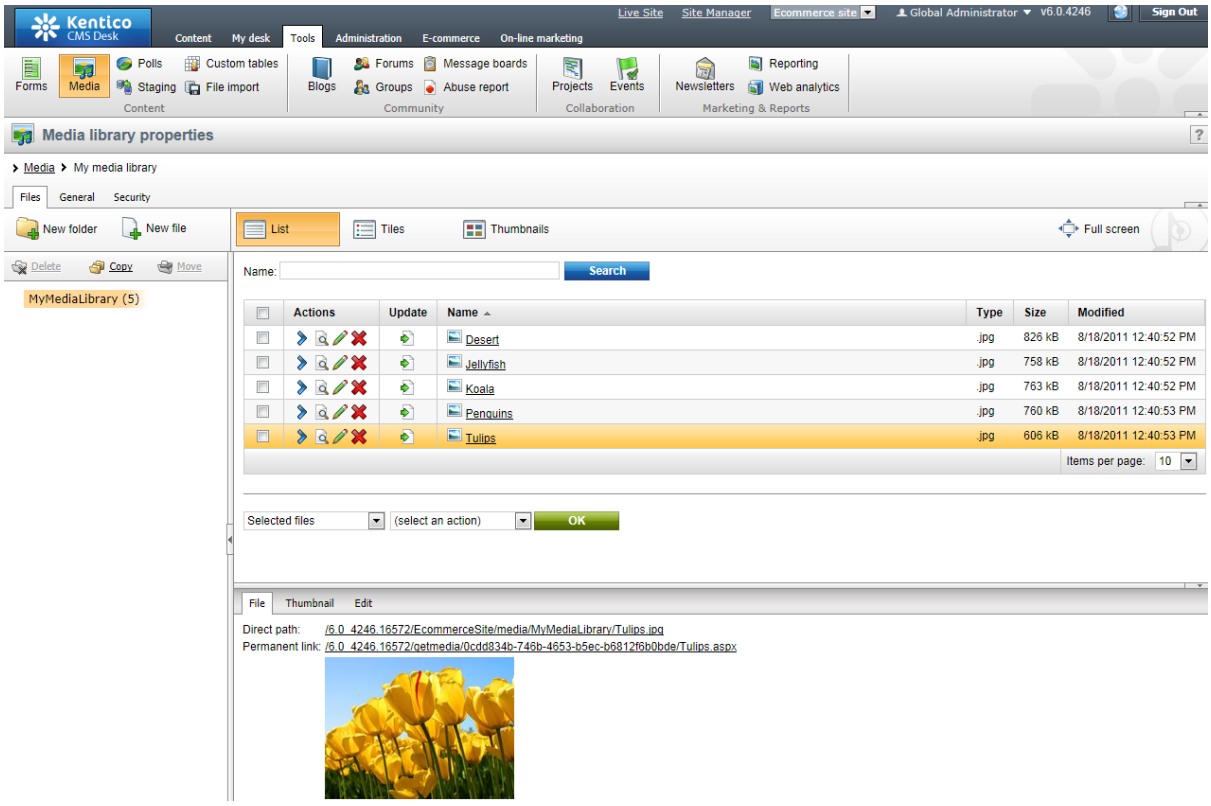
How it works

1. First you need to select one or more files from a given location.



2. The files are downloaded as temporary files to a temporary repository (*App_Data\CMSTemp\MultiFileUploader*). However, to upload the files successfully, you need to avoid POST BACK on the current page during the upload.

3. When the upload is finished, the temporary files become the requested files:



Please note

If you don't avoid POST BACK during the upload of multiple files or some other error occurs, the temporary files remain in the temporary repository.
That's why there is the **Delete old temporary upload files** scheduled task (configurable in Site Manager -> Administration -> Scheduled tasks), which deletes these temporary files, by default older than 24h. However, you can change this setting by changing the value of the **CMSDeleteTemporaryUploadFilesOlderThan** key (please note that you need to manually add the key into the *web.config* file in the current web project folder).

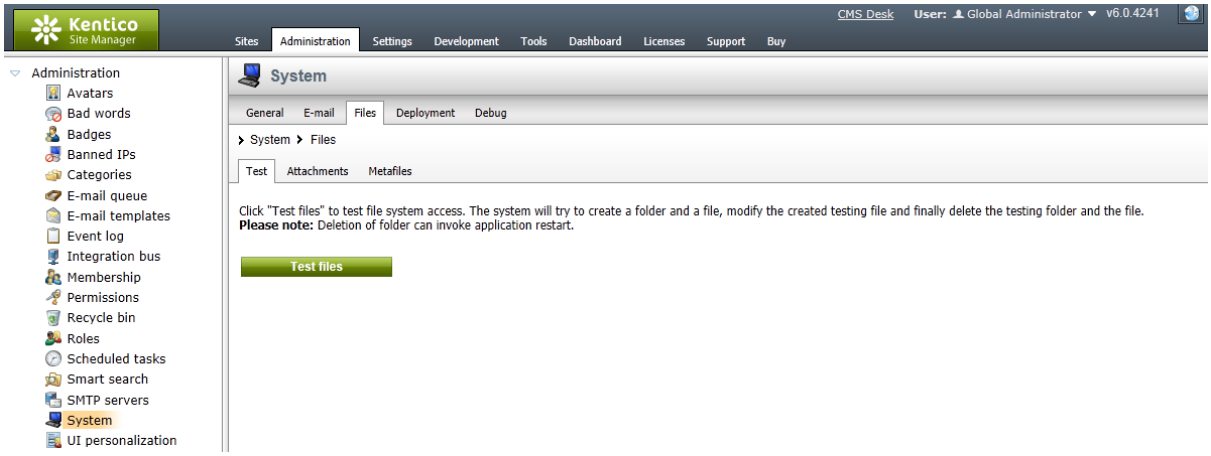
4.8.10 File administration UI

A user interface for global-level administration of files stored by the CMS is located on the **Files** tab in **Site Manager -> Administration -> System**. This part of the user interface is divided into three tabs described in the following text.

Test tab

On the **Test** tab, you can perform a test that verifies that Kentico CMS has access to the file system. The test consists of creation of a folder and a file, modification of the created file and deletion of the folder and the file. To perform the test, just click the **Test files** button.

If the test detects that Kentico CMS is not able to access the file system, you can find a solution to this problem in the Installation and deployment -> Troubleshooting installation issues -> Disk permissions problems chapter of this guide.



Attachments tab

On the **Attachments** tab, you can find a list of document attachments stored by the CMS. Based on selection in the **Site** drop-down list, you can either select a site and view all document attachments on the site, or choose **(all)** to view all document attachments on all websites in the system.

If there are more than 25 document attachments (or a different number set by means of the *CMSDefaultListingFilterLimit web.config* key), a filter is displayed above the grid. Using the filter, you can filter displayed attachments by their name, extension, size or depending on if they are stored in the database.

In the **Actions** column in the grid, action icons are displayed depending on the **Store files in file system** and **Store files in database** settings in **Site Manager -> Settings -> System -> Files**. The displayed actions are the following:

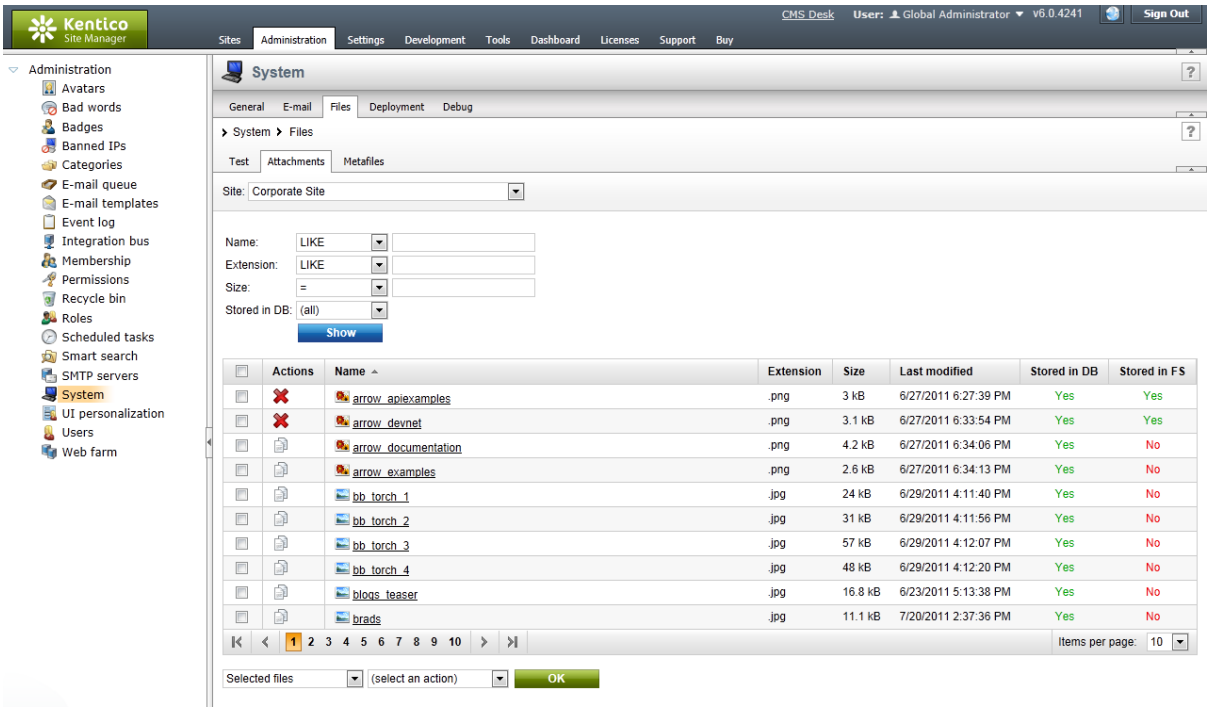
- **Copy to database** - copies the attachment to the database. This action is only displayed if the attachment is stored only in the file system while settings are configured for files to be stored both in the file system and in the database.
- **Delete from database** - deletes the attachment from the database. This action is only displayed if the attachment is stored both in the file system and in the database while settings are configured for files to be stored only in the file system or to be stored both in the file system and in the database.

- **Copy to file system** - copies the attachment to the file system. This action is only displayed if the attachment is stored only in the database while settings are configured for files to be stored only in the file system or both in the file system and in the database.
- **Delete from file system** - deletes the attachment from the file system. This action is only displayed if the attachment is stored both in the file system and in the database while settings are configured for files to be stored only in the database.

Using the two drop-down lists below the grid, you can perform the above listed actions for multiple attachments at once. In the first drop-down list, you can choose from the following:

- **Selected files** - performs the action for files selected using the check-boxes (☑) in the grid.
- **All files** - performs the action for all listed files.

Then you can choose the required action from the second drop-down list and click **OK** to perform it. The same actions as listed above are offered in the drop-down list, while the conditions described above need to be met for the actions to be offered.



Metafiles tab

On the **Metafiles** tab, you can find a list of metafiles stored by the CMS. Based on selection in the **Site** drop-down list, you can select:

- **(all)** - displays all metafiles stored by the system.
- **(global)** - displays metafiles of global, i.e. not site-related objects.
- **<website>** - displays metafiles of site-related objects belonging to the selected site.

Using the **Object type** drop-down list, you can further limit which metafiles will be displayed. By selecting **(all)**, you get all metafiles that match the selection in the drop-down list above. The other options in the drop-down are particular object types, while choosing one displays only metafiles of these objects that match the selection in the drop-down list above.

If there are more than 25 metafiles (or a different number set by means of the *CMSDefaultListingFilterLimit web.config* key), a filter is displayed above the grid. Using the filter, you can filter displayed metafiles by their name, extension, size or depending on if they are stored in the database.

In the **Actions** column in the grid, action icons are displayed depending on the **Store files in file system** and **Store files in database** settings in **Site Manager -> Settings -> System -> Files**. The displayed actions are the following:

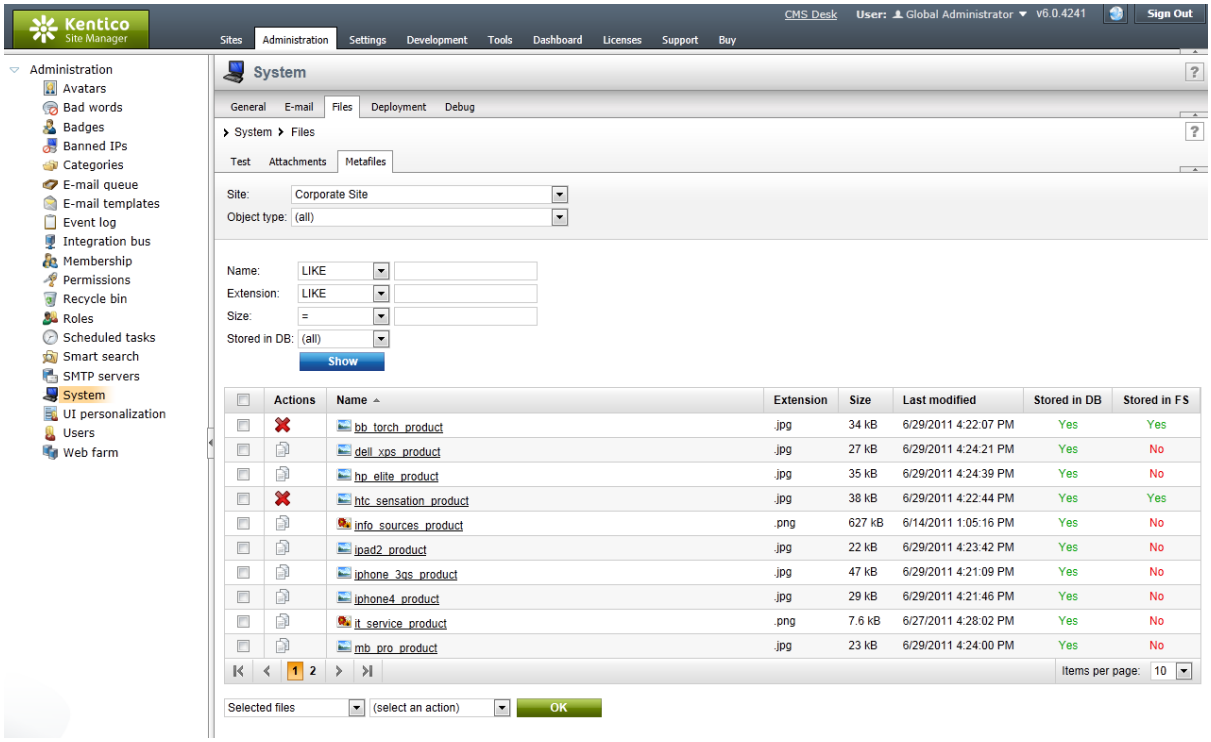
- **Copy to database** - copies the metafile to the database. This action is only displayed if the metafile is stored only in the file system while settings are configured for files to be stored both in the file system and in the database.
- **Delete from database** - deletes the metafile from the database. This action is only displayed if the metafile is stored both in the file system and in the database while settings are configured for files to be stored only in the file system or to be stored both in the file system and in the database.

- **Copy to file system** - copies the metafile to the file system. This action is only displayed if the metafile is stored only in the database while settings are configured for files to be stored only in the file system or both in the file system and in the database.
- **Delete from file system** - deletes the metafile from the file system. This action is only displayed if the metafile is stored both in the file system and in the database while settings are configured for files to be stored only in the database.

Using the two drop-down lists below the grid, you can perform the above listed actions for multiple metafiles at once. In the first drop-down list, you can choose from the following:

- **Selected files** - performs the action for files selected using the check-boxes ( ) in the grid.
- **All files** - performs the action for all listed files.

Then you can choose the required action from the second drop-down list and click **OK** to perform it. The same actions as listed above are offered in the drop-down list, while the conditions described above need to be met for the actions to be offered.



4.9 Multilingual content

4.9.1 Overview

Kentico CMS websites can host content in multiple languages (cultures). Multilingual websites have separate versions of documents for each language. The system allows you to automatically offer particular culture versions of the website to visitors based on various settings. Users may also switch between individual languages manually using a dedicated web part.

You can find a list of the available content cultures in **Site Manager -> Development -> Cultures**. All major cultures are provided by default. You can assign these cultures to websites and then translate the content to the respective languages.

The administration interface of Kentico CMS can also be multilingual. To learn more, refer to the Development -> UI cultures and localization chapter of this guide.

Topics:

- Configuring multilingual content
- Using language-specific page templates
- Selecting default languages for visitors
- Languages and URLs
- Managing multilingual websites
- Translation services

4.9.2 Configuring multilingual content

This topic explains how to configure a website to display content in multiple languages.

Setting the default content culture

When you create a new website, its default culture is the one specified in Step 2 of the New site wizard. If you create a website based on a web template, it uses the culture of the given web template. All default Kentico CMS web templates use **English - United States** as their default culture.

Default culture settings can be modified in the following sections of the user interface:

- **Site Manager -> Settings -> Content -> Default content culture**
- **Site Manager -> Sites -> edit () -> Default content culture**

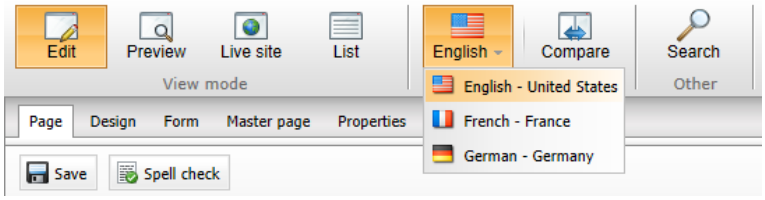
The two settings are interlinked, i.e. values configured in one section are reflected in the other one.

Configuring multilingual content

The following example demonstrates how to add extra language versions to documents on the sample Corporate Site:

1. Go to **Site Manager -> Sites** and click the **Edit site** () icon next to the **Corporate Site**. Then open the **Cultures** tab and click the **Add cultures** button. In the selection dialog, choose the **French - France** and **German - Germany** cultures and click **OK**.

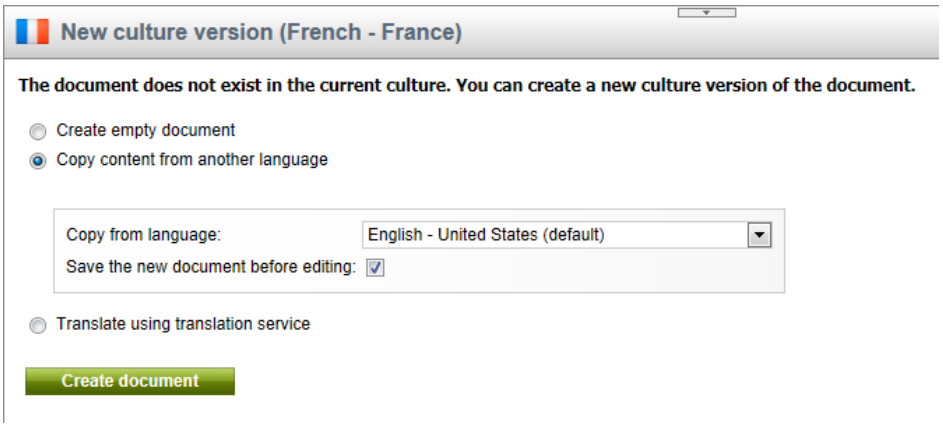
2. Open CMS Desk. You can see there is a new section with language selection options in the main toolbar. Using this toolbar, you can switch between particular language versions of the currently selected document. You can also click the **Compare** ( ) button to switch to Language version comparison mode and edit language versions side-by-side.



3. Select the root document in the content tree and switch to the **French** culture. Since the French version of the document does not exist yet, you need to create it through one of the following ways:

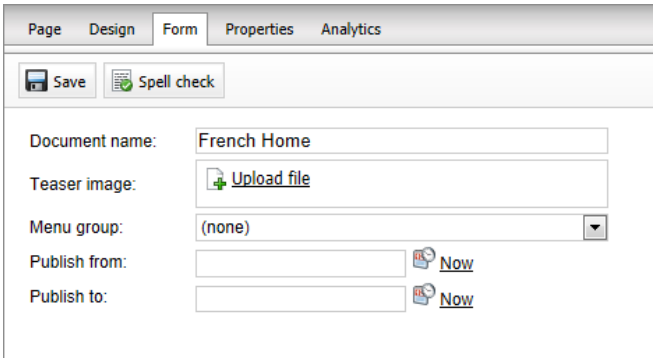
- **Create empty document** - creates a new document of the same type, but without any content.
- **Copy content from another language** - creates a copy of the document with content loaded from the selected language version. If *Save the new document before editing* is checked, the system adds the new document version and immediately saves the copied content.
- **Translate using translation service** - with this option, the content of the document's new language version will be provided by an external translation service. See the Translation services sub-chapter to learn more.

Choose to copy content from the English version and click **Create document**.

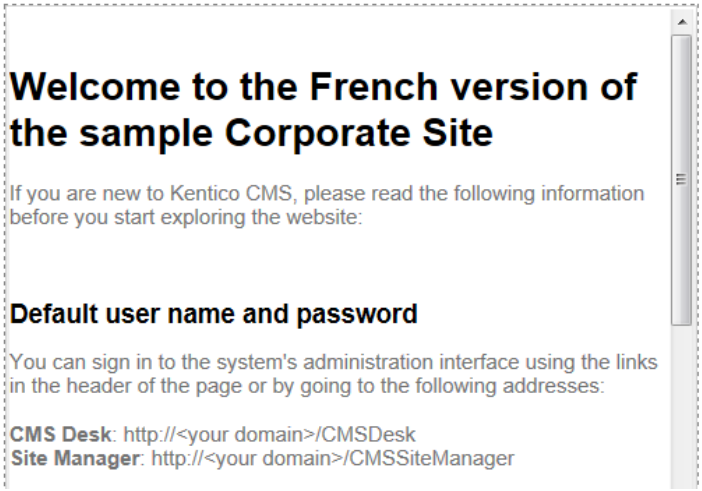


The French version of the root document has now been created.

4. Repeat the same for the **Home** page, but this time disable the **Save the new document before editing** option before creating the document. The editing form of the new document version opens. Here, change the **Document name** to *French Home* and click **Save**.



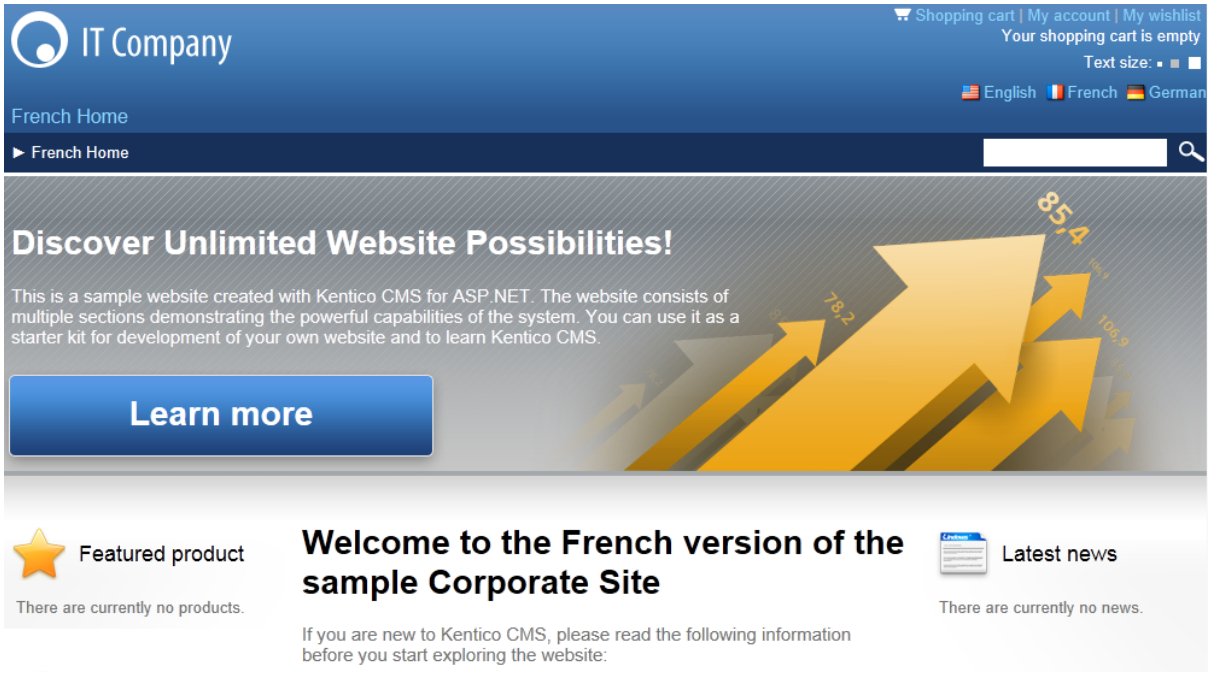
5. Switch to the **Page** tab and change the text of the heading in the second editable region to *Welcome to the French version of the sample Corporate Site*. Click **Save**.



This step shows that you can modify the content of a page's editable regions to be different for each

specific language version. It is also possible to translate the content entered on a document's **Form** tab, and even assign culture specific values for many of its **Properties**.

6. Sign out of CMS Desk and you will be able to see the French version of the live site.



You can switch between languages using the language selection links at the top right of the website header. They are displayed automatically using the Language selection with flags web part.

7. Sign in as an administrator to **Site Manager** and go to **Settings -> Content**. Choose the Corporate Site in the drop-down list and enable the **Combine with default culture** checkbox. Click **Save**. Sign out and view the French version of the live website again. The documents that are not available in French are now displayed in the default culture (English).

Creating content for another language

When adding a new language to the website, please be sure to always create a language version for at least the **root** document and the default **home page**. Otherwise, the website will not be displayed correctly.

The system adds a cross (✖) icon next to documents in the content tree that do not exist in the currently selected culture. This allows you to can see which parts of the site still need to be translated.

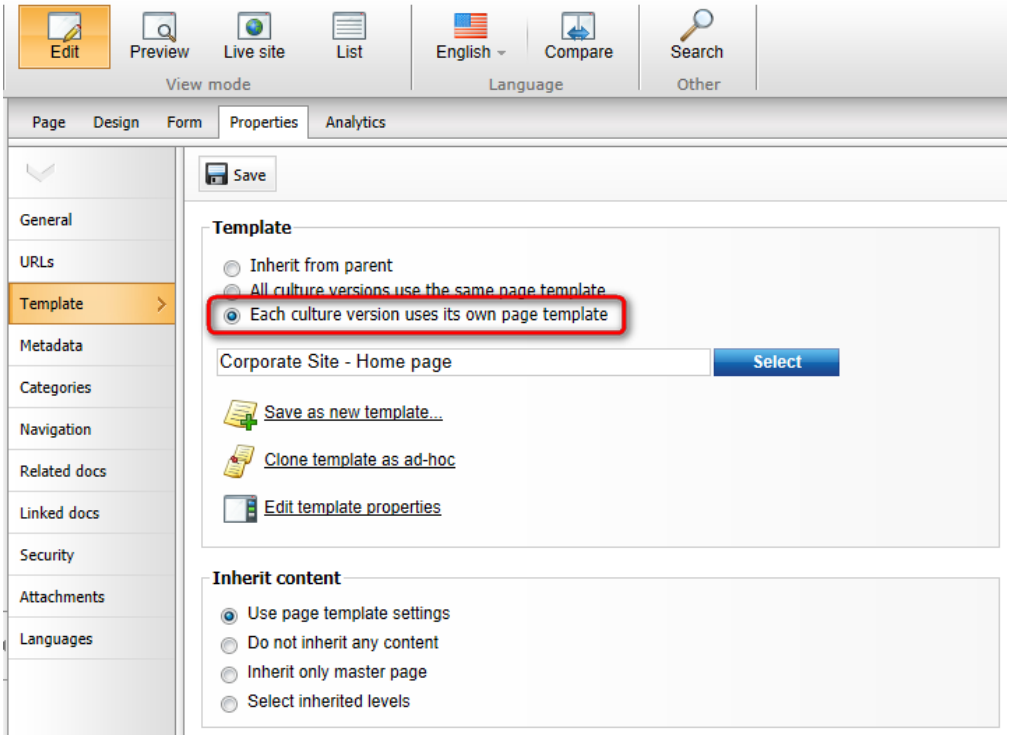
4.9.3 Using language-specific page templates

When creating a new language version of a document, it should usually be enough to edit the original document on the level of page content, i.e. translate the text on its **Page** or **Form** tabs. However, you can also make major changes to the basic layout and design of each culture version by using a different page template. By default, documents share the same page template for all of their language versions.

The following steps demonstrate how to assign a unique page template to a specific language version of a document and then modify it as required. This continues the example from the Configuring multilingual content topic, so there should already be some additional cultures added to the website and a French version of the Home page.

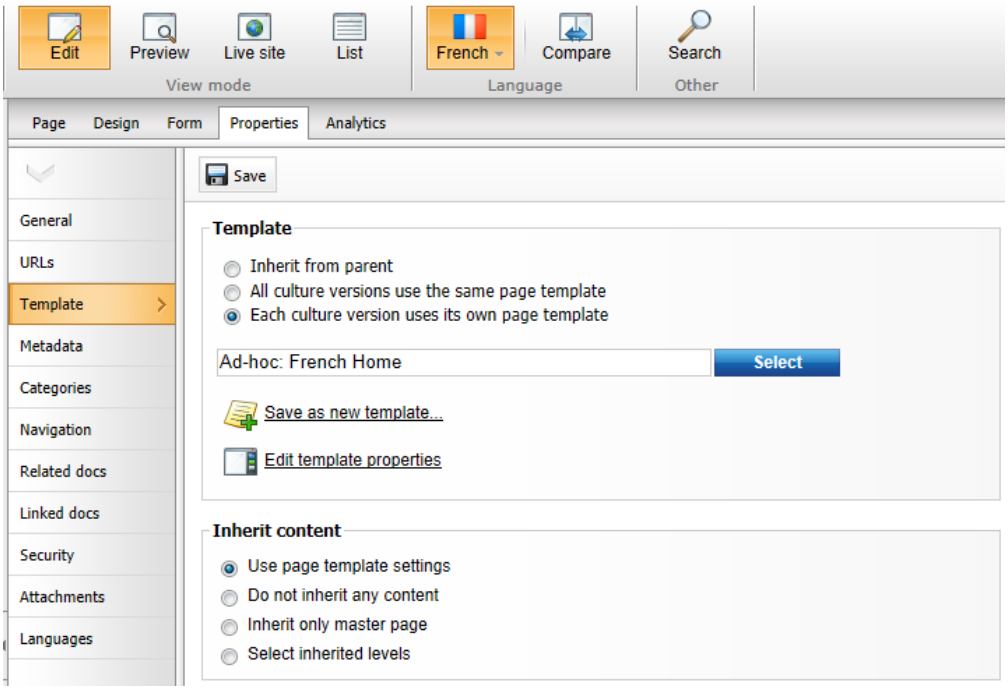
1. Open CMS Desk on the **Content** tab and select the **Home** page in the content tree. Make sure you are in **Edit** mode and switch to the **Properties -> Template** tab. Here, choose the **Each culture**

version uses its own page template option for the document and click **Save**.



This setting is shared between all versions of the document, so it is not important which language is currently selected.

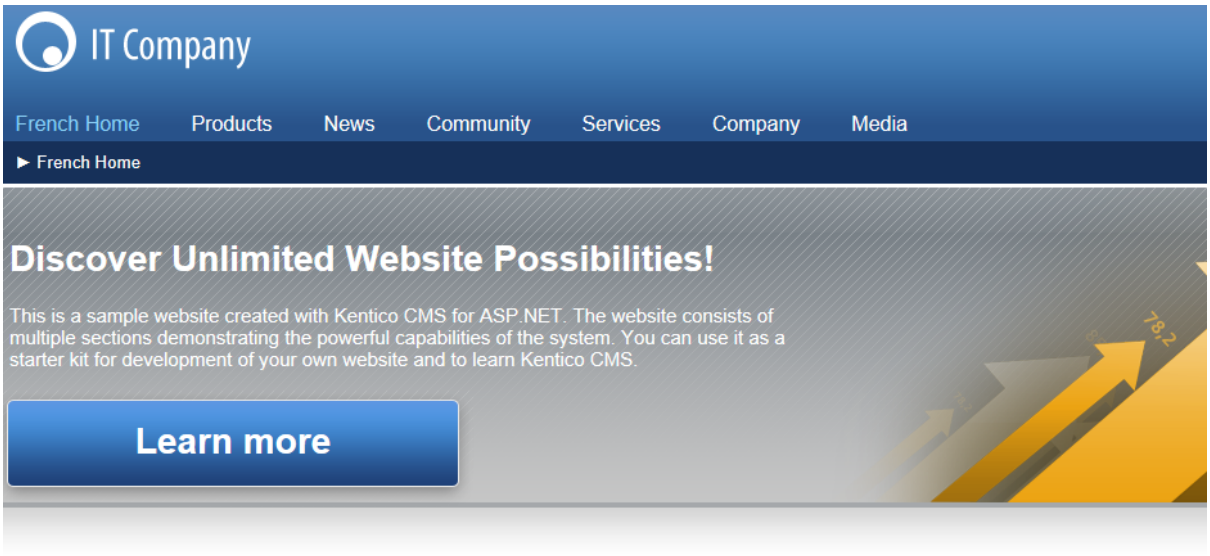
2. Switch to the French language and click the **Clone template as ad-hoc** action. This creates a separate copy of the current Home page template and assigns it to the French version of the document. You can now change the design of the French page without affecting the other language versions.



Cloning the template used by the default culture version provides a convenient starting point from which you can quickly make modifications, but it is also possible to select a completely different template if necessary.

3. Open the page's **Design** tab and add (+) a new **Text & Images -> Editable text** web part into the **Actions zone**. Then switch to the **Page** tab and you can notice there is an additional editable region. Cut the header text from the bottom region and paste it into the empty region that was added. Click **Save** to confirm the change.

4. Sign out and view the Home page in the French language. The header text is now aligned on the left and placed above the remaining content.



If you switch to English, the page is unchanged, because it still uses the original template.

This is only a very simple example of a modification that can be done when the language version has a separate page template. There are many other possibilities, such as setting different properties for web parts or editing the overall layout of the page.

4.9.4 Selecting default languages for visitors

You can configure which language version the system displays to users when they arrive on the website.

Configuration for registered users

The website culture displayed to registered website users depends on each user's **Preferred content culture** setting. Users themselves can change this setting in **CMS Desk -> My desk -> My profile**, or on the live site by means of the My account web part. Website administrators can adjust these settings in **Site Manager -> Administration -> Users -> edit (✎) -> General**.

Depending on this configuration, the user will either see the content in their preferred culture (if available), or in the website's default content culture (if set to *(default)* or if the preferred culture is not available).

Please note that the act of logging in does not change the current culture, so this setting only has an

effect if users are already logged in when they first reach the website (e.g. when using Windows authentication).

Configuration for anonymous visitors

It is also possible to configure the default content culture displayed to anonymous visitors of the website. This can be done separately for the main website domain and for individual domain aliases:

- **Site Manager -> Sites -> edit () site -> General -> Visitor culture** - used for visitors accessing the website through a URL containing the main domain name.
- **Site Manager -> Sites -> edit () site -> Domain Aliases -> edit () alias -> Visitor culture** - used for visitors accessing the website through a URL that contains the respective domain alias.

The values can either be set to one of the cultures available for the website, or to **(Automatic)**, in which case the **user's browser settings** are used (for example, in Internet Explorer, you can set the default language in Tools -> Internet options -> General -> Languages).

Setting the default language for specific domains

Using the **Visitor culture** settings of the site and its domain aliases (described in the section above), you can configure the system to use a different default language based on the current domain. For example, if you use the following domains:

- example.com (main domain)
- example.de (domain alias)
- example.fr (domain alias)

You can set English, German and French as the default cultures for the domain and domain aliases (respectively). Now, when a visitor arrives at *example.fr*, the website automatically displays its French content. Refer to the Languages and URLs topic for additional information about cultures, domains and URLs.

4.9.5 Languages and URLs

On multilingual websites, all language versions of a page use the same URL by default. This URL is based on the given document's alias path. For example, the home page always has the following URL: */Home.aspx*

If you want to see the same page in French, you need to change the culture by entering the URL with an appropriate query string parameter: */Home.aspx?lang=fr-fr*

Once the language is selected this way, the system stores it in the visitor's browser using a cookie. This means that the given visitor automatically sees the French version when returning to */Home.aspx* or another translated page, even without any parameters in the URL.

Having to type a URL with a query string manually would be inconvenient for visitors, so you can provide a more user-friendly way to switch between cultures by placing one of the following language selection web parts onto the pages of your website:

- Language selection
- Language selection drop-down

- Language selection with flags

These web parts automatically generate appropriate links to the language versions of the current document.

Renaming the language selection parameter

The default name of the language selection parameter is *lang*.

To rename the parameter:

1. Add the **CMSLanguageParameterName** key into the */configuration/appSettings* section of your application's **web.config** file.

2. Enter the new name as the key's **value**.

```
<add key="CMSLanguageParameterName" value="sprache" />
```

Even though switching between languages using URL query string parameters is the default approach, it is generally not recommended for live production websites. Parameters make URLs more confusing and are not friendly toward search engines. Instead, you can configure the system to use one of the options described in the sections below.

Enforcing separate domains for languages

Having a different domain name for each culture is a good way to let visitors know that the particular version of the site is intended for a certain language audience. It is also the best option for multilingual websites with regard to Search engine optimization.

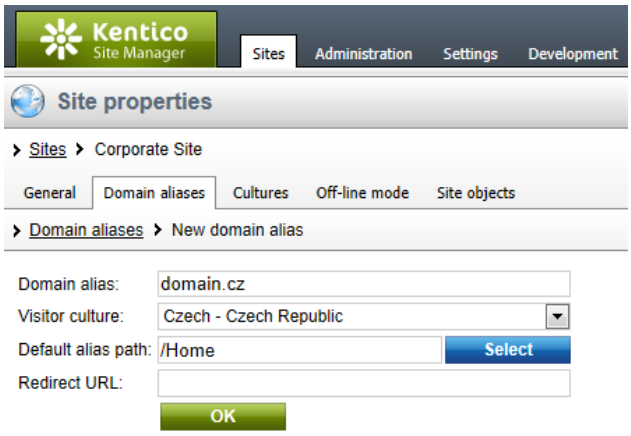
For example, if the English version of your site is available under *domain.com*, the French version could use *domain.fr* and so on. This scenario uses a different country-code top-level domain, but you can set any other domain name format, such as unique subdomains for each culture.

To implement domain separation based on languages:

1. Go to **Site Manager -> Sites** and edit () your site.

2. On the **General** tab, select your website's primary language in the **Visitor culture** field.
   - This assigns the given language to the site's main domain name.

3. Set up additional domain names for other languages as domain aliases on the **Domain aliases** tab.
   - Just like with the main domain, you can assign a language to each alias by selecting the matching culture in the **Visitor culture** field.



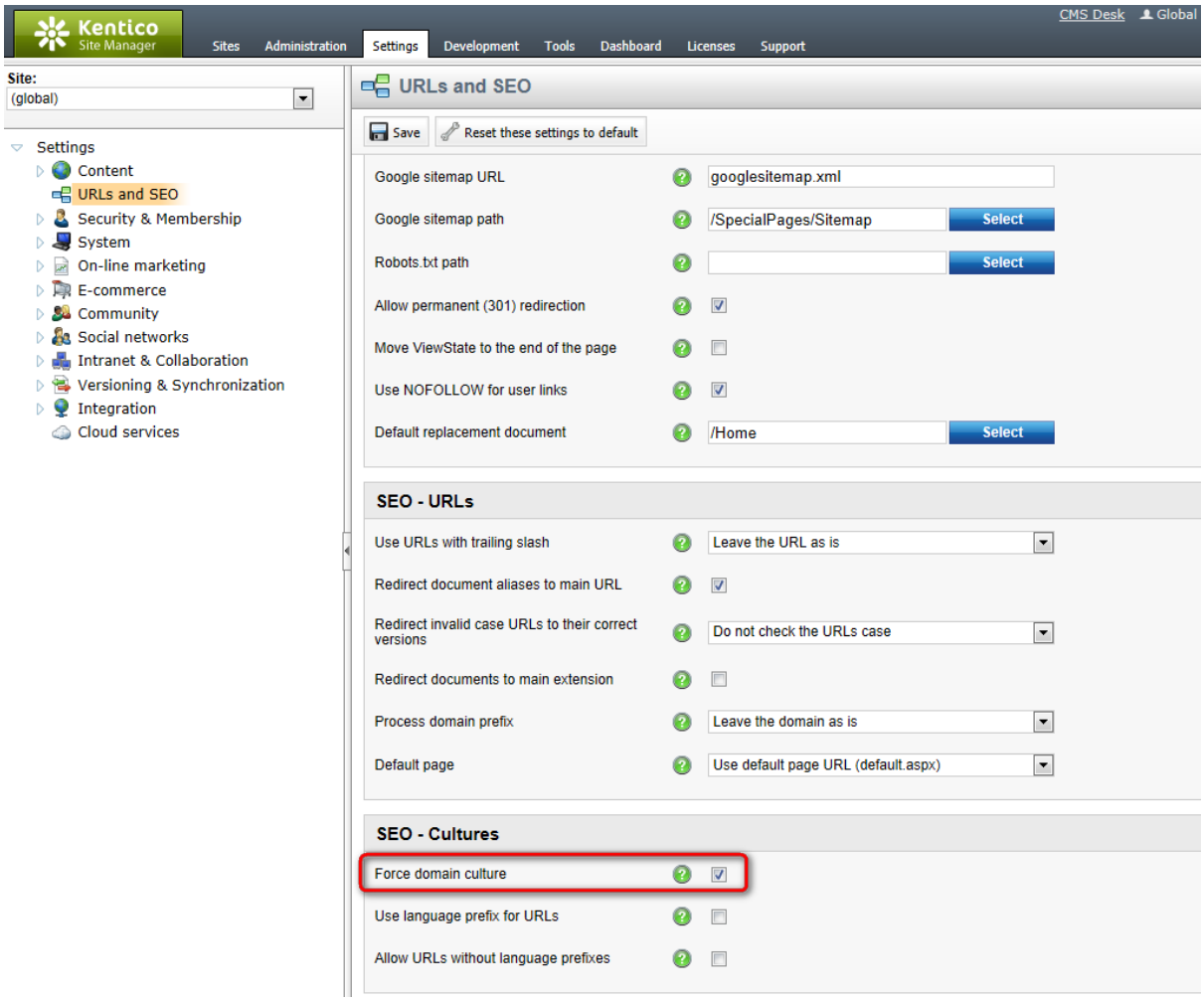
Configuration requirements

When setting up the **Visitor culture** of the site and its aliases, only assign each language to a single domain and do not use the *(Automatic)* value. Otherwise the system will not be able to determine with certainty which content culture to display.

Also make sure that your domain aliases do *not* use a **Redirect URL** leading to another domain name.

4. Go to **Site Manager -> Settings -> URLs and SEO** and enable the **Force domain culture** setting.

Note: You cannot use forced culture domains in combination with the language URL prefixes described in the next section.



The system now ensures the following:

- The default content culture is selected based on the domain name through which the website is opened.
- All documents URLs (e.g. in navigation elements) are generated using the domain name to which the current content culture is assigned.
- When a user selects a different language through one of the language selection web parts, the system redirects the URL to the corresponding domain name.

Authentication with multiple domains

The user context is not carried over when switching between domains on the live site. If you are using standard Forms authentication, registered users who change the content culture while browsing need to log in separately on each language version of the website.

If this is a problem, you can work around it by implementing a single sign-on mechanism for all domains.

Using language prefixes for URLs

If you do not own a different domain name for each language version of the website, but still want to ensure unique URLs for each culture without adding query string parameters, you can use language prefixes. Language prefixes insert a subdirectory into the URL in format **<domain>/<language prefix>/<URL path>**, for example: *<domain>/fr-FR/Home.aspx*

To enable language prefixes, go to **Site Manager -> Settings -> URLs and SEO**, and check the **Use language prefix for URLs** field.

Note: You cannot use language prefixes if the **Force domain culture** setting is enabled.

Unless you check the **Allow URLs without language prefixes** setting, the system automatically redirects all URLs without a language prefix to a corresponding URL that includes a language prefix (according to the current content culture).

If language prefixes are enabled, the language selection web parts used on your site generate links containing the appropriate URL prefix by default. If you wish to configure a specific web part to use links with the standard query string parameter, disable its **Use culture specific URLs** property. Please keep in mind that all other settings still apply, so the resulting URLs may be redirected.

Changing the language prefix text:

By default, the language prefix matches the culture code of the requested language. If a *culture alias* is set for the language, it takes precedence and the system uses the alias instead of the culture code. If you want to change the language prefix for a certain language, the best way is to set a culture alias.

For example, to change the language prefix for the French language to *France*:

1. Go to **Site Manager -> Development -> Cultures**.
2. Type *fr-fr* into the **Culture code** field of the filter and click **Show**.
3. Edit (🖉) the *French - France* culture.
4. Set the **Culture alias** property to *France*.
5. Click 💾 **Save**.

The system now displays the language prefix in URLs of French document as for example: *<domain>/France/Home.aspx*

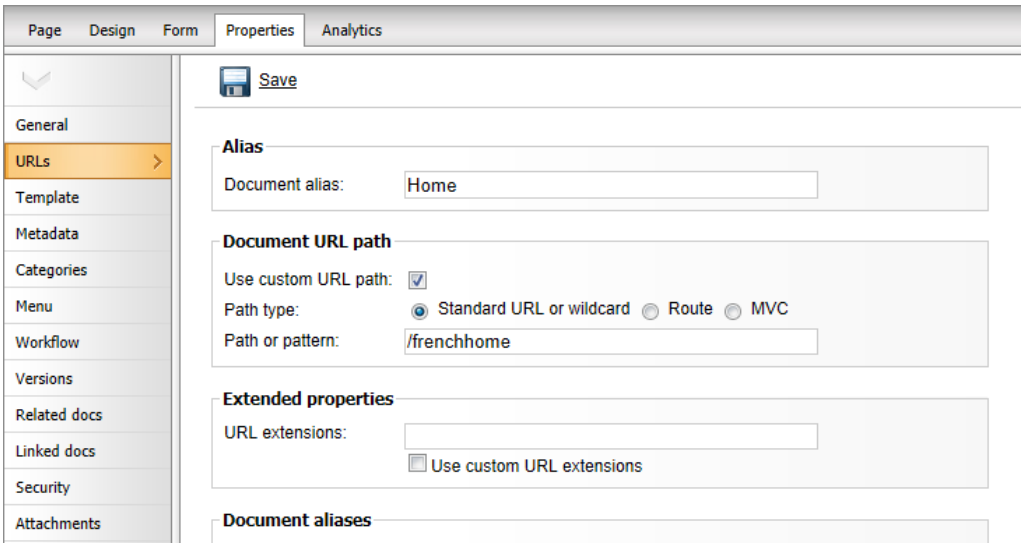
Using a custom document URL path for different culture versions

Forcing culture domains or using language prefixes for URLs ensures that you have a different URL for

every document culture version. You can use the following approach if the features described above are disabled or if you want different culture versions of documents to have different names in the URL path.

For example, if you want the French home page to have a different URL than the English version:

1. Go to **CMS Desk -> Content**, and select the Home page in the content tree.
2. Select the French language on the main toolbar.
3. Open the **Properties -> URLs** tab.
4. In the **Document URL path** section, check **Use custom URL path**.
5. Leave the **Standard URL or wildcard** option selected as the **Path type**.
6. Set the **Path or pattern** value to: */frenchhome*
7. Click **Save**.



To try out how the custom URL works, sign out and view the English version of the website. If you open the URL <*domain*>*/frenchhome.aspx*, the website automatically switches the culture to French and displays the French version of the home page.

By default, language selection web parts reflect custom URL paths for documents when generating links to the corresponding culture version. If you wish to disable this behavior for a specific web part and have it use links with the standard document URL, disable its **Use culture specific URLs** property.

You can also set Document aliases for a pages dedicated to specific cultures. When the document is accessed through this type of alias, the system always opens it in the corresponding culture.

Note: that you may encounter problems when using wildcards in URLs on multilingual sites. You can find more details and a possible solution in the Development -> Page processing and URLs -> Wildcard URLs topic.

4.9.6 Managing multilingual websites

Kentico CMS provides several additional features that help localize websites and manage multilingual content:

- Language status overview in List mode - allows you to check which language versions of documents exist on the website.

- Languages document tab - displays information about all language versions of the current document.
- Language version comparison - allows content editors to work with two language versions of a document side-by-side.
- Culture-dependent workflow scopes - each language version of the website can have its own workflow definition.
- Language-bound editors - you can limit which users are allowed to edit particular language versions of the website.

4.9.6.1 Language status overview

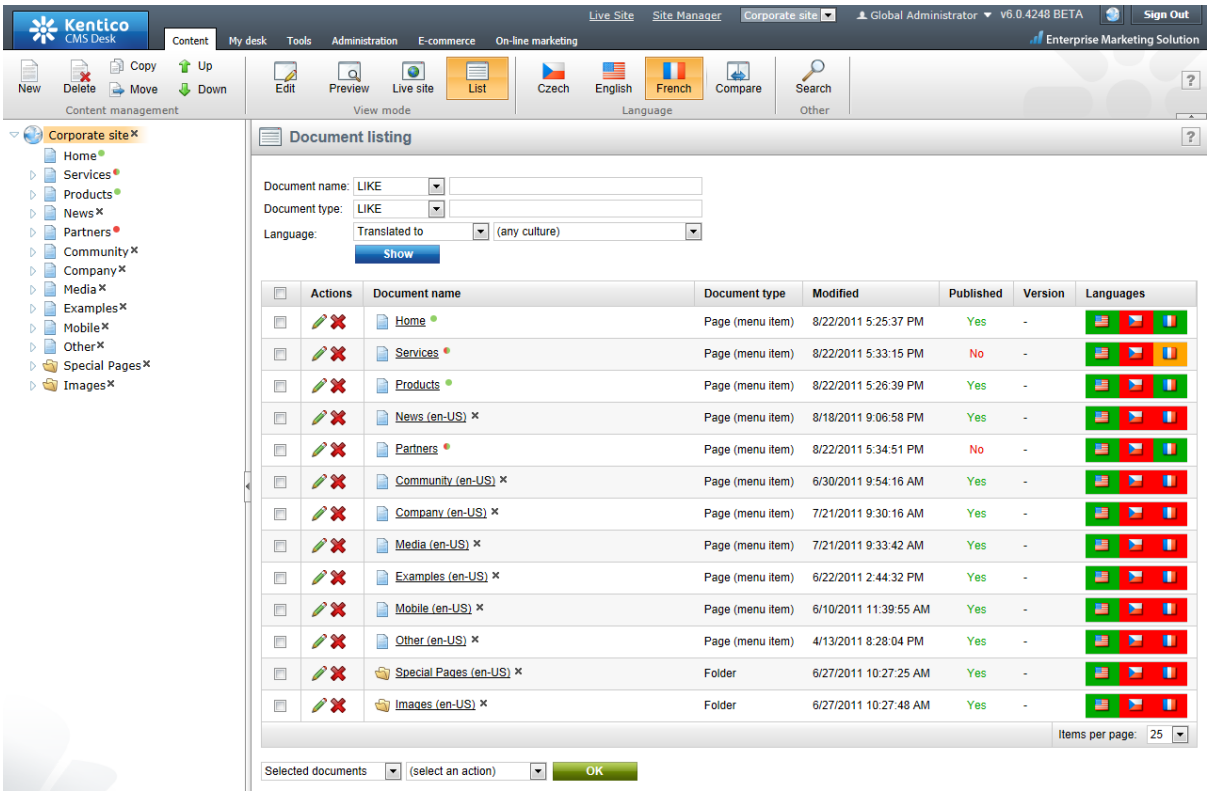
The **List** mode of CMS Desk provides an overview of document language versions. You can select a node in the content tree and switch to List mode, which shows all child documents placed under the given node.

The **Languages** column displays flags with differently colored backgrounds for all languages available on the website. The flags represent particular language versions of documents and indicate the following statuses:

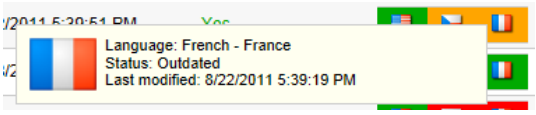
- **Green - Translated** - the document is available in the given language and up-to-date. The actual language of the document's content has no effect on the status, the system only checks whether the language version exists.
- **Orange - Outdated** - the language version exists for the document, but is outdated. The system considers language versions to be outdated if the default language version of the document has been modified (or published when using workflow) more recently.
- **Red - Not available** - the document does not exist in the given language. You can also see the  icon next to documents in this status, both in the listing and in the content tree.

Clicking a **Translated** or **Outdated** flag redirects you to the **Edit -> Page** tab of the appropriate language version of the document. If you click a **Not available** flag, the language versions's creation dialog opens.

The **Document name** column displays the names of the documents in the currently edited culture. If the document version **does not exist** in this language, the column displays the name from the **default culture** with the default culture code appended in brackets.



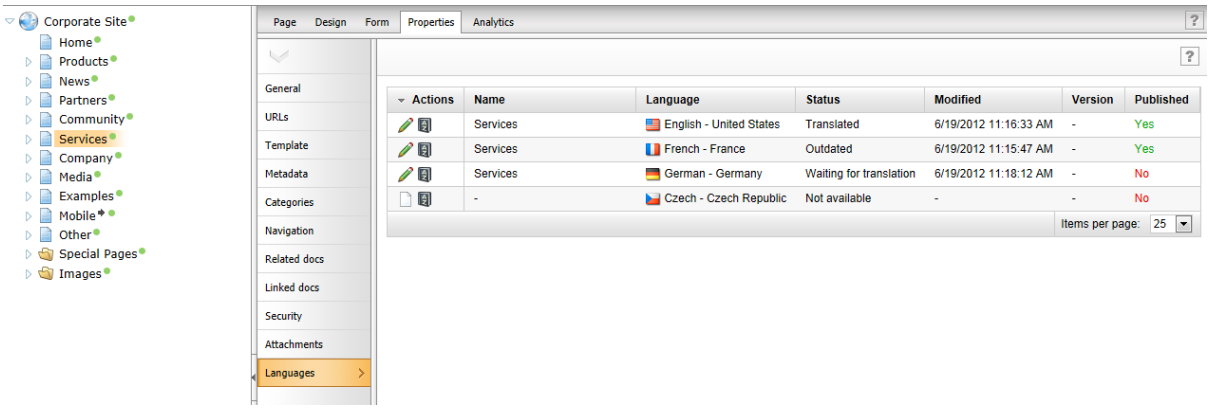
If you hover a flag with the mouse cursor, an info box is displayed, giving additional information about the language version.



The Languages tab

You may also view the language statuses separately for individual documents. To do so, select a document in the content tree and switch to the **Properties -> Languages** tab. Here you can also use the following actions for individual language versions:

- **Edit culture version** - allows you to edit the fields for the given language version of the document (i.e. opens the **Edit -> Form** tab in CMS Desk).
- **Create new culture version** - shown for languages in which the document is not available. Clicking the action creates a new version of the document in the given language.
- **Translate document** - opens a dialog where you can submit the document for translation to the given language, using a specified Translation service.



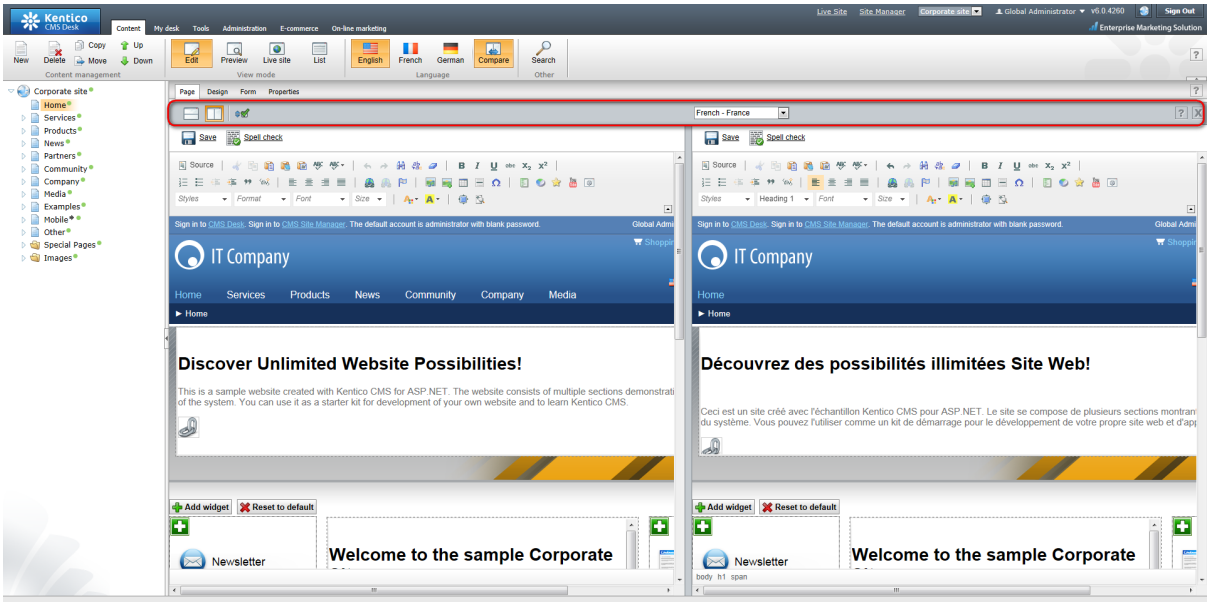
Hiding the Languages tab

The **Languages** tab can be displayed or hidden for members of particular roles. This can be set up using UI personalization.

4.9.6.2 Language version comparison

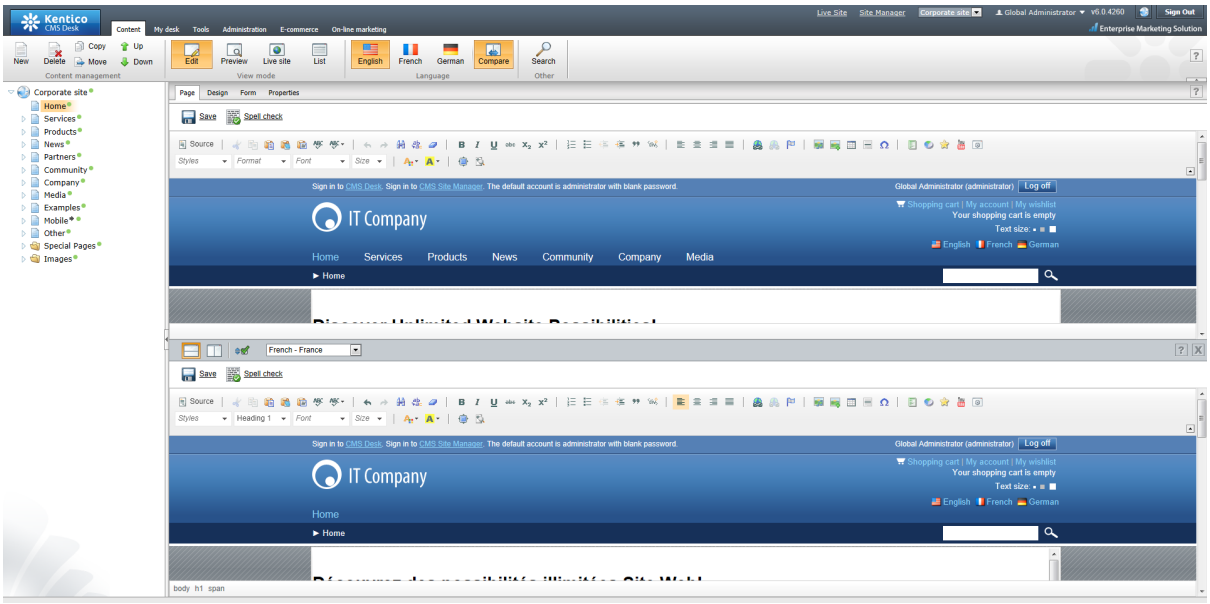
Language version comparison enables content editors to view and edit two different language versions of a document side-by-side. This is convenient when translating website content into multiple languages, as it eliminates the need to switch to the original language version to read source content and back to write its translation.

To switch to the language version comparison mode, select a document in the content tree and click the **Compare** ( ) button in the **Language** toolbar. It results in the language version comparison toolbar being displayed (as highlighted in the following screenshot), while the document editing area below the toolbar gets split into two parts. The left part displays the document in the language selected in the **Language** toolbar, while the right part displays the language selected using the drop-down list in the language version comparison toolbar. Both displayed language versions can be edited independently at the same time.

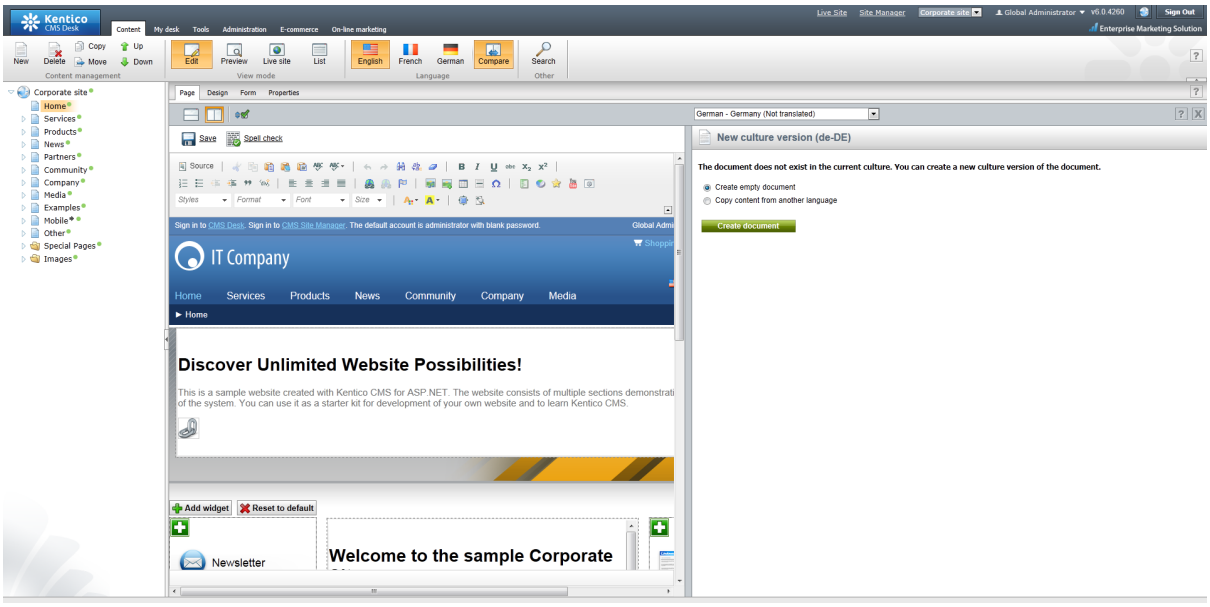


On the language version comparison toolbar, you can find the following controls:

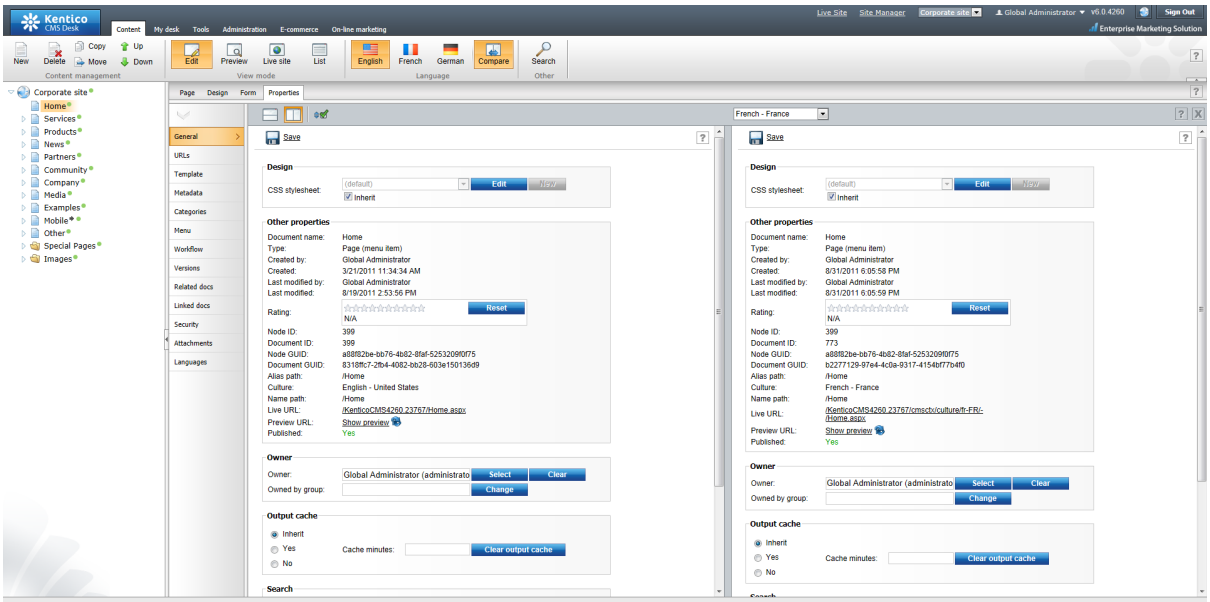
Horizontal layout	Splits the editing area horizontally (as in the screenshot below). The top part displays the document in the language selected in the **Language** toolbar, while the bottom part displays the language selected using the drop-down list in the language version comparison toolbar.
Vertical layout	Splits the editing area vertically (as in the screenshot above). The left part displays the document in the language selected in the **Language** toolbar, while the right part displays the language selected using the drop-down list in the language version comparison toolbar.
Synchronize scrollbars	If enabled, scrollbars in both parts of the editing area will be synchronized, resulting in the same sections of both language versions being displayed while scrolling.
Comparison language drop-down list	Using this drop-down list, you choose which language version of the currently selected document will be compared with the language version selected in the **Language** toolbar.
Close comparison mode	Closes the comparison mode and returns to the standard view of a single language version. The same can be achieved by clicking the **Compare** ( ) button in the **Language** toolbar again.



If you select a document whose version in the comparison language is not created yet, the standard **New culture version** dialog is displayed in the respective part of the editing area, letting you create a new empty document or copy content from another language.



Language version comparison mode is available in all three view modes: **Edit**, **Preview** and **Live site**. In **Edit** mode, you can use it on the **Page**, **Design**, **Form**, **Master page** and **Properties** tabs. On the **Properties** tab, it is available on all sub-tabs except for **Linked docs**, **Related docs**, **Security** and **Languages**, as document configuration performed on these tabs is shared by all language versions of the document.



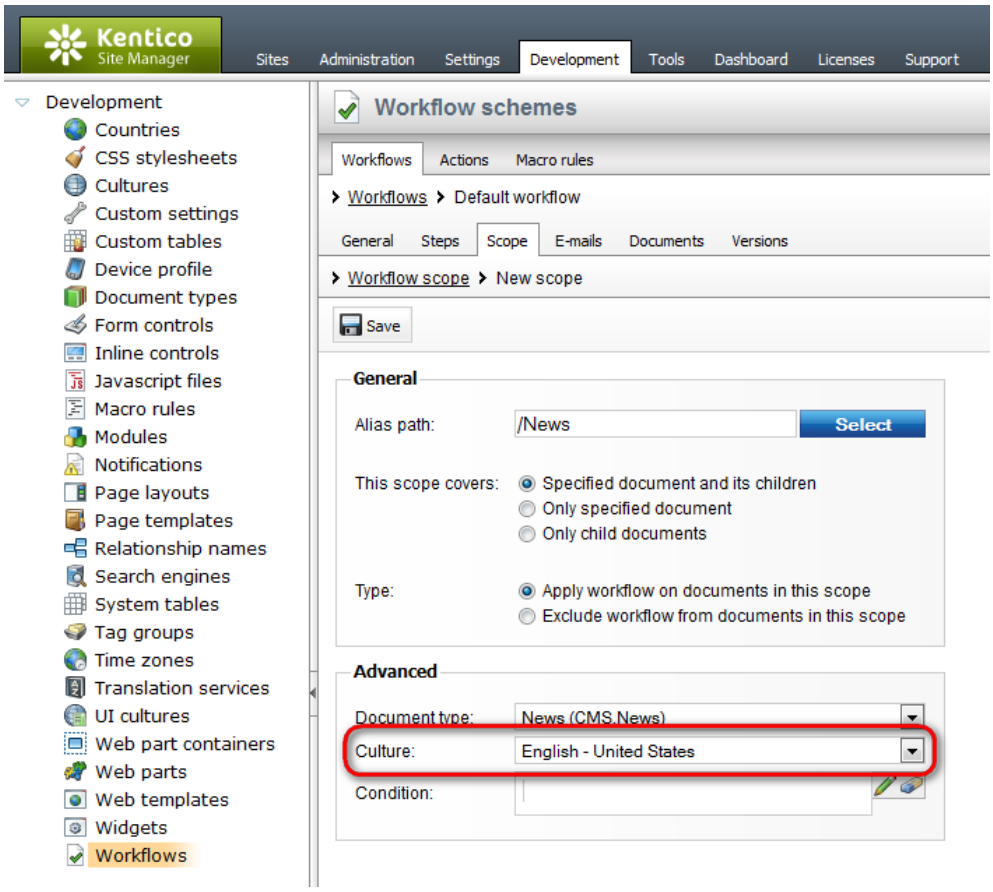
4.9.6.3 Culture-dependent workflow scopes

Workflows can be defined separately for each language version of your documents by using culture-dependent workflow scopes. A workflow scope determines which documents are affected by a particular workflow.

To learn how to create workflows and workflow scopes, refer to the Workflow and versioning chapter.

The system applies workflow scopes with the following priority (from highest to lowest):

1. Scopes with a specified document type and culture
2. Scopes with a specified document type
3. Scopes with a specified culture
4. Scopes without a specified document type and culture

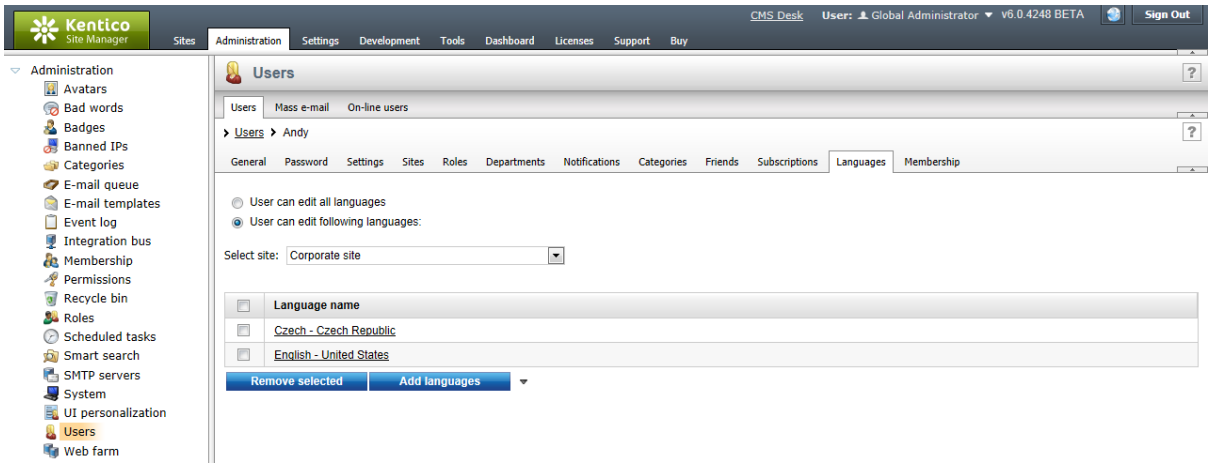


4.9.6.4 Language-bound editors

The system allows you to specify which language versions of documents users are allowed to edit.

1. Go to **CMS Desk / Site Manager -> Administration -> Users**.
2. **Edit** ( ) a particular user.
3. Switch to the **Languages** tab and select one of the following options:

- **User can edit all languages** - the current user can edit documents in any language.
- **User can edit following languages** - you can choose which language versions the user can edit. The *Select site* list allows you to configure the language bindings for specific websites.



4.9.7 Translation services

4.9.7.1 Overview

Translation services provide a way to automate or outsource the translation of website content. By utilizing translation services, you can create multilingual sites without having to edit the text in the Kentico CMS administration interface. The system automatically ensures the transfer of data between the website's documents and translation providers. Translation data is exported and imported via files using the standard XML-based XLIFF format.

There are two general categories of translation services:

- **Machine services** - use translation software to programatically convert source text from one language to another.
- **Standard services (human translation)** - provide an automated interface between the CMS and external human translators.

The system includes several default translation services of both types and also supports integration of custom services.

Topics:

- Translating documents
- Configuring content for translation

- Machine translation services
  - Translating localization strings
  - Microsoft Translator
  - Google Translate

- Standard translation services
  - Managing translation submissions
  - Importing completed translations
  - Translations.com
  - Manual translation
  - E-mail translation

- Adding custom translation services
- Translating via the REST service

- Security

4.9.7.2 Translating documents

The main purpose of translation services is to create language versions of the website's documents based on the content in another language. There are several different ways to submit documents for translation.

Requirements:

- The **Enable translation services** setting must be enabled in **Site Manager -> Settings -> Content -> Translation services**.
- Your website needs to be configured for multiple content cultures, including all languages that you plan to use.

Translating documents to other languages

To create new language versions of documents using a translation service:

1. Select the document that you wish to translate in the content tree.

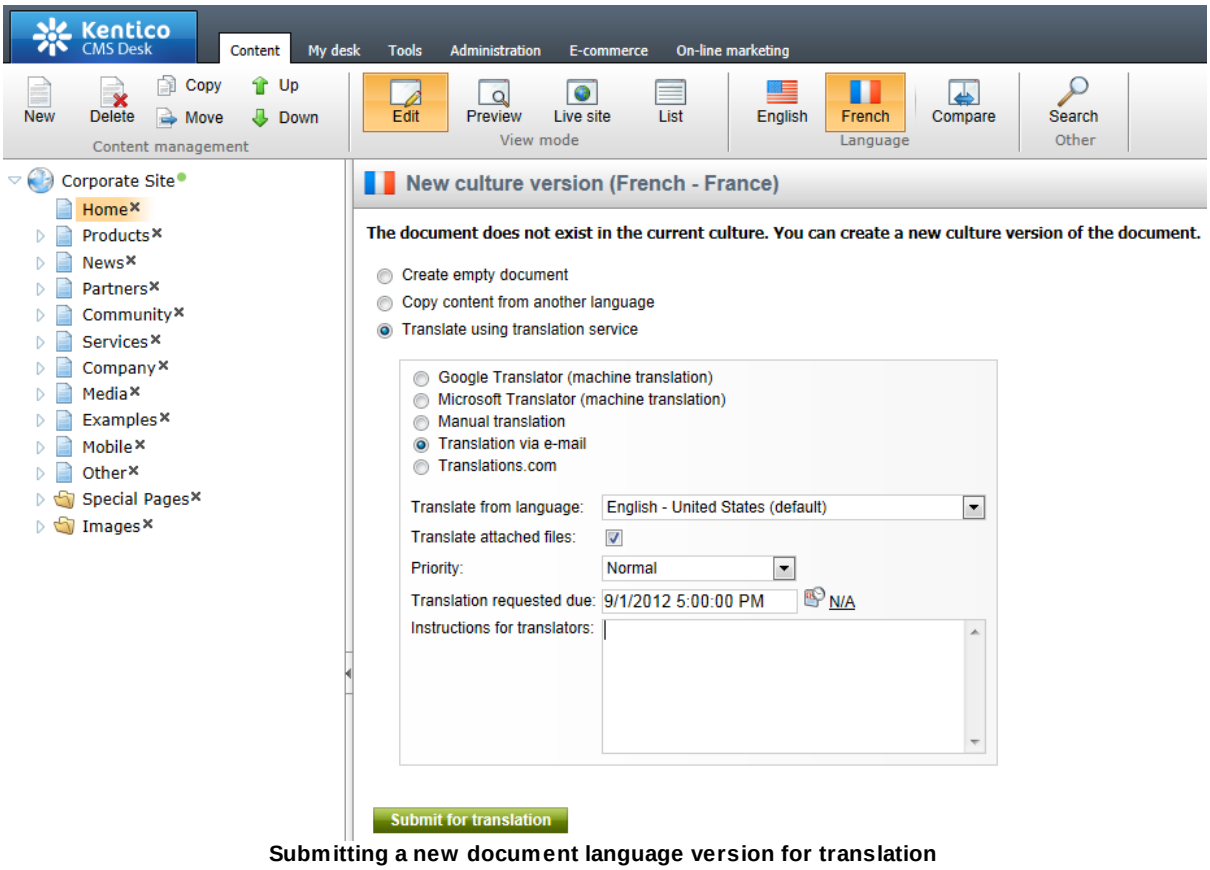
2. Switch to the target language using the selector on the main CMS Desk toolbar.

3. Select **Translate using translation service** in the *New culture version* dialog.

4. Choose one of the available translation services.

5. Fill in the following details for the translation (some of the options may not be supported by the selected service):

Translate from language	If the document already exists in more than one language, you can choose which one the service uses as the source for the translation.
Human translation parameters	
Translate attached files	Enable this option to submit the document's file attachments for translation along with the main content. The system adds file attachments directly into the XLIFF translation source as binary data encoded in Base64 format.
Priority	Sets the priority of the translation (low, normal or high).
Translation requested due	Specifies a deadline date for the translation.
Instructions for translators	Here you can enter additional instructions for human translators.



Submitting a new document language version for translation

6. Click **Submit for translation**.

The system adds the new language version of the document.

- Machine services immediately insert the translated content into the new document version.
- When using standard (human) translation services, the system initially creates the new document version as a copy of the source language. The content tree shows the *Waiting for translation* status icon (●) next to documents in this state. You can manage the translation through a submission ticket in **CMS Desk -> Tools -> Translations**. Once the service completes the translation, you can process the submission to transfer the new content into the appropriate language version of the document.

Tip

You can integrate document translation via services into your workflow process. Advanced workflow offers action steps that allow you to automatically manage document translations.

There are two predefined workflows (in **Site Manager -> Development -> Workflows**) that you can configure and apply to your website:

- **Translation - Default language version** - automatically sends documents for translation via a selected service when a user approves the default language version.
- **Translation - Import to other language versions** - periodically checks for translations completed by standard services and imports them into the appropriate documents when ready.

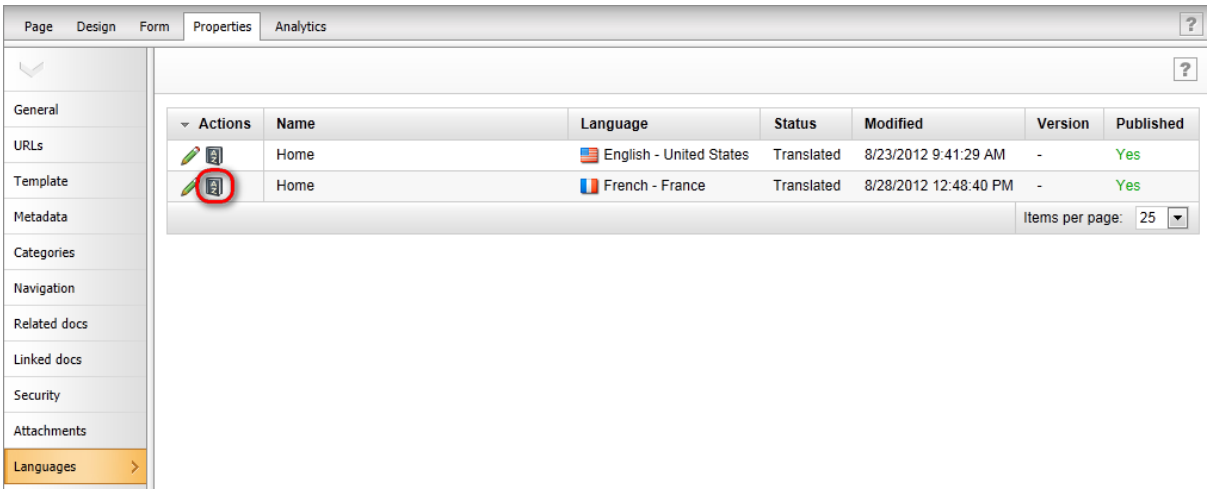
Re-translating existing document versions

You can submit documents for translation even if they already exist in the target language. This allows you to use translation services to update the language versions of a document after making changes to its main content.

1. Select the document in the content tree.

2. Open the **Properties -> Languages** tab.

3. Click the **Translate document** ( ) action for the language version that you wish to re-translate.



A translation dialog opens.

4. Choose a service and specify the parameters of the translation.

5. Click **Translate**.

The system submits the document for translation just like when creating a new language version. The text returned by the translation service overwrites the original content of the document in the selected language.

Note

You may also use the **Properties -> Languages** tab to create new language versions of documents (even if the given document does not exist in the target language).

Translating multiple documents

To translate more than one document using a single action:

1. Open the **List** mode of CMS Desk.

2. Select the *parent* of the documents that you wish to translate in the content tree.

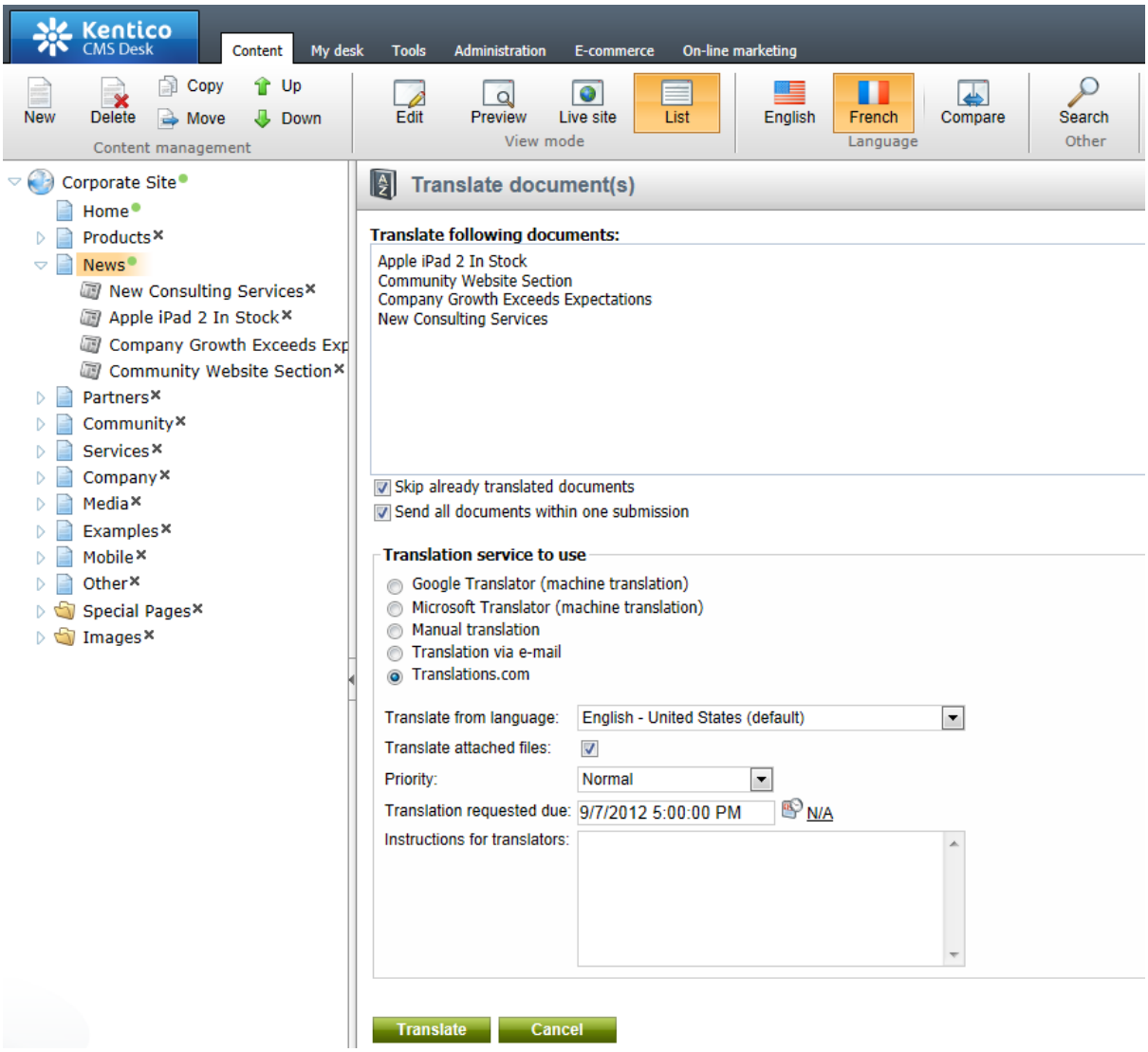
3. Select the target language of the translation on the main CMS Desk toolbar.

4. Mark documents for translation using the checkboxes next to the listed items.

5. Select the **Translate** action below the document list and click **OK**.

6. Choose a service and specify the parameters of the translation. When submitting multiple documents for translation, you can also enable or disable the following options:

Skip already translated documents	If checked, documents that already exist in the target language are not submitted for translation. Disable this option to overwrite the current content of the documents with the text returned by the translation service.
Send all documents within one submission	If enabled, the system includes all listed documents in a single translation submission. Otherwise, separate submissions are created for individual documents. Only applicable when using a human translation service.



Translating all documents under the News sections of a website

7. Click **Translate**.

The translation process works the same way as for single items, but the system performs it for all selected documents.

4.9.7.3 Configuring content for translation

Translation services can process the following types of document content:

- Content of editable regions
- Values of document fields (entered on the **Form** tab)
- The properties of web parts placed on the given document
- Document file attachments

The system allows you to specify exactly which parts of the content are translated. These settings affect all types of translation services.

Configuring document fields

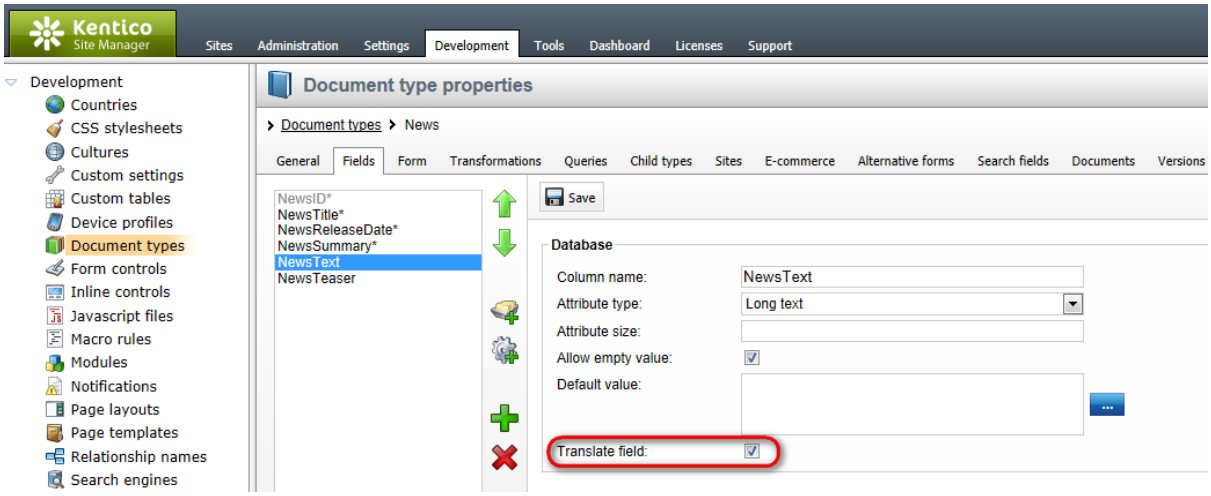
To choose which document fields are sent for translation, adjust the form definition of individual document types:

1. Go to **Site Manager -> Development -> Document types**.

2. Edit () a specific document type and open the **Fields** tab.

3. Select the field that you wish to configure and enable or disable the **Translate field** flag. Translation is only available for fields with the *Text*, *Long text* or *File* **Attribute type**.

4. Click  **Save** to confirm the changes to the field.



Enabling the Translate field flag in the field editor

When users submit documents of the given type for translation, only the content of fields that have the **Translate field** flag enabled is included in the translation.

Configuring web part content

Pages may also contain text added via Web parts and Widgets. The only way to translate web part content is to localize the values of the corresponding web part properties. You can configure the system to automatically include specific web part properties in translations. This is only recommended for web parts that directly add text onto the page (such as Static text).

1. Go to **Site Manager -> Settings -> Content -> Translation services** and enable the **Translate web part properties** setting.

2. Open **Site Manager -> Development -> UI cultures**, select the **All cultures** tab, and add UI cultures for all content languages that you use on your website.

3. Navigate to **Site Manager -> Development -> Web parts**.

4. Select the appropriate web part and open the **Properties** tab.

5. Enable or disable the **Translate field** flag for individual properties. Translation is only available for properties that use the *Text*, *Long text* or *File* **Attribute type**.

6. Click **Save** for every modified property.

If a document that contains the given web part is submitted for translation, the system includes the values of all properties that have the **Translate field** flag enabled. When importing the translation, the system creates a resource string for every translated property, containing the value in the target language. The original text values of the properties are replaced by localization macros, which insert the appropriate resource strings.

Editable text regions

Translation submissions always include the page content entered by users into Editable text web parts, regardless of the web part translation settings.

Allowing translation of document attachments

Human translation services allow users to submit file attachments for translation along with the main content of documents. To configure attachment translation for your website:

1. Go to **Site Manager -> Settings -> Content -> Translation services**.

2. Enable the **Allow attachment translation** setting.

3. Specify which types of files should be translated using the **Translation attachments file types** setting. Enter a list of all allowed file extensions separated by semicolons (for example: *txt;docx;pdf*). Leave the setting empty to allow translation for all types of attached files.

4. Click **Save**.

Users can check the **Translate attached files** option when submitting documents for translation, and the system adds all allowed file attachments into the translation source. When the completed translation is imported, the translated file is attached to the appropriate language version of the document.

File format

The system adds file attachments directly into the XLIFF translation source as binary data encoded in Base64 format.

4.9.7.4 Customizing XLIFF export

Note that by default, the exported XLIFF files use CDATA notation. You can turn CDATA off in exported XLIFF files by adding the *CMSTranslationServicesUseCDATAForTransUnit* key into your web.config file.

Example of an XLIFF file with the CDATA notation.

```
<xliff version="1.2">
  <file original="cms.document;25" source-language="en-US" target-language="fr-FR"
 datatype="htmlbody">
    <body>
    ...

      <trans-unit id="newstitle">
        <source><![CDATA[Apple iPad Mini In Stock]]></source>
      </trans-unit>
      <trans-unit id="newssummary">
        <source><![CDATA[Today, we have good news for all fans of the awesome
Apple iPad. We are glad to announce that its new version, Apple iPad Mini, is
available in our web shop. Furthermore, we keep our reasonable pricing policy,
providing the lowest price currently available on the Web.]]></source>
      </trans-unit>

    ...
    </body>
  </file>
</xliff>
```

Disabling CDATA in exported XLIFF files

1. Open the web.config file located in the root of the website.

2. Add the *CMSTranslationServicesUseCDATAForTransUnit* key into the <app settings> section of the file.

```
<appSettings>
  ...

  <add key="CMSTranslationServicesUseCDATAForTransUnit" value="false" />

  ...
</appSettings>
```

3. Save the web.config file.

Now, whenever you submit a document for translation, the system exports it without the CDATA notation.

Example of an XLIFF file with the CDATA notation turned off.

```
<xliff version="1.2">
  <file original="cms.document;25" source-language="en-US" target-language="fr-FR"
 datatype="htmlbody">
    <body>
```

```
    ...

      <trans-unit id="newstitle">
        <source>Apple iPad Mini In Stock</source>
      </trans-unit>
      <trans-unit id="newssummary">
        <source>Today, we have good news for all fans of the awesome Apple iPad.
We are glad to announce that its new version, Apple iPad Mini, is available in our
web shop. Furthermore, we keep our reasonable pricing policy, providing the lowest
price currently available on the Web.</source>
      </trans-unit>

    ...
    </body>
  </file>
</xliff>
```

4.9.7.5 Machine translation services

Machine services produce translations by submitting the source text to a dedicated translation application, typically a web service. There is no human involvement in the translation procedure, which allows the services to return the translated text immediately (after a short processing period). In most cases, the output text is not directly suitable for publishing without additional editing, but you can use machine translation to quickly create a foundation of the website's content in any language.

In addition to translating documents, machine services also allow users to automatically translate localization strings.

Default machine services:

- Microsoft Translator
- Google Translate

4.9.7.5.1 Translating localization strings

You can use machine services to assist in the translation of localization strings. Multilingual resource strings allow you to dynamically display text in the language selected by the current user, both on the live website and inside the Kentico CMS administration interface.

Requirements:

- The system needs to be configured for multiple UI cultures, including all languages that you wish to use in your translations.
- At least one machine translation service must be enabled and configured for the website.

The following example demonstrates how to insert and translate a string into the display name of a category:

1. Open CMS Desk and navigate to **Administration -> Categories**.

2. Select a category in the tree menu and switch to the **General** tab.

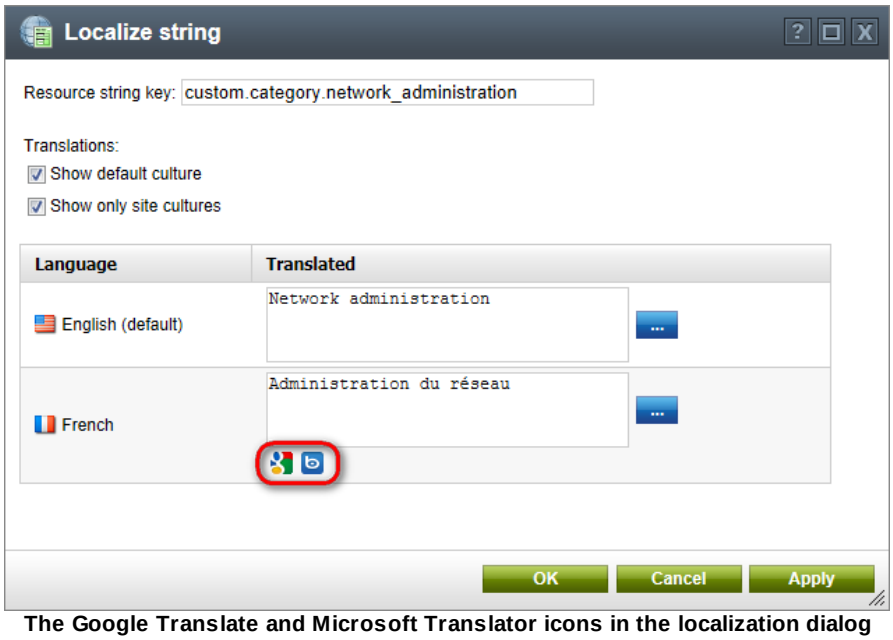
3. Click the **Localize** icon ( ) next to the **Display name** field.

The **Localize field** dialog opens.

4. Leave the default **Create new resource string** option and type in a name for the resource string. Click **OK**.

The **Localize string** dialog shows the content of the string in the default language, and you can fill in a translation for all other available UI cultures.

5. Click on the icon representing the preferred machine translation service under the editing area of a particular language.



The Google Translate and Microsoft Translator icons in the localization dialog

The service automatically translates the string's text from the default language.

6. Click **OK** to save the new content of the string.

7. Click **Save** on the category's **General** tab to insert the localization string into the **Display name** field.

Users now see the name of the category translated according to their currently selected language (or UI culture in the administration interface).

You can use the same approach when translating any type of localization string. Machine translation services are also available when editing resource strings directly in the **Site Manager -> Development -> UI cultures** interface.

4.9.7.5.2 Default machine services

4.9.7.5.2.1 Microsoft Translator

This service uses the Microsoft (Bing) Translator API to translate text. Microsoft Translator offers a free subscription for up to 2 million characters per month, higher editions must be purchased.

Character limit

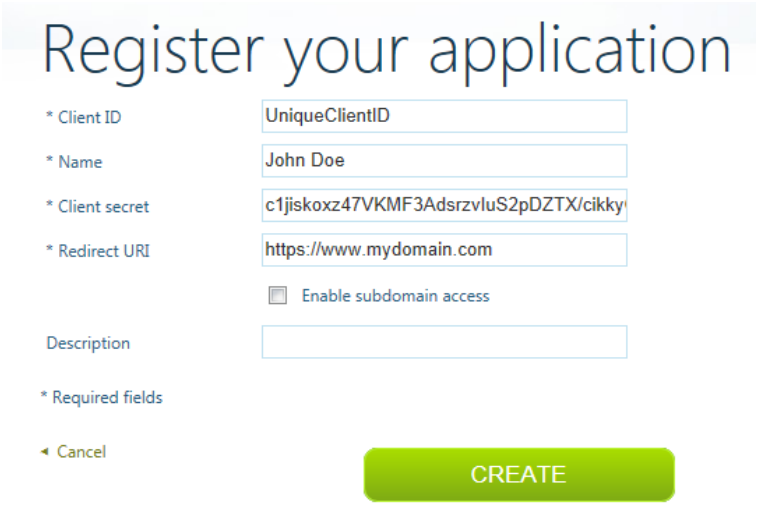
This service has a 2000 character limit for every translation request. When translating documents, the service submits the content of every field and editable region as a separate unit, so the total number of characters per document is not restricted.

Configuring the Microsoft Translator service

To set up the Microsoft Translator service on your website:

1. Sign up for a subscription to the Microsoft Translator on the Windows Azure Marketplace. You need to create a Windows Live ID account if you do not already have one.

2. Register your website as an application on the Azure DataMarket.



3. Open Kentico CMS and go to **Site Manager -> Settings -> Content -> Translation services**.

4. Enter the **Client ID** and **Client secret** of your application into the corresponding settings in the **Microsoft Translator** section. Click **Save**.

You can now translate documents and localization strings using the Microsoft Translator service.

4.9.7.5.2.2 Google Translate

This service uses the Google Translate API to translate text. Google Translate is a paid service.

Character limit

This service has a 2000 character limit for every translation request. When translating documents, the service submits the content of every field and editable region as a separate unit, so the total number of characters per document is not restricted.

Configuring the Google Translate service

To set up the Google Translate service on your website:

1. Sign in to the Google APIs console using a Google account.
2. Create a new project and activate the **Translate API** service.
3. Obtain your project's API key on **API Access** pane.
4. Open Kentico CMS and go to **Site Manager -> Settings -> Content -> Translation services**.
5. Copy the **API Key** into the corresponding setting in the **Google Translator** section. Click **Save**.

You can now translate documents and localization strings using the Google Translate service.

4.9.7.6 Standard translation services

Standard services automatically deliver the translation source data to designated human translators. Human translation takes time, so the results are not available immediately, but it should provide significantly higher translation quality than machine services. The services export and import data using the XLIFF format, which is generally supported by translation tools.

Document translation via standard services uses the following life cycle:

1. Users submit documents for translation in CMS Desk.
2. If the given documents do not exist in the target language, the system creates the new culture versions as direct copies of the source language. The content tree marks the documents with the *Waiting for translation* status icon ( ).

Important!

When translating documents on live websites, apply Workflow to avoid publishing content before the translations are complete.

3. The system creates a submission ticket for every translation. You can manage the submissions in **CMS Desk -> Tools -> Translations**.

4. Once the service provider finishes the translation, you can import the data and process the corresponding submission. This inserts the translated content into the appropriate language version of the submitted documents.

Default human translation services:

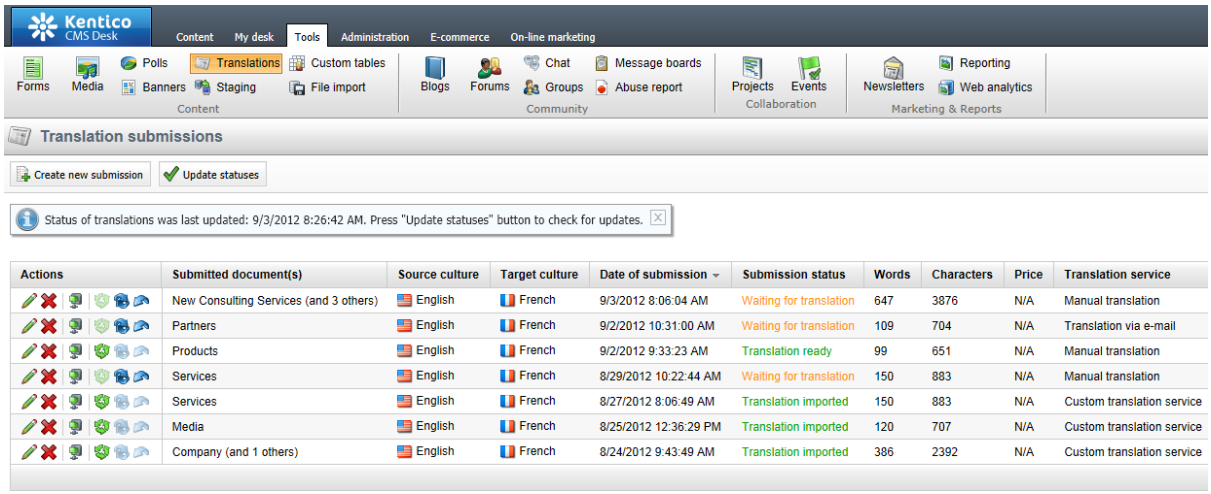
- Translations.com
- Manual translation
- E-mail translation

4.9.7.6.1 Managing translation submissions

Standard services use submission tickets to keep track of translation tasks. The system automatically creates a new submission whenever a user assigns a document (or batch of documents) for translation via a human service. To access a list of all translation submissions created on a specific website, go to **CMS Desk -> Tools -> Translations**.

You can manage the listed submissions through the following actions:

- **Edit** - allows you to modify the details of the submission. For example, you can update the submission's deadline date and then resubmit it to the service.
- **Delete** - removes the translation submission. When deleting submissions that have the *Waiting for translation* status, the system also performs the *Cancel submission* action.
- **Export all to zip** - allows you to download a zip package containing the translation source data for all documents in the submission. The data is stored in *.xlf* files using the XLIFF format.
- **Process translations** - inserts the translated content into the corresponding language versions of the documents included in the submission. Only available for submissions that have a completed translation (i.e. those in the *Translation ready* status). See more in Importing completed translations.
- **Resubmit to translation service** - resends the translation assignment to the service according to the current settings of the submission and the latest content of the source documents.
- **Cancel submission** - switches the submission to the canceled status. This also informs the translators that they no longer need to finish the translation. The exact functionality performed when canceling a submission depends on the implementation of the used service.



4.9.7.6.2 Importing completed translations

Because of the waiting period involved in human translation, the system provides an import mechanism that checks the status of translations and retrieves the required data when it is ready. Most human translation services support automatic loading of completed translations.

The translation retrieval process consists of two steps:

1. Importing the translation data - loads the XLIFF data of the finished translation from the service and transfers it into the matching submission. This does not affect the website's content.

2. Processing the submission - inserts the translated content into the actual documents. This overwrites the appropriate language version of the documents that were submitted for translation.

Note

You can combine both steps of the translation retrieval process by enabling the **Automatically import translated submissions** setting in **Site Manager -> Settings -> Content -> Translation services**. In this case, the system automatically inserts translated content into documents when the data is imported into submissions.

Warning: This overwrites documents immediately when a service finishes the translations, without the approval of the content administrators.

To import the content of translated submissions:

1. Go to **CMS Desk -> Tools -> Translations**. All untranslated submissions are in the *Waiting for translation* status.

2. Click **Update statuses** above the list of submissions. This imports all completed translations and switches the corresponding submissions to the *Translation ready* status.

You can automate the translation status updates by enabling and configuring the **Translation retrieval** scheduled task for your site. With this task running, the system regularly updates all translation submissions according to a predefined time interval.

3. Click the **Process translations** action for submissions that are in the *Translation ready* status.

The system transfers the translated content into the corresponding document(s) and changes the submission status to *Translation imported*.

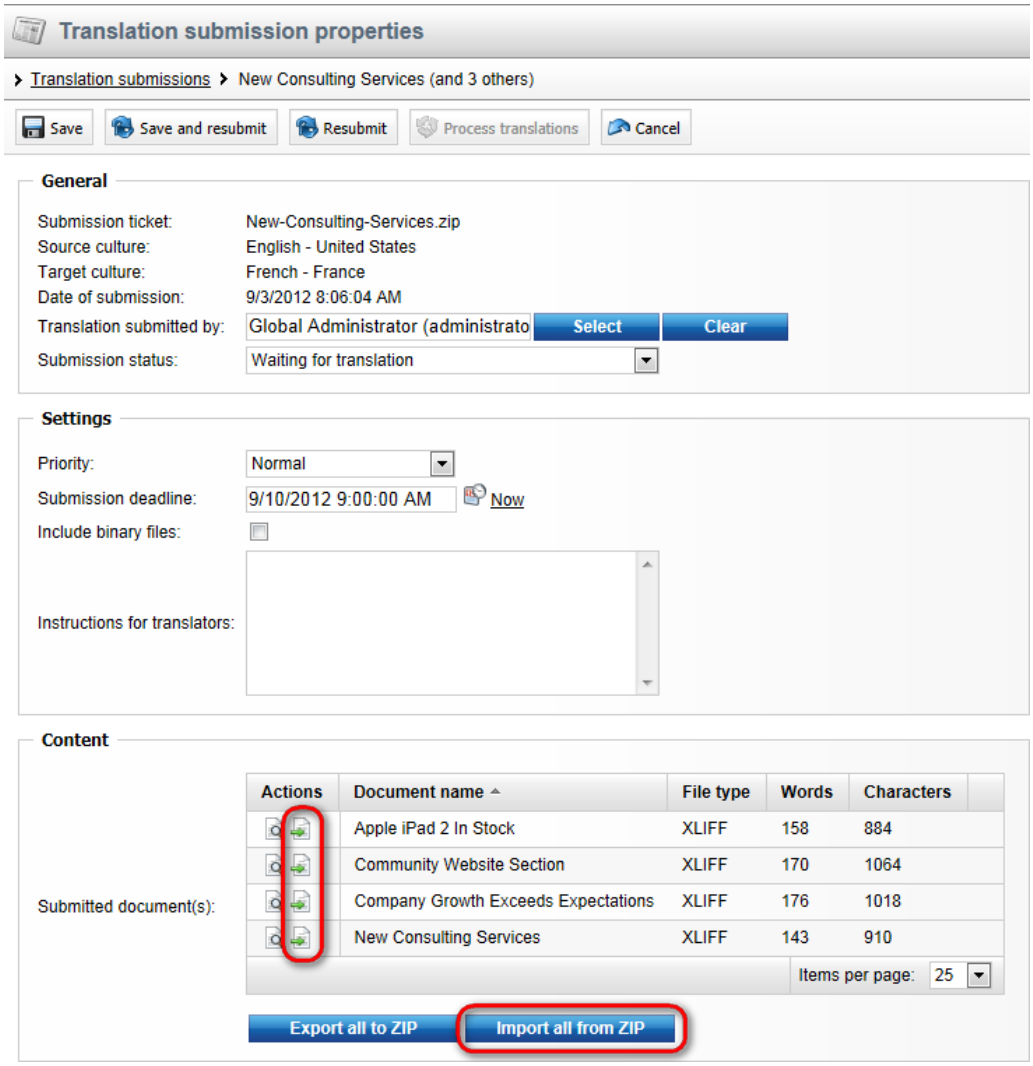
Manually uploading translation data

The system allows you to directly upload the XLIFF translation data into submissions. You can use this approach if you encounter any problems with the translation retrieval, or if your service does not support automatic loading of finished translations.

1. Edit () the given translation submission in **CMS Desk -> Tools -> Translations**.

2. Load the translated data into the submission. You have two options:

- To upload an *.xlf* file containing the translated data of a specific document, use the **Upload translated XLIFF file** action next to the matching item in the list.
- If you have the completed translation for the entire submission in a zip archive, upload it by clicking the **Import all from ZIP** button.



If the upload is successful, the submission switches to the *Translation ready* status. You can then process the submission to transfer the translated content into the website's documents.

4.9.7.6.3 Default standard services

4.9.7.6.3.1 Translations.com

Translations.com is a worldwide translation provider that offers website localization in over 170 languages. Kentico CMS includes a default service that automatically ensures the transfer of translation data between the system and Translations.com.

To set up the Translation.com service:

1. Contact a Translations.com representative and arrange a translation project for your website.

2. Open Kentico CMS and go to **Site Manager -> Settings -> Content -> Translation services**.

3. Fill in the information of your Translation.com project into the settings under the **Translations.com** section:

- Client user name
- Client user password
- Project short code

Click **Save**.

4. Add the following endpoints into the <*system.serviceModel*><*client*> section of your application's web.config file:

```
<endpoint address="" binding="basicHttpBinding"
contract="TranslationsComSessionService.SessionService2PortType" />

<endpoint address="" binding="basicHttpBinding"
contract="TranslationsComProjectService.ProjectService2PortType" />

<endpoint address="" binding="basicHttpBinding"
contract="TranslationsComDocumentService.DocumentService2PortType" />

<endpoint address="" binding="basicHttpBinding"
contract="TranslationsComSubmissionService.SubmissionService2PortType" />

<endpoint address="" binding="basicHttpBinding"
contract="TranslationsComTargetService.TargetService2PortType" />

<endpoint address="" binding="basicHttpBinding"
contract="TranslationsComUserProfileService.UserProfileService2PortType" />
```

Valid endpoint addresses for your application will be provided by Translations.com.

You can now submit documents for translation using the Translations.com service.

Translation submission logs

Whenever a user submits a translation to Translations.com, the system creates a record in the event log (**Administration -> Event log**). The service also logs any errors that may occur while sending or receiving translation data.

4.9.7.6.3.2 Manual translation

The manual service exports the translation data onto the file system. Every translation submission is converted to a zip archive containing *.xlf* files (the source data in XLIFF format) and instructions. This allows you to deliver the translation assignment by sharing the export folder or by manually sending the submission package to the translators.

The service checks for finished translations in a designated import folder. The translators can also manually upload the completed zip package through a specific system page (the default instructions provide a link).

Important!

The package containing the completed translation must have the same names as the original source package for all .xlf files and the parent zip archive. The system uses the names as identifiers when importing the translated data.

Configuring the manual translation service

To set up the manual translation service, go to **Site Manager -> Settings -> Content -> Translation services** and configure the settings in the **Manual translation** section:

Translation submission export folder	Specifies the path of the folder where the service creates the zip packages with the translation source data (XLIFF files and instructions). If empty, the *~/App_Data/Translations/Export/* folder is used.
Translations import folder	Sets the path of the folder from which the system imports zip packages containing completed translations. If empty, the *~/App_Data/Translations/Import/* folder is used.
Delete ZIP after successful download	If enabled, the system deletes the zip packages containing finished translations after importing the XLIFF data.

Customizing the instruction files

In addition to the .xlf files that store the translation source data, the manual translation service places instruction files into the exported zip packages. The instruction files are created as copies of template files found in the ***~/App_Data/CMSModules/Translations/Template/*** folder under your web project.

By default, the folder contains the **Instructions.html** file. You can modify the content of this file according to your own requirements and even add additional instruction files that the service will include in the zip packages.

You can use context macro expressions in the text of instruction files to dynamically insert the values of the given submission. For example:

- **{% Submission.SubmissionSourceCulture %}** - returns the culture code of the translation source language.

- **{% Submission.SubmissionTargetCulture %}** - returns the culture code of the target language.
- **{% GetTranslationPriority(Submission.SubmissionPriority) %}** - returns the priority set for the submission (low, normal or high).
- **{% Submission.SubmissionDeadline %}** - returns the date and time of the submission's deadline.
- **{% Submission.SubmissionInstructions %}** - returns the additional instructions entered for the submission.
- **{% Submission.SubmissionWordCount %}** - returns the total number of words in the translation source text.

- **{%SubmissionLink%}** - returns the URL of a system page where users can upload finished translation packages.

4.9.7.6.3.3 E-mail translation

This service delivers the translation data via e-mail. When a user submits a document for translation, the service creates a zip package containing the XLIFF source file(s) and sends it to the specified translators as an attachment of an e-mail message. The e-mail service allows you to exchange data with translators or agencies that do not have their own API for managing submissions.

The system cannot cancel e-mail translation submissions once they have been sent. In these cases, you need to contact the translators directly.

The e-mail translation service does not support automatic retrieval of completed translations. The translators can submit the files by:

- Uploading the zip archive containing the finished translation via a link included in the original assignment e-mail
- Sending the translation (in XLIFF format) to a member of the website's staff, who can then import it manually through the **CMS Desk -> Tools -> Translations** interface

Once the translation is imported into the corresponding submission, you can process it to transfer the new content into the documents.

Configuring the e-mail translation service

To set up the e-mail translation service:

1. Ensure that your application is configured to send e-mails through an appropriate SMTP server.

2. Go to **Site Manager -> Settings -> Content -> Translation services** and fill in the following settings in the **Translation via e-mail** section:

Translation submission recipients	Sets the addresses to which the system sends translation assignments through the e-mail service. You can enter multiple e-mail addresses separated by semicolons.
E-mail sender	When sending translation assignment e-mails, the system automatically sets the sender to the address of the user who submitted the translation. The value of this setting is used if no user address is available, for example in the case of translations submitted automatically or via the API.

Click **Save**.

The service sends the translation e-mails using the specified addresses.

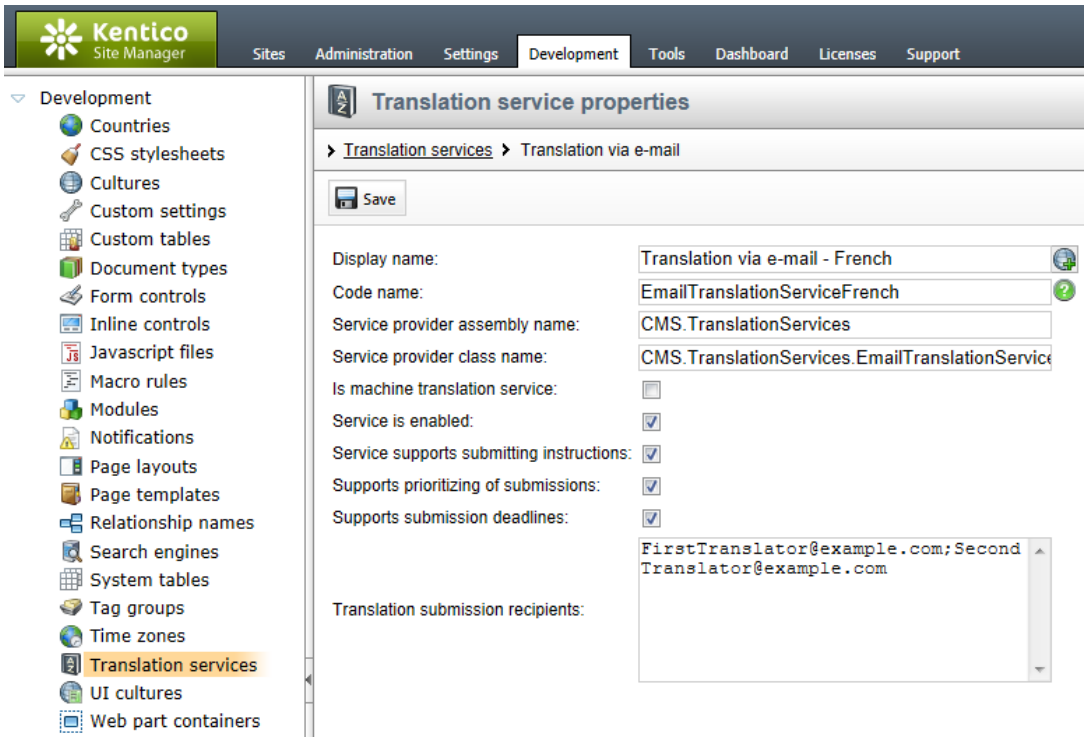
Setting language-specific recipients

If you need to assign different translator addresses for individual languages:

1. Go to **Site Manager -> Development -> Translation services**.

2. Create a **Clone** () of the *Translation via e-mail* service for each additional language.

3. Edit the services and enter the appropriate translator addresses into the **Translation submission recipients** property. This property overrides the **E-mail sender** setting.



Cloned e-mail translation service for the French language

When submitting documents for translation, users need to select the clone of the e-mail translation service that matches the target language.

Customizing the translation e-mails

To change the content of the e-mail messages that the service sends to translators, you need to modify the appropriate e-mail template:

1. Go to **Site Manager -> Administration -> E-mail templates**.

2. Edit the **Translation services - Submit translation** template.

3. Alter the template according to your specific requirements and click **Save**.

You can use context macro expressions to dynamically insert the values of the given translation submission. For example:

- **{% Submission.SubmissionSourceCulture %}** - returns the culture code of the translation source language.
- **{% Submission.SubmissionTargetCulture %}** - returns the culture code of the target language.
- **{% GetTranslationPriority(Submission.SubmissionPriority) %}** - returns the priority set for the submission (low, normal or high).
- **{% Submission.SubmissionDeadline %}** - returns the date and time of the submission's deadline.
- **{% Submission.SubmissionInstructions %}** - returns the additional instructions entered for the submission.
- **{% Submission.SubmissionWordCount %}** - returns the total number of words in the translation source text.

- **{%SubmissionLink%}** - returns the URL of a system page where users can upload finished translation packages.

4. (Optional) Follow the steps described for the manual translation service if you wish to customize the instruction files included in the zip package attached to the e-mails.

The service sends all translation assignment e-mails based on the current e-mail and instruction file templates.

4.9.7.7 Adding custom translation services

In addition to the translation providers available by default, you can also integrate any other services or your own translation mechanisms.

To add a custom service:

1. Create a class that defines the translation functionality or the communication logic between Kentico CMS and an external service. The implementation rules for the class are different for machine and human translation services. You can find detailed information and custom development examples for the two types of services in the dedicated topics:

- Creating machine services
- Creating human translation services

2. Connect the class to the CMS application. There are two options:

- Create a new assembly (Class library) in your web project and include the service class there. In this case, you must add the appropriate references to both the assembly and the main CMS project.
- Define the service in **App_Code** and load the class via the API. This ensures that the system automatically compiles the service class and you do not need to use a separate assembly.

3. Register the service in the system via the **Site Manager -> Development -> Translation services** interface and specify its properties.

4. Create Custom settings for any additional parameters required by the service, such as connection credentials or API keys. This allows administrators to configure the service for specific websites or translation projects.

When translating documents or resource strings, users can choose custom services among the translation options offered by the system.

4.9.7.7.1 Creating machine services

To develop a custom machine translation service, you need to define a class that inherits from **AbstractMachineTranslationService** (found in the **CMS.TranslationServices** namespace).

The service class must override and implement the following abstract methods:

Method	Description
Translate	The main translation method which calls the appropriate web service (Google, Bing etc.) or otherwise translates the specified text from the source language into a target language. When a user submits a translation to the service, the system calls the **Translate** method for every translation unit (<*trans-unit*> element) in the XLIFF source data. The method's *text* parameter contains the source text of the unit.
Detect	You can use this method to call the logic of the translation service to automatically determine the language of the source text. Kentico CMS does not use the *Detect* method by default, but you can implement it if you need language detection functionality in your custom API.
Speak	Converts text to a stream of sound using the given service's text-to-speech engine. Kentico CMS does not use the *Speak* method by default, but you can implement it if you need text-to-speech functionality in your custom API. If your service does not support such an option, throw a not implemented exception.
IsAvailable	Checks whether the service is appropriately configured and ready to be used. For example, you can confirm that the target service is currently online, or load any credentials and API keys required by the service from the website settings and confirm their validity. The system only offers the service when translating documents and resource strings if the *IsAvailable* method returns a *true* value.

Defining machine service classes

This example demonstrates how to write a class providing functionality for a machine translation service. The sample class does not use a real translation service, it only converts the source text to upper case. When creating your own classes, replace the code of the methods with your own translation logic or call the API of the appropriate web service.

1. Open your web project in Visual Studio and add a new class into the **App_Code** folder (or

Old_App_Code if the project is installed as a web application). For example, name the class **SampleMachineTS.cs**.

2. Edit the class and change its code to the following:

[C#]

```
using System;

using CMS.TranslationServices;
using CMS.IO;

/// <summary>
/// Sample machine translation service.
/// </summary>
public class SampleMachineTS : AbstractMachineTranslationService
{
    /// <summary>
    /// Translates the specified text from the source language to the target
language.
    /// Must return the translated text as a string.
    /// </summary>
    /// <param name="text">Text to translate</param>
    /// <param name="sourceLang">Source language culture code</param>
    /// <param name="targetLang">Target language culture code</param>
    public override string Translate(string text, string sourceLang, string
targetLang)
    {
        // "Translates" the text to upper case.
        return text.ToUpper();
    }

    /// <summary>
    /// Automatically attempts to detect the language of the source text.
    /// Returns the culture code of the detected language.
    /// </summary>
    /// <param name="text">Text to be processed</param>
    public override string Detect(string text)
    {
        // Use your service to detect the language. This example always returns
English.
        return "en-US";
    }

    /// <summary>
    /// Returns a stream of sound generated by text-to-speech services (if
supported).
    /// </summary>
    /// <param name="text">Text to be processed</param>
    /// <param name="lang">Culture code of the text's language</param>
    public override Stream Speak(string text, string lang)
    {
        // This service provider does not support Text-to-speech.
        throw new NotImplementedException();
    }

    /// <summary>
```

```
    /// Checks the necessary prerequisities needed for the service to work,
    /// e.g. availability of credentials for connecting to the service etc.
    /// </summary>
    public override bool IsAvailable()
    {
        // This sample provider does not require any settings and does not depend
on
        // any other services, therefore is always available.
        return true;
    }
}
```

The system calls the methods of the class as needed when the given translation service is used.

3. Follow the instructions in the Loading service classes from App_Code topic to ensure that the application can access the custom class.

Registering machine services in the system

Once you have implemented the class with the required functionality, you need to register the translation service as an object in Kentico CMS:

1. Go to **Site Manager -> Development -> Translation services** and click **New translation service**.

2. Enter the following values into the service's properties:

- **Display name**: Sample machine service
- **Code name**: Leave the *(automatic)* option. The system will generate the code name as *Sample_machine_service* (based on the display name).
- **Service provider assembly name**: App_Code
- **Service provider class name**: SampleMachineTS
- **Is machine translation service**: Yes (checked)
- **Service is enabled**: yes (checked)

Click **Save**.

The service is now ready to be used.

Assigning icons to machine services

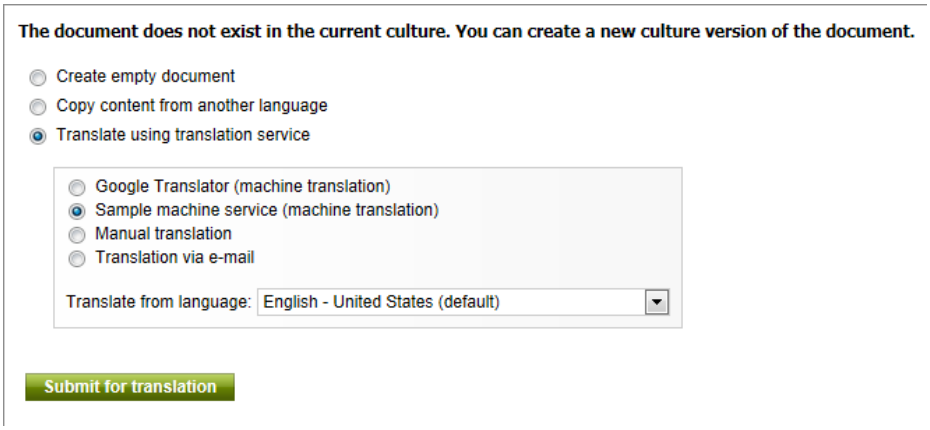
When editing resource strings for specific user interface cultures, users can translate the text from the default language by clicking on icons representing the available machine translation services. To set an icon for your custom service:

1. Open the *\App_Themes\Default\Images* folder in your web project.
2. Extract the **CMSModules** folder from the **Images.zip** archive and place it into the *\App_Themes\Default\Images* directory.
3. Add the icon image file into the *CMSModules\CMS_TranslationServices* folder. The name of the file must match the code name of the given translation service, so use *Sample_machine_service.png* for this example. The recommended size for the icon is 16 x 16 px.

The system automatically loads the service icon images from this folder.

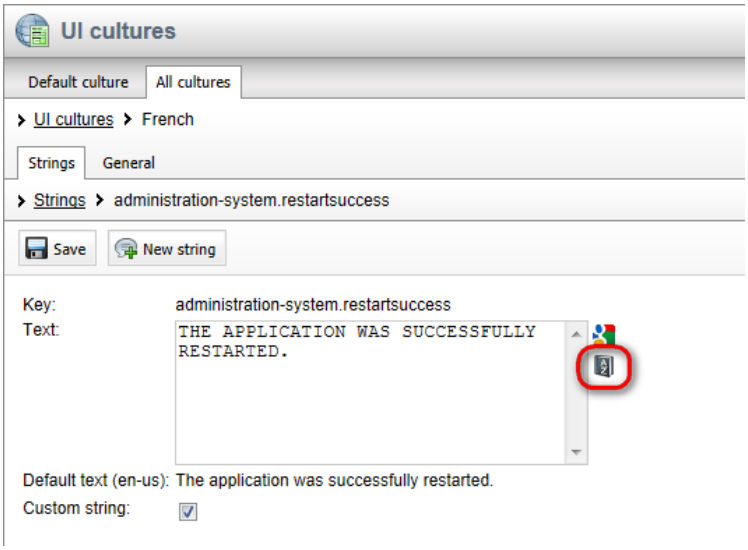
Result

When submitting documents for translation, the dialog offers the *Sample machine service* as one of the translation service options. Using this option creates the new language version of the document as a copy of the original content, with all characters converted to upper case.



Submitting a document for translation using the sample machine service

Users can also call the custom translation service when localizing resource strings.



"Translating" a resource string using the sample service

4.9.7.7.2 Creating human translation services

To develop a custom human translation service, you need to define a class that inherits from **AbstractHumanTranslationService** (found in the **CMS.TranslationServices** namespace).

The service class must override and implement the following abstract methods:

Method	Description

IsAvailable	Checks whether the service is appropriately configured and ready to be used. For example, you can confirm that valid connection credentials are specified for the service in the website settings or check whether there is sufficient credit to perform translations. When translating documents, users can only choose the service if the *IsAvailable* method returns a *true* value.
IsLanguageSupported	Checks whether the service supports translation to a specific language. The system calls this method before creating new translation submissions for the service. Users can only create submissions if the *IsLanguageSupported* method of the selected service returns a true value for the target language.
CreateSubmission	Delivers the translation assignment to the translators. Called when creating new translation submissions for the service or resubmitting existing ones.
CancelSubmission	Executed when a user cancels a translation submission in **CMS Desk -> Tools -> Translations**. Depending on the type of the service, you can delete the submission's physical file, call the appropriate service API or otherwise inform the translators that the submission has been canceled.
DownloadCompletedTranslations	Retrieves translated content from the service and imports it into the submission tickets. The system automatically calls this method when updating the status of translation submissions, which can be triggered by: • Users clicking **Update statuses** in **CMS Desk -> Tools -> Translations** • Execution of the **Translation retrieval** scheduled task

Defining human service classes

This example shows the implementation of a translation service that saves translation submissions into zip packages and exports them into a designated folder. It also provides a way to automatically load completed translations from an import folder. This sample service is a simplified version of the default Manual translation service.

1. Open your web project in Visual Studio and add a new class into the **App_Code** folder (or **Old_App_Code** if the project is installed as a web application). For example, name the class **SampleTS.cs**.

2. Change the class declaration and add references according to the following code:

[C#]

```
using System;
using System.Text;
using System.Data;
using System.Collections.Generic;
using System.Collections;
using System.Web;

using CMS.IO;
using CMS.SettingsProvider;
using CMS.GlobalHelper;
using CMS.TranslationServices;

/// <summary>
/// Sample human translation service.
/// </summary>
public class SampleTS : AbstractHumanTranslationService
{

}
```

3. Add the following properties into the class:

[C#]

```
/// <summary>
/// Gets the path of the folder to which the service exports translation
submissions.
/// </summary>
public string ExportFolder
{
    get
    {
        string folder = SettingsKeyProvider.GetStringValue(SiteName +
".SampleTranslationExportFolder");
        if (string.IsNullOrEmpty(folder))
        {
            // Sets a default export folder if the path can't be loaded from the
site settings.
            folder = "~/App_Data/Translations/Export/";
        }
        return URLHelper.GetPhysicalPath(folder);
    }
}

/// <summary>
/// Gets the path of the folder that the service checks for files containing
completed translations.
/// </summary>
public string ImportFolder
{
    get
    {
        string folder = SettingsKeyProvider.GetStringValue(SiteName +
".SampleTranslationImportFolder");
        if (string.IsNullOrEmpty(folder))
        {
```

```
            // Sets a default import folder if the path can't be loaded from the
site settings.
            folder = "~/App_Data/Translations/Import/";
        }
        return URLHelper.GetPhysicalPath(folder);
    }
}
```

The **ExportFolder** and **ImportFolder** properties load the translation folder paths from the website settings. If the setting values are not available, the properties return default folder paths. The **SiteName** property used to build the setting key names is inherited from the parent class. It gets the code name of the site from which the translation was submitted.

4. Define the required methods inside the class:

IsAvailable

[C#]

```
/// <summary>
/// Checks if all conditions required to run the service are fulfilled.
/// </summary>
public override bool IsAvailable()
{
    // This service is always available.
    return true;
}
```

The sample service only needs to know the paths for its import and export folders, but it uses default paths if the values are not specified through the settings. The method returns a *true* value to indicate that the service is always ready.

IsLanguageSupported

[C#]

```
/// <summary>
/// Checks if the service supports a specific language.
/// </summary>
/// <param name="langCode">Culture code of the language to be checked</param>
public override bool IsLanguageSupported(string langCode)
{
    // All languages are supported.
    return true;
}
```

CreateSubmission

[C#]

```
/// <summary>
/// Creates a new submission or resubmits it if the submission ticket already
exists.
/// Returns an empty string if all operations are successful or the details of the
encountered error.
/// </summary>
/// <param name="submission">Info object representing the translation submission</
param>
public override string CreateSubmission(TranslationSubmissionInfo submission)
{
    try
    {
        if (submission != null)
        {
            // Gets the path of the zip file containing the submission.
            string path = null;
            if (string.IsNullOrEmpty(submission.SubmissionTicket))
            {
                path = Path.Combine(this.ExportFolder,
ValidationHelper.GetSafeFileName(submission.SubmissionName) + ".zip");
                path = FileHelper.GetUniqueFileName(path);

                // Assigns the zip file name as the ticket ID of the new
submission.
                submission.SubmissionTicket = Path.GetFileName(path);
            }
            else
            {
                // The resubmit action uses the existing file path stored in the
ticket ID.
                path = Path.Combine(this.ExportFolder,
submission.SubmissionTicket);
            }

            // Writes the zip file under the specified path. Overwrites the file
if it exists.
            DirectoryHelper.EnsureDiskPath(path, null);
            using (FileStream stream = File.Create(path))
            {
                TranslationServiceHelper.WriteSubmissionInZIP(submission, stream);
            }
        }
    }
    catch (Exception ex)
    {
        TranslationServiceHelper.LogEvent(ex);
        return ex.Message;
    }
    return null;
}
```

The **TranslationServiceHelper.WriteSubmissionInZip** method creates the zip archive with the translation assignment. The archive contains the source text of the submitted documents in XLIFF format and an HTML instructions file with the details entered for the translation.

CancelSubmission

[C#]

```
/// <summary>
/// Cancels the specified submission.
/// Returns an empty string if all operations are successful or the details of the
encountered error.
/// </summary>
/// <param name="submission">Info object representing the canceled submission</
param>
public override string CancelSubmission(TranslationSubmissionInfo submission)
{
    try
    {
        if (submission != null)
        {
            // Tries to delete the assignment zip file (loads the file name from
the submission ticket ID).
            string path = Path.Combine(this.ExportFolder,
submission.SubmissionTicket);
            if (File.Exists(path))
            {
                File.Delete(path);
            }
        }
    }
    catch (Exception ex)
    {
        TranslationServiceHelper.LogEvent(ex);
        return ex.Message;
    }
    return null;
}
```

DownloadCompletedTranslations

[C#]

```
/// <summary>
/// Retrieves completed XLIFF translation files from the service and imports them
into the system.
/// Returns an empty string if all operations are successful or the details of the
encountered error.
/// </summary>
public override string DownloadCompletedTranslations()
{
    try
    {
        if (Directory.Exists(this.ImportFolder))
        {
            // Gets all zip files from the import folder.
            string[] files = Directory.GetFiles(this.ImportFolder, "*.zip");
            foreach (string filePath in files)
            {
                string file = Path.GetFileName(filePath);

                // Gets all translation submissions matching the zip file name.
```

```
                DataSet ds = TranslationSubmissionInfoProvider.GetSubmissions
("SubmissionTicket = '" + SqlHelperClass.GetSafeQueryString(file, false) + "'",
null);
                if (!DataHelper.DataSourceIsEmpty(ds))
                {
                    foreach (DataRow dr in ds.Tables[0].Rows)
                    {
                        TranslationSubmissionInfo submission = new
TranslationSubmissionInfo(dr);

                        // Only imports content for submissions in the 'Waiting
for translation' status.
                        if (submission.SubmissionStatus ==
TranslationStatusEnum.WaitingForTranslation)
                        {
                            // Gets the zip name from the submission ticket.
                            string fileName = submission.SubmissionTicket;
                            string path = Path.Combine(this.ImportFolder,
fileName);
                            if (File.Exists(path))
                            {
                                // Imports XLIFF file content from the zip
package.
                                string err =
TranslationServiceHelper.ImportXLIFFfromZIP(submission, FileStream.New(path,
FileMode.Open));
                                if (string.IsNullOrEmpty(err))
                                {
                                    // Changes the status to 'Translation ready'
and saves the submission.
                                    submission.SubmissionStatus =
TranslationStatusEnum.TranslationReady;
                                    TranslationSubmissionInfoProvider.SetTranslati
onSubmissionInfo(submission);
                                }
                                else
                                {
                                    return err;
                                }
                            }
                        }
                    }
                }
            }
        }
        return null;
    }
    catch (Exception ex)
    {
        TranslationServiceHelper.LogEvent(ex);
        return ex.Message;
    }
}
```

The **TranslationServiceHelper.ImportXLIFFfromZIP** method loads the translated content from all XLIFF files in the zip package and saves it into the submission ticket.

5. Follow the instructions in the Loading service classes from App_Code topic to ensure that the

application can access the custom class.

Registering human translation services

Once you have implemented the class with the required functionality, you need to register the translation service as an object in Kentico CMS:

1. Go to **Site Manager -> Development -> Translation services** and click **New translation service**.

2. Enter the following values into the service's properties:

- **Display name**: Sample translation service
- **Service provider assembly name**: App_Code
- **Service provider class name**: SampleTS
- **Service is enabled**: yes (checked)

- **Service supports submitting instructions** - yes
- **Supports prioritizing of submissions** - yes
- **Supports submission deadlines** - yes

Click **Save**.

The service is now ready to be used.

Adding custom settings for translation services

To allow administrators to configure the import and export folder paths of the service, you need to create custom settings:

1. Go to **Site Manager -> Development -> Custom settings** and click **New settings group**.

2. Enter the following names for the group:

- **Display name**: Sample translation paths
- **Code name**: SampleTranslationPaths

Click **Save**.

3. Click **New settings key** under the new *Sample translation paths* section and define two setting keys:

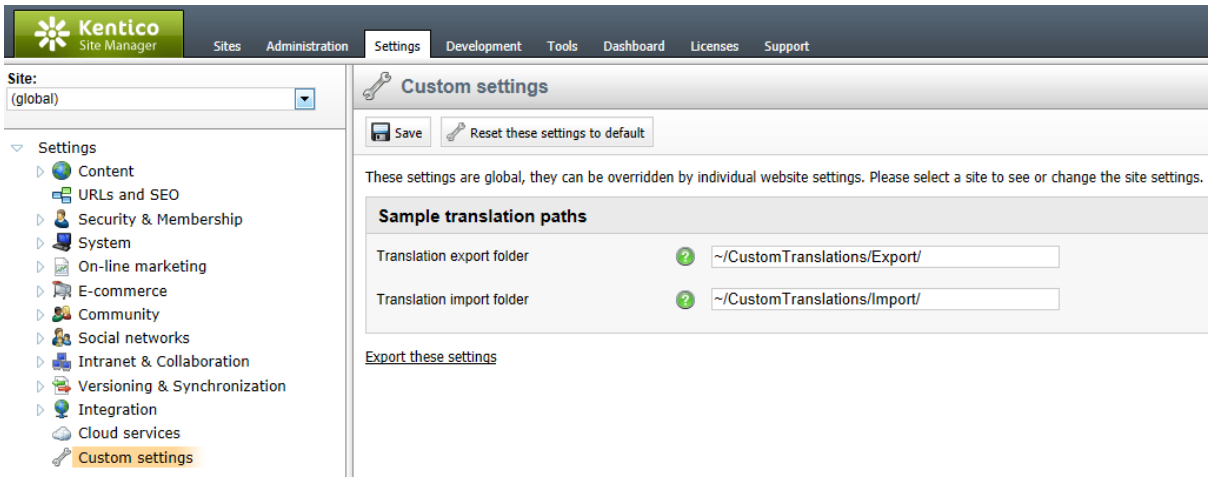
- **Display name**: Translation export folder
- **Code name**: SampleTranslationExportFolder (matches the name of the key loaded by the *ExportFolder* property in the code of the sample service class)
- **Description**: Sets the path of the folder where the system creates translation submissions. If empty, the ~/App_Data/Translations/Export/ folder is used.
- **Type**: String

- **Display name**: Translation import folder
- **Code name**: SampleTranslationImportFolder (matches the name of the key loaded by the *ImportFolder* property in the code of the sample service class)

- **Description**: Sets the path of the folder from which the system loads completed translation packages. If empty, the ~/App_Data/Translations/Import/ folder is used.
- **Type**: String

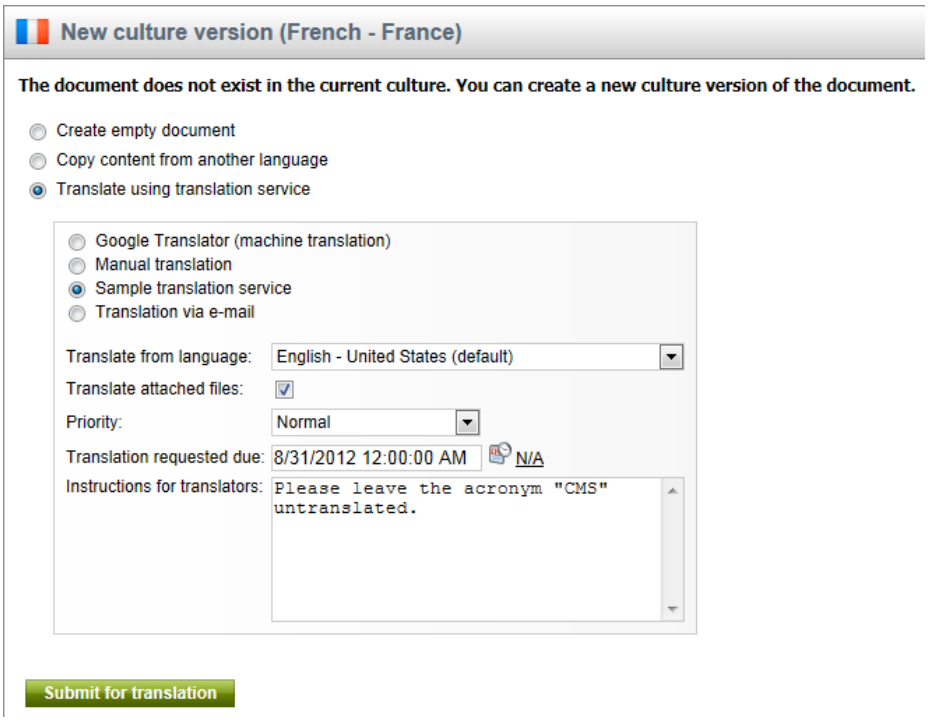
Click **Save** for each key.

You can now set the service's folder paths for specific websites in **Site Manager -> Settings -> Custom settings**.



Result

When submitting documents for translation, the dialog offers the *Sample translation service* as one of the translation options.



Submitting a document for translation using the sample service

Try translating a document using the custom service:

1. Click **Submit for translation** in the *New culture version* dialog. The system creates a new submission and adds the translation zip package into the specified export folder.

2. Translate the content and add the modified *.xlf* file into a new zip package (with the same name as the export zip file). Place the zip package into the import folder.

3. Go to **CMS Desk -> Tools -> Translations** and click **Update statuses**. The service imports the translated content and switches the matching submission to the *Translation ready* status.

4. Click the **Process submission** action of the submission. The system transfers the translated content into the corresponding document and changes the submission status to *Translation imported*.

The document is now available in the target language.

4.9.7.7.3 Loading service classes from App_Code

For translation services defined in the *App_Code* folder, you need to ensure that the system loads the appropriate class when calling the given service. You can find additional information related to this topic in Registering custom classes in App_Code.

1. Create a dedicated class in the **App_Code** folder (or **Old_App_Code** if the project is installed as a web application). For example, name the class **ClassLoader.cs**.

2. Edit the class and add the following reference:

[C#]

```
using CMS.SettingsProvider;
```

3. Delete the default class declaration and its content. Instead, extend the **CMSModuleLoader** partial class and define a new attribute inheriting from *CMS.SettingsProvider.CMSLoaderAttribute*:

[C#]

```
[ClassLoader]
public partial class CMSModuleLoader
{
    /// <summary>
    /// Attribute class for loading custom classes.
    /// </summary>
    private class ClassLoader : CMSLoaderAttribute
    {

    }
}
```

4. Add the following code into the **ClassLoader** attribute class:

Warning!

This loads the sample machine and human translation providers created according to the instructions in this chapter.

You need to adjust the code to match the names of your own custom service classes. If you do not have the sample classes in your web project, delete or comment out the corresponding cases in the switch statement.

[C#]

```
/// <summary>
/// Called automatically when the application starts.
/// </summary>
public override void Init()
{
    // Assigns a handler for the OnGetCustomClass event.
    ClassHelper.OnGetCustomClass += ClassHelper_OnGetCustomClass;
}

/// <summary>
/// Gets a custom class object based on the given parameters.
/// </summary>
private void ClassHelper_OnGetCustomClass(object sender, ClassEventArgs e)
{
    if (e.Object == null)
```

```
    {
        // Checks the name of the requested class.
        switch (e.ClassName)
        {
            // Gets an instance of the SampleMachineTS class.
            case "SampleMachineTS":
                e.Object = new SampleMachineTS();
                break;

            // Gets an instance of the SampleTS class.
            case "SampleTS":
                e.Object = new SampleTS();
                break;
        }
    }
}
```

The value of the **ClassName** property of the **ClassHelper_OnGetCustomClass** handler's *ClassEventArgs* parameter matches the **Service provider class name** specified for the service in the system. The handler checks the class name to identify which translation service is being requested and then passes on an instance of the appropriate class.

4.9.7.7.4 Translation service properties

You can configure the following properties when creating or editing translation services in **Site Manager -> Development -> Translation services**:

Display name	Sets the name of the service displayed to users in the administration interface, e.g. in the list of possible options when submitting a document for translation.
Code name	Sets a name that serves as a unique identifier for the service, for example in the API. You can leave the default *(automatic)* option to have the system generate an appropriate code name based on the display name.
Service provider assembly name	Must contain the name of the assembly where the service is implemented. Enter *App_Code* for classes defined in the application's App_Code (or Old_App_Code) folder.
Service provider class name	Specifies the exact class (including any namespaces) that defines the functionality of the translation service.
Is machine translation service	Indicates whether the service is based on machine translation. If disabled, it is considered as a human translation service. • When translating documents using a machine service, the system creates the new document versions in the target language immediately. • With human translation services, there is a waiting period and the system tracks translations through submission tickets.
Service is enabled	Can be used to manually enable or disable the service. Users can only choose enabled services when submitting documents for translation.

Service supports submitting instructions	If enabled, users can include additional instructions (for human translators) when creating submissions for this service.
Supports prioritizing of submissions	If checked, users are able to set a priority (low, normal or high) for individual translation submissions that use the service.
Supports submission deadlines	Indicates if users are allowed to specify deadline dates for translation submissions that use this service.
Service parameter	The text data entered into this field may be accessed from within the code implementing the service. It serves as a string parameter through which you can modify the functionality of the service without having to edit its code. You can load the value in the code of your service classes through the **CustomParameter** property, which is inherited from the parent class (*AbstractMachineTranslationService* or *AbstractHumanTranslationService*). **C# example**: `CMS.GlobalHelper.ValidationHelper.GetString(CustomParamete`

4.9.7.8 Translating via the REST service

Kentico CMS provides a way to retrieve and submit translation data through REST. This service allows you to manage document translations using HTTP request methods. You can utilize the REST service in the code of your Custom translation services to transfer data in and out of the system.

Use the following root URL to access the REST translation service:

~/rest/translate/

Replace the ~ character with the domain name and virtual directory of your website, for example: *http://localhost/KenticoCMS/rest/translate/*

Enabling the REST translation service

To set up your website for handling translation operations via the REST service:

1. Ensure that all REST prerequisites are fulfilled on the machine hosting your website.

2. Configure your application's *web.config* according to the instructions in the Configuration for REST topic. The settings in **Site Manager -> Settings -> Integration -> REST** do *not* affect the translation service. You can leave the general Kentico CMS REST service disabled.

3. Go to **Site Manager -> Settings -> Content -> Translation services** and enable the **Enable REST translation service** setting.

Retrieving translation data from the system

To retrieve document translation data, send an HTTP GET request to a URL in one of the following formats:

- **~/rest/translate/content/currentsite/<*culture code*>/document/<*alias path*>**
- **~/rest/translate/content/site/<*site code name*>/<*culture code*>/document/<*alias path*>**

Set the target language of the translation by adding the corresponding culture code into the **targetlanguage** URL parameter. The system automatically fills in the source language based on the language version of the specified document.

The request returns a response containing XLIFF data that you can use as the translation source. The REST service only supports retrieval of a single object per request.

Examples:

~/rest/translate/content/site/CorporateSite/en-us/document/Services?targetlanguage=fr-fr
~/rest/translate/content/currentsite/fr-fr/document/Company/Careers?targetlanguage=cs-cz

You can also use the following boolean parameters in the URL:

- **TranslateDocCoupledData** (true by default)
  If true, the returned translation source data includes the content of the document's coupled data fields (i.e. the fields of the specific document type). The service only exports fields that have the *Translate field* flag enabled in the document type's form definition.

- **TranslateEditableItems** (true by default)
  If true, the translation data includes the content of the document's editable regions.

- **TranslateWebpartProperties** (false by default)
  If true, the translation data includes the properties of web parts placed on the document. The service only exports the values of properties that have the *Translate field* flag enabled in the web part definition.

- **TranslateAttachments** (false by default)
  If true, the translation data includes the document's file attachments. Attachment translation must also be allowed in the website's settings.

See also: Configuring content for translation

Submitting completed translations

To import a translation into the system, send a POST request to the root URL of the REST translation service:

~/rest/translate/

Add the XLIFF content of the completed translation into the body of the request.

The root URL of the service is the only access point used for submitting translations. The system automatically identifies the target document based on the XLIFF data. When the system processes the

request, it creates the appropriate language version of the document (if necessary) and inserts the translation data.

Sample POST request:

```
POST http://localhost/KenticoCMS/rest/translate/
User-Agent: REST client
Authorization: Basic <enter <username>:<password> encoded in Base64>
Host: localhost
Content-Type: text/xml

<xliff version="1.2">
  <file original="cms.document;9" source-language="en-US" target-language="fr-FR"
datatype="htmlbody">
    <body>
      <trans-unit id="menuitemname">
        <source><![CDATA[Company]]></source>
        <target><![CDATA[Entreprise]]></target>
      </trans-unit>
      <trans-unit id="documentname">
        <source><![CDATA[Company]]></source>
        <target><![CDATA[Entreprise]]></target>
      </trans-unit>
      <trans-unit id="editable;we">
        <source><![CDATA[<p>Source text.</p>]]></source>
        <target><![CDATA[<p>Translated text.</p>]]></target>
      </trans-unit>
      <trans-unit id="editable;wh">
        <source><![CDATA[Company]]></source>
        <target><![CDATA[Entreprise]]></target>
      </trans-unit>
    </body>
  </file>
</xliff>
```

4.9.7.9 Security

There are several module permissions that determine how users are allowed to work with translation services. To configure these permissions for specific roles, go to **Site Manager** / **CMS Desk -> Administration -> Permissions**.

Module	Permission	Description
Content	Submit for translation	Allows users to translate documents using the available services.
	Create	Required to create documents, including specific language versions. Users need this permission to translate documents into new languages. Alternatively, you can assign the *Create* permission only for specific document types.
Translations	Read	Allows users to view translation submissions and their data in **CMS**

		Desk -> Tools -> Translations.
	Modify	Allows users to create, edit, and delete translation submissions in **CMS Desk -> Tools -> Translations**. Also required to perform all submission-related actions (re-submitting, canceling, updating status, processing).

4.10 Workflow and versioning

4.10.1 Overview

Kentico CMS allows you to organize and control the process of creating, updating and publishing documents on your website by applying a workflow on them.

In this chapter, you will learn how to define and apply workflow processes for your documents. You will also learn how to manage the workflow process of a particular document in CMS Desk.

What is a workflow?

Workflow is a sequence of steps that define the life cycle of a document. This allows for setting up a reviewing and approval process to ensure quality of content and design. In such a process, you can specify the roles that different people will play and the places in the flow where the individual people will have influence on a particular document.

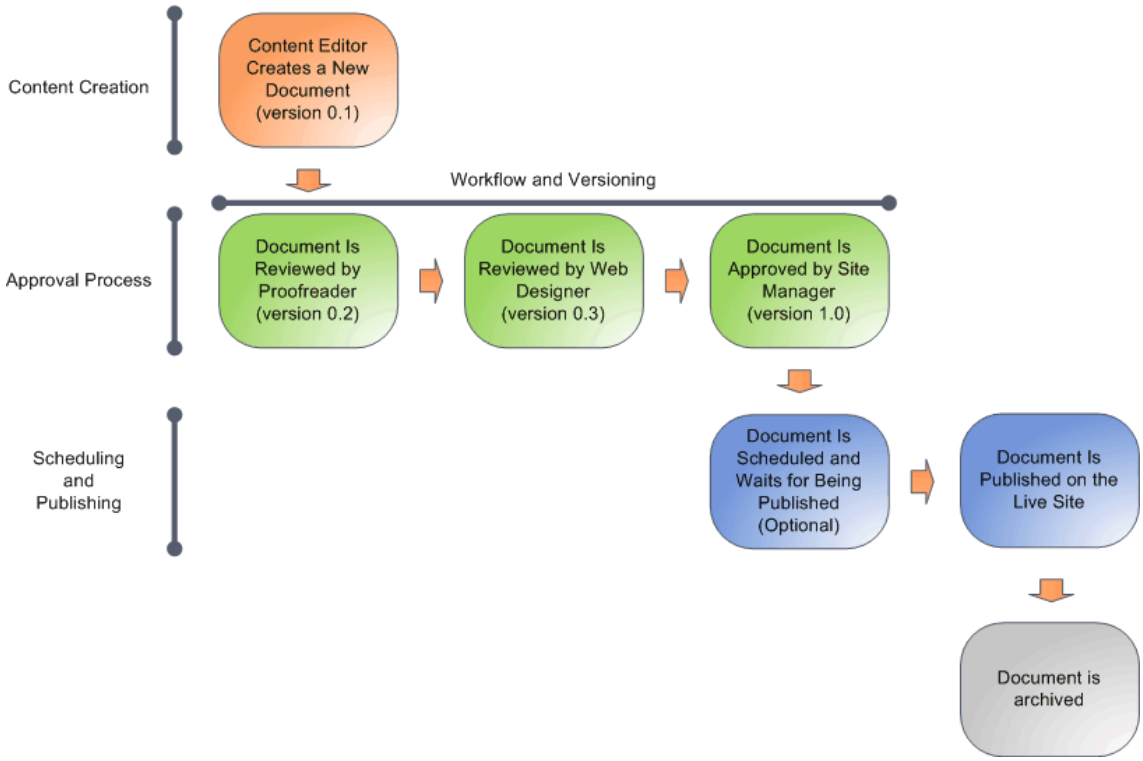
An example of a simple workflow process could be a hypothetical website that publishes scientific articles. An author can write an article and send it for approval to an editor, who will make corrections and submit the article to the head of the respective department. The head will then publish the document, thus making it publicly available to read on the website.

What features are available?

Kentico CMS allows you to use workflow for all documents in the content tree, including uploaded files. It consists of the following features:

- The **workflow** support allows you to organize the process of content creation, updates and publishing on your website.
- **Advanced workflow** adds support of branching the workflow process based on macro conditions and user decisions. It also allows for automatic manipulation of the documents as they go through the workflow process.
- The **versioning** support is tightly bound with workflow and allows you to store, view and roll back previous versions of the content.
- **Versioning without workflow** creates a new version whenever a document is modified, while the whole editing process remains as if the document was using no workflow at all.
- The **content locking (check-in/check-out)** support allows you to avoid concurrent modifications of the same document by multiple users.
- The **preview** support allows you to see the content in the context of the site before it's published. This feature is only available for documents that use workflow - you can preview the documents before switching them to the following step.
- The **archiving** support allows you to archive a document. Such a document is no longer displayed on the website, but it's kept in the content tree and can be re-published at any time later.

The following figure shows an example of document life cycle with a workflow:



The workflow process consists of any number of custom steps. Only authorized users or roles are allowed to modify and approve/reject a document in a particular step. Any authorized user can switch a document to the following or the preceding step. The current workflow step of a document can be found on its **Properties -> Workflow** tab. It is also indicated by the document status icons displayed next to documents in the content tree.

What type of workflow should I use?

Kentico CMS offers two types of workflows - basic and advanced.

Basic workflow

In a basic workflow, one step is followed by another in a direct sequence. A basic workflow cycle of a document starts in the **Edit** step, which means the document is newly created or that there are yet unapproved changes in the document. After going through the defined approval steps, the document gets into the **Published** step. At that point the system makes the last version of the document visible on the live site.

Advanced workflow

Advanced workflow significantly differs from basic workflow in the way how you can design the flow of steps. You can let a document go through different steps based on various conditions, you can allow editors to choose which step they want the document to be moved to, or you can set timeouts for automatically moving documents from one step to another.

4.10.2 Defining a workflow

This chapter explains how to define a workflow from scratch. You will learn how to:

- create a workflow
- apply the workflow on documents by creating workflow scopes
- define workflow steps for basic and advanced workflow

4.10.2.1 Creating a workflow

Workflow is a sequence of steps that define the life cycle of a document. This allows for setting up a reviewing and approval process to ensure quality of content and design. In such a process, you can specify the roles that different people will play and the places in the flow where the individual people will have influence on a particular document.

You can create and manage workflows in **Site Manager -> Development -> Workflows**.

Creating a basic workflow

In a basic workflow, one step is followed by another in a direct sequence. A basic workflow cycle of a document starts in the **Edit** step, which means the document is newly created or that there are yet unapproved changes in the document. After going through the defined approval steps, the document gets into the **Published** step. At that point the system makes the last version of the document visible on the live site.

1. Log in to **Site Manager** and navigate to **Development -> Workflows**.
2. Click **New workflow**.
3. Choose a name for the new workflow and type it in the **Display name** text box.
4. Click **Save**.

The workflow appears in the list on the **Workflows** page. Now you can proceed with applying the workflow on documents and creating workflow steps.

Creating an advanced workflow

Advanced workflow significantly differs from basic workflow in the way how you design the flow of steps. You can let a document go through different steps based on various conditions, you can allow editors to choose which step they want the document to be moved to, or you can set timeouts for automatically moving documents from one step to another.

1. Log in to **Site Manager** and navigate to **Development -> Workflows**.
2. Click **New advanced workflow**.
3. Choose a name for the new workflow and type it in the **Display name** text box.
4. Click **Save**.

The workflow appears in the list on the **Workflows** page. Now you can proceed with applying the

workflow on documents and designing the workflow process.

Converting a basic workflow to advanced workflow

The Workflow page allows you to convert a basic workflow to an advanced workflow.

1. Log in to **Site Manager** and navigate to **Development -> Workflows**.

2. Edit () a basic workflow.

3. If the workflow has the **Automatically publish changes** property turned on, turn the property off and save the workflow.

4. Click **Convert to advanced workflow**. A message will appear asking if you really want to convert the workflow to advanced workflow.

5. Click **OK** to finish the conversion or click **Cancel** to return to the workflow editing page.

The system converted the basic workflow to an advanced workflow. Also, all the defined steps have been converted into advanced workflow steps.

4.10.2.2 Applying workflow on documents

A workflow scope defines the conditions based on which a workflow will be applied to documents. When creating a workflow scope, you have to specify a document's alias path and whether you want to include child documents under this path. Then you can narrow down the set of covered documents by specifying some of the following parameters:

- document type
- culture
- macro condition

The system offers two types of scopes - **allowed** and **excluded**. This way you can apply a workflow on a large section of a site and then exclude sections, which you don't want the workflow to be applied on.

Workflow priority

If a document matches scopes of multiple workflows, the workflow with the most specific parameters is applied on the document. Scopes are applied with the following priorities, from highest to lowest:

1. Scope with specified **document type and culture**
2. Scope with specified **document type**
3. Scope with specified **culture**
4. Scope **without** specified document type and culture.
5. Scope with the longest **path**.
6. Scope that covers only the specified **document without children**.
7. Scope that has its type set to **Excluded**.
8. Scope with specified **macro condition**.

Workflow priority examples

The following table presents most common scenarios, where scopes from two different workflows (**Scope 1** and **Scope 2** columns in the table) conflict with each other. The **Result** column explains which scope takes priority.

Scope 1 setup	Scope 2 setup	Result
Alias path: /News	**Alias path:** /News **Document type:** Press releases	Press releases under the News document will use Scope 2, the News document itself and its other children will use Scope 1.
Alias path: /Products	**Alias path:** /Products/Smartphones	All documents under the Products document will use Scope 1, except for the Smartphones document and its child documents, which will use Scope 2.
Alias path: /Products	**Alias path:** /Products/Smartphones **Coverage**: Only specified document	All documents under the Products document will use Scope 1, except for the Smartphones document, which will use Scope 2. However, children under the Smartphones document will use Scope 1.

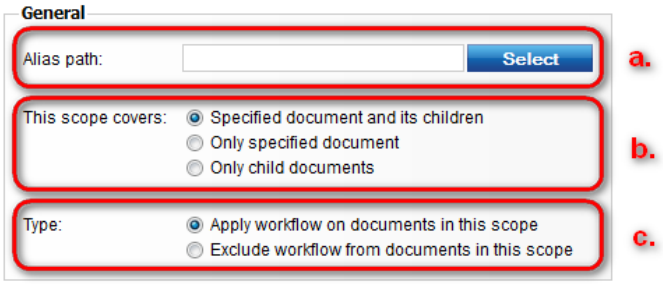
Creating a workflow scope

1. Edit an existing workflow and select the **Scope** tab.

2. Choose a site you want the scope to be effective on using the **Site** drop-down list and click **New workflow scope**.

3. Define the part of the content tree that you want to be covered by the scope.

   a. Click **Select** next to the **Alias path** property and in the dialog that opens, choose the document you want covered by the scope.
   b. Specify whether you want to cover the document and its child documents, only the document or only its children.
   c. Select if you want the scope to be allowed or excluded. Keep the following behavior in mind when working with exclusions.



Tip

If you want the scope to cover the whole site, or you want to specify only the parameters in the Advanced section, enter "**/**" (i.e. the path of the root document) in the **Alias path** property. Make sure that the scope covers the specified document including its children.

4. (Optional) Choose a document type. The workflow will be applied only on documents of this type.

a. Make sure the desired document type is assigned to the site you selected in step 2.
b. If you had to assign the document type to the site, click **Save** to refresh the page and load the newly assigned document type.
c. Select the document type from the **Document type** drop-down list.

5. (Optional) Specify a culture. The workflow will be applied only on documents in this culture.

a. Make sure the culture is assigned to the site you selected in step 2.
b. If you had to assign the culture to the site, click **Save** to refresh the page and load the newly assigned culture.
c. Select the culture from the **Culture** drop-down list.

6. (Optional) Enter a macro condition. If the condition is met when starting a workflow cycle on a document within the scope, the workflow will be applied to the document.

7. Click **Save**.

8. If you want to apply the workflow to an existing document within the scope, make changes to the document.

- If you're using basic workflow, navigate to the document in CMS Desk and click **Save**.
- If you're using advanced workflow, navigate to the document in CMS Desk and click **New version**.

Please note

Exclude workflow from documents in this option applies only to the workflow under which it has been specified and does not override scopes in different workflows.

For example, under *Workflow 1*, you have *"/Products" and its children* allowed and *"/Products/Smartphones"* excluded. Under *Workflow 2,* however, you have *"/Products" and its children* allowed and no exclusion specified. The exclusion specified under *Workflow 1* will have no effect on the scope of *Workflow 2*.

4.10.2.3 Defining workflow steps

This chapter explains how to design a workflow process. Here you can find instructions how to accomplish the following tasks:

- creating basic workflow steps
- designing an advanced workflow process
- configuring workflow operators, i.e., users and roles allowed for approving or rejecting in different steps
- configuring e-mail notifications

4.10.2.3.1 Creating and managing basic workflow steps

A workflow consists of a series of steps. Each step in a workflow defines a part of the document approval process. Each workflow process in Kentico CMS has three default steps, whose order cannot be changed. However, you can add any number of custom steps between the Edit and Published step.

Edit step

The Edit step is the first step in every workflow process. When you create a document, it starts in the Edit step. When you make changes to a published document, it gets moved to the Edit step. Live site visitors will not see changes made to a document while in the Edit step until it gets moved to the Published step.

Published step

When a document reaches the Published step, the system makes it visible to live site visitors. After a document is published, you can make changes to it, while live site visitors see the published version of the document.

Archived step

When you move a document to the Archived step, it gets pulled off from the live site. Documents in the Archived step will not be accessible by live site visitors. Making changes to an archived document moves it back to the Edit step.

Creating a workflow step

1. Edit a basic workflow and switch to the **Steps** tab.

2. Click **New workflow step**.

3. Enter a name into the **Display name** field. This is the name that will be displayed to editors when viewing documents that are in the step.

4. Click **Save**.

The system inserts the step immediately before the Published step. You can view the step on the Steps page under the workflow you're editing.

You can proceed with configuring operators and e-mail notifications for this step.

Rearranging workflow steps

You can move steps up and down using the **Move up** and **Move down** buttons in the list of steps. However, you cannot move the default steps, nor can you move any custom step before the Edit step or past the Published step. If you wish to design a flexible workflow process where the cycle doesn't begin and end with the default steps, consider using an advanced workflow.

Allowing or denying users to manage documents in the step

Refer to the Configuring workflow operators topic to learn how to set up security rules of a step.

4.10.2.3.2 Designing advanced workflow

In this chapter, you will learn how to design an advanced workflow process. The topics in this chapter explain the different types of workflow steps, their configuration, and usage.

This chapter contains the following topics and sub-chapters:

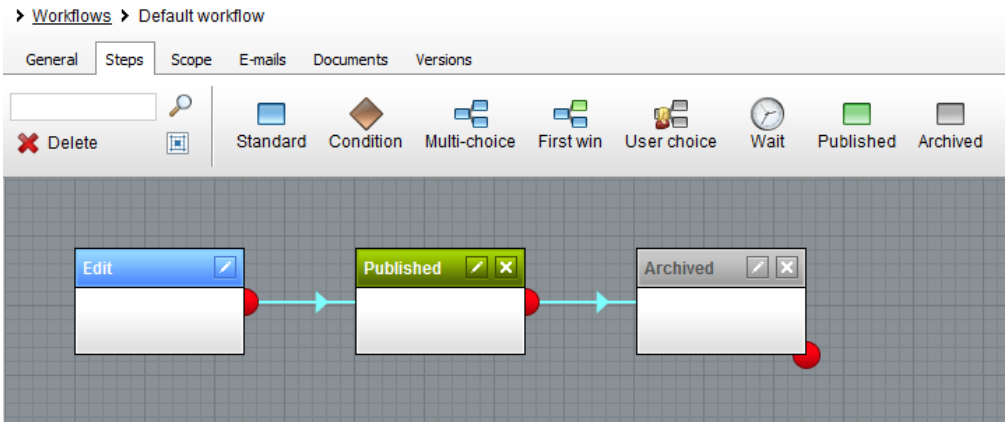
- Introduction to workflow designer
- Connecting steps
- Adding and configuring steps
- Defining if/else conditions
- Defining user decisions
- Defining automatic decisions
- Setting timeouts for automatic approving
- Adding action steps

4.10.2.3.2.1 Introduction to workflow designer

This topic describes the workflow designer, which is a tool for creating complex workflow processes.

You can find the workflow designer on the **Steps** tab when editing an advanced workflow. It consists of a toolbar and a grid. By placing workflow steps from the toolbar on the grid and connecting them, you form a graph that represents the workflow process.

The following image shows a basic workflow process consisting of the Edit, Published and Archived steps, as it appears in the workflow designer. The steps are connected with transition lines. The direction in which the arrows point is the direction in which a document will be moved through the steps.



If your graph doesn't fit into the designer area, you can click and drag an unoccupied area of the grid to move it and expose additional free space.

Step types

In advanced workflow, you can use the following types of steps:

- **Standard** step allows approving or rejecting a document.
- **Condition** allows splitting the workflow process into two branches, according to whether the condition is met or not.
- **Multi-choice** step allows splitting the workflow process into multiple branches. You can define a number of conditions, one of which will determine the branch the process will automatically continue in. If more than one condition is met, users will be able to choose.
- **First-win** step allows defining a number of conditions, where the first condition that is met determines the branch the process will automatically continue in.
- **User choice** allows defining branches of the process, where the user will chose the branch the process will continue in.
- **Wait** allows to define a time period after which a document will be automatically moved to the next step.
- **Published** step puts a document's latest version up to the live site.
- **Archived** step makes a document inaccessible from the live site.

You can also embed actions that will be automatically performed when a document reaches an action step:

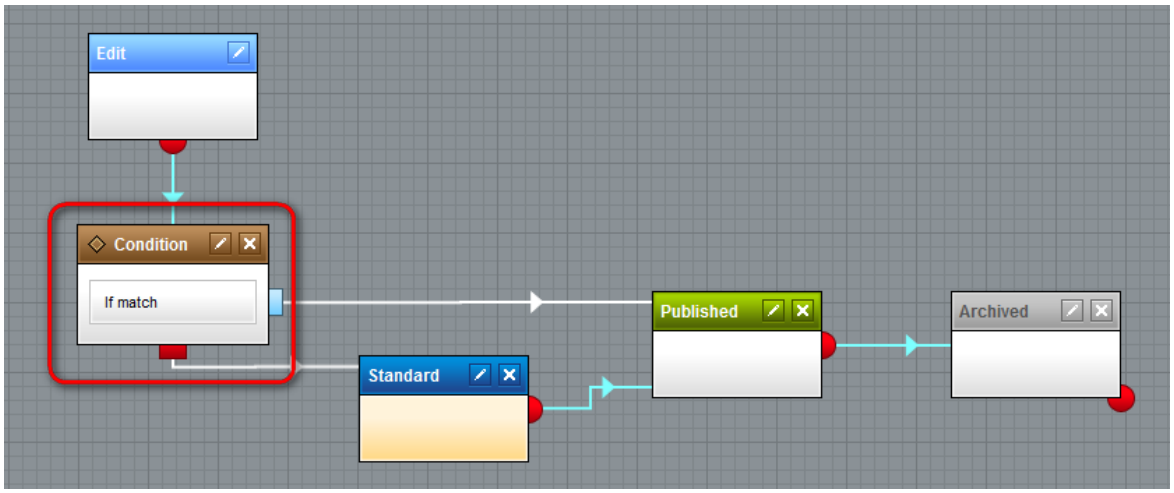
- **Copy document** - copies a specified document to another location.
- **Delete document** - deletes a specified document.
- **Import translation** - imports translations of the document that are ready.
- **Link document** - creates a linked document.
- **Move document** - moves a specified document to another location.
- **Publish to Facebook** - automatically sends a predefined post on Facebook.
- **Publish to Twitter** - automatically sends a predefined post on Twitter.
- **Send for translation** - submits a document for translation using the specified translation service.
- **Send e-mails** - sends e-mails based on a specified e-mail template.
- **Send notification e-mails** - sends e-mails according to the e-mail settings defined for the step that immediately follows.
- **Set document property** - assigns a specified value to a document's property.
- **Synchronize document** - synchronizes a specified document to all enabled staging servers.

You can also script your own actions.

Source points

The red points are **source points**. You can drag a source point with your mouse and drop it on another step to create a connection between the two steps.

Some steps can have more than a single source point, as you can see in the following image.



The highlighted step is an *if/else* condition, which allows you to create branches of the process. Therefore, it has two source points. The blue point is used when the specified condition is met. The red point indicates where the process should continue if the condition is not met. Steps that offer multiple conditions have multiple blue source points, one for each condition.

To learn how to connect steps together, refer to the Connecting steps topic.

4.10.2.3.2.2 Adding and configuring steps

An advanced workflow consists of a set of interconnected steps. The steps represent state of documents under the workflow, allow you to define automatic or user-made decisions and perform scripted actions.

To design an advanced workflow process, use the workflow designer.

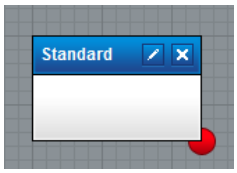
Adding steps

1. Click a step button on the toolbar and hold the mouse button.

2. Drag the step onto the grid.

3. Release the mouse button.

The step appears in the grid as in the following image.

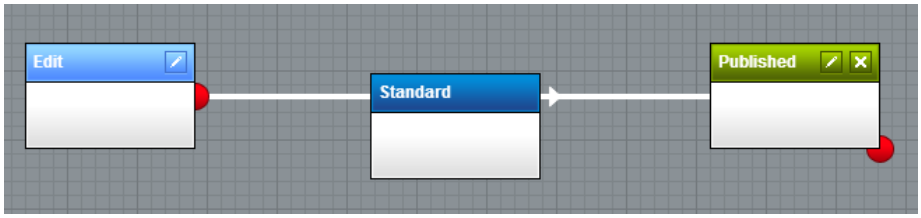


Placing steps onto an existing connection

1. Click a step button on the toolbar and hold the mouse button.

2. Drag the step onto an existing connection between two steps.

The connection increases in thickness once it is in the correct position under your mouse pointer.



3. Release the mouse button.

The step automatically splits the connection and connects to the steps.

Note that you can only place steps directly from the toolbar to be able to split a connection.

Renaming steps

1. Double-click the step name in the graph.

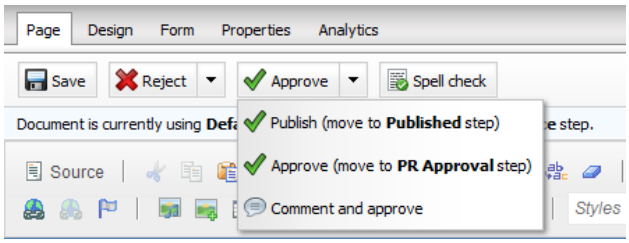
2. Type in a new name.

3. Press **Enter** to save the new name.

Changing the action button text

You can change how buttons that enable users to move a document to different steps look. The default text is determined by the type of step that the button sends the document to.

The following image shows the action buttons available when editing a document that is in a **User choice** step, that has two choices. The first choice sends the document to the Published step, the second sends the document to a standard approval step named "PR Approval".



You can change the text for all steps that require user interaction and for choices, if they are available in the particular step.

1. Edit the step where you want the button to appear differently.

2. If the step you're editing allows defining cases, edit the case that you want to appear differently.

3. Fill one or both the following fields:

- User action text
- User action tooltip

4. Click **Save**.

4.10.2.3.2.3 Connecting steps

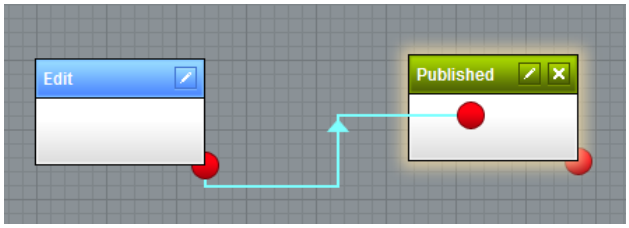
To enable documents to be moved from one step to another, you must connect the steps first. Each step has one or more source points, which you can drag to another step to create a connection.

Creating a connection between steps

1. Click a source point and hold the mouse button.

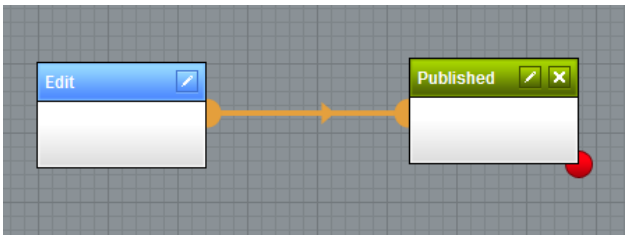
2. Drag the source point to the step you want to connect to.

The step will be highlighted with a yellow glow around it, as shown in the following image.



3. Release the mouse button.

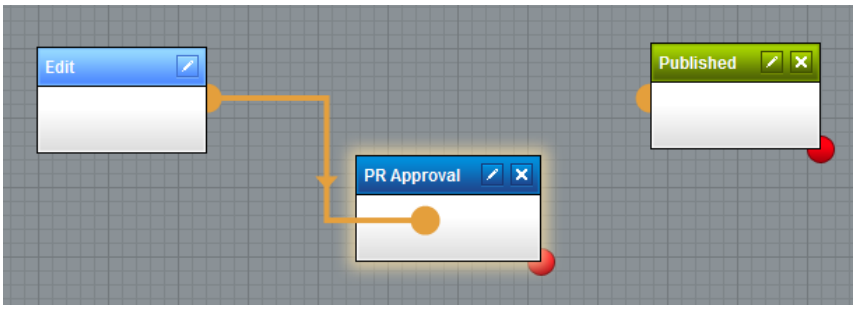
The resulting connection will look like the following image. After you create the connection, it stays selected, so you can immediately change it or delete it.



Changing a connection to a different step

1. Make sure you place the step that you want to change the connection to onto the grid.
2. Click the connection that you want to change and hold the mouse button.
3. Drag the connection to the step you want to change the connection to.

The target step highlights with a yellow glow around it.



4. Release the mouse button.

Deleting a connection

1. Click a connection to select it. The connection will change its color and increase thickness.
2. Click **Delete** in the toolbar or press Delete on the keyboard. A confirmation message appears.
3. Click **OK** to confirm that you want to delete the connection.

4.10.2.3.2.4 Defining if/else conditions

Advanced workflow allows you to define conditions based on which the workflow process can be split into branches.

Adding a condition

1. Drag and drop the **Condition** step onto the grid.
2. Edit the step.

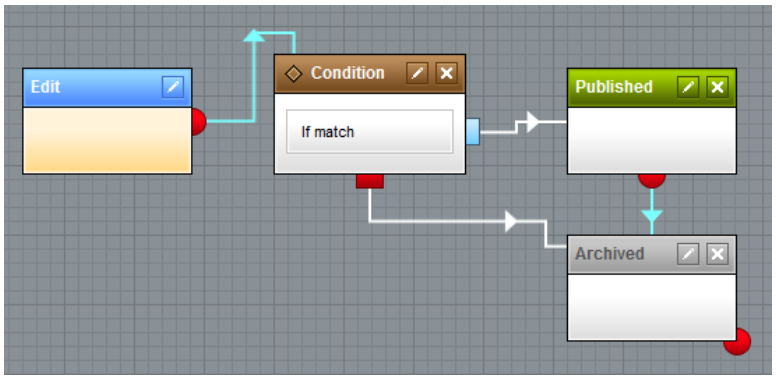
The **Workflow step properties** dialog opens.

3. Enter a macro condition into the **Condition** field. The condition must evaluate as true or false.

4. Click **Save** and close the dialog.

5. Connect a step that will precede this step and connect both source points with other steps.

The following image shows an example result.



4.10.2.3.2.5 Defining user decisions

The **User choice** step type allow editors to make decisions.

Adding a user choice

1. Drag and drop the **User choice** step from the toolbar onto the grid.

2. Connect a step that will precede this step.

3. Define choices.

4. Connect the choices with other steps.

Defining choices

1. Set the number of choices that you want users to have.

- Click **Add choice** (plus sign) to add a new choice.
- Click **Delete choice** (a cross next to the particular choice) to delete unneeded choices.

2. (Optional) Give names to the choices.

a. Double-click the choice name.
b. Type in a new name.
c. Press **Enter** to save the name.

3. (Optional) Configure additional parameters:

- User action button parameters

- Macro condition based on which particular choices will be offered to users
- Roles and users who can make particular choices

4.10.2.3.2.6 Defining automatic decisions

Advanced workflow provides two types of steps that split a workflow process based on multiple conditions - **multi-choice** and **first win**.

Adding a multi-choice step

The multi-choice step allows you to define a set of cases that will be automatically evaluated when a document reaches the step. If a matching case is found, the workflow process continues to the step connected with the matching case's source point. If more than one matching case is found, the system lets the editor choose from the matching cases. If no matching case is found, the process continues through the *else* branch.

1. Drag and drop the Multi-choice step from the toolbar onto the grid.

2. Connect a step to the entry point (green).

3. Connect the *else* source point (red) with another step.

4. Define cases.

5. Connect the cases' source points with other steps.

Adding a first win step

The first win step allows you to define a set of cases that will be automatically evaluated when a document reaches the step. If a matching case is found, the workflow process continues to the step connected with the matching case's source point. After the system finds the first matching case, it doesn't evaluate the other cases.

1. Drag and drop the First win step from the toolbar onto the grid.

2. Connect a step to the entry point (green).

3. Connect the *else* source point (red) with another step.

4. Define cases.

5. Connect the cases' source points with other steps.

Defining cases

1. Set the number of cases that you want to evaluate.

   - Click **Add new case** (plus sign) to add a new case.
   - Click **Delete case** (a cross next to the particular case) to delete an unneeded case.

2. Give names to the cases.

a. Double-click a case name.
b. Type in a new name.
c. Press **Enter** to save the name.

3. Define conditions that will be evaluated.

a. Click **Edit case properties** (pencil icon).
b. Enter the condition into the **Condition** field.

4. (Optional) Configure additional parameters:

- User action button parameters
- Roles and users who will be able to move a document through particular cases

4.10.2.3.2.7 Setting timeouts for automatic approving

Advanced workflow allows you to define a timeout, after which the system automatically moves a document in a certain step to the next step in the workflow process.

You have two options how to define a timeout:

- Setting a timeout in a step's properties
- Adding a Wait step into the process

When you set a timeout for a specific step, you can ensure that documents won't stay in the particular step longer than what is necessary. If nobody moves a document to another step before the time expires, the system moves it automatically. If you set a timeout for a step that creates multiple branches, you can choose the branch to continue in after the timeout expires. For all steps, you can also use a special timeout branch.

The Wait step acts as a stand-alone approval step, with one incoming and one outgoing connection. During the time a document is in this step, users can approve or reject it as they're used to. After the specified timeout expires, the workflow process automatically continues in the next step.

Setting a timeout in a step's properties

1. Edit (pencil icon) the step you want to set a timeout for.

2. Specify when you want to move to the next step on the **Timeout settings** panel. You have two options:

- Select **Specific interval** and specify a time interval. The time starts running when documents get approved to the step.
- Select **Specific day** and choose the date and optionally the time when you want documents to move to next step.

3. Select the step that you want the process to continue in after the timeout expires in the **Leave through** drop-down list. Note that besides the standard source points of the step you're editing, the drop-down list offers an extra timeout source point.

4. Click  **Save** and close the properties dialog.

5. If you selected the timeout source point in the **Leave through** drop-down list, connect another step to the source point.

Adding a wait step

1. Drag and drop the Wait step from the toolbar onto the grid.

2. Connect the step with a step that will precede it and with a step that will follow.

3. Open the step's properties by clicking the Edit (pencil icon) button.

4. Specify when you want to move to the next step. You have two options:

- Select **Specific interval** and specify a time interval. The time starts running when documents get approved to the step.
- Select **Specific day** and choose the date and optionally the time when you want documents to move to next step.

5. Click **Save**.

4.10.2.3.2.8 Adding action steps

This chapter describes how to enhance a workflow process with configurable automatic actions.

Using **action steps**, you can automate tasks that editors would have normally had to do manually after a document reached a certain step. For example, you might want to move a document to another section of the site the moment it reaches the Archived step. Advanced workflow makes this possible with the **Move document** action step.

The following is a list of all available action steps:

- **Copy document** - copies a specified document to another location.
- **Delete document** - deletes a specified document.
- **Import translation** - imports translations of the document that are ready.
- **Link document** - creates a linked document.
- **Move document** - moves a specified document to another location.
- **Publish to Facebook** - automatically sends a predefined post on Facebook.
- **Publish to Twitter** - automatically sends a predefined post on Twitter.
- **Send for translation** - submits a document for translation using the specified translation service.
- **Send e-mails** - sends e-mails based on a specified e-mail template.
- **Send notification e-mails** - sends e-mails according to the e-mail settings defined for the step that immediately follows.
- **Set document property** - assigns a specified value to a document's property.
- **Synchronize document** - synchronizes a specified document to all enabled staging servers on the site the document is on.

For information how to add and configure a particular action step, refer to its respective section of this topic.

Manipulating documents within a workflow process

Copying a document

1. Drag and drop the **Copy document** step onto the grid.

2. Connect the step with a preceding step and optionally with a following step.

3. Edit the step (click the pencil icon).

4. (Optional) Specify the document that you want to copy using the following properties:

- **Source site** - site that the document should be copied from. If you don't specify a site, the current site will be used.
- **Source path** - path to the document that should be copied. If you don't specify a document, the current document (i.e., the one that reaches the step) will be used.

5. (Optional) Adjust the following properties:

- **Include child documents** - copies also all child documents of the copied document.
- **Copy permissions** - prevents resetting permissions for the document.

6. Specify the document under which you want to make the copy using the following properties:

- (Optional) **Target site** - site that contains the parent document for the copy. If you don't specify a site, the current site will be used.
- **Target path** - path to the parent document for the copy.

7. Click **Save**.

Moving a document

1. Drag and drop the **Move document** step onto the grid.

2. Connect the step with a preceding step and optionally with a following step.

3. Edit the step (click the pencil icon).

4. (Optional) Specify the document that you want to move using the following properties:

- **Source site** - site that the document should be moved from. If you don't specify a site, the current site will be used.
- **Source path** - path to the document that should be moved. If you don't specify a document, the current document (i.e., the one that reaches the step) will be used.

5. (Optional) Adjust the following properties:

- **Only child documents** - makes the action step move only the child documents of the specified document and not the document itself.
- **Preserve permissions** - moves the document without resetting its permissions.

6. Specify the document under which you want to move the document:

- (Optional) **Target site** - site that contains the parent document for the move. If you don't specify a

site, the current site will be used.
- **Target path** - path to the parent document for the move.

7. Click **Save**.

Creating a linked document

1. Drag and drop the **Link document** step onto the grid.

2. Connect the step with a preceding step and optionally with a following step.

3. Edit the step (click the pencil icon).

4. (Optional) Specify the document that you want to link using the following properties:

- **Source site** - site that the document should be linked from. If you don't specify a site, the current site will be used.
- **Source path** - path to the document that you want to link. If you don't specify a document, the current document (i.e., the one that reaches the step) will be used.

5. (Optional) Adjust the following properties:

- **Include child documents** - creates also links to the child documents of the document.
- **Copy permissions** - copies the permissions for the document to the linked document.

6. Specify the document under which you want to create the link:

- (Optional) **Target site** - site that contains the parent document for the link. If you don't specify a site, the current site will be used.
- **Target path** - path to the parent document for the link.

7. Click **Save**.

Deleting a document

1. Drag and drop the **Delete document** step onto the grid.

2. Connect the step with a preceding step and optionally with a following step.

3. Edit the step (click the pencil icon).

4. Specify the document that you want to delete using the following properties:

- **Source site** - site that the document should be copied from. If you don't specify a site, the current site will be used.
- **Source path** - path to the document that should be copied. If you don't specify a document, the current document (i.e., the one that reaches the step) will be used.

5. (Optional) Adjust the following deletion properties:

- **Destroy document** - deletes the document without a chance to restore it from the Recycle bin.
- **Delete all culture versions** - if the document is translated to more than one culture, the property

makes sure that all the culture versions will be deleted.

6. (Optional) Specify what to do with SKUs (stock keeping units, also known as products) assigned to the deleted documents in the **Assigned SKUs** section.

7. (Optional) Define a replacement document for the deleted document. The replacement document will be served when visitors request the deleted document's URL. You can also adjust the following related settings:

- **Copy all paths** - the replacement document will be accessible via all the deleted document's URLs instead of only the default one.
- **Include child documents** - the replacement document will be applied also to child documents of the deleted documents.

8. Click **Save**.

Adjusting document properties

You can use the **Set document property** action step to modify properties of a document, including document type specific fields.

1. Drag and drop the **Set document property** step onto the grid.

2. Connect the step with a preceding step and optionally with a following step.

3. Edit the step (click the pencil icon).

4. Specify the document that you want to modify using the following attributes:

- **Source site** - site that the document is on. If you don't specify a site, the current site will be used.
- **Source path** - path to the document that you want modified. If you don't specify a document, the current document (i.e., the one that reaches the step) will be used.

5. Enter the name of the property that you want to modify. To learn what properties documents have, consult the **Kentico CMS API reference**. The properties are listed in tables *CMS_Tree* and *CMS_Document*.

6. Enter a new value for the property. Make sure the type of the data you entered matches the data type of the respective database column.

7. Click **Save**.

Synchronizing a document to staging servers

You can use the **Synchronize document** action step to synchronize changes in a specified document to other servers using the Staging module.

1. Drag and drop the **Synchronize document** step onto the grid.

2. Connect the step with a preceding step and optionally with a following step.

3. Edit the step (click the pencil icon).

4. Specify the document that you want to synchronize using the following properties:

- **Source site** - site that the document should be synchronized from. If you don't specify a site, the current site will be used.
- **Source path** - path to the document that should be synchronized. If you don't specify a document, the current document (i.e., the one that reaches the step) will be used.

5. (Optional) Adjust the following properties:

- **Include child documents** - synchronizes also all child documents of the synchronized document.
- **Log update tasks** - forces synchronization of documents that don't have any tasks generated for them. If you turn this setting off, the system will check whether there are any staging tasks generated for the document and synchronize them.

6. Click **Save**.

Configuring workflow to automatically post on Facebook and Twitter

1. Configure the settings in **Site Manager -> Settings -> Social networks -> Facebook/Twitter**. Please use the corresponding context help for details about how to acquire the required keys (all fields in the **General** section of the settings must be filled in).

2. Drag and drop the **Publish to Facebook** or **Publish to Twitter** step onto the grid.

3. Connect the step with a preceding and following step (the best position is before the Publish step).

4. Edit the workflow step (click the pencil icon).

5. Enter the **Text** of the message which will be automatically posted on Facebook/Twitter when going to the next step. You can click the provided button to open an editor with a macro selection control to define a message using macros.
   - For example, *Check out our new blog post at www.mydomain.com{% GetDocumentURL() %}*.

6. Click **Save**.

7. Edit the previous step and set a **User action text** field to a suitable button text, for example, *Publish to Facebook*. The default button text, which will be visible to content editors, is **Submit for approval**.

8. Click **Save**.

Configuring workflow to automatically send e-mails

Advanced workflow offers two ways of incorporating sending of e-mails into a workflow process:

- **Send e-mails action step** - sends an e-mail to specified recipients. You can choose an e-mail template or write the text of the e-mail from scratch.
- **Send notification e-mails action step** - sends e-mails associated with the next step to the next step's workflow operators. You can use this step in combination with the Wait step or timeout settings to notify users that a document is waiting for their approval for a long time.

Adding a Send e-mails step

1. Drag and drop the **Send e-mails** step onto the grid.

2. Connect the step with a preceding and a following step.

3. Edit the step (click the pencil icon).

4. (Optional) Enter the sender's e-mail address. If you don't enter any address, the value from *Settings -> Content -> Content management -> Send workflow e-mails from* will be used.

5. Enter the recipients of the e-mail into the **To** field. Separate multiple recipients by semicolons.

6. Specify the text of the e-mail. You have two options:

- **Select an e-mail template** or create a new one.
- Select **HTML formatted text** in the **Based on** field and write the text into the text area that appears.

7. Click **Save**.

Adding a Send notification e-mails step

1. Drag and drop the **Send notification e-mails** step onto the grid.

2. Connect the step with a step that will precede. This should typically be a Wait step or a step with a defined timeout.

3. Connect the step with a step that will follow. This is the step that the notification steps will be sent for.

Managing document translations within workflow

You can leverage the following two action steps to facilitate translation of documents within a workflow process:

- **Send for translation** - submits the current document for translation using the specified translation service.
- **Import translation** - if the current document's translation is ready, imports the translated data into the respective culture version of the document.

You can find more information about translation services in the Translation services chapter.

Sending a document for translation

1. Drag and drop the **Send for translation** step onto the grid.

2. Connect the step with a preceding and optionally a following step.

3. Edit the step (click the pencil icon).

4. Set the following attributes:

- Translation service to use
- Target language
- Translate document attributes
- Priority
- Translation deadline
- Instructions for translators

5. Click **Save**.

Importing translation

1. Drag and drop the **Import translation** step onto the grid.

2. Connect the step with a preceding and optionally a following step.

4.10.2.3.2.9 Creating custom action steps

Kentico CMS enables you to write your own actions that you will be able to incorporate into a workflow process. The procedure consists of two steps - writing the code of the action and registering the action into the system.

Writing actions

1. Create a new class and make it inherit from *CMS.DocumentEngine.DocumentWorkflowAction*.

2. Override the *Execute()* method from the base class.

3. Write the code that you want to be executed with the action into the *Execute* method's body.

Action steps can have parameters (settings) specified that you can use to modify their behavior. You can then enter the parameter's values when configuring an action step in the workflow designer. If you want your custom action step to rely on parameters, you can specify them on the **Actions** tab in **Site Manager -> Development -> Workflows**. To access the parameter's values in the code of an action, use the *GetResolvedParameter* method.

To access the data of the document that will go through the action step, use the *Node* object and its properties.

```
using System;
using CMS.DocumentEngine;
using CMS.CMSHelper;

public class CustomAction : DocumentWorkflowAction
{
    public override void Execute()
    {
        // Get the parameter value or assign a default value - an empty string
        string appendText = GetResolvedParameter("MyParameter", "");

        // Append the parameter value to the name of the document
```

```
        Node.DocumentName += appendText;

        // Save the changes into the database
        TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);
        DocumentHelper.UpdateDocument(Node, tree);

    }
}
```

Registering actions in the system

1. Navigate to **Site Manager -> Workflows -> Actions** and click **New action**.

2. Fill in the following mandatory fields:

- **Display name** - name of the action step.
- **Assembly name** - name of the library that contains the action's code.
- **Class name** - full name of the class that contains the action's code.

3. (Optional) Select the module that the action will belong to.

4. (Optional) Define the action's parameters on the **Parameters** tab.

5. If you created the class in the App_Code folder (or Old_App_Code if you're using web application), you must register it in code to make sure the system loads it.

4.10.3 Defining macro rules

To speed up and simplify the process of defining conditions in workflows, you can prepare a set of macro rules. For information on how to create and manage Macro rules, refer to the Macro conditions chapter.

4.10.4 Configuring workflow operators

This topic shows how to allow or deny site editors to approve and reject documents in certain steps, or make certain choices.

Allowing users to approve or reject documents

Kentico CMS allows you to assign different users and roles to different steps, which in turn allows the specified users and roles to approve or reject documents in a particular step.

The only steps to which you cannot assign different users and roles are the default:
- **Edit step** - any user who is editing the document can move the document to the next step.
- **Published and Archived steps** - only users with the Manage workflow permission under the Content module, and Global administrators can move the document from these steps.

This procedure applies to both basic and advanced workflow.

To allow users to approve or reject a document in a certain step, follow these instructions:

1. Edit a step.

- If you're using basic workflow, click **Edit** () next to the step.
- If you're using advanced workflow, click **Edit** (pencil icon) in the step's header.

2. Switch to the **Security** tab.

3. Select which roles you want to be allowed to approve or reject the step. You have the following options:

- **None**
- **Only listed** - if you select this option, click **Add roles** to choose roles you want to add. If you don't add any roles, the result will be the same is if you select the **None** option.
- **All except listed** - if you select this option, click **Add roles** to choose roles you want to exclude. All other roles will be allowed to approve or reject the step.

4. Choose if you want to include or exclude individual users from the list of allowed operators.

- **No extra users**
- **Include the following users** - if you select this option, click **Add users** to choose users you want to add to the list of allowed users. If you don't add any users, the result will be the same is if you select the **No extra users** option.
- **Exclude the following users** - if you select this option, click **Add users** to choose users you don't want to be allowed to approve or reject documents in this step.

Allowing users to make a choice

When using workflow steps that allow multiple choices, you can allow or deny roles and users to send a document through certain branches of the workflow process. This feature is only available in advanced workflows, as basic workflows don't support multi-choice steps.

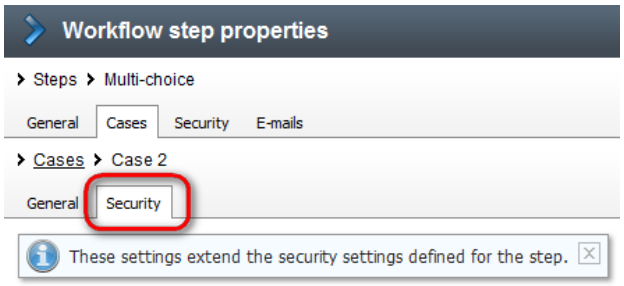
You can allow or deny making choices in the properties of the individual choices. By default, choices inherit security settings form the step they belong to. If you adjust security settings for a choice, these settings will be added to the settings defined for the step.

When determining whether the current user can make a choice, the system first checks security settings for the step. If the user is allowed to approve documents in the step, the system checks whether the user is allowed to make choices. If a user is not allowed to approve documents in the step, the user will not be able to make any choices that belong to the step, regardless of the choices' settings.

To allow or deny users to make choices, follow these steps:

1. Edit the choice by clicking the **Edit** (pencil icon) button next to it.

2. Switch to the Security tab.



3. Define roles, which can approve (select) the choice. You have the following options:

- **Use step settings**
- **Only listed** - when selected, you can add roles that you want the choice to be allowed for using the **Add roles** button.
- **All except listed** - when selected, all users in roles on all sites will be allowed the choice, except for users in those roles, which you select using the **Add roles** button.

4. Define additional users, who can approve (select) the choice. You have the following options:

- **Use step settings**
- **Include the following users** - if you select this option, click **Add users** to choose users you want to add to the list of allowed users. If you don't add any users, the result will be the same is if you select the **No extra users** option.
- **Exclude the following users** - if you select this option, click **Add users** to choose users you don't want to be allowed to approve or reject documents in this step.

4.10.5 Configuring e-mail notifications

When a document reaches a workflow step, the system can send out notification e-mail messages to those involved in the workflow process. This topic describes how to enable e-mail notifications globally, for a workflow or for a workflow step. The topic applies to both basic and advanced workflow.

The e-mails are based on e-mail templates. Kentico CMS comes with several default templates, which you can use without any additional configuration. If you want to create your own e-mail templates for workflow notifications, follow the instructions in the Writing workflow e-mail templates section.

The system sends notification e-mails according to the following rules:

- When a document reaches a step (except Edit, Published and Archived), the **Waiting for approval** e-mail is sent to users allowed to approve or reject the current (new) step.
- When someone approves a document, i.e., moves it from an approval step to the next step, the **Approved** e-mail is sent to the user who submitted the document for approval.
- When someone rejects a document, i.e, moves it to the previous step, the **Rejected** e-mail is sent to the user who submitted the document for approval.
- E-mails are never sent to users who performed the action that invoked the sending.

Enabling workflow e-mail notifications globally

1. Open **Site Manager** and navigate to **Settings -> Content -> Content management**.

2. Turn the **Send workflow e-mails** setting on.

4. Type an address into the **Send workflow e-mails from** setting. This address will appear to recipients as the sender of the e-mails.

5. Click **Save**.

Configuring e-mail notifications for a workflow

1. Edit () the workflow and switch to the **E-mails** tab.

2. Set the **Send notification e-mails** setting to **Yes**. Alternatively, you can **Use site settings**. See Enabling workflow e-mail notifications globally.

3. Check the box next to events that you want users to be notified about:

- **Send waiting for approval** - sends an e-mail when a document reaches an approval step (i.e. all steps except Edit, Published and Archived).
- **Send approved** - sends an e-mail when a document leaves an approval step in the direction towards the end of the workflow process.
- **Send rejected** - sends an e-mail when a document leaves a step in the direction towards the beginning of the workflow process.
- **Send published** - sends an e-mail when a document reaches a Published step.
- **Send archived** - sends an e-mail when a document reaches an Archived step.

4. (Optional) Select an e-mail template for events for which you don't want to use the default template.

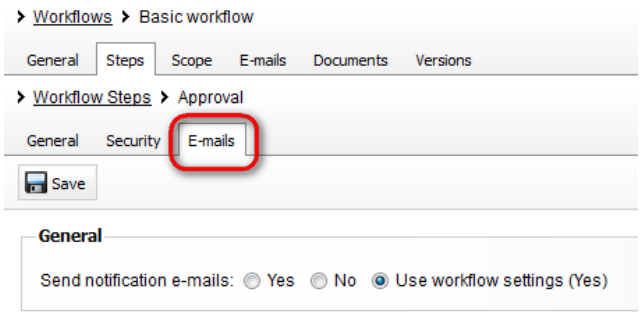
- To bring up the list of available templates, click **Select**.
- To edit the currently selected template, click **Edit**.
- To create a new template, click **New**.
- To clear your template selection and revert back to the default template, click **Clear**.

5. (Optional) Specify whether you want to notify users who are not involved in the process. This notification will be sent every time a document reaches a step that needs to be manually approved.
   a. Select users using the **Add users** button.
   b. (Optional) Select an e-mail template. Otherwise, the default template will be used.

6. Click **Save**.

Configuring e-mail notifications for a particular workflow step

1. Edit the step and switch to the **E-mails** tab.



2. Set the **Send notification e-mails** setting to **Yes**. Alternatively, you can **Use workflow settings**. See Configuring e-mail notifications for a workflow.

3. Choose events you want users to be notified about.

- **Send waiting for approval** - sends an e-mail when a document reaches the step.
- **Send approved** - sends an e-mail when a document leaves the step in the direction towards the end of the workflow process.
- **Send rejected** - sends an e-mail when a document leaves the step in the direction towards the beginning of the workflow process.

4. (Optional) Select an e-mail template for events for which you don't want to use the default template.

- To bring up the list of available templates, click **Select**.
- To edit the currently selected template, click **Edit**.
- To create a new template, click **New**.
- To clear your template selection and revert back to the default template, click **Clear**.

5. Click **Save**.

Writing workflow e-mail templates

1. Navigate to **Site Manager -> Administration -> E-mail templates**.

2. Select a site you want the template to be available on and click **New template**.

3. Enter a name for the new template into the **Display name** field.

4. Choose **Workflow** from the **E-mail type** drop-down list. This ensures that you will have relevant macros available to insert into the template's text.

5. (Optional) Specify a sender, a recipient, copy and blind copy recipients and a subject (**From**, **Cc**, **Bcc** and **Subject** fields).

6. Write the text of the template. An e-mail template can have two formats - HTML and plain text. You should write the template text at least in the format that you use on you websites. By default, the system sends HTML e-mails. You can find out what your settings are in **Site Manager -> Settings -> System -> E-mails -> E-mail format**.

You can insert context-specific information into the template's text, such as the current step name, the link to edit the document, etc., by using macros. See the Macros in workflow e-mail templates section for a list of available macros.

7. Click **Save**.

Macros in workflow e-mail templates

In e-mail templates of the **Workflow** type, you can use macros that hold information about the workflow and about the document that invoked sending the e-mail. The following list presents workflow-related macros that you can use:

- **ActionDefinition** - object that holds the parameters of the current step, if the current step is an action step.
- **ApplicationURL** - URL address of the application.
- **Comment** - comment that users can add when approving or rejecting documents.
- **CurrentStep** - object that holds data of the step the document is currently in.
- **CurrentUser** - object that holds data of the currently logged in user.
- **Document** - object that holds data of the document.
- **DocumentPreviewUrl** - address where anyone can view the latest version of the document as it would appear on the live site.
- **DocumentEditUrl** - address pointing to the page in CMS Desk, where editors can modify the document.
- **DocumentActionName** - name of the action that was performed with the document. The macro has four possible values.
  - Approve
  - Archive
  - Publish
  - Reject
- **OriginalStep** - object that holds data of the step the document was in before it was moved to the current step.
- **Workflow** - object that holds data of the workflow process.

Besides the listed macros, you can also use any other macro expression, as described in the Macro expressions chapter.

4.10.6 Using workflows

This topic describes how you can manage documents under a workflow.

Please note

The following tasks refer to buttons in CMS Desk by their default captions, i.e., Submit for approval, Approve, Reject. However, the caption texts are customizable, so you may not always see the same buttons in the user interface.

Submitting a document for approval

When a document is in the Edit step, you can submit it for approval to move it to the next step.

1. In **CMS Desk -> Content**, navigate to the document that you want to submit for approval.

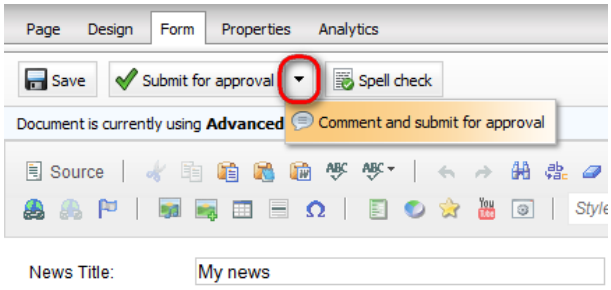
2. Click **Submit for approval**. The document will be moved to the next step in the workflow process.

Submitting a document for approval with a comment.

To submit a document for approval and add a comment for the person who will be approving it, follow these steps:

1. In **CMS Desk -> Content**, navigate to the document that you want to submit for approval.

2. Click the arrow icon next to the **Submit for approval** button and then click **Comment and submit for approval**. A dialog box will pop up.



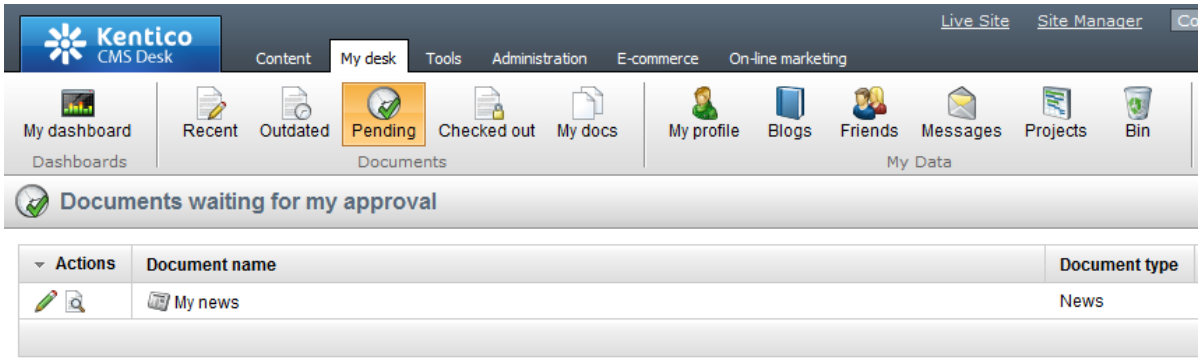
3. Enter your comment and click **Approve**. The document will be moved to the next step in the workflow process.

Viewing documents waiting for your approval

To view documents that are in an approval step, which you're an operator of, you can use the **Pending** page in **My desk**.

1. In **CMS Desk**, switch to **My desk**.

2. Switch to the **Pending** page.



3. (Optional) Click **Navigate to document** to view the current version of the document on the live site.

4. (Optional) Click **Edit** to make changes to the document and approve or reject it.

Approving and rejecting a document

When a document reaches a step that requires user interaction, you can approve the document, or, if settings of the step permit, reject it.

1. Navigate to the document you want to approve or reject.

2. (Optional) Make changes to the document.

3. (Optional) Disable the **Send notification e-mails** checkbox to prevent the system from sending e-mails to users involved in the next step.

4. Approve or reject the document. Note that if the step that follows is the **Published** step, the default text of the **Approve** button changes to **Publish**.

- To approve the document without commenting on the approval, click **Approve**.
- To reject the document without commenting on the rejection, click **Reject**.
- To approve the document and make a comment, click the arrow icon next to the **Approve** button, then click **Comment and approve**. Enter your comment into the dialog box and click **Approve**.
- To reject the document and make a comment, click the arrow icon next to the **Reject** button, then click **Comment and reject**. Enter your comment into the dialog box and click **Reject**.

Publishing a document

When a document is published, its most recent version will be made available to visitors on the live site. You can publish documents in one of the following ways:

- Manually moving through the whole workflow process
- Directly publishing if the current step allows it
- Publishing multiple document at once

Manually moving through the whole workflow process

1. Navigate to the document that you want to publish.

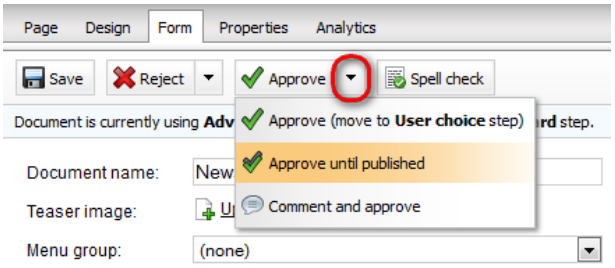
2. Repeatedly **approve** the document until it is in the **Published** step.

Directly publishing a document

When a document is in a standard approval step and the settings of the step allow it, you can publish the document immediately. To do this, a direct approval path must exist in the workflow process. Direct publishing won't work if your workflow process contains user decisions.

1. Navigate to the document that you want to publish.

2. Click the arrow icon next to the **Approve** button, then click **Approve until published**.



Publishing multiple documents at once

You can publish entire sections of your website using the **List** mode.

1. Click **List** in the **View mode** section of the toolbar in **CMS Desk**.

2. Navigate to the document, which is a parent of the documents that you want to publish.

3. Select the documents you want to publish. You have two options:

- Check the boxes next to documents you want to publish.
- Select All documents in the drop-down list below the document list.

4. Select **Publish** in the drop-down list marked *(select an action)* and click **OK** to confirm. You will be redirected to a page with the list of documents to be published.

5. (Optional) Adjust the following settings:

- **Publish also all child documents**
- **Perform Undo check-out for checked out documents** - if checked, all documents that are checked out will be checked in and the changes made during the last check-out will be discarded.

6. Click **Yes** to start publishing the documents.

Archiving a document

When you archive a document, it stays in its location in the content tree, but disappears from the live site. You can archive documents in one of the following ways:

- Manually archiving
- Archiving multiple documents at once

Manually archiving a document

When a document reaches a step that is immediately followed by the Archived step, you can click the **Archive** button to archive the document.

Archiving multiple documents at once

You can archive entire sections of your website using the **List** mode.

1. Click **List** in the **View mode** section of the toolbar in **CMS Desk**.

2. Navigate to the document, which is a parent of the documents that you want to archive.

3. Select the documents you want to archive. You have two options:

- Check the boxes next to documents you want to archive.
- Select All documents in the drop-down list below the document list.

4. Select **Archive** in the drop-down list marked *(select an action)* and click **OK** to confirm. You will be redirected to a page with the list of documents to be archived.

5. (Optional) Adjust the following settings:

- **Archive also all child documents**
- **Perform Undo check-out for checked out documents** - if checked, all documents that are checked out will be checked in and the changes made during the last check-out will be discarded.

6. Click **Yes** to start archiving the documents.

Viewing a document's workflow history

For all documents under a workflow, the system records every move from one step to another. That means that you can review all the workflow actions that were performed on the document, the users who performed them and comments.

To view the workflow history of a document, select it in the content tree and navigate to **Properties -> Workflow**.

Restarting a workflow cycle

When a document reaches the Published or Archived step (whichever comes first), it completes its workflow cycle. That means that the document has gone all the way from the Edit step through the whole process to the final step. To start a new cycle and put the document back to the edit step, do the following:

- If you're using basic workflow, open the document in CMS Desk, make changes to it and click **Save**. This will create a new version of the document.
- If you're using advanced workflow, click **New version**.

When you start a new cycle, the system re-evaluates workflow scopes that cover this document and chooses the workflow with the highest priority.

Removing documents from a workflow

If you want to remove documents from an existing workflow. That is, documents that are under a workflow scope, and that are in any of the steps belonging to the workflow, then proceed as follows:

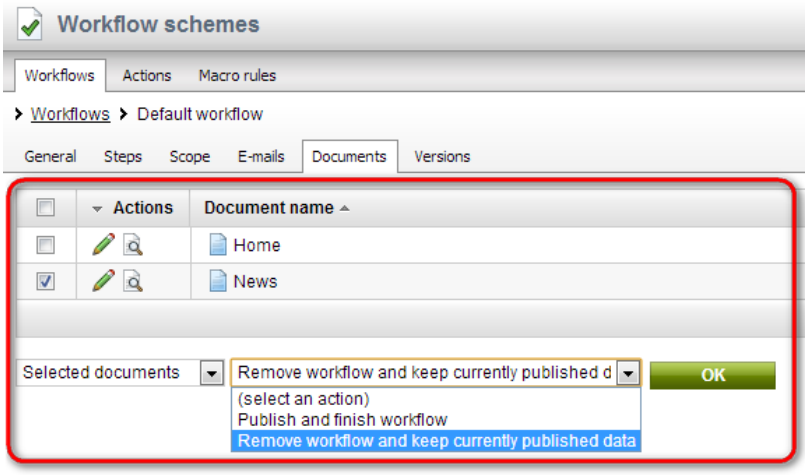
1. Navigate to **Site manager -> Development -> Workflows**.

2. **Edit ()** the workflow from which you want to remove the documents.

3. Switch to the **Documents** tab.

4. Select the documents that you want to remove from the workflow.

5. Choose the **Remove workflow and keep currently published data** option in the second drop-down list under the list of documents.



6. Click **OK**.

Note that this action does not remove the document from under the workflow scope. It may become part of the workflow again when modified.

4.10.7 Versioning

The versioning functionality in workflows stores document versions as documents are edited. Versioning enables viewing and comparing document versions and rolling documents back to previous versions.

Versioning is closely tied to the workflow module. To use versioning on documents, you need to apply a workflow on them. If you don't want documents to go through any workflow process and use versioning at the same time, refer to the Versioning without workflow topic.

Version numbering

Every document version is identified by a version number. By default, the system assigns numbers to versions automatically according to the following rules:

- When you create a document, it gets version number **0.1**.
- When you save a document, a minor version is created and the number after the period increases. Example: **0.2**, **0.3**, etc.
- When you publish a document, the latest minor version gets promoted to a major version. The number before the period increases and the number after the period resets to zero. Example: **1.0**, **2.0**, etc.

If you use content locking, you can disable the automatic numbering in **Settings -> Content -> Content management**, by turning off the **Use automatic version numbering** setting.

Viewing and comparing document versions

1. Navigate to the document that you want to view versions of.

2. Switch to **Properties -> Versions**.

3. Click **View** next to the version you want to view. The **Document version** window opens.

4. Select a version for comparison using the **Compare to** drop-down list. When comparing document versions, the older version appears on the left.

Rolling back versions

1. Navigate to the document that you want to view versions of and switch to **Properties -> Versions**.

2. Click **Roll back** next to a version. The system will make a copy of the version and mark the copy as the latest version.

4.10.7.1 Content locking

Content locking allows content editors to lock a document for editing, so that other editors cannot modify the document at the same time.

Content locking is based on the check-in/check-out principle. The following is an example of operations in the document life cycle when content locking is used:

1. A user creates a new document and saves it. The system automatically checks it out (locks it), so that other users cannot edit it.
2. The user finishes the changes, checks in the document and submits it for approval.
3. A reviewer checks out the document and makes changes, then checks it in and approves it.

Enabling content locking

You can enable content locking for versioned documents at two levels:

- Globally
- For a particular workflow

To enable content locking **globally** for all sites or a particular site, navigate to **Site Manager -> Settings -> Content -> Content management** and turn on the **Use check-in/check-out** setting.

To enable content locking **for a particular workflow**, edit it and configure the **Use check-in/check-out** property.

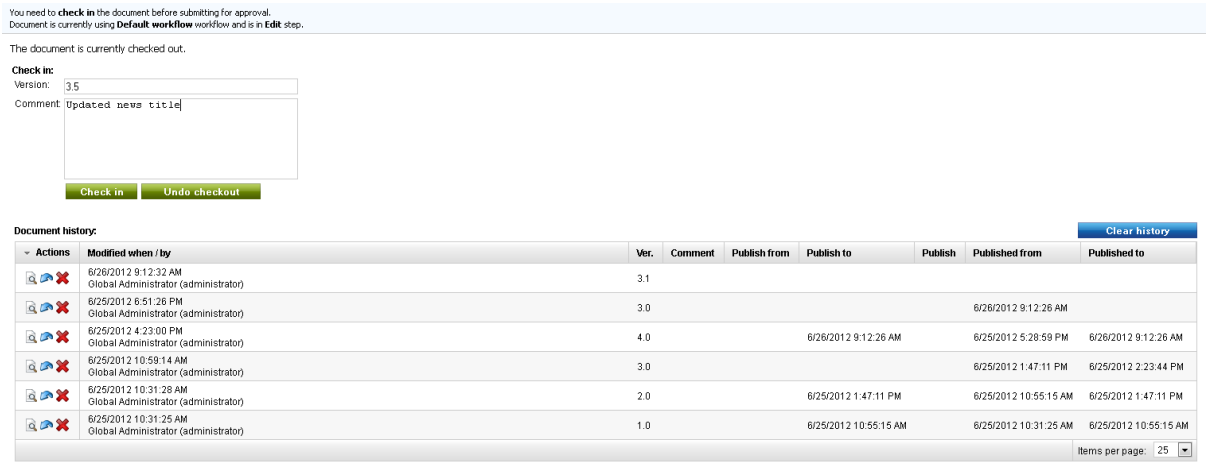
Checking documents in and out

To check out documents when editing them, use the **Check out** button. When you check out a document, you can then check it in with the **Check in** button.

To discard the changes you made while the document was checked out, click **Undo check-out**. The document will revert back to the latest version before the check-out.

Changing version numbers and commenting on versions

You can use the **Check in/Check out/Undo check-out** buttons on the **Properties -> Versions** page. This page enables you to specify custom version numbers and comments for each version when you check in the document.



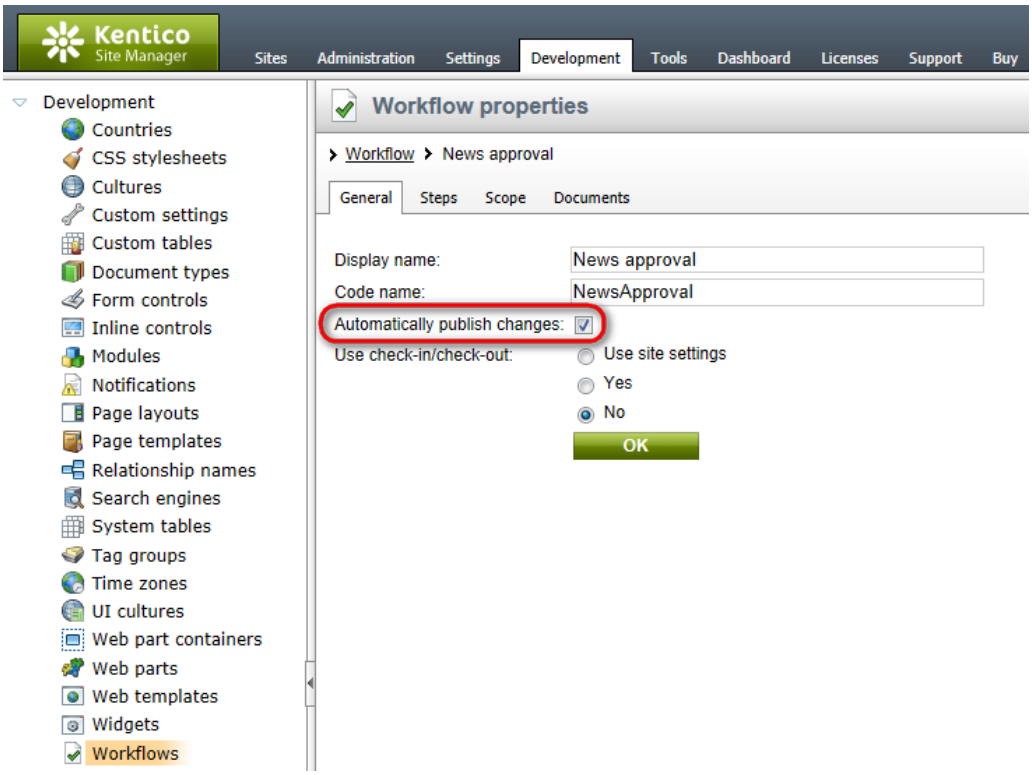
Checking in any document

Users to whom the **Check in any document** permission was granted can check in any document, even if they haven't checked out the document. This permission can be set in **Site Manager -> Administration -> Permissions**, in the permission matrix under **Modules -> CMS Content**.
However, this check-in can only be performed from the **Properties -> Versions** tab of the selected document, by clicking either the **Check in** or the **Undo check-out** button.

4.10.7.2 Versioning without workflow

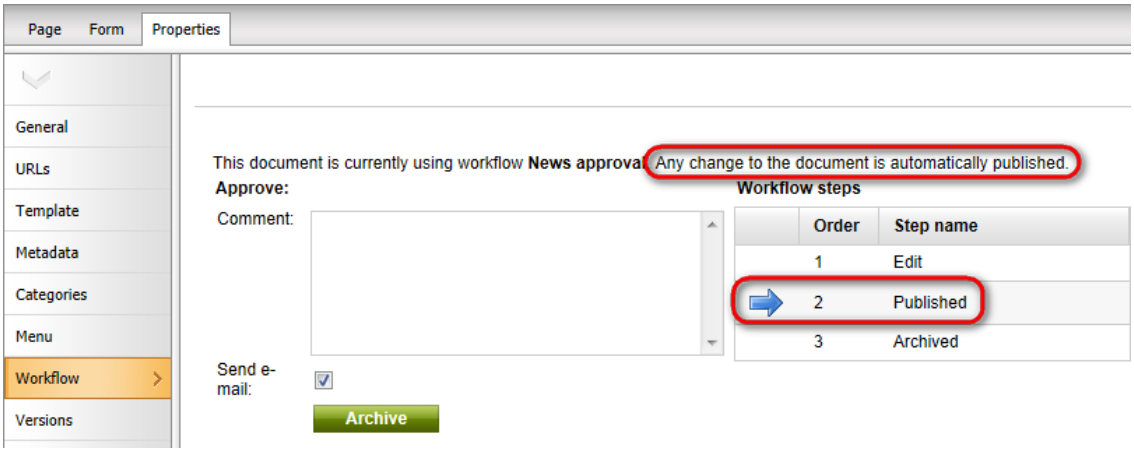
It is possible to use versioning without actually using any workflow steps. In other words, it is possible to have a new document version created whenever a document is modified and saved.

To enable versioning without workflow, you only need to enable the **Automatically publish changes** option either when creating a new workflow, or when editing () a workflow on its **General** tab. With this option enabled, custom steps can't be created on the **Steps** tab. You only need to define appropriate scope(s) so that the workflow affects the required documents.



From the end-user point of view, editing a document using versioning without workflow is the same as if the document was using no workflow at all. The ✔ **Submit to approval**, ✔ **Approve** and ✖ **Reject** buttons are not displayed on the document editing form - there is just the standard **Save** button present.

On the **Properties -> Workflow** tab, you can recognize a document using versioning without workflow by the *"Any change to the document is automatically published."* sentence, as highlighted below. You may also notice that the document remains in the **Published** step all the time until it gets archived.



4.10.8 Workflows internals and API

4.10.8.1 Overview

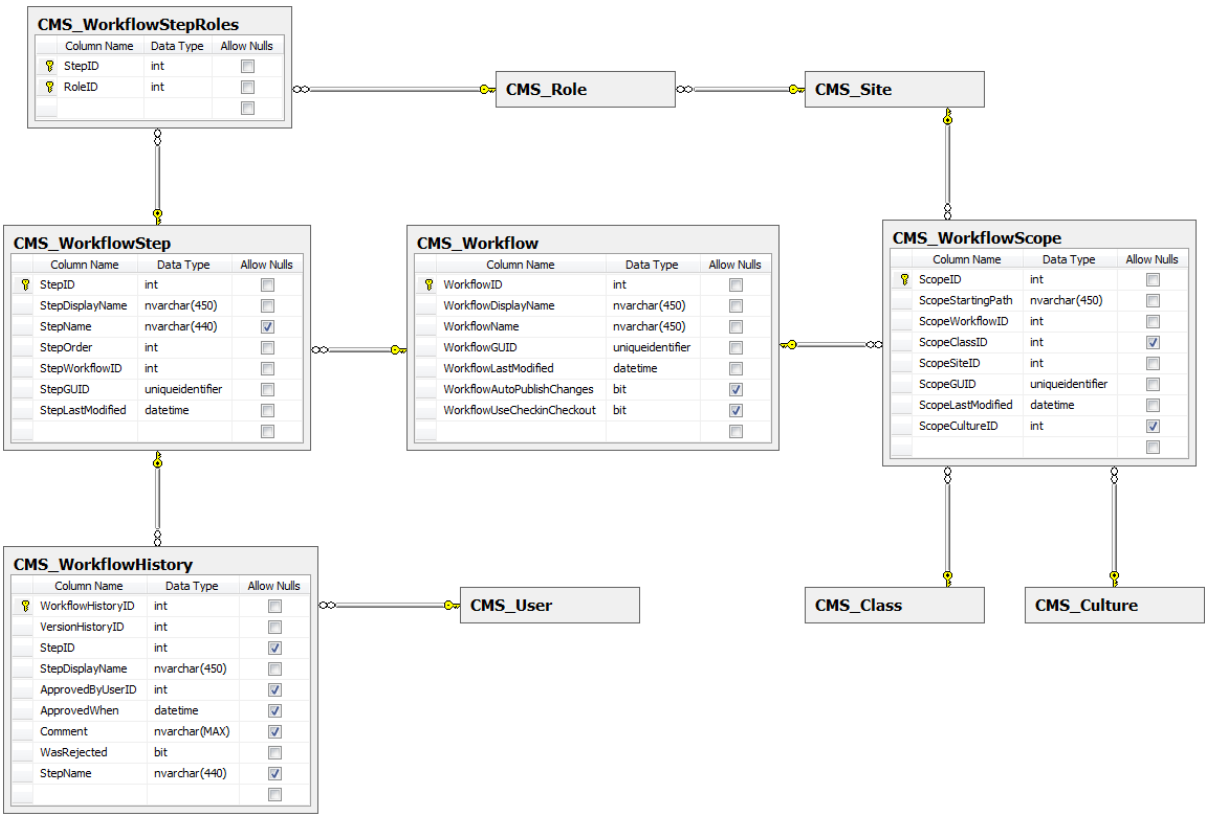
In this chapter, you will learn which database tables and API classes are used by workflow and versioning. You will also see the most common API examples.

Advanced API examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all methods from the module classes, please refer to Kentico CMS API Reference.

4.10.8.2 Database tables

The following database tables are used by workflows:

- **CMS_Workflow** - contains records representing defined workflows.
- **CMS_WorkflowStep** - contains records representing defined workflow steps of all defined workflows.
- **CMS_WorkflowScope** - contains records representing defined workflow scopes.
- **CMS_WorkflowStepRoles** - contains relationships between defined workflow steps and roles whose members can approve documents in these steps.
- **CMS_WorkflowHistory** - contains workflow history records, i.e. the information stored when a document is moved to a different workflow step.



4.10.8.3 API classes

If you are not familiar with database table data management through Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

The classes used by workflows can be found in the **CMS.WorkflowEngine** namespace.

CMS_Workflow table API:

- **WorkflowInfo** - represents one workflow object.
- **WorkflowInfoProvider** - provides management functionality for workflows.

CMS_WorkflowStep table API:

- **WorkflowStepInfo** - represents one workflow step object.
- **WorkflowStepInfoProvider** - provides management functionality for workflow steps.

CMS_WorkflowScope table API:

- **WorkflowScopeInfo** - represents one workflow scope object.
- **WorkflowScopeInfoProvider** - provides management functionality for workflow scopes.

4.10.8.4 API examples

4.10.8.4.1 Overview

These topics show examples of how the API for workflows can be used:

- Managing workflows
- Managing workflow steps
- Managing workflow scopes

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Development\Workflows\Default.aspx.cs***.

The workflows API examples use the following namespaces:

```
using System;
```

```
using System.Data;

using CMS.GlobalHelper;
using CMS.UIControls;
using CMS.CMSHelper;
using CMS.SiteProvider;
using CMS.WorkflowEngine;
using CMS.SettingsProvider;
```

4.10.8.4.2 Managing workflows

The following example creates a workflow.

```
private bool CreateWorkflow()
{
    // Create new workflow object
    WorkflowInfo newWorkflow = new WorkflowInfo();

    // Set the properties
    newWorkflow.WorkflowDisplayName = "My new workflow";
    newWorkflow.WorkflowName = "MyNewWorkflow";

    // Save the workflow
    WorkflowInfoProvider.SetWorkflowInfo(newWorkflow);

    // Create the three default workflow steps
    WorkflowStepInfoProvider.CreateDefaultWorkflowSteps(newWorkflow.WorkflowID);

    return true;
}
```

The following example gets and updates the workflow created by the code example above.

```
private bool GetAndUpdateWorkflow()
{
    // Get the workflow
    WorkflowInfo updateWorkflow = WorkflowInfoProvider.GetWorkflowInfo
("MyNewWorkflow");
    if (updateWorkflow != null)
    {
        // Update the properties
        updateWorkflow.WorkflowDisplayName =
updateWorkflow.WorkflowDisplayName.ToLower();

        // Save the changes
        WorkflowInfoProvider.SetWorkflowInfo(updateWorkflow);

        return true;
    }
```

```
    return false;
}
```

The following example gets and bulk updates workflows matching the specified WHERE condition.

```
private bool GetAndBulkUpdateWorkflows()
{
    // Prepare the parameters
    string where = "WorkflowName LIKE N'MyNewWorkflow%'";

    // Get the data
    DataSet workflows = WorkflowInfoProvider.GetWorkflows(where, null, 0, null);
    if (!DataHelper.DataSourceIsEmpty(workflows))
    {
        // Loop through the individual items
        foreach (DataRow workflowDr in workflows.Tables[0].Rows)
        {
            // Create object from DataRow
            WorkflowInfo modifyWorkflow = new WorkflowInfo(workflowDr);

            // Update the properties
            modifyWorkflow.WorkflowDisplayName =
modifyWorkflow.WorkflowDisplayName.ToUpper();

            // Save the changes
            WorkflowInfoProvider.SetWorkflowInfo(modifyWorkflow);
        }

        return true;
    }

    return false;
}
```

The following example deletes the workflow created by the first example on this page.

```
private bool DeleteWorkflow()
{
    // Get the workflow
    WorkflowInfo deleteWorkflow = WorkflowInfoProvider.GetWorkflowInfo
("MyNewWorkflow");

    // Delete the workflow
    WorkflowInfoProvider.DeleteWorkflowInfo(deleteWorkflow);

    return (deleteWorkflow != null);
}
```

4.10.8.4.3 Managing workflow steps

The following example creates a workflow step.

```
private bool CreateWorkflowStep()
{
    // Get the workflow
    WorkflowInfo myWorkflow = WorkflowInfoProvider.GetWorkflowInfo
("MyNewWorkflow");
    if (myWorkflow != null)
    {
        // Create new workflow step object
        WorkflowStepInfo newStep = new WorkflowStepInfo();

        // Set the properties
        newStep.StepWorkflowID = myWorkflow.WorkflowID;
        newStep.StepName = "MyNewWorkflowStep";
        newStep.StepDisplayName = "My new workflow step";
        newStep.StepOrder = 1;

        // Save the step into database
        WorkflowStepInfoProvider.SetWorkflowStepInfo(newStep);

        // Ensure correct step order
        WorkflowStepInfoProvider.InitStepOrders(newStep.StepWorkflowID);

        return true;
    }

    return false;
}
```

The following example adds a role to the workflow step created by the code example above. Members of the added role will be able to approve documents in the step.

```
private bool AddRoleToStep()
{
    // Get the workflow
    WorkflowInfo workflow = WorkflowInfoProvider.GetWorkflowInfo("MyNewWorkflow");

    if (workflow != null)
    {
        // Get the custom step
        WorkflowStepInfo step = WorkflowStepInfoProvider.GetWorkflowStepInfo
("MyNewWorkflowStep", workflow.WorkflowID);

        if (step != null)
        {
            // Get the role to be assigned to the step
            RoleInfo role = RoleInfoProvider.GetRoleInfo("CMSEditor",
CMSContext.CurrentSiteID);
```

```
            if (role != null)
            {
                // Make the assignment
                WorkflowStepRoleInfoProvider.AddRoleToWorkflowStep(step.StepID,
role.RoleID);

                return true;
            }
            else
            {
                // Role was not found
                apiAddRoleToStep.ErrorMessage = "Role 'CMS Editors' was not
found.";
            }
        }
        else
        {
            // Step was not found
            apiAddRoleToStep.ErrorMessage = "Step 'My new workflow step' was not
found.";
        }
    }

    return false;
}
```

The following example removes the role added by the above code example from the workflow step.

```
bool RemoveRoleFromStep()
{
    // Get the workflow
    WorkflowInfo workflow = WorkflowInfoProvider.GetWorkflowInfo("MyNewWorkflow");

    if (workflow != null)
    {
        // Get the custom step
        WorkflowStepInfo step = WorkflowStepInfoProvider.GetWorkflowStepInfo
("MyNewWorkflowStep", workflow.WorkflowID);

        if (step != null)
        {
            // Get the role to be assigned to the step
            RoleInfo role = RoleInfoProvider.GetRoleInfo("CMSEditor",
CMSContext.CurrentSiteID);

            if (role != null)
            {
                // Get the step - role relationship
                WorkflowStepRoleInfo stepRoleInfo =
WorkflowStepRoleInfoProvider.GetWorkflowStepRoleInfo(step.StepID, role.RoleID);

                if (stepRoleInfo != null)
```

```
                {
                    // Remove the assignment
                    WorkflowStepRoleInfoProvider.RemoveRoleFromWorkflowStep
(step.StepID, role.RoleID);

                    return true;
                }
                else
                {
                    // The role is not assigned to the step
                    apiRemoveRoleFromStep.ErrorMessage = "The 'CMS Editors' role
is not assigned to the step.";
                }
            }
            else
            {
                // The role was not found
                apiRemoveRoleFromStep.ErrorMessage = "The role 'CMS Editors' was
not found.";
            }
        }
        else
        {
            // The step was not found
            apiRemoveRoleFromStep.ErrorMessage = "The step 'My new workflow step'
was not found.";
        }
    }

    return false;
}
```

The following example deletes the workflow step created by the first example on this page.

```
private bool DeleteWorkflowStep()
{
    // Get the workflow
    WorkflowInfo workflow = WorkflowInfoProvider.GetWorkflowInfo("MyNewWorkflow");

    if (workflow != null)
    {
        // Get the custom step
        WorkflowStepInfo deleteStep = WorkflowStepInfoProvider.GetWorkflowStepInfo
("MyNewWorkflowStep", workflow.WorkflowID);

        if (deleteStep != null)
        {
            // Remove the step
            WorkflowStepInfoProvider.DeleteWorkflowStepInfo(deleteStep);
            return true;
        }
    }
```

```
    return false;
}
```

4.10.8.4.4 Managing workflow scopes

The following example creates a workflow scope.

```
private bool CreateWorkflowScope()
{
    // Get the workflow
    WorkflowInfo workflow = WorkflowInfoProvider.GetWorkflowInfo("MyNewWorkflow");

    if (workflow != null)
    {
        // Create new workflow scope object
        WorkflowScopeInfo newScope = new WorkflowScopeInfo();

        // Get the site default culture from settings
        string cultureCode = SettingsKeyProvider.GetStringValue
(CMSContext.CurrentSiteName + ".CMSDefaultCultureCode");
        CultureInfo culture = CultureInfoProvider.GetCultureInfo(cultureCode);

        // Get root document type class ID
        int classID = DataClassInfoProvider.GetDataClass("CMS.Root").ClassID;

        // Set the properties
        newScope.ScopeStartingPath = "/";
        newScope.ScopeCultureID = culture.CultureID;
        newScope.ScopeClassID = classID;

        newScope.ScopeWorkflowID = workflow.WorkflowID;
        newScope.ScopeSiteID = CMSContext.CurrentSiteID;

        // Save the workflow scope
        WorkflowScopeInfoProvider.SetWorkflowScopeInfo(newScope);

        return true;
    }

    return false;
}
```

The following example gets and updates the workflow scope created by the code example above.

```
private bool GetAndUpdateWorkflowScope()
{

    // Get the workflow
    WorkflowInfo workflow = WorkflowInfoProvider.GetWorkflowInfo("MyNewWorkflow");
```

```
    if (workflow != null)
    {
        // Get the workflow's scopes
        DataSet scopes = WorkflowScopeInfoProvider.GetWorkflowScopes
(workflow.WorkflowID);

        if (!DataHelper.DataSourceIsEmpty(scopes))
        {
            // Create the scope info object
            WorkflowScopeInfo updateScope = new WorkflowScopeInfo(scopes.Tables[0]
.Rows[0]);

            // Update the properties - the scope will include all cultures and
document types
            updateScope.ScopeCultureID = 0;
            updateScope.ScopeClassID = 0;

            // Save the changes
            WorkflowScopeInfoProvider.SetWorkflowScopeInfo(updateScope);

            return true;
        }
        else
        {
            // No scope was found
            apiGetAndUpdateWorkflowScope.ErrorMessage = "The scope was not
found.";
        }
    }

    return false;
}
```

The following example deletes the workflow scope created by the first example on this page.

```
private bool DeleteWorkflowScope()
{
    // Get the workflow
    WorkflowInfo workflow = WorkflowInfoProvider.GetWorkflowInfo("MyNewWorkflow");

    if (workflow != null)
    {

        // Get the workflow's scopes
        DataSet scopes = WorkflowScopeInfoProvider.GetWorkflowScopes
(workflow.WorkflowID);

        if (!DataHelper.DataSourceIsEmpty(scopes))
        {
            // Create the scope info object
            WorkflowScopeInfo deleteScope = new WorkflowScopeInfo(scopes.Tables[0]
.Rows[0]);
```

```
            // Delete the workflow scope
            WorkflowScopeInfoProvider.DeleteWorkflowScopeInfo(deleteScope);

            return true;
        }
        else
        {
            // No scope was found
            apiDeleteWorkflowScope.ErrorMessage = "The scope was not found.";
        }
    }

    return false;
}
```

4.11 Validators

4.11.1 Overview

Kentico CMS comes with built-in page validation features. The main advantage of these built-in validators over standard web based validation services is that your pages needn't be live to get validated. All that is required is to have an Internet connection. This way, you can validate pages before you publish them and ensure that they are valid before making them available to the general public.

Validation can be performed on the **Validate** tab in **CMS Desk -> Content**, which is only available in **Live site** and **Preview** modes. The validators always validate the version of the document that is displayed in the currently selected view mode. Three types of validators are currently available on the tab:

- HTML validator - checks page output code validity against the (X)HTML standard version declared in page code.
- CSS validator - checks validity of CSS styles used by the page against the CSS 2.1 standard.
- Link checker - checks availability of links on a page.
- Accessibility validator - checks accessibility of the page.

There are situations when the validators may not be available: the **Validate** tab is not accessible in **Live site** mode if the document is not published and in **Preview** mode on multilingual websites if the document does not have its version in the currently selected language.

The **Validate** tab and its sub-tabs can also be hidden to members of certain roles by means of UI personalization. The **Content** module has the **Validation**, **HTML**, **CSS, Link checker** and **Accessibility validator** UI elements. These elements represent the respective tabs, while access to the tabs can be restricted by allowing access to the UI elements only to members of specified roles. See Development -> Membership -> UI personalization -> UI profile configuration for more details.

4.11.2 HTML validator

On the **Validate -> HTML** tab, you can perform validation of output HTML code of the page currently selected in the content tree, specifically of its version displayed in the currently selected view mode (**Preview** or **Live site**). Please note that even if you navigate to some other page in the preview or live

site mode using links on displayed pages and then switch to the **Validate** tab, actions performed on this tab will still be related to the page selected in the content tree.

Validation is always performed against the version of HTML declared in the validated code.

Important!

Your data is sent to an external 3rd party validation service. If you have sensitive, non-public content, we recommend that you use some other way to validate your website.

The following actions are available in the top row:

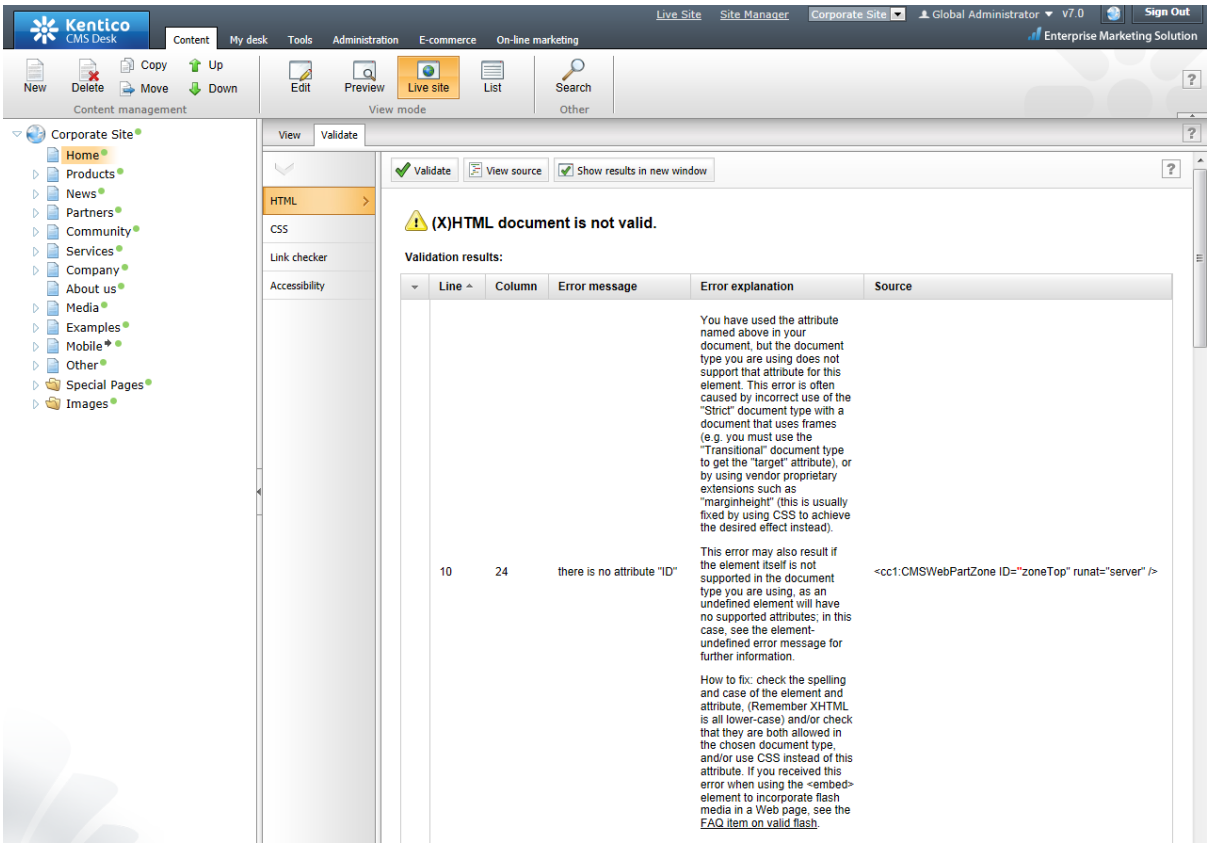
- **Validate** - performs HTML validation of the page currently selected in the content tree and displays validation results below.
- **View source** - opens a new pop-up window where source code of the page selected in the content tree will be displayed.

The following actions are only available after performing validation which showed that the document is not valid:

- **Show results in new window** - displays validation results in a new window.
- **Export to various formats** - After clicking the icon in the header of the **Actions** column, a context menu is displayed, offering options for export of listed data to various types of files.

When validity issues are found in the page's output HTML code, they are listed in a grid in the main area. You can see the following information about each issue:

- **Line** - line of the HTML code on which the issue appears.
- **Column** - column of the HTML code (i.e. number of characters from the beginning of the respective line) where the issue appears.
- **Error message** - message describing the validity issue.
- **Error explanation** - detailed explanation of the validity issue.
- **Source** - extract of the part of the code where the validity issue appears.



4.11.3 CSS validator

The system allows you to validate the CSS style definitions used on pages against the CSS 2.1 standard. The validator checks styles from:

- Linked CSS stylesheets
- Styles declared in the head section of the page code

Inline styles declared as attributes of individual HTML elements are not validated.

To validate a page's CSS:

Warning: During validation, your data is sent to an external 3rd party validation service. If you have sensitive, non-public content, we recommend using other types of validation.

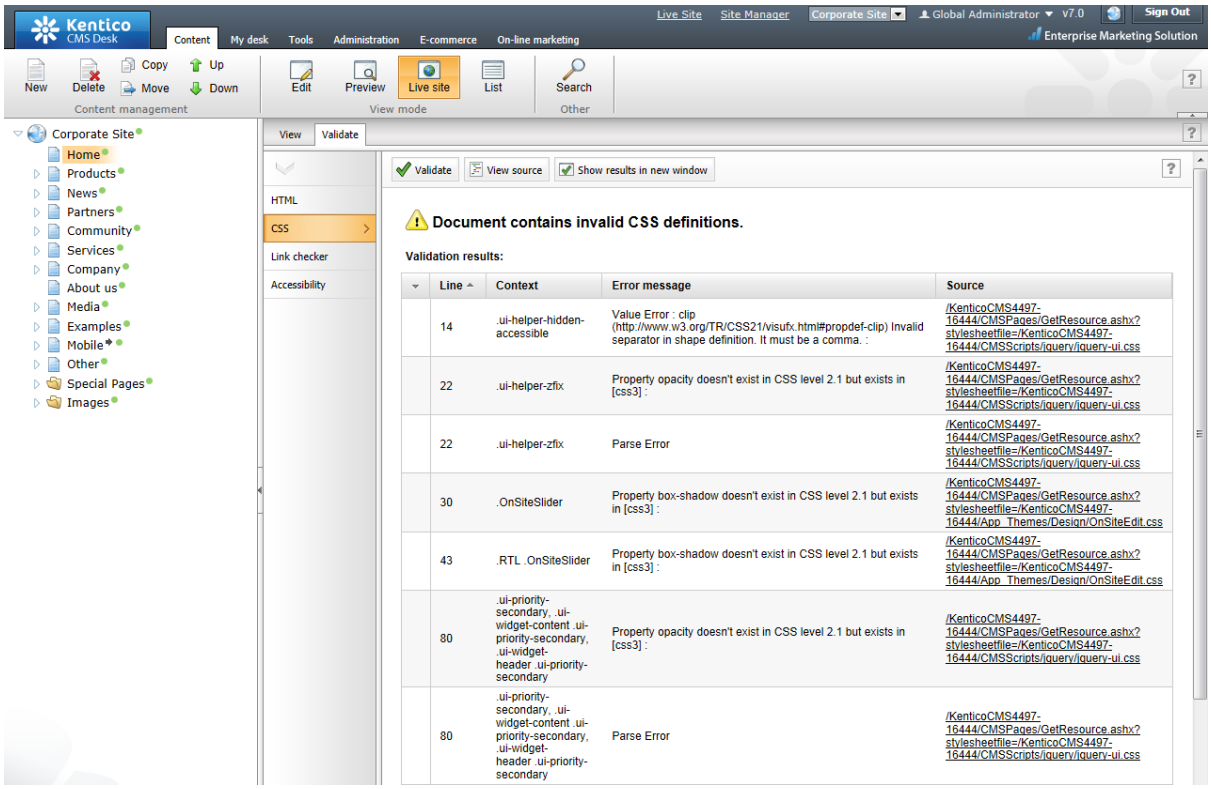
1. Go to **CMS Desk -> Content**.
2. Select the page in the content tree.
3. Switch to **Preview** or **Live site** mode.
4. Open the **Validate -> CSS** tab.
5. Click ✔ **Validate**.

If the validator finds invalid style definitions, the page displays a list of all detected issues. You can see the following information for each issue:

- **Line** - the line number in the CSS stylesheet (or approximate line of the HTML source code) where the issue was found.
- **Context** - name of the CSS class that contains the issue.
- **Error message** - message explaining the issue.
- **Source** - if the issue was found in a stylesheet, the column displays a link to the stylesheet. If you have permissions to edit the stylesheet, the link opens an editing dialog.

Notes:

- We recommend waiting at least 30 seconds between validation requests. If you submit a page for validation multiple times within a short interval, the validation service may temporarily block your page's URL.
- If you navigate to another page in preview mode using links on the displayed pages, and then switch to the **Validate** tab, the system still validates the page selected in the content tree.



4.11.4 Link checker

On the **Validate -> Link checker** tab, you can perform a check for broken links on the page selected in the content tree, specifically of its version displayed in the currently selected view mode (**Preview** or

Live site). Please note that even if you navigate to some other page in the preview or live site mode using links on displayed pages and then switch to the **Validate** tab, actions performed on this tab will still be related to the page selected in the content tree.

Only links within **<a>**, **<img>**, **<script>** and **<link>** elements are checked by the link checker.

The following actions are available in the top row:

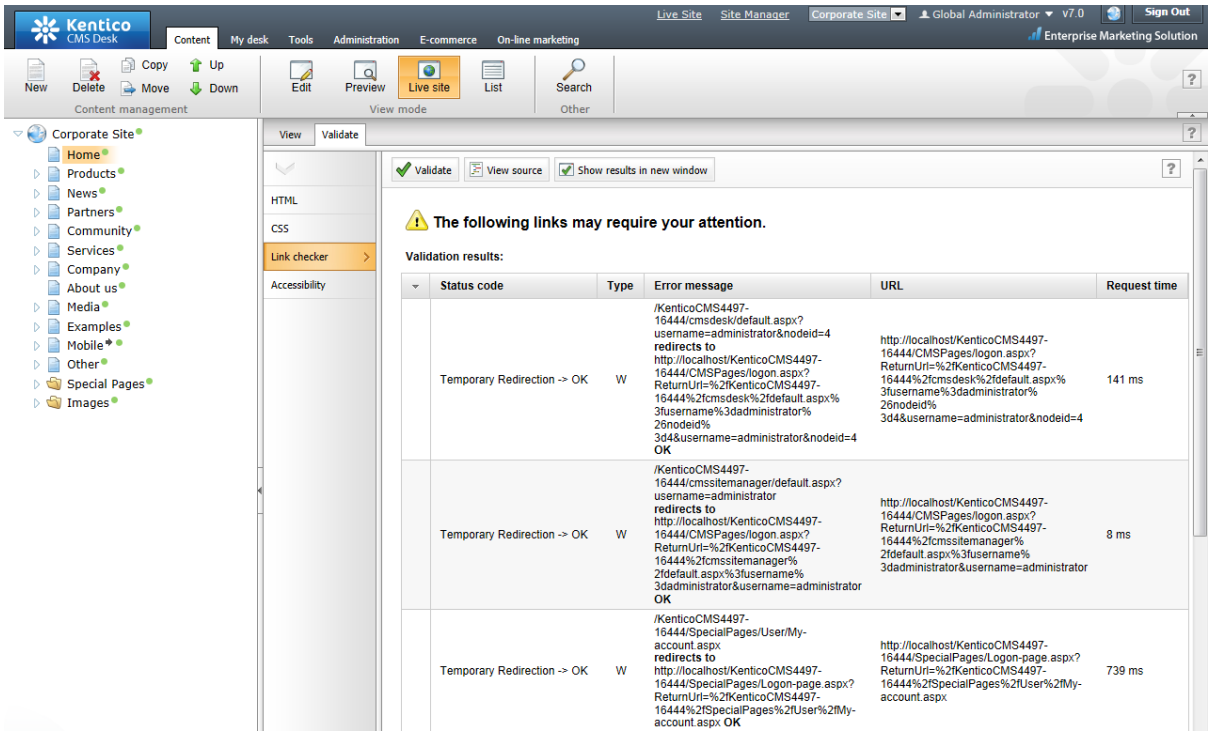
- **Validate** - performs validation of links on the page currently selected in the content tree and displays validation results below.
- **View source** - opens a new pop-up window where source code of the page selected in the content tree will be displayed.

The following actions are only available after performing validation which showed that the document contains broken links:

- **Show results in new window** - displays validation results in a new window.
- **Export to various formats** - After clicking the icon in the header of the **Actions** column, a context menu is displayed, offering options for export of listed data to various types of files.

If some broken links are found on the page, a grid with these links is displayed in the main area. You can see the following information about each broken link:

- **Status code** - HTML status code returned when the link is accessed. Statuses of redirected links are shown for each page within the redirection (e.g. 301 -> 200, 301 -> 301 -> 404, etc.).
- **Type** - W is shown when the link is redirected to a different location (301 status code), E is shown when the link is broken (404 status code).
- **Error message** - message providing details about the link issue.
- **URL** - URL to which the link is pointing.
- **Request time** - time elapsed between sending a request to the link and receiving a response.



4.11.5 Accessibility validator

On the **Validate -> Accessibility** tab, you can perform accessibility validation of the page currently selected in the content tree, specifically of its version displayed in the currently selected view mode (**Preview** or **Live site**). Please note that even if you navigate to some other page in the preview or live site mode using links on displayed pages and switch to the **Validate** tab, actions performed on this tab will still be related to the page selected in the content tree.

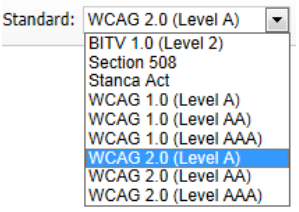
Important!

Your data is sent to an external 3rd party validation service. If you have sensitive, non-public content, we recommend that you use some other way to validate your website.

You can perform the validation against several standards:

- BITV 1.0 (Level 2)
- U.S. Section 508
- Stanca Act
- WCAG 1.0 (Level A)
- WCAG 1.0 (Level AA)
- WCAG 1.0 (Level AAA)
- WCAG 2.0 (Level A)
- WCAG 2.0 (Level AA)
- WCAG 2.0 (Level AAA)

You can choose the standard using the **Standard** drop-down list.



Make sure to use the HTML validator, CSS validator and Link checker before you use the accessibility validator to ensure valid markup.

The following actions are available in the top row:

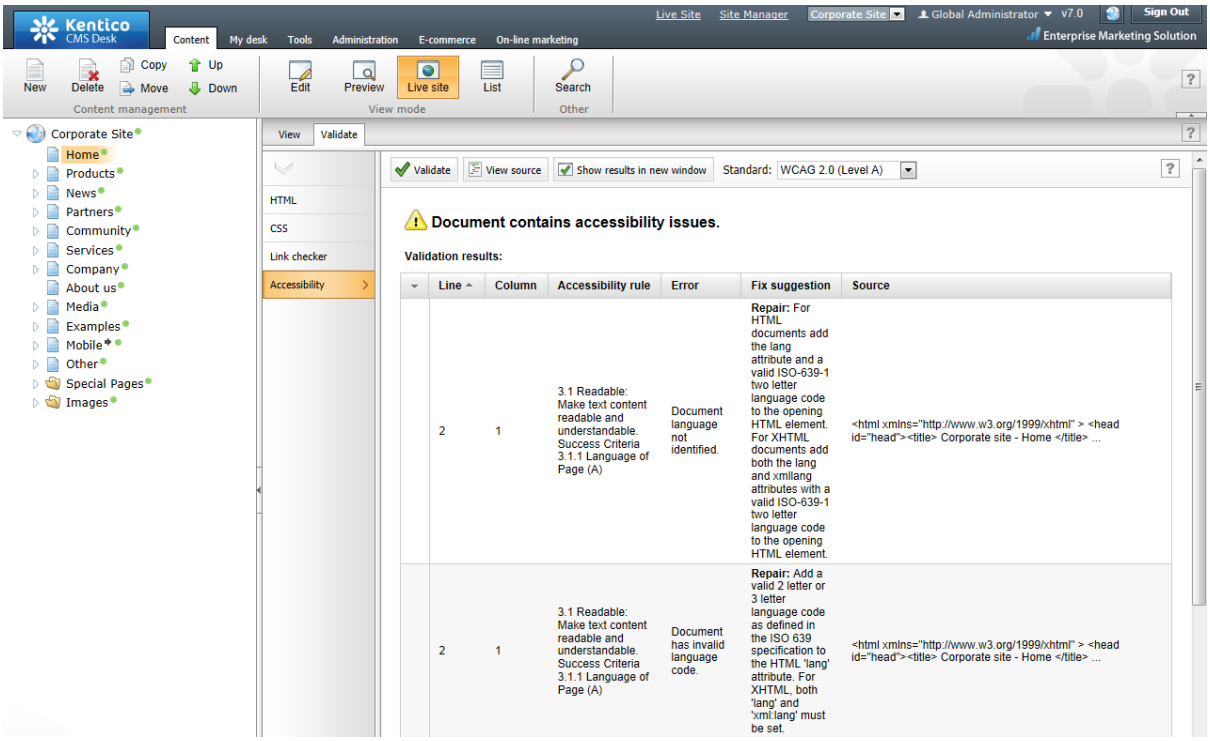
- **Validate** - performs accessibility validation on the page currently selected in the content tree and displays validation results below.
- **View source** - opens a new pop-up window displaying the source code of the page selected in the content tree.

The following actions are only available after performing validation which showed that the document contains accessibility issues:

- **Show results in new window** - displays validation results in a new window.
- **Export to various formats** - after clicking the icon in the first column of the results grid, a context menu is displayed, offering options for export of listed data to various types of files.

If the document contains accessibility issues, a grid is displayed in the main area, listing all issues detected by the validation. You can see the following information about each issue:

- **Line** - line of the HTML code on which the issue appears.
- **Column** - column of the HTML code (i.e. number of characters from the beginning of the respective line) where the issue appears.
- **Accessibility rule** - the standard's accessibility rule based on which the error was generated.
- **Error** - message describing the validity issue.
- **Fix suggestion** - description of steps to be taken to address the accessibility issue.
- **Source** - extract of the part of the code where the accessibility issue appears.

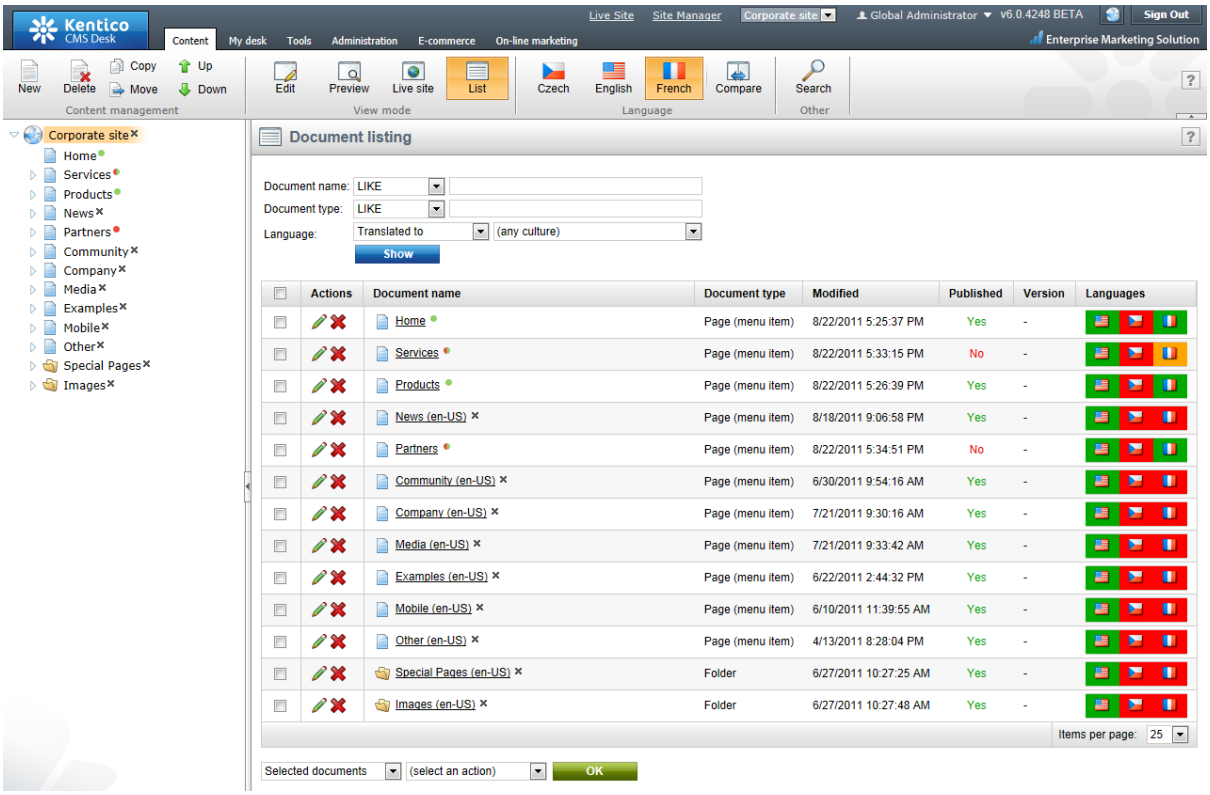


4.12 Document listing

On the **CMS Desk -> Content -> List** tab, you can see an **overview of documents placed under the currently selected document** in the content tree. You can filter the displayed pages using the filter above the list.

The **Document name** column displays the names of the documents in the currently edited culture. If the document's version in this culture **does not exist**, the column displays the name from the **default culture** with the default culture code appended in brackets. Document status icons are displayed next to document names in this column, the same as in the content tree. To learn more about these icons, please refer to Content management -> Editing content -> Document status icons.

The **Languages** column displays the status of the document's particular language versions. More information about language statuses can be found in Development -> Multilingual content -> Managing multilingual websites -> Language status overview.



Single document actions on the List tab

The following operations can be performed for individual documents:

- **Edit** - by clicking the icon, you get redirected to the document's **Edit -> Page** tab in the currently selected culture.
- **Delete** - deletes the document.
- The action opens a context menu with further options that may be used to change the position of the document on the given level of the content tree:
  - **Top**
  - **Up**
  - **Down**
  - **Bottom**
- Clicking a document's name selects it in the content tree, which causes the list to display the documents found under it.

Bulk actions on the List tab

You can also perform **bulk actions** with the listed documents using the two drop-down lists at the bottom. First, you need to select from the following two options in the first drop-down list:

- **Selected documents** - performs the action only with the documents selected by the check-boxes ( ).
- **All documents** - performs the action with all listed documents.

Then you need to choose the action:

- **Move** - moves the documents to the location specified in a raised pop-up dialog.
- **Copy** - copies the documents to the location specified in a raised pop-up dialog.
- **Link** - creates a linked document in the location specified in a raised pop-up dialog.
- **Delete** - deletes the documents.
  - **Delete documents and their history (documents can't be restored)** - deletes the documents permanently without storing them in the recycle bin, which can be accessed in *CMS Desk -> My Desk -> Bin*.
  - **Delete all culture version of the specified documents** - if the documents have multiple language versions, all of them will be removed. If disabled, only the versions belonging to the currently selected culture will be deleted.
- **Translate -** submits the documents for translation into the language currently selected on the main toolbar (via an external service).
- **Publish/Archive** - this option publishes or archives the selected documents. Before the operation is performed, a dialog is displayed, listing the selected documents and offering the following extra options:
  - **Publish/Archive all culture versions of specified documents** - if enabled, all culture versions of the documents will be published/archived.
  - **Publish/Archive also all child documents** - if enabled, all documents located under the selected documents will be published/archived too.
  - **Perform Undo check-out for checked out documents** - this option is displayed only if you want to perform the operation for a document which is checked out (currently edited by some user). If you enable this options, such documents will be published/archived too.

and click **OK** to perform the action.

4.13 Content search

The interface for searching the documents in the content tree can be accessed by clicking the **Search** (🔎) icon in **CMS Desk -> Content**.

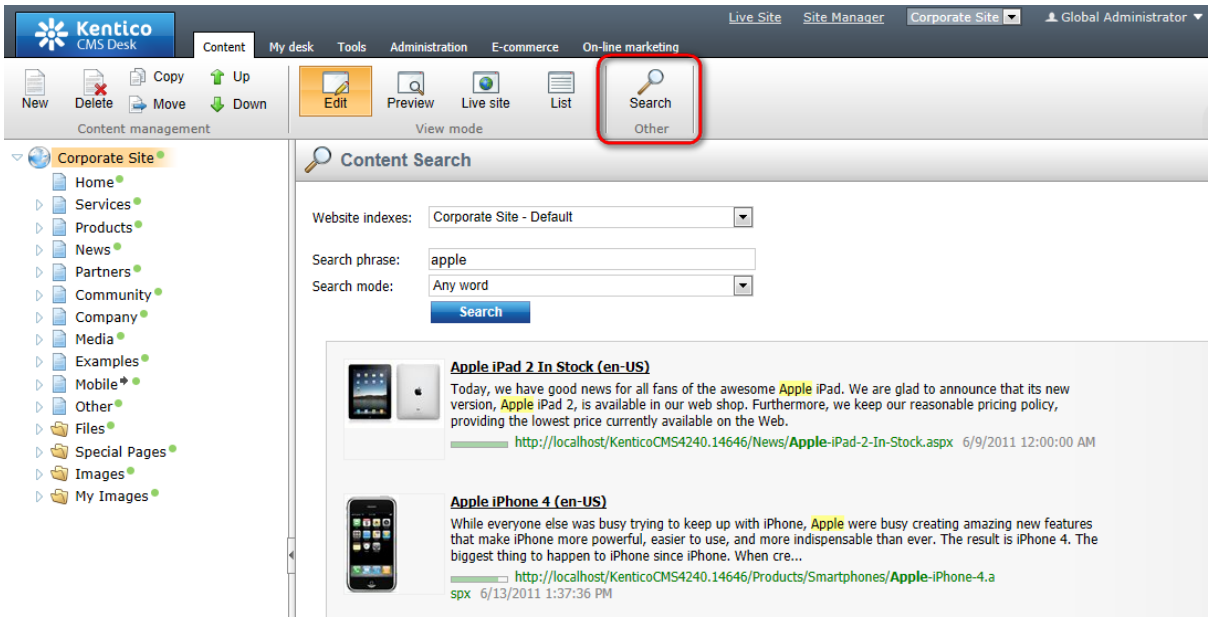
1. First, you need to decide if you want to perform the search using Smart search or SQL search. This depends on the selection made in the **website indexes** drop-down:

- **SQL search** - you need to select *(SQL Search - default)*; slower, but supports search in both published and unpublished documents
- **Smart search** - you need to select a particular smart search index; fast, but does not support search in unpublished documents

2. Then you need to specify the following criteria:

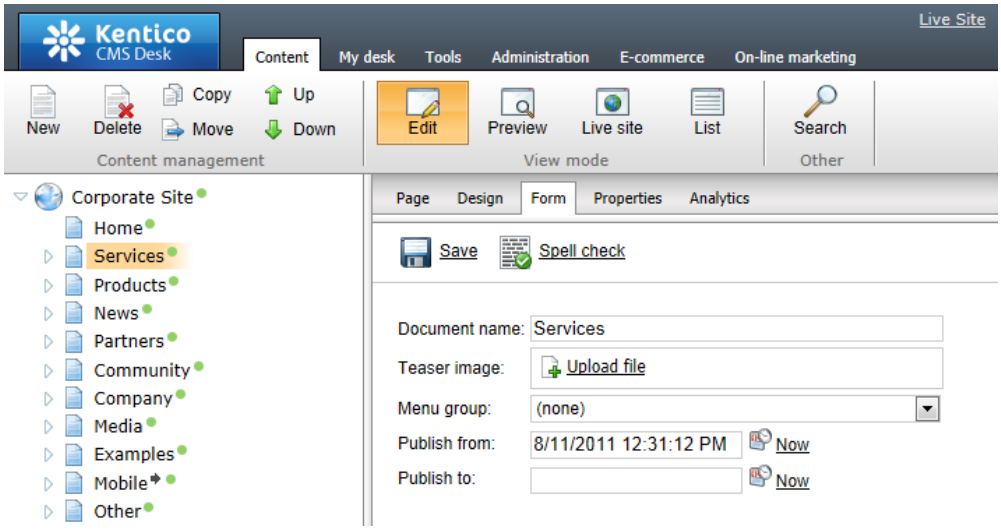
- **Search phrase**: the text that you are looking for; if you are searching using a Smart search index, you can use the syntax described here; if you are searching using SQL search, standard SQL syntax can be used
- **Search mode**: specifies how the search phrase will be used:
  - *Exact phrase* - returns documents where the search phrase is found exactly as entered
  - *Any word* - returns documents where at least one word of the search phrase is found
  - *All words* - returns documents where all words of the search phrase are found, regardless of their position or order in the text

Click **Search**. Found documents will be listed as in the screenshot below.



4.14 Content scheduling

Kentico CMS allows you to specify when the document will be published. When you edit a document on its **Form** tab, you can typically find the **Publish from**/**Publish to** fields at the bottom of the form:



When you set these values, the document will be displayed on the website only in the given time period.

If you do not set the **Publish from** value, the document will be displayed on the live site immediately.
If you do not set the **Publish to** value, the document will never expire.

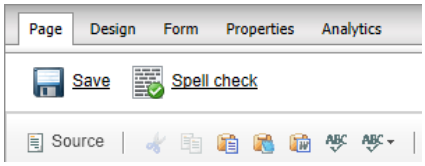
Content scheduling and workflow/versioning

If you set publish from/to values to documents that use workflow, they will not be published before they

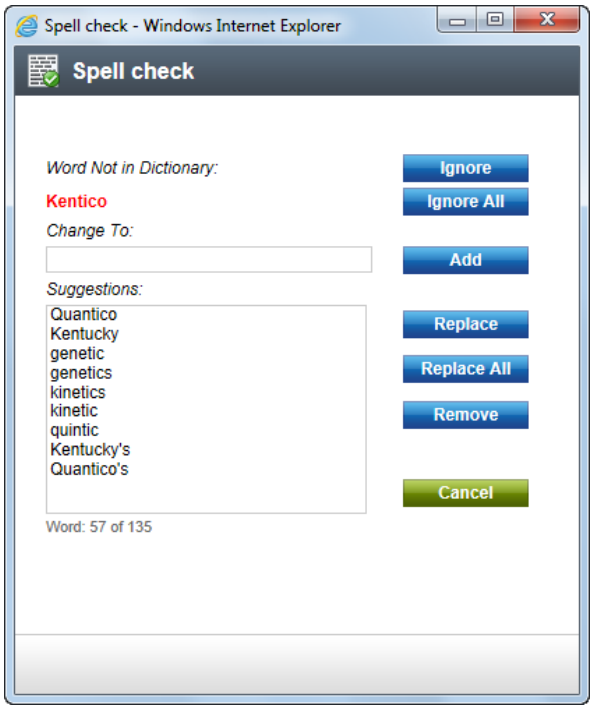
are approved. However, the publishing time may not be exact since the publishing is ensured by a scheduled process that is executed every minute by default. You can check the status of this process in **CMS Desk -> Administration -> Scheduled tasks** -> choose the website and search for the **Content publishing** task.

4.15 Using the built-in spell-checker

You can spell-check all the content on the **Page** and **Form** tabs using the built-in spell-checker:



When you click the **Spell check** button, the spell-checker reads all text fields and checks their content. If it finds any typo, it shows the dialog like this:



You can then ignore the word, add it to the dictionary or replace it with suggested word.

Please note: If you add a new word to dictionary it's only saved in the current session. The next time you sign in to Kentico CMS, the added words will be lost.

Dictionaries

The dictionary is used based on the currently chosen content culture. If no dictionary is available for the current content culture, the default dictionary is used. The default dictionary is specified in the **CMSDefaultSpellCheckerCulture** configuration key in the **appSettings** section of the **web.config** file. By default, it's set to en-US.

Adding addtional dictionaries

The dictionaries are stored in folder **<web project>\App_Data\Dictionaries**. If you need some additional dictionaries, you can download them from the following URLs:

- AR-ae: http://www.kentico.com/Downloads/SpellChecker/ar-AE.zip
- CS-cz: http://www.kentico.com/Downloads/SpellChecker/cs-cz.zip
- DE-de: http://www.kentico.com/Downloads/SpellChecker/de-de.zip
- EL-gr: http://www.kentico.com/Downloads/SpellChecker/el-gr.zip
- EN-au: http://www.kentico.com/Downloads/SpellChecker/en-au.zip
- EN-ca: http://www.kentico.com/Downloads/SpellChecker/en-ca.zip
- EN-gb: http://www.kentico.com/Downloads/SpellChecker/en-gb.zip
- EN-us: http://www.kentico.com/Downloads/SpellChecker/en-us.zip
- ES-es: http://www.kentico.com/Downloads/SpellChecker/es-es.zip
- ES-mx: http://www.kentico.com/Downloads/SpellChecker/es-mx.zip
- FR-fr: http://www.kentico.com/Downloads/SpellChecker/fr-fr.zip
- HE-il: http://www.kentico.com/Downloads/SpellChecker/he-il.zip
- IT-it: http://www.kentico.com/Downloads/SpellChecker/it-it.zip
- NB-no: http://www.kentico.com/Downloads/SpellChecker/nb-no.zip
- NL-nl: http://www.kentico.com/Downloads/SpellChecker/nl-nl.zip
- NN-no: http://www.kentico.com/Downloads/SpellChecker/nn-no.zip
- PL-pl: http://www.kentico.com/Downloads/SpellChecker/pl-pl.zip
- PT-pt: http://www.kentico.com/Downloads/SpellChecker/pt-pt.zip
- RO-ro: http://www.kentico.com/Downloads/SpellChecker/ro-ro.zip
- RU-ru: http://www.kentico.com/Downloads/SpellChecker/ru-ru.zip
- TH-th: http://www.kentico.com/Downloads/SpellChecker/th-th.zip

and unpack them to the dic folder. Then, you should restart the website using **Site Manager -> Administration -> System dialog -> click Restart application**. The file name of the dictionary must match the culture code of the currently edited content - e.g. fr-fr.

4.16 Permissions and security

Kentico CMS allows you to configure permissions for particular site sections or even particular documents. You can find more details in the Development -> Membership, permissions and security chapter.

4.17 FAQ

This topic provides information about the most common issues and their solutions. If you do not find the answer here, please contact our technical support.

Q: The CMS Desk doesn't display the content tree or the WYSIWYG editor doesn't open.
A: Please check that you use one of the supported browsers and that you have JavaScript enabled. You should also unblock the pop-up blocker or any similar blocker for the website with Kentico CMS.

Q: I have modified a document, but the changes are not displayed on the live site.
A: This may be caused by several reasons:

1. **Caching** - if caching is used, the changes may not be displayed on the website immediately.
2. **Workflow** - if you use workflow, the changes are published on the live website only after the

document has been approved in all workflow steps.

Q: I have defined related documents, but they are not displayed anywhere on the page.
A: You (your developer) need to add some web part/control that will display the related documents - e.g. the **Listings and viewers/Related documents** web part or **Repeater** web part.

Q: I need to add a custom field to the document. Is it possible?
A: Yes, every structured document type can be customized with your own fields. See Document types for details.

4.18 Content management internals and API

4.18.1 Overview

In this chapter, you will learn which database tables and API classes are to handle documents. Examples of how the documents API can be used are divided into separate sub-chapters:

- Basics
- Advanced
- Document attachments
- Document relationships
- Workflow basics
- Workflow advanced

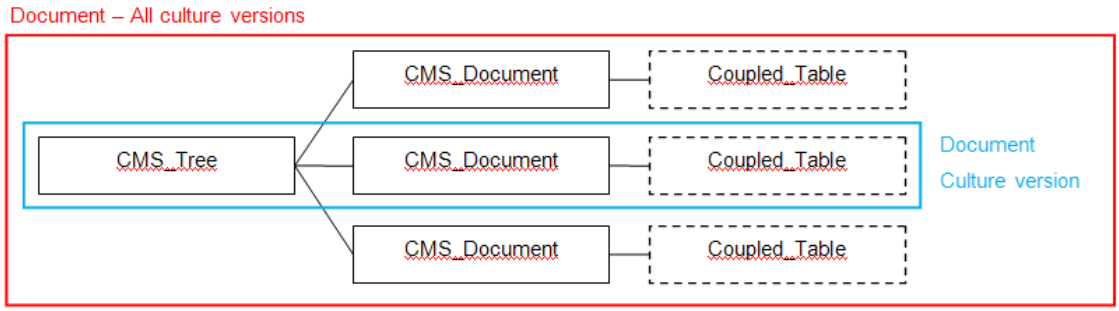
Advanced API examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all methods from the module classes, please refer to Kentico CMS API Reference.

4.18.2 Database tables

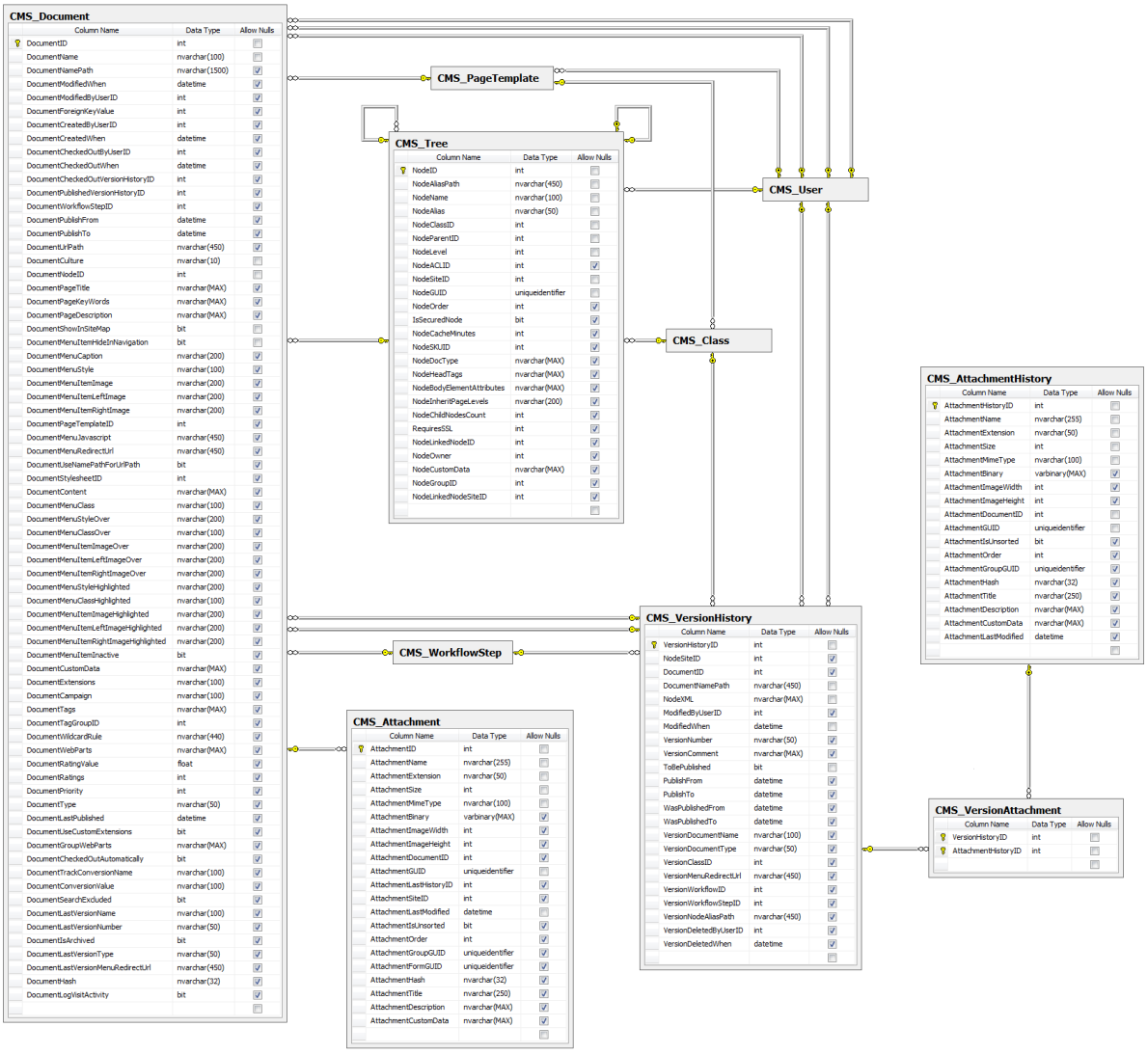
Document data is stored in several joined tables that are used for the tree structure, multilingual support and custom document fields. The document record consists of the following tables:

- **CMS_Tree** - a table with basic document data shared between different language versions of same document. This table determines the tree structure of the website document content. It contains **one record for all culture versions** of the document. The table does not contain any versioned data.
- **CMS_Document** - a table with document data of a specific language version of the document. It contains **several records** for every document, each one representing one language version of the document. Some of the document columns are versioned columns.
- **Coupled table** - a table that contains document-type specific fields defined by the developer. For example the News document type has a Content_News coupled table that contains NewsTitle, NewsSummary, NewsText and other fields specific for news. Coupled table primary key is referenced by the value of **DocumentForeignKeyValue** column and the table is determined by type of the document. Container document types, such as folder, do not have any coupled table. All columns from coupled tables are versioned. Each culture version of the coupled document contains one record in the coupled table.

The following figure shows how the three tables are connected:

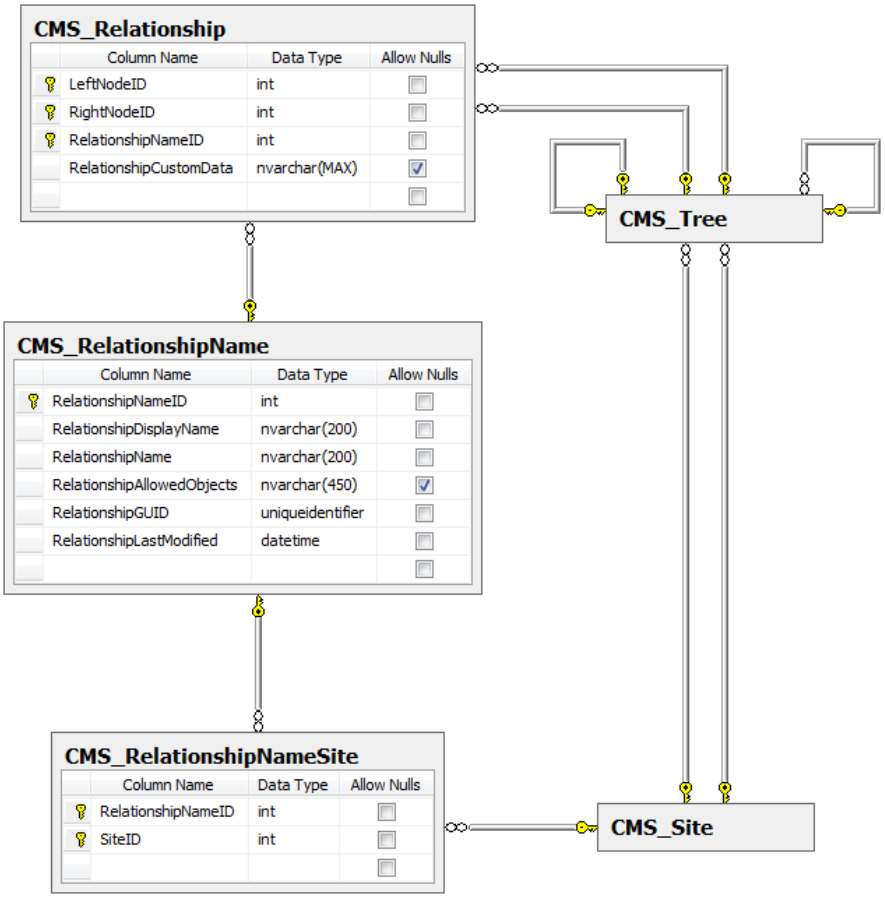


- **CMS_Attachment** - contains records representing document attachments.
- **CMS_VersionHistory** - contains records representing document versions. When a new document version is being created, it is only stored in this table. When it is published, respective records in the *CMS_Tree* and *CMS_Document* tables are updated with data from the new version record in this table.
- **CMS_AttachmentHistory** - contains records representing attachments of document versions. The main purpose of this table is to avoid redundancy when a new document version with the same attachments is created. When a new document version is published, respective records in the *CMS_Attachment* table are updated with data from the new version record in this table.
- **CMS_VersionAttachment** - contains records representing relationships between document versions and their document attachments.



Document relationships use the following database tables:

- **CMS_Relationship** - contains records representing relationships between two documents.
- **CMS_RelationshipName** - contains records representing relationship names.
- **CMS_RelationshipNameSite** - contains records representing assignments of relationship names to websites.



4.18.3 API classes

If you are not familiar with database table data management through Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

CMS_Tree table API:

- **CMS.DocumentEngine.TreeNode** - encapsulates data from the CMS_Tree and CMS_Document tables and the respective coupled tables.
- **CMS.DocumentEngine.TreeProvider** - provides functionality for manipulation with tree nodes.

CMS_Relationship table API:

- **CMS.DocumentEngine.RelationshipInfo** - represents a relationship between two documents.
- **CMS.DocumentEngine.RelationshipProvider** - provides functionality for management of document relationships.

CMS_RelationshipName table API:

- **CMS.SiteProvider.RelationshipNameInfo** - represents one relationship name object.
- **CMS.SiteProvider.RelationshipNameInfoProvider** - provides functionality for management of relationship names.

CMS_RelationshipSiteName table API:

- **CMS.SiteProvider.RelationshipNameSiteInfo** - represents assignment of one relationship name to one website.
- **CMS.SiteProvider.RelationshipNameSiteInfoProvider** - provides functionality for management of relationship name assignments to websites.

4.18.4 API examples

4.18.4.1 Basics

4.18.4.1.1 Overview

These topics show examples of how the documents API can be used:

- Creating documents
- Managing documents
- Deleting documents

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Documents\Basics\Default.aspx.cs***.

The documents API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.CMSHelper;
using CMS.GlobalHelper;
using CMS.SiteProvider;
using CMS.DocumentEngine;
using CMS.UIControls;
```

4.18.4.1.2 Creating documents

The following example creates a document structure used by all the following code examples in this chapter.

```
private bool CreateDocumentStructure()
{
    // Add a new culture to the current site
    CultureInfo culture = CultureInfoProvider.GetCultureInfo("de-de");
    CultureSiteInfoProvider.AddCultureToSite(culture.CultureID,
```

```
CMSContext.CurrentSiteID);

    // Create new instance of the Tree provider
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get parent node
    TreeNode parentNode = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/",
"en-us");

    if (parentNode != null)
    {
        // Create the API Example folder
        TreeNode newNode = TreeNode.New("CMS.Folder", tree);

        newNode.DocumentName = "API Example";
        newNode.DocumentCulture = "en-us";

        newNode.Insert(parentNode);

        parentNode = newNode;

        // Create the Source folder - the document to be moved will be stored here
        newNode = TreeNode.New("CMS.Folder", tree);

        newNode.DocumentName = "Source";
        newNode.DocumentCulture = "en-us";

        newNode.Insert(parentNode);

        // Create the Target folder - a document will be moved here
        newNode = TreeNode.New("CMS.Folder", tree);

        newNode.DocumentName = "Target";
        newNode.DocumentCulture = "en-us";

        newNode.Insert(parentNode);

        return true;
    }

    return false;
}
```

The following example creates a document.

```
private bool CreateDocument()
{
    // Create an instance of the Tree provider first
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get the parent node - the API Example folder
    TreeNode parentNode = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/API-
```

```
Example", "en-us");

    if (parentNode != null)
    {
        // Create a new instance of the Tree node
        TreeNode newNode = TreeNode.New("CMS.MenuItem", tree);

        // Set the document's properties
        newNode.DocumentName = "My new document";
        newNode.DocumentCulture = "en-us";

        // Insert the document into the content tree
        newNode.Insert(parentNode);

        return true;
    }

    return false;
}
```

The following example creates a new culture version of the document created by the example above.

```
private bool CreateNewCultureVersion()
{
    // Create an instance of the Tree provider first
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get the document in the english culture
    TreeNode node = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/API-
Example/My-new-document", "en-us");

    if (node != null)
    {
        // Translate its name
        node.DocumentName = "My new translation";
        node.SetValue("MenuItemName", "My new translation");

        // Insert into database
        node.InsertAsNewCultureVersion("de-de");

        return true;
    }

    return false;
}
```

The following example creates a linked document based on the document created by the first code example on this page.

```
private bool CreateLinkedDocument()
```

```
{
    // Create an instance of the Tree provider first
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get the parent document
    TreeNode parentNode = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/API-
Example", "en-us");

    if (parentNode != null)
    {
        // Get the original document
        TreeNode node = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/API-
Example/My-new-document", "en-us");

        if (node != null )
        {
            // Insert the link
            node.InsertAsLink(parentNode);

            return true;
        }
        else
        {
            apiCreateLinkedDocument.ErrorMessage = "Document 'My new document' was
not found.";
        }
    }

    return false;
}
```

4.18.4.1.3 Managing documents

Please note

For these examples to produce meaningful results, it is expected to run the API examples on the Creating documents page first.

The following example gets all documents located under a folder in the content tree and updates them.

```
private bool GetAndUpdateDocuments()
{
    // Create an instance of the Tree provider first
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Fill dataset with documents
    DataSet documents = tree.SelectNodes(CMSContext.CurrentSiteName, "/API-
Example/%", "en-us", false, "CMS.MenuItem");
```

```
    if (!DataHelper.DataSourceIsEmpty(documents))
    {
        // Loop through all documents
        foreach (DataRow documentRow in documents.Tables[0].Rows)
        {
            // Create a new Tree node from the data row
            TreeNode editDocument = TreeNode.New(documentRow, "CMS.MenuItem",
tree);

            string newName = editDocument.DocumentName.ToLower();

            // Change coupled data
            editDocument.SetValue("MenuItemName", newName);
            // Change document data
            editDocument.DocumentName = newName;

            // Save to database
            editDocument.Update();

        }

        return true;
    }

    return false;
}
```

The following example creates a copy of a document.

```
private bool CopyDocument()
{
    // Create an instance of the Tree provider first
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get the document
    TreeNode node = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/API-
Example/My-new-document", "en-us");

    // Get the new parent document
    TreeNode parentNode = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/API-
Example/Source", "en-us");

    if ((node != null) && (parentNode != null))
    {
        // Copy the document
        tree.CopyNode(node, parentNode.NodeID);

        return true;
    }

    return false;
```

```
}
```

The following example moves a document.

```
private bool MoveDocument()
{
    // Create an instance of the Tree provider first
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get the document
    TreeNode node = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/API-
Example/Source/My-new-document", "en-us");

    // Get the new parent document
    TreeNode parentNode = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/API-
Example/Target", "en-us");

    if ((node != null) && (parentNode != null))
    {
        // Move the document
        tree.MoveNode(node, parentNode.NodeID);

        return true;
    }

    return false;
}
```

4.18.4.1.4 Deleting documents

Please note

For these examples to produce meaningful results, it is expected to run the API examples on the Creating documents page first.

The following example deletes a linked document.

```
private bool DeleteLinkedDocuments()
{
    // Create an instance of the Tree provider first
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    string siteName = CMSContext.CurrentSiteName;

    // Get the document
    TreeNode node = tree.SelectSingleNode(siteName, "/API-Example/My-new-
```

```
document", "en-us");

    if (node != null)
    {
        // Prepare the where condition
        string where = "NodeLinkedNodeID = " + node.NodeID;

        // Get linked documents' IDs
        DataSet documents = tree.SelectNodes(siteName, "/API-Example/%", "en-us",
false, null, where, null, -1, false, -1, "NodeID");

        if (!DataHelper.DataSourceIsEmpty(documents))
        {
            // Loop through the documents and delete them. Alternatively, the
DeleteLinks method from the TreeProvider can be used to delete all document's
links.
            foreach (DataRow documentRow in documents.Tables[0].Rows)
            {
                // Get additional document data
                TreeNode document = tree.SelectSingleNode
(ValidationHelper.GetInteger(documentRow["NodeID"], 0));

                // Delete the document
                document.Delete();
            }

            return true;
        }
        else
        {
            apiDeleteLinkedDocuments.ErrorMessage = "No linked documents were
found.";
        }
    }

    return false;
}
```

The following example deletes a document's culture version.

```
private bool DeleteCultureVersion()
{
    // Create an instance of the Tree provider first
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get the german culture version of the document
    TreeNode node = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/API-
Example/My-new-document", "de-de");

    if (node != null)
    {
        // Delete the document
```

```
        node.Delete();

        return true;
    }

    return false;
}
```

The following example deletes a document.

```
private bool DeleteDocument()
{
    // Create an instance of the Tree provider first
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get the document
    TreeNode node = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/API-
Example/My-new-document", "en-us");

    if (node != null)
    {
        // Delete the document and all its culture versions
        node.DeleteAllCultures();

        return true;
    }

    return false;
}
```

The following example deletes the example document structure.

```
private bool DeleteDocumentStructure()
{
    // Create an instance of the Tree provider first
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get the API Example folder
    TreeNode node = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/API-
Example", "en-us");

    if (node != null)
    {
        // Delete the folder and all child documents
        node.DeleteAllCultures();
    }

    CultureInfo culture = CultureInfoProvider.GetCultureInfo("de-de");

    // Remove the example culture from the site
```

```
    CultureSiteInfoProvider.RemoveCultureFromSite(culture.CultureID,
CMSContext.CurrentSiteID);

    return true;
}
```

4.18.4.2 Advanced

4.18.4.2.1 Overview

These topics show examples of how the documents API can be used for advanced tasks:

- Sample document structure
- Organizing documents
- Recycle bin

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Documents\Advanced\Default.aspx.cs***.

The documents API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.CMSHelper;
using CMS.GlobalHelper;
using CMS.DocumentEngine;
using CMS.UIControls;
using CMS.WorkflowEngine;
```

4.18.4.2.2 Sample document structure

The following example creates a sample document structure that will be used by the API examples in the following topics.

```
private bool CreateDocumentStructure()
{
    // Create an instance of the Tree provider
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // First get the root node
```

```
    TreeNode parentNode = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/",
"en-us");

    if (parentNode != null)
    {
        // First create a folder
        TreeNode node = TreeNode.New("CMS.Folder", tree);

        node.DocumentName = "API Example";
        node.DocumentCulture = "en-us";

        node.Insert(parentNode);

        parentNode = node;

        // Create a few documents
        for (int i = 1; i <= 3; i++)
        {
            node = TreeNode.New("CMS.MenuItem", tree);

            node.DocumentName = "Document " + i;
            node.DocumentCulture = "en-us";

            node.Insert(parentNode);
        }

        return true;
    }

    return false;
}
```

The following example deletes the sample document structure created by the code example above.

```
private bool DeleteDocumentStructure()
{
    // Create an instance of the Tree provider
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get the API Example folder
    TreeNode node = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/API-
Example", "en-us");

    // Delete the folder including all dependencies and child documents
    node.Delete();

    return true;
}
```

4.18.4.2.3 Organizing documents

The following example moves **Document 2** created by the code example in the Sample document structure topic up in the content tree.

```
private bool MoveDocumentUp()
{
    // Create an instance of the Tree provider
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Select a node
    TreeNode node = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/API-
Example/Document-2", "en-us");

    if (node != null)
    {
        // Move the node up
        tree.MoveNodeUp(node.NodeID);

        return true;
    }

    return false;
}
```

The following example moves **Document 1** created by the code example in the Sample document structure topic down in the content tree.

```
private bool MoveDocumentDown()
{
    // Create an instance of the Tree provider
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Select a node
    TreeNode node = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/API-
Example/Document-1", "en-us");

    if (node != null)
    {
        // Move the node up
        tree.MoveNodeDown(node.NodeID);

        return true;
    }

    return false;
}
```

The following example sorts documents created by the code example in the Sample document structure topic alphabetically.

```
private bool SortDocumentsAlphabetically()
{
    // Create an instance of the Tree provider
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Select a node
    TreeNode node = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/API-
Example", "en-us");

    if (node != null)
    {
        // Sort its child nodes alphabetically - ascending
        tree.OrderNodesAlphabetically(node.NodeID, true);

        return true;
    }

    return false;
}
```

The following example sorts documents created by the code example in the Sample document structure topic by date and time of creation from the oldest to the newest.

```
private bool SortDocumentsByDate()
{
    // Create an instance of the Tree provider
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Select a node
    TreeNode node = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/API-
Example", "en-us");

    if (node != null)
    {
        // Sort its child nodes by date - descending
        tree.OrderNodesByDate(node.NodeID, false);

        return true;
    }

    return false;
}
```

4.18.4.2.4 Recycle bin

The following example moves **Document 1** created by the code example in the Sample document structure topic the recycle bin.

```
private bool MoveToRecycleBin()
{
```

```
    // Create an instance of the Tree provider
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get the document
    TreeNode node = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/API-
Example/Document-1", "en-us");

    if (node != null)
    {
        // Delete the document without destroying its history
        DocumentHelper.DeleteDocument(node, tree, true, false, true);

        return true;
    }

    return false;
}
```

The following example restores the document moved to the recycle bin by the example above back to the content tree.

```
private bool RestoreFromRecycleBin()
{
    // Prepare the where condition
    string where = "VersionNodeAliasPath LIKE N'/API-Example/Document-1'";

    // Get the recycled document
    DataSet recycleBin = VersionHistoryInfoProvider.GetRecycleBin
(CMSContext.CurrentSiteID, where, null, 0, "VersionHistoryID");

    if (!DataHelper.DataSourceIsEmpty(recycleBin))
    {
        // Create a new version history info object from the data row
        VersionHistoryInfo version = new VersionHistoryInfo(recycleBin.Tables[0]
.Rows[0]);

        // Create a new version manager instance and restore the document
        VersionManager manager = VersionManager.GetInstance(new TreeProvider
(CMSContext.CurrentUser));
        manager.RestoreDocument(version.VersionHistoryID);

        return true;
    }

    return false;
}
```

4.18.4.3 Document attachments

4.18.4.3.1 Overview

These topics show examples of how the document attachments API can be used:

- Sample document structure
- Managing attachments

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Documents\Attachments\Default.aspx.cs***.

The document attachments API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.GlobalHelper;
using CMS.UIControls;
using CMS.CMSHelper;
using CMS.DocumentEngine;
using CMS.WorkflowEngine;
```

4.18.4.3.2 Sample document structure

The following example is not directly related to document attachments. It only creates a sample document to which attachments will be added and managed by the code examples in the following topic.

```
private bool CreateExampleDocument()
{
    // Create a new instance of the Tree provider
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get site root node
    TreeNode parentNode = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/",
"en-us");

    if (parentNode != null)
    {
        // Create the document
        TreeNode node = TreeNode.New("CMS.MenuItem", tree);

        node.DocumentName = "API Example";
        node.DocumentCulture = "en-us";
```

```
        node.Insert(parentNode);

        return true;
    }

    return false;
}
```

The following example deletes the sample document created by the code example above.

```
private bool DeleteExampleDocument()
{
    // Create a new instance of the Tree provider
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get the example document
    TreeNode node = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/API-
Example", "en-us");

    if (node != null)
    {
        // Delete the document
        node.Delete();

        return true;
    }

    return false;
}
```

4.18.4.3.3 Managing attachments

The following example adds an unsorted attachment to the document created by the example in the previous topic.

```
private bool InsertUnsortedAttachment()
{
    // Create a new instance of the Tree provider
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get the document
    TreeNode node = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/API-
Example", "en-us");

    // Path to the file to be inserted. This example uses an explicitly defined
file path. However, you can use an object of the HttpPostedFile type (uploaded via
an upload control).
    string postedFile = Server.MapPath("Files/file.png");
```

```
    if (node != null)
    {
        // Insert the attachment
        return (DocumentHelper.AddUnsortedAttachment(node, Guid.NewGuid(),
postedFile, tree, ImageHelper.AUTOSIZE, ImageHelper.AUTOSIZE,
ImageHelper.AUTOSIZE) != null);
    }

    return false;
}
```

The following example inserts a field attachment to the document created by the example in the previous topic.

```
private bool InsertFieldAttachment()
{
    // Create a new instance of the Tree provider
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get the example document
    TreeNode node = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/API-
Example", "en-us");

    if (node != null)
    {
        AttachmentInfo newAttachment = null;

        // Path to the file to be inserted. This example uses an explicitly
defined file path. However, you can use an object of the HttpPostedFile type
(uploaded via an upload control).
        string postedFile = Server.MapPath("Files/file.png");

        // Insert the attachment and update the document with its GUID
        newAttachment = DocumentHelper.AddAttachment(node, "MenuItemTeaserImage",
postedFile, tree);
        node.Update();

        if (newAttachment != null)
        {
            return true;
        }

        apiInsertFieldAttachment.ErrorMessage = "Couldn't insert the attachment.";
    }

    return false;
}
```

The following example moves the unsorted attachment created by the first example on this page down in the order of the document's unsorted attachments.

```
private bool MoveAttachmentDown()
{
    // Create a new instance of the Tree provider
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get the example document
    TreeNode node = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/API-
Example", "en-us");

    if (node != null)
    {
        string where = "AttachmentIsUnsorted = 1";
        string orderBy = "AttachmentLastModified DESC";

        // Get the document's unsorted attachments with the latest on top
        DataSet attachments = DocumentHelper.GetAttachments(node, where, orderBy,
false, tree);

        if (!DataHelper.DataSourceIsEmpty(attachments))
        {
            // Create attachment info object from the first DataRow
            AttachmentInfo attachment = new AttachmentInfo(attachments.Tables[0]
.Rows[0]);

            // Move the attachment
            DocumentHelper.MoveAttachmentDown(attachment.AttachmentGUID, node);

            return true;
        }
        else
        {
            apiMoveAttachmentDown.ErrorMessage = "No attachments were found.";
        }
    }

    return false;
}
```

The following example moves the unsorted attachment created by the first example on this page up in the order of the document's unsorted attachments.

```
private bool MoveAttachmentUp()
{
    // Create a new instance of the Tree provider
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get the example document
    TreeNode node = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/API-
Example", "en-us");

    if (node != null)
    {
```

```
        string where = "AttachmentIsUnsorted = 1";
        string orderBy = "AttachmentLastModified DESC";

        // Get the document's unsorted attachments with the latest on top
        DataSet attachments = DocumentHelper.GetAttachments(node, where, orderBy,
false, tree);

        if (!DataHelper.DataSourceIsEmpty(attachments))
        {
            // Create attachment info object from the first DataRow
            AttachmentInfo attachment = new AttachmentInfo(attachments.Tables[0]
.Rows[0]);

            // Move the attachment
            DocumentHelper.MoveAttachmentUp(attachment.AttachmentGUID, node);

            return true;
        }
        else
        {
            apiMoveAttachmentDown.ErrorMessage = "No attachments were found.";
        }
    }

    return false;
}
```

The following example edits metadata of the unsorted attachment created by the first example on this page. It modifies its name, title and description.

```
private bool EditMetadata()
{
    // Create a new instance of the Tree provider
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get the example document
    TreeNode node = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/API-
Example", "en-us");

    if (node != null)
    {
        string where = "AttachmentIsUnsorted = 1";
        string orderBy = "AttachmentLastModified DESC";

        // Get the document's unsorted attachments with the latest on top
        DataSet attachments = DocumentHelper.GetAttachments(node, where, orderBy,
false, tree);

        if (!DataHelper.DataSourceIsEmpty(attachments))
        {
            // Create attachment info object from the first DataRow
            AttachmentInfo attachment = new AttachmentInfo(attachments.Tables[0]
```

```
.Rows[0]);

            // Edit its metadata
            attachment.AttachmentName += " - modified";
            attachment.AttachmentTitle += "Example title";
            attachment.AttachmentDescription += "This is an example of an unsorted
attachment.";

            // Ensure that the attachment can be updated without supplying its
binary data.
            attachment.AllowPartialUpdate = true;

            // Save the object into database
            AttachmentInfoProvider.SetAttachmentInfo(attachment);

            return true;
        }
        else
        {
            apiEditMetadata.ErrorMessage = "No attachments were found.";
        }
    }

    return false;
}
```

The following example deletes the two document attachments created by the first and second example on this page.

```
private bool DeleteAttachments()
{
    // Create a new instance of the Tree provider
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get the example document
    TreeNode node = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/API-
Example", "en-us");

    if (node != null)
    {
        // Get the document's unsorted attachments with the latest on top
        DataSet attachments = DocumentHelper.GetAttachments(node, null, null,
false, tree);

        if (!DataHelper.DataSourceIsEmpty(attachments))
        {
            foreach (DataRow attachmentRow in attachments.Tables[0].Rows)
            {
                // Create attachment info object from the first DataRow
                AttachmentInfo attachment = new AttachmentInfo(attachmentRow);

                // Delete the attachment
                DocumentHelper.DeleteAttachment(node, attachment.AttachmentGUID,
```

```
tree);
            }

            return true;
        }
        else
        {
            apiDeleteAttachments.ErrorMessage = "No attachments found.";
        }
    }

    return false;
}
```

4.18.4.4 Document relationships

4.18.4.4.1 Overview

These topics show examples of how the document relationships API can be used:

- Managing relationship names
- Managing relationships

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Documents\Relationships\Default.aspx.cs***.

The document relationships API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.GlobalHelper;
using CMS.UIControls;
using CMS.CMSHelper;
using CMS.SiteProvider;
using CMS.DocumentEngine;
```

4.18.4.4.2 Managing relationship names

The following example creates a relationship name.

```
private bool CreateRelationshipName()
```

```
{
    // Create new relationship name object
    RelationshipNameInfo newName = new RelationshipNameInfo();

    // Set the properties
    newName.RelationshipDisplayName = "My new relationship name";
    newName.RelationshipName = "MyNewRelationshipName";

    // Save the relationship name
    RelationshipNameInfoProvider.SetRelationshipNameInfo(newName);

    return true;
}
```

The following example gets and updates the relationship name created by the code example above.

```
private bool GetAndUpdateRelationshipName()
{
    // Get the relationship name
    RelationshipNameInfo updateName =
RelationshipNameInfoProvider.GetRelationshipNameInfo("MyNewRelationshipName");
    if (updateName != null)
    {
        // Update the properties
        updateName.RelationshipDisplayName =
updateName.RelationshipDisplayName.ToLower();

        // Save the changes
        RelationshipNameInfoProvider.SetRelationshipNameInfo(updateName);

        return true;
    }

    return false;
}
```

The following example gets multiple relationship names based on a WHERE condition and bulk updates them.

```
private bool GetAndBulkUpdateRelationshipNames()
{
    // Prepare the parameters
    string where = "RelationshipName LIKE N'MyNewRelationshipName%'";

    // Get the data
    DataSet names = RelationshipNameInfoProvider.GetRelationshipNames(where,
null);
    if (!DataHelper.DataSourceIsEmpty(names))
    {
        // Loop through the individual items
        foreach (DataRow nameDr in names.Tables[0].Rows)
```

```
        {
            // Create object from DataRow
            RelationshipNameInfo modifyName = new RelationshipNameInfo(nameDr);

            // Update the properties
            modifyName.RelationshipDisplayName =
modifyName.RelationshipDisplayName.ToUpper();

            // Save the changes
            RelationshipNameInfoProvider.SetRelationshipNameInfo(modifyName);
        }

        return true;
    }

    return false;
}
```

The following example adds the relationship name created by the first code example on this page to the current website

```
private bool AddRelationshipNameToSite()
{
    // Get the relationship name
    RelationshipNameInfo name =
RelationshipNameInfoProvider.GetRelationshipNameInfo("MyNewRelationshipName");
    if (name != null)
    {
        int nameId = name.RelationshipNameId;
        int siteId = CMSContext.CurrentSiteID;

        // Save the binding
        RelationshipNameSiteInfoProvider.AddRelationshipNameToSite(nameId,
siteId);

        return true;
    }

    return false;
}
```

The following example removes the relationship name added to the current website by the code example above from the

```
private bool RemoveRelationshipNameFromSite()
{
    // Get the relationship name
    RelationshipNameInfo removeName =
RelationshipNameInfoProvider.GetRelationshipNameInfo("MyNewRelationshipName");
    if (removeName != null)
    {
        int siteId = CMSContext.CurrentSiteID;
```

```
        // Get the binding
        RelationshipNameSiteInfo nameSite =
RelationshipNameSiteInfoProvider.GetRelationshipNameSiteInfo
(removeName.RelationshipNameId, siteId);

        // Delete the binding
        RelationshipNameSiteInfoProvider.DeleteRelationshipNameSiteInfo(nameSite);

        return true;
    }

    return false;
}
```

The following example deletes the relationship name created by the first code example on this page.

```
private bool DeleteRelationshipName()
{
    // Get the relationship name
    RelationshipNameInfo deleteName =
RelationshipNameInfoProvider.GetRelationshipNameInfo("MyNewRelationshipName");

    // Delete the relationship name
    RelationshipNameInfoProvider.DeleteRelationshipName(deleteName);

    return (deleteName != null);
}
```

4.18.4.4.3 Managing relationships

The following example creates a relationship between two documents using the relationship name created by the first code example on the previous page.

```
private bool CreateRelationship()
{
    // Get the relationship name
    RelationshipNameInfo relationshipName =
RelationshipNameInfoProvider.GetRelationshipNameInfo("MyNewRelationshipName");
    if (relationshipName != null)
    {
        // Get the tree structure
        TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

        // Get documents for relationship (the Root document is used for both in
this example)
        TreeNode root = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/",
null, true);

        // Create the relationship between documents
```

```
        RelationshipProvider.AddRelationship(root.NodeID, root.NodeID,
relationshipName.RelationshipNameId);

        return true;
    }

    return false;
}
```

The following example deletes the relationship created by the code example above.

```
private bool DeleteRelationship()
{
    // Get the relationship name
    RelationshipNameInfo relationshipName =
RelationshipNameInfoProvider.GetRelationshipNameInfo("MyNewRelationshipName");
    if (relationshipName != null)
    {
        // Get the tree structure
        TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

        // Get documents which are in relationship (the Root document is used for
both in this example)
        TreeNode root = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/",
null, true);

        // Delete the relationship between documents
        RelationshipProvider.RemoveRelationship(root.NodeID, root.NodeID,
relationshipName.RelationshipNameId);

        return true;
    }

    return false;
}
```

4.18.4.5 Workflow basics

4.18.4.5.1 Overview

These topics show examples of how the workflow API can be used:

- Sample document structure
- Creating documents
- Managing documents

> **Please note**
>
> All API examples can be simulated on your Kentico CMS website through the API

examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Documents\Workflow\Basics\Default.aspx.cs***.

The workflow basics API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.CMSHelper;
using CMS.GlobalHelper;
using CMS.SiteProvider;
using CMS.DocumentEngine;
using CMS.UIControls;
using CMS.WorkflowEngine;
```

4.18.4.5.2 Sample documents and objects

The following example creates a sample documents structure and workflow objects required for the API examples on the following pages to be executed.

```
private bool CreateExampleObjects()
{
    // Add a new culture to the current site
    CultureInfo culture = CultureInfoProvider.GetCultureInfo("de-de");
    CultureSiteInfoProvider.AddCultureToSite(culture.CultureID,
CMSContext.CurrentSiteID);

    // Create a new tree provider
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get the root node
    TreeNode parent = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/", "en-
us");

    if (parent != null)
    {
        // Create the API example folder
        TreeNode node = TreeNode.New("CMS.Folder", tree);

        node.DocumentName = "API Example";
        node.DocumentCulture = "en-us";

        // Insert it to database
        DocumentHelper.InsertDocument(node, parent, tree);

        parent = node;

        // Create the Source folder for moving
```

```
        node = TreeNode.New("CMS.Folder", tree);

        node.DocumentName = "Source";
        node.DocumentCulture = "en-us";

        DocumentHelper.InsertDocument(node, parent, tree);

        // Create the Target folder for moving
        node = TreeNode.New("CMS.Folder", tree);

        node.DocumentName = "Target";
        node.DocumentCulture = "en-us";

        DocumentHelper.InsertDocument(node, parent, tree);

        // Get the default workflow
        WorkflowInfo workflow = WorkflowInfoProvider.GetWorkflowInfo("default");

        if (workflow != null)
        {
            // Get the example folder data
            node = DocumentHelper.GetDocument(parent, tree);

            // Create new workflow scope
            WorkflowScopeInfo scope = new WorkflowScopeInfo();

            // Assign to the default workflow and current site and set starting
alias path to the example document
            scope.ScopeWorkflowID = workflow.WorkflowID;
            scope.ScopeStartingPath = node.NodeAliasPath;
            scope.ScopeSiteID = CMSContext.CurrentSiteID;

            // Save the scope into the database
            WorkflowScopeInfoProvider.SetWorkflowScopeInfo(scope);

            return true;
        }
        else
        {
            apiCreateExampleObjects.ErrorMessage = "The default workflow was not
found.";
        }
    }

    return false;
}
```

The following example deletes the sample document structure created by the example above, including the linked and copied documents created by executing the examples on the following pages.

```
private bool DeleteDocuments()
{
```

```
    // Create new tree provider
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Prepare parameters
    string siteName = CMSContext.CurrentSiteName;
    string aliasPath = "/API-Example";
    string culture = "en-us";
    bool combineWithDefaultCulture = false;
    string classNames = TreeProvider.ALL_CLASSNAMES;
    string where = null;
    string orderBy = null;
    int maxRelativeLevel = -1;
    bool selectOnlyPublished = false;
    string columns = null;

    // Get the example folder
    TreeNode node = DocumentHelper.GetDocument(siteName, aliasPath, culture,
combineWithDefaultCulture, classNames, where, orderBy, maxRelativeLevel,
selectOnlyPublished, columns, tree);

    if (node != null)
    {
        // Prepare delete parameters
        bool deleteAllCultures = true;
        bool destroyHistory = true;
        bool deleteProduct = false;

        // Delete all culture versions of the document and destroy its version
history. This method also destroys all child documents.
        DocumentHelper.DeleteDocument(node, tree, deleteAllCultures,
destroyHistory, deleteProduct);

        return true;
    }

    return false;
}
```

The following example deletes the sample workflow objects created by the first example on this page.

```
private bool DeleteObjects()
{
    CultureInfo culture = CultureInfoProvider.GetCultureInfo("de-de");

    // Remove the example culture from the site
    CultureSiteInfoProvider.RemoveCultureFromSite(culture.CultureID,
CMSContext.CurrentSiteID);

    // Prepare parameters
    string where = "ScopeStartingPath LIKE '/API-Example%'";
    string orderBy = null;
    int topN = 0;
```

```
    string columns = null;

    DataSet scopes = WorkflowScopeInfoProvider.GetWorkflowScopes(where, orderBy,
topN, columns);

    if (!DataHelper.DataSourceIsEmpty(scopes))
    {
        // Loop through all the scopes in case more identical scopes were
accidentally created
        foreach (DataRow scopeRow in scopes.Tables[0].Rows)
        {
            // Create scope info object
            WorkflowScopeInfo scope = new WorkflowScopeInfo(scopeRow);

            // Delete the scope
            scope.Delete();
        }

        return true;
    }

    return false;
}
```

4.18.4.5.3 Creating documents

The following example creates a new document under workflow.

```
private bool CreateDocument()
{
    // Create new tree provider
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Prepare parameters
    string siteName = CMSContext.CurrentSiteName;
    string aliasPath = "/API-Example";
    string culture = "en-us";
    bool combineWithDefaultCulture = false;
    string classNames = TreeProvider.ALL_CLASSNAMES;
    string where = null;
    string orderBy = null;
    int maxRelativeLevel = -1;
    bool selectOnlyPublished = false;
    string columns = null;

    // Get the example folder
    TreeNode parentNode = DocumentHelper.GetDocument(siteName, aliasPath, culture,
combineWithDefaultCulture, classNames, where, orderBy, maxRelativeLevel,
selectOnlyPublished, columns, tree);

    if (parentNode != null)
    {
```

```
        // Create a new node
        TreeNode node = TreeNode.New("CMS.MenuItem", tree);

        // Set the required document properties
        node.DocumentName = "My new document";
        node.DocumentCulture = "en-us";

        // Insert the document
        DocumentHelper.InsertDocument(node, parentNode, tree);

        return true;
    }

    return false;
}
```

The following example creates a new culture version of the document under workflow created by the example above.

```
private bool CreateNewCultureVersion()
{
    // Create an instance of the Tree provider
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Prepare parameters
    string siteName = CMSContext.CurrentSiteName;
    string aliasPath = "/API-Example/My-new-document";
    string culture = "en-us";
    bool combineWithDefaultCulture = false;
    string classNames = TreeProvider.ALL_CLASSNAMES;
    string where = null;
    string orderBy = null;
    int maxRelativeLevel = -1;
    bool selectOnlyPublished = false;
    string columns = null;

    // Get the document in the English culture
    TreeNode node = DocumentHelper.GetDocument(siteName, aliasPath, culture,
combineWithDefaultCulture, classNames, where, orderBy, maxRelativeLevel,
selectOnlyPublished, columns, tree);

    if (node != null)
    {
        // Translate the name of the document
        node.DocumentName = "My new translation";
        node.SetValue("MenuItemName", "My new translation");

        // Insert into database
        DocumentHelper.InsertNewCultureVersion(node, tree, "de-de");

        return true;
    }
```

```
    return false;
}
```

The following example creates a new linked document based on the document under workflow created by the first code example on this page.

```
private bool CreateLinkedDocument()
{
    // Create new tree provider
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Prepare parameters
    string siteName = CMSContext.CurrentSiteName;
    string aliasPath = "/API-Example";
    string culture = "en-us";
    bool combineWithDefaultCulture = false;
    string classNames = TreeProvider.ALL_CLASSNAMES;
    string where = null;
    string orderBy = null;
    int maxRelativeLevel = -1;
    bool selectOnlyPublished = false;
    string columns = null;

    // Get the example folder
    TreeNode parentNode = DocumentHelper.GetDocument(siteName, aliasPath, culture,
combineWithDefaultCulture, classNames, where, orderBy, maxRelativeLevel,
selectOnlyPublished, columns, tree);

    if (parentNode != null)
    {
        // Change the alias path
        aliasPath += "/My-new-document";

        // Get the original document
        TreeNode node = DocumentHelper.GetDocument(siteName, aliasPath, culture,
combineWithDefaultCulture, classNames, where, orderBy, maxRelativeLevel,
selectOnlyPublished, columns, tree);

        if (node != null)
        {
            // Insert the link
            DocumentHelper.InsertDocumentAsLink(node, parentNode.NodeID, tree);

            return true;
        }
        else
        {
            apiCreateLinkedDocument.ErrorMessage = "Document 'My new document' was
not found.";
        }
    }
```

```
    return false;
}
```

4.18.4.5.4 Managing documents

The following example gets and updates the document under workflow created by the first example in the Creating doc

```
private bool GetAndUpdateDocuments()
{
    // Create an instance of the Tree provider first
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Prepare parameters
    string siteName = CMSContext.CurrentSiteName;
    string aliasPath = "/API-Example/%";
    string culture = "en-us";
    bool combineWithDefaultCulture = false;
    string classNames = "CMS.MenuItem";
    string where = null;
    string orderBy = null;
    int maxRelativeLevel = -1;
    bool selectOnlyPublished = false;

    // Fill dataset with documents
    DataSet documents = DocumentHelper.GetDocuments(siteName, aliasPath, culture,
combineWithDefaultCulture, classNames, where, orderBy, maxRelativeLevel,
selectOnlyPublished, tree);

    if (!DataHelper.DataSourceIsEmpty(documents))
    {
        // Create a new Version manager instance
        VersionManager manager = VersionManager.GetInstance(tree);

        // Loop through all documents
        foreach (DataRow documentRow in documents.Tables[0].Rows)
        {
            // Create a new Tree node from the data row
            TreeNode editDocument = TreeNode.New(documentRow, "CMS.MenuItem",
tree);

            // Check out the document
            manager.CheckOut(editDocument);

            string newName = editDocument.DocumentName.ToLower();

            // Change document data
            editDocument.DocumentName = newName;

            // Change coupled data
            editDocument.SetValue("MenuItemName", newName);
```

```
            // Save the changes
            DocumentHelper.UpdateDocument(editDocument, tree);

            // Check in the document
            manager.CheckIn(editDocument, null, null);
        }

        return true;
    }

    return false;
}
```

The following example creates a copy of the document under workflow created by the first example in the Creating doc

```
private bool CopyDocument()
{
    // Create an instance of the Tree provider first
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Prepare parameters
    string siteName = CMSContext.CurrentSiteName;
    string aliasPath = "/API-Example/My-new-document";
    string culture = "en-us";
    bool combineWithDefaultCulture = false;
    string classNames = TreeProvider.ALL_CLASSNAMES;
    string where = null;
    string orderBy = null;
    int maxRelativeLevel = -1;
    bool selectOnlyPublished = false;
    string columns = null;

    // Get the example folder
    TreeNode node = DocumentHelper.GetDocument(siteName, aliasPath, culture,
combineWithDefaultCulture, classNames, where, orderBy, maxRelativeLevel,
selectOnlyPublished, columns, tree);

    aliasPath = "/API-Example/Source";

    // Get the new parent document
    TreeNode parentNode = DocumentHelper.GetDocument(siteName, aliasPath, culture,
combineWithDefaultCulture, classNames, where, orderBy, maxRelativeLevel,
selectOnlyPublished, columns, tree);

    if ((node != null) && (parentNode != null))
    {
        // Copy the document
        DocumentHelper.CopyDocument(node, parentNode.NodeID, false, tree);

        return true;
    }
```

```
    return false;
}
```

The following example moves the document under workflow created by the first example in the Creating documents topic to a different location.

```
private bool MoveDocument()
{
    // Create an instance of the Tree provider first
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Prepare parameters
    string siteName = CMSContext.CurrentSiteName;
    string aliasPath = "/API-Example/My-new-document";
    string culture = "en-us";
    bool combineWithDefaultCulture = false;
    string classNames = TreeProvider.ALL_CLASSNAMES;
    string where = null;
    string orderBy = null;
    int maxRelativeLevel = -1;
    bool selectOnlyPublished = false;
    string columns = null;

    // Get the example folder
    TreeNode node = DocumentHelper.GetDocument(siteName, aliasPath, culture,
combineWithDefaultCulture, classNames, where, orderBy, maxRelativeLevel,
selectOnlyPublished, columns, tree);

    aliasPath = "/API-Example/Target";

    // Get the new parent document
    TreeNode parentNode = DocumentHelper.GetDocument(siteName, aliasPath, culture,
combineWithDefaultCulture, classNames, where, orderBy, maxRelativeLevel,
selectOnlyPublished, columns, tree);

    if ((node != null) && (parentNode != null))
    {
        // Move the document
        DocumentHelper.MoveDocument(node, parentNode.NodeID, tree);

        return true;
    }

    return false;
}
```

4.18.4.6 Workflow advanced

4.18.4.6.1 Overview

These topics show advanced examples of how the workflow API can be used:

- Sample document structure
- Managing documents
- Workflow process
- Versioning

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Documents\Workflow\Advanced\Default.aspx.cs***.

The advanced workflow API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.CMSHelper;
using CMS.GlobalHelper;
using CMS.DocumentEngine;
using CMS.UIControls;
using CMS.WorkflowEngine;
```

4.18.4.6.2 Sample documents and objects

The following example creates a sample documents structure and workflow objects required for the API examples on the following pages to be executed.

```
private bool CreateExampleObjects()
{
    // Create a new tree provider
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get the root node
    TreeNode parent = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/", "en-
us");

    if (parent != null)
    {
        // Create the API document
        TreeNode node = TreeNode.New("CMS.MenuItem", tree);

        node.DocumentName = "API Example";
        node.DocumentCulture = "en-us";

        // Insert it to database
        DocumentHelper.InsertDocument(node, parent, tree);
```

```
        // Get the default workflow
        WorkflowInfo workflow = WorkflowInfoProvider.GetWorkflowInfo("default");

        if (workflow != null)
        {
            // Get the document data
            node = DocumentHelper.GetDocument(node, tree);

            // Create new workflow scope
            WorkflowScopeInfo scope = new WorkflowScopeInfo();

            // Assign to the default workflow and current site and set starting
alias path to the example document
            scope.ScopeWorkflowID = workflow.WorkflowID;
            scope.ScopeStartingPath = node.NodeAliasPath;
            scope.ScopeSiteID = CMSContext.CurrentSiteID;

            // Save the scope into the database
            WorkflowScopeInfoProvider.SetWorkflowScopeInfo(scope);

            // Create a new workflow step
            WorkflowStepInfo step = new WorkflowStepInfo();

            // Set its properties
            step.StepWorkflowID = workflow.WorkflowID;
            step.StepName = "MyNewWorkflowStep";
            step.StepDisplayName = "My new workflow step";
            step.StepOrder = 1;

            // Save the workflow step
            WorkflowStepInfoProvider.SetWorkflowStepInfo(step);

            // Ensure correct step order
            WorkflowStepInfoProvider.InitStepOrders(workflow);

            return true;
        }
        else
        {
            apiCreateExampleObjects.ErrorMessage = "The default workflow was not
found.";
        }
    }

    return false;
}
```

The following example deletes the sample document structure and objects created by the example above.

```
private bool DeleteExampleObjects()
```

```
{
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get the example document
    TreeNode node = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/API-
Example", "en-us");

    if (node != null)
    {
        // Delete the document
        DocumentHelper.DeleteDocument(node, tree, true, true, true);
    }

    string where = "ScopeStartingPath LIKE '/API-Example%'";

    // Get example workflow scopes
    DataSet scopes = WorkflowScopeInfoProvider.GetWorkflowScopes(where, null, 0,
null);

    if (!DataHelper.DataSourceIsEmpty(scopes))
    {
        // Loop through all the scopes in case more identical scopes were
accidentally created
        foreach (DataRow scopeRow in scopes.Tables[0].Rows)
        {
            // Create scope info object
            WorkflowScopeInfo scope = new WorkflowScopeInfo(scopeRow);

            // Delete the scope
            WorkflowScopeInfoProvider.DeleteWorkflowScopeInfo(scope);
        }
    }

    // Get the default workflow
    WorkflowInfo workflow = WorkflowInfoProvider.GetWorkflowInfo("default");

    if (workflow != null)
    {
        // Get the example step
        WorkflowStepInfo step = WorkflowStepInfoProvider.GetWorkflowStepInfo
("MyNewWorkflowStep", workflow.WorkflowID);

        if (step != null)
        {
            // Delete the step
            WorkflowStepInfoProvider.DeleteWorkflowStepInfo(step);
        }
    }

    return true;
}
```

4.18.4.6.3 Managing documents

The following example performs checkout of the sample document created by the example in the Sample documents and objects topic. Please note that check-in/check-out must be enabled for the respective workflow for this example to be functional.

```
private bool CheckOut()
{
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Prepare parameters
    string siteName = CMSContext.CurrentSiteName;
    string aliasPath = "/API-Example";
    string culture = "en-us";
    bool combineWithDefaultCulture = false;
    string classNames = TreeProvider.ALL_CLASSNAMES;
    string where = null;
    string orderBy = null;
    int maxRelativeLevel = -1;
    bool selectOnlyPublished = false;
    string columns = null;

    // Get the document
    TreeNode node = DocumentHelper.GetDocument(siteName, aliasPath, culture,
combineWithDefaultCulture, classNames, where, orderBy, maxRelativeLevel,
selectOnlyPublished, columns, tree);

    if (node != null)
    {
        // Create a new Workflow manager instance
        WorkflowManager workflowmanager = WorkflowManager.GetInstance(tree);

        // Make sure the document uses workflow
        WorkflowInfo workflow = workflowmanager.GetNodeWorkflow(node);

        if (workflow != null)
        {
            // Check if the workflow uses check-in/check-out functionality
            if (workflow.UseCheckInCheckOut(CMSContext.CurrentSiteName))
            {
                // The document has to be checked in
                if (!node.IsCheckedOut)
                {
                    // Create a new version manager instance
                    VersionManager versionmanager = VersionManager.GetInstance
(tree);

                    // Check out the document to create a new document version
                    versionmanager.CheckOut(node);

                    return true;
                }
                else
                {
```

```
                    apiCheckOut.ErrorMessage = "The document has already been
checked out.";
                }
            }
            else
            {
                apiCheckOut.ErrorMessage = "The workflow does not use check-in/
check-out. See the \"Edit document\" example, which checks the document out and in
automatically.";
            }
        }
        else
        {
            apiCheckOut.ErrorMessage = "The document doesn't use workflow.";
        }
    }

    return false;
}
```

The following example edits the document checked out by the previous example. If the document hasn't been checked out, it creates a new modified version of the document.

```
private bool EditDocument()
{
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Prepare parameters
    string siteName = CMSContext.CurrentSiteName;
    string aliasPath = "/API-Example";
    string culture = "en-us";
    bool combineWithDefaultCulture = false;
    string classNames = TreeProvider.ALL_CLASSNAMES;
    string where = null;
    string orderBy = null;
    int maxRelativeLevel = -1;
    bool selectOnlyPublished = false;
    string columns = null;

    // Get the document
    TreeNode node = DocumentHelper.GetDocument(siteName, aliasPath, culture,
combineWithDefaultCulture, classNames, where, orderBy, maxRelativeLevel,
selectOnlyPublished, columns, tree);

    if (node != null)
    {
        WorkflowManager workflowmanager = WorkflowManager.GetInstance(tree);

        // Make sure the document uses workflow
        WorkflowInfo workflow = workflowmanager.GetNodeWorkflow(node);

        if (workflow != null)
```

```
        {
            // Check if the workflow uses check-in/check-out
            bool autoCheck = !workflow.UseCheckInCheckOut
(CMSContext.CurrentSiteName);

            // Create a new version manager instance
            VersionManager versionmanager = VersionManager.GetInstance(tree);

            // If it does not use check-in/check-out, check out the document
automatically
            if (autoCheck)
            {
                versionmanager.CheckOut(node);
            }

            if (node.IsCheckedOut)
            {
                // Edit the last version of the document
                string newName = node.DocumentName.ToLower();

                node.DocumentName = newName;
                node.SetValue("MenuItemName", newName);

                // Save the document version
                DocumentHelper.UpdateDocument(node, tree);

                // Automatically check in
                if (autoCheck)
                {
                    versionmanager.CheckIn(node, null, null);
                }

                return true;
            }
            else
            {
                apiEditDocument.ErrorMessage = "The document hasn't been checked
out.";
            }
        }
        else
        {
            apiEditDocument.ErrorMessage = "The document doesn't use workflow.";
        }
    }

    return false;
}
```

Use the following example to check the document in. Please note that check-in/check-out must be enabled for the respective workflow for this example to be functional.

```
private bool CheckIn()
```

```
{
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Prepare parameters
    string siteName = CMSContext.CurrentSiteName;
    string aliasPath = "/API-Example";
    string culture = "en-us";
    bool combineWithDefaultCulture = false;
    string classNames = TreeProvider.ALL_CLASSNAMES;
    string where = null;
    string orderBy = null;
    int maxRelativeLevel = -1;
    bool selectOnlyPublished = false;
    string columns = null;

    // Get the document
    TreeNode node = DocumentHelper.GetDocument(siteName, aliasPath, culture,
combineWithDefaultCulture, classNames, where, orderBy, maxRelativeLevel,
selectOnlyPublished, columns, tree);

    if (node != null)
    {
        WorkflowManager workflowmanager = WorkflowManager.GetInstance(tree);

        // Make sure the document uses workflow
        WorkflowInfo workflow = workflowmanager.GetNodeWorkflow(node);

        if (workflow != null)
        {
            if (node.IsCheckedOut)
            {
                VersionManager versionmanager = VersionManager.GetInstance(tree);

                // Check in the document
                versionmanager.CheckIn(node, null, null);

                return true;
            }
            else
            {
                apiCheckIn.ErrorMessage = "The document hasn't been checked out.";
            }
        }
        else
        {
            apiCheckIn.ErrorMessage = "The document doesn't use workflow.";
        }
    }

    return false;
}
```

You can also check the document back in by reverting the check-out used above.

```
private bool UndoCheckout()
{
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Prepare parameters
    string siteName = CMSContext.CurrentSiteName;
    string aliasPath = "/API-Example";
    string culture = "en-us";
    bool combineWithDefaultCulture = false;
    string classNames = TreeProvider.ALL_CLASSNAMES;
    string where = null;
    string orderBy = null;
    int maxRelativeLevel = -1;
    bool selectOnlyPublished = false;
    string columns = null;

    // Get the document
    TreeNode node = DocumentHelper.GetDocument(siteName, aliasPath, culture,
combineWithDefaultCulture, classNames, where, orderBy, maxRelativeLevel,
selectOnlyPublished, columns, tree);

    if (node != null)
    {
        WorkflowManager workflowmanager = WorkflowManager.GetInstance(tree);

        // Make sure the document uses workflow
        WorkflowInfo workflow = workflowmanager.GetNodeWorkflow(node);

        if (workflow != null)
        {
            if (node.IsCheckedOut)
            {
                VersionManager versionmanager = VersionManager.GetInstance(tree);

                // Undo the checkout
                versionmanager.UndoCheckOut(node);

                return true;
            }
            else
            {
                apiUndoCheckout.ErrorMessage = "The document hasn't been checked
out.";
            }
        }
        else
        {
            apiUndoCheckout.ErrorMessage = "The document doesn't use workflow.";
        }
    }

    return false;
}
```

4.18.4.6.4 Workflow process

The following example moves the document under workflow created in the Sample documents and objects topic to the subsequent workflow step.

```
private bool MoveToNextStep()
{
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Prepare parameters
    string siteName = CMSContext.CurrentSiteName;
    string aliasPath = "/API-Example";
    string culture = "en-us";
    bool combineWithDefaultCulture = false;
    string classNames = TreeProvider.ALL_CLASSNAMES;
    string where = null;
    string orderBy = null;
    int maxRelativeLevel = -1;
    bool selectOnlyPublished = false;
    string columns = null;

    // Get the document
    TreeNode node = DocumentHelper.GetDocument(siteName, aliasPath, culture,
combineWithDefaultCulture, classNames, where, orderBy, maxRelativeLevel,
selectOnlyPublished, columns, tree);

    if (node != null)
    {
        WorkflowManager workflowManager = WorkflowManager.GetInstance(tree);

        WorkflowInfo workflow = workflowManager.GetNodeWorkflow(node);

        // Check if the document uses workflow
        if (workflow != null)
        {
            // Check if the workflow doesn't use automatic publishing, otherwise,
documents can't change workflow steps.
            if (!workflow.WorkflowAutoPublishChanges)
            {
                // Check if the current user can move the document to the next
step
                if (workflowManager.CheckStepPermissions(node,
WorkflowActionEnum.Approve))
                {
                    // Move the document to the next step
                    workflowManager.MoveToNextStep(node, null);

                    return true;
                }
                else
                {
                    apiMoveToNextStep.ErrorMessage = "You are not authorized to
approve the document.";
                }
```

```
            }
            else
            {
                apiMoveToNextStep.ErrorMessage = "The document uses versioning
without workflow, changes are published automatically.";
            }
        }
        else
        {
            apiMoveToNextStep.ErrorMessage = "The document doesn't use workflow.";
        }
    }

    return false;
}
```

The following example moves the document under workflow created in the Sample documents and objects topic to the previous workflow step.

```
private bool MoveToPreviousStep()
{
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Prepare parameters
    string siteName = CMSContext.CurrentSiteName;
    string aliasPath = "/API-Example";
    string culture = "en-us";
    bool combineWithDefaultCulture = false;
    string classNames = TreeProvider.ALL_CLASSNAMES;
    string where = null;
    string orderBy = null;
    int maxRelativeLevel = -1;
    bool selectOnlyPublished = false;
    string columns = null;

    // Get the document
    TreeNode node = DocumentHelper.GetDocument(siteName, aliasPath, culture,
combineWithDefaultCulture, classNames, where, orderBy, maxRelativeLevel,
selectOnlyPublished, columns, tree);

    if (node != null)
    {
        WorkflowManager workflowManager = WorkflowManager.GetInstance(tree);

        WorkflowInfo workflow = workflowManager.GetNodeWorkflow(node);

        // Check if the document uses workflow
        if (workflow != null)
        {
            // Check if the workflow doesn't use automatic publishing, otherwise,
documents can't change workflow steps.
            if (!workflow.WorkflowAutoPublishChanges)
```

```
            {
                // Check if the current user can move the document to the next
step
                if (workflowManager.CheckStepPermissions(node,
WorkflowActionEnum.Reject))
                {
                    // Move the document to the previous step
                    workflowManager.MoveToPreviousStep(node, null);

                    return true;
                }
                else
                {
                    apiMoveToPreviousStep.ErrorMessage = "You are not authorized
to reject the document.";
                }
            }
            else
            {
                apiMoveToPreviousStep.ErrorMessage = "The document uses versioning
without workflow, changes are published automatically.";
            }
        }
        else
        {
            apiMoveToPreviousStep.ErrorMessage = "The document doesn't use
workflow.";
        }
    }

    return false;
}
```

The following example publishes the document under workflow created in the Sample documents and objects topic.

```
private bool PublishDocument()
{
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Prepare parameters
    string siteName = CMSContext.CurrentSiteName;
    string aliasPath = "/API-Example";
    string culture = "en-us";
    bool combineWithDefaultCulture = false;
    string classNames = TreeProvider.ALL_CLASSNAMES;
    string where = null;
    string orderBy = null;
    int maxRelativeLevel = -1;
    bool selectOnlyPublished = false;
    string columns = null;

    // Get the document
```

```
    TreeNode node = DocumentHelper.GetDocument(siteName, aliasPath, culture,
combineWithDefaultCulture, classNames, where, orderBy, maxRelativeLevel,
selectOnlyPublished, columns, tree);

    if (node != null)
    {
        WorkflowManager workflowManager = WorkflowManager.GetInstance(tree);

        WorkflowInfo workflow = workflowManager.GetNodeWorkflow(node);

        // Check if the document uses workflow
        if (workflow != null)
        {
            // Publish the document
            workflowManager.PublishDocument(node, null);

            return true;
        }
        else
        {
            apiArchiveDocument.ErrorMessage = "The document doesn't use
workflow.";
        }
    }

    return false;
}
```

The following example archives the document under workflow created in the Sample documents and objects topic.

```
private bool ArchiveDocument()
{
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Prepare parameters
    string siteName = CMSContext.CurrentSiteName;
    string aliasPath = "/API-Example";
    string culture = "en-us";
    bool combineWithDefaultCulture = false;
    string classNames = TreeProvider.ALL_CLASSNAMES;
    string where = null;
    string orderBy = null;
    int maxRelativeLevel = -1;
    bool selectOnlyPublished = false;
    string columns = null;

    // Get the document
    TreeNode node = DocumentHelper.GetDocument(siteName, aliasPath, culture,
combineWithDefaultCulture, classNames, where, orderBy, maxRelativeLevel,
selectOnlyPublished, columns, tree);
```

```
    if (node != null)
    {
        WorkflowManager workflowManager = WorkflowManager.GetInstance(tree);

        WorkflowInfo workflow = workflowManager.GetNodeWorkflow(node);

        // Check if the document uses workflow
        if (workflow != null)
        {
            // Archive the document
            workflowManager.ArchiveDocument(node, null);

            return true;
        }
        else
        {
            apiArchiveDocument.ErrorMessage = "The document doesn't use
workflow.";
        }
    }

    return false;
}
```

4.18.4.6.5 Versioning

The following example roll back the document created in the Sample documents and objects topic back a specified version. Please note that the document needs to have at least one previous version for this example to be functional.

```
private bool RollbackVersion()
{
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Prepare parameters
    string siteName = CMSContext.CurrentSiteName;
    string aliasPath = "/API-Example";
    string culture = "en-us";
    bool combineWithDefaultCulture = false;
    string classNames = TreeProvider.ALL_CLASSNAMES;
    string where = null;
    string orderBy = null;
    int maxRelativeLevel = -1;
    bool selectOnlyPublished = false;
    string columns = null;

    // Get the document
    TreeNode node = DocumentHelper.GetDocument(siteName, aliasPath, culture,
combineWithDefaultCulture, classNames, where, orderBy, maxRelativeLevel,
selectOnlyPublished, columns, tree);

    if (node != null)
```

```
    {
        // Prepare the WHERE condition for the oldest document version
        where = "DocumentID = " + node.DocumentID;
        orderBy = "ModifiedWhen ASC";
        int topN = 1;

        // Get the version ID
        DataSet versionHistory = VersionHistoryInfoProvider.GetVersionHistories
(where, orderBy, topN, columns);

        if (!DataHelper.DataSourceIsEmpty(versionHistory))
        {
            // Create the Version history info object
            VersionHistoryInfo version = new VersionHistoryInfo
(versionHistory.Tables[0].Rows[0]);

            VersionManager versionManager = VersionManager.GetInstance(tree);

            // Roll back version
            versionManager.RollbackVersion(version.VersionHistoryID);

            return true;
        }
        else
        {
            apiRollbackVersion.ErrorMessage = "The document's version history is
empty.";
        }
    }

    return false;
}
```

The following example deletes the latest version of the document created in the Sample documents and objects topic. Please note that the document needs to have at least one version for the example to be functional.

```
private bool DeleteVersion()
{
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Prepare parameters
    string siteName = CMSContext.CurrentSiteName;
    string aliasPath = "/API-Example";
    string culture = "en-us";
    bool combineWithDefaultCulture = false;
    string classNames = TreeProvider.ALL_CLASSNAMES;
    string where = null;
    string orderBy = null;
    int maxRelativeLevel = -1;
    bool selectOnlyPublished = false;
    string columns = null;
```

```
    // Get the document
    TreeNode node = DocumentHelper.GetDocument(siteName, aliasPath, culture,
combineWithDefaultCulture, classNames, where, orderBy, maxRelativeLevel,
selectOnlyPublished, columns, tree);

    if (node != null)
    {
        // Prepare the WHERE condition for the latest document version
        where = "DocumentID = " + node.DocumentID;
        orderBy = "ModifiedWhen DESC";
        int topN = 1;

        // Get the version ID
        DataSet versionHistory = VersionHistoryInfoProvider.GetVersionHistories
(where, orderBy, topN, columns);

        if (!DataHelper.DataSourceIsEmpty(versionHistory))
        {
            // Create the Version history info object
            VersionHistoryInfo version = new VersionHistoryInfo
(versionHistory.Tables[0].Rows[0]);

            VersionManager versionManager = VersionManager.GetInstance(tree);

            // Delete the version
            versionManager.DestroyDocumentVersion(version.VersionHistoryID);

            return true;
        }
        else
        {
            apiDeleteVersion.ErrorMessage = "The document's version history is
empty.";
        }
    }

    return false;
}
```

The following example deletes the entire version history of the document created in the Sample documents and objects topic.

```
private bool DestroyHistory()
{
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Prepare parameters
    string siteName = CMSContext.CurrentSiteName;
    string aliasPath = "/API-Example";
    string culture = "en-us";
    bool combineWithDefaultCulture = false;
```

```
    string classNames = TreeProvider.ALL_CLASSNAMES;
    string where = null;
    string orderBy = null;
    int maxRelativeLevel = -1;
    bool selectOnlyPublished = false;
    string columns = null;

    // Get the document
    TreeNode node = DocumentHelper.GetDocument(siteName, aliasPath, culture,
combineWithDefaultCulture, classNames, where, orderBy, maxRelativeLevel,
selectOnlyPublished, columns, tree);

    if (node != null)
    {
        VersionManager versionManager = VersionManager.GetInstance(tree);

        // Destroy the version history
        versionManager.DestroyDocumentHistory(node.DocumentID);

        return true;
    }

    return false;
}
```

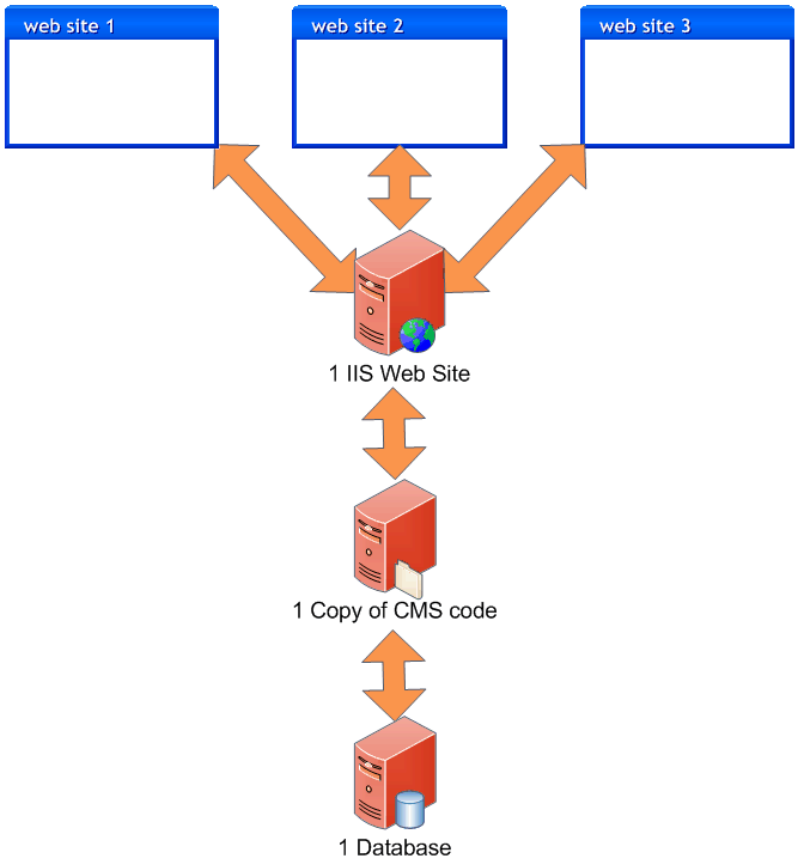
Part V

Managing sites

5 Managing sites

5.1 Site Management Overview

Kentico CMS allows you to manage multiple sites in a single installation. The database contains data for all websites and the websites are managed using a single administration interface (single copy of code). The following figure shows the multi-site configuration where one database and one copy of CMS code are used for multiple websites.



When you use a multi-site configuration, you can share:

- documents
- users
- global settings and system tables
- document types
- page templates
- web parts

This feature is useful if you need to create multiple websites for a single company and share users/ documents/settings between them.

When to choose separate installations

There are situations in which we recommend running separate instances of Kentico CMS for every website:

- You build websites with **many documents** and performance is critical for you.
- Your customers have very **different requirements** and you need to customize some common parts of the system, such as the administration interface or the structure of shared tables.
- Your customers are very sensitive to **security** and you do not want to risk that some other client will get access to other websites because of an administrator's mistake.

5.2 Managing sites

You can manage websites through the **Site Manager -> Sites** dialog. Please see New site wizard for more details on creating a new website. If you **Edit** () a specific site, you can configure its details on the following tabs:

General tab

Site display name	The name of the site displayed in the administration interface and in site selectors.
Site code name	The name of the site used in the code.
Site domain name	The main domain of the website. Enter the domain name without the *http://* protocol and *www* prefix. If you wish to use a different port than *80*, specify it as well. **Correct**: mycompany.com partners.mycompany.com mycompany.com:8080 **Incorrect**: http://mycompany.com www.mycompany.com
Default content culture	Default culture of the site's content. It can be changed using the **Change** button.
Visitor culture	Sets the content culture that should be displayed by default to visitors who do not have a preferred culture selected (the value of a browser cookie is checked). The *(Automatic)* option means that the culture will be decided based on the preferences configured in the visitor's browser. If the **Site Manager -> Settings -> URLs and SEO -> Force domain culture** setting is enabled, the culture specified here will always be selected and displayed when the site is accessed through its main domain name. All URLs of documents in the given culture will also be generated with the main domain name.

Site CSS stylesheet	Default CSS stylesheet used for all pages on the site unless they override the value with their own CSS stylesheet.
Editor CSS stylesheet	CSS stylesheet used for the WYSIWYG editor content.
Site description	Optional text description of the website for internal use.

Domain aliases tab

Here you can manage the website's domain aliases. Each alias represents a different domain name under which the site will be available. For example, if your website uses *mycompany.com* as its main domain and you also wish to assign the *my-company.co.uk* domain to the same website, you need to add this domain name to the list of domain aliases. This can be particularly useful for multilingual websites.

The following fields can be set when creating a new alias or editing () an existing one:

Domain alias	Specifies the domain name used for the alias. Like with the site's main domain name, include the port number if it is not *80*, but do not enter the *http://* protocol or *www* prefix.
Visitor culture	Sets the content culture that should be displayed by default to visitors who arrive on the website through a URL containing this domain alias. If the **Force domain culture** setting is enabled for the website, the culture specified here will always be selected and displayed when the site is accessed through the alias's domain name. All URLs of documents in the given culture will also be generated with the this domain name.
Default alias path	Used to select the default page (document) that should be displayed when the website is accessed through the alias. This overrides the **Site Manager -> Settings -> Content -> Default alias path** setting.
Redirect URL	Users will be redirected to the entered URL when they access the website through this domain alias. Both absolute and relative URLs are accepted. This field supports macro expressions. If the only purpose of your aliases is to make the site available under multiple domain names, it is recommended to redirect them to the website's main domain. This is a common Search engine optimization practice that prevents duplicate web content (i.e. having several URLs leading to the same content). **Example**: Imagine that the main domain of your website is *domain.com* and you have *domain.co.uk* defined as a domain alias with the **Redirect URL** field set to: *{%protocol%}://domain.com{%relativepath%}* These macros can be used to dynamically redirect users to the correct

	page under a different domain. **{%relativepath%}** - is resolved into the current relative URL path when the redirection takes place **{%protocol%}** - is resolved into the current URL scheme name (protocol) when the redirection takes place Now if a user accesses for example *http://domain.co.uk/example.aspx*, they will be redirected to *http://domain.com/example.aspx*.

Cultures tab

Here you can assign content cultures to the site, which can then be used to create translated versions of the website's documents. You will use this tab to configure websites that provide content in multiple languages. To learn more, please see the Content management -> Multilingual content chapter.

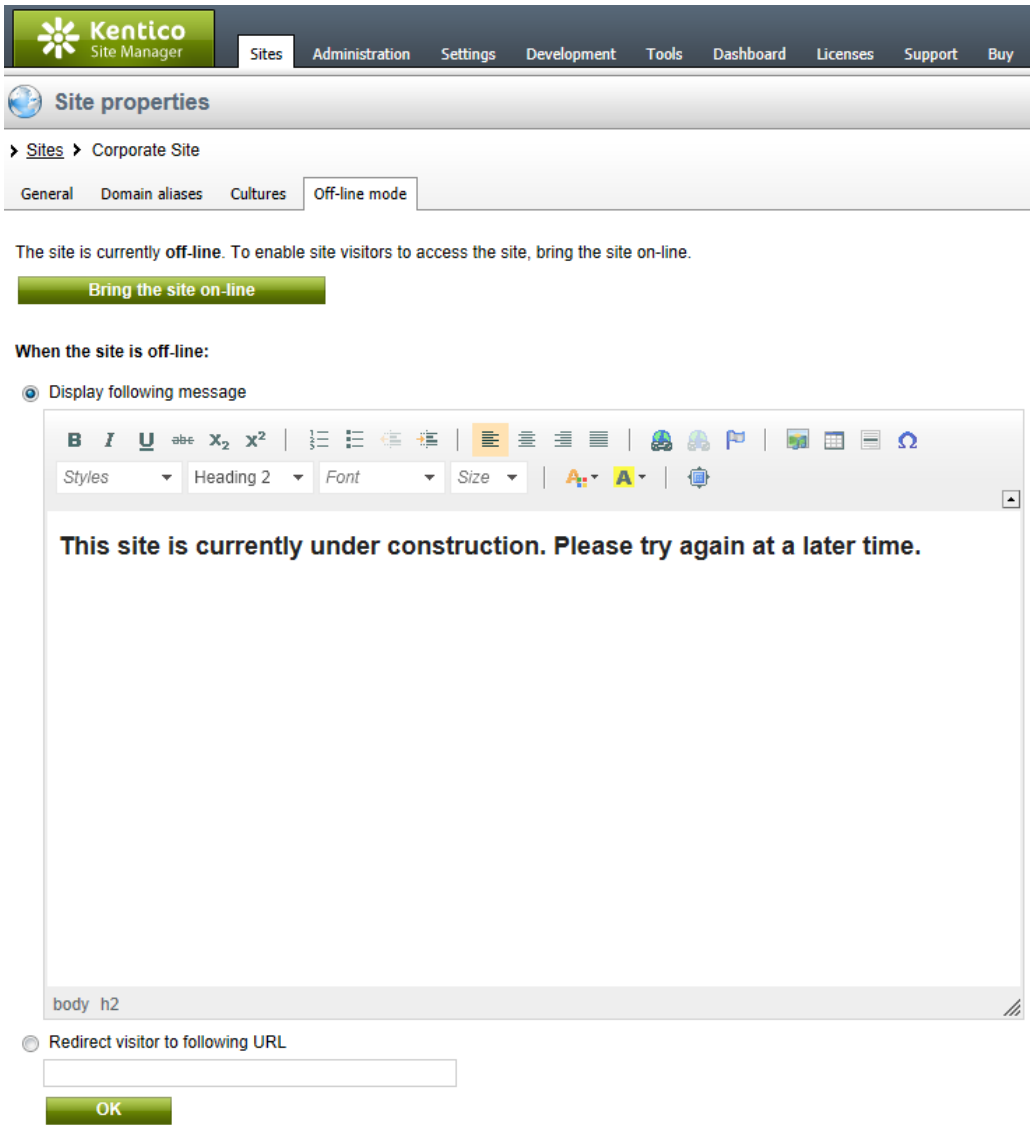
Off-line mode tab

Here you can make the currently edited site unavailable for regular users.

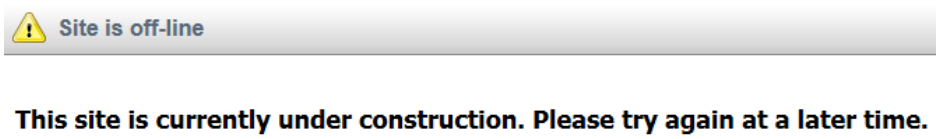
When in off-line mode, the live version of the website will not be accessible by visitors, but the site will still remain running and can be worked with normally through the **CMS Desk** or **Site Manager** interface by editors and administrators. This can be useful if you need to make major modifications to the site and wish to hide it while the update is in progress. To enable this mode, click the **Take the site off-line** button.

You can choose between two ways of handling users who attempt to access the site while it is off-line:

1. Select the **Display following message** option and add some content using the WYSIWYG editor:



Click **OK** to save the message and its content will be displayed to all visitors. The HTTP response status code of the page will be set to *503*, so applications and crawlers who access the site should be able to correctly recognize the situation.



2. Choose **Redirect visitor to following URL** and enter the address of an alternate website or page to which users will automatically be redirected until the site is returned back on-line.

At any time, you can allow visitors to access the site again by clicking the **Bring the site on-line** button.

5.3 Starting and stopping sites

You can run and stop websites using the **Start site** and **Stop site** buttons in the **Sites** dialog.

Switching between Sites on a Single Domain

If you try to run a site that uses the **same domain name (or alias)** as another site that is already running, you will get an error message and the site will not be started.

If you need to test **several websites on a single domain**, such as http://localhost, you need to specify the domain (localhost) for multiple websites and start only one of them.

If you cannot use your own domain names, you can use several alternatives that point to the same computer with different host name http://localhost, http://127.0.0.1 or http://mycomputer.

You can find more details on how to configure the websites in chapter Configuring multiple websites.

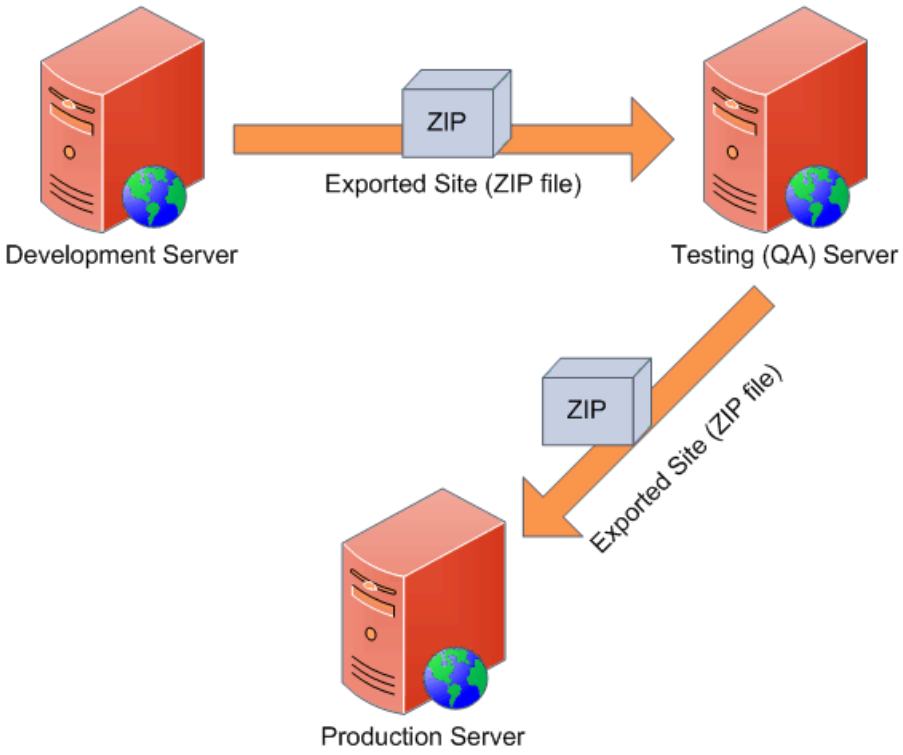
5.4 Creating a new site

Please see New site wizard.

5.5 Export and import

5.5.1 Overview

You can export and import website content and settings from one Kentico CMS instance to another. You can use this feature to move website or chosen objects between development, testing (QA) and production (live) server as illustrated in the figure below:



5.5.2 Folder structure and import/export

Kentico CMS uses a single folder structure, even if you manage multiple websites in a single installation. The following list describes the main folders and how they are affected during the import and export:

- **App_Browsers**
- **App_Code** (or **Old_App_Code** if you installed Kentico CMS as a web application)
- **-** CMSModules**<module_name>** (folders of custom modules)
- - **Global** (exports with any site, needs to be created manually; the folder is exported if the 'Export global folders' option is checked in Step 2 of the export process)
  - **<site code name>** (exports with given site, needs to be created manually; the folder is exported if the 'Export site folders' option is checked in Step 2 of the export process)
- **App_Data**
- **-** CMSModules**<module_name>** (folders of custom modules)
- **App_Themes**
  - **<stylesheet name>** (all folders related to stylesheets assigned to or used on the website)
- **App_WebReferences**
- **aspnet_client**
- **bin**
- **ClientBin**
- **CMSAdminControls**
- **CMSControlsExamples**
- **CMSDesk**
- **CMSFormControls** (all form controls selected in Step 2 of the export process are exported with any site)
- **CMSGlobalFiles** (exports with any website, needs to be created manually; folder is exported if the

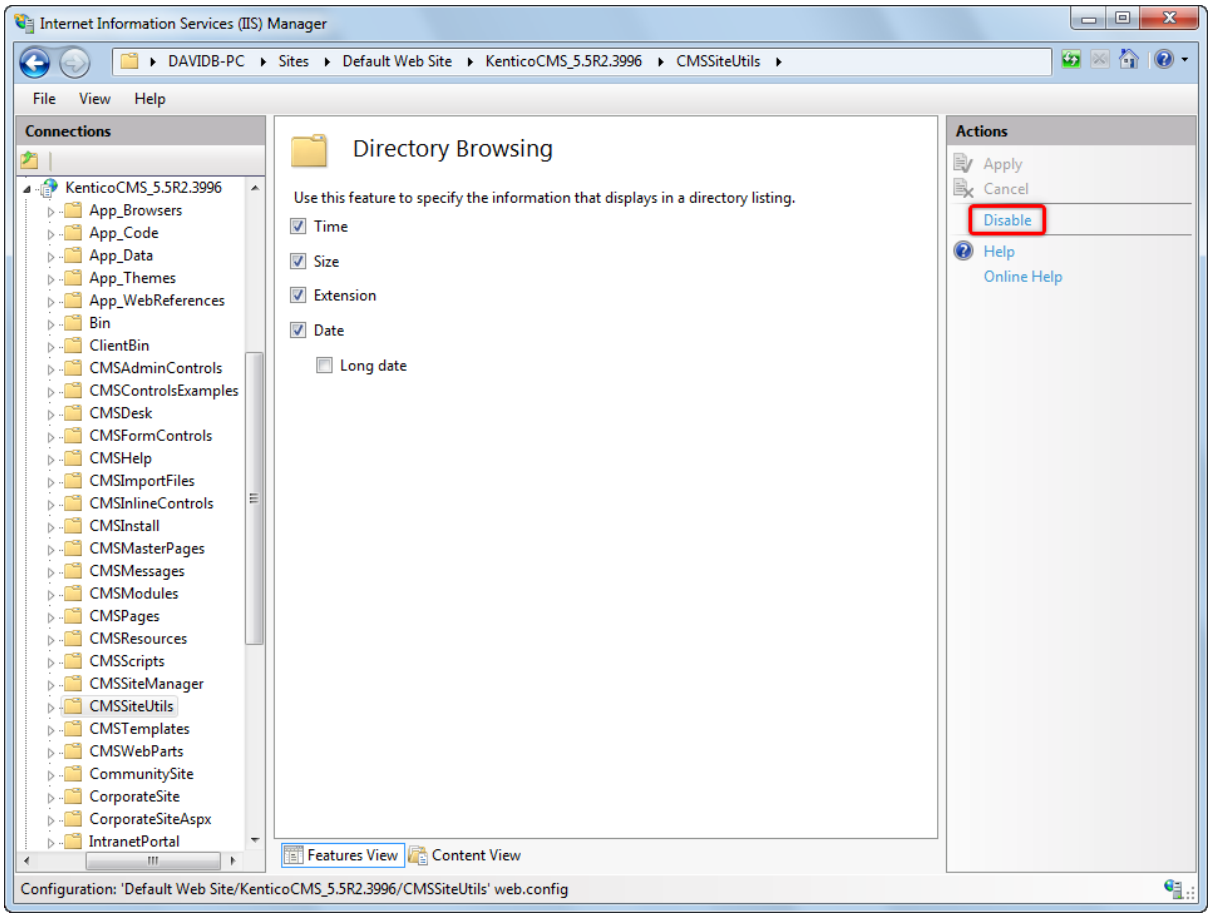
'Export global folders' option is checked in Step 2 of the export process)
- **CMSImportFiles**
- **CMSInlineControls** (all inline controls selected in Step 2 of the export process are exported with any site)
- **CMSInstall**
- **CMSMasterPages**
- **CMSMessages**
- **CMSModules**
- - Forums\Controls\Layouts**Custom** (forum custom layouts; exported with any website, needs to be created manually;folder is exported only if the 'Copy forum custom layouts folder' option is checked in Step 2 of the export process)
- - **<module name>** (folders of custom modules)
- **CMSPages**
- **CMSResources**
- **CMSScripts** (**CMSScripts\Custom** is a custom file folder and is part of the export package—the folder contains JavaScript files)
- **CMSSiteManager**
- **CMSSiteUtils**
- **CMSTemplates** (all files for selected ASPX page templates are exported with any site, page templates in other folders are exported as well if they are assigned/used in the given site;whole folder is exported if the 'Export ASPX templates folder' option is checked in Step 2 of the export process)
- **CMSWebParts** (all web parts selected in Step 2 of the export process are exported with any site; if the web part uses some additional files, they must be placed in a folder in format *<webpartCodeName>_files*; files that are not registered as web parts are not included in the export package)
- **<site code name>** (exports with given site, needs to be created manually or may be created automatically when storing files on the disk; the folder is exported if the 'Export site folders' option is checked in Step 2 of the export process)
  - Files (default folder for storing files if the system is configured for saving files on the disk)

Here's the explanation of colors:

- **red** - system folder, do not make changes or place your files here unless you want to modify the administration interface
- **blue** - folders for custom files, part of the export package
- **green** - folders for custom files, part of the export package, may need to be created manually
- **black** - service folders (import files, import/export)

Export/import package security

It is highly recommended to disable **Directory Browsing** in IIS for websites on live servers, at least for the **CMSSiteUtils** directory. If enabled, sensitive data from site export/import packages, such as user credentials, could be accessed directly from the browser.



5.5.3 Excluding files and folders from export

Files and folders that are exported into the **Files** folder of the export package can be filtered in order to be excluded from export. This is achieved by adding the following keys into the *AppSettings* section of your site's *web.config* file:

- **Excluding folders from export**: `<add key="CMSExportExcludedFolders" value="test*;cms*" />`
- **Excluding files from export**: `<add key="CMSExportExcludedFiles" value="test*;cms*" />`

Values of the keys define names of files/folders which will be excluded from export, i.e. will not be exported. Multiple values can be entered separated by semicolons.

You can also use the standard file system mask using the * wildcard, which substitutes any number of characters in the name. No other file system mask wildcards are supported.

If the keys are not entered in the *web.config* file, files with the **.scc** extension and folders with the **.svn** extension are excluded by default. However, if the keys are used, only the files and folders specified in their values are excluded.

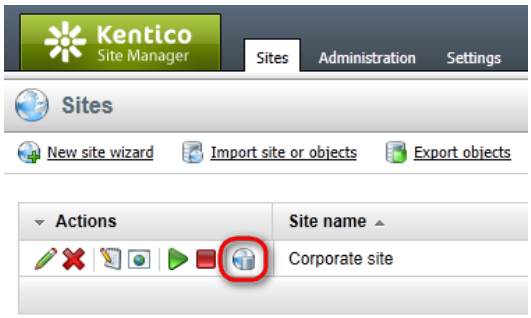
5.5.4 Export

5.5.4.1 Exporting a site

Kentico CMS allows you to export entire websites including their settings and related objects (such as document types, workflows, web parts, page templates, etc.) into a single file archive that can be imported on the same or different Kentico CMS instance.

Exporting a site

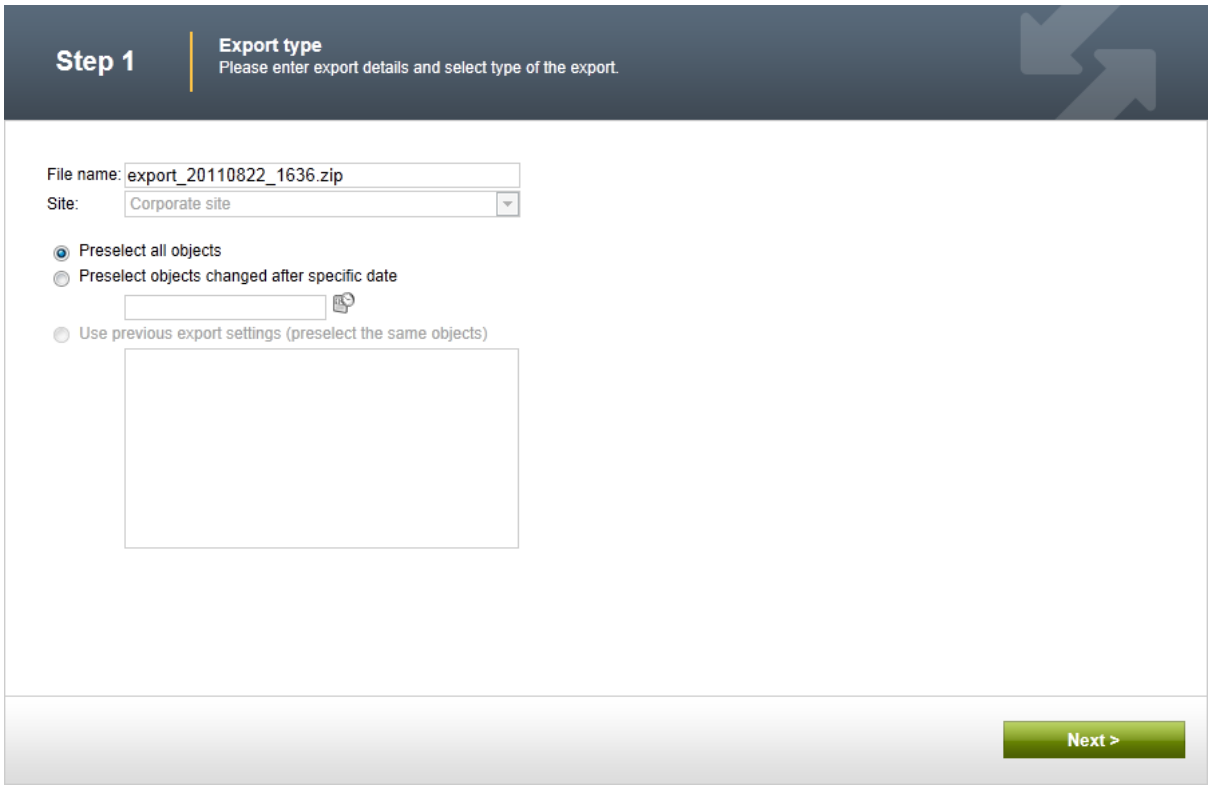
1. Go to **Site Manager -> Sites** and click the **Export site** icon next to the site you want to export. This starts the export wizard.



2. In the first step of the wizard, fill in the name of the export package and choose the object pre-selection type:

File name	Sets the name of the export package. The wizard creates the zip file containing the export data in the *~/CMSSiteUtils/Export* folder.
Preselect all objects	If chosen, all site objects will be preselected in the next step.
Preselect objects changed after specific date	If chosen, only objects changed after the specified date will be selected in the next step.
Use previous export settings	If chosen, settings used in a previous export selected from the list below will be used.

Select the option that suits your purposes and click **Next**.

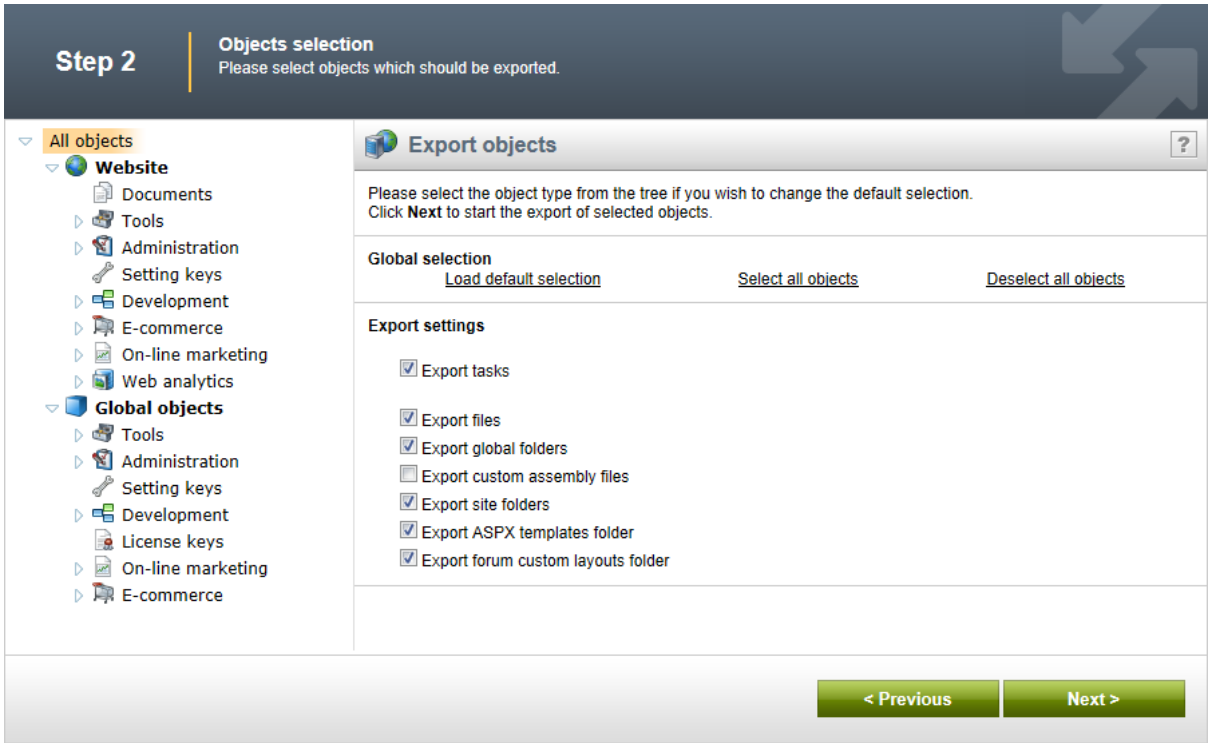


3. In **Step 2**, you can select which objects will be exported. The tree on the left represents object categories. These reflect the sections of the administration interface under which the objects can be found. The **Site** category contains objects related to the selected website. The **Global objects** category contains global objects that can be used by all sites. Selecting a category opens a set of check boxes in the right part of the screen, allowing you to select which objects will be exported.

Select the root of the tree (*All objects*) to access the following general export settings:

Global selection	
Load default selection	Preselects all objects based on your choices in Step 1 of the import wizard.
Select all objects	Selects all available objects.
Deselect all objects	Unselects all objects.
Export settings	
Export tasks	If enabled, delete tasks (incremental deployment) will be included in the package.
Export files	Some database objects are linked with physical files stored on the file system inside the web project. If you check this option, such files will be exported along with the corresponding database objects.
Export global folders	If enabled, all files under the following folders will be added to the export package:

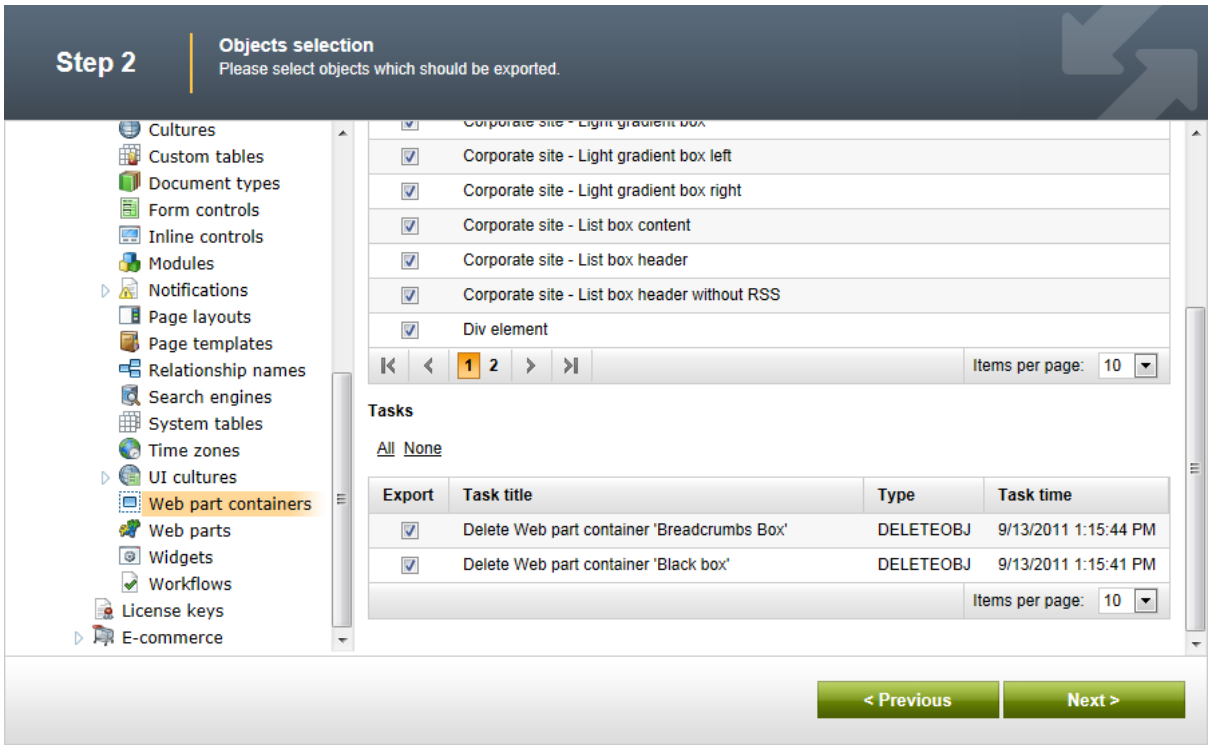
	• ~/App_Code/Global/ • ~/CMSGlobalFiles/ • ~/CMSScripts/Custom/ • ~/App_Code/Controllers/ • ~/Controllers/Global/ • ~/Views/Global/
Export custom assembly files	If enabled, bound custom assembly files will be exported together with notification gateways, payment options, integration connectors, scheduled tasks and smart search indexes. Custom assemblies are those whose names do not begin with the *CMS.* prefix.
Export site folders	This option is only available when exporting websites, not just separate objects. If enabled, all files under the following site-related folders will be added to the export package: • ~/App_Code/<site code name>/ • ~/App_Data/<site code name>/ • ~/<site code name>/ • ~/Controllers/<site code name>/ • ~/Views/<site code name>/
Export ASPX templates folder	If enabled, the folder with ASPX page templates will be exported: • *~/CMSTemplates/*
Export forum custom layouts folder	If enabled, folder with custom forum layouts will be exported: • *~/CMSModules/Forums/Controls/Layouts/Custom/*



4. The following categories contain extra options to be set:

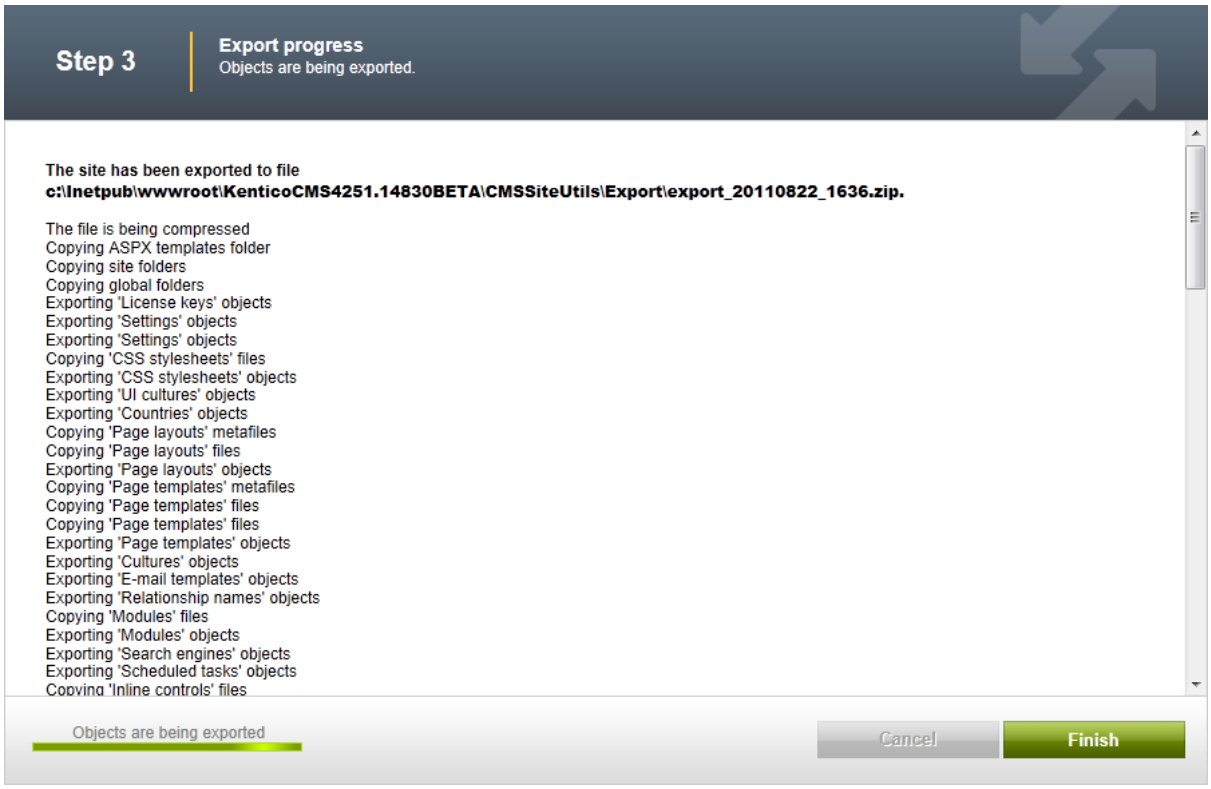
Custom tables	
Export custom table data	If checked, custom table records (the actual data stored in the tables) will be exported along with the selected custom tables.
Documents	
Export documents	If checked, documents will be exported
Export document histories	If checked, histories of all exported documents will be exported.
Export document relationships	If checked, relationships of all exported documents will be exported.
Export document level permissions	If checked, document security settings made in CMS Desk will be exported.
Export blog comments	If checked, blog comments will be exported.
Export event attendees	If checked, event attendees will be exported for all exported events.
Forms	
Export forms data	If checked, stored forms' data will be exported together with the exported forms.
Export physical files	If checked, physical files saved within form records (if there are some) will also be imported.
Forums	
Export forum posts	If checked, forum posts will be exported together with the exported forums.
Message boards	
Export board messages	If checked, board messages will be exported together with particular message boards.
Media libraries	
Export media files	If checked, media files stored in the database will be exported.
Export physical files	If checked, physical media files stored in the file system will be exported. This option is not selected by default as it may cause the package size grow extremely large; instead, it is recommended to export these files manually.

5. If you have the **Log export tasks** option enabled in **Site Manager -> Settings -> Versioning & synchronization -> Staging**, a list of object deletion tasks may also be displayed at the bottom of the list. This happens when some objects have been deleted (just as the two web part containers in the screenshot below). If you leave the check-boxes checked, the objects will be deleted after importing the package on the target server.



6. After making all required selections, click **Next** to proceed to the next step. A log appears, showing you the progress of exporting. You can abort exporting by clicking **Cancel** at any time. When exporting finishes, a message appears at the top of the log, telling you the full path to the exported file.

Click **Finish**. You will be redirected back to **Site manager** -> **Sites**.



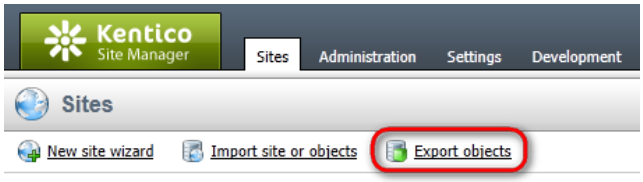
Now you can copy the exported package into the **~/CMSSiteUtils/Import** folder of the target installation of Kentico CMS, and use the **Import site or objects** wizard described in the Importing a site or objects topic.

5.5.4.2 Exporting objects

Besides exporting the whole website, you can also choose to export only selected objects (web parts, document types, page templates, etc.). This is useful when you update some object on the development machine and want to copy the updated object to the staging or production server.

Exporting selected objects

1. Go to **Site Manager -> Sites** and click **Export objects**. The Export objects wizard starts.

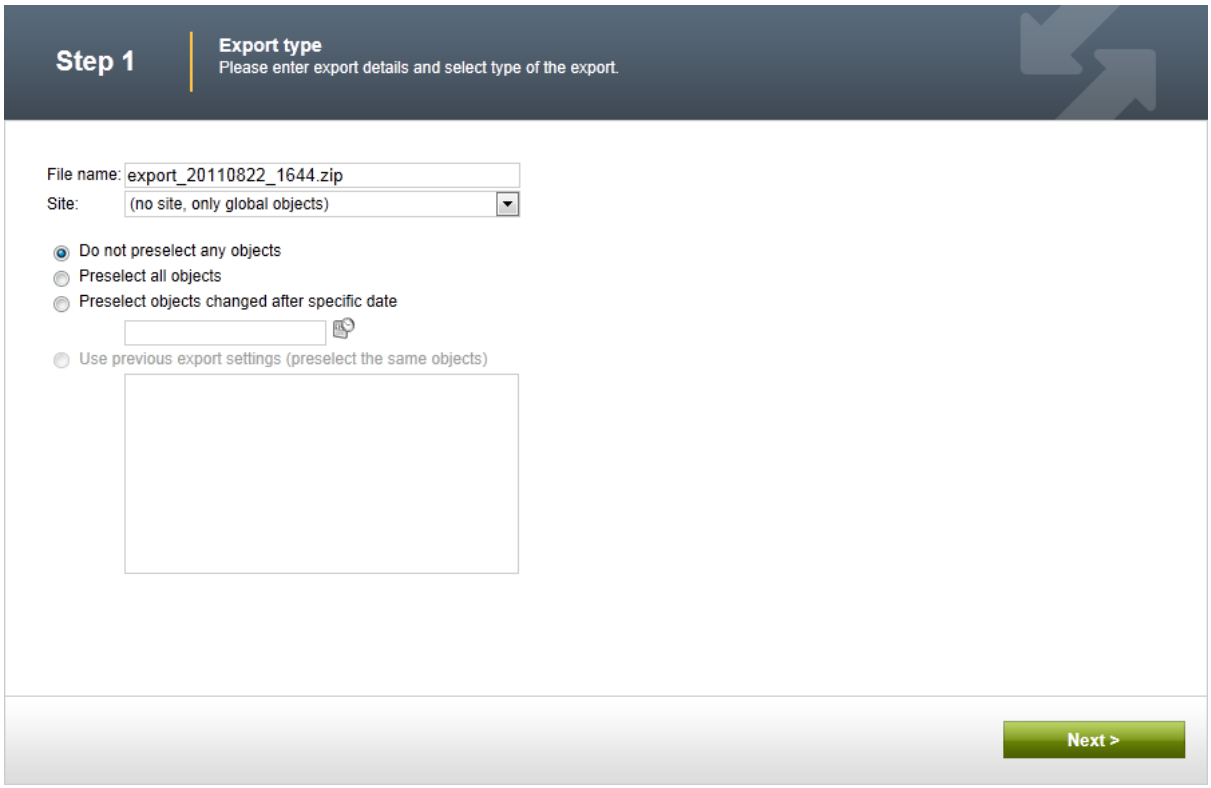


2. In the first step of the wizard, fill in the name of the export package and choose the object pre-selection type:

File name	Sets the name of the export package. The wizard creates the zip file containing the export data in the *~/CMSSiteUtils/Export* folder.

Do not preselect any objects	If chosen, no objects will be preselected in the next step.
Preselect all objects	If chosen, all objects will be preselected in the next step.
Preselect objects changed after specific date	If chosen, only objects changed after the specified date will be selected in the next step.
Use previous export settings	If chosen, settings used in a previous export selected from the list below will be used.

Select the option that suits your purposes and click **Next**.

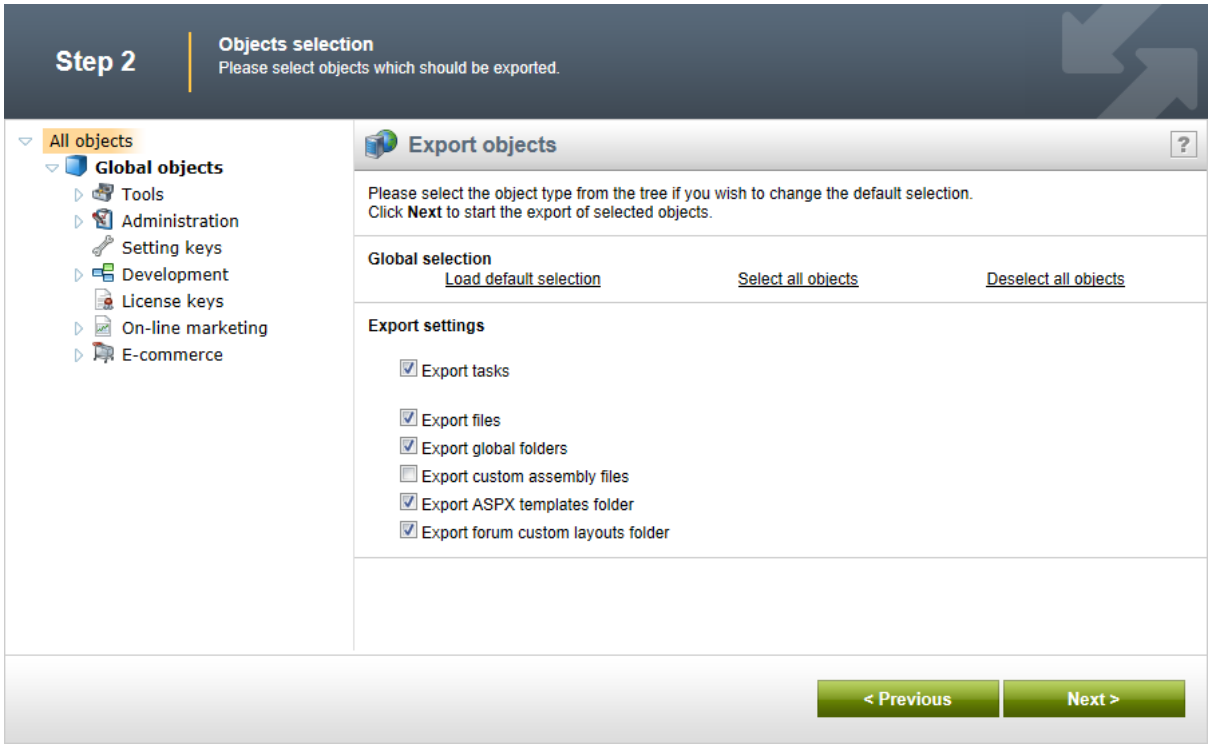


3. In **Step 2**, you can select which objects will be exported. The tree on the left represents object categories. These reflect the sections of the administration interface under which the objects can be found. Selecting a category opens a set of check boxes in the right part of the screen, allowing you to select which objects will be exported.

Select the root of the tree (*All objects*) to access the following general export settings:

Global selection	
Load default selection	Preselects all objects based on your choices in Step 1 of the import wizard.
Select all objects	Selects all available objects.
Deselect all objects	Unselects all objects.

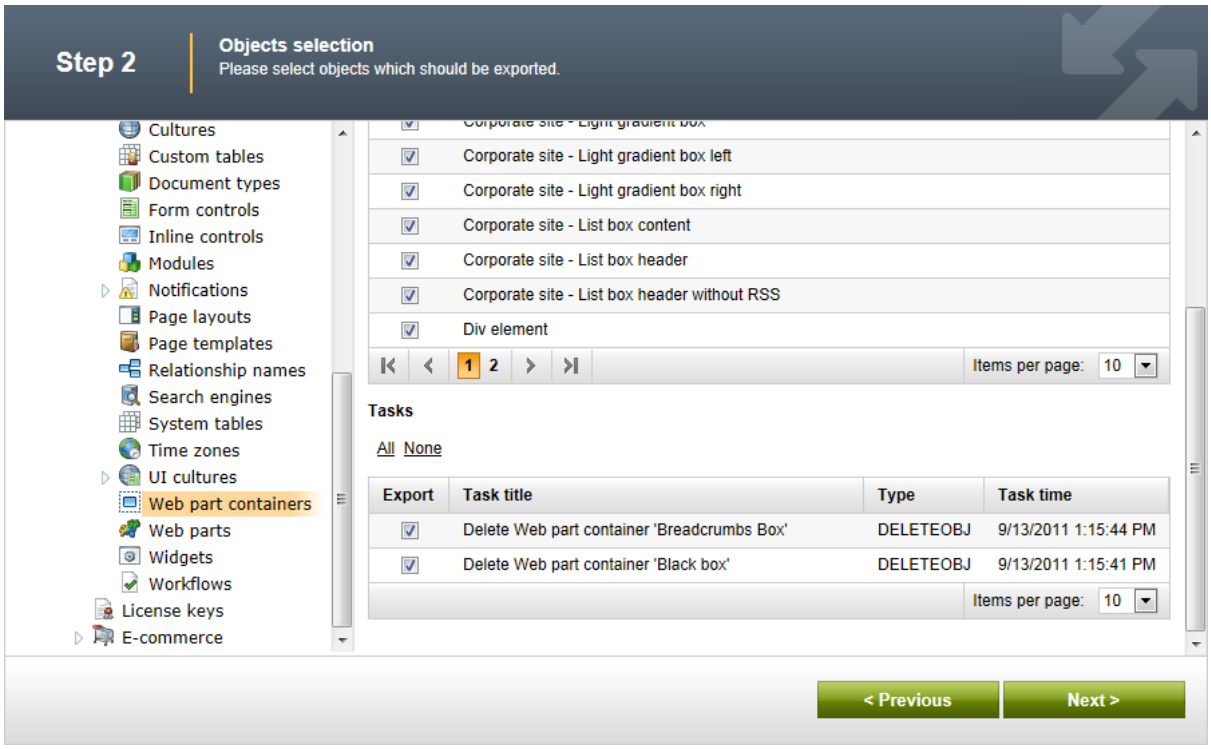
Export settings	
Export tasks	If enabled, delete tasks (incremental deployment) will be included in the package.
Export files	Some database objects are linked with physical files stored on the file system inside the web project. If you check this option, such files will be exported along with the corresponding database objects.
Export global folders	If enabled, all files under the following folders will be added to the export package: • ~/App_Code/Global/ • ~/CMSGlobalFiles/ • ~/CMSScripts/Custom/ • ~/App_Code/Controllers/ • ~/Controllers/Global/ • ~/Views/Global/
Export custom assembly files	If enabled, bound custom assembly files will be exported together with notification gateways, payment options, integration connectors, scheduled tasks and smart search indexes. Custom assemblies are those whose names do not begin with the *CMS.* prefix.
Export ASPX templates folder	If enabled, the folder with ASPX page templates will be exported: • *~/CMSTemplates/*
Export forum custom layouts folder	If enabled, folder with custom forum layouts will be exported: • *~/CMSModules/Forums/Controls/Layouts/Custom/*



4. The following categories contain extra options to be set:

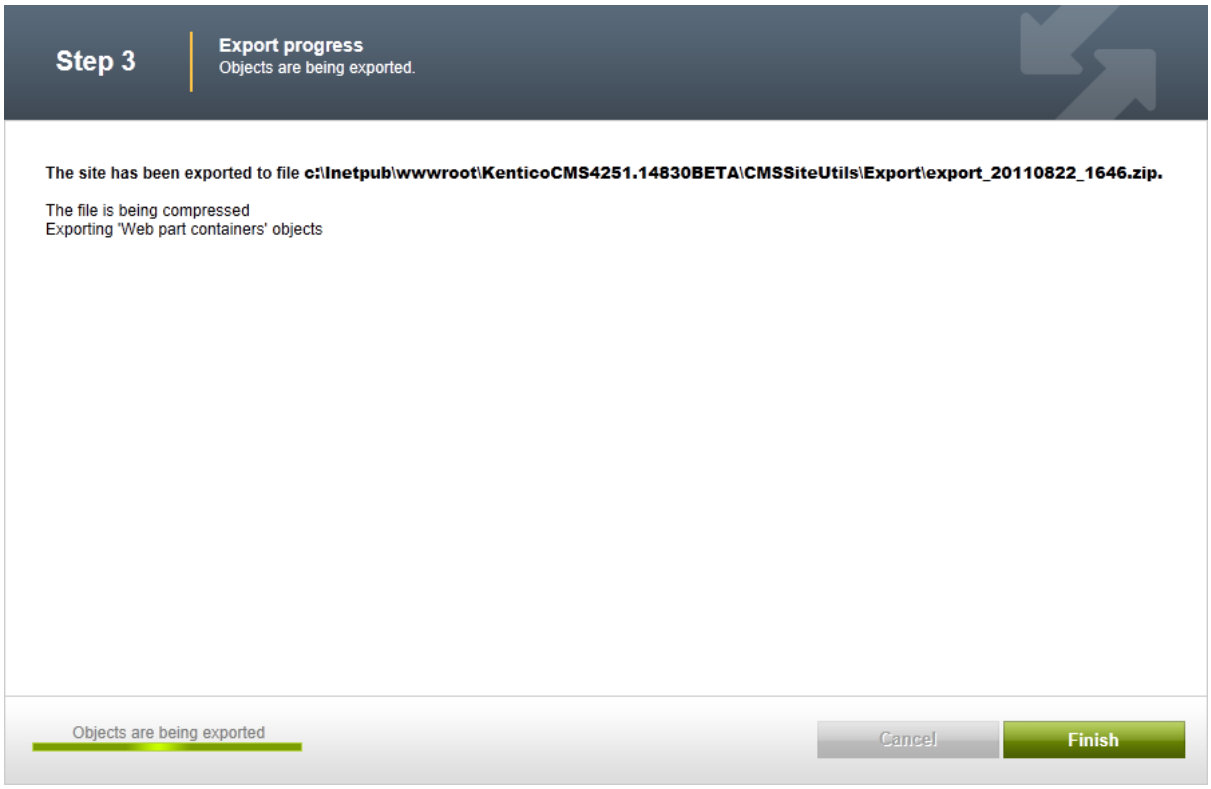
Custom tables	
Export custom table data	If checked, custom table records (the actual data stored in the tables) will be exported along with the selected custom tables.
Documents	
Export documents	If checked, documents will be exported
Export document histories	If checked, histories of all exported documents will be exported.
Export document relationships	If checked, relationships of all exported documents will be exported.
Export document level permissions	If checked, document security settings made in CMS Desk will be exported.
Export blog comments	If checked, blog comments will be exported.
Export event attendees	If checked, event attendees will be exported for all exported events.
Forms	
Export forms data	If checked, stored forms' data will be exported together with the exported forms.
Export physical files	If checked, physical files saved within form records (if there are some) will also be imported.
Forums	
Export forum posts	If checked, forum posts will be exported together with the exported forums.
Message boards	
Export board messages	If checked, board messages will be exported together with particular message boards.
Media libraries	
Export media files	If checked, media files stored in the database will be exported.
Export physical files	If checked, physical media files stored in the file system will be exported. This option is not selected by default as it may cause the package size grow extremely large; instead, it is recommended to export these files manually.

5. If you have the "**Log export tasks**" option enabled in **Site Manager -> Settings -> Versioning & synchronization -> Staging**, a list of object deletion tasks may also be displayed at the bottom of the list. This happens when some objects have been deleted (just as the two web part containers in the screenshot below). If you leave the check-boxes checked, the objects will be deleted after importing the package on the target server.



6. After making all required selections, click **Next** to proceed to the next step. A log appears, showing you the progress of exporting. You can abort exporting by clicking **Cancel** at any time. When exporting finishes, a message appears at the top of the log, telling you the full path to the exported file.

Click **Finish**. You will be redirected back to **Site manager** -> **Sites**.



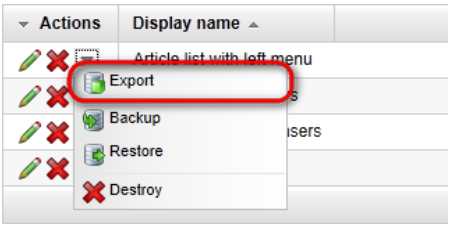
Now you can copy the exported file into the **~/CMSSiteUtils/Import** folder of the target installation of Kentico CMS, and use the **Import site or objects** wizard described in the Importing a site or objects topic.

5.5.4.3 Exporting single objects

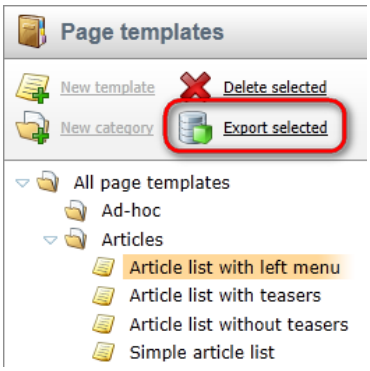
Some objects can also be exported separately as single object packages. This is useful when you want to quickly export only one or few objects apart from the rest of all other objects.

Single object export is supported for: CSS stylesheets, Document types, E-mail templates, Form controls, Inline controls, Page layouts, Page templates, Web part containers, Web parts, Workflow schemas.

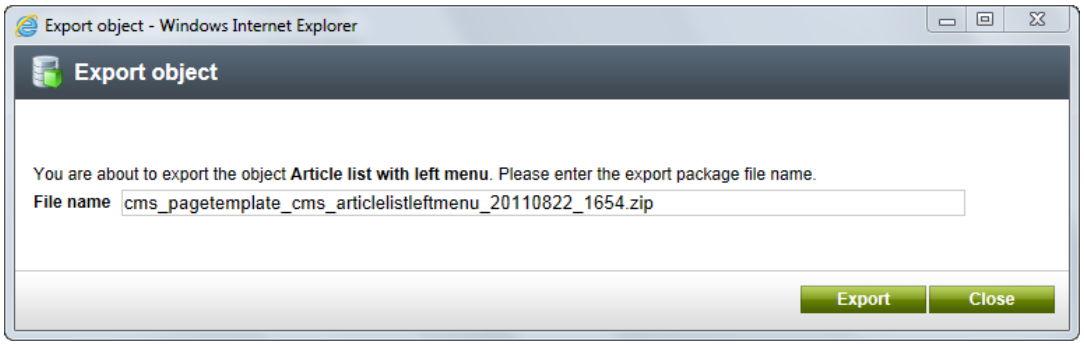
These objects can be found in the corresponding sections in **Site manager -> Development** or **Administration**. You can export an object by clicking the **Export object** ( ) icon.



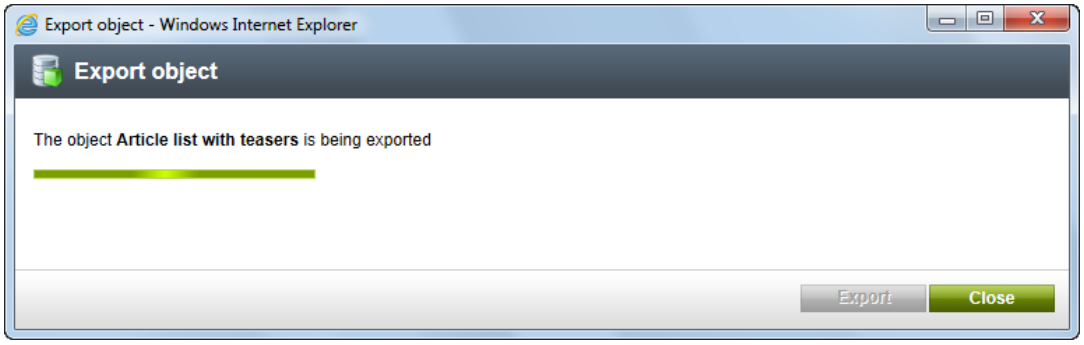
Or **Export selected** in case of Page templates and Web parts.



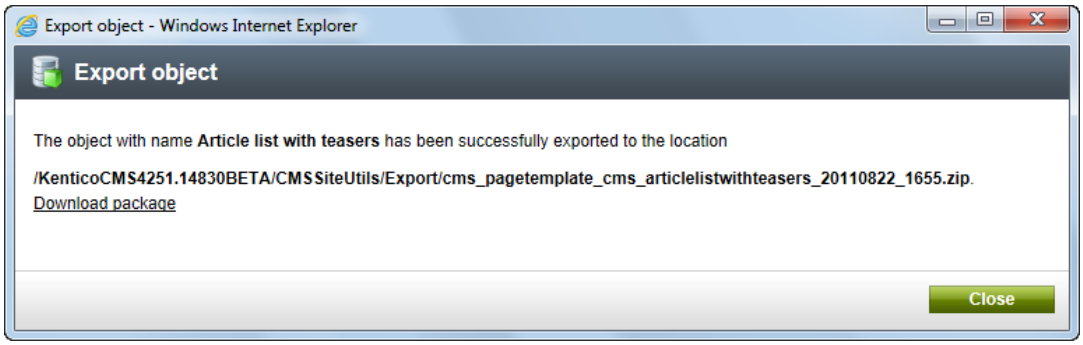
The following window will appear. Enter the name of the export file (default name will be pre-entered) and click **OK**.



Now wait while the object is being exported.



When the exporting is successfully finished, the following message with path to the exported file will be displayed. The **Download object** link below can be used for storing the file into a different location. Click it to open the typical file download dialog of your web browser.



Close the window by clicking **Close**.

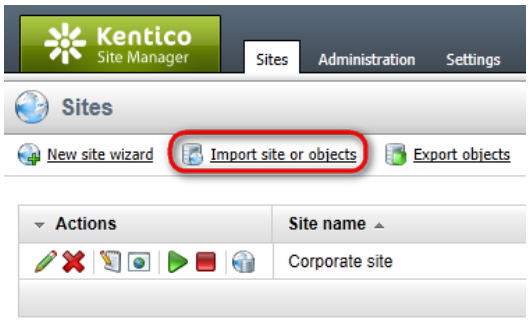
5.5.5 Import

5.5.5.1 Importing a site or objects

After you export a site or objects using the **Export site wizard** or **Export objects wizard**, you can import the content using the **Import site or objects wizard**. Before you start the wizard, you need to copy your export packages into the application's **~/CMSSiteUtils/Import** folder.

Importing a site

1. Go to **Site Manager -> Sites** and click **Import site or objects**.



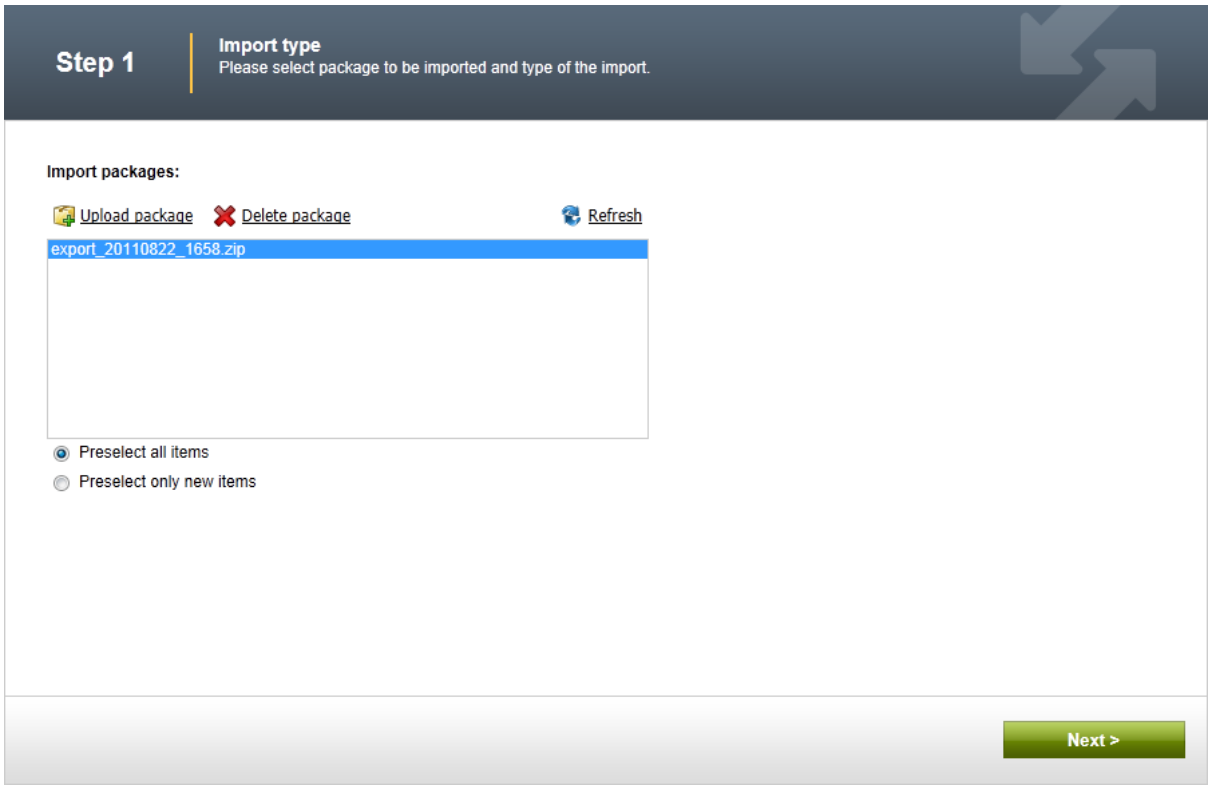
2. In the first step of the wizard, you need to choose the import package that you want to import. Packages that are already stored in the in *~/CMSSiteUtils/Import* folder are listed in the **Import packages** listbox.

You can upload additional packages to the folder (and hence add them to the list) by clicking **Upload package**. You can also add the package to the folder manually and then click **Refresh** (). Packages can be deleted (both from the list and from the file system) by clicking **Delete package**.

With a package selected, you can choose from the following options:

Preselect all items	If selected, all items in the package will be preselected in the next step.
Preselect only new items	If selected, only items that are not already in the database will be preselected.

Make the desired selection and proceed to the next step by clicking **Next**.

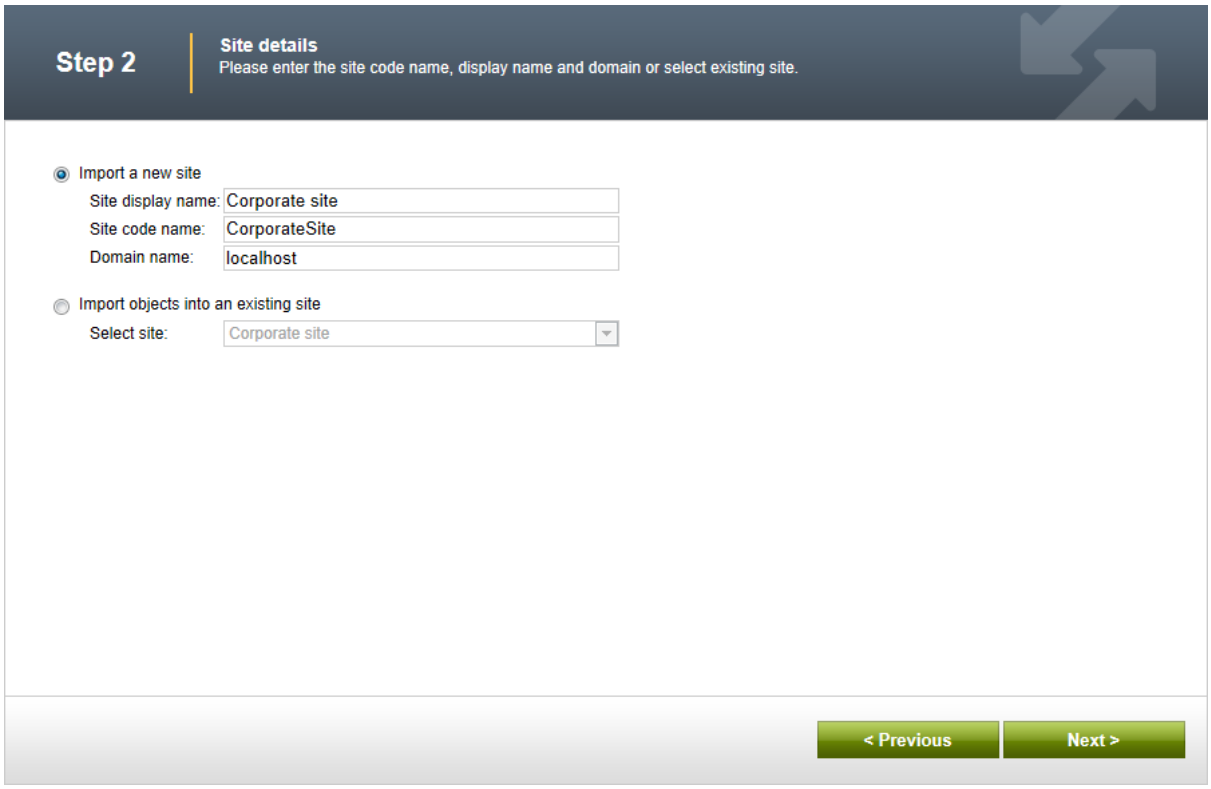


3. **Step 2** appears only when importing a site package. If you are importing a package containing only global objects, Step 3 is displayed instead.

You have the following two options in Step 2:

Import a new site	A new site will be created based on the contents of the package. You need to enter the site's display name, code name and domain name.
Import objects into an existing site	The wizard imports the content of the package into the selected site.

Click **Next**.



4. In **Step 3**, select which of the objects from the package will be imported.

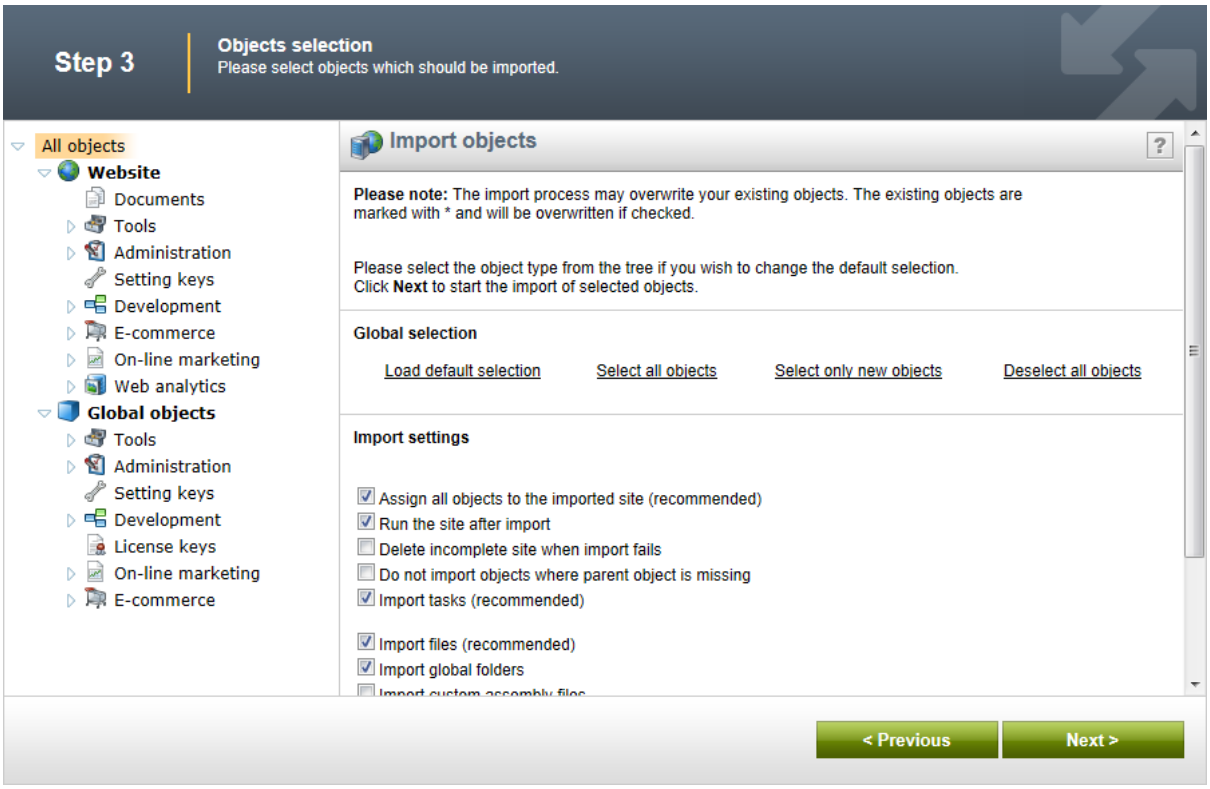
The tree on the left represents object categories. These reflect the sections of the administration interface under which the objects can be found. Selecting a category opens a set of check boxes in the right part of the screen, allowing you to choose which objects will be imported. Objects that already exist on the target server are marked with an asterisk (*). If you import an existing object, the new version overwrites the current one.

Select the root of the tree (*All objects*) to access the following import settings:

Global selection	
Load default selection	Preselects all objects based on your choices in Step 1 of the import wizard.
Select all objects	Selects all objects in the import package.
Select only new objects	Selects only those objects that do not exist in the target database (existing objects are unselected).
Deselect all objects	Unselects all objects in the import package.
Import settings	
Update site definition	Displayed only when importing to an existing site. If enabled, all settings stored as part of the site object will be updated with those contained in the package. These settings are stored in the *Site\cms_site.xml* file inside the export package.

Assign all objects to the imported site (recommended)	If enabled, all imported site-related objects will be assigned to the imported site.
Run the site after import	If enabled, the imported site automatically starts once the import process is complete.
Delete incomplete site when import fails	If enabled, partially imported sites are automatically deleted if the import process fails.
Do not import objects where parent object is missing	If enabled, the import process skips any child objects whose parent objects are not found in the import package.
Import tasks (recommend)	If enabled, delete tasks (incremental deployment) included in the package will be performed.
Import files (recommended)	Some database objects are linked with physical files stored on the file system inside the web project. If you check this option, the import process creates all files that are included in the import package.
Import global folders	If checked, the import process includes global files originally exported from the following folders: • ~/App_Code/Global/ • ~/CMSGlobalFiles/ • ~/CMSScripts/Custom/ • ~/App_Code/Controllers/ • ~/Controllers/Global/ • ~/Views/Global/
Import custom assembly files	If enabled, the import process includes custom assembly files bound to notification gateways, payment options, integration connectors, scheduled tasks and smart search indexes included in the import package. Custom assemblies are those whose names do not begin with the *CMS.* prefix.
Import site folders	If checked, the import process includes site-related files originally exported from the following folders: • ~/App_Code/<site code name>/ • ~/App_Data/<site code name>/ • ~/<site code name>/ • ~/Controllers/<site code name>/ • ~/Views/<site code name>/
Log staging synchronization tasks	If enabled, the system logs staging tasks reflecting all changes made by the import. Check this option to synchronize the imported data to other servers connected through Content staging.
Log integration tasks	If enabled, the system logs outgoing integration tasks for all changes made by the import. Check this option if you want to transfer the imported data to a system connected via the System integration bus).
Overwrite system queries	Available only when importing from an older version of Kentico CMS. If checked, the system imports all queries from the package and overwrites the current ones.

	Important: If the **package contains custom queries** that you added to the system, it is necessary to have this option enabled.



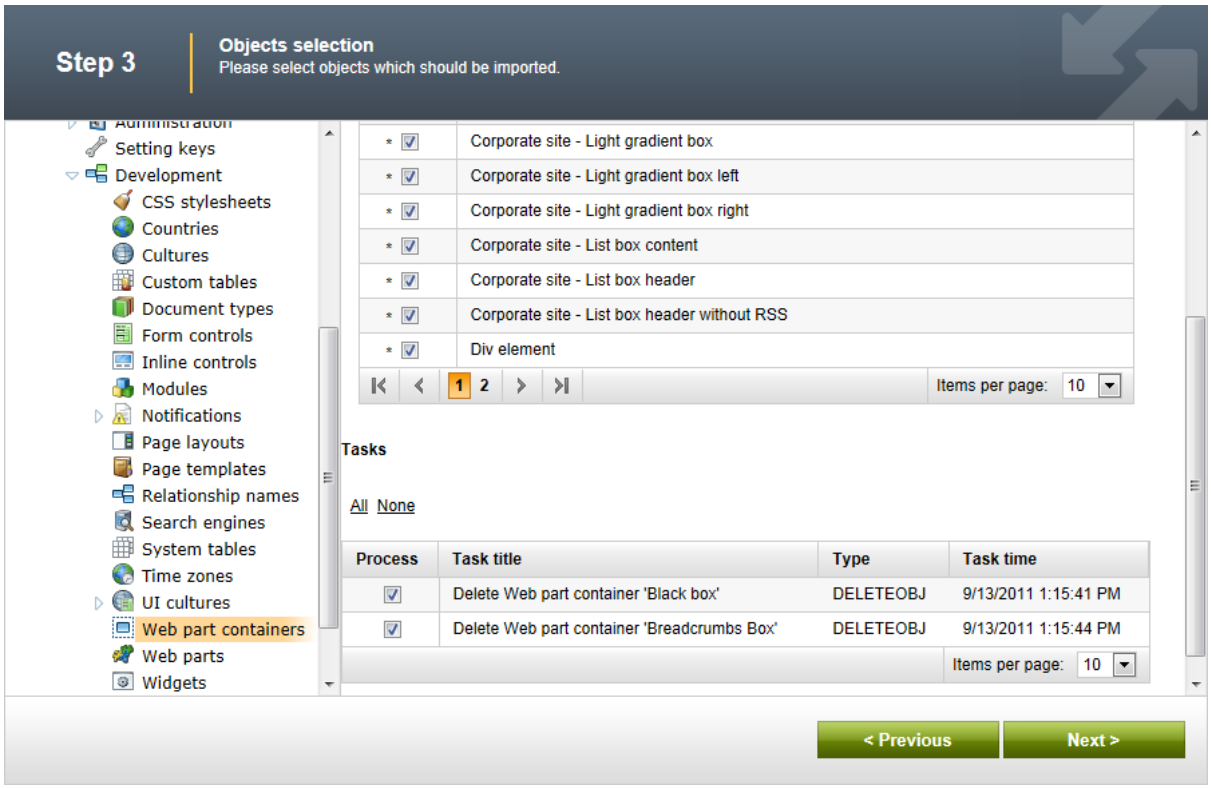
5. The following object categories contain extra options that you can set:

Automation processes	
Import all site specific triggers	If checked, all site specific triggers will be imported, even if the respective processes are not selected. This check-box is only displayed when the import package contains an exported website, not only separate objects. If you are importing to an existing site, this option is deselected by default.
Custom tables	
Import custom table data	If checked, custom table records (the actual data stored in the tables) included in the package will be imported together with the respective custom tables. If you are importing to an existing site, this option is disabled by default.
Documents	
Import new documents	If enabled, documents included in the import package will be imported. Please note: when importing into an existing site, only new documents can be imported. Modified documents that are already

	present on the target server will not be overwritten.
Import document histories	If enabled, document histories (i.e. previous versions of documents) will be imported.
Import document relationships	If enabled, document relationships will be imported.
Import document-level permissions	If enabled, document security settings made in **CMSDesk -> Content -> Edit -> Properties -> Security** will be imported.
Import blog comments	If enabled, blog comments will be imported together with blog post documents.
Import event attendees	If checked, event attendees will be imported together with all imported events.
Import all user personalization	If checked, user personalization of all users will be imported, even for users who are not selected to be imported. If you are importing to an existing site, this option is de-selected by default.
Forms	
Import form data	If enabled, forms data included in the package will be imported together with the forms.
Import physical files data	If checked, physical files saved within form records (if there are some) will also be imported.
Forums	
Import forum posts	If enabled, forum posts included in the package will be imported together with the forums.
Groups	
Import all group memberships	If enabled, user memberships will be imported together with the selected groups.
Message boards	
Import board messages	If enabled, board messages included in the package will be imported together with the message boards.
Media libraries	
Import media files	If enabled, media files (stored in database) included in the package will be imported together with the media libraries.
Import physical files	If enabled, physical files (stored in the file system) included in the package will be imported together with the media libraries.
Page templates	
Import all site specific page template scopes	If checked, site-specific page template scopes will be imported for all page templates, even if a page template itself is not selected to be imported. This check-box is only displayed with page templates in the **Global** section of the tree and only when the import package contains an exported website, not only separate objects. If you are importing to an existing site, this option is de-selected by default.

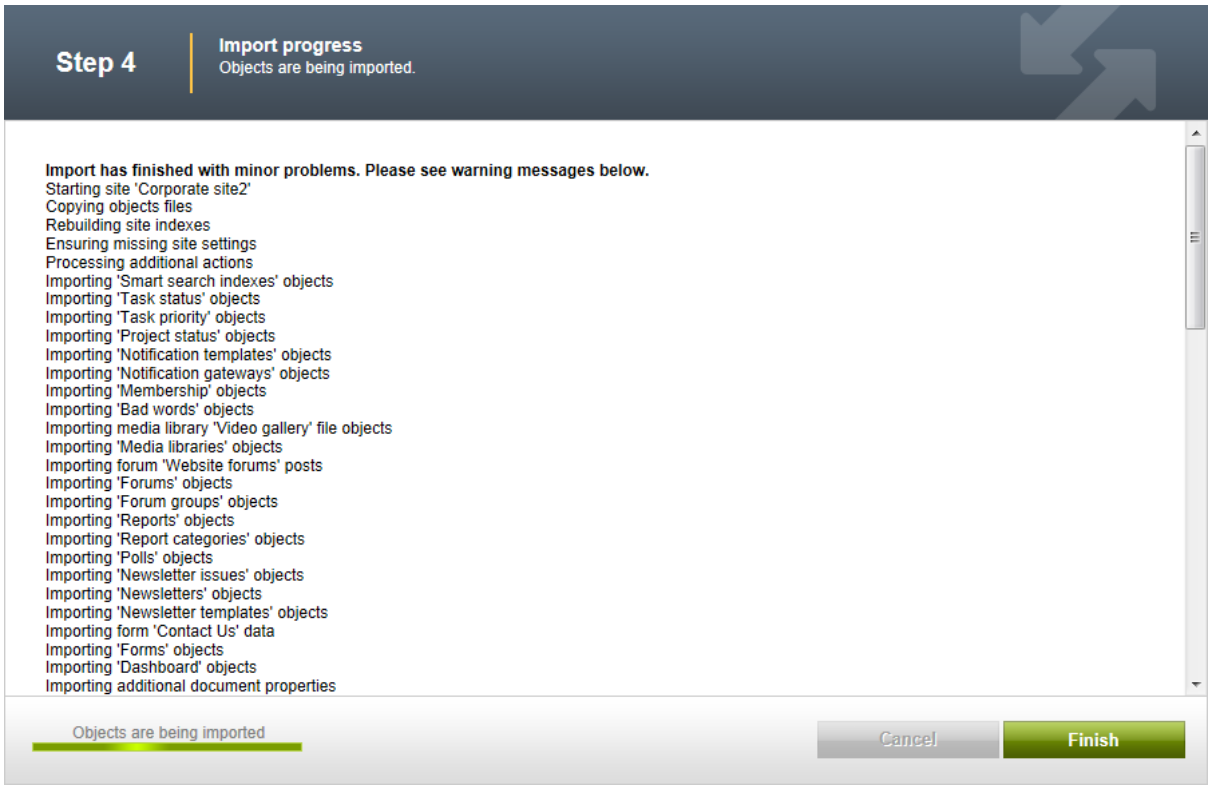
Import web part and zone variants with the selected page templates	If checked, web part and zone variants (from MVT testing or Content personalization) included in the package will be imported together with the selected page templates.
Users	
Import user dashboards	Indicates if user dashboards should be imported together with the imported users. If you are importing to an existing site, this option is de-selected by default so that existing dashboards are not overwritten.
Import all user site specific dashboards	If checked, all user dashboards will be imported, even if the respective users are not selected to be imported. This check-box is only displayed when the import package contains an exported website, not only separate objects. If you are importing to an existing site, this option is de-selected by default so that existing dashboards are not overwritten.
Workflows	
Import all workflow scopes	If checked, all workflow scopes will be imported, even if the respective workflows are not selected. Import web part and zone variants with the selected page templates. This check-box is only displayed when the import package contains an exported website, not only separate objects. If you are importing to an existing site, this option is deselected by default.

6. If you have the **Log export tasks** option enabled in **Site Manager -> Settings -> Versioning & Synchronization -> Staging**, a list of tasks may also be displayed under the objects list. This happens when some global objects were deleted (like the two Web part containers in the screenshot below). If you leave the boxes checked, these objects will also be deleted on the target server.



7. Click **Next** to execute the import process. An import log appears, showing the progress of the import (you can abort the import process at any time by clicking **Cancel**).

Click **Finish**. You will be redirected back to **Site manager** -> **Sites**.



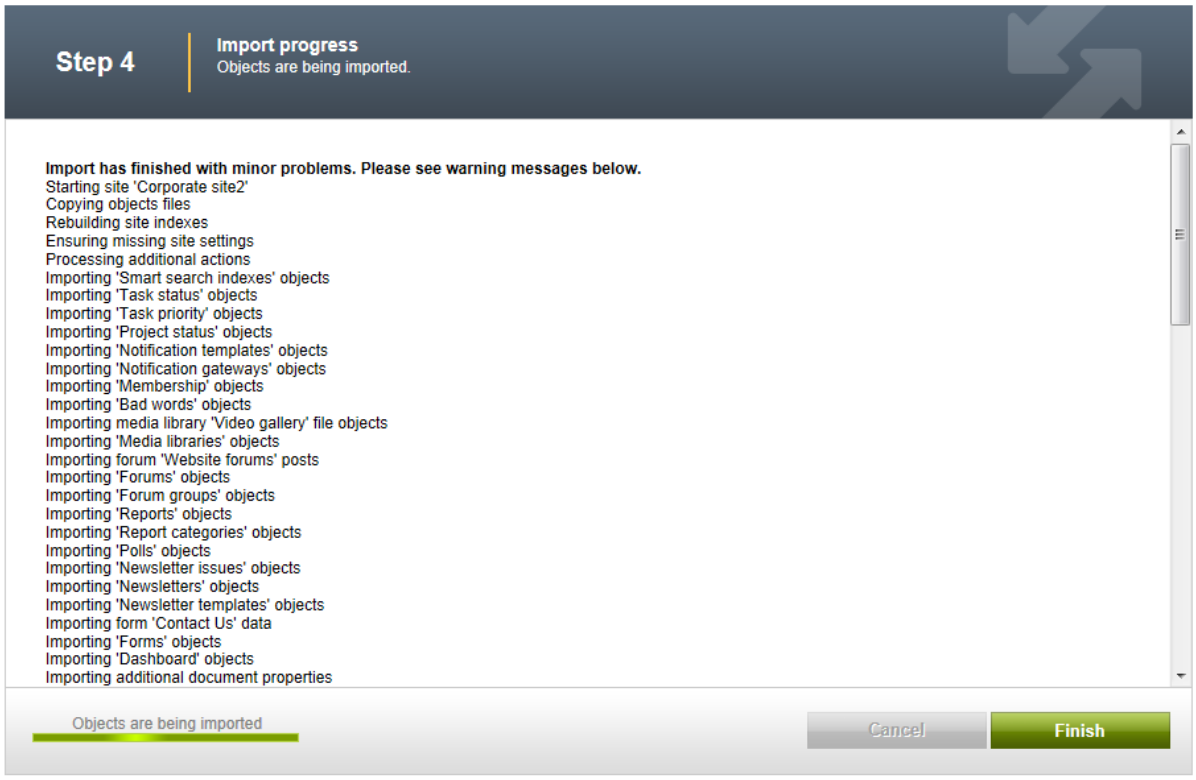
Please note

Packages from different versions of Kentico CMS have a different structure. When importing packages from an older version of Kentico CMS to a newer one, the system automatically converts the package to the newer format.

Please pay special attention when importing **Form control**, **Inline control** and **Web part** objects from older packages. If possible, avoid overwriting your current objects of these types, as it may cause incompatibility problems.

Conflicts of running sites

If the imported site uses the same domain name or alias as one of the websites already running in your system, the import displays a warning at the end and the imported site does not start.



WARNING: Failed to start site 'Corporate site2', there is already a site running under this domain alias. You first need to stop the existing site and then start the new site manually.

In such cases, you need to:

1. Go to **Site Manager -> Sites**.
2. Edit () the imported site and change the conflicting **Site domain name** or **Domain alias**.
3. Return to the list of sites and click **Start site** () next to the imported site.

Application restart

If you get the following error message at the end of the import process:

"Application has been restarted and the logging of the import process has been terminated. Please see context help in this section for more details and how to solve this issue."

you need to finish the import process manually:

1. Open the import package and extract all content of the *<package>\Data\Files* folder.

2. Remove the *.export* extension from the names of all files in all extracted sub-folders.

3. Rename all folders named ##SITENAME## to the code name of the target website

and copy them to the root of the web project.

4. Copy all contents of the *cms_attachments* folder (if it is present among the extracted folders) to the location in the target project where document attachments are stored (as configured in Site Manager -> Settings -> System -> Files). Please note that this is applicable only when the system is configured to store document attachments in the file system (not in the database).

5. Copy the content of all folders named as an object type (e.g. *cms_avatar*, *cms_documenttype*, *forums_forum*, etc.) to the root of your web project.

Re-signing imported macro expressions *[requires hotfix 7.0.9 or newer]*

The system uses signatures to ensure the security of macro expressions. The signatures are only valid in the environment of the application where the macros were originally saved.

To ensure that macros inside the data of imported sites or objects work correctly, you need to re-sign the macros using the **Site Manager -> Administration -> System -> Macros** interface. See Macro security for additional information.

5.5.5.2 Importing from web site to web application projects

Due to the differences between website and web application projects, the system cannot import from one to another automatically. After completing the standard import procedure, manually perform the following steps to ensure that your web application runs correctly:

1. The import places all physical files from the imported package into the *~/App_Data/CMSTemp/ImportExport/Files* folder. Copy the content of the folder into the root of your web application project.

Important!

We strongly recommend creating a backup of your project before overwriting any files.

2. Open the solution file (*WebApp.sln*) in Visual Studio from the web project directory.

3. Include the files that did not exist in the project before the import:

a. Click **Show all files** at the top of the Solution Explorer.
b. Locate the newly added files.
c. Select the files you wish to include one-by-one while holding the **Ctrl** key.
d. Right-click one of the files and select **Include in Project**.

4. Right-click the project node (CMSApp) in the Solution Explorer, and select **Convert To Web Application**.

5. Rebuild the solution.

5.5.6 Export and import internals and API

5.5.6.1 Overview

In this chapter, you will learn which API classes are used by export and import and see the most common API examples.

Advanced API examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all methods from the module classes, please refer to Kentico CMS API Reference.

5.5.6.2 API classes

If you are not familiar with database table data management through Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

The classes used by export and import can be found in the **CMS.CMSImportExport** namespace.

Import API classes:

- **ImportProvider** - provides functionality for import of sites and objects.
- **SiteImportSettings** - object representing settings of website import.
- **ImportTypeEnum** - enumeration used to determine object pre-selection in import settings.

Export API classes:

- **ExportProvider** - provides functionality for export of sites and objects.
- **SiteExportSettings** - object representing settings of website export.
- **ExportTypeEnum** - enumeration used to determine object pre-selection in export settings.

Other classes:

- **ImportExportHelper** - provides helper methods for export and import.

5.5.6.3 API examples

5.5.6.3.1 Overview

These topics show examples of how the export and import API can be used:

- Import
- Export

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Tools\ImportExport\Default.aspx.cs***.

The API examples of export and import use the following namespaces:

```
using System;

using CMS.GlobalHelper;
using CMS.UIControls;
using CMS.CMSHelper;
using CMS.SiteProvider;
using CMS.CMSImportExport;
using CMS.IO;
using CMS.DocumentEngine;
using CMS.WorkflowEngine;
```

5.5.6.3.2 Import

The following example imports a single user object included in a sample *APIExample_User.zip* import package stored in *~\CMSAPIExamples\Code\Tools\ImportExport\Packages*.

```
private bool ImportObject()
{
    // Create site import settings
    SiteImportSettings settings = new SiteImportSettings(CMSContext.CurrentUser);

    // Initialize the settings
    settings.WebsitePath = Server.MapPath("~/");
    settings.SourceFilePath = settings.WebsitePath + "\\CMSAPIExamples\\Code\
\Tools\\ImportExport\\Packages\\APIExample_User.zip";
    settings.ImportType = ImportTypeEnum.All;
    settings.LoadDefaultSelection();

    // Import
    ImportProvider.ImportObjectsData(settings);

    // Delete temporary data
    ImportProvider.DeleteTemporaryFiles(settings, false);

    return true;
}
```

The following example imports a complete website included in a sample *APIExample_Site.zip* import package stored in *~\CMSAPIExamples\Code\Tools\ImportExport\Packages*.

```
private bool ImportSite()
{
    // Prepare the properties
    string websitePath = Server.MapPath("~/");
    string sourceFilePath = websitePath + "\\CMSAPIExamples\\Code\\Tools\
\ImportExport\\Packages\\APIExample_Site.zip";
    string siteDisplayName = "My new imported site";
    string siteName = "MyNewImportedSite";
    string siteDomain = "127.0.0.1";

    // Ensure there is no site with the set name
    if (SiteInfoProvider.GetSiteInfo(siteName) == null)
    {
        // Import
        ImportProvider.ImportSite(siteName, siteDisplayName, siteDomain,
sourceFilePath, websitePath, CMSContext.CurrentUser);

        return true;
    }

    return false;
}
```

5.5.6.3.3 Export

The following example exports the user object imported by the first example on the previous page into a new export package.

```
private bool ExportObject()
{
    // Delete temporary data
    try
    {
        ExportProvider.DeleteTemporaryFiles();
    }
    catch
    {
    }

    // Get user
    UserInfo exportedUser = UserInfoProvider.GetUserInfo("MyNewImportedUser");

    // Ensure that user exists
    if (exportedUser != null)
    {
        // Prepare the properties
        string websitePath = Server.MapPath("~/");
        string exportFileName = string.Format("APIExample_User_{0:yyyy-MM-dd_hh-
mm}.zip", DateTime.Now);
        string exportFilePath = FileHelper.GetFullPhysicalPath
(ImportExportHelper.GetSiteUtilsFolder(), websitePath) + "Export\\" +
exportFileName;
```

```
        // Ensure there is no exported package with the same name
        if (!File.Exists(exportFilePath))
        {
            // Export
            ExportProvider.ExportObject(exportedUser, exportFilePath,
websitePath, CMSContext.CurrentUser);

            return true;
        }
    }

    return false;
}
```

The following example exports the website imported by the first example on the previous page into a new export package.

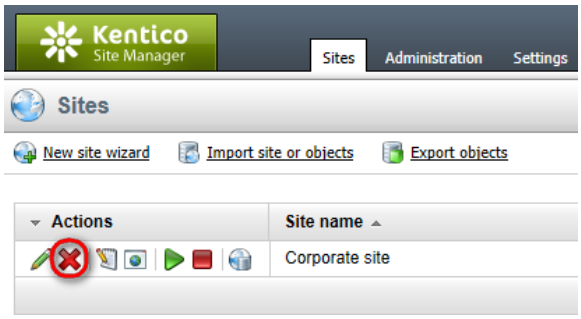
```
private bool ExportSite()
{
    // Delete temporary data
    try
    {
        ExportProvider.DeleteTemporaryFiles();
    }
    catch
    {
    }

    // Prepare the properties
    string websitePath = Server.MapPath("~/");
    string exportFileName = string.Format("APIExample_Site_{0:yyyy-MM-dd_hh-mm}
.zip", DateTime.Now);
    string exportFilePath = FileHelper.GetFullPhysicalPath
(ImportExportHelper.GetSiteUtilsFolder(), websitePath) + "Export\\" +
exportFileName;
    string siteName = "MyNewImportedSite";

    // Ensure that site exists
    if (SiteInfoProvider.GetSiteInfo(siteName) != null)
    {
        // Ensure there is no exported package with the same name
        if (!File.Exists(exportFilePath))
        {
            // Export
            ExportProvider.ExportSite(siteName, exportFilePath, websitePath,
false, CMSContext.CurrentUser);

            return true;
        }
    }
```

```
    return false;
}
```

5.6 Deleting sites

You can delete sites in the system in **Site Manager -> Sites**.

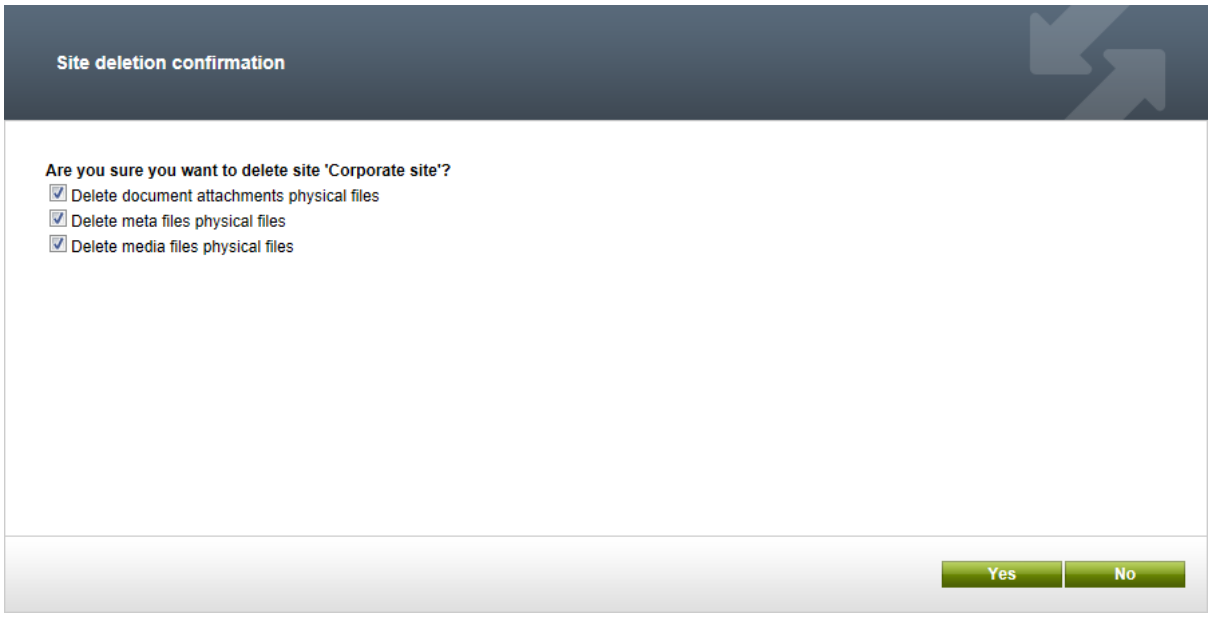
1. Clicking the **Delete** (✖) icon of the site that you want to delete.



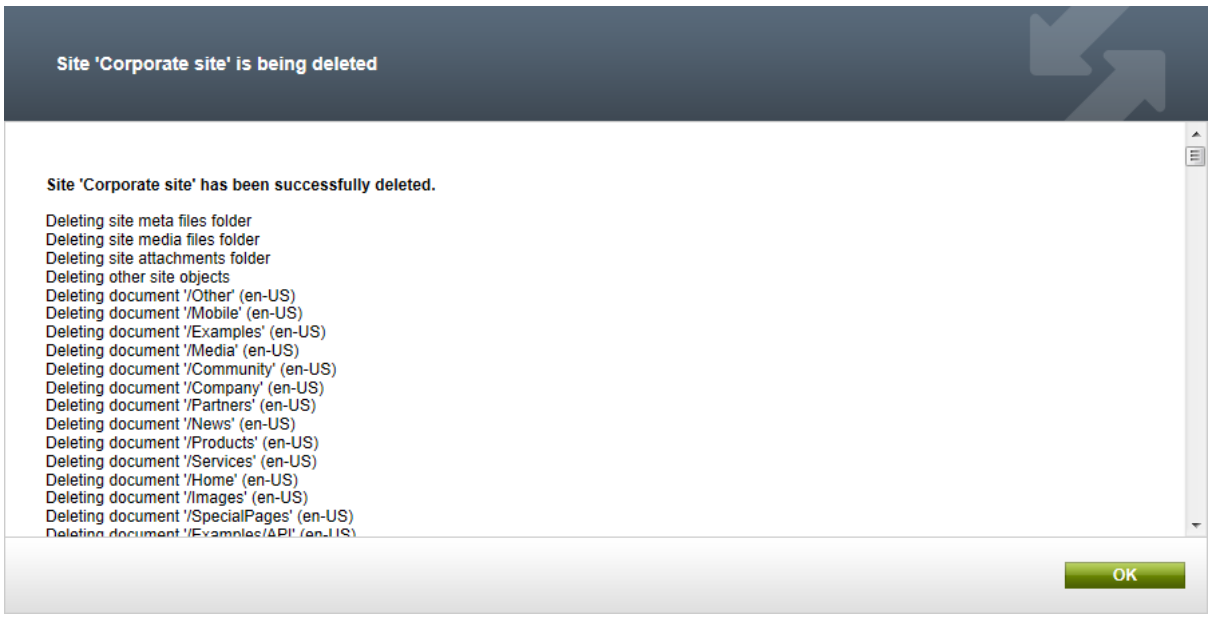
2. In the **Site deletion confirmation** dialog, you can select the following options:

- **Delete document attachments physical files** - if checked, document attachment files stored in the file system will be deleted; these files are stored in the *<web project>\<site name>\files* folder
- **Delete meta files physical files** - if checked, meta files stored in the file system will be deleted; these files are stored in the *<web project>\<site name>\metafiles* folder
- **Delete media files physical files** - if checked, physical files stored in media libraries will be deleted; these files are stored in the *<web project>\<site name>\media* folder

Make the selection and click **Yes** to continue deleting the site.



3. A log will be displayed, showing you the progress of site deletion. When the process finishes, click **OK**. You will be redirected back to **Site Manager -> Sites**, where the deleted site will not be listed.

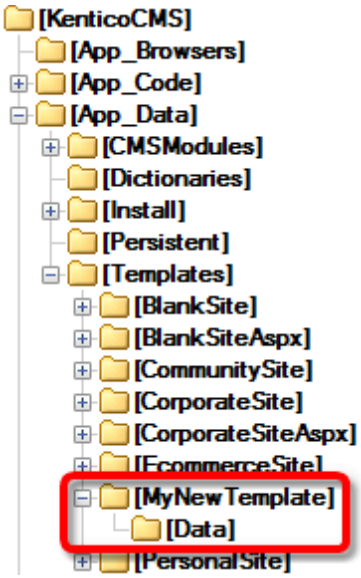


5.7 Creating web templates

In case that you want to use your current site created with Kentico CMS as a web template, so that you can use it a a starting point for developing new sites, you have to take the following steps:

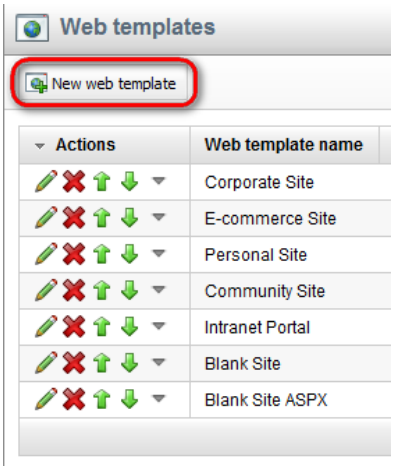
1. Export your site. For a step-by-step tutorial on how to export a site, please refer to the Exporting a site chapter of this guide.

2. Go to **<web project>\App_Data\Templates**. As you can see, this is the folder where all the default templates, such as Community or Corporate site, are stored. Create a new folder with the name of your new page template. Then create one sub-folder under the newly created folder and give it the name **Data**.



3. Extract the content of your export package into the **Data** folder.

4. Go to **Site Manager -> Development -> Web templates** and click **New web template**.



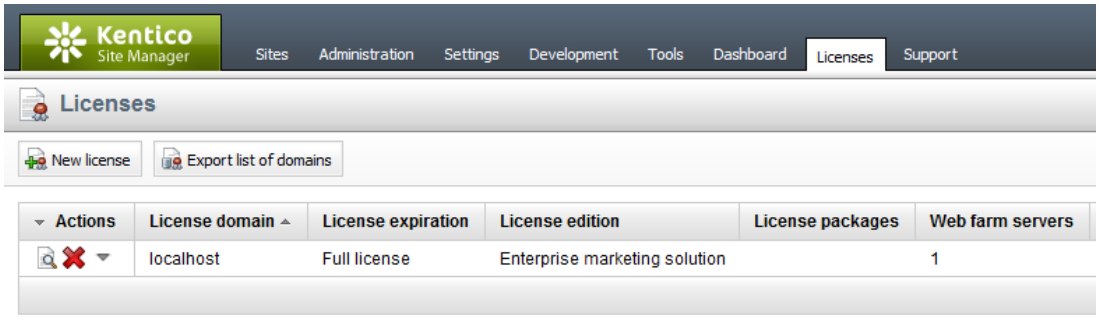
5. Enter the following details:

- **Web template display name** - name of the web template displayed in the administration interface
- **Web template code name** - name of the web template used in code
- **Web template folder name** - path to the folder where you have extracted the content of the export package; *~\App_Data\Templates\<your folder>*
- **Web template description** - text describing your new web template
- **License editions** - editions of Kentico CMS in that this web template will be available; check all for full availability

and click **OK**. Your new web template should now be present in the list.

5.8 License management

Kentico CMS requires an appropriate license key for every domain you use. The licenses can be managed in **Site Manager -> Licenses**:



The list displays the information about licensed domain, expiration and edition. When you get full or trial key for a particular domain, you need to click **New license** and enter the full text of the key into the **License key** field.

You can also use **Export list of domains** to export your domains into a text file.

How licensing works

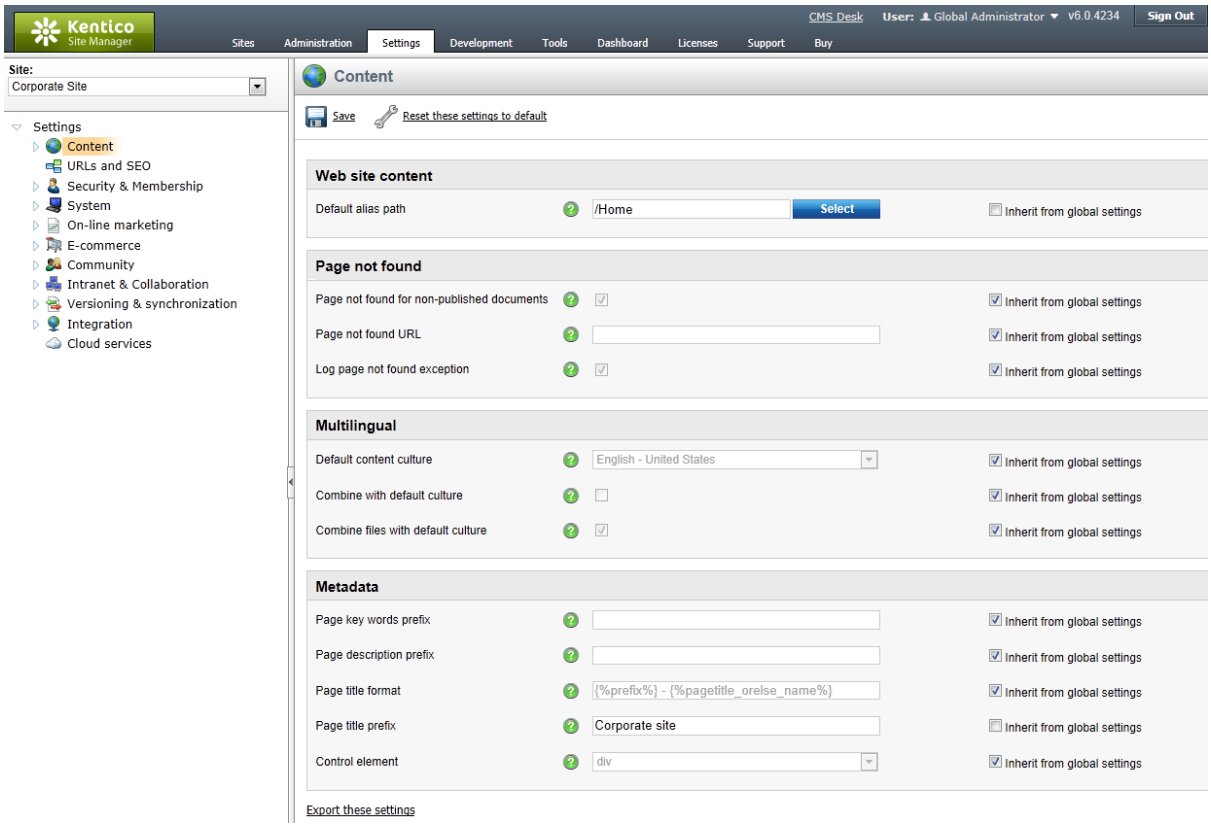
If you're running a website on domain *example.com*, you need a single license key that will work for the following domains:

- http://example.com
- https://example.com
- http://www.example.com
- https://www.example.com
- http://localhost
- http://127.0.0.x (where x is between 1 and 255)

If you use a domain alias (different domain name that points to the same website), such as *example1.com* or *example.net*, you need extra license keys for these domain aliases. Please ask Kentico support for generating the additional keys (they are free of charge if you already own a license for the main domain).

5.9 Managing site settings

Most of the site settings can be configured in **Site Manager -> Settings** section.



There are two basic types of settings:

- **Global** – such settings apply to all sites.
- **Site-specific** – such settings apply to the particular site and they override the global settings values.

If you want to inherit value from the global settings, you need to check the “inherit from global settings” button and click Save.

Tip: If you mouse-over the name of the settings key, you will see the description of the key.

5.10 Configuring multiple web sites

Kentico CMS allows you to run multiple websites from a single installation (code base) and database. All websites run as a single website in IIS. The following tutorial explains how to set up multiple websites on Windows XP and Windows Server 2003.

We will configure two websites:

- **mysite.com**
- **mysite2.com**

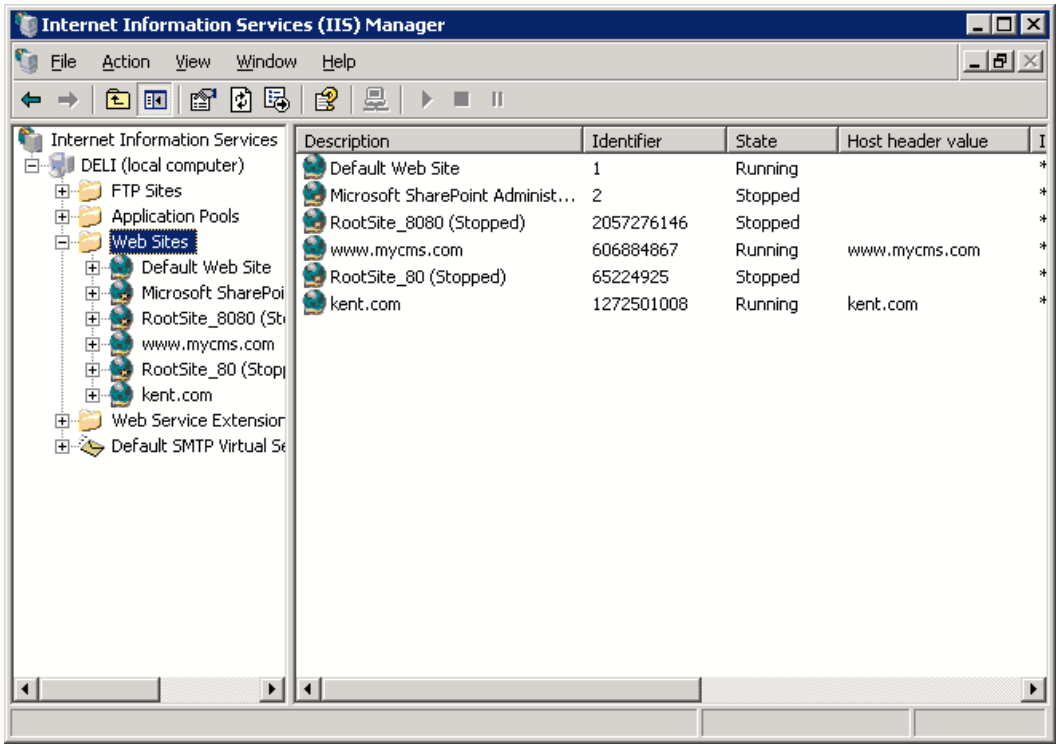
Configuring multiple sites in Kentico CMS (common for all operating systems)

This part is common for all operating systems.

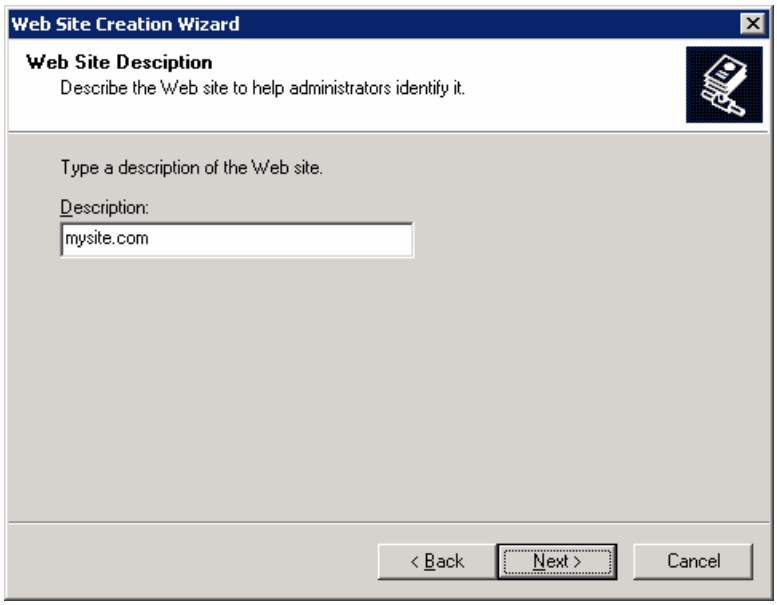
1. Create two websites in **Site Manager -> Sites** - you can either import your existing websites or you can create new websites using the New site wizard.
2. Edit the properties of each website in **Site Manager -> Sites** and set the **Site domain name** value of the each website to the appropriate domain (without www prefix and without http:// protocol).
3. Make sure both sites are running. You can check this in the Site list, in the Status column.
4. Make sure you have valid license keys for both domains in **Site Manager -> Licenses**.

Configuring multiple sites on Windows Server 2003

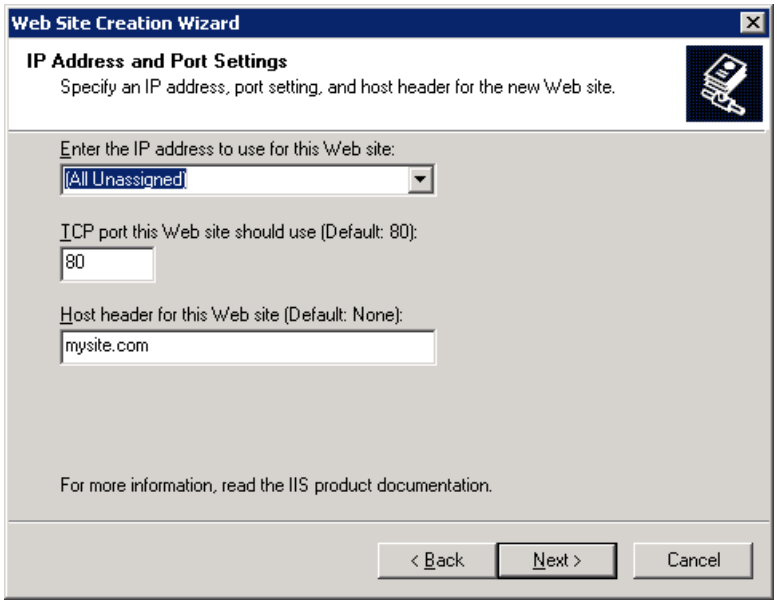
Open **Start -> Administrative tools -> Microsoft Internet Information Services (IIS) Manager**, go to websites section and if the website does not exist yet, create it.



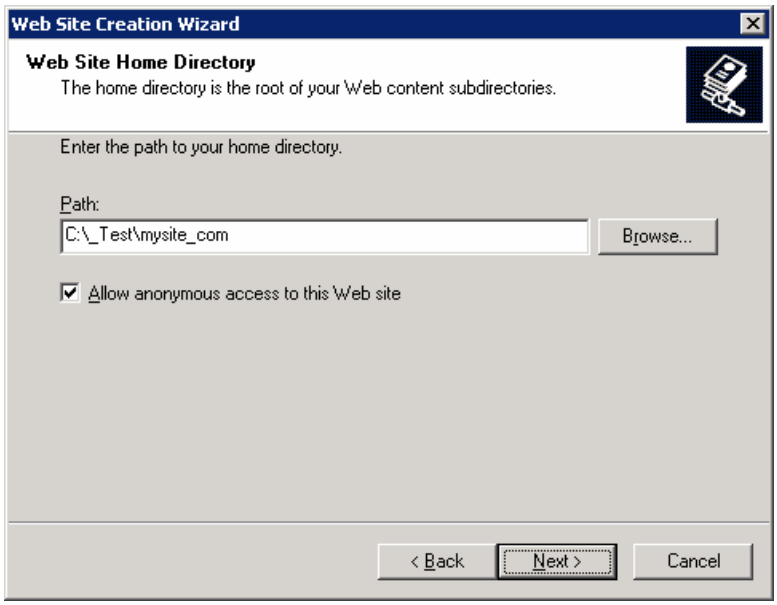
Right-click **websites** and choose **New -> website...** The website Creation Wizard starts. Enter the site descriptive name, such as **mysite.com**:



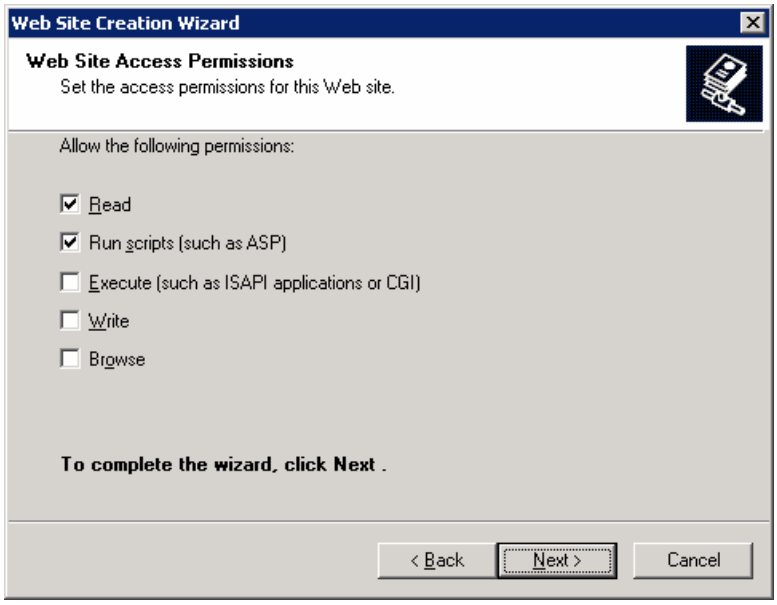
Click **Next**. Enter the **Host header for this website** as **mysite.com**:



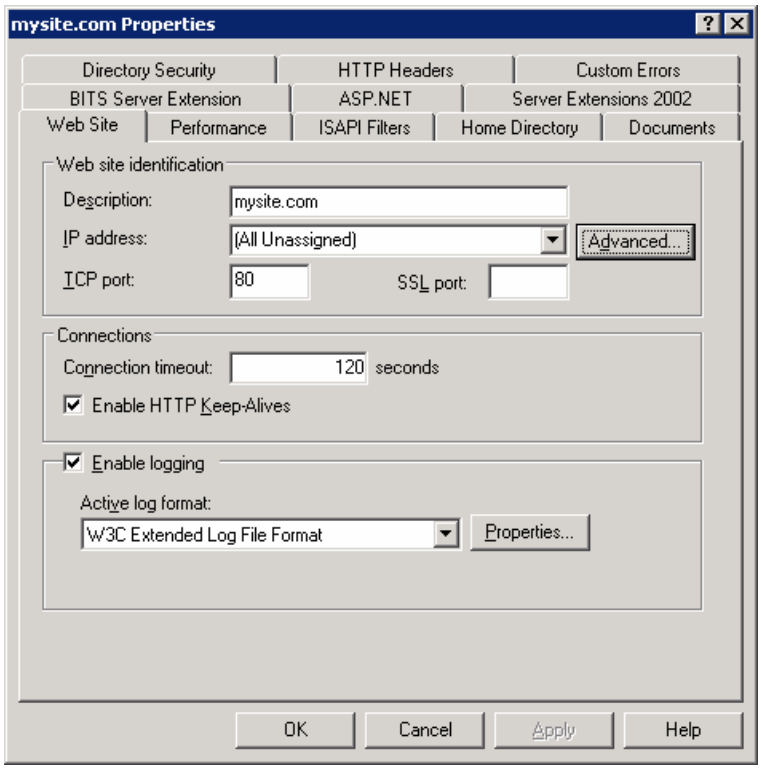
Click **Next**. Select the disk path where the root of your website is placed. This must be the folder where web.config file is placed.



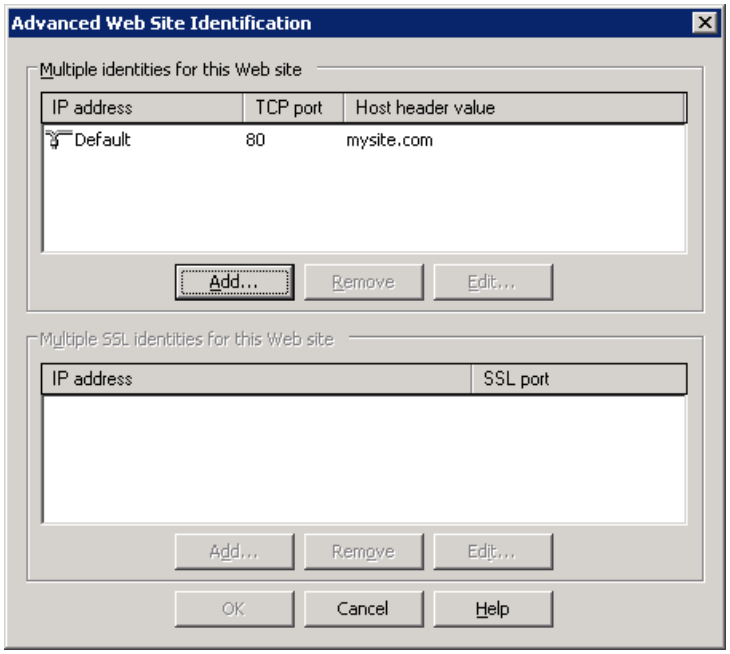
Click **Next**. In the next step, enable the **Read** and **Run scripts** boxes.



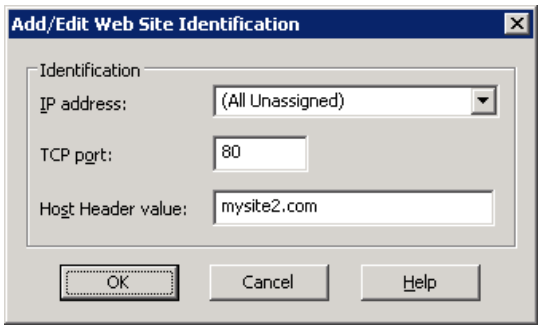
Click **Next**. The wizard is finished. Click **Finish**. Now we need to add the other domain name to the list of URLs hosted by this website. Right-click the newly created website and display website properties. On the **website** tab, click **Advanced...**



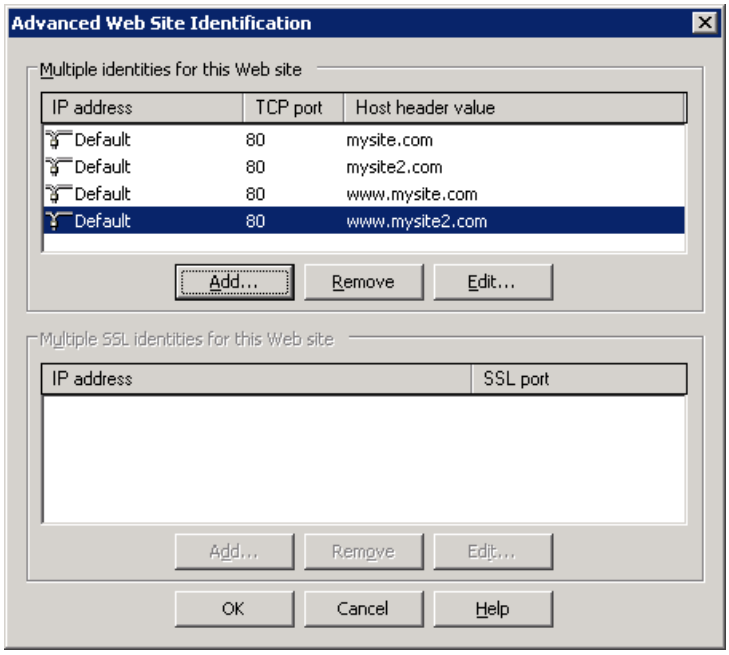
Now you need to add another host header value for your second domain:



Click the top **Add...** button and enter the appropriate values. The standard HTTP port is 80:

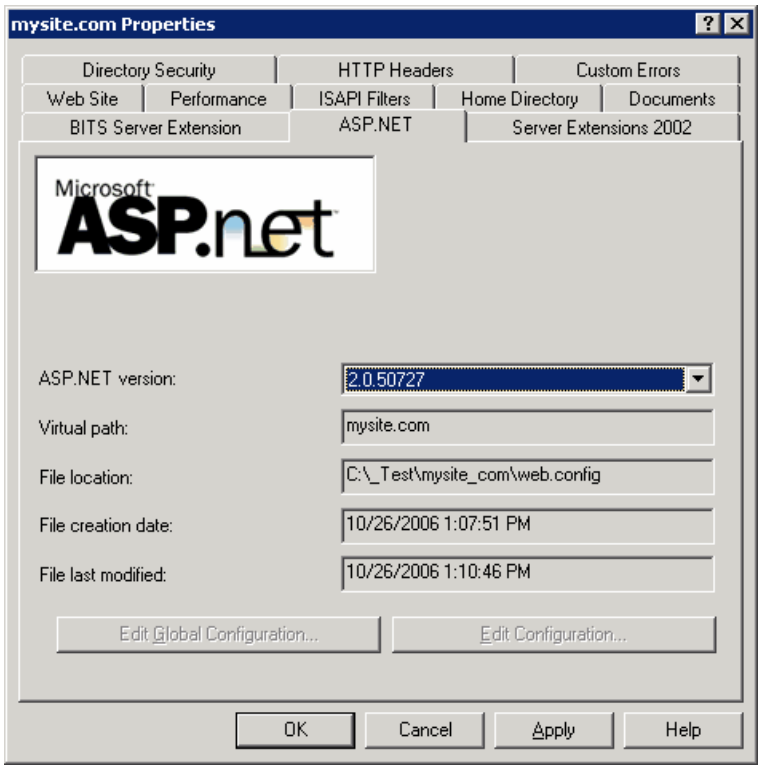


You need to repeat this also for www.* alternatives of your website:



Click **OK** on all dialogs.

Please note: You may also need to configure the website for ASP.NET 2.0 in website Properties, on the **ASP.NET** tab. IIS 6 may display the **ASP.NET version as 2.0.50727** even if you have **ASP.NET 3.5** or higher installed and registered:



Your new website is now configured to host all incoming requests for domains mysite.com and mysite2.com (or other domains depending on your particular situation). You may need to ask your network administrator to redirect the domain in the DNS records to your website.

If you do not own the domain, you can test it by modifying the C:\WINDOWS\system32\drivers\etc\hosts file in notepad and adding the following lines to the end of the file:

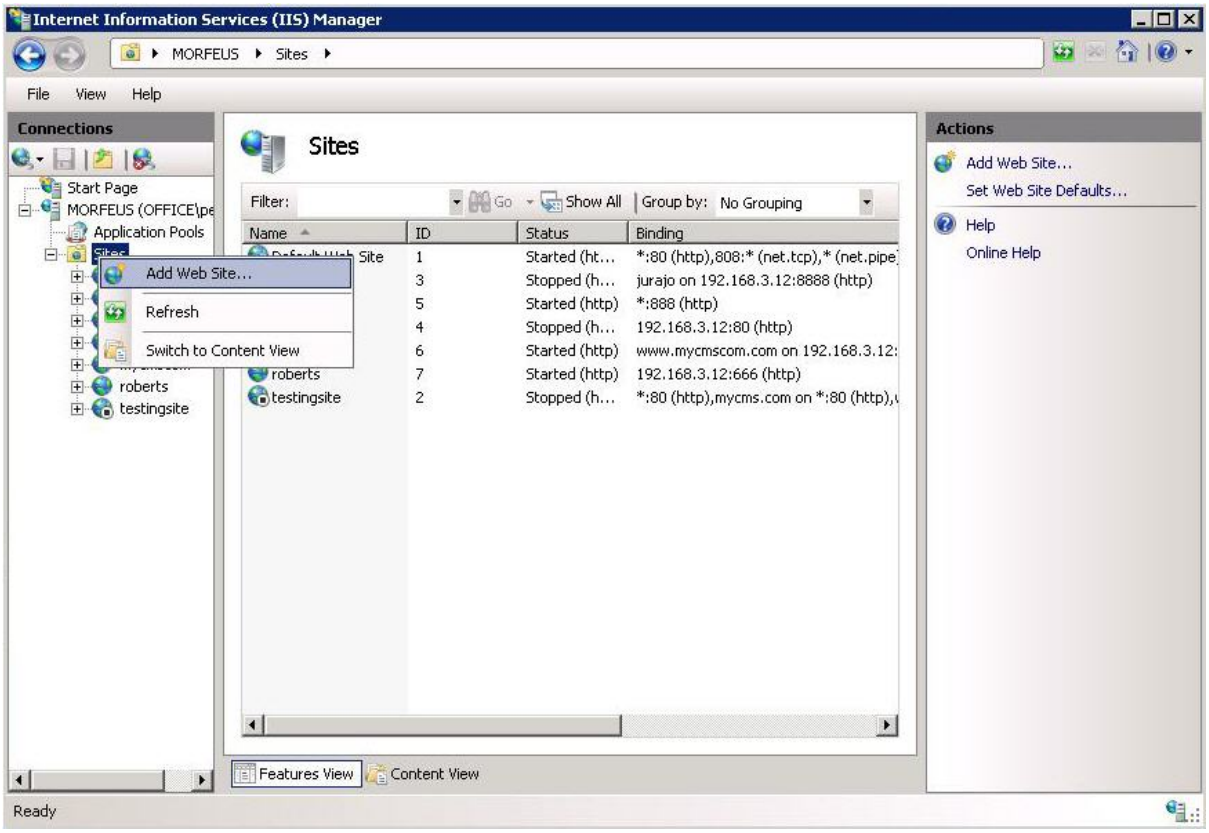
```
127.0.0.1       mysite.com
127.0.0.1       www.mysite.com
127.0.0.1       mysite2.com
127.0.0.1       www.mysite2.com
```

Save the file. Please note: these are client settings, which means they will work only if you use web browser on your server.

Now, when you go to *http://www.mysite.com* and to *http://www.mysite2.com* (or your own domain names), you should see two different websites.

Configuring multiple sites on Windows Vista / Server 2008

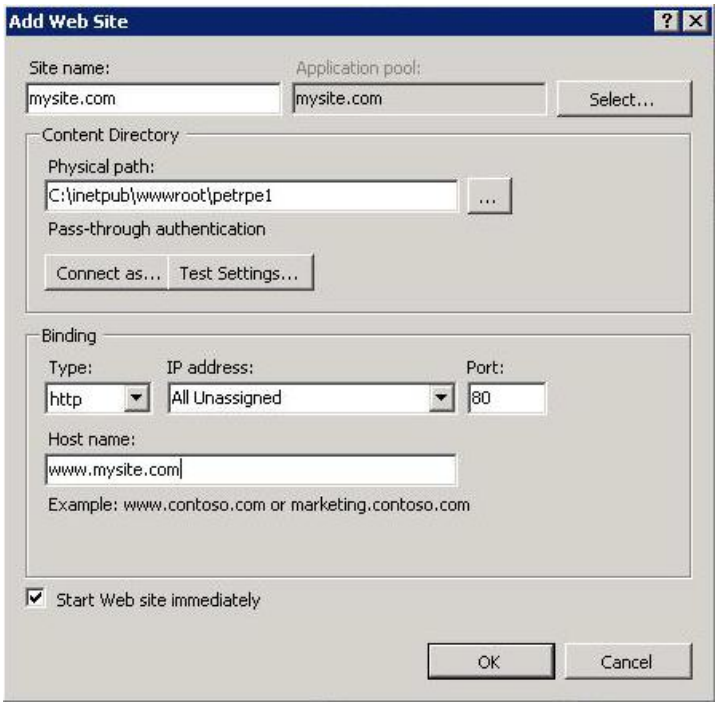
Go to **Start -> Control Panel -> Administrative tools -> Microsoft Internet Information Services (IIS) Manager.** In the tree view, right click **Sites** and choose **Add website...**.



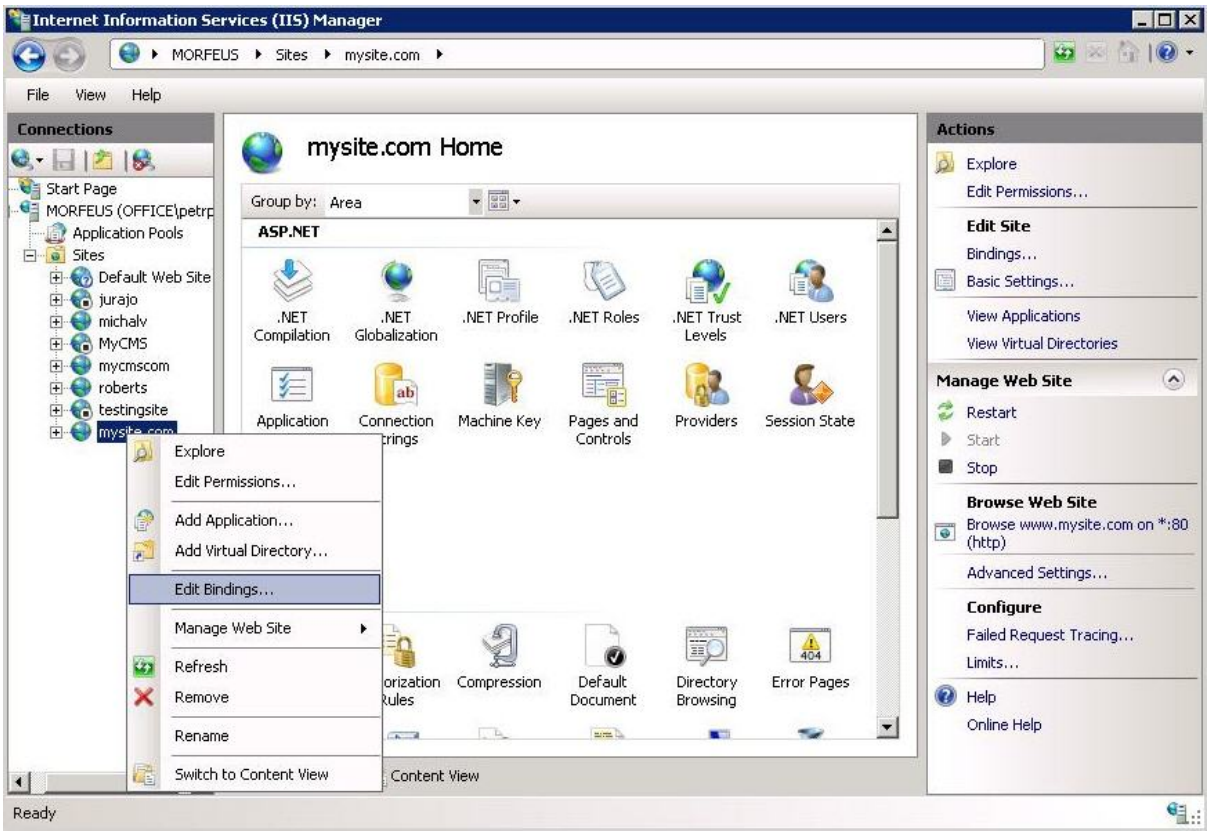
Enter the following details:

- **Site name**: mysite.com
- **Physical path**: disk path to the location where your website is placed; this must be the location where the *web.config* file is stored
- **Host name**: www.mysite.com

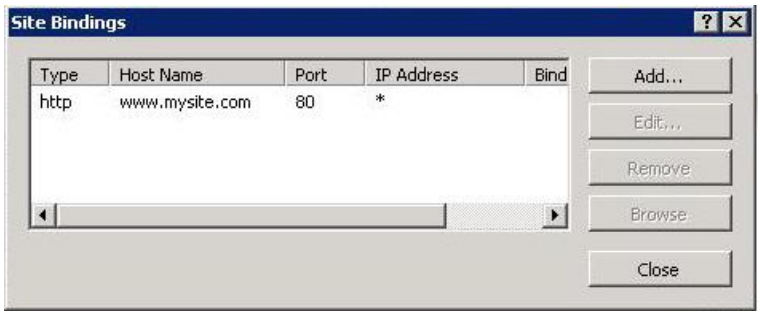
Click **OK**.



The site should now appear in the tree, under the **Sites** node. Right click the site and choose **Edit bindings** (or **Bindings** on Vista).

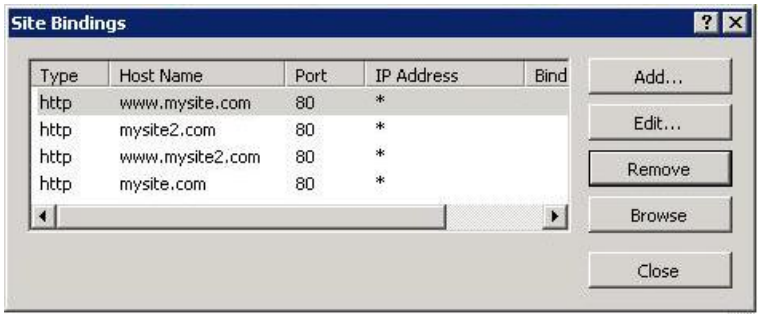


The site bindings dialog appears. Click **Add...**.



Enter the domain name of your second website (*www.mysite2.com* in this case) into the **Host name** field and click **OK**. Repeat this for both of the sites without the *'www.'* prefix.

The result should look like the following screenshot.



Your new website is now configured to host all incoming requests for domains mysite.com and mysite2.com (or other domains depending on your particular situation). You may need to ask your network administrator to redirect the domain in the DNS records to your website.

If you do not own the domain, you can test it by modifying the C:\WINDOWS\system32\drivers\etc\hosts file in notepad and adding the following lines to the end of the file:

```
127.0.0.1       mysite.com
127.0.0.1       www.mysite.com
127.0.0.1       mysite2.com
127.0.0.1       www.mysite2.com
```

Save the file.

Please note: these are client settings, which means they will work only if you use web browser on your server.

Now, when you go to http://www.mysite.com and to http://www.mysite2.com (or your own domain names), you should see two different websites.

Configuring mulitple sites on Windows XP

On Windows XP, the support of multiple sites and domains in IIS is limited, so we will use a single IIS website and define "virtual" domains in our hosts file:

Open the c:\WINDOWS\system32\drivers\etc\hosts file in notepad and add the following lines to the end of the file:

```
127.0.0.1       mysite.com
127.0.0.1       www.mysite.com
127.0.0.1       mysite2.com
127.0.0.1       www.mysite2.com
```

Save the file. Please note: these are client settings, which means they will work only if you use web browser on your local computer.

Go to http://www.mysite.com or http://www.mysite.com/kenticocms (where kenticocms is the name of the virtual directory with Kentico CMS website) and to http://www.mysite2.com. You should see two different websites.

Multiple websites on a single domain (in subfolders)

If you cannot use (for some reason) multiple domain names, you can configure Kentico CMS so that it differentiates websites by subfolder (virtual directory). Read chapter Multiple websites on a single domain (in subfolders) for more details.

Using multiple websites on Windows XP

Windows XP allows you to run only one IIS website at a time. If you need to develop multiple websites (multiple Kentico CMS instances) in the root folder, you may need to create additional websites and switch between them using the IISAdmin utility that can be downloaded at http://jetstat.com/iisadmin/download.asp.

5.11 Multiple web sites on a single domain (in subfolders)

In some cases, you may want to run multiple websites in separate subfolders, without getting a new domain for each one. This can be achieved by configuring IIS and the domain names (or domain aliases)

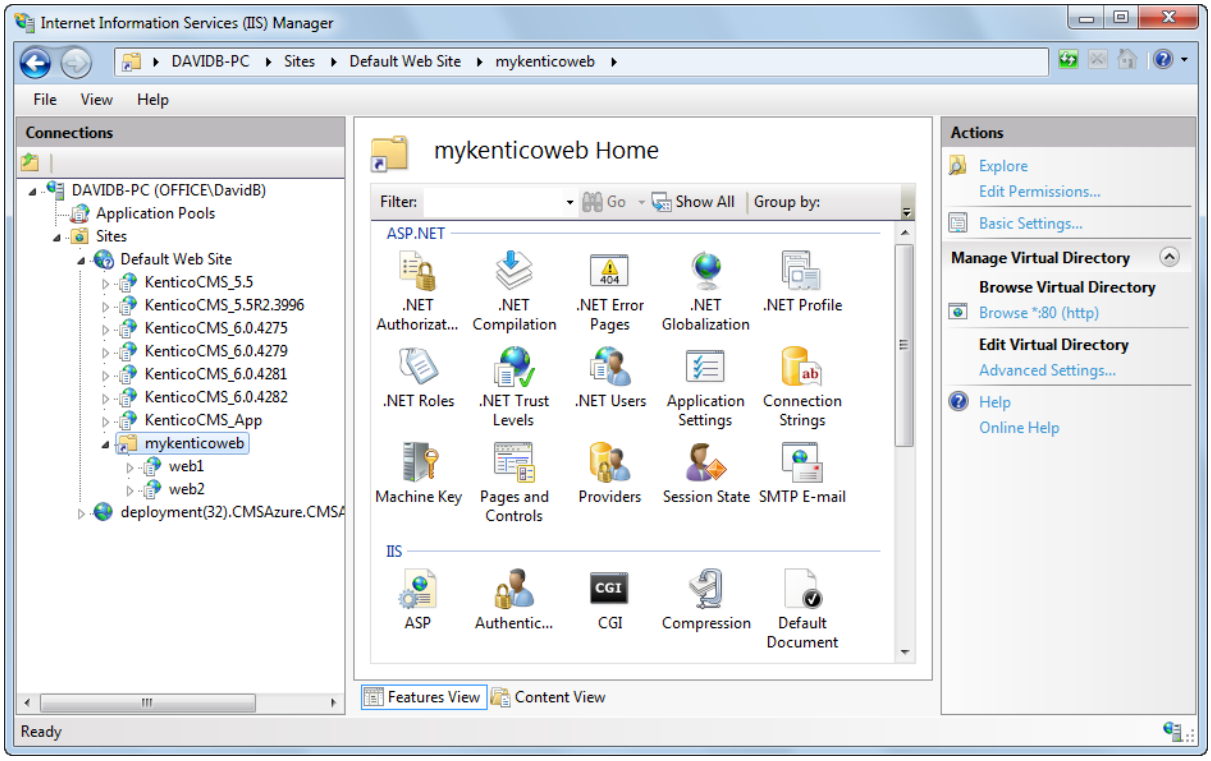
of the websites like in the following example:

Example

1. Install Kentico CMS to the following folder: **C:\inetpub\wwwroot\mykenticofolder**. Make sure the name of the folder is different from the virtual directory name that you will later enter in IIS. In the same step of the installer, choose **This is an installation to the root (do not create virtual directory)**. When the setup finishes, the link to your new website will not work as the virtual directory is not created in IIS.

2. Open the IIS console (Control Panels -> Administrative Tools -> Internet Information Services) and create a new virtual directory named **mykenticoweb**. Assign it to a **non-website-root folder** on your disk. Ideally, create an empty folder on the disk for this purpose (e.g. c:\empty).

3. Create two applications under **mykenticoweb** called **web1** and **web2**. Both of them need to be have their physical path assigned to the installation folder on the disk, which is **C:\inetpub\wwwroot\mykenticofolder** in this example. Remember to set an appropriate application pool according to the type of installation that you specified.



4. Now open your browser and type in either **http://localhost/mykenticoweb/web1** or **http://localhost/mykenticoweb/web2**. Kentico CMS Database setup appears. Continue through the setup as usual and install the first site.

5. When the setup finishes, go to **Site Manager -> Sites** and install the second site.

6. Configure the domain name for website 1 as: **localhost/mykenticoweb/web1**

7. Configure the domain name for website 2 as: **localhost/mykenticoweb/web2**

Now when you go to **http://localhost/mykenticoweb/web1**, you will see website 1. If you go to **http://localhost/mykenticoweb/web2**, you will see website 2.

8. To ensure the synchronization of settings and global objects between the two sites, it is necessary to set up a Web farm scenario. This can be done by adding the following keys to the *<appSettings>* section of the **web.config** file in the installation directory shared by both websites:

```
<add key="CMSWebFarmEnabled" value="true"/>
<add key="CMSWebFarmSynchronizeFiles" value="false" />

<add key="/mykenticoweb/web1:CMSWebFarmServerName" value="server1" />
<add key="/mykenticoweb/web2:CMSWebFarmServerName" value="server2" />
```

This enables web farms in general and disables synchronization of files, which is not needed since the applications already use the same physical folder. Notice that each application has a different web farm server name specified via a prefix in the name of the **CMSWebFarmServerName** key. This prefix must match the path that was set for the corresponding application in IIS, including the virtual directory.

Then go to **Site Manager -> Administration -> Web farm** on one of the sites and create a web farm server for each application according to the instructions in Defining web farm servers.

9. Next, to prevent potential problems with conflicts during Smart search indexing operations, specify a different physical folder for each application's search index files through the keys shown below:

```
<add key="/mykenticoweb/web1:CMSSearchIndexPath"
value="App_Data\CMSModules\SmartSearch\Web1\" />
<add key="/mykenticoweb/web2:CMSSearchIndexPath"
value="App_Data\CMSModules\SmartSearch\Web2\" />
```

Once this is done, your websites should work without any further issues.

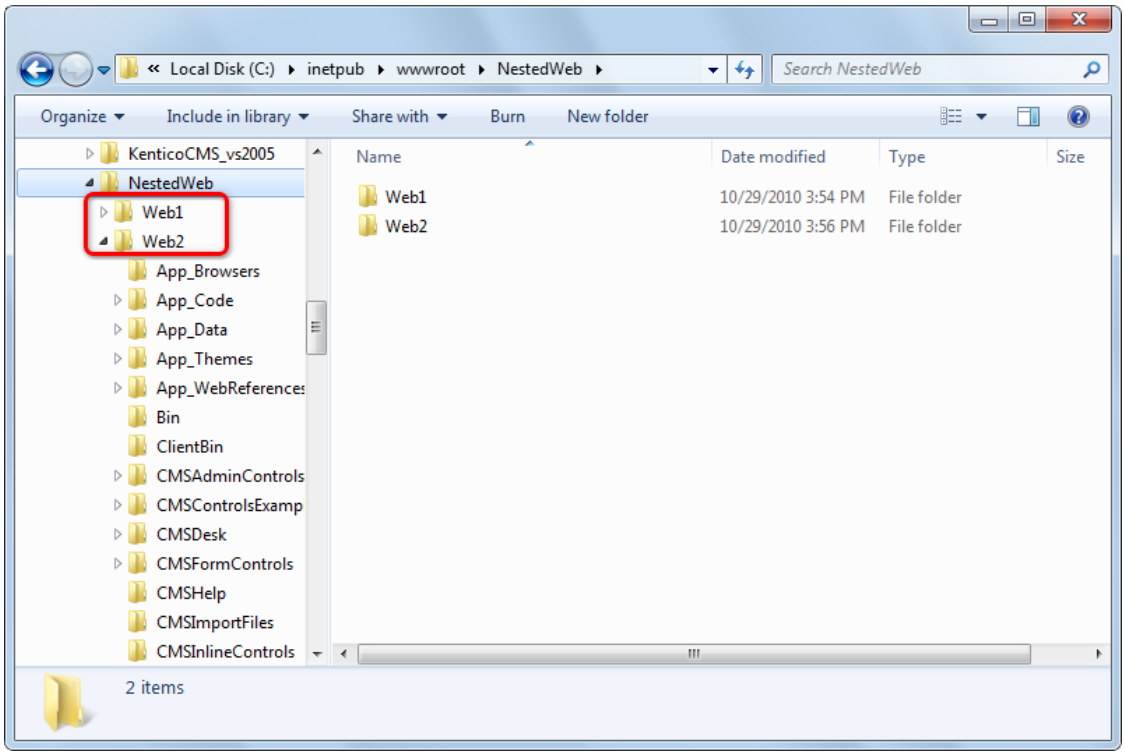
5.12 Configuring nested web sites

In the following example, you will learn how to set up two nested websites. It should be possible to achieve any level of nesting if the following steps are complied with.

1. Install two projects into two independent, not overlapping folders. Choose folder names that won't be the same as the names in the URL to prevent later colision when seting up the IIS. The installation folders may be for example like this:

- **Inetpub/wwwroot/NestedWeb/Web1** (first website from Web installer)
- **Inetpub/wwwroot/NestedWeb/Web2** (second website from Web installer)

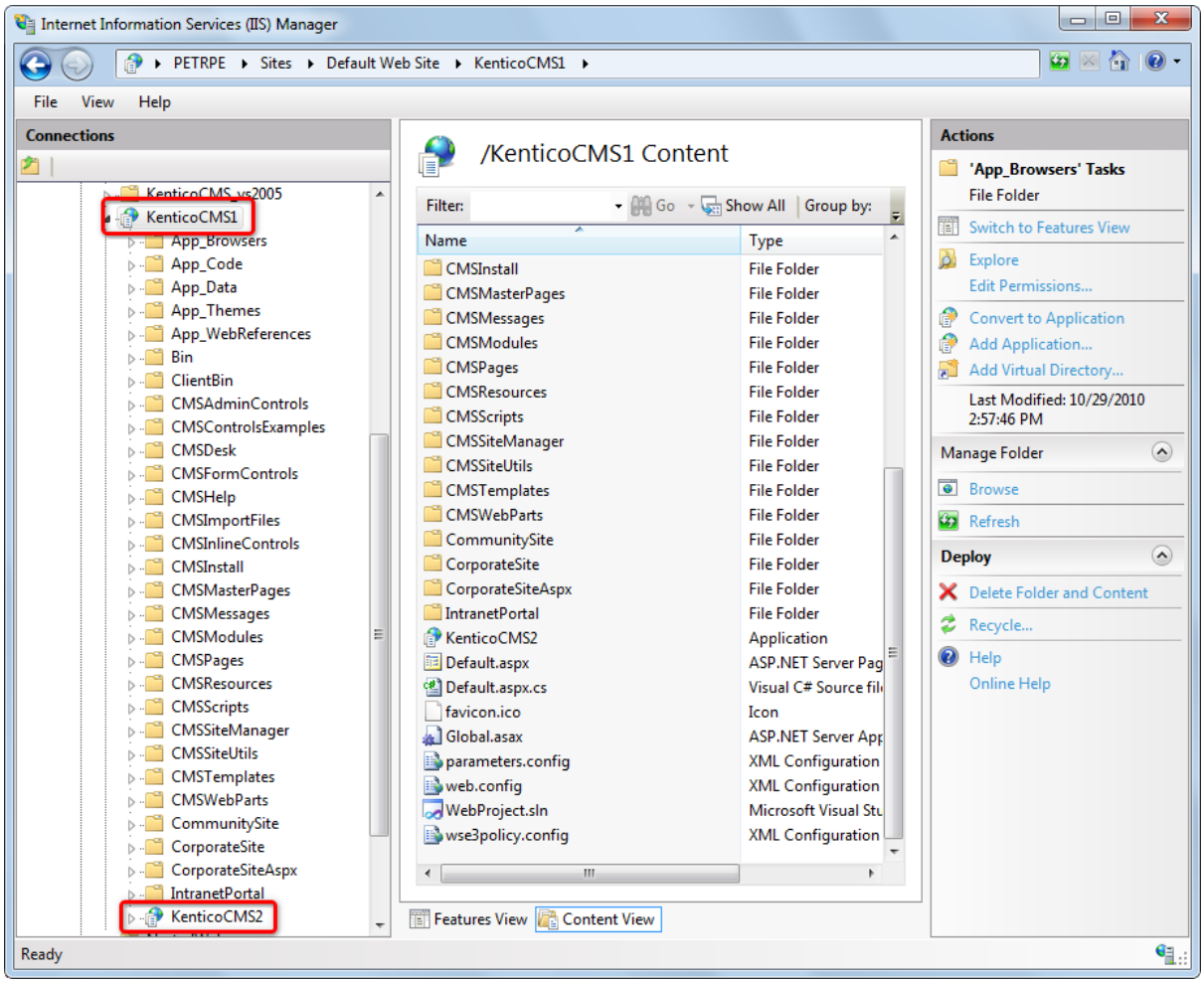
This means that you have two completely independent projects, as you can see in the screenshot below:



2. Open IIS management console and create two applications under the *Default website*:

- **[Default website]/KenticoCMS1** (pointing to physical directory Web1)
- **[Default website]/KenticoCMS1/KenticoCMS2** (child directory pointing to physical directory Web2)

Like this, you have two independent projects configured as nested only in the IIS.

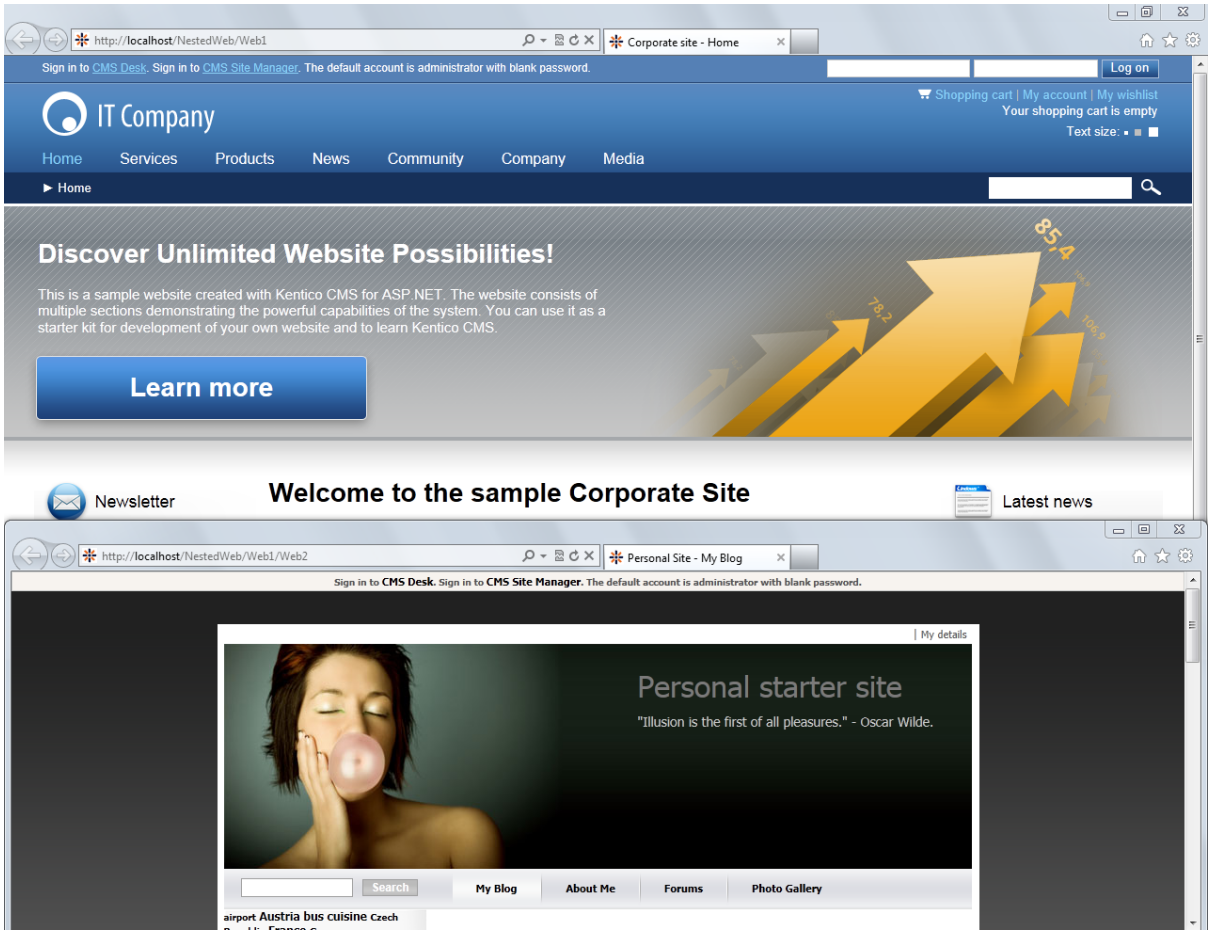


3. The last step is to avoid duplicate module and handler definition keys in the two projects' *web.config* files. To achieve this, open the nested application's (*Web2*) *web.config* file and remove all module and handler definitions from the two sections highlighted in the following code extract.

In case that you needed any additional custom keys in these sections, please ensure that they are not duplicate in the two *web.config* files, i.e. that they are only added in one of the two files.

```
<system.webServer>
  <validation validateIntegratedModeConfiguration="false" />
  <modules>

    // remove or comment out all keys in this section

  </modules>
  <handlers>

    // remove or comment out all keys in this section

  </handlers>
</system.webServer>
```

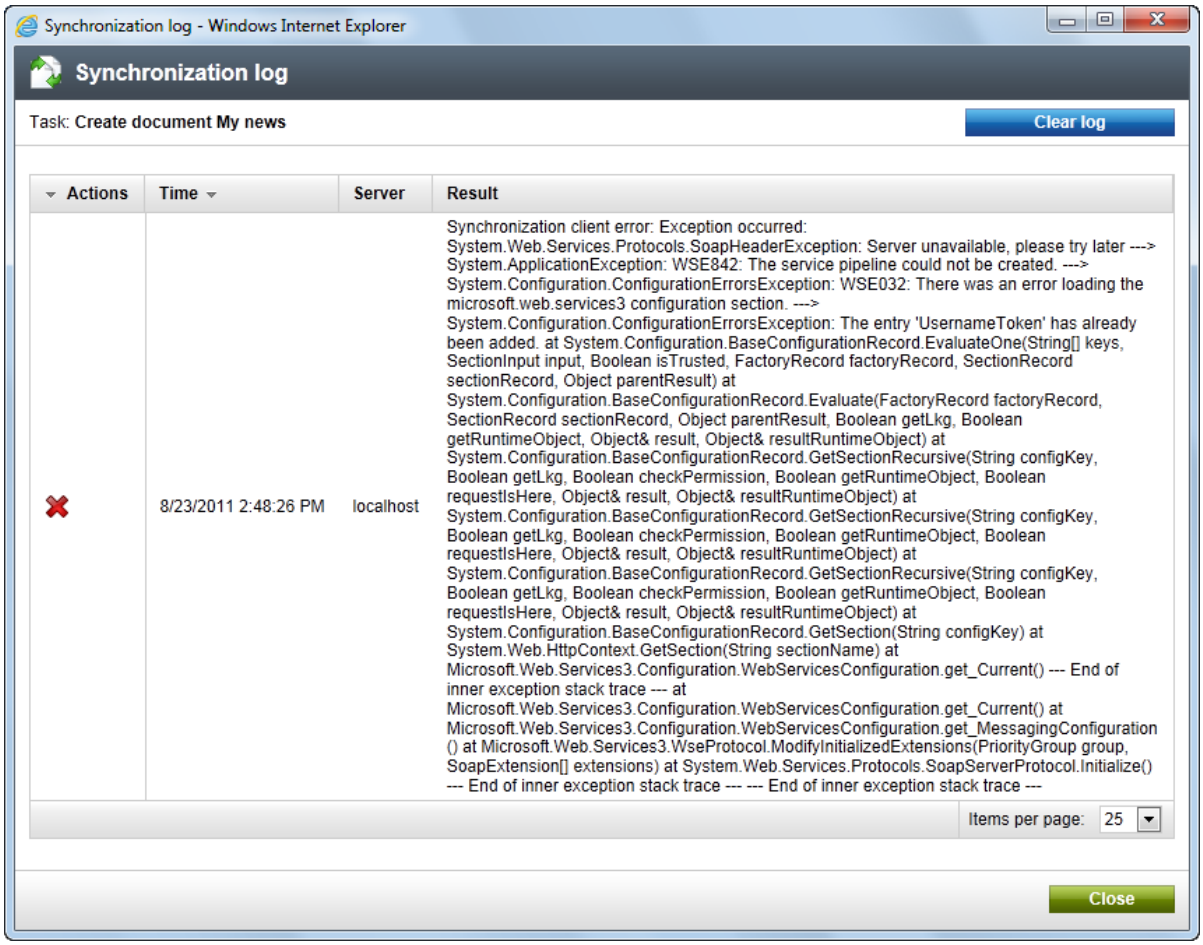
4. The websites are now accessible via nested URLs as you can see in the screenshot below. The websites can be configured independently without experiencing any issues.



Additional configuration for Staging

You may come to the point where you want to set up Staging from one of these sites to another. Staging has some sections in the *web.config* that collide. Config files are inherited within the IIS virtual directories structure (even when the projects are not nested on the file system), but you cannot have the same section of *web.config* twice in the config file.

So if we configure staging from the **KenticoCMS1** site to the **KenticoCMS1/KenticoCMS2** site (see details on how to configure the staging here), the inner project may have issues with the configuration. You would get an error like this:

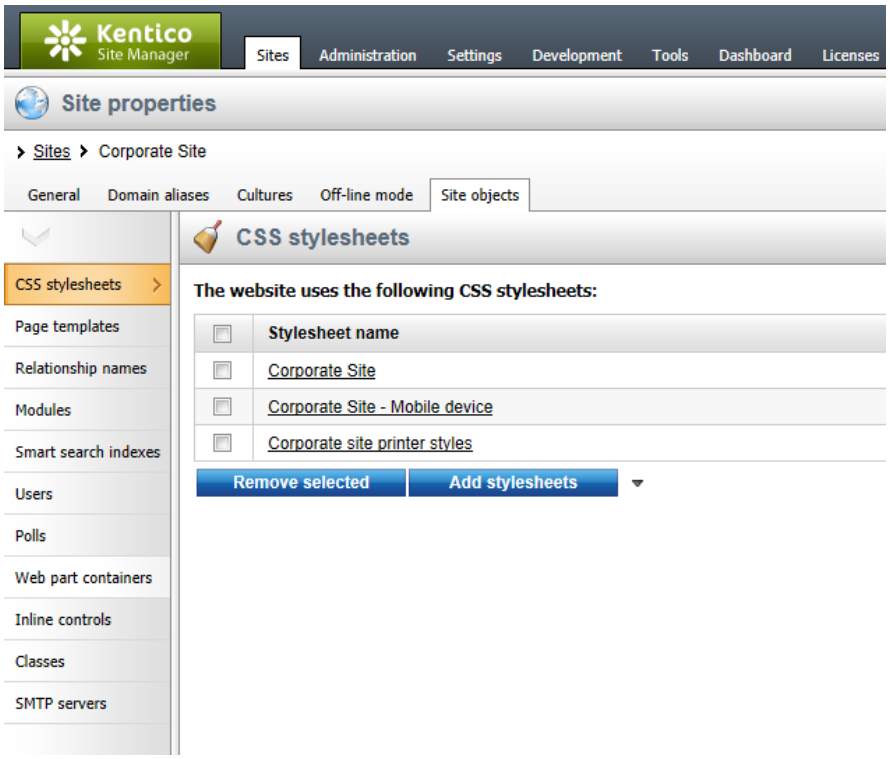


The imporant part of the error message is "**The username token has already been added**", that means that some of the **configuration is duplicate**.

User name token authentication is defined in the policy file which is referenced from the **<microsoft.web.services3>** section, so **remove the whole <microsoft.web.services3> section** from the **web.config** of the **inner project (Web2)**. Do not remove it from the Web1 (outer) project since this configuration will be used for both websites. After doing this, Staging should work flawlessly between the sites.

5.13 Site objects

Kentico Site Manager provides functionality that allows you to assign multiple objects to a site using a single interface. The interface is available in **Site Manager -> Sites -> Edit (✎) site -> Site objects.**



The following object types can be assigned to a site using this interface:

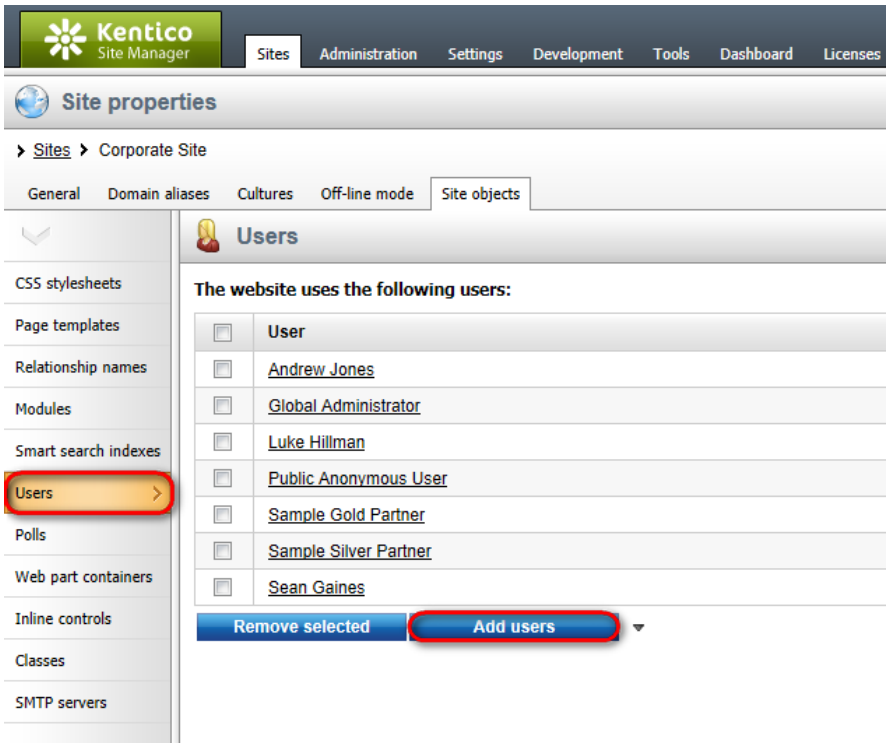
- CSS stylesheets
- Page templates
- Relationship names
- Modules
- Smart search indexes
- Users
- Polls
- Web part containers
- Inline controls
- Classes
- SMTP servers

Example

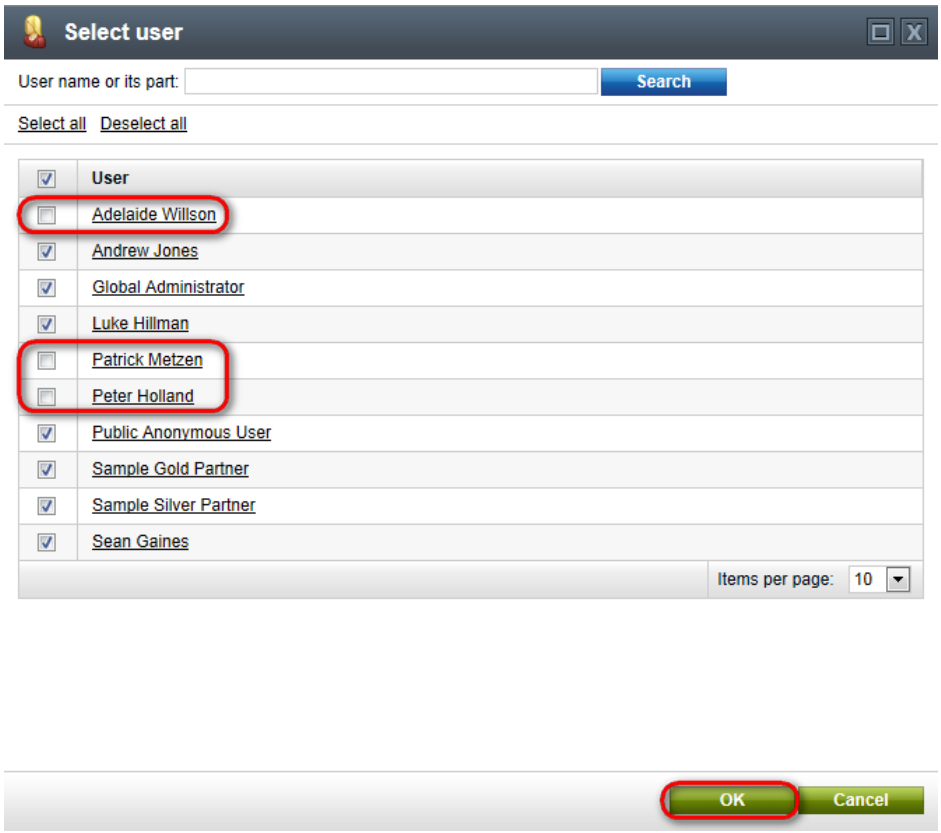
You may have created a new site using the Kentico CMS Site Manager. Now, you would like to have several of the users that you have created for a different site added to the new one.

1. Switch to **Site Manager -> Sites -> Edit (🖉) site -> Site objects.**

2. Select **Users** (you will see a list of existing users) and then click on **Add users.**



3. The users created for the previous site will be deselected by default. Select the users that you want to be added to the site and click **OK.**



5.14 Sites internals and API

5.14.1 Overview

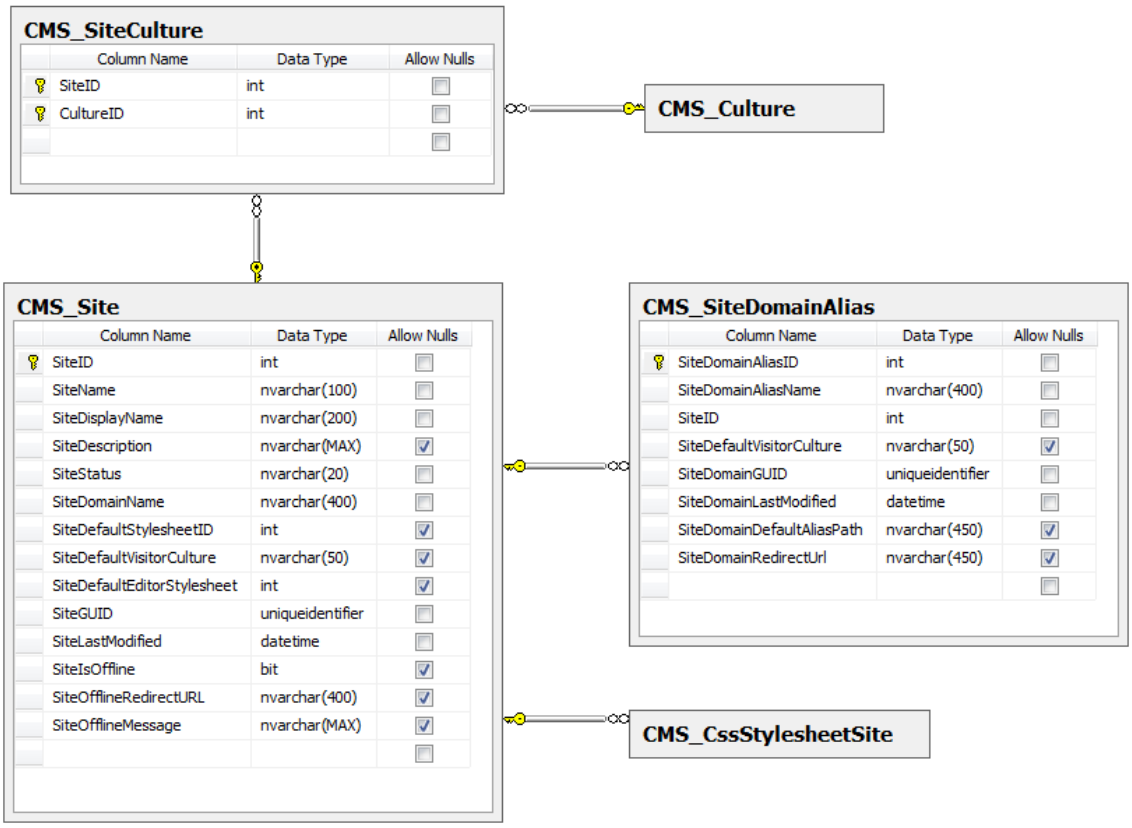
In this chapter, you will learn which database tables and API classes are used for websites. You will also see the most common API examples.

Further API-related information and examples can be found in the API programming and Kentico CMS internals section of this guide, specifically the Site management, import and export sub-chapter for website management. For detailed API documentation, such as a list of all members of the related classes, please refer to Kentico CMS API Reference.

5.14.2 Database tables

The following database tables are used to store information about sites:

- **CMS_Site** - contains records representing websites and their settings.
- **CMS_SiteCulture** - stores relationships between sites and cultures. Each entry in this table indicates that a specific culture will be available for a given site.
- **CMS_SiteDomainAlias** - contains all domain aliases of sites.



5.14.3 API classes

If you are not familiar with database table data management through Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

The classes for managing sites can be found in the **CMS.SiteProvider** namespace.

CMS_Site table API:

- **SiteInfo** - represents one site object.
- **SiteInfoProvider** - provides management functionality for sites.

CMS_SiteCulture table API:

- **CultureSiteInfo** - represents a relationship between a site and a culture.
- **CultureSiteInfoProvider** - provides management functionality for site-culture relationships.

CMS_SiteDomainAlias table API:

- **SiteDomainAliasInfo** - represents a domain alias object.
- **SiteDomainAliasInfoProvider** - provides management functionality for domain aliases.

Other classes:

- **SiteContext** - provides site-related information depending on the current context.

5.14.4 API examples

5.14.4.1 Overview

These topics show examples of how the API for managing sites can be used:

- Managing sites
- Managing site cultures
- Managing domain aliases
- Starting and stopping sites

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Sites\Default.aspx.cs***.

The site API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.GlobalHelper;
using CMS.CMSHelper;
using CMS.SiteProvider;
using CMS.DocumentEngine;
```

5.14.4.2 Managing sites

The following example creates a site.

```
private void CreateSite()
{
    // Create new site object
    SiteInfo newSite = new SiteInfo();

    // Set the properties
    newSite.DisplayName = "My new site";
    newSite.SiteName = "MyNewSite";
    newSite.Status = SiteStatusEnum.Stopped;
    newSite.DomainName = "127.0.0.1";

    // Save the site
    SiteInfoProvider.SetSiteInfo(newSite);
}
```

The following example gets and updates a site.

```
private bool GetAndUpdateSite()
{
    // Get the site
    SiteInfo updateSite = SiteInfoProvider.GetSiteInfo("MyNewSite");
    if (updateSite != null)
    {
        // Update the properties
        updateSite.DisplayName = updateSite.DisplayName.ToLower();

        // Save the changes
        SiteInfoProvider.SetSiteInfo(updateSite);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates sites.

```
private bool GetAndBulkUpdateSites()
{
    // Prepare the parameters
    string where = "SiteName LIKE N'MyNewSite%'";

    // Get the data
    DataSet sites = SiteInfoProvider.GetSites(where, null);
    if (!DataHelper.DataSourceIsEmpty(sites))
    {
        // Loop through the individual items
        foreach (DataRow siteDr in sites.Tables[0].Rows)
        {
            // Create object from DataRow
            SiteInfo modifySite = new SiteInfo(siteDr);

            // Update the properties
            modifySite.DisplayName = modifySite.DisplayName.ToUpper();

            // Save the changes
            SiteInfoProvider.SetSiteInfo(modifySite);
        }

        return true;
    }

    return false;
}
```

The following example deletes a site.

```
private bool DeleteSite()
{
    // Get the site
    SiteInfo deleteSite = SiteInfoProvider.GetSiteInfo("MyNewSite");

    if (deleteSite != null)
    {
        TreeProvider treeProvider = new TreeProvider(CMSContext.CurrentUser);

        // Delete documents belonging under the site
        DocumentHelper.DeleteSiteTree("MyNewSite", treeProvider);

        // Delete the site
        SiteInfoProvider.DeleteSite(deleteSite);

        return true;
    }

    return false;
}
```

5.14.4.3 Managing site cultures

The following example assigns a culture to a site.

```
private bool AddCultureToSite()
{
    // Get site and culture objects
    SiteInfo site = SiteInfoProvider.GetSiteInfo("MyNewSite");
    CultureInfo culture = CultureInfoProvider.GetCultureInfo("ar-sa");

    if ((site != null) && (culture != null))
    {
        // Add culture to site
        CultureSiteInfoProvider.AddCultureToSite(culture.CultureID, site.SiteID);

        return true;
    }

    return false;
}
```

The following example removes a culture from a site.

```
private bool RemoveCultureFromSite()
{
    // Get site and culture objects
    SiteInfo site = SiteInfoProvider.GetSiteInfo("MyNewSite");
    CultureInfo culture = CultureInfoProvider.GetCultureInfo("ar-sa");

    if ((site != null) && (culture != null))
    {
        // Delete the culture site
        CultureSiteInfoProvider.RemoveCultureFromSite(culture.CultureID,
site.SiteID);

        return true;
    }

    return false;
}
```

5.14.4.4 Managing domain aliases

The following example adds a domain alias to a site.

```
private bool AddDomainAliasToSite()
{
    // Get the site object
    SiteInfo site = SiteInfoProvider.GetSiteInfo("MyNewSite");
```

```
    if (site != null)
    {
        // Create new site domain alias object
        SiteDomainAliasInfo newAlias = new SiteDomainAliasInfo();

        // Set the properties
        newAlias.SiteDomainAliasName = "127.0.0.1";
        newAlias.SiteID = CMSContext.CurrentSiteID;

        // Save the site domain alias
        SiteDomainAliasInfoProvider.SetSiteDomainAliasInfo(newAlias);

        return true;
    }

    return false;
}
```

The following example removes a domain alias from a site.

```
private bool DeleteSiteDomainAlias()
{
    // Get the site object
    SiteInfo si = SiteInfoProvider.GetSiteInfo("MyNewSite");

    if (si != null)
    {
        // Get the site domain alias
        SiteDomainAliasInfo deleteAlias =
SiteDomainAliasInfoProvider.GetSiteDomainAliasInfo("127.0.0.1", si.SiteID);

        // Delete the site domain alias
        SiteDomainAliasInfoProvider.DeleteSiteDomainAliasInfo(deleteAlias);

        return (deleteAlias != null);
    }

    return false;
}
```

5.14.4.5 Starting and stopping sites

The following example runs a site.

```
private bool RunSite()
{
    // Get the site
    SiteInfo site = SiteInfoProvider.GetSiteInfo("MyNewSite");
    if (site != null)
    {
```

```
        // Start site
        SiteInfoProvider.RunSite(site.SiteName);

        return true;
    }

    return false;
}
```

The following example stops a site.

```
private bool StopSite()
{
    // Get the site
    SiteInfo site = SiteInfoProvider.GetSiteInfo("MyNewSite");
    if (site != null)
    {
        // Stop site
        SiteInfoProvider.StopSite(site.SiteName);

        return true;
    }

    return false;
}
```

5.15 Web templates internals and API

5.15.1 Overview

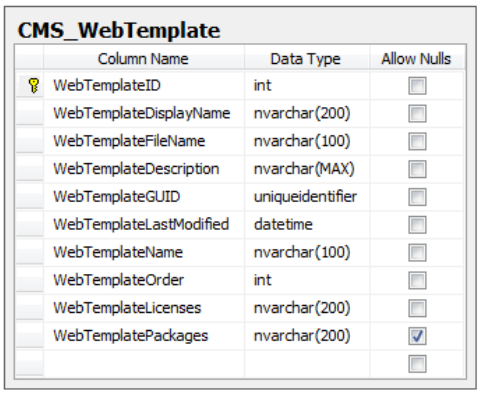
In this chapter, you will learn which database tables and API classes are used for web templates. You will also see the most common API examples.

Further API-related information and examples can be found in the API programming and Kentico CMS internals section of this guide, specifically the Site management, import and export sub-chapter for website management. For detailed API documentation, such as a list of all members of the related classes, please refer to Kentico CMS API Reference.

5.15.2 Database tables

The following database tables are used to store information about sites:

- **CMS_WebTemplate** - contains records representing web templates.



CMS_WebTemplate

Column Name	Data Type	Allow Nulls
WebTemplateID	int	
WebTemplateDisplayName	nvarchar(200)	
WebTemplateFileName	nvarchar(100)	
WebTemplateDescription	nvarchar(MAX)	
WebTemplateGUID	uniqueidentifier	
WebTemplateLastModified	datetime	
WebTemplateName	nvarchar(100)	
WebTemplateOrder	int	
WebTemplateLicenses	nvarchar(200)	
WebTemplatePackages	nvarchar(200)	☑

5.15.3 API classes

If you are not familiar with database table data management through Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

The classes for managing web templates can be found in the **CMS.SiteProvider** namespace.

CMS_WebTemplate table API:

- **WebTemplateInfo** - represents one web template object.
- **WebTemplateInfoProvider** - provides functionality for management of web templates.

5.15.4 API examples

5.15.4.1 Overview

These topics show examples of how the API for managing web templates can be used:

- Managing web templates

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Development\WebTemplates\Default.aspx.cs***.

The web templates API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.GlobalHelper;
using CMS.UIControls;
using CMS.SiteProvider;
```

5.15.4.2 Managing web templates

The following example creates a web template.

```
private bool CreateWebTemplate()
{
    // Create new web template object
    WebTemplateInfo newTemplate = new WebTemplateInfo();

    // Set the properties
    newTemplate.WebTemplateDisplayName = "My new template";
    newTemplate.WebTemplateName = "MyNewTemplate";
    newTemplate.WebTemplateDescription = "This is web template created by API
Exapmle";
    newTemplate.WebTemplateFileName = "~\\App_Data\\Templates\\MyNewTemplate";
    newTemplate.WebTemplateLicenses = "F;S;B;N;C;P;R;E;U;";
    newTemplate.WebTemplatePackages = "ECM;SCN;ADV;DOC;";

    // Set the web template order
    DataSet webTemplates = WebTemplateInfoProvider.GetWebTemplates(null, null, 0,
"WebTemplateID", false);
    if (!DataHelper.DataSourceIsEmpty(webTemplates))
    {
        newTemplate.WebTemplateOrder = webTemplates.Tables[0].Rows.Count + 1;
    }
    else
    {
        newTemplate.WebTemplateOrder = 1;
    }

    // Save the web template
    WebTemplateInfoProvider.SetWebTemplateInfo(newTemplate);

    return true;
}
```

The following example gets and updates the web template created by the API example above.

```
private bool GetAndUpdateWebTemplate()
{
    // Get the web template
    WebTemplateInfo updateTemplate = WebTemplateInfoProvider.GetWebTemplateInfo
("MyNewTemplate");
```

```
    if (updateTemplate != null)
    {
        // Update the properties
        updateTemplate.WebTemplateDisplayName =
updateTemplate.WebTemplateDisplayName.ToLower();

        // Save the changes
        WebTemplateInfoProvider.SetWebTemplateInfo(updateTemplate);

        return true;
    }

    return false;
}
```

The following example moves the web template created by the first example on this page down by one position in the order in which the templates are listed when creating a new website from a web template.

```
private bool GetAndMoveWebTemplateDown()
{
    // Get the web template
    WebTemplateInfo moveDownTemplate = WebTemplateInfoProvider.GetWebTemplateInfo
("MyNewTemplate");
    if (moveDownTemplate != null)
    {
        // Move template down
        WebTemplateInfoProvider.MoveTemplateDown(moveDownTemplate.WebTemplateId);

        return true;
    }

    return false;
}
```

The following example moves the web template created by the first example on this page up by one position in the order in which the templates are listed when creating a new website from a web template.

```
private bool GetAndMoveWebTemplateUp()
{
    // Get the web template
    WebTemplateInfo moveUpTemplate = WebTemplateInfoProvider.GetWebTemplateInfo
("MyNewTemplate");
    if (moveUpTemplate != null)
    {
        // Move template up
        WebTemplateInfoProvider.MoveTemplateUp(moveUpTemplate.WebTemplateId);

        return true;
    }
```

```
    return false;
}
```

The following example gets multiple web templates based on a WHERE condition and bulk updates them.

```
private bool GetAndBulkUpdateWebTemplates()
{
    // Prepare the parameters
    string where = "WebTemplateName LIKE N'MyNewTemplate%'";

    // Get the data
    DataSet templates = WebTemplateInfoProvider.GetWebTemplates(where, null);
    if (!DataHelper.DataSourceIsEmpty(templates))
    {
        // Loop through the individual items
        foreach (DataRow templateDr in templates.Tables[0].Rows)
        {
            // Create object from DataRow
            WebTemplateInfo modifyTemplate = new WebTemplateInfo(templateDr);

            // Update the properties
            modifyTemplate.WebTemplateDisplayName =
modifyTemplate.WebTemplateDisplayName.ToUpper();

            // Save the changes
            WebTemplateInfoProvider.SetWebTemplateInfo(modifyTemplate);
        }

        return true;
    }

    return false;
}
```

The following example deletes the web template created by the first example on this page.

```
private bool DeleteWebTemplate()
{
    // Get the web template
    WebTemplateInfo deleteTemplate = WebTemplateInfoProvider.GetWebTemplateInfo
("MyNewTemplate");

    // Delete the web template
    WebTemplateInfoProvider.DeleteWebTemplateInfo(deleteTemplate);

    return (deleteTemplate != null);
}
```

Part VI

Website settings

6 Website settings

6.1 Overview

Site settings

- **Built-in settings** - these settings are added during the installation process and cannot be modified. They are used to set the built-in modules, same as other features of the whole CMS. You can configure them in **Site Manager -> Settings**. Please refer to the Configuring settings topic for more details.

- **Custom settings** - the user can create their own settings for modules and other functionality they added to the CMS. This can be done in **Site Manager -> Development -> Custom settings**; please refer to the Adding custom settings and Managing custom settings topics for more details. If you need to configure custom settings, you can do it the same way like with the built-in settings, i.e. in **Site Manager -> Settings**; please refer to the Configuring settings topic for more details.

Web.config settings

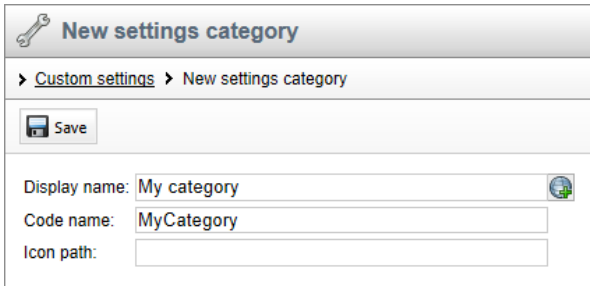
These settings are used for low-level configuration of the CMS system. More details can be found in Appendix B - Web.config parameters.

6.2 Adding custom settings

Here you will learn how to add your own settings to a website. Please note that you first need to create a category with a group within where a settings key can be put.

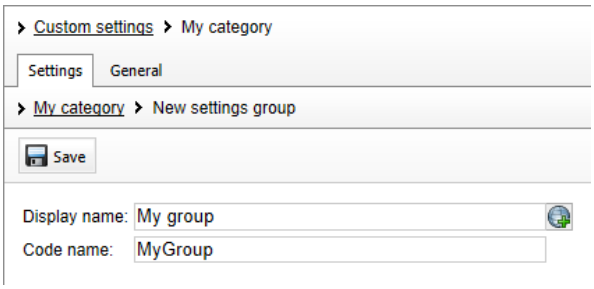
Adding custom setting categories

If you need to create a new category, go to **Site Manager -> Development -> Custom settings**, click **New category**, enter category properties and click **Save**.



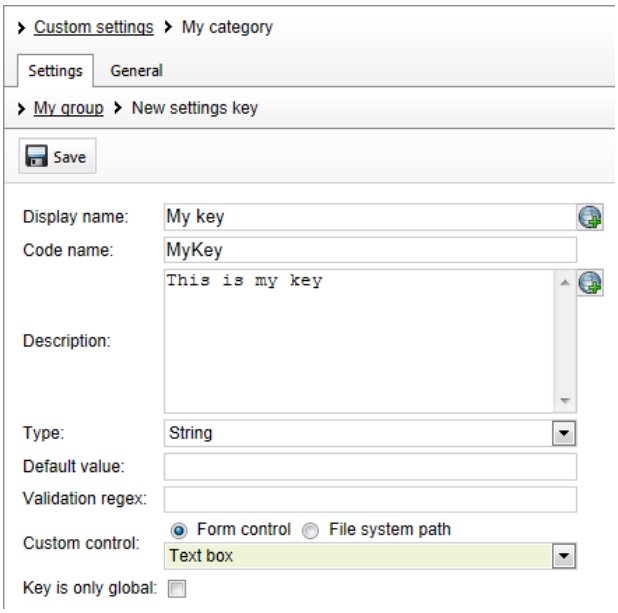
Adding custom setting groups

If you need to create a new group, in the **Custom settings** categories tree, select a category and on its **Settings** tab click **New settings group**. Enter group properties and click **Save**.



Adding custom setting keys

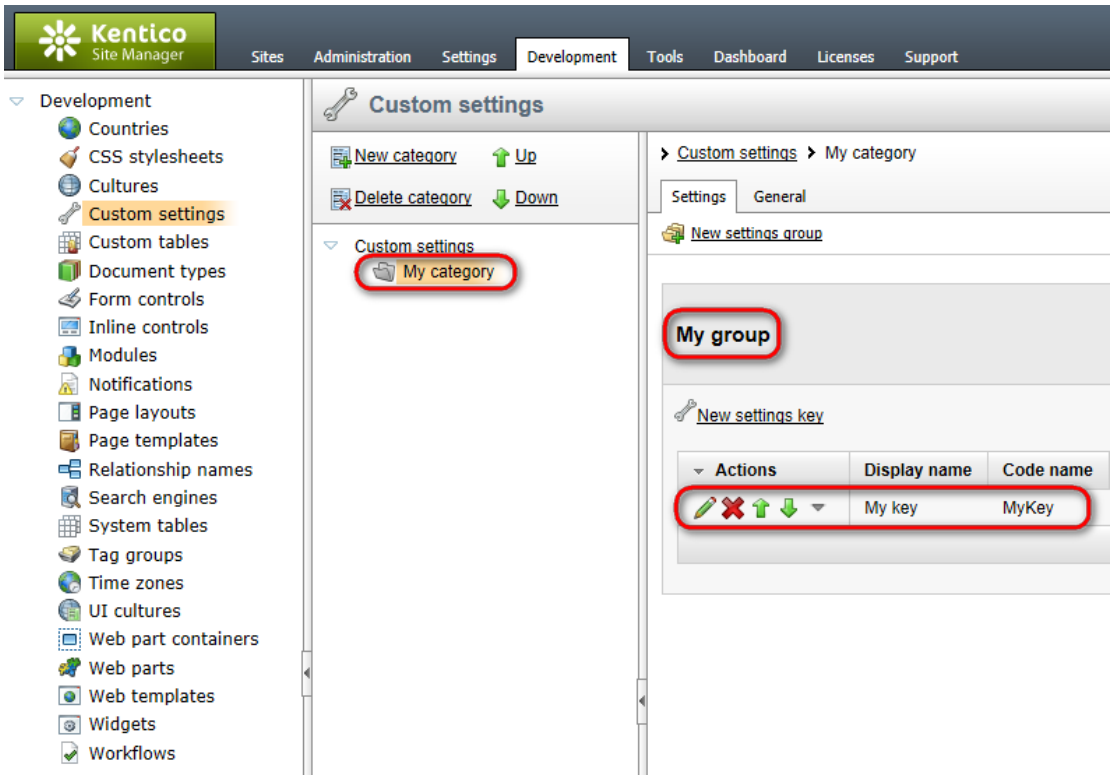
On the **Settings** tab of the selected category click **New settings key** beside the group where you would like to add the particular key. Enter key properties and click **Save**. For more information on these properties, please refer to the Managing custom settings topic.



Result

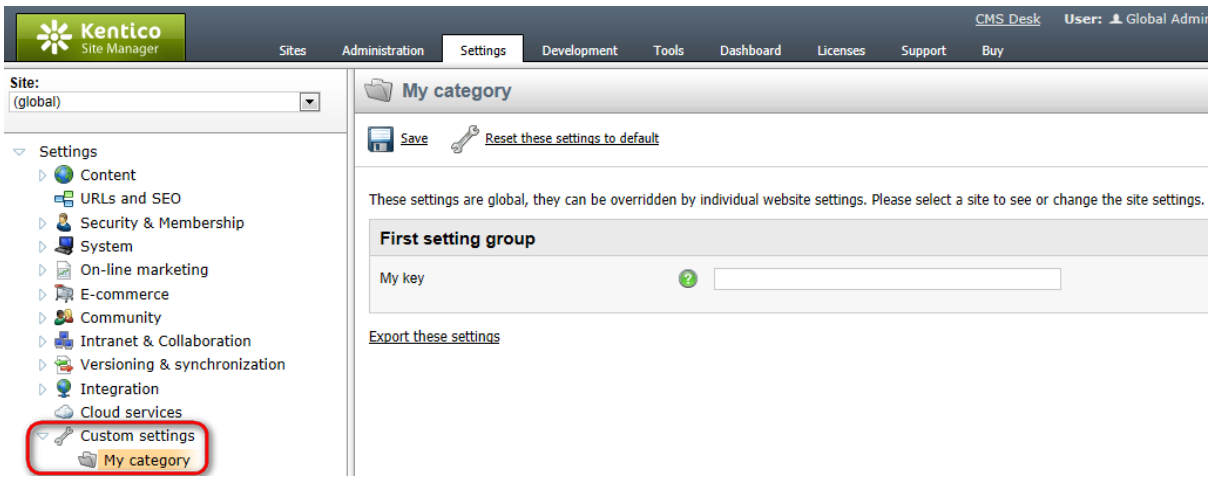
The following screenshots show the result of adding a custom category, group and key:

1. In **Site Manager -> Development -> Custom settings**.



If you would like to learn more about custom settings management, please refer to the Managing custom settings topic.

2. In **Site Manager -> Settings**.



If you would like to learn how you can configure your custom settings, please refer to the Configuring settings topic.

Please note

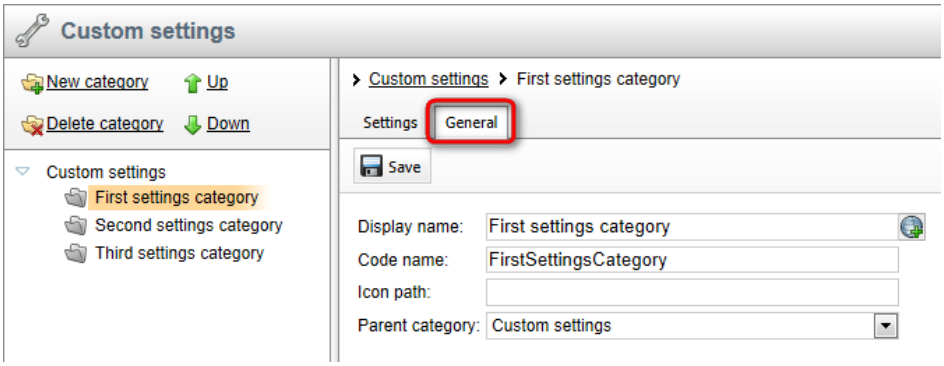
All custom settings added according to the description given in this topic can be configured along with the built-in settings in **Site Manager -> Settings**; if you would like to learn how to do it, please refer to the Configuring settings topic.
By default, there are no custom settings available after installation.

6.3 Managing custom settings

Here you will learn how to manage custom settings categories, groups and keys on your website. This can be done in **Site Manager -> Development -> Custom settings**.

Managing custom settings categories

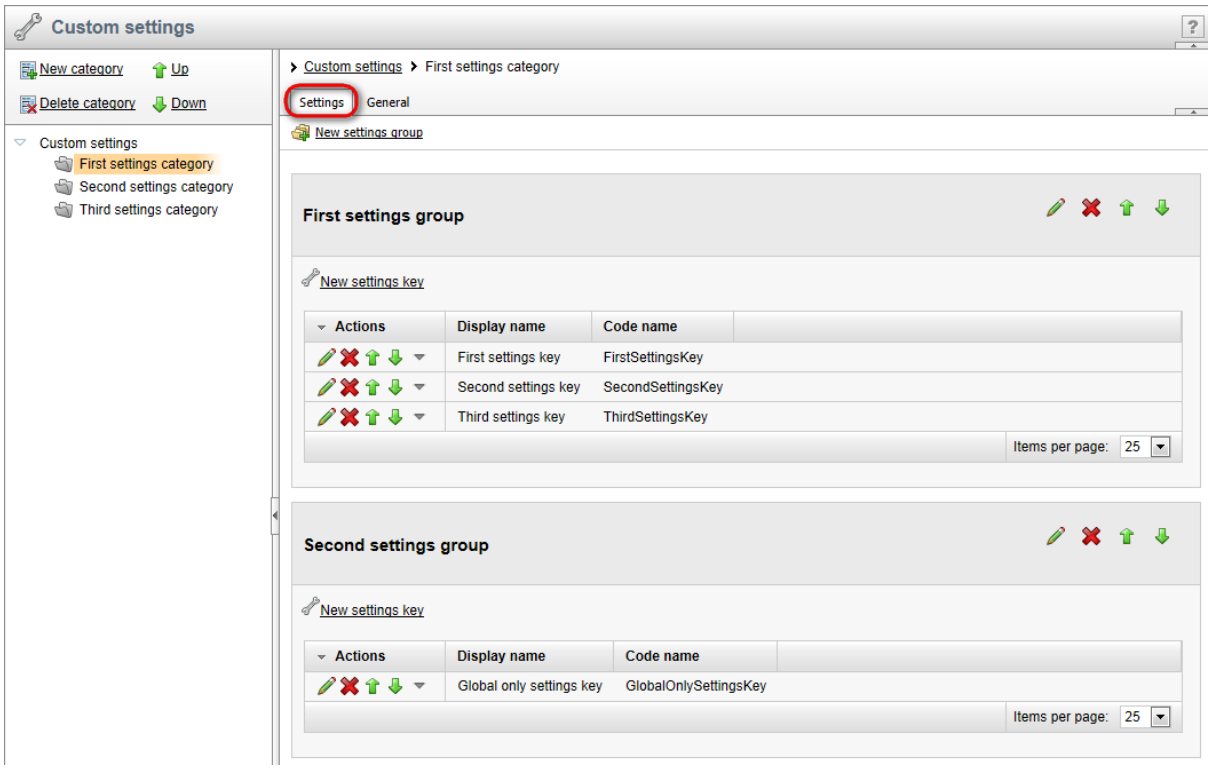
If you need to edit properties of a category, click the category in the categories tree. On the **General** tab, you can modify its general properties:



Category properties:

Display name	The name of the category displayed to website administrators.
Code name	The name of the category used in the code.
Icon path	The icon displayed next to the category caption; you can enter either a full relative path beginning with ~ (e.g. *~/App_Themes/Default/Images/CMSModules/list.png*) or a short path beginning under the *Images* folder of the current skin (e.g. *CMSModules/list.png*).
Parent category	The category where this custom category belongs; you can select another parent category to move the custom category under a different category.

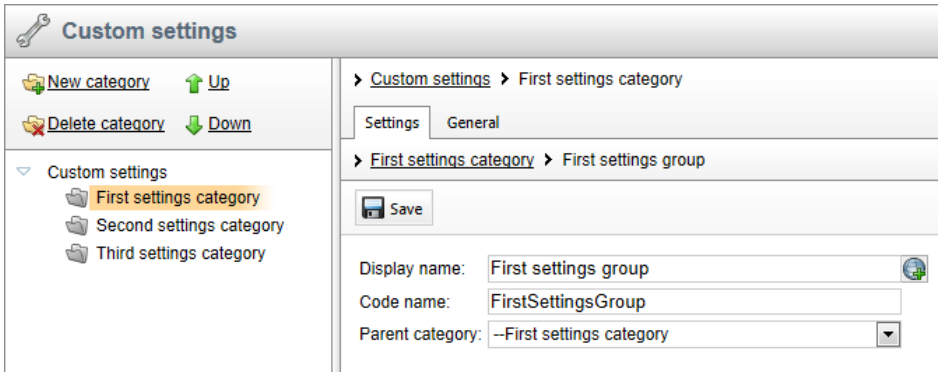
The **Settings** tab allows you to manage category groups and their keys:



Custom categories can also be deleted using **Delete category** and they can be moved **Up** and moved **Down** in the **Custom settings** categories tree.

Managing custom settings groups

If you need to edit properties of a group, on the **Settings** tab of the selected category choose to **Edit** ( ) the particular group and make the required alterations:



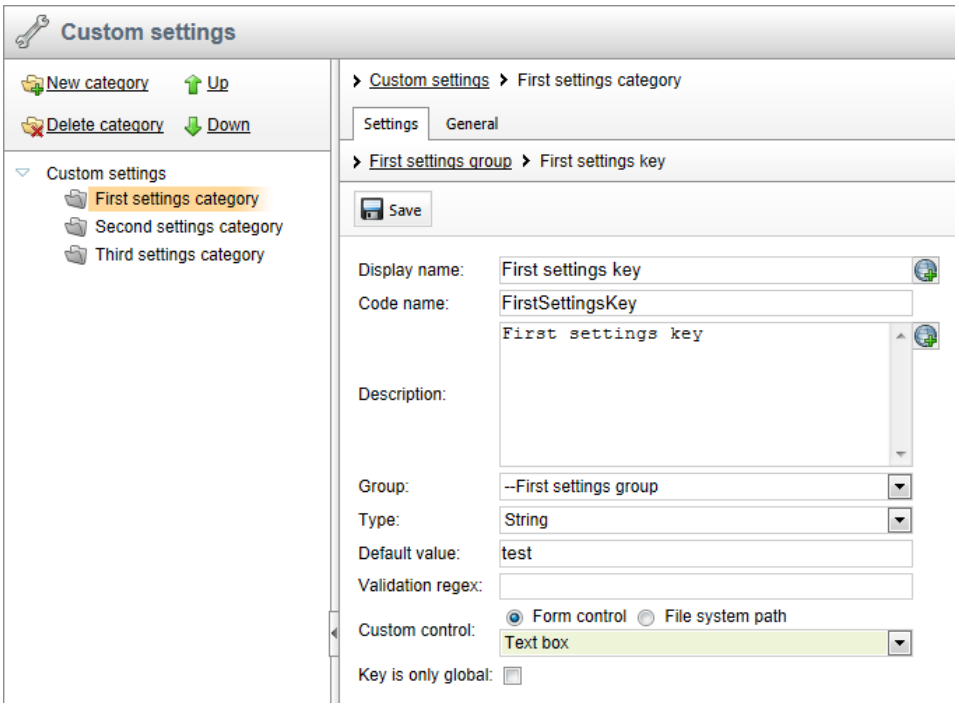
Group properties:

Display name	The name of the group displayed to website administrators.
Code name	The name of the group used in the code.
Parent category	The category where this group belongs; you can select another parent category to move the custom group to a different category.

The group can also be **Deleted** (), **Moved Up** () and **Moved down** ().

Managing custom settings keys

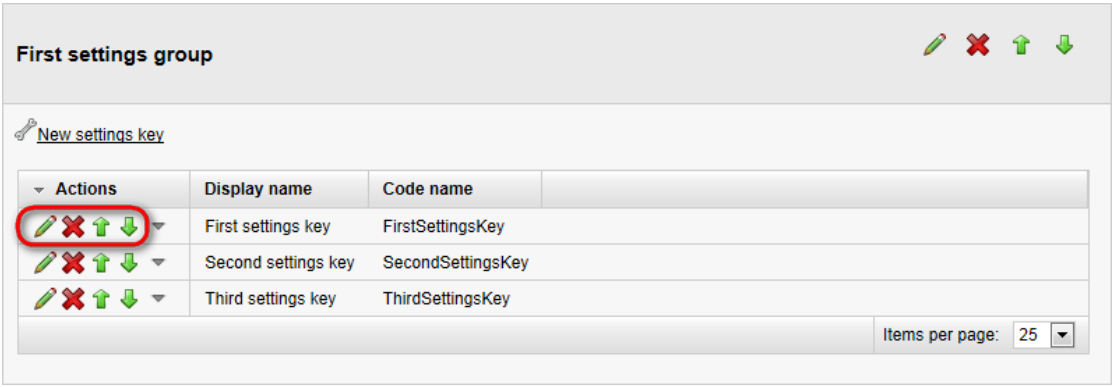
If you need to edit properties of a key, on the **Settings** tab of the respective category choose to **Edit** () the particular key and make the required alterations:



Key properties:

Display name	The name of the key displayed to website administrators.
Code name	The name of the key used in the code.
Description	Text describing the key and its purpose.
Group	The group under which this key belongs. You can move the key to a different group by selecting it from the list.
Type	The type of the key's value. You can choose the boolean, integer, double or string value type. The value entered in *Site Manager -> Settings -> Custom settings* will be validated against the *Type* property.
Default value	The default value of the key. Values of custom keys can be reset to default in *Site Manager -> Settings -> Custom settings* using the *Reset these settings to default* link.
Validation regex	If you wish to define custom validation rules for the setting's value, you can do so by entering a regular expression here.
Custom control	Specifies the control that will be used to edit the key's value in the settings form. The **Form control** option allows you to choose one of the form controls registered in the system. If you select **File system path**, you can directly specify the path to a user control file in the web project, e.g. *~/CMSFormControls/SimpleCountrySelector.ascx*.
Key is only global	If checked, all available sites will share the same value for the custom key. Additionally, the key will only be available when editing *(global)* settings.

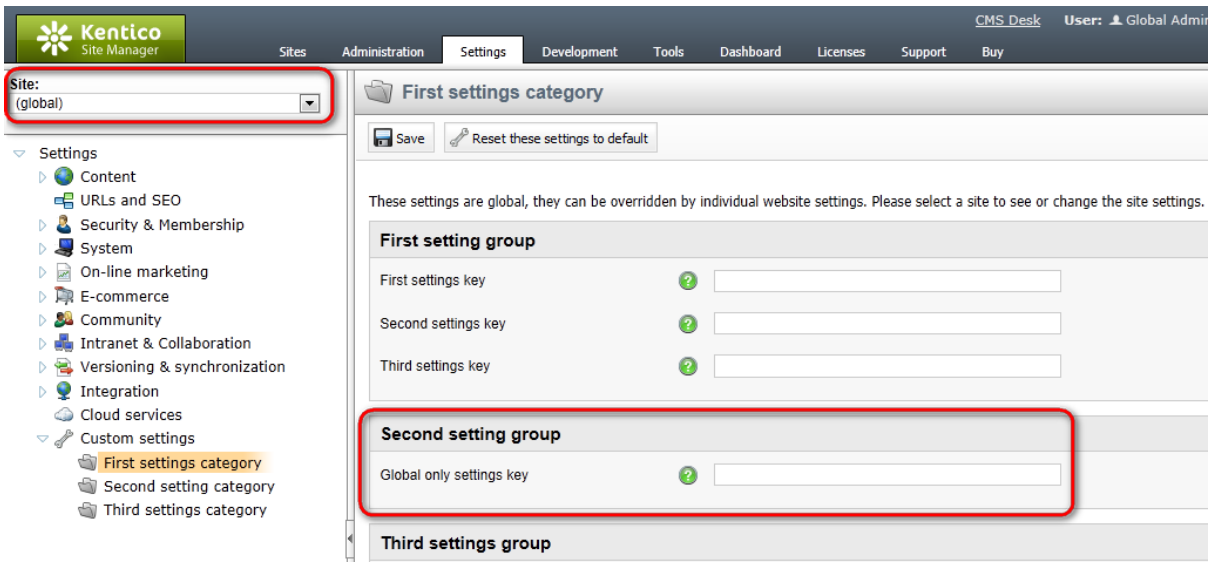
The key can also be **Deleted** (✖), **Moved Up** (⬆) and **Moved Down** (⬇).



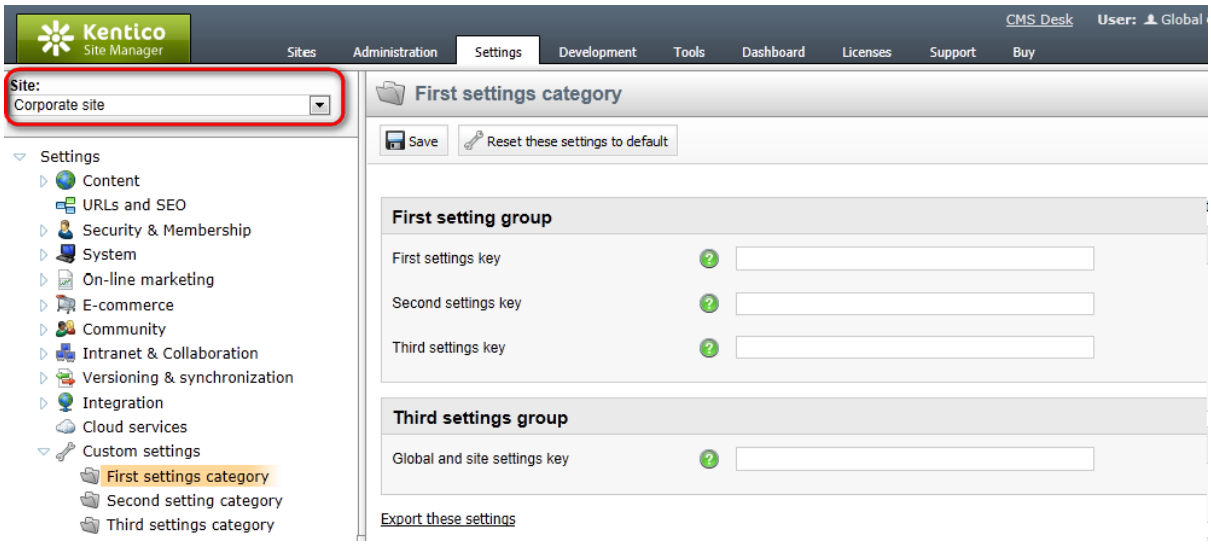
Please note

If you check the **Key is only global** checkbox while editing a key, this key will not be available in **Site Manager -> Settings** unless you choose ***(global)*** from the **Site** drop-down list. See examples below:

a) The ***(global)*** option is selected -> the *Global only settings key* is visible.

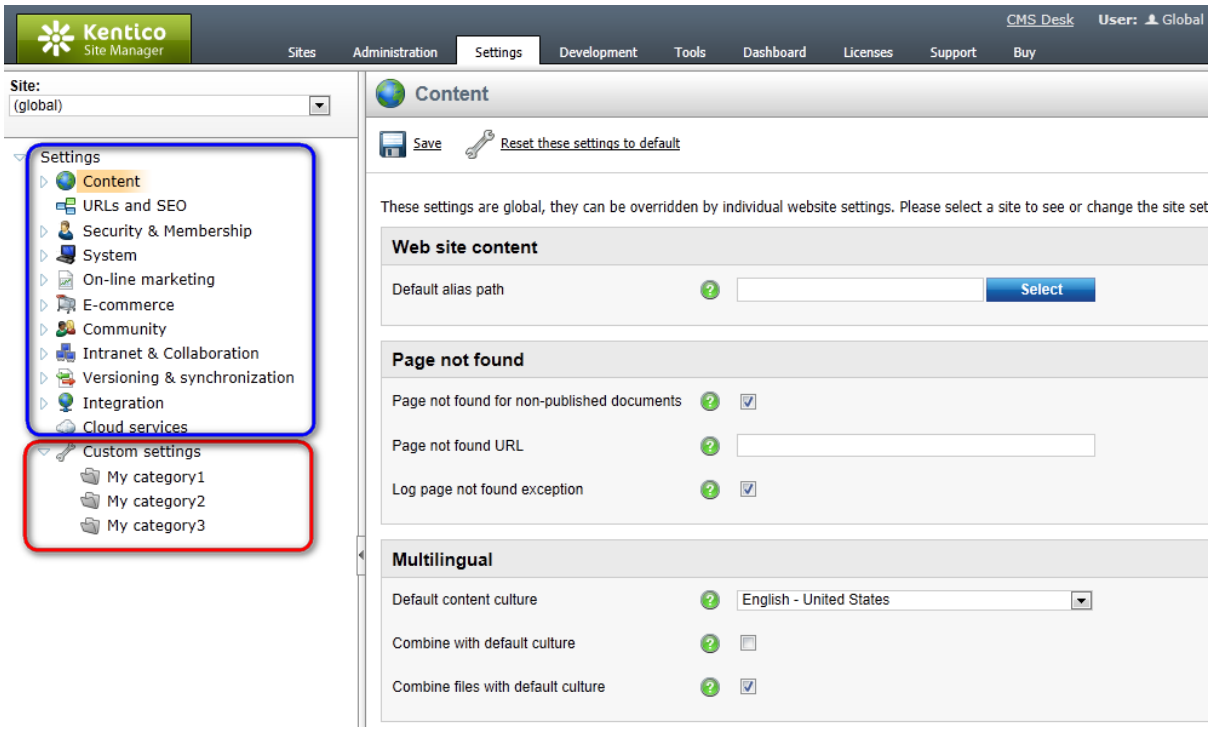


b) A particular site is selected -> the *Global only settings key* is hidden.



6.4 Configuring settings

Here you will learn how to configure settings. There are a lot of **built-in settings** which come with your Kentico CMS installation. Besides, you can also create your **custom settings** and configure them in the same way.



- **Blue box** - built-in settings which come with the installation.
- **Red box** - custom settings which can be created in **Site Manager -> Development -> Custom settings**.

Please note

By hovering over the **Help** (?) icon to the left from the key input form, you can display the description of the key as defined in **Site Manager -> Development -> Custom settings**.

How to configure settings keys

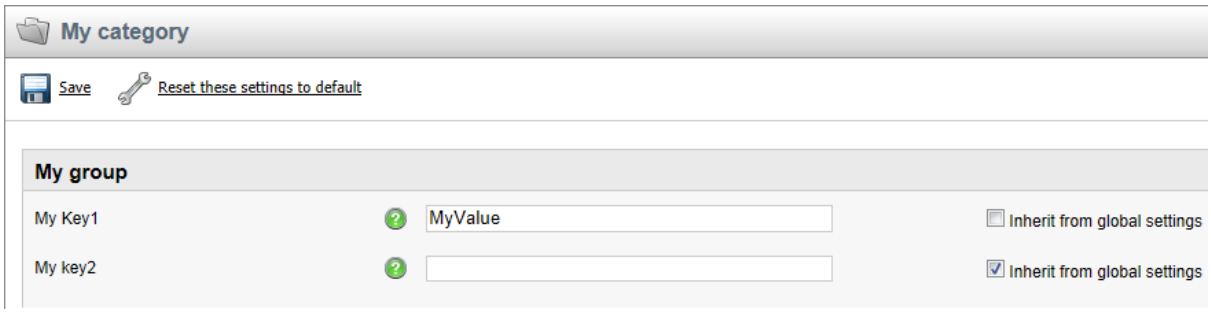
From the **Site** drop-down list, you can choose either a particular site and thus define settings for this particular site - see Site-specific settings keys or you can choose ***(global)*** and define global settings - see Global settings keys.

1. Site-specific settings keys

If you select a specific site, you can set a specific value for each key for the given site (the **Inherit from global settings** checkbox is unchecked and the value of the key is editable) or the corresponding global values can be used (the **Inherit from global settings** checkbox is checked and the value of the key is not editable). Make your site-specific settings as appropriate and click **Save**.

Please note

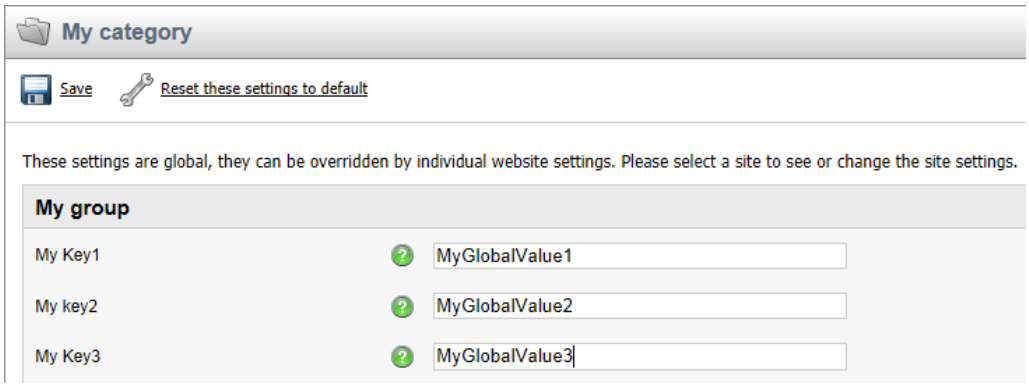
If one of the available sites is selected, all keys whose **Key is only global** property is set to true are hidden; for more information on how to set a key as global only, please refer here.



The *My key 1* key is site-specific and thus has its own value, the *My key 2* key inherits its value from global settings and the *My key 3* key is hidden because it is set as a global only key.

2. Global settings keys

If you choose ***(global)*** from the **Site** drop-down list, you can configure global settings. That is why the **Inherit from global settings** checkbox is missing and the values of all keys are editable. Make your global settings as appropriate and click **Save**.



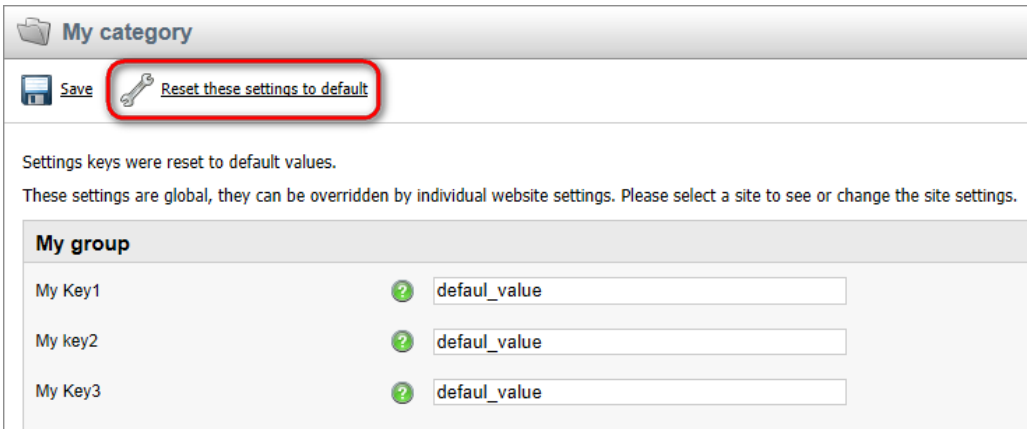
All the three keys are visible and you can edit their values.

How to reset settings keys

Here you will learn how to reset values of settings keys to their default values.

1. Click **Reset these settings to default** to reset each key in the given category.
2. In the message window that pops up, click **OK** to confirm the reset.
3. All keys in the given category have been reset to their default values as defined in the **Default value** property of the particular key.
4. Click **Save** to save the changes; for more information on how to set default values for settings

keys, please refer here.

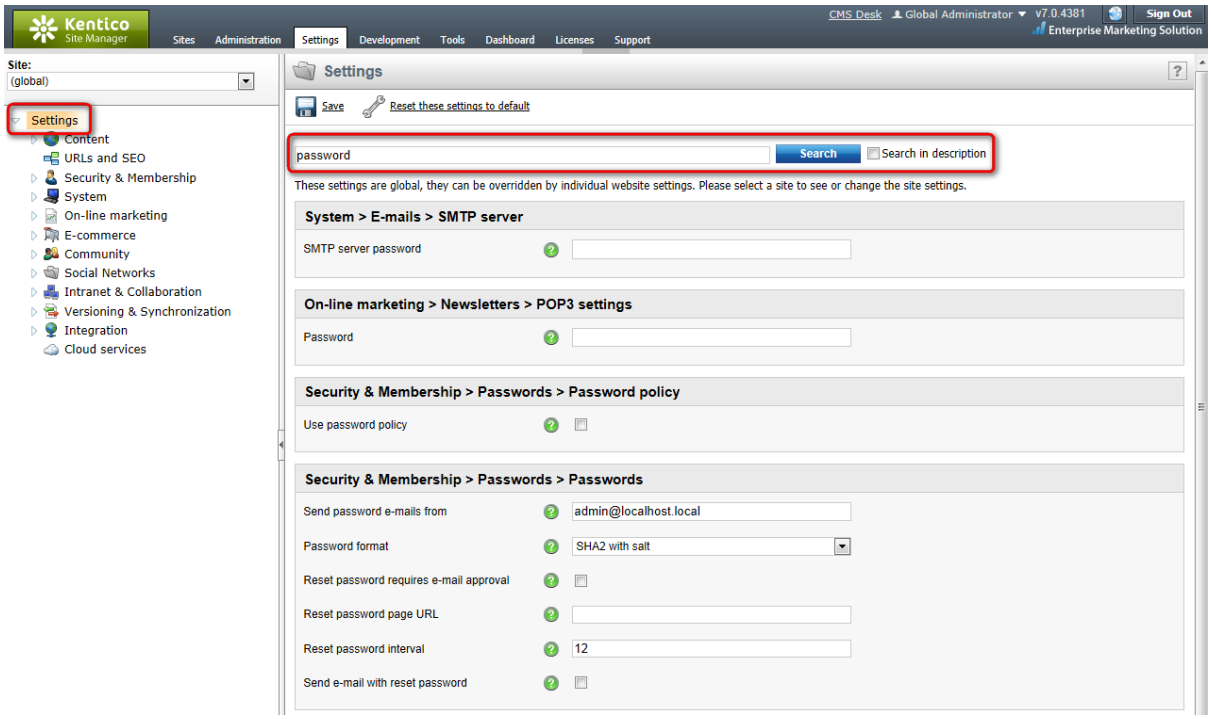


Please note

You can export the settings to a TXT file using **Export these settings** on bottom of the page. This may be useful if, for example, consulting an issue with the support staff who need the values of your settings.

Searching for specific settings

Because there is a very large number of built-in settings in Kentico CMS, it may sometimes be difficult to find a particular setting among all the categories. In these cases, you can select the root of the settings tree (**Settings**) and use the search function. Simply enter any text into the field at the top of the page and click the **Search** button. All settings that contain this text in their name will be displayed below, including the path of their category, and you can edit their values directly. If the **Search in description** box is checked, the search will also return those settings that have the given text in their description (tooltip).



6.5 Website settings internals and API

6.5.1 Overview

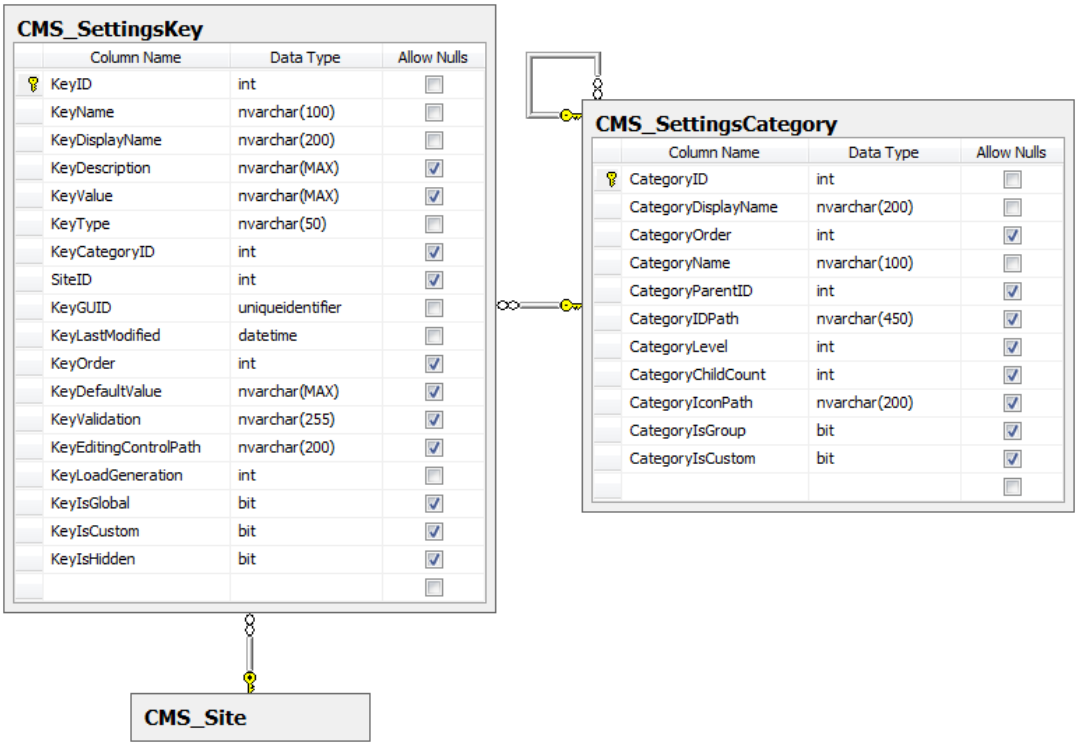
In this chapter, you will learn which database tables and API classes are used in Website settings. You will also see the most common API examples.

Advanced API examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all methods from the module classes, please refer to Kentico CMS API Reference.

6.5.2 Database tables

The following database tables are used in Website settings:

- **CMS_SettingsCategory** - contains records representing settings categories and groups.
- **CMS_SettingsKey** - contains records representing settings keys.



6.5.3 API classes

If you are not familiar with the database table data management by Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

The Website settings classes use the **CMS.SiteProvider** namespace.

CMS_SettingsCategory table API:

- **SettingsCategoryInfo** - represents one settings category or group.
- **SettingsCategoryInfoProvider** - provides management of settings categories and groups.

CMS_SettingsKey table API:

- **SettingsKeyInfo** - represents one settings key.
- **SettingsKeyProvider** - provides management of settings keys.

6.5.4 API examples

6.5.4.1 Overview

Please note

All API examples can be simulated in the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Settings\Default.aspx.cs***.

Part

Development

7 Development

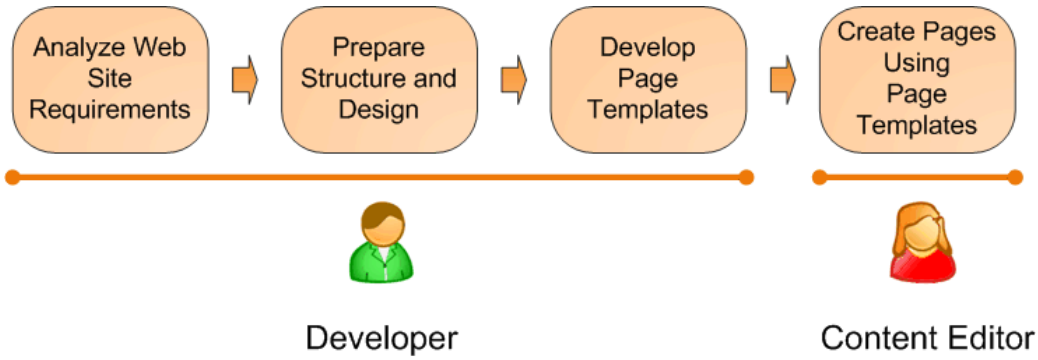
7.1 Overview

This section provides reference on various topics related to website development and system administration. Open the particular sub-chapters in order to get the related topics displayed.

7.2 Web development overview

7.2.1 The role of a web developer

The role of a web developer is described in the following figure:



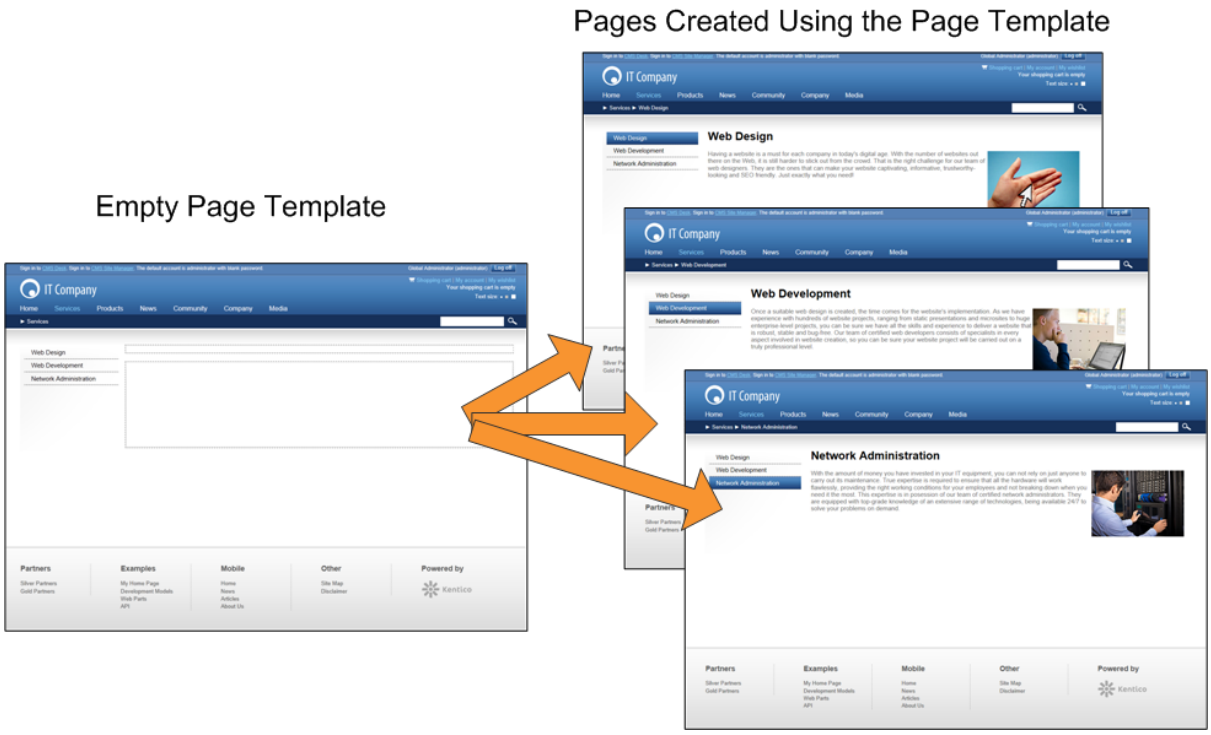
This figure shows how you develop a website and how roles are split between developers and editors. The typical development process consists of the following steps:

1. The developers analyze client requirements.
2. The developers prepare the site structure (site map) and the overall web design. During this stage, Wireframing may be used to create a rough outline of the website that can serve as a visual guide in the next steps.
3. The developers create **page templates** for every type of page required on the site (home page, solutions, products, news, etc.)
4. The content editors create new pages — they enter text and images into the page templates defined by the developer.

7.2.2 What is a page template

A page template is a predefined look for pages that allows content editors to enter the content. A single template can be re-used for multiple pages with the same structure and design, but with different content.

Templates allow content editors to focus just on content editing, without the need to take care of page formatting. They also help keep the web design consistent throughout the whole website. The following figure shows how a single page template can be used for multiple pages:



The next topic describes the development models that may be used to create various types of page templates with Kentico CMS.

7.2.3 Development model overview

Kentico CMS provides three basic development models and you can choose the one that suits your needs best:

- **Portal Engine** - this model allows you to build websites using a portal engine. It is the recommended approach for most developers since it doesn't require programming in Visual Studio. Instead, you can simply build websites using web parts in the **browser-based** user interface.
- **ASPX Templates** - this model can be chosen by advanced ASP.NET developers who prefer to create websites using standard ASP.NET architecture and using standard development tools, such as **Visual Studio**. This model requires you to be familiar with ASP.NET web form development and have at least basic programming knowledge of C# or VB.NET.
- **MVC** - this option allows developers to create websites or specific pages using the Model-View-Controller architectural pattern (based on the ASP.NET MVC framework). Working with this model requires knowledge of programming and ASP.NET MVC.

We recommend using the portal engine, but if you are an advanced .NET developer or wish to integrate existing functionality built on standard ASP.NET architecture, you may want to use ASPX templates or the MVC development model.

If required, the models can be combined on a single website. For example, you can place documents using ASPX or MVC page templates onto a portal engine website, and even insert custom ASPX pages implementing your own applications. On the other hand, special areas can be defined on ASPX templates which you can edit through the portal engine.

The following table compares the available development models:

	Portal Engine	ASPX Templates	MVC
How you work	You build the website and design pages using a browser-based interface. No programming knowledge is required for common tasks.	You build ASPX pages (web forms) that are used to display content from Kentico CMS. At least basic programming knowledge of ASP.NET and either C# or VB.NET is required.	You implement *model* objects, MVC *controller* classes and *views* for rendering pages. Requires knowledge of MVC architecture, ASP.NET and C# or VB.NET.
How you assemble pages	You use built-in or custom web parts that you place into customizable page layouts (HTML code with placeholder zones for web parts).	You use built-in or custom ASP.NET server controls and place them onto the ASPX pages. These are standard ASPX pages that are part of the web project, so you can also work with their code behind files. It is also possible to place web parts (which are actually ASCX user controls) on the page templates if the required server control is not available.	The appearance of pages is defined through MVC views, which are composed of HTML and inline code. The overall logic of pages and interaction with Kentico CMS data can be ensured through the controller and model classes. The controller and action used when a page is viewed are determined by the corresponding document's MVC page template or its specific URL pattern.
Master pages and visual inheritance	Sub-pages inherit content from their parent pages by default (so called "visual inheritance"). The inheritance can optionally be broken if you want to create a page without parent content.	Page templates may inherit content from a master page, which works just like a standard ASP.NET master page (.master file). Pages do not inherit content from their parents in the website content hierarchy, they only inherit from the master page (if it is used).	Content inheritance can be ensured through standard MVC master pages for views (master pages that inherit from the *ViewMasterPage* class). Nested views and Razor layouts are also supported. Pages do not inherit content from parents in the website content hierarchy, only from their specific master page or view.
Custom code integration and extensibility	You can create your own user controls (ASCX files) or web parts (ASCX files with a portal engine interface) if you need to	You build standard ASPX pages with code behind files, which means you can place any custom controls and code onto the page.	You prepare the content of the MVC views and the functionality of the controller and model classes in Visual Studio,

	integrate custom functionality. Any custom controls or code can be added to the web parts placed on the website.		so have full control over customization.
Advantages	• Easier and faster way to build a website. • ASP.NET programming knowledge is not required for common tasks. • You can build the whole website very quickly, using only a web browser.	• Standard ASP.NET architecture. • You can use your favorite development tools, such as Visual Studio.	• Model-View-Controller architecture. • The option of using the Razor view engine • Development via standard tools (Visual Studio).
Disadvantages	• Proprietary architecture and development process.	• Requires ASP.NET programming knowledge. • The design of the web pages cannot be fully managed via the browser-based administration interface.	• Development tasks require knowledge of ASP.NET MVC and programming. • The design of the web pages cannot be managed via the browser-based administration interface. • Kentico CMS itself is not an MVC application, so you cannot use the built-in set of components and web parts while fully maintaining MVC architecture.

7.2.4 Portal engine development model

7.2.4.1 Portal engine overview

Every page on a Kentico CMS website is based on a page template. A **page template** consists of a layout and specific instances of web parts that are configured in a certain way. The portal engine page templates are physically stored in the database, so you will not find them on the disk.

The structure of a page template is determined by its **page layout**, which can either be defined through ASCX markup or standard HTML code. The code can be used to set up a two-column, three-column, or virtually any custom layout you can think of. The page layout contains special markup tags that define the areas where developers can place web parts — so called **web part zones**. Each web part zone may contain any number of web parts.

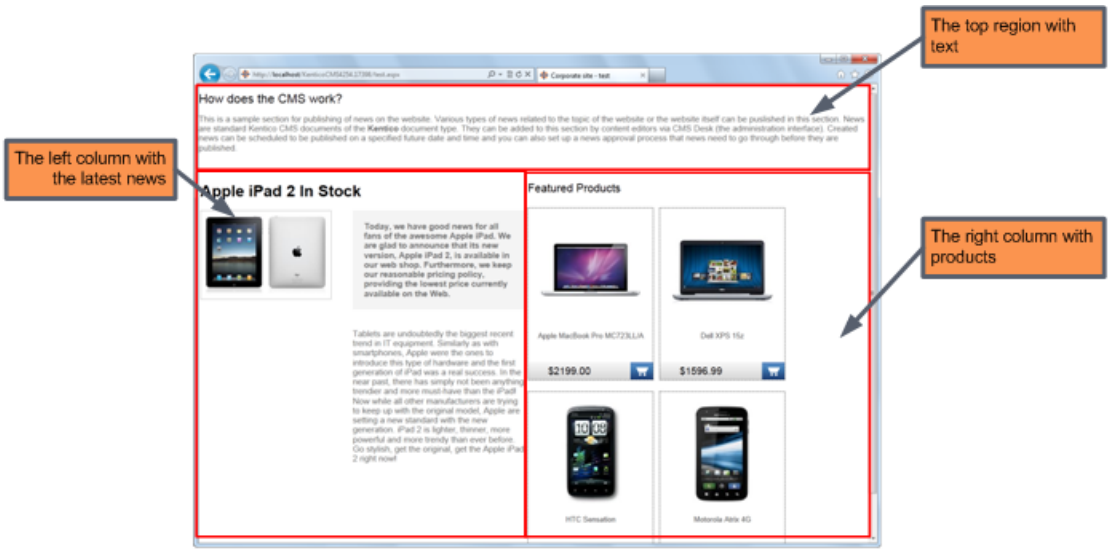
A **web part** (also called "servlet", "portlet" or "module" in other solutions) is a component that displays some type of content or provides background functionality. Technically, web parts are standard ASCX user controls with a predefined programming interface. Detailed information about web parts can be

found in the Development -> Web parts chapter.

The following figures show an example of a simple page, its page template, layout, web part zones and web parts:

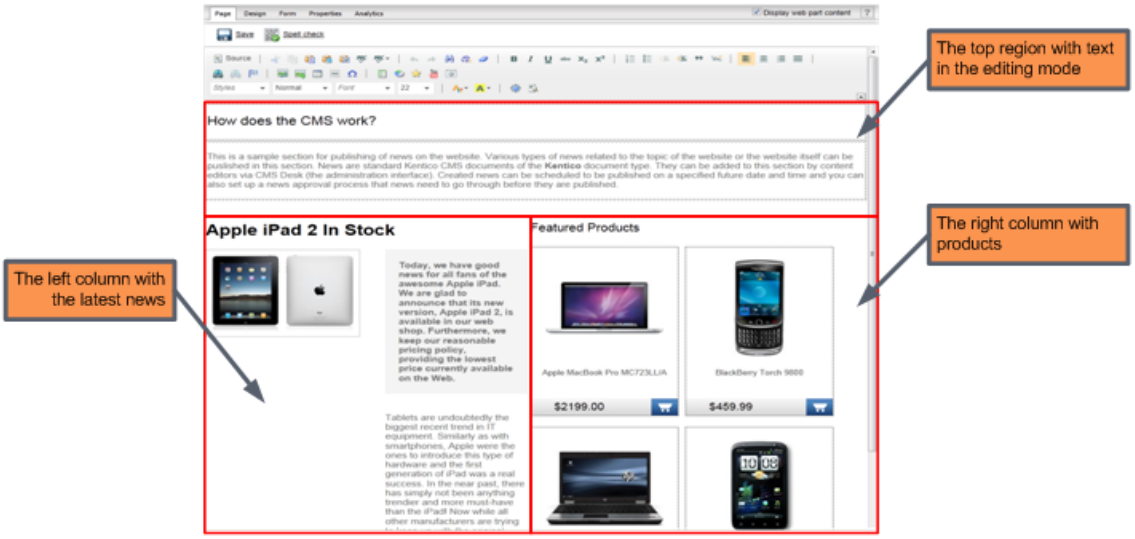
1) The sample page on the live site

This image shows a sample page on the live site. It displays text at the top, news in the left column and products on the right.



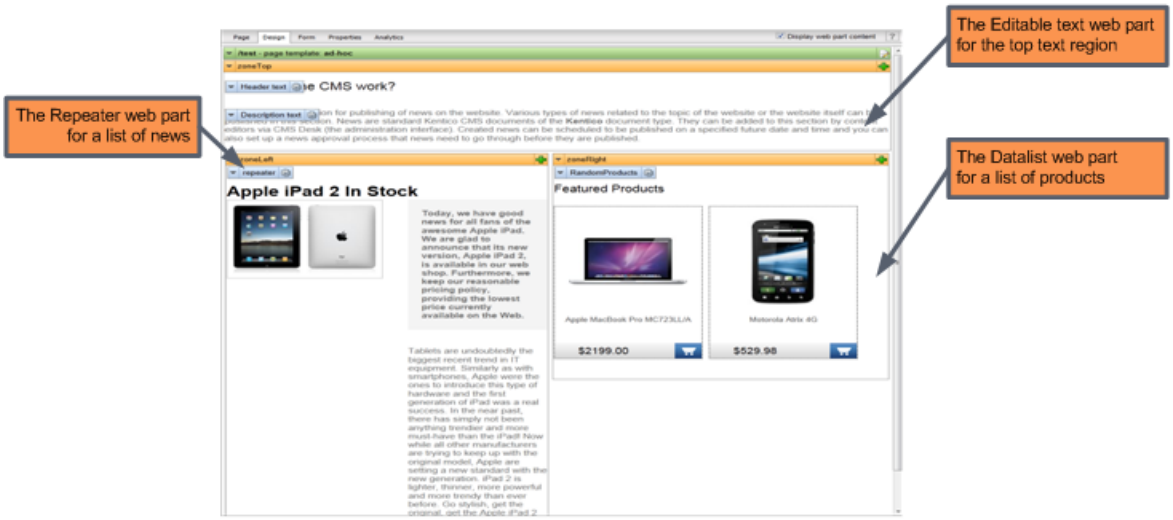
2) The sample page in editing mode

Here you can see the top section as an editable text region. The lists of news items and products are still shown as on the live site. Additional actions may also be available in the lists for editing and managing the related documents.



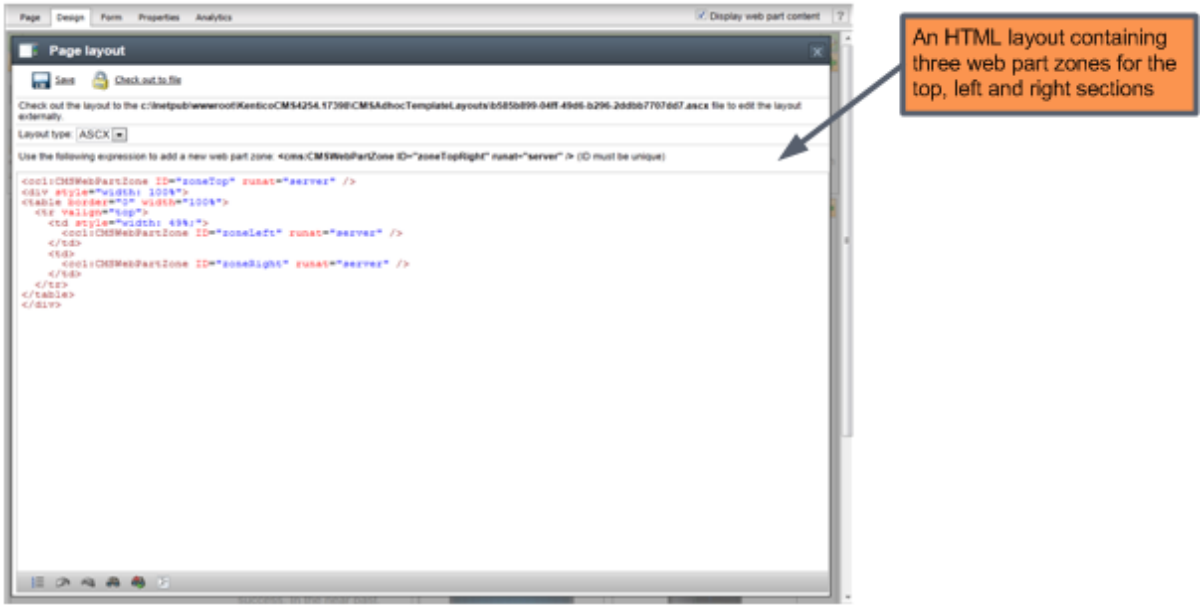
3) The page template of the sample page on the Design tab

This is the **Design** mode view of the page template used for the sample page. It consists of three web part zones - **zoneTop**, **zoneLeft** and **zoneRight**. The web part zones contain particular web parts.



4) Layout of the page template

This image shows the layout code of the page template with tags representing web part zones at the top and in the table columns. This example uses tables to build the desired layout, but you can easily implement a CSS-based layout instead — you have full control over the HTML code and CSS styles.



7.2.4.2 Creating a new page template

There are two ways to create a new page template:

- Creating a blank page using an ad-hoc template
- Cloning and modifying an existing page template

What is an ad-hoc template?

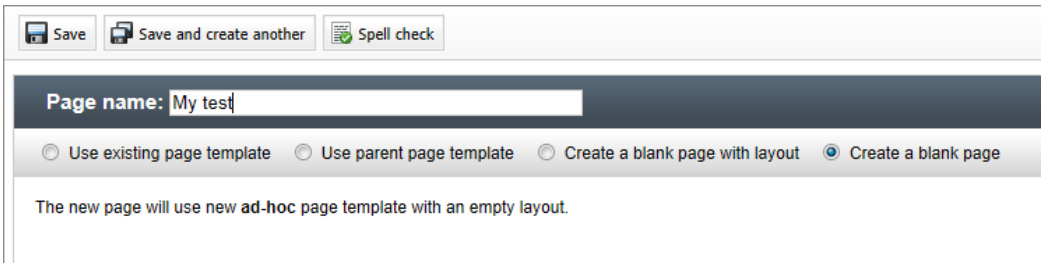
An ad-hoc template is a page template that is used only by one specific page. Ad-hoc templates do not have a particular name. If you want to re-use this type of template for multiple pages, you need to save it as a named template first.

Creating a blank page using an ad-hoc template

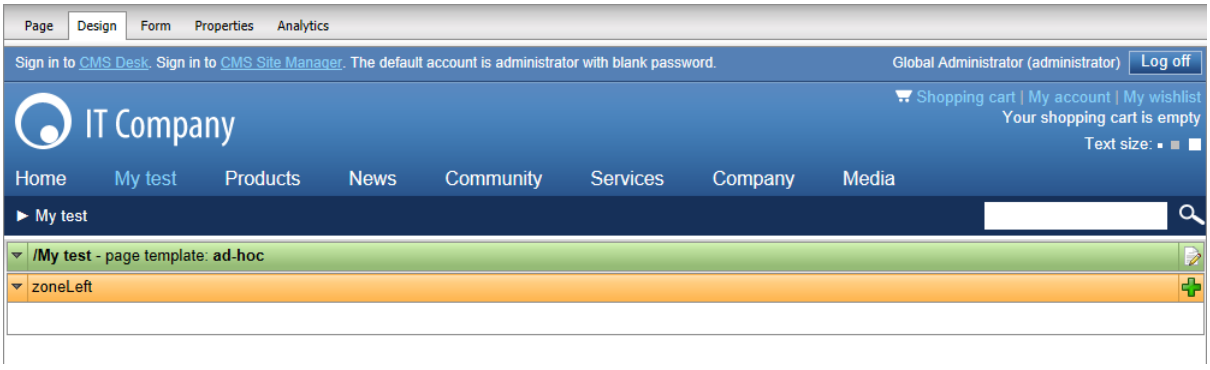
Note: It is recommended to use the sample **Corporate Site** for this example.

1. Go to **CMS Desk -> Content**, select the root of the content tree, click **New** and choose the **Page (menu item)** document type.

2. Enter *My test* into the **Page name** field, choose the **Create a blank page** option and click **Save**.

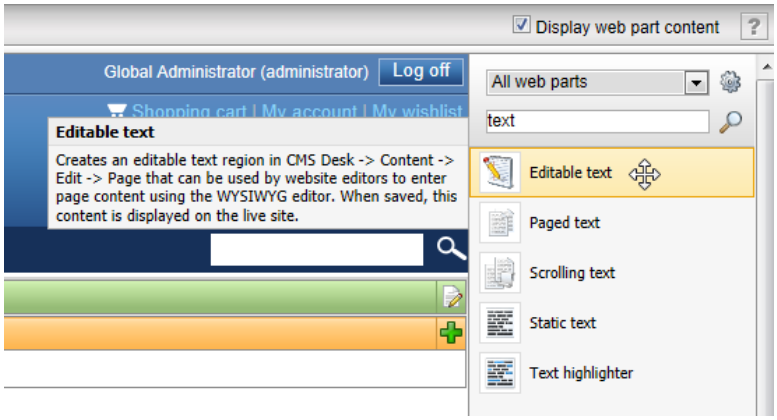


3. The page will now be added to the content tree and automatically selected. Open the **Design** tab and you will see an empty page as shown in the screenshot below.

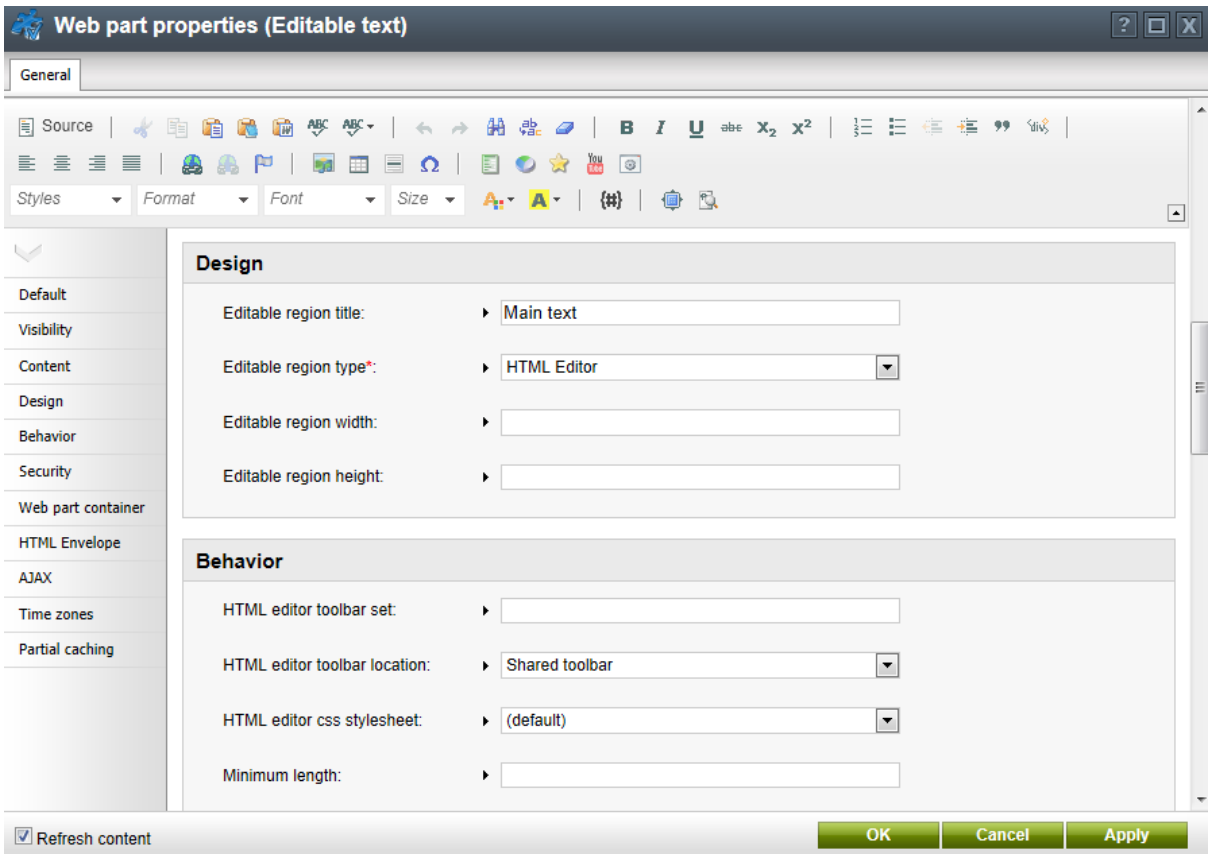


The green box with the **/My test** title displays the content of the current page template. As you can see, we are now using an ad-hoc template that was created specifically for this page. The yellow box with the **zoneLeft** title represents a web part zone, where web parts can be placed.

4. You can add web parts using the toolbar displayed on the right side of the tab (by default). Enter the word *text* into the search textbox (🔎) at the top of the toolbar, which will ensure that only web parts with this word in their name are offered.

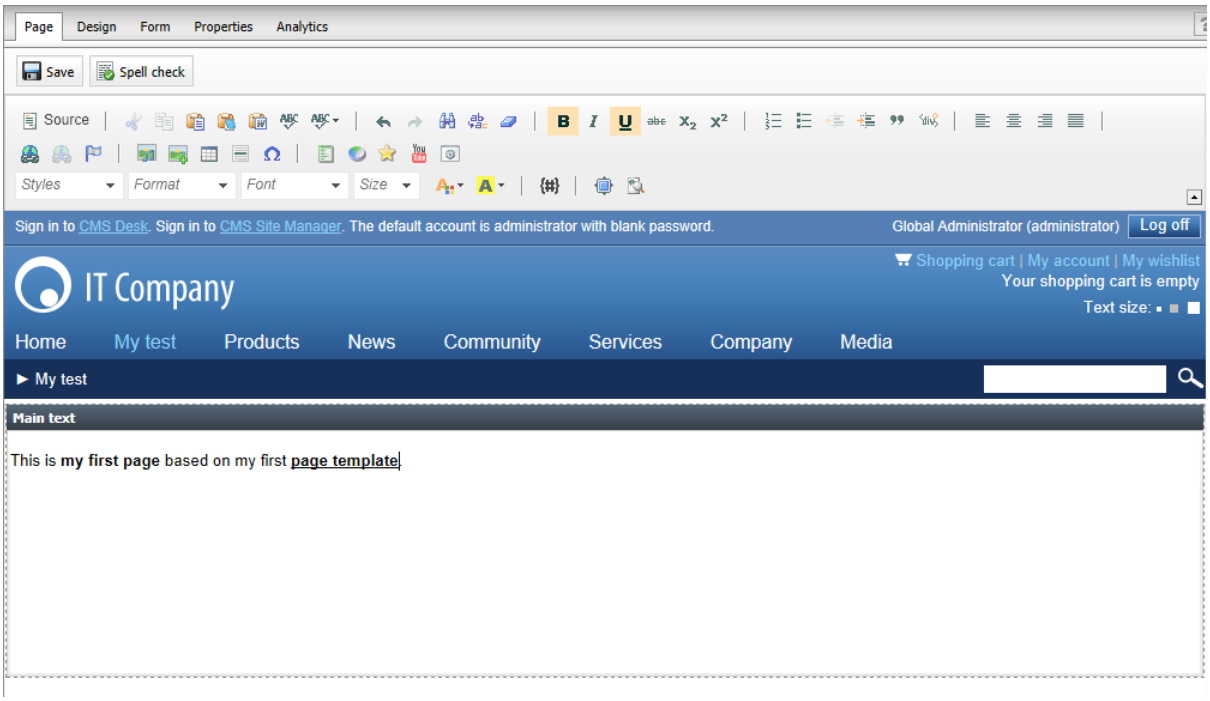


5. Now hover over the **Editable text** web part, hold down the mouse button and drag it from the toolbar into the **zoneLeft** web part zone. This web part allows you to create editable regions where editors can insert text and other content. Once you drop the web part into the zone, the **Web part properties** dialog will be opened.

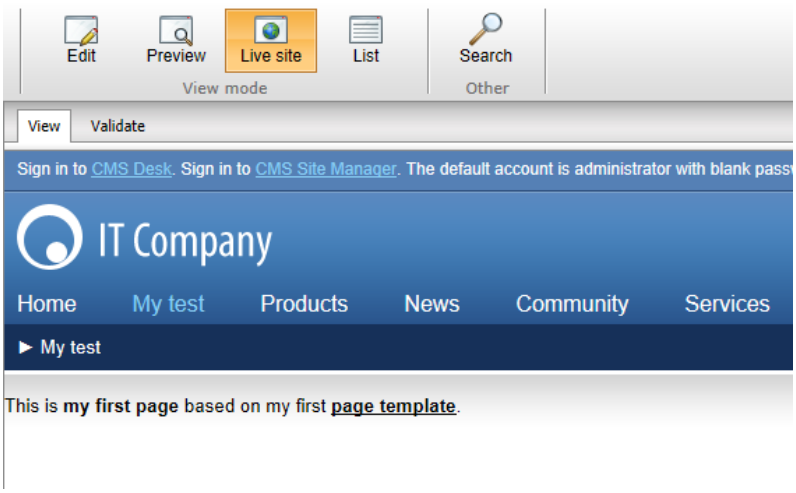


6. Scroll down to the **Editable region title** property, enter *Main text* as its value and click **OK**. The web part instance will now be added to the zone.

7. Next, switch to the **Page** tab, which displays the page in editing mode. The web part that you added provides an editable region that can be used in this mode. Enter some text into the region and click **Save**.



8. Click **Live site** in the main toolbar. You will see the live version of the page that is displayed to the website's visitors.



Result

You have created a new page based on an ad-hoc page template. The page template defines the page structure and contains the editable text web part, which enables content editors to enter text onto the page.

Important!

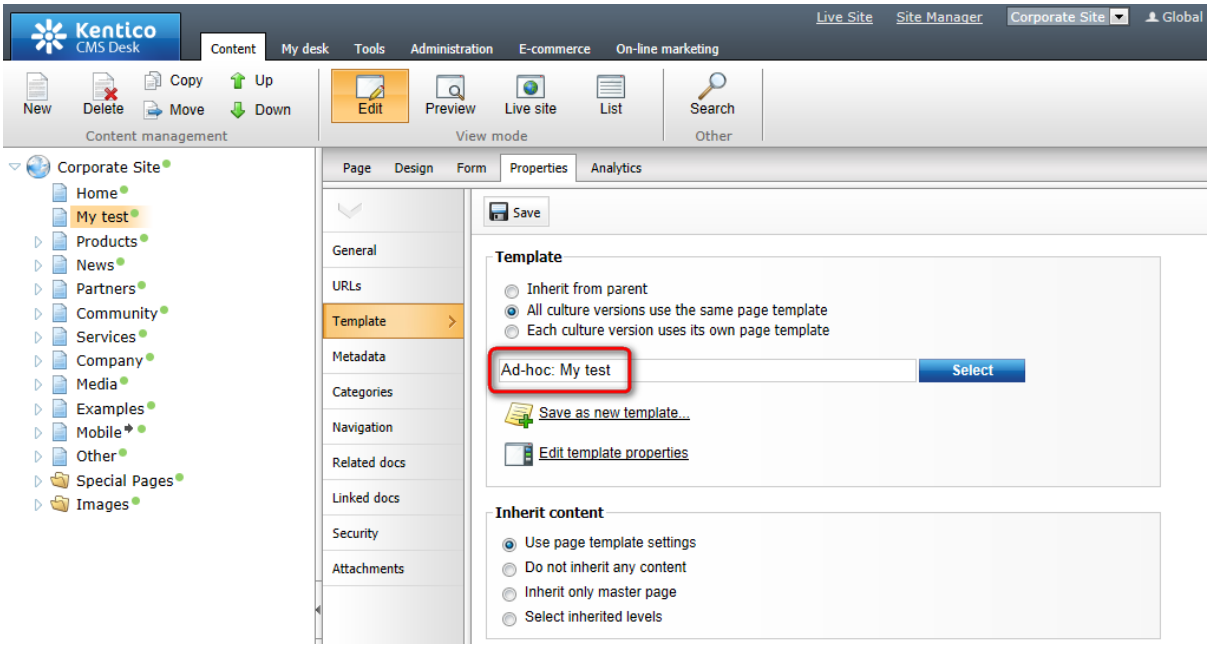
Keep in mind that when you create a page based on an ad-hoc page template and later delete the page, the corresponding ad-hoc page template is deleted as well.

7.2.4.3 Re-using an ad-hoc page template

In many cases, you may want to re-use an ad-hoc page template for other pages. In this case, you need to **save the ad-hoc template as a new re-usable template**. The following example demonstrates how this can be done:

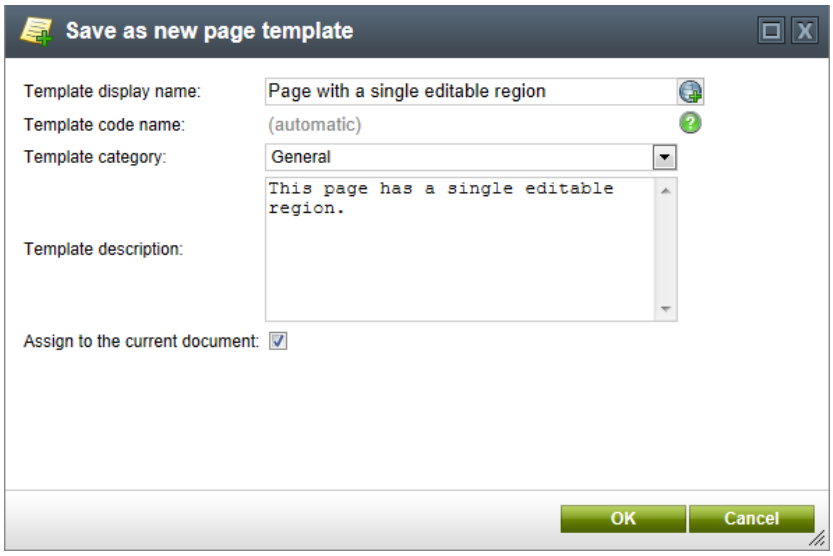
1. Open CMS Desk on the **Content** tab, switch to **Edit** mode using the main toolbar and select the new page created in the previous example.

2. Go to the **Properties -> Template** tab, where you can see that the current page is based on an ad-hoc page template.



3. Click **Save as new template**. The **Save as new page template** dialog opens. Fill in the following values:

- **Template display name**: Page with a single editable region
- **Template category**: General
- **Assign to the current document**: Yes (leave it checked)

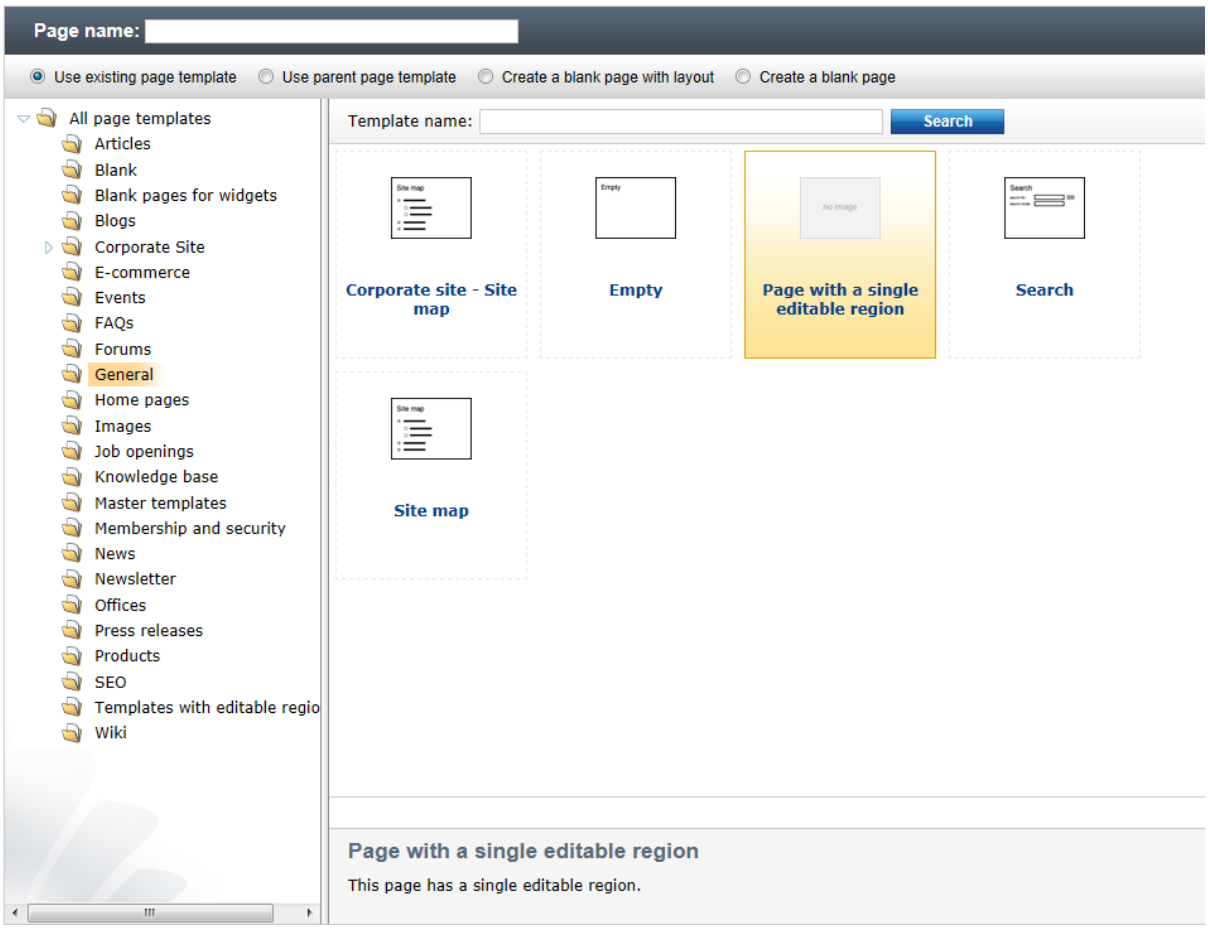


Confirm the creation of the new page template by clicking **OK**.

Result

When creating pages, you can now assign the new page template by selecting it from the **General** category. Because the **Assign to the current document** option was checked, the system also automatically assigns this template to the current page instead of the previous ad-hoc template.

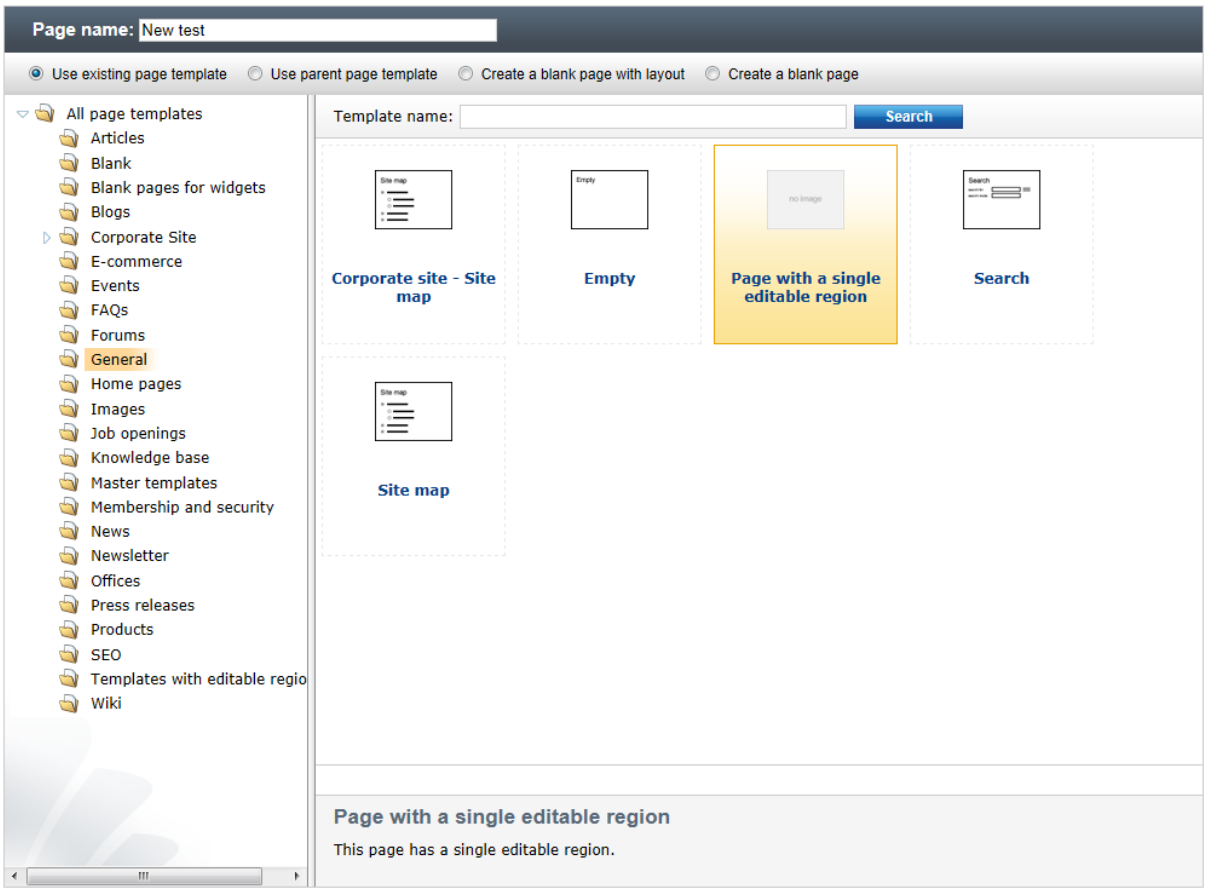
You have converted the previously created ad-hoc page template that was specifically bound to a single page into a re-usable page template that can be used for any number of pages.



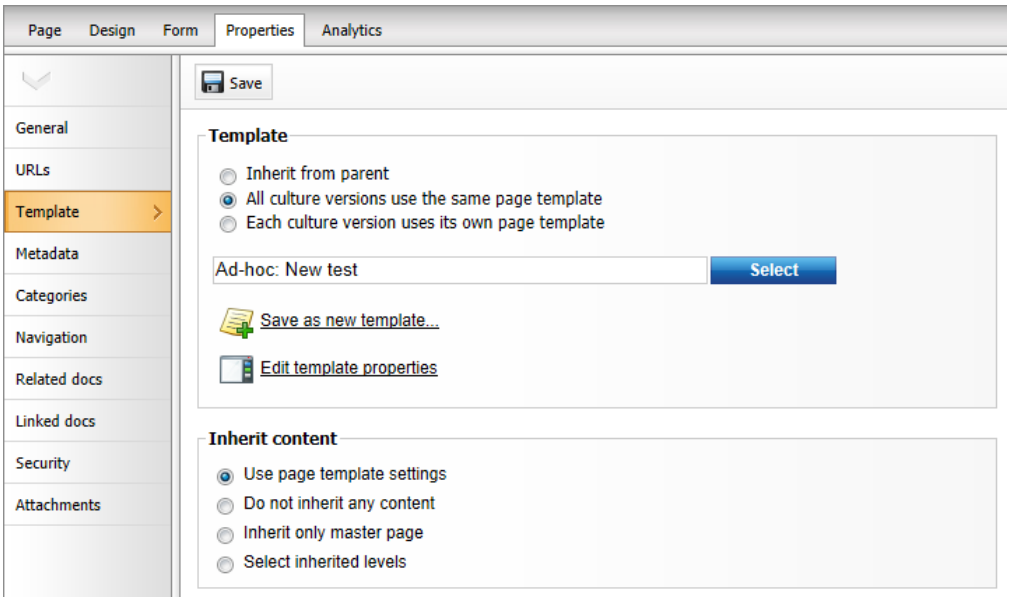
7.2.4.4 Cloning and modifying a page template

You may need to create a new page that will be similar to existing pages, but with some minor modifications of the page template. For example, if you wish to display the editable region from the previously created page template in a container box.

Instead of creating the page template from scratch, simply create a new page based on your existing page template.

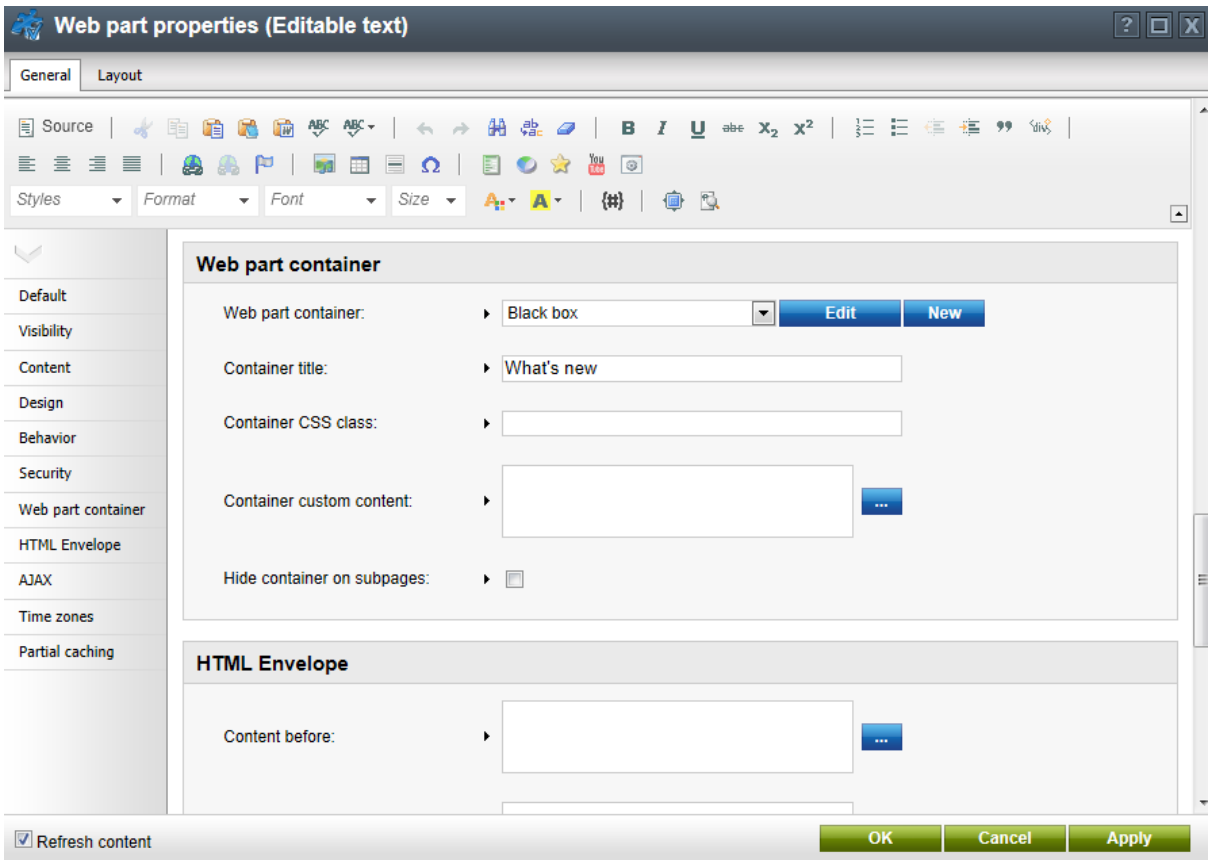


Then select the page in the content tree, navigate to its **Properties -> Template** tab and click **Clone template as ad-hoc**. A new ad-hoc template is now created and you can edit it without modifying the existing pages based on the original template.

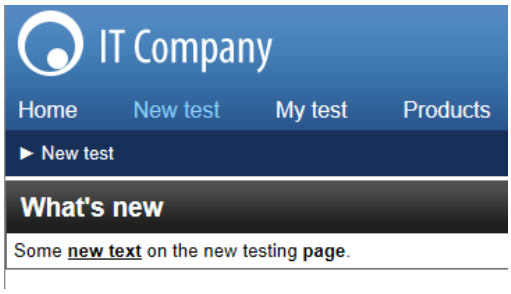


Now switch to the **Design** tab and click the **Configure** ( ) button of the **Editable text** web part. Select

the *Black box* container from the **Web part container** drop-down list and enter *What's new* into the **Container title** property.



Click **OK**. Switch to the **Page** tab and enter some text into the editable region. Click **Save** and select **Live site** mode through the main toolbar. You will see a page like this:



As you can see, the text is now surrounded by a black container. When you check the original **My test** page, you will see that it still uses the original design and it wasn't affected by the change made to the new copy of the page template.

Re-using the modified template

If you want to use the new, adjusted page template for other pages, you can save it as

a named template by clicking the **Save as new template** action on the **Properties -> Template** tab.

Important!

Please keep in mind that when you create a page based on an ad-hoc page template and later delete the page, the corresponding ad-hoc page template will also be deleted and **cannot be restored**.

7.2.4.5 Managing the page template catalog

You can manage page templates using the **Site Manager -> Development -> Page templates** interface. Here you can organize the templates into categories, configure their properties, create new templates, etc.

These configuration options can also be accessed for the templates of individual pages by selecting the appropriate document in the content tree of **CMS Desk**, and using the **Edit template properties** action on the **Edit -> Properties -> Template** tab.

Each template has the following properties:

General tab

On this tab, you can specify the following properties of the page template:

Template display name	The name of the template displayed to users.
Template code name	The name of the template used in website code.
Category	Here you can choose the category of the page template.
Template description	Description of the template shown in the page template selection catalog.
Thumbnail	Teaser (preview) image assigned to the template. It is displayed in the page template catalog, e.g. when creating a new page.
Clone as ad-hoc for new documents	When a user selects a template with this option enabled for a new page, the system automatically creates an ad-hoc copy of the template and assigns it to the page. This way, users can immediately modify the design of the new page without changing the default re-usable template. Only available if the **Template type** is set to *Portal page*. **Note**: The system always clones templates as ad-hoc when creating wireframes, even if this setting is disabled for the assigned template.
Template type	Determines the type of pages that the template can be used for. The

	following options are available: • **Portal page** - functions as a standard portal engine page template. • **ASPX page** - uses the ASPX page template development model and is based on a standard ASPX web form. If selected, the path to the source file must be entered into the **File name** property. • **ASPX + Portal page** - works the same way as the *ASPX page* option, but also supports certain portal engine functionality, such as managing web parts or widgets in predefined zones on the *Design* tab of CMS Desk directly through the browser. • **Dashboard page** - provides a template for dashboard sections of the CMS interface. This type of page template cannot be used for standard content tree pages. • **MVC** - can be used to create pages defined through an MVC controller and action, which must be specified via the additional *Default controller* and *Default action* properties. Pages based on templates of this type automatically use the live URL and view in editing mode (i.e. on the *Page* tab of CMS Desk). Please see the MVC development model chapter for more information.
Master template	Only available if the **Template type** is set to *Portal page*. Indicates if the pages that use the template should be Master pages. Master page templates are usually used for the main menu and logo of a website. Enabling this also causes the template to be selectable as the root master page template in the **New site wizard**.
Inherit content	Only available if the **Template type** is set to *Portal page*. Configures the visual inheritance as explained in the Visual inheritance topic. You can choose from the following options: • **Inherit all** - inherits the content of all parent pages • **Do not inherit any content** - displays only the current page without any parent content • **Inherit only master page** - inherits from the first master page above the page in the content tree, i.e. if there are more master pages, it inherits only from the one which is the closest above it in the hierarchy • **Select inherited levels** - inherits only content from the chosen content tree levels
File name	Only available if the **Template type** is set to *ASPX page* or *ASPX + Portal page*. Specifies the path to the **.aspx** file that the page template should be based on. The file may either be chosen using the **Select** button, or its path can be entered manually. The tilde character (~) represents the root directory of the project folder, e.g. *~/CMSTemplates/CorporateSite/Blog.aspx*
Default controller	Only available if the **Template type** is *MVC*. Sets the name of the controller class containing the MVC action that should be performed when pages using this template are accessed (without the *Controller* part at the end). For example, if the class is

	called *NewsMVCController*, enter *NewsMVC*. The system first searches for the specified class in the *CMS.Controllers.<current site code name>* namespace. If it is not found there, the *CMS.Controllers.Global* namespace is searched.
Default action	Only available if the **Template type** is *MVC*. Specifies the exact action defined within the controller class that should be performed when pages based on this template are loaded.

Design tab

This tab is available only for *Portal page* or *Dashboard page* type templates. It can be used to define the content of the page template in the same way as on the **Design** tab in **CMS Desk**, but without requiring the context of a specific site. Web parts and widgets can be added (), removed (), configured () or relocated using drag-and-drop. Individual zones may also be managed via their context menu that can be expanded by right clicking the header of the given zone or using the **Web part zone menu** () action. Any changes made here will affect all pages using the edited page template.

Sites tab

On this tab, you can choose which websites the page template will be available for.

Scopes tab

Please refer to the Page template scopes topic for information about this tab.

Layout tab

Here you can edit the layout of the page template. You can either choose to use one of the pre-defined (shared) layouts or you can create a unique layout specifically for the given page template (custom layout). Learn more in the Page layouts topic.

Theme tab

This tab allows you to manage the files contained in the page template's theme folder. This usually includes any files required by the CSS styles added for the template's custom page layout on the **Layout** tab (e.g. images). For more information, please see the Development -> CSS stylesheets and design -> App themes topic.

Web parts

Here you can see the XML document with the configuration of the web parts. Use this dialog only in a situation when the standard **Design** mode isn't working as expected (e.g. due to an error or data inconsistency).

Header

Here you can define custom HTML code that will be placed into the **<head>** element (i.e. inserted between the *<head></head>* tags) of all pages that use this page template. This can be useful if you

need to link some additional CSS or JavaScript files.

The head content can also be inherited by child documents under pages that use this template. The inheritance depends on the following options, which can be specified on this tab for each template:

- **Allow child templates to inherit the current header** - if enabled, the head content of this template will be inheritable by child documents.
- **Inherit header(s) from parent template(s)** - indicates if documents based on this template should be allowed to inherit head content from the templates of documents that are above them in the content tree.

Please note that the header inheritance also follows the rules of the standard content inheritance, which can be configured for individual documents and templates. For example, all documents displayed within a master page would also inherit the head content of its template, but a child document without any visual inheritance would not. This way, the head code will only be added to pages where it is required to correctly display the master template's content.

Documents tab

On this tab, you can see a list of documents (pages) that are using the page template. By clicking a document in the list, you will be redirected to the document's **CMS Desk -> Edit** mode. The documents in the list can be filtered according to the **Site** they are on, their **Document name** and their **Document type**.

7.2.4.6 Page template scopes

Page template scopes define where a page template can be used. **Important - template scopes are not a security feature!** They only provide a way how to simplify the user experience by not offering too many page templates to the users!

This means that you can limit which page templates will be offered to the users in the page template catalog in a certain location. However, when a user creates a page there, they can move it to another location, even to one which is not allowed by the template scope.

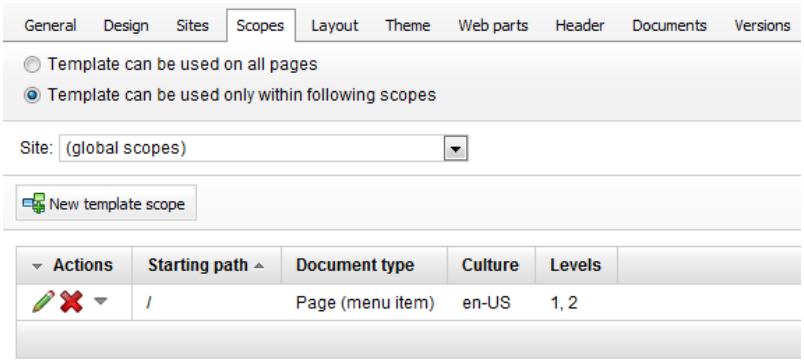
Template scopes can be defined on the **Scopes** tab when selecting a page template from the tree in **Site Manager -> Development -> Page templates**. There are two basic options available:

- **Template can be used on all pages** - the use of the page template is not limited (it can be used in all parts of the website).
- **Template can be used only within following scopes** - if selected, the page can be used according to the rules defined by the template's scopes.

If you choose the second of the options mentioned above, UI controls for managing the scopes will be displayed below the line.

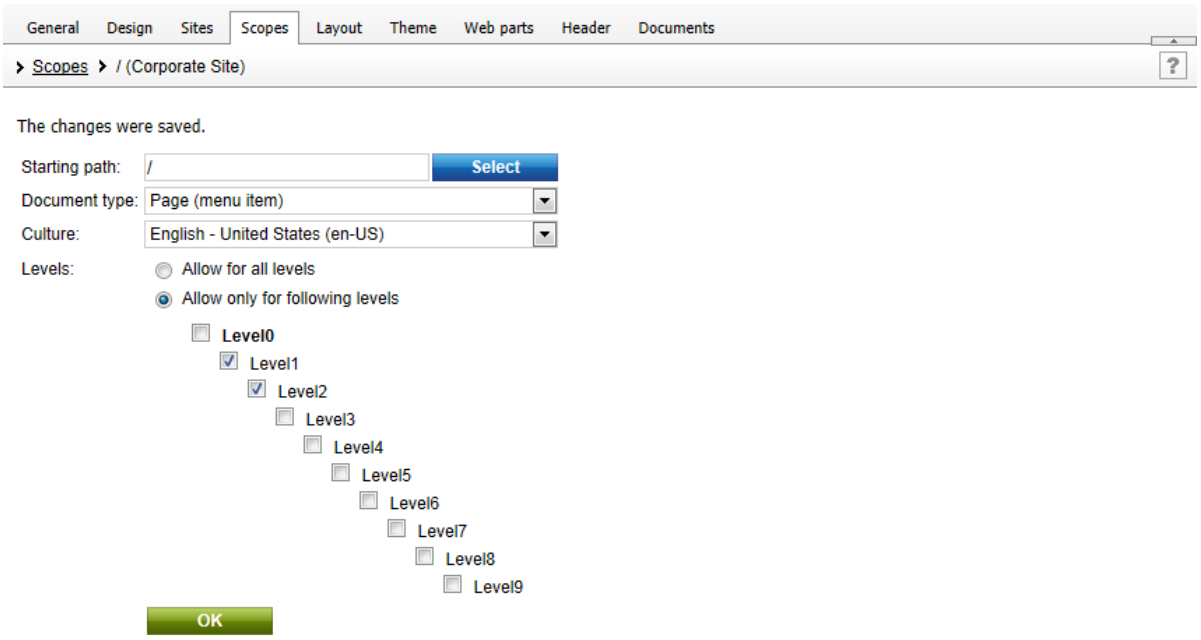
Based on the site selected in the **Site** drop-down list, you can display either site-related scopes or global scopes. Site-related scopes are applied only on the selected site, while the **(global scopes)** option indicates that the scopes are valid on all sites in the system.

Using **New template scope**, you can create new template scopes. Listed scopes can be **Edited** ( ) or **Deleted** ( ).



When creating a new scope or editing an existing one, you can specify the following details, which define where the page template can be used:

- **Starting path** - path in the content tree where the page template can be used. The page template will not be offered outside of this path.
- **Document type** - the page template can be used only for documents of the specified type.
- **Culture** - the page template can be used only in the specified culture.
- **Levels** - the page can either be used on all levels or only on the selected levels in the content tree.



7.2.4.7 Page layouts

The structure of every page template is determined by a *page layout*. A page layout consists of **layout code** and **web part zones** that specify regions where designers can place web parts. Page layouts allow you to define the basic **layout and design of your site**.

There are two general types of page layouts:

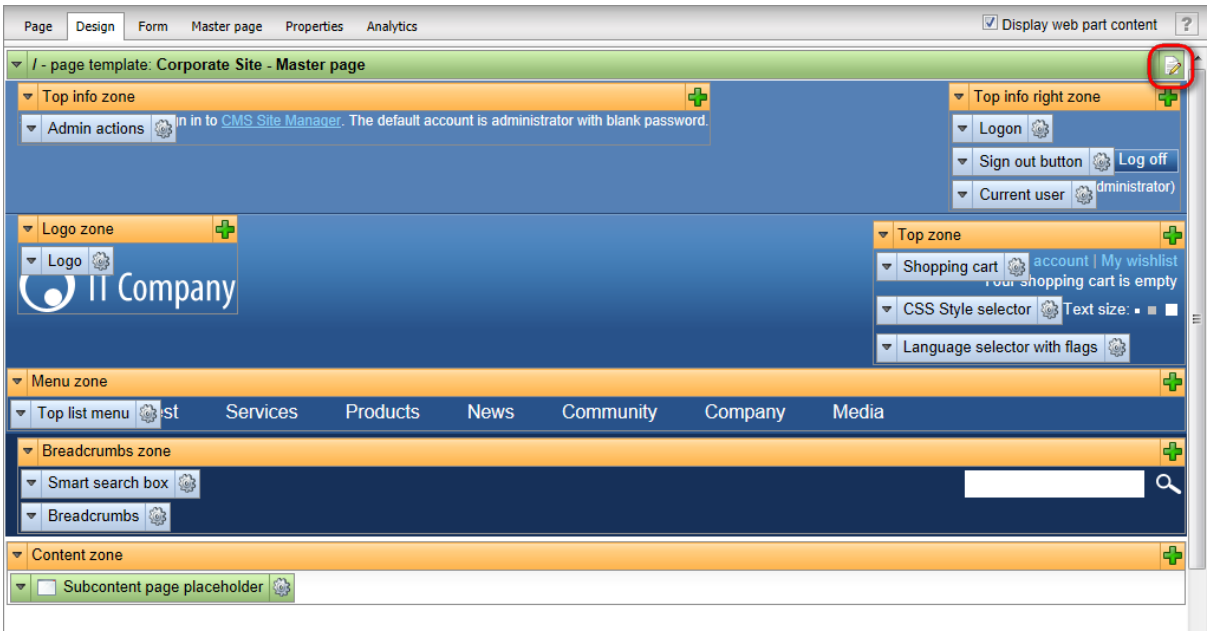
- **Custom** - used only by one specific page template.
- **Shared** - stored as separate objects that can be assigned to any number of page templates. Modifying a shared layout affects all templates that use it.

To edit the layout of a page:

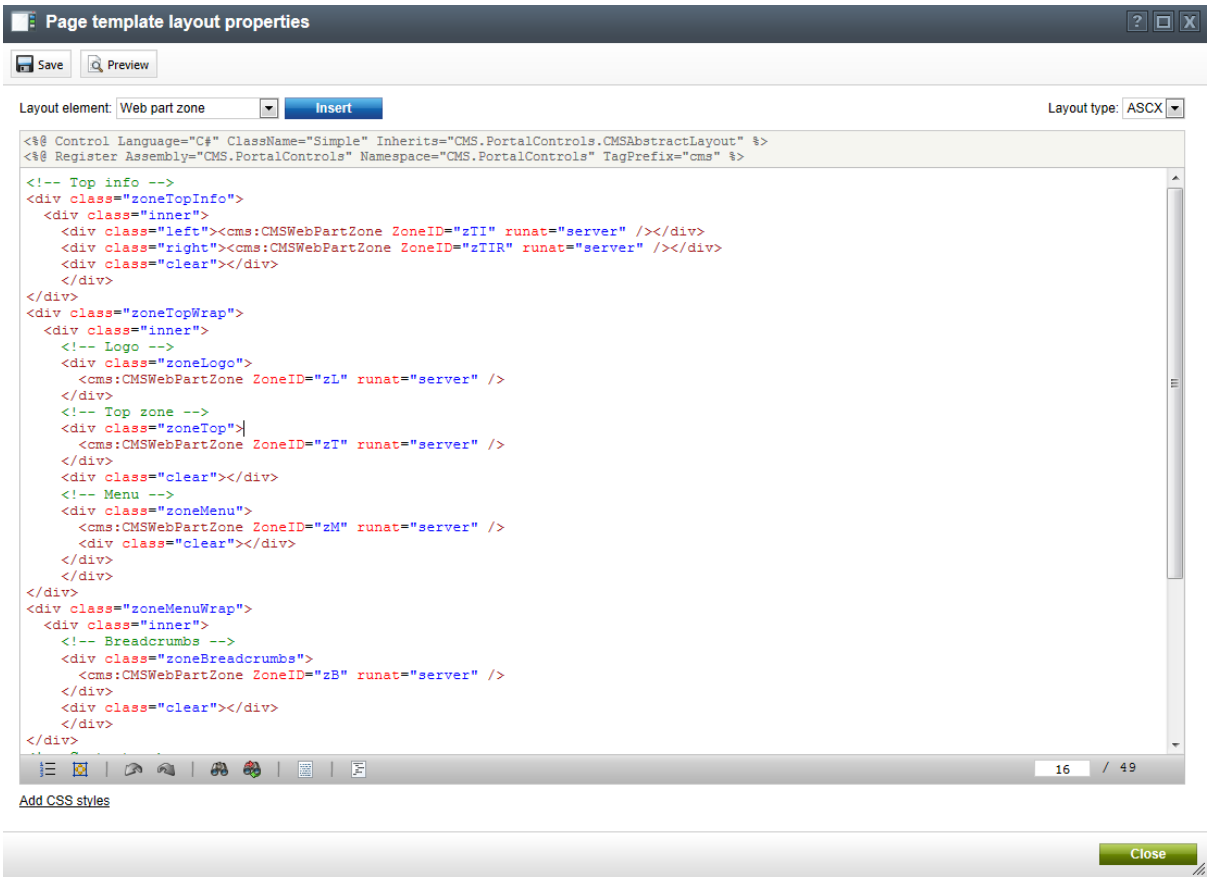
1. Open CMS Desk and select the document representing the page in the content tree.

2. Switch to the **Design** tab.

3. Click the **Edit layout** ( ) button at the top right of the green template area:



4. Modify the layout code as required.



The **Layout type** selector allows you to choose between two types of layout code:

Layout type	Description
ASCX	This type of layout code supports both HTML and ASCX markup, i.e. the same syntax that you would use to edit a standard web form or user control, including inline code and embedded controls. **Important**: For security reasons, ASCX layouts may only be edited by users who have the *Edit ASCX* code permission for the *Design* module. Only global administrators can assign this permission. You can **Insert** web part zones as control tags: `<cms:CMSWebPartZone ZoneID="zoneA" runat="server" />` The *ZoneID* value must be unique for every web part zone within the given layout.
HTML	The system processes the layout code as basic HTML. This means that ASCX markup, such as controls or inline code, is not functional when the layout is rendered.

	You can **Insert** web part zones into HTML layouts as macro expressions: `{^ WebPartZone|(id)zoneA ^}` The value of the *id* parameter must be unique for every web part zone within the given layout. If you need to insert dynamic values into HTML layouts, you can enter Kentico CMS Macro expressions or methods. HTML layouts do not need to be compiled, so you can use and modify them even if the Virtual path provider is not available, such as in a pre-compiled or medium trust environment.

Previewing the layout

By clicking the **Preview** button in the header of the editing dialog, you can write the layout code side-by-side with a preview of how the changes affect the live site version of the page.

See the Design preview topic for additional details about the preview mode.

Sample layout code

Page layouts are composed of standard HTML elements, which means **you have full control over how the system renders the page**. You can choose between a table and CSS-based layouts.

The following sample page layout code uses a table to define a two-column layout:

```
<table>
  <tr>
    <td>
      <cms:CMSWebPartZone ID="zoneA" runat="server" />
    </td>
    <td>
      <cms:CMSWebPartZone ID="zoneB" runat="server" />
    </td>
  </tr>
</table>
```

The following layout code defines the same two-column layout, but using DIV elements and CSS styles:

```
<div style="width: 50%;">
  <div style="width: 80%; float: left;">
    <cms:CMSWebPartZone ID="zoneA" runat="server" />
```

```
  </div>
  <div style="width: 50%; float: right;">
    <cms:CMSWebPartZone ID="zoneB" runat="server" />
  </div>
</div>
```

Adding CSS styles to layouts

Page layouts allow you to directly define any CSS classes used within the layout code.

Requirement: Enable the **Allow CSS from components** setting in **Site Manager -> Settings -> System -> Performance**.

1. Click Add CSS styles below the page layout's code. The **CSS styles** editor appears.

2. Enter the definitions of the required CSS classes.

3. Click **Save**.

All pages that use the layout automatically load the specified styles. See also: CSS for page components

Conditional layouts

When editing the code of ASCX page layouts, you can **Insert** *Conditional layout* elements. This allows you to create flexible layouts that display content based on certain criteria. The page layout renders the content between the *CMSConditionalLayout* tags only if the conditions specified by the properties are fulfilled.

For example:

```
<div class="padding">
  <cms:CMSConditionalLayout runat="server" id="goldLayout" GroupName="Roles"
VisibleForRoles="GoldPartners">
    <cms:CMSWebPartZone runat="server" ZoneID="zGold" />
  </cms:CMSConditionalLayout>

  <cms:CMSConditionalLayout runat="server" id="silverLayout" GroupName="Roles"
VisibleForRoles="SilverPartners">
    <cms:CMSWebPartZone runat="server" ZoneID="zSilver" />
  </cms:CMSConditionalLayout>

  <cms:CMSConditionalLayout runat="server" id="defaultLayout" GroupName="Roles" >
    <cms:CMSWebPartZone runat="server" ZoneID="zDefault" />
  </cms:CMSConditionalLayout>
</div>
```

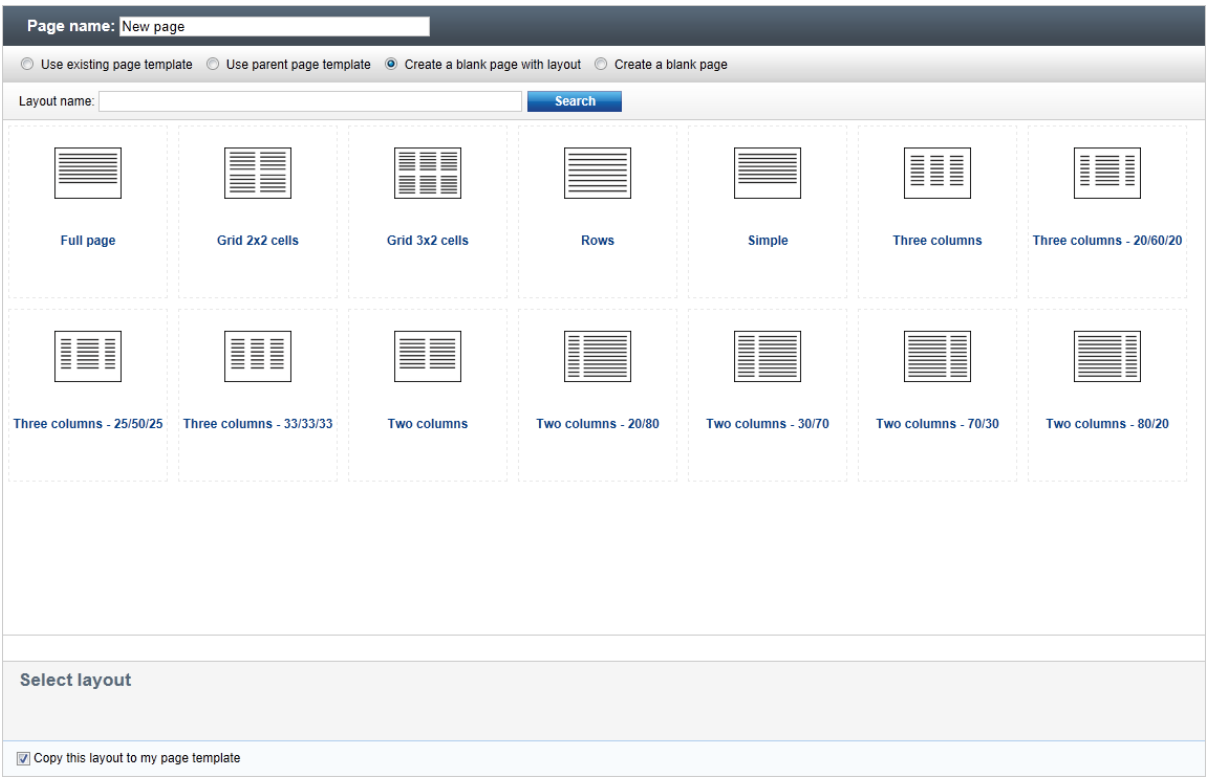
This sample layout displays one of three possible web part zones based on the roles of the user viewing the page. Gold partners see the content of the *zGold* zone, Silver partners see the *zSilver* zone and all other users see *zDefault*.

The following properties are available for conditional layouts:

Property	Description
GroupName	Allows you to group conditional layout elements together. When multiple conditional layouts use the same group name, the page only displays the first one (from the top of the code) that has its visibility condition fulfilled.
VisibleForDocumentTypes	Adds a visibility condition that checks if the current page is of a specific document type. Enter the value as a list of document type code names separated by semicolons. For example: *VisibleForDocumentTypes="CMS.MenuItem;CMS.News"*
VisibleForRoles	Adds a visibility condition that checks if the user viewing the page belongs to specific roles. Enter the value as a list of role code names separated by semicolons. For example: *VisibleForRoles="MarketingManager; ChatSupportEngineers"*
VisibleForDeviceProfiles	Adds a visibility condition that checks if the user viewing the page matches a specific device profile. Enter the value as a list of device profile names separated by semicolons. For example: *VisibleForDeviceProfiles="iPad;iPhone"*
VisibleForDomains	Adds a visibility condition that checks if the site is being accessed under a specific domain name. Enter the value as a list of domain names separated by semicolons.
ActiveInDesignMode	If set to *true*, the conditional layout also evaluates its visibility condition in *Design* mode. *False* by default.

Creating pages with shared layouts

When creating a new page, you can select the **Create a blank page with layout** option and choose from a number of pre-defined page layouts:



The **Copy this layout to my page template** checkbox at the bottom of the selection dialog determines whether the system creates an ad-hoc copy of the layout specifically for the page template, or if the shared layout should be assigned directly. If you disable the option and then modify the layout code, the changes affect all pages with templates that use the shared page layout. Leave the option enabled unless your intention is to create pages with the same layout, that can be edited in one place.

Managing shared page layouts

You can manage the pre-defined (shared) page layouts in **Site Manager -> Development -> Page layouts**. When editing a layout on the **General** tab, you can modify its code and also configure the following properties:

Display name	Name of the layout displayed in the page layout list.
Code name	A unique name that serves as an identifier for the page layout (e.g. in the API).
Description	Allows you to enter an optional text description of the page layout.
Thumbnail	Upload field for the layout preview image. Users see this image in the page layout selection dialog when creating new blank pages.
Is convertible	If enabled, you can use automatic mapping to assign replacement layouts that the system loads instead of the current layout for specific device profiles.
Number of zones	Indicates how many web part zones the layout uses. The number of zones helps users find appropriate matches when mapping layouts for device profiles.

	The system automatically counts the number of zones in the layout code, but you can manually override the value (for example in the case of conditional layouts or layouts that load web part zones dynamically).

On the **Page templates** tab, you can check which templates use the given layout. Templates with a custom page layout are not included here, even if they were created as a copy based on the currently edited shared layout.

Using layout web parts

It is also possible to define the layout of a page template by adding web parts of a special type designed for this purpose. This approach allows you to place web part zones onto a page template without the need to write or edit the page layout code.

Simply add a layout web part to a page containing a single zone and then you can configure the required layout via the web part's properties dialog or even directly on the **Design** tab. To learn more, read the Development -> Web parts -> Layout web parts chapter of this guide.

7.2.4.8 The master page concept

Master pages represent the powerful concept of sharing the same header and footer for all pages on the website. It allows you to manage repeated items, such as the site logo, main menu and footer content in a single place.

The root of the content tree is always a master page. You can also configure any other page template to be a master page by enabling the **Master page** option on the **General** tab of the page template editing interface at **Site Manager -> Development -> Page templates**.

The following figure shows how the same master page is used for the home page and product page. As you can see, the pages are inserted inside the master page:

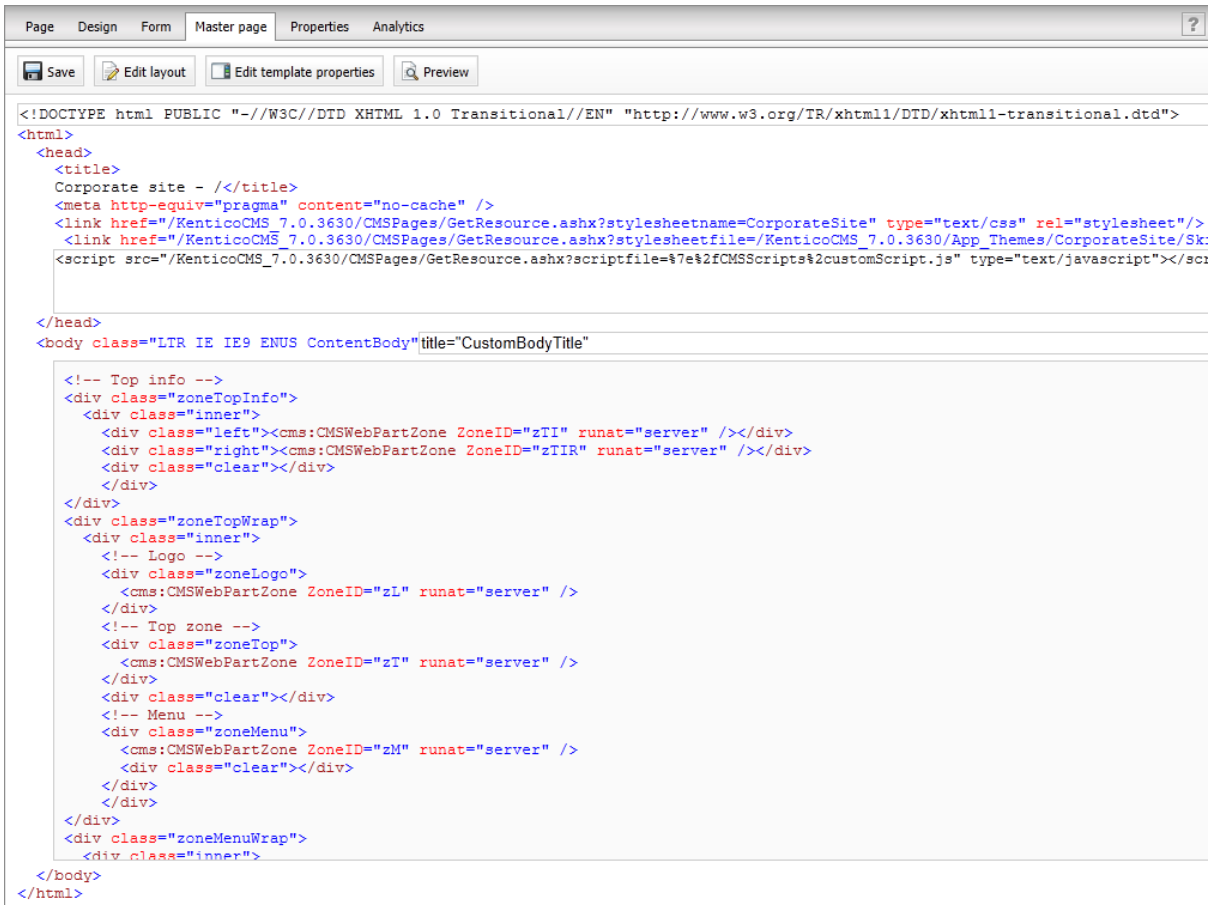
7.2.4.9 Editing the master page

Master pages are either the root of the content tree or pages whose page template has the **Master template** option enabled on the **General** tab of the page template editing interface in **Site Manager -> Development -> Page templates**. They can be edited just like any other page. You can use the **Design** tab to edit the web parts and layout of the master page.

Page placeholder

A master page must always contain the General -> Layout -> Page placeholder web part that specifies where the content of sub-pages should be loaded. Visual inheritance is described in more detail in the next topic.

In addition, there's a special **Master page** tab available in CMS Desk only for master pages.



This tab allows you to define sections of the master page's HTML code. This code is also added to all pages that inherit content from the master page.

- **DOCTYPE** - insert any code that needs to be placed at the beginning of the page's HTML source, typically the DOCTYPE definition.
- **HEAD** - allows you to add HTML code inside the **<head>** element of the master page and all child pages that inherit its content.
- **BODY** - here you can add custom attributes to the **<body>** element.

- **Master page layout** - the area between the *<body>* tags displays the layout code of the master page template. To modify the layout code, click *Edit layout* in the tab header.

The code outside of the editable sections is only informative and may not be identical to the actual code rendered for pages.

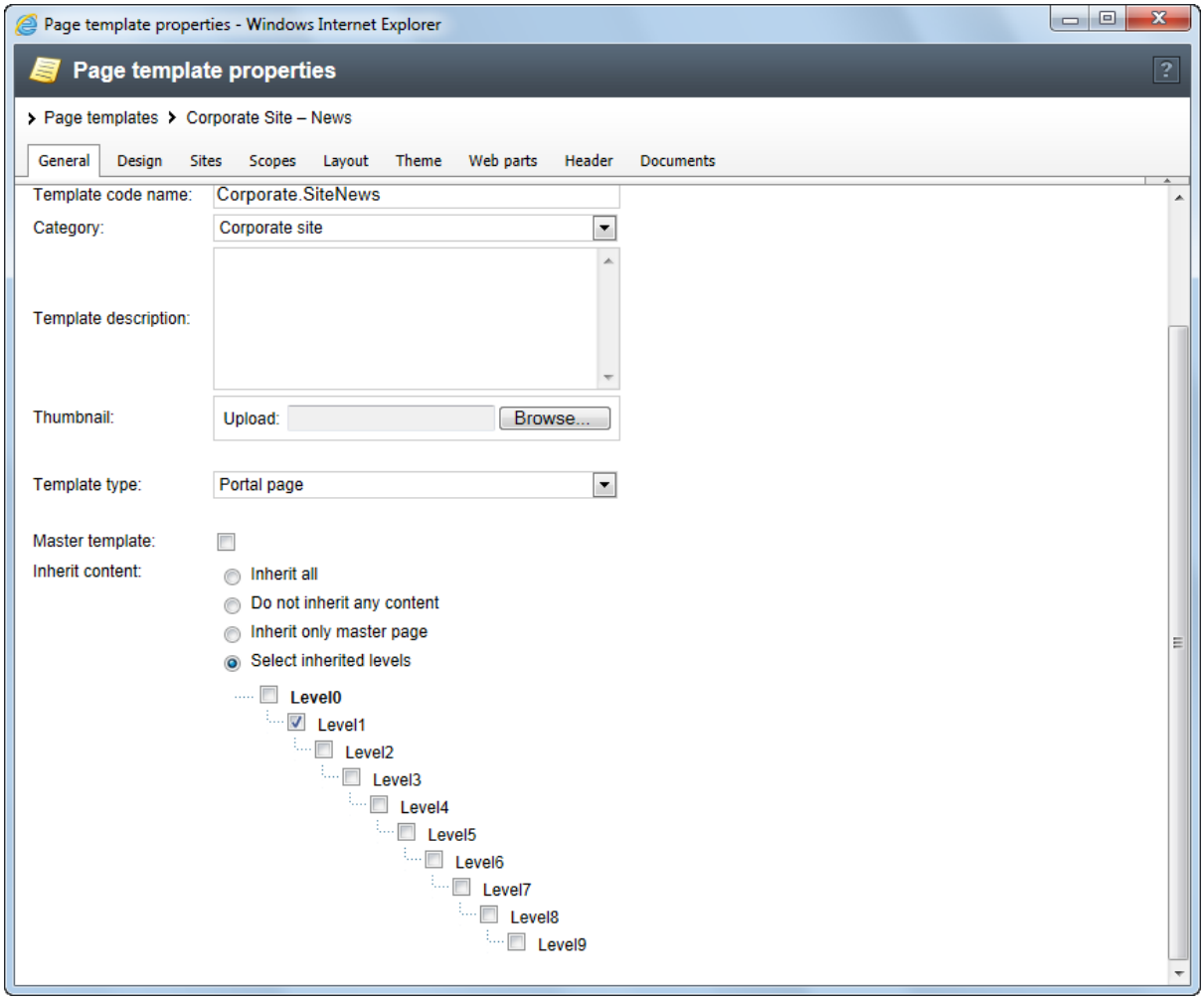
7.2.4.10 Visual inheritance

As you can see in the The master page concept topic, the content of sub-pages is displayed within the master page or generally within any parent page using the Page placeholder web part. The result of this approach is that the sub-page content is "nested" inside the content of the parent pages.

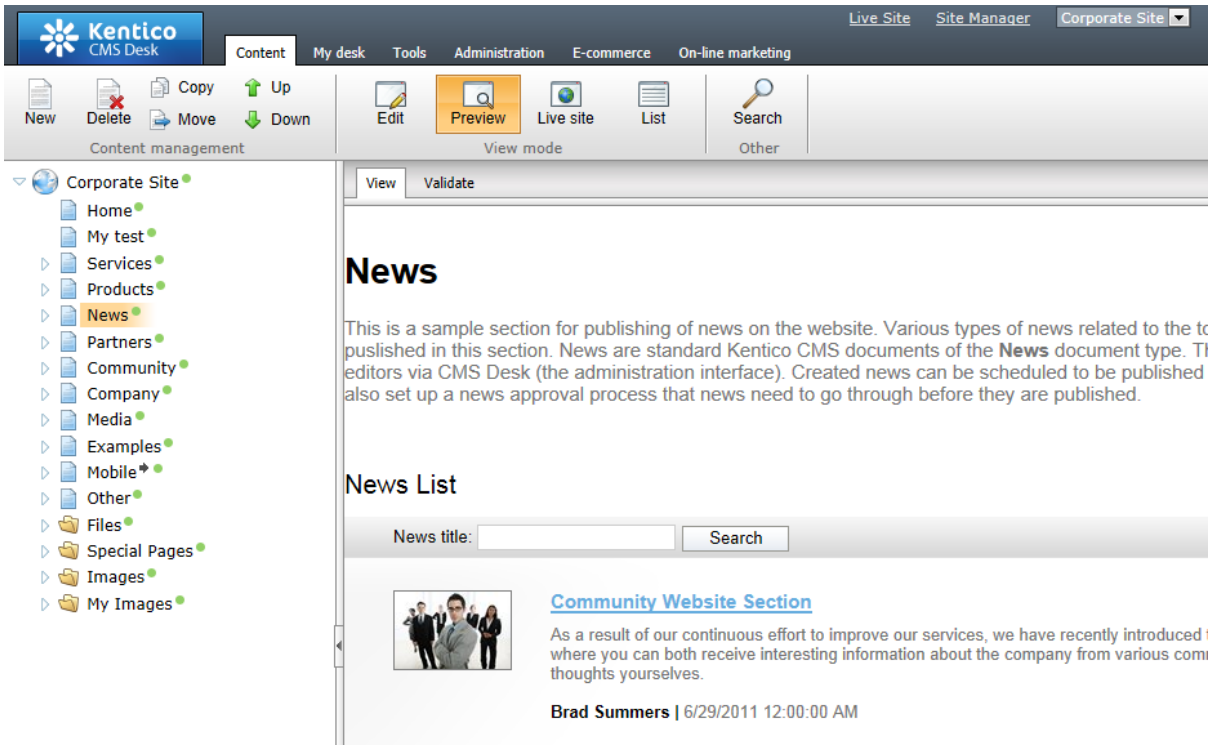
In some cases, you may wish to hide parts of the parent page. There are several ways to achieve that:

Using the "Inherit content" property of the page template

Select the **/News** page, go to its **Properties -> Template** tab and click the **Edit template properties** action. Now you can set the **Inherit content** value to *Select inherited levels* and check only the **Level1** box. This means that only the content from the first level of the content hierarchy will be displayed and the master page (root) is not inherited. Click **Save** to confirm the changes.

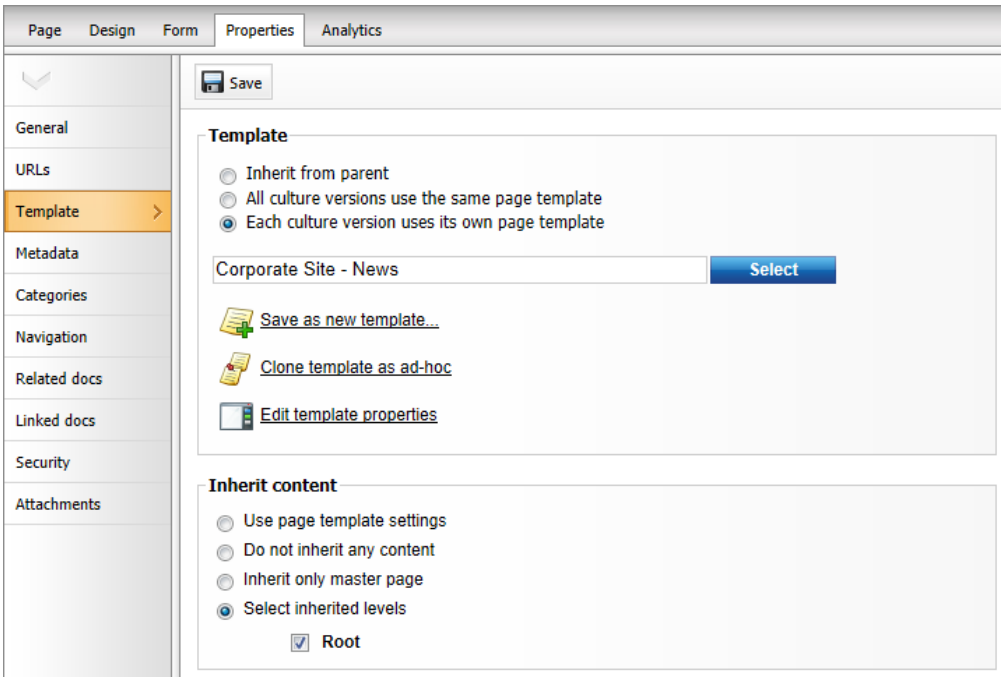


The page will now look like this:



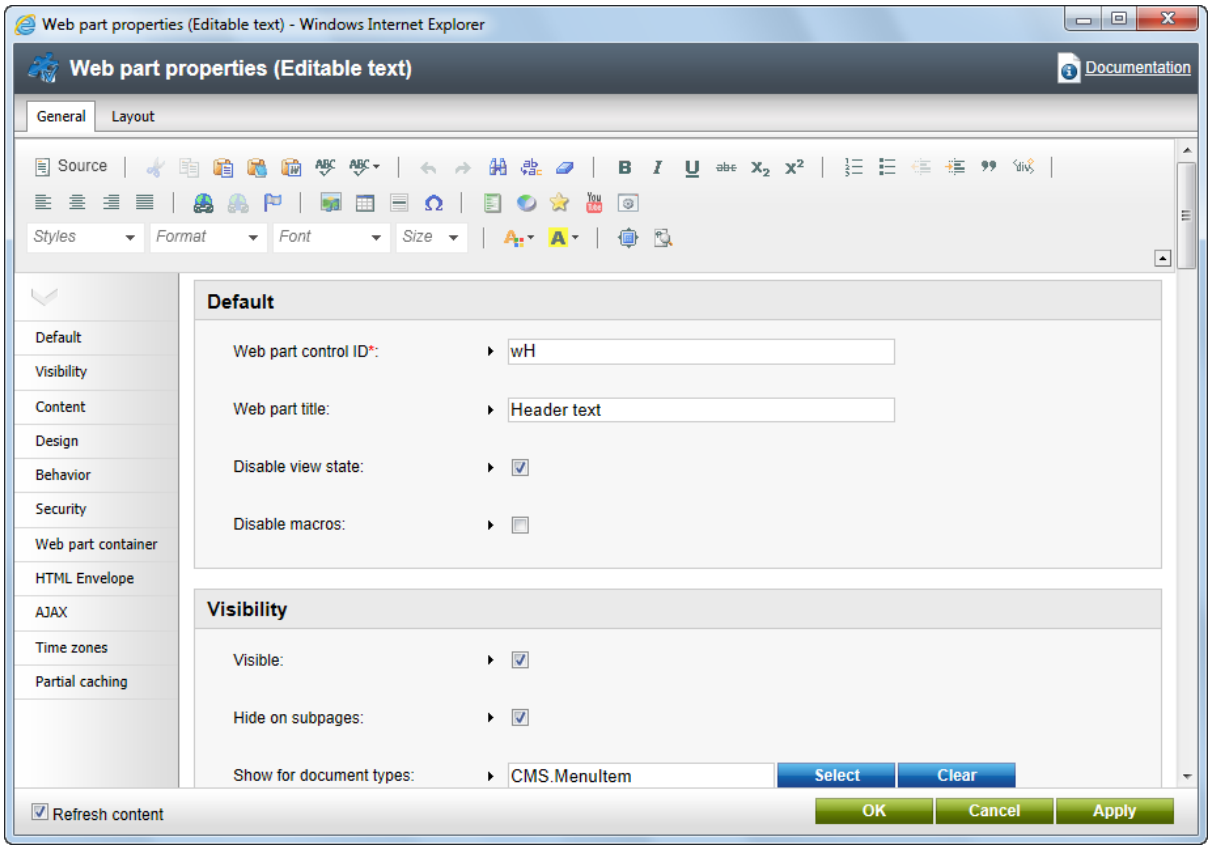
Set the **Inherit content** value back to **Level0** for now.

Similarly, you can set the content inheritance on the level of **individual pages** using the **Properties -> Template** dialog. The content inheritance settings you configure for documents override the page template settings:

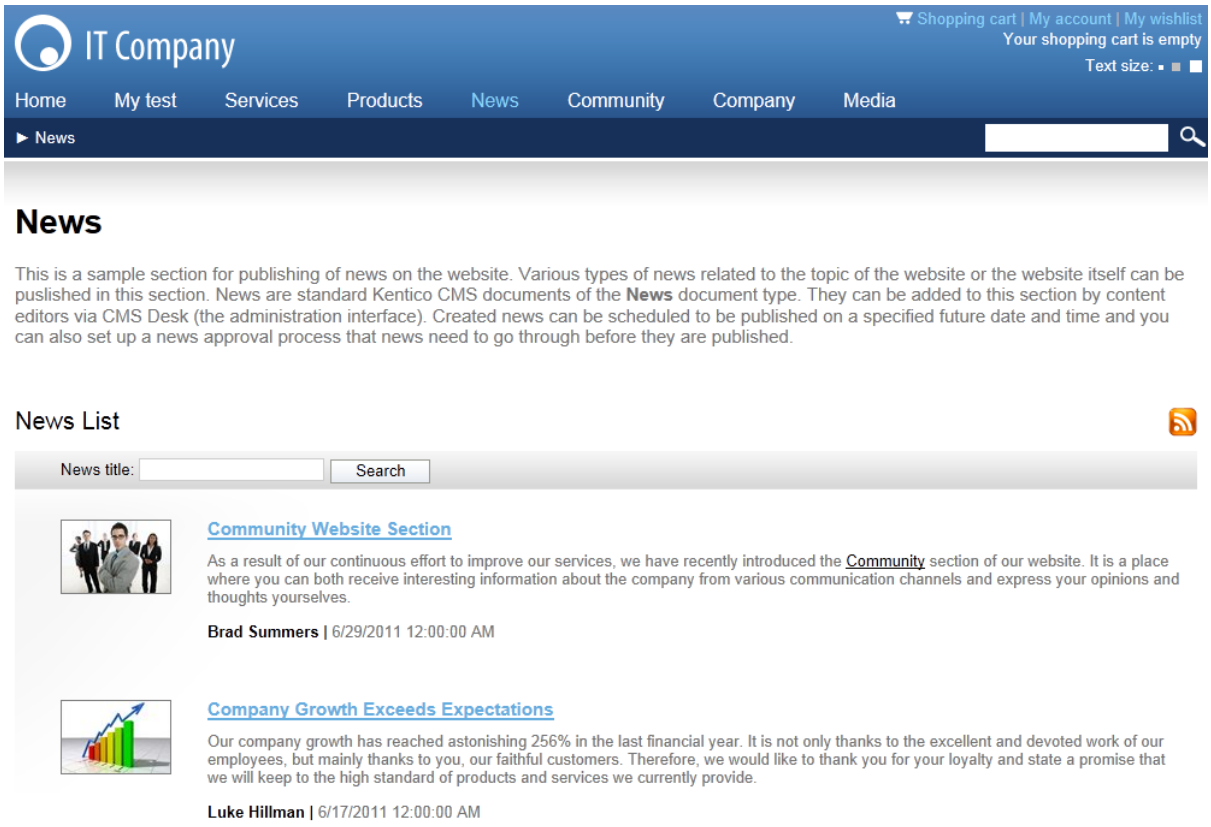


Using the "Hide on subpages" web part property

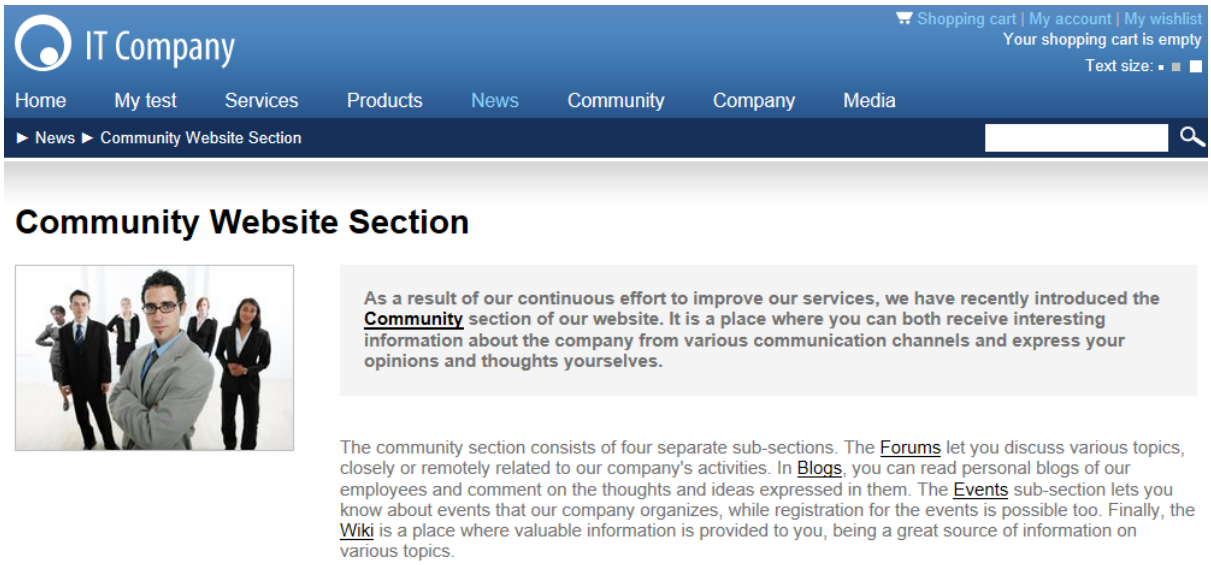
Click **/News**, switch to the **Design** tab and click the **Configure** ( ) button of the **HeaderText** web part. The web part has the property **Hide on subpages** set to true:



Click **Cancel** and click **Live site**. Please note that when you display the list of news, the title **News** is displayed:



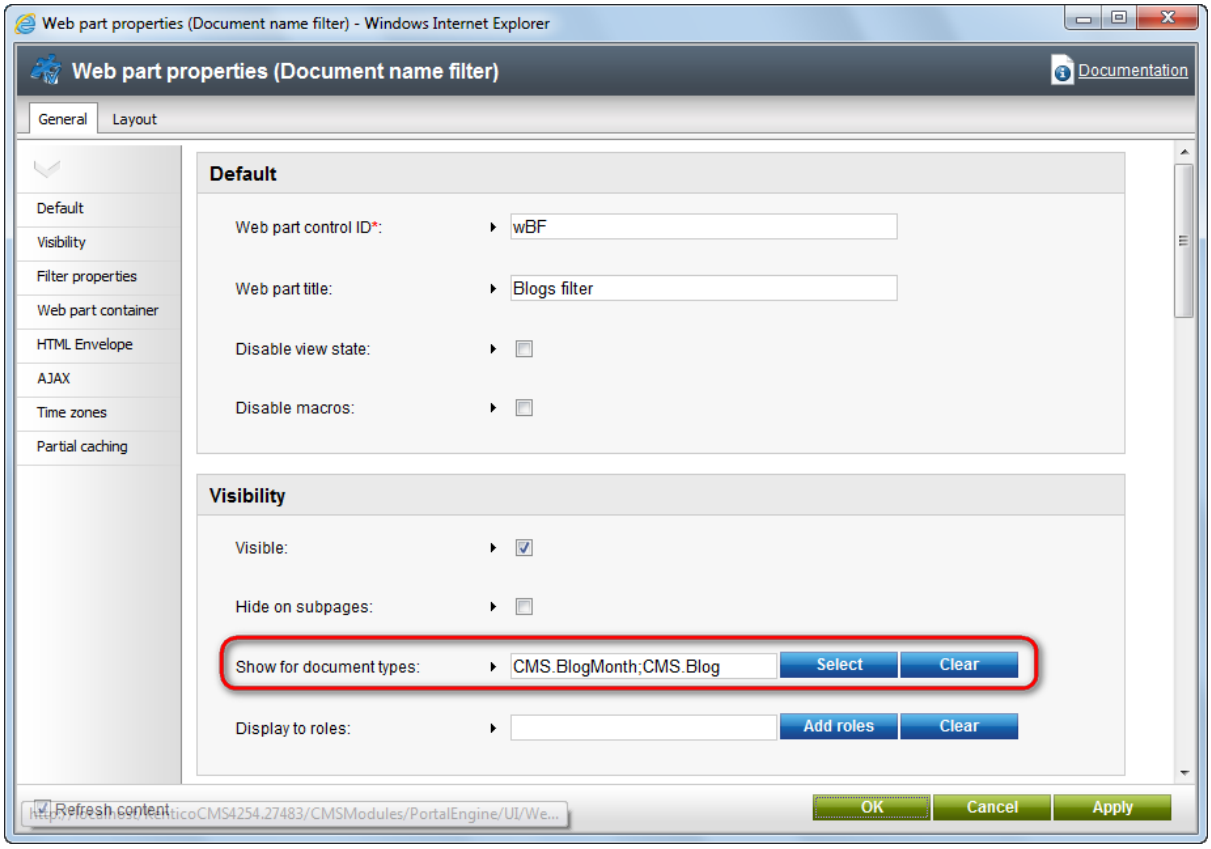
If you go to some particular news item, the title is hidden:



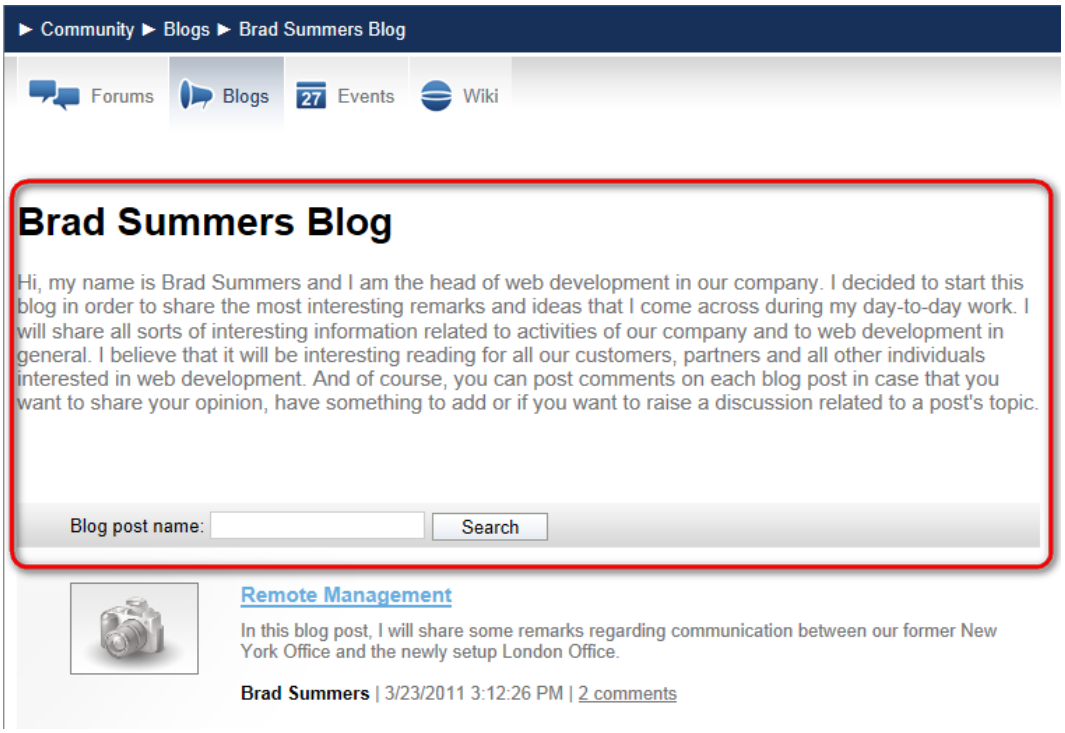
This is ensured by the **Hide on subpages** property that hides the web part displaying the **News** title.

Using the "Show for document types" web part property

The **Show for document types** property allows you to define a list of document types for which the web part will be displayed. To see how it works, select **/Community/Blogs/Brad-Summers-Blog** in the content tree. On the **Design** tab, click **Configure** (⚙) for the **Header text**, **Description text** or **Blogs filter** web part. All of them have the **Show for document types** property set to *CMS.BlogMonth; CMS.Blog*. This means that the web part will only be displayed on blog month and blog documents, not on blog posts which are stored under blog months. Click **Cancel**.



Still viewing the **Brad Summers Blog** document, switch to the **Live site** mode. The header text, description text and filter are all displayed above the repeater displaying blog posts.



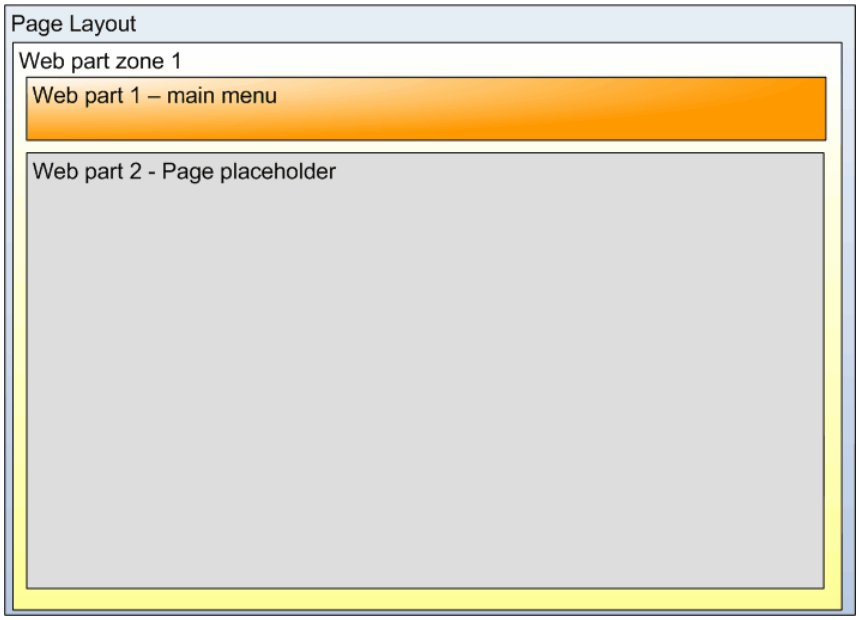
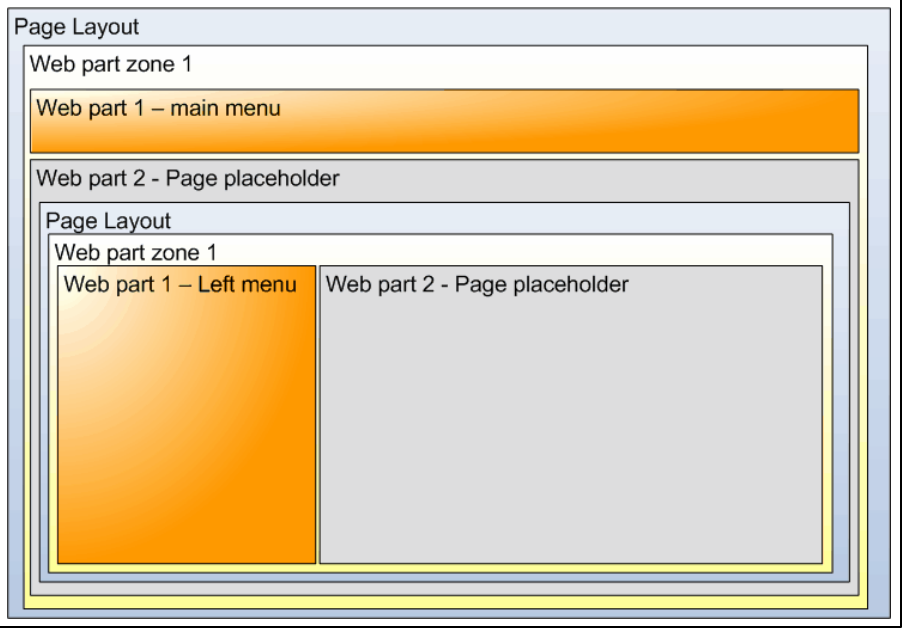
When you display some particular blog post, the web parts are not displayed because *CMS.BlogPost* is not among the enumerated document types.

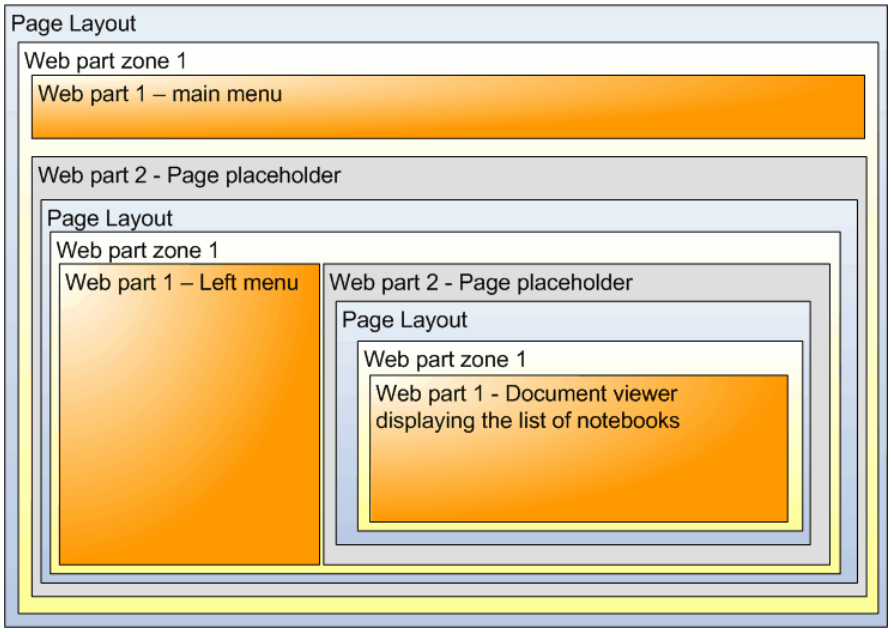
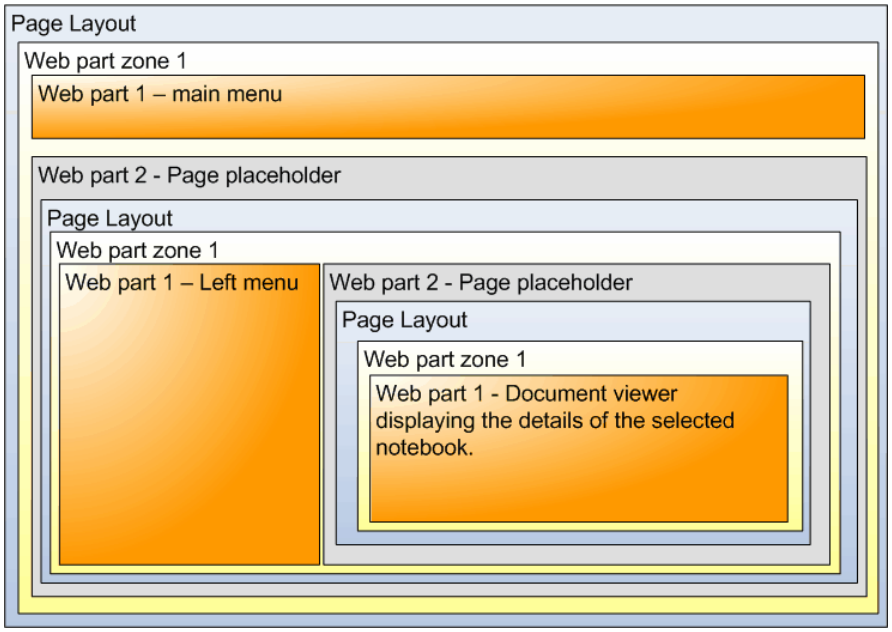
You have learned how to break inheritance of content, hide content on subpages and how to display content based on the current document type.

7.2.4.11 Content tree and page templates

The content tree defines not only the site map of your website, but it also defines how the pages are nested.

For example, when you request the */Products/Laptops/Acer-Aspire-3105WLMi* page, the portal engine loads the content in the following order:

Processed path	Page template	Rendered page
/ (root)	Main menu page template	
/Products	Products page template	

/Products/Laptops	Laptops page template	
/Products/Laptops/Acer-Aspire-3105WLMi	*Inherits page template from parent page. The product detail is displayed automatically instead of the listing by the document viewer.*	

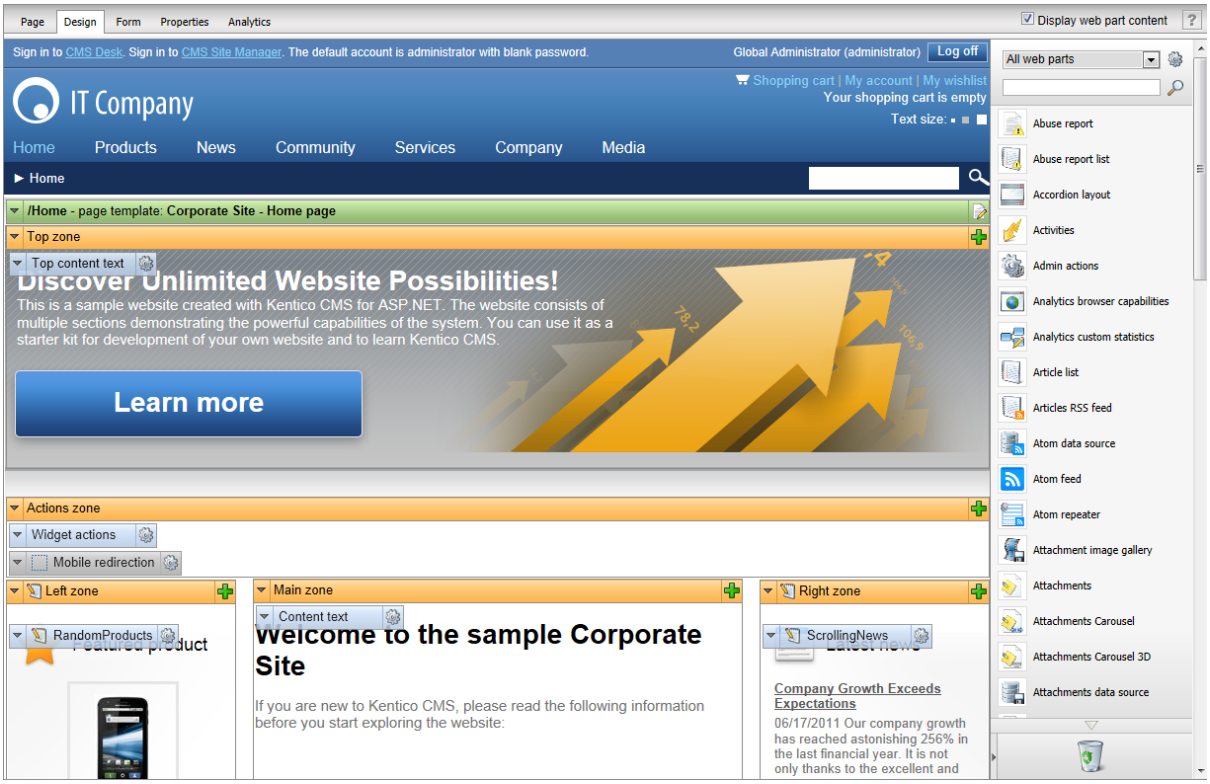
7.2.4.12 Using and configuring web parts

Web parts can be used on both portal page templates and ASPX templates. However, with ASPX page templates, you lose the friendly browser-based interface — the web parts need to be added and configured in Visual Studio just like standard user controls.

This topic describes how you can work with instances of web parts when editing page templates through the portal engine. Please see the Development -> Web parts chapter for detailed information about the management of web part objects and instructions on how to perform various types of development tasks.

Start by opening **CMS Desk** on the **Content** tab in **Edit** mode. Select any page document from the

content tree (for example *Home*) and switch to the **Design** tab. This tab allows you to view the structure of the page's template in design mode and modify its web part content as needed.



The most direct way to add a new web part to the template is provided by the *web part toolbar*, which is displayed on the right side of the tab by default. Because there is a large total number of web parts available, you can choose which of them should be listed in the toolbar by selecting an appropriate category from the drop-down list at the top. It is also possible to look up specific web parts by entering their name or its part into the search textbox (). Once you find the web part that you wish to add, simply drag it from the toolbar and drop it into the desired location in one of the template's web part zones.

You can remove existing web part instances from the page template by dragging them into the "trash bin" area () of the toolbar.

Web part toolbar settings

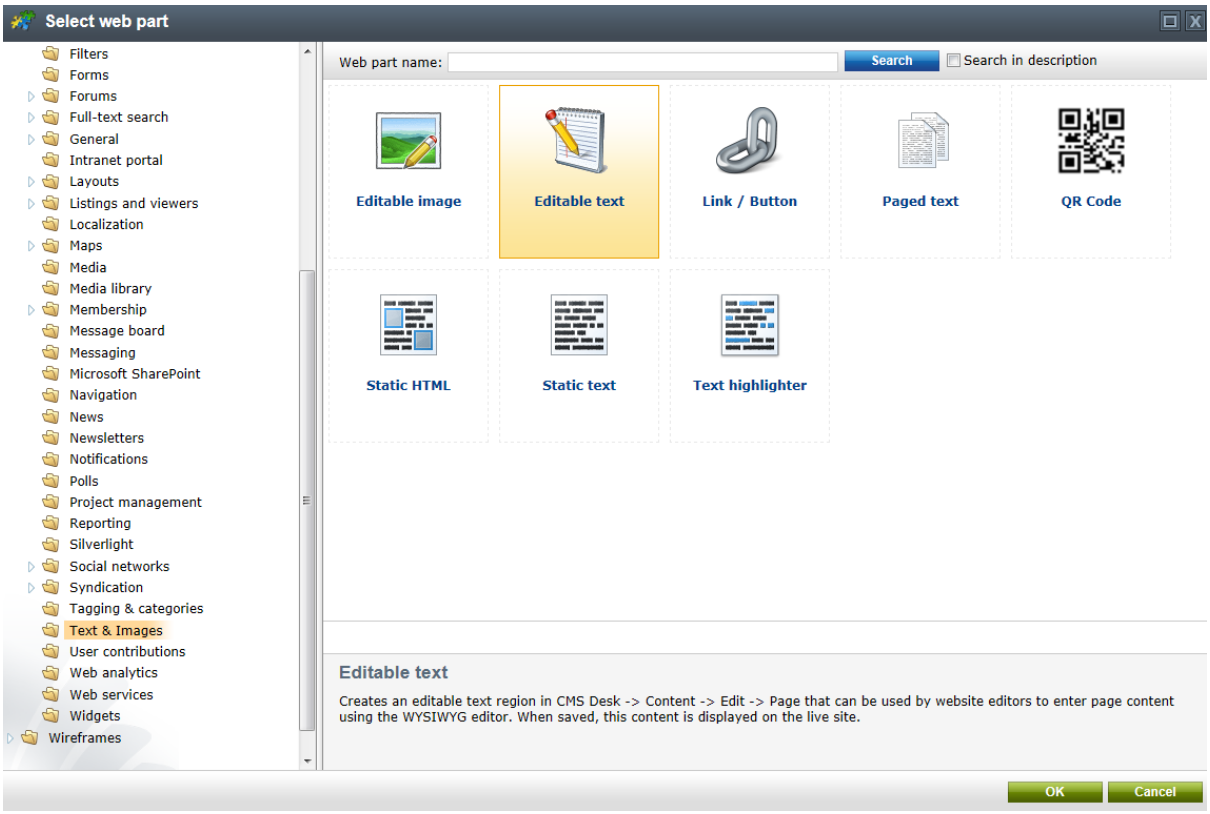
Each user can configure the toolbar according to their preferences. There are four available positions (all sides of the Design tab) and it can also be completely disabled.

These settings can be accessed directly on the toolbar by clicking the **Settings** () button next to the category selector. The same options can also be found by going to **CMS Desk -> My Desk -> My profile -> Details**, so it is possible to change them even for users who have disabled their toolbar.

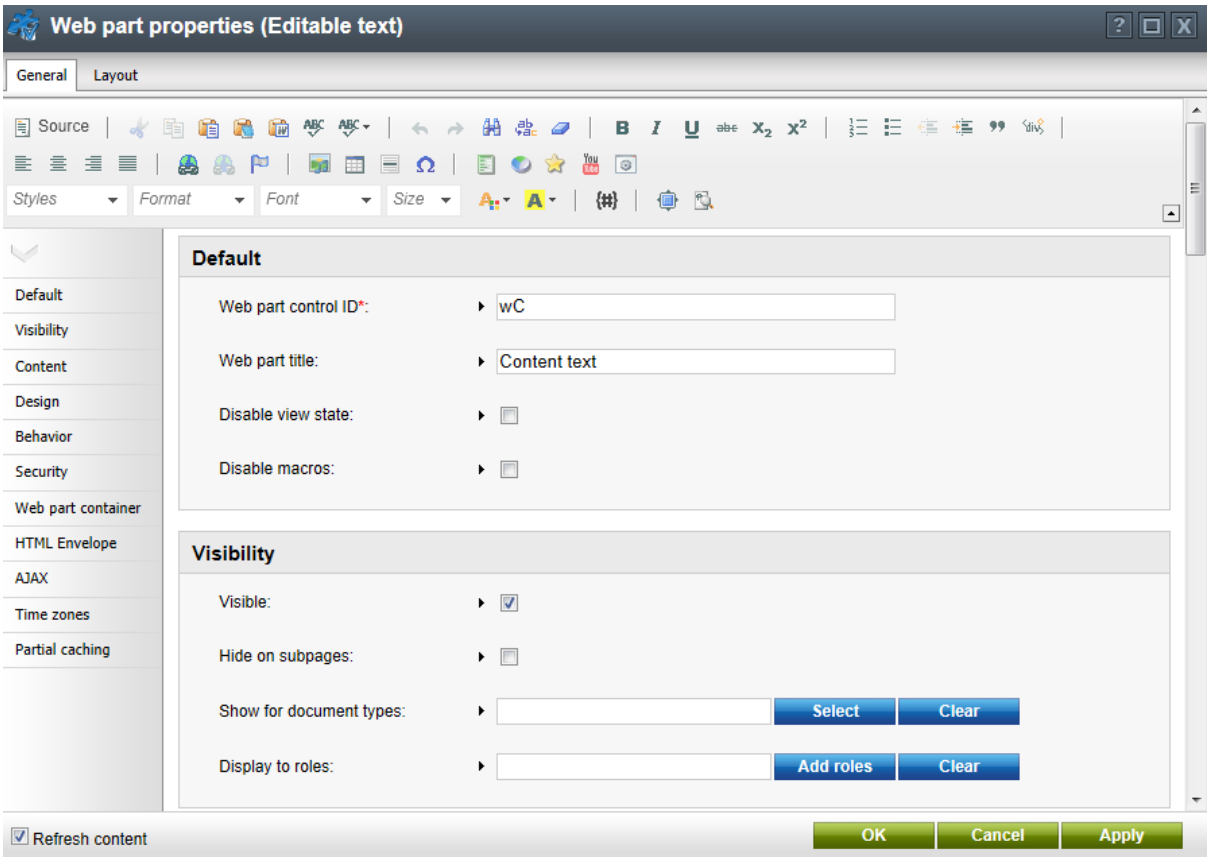
If you only need to increase the size of the Design tab workspace temporarily, you can

collapse the toolbar by clicking the arrow on the border next to the trash bin area.

If you do not wish to use the toolbar, you can instead click on the **Add web part** (✚) icon in the top right corner of the zone where you want to insert the web part. This opens the **Select web part** dialog, which contains a catalog of all available web parts. You can locate a specific web part by browsing through the category tree and then confirm your selection by clicking **OK**.

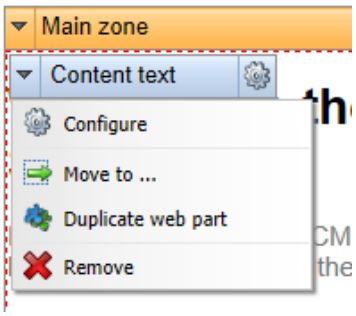


When a new web part instance is added into a zone, using either the toolbar or the zone action buttons, the **Web part properties** dialog will be opened (unless the given web part is configured to skip it). Here you can set up and fine tune the behavior of the web part by entering appropriate values for its properties.



The same configuration dialog can be opened for existing web part instances at any time by clicking the **Configure** ( ) button in their header on the Design tab. You can also use the drag-and-drop functionality to relocate web parts to different positions or other zones.

Additionally, every web part has a context menu that can be opened by right clicking on its header or using the arrow icon ( ). The actions in this menu provide an alternative way to perform common tasks.



A similar menu is also available for entire web part zones. This allows you to configure the zone's properties or carry out mass actions for all contained web parts.

All web part modifications are applied immediately and reflected on the live site. Page templates are not connected to workflow, but it is possible to use object versioning to keep track of the changes made to a template, including its web part content (and roll back to previous versions if necessary). Please read the Development -> Object versioning chapter to learn more.

Impacts of modifying page templates

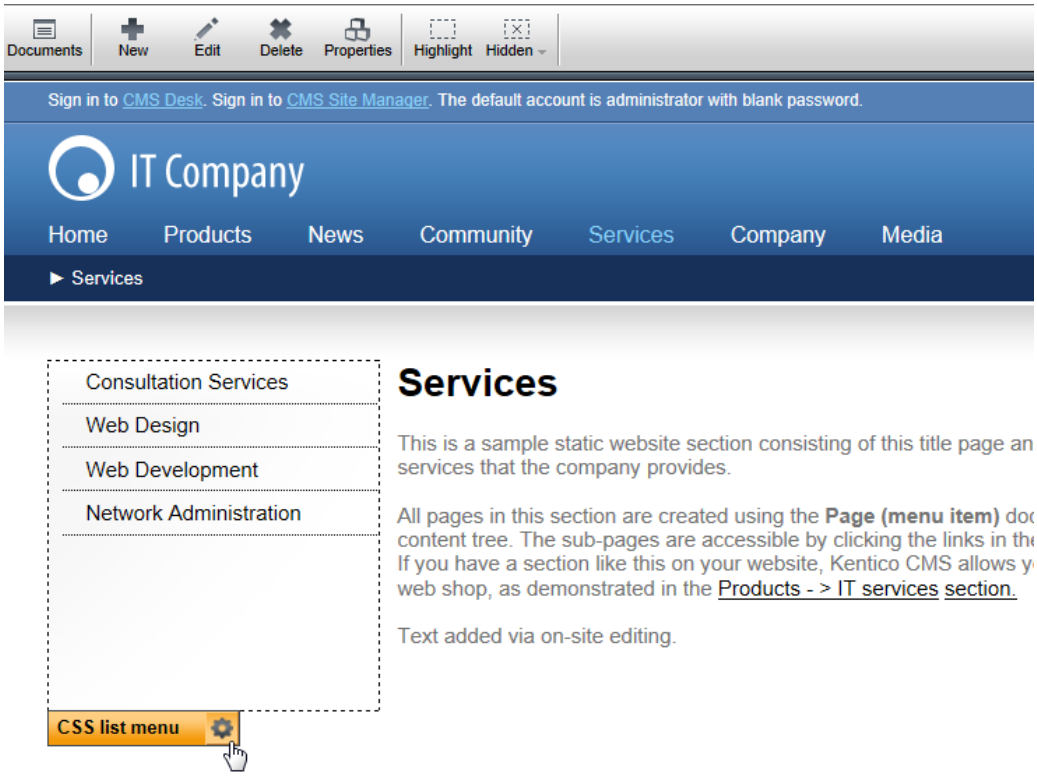
When you edit a re-usable page template that is shared by several pages, the changes will affect all of the pages.

If you wish to modify the design of a single page only, you need to clone its template as an ad-hoc template or as a new page template. See the Cloning and modifying a page template topic to learn more.

Configuring web parts via on-site editing

Even though On-site editing is primarily intended for modifying basic page content and managing documents, users with design permissions may also use it to configure the properties of web parts directly while browsing the website.

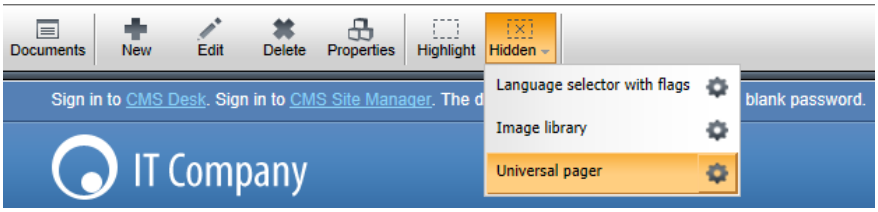
When in on-site editing mode, you can highlight individual web parts simply by moving the mouse over the corresponding part of the page. Highlighted web parts are enclosed in a dotted outline, with an additional box displaying their title. The configure (⚙) action next to the web part title may be used to open the same web part properties dialog as the one on the **Design** tab of CMS Desk.



If you wish to view the entire web part structure of a page, you can enable the **Highlight** action on the on-site editing toolbar. This highlights all web parts placed on the given page.

Some web parts may not have any visible output in the current state of the viewed page. For example, a

paging web part would only be shown if the connected listing contains a sufficiently large number of items. You will not be able to find such web parts directly on the page. However, their properties can still be accessed by clicking the **Hidden** button on the toolbar and configuring (⚙) the appropriate item in the displayed drop-down.



Please note that the **Hidden** list only includes web parts that have the potential to affect the appearance of the page. Instances of invisible web parts that only perform background functionality must be configured in CMS Desk.

Please note

Because of master pages and visual inheritance, the content displayed on the website may often be loaded from several different page templates that are nested into each other.

On-site editing mode does not differentiate between templates, so it is possible to configure any web parts displayed on the current page, even those that actually belong to parent documents.

7.2.4.13 Common web part properties

Many web parts use the same or similar properties. The following table summarizes the most important properties found in web parts:

Default properties

Property Name	Description	Sample Value
Web part control ID	Serves as an identifier for the web part. This ID must be unique within the context of each page template. The value of this property may only contain alphanumeric characters and the underscore character (_).	text1
Web part title	Title of the web part displayed on the *Design* tab of CMS Desk and in on-site editing mode. If empty, the value of the **Web part control ID** property is used for this purpose.	

Visibility

Property Name	Description	Sample Value
Visible	Indicates if the web part should be displayed.	
Hide on sub-pages	Indicates if the web part should be hidden on sub-pages. If checked, the web part will not be displayed on documents that inherit the web part from a parent document.	
Show for document types	Contains a list of document types on which the web part should be displayed. If the currently selected document uses the page template containing the web part, but its type is not specified by this property, the web part will be hidden. The document types in the list must be specified by their code names and separated by semicolons (;). If empty, the web part will be displayed on all document types.	cms.news;cms.event
Display to roles	Contains a list of roles to which the web part should be displayed. This may be used to implement documents with specific functionality for different types of users. The roles in the list must be specified by their code names and separated by semicolons (;). If empty, the web part will be displayed to all users.	cmseditors;customrole

Web part container

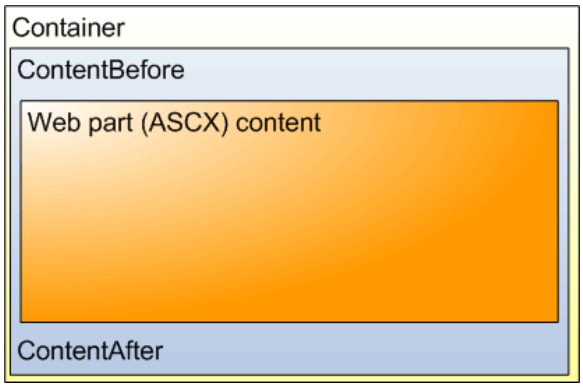
Property Name	Description	Sample Value
Web part container	Specifies the name of the container (box) to be displayed around the web part. Only the containers defined at **Site Manager -> Development -> Web part containers** can be selected. The selected container can be edited directly by using the **Edit** button.	
Container title	Sets a title for the container (if one is specified for the web part). This title is displayed only if the *{%ContainerTitle%}* macro is used in the code of the container.	Latest news
Container CSS class	CSS class used for the web part's container.	

	Applied only if the *{%ContainerCSSClass%}* macro is used as the value of the *Class* attribute in the code of the container.	
Container custom content	Custom content to be used in the web part's container. Applied only if the *{% ContainerCustomContent%}* macro is used in the code of the container.	
Hide container on subpages	If enabled, the container will not be displayed around the web part on child documents that inherit it through visual inheritance. For example, this allows you to add a container only for the master page.	

HTML Envelope

Property Name	Description	Sample Value
Content before	HTML content placed before the web part. Can be used to display a header or add some encapsulating code, such as * * or *<table>* elements to achieve the required layout.	<table style="background-color: red"><tr><td>
Content after	HTML content placed after the web part. Can be used to display a footer or close the tags contained in the **ContentBefore** value, such as * * or *</table>* elements.	</td></tr></table>

The structure of a web part and its envelope looks like this:



You can find more details on web part containers in the Containers overview topic.

Content

Property Name	Description	Sample Value

Path	Path of the document(s) from the content tree that should be loaded. In addition to basic paths, the property also supports special characters that may be used to specify entire sections of the content or relative paths. For more information about path formatting options, please see Appendix A - Path expressions.	/news/%
Highlighted node path	Alias path of the node that should be selected in a menu control. If you do not specify any value, the current path is used.	/products/nokia
Query	Specifies the name of the query that the web part uses to retrieve data from the Kentico CMS database. You can define queries for document types and custom tables. To manage the queries, edit a particular item in **Site Manager -> Development -> Document types / Custom tables** on the **Queries** tab, or use the buttons next to the field in the web part properties dialog.	cms.news.selectlatest

Content filter

Property Name	Description	Sample Value
Document types	Determines which types of documents should be selected. Specified through the document type code names, separated by semicolons (;). The * wildcard can be used as a substitute for a random sequence of characters. For example *Product.** would include the document types *Product.Camera*, *Product.CellPhone*, *Product.Computer* etc. If the property is left empty, web parts retrieve all document types by default. In the case of menu and navigation web parts, page (*cms.MenuItem*) documents are selected by default. **Please note**: If all document types are loaded (empty value), only the common data columns from the *View_CMS_Tree_Joined* view will be available in the retrieved data. The specific fields of individual document	cms.article;cms.news; cms.menuitem

	types will not be included. This should be considered in the code of transformations, WHERE conditions, ORDER BY expressions etc.	
Combine with default culture	Indicates if the default language version of the document should be displayed if the document is not translated to the current language.	You can choose between **Yes** and **No** or you can use website-level settings.
Culture code	Culture of the content to be displayed.	en-us
Maximum nesting level	Maximum nesting level. It specifies the number of sub-levels in the content tree that should be included in the displayed content. The value 1 indicates that only the current document should be returned. The value -1 indicates all child documents.	
ORDER BY expression	ORDER BY part of the SELECT query.	ProductName ASC, ProductPrice DESC
Select only published	Indicates if only published documents should be displayed.	
Site name	Code name of the website from which you want to display the content. If you leave the value empty, the content is retrieved from the current website.	CorporateSite
WHERE condition	WHERE part of the SELECT query.	ProductPrice > 100 AND ProductColor='green'
Filter out duplicate documents	If the displayed data contains multiple links to the same document (see Linked docs for details), you can choose to display only one of them.	

System settings

Property Name	Description	Sample Value
Check permissions	Indicates if the permissions of the current user should be checked for the content of the web part. If enabled, only documents for which the user has the *read* permission will be loaded.	
Cache item name	Sets the name of the cache key used for the content of the web part. If not specified, this name is generated automatically based on the site, document path, **Web part control ID** and current user.	

	A cache key can be shared between multiple web parts displaying the same content on different pages in order to avoid keeping redundant data in the memory.	
Cache minutes	Sets the number of minutes for which the content of the web part should remain cached before its latest version is reloaded from the database. If left empty, the value entered into the **Site Manager -> Settings -> System -> Performance -> Server content caching -> Cache content (minutes)** setting will be used instead. If set to *0*, caching will be disabled for the web part.	10
Cache dependencies	Contains a list of cache keys on which the content cache of the web part depends. When the specified cache items change, the content cache of the web part is deleted. Each line may only contain a single item. If the **Use default cache dependencies** box is checked, the default dependencies will be used, which include all possible object changes that could affect the content of the specific web part.	

Design

Property Name	Description	Sample Value
CSS prefix	Prefix used for CSS class names. This property allows you to set up different CSS styles for particular menu levels. Every level of the menu will use the prefix for CSS class names that you specify.	Main;Sub1;Sub2
Highlight all items in path	Indicates if all items in the currently selected path of the menu control should be displayed as highlighted.	
Submenu indicator	Path to the image that should be displayed next to items that have sub-items.	/images/submenu.gif
Use alternating styles	Indicates if odd and even items should have different styles.	

Paging

Property Name	Description	Sample Value
Enable paging	Indicates if displayed data should be paged.	
Paging mode	Type of paging parameter - it can be passed either through the URL (**Query string**) or through postback (**Postback**).	
Pager position	Position of the pager - top or bottom	
Page size	Number of records per page.	10
Query string key	The name of the URL parameter that will contain the current page number.	mylistpage
Show first and last buttons	Indicates the buttons that link to the first and last page should be displayed.	

Relationships

These settings allow you to configure the web part so that it displays only content that is related to the specified (main) document.

Property Name	Description	Sample Value
Main document	Document for which you want to display related documents.	
Relationship name	Name of the relationship between documents.	Is related to
Main document is on the left side	Indicates if the main document is on the left side of the relationship.	

No data behavior

Property Name	Description	Sample Value
Hide if no record found	Indicates if the web part should be hidden when no record is found.	
No record found text	Text that should be displayed if no data is found.	No data found.

Editing buttons

Property Name	Description	Sample Value
Show New button	Indicates if the button for adding new items	

	should be displayed in the **Edit** mode when viewing the page.	
New button text	New button description text.	Add new news.
Show edit and delete buttons	Indicates if edit and delete buttons should be automatically shown for each displayed item in the **Edit** mode.	

Transformations

Property Name	Description	Sample Value
Alternating transformation	Transformation used for even items in list view mode in format *<class prefix>.<document type>.<transformation name>*.	cms.news.preview_alternating
Transformation	Transformation used for items in list view mode in format *<class prefix>.<document type>.<transformation name>*.	cms.news.preview
Selected item transformation	Transformation used for items in detail view mode in format *<class prefix>.<document type>.<transformation name>*.	cms.news.default
Item separator	Item separator displayed between records.	

Macros in web part properties

All web part properties also support macro expressions, which allow you to insert dynamic values instead of constants. Such dynamic values are evaluated at run-time, which means the web part will be able to work with context-dependent values. See Development -> Macro expressions for details.

7.2.4.14 Adding custom code to a portal page template

The most direct way to insert custom code into a portal engine-based website is using standard ASCX user controls. This chapter will show you how to do this. If you're not familiar with Visual Studio development, you can skip this chapter.

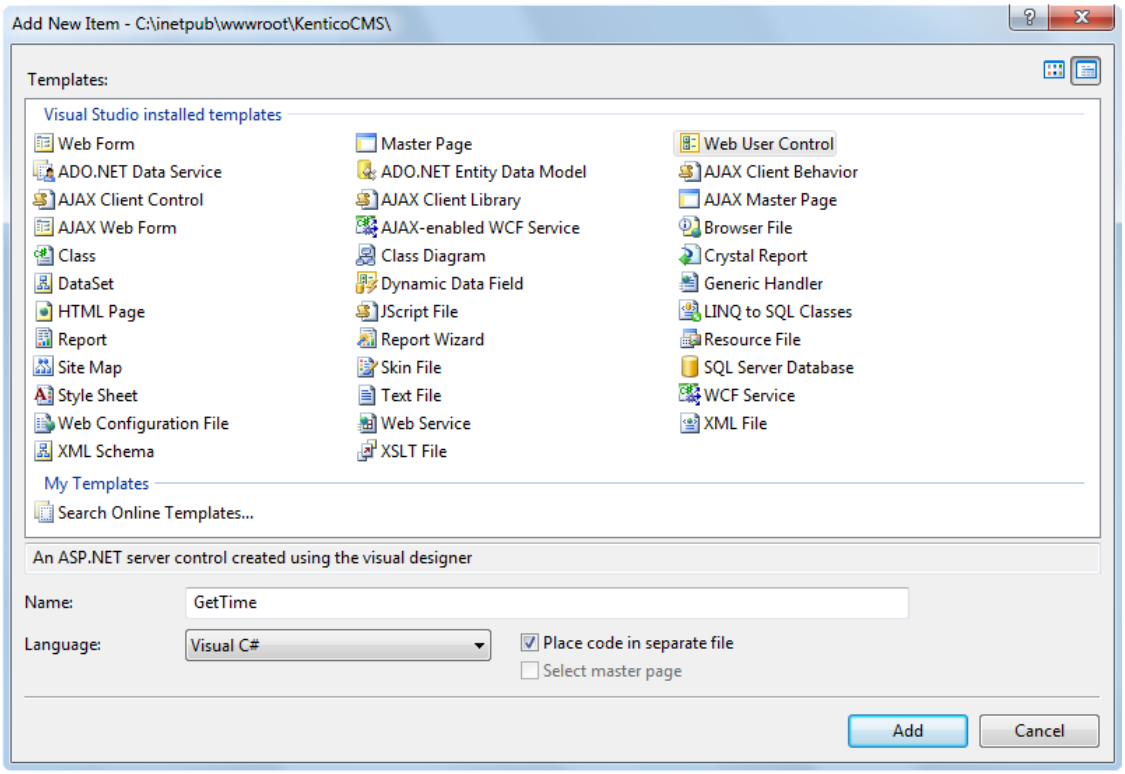
Current time example

In this example, we will create a simple user control (ASCX) using Visual Studio and integrate it into our home page.

Open the website project using the WebProject.sln file that is placed in the folder where you deployed the website. Right-click the web project in the Solution Explorer window and click **New Folder**. Name the folder the same as the code name of your site, e.g. CorporateSite - this folder will be exported with

your project when you decide to export the site and import it on a live server.

Right-click the new folder and click the **Add new item...** option. Choose to create a new Web User Control and set its name to **GetTime.ascx**. You can set the programming language option to either Visual C# or Visual Basic.



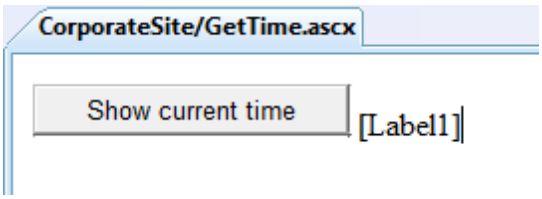
Click **Add**. Switch to the **Design** tab and drag and drop the following controls and set their properties:

Button control:

- **ID**: Button1
- **Text**: Show current time

Label control:

- **ID**: Label1
- **Text**: <clear the value>



Double-click the **Show current time** button and enter the following code into the **Button1_Click** method:

[C#]

```
Label1.Text = DateTime.Now.ToString();
```

[VB.NET]

```
Label1.Text = DateTime.Now.ToString()
```

This code ensures that the label displays the current date and time when the button is clicked.

Save all changes. It's not necessary to compile the project — user controls are compiled automatically at run time.

Adding the user control on the page

Sign in to **Kentico CMS Desk**, select the **Home** page and open the **Design** tab. Use the web part toolbar or the **Add web part** () button to insert the **User control** web part from the **General** category into the **Main zone**. Enter the following value into the **User control virtual path** property:

~/CorporateSite/GetTime.ascx

(the folder name must reflect the folder that you previously created)

The ~ character represents the root of your web application. Click **OK**. Switch to **Live site** mode and you will be able to see the user control on the page. When you click the **Show current time** button now, the current date and time is displayed next to the button.

In this short example, you have seen that you can easily add any custom code developed as an ASCX user control in Visual Studio. This user control can contain any .NET controls, third-party controls or ADO.NET code that will retrieve data from an external database.

User controls versus web parts

Another option how to insert custom code into the page is creating your own web part. A web part is basically an ASCX user control, but it inherits some standardized properties and methods from the *CMSAbstractWebPart* class. You will usually build web parts in cases where you need to create re-usable, parameterized user controls. Web part development is described in the Development -> Web parts -> Developing web parts topic.

7.2.4.15 Displaying data from an external database or Web Service

Besides displaying Kentico CMS content, you can also display data from your external database or Web Service. In this case, you need to develop a user control (ASCX) that will use ADO.NET to retrieve the data or that will contact the Web Service and call its methods. Since you can place any custom code into the user control, you will simply use the standard ASP.NET code you would use when creating the website from scratch.

Example: Retrieving data from the sample Northwind database

In this simple example, you will see how to display data on the sample Corporate Site from the *Categories* table of the Northwind database using ADO.NET. You may need to use some other database if you do not have the sample Northwind database on your server.

Open the web project in Visual Studio using the **WebProject.sln** file. Create the folder **CorporateSite** and create a new user control **CustomData.ascx** in this folder. It's important to create the control in this folder so that it's exported with your website later, when you decide to import the website on your live server.

Drag and drop the standard ASP.NET **GridView** control on your user control and set its ID to **GridView1**.

Add the following line to the beginning of the code behind file:

[C#]

```
using System.Data;
using System.Data.SqlClient;
```

Add the following code to the **Page_Load** method and replace the sample details in the connection string (database, server, user id, password, ...) with appropriate real ones:

[C#]

```
// create sql connection - you could use Oracle or OLEDB provider as well
```

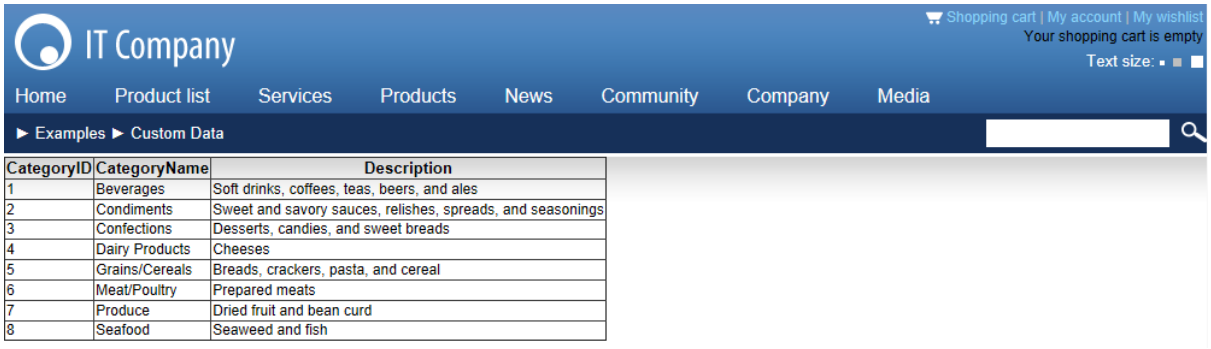
```
SqlConnection cn = new SqlConnection("Persist Security Info=False;
database=northwind;server=server1;user id=sa;password=psswd;Current
Language=English;Connection Timeout=120;");

// create data adapter
SqlDataAdapter da = new SqlDataAdapter("SELECT * FROM categories", cn);
DataSet ds = new DataSet();

// fill the dataset with data from database
da.Fill(ds);

// bind data to the grid view
GridView1.DataSource = ds;
GridView1.DataBind();
```

Save all changes.

Sign in as administrator to **CMS Desk**, select the **/Examples** document in the content tree and create a new **Page (menu item)**. Call it **Custom Data**,choose the **Create a blank page** option and click **Save**.

Switch to the **Design** tab, add a new web part **General/User control** and set its **User control virtual path** property value to *~/corporatesite/customdata.ascx*.

Click **OK** and switch to the **Live site** mode of the page. You will see a grid with data from the external database:



CategoryID	CategoryName	Description
1	Beverages	Soft drinks, coffees, teas, beers, and ales
2	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings
3	Confections	Desserts, candies, and sweet breads
4	Dairy Products	Cheeses
5	Grains/Cereals	Breads, crackers, pasta, and cereal
6	Meat/Poultry	Prepared meats
7	Produce	Dried fruit and bean curd
8	Seafood	Seaweed and fish

As you can see we used standard ASP.NET methods to display external data on the website.

7.2.4.16 Page template internals and API

7.2.4.16.1 Overview

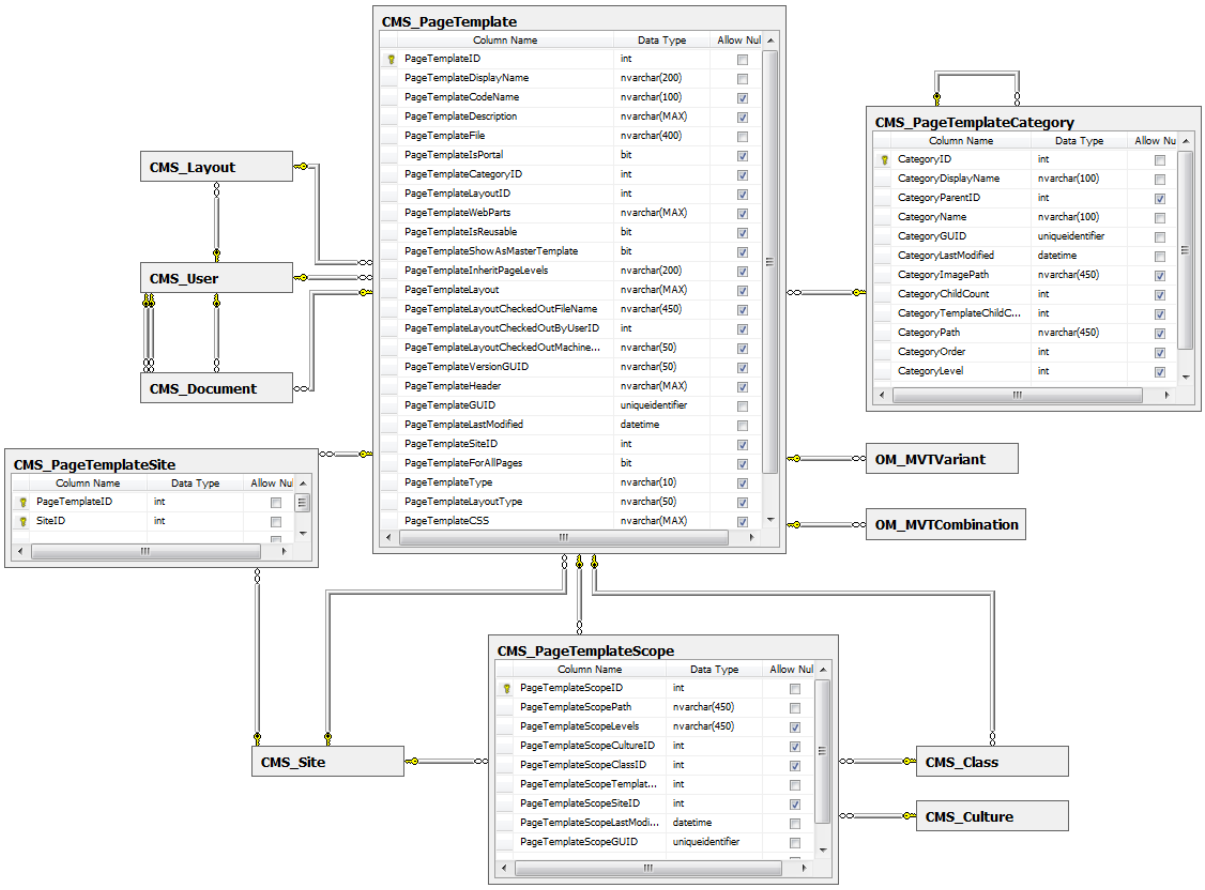
In this chapter, you will learn which database tables and API classes are used by page templates. You will also see the most common API examples.

Further API-related information and examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all members of the related classes, please refer to Kentico CMS API Reference.

7.2.4.16.2 Database tables

The following database tables are used to store information about page templates:

- **CMS_PageTemplateCategory** - contains records representing page template categories.
- **CMS_PageTemplate** - contains page templates and their content, including custom page layout code and web part content/configuration (in XML format).
- **CMS_PageTemplateSite** - stores relationships between page templates and sites. Each entry indicates that a specific page template can be used on a given site.
- **CMS_PageTemplateScope** - contains records representing page template scopes.



7.2.4.16.3 API classes

If you are not familiar with database table data management through Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

The classes of page templates can be found in the **CMS.PortalEngine** namespace.

CMS_PageTemplateCategory table API:

- **PageTemplateCategoryInfo** - represents one page template category.
- **PageTemplateCategoryInfoProvider** - provides management functionality for page template categories.

CMS_PageTemplate table API:

- **PageTemplateInfo** - represents one page template object.
- **PageTemplateInfoProvider** - provides management functionality for page templates.

CMS_PageTemplateSite table API:

- **PageTemplateSiteInfo** - represents a relationship between a page template and a site.
- **PageTemplateSiteInfoProvider** - provides management functionality for page template-site relationships.

CMS_PageTemplateScope table API:

- **PageTemplateScopeInfo** - represents one page template scope.
- **PageTemplateScopeInfoProvider** - provides management functionality for page template scopes.

Other classes:

- **PageTemplateInstance** - represents a specific instance of a page template.

7.2.4.16.4 API examples

7.2.4.16.4.1 Overview

These topics show examples of how the API of the page template can be used:

- Managing page template categories
- Managing page templates
- Page templates and sites
- Managing page template scopes

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Development\PageTemplates\Default.aspx.cs***.

The page template API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.GlobalHelper;
using CMS.CMSHelper;
using CMS.PortalEngine;
```

7.2.4.16.4.2 Managing page template categories

The following example creates a page template category.

```
private void CreatePageTemplateCategory()
{
    // Create new page template category object
    PageTemplateCategoryInfo newCategory = new PageTemplateCategoryInfo();

    // Set the properties
    newCategory.DisplayName = "My new category";
    newCategory.CategoryName = "MyNewCategory";

    // Save the page template category
    PageTemplateCategoryInfoProvider.SetPageTemplateCategoryInfo(newCategory);
}
```

The following example gets and updates a page template category.

```
private bool GetAndUpdatePageTemplateCategory()
{
    // Get the page template category
    PageTemplateCategoryInfo updateCategory =
PageTemplateCategoryInfoProvider.GetPageTemplateCategoryInfo("MyNewCategory");
    if (updateCategory != null)
    {
        // Update the properties
        updateCategory.DisplayName = updateCategory.DisplayName.ToLower();

        // Save the changes
        PageTemplateCategoryInfoProvider.SetPageTemplateCategoryInfo
```

```
(updateCategory);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates page template categories.

```
private bool GetAndBulkUpdatePageTemplateCategories()
{
    // Prepare the parameters
    string where = "CategoryName LIKE N'MyNewCategory%'";

    // Get the data
    DataSet categories = PageTemplateCategoryInfoProvider.GetCategoriesList
(where, null);
    if (!DataHelper.DataSourceIsEmpty(categories))
    {
        // Loop through the individual items
        foreach (DataRow categoryDr in categories.Tables[0].Rows)
        {
            // Create object from DataRow
            PageTemplateCategoryInfo modifyCategory = new PageTemplateCategoryInfo
(categoryDr);

            // Update the properties
            modifyCategory.DisplayName = modifyCategory.DisplayName.ToUpper();

            // Save the changes
            PageTemplateCategoryInfoProvider.SetPageTemplateCategoryInfo
(modifyCategory);
        }

        return true;
    }

    return false;
}
```

The following example deletes a page template category.

```
private bool DeletePageTemplateCategory()
{
    // Get the page template category
    PageTemplateCategoryInfo deleteCategory =
PageTemplateCategoryInfoProvider.GetPageTemplateCategoryInfo("MyNewCategory");

    // Delete the page template category
    PageTemplateCategoryInfoProvider.DeletePageTemplateCategory(deleteCategory);
```

```
    return (deleteCategory != null);
}
```

7.2.4.16.4.3 Managing page templates

The following example creates a page template.

```
private bool CreatePageTemplate()
{
    // Get the page template category
    PageTemplateCategoryInfo category =
PageTemplateCategoryInfoProvider.GetPageTemplateCategoryInfo"MyNewCategory");
    if (category != null)
    {
        // Create new page template object
        PageTemplateInfo newTemplate = new PageTemplateInfo();

        // Set the properties
        newTemplate.DisplayName = "My new template";
        newTemplate.CodeName = "MyNewTemplate";
        newTemplate.Description = "This is page template created by API Example";
        newTemplate.PageTemplateSiteID = CMSContext.CurrentSiteID;
        newTemplate.FileName = " ";
        newTemplate.ShowAsMasterTemplate = false;
        newTemplate.IsPortal = true;
        newTemplate.InheritPageLevels = ""; // inherits all
        newTemplate.IsReusable = true;
        newTemplate.CategoryID = category.CategoryId;

        // Save the page template
        PageTemplateInfoProvider.SetPageTemplateInfo(newTemplate);

        return true;
    }

    return false;
}
```

The following example gets and updates a page template.

```
private bool GetAndUpdatePageTemplate()
{
    // Get the page template
    PageTemplateInfo updateTemplate = PageTemplateInfoProvider.GetPageTemplateInfo
("MyNewTemplate");
    if (updateTemplate != null)
    {
        // Update the properties
        updateTemplate.DisplayName = updateTemplate.DisplayName.ToLower();
```

```
        // Save the changes
        PageTemplateInfoProvider.SetPageTemplateInfo(updateTemplate);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates page templates.

```
private bool GetAndBulkUpdatePageTemplates()
{
    // Prepare the parameters
    string where = "PageTemplateCodeName LIKE N'MyNewTemplate%'";

    // Get the data
    DataSet templates = PageTemplateInfoProvider.GetTemplates(where, null);
    if (!DataHelper.DataSourceIsEmpty(templates))
    {
        // Loop through the individual items
        foreach (DataRow templateDr in templates.Tables[0].Rows)
        {
            // Create object from DataRow
            PageTemplateInfo modifyTemplate = new PageTemplateInfo(templateDr);

            // Update the properties
            modifyTemplate.DisplayName = modifyTemplate.DisplayName.ToUpper();

            // Save the changes
            PageTemplateInfoProvider.SetPageTemplateInfo(modifyTemplate);
        }

        return true;
    }

    return false;
}
```

The following example deletes a page template.

```
private bool DeletePageTemplate()
{
    // Get the page template
    PageTemplateInfo deleteTemplate = PageTemplateInfoProvider.GetPageTemplateInfo
("MyNewTemplate");

    // Delete the page template
    PageTemplateInfoProvider.DeletePageTemplate(deleteTemplate);
```

```
    return (deleteTemplate != null);
}
```

7.2.4.16.4.4 Page templates and sites

The following example assigns a page template to a site.

```
private bool AddPageTemplateToSite()
{
    // Get the page template
    PageTemplateInfo template = PageTemplateInfoProvider.GetPageTemplateInfo
("MyNewTemplate");
    if (template != null)
    {
        int templateId = template.PageTemplateId;
        int siteId = CMSContext.CurrentSiteID;

        // Save the binding
        PageTemplateSiteInfoProvider.AddPageTemplateToSite(templateId, siteId);

        return true;
    }

    return false;
}
```

The following example removes a page template from a site.

```
private bool RemovePageTemplateFromSite()
{
    // Get the page template
    PageTemplateInfo removeTemplate = PageTemplateInfoProvider.GetPageTemplateInfo
("MyNewTemplate");
    if (removeTemplate != null)
    {
        int siteId = CMSContext.CurrentSiteID;

        // Get the binding
        PageTemplateSiteInfo templateSite =
PageTemplateSiteInfoProvider.GetPageTemplateSiteInfo
(removeTemplate.PageTemplateId, siteId);

        // Delete the binding
        PageTemplateSiteInfoProvider.DeletePageTemplateSiteInfo(templateSite);

        return true;
    }

    return false;
}
```

7.2.4.16.4.5 Managing page template scopes

The following example creates a page template scope and assigns it to a template.

```
private bool CreatePageTemplateScope()
{
    // Get template object
    PageTemplateInfo template = PageTemplateInfoProvider.GetPageTemplateInfo
("MyNewTemplate");

    // If template exists
    if (template != null)
    {
        // Page template isn't available for all pages
        template.PageTemplateForAllPages = false;

        // Create new template scope
        PageTemplateScopeInfo newScope = new PageTemplateScopeInfo();

        // Set some properties
        newScope.PageTemplateScopeTemplateID = template.PageTemplateId;
        newScope.PageTemplateScopeSiteID = CMSContext.CurrentSiteID;
        newScope.PageTemplateScopePath = "/";
        newScope.PageTemplateScopeLevels = "/{0}/{1}";

        // Save scope to database
        PageTemplateScopeInfoProvider.SetPageTemplateScopeInfo(newScope);

        // Update page template
        PageTemplateInfoProvider.SetPageTemplateInfo(template);

        return true;
    }

    return false;
}
```

The following example removes a page template scope from a template and deletes it.

```
private bool DeletePageTemplateScope()
{
    string columns = "";
    string where = "";

    // Get template object
    PageTemplateInfo template = PageTemplateInfoProvider.GetPageTemplateInfo
("MyNewTemplate");

    // If template exists
    if (template != null)
    {
```

```
        where = "PageTemplateScopeTemplateID = " + template.PageTemplateId;
    }

    DataSet scopes = PageTemplateScopeInfoProvider.GetTemplateScopes(columns,
where, null, 0);

    if (!DataHelper.DataSourceIsEmpty(scopes))
    {
        // Get the page template scope
        PageTemplateScopeInfo deleteScope = new PageTemplateScopeInfo
(scopes.Tables[0].Rows[0]);

        // Delete the page template scope
        PageTemplateScopeInfoProvider.DeletePageTemplateScopeInfo(deleteScope);

        return true;
    }

    return false;
}
```

7.2.4.17 Page layout internals and API

7.2.4.17.1 Overview

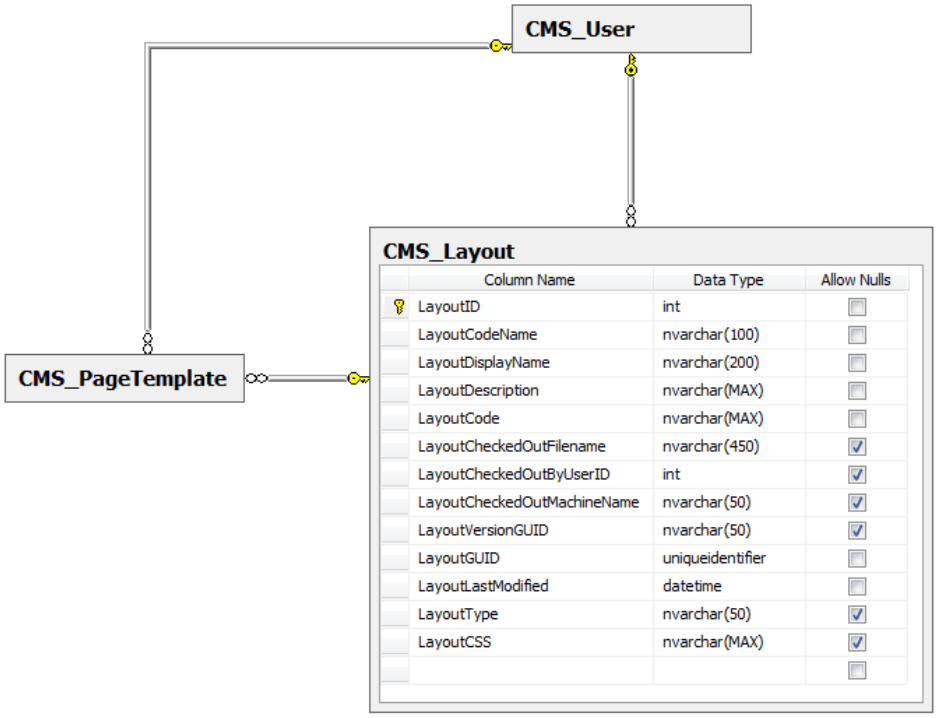
In this chapter, you will learn which database tables and API classes are used by page layouts. You will also see the most common API examples.

Further API-related information and examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all members of the related classes, please refer to Kentico CMS API Reference.

7.2.4.17.2 Database tables

The following database table is used to store information about page layouts:

- **CMS_Layout** - contains records representing shared page layouts.



Custom page layouts

Only pre-defined page layouts are stored in the **CMS_Layout** table. These can be shared by multiple page templates.

However, the code of custom layouts that are unique for specific page templates is saved in the **PageTemplateLayout** column of the corresponding record in the **CMS_PageTemplate** table. This table also has columns that track the check-out status of custom page layouts.

7.2.4.17.3 API classes

If you are not familiar with database table data management through Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

The classes for managing page layouts can be found in the **CMS.PortalEngine** namespace.

CMS_Layout table API:

- **LayoutInfo** - represents one page layout object.
- **LayoutInfoProvider** - provides management functionality for page layouts.

7.2.4.17.4 API examples

7.2.4.17.4.1 Overview

The following topic shows examples of how the API of page layouts can be used:

- Managing page layouts

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Development\PageLayouts\Default.aspx.cs***.

The page layout API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.GlobalHelper;
using CMS.PortalEngine;
```

7.2.4.17.4.2 Managing page layouts

The following example creates a page layout.

```
private void CreateLayout()
{
    // Create new layout object
    LayoutInfo newLayout = new LayoutInfo();

    // Set the properties
    newLayout.LayoutDisplayName = "My new layout";
    newLayout.LayoutCodeName = "MyNewLayout";
    newLayout.LayoutDescription = "This is layout created by API Example";
    newLayout.LayoutCode = "<cc1:CMSWebPartZone ZoneID=\"zoneLeft\"
runat=\"server\" />";

    // Save the layout
    LayoutInfoProvider.SetLayoutInfo(newLayout);
}
```

The following example gets and updates a page layout.

```
private bool GetAndUpdateLayout()
{
    // Get the layout
    LayoutInfo updateLayout = LayoutInfoProvider.GetLayoutInfo("MyNewLayout");
    if (updateLayout != null)
    {
        // Update the properties
        updateLayout.LayoutDisplayName = updateLayout.LayoutDisplayName.ToLower();

        // Save the changes
        LayoutInfoProvider.SetLayoutInfo(updateLayout);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates page layouts.

```
private bool GetAndBulkUpdateLayouts()
{
    // Prepare the parameters
    string where = "LayoutCodeName LIKE N'MyNewLayout%'";

    // Get the data
    DataSet layouts = LayoutInfoProvider.GetLayouts(where, null);
    if (!DataHelper.DataSourceIsEmpty(layouts))
    {
        // Loop through the individual items
        foreach (DataRow layoutDr in layouts.Tables[0].Rows)
        {
            // Create object from DataRow
            LayoutInfo modifyLayout = new LayoutInfo(layoutDr);

            // Update the properties
            modifyLayout.LayoutDisplayName =
modifyLayout.LayoutDisplayName.ToUpper();

            // Save the changes
            LayoutInfoProvider.SetLayoutInfo(modifyLayout);
        }

        return true;
    }

    return false;
}
```

The following example deletes a page layout.

```
private bool DeleteLayout()
```

```
{
    // Get the layout
    LayoutInfo deleteLayout = LayoutInfoProvider.GetLayoutInfo("MyNewLayout");

    // Delete the layout
    LayoutInfoProvider.DeleteLayoutInfo(deleteLayout);

    return (deleteLayout != null);
}
```

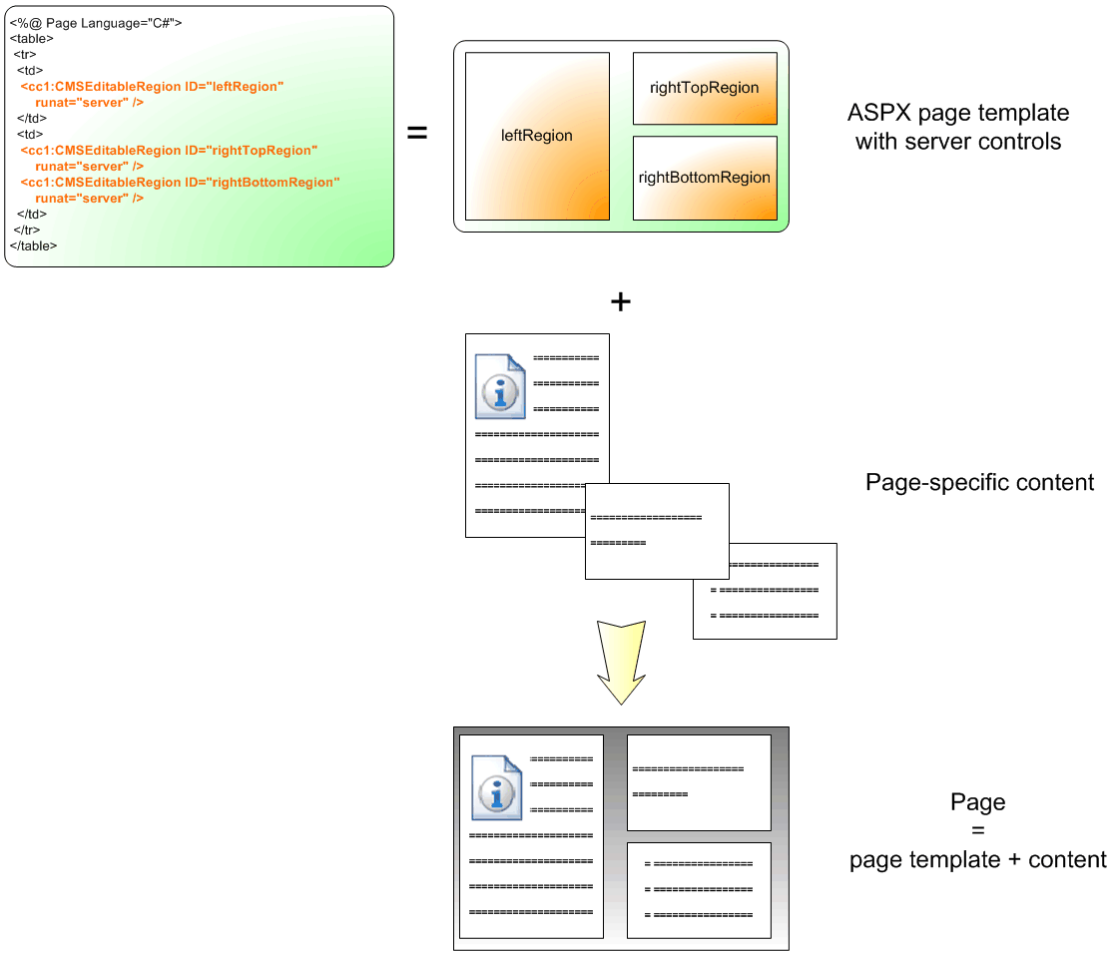
7.2.5 ASPX page template development model

7.2.5.1 Overview

If you are familiar with ASP.NET development in Visual Studio, you may want to choose to develop websites using standard ASPX page templates. ASPX page templates in Kentico CMS are standard ASP.NET pages that display content from Kentico CMS. They receive the document **aliasPath** as a URL parameter that informs the page template which page should be displayed.

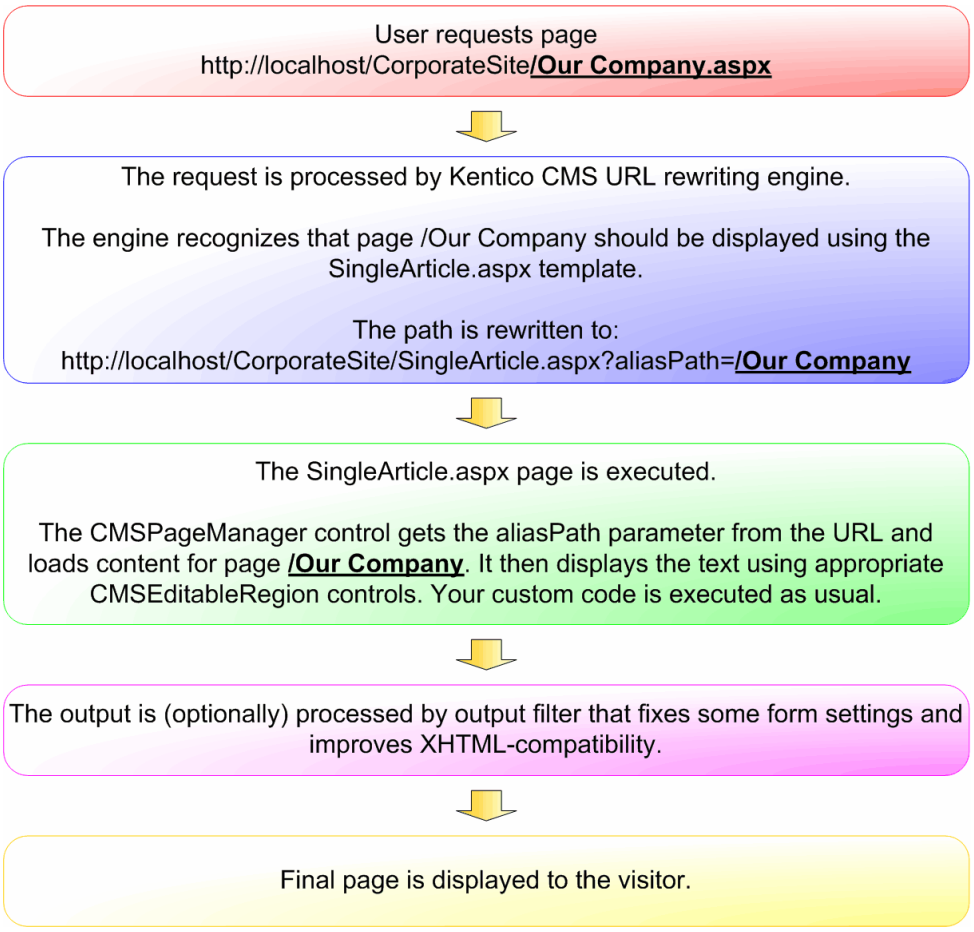
What does an ASPX page template consist of?

A page template is a combination of static HTML code and ASP.NET server controls (or user controls) that render dynamic content. You can also use code behind (using either VB.NET or C#) to modify page behavior and add custom functionality. The following figure illustrates how an ASPX page template and page content are combined to display a page:



How are ASPX page templates processed?

When a user requests a page, such as **~/services/web-development.aspx**, the system calls the assigned page template with the **aliasPath** URL parameter, which specifies what content (which page from the content tree) should be displayed using the given template:



The built-in Kentico CMS controls understand the **aliasPath** parameter in the URL and render the appropriate content automatically.

As you can see, the system uses standard ASP.NET architecture. If you developed the website without Kentico CMS, you would most likely use URLs like this: **/news.aspx?newsid=127** which is similar to **/news.aspx?aliaspath=/news/your-first-news.aspx** as used in Kentico CMS. Kentico CMS also uses friendly URLs in format **/news/your-first-news.aspx** that are better for search engine optimization.

7.2.5.2 Creating a new ASPX page template

This topic describes how to create a new ASPX page template. We will create a new page with two columns that contain editable regions.

Adding Kentico CMS Controls to your Visual Studio Toolbox

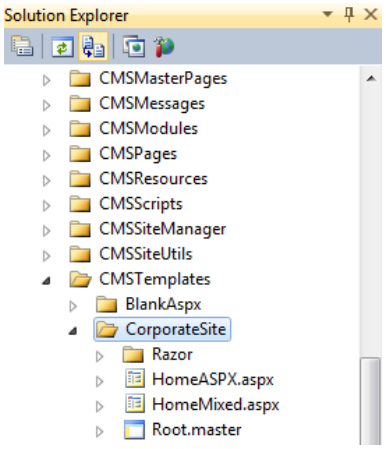
Before you start developing ASPX page templates, it is recommended to add the Kentico CMS Controls to your Visual Studio Toolbox so that you can drag-and-drop the controls onto the ASPX pages.

See the Adding Kentico CMS Controls to the Toolbox topic for step-by-step

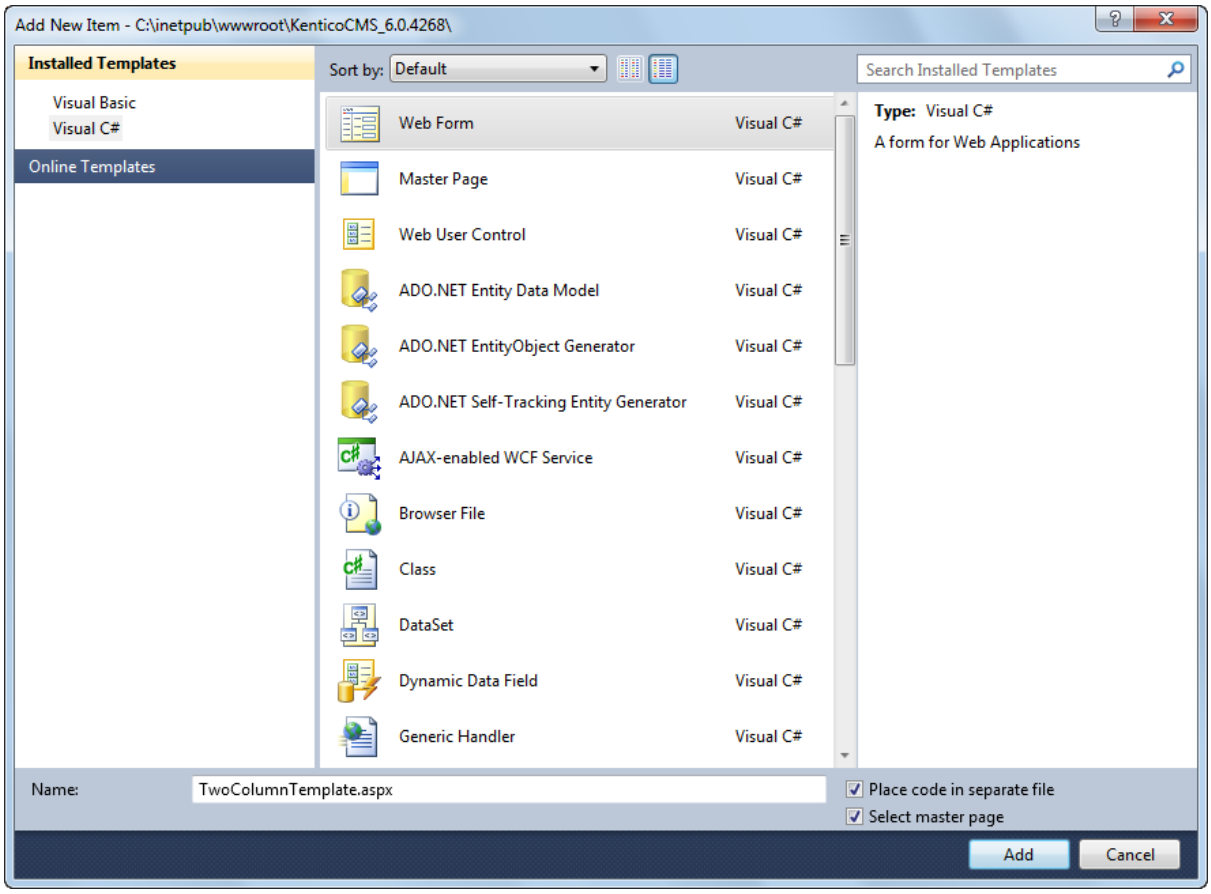
instructions.

1. Open the web project in Visual Studio. You can open it either through the **WebProject.sln** file or using the **File -> Open Web Site** option in the menu.

2. Right-click the **CMSTemplates/CorporateSite** folder in the Solution Explorer and select **Add New Item**.

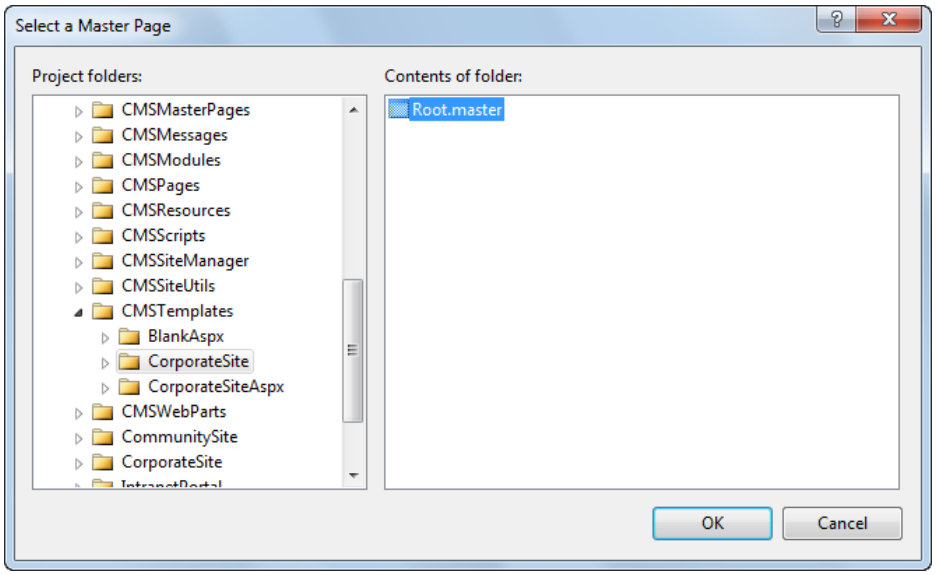


3. Create a new **Web form** named *TwoColumnTemplate.aspx* and check the **Select master page** box. Click **Add**.



The **Select a Master Page** dialog opens.

4. Choose the **CMSTemplates/CorporateSite** folder, select the **Root.master** file and click **OK**.



Writing the ASPX code

5. Open the **Source** view of the newly created ASPX page and add the following code inside the **<asp: Content>** element:

```
<table width="100%">
  <tr valign="top">
    <td width="50%">
        <cms:CMSEditableRegion ID="txtLeft" runat="server" DialogHeight="400"
            RegionType="HtmlEditor" RegionTitle="Left column" />
    </td>
    <td width="50%">
        <cms:CMSEditableRegion ID="txtText" runat="server" DialogHeight="400"
            RegionType="HtmlEditor" RegionTitle="Right column" />
    </td>
  </tr>
</table>
```

The **<asp:Content>** control allows you to use standard ASP.NET master pages. When the system renders the page, it loads the content of the control into the assigned master page (as defined in the *Root.master* file).

The **CMSEditableRegion** control defines an editable region that the page displays as an HTML editor on the **Content -> Edit -> Page** tab of **CMS Desk**. On the live site, the control renders the content entered into the editor.

Note

This example uses a table layout. If you prefer a CSS layout, replace the surrounding HTML code with <DIV> elements. You have full control over the content.

6. Switch to the code behind file (*TwoColumnTemplate.aspx.cs*) and add a reference to the **CMS.UIControls** namespace:

[C#]

```
using CMS.UIControls;
```

7. Modify the class definition so that it inherits from the **TemplatePage** class:

[C#]

```
public partial class CMSTemplates_CorporateSite_TwoColumnTemplate : TemplatePage
```

Inheriting from this class allows you to use the web form as a page template in Kentico CMS.

Keep in mind that the name of the class must be identical to the value of the **Inherits** attribute of the **<% @ Page %>** directive on the ASPX page. This is case sensitive.

Registering the ASPX page as a page template

Now that we have created a new ASPX page, we need to register it in Kentico CMS as a page template, so that it can be used by content editors.

1. Sign in to **Site Manager** and go to **Development -> Page templates**.

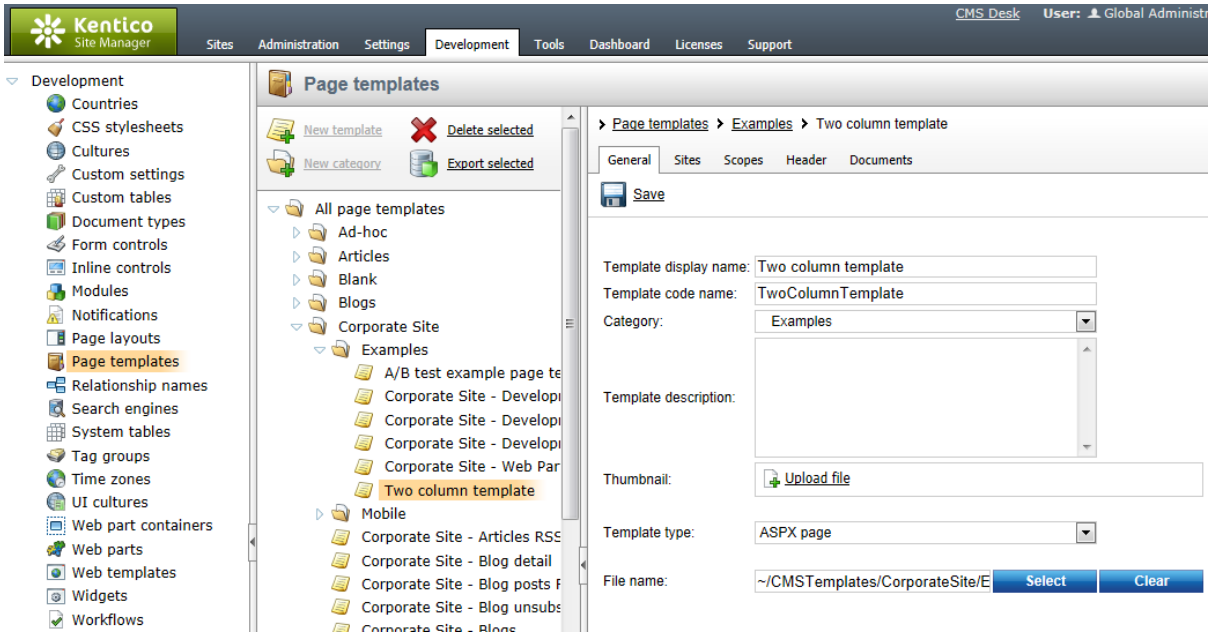
2. Select the **Corporate Site/Examples** folder, click **New template** and type *Two column template* into the **Template display name** field.

3. Click **Save**. The system creates the template and displays its **General** tab.

4. Set the following values on the **General** tab:

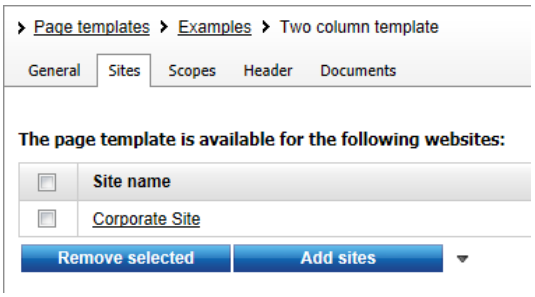
- **Template type**: ASPX
- **File name**: ~/CMSTemplates/CorporateSite/TwoColumnTemplate.aspx

This is the virtual path of the ASPX page. Alternatively, you can click the **Select** button and manually select the file.



5. Click **Save**.

6. Switch to the **Sites** tab and click the **Add sites** button. Choose the sites where you wish to use the page template and click **OK**.



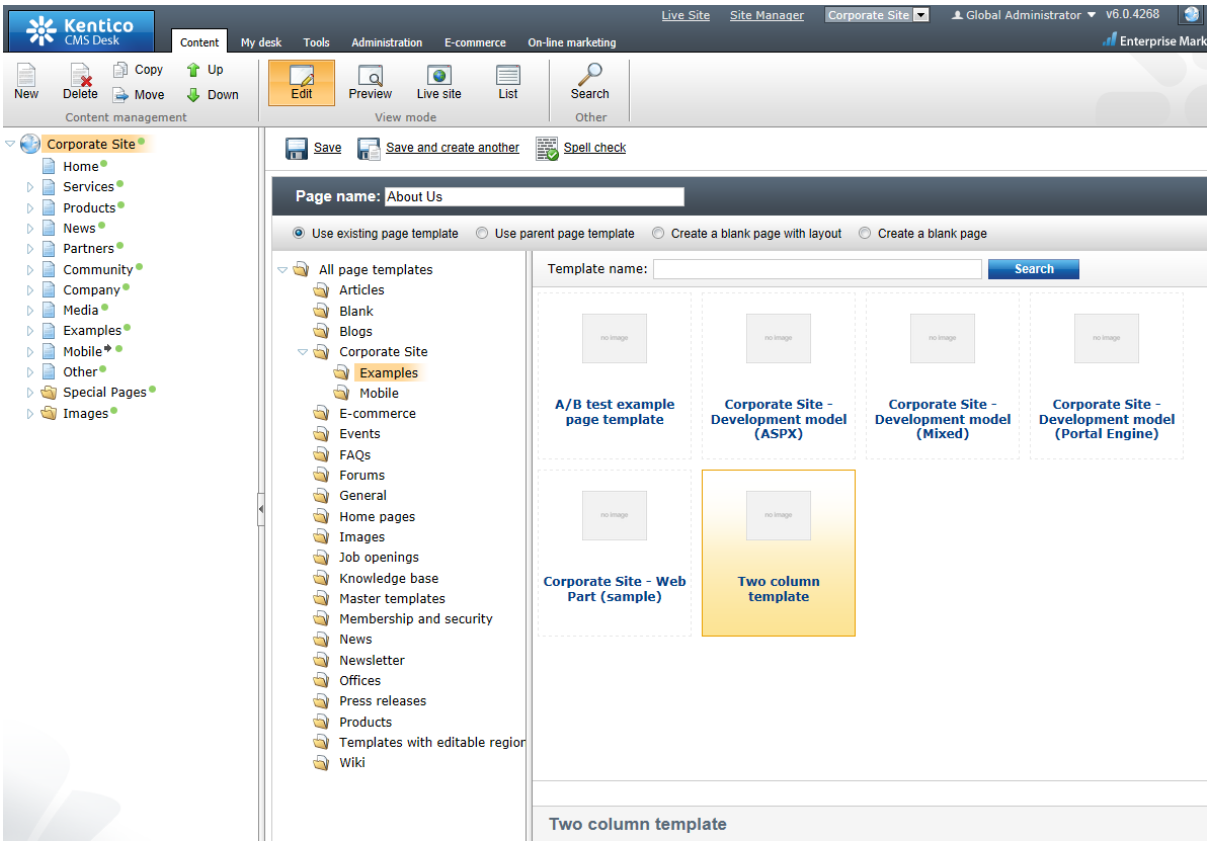
Creating an About Us page based on the new page template

1. Go to **CMS Desk -> Content**.

2. Select **Corporate Site** (the root of the content tree) and click **New** in the main menu of the **Content** section.

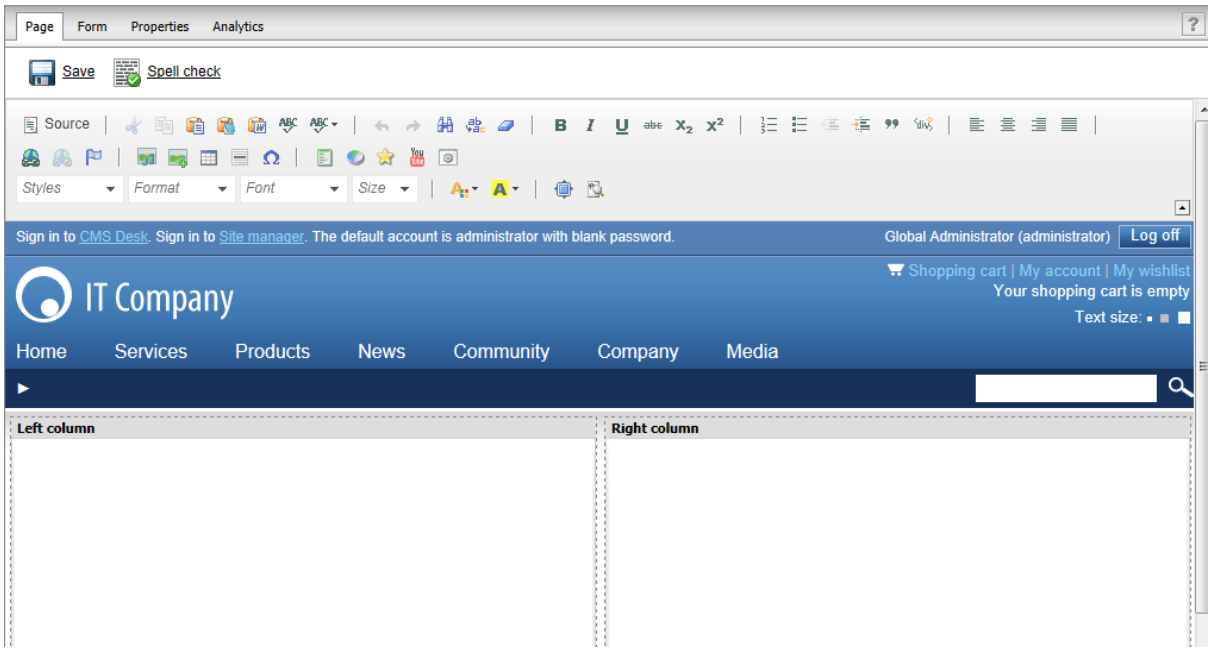
3. Choose the **Page (menu item)** document type.

4. Type *About Us* into the **Page name** field and choose the **Use existing page template** option. Select the **Corporate Site/Examples** category and the **Two column template** page template.



5. Click **Save** to create the new page.

On the **Page** tab, you can see the page and its editable regions.



You have created your first page based on an ASPX page template. Now you can enter some text and click **Save** to store the content.

7.2.5.3 Creating ASPX master pages

You can use standard ASP.NET master pages together with ASPX page templates. This is a powerful concept that allows you to share content across all pages without having to add it separately to every page template. For example, you can create master pages containing header and footer sections with a logo, navigation menu, search box etc.

Master pages are defined in files with the **.master** extension. You can assign one master page to every ASPX page. Master pages must always contain one or more **ContentPlaceHolder** controls as shown here:

```
<asp:ContentPlaceHolder ID="plcMain" runat="server"></asp:ContentPlaceHolder>
```

The **ContentPlaceHolder** control specifies where child pages display their content inside the master page.

It is recommended to store master pages in the **CMSTemplates** folder together with the page template files, so that the system can export them along with your website when you deploy it to another instance of Kentico CMS.

Preparing master pages for ASPX templates

The following code sample shows the markup of a basic master page.

Important!

If you installed the Kentico CMS project as a web application, you need to rename the *CodeFile* attribute on the first line to *Codebehind* for the code example to be functional. This attribute's value needs to match the name of the master page's code behind file.

Set the value of the *Inherits* attribute according to the location and name of the master page file.

```
<%@ Master Language="C#" AutoEventWireup="true" CodeFile="Custom.master.cs"
Inherits="CMSTemplates_CorporateSite_Custom" %>

<%=DocType%>

<html xmlns="http://www.w3.org/1999/xhtml" <%=XmlNamespace%>>
<head id="Head1" runat="server">
    <title id="Title1" runat="server">My site</title>
    <asp:literal runat="server" id="ltlTags" enableviewstate="false" />
</head>
<body class="<%=BodyClass%>" <%=BodyParameters%>>
    <form id="form1" runat="server">
        <asp:PlaceHolder runat="server" ID="plcManagers">
            <ajaxToolkit:ToolkitScriptManager ID="manScript" runat="server"
EnableViewState="false" ScriptMode="Release" />
            <cms:CMSPortalManager ID="CMSPortalManager1" runat="server"
EnableViewState="false" />
        </asp:PlaceHolder>

        <cms:CMSMenu ID="cmsmenu1" runat="server" Cursor="Pointer"
HighlightAllItemsInPath="true" Layout="Horizontal" Padding="0" Spacing="1" />

        <asp:ContentPlaceHolder ID="plcMain" runat="server">
        </asp:ContentPlaceHolder>
    </form>
</body>
</html>
```

All ASPX page templates require the following manager controls, so it is a good practice to add them onto your website's master page:

- **ajaxToolkit:ToolkitScriptManager** - allows pages to use AJAX components. This is required by all pages that contain the *CMSPortalManager* control.
- **CMSPortalManager** - ensures the transferring of content between the database and editable regions. It also provides the management functionality needed for portal engine zones. Always place this control after the *ToolkitScriptManager* control.

CMSPageManager control

You may use the **CMSPageManager** control instead of the *CMSPortalManager*. This control is an older equivalent available for the purposes of backwards compatibility. It does not support ASPX templates containing portal engine web part or widget zones.

The **CMSMenu** control is one of the options that you can use to generate a drop-down menu for website navigation.

Writing the master page code behind

You also need to modify the code behind file of the master page:

1. Add a reference to the **CMS.UIControls** namespace:

[C#]

```
using CMS.UIControls;
```

[VB.NET]

```
Imports CMS.UIControls
```

2. Change the class definition to match the following (the name of the class may be different):

[C#]

```
public partial class CMSTemplates_CorporateSite_Custom : TemplateMasterPage
```

[VB.NET]

```
Partial Class CMSTemplates_CorporateSite_Custom
    Inherits TemplateMasterPage
```

Master pages must always inherit from the **TemplateMasterPage** class.

3. Add the following code into the master page's code behind class:

[C#]

```
protected override void CreateChildControls()
{
    base.CreateChildControls();
    PageManager = CMSPortalManager1;
}
```

```
protected override void OnPreRender(EventArgs e)
{
    base.OnPreRender(e);
    this.ltlTags.Text = this.HeaderTags;
}
```

[VB.NET]

```
Protected Overrides Sub CreateChildControls()
    MyBase.CreateChildControls()
    PageManager = CMSPortalManager1
End Sub

Protected Overrides Sub OnPreRender(e As EventArgs)
    MyBase.OnPreRender(e)
    Me.ltlTags.Text = Me.HeaderTags
End Sub
```

Adjust the value of the **PageManager** property according to the ID of the *CMSPortalManager* (or *CMSPageManager*) control placed on the master page.

This code ensures that ASPX templates using the given master page support all required functionality.

7.2.5.4 Adding portal engine functionality to ASPX templates

When developing or maintaining a website built on ASPX page templates, one of the drawbacks is that the code of a page must be modified manually every time you wish to change its design. It is possible to add flexibility to ASPX templates by defining areas that may be edited directly through the browser in the CMS administration interface, just like when using the Portal engine development model. Depending on their configuration, these areas allow Web parts or Widgets to be used on ASPX page templates.

This is achieved by placing the following elements into the code of a page template:

```
<cms:CMSPagePlaceholder ID="plcZones" runat="server">
    <LayoutTemplate>

        ...

        <cms:CMSWebPartZone ID="zoneCenter" runat="server" />

        ...

    </LayoutTemplate>
</cms:CMSPagePlaceholder>
```

The **CMSPagePlaceholder** control creates an area on the page that behaves in a way similar to a portal engine page template. You may place multiple controls of this type onto a single page.

The **<LayoutTemplate>** element defines the layout of the area. This is typically done by specifying a table structure or a CSS-based layout applied through HTML elements (e.g. , , etc.).

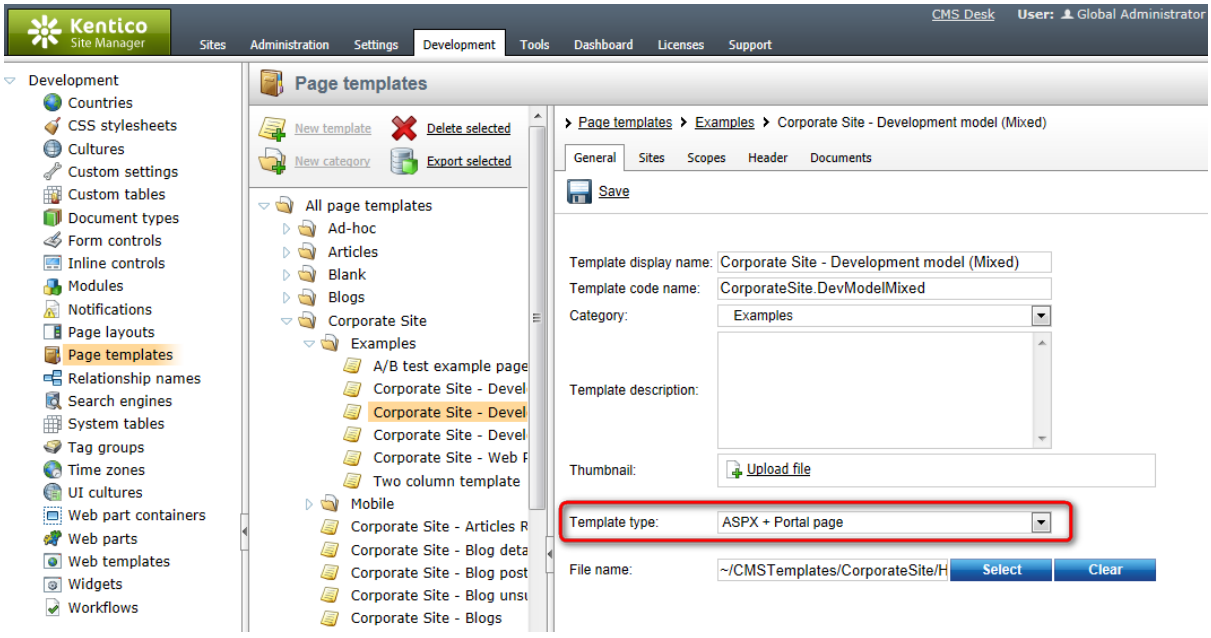
The layout may contain multiple **CMSWebPartZone** controls, which represent fully functional portal engine zones. You can configure every zone to serve as either a standard web part zone or any type of widget zone. Users may manage these zones when editing pages based on the page template on the **Edit -> Design** tab of **CMS Desk**. When web part or widget content is added to a zone, information about it is stored in the database along with the respective page template object, not in the actual code of the ASPX page.

CMSPortalManager control required!

In order for portal engine functionality to be possible on pages using ASPX templates, the **CMSPortalManager** control must be present. Usually it is located on the master page set for the template.

Some master pages may use the older **CMSPageManager** control, which does not support web part/widget zones. If this is the case, you need to replace the control before adding any of the elements described in this topic.

Page templates with this type of content also require additional configuration in **Site Manager -> Development -> Page templates**. You need to set the **Template type** to *ASPX + Portal page* in order for the **Design** tab to be available when editing pages using the given template in **CMS Desk**.



Example

The following example demonstrates the creation of a simple ASPX page template with zones that users can design via the portal engine:

1. Build a new page template according to the guide in the Creating a new ASPX page template topic. When writing its ASPX code, place the following inside the **<asp:Content>** element:

```
<cms:CMSPagePlaceholder ID="plcZones" runat="server">
    <LayoutTemplate>
        <table width="100%" cellspacing="0" cellpadding="0">
            <tr valign="top">
              <td width="50%">
                <cms:CMSWebPartZone ID="zoneLeft" runat="server" />
              </td>
              <td width="50%">
                <cms:CMSWebPartZone ID="zoneRight" runat="server" />
              </td>
            </tr>
        </table>
    </LayoutTemplate>
</cms:CMSPagePlaceholder>
```

This code defines two editable web part zones in a basic two column table layout.

2. When registering the page template in **Site Manager -> Development -> Page templates**, select the *ASPX + Portal page* option for the **Template type** property.

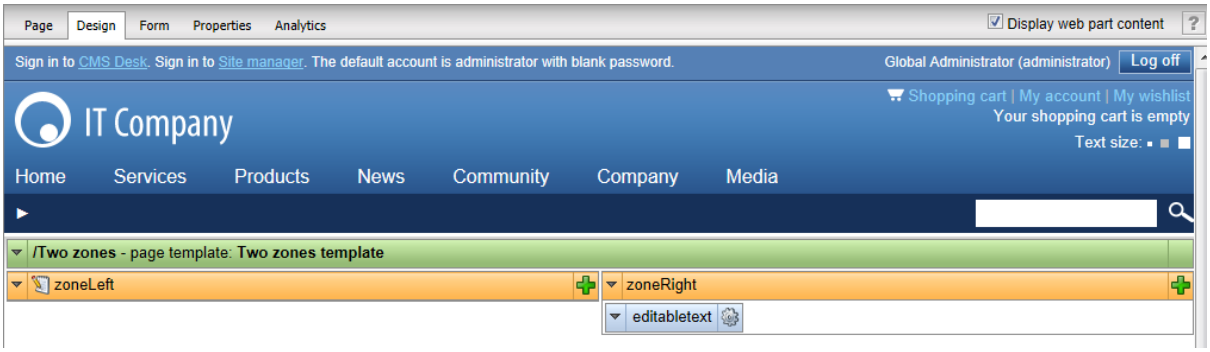
3. Go to **CMS Desk** and add a *Page (Menu item)* document to the content tree using the new page template.

4. Switch to the **Design** tab of the new page, where you can see two web part zones.

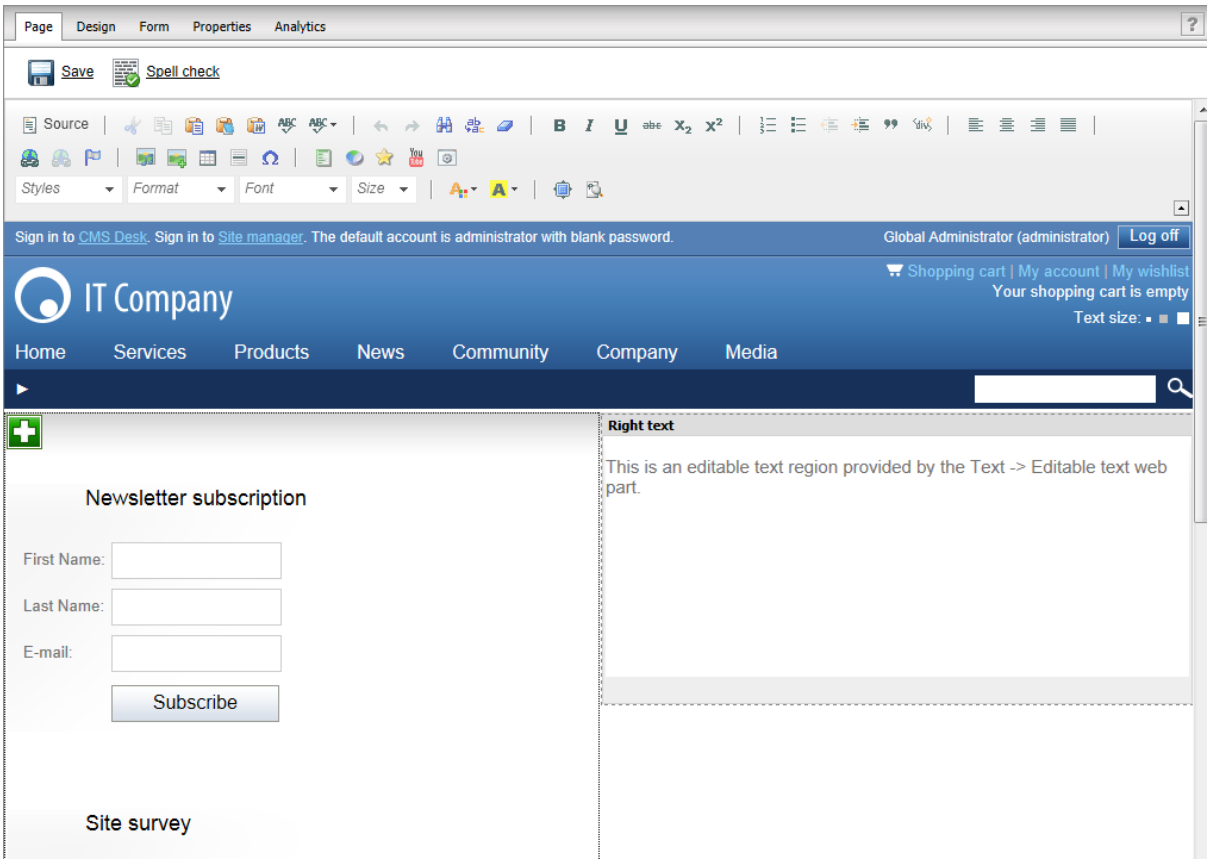
5. Expand the menu (▾) of the **zoneLeft** zone, select **Properties** and switch the **Widget zone type** property from *None* to *Customization by page editor*.

This zone now serves as a widget zone for page editors.

6. Add (✚) some web part to the **ZoneRight** zone, for example **Text & Images -> Editable text**.



7. Open the **Page** tab, where you can manage the editor widget zone on the left and enter content into the editable text region displayed by the web part on the right. Try placing some widgets onto the page using the **Add widget** (✚) button in the top left corner of the widget zone.



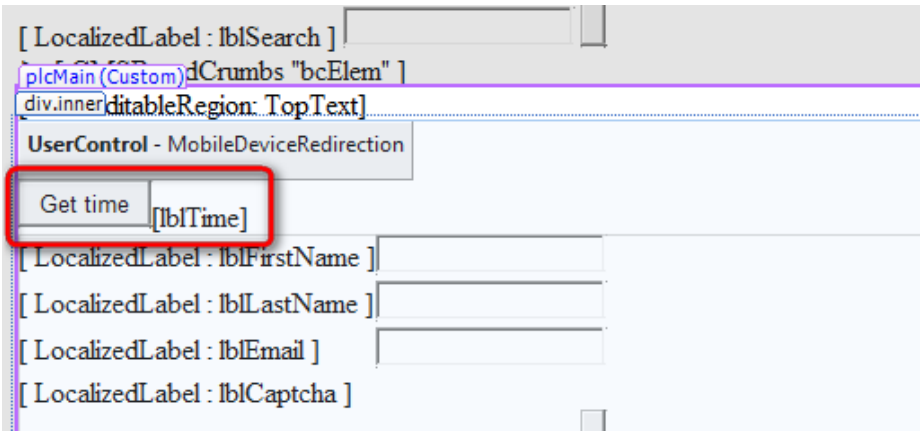
The example demonstrates how to use web parts or widgets to build the design of pages based on ASPX page templates. This approach combines the standard architecture and development process of ASPX templates with the flexibility and user-friendliness of the portal engine.

7.2.5.5 Adding custom code to an ASPX template

In this simple example, you will see how you can easily add custom code to an ASPX page template. You will see that you can add custom code in Visual Studio, as you usually would. You will need to use the sample **Corporate Site** website for this example.

1. Open the web project in Visual Studio using the **WebProject.sln** file and edit the **HomeASPX.aspx** page located in the **CMSTemplates\CorporateSite** folder.

2. Switch to the **Design** tab and add a new button to the page from the toolbox. Set its **ID** to *btnGetTime* and set its **Text** property to *Get time*. Then add a new label, name it it *lblTime* and clear its text.

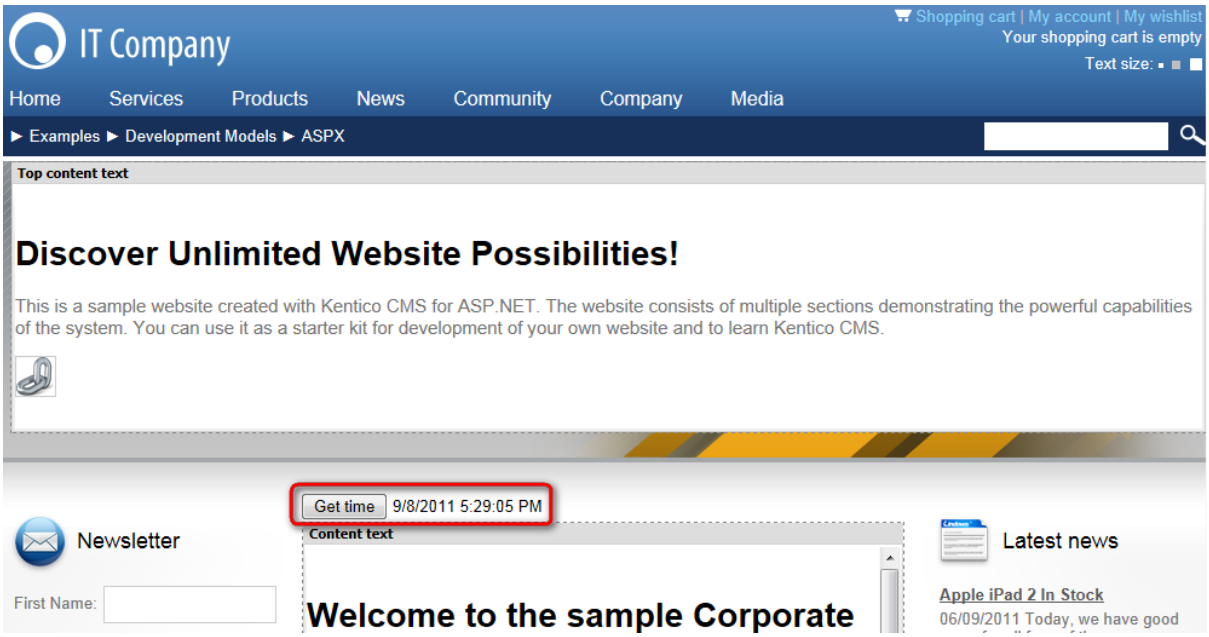


3. Double-click the button and add the following code inside the click event handler:

[C#]

```
lblTime.Text = DateTime.Now.ToString();
```

4. Save the changes and open the Corporate Site in your browser. Go to the **Examples -> Development Models -> ASPX** page, which is based on the edited template and view it in live site mode. When you click the added button, you can see that the label displays the current date and time:



As you can see, you can use standard programming methods as usual. You can also use the standard debugging process in Visual Studio.

7.2.5.6 Combining ASPX templates and portal engine templates

It is possible to combine ASPX page templates and Portal engine page templates on the same website.

Note: When creating websites with both ASPX template and portal engine pages, you need to make sure that the portal engine pages have visual inheritance disabled for all ancestors that use ASPX page templates.

Sharing a master page on hybrid websites

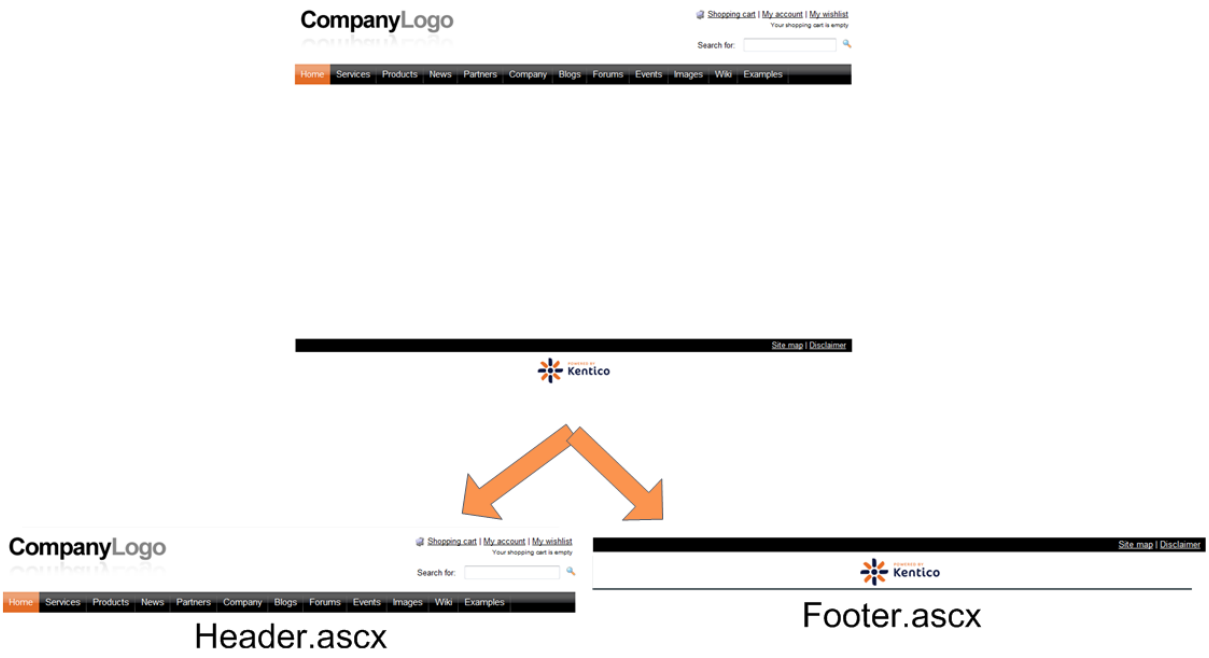
If you have a website built using the portal engine development model, the master page and all other pages use portal engine page templates. Unfortunately, you cannot "insert" ASPX pages directly into the portal engine master page.

However, you can use the following workaround:

1. Create a copy of your portal engine master template as an ASPX master page (**.master** file) and assign it to your ASPX templates.
   - This scenario works, but the drawback is that you now need to manage the master page in two locations.

2. Take all content (HTML and controls) that you have above the **Page placeholder** web part on the portal master page and place it into a new user control named **Header.ascx**.

3. Take all content from below the Page placeholder and place it into a user control named **Footer.ascx**.



4. Delete the content of your portal engine master page and replace it with the following web parts:

- **User control** - set the *User control virtual path* property to load **Header.ascx**
- **Page placeholder**
- **User control** - set the *User control virtual path* property to load **Footer.ascx**

5. Remove the content from your ASPX master page and replace it with the following controls:

- the **Header.ascx** user control
- the **ContentPlaceHolder** control
- the **Footer.ascx** user control

You can now manage the header and footer in a single place for both the portal engine and ASPX master page.

7.2.5.7 Integration with your existing ASP.NET application

If you need to integrate some existing ASP.NET application with a Kentico CMS website, there are several issues you need to consider. This chapter contains a summary of this topic, if you need more details or help with some particular issues, please contact Kentico support.

Location of CMS and your application

There are three ways how you can organize the CMS web project and your application web project:

1. Mixing both together

It makes sense to mix both applications into a single project if you wish to share functionality, content, security information and session or application variables between the CMS and your application. The easiest way is to use the Kentico CMS web project, as the main project since it's already correctly configured for the CMS, and add your own ASPX pages and other files to this project.

If you need to display your own ASPX pages inside the context of the website, you can simply register them as page templates and then create new pages based on these page templates in the standard website navigation (in the content tree). You will also need to modify your ASPX pages so that they use the master page (.master file) of the Kentico CMS website.

If you wish to use a website built using the Kentico CMS portal engine development model, please also read the Combining ASPX templates and portal engine templates topic.

2. Having the CMS in the root and your application in a sub-folder

If your application can or needs to run separately from the CMS and you want the CMS to manage the main website, you can place the Kentico CMS web project in the root of the website and your application into a sub-folder. You will need to create a virtual directory for the sub-folder so that your application runs correctly.

3. Having your application in the root and the CMS in a sub-folder

If your application can or needs to run separately from the CMS and your application is the main part of the website and you wish to use the CMS only for some sub-section of the website where you publish the content, you need to place the CMS into a sub-folder and create a virtual directory for it.

Interaction between the CMS and your application

If you need to include your application inside the website (front-end), you can do so either through ASPX pages (see paragraph **Mixing both together** above) or you need to create ASCX user controls that you

place into the CMS website.

If you need to build an application that will interact with the CMS system, but will mostly provide back-end user interface, you can create a custom module as described in the Custom modules topic.

Sharing security information between the CMS and your application (single-sign-on)

If you wish to use single-sign-on for both your application and the CMS, you need to configure your environment as described in the Single sign-on topic. If you wish to use a single system of permissions (authorization), you can leverage the permission system for custom modules as described in the Custom modules topic.

7.2.5.8 Displaying data from an external database

Besides displaying Kentico CMS content, you can also display data from your external database or Web Service. In this case, you need to add custom code to your ASPX page that will use ADO.NET to retrieve the data or that will contact the Web Service and call its methods. Since you can place any custom code into the page (page template), you will simply use the standard ASP.NET code you would use when creating the website from scratch.

Example: Retrieving data from the sample Northwind database

In this simple example, you will see how to display data on the sample Corporate Site website from the *Categories* table of the Northwind database using ADO.NET. You may need to use some other database if you do not have the sample Northwind database on your server.

Open the web project in Visual Studio using the **WebProject.sln** file. Open the *CMSTemplates/CorporateSite/HomeASPX.aspx* page.

Drag and drop the standard ASP.NET **GridView** Data control onto the page and set its ID to **GridView1**.

Add the following line to the beginning of the code behind file:

[C#]

```
using System.Data;
using System.Data.SqlClient;
```

Add the following code to the **Page_Load** method:

[C#]

```
// create sql connection - you could use Oracle or OLEDB provider as well
SqlConnection cn = new SqlConnection("Persist Security Info=False;
database=northwind;server=server1;user id=sa;password=psswd;Current
Language=English;Connection Timeout=120;");
```

```
// create data adapter
SqlDataAdapter da = new SqlDataAdapter("SELECT * FROM categories", cn);
DataSet ds = new DataSet();

// fill the dataset with data from database
da.Fill(ds);

// bind data to the grid view
GridView1.DataSource = ds;
GridView1.DataBind();
```

Save all changes. Open the Corporate Site and look at the **Examples -> Development Models -> ASPX** page on the live website. You will see a grid with data from the external database:

CategoryID	CategoryName	Description
1	Animals	Dogs, cats, horses
2	Colors	Red, green, blue, orange,...
3	Names	John, Kevin, Julia, Robin

As you can see we used standard ASP.NET methods to display external data on the website.

7.2.6 MVC development model

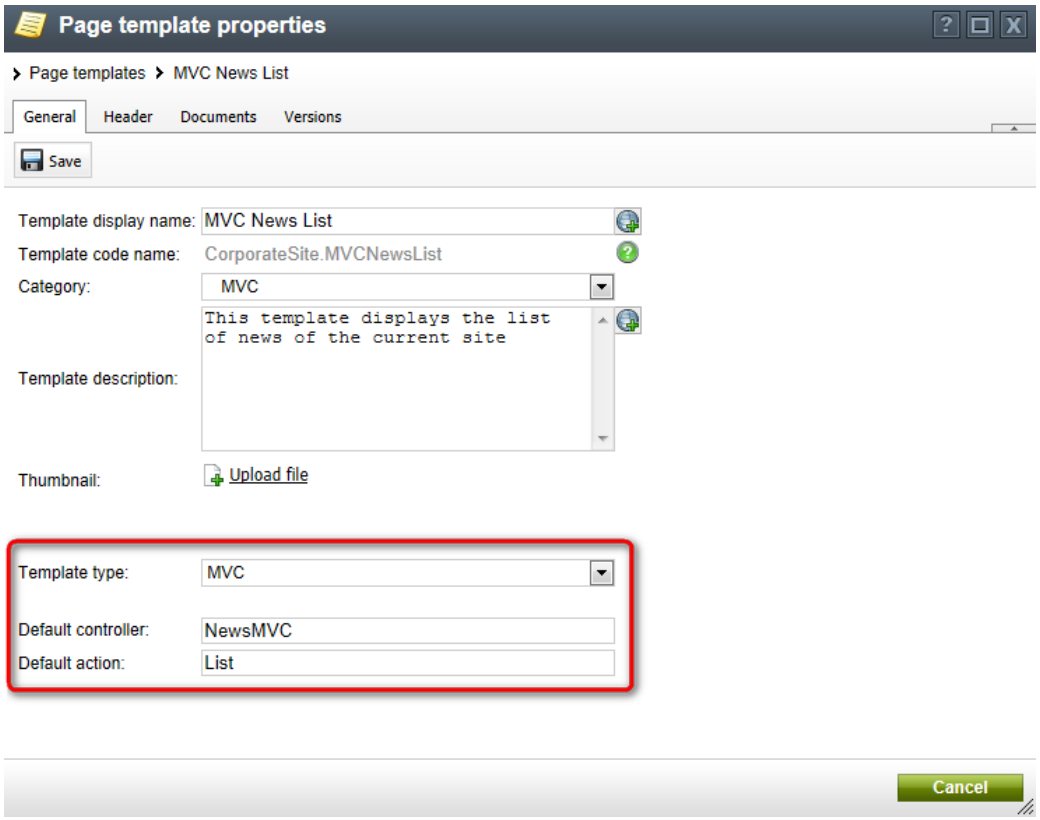
7.2.6.1 MVC development overview

Kentico CMS supports website development using ASP.NET MVC. You can find an example of a page developed using the MVC development model in the **Examples/Development-models/MVC** section of the sample **Corporate Site**.

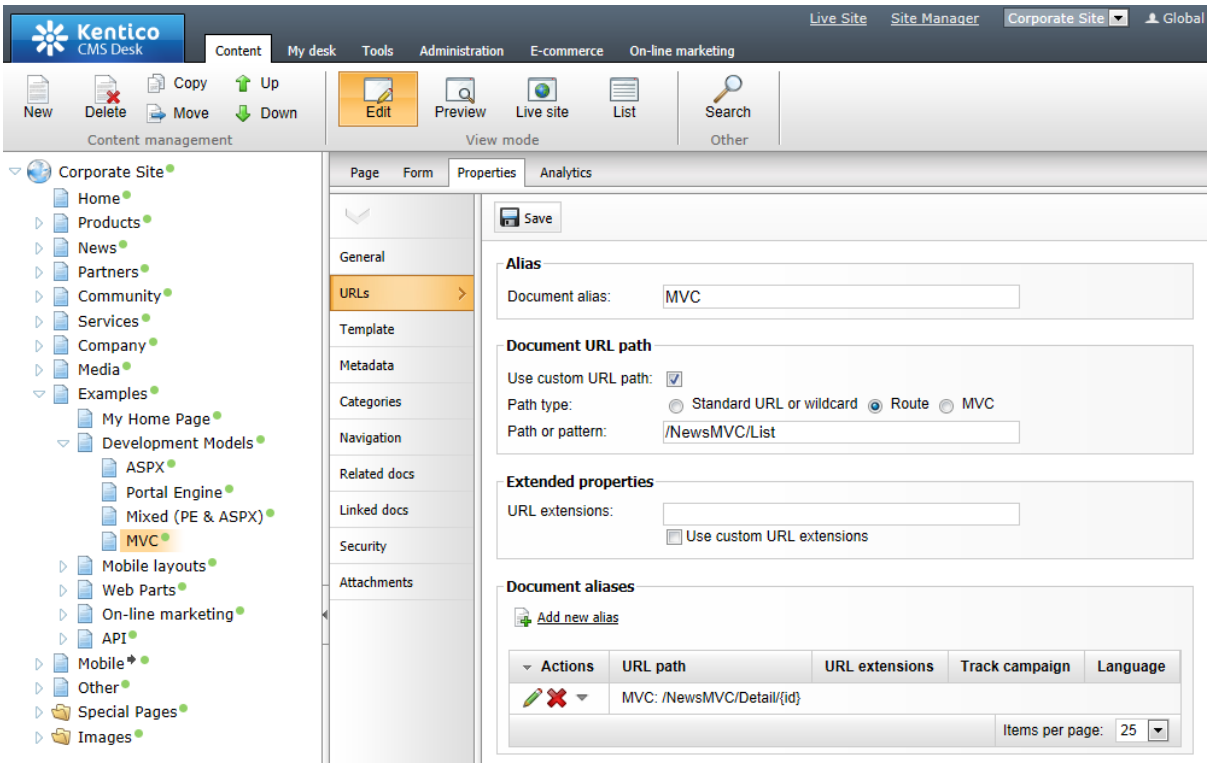
Document configuration

The document uses an **MVC** type page template, which specifies the default controller, as well as the action that is executed when the page is viewed. If you edit the document on the **Properties -> Template** tab and click the **Edit template properties** button, you can see the configuration of the page's template as shown below:

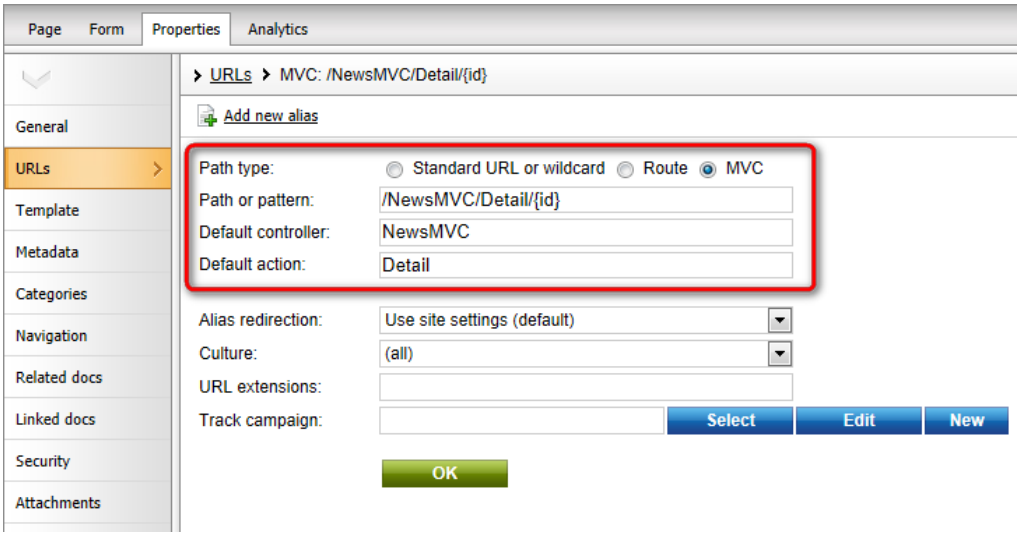
- **Default controller** - sets the name of the controller class containing the MVC action that should be performed when pages using the template are accessed (without the *Controller* part at the end). For example, if the class is called *NewsMVCController*, enter *NewsMVC*. The system first searches for the specified class in the *CMS.Controllers.<current site code name>* namespace. If it is not found there, the *CMS.Controllers.Global* namespace is searched.
- **Default action** - specifies the exact action defined within the controller class that is performed when pages based on this template are loaded.



Close the template editing dialog and open the document's **Properties -> URLs** tab. In this case, the page's URL is defined as an ASP.NET **Route**, but the standard alias path reflecting the document's position in the content tree can also be used for MVC pages as long as they use the appropriate template type.



The document has a registered document alias, which handles the */NewsMVC/Detail/{id}* pattern. If you edit (🖉) the alias, you can see that its **Default controller** is set to *NewsMVC* and the **Default action** is *Detail*. These will be used instead of the default template configuration when the page is accessed through the alias URL.



Further advanced options are also available for MVC URL patterns:

- **{controller}, {action}** - allows you to specify controllers and actions dynamically through URL parameters. For example, the */{controller}/{action}/{id}* pattern provides the same controller, action and page output as the example above when the */NewsMVC/Detail/My-First-News* URL is accessed. However, it could also be used to direct the users to pages generated by other controllers or actions

(the parameter parts of the URL may vary).

- **{name;value}** - provides a way to set the default value of the parameters in the URL path. For example, */{controller;NewsMVC}/{action;Detail}/{id;My-First-News}* has the same functionality as the previous option, but the system is able to automatically generate the default URL */NewsMVC/Detail/My-First-News* for the document if the parameters are not specified.
- **{*name;value*}** - hidden value, which is passed as a parameter to the action handler in the controller class, but is not incorporated into the URL itself. For example, */NewsMVC/List{*TopN;10*}* generates the */NewsMVC/List* pattern and provides the value of the *TopN* parameter during the processing of the request. Note that the TOP N functionality is not implemented in the sample controller and you need to handle it in the code if you want to try it. The URL of the page will still be the same as before (the parameter is hidden). This way, the pages can be parametrized by the developer even if they use the same controller and action.

Using MVC views in editing mode

To configure documents to show the MVC view in editing mode (i.e. when working with documents on the *Page* tab of CMS Desk), enable the **Use live URL for editing modes** setting in **Site Manager -> Settings -> Content -> Content management**.

You do not need to enable this setting if your documents are based on *MVC* type page templates, which automatically use the MVC view in editing mode.

MVC code files

The **NewsMVC** controller specified in the **Default controller** field above is physically located in *~\App_Code\Controllers\Global\NewsMVCController.cs* (or *~\Controllers\Global\NewsMVCController.cs.* if you installed Kentico CMS as a web application). The **List** action specified in the **Default action** field of the sample page's template is implemented in its code, along with the **Detail** action, which is used when the page is accessed through the document alias URL. The following is the full code of the controller class:

```
using System.Data;
using System.Web.Mvc;

using CMS.CMSHelper;
using CMS.DocumentEngine;
using CMS.URLRewritingEngine;

namespace CMS.Controllers.Global
{
    /// <summary>
    /// Sample controller for the news.
    /// </summary>
    public class NewsMVCController : Controller
    {
        /// <summary>
        /// Processes the detail action.
        /// </summary>
        public ActionResult Detail()
        {
```

```
            // Prepares the data for the view.
            TreeNode document = TreeHelper.GetDocument(CMSContext.CurrentSiteName,
"/News/" + RouteData.Values["id"], CMSContext.PreferredCultureCode, true,
"CMS.News", true);
            if (document != null)
            {
                document.SetValue("NewsTitle", document.GetValue("NewsTitle"));
                ViewData["Document"] = document;
            }
            else
            {
                // Return a page not found error if the document does not exist.
                URLRewriter.PageNotFound();
                return null;
            }

            return View();
        }

        /// <summary>
        /// Processes the list action.
        /// </summary>
        public ActionResult List()
        {
            // Prepares the data for the view.
            var documents = TreeHelper.GetDocuments(CMSContext.CurrentSiteName, "/
News/%", CMSContext.PreferredCultureCode, true, "CMS.News", null, "NewsReleaseDate
DESC", -1, true, 0);
            if (documents != null)
            {
                ViewData["NewsList"] = documents;
            }

            return View();
        }
    }
}
```

The views that display the pages when the respective actions are performed are located under ~*Views**Global**NewsMVC*\. The following code is the full code of the *Detail.aspx* file of the **Detail** view:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Detail.aspx.cs"
Inherits="Views_Global_NewsMVC_Detail" MasterPageFile="Root.master" %>

<asp:Content ID="cntMain" ContentPlaceHolderID="plcMain" runat="Server">
    <div class="innerContent">
        <h1>
            MVC Example</h1>
        <p>
            This is a sample for the page rendered by MVC. It displays the news
detail. The detail is provided by <strong>NewsMVC</strong> controller and
<strong>~/Views/Global/NewsMVC/Detail.aspx</strong> view and routed through MVC
URL pattern <strong>/NewsMVC/Detail/{id}</strong> added as document alias to the
same document as the main document. This way, there is no need to provide separate
documents for detail view.
        </p>
```

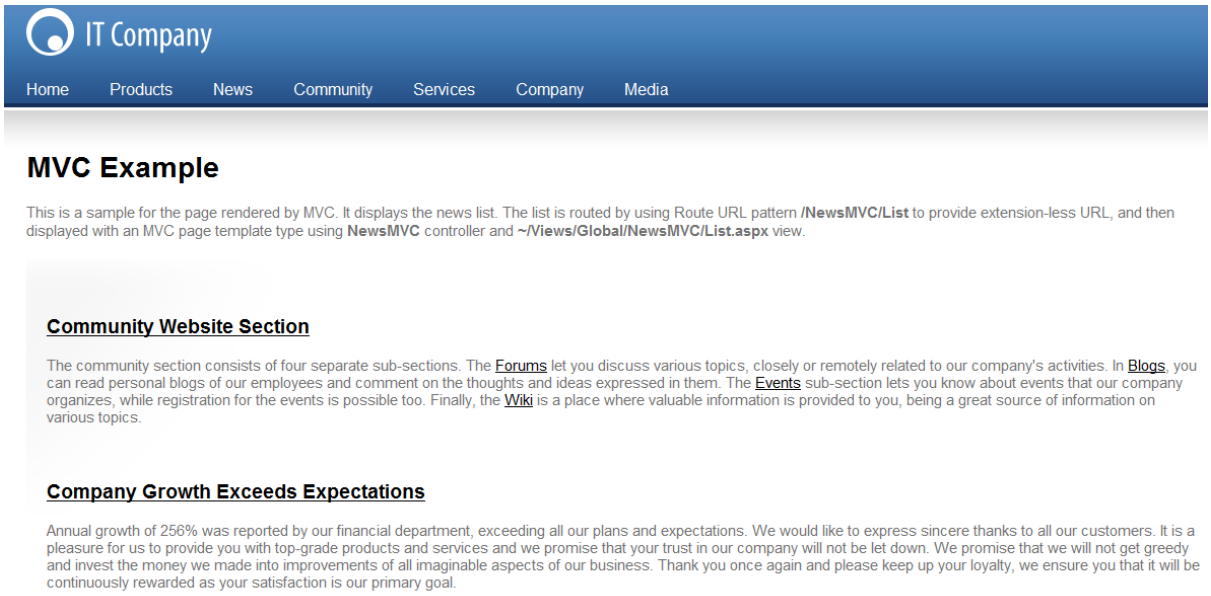
```
        <div class="LightGradientBox ">
            <h2>
                <%=Document.GetValue("NewsTitle")%></h2>
            <p>
                <%=Document.GetValue("NewsText")%></p>
            <p>
                <a href="<%=ResolveUrl("~/newsmvc/list/")%>">Back to the list of
news</a>
            </p>
        </div>
    </div>
    </div>
</asp:Content>
```

And the following code is the full code of its *Detail.aspx.cs* code behind file:

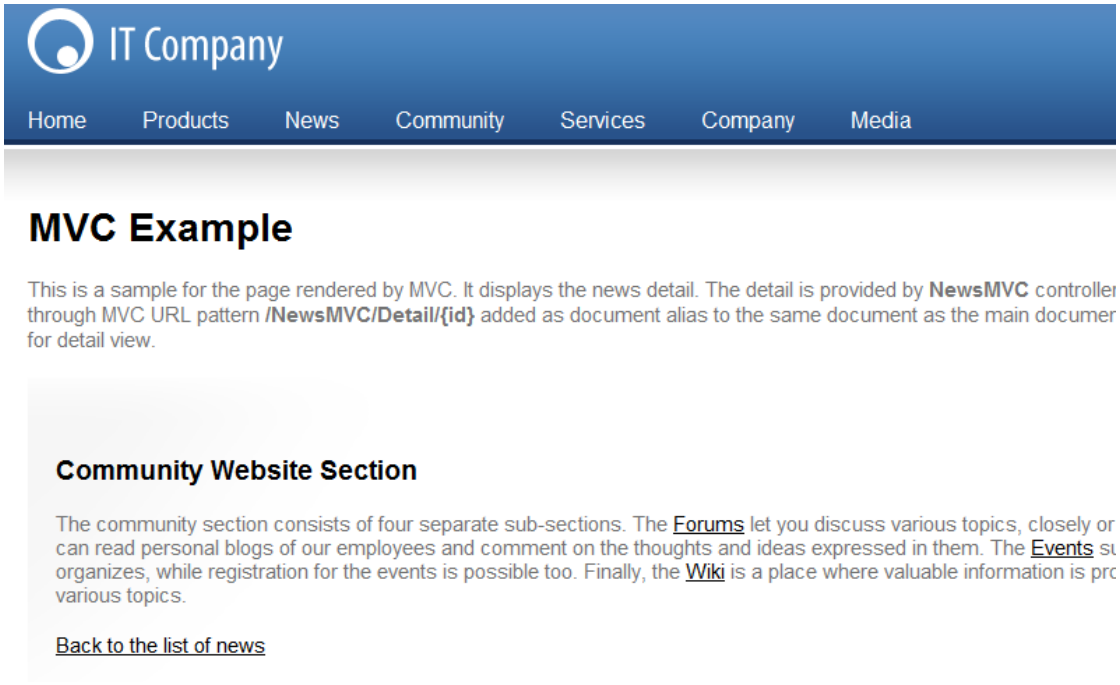
```
using System;
using System.Data;
using System.Web.Mvc;

using CMS.DocumentEngine;

/// <summary>
/// Sample view for the news.
/// </summary>
public partial class Views_Global_NewsMVC_Detail : ViewPage
{
    /// <summary>
    /// Returns the displayed document
    /// </summary>
    public TreeNode Document
    {
        get
        {
            return (TreeNode)ViewData["Document"];
        }
    }
}
```

How it works on the live site

When you navigate to the page on the live site, you can initially see the list mode view, which is specified as the default option for the page's template (the **List** action from the **NewsMVC** controller is executed). It displays a list of all news items from the **News** section, as can be seen in the screenshot below.



The titles of the news items contain links to the */NewsMVC/Detail/{id}* URL, where the *{id}* wildcard parameter is substituted by the alias of the corresponding news document. If you click it, the **Detail** action of the **NewsMVC** controller is executed, which displays a detail of the given news item. This is all handled by a single document, the different content is ensured through the special document alias URL, which triggers the detail view.



7.3 Caching and performance

7.3.1 Overview

The performance of your website depends on many aspects:

1. Hardware on which your website and database server are running.
2. Available system performance when you're sharing system resources with other applications (relevant in a shared hosting environment).
3. Amount of documents on your website.
4. Complexity of the website (number of nesting levels, number of web parts on a page, etc.)
5. Custom code added to the website.
6. Use of caching.
7. Other special circumstances, such as network connectivity between web server and SQL server, etc.

7.3.2 Caching options

Cache is a storage space which duplicates commonly used data to provide faster access to the data. Kentico CMS stores all its data in an SQL database, which requires the application to connect to the SQL server and transfer data over ineffective lines. This is why the application has a built-in cache, which copies the data that it uses the most to the application memory, and then retrieves the data from the memory instead of the database.

Cache types

Kentico CMS provides the following types of caching:

- Content cache - stores data used by web parts and controls that display structured content, such as repeaters and listings.
- Files cache - stores images. Supports both server and client-side caching
- Output cache (full page) - caches the full HTML output of pages.
- Page info cache - stores the basic information about documents, such as document's name and properties.
- Partial cache - caches the HTML output of specific web parts or controls.

Enabling page info cache

When a visitor requests a page, the system first queries the database for information about the page. Page info cache allows the system to send the queries only once and serve the results from the application's memory the next time. The basic page information includes:

- alias path
- ID and name
- metadata
- properties
- SKU information
- workflow information

For the full list of database columns that the page info cache stores, see View_CMS_Tree_Joined in the database reference.

To enable page info cache:

1. Open **Site Manager** and go to **Settings -> System -> Performance**.

2. Type a number of minutes into the **Cache page info (minutes)** setting. This is how long the cache will retain a particular page info item.

3. Save the settings.

When you request a page, its basic info will be stored in the cache. You can check that in the cache debugging interface. The cache keys begin with "pageinfo" or "pageinfobyurl". The information stays in the cache for the amount of time you specified in the **Cache page info** setting, or until someone modifies the related document.

Enabling content cache

Content cache helps the system to ease the load on SQL servers by storing structured data in the application's memory. You can set it up on two levels - either on a global level, which influences a whole site, or you can enable content cache for individual instances of web parts.

To enable content cache globally:

1. Open **Site Manager** and go to **Settings -> System -> Performance**.

2. Type a number of minutes into the **Cache content (minutes)** setting. This is how long the cache will retain cache items.

3. Save the settings.

With content cache enabled globally, the system stores structured data displayed by all web parts such as repeaters or listings. The data stays in the cache for the amount of time specified in the Cache content setting.

To enable content cache for particular web part instances:

1. Edit the web part in design mode.

2. Type a number of minutes into the **Cache minutes** property. This is how long the cache will store the web part's data.
   - The Cache minutes property is only available for web parts that load structured data from the database (i.e. data sources).

3. (Optional) Define Cache dependencies.

4. Click **OK** to save the web part's properties.

With content cache enabled, the system stores data displayed by the particular web part instance. The data stays in the cache for the amount of time specified in the Cache minutes property.

To disable content cache for a particular web part when content cache is globally enabled:

1. Edit the web part in design mode.

2. Type **0** (zero) into the **Cache minutes** property.

3. Click **OK** to save the web part's properties.

When you disable content cache for a particular web part, it doesn't store any of its structured data in the cache.

Enabling files cache

The system supports two types of file caching:

- **Server-side** - stores images in the application's memory so that the CMS doesn't have to access the database and file system every time an image is requested.
- **Client-side** - uses HTTP request headers to control file caching in client browsers and intermediate network caches.

To enable server-side file caching:

1. Open **Site Manager** and go to **Settings -> System -> Performance**.

2. Type a number of minutes into the **Cache images (minutes)** setting. This is how long the cache will retain images.

3. Save the settings.

With files cache enabled, the system stores images in the memory when they are requested. On subsequent requests, the system serves the images from the memory. Images stay in the cache for the amount of time specified in the **Cache images** setting.

Please note

Even when the files cache is off, the system still caches images for one minute to prevent the application from overloading in case of a DoS attack.

To enable client-side file caching:

1. Open **Site Manager** and go to **Settings -> System -> Performance**.

2. Type a number of minutes into the **Client cache (minutes)** setting. This determines the expiration time in the client cache.

3. Save the settings.

With client cache enabled, the system allows browsers to cache files for the number of minutes specified in the **Client cache** setting.

Enabling output cache

The most effective way to increase website performance is output cache. Output cache stores the full HTML source of pages. This way the application doesn't have to communicate with SQL servers and

process data every time a page is requested.

This option is not suitable for pages with web parts that need to be refreshed very often (e.g., the Random document web part) since you cannot disable caching for particular web parts. For such pages, it's recommended that you do not use output caching and use content caching instead.

If you wish to use output cache, you need to enable it globally in settings and then enable it for specific documents. Documents can inherit output cache settings from their parent document.

To allow output cache globally:

1. Open **Site Manager** and go to **Settings -> System -> Performance**.
2. Set **Enable output caching** to true.
3. Save the settings.

To enable output cache for documents:

1. Open the document's properties and select the **General** tab.
   - To apply output caching on all documents on your site, select the root document. Note that the underlying documents need to have the **Use output cache** property set to **Inherit**.
2. In the **Output cache** section, select *Yes* next to **Use output cache**.
3. Type a number of minutes into **Cache minutes**.
4. **Save** the document.

When a page is requested, the system stores its HTML source into the output cache. The data stays in the cache for the amount of time specified in **Cache minutes** or until someone modifies the page. However, if the page displays other documents (such as a news list) and you modify these documents, the page will stay in the cache and won't be updated until the cache expires.

Enabling partial cache

Partial cache stores the HTML output of web parts and controls as individual cache items. You can use partial cache when full page output caching is not feasible.

To allow partial caching globally:

1. Open **Site Manager** and go to **Settings -> System -> Performance**.
2. Set **Enable partial caching** to true.
3. Save the settings.

The website now allows partial caching for web parts and controls. By default, web parts do not use partial caching and you need to enable it for individual instances.

To enable partial caching for particular web part instances:

1. Edit the web part's properties.

2. Locate the Performance section and type a number of minutes into the **Partial cache minutes** property.

3. Specify cache dependencies.

4. Click **OK**.

When a page containing the web part is requested, the system stores the HTML output of the web part into the cache. The data stay in the cache for the amount of time specified in **Partial cache minutes** or until a dependent object is modified.

Cache dependencies

Cache dependencies allow the application to clear cached data based on relationships with other objects. A cache dependency is represented by a dummy cache key (a cache item that doesn't contain any data) that identifies an object or a set of objects. When an object that matches the dummy key is modified, the dummy key is "touched" and all cache items depending on the dummy key are removed from the cache.

For example, you have a page (with alias path */Products*) that contains a repeater. The repeater displays a list of products that reside under the Products page. Only the content cache is turned on. A visitor requests the Products page, the repeater fetches the products from the database and stores them in the cache. The system also creates the following dummy key: *node|corporatesite|/products|childnodes*. The dummy key ensures that once a child node of the Products node is modified, the depending cache items get cleared.

Entering custom cache dependencies

You can use cache dependencies to specify conditions that should be met in order to delete a web part's output from the memory. Each web part has its default dependencies which include all possible object changes that the content of the web part could depend on.

Note: The following steps describe how to specify dependencies for partial cache. Before you specify partial cache dependencies, enable partial caching for the given web part.

You can also define custom dependencies for output cache of a page using the Output cache dependencies web part.

To specify custom cache dependencies for partial cache:

1. Open the web part's properties.

2. Locate the Performance section.

3. Type one dummy key per line into the **Partial cache dependencies** field.

The following table shows which dummy cache keys get touched when an object gets changed. The table also lists examples of the dummy keys.

Object	Touched keys	Sample values

Document (TreeNode)	node|<sitename>|<aliaspath>|<culture> node|<sitename>|<aliaspath> nodeid|<nodeid> nodeid|<linkednodeid> nodes|<sitename>|<classname>|all **+ for every parent node**: node|<sitename>|<aliaspath>|childnodes	node|corporatesite|/home|en-us node|corporatesite|/home nodeid|12 nodeid|34 nodes|corporatesite|cms.menuitem|all node|sitename|/|childnodes
Any object (except documents)	<classname>|all <classname>|byid|<id> <classname>|byname|<codename> <classname>|byguid|<guid>	cms.user|all cms.user|byid|53 cms.user|byname|administrator cms.user|byguid|1ced44f3-f2fc- ...
Metafile	metafile|<guid>	metafile|1ced44f3-f2fc- ...
Document attachment	attachment|<guid>	attachment|1ced44f3-f2fc- ...
Forum attachment	forumattachment|<guid>	forumattachment|1ced44f3-f2fc- ...
Avatar	avatarfile|<guid>	avatarfile|1ced44f3-f2fc- ...
Media file	mediafile|<guid> mediafile|preview|<guid>	mediafile|1ced44f3-f2fc- ... mediafile|preview|1ced44f3-f2fc- ...
Page template	template|<id>	template|12
CacheHelper .ClearFullPageCache	fullpage	fullpage

Example 1: you have a web part displaying some information about users. Therefore, whenever some user gets their details modified, the web part's cache should be cleared. To ensure this, you need to enter **cms.user|all** into the field, which is the dummy key that would get touched whenever some user's details get changed.

Example 2: your web part is displaying information about one particular user - the administrator. Their user name is *administrator*, ID is *53* and GUID is something beginning with *1ced44f3-f2fc*. So if you want to have the cache cleared whenever this user's details are changed, you can use any of the following three keys that identify the user by the previously named properties:

- **cms.user|byid|53**
- **cms.user|byname|administrator**
- **cms.user|byguid|1ced44f3-f2fc-...**

Clearing the cache

Kentico CMS allows you to clear the contents of its cache.

To delete all items from the cache:

1. Go to **Site Manager -> Administration -> System**.

2. Click **Clear cache**.

The system empties the cache, including all dummy keys.

To delete a particular page from the output cache:

1. Select the document in CMS Desk content tree,

2. Navigate to **Properties -> General**.

3. Click **Clear output cache** in the Output cache section.

The system deletes the document's HTML output from the cache.

Debugging cache

The system debugging interface in **Site Manager -> Administration -> System** allows you to see the list of items that are in the system's cache and the list of dummy keys (cache dependencies). You can also view what cache operations the system performed when serving a particular page request.

Reference: Cache settings

Global

The following cache settings are available in **CMS Site Manager -> Settings -> System -> Performance**:

Server content caching	
Cache page info (minutes)	Sets the number of minutes for which the system caches page information. This option caches page properties and metadata. Kentico CMS retrieves page information many times during the processing of a single page, so always set this value to **at least 10 minutes!** When a page is modified, the system automatically clears the corresponding part of the page info cache, so the website will not display outdated information.
Cache content (minutes)	Sets the number of minutes for which all web parts/controls cache the content that they retrieve from the Kentico CMS database. You can override this value for specific web part instances by setting their **Cache minutes** property. Using 0 as the value disables content caching for the given web part instance. It is recommended to cache all possible content that is not modified often. The drawback of this option is that if content is modified, the changes appear on the live site only after the old version expires in the cache.
Use progressive caching	If checked, the system optimizes access to uncached data so that concurrent threads only use a single data access operation and share the results. This leads to better performance if the website is under a heavy load, without the drawback of not having the latest data available.

Server file caching	
Cache images (minutes)	Sets the number of minutes for which the system caches image files on the server. It is recommended to **always use image caching**. Kentico CMS automatically removes images from the cache if they are modified, so image caching cannot cause the website to display outdated content.
Maximum file size to cache	Specifies the maximum file size in kilobytes that is allowed to be cached.
Redirect files to disk	If checked, file requests are redirected to the corresponding physical file in the file system if possible.
File client caching	
Client cache (minutes)	Sets the number of minutes for which client browsers are allowed to cache files. If the value is 0, client caching for files is disabled.
Client cache must revalidate	If enabled, client browsers need to revalidate cached content by calling the server. If disabled, browsers do not perform server requests if the requested content is already cached.
Output caching	
Enable output caching	If checked, the system allows output caching. Output cache stores the full HTML source of pages. If unchecked, output caching is disabled on the whole website. To enable output caching for pages, configure the **Output cache** properties of individual documents in CMS Desk on the **Properties -> General** tab. Both the main website setting and document settings must be enabled to use output caching.
Cache output in file system (minutes)	Specifies the number of minutes for which the system stores output cache in the file system. This provides persistent cache storage in case of application restarts. If not set, only the standard caching mechanism (in memory) is used. If you enter a value, the system checks both types of cache. To enable output caching in the file system for pages, configure the **Allow file system cache** property of individual documents in CMS Desk on the **Properties -> General** tab.
Enable partial caching	If checked, the system allows partial caching of web parts and controls. The partial cache stores the HTML output of individual web part or control instances. If unchecked, partial caching is disabled on the whole website for all web parts. By default, web parts do not use partial caching and you need to enable it for individual instances using their **Partial cache minutes** property.

Documents

Individual documents have the following output cache settings in CMS Desk on the **Properties -> General** tab:

Output cache	
Use output cache	Indicates if the system caches the full HTML output of the page. Output caching can greatly improve the performance of the page, but is not suitable for pages with dynamic content. You can inherit the output cache settings from the parent document. **Important**: Output caching must also be allowed in the website's settings. Administrators can enable output caching in **Site Manager -> Settings -> System -> Performance**.
Cache minutes	Determines how long the system keeps the output code of the page in the cache (if output caching is enabled). The page's output cache is automatically cleared if someone modifies the page. You can manually remove the page from the output cache by clicking **Clear output cache**.
Allow file system cache	Indicates if the system stores the page's output cache on the server's file system. This provides persistent caching in case of application restarts. If disabled, the application only caches the page output in its memory. If enabled, the system checks both types of cache. You can inherit the setting from the parent document. The file system cache is stored for the number of minutes specified in the **Cache output in file system (minutes)** setting in Site Manager -> Settings -> System -> Performance.

Web parts

Instances of web parts provide the following properties for configuring content caching and partial output caching:

System settings	
Cache item name	Sets the name of the cache key used for the content of the web part. If not specified, this name is generated automatically based on the site, document path, **Web part control ID** and current user. A cache key can be shared between multiple web parts displaying the same content on different pages in order to avoid keeping redundant data in the memory.

Cache minutes	Sets the number of minutes for which the content of the web part remains cached before the latest version is reloaded from the database. If empty, the web part uses the value entered into the **Site Manager -> Settings -> System -> Performance -> Server content caching -> Cache content (minutes)** setting. If set to *0*, the web part does not use content caching.
Cache dependencies	Contains a list of cache keys on which the content cache of the web part depends. When the specified cache items change, the content cache of the web part is deleted. Each line may only contain a single item. If you check **Use default cache dependencies**, the web part uses dependencies that include all possible object changes that could affect the content of the given web part.
Performance	
Partial cache (minutes)	Sets the number of minutes for which system caches the output HTML code of the web part. Partial caching is similar to to full-page caching, but only for the code of the web part specifically. If left empty or set to 0, partial caching is not used for the web part. **Important**: Partial caching must also be allowed in the website's settings. Administrators can enable partial caching in **Site Manager -> Settings -> System -> Performance**.
Partial cache dependencies	Allows you to specify a list of cache keys on which the partial cache of the web part depends. When the specified cache items change, the system clears the partial cache of the web part. Each line may only contain a single item. If you check **Use default cache dependencies**, the web part uses dependencies that include all possible object changes that could affect the given web part.

7.3.3 Code minification and compression

An important factor that influences the performance of a website is the size of code resources, which are requested by the client browser when it renders a page. The most significant among these are CSS stylesheets and JavaScript source files, which may in some cases reach sizes that considerably impact page load time.

Kentico CMS has two types of built-in functionality that can be used to optimize the performance of requests for the above mentioned resources:

- **Code minification** - removes all unnecessary characters from the code that are not required by the browser to correctly process the resource. This includes white spaces, line break characters, comments, bookmarks etc. The behavior of a minified resource remains exactly the same as the original and it can immediately be handled by the browser, without any additional steps. Minified

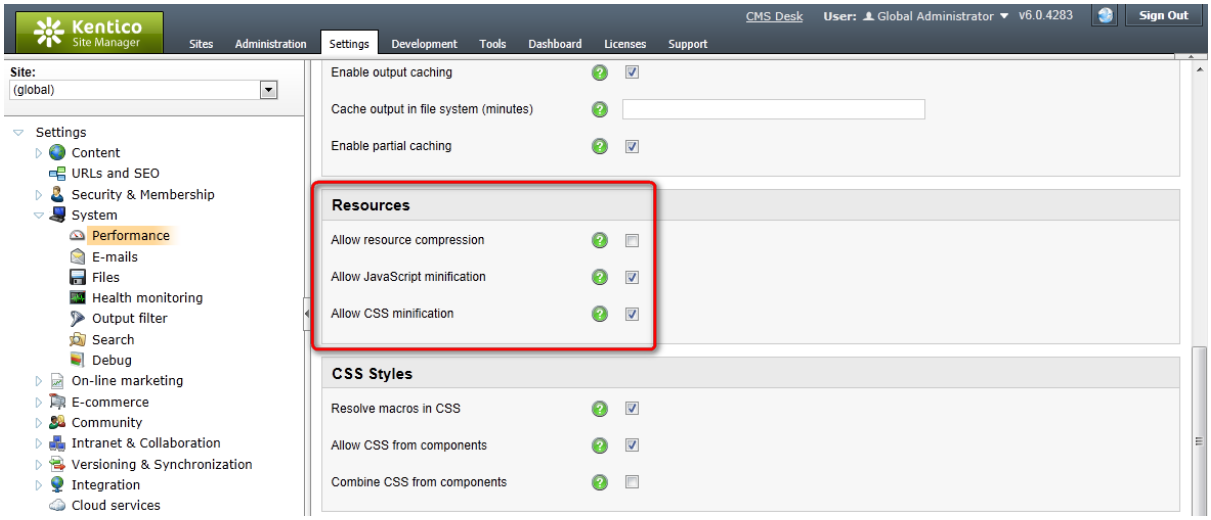
code is generally difficult to read (for humans) and is therefore unsuitable for debugging.

- **Resource compression** - encodes the data of the resource so that its size is reduced. In order for compression to be applied, minification must also be enabled for resources of the given type. When the client browser receives a compressed resource, it must first decompress the data. All browsers officially supported by Kentico CMS should be able to correctly decompress files. In cases where the client cannot process compressed data, the unmodified resource is sent instead.

Reducing the size of requested resources saves bandwidth and improves the response time of your website. Minification can decrease the size of a resource by approximately 20–40%, depending on the code of the given object. If compression is also used, resources can be reduced to roughly 30% of their original size.

Please note that these reductions are only applied to external resources stored individually in the file system or database. Inline code inserted into the HTML markup of pages is not affected.

Code minification and compression can be enabled or disabled globally for all sites in **Site Manager -> Settings -> System -> Performance**.



The minification or compression process slightly increases the server CPU load and may also cause a short time delay. To counter this issue, the server only performs minification/compression once per resource (when it is requested for the first time) and stores the results in the application's cache. When a given resource is requested, the appropriate transformed version is taken from the cache. Both compressed and uncompressed versions of resources are cached, so they are readily available even for clients without compression support.

Additionally, client-side browser caching can be used (enabled by default), which means that resources have to be reloaded from the server only if the cache expires or the content of the resource is outdated. Client caching of minifiable resources specifically can be enabled or disabled by setting the **CMSAlwaysCacheResources** key, which can be added into the **appSettings** section of your web.config file, for example:

```
<add key="CMSAlwaysCacheResources" value="false"/>
```

All of the mentioned functionality is automatically ensured by the **~/CMSPages/GetResource.ashx**

handler, which manages resource requests according to the specified settings. If minification is enabled, CSS and JavaScript requests generated by the system use this handler. If you need to manually write resource requests in your code, the following URL parameters can be used with the handler to specify which resource should be loaded:

- **stylesheetname** - used to request a CSS stylesheet from the database. The code name of the requested stylesheet must be entered as the value of the parameter.
  Link example:
  <link href="/KenticoCMS/CMSPages/GetResource.ashx?stylesheetname=CorporateSite" type="text/css" rel="stylesheet">

- **_transformations, _layouts, _templates, _devicelayouts, _webparts, _webpartlayouts, _containers** - used to request the internal stylesheets of the corresponding type of page component. The object ID values of the given components must be entered as the value of the parameter, separated by semicolons (if multiple component stylesheets are requested). See CSS stylesheets and design -> CSS for page components for additional information.
  Link examples:
  <link href="/KenticoCMS/CMSPages/GetResource.ashx?_containers=1;14" type="text/css" rel="stylesheet"/>
  <link href="/KenticoCMS/CMSPages/GetResource.ashx?_transformations=3511" type="text/css" rel="stylesheet">

- **stylesheetfile** - used to request static CSS resources from the file system. The relative path of the requested .css file must be entered into the parameter.
  Lnk example:
  <link href="/KenticoCMS/CMSPages/GetResource.ashx?stylesheetfile=/KenticoCMS/App_Themes/Design/OnSiteEdit.css" type="text/css" rel="stylesheet">

- **scriptfile** - used to request static JavaScript resources from the file system. The relative path of the requested .js file must be entered into the parameter.
  Link example:
  <script src="/KenticoCMS/CMSPages/GetResource.ashx?scriptfile=%7e%2fCMSScripts%2fmootools.js" type="text/javascript"></script>

Requests with minification disabled

If minification is disabled, the system generates requests with a direct URL (for JavaScript files) or using the **~/CMSPages/GetCSS.aspx** system page (for CSS stylesheets).

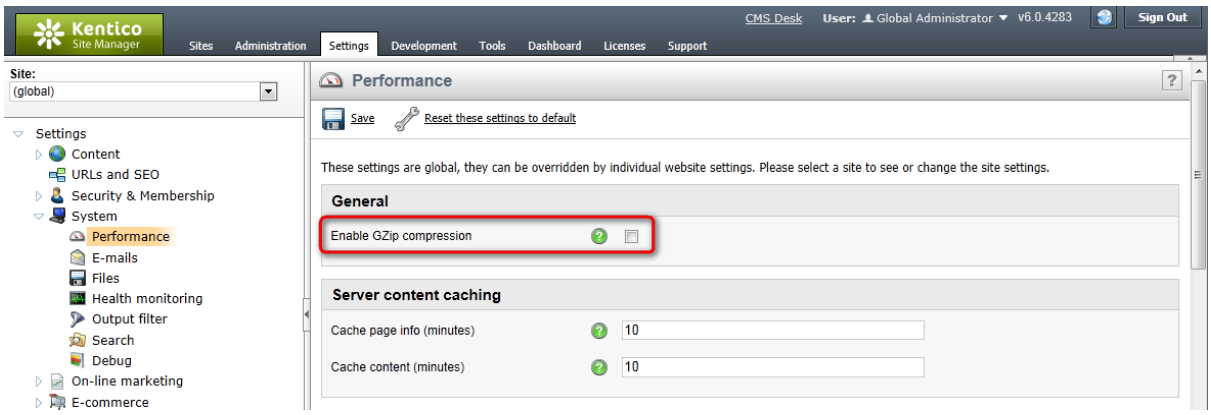
Requests with this URL format are always supported, but they do not perform minification or compression of resources.

GZip compression of page output

It is also possible to enable GZip compression of the output code of all pages rendered by Kentico CMS. This can be done either by adding the following key into the **appSettings** section of your web.config file:

```
<add key="CMSAllowGZip" value="true" />
```

or by enabling the **Enable GZip compression** option in **Site Manager -> Settings -> System -> Performance**.



7.3.4 File management and performance

Files can be stored in the file system (faster) or in database. If you're experiencing problems with slow image viewing, please try to configure the following values in the **Site Manager -> Settings -> System -> Files** section:

- **Store files in file system**: yes
- **Generate thumbnails**: yes

Enabling the **Redirect files to disk** setting in the **Settings -> System -> Performance** section could also be helpful.

You can find more details on file management in the Where the files are stored topic.

7.3.5 Troubleshooting performance issues

If you encounter performance issues, please try to follow these steps to make sure that your system is optimized for best performance:

1. **Make sure you're using the latest version of Kentico CMS**
   We improve the performance with every new release. Especially the 2.0 and 2.1 versions didn't provide a very good performance. You can find the version number in the lower right corner of Kentico CMS Desk -> Content dialog or on the logon screen of the administration interface.

2. **Make sure caching is configured on your website.**
   Go to **Site Manager -> Settings**, choose the appropriate website in the **Sites** drop-down list and choose the **System -> Performance** category. Make sure the values are set like these:

   - **Cache content** >= 0
   - **Cache images** > 0, at least 10 minutes recommended (images are automatically removed from the cache and reloaded if you modify them)
   - **Always set Cache page info > 0**, at least 10 minutes recommended (page data is automatically

removed from the cache and reloaded if you modify it)

3. **Try configuring full-page caching**
   Full-page caching is the most powerful caching option. Once the page is cached in the memory during the first view, it's displayed without contacting SQL server and running the page code. You can configure full-page caching in **CMS Desk -> Content -> Properties -> General -> Cache**.

   The page is automatically removed from the memory and reloaded when you modify it's content. Please note that the performance improvement will be visible only during the second load of the page.

4. **Turning off output filters**
   In special cases, the website may be slowed down by output filters. Go to **Site Manager -> Settings**, choose the appropriate website in the **Sites** drop-down list and choose the **System -> Output filter** category. You can try to temporarily turn off all output filters by setting the following values to / (slash), which will disable the given filter for the whole website:

   - **Excluded output form filter URLs**
   - **Excluded resolve filter URLs**
   - **Excluded XHTML filter URL**

   If it helps, please contact us and we will help you find a workaround (if possible).

5. **Configure file caching**
   Displaying files stored in Kentico CMS repository may be require lots of CPU time and may harm the overall website performance. Please try to configure these values in the **Site Manager -> Settings -> System -> Files** section (your web application needs to have Modify permissions on the disk):

   - Generate thumbnails: yes
   - Redirect files to disk: yes
   - Store files in file system: yes

6. **Check your code**
   If you integrated any custom .NET code into the website, please make sure it works properly. Please be sure to avoid too many database operations and be sure to close the database connections properly. Try to comment out your code and see if it improves the performance.

7. **Check your hardware**
   It's recommended that your system has at least 1 GB RAM and Pentium 4 or Pentium Core 2 Duo (or similar) processor.

8. **Check the other applications/websites running on the same server**
   Whether you use your own server or shared hosting, make sure that the other applications do not take all the server performance. It's highly recommended that you run Kentico CMS in a **separate application pool** on Windows Server 2003.

9. **Setting application pool Idle time-out in IIS**
   If your website is accessed not very frequently (less that every 20 minutes by default), users may experience long delays on first access to the site. To prevent this, you need to set the **Idle time-out (minutes)** property of the application pool to a higher value. This property can be accessed through:

   IIS 6: open **IIS Manager** -> select **<machine>/Application Pools** -> right click the application pool -> select Properties -> switch to the **Performance** tab -> set the **Shutdown worker process after being idle for (time in minutes)** to a higher value or disable the option completely by unchecking

the box

IIS 7: open **IIS Manager** -> select **<machine>/Application Pools** -> right click the application pool -> select **Advanced Settings** -> the property is located in the **Process Model** section

If the above mentioned settings don't help, please send us an exported copy of your website and we will try to analyze it.

7.3.6 Caching in custom code

In the following code extract, you can see how a *CachedSection* object can be used to cache data in your custom code. It checks if the item is in the cache (it also handles if the item should be cached or not based on the given cache minutes and boolean result of a specified condition). If it is in the cache, it returns the item, if not, it reports that the data should be loaded, you handle the loading and put the result back to the object. Then, the object stores the data to the cache automatically.

```
using CMS.SiteProvider;
using CMS.GlobalHelper;

private void CachingExample()
{
    DataSet data = null;

    // Cache the data for 10 minutes with key "mykey"
    using (CachedSection<DataSet> cs = new CachedSection<DataSet>(ref data, 10,
true, null, "mykey"))
    {
        if (cs.LoadData)
        {
            data = UserInfoProvider.GetAllUsers(); // Get data from database
            cs.CacheDependency = CacheHelper.GetCacheDependency("somekey"); //
Cache dependency
            cs.Data = data; // Save data to cache
        }
    }
}
```

7.4 Cookies

7.4.1 Overview

The cookie law consent functionality was implemented because of the Cookie law (you can find more information at http://www.ico.gov.uk/), which is in effect in some European countries and which requires website developers to ask site visitors for consent with using cookies.

We divided cookies that Kentico CMS uses into several cookie levels according to their importance in the system. We also created three main levels designed for you to be able to comply with the law regulations quickly and easily.

To display a text message and buttons asking for site visitors' consent with storing cookies on their computers, you can use two provided web parts. With the preconfigured **Simple cookie law consent**

web part, you can introduce the consent on your website quickly and easily.

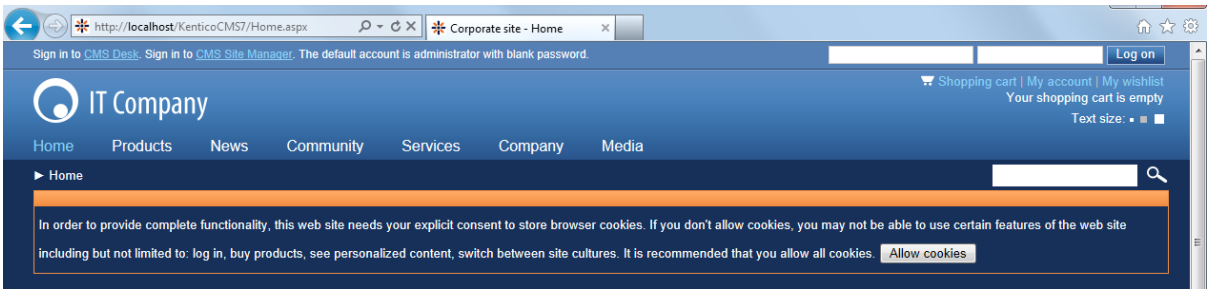
For the information to be complete, we provide a list of all cookies that Kentico CMS may currently use with assigned levels and a brief description.

If you need details about cookies API, you can find more information in the Cookie Law Support in Kentico CMS 6 blog post.

How to quickly include a cookie consent in your website

1. Navigate to a page and zone, where you want to place the message of the consent.

2. Choose the **Simple cookie law consent** web part under the **General -> Cookies** category in the web part catalog.
   - You are not required to configure any settings. It can be useful though, to visually separate the text from the rest of the content using a web part container (e.g, **Orange box**).



7.4.2 Cookie levels

We divide cookies into several levels to differentiate their purpose and to provide the option of setting a certain cookie level for the user. We provide more levels than you probably need in order to offer more options for configuration and future updates. These are the default levels available in Cookie helper API (CookieLevel class):

- **None (-1000)** - Absolutely no cookies are allowed, including the cookie indicating the cookie level that the user chose. This makes sense only in the case when you want to disable absolutely every cookie by default.
- **System (-100)** - This level allows the session cookie (due to security on postback), and CookieLevel cookie which remembers the cookie level that the user chose. In terms of the end user and the Cookie law, this means "Cookies are not allowed"
- **Essential (0)** - Cookies which are required for all website functionality needed by site visitors. This includes authentication, shopping cart, votes in polls etc. From the visitor's perspective, this means "Allow only cookies that I may need, but do not track me".
- **Editor (100)** - Cookies which are needed for the admin UI to work properly. This mainly includes the view mode, remembering tabs and sliders and a few others.
- **Visitor (200)** - All cookies which aren't assigned to any explicit level (which also includes your custom cookies if you use any) are considered to be cookies identifying the visitor, i.e. user tracking cookies, which are not really needed, but are useful for the site owner if using EMS.
- **All (1000)** - This is a system level, not a particular cookie level. This level allows all cookies, no matter what their level is. From the site visitor's perspective this means "Allow all cookies now and in the future".

The numbers associated with each cookie level are integer constants, which allow you to further customize the levels or make custom levels in case you need it. Also for this reason, the levels "None" and "All" have an absolute value of 1000 to provide enough space for potential future updates.

For simplicity, we created 3 main levels, which represent choices to be offered to the user when asking for the user's consent with the cookies:

- **No cookies** (level System) - Enables the minimum cookies required for the website to work, and remembers whether the user allows cookies or not.
- **Only essential cookies** (level Essential) - Enables all cookies required for website functionality, but no tracking cookies.
- **All cookies** (level All) - Enables all cookies used in the website.

These three levels are also used in the provided cookie web parts.

Administrator and cookies

If a user logs in to admin UI, all cookies are automatically enabled for the user to provide full editing functionality. We expect that login to admin UI identifies the user as staff; therefore there is no need for cookie consent in this case.

7.4.3 Available web parts

To offer the user an option to choose the desired level of allowed cookies, you can use these two web parts. Technically, it is a single web part implemented in ~/CMSWebParts/General/CookieLaw.ascx and the "simple" one is just a preconfigured version of the same web part. These web parts are not visually designed, and you need to update your CSS style to get the desired look and feel.

You can find both of these web parts under the **General -> Cookies** category in the web part catalog.

Cookie law consent

This web part can be used to ask a user for a consent with using cookies. It offers three levels to choose from (as described in Cookie levels topic) - **No cookies**, **Only essential cookies** and **All cookies**. On each level, actions to switch to the remaining levels are offered. You may customize the text messages, available actions as well as button texts.

The option **Compare current cookie level to** is a basis for other settings in the web part. It defines a level to which the user's current cookie level is compared. Then, according to this setting, three states with their own settings are defined: One state is when the current level is below the compared level, the second when the level exactly matches it, and the third when the current level is above the compared level. This should give you enough options to configure the web part to suit your needs.

Note that changing the cookie level works only in the live site mode, as you do not need to clear your cookies in the design mode. For designing purposes, there is another property which simulates the user level in the Edit and Preview mode.

The default cookie level is currently not set in this web part so it keeps the level "All" as the starting level, but you may change this setting too.

Simple cookie law consent

This is a preconfigured version of the previous web part. It allows you to easily implement a simple cookie law consent. The preset behavior is: Disable cookies by default, request allowing, and once allowed, hide the web part.

Live site

Once you place the web part into your website, you might not see it's message after switching to the Live site. The reason is that administrators have cookies enabled by default and the cookies are already stored on the computer you are working on. To check whether your configuration of the cookie web part is correct, log out of CMS Desk and clear all cookies in your browser. Or try the website in a different browser or computer.

7.4.4 Reference: Kentico CMS cookies

In the following table, you can find a list of cookies used in the system with their assigned levels and a brief description.

Cookie name	Level	Description
CMSCookieLevel	**System**	Specifies which cookies are enabled by the visitor.
ASP.NET_SessionId	System	Keeps the user session ID for security reasons.
CMSPreferredCulture	**Essential**	Stores the visitor's preferred content culture.
CMSMobileRedirected	Essential	Indicates if the visitor has been redirected to the mobile version of the website by the Mobile device redirection web part.
CMSCurrentTheme	Essential	Stores the name of the current visual theme to provide proper design to the dialog windows.
Webauthtoken	Essential	Live ID authentication cookie.
CMSForumPostAnswer	Essential	Keeps a list of Question-Answer forum posts in which the user user voted for an answer to prevent repeated votes.
CMSVotedPolls	Essential	Keeps a list of polls where the user voted to prevent repeated votes.
CMSRatedDocuments	Essential	Keeps a list of documents that the user rated to prevent repeated votes.
CMSShowDesktopVersion	Essential	Indicates that the visitor has switched to the desktop (default) version of the website from a specific device profile.
.ASPXFORMSAUTH	Essential	Stores the user's encrypted authentication ticket when using forms authentication.
FormState	**Editor**	Form state flag to allow restoration of failed requests.

CMSPreferredUICulture	Editor	Stores the preferred UI culture of the user.
CMSViewMode	Editor	Stores the user's current view mode (Edit, Preview, Design, etc.).
CMSUserWords	Editor	User's custom word dictionary kept by the spell checker.
DisplayContentInDesignMode	Editor	Remembers the user's setting of the *Display web part content* checkbox on the Design tab in CMS Desk.
CMSMacroDesignerTab	Editor	Remembers the last active tab of the macro designer.
CMSSiteFilter	Editor	Remembers the state of the site selector within the UI.
CMSSplitMode	Editor	Remembers the state of the language version split-view mode when editing multilingual websites.
CMSAnalyticsTab	Editor	Remembers the last active tab of the Analytics section in CMS Desk -> Content.
CMSPropertyTab	Editor	Remembers the last active tab of the Properties section in CMS Desk -> Content.
CMSPreviewState	Editor	Stores the user's latest page design preview preferences.
CMSViewTab	Editor	Remembers the last active tab of the View section in CMS Desk -> Content.
CMSValidationTab	Editor	Remembers the last active tab of the Validation section in CMS Desk -> Content.
CMSProductTab	Editor	Remembers the last active tab of the Product section in CMS Desk -> Content.
CMSEdVariantSliderPositions<templateid>	Editor	Remembers the position of the variant slider when defining variants for MVT or Content personalization.
CMSOnSiteButtonMinimized	Editor	Remembers if the user minimized the button used to open the website in On-site editing mode.
CMSWebPartToolbarCategory	Editor	Stores the selected web part category on the web part toolbar.
CMSWebPartToolbarMinimized	Editor	Remembers if the user minimized the web part toolbar on the Design tab.
CMSCurrentDeviceProfileName	Editor	Stores the selected device profile when editing documents.
CMSDeviceProfileRotate	Editor	Stores the rotation preferences applied in Preview mode when using device profiles.
CMSUniGraph	Editor	Stores the user's *Snap to grid* preference in the advanced workflow and marketing automation designer.
CMSSessionToken	Editor	Stores the token used by the web service that provides the advanced workflow and marketing automation designer.
VisitorStatus	**Visitor**	Indicates if the visitor is new or returning. Used for tracking the visitors statistic by the Web analytics module.
Campaign	Visitor	Stores the web analytics Campaign assigned to the

		visitor.
UrlReferrer	Visitor	Stores the URL referrer from which the user arrives on the website.
CurrentContact	Visitor	Stores the GUID of the contact related to the current site visitor. Used to track activities on the website.
CMSAB<ABtestname>	Visitor	Stores the name of the page variant assigned to the visitor by an A/B test. Used to track conversions for the test and maintain consistent page content for the visitor.
CMSMVT<mvtestname>	Visitor	Stores the combination of variants assigned to the visitor by an MVT test. Used to track conversions for the test and maintain consistent page content for the visitor.
CMSNoTestMVT<templateid>	Visitor	Stores the currently selected MVT combination for editors in the administration interface.
CMSShoppingCart	Visitor	Stores a reference to the user's active shopping cart.
CMSBodyClass	Visitor	Body element class to provide accessibility standards.
CMSEd<GUID>Current	Visitor	Stores the current step of Wizard layout web parts.
ChatLoggedInToken	Visitor	Stores the login state for the Chat module (indicates if the user is in the online state).
ChatSupportLoggedInToken	Visitor	Indicates if the user is logged in to the support chat.
<Window name>_<Group ID>_roomID	Visitor	Stores bindings between groups of chat web parts and chat rooms (for a specific window or tab).
chat_autoinitchat_displayed_<GUID>	Visitor	Remembers if the *Automatically initiated chat* web part was shown to the user (prevents multiple chat initiation messages).
chat_kick_roomid_<room ID>	Visitor	Indicates that the user was kicked from the specified chat room (and is not allowed to return).

7.5 CSS stylesheets and design

7.5.1 Overview

CSS stylesheets allow you to modify the appearance and design of your website. You can use standard CSS styles with Kentico CMS just like you normally would during website development.

Every website has a **default CSS stylesheet**. To assign it, edit () a site in **Site Manager -> Sites** and select an option from the **Site CSS stylesheet** field.

Individual pages can either use the default website stylesheet or override it with their own stylesheet. The stylesheet used by a specific page can be configured in **CMS Desk -> Content -> Properties -> General**, through the **CSS stylesheet** selector. The stylesheet assigned to a page is automatically requested when the given page is displayed.

CSS styles may also be defined for various types of components that can be placed on pages. Please see the CSS for page components topic for further information.

You can check individual pages to detect if the styles they use are valid against the CSS 2.1 standard. For more information, see Content management -> Validators -> CSS validator.

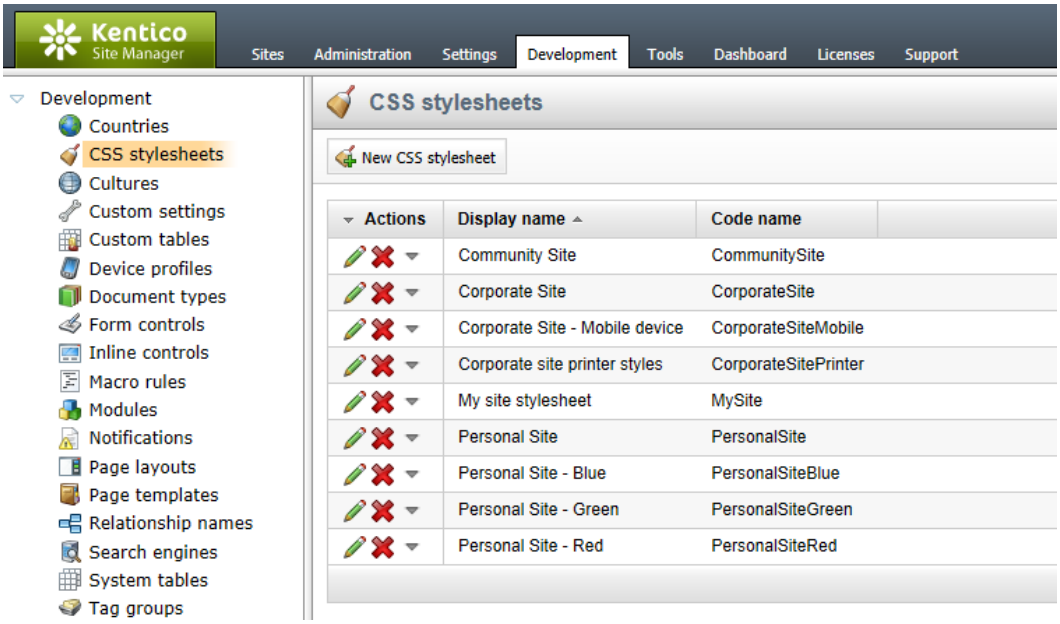
Tip

You can create stylesheets using the LESS dynamic language. LESS syntax allows you to write stylesheets that are shorter and easier to maintain.

Download the free LESS stylesheets module from the Kentico Marketplace.

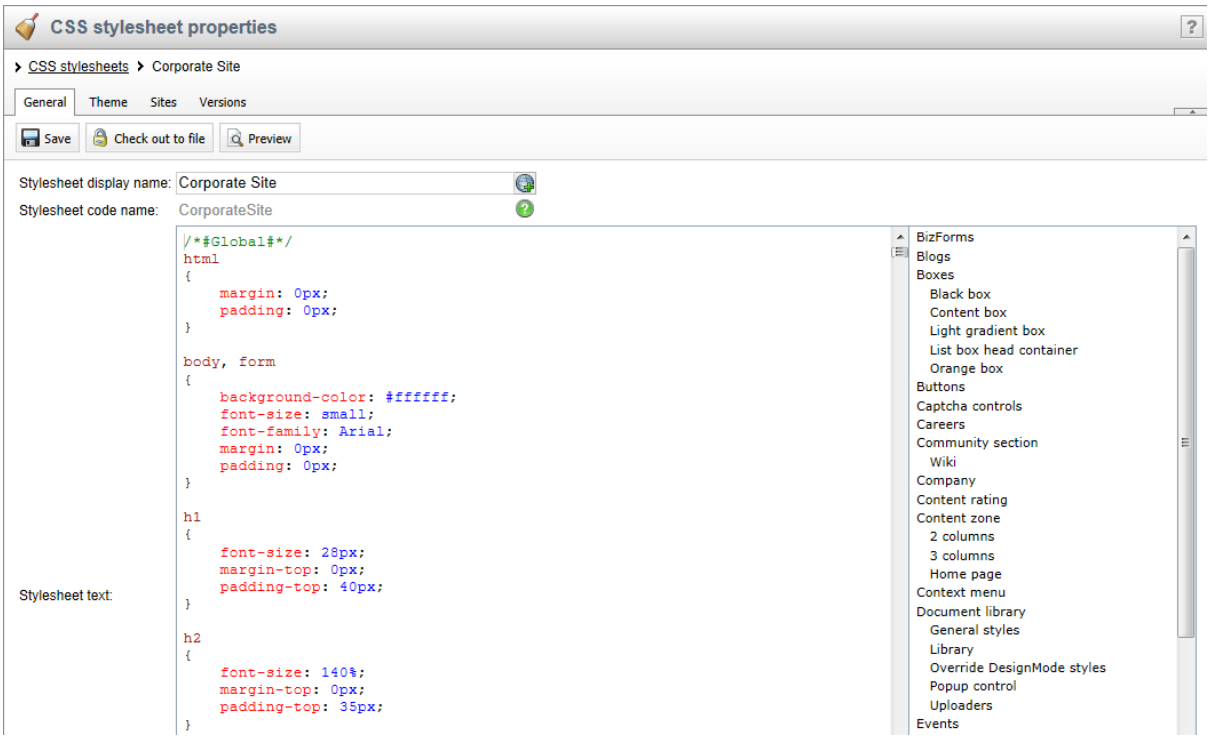
Creating and managing stylesheets

You can create and manage CSS stylesheets in **Site Manager -> Development -> CSS stylesheets**.



When editing ( ) a stylesheet, the following property fields are available on the **General** tab:

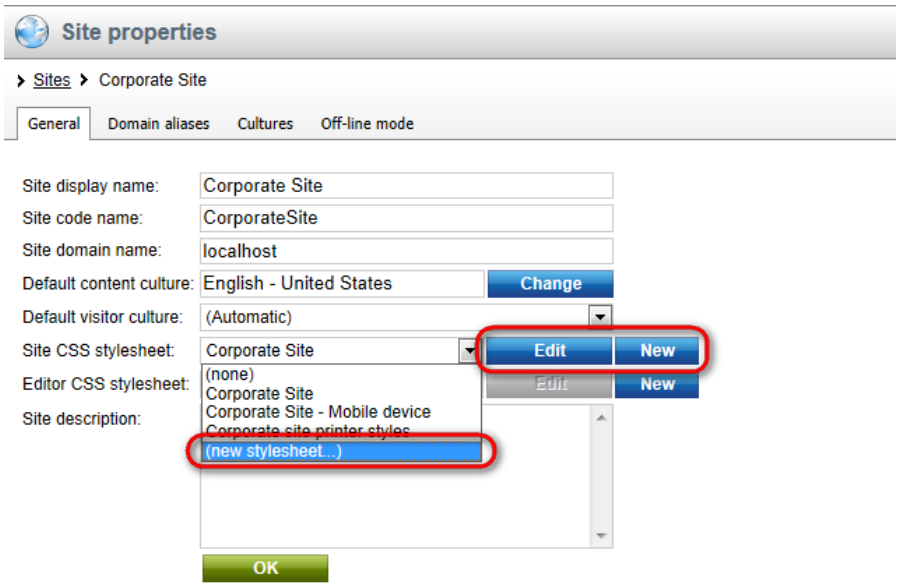
- **Stylesheet display name** - name of the stylesheet displayed in the administration interface.
- **Stylesheet code name** - a unique name used to identify the stylesheet in code and URLs.
- **Stylesheet text** - here you can enter the definitions of the classes that should be included in the stylesheet (using standard CSS code). This field uses the code editor to make working with the code more user-friendly. You can click the *Preview* button in the tab's header to edit the stylesheet's code side-by-side with a preview of how it affects the design of the website's pages. Please see the Design preview topic for additional details about the preview mode.



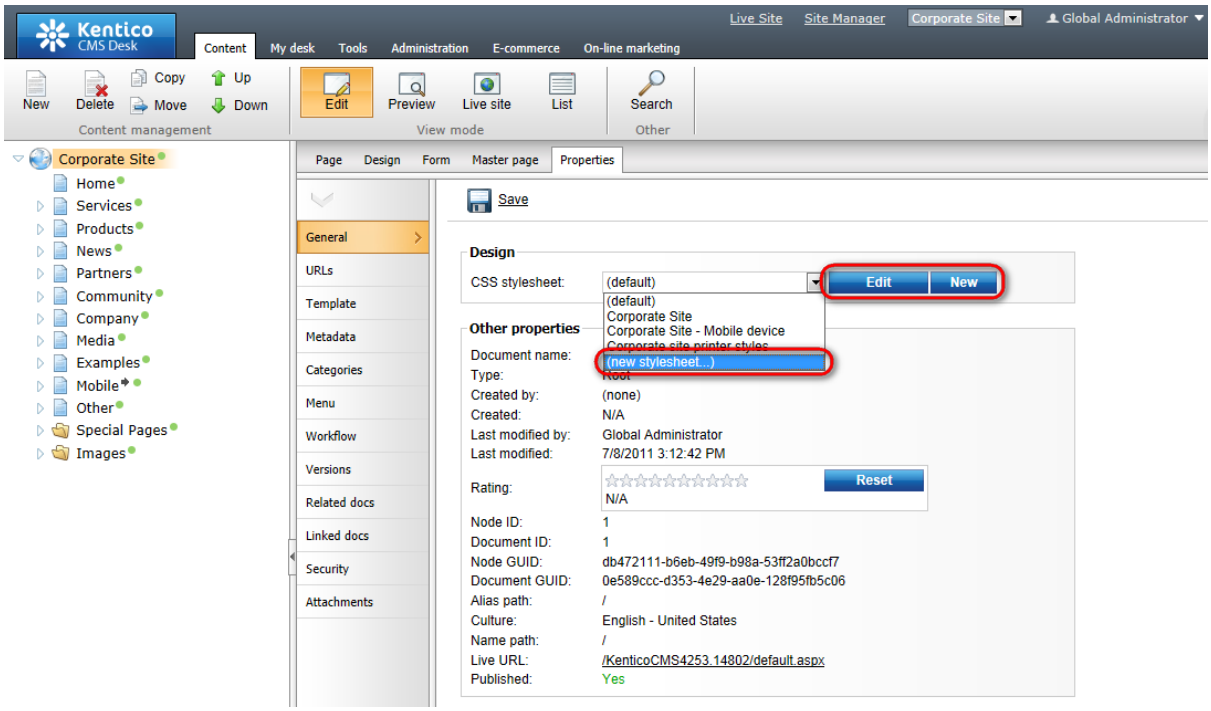
The **Theme** tab allows the management of files used by the edited stylesheet, such as external stylesheets, images or skin files. More information about associated files can be found in the App themes topic.

A stylesheet must also be assigned to particular websites on the **Sites** tab.

Stylesheets can also be created or edited directly in **Site Manager -> Sites -> edit (🖉) a site**. New stylesheets can be created by clicking the **New** button or by selecting **(new stylesheet...)** from the **Site CSS stylesheet** drop-down list. If an existing stylesheet is selected from this drop-down list, you can edit its code by clicking the **Edit** button.



The same options as described above are also available in **CMS Desk -> Edit -> Properties -> General**. Here again, you can create a new stylesheet using the **New** button or by selecting **(new stylesheet...)** from the drop-down list, or edit the code of a selected stylesheet by clicking the **Edit** button.



Stylesheet URL

You can view the code of any stylesheet using a URL in the following format:

<domain>/CMSPages/GetCSS.aspx?stylesheetname=<stylesheet code name>

The **GetCSS.aspx** system page retrieves unmodified, user-friendly stylesheet code even if Minification of stylesheet resources is enabled.

Browser and language-specific styles

Pages automatically have CSS classes assigned to their <*body*> element according to the characteristics of their language (its text direction and specific culture) and depending on the browser in which they are currently viewed. For example:

```
<body class="LTR IE IE9 ENUS">
```

As you can see, four types of classes are added:

- **Text direction** - the *LTR* class is assigned for left-to-right languages and *RTL* for right-to-left.
- **Browser type** - this class is added according to the browser in which the page is currently opened. The following classes are used:

Browser	Class name(s)
Internet Explorer	IE
Firefox	Gecko
Google Chrome	Chrome, Safari
Safari	Safari
Opera	Opera

- **Browser version** - the class name is the same as for the browser type, but with the number of the browser's major version appended, e.g. *IE9*, *Gecko5* etc.
- **Culture** - the name of the class is added based on the culture code of the page's content (without the hyphen), for example *ENUS* for pages using the *en-US* culture.

This feature allows you to style page elements differently according to the browsing environment of the current visitor. You can define styles for any combination of the classes mentioned above.

For example, you can add the following into a website's stylesheet:

[CSS]

```
.IE8 .MyFont
{
  font-size: 20px;
}

.Opera .MyFont
{
```

```
  font-size: 18px;
}
```

Now elements styled using the *MyFont* class will have a different font size when viewed in the Internet Explorer 8 or Opera (all versions) browsers.

Using CSS blocks for easier navigation in CSS code

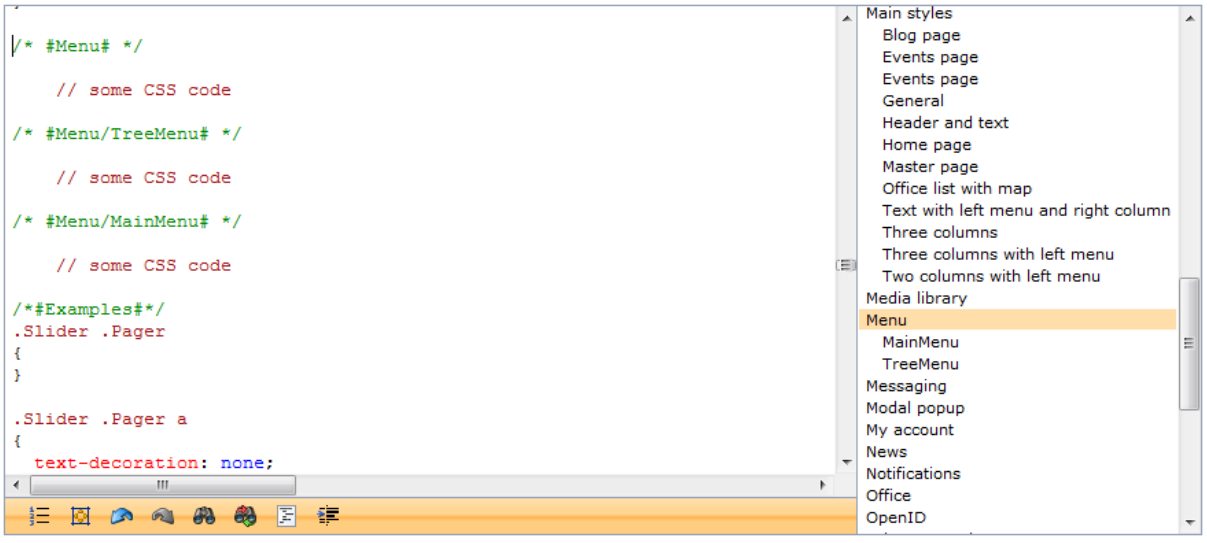
You can use comments in format */* #BLOCKNAME# */* to make navigation in the CSS code of long stylesheets easier. This creates a code block within the stylesheet and adds a bookmark to the list on the right side of the code editor. Blocks may be organized into a hierarchy of sub-blocks by separating the names of individual levels using a slash, such as for example */* #BLOCKNAME/SUBBLOCK# */*.

Example:

[CSS]

```
/* #Menu# */

// some CSS code

/* #Menu/TreeMenu# */

// some CSS code

/* #Menu/MainMenu# */

// some CSS code
```

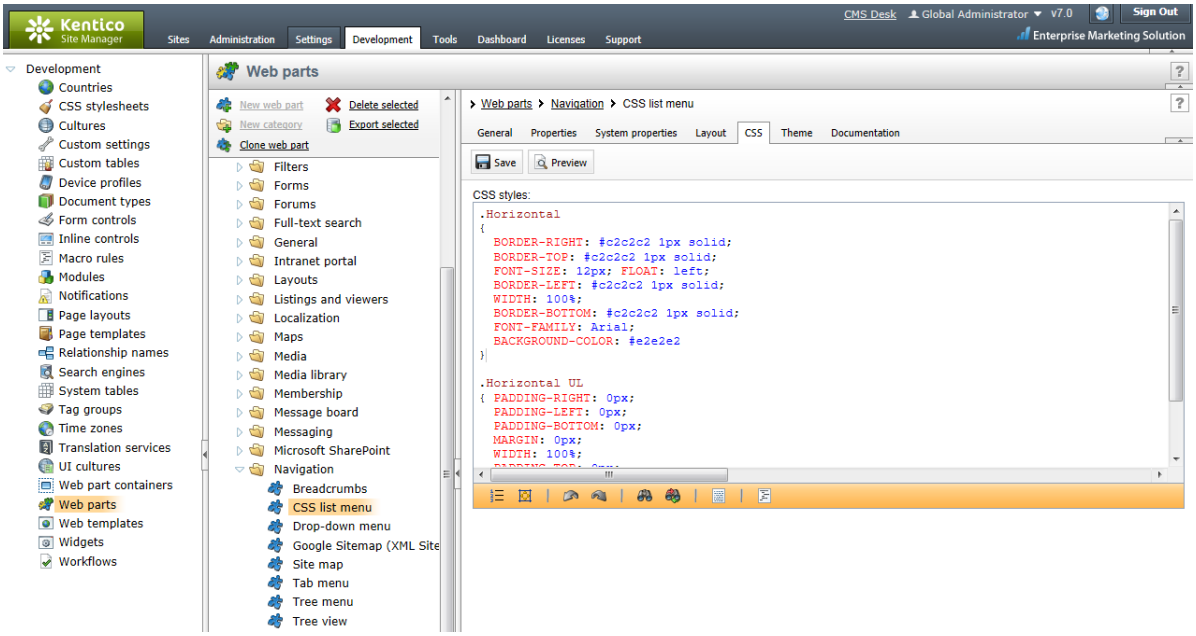
The outlined structure will look like this:



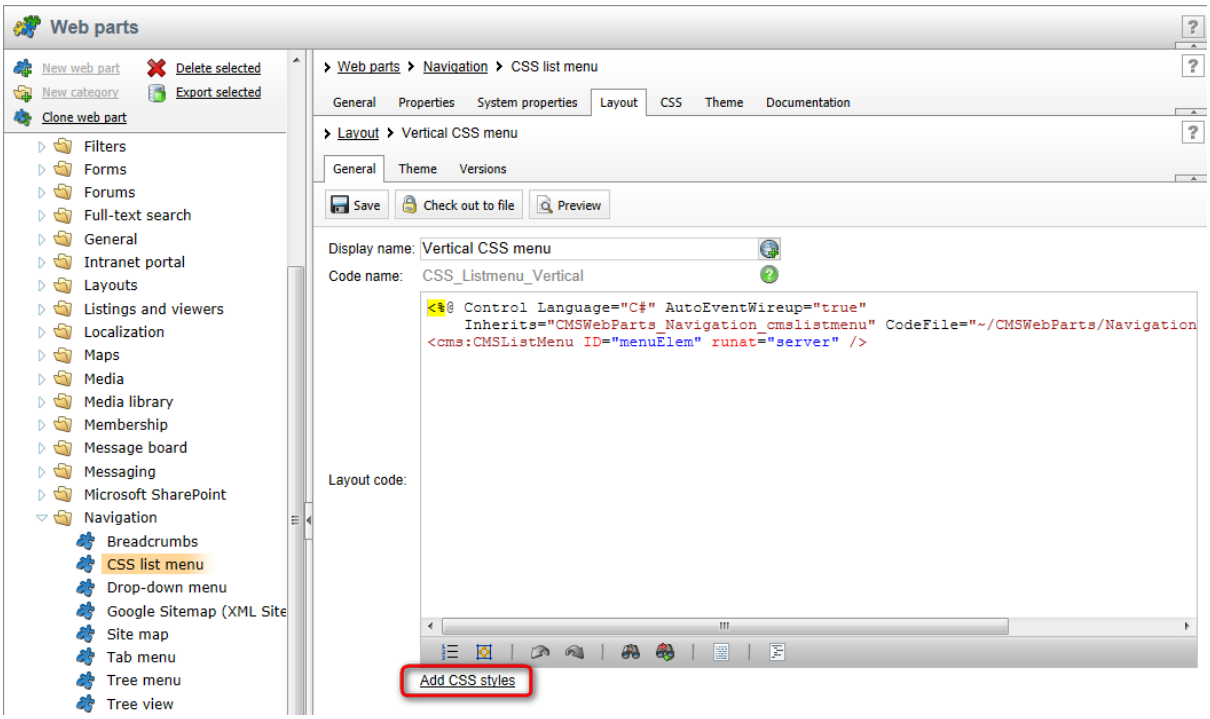
7.5.2 CSS for page components

In addition to the main stylesheet objects that can be assigned to sites or individual pages, it is possible to specify CSS styles directly for various types of components that make up the content of pages. This includes the following types of Kentico CMS objects:

- **Web parts** - CSS styles can be added to a Web part by editing it in *Site Manager -> Development -> Web parts* on its *CSS* tab. These styles will then be requested on pages containing the given web part.



- **Web part layouts** - to define styles for a specific Web part layout, navigate to the *Layout* tab of the given web part's configuration interface, edit ( ) the layout and click the Add CSS styles link below the layout code editor.



The *CSS styles* field will be displayed, where you can add any required CSS classes. The styles entered here will only be requested on pages that contain a web part using the specific layout.

- **Web part containers** - to manage the styles of Web part containers, go to *Site Manager -> Development -> Web part containers*, edit the given container and click the Add CSS styles link. The entered styles will be loaded by pages where the container is used.

- **Transformations** - when editing the Transformation of a document type or custom table, you can define its CSS styles by clicking Add CSS styles below the transformation code editor. These styles will be requested on pages where the given transformation is used to display data (e.g. through a viewer web part).

- **Shared (pre-defined) page layouts** - CSS styles can be added to Page layouts in *Site Manager -> Development -> Page layouts*, where the Add CSS styles link is available when editing a specific layout. Once the shared layout is assigned to a Page template, the specified styles will be loaded on all pages that use the given template.

- **Custom page layouts for specific page templates** - in many cases, page templates use a custom page layout that is unique for the given template. To add CSS styles to this type of template, edit it in *Site Manager -> Development -> Page templates,* switch to its *Layout* tab and click Add CSS styles below the code of the custom layout.

- **Custom device layouts of specific page templates** - page templates support the creation of custom page layouts for specific device profiles. To add CSS styles to device layouts, edit the template in *Site Manager -> Development -> Page templates*, open the *Device layouts* tab, edit the device layout and click Add CSS styles below the code of the custom layout.

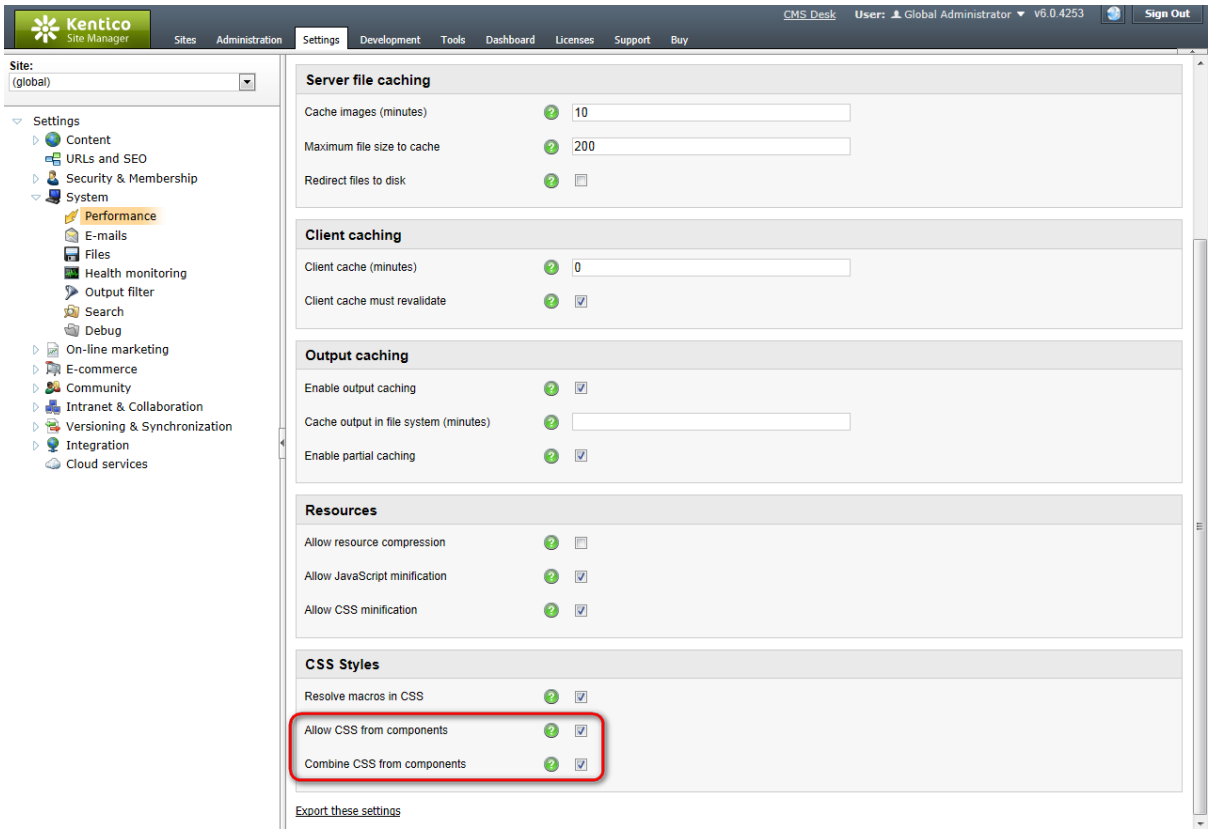
When a page is displayed in a user's browser, the system loads the assigned stylesheet (as described in the Overview topic) and then requests any styles defined for the components used on the given page. The final stylesheet available for the page is a combination of the main stylesheet and all component

styles. If any of the components contain an alternative definition for a CSS class that already exists in the main stylesheet, then the component style has a greater priority and overrides the original class.

By defining styles directly for components, you can reduce the size of your site (or page) stylesheets and organize them into more manageable sections. It also means that pages need to load less total CSS data, since each page only has to request styles for those components that are actually used on it. Additionally, the styles of components are automatically exported and imported along with the corresponding objects, which makes it easier to deploy them to websites that use a different stylesheet. The disadvantage of this approach is that at least one additional resource request must be added to pages containing styled components to ensure that all required CSS code is loaded.

Configuring component CSS

There are two global settings that affect the behavior of component CSS on all sites in the system. You can configure them in **Site Manager -> Settings -> System -> Performance** (only available if the *(global)* option is selected from the **Site** drop-down list).



The **Allow CSS from components** setting indicates if the styles of components should automatically be requested by the pages where they are placed. This is enabled by default. If disabled, it is necessary to either have all styles defined directly in the main stylesheet, or to link the styles of the required components into the stylesheet via CSS macros. Component styles may be linked into other stylesheets using the following expressions, depending on their type:

- **{%CSS.WebParts["<web part code name>"]%}**
- **{%CSS.WebPartLayouts["<web part code name>.<layout code name>"]%}**
- **{%CSS.Containers["<container code name>"]%}**

- **{%CSS.Transformations["<full transformation name>"]%}** - the transformation name must include the class name of the parent document type or custom table, for example: *CMS.News.Preview*
- **{%CSS.Layouts["<page layout code name>"]%}**
- **{%CSS.Templates["<page template code name>"]%}**

These macros are dynamically resolved into the CSS content defined for the specified component. The **Resolve macros in CSS** setting must be enabled if you wish to link styles using macros.

Style priority

Please note that component styles do not automatically take priority when linked through macros. The styles that are processed last (i.e. those which are "lower" in the code) are applied to the page.

As a result, it is recommended to place the linking macros below all other code in the given stylesheet if you want your components to override the definitions of existing CSS classes.

If **Combine CSS from components** is enabled, pages will load the CSS styles of all contained components via a single request. Otherwise, different types of components will each generate a separate request. The styles of multiple components of the same type (e.g. several web parts) are always retrieved by one request. Combining CSS requests may improve the load time of individual pages and is recommended in most cases.

7.5.3 App themes

When creating styles for your websites, you may leverage the built-in support for ASP.NET themes. You can use them to set styles for controls that do not have their own CSS class name, such as the Datagrid or Calendar.

Themes must be defined in a folder located under the **App_Themes** directory. The name of this folder needs to be the same as the code name of the used CSS stylesheet. For example, if you use the *Green* stylesheet on your site, your themes must be stored in the *App_Themes\Green* sub-folder.

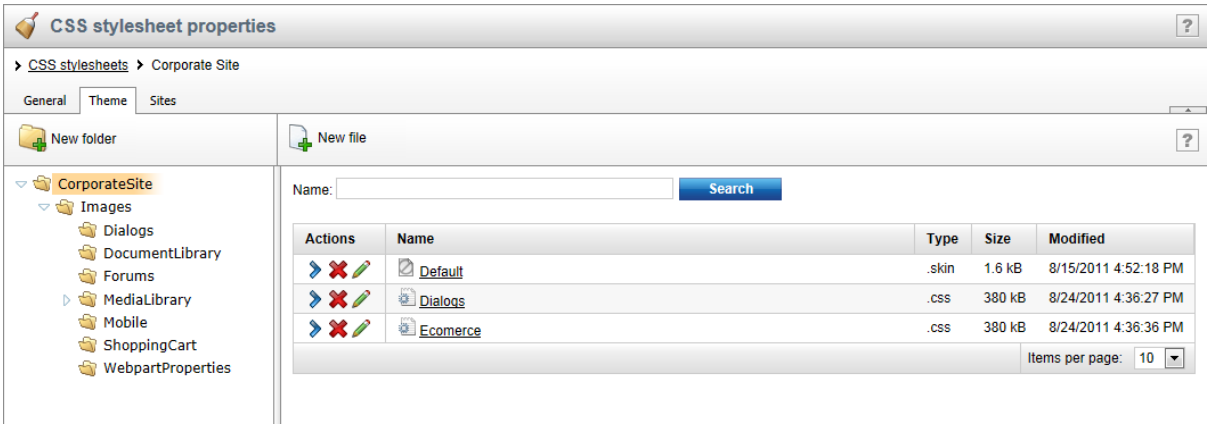
Skins for your controls need to be added to the **Default.skin** file under this folder. Here's an example of a skin for the **CMSCalendar** control / **Calendar** web part:

```
<cms:CMSCalendar Runat="server">
    <NextPrevStyle ForeColor="Red"></NextPrevStyle>
    <WeekendDayStyle BackColor="#E0E0E0"></WeekendDayStyle>
</cms:CMSCalendar>
```

Website design files

It is recommended to store all image files, flash movies and other resources that are part of the website design template under the theme folder of the stylesheet that uses them. This ensures that the files are exported together with the stylesheet if you deploy it to some other server.

The content of a stylesheet's theme folder may be managed directly from the administration interface in **Site Manager -> Development -> CSS stylesheets** by editing (✎) the given stylesheet and switching to the **Theme** tab.



Only file formats commonly used by CSS stylesheets are displayed here. This includes files with the following extensions:
.css, **.skin**, **.gif**, **.png**, **.bmp**, **.jpg**

Individual files of all types can be be edited (✎). Text files (.css or .skin) can be modified using an editor with syntax highlighting support and images are opened in the built-in Image editor.

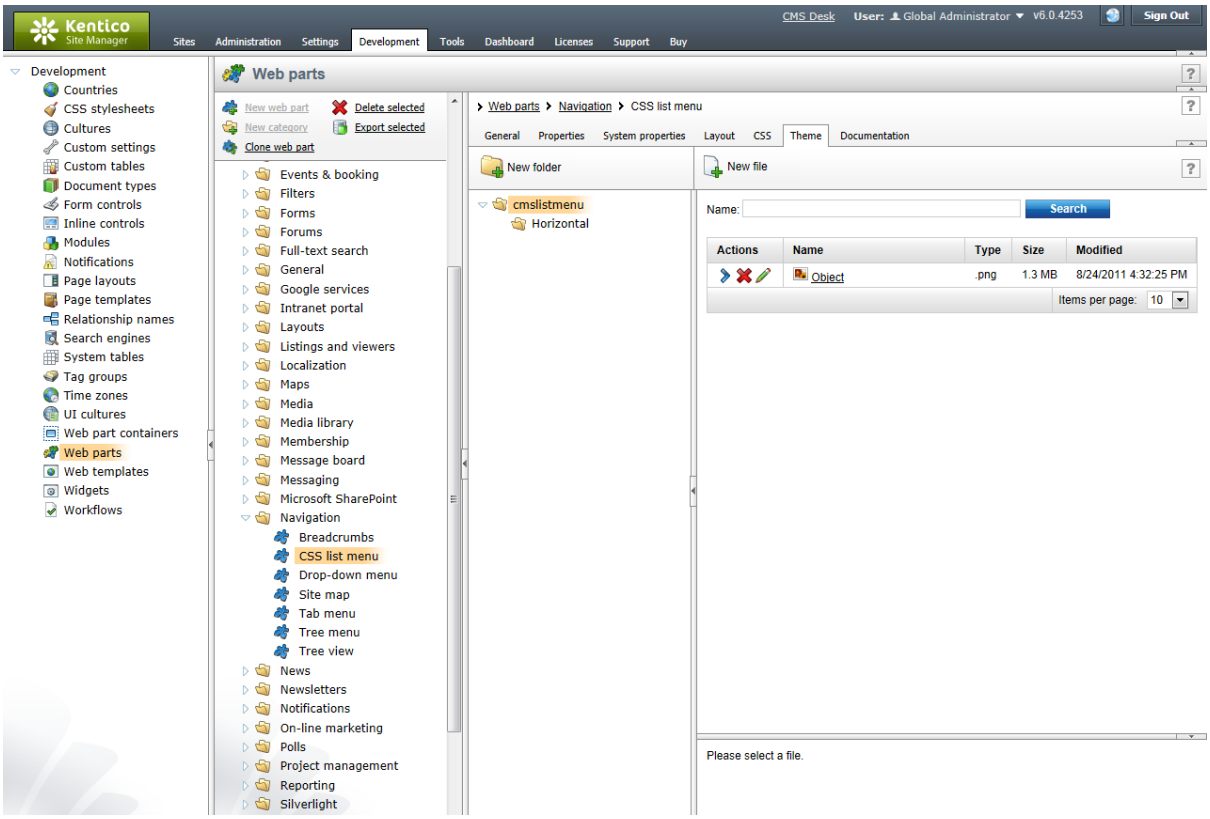
Themes folders for component styles

Design files may also be stored separately for the CSS styles assigned to the page components described in the CSS for page components topic. The theme folders of these components are located under the **App_Themes\Components** directory. Further sub-folders depend on the type of the given component:

- **WebParts\<web part code name>**
- **WebParts\<web part code name>\Layouts\<layout code name>**
- **Containers\<container code name>**
- **Transformations\<class name>\<transformation name>**
- **Layouts\<layout code name>**
- **PageTemplates\<template code name>**

Just like for stylesheets, it is recommended to store all required files in the appropriate theme folder for each component. The files in the theme folders are automatically exported and imported along with the objects representing the given component.

The content of these theme folders may be managed directly from the administration interface through the same type of dialog that is used for CSS stylesheets. It can be accessed on the **Theme** tab, which is available when editing any of the component types in **Site Manager -> Development**.



7.5.4 Printer friendly CSS styles

This chapter explains how to use printer friendly CSS styles on your website. These styles are used only if a document is sent to a printer.

1. Create a new CSS stylesheet in **Site Manager** -> **Development** -> **CSS stylesheets**, name it *Printer styles* and set its code name to *Printer_styles*, for example. See the simple example below for an illustration of the printer friendly CSS styles created for our default sample **Corporate site**.

[CSS]

```
.zoneLeft, .zoneRight, .zoneTopInfo, .zoneTop, .horizontalmenu, .zoneBottom
{
	display: none;
}
.eventCalendarDetail .zoneLeft, .eventCalendarDetail zoneRight
{
	display: block;
}
.eventCalendarDetail zoneRight
{
	float: left;
}
.logonReg .zoneLeft, .logonReg .zoneRight
{
```

```
        display: block;
}
.logonReg .zoneRight
{
        float: left;
}
.zoneContent
{
        float: left !important;
}
```

Please note that you have to hide all the elements that should not be visible in the print version. You can do this by adding **display:none;** to the given style.

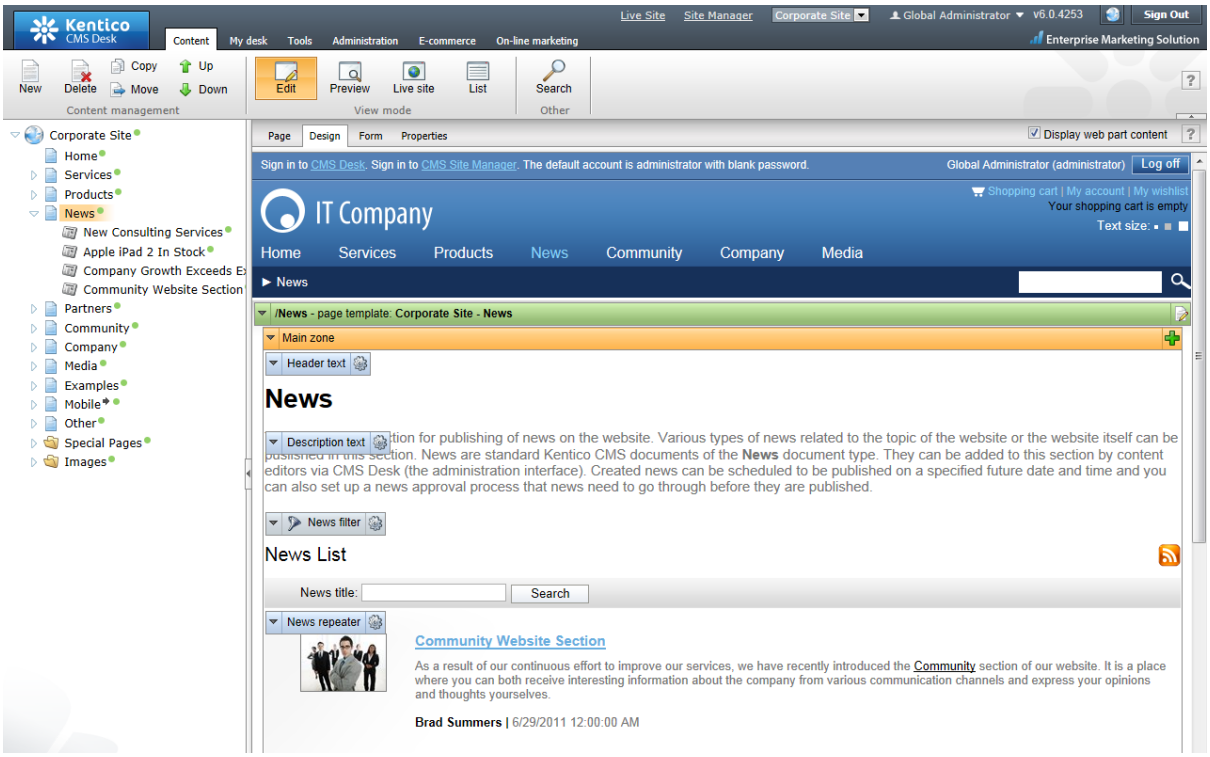
2. Add a link to this CSS stylesheet into the <*head*> tag of your master page. This can be done by selecting the root document of your website in **CMS Desk -> Content -> Edit** and switching to the **Master page** tab. For example:

```
<link href="CMSPages/GetResource.ashx?stylesheetname=Printer_styles" type="text/
css" rel="stylesheet" media="print" />
```

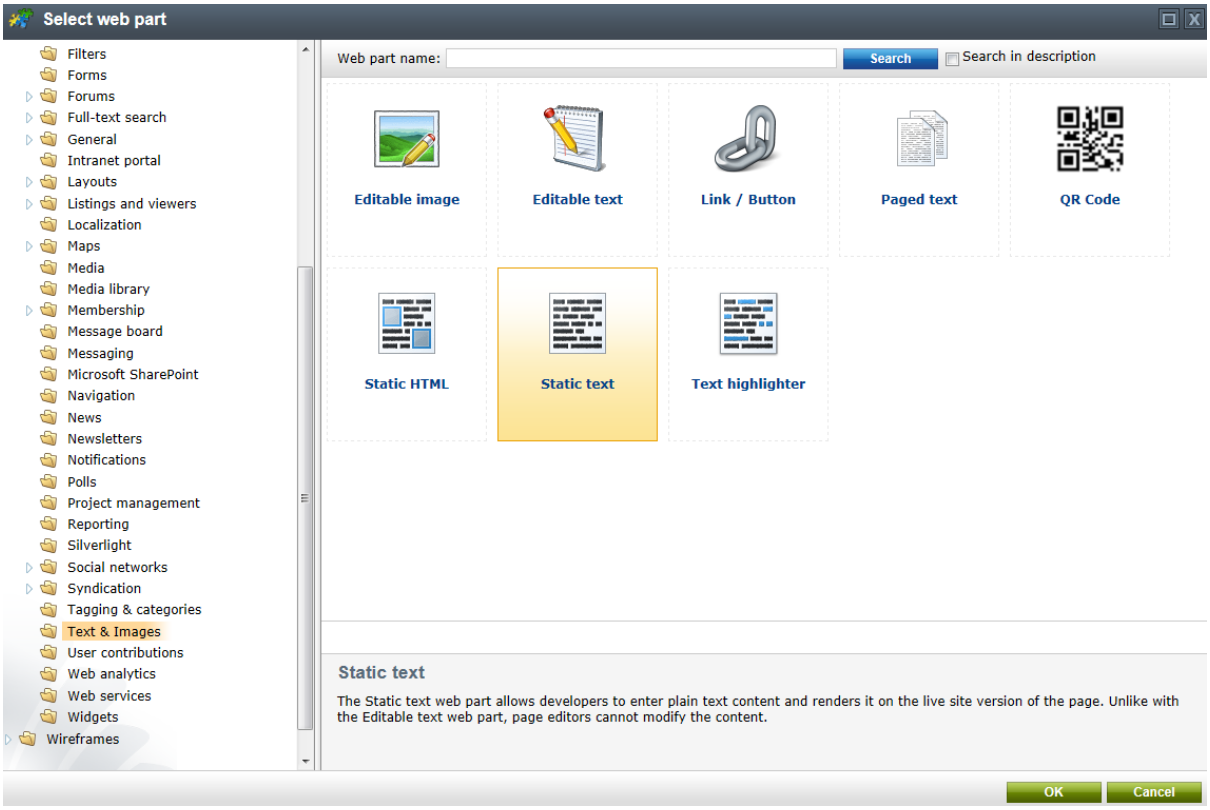
7.5.5 Print page

Kentico CMS allows you to add a link button to your web page that will create print version of the given document. The following example shows you on the sample Corporate Site how to create the given button for the news section.

1. Go to **CMS Desk** -> **Content** -> **News**, switch to the **Design** tab and click the **Add web part** () button of the **Main zone** web part zone.

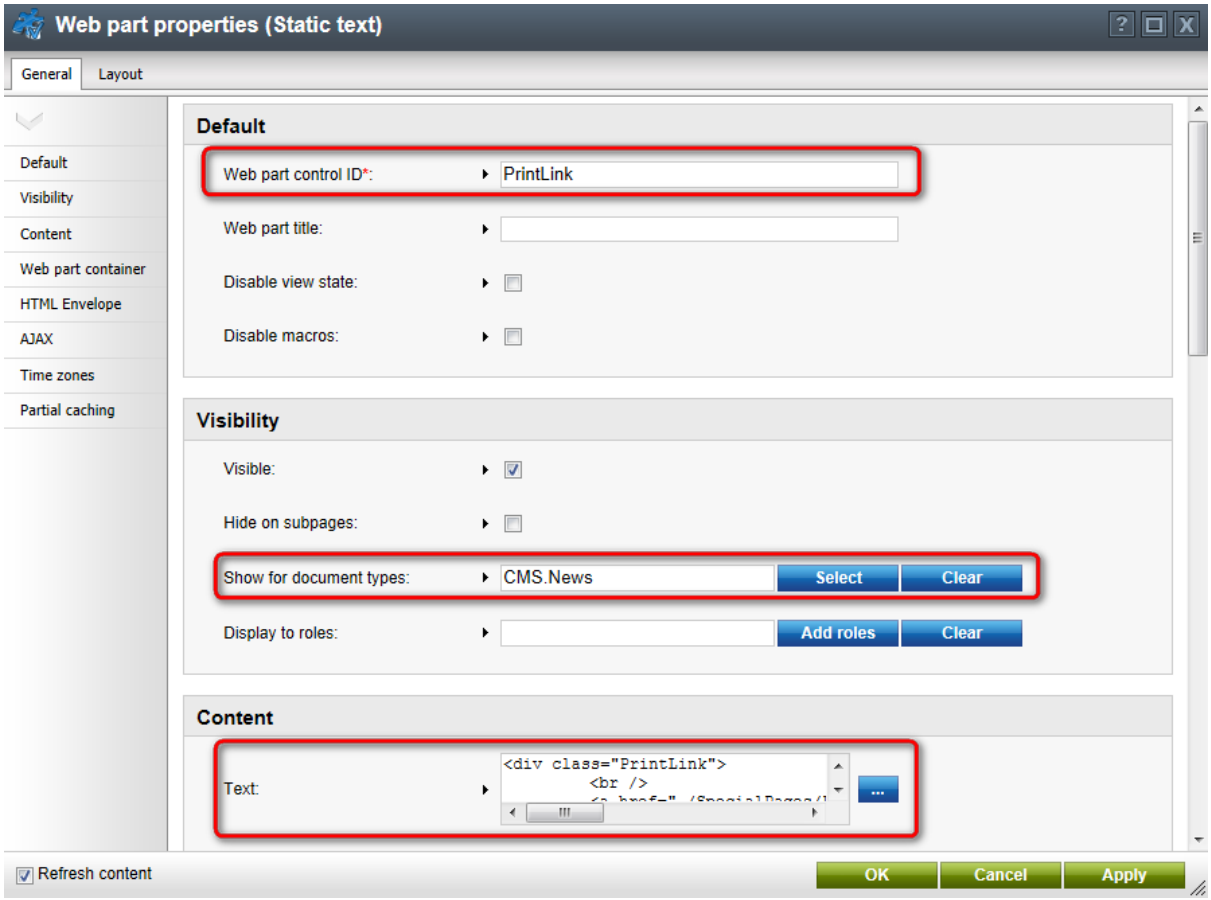


2. Select **Text & Images -> Static Text**.

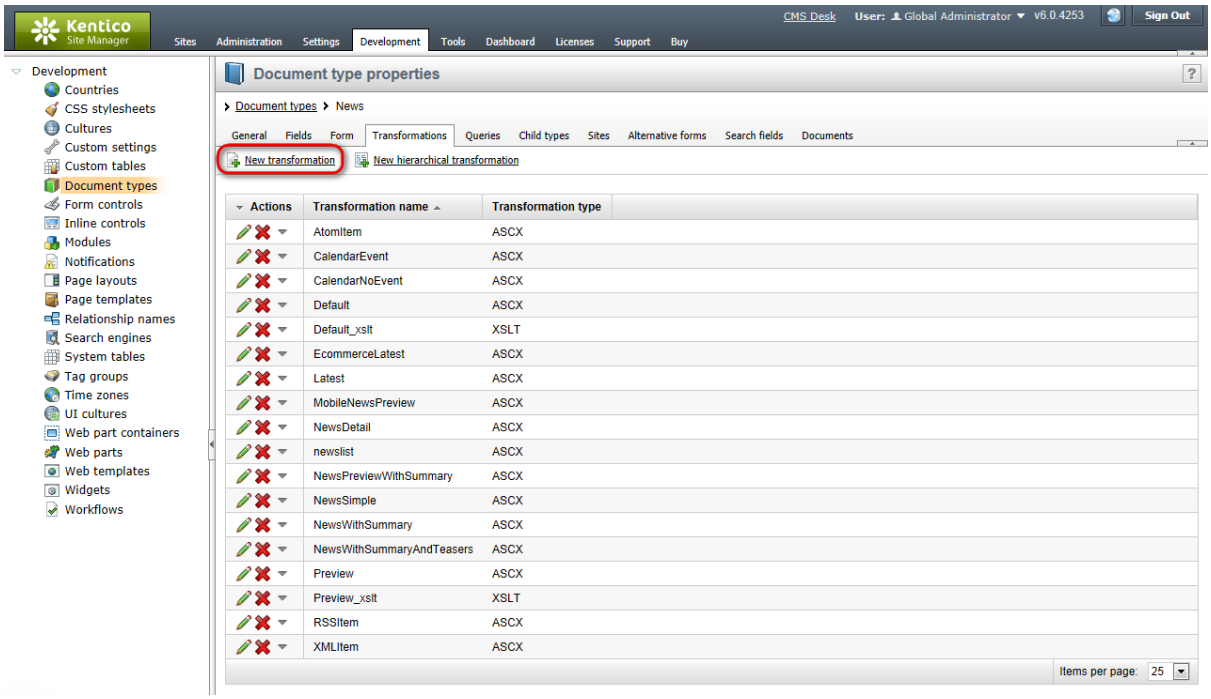


3. In the web part properties, enter *PrintLink* as the **Web part control ID** and choose **CMS.News** in the **Show for document types** property. Then enter the following code into the **Text** field and click **OK**.

```
<div class="PrintLink">
       <br />
       <a href="~/SpecialPages/E-Shop/Print.aspx?printpath={%NodeAliasPath%}
&classname={%ClassName%}" target="_blank" >
              <img class="PrintImage" src="~/App_Themes/CorporateSite/Images/
Print.gif" alt="Print" />
              Print
       </a>
</div>
```



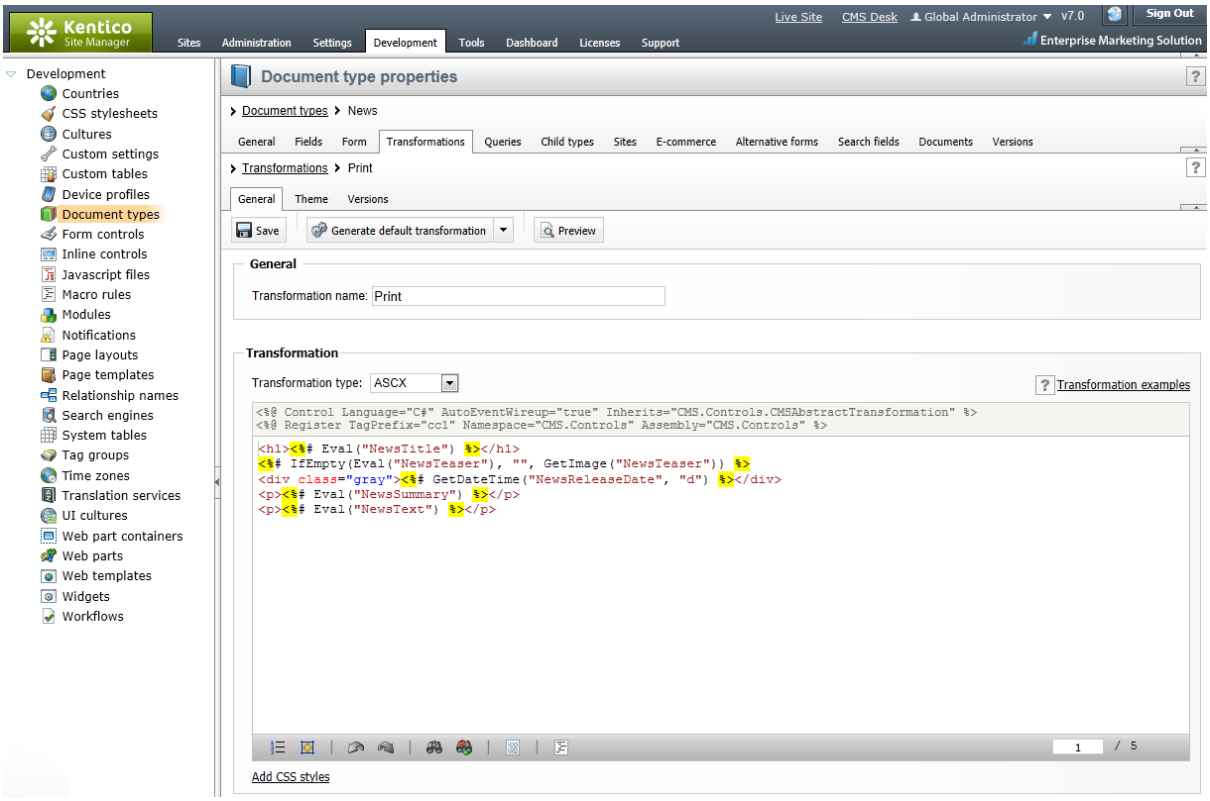
4. Now you need to create the print transformation for the News document type. Go to **CMS Site Manager** -> **Development** -> **Document types** and click the **Edit** () button next to the **News** document type. Switch to the **Transformation** tab and click **New Transformation**.



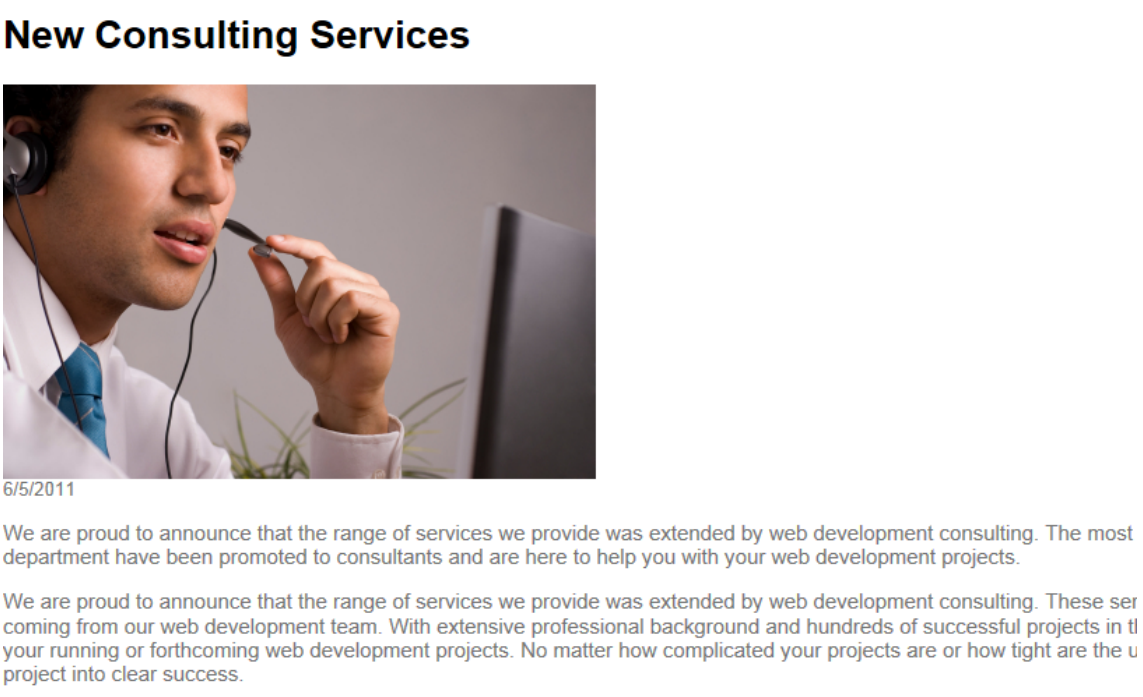
5. Type *Print* as the **Transformation name** and copy the following content into the code editor.

```
<h1><%# Eval("NewsTitle") %></h1>
<%# IfEmpty(Eval("NewsTeaser"), "", GetImage("NewsTeaser")) %>
<div class="gray"><%# GetDateTime("NewsReleaseDate", "d") %></div>
<p><%# Eval("NewsSummary") %></p>
<p><%# Eval("NewsText") %></p>
```

Click **Save**.



6. Now, go back to **CMS Desk** -> **Content** -> **News** -> **New Consulting Services** and click the newly created **Print** button. You will be redirected to the Corporate site's **Print** page that displays the print version of the given news item.

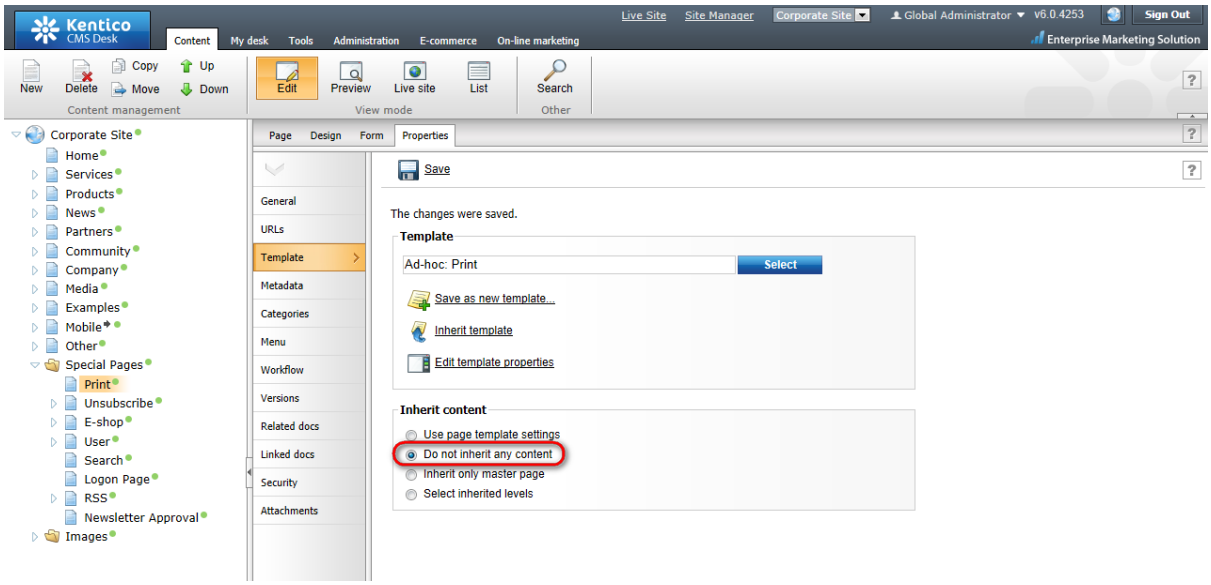


Creating the Print page on your site

The **Print** page used in the previous example is already included as part of the sample Corporate Site. On your own website, you will have to create it by yourself. The following steps need to be taken in order for the Print page to be created:

1. Add a new **Page (menu item)** document under the **Special pages** folder. Name it **Print** and use the **Create a blank page** option when selecting the page template.

2. First, you have to disable content inheritance in **Properties** -> **Template**.



3. Every **Print** page should contain a Repeater web part that renders the **Print** transformation for the given document type. The transformation can be specified in **Site Manager** -> **Development** -> **Document Types** -> **<edit (🖉) document type>** -> **Transformations**.

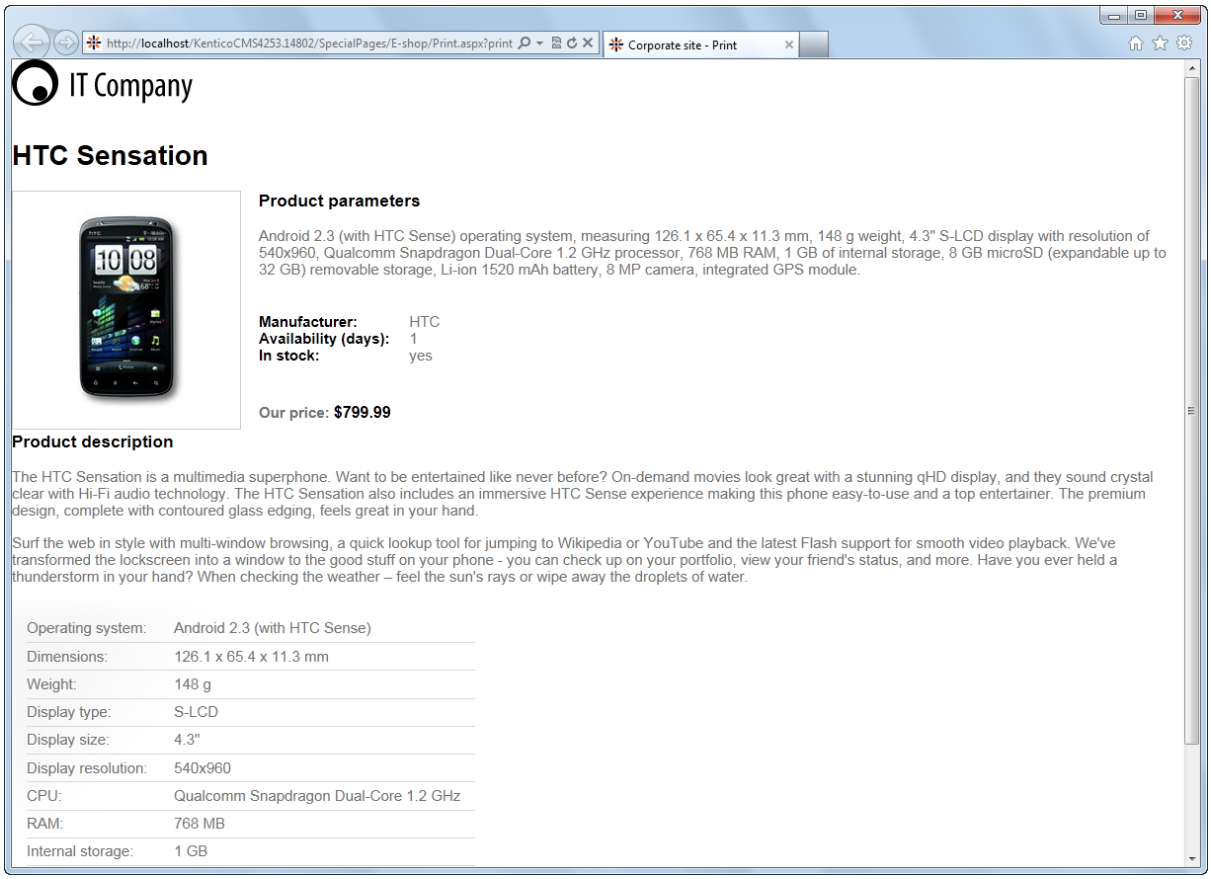
Therefore, add the **Repeater** web part and set its properties to the following values:

- **Path:** {?printpath|/%?}
- **Document types:** {?classname|cms.root?}
- **Transformation:** {?classname|cms.root?}.print

The **Path** property specifies the path to the document whose print version you want to make. The **Document types** specifies the document types that should be displayed. The **Transformation** property sets the name of the transformation that should be used to render the print version of the page.

The values above are macro expressions that load the actual values from the URL query string parameters included in the print link (*printpath* and *classname*). If no value is supplied in the URL (for instance if you go directly to the **Print** page from the content tree and not through the print link button), the default **Print** transformation for **cms.root** is displayed.

4. You can try out the functionality of the **Print** page by printing the detail of any product, because the **Print** link button is already created for all products on the Corporate Site.



7.5.6 Combining stylesheets

Kentico CMS allows you to easily combine the content of different stylesheets by entering special macros into stylesheet text. When such macros are resolved, they are replaced by the content of the specified stylesheet. This can be useful in certain situations, e.g. when creating a page-specific stylesheet as an extension of the main website stylesheet with some additional CSS classes.

You can link a stylesheet into the code of another one by inserting a special macro expression in the following format:

- *{% CSS["<stylesheet code name>"] %}*, e.g. *{% CSS["CorporateSite"] %}*
- *{% CSS.Stylesheets["<stylesheet code name>"] %}*, e.g. *{% CSS.Stylesheets["CorporateSite"] %}*

If the code name of the stylesheet does not contain any dots, you can also use the following simplified notation:

- *{% CSS.<stylesheetname> %}*, e.g. *{% CSS.CorporateSite %}*

Similarly, you can link component stylesheets the following way:

- *{% CSS.Containers["<web part container code name>"] %}*, e.g. *{% CSS.Containers["BlackBox"] %}*
- *{% CSS.Templates["<page template code name >"] %}*, e.g. *{% CSS.Templates ["CorporateSite.HomePage"] %}*
- *{% CSS.WebParts["<web part code name>"] %}*, e.g. *{% CSS.WebParts["AbuseReport"] %}*

- *{% CSS.WebPartLayouts["<web part layout full name>"] %}*, e.g. *{% CSS.WebPartLayouts ["AbuseReport.MyCustomLayout"] %}*
- *{% CSS.Transformations["<transformation full name>"] %}*, e.g. *{% CSS.Transformations ["CMS.Article.Default"] %}*

The simplified notation can also be used if the component name does not contain any dots (web part layouts and transformations always contain dots in their full names, hence their stylesheets can only be linked using the notation above):

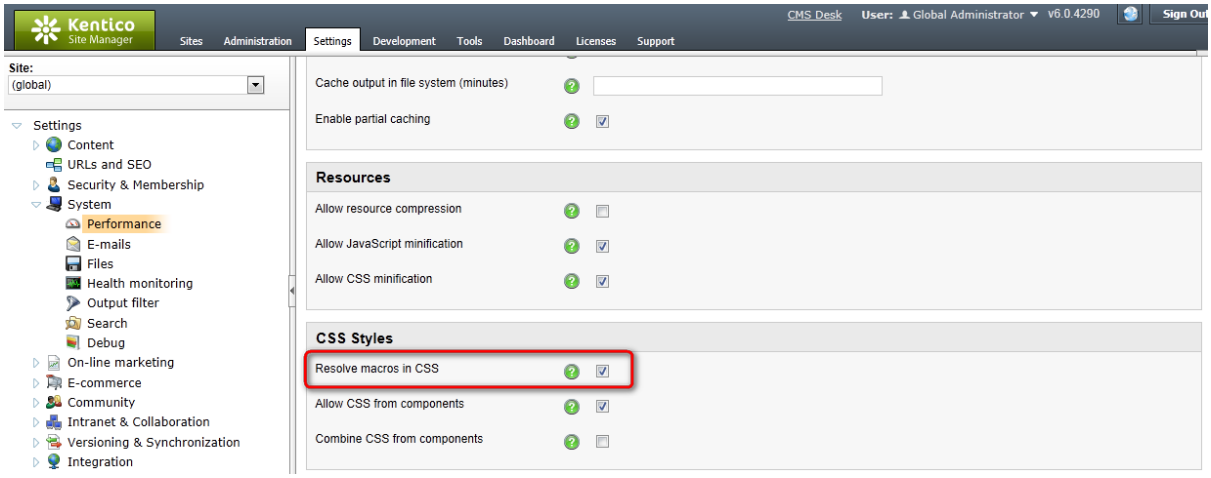
- *{% CSS.Containers.<web part container code name> %}*, e.g. *{% CSS.Containers.BlackBox %}*
- *{% CSS.Templates.<page template code name > %}*, e.g. *{% CSS.Templates.MyTemplate %}*
- *{% CSS.WebParts.<web part code name> %}*, e.g. *{% CSS.WebParts.AbuseReport %}*

Disabling resolving of CSS macros

Resolving of CSS macros is enabled by default, but can be disabled if needed via the **Resolve macros in CSS** setting in **Site Manager -> Settings -> System -> Performance** (only available if the *(global)* option is selected from the **Site** drop-down list).

Warning!

If you wish to disable the **Resolve macros in CSS** setting, it is first necessary to remove all occurrences of macros from your stylesheets. Unresolved macro expressions are not valid CSS code and as a result, the given stylesheets will not be processed correctly by browsers.



Using the @import directive

You can alternatively load the content of other stylesheets by adding the standard **@import** directive to the beginning of the stylesheet code, for example:

```
@import url(/KenticoCMS/CMSPages/GetResource.ashx?stylesheetname=corporatesite);
```

This may also be used to import external CSS stylesheets from static files under your web project, e.g.:

```
@import url(/KenticoCMS/CMSPages/GetResource.ashx?stylesheetfile=/KenticoCMS/
App_Themes/CorporateSite/Blue.css);
```

7.5.7 CSS stylesheet internals and API

7.5.7.1 Overview

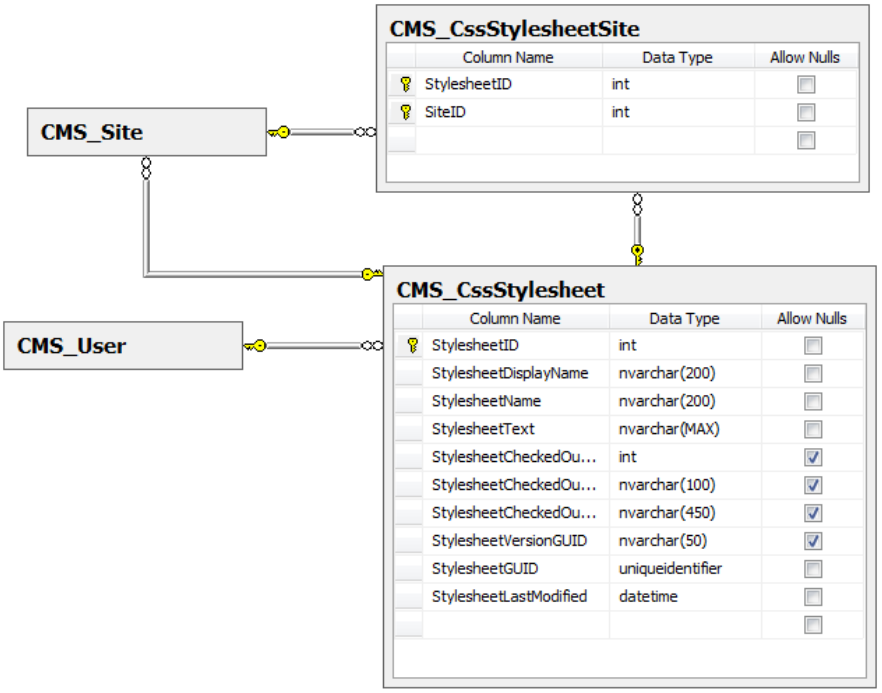
In this chapter, you will learn which database tables and API classes are used by CSS stylesheets. You will also see the most common API examples.

Further API-related information and examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all methods from the module's classes, please refer to Kentico CMS API Reference.

7.5.7.2 Database tables

The following database tables are used to store information about CSS stylesheets:

- **CMS_CssStylesheet** - contains records representing css stylesheets.
- **CMS_CssStylesheetSite** - stores relationships between stylesheets and sites. Each entry in this table indicates that a specific stylesheet can be used on a given site.



7.5.7.3 API classes

If you are not familiar with database table data management through Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

The classes of CSS stylesheets can be found in the **CMS.SiteProvider** namespace.

CMS_CssStylesheet table API:

- **CssStyleSheetInfo** - represents one stylesheet object.
- **CssStyleSheetInfoProvider** - provides management functionality for stylesheets.

CMS_CssStylesheetSite table API:

- **CssStyleSheetSiteInfo** - represents a relationship between a stylesheet and a site.
- **CssStyleSheetSiteInfoProvider** - provides management functionality for stylesheet-site relationships.

7.5.7.4 API examples

7.5.7.4.1 Overview

These topics show examples of how the API of CSS stylesheets can be used:

- Managing CSS stylesheets
- CSS stylesheets and sites

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Development\CSSStyleSheets\Default.aspx.cs***.

The CSS stylesheet API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.GlobalHelper;
using CMS.CMSHelper;
using CMS.SiteProvider;
```

7.5.7.4.2 Managing CSS stylesheets

The following example creates a CSS stylesheet.

```
private void CreateCssStylesheet()
{
    // Create new css stylesheet object
    CssStylesheetInfo newStylesheet = new CssStylesheetInfo();

    // Set the properties
    newStylesheet.StylesheetDisplayName = "My new stylesheet";
    newStylesheet.StylesheetName = "MyNewStylesheet";
    newStylesheet.StylesheetText = "Some CSS code";

    // Save the css stylesheet
    CssStylesheetInfoProvider.SetCssStylesheetInfo(newStylesheet);
}
```

The following example gets and updates a stylesheet.

```
private bool GetAndUpdateCssStylesheet()
{
    // Get the css stylesheet
    CssStylesheetInfo updateStylesheet =
CssStylesheetInfoProvider.GetCssStylesheetInfo("MyNewStylesheet");
    if (updateStylesheet != null)
    {
        // Update the properties
        updateStylesheet.StylesheetDisplayName =
updateStylesheet.StylesheetDisplayName.ToLower();

        // Save the changes
        CssStylesheetInfoProvider.SetCssStylesheetInfo(updateStylesheet);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates stylesheets.

```
private bool GetAndBulkUpdateCssStylesheets()
{
    // Prepare the parameters
    string where = "StylesheetName LIKE N'MyNewStylesheet%'";

    // Get the data
    DataSet stylesheets = CssStylesheetInfoProvider.GetCssStylesheets(where,
null);
    if (!DataHelper.DataSourceIsEmpty(stylesheets))
```

```
    {
        // Loop through the individual items
        foreach (DataRow stylesheetDr in stylesheets.Tables[0].Rows)
        {
            // Create object from DataRow
            CssStylesheetInfo modifyStylesheet = new CssStylesheetInfo
(stylesheetDr);

            // Update the properties
            modifyStylesheet.StylesheetDisplayName =
modifyStylesheet.StylesheetDisplayName.ToUpper();

            // Save the changes
            CssStylesheetInfoProvider.SetCssStylesheetInfo(modifyStylesheet);
        }

        return true;
    }

    return false;
}
```

The following example deletes a stylesheet.

```
private bool DeleteCssStylesheet()
{
    // Get the css stylesheet
    CssStylesheetInfo deleteStylesheet =
CssStylesheetInfoProvider.GetCssStylesheetInfo("MyNewStylesheet");

    // Delete the css stylesheet
    CssStylesheetInfoProvider.DeleteCssStylesheetInfo(deleteStylesheet);

    return (deleteStylesheet != null);
}
```

7.5.7.4.3 CSS stylesheets and sites

The following example assigns a CSS stylesheet to a site.

```
private bool AddCssStylesheetToSite()
{
    // Get the css stylesheet
    CssStylesheetInfo stylesheet = CssStylesheetInfoProvider.GetCssStylesheetInfo
("MyNewStylesheet");
    if (stylesheet != null)
    {
        int stylesheetId = stylesheet.StylesheetID;
        int siteId = CMSContext.CurrentSiteID;
```

```
        // Save the binding
        CssStylesheetSiteInfoProvider.AddCssStylesheetToSite(stylesheetId,
siteId);

        return true;
    }

    return false;
}
```

The following example removes a stylesheet from a site.

```
private bool RemoveCssStylesheetFromSite()
{
    // Get the css stylesheet
    CssStylesheetInfo removeStylesheet =
CssStylesheetInfoProvider.GetCssStylesheetInfo("MyNewStylesheet");
    if (removeStylesheet != null)
    {
        int siteId = CMSContext.CurrentSiteID;

        // Get the binding
        CssStylesheetSiteInfo stylesheetSite =
CssStylesheetSiteInfoProvider.GetCssStylesheetSiteInfo
(removeStylesheet.StylesheetID, siteId);

        // Delete the binding
        CssStylesheetSiteInfoProvider.DeleteCssStylesheetSiteInfo(stylesheetSite);

        return true;
    }

    return false;
}
```

7.6 Debugging and system information

7.6.1 Overview

Kentico CMS provides a useful user interface where general information about the system can be viewed. If you need to find out more detailed information internal activity within the system, you can use one of the many available debugs.

General system information

General information about the system and the environment where it is running can be viewed on the **General** tab in **Site Manager -> Administration -> System**. More information about the information displayed in this UI and the actions that can be performed there can be found in the System information UI topic.

Debugging in the UI

The **Administration -> System -> Debug** tab can be used for system debugging, i.e. finding and fixing performance or setting issues on your website. Debugs can also be particularly useful when reporting an issue to our support department. Every extra bit of information related to your issue that you find in the debugs can significantly shorten the time needed to find the solution.

The debugging interface is divided into several sub-tabs. Only the following two tabs are displayed by default. Click the name of the tab to learn more.

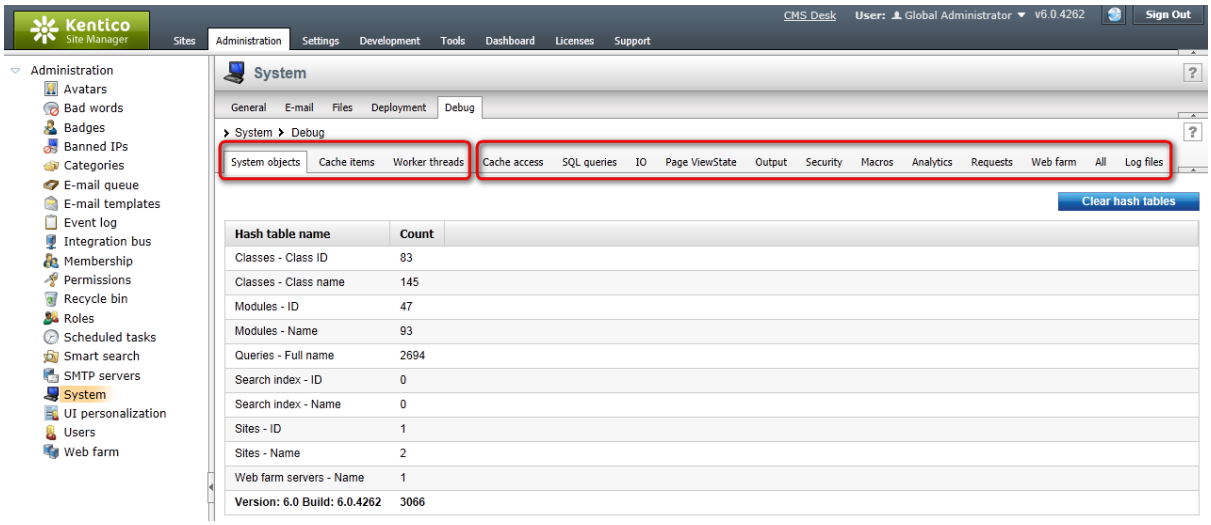
- System objects
- Cache items
- Worker threads

The other sub-tabs are displayed only after enabling certain settings in **Site Manager -> Settings -> System -> Debug** (recommended) or adding certain keys into the **AppSettings** section of your **web.config** file. Here again, click the name of the tab to get detailed information.

- Cache access
- SQL queries
- IO
- Page ViewState
- Output
- Security
- Macros
- Analytics
- Requests
- Web farm
- All
- Log files

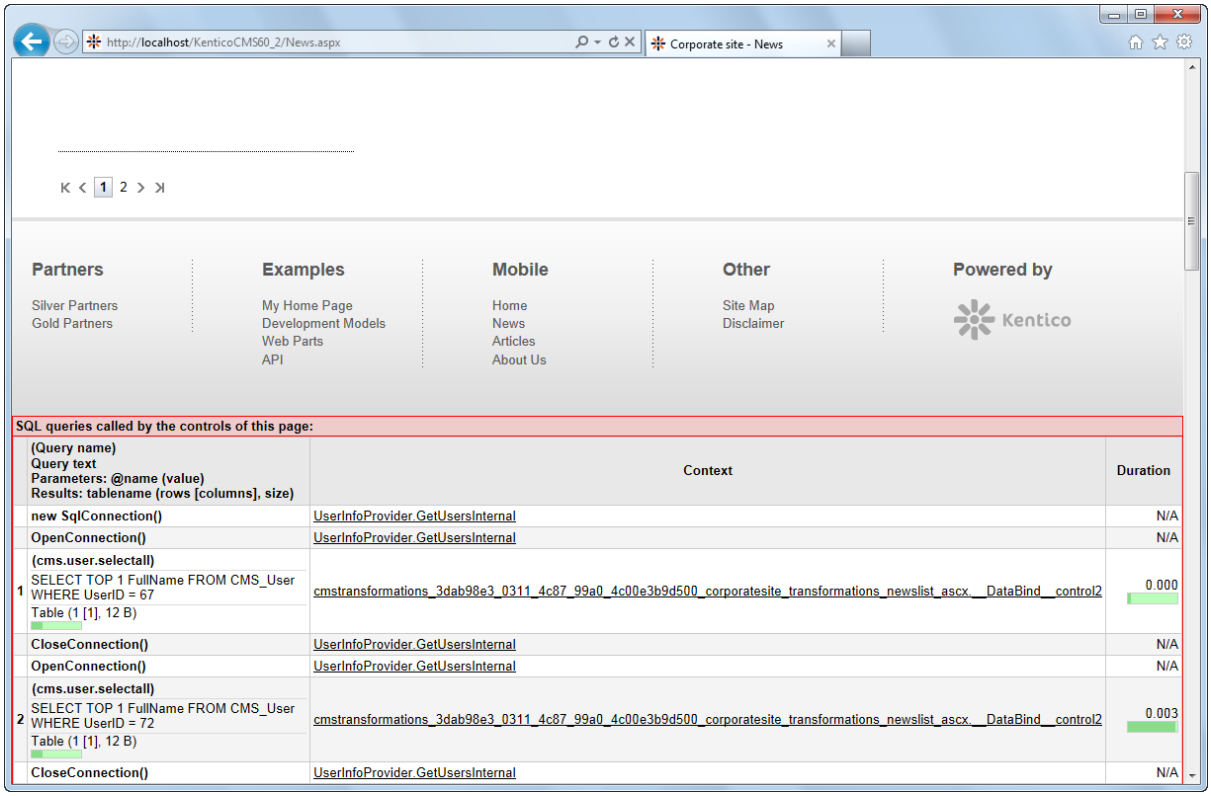
E-mail debugging is enabled via modifying the web.config file exclusively and does not have its own tab in **Administration -> System -> Debug**. Refer to the E-mail debugging topic for more information.

You can also enable all these debugs at once using the general settings and keys.



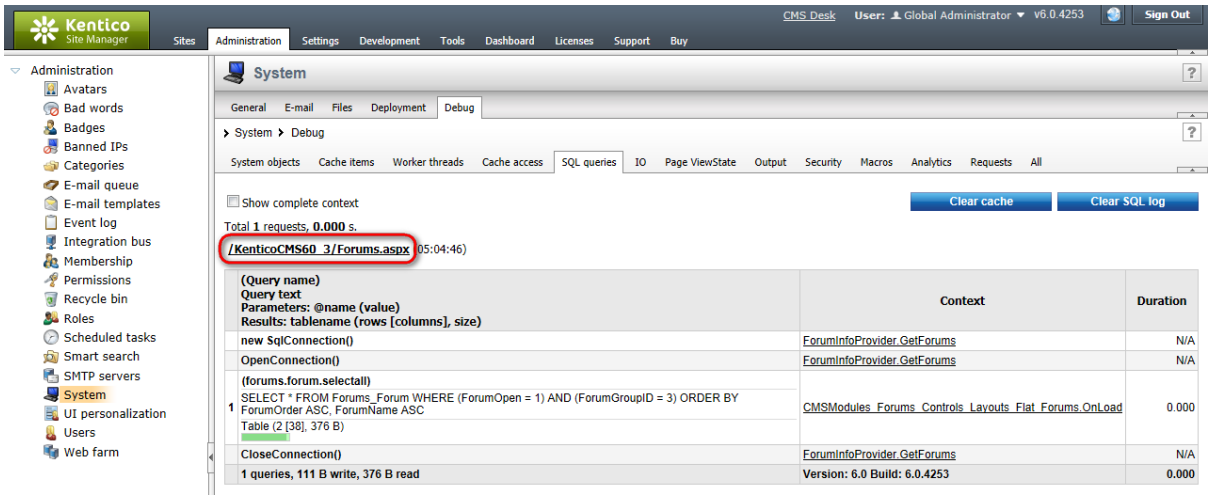
Debugging on the live site

You can also enable debugging directly on the live site. In this case, each loaded page also displays a debug log below its regular content. Only information related to the given page is shown. Like UI debugs, live site debugging can also be enabled either separately using the dedicate settings and keys (follow the links above), or all at once using the general settings and keys.

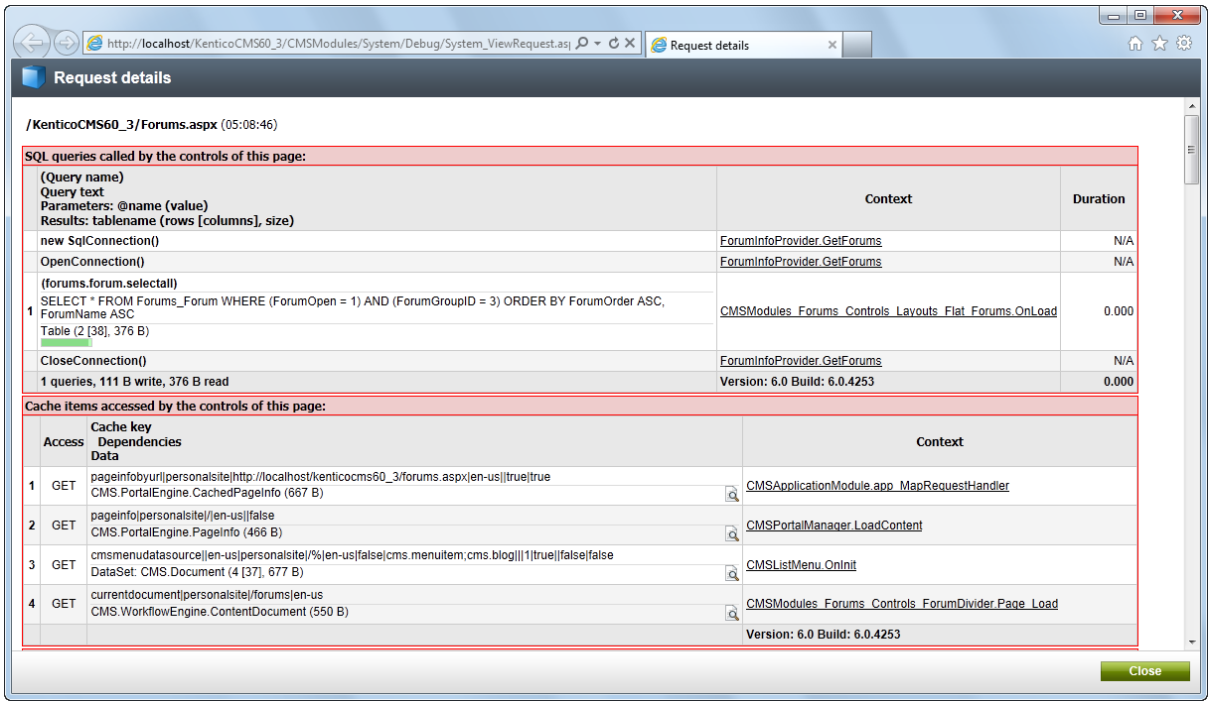


Debug request details

You can display request details for each debugged request by clicking its URL in respective debug logs in the **Site Manager -> Administration -> System -> Debug** interface.



After doing so, a new pop-up window opens, listing the request's debug log for all enabled debugs.



7.6.2 System information UI

On the **General** tab in **Site Manager -> Administration -> System**, you can find general information about the system and the environment in which it is running. The following information and actions are available on the tab:

System information

- **Machine name** - name of the machine on which the system is running.
- **ASP.NET account** - name of the user account used by ASP.NET in the operating system on your machine.

- **ASP.NET version** - version of ASP.NET on which the system is running.
- **Application pool name** - name of the application pool in which this instance of Kentico CMS is running.
- **Application trust level** - trust level of the environment where the application is running.
- **Your IP address** - IP address from which you are accessing the system.

Database information

- **Server name** - name of the database server on which the system database is running.
- **Server version** - version of SQL Server installed on the database server.
- **Database name** - name of the system database on the database server.
- **Database size** - current size of the database, including the actual database data (the *.mdf* file) and the database log (the *.ldf* file).

Contact management database separation

If the application has a separated contact management (on-line marketing) database, the information will be displayed for both databases.

Using the **Refresh interval (seconds)** drop-down list, you can set the interval after which the values in the sections below will be refreshed.

Memory statistics

- **Allocated memory** - size of memory allocated for the CMS.
- **Peak memory usage** - maximum memory used by the process.
- **Process physical memory** - physical memory used by the process.
- **Process virtual memory** - memory allocated by the process in the virtual memory space.

- **Clear unused memory** - clicking this button clears unused data stored in the memory so that more free memory is available.

Garbage collection statistics

- **GC collection of gen 0** - number of unused memory cleanings initiated by the garbage collector for generation 0 objects.
- **GC collection of gen 1** - number of unused memory cleanings initiated by the garbage collector for generation 1 objects.
- **GC collection of gen 2** - number of unused memory cleanings initiated by the garbage collector for generation 2 objects.

Page view statistics

- **View of content pages** - number of content pages displayed since last restart.
- **File downloads and views** - number of files downloaded and viewed since last restart.
- **Views of system pages** - number of system pages viewed since last restart.
- **Non-page requests** - number of non-page requests handled since last restart.
- **Number of pages not found** - number of page view requests which resulted in the *404: Page Not Found* error since the last restart.

- **Pending requests** - number of currently pending (not yet processed) page requests.

Cache statistics

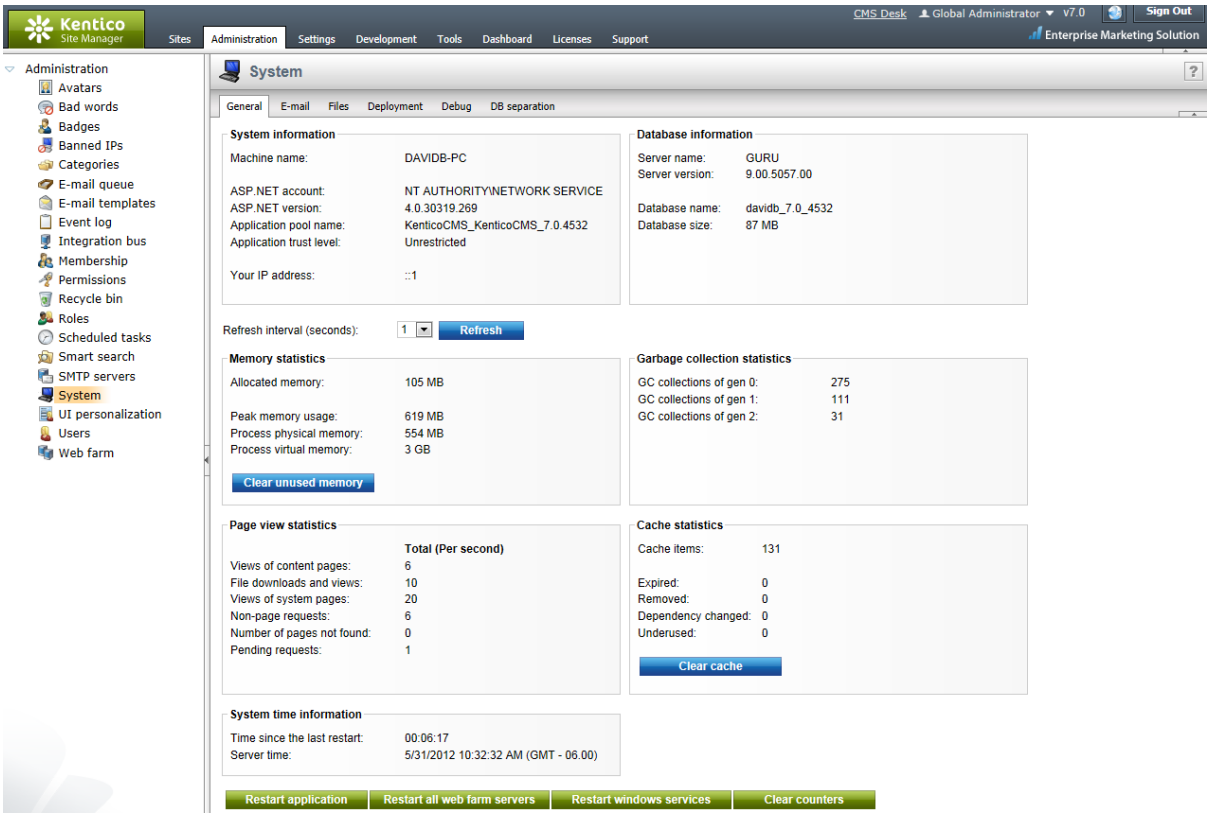
- **Cache items** - number of items in the system cache.
- **Expired** - number of expired cache items in the system cache.
- **Removed** - number of items removed from the system cache since last restart.
- **Dependency changed** - cache items dropped based on the change of their dependencies.
- **Underused** - cache items dropped earlier than configured, possible due to lack of memory.

- **Clear cache -** clears all items stored in the cache.

- **Time since last restart** - time since the system was last restarted (or since it was launched if no restart was performed).

You can also perform the following actions using the buttons at the bottom part of the tab:

- **Restart application** – restarts Kentico CMS (in need to take effects of some changes).
- **Restart all web farm servers** - restarts all web farm servers running this instance of Kentico CMS. This action is only displayed when the system is configured to run in web farm environment. See Modules -> Web farm synchronization for more details.
- **Restart windows services** - restarts all Kentico CMS Windows services related to this instance. This action is only displayed when there is at least one Windows service installed for this instance of Kentico CMS. See External utilities -> Service Manager for more details.
- **Clear counters** - clears the values currently stored in all performance counters registered for the current Kentico CMS instance. See Modules -> Health monitoring for more details. Clicking this button also resets the *Page view statistics* section of this tab.



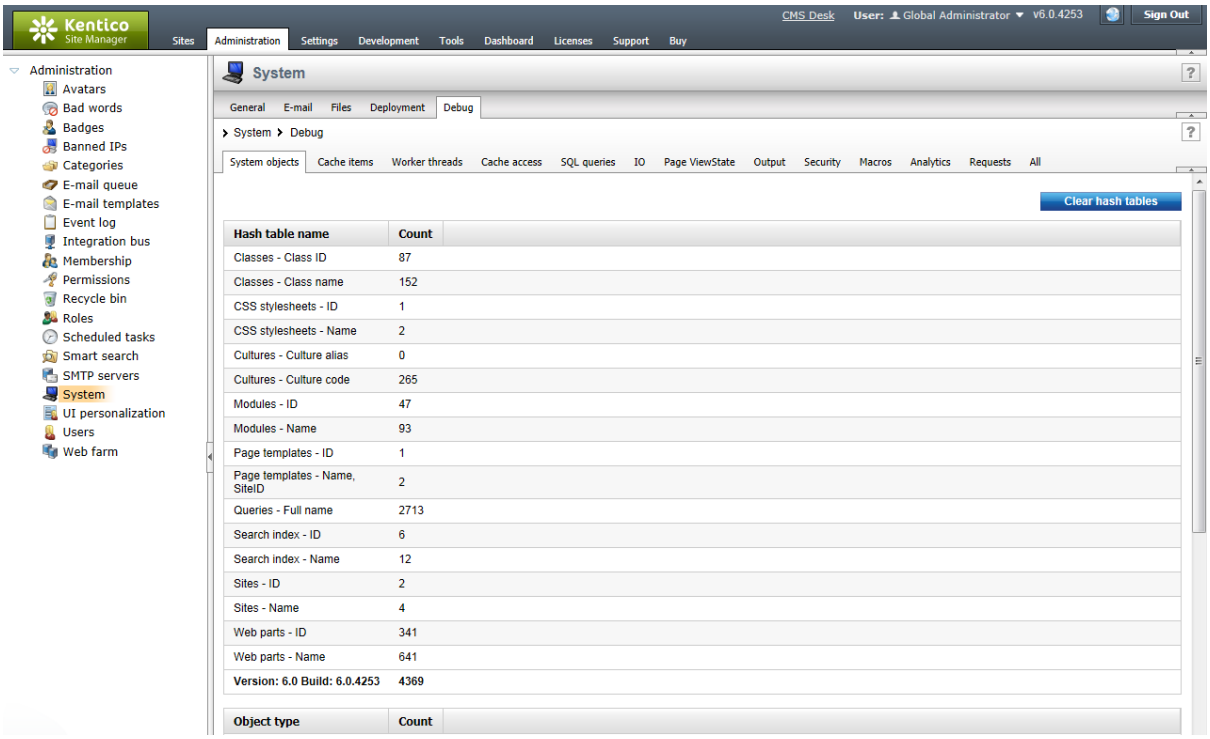
7.6.3 Particular debug tabs

7.6.3.1 System objects

On the **Debug -> System objects** tab, you can see the number of objects currently stored in hash tables on the server.

The first table shows the names of particular **hash tables** and the number of records in them. The second table shows particular **Object types** and the number of currently hashed objects of each type.

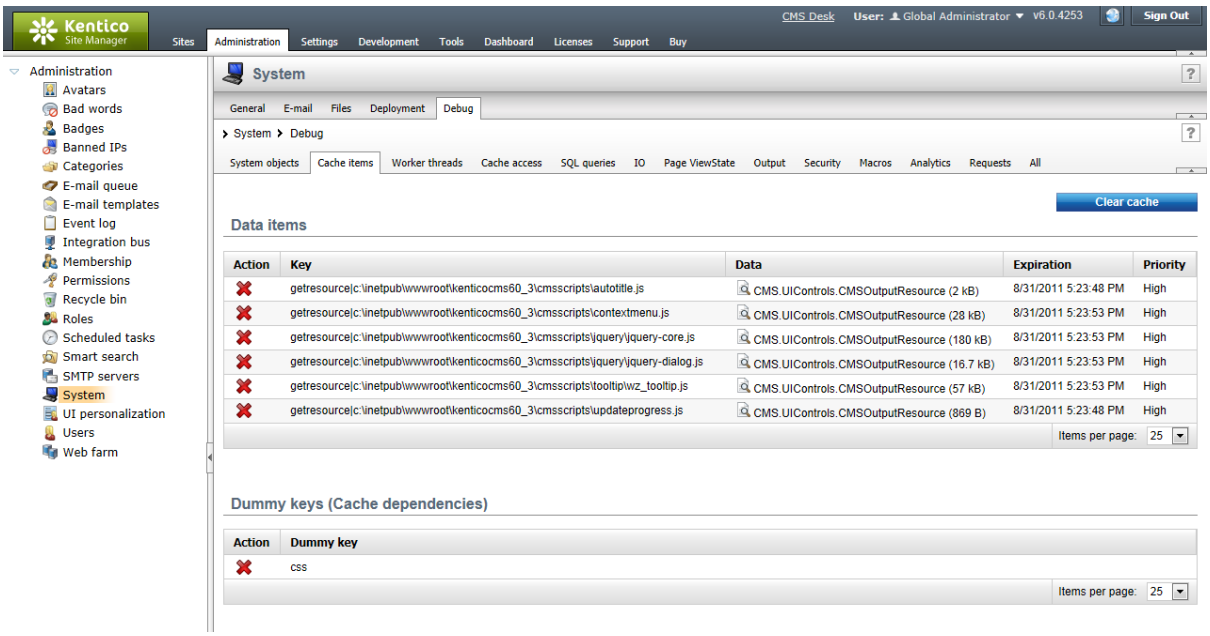
All hash tables can be cleared using the **Clear hash tables** button.



7.6.3.2 Cache items

On the **Debug -> Cache items** tab, you can see which **Keys** and **Dummy keys** are currently stored in the system cache.

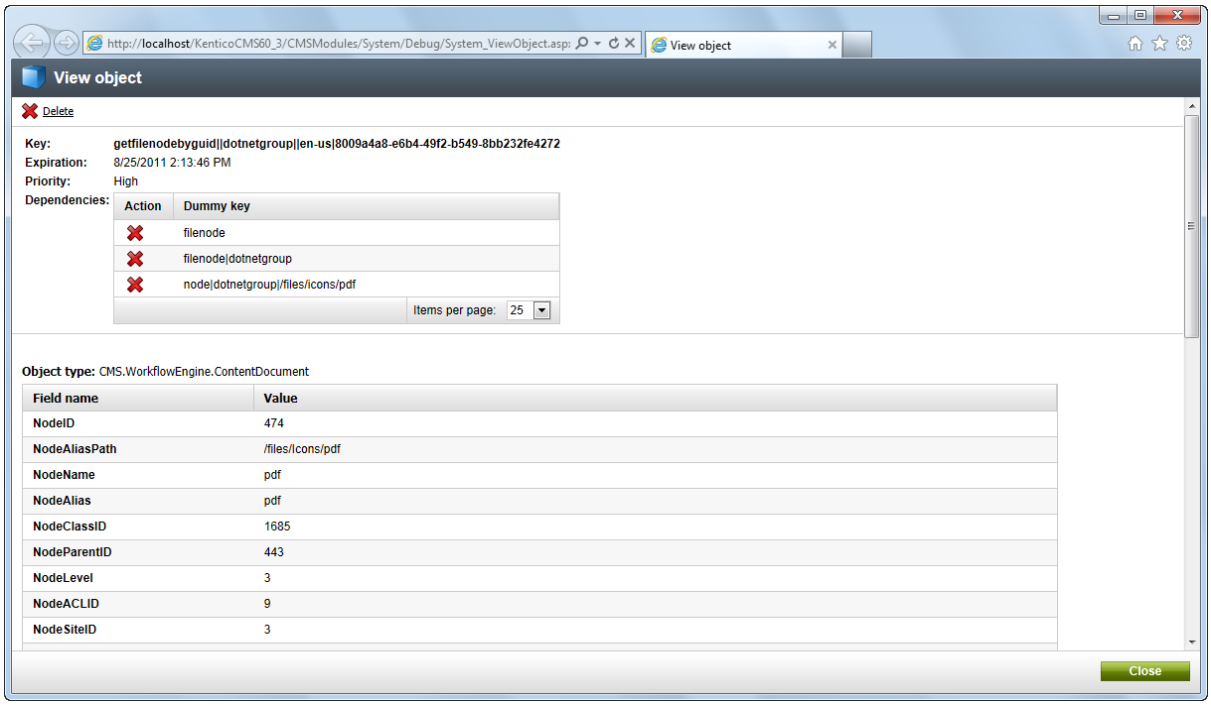
Both real and dummy keys can be deleted from within this part of the user interface using the **Delete** (✖) icon next to them. All dummy keys can be cleared from the cache using the **Clear cache** button.



If you click the **View object** ( ) icon in a particular row, a pop-up window gets opened as in the screenshot below. There, you can see detailed information about the cached object:

- **Key** - the key under which the object is stored in the cache.
- **Expiration** - date and time when the cache item will expire (i.e. will be removed from the cache).
- **Priority** - priority of the cache item. The same as ASP.NET cache item priorities, while only *High* and *NotRemovable* are used in Kentico CMS.
- **Dependencies** - dummy keys on which the cache item is enabled.
- **Object type** - type of the cached object.

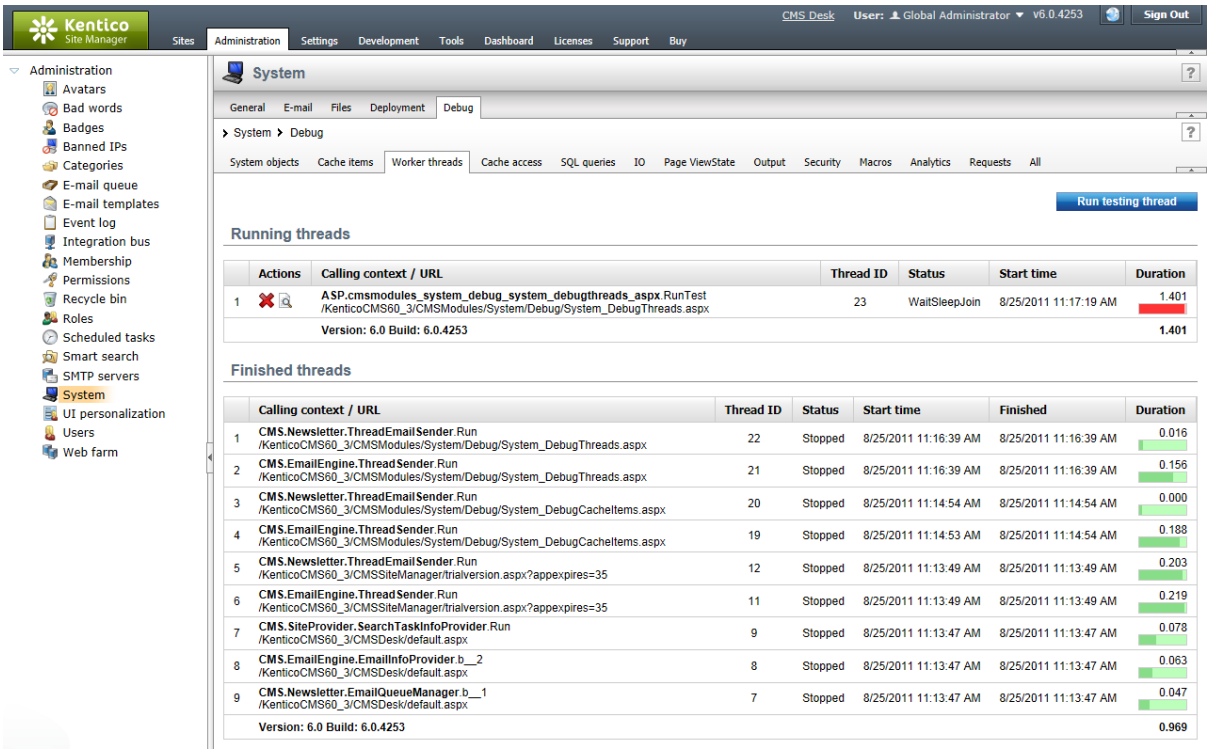
Finally, the table under the **Object type** value represents the actual data of the cached object. Displayed fields vary based on the currently displayed object type. The cache item can be cleared from cache using the **Delete** button in the top part of the window.



For more information about caching in Kentico CMS, please refer to Development -> Caching and performance -> Caching options.

7.6.3.3 Worker threads

On the **Debug -> Worker threads** tab, you can see which worker threads are currently running in the system and which threads have finished recently. This is particularly useful if your application is consuming significantly more resources than it normally should. It may often be caused by threads running in the background, e.g. threads for Smart Search re-indexing or when sending mass e-mails.



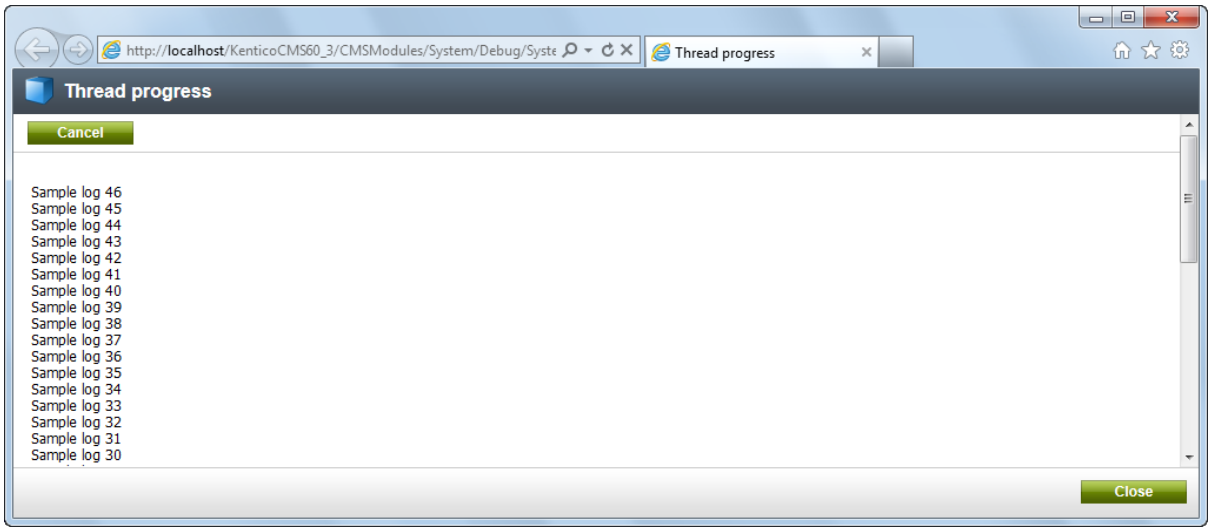
The UI is split into two sections:

Running threads

In this section, you can see a list of all worker threads currently running in the system. The **Run testing thread** button can be used to launch a testing thread in order to verify that thread debugging works correctly.

You can perform the following actions with the listed threads:

- **Cancel** - cancels execution of the thread (useful if the original window where the thread was started is already closed).
- **View** - displays a pop-up window showing the current content of the thread log (in case that there is a log), as shown in the screenshot below.



Finished threads

In this section, you can see a list of the latest threads that have finished their activity in the recent past. By adding the following key into the *appSettings* section of your *web.config* file, you can limit the number of threads displayed in the list. If the key is not used, 20 threads are displayed by default.

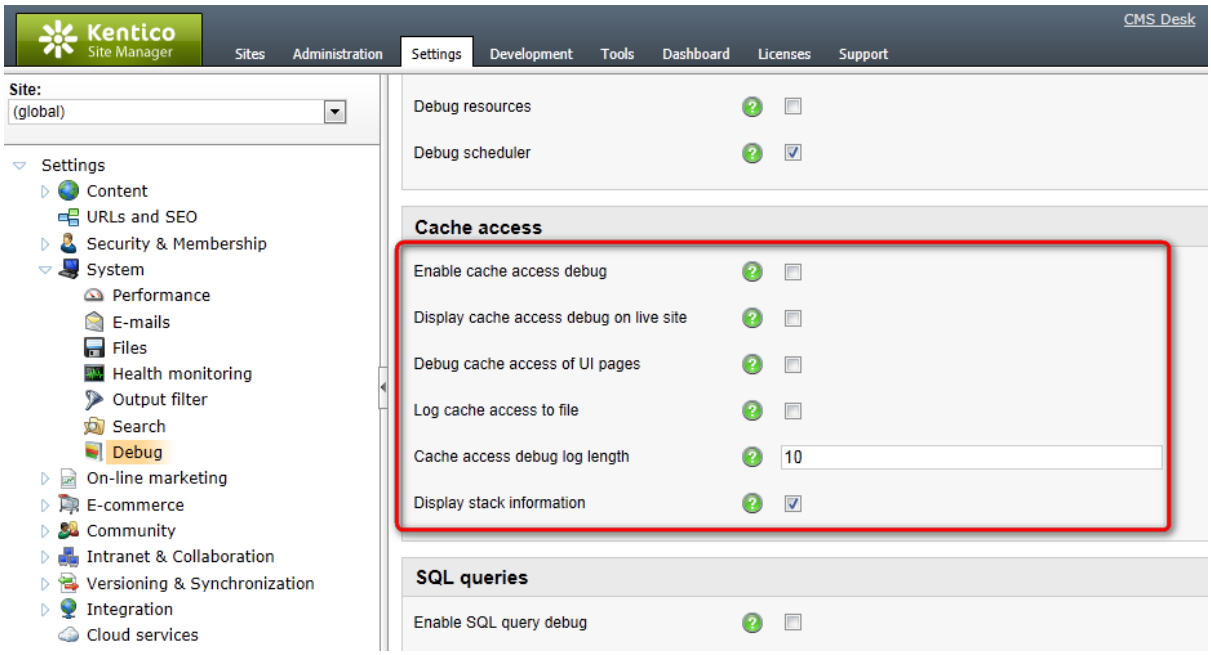
```
<add key="CMSMaxFinishedThreadsLogged" value="10" />
```

7.6.3.4 Cache access

Cache access debugging can be turned on and configured either by adjusting certain settings in **Site Manager -> Settings -> System -> Debug**, or by adding certain keys into the *AppSettings* section of your *web.config* file. The following table lists and explains these settings and keys:

Setting	Web.config key	Description
Enable cache access debug	CMSDebugCache	Enables cache access debugging and the **Cache access** tab in **Site Manager -> Administration -> System -> Debug**.
Display cache access debug on live site	CMSDebugCacheLive	If enabled, cache access debug information is also displayed at the bottom of each live site page. This option requires cache access debugging to be enabled.
Debug cache access of UI pages	CMSDebugAllCaches	If enabled, access to data cached for pages of the administration interface (CMS Desk and Site Manager) will also be included in the cache access debug. This option requires cache access debugging to be enabled.
Log cache access to file	CMSLogCache	If enabled, cache access debug log is saved into the *logcache.log* file in the ~\App_Data folder. This option does not require cache access debugging to

		be enabled.
Cache access debug log length	CMSDebugCacheLogLength	Sets the maximum length of the cache access debug log on the **Debug -> Cache access** tab, i.e. the number of requests for which debug information is preserved and displayed on the tab. If empty, value of the **Default log length** setting (or the **CMSDebugEverythingLogLength** key) is used.
Display stack information	CMSDebugCacheStack	If enabled, stack is tracked when debugging cache access and is displayed in the **Context** column. This information is only available in the debugging UI and on the live site, not in the debug log written into the *logcache.log* file.



It may happen that you specify different configuration in the settings and in the *web.config* file. In such cases, boolean settings (true/false) need to be enabled at least in one place (in *web.config* or in settings) in order to be enabled, while log lengths specified in **Site Manager -> Settings** have higher priority than log lengths specified in the web.config.

Here is a list of the keys for easy copy&paste into your *web.config*:

```
<add key="CMSDebugCache" value="true" />
<add key="CMSDebugCacheLive" value="true" />
<add key="CMSDebugAllCaches" value="true" />
<add key="CMSLogCache" value="true" />
<add key="CMSDebugCacheLogLength" value="10" />
<add key="CMSDebugCacheStack" value="true" />
```

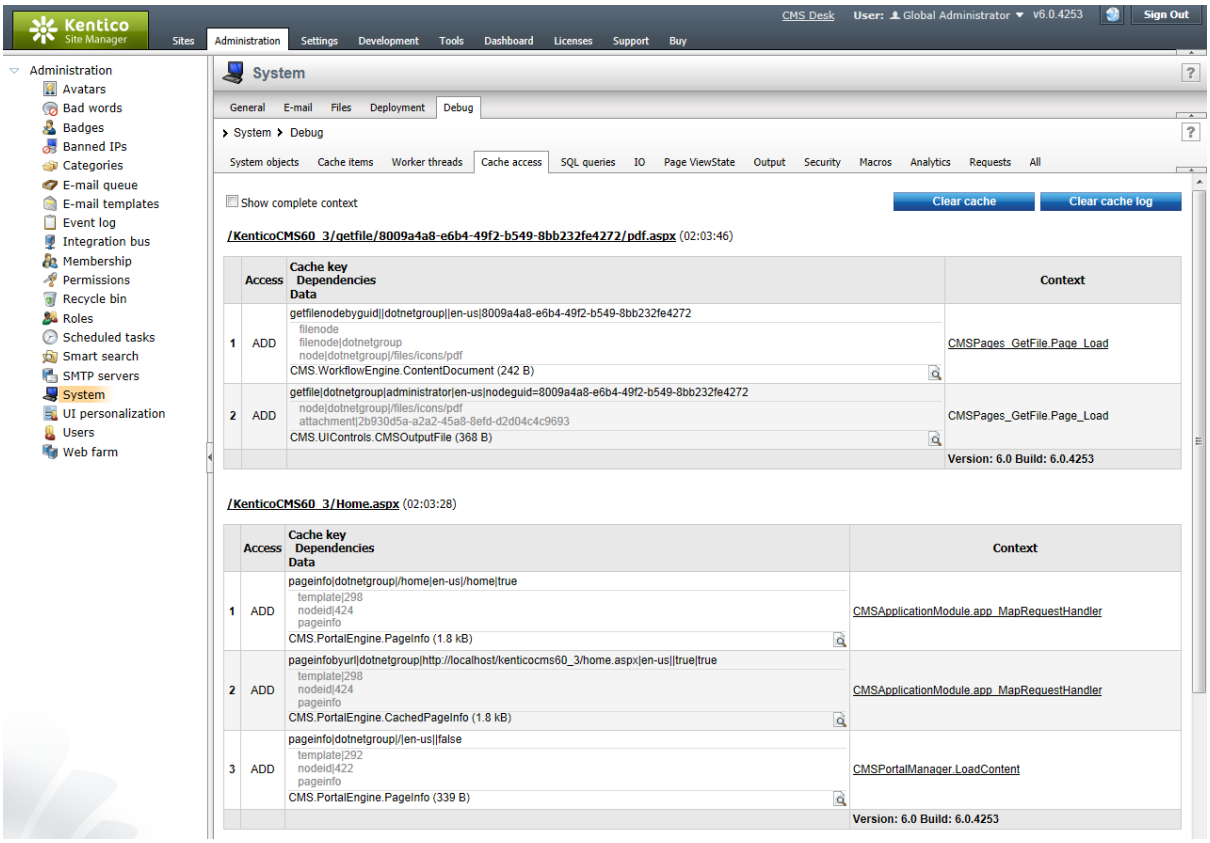
Cache access debugging can also be enabled using the general settings and keys.

User interface

On the **Debug -> Cache access** tab, you can see a log of requests that accessed the system cache.

For each request, you can see the URL and time when it was processed. The table below the URL always shows the type of **Access**, the accessed **Cache key**, its cache dependencies, the object type and size of cached **Data** and the **Context** from which the cache was accessed. If you enable the **Show complete context** option, complete context of the cache access, i.e. not only the method that accessed the cache item, but also the methods from which the first one was called, will be shown in the **Context** column.

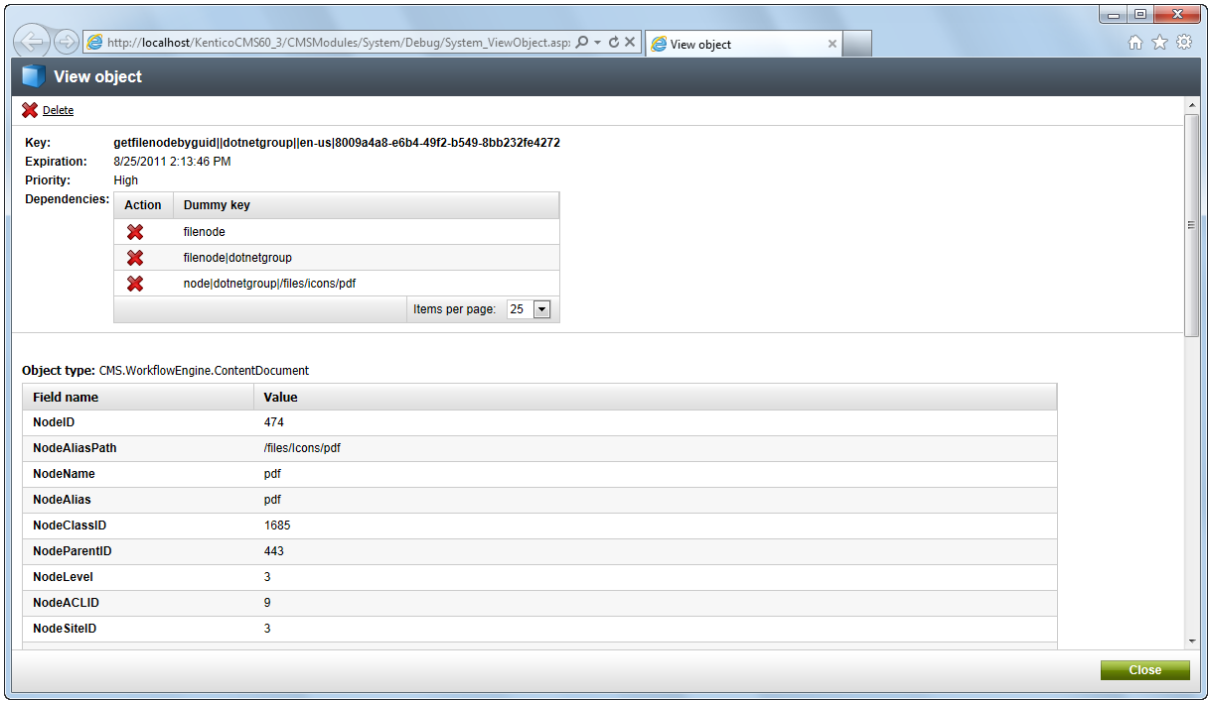
Enabling the **Show complete context** check-box displays complete context (not only the topmost item) in the **Context** column. All dummy keys can be cleared from the cache using the **Clear cache** button. Clicking the **Clear cache log** button clears all records in the log.



If you click the **View object** ( ) icon in a particular row, a pop-up window gets opened as in the screenshot below. There, you can see detailed information about the cached object:

- **Key** - the key under which the object is stored in the cache.
- **Expiration** - date and time when the cache item will expire (i.e. will be removed from the cache).
- **Priority** - priority of the cache item. The same as ASP.NET cache item priorities, while only *High* and *NotRemovable* are used in Kentico CMS.
- **Dependencies** - dummy keys on which the cache item is enabled.
- **Object type** - type of the cached object.

Finally, the table under the **Object type** value represents the actual data of the cached object. Displayed fields vary based on the currently displayed object type. The cache item can be cleared from cache using the **Delete** button in the top part of the window.



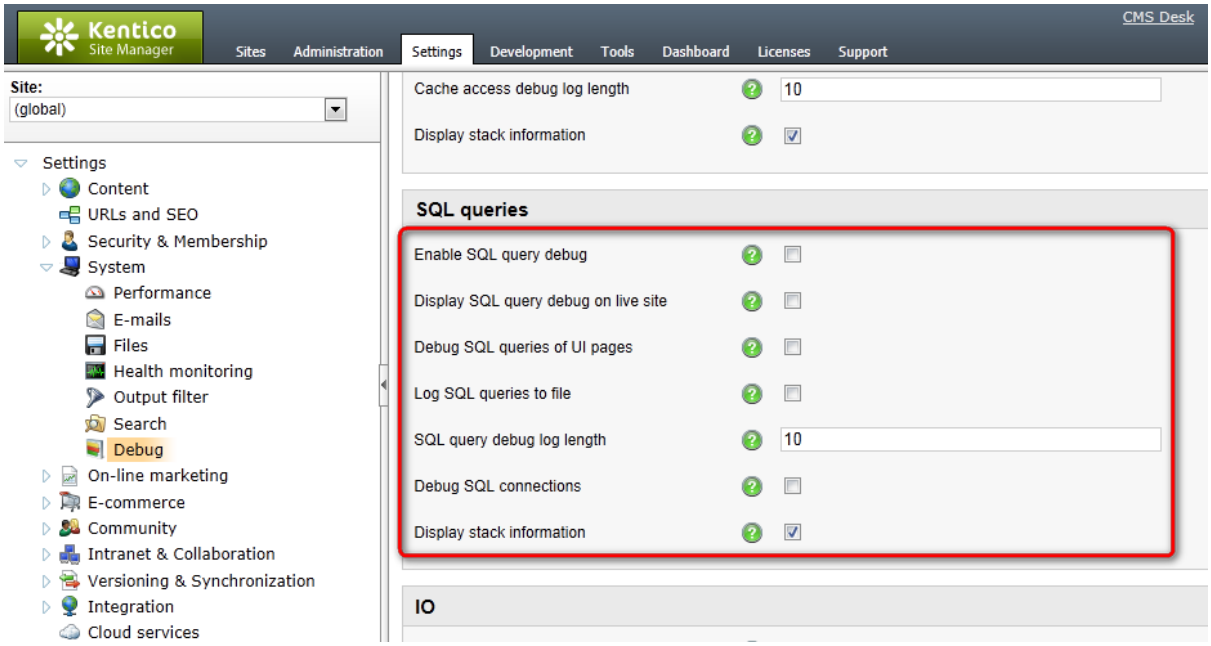
For more information about caching in Kentico CMS, please refer to Development -> Caching and performance -> Caching options.

7.6.3.5 SQL queries

SQL query debugging can be turned on and configured either by adjusting certain settings in **Site Manager -> Settings -> System -> Debug**, or by adding certain keys into the *AppSettings* section of your *web.config* file. The following table lists and explains these settings and keys:

Setting	Web.config key	Description
Enable SQL query debug	CMSDebugSQLQueries	Enables SQL query debugging and the **SQL queries** tab in **Site Manager -> Administration -> System -> Debug**.
Display SQL query debug on live site	CMSDebugSQLQueriesLive	If enabled, SQL query debug information is also displayed at the bottom of each live site page. This option requires SQL query debugging to be enabled.
Debug SQL queries of UI pages	CMSDebugAllSQLQueries	If enabled, SQL queries called by pages of the administration interface (CMS Desk and Site Manager) will also be included in the SQL query debug. This option requires SQL query debugging to be enabled.

Log SQL queries to file	CMSLogSQLQueries	If enabled, SQL query debug log is saved into the *logsql.log* file in the ~*App_Data* folder. This option does not require SQL query debugging to be enabled.
SQL query debug log length	CMSDebugSQLQueriesLogLength	Sets the maximum length of the SQL query debug log on the **Debug -> SQL queries** tab, i.e. the number of requests for which debug information is preserved and displayed on the tab. If empty, value of the **Default log length** setting (or the **CMSDebugEverythingLogLength** key) is used.
Debug SQL connections	CMSDebugSQLConnections	If enabled, SQL connection operations (new, open, close) are logged in the SQL query debug log.
Display stack information	CMSDebugSQLQueriesStack	If enabled, stack is tracked when debugging SQL queries and is displayed in the **Context** column. This information is only available in the debugging UI and on the live site, not in the debug log written into the *logsql.log* file.



It may happen that you specify different configuration in the settings and in the *web.config* file. In such cases, boolean settings (true/false) need to be enabled at least in one place (in *web.config* or in settings) in order to be enabled, while log lengths specified in **Site Manager -> Settings** have higher priority than log lengths specified in the web.config.

Here is a list of the keys for easy copy&paste into your *web.config*:

```
<add key="CMSDebugSQLQueries" value="true" />
<add key="CMSDebugSQLQueriesLive" value="true" />
<add key="CMSDebugAllSQLQueries" value="true" />
```

```
<add key="CMSLogSQLQueries" value="true" />
<add key="CMSDebugSQLQueriesLogLength" value="10" />
<add key="CMSDebugSQLConnections" value="true" />
<add key="CMSDebugSQLQueriesStack" value="true" />
```

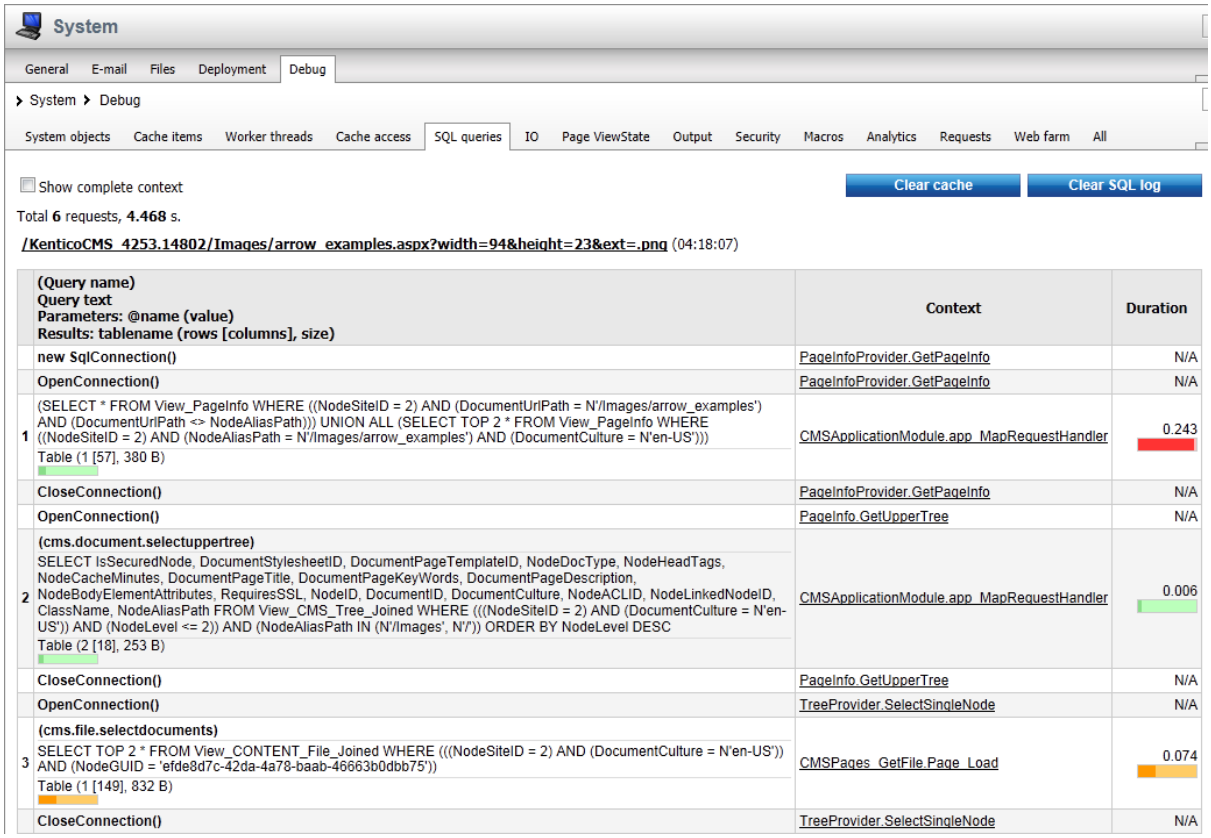
SQL query debugging can also be enabled using the general settings and keys.

User interface

On the **Debug -> SQL queries** tab, you can see a log of the SQL queries called when particular data is loaded. Each record contains the request used to access the given item and the time when it was loaded. Below it, you can find a table listing all SQL queries executed within the context of the request.

For each query, there are two lines in the table. The first line contains the exact text of the query and the second shows the number of loaded rows, columns in each row and the total size of the loaded data. The **Context** column shows the context where the query was called from. The last column of the table displays the exact duration of query execution. The last line of the table displays the total volume of data and loading time for all queries under the given request.

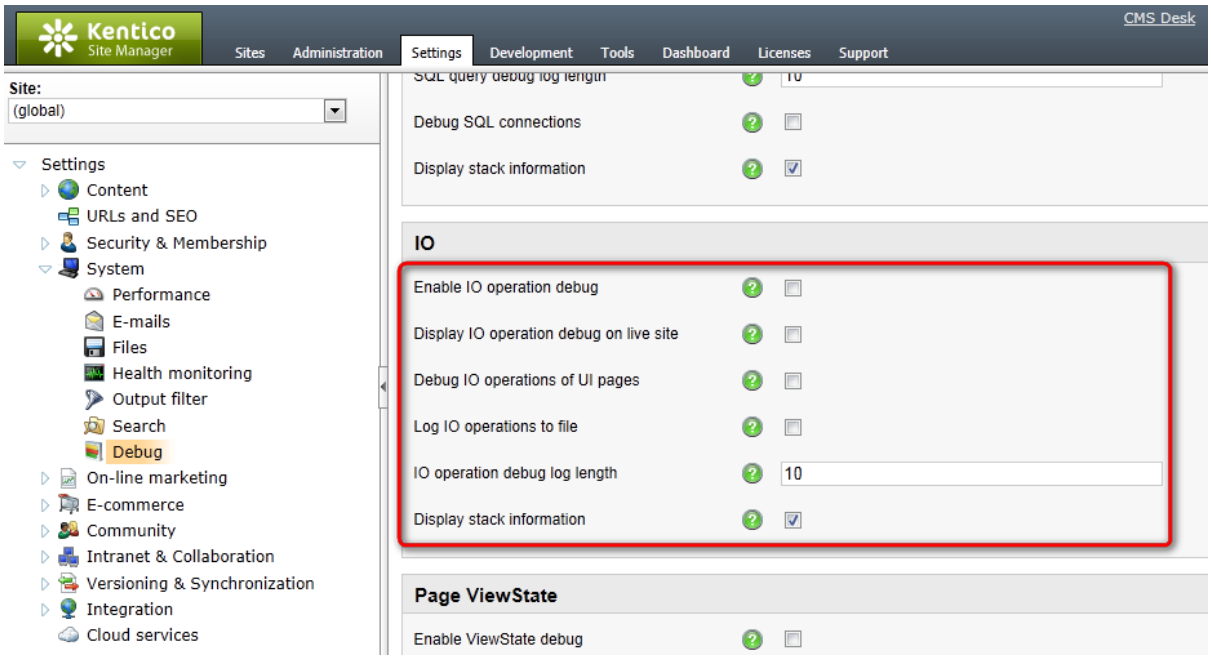
Enabling the **Show complete context** check-box displays complete context (not only the topmost item) in the **Context** column. Clicking the **Clear SQL log** button clears all records in this debug log.



7.6.3.6 IO

IO operation debugging can be turned on and configured either by adjusting certain settings in **Site Manager -> Settings -> System -> Debug**, or by adding certain keys into the *AppSettings* section of your *web.config* file. The following table lists and explains these settings and keys:

Setting	Web.config key	Description
Enable IO operation debug	CMSDebugFiles	Enables IO operation debugging and the **IO** tab in **Site Manager -> Administration -> System -> Debug**.
Display IO operation debug on live site	CMSDebugFilesLive	If enabled, IO operation debug information is also displayed at the bottom of each live site page. This option requires IO operation debugging to be enabled.
Debug IO operations of UI pages	CMSDebugAllFiles	If enabled, IO operations called by pages of the administration interface (CMS Desk and Site Manager) will also be included in the IO operation debug. This option requires IO operation debugging to be enabled.
Log IO operations to file	CMSLogFiles	If enabled, IO operation debug log is saved into the *logfiles.log* file in the *~\App_Data* folder. This option does not require IO operation debugging to be enabled.
IO operation debug log length	CMSDebugFilesLogLength	Sets the maximum length of the IO operation debug log on the **Debug -> IO** tab, i.e. the number of requests for which debug information is preserved and displayed on the tab. If empty, value of the **Default log length** setting (or the **CMSDebugEverythingLogLength** key) is used.
Display stack information	CMSDebugFilesStack	If enabled, stack is tracked when debugging IO operations and is displayed in the **Context** column. This information is only available in the debugging UI and on the live site, not in the debug log written into the *logfiles.log* file.



It may happen that you specify different configuration in the settings and in the *web.config* file. In such cases, boolean settings (true/false) need to be enabled at least in one place (in *web.config* or in settings) in order to be enabled, while log lengths specified in **Site Manager -> Settings** have higher priority than log lengths specified in the web.config.

Here is a list of the keys for easy copy&paste into your *web.config*:

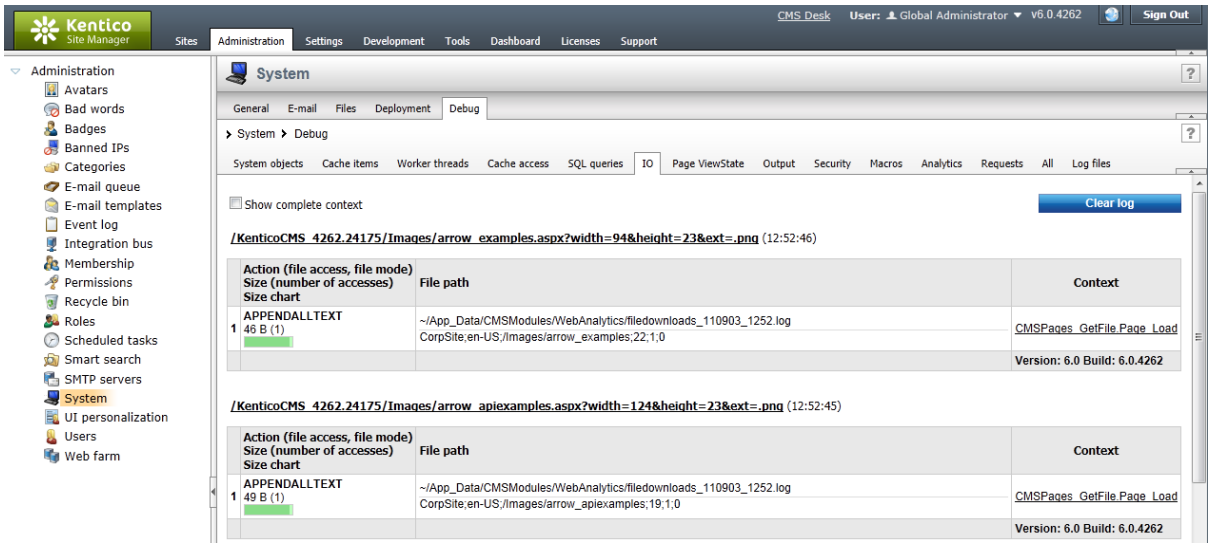
```
<add key="CMSDebugFiles" value="true" />
<add key="CMSDebugFilesLive" value="true" />
<add key="CMSDebugAllFiles" value="true" />
<add key="CMSLogFiles" value="true" />
<add key="CMSDebugFilesLogLength" value="10" />
<add key="CMSDebugFilesStack" value="true" />
```

IO operation debugging can also be enabled using the general settings and keys.

User interface

On the **Debug -> IO** tab, you can see which file operations were carried out on each request. For each operation, you can see its type, size of transferred data, number of accesses and the path to the accessed file. In the **Context** column, you can see the context from which the file access operation was called.

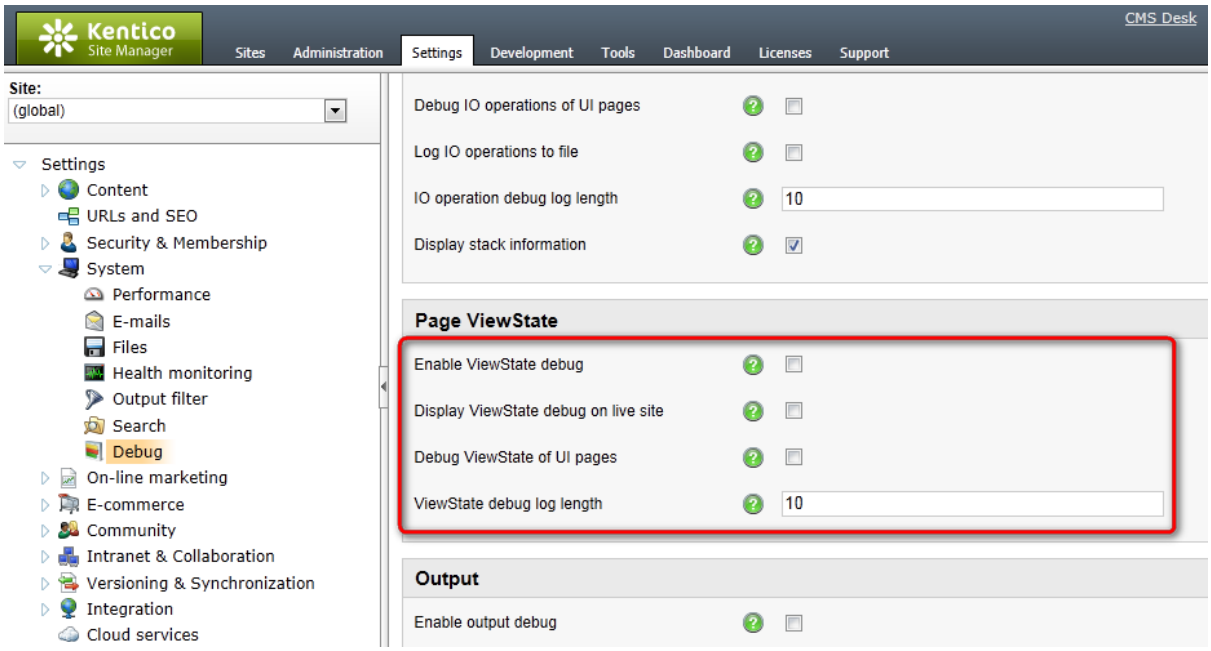
Enabling the **Show complete context** check-box displays complete context (not only the topmost item) in the **Context** column. The log can be cleared using the **Clear log** button.



7.6.3.7 Page ViewState

Page ViewState debugging can be turned on and configured either by adjusting certain settings in **Site Manager -> Settings -> System -> Debug**, or by adding certain keys into the *AppSettings* section of your *web.config* file. The following table lists and explains these settings and keys:

Setting	Web.config key	Description
Enable ViewState debug	CMSDebugViewState	Enables ViewState debugging and the **Page ViewState** tab in **Site Manager -> Administration -> System -> Debug**.
Display ViewState debug on live site	CMSDebugViewStateLive	If enabled, ViewState debug information is also displayed at the bottom of each live site page. This option requires ViewState debugging to be enabled.
Debug ViewState of UI pages	CMSDebugAllViewStates	If enabled, ViewState of administration interface pages (CMS Desk and Site Manager) will also be included in the ViewState debug. This option requires ViewState debugging to be enabled.
ViewState debug log length	CMSDebugViewStateLogLength	Sets the maximum length of the ViewState debug log on the **Debug -> ViewState** tab, i.e. the number of requests for which debug information is preserved and displayed on the tab. If empty, value of the **Default log length** setting (or the **CMSDebugEverythingLogLength** key) is used.



It may happen that you specify different configuration in the settings and in the *web.config* file. In such cases, boolean settings (true/false) need to be enabled at least in one place (in *web.config* or in settings) in order to be enabled, while log lengths specified in **Site Manager -> Settings** have higher priority than log lengths specified in the web.config.

Here is a list of these keys for easy copy&paste into your *web.config*:

```
<add key="CMSDebugViewState" value="true" />
<add key="CMSDebugViewStateLive" value="true" />
<add key="CMSDebugAllViewStates" value="true" />
<add key="CMSDebugViewStateLogLength" value="10" />
```

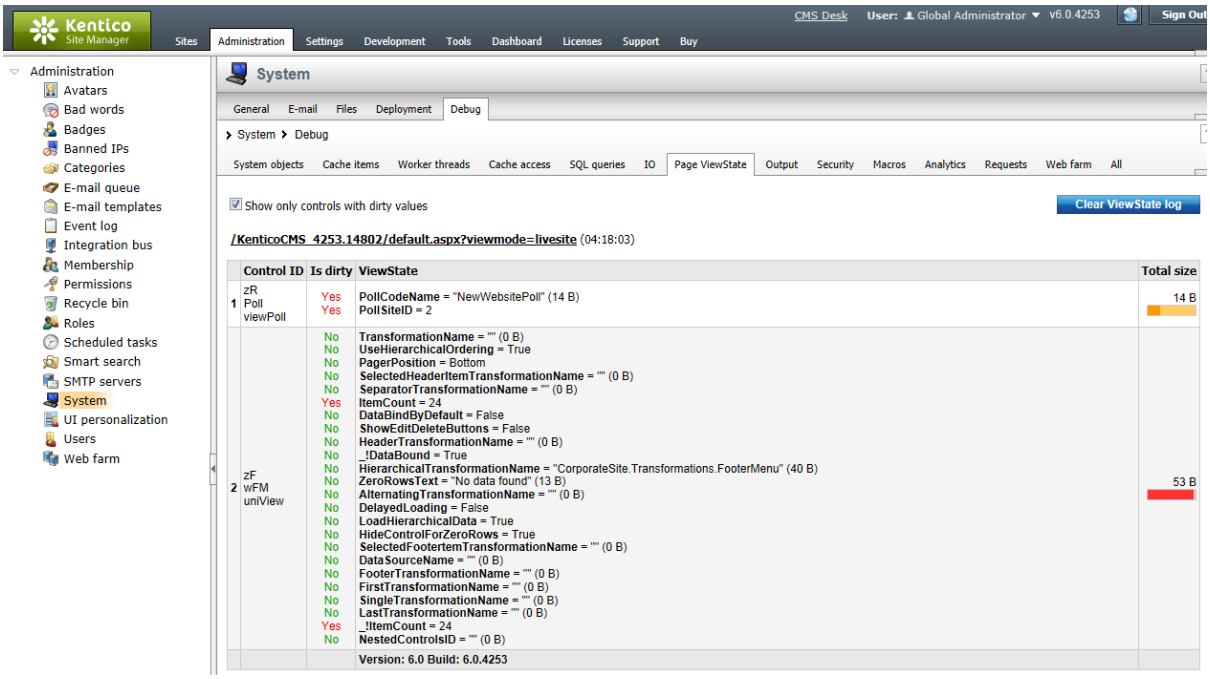
Page ViewState debugging can also be enabled using the general settings and keys.

User interface

On the **Debug -> Page ViewState** tab, you can see the ViewState of particular controls on recently loaded pages. In the **Control ID** column, you can see the ID of each control on the requested page. The **Is dirty** column indicates if the item was added to the ViewState after the *TrackViewState()* method was called (typically occurs during *OnInit*). You can display only controls with such items by enabling the **Show only controls with dirty values** check-box above the grid. The **ViewState** column displays the control's ViewState data and the **Total size** column contains information about the total size of the control's ViewState data.

The log can be cleared using the **Clear ViewState log** button.

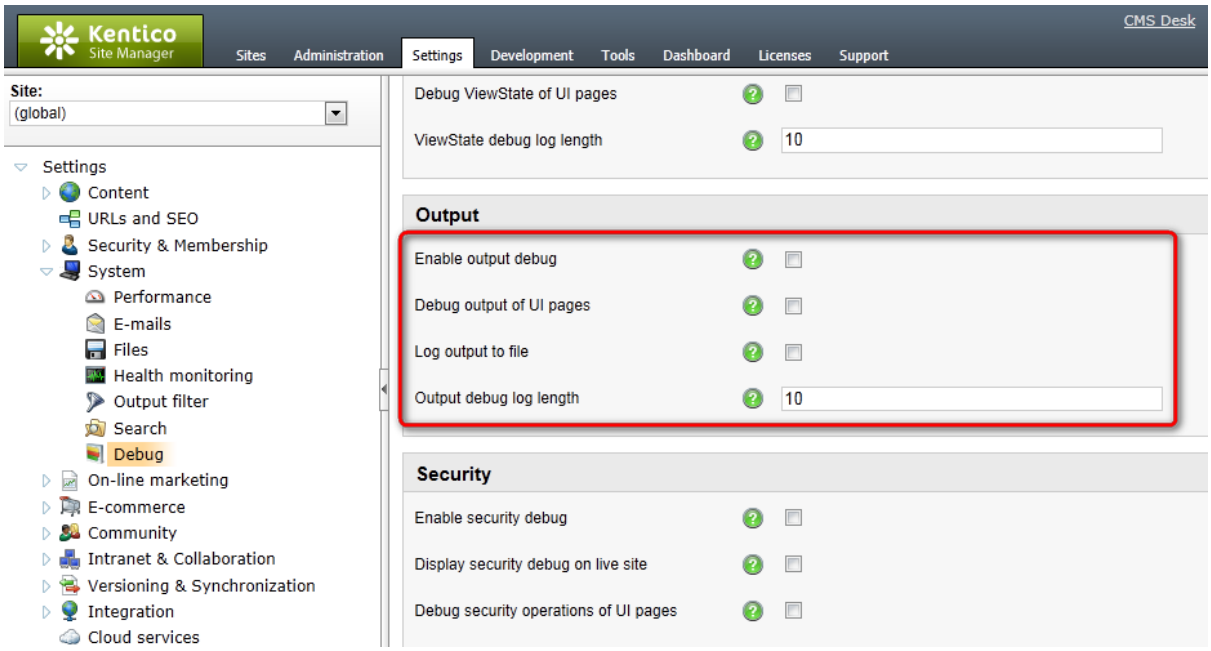
Please note: ViewState of the controls is retrieved using reflection. Therefore this debug may not work properly in a medium trust environment or other specific circumstances.



7.6.3.8 Output

Output debugging can be turned on and configured either by adjusting certain settings in **Site Manager -> Settings -> System -> Debug**, or by adding certain keys into the *AppSettings* section of your *web.config* file. The following table lists and explains these settings and keys:

Setting	Web.config key	Description
Enable output debug	CMSDebugOutput	Enables page output debugging and the **Output** tab in **Site Manager -> Administration -> System -> Debug**.
Debug output of UI pages	CMSDebugAllOutputs	If enabled, output of administration interface pages (CMS Desk and Site Manager) will also be included in the output debug. This option requires output debugging to be enabled.
Log output to file	CMSLogOutputToFile	If enabled, output debug log is saved into the *logOutput.log* file in the *~\App_Data* folder. This option does not require output debugging to be enabled.
Output debug log length	CMSDebugOutputToFile	Sets the maximum length of the output debug log on the **Debug -> Output** tab, i.e. the number of requests for which debug information is preserved and displayed on the tab. If empty, value of the **Default log length** setting (or the **CMSDebugEverythingLogLength** key) is used.



It may happen that you specify different configuration in the settings and in the *web.config* file. In such cases, boolean settings (true/false) need to be enabled at least in one place (in *web.config* or in settings) in order to be enabled, while log lengths specified in **Site Manager -> Settings** have higher priority than log lengths specified in the web.config.

Here is a list of the keys for easy copy&paste into your *web.config*:

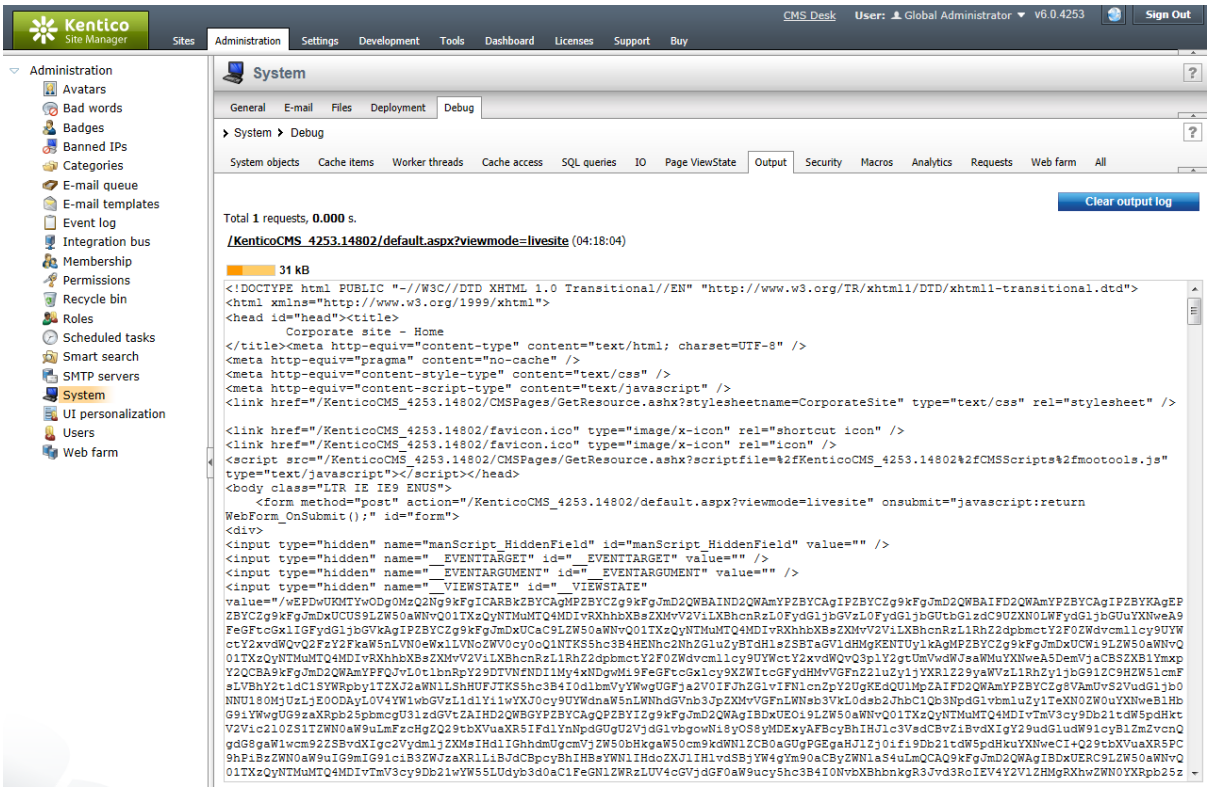
```
<add key="CMSDebugOutput" value="true" />
<add key="CMSDebugAllOutputs" value="true" />
<add key="CMSLogOutputToFile" value="true" />
<add key="CMSDebugOutputLogLength" value="10" />
```

Output debugging can also be enabled using the general settings and keys.

User interface

On the **Debug -> Output** tab, you can see the exact output code of recently displayed pages. This is particularly useful in case of AJAX request, whose code cannot be viewed as part of the page source directly in the browser.

The log can be cleared using the **Clear output log** button.

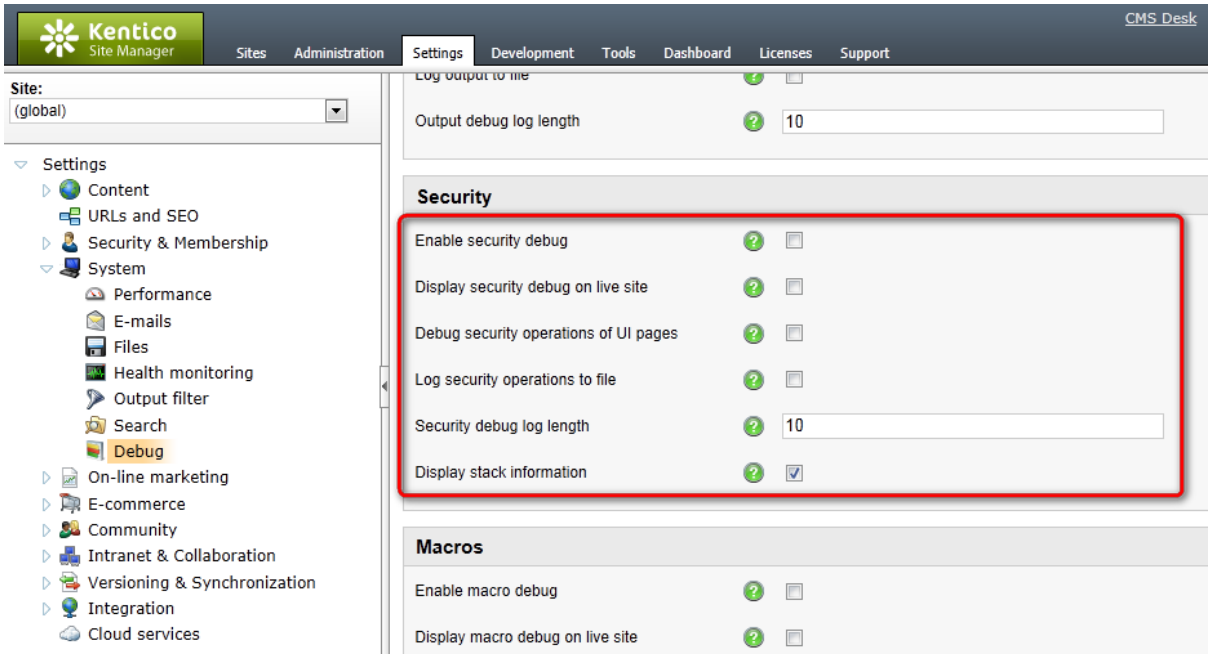


7.6.3.9 Security

Security debugging can be turned on and configured either by adjusting certain settings in **Site Manager -> Settings -> System -> Debug**, or by adding certain keys into the *AppSettings* section of your *web.config* file. The following table lists and explains these settings and keys:

Setting	Web.config key	Description
Enable security debug	CMSDebugSecurity	Enables security operation debugging and the **Security** tab in **Site Manager -> Administration -> System -> Debug**.
Display security debug on live site	CMSDebugSecurityLive	If enabled, security operation debug information is also displayed at the bottom of each live site page. This option requires security debugging to be enabled.
Debug security operations of UI pages	CMSDebugAllSecurity	If enabled, security checks performed by pages of the administration interface (CMS Desk and Site Manager) will also be included in the security debug. This option requires security debugging to be enabled.
Log security operations to file	CMSLogSecurity	If enabled, security debug log is saved into the *logSecurity.log* file in the *~\App_Data* folder. This option does not require security debugging to be enabled.

Security debug log length	CMSDebugSecurityLogLength	Sets the maximum length of the security debug log on the **Debug -> Security** tab, i.e. the number of requests for which debug information is preserved and displayed on the tab. If empty, value of the **Default log length** setting (or the **CMSDebugEverythingLogLength** key) is used.
Display stack information	CMSDebugSecurityStack	If enabled, stack is tracked when debugging security and is displayed in the **Context** column. This information is only available in the debugging UI and on the live site, not in the debug log written into the *logSecurity.log* file.



It may happen that you specify different configuration in the settings and in the *web.config* file. In such cases, boolean settings (true/false) need to be enabled at least in one place (in *web.config* or in settings) in order to be enabled, while log lengths specified in **Site Manager -> Settings** have higher priority than log lengths specified in the web.config.

Here is a list of the keys for easy copy&paste into your *web.config*:

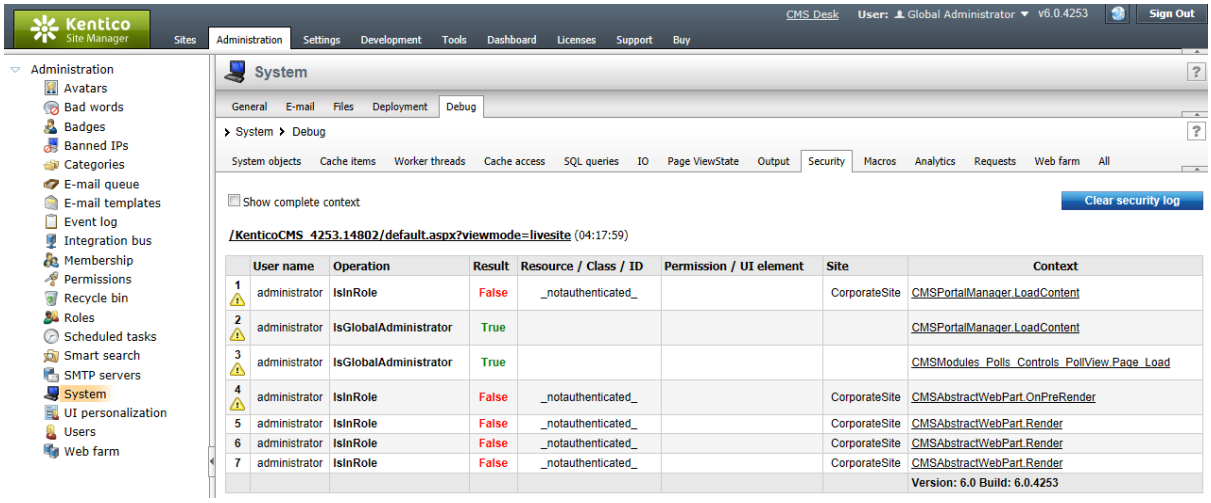
```
<add key="CMSDebugSecurity" value="true" />
<add key="CMSDebugSecurityLive" value="true" />
<add key="CMSDebugAllSecurity" value="true" />
<add key="CMSLogSecurity" value="true" />
<add key="CMSDebugSecurityLogLength" value="10" />
<add key="CMSDebugSecurityStack" value="true" />
```

Security debugging can also be enabled using the general settings and keys.

User interface

On the **Debug -> Security** tab, you can see which security checks were recently performed on the site. This is particularly useful if you want to quickly find out why some user is not able to access some section of the UI or gets the *Access denied* page displayed.

Enabling the **Show complete context** check-box displays complete context (not only the topmost item) in the **Context** column. The log can be cleared using the **Clear security log** button.

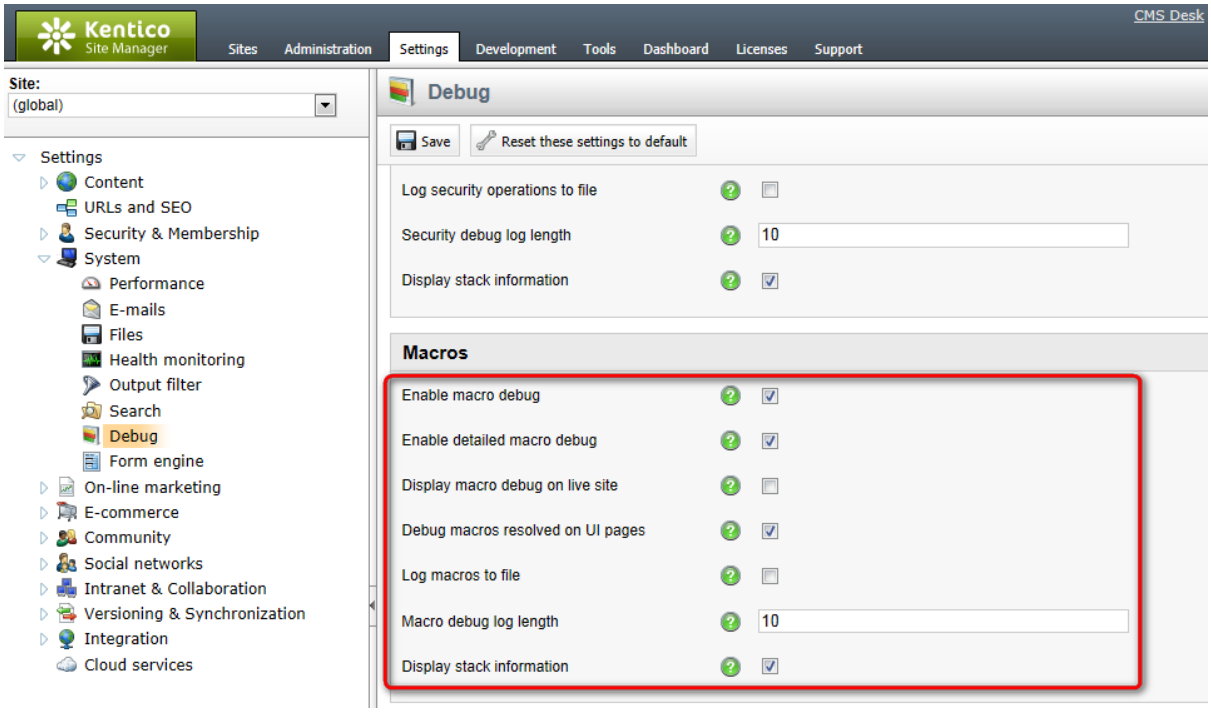


7.6.3.10 Macros

Macro debugging can be turned on and configured either by adjusting certain settings in **Site Manager -> Settings -> System -> Debug**, or by adding certain keys into the *AppSettings* section of your *web.config* file. The following table lists and explains these settings and keys:

Setting	Web.config key	Description
Enable macro debug	CMSDebugMacros	Enables macro debugging and the **Macros** tab in **Site Manager -> Administration -> System -> Debug**.
Enable detailed macro debug	N/A	If enabled, the macro debug will additionally display the results of all sub-elements used within macro expressions. This allows you to check the exact data content of a macro's components during each step of the resolving process. Detailed macro debugging is highly recommended if you encounter problems with complex expressions. If disabled, only the final result of each macro will be shown in the debug. The detailed debug may also be enabled only for specific expressions by adding the **|(debug)** parameter to the given macro.

Display macro debug on live site	CMSDebugMacrosLive	If enabled, macro debug information is also displayed at the bottom of each live site page. This option requires macro debugging to be enabled.
Debug macros resolved on UI pages	CMSDebugAllMacros	If enabled, macros resolved on pages of the administration interface (CMS Desk and Site Manager) will also be included in the macro debug. This option requires macro debugging to be enabled.
Log macros to file	CMSLogMacros	If enabled, macro debug log is saved into the *logmacros.log* file in the ~*App_Data* folder. This option does not require macro debugging to be enabled.
Macro debug log length	CMSDebugMacrosLogLength	Sets the maximum length of the macro debug log on the **Debug -> Macros** tab, i.e. the number of requests for which debug information is preserved and displayed on the tab. If empty, value of the **Default log length** setting (or the **CMSDebugEverythingLogLength** key) is used.
Display stack information	CMSDebugMacrosStack	If enabled, stack is tracked when debugging cache macros and is displayed in the **Context** column. This information is only available in the debugging UI and on the live site, not in the debug log written into the *logmacros.log* file.



It may happen that you specify different configuration in the settings and in the *web.config* file. In such cases, boolean settings (true/false) need to be enabled at least in one place (in *web.config* or in settings) in order to be enabled, while log lengths specified in **Site Manager -> Settings** have higher priority than log lengths specified in the web.config.

Here is a list of these keys for easy copy&paste into your *web.config*:

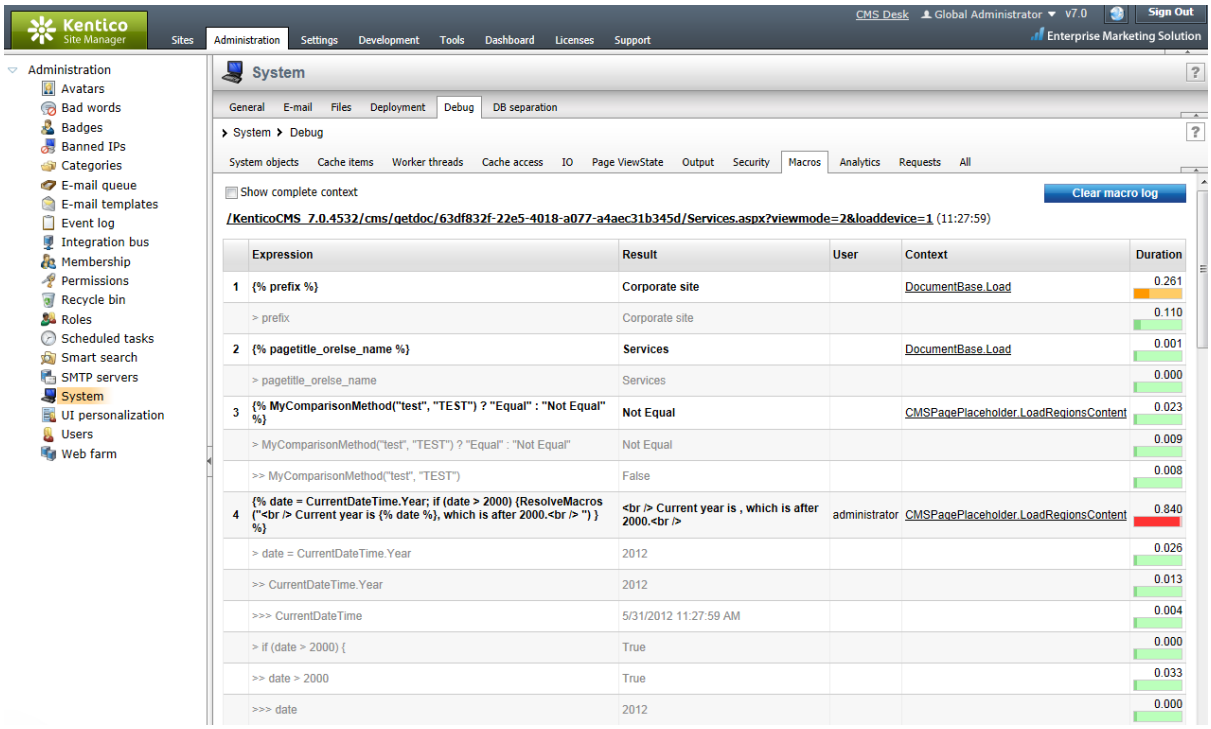
```
<add key="CMSDebugMacros" value="true" />
<add key="CMSDebugMacrosLive" value="true" />
<add key="CMSDebugAllMacros" value="true" />
<add key="CMSLogMacros" value="true" />
<add key="CMSDebugMacrosLogLength" value="20" />
<add key="CMSDebugMacrosStack" value="true" />
```

Macro debugging can also be enabled using the general settings and keys.

User interface

On the **Debug -> Macros** tab, you can see which macros were recently resolved. For each macro, the log displays the exact expression, the result (the value into which it was resolved) and the context from which it was called. If detailed macro debugging is enabled, the results will also be displayed for individual sub-elements that make up each macro expression. This is useful if you want to analyze how macros were processed.

Enabling the **Show complete context** check-box displays the complete context (not only the topmost item) in the **Context** column. The log can be cleared using the **Clear macro log** button.

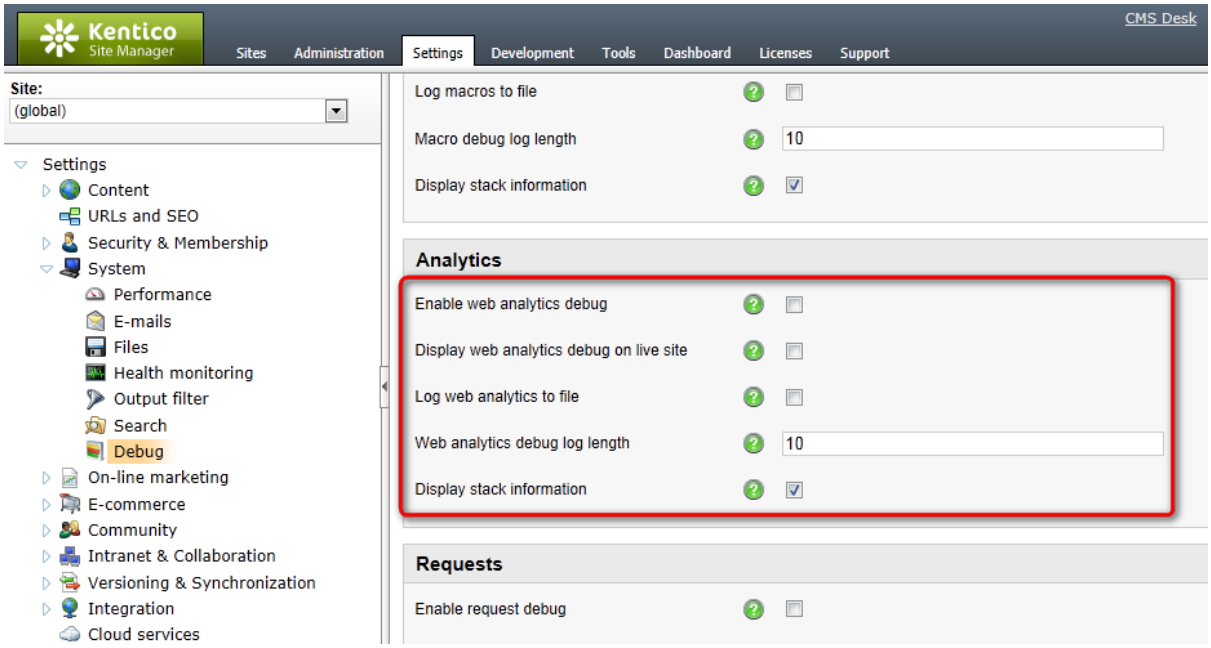


7.6.3.11 Analytics

Web analytics debugging can be turned on and configured either by adjusting certain settings in **Site Manager -> Settings -> System -> Debug**, or by adding certain keys into the *AppSettings* section of

your *web.config* file. The following table lists and explains these settings and keys:

Setting	Web.config key	Description
Enable web analytics debug	CMSDebugAnalytics	Enables web analytics debugging and the **Analytics** tab in **Site Manager -> Administration -> System -> Debug**.
Display web analytics debug on live site	CMSDebugAnalytics Live	If enabled, web analytics debug information is also displayed at the bottom of each live site page. This option requires web analytics debugging to be enabled.
Log web analytics to file	CMSLogAnalytics	If enabled, web analytics debug log is saved into the *loganalytics.log* file in the *~\App_Data* folder. This option does not require web analytics debugging to be enabled.
Web analytics debug log length	CMSDebugAnalytics LogLength	Sets the maximum length of the web analytics debug log on the **Debug -> Analytics** tab, i.e. the number of requests for which debug information is preserved and displayed on the tab. If empty, value of the **Default log length** setting (or the **CMSDebugEverythingLogLength** key) is used.
Display stack information	CMSDebugAnalytics Stack	If enabled, stack is tracked when debugging web analytics and is displayed in the **Context** column. This information is only available in the debugging UI and on the live site, not in the debug log written into the *loganalytics.log* file.



It may happen that you specify different configuration in the settings and in the *web.config* file. In such cases, boolean settings (true/false) need to be enabled at least in one place (in *web.config* or in

settings) in order to be enabled, while log lengths specified in **Site Manager -> Settings** have higher priority than log lengths specified in the web.config.

Here is a list of the keys for easy copy&paste into your *web.config*:

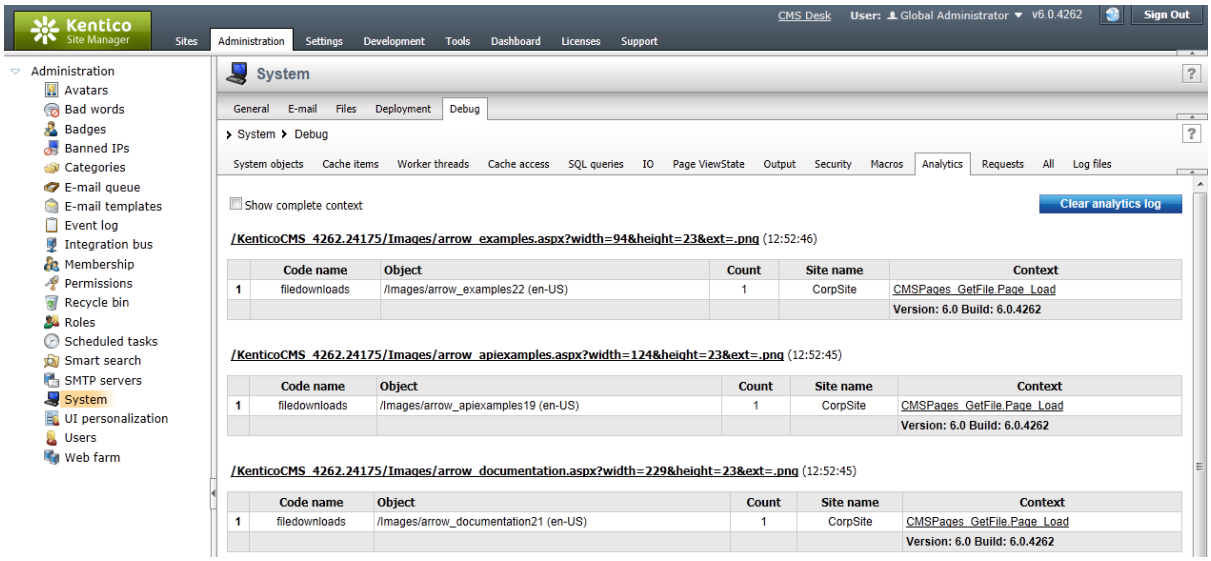
```
<add key="CMSDebugAnalytics" value="true" />
<add key="CMSDebugAnalyticsLive" value="true" />
<add key="CMSLogAnalytics" value="true" />
<add key="CMSDebugAnalyticsLogLength" value="10" />
<add key="CMSDebugAnalyticsStack" value="true" />
```

IO operation debugging can also be enabled using the general settings and keys.

User interface

On the **Analytics** tab, you can see which statistics were logged by the Web analytics module for individual requests. For each logged item, you can see the code name of the logged operation, the object to which the operation relates, number of the performed operations and the name of the site where the operation was performed. In the **Context** column, you can see the context from which the operation was logged.

Enabling the **Show complete context** check-box displays complete context (not only the topmost item) in the **Context** column. The log can be cleared using the **Clear analytics log** button.

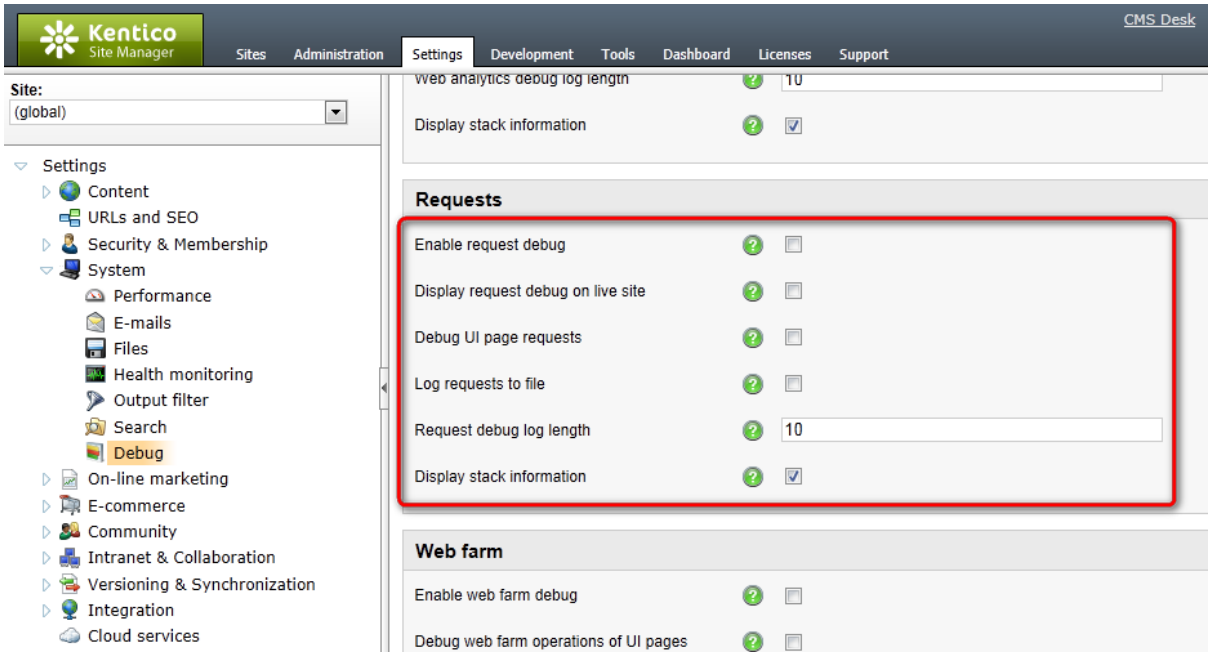


7.6.3.12 Requests

Request debugging can be turned on and configured either by adjusting certain settings in **Site Manager -> Settings -> System -> Debug**, or by adding certain keys into the *AppSettings* section of your *web.config* file. The following table lists and explains these settings and keys:

Setting	Web.config key	Description

Enable request debug	CMSDebugRequests	Enables request debugging and the **Requests** tab in **Site Manager -> Administration -> System -> Debug**.
Display request debug on live site	CMSDebugRequestsLive	If enabled, request debug information is also displayed at the bottom of each live site page. This option requires request debugging to be enabled.
Debug UI page requests	CMSDebugAllRequests	If enabled, administration interface (CMS Desk and Site Manager) page requests will also be included in the macro debug. This option requires request debugging to be enabled.
Log requests to file	CMSLogRequests	If enabled, request debug log is saved into the *logRequests.log* and *logRequestsUrls.log* files in the *~\App_Data* folder. The first file contains the full log, while the second one only lists requested URLs. This option does not require request debugging to be enabled.
Request debug log length	CMSDebugRequestsLogLength	Sets the maximum length of the request debug log on the **Debug -> Requests** tab, i.e. the number of requests for which debug information is preserved and displayed on the tab. If empty, value of the **Default log length** setting (or the **CMSDebugEverythingLogLength** key) is used.
Display stack information	CMSDebugRequestsStack	If enabled, stack is tracked when debugging requests and is displayed in the **Context** column. This information is only available in the debugging UI and on the live site, not in the debug log written into the *logRequests.log* file.



It may happen that you specify different configuration in the settings and in the *web.config* file. In such cases, boolean settings (true/false) need to be enabled at least in one place (in *web.config* or in settings) in order to be enabled, while log lengths specified in **Site Manager -> Settings** have higher priority than log lengths specified in the web.config.

Here is a list of the keys for easy copy&paste into your *web.config*:

```
<add key="CMSDebugRequests" value="true" />
<add key="CMSDebugRequestsLive" value="true" />
<add key="CMSDebugAllRequests" value="true" />
<add key="CMSLogRequests" value="true" />
<add key="CMSDebugRequestsLogLength" value="5" />
<add key="CMSDebugRequestsStack" value="true" />
```

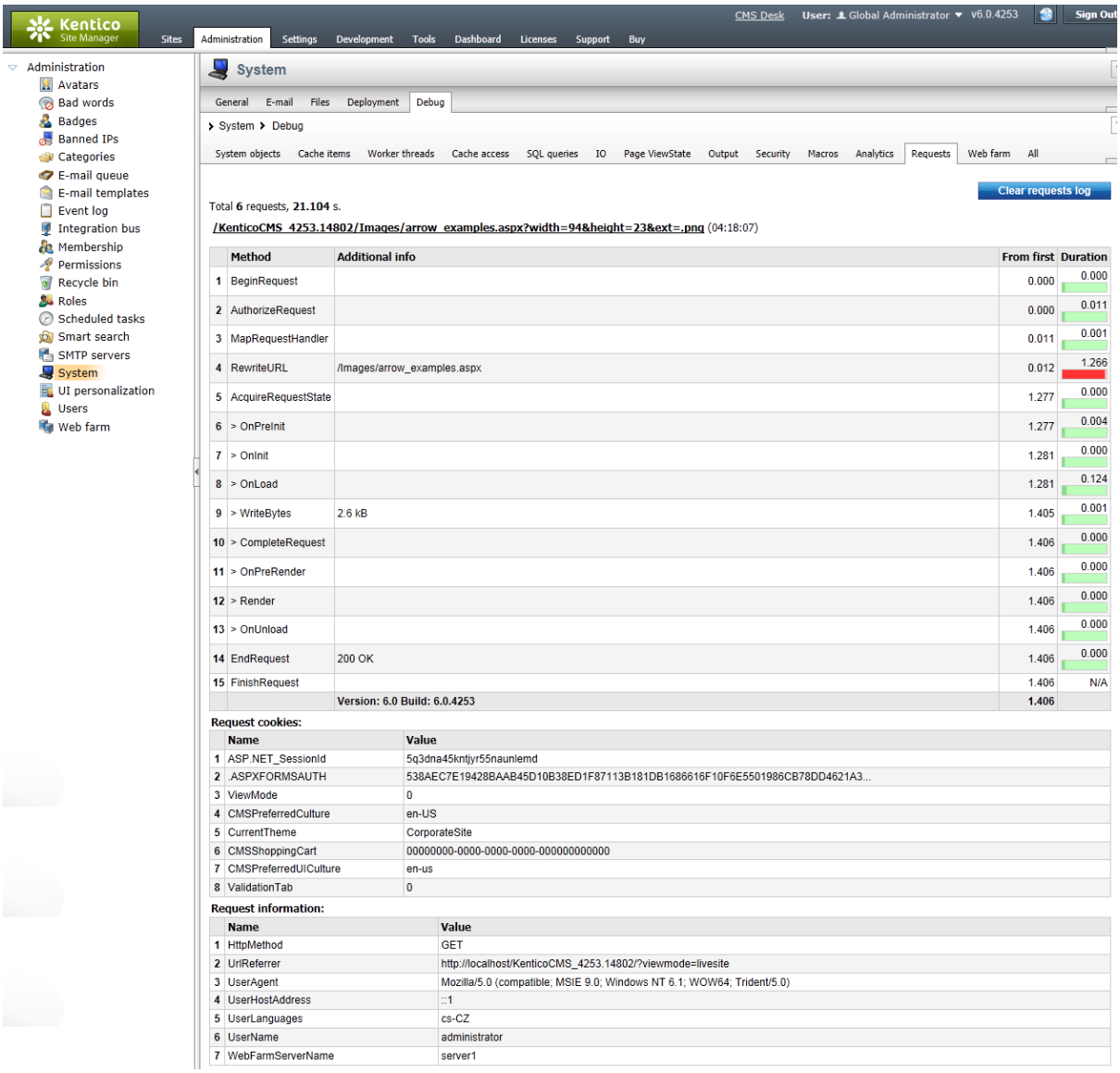
Request debugging can also be enabled using the general settings and keys.

User interface

On the **Debug -> Requests** tab, you can see the time each of the recent page requests took to process. You not only see the overall time, but also separate times of particular parts of each requests, along with other detailed information about the request.

This is particularly useful if your response time is too large and you need to figure out why. You can basically see whether the issue is outside the application or inside it by comparing the real response time and response time spent in the application, what data came with the request and from which context.

Enabling the **Show complete context** check-box displays complete context (not only the topmost item) in the **Context** column. The log can be cleared using the **Clear request log** button.

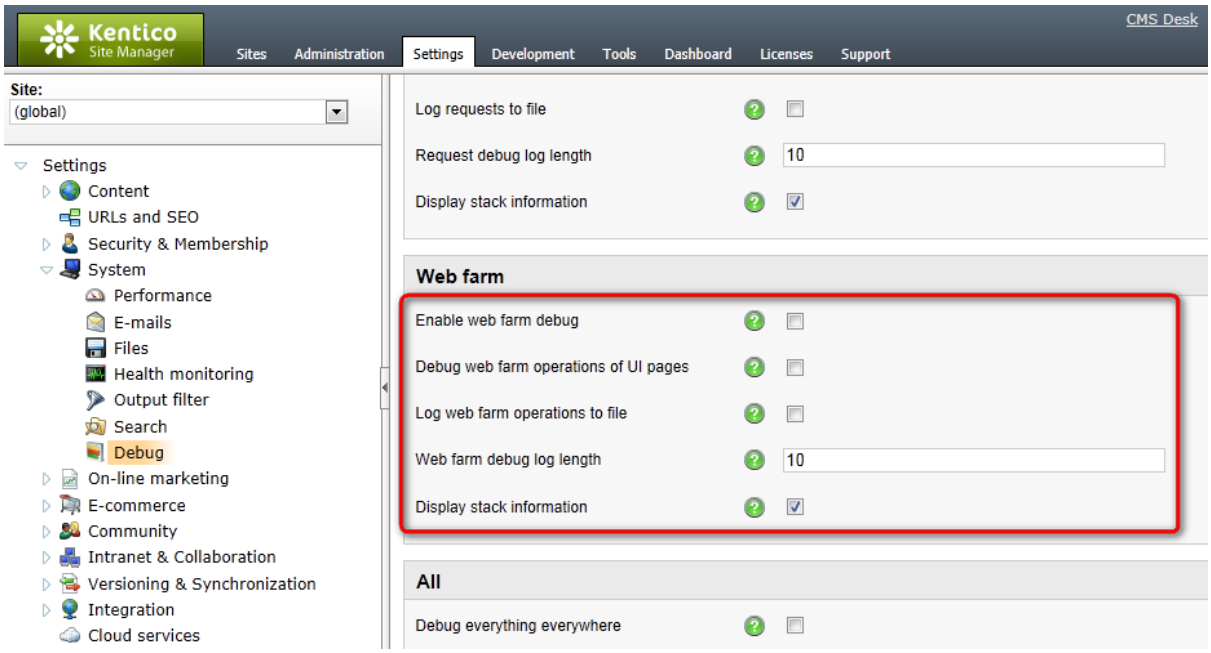


7.6.3.13 Web farm

Web farm debugging can be turned on and configured either by adjusting certain settings in **Site Manager -> Settings -> System -> Debug**, or by adding certain keys into the *AppSettings* section of your *web.config* file. The following table lists and explains these settings and keys:

Setting	Web.config key	Description
Enable web farm debug	CMSDebugWebFarm	Enables request debugging and the **Web farm** tab in **Site Manager -> Administration -> System -> Debug**.
Debug web farm operations of UI pages	CMSDebugAllWebFarm	If enabled, operations performed via the administration interface (CMS Desk and Site Manager) will also be included in the web farm debug. This option requires web farm debugging to

		be enabled.
Log web farm operations to file	CMSLogWebFarm	If enabled, web farm debug log is saved into the *logwebfarm.log* file in the ~\App_Data folder. This option does not require web farm debugging to be enabled.
Web farm debug log length	CMSDebugWebFarm LogLength	Sets the maximum length of the web farm debug log on the **Debug -> Requests** tab, i.e. the number of requests for which debug information is preserved and displayed on the tab. If empty, value of the **Default log length** setting (or the **CMSDebugEverythingLogLength** key) is used.
Display stack information	CMSDebugWebFarm Stack	If enabled, stack is tracked when debugging web farm operations and is displayed in the **Context** column. This information is only available in the debugging UI and on the live site, not in the debug log written into the *logwebfarm.log* file.



Please note that web farm debugging is only functional if web farms are enabled in the *web.config* and at least one server is defined (as described in Modules -> Web farm synchronization -> Defining web farm server).

It may happen that you specify different configuration in the settings and in the *web.config* file. In such cases, boolean settings (true/false) need to be enabled at least in one place (in *web.config* or in settings) in order to be enabled, while log lengths specified in **Site Manager -> Settings** have higher priority than log lengths specified in the web.config.

Here is a list of the keys for easy copy&paste into your *web.config*:

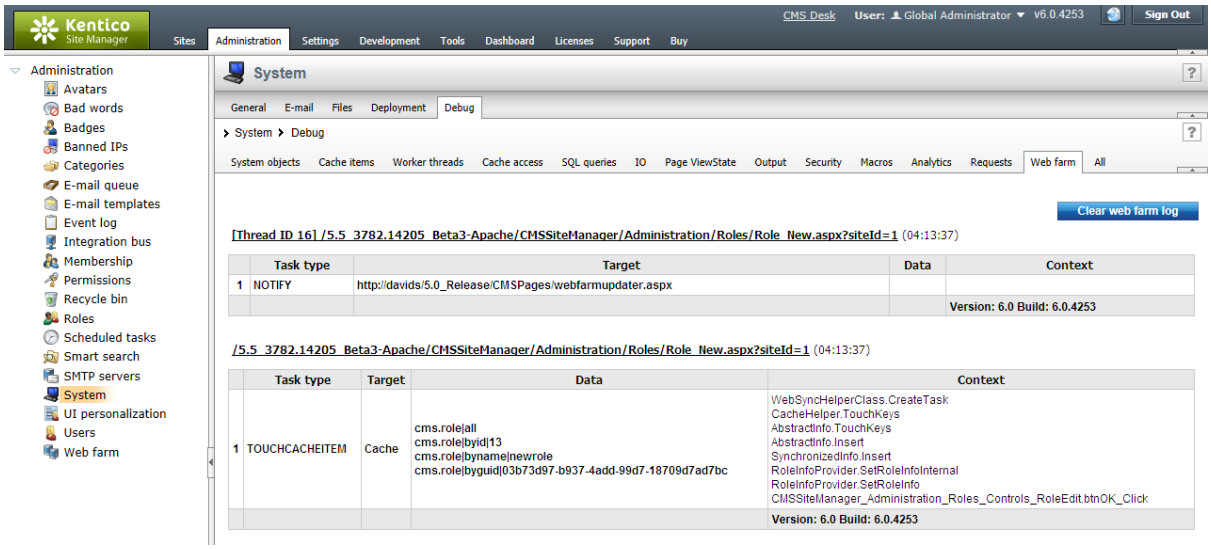
```
<add key="CMSDebugWebFarm" value="true" />
<add key="CMSDebugAllWebFarm" value="true" />
<add key="CMSLogWebFarm" value="true" />
<add key="CMSDebugWebFarmLogLength" value="20" />
<add key="CMSDebugWebFarmStack" value="true" />
```

Web farm debugging can also be enabled using the general settings and keys.

User interface

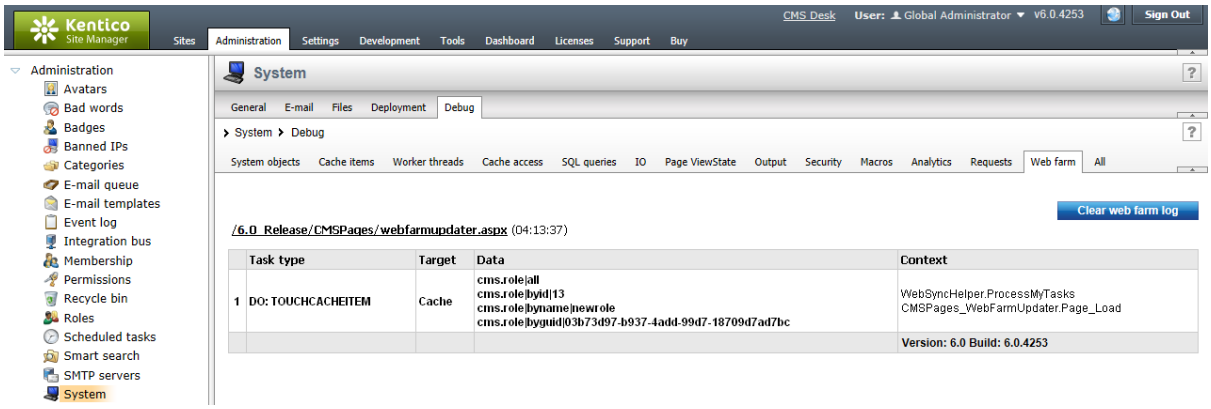
On the **Debug -> Web farm** tab, you can see which web farm synchronization tasks were logged and which servers were notified about the changes. This is particularly useful if you want to find out if the web farm works effectively and synchronizes data correctly, or generally if you want to troubleshoot web-farm-related issues.

The following screenshot shows how the debug looks on the source server (the server where the data was modified). The bottom part displays the logged tasks, while the top part shows asynchronous notifications for other web farm servers.



The second screenshot shows how the debug looks on the target server after it is notified about the modification. You can recognize the task by the *DO:* prefix, which indicates that the synchronization task was (is being) processed.

Enabling the **Show complete context** check-box displays complete context (not only the topmost item) in the **Context** column. The log can be cleared using the **Clear web farm log** button.

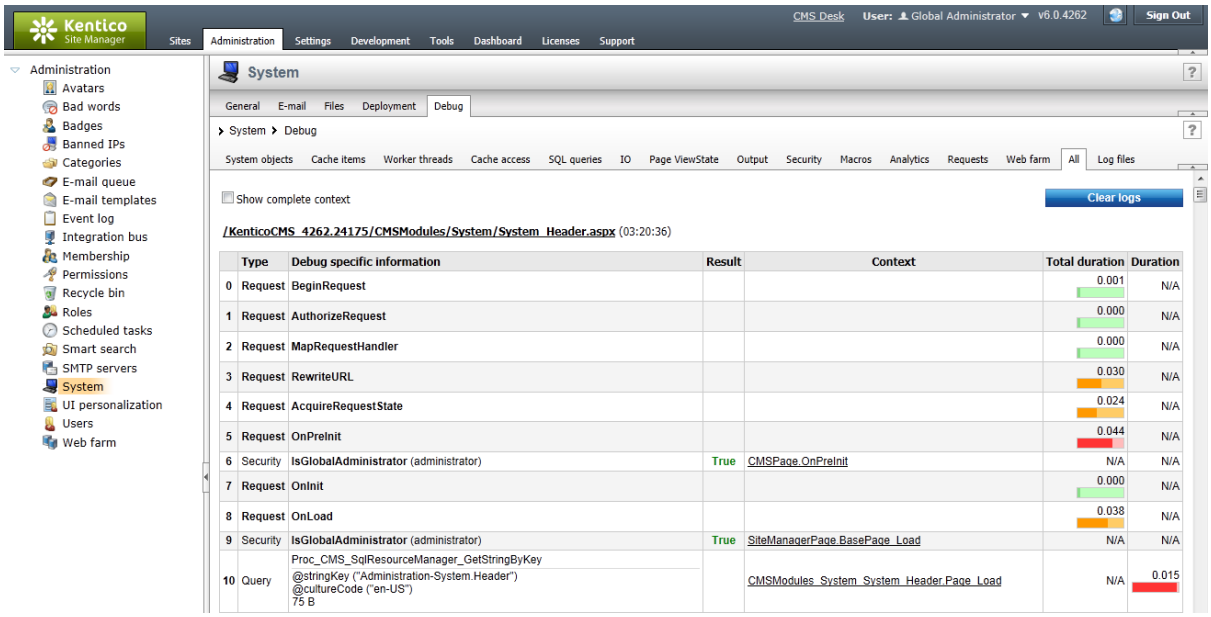


7.6.3.14 All

The **Debug -> All** tab is only displayed when at least two of the debugs that are not enabled by default (i.e. all except **System objects**, **Cache items** and **Worker threads**) are enabled, while the **Request** debug must be one of them.

This tab works as an aggregator of information logged by all enabled types of debugs. For each requests, it displays debug information from all enabled debugs. The information is listed in the order in which the respective action was processed within the request.

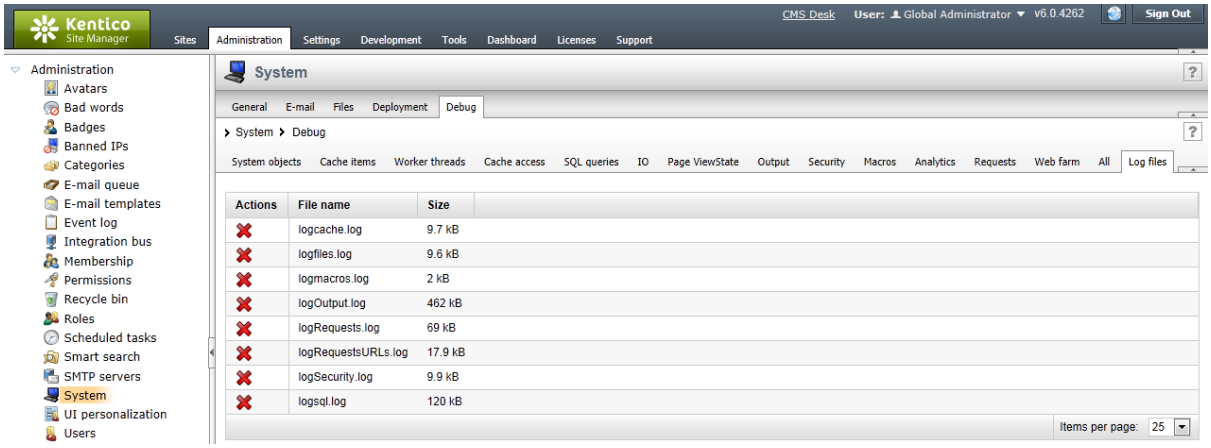
Enabling the **Show complete context** check-box displays complete context (not only the topmost item) in the **Context** column. The logged debug information can be cleared using the **Clear log** button.



7.6.3.15 Log files

The **Debug -> Log files** tab is only displayed when logging of debug information into log files is enabled for at least one type of debug. This can be achieved either by adjusting certain settings or by adding certain keys into the *appSettings* section of your *web.config* file. Please see the topics about particular debugs above or the General settings and keys for all debugs topic below for more details.

This tab provides an overview of all log files stored in the ~*App_Data*\ folder. By clicking the **Delete** ( ) button, you can delete the respective file from the *App_Data* folder.



7.6.3.16 E-mails

E-mail debugging can help you resolve problems with newsletter, notification and other e-mails sent by Kentico CMS. To enable it, add the following keys to the web.config file of your web project:

```
<add key="CMSLogEmails" value="true" />
<add key="CMSDebugEmails" value="true" />
```

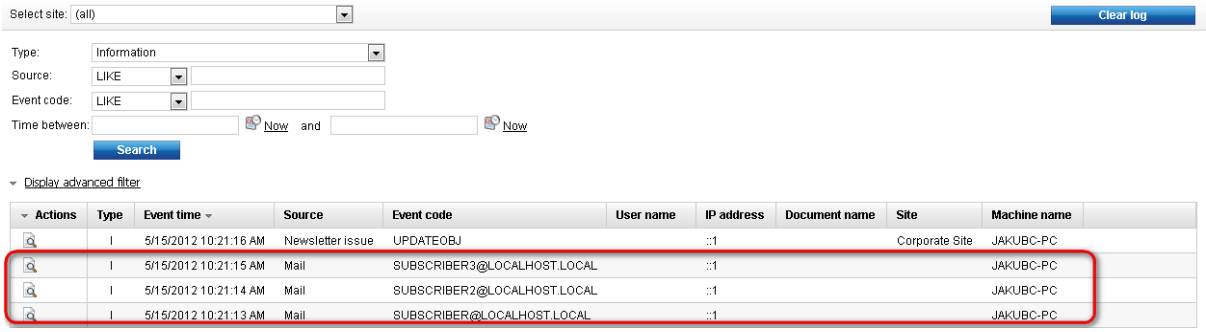
The **first key** enables you to log all the sent e-mails. You can find the logged e-mails in **<web root>/ AppData/logemails.log**. The log contains each e-mail's recipient and subject. The following extract from the log shows 2 newsletters sent to 3 recipients each.

```
Rcpt: subscriber@localhost.local
Subject: Newsletter #1

Rcpt: subscriber2@localhost.local
Subject: Newsletter #1

Rcpt: subscriber3@localhost.local
Subject: Newsletter #1

Rcpt: subscriber@localhost.local
Subject: Newsletter #2

Rcpt: subscriber2@localhost.local
Subject: Newsletter #2

Rcpt: subscriber3@localhost.local
Subject: Newsletter #2
```

The **second key** enables you to log all the sent e-mails into event log without actually sending them to the recipients. This is helpful when you need to test the functionality but do not want the testing e-mails

to actually reach the recipient.

You can view the event log in **Site Manager -> Administration -> Event log**. The e-mails are logged as an **Information** type of event. The following picture shows a logged e-mail sent to three different recipients. The recipient's address is mentioned in the **Event code** column of the Event log.



Additionally, with the second key enabled, a **'Sending failed for <recipient's e-mail address>'** error is generated at random to simulate an error when sending an e-mail.

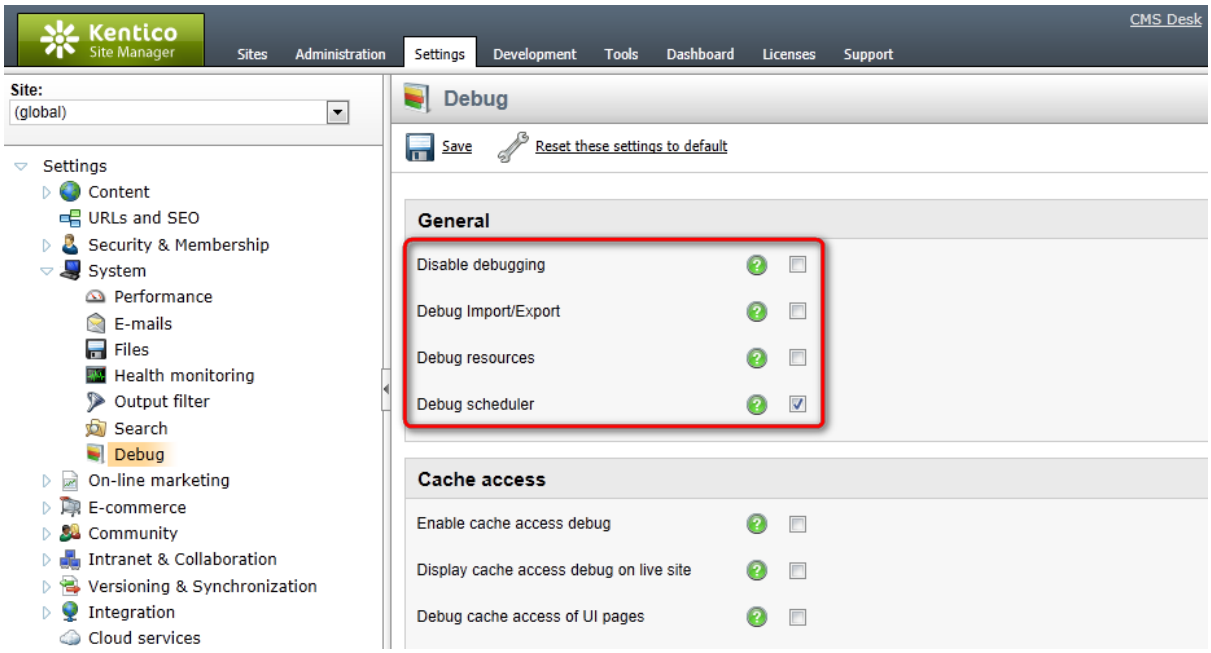
You can find more information on viewing logged events in Developer's guide -> Modules -> Event log.

7.6.4 General settings and keys for all debugs

Apart from the settings and *web.config* keys for individual debugs listed in the topics above, you can use the following general settings and keys to configure debugging on a global level, i.e. to affect all the individual debugs. In **Site Manager -> Settings -> System -> Debug**, the global settings are located in two setting groups.

In the **General** group, you can find the following settings:

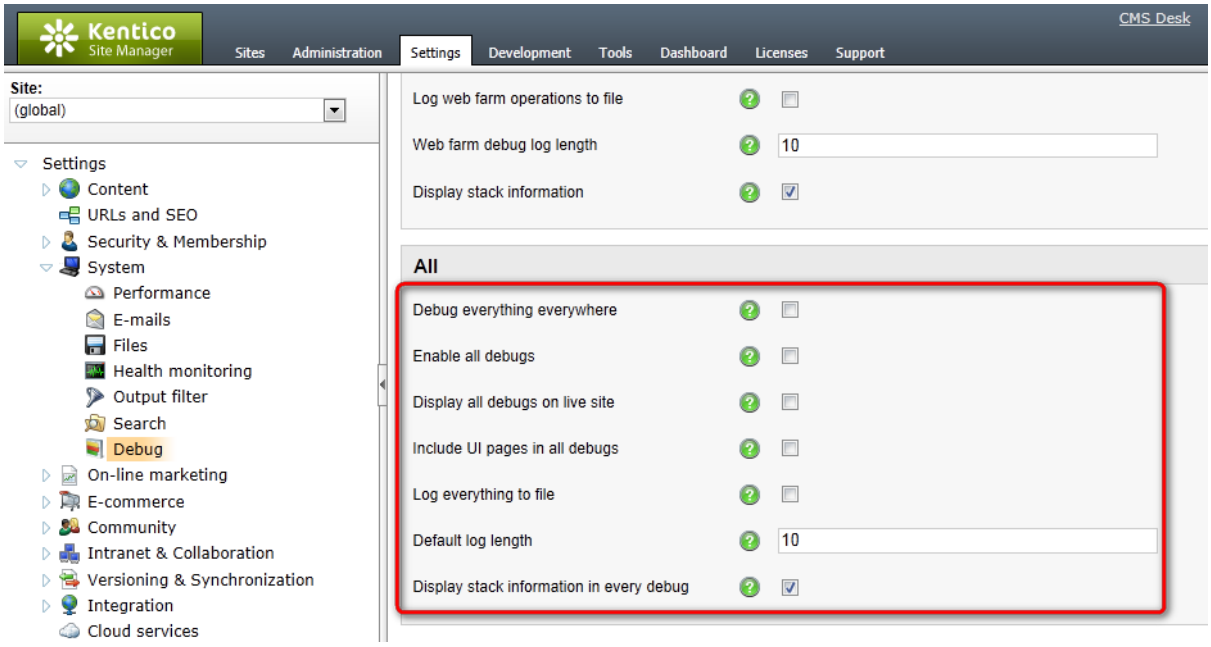
Setting	Web.config key	Description
Disable debugging	CMSDisableDebug	Globally disables all debugs, regardless of individual debug settings.
Debug Import/Export	CMSDebugImportExport	If disabled, debug information is not logged for the Import/Export user interface. To optimize system performance, it is recommended to have this option disabled unless you really need to debug the Import/Export process.
Debug resources	CMSDebugResources	If false, all resource requests (*GetResource* and *GetCSS*) are ignored by all debugs.
Debug scheduler	CMSDebugScheduler	If false, all scheduler operations are excluded from all debugs.



In the **All** group, the following settings can be found:

Setting	Web.config key	Description
Debug everything everywhere	CMSDebugEverythingEverywhere	Enables all debugs, includes UI pages in all debugs and ensures that the debugs are displayed both in the **Site Manager -> Administration -> System -> Debug** interface and on the live site.
Enable all debugs	CMSDebugEverything	Enables all debugs and ensures that the corresponding tabs are displayed in **Site Manager -> Administration -> System -> Debug**.
Display all debugs on live site	CMSDebugEverythingLive	If enabled, debug information of all debug types is displayed at the bottom of each live site page. This only applies to debugs that are already enabled.
Include UI pages in all debugs	CMSDebugAllForEverything	If enabled, actions performed on UI pages are included in all debugs. This only applies to debugs that are already enabled.
Log everything to file	CMSLogEverythingToFile	Enables logging of all possible operations (including the Event log and E-mail sending log) into *.log* files stored in the *~/App_Data/* folder.
Default log length	CMSDebugEverythingLogLength	Sets the default maximum length of all debugs on the respective tabs in **Site Manager -> Administration ->System -> Debug**. This value is used if no log length is configured for the respective type of debug.
Display stack information in every	CMSDebugStackForEverything	If enabled, stack is tracked by all debugs and is displayed on the respective tabs in **Site Manager ->**

debug		**Administration -> System -> Debug**.



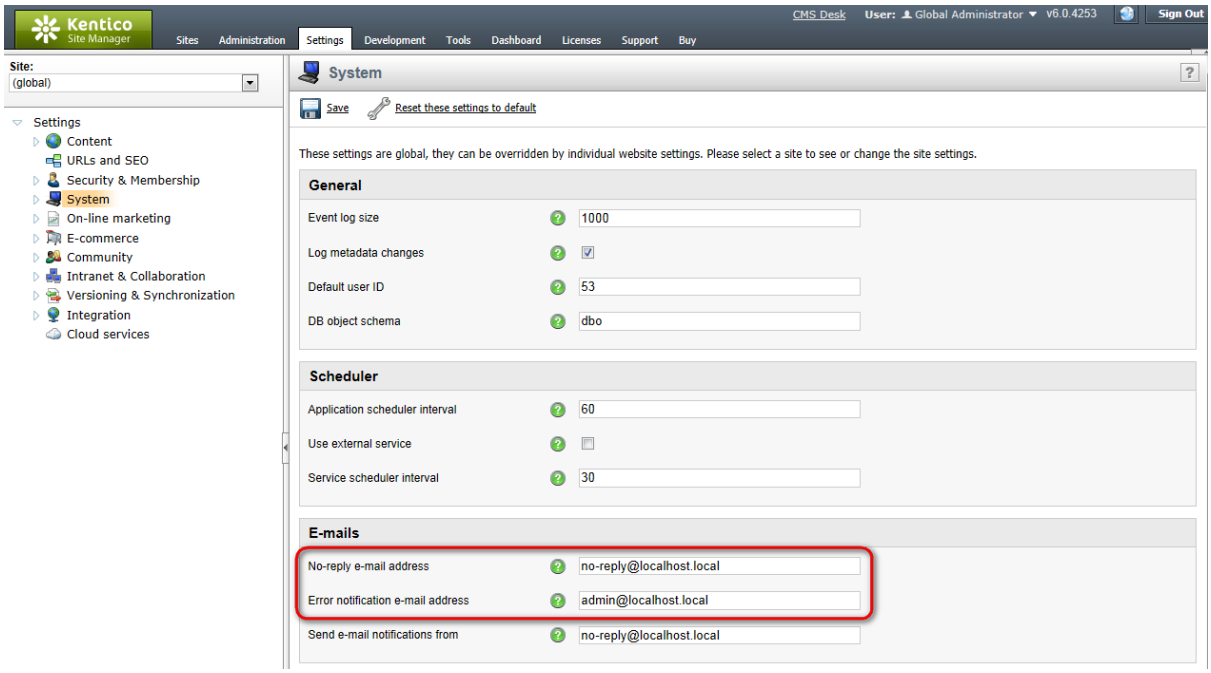
Here is a list of the keys for easy copy&paste into your *web.config*:

```
<add key="CMSDisableDebug" value="true"/>
<add key="CMSDebugStackForEverything" value="true"/>
<add key="CMSDebugImportExport" value="true"/>
<add key="CMSDebugResources" value="true"/>

<add key="CMSDebugEverythingEverywhere" value="true"/>
<add key="CMSDebugEverything" value="true"/>
<add key="CMSDebugEverythingLive" value="true"/>
<add key="CMSDebugAllForEverything" value="true"/>
<add key="CMSLogEverythingToFile" value="true"/>
<add key="CMSDebugEverythingLogLength" value="10"/>
```

7.6.5 System error notifications

If you go to **Site Manager -> Settings -> System** and fill in an e-mail address into the **Error notification e-mail address**, notifications about internal errors in Kentico CMS system will be sent to this address whenever such an error occurs.

The e-mail address specified in the **No-reply e-mail address** field will be used as the sender ('From') e-mail address.



7.7 Document types and transformations

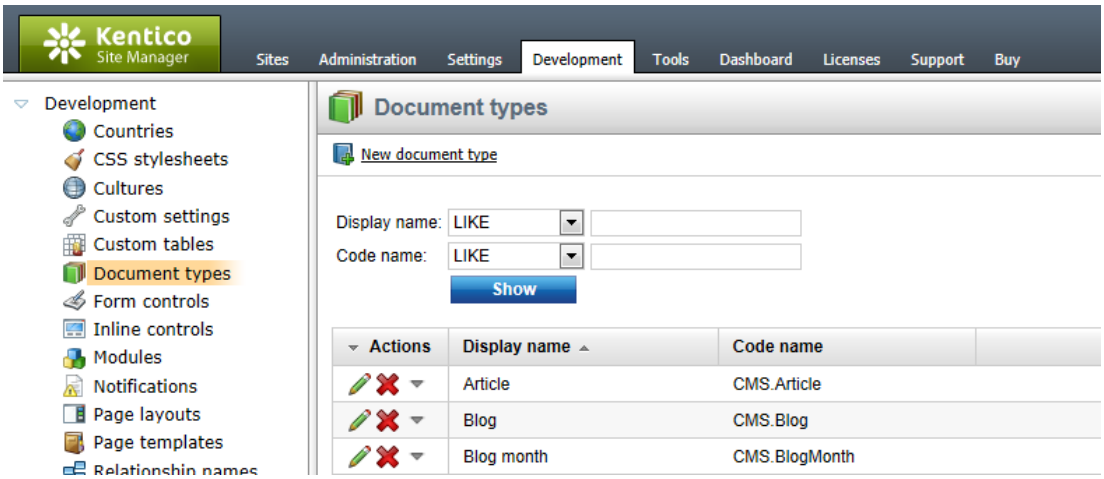
7.7.1 Overview

Each document in Kentico CMS is of some type. Each document type has its own:

- fields (data structure)
- editing form layout
- transformations (design)
- queries

and other settings.

Document types are fully customizable - you can add, modify and delete custom fields. The advantage of using custom document types is that you can define custom structure of documents and store content (data) separated from design. This can be done in **Site Manager -> Development -> Document types**.



- To learn how to define a new document type, please refer to the Defining a new document type topic.
- To learn about document type properties, please refer to the Document type properties topic.
- To learn how to upload your own icons and associate them with document types, please refer to the Uploading document type icons topic.
- Transformations as used in Kentico CMS are described in the Transformations subchapter.

To see a practical example of creating a new document, please refer to the Creating a new structured document topic in the Content management -> Editing content section of this guide.

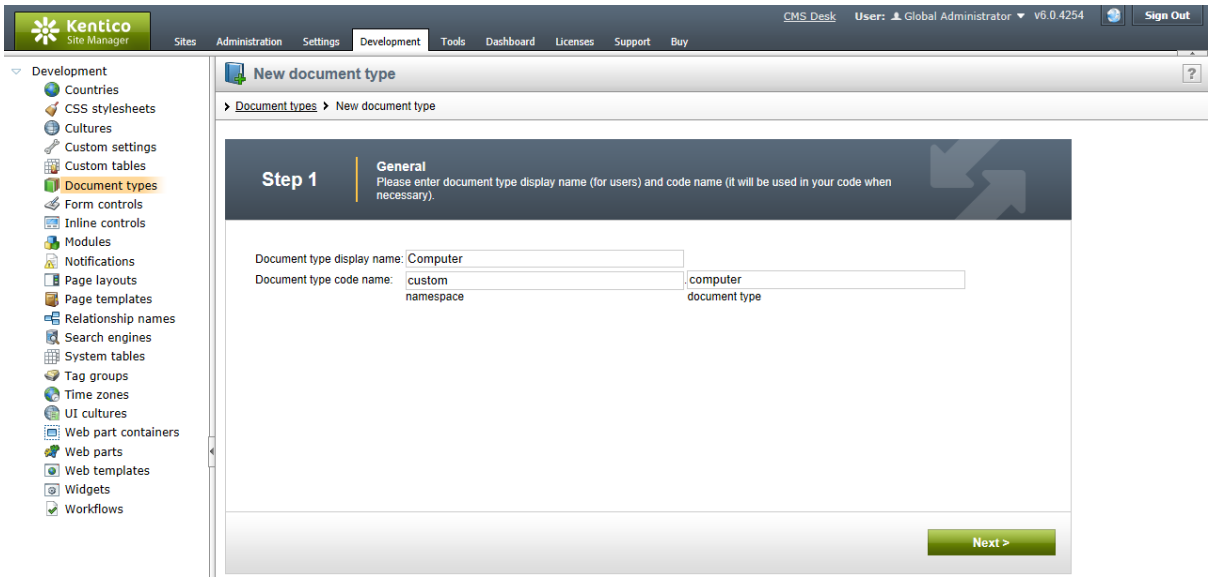
7.7.2 Defining a new document type

In this example, you will learn how to define a new custom document type.

1. Go to **Site Manager -> Development -> Document types** and click **New document type**. You will be redirected to the New document type wizard. In the first step, enter the following values:

- **Document type display name:** Computer (this name will be displayed to users in the administration interface)
- **Document type code name:** custom.computer (*custom* is your namespace to distinguish your own document types from system types that use the *cms* namespace, *computer* is the document type). You will use this value in web part properties later.

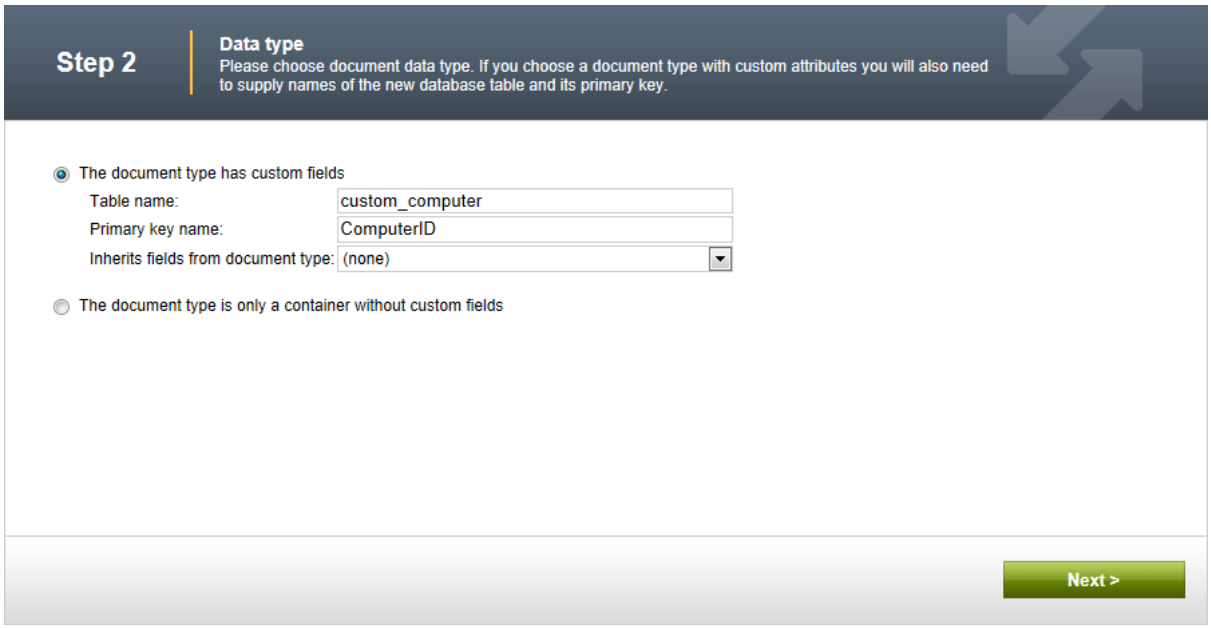
Click **Next**.



2. In Step 2, you need to choose the name of the database table that will be used for storing computer details. You also need to enter the name of the primary key in this table. Enter the following values:

- **Table name:** CUSTOM_Computer
- **Primary key name:** ComputerID
- **Inherits fields from document type**: (none)

Click **Next**.



3. The wizard has created a new database table for computers. Now you need to define the fields of the document type (columns of the table). Use the **New attribute** (➕) button to create the following fields. For each field, enter the values, click 💾 **Save** and repeat the procedure until you have all the listed fields defined.

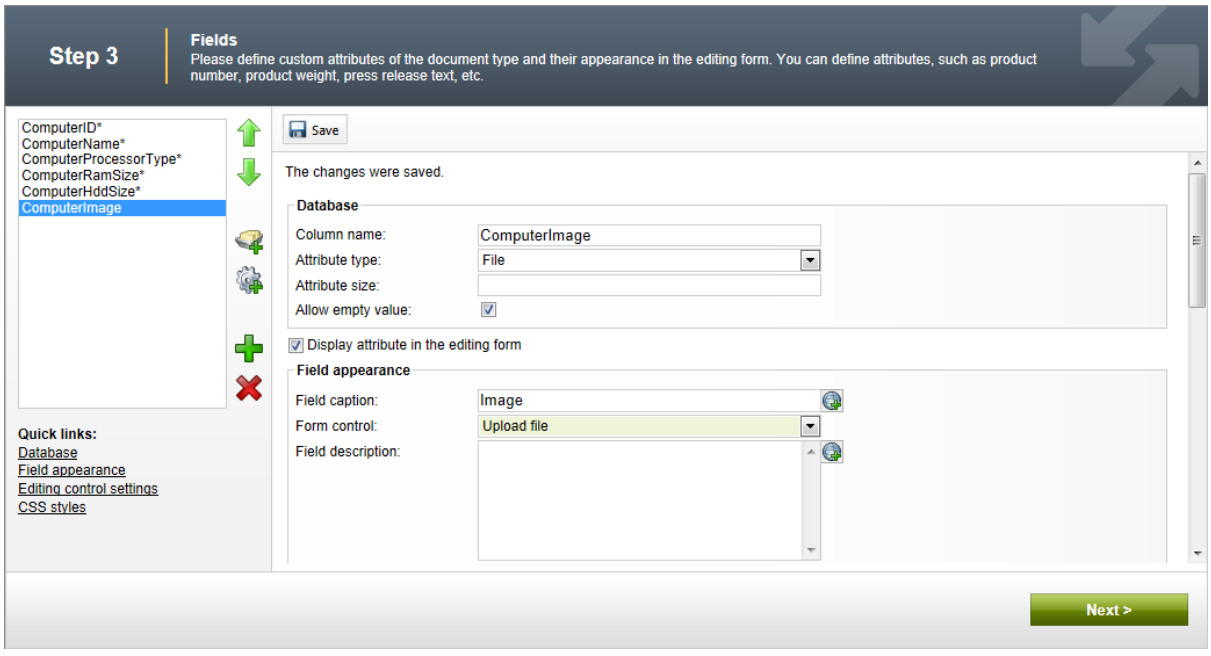
- **Column name:** ComputerName
- **Attribute type:** Text
- **Attribute size:** 200
- **Field caption:** Computer name
- **Form control:** Text box

- **Column name:** ComputerProcessorType
- **Attribute type:** Text
- **Attribute size:** 200
- **Field caption:** Processor type
- **Form control:** Drop-down list
- **Editing control settings -> Data source:** select *Options* and enter the following items into the text area, one per line:
  Athlon;Athlon
  Pentium XEON;Pentium XEON
  Pentium Core 2 Duo;Pentium Core 2 Duo

- **Column name:** ComputerRamSize
- **Attribute type:** Integer number
- **Field caption:** RAM (MB)
- **Form control:** Text box

- **Column name:** ComputerHddSize
- **Attribute type:** Integer number
- **Field caption:** HDD (GB)
- **Form control:** Text box

- **Column name:** ComputerImage
- **Attribute type:** File
- **Allow empty value:** Enabled (check the box)
- **Field caption:** Image
- **Form control:** Upload file

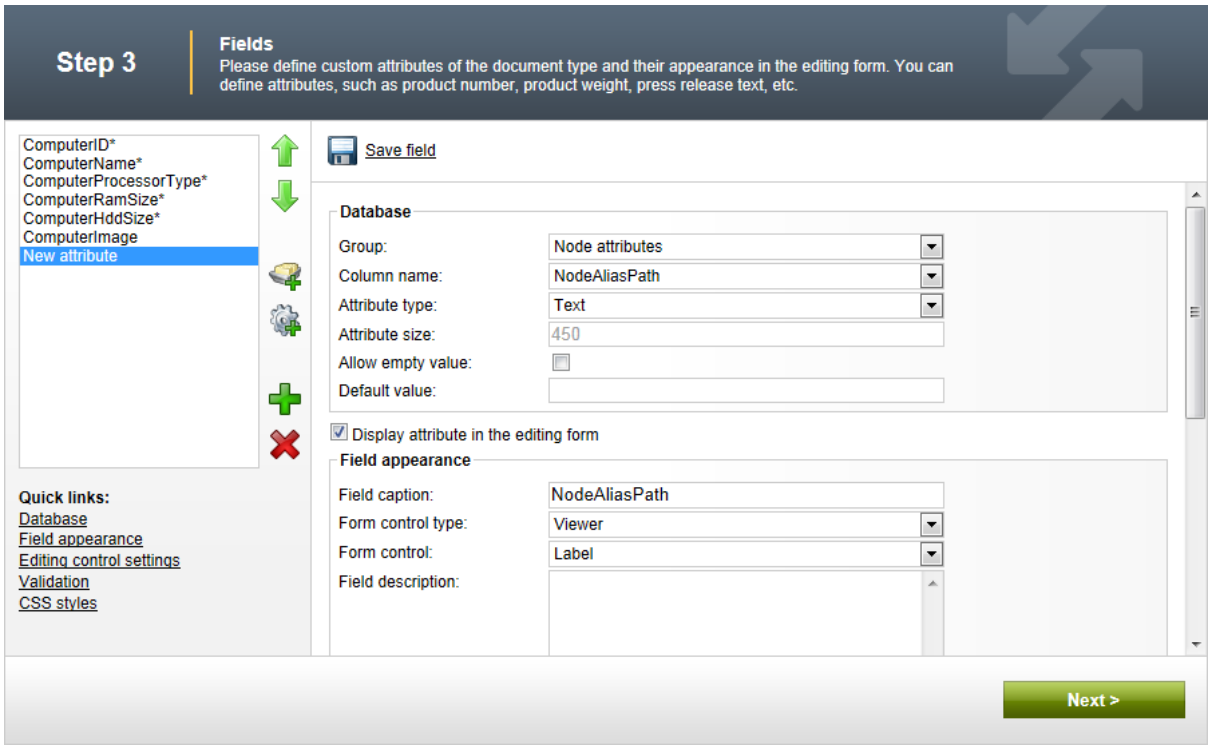


Please note that you can also define system fields that will be displayed when editing documents of this type on the **Form** tab. This can be done by clicking the **New system attribute** ( ) button. Using the **Group** drop-down list, you can then choose from the following two groups of system fields:

- **Document attributes** - offers the system fields of documents.
- **Node attributes** - offers the system fields of content tree nodes.

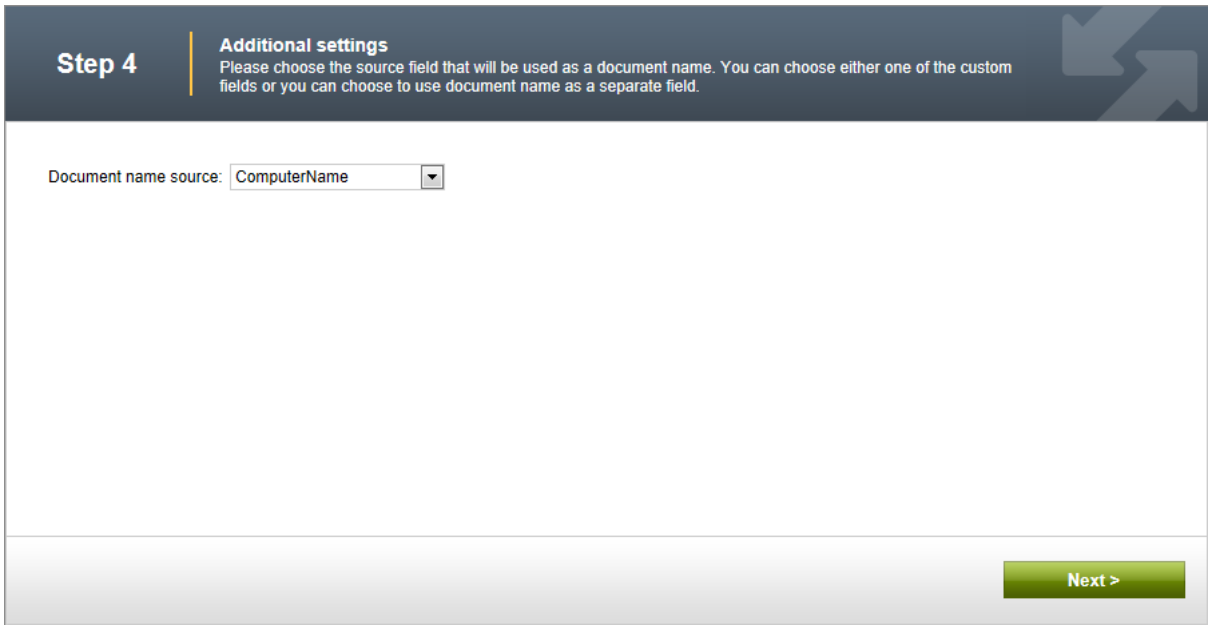
Document or node system fields will then be offered in the **Attribute name** drop-down list. If you leave the **Display attribute in the editing form** check-box checked, the field will be visible on the document **Form** tab.

Click **Next**.



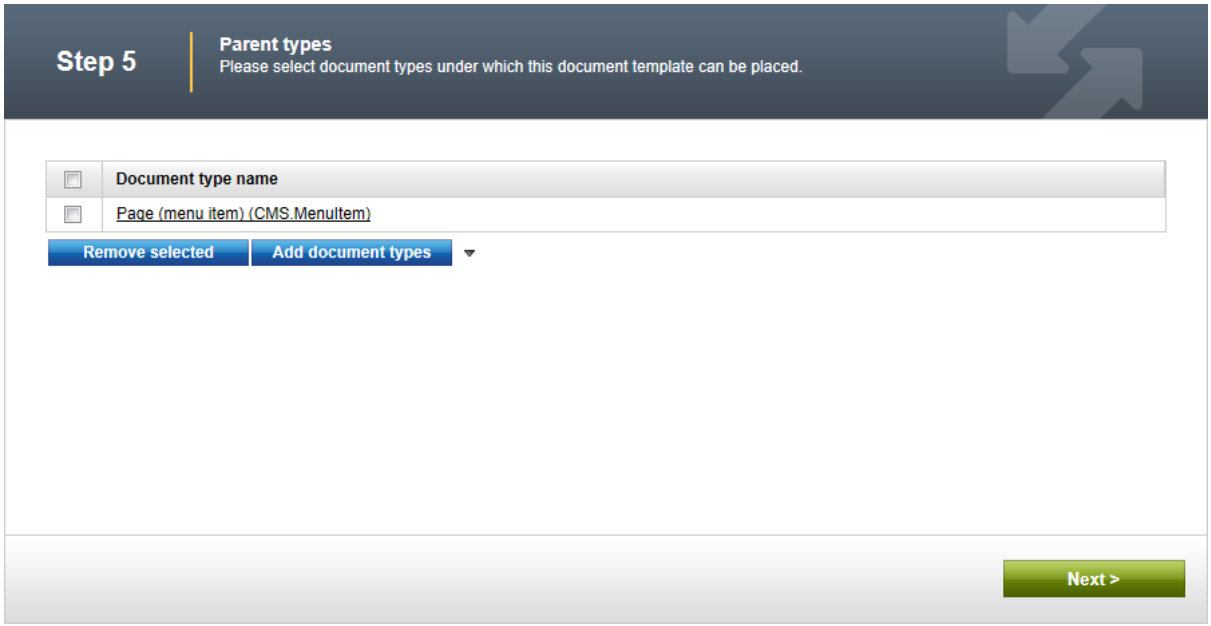
4. Now you need to choose the field that will be used as the name for document's of this type. Choose the **ComputerName** option from the drop-down list. This ensures that computer documents will use the value of the **ComputerName** field as the name displayed in site navigation and in the CMS Desk content tree.

Click **Next**.

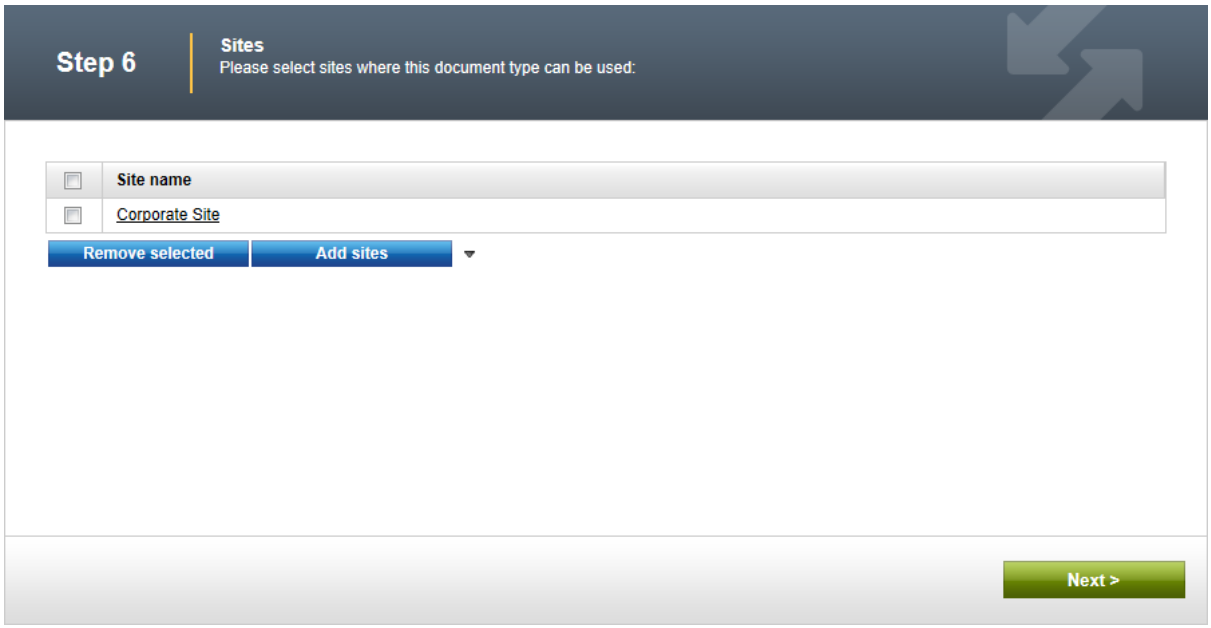


5. In Step 5, you need to select the document types under which computer documents will be displayed. Check only the **Page (menu item)** value, which means that editors will be able to create computer documents only under some page, not under an article or news document in the content tree.

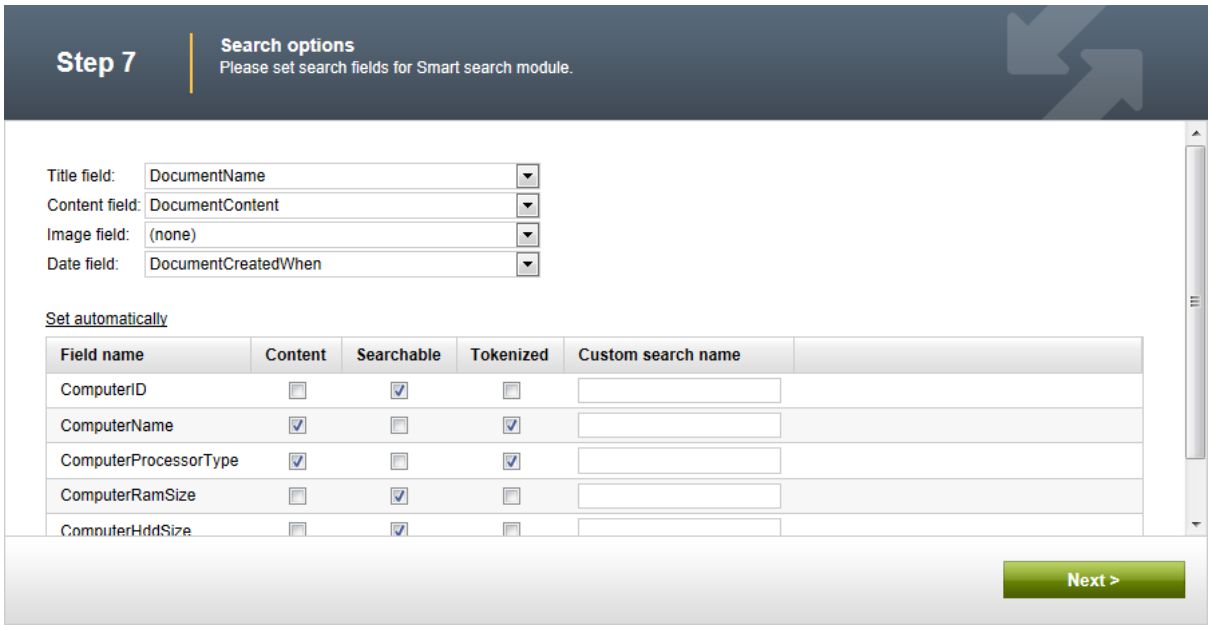
Click **Next**.



6. In Step 6, you need to choose which websites will use this document type. Check the appropriate website and click **Next**.

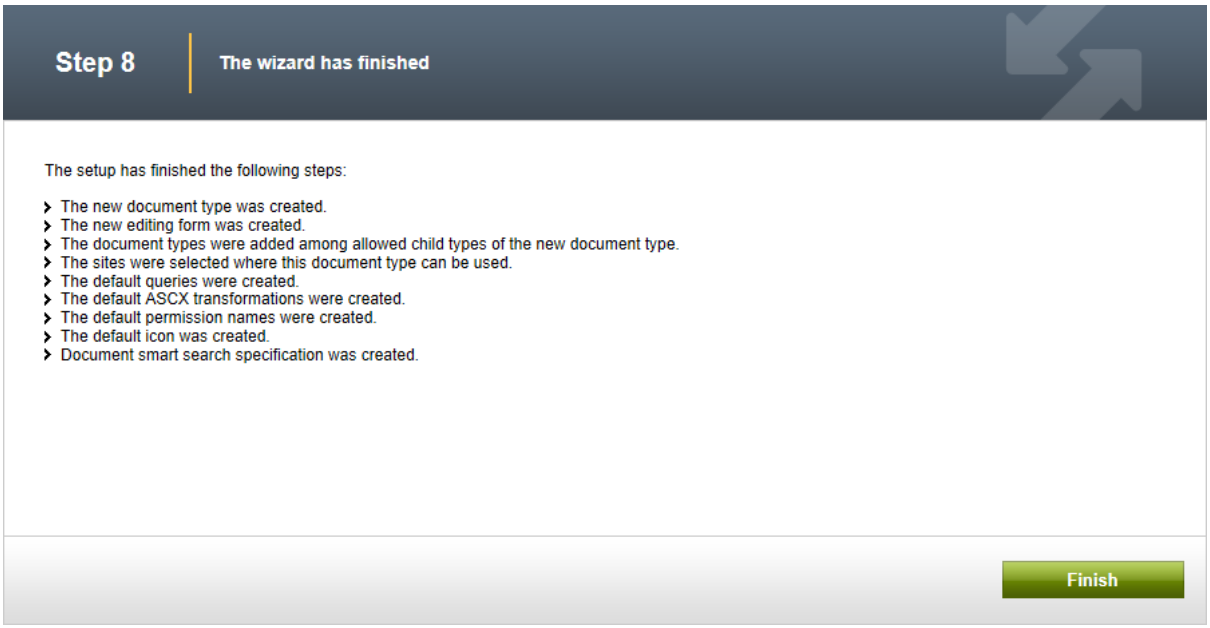


7. In Step 7, you are asked to specify how documents of this type will be indexed and displayed in the search results. For more information on these settings, please refer to this topic. Make your choice and click **Next**.



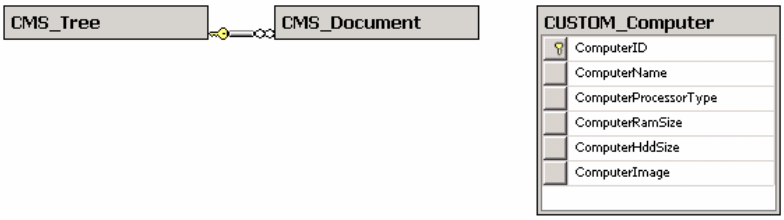
8. The wizard has finished the configuration of the new document type. It has automatically created not only the database table, but also the SQL queries for SELECT, INSERT, UPDATE and DELETE operations and default transformations.

Click **Finish**. Congratulations, you have learned how to define a new document type.



How the content is stored

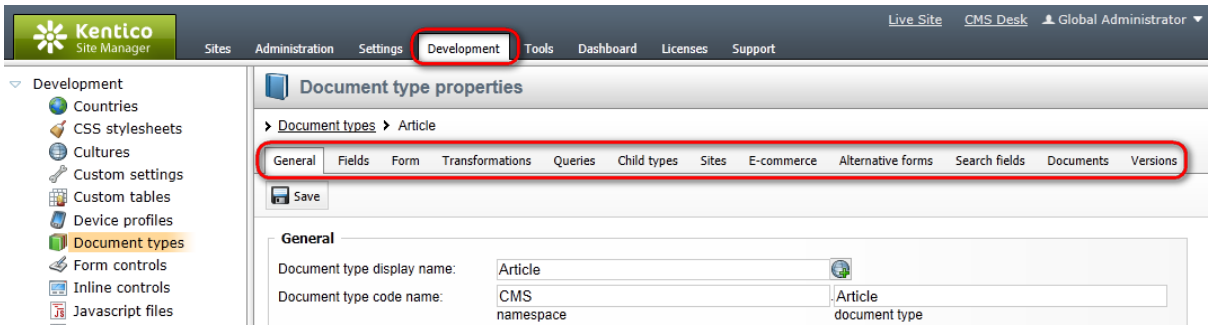
As you already know, the new document type *Computer* has its own database table for its specific fields. Each document is stored in three tables: *CMS_TREE* (tree structure), *CMS_Document* (document properties, metadata and content defined on the Page tab) and the custom table - in this case *CUSTOM_Computer*:



The system automatically ensures all operations are performed correctly on these tables. The advantage of this storage is that it is very fast and you can easily write standard SQL SELECT queries to retrieve data from the repository (i.e. from the Microsoft SQL Server database).

7.7.3 Document type properties

Document type properties can be accessed in **Site Manager -> Development -> Document types -> Edit ( ) Document type**.



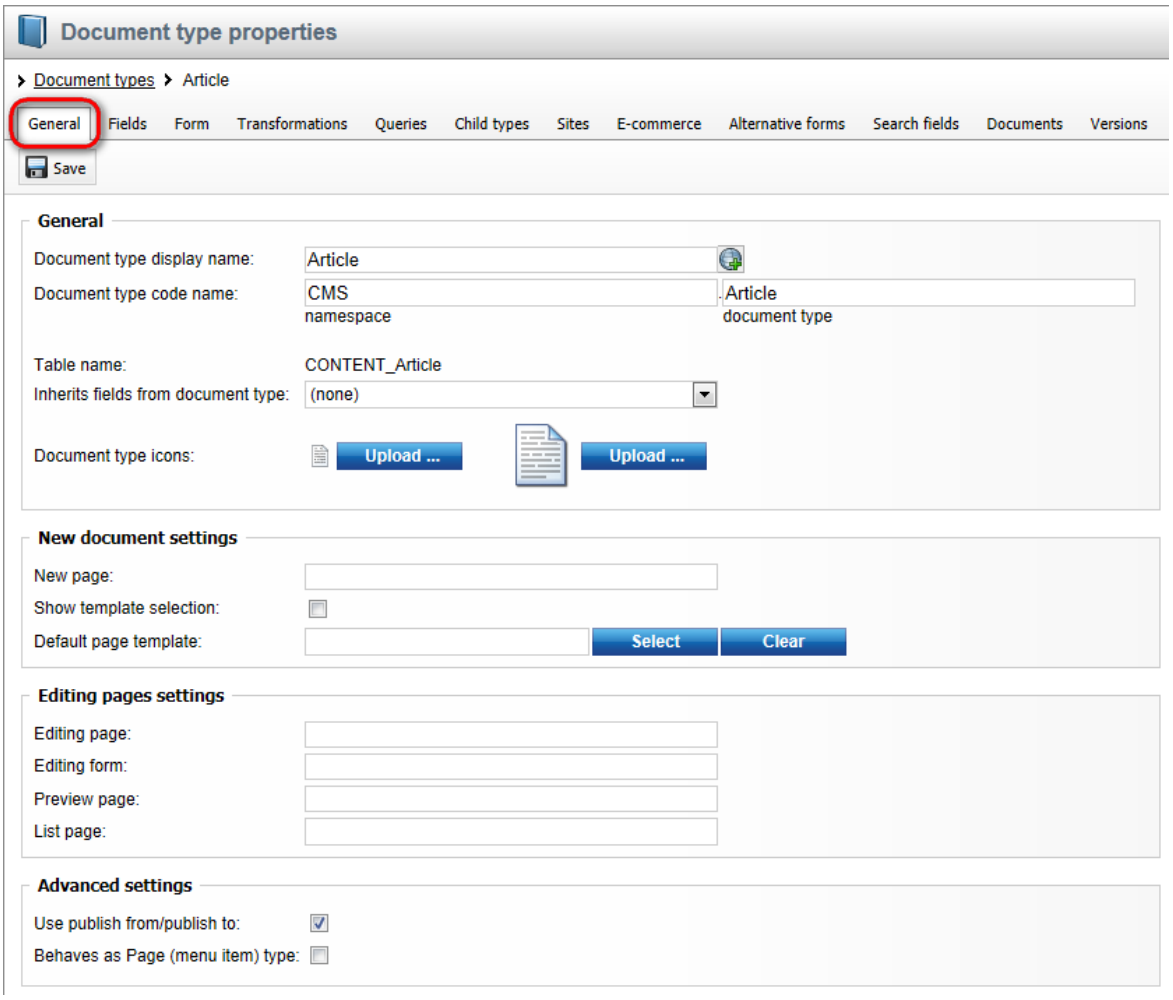
The Document types properties user interface consists of the following tabs: General, Fields, Form, Transformations, Queries, Child types, Sites, E-commerce, Alternative forms, Search fields, Documents and Versions.

General tab

The general properties of a document type define its basic characteristics as shown below:

General	
Document type display name	The name of the document type displayed in the administration and content editing interface.
Document type code name	Sets a namespace prefix and unique name that serves as an identifier for the document type, for example in selectors or the API.
Table name	Displays the name of the database table used to store coupled data for documents of this type, i.e. the content of its specific fields.
Inherit fields from document type	If a document type is selected and saved, the definitions of its fields will be loaded and added to the current document type. These fields can be viewed and modified on the **Fields** tab.
Document type icons	Allows you to upload custom icons that will be used to represent the document type in various parts of the system.
New document settings	
New page	URL of the page that will be used when creating new documents of this type.
Show template selection	Indicates if users will be required to select a page template when creating a new document of this type.
Root page template category	May be used to limit the page template options available for documents of this type to a specific sub-section of the template catalog. When creating new documents of the given type, users can only choose from the templates located under the specified category or in its child categories.
Default page template	Sets the page template used by default when the document is created. If no page template is specified, new documents will inherit the template of their parent by default.
Editing pages settings	

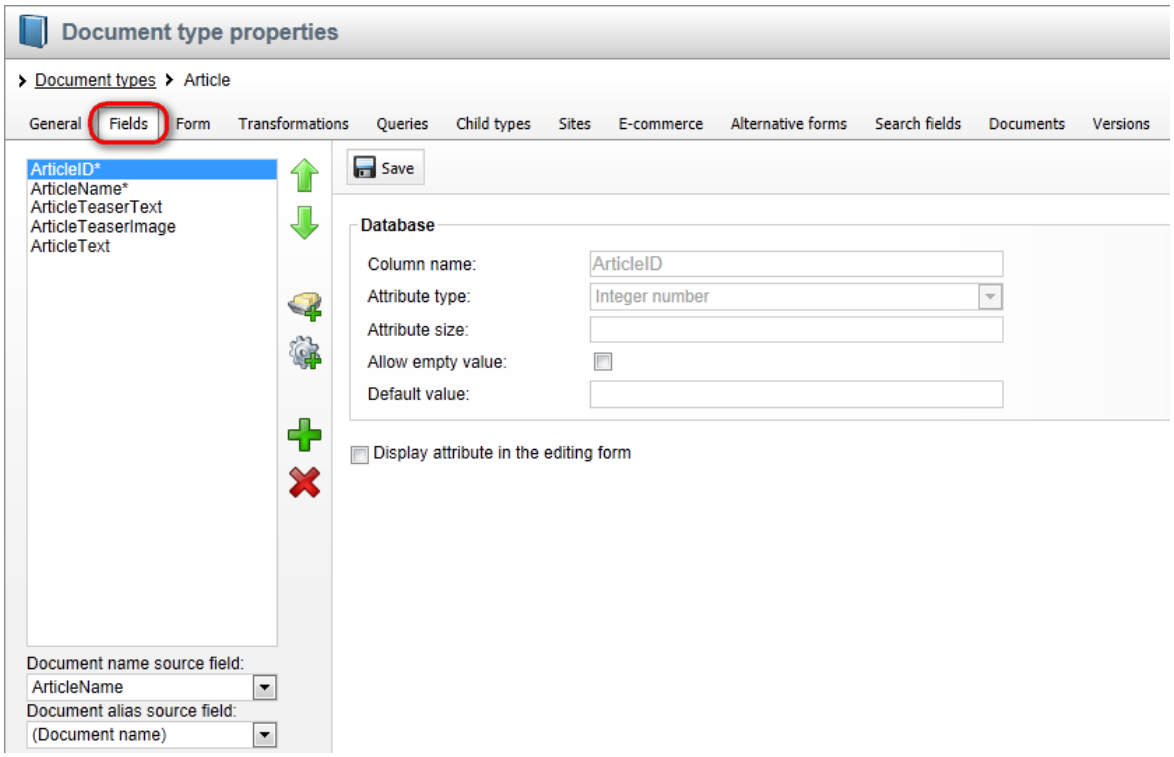
(these should only be specified if you do not wish to use the system's default editing pages)	
Editing page	URL of the editing page that will be used when the document is displayed in editing mode via the **Page** tab.
Editing form	URL of the editing page that will be used when the document is displayed in editing mode on the **Form** tab.
Preview page	URL of the page that will be used when the document is displayed in preview mode.
List page	URL of the editing page that will be used when the document is displayed in list mode.
Advanced settings	
Use publish from/publish to	Indicates if the **Publish from/to** fields should be offered on the **Form** tab for documents of this type. These may be used to schedule the page represented by the document to be published on the live site on a specific date and time, or shown only during a limited time interval.
Behaves as Page (menu item) type	Indicates if the document type should behave similar to the default *Page (menu item)* document type. The default view mode for such a document type is the Page tab. Viewer web parts are automatically set to display child documents if the path is not configured and the document does not inherit its parent template by default.



Fields tab

The **Fields** tab contains a field editor that allows you to manage the data fields (columns) of the document type. Using the tab, you can add a **New attribute** (), **New system attribute** () or **New category** () and you can **Save** (), **Delete** (), **Move Up** () and **Move Down** () the fields. You can also define which fields should be used as the source for the default name and alias of new documents of this type. This can be done via the drop-down lists below the field listbox.

For a practical example of the use of the **Fields** tab, please refer to the File Management -> Document attachments -> Example: Grouped attachment topic in the **Content management** section of this guide.



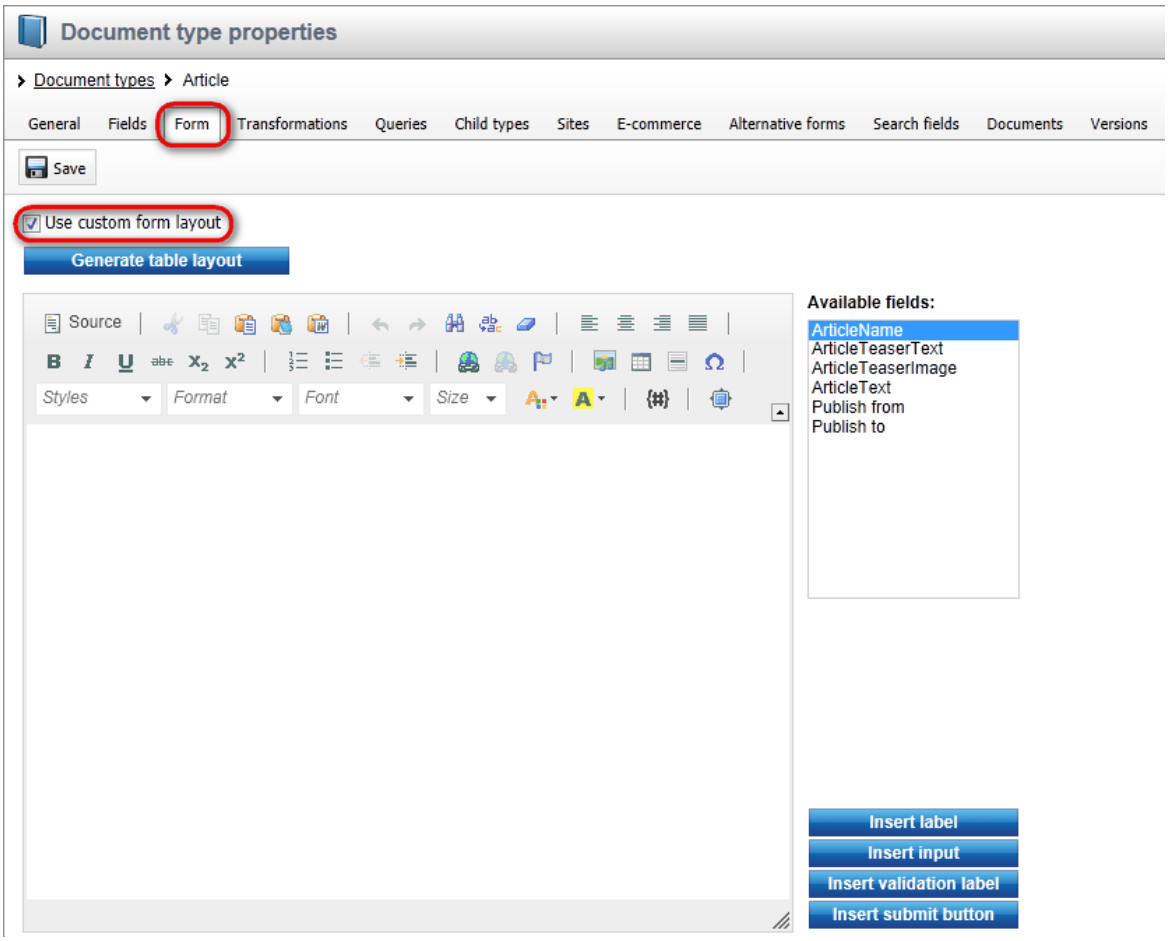
Please note

If a document type is inherited from another document type, the inherited fields are grayed out and must be edited in the parent type. Any changes made to the fields in the parent type are automatically reflected in the child type.

Form tab

The **Form** tab allows you to create a custom form layout that will be used for adding and editing data items. You can use the WYSIWYG editor and you can also insert a field label, input, validation label and submit button. This functionality can be enabled by checking the **Use custom form layout** checkbox. Please note that if no custom layout is available, the default layout will be used.

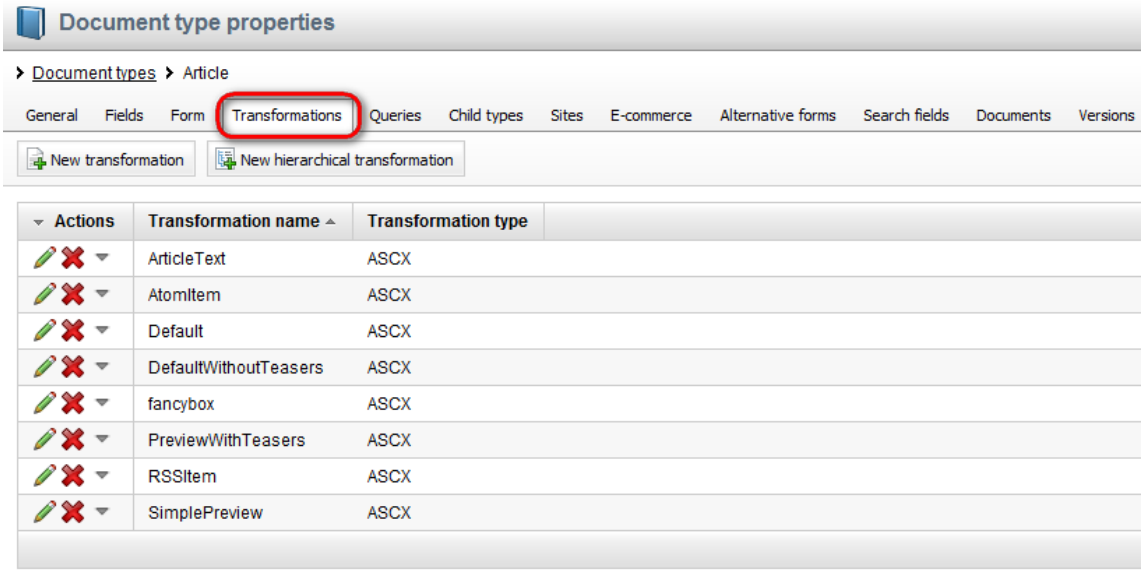
More details about the use of the **Form** tab can be found in Forms -> Defining custom form layout in the **Modules** section or in Custom fields visibility -> Use in custom form layout in the **Development -> Membership, permissions and security** section of this guide.



Transformations tab

On the **Transformations** tab, you can see the list of all available transformations added to the given document type. New transformations can be added using **New transformation** or **New hierarchical transformation**. The transformations can also be **Edited** () or **Deleted** () here.

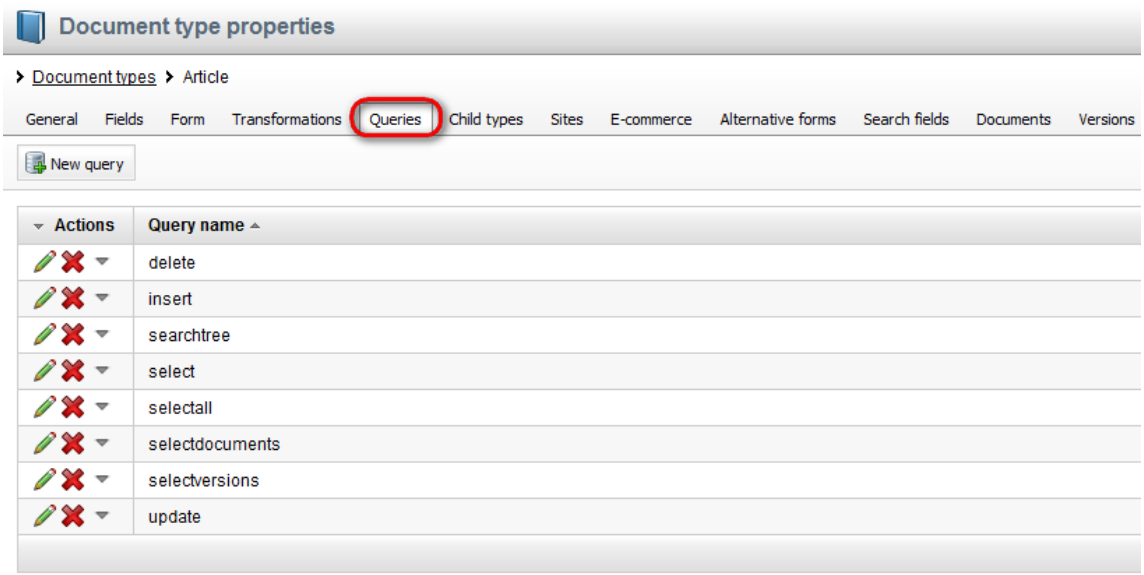
More information about transformations can be found in this chapter in the Transformations, Adding custom functions to transformations, Writing transformations and Hierarchical transformations topics.



Queries tab

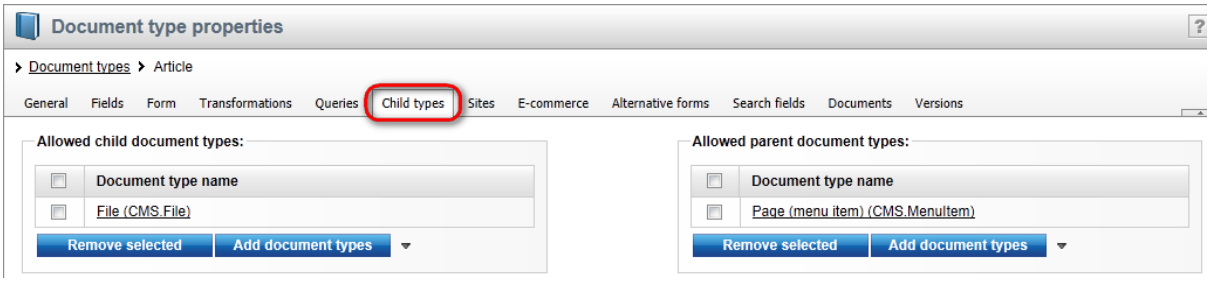
The **Queries** tab displays a list of all available SQL queries added to the particular document type. You can **Edit ( )** or **Delete ( )** items from the list. New queries can be added using **New query**.

More information about the use of SQL queries can be found in the Data layer chapter in the **API programming and Kentico CMS internals** section of this guide.



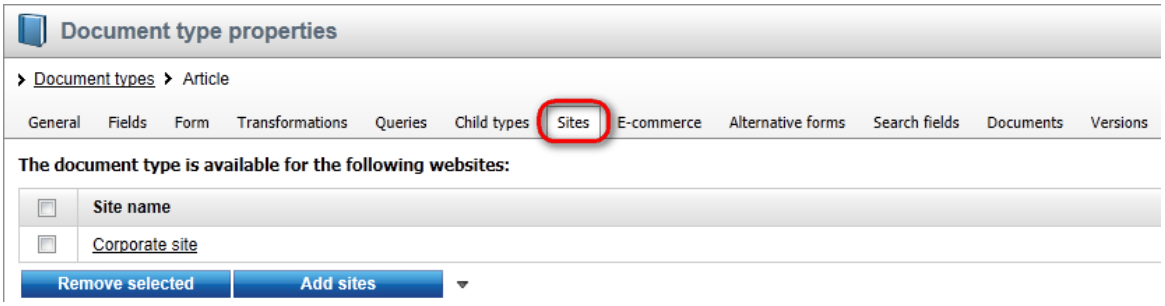
Child types tab

The **Child types** tab allows you to select document types (child document types) that can be placed under documents of the given type. And vice versa, you can define under which document types (parent document types) documents of the given document type can be placed.



Sites tab

The **Sites** tab enables you to select sites where the given document type can be used. If you need to add a site, click the **Add sites** button and select it from the list of available sites. To remove a site, check the checkbox next to the site you would like to remove and click the **Remove selected** button.



E-commerce tab

The **E-commerce** tab allows you to specify whether the current document type should be offered as a product type or section type.

Document's relation to product

- **Document type represents a product type** - indicates if the document type represents a product type.
- **Document type represents a product section** - indicates if the document type represents a product section.

Fields mapping

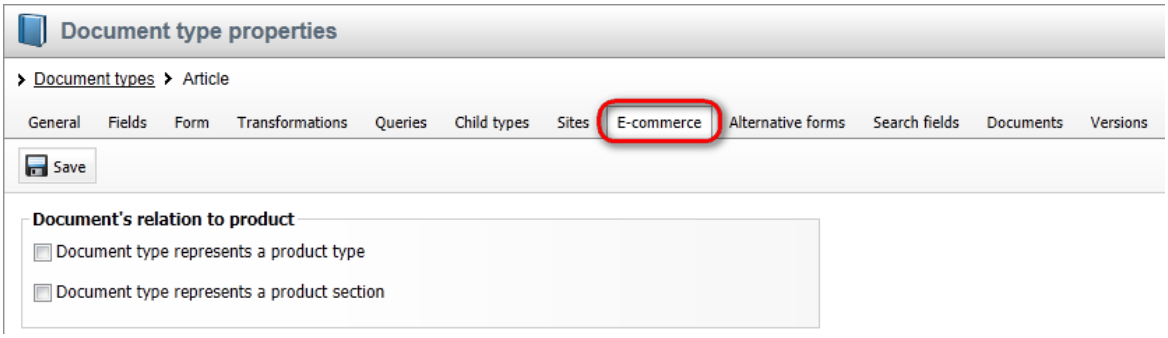
The section is available only if the currently edited document type is set to represent a product type. You can optionally select fields here that will be automatically populated with values from the specified SKU fields.

New product creation

The section is available only if the currently edited document type is set to represent a product type.

- **Default department** - can be used to specify a department where, by default, products created on the basis of the given product type belong.
- **This document type represents** - can be used to choose default representation for the product.
- **Create SKU automatically when a new document of this type is created** - indicates if an SKU

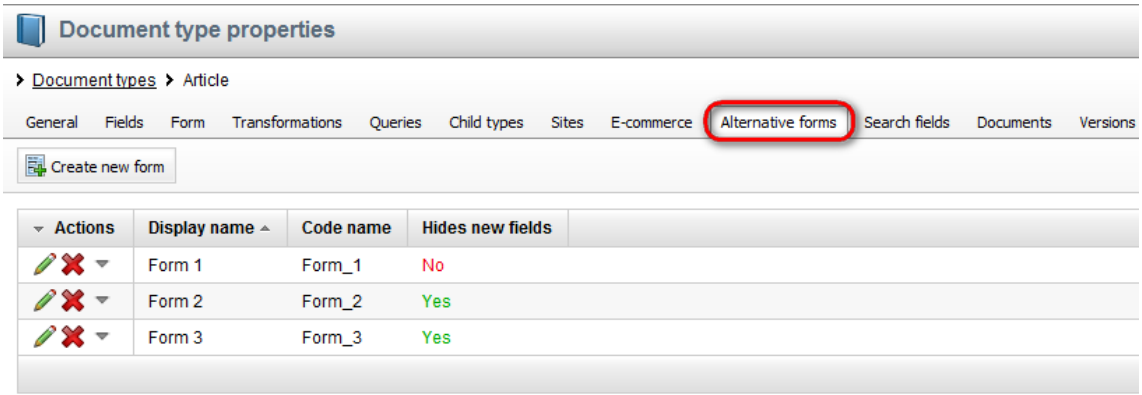
should be created automatically when a new document of this type is created.



Alternative forms

The **Alternative forms** tab allows management of alternative forms for the selected document type. New forms can be created using the **Create new form**. You can also **Edit ()** or **Delete ()** the existing alternative forms listed in the table.

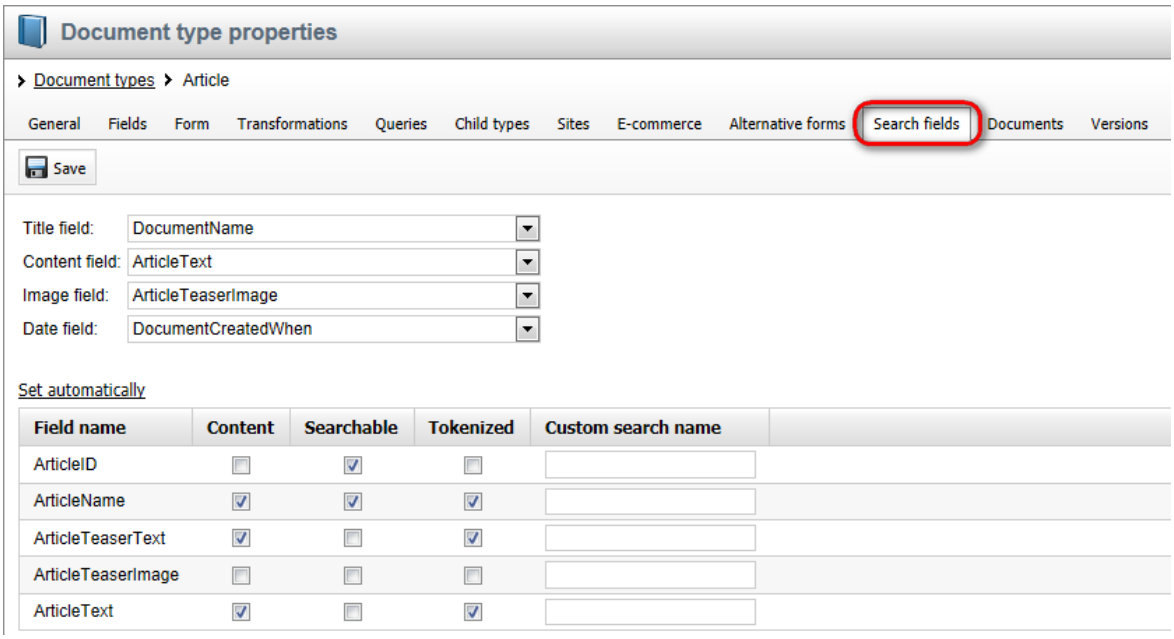
More information about alternative forms can be found in the Alternative forms chapter in the **Modules** section of this guide.



Search fields

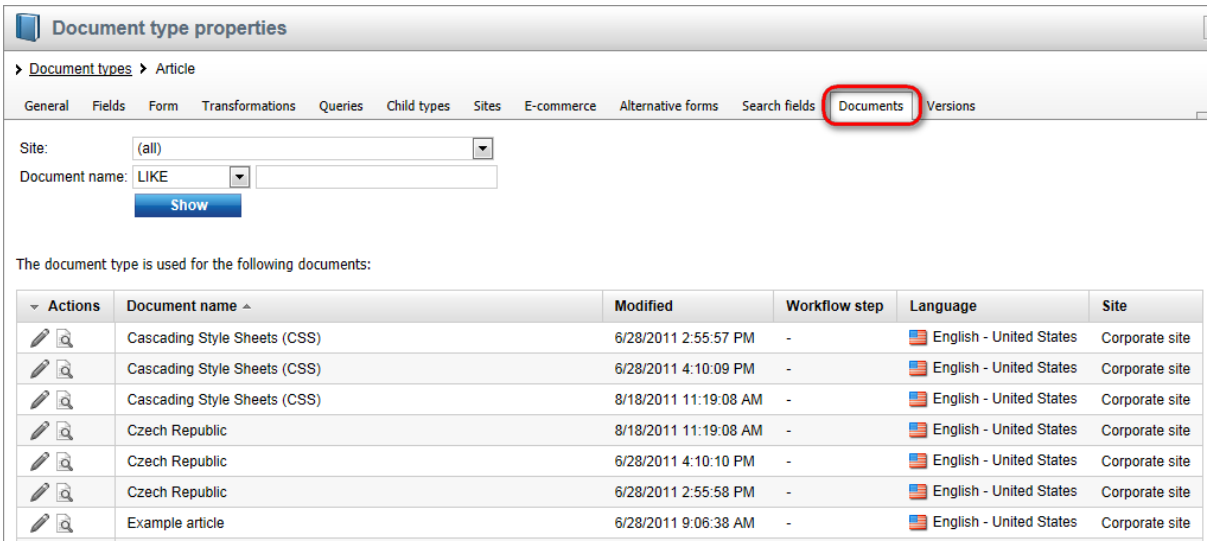
On the **Search fields** tab, you can define how data stored in the document type will be indexed by the **Smart search** module. In the top part, you can specify how documents of the given type will be displayed in search results. Lines of the table in the bottom part of the tab represent document fields defined on the Fields tab, while columns correspond to **Smart search** properties. Please note that you can use the **Set automatically** link to have the table configured automatically.

For more details on how the content of documents of this document type and of the whole website is searched, please refer to the Smart search chapter.



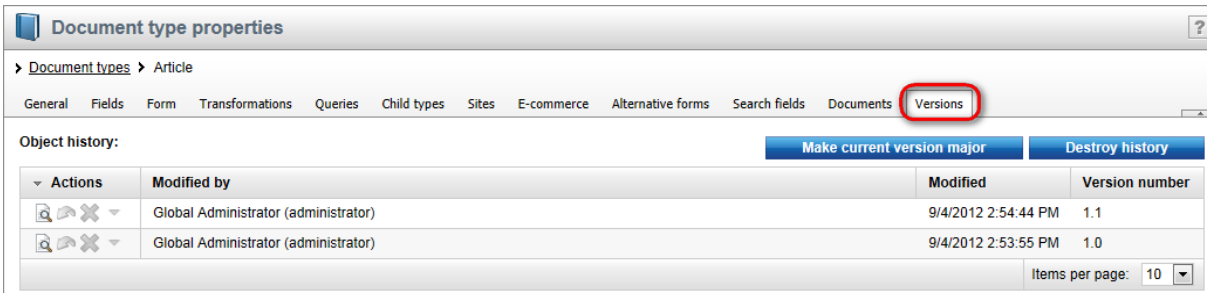
Documents

By switching to the **Documents** tab, you can view a list of all documents of the currently edited document type. Using the **Site** drop-down list, you can choose whether to display all documents of the type in the system or only on a particular site. The **Document name** fields allow you to search for documents that match the entered parameters.



Versions tab

On the **Versions** tab, you can see all versions of the currently edited document type. You can find more details in the Development -> Object versioning section.



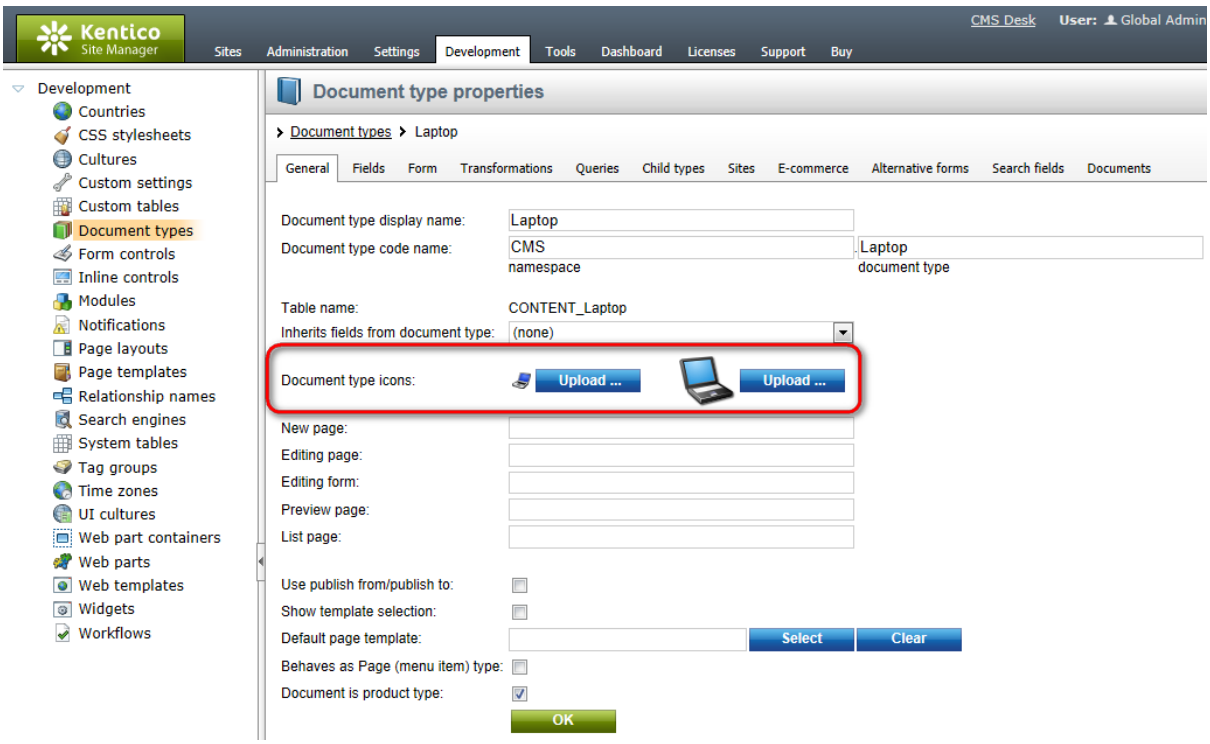
7.7.4 Uploading document type icons

Kentico CMS allows you to upload your own icons and associate them with document types. This can be useful if you create custom document types or if you want particular document types to be more easily recognizable in various parts of the CMS.

Please note

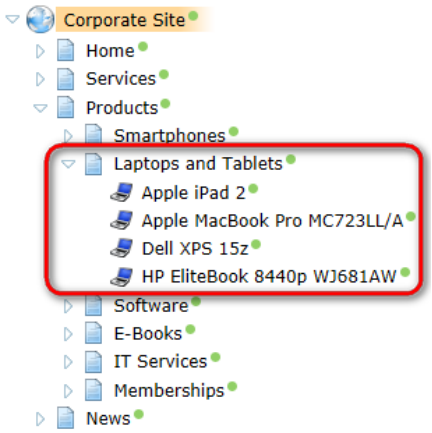
When importing and upgrading your custom document types using an export package, the associated icons are also included in this package.

1. If you need to add your own icon to a particular document type, go to **Site Manager -> Development -> Document types -> Edit ( ) Document type** and in the **Document type properties** user interface switch to the **General** tab. You should see a page similar to the one depicted in the following screenshot:

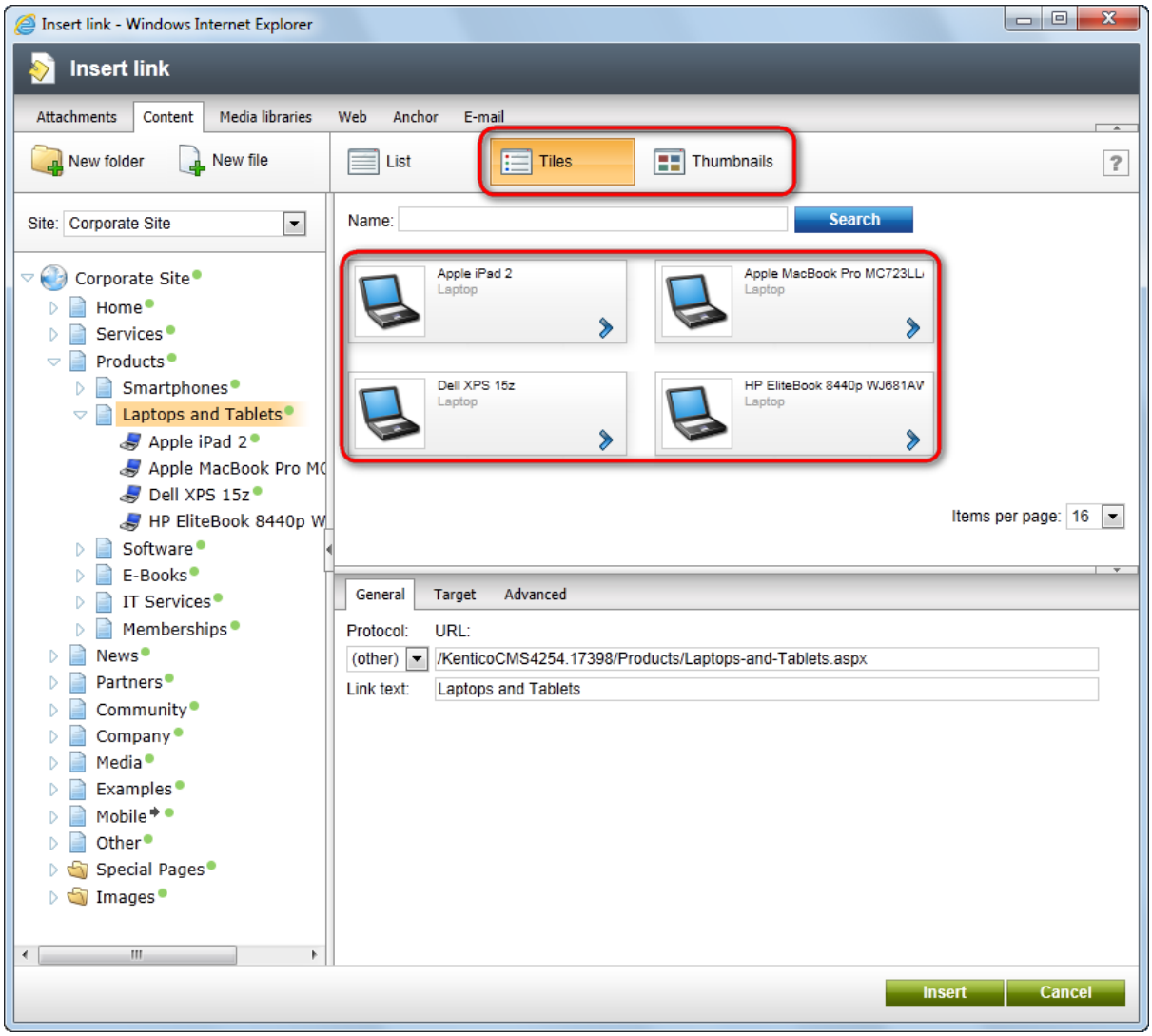


2. Click one of the **Upload** buttons and select a **.png image** from your local or network file system. Its size will be changed automatically. Click **OK**. From now on, this image will be associated with the given document type.

The smaller document type icons (16x16) are used in various parts of Kentico CMS, e.g. in the content tree in **CMS Desk -> Content**:



The larger document type icons (48x48) are used in the Insert link webpage dialog on the **Tiles** and **Thumbnails** tabs:



Please note

You can associate just one image file with a document type. However, the same icon will be used in the **Insert link** dialog and elsewhere in the CMS.

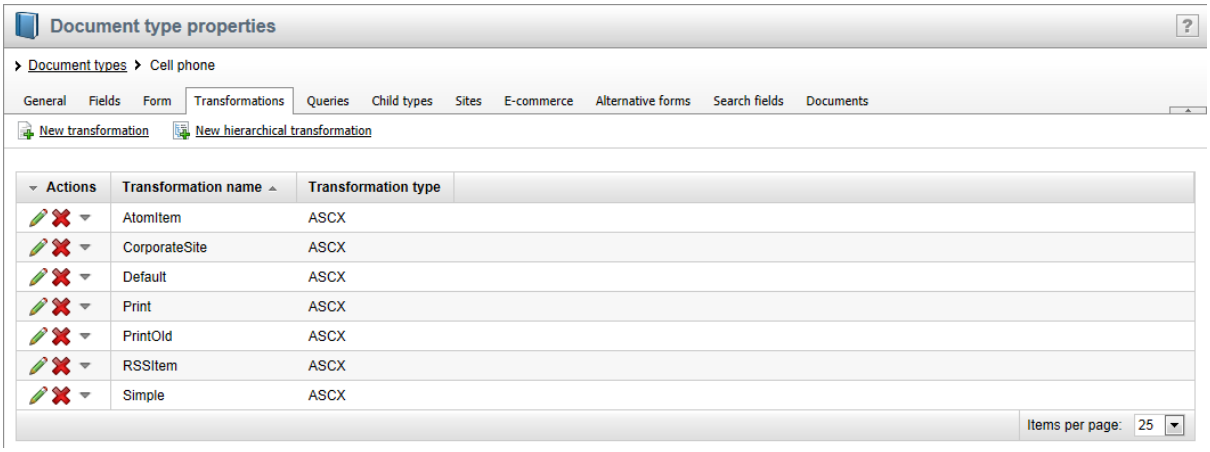
7.7.5 Transformations

7.7.5.1 Overview

Transformations are pieces of code that determine how Kentico CMS documents, or certain parts of them, are rendered by listing web parts and controls. They take raw data from the Kentico CMS database and transform it into the form you wish it to appear in. This makes them a crucial tool when displaying documents and document related data on the pages of your website.

Transformations can be accessed in **Site Manager -> Development -> Document types -> Edit ( )**

a *document type*, on the **Transformations** tab.



- To learn about the functionality of transformations and to see examples of their code, please refer to the Transformations topic.
- To see an example of writing transformations for a particular document type, please refer to the Writing transformations topic.
- To learn how to display documents in a hierarchical structure, please refer to the Hierarchical transformations topic.
- To learn how to add context menus to web parts displaying users or groups, please refer to the Context menus in transformations topic.
- To learn how to add your own functions to transformations, please refer to the Adding custom functions to transformations topic.
- Transformations may also be applied to documents (and other data) loaded through macro expressions. Please see the Transformations in macro expressions topic for details and examples.

7.7.5.2 Transformations

The functionality of transformations is very similar to that of templates used by standard ASP.NET list controls such as the Repeater. The main difference is that our transformations are stored separately in the database and can easily be used repeatedly. They can be assigned to listing web parts or controls through the appropriate **Transformation** properties. The use of transformations is supported by all web parts that can display document data, as well as by those listing controls that are designed to work directly with Kentico CMS documents.

There are several different approaches that you can use when writing transformations. You can select one of the following transformation types, which determines how the system renders the transformation's code:

- **ASCX** - with this option, the code of the transformation supports ASCX markup, i.e. the same syntax that you would use to edit a standard web form or user control, including inline code, embedded controls, standard ASP.NET data binding expressions and special methods designed for use in transformations. You can access the fields of the transformed document through standard data binding expressions in format: *<%# Eval("ColumnName") %>*

- **Text/XML** - the system processes the code only as basic HTML. This means that any ASCX markup, such as controls or inline code, are not functional when the transformation is rendered. You may use Kentico CMS Macro expressions and methods to insert dynamic values into the content. Expressions in the following format allow you to easily get the values of the given document's fields:

{%ColumnName%}

- **HTML** - works the same way as the **Text/XML** option, but editing is done through the WYSIWYG editor. The rendered output of HTML code will be shown inside the editor.
- **XSLT** - this option can be selected for transformations that use XSL elements to render the data. The code must be in valid XML format.
- **jQuery** - these transformations provide a way to define the jQuery templates used by the Chat module.

For security reasons, the code of *ASCX* type transformations may only be edited by users who have the **Edit ASCX** code permission for the **Design** module. This permission can only be assigned by global administrators.

Since text-based transformations (**Text/XML** or **HTML** types) are only processed as basic HTML, they cannot be used to compromise the security of the website. Another advantage of these transformation types is that they do not need to be compiled, which means they may be used and modified even if the Virtual path provider is not available, such as in a precompiled or medium trust environment.

Transformations are categorized under the document types that they are supposed to display. They can be managed in the Kentico CMS administration interface at **Site Manager -> Development -> Document types -> ... Edit document type ... -> Transformations**. Some document types do not represent an object but serve only as a container for transformations and queries.

The full name used to identify transformations uses the following format: **<document type code name>.<transformation name>**

The sample sites include many transformations for all document types and you can modify them or write new transformations to suit any of your requirements.

Example

The code of the **CorporateSite.Transformations.ProductList** ASCX transformation, which is used to display lists of products on the sample Corporate Site, looks like this:

```
<%@ Register Src="~/CMSModules/Ecommerce/Controls/ProductOptions/
ShoppingCartItemSelector.ascx" TagName="CartItemSelector" TagPrefix="uc1" %>
<div class="ProductPreview">
  <div class="ProductBox">
    <div class="ProductImage">
      <a href="<%# GetDocumentUrl() %>" style="display:block;">
<%# EcommerceFunctions.GetProductImage(Eval("SKUImagePath"), 180, Eval
("SKUName"))%>
      </a>
    </div>
    <a href="<%# GetDocumentUrl() %>">
      <span class="ProductTitle textContent">
        <%# HTMLEncode(ResHelper.LocalizeString(Convert.ToString(Eval
("SKUName")))) %>
      </span>
    </a>
```

```
    <div class="ProductFooter">
      <div class="productPrice"><%# GetSKUFormattedPrice(true, false) %></div>
<uc1:CartItemSelector id="cartItemSelector" runat="server"
SKUID='<%# ValidationHelper.GetInteger(Eval("SKUID"), 0) %> '
SKUEnabled='<%# ValidationHelper.GetBoolean(Eval("SKUEnabled"), false) %> '
AddToCartImageButton="btn_addToShoppingCart.png" ImageFolder="~/App_Themes/
CorporateSite/Images/" />
    </div>
  </div>
</div>
```

When this transformation is assigned to a listing web part or control that has products (SKUs) in its data source, the output HTML code of individual products contains the values returned by the methods and data binding expressions, like in the following example (please note that a part of the code is omitted to save space):

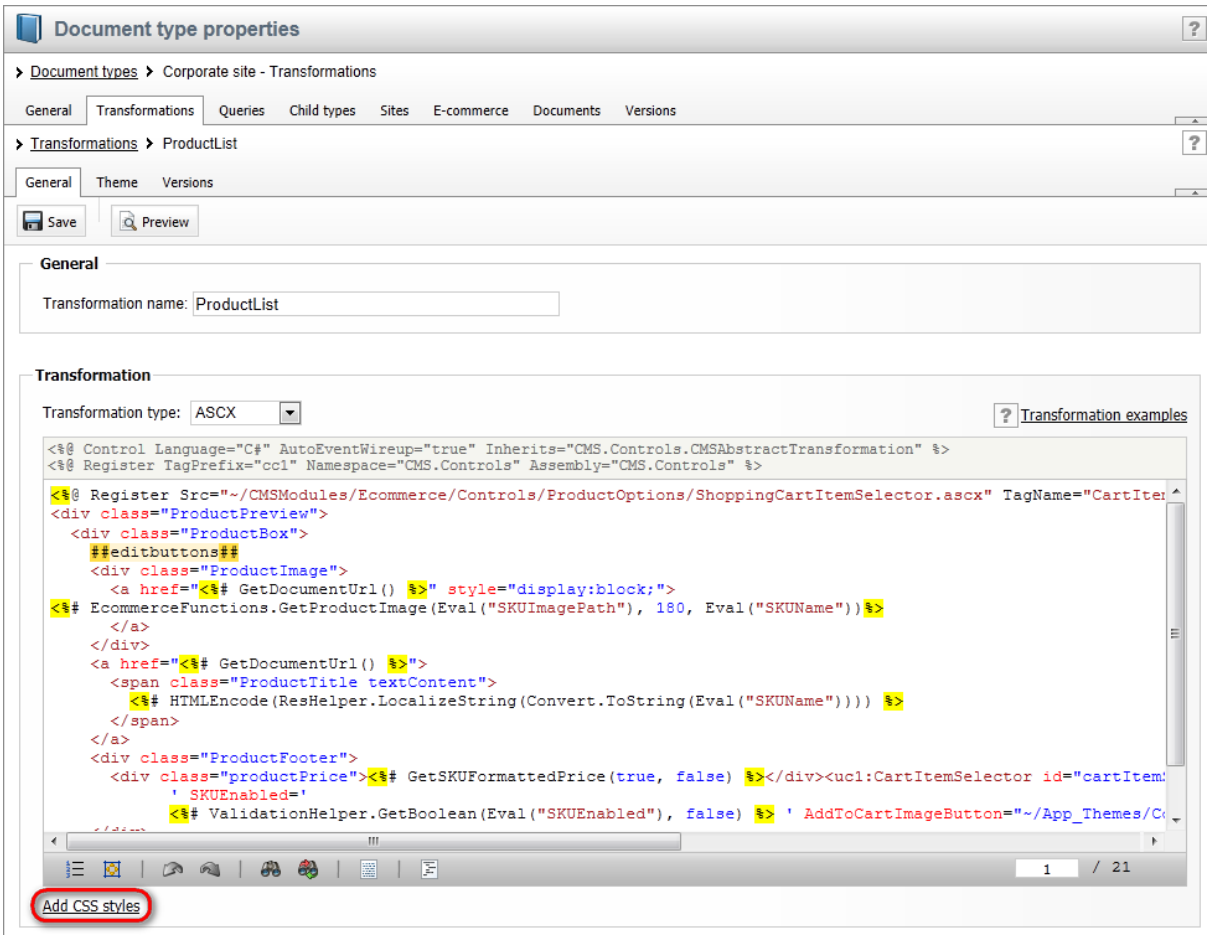
```
<div class="ProductPreview">
  <div class="ProductBox">
    <div class="ProductImage">
      <a href="/KenticoCMS/Products/Laptops-and-Tablets/Apple-iPad-2.aspx"
style="display:block;">
<img alt="Apple iPad 2" src="/KenticoCMS/getmetafile/f1e09daf-f8c5-4252-ba7b-
222d33e01ff8/ipad2_product.aspx?maxsidesize=180" border="0" />
      </a>
    </div>
    <a href="/KenticoCMS/Products/Laptops-and-Tablets/Apple-iPad-2.aspx">
      <span class="ProductTitle textContent">
        Apple iPad 2
      </span>
    </a>
    <div class="ProductFooter">
      <div class="productPrice">$510.99</div>

      ...

  </div>
</div>
```

The final output of this product on the website then looks like this:

Adding CSS styles to transformations

The CSS classes used in the transformation code can either be defined in the stylesheets used by the website and individual pages, or added directly to the transformation object. To do this:

1. Edit the transformation on its **General** tab.
2. Click Add CSS styles below the code editor.
3. The **CSS styles** field appears, where you can add any required CSS classes.



4. Click **Save**.

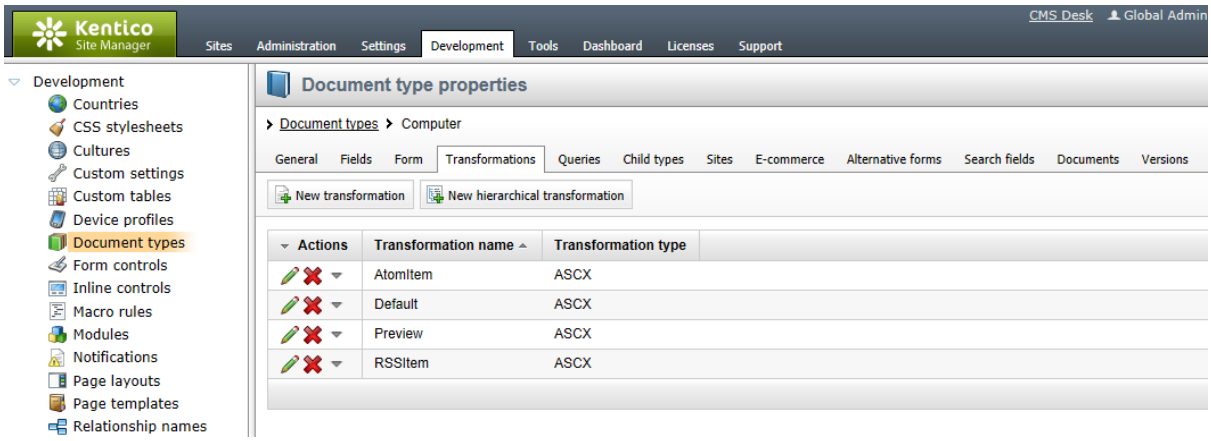
If the styles require any additional files (such as images), you can add them on the **Theme** tab.

For more information about page component CSS styles, see the Development -> CSS stylesheets and design -> CSS for page components topic.

7.7.5.3 Writing transformations

In this topic, you will learn how to write transformations for the Computer document type created according to the instructions in the Defining a new document type topic.

Typically, transformations are stored under the document type they are meant to display. Go to **Site Manager -> Development -> Document types**, edit (✎) the **Computer** document type and open its **Transformations** tab. This is the main management interface for the transformations of the given document type.



The document type creation wizard has automatically generated several default transformations.

Preparing the source documents

Before you can use the transformations, you first need to add some computer documents to the website.

1. Go to **CMS Desk -> Content**, select the root of the site and click **New**.

2. Choose the **Page (menu item)** document type.

3. Type in *Computers* as the **Page name** and select the **Create a blank page** template option. Click **Save** to create the page.

4. Switch to the **Design** tab of the new page and add a **Listing and viewers -> Datalist** web part into *zoneA*.

5. Set the following property values for the Datalist web part:

- **Document types**: *custom.computer*; you can choose the document type from a list by clicking the **Select** button.
- **Transformation**: *custom.computer.preview*; to easily choose from a list of available transformations, click **Select**, choose the *Computer* document type in the dialog and then click on the required transformation.
- **Selected item transformation**: *custom.computer.default*

6. Click **OK** to insert the web part. It will be used to display the data of computer documents on the page according to the specified transformations. For now, the page is empty, because there are no documents of the *custom.computer* type on the website.

7. Click **New** on the main CMS Desk toolbar with the *Computers* page selected and choose the **Computer** document type. This opens an editing form with the fields of the computer document type.

8. Enter the following values:

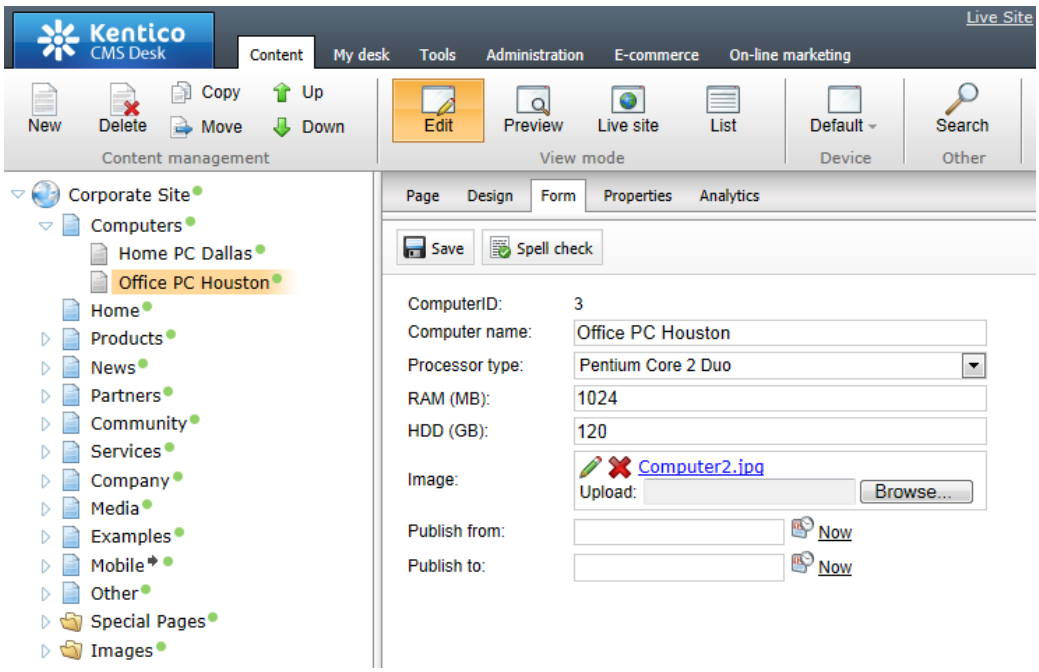
- **Computer name**: Home PC Dallas
- **Processor type**: Athlon
- **RAM (MB)**: 2048

- **HDD (GB)**: 160
- **Image**: Upload an image from your local disk using the *Browse* button (you can find sample images in the *<Kentico CMS setup directory>\CodeSamples\SampleWebTemplate\Computer_Images* folder) .

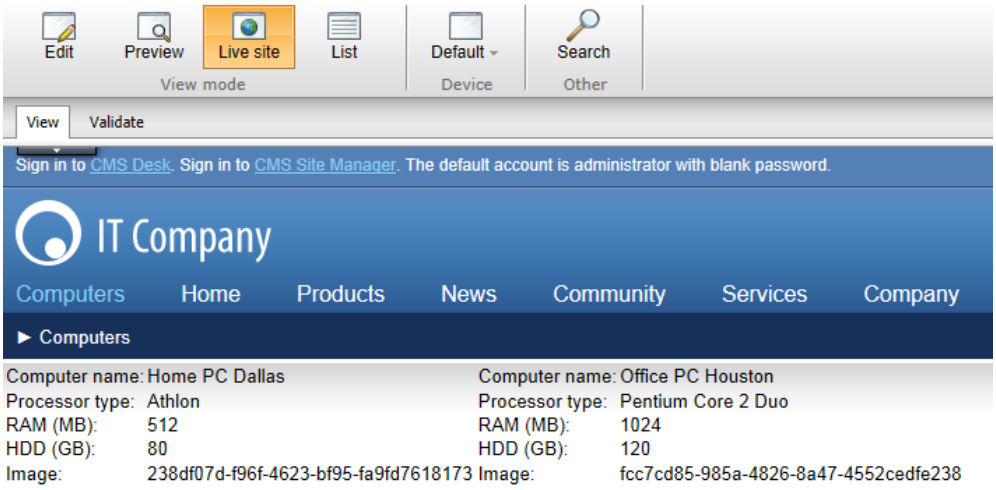
9. Click **Save and create another** and enter the following values for the second computer document:

- **Computer name**: Office PC Houston
- **Processor type**: Pentium Core 2 Duo
- **RAM (MB)**: 4096
- **HDD (GB)**: 200
- **Image**: Upload another image.

Click **Save**.



Select the **Computers** page again and view it in **Live site** mode. The data of the two computer documents will be displayed as shown below:

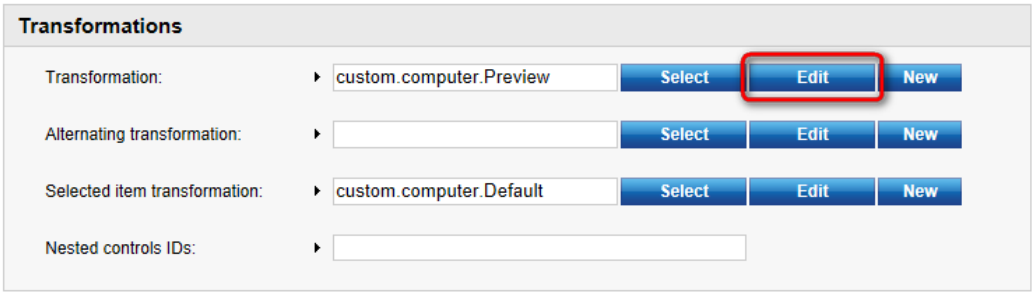


Editing the transformations

The format of the data on the *Computers* page is determined by the automatically generated default transformation. You can fully customize the data format by modifying the code of the transformations through the main interface in Site manager. CMS Desk also allows you to edit transformations, which is more convenient in many cases.

1. Return to **Edit** mode and open the **Design** tab of the *Computers* page.

2. **Configure** (⚙) the Datalist web part, scroll down to the **Transformation** property and click the **Edit** button.



The editing dialog of the *custom.computer.Preview* transformation opens (the same dialog that you would get when editing the transformation in Site Manager). The Datalist web part uses this transformation for displaying computers in list mode.

3. Delete the default code and enter the following instead:

```
<div style="text-align:center;padding: 8px;margin: 4px;border: 1px solid #CCCCCC">
  <h2>
    <a href="<%# GetDocumentUrl() %>"><%# Eval("ComputerName") %></a>
  </h2>
  <%# GetImage("ComputerImage", 120) %>
```

```
</div>
```

Because this is an **ASCX** type transformation, the code is similar to standard ItemTemplates that you may already be familiar with from using ASP.NET Repeater or DataList controls. It combines HTML with ASP.NET commands and data binding expressions. You may also use built-in methods that simplify various tasks. For more information about the available transformation methods, click the Transformation examples link above the code editor.

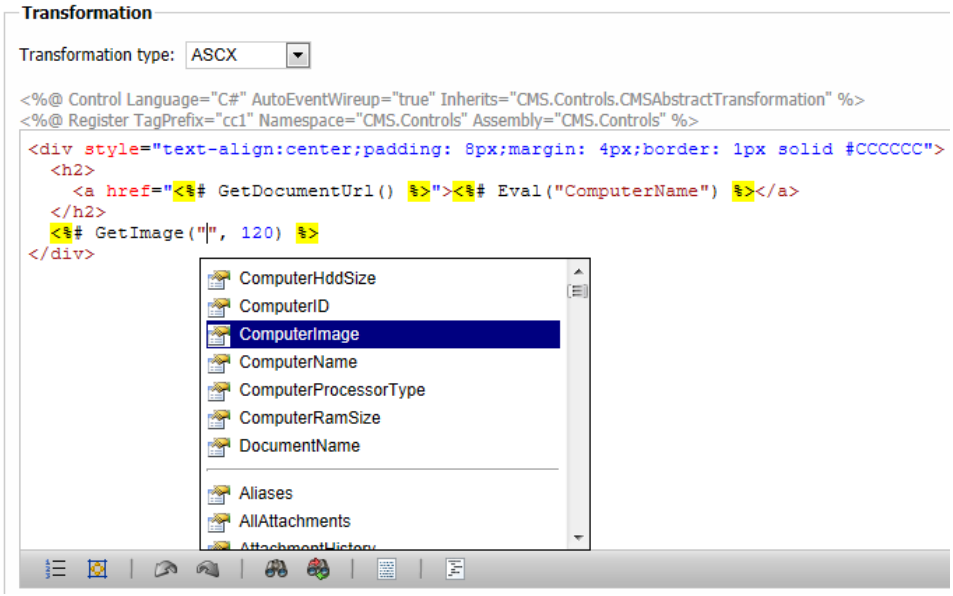
Note the code used to create the link to specific documents. It consists of a standard HTML link tag and inserts the appropriate URL and link text dynamically:

```
<a href="<%# GetDocumentUrl() %>"><%# Eval("ComputerName") %></a>
```

You can generate an image tag containing the file uploaded into the given document's **ComputerImage** field using the **GetImage** method. The sample code calls the method with a parameter that ensures automatic server-side resizing of the image's longest side to 120 pixels:

```
<%# GetImage("ComputerImage", 120) %>
```

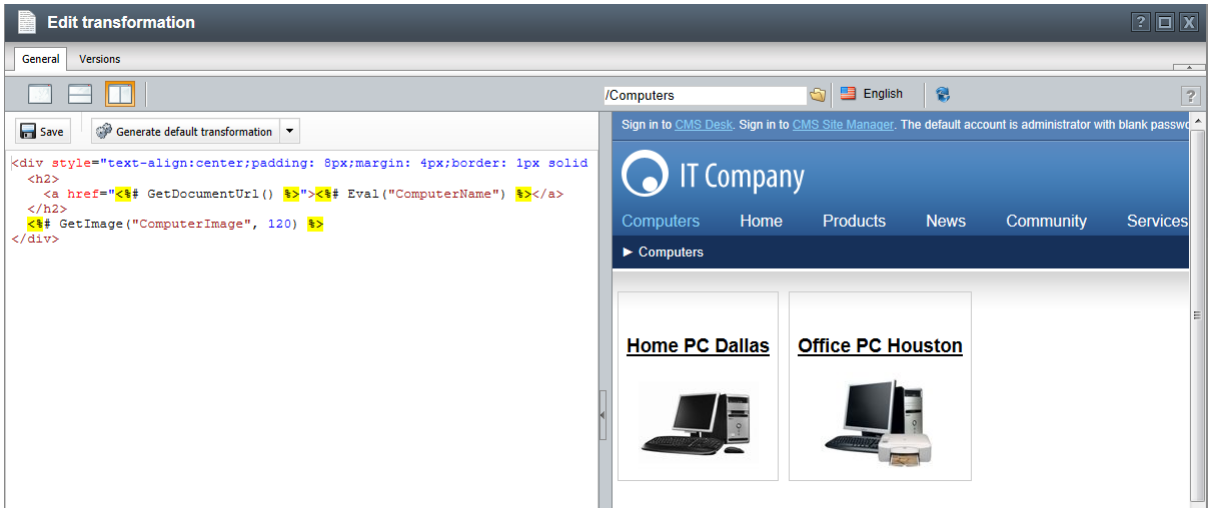
When writing ASCX transformations, you often need to specify the names of data fields as parameters of the *Eval* data binding expression or other methods, such as *ComputerName* and *ComputerImage* in the example above. You can either type the names manually, or press the **CTRL + SPACE** key combination to access a list of available document fields and related objects.



By clicking on an item in the list or selecting it and pressing enter, you can insert it to the current cursor position in the code. The specific fields of the given document type are prioritized at the top.

4. Click **Save** to apply the changes in the code.

5. Click the **Preview** button to see how the new transformation affects the page. This opens a split view that allows you to check the appearance of the web part directly while editing the transformation code. The computer documents should now be displayed as shown below.



Next, define the transformation used for viewing the details of computer documents.

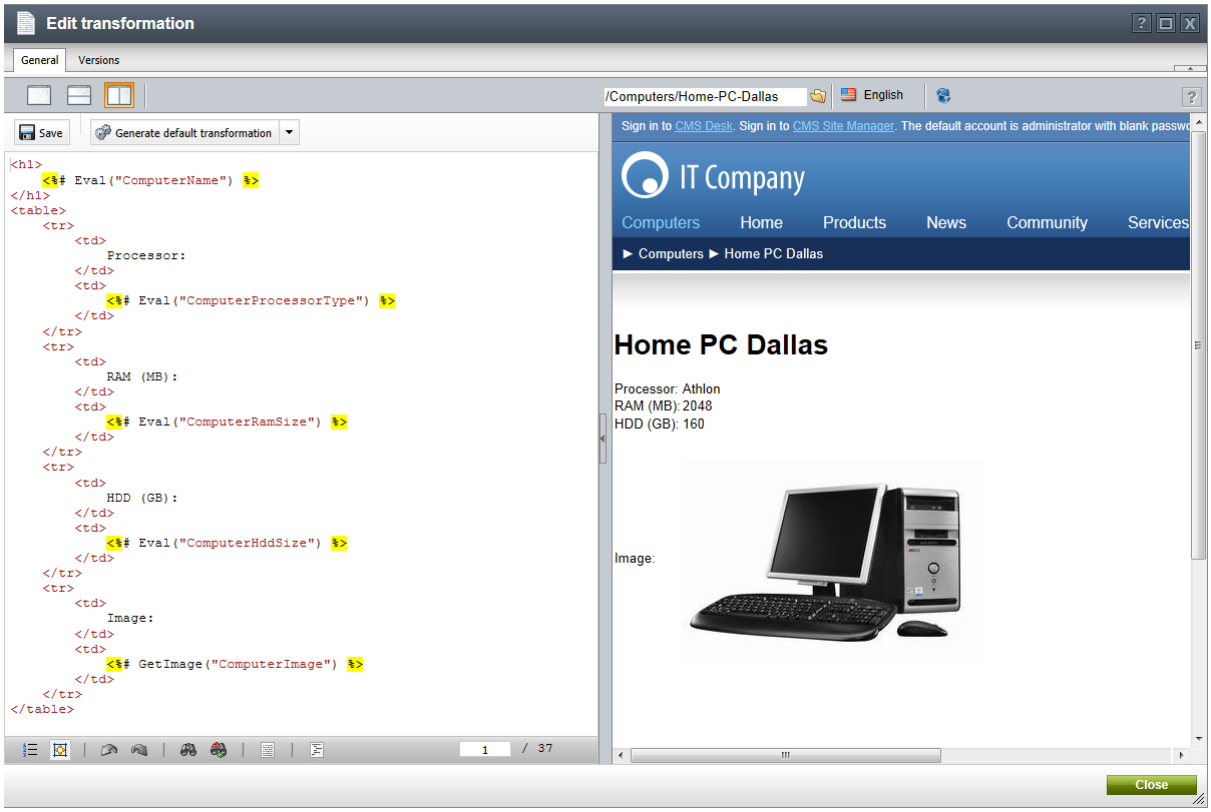
6. Close the **Edit transformation** dialog and click the **Edit** button next to the *custom.computer.Default* transformation specified in the **Selected item transformation** property of the Datalist web part. Remove the original code and enter the following:

```
<h1><%# Eval("ComputerName") %></h1>

<table>
<tr>
<td>
Processor:
</td>
<td>
<%# Eval("ComputerProcessorType") %>
</td>
</tr>

<tr>
<td>
RAM (MB):
</td>
<td>
<%# Eval("ComputerRamSize") %>
</td>
</tr>

<tr>
<td>
HDD (GB):
</td>
<td>
```

```
<%# Eval("ComputerHddSize") %>
</td>
</tr>

<tr>
<td>
Image:
</td>
<td>
<%# GetImage("ComputerImage") %>
</td>
</tr>
</table>
```

7. Click **Save**.

The web part applies the *Selected item transformation* when one of the displayed documents is the currently active page, e.g. when a visitor clicks on the link in the titles of the computers on the **Computers** page. To see how the detail view looks like for a specific computer document, enter */Computers/Home-PC-Dallas* into the path textbox on the preview toolbar and **Refresh** () the page section.



If you close the configuration dialogs and view the *Computers* page on the live site, you will see that both the list of computers and the detail pages of individual computer documents are rendered according to the new data format.

You have learned how to write *ASCX* transformations for displaying the content of structured documents. Other types of transformations may be used in the same way, only with different transformation code syntax.

Please note: If you wish to use *XSLT* transformations, it is necessary to display the data through the XSLT viewer or Universal document viewer web parts (or **CMSViewer** server control), otherwise the transformation will not work.

Transformations for multilingual websites

In some cases, you may need to display different text in transformations based on the currently selected language. If you are using the built-in multilingual support, you can achieve this by creating a separate transformation for each language, using the appropriate culture code as a suffix in the transformation name.

Example:
English (default language) transformation code name: *cms.news.detail*
French transformation code name: *cms.news.detail_fr-fr*

When a user switches the content culture to French, web parts and controls automatically load the French version of the transformation.

7.7.5.4 Hierarchical transformations

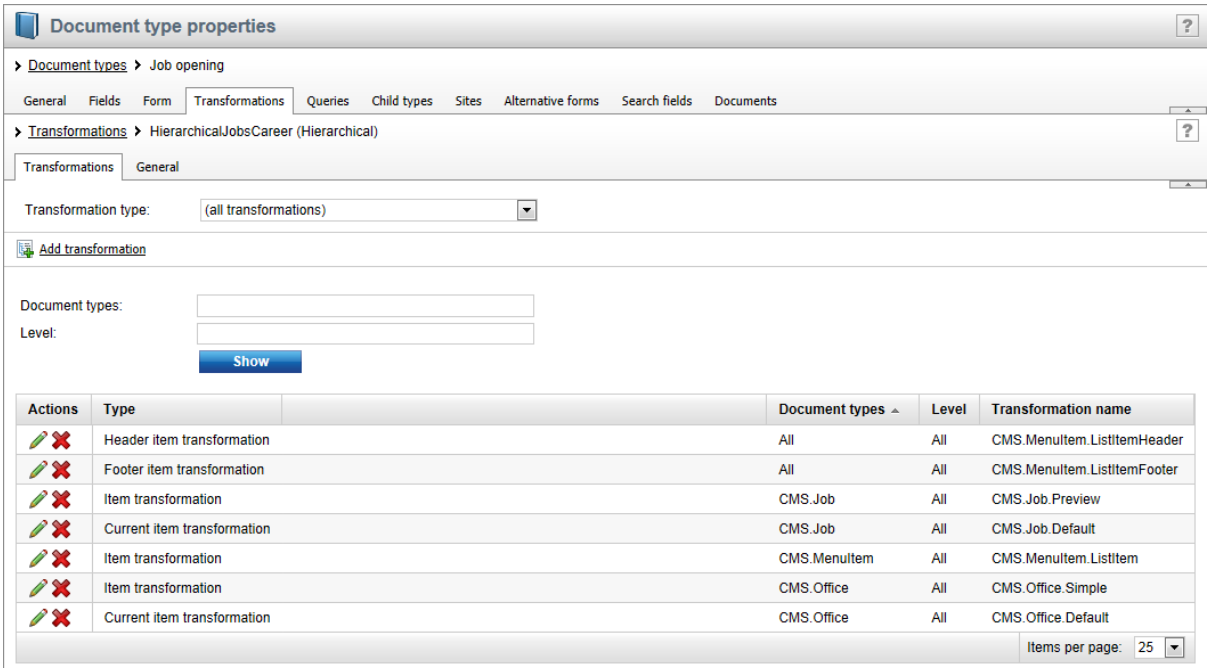
Hierarchical transformations are special types of transformation that can be used to display documents (and other types of appropriately organized data) in a hierarchical structure. These transformations can affect multiple document types and may be customized differently for every level of the hierarchy (e.g. levels of the document content tree). Currently, they are meant to be used by the following web parts from the **Listings and viewers** category:

- Hierarchical viewer
- Universal viewer
- Universal viewer with custom query

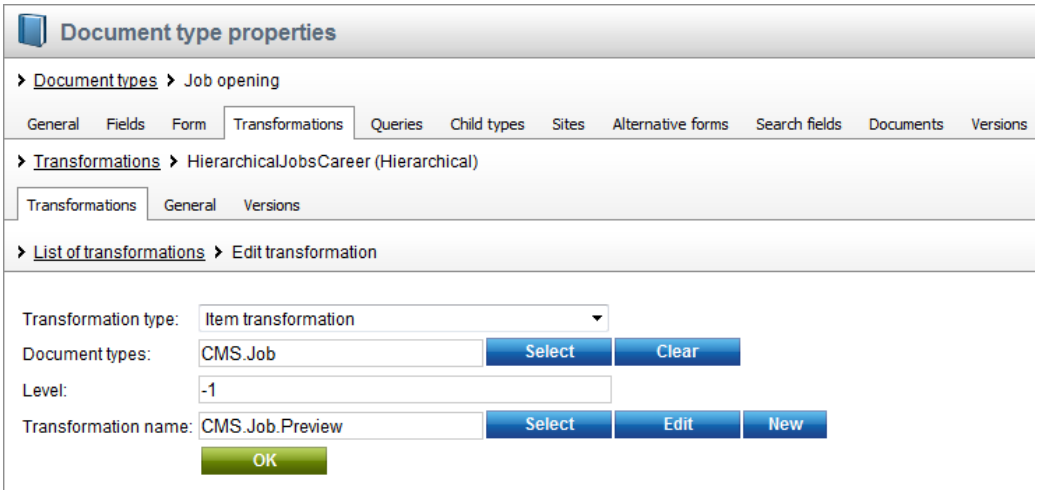
As well as the following Kentico CMS controls:

- UniView
- BasicUniView
- CMSUniView
- QueryUniView

They can be managed in the same way as standard transformations by editing a document type on its **Transformations** tab. However, a hierarchical transformation is not directly defined by code. Instead, it serves as a container for a number of standard transformations, which are then applied to the appropriate parts of the source data when it is rendered. A proper hierarchical transformations can be composed of many sub-transformations that define its layout and the content or format of the displayed items.



You can **Add** (), **Edit** () and **Delete** () these sub-transformations. The delete action does not remove the actual transformation from the system (only from the given hierarchical transformation). The purpose of each transformation depends on its individual settings, which can be configured when it is added or later via the edit action.



A **Transformation type** can be selected for each sub-transformation from among the following options:

- **Item transformation** - applied to all displayed items that are not covered by a specialized transformation type (e.g. alternating items, first items etc.).
- **Alternating item transformation** - applied to items that have an even position in the listing order. Every level in the hierarchy has its own separate alternation pattern.
- **First item transformation** - applied to the first item on every level in the hierarchy. Only works for levels that contain more than one item.
- **Last item transformation** - applied to the last item on every level. Only works for levels that contain more than one item.

- **Header transformation** - rendered at the beginning of every level (before the first item on the level). These transformations provide a convenient way to visually separate or style individual levels.
- **Footer transformation** - rendered at the end of every level (after the last item on the level). Can be used to close encapsulating elements from the *Header*.
- **Single item transformation** - applied in cases where there is only one item on a level in the hierarchy.
- **Separator transformation** - rendered between items on the same level. It is not placed between items on different hierarchy levels (i.e. between a parent item and its child).
- **Current item transformation** - applied to the currently selected item (i.e. the document that is being viewed). Please note that it is always necessary to specify a *Document type* (or types) for this kind of transformation.

If there are multiple transformations of different types included in the hierarchical transformation, more than one may be applicable to a single item. In these cases, the item will be displayed using the transformation type with the highest priority, according to the following order:

1. Current item (top priority)
2. First/Last/Single item
3. Alternating item
4. Item

It is also possible to restrict which **Document types** should be affected by the given transformation. You can specify multiple document types by entering their code names separated by semicolons, e.g. *CMS.News;CMS.Article*. Leaving this field empty means that the transformation will be applied to all document types. Transformations that are dedicated to specific document types have a greater priority than those set to include all types. The field has no effect for *Header*, *Footer* or *Separator* transformation types.

The **Level** field sets the level in the hierarchy from which the transformation should be applied. For example, if *2* is entered, the transformation will affect items on the third level (the number of the first level in the hierarchy is *0*) and will also be **inherited** by all levels below it. Leaving an empty value or entering *-1* specifies all levels. Levels are always counted from the start of the particular data hierarchy that is being displayed. In the case of documents, this means that level *0* represents the first level under the selected path, not the root of the entire website.

Breaking level inheritance

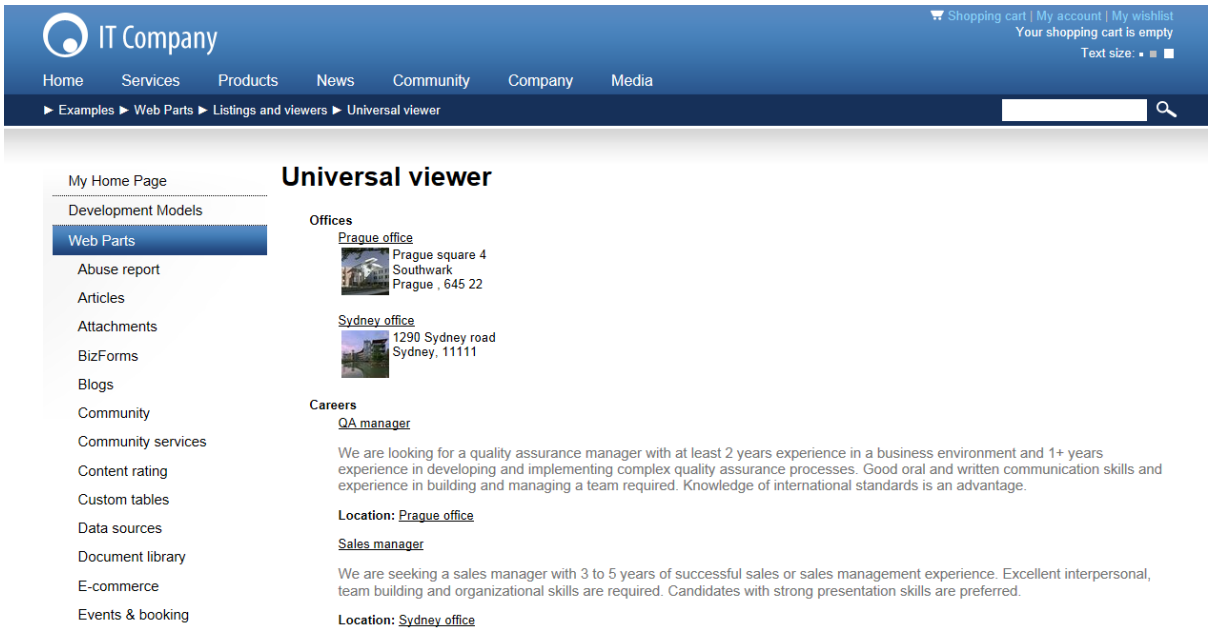
Please note that you can override the level inheritance by adding another transformation for a lower level (it must use the same transformation type and affect the same document types).

Finally, it is necessary to set the actual transformation in the **Transformation name** field and it will be used for the defined purpose. Please refer to the Transformations topic to learn what options are available when creating or editing standard transformations.

By adding transformations as components and configuring their fields according to the descriptions above, you can create complex hierarchical transformations suitable for displaying any kind of hierarchical data.

You can find an example of a *Universal viewer* web part using a hierarchical transformation on the Corporate Site sample website in the **Examples -> Web Parts -> Listings and viewers -> Documents**

-> Universal viewer section of the content tree.



Data requirements for hierarchical transformations

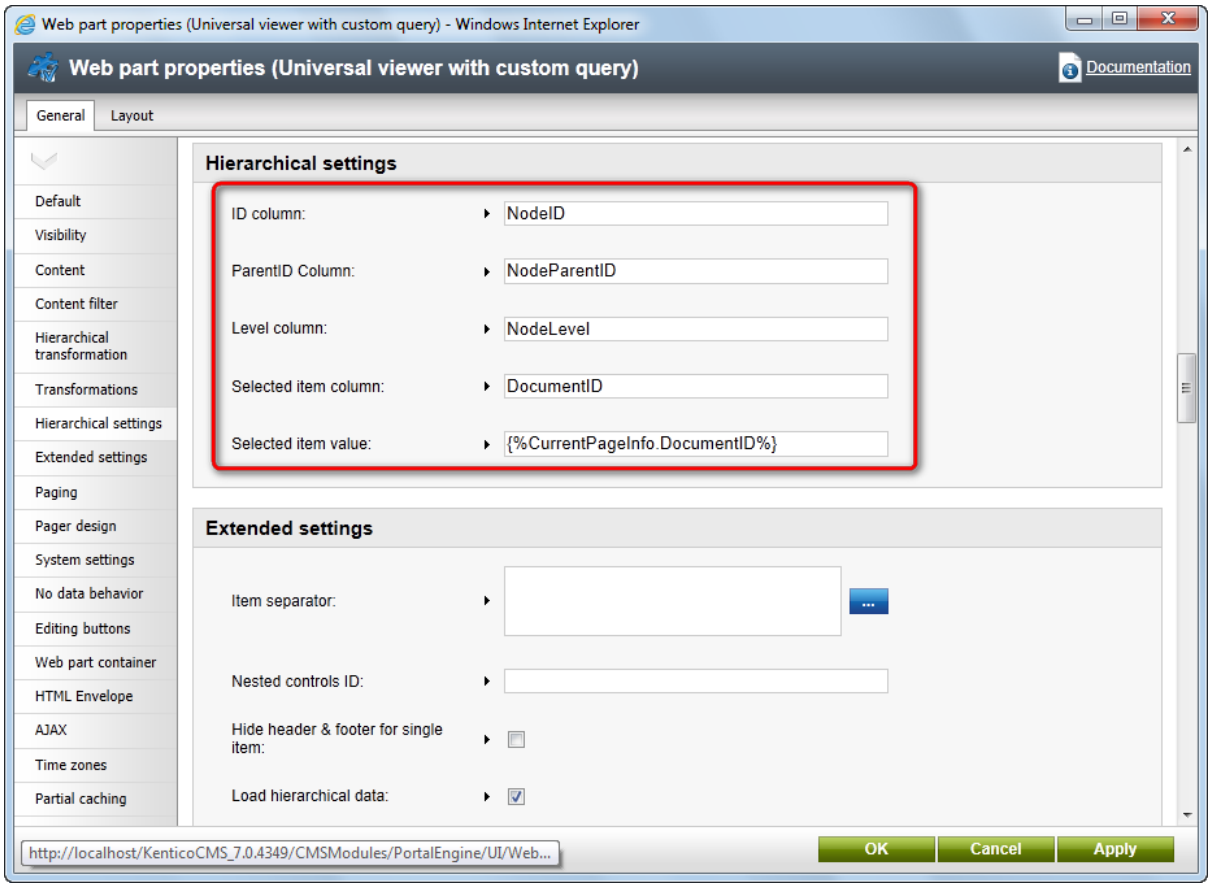
If you wish to use a hierarchical transformation, the source data must be organized in the appropriate hierarchical format. This can be ensured by enabling the **Load hierarchical data** property, which is available for all supported web parts and controls. Please note that paging is not possible in this scenario, because the separation of data into pages would break the hierarchy.

Additionally, all records in the source data need to contain a value that determines their level in the hierarchy and must be connected through parent-child relationships. If any parent records are missing from the used data (for example due to content filtering, WHERE conditions etc.), the disconnected parts of the hierarchy will not be displayed. The most common example of a suitable type of hierarchical data are documents in the Kentico CMS content tree.

However, the *Universal viewer with a custom query* web part (or *QueryUniView* control) may also be used to load other types of data from the database, as long as they meet the given requirements. In this case, you need to specify how the data retrieved by the query should be organized into a hierarchy via the properties in the **Hierarchical settings** section:

The **ID column** and **ParentID column** properties are used to set up the parent-child relationships of the data. The *ID column* needs to be a unique identifier for records. The value of a record's *parentID column* must match the ID of the item that should serve as its parent in the hierarchy (it should never be *null* for any of the items in the data source). Then, the name of the column whose value determines what level an item belongs to needs to be entered into the **Level column** property.

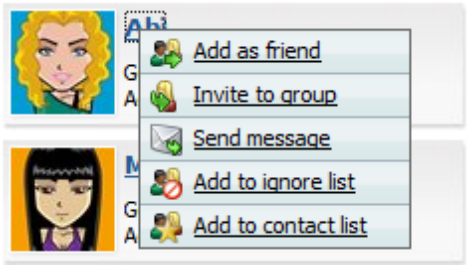
If you also wish to set up special behavior for the displayed items that reflects some type of selection made by the current user, you can enter the name of a column into the **Selected item column** property (the values of this column must be unique for every record to ensure correct functionality). Typically, you will then need to insert a Macro expression as the **Selected item value** in order to dynamically retrieve the appropriate value from the current context. The item whose value in the specified column matches the result of the macro will be treated as the currently selected item.



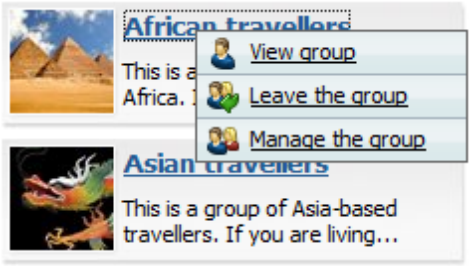
7.7.5.5 Context menus in transformations

When writing transformations for a web part displaying **Users** or **Groups**, you can enclose the transformation in a menu container control, which ensures displaying context menus after right-clicking on a user or group. You can see a live example of these context menus on the sample **Community Starter site**:

In the screenshot below, you can see the context menu displayed when you right-click one of the users listed in the **Members** section:



The following screenshot shows the context menu displayed when you right-click one of the groups listed in the **Groups** section:



How is this achieved? As you can see when you view the transformation code used in the **Users viewer** or **Groups viewer** web parts, you need to have enclosed your transformation in the **cms:usermenucontainer** or **cms:groupmenucontainer** control:

Users

```
<cms:usermenucontainer runat="server" ID="userMenuElem" MenuID="userContextMenu"
Parameter='<%# Eval("UserID").ToString() %>' ContextMenuCssClass="UserContextMenu"
 >

... transformation code ...

</cms:usermenucontainer>
```

Groups

```
<cms:groupmenucontainer runat="server" ID="groupMenuElem"
MenuID="groupContextMenu" Parameter='<%# Eval("GroupID").ToString() %>'
ContextMenuCssClass="UserContextMenu" >

... transformation code ...

</cms:groupmenucontainer>
```

Please note

It is only possible to add controls into transformations that use the **ASCX** type. This means that the context menu controls will not be rendered correctly by the other transformation types.

Modifying context menu design

The default controls used for context menus are stored in **<web project>\CMSAdminControls\ContextMenus**:

- **GroupContextMenu.ascx**
- **UserContextMenu.ascx**

These two controls are used automatically for the Group or User context menus. If you want to modify the design of the context menus, you can edit these controls in Visual Studio.

You can also develop your custom controls for this purpose. In this case, you need to include the **MenuControlPath** parameter in the *cms:usermenucontainer* or *cms:groupmenucontainer* controls in the transformation and set its value to the path to your control:

```
<cms:groupmenucontainer runat="server" ID="groupMenuElem"
MenuID="groupContextMenu" Parameter='<%# Eval("GroupID").ToString() %>'
ContextMenuCssClass="UserContextMenu"
MenuControlPath="~\CMSAdminControls\ContextMenus\MyGroupContextMenu.ascx" >

... transformation code ...

</cms:groupmenucontainer>
```

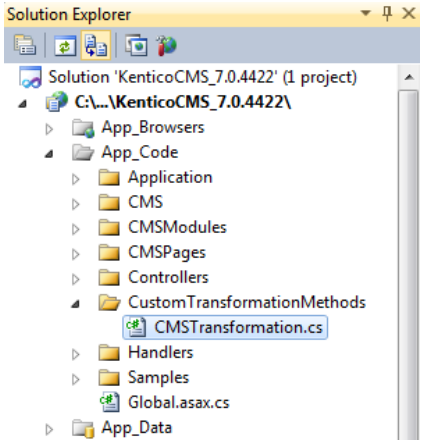
7.7.5.6 Adding custom functions to transformations

In many cases, you may need to add custom logic into your transformations. This can be achieved by implementing custom methods and calling them in the transformation code. Using this approach, you can process field values, display data in a specific format, add custom conditions etc.

The following example demonstrates how to create a custom method that trims text values to a specified number of characters, and shows how to use it in an ASCX transformation:

1. Open your Kentico CMS web project in Visual Studio. Create a new folder under the **App_Code** folder (or **Old_App_Code** if you installed the project as a web application) and name it *CustomTransformationMethods*.

2. Right click the folder and select **Add New Item**. Choose to add a new **Class** called **CMSTransformation.cs**. Please note that transformation methods must be developed in C# at this time.



3. Remove the default content of the class (apart from the basic references) and enter the following code instead:

[C#]

```
namespace CMS.Controls
{
    /// <summary>
    /// Extends the CMSTransformation partial class.
    /// </summary>
    public partial class CMSTransformation
    {

        /// <summary>
        /// Trims text values to the specified length.
        /// </summary>
        /// <param name="txtValue">Original text to be trimmed</param>
        /// <param name="leftChars">Number of characters to be returned</param>
        public string CustomTrimText(object txtValue, int leftChars)
        {
            // Checks that text is not null.
            if (txtValue == null | txtValue == DBNull.Value)
            {
                return "";
            }
            else
            {
                string txt = (string)txtValue;

                // Returns a substring if the text is longer than specified.
                if (txt.Length <= leftChars)
                {
                    return txt;
                }
                else
                {
                    return txt.Substring(0, leftChars) + "...";
                }
            }
        }

    }
}
```

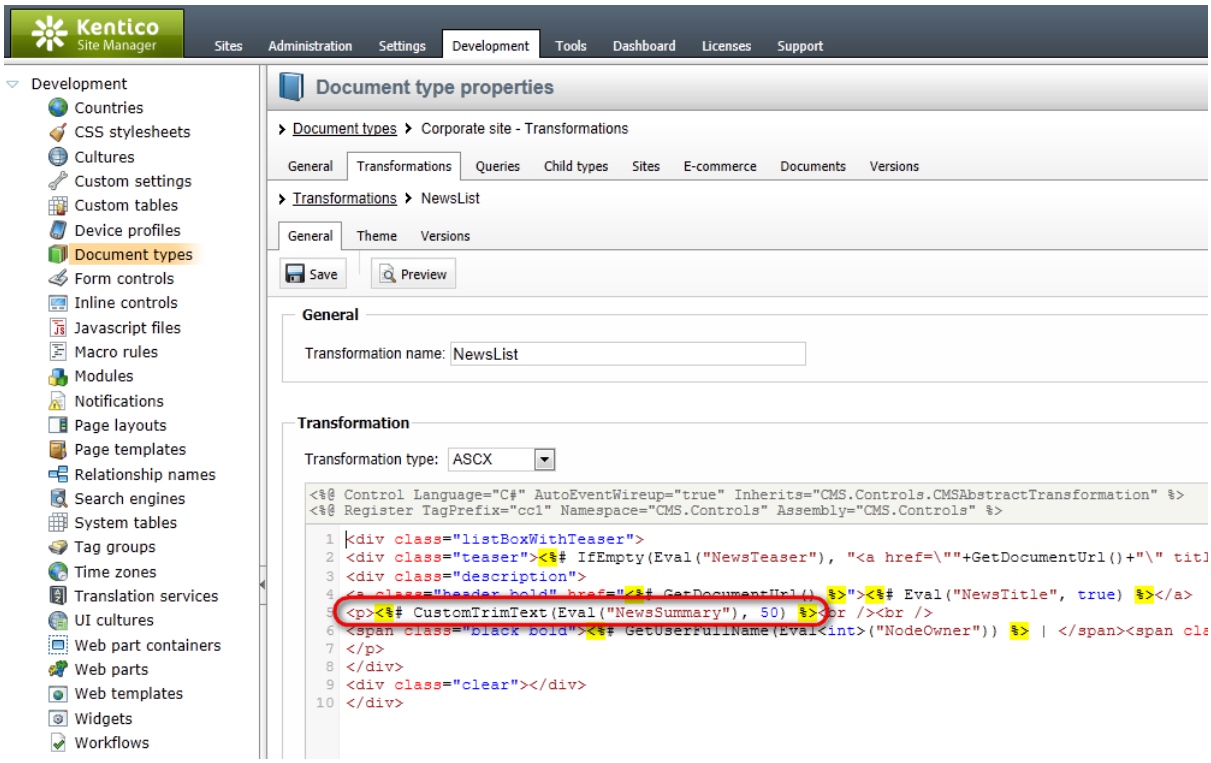
As shown above, you can extend the **CMSTransformation** partial class in the **App_Code** folder, and then add the definitions of the methods that you wish to call in your transformations.

Save the changes made to the file. **Build** the project if it is installed as a web application.

4. Now open the **Site Manager** administration interface, go to **Development -> Document types** and **Edit (**  **)** the **Corporate site - Transformations** document type. Select the **Transformations** tab, edit the **NewsList** transformation and change line 5 of its code to the following:

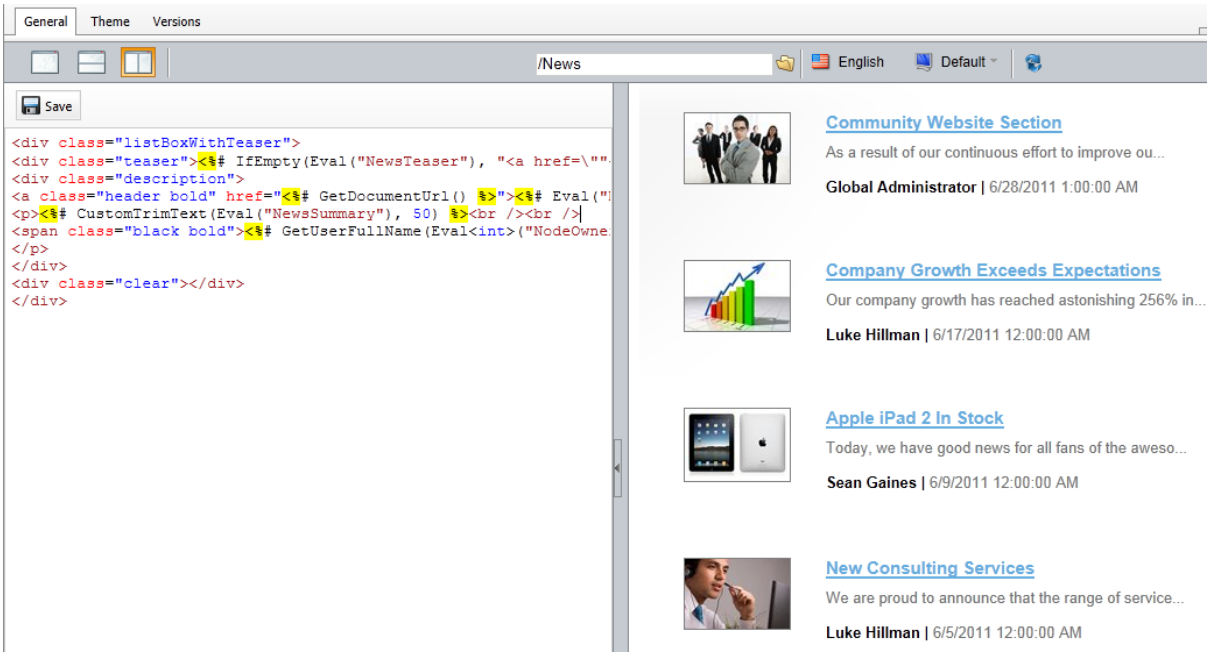
```
..

<p><%# CustomTrimText(Eval("NewsSummary"), 50) %><br /><br />
```

```
..
```



Click **Save** to confirm the modification.

5. To check the result, click the **Preview** button, enter */News* into the path textbox on the toolbar and **Refresh** () the page section. In the preview of the news page that uses the transformation, you can see the text of the summaries truncated to the first 50 characters:



In this topic, you have learned how to write your own methods for ASCX transformations. If you wish to add custom functionality to a **Text** transformation, you may implement a custom macro method for this purpose as described in Development -> Macro expressions -> Registering custom macro methods.

7.7.5.7 Transformations in macro expressions

In addition to being used by web parts and controls, transformations of the **Text/XML** or **HTML** type may also be applied to documents and other objects retrieved via Macro expressions.

This can be particularly useful if you need to display dynamically loaded data in content that is based on text and HTML, where it is not possible to add listing web parts or controls. Typical examples include the E-mail templates on which system e-mails are based, and Newsletter templates. Of course, this technique may be used in all locations where macro resolving is supported.

The functionality described above is achieved by calling the following method within your macro expression:

ApplyTransformation (String transformationName)

The parameter must match the full name of the transformation that you wish to use. An overload with three parameters is also available, which allows you to place additional transformations before and after the displayed data:

ApplyTransformation (String transformationName, String contentBeforeTransformationName, String contentAfterTransformationName)

This method can be called for collections of objects that implement the *IEnumerable* interface or for single instances of an object. When such a macro expression is resolved, it returns the objects in the given collection, formatted into output code according to the given transformation.

Security considerations

Macros are saved with a security signature, which is used to check access permissions for the data collections specified in the expression. The signature depends on the user who entered and saved the given macro expression, not the user viewing the resolved result.

This means that macro expressions will not be resolved if their author does not have permissions to access the given data. Please refer to the Macro security topic to learn more.

Examples

Start by creating a new **Page (menu item)** document on your website to prepare an environment where you can easily try out the examples below. You can choose the **Create a blank page** option when selecting the page template. On the **Design** tab, add (✚) a **Text & Images -> Editable text** web part into the zone on the page. You can now insert the macro expressions described below into the editable region on the **Page** tab of the document and they will be resolved on the live version of the page.

Please keep in mind that this scenario is intended primarily for demonstration purposes. The recommended way to display data on standard website pages is using listing web parts or controls, which provide support for transformations through the appropriate properties.

Displaying the current user

First it is necessary to create the desired transformation. Go to **Site Manager -> Development -> Document types**, edit (✎) the **Root** document type and open its **Transformations** tab. Click **New transformation** and enter the following data:

- **Transformation name**: *UsersInText*
- **Transformation type**: *Text / XML*
- **Code**:

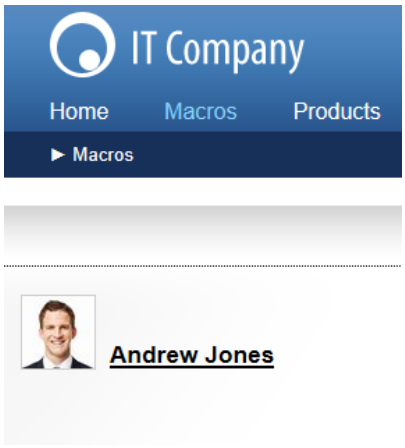
```
<div class="member">
  <a href="{% GetMemberProfileUrl(UserName) %}">
    {% GetUserAvatarImage(UserAvatarID, UserID, FullName, 52, 0, 0) %}
  </a>
<div class="memberInfo">
<p>
  <h3>
    <a href="{% GetMemberProfileUrl(UserName) %}">
      {% FullName %}
    </a>
  </h3>
</p>
</div>
</div>
```

Click **Save**.

Now open CMS Desk and go to the **Page** tab of the previously created document. Enter the following expression into the editable region and **Save** the page.

```
{%CurrentUser.ApplyTransformation("CMS.Root.UsersInText")%}
```

This expression retrieves an object containing the data of the user currently viewing the page, which is then formatted according to the specified transformation. You can view the page on the live site to see how the macro is resolved.



Displaying documents from the content tree

First, use the approach described in the previous example to create a new transformation named *NewsInText*. Remember to set the type to *Text / XML* and enter the following code:

```
<div class="description">
  <a class="header bold" href="{% GetDocumentUrl() %}">
    {% NewsTitle %})
  </a>
  <p>
    {% NewsSummary %}
    <br />
  </p>
</div>
```

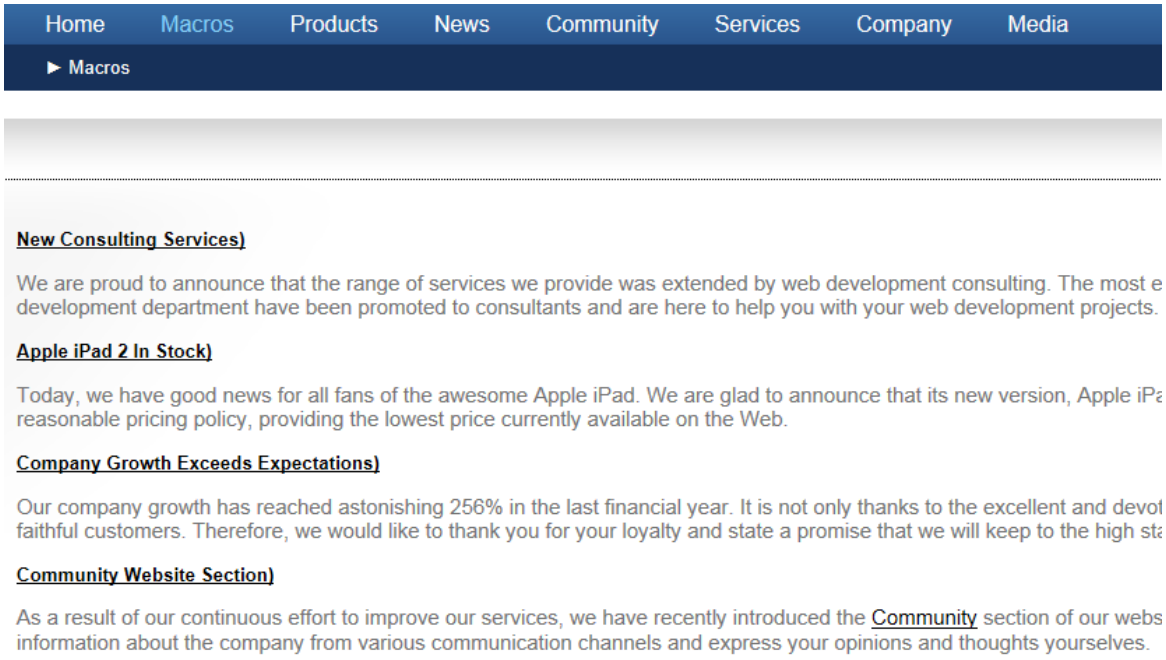
Switch to the sample document in CMS Desk and insert the following macro expression on the **Page** tab:

```
{%Documents["/News"].Children.WithAllData.ApplyTransformation
("CMS.Root.NewsInText")%}
```

In the expression above, the document under the */News* path is selected from the **Documents** collection. Through its **Children** property, we then access a collection containing all child documents. Using the **WithAllData** property ensures that the retrieved document objects include their coupled data,

i.e. the specific fields defined for the given document type.

The macro will be resolved as shown below.



Retrieving and displaying site objects

In this example, it is necessary to create three separate *Text / XML* transformations:

- **ProductTableHeader**

```
<table border="2" cellpadding="3">
  <tr>
    <td width="200"><b>Product name</b></td>
    <td width="100"><b>Price</b></td>
  </tr>
```

- **ProductTableRow**

```
<tr>
  <td>{% SKUName %}</td>
  <td>{% SKUPrice %}</td>
</tr>
```

- **ProductTableFooter**

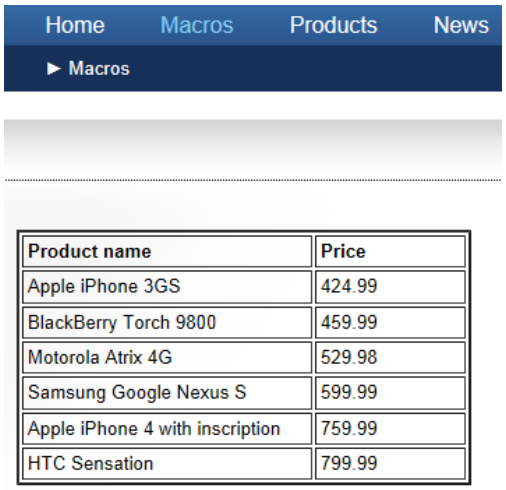
```
</table>
```

This time, copy the following macro into the editable region on the **Page** tab:

```
{%SiteObjects.SKUs.Where("SKUDepartmentID = 4").OrderBy("SKUPrice")
.ApplyTransformation ("CMS.Root.ProductTableRow", "CMS.Root.ProductTableHeader",
"CMS.Root.ProductTableFooter")%}
```

Product objects (SKUs) are retrieved from the **SiteObjects** collection. The **Where** macro method is then used to filter the collection according to a standard SQL condition specified as the parameter. In this case, only products from the Smartphones department are loaded. The **OrderBy** method sorts the objects according to the values in their **SKUPrice** field. The Where and OrderBy methods may be applied to all types of collections, including documents.

The **ApplyTransformation** method is called with additional parameters to add the header and footer transformations before and after the main data items. This ensures that the transformations are combined to achieve the desired result:



7.8 Editing code in the UI

7.8.1 Code editor overview

The interface of Kentico CMS contains areas where various types of code can be entered to define certain aspects of the website's appearance or behavior. These code fields use an editor that provides syntax highlighting support and other features to help website developers easily write and maintain sections of code.

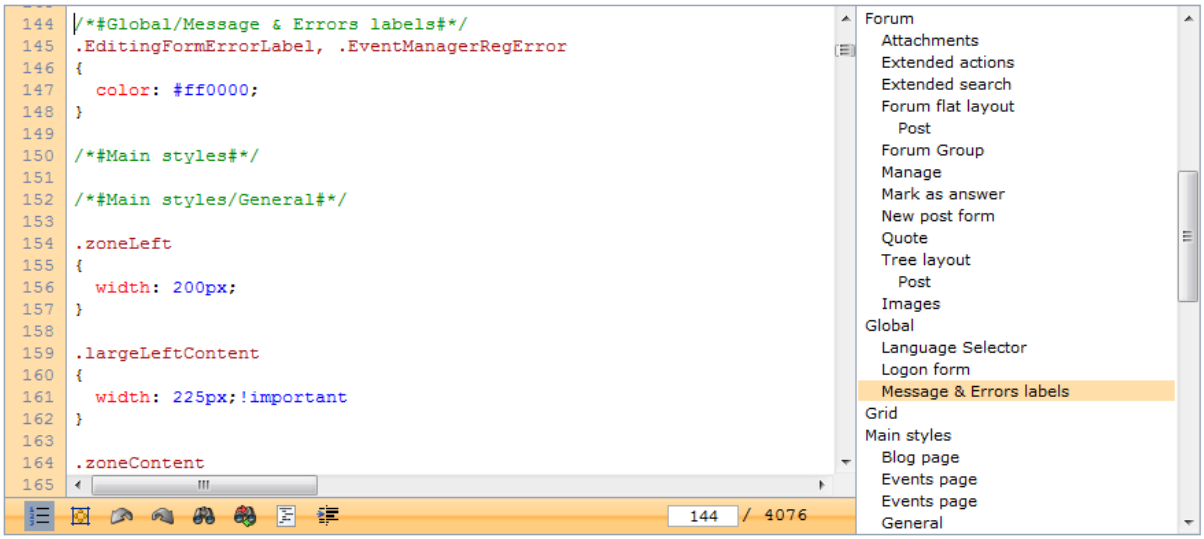
Highlighting ensures that text elements are displayed in different colors depending on the syntax of the given language. This greatly improves the readability of the code and makes it easier to spot and avoid mistakes. All languages commonly used for web development are supported, including HTML, ASPX markup, CSS, JavaScript, SQL, XML and C#. Each language has its own set of highlighting rules used to format the code.

Syntax highlighting customization

It is possible to modify the way languages are displayed in the code editor, including font properties, the colors applied to various syntax tokens and styles in general.

To do this, simply open the **CMSAdminControls/CodeMirror/css** folder under your web project and edit the stylesheet of the language that you wish to customize, e.g. **sqlcolors.css** to change the styles applied to SQL code.

In addition to syntax highlighting, the editor also provides other types of functionality.



The following buttons are located on the toolbar below the code editing area:

- **Show/hide line numbers** - enables or disables the panel displaying line numbers on the left of the editing region.
- **Toggle fit-to-window mode** - toggles the editor between its normal size and an expanded one covering the entire window (frame) in which it is placed.
- **Show hide/bookmarks** - enables or disables the bookmark panel on the side of the editor. Only available for CSS code by default.
- **Undo (CTRL+Z)** - removes the last change made to the code in the editor, restoring it to its previous state. The history of changes is saved, so this action can be used multiple times.
- **Redo (CTRL+Y)** - reverses one previously made **Undo** action every time it is used.
- **Search** - opens a dialog that allows the code in the editor to be searched for a word or phrase. You can select if the search should be case sensitive and the direction of the search.
- **Replace** - opens a dialog that can be used to find a word or phrase and replace it with the entered text. Either all occurrences in the code can be replaced at once or processed one by one.
- **Edit source** - opens a resizable dialog where the code can be edited without syntax highlighting or any other advanced functionality.
- **Indent all** - sets the indentation of all code in the editor according to the conventions of the given language.

The right side of the bottom toolbar contains a **line counter** displaying the number of the line where the cursor is currently positioned. A number can be manually entered here, which allows users to jump directly to the specified line.

The panel on the right is an alphabetical list of bookmarks that can be used to jump to marked sections in the code. Bookmarks are inserted using code blocks (regions) following the syntax of the currently edited language, e.g. */* #<*bookmark name*># */* for CSS. By default, the bookmarks panel is only displayed when editing CSS stylesheets.

Global code editor settings

The behavior of the code editor may be configured by adding the following keys into the **/configuration/appSettings** section of your project's *web.config* file:

- **CMSEnableSyntaxHighlighting** - globally enables or disables the advanced editor and syntax highlighting support for all code fields in the user interface. This can be used to turn off the editor if it is causing performance issues or other problems. The default value is *true*.
- **CMSEnableSyntaxHighlighting.<Language>** - can be used to disable the advanced editor and syntax highlighting support for fields that display code in a specific language. Replace the <*Language*> string with the name of the language that you wish to disable. The following language options are available: *Text*, *HTML*, *CSS*, *JavaScript*, *XML*, *CSharp*, *SQL*, *HTMLMixed*, *ASPNET*, *CMSSharp*. All languages are enabled by default.
- **CMSShowLineNumbers** - if set to *true*, the line number panel will be displayed in the editor by default when the page is loaded. Please note that some fields in the user interface are not affected by this setting. The default value is *false*.

Sample key values:

```
<add key="CMSEnableSyntaxHighlighting" value="false" />
<add key="CMSEnableSyntaxHighlighting.CSS" value="false" />
<add key="CMSShowLineNumbers" value="true" />
```

Customization of individual editors

Internally, the functionality of the code editor is provided by the **ExtendedTextArea** server control from the **CMS.ExtendedControls** namespace. If you wish to customize the code editor used for a specific field, you must locate the corresponding control in the code of the Kentico CMS user interface and configure its properties. This way you can switch between the advanced editor and a simple text area, define the language syntax according to which highlighting will be performed, hide the button toolbar, enable the bookmarks panel and much more. Information about the properties of the control can be found in the Kentico CMS API Reference.

The editor is also integrated into the **Large text area** (**Input** type) Form control, which can be used if you wish to create a custom field somewhere in the UI, which will be used to enter a certain type of code.

7.8.2 Design preview

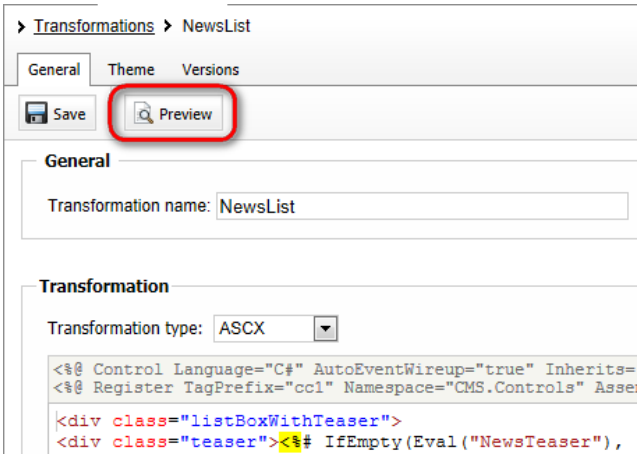
The administration interface provides a convenient way to edit the code of web design components side-by-side with a view of the results on the actual website. This allows designers to quickly check how changes affect pages without having to leave the editing interface or switch between browser tabs.

This preview functionality is available for all objects in Kentico CMS that influence the design or

appearance of the site's pages, including:

- Transformations
- CSS stylesheets
- Page layouts
- Web part containers
- Web part layouts

To access the preview mode, click the **Preview** button in the header above the given code editor.

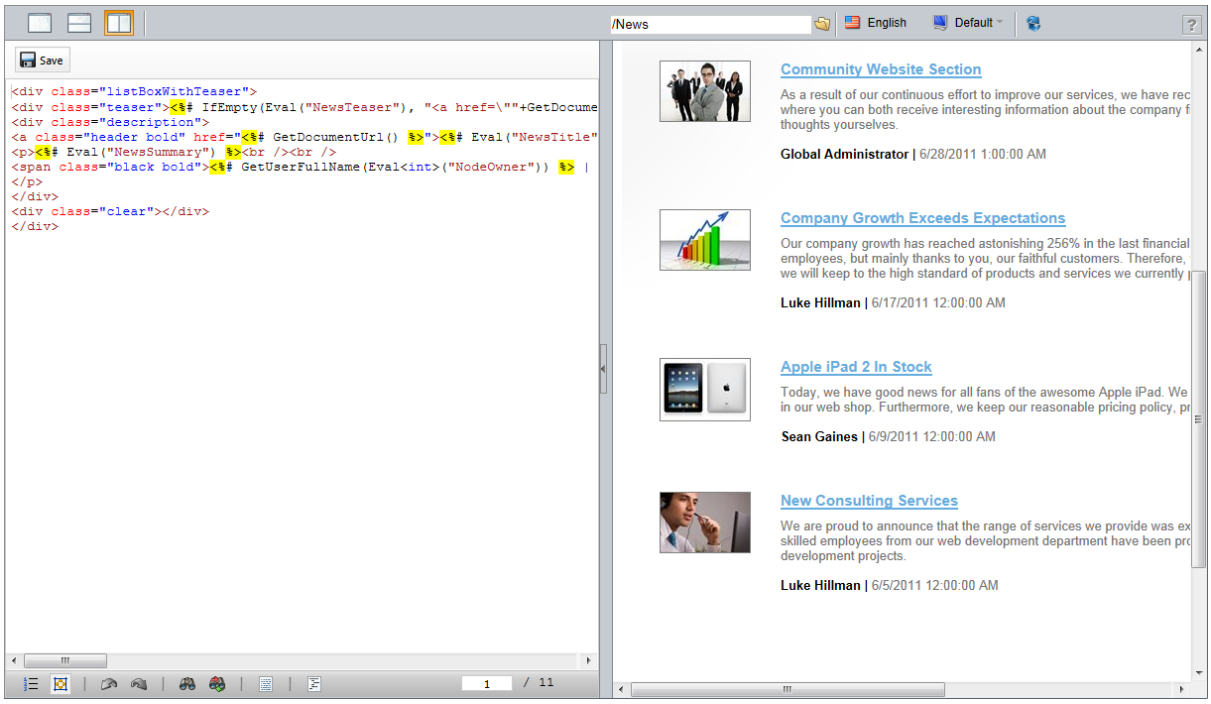


The preview interface is divided into two parts: an editing area where you can modify the code or other fields of the given object and a section displaying a live site view of a selected page.

The following options are available on the preview toolbar:

- **Close preview** - returns to the standard editing form without a preview section.
- **Horizontal layout** - splits the area horizontally. The top section contains the editing area (i.e. the code), and the preview of the page is displayed in the bottom half.
- **Vertical layout** - splits the area vertically, with the editing area on the left and the page preview on the right.

This allows you to choose the layout that best fits the dimensions of the previewed page and the edited code.



By default, the code editor automatically expands to completely cover the first section. The other fields on the form can be accessed by minimizing the editor using the **Toggle fit-to-window mode** ( ) button on the toolbar below the code. Saving ( ) the editing form automatically refreshes the page preview.

Setting the preview page

It is important to view an appropriate page where you can see the effects of the changes made in the code. For example, when editing a transformation, you need to select a page containing a web part that uses the given transformation to display data.

To configure the page in the preview section:

1. Select a document via the path selector on the toolbar. You have two options:

- Click the **Select** ( ) button to choose a document from the content tree.
- Specify a page by manually entering its alias path into the textbox. In this case, you then need to click **Refresh** ( ) to reload the page in the preview section.

When editing a design component from CMS Desk, the preview section automatically preloads the currently selected document.

2. On multilingual websites, switch to the appropriate culture version of the selected document using the language selector on the toolbar.

3. If your website is configured to display different content for specific device profiles, choose the preferred profile for the preview. When previewing content for a specific device profile, you can also click **Rotate device** to rotate the preview 90 degrees.

The section displaying the page works like the **Preview** mode of CMS Desk, which means that the

specified document does not actually need to be published on the live site. You may also preview documents that are scheduled to be published in the future or are currently in an unapproved Workflow step.

7.9 E-mail templates

7.9.1 E-mail templates

Kentico CMS sends automatic system e-mails for various purposes, such as workflow notifications. The content of these e-mails is determined by templates according to the type of a given e-mail. Many modules and features of Kentico CMS send e-mails based on predefined templates that are included in the default installation.

There are two types of e-mail templates: global and site-specific. If a site-specific template of a certain type is not defined, the respective global template is used by the site instead. All e-mail templates can be modified in **Site Manager -> Administration -> E-mail templates** and the templates of a given site may also be accessed in **CMS Desk -> Administration -> E-mail templates**. Editing templates allows you to alter the e-mails sent by the system to match the required design and/or language.

An e-mail template has the following properties:

Display name	The name of the template displayed in the user interface.
Code name	The name of the template used in code.
E-mail type	Identifies the type of functionality to which the template is related. This can be used to categorize and filter e-mail templates.
From	E-mail address that will be used as the sender (From) address of the e-mail.
Cc	E-mail addresses of copy recipients.
Bcc	E-mail addresses of blind copy recipients. These will get a copy of the e-mail, but won't see the addresses of other recipients in the mail.
Subject	Subject of the e-mail.
HTML version	Defines the content that is used for the template when sending e-mails in HTML format. The preferred format can be selected using the **Site Manager -> Settings -> System -> E-mails -> E-mail format** setting.
Plain text version	Plain text version of the e-mail template.

Example of the HTML version of a template:

```
<html>
  <head>
  </head>
  <body style="font-size: 12px; font-family: arial">
```

```
    <p>
      This is an automatic notification sent by Kentico CMS. The following
document is waiting for your approval. Please sign in to Kentico CMS Desk and
approve it.
    </p>
    <p>
      <strong>Document:</strong> <a href="{%DocumentEditUrl%}">{%documentname%}</
a> {% ifEmpty(DocumentPreviewUrl, "", "(<a href=\"" + DocumentPreviewUrl="" +
"\">preview</a>)")|(encode)false %}
      <br />
      <strong>Last approved by:</strong> {%approvedby%}
      <br />
      <strong>Last approved when:</strong> {%approvedwhen%}
      <br />
      <strong>Original step:</strong> {%originalstepname%}
      <br />
      <strong>Current step:</strong> {%currentstepname%}
      <br />
      <strong>Comment:</strong>
      <br />
      {%comment%}
    </p>
  </body>
</html>
```

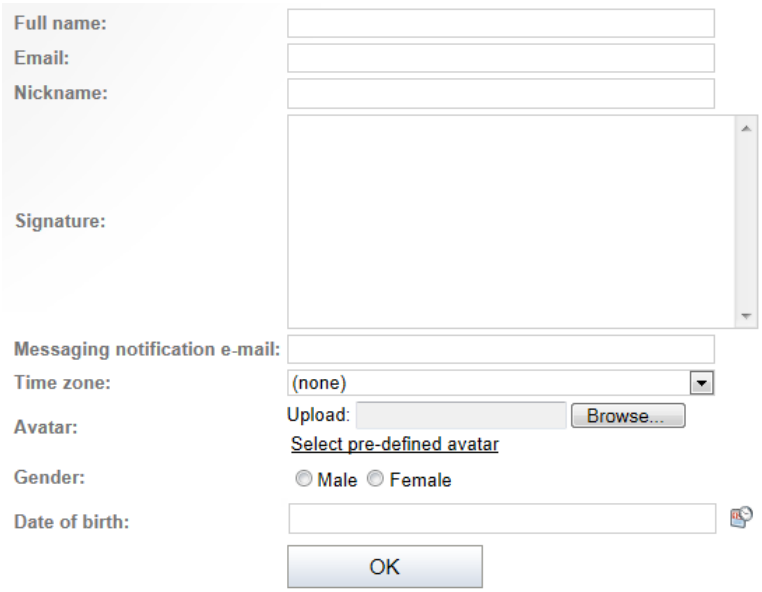
Please note that e-mail template properties may contain macro expressions, such as those marked in the code above, that are resolved (merged) when the e-mail is sent. The use of macros is necessary to ensure that individual e-mails contain information relevant to the situation that caused them to be sent. In some cases, you may wish to display the data loaded via a macro in a specific format. This can be done by applying transformations as described in the Transformations in macro expressions topic.

It is possible to attach files to an e-mail template through the **Attachments** button in the header of the template editing page. When e-mails based on the given template are sent out, the attachments will be included. Clicking the button opens a dialog where the attachments can be managed.

7.10 Form controls

7.10.1 Overview

Form controls provide the interface for various editing forms that allow users to input data into Kentico CMS, both in the administration interface and on the live site. Each control represents a single field by displaying a certain form element, like a text box or area for user input, a group of radio buttons, an object selector etc.



All form controls are registered as objects in the system and may be configured in various ways. More information can be found in the Managing form controls topic.

The controls can be placed into all editing forms that are based on the Kentico CMS form engine, which includes the following:

- Document type editing forms (Form tab)
- Web part and Widget properties
- Editing forms of System table objects
- Editing forms of Custom tables
- On-line forms
- Alternative forms for the objects mentioned above
- Custom website settings
- Report parameters
- Macro rule parameters

These forms are defined through a special field editor interface as described in the Assigning form controls to fields topic.

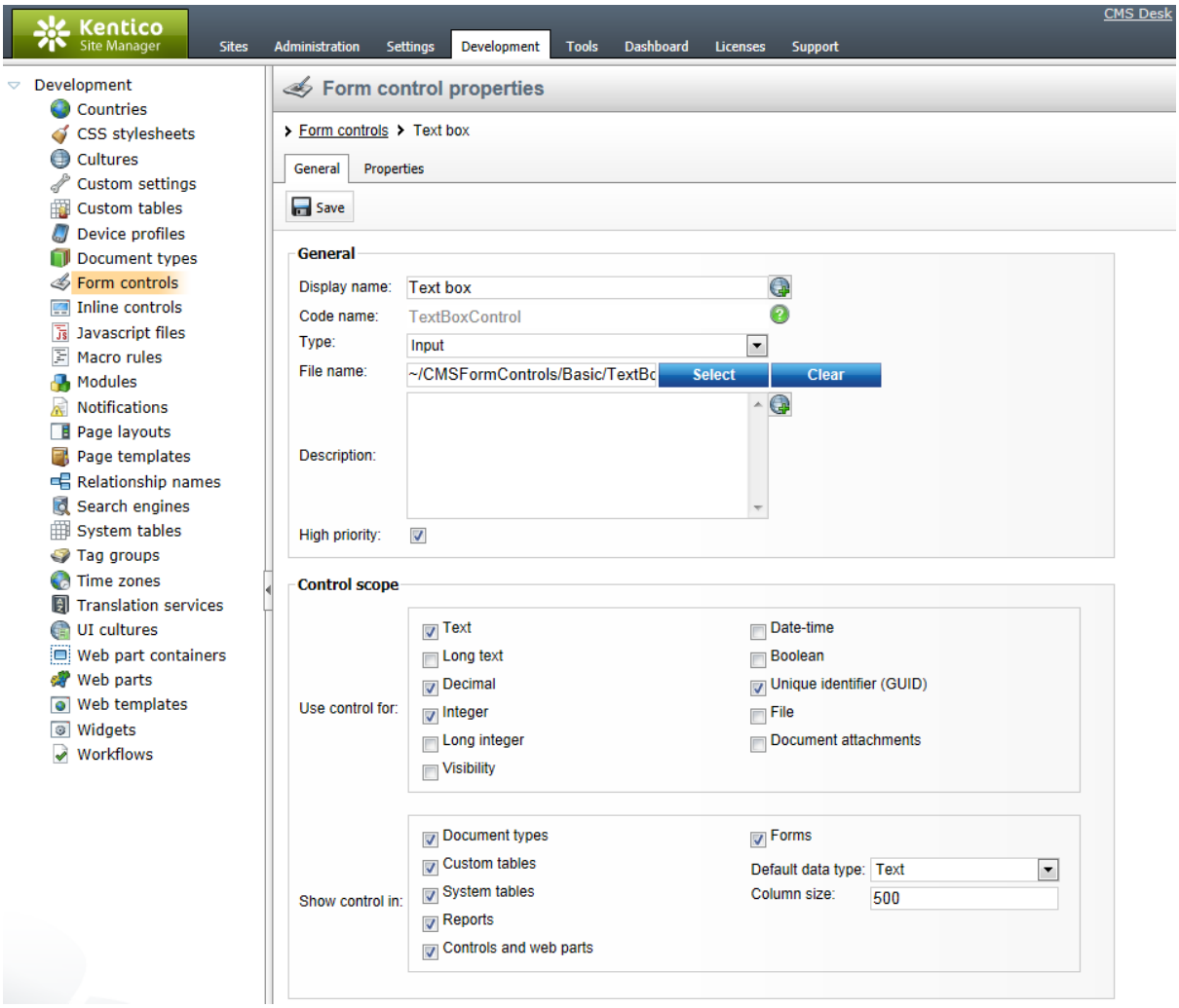
From a website development point of view, form controls are implemented as standard user controls (*.ascx* files) that inherit from the **CMS.FormControls.FormEngineUserControl** class.

Even though the built-in set of controls covers most standard types of form functionality, it is also possible to customize existing controls or create completely new ones to match any specific requirements. Please refer to the Developing form controls topic for additional information and an example of how to create a custom form control, register it in the system and insert it into a form.

7.10.2 Managing form controls

You can view the catalog of available form controls in **Site Manager -> Development -> Form controls** and register new ones as necessary. When edited, form controls have the following properties available on the **General** tab:

- **Display name** - sets the name of the form control displayed in the administration interface, e.g. when selecting a control in the field editor.
- **Code name** - sets the name of the control that is used as its unique identifier, for example in the API.
- **Type** - allows the selection of a category under which the form control will be placed. The type can be used to filter form controls.
- **File name** - contains the relative path to the *.ascx* file that implements the form control, for example: *~/CMSFormControls/Basic/TextBoxControl.ascx*
- **Description** - may be used to enter a text description of the control or comments for other administrators.
- **High priority** - when selecting a form control in the field editor, controls with high priority are offered at the top of the list and highlighted using a colored background. This option should only be enabled for those controls that are most commonly used.



The properties under the **Use control for** section determine for which attribute types the form control should be available. Possible attribute types for fields include standard data types (Text, Long text, Decimal, Integer, Long integer, Boolean, Date-time), as well as the following special options:

- **Visibility** - this attribute type is used for controls that modify the visibility options of other fields. Please see the Development -> Membership, permissions and security -> Custom field visibility

chapter for more details.
- **Unique identifier (GUID)** - attributes of this type are 32-character strings used as globally unique identifiers for objects.
- **File** - used for fields that provide file management.
- **Document attachments** - this attribute type is used for fields that allow the management of document attachments.

The **Show control in** section is used to specify in which types of forms the control can be used i.e. where in the user interface it can be selected. The following options are available:

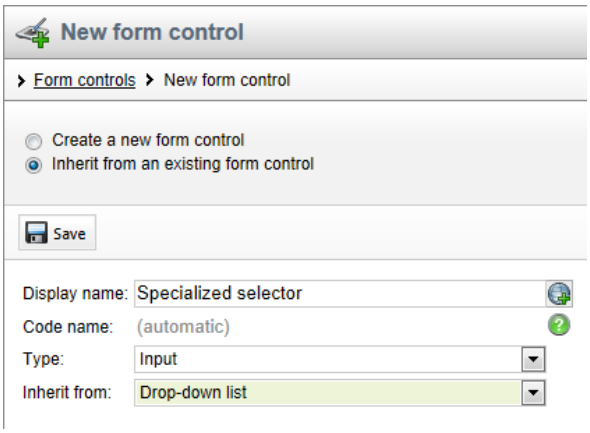
- **Document types** - allows use in document type fields, which can be edited in *Site Manager -> Development -> Document types*. The actual form is displayed when editing a document on the *Form* tab in *CMS Desk -> Content -> Edit*.
- **Custom tables** - allows the control in custom table fields, which can be edited in *Site Manager -> Development -> Custom tables*. The corresponding data editing form can be found in *CMS Desk -> Tools -> Custom tables*.
- **System tables** - allows use in system table fields, which can be edited in *Site Manager -> Development -> System tables*. The editing forms of system objects are located in various sections of the administration interface or on the live site, depending on the specific object.
- **Reports** - if checked, the control may be used to provide the interface for entering report parameters, which can be defined when editing a report in *CMS Desk -> Tools -> Reporting*. The reporting parameter form is displayed when a report is viewed in the interface or published on the live site.
- **Controls and web parts** - allows the control to be used for the properties of web parts and widgets, as defined in *Site Manager -> Development -> Web parts/Widgets*. This editing form provides the configuration dialog of individual web parts or widget instances.
- **Forms** - if checked, the control may be used to represent the fields of on-line forms, which can be edited in *CMS Desk -> Tools -> Forms*. The *High priority* flag of the form control also needs to be enabled if you wish to have it offered in the form field editor's *simple* mode. If you allow a form control for forms, you also need to enter the following default values that will be used if the field is edited in simple mode:
  - **Default data type** - the data type of the field that will be used by default when a user chooses to create a new field of this type.
  - **Column size** - only applies to the *Text* data type. This sets the maximum size of the database column used to store the field, which also limits the maximum number of characters that may be entered into the field.

The **Properties** tab is used to define parameters for the specific form control. Further information can be found in the Form control parameters topic.

Form control inheritance

When creating a new form control, it is possible to inherit from an existing control rather than starting from scratch. Inherited controls work the same way as their parent, but it is possible to change their general settings (i.e. set where they can be used) and assign different default values for their parameters. This allows you to create specialized controls based on general ones. Such controls can be particularly useful for users who work with the field editor in simple mode.

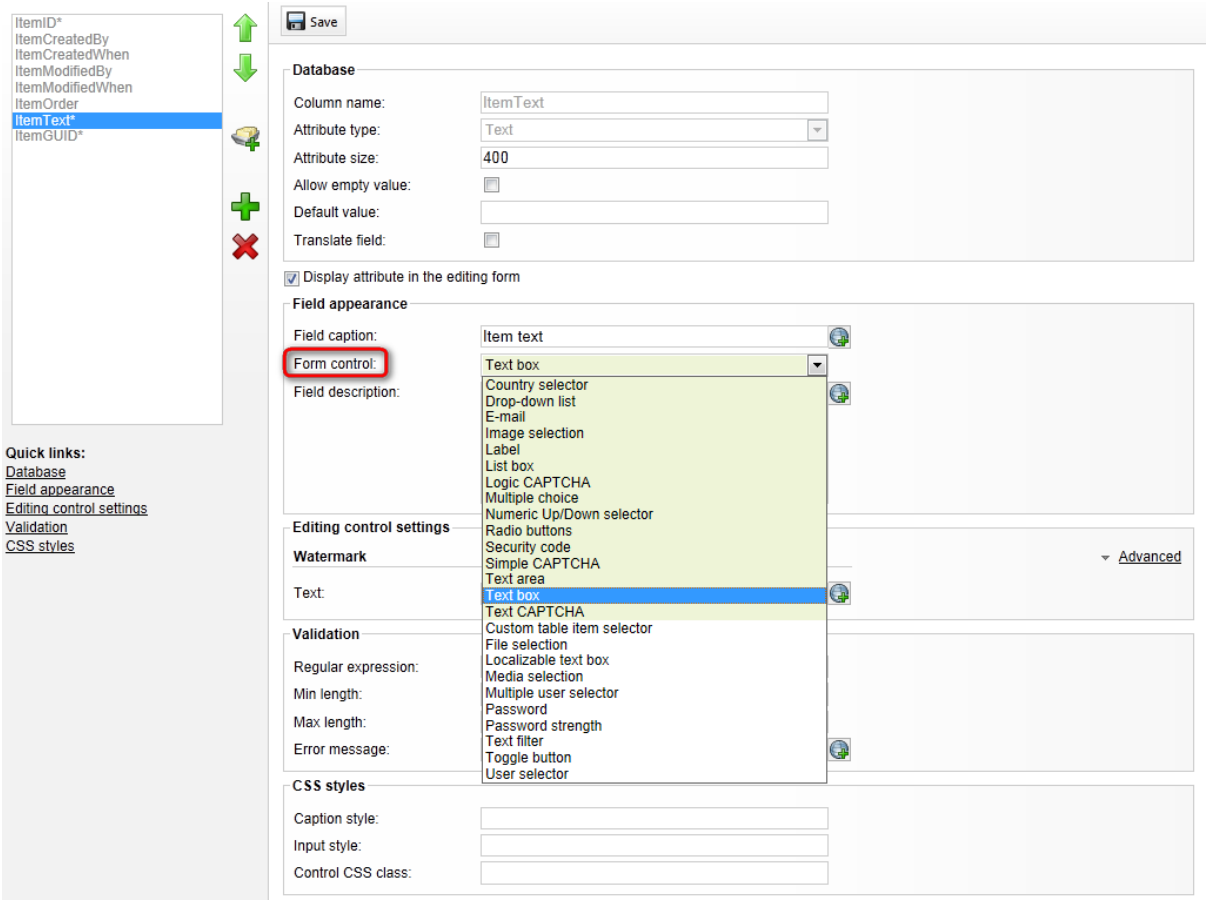
For example, imagine that you need to define a reusable drop-down list with a specific set of options. This is a typical scenario that can easily be achieved by creating an inherited form control. You could use the general *Drop-down list* form control as the parent and then simply set the required options through the default values of the parameters available on the **Properties** tab.



Inherited form controls use the same source file as their parent, so they do not have a **File name** property. Instead, you simply select the parent when creating the inherited control.

7.10.3 Assigning form controls to fields

To allow users to interact with the fields in an editing form, it is necessary to select an appropriate form control for each displayed field. This can be done through the field editor of the given form, specifically using the **Form control** property in the **Field appearance** section.



As you can see, the high priority form controls are highlighted and offered first in the selection list. If there are too many form control options and you cannot find the one that you need directly in the drop-down, you can assign it via the full selection dialog opened by clicking the (*more items ...*) option.

Please keep in mind that the selection options are limited by the scope settings of individual form controls. This means that different controls will be offered based on the type of form that is being edited and the **Attribute type** selected for the current field.

Default form control selection

You can use the settings in **Site Manager -> Settings -> System -> Form engine** to choose which form controls should be offered as the default option when adding new fields in the field editor. A separate setting is available for each data type (Attribute type) that can be assigned to a field.

These settings are global and affect all sites in the system.

In addition to the choice of a form control, the field editor provides many configuration options for individual fields as described in the table below. Please note that not all settings may be available depending on the selected **Attribute type** and the field editor may also have other options related to the specific object type for which the form is being defined.

Database	
Column name	Sets the name used for the database column storing the values of the field. It also serves as an identifier for the field.
Attribute type	Determines the data type of the value that the field will contain.
Attribute size	Sets the maximum amount of characters that can be entered into the field. Only available if the **Attribute type** is set to *Text*.
Allow empty value	If enabled, the field will allow empty values. If disabled, it will not be possible to save the form unless there is a value entered in the given field.
Default value	The default value of the field that will be pre-filled when the form is loaded.
Translate field	Indicates if the field should be included when using translation services with the objects for which the form is defined. Only available if the selected **Attribute type** is *Text* or *Long text*.
Display attribute in the editing form	Indicates if the given field should be shown in the form to users. This can be disabled for fields that store internal or system values.
Field appearance	

Field caption	Sets the text displayed in the editing form next to the field.
Form control	As described in the text above, this field selects the form control that will be used to interact with the given field in the editing form.
Field description	Tooltip which will be displayed if a user hovers over the field.
Editing control settings	
The settings in this section provide a way to configure the parameters defined for the selected **Form control**. Please see the Form control parameters topic to learn more.	
Validation	
Regular expression	This regular expression will be used by the field's validator to determine which values are acceptable. Only available if the selected **Attribute type** is *Text* or *Long text*.
Min/Max length	Sets the minimum/maximum length for entered values. Only available if the selected **Attribute type** is *Text* or *Long text*.
Min/Max value	Sets the minimum/maximum value that can be entered. Only available if the selected **Attribute type** is numerical.
From/To	Can be used to specify a time interval into which the entered value must belong. Only available if the selected **Attribute type** is *Date and time*.
Error message	Error message displayed if a user enters invalid input into the field and attempts to save the editing form.
CSS styles	
Caption style	Used to set CSS styles for the caption of the given field.
Input style	Used to set CSS styles for the input entered into the field.
Control CSS class	Name of the CSS class used to style the field.
Field advanced settings	
Visible condition	May be used to enter a macro condition that must be fulfilled in order for the specific field to be visible in the editing form. The condition is dynamically resolved when the form is loaded. You can write any condition according to your specific requirements. For details about available macro options and syntax, please refer to the Development -> Macro expressions chapter of the Developer's Guide. If you wish to create a condition that depends on the state of the other fields in the form, you can access them in the macro expression using the corresponding **Column name**. The data of the fields may then be retrieved through the following properties:

	• **Value** - returns the current value of the field. For example, *FirstName.Value* is resolved into the value entered into the FirstName field. • **Visible** - returns a true value if the given field is currently visible in the form. • **Enabled** - true if the field is currently enabled, i.e. its value can be edited. • **Info.<field setting>** - may be used to access various settings configured for the field, for example: *FirstName.Info.AllowEmpty*
Enabled condition	Allows you to enter a macro condition that determines when the field should be enabled. If this condition is resolved as false, the field will be visible, but it will not be possible to edit its value. You can use the same macro options as described above for the **Visible condition** property.
Has depending fields	If enabled, the editing form will be refreshed via autopostback whenever the value of the given field is changed. This means that other fields can be dynamically updated according to the current value of the field. Please note that the actual logic of the dependencies needs to be implemented in the code of the used form controls.
Depends on another field	This option must be enabled if you wish to have the behavior of the field dynamically changed according to the value of some other field.

Forms may also contain categories, which allow you to group multiple fields together. Using categories is recommended in large forms to make orientation easier for users. Each category will include those fields that are positioned below it in the field editor. Categories can be created by clicking the **New category** () button.

The following settings are available when creating or editing a category:

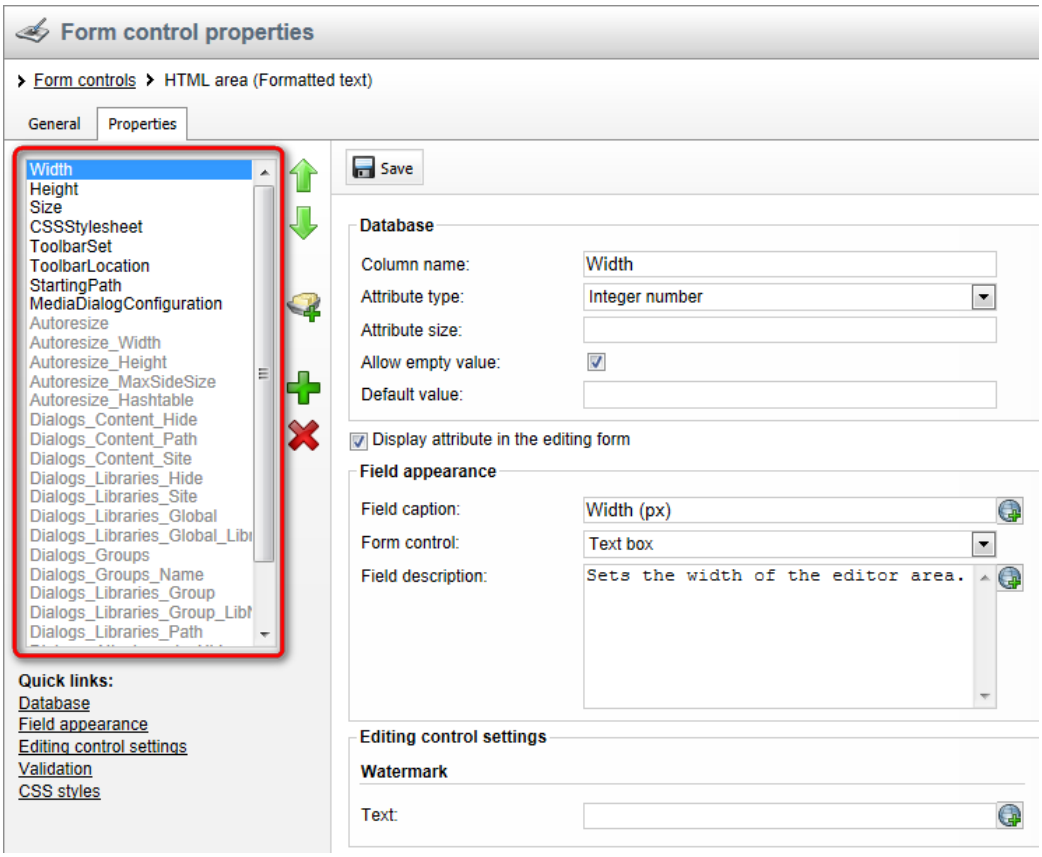
Category	
Category name	Sets the name of the category that will be displayed in the editing form.
Collapsible	If enabled, users will be able to collapse and expand the content of the category.
Collapsed by default	If enabled, the category will initially be collapsed when the form is loaded.
Visible	If disabled, the category and all of the fields it contains will be hidden in the form.
Visible condition	May be used to enter a macro condition that must be fulfilled in order for the category to be visible. You can use the same

	macro options as described above for the **Visible condition** property of fields.

7.10.4 Form control parameters

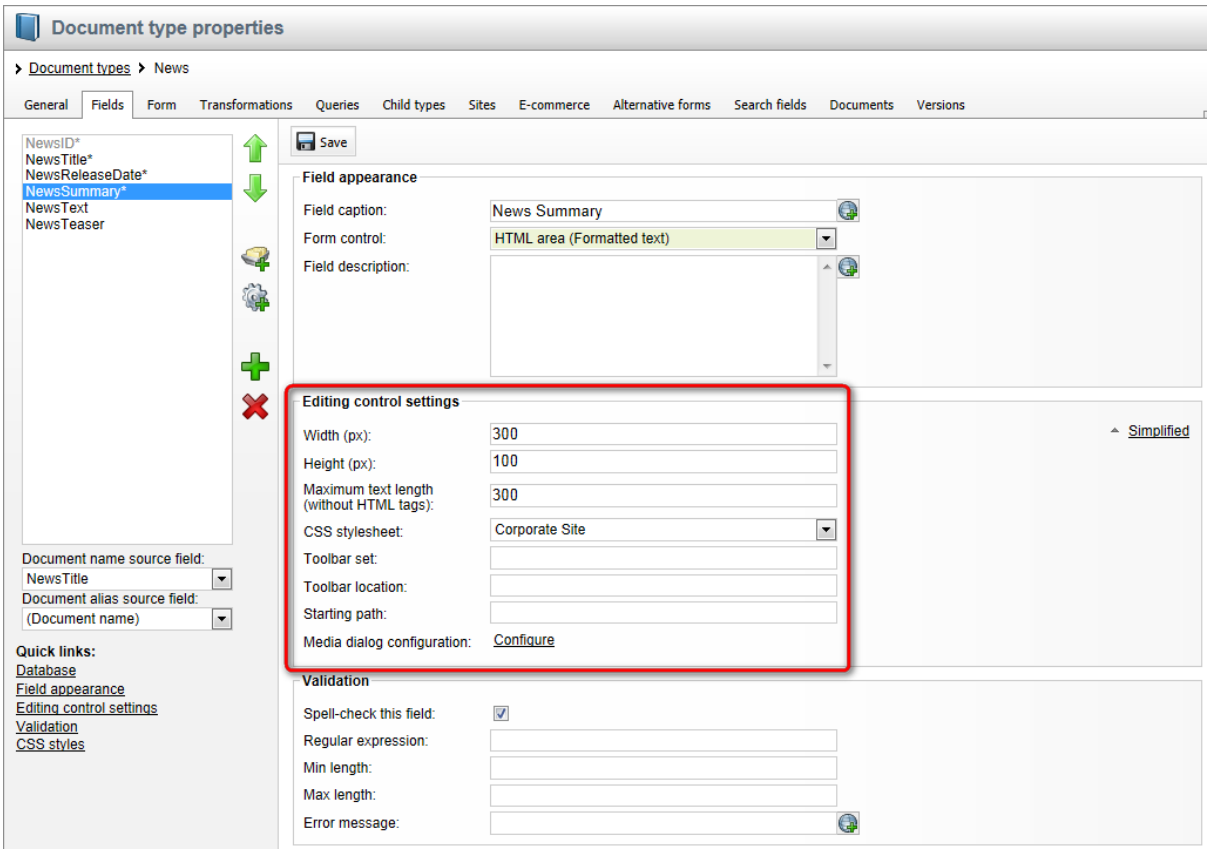
It is possible to define parameters for form controls that can be used to further specify the behavior or appearance of the generated form fields. Parameters can greatly increase the flexibility of a form control, since they provide additional configuration options for individual fields that use the control. This allows you to reuse the same control without having to develop a new one for every small difference in functionality or design.

To access the parameters of a form control, edit () the given control in **Site Manager -> Development -> Form controls** and switch to the **Properties** tab. Here, you can manage the fields that represent individual parameters.



As you may have noticed, the parameters themselves are created through a field editor and the form used to edit their values is based on the standard form engine. The editing interface for every parameter is provided by a selected form control, which means you can create completely custom parameters according to your own requirements.

The defined parameters are then displayed when the given form control is selected in the field editor of an object. You can view and configure them under the **Editing control settings** section of the given field.



There are two possible modes for this group of settings: *Advanced* and *Simplified*. You can switch between them using the corresponding link in the given section of the field editor. Simplified mode only offers the parameters that have the **Display in simple mode** option enabled (on the **Properties** tab of the given form control under the **Field advanced settings** section).

The parameter values entered here will be applied to the given field.

Warning!

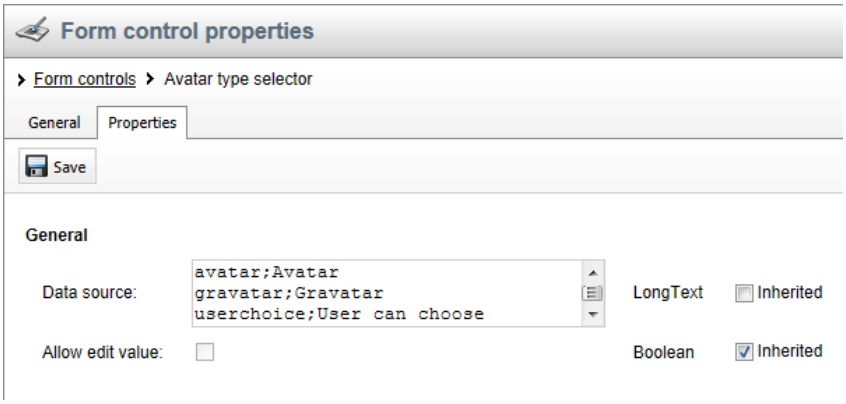
Removing or modifying the parameters of the built-in form controls may cause the system to behave incorrectly in some cases, since the controls are used in various parts of the Kentico CMS interface.

We recommend implementing a new form control or inheriting from an existing example if you wish to create a custom field.

Parameters of inherited form controls

When editing an inherited form control on the **Properties** tab, the parameters are automatically loaded from the parent control. The configuration of the parameters cannot be modified, it is only possible to change their default values. This allows you to create specialized form controls that may easily be used in the field editor without any configuration of their **Editing control settings**.

To modify the default value of a parameter, uncheck the **Inherited** box next to it and then make the necessary changes. Otherwise, the default value set for the given property in the parent control will be used.



An inherited form control may only override the default value for parameters that have the **Display attribute in the editing form** option enabled in the definition of the parent control.

Handling parameters in the code of custom form controls

In order to be reflected by the field in the resulting form, each parameter must be processed in the code of the form control. The following methods can be used to access the value of a parameter:

- *GetValue("<ParameterName>")*
- *SetValue("<ParameterName>", Object value)*

Replace the <*ParameterName*> expression with the **Column name** of the parameter that you wish to get or set. These two methods are inherited from the **FormEngineUserControl** class, so they are available for every form control. For easy storage and usability, it is recommended to define a separate property for every parameter (as a member of the control's class). Once you have the value of the parameter, you can implement code that modifies the control accordingly.

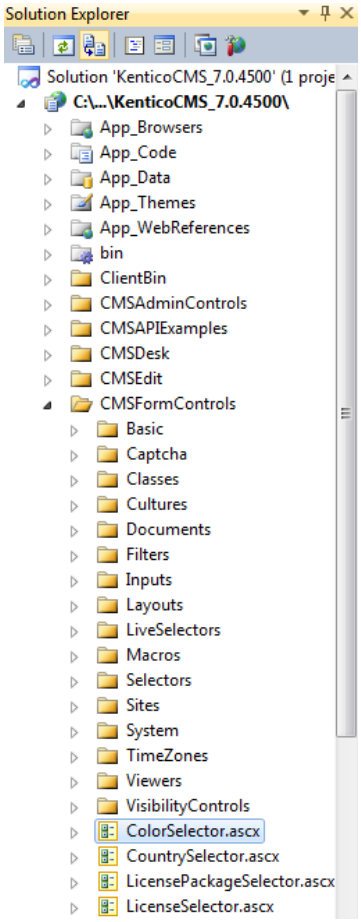
For more details, please see the example in the Developing form controls topic, which demonstrates how a parameter can be added to a form control and handled in the code.

7.10.5 Developing form controls

The following example shows how to create a form control that will allow users to choose a color from a drop-down list. The example is relatively simple, but the same basic approach can be used to create any type of custom form control or modify an existing one.

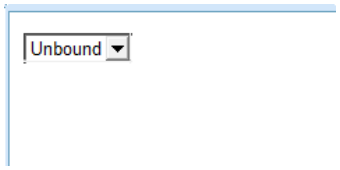
1. Open your web project in Visual Studio (or Visual Web Developer) using the **WebProject.sln** file or via **File -> Open -> Web site** in Visual Studio.

2. Right-click the **CMSFormControls** folder and choose **Add New Item**. Choose to create a new **Web User Control** and call it *ColorSelector.ascx*.



This folder (or a sub-folder under it) should always be used to store the source files of custom form controls, since it ensures that registered form controls are exported correctly along with the site.

3. Edit the new control on the **Design** tab. Drag and drop a standard **DropDownList** control onto the form from the Toolbox:



4. Edit the properties of the DropDownList and change its **ID** to *drpColor*.

5. Switch to the code behind and add a reference to the following namespace:

[C#]

```
using CMS.FormControls;
using CMS.GlobalHelper;
```

[VB.NET]

```
Imports CMS.FormControls
Imports CMS.GlobalHelper
```

6. Next, modify the class definition according to the following:

[C#]

```
public partial class CMSFormControls_ColorSelector : System.Web.UI.UserControl
```

to

```
public partial class CMSFormControls_ColorSelector : FormEngineUserControl
```

[VB.NET]

```
Partial Class CMSFormControls_ColorSelector
    Inherits System.Web.UI.UserControl
```

to

```
Partial Class CMSFormControls_ColorSelector
    Inherits FormEngineUserControl
```

This ensures that our form control inherits from the **CMS.FormControls.FormEngineUserControl** class and can use its standard properties.

7. Now add the following members into the class:

[C#]

```
/// <summary>
/// Gets or sets the value entered into the field, a hexadecimal color code in
this case.
/// </summary>
public override Object Value
{
    get
    {
        return drpColor.SelectedValue;
    }
    set
    {
        // Ensure drop down list options
        EnsureItems();
        drpColor.SelectedValue = System.Convert.ToString(value);
    }
}
```

```
/// <summary>
/// Property used to access the Width parameter of the form control.
/// </summary>
public int SelectorWidth
{
    get
    {
        return ValidationHelper.GetInteger(GetValue("SelectorWidth"), 0);
    }
    set
    {
        SetValue("SelectorWidth", value);
    }
}

/// <summary>
/// Returns an array of values of any other fields returned by the control.
/// </summary>
/// <returns>It returns an array where the first dimension is the attribute name
and the second is its value.</returns>
public override object[,] GetOtherValues()
{
    object[,] array = new object[1, 2];
    array[0, 0] = "ProductColor";
    array[0, 1] = drpColor.SelectedItem.Text;
    return array;
}

/// <summary>
/// Returns true if a color is selected. Otherwise, it returns false and displays
an error message.
/// </summary>
public override bool IsValid()
{
    if ((string)Value != "")
    {
        return true;
    }
    else
    {
        // Set form control validation error message.
        this.ValidationError = "Please choose a color.";
        return false;
    }
}


/// <summary>
/// Sets up the internal DropDownList control.
/// </summary>
protected void EnsureItems()
{
    // Applies the width specified through the parameter of the form control if it
is valid.
    if (SelectorWidth > 0)
    {
        drpColor.Width = SelectorWidth;
    }
```

```
    if (drpColor.Items.Count == 0)
    {
        drpColor.Items.Add(new ListItem("(select color)", ""));
        drpColor.Items.Add(new ListItem("Red", "#FF0000"));
        drpColor.Items.Add(new ListItem("Green", "#00FF00"));
        drpColor.Items.Add(new ListItem("Blue", "#0000FF"));
    }
}

/// <summary>
/// Handler for the Load event of the control.
/// </summary>
protected void Page_Load(object sender, EventArgs e)
{
    // Ensure drop-down list options
    EnsureItems();
}
```

[VB.NET]

```
''' <summary>
''' Gets or sets the value entered into the field, a hexadecimal color code in
this case.
''' </summary>
Public Overrides Property Value() As Object
    Get
        Return drpColor.SelectedValue
    End Get
    Set(ByVal value As Object)
        EnsureItems()
        drpColor.SelectedValue = System.Convert.ToString(value)
    End Set
End Property

''' <summary>
''' Property used to access the Width parameter of the form control.
''' </summary>
Public Property SelectorWidth() As Integer
    Get
        Return ValidationHelper.GetInteger(GetValue("SelectorWidth"), 0)
    End Get
    Set(ByVal value As Integer)
        SetValue("SelectorWidth", value)
    End Set
End Property

''' <summary>
''' Sets values for any other fields (attributes) of the object in which the form
control is used.
''' </summary>
''' <returns>Returns an array where the first dimension is the field's column name
and the second is its value.</returns>
Public Overrides Function GetOtherValues() As Object(,)
    Dim arr(0, 1) As Object
    arr(0, 0) = "ProductColor"
    arr(0, 1) = drpColor.SelectedItem.Text
```

```
    Return arr
End Function

''' <summary>
''' Returns true if a color is selected. Otherwise, it returns false and displays
an error message.
''' </summary>
Public Overrides Function IsValid() As Boolean
    If CType(Value, String) <> "" Then
        Return True
    Else
        ' Set form control validation error message.
        Me.ValidationError = "Please choose a color."
        Return False
    End If
End Function


''' <summary>
''' Sets up the internal DropDownList control.
''' </summary>
Public Sub EnsureItems()
    ' Applies the width specified through the parameter of the form control if it
is valid.
    If SelectorWidth > 0 Then
        drpColor.Width = SelectorWidth
    End If

    If drpColor.Items.Count = 0 Then
        drpColor.Items.Add(New ListItem("(select color)", ""))
        drpColor.Items.Add(New ListItem("Red", "#FF0000"))
        drpColor.Items.Add(New ListItem("Green", "#00FF00"))
        drpColor.Items.Add(New ListItem("Blue", "#0000FF"))
    End If
End Sub

''' <summary>
''' Handler for the Load event of the control.
''' </summary>
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Me.Load
    ' Ensure drop down list options
    EnsureItems()
End Sub
```

The above code overrides three members inherited from the **FormEngineUserControl** class that are most commonly used when developing form controls:

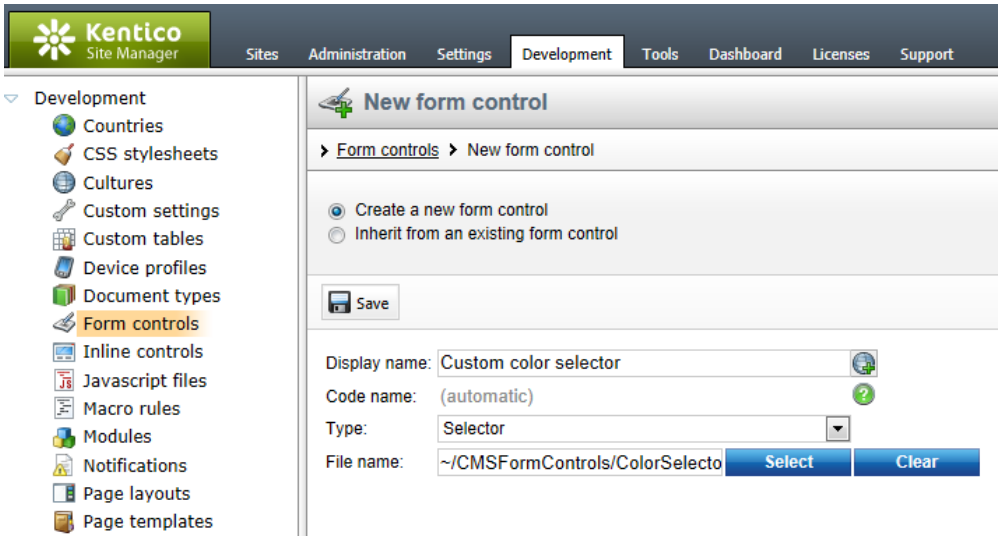
- **Value** - it is necessary to override this property for every form control. It is used to get and set the value of the field provided by the control.
- **GetOtherValues()** - this method is used to set values for other fields of the object in which the form control is used. It must return a two dimensional array containing the names of the fields (columns) and their assigned values. Typically used for multi-field form controls that need to store data in multiple database columns, but only occupy a single field in the form.
- **IsValid()** - this method is used to implement validation for the values entered into the field. It must return *true* or *false* depending on the result of the validation.

Also notice that a **SelectorWidth** property was defined for the form control. It serves as a way to access the value of a parameter that will be defined for the form control later in the example. This property is used in the **EnsureItems()** method to set the width of the internal drop-down list .

Remember to save the changes to both code files and **Build** your project if it is installed as a web application.

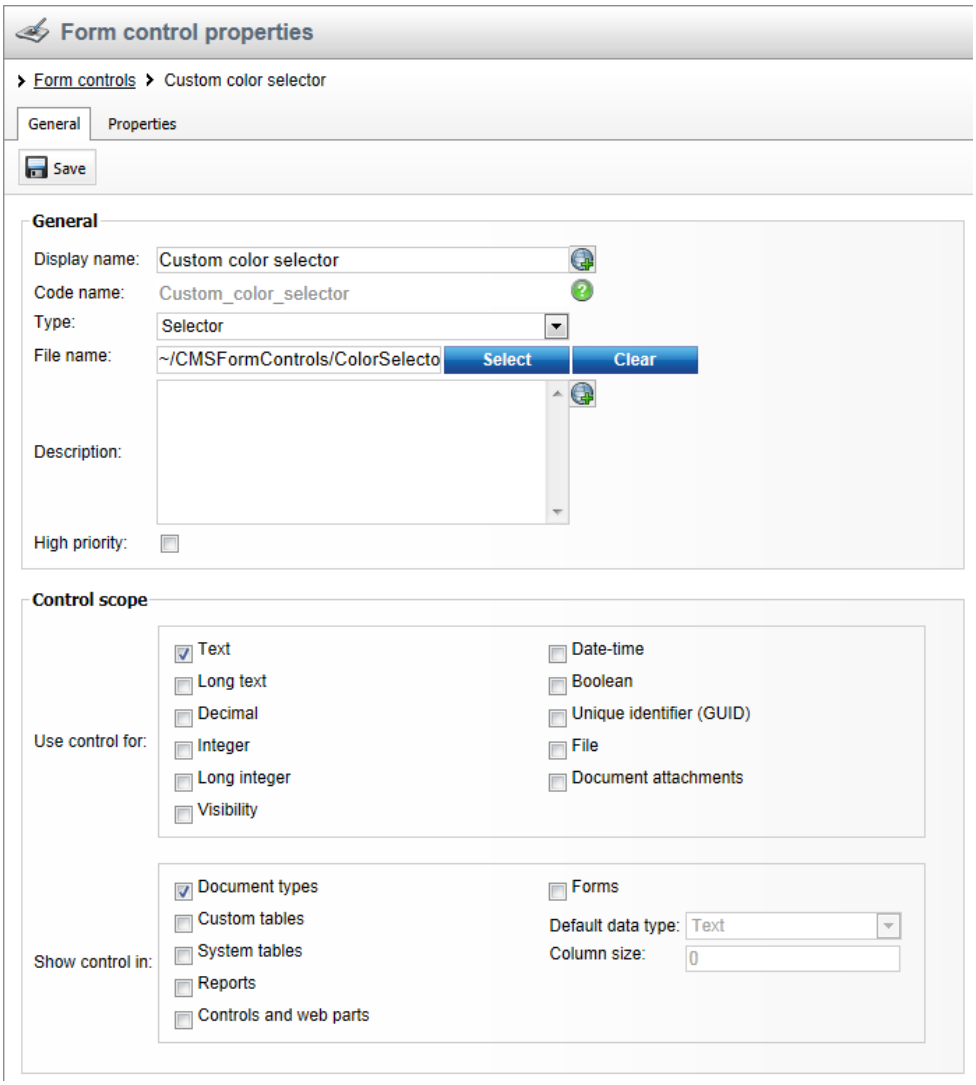
8. Go to **Site Manager -> Development -> Form controls** where you can register the new form control in the system. Click the **New form control** button, select the **Create a new form control** option and enter the following values:

- **Display name**: Custom color selector
- **Code name**: Leave the *(automatic)* option
- **Type**: Selector
- **File name**: ~/CMSFormControls/ColorSelector.ascx (you can use the **Select** button to choose the file)



Click **Save** and the form control will be created.

9. You will be redirected to the control's **General** tab. Here, check the **Use control for** - **Text** and **Show control in** - **Document types** boxes and click **Save** again.



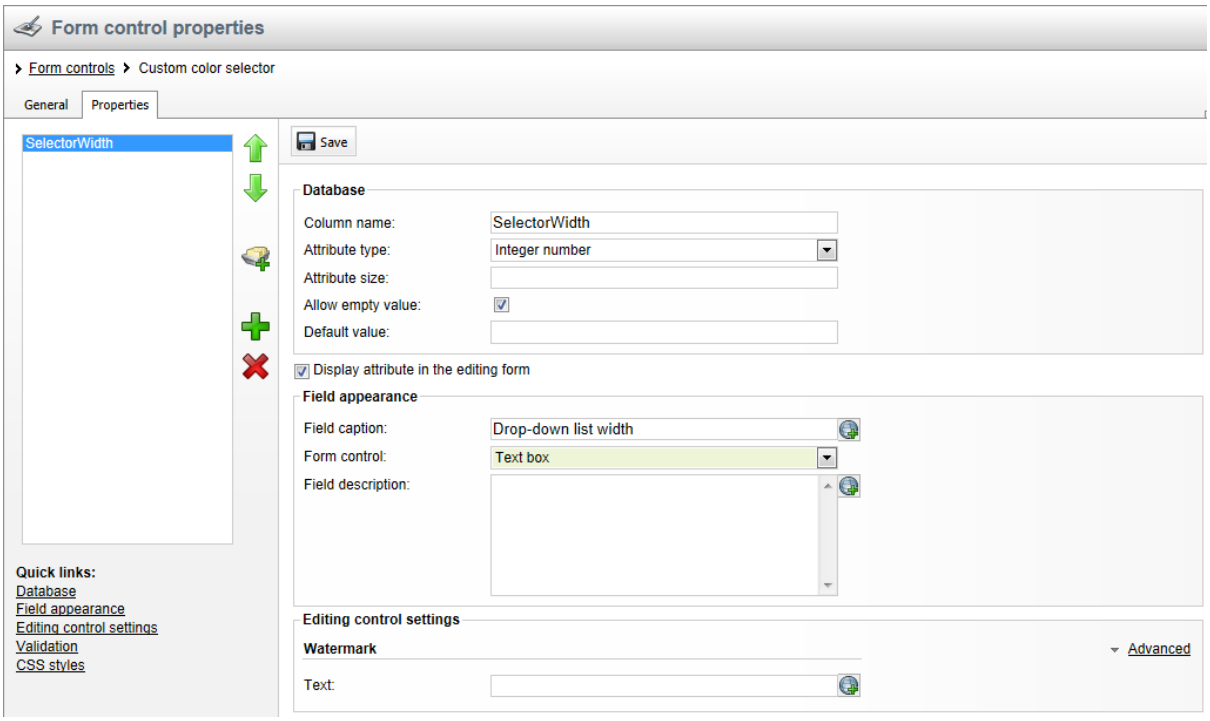
10. Switch to the **Properties** tab where parameters can be defined for the form control. Use the **New attribute** (✚) action and set the following values for the parameter:

- **Column name**: SelectorWidth
- **Attribute type**: Integer number
- **Allow empty value**: true (checked)

- **Display attribute in editing form**: true

- **Field caption**: Drop-down list width
- **Form control**: Text box

When finished, click **Save**.



This parameter will allow users to specify the width of the color selector directly from the administration interface whenever they add this control to a form. The code of the form control already ensures that the value is properly applied.

11. Now we will test this control by placing it onto a document editing form. Go to **Site Manager -> Development -> Document types** and edit (🖉) the **Product** document type. Select the **Fields** tab to access the field editor for this document type. Add two new fields using the **New attribute** (✚) action. Set the following properties for the fields:

- **Column name**: ProductColor
- **Attribute type**: Text
- **Attribute size**: 100
- **Display attribute in editing form**: false

This field will store the name of the color selected for the product. It will not be available in the editing form, its value will be set automatically by the **GetOtherValues()** method of the **ColorSelector.ascx** control (notice that the *Column name* matches the name used in the code of the method).

Click 💾 **Save** and create the next field:

- **Column name**: ProductHexaColor
- **Attribute type**: Text
- **Attribute size**: 100
- **Allow empty value**: true (checked)

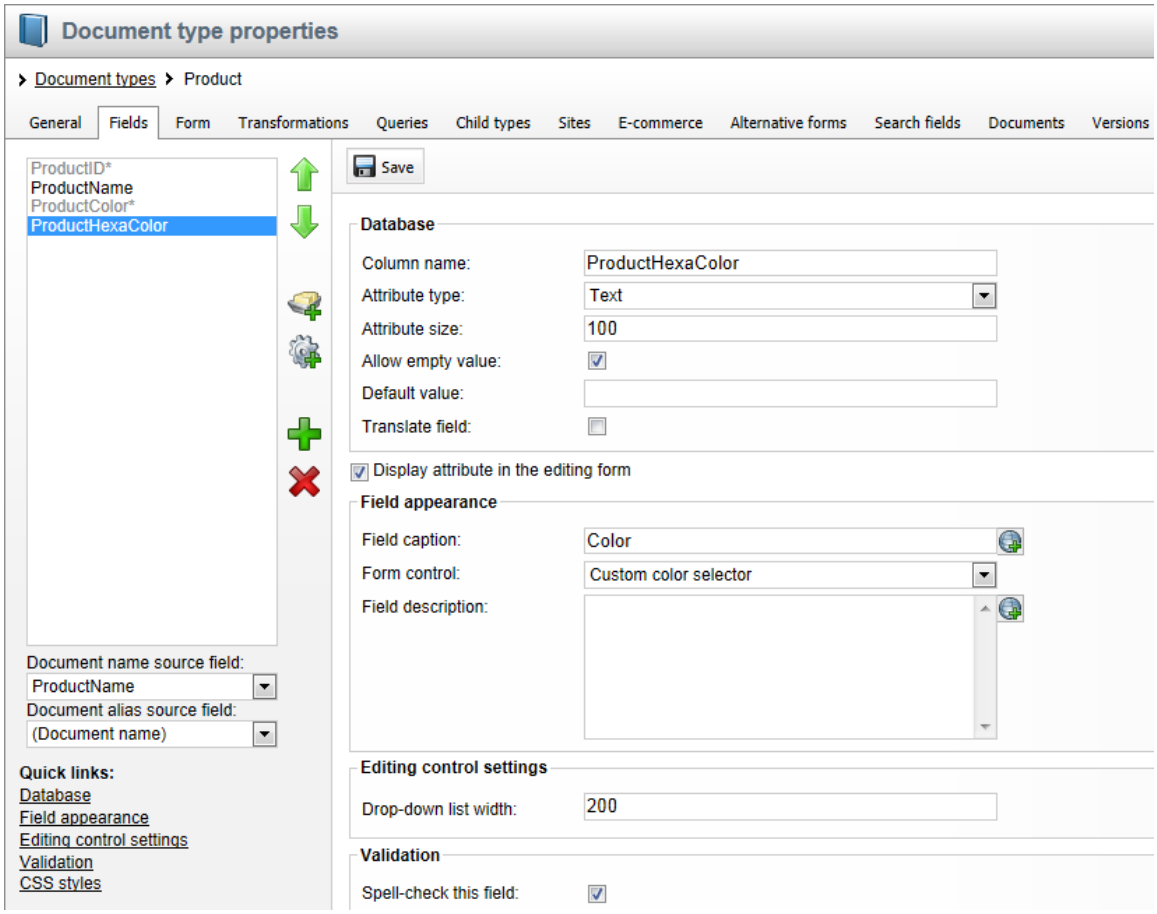
- **Display attribute in editing form**: true

- **Field caption**: Color
- **Form control**: Custom color selector

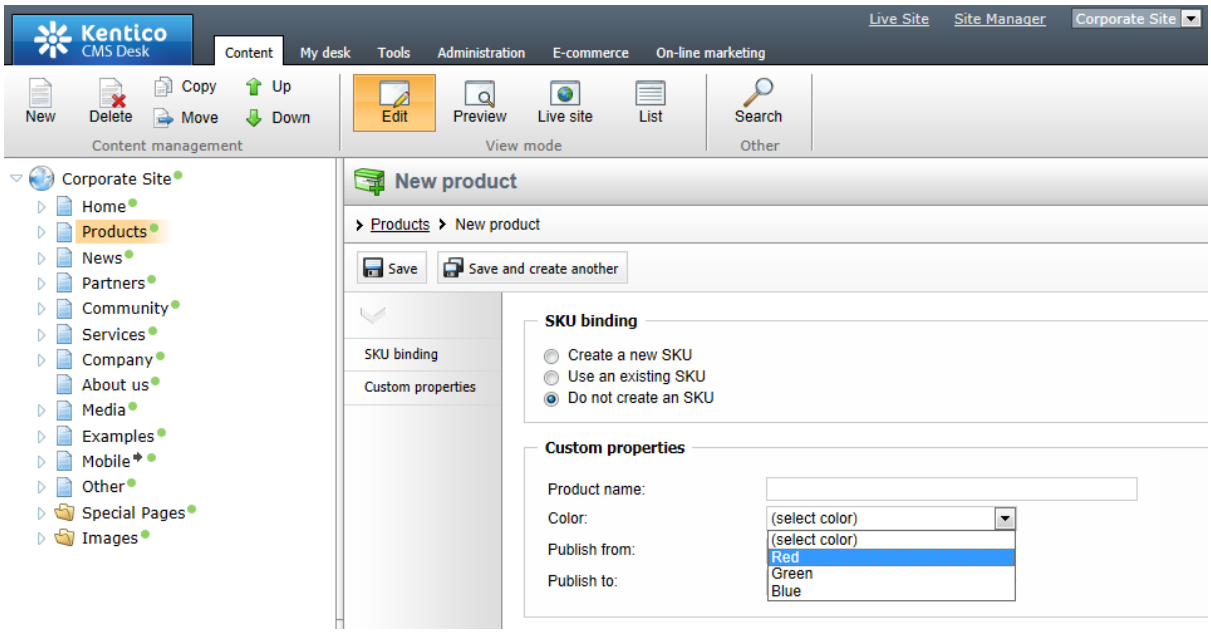
This field will store the hexadecimal code of the selected color. In the code of the form control, this value is handled through the **Value** property. The field will be displayed in the document's editing form according to the design of the custom form control.

Notice that the **Editing control settings** section of this field contains the **Drop-down list width** field. This is the **SelectorWidth** parameter defined for the form control in the previous step. Try to specify the width of the selector by entering some number, for example *200*.

Click **Save** again.



12. Go to **CMS Desk -> Content** and create a new document of the **Product** document type under the **Products** section. For the purposes of this example, choose the **Do not create an SKU** option in the **SKU binding** section. The document's editing form will contain the new form control as shown below:



As you can see, the document field can be managed using the custom form control. The width of the displayed drop-down list will match the value that you entered into the form control's parameter. If you do not choose any color, the validation error message defined in the code of the form control will be displayed.

By implementing custom form controls as shown in this example, it is possible to create editing forms with almost unlimited flexibility, both in the administration interface and on the live site.

Getting and setting values of other fields using the API

The data of the current form can be accessed through the **Form** property of the form control inherited from the **FormEngineUserControl** class.

You can retrieve the values entered into other fields using the following method:

- **Form.GetFieldValue(string columnName)** - returns an object containing the value of the specified field.

For example, the following code could be used to get the value of the **ProductName** field from the current form (*New product* is returned if the field is empty):

[C#]

```
string productName = CMS.GlobalHelper.ValidationHelper.GetString
(Form.GetFieldValue("ProductName"), "New product");
```

To set the value of a field, you can use the following approach:

- **Form.Data.SetValue(string columnName, object value)** - sets a value for the specified field.

To modify the value of a field **before the form is saved**, you need to place the code inside the **IsValid** method of your form control.

7.11 Inline controls

7.11.1 Overview

Inline controls are user controls (ASCX) that can be placed into the text of editable regions using a special expression in format *%%control:MyUserControl%%,* where *MyUserControl* is the name of the inline control. The system dynamically loads the controls when the page is displayed on the live site.

The controls may contain any functionality, such as "polls", "latest news", "mortgage calculator", "travel destination search", etc. The advantage of inline controls is that any content editor can place them anywhere into the text without programming knowledge.

How to insert inline controls into text

Inline controls can be inserted into text using the **Insert inline control** ( ) button on the WYSIWYG editor toolbar. If the text is not edited in the WYSIWYG editor, you can insert inline controls by typing the *%%control:MyUserControl%%* expressions manually. Only controls that have been registered in the system and assigned to the currently edited site can be inserted. This can be done at **Site Manager -> Development -> Inline controls**.

The inline control may also have a single parameter. In this case, you must use one of the following expression formats:

- %%control:BizFormControl?form1%%
- {^BizFormControl|form1^}
- {^BizFormControl|(formname)form1^}

To learn how to create a new inline control and add it to the system, please see the How to develop inline controls topic.

Important!

Inline controls are obsolete and can no longer be added using the WYSIWYG editor toolbar by default. It is recommended to implement your custom controls as Widgets and insert them into text using the **Insert/Edit widget** ( ) button. Further details can be found in the Inline widgets topic.

This approach offers several advantages, such as increased flexibility, an unlimited amount of parameters for the controls, and a more user-friendly configuration dialog.

For the purposes of backward compatibility, existing inline controls are still resolved correctly. If you wish to use inline controls via the WYSIWYG editor, you can enable

this functionality by adding the following key into the **configuration/appSettings** section of your web.config file:

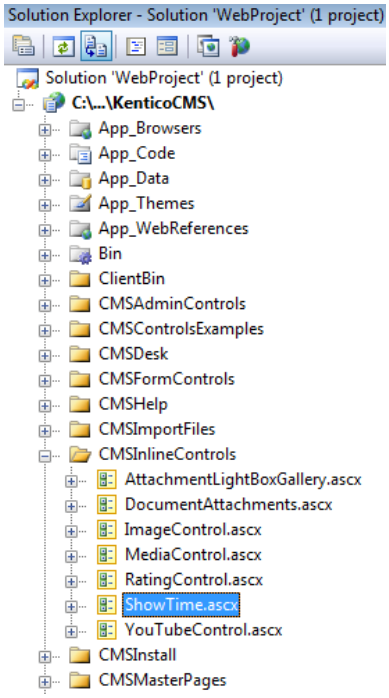
```
<add key="CMSEnableInlineControls" value="true" />
```

7.11.2 How to develop inline controls

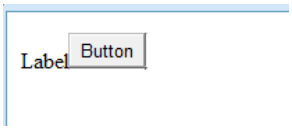
This topic will show you an example of inline control development. We will create a simple control that will display the current time when a button is clicked.

1. Open the web project in Visual Studio (or Visual Web Developer) using the **WebProject.sln** file or using **File -> Open -> Web Site...** in Visual Studio.

2. Right-click the **CMSInlineControls** folder and choose **Add New Item**. Choose to create a new **Web User Control** and call it **ShowTime.ascx**.



3. Switch to the **Design** tab and drag and drop a **Label** control and a **Button** control onto the form:



4. Double-click the Button control and enter the following code into the **Button1_Click** method. It will ensure the displaying of the current date and time by the label control.

[C#]

```
Label1.Text = DateTime.Now.ToString();
```

[VB.NET]

```
Label1.Text = DateTime.Now.ToString()
```

5. Change the following line:

[C#]

```
public partial class CMSInlineControls_ShowTime : System.Web.UI.UserControl

to

public partial class CMSInlineControls_ShowTime :
CMS.ExtendedControls.InlineUserControl
```

[VB.NET]

```
Partial Class CMSInlineControls_ShowTime
    Inherits System.Web.UI.UserControl

to

Partial Class CMSInlineControls_ShowTime
    Inherits CMS.ExtendedControls.InlineUserControl
```

What you did

You have changed the user control so that it inherits from the **InlineUserControl** class. It allows you to access the parameter of the control in the next step.

6. Add the following code to the **Page_Load** method of the control:

[C#]

```
Button1.Text = this.Parameter;
```

[VB.NET]

(The **Page_Load** method is not generated by default in VB.NET)

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
```

```
Handles Me.Load
    Button1.Text = Parameter
End Sub
```

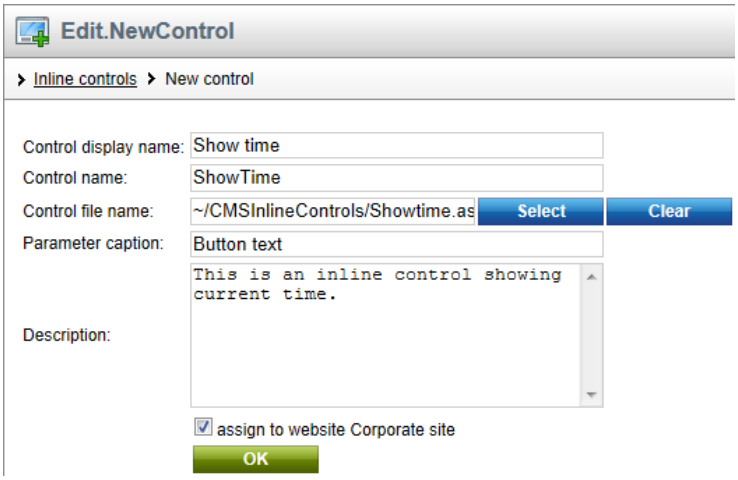
What you did

This code sets the Button1 caption based on the parameter of the inline control.

7. Save the changes.

8. Go to **CMS Site Manager -> Development -> Inline controls**. Click **New control** and enter in the following values:

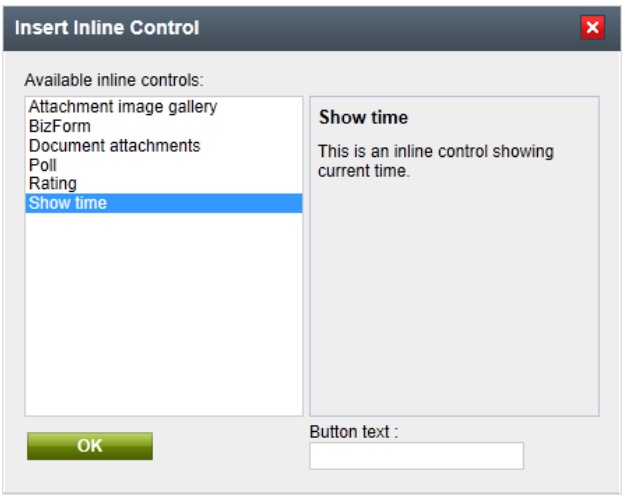
- **Control display name:** Show time
- **Control name:** ShowTime (the name of the user control without the extension)
- **Control file name:** ~/CMSInlineControls/ShowTime.ascx
- **Parameter caption:** Button text



Click **OK**.

9. Now click the **Sites** tab and check the boxes for all sites where content editors should be able to insert this control. Click **OK**.

10. Go to CMS Desk, edit some page with editable regions and click the **Insert Inline Control** ( ) button. Select the control and set the **Button text** value to *Show time*. Click **OK**. The special expression is now inserted into the text.

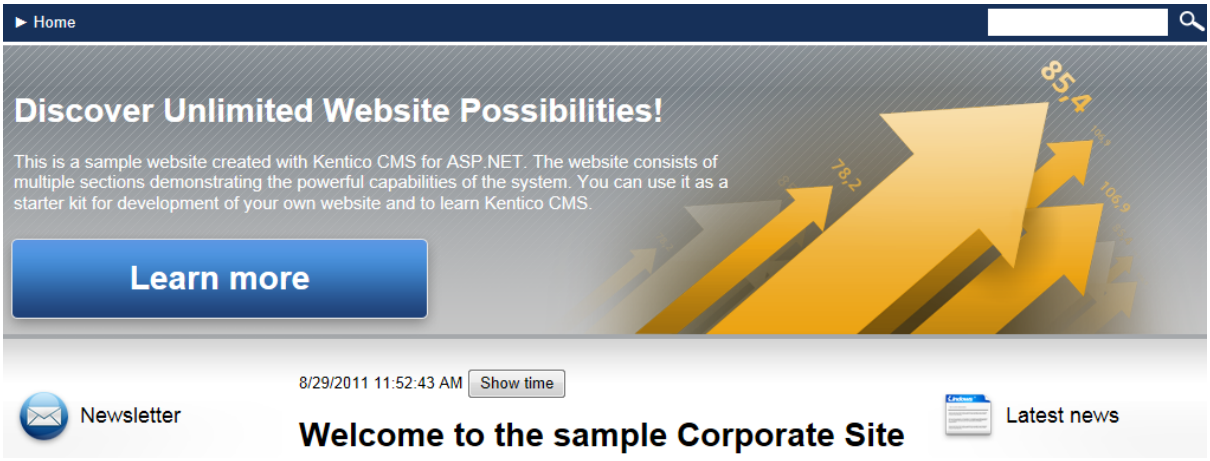


Please note

Due to the fact that in-line controls are obsolete, the **Insert Inline Control** ( ) button is only available in the toolbar after adding the following key into the **configuration/ appSettings** section of your web.config file:

```
<add key="CMSEnableInlineControls" value="true" />
```

11. Click **Save** to save changes and click **Live site** to see the live version of the page. The user control is now displayed inside your text. The button has the caption you have specified. When you click the button, the label displays the current date and time:



7.12 Macro expressions

7.12.1 Overview

Macros are strings that are automatically resolved into their value equivalents. They represent a powerful option that can often eliminate writing custom .NET code.

There are several types of macros, all of which are listed and explained in the Types of macros topic. The most powerful are **context macros**, using which you can achieve most of the results achievable by using all other types of macros. Context macros can contain advanced expressions written in the K# language, whose syntax and features are described in the K# syntax topic. Methods that can be used in K# are listed and described in the Available macro methods topic. Macro parameters known from older versions of Kentico CMS are still functional and available, even though the same results can now usually be achieved through macro methods.

The user interface of Kentico CMS has several built-in features which facilitate entering of macro expressions. These are explained in the Entering macro expressions topic.

One of the ways to leverage macros is defining dynamic conditions for various types of functionality. There are several features available that make it easier to build such conditions. It is even possible to prepare components that allow users to work with conditions without any knowledge of the macro syntax. Please see the Macro conditions topic to learn more.

When certain macros are resolved, security checks are performed to verify that the user who entered the expression has permissions to read the resolved data. These checks are covered in the Macro security topic.

The default functionality of the macro engine can be enhanced by your custom code. It is possible to define custom macro methods and use them in macro expressions along with the default ones. The process of creating such methods is described in the Registering custom macro methods topic. Additionally, macro expressions can be resolved in your custom code using the API, and it is even possible to customize macro resolvers. This is described in the Resolving macros using the API topic.

For performance reasons, the system also provides a way to cache the values of resolved macros, so that certain macros do not need to be resolved repeatedly. More information on this can be found in the Macro result caching topic.

7.12.2 Types of macros

The macro type is determined by the character at the beginning and end of the macro. Macros look like:

{<**type character**><expression><parameters><**type character**>}

Where the **type character** can be one of the following:

- % - Context (data) macro
- $ - Localization macro
- ? - QueryString macro
- @ - Cookie macro
- ^ - Control macro
- & - Path macro
- # - Custom macro
- ~ - Substitution macro

Macros of the same type can be nested using the "index parentheses" syntax, e.g. *{(1)%FullName|(default){(2)%Username%(2)}%(1)}*. Click particular macro types in the listing above to get redirected to their detailed explanation.

Context (data) macros

This type of macros evaluates data based on current context. These macros can be used for example to parametrize web parts' parameters with current document or user values. These macros can be used in combination with the K# language. For more information on its syntax and possibilities, please refer to the K# syntax and Available macro methods topics.

The format of the macro is *{%ColumnName%}* and the value is replaced with the appropriate value of the column based on current context. You can use all column names from **current site** data, **current user** data and **current document** data.

Example: "*Welcome {%FullName%}*" will be resolved as "*Welcome Global administrator*" when the administrator is the current user.

There are also data macros with selectors to precisely identify the source of the data, these macros look like *{%CurrentDocument.DocumentName%}* and can be used for accurate determination of the data source. Availability of the selectors depends on the context where the macros are evaluated.

Localization macros

These macros are usually used on multilingual websites to help localize system strings. There are two types of localization macros:

Basic

Basic localization macros are entered in format *{$string.key$}*. Alternatively, you can enter them as context macros in the *{% GetResourceString("resourcestring.key") %}* format. The system uses the *ResHelper.GetString("string.key")* method and replaces the macro with the appropriate language version of the given resource string.

Example: "*The weather is {$General.OK$}*" is resolved as "*The weather is OK*".

In-place localization

You can include the string and its localized equivalents directly within the macro. Such macros are entered in format *{$=Default string|cs-cz=Czech string|de-de=German string$}*. They either return a localized string for the current language, or the default (first) string if a matching localized value is not available.

Example: **"***The weather is {$=OK|cs-cz=dobre|de-de=gut$}*" is resolved into "*The weather is dobre*" in the Czech culture, "*The weather is gut*" in the German culture and "*The weather is OK*" in any other culture.

QueryString macros

These macros evaluate querystring parameters information. These macros can be used for example to dynamically parameterize the controls by the querystring parameters.

The macros are entered in the *{?querystring_key?}* format. Alternatively, you can also enter them as data macros like the following: *{%QueryString.querystring_key%}*. The macro is replaced by the querystring parameter value.

Example: "*Current node ID: {?nodeid?}*" will be resolved as "*Current node ID: 10*" for a URL like "*default.aspx?nodeid=10*"

Cookie macros

These macros evaluate the values of the current user's browser cookies. For example, these macros can be used to parametrize web parts using client-based persistent values like styles or user options.

The macro is entered in format *{@cookie_name@}*. Alternatively, you can also enter them as data macros like the following: *{%Cookies.cookie_name%}.* The macro is replaced by the given cookie's value.

Example: *"Current style: {@StyleCookie@}*" will be resolved as "*Current style: Red*" if the *StyleCookie* value is set to "*Red*"

Control macros

These macros are resolved into inline controls, which can be used to place dynamic content into editable text regions and HTML layouts. The macros are specified in format *{^BizFormControl^}* and can (usually must) contain parameters for the control using standard parameterized macro syntax, such as *{^BizFormControl\|(FormName)ContactForm^}*. The parameters will be used to initialize the control when the macro is resolved.

Path macros

These macros are entered in the *{&path&}* format. Alternatively, you can also enter them as data macros like the following: *{%Path.path%}*. They can be used to resolve current document *Alias path* the same way as the *Path* property of controls. The macro is replaced by the resolved path.

Example: WhereCondition: *NodeAliasPath LIKE {&../%&}.* In this case, the macro is resolved as the path of the parent document and results in selecting all siblings of the current document and their child documents. This macro is intended mostly for including the document structure context into the controls WHERE condition, but can be used for many more purposes.

See Appendix A - Path expressions for more details on how paths can be entered.

Custom macros

Custom macros allow you to define your own macro expressions. Their basic format is: *{#Expression#}*. Alternatively, you can also enter custom macros as data macros with the *Custom.* prefix: *{% Custom.Expression%}*.

Defining custom macros

When a custom macro needs to be resolved, the system calls the methods registered as handlers for the **MacroResolver.OnResolveCustomMacro** event. To create a new custom macro, you must

implement a handler method that ensures the resolving of individual expressions into the appropriate results.

1. Open your web project in Visual Studio and add a new class into the **App_Code** folder (or **Old_App_Code** if the project is installed as a web application). For example, name the class **CustomMacroLoader.cs**.

2. Edit the class and add the following references:

[C#]

```
using CMS.SettingsProvider;
using CMS.GlobalHelper;
```

3. Delete the default class declaration and its content. Instead, extend the **CMSModuleLoader** partial class and define a new attribute:

[C#]

```
[CustomMacroLoader]
public partial class CMSModuleLoader
{
    /// <summary>
    /// Attribute class ensuring the registration of macro handlers.
    /// </summary>
    private class CustomMacroLoader : CMSLoaderAttribute
    {
        ...
    }
}
```

4. Enter the following code into the **CustomMacroLoader** attribute class:

[C#]

```
/// <summary>
/// Called automatically when the application starts.
/// </summary>
public override void Init()
{
    // Assigns a custom macro resolving handler.
    MacroResolver.OnResolveCustomMacro += MacroResolver_OnResolveCustomMacro;
}


/// <summary>
/// Resolves custom macros.
/// </summary>
/// <param name="sender">Sender</param>
/// <param name="e">Event arguments representing the resolved macro</param>
```

```
private void MacroResolver_OnResolveCustomMacro(object sender, MacroEventArgs e)
{
    // Checks that the macro is not resolved yet.
    if (!e.Match)
    {
        // Defines the return values of specific custom macro expressions.
        switch (e.Expression.ToLower())
        {
            // Handles the {#CustomExpression#} macro.
            case "customexpression":
                e.Match = true;
                e.Result = "Resolved expression";
                break;
        }
    }
}
```

- The override of the **Init** method registers the *MacroResolver_OnResolveCustomMacro* method as a handler for the *OnResolveCustomMacro* event.
- The **MacroResolver_OnResolveCustomMacro** method defines the return value of the custom expression. The sample code only assigns a string constant as the macro result, but you may add any required API logic to set the value dynamically. You can create any number of custom macros by adding further cases into the switch statement.

The system now recognizes the *{#CustomExpression#}* macro and resolves it into the *Resolved expression* string.

Substitution macros

Substitution macros are entered in format *{~expression~}*. They are similar to Custom macros. The difference is that the system resolves substitution macros **in page output code**, i.e. the page's final code before it is sent to the browser for rendering. The main advantage of substitution macros is that they are resolved on each request, even for pages that use full-page caching.

Defining substitution macros

When a substitution macro needs to be resolved, the system calls the methods registered as handlers for the **OutputFilter.OnResolveSubstitution** event. To create a new substitution macro, you must implement a handler method that ensures the resolving of individual expressions into the appropriate results.

1. Open your web project in Visual Studio and add a new class into the **App_Code** folder (or **Old_App_Code** if the project is installed as a web application). For example, name the class **CustomMacroLoader.cs**.

2. Edit the class and add the following references:

[C#]

```
using CMS.SettingsProvider;
```

```
using CMS.GlobalHelper;
using CMS.CMSOutputFilter;
using CMS.OutputFilter;
```

3. Delete the default class declaration and its content. Instead, extend the **CMSModuleLoader** partial class and define a new attribute:

[C#]

```
[CustomMacroLoader]
public partial class CMSModuleLoader
{
    /// <summary>
    /// Attribute class ensuring the registration of macro handlers.
    /// </summary>
    private class CustomMacroLoader : CMSLoaderAttribute
    {
        ...
    }
}
```

4. Enter the following code into the **CustomMacroLoader** attribute class:

[C#]

```
/// <summary>
/// Called automatically when the application starts.
/// </summary>
public override void Init()
{
    // Assigns a substitution macro resolving handler.
    OutputFilter.OnResolveSubstitution += OutputFilter_OnResolveSubstitution;
}


/// <summary>
/// Resolves output substitution macros.
/// </summary>
/// <param name="sender">Sender</param>
/// <param name="e">Event arguments representing the resolved macro</param>
private void OutputFilter_OnResolveSubstitution(object sender,
SubstitutionEventArgs e)
{
    // Checks that the macro is not resolved yet.
    if (!e.Match)
    {
        // Defines the return values of specific substitution macros.
        switch (e.Expression.ToLower())
        {
            // Handles the {~CustomSubstitution~} macro.
            case "customsubstitution":
```

```
                e.Match = true;
                e.Result = "Resolved substitution";
                break;
            }
        }
    }
```

- The override of the **Init** method registers the *OutputFilter_OnResolveSubstitution* method as a handler for the *OnResolveSubstitution* event.
- The **OutputFilter_OnResolveSubstitution** method defines the return value of the custom expression. The sample code only assigns a string constant as the substitution result, but you may add any required API logic to set the value dynamically. You can create any number of substitution macros by adding further cases into the switch statement.

The system now recognizes the *{~CustomSubstitution~}* macro and resolves it into the *Resolved substitution* string.

7.12.3 K# syntax

The syntax of data macros is very similar to syntax of the **C#** language — that is why the name **K#** was given to the macro language. The following text summarizes the main specifics and features of the language and provides examples of how individual features can be used:

Basic features

- K# is **case-insensitive**. This means that upper- or lower-case letters are handled equally in the macro expressions.

- K# supports **variable declarations**. The scope where declared variables can be used spans from the point of their declaration towards the end of the whole text where the macro is used (e.g. an e-mail template, transformation, etc.).

```
// returns "12"
{% x = 5; x + 7 %}
```

- K# supports **compound expressions**. These are expressions consisting of multiple sub-expressions separated by semicolons, where the result returned by the macro is the result of the last sub-expression.

```
// returns "10"
{% x = 5; y = 3; x += 2; x + y %}
```

- K# supports **suffix method calls**.

```
// returns "TEST"
{% "test".ToUpper() %}
```

- K# supports **infix method calls**.

```
// returns "TEST"
{% ToUpper("test") %}
```

- K# supports **indexing**. The following is a list of indexable objects:
  - DataRow, DataRowView, DataRowContainer (returns value at given index)
  - DataTableContainer (returns row at given index)
  - DataSetContainer (returns table at given index)
  - String (returns char at given index)
  - IList
  - IEnumerable

```
//returns "e"
{% "hello"[1] %}

// returns value of the FirstName column from the DataRow
{% dataRow["FirstName"] %}
```

- K# supports **lambda expressions**, which are ad-hoc declarations of in-line functions that can be used in your macro expressions. The scope where declared lambda expressions can be used span from the point of their declaration towards the end of the whole text where the macro is used (e.g. an e-mail template, transformation, etc.).

```
// returns "6"
{% myMul = ((x, y) => x * y); myMul(2,3) %}

// returns "4"
{% mySucc = (x => x + 1); mySucc(3) %}
```

- K# supports **comments in macro expression text**. You can use one-line, multi-line and in-line comments.

```
{%

// this is a one line comment, it spans across one single line initiated by two
forwardslashes

/*
this is a multi-line comment, it can span across any number of lines
it begins with the forwardslash-asterisk sequence and ends with the asterisk-
forwardslash sequence
*/

x = 5; y = 3; /* this is an in-line comment nested in the middle of an expression
*/ x+= 2; x + y
```

```
%}
```

Flow control commands

K# supports the following types of flow control commands:

- **Ternary operators** for returning of conditional output written in the following format: <*condition*> *?* <*expressions 1*> *:* <*expressions 2*>. In case of a true condition, return value of <*expressions 1*> is returned. In case of a false condition, return value of <*expressions 2*> is returned.

```
// returns "The second parameter is grater."
{% GreaterThan(1,2) ? "The first parameter is greater." : "The second parameter is
greater." %}
```

- The **if** flow control command written in format: *if (*<*condition*>*) {*<*expressions*>*}.* In case of a true condition, return value of <*expressions*> is returned. In case of a false condition, null is returned.

```
// returns "z is lesser than 3"
{% z = 1; if (z<3) {"z is lesser than 3"} %}
```

- The **if-else** flow control command written in format: *if (*<*condition*>*) {*<*expressions 1*>*} else {*<*expressions 2*>*}*. In case of a true condition, return value of <*expressions 1*> is returned. In case of a false condition, return value of <*expressions 2*> is returned.

```
// returns "z is greater than 3"
{% z = 5; if (z<3) {"z is lesser than 3"} else {"z is greater than 3"} %}
```

- The **for** iteration command written in format: *for (*<*init_expression*>*;* <*condition*>*;* <*incr_expression*>*) {*<*expressions*>*}*. The command itself has a return value. It returns a concatenated list of return values of <*expressions*> from each iteration of the loop.

```
// returns "5"
{% z = 0; for (i = 0; i < 5; i++) { z += 1 }; z %}

// returns "12345"
{% z = 0; for (i = 0; i < 5; i++) { z += 1 } %}
```

- The **foreach** iterator written in format: *foreach (*<*variable*> *in* <*enumerable*>*) {*<*expressions*>*}*. The command itself has a return value. It returns a concatenated list of return values of <*expressions*> from each iteration of the loop.

```
// returns "HELLO"
{% z = ""; foreach (x in "hello") {z += x.toupper()}; z %}
```

```
// returns "H HE HEL HELL HELLO"
{% z = ""; foreach (x in "hello") {z += x.toupper()}%}
```

- The **while** loop command written in format: *while (<condition>) {<expressions>}*. The command itself has a return value. It returns a concatenated list of return values of <*expressions*> from each iteration of the loop.

```
// returns "10"
{% z = 1; while (z<10) {++z}; z %}

// returns "2345678910"
{% z = 1; while (z<10) {++z} %}
```

- The **break** command in loops. This command terminates the nearest enclosing loop in which it appears.

```
// returns "12345"
{% z = 0; while (z < 10) {if (z > 4) {break}; ++z} %}
```

- The **continue** command in loops. This command passes control to the next iteration of the closest enclosing loop in which it appears.

```
// returns "01245"
{% for (i=0; i<=5 ; i++) {if (i == 3) {continue}; i} %}
```

- The **return** command in loops. The command returns current value of the provided expression as final result of the macro in which it appears and terminates its further processing. If no expression is provided, it returns the current console output (see below for more details on console output).

```
// returns "3"
{% i = 0; while (i < 10) {i++; if (i>2) {return i;}} %}

// returns "012"
{% i = 0; while (i < 10) {print(i++); if (i > 2) {return;}} %}
```

Open conditions and loops

When defining conditions or loops through the commands described above, it is possible to leave the body of the loop/condition open and have it closed later in another macro expression. This allows you to apply the command to text content or HTML code placed between the macro expressions, for example:

```
{% date = CurrentDateTime; if (date.Year > 2011) { %}
The current date is: {%date%}. The registration period has ended.
{% } %}
```

As you can see, additional macro expressions may also be nested inside open loops or conditions. This technique can be particularly useful in macro-based transformations or various types of templates.

Console output

- K# supports **console output** using the *print(<expressions>)* or *println(<expressions>)* syntax. This is useful if you want to output current values during loop iterations without the need to declare a variable where values would be stored in each iteration and returned in the end. Console output has higher priority than standard output of an expression, as you can see in the example below.

```
// returns "2345678910", the "ignore" string at the end is ignored as console
output has higher priority
{% z = 1; while (z < 10) {print(++z)}; "ignored" %}
```

Overloaded operators

- The + operator can be used in the following ways:

  Double (Double leftOperand)+(Double rightOperand)
  Takes two numbers as operands and returns their sum as Double.

  DateTime (DateTime leftOperand)+(TimeSpan rightOperand)
  Takes a DateTime and a TimeSpan as operands, adds the TimeSpan to the DateTime and returns the result as DateTime.

  TimeSpan (TimeSpan leftOperand)+(TimeSpan rightOperand)
  Takes two TimeSpans as operands and returns their sum as TimeSpan.

  String (String leftOperand)+(String rightOperand)
  Takes two Strings as operands and returns their concatenation as String. A String concatenation is also performed if none of the operand type combinations above are detected (e.g. if you provide a DateTime together with a String, you will get a concatenation of their String representations).

- The - operator can be used in the following ways:

  Double (Double leftOperand)-(Double rightOperand)
  Takes two numbers as arguments, subtracts the second one from the first one and returns the result as a Double value.

  DateTime (DateTime leftOperand)-(TimeSpan rightOperand)
  Subtracts a TimeSpan entered as right operand from a DateTime entered as left operand and returns the result as DateTime.

TimeSpan (DateTime leftOperand)-(DateTime rightOperand)
Subtracts a DateTime entered as right operand from a DateTime entered as left operand and returns the result as TimeSpan.

TimeSpan (TimeSpan leftOperand)-(TimeSpan rightOperand)
Subtracts a TimeSpan entered as right operand from a TimeSpan entered as left operand and returns the result as TimeSpan.

Differences from C# syntax

- To perform the modulus operation (computing the remainder after division of two integer operands), use the **mod** operator or the **Modulo (Int32 left, Int32 right)** macro method. You **cannot** use the % operator for this purpose.

```
// returns "1"
{% x = 5; x mod 2 %}
```

- The **%** character provides a way to enter percentage values. It converts the associated number to a double equivalent (i.e. multiplies the number by 0.01).

```
// returns "0.3"
{% 30% %}
```

7.12.4 Available macro methods

You can use a large variety of methods in your macro expressions. This topic lists all methods intended specifically for use in Kentico CMS macro expressions that are members of the *GlobalHelper.MacroMethods.cs*. They are listed in the following categories according to the type of functionality they provide:

Boolean methods

Boolean (Boolean left) && (Boolean right)
Returns logical product of given parameters.
Boolean left: Left operand.
Boolean right: Right operand.

Boolean (Boolean left) || (Boolean right)
Returns logical addition of given parameters.
Boolean left: Left operand.
Boolean right: Right operand.

Boolean Not (Boolean value)
Boolean ! (Boolean value)
Returns logical negation of the provided value.
Boolean value: Value to negate.

Boolean And (Boolean left, Boolean right)
Returns logical product of given parameters.
Boolean left: Left operand.
Boolean right: Right operand.

Boolean Or (Boolean left, Boolean right)
Returns logical addition of given parameters.
Boolean left: Left operand.
Boolean right: Right operand.

Int32 LogicalAnd (Int32 left, Int32 right)
Int32 (Int32 left) & (Int32 right)
Returns bit logical product of given parameters.
Int32 left: Left operand.
Int32 right: Right operand.

Int32 LogicalOr (Int32 left, Int32 right)
Int32 (Int32 left) | (Int32 right)
Returns bit logical addition of given parameters.
Int32 left: Left operand.
Int32 right: Right operand.

Int32 LogicalXor (Int32 left, Int32 right)
Int32 (Int32 left) ^ (Int32 right)
Returns bit logical exclusive addition of given parameters.
Int32 left: Left operand.
Int32 right: Right operand.

Int32 LeftShift (Int32 left, Int32 right)
Int32 (Int32 left) << (Int32 right)
Shifts its first operand left by the number of bits specified by its second operand.
Int32 left: Left operand.
Int32 right: Right operand.

Int32 RightShift (Int32 left, Int32 right)
Int32 (Int32 left) >> (Int32 right)
Shifts its first operand right by the number of bits specified by its second operand.
Int32 left: Left operand.
Int32 right: Right operand.

Comparison methods

Boolean GreaterThan(Double left, Double right)
Boolean (Double left) > (Double right)
Returns true if the first parameter is greater than second. The method also accepts DateTime values as operands.
Double left: Left operand.
Double right: Right operand.

Boolean LowerThan(Double left, Double right)
Boolean (Double left) < (Double right)
Returns true if the first parameter is lower than second. The method also accepts DateTime values as operands.
Double left: Left operand.
Double right: Right operand.

Boolean GreaterThanOrEqual(Double left, Double right)
Boolean (Double left) >= (Double right)
Returns true if the first parameter is greater than or equal to second. The method also accepts DateTime values as operands.
Double left: Left operand.
Double right: Right operand.

Boolean LowerThanOrEqual(Double left, Double right)
Boolean (Double left) <= (Double right)
Returns true if the first parameter is lower than or equal to second. The method also accepts DateTime values as operands.
Double left: Left operand.
Double right: Right operand.

Boolean Equals(Object left, Object right)
Boolean (Object left) == (Object right)
Returns true if the first parameter is equal to the second. The method also accepts DateTime values as operands.
Object left: Left operand.
Object right: Right operand.

Boolean NotEquals(Object left, Object right)
Boolean (Object left) != (Object right)
Returns true if first parameter is not equal to the second.
Object left: Left operand.
Object right: Right operand.

Conversion methods

Int32 ToInt(Object value, *Int32 defaultValue*)
Converts value to int, if it is not possible, returns default value. This method is overloaded, while the *Int32 defaultValue* parameter is optional.
Object value: Object to convert.
Int32 defaultValue: Default value.

Boolean ToBool(Object value, *Boolean defaultValue*)

Converts value to bool, if it is not possible, returns default value. This method is overloaded, while the *Boolean defaultValue* parameter is optional.
Object value: Object to convert.
Boolean defaultValue: Default value.

Double ToDouble(Object value, *Double defaultValue*, *String culture*)
Converts value to double, if it is not possible, returns default value. This method is overloaded, while the *Double defaultValue* and *String culture* parameters are optional.
Object value: Object to convert.
Double defaultValue: Default value.
String culture: Culture to use.

Guid ToGuid(Object value, *Guid defaultValue*)
Converts value to Guid, if it is not possible, returns default value. This method is overloaded, while the *Guid defaultValue* parameter is optional.
Object value: Object to convert.
Guid defaultValue: Default value.

DateTime ToDateTime(Object value, *DateTime defaultValue*, *String culture*)
Converts value to DateTime, if it is not possible, returns default value. This method is overloaded, while the *DateTime defaultValue* and *String culture* parameter are optional.
Object value: Object to convert.
DateTime defaultValue: Default value.
String culture: Culture to use.

DateTime FromOADate(Double value)
Converts double representation of DateTime (OLE Automation Date) to DateTime.
Double value: OADate double representation to convert.

TimeSpan ToTimeSpan(Object value)
Parses the provided string value and converts it to TimeSpan.

Data manipulation methods

Object GetValue(ISimpleDataContainer container, String column)
Gets value of the specified column from the object.
ISimpleDataContainer container: Data container.
String column: Column name.

Object GetProperty(IHierarchicalObject collection, String property)
Gets value of the specified property from the object.
IHierarchicalObject collection: DataContainer.
String property: Property name.

Object GetItem(IEnumerable collection, Int32 index)
Gets the item stored at the specified index from the collection.
IEnumerable collection: Collection.
Int32 index: Index of the item.

AbstractObjectCollection OrderBy(AbstractObjectCollection collection, String orderBy)
Applies an Order by clause to the collection.
AbstractObjectCollection collection: Collection to order.

String orderBy: ORDER BY column(s).

AbstractObjectCollection Where(AbstractObjectCollection collection, String where)
Can be used to filter the objects in a collection by applying a Where condition.
AbstractObjectCollection collection: Collection to filter.
String where: WHERE condition.

AbstractObjectCollection TopN(AbstractObjectCollection collection, Int32 parameters)
Limits how many objects should be included in the collection.
AbstractObjectCollection collection: Collection to filter.
Int32 parameters: TOP N parameter.

AbstractObjectCollection Columns (AbstractObjectCollection collection, String columns)
Specifies which object data columns should be included in the collection.
AbstractObjectCollection collection: Collection to filter.
String columns: Columns parameter.

InfoObjectCollection Filter(InfoObjectCollection collection, String condition)
Returns the collection filtered by given condition.
InfoObjectCollection collection: Collection of items.
String condition: Filtering macro condition to evaluate for each item.

TreeNodeCollection ClassNames(TreeNodeCollection collection, String classNames)
Filters a collection of tree nodes (documents) to only include the specified document types.
TreeNodeCollection collection: Collection of documents.
String classNames: A list of document types that should be returned in the filtered collection (specified by their code names and separated by semicolons).

ArrayList List(Object items)
Creates an ArrayList from given list of items.
Object items: List of items to add.

Boolean InList(Object object, IEnumerable collection)
Returns true if specified object exists within the given collection.
Object object: Object which should be checked for existence within the collection.
IEnumerable collection: Collection of items.

Boolean Exists(InfoObjectCollection collection, String condition)
Returns true if there is at least one object in the collection which matches given condition.
InfoObjectCollection collection: Collection of items.
String condition: Filtering macro condition to evaluate for each item.

Object Cache(Object expression, Int32 cacheMinutes, Boolean condition, String cacheItemName, String cacheItemNameParts, CMSCacheDependency cacheDependency)
Evaluates the given expression and puts it to the cache. The expression is evaluated only when not found in cache.
Object expression: Expression to be evaluated and cached.
Int32 cacheMinutes: Cache minutes
Boolean condition: Cache condition
String cacheItemName: Cache item name.
String cacheItemNameParts: Cache item name parts.
CMSCacheDependency cacheDependency: Cache dependency object (use GetCacheDependency method to get the object).

CMSCacheDependency GetCacheDependency(String dependencies)
Returns CacheDependency object created from given string.
String dependencies: Cache dependency strings.

DateTime methods

DateTime AddMilliseconds(DateTime datetime, Int32 milliseconds)
Adds the specified number of milliseconds to the specified base DateTime value.
DateTime datetime: Base DateTime value to which the milliseconds should be added.
Int32 milliseconds: Number of milliseconds to be added to the base DateTime value.

DateTime AddSeconds(DateTime datetime, Int32 seconds)
Adds the specified number of seconds to the specified base DateTime value.
DateTime datetime: Base DateTime value to which the seconds should be added.
Int32 seconds: Number of seconds to be added to the base DateTime value.

DateTime AddMinutes(DateTime datetime, Int32 minutes)
Adds the specified number of minutes to the specified base DateTime value.
DateTime datetime: Base DateTime value to which the minutes should be added.
Int32 minutes: Number of minutes to be added to the base DateTime value.

DateTime AddHours(DateTime datetime, Int32 hours)
Adds the specified number of hours to the specified base DateTime value.
DateTime datetime: Base DateTime value to which the hours should be added.
Int32 hours: Number of hours to be added to the base DateTime value.

DateTime AddDays(DateTime datetime, Int32 days)
Adds the specified number of days to the specified base DateTime value.
DateTime datetime: Base DateTime value to which the days should be added.
Int32 days: Number of days to be added to the base DateTime value.

DateTime AddWeeks(DateTime datetime, Int32 weeks)
Adds the specified number of weeks to the specified base DateTime value.
DateTime datetime: Base DateTime value to which the weeks should be added.
Int32 weeks: Number of weeks to be added to the base DateTime value.

DateTime AddMonths(DateTime datetime, Int32 months)
Adds the specified number of months to the specified base DateTime value.
DateTime datetime: Base DateTime value to which the months should be added.
Int32 months: Number of months to be added to the base DateTime value.

DateTime AddYears(DateTime datetime, Int32 years)
Adds the specified number of years to the specified base DateTime value.
DateTime datetime: Base DateTime value to which the years should be added.
Int32 years: Number of years to be added to the base DateTime value.

GlobalHelper methods

String UrlEncode (String url)
Encodes the URL.
String url: URL to be encoded.

String ResolveBBCode (String text)
Resolves BB code in the provided string.
String text: Text to be resolved.

String JSEscape (String text)
Escapes the string for usage in JavaScript to avoid XSS.
String text: Text to be processed.

String SQLEscape (String text)
Escapes the string for usage in SQL to avoid SQL injection. Single quotes are replaced by two single quotes.
String text: Text to be processed.

String StripTags (String text)
Removes all HTML tags from the provided string.
String text: Text to be processed.

String ResolveUrl (String url)
Resolves the URL.
String url: URL to be resolved.

String UnresolveUrl (String url)
Unresolves the URL.
String url: URL to be unresolved.

String MapPath (String path)
Maps the virtual path to the disk.
String path: Virtual path.

String LimitLength (String text, Int32 length)
Limits the length of the string to specified number of characters.
String text: Text to be processed.
Int32 length: Length of the result.

String RegexReplace (String text, String regex, String replacement)
Replaces the string using regular expressions.
String text: Text to be processed.
String regex: Regular expression.
String replacement: Replacement string.

String GetMatch (String text, String regex)
Matches the string provided in the first parameter to the regular expression and returns the match.
String text: Text to be processed.
String regex: Regular expression.

Boolean Matches (String text, String regex)
Matches the string provided in the first parameter to the regular expression and returns true if the it matches, false if not.
String text: Text to be processed.
String regex: Regular expression.

String Localize (String inputText, String culture)

Localizes given text (resolves localization macros). This method is overloaded, while the *String culture* parameter is optional.
String inputText: Text to be localized.
String culture: Required culture of the translation.

String GetResourceString (String resourceStringKey, *String culture*)
Translates given resource string. This method is overloaded, while the *String culture* parameter is optional.
String resourceStringKey: Name of the resource string.
String culture: Target culture of the translation.

Mathematical methods

The following methods are only offered under the **Math** namespace by macro autocompletion (type e.g. *Math.Add(2,3)*). However, if you enter them manually without the namespace, they are functional as well.

Double Multiply (Double[] parameters)
Double (Double parameters) * (Double parameter) * ...
Returns product of the parameters.
Double[] parameters: List of numbers to multiply.
Double parameter: Number to multiply.

Double Divide (Double left, Double right)
Double (Double left) / (Double right)
Divides two number values.
Double left: Left operand.
Double right: Right operand.

Double Percent (Double percent)
Double (Double percent) %
Returns the 0.01 multiple of the first argument.
Double percent: Number of percent (number to be multiplied by 0.01).

Int32 Modulo (Int32 left, Int32 right)
Int32 mod (Int32 left, Int32 right)
Returns modulo of two values.
Int32 left: Left operand.
Int32 right: Right operand.

Double Max (Double[] parameters)
Returns maximum from given numbers.
Double parameters: List of numbers.

Double Min (Double[] parameters)
Returns minimum from given numbers.
Double parameters: List of numbers.

Double Abs (Double number)
Returns the absolute value of a specified number.
Double number: Number to do the operation on.

Double Acos (Double number)

Returns the angle whose cosine is the specified number.
Double number: Number to do the operation on.

Double Asin (Double number)
Returns the angle whose sine is the specified number.
Double number: Number to do the operation on.

Double Atan (Double number)
Returns the angle whose tangent is the specified number.
Double number: Number to do the operation on.

Double Ceiling (Double number)
Returns the smallest whole number greater than or equal to the specified number.
Double number: Number to do the operation on.

Double Cos (Double number)
Returns the cosine of the specified angle.
Double number: Number to do the operation on.

Double Cosh (Double number)
Returns the hyperbolic cosine of the specified angle.
Double number: Number to do the operation on.

Double Exp (Double number)
Returns e raised to the specified power.
Double number: Number to do the operation on.

Double Floor (Double number)
Returns the largest whole number less than or equal to the specified number.
Double number: Number to do the operation on.

Double Log (Double number)
Returns the logarithm of the specified number.
Double number: Number to do the operation on.

Double Log10 (Double number)
Returns the base 10 logarithm of the specified number.
Double number: Number to do the operation on.

Double Pow (Double base, Double exp)
Returns a specified number raised to the specified power.
Double base: Base.
Double exp: Exponent.

Double Round (Double number)
Returns the nearest whole number to the specified number.
Double number: Number to do the operation on.

Double Sign (Double number)
Returns a value indicating the sign of a number.
Double number: Number to do the operation on.

Double Sin (Double number)

Returns the sine of the specified angle.
Double number: Number to do the operation on.

Double Sinh (Double number)
Returns the hyperbolic sine of the specified angle.
Double number: Number to do the operation on.

Double Sqrt (Double number)
Returns the square root of a specified number.
Double number: Number to do the operation on.

Double Tan (Double number)
Returns the tangent of the specified angle.
Double number: Number to do the operation on.

Double Tanh (Double number)
Returns the hyperbolic tangent of the specified angle.
Double number: Number to do the operation on.

Int32 ~ (Int32 number)
Returns a bitwise complement.
Int32 number: Number to do the operation on.

Boolean IsOdd (Int32 number)
Returns true if the given number is odd.
Int32 number: Number to do the operation on.

Boolean IsEven (Int32 number)
Returns true if the given number is even.
Int32 number: Number to do the operation on.

Double Average(InfoObjectCollection collection, String columnName)
Returns an average of all collection items over specified column.
InfoObjectCollection collection: Collection of items.
String columnName: Name of the column.

Double Sum(InfoObjectCollection collection, String columnName)
Returns a sum of all collection items over specified column.
InfoObjectCollection collection: Collection of items.
String columnName: Name of the column.

Int32 GetRandomInt(Int32 minValue, Int32 maxValue, Int32 seed)
Returns a random integer within a specified range.
Int32 minValue: The inclusive lower bound of the number.
Int32 maxValue: The inclusive upper bound of the number.
Int32 seed: Seed for the pseudorandom generator - default is system time.

Double GetRandomDouble(Int32 minValue, Int32 maxValue, Int32 seed)
Returns a random double within a specified range.
Int32 minValue: The inclusive lower bound of the number.
Int32 maxValue: The inclusive upper bound of the number.
Int32 seed: Seed for the pseudorandom generator - default is system time.

Membership methods

Boolean IsInRole(Object User, String userRole)
Returns true if user is in role.
Object User: User info object
String userRole: Name of the role to test whether user is in.

Boolean HasMembership(Object User, String userMembership)
Returns true if user is in membership.
Object User: User info object
String userMembership: Name of the membership to test whether user is in.

Boolean IsInGroup(Object User, String userGroup)
Returns true if user is in group.
Object User: User info object
String userGroup: Name of the group to test whether user is in.

String methods

String ToLower (String text)
Converts the string to lower case letters.
String text: Text to convert.

String ToUpper (String text)
Converts the string to upper case letters.
String text: Text to convert.

Boolean EndsWith (String text, String findText)
Determines whether the end of the first string matches the second string.
String text: Text to check.
String findText: Text to find.

Boolean StartsWith (String text, String findText)
Determines whether the beginning of the first string matches the second string.
String text: Text to check.
String findText: Text to find.

String Substring (String text, Int32 index, *Int32 length*)
Retrieves a substring from the base string. The method is overloaded, while the *Int32 length* parameter is optional.
String text: Base text.
Int32 index: The index of the start of the substring.
Int32 length: The number of characters in the substring.

String[] Split (String text, String delimiters)
Identifies the substrings in this instance that are delimited by one or more characters specified in an array, then places the substrings into a string array.
String text: String to split.
String delimiters: Delimiters string. Each character of this string will be taken as a delimiter.

String Join (IEnumerable list, String separator)
Concatenates a specified separator string between each element of a specified string array, yielding a

single concatenated string.
IEnumerable list: IEnumerable to be joined.
String separator: Separator string.

Boolean Contains (String text, String search)
Returns a value indicating whether the string specified in the second parameter occurs within the string specified in the first one.
String text: Text to search in.
String search: Text to search.

String Trim (String text, *String charsToTrim*)
Removes all occurrences of white space characters from the beginning and end of the string specified in the first parameter. The method is overloaded, while the *String charsToTrim* parameter is optional.
String text: Text to be trimmed.
String charsToTrim: String divided to individual characters which are then trimmed.

String TrimEnd (String text, *String charsToTrim*)
Removes all occurrences of a set of characters specified in an array from the end of the string specified in the first parameter. The method is overloaded, while the *String charsToTrim* parameter is optional.
String text: Text to be trimmed.
String charsToTrim: String divided to individual characters which are then trimmed.

String TrimStart (String text, *String charsToTrim*)
Removes all occurrences of a set of characters specified in an array from the beginning of the string specified in the first parameter. The method is overloaded, while the *String charsToTrim* parameter is optional.
String text: Text to be trimmed.
String charsToTrim: String divided to individual characters which are then trimmed.

String Replace (String text, String replace, String replacement)
Replaces all occurrences of a specified Unicode character or string in the string specified in the first parameter with another specified Unicode character or string specified in the third paramter.
String text: Base text.
String replace: Text to be replaced.
String replacement: Replacement text.

String Remove (String text, Int32 position, Int32 length)
Deletes a specified number of characters from the string specified in the first parameter, beginning at the specified position.
String text: Base text.
Int32 position: The position in the base text where to begin deleting characters.
Int32 length: The number of characters to delete.

String PadLeft (String text, Int32 length, *String paddingString*)
Left-aligns the characters in the string, padding on the right with a specified Unicode character, for a specified total length. This method is overloaded, while the *String paddingString* parameter is optional.
String text: Base text.
Int32 length: The number of characters in the resulting string, equal to the number of original characters plus any additional padding characters.
String paddingString: A Unicode padding character (if not specified, space is used).

String PadRight (String text, Int32 length, String paddingString)
Right-aligns the characters in this string, padding on the left with a specified Unicode character, for a

specified total length. This method is overloaded, while the *String paddingString* parameter is optional.
String text: Base text.
Int32 length: The number of characters in the resulting string, equal to the number of original characters plus any additional padding characters.
String paddingString: A Unicode padding character (if not specified, space is used).

Int32 IndexOf (String text, Object searchFor)
Reports the index of the first occurrence of a string within this instance.
String text: Base text.
Object searchFor: The string to seek.

Int32 LastIndexOf (String text, Object searchFor)
Reports the index position of the last occurrence of a specified Unicode character or string within the string specified in the first parameter.
String text: Base text.
Object searchFor: The string to seek.

String Format (Object formattedObject, String format)
Formats given value to requested format.
Object formattedObject: Object to format.
String format: Formatting string.

String LoremIpsum (*Int32 length*)
Generates Lorem Ipsum text of given length. This method is overloaded, while the *Int32 length* parameter is optional. If the parameter is not used, the returned text is 1002 characters long (including white spaces).
Int32 length: Length of the text.

Transformation methods

String ApplyTransformation (IEnumerable collection, String transformationName)
Applies the specified transformation to list of items.
IEnumerable collection: Collection of items.
String transformationName: Transformation name.

String Transform (IEnumerable parameters, String transformationText)
Applies ad-hoc text transformation to the provided list of items.
IEnumerable parameters: Collection of items.
String transformationText: Text of the transformation.

For more information and practical examples demonstrating how the *ApplyTransformation* method can be used, please refer to the Transformations in macro expressions topic.

In addition to the methods mentioned above, you can also use all methods usable in transformation text. The methods are listed and described in Context Help -> General -> Methods in transformations. These methods are only offered under the **Transformation** namespace by macro autocompletion (type e.g. *Transformation.Eval(DocumentName)*) when entered elsewhere than in transformation text. However, if you enter them manually without the namespace, they will be functional as well.

Other methods

void LogToDebug (params object[] valuesToLog)

Logs values of provided parameters into the macro debug in *Site Manager -> Administration -> System -> Debug -> Macros*.
params object[] valuesToLog: Array of parameters whose values will be logged into the macro debug.

7.12.5 Macro parameters

It is possible to create macros with parameters to get better or specific functionality, especially for data (context) macros. Each parameter of a macro is separated with the "|" character located after the macro expression. You can use multiple macro parameters.

Examples: *{%SKUPrice**|(culture)en-us**%}*, *{%SKUPrice**|(culture)en-us|(format){0:f1}**%}*

The "|" character can be escaped using the "\|" sequence, e.g. *{%SKUPrice|(default)N**|*A%}* will display "N|A", i.e. the "|A" sequence will not be interpreted as a new parameter.

Please note

The same results that can be achieved using macro parameters can be achieved through macro methods. Macro parameters are still functional due to backward compatibility with previous versions of Kentico CMS, but they are now obsolete and using macro methods is recommended instead.

Currently available parameters are:

- **|(culture)<code>** - specifies culture that should be used for the macro result.
- **|(format)<format>** – formats the macro result according to the provided formatting string.
- **|(default)<value>** or **|<value>** - saying what should be returned when the macro returns an empty value.
- **|(encode)<true/false>** – processes the macro result result with *HTMLHelper.HTMLEncode*.
- **|(urlencode)<true/false>** – processes the result with *HttpUtility.UrlEncode*.
- **|(tolower)<true>** – converts the macro result to lowercase.
- **|(toupper)<true>** – converts the macro result to uppercase.
- **|(toint)<default value>** – converts the macro result to integer, if not successful, uses the default value.
- **|(tobool)<default value>** – converts the macro result to Boolean.
- **|(toguid)<default value>** – converts the macro result to GUID.
- **|(todouble)<default value>** – converts the macro result to Double.
- **|(todatetime)<default value>** – converts the macro result to DateTime.

- **|(resolvebbcode)<true/false>** – resolves the BB code in the result of the macro.
- **|(equals)<value>** – returns *true* if the resolved value matches the given value, otherwise returns *false*.
- **|(notequals)<value>** - returns *false* if the resolved value matches the given value, otherwise returns *true*.
- **|(truevalue)<value>** – output settings for the positive output of comparison.
- **|(falsevalue)<value>** – output settings for the negative output of comparison.
- **|(add)<number>** - adds the provided number to the macro result.
- **|(multiply)<number>** - multiplies the macro result by the specified number.
- **|(divide)<number>** - divides the macro result by the specified number.
- **|(sin)** - returns sinus of the macro result.
- **|(cos)** - returns cosinus of the macro result.
- **|(tan)** - returns tangens of the macro result.
- **|(sqrt)** - returns square root of the macro result.
- **|(pow)<number>** - returns the <*number*>-th power of the macro result.
- **|(startswith)<string>** - returns *true* if the macro result starts with the specified string.
- **|(endswith)<string>** - returns *true* if the macro result ends with the specified string.
- **|(contains)<string>** - returns *true* if the macro result contains the specified string.
- **|(not)** - returns logical negation of the macro result.
- **|(append)<string>** - appends the specified string to the macro result.
- **|(prepend)<string>** - prepends the specified string to the macro result.
- **|(trim)<chars>** - removes all characters contained in the <*chars*> string from the beginning and end of the macro result if they are present there.
- **|(trimend)<chars>** - removes all characters contained in the <*chars*> string from the end of the macro result if they are present there.
- **|(trimstart)<chars>** - removes all characters contained in the <*chars*> string from the beginning of the macro result if they are present there.
- **|(padleft)<totalwidth>(with)<char>** - returns the macro result with the beginning of the string padded with the <*char*> character to the total length of <*totalwidth*>.

- **|(padright)<totalwidth>(with)<char>** - returns the macro result with its end padded with the <*char*> character to the total length of <*totalwidth*>.
- **|(substring)<start>;<length>** - returns a <*length*>-long substring of the macro result beginning on the <*start*> index.
- **|(replace)<src>(with)<dest>** - replaces all occurrences of the <*scr*> string in the macro result with the <*dest*> string.
- **|(matches)<regex>** - returns true if the macro result matches the provided regular expression.
- **|(getmatch)<regex>** - returns match of the provided regular expression from the macro result.
- **|(regexreplace)<regex>(with)<dest>** - replaces matches of the provided regular expression in the macro result with the <*dest*> string.
- **|(striptags)** - strips the macro result of HTML tags.
- **|(limitlength)<length>** - limits length of the macro result to the specified number of characters.
- **|(resolveurl)** - resolves the URL returned by the macro expression.
- **|(unresolveurl)** - unresolves the URL returned by the macro expression.
- **|(mappath)** - maps the virtual path returned by the macro expression to a disk path.
- **|(lowerthan)<number>** - returns *true* if the macro result is lower than the provided number.
- **|(greaterthan)<number>** - returns *true* if the macro result is greater than the provided number.
- **|(jsescape)** - escapes the macro result for usage in JavaScript to avoid XSS.
- **|(sqlescape)** - escapes the macro result for usage in SQL to avoid SQL injection (single quote characters are replaced by two single quotes).
- **|(resolve)** - resolves macros again in the result of the macro expression.
- **|(casesensitive)** - by default, string comparison is performed as case insensitive, unless it is switched to case sensitive by adding the *<add key="CMSMacrosCaseSensitiveComparison" value="true">* key to the *appSettings* section of the *web.config* file. You can use this parameter to switch string compariton to case sensitive. You can also use the **|(casesensitive)false** and **|(casesensitive)true** notation to turn case sensitive string comparison on or off when the opposite default way of comparison is configured.
- **|(debug)** - indicates that detailed debugging should be used for the given macro expression. Detailed debugging may also be enabled globally for all macros through the *Site Manager -> Settings -> System -> Debug -> Enable detailed macro debug* setting. Please see the Development -> Debugging and system information chapter for additional information.

Disabling resolving of macro parameters

It is possible to disable resolving of macro parameters. This can be done by adding the following key to

the appSettings section of your project's *web.config* file:

```
<add key="CMSDisableMacroParameters" value="true" />
```

With this key added, macro parameters will be ignored, so that e.g. **{%"HELLO"|(tolower)%}** will be resolved as *HELLO* and not as *hello*.

7.12.6 Entering macro expressions

This topic provides a summary of features that facilitate entering of macro expressions in various parts of the system. These features include:

- Macro autocompletion
- Macro selection control
- Insert macro WYSIWYG editor dialog
- Edit value dialog in web part properties
- Macro condition editor

Click the feature name in the list above learn more about it.

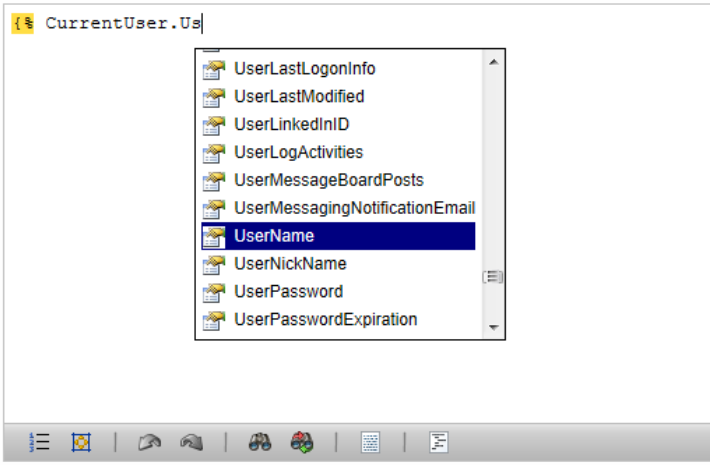
Macro autocompletion

Automatic completion of macro expressions is available when writing macro expressions in all dedicated macro editors, as well as the following places:

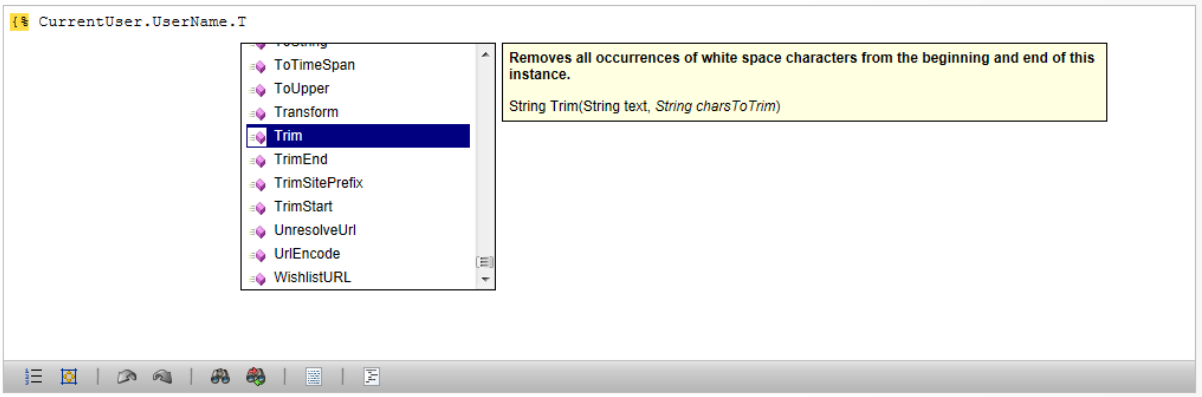
- **E-mail templates**
- **Macro-based transformations**

This feature is similar to IntelliSense in Visual Studio — as you type, a box with methods or properties that begin with the letters you wrote is displayed above or below the cursor. Only those methods and properties that are relevant in the current context are displayed in the box. The methods and properties are listed in alphabetical order and you can navigate through them using the up and down arrows or the mouse. Once you select the appropriate one, press the Enter, space or dot key to insert it into the text.

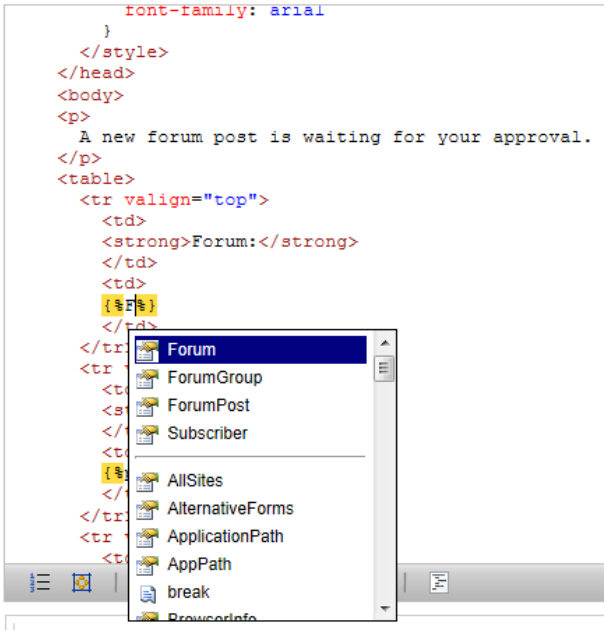
The box with available methods and properties is also displayed when adding further parts of expressions using the dot-suffix notation.



If a macro method is selected from the drop-down list, its description and signature (containing the return type and parameter types) is displayed in a tooltip next to the drop-down list. As some methods are overloaded, i.e. they can accept different numbers of parameters, parameters present in all overloads are displayed in standard letters (as "String text" in the screenshot below), while additional parameters present only in certain overloads are displayed in italics (as *String charsToTrim* in the screenshot below) .



If the macro is being added within a particular type of context, for example an e-mail template belonging to a specific module, the related objects and properties will be prioritized. This means that the most frequently needed items can be accessed at the top of the autocompletion box, separated by a horizontal line.



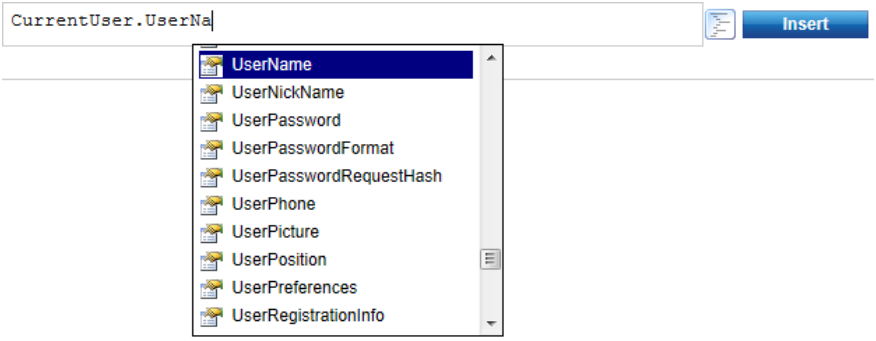
Macro selection control

Another feature that makes entering of macro expressions easier is the macro selection control.

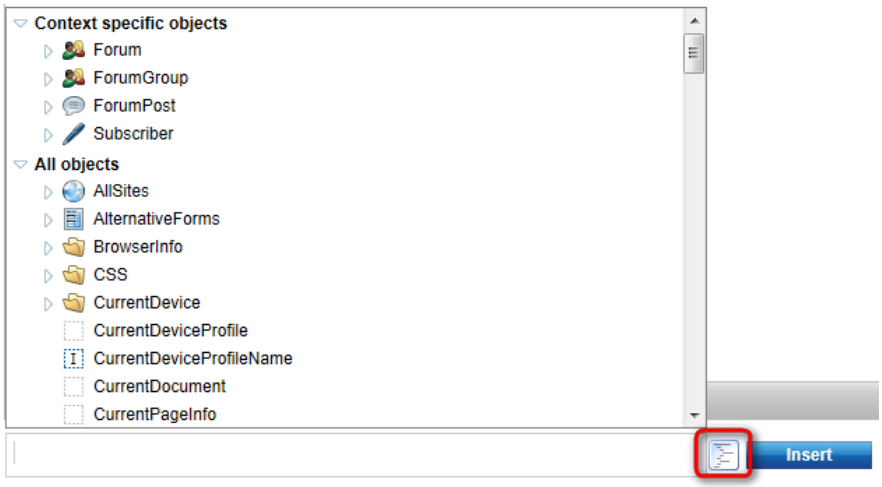
It is present in the following locations:

- **E-commerce invoice templates**
- **E-mail templates**
- **Web part properties**

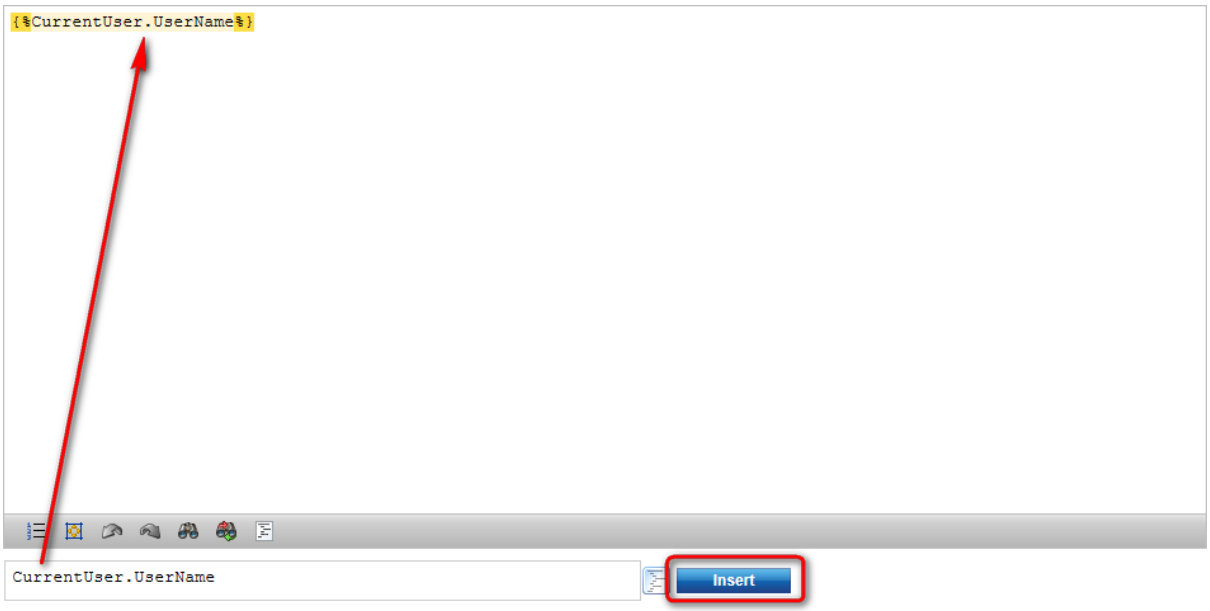
The control can be used in two different ways. The first one is simply typing the macro text into the text box, where macro autocompletion is available.



The other option is to select the required macro by clicking the **Show/hide macro object tree** ( ) button. After doing so, an object tree is opened above the text box, letting you choose objects or their properties from the current context. Like with the autocompletion feature, certain objects may be prioritized and offered in a separate tree at the top (**Context specific objects**). Clicking an object or its property automatically enters the corresponding macro expression into the text box.

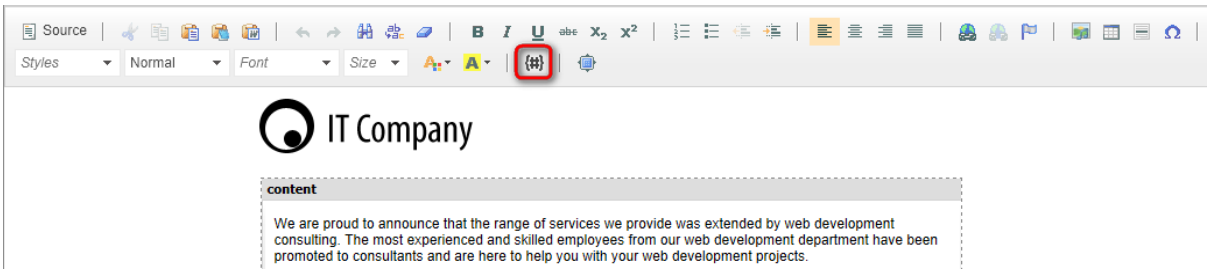


When you have the required expression in the text box, click the **Insert** button to paste it into the current position in the edited text. The expression will be pasted, enclosed within the {%%} data macro parentheses.

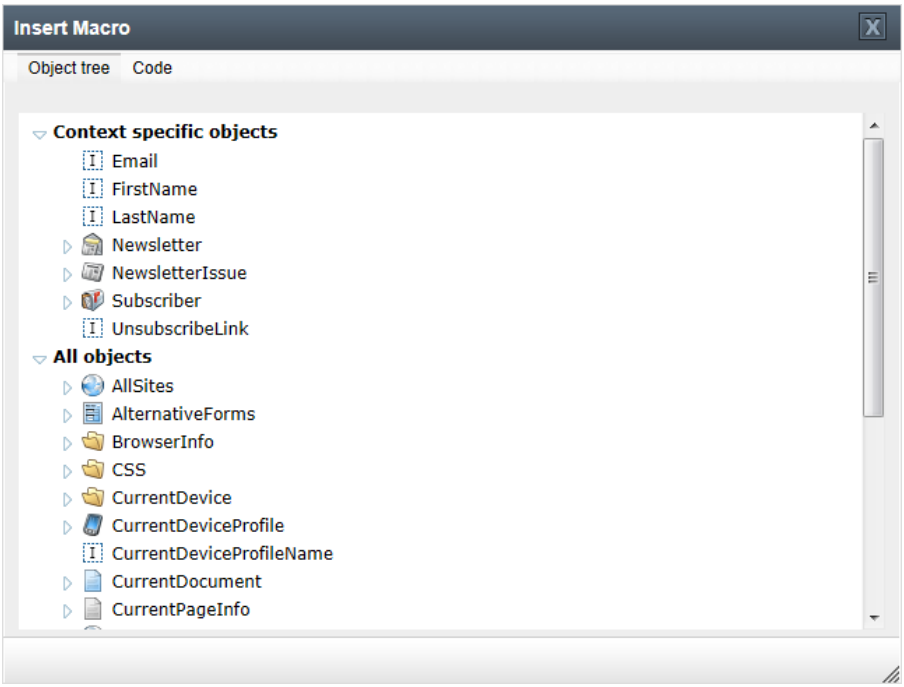


Insert macro WYSIWYG editor dialog

Macros may also be easily added into any content areas where the WYSIWYG editor is provided, such as when editing documents in CMS Desk on the *Page* tab or writing the content of newsletters. In this case, macro functionality can be accessed by clicking the **Insert macro** ({#}) button on the editor's toolbar.



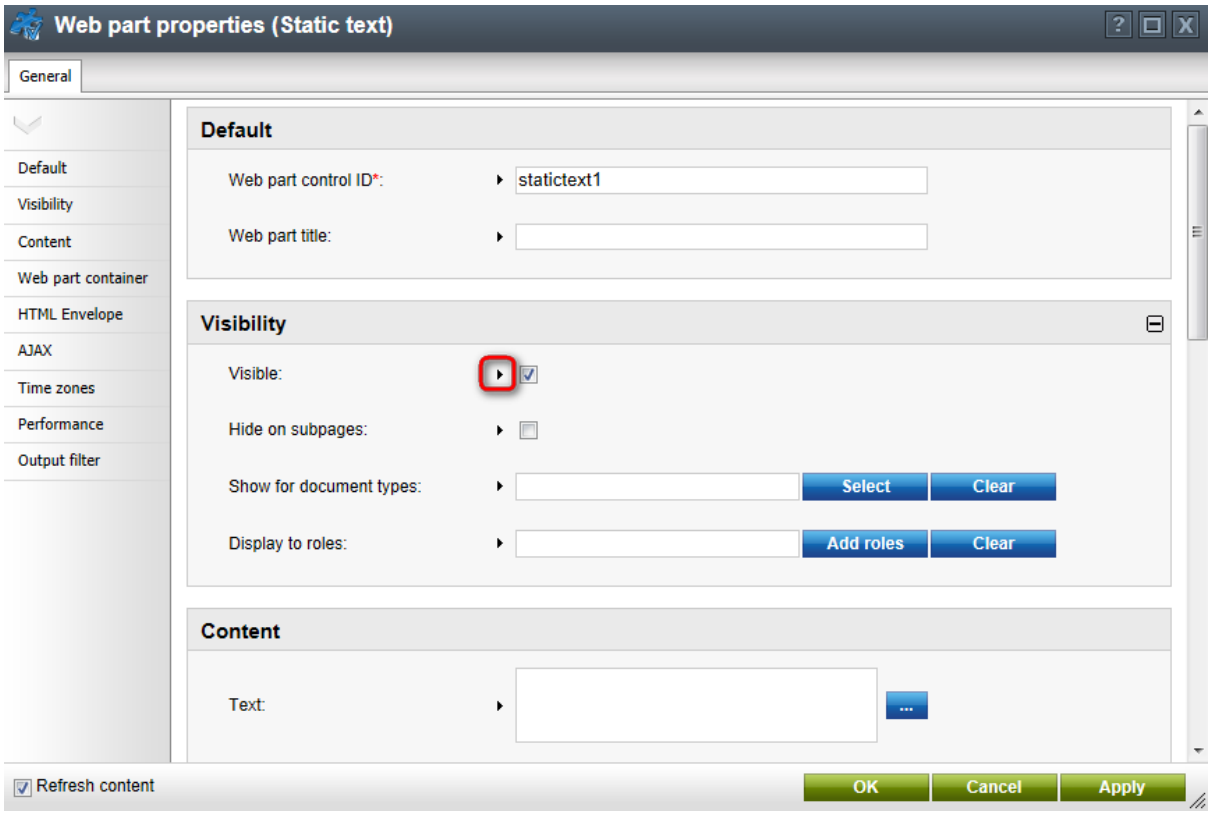
This opens a dialog with an object tree that works the same way as described above for the Macro selection control. Clicking on an item in the tree inserts the appropriate macro at the current position of the cursor in the edited area.



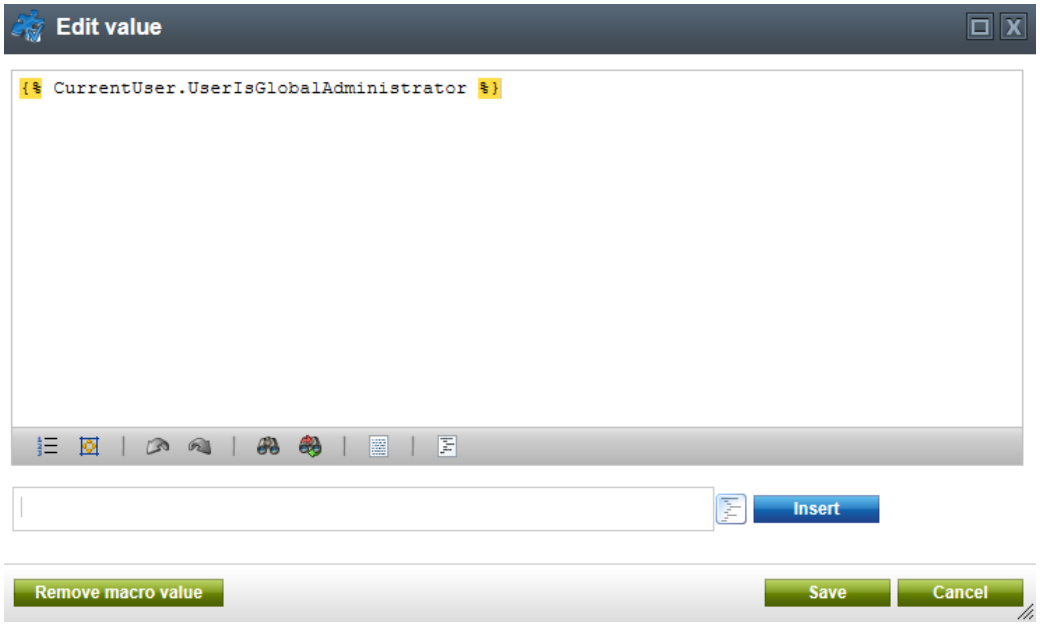
If you wish to write the macro expression directly (with autocompletion support), you can do so by switching to the **Code** tab of the dialog and clicking the **Insert macro** button when done.

Edit value dialog in web part properties

Macro expressions can be used in the values of web part properties. You may enter any required macros directly into the content of properties that allow you to specify text values. Additionally, all properties provide the ▸ button highlighted in the screenshot below.



If clicked, the **Edit value** dialog pops up and lets you enter a macro that will determine the value of the property dynamically. In the dialog, you can use both macro autocompletion and the macro selection control described above. This option allows you to use macros even for properties that do not otherwise allow text input, such as checkboxes or lists of options.



For example, the *{% CurrentUser.UserIsGlobalAdministrator %}* macro shown above returns a true or

false value depending on if the current user is a global administrator. When added into the **Visible** property, this ensures that the given web part is only displayed to global administrators.

SQL injection protection in web part properties

Some web part properties are secured against SQL injection attacks, which may affect how macros are resolved in specific cases. By default, this is applied to macros entered into the **WhereCondition** and **OrderBy** web part properties.

If the macro returns a string value that contains single quote characters ('), they will be escaped and replaced by two single quotes (''). This may cause an SQL syntax error if you are using the macro to dynamically insert a part of a query, such as a WHERE clause.

It is possible to disable single quote escaping for a specific macro expression by adding the **handlesqlinjection** parameter and setting its value to *false*:

{% ... |(handlesqlinjection)false %}

To disable SQL escaping globally for all properties of a specific type of web part, edit its code behind file (e.g. *~/CMSWebParts/Viewers/Documents/cmsrepeater.ascx.cs* for the **Repeater** web part) and add the following line of code into the **SetupControl()** method:

[C#]

```
this.SQLProperties = "";
```

The **SQLProperties** property is inherited from the **CMSAbstractWebPart** base class by all web parts, but you can override its value to set which properties should be protected.

If you wish to enable SQL escaping for additional web part properties, you can enter their names into the value separated by semicolons, for example:

```
this.SQLProperties = "wherecondition;orderby;sqlquery";
```

Please note that disabling SQL protection may create a **security vulnerability** if the macro resolves its value according to data that can be modified by the website's users, such as in the case of QueryString macros.

7.12.7 Macro conditions

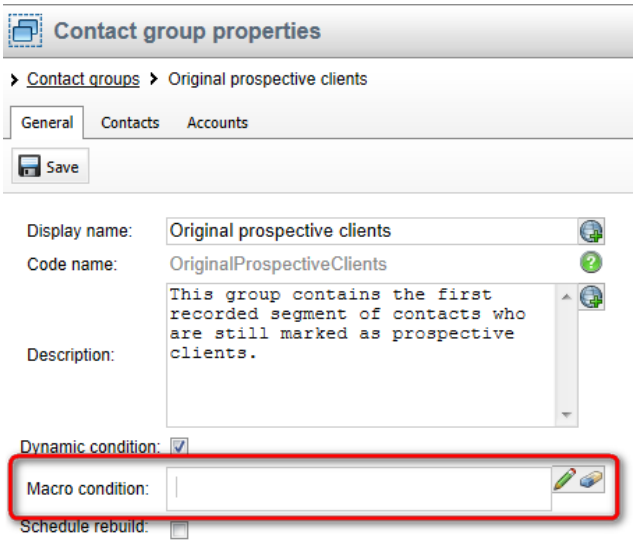
Macro conditions add a large degree of flexibility to various types of functionality in the system. Each condition is defined by a macro expression that returns a boolean value (true or false). Whenever the condition is evaluated, the specified macro is resolved according to the currently available context data and its return value determines whether the condition is fulfilled or not. For example, such conditions

may be used to dynamically set when a certain action should be performed, under which circumstances an item should be visible and so on.

Conditions are available for the following types of objects:

- **Content personalization variants**
- **Dynamic contact groups**
- **Web analytics campaigns**
- **Scheduled tasks**
- **Report subscriptions**
- **Fields and categories in the field editor**
- **Workflow scopes and advanced workflow steps**
- **Automation process triggers**

The interface used to enter these conditions is provided by a special *Macro condition editor*.



It is not necessary to enclose the expression between the standard macro parentheses — the entire content of the condition is automatically processed as a context macro. You can also leverage Macro autocompletion support when typing in condition fields. Clicking the **Clear condition** ( ) action deletes the current condition.

The editor itself can be accessed in a separate dialog, opened by clicking the edit icon ( ) next to the respective condition field. There are several different approaches that may be used to define conditions, as described in the sections below.

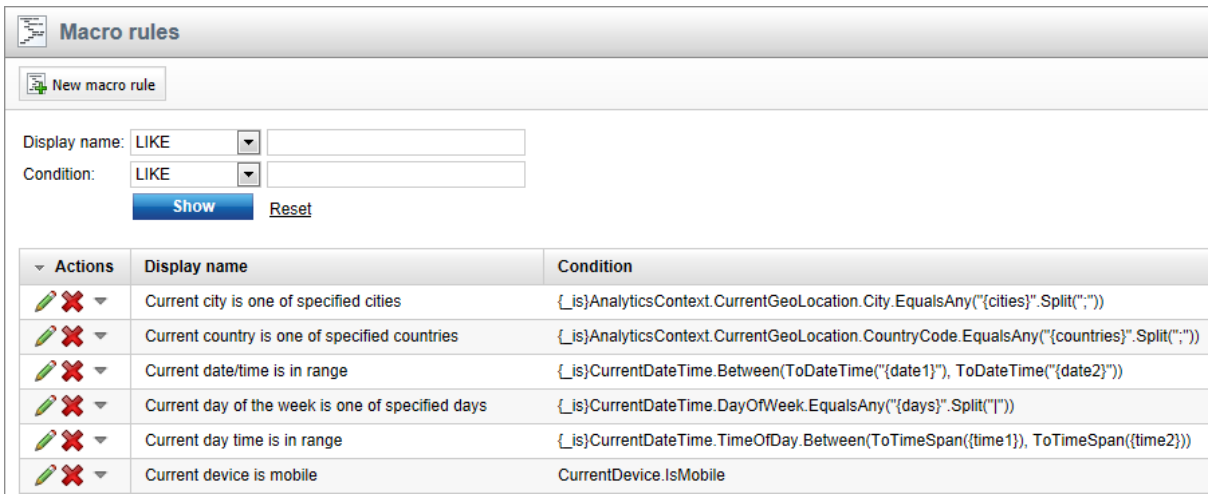
Macro rules

In some cases, macro conditions may need to be created by non-technical users without any knowledge of K# macro syntax. For this reason, the condition editor provides a way to build conditions simply by adding and configuring predefined components called *Macro rules*. Each macro rule is implemented by a standard macro expression, which is wrapped into a user friendly, text-based interface.

Macro rules need to be prepared in advance before they may be used in conditions. Many different types of rules are included by default, but you can also define any additional ones that your users might need.

There are several categories of macro rules, which can be managed in different sections of the administration interface. The rules from each category are offered when designing the corresponding types of conditions, as described in the following table:

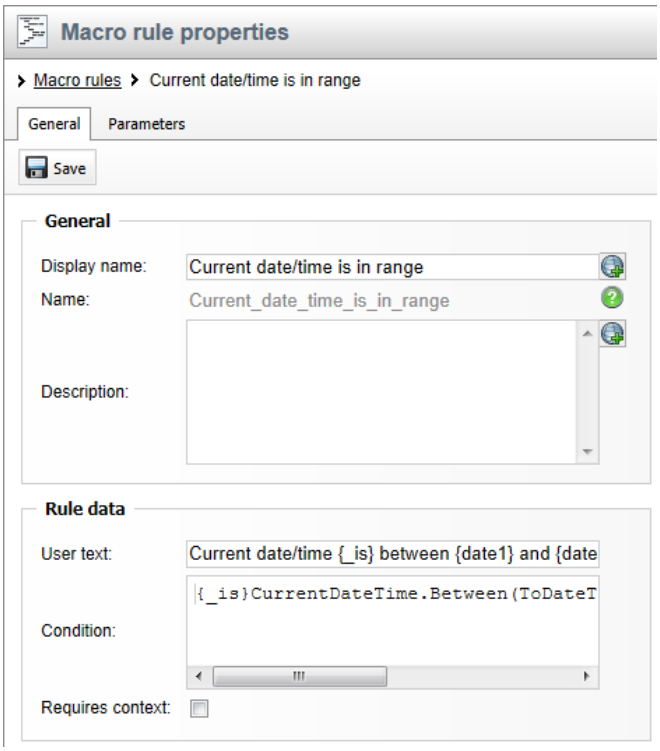
Rule category	Description
Global	Global rules may be used for all conditions in any part of the system. They can be defined in **Site Manager -> Development -> Macro rules**.
Workflow	Offered when adding conditions to Workflow scopes or advanced workflow steps. Workflow rules may be managed in **Site Manager -> Development -> Workflows -> Macro rules**.
On-line marketing	On-line marketing rules can be selected when building conditions for the following items: • Content personalization variants • Dynamic contact groups • Campaigns They may be defined in **CMS Desk -> On-line marketing -> Configuration -> Macro rules**.
Reporting	These rules are available when specifying conditions for Report subscriptions. To add reporting macro rules, go to **CMS Desk -> Tools -> Reporting**, select any report category in the tree and choose the **Macro rules** tab.



When creating a new macro rule or editing ( ) an existing one on the **General** tab, the following options can be specified:

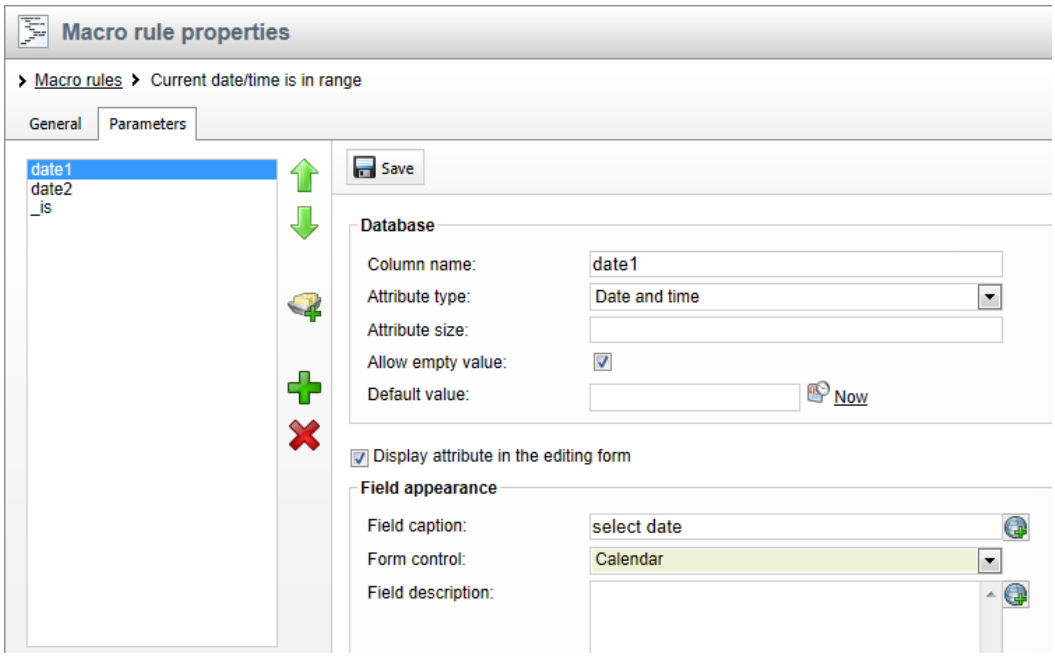
- **Display name** - sets the name of the macro rule displayed to users. This name is shown to users in the list of available macro rules, so it should also serve as a basic description.
- **Name** - sets the name used as an identifier for the rule (for example in the API).
- **Description** - may be used to enter a description for the rule. This text is displayed to users as a tooltip when they hover over the rule in the list of available options.

- **User text** - defines the text clause displayed when the rule is inserted into a macro condition. It should accurately describe the requirements that must be fulfilled in order for the rule to be true.
- **Condition** - this is where the actual condition represented by the rule needs to be defined using standard context macro code (K#). The entered macro expression is resolved whenever the rule is evaluated, and its return value determines whether the rule is true or false. Autocompletion support is provided when writing in this field.
- **Requires context** - this property should be enabled if the rule's condition needs to access context data (information about the current user, the currently viewed page etc.) in order to work correctly. Doing so ensures that the rule will only be offered for conditions that have the context available when they are resolved. For example, the context is accessible when evaluating the conditions of content personalization variants on the website's page, but not when the system is building dynamic contact groups.



The functionality of a macro rule can be modified through parameters. After a user adds the rule to a condition, they can configure its exact behavior by setting the values of its parameters. This makes rules more flexible and allows them to be used for various different scenarios. For example, in a macro rule that requires the current user to be a member of a role, the exact role could be determined by a parameter.

Parameters can be created by editing the rule on its **Parameters** tab. Each parameter is defined as a field with a certain data type and various other settings. The interface shown to users when editing the parameter's value is provided by a selected Form control.



Once the required parameter fields are prepared, it is necessary to add them into the rule's text clause and condition code back on the **General** tab. To place a parameter into the **User text**, enter its *Column name* enclosed in curly brackets, for example *{role}*. Also use the same format to insert parameters into the appropriate position in the rule's **Condition** code. When the condition is being resolved, the parameter expressions are replaced by the values specified by users for the given occurrence of the rule.

There are several predefined parameters that may be used to add commonly required functionality to macro rules:

- **{_is}**, **{_has}**, **{was}**, **{_will}**, **{_perfectum}** - these provide an easy way to negate a rule's condition. Several variants with different wording are available, so you can use the one that matches the text of the particular rule. They allow users to choose between a positive and negative option, e.g. *is* or *is not*. If the negative option is selected, the parameter is resolved into the K# negation operator "!" in the condition code. With the positive option, it returns an empty string.
- **{_any}** - useful for rules where a list of items needs to be specified through another parameter. It allows users to switch between two options that determine how the item list will be processed — *any* (at least one of the items must meet the given condition) or *all* (the condition must be fulfilled for all items in the list). When resolved, the parameter returns either a *false* (any) or *true* (all) value. In the condition code, the parameter can be inserted as an additional argument of macro methods that work with object lists, which automatically ensures the required functionality. For example: *CurrentUser.IsInRole("{roles}", {_any})*

When one of these parameters is entered into the **User text** field of a rule and saved, the corresponding definition with the appropriate form control and default values is automatically created on the **Parameters** tab of the given rule.

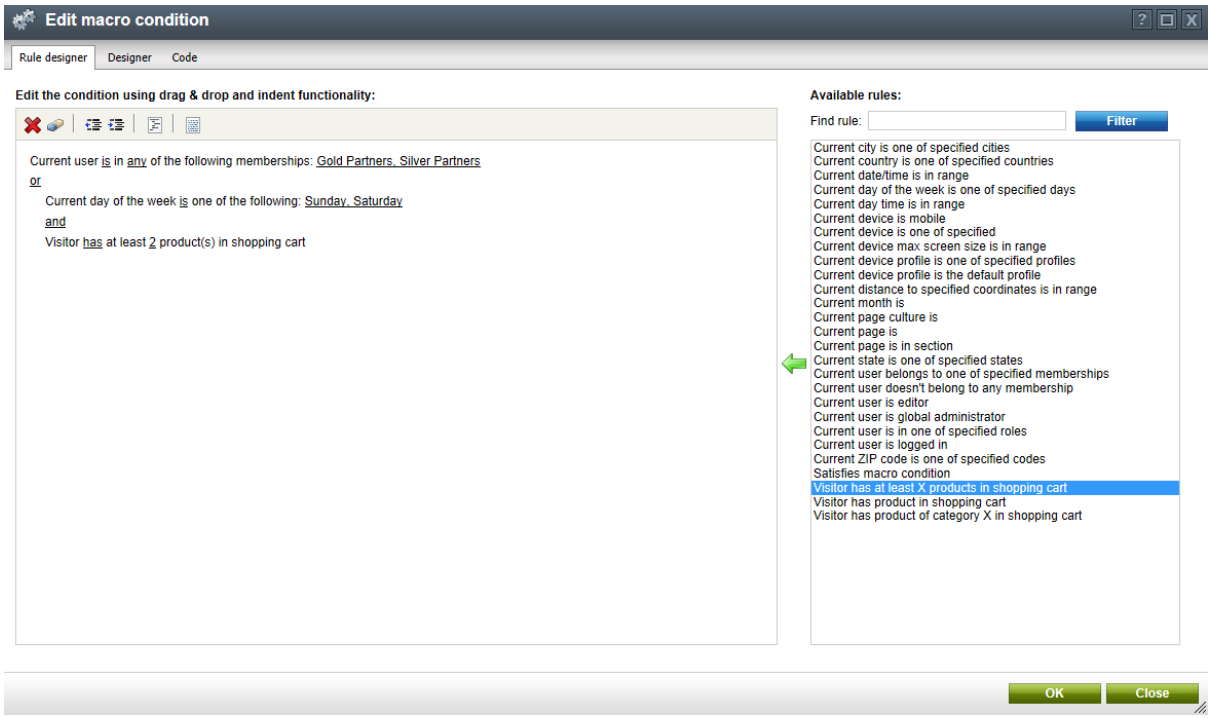
Adding macro rules to conditions

When creating a condition via the macro condition editor, you can work with macro rules on the **Rule designer** tab.

All rules prepared for the type of condition that is being edited are listed under the **Available rules** section. To add a rule to the condition, select it in the list and use the **Add rule** () button. This inserts the given rule clause into the main designer area, where it is displayed as a text description of the corresponding requirements.

If multiple rules clauses are present in the condition, logical operators are automatically added between them. The **and** operator means that the combination of two rules will only be true if both of them are also true. If **or** is placed between two rules, the result will be true if at least one of them is fulfilled. You can switch between these two options by clicking on individual operators. The final result of the condition is calculated based on the used operators.

Any parameters in the text of the added rules are underlined. Clicking on a parameter opens a separate dialog where the user can set its value.



You can select a rule clause in the designer simply by clicking on it (outside of any underlined parameter text). The rules may be managed through the actions available in the header of the designer area:

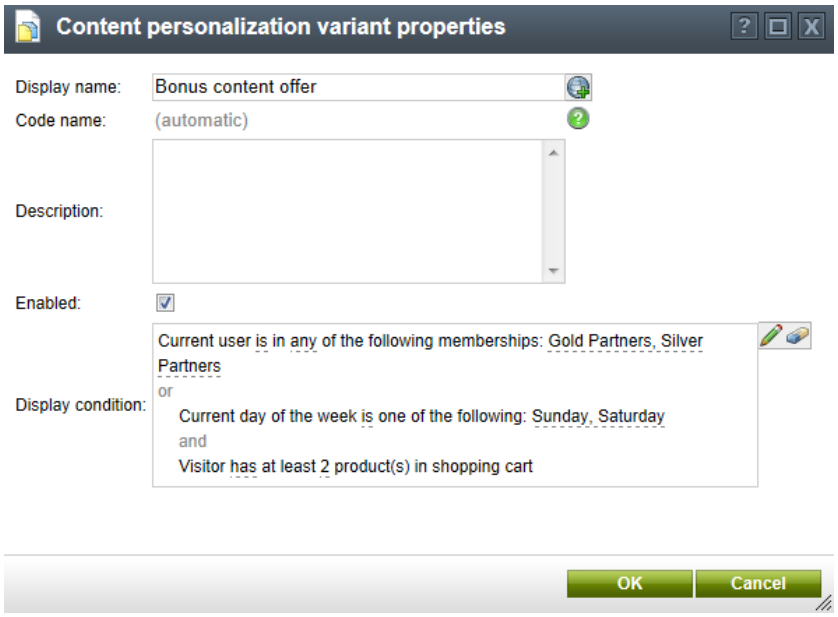
- **Delete** - removes the currently selected macro rule clause.
- **Clear all rules** - removes all rules from the condition.

- **Unindent** - this action decreases the indentation of the selected macro rule. Indentation in the rule designer provides a way to group together macro rules and set the precedence of the logical operators between them. The rules on the level with the largest indentation are evaluated first. The result is then passed to the level above, and this continues until the final result of the condition is known.
- **Indent** - increases the indentation level of the selected macro rule.
- **Auto indent** - changes the indentation of all rules in the condition according to standard operator priorities.

- **View condition in K#** - allows you to view the K# code of the overall condition defined through the macro rules.

It is also possible to change the order of the rules by dragging them to a different position in the designer area.

Once all required rules are added, you can insert the result into the edited condition field by clicking **OK**.



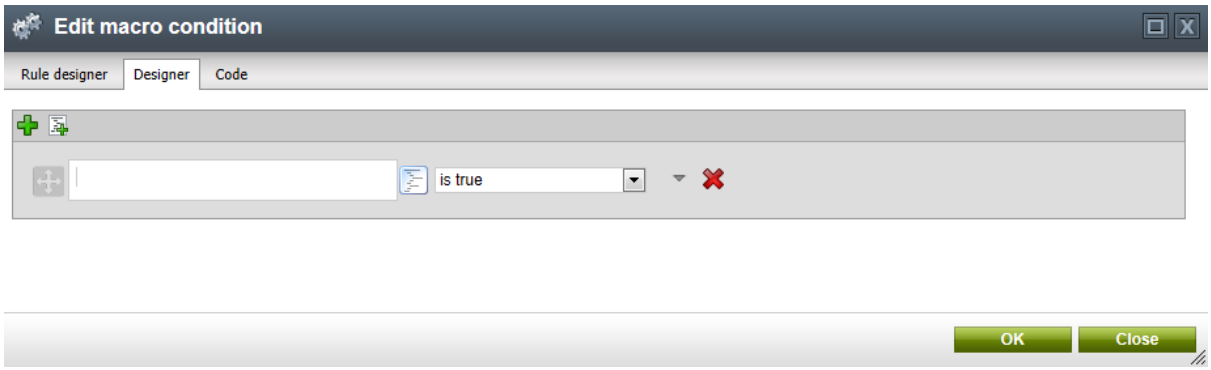
Designing macro conditions directly

In addition to the Rule designer, the macro condition editor also offers other ways to create conditions that are directly based on writing K# code. This approach offers much more flexibility and does not need any predefined components. However, it requires users to have at least basic understanding of the macro syntax and programming principles.

The **Designer** mode of the editor provides a graphical interface that makes it easier to build complex conditions. It allows you to divide the condition into individual expressions and groups of expressions, which can then be managed as separate elements. The overall condition is defined by combining these sub-elements as necessary.

You can add elements into the designer using the actions in the headers of groups or the main area:

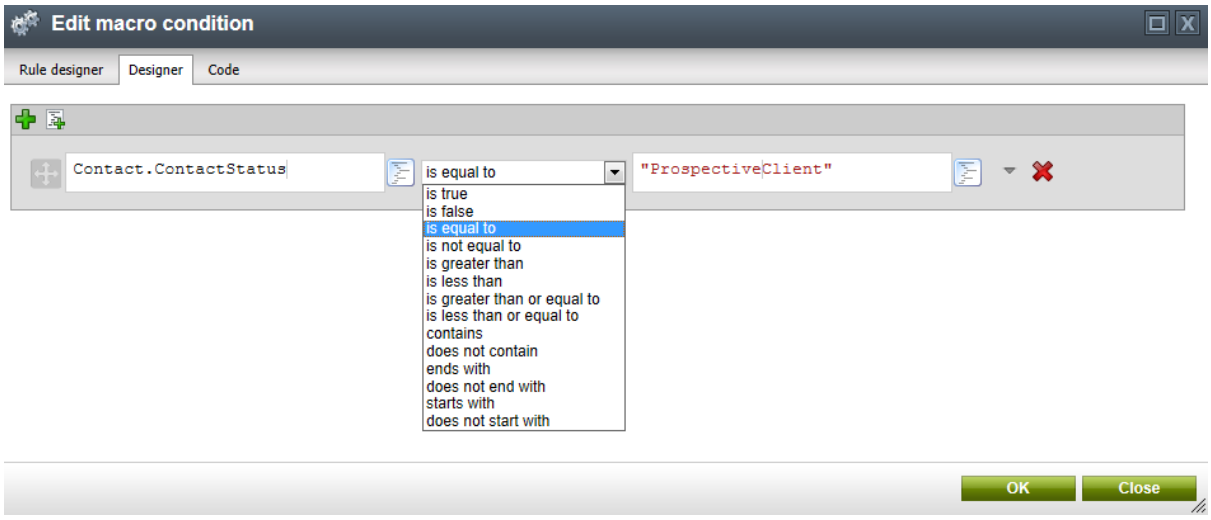
- **Add expression** - adds a new macro expression into the group.
- **Add group** - adds a new sub-group under the given group.



Expressions are composed of two macro selection controls with a relation drop-down list between them. You can either input context macro code or static values into these fields. Macro autocompletion and the **Show/hide macro object tree** ( ) button are available for easier macro specification.

Each expression must always return a boolean result (true or false). Through the relation drop-down list, you can specify the operator that will be used to evaluate the values on the left and right of the expression. The following operator options are available:

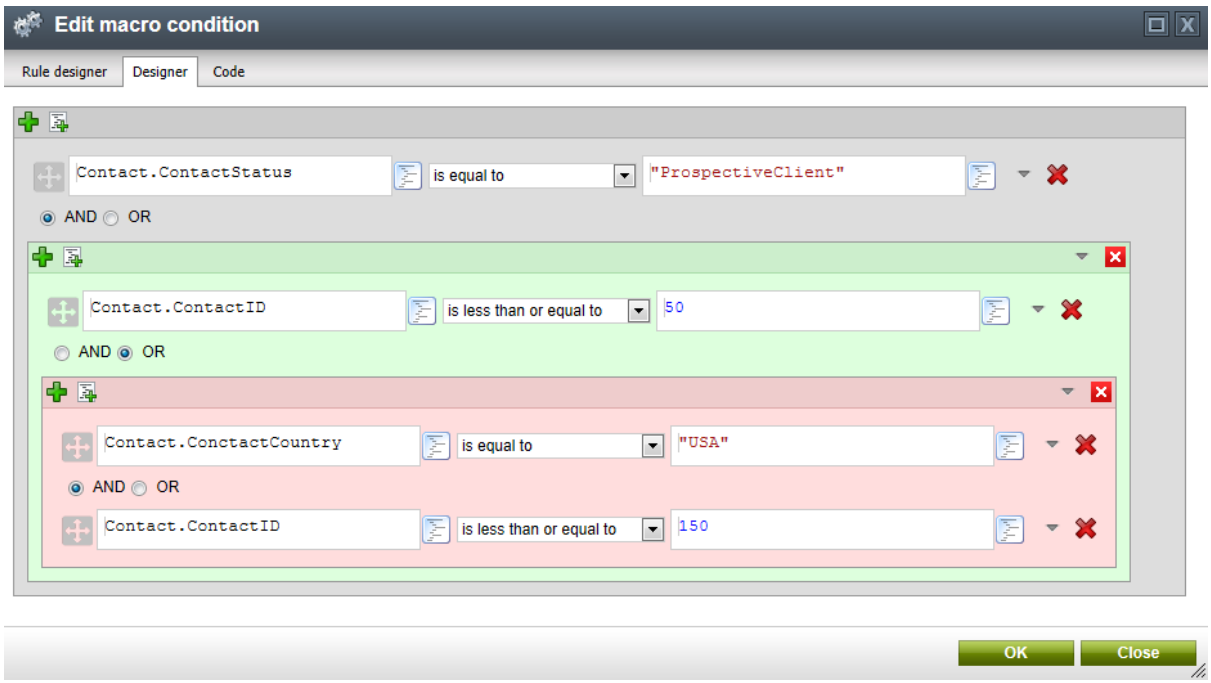
- **is true**, **is false** - these options may be used to evaluate macro expressions that already return a boolean value. Only one macro field is displayed in this case, as there is no need to compare two values.
- **is equal to (==)**, **is not equal to (!=)** - determine whether the values returned by the expressions on the left and right are equal.
- **greater than (>)**, **less than (<)**, **greater than or equal (>=)**, **lesser than or equal (<=)** - allow you to compare numeric values returned by the expressions on the left and right.
- **contains**, **ends with**, **starts with** (and **does not** equivalents) - these operators allow you to compare text-based (string) values in various ways.



By adding groups, you can organize expressions and set precedence. Groups in the macro designer are the equivalent of parentheses. Each group of expressions is evaluated separately before being combined with the remaining elements.

Expressions and groups are added together using logical conjunction (**AND**) or logical disjunction (**OR**). You can choose which of these you wish to use by selecting the corresponding radio button between

any two elements. The final result of the condition is determined by the total combined value of all expressions and groups.



Existing expressions or groups can be removed from the condition using the following action icons:

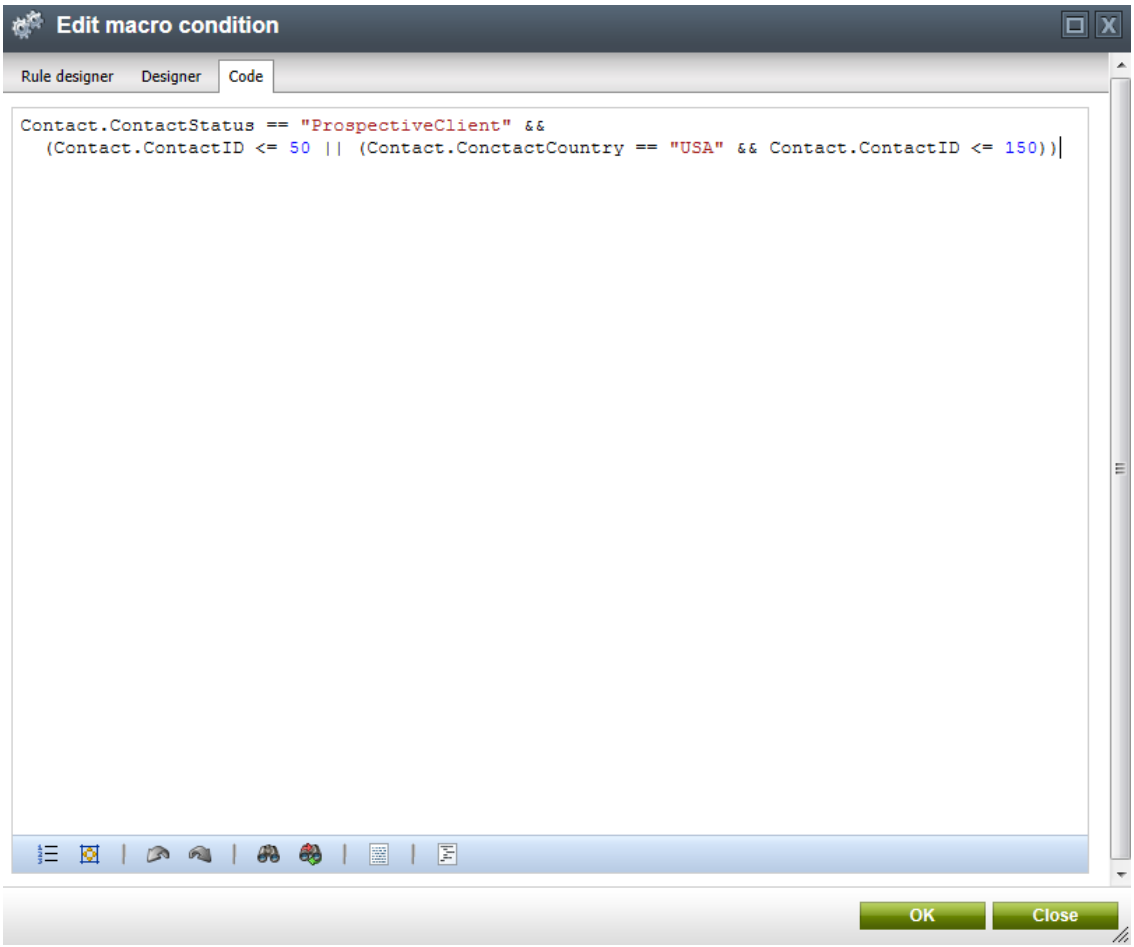
- **Remove group** - removes the whole group, including all expressions and sub-groups that it contains.
- **Remove expression** - removes the given macro expression.

You can also change the order and position of elements using the actions available in the context menus that can be opened by clicking the respective icons:

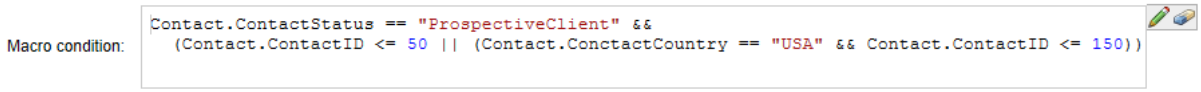
- **Move up** - moves the macro expression or group up before the one above it.
- **Move down** - moves the macro expression or group down after the one below it.
- **Move to parent** - moves the macro expression or group from its current group to the parent group.

Items can also be moved simply by dragging them into the desired location. Groups need to be picked up by their header row, expressions need to be dragged using the cross icon.

If necessary, the overall macro code of the defined condition can be reviewed or directly modified by switching to the **Code** tab.



Finally, when you finish specifying the condition on either the **Designer** or **Code** tab, click **OK** to insert it. In this case, the resulting macro code will be displayed in the condition field.



Incompatibility with the rule designer

Please note that the system is not capable of converting general macro code to macro rules. Because of this, making any changes to the condition on the **Designer** or **Code** tab deletes the content of the **Rule designer**.

7.12.8 Macro security

Whenever a user saves a macro expression, the system automatically adds a security signature. The signature contains the user name of the macro's author and a hash of the given expression. To improve security, the hash function used when creating macro signatures appends a salt to the input (a sequence of additional characters). The salt value depends on the configuration of the application's environment, so the signatures are in most cases not valid when deploying macros to other instances of

Kentico CMS.

You can recognize signed macro expressions by the # character, which the system automatically inserts before the closing **%}** parentheses when saving the text containing the macro.

```
{% CurrentUser.Children["cms_category"][0].CategoryName #%}
```

If the execution of the macro requires any permissions, the system resolves the macro only if the **user specified by the signature** has the appropriate permissions.

Permissions are only checked when resolving macro expressions that:

- access an *InfoObject* through another *InfoObject* (i.e. access an encapsulated object), for example: *{% CurrentDocument.DocumentPageTemplate %}*
- access an *InfoObjectCollection*, for example: *{% CurrentUser.Children["cms_category"][0] .CategoryName %}*

The system carries out a security check for each access to any collection, not just for the final macro result.

Note

Unsuccessful security checks prevent the system from resolving macros. If you encounter such problems, you can view the event log or the macro debug log, which provide information about performed security checks and their results.

Disabling the security signature for specific macros

If you manually add the @ character before the closing **%}** sequence of a macro, the system does not add a signature.

```
{% CurrentDocument.DocumentName @%}
```

These macros do not store the user name of the author or the security hash, so the internal length of the expression does not exceed the visible number of characters. This allows you to safely use macros in fields with a limited character count.

Unsigned macros are always evaluated with the permissions of a public user.

Configuring the hash salt for macro signatures

The system appends a salt to the input of the hash function that creates macro signatures.

The default salt is the application's main database connection string (the exact **CMSConnectionString** value set in the web.config file). However, ***we strongly recommend using a custom salt***.

A custom salt provides the following benefits:

- Stability (you do not need to re-sign macros if your connection string changes)
- You can use the same salt for other instances of Kentico CMS, which makes deployment easier and allows content staging of macros

The best option is to set the hash salt value before you start creating content for your website. Changing the salt causes all current hash values to become invalid (including macro signatures).

To set a custom hash salt value for your application, add the **CMSHashStringSalt** key to the *appSettings* section of your web.config file. You can use any string as the value, but the salt should be random and at least 16 characters long. For example, a randomly generated GUID is a strong salt:

```
<add key="CMSHashStringSalt" value="e68b9ad6-a461-4707-8e3e-ece73f03dd02" />
```

Warning

In addition to macro signatures, the system uses the **CMSHashStringSalt** value for other hash functions. Changing the hash salt on a website that already has defined content may break dialog links and images on your website.

If you encounter such problems, you need to re-save the given content (the system then creates the hashes using the new salt).

Re-signing existing macros *[Only available after applying hotfix 7.0.9 or newer]*

If the hash salt value used by your application changes, the security signatures of existing macro expressions become invalid. This may lead to problems with unresolved macros, for example when:

- The connection string of your application has changed, e.g., when moving to a different server or after setting a new database password.
- You are using Content staging to transfer data containing macros to an instance of Kentico CMS with a different connection string.
- You have set a new custom salt via the **CMSHashStringSalt** web.config key.

You can repair invalid macro signatures by re-signing all macros using a new hash salt:

1. Go to **Site Manager -> Administration -> System -> Macros**.

2. Fill in the **Old salt** field.

- If you leave the **Sign all macros** option *disabled*, the system attempts to re-sign macros based on the data in the original signatures. As a result, only macros that have a valid signature under the old salt are re-signed and the user names of the macro authors remain unchanged. You need to enter the old salt that was used to generate the security hash of the existing macro expressions in the system.

- By default, the old salt is the value of the application's previous connection string in format: *Persist Security Info=False;database=DBName;server=ServerName;user id=DBUser; password=pwd;Current Language=English;Connection Timeout=240;*
- If your application uses a custom hash salt, enter the original value of the **CMSHashStringSalt** web.config key.

- If you *enable* **Sign all macros**, the macro re-signing process skips the signature integrity check and creates new signatures for all macros. This includes macros that are unsigned or have invalid signatures. The new signatures use the name of the user who started the re-signing procedure. You do not need to enter the old salt value in this case.

3. Type in the **New salt** that will be used to re-sign the macros.
   - By default, the field automatically loads the current application's hash salt value. To enter a different value, disable the **Use current salt** option.

Important!

In order for the system to correctly validate macro signatures, the new hash salt value must match the current salt of the application (the connection string or **CMSHashStringSalt** key value).

Only set a different value than the current salt if you are planning to change the connection string or custom salt.

4. Click **Update macro signatures**.

The system replaces the security signature in all occurrences of macros based on the new salt.

7.12.9 Registering custom macro methods

In addition to the macro methods available by default, you can also define your own custom methods. Users can then call these methods in their macro expressions.

Code samples

The Kentico CMS installation includes code examples of custom macro method registration that you can add directly to your web project. To access these samples:

1. Open your Kentico CMS installation directory (by default *C:\Program Files\KenticoCMS\<version>*).
2. Expand the *CodeSamples\App_Code Samples* sub-directory.
3. Copy the **Samples** folder into the **App_Code** folder of your web project.

Web application installations

If your Kentico CMS project was installed in the web application format, copy the

samples into the **Old_App_Code** folder instead.

You must also manually include the sample class files into the project:

1. Open your application in Visual Studio.
2. Click **Show all files** at the top of the Solution Explorer.
3. Expand the *Old_App_Code* folder, right-click the new **Samples** sub-folder and select **Include in Project**.

You can find the macro method examples in the following classes:

- **Samples/Classes/CustomMacroMethods.cs** - class containing the sample macro methods.
- **Samples/Modules/SampleMacroModule.cs** - initializes the custom method registration.

Creating custom macro methods

1. Open your web project in Visual Studio and add a new class into the **App_Code** folder (or **Old_App_Code** if the project is installed as a web application). For example, name the class **CustomMacroMethods.cs**.

2. Edit the class and add the following references:

[C#]

```
using CMS.SettingsProvider;
using CMS.GlobalHelper;
```

3. Implement all overloads of your custom method, so that it performs the required functionality and returns the correct value.

[C#]

```
/// <summary>
/// Appends "default" to the specified string.
/// </summary>
public static string MyMethod(string param1)
{
    return MyMethod(param1, "default");
}

/// <summary>
/// Concatenates two strings.
/// </summary>
public static string MyMethod(string param1, string param2)
{
    return param1 + " " + param2;
}
```

4. Create a wrapper suitable for registration as a macro method. The signature and return type of the wrapper method must be exactly the same as shown in the following code:

[C#]

```
/// <summary>
/// Wrapper for MyMethod suitable for registration as a macro method.
/// </summary>
/// <param name="parameters">Parameters of the method</param>
public static object MyMethod(params object[] parameters)
{
    switch (parameters.Length)
    {
        case 1:
            // Overload with one parameter
            return MyMethod(ValidationHelper.GetString(parameters[0], ""));

        case 2:
            // Overload with two parameters
            return MyMethod(ValidationHelper.GetString(parameters[0], ""),
ValidationHelper.GetString(parameters[1], ""));

        default:
            // No other overload is supported
            throw new NotSupportedException();
    }
}
```

5. (Optional) Add another method with an additional *MacroResolver* parameter:

[C#]

```
public static object MyComparisonMethod(MacroResolver resolver, params object[]
parameters)
{
    switch (parameters.Length)
    {
        case 2:
            // Compares the string parameters. Ignores case if the resolver is not
case sensitive.
            return ValidationHelper.GetString(parameters[0], "").EqualsCSafe
(ValidationHelper.GetString(parameters[1], ""),
!resolver.IsCaseSensitiveComparison);

        default:
            throw new NotSupportedException();
    }
}
```

The parameter allows you to access the macro resolver used for the context in which the method was called. The sample method above checks if the resolver is case sensitive and then compares the string parameters accordingly.

6. Add a registration method for your custom macro methods:

[C#]

```
public static void RegisterMethods()
{
    // Registers MyMethod with two parameter overloads.
    MacroMethod myMethod = new MacroMethod("MyMethod", MyMethod)
    {
        Comment = "Returns a concatenation of two strings.",
        Type = typeof(string),
        AllowedTypes = new List<Type>() { typeof(string) },
        MinimumParameters = 1
    };
    myMethod.AddParameter("param1", typeof(string), "First string to
concatenate.");
    myMethod.AddParameter("param2", typeof(string), "Second string to
concatenate.");
    MacroMethods.RegisterMethod(myMethod);

    // Registers a code snippet for MyMethod.
    MacroMethod myMethodSnippet = new MacroMethod("MyMethodSnippet", MyMethod)
    {
        Comment = "Calls MyMethod for the current user's name in lower case.",
        Type = typeof(string),
        AllowedTypes = new List<Type>() { typeof(string) },
        Snippet = "MyMethod(ToLower(CurrentDocument.DocumentName), |);"
    };
    MacroMethods.RegisterMethod(myMethodSnippet);

    // Registers MyComparisonMethod, which uses a resolver object parameter.
    MacroMethod myComparisonMethod = new MacroMethod("MyComparisonMethod",
MyComparisonMethod)
    {
        Comment = "Compares two strings according to resolver
IsCaseSensitiveComparison setting.",
        Type = typeof(string),
        AllowedTypes = new List<Type>() { typeof(string) },
        MinimumParameters = 2
    };
    myComparisonMethod.AddParameter("param1", typeof(string), "First string to
compare.");
    myComparisonMethod.AddParameter("param2", typeof(string), "Second string to
compare.");
    MacroMethods.RegisterMethod(myComparisonMethod);
}
```

Register each macro method through the following steps:

a. Prepare a **CMS.GlobalHelper.MacroMethod** object representing the custom macro method.

- The first parameter of the MacroMethod constructor sets the name used for the method in macro expressions.
- The second parameter specifies the method delegate (wrapper method) containing the definition in the code.

b. Specify the following basic properties:

Property	Description

Comment	Comment displayed for the method in the macro autocompletion.
Type	The return type of the method.
Allowed types	List of object types for which the method can be called. Set to *null* for all types to be allowed. If unspecified, the method is applicable to objects of the type specified for the first parameter.
Minimum parameters	The minimum number of parameters that must be entered when calling the method (minimum overload).
Snippet (optional)	Allows you to define methods with a code snippet. Users can insert the snippet by pressing the TAB key when they have the method selected in the macro autocompletion. Use the pipe character "|" to determine the cursor position after the insertion of the snippet.

c. Define the method's parameters one-by-one through the **AddParameter** method of the *MacroMethod* object. You must specify the name, data type and comment text of each parameter. When the system needs to resolve the custom method in a macro expression, it passes the values of these parameters to the parameter array of the wrapper method.

d. Register the method by calling **MacroMethods.RegisterMethod(MacroMethod methodName)**.

Initializing the custom method registration

To complete the registration, you also need to ensure that the system calls the *RegisterMethods* method of the *CustomMacroMethods* class created in the previous section.

1. Create another class in the **App_Code** folder (or **Old_App_Code** if the project is installed as a web application). For example, name the class **MacroMethodLoader.cs**.

2. Edit the class and add the following reference:

[C#]

```
using CMS.SettingsProvider;
```

3. Delete the default class declaration and its content. Instead, extend the **CMSModuleLoader** partial class and define a new attribute inheriting from *CMS.SettingsProvider.CMSLoaderAttribute*:

[C#]

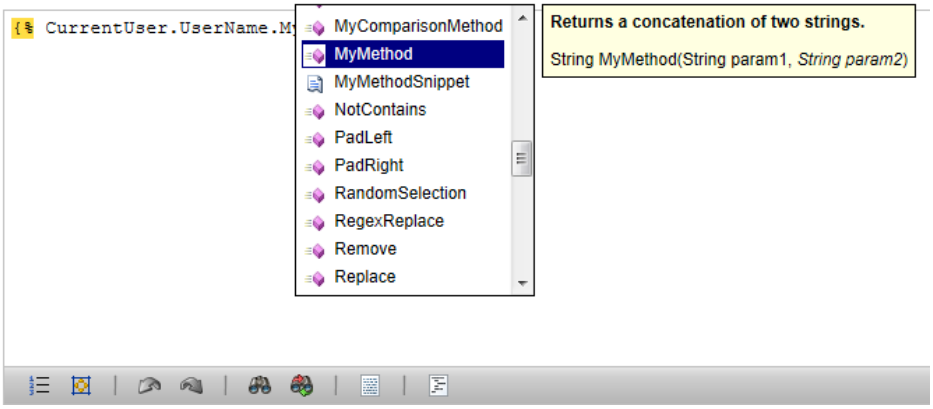
```
[MacroMethodLoader]
public partial class CMSModuleLoader
{
    /// <summary>
    /// Attribute class ensuring the registration of custom macro methods.
    /// </summary>
    private class MacroMethodLoader : CMSLoaderAttribute
    {
        /// <summary>
```

```
        /// Called automatically when the application starts.
        /// </summary>
        public override void Init()
        {
            // Calls the registration method.
            CustomMacroMethods.RegisterMethods();
        }
    }
}
```

When the CMS application starts, it automatically executes the override of the **Init** method, which registers your custom methods.

Result

Your methods are now ready to be used in macro expressions. Open any macro code editor in the Kentico CMS interface and enter a macro with a string data type. The autocompletion box offers the new methods.



The second parameter in the **MyMethod** signature (*String param2*) is displayed in italic letters. This means that the method is overloaded and the parameter is optional.

You can also select the *MyMethodSnippet* method and press the TAB key to insert the defined code snippet.

7.12.10 Resolving macros using the API

If you need to resolve text containing macro expressions in your custom code, you can do so by using the following code:

[C#]

```
string CMS.CMSHelper.CMSContext.CurrentResolver.ResolveMacros(string inputText)
```

As you can see, macros are processed by calling the *ResolveMacros* static method of an appropriate resolver object, typically of the *CMS.CMSHelper.ContextResolver* class.

If you wish to add custom functionality to the resolving process, you can create a new resolver, load additonal data into it and then use it to resolve the macro expression text. This is shown in the following example:

[C#]

```
using CMS.SiteProvider;
using CMS.CMSHelper;

private void CustomResolverExample()
{
    string resolvedText = "";

    // Create child resolver of ContextResolver
    ContextResolver resolver = CMSContext.CurrentResolver.CreateContextChild();

    // Fill the resolver with custom data
    resolver.SetNamedSourceData("MyUser", UserInfoProvider.GetUserInfo
("administrator"));
    resolver.SourceParameters = new object[,] { { "MySimpleValue", "RESOLVED
Simple value!" }, { "MySecondSimpleValue", "RESOLVED Second simple value!" } };

    // Use the resolver to resolve macros in text
    resolvedText = resolver.ResolveMacros("The name of my user is (in upper case):
{% MyUser.UserName.ToUpper() %}. First eight characters of the first simple value:
{% MySimpleValue.Substring(0, 8) %}");
}
```

There are several ways to register new items into macro resolvers. The **SetNameSourceData** method adds a single new macro, with the first parameter specifying the name of the macro and the second one providing the object that will be returned when the macro is resolved. The **SourceParameters** property may be used to add multiple pairs of this type simultaneously through a two dimensional array of objects.

Alternatively, you can define the value of a macro using a callback:

[C#]

```
using CMS.CMSHelper;
using CMS.GlobalHelper;

private object MyPropertyEvaluator(MacroResolver resolver)
{
    return "Macro return value";
}

private void CustomResolverExample()
{
    string resolvedText = "";

    ContextResolver resolver = CMSContext.CurrentResolver.CreateContextChild();
```

```
    // Fill the resolver with custom data
    resolver.SetNamedSourceDataCallback("MyProperty", MyPropertyEvaluator);

    // Use the resolver to resolve macros in text
    resolvedText = resolver.ResolveMacros("Example: {% MyProperty %}";
}
```

This approach allows you to implement a separate method that will return the macro's value instead of having to specify an object directly. The example above is only meant as a demonstration of the basic principles. The method can be as complex as required.

Please note that the callback is executed for every occurrence of the custom macro in an expression. If the evaluator method contains any computationally intensive logic, it is recommended to optimize performance by implementing a caching mechanism for the result.

Resolving localization macros

If you only need to resolve a localization macro, you may use the following static method instead:

```
string CMS.GlobalHelper.ResHelper.LocalizeString(string inputText)
```

Adding custom macros to the global resolver

The techniques described above may also be used to define additional macro properties for existing resolvers. Macro resolvers are organized in a hierarchy that allows child resolvers to inherit all macro options defined for their parent. You can create a child resolver by calling the *CreateContextChild()* method of an existing resolver, as shown in the examples above.

The following step-by-step example demonstrates how custom macros can be added to the global resolver, i.e. the parent of all other resolvers. By adding data items into the global resolver, you can create new macros that will be available in all parts of the system.

1. Open your web project in Visual Studio, expand the **App_Code** folder (or **Old_App_Code** if you installed the project as a web application) and add a new class called **GlobalResolverExtender.cs**.

2. Edit the class and add the following references:

[C#]

```
using CMS.SettingsProvider;
using CMS.GlobalHelper;
using CMS.CMSHelper;
using CMS.SiteProvider;
```

3. Next, delete the default class declaration and its content. Instead extend the **CMSModuleLoader** partial class and define an attribute for it as shown below:

[C#]

```
[GlobalResolverExtender]
public partial class CMSModuleLoader
{
    /// <summary>
    /// Module registration
    /// </summary>
    private class GlobalResolverExtenderAttribute : CMSLoaderAttribute
    {
        ...
    }
}
```

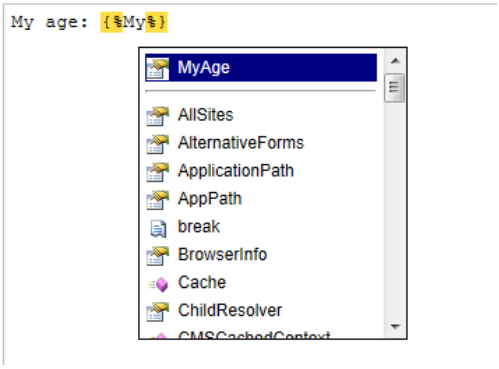
4. Then enter the following code into the **GlobalResolverExtenderAttribute** class:

[C#]

```
/// <summary>
/// Called automatically when the application starts
/// </summary>
public override void Init()
{
    ContextResolver.GlobalResolver.SetNamedSourceDataCallback("MyAge",
MyAgeEvaluator);
}

private object MyAgeEvaluator(MacroResolver resolver)
{
    System.DateTime dateOfBirth =
CMSContext.CurrentUser.UserSettings.UserDateOfBirth;
    System.DateTime now = DateTime.Now;

    int age = now.Year - dateOfBirth.Year;
    if (dateOfBirth > now.AddYears(-age))
    {
        age--;
    }
    return age;
}
```

The code in the override of the **Init()** method (executed automatically when the application starts) is used to add the **MyAge** macro property into the global resolver. The value of this macro is retrieved via callback by the private **MyAgeEvaluator** method.

5. **Save** the file and **Build** the project if it is installed as a web application. You can now try entering the *{% MyAge %}* expression in any part of the system where macros are supported. It will automatically be offered in the autocompletion box, as well as the object tree provided by the macro selection control.



As you can see, the custom property is displayed in the high priority section at the top of the list. By default, all items registered through the *SetNameSourceData* or *SetNameSourceDataCallback* methods are prioritized. If you wish to add a property with normal priority, you can use an overload of the methods that includes the *isPrioritized* boolean parameter and set it as *false*. For example:

[C#]

```
ContextResolver.GlobalResolver.SetNamedSourceDataCallback("MyAge",
MyAgeEvaluator, false);
```

7.12.11 Macro result caching

It is possible to cache macro results so that certain expressions don't have to be evaluated repeatedly. This is possible by enclosing the expression into the **Cache** method. The method has the following signature:

- **Object Cache(Object expression, *Int32 cacheMinutes*, *Boolean condition*, *String cacheItemName*, *String cacheItemNameParts*, *CMSCacheDependency cacheDependency*)**

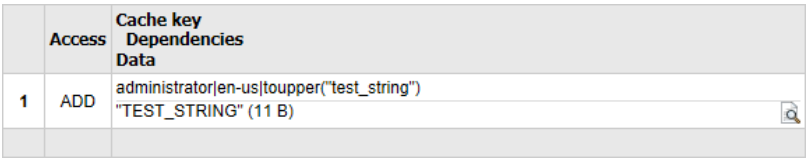
The method is overloaded and only the first parameter is mandatory — the rest of the parameters are optional. The parameters determine the following:

- string macroExpression: Macro expression (without encapsulating character sequences, e.g. *{% %}*) to be evaluated and cached.
- *Int32 cacheMinutes*: Number of minutes for which the value will be cached.
- *Boolean condition:* Boolean expression whose value determines if the macro expression should be cached.
- *String cacheItemName*: Name of the cache item under which the value will be stored in the cache.
- *String cacheItemNameParts*: Any number of parameters whose values will be concatenated (together with the *cacheItemName* parameter value) and used as the name of the cache item.
- *CMSCacheDependency cacheDependency*: Cache dependency (use the *GetCacheDependency* method to get the object).

The following example demonstrates how it can be used to cache a simple *"test_string".ToUpper()* expression.

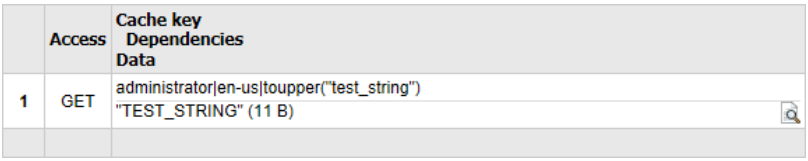
```
{% Cache("test_string".ToUpper()) %}
```

When the macro is first evaluated, it is resolved as TEST_STRING and the result is stored in cache. In the screenshot below, you can see the respective cache access debug record for this operation.



	Access	Cache key Dependencies Data
1	ADD	administrator|en-us|toupper("test_string") "TEST_STRING" (11 B)

On each subsequent resolving, the expression is not resolved again and its value is taken from cache.

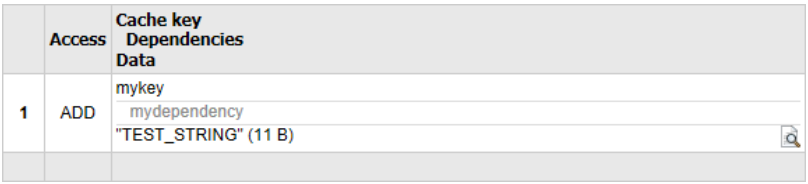


	Access	Cache key Dependencies Data
1	GET	administrator|en-us|toupper("test_string") "TEST_STRING" (11 B)

The following example shows how the method can be used with the optional parameters to specify the cache item name, time period for which the value will be cached and a cache dependency.

```
{% Cache("test_string".ToUpper(), 5, true, "mykey", GetCacheDependency
("mydependency")) %}
```

In this case, the cache access debug record looks as in the following screenshot.



	Access	Cache key Dependencies Data
1	ADD	mykey mydependency "TEST_STRING" (11 B)

7.13 Membership, permissions and security

7.13.1 Security model overview

Kentico CMS provides a flexible security model that allows you to configure granular access permissions for content and modules.

The security model consists of:

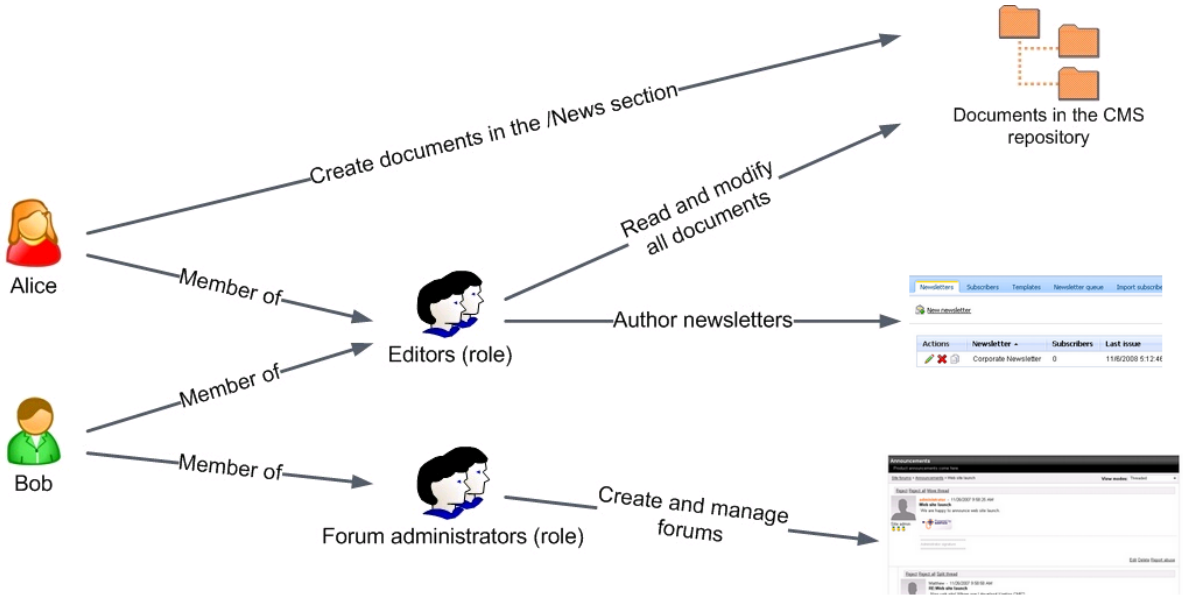
- users (shared among websites)
- roles (defined for particular websites or globally for all sites in the system)
- memberships (collections of roles that can be assigned to users)
- module permissions
- document permissions
- UI personalization

Users, roles and permissions can be managed on two levels:

- In **Site Manager -> Administration**, where global administrators can configure the data of all sites and define global objects.
- In **CMS Desk -> Administration**, where local administrators can edit only data related to the current website (the current website is recognized by the current domain).

Relationships between users, roles and permissions

The following figure shows how users are assigned to roles and how permissions for documents and modules are granted to users and roles:



Users can be members of any number of roles. Permissions for particular documents in the CMS repository can be granted to them. If you want to grant permissions for some module to a user, you need to make the user a member of a role and grant the permissions to the role (i.e. permissions for modules cannot be granted to users directly).

Roles in Kentico CMS are fully customizable. It means you're not limited to a predefined set of roles. Instead, you can define your own roles with custom sets of permissions.

If a user is a member of multiple roles, their **permissions for modules** are calculated as a sum of all permissions granted to all roles.

If **permissions for documents** in the CMS repository are granted to both a user and their roles, document permissions are calculated as a sum of all permissions granted to the user and to all roles. If a user or some of their roles are **denied to make some action** (such as modify document), then the result is always "denied" for the given permission, even if some of the roles are allowed to perform the action.

7.13.2 User management

A user can be a member of any number of roles and can be assigned to any number of websites.

There are two important attributes of every user account:

- **Is editor** - the user can access **CMS Desk** and the On-site editing interface. The attribute does not grant any particular permissions — it only differentiates between site editors and registered users who are limited to the live website. This provides an extra security layer. Users who are editors can access the editing interface of all sites to which they are assigned on the **Sites** tab.

- **Is global administrator** - the user is authorized to access all sections of the system and perform any operations, regardless of permissions or other settings. Global administrators are the only users who can use the **Site Manager** interface.

Global administrators

Global administrators are the only users allowed to manage site settings and all development tools. Their permissions cannot be denied or limited – they have access to all features and data.

Local administrators cannot modify global administrator accounts.

Default user accounts

The following default user accounts are available:

- **Administrator** – global administrator user with full permissions.
- **Public** – user that represents an anonymous visitor of the site.

Creating a new user

New user accounts are typically created when a user goes through registration on the live site. However, you can also create accounts manually in **Site manager -> Administration -> Users** or **CMS Desk -> Administration -> Users**. Click **New user** and enter the following properties into the displayed form:

- **User name** - the user's user name (login). By default, it must be unique across all websites in the system.
- **Full name** - user's full name (first name, middle name and last name).
- **E-mail** - user's e-mail address.
- **Enabled** - indicates if the user account is enabled and the user can sign in.
- **Is editor** - indicates if the user is allowed to sign in to CMS Desk and access the On-site editing interface. This is used to differentiate between users who are only allowed to visit member areas of the website and content editors.
- **Password** - user's password.
- **Confirm password** - user's password again for confirmation.

User passwords

It is highly recommended to set a safe password for every user account to ensure the security of your website. Global administrators can monitor the list of users for accounts that have empty passwords, which will be marked with a warning icon ( ).

You can add a password manually by editing the given users, specifically on the **Password** tab.

The system can be configured to require users to enter passwords matching specific strength requirements. For more information, please see the Password management -> Password strength policy topic.

Editing user properties

You can edit user properties in **Site manager -> Administration -> Users** -> click the **Edit** () icon of the chosen the user.

General properties

The following properties can be set on the **General** tab:

- **User name** - the user's user name (login). By default, it must be unique across all websites in the system.
- **Full name** - user's full name (first name, middle name and last name).
- **First name** - user's first name.
- **Middle name** - user's middle name.
- **Last name** - user's last name.
- **E-mail** - user's e-mail address.

- **Enabled** - indicates if the user account is enabled and the user can sign in.
- **Is editor** - indicates if the user is allowed to sign in to CMS Desk and access the On-site editing interface. This is used to differentiate between users who are only allowed to visit member areas of the website and content editors.
- **Is global administrator** - indicates if the user is a global administrator. Global administrators have full permissions for all features and data across the system and are not affected by permission settings for particular modules.
- **Is external user** - this attribute is used when you are using an integration with an external user database.
- **Is domain user** - indicates if the user was imported from Active Directory.
- **Is hidden** - if true, the user is not visible on the site (e.g. on-line user monitoring, repeaters displaying users, etc.).
- **Disable site manager** - this option is available only when editing a global administrator, but not when a global administrator is editing their own account. If enabled, the user will still be designated as a global administrator, but will not be able to access the Site Manager interface, i.e. will only be allowed to perform actions in CMS Desk.

- **Preferred content culture** - preferred culture in which the content is displayed to the user.
- **Preferred user interface culture** - preferred culture in which the users wants to see the user interface (CMS Desk and Site Manager).

- **Created** - date and time when the user account was created.
- **Last logon** - date and time when the user last logged in.
- **Last logon information** - information about the IP address and browser user agent of the user's last logon.

- **Starting alias path** - the starting alias path of the content tree in CMS Desk -> Content; if you specify this value, the user is not allowed to browse other sections of the website in the content tree; please note that this feature is only intended for better usability and it doesn't ensure security control - if you need to establish access rights for a given user, grant appropriate document permissions (Properties -> Security) to them

You can also view the following information and perform related actions:

- **Invalid logon attempts** - number of unsuccessful attempts to log in with a wrong password. You can reset the value to zero and unlock the user's account by clicking the **Reset & enable** button.
- **Password expires in** - number of days left until the user's password expires. You can reset the validity to the maximum value by clicking **Extend validity & enable**.

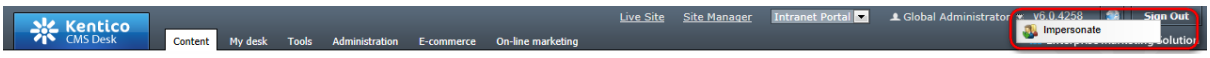
User impersonation

Global administrators have the option of using the **Log in as this user** button displayed at the top of the **General** tab. This allows them to sign in as the currently edited user and view the website or CMS Desk interface from their perspective. When this action is performed, the administrator will be redirected depending on the type of the impersonated user:

- **Editor** - if impersonating an editor (a user with the **Is editor** option enabled), you will be redirected to CMS Desk.
- **Standard user** - if impersonating a standard user, you will be redirected to the title page of the live site.

It is not possible to impersonate other global administrators.

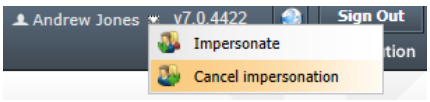
User impersonation may also be performed from anywhere in the administration interface by opening the context menu located on the main header of **CMS Desk** or **Site Manager** and selecting **Impersonate**.



When clicked, a dialog containing a list of all users under the current site is displayed, where you can select which specific user should be impersonated.

Actions carried out while impersonating a user are logged in **Site Manager -> Administration -> Event log** under a user name in format *<user name> (<original user name>)*, where the original user is the administrator using the impersonate function.

When impersonating an editor in CMS Desk, you can return to the original global administrator at any time by selecting the **Cancel impersonation** option available in the same context menu.



Password

Here you can change the user's password:

- **Password** - user's password.

- **Confirm password** - user's password again for confirmation.

You can either enter a new password directly, or have the system generate a new one. The tab also provides the option to send an automatic notification e-mail to the given user containing the new password.

This tab is hidden if the edited user is authenticated using either an external user database or Active Directory, i.e., if the user has the **Is external user** property enabled on the **General** tab of the user editing interface or if **Is domain user** is enabled and the application is configured to use Windows authentication.

Settings

On the **Settings** tab, you can edit the following properties of the user:

- **User nick name** - nick name of the user used in website forums, on the user's profile, etc.
- **User picture** - user's avatar image; this image will be used in forums and on user's profile; you can either upload an image or select a pre-defined avatar
- **User signature** - user's signature that will be used below the user's forum posts
- **Description** - optional text describing the user
- **URL referrer** - URL from that the user came to the site when they performed registration
- **Campaign** - if the given user arrived on the website through a campaign before registering, this field will store the name of that campaign. Please see the Modules -> Web analytics -> Campaigns chapter for details.
- **Messaging notification e-mail** - notifications about new messages received in the messaging module will be sent to this e-mail address

- **Time zone** - user's time zone; if set, this time zone will be used where applicable instead of the site time zone
- **Badge** - user's badge; depends on the number of gained activity points

- **User activity points** - number of user's activity points; these points are gained for forum posts, message board posts, blog posts and blog post comments
- **Live ID** - user's Live ID token; this is a hexadecimal number that the user is identified by when logging-in via Windows Live ID
- **Facebook user ID** - user's Facebook user ID; it is used when the user is logging in via Facebook Connect
- **OpenID** - user's OpenID; it is used when the user is logging in via OpenID
- **LinkedIn ID** - user's LinkedIn ID; it is used when the user is logging in via LinkedIn authentication

- **Activation date** - date of the user's account activation
- **Activated by user** - user who activated this user's account
- **Registration info** - user's IP and browser agent detected on registration

- **Gender** - user's gender
- **Date of birth** - user's date of birth

- **Skype account** - user's Skype account
- **Instant messenger** - user's instant messenger; format of values of the field is not strictly required, you may use any string of characters according to your specific needs (e.g. *ICQ: 123456789*)
- **Phone number** - user's phone number; the number may be entered in any format, no validation is applied

- **Log activities** - indicates if on-line marketing activities should be logged for this user
- **Waiting for approval** - if checked, the user account is not active yet and is waiting for an administrator's approval
- **Show splash screen** - determines if splash screen should be displayed to the user when accessing Kentico CMS administration interface
- **Show web part toolbar** - determines whether the web part toolbar should be displayed for the given user when editing documents in CMS Desk on the *Content -> Edit -> Design* tab. This is only relevant for users who have the *Design web site* permission
- **Web part toolbar position** - if the web part toolbar is enabled for the user, the choice made here sets its location on the Design tab

- **Forum posts** - number of user's forum posts
- **Forum comments** - number of user's forum comments
- **Blog comments** - number of user's blog comments
- **Message board posts** - number of user's message board posts

Custom Fields

Here you can edit the custom fields added to the user profile. The custom fields can be defined in **Site Manager -> Development -> System tables -> User**.

Sites

Here you can specify the sites into which the user can sign in using their user name and password credentials. To assign the user to a site, simply click the **Add sites** button, check the appropriate boxes in the displayed dialog and click **OK** to save the changes.

Please note

The sites assigned here only limit access to the **CMS Desk** interface. Logging in on the live site is possible even for users who are not assigned to the given site.

This is intended to allow the separation of access privileges for content editors responsible for different websites.

Roles

Here you can manage the roles to which the edited user is assigned. Depending on the permissions available for individual roles, the user will be authorized to perform various actions on the website or in the administration interface. Please refer to the Role management topic for further information about roles.

Departments

Here you can specify the E-commerce module departments in which the user is authorized to manage products.

Notifications

On this tab, you can see a list of all notification subscriptions of the currently edited user. You can **Delete** () subscriptions in the list, which unsubscribes the user from receiving notifications.

Categories

This tab displays a list of the user's custom categories. Each of the categories can be edited () or deleted ().

By clicking the **New category**, you can create a new category that will behave the same way as if it was created by the user in **CMS Desk -> Edit -> Properties -> Categories**.

The following details will be required when creating a new category:

- **Display name** - name of the category displayed in the user interface
- **Code name** - name of the category used in website code

Friends

On this tab, you can manage the currently edited user's friends.

Subscriptions

On this tab, you can manage the user's subscriptions to newsletters, blog posts (comment notifications), message boards, forums and reports.

Languages

On this tab, you can specify which cultural versions of documents can be edited by the user. You have the following options:

- **User can edit all languages** - if selected, the currently edited user can edit documents in all language versions of all sites in the system
- **User can edit following languages** - if selected, you can specify which language versions can be edited by the user by checking the check-boxes in the list of language versions; this can be set separately for each site in the system using the **Select site** drop-down list

Memberships

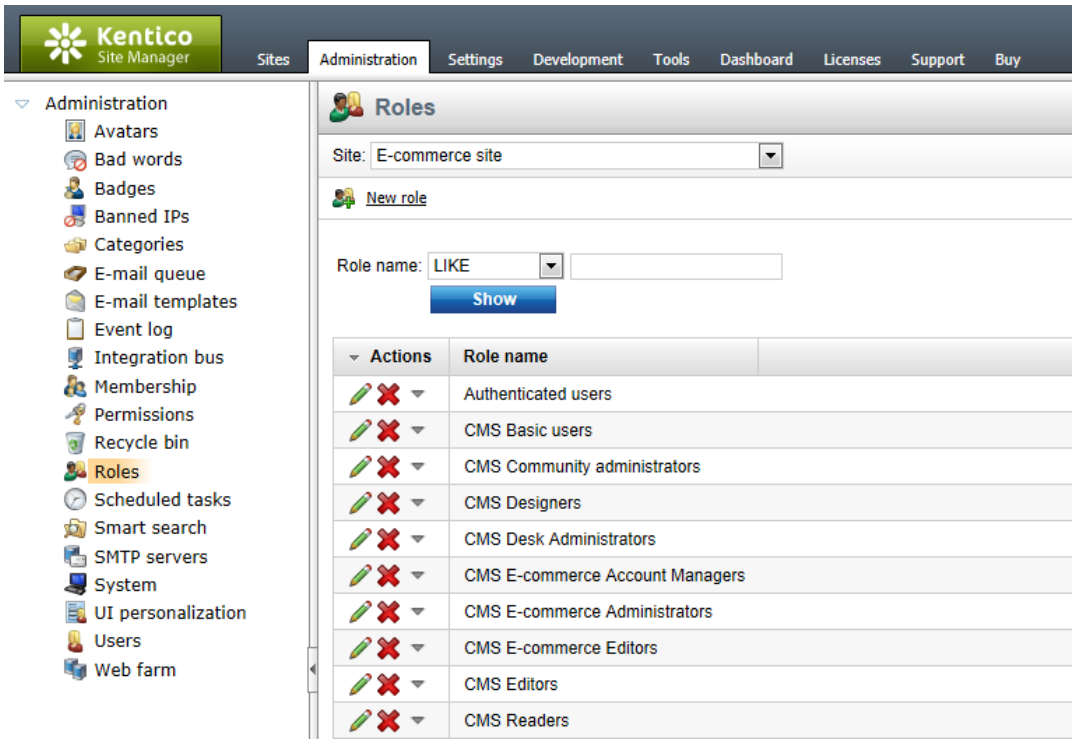
Here you can manage special types of website membership assigned to the edited user. Each membership represents a collection of roles. When a membership is assigned to a user, it automatically authorizes that user to perform any actions allowed for all contained roles. Please refer to Memberships to learn more.

7.13.3 Role management

Roles are objects that define authorization options for users, i.e. which actions they are allowed to perform on the website and within the Kentico CMS administration interface. Roles provide an interface that maps permissions to users in a way that can easily be reused. Each role can be assigned to any amount of users and vice versa, a user can be a member of an unlimited number of roles.

The management interface for roles can be found in the **Administration -> Roles** section, both in **CMS**

Desk and **Site Manager**.



Roles can either be assigned to a specific site or defined as global objects that are available for all sites. In **CMS Desk**, only roles that belong under the current site can be managed. When in **Site Manager**, you can select the site context using the **Site** drop-down list at the top of the page. To access the list of all global roles in the system, choose the *(global)* option. Using global roles can save a lot of time when working with a large number of sites that require similar types of authorization options. Please note that global roles may only be assigned to users by global administrators.

It is possible to edit (✎) or delete (✖) the roles displayed in the list. New roles can be created for the selected site (or globally) by clicking **New role**. You can specify the following properties when adding a new role:

- **Role display name** - sets a name for the role displayed to users in the administration interface.
- **Role code name** - sets a name that serves as an identifier for the role.
- **Role description** - can be used to enter an optional text description for the role.
- **Is domain role** - indicates if the role was imported from Active Directory.

> **Code names of global roles**
>
> Code names are only checked for uniqueness within the context of individual sites, which means that it is possible for a global role to have the same code name as a site-specific role.
>
> If you need to specify a global role using its code name in your custom website code or via the API, you can add the period character (".") as a prefix. This ensures that only the global role will be selected and any site roles with the same code name will be

ignored (for example *.Content_admin*).

Editing a role

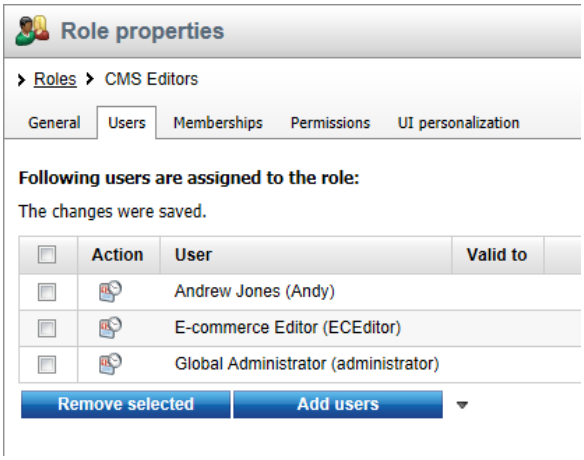
There are four tabs available when editing ( ) a role:

General

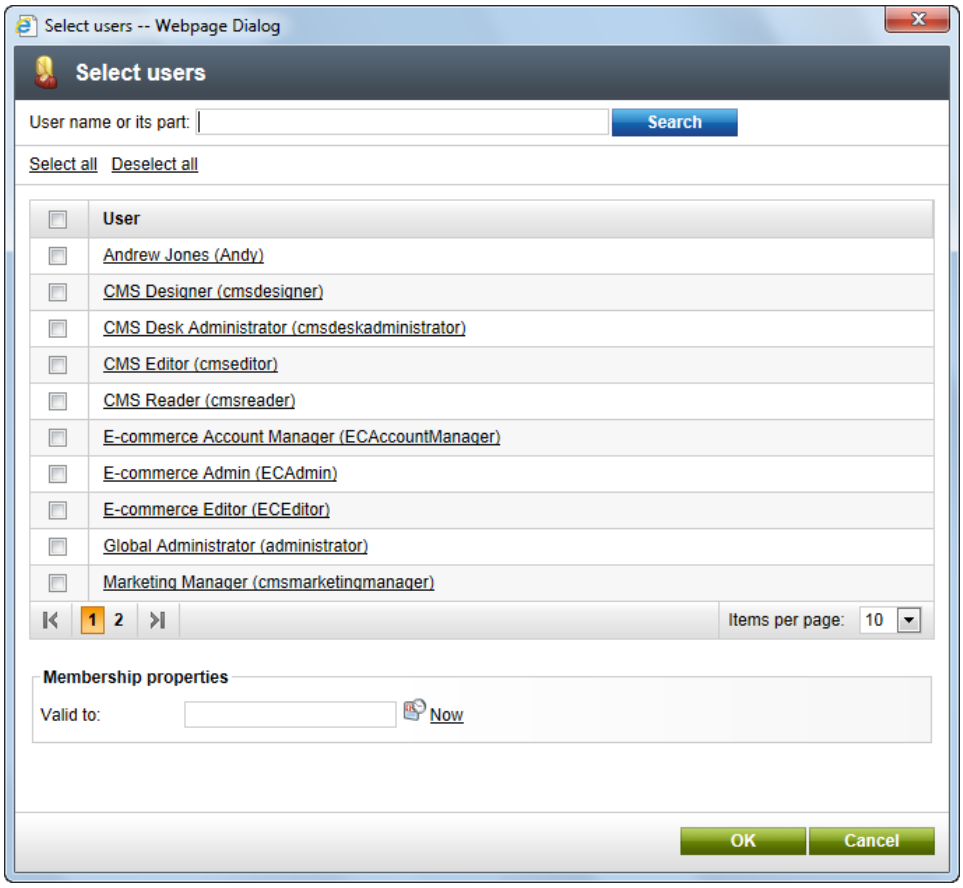
On this tab you can edit the same properties that were specified when the role was created.

Users

Here you can add or remove users to/from the currently edited role. These users will be authorized to perform actions according to the permissions granted to the role on the **Permissions** tab. Roles can either be assigned to users permanently or only until a specified date and time.



If you wish to add users, click the **Add users** button and check the boxes next to the appropriate users in the displayed selection dialog.



Only users who are assigned to the same site as the role can be chosen (global roles may be assigned to all users in the system). You can enter a name or its part into the textbox above the list and click **Search** to quickly find any user. The **Valid to** field at the bottom of the dialog can be used to assign users to the role for a limited time only. The **Calendar** button can be clicked to easily select the exact date and time when the role should expire for the user. If this field is left empty, the users will be assigned to the role for an unlimited time period. Click **OK** to apply any changes.

The **Change validity** action that is available for every listed user may be used to prolong or shorten the time interval for which the user should be assigned to the role. This way you can set an expiration date or reactivate expired roles for users.

Users can be removed from the role at any time using the checkboxes in the list together with the **Remove selected** button.

Permissions

On this tab you can configure which permissions should be assigned to the given role. If you wish to add permissions, select the type of the permission that you wish to assign using the two drop-down lists. Once this is done, individual permissions for the specified module or document type will be displayed below and can be enabled by checking the corresponding boxes.

UI Personalization

From the this tab, you can choose which parts of the CMS Desk interface will be displayed to members

of the given role. You can also change the settings of the WYSIWYG Editor for this particular role.

Individual interface elements can be configured from the **Dialogs** sub-tab. This is where you can select a module from the **Module** drop-down list and then enable or disable the visibility of individual dialogs by checking the corresponding boxes. Only dialogs that are checked will be displayed to users assigned to the given role. You can easily reach nested dialogs by clicking the **Expand all** and **Collapse all** links, respectively. You can also expand and collapse dialogs on different levels of the dialogs tree. If you would like to adjust WYSIWYG Editor settings for the given role, switch to the **Editor** tab. You can make adjustments in a similar way as with dialogs.

For more information, please see the UI personalization chapter.

7.13.4 Username customization

If you want to customize the way usernames are displayed in the administration interface, you can do it by modifying the **GetFormattedUsername** method in **~/AppCode/CMS/Functions.cs**. The method has four overrides and is used to retrieve usernames in the whole administration interface.

Example: Usernames are displayed in the *<full name> (<user name>)* (e.g. *Abigail Woodwarth (Abi)*) format in some parts of the system, e.g. in document **Properties -> General -> Owner**. The following code example shows how you can modify the method to get usernames in format *<user name> [<full name>]* (e.g. *Abi [Abigail Woodwarth]*):

[C#]

```
public static string GetFormattedUserName(string username, string fullname, bool
isLiveSite)
{
    if (!String.IsNullOrEmpty(DataHelper.GetNotEmpty(fullname, "").Trim()))
    {
        return String.Format("{1} [{0}]", fullname, username);
    }
    else
    {
        return username;
    }
}
```

7.13.5 Permissions

7.13.5.1 Permissions overview

Permissions provide a way how you can control access to particular sections of the Kentico CMS administration interface (modules), documents in the content tree and custom tables.

Permissions for roles

In addition to global roles defined for all sites in the system, every website has its own set of roles. Permissions are assigned to these roles, which means that every website can use a different configuration of role permissions as necessary. Permissions for roles can be configured in the **Administration -> Permissions** section of both **CMS Desk** and **Site Manager**. The difference between the two locations is that in CMS Desk, permissions can only be configured for roles belonging under the

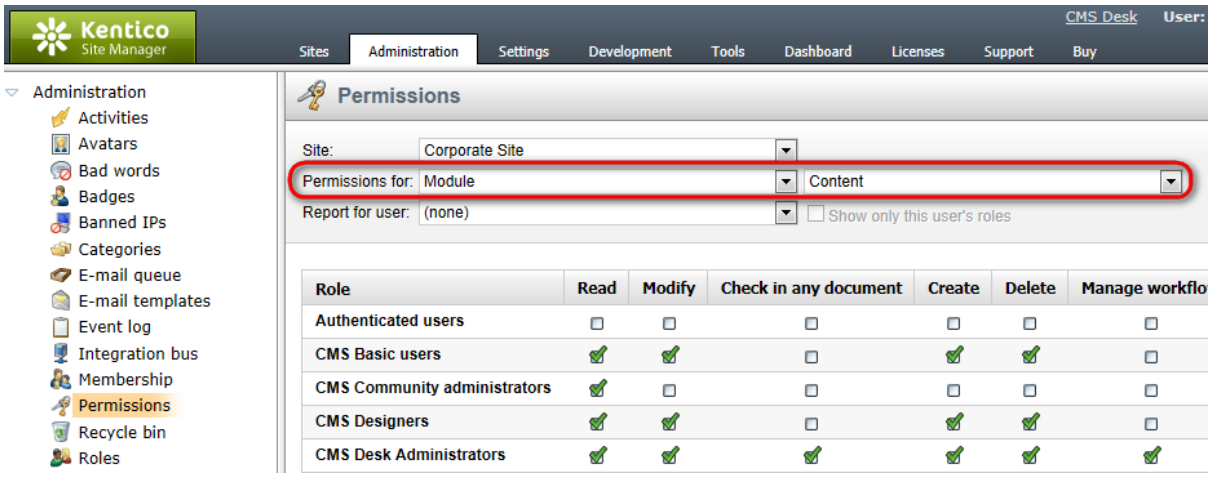
currently edited website, while in **Site Manager**, you can configure permissions for all sites in the system or for global roles by selecting a site from the **Site** drop-down.

Based on the selection made by the first **Permission for** drop-down list, you can choose from the following three types of permissions:

- **Modules** - permissions for specified actions in Kentico CMS modules. You can find details on particular permissions in documentation of respective modules.
- **Document types** - permissions applied to all documents of a particular type. These permissions represent one level of the three-level document permissions hierarchy, as described in the Document permissions topic.
- **Custom tables** - permissions for the custom tables module, see Modules -> Custom tables -> Security for more info.

Then you need to select the appropriate module, document type or custom table from the second drop-down list and grant the permissions to roles using the check-boxes:

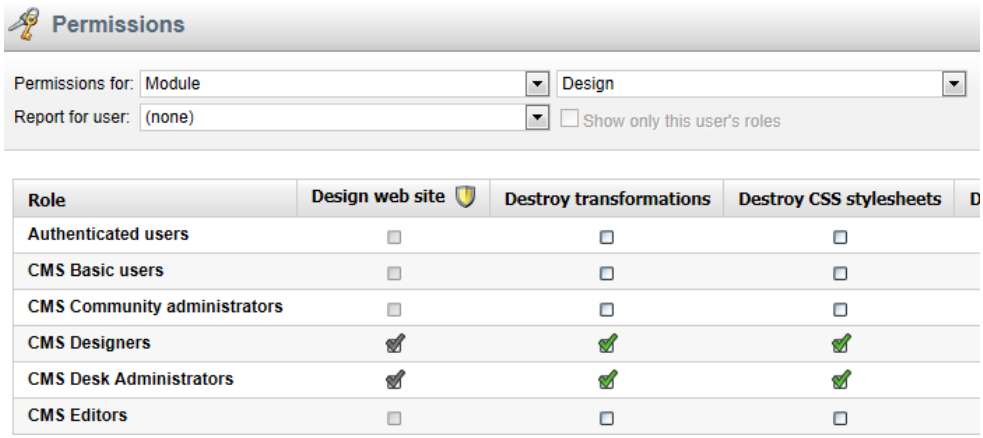
- ☑ - the permission is granted to the role.
- ☐ - the permission is not granted to the role.



When performing this task in **CMS Desk -> Administration -> Permissions** while you are not a global administrator, you may come across the following grayed-out check boxes:

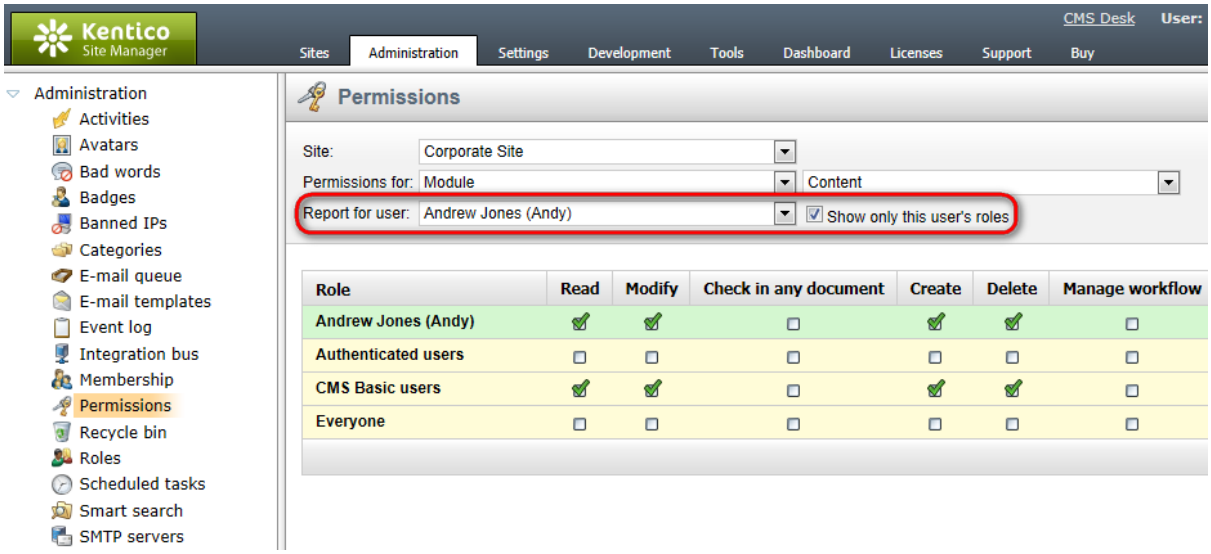
- ☑ - the permission is granted to the role, and only a global administrator can change it.
- ☐ - the permission is not granted to the role, and only a global administrator can change it.

These grayed-out check-boxes are also accompanied by the shield icon in the header row of the table, indicating that the permission can only be granted to roles by the global administrator, as can be seen in the screenshot below.



Permission reports for users

As permissions are assigned to roles, not directly to users, it is possible to display a permission report for each website user. This can be achieved by selecting the user from the **Report for user** drop-down list. After doing so, a sum of all permissions granted to the user's roles is displayed in the first line, highlighted in green color. Roles where the selected user is a member will be highlighted in yellow color. If you enable the **Show only this user's roles** check-box, only the yellow roles will be displayed in the matrix.



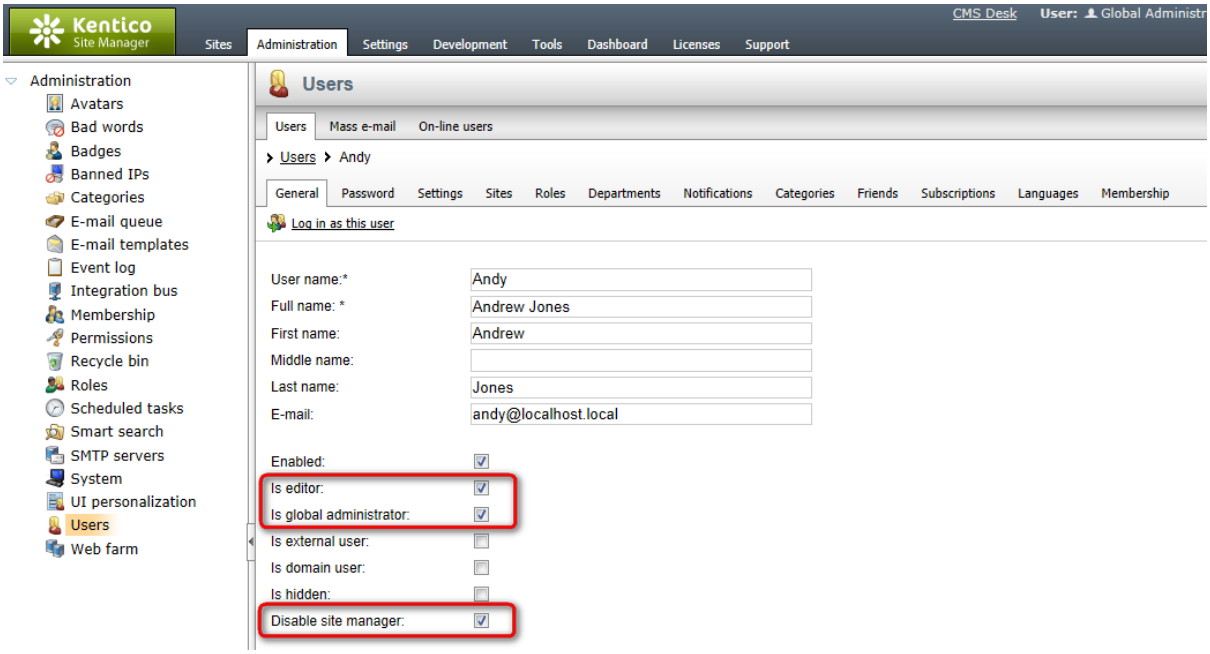
Permission-related settings for users

When editing ( ) a user in **Administration -> Users**, you can enable the following two options. These options have impact on permission checking and provide an extra security layer:

- **Is global administrator** - the user is authorized to perform all operations and their access cannot be denied by permissions or otherwise limited. Global administrators are the only users who can use the Site Manager interface.
- **Is editor** - the user can access CMS Desk and the On-site editing interface. This attribute does not

grant any particular permissions — it only differentiates between site editors and registered users who are limited to the live website. Users who are editors can access the editing interface of all sites to which they are assigned on the **Sites** tab.

- **Disable site manager** - this option is only available for users who are designated as global administrators. If enabled, the user will have unrestricted access to all actions in CMS Desk like all global administrators, but will be unable to use the Site Manager interface. This combination of options can be used to authorize users as administrators for specific sites without having to worry about setting individual permissions. Administrators cannot change the value of this property for their own account.



7.13.5.2 Document permissions

7.13.5.2.1 Document permissions

Permissions for access to Kentico CMS documents can be configured at three levels of the three-level permissions hierarchy. Click the links in the **Document permission level** column to learn more about each of them.

Document permission level	Granted to	Applied to	Configurable in
Permissions for all content	roles	all documents	CMS Desk / Site Manager -> Administration -> Permissions
Document type permissions	roles	all documents of one particular document type	CMS Desk / Site Manager -> Administration -> Permissions
Document-level permissions	roles or individual	one particular document	CMS Desk -> Content -> Edit -> Properties -> Security

	users		

Permissions from these three levels are merged together when checking if a user is a permitted to perform an action with a document. For example, to read a *CMS.News* document, a user must have the *Read* permission on at least one of the three levels: either on document-level, or for the *CMS.News* document type, or for all content.

There is also one special setting that allows hiding of documents in the content tree depending on document permissions granted to the current user. See the Hiding documents in the content tree based on permissions topic for more information on this possibility.

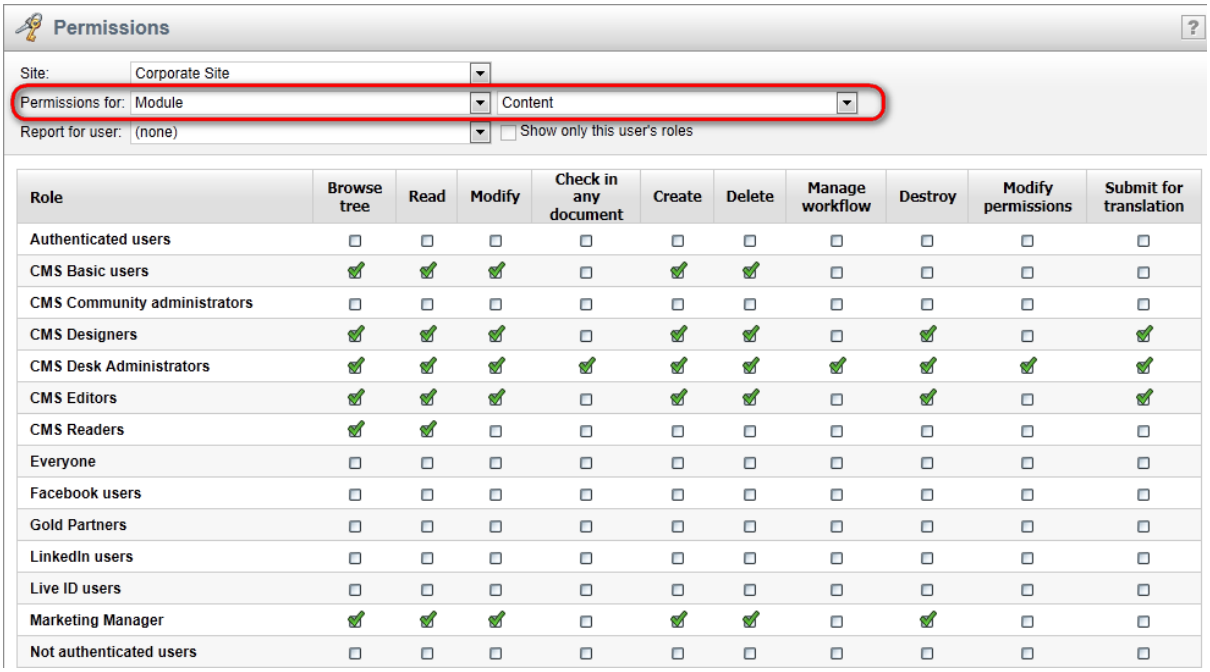
In the Document permissions internals and API sub-chapter, you can find information about database tables and classes that are used for document permissions, as well as several examples of how permissions can be logged and managed using Kentico CMS API.

7.13.5.2.2 Permissions for all content

In **Site Manager -> Administration -> Permissions**, you can find a special permission matrix for controlling access to all documents within the content tree. It is the **Module -> Content** permission matrix. The following global permissions can be granted to particular roles:

Browse tree	Allows members of the role to view the document content tree in CMS Desk (not necessarily the actual document content — that is determined by the *Read* permission). For users without this permission, the system blocks the entire **Content** tab (except for users who belong to roles with the **Browse tree** permission for the *CMS.Root* document type or for the given website's *Root* document on the document level). Users also need this permission to perform document management actions through the On-site editing interface.
Read	Allows members of the role to view the content and settings of documents in the CMS Desk content tree. Additionally, users require this permission to access the on-site editing interface.
Modify	Allows members of the role to modify any document in the content tree. Note that for for approving or rejecting documents within a workflow, the **Manage workflow** permission is sufficient.
Check in any document	Authorizes members of the role to perform the **Check-in** and **Undo check-out** actions on the **Properties -> Versions** tab of a document's editing interface.
Create	Allows members of the role to create documents of any document type in the content tree.
Delete	Allows members of the role to delete any document in the content tree.
Manage workflow	Allows members of the role to approve or reject any document at any workflow step. Note that the **Modify** permission is not necessary for approving or rejecting documents.
Destroy	Allows members of the role to destroy (delete without the *Undo* option) any document.

Modify permissions	Allows members of the role to manage document-level permissions of any document in **CMS Desk -> Content -> Edit -> Properties -> Security**.
Submit for translation	Allows members of the role to create new language versions of documents using Translation services.



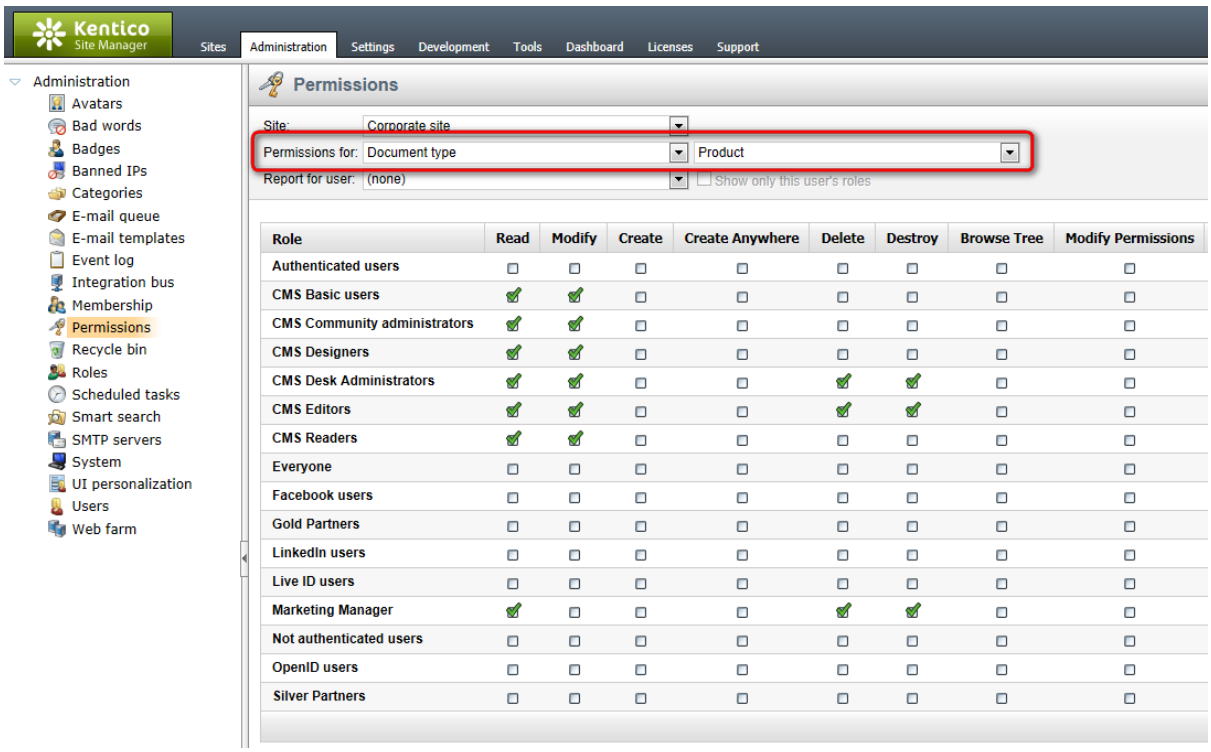
Role	Browse tree	Read	Modify	Check in any document	Create	Delete	Manage workflow	Destroy	Modify permissions	Submit for translation
Authenticated users	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
CMS Basic users	☑	☑	☑	☐	☑	☑	☐	☐	☐	☐
CMS Community administrators	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
CMS Designers	☑	☑	☑	☐	☑	☑	☐	☑	☐	☑
CMS Desk Administrators	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑
CMS Editors	☑	☑	☑	☐	☑	☑	☐	☑	☐	☑
CMS Readers	☑	☑	☐	☐	☐	☐	☐	☐	☐	☐
Everyone	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
Facebook users	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
Gold Partners	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
LinkedIn users	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
Live ID users	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
Marketing Manager	☑	☑	☑	☐	☑	☑	☐	☑	☐	☐
Not authenticated users	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐

7.13.5.2.3 Document type permissions

Document type permissions allow control of access to all documents of a particular document type. These permissions are assigned to roles in the **Administration -> Permissions** section of both **Site Manager** and **CMS Desk**. In this section, you have to choose *Document type* from the first **Permissions for** drop-down list and then choose the required document type from the second one. You can grant the following document type permissions to particular roles:

Read	Allows members of the role to view any document of this type.
Modify	Allows members of the role to edit any document of this type.
Create	Allows members of the role to create documents of this type. With this permission, users must also have the **Create** document-level permission on the parent document under which they want the new document to be created.
Create anywhere	Allows members of the role to create documents of this type anywhere in the content tree, without the need to have the **Create** document-level permission on the parent document under which they want the new document to be created.
Delete	Allows members of the role to delete any document of this type.
Destroy	Allows members of the role to destroy (delete without the *Undo* option)

	any document of this type.
Browse tree	Allows members of the role to see documents found under documents of this type in the content tree.
Modify permissions	Allows members of the role to manage document-level permissions of all document of this type in **CMS Desk -> Content -> Edit -> Properties -> Security**.



7.13.5.2.4 Document-level permissions

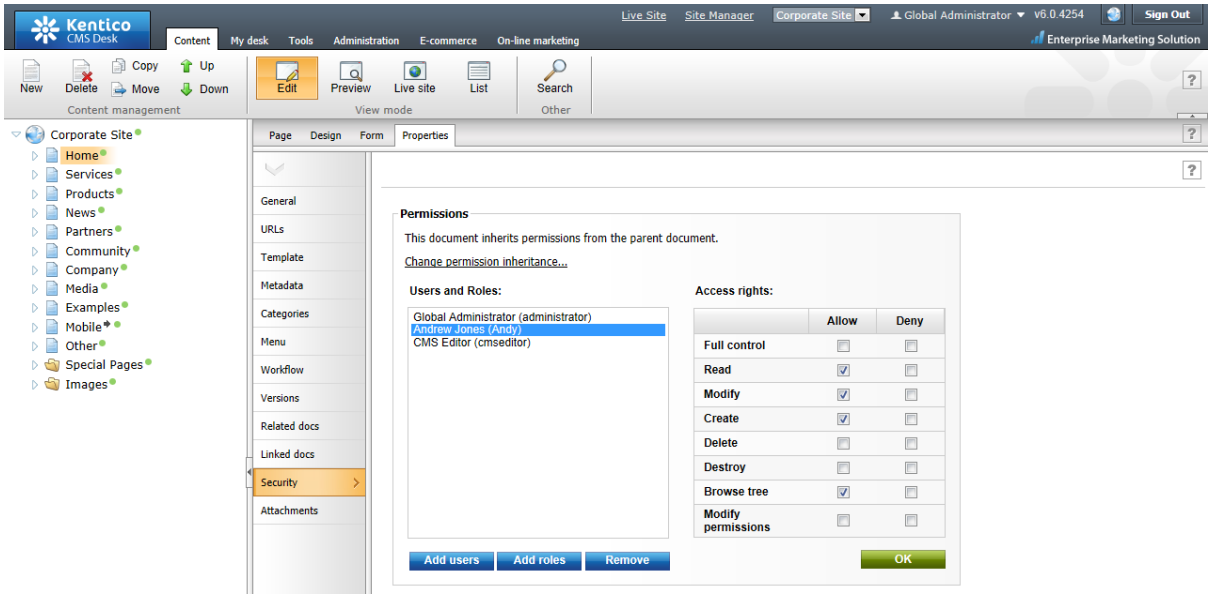
You can manage document-level permissions (i.e. permissions for a particular document or a particular website section) in **CMS Desk -> Content -> Edit -> Properties -> Security**. Select the appropriate user or role in the left box. If the user or role is not available in the box, you may need to add them using the **Add users** or **Add roles** button. Now you can choose if the permissions should be allowed or denied:

- **Allow** - the action will be allowed to the user or role.
- **Deny** - the action will not be allowed even if the user or role has the permission assigned on a global level. I.e. the *Deny* option overrides settings for this permission on the other two levels.

The following permissions can be allowed or denied:

Full control	Allows the user or members of the role to perform any action with this document.
Read	Allows the user or members of the role to view this document.

Create	Allows the user or members of the role to create new documents under this document.
Modify	Allows the user or members of the role to edit this document.
Delete	Allows the user or members of the role to delete this document.
Destroy	Allows the user or members of the role to destroy (delete without the *Undo* option) this document.
Browse tree	Allows the user or members of the role to see documents found under this document in the content tree.
Modify permissions	Allows the user or members of the role to manage document-level permissions of this document in **CMS Desk -> Content -> Edit -> Properties -> Security**.



Permission inheritance

You will typically need to set up permissions for site sections rather than for particular documents. In this case, you can grant permissions for the section's parent document and inherit them by all child documents.

Example

Consider a website structure like this:

- Root
  - Home
  - News
  - Products
    - Category 1
    - Category 2

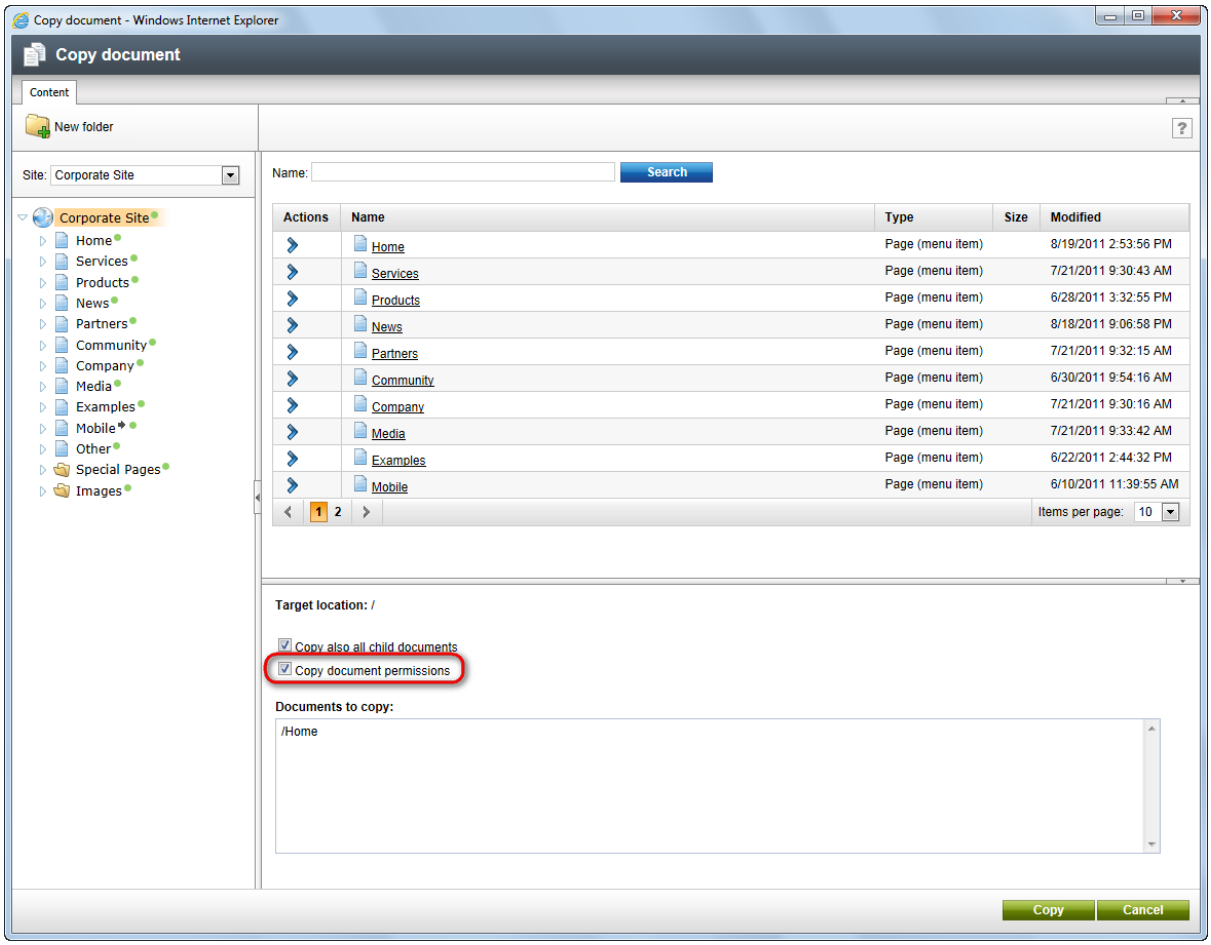
You may want to grant the following permissions to the users:

JohnS	**Marketing manager** John can manage all content.	Grant the **Full control** permission on the root to the user or grant permissions for the **CMS Content** module to some of this user's roles.
MarkJ	**Product manager** Mark can manage only the documents in the /Products section.	Grant the **Browse tree** permission on the root to the user so that they can browse the Products section. Grant the **Read**, **Modify**, **Create**, **Delete**, **Destroy** and **Browse tree** permissions on the /Products document to the user. These permissions are inherited by all child documents under the /Products section. Please note that if you click the /Products/Category 1 document, the Browse tree permission is grayed and disabled. It means that this **permission is inherited** and cannot be removed - you can only **deny** the permission (unless you break inheritance - see below).
AliceM	**Copy writer** Alice can modify the copy of all documents, but Mark prefers to manage the copy of the /Products section by himself only.	Grant the **Read**, **Modify**, **Create**, **Delete** and **Browse tree** permissions for the root to the user. Go to the /Products document and **deny** the **Modify**, **Create**, **Delete** permissions to the user so that Alice cannot modify the copy in the /Products section.

Please note: It's recommended that you configure local permissions for roles and then only assign users to the appropriate roles. In this example, you would first create roles "Marketing manager", "Product manager" and "Copy writer" and then configure their permissions.

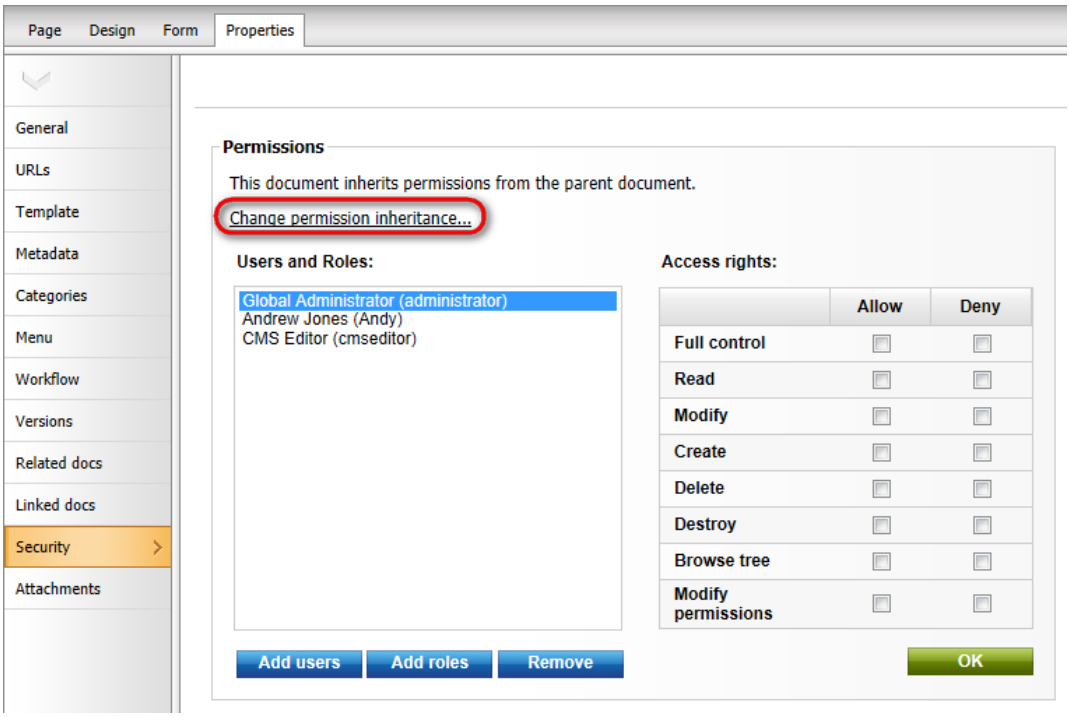
Copying permissions along with documents

If you copy, move or link a document, its permissions can be transferred along with it. You only need to enable the **Copy/Preserve document permissions** option in the **Copy/move/link document** dialog. This applies only to permissions configured for the particular document - parent or inherited permissions are not transferred. If you leave the option disabled, the copy will inherit permissions from its parent in the target location.



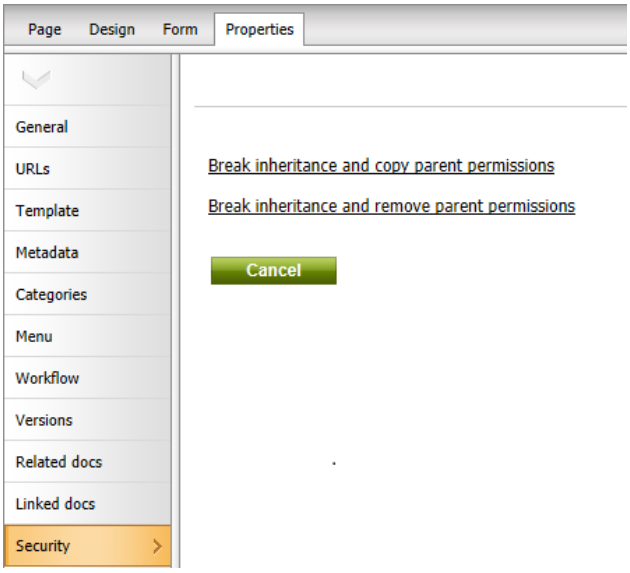
Changing permission inheritance

In case you need to break permission inheritance and configure different permissions for some site section, you need to click **Change permission inheritance...** link in the **Security** dialog.



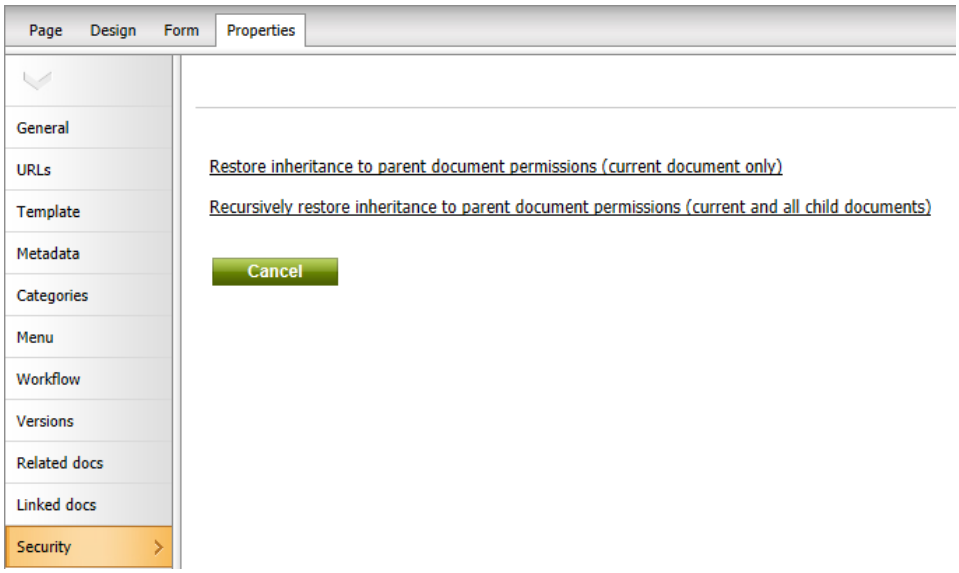
If permissions are inherited by the current document, the following two options will be offered:

- **Break inheritance and copy parent permissions** - breaks inheritance and adds parent permissions to the document, while original permissions configured for the document are preserved.
- **Break inheritance and remove parent permissions** - breaks inheritance and removes all permissions inherited from the parent, while additional permissions configured for the document are preserved.



If you decide to inherit the permissions from the parent again, click the **Change permission inheritance...** link again. This time, the following two options will be offered:

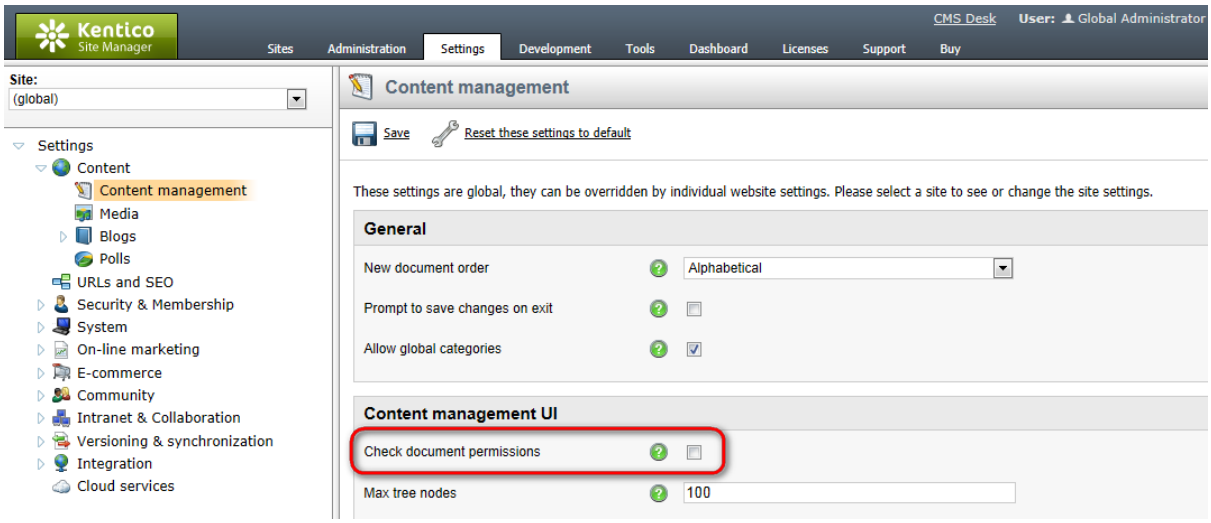
- **Restore inheritance to parent document permissions (current document only)** - makes the current document inherit permissions of the parent document.
- **Recursively restore inheritance to parent document permissions (current and all child documents)** - makes the current document and all its child documents inherit permissions of the parent document, while only documents which do not inherit parent permissions are affected by this action.



7.13.5.2.5 Hiding documents in the content tree based on permissions

If you enable the **Check document permissions** option in **Site Manager -> Settings -> Content -> Content management**, each user will only be able to see the documents for which they have the **Read** permission assigned on any of the three levels. This applies in **CMS Desk -> Content,** in document listings and in various dialogs where the content tree is displayed.

If a user has the **Read** permission for a child document but not for its parent, the child document is not displayed as well. In case that a user doesn't have the **Read** permissions for any document at all, a message is displayed instead of the content tree, saying that you have insufficient permissions to see the root document.



7.13.5.2.6 Document permisssions internals and API

7.13.5.2.6.1 Overview

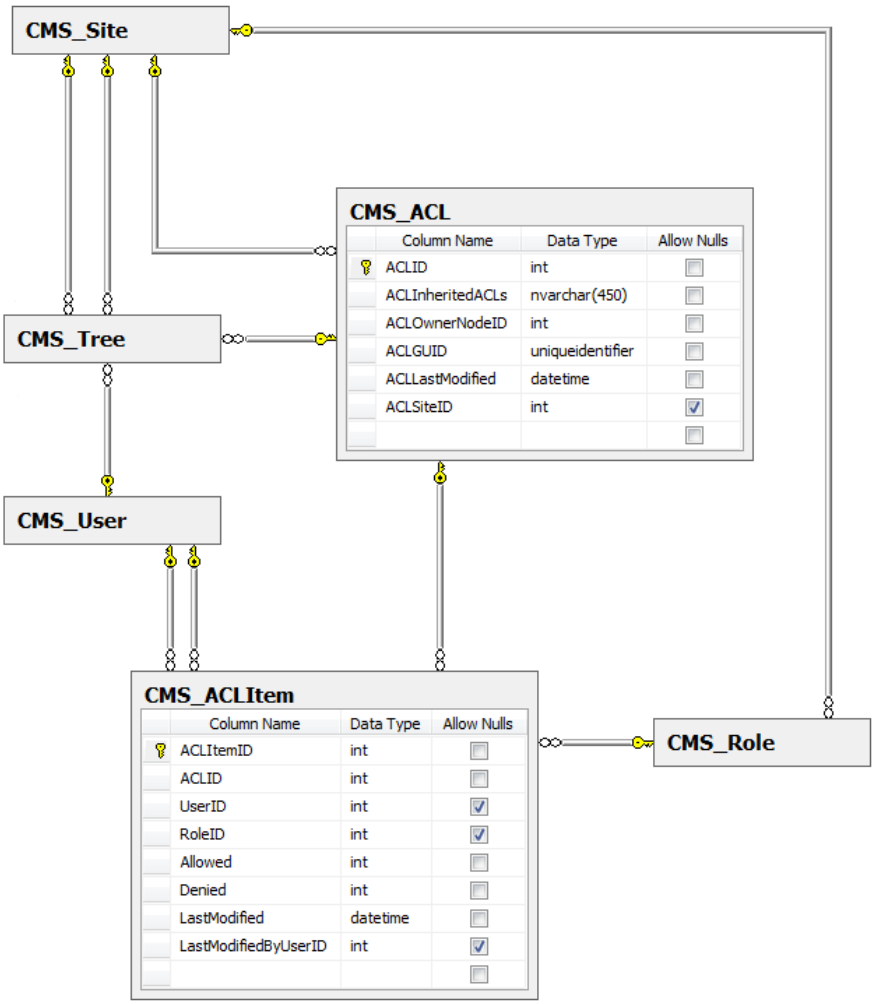
In this chapter, you will learn which database tables and API classes are used for managing document-level permissions. You will also see the most common API examples.

Advanced API examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all methods from the used classes, please refer to Kentico CMS API Reference.

7.13.5.2.6.2 Database tables

The following database tables are used to store document level permissions:

- **CMS_ACL** - contains records representing ACLs (access control lists), i.e. document level permission matrices of single documents.
- **CMS_ACLItem** - contains records representing document level permission settings for individual users or roles within particular ACLs.



7.13.5.2.6.3 API classes

If you are not familiar with database table data management through Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

The classes used for document permissions can be found in the **CMS.DocumentEngine** namespace.

CMS_ACL table API:

- **AclInfo** - represents one document level permission matrix.
- **AclProvider** - provides management functionality for document level permission.

7.13.5.2.6.4 API examples

These topics show examples of how the document permissions API can be used:

- Preparing document structure
- Settings document level permissions
- Managing permission inheritance
- Checking permissions

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Documents\Security\Default.aspx.cs***.

The document permissions API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.CMSHelper;
using CMS.GlobalHelper;
using CMS.SettingsProvider;
using CMS.SiteProvider;
using CMS.DocumentEngine;
using CMS.UIControls;
```

The following example is not directly related to document permissions. It only prepares a sample document structure used by the API examples on the following pages.

```
private bool CreateDocumentStructure()
{
    // Create new instance of the Tree provider
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get default culture code
    string culture = SettingsKeyProvider.GetStringValue
(CMSContext.CurrentSiteName + ".CMSDefaultCultureCode");

    // Get parent node
    TreeNode parentNode = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/",
culture);

    if (parentNode != null)
    {
        // Create the API Example document
```

```
        TreeNode newNode = TreeNode.New("CMS.MenuItem", tree);

        newNode.DocumentName = "API Example";
        newNode.DocumentCulture = culture;

        newNode.Insert(parentNode);

        parentNode = newNode;

        // Create the API Example subpage
        newNode = TreeNode.New("CMS.MenuItem", tree);

        newNode.DocumentName = "API Example subpage";
        newNode.DocumentCulture = culture;

        newNode.Insert(parentNode);

        return true;
    }

    return false;
}
```

The following example deletes the sample document structure created by the API example above.

```
private bool DeleteDocumentStructure()
{
    // Create an instance of the Tree provider
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get default culture code
    string culture = SettingsKeyProvider.GetStringValue
(CMSContext.CurrentSiteName + ".CMSDefaultCultureCode");


    // Get the API Example document
    TreeNode node = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/API-
Example", culture);

    if (node != null)
    {
        // Delete the document and all child documents
        node.DeleteAllCultures();
    }

    return true;
}
```

The following example grants the **Modify permissions** permission to the **CMSEditor** user on document level for a single document.

```
private bool SetUserPermissions()
{
    // Create an instance of the Tree provider
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get default culture code
    string culture = SettingsKeyProvider.GetStringValue
(CMSContext.CurrentSiteName + ".CMSDefaultCultureCode");

    // Get the API Example document
    TreeNode node = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/API-
Example", culture);

    if (node != null)
    {
        // Get the user
        UserInfo user = UserInfoProvider.GetUserInfo("CMSEditor");

        if (user != null)
        {
            // Prepare allowed / denied permissions
            int allowed = 0;
            int denied = 0;
            allowed += Convert.ToInt32(Math.Pow(2, Convert.ToInt32
(NodePermissionsEnum.ModifyPermissions)));

            // Create an instance of ACL provider
            AclProvider acl = new AclProvider(tree);

            // Set user permissions
            acl.SetUserPermissions(node, allowed, denied, user);

            return true;
        }
    }

    return false;
}
```

The following example grants the **Modify** permission to the **CMSEditor** role on document level for a single document.

```
private bool SetRolePermissions()
{
    // Create an instance of the Tree provider
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get default culture code
    string culture = SettingsKeyProvider.GetStringValue
(CMSContext.CurrentSiteName + ".CMSDefaultCultureCode");

    // Get the API Example document
```

```
    TreeNode node = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/API-
Example", culture);

    if (node != null)
    {
        // Get the role ID
        RoleInfo role = RoleInfoProvider.GetRoleInfo("CMSEditor",
CMSContext.CurrentSiteName);

        if (role != null)
        {
            // Prepare allowed / denied permissions
            int allowed = 0;
            int denied = 0;
            allowed += Convert.ToInt32(Math.Pow(2, Convert.ToInt32
(NodePermissionsEnum.Modify)));

            // Create an instance of ACL provider
            AclProvider acl = new AclProvider(tree);

            // Set role permissions
            acl.SetRolePermissions(node, allowed, denied, role.RoleID);

            return true;
        }
    }

    return false;
}
```

The following example removes all document level permissions granted to both users and roles on document level for a single document.

```
private bool DeletePermissions()
{
    // Create an instance of the Tree provider
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get default culture code
    string culture = SettingsKeyProvider.GetStringValue
(CMSContext.CurrentSiteName + ".CMSDefaultCultureCode");

    // Get the API Example document
    TreeNode node = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/API-
Example", culture);

    if (node != null)
    {
        // Create an instance of ACL provider
        AclProvider acl = new AclProvider(tree);

        // Get ID of ACL used on API Example document
```

```
        int nodeACLID = ValidationHelper.GetInteger(node.GetValue("NodeACLID"),
0);

        // Delete all ACL items
        acl.ClearACLItems(nodeACLID);

        return true;
    }

    return false;
}
```

The following example breaks permission inheritance on document level of a single document.

```
private bool BreakPermissionInheritance()
{
    // Create an instance of the Tree provider
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get default culture code
    string culture = SettingsKeyProvider.GetStringValue
(CMSContext.CurrentSiteName + ".CMSDefaultCultureCode");

    // Get the API Example document
    TreeNode node = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/API-
Example/API-Example-subpage", culture);

    if (node != null)
    {
        // Create an instance of ACL provider
        AclProvider acl = new AclProvider(tree);

        // Break permission inheritance (without copying parent permissions)
        bool copyParentPermissions = false;
        acl.BreakInherintance(node, copyParentPermissions);

        return true;
    }

    return false;
}
```

The following example restores permission inheritance on document level of a single document.

```
private bool RestorePermissionInheritance()
{
    // Create an instance of the Tree provider
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get default culture code
```

```
    string culture = SettingsKeyProvider.GetStringValue
(CMSContext.CurrentSiteName + ".CMSDefaultCultureCode");

    // Get the API Example document
    TreeNode node = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/API-
Example/API-Example-subpage", culture);

    if (node != null)
    {
        // Create an instance of ACL provider
        AclProvider acl = new AclProvider(tree);

        // Restore permission inheritance
        acl.RestoreInheritance(node);

        return true;
    }

    return false;
}
```

The following example checks if the **CMSEditor** user has the **Read** permission for the **Content** module.

```
private bool CheckContentModulePermissions()
{
    // Get the user
    UserInfo user = UserInfoProvider.GetUserInfo("CMSEditor");

    if (user != null)
    {
        // Check permissions and perform an action according to the result
        if (UserInfoProvider.IsAuthorizedPerResource("CMS.Content", "Read",
CMSContext.CurrentSiteName, user))
        {

            apiCheckContentModulePermissions.InfoMessage = "User 'CMSEditor' is
allowed to read module 'Content'.";
        }
        else
        {
            apiCheckContentModulePermissions.InfoMessage = "User 'CMSEditor' is
not allowed to read module 'Content'.";
        }

        return true;
    }

    return false;
}
```

The following example checks if the **CMSEditor** user has the **Read** permission for the **CMS.MenuItem**

document type.

```
private bool CheckDocTypePermissions()
{
    // Get the user
    UserInfo user = UserInfoProvider.GetUserInfo("CMSEditor");

    if (user != null)
    {
        // Check permissions and perform an action according to the result
        if (UserInfoProvider.IsAuthorizedPerClass("CMS.MenuItem", "Read",
CMSContext.CurrentSiteName, user))
        {

            apiCheckDocTypePermissions.InfoMessage = "User 'CMSEditor' is allowed
to read document type 'MenuItem'.";
        }
        else
        {
            apiCheckDocTypePermissions.InfoMessage = "User 'CMSEditor' is not
allowed to read document type 'MenuItem'.";
        }

        return true;
    }

    return false;
}
```

The following example checks if the **CMSEditor** user has the **Read** permission on document level for a single document.

```
private bool CheckDocumentPermissions()
{
    // Create an instance of the Tree provider
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get default culture code
    string culture = SettingsKeyProvider.GetStringValue
(CMSContext.CurrentSiteName + ".CMSDefaultCultureCode");

    // Get the API Example document
    TreeNode node = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/API-
Example", culture);

    if (node != null)
    {
        // Get the user
        UserInfo user = UserInfoProvider.GetUserInfo("CMSEditor");

        if (user != null)
        {
```

```
            // Check permissions and perform an action according to the result
            if (TreeSecurityProvider.IsAuthorizedPerNode(node,
NodePermissionsEnum.ModifyPermissions, user) == AuthorizationResultEnum.Allowed)
            {
                apiCheckDocumentPermissions.InfoMessage = "User 'CMSEditor' is
allowed to modify permissions for document 'API Example'.";
            }
            else
            {
                apiCheckDocumentPermissions.InfoMessage = "User 'CMSEditor' is not
allowed to modify permissions for document 'API Example'.";
            }

            return true;
        }
    }

    return false;
}
```

The following example gets multiple document into a DataSet, filters the documents depending on if the **Modify permissions** permission was granted to the **CMS Editor** user on their document level, and breaks permission inheritance of the filtered documents.

```
private bool FilterDataSet()
{
    // Create an instance of the Tree provider
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Set the parameters for getting documents
    string siteName = CMSContext.CurrentSiteName;
    string aliasPath = "/%";
    string culture = SettingsKeyProvider.GetStringValue
(CMSContext.CurrentSiteName + ".CMSDefaultCultureCode");
    bool combineWithDefaultCulture = true;

    // Get data set with documents
    DataSet documents = tree.SelectNodes(siteName, aliasPath, culture,
combineWithDefaultCulture);

    // Get the user
    UserInfo user = UserInfoProvider.GetUserInfo("CMSEditor");

    if (user != null)
    {
        // Filter the data set by the user permissions
        TreeSecurityProvider.FilterDataSetByPermissions(documents,
NodePermissionsEnum.ModifyPermissions, user);

        if (!DataHelper.DataSourceIsEmpty(documents))
        {
            // Create an instance of ACL provider
```

```
            AclProvider acl = new AclProvider(tree);

            // Loop through filtered documents
            foreach (DataRow documentRow in documents.Tables[0].Rows)
            {
                // Create a new Tree node from the data row
                TreeNode node = TreeNode.New(documentRow, "CMS.MenuItem", tree);

                // Break permission inheritance (with copying parent permissions)
                acl.BreakInherintance(node, true);
            }

            // Data set filtered successfully - permission inheritance broken for
filtered items
            apiFilterDataSet.InfoMessage = "Data set with all documents filtered
successfully by permission 'Modify permissions' for user 'CMSEditor'. Permission
inheritance broken for filtered items.";
        }
        else
        {
            // Data set filtered successfully - no items left in data set
            apiFilterDataSet.InfoMessage = "Data set with all documents filtered
successfully by permission 'Modify permissions' for user 'CMSEditor'. No items
left in data set.";
        }

        return true;
    }

    return false;
}
```

7.13.5.3 Design permissions

The Design permission matrix is used to set up important permissions related to the design of page templates and their components.

To access the matrix in **Site manager**:

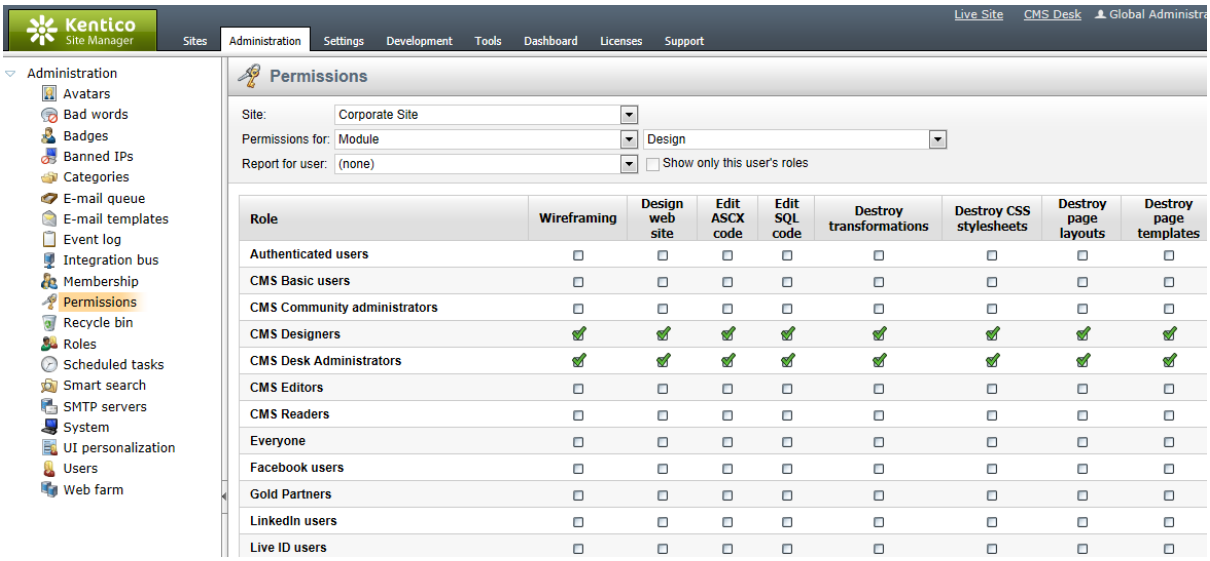
1. Navigate to **Administration -> Permissions**.

2. In the **Site** drop-down list, choose the site you want to configure the permissions, e.g., for Corporate site.

3. Choose **Module** in the first and **Design** in the second **Permissions for** drop-down list.

You can also choose for which user you want a permission report to be shown by using the **Report for user** drop-down list.

In **CMS Desk**, you can access the matrix for a given site by going to **Administration -> Permissions**.



You can assign the following permissions to members of the specified roles:

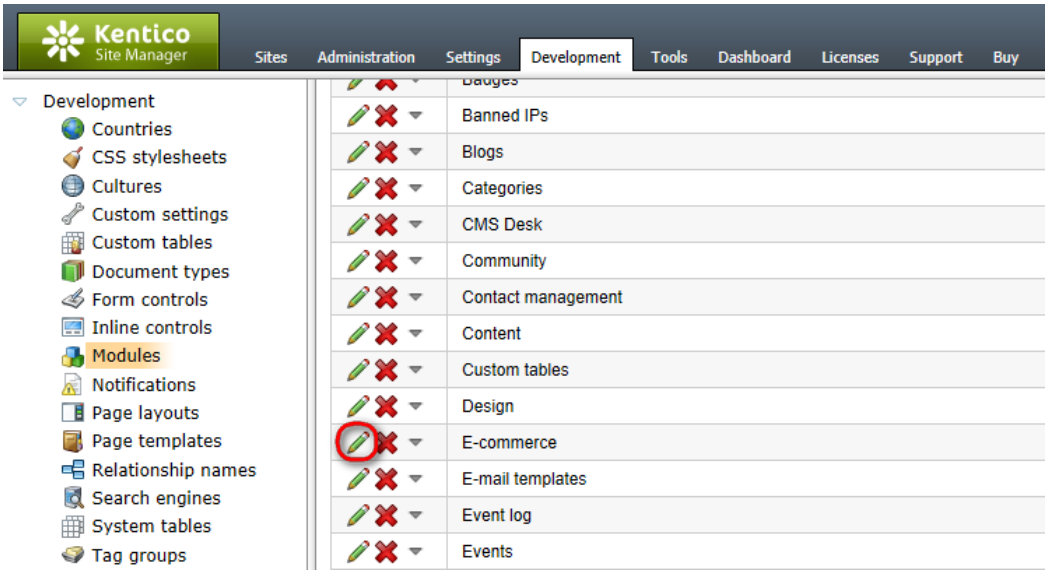
Wireframing	Allows users to edit the content of wireframe schematics in **CMS Desk -> Content** on the **Wireframe** tab and create new wireframe documents. Users without this permission can view wireframes, but are not allowed to make any modifications.
Design website	Allows users to edit documents on the **Design** tab of **CMS Desk**. Note that even though the changes on the **Design tab** are made on a specific website, this may also affect other sites in the system if they use the same shared page template. This permission also determines whether a given role is allowed to configure the properties of web parts through the **On-site editing** interface.
Edit ASCX code	Allows users to modify the ASCX code of page layouts and transformations. This permission does not affect the ability to edit the HTML versions of these objects. This should be considered a high-level permission, because it gives users the option to add and execute inline code.
Edit SQL code	Allows users to create or modify query objects and edit other fields containing SQL code, such as the WHERE condition properties of web parts. This should be considered a high-level permission, because it gives users the power to write and execute SQL queries against the website's database.
Destroy transformations	Allows users to delete the version history of transformation objects.
Destroy CSS stylesheets	Allows users to delete the version history of CSS stylesheet objects.
Destroy page layouts	Allows users to delete the version history of shared page layouts.
Destroy page templates	Allows users to delete the version history of page templates.
Destroy web part containers	Allows users to delete the version history of web part containers.

Destroy web part layouts	Allows users to delete the version history of web part layouts.

For security reasons, the **Edit ASCX code** and **Edit SQL code** permissions may only be assigned by users designated as global administrators.

7.13.5.4 Module permissions management

Module permissions can be managed in the administration interface of each particular module. This is accessible after clicking the **Edit** (✎) icon next to the required module in **Site Manager -> Development -> Modules**.



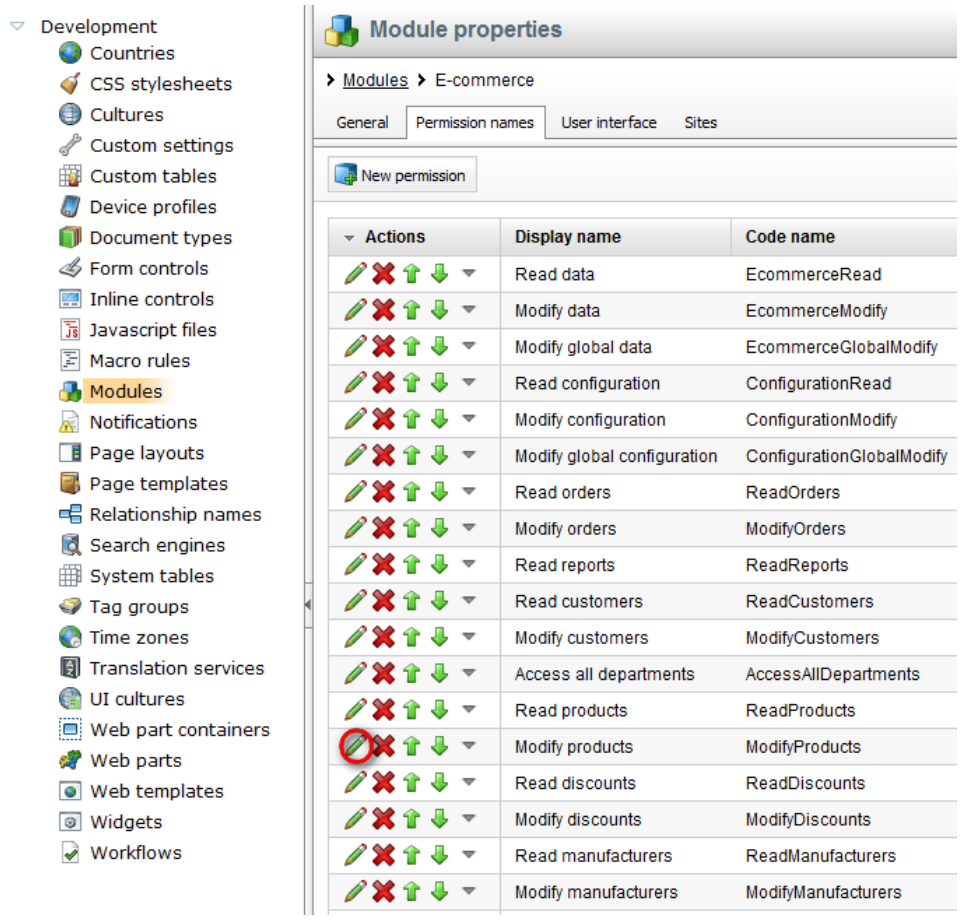
In the module's administration UI, switch to the **Permission names** tab. Here, you can see a list of all permissions defined for the module. They are listed in the same order that will be used in the permission matrix (top-down order here equals to left-right order in the matrix). The **Move up** (↑) and **Move down** (↓) icons can be used to re-order permissions according to your needs.

Please note

New permissions can be created by clicking **New permission** above the list. However, each permission must be handled by the code of the module to actually take effect. Creating a permission in this UI on its own does not have any effects.

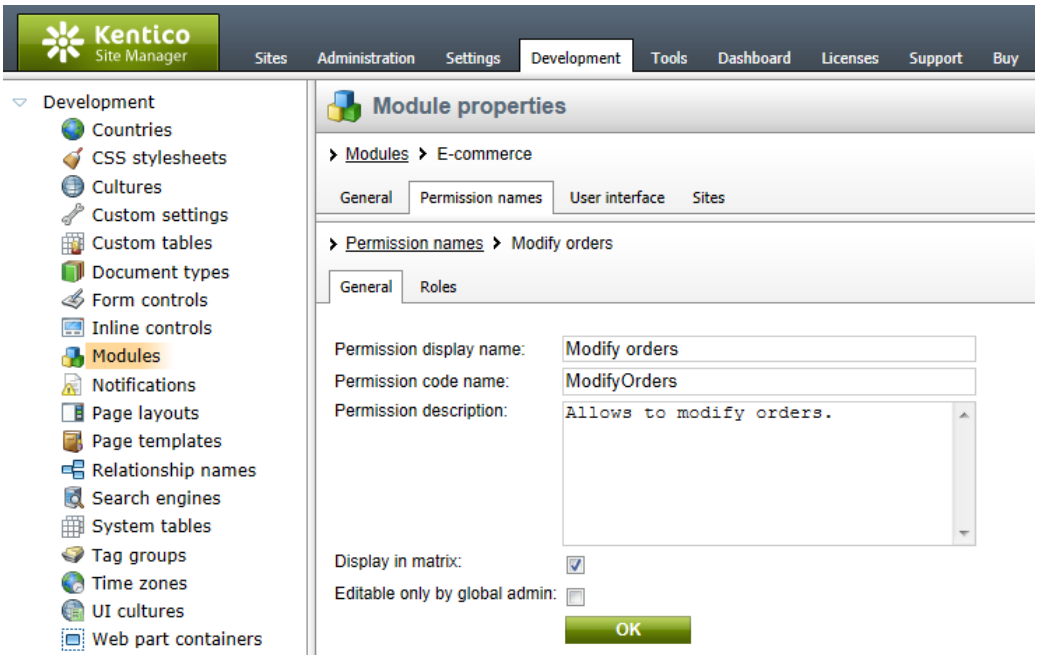
An example of handling permissions in custom module code can be found in the Modules -> Developing custom modules topic.

To adjust configuration of an existing permission, click the **Edit** (✎) icon next to it.



The following properties of each permission can be configured on the **General** tab of its editing interface.

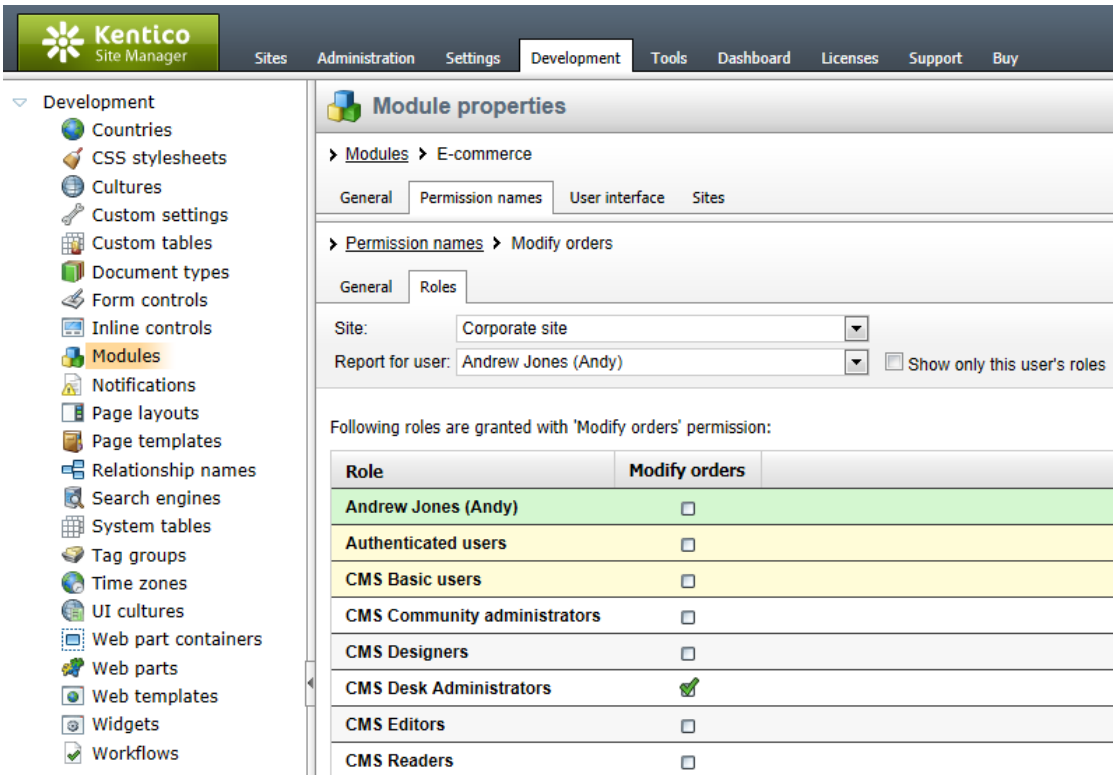
- **Permission display name** - name of the permission displayed in Kentico CMS UI.
- **Permission code name** - name of the permission used in website code.
- **Permission description** - text providing more details about the permission (typically where and when it is checked).
- **Display in matrix** - indicates if the permission should be displayed in the module's permissions matrix.
- **Editable only by global admin** - if enabled, the permission can only be granted to users or roles by the global administrator.



Direct assigning of permissions to roles

To assign the currently edited permission to particular roles conveniently without going to the **Administration -> Permissions** section, you can switch to the **Roles** tab of a permission's editing interface. Then select the site from the **Site** drop-down list and assign the permission to given roles using the check-boxes.

To check if the permission is granted to a user, you can select the user from the **Report for user** drop-down list. After doing so, a sum of all permissions granted to the user's roles is displayed in the first line, highlighted in green color. Roles where the selected user is a member will be highlighted in yellow color. If you enable the **Show only this user's roles** check-box, only the yellow roles will be displayed in the matrix.



7.13.6 UI personalization

7.13.6.1 Overview

UI personalization enables you to provide certain users of the website with **simplified user interface**. This is useful **typically for business users** who don't need to see all the **tabs**, **menu items**, **actions** and **parts of UI pages** which they don't use. Instead, they have the possibility to see only the things that they need for their job and are not overwhelmed by loads of other options.

Setting up personalized UI can significantly decrease the learning time for new end-users and makes the system generally easier to use and understand for them.

If you are new to UI personalization in Kentico CMS, we recommend you to take the following steps to fully understand it:

1. For UI personalization to be functional, you first need to enable it as described here.

2. It is a good idea to start with the Quick example topic, where you can instantly see what UI personalization is good for.

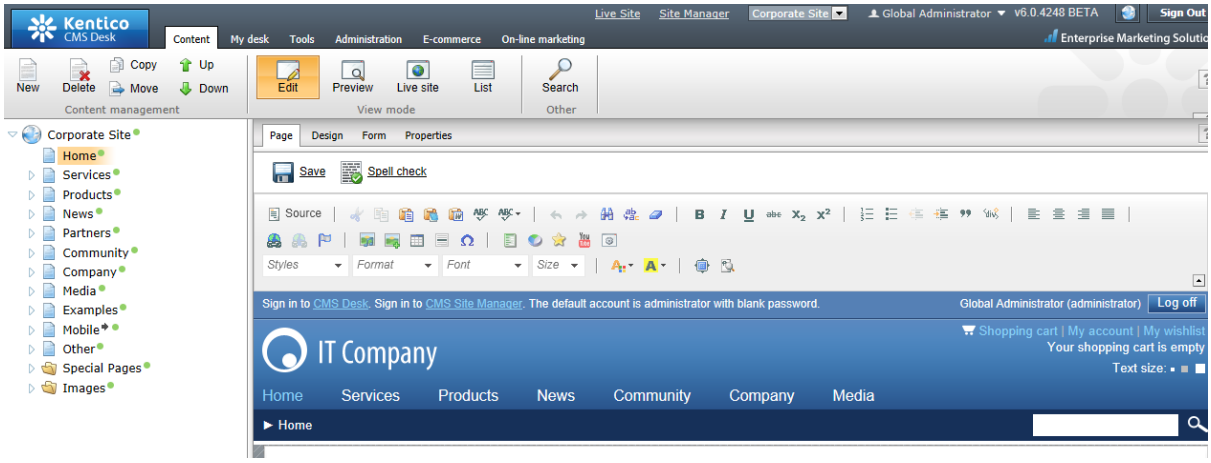
3. With some basic information from the Quick example, you can get deeper knowledge of the terminology, concept and internals in the How it works chapter.

4. Finally, you can learn what parts of CMS Desk can be hidden in Personalizable parts of CMS Desk.

With the knowledge gained in the chapters above, you can start making UI personalization settings for particular roles as described in the UI profile configuration topic. You can also modify or even create your custom UI elements as described in the UI elements management and the related example topics.

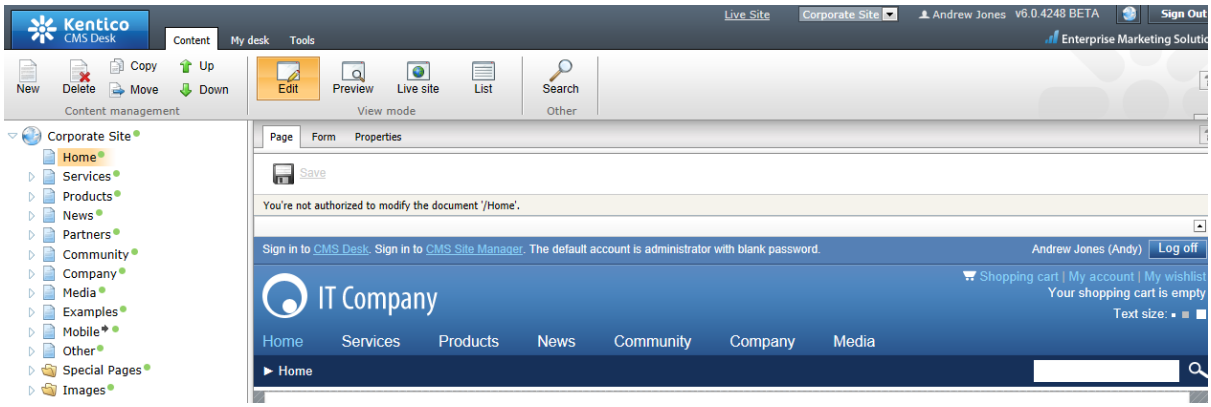
7.13.6.2 Quick example

To see UI personalization in practice, first enable UI personalization as described here. Install one of the sample sites (e.g. Corporate Site) and try logging in to CMS Desk as the **Global Administrator** (login *administrator* with blank password). You will see the full-featured user interface as in the screenshot below.

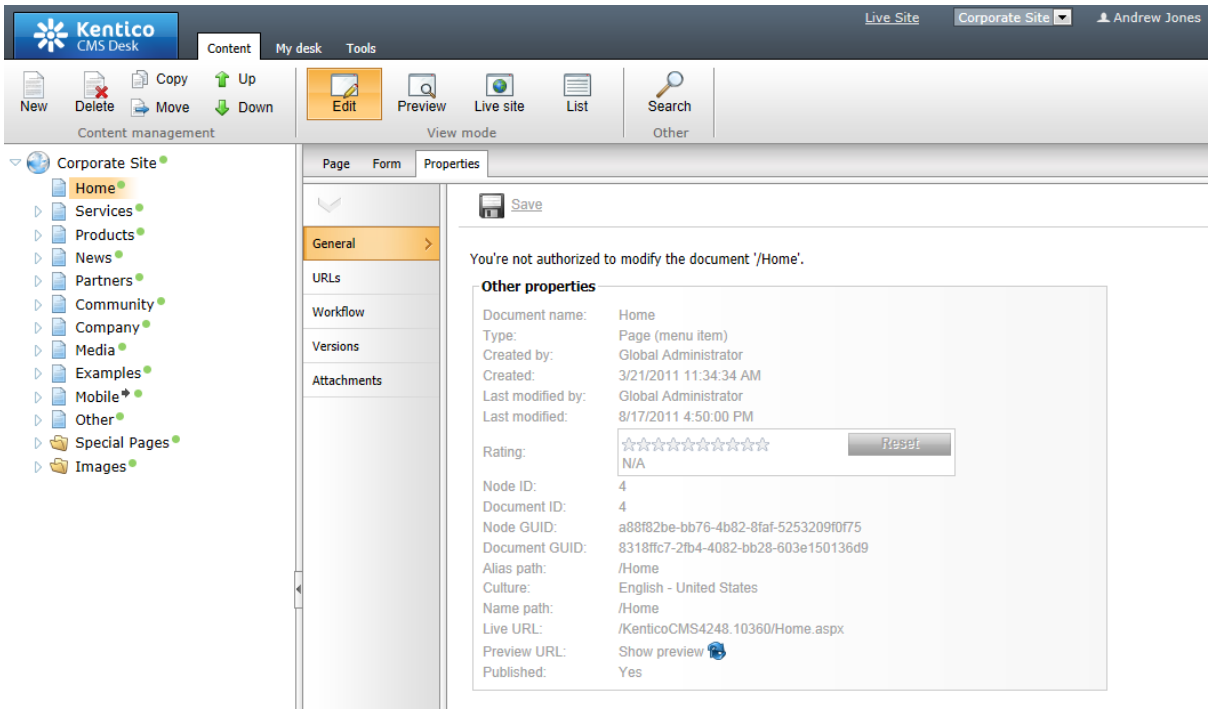


Now log out and try logging in as **Andrew Jones** (login *andy* with blank password). Andy is a member of the **CMS Basic users** role only. This is a role which was added to Kentico CMS to demonstrate the capabilities of UI personalization and its members see a simplified user interface.

As you can see in the screenshot below, the **Administration** tab in the top menu is not present. The **Design** tab in **Edit** mode is missing too. You can also notice the **WYSIWYG editor**, which offers only a limited number of actions.

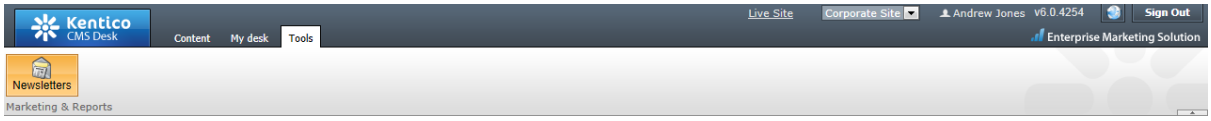


If you switch to the **Properties** tab, you can notice the limited number of items in the left menu. Particular sections accessible via the menu also do not contain all the options available for administrators.



The **My desk** tab contains only three items.

The **Tools** tabs offers only the **Newsletters** module.



It is obvious that this UI is much easier to understand for an end user who only needs to send out newsletters and does not need to do anything else with the CMS.

To see a full overview of which parts of the UI can be hidden, please see Personalizable parts of CMS Desk.

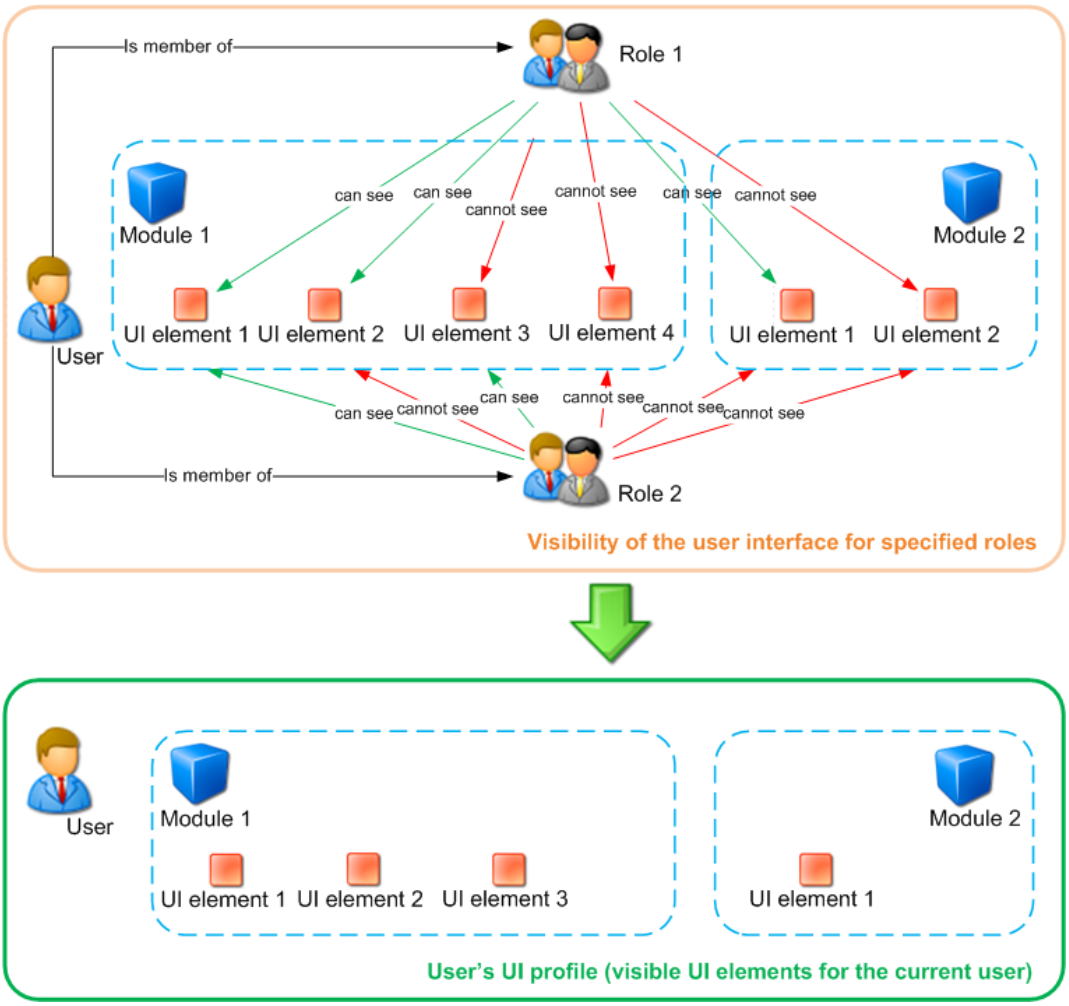
7.13.6.3 How it works

This chapter explains the principles behind Kentico CMS UI personalization. It is important to get familiar with the following two terms in order to fully understand the concept:

- **UI element** - page or part of a page in CMS Desk which can be hidden for users belonging to specified roles. It can be a *tab*, a *menu item* or a *group of controls*.
- **UI profile** - visibility settings of specified UI elements for a particular role. It is defined as a set of (role) x (UI element) relationships.

Each user's UI profile is defined by UI profiles of their roles. If a user wants to see a UI element, at least one role where the user is a member needs to have the UI element set as visible.

The following diagram illustrates how UI personalization settings from two roles are merged to create a user's UI profile.



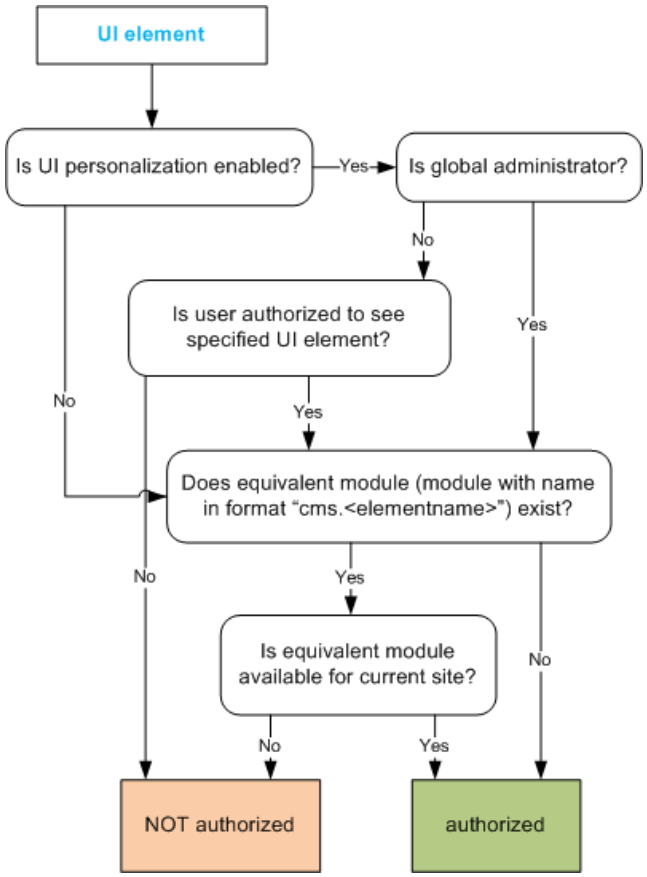
Please note: The roles assigned to a user determine which UI profile will be displayed. A user who is assigned to site-specific roles will have a **site-related** UI profile. This means that a user can see a certain personalized UI when editing one site and a completely different UI when editing another site. On the other hand, if a user is a member of only global roles, the same UI profile will be displayed to them on all sites.

UI element display authorization

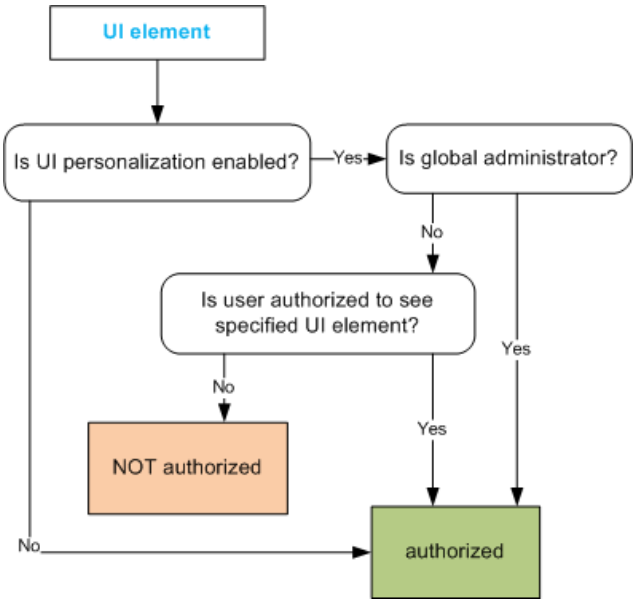
The diagram below shows the process of authorization when deciding if a single UI element should be

displayed to a user. This process is applied to UI elements in the following parts of the UI:

- main tabs in CMS Desk (Content, My Desk, Tools, Administration - only the tabs, not their content!)
- left menu in CMS Desk -> My Desk
- left menu in CMS Desk -> Tools
- left menu in CMS Desk -> Administration



This diagram shows the same authorization process in the remaining tabs and parts of pages:



UI element definitions vs. UI elements in the real user interface

The relation between the UI element definition in **Site Manager -> Modules -> edit ( ) a module -> User interface** and the UI elements in the real UI is different according to the type of UI element.

There are three basic types of UI elements: **tabs**, **menu items** and **groups of controls**.

Tabs in the following locations are **generated dynamically** from the defined UI elements, which means that when a new UI element is defined, the element gets instantly displayed in the UI:

- main tabs in CMS Desk (Content, My desk, ...)
- main tabs in Site Manager (Sites, Administration, Settings, ...)
- tabs in CMS Desk -> Edit (Page, Design, ...)
- vertical tabs in CMS Desk -> Edit -> Properties (General, Metadata, ...)
- tabs in CMS Desk -> My desk -> Account

Menu items in the following menus are also **generated dynamically**:

- CMS Desk -> My desk -> top menu
- CMS Desk -> Tools -> left menu
- CMS Desk -> Administration -> left menu

Groups of controls on particular UI pages (e.g. the Design, Other properties, Owner and Cache sections in CMS Desk -> Content -> Properties -> General) are **pre-defined** and their UI elements are bound to them. You cannot define a new group of controls on a page just by defining a new UI element.

More information related to this topic can be found in Personalizable parts of CMS Desk.

7.13.6.4 Personalizable parts of CMS Desk

7.13.6.4.1 Overview

UI personalization can be applied to **CMS Desk only**. Site Manager cannot be personalized as it is typically used by administrators and developers who need the full UI rather than a simplified version.

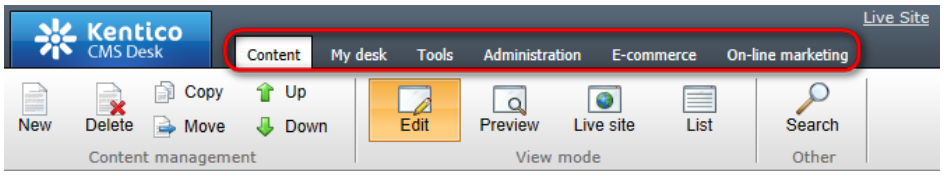
As for CMS Desk, you can personalize the UI elements in the following parts of CMS Desk. Module names in parentheses show which module represents these parts of CMS Desk:

- CMS Desk main tabs (module CMS Desk)
- CMS Desk -> Content tab (module Content)
- CMS Desk -> Content -> New (Module New)
- CMS Desk -> My desk tab (module My desk)
- CMS Desk -> Tools tab (module Tools)
- CMS Desk -> Administration tab (module Administration)
- WYSIWYG editor (module WYSIWYG editor)
- Media dialog (module Media dialog)

Click particular sections to learn more.

7.13.6.4.2 CMS Desk main tabs

By default, CMS Desk has the following tabs: **Content**, **My desk**, **Tools**, **Administration**, **E-commerce** and **On-line marketing** - as highlighted in this screenshot:



To add a tab to CMS Desk:

1. Navigate to **Site Manager -> Development -> Modules**.

2. Edit ( ) the **CMS Desk** module and click the **User interface** tab.

3. Click **New element** and type a **Display name**, **Caption** and a **Target URL** that should be loaded when a user clicks the new tab.

4. Click **Save**.

When you open CMS Desk, you can see the new tab added at the end of the main menu. Click the tab to load the page that you defined in Target URL.

To show or hide a CMS Desk tab to or from a particular role:

1. Navigate to **Site Manager -> Development -> Modules**.

2. Edit ( ) the **CMS Desk** module and click the **User interface** tab.

3. In the UI element tree, click the tab that you want to hide and then click **Roles**.

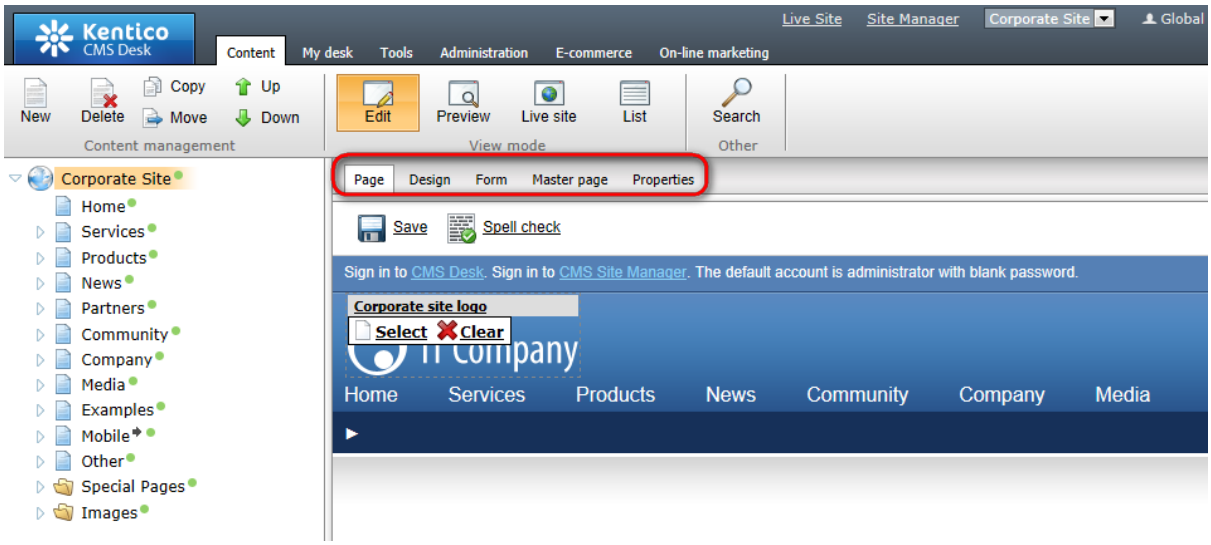
4. Select a site and check the boxes next to roles that you want to allow to see the tab, or uncheck the boxes next to roles that you want to hide the tab from.

7.13.6.4.3 CMS Desk -> Content tab

The **Content** tab provides vast possibilities of UI personalization. For easier explanation, we will divide them in **three groups**: **Edit mode** tabs, **Design** tab and **Properties** tab.

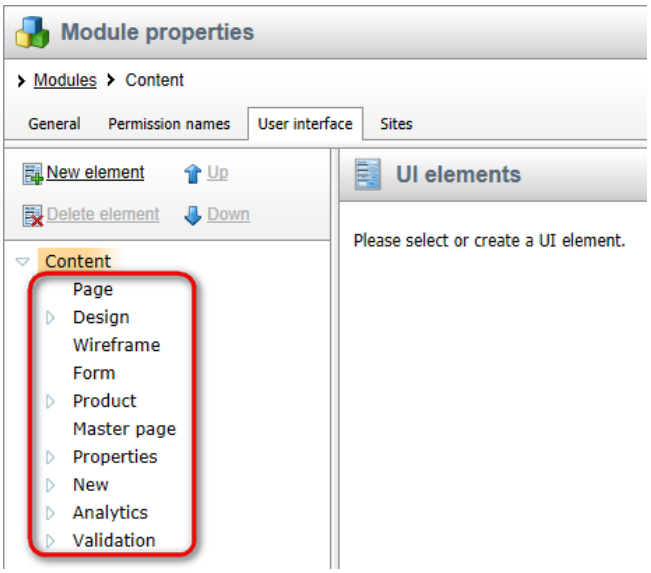
Edit mode tabs

You can hide any of the main tabs displayed in **Edit** mode as highlighted in the screenshot below.



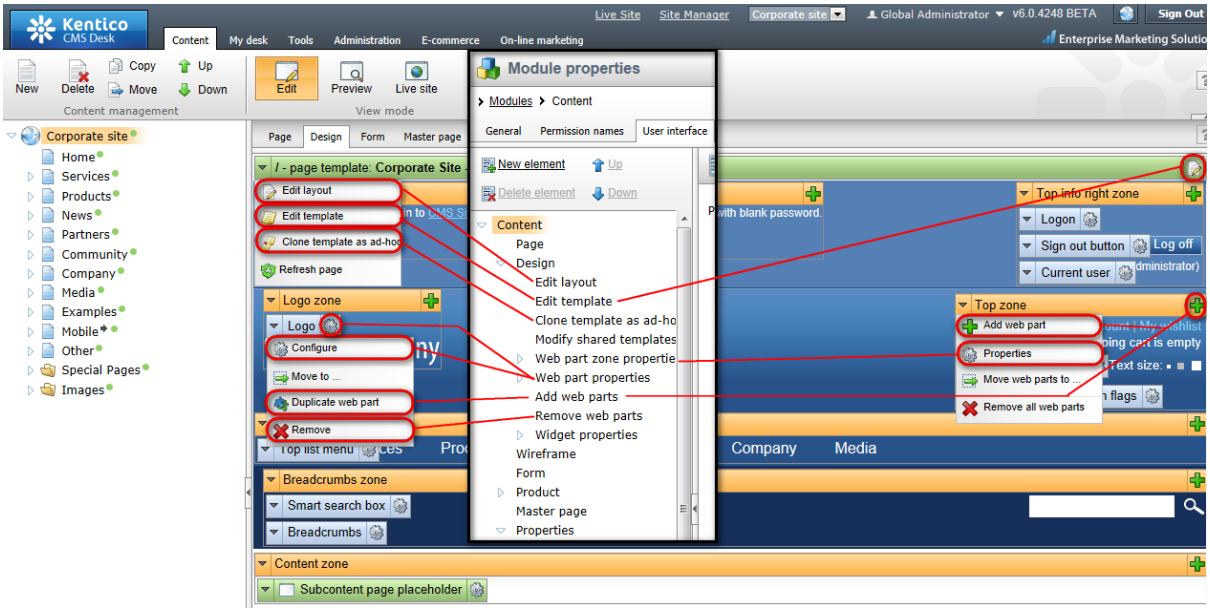
These main tabs are represented by the first-level UI elements of the **Content** module as highlighted below.

Please note that the **Product** tab is not displayed in the screenshot above as it is only displayed when a product type document is selected in the content tree. This is also the case of the **Master page** tab, which is displayed only when you are using Portal engine and the root of the content tree (the master page) is selected.



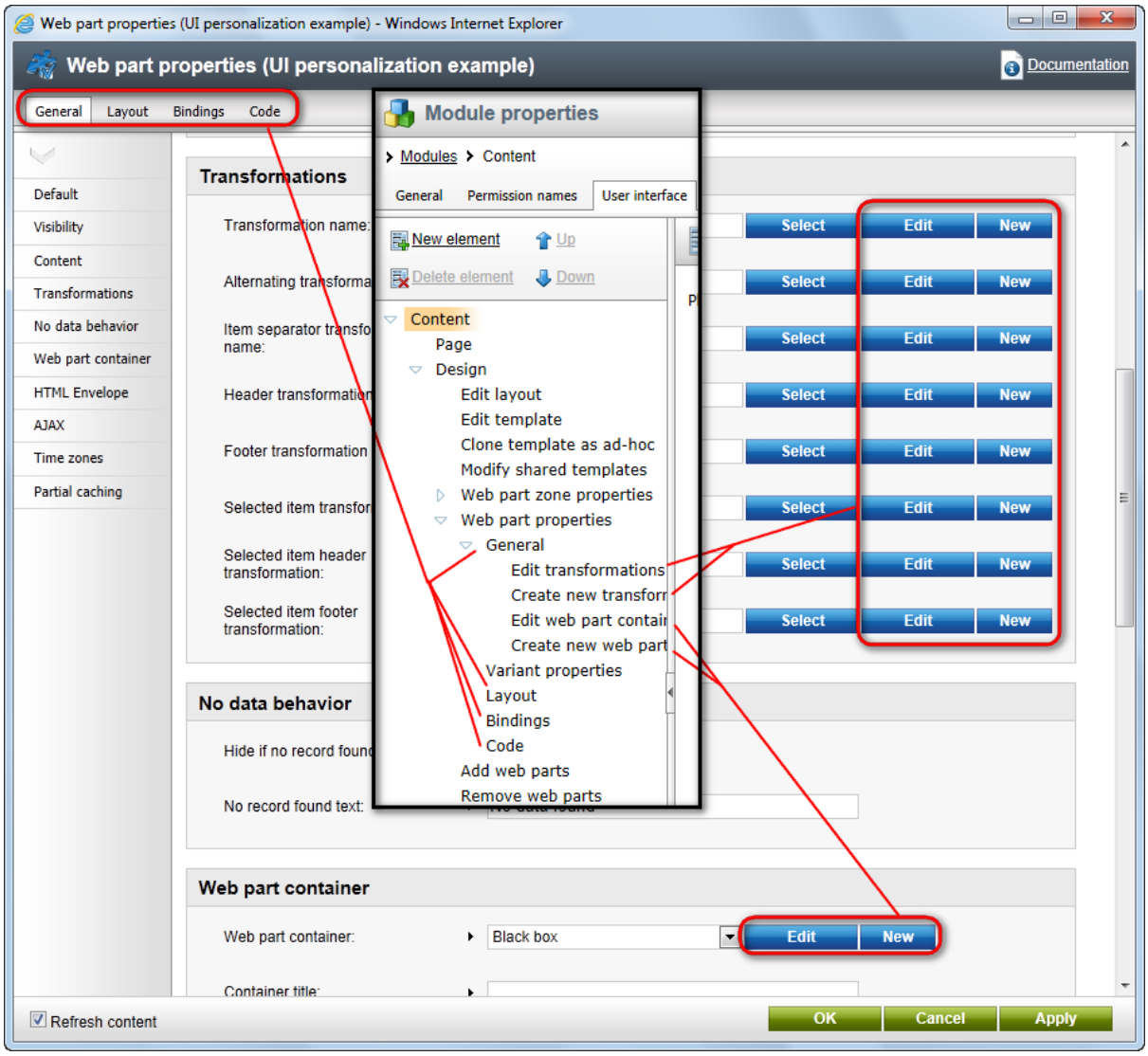
Design tab

On the **Design** tab, you customize all the essential editing actions highlighted in the screenshot below. The corresponding UI elements can be found under the **Design** node of the **Content** module's UI elements tree.



Web part properties dialog

You can hide the four main tabs - **General**, **Layout**, **Bindings** and **Code** - in the **Web part properties** dialog.



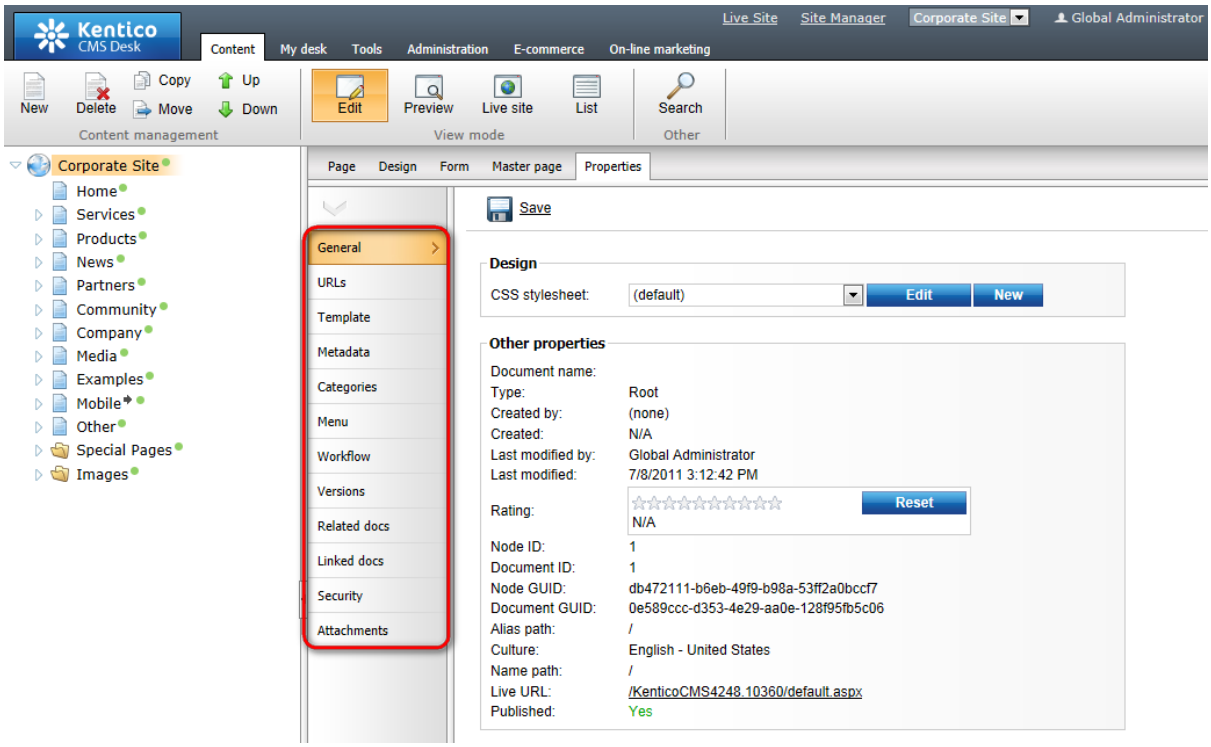
Properties tab

You can hide all items in the left menu on the **Properties** tab. You can also hide particular parts of the **General**, **URLs**, **Template**, **Metadata**, **Navigation** and **Security** pages. More information can be found on this page.

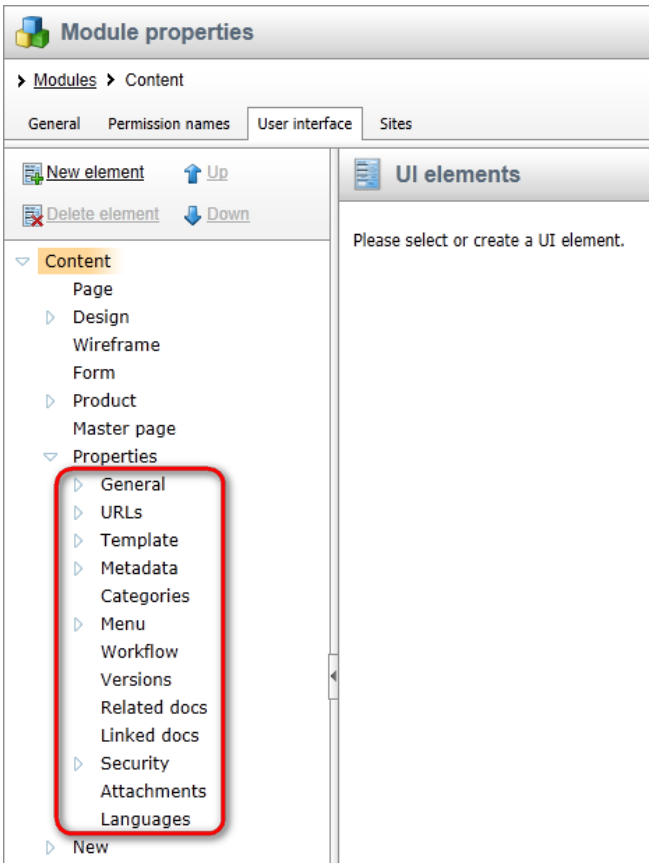
7.13.6.4.4 CMS Desk -> Content -> Edit -> Properites tab

This chapter describes UI personalization possibilities for the **Content -> Edit -> Properties** tab of a document.

First of all, you can hide all items in the left menu on the **Properties** tab as highlighted in the screenshot below.



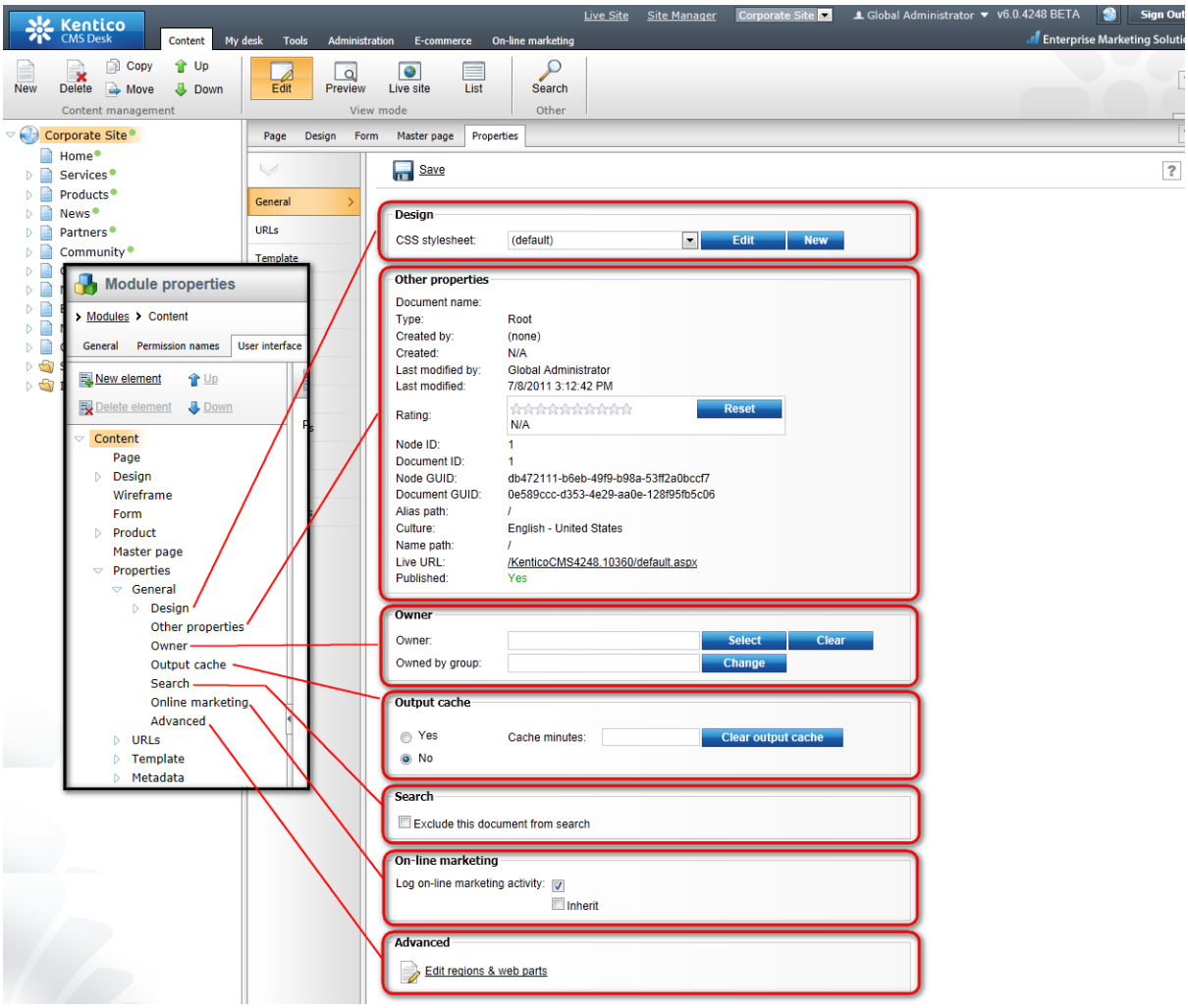
In the **Content** module, you can find the relevant UI elements under the **Properties** node. These elements have the same names as the names of particular menu items.



You can also expand the **General**, **URLs**, **Template**, **Metadata**, **Navigation** and **Security** UI elements using the ⊞ icon in order to display their child UI elements. These child UI elements represent parts of the relevant pages as described in the text below:

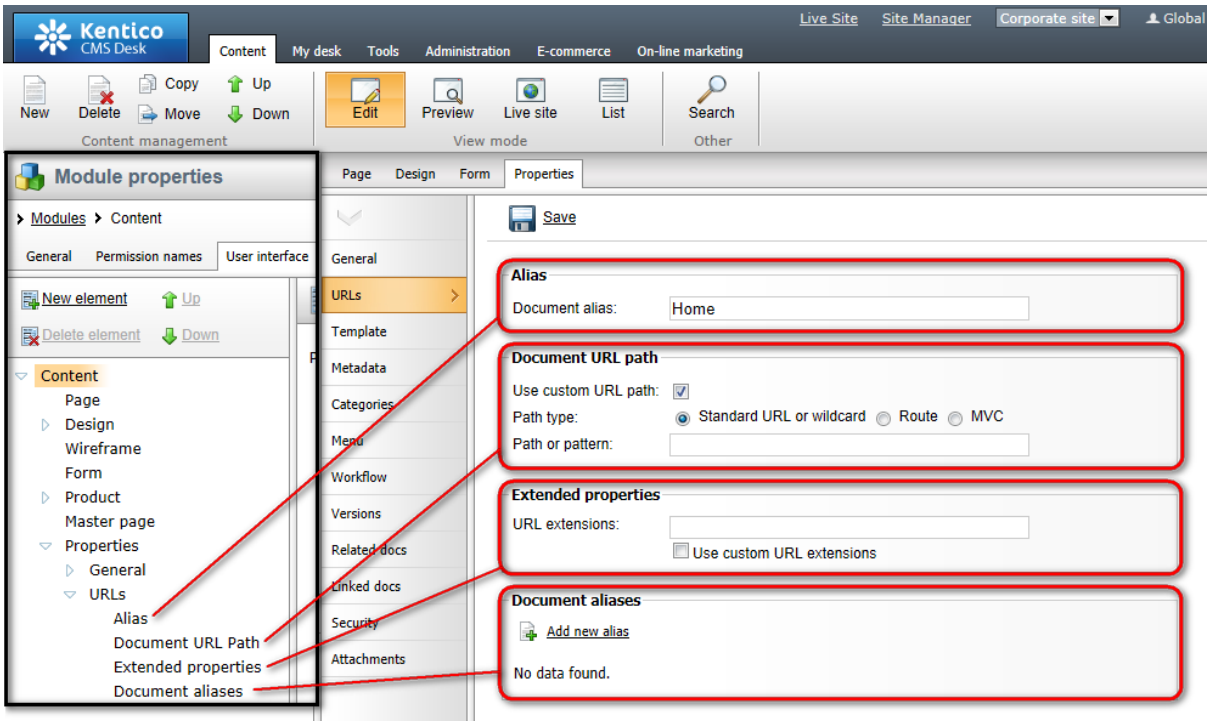
General tab

The **General** tab consists of the **Design**, **Other properties**, **Owner**, **Cache** and **Advanced** sections. The related UI elements have the same names as these sections, as you can see in the following screenshot:



URLs tab

The URLs tab consists of the **Path**, **Extended properties** and **Document aliases** sections. The related UI elements have the same names as these sections, as you can see in the following screenshot:

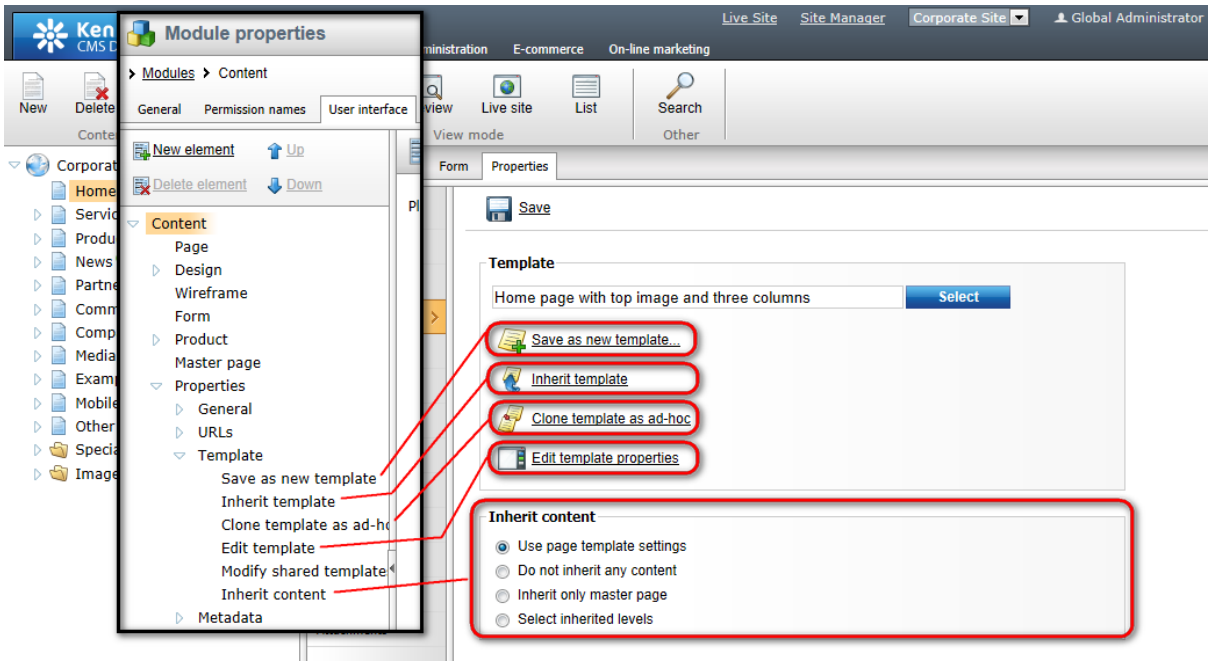


Template tab

On the **Template** tab, you can hide the **Save as new template**, **Clone template as ad-hoc** and **Edit template properties** actions. These actions are represented by the UI elements with the same names, as you can see in the screenshot below.

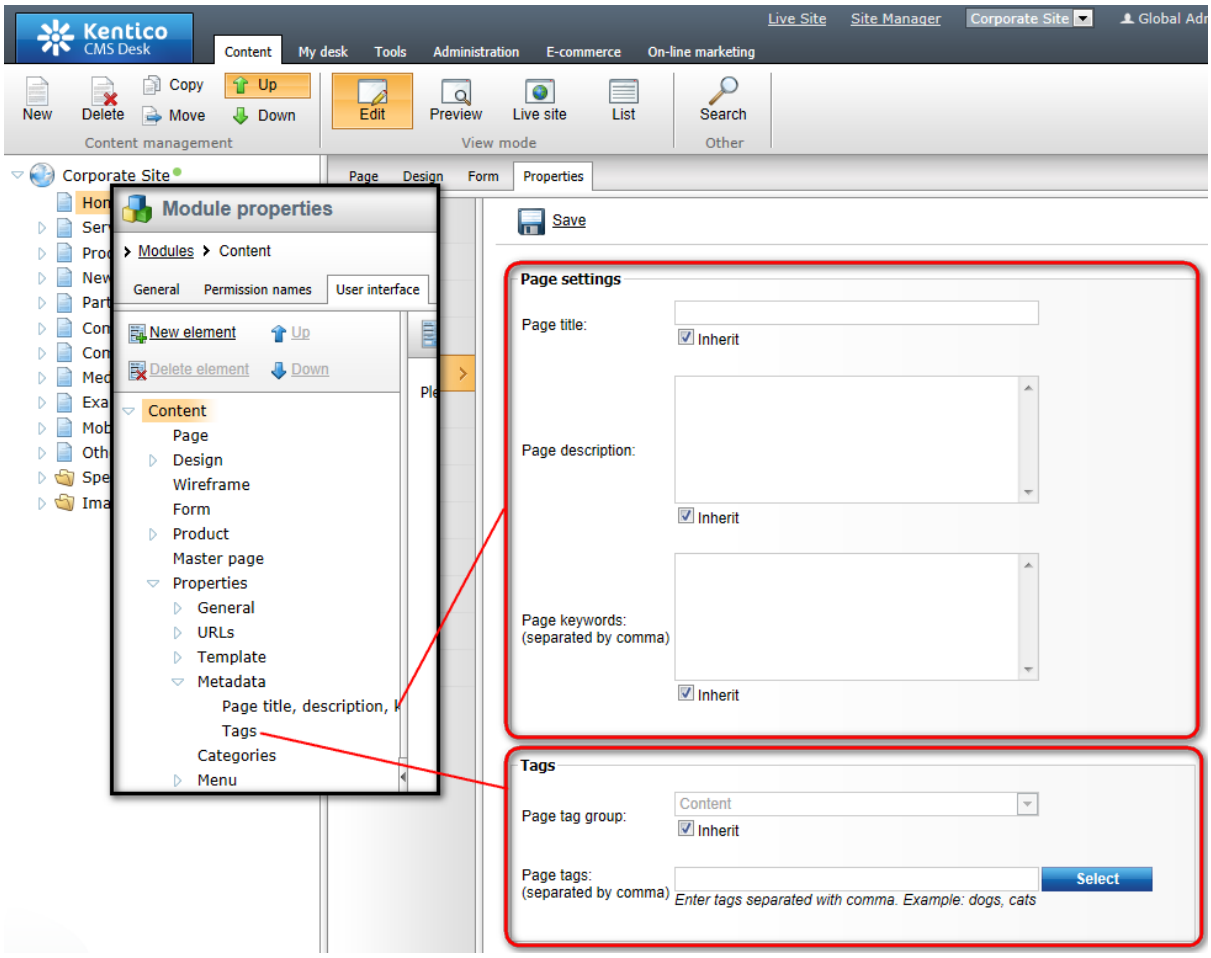
You can also hide the whole **Inherit content** section using the UI element with the same name.

The **Modify shared templates** UI element works as a permission: it determines if the users can modify shared page templates both from here and on the **Design** tab.



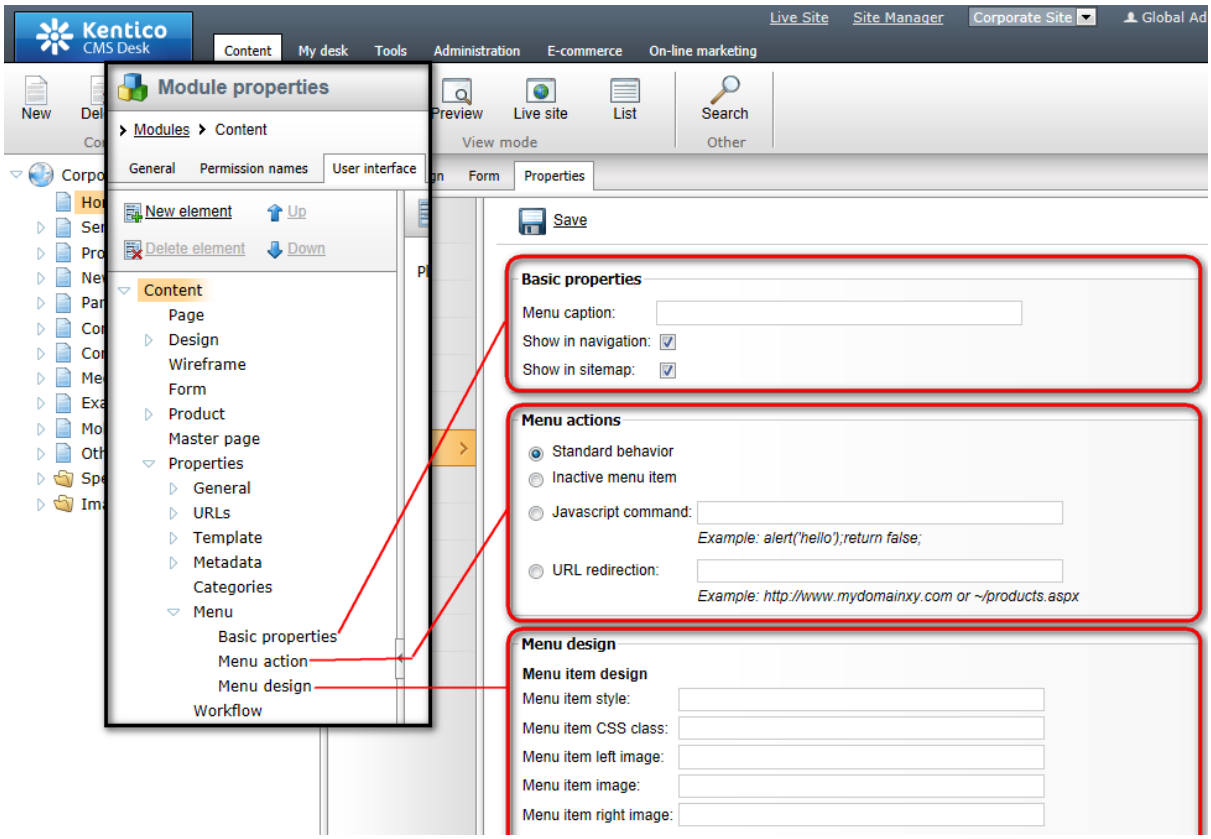
Metadata tab

The Metadata tab consists of two main sections. The **Page settings** section is represented by the **Page title, description, keywords** UI element. The **Tags** section is represented by the **Tags** UI element.



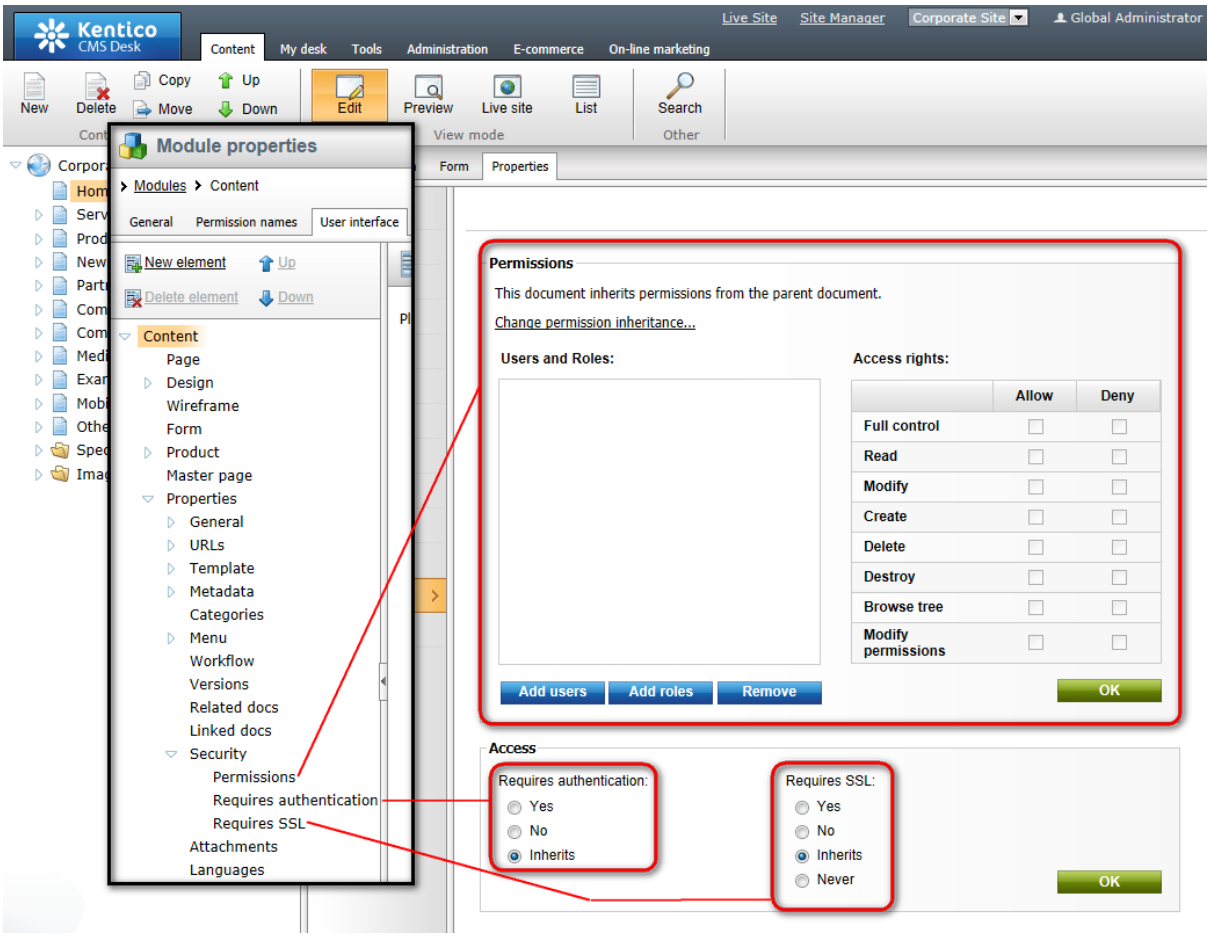
Navigation tab

The **Navigation** tab consists of three sections - **Basic properties**, **Menu actions**, **Search**, **Menu design** - while these sections are represented by UI elements with the same names as shown in the following screenshot:



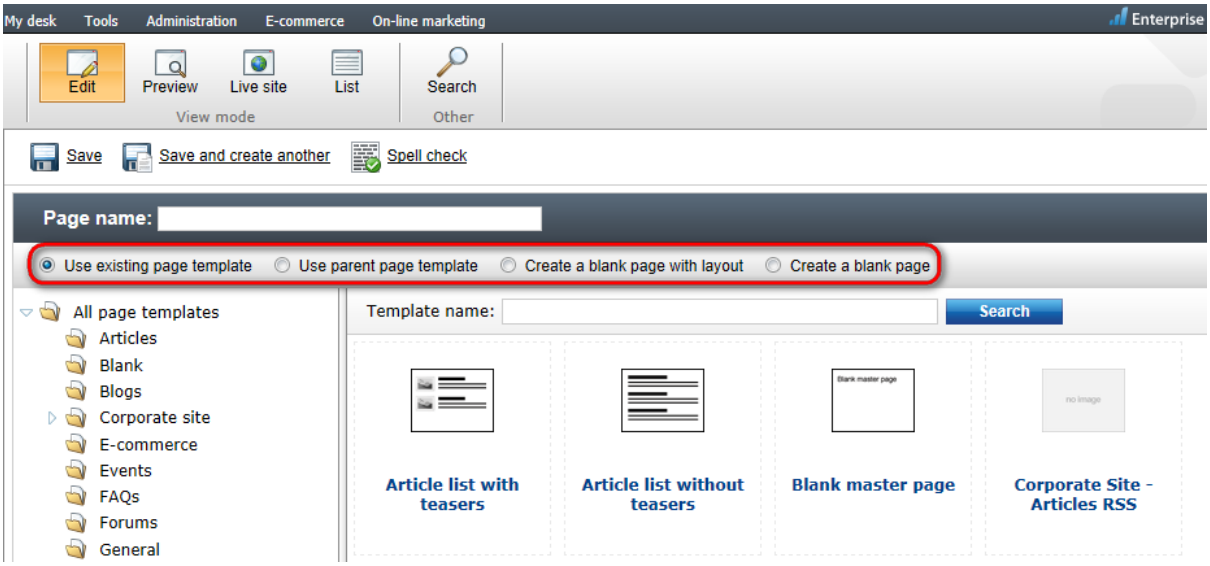
Security tab

The **Security** tab consists of the **Permissions** section, which is represented by the **Permissions** UI element, and the **Access** section, which is divided into two UI element - **Requires authentication** and **Requires SSL**:



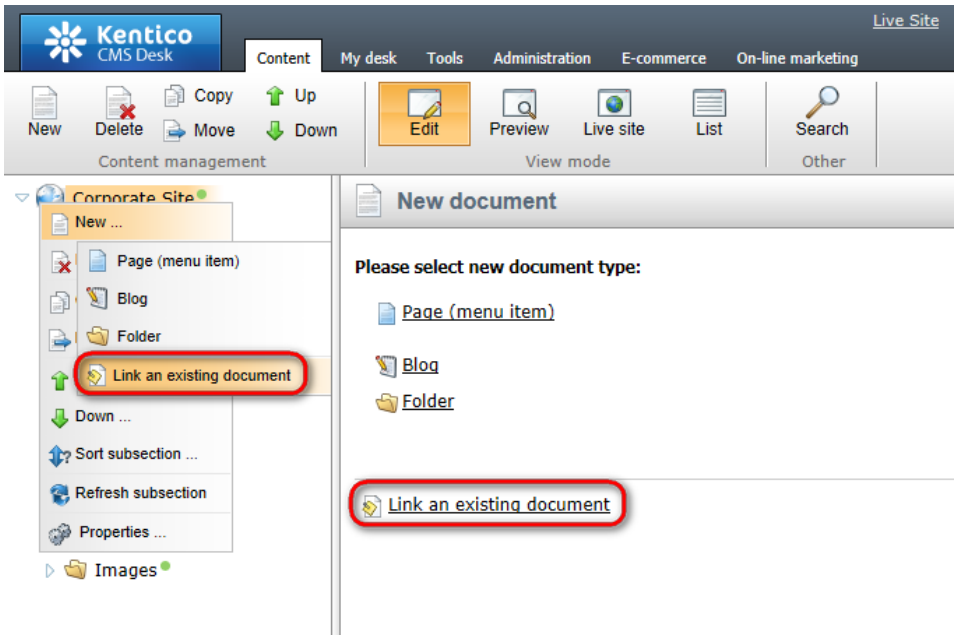
7.13.6.4.5 CMS Desk -> Content -> New

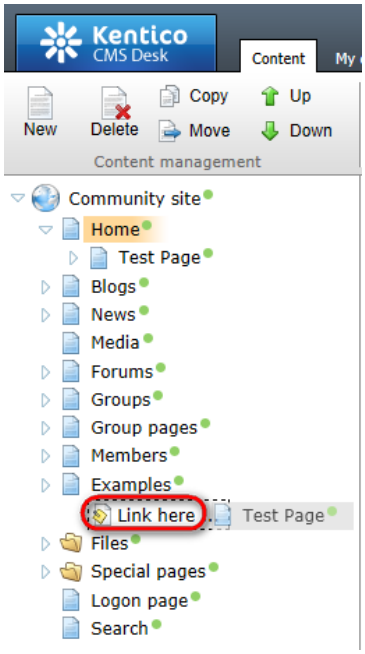
You can hide individual UI elements which are used when choosing a page template while creating a document.



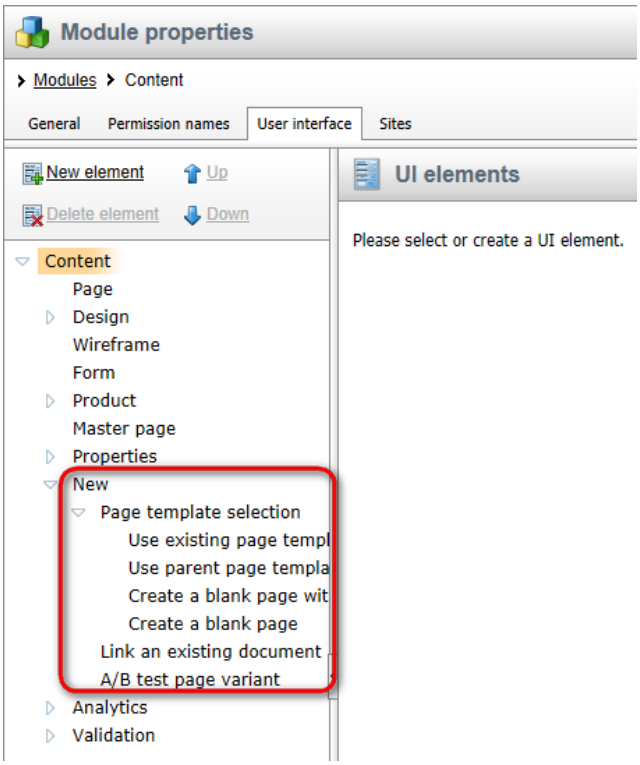
You can also hide the **Link an existing document** link from:

- Context menu
- Document type selection page
- Drag & Drop operation



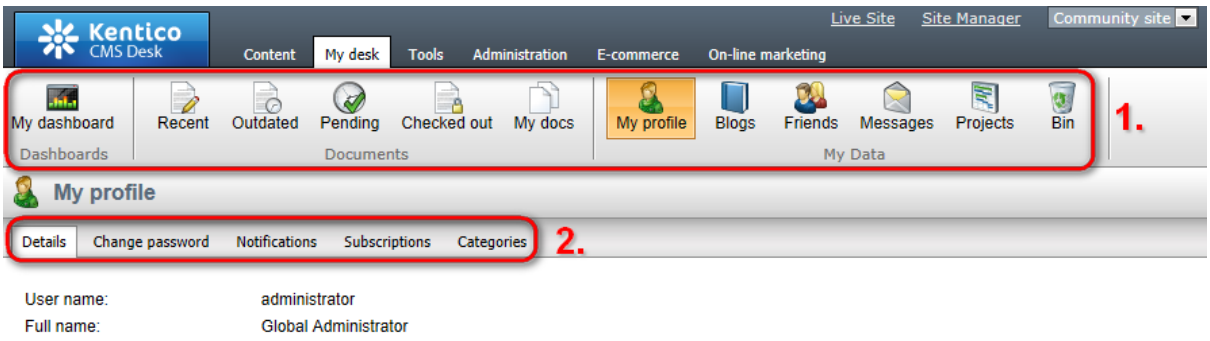


The list of these elements can be found on the **User interface** tab in **Site Manager -> Development -> Modules -> Edit** (🖉) ***Content***.

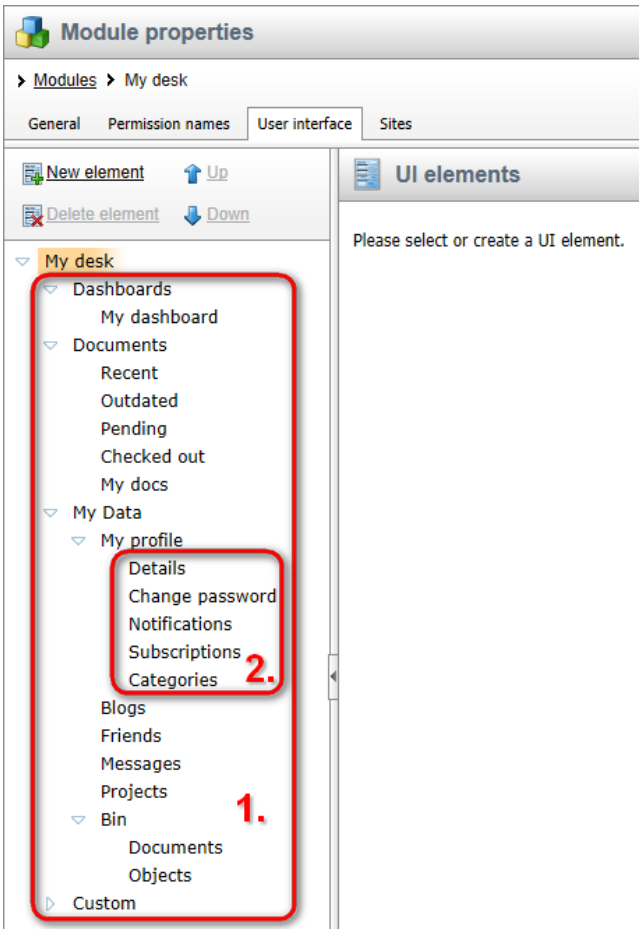


7.13.6.4.6 CMS Desk -> My desk tab

You can hide individual items or entire item groups in the top ribbon menu (1.) of My Desk. Additionally, you can also hide all five tabs of the **My profile** section (2.).

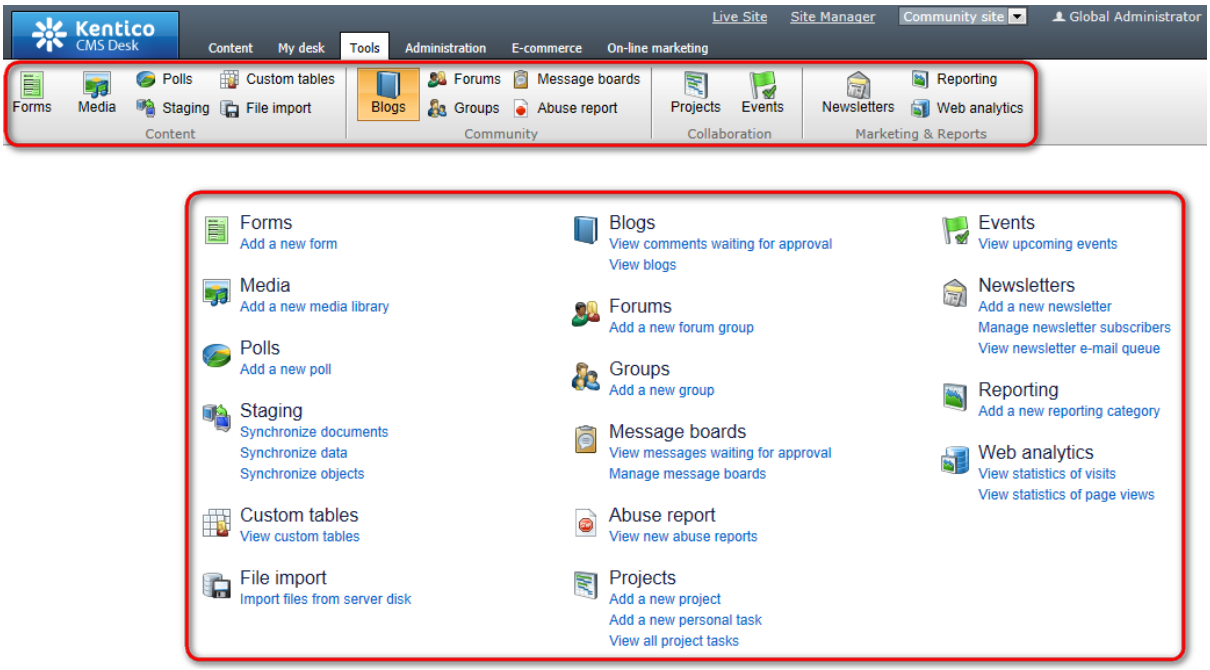


In the screenshot below, you can see the UI elements of the **My desk** module. Their names match the names of the menu items and tabs on the screenshot above.

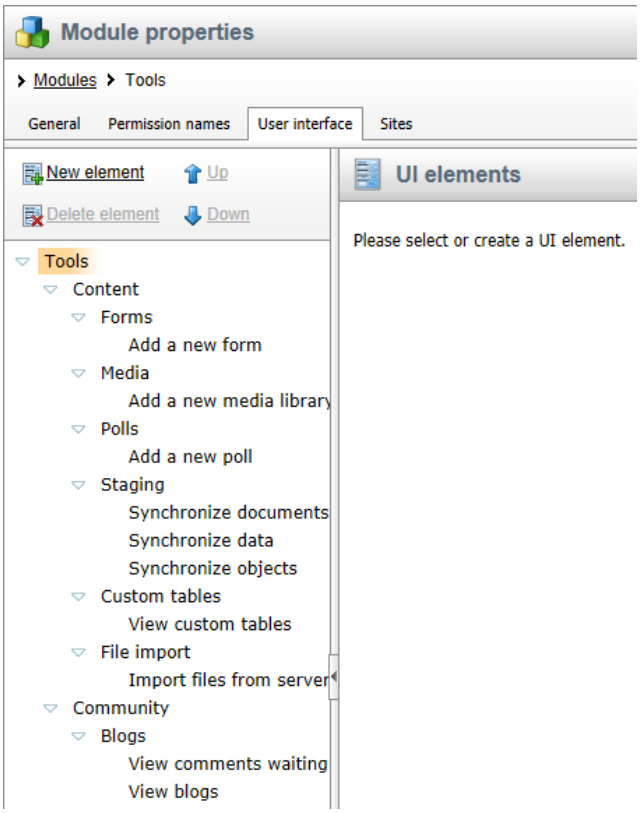


7.13.6.4.7 CMS Desk -> Tools tab

You can hide any item in the Tools ribbon and panel menu as highlighted in the following screenshot. Please note that the first-level UI elements, i.e. Content, Community etc., are not displayed in the panel menu:

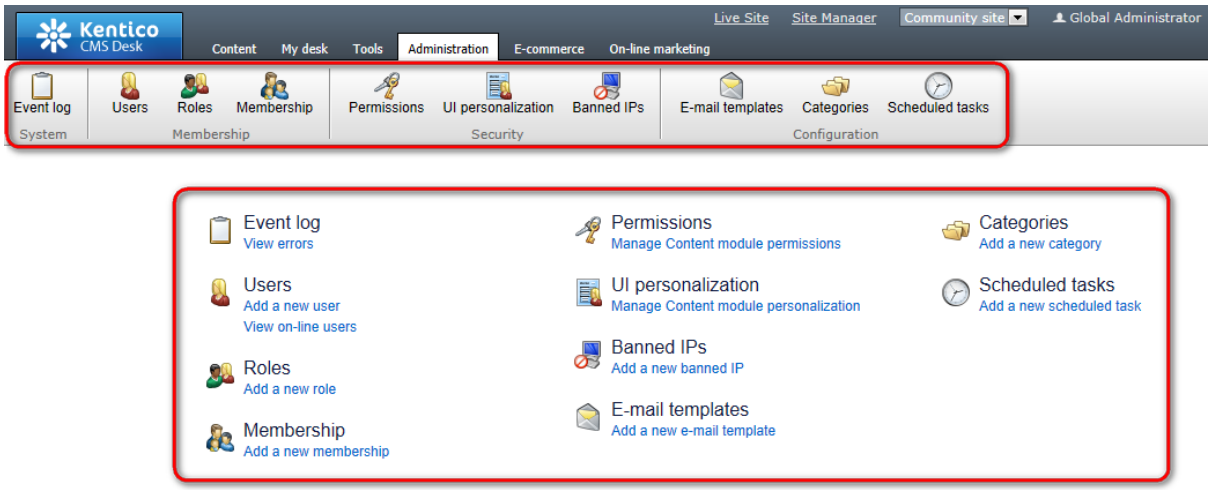


The Tools module has UI elements with the same names as the ribbon and panel menu items in the screenshot above.

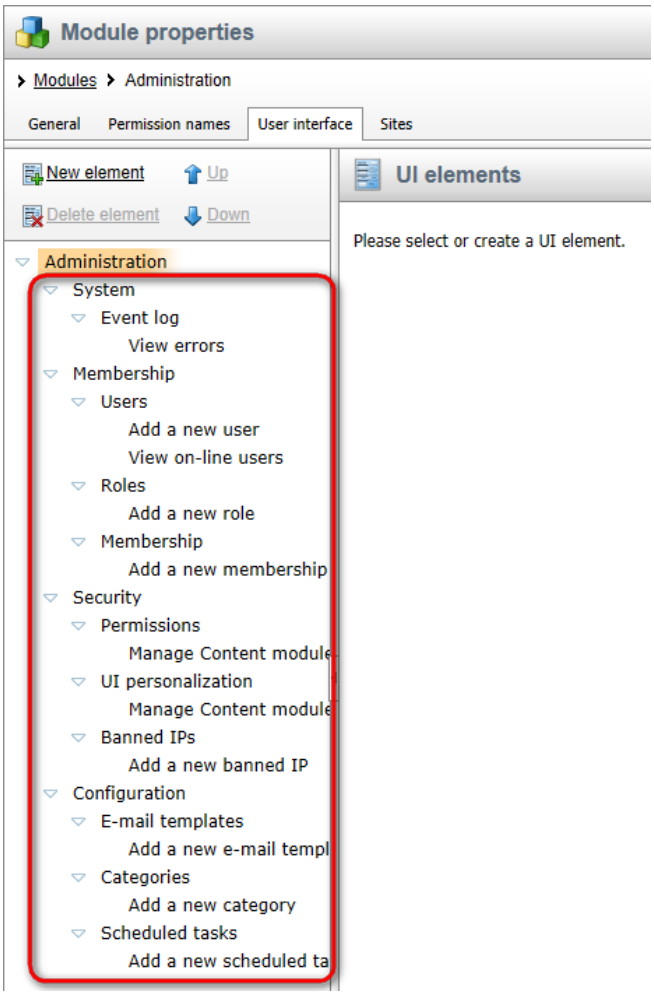


7.13.6.4.8 CMS Desk -> Administration tab

You can hide any item in the Administration ribbon and panel menu as highlighted in the following screenshot. Please note that the first-level UI elements, i.e. System, Membership etc., are not displayed in the panel menu:



The Administration module has UI elements with the same names as the ribbon and panel menu items in the screenshot above.



7.13.6.4.9 CMS Desk -> E-commerce tab

To learn how to customize your **CMS Desk -> E-commerce** tab, please refer to the UI personalization topic in the Security section of the E-commerce Guide.

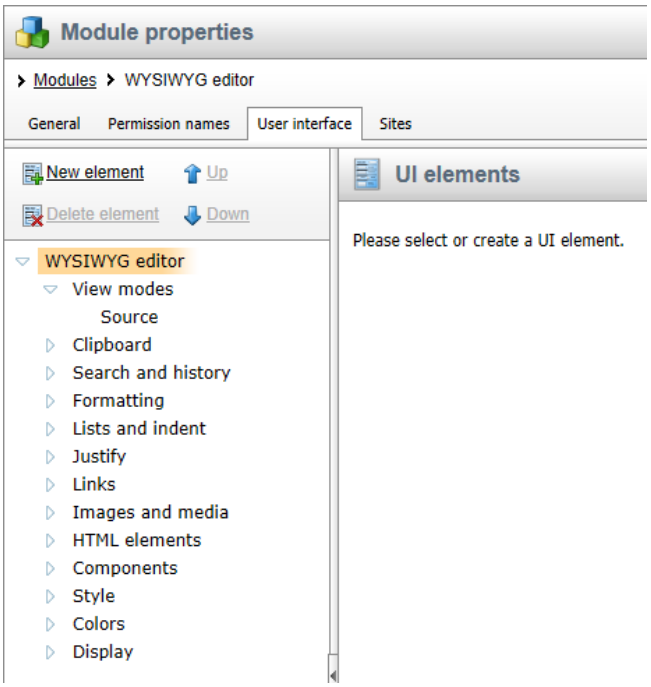
7.13.6.4.10 WYSIWYG editor

All action buttons on the WYSIWYG editor toolbar can be hidden. This can be particularly useful for content editors who do not need to use all these actions.

The **WYSIWYG editor** module has many UI elements grouped under several parent UI elements. The parent elements serve only for categorization purposes and do not represent any action buttons in the toolbar. You can display the sub-elements by clicking the icon next to a particular UI element.

UI element names match tooltips displayed when you place your cursor over a particular

button in the WYSIWYG editor toolbar.



Custom toolbar actions

If you use your custom plug-in in the WYSIWYG editor and want its action button to be displayed or hidden based on UI profile settings, you need to create a new UI element in the WYSIWYG Editor module with the same code name as the code name of the plug-in.

How it works

1. Action buttons are loaded into the toolbar based on the toolbar definition.
2. If UI personalization is enabled, action buttons get filtered according to the user's UI profile.

This means that only such UI elements are displayed in the toolbar that are available both in the user's UI profile and in the toolbar definition.

Toolbar personalization on the live site

Toolbar personalization on the live site is disabled by default. To change this, you need to enable it in your web.config file by adding the following key to the **/configuration/appSettings** section:

```
<add key="CKEditor:PersonalizeToolbarOnLiveSite" value="true" />
```

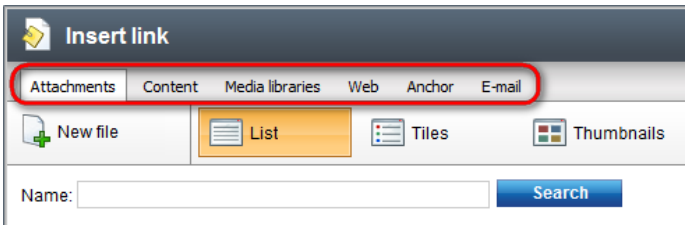
Once this is done, all UI personalization settings will be applied to the WYSIWYG editor on the live site.

More web.config settings can be found in Appendix B - Web.config parameters -> WYSIWYG editor settings.

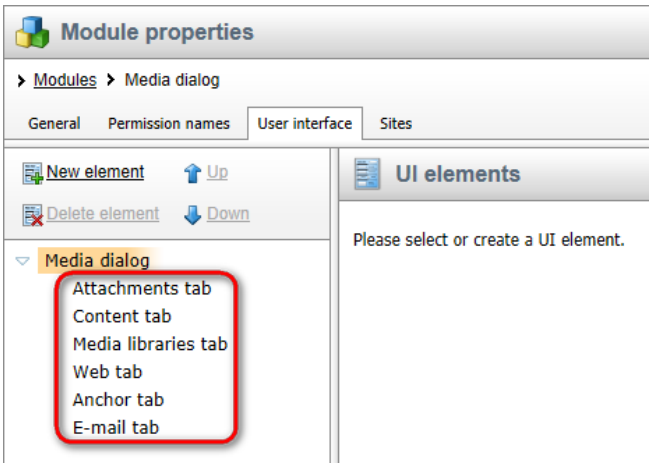
7.13.6.4.11 Media dialog

This dialog is displayed e.g. when using the Insert image or media or Insert link actions in the WYSIWYG editor or when selecting an image in the **Editable image** web part.

You can hide the dialog's tabs - **Attachments**, **Content**, **Media libraries**, **Web**, **Anchor** and **E-mail**.



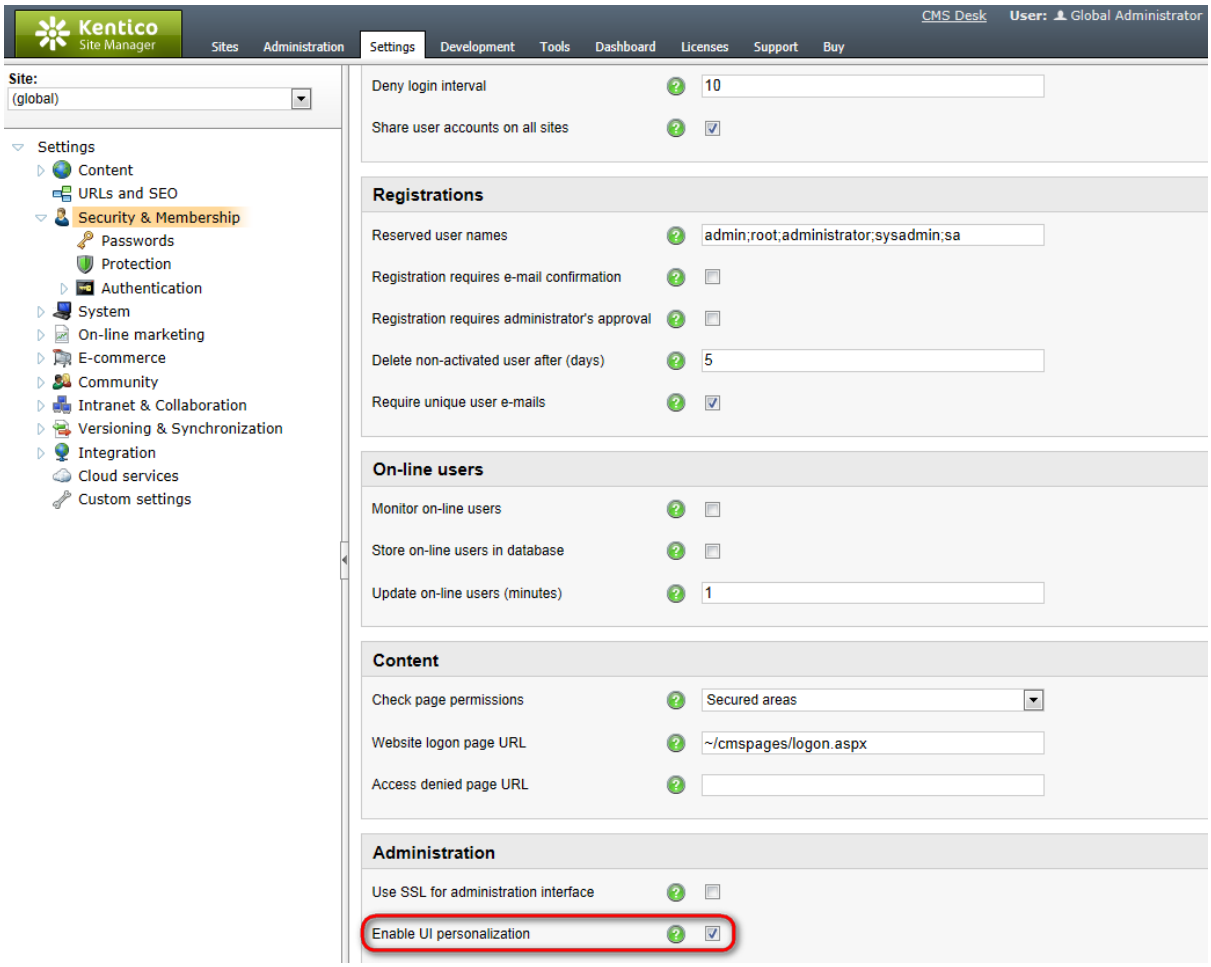
UI elements are defined centrally in the **Media dialog** module. They are applied to all occurrences and versions of the dialog, even though some of these tabs are not displayed in all cases (e.g. the E-mail tab is not displayed in Insert image or media dialog).



7.13.6.5 Enabling UI personalization

For UI personalization to be functional, you need to go to **Site Manager -> Settings -> Security & Membership** and enable it using the **Enable UI personalization** check-box.

Using the **Site** drop-down, you can decide if you want to turn it on globally, or just for some particular sites in the system.



7.13.6.6 UI profile configuration

In the following step-by-step example, you will learn how to define UI profile for a new role in **Site Manager -> Administration -> UI personalization**.

Let's presume that we want some users to be **Forum administrators** of our website. This means that they will not need to use the **Content** or **Administration** tabs at all, all they will be concerned about is the Forums module administration interface in the **Tools** menu and a few parts of the **My desk** section.

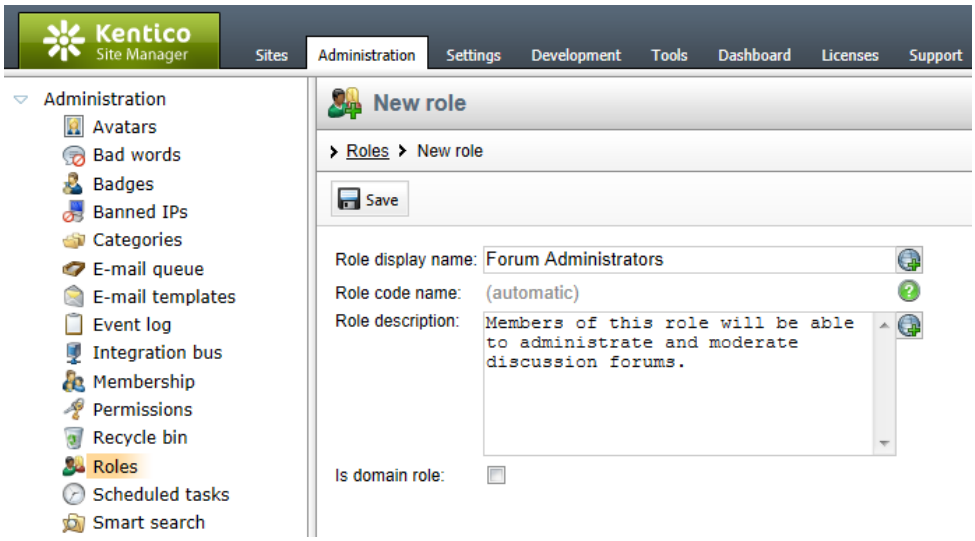
How to achieve it? We will create a new role and a new user who will be a member of this role only. Then we will define UI profile settings for this role and try logging in as the new user to see the simplified UI.

1. Install some of the sample sites or your own site and enable UI personalization for the site as described here.

2. Go to **Site Manager -> Administration -> Roles**, choose your site in the **Site** drop-down and click on **New role** ( ). In the following dialog, enter these details:

- **Role display name**: Forum Administrators
- **Role description**: text describing the role
- **Is domain role**: leave the field unchecked

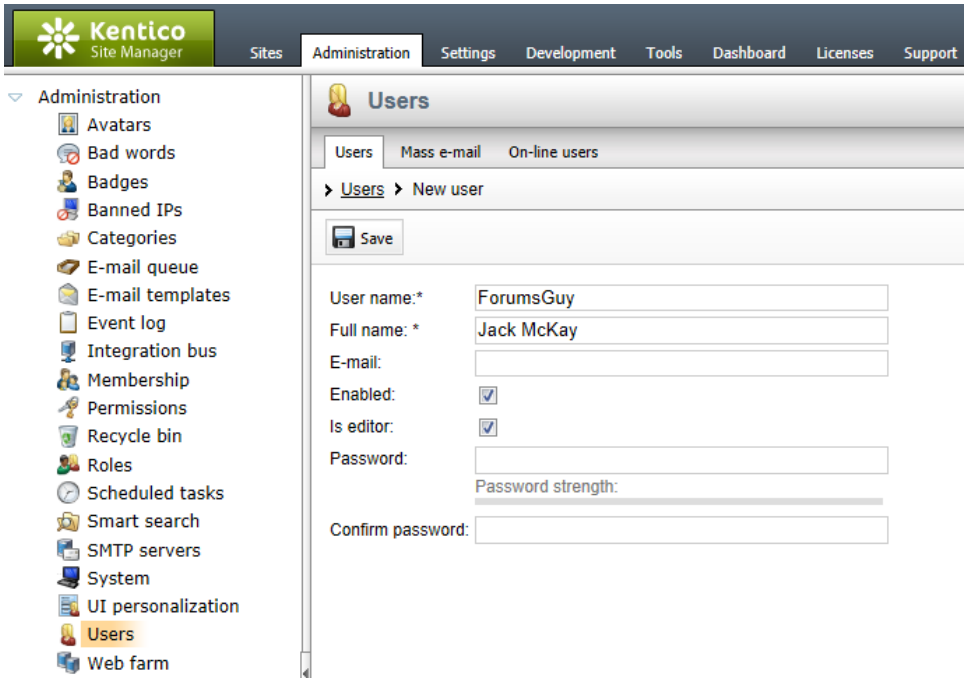
Click **Save**.



3. Now let's create the user. Go to **Site Manager -> Administration -> Users** and click **New user**. Enter the following details in the New user dialog:

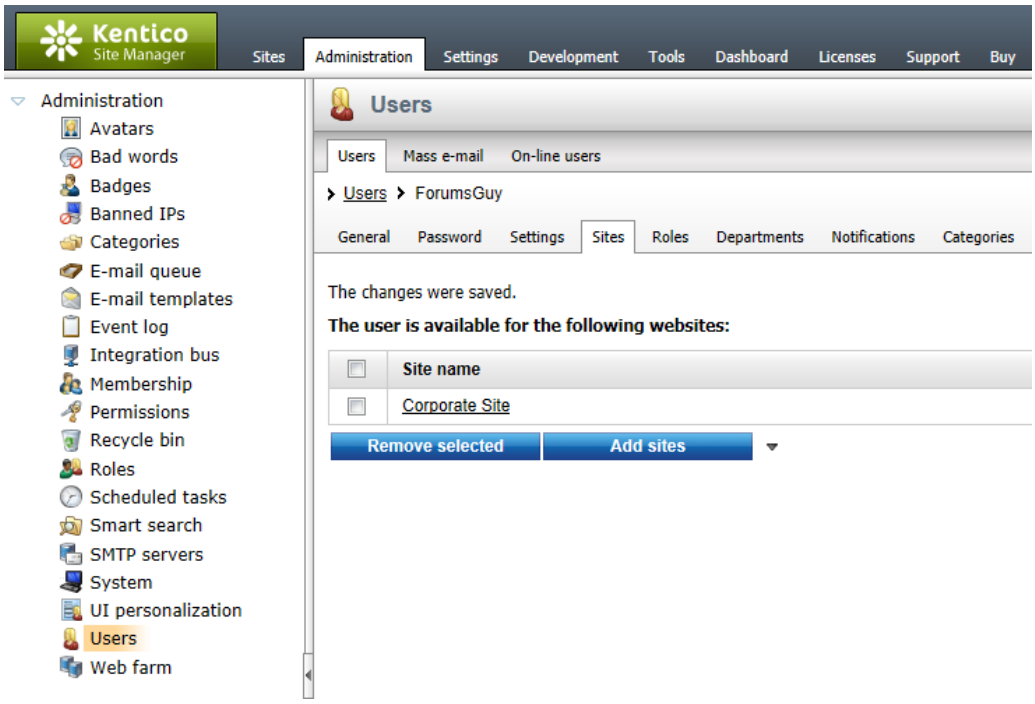
- **User name**: ForumsGuy
- **Full name**: Jack McKay
- **Enabled**: enabled
- **Is editor**: enabled

Click **Save**.

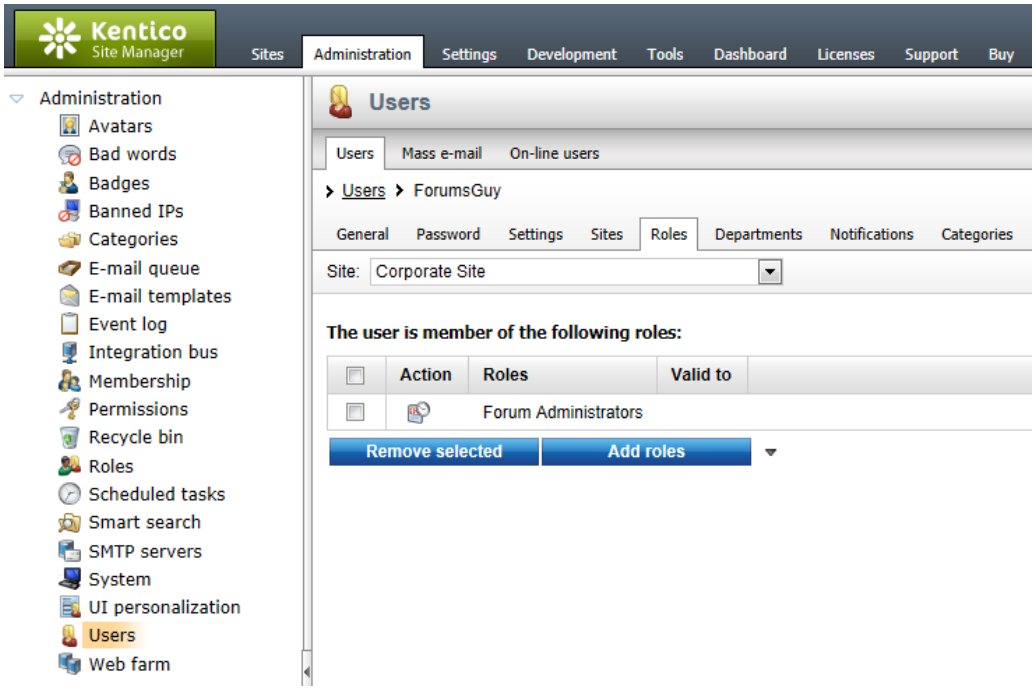


4. You will be redirected to the new user's editing interface. We first need to assign the user to our site.

Go to the **Sites** tab and add the site using the **Add sites** button.



5. Now that the user belongs to our site, we can assign him to the **Forum Administrators** role, which also belongs to the site. Switch to the **Roles** tab, select your site from the **Site** drop-down list and use the **Add role** button to add the Forum Administrators role.



6. Now that the user belongs to our role, we can set up UI personalization for the role. Go to **Site Manager -> Administration -> UI personalization** and choose:

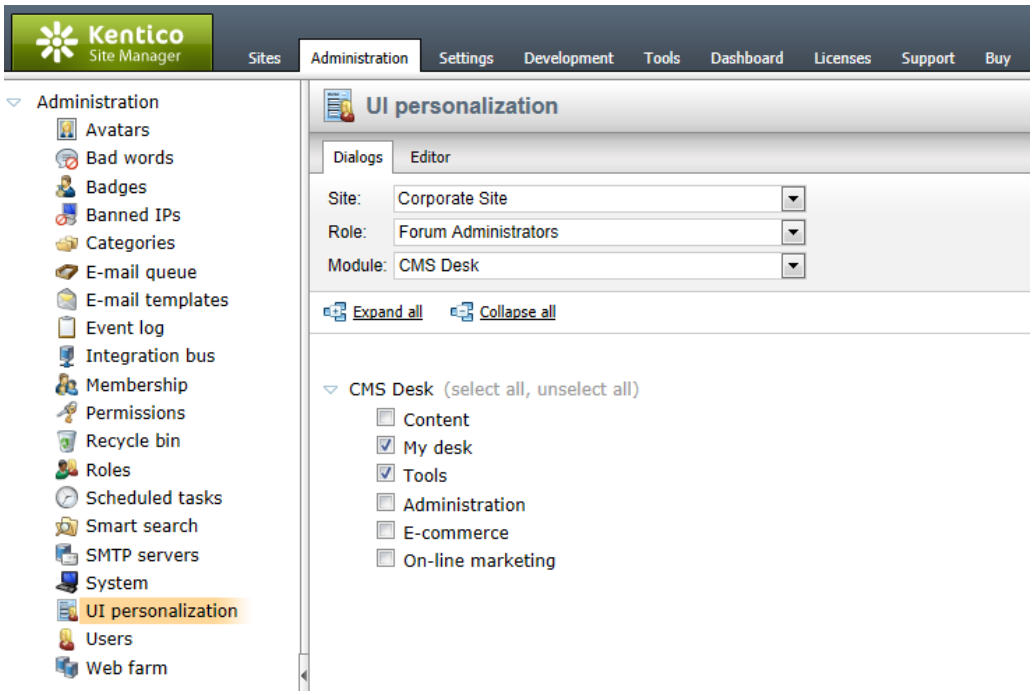
- **Site**: your site
- **Role**: Forum Administrators

Please note: UI personalization settings are **site-related** unless they are defined for global roles. This means that members of one site-specific role can see a certain personalized UI when editing one site and a completely different UI when editing another site. More information on how UI personalization works can be found here.

You can choose the module whose UI elements you want to set up using the **Module** drop-down. Full reference on the personalizable parts of CMS Desk and the appropriate modules can be found here.

7. Let's start with the main tabs in CMS Desk, where we want only the **My desk** and **Tools** tabs visible. Choose **Module**: CMS Desk and make the following settings:

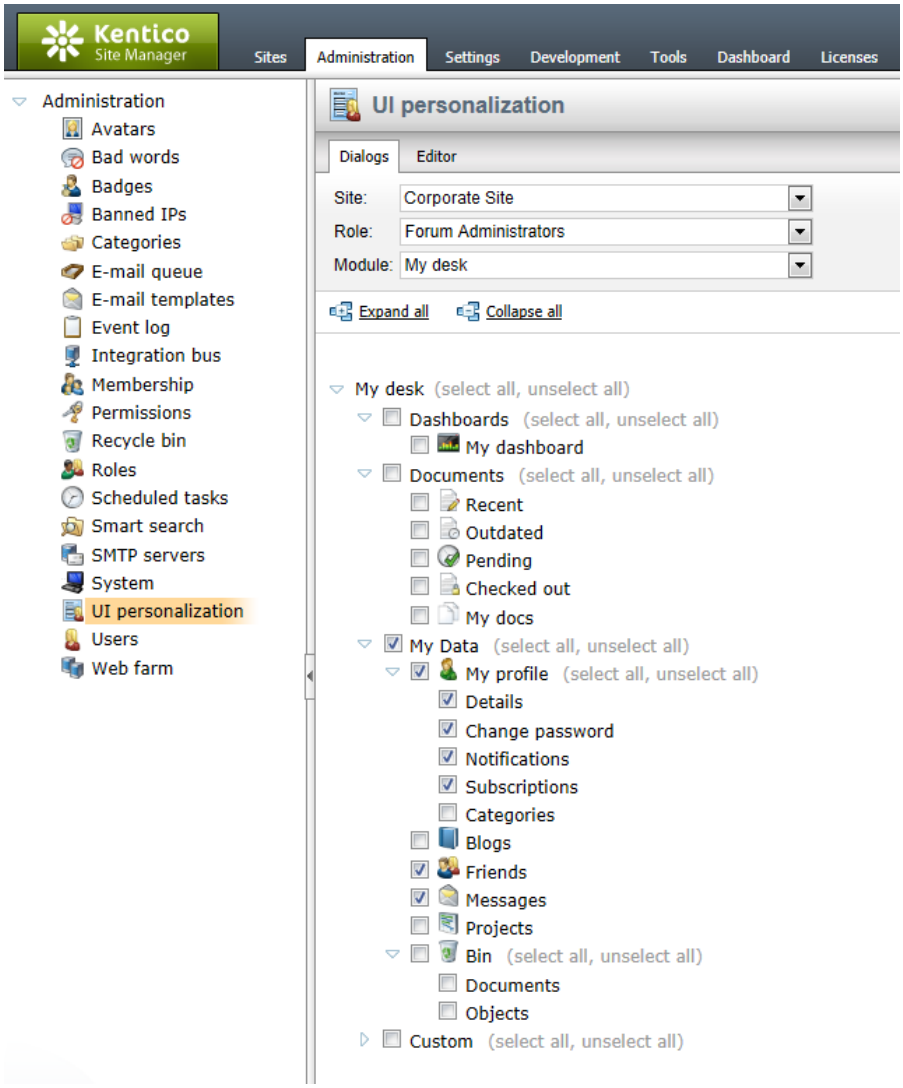
- **Content**: disabled
- **My desk**: enabled
- **Tools**: enabled
- **Administration**: disabled



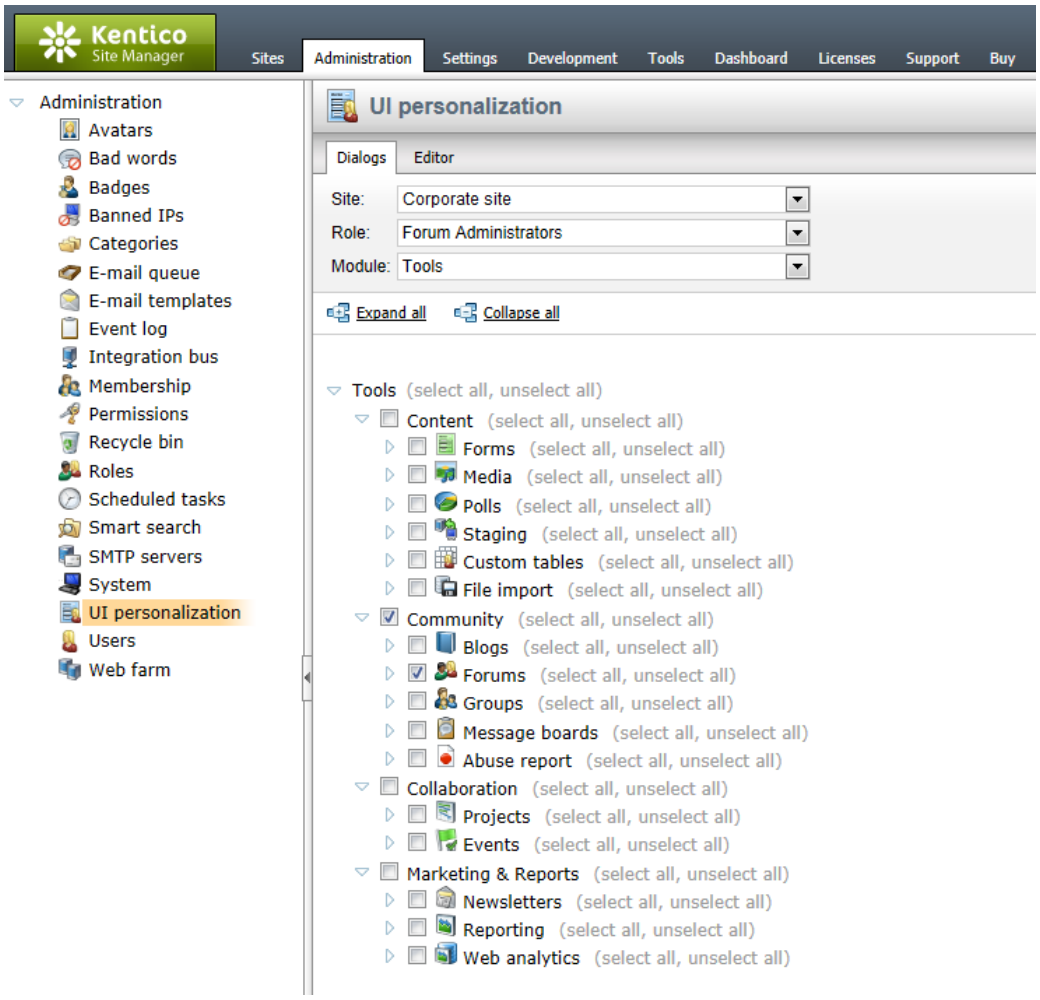
8. On the My desk tab, we will need only the Account, Messages and Friends sections, all others are not needed for forum administrators. Choose **Module**: My Desk and enable only the UI elements listed below, do not enable the rest.

- **My data**: enabled
  - **Account**: enabled
    - **Details**: enabled
    - **Change password**: enabled
    - **Notifications**: enabled
    - **Subscriptions**: enabled

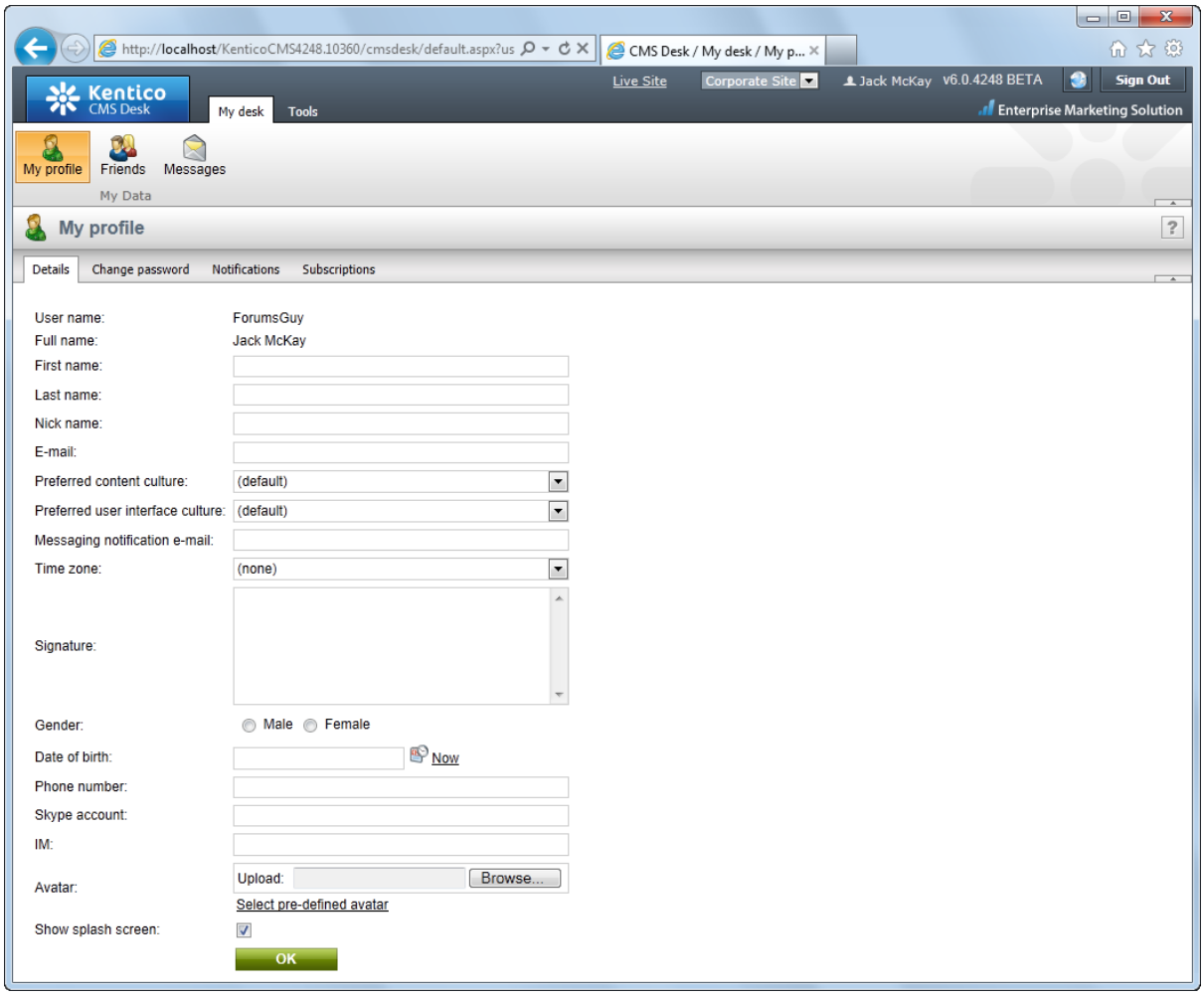
- **Friends**: enabled
- **Messages**: enabled



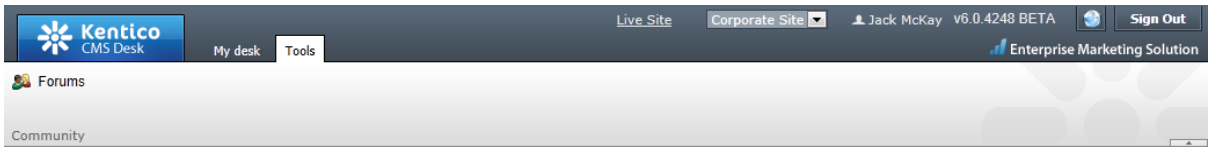
9. Finally, we need only the **Forums** option in the **Tools** menu. Choose **Module**: Tools and enable only the **Community** and **Tools** UI elements as you can see in the screenshot below. Leave all the remaining UI elements disabled.



10. We can now verify what we have achieved. Log out of **Site Manager** and log back in to **CMS Desk** as the *ForumsGuy* user created in step 3. You will see the a simplified user interface as in the following screenshot.



If you switch to the **Tools** tab, there is only the **Forums** menu item present. It is evident that this UI is much easier to understand for a first-time end user.



7.13.6.7 UI element management

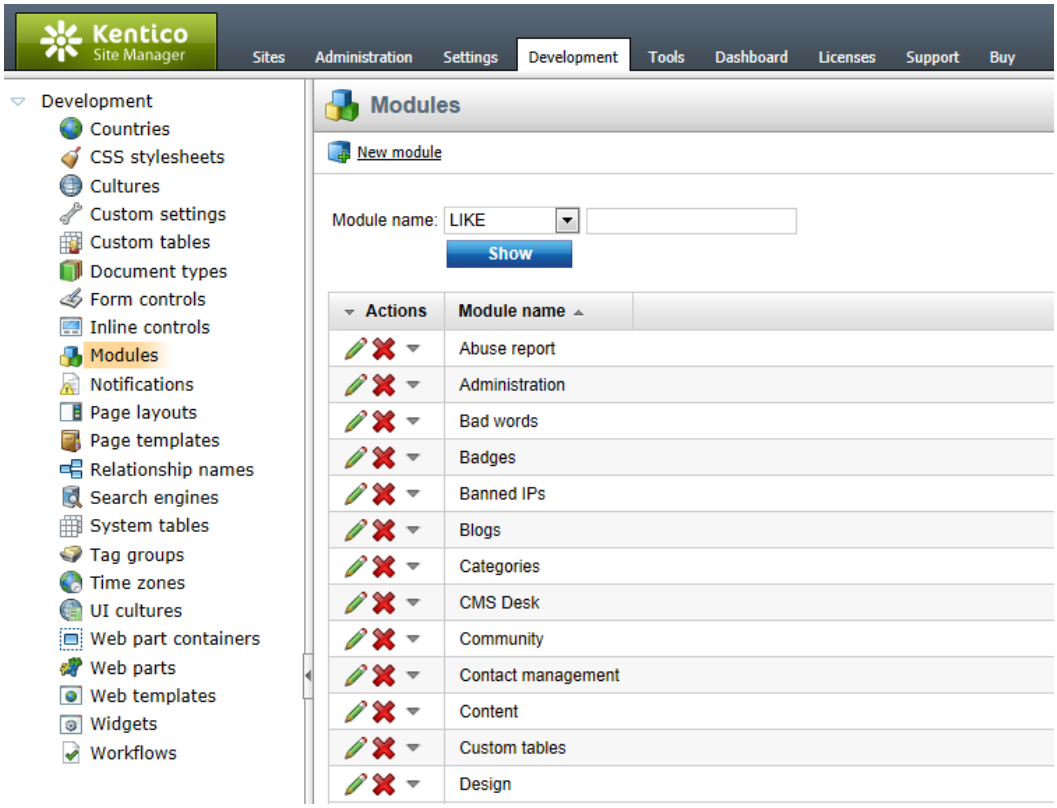
7.13.6.7.1 UI element management

The Kentico CMS user interface consists of modules. Modules contain particular UI elements. UI element is a page or part of a page in CMS Desk which can be hidden from a specified user. It can be:

- **tab**
- **menu item**
- **group of controls**

UI elements of a particular module can be edited or added in **Site Manager -> Development ->**

Modules. This is where you can see a list of all modules in the system. For information on which module represents which parts of the CMS, please refer to the Personalizable parts of CMS Desk chapter.



If you want to manage UI elements of some module, click the **Edit** ( ) icon next to the module and switch to the **User interface** tab. On the **User interface** tab, you can see all UI elements of the module in the tree on the left.

- You can re-order UI elements using the **Up** ( ) and **Down** ( ) buttons. This order will then be reflected in the real UI for tabs and menu items. Parts of pages (groups of controls) cannot be re-ordered this way.
- You can delete existing elements using the **Delete element** ( ) button.
- New elements can be created using the **New element** ( ) button (see the example), while the new element is always created under the currently selected node.

Settings for each UI element are divided into two tabs - **General** and **Roles**.

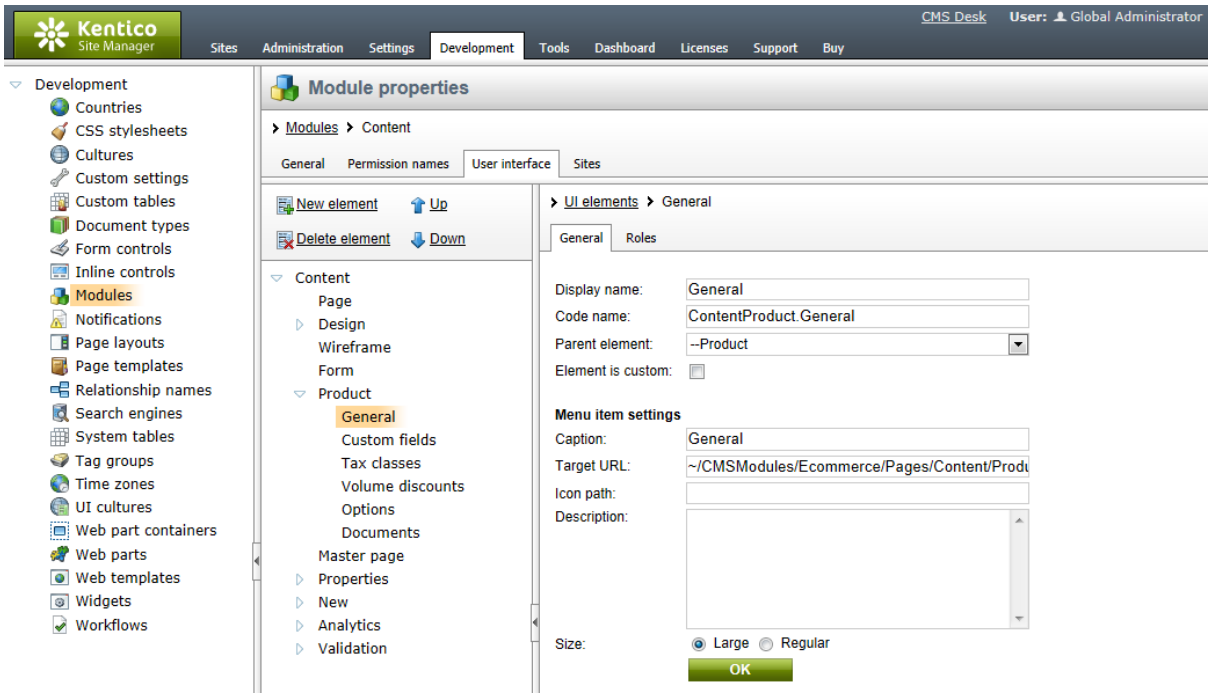
On the **General** tab, you can set the following properties of the UI element.

- **Display name** - the name of the element displayed in the administration interface (i.e. in the settings, not in the final UI). It can be entered either as plain text or as a localizable string in the *{$mystringname$}* format.
- **Code name** - the name of the element used by developers in website code. It must be unique within a module.
- **Parent element** - using this drop-down list, you can change the hierarchical position of the element in the UI elements tree for this module. You can select either the name of the module (which places the element under the root) or one of the other UI elements (which places the element under the

selected element).
- **Element is custom** - if false, the element is a native part of Kentico CMS. Set this value to *TRUE* for your custom UI elements.

Menu item settings
- **Caption** - the caption of the UI element displayed in the final UI. It can be entered either as plain text or as a localizable string in the *{$mystringname$}* format.
- **Target URL** - URL of the page which is the content of the UI element. You can enter both absolute (e.g. *http://www.google.com*) and relative (e.g. *~/CMSModules/Content/CMSDesk/Default.aspx*) URLs.
- **Icon path** - the icon displayed next to the UI element caption - applicable **only for menu items**. You can enter either a full relative path begining with ~ (e.g. *~/App_Themes/Default/Images/CMSModules/list.png*) or a short path begining under the *Images* folder of the current skin (e.g. *CMSModules/list.png*).
- **Description** - the description of the UI element.
- **Size** - the size of the UI element icon. Takes effect only for UI elements which are included in a ribbon-like toolbar.

Localization expressions, i.e. **Localize other languages** (), **Remove localization** () and **Localize** (), are described in detail in the Localization expressions topic in the Development -> Multilingual and international support section.



On the **Roles** tab, you can directly configure which roles will be able to see the selected UI element. All roles belonging to the site selected in the **Site** drop-down list will be displayed in the matrix. The *(global)* option can be selected to configure global roles, i.e. those that are not limited to a single site.

UI personalization configuration is then performed the following way:

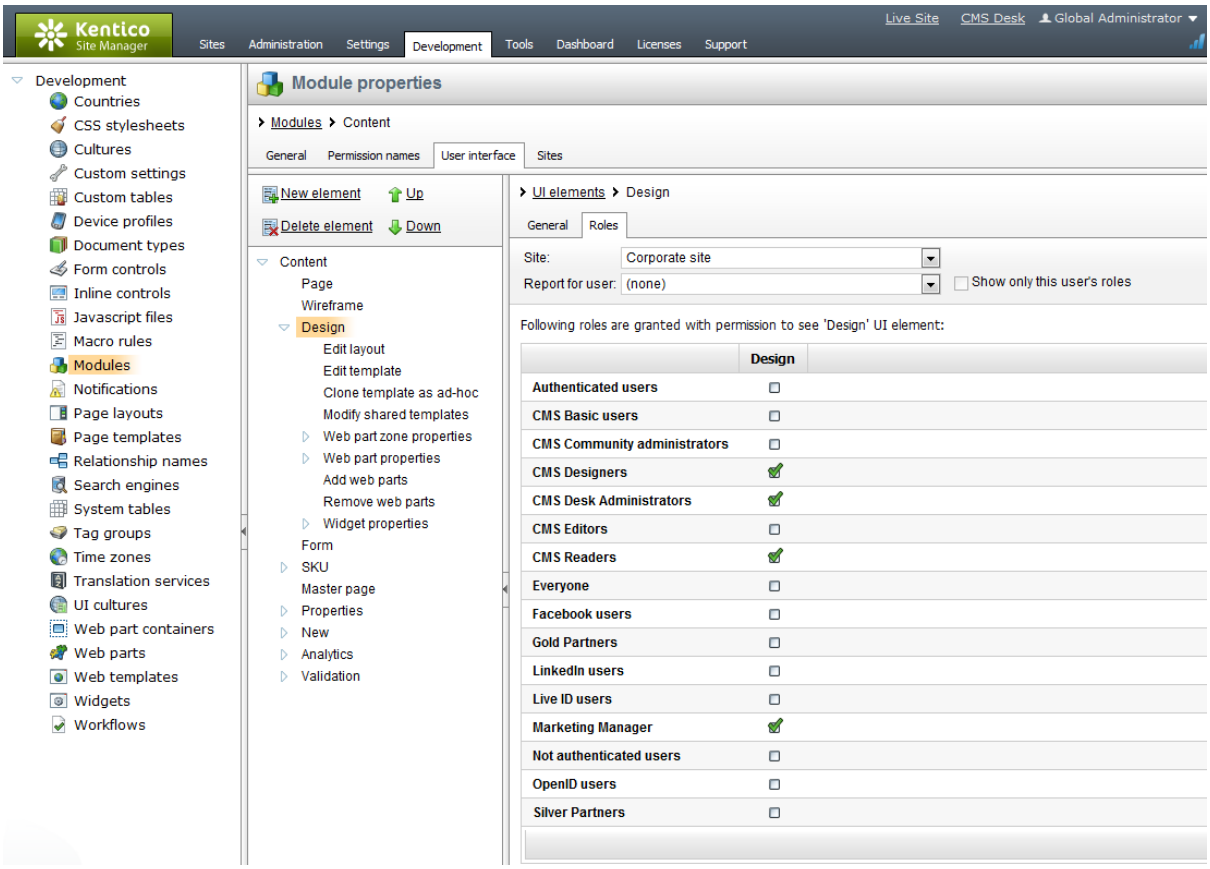
- If you enable () a check-box, the UI element will be visible to members of the role.
- If you disable () a check-box, members of the role will not see the UI element (unless they are

members of some other role which has it allowed).

To have the same UI access permissions like the parent assigned for the given UI element, click **Copy permissions from parent**. All existing UI element permissions will be deleted and new ones based on the parent settings will be assigned.

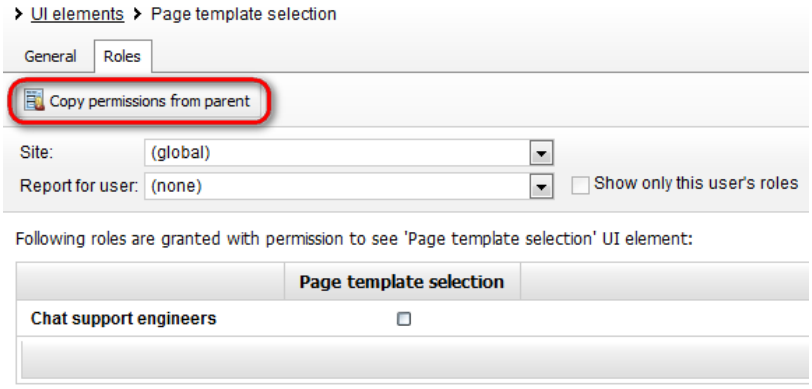
Each user's UI profile is defined by UI personalization settings for their roles. If a user wants to see a UI element in the UI, at least one role where the user is member needs to have the UI element set as visible.

To check if a particular element is visible to a user, you can select the user from the **Report for user** drop-down list. After doing so, a sum of all permissions granted to the user's roles is displayed in the first line, highlighted in green color. Roles where the selected user is a member will be highlighted in pink color. If you enable the **Show only this user's roles** check-box, only the pink roles will be displayed in the matrix.

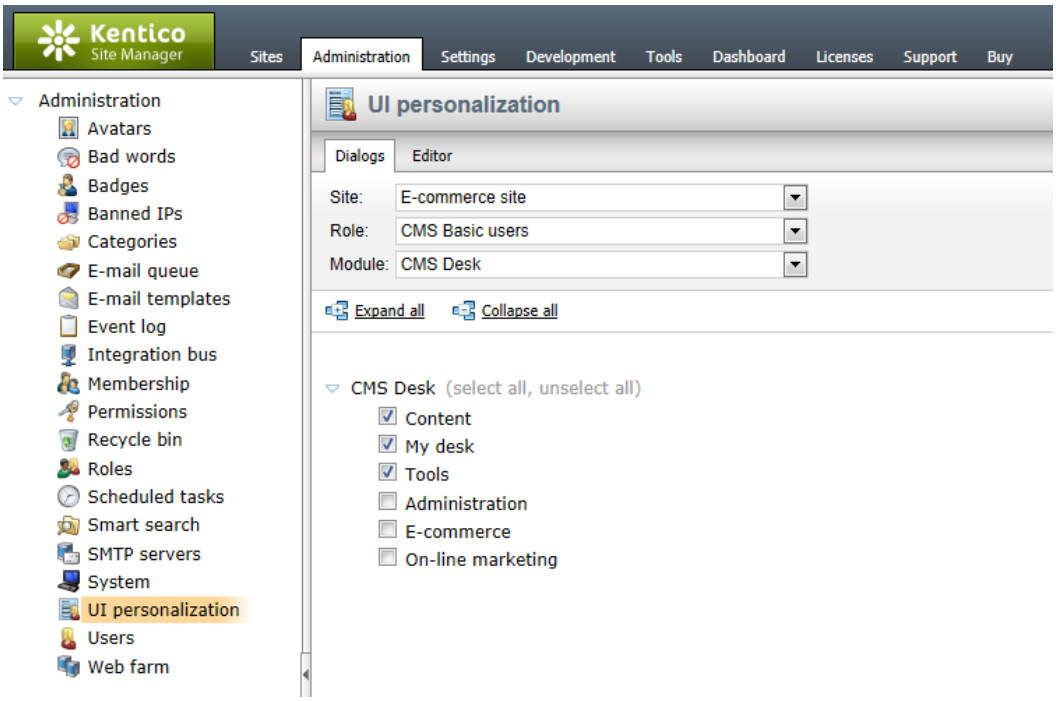


Please note

Using **Copy permissions from parent** affects all available sites, i.e. not only the currently selected one.
Because the root-level UI element cannot have any UI access permissions assigned, the button is disabled for all first-level UI elements.



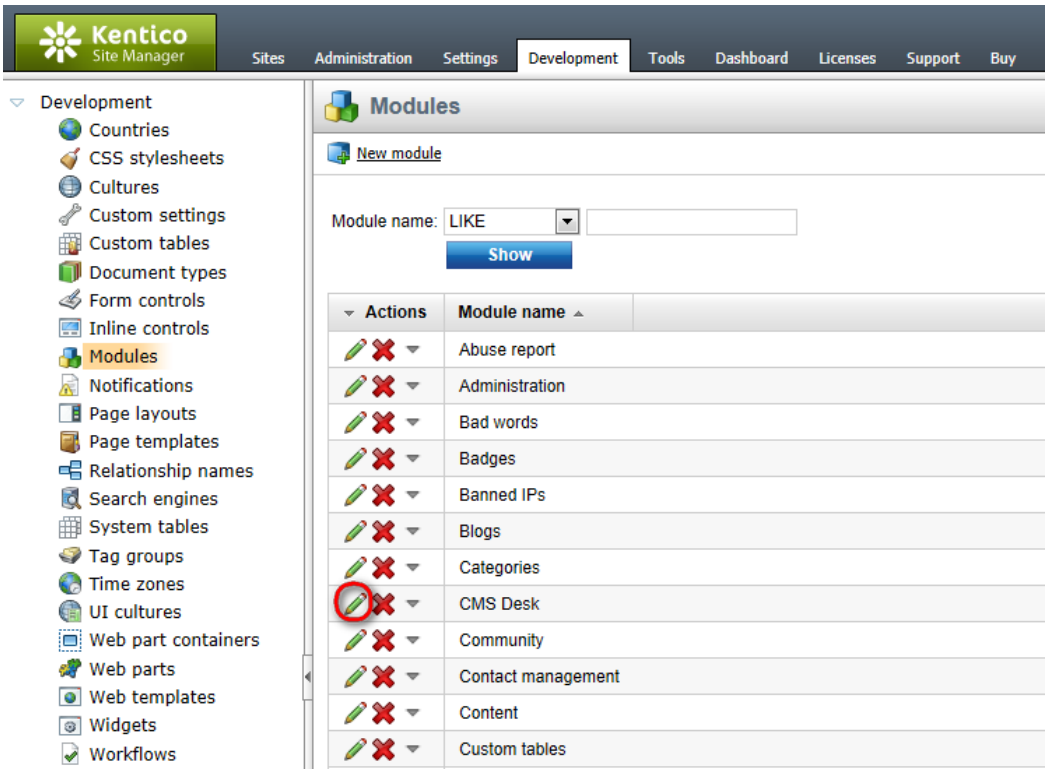
This is just another way to make the same configuration as described in the UI profile configuration topic, i.e. settings made here are reflected in **Site Manager -> Administration -> UI personalization** (see the screenshot below) and vice-versa.



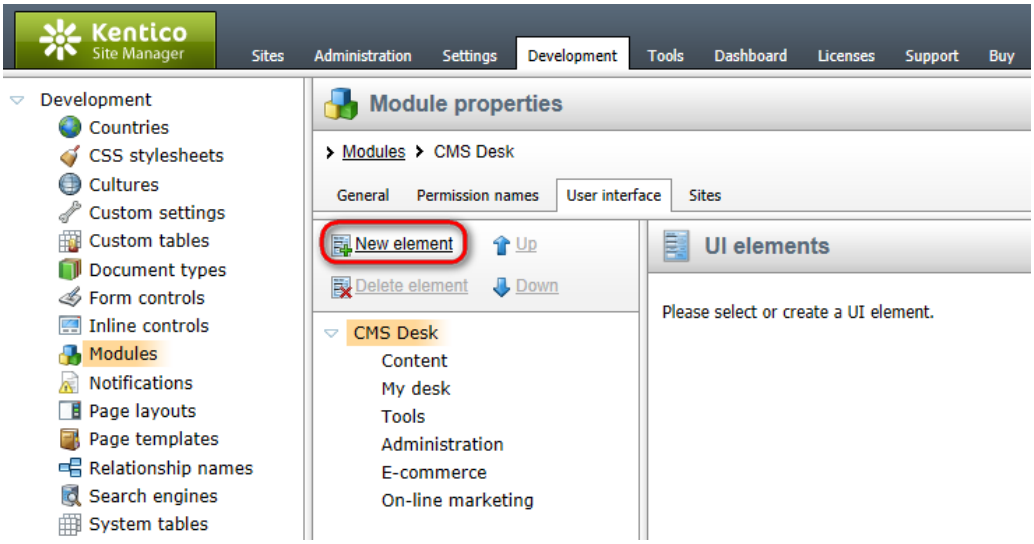
7.13.6.7.2 Example: Adding a new main tab to CMS Desk

In this example, you will learn how to add another tab next to the **Content**, **My desk**, **Tools**, **Administration**, **E-commerce** and **On-line marketing** tabs in **CMS Desk**. This procedure can be used to add your custom UI elements to any other personalizable part of the CMS. You can also integrate your custom modules into the UI this way, as described here.

1. We know from this chapter that the main tabs belong to the **CMS Desk** module. Go to **Site Manager -> Development -> Modules** and **Edit** (🖉) the **CMS Desk** module.



2. In the module's administration interface, switch to the **User interface** tab. You can see the six UI elements representing the tabs in the tree on the left. Select the root of the tree and click the **New element** ( ) button.

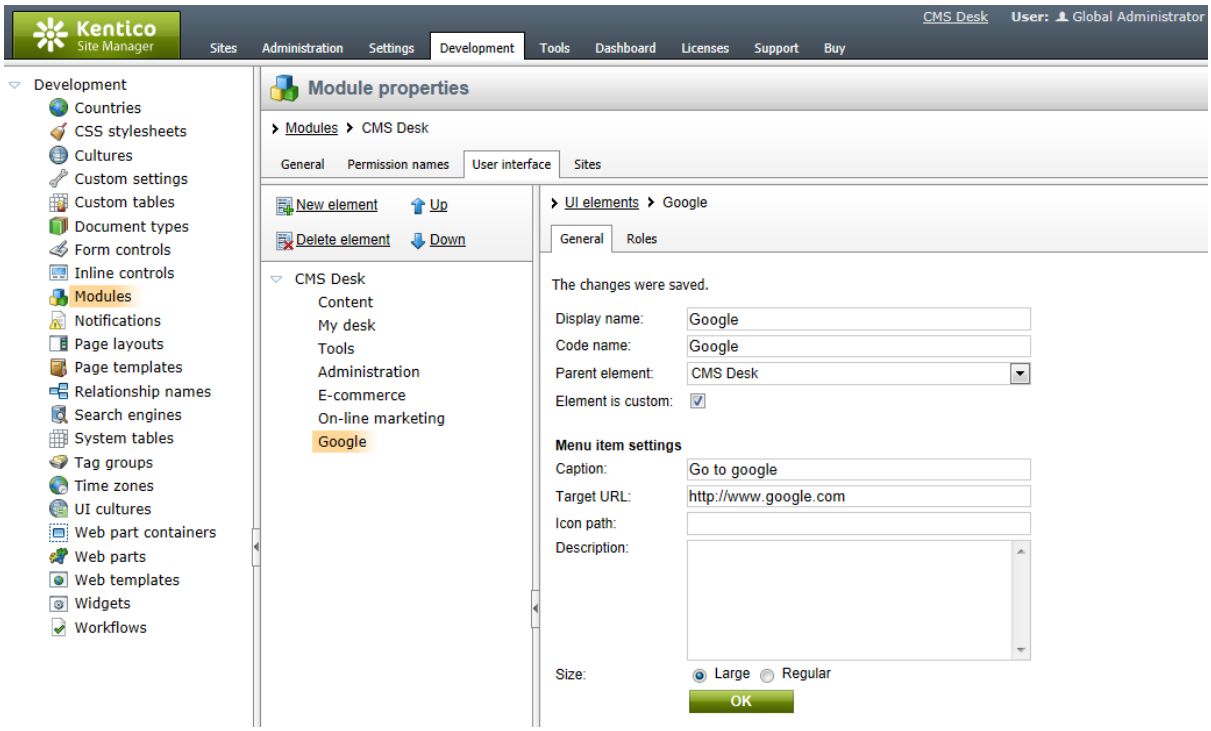


3. In the **New element** dialog, enter the following details:

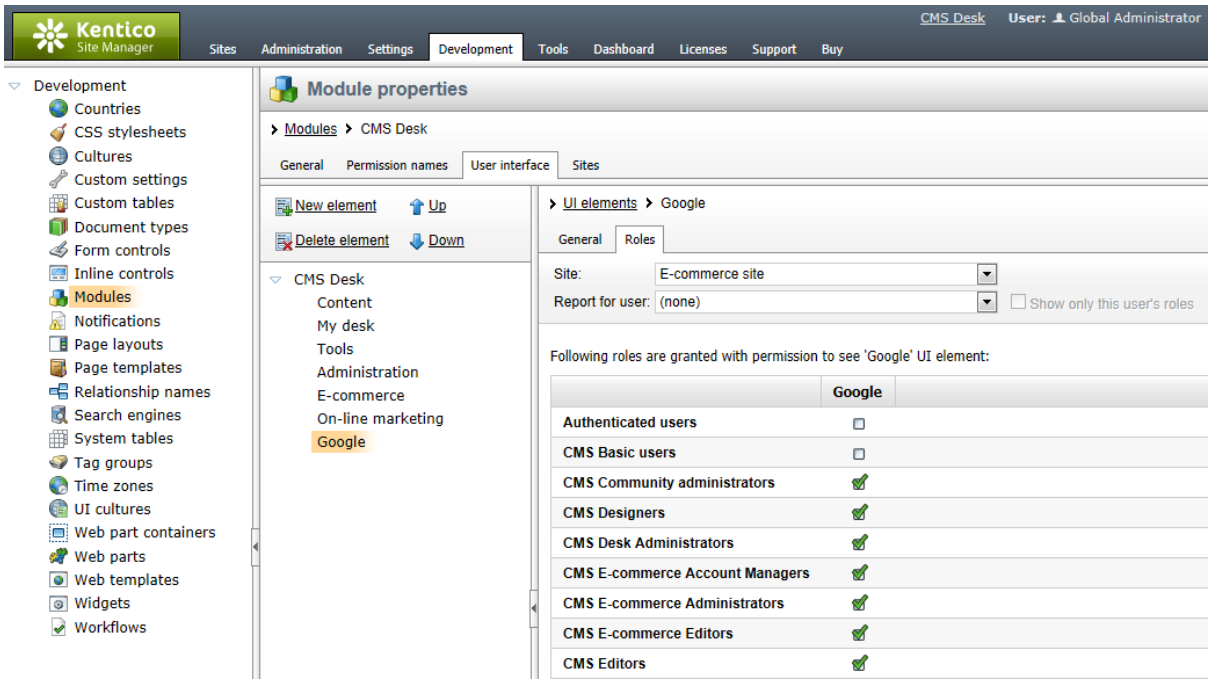
- **Display name**: *Google*.
- **Code name**: *Google*.
- **Element is custom**: *TRUE*.

- **Caption**: *Go to Google*.
- **Target URL**: *http://www.google.com*.
- **Icon path** - please leave blank (icons can be used only for menu items, not for tabs).
- **Description** - you can leave blank.
- **Size** - either value (this property applies to the size of the UI element icon; takes effect only for UI elements which are included in a ribbon-like toolbar).

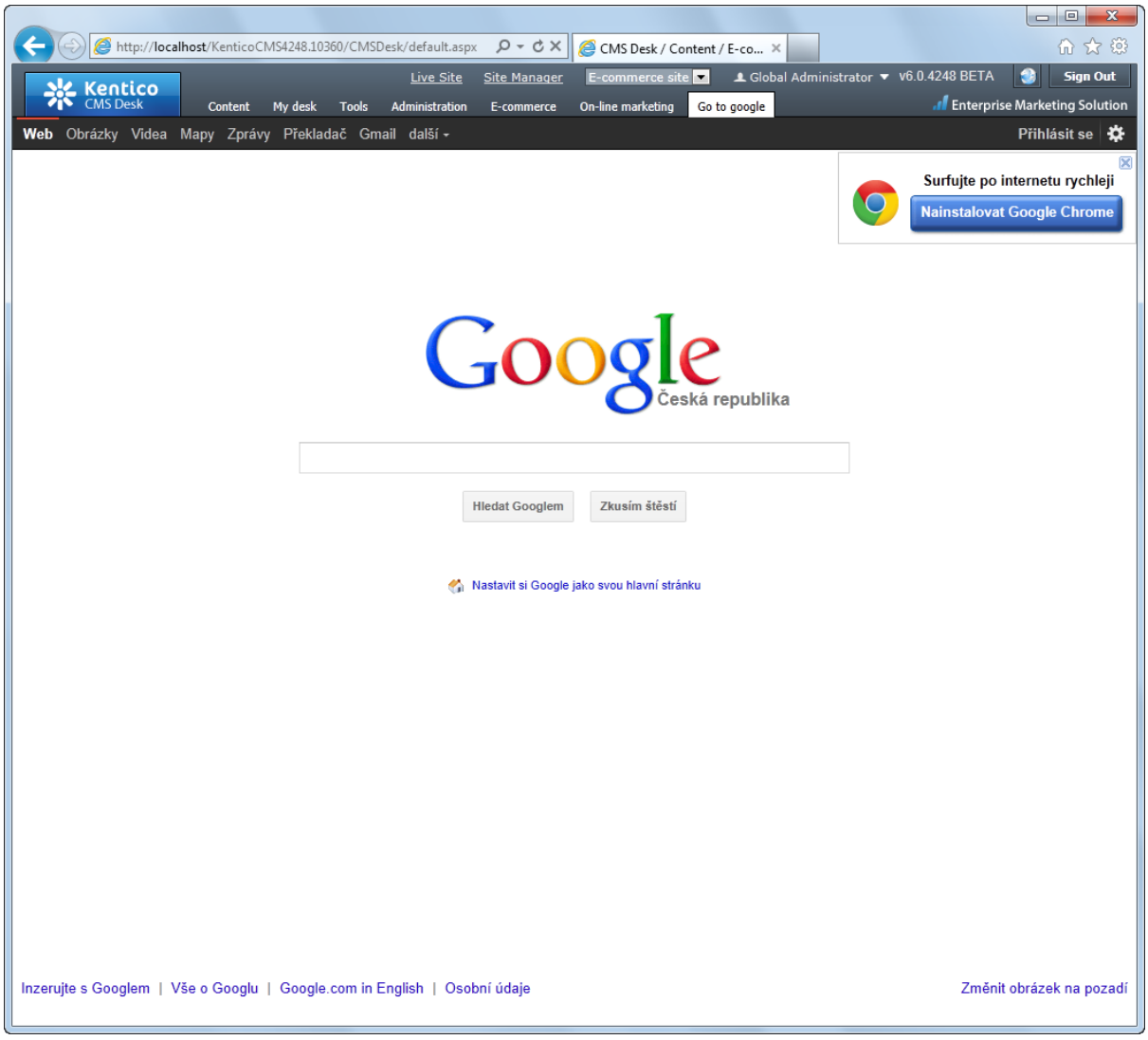
and click **OK**. The new UI element will be added to the tree as you can see in the screenshot below.



4. Switch to the **Roles** tab and enable the UI element for the desired roles.



5. Now switch to **CMS Desk**, logged in as a member of one of the roles enabled in the previous step. You will see the new **Go to Google** tab next to the original six tabs as depicted below. If you click the tab, you will be able to view Google title page within your CMS UI.



Please note

If you try to add your own page that needs scrolling, the scrollbars will not be displayed automatically in the top level menu; i.e. under **CMS Desk header tabs**. This is because pages displayed under these tabs handle scrolling in a special way.
To enable page scrolling here, it is recommended to create a new ***.aspx* page** containing a **frameset with just one scrolling-enabled frame** pointing to your page.

7.13.7 Memberships

7.13.7.1 Membership overview

This feature can be used to define special types of user membership for the website (or globally for all sites in the system).

Within the security model of Kentico CMS, membership objects perform a function similar to roles. Users that belong to memberships are authorized to perform certain actions on the website, access secured content or similar. However, memberships do not directly allow individual permissions, but instead group together one or more existing roles. When a membership is assigned, it grants the given user the sum of all permissions defined for the contained roles. To learn how memberships can be created, configured and assigned to users in the administration interface, please proceed to the Managing memberships topic.

Typically, a membership will be associated with a certain product, which can then be purchased by users through the website's e-commerce features. This will allow them to gain access to restricted sections of the website or other types of premium content for a specified amount of time. For additional information about how you can offer various types of website membership as products, please see the Product types -> Membership topic in the E-commerce Guide.

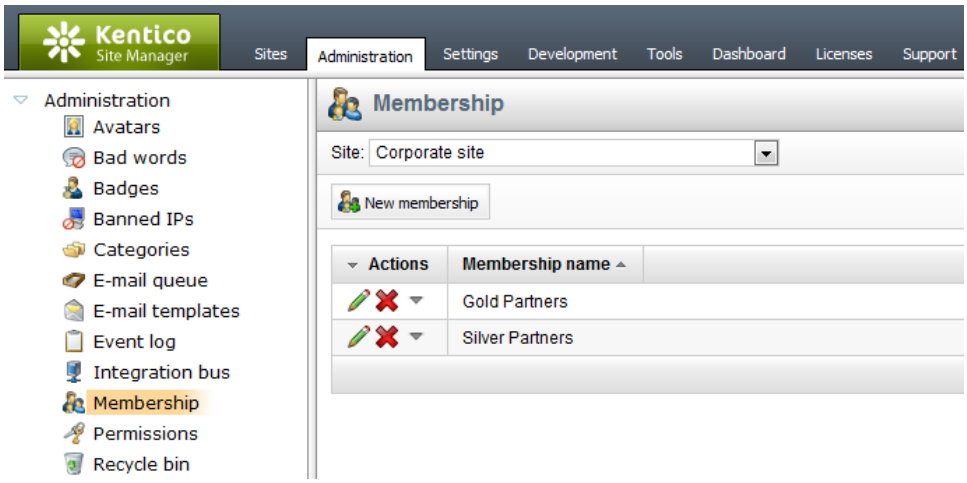
Memberships versus roles

Memberships are mainly intended to be used in combination with e-commerce for live site users and customers, or for other specific purposes where an additional security layer that groups together multiple roles is useful.

If you need to define authorization options for different types of users, such as content editors or administrators for specific modules, it is recommended to do so directly using standard roles.

7.13.7.2 Managing memberships

The management interface for memberships can be found in the **Administration -> Membership** section, both in **CMS Desk** and **Site Manager**.



Memberships can either be assigned to a specific site or defined as global objects that are available for all sites. In **CMS Desk**, only memberships that belong under the current site can be managed. When in **Site Manager**, you can select the site context using the **Site** drop-down list at the top of the page. To access the list of all global memberships in the system, choose the *(global)* option.

It is possible to edit () or delete () the memberships displayed in the list. New memberships can be created for the selected site (or globally) by clicking  **New membership**. You can specify the following properties when adding a new membership:

- **Membership name** - sets a name for the membership which is displayed in the administration interface.
- **Membership code name** - sets a name that serves as an identifier for the membership.
- **Membership description** - can be used to enter an optional text description for the membership.

There are four tabs available when editing () a membership:

General

On this tab you can edit the same properties that were specified when the membership was created.

Roles

This tab is used to define which roles the membership should contain. When the membership is assigned to a user, it will grant all permissions allowed by the specified roles until it expires.

To add roles, click the **Add roles** button and check the boxes next to the appropriate roles in the displayed selection dialog. Only roles that belong under the same site as the membership can be chosen (global memberships may only contain global roles). Roles can be removed from the membership at any time using the checkboxes in the list together with the **Remove selected** button.

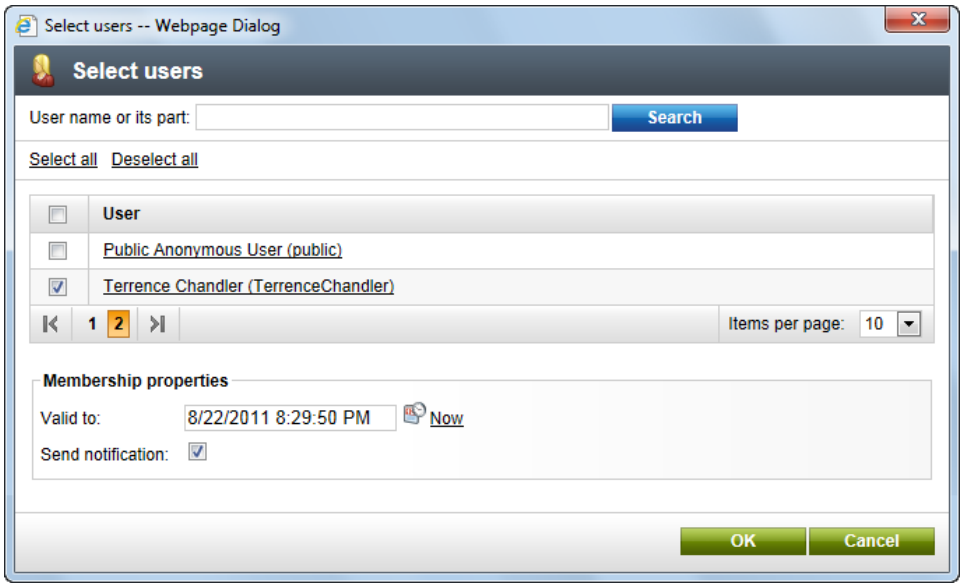
If you need to add a role under multiple memberships, you can save time by navigating to **Administration -> Roles** and editing () the given role on its **Memberships** tab. Here you may easily assign it to any number of memberships using a single action.

Users

Here you can view which users are assigned to the currently edited membership. For as long as their membership is valid, the listed users will be authorized to perform any actions allowed for the roles that the membership contains, as defined on the **Roles** tab.

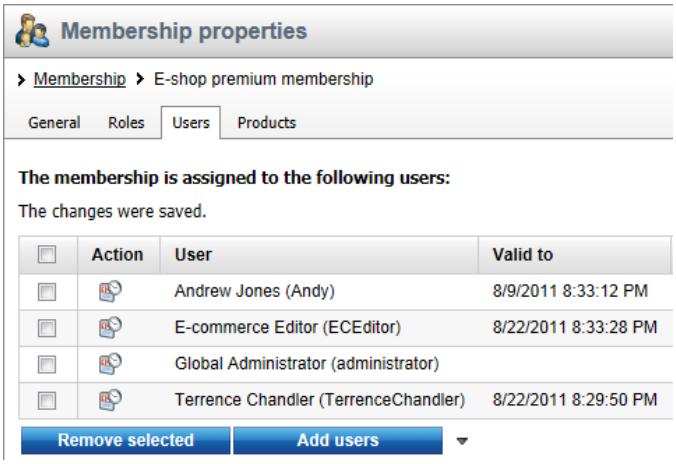
Typically users will be added automatically when they purchase the membership as a product, so it is not necessary to manually assign them one by one. The purpose of this tab is to allow administrators to monitor the membership and override its settings if necessary.

If you wish to add users, click the **Add users** button and check the boxes next to the appropriate users in the displayed selection dialog.



Only users who are assigned to the same site as the membership can be chosen (global memberships may be assigned to all users in the system). You can enter a name or its part into the textbox above the list and click **Search** to quickly find any user. The **Valid to** field can be used together with the **Calendar** button to specify the exact date and time when the membership should expire for the given users. If this field is left empty, the users will be assigned to the membership for an unlimited time period. If you set an expiration date for the membership, you can also check the **Send notification** box to enable e-mail reminders that will be sent to the selected users before the membership becomes invalid.

Click **OK** to apply any changes.



The **Change validity** action that is available for every listed user may be used to prolong or shorten the time interval for which the user should be assigned to the membership. This way you can set an expiration date or reactivate expired memberships for users.

Products

This tab displays a list of products with which the membership is associated, as well as additional

details. When purchased, these products assign the membership to the customer along with the authorization options that it provides. Membership as a type of product is described in the Product types -> Membership topic of the E-commerce Guide.

Setting memberships for users

It is also possible to directly manage the memberships of individual users via the administration interface in **CMS Desk / Site Manager -> Administration -> Users**. When editing a user, simply switch to the **Membership** tab where you can add or remove memberships to/from the user. Please note that global memberships can only be assigned by global administrators. Using the approach described above, you can set an expiration date for each of the memberships assigned to the user.

Displaying memberships on the live site

The My account web part from the **Membership -> Logon & Registration** category can be used to allow users to view a summary of their current memberships on the live site. If its **Other tabs -> Display my membership** property is enabled, the web part will display a list of all memberships assigned to the current user and their expiration dates.

The web part may also be configured using its **Memberships -> Memberships page URL** property to generate a link to a page where users can buy new memberships for the website or renew existing ones.

Membership expiration reminders

To help users keep track of their memberships and ensure that they know when it is necessary to renew a paid membership, the module includes an automatic notification feature.

There is a global scheduled task named **Membership reminder** that periodically checks memberships (once per day by default) and sends a notification e-mail to users whose membership will expire within the amount of days specified by the **Site Manager -> Settings -> Security & Membership -> Send Membership reminder (days)** field. You can configure this task and setting as needed.

The following memberships will expire soon:

- My paid membership, expires 8/17/2011 9:14:01 AM

To renew it, please follow these steps:

1. In My profile section on My memberships tab click the Buy membership button. You will be redirected to the Buy membership page.
2. Choose the required membership and add it to your shopping cart.
3. Finish your order.
4. Once the order is paid, your membership will be renewed.

This is an automatic reminder, please do not respond.
Thank you.

These reminders are only sent for memberships that were assigned with the **Send notification** flag enabled and for those that were purchased as a product with a limited duration. The content of the reminder e-mails is taken from the **Membership - Expiration notification** e-mail template, which can

be edited via the **Administration -> E-mail templates** interface.

When editing this e-mail template, you can use the **{% MembershipsTable %}** context macro to insert a list of all memberships that will soon expire for the given user. This list must be formatted via a **Text/XML** type transformation, which you can specify through the *ApplyTransformation* method, for example:

{%MembershipsTable.ApplyTransformation ("Ecommerce.Transformations.Order_MembershipsTable")%}

In addition to this special macro, you can also use all other standard macro expressions in the templates. See the Development -> Macro expressions chapter of this guide for more information about macro expressions in Kentico CMS.

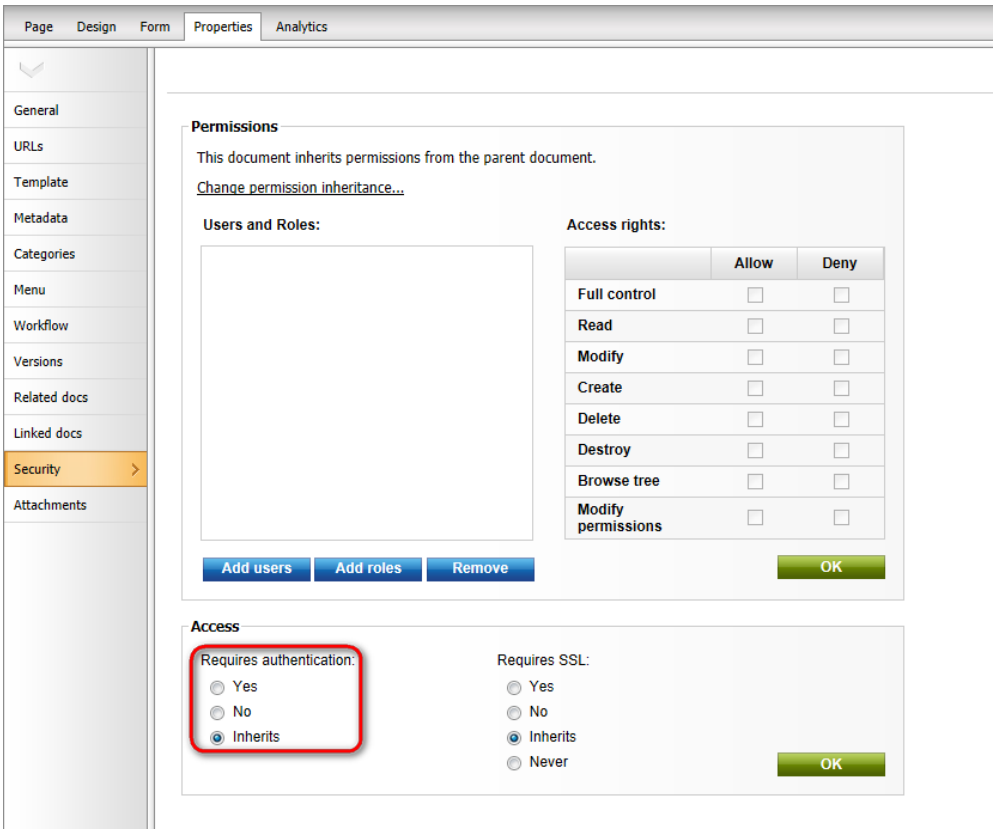
7.13.8 Security

7.13.8.1 Secured website areas

Kentico CMS allows you to easily create secured website areas that are accessible only by authenticated users. When an non-authenticated (public) user comes to the secured section, they are redirected to the logon page specified for the site at **Site Manager -> Settings -> Security & Membership -> Website logon page URL**.

You can mark any section of the website as a secured site area by setting **Properties -> Security -> Requires authentication** to:

- **Yes** - page is secured, authentication is required to access it
- **No** - authentication is not required to access the page
- **Inherits** - value of the setting is required from the parent page



Configuration of a secured site area

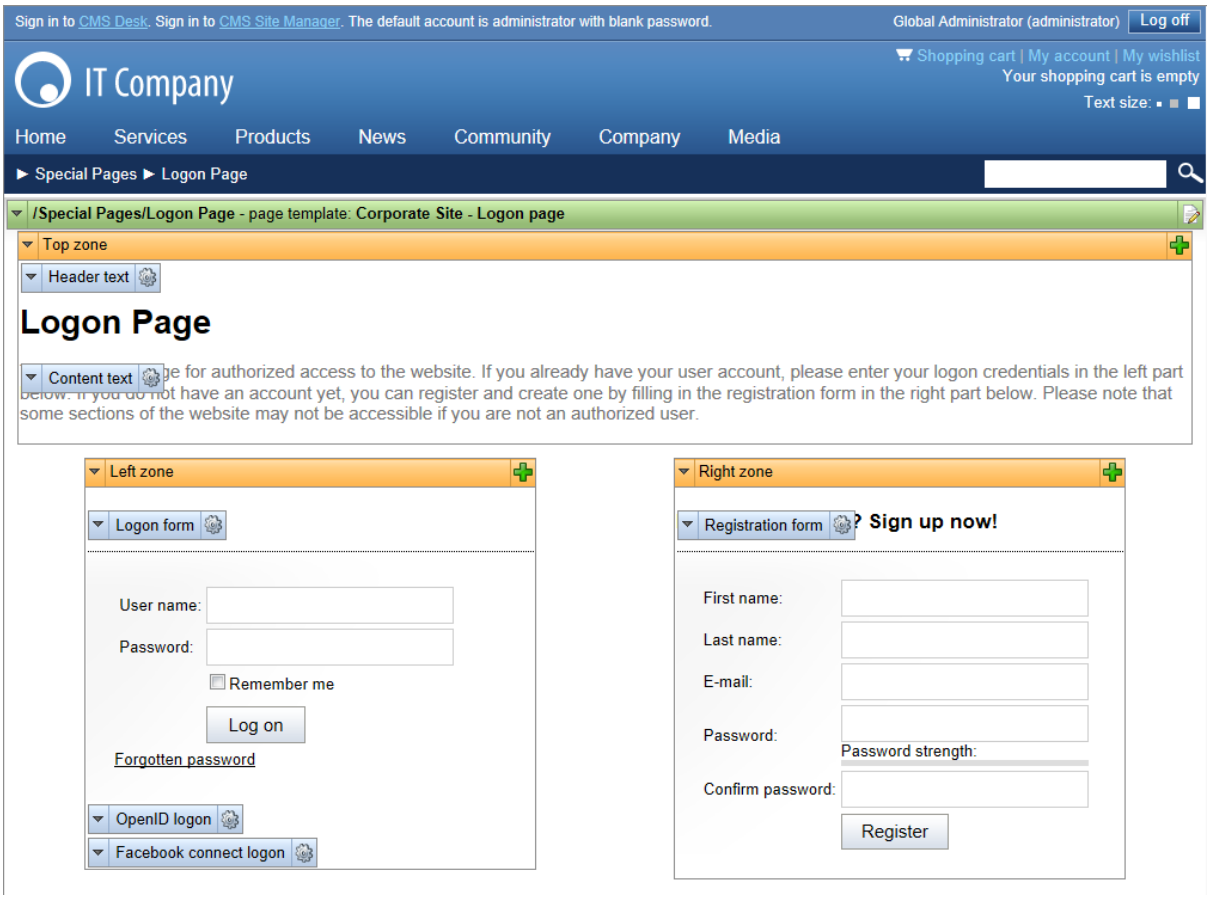
This example explains how to secure the **Products** section of the sample Corporate Site.

1. Sign in as administrator to **CMS Desk**. Go to the **Content** section and click the **Products** document in the content tree.

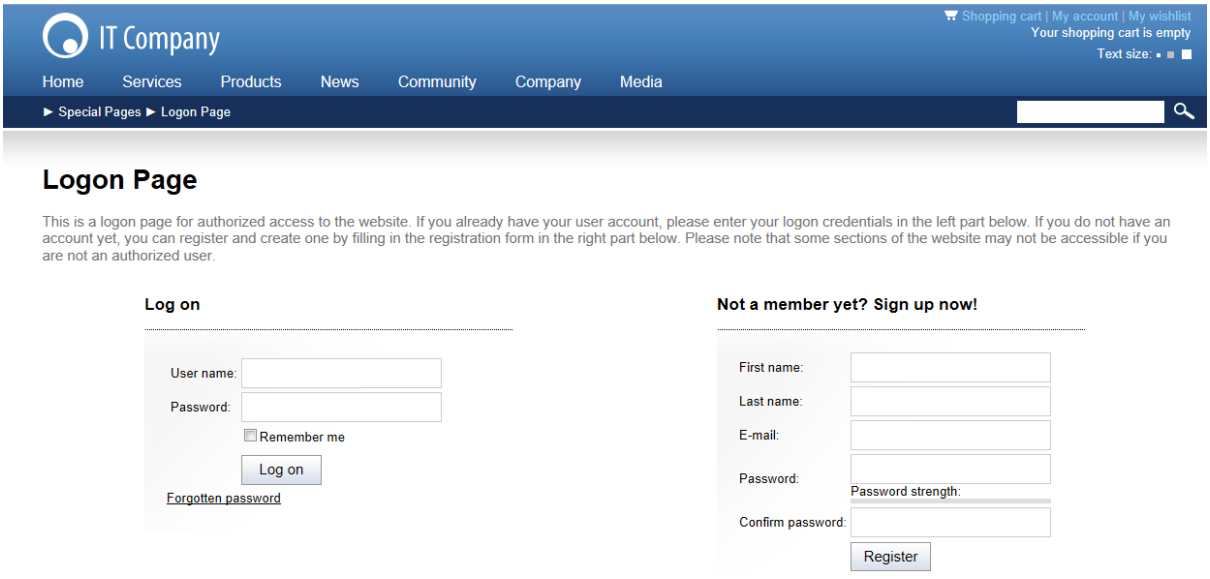
2. Click **Properties -> Security**. Set the value of the **Requires authentication** attribute to **Yes** and click **OK**.

3. Go to **Site Manager -> Settings -> Security & Membership** and choose the *Corporate Site* site in the drop-down list. Make sure the **Website logon page URL** is set to *~/SpecialPages/Logon-page.aspx*. This is the URL of the site's logon page. You can either use the system logon page *~/cmspages/logon.aspx* or you can define your own as demonstrated on the sample Corporate Site.

4. Go to **CMS Desk -> Content**, click the *Logon Page* document under the *Special Pages* folder and select the **Design** tab. As you can see, the page is based on the *Corporate Site - Logon page* page template that contains the *Logon form web* part and the *Registration form* web part.



5. Sign out and click **Products** in the main menu. You are redirected to the logon form:



6. Sign in as administrator and you will see the Products section.

Checking access to page content

Page content is not secured by default, even if the current user is has the **Read** permission denied for the given page. You need to configure this either by setting **Check permissions** to true in the Editable region web part properties (local configuration) or globally by setting the value in **Site Manager -> Settings -> Security & Membership -> Check page permissions** to one of the following values:

- **All pages** - permissions will be checked for all documents on the website.
- **No page** - permissions will not be checked for any documents.
- **Secured areas** - permissions will be checked only for documents that are configured to require authentication.

If a user is not authorized to read a page, the *Access denied* page will be displayed to them. You can configure a custom access denied page by specifying its URL in the **Site Manager -> Settings -> Security & Membership -> Access denied page URL** field.

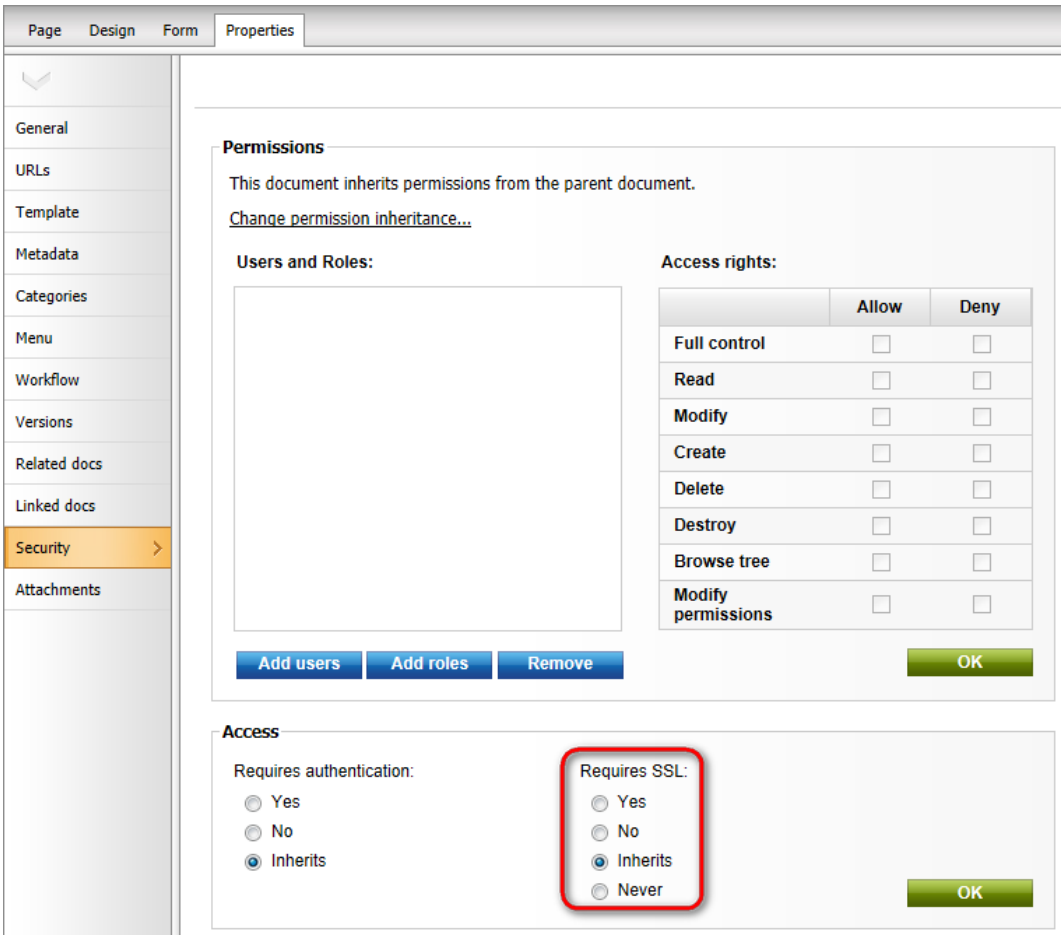
7.13.8.2 SSL (HTTPS) support

Kentico CMS allows you to specify which of the website's pages should only be accessible over the secured SSL protocol. When a user tries to open such a page with the standard HTTP protocol, they will automatically be redirected to the secured version of the same page (using the **https** URL scheme).

Please note: Kentico CMS does not configure your website for SSL/HTTPS. It may only be used to automatically redirect users to the appropriate URL. You need to perform the configuration itself manually using the standard IIS settings, as described in IIS documentation or in this article: http://msdn.microsoft.com/cs-cz/magazine/cc301946(en-us).aspx

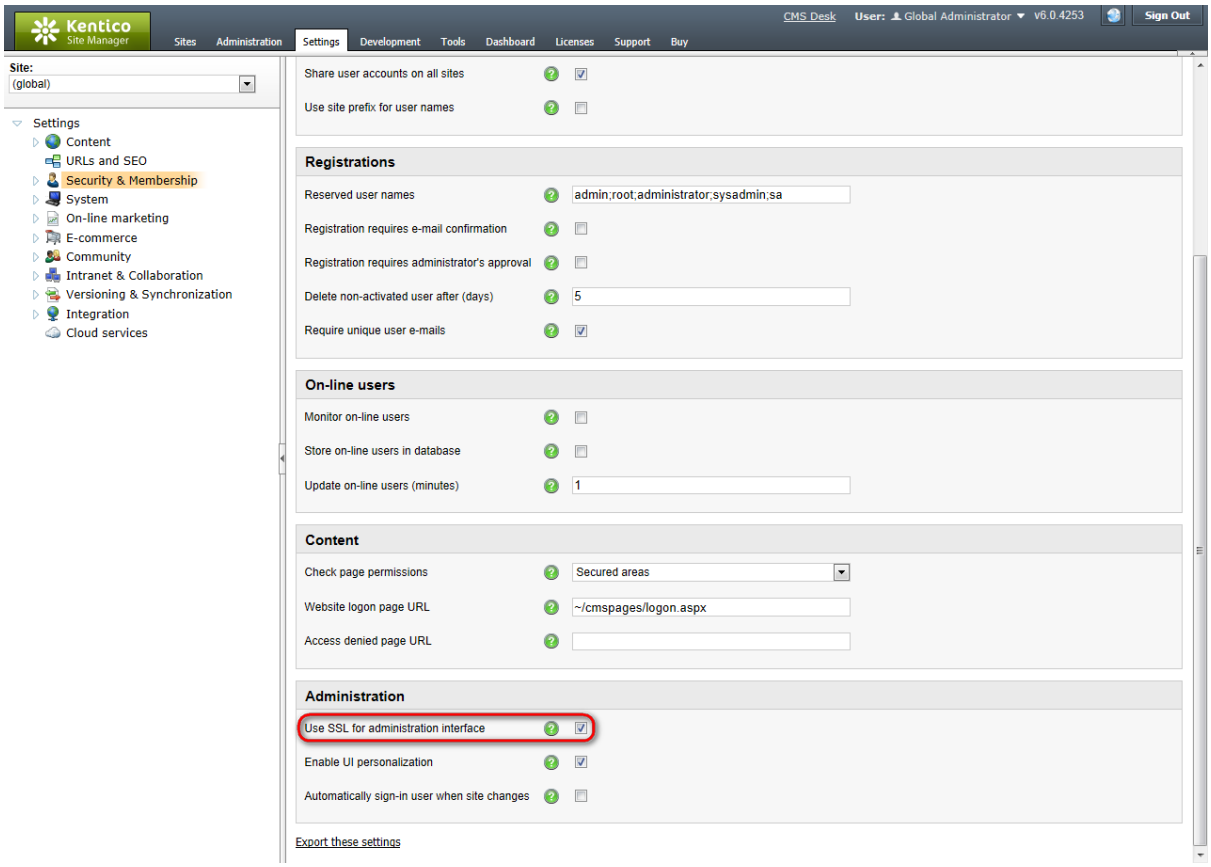
In order to specify that a page should only be available through the SSL protocol, you need to go to **CMS Desk -> Content** and select the corresponding document in the content tree. Then you can choose one of the following options in **Properties -> Security -> Requires SSL**:

- **Yes** - users attempting to access the page will always be redirected to the HTTPS version of the page's URL.
- **No** - users will not be explicitly redirected to a secured URL when they access the page, but the protocol used in the current URL will remain unchanged (i.e. if a user navigates to the page from another page that is secured, this page will also use SSL).
- **Inherits** - the settings defined for the parent document will be used.
- **Never** - users trying to access the page through a secured URL will be explicitly redirected to the unsecured version of the page.



Securing the administration interface

You may also configure the same functionality for all pages that belong to the administration interface (**CMS Desk** and **Site Manager**). To do this, go to **Site Manager -> Settings -> Security & Membership** and check the **Use SSL for administration interface** field in the **Administration** category.



Now all editors and administrators will be redirected to pages using the **https** URL scheme when they log in. Please note that this setting is applied to all sites in the system, so it is only available if you select the *(global)* option from the **Site** drop-down list.

Indirect SSL scenarios

In some cases, SSL decryption and encryption may not be performed directly by your application's server, but instead via a reverse proxy, such as an SSL offload device (for example an SSL accelerator in a load balanced web farm scenario). This means that requests are forwarded to the application internally using the standard HTTP protocol, even when the client accesses the page through HTTPS. If the settings described in the sections above are enabled for the website, it may result in a redirection loop.

You can solve this issue by adding custom code to the application's request handlers. It is necessary to appropriately set the **IsSSL** static property of the **CMS.GlobalHelper.URLHelper** class. If set to *true*, the system will treat all requests as secured, regardless of their URL format, and redirection to HTTPS page versions will not be performed by the application. Of course, it is necessary to correctly identify which requests originally used SSL, e.g. by checking the request headers.

Example

1. Open your web project, expand the **App_Code** folder (or **Old_App_Code** if you installed the project as a web application) and add a new class called **SSLRequestLoader.cs**.

2. Edit the class and add the following references:

[C#]

```
using CMS.SettingsProvider;
using System.Collections.Specialized;
using CMS.GlobalHelper;
```

3. Delete the default class declaration and its content. Instead extend the **CMSModuleLoader** partial class and define a new attribute for it as shown below:

[C#]

```
[SSLRequestLoader]
public partial class CMSModuleLoader
{
    /// <summary>
    /// Module registration
    /// </summary>
    private class SSLRequestLoaderAttribute : CMSLoaderAttribute
    {
        ...
    }
}
```

4. Enter the following code into the **SSLRequestLoaderAttribute** class:

[C#]

```
/// <summary>
/// Called automatically when the application starts
/// </summary>
public override void Init()
{
    // Assigns a handler called before each request is processed
    CMSRequestEvents.Begin.Before += HandleSSLRequests;
}


// Checks requests if they are forwarded as SSL
private static void HandleSSLRequests(object sender, EventArgs e)
{
    if ((HttpContext.Current != null) && (HttpContext.Current.Request != null))
    {
        // Loads the request headers as a collection.
        NameValueCollection headers = HttpContext.Current.Request.Headers;

        // Gets the value from the X-Forwarded-Ssl header.
        string forwardedSSL = headers.Get("X-Forwarded-Ssl");

        URLHelper.IsSSL = false;
```

```
        // Checks if the original request used HTTPS.
        if (forwardedSSL == "On")
        {
            URLHelper.IsSSL = true;
        }
    }
}
```

The override of the **Init()** method (executed automatically when the application starts) is used to assign a handler that will be called before each request is processed.

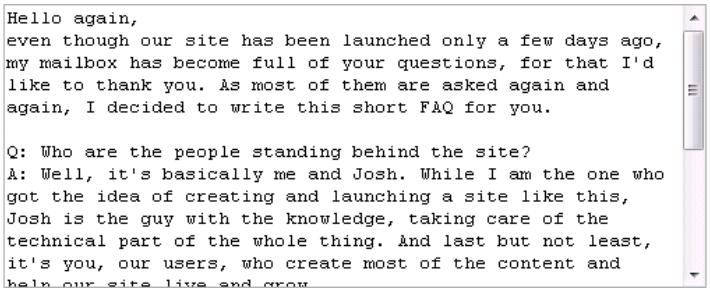
This example uses the **X-Forwarded-Ssl** header to check if the original request was submitted via HTTPS before it was forwarded to the application by the SSL offload device or load balancer. If this is the case, the **IsSSL** property is set to *true* and the system will process the request as if it used the HTTPS protocol.

The proxy device may use a different method to identify requests that were originally secured by SSL, so you will have to write a condition that fits your specific scenario (another typical approach is to check if the value of the **X-Forwarded-Proto** request header is *https*). You may also include additional custom code to fulfill any other security requirements, such as validation of the proxy's IP address.

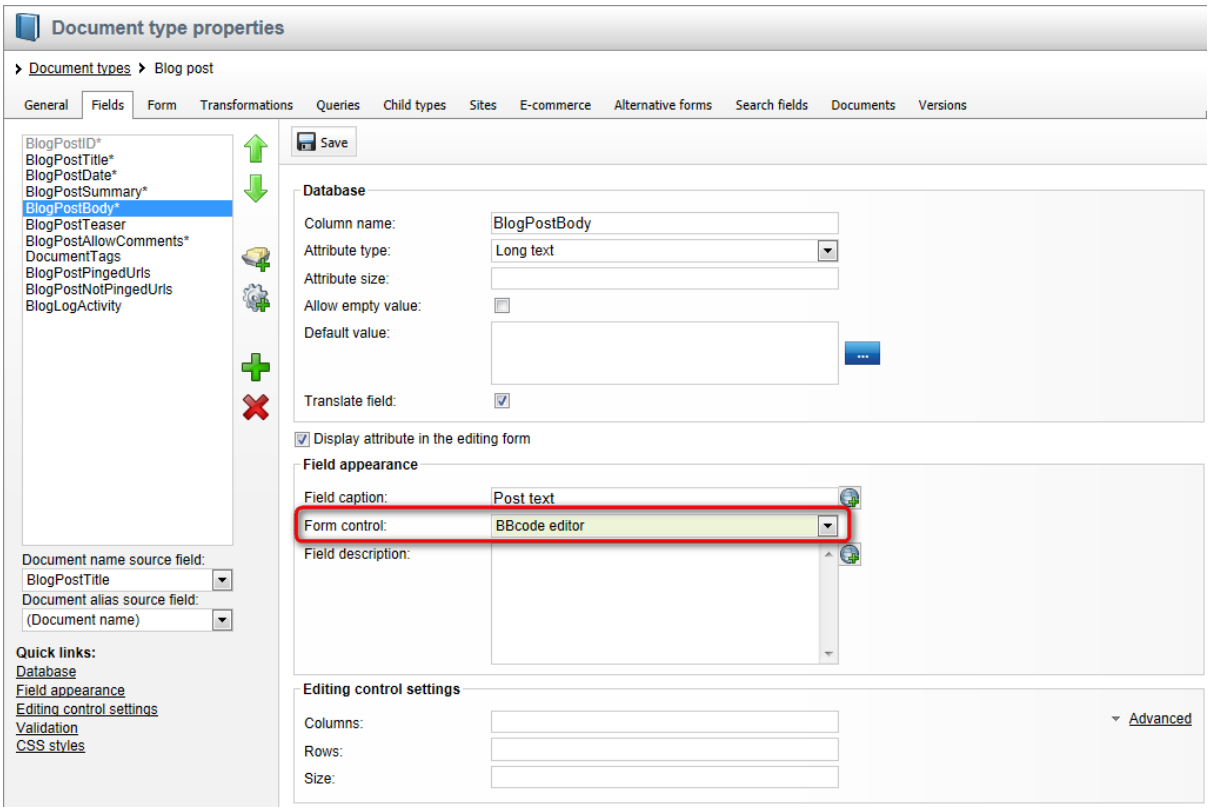
7.13.8.3 Cross site scripting (XSS)

For your websites to be Cross site scripting (XSS) safe, the following rules need to be followed:

1. **Do not use the built-in WYSIWIG editor** (**HTML area (Formatted Text)** form control) to allow live site users to enter text (e.g. in user profiles, forums, etc.). Instead, use the **BBcode editor**.



You can make users use the editor by selecting *BBcode editor* as the value of the **Form control** property when defining fields for document types (or other objects), as shown in the following image.



2. When writing your transformations, use the following method to resolve text entered via the BBcode editor:

```
CMSHelper.CMSContext.ResolveDiscussionMacros(string inputText)
```

3. When writing your transformations, use the `Eval(string columnName, bool encode)`method with the second parameter enabled to display content of any field whose content was entered by the users, e.g.:

```
<%# Eval("PostSubject",true) %>
```

7.13.8.4 Clickjacking protection

Clickjacking is a type of attack where the attacker tricks a website user into clicking something different that what they see, thus performing an action that may for example reveal confidential data or have any other negative impact on the user. In a typical clickjacking scenario, the attacker places a transparent frame with a page, that contains a button or a link, over another element on a website. The underlying element can be an image or a video, which the user expects to play when they click it. Instead, they click the concealed link or button. This way the attacker can make the users perform unintended actions, usually on websites the users are authenticated on.

To prevent such attacks, Kentico CMS disallows embedding pages it renders into frames. It does that by including a special entry in the HTTP response headers:

```
X-frame-options: SAMEORIGIN
```

The header ensures that pages that are displayed in frames originate on the same server as the parent page. If they don't, browsers do not render them.

This feature is enabled by default for all websites and their pages, however, you can exclude paths where you don't want the header to be used. To do that, add the following key into the appSettings section of your web.config:

```
<add key="CMSXFrameOptionsExcluded" value="/Services" />
```

As a value, you can enter any alias path. All documents under this path will be excluded from the protection. Entering "/" turns off the protection altogether.

7.13.8.5 Configuration of allowed request parameters

In some cases, you may need to use super-secure configuration where any non-standard GET or POST parameter sent to your website results in an error. This allows you to avoid some of the possible vulnerabilities, including cross-site scripting and SQL injection.

This functionality is only used for the website, not for the administration interface.

How to configure the allowed parameters

First, you need to enable allowed parameter checking in the web.config file by setting the value CMSCheckParameters to true:

```
<add key="CMSCheckParameters" value="true" />
```

If you're not sure which parameters cause the problem, you can turn on reporting using the following web.config key:

```
<add key="CMSReportCheckParameters" value="true" />
```

All parameters are defined in the **~/parameters.config** file. The schema of the file is described in the file itself and it's rather simple. For every page or site section, you need to define a new <location> section with path specifying the page and allowed form (POST) and query (GET) parameters. The following example allows URL parameter pagenumber in the whole products section of the website:

```
<location path="/products/%">
  <queryparameters>
    <allow param="pagenumber" />
  </queryparameters>
</location>
```

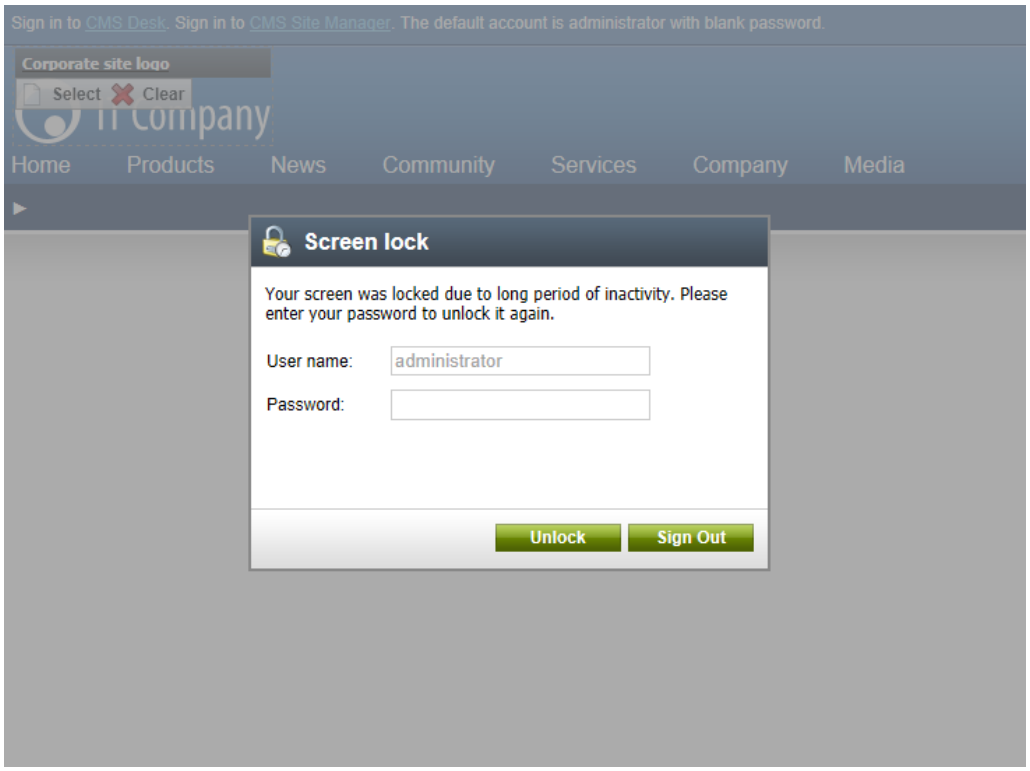
The **path** location specifies the path of the pages based on their alias path in Kentico CMS, while the **page** location is used for single pages that are not part of the Kentico CMS content (custom applications, etc.). The page location starts at the root of the web application and is used without slash (/) at the beginning.

Default allowed parameters

The common parameters of ASP.NET web forms and URL parameters **aliaspath** and **lang** are allowed by default.

7.13.8.6 Screen locking

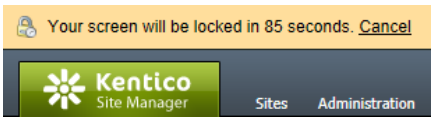
To mitigate the risk of a security breach caused by a user, who is signed in to the administration interface and leaves their workstation unattended, Kentico CMS allows you to set up screen locking. This feature locks the working area of the browser with the administration interface after a specified time of inactivity, requiring to enter a password to unlock it.



To enable screen locking, open **Site Manager** and navigate to **Settings -> Security & Membership -> Protection -> Screen lock**, then turn on the **Enable screen lock** setting.

The **Screen lock** settings group offers the following additional settings:

- **Lock interval** - time of user inactivity in minutes until their screen is locked.
- **Warning interval** - time in seconds before the lock during which a warning with countdown to the lock will be displayed.



During the warning interval, users can click **Cancel** to start counting the lock interval again.

7.13.9 User registration

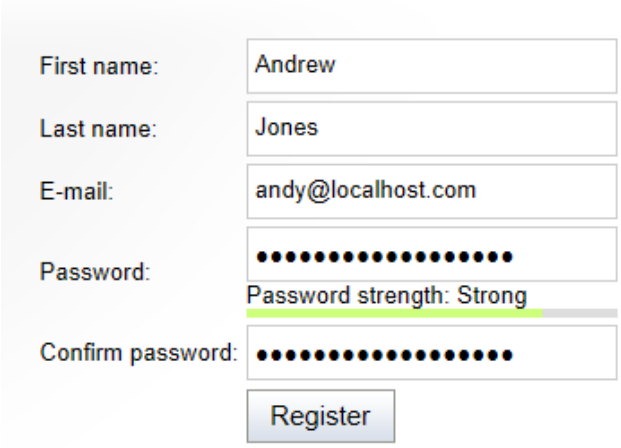
7.13.9.1 Available registration web parts

There are three different ways how you can let site visitors register on your site:

- Using the Registration form web part.
- Using the Custom registration form web part.
- Via Third-party authentication.

7.13.9.2 Registration form web part

The **Registration form** web part is a ready-made web part that can be used right out of the box. You can just place it on any page of your website without setting any web part properties. However, if you want to modify the default behavior of the web part, you can set a number of properties. You can find a detailed description of these properties in the Kentico CMS Web Parts reference.



The password field in the form supports validation according to a password policy specified for the website and also calculates and displays the relative strength of the currently entered password. To learn more about how these security options can be configured, please see the Password management -> Password strength policy topic.

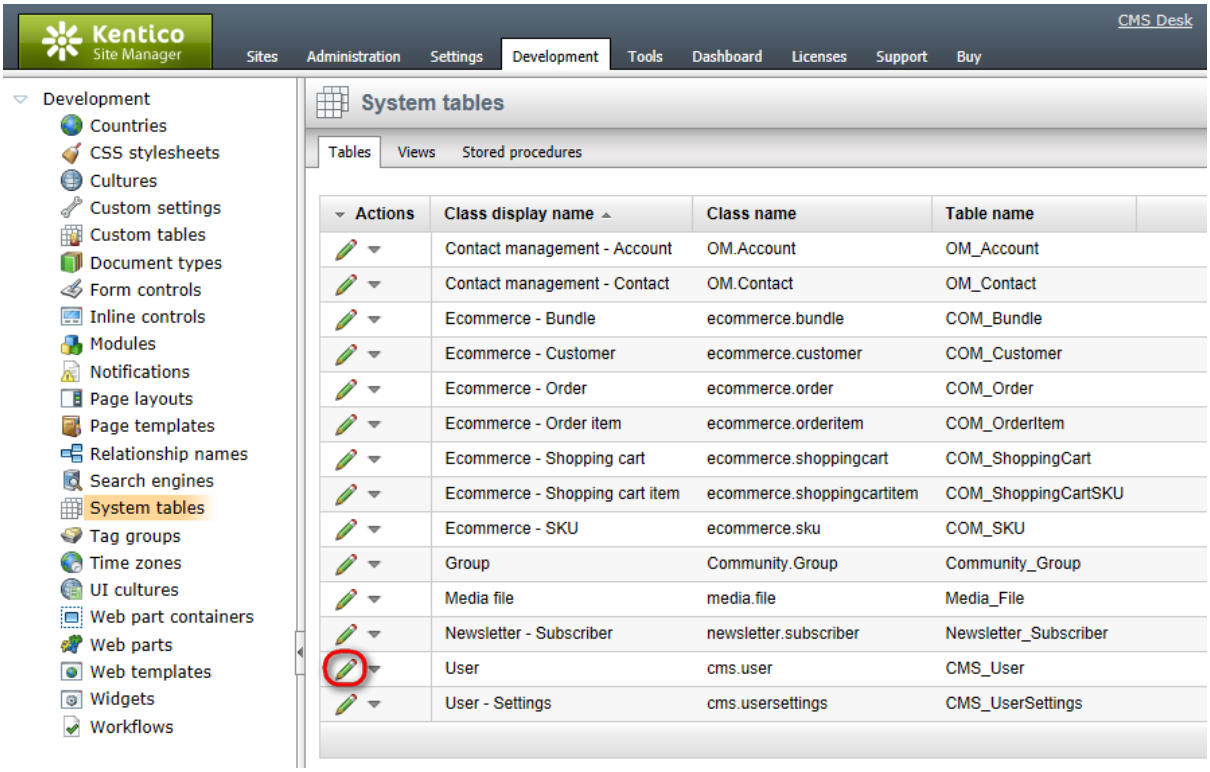
7.13.9.3 Creating a custom registration form

The **Custom registration form** web part can be used in situations when you want to use a different registration form than the default one provided by the **Registration form** web part. This is typically when you want users to provide different details or when you want to customize the form's layout.

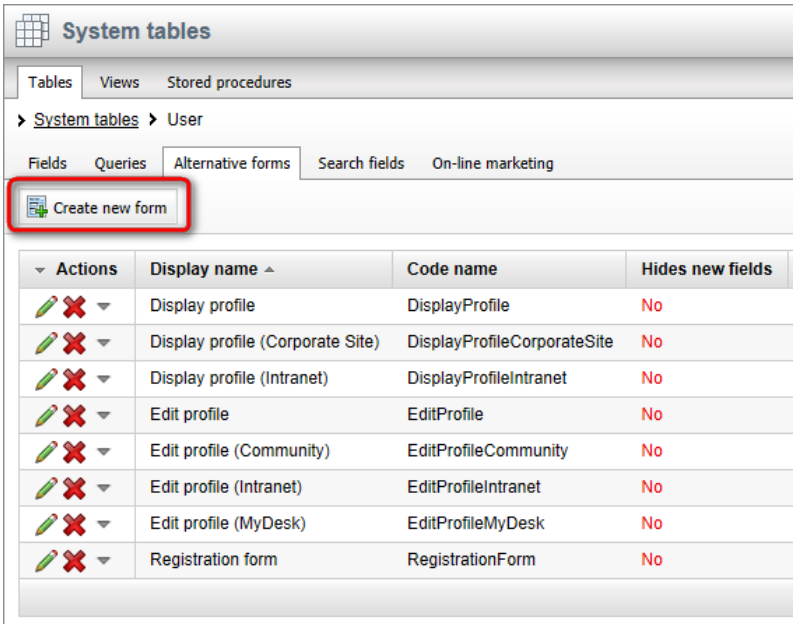
In the following example, you will learn how to use a custom registration form on your site. You will create an **alternative form** and use it for registration via the **Custom registration form** web part. If you

are not familiar with the **Alternative forms** concept, please refer to the Alternative forms chapter first.

1. Go to **Site Manager -> Development -> System tables** and choose to **Edit** (pencil) the **User** system table.



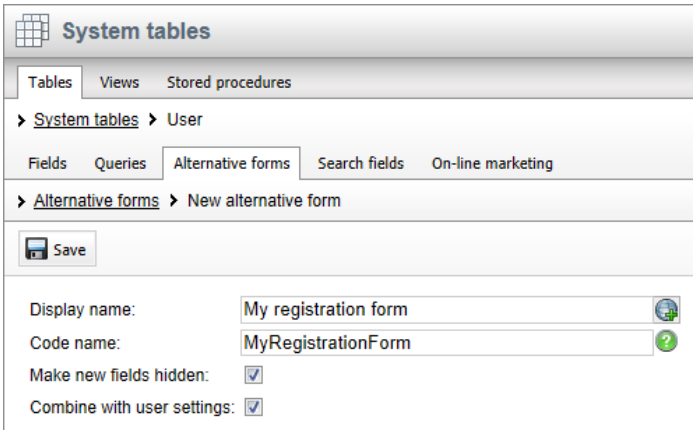
2. Switch to the **Alternative forms** tab and click the **Create new form** button above the list.



3. Fill in the following details:

- **Display name**: My registration form
- **Code name**: MyRegistrationForm
- **Make new fields hidden**: checked; this ensures that new fields added to the system table are not displayed in the alternative form by default.
- **Combine with user settings**: checked; this ensures that all default user fields will be available.

Click **Save**.

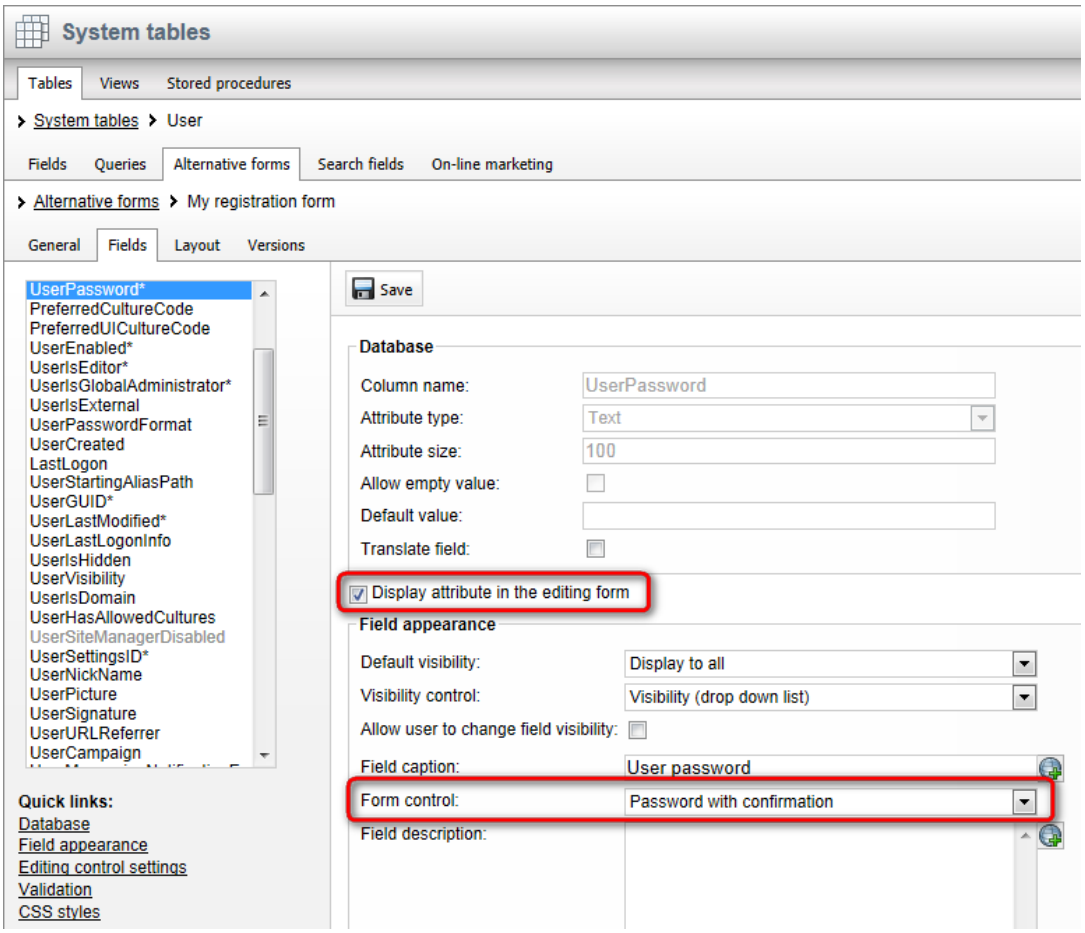


4. Your new form is now created and you are redirected to the editing interface of it. Switch to the **Fields** tab.

On this tab, you can see the fields (columns) defined in the **User** system table. You can select a field from the list on the left. In the right part, you can modify its properties. We will want our registration form to contain the following fields:

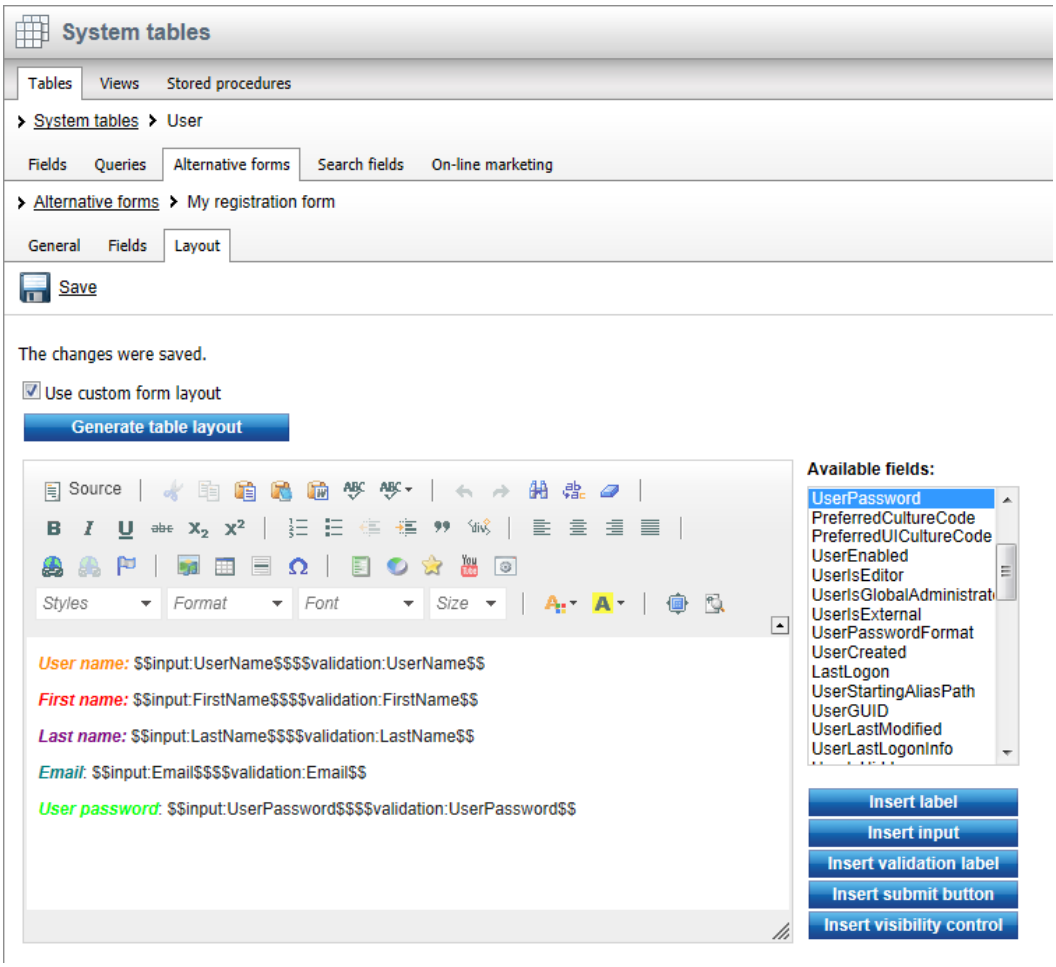
- **UserName**
- **FirstName**
- **Email**
- **UserPassword**

Go through the attributes and check the **Display attribute in the editing form** check-box for those mentioned above, which will make them visible on our registration form. Click **Save** to confirm the changes for every field. Uncheck the check-box for the remaining fields. For the **UserPassword** field, also choose the *Password with confirmation* option in the **Form control** drop-down.



5. You can also switch to the **Layout** tab and define the registration form's layout using the built-in WYSIWYG editor.

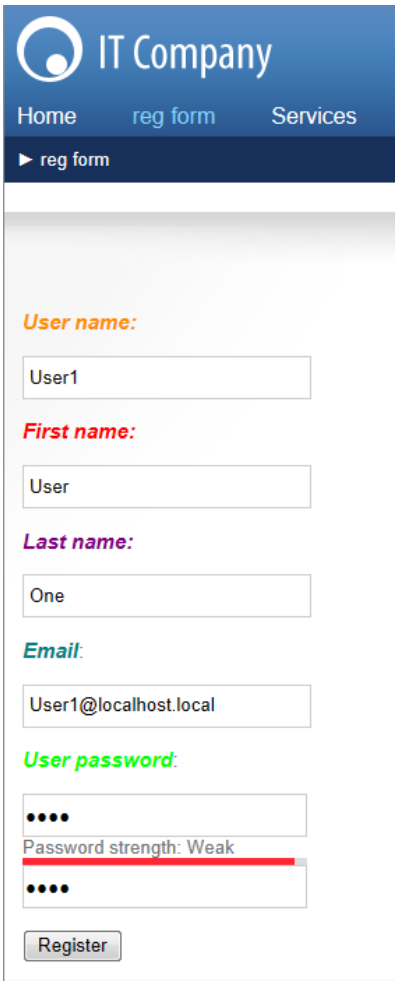
To do it, check the **Use custom layout** check box and click the **Generate table layout** button. A default layout will appear in the editor. Try playing around a bit with the layout or just create something similar to what you see in the screenshot below. Click **Save** when you are finished.



6. Now switch to **CMS Desk**. Choose or create a page where you want to add the registration form and select it in the content tree. Switch to the **Design** tab and add (✚) the **Membership -> Custom registration form** web part to the page. Set the following property of the web part:

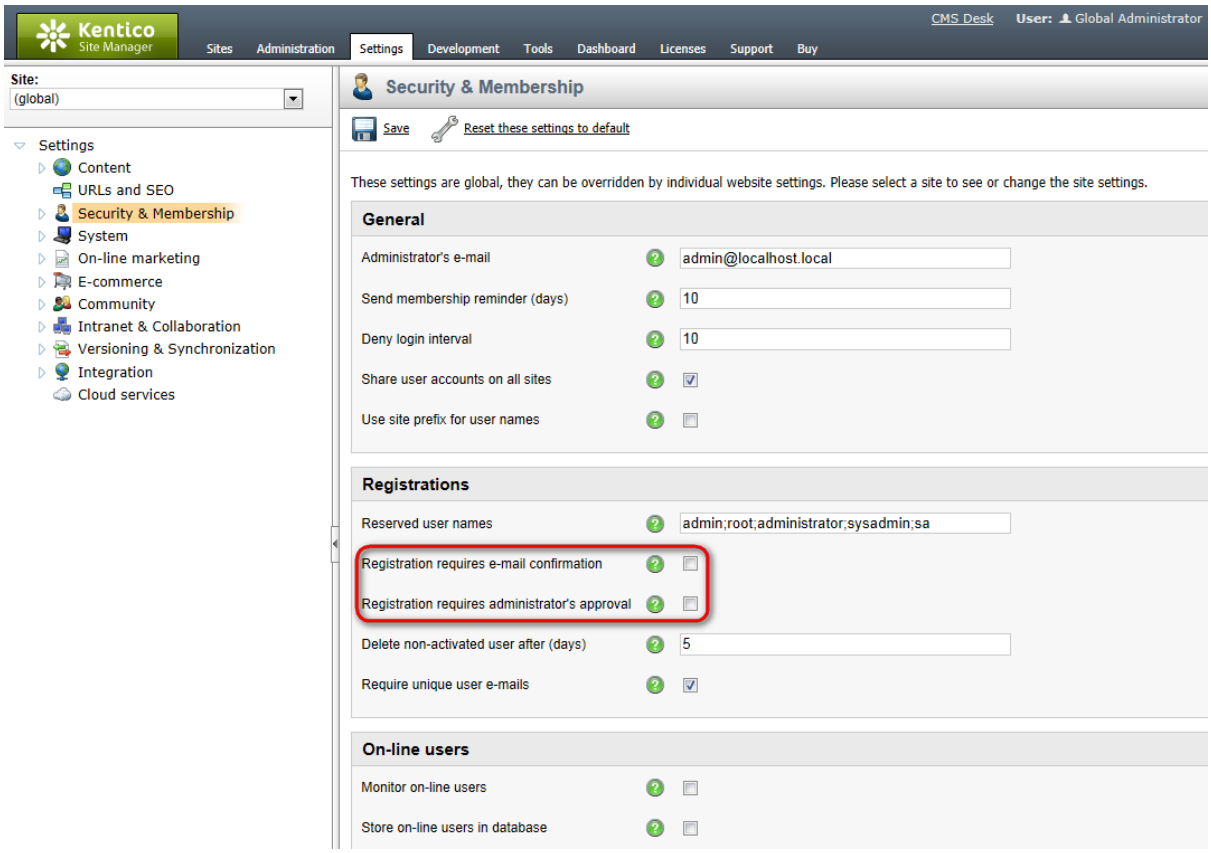
- **Alternative form** - cms.user.MyRegistrationForm

7. If you switch to the live site now and go to your page with the Custom registration form web part, you should see the custom registration form that you created. You can now try to register to your website using the form and verify that the user has been created in **Site Manager -> Administration -> Users**.



7.13.9.4 Registration approval and double opt-in

By default, users can sign-in to the site immediately after successful registration. However, the two options highlighted in the following screenshot can be enabled in **Site Manager -> Settings -> Membership & Security**. By enabling these options, you can include additional steps in the registration procedure.



Registration requires e-mail confirmation

If checked, newly registered users will receive a confirmation e-mail to the e-mail address specified during registration. This e-mail contains a confirmation link that has to be clicked in order to activate the account. The e-mail is based on the **Membership - Registration confirmation** e-mail template.

Thank you for registering at our site. Please click the link below to complete your registration:
http://localhost/KenticoCMS_7.0.4562/CMSPages/Dialogs/UserRegistration.aspx?userguid=80cd1b3a-615d-4308-9f2b-fa186ed9f637

You can find your credentials below:
Username: j.smith@localhost.local
Password: 123456

After clicking the link, the user will be redirected to the page specified in the **E-mail confirmation page** property of the web part used for registration. This page must contain the Registration e-mail confirmation web part to work correctly.

By default, the *~/CMSPages/Dialogs/UserRegistration.aspx* page is used, which displays the following message:

> Your user account is now active. You can sign in using your user name and password. Click here to continue.

The link at the end of the message will redirect the user to the title page of the website. The user can

then log in using the registration details received in the e-mail.

Registration requires administrator's approval

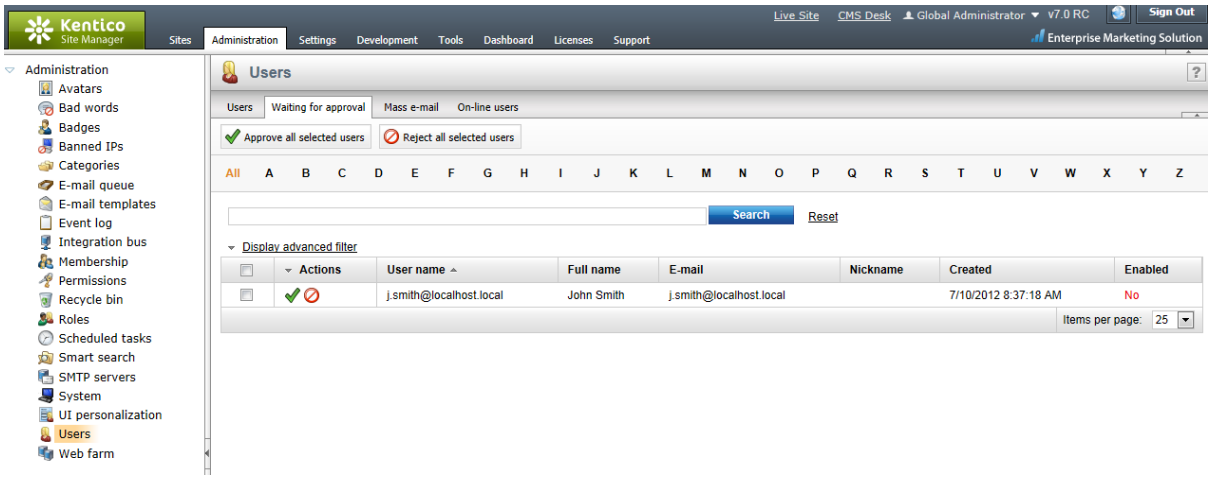
If this option is enabled, users will not be able to sign-in immediately after registration. Their registration will have to be approved by the site administrator. At this point, users will receive an e-mail based on the **Membership - Registration waiting for approval** e-mail template. You can see the default version of the e-mail in the screenshot below.

Thank you for registering at our site CorporateSite. Your registration must be approved by administrator.

Registration details:

Username: j.smith@localhost.local
Password: 123456

If the option is enabled, the **Waiting for approval** tab will be displayed in **Site Manager -> Administration -> Users**. On this tab, site administrators can **Approve** (✔) or **Reject** (⊘) a user's registration.



After the administrator's approval, users receive another e-mail, confirming that their account has been approved and can be used. The e-mail is based on the **Membership - Registration approved** e-mail template. You can see the default appearance of the e-mail in the screenshot below.

Your registration has been approved by administrator. Now you can sign in using your username and password.

Click here to navigate to the web site

Enabling both options

When both of the options mentioned above are enabled, the e-mail with the confirmation link is sent first. After the user's confirmation, registration will have to be approved by the administrator.

Default redirection

If you have one or both of the options enabled, it is important to properly set the **Redirect to URL** property of the web part used for registration. This means that users should not be redirected to any page displaying information about their user account (like the **Members -> Profile** page on the sample Community Starter site), because the account will not be active yet (it is waiting for e-mail activation or approval). Such a page would display an error message, which might be misleading for the users.

E-mail notification addresses

In case you wish to use notification e-mails to inform administrators about new user registrations that require approval, the target address cannot be specified via the registration web part itself if e-mail confirmation is also enabled.

Instead, it must be entered into the properties under the **E-mail settings** category of the Registration e-mail confirmation web part placed on the used confirmation page.

Third party authentication issues

Using double opt-in or registration approval in combination with Third-party authentication services may cause certain problems for first-time users. In these cases, new users are by default created without an e-mail address when they log in for the first time. This means that they cannot activate their account via e-mail confirmation or receive notifications informing that their account must be approved by an administrator.

These issues can be avoided by creating a **Required user data page** where users must enter an e-mail address for their account, as described in the chapter linked above.

7.13.9.5 User registration e-mails

In **Site manager -> Administration -> E-mail templates**, you can find the E-mail templates used for the automatic e-mail notifications related to user registration. Messages based on the following templates are sent to users when they register, depending on the site configuration (please see Registration approval and double opt-in for more information):

- **Membership - Registration** - sent to new users as a welcome e-mail after they successfully register (if the used registration web part is configured to do so).
- **Membership - Registration confirmation** - sent to users after registration if e-mail confirmation (double opt-in) is required.
- **Membership - Registration waiting for approval** - sent to users after registration if an administrator's approval is needed to activate the account.
- **Membership - Registration approved** - informs users that their account has been approved by an administrator.

The templates below are used for notifications sent to administrators:

- **Membership - Notification - New registration** - notifies administrators when a new user registers on the site.
- **Membership - Notification - Waiting for approval** - lets administrators know that a new user is waiting for their approval.

Any of these templates can be edited as needed, so you may fully customize the content of the e-mails. You can use the following context macros to include dynamic values in their text. Please note that some of the macros are only available in specific templates.

- **{% FirstName %}** - the first name of the new user.
- **{% LastName %}** - the last name of the new user.
- **{% Email %}** - the e-mail address entered during registration by the user.
- **{% UserName %}** - the user name of the new account. If you are using site prefixes for user names, all occurrences of this macro in e-mail templates should have the prefix trimmed out through the following method: *{%TrimSitePrefix(UserName)%}*
- **{% Password %}** - the password specified for the new account.
- **{% ConfirmAddress %}** - returns the URL of the page where the user can confirm their registration (intended for sites where double opt-in is enabled).
- **{% HomePageURL %}** - resolves into the URL of the site's home page. This is only available in the *Registration approved* template.

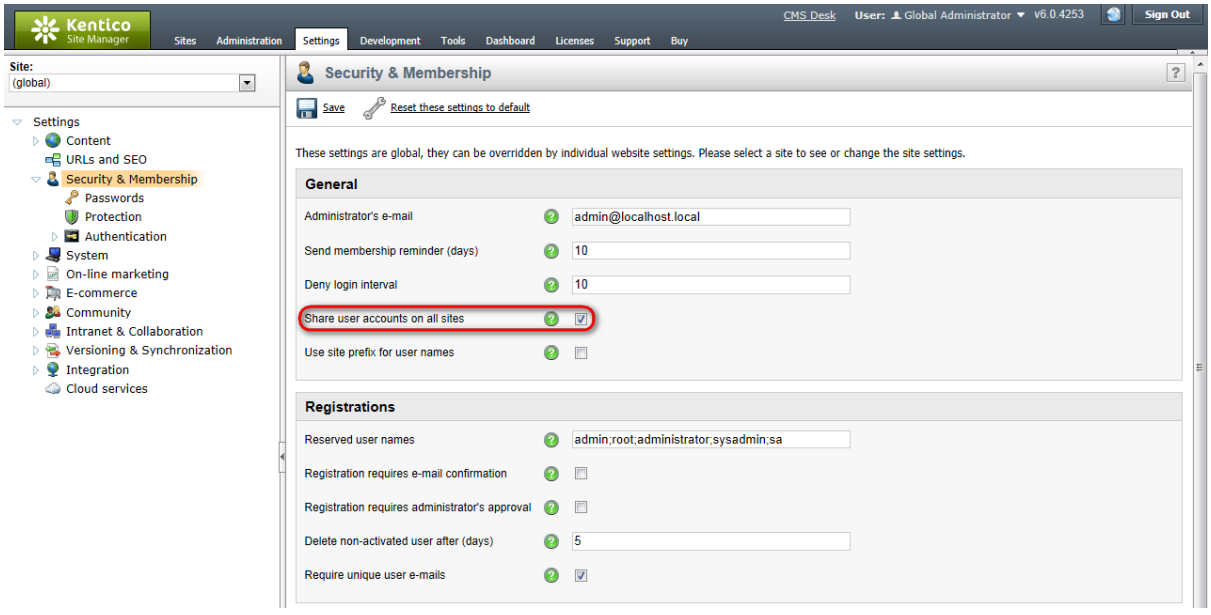
In addition to the special ones listed above, you can also use all other standard macro expressions in the templates. See the Development -> Macro expressions chapter of this guide for more information about macro expressions in Kentico CMS.

7.13.9.6 Shared user accounts

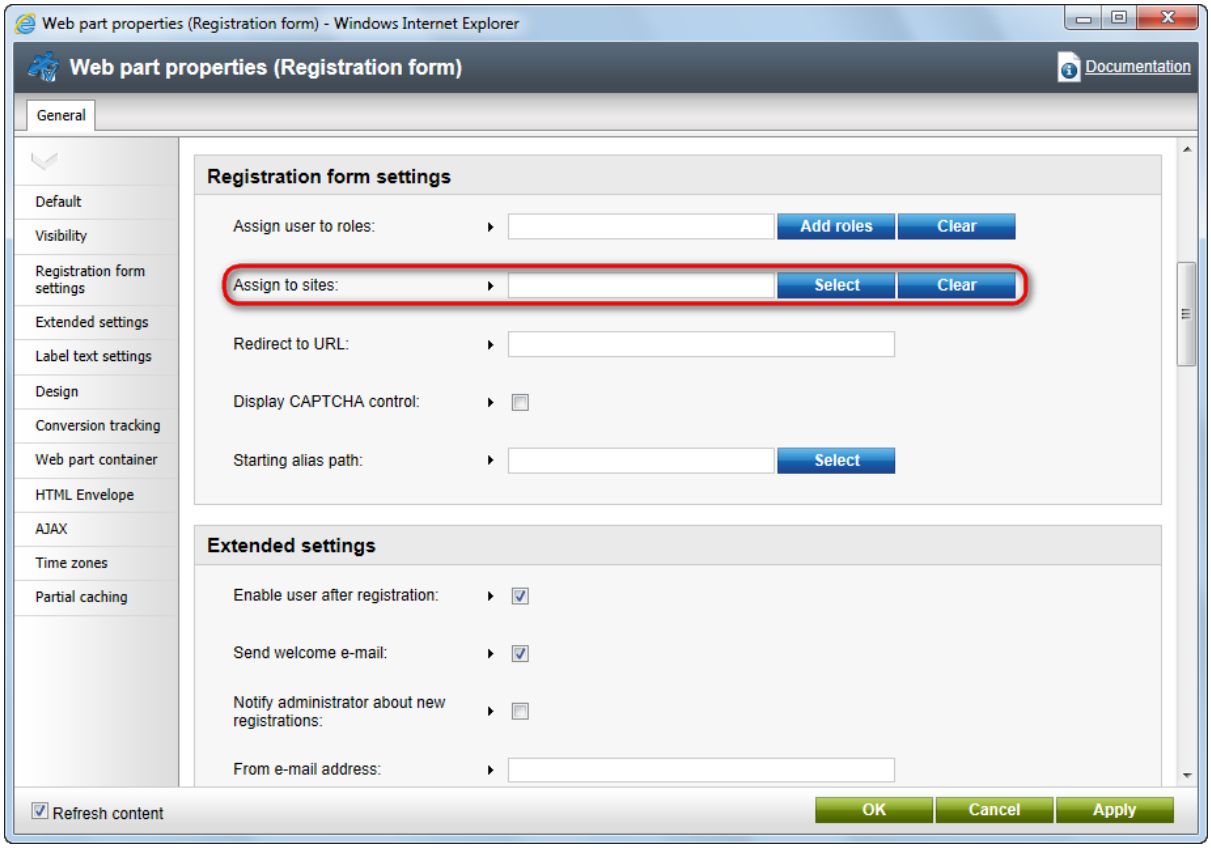
User accounts can be shared among all sites running on one Kentico CMS installation. This means that if a user creates an account on one site, they can automatically log-on to the other sites running on the same installation using the same credentials.

This behavior can be switched on or off in **Site Manager -> Settings -> Membership & Security**, using the **Share user accounts on all sites** check-box.

- If the check-box is enabled, user accounts created on one site will be shared among all the sites running on the installation.
- If the check-box is disabled, new accounts will be assigned only to the current site and not the others.



However, registration web parts have the **Assign to sites** property. Using this property, you may determine which sites the user accounts created via the web part will be assigned to.



User name prefixes and shared accounts

It is recommended to avoid using shared accounts together with the **Use site prefixes for user names** option that can also be enabled in **Site Manager -> Settings -> Security & Membership**.

Prefixes allow the creation of users with names that are not globally unique. If the **Share user accounts on all sites** setting is enabled, this may lead to problems with multiple identical user names on a single site.

7.13.10 Badges

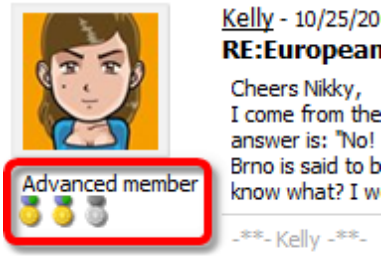
7.13.10.1 Badges

Users may be labeled with badges. These are images with a short title, expressing a user's activity level, importance or role in the context of the website. They can be displayed in forum posts, on user public profiles or in your own custom controls.

There are two types of badges:

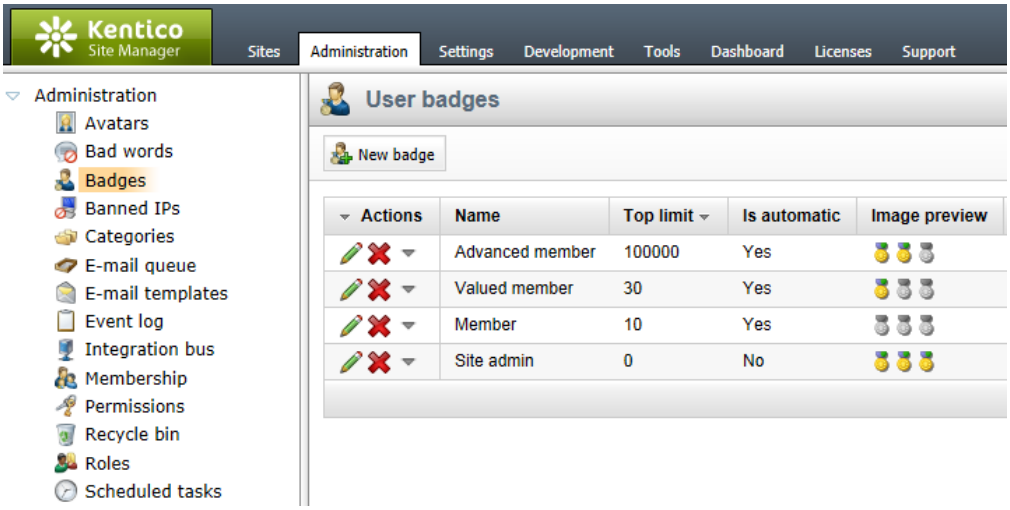
- **Automatic badges** - these are assigned to users based on their number of activity points. Every time a user gains activity points, for example by posting on the website's forums, the system automatically checks whether the level required for a new badge has been reached and updates it if this is the case.
- **Non-automatic badges** - assigned to users by site administrators. These are permanent unless manually changed, regardless of the number of activity points acquired by the given user.

In the screenshot below, you can see one of the pre-defined badges in a forum post.



7.13.10.2 Defining badges

Badges can be defined in **Site Manager -> Administration -> Badges**. On this page, you can see a list of currently defined badges. You can **Edit** (✎) or **Delete** (✖) badges in the list or define a new badge by clicking the **New badge** button at the top part of the page.



When creating a new badge, the following properties can be specified:

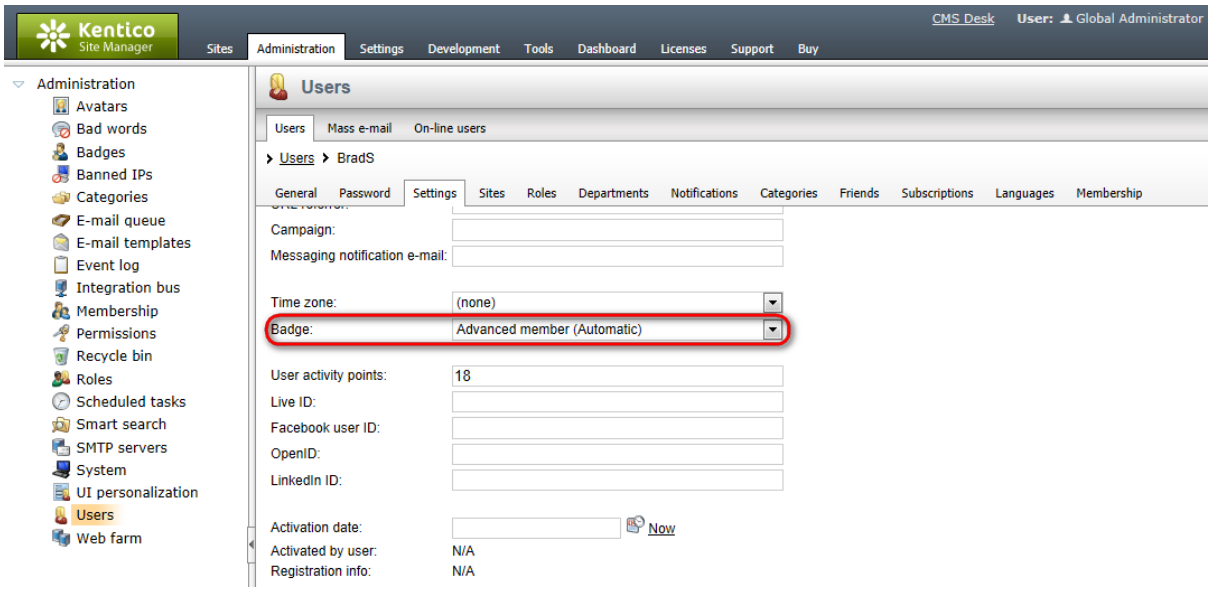
- **Display name** - name of the badge displayed on the website and in the administration interface.
- **Code name** - a unique name that serves as an identifier for the badge, for example in the API. You can leave the default *(automatic)* option to have the system generate an appropriate code name based on the display name.
- **Image URL** - sets the path of the image that will be displayed as a representation of the badge. This can be a URL, a relative path from the root of the web project (starting with "~/") or a path under the default theme folder (*~/App_Themes/Default/Images/*), e.g. */Objects/CMS_Badge/Default/advancedmember.gif*.

- **Is automatic** - if checked, the badge will be assigned to users automatically based on the number of gained activity points. Otherwise the badge will need to be assigned manually by site administrator and will remain assigned permanently, regardless of the number of activity points.
- **Top limit** - this property only affects automatic badges. The system assigns the badge to users who have less activity points than the limit. If a user fulfills this condition for multiple badges, the system always assigns the one with the lowest *Top limit* value.

7.13.10.3 Assigning badges to users

Site administrators can assign badges to users in **Site Manager / CMS Desk -> Administration -> Users -> Edit (🖉) user -> Settings**. It can be done via the **Badge** drop-down list.

This approach is typically used to assign non-automatic badges to users. However, automatic badges can also be assigned in this way.

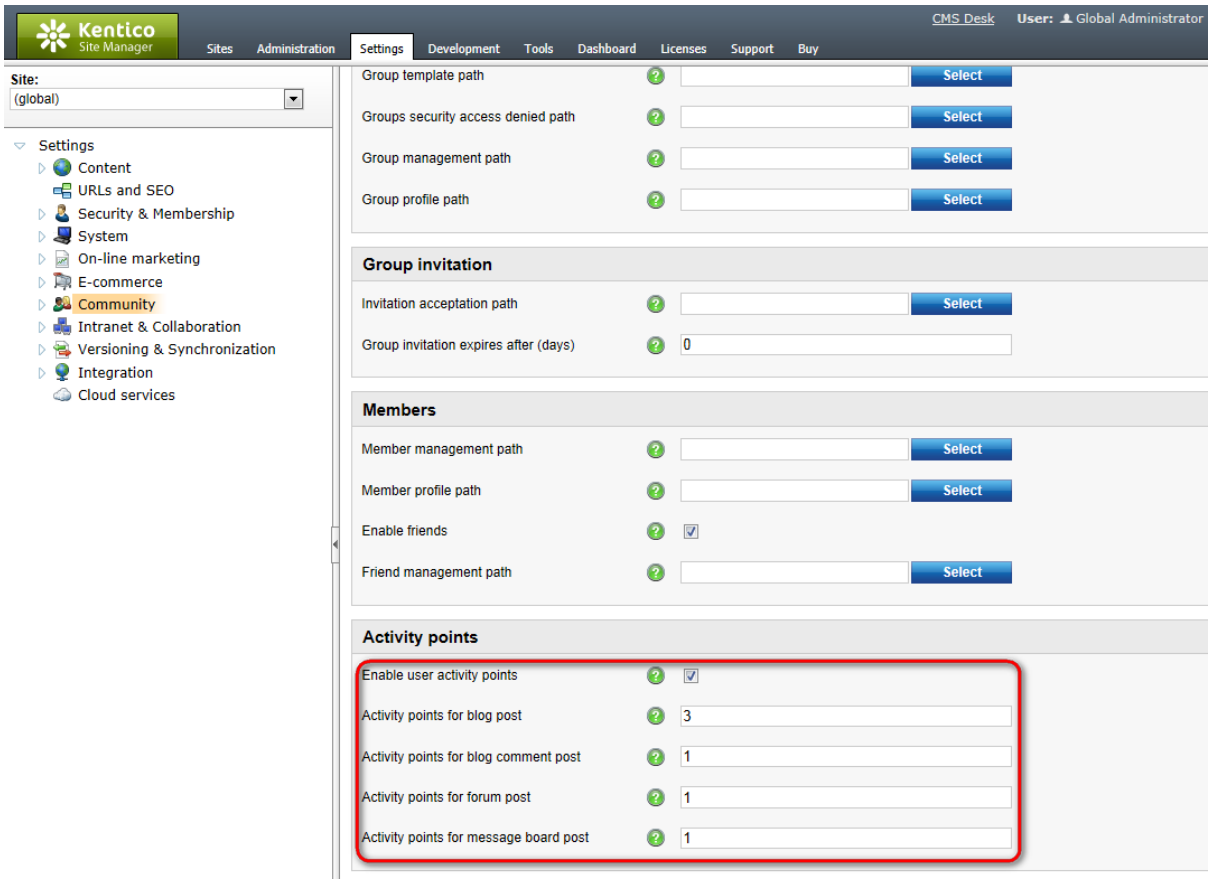


7.13.10.4 Activity points

Users can gain activity points for their activity on the site. Based on the number of gained activity points, automatic badges can be assigned to users. For this feature to be functional, you have to configure the following settings in **Site Manager -> Settings -> Community**:

- **Enable user activity points** - enables the logging of activity points for users.

- **Activity points for blog posts** - number of activity points that users receive for adding a blog post.
- **Activity points for blog comment post** - number of activity points that users receive for adding a blog post comment.
- **Activity points for forum post** - number of activity points that users receive for adding a forum post.
- **Activity points for message board post** - number of activity points that users receive for adding a message board post.



7.13.10.5 Available form controls

The following form controls can be used in your custom fields to display the badges of users:

- **Viewer - Badge image (ViewBadgeImage) -** used for displaying the image of a badge.
- **Viewer - Badge text (ViewBadgeText) -** used for displaying the *Display name* of a badge.

7.13.10.6 Badges internals and API

7.13.10.6.1 Overview

In this chapter, you will learn which database tables and API classes are used by Badges. You will also see the most common API examples.

Further API-related information and examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all methods from the module's classes, please refer to Kentico CMS API Reference.

7.13.10.6.2 Database tables

The following database table is used to store information about badges:

- **CMS_Badge** - contains records representing badges.

CMS_Badge

	Column Name	Data Type	Allow Nulls
🔑	BadgeID	int	☐
	BadgeName	nvarchar(100)	☐
	BadgeDisplayName	nvarchar(200)	☐
	BadgeImageURL	nvarchar(200)	☑
	BadgeIsAutomatic	bit	☐
	BadgeTopLimit	int	☑
	BadgeGUID	uniqueidentifier	☐
	BadgeLastModified	datetime	☐
			☐

7.13.10.6.3 API classes

If you are not familiar with database table data management through Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

The classes for managing badges can be found in the **CMS.SiteProvider** namespace.

CMS_Badge table API:

- **BadgeInfo** - represents one badge object.
- **BadgeInfoProvider** - provides management functionality for badges.

7.13.10.6.4 API examples

7.13.10.6.4.1 Overview

These topics show examples of how the API of badges can be used:

- Managing badges
- Managing user badges

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Administration\Badges\Default.aspx.cs***.

The badge API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.GlobalHelper;
using CMS.CMSHelper;
using CMS.SiteProvider;
```

7.13.10.6.4.2 Managing badges

The following example creates a badge.

```
private void CreateBadge()
{
    // Create new badge object
    BadgeInfo newBadge = new BadgeInfo();

    // Set the properties
    newBadge.BadgeDisplayName = "My new badge";
    newBadge.BadgeName = "MyNewBadge";
    newBadge.BadgeTopLimit = 50;
    newBadge.BadgeImageURL = "~/App_Themes/Default/Images/Objects/CMS_Badge/
Default/siteadmin.gif";
    newBadge.BadgeIsAutomatic = true;

    // Save the badge
    BadgeInfoProvider.SetBadgeInfo(newBadge);
}
```

The following example gets and updates a badge.

```
private bool GetAndUpdateBadge()
{
    // Get the badge
    BadgeInfo updateBadge = BadgeInfoProvider.GetBadgeInfo("MyNewBadge");
    if (updateBadge != null)
    {
        // Update the properties
        updateBadge.BadgeDisplayName = updateBadge.BadgeDisplayName.ToLower();

        // Save the changes
        BadgeInfoProvider.SetBadgeInfo(updateBadge);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates badges.

```
private bool GetAndBulkUpdateBadges()
{
    // Prepare the parameters
    string where = "BadgeName LIKE N'MyNewBadge%'";

    // Get the data
    DataSet badges = BadgeInfoProvider.GetBadges(where, null);
    if (!DataHelper.DataSourceIsEmpty(badges))
    {
        // Loop through the individual items
        foreach (DataRow badgeDr in badges.Tables[0].Rows)
        {
            // Create object from DataRow
            BadgeInfo modifyBadge = new BadgeInfo(badgeDr);

            // Update the properties
            modifyBadge.BadgeDisplayName = modifyBadge.BadgeDisplayName.ToUpper();

            // Save the changes
            BadgeInfoProvider.SetBadgeInfo(modifyBadge);
        }

        return true;
    }

    return false;
}
```

The following example deletes a badge.

```
private bool DeleteBadge()
{
    // Get the badge
    BadgeInfo deleteBadge = BadgeInfoProvider.GetBadgeInfo("MyNewBadge");

    // Delete the badge
    BadgeInfoProvider.DeleteBadgeInfo(deleteBadge);

    return (deleteBadge != null);
}
```

7.13.10.6.4.3 Managing user badges

The following example assigns a badge to a specific user.

```
private bool AddBadgeToUser()
{
    // Get user object
    UserInfo user = UserInfoProvider.GetUserInfo(CMSContext.CurrentUser.UserName);
```

```
    // Get badge object
    BadgeInfo myBadge = BadgeInfoProvider.GetBadgeInfo("MyNewBadge");

    if ((user != null) && (myBadge != null))
    {
        // Add badge to user settings
        user.UserSettings.UserBadgeID = myBadge.BadgeID;

        // Update user object in database
        UserInfoProvider.SetUserInfo(user);

        return true;
    }

    return false;
}
```

The following example updates the activity points of a specific user.

```
private bool UpdateActivityPoints()
{
    // Get user
    UserInfo user = UserInfoProvider.GetUserInfo(CMSContext.CurrentUser.UserName);

    // If user exists
    if (user != null)
    {
        // Update activity points of the user
        BadgeInfoProvider.UpdateActivityPointsToUser
(ActivityPointsEnum.BlogCommentPost, user.UserID, CMSContext.CurrentSiteName,
true);

        return true;
    }

    return false;
}
```

The following example removes a user's badge.

```
private bool RemoveBadgeFromUser()
{
    // Get user
    UserInfo user = UserInfoProvider.GetUserInfo(CMSContext.CurrentUser.UserName);

    // If user exists
    if (user != null)
    {
        user.UserSettings.UserBadgeID = 0;

        // Save updates
```

```
            UserInfoProvider.SetUserInfo(user);

            return true;
        }

        return false;
    }
```

7.13.11 Custom field visibility

7.13.11.1 How it works

The visibility control functionality is designed to enable registered users to decide which fields of their public profile will be visible to other users. You can find an example of how this works on the sample **Community site**.

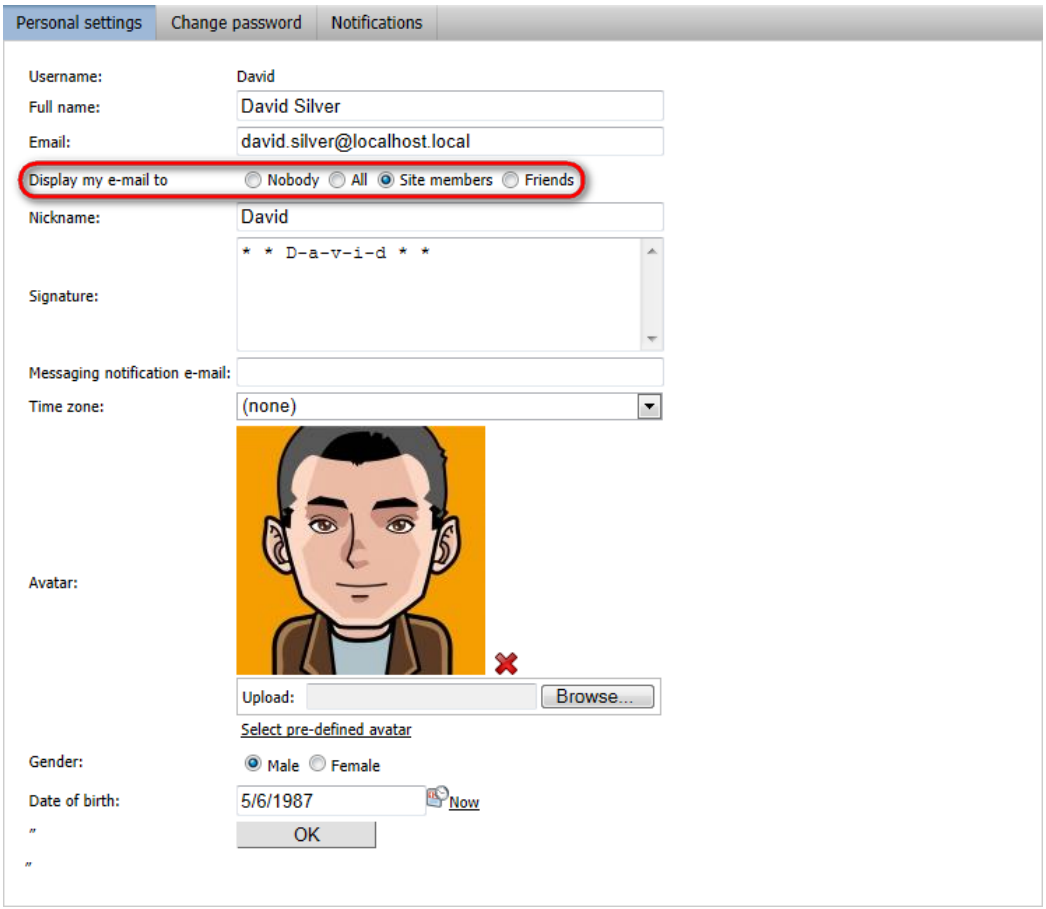
1. Run the **Community site** and sign out of the administration interface. Log on to the site using the **Sign in** form on the right. Enter *David* with a blank password and click **Log on**.

2. Once logged in, the **Shortcuts** menu is displayed instead of the Sign in form. Click the **Edit my profile** link to open the user's profile editing page.

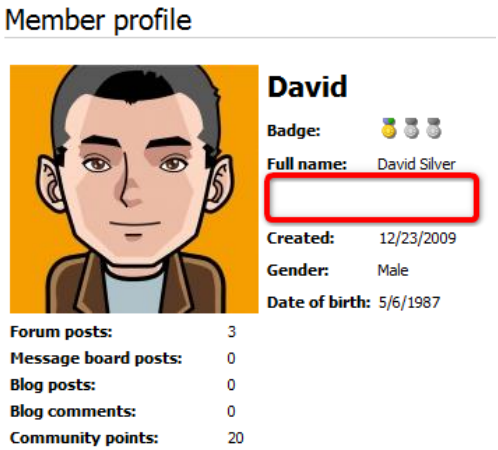
3. There is a set of radio buttons under the **E-mail** field. This control allows users to configure the visibility of their e-mail address in the profile. The following four options are available:

- **Nobody** - the field cannot be seen by anyone (only by the profile's owner).
- **All** - the field is visible for all visitors.
- **Site members** - the field is only displayed to authenticated users.
- **Friends** - the field is only displayed to the user's friends.

Choose **Site members** and click **OK**.



4. You have just configured the user's profile so that only authenticated users can see the e-mail address. Let's verify that it really works. Sign out and visit David's profile as an unauthenticated site visitor. From the site's main menu, select **Members** and click David's icon in the list below. You should see his profile, but the e-mail address is not present.



5. Now sign in the same way as you did in step one, but use *Mia* with a blank password instead. This signs you in as an authenticated member of the site, which fulfills the visibility requirements of the E-mail field. So once signed in, view David's profile again. The e-mail address is visible.

7.13.11.2 Enabling visibility controls

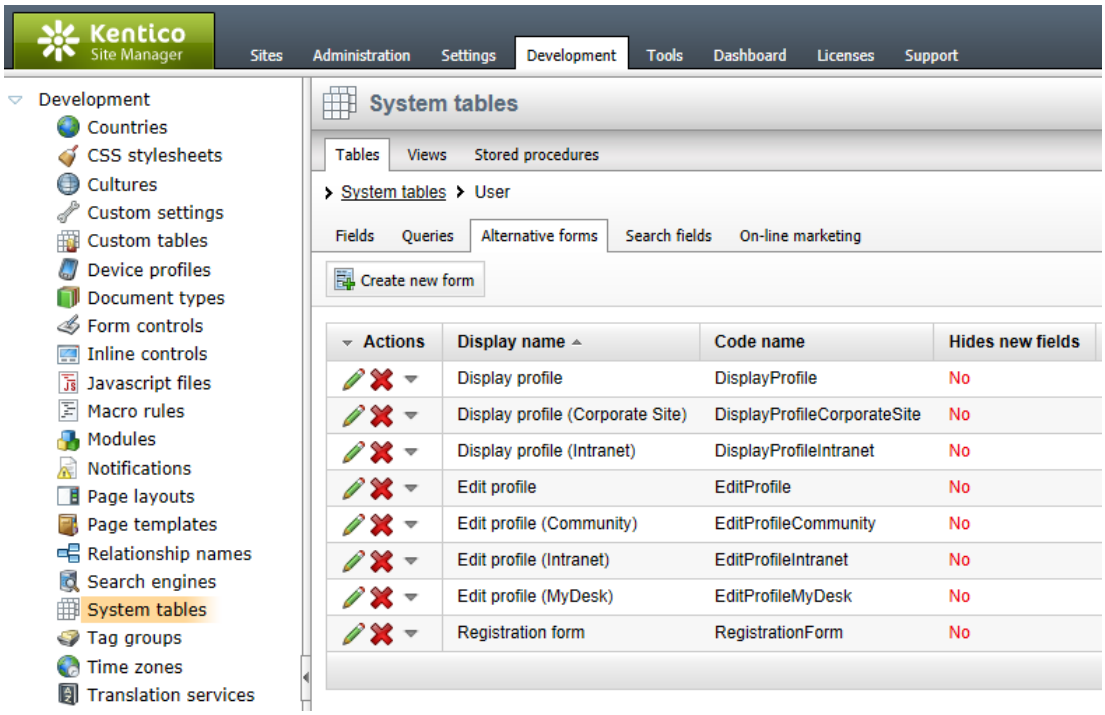
The visibility selection drop-down list can be added to any field of the user's profile, not just the e-mail field shown in the example in the How it works topic. This can be changed in **Site Manager -> Development -> System tables**. Choose to **Edit** (✎) the **User (CMS_User)** system table and switch to the **Alternative forms** tab.

If you are not familiar with the **Alternative forms** concept, please read the Module Alternative forms chapter first.

The **User** system table represents the database table that stores information about registered users. Each of the four alternative forms that you see in the list is used in a specific situation when the system accesses the table:

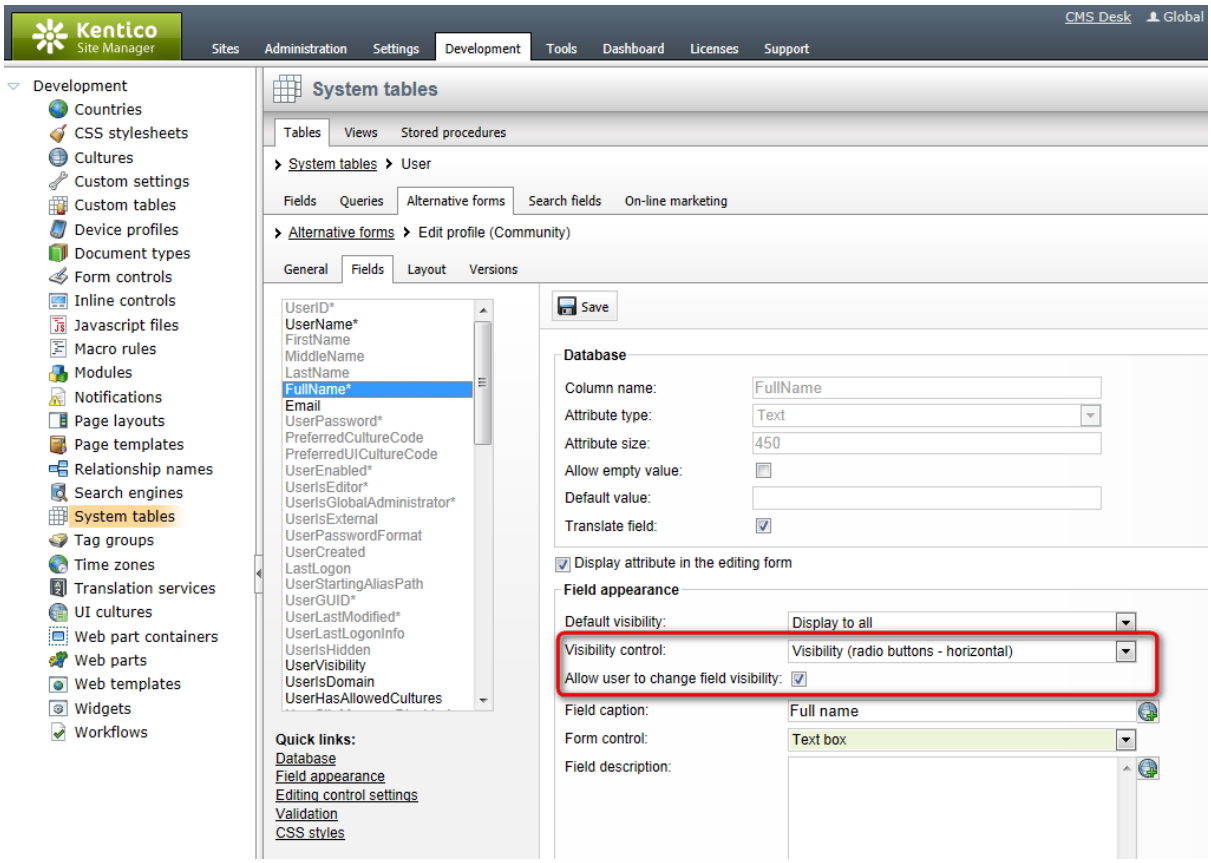
- **Registration form** - when registering a new user using the **Custom registration form** web part.
- **Display profile** - when displaying a user's public profile using the **User public profile** web part.
- **Edit profile** - when editing a user's profile using the **My account** web part.
- **Edit profile (MyDesk)** - when editing the user profile in **CMS Desk -> My Desk -> Account -> Details**.

The visibility field can be set in each of these forms.



Let's assume that we want the **Full name** field to be optionally hidden in the public profiles of users, based on the configuration made in the profile editing section of each specific user.

1. **Edit** (🖉) the **Edit profile (Community)** alternative form, which is the form that is used by the **My profile** web part on the Community site. Switch to the **Fields** tab and select **FullName** from the list on the left. Select **Visibility (radio buttons - horizontal)** from the **Visibility control** drop-down. Check the **Allow user to change field visibility** checkbox and finally click 💾 **Save**.

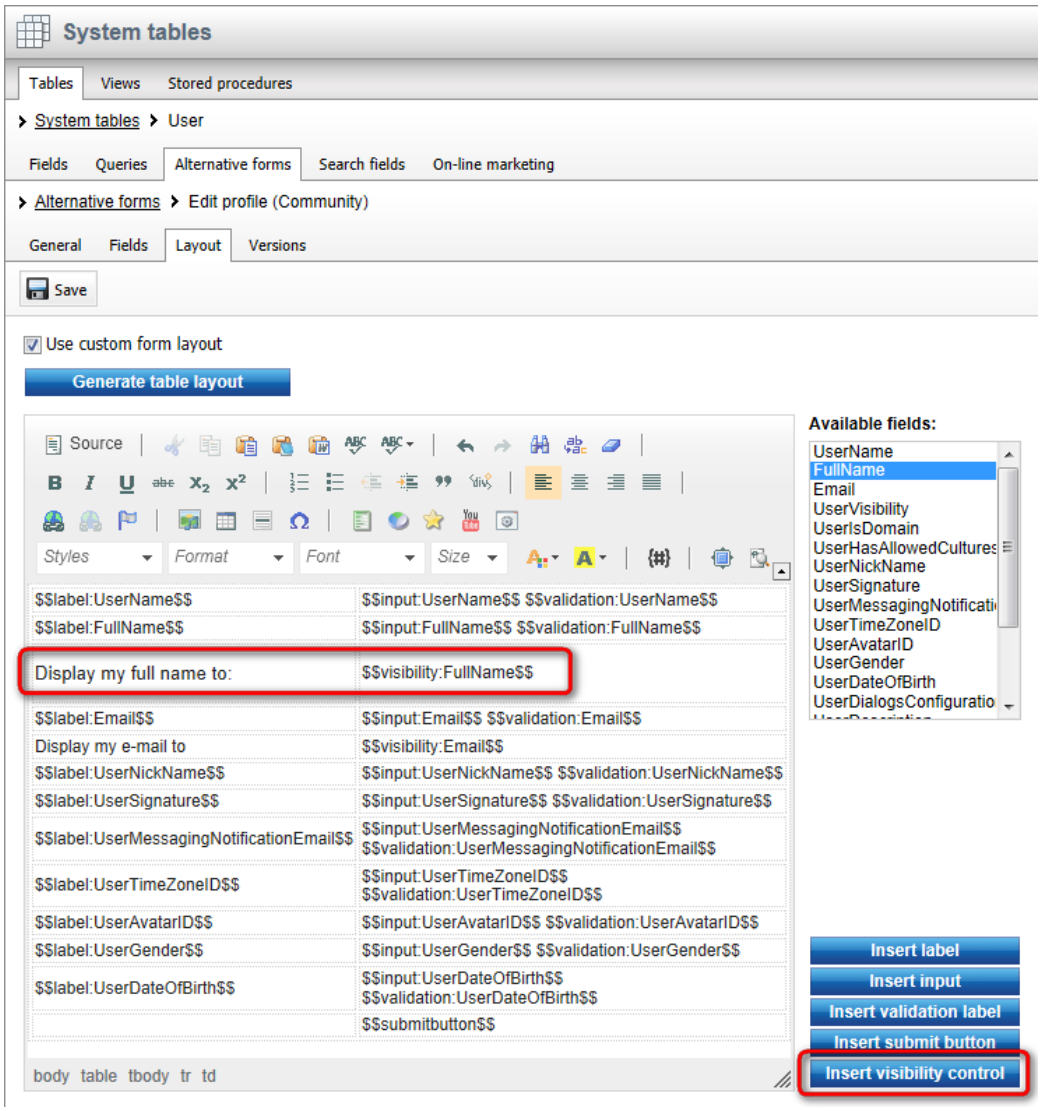


2. Switch to the **Layout** tab. We will need to create another line in the table and add the visibility control into it.

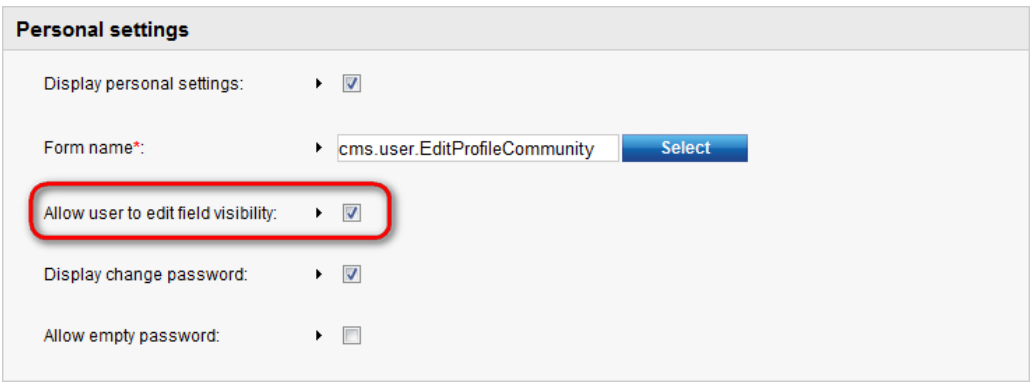
Place the cursor into the FullName line (where the $$label:FullName$$ value is). Right click and choose **Row -> Insert Row After** from the context menu. Into the first column of the new row, enter: *Display my full name to:*

Now we will add the visibility control itself. Place the cursor into the second column, select **FullName** from the **Available fields** list and click the **Insert visibility control** button. The result should look as in the screenshot below.

When you are finished, click **Save**.

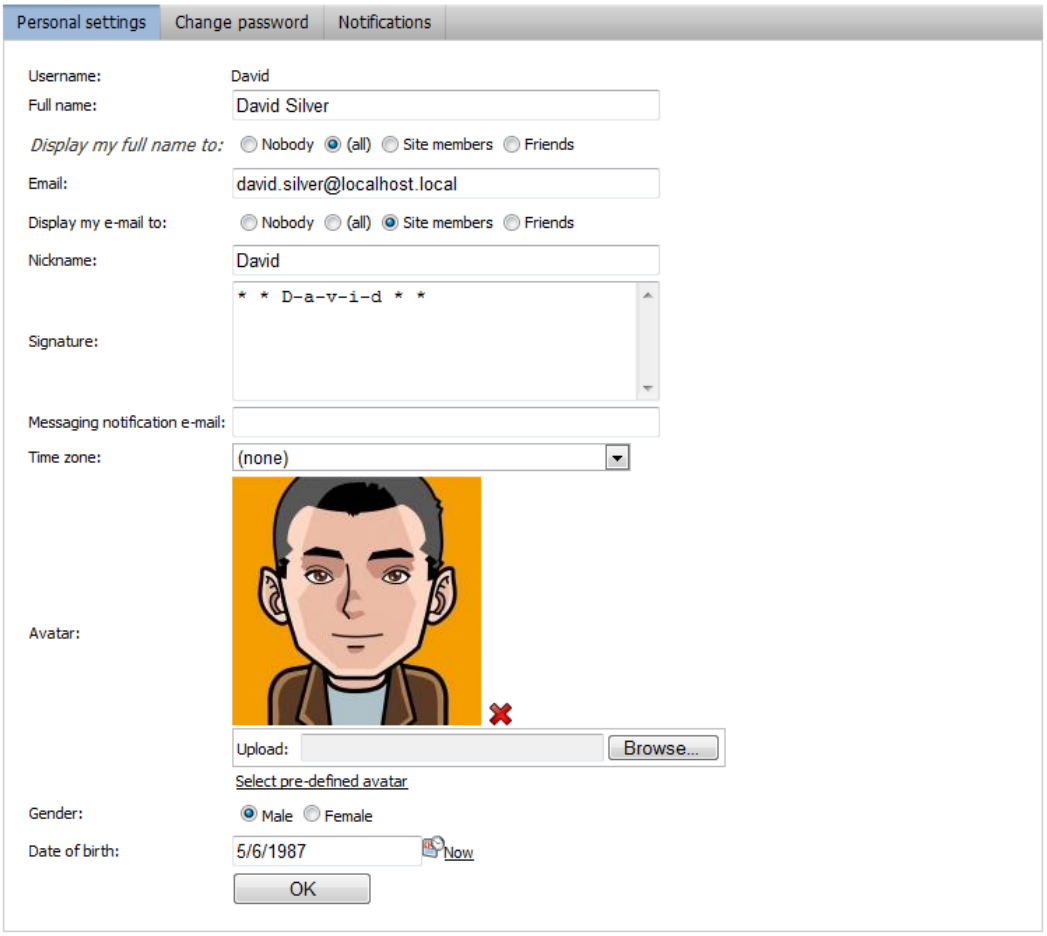


3. Switch to CMS Desk and select the **Members -> Management** document in the content tree. Open the **Design** tab and **Configure** ( ) the **MyAccount** web part. Make sure that the **Allow user to edit field visibility** property is checked and click **OK**.



4. Now sign out of the administration interface and sign in to the website as *David* with a blank

password. Click the **Edit my profile** link in the right **Shortcuts** menu. You should see a set of radio buttons below the *Full name* field, which allow users to determine the visibility of their full name.



7.13.11.3 Use in custom form layouts

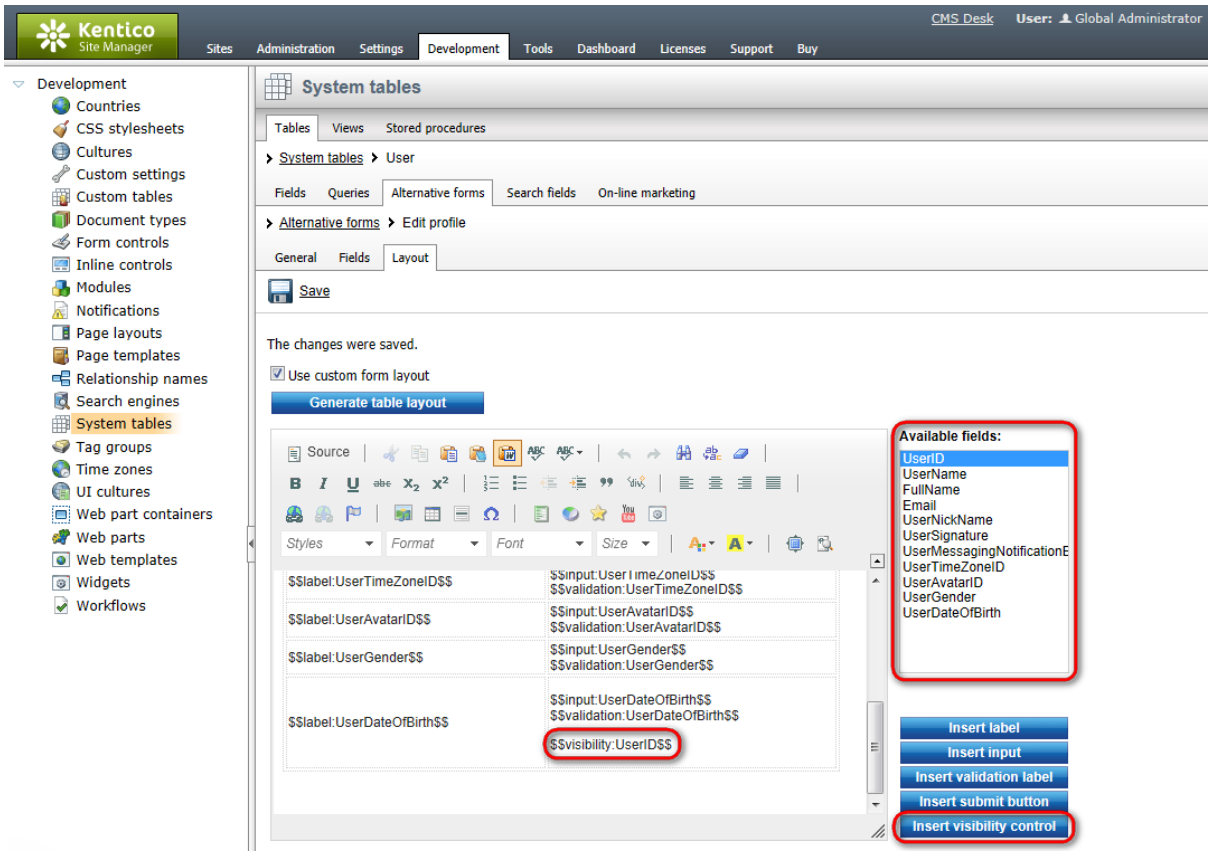
If you want to define a custom form layout and use the visibility drop-down list for some field, you have to do the following two things:

- the **Allow user to change visibility** check-box must be enabled for each field that you want to use the drop-down list for. If it is not enabled, the drop-down list will not be functional.
- add the drop-down list manually to the form, using the **Insert visibility control** button:

1. Go to **Site Manager -> Development -> System tables**. Choose to **Edit** (🖉) the **User** system table and switch to the **Alternative forms** tab. Choose to **Edit** (🖉) the **Edit profile** alternative form and switch to the **Layout** tab. Check the **Use custom form layout** check-box.

2. Click the **Generate form layout** button. A default form layout will be generated and you can make modifications to it.

3. Enter the visibility controls by placing the cursor to the desired location, selecting the appropriate field and clicking the **Insert visibility control** button. Click **Save** when you are finished.



7.13.11.4 Configuring the web parts

If you want to enable visibility controls in these web parts, you have to add the controls to the appropriate alternative forms and set the following properties of the web parts:

User public profile

- **Form name** - specifies the full name of the desired alternative form (*cms.user.DisplayProfile* by default).
- **Apply visibility settings** - check this to enable the visibility controls.
- **Use visibility settings from form** - can be used to select from which form visibility settings should be loaded if the Apply visibility settings property is enabled. If empty, the form specified in the Form name property will be used.

Custom registration form

- **Alternative form** - specify the full name of the desired alternative form (*cms.user.RegistrationForm* by default).

My account

- **Form name** - specify the full name of the desired alternative form (*cms.user.EditProfile* by default).
- **Allow user to edit field visibility** - check to enable the visibility controls.

7.13.12 Authentication

7.13.12.1 Authentication overview

The system supports both forms and Windows authentication. The **forms authentication** stores user names and passwords in the database and requires users to log on. The **Windows authentication** gets user identity from the network credentials and automatically creates a corresponding user in the database, including the user's roles (if they exist in the CMS database).

Accessing current user data in code

When the user is authenticated, a CMS.CMSHelper.CurrentUserInfo object representing the current user is stored in the session variable **CMSCurrentUser** and is accessible through the **CMSHelper.CMSContext.CurrentUser** property. All operations after authentication then use the user profile and user roles assigned to this object.

[C#]

```
// gets the user name of the current user
string userName = CMS.CMSHelper.CMSContext.CurrentUser.UserName;
```

Configuring forms authentication

Forms authentication is configured as the default option. It uses standard ASP.NET forms authentication and its settings, which you can find in your application's **web.config** file:

```
<system.web>

...

  <authentication mode="Forms">
    <forms loginUrl="CMSPages/logon.aspx" defaultUrl="Default.aspx"
name=".ASPXFORMSAUTH" timeout="60000" slidingExpiration="true" />
  </authentication>

...

<system.web>
```

If you're running multiple web projects in virtual directories, and the projects have the same machine key defined, users logging in to one of the websites will be automatically logged in to sites running on other projects. To prevent that, add the **path** parameter to the above code in each project, as in the following example:

```
<authentication mode="Forms">
  <forms loginUrl="CMSPages/logon.aspx" defaultUrl="Default.aspx"
name=".ASPXFORMSAUTH" timeout="60000" slidingExpiration="true" path="KenticoCMS" /
>
```

```
</authentication>
```

Additional configuration options related to user passwords may also be defined in the system, as described in the Password management chapter.

Membership provider and ASP.NET 2.0 Membership support

Kentico CMS contains an ASP.NET 2.0 Membership provider for its user database. This means you can use ASP.NET 2.0 Membership API and controls, such as Login control. However, Kentico CMS uses its own user information database instead of the ASP.NET 2.0 Membership tables. Please see Membership internals and API -> Database tables for detailed information about the membership database structure.

Configuring Windows authentication

Please see the Windows authentication (Active Directory) sub-chapter to learn more.

Configuring custom authentication

If you want to retrieve user and role information from an external source (such as a custom database), you need to configure the system as described in the Integrating authentication with external systems topic.

7.13.12.2 Session management

This topic provides a brief overview of how Kentico CMS handles sessions and a provides advice on how to mitigate session-based security threats.

When a public user requests a page for the first time, the system creates a session for her. The session stores information about the user, such as the current culture, or the contents of the user's shopping cart.

When the user logs in to the system, the system maintains her session so that the user can keep her session data. However, this can pose a threat, since an attacker could create a session and then trick a user into using the session. The user will then share the same session with the attacker. This way the attacker can acquire sensitive information from the session.

To prevent session based attacks, you can insert the following settings key into the **appSettings** section of your **web.config** file:

```
<add key="CMSRenewSessionAuthChange" value="true" />
```

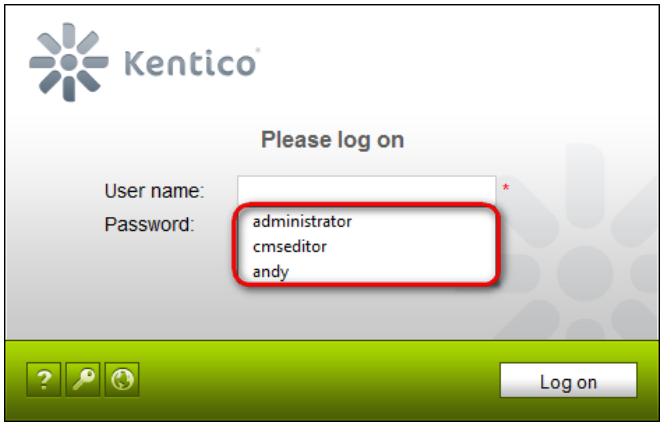
The key will cause the system to renew the session (abandon the old one and create a new one) every time a user logs in or out.

> **Important!**

If you enable this setting, users will not be able to preserve their session data after logging in or out.

7.13.12.3 Automatic user name completion

Most web browsers support storing user names that you use to access secured areas of websites. This feature, called **Autocomplete**, allows you to conveniently log into frequently used services without the need to type your user name every single time. However, Autocomplete can pose a security threat when used, e.g., on a public computer.



Kentico CMS allows you to set whether you want browsers to offer this feature to users logging in to your website. By default, it is turned on. You can change this in **Settings -> Security & Membership -> Protection -> Enable Autocomplete**. This setting influences the following log-on dialogs:

- Logon page to CMS Desk and Site manager
- Logon web parts
- Shopping cart web part

7.13.12.4 Windows authentication (Active Directory)

7.13.12.4.1 Configuring Windows authentication (Active Directory)

Kentico CMS supports Windows integrated authentication. It means that when a user signs in to a Windows domain, Kentico CMS automatically recognizes their identity without requiring a user name and password.

Moreover, Kentico CMS is able to automatically import the authenticated users from domain (Active Directory) into the user database, including their roles.

Configuration

1. Before you configure your project for *Windows authentication*, you need to create a user account that will be the same as your current domain name and assign this user account with administrator permissions. This will allow you to access all features as an administrator once you sign in using *Windows authentication*.

2. Sign in as an administrator to **Site Manager** and go to **Administration -> Users**. Create a new user with the following values:

- **User name:** your domain user name in format *domain-username*, example: *office-johns*
- **Full name:** your full name

Click **OK**.

3. On the **General** tab, set the following values:

- **Is global administrator:** yes
- **Is external user:** yes
- **Is domain user:** yes

Click **OK**.

4. Now you can switch the application to the *Windows authentication* mode. Edit the *web.config* file of the web project and change the following line:

```
<authentication mode="Forms">
```

to:

```
<authentication mode="Windows">
```

5. (Optional) When using *Windows authentication*, you may also want to have the following settings in your *web.config* file so that the windows authentication is required for access to the live site. By default, this code is already present in a commented block in your *web.config*, so you can just uncomment it to achieve the result.

```
<location path="">
  <system.web>
    <authorization>
      <deny users="?"/>
    </authorization>
  </system.web>
</location>
```

6. Save the modified web.config file. Close all browsers with Kentico CMS and open the website in a new browser. Try to go to **<web project>\cmssitemanager** to make sure you are recognized as a global administrator.

With this configuration, when an authenticated user comes to the site, their user account is created in Kentico CMS database automatically and their domain groups are imported as roles into Kentico CMS database. It means that the users and roles are not imported on a regular basis, but they are imported when the user comes to Kentico CMS website.

If you are experiencing the 401 error on Windows 7 or Windows Server 2008, learn the solution to the problem here.

Sign out button missing with Windows authentication

When Windows authentication is enabled, the **Sign out** button in the top right corner of **CMS Desk** or **Site Manager** is not displayed. The same applies to the live site, where the sign out link is not displayed in all web parts that can be used to sign out.

Forbidden characters replacement on Active Directory import

When importing users and roles, forbidden characters in their names are replaced by the character defined in **Site Manager -> Settings -> URLs and SEO -> Forbidden characters replacement**.

Dash "-" is the default value and therefore it is used in this example (*domain-username* instead of *domain\username*). If you are using a different character, please change the entered user name accordingly.

You can override this setting by using the following keys in the *AppSettings* section of your *web.config* file. In both cases, the value must be exactly one character which will be used as the replacement character:

- *<add key="CMSForbiddenUserNameCharactersReplacement" value="-" />*
- *<add key="CMSForbiddenRoleNameCharactersReplacement" value="-" />*

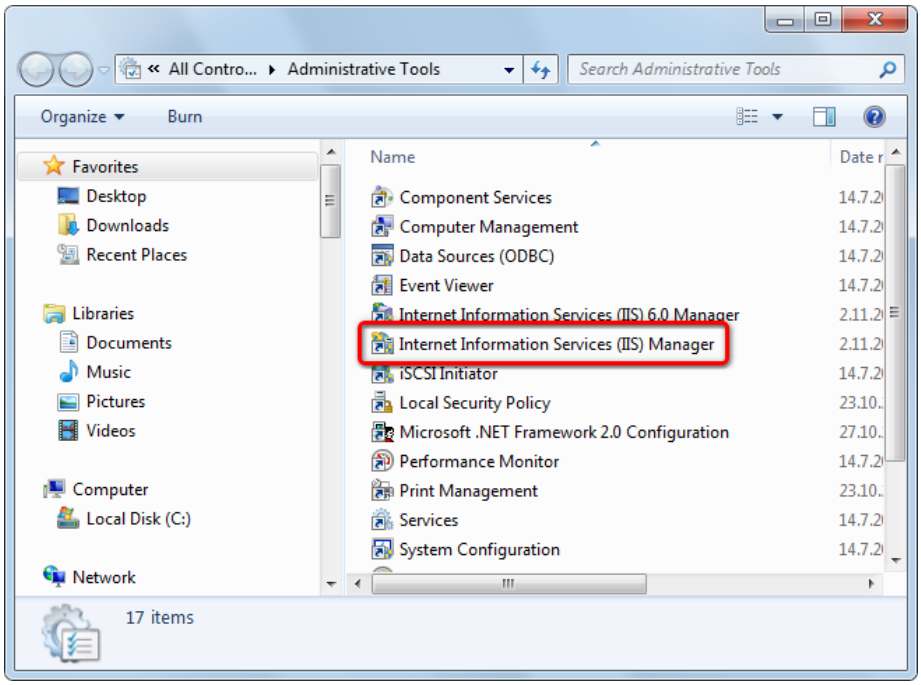
If you want to achieve the same functionality as in **older versions** of Kentico CMS (**office\username**), forbidden characters replacement can be turned off completely using the following two keys. This may cause problems when using wildcard URLs with user names in the wildcard part and is therefore not recommended.

- *<add key="CMSEnsureSafeUserNames" value="false" />*
- *<add key="CMSEnsureSafeRoleNames" value="false" />*

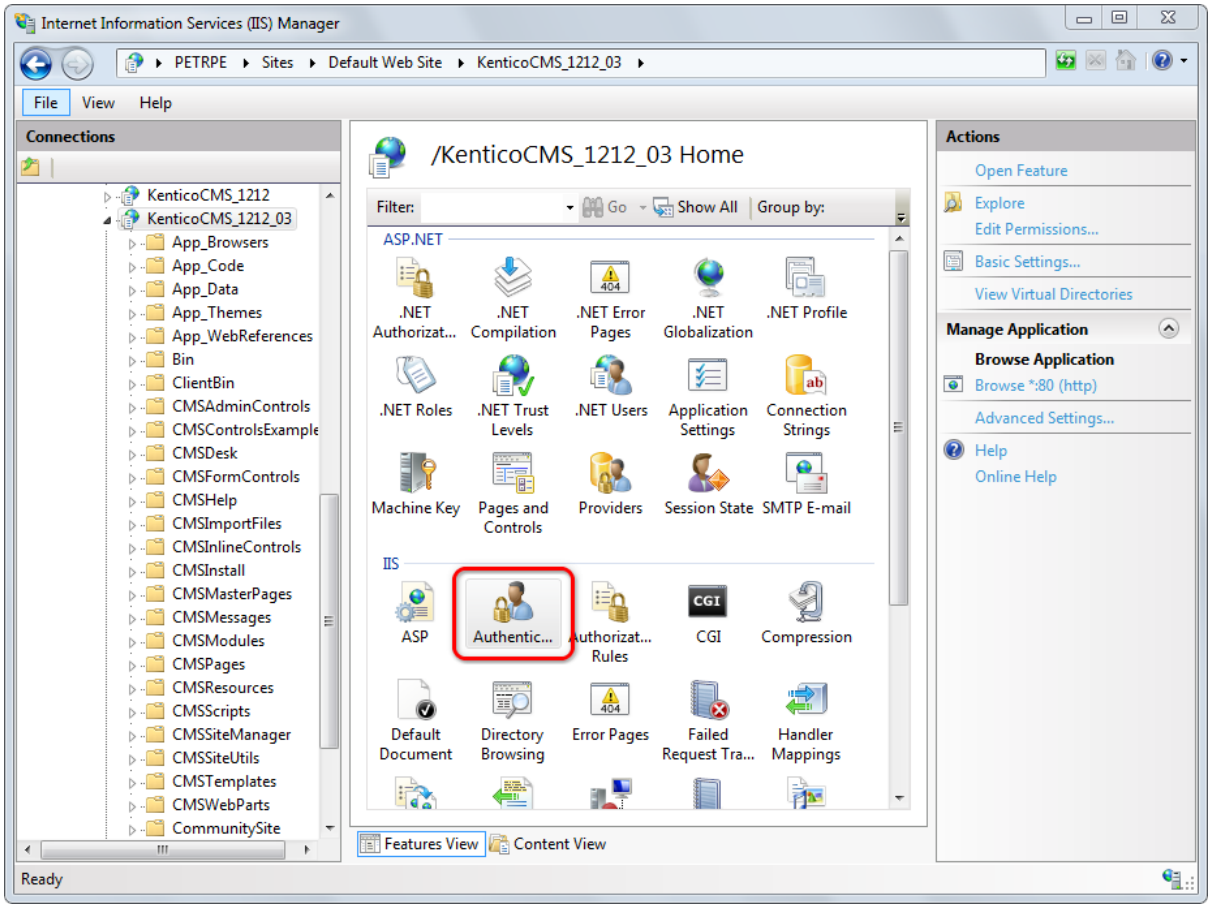
7.13.12.4.2 Windows authentification on Windows 7/2008 R2/Vista (IIS7 or higher)

If you are experiencing the **401 error** with Windows authentication on **Windows 7**, **Windows Server 2008 R2** or **Windows Vista**, you have to **set up your IIS** the following way:

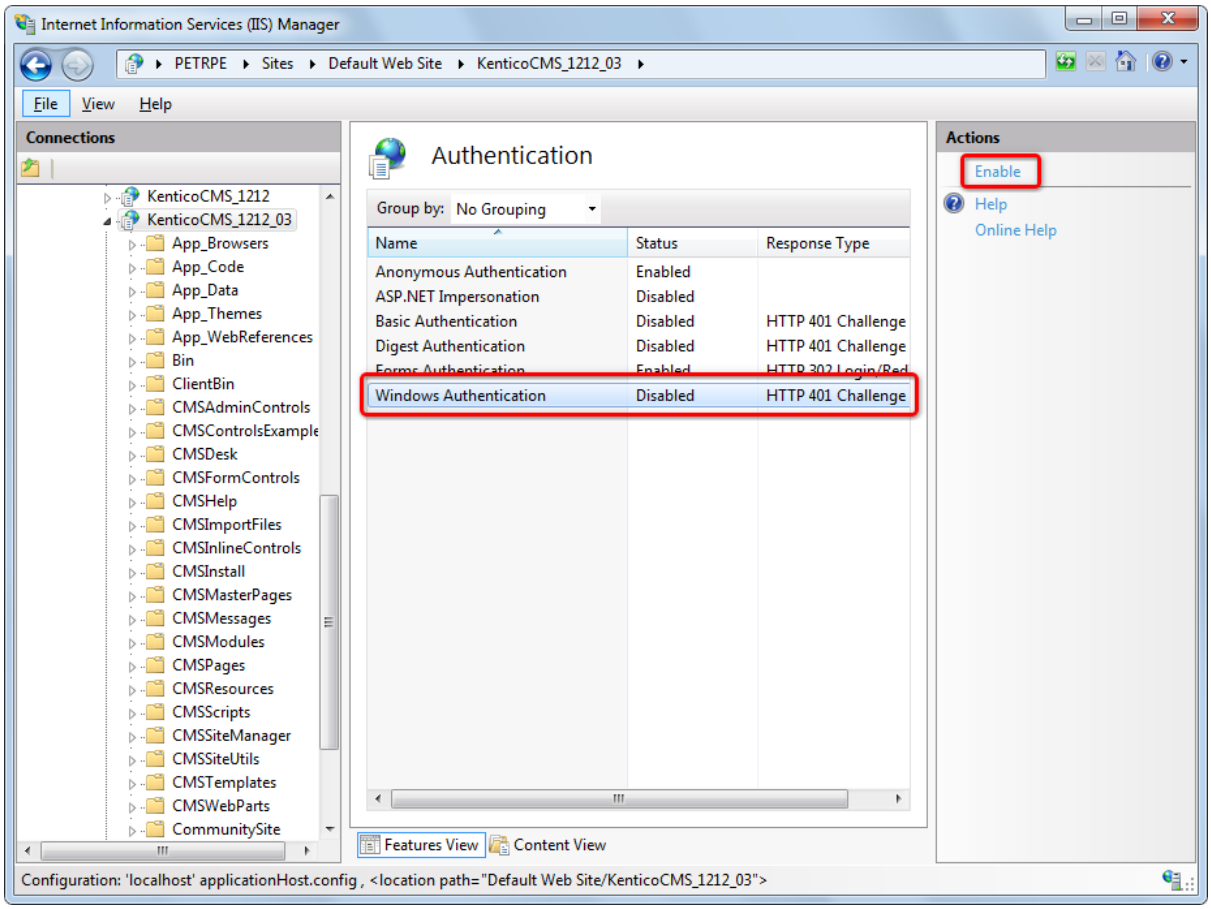
1. Go to **Start** -> **Control Panel** -> **Administrative Tools** and start the **Internet Information Services (IIS) Manager**.



2. Locate and select your site in IIS tree and click on the **Authentication** icon.



3. Enable **Windows Authentication** by clicking on the **Enable** link in the **Actions** menu.



Windows Authentication missing in the list

The default installation of Windows 7, Windows 2008 R8 and Windows Vista does not contain Windows Authentication by default. If it is missing in the list is step 3 above, you need to install it.

To do it, go to **Control Panel -> Programs and Features -> Turn windows features on or off**. Scroll down to *IIS*, expand all of the nodes to find the *Security* node and under it, check the Windows Authentication check-box. Click **OK** to save the settings.

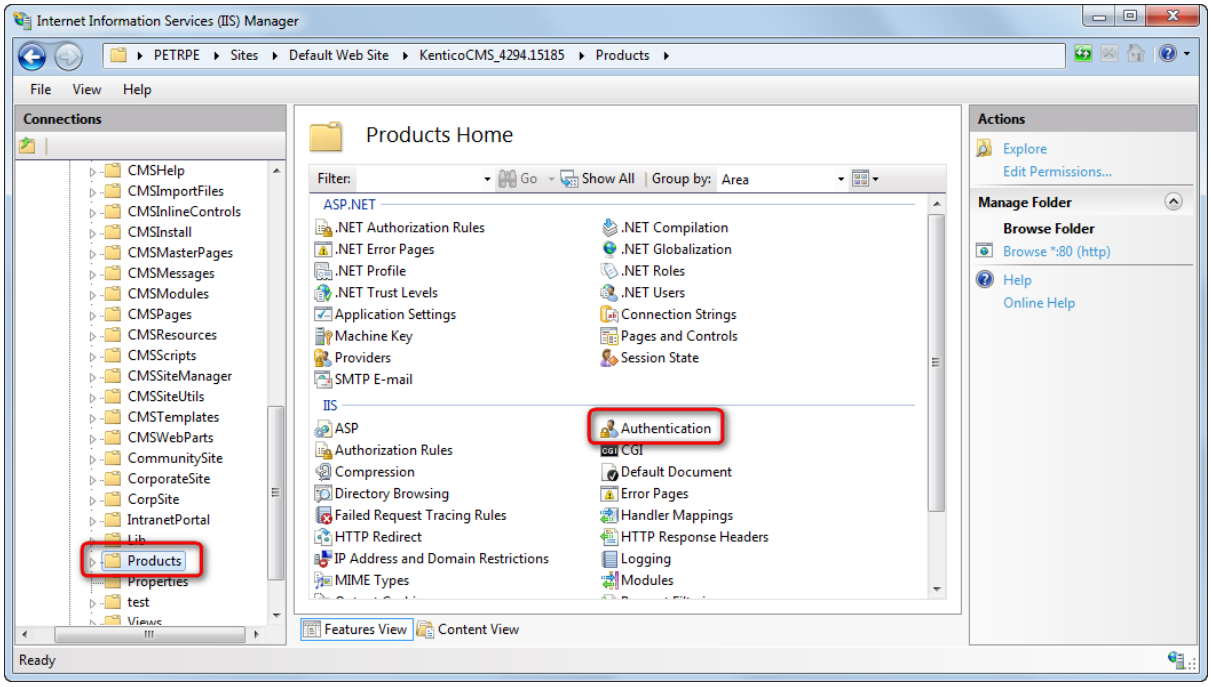
7.13.12.4.3 Securing a web site section using Windows authentication

It is also possible to secure only a certain section of your website using Windows authentication. In the following example, you will learn how to set the **Products** section of the sample **Corporate site** to be secured by the Windows authentication:

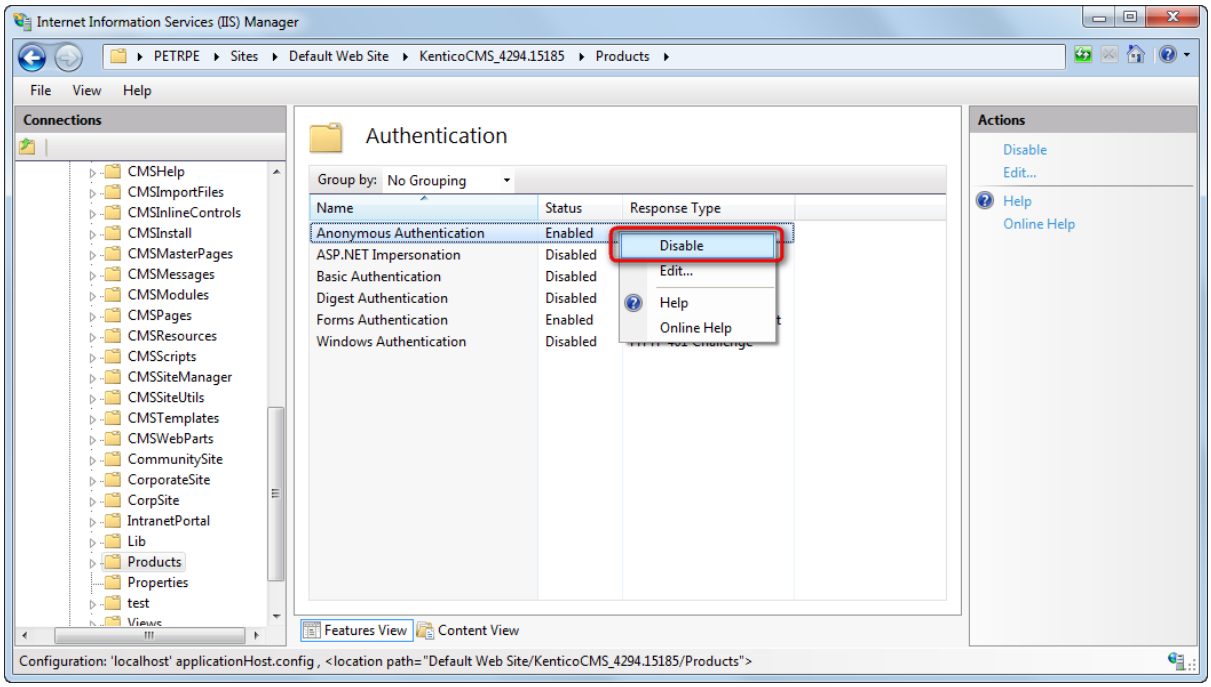
1. Locate your web project on the disk (typically *c:\Inetpub\wwwroot\<web project>*). Create a new directory in your web project's folder and give it the same name as the filename in the document's URL.

In this case, the filename is *Products.aspx*, so we will create a folder named **Products**.

2. Open the IIS and locate the folder in the tree. Select it and open the **Authentication** configuration on the right.



3. Right-click **Anonymous Authentication** and choose to **Disable** it.



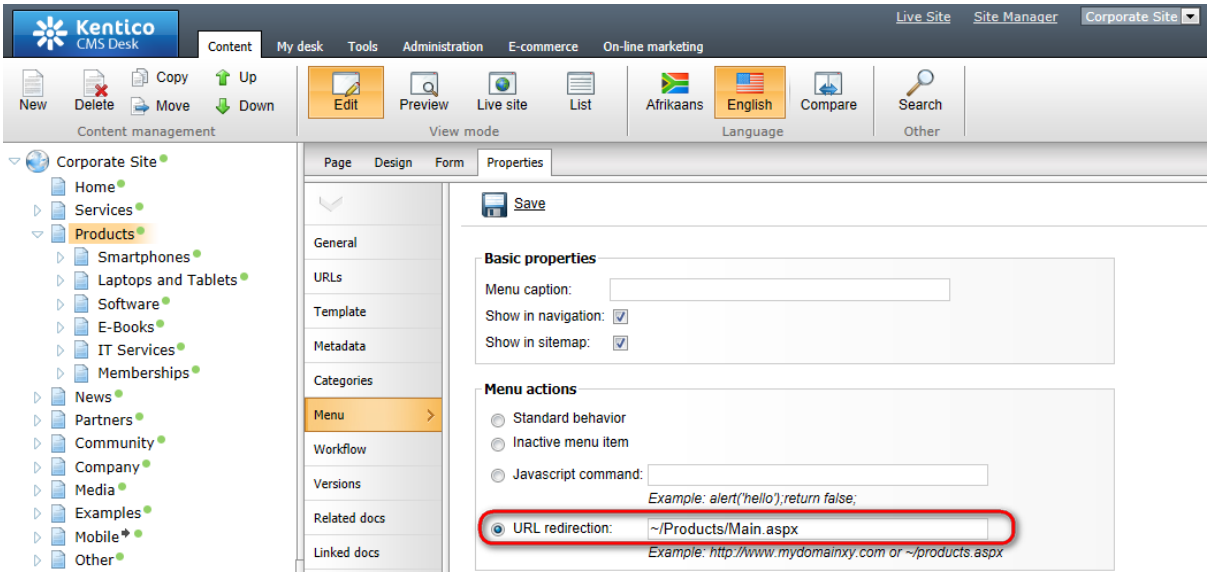
4. Open the web.config file of your web project and change value of the **mode** attribute of the

authentication tag to **Windows**. Also find the section marked with **Windows authentication BEGIN** and change the path parameter of the location tag to the name of the created directory, which will be **Products** in our case:

```
...
<authentication mode="Windows">

...

<!-- Windows authentication BEGIN -->
<location path="Products">
  <system.web>
    <authorization>
      <deny users="?"/>
      <allow users="*"/>
    </authorization>
  </system.web>
</location>
<!-- Windows authentication END -->
```

5. The authentication is now configured. If you try to access any of the menu items placed under the **Products** section, **Windows authentication** will be required. However, if you also want the authentication to be required for the **Products** main page (which is obviously not located under itself, hence requires no authentication now), you will have to use the following workaround.

Create a new page under the **Products** section, give it the same content as the main page has got and redirect the Products link in the menu to this new page. Because the new page is located under the **Products** section, windows authentication will be required for it.



7.13.12.5 Configuring mixed mode authentication

Mixed mode authentication enables users to sign in to your website using both Windows authentication and standard forms authentication.

Important!

During a sign-in, if an already existing forms user has the same user name as a domain user you want to sign in with, the system signs in the forms user. As a result, an account cannot be created for the domain user. You can avoid this behavior by renaming the existing forms user.

To enable mixed authentication mode:

1. Edit your application's *web.config* file.

2. Add the LDAP connection string of your Active Directory service into the **configuration/connectionStrings** section:

```
<connectionStrings>

  ...

  <add name="CMSADConnectionString" connectionString="<LDAP connection string>" />
</connectionStrings>
```

Replace the <LDAP connection string> text in the code above with the actual connection string. Enter it in according to the following format:

```
LDAP://mydomain.example.com/DC=mydomain,DC=example,DC=com
```

The first part is the full domain. In the second part, the same domain is divided into DC (domain component) units.

3. Modify the **membership** and **roleManager** elements under the **configuration/system.web** section according to the following:

```
<membership defaultProvider="CMSProvider" userIsOnlineTimeWindow="30">
  <providers>
    <clear/>
    <add name="CMSProvider" type="CMS.MembershipProvider.CMSMembershipProvider"
connectionStringName="CMSConnectionString" enablePasswordRetrieval="false"
enablePasswordReset="true" requiresQuestionAndAnswer="false"
requiresUniqueEmail="true" passwordFormat="Hashed"/>
    <add name="CMSADProvider"
type="CMS.MembershipProvider.CMSADMembershipProvider"
connectionStringName="CMSADConnectionString" connectionUsername="username"
connectionPassword="password" />
  </providers>
</membership>
```

```
<roleManager defaultProvider="CMSRoleProvider" enabled="true"
cacheRolesInCookie="true" cookieName=".ASPROLES" cookieTimeout="30" cookiePath="/"
 cookieRequireSSL="false" cookieSlidingExpiration="true" cookieProtection="All">
  <providers>
    <clear/>
    <add name="CMSRoleProvider" type="CMS.MembershipProvider.CMSRoleProvider"
connectionStringName="CMSConnectionString" applicationName="SampleApplication"
writeExceptionsToEventLog="false"/>
    <add name="CMSADRoleProvider" type="CMS.MembershipProvider.CMSADRoleProvider"
connectionStringName="CMSADConnectionString" connectionUsername="username"
connectionPassword="password" />
  </providers>
</roleManager>
```

Replace the following values:

- **username** - your own active directory user name, including the fully qualified domain name. For example, office.example.com\johns
- **password** - your active directory password

When you have entered this code into your *web.config*, users can log in using their Active Directory user name (without the domain) and password, or using their standard Kentico CMS user name and password.

Mixed authentication on Windows XP

When running the application on Windows XP, users need to enter AD usernames including the domain name: <*domain name*>*user*

You can also allow users to sign in using their full Active Directory user name (e.g. *MyName@office.example.com*). For this to work, you have to add the following key to the *AppSettings* section of your *web.config* file:

```
<add key="CMSADDefaultMapUserName" value="userPrincipalName" />
```

7.13.12.6 Integrating authentication with external systems

Kentico CMS allows you to write a custom authentication provider. In this way, the provided user name and password are checked against an external user profile source/authentication source and if the user is successfully authenticated, the user account is automatically created/updated in the Kentico CMS database, without copying the user password.

You can integrate your custom authentication provider with Kentico CMS using the Global events and their handling approach.

7.13.12.7 Single sign-on

Single sign-on is a feature which enables users to authenticate just once and then access multiple websites without the need to enter logon credentials again for each site. There are two ways how you can achieve this:

- Single sign-on on the same main domain - this approach lets you configure single sign-on for multiple sites running on the same main domain (e.g. site1.example.com, site2.example.com, etc.) in the IIS. The sites need not be using Kentico CMS.
- Single sign-on across different domains - this approach requires all websites to be running in a single instance of Kentico CMS, while they can use completely different domains.

The sections below describe necessary configuration for each approach.

Single sign-on on the same main domain

Single sign-on on the same main domain is supported in the following scenarios:

Forms Authentication

If you are using Forms authentication and you need to share user identity across applications that run on the same main domain while all of them use standard ASP.NET 2.0 Forms authentication, you need to ensure that:

1. All applications use the same user database or at least the same user names. You may need to integrate the authentication using a custom security handler.

2. The *web.config* file of all applications uses the same authentication cookie name and the path is set to "/":

```
<authentication mode="Forms">
  <forms name=".ASPXFORMSAUTH" path="/" ...="" />
</authentication>
```

3. The *web.config* file of all applications uses the same machine key. The machine key is not present in the web.config by default. You can generate is using various machine key generators that can be found on the Internet. Once you have a key generated, you can add it to the <*system.web*> section the following way:

```
<system.web>
  ...
  <machineKey validationKey="ABCD0708...." decryptionKey="DDFF8943...."
validation="SHA1" />
  ...
</system.web>
```

4. If your applications run on different sub-domains, such as *www.example.com* and *forums.example.com*, you need to set the domain attribute of the authentication cookie to the main domain so that it's shared across domains:

```
<forms name=".ASPXFORMSAUTH" path="/" domain=".mywebsite.com" ...="" />
```

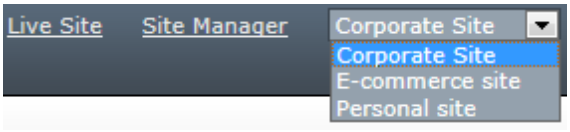
Windows Authentication

If you are using Windows authentication, the user identity is shared within the Windows domain. No additional configuration is required.

Single sign-on across different domains

Single sign-on across completely different domains in the same instance of Kentico CMS can be enabled by checking the **Automatically sign-in user when site changes** check-box in **Site Manager -> Settings -> Security & Membership**.

With this option enabled, no further configuration is necessary - users only need to enter their logon credentials once. After that, they can switch between different sites running in the CMS using the **Site** drop-down list in CMS Desk, without the need to enter their logon credentials for each domain.



The single sign-on functionality is also achievable on your custom pages using Kentico CMS API. The following code example shows how you can authenticate a user with a particular username in your code:

[C#]

```
string userName = "testuser";
// Authenticates user with specified user name
CMSContext.AuthenticateUser(userName, false);
```

The second code example shows how you can generate a URL with a user authentication token. When a user accesses this URL, they are automatically authenticated.

[C#]

```
string userName = "testuser";
// Get user with specified user name
UserInfo userInfo = UserInfoProvider.GetUserInfo(userName);
// Get authentication URL for specified user and target URL
string url = AuthenticationHelper.GetUserAuthenticationUrl(userInfo, "/
default.aspx");
// Redirect user to target URL and authenticate him
URLHelper.Redirect(url);
```

7.13.12.8 Displaying personalized content

Kentico CMS allows you to personalize the displayed content based on the current user.

Personalization in short

1. If you want to customize content displayed to the users, you need to grant or deny the READ permission to these users and turn on the **Check permissions** attribute of the appropriate web parts.

2. When the user is not authenticated, the system uses a special user **Public anonymous User (public)**.

Checking the permissions of the current user allows basic personalization, but the Content personalization module may be used to achieve much more advanced personalization scenarios according to completely custom conditions.

Example: Personalizing the Products menu

In this example, you will learn how to display the **Products** section only to members of the **Customers** and **Partners** roles and how to display the **Smartphones** category only to the members of the **Partners** role.

1. Sign in to **CMS Desk** as administrator.

2. Go to **Administration -> Roles**, select the *Corporate Site* from the **Sites** drop-down list and create new roles **Customers** and **Partners**.

3. Go to **Administration -> Users** and create a new user **Customer1** with the following values:

- **User name**: Customer1
- **Full name**: Testing Customer
- **Enabled**: true
- **Is editor**: no

Click **OK**. Go to the **Sites** tab and assign the user to the *Corporate Site*. Go to the **Roles** tab and add the user to the **Customers** role.

4. Create another user **Partner1** with the following values:

- **User name**: Partner1
- **Full name**: Testing Partner
- **Enabled**: true
- **Is editor**: no

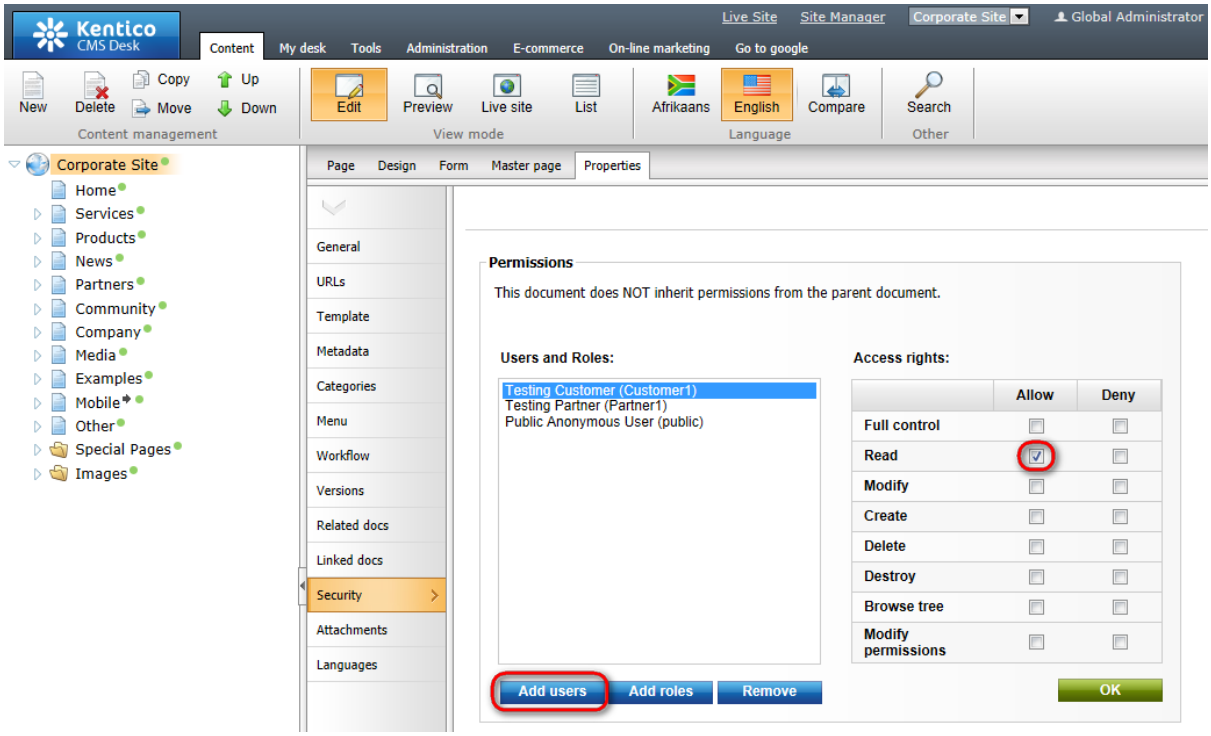
Click **OK**. Go to the **Sites** tab and assign the user to the *Corporate Site*. Go to the **Roles** tab and add the user to the **Partners** role.

5. Navigate to **Site Manager -> Settings -> Security & Membership** and set **Check page permissions** to *All pages*. This ensures that the security settings assigned in the following steps will be checked for all documents (pages) on the website.

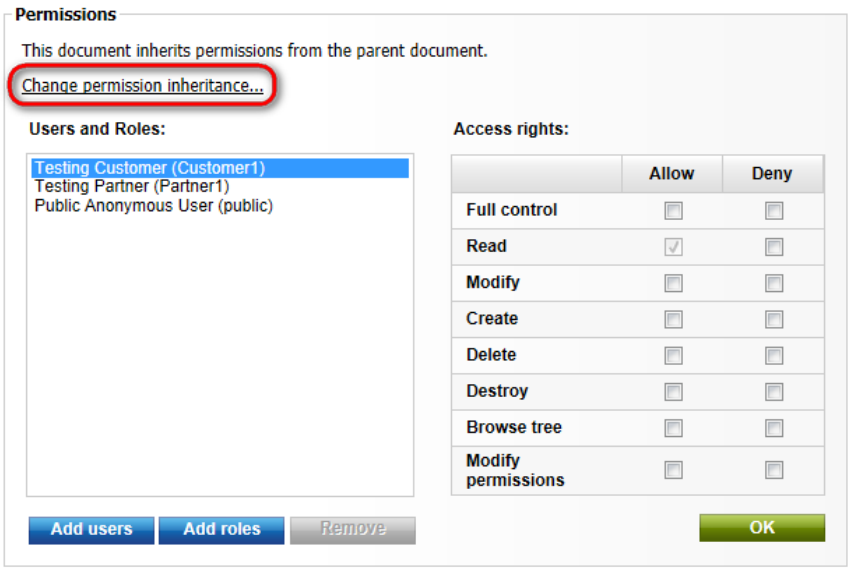
6. Switch to **CMS Desk -> Content** and click the root of the content tree. Switch to **Properties -> Security**.

First, we need to grant the **Read** permission for the whole website to the **Public anonymous user (public)**, **Customers** and **Partners** roles.

Click **Add users**, select these three users and click **OK**. When the users are added, grant the **Read** permissions to them using the check-box (☑), as you can see in the screenshot below. You need to click **OK** in order to save the settings.



7. Now we will hide the product categories from public users. In the content tree, click **/Products/Smartphones** and switch to the **Properties -> Security** tab. The permissions that you configured for the root of the content tree are inherited by this document. We need to break the inheritance. Click the **Change permission inheritance** link.



In the following dialog, choose **Break inheritance and copy parent permissions**.

8. Now select the **Public Anonymous User (public)** and deny the **Read** permission.

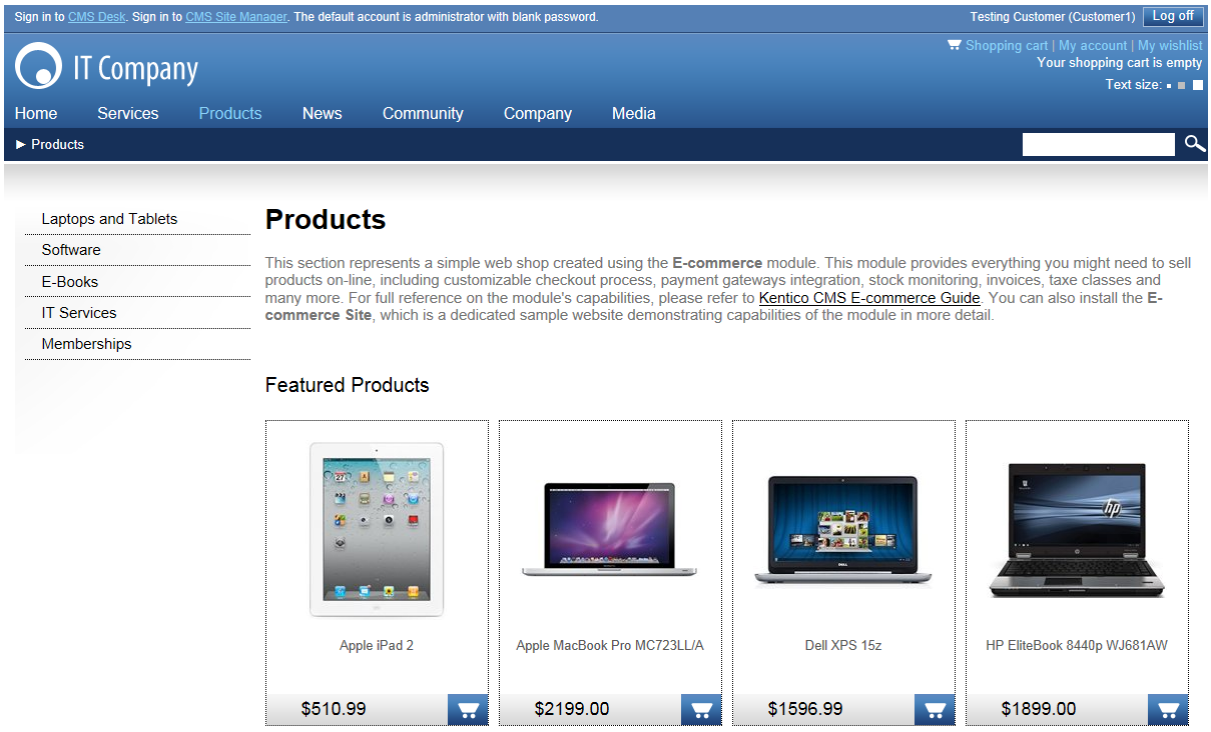
9. Repeat the last two steps for the **Laptops and Tablets** page. Also deny the **Smartphones** category for the **Customers** role.

10. The permissions are not checked by web parts by default, so we need to configure the web parts so that they check the **Read** permission of the current user.

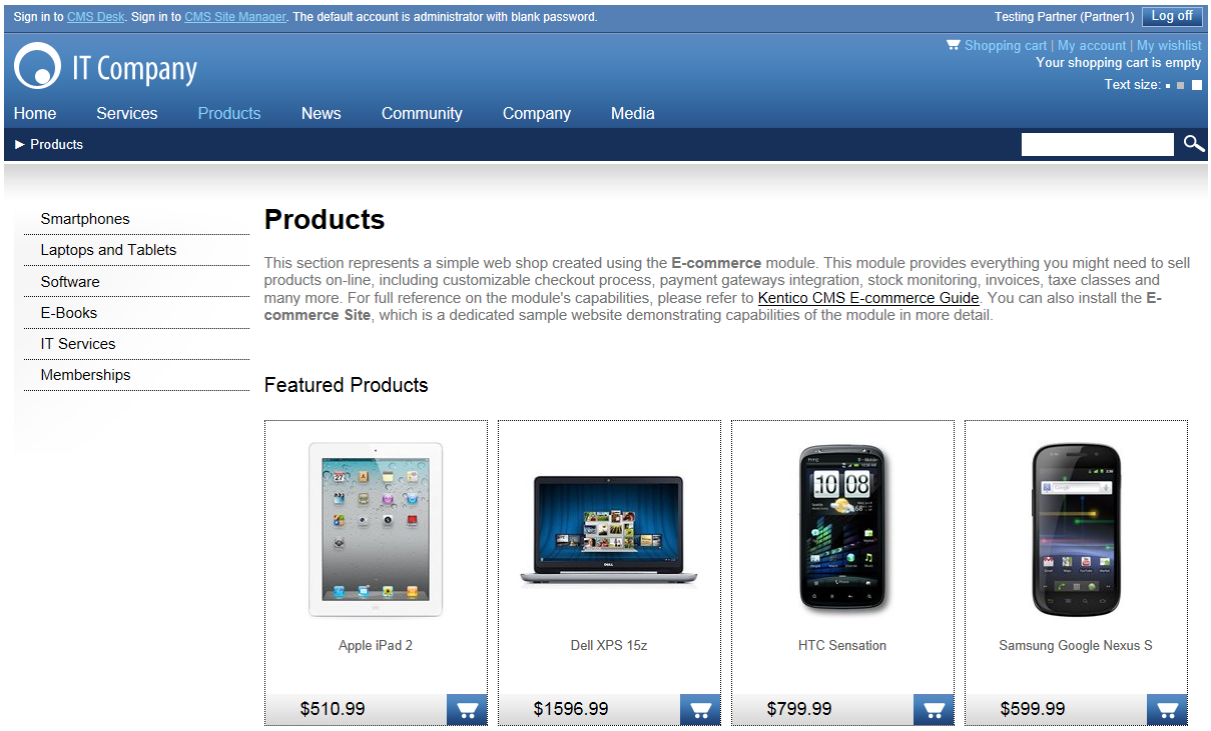
Choose the root in the content tree and click the **Design** tab. Configure ( ) the **Top list menu** (CSS list menu) web part and set its **Check permissions** property in the **System settings** category to true ( ). Click **OK**. Repeat the same for the **CSS list menu** and **Featured products** web parts on the **/Products** page.

11. Sign out. If you mouse-over the **Products** menu item as a public user, you will see that the sub-categories representing the restricted sections are no longer displayed.

12. Sign in as user **Customer1** (use the logon mini-form at the top right of the page) and navigate to **Products** via the main menu. You will see a page like the one below. As you can see, the **Smartphones** section and **Smartphone** products are not displayed to this user.



Now sign out and sign in again as user **Partner1**. You will see all all categories and products:

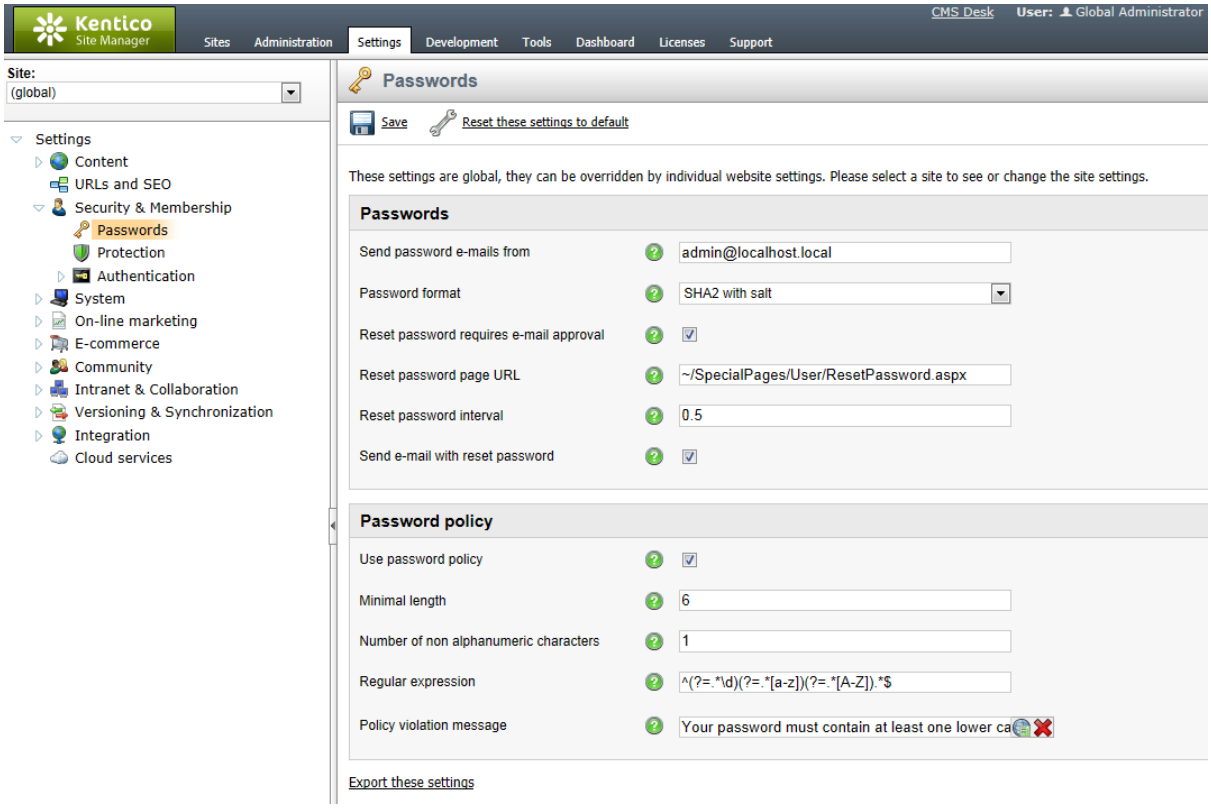


You have learned how to personalize the content displayed to users based on their permissions.

7.13.13 Password management

7.13.13.1 Overview

Passwords are a critical part of any authentication process. Kentico CMS provides various password-related features that you can leverage to achieve the level of security required by your website. These settings can be found in **Site Manager -> Settings -> Security & Membership -> Passwords**.



The features include:

- A set of password encryption format options.
- Ability to reset and recover user passwords.
- Ability to specify password strength and policy to enforce the specified strength.
- Ability to set time after which passwords expire and users are required to change them.
- Locking a user's account when they enter an incorrect password too many times.

7.13.13.2 Password format

There are multiple different formats that can be used to store passwords in the database. They may be saved either in plain text or as the result of a security hash function. You can choose which option should be used via the **Password format** setting.

By default, passwords are stored using the *SHA2* hash format with the additional application of a *salt*. A salt is a string that is appended to passwords before they are encrypted, which helps protect against dictionary or other types of brute force attacks. It also ensures that every user has a different password hash, even if multiple users have the same password. The GUID of each user is assigned as the salt for the password. If you wish to further increase the length of the salt, you can add the following key to the *I*

configuration/appSettings section of your web.config file:

```
<add key="CMSPasswordSalt" value="SaltText" />
```

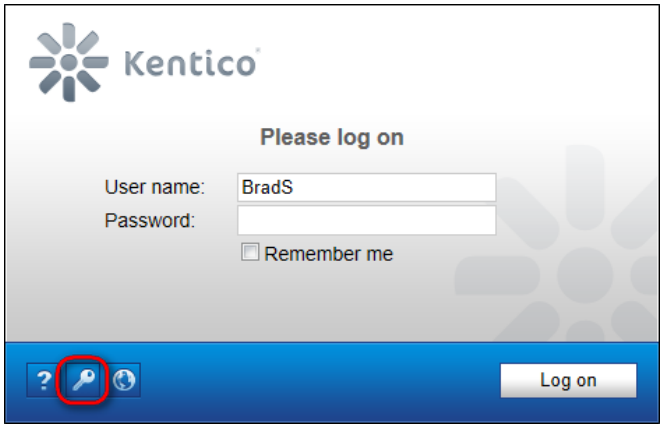
The value of this key will be added to the salt after the user GUID.

If you change the password format, please keep in mind that this only affects how future passwords will be stored. Existing passwords will remain unchanged. You will need to reset all passwords, so that they are stored in the new format. For this reason, it is recommended to set the appropriate format directly after installation, before you create user accounts or allow users to start registering.

7.13.13.3 Password recovery

If a user forgets their password, they may retrieve or reset it, provided they have access to the e-mail address specified for their account. A password may be recovered by submitting a request through one of the website's logon forms.

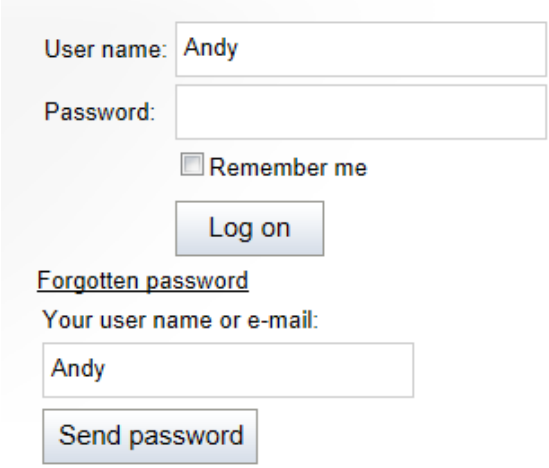
By default, a forgotten password button is included on the logon page of the CMS Desk and Site Manager administration interface.



You can hide the button by adding the following key to the **/configuration/appSettings** section of your web.config file:

```
<add key="CMSShowForgottenPassLink" value="false" />
```

On the live site, users can recover their password through Logon form web parts that have their **Allow forgotten password retrieval** property enabled.



When submitting the request, users can either type in their user name or e-mail address. If a user name is entered, the recovery e-mail will be sent to the given account's address. In cases where an address is used, the request will affect the password of the user account with the corresponding address. Password recovery e-mails are sent from the address specified in the **Send password e-mails from** setting.

Depending on the value of the **Reset password requires e-mail approval** setting, one of two possible password recovery modes will be used.

Password reset without e-mail approval

If the **Reset password requires e-mail approval** setting is *disabled*, then users who request their password will receive an e-mail containing the password directly. If the current password format is plain text, the existing password will be sent to the user. If an encrypted password format is used, the system will generate a new password for the user.

Password reset with e-mail approval

If the **Reset password requires e-mail approval** setting is *enabled*, several steps will be added to the process. Users who submit a password recovery request through a logon form will first receive an e-mail containing a link to a page where they can manually set a new password. This option is more secure, because the password cannot be read from the e-mail by potential attackers. Also, the reset link is only valid temporarily. The time period during which the link is valid can be specified in hours by the **Reset password interval** setting.

When a user clicks the link in the e-mail, they will be redirected to the default **~/CMSModules/Membership/CMSPages/ResetPassword.aspx** system page, where they will be able to enter a new password. The URL of the link contains a token in its query string that automatically identifies the user whose password should be changed. After someone visits the link, it becomes invalid and cannot be accessed again.

If you wish to use a custom page for this purpose, simply create a new page on the website according to your specific requirements and place the Reset password web part on it. This web part displays a form with the same functionality as described above for the **ResetPassword.aspx** system page. After you create the page, enter its URL into the **Reset password page URL** website setting, or into the same property of individual **Logon form** web parts.

If the **Send e-mail with reset password** setting is enabled, users will receive another e-mail containing their new password once they successfully reset it.

Global administrator password

If you happen to lose the password for your administrator account and cannot access the management interface, you can use on of the following techniques to recover:

- **Reset password via web.config key** - insert the following key to the **appSettings** section of your web.config:

```
<add key="CMSAdminEmergencyReset" value="<your username>;<your new password>;[true/false]" />
```

  The first and second parameter specify your user name and your new password, delimited by a semicolon. The third parameter is optional and indicates whether you want to create a new user with global administrator rights.

  The key will be automatically deleted after you gain access to the user interface.

- **Clear password in database** - find your user record in the **CMS_User** table and clear the contents of the **UserPassword** column. Then sign in to the administration interface with a blank password and set a new password.

Password recovery e-mail templates

The e-mails sent to users during the password retrieval process are based on E-mail templates, which can be found in **Site manager -> Administration -> E-mail templates**. The following password-related templates are available:

- **Membership - Forgotten password** - sent to users when they use the password recovery feature and the *Reset password requires e-mail approval* setting is disabled.
- **Membership - Change password request** - sent as a reply to password recovery requests if *Reset password requires e-mail approval* is enabled.
- **Membership - Changed password** - sent to users if their password is changed by an administrator, either manually or by generating a new one.
- **Membership - Resend password** - used if the current password information is sent to a user from the administration interface (this can only be done if passwords are stored in plain text format).

These templates can be edited as needed, so you may fully customize the content of the e-mails. You can enter the following context macros to include dynamic values in their text:

- **{% UserName %}** - the name of the user's account. If you are using site prefixes for user names, all occurrences of this macro in e-mail templates can have the prefix trimmed out with the following method: *{% TrimSitePrefix(UserName)%}*
- **{% Password %}** - the current (new) password of the given user.
- **{% LogonURL %}** - returns the URL of the page where the retrieval password request was submitted. Only available in the *Forgotten password* template.

The two macros below are available specifically in the **Change password request** template:

- **{% ResetPasswordURL %}** - resolves into the URL of the page where the user can change their password.
- **{% CancelURL %}** - returns the URL of a page that will cancel the request when opened. This can be used to create links that users can click in situations where someone else requested a new password for their user account (either intentionally or accidentally).

In addition to the special ones listed above, you can also use all other standard macro expressions in the templates. See the Development -> Macro expressions chapter of this guide for more information about macro expressions in Kentico CMS.

7.13.13.4 Password strength policy

The system can be configured to use a password policy, which means that new passwords entered by users will be validated according to a certain set of requirements. Passwords that do not meet the specified conditions will be rejected.

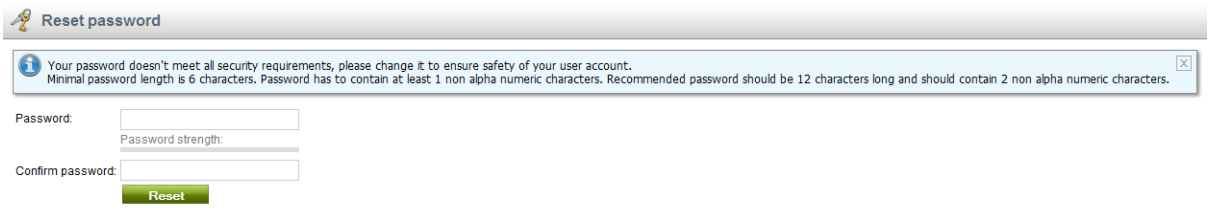
To enforce a password policy on your website, enable the **Use password policy** setting. The specific rules of the policy can be configured through the remaining settings in the category:

- **Minimal length** - sets the minimum number of total characters required for user passwords.
- **Number of non alphanumeric characters** - sets the minimum number of non alphanumeric characters (i.e. any character except for numbers and letters) that must be present in a password in order for it to be accepted.
- **Regular expression** - can be used to enter a regular expression that will be used to validate user passwords. For example: *^(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).*$*
  This sample expression would require passwords to contain at least one lower case letter, upper case letter and number. The minimum amount of characters would be determined by the other policy settings.

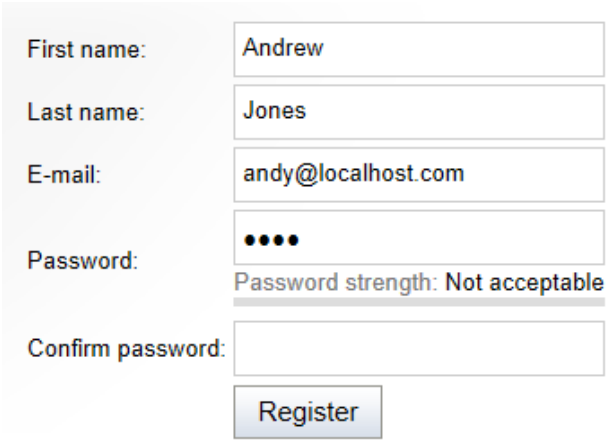
The requirements defined by all three settings are combined together to form the final password policy.

The policy is applied in all sections of the website where a new password can be entered. This includes various types of web parts that display forms on the live site, such as My account or the Registration form, and the administration interface (**Administration -> Users**). The requirements of the policy, except for the regular expression, are additionally observed when the system automatically generates new passwords. This is also the case if the **Use password policy** setting is disabled, so you can affect how passwords should be generated even if you do not wish to set a policy for your website's users.

When you introduce a password strength policy, existing users are by default allowed to keep their passwords unchanged. To force existing users to observe the policy, enable the **Force password policy on logon** setting. With this setting enabled, the system will check whether a user's password meets the policy requirements every time a user logs in. When the password doesn't meet the requirements, the user is presented with a form that allows to change the password.



When a user types in a password, it is validated in real time and its status is reflected by an indicator below the field. If a policy is set, passwords that do not fulfill the requirements will be rejected with the *Not acceptable* status.



Valid passwords will have a different status displayed according to their relative strength, which is calculated based on the recommended values for the total password length (*12* by default) and number of non alphanumeric characters (*2* by default). If a password policy is not enabled for the website, the current strength status of passwords will still be shown, but only as a recommendation and all passwords will be accepted.

To help users come up with an appropriate password, you can use the **Policy violation message** setting to specify a text message that will be displayed to users who attempt to enter a password that does not fulfill the requirements of the password policy. If left empty, a default message will be shown, informing about the minimum password length and number of non alphanumeric characters. If you wish to use a regular expression, it is recommended to describe its requirements in a custom message. If your site has multiple cultures (languages) assigned to it, you can enter a different message for each language via the **Localize** ( ) action available next to the setting's field.

Customizing the password strength indicator

You can change the recommended values that are used to calculate the password strength by editing the code of the appropriate controls:

To set different values globally for the entire application, edit the code behind of the **~/CMSModules/Membership/FormControls/Passwords/PasswordStrength.ascx** control and enter different numbers into the `mPreferedLength` and `mPreferedNonAlphaNumChars` variables.

You can also override the values for specific instances where this control is used through its **PreferedLength** and **PreferedNonAlphaNumChars** properties (e.g. in the code of the *Registration form* web part).

The appearance of individual password strength status labels may be customized through CSS styles. Each one has a different class assigned, e.g. *PasswordStrengthNotAcceptable*

Password policy and strength in custom forms

When creating custom forms, you can easily add password fields that validate according to the specified policy and display password strength.

To do this, specify either the **Password strength** or **Password with confirmation** form control for the given field, which will automatically ensure the functionality described above.

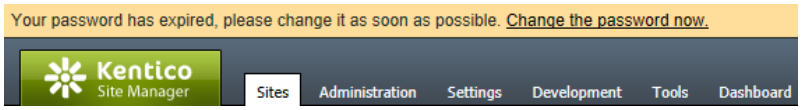
7.13.13.5 Password expiration

With the available password settings, you can set the passwords to expire after a specified amount of time.

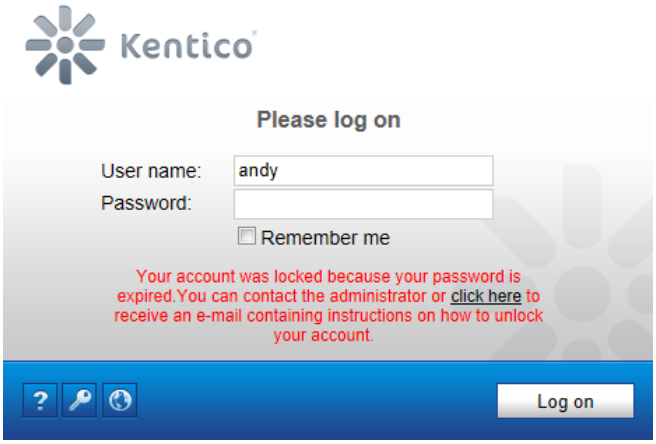
You can turn on password expiration with the **Enable password expiration** setting. When a user logs in to the system, the password expiration period (specified in days by the **Password expiration period** setting) is added to the time when the user last modified their password, and then compared with the current time. If the resulting time is earlier than the current time, the particular user's password has expired.

The expiration leads to one of the two following scenarios, which can be set by the **Password expiration behavior** setting.

- **Show warning** - displays a warning text. The user can click the **Change the password now** link to open a dialog that will allow them to conveniently change their password.



- **Lock account** - locks the user's account, requiring the user to unlock their account and change their password.



The system can warn the users that their password is about to expire. You can adjust the period during which users will be displayed with the warning via the **Password expiration warning period** setting.

Notifying live site users

By default, notifications related to password expiration are displayed only in the administration interface. To notify also live site users, place the **Password expiration** web part on a page.

Resetting a password

Users can change their expired password on a special page. You can either use the default page (*~/CMSModules/Membership/CMSPages/ResetPassword.aspx*), or specify a custom page in the **Reset password page URL** setting.

A custom password reset page should contain one of the following components:

- ***Reset password* web part** - a web part you can use in the Portal engine development model.
- ***ResetPassword* control** - an alternative to the Reset password web part, which can be placed on an ASPX page. The control is located in *~/CMSModules/Membership/Controls/ResetPassword.ascx*.

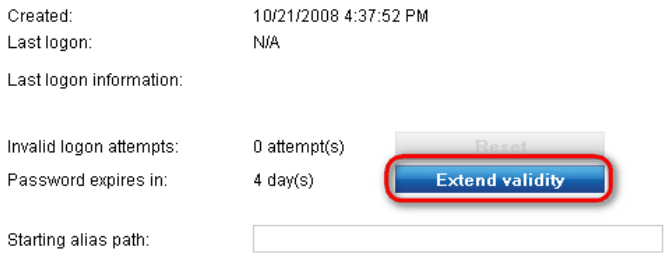
Notifying users by e-mail

By turning the **Send password expiration e-mail setting** on or off, you can specify whether you want to notify users about the expiration of their password via e-mail.

The **Site Manager -> Administration -> E-mail templates** section contains a predefined template (**Membership - Password expiration**) that is sent to users when their password expires. The template contains the *{% ResetPasswordUrl %}* macro, which is resolved to a link that points to the URL of the page that allows to change the user's password.

Extending password validity

To extend the validity of any user's password, edit the user in **CMS Desk or Site Manager -> Administration -> Users** and on the **General** tab, click **Extend validity**. The password's validity will be reset to the **Password expiration period** setting's value and the user enabled in case their account has been locked due to expired password.



7.13.13.6 Invalid logon attempts

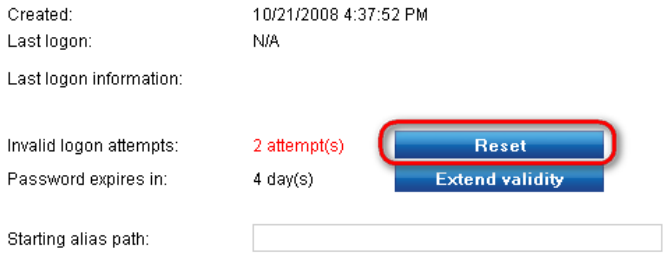
One of the most common threats to website security is stealing user accounts. To compromise an account, attackers use a simple method, which tries to guess the password for that account, either by combining different characters, or by selecting passwords from a dictionary. This threat can be easily eliminated by introducing a limit of invalid logon attempts, which means users will have their account locked after entering an incorrect password for the specified number of times. Users cannot log in to a locked account.

You can set up limiting the number of allowed invalid logon attempts in **Settings -> Security & Membership -> Protection** in the **Invalid logon attempts** group, which contains the following settings:

- **Maximum invalid logon attempts** - specifies the number of attempts to log in that the user can try before the system locks their account and denies access. If set to zero, account locking will be disabled.
- **Send unlock account e-mail** - indicates whether an e-mail should be sent to the user if their account gets locked.
- **Unlock user account path** - allows selecting the path (or typing in the URL) of a custom page, on which the user can unlock their account.

Resetting the number of invalid logon attempts

When you edit a user in **Site Manager -> Administration -> Users**, you can view the number of invalid logon attempts the user made in the **Invalid logon attempts** field. To reset the number back to zero and unlock (enable) the user's account in case the user has reached the limit, click the **Reset** button.



7.13.13.7 Unlocking an account

A user account can be locked for one of the following reasons:

- The user's password expires.

- The user reaches the limit of invalid logon attempts.

The following text describes how you can provide users with means to unlock their accounts.

Password expired

When an account is locked due to password expiration, the particular user will be required to change their password in order to unlock their account. To learn how to facilitate that, refer to the Password expiration topic.

Alternatively, you can extend the password's validity, as described in the Password expiration topic.

Invalid logon attempts exceeded

When an account is locked due to exceeding the number of invalid logon attempts, the particular user will have to manually unlock their account. You can enable them to do that by directing them to the *~/CMSModules/Membership/CMSPages/UnlockUserAccount.aspx* page.

Alternatively, you can create a custom page for unlocking accounts, on which you can place one of the following components:

- ***Unlock user account* web part** - a web part you can use in the Portal engine development model.
- ***UnlockUserAccount* control** - an alternative to the Unlock user account web part, which can be placed on an ASPX page. The control is located in *~/CMSModules/Membership/Controls/UnlockUserAccount.ascx*.

Notifications

You can then specify whether you want users to receive an e-mail notifying them that their account has been locked in the **Send unlock account e-mail** setting. The notification e-mail uses the **Membership - User account locked** template. You can inset a link to the account unlock page with the *{% UnlockAccountUrl %}* macro.

Users can also be notified directly when logging in. To enable this option, set the **Display account lock information message** setting to true. However, enabling this feature is not recommended, since it can reveal to a potential attacker the fact, that they've managed to lock a user's account.

7.13.14 Third-party authentication services

7.13.14.1 Overview

Third-party authentication services provide an alternative way how users can log in and register to your site. This way, they do not need to go through the registration process and create a new username and password for your site. Instead, they can use the same username and password that they already use on some popular site (like Windows Live, Facebook, LinkedIn, Google, Yahoo!, etc.). Even new users can log on to your site like this, in which case a new user account is created.

Kentico CMS supports the following authentication providers:

- Windows Live ID
- OpenID

- Facebook Connect
- LinkedIn

Click the links above to learn more about how to set up authentication using the particular provider.

7.13.14.2 Windows Live ID

7.13.14.2.1 Overview

Windows Live ID is a single sign-on service provided and maintained by Microsoft. By integrating Live ID into your website, you can allow site visitors to log in to your website using their Live ID login and password.

How to start using Windows Live ID

For this to work on your site, you have to do the following:

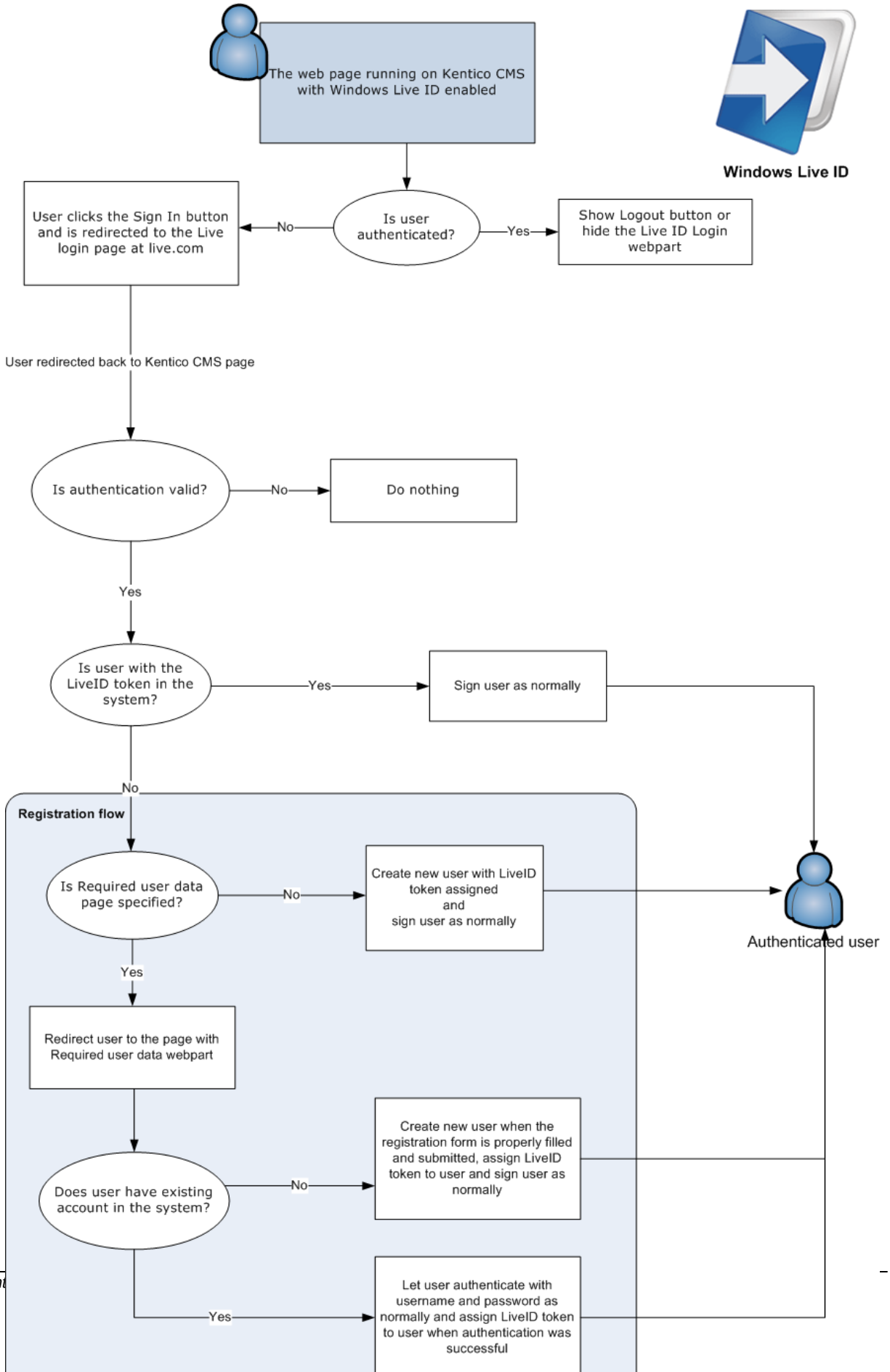
1. Register your website at https://manage.dev.live.com/ - to learn how this can be done, please refer to Registering your application.

2. Set up Kentico CMS for Live ID authentication - for more details, please see the Settings topic.

3. Use one of the Live ID web parts on the appropriate page on your site - more information in the Available web parts topic.

How it works

The following diagram shows how the process of Live ID login works.


The web page running on Kentico CMS with Windows Live ID enabled
Windows Live ID
Is user authenticated?
No
Yes
User clicks the Sign In button and is redirected to the Live login page at live.com
Show Logout button or hide the Live ID Login webpart
User redirected back to Kentico CMS page
Is authentication valid?
No
Do nothing
Yes
Is user with the LiveID token in the system?
Yes
Sign user as normally
No
Registration flow
Is Required user data page specified?
No
Create new user with LiveID token assigned and sign user as normally
Authenticated user
Yes
Redirect user to the page with Required user data webpart
Does user have existing account in the system?
No
Create new user when the registration form is properly filled and submitted, assign LiveID token to user and sign user as normally
Yes
Let user authenticate with username and password as normally and assign LiveID token to user when authentication was successful

Registration approval and double opt-in

If your site is configured to require registration approval or double opt-in, first-time users who attempt to log in with their Live ID will be redirected to the standard logon page without any further information, which may lead to confusion.

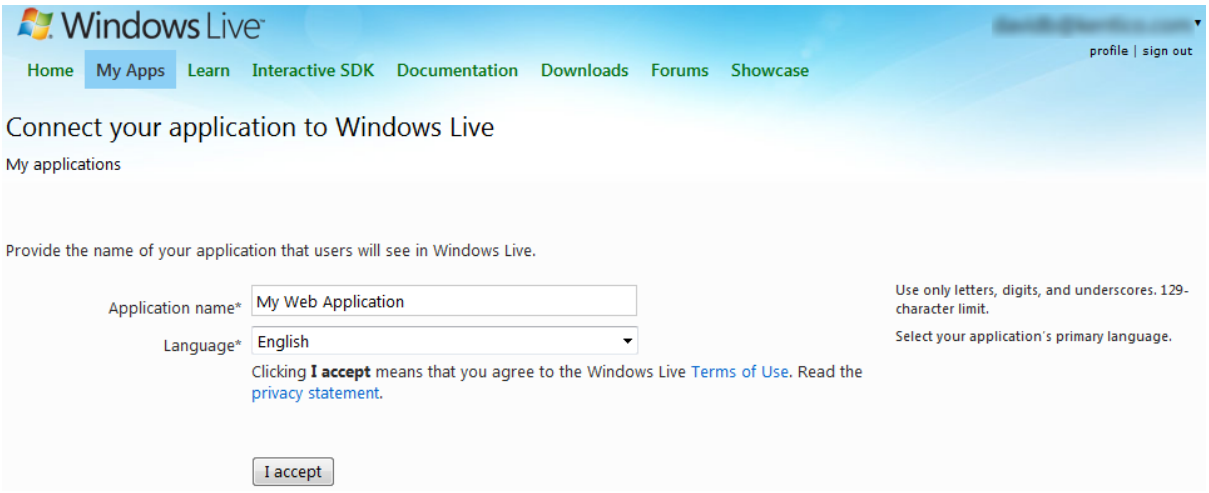
This issue can be avoided by creating a **Required user data page** (described later in this chapter) where users must enter an e-mail address for their account. When this is done, users will receive a notification e-mail about the status of their account.

7.13.14.2.2 Registering your application

To enable Live ID logon for your website, you must register it on the Windows Live application management site at the following address: https://manage.dev.live.com/.

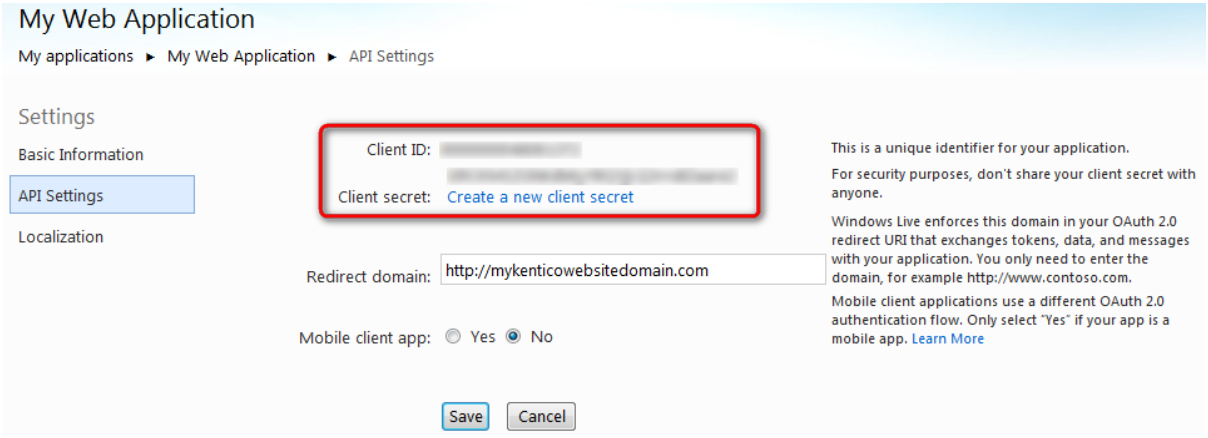
1. Once on the page, sign in with your Windows Live ID (you may need to create an account if you don't already have one) and you will be redirected to the application registration page.

2. Here, enter an appropriate **Application name** that will be used to identify your website in the Windows Live user interface (click **Create application** if you already have an application registered and wish to add another one):



Read the terms of use and the privacy statement, then click the **I accept** button to register the application.

3. On the next screen, click on the **Application Settings Page** link and switch to the **API Settings** tab. Type the fully qualified domain name of your website into the **Redirect domain** field and select the appropriate **Mobile client app** option.



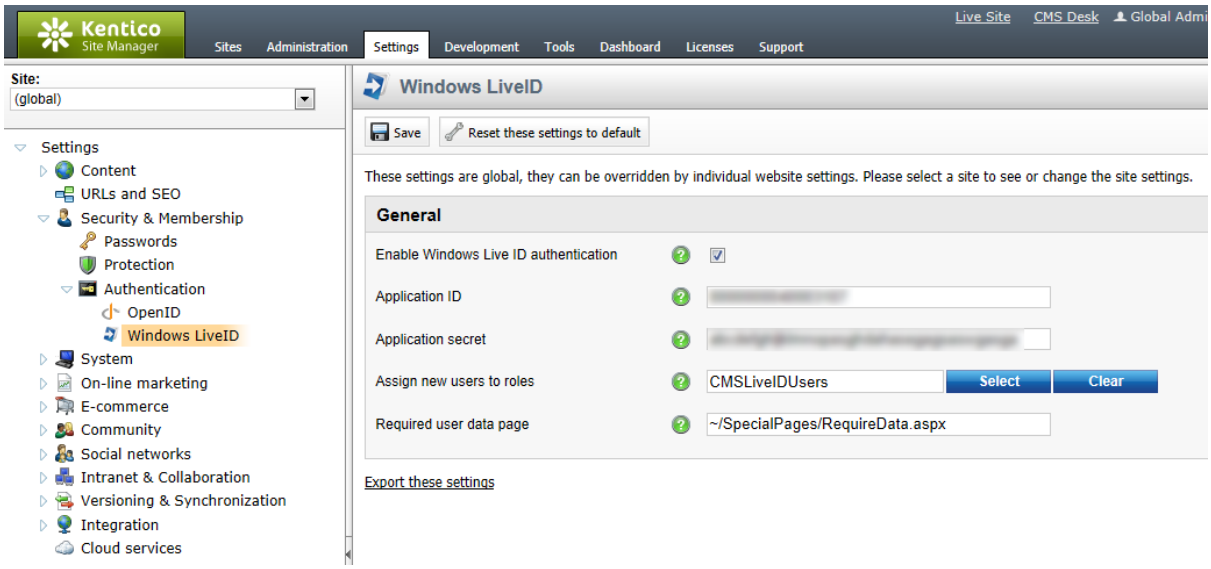
Click **Save** when you are done. Note the **Client ID** and **Client secret** values, since they will be required later when filling in the Live ID authentication settings in Kentico CMS.

4. You can also specify additional details for your application on the **Basic Information** tab, such as a logo or links to the terms of service and privacy policy of your website. The **Localization** tab may be used to set a different application name for individual languages. These options affect the logon page to which users will be redirected when they try to authenticate on your website using Live ID.

7.13.14.2.3 Settings

Live ID authentication settings are located in **Site Manager -> Settings -> Security & Membership -> Authentication -> Windows LiveID**. Before you start entering values, make sure you have the right site selected using the **Site** drop-down list in the top left part of the page.

- **Enable Windows Live ID authentication** - indicates if Windows Live ID authentication should be enabled.
- **Application ID** - identifier of your website. Enter the *Client ID* that you received when registering your website.
- **Application secret** - secret code that will be used to encrypt messages transferred between your website and the Live ID server. Enter the *Client secret* that you received when registering your website.
- **Assign new users to roles** - new users registered via Live ID authentication will be assigned to the roles specified here.
- **Required user data page** - URL of a page containing the *Required LiveID user data* web part. If entered, new Live ID users who log into the site will not have their user account created immediately, but will first be redirected to the specified page where they will be required to enter some additional data (or merge with an existing account) using the web part.



Compatibility with Live ID users created in older versions

Due to changes in the Windows Live ID service, Kentico CMS currently uses a different authentication mode (by default) than versions prior to 6.0, i.e. 5.5 R2 or older. Each mode generates a different authentication token for the same Live ID. As a result, users created under the original mode cannot be recognized or authenticated by the new one.

If your system contains Live ID users from an older version (e.g. after performing an upgrade procedure or as a result of a user import), you may wish to switch back to the original authentication mode in order to preserve the functionality of these user accounts. To do this, add the **CMSUseServerSideLiveIDAuthentication** key into the **/configuration/appSettings** section of your *web.config* file, as shown below:

```
<add key="CMSUseServerSideLiveIDAuthentication" value="false" />
```

Setting its value to *false* will ensure that backward compatibility is kept. Please note that new users registered via Live ID authentication while this key is false will only work with the original mode (additionally, Live ID users created under the new mode will no longer be recognized). Working with both authentication modes at the same time is currently not possible.

7.13.14.2.4 Available web parts

After you register your site at https://manage.dev.live.com/ and enter the necessary settings, you can use the following two web parts on your website to allow authentication via Windows Live ID. The web parts are located under the **Membership -> LiveID** category in the web part catalog.

Windows LiveID

This web part can be used to let site visitors sign in to your website using their Live ID. It can be placed on any page of your website. The web part is hidden to authenticated users and will be displayed only to unauthenticated public site visitors. With default settings, the web part appears as in the following screenshot.

Although the web part works fine with default settings, it has the following properties to fine-tune its behavior:

- **Sign in text** - if entered, a link with the entered text will be used instead of the default sign in image.
- **Sign out text** - if entered, a link with the entered text will be used instead of the default sign out image.
- **Show sign out** - if enabled, the web part will display a sign out link to authenticated users.
- **Show as button** - if enabled, buttons will be used instead of standard text links.
- **Sign in image** - specifies the URL of an image that will be used for the sign in button if the *Sign in text* property is empty.
- **Sign out image** - specifies the URL of an image that will be used for the sign out button if the *Sign out text* property is empty.

- **Notify administrator about new registrations** - if enabled, a notification e-mail will be sent to the website administrator when a new registration is performed via the web part.

LiveID required data

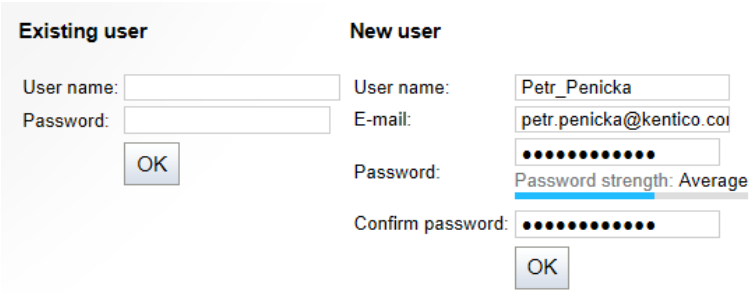
In some cases, you may want new users to provide some extra details before their account is created. For example, if your site is configured to require registration approval or double opt-in, all users need a valid e-mail address to help activate their account.

This can be achieved using the *Live ID required data* web part. The web part must be placed on the page specified by the **Required user data page** in **Site Manager -> Settings -> Security & Membership -> Authentication -> Windows LiveID**.

The following properties of the web part are the most important:

- **Allow forms authentication** - if checked, new users will be able to enter a password for their new account so that they can log in the usual way as well as via LiveID.
- **Allow existing user** - if enabled, users are allowed to join their existing account with their Live ID.
- **Default target URL** - if no return URL is passed, users will be redirected to the URL entered here after merging or creating their account.
- **Hide for no LiveID** - if checked, the web part will be hidden if the page with it is displayed without being a Live ID logon request (e.g. when accessed by entering its URL into the browser).

Other specific web part properties are explained in Kentico CMS Web Parts reference or after clicking the help link (?) link in the top right corner of the web part properties window.



The screenshot above shows how the web part looks with both the **Allow forms authentication** and **Allow existing user** options enabled. In the left part, existing site users can merge their current user account with the Live ID just by entering their user name and password. In this case, the Live ID Token will be added to the LiveID field of the existing user account. New users can enter the required details in the right part of the web part.

7.13.14.3 OpenID

7.13.14.3.1 Overview

OpenID is an open, decentralized standard for authenticating users. It is currently being used by popular websites like Google, Yahoo!, MySpace, Flickr and many more. Logon credentials used on all of these sites can be used to log on to your Kentico CMS site.

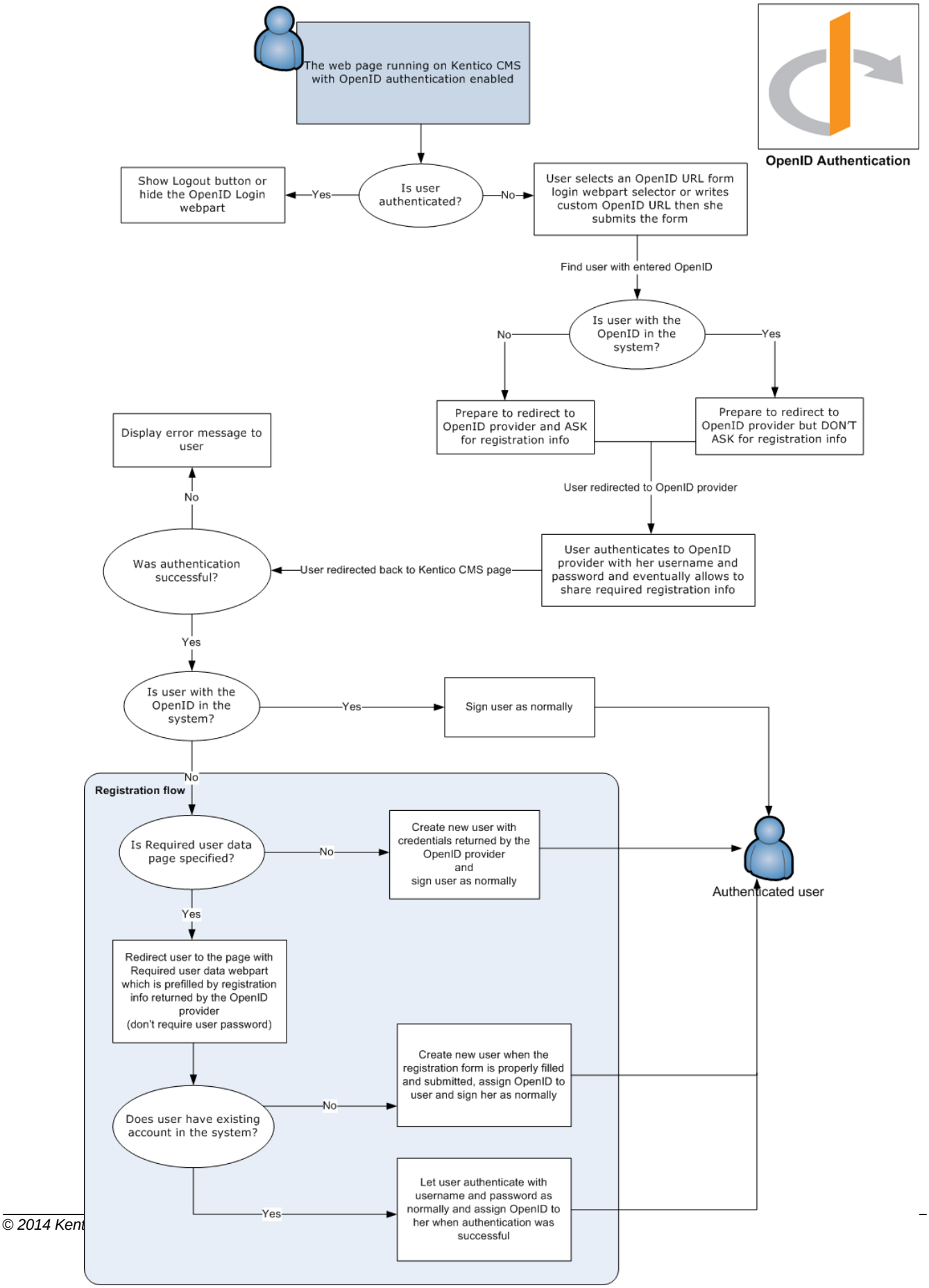
How to start using OpenID

OpenID is easier to use then the rest of the supported third-party authentication services. For it to work on your site, you don't need to register your site, all you need to do is take the following steps:

1. Set up Kentico CMS for OpenID authentication - see the Settings topic.
2. Use one of the OpenID web parts on any page of your site - see the Available web parts topic.

How it works

The following diagram shows how the process of OpenID login works:

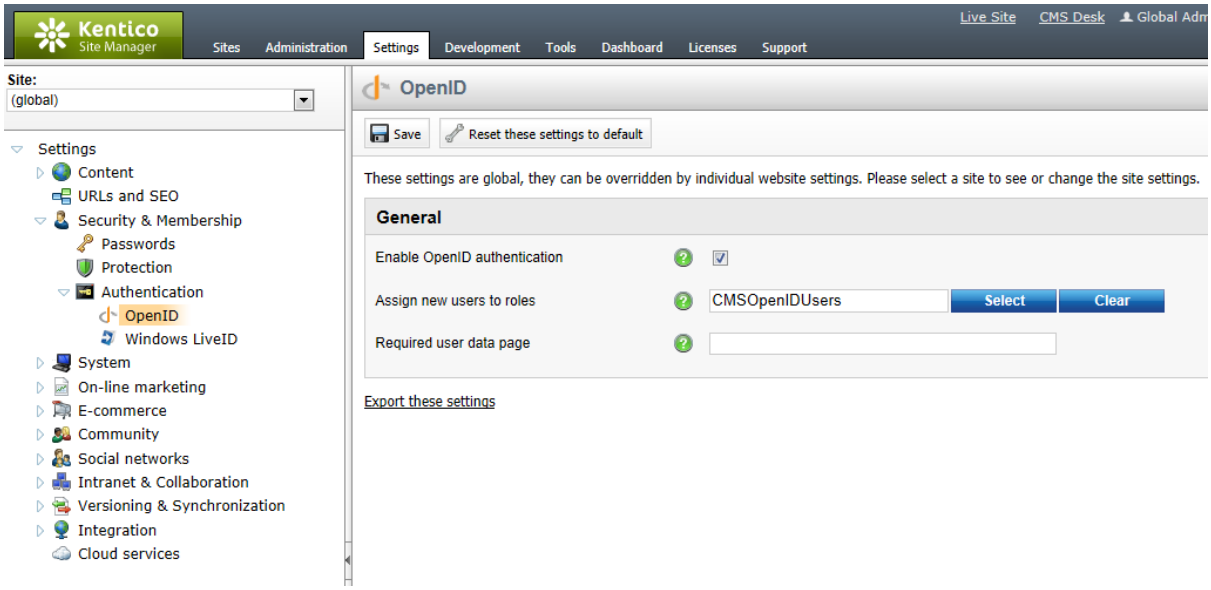

The web page running on Kentico CMS with OpenID authentication enabled
OpenID Authentication
Is user authenticated?
Yes
Show Logout button or hide the OpenID Login webpart
No
User selects an OpenID URL form login webpart selector or writes custom OpenID URL then she submits the form
Find user with entered OpenID
Is user with the OpenID in the system?
No
Yes
Prepare to redirect to OpenID provider and ASK for registration info
Prepare to redirect to OpenID provider but DON'T ASK for registration info
User redirected to OpenID provider
User authenticates to OpenID provider with her username and password and eventually allows to share required registration info
User redirected back to Kentico CMS page
Was authentication successful?
No
Display error message to user
Yes
Is user with the OpenID in the system?
Yes
Sign user as normally
No
Registration flow
Is Required user data page specified?
No
Create new user with credentials returned by the OpenID provider and sign user as normally
Yes
Redirect user to the page with Required user data webpart which is prefilled by registration info returned by the OpenID provider (don't require user password)
Does user have existing account in the system?
No
Create new user when the registration form is properly filled and submitted, assign OpenID to user and sign her as normally
Yes
Let user authenticate with username and password as normally and assign OpenID to her when authentication was successful
Authenticated user

7.13.14.3.2 Settings

First of all, you need to browse to the **~/Bin/DotNetOpenAuth.dll.rename** file and rename it to **DotNetOpenAuth.dll**. The library is renamed in the default installation because it doesn't support medium-trust environment. Then you need to configure OpenID settings in the administration interface of the CMS.

OpenID authentication settings are located in **Site Manager -> Settings -> Security & Membership -> Authentication -> OpenID**. Before you start making the settings, make sure you have the right site selected using the **Site** drop-down list at the top left part of the page.

- **Enable OpenID** - indicates if OpenID authentication is enabled.
- **Assign new users to roles** - new users registered via OpenID will be assigned to these roles.
- **Required user data page** - URL of the page where the *OpenID required data* web part resides; if entered and a new OpenID user logs in to the site for the first time, their user account is not created automatically, but they are redirected to this page and required to enter some additional data (or merge with an existing account) using the web part.



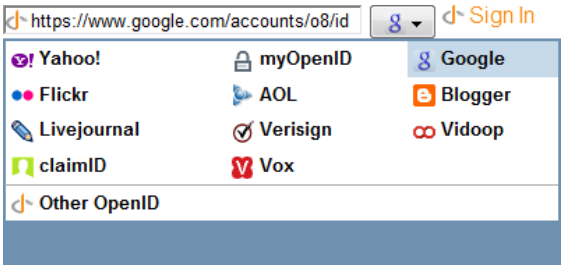
7.13.14.3.3 Available web parts

After configuring the settings, you can use the following two web parts on your website to allow OpenID authentication. The web parts are located under the **Membership** category in the web part catalog.

OpenID logon

This web part can be used to let site visitors sign in to your website using their OpenID. It can be placed on any page of your website. With default settings, the web part appears as in the screenshot below.

It lets you choose from a number of websites which support OpenID and where you might already have an account. If you choose one and click Sign In, you will be redirected to the logon page on that site. After successful logon, you will be redirected back to the Kentico CMS site where you came from, but you will be authenticated and a new account will be created in case this is your first logon.



Even though the web part works fine with default settings, it has the following specific properties to fine-tune its behavior:

- **Providers** - Providers used for OpenID login. Each provider must be specified on a single line. Total number of 3 parameters should be included for each provider:

  1 - provider display name
  2 - provider login URL
  3 - provider icon name placed in *~/CMSWebParts/Membership/OpenID/OpenID_files/*.

  Each parameter must be separated by '|'. The third parameter is optional and if not supplied then the default OpenID icon will be displayed. Provided URL must be the login URL of the OpenID provider. If the username (or blog name, user id, etc.) is part of the URL, then use the *##username##* macro to replace the username part of the URL.

  Example: *myOpenID|http://##username##.myopenid.com/|myopenid.ico*

- **Display textbox** - indicates if the OpenID provider textbox should be visible; if disabled then only the sign in button will be visible
- **Sign in text** - if entered, a link with the entered text will be used instead of the default sign in image
- **Sign out text** - if entered, a link with the entered text will be used instead of the default sign out image
- **Show sign out** - if checked, the sign out link will be displayed after the user logs in
- **Show as button** - if checked, buttons will be used instead of links
- **Sign in image** - if set, the image will be used as the sign in link
- **Sign out image** - if set, the image will be used as the sign out link

- **Required data for new users** - using these settings, you can request additional data from the OpenID provider, which will be added to the newly created user account. In case that you are using the *OpenID required data* web part, the requested data will be pre-filled in the web part.

OpenID required data

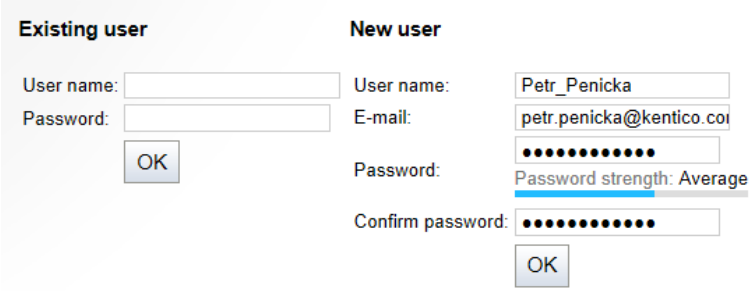
In some cases, you may want new users to provide some extra details before their account is created. For example, if your site is configured to require registration approval or double opt-in, all users need a valid e-mail address to help activate their account.

This can be achieved using the *Open ID required data* web part. The web part must be placed on the page specified by the **Required user data page** value in **Site Manager -> Settings -> Security & Membership -> Authentication -> OpenID**.

The following properties of the web part are the most important:

- **Allow forms authentication** - if checked, new users will be able to enter a password for their new account so that they can log in the usual way as well as via OpenID
- **Allow existing user** - if enabled, users are allowed to join their existing account with OpenID
- **Default target URL** - if no return URL is passed, users will be redirected to the URL entered here after merging or creating the account
- **Hide for no OpenID** - if checked, the web part will be hidden if the page with it is displayed without OpenID logon (e.g. when accessed by entering its URL into the browser)

Other specific web part properties are explained in Kentico CMS Web Parts Reference or after clicking the help link (?) link in the top right corner of the web part properties window.



The screenshot above shows how the web part looks with both the **Allow forms authentication** and **Allow existing user** options enabled. In the left part, an existing site users can merge their current user account with the OpenID by just entering their user name and password. In this case, the OpenID will be added to the OpenID field of the existing user account. New users can enter the required details in the right part of the web part.

7.13.14.4 Facebook Connect

7.13.14.4.1 Overview

Facebook is one of the most popular social networking sites in the world. With its continuously growing number of registered users, it is likely that visitors who come to your site will already have a Facebook account. In this case, it could be convenient for them to use their Facebook logon details than to go through another registration on your site. That's why you should consider using Facebook Connect authentication.

How to start using Facebook Connect

For Facebook Connect to work on your site, you need to register the site. Follow these steps:

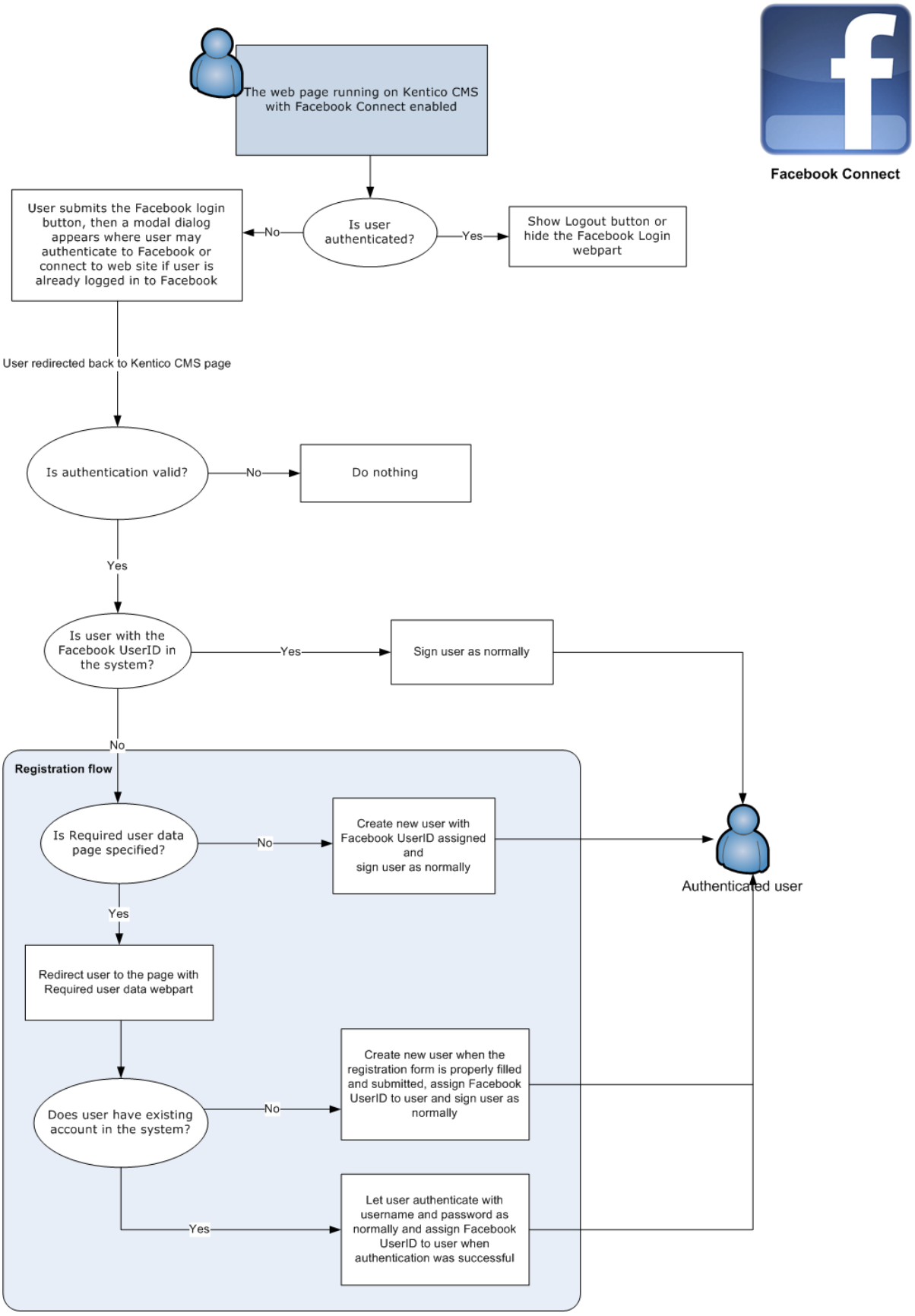
1. Register your website at http://www.facebook.com/developers/createapp.php - for more information, see the Registering your application topic.

2. Set up Kentico CMS for Facebook Connect authentication - see the Settings topic.

3. Use one of the Facebook Connect web parts on any page of your site - see the Available web parts topic.

How it works

The following diagram shows how the process of Facebook Connect login works:



7.13.14.4.2 Registering your application

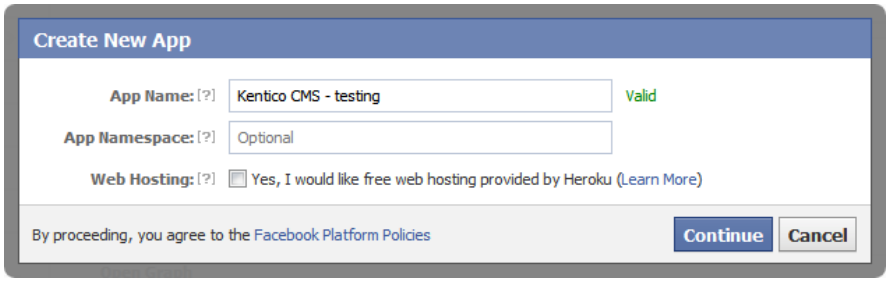
To use Facebook Conntect authentication on your website, you need to register the site at http://www.facebook.com/developers/createapp.php. You need to have a Facebook account to get to the page (you can create it here http://www.facebook.com/).

The following text describes the registration procedure and points out important information that you obtain during it. Please note that the design of the screenshots below may not be identical as Facebook changes its design quite often.

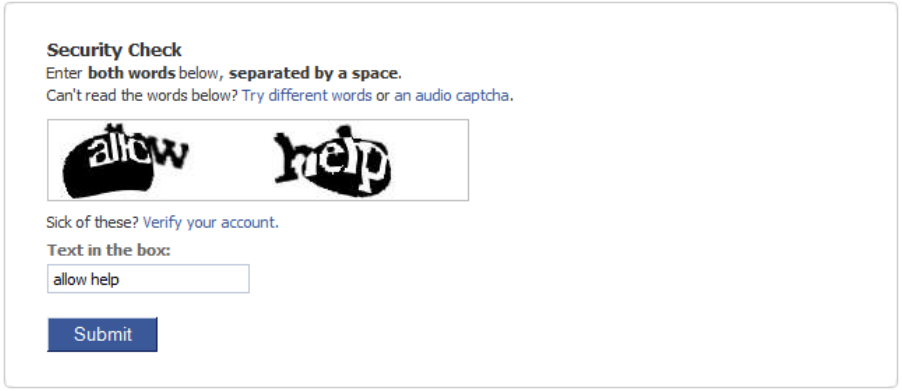
1. With a facebook account created and logged in, visit http://www.facebook.com/developers/createapp.php. Enter the following details into the form:

- **App Name**: enter the name of your website, e.g. Kentico CMS - testing

Read and **Agree** to the *Facebook Platform Policies* and click **Continue**.



2. Retype the CAPTCHA on and click **Submit**.



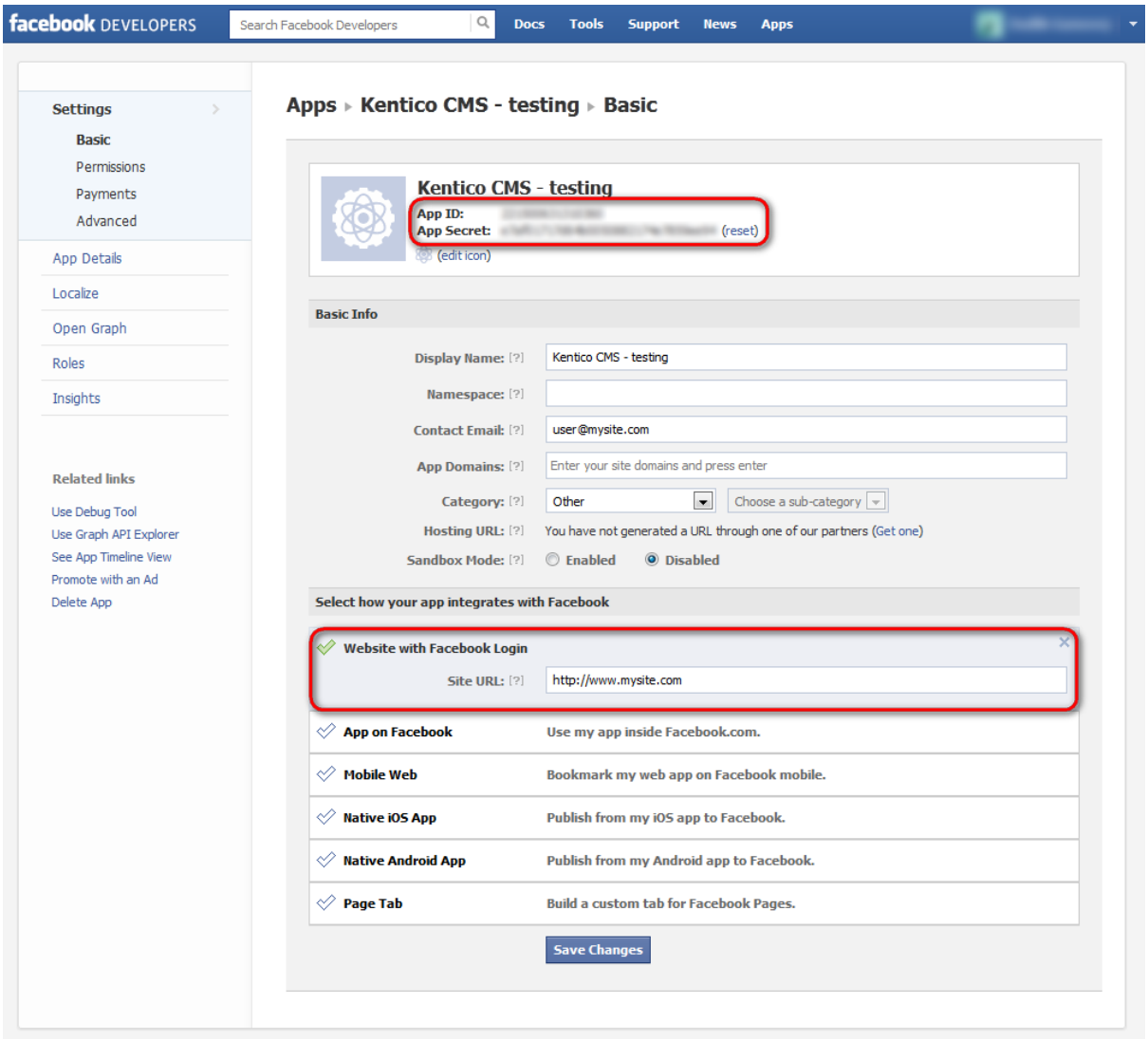
3. Your application will be registered and you will see the **Basic** tab of its editing interface. Here, you should focus on the two keys highlighted in the screenshot below, as you will need to enter them in **Site Manager -> Settings -> Social networks -> Facebook**:

- **App ID** - this is what you will enter into the **API Key** field
- **App secret** - this is what you will enter into the **Application secret** field

Copy them accordingly into your Kentico CMS instance as described in the Settings chapter.

4. Check **Website with Facebook login** option. Here, you need to enter the URL of your website into

the **Site URL** field. Click **Save Changes**.



Authentication on localhost

Using Facebook Connect authentication on **localhost** may result in incorrect behavior of the authentication. This behavior is by design of the Facebook Connect API. To achieve the required functionality, a standard domain name must be used and the domain name entered in this step must match the domain where your web application is actually running.

Your web application is now registered at Facebook. With proper settings and web parts, users will be able to authenticate to your site using their Facebook logon details. You can also set up Autoposting of content to Facebook directly from CMS Desk.

7.13.14.4.3 Settings

Facebook Connect authentication settings are located in **Site Manager -> Settings -> Social networks -> Facebook**. Before you start adjusting the settings, make sure you have the right site selected using the **Site** drop-down list at the top left part of the page.

General

- **API key** - key obtained during registration of your application on Facebook (as described in Registering your application topic).
- **Application secret** - key obtained during registration of your application on Facebook.

The two following settings are not required for Facebook Connect authentication. They are used only for the purposes of auto-posting on a website's Facebook page.

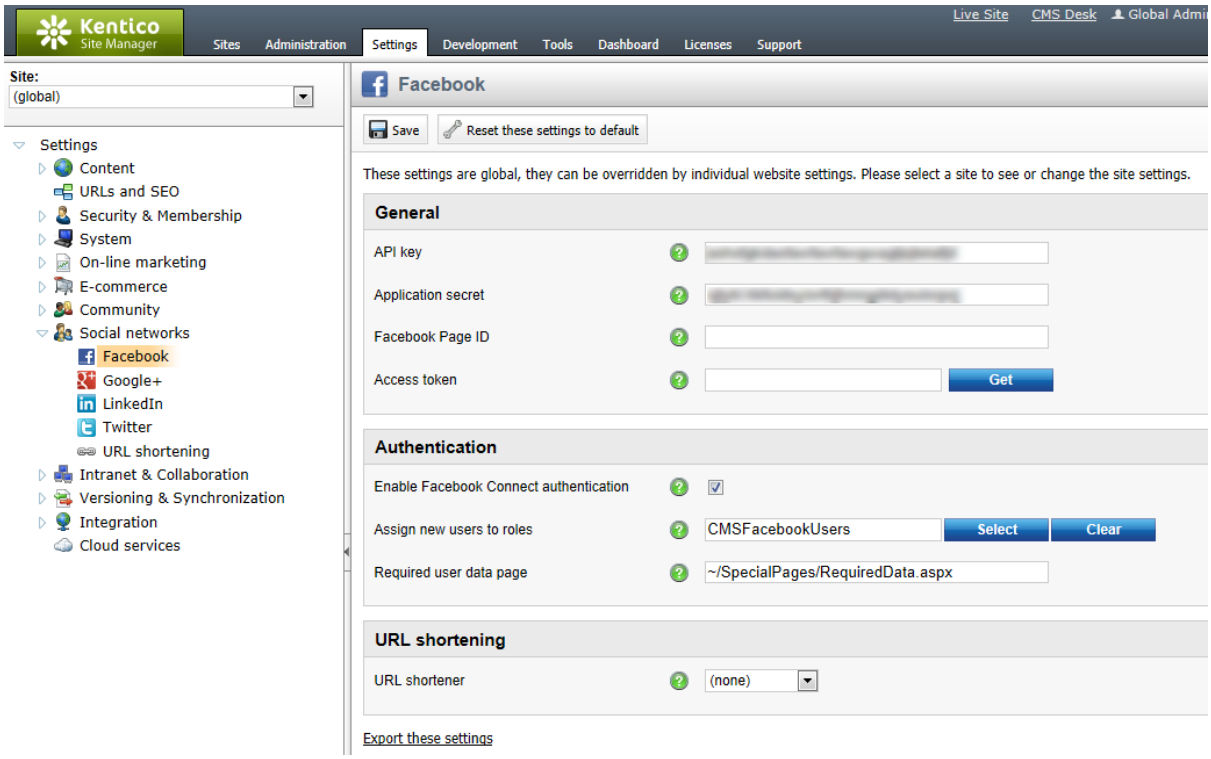
- **Facebook Page ID** - Specifies the Page ID (approximately 15 digits) obtained from the URL of the Facebook page associated with the website.
- **Access token** - Click the **Get** button to retrieve a token from Facebook after confirming permissions for the application. The API key, Application secret and Facebook Page ID fields must be filled in. When the token is retrieved, you can see it's expiration date. After it expires or when auto-posting is no longer functional, click the **Get** button again to renew the token.

Authentication

- **Enable Facebook Connect** - indicates if Facebook Connect authentication is enabled.
- **Assign new users to roles** - new users registered via Facebook Connect will be assigned to these roles.
- **Required user data page** - URL of the page where the *Facebook Connect required data* web part resides. If entered and a new Facebook Connect user logs in to the site, their user account is not created automatically, but they are redirected to this page and required to enter some additional data (or merge with an existing account) using the web part.

URL shortening

- **URL shortener** - May be used to select a URL shortening service that will be used for all links in Facebook posts.



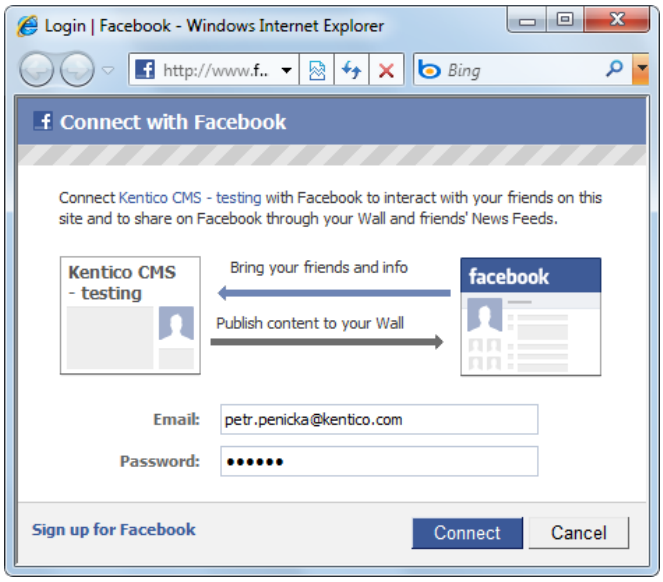
7.13.14.4.4 Available web parts

After configuring the settings, you can use the following two web parts on your website to allow Facebook Connect authentication. The web parts are located under the **Membership** category in the web part catalog.

Facebook Connect logon

This web part can be used to let site visitors sign in to your website using their Facebook username and password. It can be placed on any page of your website. The web part is hidden to authorized users and will be displayed only to unauthorized public site visitors. With default settings, the web part appears as in the screenshot below.

If you click the button, a new pop-up window opens, letting you log on using your Facebook username and password.



After successful logon, you will be redirected back to the Kentico CMS site where you came from, but you will be authenticated and a new account will be created in case that this is your first logon.

Even though the web part works fine with default settings, it has the following specific properties to fine-tune its behavior:

- **Sign in text** - if entered, link with the entered text will be used instead of the default sign in image
- **Size** - you can choose from 5 pre-defined sizes of the button
- **Length** - specifies which text label to use on a button with size specified as *small*, *medium*, *large*, or *xlarge*; specify short for the text label *Connect* only or long for the text label *Connect with Facebook*
- **Use latest version** - check this box if you want to use the latest Facebook Connect login buttons; don't check this box if you need to use the original Facebook Connect login buttons

- **Show sign out** - if checked, sign out link will be displayed after the user logs in
- **Sign out text** - if entered, link with the entered text will be used instead of the default sign out image
- **Show as button** - if checked, buttons will be used instead of links
- **Sign out image** - if set, the image will be used as sign out link

Facebook Connect required data

In some cases, you may want new users to provide some extra details before their account is created. For example, if your site is configured to require registration approval or double opt-in, all users need a valid e-mail address to help activate their account.

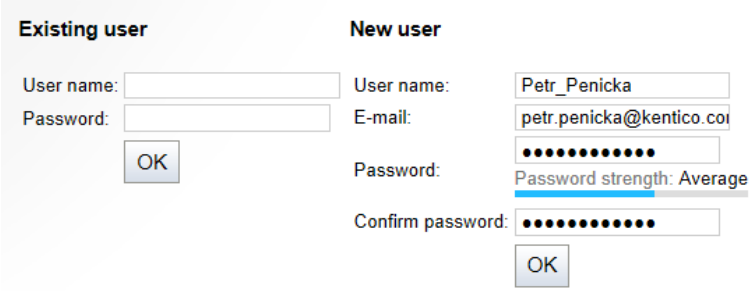
This can be achieved using the *Facebook Connect required data* web part. The web part must be placed on the page specified by the **Required user data page** value in **Site Manager -> Settings -> Social networks -> Facebook**.

The following properties of the web part are the most important:

- **Allow forms authentication** - if checked, new users will be able to enter a password for their new account so that they can log in the usual way as well as via Facebook Connect

- **Allow existing user** - if enabled, users are allowed to join their existing account with their Facebook user ID
- **Default target URL** - if no return URL is passed, users will be redirected to the URL entered here after merging or creating the account
- **Hide for no Facebook user ID** - if checked, the web part will be hidden if the page with it is displayed without Facebook Connect logon (e.g. when accessed by entering its URL into the browser)

Other specific web part properties are explained in Kentico CMS Web Parts Reference or after clicking the help link (?) link in the top right corner of the web part properties window.



The screenshot above shows how the web part looks with both the **Allow forms authentication** and **Allow existing user** options enabled. In the left part, an existing site user can merge their current user account with the Facebook user ID by just entering their user name and password used on your site. In this case, the Facebook user ID will be added to the **Facebook user ID** field of the existing user account. New users can enter the required details in the right part of the web part.

7.13.14.5 LinkedIn

7.13.14.5.1 Overview

LinkedIn is a business-related social networking website. By integrating LinkedIn authentication into your website, you can let users log on to your website using their LinkedIn user name and password.

How to start using LinkedIn authentication

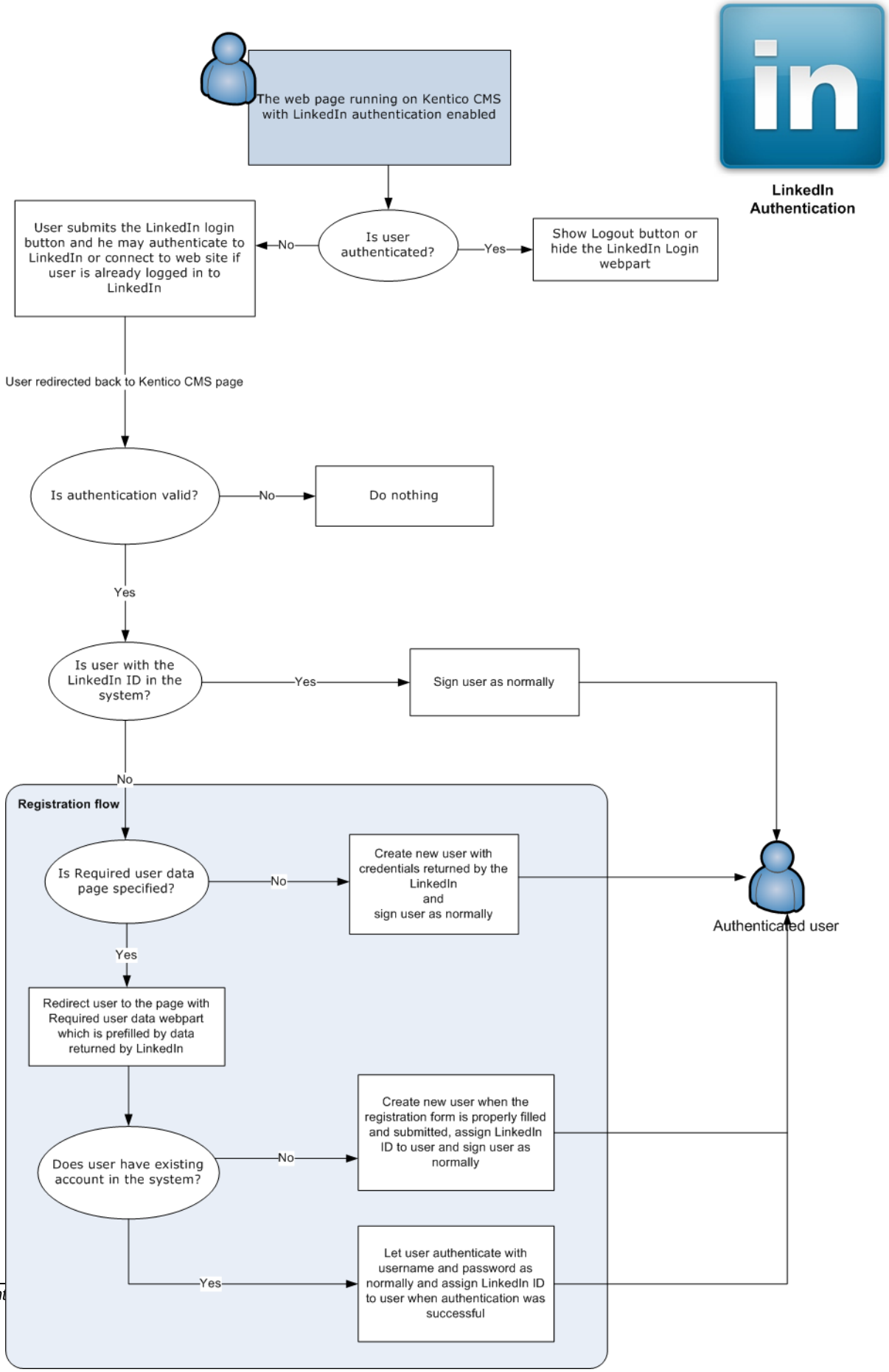
1. Register your application at https://www.linkedin.com/secure/developer - see the Register your application topic.

2. Set up Kentico CMS for LinkedIn authentication - see the Settings topic.

3. Use one of the LinkedIn web parts on any page of your site - see the Available web parts topic.

How it works

The following diagram shows how the process of LinkeIn authentication works:

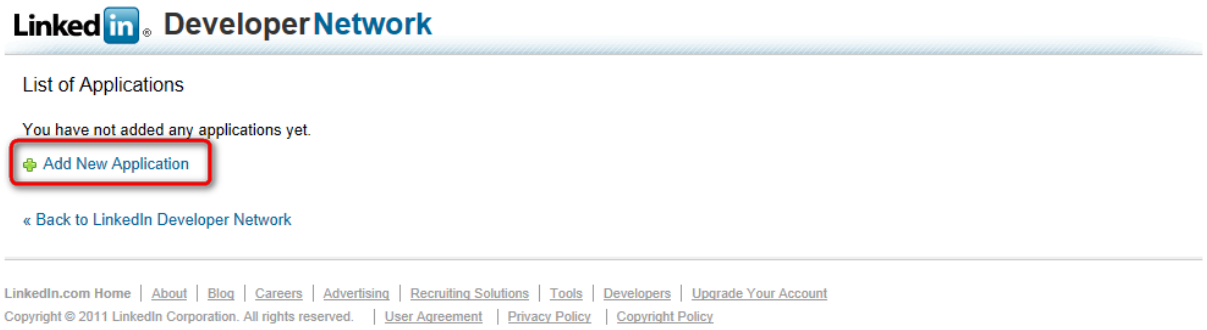

LinkedIn Authentication
The web page running on Kentico CMS with LinkedIn authentication enabled
Is user authenticated?
No
Yes
User submits the LinkedIn login button and he may authenticate to LinkedIn or connect to web site if user is already logged in to LinkedIn
Show Logout button or hide the LinkedIn Login webpart
User redirected back to Kentico CMS page
Is authentication valid?
No
Do nothing
Yes
Is user with the LinkedIn ID in the system?
Yes
Sign user as normally
No
Registration flow
Is Required user data page specified?
No
Create new user with credentials returned by the LinkedIn and sign user as normally
Yes
Redirect user to the page with Required user data webpart which is prefilled by data returned by LinkedIn
Does user have existing account in the system?
No
Create new user when the registration form is properly filled and submitted, assign LinkedIn ID to user and sign user as normally
Yes
Let user authenticate with username and password as normally and assign LinkedIn ID to user when authentication was successful
Authenticated user

7.13.14.5.2 Registering your application

To use LinkedIn authentication on your website, you first need to register your application at LinkedIn and generate an API key for it. To do that, go through the following steps:

1. Go to https://www.linkedin.com/secure/developer and log on using your LinkedIn logon credentials. You need to have a LinkedIn account in order to do this, so if you do not have one, please create it first by clicking the **Join LinkedIn** link in the logon form.

2. Once logged in, you will be redirected to the list of your registered applications. Click the **Add New Application** link.



3. The **Add New Application** dialog will be displayed. Fill in at least the required fields (marked with red asterisks), agree to the terms of use and finally click the **Add Application** button.

LinkedIn Developer Network

Add New Application

Fill out the form to register a new application:

Company Info

* Company Name: My Company

Account Administrators: You will be assigned as an account administrator.

Additional Administrators:

Start typing the name of a connection

Administrators appearing here will be account administrators for all applications from this company. Administrators can edit application details and add/remove other administrators and developers.

Application Info

* Application Name: Kentico CMS Auth Test

* Description: This is a testing application

Integration URL:

Example URL where the integration will go live.

App Logo Secure URL:

URL of an 80x80 logo for your app. SSL is required now.

JavaScript API Domain:

Fully-qualified domain name of all pages that will call the JavaScript API with this key (required if using JS API).

* Application Type: Web application

If your application qualifies as more than one type, create a new application for each one.

* Application Use: Other

What best describes your application?

* Live Status: Development

While in development, your network updates will only go to the developers you choose. When live, they will go to your connections.

Application Developers: Start typing the name of a connection

Network updates you send will appear only for developers you list.

☑ Include yourself as a developer for this application

Interface Language:

- ☐ English
- ☐ French
- ☐ German
- ☐ Italian
- ☐ Portuguese

Enter all the languages for your application interface. LinkedIn uses this information to better provide you support.

Programming Tools:

- ☐ PHP
- ☐ C++
- ☐ C#
- ☐ Java
- ☐ ColdFusion

Enter all the development languages and tools you use. LinkedIn uses this information to provide you better support.

Contact Info

* Developer Contact Email: petr.penicka@kentico.com

* Phone: [redacted]

Support Contact Email:

Phone:

Business Contact Email:

Phone:

OAuth User Agreement

OAuth Redirect URL:

URL to return users to your app after they grant access. Only used if you do not pass in a redirect URL when you send the user to grant access.

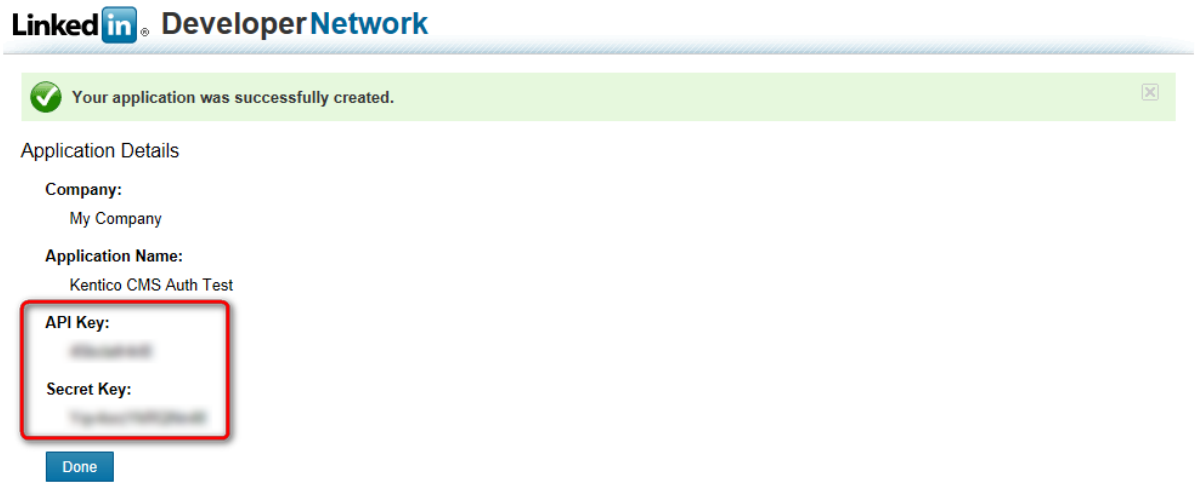
* Agreement Language: English

Select the display language of the user agreement screen. Browser Locale Setting is recommended.

Term of Service

*Agree: ☐ I have read and agree to the LinkedIn API Terms of Use

4. If you specified all required details correctly, you will be redirected to a confirmation page. On this page, you can find the **API Key** and **Secret Key** values. These values need to be entered in setting of Kentico CMS in order for the LinkedIn authentication to be functional. See the Settings topic for more details.



7.13.14.5.3 Settings

First of all, you need to navigate to the **~/Bin/DotNetOpenAuth.dll.rename** file and rename it to **DotNetOpenAuth.dll**. The library is renamed in the default installation because it doesn't support medium-trust environment. Then you need to configure LinkedIn settings in the administration interface of the CMS.

LinkedIn authentication settings are located in **Site Manager -> Settings -> Social networks -> LinkedIn**. Before you start configuring the settings, make sure you have the right site selected using the **Site** drop-down list at the top left part of the page.

General

- **API key** - key obtained during registration of your application at LinkedIn.
- **Application secret -** key obtained during registration of your application at LinkedIn.

The following setting is not required for LinkedIn authentication. It is used mainly for certain LinkedIn web parts.

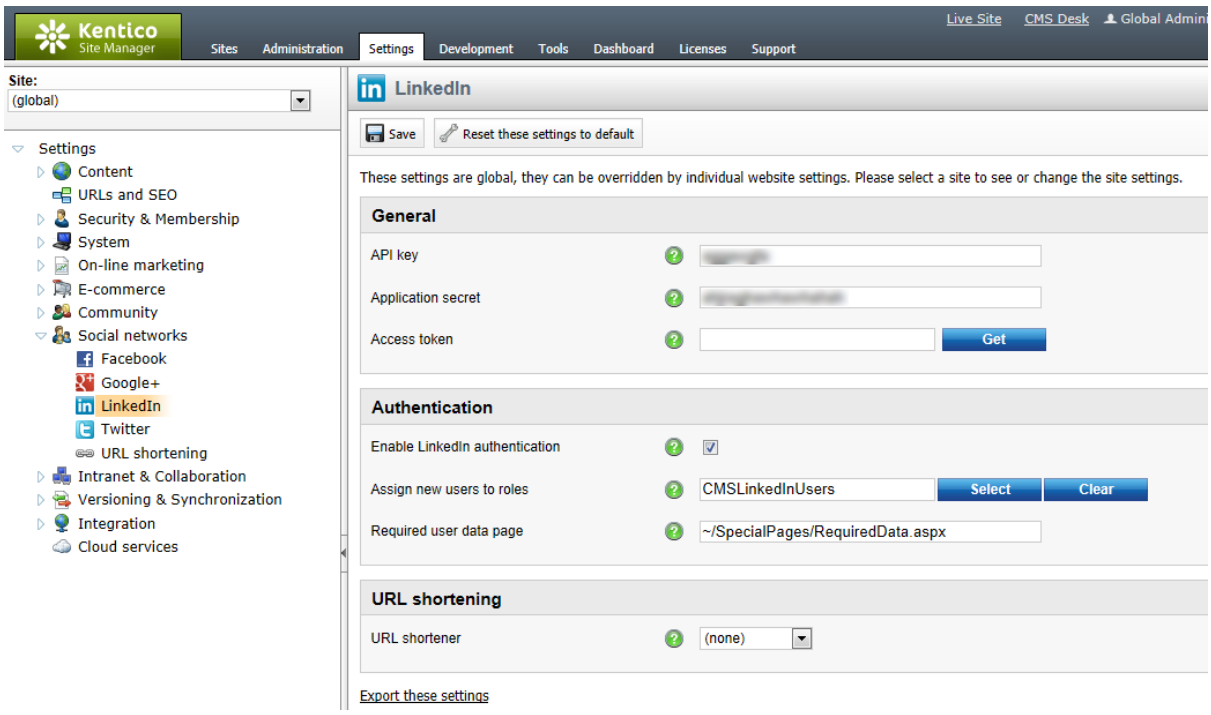
- **Access token** - Sets the site-wide access token required to execute LinkedIn API calls that require authorization.

Authentication

- **Enable LinkedIn authentication** - indicates if LinkedIn authentication is enabled.
- **Assign new users to roles** - new users registered via LinkedIn authentication will be assigned to these roles.
- **Required user data page** - URL of the page where the *LinkedIn required data* web part resides. If entered and a new LinkedIn user logs in to the site for the first time, their user account is not created automatically, but they are redirected to this page and required to enter some additional data (or merge with an existing account) using the web part.

URL shortening

- **URL shortener** - May be used to select a URL shortening service that will be used for all links in LinkedIn posts.

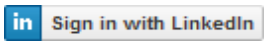


7.13.14.5.4 Available web parts

When you have registered your application as described in the Registering your application topic and after configuration of all required Settings, you may add the following web parts to pages of your website in order to enable users authenticate using their LinkedIn logon credentials.

LinkedIn logon

This web part can be used to let site visitors sign in to your website using their LinkedIn logon credentials. It can be placed on any page of your website. The web part is hidden to authenticated users and will be displayed only to unauthenticated public site visitors. With default settings, the web part appears as in the following screenshot.



Although the web part works fine with default settings, it has the following properties to fine-tune its behavior:

- **Sign in text** - if entered, a link with the entered text will be used instead of the default sign in image.
- **Sign out text** - if entered, a link with the entered text will be used instead of the default sign out image.
- **Show sign out** - if enabled, the web part will display a sign out link to authenticated users.
- **Show as button** - if enabled, buttons will be used instead of standard text links.
- **Sign in image** - specifies the URL of an image that will be used for the sign in button if the *Sign in text* property is empty.
- **Sign out image** - specifies the URL of an image that will be used for the sign out button if the *Sign*

out text property is empty.

- **Required data for new users** - using these settings, you can request additional data from the LinkedIn user account, which will be added to the newly created user account. In case that you are using the *LinkedIn required data* web part, the requested data will be pre-filled in the web part.

- **Notify administrator about new registrations** - if enabled, a notification e-mail will be sent to the website administrator when a new registration is performed via the web part.

LinkedIn required data

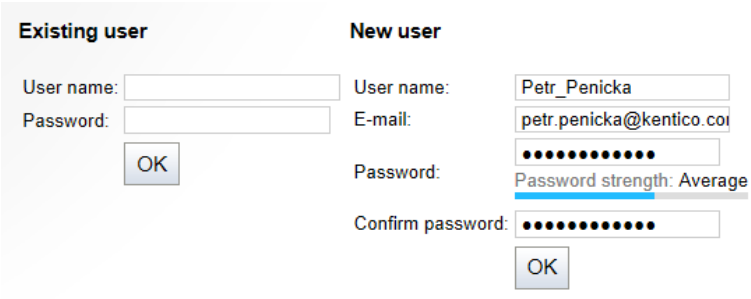
In some cases, you may want new users to provide some extra details before their account is created. For example, if your site is configured to require registration approval or double opt-in, all users need a valid e-mail address to help activate their account.

This can be achieved using the *LinkedIn required data* web part. The web part must be placed on the page specified by the **Required user data page** value in **Site Manager -> Settings -> Social networks -> LinkedIn**.

The following properties of the web part are the most important:

- **Allow forms authentication** - if checked, new users will be able to enter a password for their new account so that they can log in the usual way as well as via their LinkedIn logon credentials.
- **Allow existing user** - if enabled, users are allowed to join their existing accounts with the LinkedIn accounts.
- **Default target URL** - if no return URL is passed, users will be redirected to the URL entered here after merging or creating the account.
- **Hide for no LinkedIn** - if checked, the web part will be hidden if the page with it is displayed without LinkedIn logon (e.g. when accessed by entering its URL into the browser).

Other specific web part properties are explained in Kentico CMS Web Parts Reference or after clicking the help link (?) link in the top right corner of the web part properties window.



The screenshot above shows how the web part looks with both the **Allow forms authentication** and **Allow existing user** options enabled. In the left part, an existing site users can merge their current user account with the LinkedIn account by just entering their user name and password. In this case, their LinkedIn ID will be added to the LinkedIn ID field of the existing user account. New users can enter the required details in the right part of the web part.

7.13.14.6 Managing imported users

When a user signs in through a third-party authentication service for the first time, Kentico CMS automatically creates a new user account for them. If you edit such an account in **Site Manager -> Administration -> Users**, you can see that it has the following specific settings:

General tab

On the General tab, you can notice that the **User name** and **Full name** fields have been automatically filled with a string in the following format:

User name

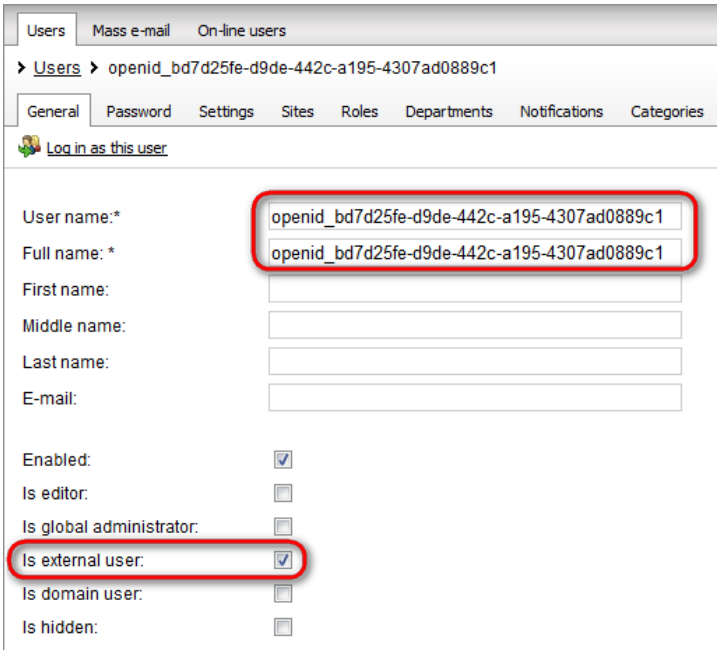
- **Live ID**: *liveid_<liveidtoken>*
- **OpenID**: *openid_<openid>*
- **Facebook Connect**: *facebookid_<facebookuserid>*

Full name

- **Live ID**: *LiveID - <liveidtoken>*
- **OpenID**: *OpenID - <openid>*
- **Facebook Connect**: *Facebook ID - <facebookuserid>*

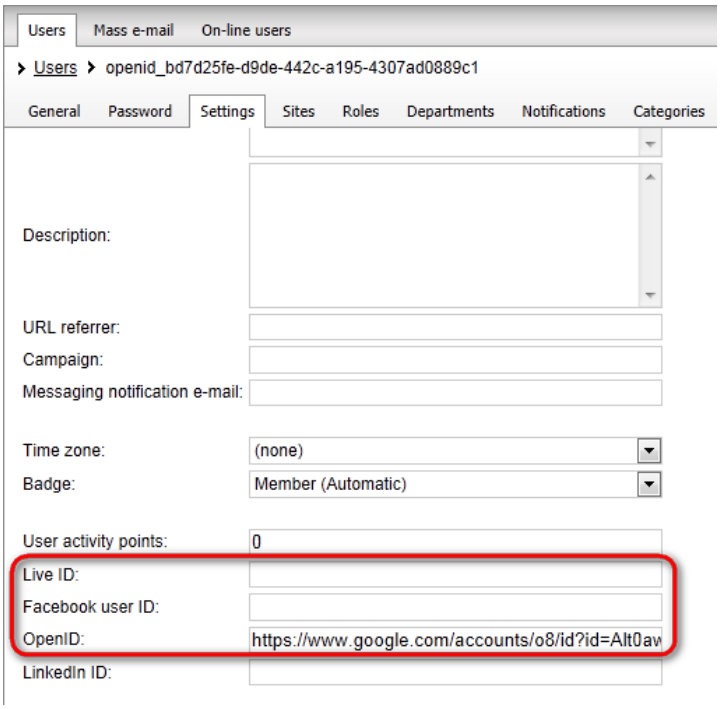
These values can be changed manually without any effects on the functionality.

You can also notice that the **Is external user** check-box is enabled. It indicates that the user account is imported from some external user database and disables forms authentication for the user, i.e. the user can only log in using the third-party authentication service. When merging an existing account with third-party authentication using one of the **<provider> required user data** web parts, the existing account that you want to merge must not have this option enabled (because forms authentication using the web part would not work).



If you switch to the **Settings** tab, you can notice the **Live ID**, **Facebook user ID** and **OpenID** fields. This is where the user's ID from the particular provider is stored.

You can change the values manually if you need to. You just need to make sure that the entered ID is valid. In such a case, the newly entered ID will be used when the user logs in. You can also delete the value, in which case no ID will be assigned to the user. In this case, please remember to disable the **Is external user** option on the **General** tab so that the user can log in using forms authentication.



7.13.15 Membership internals and API

7.13.15.1 Overview

In this chapter, you will learn which database tables and API classes are used for the management of website membership (users and roles). You will also see the most common API examples.

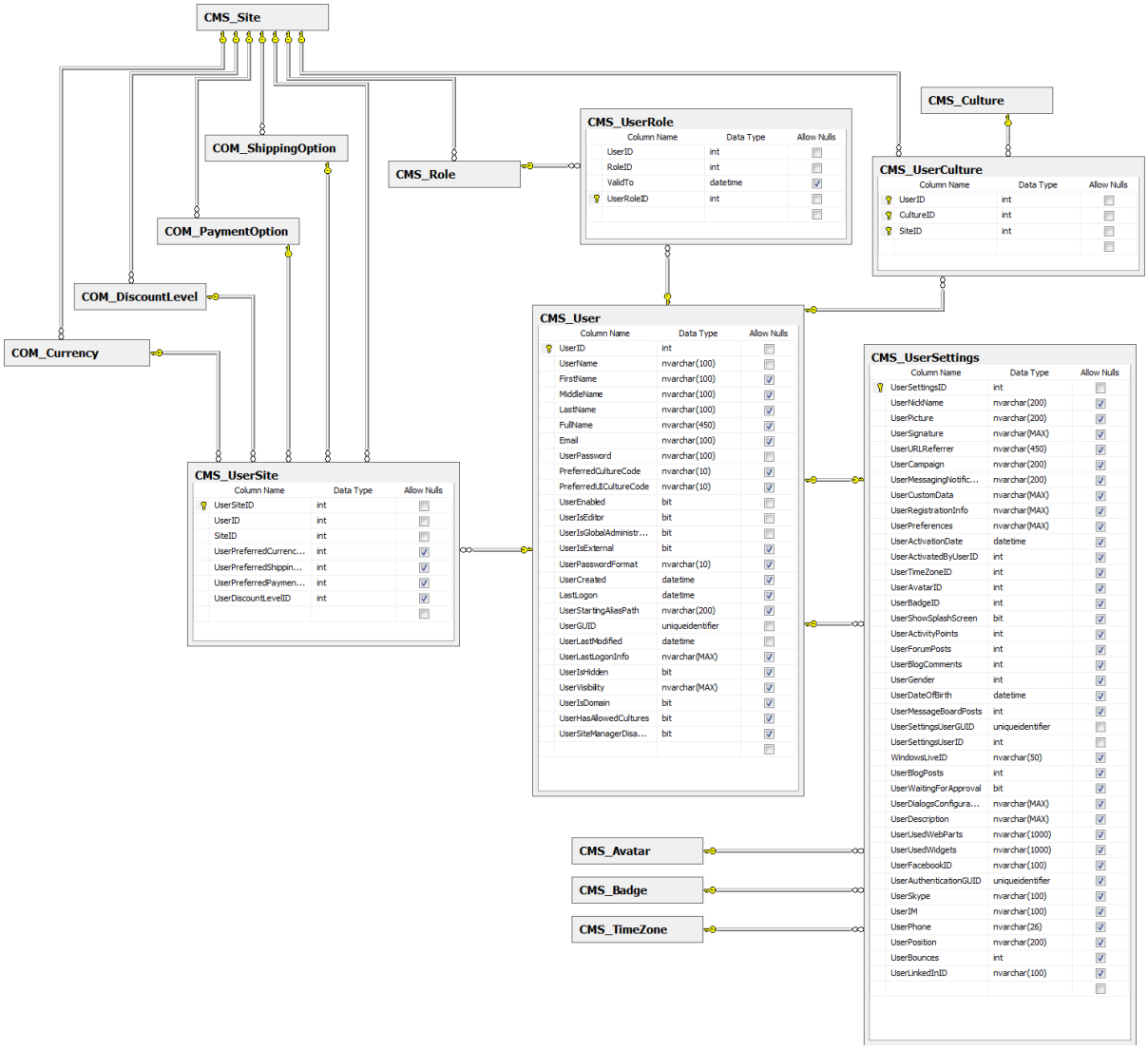
Further API-related information and examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all methods from the module's classes, please refer to Kentico CMS API Reference.

7.13.15.2 Database tables

The following database tables are used to store information about users:

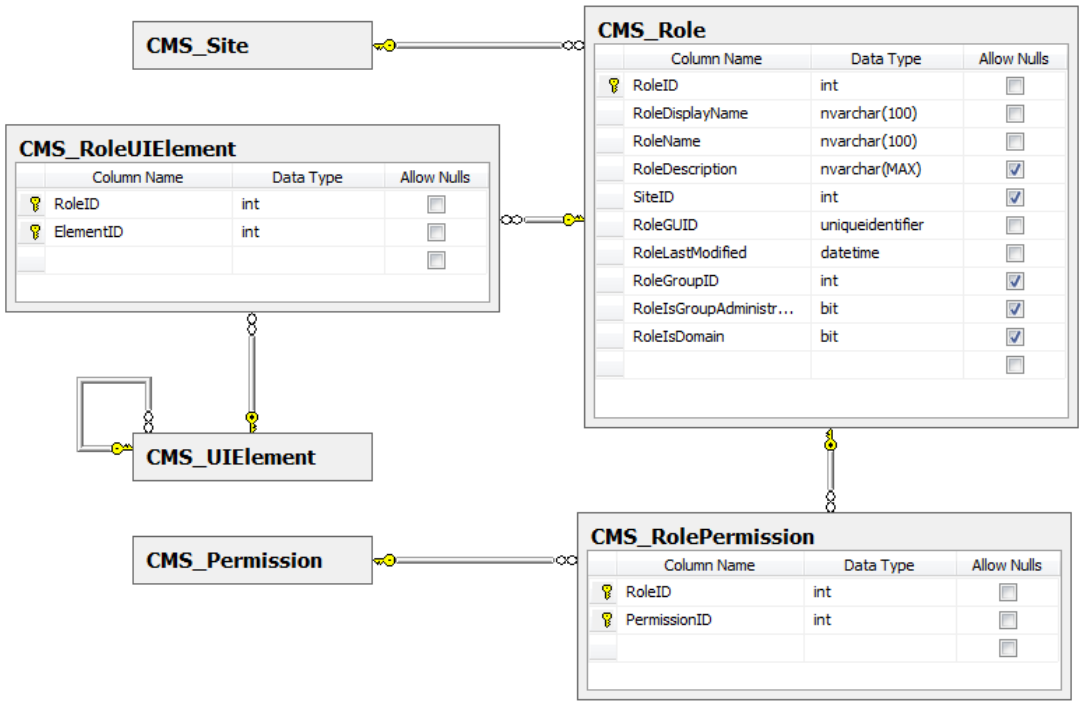
- **CMS_User** - contains records representing users.
- **CMS_UserSite** - stores relationships between users and sites. Each entry indicates that a specific user account is available on a given site.
- **CMS_UserRole** - stores relationships between users and roles. Each entry indicates that a specific role is assigned to a given user. This table is also used to store the expiration date for roles that are assigned for a limited time period.

- **CMS_UserCulture** - stores relationships between users and content cultures. Each entry indicates that a user may edit the content of documents belonging to a specified culture/language (if the given user is an editor).
- **CMS_UserSettings** - stores the advanced settings of users.



The following tables store information about roles and permissions:

- **CMS_Role** - contains records representing roles.
- **CMS_Permission** - stores permissions.
- **CMS_RolePermission** - stores relationships between roles and permissions. Each entry indicates that a permission is granted to a given role.
- **CMS_RoleUIElement** - stores relationships between roles and UI elements. Each entry indicates that a specified UI element will be displayed to members of a given role.



7.13.15.3 API classes

If you are not familiar with database table data management through Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

The classes used for membership management can be found in the **CMS.SiteProvider** namespace.

CMS_User table API:

- **UserInfo** - represents one user object.
- **UserInfoProvider** - provides management functionality for users.

CMS_UserSite table API:

- **UserSiteInfo** - represents a relationship between a user and a site.
- **UserSiteInfoProvider** - provides management functionality for user-site relationships.

CMS_UserRole table API:

- **UserRoleInfo** - represents a relationship between a user and a role.
- **UserRoleInfoProvider** - provides management functionality for user-role relationships.

CMS_UserCulture table API:

- **UserCultureInfo** - represents a relationship between a user and a culture.
- **UserCultureInfoProvider** - provides management functionality for user-culture relationships.

CMS_UserSettings table API:

- **UserSettingsInfo** - represents the advanced settings of a user.
- **UserSettingsInfoProvider** - provides management functionality for user settings.

CMS_Role table API:

- **RoleInfo** - represents one role object.
- **RoleInfoProvider** - provides management functionality for roles.

CMS_Permission table API:

- **PermissionNameInfo** - represents one permission object.
- **PermissionNameInfoProvider** - provides management functionality for permissions.

CMS_RolePermission table API:

- **RolePermissionInfo** - represents a relationship between a role and a permission.
- **RolePermissionInfoProvider** - provides management functionality for role-permission relationships.

CMS_RoleUIElement table API:

- **RoleUIElementInfo** - represents a relationship between a role and a UI element.
- **RoleUIElementInfoProvider** - provides management functionality for relationships between roles and UI elements.

7.13.15.4 API examples

7.13.15.4.1 Overview

These topics show examples of how the API of for membership management can be used:

- Managing users
- Users and sites
- Managing roles
- Roles and user

> **Please note**
>
> All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.
>
> If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Administration\Membership\Default.aspx.cs***.

The membership API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.GlobalHelper;
using CMS.CMSHelper;
using CMS.SiteProvider;
```

7.13.15.4.2 Managing users

The following example creates a user.

```
private void CreateUser()
{
    // Create new user object
    UserInfo newUser = new UserInfo();

    // Set the properties
    newUser.FullName = "My new user";
    newUser.UserName = "MyNewUser";

    // Save the user
    UserInfoProvider.SetUserInfo(newUser);
}
```

The following example gets and updates a user.

```
private bool GetAndUpdateUser()
{
    // Get the user
    UserInfo updateUser = UserInfoProvider.GetUserInfo("MyNewUser");
    if (updateUser != null)
    {
        // Update the properties
        updateUser.FullName = updateUser.FullName.ToLower();

        // Save the changes
        UserInfoProvider.SetUserInfo(updateUser);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates users.

```
private bool GetAndBulkUpdateUsers()
{
    // Prepare the parameters
    string where = "UserName LIKE N'MyNewUser%'";

    // Get the data
    DataSet users = UserInfoProvider.GetUsers(where, null);
    if (!DataHelper.DataSourceIsEmpty(users))
    {
        // Loop through the individual items
        foreach (DataRow userDr in users.Tables[0].Rows)
        {
            // Create object from DataRow
            UserInfo modifyUser = new UserInfo(userDr);

            // Update the properties
            modifyUser.FullName = modifyUser.FullName.ToUpper();

            // Save the changes
            UserInfoProvider.SetUserInfo(modifyUser);
        }

        return true;
    }

    return false;
}
```

The following example deletes a user.

```
private bool DeleteUser()
{
    // Get the user
    UserInfo deleteUser = UserInfoProvider.GetUserInfo("MyNewUser");

    // Delete the user
    UserInfoProvider.DeleteUser(deleteUser);

    return (deleteUser != null);
}
```

The following example checks if the credentials of a user are valid for a specified site using forms authentication.

```
private bool AuthenticateUser()
{
    // Get the user
    UserInfo user = UserInfoProvider.GetUserInfo("MyNewUser");
    if (user != null)
    {
        if (AuthenticationHelper.AuthenticateUser("MyNewUser", "",
```

```
CMSContext.CurrentSiteName) != null)
        {
            return true;
        }
    }

    return false;
}
```

7.13.15.4.3 Users and sites

The following example assigns a user to a site.

```
private bool AddUserToSite()
{
    // Get the user
    UserInfo user = UserInfoProvider.GetUserInfo("MyNewUser");
    if (user != null)
    {
        int userId = user.UserID;
        int siteId = CMSContext.CurrentSiteID;

        // Save the binding
        UserSiteInfoProvider.AddUserToSite(userId, siteId);

        return true;
    }

    return false;
}
```

The following example removes a user from a site.

```
private bool RemoveUserFromSite()
{
    // Get the user
    UserInfo removeUser = UserInfoProvider.GetUserInfo("MyNewUser");
    if (removeUser != null)
    {
        int siteId = CMSContext.CurrentSiteID;

        // Get the binding
        UserSiteInfo userSite = UserSiteInfoProvider.GetUserSiteInfo
(removeUser.UserID, siteId);

        // Delete the binding
        UserSiteInfoProvider.DeleteUserSiteInfo(userSite);

        return true;
    }
```

```
    return false;
}
```

7.13.15.4.4 Managing roles

The following example creates a role.

```
private void CreateRole()
{
    // Create new role object
    RoleInfo newRole = new RoleInfo();

    // Set the properties
    newRole.DisplayName = "My new role";
    newRole.RoleName = "MyNewRole";
    newRole.SiteID = CMSContext.CurrentSiteID;

    // Save the role
    RoleInfoProvider.SetRoleInfo(newRole);
}
```

The following example gets and updates a role.

```
private bool GetAndUpdateRole()
{
    // Get the role
    RoleInfo updateRole = RoleInfoProvider.GetRoleInfo("MyNewRole",
CMSContext.CurrentSiteID);
    if (updateRole != null)
    {
        // Update the properties
        updateRole.DisplayName = updateRole.DisplayName.ToLower();

        // Save the changes
        RoleInfoProvider.SetRoleInfo(updateRole);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates roles.

```
private bool GetAndBulkUpdateRoles()
{
    // Prepare the parameters
```

```
    string where = "RoleName LIKE N'MyNewRole%'";

    // Get the data
    DataSet roles = RoleInfoProvider.GetRoles(where, null);
    if (!DataHelper.DataSourceIsEmpty(roles))
    {
        // Loop through the individual items
        foreach (DataRow roleDr in roles.Tables[0].Rows)
        {
            // Create object from DataRow
            RoleInfo modifyRole = new RoleInfo(roleDr);

            // Update the properties
            modifyRole.DisplayName = modifyRole.DisplayName.ToUpper();

            // Save the changes
            RoleInfoProvider.SetRoleInfo(modifyRole);
        }

        return true;
    }

    return false;
}
```

The following example deletes a role.

```
private bool DeleteRole()
{
    // Get the role
    RoleInfo deleteRole = RoleInfoProvider.GetRoleInfo("MyNewRole",
CMSContext.CurrentSiteID);

    // Delete the role
    RoleInfoProvider.DeleteRoleInfo(deleteRole);

    return (deleteRole != null);
}
```

7.13.15.4.5 Roles and users

The following example assigns a role to a user.

```
private bool CreateUserRole()
{
    // Get role and user objects
    RoleInfo role = RoleInfoProvider.GetRoleInfo("MyNewRole",
CMSContext.CurrentSiteID);
    UserInfo user = UserInfoProvider.GetUserInfo("MyNewUser");
```

```
    if ((role != null) && (user != null))
    {
        // Create new user role object
        UserRoleInfo userRole = new UserRoleInfo();

        // Set the properties
        userRole.UserID = user.UserID;
        userRole.RoleID = role.RoleID;

        // Save the user role
        UserRoleInfoProvider.SetUserRoleInfo(userRole);

        return true;
    }

    return false;
}
```

The following example removes a role from a user.

```
private bool DeleteUserRole()
{
    // Get role and user objects
    RoleInfo role = RoleInfoProvider.GetRoleInfo("MyNewRole",
CMSContext.CurrentSiteID);
    UserInfo user = UserInfoProvider.GetUserInfo("MyNewUser");

    if ((role != null) && (user != null))
    {
        // Get the user role
        UserRoleInfo deleteRole = UserRoleInfoProvider.GetUserRoleInfo
(user.UserID, role.RoleID);

        // Delete the user role
        UserRoleInfoProvider.DeleteUserRoleInfo(deleteRole);

        return true;
    }

    return false;
}
```

7.14 Mobile development

The mobile development features available in Kentico CMS allow you to craft the design and contents of pages to be easily viewable on various types of mobile devices. By grouping devices with similar properties into device profiles and creating page layouts specifically designed for the device profiles, you can provide visitors with websites that bring an optimal browsing experience and require a minimum amount of scrolling and zooming.

Topics:

7.14.1 Enabling mobile development features

Enabling device profiles

Device profiles in Kentico CMS allow you to customize the layout and content of pages according to the specific devices that visitors use to view the pages. To start using device profiles, enable them in the settings first.

1. Open **Site Manager** and navigate to **Settings -> Content -> Content management**.

2. (Optional) Select a site where you want to enable device profiles.

3. In the **Mobile development** section, check the **Enable device profiles** setting.

4. Click **Save**.

With device profiles enabled, the system evaluates whether the device that is requesting a page belongs to one of the device profiles defined in **Site Manager -> Development -> Device profiles**. See Creating device profiles for details.

Setting up device detection

By default, Kentico CMS only detects a limited set of mobile devices. The basic detection uses XML data stored in the *~/App_Data/CMSModules/DeviceProfile/devices.xml* file, which contains information about specific devices.

Each device has a set of properties describing its capabilities and a list of user agents used for identification. For example:

```
<Device name="Nokia_710" displayname="Nokia - 710" width="480" height="800"
mobile="true">
  <UserAgent>
    Mozilla/5.0 (compatible; MSIE 9.0; Windows Phone OS 7.5; Trident/5.0;
IEMobile/9.0; NOKIA; Lumia 710)
  </UserAgent>
  <UserAgent>
```

```
    Mozilla/5.0 (compatible; MSIE 9.0; Windows Phone OS 7.5; Trident/5.0;
IEMobile/9.0; NOKIA; 710)
  </UserAgent>
</Device>
```

You can manually expand the data by adding devices and their user agents. The system reloads the data every time the application starts, so you need to restart the application for changes to take effect.

When creating device profiles, you can assign the devices from the XML file using the device selector.

Using 51degrees.mobi device data

The 51degrees.mobi mobile device detection software integrated into Kentico CMS allows you to extend the detection data to include all devices that are available on the market. The data also provides access to all device properties (listed in the device macro reference).

To use this data, you need to obtain a license from 51degrees.mobi and register it within Kentico CMS.

1. Buy a license for Premium Data at 51degrees.mobi. You will receive either a license key that enables you to download device data automatically, or a data file that you need to manually upload into the system.

2. Go to **Site Manager -> Settings -> Integration -> 51degrees.mobi**.

3. Enter your 51degrees.mobi license key into the text box and click **Activate**, or select a data file from your disk and click **Upload**.

4. Click **Save**.

The system now has access to the 51degrees.mobi mobile device data set. You can select devices directly from the device selector when creating device profiles and you can access additional device properties within the CurrentDevice object.

7.14.2 Creating device profiles

Device profiles in Kentico CMS allow you to customize the layout and content of pages according to the parameters of the devices that visitors use to view the pages. Before you start using device profiles, enable them in the settings first.

1. Open **Site Manager** and navigate to **Development -> Device profiles**.

2. Click **New profile**.

3. Type the **Display name** of the device profile.

4. Fill in the parameters that define which devices belong to the profile. You can combine these parameters as needed.

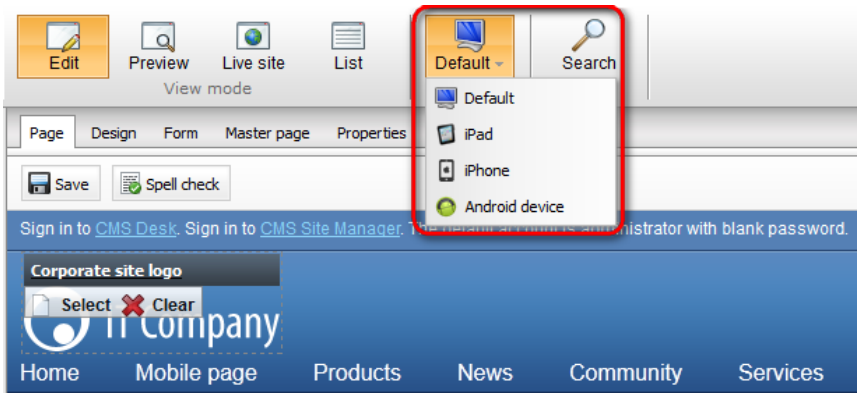
- **Devices** - select devices directly from a list.
- **User agents** - allows you to identify devices based on user agents. The profile includes devices that contain the entered text in their user agent string. You can add multiple entries, each on a new line.

- **Macro** - use the macro rule designer to specify the condition that devices must meet to fit the device profile, or type in any macro condition. You can use the *CurrentDevice* object to retrieve data of the current device. See Device macro reference and Macro expressions documentation for more details.

5. (Optional) Set the **Preview width** and **Preview height**. The preview dimensions of the device profile determine the following:
   - The maximum size of images if automatic image resizing for devices is enabled.
   - The size of the displaying area when previewing pages in CMS Desk.

6. Click **Save**.

Once you create a device profile, it appears in the device selector in CMS Desk.



You can then select a device profile using the selector and start Creating page layouts for devices, which will be used on individual pages instead of the default layouts if the current device fits the profile. Or you can Map layouts that will replace shared layouts when the current device fits the profile.

7.14.3 Resizing images for devices

You can configure the system to automatically resize images to match the dimensions of device profiles:

1. Go to **Site Manager -> Settings -> Content -> Content management**.
2. Enable the **Resize images according to device profile** setting.
3. Click **Save**.

When a user views the website on a device that matches a certain profile, images automatically reduce their maximum side size to the larger of the dimensions set for the given profile.

To change the dimensions of device profiles:

1. Go to **Site Manager -> Development -> Device profiles**.
2. Edit () a profile.
3. Set the **Preview width** and **Preview height** properties (the values are in pixels).
4. Click **Save**.

Note

The resizing only works for images loaded through one of the Kentico CMS system pages (getattachment, getmetafile, getmedia, etc.). Device profile resizing does not work for images that use web links or direct paths in their source.

See also:

- Linking pages and files
- GetFile.aspx parameters

Disabling the resizing for individual images

If you wish to disable device profile resizing for an image, add the **resizemode=1** query string parameter to the image source URL.

For example:

```
<img src="~/getattachment/Community/Blogs/LargeImage.png.aspx?
width=800&resizemode=1">
```

7.14.4 Creating page layouts for devices

Various devices have different capabilities and screen resolutions, which makes it difficult to build a single page layout suitable for all devices. Individual page templates allow you to define alternative page layouts for specific device profiles. Pages based on these templates automatically use the appropriate layout according to the device profile detected for each visitor.

To create a dedicated device layout for a page:

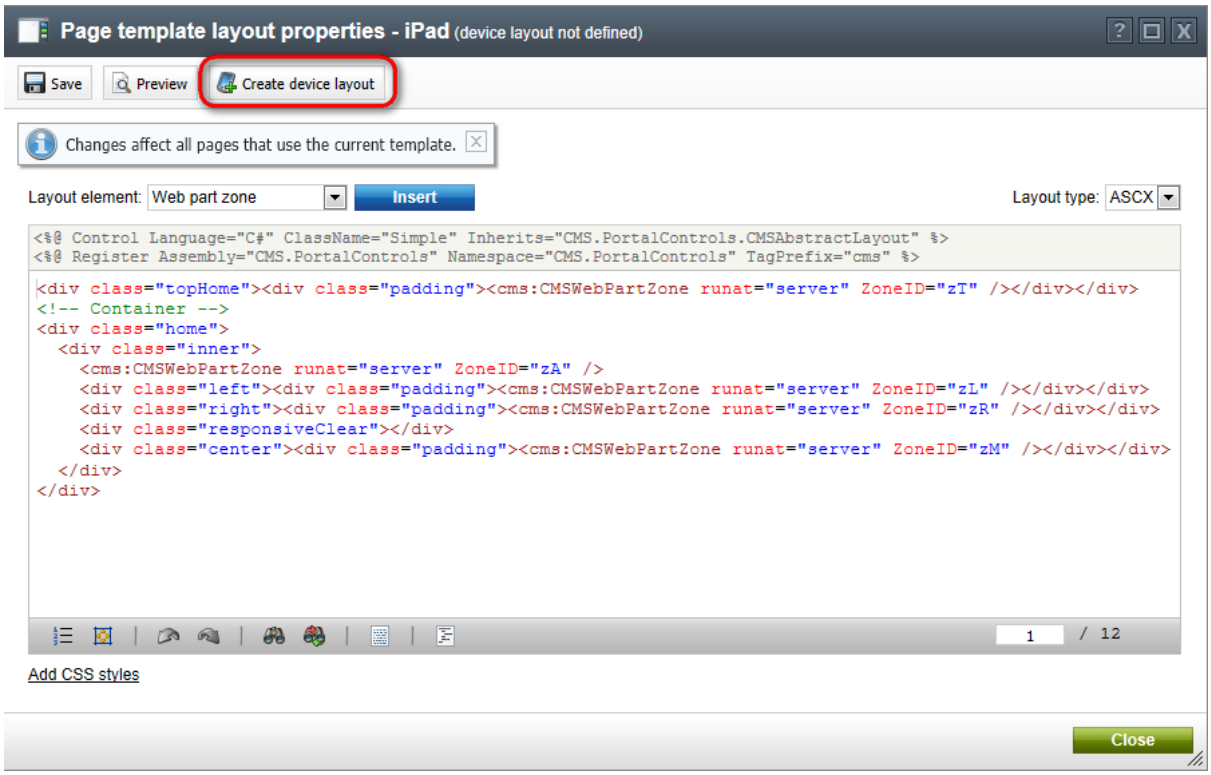
1. Open **CMS Desk** and select the document representing the page in the content tree.

2. Select the **Design** tab.

3. Switch to the device profile for which you intend to create the layout using the selector on the main CMS Desk toolbar.

4. Click **Edit layout** ( ) on the header of the page's template area. The template's page layout editing dialog opens.

5. Click **Create device layout**.



The **New device layout** dialog opens.

6. Select the source for the new device layout's initial content:

- **Copy from device profile** - copies the layout code from the template's default page layout or from one of its other device profile layouts.
- **Use existing layout** - allows you to select an existing shared page layout.
  - If you leave the **Copy as custom** box checked, the system creates the device layout as a separate copy of the selected layout. Otherwise the page template directly uses the shared layout for the device profile (changes made to the layout code affect all pages based on the same shared layout).
- **Use empty layout** - creates a new custom layout for the device profile with a single web part zone and no other formatting.

Click **OK**.

7. Define the code of the page layout for the given device profile.

Adding web part zones

The page template allows you to share web part content between the default page layout and device layouts. To implement this scenario, use identical ID values (**ZoneID** attributes) for the web part zones in the code of the template's various page layouts.

If you add web part zones with unique zone IDs, the zones will only be available in the specific device layout. When the system renders the page, it only displays the content

of zones that exist in the currently active page layout.

8. (Optional) Click Add CSS styles below the layout's code. The **CSS styles** editor appears, where you can define any device-specific CSS classes used within the layout code. The page loads these styles when a visitor views it on a device that matches the given profile. See also: CSS for page components

9. Click **Save** and **Close** the dialog.

The page now automatically adjusts its layout based on the device profile detected for individual visitors.

You can modify the web part content of the page for different device profiles by using the **Design** tab of CMS Desk in combination with the device selector. See Creating mobile pages for details.

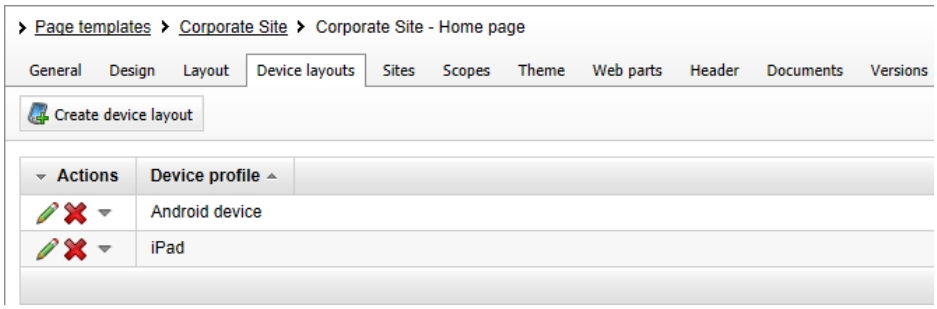
Managing device layouts

You can also manage device layouts through the page template editing interface.

1. Go to **Site Manager -> Development -> Page templates** and select a page template.

2. Open the **Device layouts** tab.

This tab displays a list of the template's page layouts designed for specific device profiles. You can **Create device layouts** and edit () or delete () existing ones.



3. Create or edit a device layout. The following sub-tabs become available:

- **Layout** - allows you to modify the device layout's code.
- **Design** - displays the current page template in Design mode (using the given device layout).
- **Versions** - allows you to view the device layout's version history.

Any changes made on these tabs have the same effect as when editing page layouts directly in CMS Desk.

7.14.5 Mapping shared layouts

Device profiles allow you to replace shared page layouts with other layouts that are customized for displaying on the particular device. For example, if a page uses a three-column layout, it won't fit on a mobile phone's display. Using layout mappings, you can replace the columns with rows and thus reorganize the content of the page to be readable on the particular device.

Preparing shared layouts for mapping

You can only create mappings for layouts that are **convertible**.

1. Go to **Site Manager -> Development -> Page layouts** and edit the layout for which you want to have alternatives for different devices.

2. Check the **Is convertible** box.

3. Enter the **Number of zones** that the layout contains. The system automatically counts the number of web part zones in the layout code, but you can manually override the value (for example in the case of conditional layouts or layouts that load web part zones dynamically).

4. Save the layout.

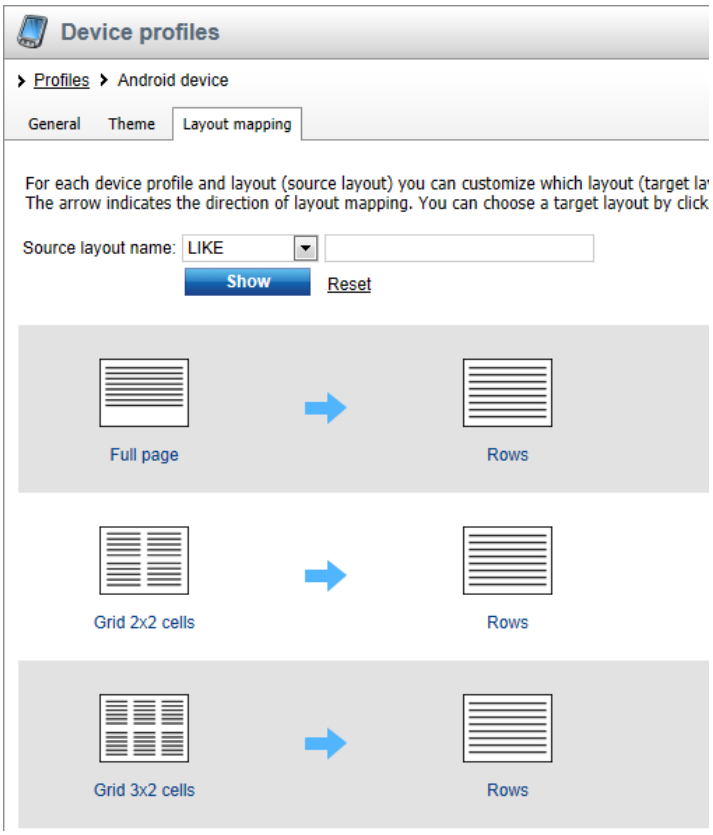
The layout appears in the list of layouts on the Layout mapping tab in the device profile editing interface, where you can map it to a layout that will replace it when the particular device requests a page that uses the layout.

Mapping shared layouts

Prerequisite: Enable the **Enable layout mapping** setting for the website in **Site Manager -> Settings -> Content -> Content management**.

1. Go to **Site Manager -> Development -> Device profiles** and edit the device profile for which you want to map layouts.

2. Switch to the **Layout mapping tab**.



3. Find the layout that you want to create a mapping for on the left side of the grid.

4. Click the question mark on the right side to create a new mapping, or click a mapped layout on the right side to change the mapping. The **Select a target layout** dialog opens.

5. If you want to select a layout that has a different number of web part zones than the original, uncheck **Show only layouts with a matching number of zones**.

6. Click a layout to select it and click **OK** to confirm the selection.

The mapping is complete. If a device requests a page that uses the layout on the left and the device fits the device profile, the page will use the mapped layout instead.

You can now start designing pages for different device profiles using the Design mode in CMS Desk in combination with the device selector. See Creating mobile pages for details.

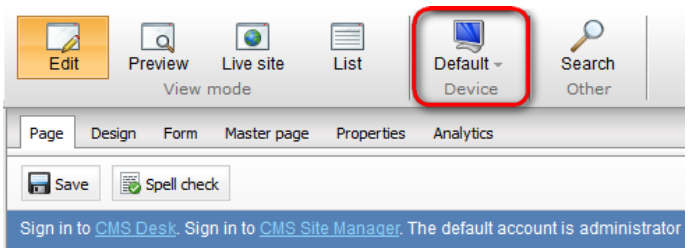
7.14.6 Creating mobile pages

This topic describes tasks that are common when developing mobile pages using Kentico CMS.

Switching device views

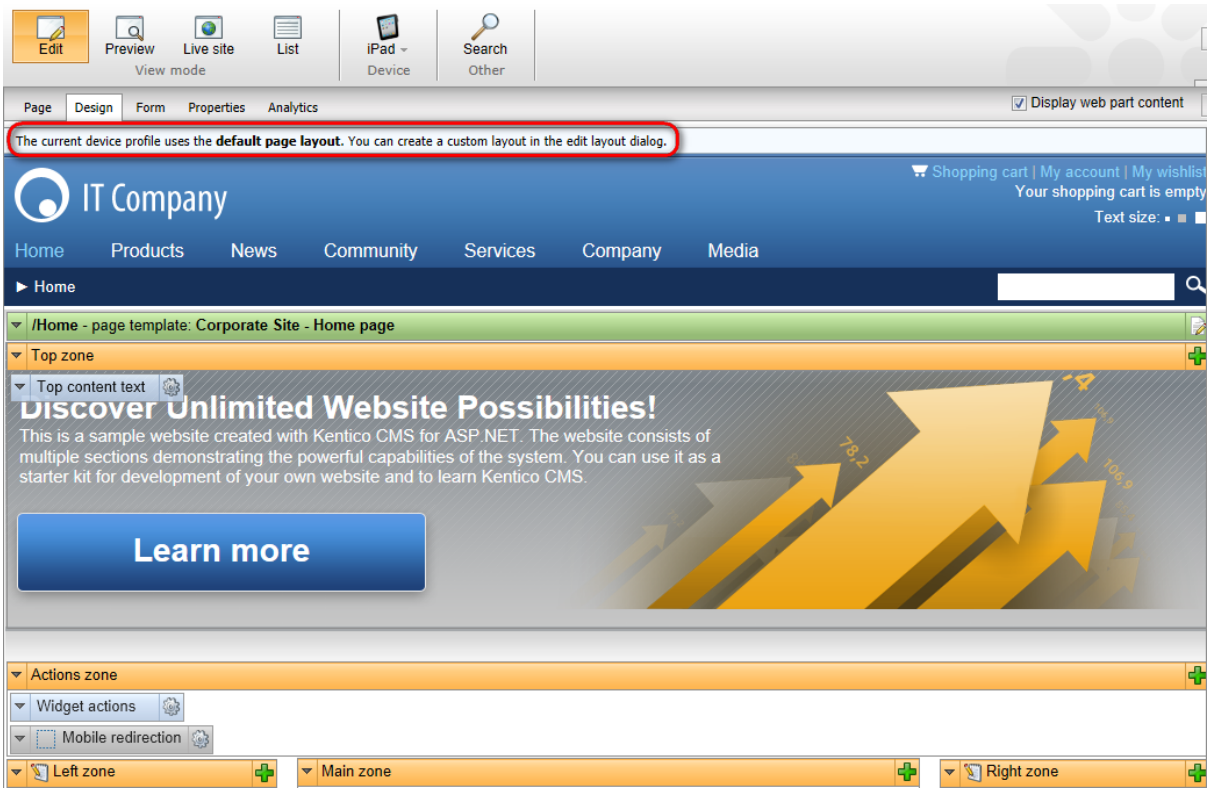
Kentico CMS allows you to develop pages and page templates for a particular device. Once you enable and create device profiles, the device selector appears in CMS Desk and you can switch the context of the Edit mode to a different device.

1. Click the Device selector. A list of enabled device profiles appears.

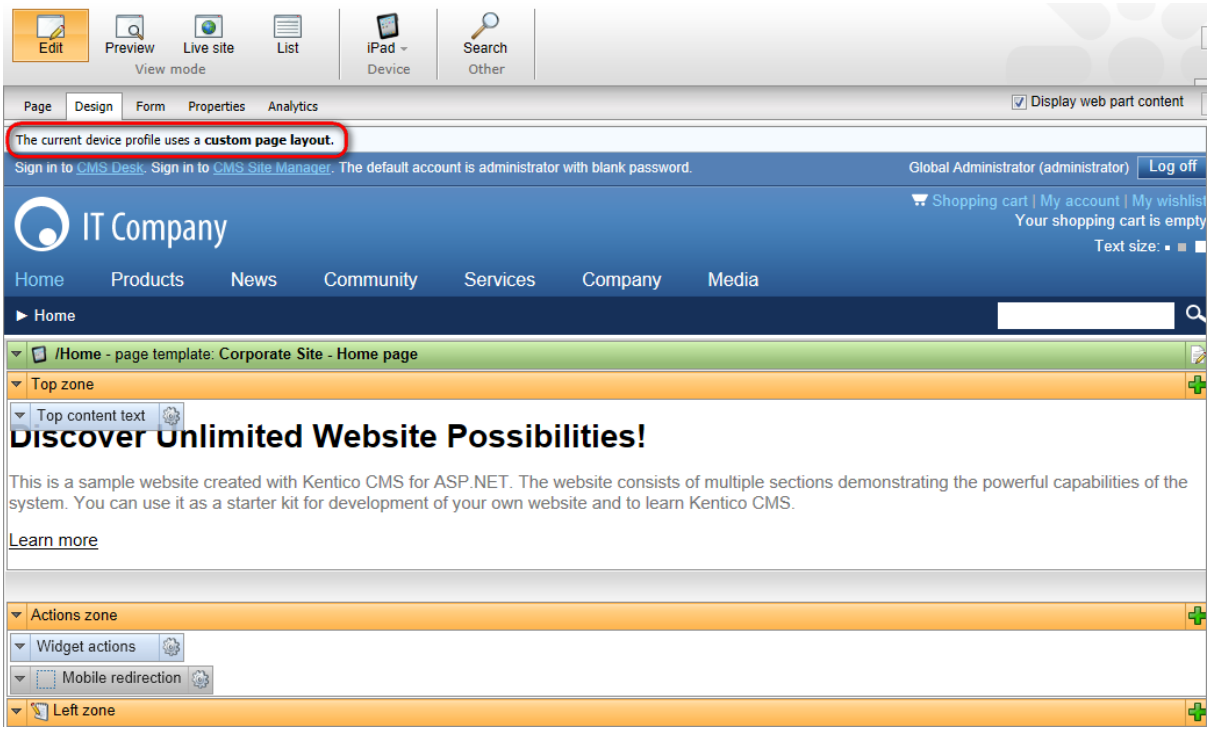


2. Click a device profile.

By default, when you switch to the **Design** tab, you can see and modify the contents of the default page layout for the current page, as in the following image:



If you created a device-specific page layout for the current page template, or if the page uses a shared layout that has a layout mapping defined for the current device profile, you can see the structure of the given device-specific layout.

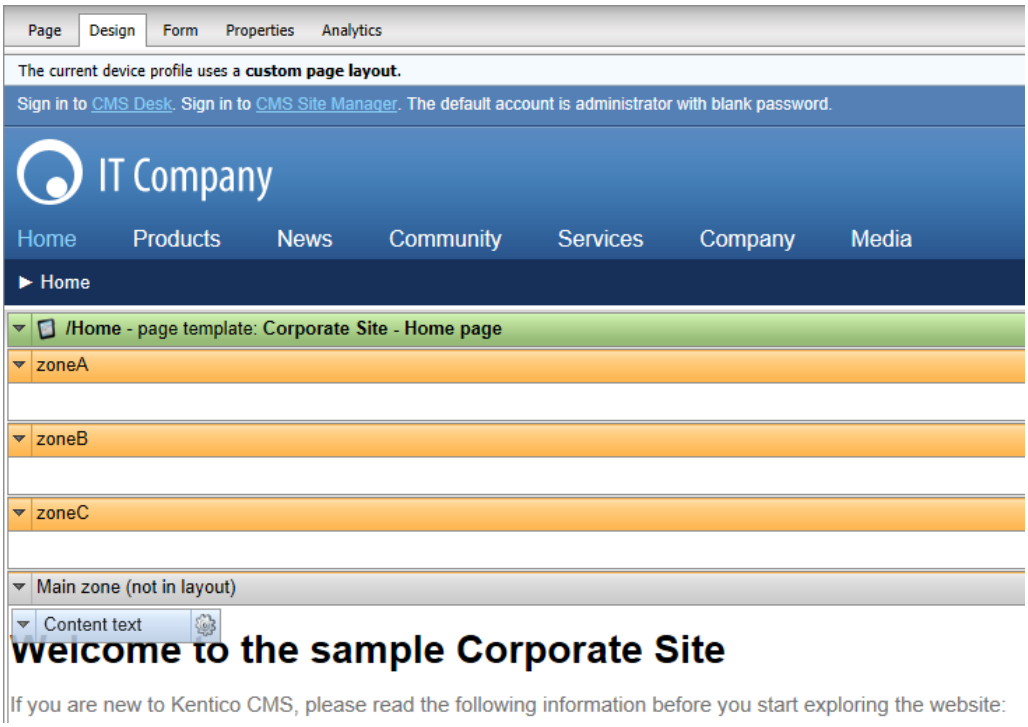


Device layout priority

Custom device layouts always take priority over shared layout mappings. If you defined a custom device layout for a page's template, the page always uses the custom layout even if the template originally uses a shared layout with a mapping for the current device.

Page templates share web part zones between their default page layout and all device layouts. If you modify the content or configuration of a web part zone, the changes also affect the output of the template's other page layouts that contain a zone with the same ID.

You can find all web part zones that do not exist in the current page layout at the bottom of the page with a gray header and the *(not in layout)* suffix.



Design tab with a device layout that uses its own unique web part zones

When the system renders the page on the live site, it only displays the content of zones that exist in the currently active page layout.

Previewing pages on different devices

The Preview and Live site modes in CMS Desk allow you to view pages as they would appear on different devices. Before you can use this, you must enable and create device profiles.

Note

The Preview and Live site mode only imitate the size and appearance of the device. The pages are still rendered with your browser engine.

1. Switch to the device profile that you want to preview.

2. Click Preview or Live site.

The page is shown in a frame that imitates a tablet device. To rotate the device 90 degrees, click **Rotate device preview**.

Customizing the preview frame

Kentico CMS allows you to customize the dimensions and looks of the Preview and Live site modes when using the modes with a device profile selected. This way you can make it look like you were

looking at the actual device with the content displayed on its screen.

The preview consists of nine pieces, eight forming the frame and one center piece, which contains the content of the page.

1. Edit the device profile for which you want to design your custom preview frame.

2. On the general tab, enter the **Preview width** and **Preview height**.

3. Switch to the **Theme** tab and upload files that will represent pieces of the frame. Upload the following pieces and their alternatives for the rotated version of the preview frame:

- top left, top center and top right piece
- center left and center right piece
- bottom left, bottom center and bottom right piece

It is recommended to upload the files into a separate folder.

4. On the Theme tab, click **New file** to create a new stylesheet. Name it *DeviceProfile.*

5. In the stylesheet, specify which image corresponds to which part of the preview frame. Each part of the frame has a CSS class. Use the following pattern to construct a CSS selector that identifies a particular class:

```
.DeviceFrame.<device profile name>[.Rotated] .<vertical direction> .<horizontal
direction>
```

- Add the .Rotated suffix to identify pieces of the device when it is rotated.
- Replace <vertical direction> with one of the following: *TopLine*, *CenterLine*, *BottomLine*
- Replace <horizontal direction> with one of the following: *LeftPiece*, *CenterPiece*, *RightPiece*

The following is an example definition of a preview frame piece.

```
.DeviceFrame.iPad.Rotated .TopLine .LeftPiece
{
    background-image: url('./Images/top_left_rotated.png');
    width: 126px;
    height: 114px;
}
```

Allowing visitors to override the device detection

Some visitors may prefer to view the standard version of the website (i.e. the content of the *Default* device profile) even when using a device that fits into a different device profile. Kentico CMS provides a component that allows users to switch between the content created for the detected device profile and the default version of the website.

1. Select a page in CMS Desk and view it on the **Design** tab. For best results, use a page that visitors can access anywhere on the site, such as the master page.

2. Add the Switch mobile device detection web part onto the page.

3. Click the **Configure** ( ) action of the web part and type in values for the following properties:

- **Mobile link content** - sets the text of the link that the web part displays on the device-specific version of the site. This link allows users to switch to the default version of the website.
- **Desktop link content** - sets the text of the link that returns visitors to the device-specific version of the site.

4. (Optional) Define the *SwitchDeviceDetection* CSS class in the website's stylesheet or directly in the web part's CSS styles. The web part applies this class to the generated links. See also: CSS stylesheets and design

The web part appears only for visitors whose device matches a non-default device profile. If a visitor switches to the default (desktop) content, the web part displays a link that re-enables device detection for the given visitor and returns them to the device-specific version of the site.

Adjusting web part properties for specific device profiles

If you need to change the behavior of web parts according to the current device profile, insert the appropriate macro expressions into the web part properties. This allows you to dynamically adjust pages that use the same content for their default page layout and device layouts.

See also: Device macro reference

Example

The following steps show how to make a web part visible only for the *Android device* profile.

1. Open CMS Desk and select the page containing the web part.

2. Switch to the **Design** tab and click the **Configure** ( ) action of the web part. The **Web part properties** dialog opens.

3. Click the **Visibility** section of the web part properties to expand it.

4. Click **Edit value** ( ) next to the checkbox of the **Visible** property. The **Edit value** dialog opens.

5. Enter the following macro expression:

[K#]

```
{% CurrentDeviceProfileName == "Android_device" %}
```

6. Click **Save** to insert the macro value into the property.

7. Click **OK** to save the configuration of the web part.

The system evaluates the macro expression when rendering the page. If the name of the current device profile matches the text (*Android_device*), the macro returns a true value, which dynamically enables the web part's **Visible** property.

Using device-specific transformations

The transformations that you use to display data have a significant effect on the design of pages. If you need to assign different transformations for specific devices or device profiles, add macro expressions into the transformation properties of the appropriate web parts.

See also: Device macro reference

Example

This example demonstrates how to create and assign a dedicated transformation for displaying news documents on mobile devices.

1. Open the sample Corporate Site in CMS Desk.

2. Select the **News** document and switch to the **Design** tab.

3. Click the **Configure** ( ) action of the *News repeater* web part. The **Web part properties** dialog opens.

4. Scroll down to the **Transformation** property and click **New** next to the textbox. The **New transformation** dialog opens.

5. Type *NewsList_mobile* as the **Transformation name** and enter the following code:

```
<div class="description" style="width:500px;">
  <a class="header bold" href="<%# GetDocumentUrl() %>"><%# Eval("NewsTitle",
true) %></a>
  <p>
    <%# Eval("NewsSummary") %><br /><br />
    <span class="black bold"><%# GetUserFullName(Eval<int>("NodeOwner")) %> | </
span>
    <span class="gray"><%# Eval("NewsReleaseDate") %></span>
  </p>
</div>
```

This is a simplified version of the default *corporatesite.transformations.NewsList* transformation, without teaser images and with a limited text width.

6. Click **Save** and **Close** the dialog.

7. Click **Edit value** ( ) next to the **Transformation** property. The **Edit value** dialog opens.

8. Enter the following macro expression:

[K#]

```
{%
```

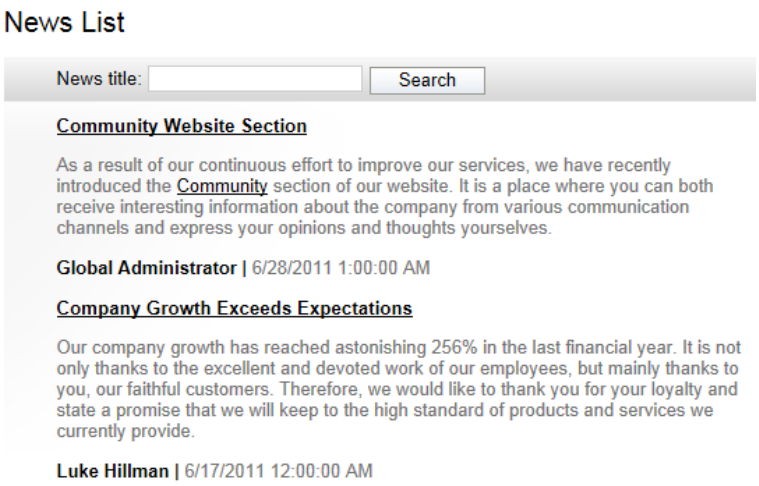
```
if (CurrentDevice.IsMobile AND
 GlobalObjects.DocumentTypes["CorporateSite.Transformations"]
.Children.Transformations.Exists(CodeName == "NewsList_mobile"))
    {"corporatesite.transformations.NewsList_mobile"}
else
    {"corporatesite.transformations.NewsList"}

%}
```

This macro evaluates a condition and returns the transformation name according to the result.

- The *CurrentDevice.IsMobile* part of the condition checks whether the page is being viewed by a mobile device. This directly searches the data available for the current device (the device profile of the current visitor does not affect the result).
- The second condition checks if the *corporatesite.transformations.NewsList_mobile* transformation actually exists in the system. While not necessary in this example, similar conditions allow you to load transformations designed for specific devices. In these cases, you can use a dynamic parameter in the transformation name, such as the exact device name (*CurrentDevice.DeviceName*).

9. Click **Save** to insert the macro value into the property and then **OK** to save the configuration of the web part.

When a visitor views the News page on a mobile device, it automatically displays the simplified list of news items according to the new transformation (*corporatesite.transformations.NewsList_mobile*).



Users with non-mobile devices can still see the original full-sized transformation (*corporatesite.transformations.NewsList*).

7.14.7 Creating a separate mobile section

Kentico CMS allows you to create a dedicated sub-section of the website for visitors with mobile devices.

You can redirect users to the mobile section by placing the Mobile device redirection web part placed onto the website's main landing page. This web part redirects mobile users to one of two URLs

according to the user agent of the detected device.

Device detection

The **Mobile device redirection** web part uses its own device detection logic that is *not* related to the device data or device profiles.

You can configure the behavior of the web part through its properties:

Small device redirection URL	URL to which the web part redirects mobile devices recognized as small. The properties below determine which devices are considered large or small.
Large device redirection URL	URL to which the web part redirects mobile devices recognized as large. The properties below determine which devices are considered large or small.
Redirect Android	Determines if Android mobile devices are considered as small or large devices. If you set the value to *Automatic*, Android devices are recognized as small. Choose *Never* to disable redirection for Android devices.
Redirect BlackBerry	Determines if BlackBerry mobile devices are considered as small or large devices. If you set the value to *Automatic*, BlackBerry devices are recognized as small. Choose *Never* to disable redirection for BlackBerry devices.
Redirect iPad	Determines if iPad devices are considered as small or large devices. If you set the value to *Automatic*, iPad devices are recognized as large. Choose *Never* to disable redirection for iPad devices.
Redirect iPhone	Determines if iPhone devices are considered as small or large devices. If you set the value to *Automatic*, iPhone devices are recognized as small. Choose *Never* to disable redirection for iPhone devices.
Redirect Nokia	Determines if Nokia mobile devices are considered as small or large devices. If you set the value to *Automatic*, Noikia devices are recognized as small. Choose *Never* to disable redirection for Nokia devices.
Always redirect	When the website is accessed from a mobile device, the system stores a redirection cookie in the device's browser. If you disable this property, the web part does not redirect devices if the cookie is already present in the browser, i.e. redirection is only done for the first visit. If enabled, the web part redirects mobile devices on every

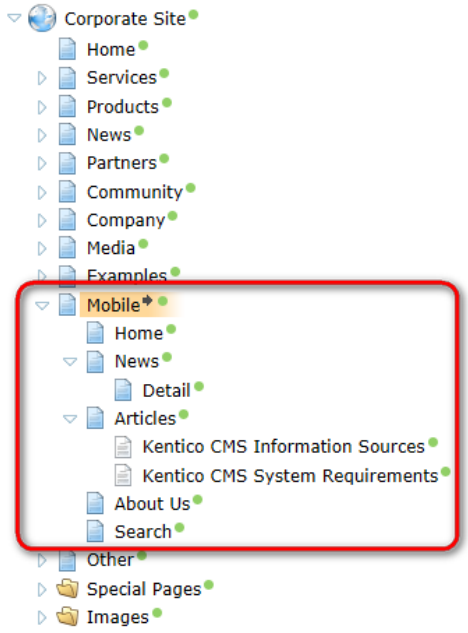
	visit.
Other small devices (User agent)	Specifies a list of additional user agents that the web part recognizes as small mobile devices. Enter each user agent on a separate line.
Other large devices (User agent)	Specifies a list of additional user agents that the web part recognizes as large mobile devices. Enter each user agent on a separate line.

Example

The sample **Corporate Site** contains an example of a mobile website section. The main **Home** page uses a *Mobile device redirection* web part to send mobile visitors to a dedicated website section.

The **Mobile** page serves as the master page of the mobile section. This page does not use template inheritance, so it doesn't display content from the site's main master page. Under it, you can find the following pages:

- **Home** - mobile users get redirected to this page from the site's main *Home* page. The page uses a Repeater web part to dynamically load content from the Editable text region on the website's main *Home* page.
- **News** - uses a Repeater web part to display the *News* documents stored under the website's main */News* section. The news item titles link to the */Mobile/News/Detail* page, which uses a wildcard URL (*/Mobile/News/{id}*) containing the *ID* of the exact news item.
- **Articles** - the Repeater web part on this page displays the articles stored under it. The content is stored directly in the mobile section. It is not shared with the rest of the site.
- **About us** - contains only two Editable text web parts. Its content is separate and used only in the mobile section.
- **Search** - this page is not accessible through navigation. The Smart search box on the *Mobile* master page redirects users here if they perform a search within the mobile section. The page displays search results using the Smart search results web part.



The figure below shows what the mobile section looks like when viewed on various mobile devices:

CSS stylesheet for the mobile section

The pages of the mobile section use a different CSS stylesheet than the rest of the website — **Corporate Site – Mobile**. The mobile section's master page has the stylesheet assigned in **Properties -> General -> CSS stylesheet**. All pages in the mobile section inherit this configuration and use the

same stylesheet.

7.14.8 Device macro reference

CurrentDevice

The *CurrentDevice* object holds information about the device detected for the current visitor. The system identifies devices based on their user agent and retrieves the data about the capabilities of devices from the integrated 3rd party component 51degrees.mobi or the *~\App_Data\CMSModules\DeviceProfile\devices.xml* file.

You can take advantage of the *CurrentDevice* object when defining conditions for device profiles and in general macro expressions.

The following list presents the properties of the **CurrentDevice** object. Most of the properties are only available in the 51degrees.mobi Premium data:

- BitsPerPixel - color depth that the device's display supports.
- BrowserName
- BrowserVendor
- BrowserVersion
- CcppAccept - list of MIME types that the device supports.
- CookieCapable - indicates if the device can accept cookies.
- DeviceName
- HardwareModel
- HardwareName
- HardwareVendor
- HasCamera
- HasKeypad
- HasQwertyPad
- HasTouchscreen
- HasTrackpad
- HasVirtualQwerty
- HtmlVersion
- IsCrawler
- IsMobile
- IsTablet
- JavaScript - indicates if the device supports JavaScript.
- LayoutEngine - name of the rendering engine that the device uses.
- MaxScreenSize - size of the longer side of the device's screen.
- PlatformName
- PlatformVendor
- PlatformVersion
- ScreenPixelsHeight
- ScreenPixelsWidth
- StreamingAccept
- SupportedBearers
- UserAgent

CurrentDeviceProfile

To get information about the device profile assigned to the current visitor, use the *CurrentDeviceProfile* object in macros:

- You can access the device profile's system data through its properties, for example: *{% CurrentDeviceProfile.ProfilePreviewWidth %}*
- To directly get the device profile's name, use the following expression: *{% CurrentDeviceProfileName %}*

7.15 Microsoft Silverlight integration

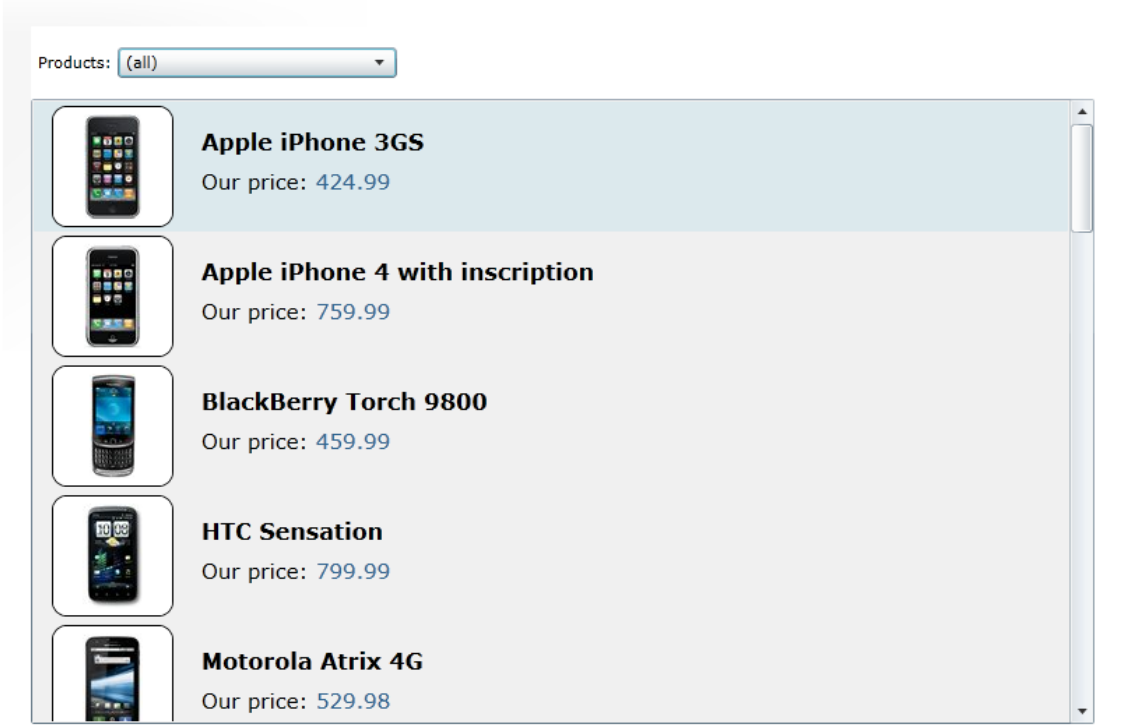
7.15.1 Overview

Kentico CMS comes with native support of **Microsoft Silverlight**. Microsoft Silverlight is a cross-browser, cross-platform technology for building and delivering the media experiences and Rich Interactive Applications (RIA) for the Web.

Silverlight applications run in the internet browser. All you need is a small plug-in installed in your browser. The plug-in is free and in case that users access a site containing a Silverlight application without this plug-in installed, they will be able to install Silverlight using an install banner leading to the download link.

You can find a live example of use of the Silverlight application web part on the **Corporate Site** sample website, on a page under **Corporate site -> Examples -> Web parts -> Silverlight -> Silverlight application**.

Example of the web part:



How it works in general

1. The developer creates a website with a built-in Silverlight application.
2. The site visitor navigates to that site using an internet browser.
3. If the user does not already have the required plug-in installed in the browser, they are automatically prompted to install it.
4. The Silverlight application is executed.

Creating Silverlight applications

Silverlight is a **.NET Framework** based technology, so if you are familiar with development using **Visual Studio** and one of the **.NET Framework languages** like **C#**, it will be much easier for you to learn Silverlight.
For developing Silverlight applications, you will need at least **Microsoft Visual Studio** with **Silverlight Tools**. There is one more powerful tool for designers - **Expression Blend**, which enables you to create application design in a really comfortable way. We also strongly recommend installing the **Silverlight Toolkit**, which brings many new controls that can be used in your Silverlight applications.

Visit the Silverlight community site where you can download all the required components. What is more, you can find valuable tutorials there, which can help you get started in the development of Silverlight applications.

We also recommend reading the official Microsoft Silverlight documentation.

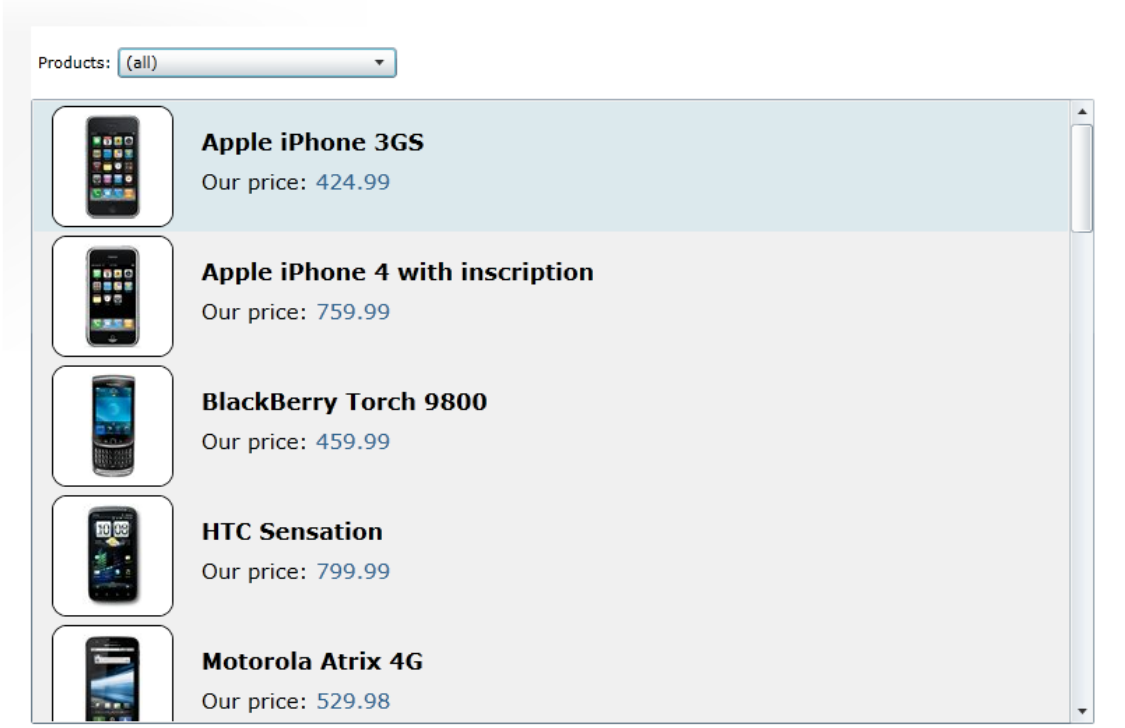
- To learn how to add a Silverlight application to your site, please refer to the Adding a Silverlight application to your site topic.
- If you would like to learn how to configure Internet Information Services (IIS) to enable Silverlight applications on your site, please refer to the IIS configuration topic.

7.15.2 Adding a Silverlight application to your site

To add a Silverlight application to your site, you will need to use the **Silverlight application** web part, which is a container for Silverlight applications.

You can find a live example of use of this web part on the **Corporate Site** sample website, on a page under **Corporate site -> Examples -> Web parts -> Silverlight -> Silverlight application**.

Example of the web part:



Adding a Silverlight application web part to a website

1. Go to **CMS Desk** and from the content tree, choose the page where you would like to add your Silverlight application.

2. Switch to the **Design** tab.

3. Click the **Add web part** () icon at the top right-hand corner of the web part zone where you want to place the application.

4. Choose the **Silverlight -> Silverlight application** web part.

5. Click **OK**.

6. Configure the web part properties:

- **Application path** - path to your Silverlight application; e.g. *~/ClientBin/MyApplication.xap.*
- **Minimum version** - minimum version of Silverlight required by the silverlight application to run by this web part.
- **Container width** - width of the application container; can be entered either as an integer value (e.g. *315*) or as a percentage value (e.g. *59%*).
- **Container height** - height of the application container; can be entered either as an integer value (e.g. *315*) or as a percentage value (e.g. *59%*).
- **Container background** - background of the application container; can be entered in hexadecimal (e.g. *#323232*) format or selected from a Color picker, which is displayed after clicking the Select button; if no value is entered, white color is used.
- **Endpoint address** - web service endpoint address the client application can connect to; if specified,

its value is added as parameter with 'endpoint' key to the application parameters collection; you need to handle this parameter in your Silverlight application for it to take effect.
- **Parameters** - Silverlight application parameters in the following format: *<key1>=<value1>, <key2>=<value2>,...*
- **Alternate content** - custom HTML content which is displayed to users when Silverlight plug-in is not installed; leave blank if you want the default alternate content to be displayed.

7. Click **OK**.

Now you can switch to the **Live site** and enjoy your Silverlight application running on your Kentico CMS website.

7.15.3 IIS configuration

Silverlight introduces three new file extensions (MIME types):

- **.xaml**
- **.xap**
- **.xbap**

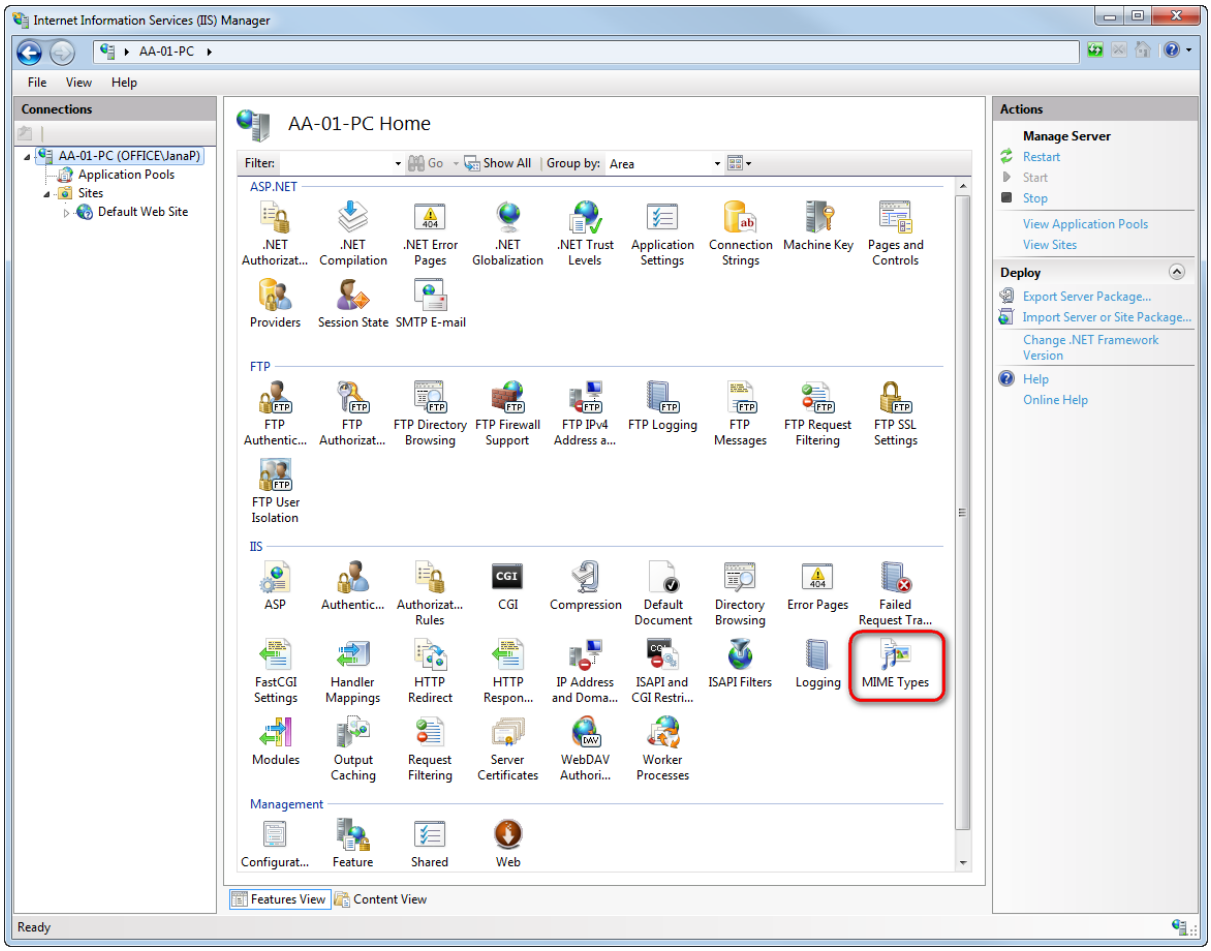
All these MIME types need to be configured in your IIS. By default, they are already implemented in IIS 7 (and above) in Windows Server 2008 and Windows Vista SP1.

If you run Windows Vista without SP1, you will have to add these extensions into your IIS.

Adding Silverlight MIME types into IIS on Windows Vista

1. Open **IIS Manager**.

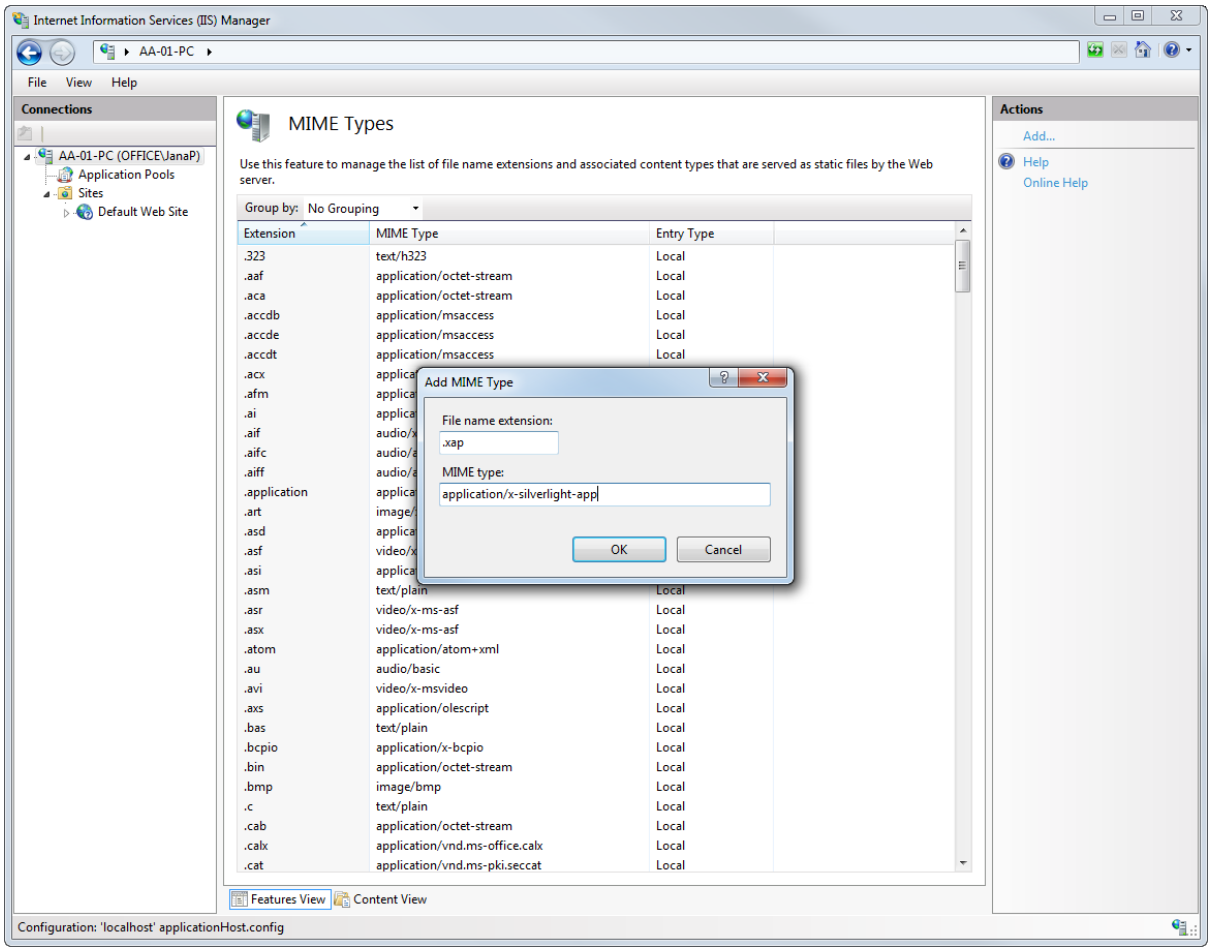
2. Double-click **MIME types** in the IIS section.



3. Click **Add...**.

4. Type the following information:

- **File name extension:** .xap
- **MIME type:** application/x-silverlight-app



5. Click **OK**.

6. Repeat for the other two types:

- **File name extension:** .xaml
- **MIME type:** application/xaml + xml

- **File name extension:** .xbap
- **MIME type:** application/x-ms-xbap

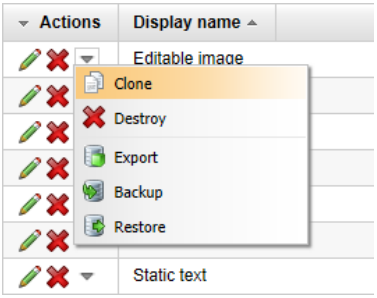
You have now configured your IIS to support Silverlight applications.

7.16 Object cloning

7.16.1 Overview

Cloning provides an easy way to create an exact copy of an object. It is supported for all objects throughout the system, from the most basic records to complex data structures such as document types or user accounts. If you wish to reuse any part of an existing item or create a modified version of it, cloning is much more efficient than adding a new object of the given type and manually carrying over the values and various configuration options from the original.

To clone an object, find it on the listing page in the appropriate section of the administration interface, click the **Other actions** button next to it and select the **Clone** option in the drop-down menu.

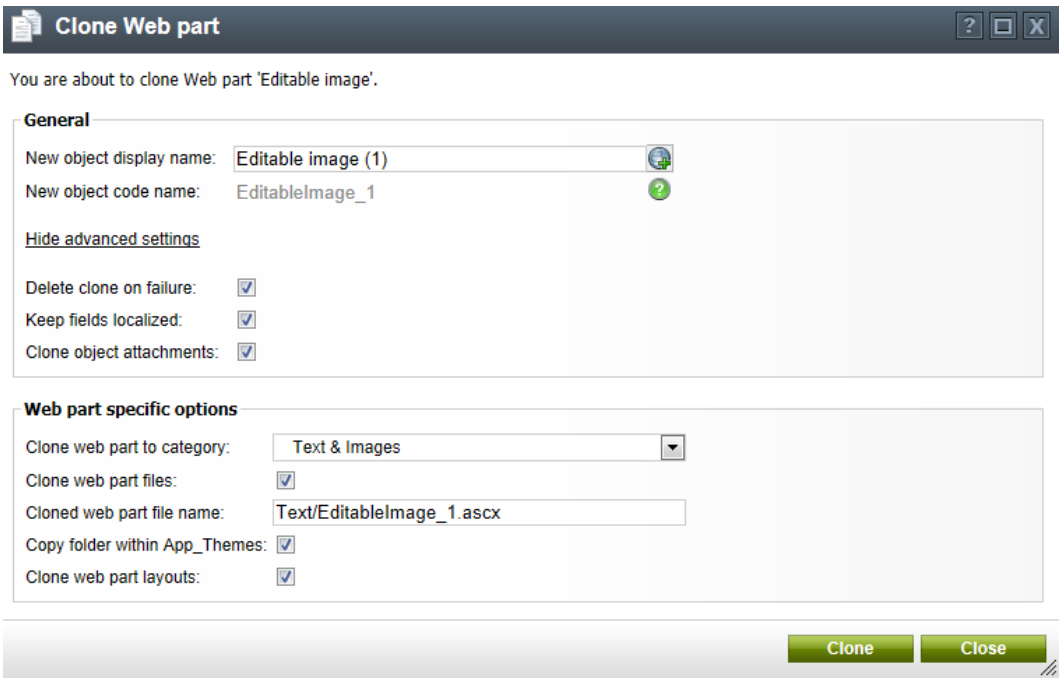


Objects that are displayed in a tree menu instead of a standard list may be cloned via a button available in the panel above the tree.

Using the clone action opens a dialog where you can specify exactly how the cloned object should be created. In most cases, you will be asked to enter a *display name* and *code name* for the new object. Further settings can be accessed by clicking **Show advanced settings**, which offers the following options:

- **Delete clone on failure** - if enabled, the new object will be removed if an error occurs during any part of the cloning process.
- **Keep fields localized** - if checked, the cloned object will transfer all localization macros (i.e. resource strings) placed inside its fields. Otherwise they will be replaced by the values entered for the default culture. This does not affect the *Display name* field, which is filled in manually using the *New object display name* field.

Additional settings depend on the type of the cloned object. They typically allow you to choose what types of associated data should be cloned along with the given object, such as for example child objects, attachments, bindings to other objects etc.



Many objects also offer even more configuration options in the **specific options** section. These must be specified so that the system can correctly create the clone (particularly in the case of complex objects). Information about individual settings can be found in the tooltip displayed when hovering over the names of fields.

7.16.2 Cloning objects through the API

If necessary, objects may also be cloned by calling the appropriate API in your custom code (for example in global event handlers or in the logic of user controls and web parts).

The following sample code demonstrates how this can be done. The **CloneCountry** method in the example creates a clone of the USA country object, including all child states. It also manually sets the country code values of the new object.

[C#]

```
using CMS.SiteProvider;
using CMS.SettingsProvider;

...

public void CloneCountry()
{
    // Gets the country to be cloned.
    CountryInfo country = CountryInfoProvider.GetCountryInfo("USA");
    if (country != null)
    {
        // Prepares the settings used for the cloning process.
        CloneSettings settings = new CloneSettings()
        {
            // Sets the clone names.
            CodeName = "MyClonedCountry",
            DisplayName = "My Cloned Country",

            // Ensures that the state objects under the country are also cloned.
            IncludeChildren = true,

            // Registers a callback method that performs additional actions before
the clone is added.
            BeforeCloneInsertCallback = ChangeCountryCodes
        };

        // Clones the country according to the defined settings.
        country.Generalized.InsertAsClone(settings);
    }
}

private void ChangeCountryCodes(CloneSettings settings, BaseInfo
cloneToBeInserted)
{
    // Changes the values of additional fields before the clone is inserted to the
DB.
```

```
    CountryInfo country = cloneToBeInserted as CountryInfo;
    if (country != null)
    {
        country.CountryThreeLetterCode = "MCC";
        country.CountryTwoLetterCode = "MC";
    }
}
```

Before an object can be cloned, it is necessary to prepare an instance of the **CloneSettings** class (from the **CMS.SettingsProvider** namespace). By assigning values to the properties of this object, you can configure how the cloning process will be performed. In addition to the code name and display name, the available properties correspond with the advanced cloning settings. In the example, the **IncludeChildren** flag is set to true, so all state objects under the given country will also be cloned and assigned to the new country.

The **BeforeCloneInsertCallback** property is used to register a custom handler method that will be executed just before the clone is inserted into the database. Such methods allow you to implement any functionality required to correctly clone the object. The object that is being cloned and the corresponding *CloneSettings* object are passed as parameters, so you can dynamically set the values based on the currently assigned clone settings.

Other possible callback options are:

- **AfterCloneInsertCallback** - executed after the cloned object itself is created and inserted, but before the system starts cloning any associated child objects or bindings.
- **AfterCloneStructureInsertCallback** - called once all objects included in the cloning process are created.

Finally, the clone action itself is performed by calling the **InsertAsClone** method for the original country object (converted to a generalized object type), with the prepared *CloneSettings* specified through the parameter.

7.17 Object versioning

7.17.1 Overview

Similarly as versioning of documents, object versioning allows creation of versions when an object in Kentico CMS is edited and saved. This is convenient in case that you want to compare particular versions or roll back to one of the previous ones. This may potentially save lots of repeated work when some unwanted modifications are made to an object and you want to get the object to the original state.

Versioning is only possible for certain object types. In the Supported object types topic, you can find their list, together with exact locations of the administration interface sections where these objects can be edited and where their versions can be viewed and managed.

Even though object versioning is enabled and functional by default, it is recommended to configure settings related to it in **Site Manager -> Settings -> Versioning & synchronization -> Object versioning**. To learn more about all settings that can be adjusted there, please refer to the Configuring object versioning topic.

Once object versioning is enabled and configured, new versions are created when objects are modified

under certain circumstanced. In the Using object versioning topic, you can find exact information about when versions are created and how they can be viewed, compared and managed.

Another feature related to object versioning is the Objects recycle bin. This feature allows certain objects to be deleted to the recycle bin instead of being deleted permanently. Objects deleted to the recycle bin can be restored, so you may avoid unwanted deletion of an object and the subsequent need to create it again.

In the Object versioning internals and API sub-chapter, you can find information about database tables and classes that are used by object versioning, as well as several examples of how object versions can be managed using Kentico CMS API.

7.17.2 Supported object types

In the following table, you can find all object types that support versioning. In the **Editing interface** column, you can find the exact location within Kentico CMS administration interface where objects of the particular type can be edited. If versioning is enabled for a particular object type, the **Versions** tab where particular versions of the edited object can be viewed and managed is available directly in the respective editing interface.

Object type	Editing interface
Alternative forms	• CMS Desk -> Tools -> Forms -> edit (✎) -> Alternative forms -> edit (✎) • Site Manager -> Development -> Document types -> edit (✎) -> Alternative forms -> edit (✎) • Site Manager -> Development -> Custom tables -> edit (✎) -> Alternative forms -> edit (✎)
CSS stylesheets	• CMS Desk -> Content -> Edit -> Properties -> General -> click **Edit** • Site Manager -> Development -> CSS stylesheets -> edit (✎) • other interfaces containing the stylesheet selector (e.g. when editing a site, department, etc.)
Custom table definitions	• Site Manager -> Development -> Custom tables -> edit (✎)
Document type definitions	• Site Manager -> Development -> Document types -> edit (✎)
E-mail templates	• Site Manager -> Administration -> E-mail templates -> edit (✎)
Form definitions	• CMS Desk -> Tools -> Forms -> edit (✎)
Media files	• CMS Desk -> Tools -> Media -> edit (✎) -> select a file
Newsletter issues	• CMS Desk -> Tools -> Newsletters -> edit (✎) -> Issues -> edit (✎)
Newsletter templates	• CMS Desk -> Tools -> Newsletters -> Templates -> edit (✎)
Page layouts *	• Site Manager -> Development -> Page layouts -> edit (✎) • CMS Desk -> Content -> Edit -> Design -> mouse over Edit layout -> Shared layout versions
Page templates	• Site Manager -> Development -> Page templates -> select a template • CMS Desk -> Content -> Edit -> Design -> mouse over Edit template -> Template versions

	• other interfaces that allow editing of page templates
Queries **	• Site Manager -> Development -> Document types -> edit () -> Queries -> edit () • Site Manager -> Development -> Custom tables -> edit () -> Queries -> edit () • web part properties dialogs of web parts that have query properties
Report graphs	• CMS Desk -> Tools -> Reporting -> select a report -> General -> select a graph and click **Edit**
Report tables	• CMS Desk -> Tools -> Reporting -> select a report -> General -> select a table and click **Edit**
Report values	• CMS Desk -> Tools -> Reporting -> select a report -> General -> select a value and click **Edit**
Report definitions	• CMS Desk -> Tools -> Reporting -> select a report
Transformations	• Site Manager -> Development -> Document types -> edit () -> Transformations -> edit () • Site Manager -> Development -> Custom tables -> edit () -> Transformations -> edit () • web part properties dialogs of web parts that have transformation properties
Web part containers	• Site Manager -> Development -> Web part containers -> edit ()
Web part layouts **	• Site Manager -> Development -> Web parts -> select a web part -> Layout -> edit () • CMS Desk -> Content -> Edit -> Design -> Configure () a web part -> Layout
* Only shared page layouts are versioned — custom layouts are versioned as part of the data of the parent page template. ** Only custom queries and web part layouts are versioned — system queries and default web part layouts are not versioned.	

7.17.3 Configuring object versioning

Object versioning settings can be adjusted in **Site Manager -> Settings -> Versioning & synchronization -> Object versioning**. The following settings are available:

General	
Enable object versioning	Globally enables or disables object versioning. This option is enabled by default. If disabled, no versions are created when objects are modified.
Delete objects to recycle bin	Determines which objects should be moved to recycle bin when deleted: • **No** - no objects are moved to recycle bin when deleted, i.e. they are deleted permanently. • **Versioned objects only** - only objects that support versioning and

	whose versioning is enabled by the settings below are moved to recycle bin when deleted. • **All objects** - all objects that support staging synchronization are moved to recycle bin when deleted.
Team development	
Use check-in/check-out for objects	Indicates if object locking (check-in/check-out) should be used for virtual objects.
Keep new objects checked out	Indicates if new objects are automatically checked out upon creation.
Version history	
Version history length (major versions)	Indicates how many major versions of a single object will be stored in its version history. If the number of major versions exceeds this value, the oldest versions are deleted. If set to 0, major versions history length is not limited.
Version history length (minor versions)	Indicates how many minor versions of a single object will be stored in its version history. If the number of versions exceeds this value, the oldest version is deleted. If set to 0, minor versions history length is not limited.
Save to last version if younger than (minutes)	If an object is edited and saved within this number of minutes since it was last saved, changes made to it are saved to the last version. If it is saved after more minutes since it was last saved, a new version is created. If set to 0, a new version is created whenever you save an edited object.
Promote to major version if older than (hours)	If an object is edited and saved after this number of hours since it was last saved, a new major version is created. If it is saved earlier, a new minor version is created or the changes are saved to the latest version (depending on the setting above). If set to 0, major versions are never created automatically.
Use object versioning for	
Alternative forms	Indicates if versioning of document type, on-line forms and custom table alternative forms is allowed.
CSS stylesheets	Indicates if versioning of CSS stylesheets is allowed.
Custom table definitions	Indicates if versioning of custom table definitions is allowed.
Document type definitions	Indicates if versioning of document type definitions is allowed.
E-mail templates	Indicates if versioning of e-mail templates is allowed.
Form definitions	Indicates if versioning of on-line form definitions (the Forms module) is allowed.
Media files	Indicates if versioning of media files (the Media module) is allowed. This options is disabled by default because versioning of large media files may consume an extensive amount of database space.
Media files versioned extensions	Extensions of versioned media files. Only media files with extensions enumerated here will be versioned.

Newsletter issues	Indicates if versioning of newsletter issues is allowed.
Newsletter templates	Indicates if versioning of newsletter templates is allowed.
Page layouts	Indicates if versioning of page layouts is allowed.
Page templates	Indicates if versioning of page templates is allowed.
Queries	Indicates if versioning of document type and custom table queries is allowed. Only custom queries are versioned — system queries are not versioned because they are re-generated by the system automatically when their parent object is modified.
Report graphs	Indicates if versioning of report graphs is allowed.
Report tables	Indicates if versioning of report tables is allowed.
Report values	Indicates if versioning of report values is allowed.
Report definitions	Indicates if versioning of report definitions is allowed.
Transformations	Indicates if versioning of document type and custom table transformations is allowed.
Web part containers	Indicates if versioning of web part containers is allowed.
Web part layouts	Indicates if versioning of web part layouts is allowed.
Workflows	Indicates if versioning of workflow objects is allowed.

7.17.4 Using object versioning

If object versioning is enabled as described in the Configuring object versioning topic, the **Versions** tab is displayed in editing interfaces of objects whose versioning is allowed. On this tab, particular versions of the currently edited object are created when objects are edited under certain circumstances described in the following paragraph.

How versions are created and numbered

The following points summarize when new versions are created and what numbering is used for them:

- The current version is always the one in the top line of the grid. If object versioning is disabled additionally when there are already some versions, the top version needn't contain current object data if the object was edited after object versioning had been disabled.
- If you create a new object, the initial 0.1 version is created on the tab automatically.
- If you edit an existing object that had been created before object versioning was enabled, there are no versions on the tab initially. When you edit it and save the modifications, two versions are created: 1.0 with the original data and 1.1 with the currently saved data.
- Every other edit and save creates a new minor version (the version number is incremented by 0.1) if you save after the number of minutes configured in the **Save to last version if younger than (minutes)** setting since last save. If you save within the number of minutes, object data is saved to the last version.

- If you edit and save after the number of hours configured in the **Promote to major version if older than (hours)** setting, the latest version is promoted to a major version and the new version follows the new major numbering (i.e. latest 2.3 is promoted to 3.0 and the new one is numbered 3.1).

- The current version can also be promoted to a major version manually by clicking the **Make current version major** button above the **Object history** listing.

Management of versions on the Versions tab

You can perform a number of actions with versions listed on the **Versions** tab. Above the grid, you can find the following two buttons:

- **Make current version major** - promotes the current object version to a new major version (e.g. if the latest version is 2.3, it is promoted to 3.0).
- **Destroy history** - deletes all listed versions of the object.

In each objects's row, you can find the following actions in the **Actions** column:

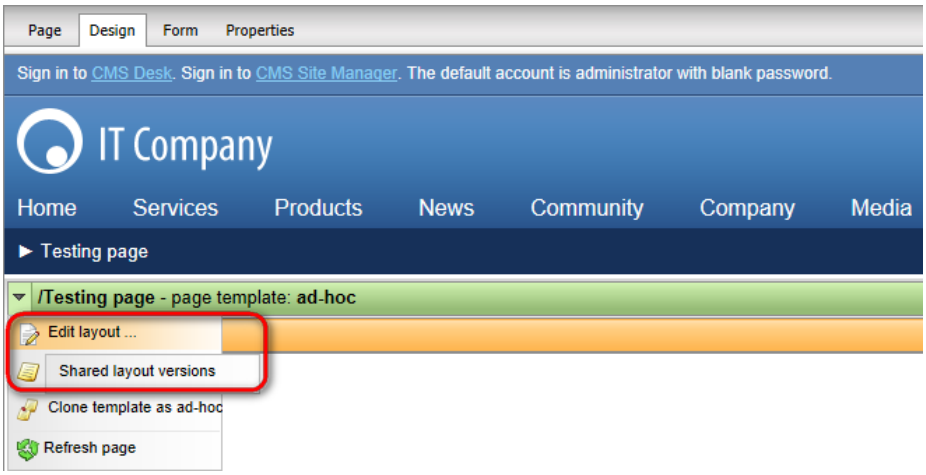
- **View version** - opens a new window where field data of the version can be inspected and compared side-by-side with data of another version. See the Version data view and comparison section below for more details.
- **Rollback version** - performs rollback of the object to the particular version. If the object has some child objects (e.g. alternative forms, transformations, queries, ...), they are not included in the rollback.
- **Delete** - deletes the version so that it is no longer available in the version history.

The following action is available in the object menu accessible by clicking the ▾ button in an object's line:

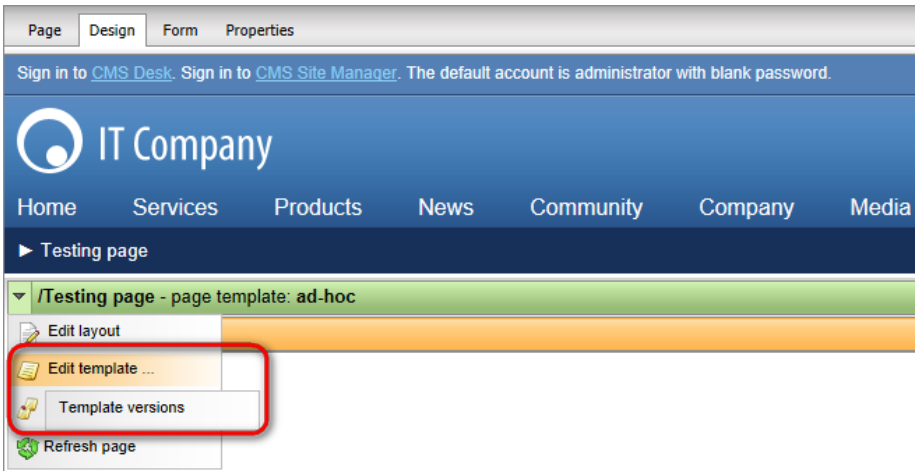
- **Rollback with children** - if the object has some child objects (e.g. alternative forms, transformations, queries, ...), it performs rollback of the object to the particular version, while child object data are rolled back as well. If it does not have any child objects, only the object itself is rolled back.

Management of versions in CMS Desk -> Content -> Edit -> Design

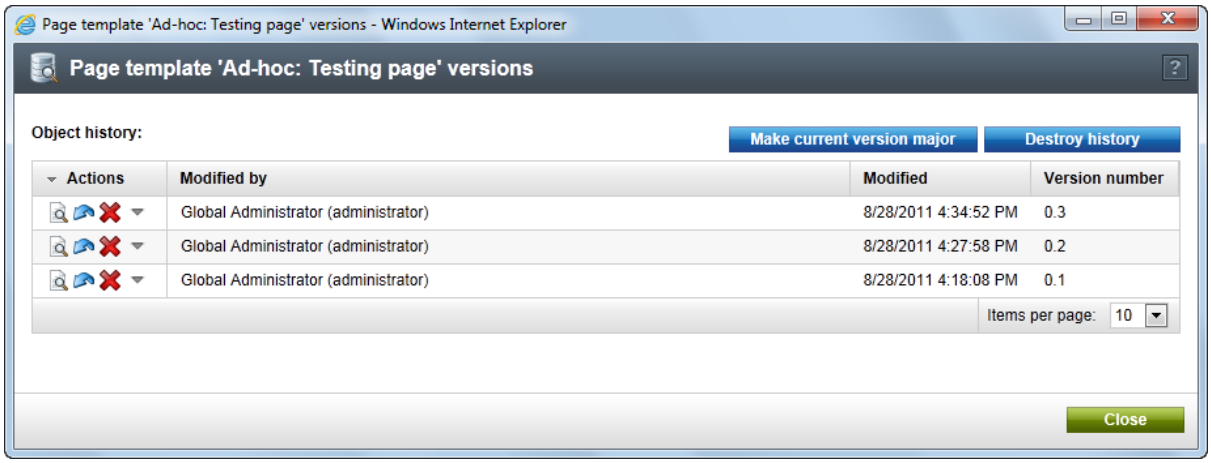
If you expand a page template's drop-down menu and hold your mouse pointer over the **Edit layout** menu item, an additional menu item **Shared version layouts** appears.



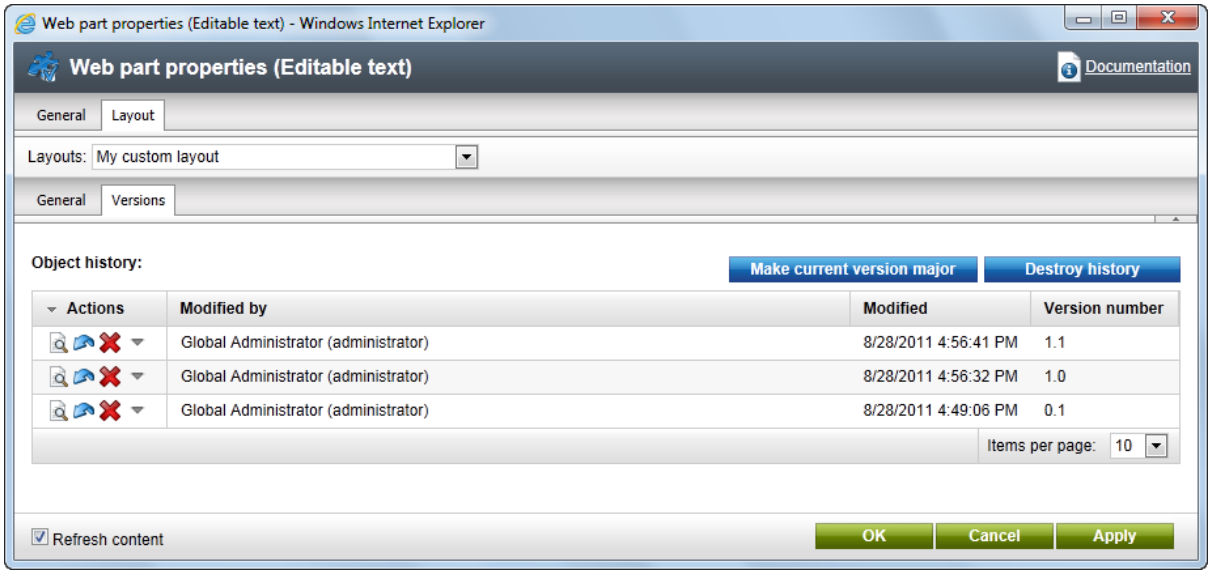
Similarly, if you hold the mouse pointer over the **Edit template** menu item, an additional menu item **Template versions** appears.



After clicking both of the menu items mentioned above, a dialog box appears, letting you view and manage versions of the page layout or template. The same actions as on the **Versions** tab described above can be performed here.



The **Versions** tab is also available in the **Web part properties** dialog if you choose a custom layout on the **Layout** tab. Again, the same options as described above can be performed here.



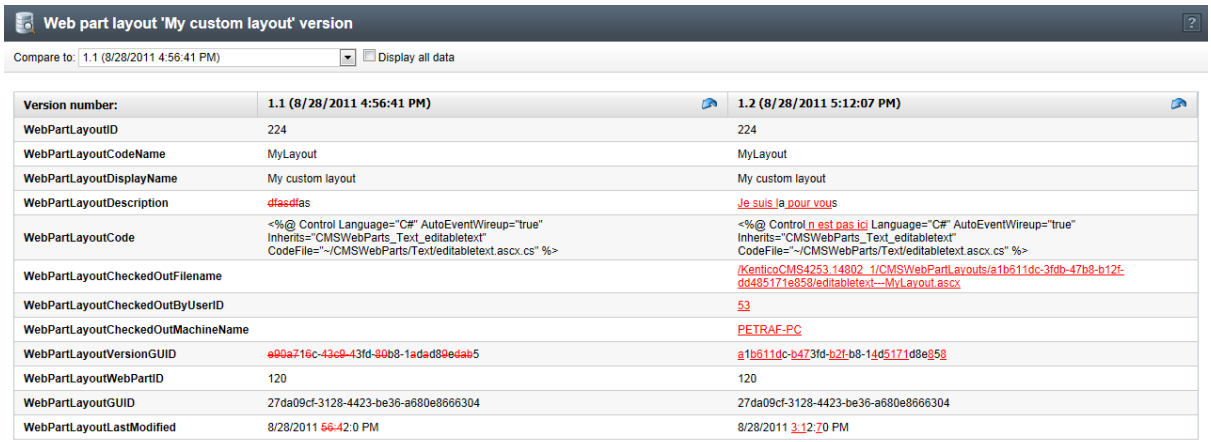
Version data view and comparison

If you click the **View version** icon in a version's row in the listing, a pop-up window is displayed. In this pop-up window, you can view data stored in fields of an object version and compare it to another version. If there is more than version of the object, the following options are available:

- **Compare to** - by selecting a version in this drop-down list, field data of the selected version get compared side-by-side with field data of the originally viewed version in the table below. Differences between the two versions are highlighted in red font.
- **Display all data** - by enabling this option, all data related to the object and all its child objects will be displayed in the table below.

In the top right corner of a version's column in the table, you can find the following action icons:

- **Roll back to this version** - this icon is displayed when the **Display all data** option is disabled. Clicking the icon performs rollback of the object to the respective version. If the object has some child objects (e.g. alternative forms, transformations, queries, ...), they are not included in the rollback.
- **Roll back with children to this version** - this icon is displayed instead of the one above when the **Display all data** option is enabled. If the object has some child objects (e.g. alternative forms, transformations, queries, ...), it performs rollback of the object to the particular version, while child object data are rolled back as well. If it does not have any child objects, only the object itself is rolled back.



Version number:	1.1 (8/28/2011 4:56:41 PM)	1.2 (8/28/2011 5:12:07 PM)
WebPartLayoutID	224	224
WebPartLayoutCodeName	MyLayout	MyLayout
WebPartLayoutDisplayName	My custom layout	My custom layout
WebPartLayoutDescription	dfasdfas	Je suis la pour vous
WebPartLayoutCode	<%@ Control Language="C#" AutoEventWireup="true" Inherits="CMSWebParts_Text_editabletext" CodeFile="~/CMSWebParts/Text/editabletext.ascx.cs" %>	<%@ Control n est pas ici Language="C#" AutoEventWireup="true" Inherits="CMSWebParts_Text_editabletext" CodeFile="~/CMSWebParts/Text/editabletext.ascx.cs" %>
WebPartLayoutCheckedOutFilename		/KenticoCMS4253_14802_1/CMSWebPartLayouts/a1b611dc-3fdb-47b8-b12f-dd485171e858/editabletext---MyLayout.ascx
WebPartLayoutCheckedOutByUserID		53
WebPartLayoutCheckedOutMachineName		PETRAF-PC
WebPartLayoutVersionGUID	a90a716c-43c9-43fd-80b8-1adad89edab5	a1b611dc-b473fd-b2f-b8-14d5171d8e858
WebPartLayoutWebPartID	120	120
WebPartLayoutGUID	27da09cf-3128-4423-be36-a680e8666304	27da09cf-3128-4423-be36-a680e8666304
WebPartLayoutLastModified	8/28/2011 56:42:0 PM	8/28/2011 3:12:70 PM

7.17.5 Objects recycle bin

It is possible to configure that deleted objects are removed to recycle bin instead of being deleted permanently. This is possible by enabling the **Delete objects to recycle bin** option in **Site Manager -> Settings -> Versioning & synchronization -> Object versioning**. See the Configuring object versioning topic for more details on how it can be configured. Once the option mentioned above is enabled, deleted objects (only those that match the configuration) are moved to the recycle bin.

Global administrators can view objects removed by all user in **Site Manager -> Administration -> Recycle bin -> Objects**. Using the **Select site** drop-down list, you can select a site for that the deleted objects will be displayed. You can also choose **(global objects)** to display only objects that are not site-related, or **(all sites or global)** to display all objects in the recycle bin. Using the filter above the grid, you can define specific criteria and click **Show** to display only objects that match the criteria.

The following actions are available for each of the listed objects:

- **View** - opens a new window where detailed view of data stored in fields of the objects is displayed.
- **Restore** - restores the object, i.e. moves it from the recycle bin back to the respective part of the administration interface.
- **Delete** - removes the object from the recycle bin so that it is removed permanently and can't be restored.

The following actions are available in the menu accessible by clicking the ▾ button. Their functionality depends on the type of object:

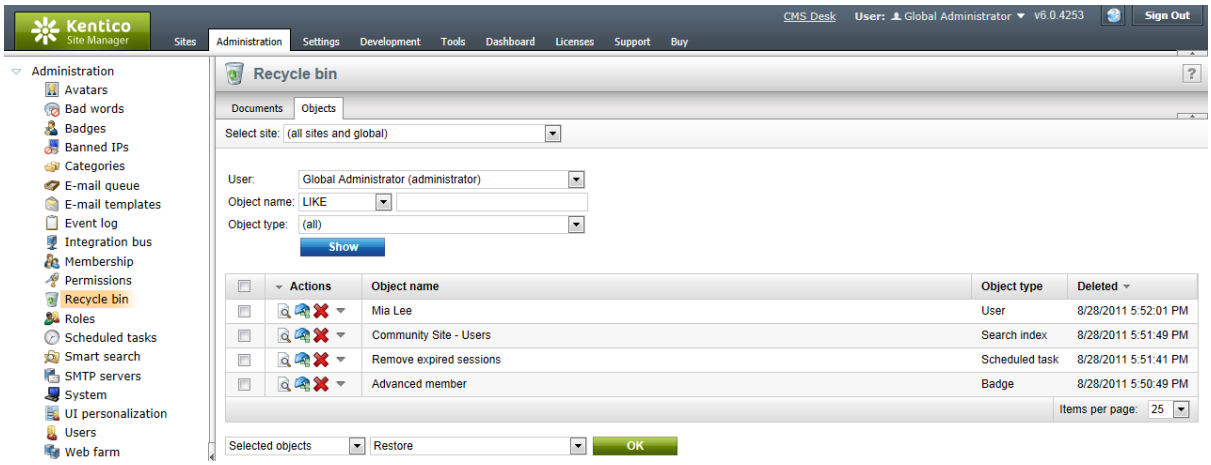
- Global objects without site bindings:
  - **Restore without site bindings** - performs the standard restore action.
  - **Restore to current site** - performs the standard restore action.
- Global objects with site bindings:
  - **Restore without site bindings** - restores the object, but does not assign it to any website.
  - **Restore to current site** - restores the object and assigns it to the current website (the one that is using the domain in the current URL).

- Site objects (objects whose definition contains the *SiteID* column):
  - **Restore without site bindings** - performs the standard restore action.
  - **Restore to current site** - restores the object and assigns it to the current website (the one that is using the domain in the current URL).

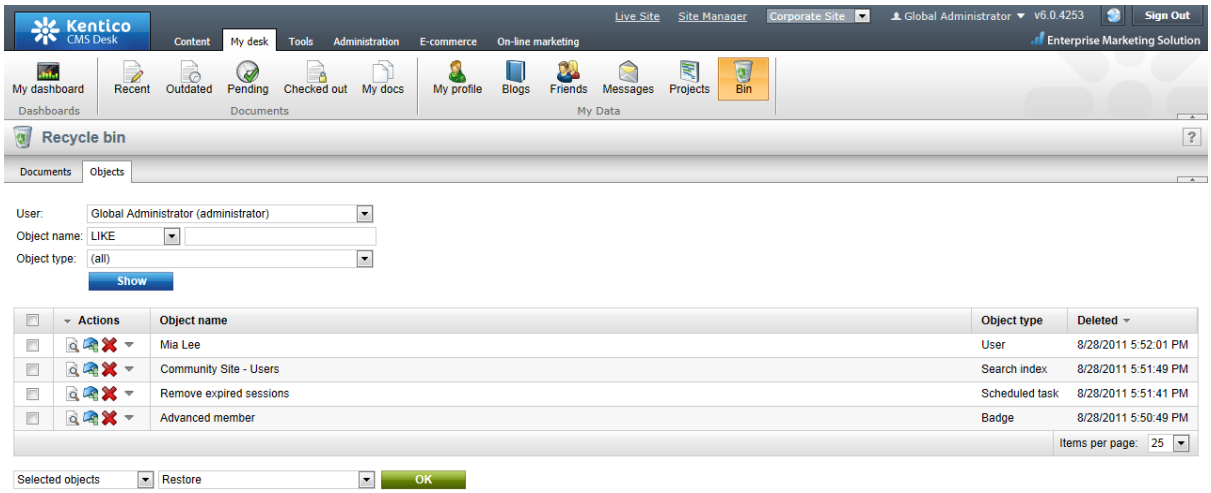
Using the two drop-down lists below the grid, you can perform the **Restore**, **Delete**, **Restore without site bindings** and **Restore to current site** actions to more objects in a single click. Using the first drop-down list, you can choose:

- **All objects** - the bulk action will be performed to all objects in the recycle bin.
- **Selected objects** - the bulk action will be performed to objects selected using the check-boxes (☑).

Using the second drop-down list, choose the required action and click **OK** to perform it.



Objects that were deleted by the current user can be viewed in **CMS Desk -> My desk -> Recycle bin -> Objects**. Here, only global objects and objects related to the current site are displayed. Therefore, the **Select site** drop-down list is not available here and the **User** drop-down list in the filter is available only to global administrators. Otherwise, the same actions as described above can be performed.



7.17.6 Object versioning internals and API

7.17.6.1 Overview

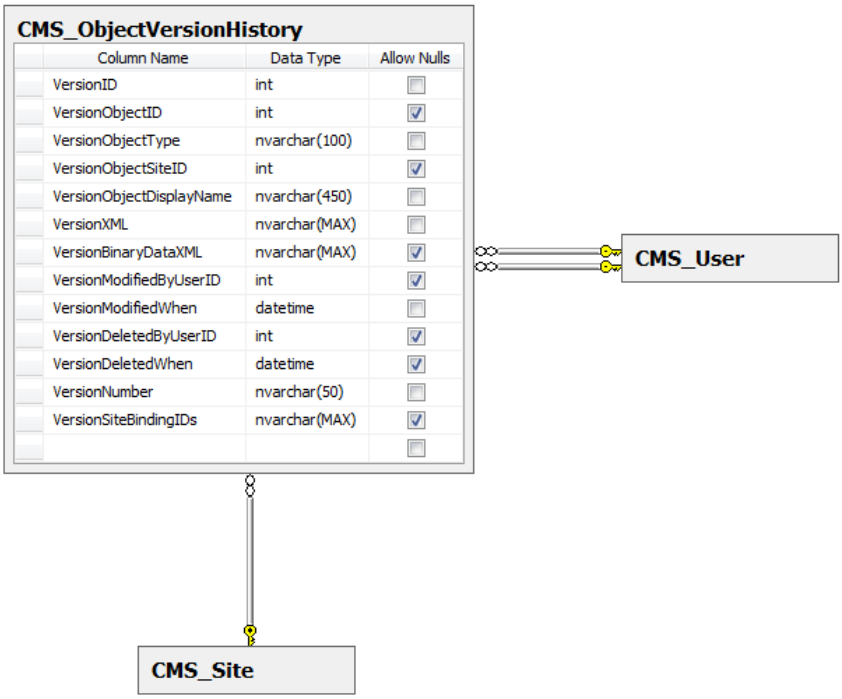
In this chapter, you will learn which database tables and API classes are used to provide the object versioning functionality. You will also see the most common API examples.

Further API-related information and examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all members of the related classes, please refer to Kentico CMS API Reference.

7.17.6.2 Database tables

The following database tables are used by object versioning:

- **CMS_ObjectVersionHistory** - contains records representing particular versions of versioned objects.



7.17.6.3 API classes

If you are not familiar with database table data management through Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

> **Please note**
>
> The classes for object versioning can be found in the **CMS.Synchronization** namespace.

CMS_ObjectVersionHistory table API:

- **ObjectVersionManager** - provides functionality for management of object versions.
- **ObjectVersionHistoryInfo** - represents one object version.

7.17.6.4 API examples

7.17.6.4.1 Overview

These topics show examples of how the object versioning API examples can be used:

- Object versioning
- Object recycle bin

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Development\ObjectVersioning\Default.aspx.cs***.

The object versioning API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.GlobalHelper;
using CMS.UIControls;
using CMS.CMSHelper;
using CMS.SiteProvider;
using CMS.Synchronization;
using CMS.SettingsProvider;
```

7.17.6.4.2 Object versioning

The following example creates a versioned CSS stylesheet.

```
private bool CreateVersionedObject()
{
    // Create new css stylesheet object
    CssStylesheetInfo newStylesheet = new CssStylesheetInfo();

    // Check if object versioning of stylesheet objects is allowed on current site
    if (ObjectVersionManager.AllowObjectVersioning(newStylesheet))
    {
        // Set the properties
        newStylesheet.StylesheetDisplayName = "My new versioned stylesheet";
```

```
        newStylesheet.StylesheetName = "MyNewVersionedStylesheet";
        newStylesheet.StylesheetText = "Some versioned CSS code";

        // Save the css stylesheet
        CssStylesheetInfoProvider.SetCssStylesheetInfo(newStylesheet);

        // Add css stylesheet to site
        int stylesheetId = newStylesheet.StylesheetID;
        int siteId = CMSContext.CurrentSiteID;

        CssStylesheetSiteInfoProvider.AddCssStylesheetToSite(stylesheetId,
siteId);

        return true;
    }

    return false;
}
```

The following example creates a new version of the CSS stylesheet created by the example above.

```
private bool CreateVersion()
{
    // Get the css stylesheet
    CssStylesheetInfo newStylesheetVersion =
CssStylesheetInfoProvider.GetCssStylesheetInfo("MyNewVersionedStylesheet");
    if (newStylesheetVersion != null)
    {
        // Check if object versioning of stylesheet objects is allowed on current
site
        if (ObjectVersionManager.AllowObjectVersioning(newStylesheetVersion))
        {
            // Update the properties
            newStylesheetVersion.StylesheetDisplayName =
newStylesheetVersion.StylesheetDisplayName.ToLower();

            // Create new version
            ObjectVersionManager.CreateVersion(newStylesheetVersion, true);

            return true;
        }
    }

    return false;
}
```

The following example performs rollback of the CSS stylesheet to the original version.

```
private bool RollbackVersion()
{
    // Get the css stylesheet
```

```
    CssStylesheetInfo stylesheet = CssStylesheetInfoProvider.GetCssStylesheetInfo
("MyNewVersionedStylesheet");
    if (stylesheet != null)
    {
        // Prepare query parameters
        string where = "VersionObjectID =" + stylesheet.StylesheetID + " AND
VersionObjectType = '" + stylesheet.ObjectType + "'";
        string orderBy = "VersionModifiedWhen ASC";
        int topN = 1;

        // Get dataset with versions according to the parameters
        DataSet versionDS = ObjectVersionHistoryInfoProvider.GetVersionHistories
(where, orderBy, topN, null);

        if (!DataHelper.DataSourceIsEmpty(versionDS))
        {
            // Get version
            ObjectVersionHistoryInfo version = new ObjectVersionHistoryInfo
(versionDS.Tables[0].Rows[0]);

            // Roll back
            ObjectVersionManager.RollbackVersion(version.VersionID);

            return true;
        }
    }

    return false;
}
```

The following example destroys the latest version of the CSS stylesheet.

```
private bool DestroyVersion()
{
    // Get the css stylesheet
    CssStylesheetInfo stylesheet = CssStylesheetInfoProvider.GetCssStylesheetInfo
("MyNewVersionedStylesheet");
    if (stylesheet != null)
    {
        // Get the latest version
        ObjectVersionHistoryInfo version = ObjectVersionManager.GetLatestVersion
(stylesheet.ObjectType, stylesheet.StylesheetID);

        if (version != null)
        {
            // Destroy the latest version
            ObjectVersionManager.DestroyObjectVersion(version.VersionID);

            return true;
        }
    }

    return false;
}
```

The following example destroys the whole version history of the CSS stylesheet.

```
private bool DestroyHistory()
{
    // Get the css stylesheet
    CssStylesheetInfo stylesheet = CssStylesheetInfoProvider.GetCssStylesheetInfo
("MyNewVersionedStylesheet");
    if (stylesheet != null)
    {
        // Destroy version history
        ObjectVersionManager.DestroyObjectHistory(stylesheet.ObjectType,
stylesheet.StylesheetID);

        return true;
    }

    return false;
}
```

The following example ensures that the CSS stylesheet has at least one version (if it does not have one, it creates it).

```
private bool EnsureVersion()
{
    // Get the css stylesheet
    CssStylesheetInfo stylesheet = CssStylesheetInfoProvider.GetCssStylesheetInfo
("MyNewVersionedStylesheet");
    if (stylesheet != null)
    {
        // Check if object versioning of stylesheet objects is allowed on current
site
        if (ObjectVersionManager.AllowObjectVersioning(stylesheet))
        {
            // Ensure version
            ObjectVersionManager.EnsureVersion(stylesheet, false);

            return true;
        }
    }

    return false;
}
```

7.17.6.4.3 Object recycle bin

The following example deletes the CSS stylesheet created on the previous page and moves it to the objects recycle bin.

```
private bool DeleteObject()
{
    // Get the css stylesheet
    CssStylesheetInfo deleteStylesheet =
```

```
CssStylesheetInfoProvider.GetCssStylesheetInfo("MyNewVersionedStylesheet");

    if (deleteStylesheet != null)
    {
        // Check if restoring from recycle bin is allowed on current site
        if (ObjectVersionManager.AllowObjectRestore(deleteStylesheet))
        {
            // Delete the css stylesheet
            CssStylesheetInfoProvider.DeleteCssStylesheetInfo(deleteStylesheet);

            return true;
        }
    }

    return false;
}
```

The following example restores the CSS stylesheet deleted by the example above from the objects recycle bin back to the administration interface.

```
private bool RestoreObject()
{
    // Prepare query parameters
    string where = "VersionObjectType = '" + SiteObjectType.CSSSTYLESHEET + "' AND
VersionDeletedByUserID = " + CMSContext.CurrentUser.UserID;
    string orderBy = "VersionDeletedWhen DESC";
    int topN = 1;

    // Get dataset with versions according to the parameters
    DataSet versionDS = ObjectVersionHistoryInfoProvider.GetVersionHistories
(where, orderBy, topN, null);

    if (!DataHelper.DataSourceIsEmpty(versionDS))
    {
        // Get version
        ObjectVersionHistoryInfo version = new ObjectVersionHistoryInfo
(versionDS.Tables[0].Rows[0]);

        // Restore the object
        ObjectVersionManager.RestoreObject(version.VersionID, true);

        return true;
    }

    return false;
}
```

The following example deletes the CSS stylesheet permanently without moving it to the object recycle bin.

```
private bool DestroyObject()
{
    // Get the css stylesheet
```

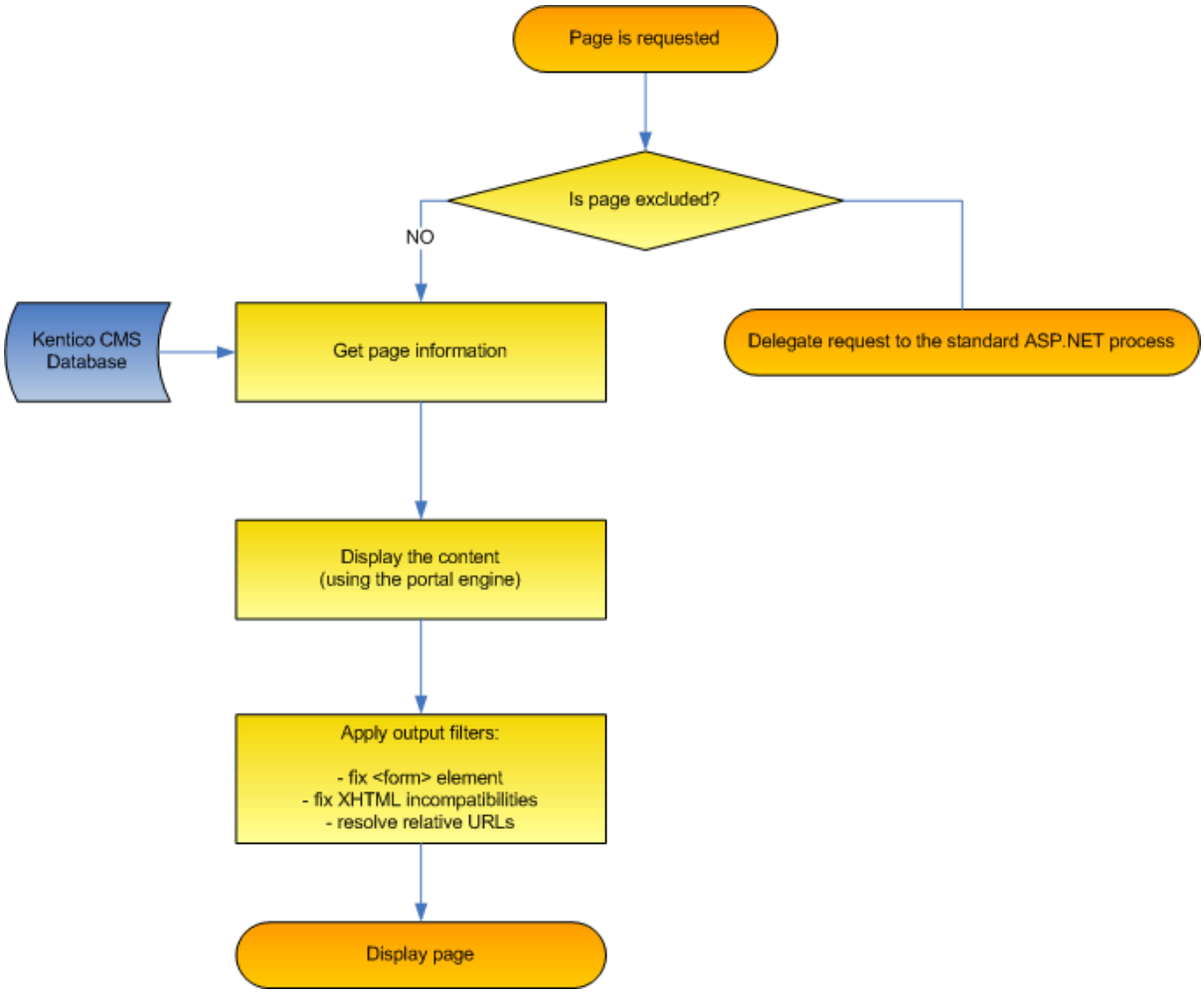
```
    CssStylesheetInfo destroyStylesheet =
CssStylesheetInfoProvider.GetCssStylesheetInfo("MyNewVersionedStylesheet");

    if (destroyStylesheet != null)
    {
        // Destroy the object (in action context with disabled creating of new
versions for recycle bin)
        using (CMSActionContext context = new CMSActionContext())
        {
            // Disable creating of new versions
            context.CreateVersion = false;

            // Destroy the css stylesheet
            CssStylesheetInfoProvider.DeleteCssStylesheetInfo(destroyStylesheet);

            return true;
        }
    }

    return false;
}
```

7.18 Page processing and URLs

7.18.1 Overview

Kentico CMS processes URLs using a **URL rewriting engine**. This engine ensures the displaying of the correct page based on the required friendly (smart) URL. After the page is processed by the rewriting engine, it is run through output filters that ensure additional changes in the rendered HTML code.

The following figure shows the basic page processing steps:



7.18.2 URL rewriting

Kentico CMS uses a system of friendly URL addresses, which allows you to use URLs like:

http://www.example.com/products/kentico-cms.aspx

instead of something like:

http://www.example.com/products.aspx?id=527

These friendly or "smart" URLs provide several benefits:

- They are easy to remember and easy to write into the browser address bar.
- They are friendly towards search engines (for more information about this topic, please see the Search engine optimization chapter).
- They represent a path that shows users where they are located on the website.
- Users can easily send the URL of a document to their friends and they will see the same page with the specific document.

Every document has its own unique URL. If the website's content is available in multiple languages,

individual language versions of documents are recognized based on a combination of their **alias path** and a URL with a culture-specific format, or through completely custom **URL paths** (if specified).

URL processing

The steps below explain how the system handles a request for the following URL: *http://www.example.com/products/kentico-cms.aspx*

1. After receiving the incoming request, the system looks up the website based on the domain name in the URL (either via the main site domain name or its domain aliases). If none of the currently running websites match the domain name, the *~/cmsmessages/invalidwebsite.aspx* page is displayed instead. If the requested domain name contains a port number that is not found, the corresponding domain name without a port number is checked.

2. The request in this example specifies a standard URL path, so it is passed on to the URL rewriting engine. The most prioritized way to identify standard documents is their **Document URL path**, so the engine first attempts to find a document with a URL path set to */products/kentico-cms*.

3. If there is no document with a matching URL path, the rewriting engine then tries to look up a document with an **Alias path** equal to */products/kentico-cms* (the alias path is generated according to the document's position in the content tree).

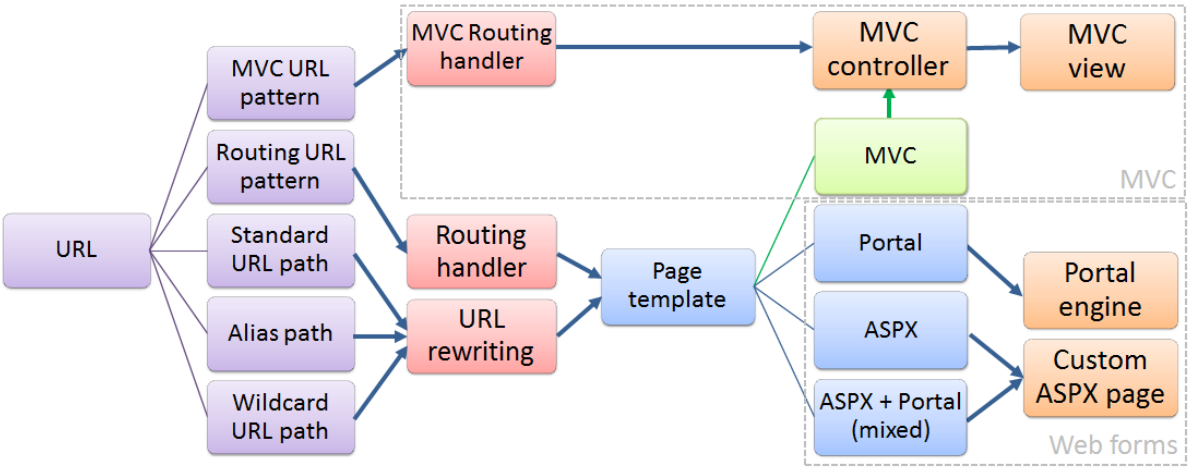
4. If a document cannot be found under the given URL or alias path, the system tries to find a document with a **Document alias** set to */products/kentico-cms*.

5. In case all of the steps described above fail to find a matching document for the requested URL, the system does not process the request. The web server returns a 404 HTTP status code and displays the Page not found error page configured for the website.

6. When the requested document is identified, the URL rewriting engine checks the type of its page template and handles it accordingly:

- If the page is managed by the Portal engine, the *~/cmspages/portaltemplate.aspx* system page is called, which renders the final output according to the web parts placed on the given template and the content of the particular document.
- For ASPX page templates, the appropriate template (web form) used to display the document is loaded using a specific internal URL, for example: */products.aspx?aliaspath=/products/kentico-cms*
- In the case of MVC templates, the system passes on the request to the MVC controller and action specified for the given template, which then renders the page using an appropriate MVC view.

The flow of the entire process for all types of URL requests is shown in the following diagram:



Processing of MVC and Routing URL patterns

If the URL of an incoming request matches the MVC or Routing pattern defined for a page, the pattern always has priority over other types of URL processing (even if it is set through a document alias).

The rewriting engine evaluates MVC and Routing patterns in the same step of the process, so precedence between the two is not defined. You should always avoid collisions between MVC and Routing patterns.

Excluding URLs from the rewriting engine

If your website uses custom physical pages (*.aspx*, *.html* or any other file types) stored inside the web project directory, you can avoid unnecessary processing by excluding their URLs from the Kentico CMS rewriting engine. When a visitor requests an excluded URL, the system skips all URL rewriting actions and attempts to access the specified page directly. This leads to better performance when loading the given pages and also allows you to prepare your own custom URL rewriting logic.

To disable rewriting for specific pages, go to **Site Manager -> Settings -> URLs and SEO** and enter the matching URL paths into the **Excluded URLs** setting.

- Use URL paths without the website's domain name or virtual directory.
- The paths must always start with a forward slash (/), without the virtual path designator (~).
- Entering a value excludes all URLs that start with the given path, including sub-directories and all possible extensions.
- You can enter multiple URLs separated by semicolons (;).

Sample values:

- **/Custom.aspx** - excludes the *~/Custom.aspx* page stored directly under the website's root.
- **/Custom** - excludes all pages whose URL path starts with */Custom*, for example: *~/Custom.aspx*, *~/Custom2.aspx*, *~/Custom/Page.htm*
- **/Custom;/Static** - excludes all pages whose URL path starts with */Custom* or */Static*.

Warning!

Be careful not to exclude the URLs used by the regular documents in the website's content tree. With URL rewriting disabled for a URL, the system always tries to load a matching physical page, which leads to a page not found error in most cases.

7.18.3 Multiple document aliases

In Kentico CMS, it is possible to have an unlimited number of different URLs leading to one document. These can be set in CMS Desk, on the **Properties -> URLs** tab of each document:

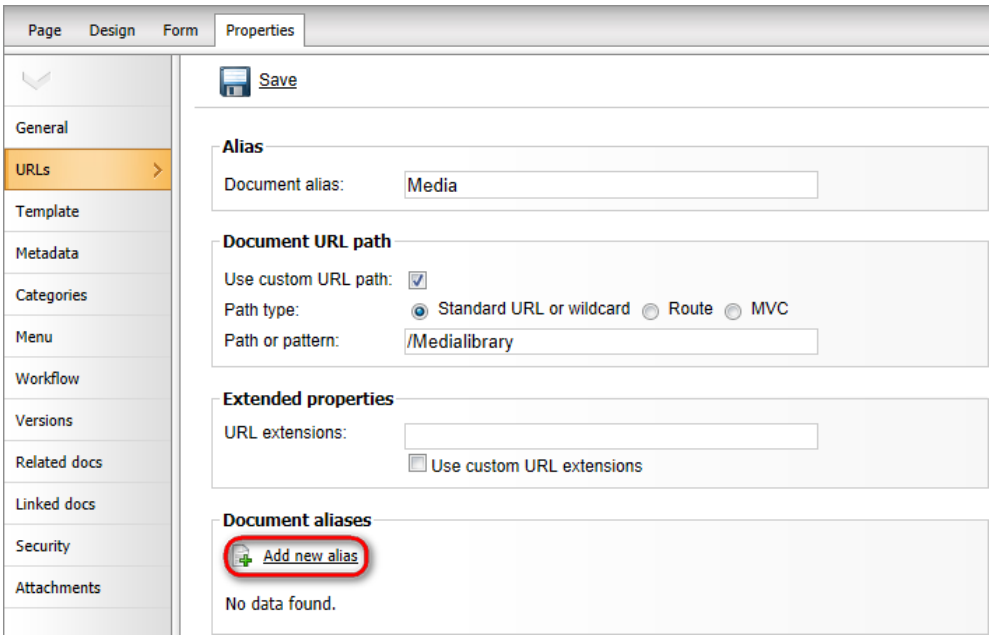
- **Document alias** - this is the unique name of the document in the given section of the website. Aliases are used to form the *alias path* of each document, which contains all parent documents up to the root of the site's content tree. The alias path of a document is then used to generate its default URL. For example, if *Media* is the alias of a document, then its URL would be: *<site domain>/<parent document alias path>/Media.aspx*

- **Document URL path** - the fields in this section may be used to specify an alternate URL for the document, regardless of its position in the website hierarchy. For example, if you have the *Standard URL or wildcard* Path type selected and enter */Medialibrary*, the URL of the document would be the following:
<domain>/Medialibrary.aspx

Further URLs can be added to a document through *document aliases*. Aliases do not change how the main URL of a document is generated, they only make it possible to access the given document through other URLs. Document aliases may be created in the following way:

1. Sign in to CMS Desk, select the appropriate document from the content tree and switch to its **Properties -> URLs** tab.

2. Click **Add new alias** in the **Document aliases** section.

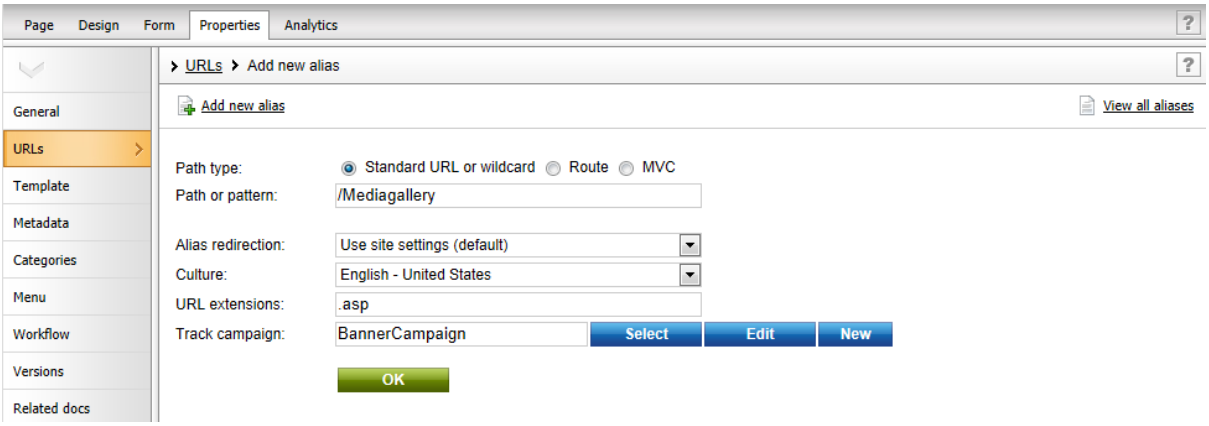


3. It is possible to fill in the following properties for each alias:

- **Path type -** you can choose several different URL types for the alias path. The selection made here determines how the value of the *Path or pattern* field will be processed. The following options are available:
  - **Standard URL or wildcard** - the alias URL will be handled by the standard Kentico CMS rewriting engine. Wildcard URLs can be used here, as described in Wildcard URLs.
  - **Route** - the alias URL will be processed as an ASP.NET Route pattern.
  - **MVC** - requests to the alias URL are handled as requests for an MVC page. See the MVC development model chapter for more information.
- **Path or pattern -** enter a URL path for the document alias according to the selected *Path type*. For example, if you enter */Mediagallery* as the value, the document will be accessible under the following URL: *<domain>/Mediagallery.aspx*
- **Default controller (MVC only)** - the name of the MVC controller containing the action that should be performed when the alias URL is accessed, without the *Controller* part at the end. For example, if the class is called *NewsMVCController*, enter *NewsMVC*. The system first searches for the specified class in the *CMS.Controllers.<current site code name>* namespace. If it is not found there, the *CMS.Controllers.Global* namespace is searched.
- **Default action (MVC only)** - specifies the action defined within the controller that should be performed when the alias URL is accessed.
- **Alias redirection** - determines how the URL should be handled (redirected) when the document is accessed through this alias (for SEO purposes). The following options are available:
  - **Use site settings (default)** - with this option, the redirection behavior will follow the *Redirect document aliases to main URL* setting specified for the website in *Site Manager -> Settings -> URLs and SEO*.
  - **Redirect to main URL** - if selected, the alias URL will always be redirected to the main URL of the document, i.e. its alias path or custom URL path (if specified).
  - **Do not redirect** - if selected, the alias URL will not be redirected when accessed. This option is recommended if the URL of the alias contains a wildcard.
- **Culture** - sets which culture version of the document should be displayed when the page is accessed through the URL of this alias.
- **URL extensions** - additional supported extensions of the URL. For these to work, you will have to

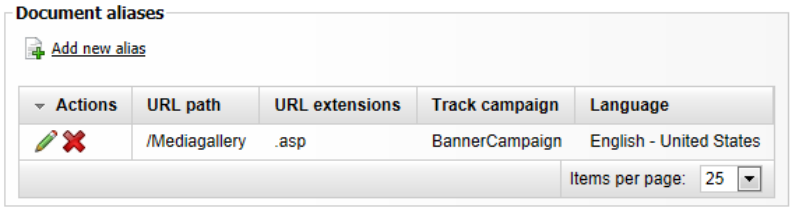
configure the system as described in Custom URL extensions and extensionless URLs. This field is optional.

- **Track campaign** - visitors who access the document through this alias will be assigned to the selected web analytics Campaign and tracked accordingly. For example, a special alias may be created for a document and its URL can then be used by links contained in marketing materials such as banners, e-mails etc. This way, the system will monitor how many page views the website receives as a result of the campaign and track the activity of the visitors who arrive as a result. This field is optional.

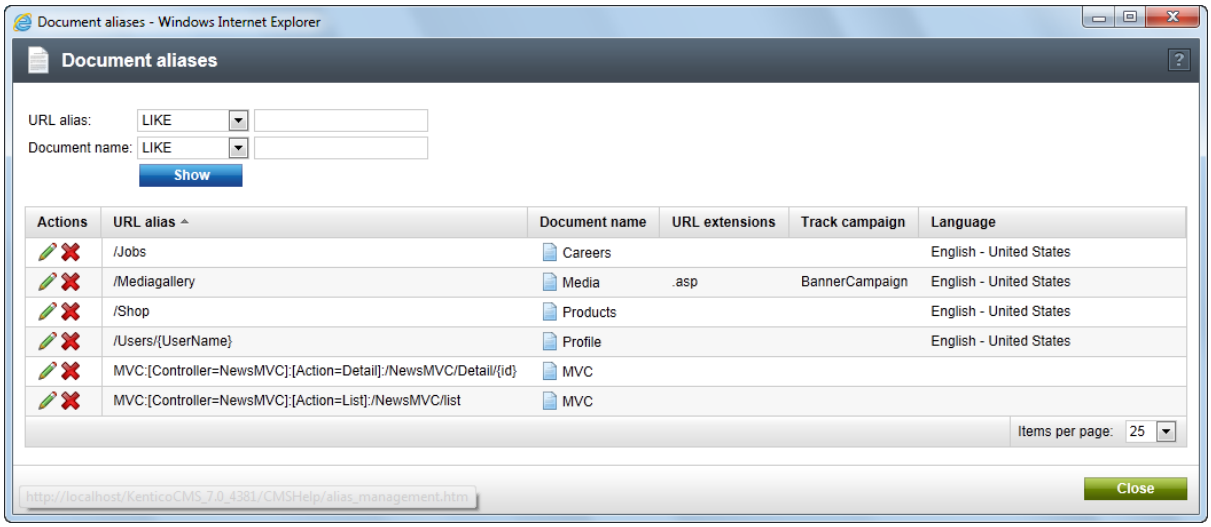


Click **OK** to create the alias.

4. Now if you switch back to the **Properties -> URLs** tab of the document, you should see the newly created alias present in the list in the **Document aliases** section, as depicted in the screenshot below. Like this, you can add an unlimited number of aliases.



5. If you edit (✎) any alias, you can use the **View all aliases** link at the top right of the page to access a list of all aliases defined on the website (for all documents). The information provided can help you avoid URL collisions when creating new aliases. The **Edit** (✎) and **Delete** (✖) actions are also available here, and through them you can easily manage all aliases and their properties from a single location.



7.18.4 URL format and configuration

Defining URL extensions

Page URLs can use various types of extensions. By default, all URLs end with *.aspx*, for example: *http://www.example.com/products/kentico-cms.aspx*

You can also use **custom extensions**, such as *.htm*, *.html* or any other sequence of characters. Alternatively, you can even use **URLs without extensions**, such as *http://www.example.com/products/kentico-cms*. In this case, you need to configure the system as described in the Configuration of custom URL extensions chapter.

Forbidden URL characters

Certain characters have special meanings when they are found in a URL, so they cannot be used in values that determine the path to pages (i.e. document aliases and URL paths). By default, the following characters are forbidden:

`\/:*?"<>|&%.'#[]+=„“` and the space character.

If needed, you can add additional forbidden characters by entering them (without any separator) into the **Forbidden URL characters** setting in **Site Manager -> Settings -> URLs and SEO**. Alternatively, a regular expression may be entered as the value of the **Allowed URL characters** setting to precisely specify which characters should be allowed in URLs.

Please note that the default characters listed above will always be forbidden unless you override them through the **CMSForbiddenURLValues** key, which can be added to the **/configuration/appSettings** section of your application's web.config file. For example:

```
<add key="CMSForbiddenURLValues" value="$\/:?&quot;&lt;&gt;|&amp;%.&apos;#[] =" />
```

Through this key, you may either allow some of the default forbidden characters or add new ones. It is recommended to keep the default characters forbidden, since they may prevent certain types of URLs

from working correctly if entered into URL paths.

Forbidden characters are automatically replaced or removed. You can specify the character that is used to replace forbidden characters through the **Forbidden characters replacement** setting in **Site Manager -> Settings -> URLs and SEO**. By default, forbidden characters located at the beginning or end of the path are removed completely and consecutive forbidden characters are only replaced by a single character. If you wish to have each forbidden character replaced individually, you can add the following key to your web.config:

```
<add key="CMSLimitUrlReplacements" value="false" />
```

This preference may also be set specifically for the **Document URL Path** property of documents (and no other URLs) via the key below:

```
<add key="CMSUseLimitReplacementsForUrlPath" value="false" />
```

Advanced character replacement customization

It is also possible to change the character replacement behavior to deal with specific scenarios, such as handling non-ASCII character sets or assigning unique replacement strings for specific characters.

This can be achieved by customizing the process used to convert URLs into a safe format. Please see Custom handling of URL path values for more information.

Using URL Prefixes

If you need to add a prefix to all URLs on a specific site (e.g. for search engine optimization), you can specify it in the **Site Manager -> Settings -> URLs and SEO -> Default Url Path Prefix** field. The URLs will then use the following format: *<domain>/<path prefix>/<document URL path>*

For example: *http://www.example.com/myprefix/products/kentico-cms.aspx*

Automatic creation of new document aliases

If you check the **Site Manager -> Settings -> URLs and SEO -> Remember original URLs when moving documents** check-box, new document aliases will automatically be created when a new extension or URL path is set.

Using language prefixes for URLs

If you wish to add a different URL prefix for every document culture version, you can specify it in the **Site Manager -> Settings -> URLs and SEO -> Use language prefix for URLs** field. If you have both language prefixes and standard URL path prefixes enabled, the standard URL prefix always precedes the language prefix in the URL. Please refer to the Multilingual and international support -> Languages and URLs topic for more details.

URL related settings

The above configuration tasks can be performed in **Site Manager -> Settings -> URLs and SEO**. The following table shows an overview of the available settings:

URL format	
Forbidden URL characters	List of additional characters that cannot be used in URLs (document aliases and URL paths). The following characters are forbidden by default: `\/:*?"<>|&%.'#[]+=„“` and the space character.
Forbidden characters replacement	Specifies the character that the system uses as a replacement for forbidden characters in URLs.
Allowed URL characters	Determines which characters are usable in URLs by means of a regular expression. Any characters not specified are forbidden. If empty, only the characters specified by the **Forbidden URL characters** setting are prohibited. When allowing special characters in the regular expression, they must be preceded by a backslash (\) as an escape character. **Example**: Entering *a-zA-Z0-9\^* as the value only allows alphanumeric characters and the caret symbol (^) to be used in URLs.
Friendly URL extension	Specifies the extensions that the system adds to document URLs. • Extensions must be preceded by the period character. • You can add multiple extensions separated by semicolons (;). • The first extension is used as the default option when generating links and page URLs. Additional extensions are supported in URLs when accessing documents. • To allow extensionless URLs, enter a semicolon without any extension. Sample value: *.aspx;.html;.htm;;*
Files friendly URL extension	Specifies the extension that the system adds to file URLs. **Example**: *getfile/<node alias>/myimage.aspx* If empty, the file URLs end with no extension: *getfile/<node alias>/myimage*
Excluded URLs	Specifies a list of URLs that are excluded from the URL rewriting engine. See Excluding URLs from the rewriting engine for more information.
Document URLs	
Default URL path prefix	Defines a default URL path prefix that will be used for all URLs of the content pages. Internally, this prefix is rewritten to the *urlpathprefix* query string parameter.
Use name path for URL path	If checked, a document's name path will automatically be copied to its

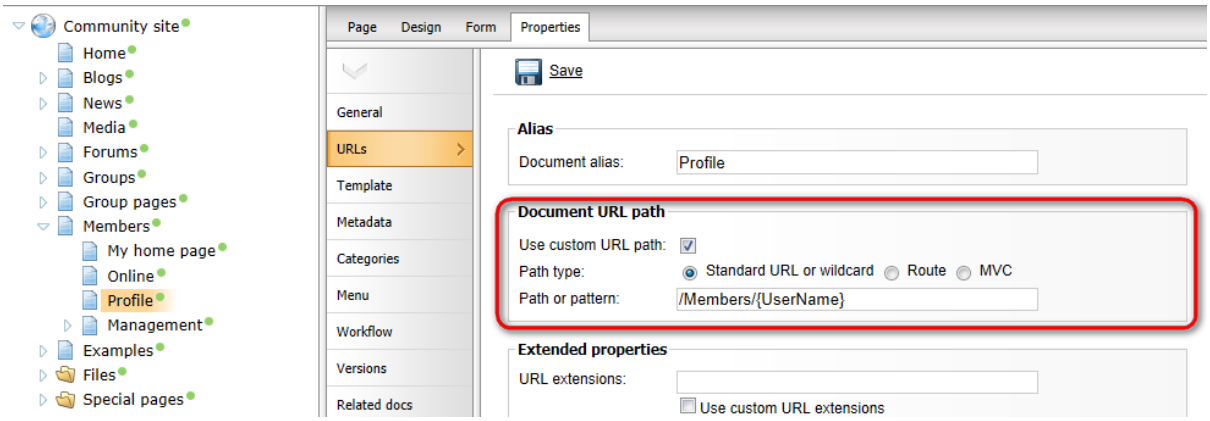
	URL path.
Use permanent URLs	If enabled, URLs of documents and document attachments will be generated in permanent format. If disabled, friendly URLs will be used. Learn more in the Linking pages and files topic.
Remember original URLs when moving documents	Determines if new document aliases should be created when a new document URL path or extension is set.
Automatically update document alias	If enabled, the alias of a document is automatically updated to match any changes in the name of the given document in the default culture. Also, the document alias property will not be editable manually.

Other settings (related to SEO) may also affect the format of URLs. To learn about these, please refer to the Search engine optimization chapter.

7.18.5 Wildcard URLs

Wildcard URLs provide a way of loading content dynamically depending on the page URL. You can find an example of how this works on the sample Community starter site. The **Members -> Profile** page uses wildcard URLs to display user profiles. As you can see, it is only one single page that can display profiles of various site users.

How is it achieved? If you go to **CMS Desk**, select the **Members -> Profile** page from the content tree and switch to its **Properties** tab, you should see **/Members/{UserName}** in the **Document URL path** field. The **{UserName}** part of the URL is the actual wildcard.



If you type ***<domain>/Members/David.aspx*** into your browser, the **Members -> Profile** page opens. The system converts the wildcard part of the URL (*David*) into a query string parameter, so that the internal URL actually looks like this: **<domain>/Members/Profile.aspx?username=David**. As you can see, the name of the parameter is taken from the name of the wildcard, while the value is the matching part of the entered URL. The User public profile web part which is placed on the page recognizes the *username* parameter in the rewritten URL and displays David's profile.

Default wildcard values

It is possible to set default values for wildcards in URLs. To do this, simply use the following format in the URL path: **{<wildcard name>;<Default value>}**. This is useful when you want to have a wildcard in a document's URL path, but expect that it might not always be entered into its URL. The default value

ensures that the URL is always generated with a certain value where the wildcard should be.

To show this on an example, select the **Members** page, switch to its **Properties -> URLs** tab and enter */Members/{UserName;David}* into the **Document URL path** field and click **Save**. If you now access the Members page on the live site, you will see that it automatically displays David's profile.

Additionally, the current value of a wildcard is always used as the default value in all other URLs on the given page that contain the same wildcard. For example, imagine a scenario with two documents that use a wildcard in their URL path: the first set to */Profile/{UserName}* and the other to */Profile/{UserName}/Details*. When the first page is accessed through the URL **<domain>/Profile/Andy.aspx**, navigation links to the second page will automatically be generated as **<domain>/Profile/Andy/Details.aspx**. The current value is used even for wildcards that have a different default value set in their URL path.

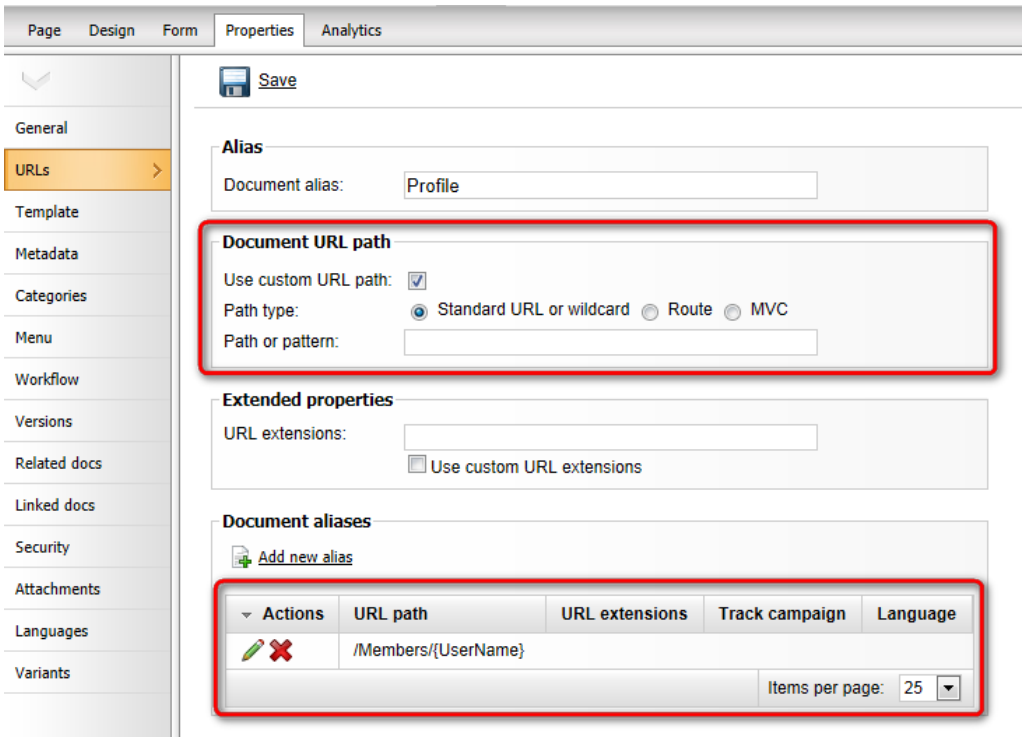
Using wildcard URLs on multi-language sites

The **Document URL path** is unique for each language version of a document. Because of this fact, you may encounter problems when referring to a page using a wildcard URL on multi-lingual sites. Let's explain the situation using the following example:

On the sample **Community Starter site**, the **Members/Profile** page has its Document URL path set to **/Members/{UserName}**. If you created a version of this page in another language, its Document URL path would get changed to **/Members/{UserName}-1** automatically. This happens because the URL path needs to be unique for every document.

Now let's presume that you have the following link leading to the page: **<domain>/Members/David.aspx**. In the original version, it works fine. But if you tried to click the link in the second language version, no profile would be found, because the URL in this language version would be **<domain>/Members/David-1.aspx**. The modification added to the end of the URL path changes the wildcard value representing the user name, which makes it impossible to find a matching user profile.

If you want to keep such links functional in all language versions, you will need to define the **/Members/{UserName}** path via the **Document aliases** section, as described in the Multiple document aliases topic. When creating the alias, select **(all)** in the **Culture** drop-down list. Before you can do this, you must erase the Document URL path value for both language versions of the document. As it has higher priority, URL paths defined in the Document aliases section would have no effect. The result should look as in the following screenshot:



Please keep in mind that it is not possible to specify *default* wildcard values through document aliases. An alias only makes the page accessible through a given URL, it does not enforce this URL for the document.

Dots in wildcard URLs

You may encounter problems when a string containing a dot "." gets into the wildcard part of a URL. A typical example of this can be found on the **Members -> Profile** page of the sample Community Starter Site.

The page's Document URL path is set to **/Members/{UserName}**. Let's presume that you have the following user name: **jack.smith**. Then the user's profile page would be located at **http://<domain>/Members/jack.smith**. As the last part of the URL after the last dot (*smith* in this case) is understood as a file extension, this URL would produce the 404 error in your browser.

To prevent this, registration on the sample Community Starter Site doesn't allow user names with dots. This is ensured through validation of the **UserName** field in the **Registration form** alternative form of the **User** system table.

If you need to allow dots in user names and use wildcard URLs with user names at the same time, you can do so by removing the validation and setting the page's Document URL path to something like **/Members/{UserName}/Profile**. In this case, the dot would be located in the middle of the URL and the URL should work fine.

7.18.6 Linking pages and files

Creating permanent links to documents (pages)

If you need to create a permanent link to a document, use a URL in the following format:

```
<domain>/getdoc/<document GUID>/<document name><extension>
```

- **<document GUID>** is the globally unique identifier of the document. You can find a document's GUID value in *CMS Desk -> Content -> Properties -> General -> Node GUID*.
- The **<document name>** value may contain any text — it is not used by the system, but allows you to specify the exact URL (e.g. for the purposes of search engine optimization). By default, the system uses the document's name.

Permanent links keep working even if you move the target document to another location.

For example:

http://www.example.com/getdoc/016fad52-0d69-46d5-80dc-daec9173c0c7/Products.aspx

is the permanent equivalent of:

http://www.example.com/company/products.aspx

Linking specific language versions of documents

If you need to link to a specific language version of a document, you use a URL in the following format:

```
<domain>/getdoc/<document GUID>/<document name>/<culture code><extension>
```

For example:

http://www.example.com/getdoc/8FG7-84E394-FABD-5678/our-services/fr-fr.aspx

Displays the given document in French (if the document is translated). It is an equivalent of:

http://www.example.com/company/our-services.aspx?lang=fr-fr

Linking attachments

If you need to create a permanent link to a file uploaded as a document attachment, you use a URL in the following format:

```
<domain>/getattachment/<file GUID>/<filename><extension>
```

- The **<file GUID>** is not the same as the document GUID. It is the GUID of the file in the *CMS_Attachment* database table. To find the GUID of an attachment, click the attachment in *CMS Desk -> Content -> Properties -> Attachments* and view the URL.
- The **<file name>** value can contain any text.

For example:

http://www.example.com/getattachment/763c8921-be94-4610-99b4-25e8d3be5b08/logo.aspx

You can find information about the query string parameters available for file URLs in GetFile.aspx parameters.

7.18.7 GetFile.aspx parameters

The system uses the **GetFile.aspx** page to retrieve uploaded files from the database. The page is called whenever you use /getdoc, /getattachment or a direct URL based on the alias path of a cms.file document.

GetFile URLs accept the following query string parameters:

Parameter name	Description	Sample value
guid	Attachment GUID value.	
nodeguid	Node GUID value.	
versionhistoryid	Version history ID of the attachment. It can only be used together with the **guid** parameter.	
width	Resizes the image to a specified width (in pixels).	100
height	Resizes the image to a specified height (in pixels).	400
maxsidesize	Resizes the image to the specified size of the longest side (in pixels).	500
disposition	Indicates the output disposition of the file. You can use one of the following disposition values: • **inline** - opens the file in the browser window if possible • **attachment** - opens the browser's "Save or Open" dialog	inline or attachment

Resizing of images

If your website has the **Resize images according to device profile** setting enabled in **Site Manager -> Settings -> Content -> Content management**, the system may override the **width**, **height** and **maxsidesize** parameters used in GetFile URLs of image files. When a user views the website on a device that matches a device profile, images automatically *reduce* their maximum side size to the larger of the dimensions set for the given profile.

To configure the dimensions of device profiles:

1. Go to **Site Manager -> Development -> Device profiles**.

2. Edit ( ) a profile.
3. Set the **Preview width** and **Preview height** properties.
4. Click **Save**.

If you wish to disable device profile resizing for an image, add the **resizemode=1** parameter to the GetFile URL.

7.18.8 Custom handling of URL path values

To ensure that URLs can be processed correctly, the system checks text values for unsafe characters before they are added to the URL path, and makes any necessary changes. This affects fields such as document aliases, document URL paths, forum names, post subjects etc.

By default, the process includes removal of diacritics and replacement of all forbidden characters according to the settings defined for the given site (as described in the URL format and configuration topic). Characters with diacritics are replaced by the appropriate base character, for example *ü* is converted to *u*.

If you need to change this default behavior in any way, you can customize it by registering and implementing a handler for one or both of the events described below:

- **OnBeforeGetSafeUrlPath** - occurs whenever a potentially unsafe text value is added to a part of a URL, e.g. when the document alias field is saved (or filled automatically based on the document's name). This event can be used to customize the final format of URL paths. For example, you can convert characters from an international character set to an equivalent in Latin characters (ASCII) or specify a unique replacement string for particular forbidden characters.
- **OnBeforeRemoveDiacritics** - occurs whenever the system removes diacritics from text. This is one of the default steps performed when creating safe URLs, but the event will also be triggered during many text parsing operations that are not directly related to URLs (e.g. when indexing text for the Smart search).

These events are members of the **URLHelper** and **TextHelper** classes respectively, which can be found under the **CMS.GlobalHelper** namespace.

The handlers for these events have the source text included as a string parameter passed by reference, so any changes made to it in the code will be reflected in the result.

Both handlers have a *boolean* return value that indicates whether the default functionality should also be performed after the handler is executed. For this reason, it is highly recommended to set the return value to *true* for all but the most extensive customizations.

Important!

Returning a *false* value should only be done if you are sure that your custom handler can take full responsibility for all URL safety or diacritics removal requirements.

Disabling the default system functionality may prevent parts of your website from working correctly, particularly in the case of handlers for the **OnBeforeGetSafeUrlPath** event.

Example

The following example demonstrates how you can define handlers for these events.

1. Open your Kentico CMS web project in Visual Studio, expand the **App_Code** folder (or **Old_App_Code** if your project was installed as a web application) and create a new class inside it. It may be placed in a sub-folder if you wish.

2. Edit the new class and add the following references:

[C#]

```
using CMS.SettingsProvider;
using CMS.SiteProvider;
using CMS.GlobalHelper;
```

3. Next, delete the default class declaration and its content. Instead, add the following code:

[C#]

```
[URLFormatHandlerLoader]
public partial class CMSModuleLoader
{
    /// <summary>
    /// Loader module registration.
    /// </summary>
    private class URLFormatHandlerLoaderAttribute : CMSLoaderAttribute
    {
        /// <summary>
        /// Called automatically when the application starts.
        /// </summary>
        public override void Init()
        {
            // Hooks a handler for the OnBeforeGetSafeURLPath event.
            URLHelper.OnBeforeGetSafeUrlPath +=
                new URLHelper.OnBeforeGetSafeUrlPathEventHandler
(Custom_OnBeforeGetSafeUrlPath);

            // Hooks a handler for the OnBeforeRemoveDiacritics event.
            TextHelper.OnBeforeRemoveDiacritics +=
                new TextHelper.OnBeforeRemoveDiacriticsEventHandler
(Custom_OnBeforeRemoveDiacritics);
        }

    }
}
```

This extends the **CMSModuleLoader** partial class and defines a new attribute for it. In a private class representing the attribute, you can override the **Init** method, which is executed automatically when the application starts. The method can then be used to assign custom handlers for the events.

4. Add the definition of the **Custom_OnBeforeGetSafeUrlPath** handler under the private class according to the code below:

[C#]

```
static bool Custom_OnBeforeGetSafeUrlPath(ref string url, string siteName,
EventArgs e)
{
    // Replaces all & characters with the word "and".
    url = url.Replace("&", "and");

    // Returns true to indicate that the default URL replacements should also be
performed (i.e. removing diacritics and forbidden characters).
    return true;
}
```

This code replaces all ampersand (&) characters with the word *and*. For example, this means that a document named *Products & Services* would have its default document alias generated as *Products-and-Services*, which would then be used in its URL. This example is only meant as a simple demonstration. In real-world scenarios, the handler could be much more complex, e.g. when mapping the character set of an international language to appropriate ASCII characters or words.

The **siteName** parameter of the handler contains the code name of the site under which the event occurred. This can be useful if you need to access the forbidden character settings of the given site in your custom code.

5. Next, add the **Custom_OnBeforeRemoveDiacritics** handler as shown below:

[C#]

```
static bool Custom_OnBeforeRemoveDiacritics(ref string text, EventArgs e)
{
    // Replaces German special characters.
    text = text.Replace( "ä", "ae" );
    text = text.Replace( "ö", "oe" );
    text = text.Replace( "ü", "ue" );
    text = text.Replace( "Ä", "Ae" );
    text = text.Replace( "Ö", "Oe" );
    text = text.Replace( "Ü", "Ue" );
    text = text.Replace( "ß", "ss" );

    // Returns true to indicate that the default diacritics removal should also be
performed.
    return true;
}
```

This ensures that the system will use custom replacements for German special characters rather than simply stripping the diacritics and leaving the base character.

Save the file. **Build** your project if it was installed as a web application. The changes will now be applied when creating new URLs and when removing diacritics from text. Please note that this will not

automatically change the aliases and URL paths of existing documents until their current value is changed or saved.

7.18.9 Output filters

The output filters are applied to the HTML code rendered by pages. They make various changes to the code before it is sent to the browser. Output filters do not affect the pages of the Kentico CMS administration interface (CMS Desk and Site Manager).

The following types of output filtering are available:

Form filter

The form filter fixes issue with non-working postbacks on pages that use URL rewriting. It ensures that forms, dialogs and buttons will work correctly on pages managed by Kentico CMS.

URL resolving filter

This filter resolves relative URLs so that they reflect the root URL of the website. For example, *~/mypage1/mypage2.aspx* would be changed to */mypage1/mypage2.aspx* (if the application is running in the root) or */VirtualDirectory/mypage1/mypage2.aspx* (when using a virtual directory).

Only URLs inside **src** and **href** attributes are changed.

XHTML filter

The XHTML filter may be used to fix certain types of XHTML incompatibilities. Specifically, the filtering functionality can ensure the following things:

- **Tag attributes** - the attributes of HTML tags are generated in valid XHTML format.
- **JavaScript tags** - the *type* and *language* attributes are included in all <*script*> tags.
- **Enforcing lower case** - all HTML tags and attributes are generated in lower case.
- **Fixing self closing tags** - all HTML elements without closing tags are properly closed, e.g. <*br*> will be replaced by <*br /*>.
- **Replacing invalid tags** - tags that are not XHTML valid are replaced by appropriate equivalents (<*strong*> instead of <*b*>, <*em*> instead of <*i*>).
- **Indentation** - the HTML code of pages is organized into a properly indented, easier to read format.

XHTML errors may also be fixed for the content entered into the WYSIWYG editor when it is saved. This can be configured globally by adding the **CMSWYSIWYGFixXHTML** key into the */configuration/appSettings* section of your application's *web.config* file as shown below:

```
<add key="CMSWYSIWYGFixXHTML" value="true" />
```

Supported values are *true* or *false*.

HTML 5 filter

The output filter provides a way to replace tag attributes that are obsolete in HTML5. Such attributes are removed and the system instead assigns CSS classes named in format <*attribute name*>_<*attribute*

value>. These classes need to be defined in the CSS stylesheet used by the website's pages.

The affected attributes are: *cellpadding*, *cellspacing*, *width*, *height*, *border*, *align*, *valign*

For example:

<*table cellpadding="2" cellspacing="4"*>

Would be replaced by:

<*table class="cellpadding_2 cellspacing_4"*>

Converting Tables to Div tags

The output filter may be used to automatically convert <*table*> elements and their child <*tr*> and <*td*> tags to <*div*> elements with appropriate CSS classes assigned (named according to the replaced tag). These classes need to be defined in the CSS stylesheet used by the website's pages.

For example:

```
<table>
  <tr><td>A</td><td>B</td></tr>
</table>
```

Would be replaced by:

```
<div class="table">
  <div class="tr">
        <div class="td">A</div>
        <div class="td">B</div>
  </div>
</div>
```

You may enable or disable this behavior for specific blocks of HTML code by marking them with *class="_divs"* or *class="_nodivs"* attributes. The filter may then be configured to convert only tables designated by the *_divs* class, or all tables except for those marked with *_nodivs* (see the descriptions of the settings below).

Output filter settings

If you do not wish to use the various types of output filters for specific sections of the website (e.g. due to performance reasons), you can disable them through the site settings in **Site Manager -> Settings -> System -> Output filter**.

Individual *Excluded URLs* fields are available for particular filter types. You can disable a certain type of output filter for all pages that start with a specific URL path by entering this path (without the ~ character and extension) into the corresponding field. Multiple URL paths may be added, separated by semicolons (;).

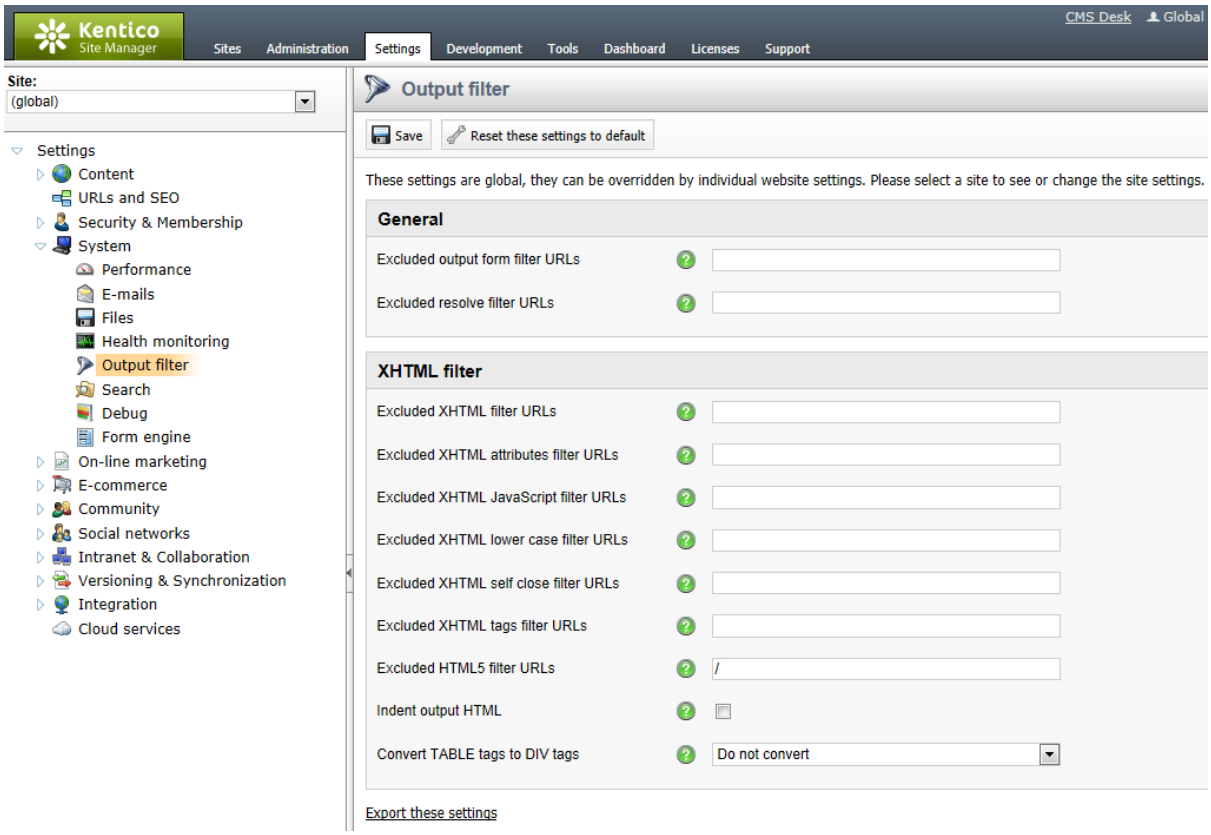
Please note that this only excludes specific URLs, not entire documents. For instance, if you exclude the */Home* URL path, but then access the Home page through a different URL or alias (such as the website root configured to display the Home page), output filtering will still be enabled.

Examples:

- **/** - disables the filter for the entire website.
- **/Products** - disables filtering for the Products page under the website root and all other pages whose URL path starts with */Products* (e.g. child documents).
- **/Services;/News** - excludes all pages whose URL path starts with */Services* or */News*.
- **/Company.** - by adding the period character (".") after the URL path, you can exclude only one specific page, but not its child documents (this will not work with extensionless URLs).

The following table describes all settings available in the Output filter category:

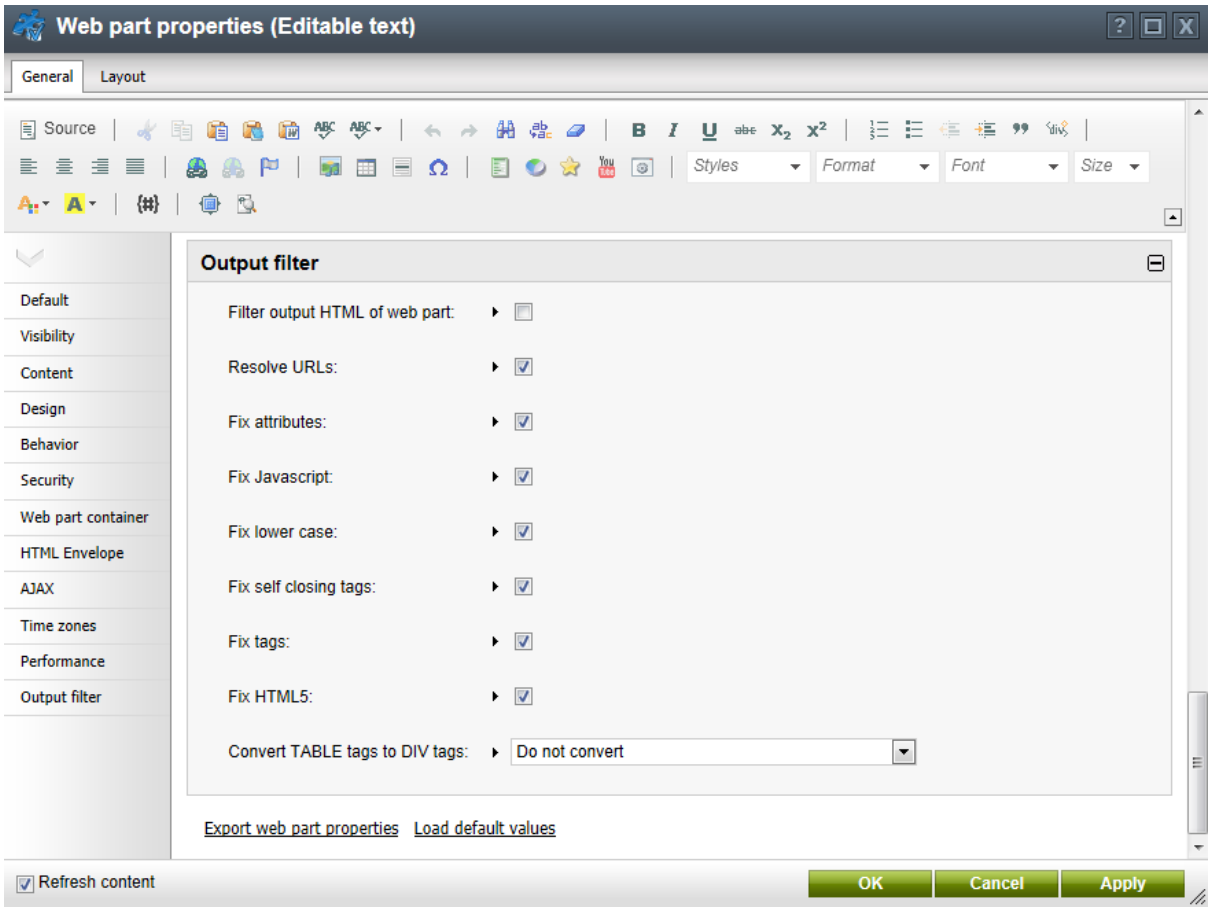
General	
Excluded output form filter URLs	Specifies the URLs of the pages that should be excluded from the Form output filter.
Excluded resolve filter URLs	May be used to disable the URL resolving output filter for specific URL paths.
XHTML filter	
Excluded XHTML filter URLs	Specifies the URLs that should be excluded from all functionality provided by the XHTML output filter.
Excluded XHTML attributes filter URLs	Specifies the URLs that should be excluded from the Tag attribute XHTML filter.
Excluded XHTML JavaScript filter URLs	Specifies the URLs that should be excluded from the JavaScript tag XHTML filter.
Excluded XHTML lower case filter URLs	Specifies the URLs that should be excluded from the Lower case XHTML filter.
Excluded XHTML self close filter URLs	Specifies the URLs that should be excluded from the Self closing tag XHTML filter.
Excluded XHTML tags filter URLs	Specify the URLs that should be excluded from the Invalid tag replacement XHTML filter.
Excluded HTML5 filter URLs	Specifies the URLs that should be excluded from the HTML5 output filter.
Indent output HTML	Indicates if the HTML output of all pages should be processed into a properly indented, easier to read format. This setting is applied to all pages on which the XHTML output filter is enabled.
Convert TABLE tags to DIV tags	Determines which tables should be converted to elements by the output filter. This behavior can either be disabled completely, enabled for all tables except for those marked by the **_nodivs** CSS class, or only enabled for the tables designated through the **_divs** class.



Enabling output filters for specific web part instances

You can enable the output filters separately for the code generated by Web parts. This allows you to filter the output of specific instances of web parts, even if their parent page is excluded from the output filter via the website settings. Output filtering is always used for web parts on pages that are not excluded.

To access these settings, configure ( ) the given web part on the **Design** tab of CMS Desk and adjust its **Output filter** properties. This section of properties is available for all web parts.



The configuration options correspond with the types of output filters described in the sections above.

7.18.10 Search engine optimization

7.18.10.1 SEO overview

Search engine optimization (SEO) is a process that attempts to improve the page rank of a website, which determines its organic position in the results of web search engines (such as Google). Being higher in these results benefits the site by naturally attracting more visitors.

While Kentico CMS cannot guarantee by itself that your website will have good search engine optimization, it provides many features that simplify related configuration tasks and make it easier to follow general best practices. Additionally, the system ensures a solid SEO foundation by generating pages with valid, standards compliant output code and URLs in a search engine friendly format.

Tracking search engine traffic

You can use the web analytics module to review the results of your website's SEO. It is possible to monitor user traffic gained from search engines and the activity of web crawlers on your site's pages can also be tracked.

> For more information, please refer to the Monitoring traffic from search engines topic in the **Modules -> Web analytics** chapter of this guide.

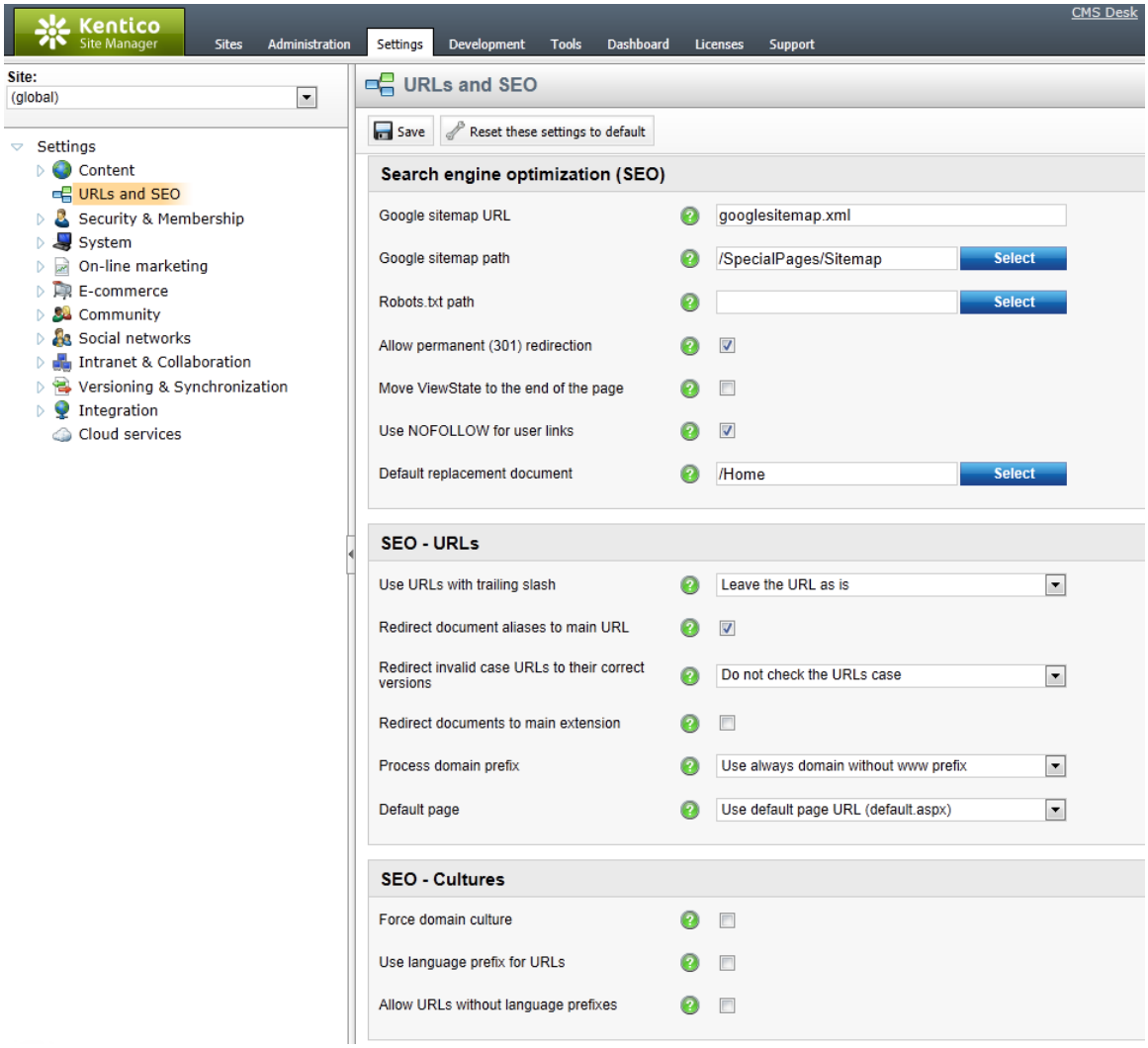
Editing the metadata of pages

An important SEO consideration is the metadata of individual pages, which you can edit by selecting the corresponding document in the content tree of CMS Desk and opening its **Properties -> Metadata tab**. Here you can set properties such as the page title or meta description.

See also: Content management -> Document properties -> Metadata

SEO settings

You can configure most of the SEO functionality of your website in the **Site Manager -> Settings -> URLs and SEO** interface. The settings here are divided into three main sections which are described below.



Search engine optimization (SEO)

Before search engines can provide results that link users to your website, they need to index the site using web crawlers (robots). The settings in this section allow you to use standard techniques for giving instructions to crawlers. You can also enable several options that help crawlers easily and accurately index the pages on your website.

Google sitemap URL	These two settings set up the URL of the website's Google (XML) sitemap and the path of the source document. The sitemap allows you to instruct web crawlers how to index the site's pages. Please see the Google Sitemaps topic to learn more about using sitemaps in Kentico CMS.
Google sitemap path	
Robots.txt path	Specifies the path of the document that generates the website's **robots.txt** file. Refer to Managing robots.txt for detailed information about setting up robots.txt content.
Allow permanent (301) redirection	If enabled, the system uses permanent (301) redirection instead of standard temporary (302) redirection. This is highly recommended, because it allows web crawlers to properly react to any changes made on your website and pass page rank to the new or main URL.
Move ViewState to the end of the page	If enabled, the system places the ViewState field at the end of the output code generated for pages. This helps search engine crawlers process more page content.
Use NOFOLLOW for user links	If enabled, the system instructs search engine crawlers not to follow links posted by users on Forums, Message boards or in Blog comments. This is achieved by including the *rel="nofollow"* attribute in the output code of all such link tags. This can prevent damage to your website's search ranking caused by user-generated links that lead to unrelated content. The setting can also help stop spammers from passing page rank to other sites.
Default replacement document	See the *Assigning replacement documents for deleted pages* section below.

Assigning replacement documents for deleted pages:

Removing documents can have a negative effect on your website's traffic. Visitors may find it confusing if pages suddenly become unavailable (e.g. if they have a deleted page bookmarked). Another thing to consider is that deleted pages might be indexed by search engines, which then provide invalid links in their search results until they re-index the site's content.

You can mitigate these problems by setting up other documents as replacements when deleting pages:

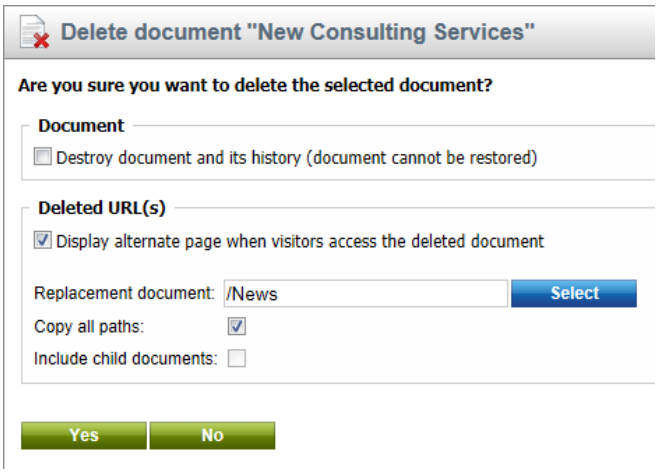
1. Select the document that you wish to remove in the CMS Desk content tree and click **Delete**.
   - The **Delete document** confirmation dialog opens.

2. Check the **Display alternate page when visitors access the deleted document** box.

3. Specify the path of the **Replacement document**.
   - The field loads its default value from the **Default replacement document** setting of the current website, but you can enter a different path for each specific case.

4. Enable or disable the following configuration options:

- **Copy all paths** - if checked, the website uses the replacement document for all possible URL paths of the deleted document (including any document aliases). If false, the replacement only covers the main URL of the deleted page, i.e. the corresponding document's custom URL path (if one is specified) or alias path.
- **Include child nodes** - if checked, the website uses the replacement document for all child documents removed along with the deleted page.



Setting a replacement document in the Delete document confirmation dialog

Once you confirm the deletion, the system automatically adds document aliases to the replacement document. These aliases ensure that the website serves up the replacement page whenever a visitor requests the URLs of the deleted document. When combined with permanent 301 redirection, the replacement aliases also properly inform search engine crawlers about the change in your website's structure.

SEO - URLs

The settings in this section allow you to configure URL rewriting that focuses on avoiding issues with duplicate content, i.e. having the same content available under multiple URLs. Such problems can have a negative effect on search ranking, so it is recommended to set up a unified URL format.

Use URLs with trailing slash	Specifies how the rewriter handles trailing slashes in URLs. Possible options: • Leave the URL as is • Always use URLs with a trailing slash • Always use URLs without a trailing slash
Redirect document aliases	Enabling this setting ensures that documents always have only one

to main URL	valid URL and other aliases are redirected to this main URL. The main URL of a document is determined either by its alias path, or custom URL path if one is specified. **Note**: You can override this setting for individual document aliases through their **Alias redirection** property. For more information about document URLs, see the Multiple document aliases topic.
Redirect invalid case URLs to their correct versions	Determines how the system handles the letter case of characters in URLs. Available options: • Do not check the URL case • Use the exact URL of each document • Redirect all requests to lower case URLs • Redirect all requests to upper case URLs
Redirect documents to main extension	If enabled, the system ensures that all document URLs use the current main extension. The main URL extension is the first one specified in the **Friendly URL extension** setting. Any URLs with a different extension are automatically redirected to a corresponding URL with the main extension.
Process domain prefix	Determines how the rewriter handles the *www* domain prefix in the website's URLs. You can leave the domain as it was entered or have it rewritten to either always or never include the www prefix. **Note**: This setting does not apply for IP addresses and top-level domains.
Default page	Allows you to redirect (permanent 301) all possible URLs that access the home page of your website to one single URL. Using a unified home page URL is highly recommended, because it prevents the duplicate content problem on your website's most important URL. You can choose from the following options for the home page URL: • **Not specified** - supports all possible home page URLs and does not perform any redirection. • **Use domain root** - always uses the base URL of the website's domain name. • **Use page defined by default alias path** - always uses the URL of the document specified by the website's *Content -> Default alias path* setting. • **Use default page URL** - always uses the *default* URL: *<domain>/default.aspx* **Important**: This setting only works correctly if the website is hosted on IIS 7 or newer, and uses an application pool with an *Integrated* Managed pipeline mode.

SEO - Cultures

The options in this section are important when performing search engine optimization of multilingual websites. The following settings allow you to set up unique URLs for different language versions of pages in a search engine friendly format.

Force domain culture	If checked, the system generates the domain name in document URLs based on the current content culture. Whenever a user switches to a different language on the website, the URL is redirected to the corresponding domain name. You can assign cultures to domains by editing your site in **Site Manager -> Sites**: • Set the culture of the website's main domain through the **Visitor culture** property on the **General** tab. • To define domain names for other languages, create **Domain aliases** with an appropriately set **Visitor culture**. **Note**: You cannot use this option in combination with language prefixes.
Use language prefix for URLs	If enabled, the system generates document URLs with language prefixes. A language prefix is a subdirectory inserted into the URL. The name of the prefix matches the culture code (or culture alias) of the content culture selected on the website. **Example**: *<domain>/en-US/Home.aspx*
Allow URLs without language prefixes	If enabled, URLs without language prefixes are allowed. Otherwise the system redirects such URLs to a corresponding URL that includes a language prefix. Only applies if **Use language prefix for URLs** is enabled.

See also:

- Languages and URLs
- Multilingual content

7.18.10.2 Google Sitemaps

Kentico CMS allows you to automatically generate sitemaps for your websites according to the Google Sitemap Protocol. Sitemaps help search engines correctly index the content of websites and can have a significant effect on the resulting search ranking.

A sitemap is an XML file that lists the URLs of a website's pages along with additional metadata. Search engine crawlers (robots) use the sitemap data to determine which pages to index and how often to re-index pages. Sitemaps only serve as a *recommendation* and do not guarantee that all crawlers will index your website strictly according to the specified data.

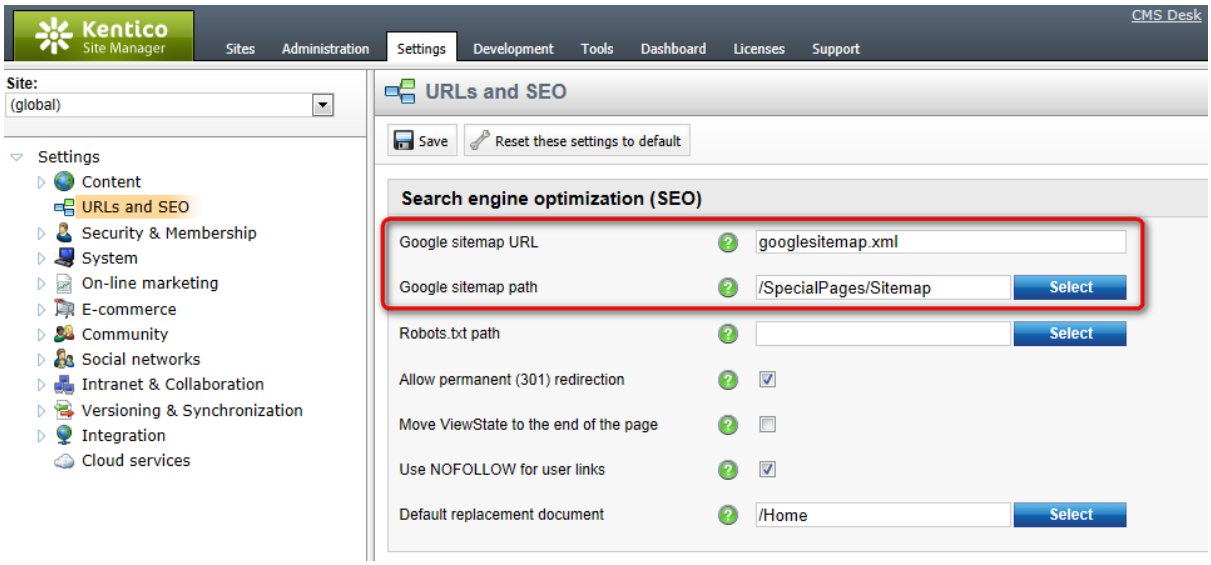
For detailed information about the Sitemap protocol, see http://www.sitemaps.org/.

Setting the sitemap URL

You can change the URL of your website's sitemap in **Site Manager -> Settings -> URLs and SEO** through the **Google sitemap URL** setting.

For example, the default value *googlesitemap.xml* means that web crawlers can access the sitemap through the following URL:

<website domain>/googlesitemap.xml



Sitemap-related website settings

Using the .xml extension

If you want to have your sitemap available under a URL with the **.xml** extension, you need to configure your application to handle all types of request extensions:

1. Edit your application's **web.config** file.
2. Find the **system.webServer** section directly under the web.config root (i.e. not under a specific *<location>* element).
3. Add the following attribute to the **<modules>** element:

```
<modules runAllManagedModulesForAllRequests="true">
```

Defining the sitemap content

The system generates sitemaps for websites based on the documents stored in the content tree.

By **default** the sitemap:

- Only contains pages (documents of the *CMS.MenuItem* type)
- Automatically excludes all pages whose parent document is not in the sitemap (such as pages stored under folders or custom document types)

You can **modify** the content of your website's sitemap by creating a dedicated sitemap document:

1. Add a new *Page (menu item)* document to your website's content tree.
   - You can use the predefined **SEO -> Google Sitemap** page template to quickly create sitemap documents. This template contains the required web part by default.

2. Place the Google Sitemap (XML Sitemap) web part onto the page.
   - Adding this web part stops the page from displaying standard content. Instead, the page returns an XML response with the sitemap data.
   - The web part only generates output when the page is accessed on the live site.

3. Configure the content of the sitemap through the web part's properties.
   - You can limit which documents are included in the sitemap by entering an appropriate **Path** expression.

4. Go to **Site Manager -> Settings -> URLs and SEO**.

5. Enter the path of your sitemap document into the **Google sitemap path** setting.

The sitemap generated according to the configuration of the web part replaces the default sitemap. Search crawlers can access the sitemap either under the main URL specified in the **Google sitemap URL** setting, or directly through the URL of the document containing the *Google Sitemap* web part.

Customizing the default sitemap directly

If you do not wish to use portal engine pages and web parts, you can instead edit the markup of the *~/CMSPages/googlesitemap.aspx* system page. This page generates the default sitemap for websites that have an empty **Google sitemap path** setting.

The **GoogleSitemap** control on the page provides the same configuration options as the *Google Sitemap* web part.

Troubleshooting:

If you encounter problems with pages missing in your sitemap, try checking for the following:

- **Manually excluded documents** - specific documents may be excluded through their sitemap properties (**Show in sitemap** or **Exclude from search**).

- **Incorrect content filtering** - review the content filtering properties of your *Google Sitemap* web part.
  - If the **Document types** property is empty, the sitemap only loads pages (*CMS.MenuItem* documents). Add the document types that you wish to have in the sitemap. You can use the asterisk (*) wildcard to specify all document types.

- **Broken document hierarchy** - sections of the website may be excluded due to parent documents

missing in the sitemap. To load all documents regardless of the parent-child hierarchy in the content tree, disable the **Hide children for hidden parent** property of the *Google Sitemap* web part.

Configuring sitemap settings for specific pages

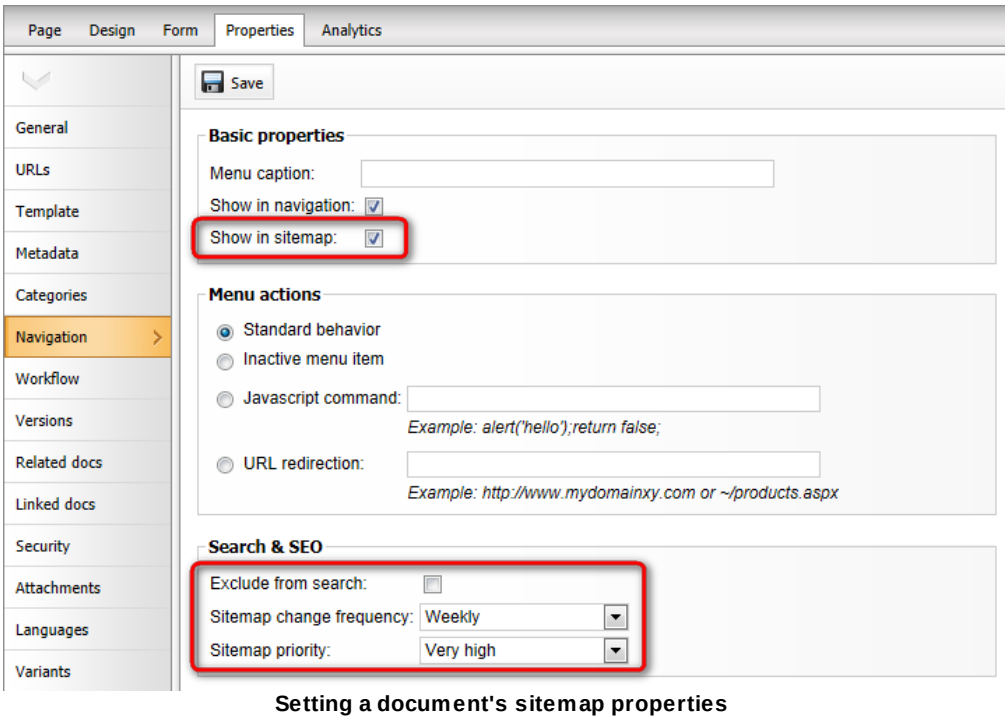
By filling in the sitemap properties of documents, you can exclude specific pages from sitemaps or give search crawlers additional details describing how to index pages:

1. Select the document in the content tree of CMS Desk.

2. Open the document's **Properties -> Navigation** tab.

3. Set up the following properties:

Basic properties	
Show in sitemap	Sitemaps only list documents that have this property enabled.
Search & SEO	
Exclude from search	Marks the document to be ignored by all forms of search, including search engines. Enabling this checkbox excludes the document from sitemaps by default. However, individual *XML Sitemap* web parts can override this setting and generate sitemaps including documents that are excluded from search.
Sitemap change frequency	Determines the value of the document's **<changefreq>** tag in the sitemap. This metadata provides a suggestion to search engines about how often they should re-index the page. Choose a value that reflects how frequently the page's content changes.
Sitemap priority	Allows you to inform web crawlers which pages you consider to be the most important. The system converts the selected priority to a decimal number between 0 and 1 and adds the number as the value of the document's **<priority>** tag in the sitemap. Web crawlers only measure the priority in relation to other pages on the website.



Setting a document's sitemap properties

If you enter the URL of the sitemap into your browser, you can review the generated XML output. The system automatically creates the required XML structure:

```
<?xml version="1.0" encoding="UTF-8"?>
- <urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9" xsi:schemaLocation="http://www.sitemaps.org/schemas/sitemap/0.9
http://www.sitemaps.org/schemas/sitemap/0.9/sitemap.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  - <url>
      <loc>http://localhost/KenticoCMS_7.0.4456/Home.aspx</loc>
      <lastmod>2012-03-20</lastmod>
      <changefreq>weekly</changefreq>
      <priority>0.9</priority>
    </url>
  - <url>
      <loc>http://localhost/KenticoCMS_7.0.4456/Products.aspx</loc>
      <lastmod>2012-03-20</lastmod>
    </url>
  - <url>
      <loc>http://localhost/KenticoCMS_7.0.4456/Products/Smartphones.aspx</loc>
      <lastmod>2012-02-17</lastmod>
    </url>
  - <url>
      <loc>http://localhost/KenticoCMS_7.0.4456/Products/Laptops-and-Tablets.aspx</loc>
      <lastmod>2011-06-23</lastmod>
    </url>
```

XML output of the Google sitemap generated for a Kentico CMS website

- The **<url>** elements represent individual pages.
- The sitemap loads the values of the **<loc>** and **<lastmod>** tags from the data of the corresponding documents.
- The **<changefreq>** and **<priority>** optional tags are added for documents that have values in their **Sitemap change frequency** and **Sitemap priority** properties.

Creating sitemap indexes

A single XML sitemap can only list up to 50 000 pages (URLs). If you need to include more pages, prepare multiple sitemaps and create a sitemap index for your website:

1. Add any number of sitemap documents, each one containing its own Google Sitemap web part.

2. Separate your website's pages between the sitemaps by configuring the content filtering properties of the web parts.
   - Each sitemap can contain a maximum of 50 000 items.
   - Avoid duplicate content — do not list the same page URLs in multiple sitemaps.

3. Create the index as another document with a *Google Sitemap* web part.
   - Switch the **Sitemap mode** property of the web part to *Sitemap index.*
   - Configure the content filtering properties so that the sitemap index web part loads only the documents representing your sitemaps.

4. Enter the path of your *sitemap index* document into the website's **Google sitemap path** setting.
   - This ensures that crawlers process the sitemap index first.

The sitemap index points search engine crawlers to the other sitemaps, which then provide the lists of page URLs in the usual way.

Customizing the XML format of sitemaps

If you need to override the default XML format of a sitemap or index, you can specify a custom transformation for the Google Sitemap web part or *GoogleSitemap* control. This allows you to react to any changes in the Sitemap protocol.

For example, the default **CMS.Root.GoogleSiteMap** transformation uses the following code to define the sitemap structure:

```
<url>
  <%# GetSitemapItem("loc") %>
  <%# GetSitemapItem("lastmod") %>
  <%# GetSitemapItem("changefreq") %>
  <%# GetSitemapItem("priority") %>
</url>
```

The **GetSitemapItem** transformation method generates XML tags according to the sitemap protocol. The method's parameter specifies the type of the tag, and the value is dynamically loaded from the data of the transformed documents.

In the final XML output, the web part automatically encloses the transformed items within either a <*urlset*> or <*sitemapindex*> element depending on the selected **Sitemap mode**.

7.18.10.3 Managing robots.txt

You can give instructions to web crawlers and other robots using the Robots Exclusion Protocol, i.e. a **robots.txt** file. The primary purpose of robots.txt files is to exclude certain pages from search engine indexing. Like with Sitemaps, the provided instructions are only considered as recommendations and may be ignored by some robots.

Creating a robots.txt file for your website

The most direct way to use robots.txt with Kentico CMS is to physically add the text file into the root of your web project. However, this scenario does not allow you to assign different robots.txt files to specific websites (if there are multiple sites running on your installation). Additionally, it may be difficult to access the file system in certain types of hosting environments.

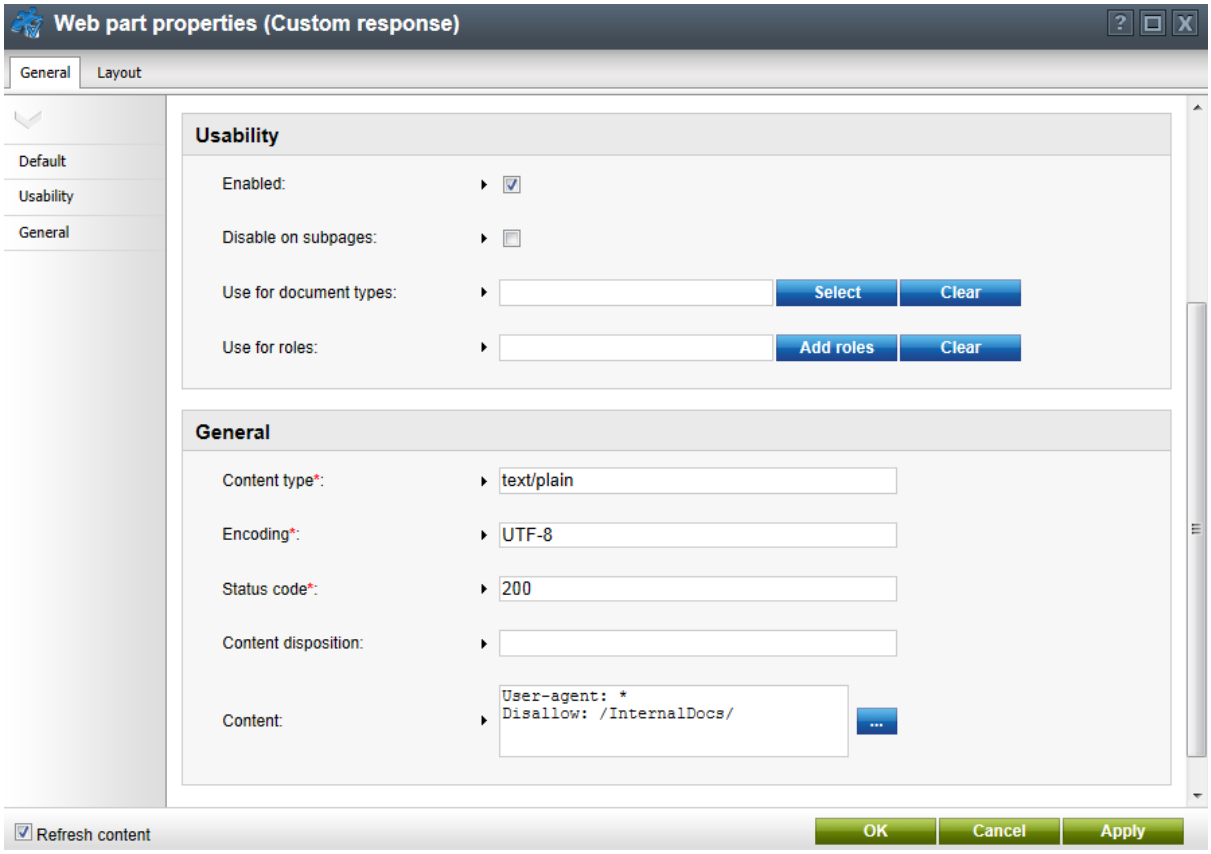
The recommended approach is to create a dedicated document in your site's content tree and make it return the appropriate text response:

1. Create a standard *Page (menu item)* document.
   - You can use the predefined **SEO -> Robots.txt** page template to quickly implement robots.txt documents.

2. Add a Custom response web part to the page.

3. Configure ( ) the *Custom response* web part to generate a valid robots.txt response according to the following steps:

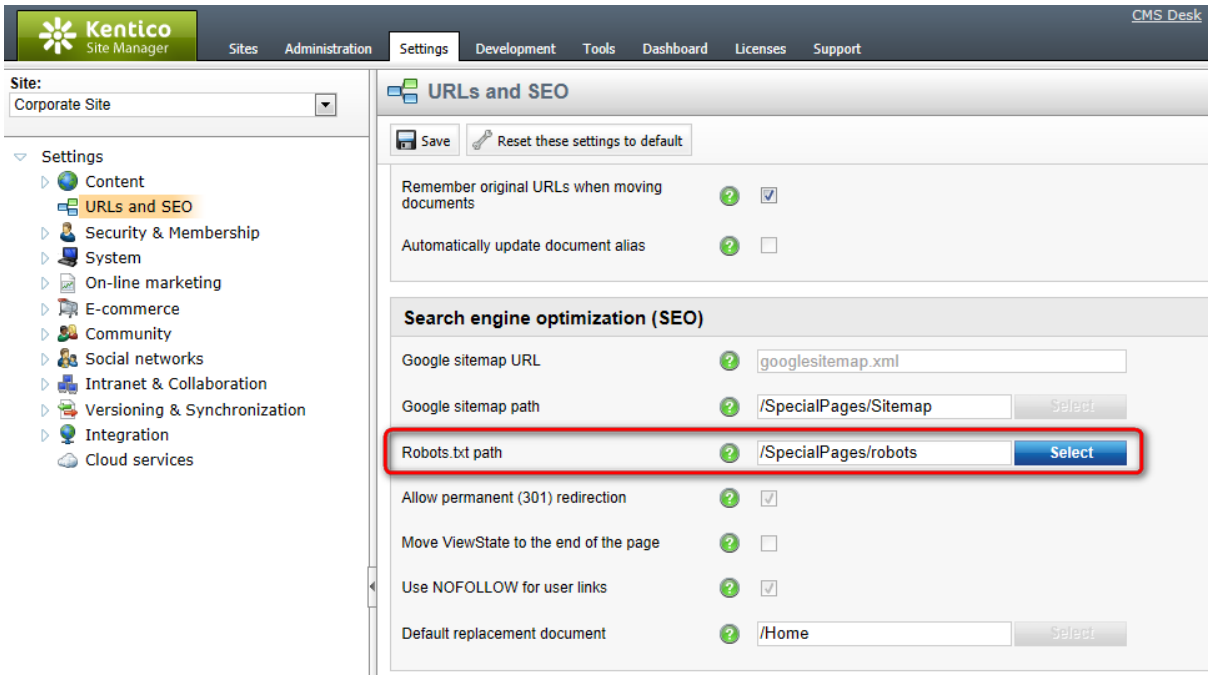
   a. Set the **Content type** property to *text/plain*.
   b. Enter an appropriate **Encoding** type, for example *UTF-8*.
   c. Set the **Status code** of the response to *200*.
   d. Add the actual robots.txt instructions into the **Content** property, just like you would in a physical text file. This property supports K# macro expressions, so you can dynamically load values from the current system data if needed.



4. Go to **Site Manager -> Settings -> URLs and SEO** and enter the path of your robots.txt document

into the **Robots.txt path** setting.

- You can specify a different value for each site by using the **Site** selector above the settings tree.



The output of the specified document is always available under the standard *<website domain>/robots.txt* URL, regardless of the document's location in the content tree. Compliant web crawlers read the instructions from this URL before processing other pages on the website.

Enabling the .txt extension

To ensure that the *<domain>/robots.txt* URL is available, you need to configure your application to handle all request extensions:

1. Edit your application's **web.config** file.
2. Find the **system.webServer** section directly under the web.config root (i.e. not under a specific *<location>* element).
3. Add the following attribute to the **<modules>** element:

```
<modules runAllManagedModulesForAllRequests="true">
```

Excluding documents manually

You can also configure individual documents to be excluded from search engine listings without the need to prepare a robots.txt file.

1. Select the given document in the CMS Desk content tree.
2. Open the **Properties -> Navigation** tab.
3. Enable the **Exclude from search** property.
4. Click **Save**.

The system automatically adds the following meta tag to the <*head*> section in the HTML output of such pages:

```
<meta name="robots" content="noindex,nofollow" />
```

This instructs web crawlers not to index the page and to ignore any links in the content.

7.18.11 Document aliases internals and API

7.18.11.1 Overview

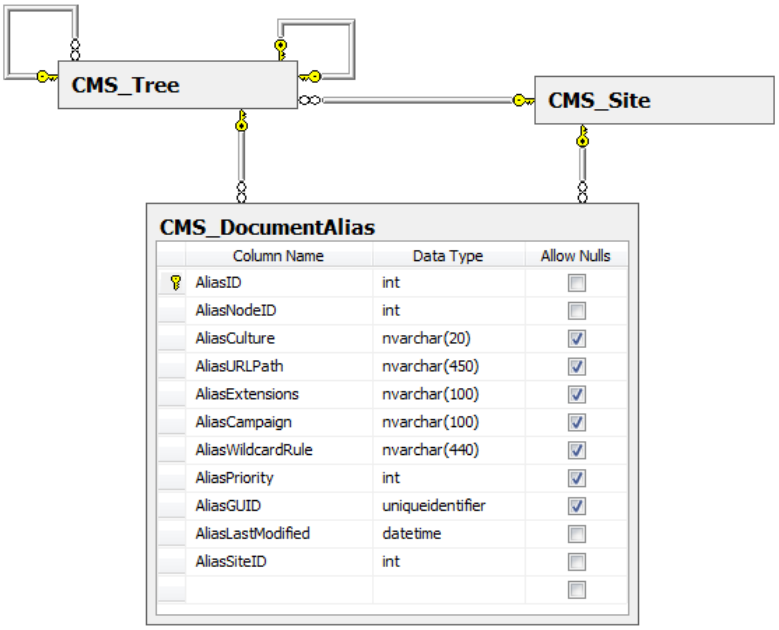
In this chapter, you will learn which database tables and API classes are used to manage document aliases. You will also see the most common API examples.

Further API-related information and examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all members of the related classes, please refer to Kentico CMS API Reference.

7.18.11.2 Database tables

The following database tables are used to store information about document aliases:

- **CMS_DocumentAlias** - contains records representing aliases of documents from the content trees of the sites in the system.



7.18.11.3 API classes

If you are not familiar with database table data management through Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

The classes for managing document aliases can be found in the **CMS.DocumentEngine** namespace.

CMS_DocumentAlias table API:

- **DocumentAliasInfo** - represents one document alias.
- **DocumentAliasInfoProvider** - provides management functionality for document aliases.

7.18.11.4 API examples

7.18.11.4.1 Overview

These topics show examples of how the API for managing document aliases can be used:

- Managing document aliases

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Documents\DocumentAliases\Default.aspx.cs***.

The document alias API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.GlobalHelper;
using CMS.UIControls;
using CMS.CMSHelper;
using CMS.DocumentEngine;
```

7.18.11.4.2 Managing document aliases

The following example creates a document alias.

```
private bool CreateDocumentAlias()
```

```
{
    // Get "Home" document
    TreeNode document = TreeHelper.GetDocument(CMSContext.CurrentSiteName, "/
Home", CultureHelper.GetPreferredCulture(), true, "CMS.MenuItem", false);

    if (document != null)
    {
        // Create new document alias object
        DocumentAliasInfo newAlias = new DocumentAliasInfo();

        // Set the properties
        newAlias.AliasURLPath = "/MyNewAlias";
        newAlias.AliasNodeID = document.NodeID;
        newAlias.AliasSiteID = CMSContext.CurrentSiteID;

        // Save the document alias
        DocumentAliasInfoProvider.SetDocumentAliasInfo(newAlias,
CMSContext.CurrentSiteName);

        return true;
    }

    return false;
}
```

The following example gets and updates a document alias.

```
private bool GetAndUpdateDocumentAlias()
{
    // Prepare the parameters
    string orderBy = "";
    string where = "AliasURLPath = N'/MyNewAlias'";

    // Get the data
    DataSet aliases = DocumentAliasInfoProvider.GetDocumentAliases(where,
orderBy);
    if (!DataHelper.DataSourceIsEmpty(aliases))
    {
        DocumentAliasInfo updateAlias = new DocumentAliasInfo(aliases.Tables[0]
.Rows[0]);

        // Update the properties
        updateAlias.AliasURLPath = updateAlias.AliasURLPath.ToLower();

        // Save the changes
        DocumentAliasInfoProvider.SetDocumentAliasInfo(updateAlias,
CMSContext.CurrentSiteName);

        return true;
    }

    return false;
```

```
}
```

The following example gets and bulk updates document aliases.

```
private bool GetAndBulkUpdateDocumentAliases()
{
    // Prepare the parameters
    string orderBy = "";
    string where = "AliasURLPath = N'/MyNewAlias'";

    // Get the data
    DataSet aliases = DocumentAliasInfoProvider.GetDocumentAliases(where,
orderBy);
    if (!DataHelper.DataSourceIsEmpty(aliases))
    {
        // Loop through the individual items
        foreach (DataRow aliasDr in aliases.Tables[0].Rows)
        {
            // Create object from DataRow
            DocumentAliasInfo modifyAlias = new DocumentAliasInfo(aliasDr);

            // Update the properties
            modifyAlias.AliasURLPath = modifyAlias.AliasURLPath.ToUpper();

            // Save the changes
            DocumentAliasInfoProvider.SetDocumentAliasInfo(modifyAlias,
CMSContext.CurrentSiteName);
        }

        return true;
    }

    return false;
}
```

The following example deletes a document alias.

```
private bool DeleteDocumentAlias()
{
    // Prepare the parameters
    string orderBy = "";
    string where = "AliasURLPath = N'/MyNewAlias'";

    // Get the data
    DataSet aliases = DocumentAliasInfoProvider.GetDocumentAliases(where,
orderBy);
    if (!DataHelper.DataSourceIsEmpty(aliases))
    {
        DocumentAliasInfo deleteAlias = new DocumentAliasInfo(aliases.Tables[0]
.Rows[0]);
```

```
        // Delete the document alias
        DocumentAliasInfoProvider.DeleteDocumentAliasInfo(deleteAlias);

        return true;
    }

    return false;
}
```

7.19 Query string parameters in UI

Query string is a part of a URL that contains parameters and values to be passed to a web application. The query string part of the URL begins with the "?" character followed by the name of the parameter (e.g., culture), the "=" character and a value (e.g., en-US). Individual parameters with values are separated by the "&" character.

In Kentico, you can use the query strings for example for:

- Linking to documents
- Creating links to specific sections of the editing UI

Available query string parameters in the Kentico CMS editing user interface

In Site Manager:

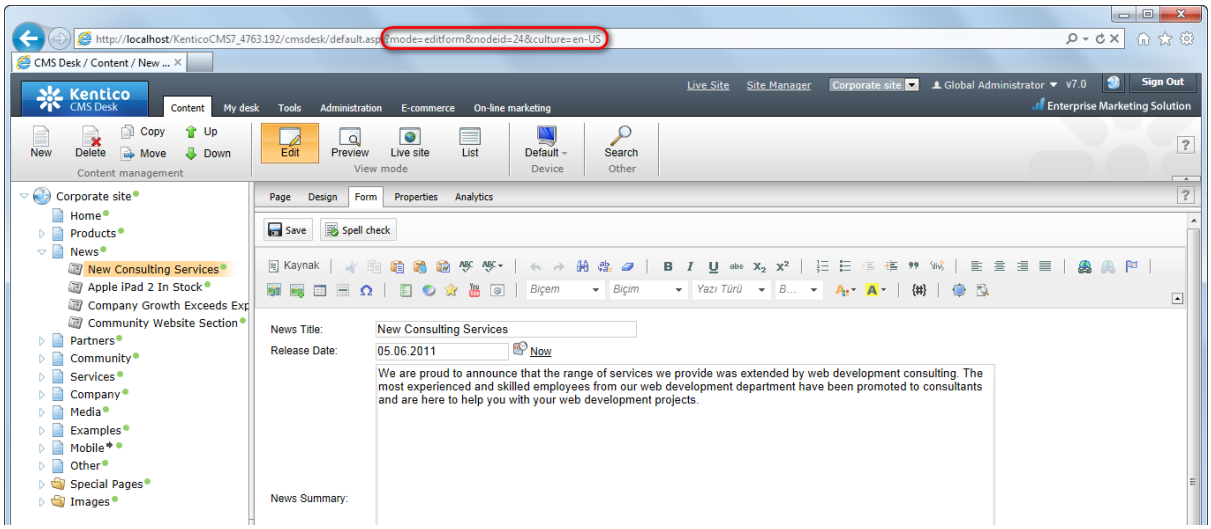
Parameter	Description	Values	Example
section	Selects the section (main tab) in Site Manager.	Code names of Site Manager sections	http://localhost/KenticoCMS/CMSSiteManager/?section=development

In CMS Desk:

Parameter	Description	Values	Example
section	Selects the section (main tab) in CMS Desk.	Code names of CMS Desk sections	http://localhost/KenticoCMS/CMSDesk/?section=tools
nodeid	Selects the document with the specified ID in the currently selected culture.	A node ID number (displayed in CMS Desk -> Edit -> Properties -> General)	http://localhost/KenticoCMS/CMSDesk/?nodeid=24
culture	Changes the current culture. Use in a combination with the *nodeid* parameter.	Four-letter country codes	http://localhost/KenticoCMS/CMSDesk/?nodeid=7&culture=cs-CZ
mode	Selects the View mode or subsections of the Edit mode (Page, Design, Properties, etc.).	*edit, preview, livesite, listing, page, wireframe, design, editform, product, masterpage, properties, analytics*	http://localhost/KenticoCMS/CMSDesk/?mode=livesite http://localhost/KenticoCMS/CMSDesk/?nodeid=4&mode=editform
device	Sets the device profile.	Code names of device profiles (displayed in Site Manager -> Development -> Device profiles -> Edit () a device)	http://localhost/KenticoCMS/CMSDesk/?device=android_device
expandnodeid	Only expands the node with the chosen ID.	A node ID number (displayed in CMS Desk -> Edit -> Properties -> General)	http://localhost/KenticoCMS/CMSDesk/?expandnodeid=7

You can also combine these parameters in various ways, for example:

- **http://localhost/KenticoCMS7/CMSDesk/default.aspx?mode=editform&nodeid=24&culture=en-US** - which selects the node with ID = 24 in the US culture and switches into the Form tab of the Edit mode.



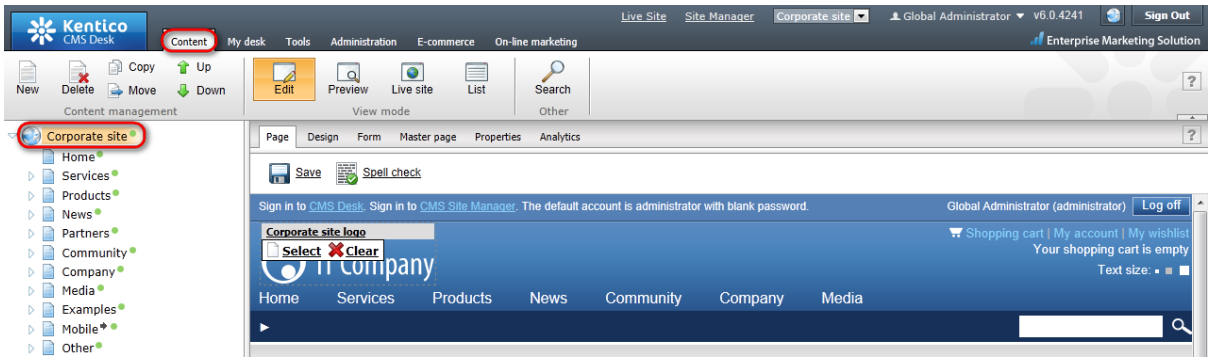
Example of a query string

7.20 Rebranding

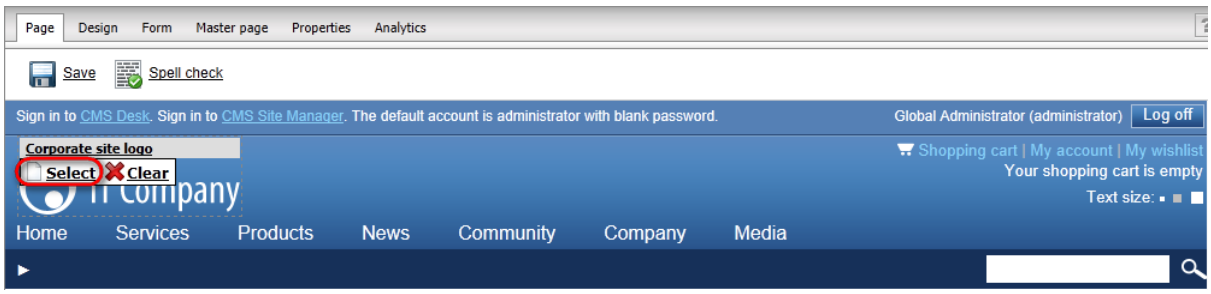
7.20.1 Changing the logo in the header

In this topic, you will learn how to change the header logo on the sample Corporate Site.

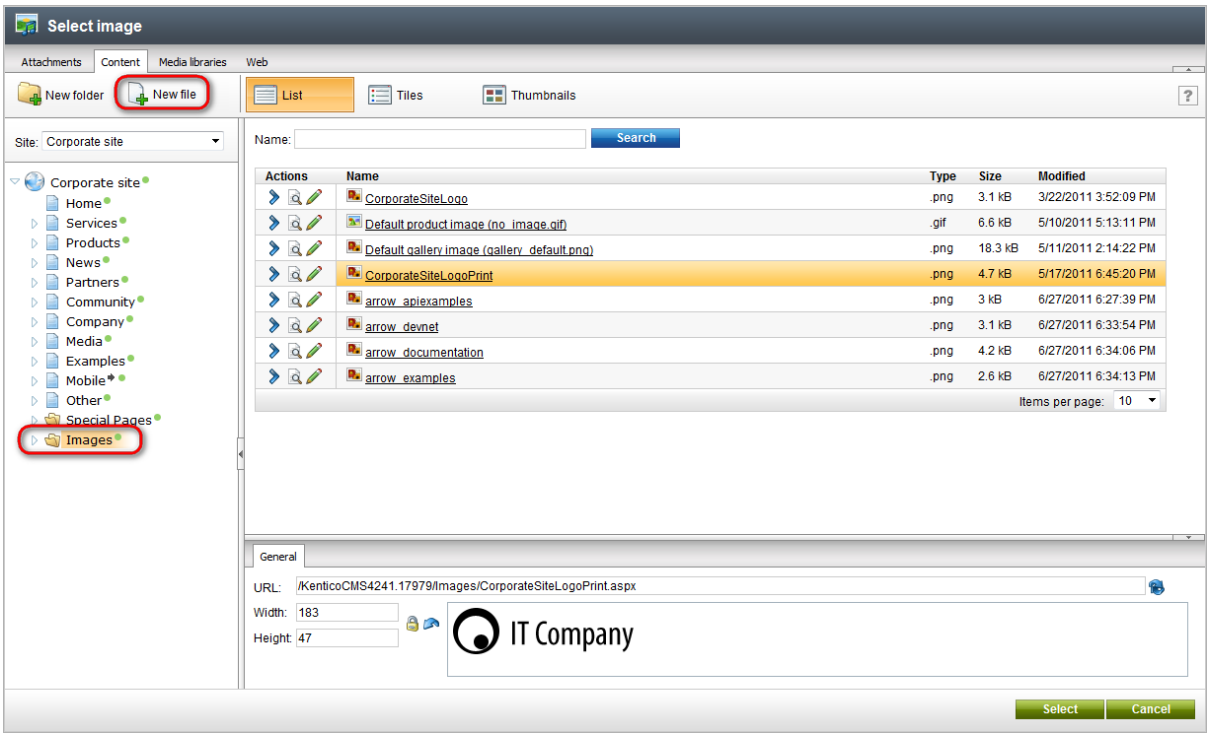
1. Go to **CMS Desk** -> **Content** and select the root of the content tree (**Corporate Site**).



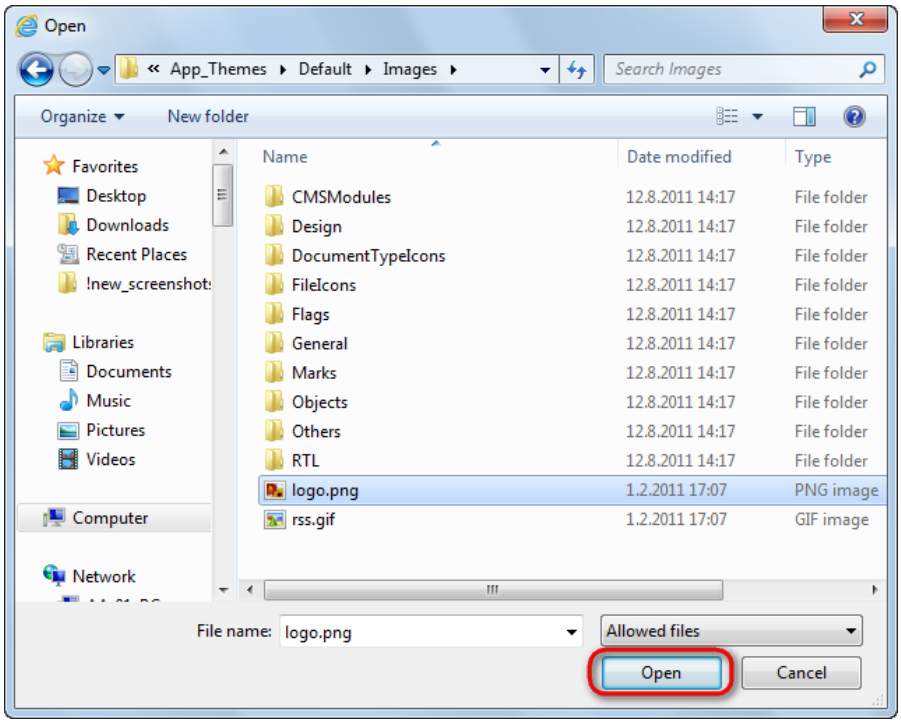
2. On the **Page** tab, click the **Select** button displayed over the logo.



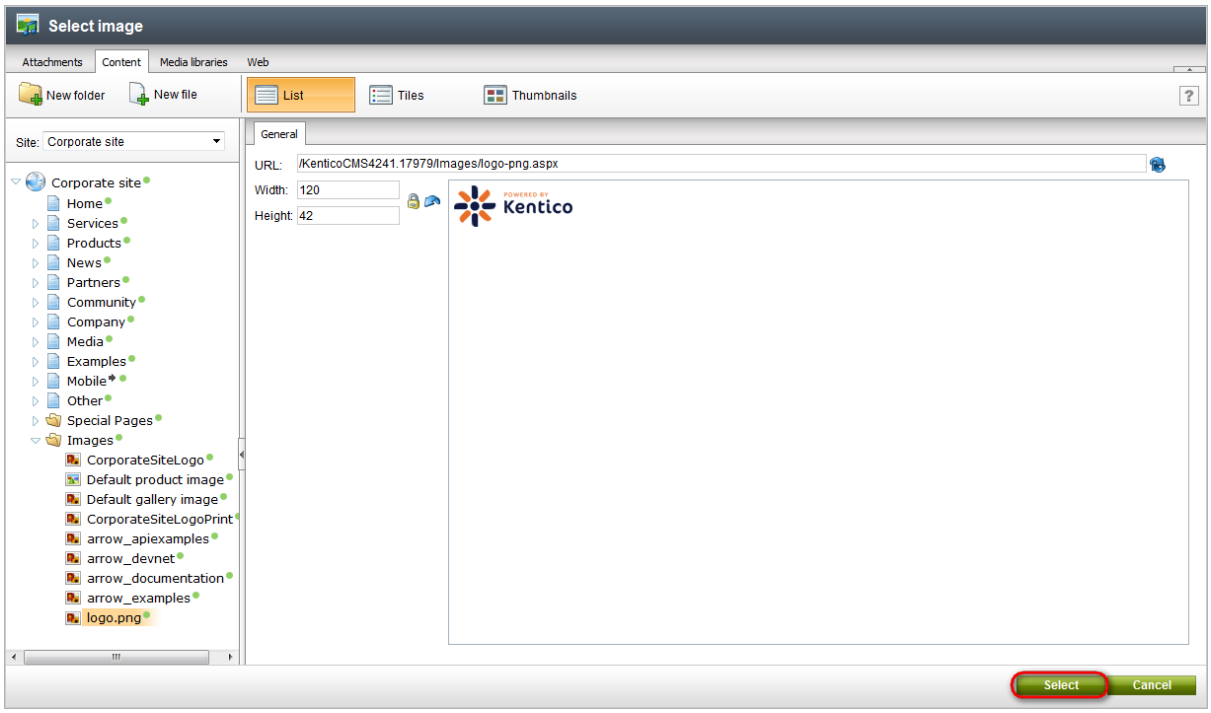
3. In the **Select image** dialog that pops up, select **Images** in the content tree and click the **New file** button above the content tree.



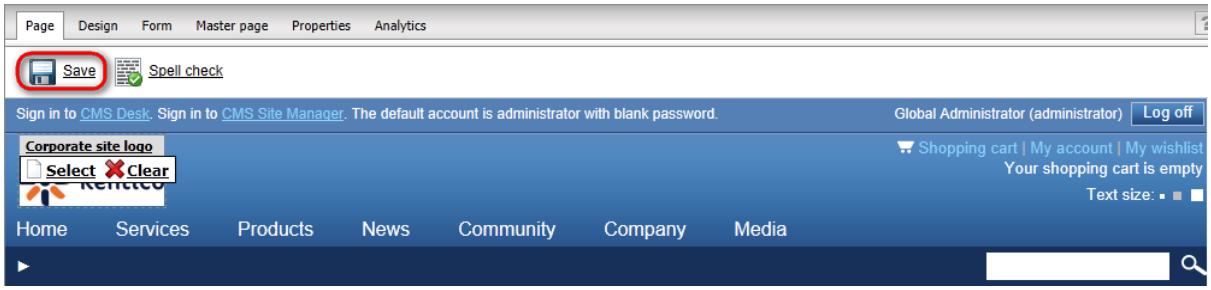
4. Find an image you want to upload and click **Open**.



7. The logo should be added to the Images folder. With the logo selected in the content tree, click **Select**.



8. Now click **Save**.

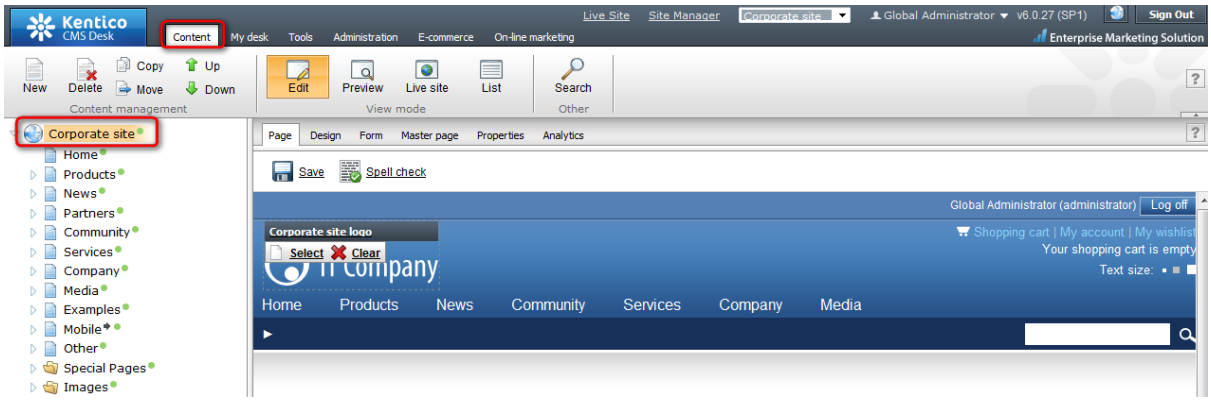


You've just publish a new logo on your website.

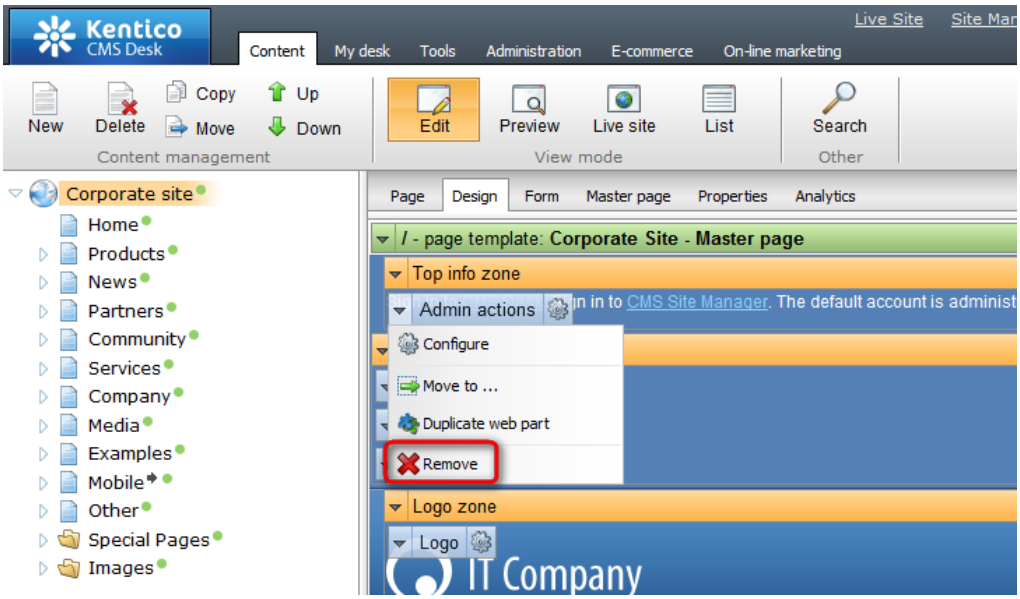
7.20.2 Removing the log-on bar

This topic demonstrates how to remove the links used to log in to the administration interface, which are displayed at the top of the sample Corporate Site by default:

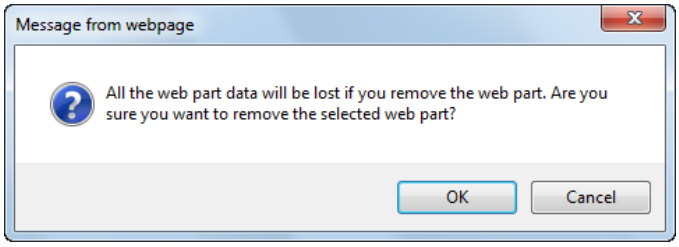
1. Go to **CMS Desk -> Content** and select the root of the content tree (**Corporate Site**).



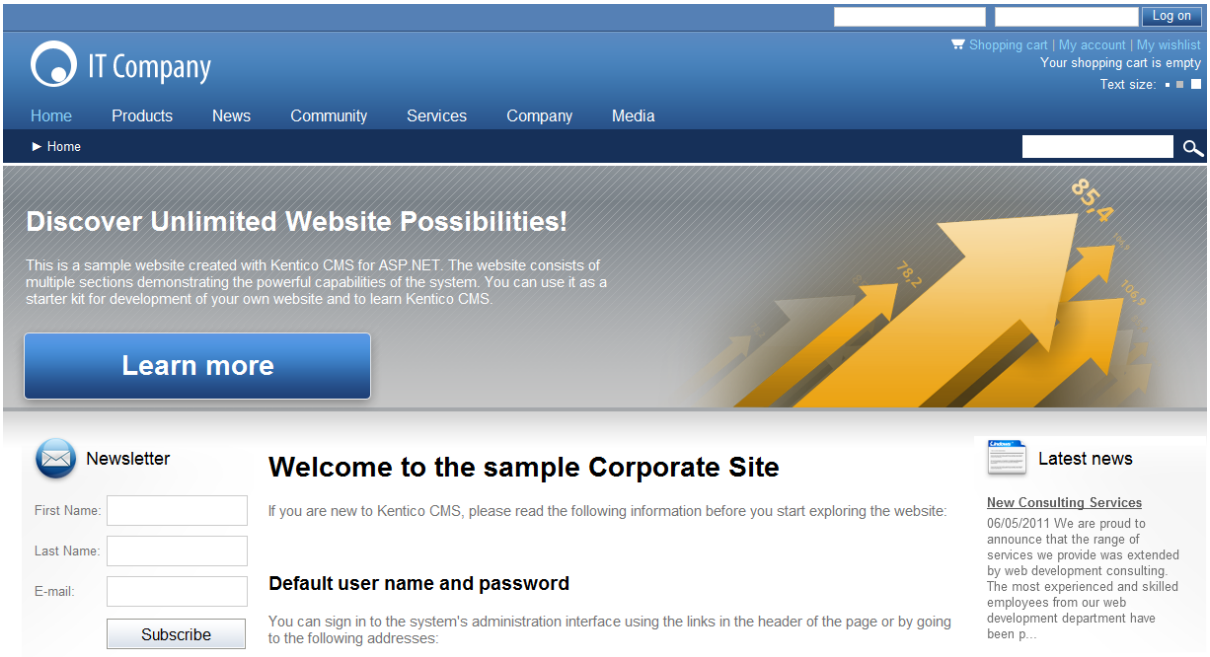
2. Switch to the **Design** tab, right click the **Admin actions** web part in the **Top info zone** and choose the **Remove** option.



3. Click **OK** to remove the links from your website.

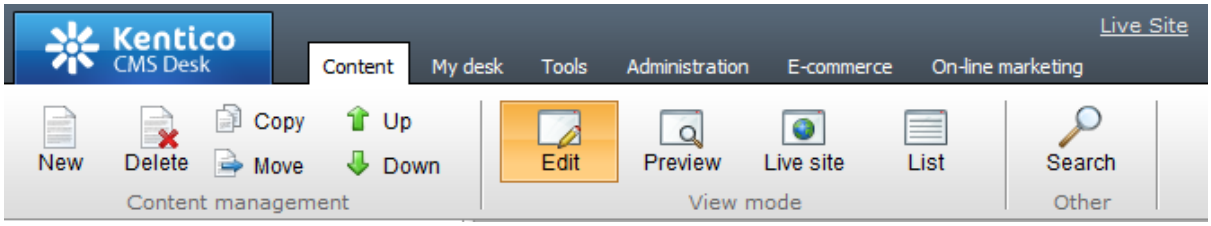


If you view the live site, the page will be displayed without the CMS Desk and Site Manager login links.



7.20.3 Changing the CMS Desk and Site Manager logo

The entire design of the administration interface is fully customizable, so it is also possible to change the brand-related logos.



To replace the header logos shown on the top left of CMS Desk or Site Manager, open the **App_Themes\Default\Images** folder in your Kentico CMS project's directory. The default image files are located inside the *Images.zip* archive, specifically the *Design\Branding* subfolder. To override them, manually create the **Design\Branding** folders outside of the archive directly in the **Images** directory and then add the replacement files containing your own logos into this location. The names of the files must match the names of the default logos in the archive.

The logon page also contains several images that you may wish to replace. This can be done using the same approach as for the header logos, but in this case the folder path that must be created for the replacement files is **Others\LogonForm**. You can check the names of the original files in the corresponding directory inside the *Images.zip* archive.

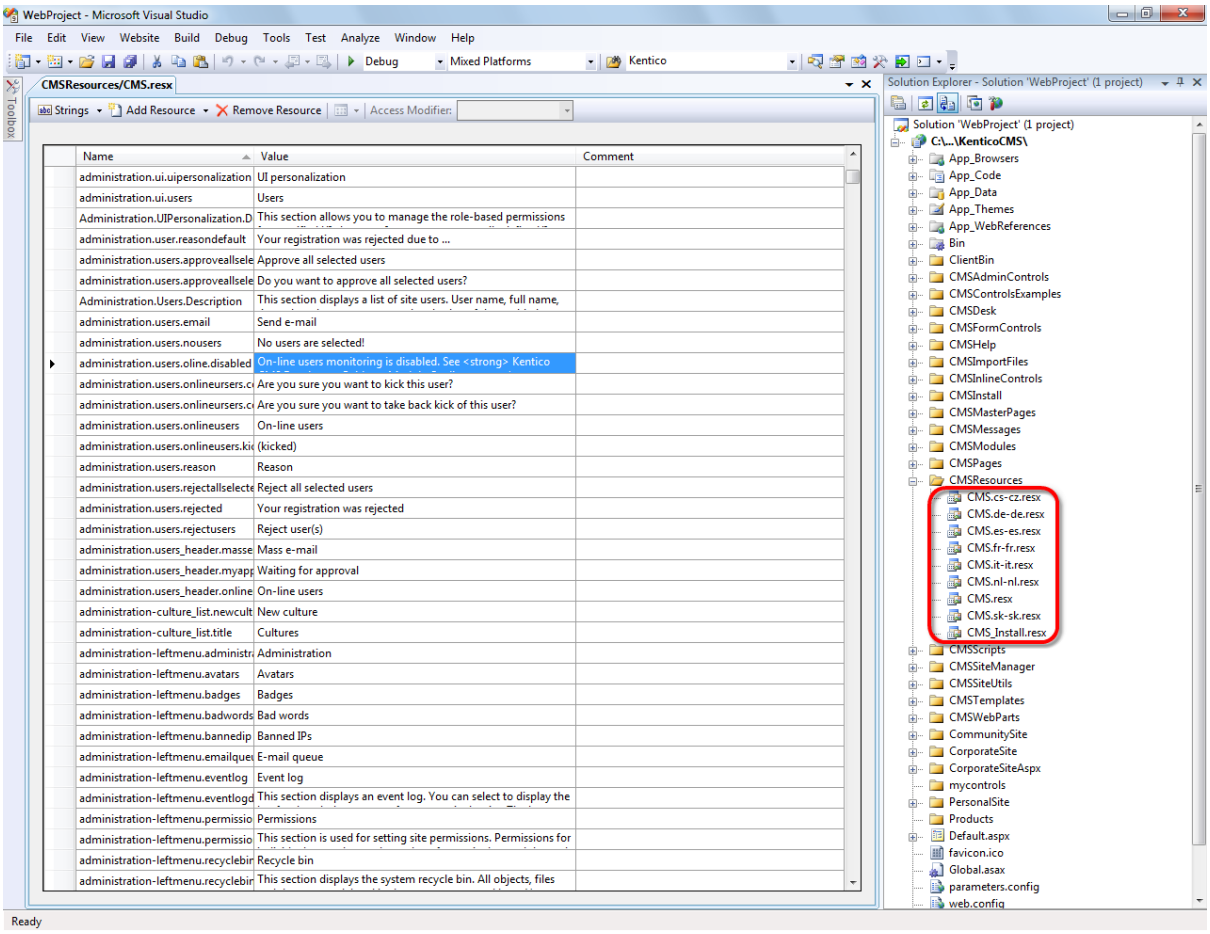
Please note

If you wish to use different image file names or file system paths, you can customize the default stylesheets of the user interface.

These can be found and edited in the project directory, specifically under the **App_Themes\Default** folder.

7.20.4 Renaming resource strings

While re-branding Kentico CMS, you might need to check all the resource strings in the **.resx** files stored in the **~/CMSResources** folder and replace occurrences of the word **Kentico** with your own brand.



7.21 REST service

7.21.1 Overview

REST is an abbreviation of **Representational state transfer**, which is a style of software architecture designed for distributed systems, typically for the World Wide Web. Kentico CMS has a built-in REST service, which can be used to read, create, update and delete virtually any object or document within the system. These operations are performed by requesting specific URLs. The Kentico CMS REST service can be referred to as RESTful, which means that it supports both directions of data transfer (from and into the system).

To enable the service, certain Prerequisites need to be met on the machine where the Kentico CMS instance is installed. Once the prerequisites are met, proceed to the Configuration for REST topic, which describes the required configuration for specific Kentico CMS instances.

Once all the configuration is performed, the REST service should be functional. At this point, you can proceed to the Data retrieval methods and Data manipulation methods topics to learn how to work with documents and objects using the REST service. In the URL parameters topic, you can learn about querystring parameters that can be used to filter data retrieved by GET requests. In the Retrieved data examples topic, you can find examples of data retrieved from Kentico CMS via the REST service in various formats.

The REST service also supports ODATA browsing by providing service documents with information about data that can be obtained by the service. More information on this can be found in the ODATA service documents topic.

The Kentico CMS REST service comes with the **Grid for REST service** web part, which can be used to display data obtained from the service in a simple grid. On the sample **Corporate Site**, you can find an example of this web part on the **/Examples/API/REST-service** page.

7.21.2 Prerequisites

In order for the Kentico CMS REST service to be functional, you must ensure the following:

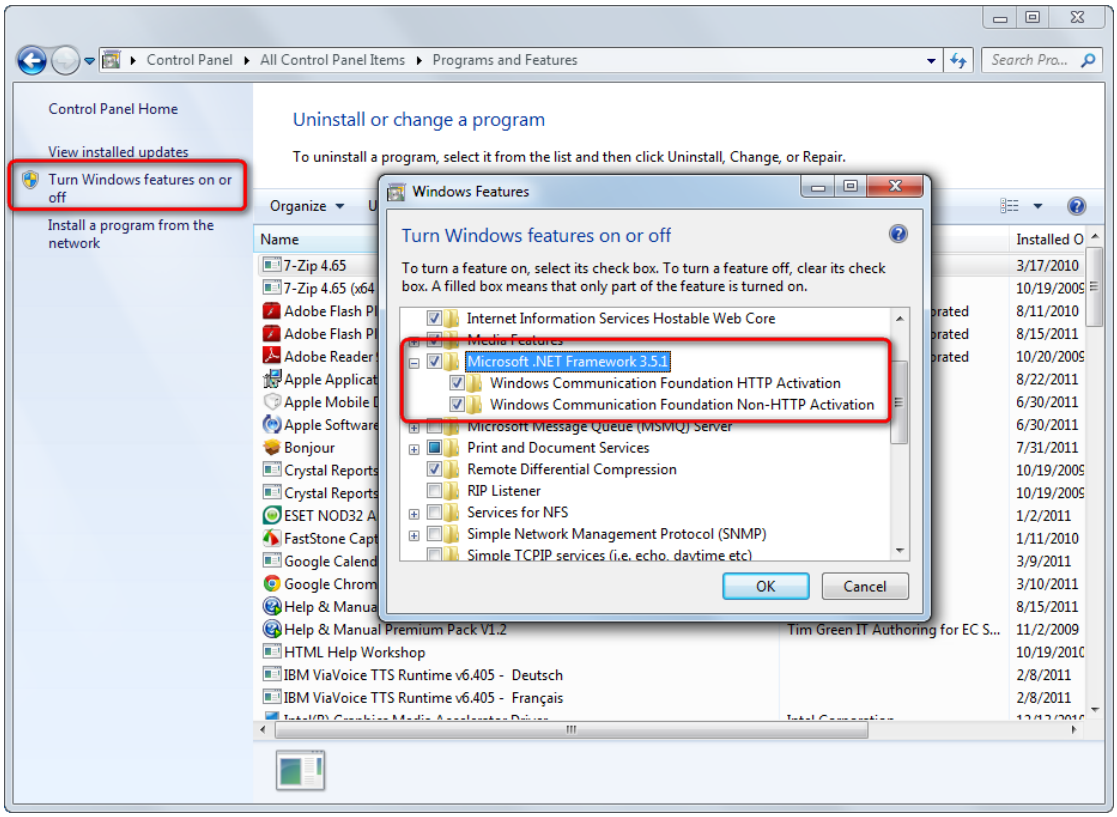
In Windows

Go to **Control Panel -> Programs and Features** and click **Turn Windows features on or off** in the left menu.

Windows 7 / Windows Server 2008

1. Expand the **Microsoft .NET Framework <version>** node in the dialog window.
2. Make sure that both of the following features are installed:

- **Windows Communication Foundation HTTP Activation**
- **Windows Communication Foundation Non-HTTP Activation**

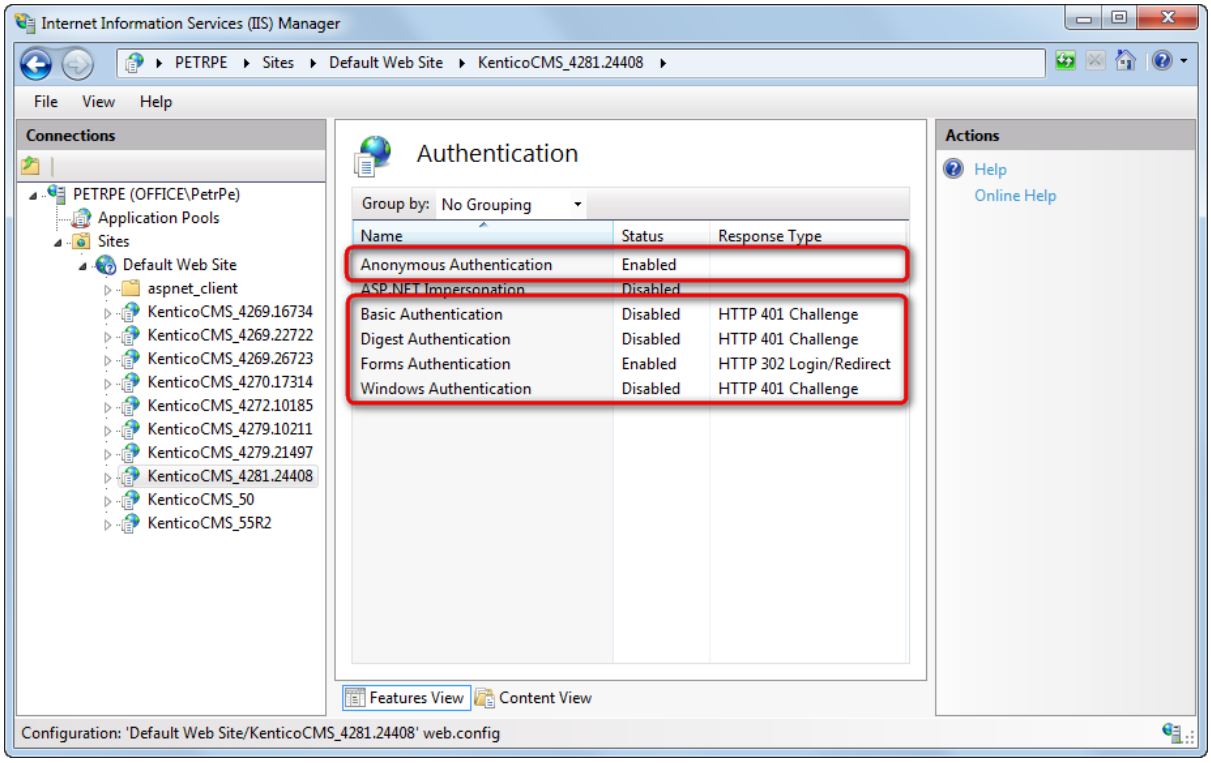


Windows 8 / Windows Server 2012

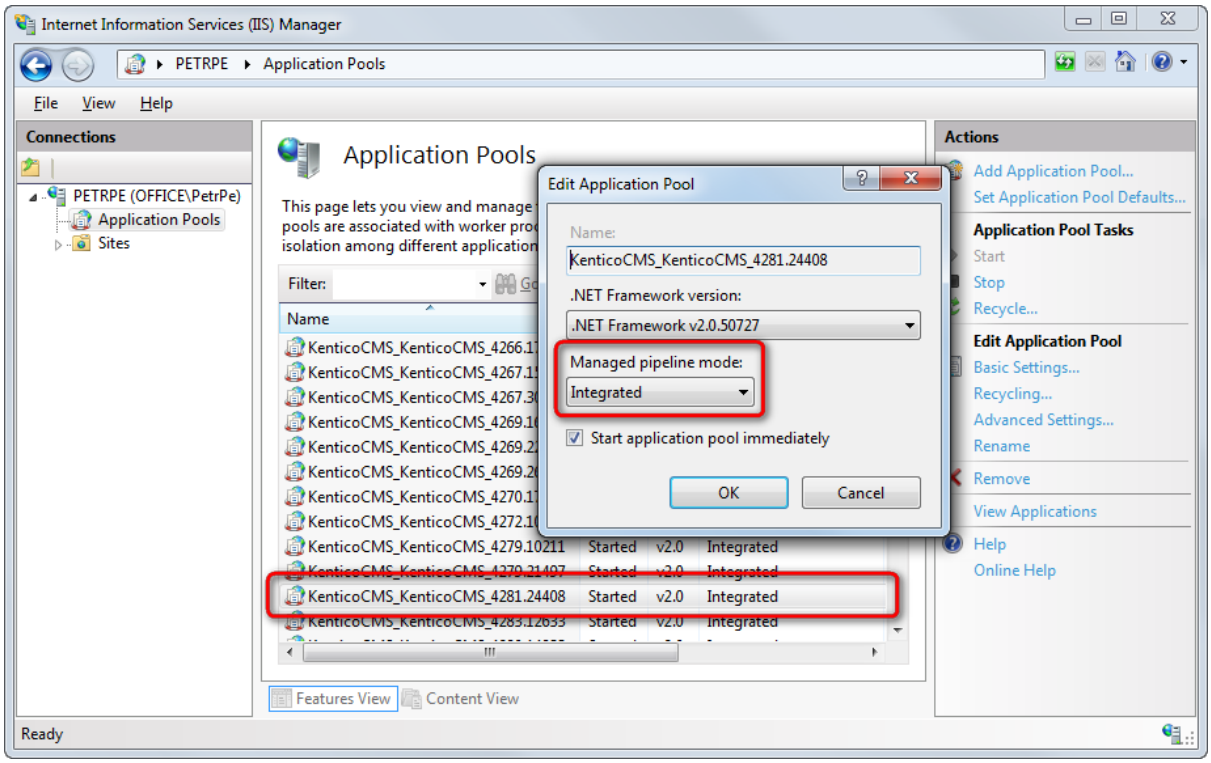
1. Expand the **.NET Framework 4.5 Advanced Services** node.
2. Make sure that the **WCF Services -> HTTP Activation** feature is installed.

In IIS Manager

1. Select the website for which you want REST to be enabled.
2. Open the **Authentication** configuration.
3. Ensure that either **Anonymous**, **Basic** or **Forms** authentication is enabled. No other authentication types are supported.
4. Disable all other types of authentication.



5. Select **Application Pools** in the navigation tree.
6. Double-click the application pool used by your website.
7. Make sure the pool uses **Integrated** Managed pipeline mode.



Once you have these prerequisites met, you can proceed to the Configuration for REST topic.

7.21.3 Configuration for REST

Once you meet the pre-requisites for using the REST service, you need to make additional settings on the level of the Kentico CMS instance:

1. Edit your application's **web.config** file, find the **system.webServer** section directly under the root (i.e. not under a specific <*location*> element) and add the following attribute to the **<modules>** element:

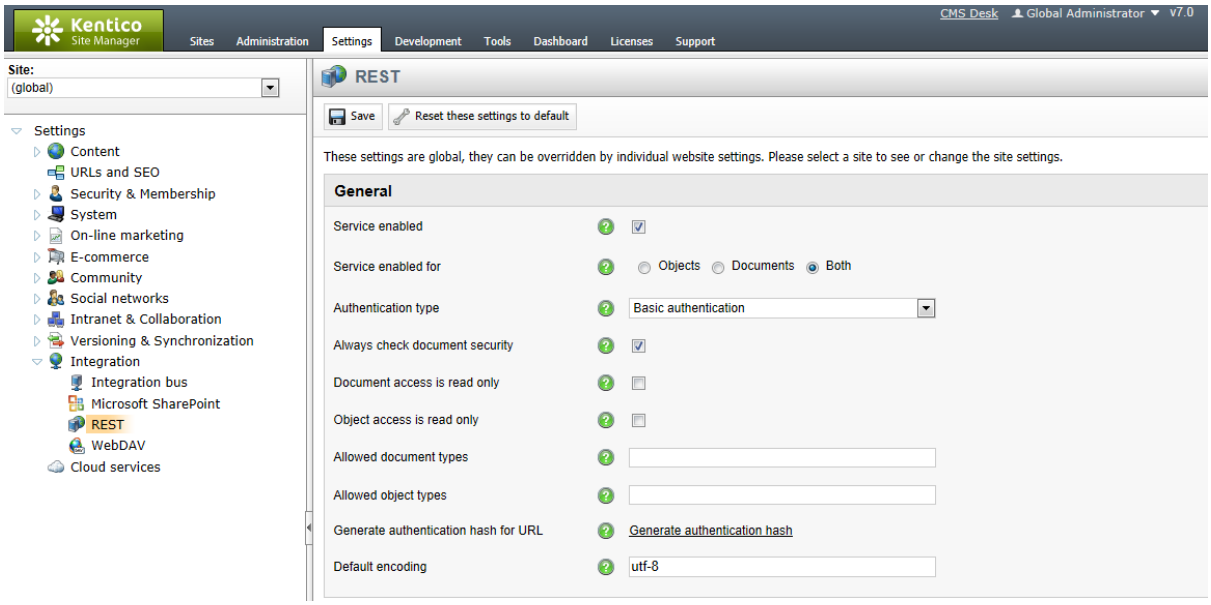
```
<modules runAllManagedModulesForAllRequests="true">
```

2. Go to **Site Manager -> Settings -> Integration -> REST** and adjust related settings.

The following settings are available:

Service enabled	Enables or disables the Kentico CMS REST service.
Service enabled for	Choose if the REST service allows access to objects, documents, or both.
Authentication type	Determines which type of authentication the REST service uses. Supported types are **Basic** and **Forms** authentication. **Note**: You can authenticate REST requests using the hash query string parameter in both modes.

Always check document security	If disabled, security is not checked when accessing published versions of documents. If enabled, security is always checked.
Document access is read only	If enabled, the REST service only allows GET requests for documents (documents cannot be modified).
Object access is read only	If enabled, the REST service only allows GET requests for objects (objects cannot be modified).
Allowed document types	Specifies a list of document types that the REST service is allowed to access. Enter the names of document types separated by semicolons. If empty, all document types are allowed.
Allowed object types	Specifies a list of objects types that the REST service is allowed to access. If empty, all object types are allowed.
Generate authentication hash for URL	Click the link to generate an authentication hash for specific REST URLs. Enter the full absolute URL of the REST request, including the protocol, website domain name, virtual directory, REST path, and query string parameters. For example: *http://mywebsite.com/rest/content/currentsite/en-us/all/news?format=json* The system adds the authentication hash parameter to the URL. You can copy the URL and use it to perform the REST request without any other type of authentication. **Restrictions**: • Only works for GET requests (read only data retrieval) • You cannot use hash parameter authentication for */all* object retrieval requests (*~/rest/<objecttype>/all*).
Default encoding	Sets the character encoding that the REST service uses for requests that do not contain a supported *Accept-Charset* header.



Additional configuration for large-size data upload

If you are planning to upload large-size data into Kentico CMS through the REST service, it is necessary to specify the required data size limit in the application's *web.config* file. This can be done by adding the following pieces of code into the ***<system.serviceModel>*** section at the end of the web.config file:

1. Insert a <*webHttpBinding*> element into the <*bindings*> sub-section as shown below:

```
<webHttpBinding>
  <!-- Limits set to 10 MB (specified value in bytes) -->
  <binding name="RESTQuotaBinding" maxReceivedMessageSize="10485760"
maxBufferPoolSize="10485760" maxBufferSize="10485760" closeTimeout="00:03:00"
openTimeout="00:03:00" receiveTimeout="00:10:00" sendTimeout="00:03:00">
    <readerQuotas maxDepth="32" maxStringContentLength="10485760"
maxArrayLength="10485760" maxBytesPerRead="10485760" />
    <security mode="None" />
  </binding>
</webHttpBinding>
```

Please note that this sample sets all limits to 10 MB. You may need to enter different values according to your specific needs.

2. Then add a <*service*> element under the <*services*> sub-section according to the following:

```
<service name="CMS.WebServices.RESTService">
  <host>
    <baseAddresses>
      <add baseAddress="http://localhost/KenticoCMS/rest" />
    </baseAddresses>
  </host>
  <endpoint address="" bindingConfiguration="RESTQuotaBinding"
binding="webHttpBinding" contract="CMS.WebServices.IRESTService" />
```

```
</service>
```

The *baseAddress* in the code above only contains a sample value and needs to be replaced with the actual root address of the REST service (depending on your website's domain name).

7.21.4 Authenticating REST requests

Every REST request needs to be authenticated. You can select the **Authentication type** of the REST service in **Site Manager -> Settings -> Integration -> REST**.

Basic authentication

With Basic authentication, you need to specify the username and password through the **Authorization** line in the header of every REST request. The header line consists of:

The authentication type (*Basic*)
The username and password connected by a colon (***username:password***), encoded using the Base64 algorithm

For example:

```
Authorization: Basic UmVzdENsaWVudDpzZWNyZXQ=
```

Note: We strongly recommend using SSL to protect the authentication credentials in the request headers. See also: SSL (HTTPS) support

Forms authentication

When using forms authentication of requests, the system identifies users based on the active session and the authentication ticket stored in the .ASPXFORMSAUTH cookie. You can use forms authentication to create a web interface with a login page, and allow users to send REST requests through their browser.

Forms authentication can pose a security threat, because logged in users may unknowingly click links that send malicious REST requests to your site. To protect yourself from such attacks, take the following steps:

1. Immediately after creating the session (i.e. first authentication), you need to set up an **authentication token** by sending a POST request to the following URL:

```
http://<your project URL>/rest/settoken?username=<your user name>&password=<your
password>&token=<your authentication token>
```

The authentication token can be an arbitrary string of characters, for example a GUID.

2. Include the authentication token in the HTTP header of every REST request:

```
Cms-Session-Token: <your authentication token>
```

Hash parameter authentication

You can authenticate individual REST requests by adding a hash parameter to the URL. The hash parameter allows you to prepare REST requests that can be executed by unauthenticated users. Requests that contain the hash parameter ignore the other types of authentication — the value of the **Authentication type** setting does not affect hash authentication.

Restrictions:

- Only works for GET requests (read only data retrieval)
- You cannot use hash parameter authentication for */all* object retrieval requests (*~/rest/<objecttype>/all*). This is an intentional security limitation that protects global data in the system.

Warning

Only use hash parameter authentication for loading data that you want to make publicly available. REST requests with hash authentication can be executed by anyone who obtains the URL (for example by intercepting the web request).

To get the authentication hash for REST requests:

1. Prepare the URL of your REST request in advance.
2. Go to **Site Manager -> Settings -> Integration -> REST**.
3. Click **Generate authentication hash**.
4. Enter the full absolute URL of the REST request, including the protocol, website domain name, virtual directory, REST path, and query string parameters. For example: *http://mywebsite.com/rest/content/currentsite/en-us/all/news?format=json*
5. Click **Authenticate**.

The system adds the authentication hash parameter to the URL. You can copy the URL and use it to perform the REST request without any other type of authentication.

7.21.5 Data retrieval methods

This page contains examples of REST URLs that can be requested to retrieve document or object data from a Kentico CMS instance using the REST service.

Note: The URLs are listed as relative, so if your instance is running at *http://localhost/KenticoCMS*, perform requests listed as *~/rest* in format *http://localhost/KenticoCMS/rest*.

Document retrieval methods

Below, you can find examples of URLs under which object GET requests can be performed to retrieve document data from a Kentico CMS instance. You can also achieve a more detailed specification of retrieved data by appending querystring parameters to the URLs.

See URL parameters for more information.

- Returns a single document of the given culture on the specified site:
  - **~/rest/content/site/<sitename>/<culture>/document/<aliaspath>**, e.g. *~/rest/content/site/corporatesite/en-us/document/company/careers*
  - **~/rest/content/currentsite/<culture>/document/<aliaspath>,** e.g. *~/rest/content/currentsite/en-us/document/company/careers*
- Returns all documents which start with the specified alias path:
  - **~/rest/content/site/<sitename>/<culture>/all/<aliaspath>**, e.g. *~/rest/content/site/corporatesite/en-us/all/news*
  - **~/rest/content/currentsite/<culture>/all/<aliaspath>**, e.g. *~/rest/content/currentsite/en-us/all/news*
- Returns all child documents of the given document:
  - **~/rest/content/site/<sitename>/<culture>/childrenof/<aliaspath>**, e.g. *~/rest/content/site/corporatesite/en-us/childrenof/news*
  - **~/rest/content/currentsite/<culture>/childrenof/<aliaspath>**, e.g. *~/rest/content/currentsite/en-us/childrenof/news*

Constants for default culture and all cultures

If you want to get documents in the default culture, there is a special constant *defaultculture*, which you can use instead of the culture string. The same applies for all cultures, while the constant is *allcultures* in this case.

Object data retrieval methods

Below, you can find examples of URLs under which object GET requests can be performed to retrieve object data from a Kentico CMS instance. You can also achieve a more detailed specification of retrieved data by appending querystring parameters to the URLs.

See URL parameters for more information.

Object ID, GUID and code name

In all of the examples below, object ID, object GUID and object code name are interchangeable, i.e. you can use any of them to specify the respective object.

When using code names of site-related objects, the object with the code name assigned to the current site is always returned, unless a site is explicitly specified in

the URL.

- Exposes the service document (for ODATA browisng). The document contains a list of all available object types and URLs under which objects of the type can be accessed:
  - **~/rest**
- Returns all objects of the given object type (both site objects from all sites and global objects):
  - **~/rest/<objecttype>/all**, e.g. *~/rest/cms.emailtemplate/all*

Note

The REST service does not allow */all* object retrieval for requests that use the hash URL parameter for authentication. This is an intentional security limitation that protects global data.

- Returns all objects of the given object type assigned to the current site. If there is no site binding for the given object type, it returns all objects of the given type:
  - **~/rest/<objecttype>**, e.g. *~/rest/cms.country*
  - **~/rest/<objecttype>/currentsite**, e.g. *~/rest/cms.country/currentsite*
- Returns all objects of the given object type assigned to the specified site. If there is no site binding for the given object type, it returns all objects of the given type:
  - **~/rest/<objecttype>/site/<sitecodename>**, e.g. *~/rest/cms.country/site/corporatesite*
- Returns all global objects of the given object type (which are not assigned to any site). If there is no site binding for the given object type, it returns all objects of the given type:
  - **~/rest/<objecttype>/global**, e.g. *~/rest/cms.emailtemplate/global*
- Returns all sites on which the specified object type is available to the currently authenticated user and URLs under which the objects for the particular sites can be retrieved (for ODATA browsing):
  - **~/rest/<objecttype>/site**, e.g. *~/rest/cms.country/site*
- Returns an object of the given type with the specified ID, GUID or code name:
  - **~/rest/<objecttype>/<id>**, e.g. *~/rest/cms.country/271*
- Returns an object of the given type with the specified code name assigned to the current site:
  - **~/rest/<objecttype>/<objectcodename>**, e.g. *~/rest/cms.country/usa*
  - **~/rest/<objecttype>/currentsite/<objectcodename>**, e.g. *~/rest/cms.country/currentsite/usa*

- Returns a global object of the given type with the specified name:
  - **~/rest/<objecttype>/global/<objectcodename>**, e.g. *~/rest/cms.emailtemplate/global/Blog.NotificationToModerators*
- Returns all supported child object types for the given object (only object ID can be used in this URL, not code name or GUID):
  - **~/rest/<objecttype>/<id>/children**, e.g. *~/rest/cms.country/271/children*
- Returns all child objects of the specified object and of the given object type (only object ID can be used in this URL, not code name or GUID):
  - **~/rest/<objecttype>/<id>/children/<childrenobjecttype>**, e.g. *~/rest/cms.country/271/children/cms.state*
- Returns all binding object types supported by the given object (only object ID can be used in this URL, not code name or GUID):
  - **~/rest/<objecttype>/<id>/bindings**, e.g. *~/rest/cms.user/53/bindings*
- Returns all binding objects of the given type of the specified object (only object ID can be used in this URL, not code name or GUID):
  - **~/rest/<objecttype>/<id>/bindings/<bindingobjecttype>**, e.g. *~/rest/cms.user/53/bindings/cms.usersite*
- Returns *TypeInfo* of the given object type:
  - **~/rest/typeinfo/<objecttype>**, e.g. *~/rest/typeinfo/cms.user*
- Evaluates the given macro expression (without the encapsulating character sequence and with forbidden characters URL-encoded) and serializes the result:
  - **~/rest/macro/<expression>**, e.g. *~/rest/macro/CurrentSite.SiteName*

Custom table and Form data items vs. class definition

When working with custom tables or forms via the REST service, you need to use different URLs when you want to specify the table's or form's class definition and when you want to specify the data items stored in the custom table or form.

For **class definition**, you need to use:

- **~/rest/cms.customtable/<customtablecodename>**, e.g. *~/rest/cms.customtable/customtable.sampletable*
- **~/rest/cms.form/<formcodename>**, e.g. *~/rest/cms.form/contactus*

For the actual **data items**, you need to use:

- **~/rest/customtableitem.<customtablecodename>**, e.g. *~/rest/customtableitem.customtable.sampletable*

- **~/rest/bizformitem.bizform.<formcodename>**, e.g. *~/rest/bizformitem.bizform.contactus*

GET request result encoding

Data retrieval (GET) requests return the results encoded using the default server encoding. If you want to get the result in a different encoding, you need to specify the required encoding in the **Accept-Charset** field of the HTTP GET request. If the specified encoding is not available, the default encoding specified in REST settings is used.

For example:

```
GET http://localhost/CMS/rest/cms.user/administrator HTTP/1.1
User-Agent: Fiddler
Authorization: Basic <enter Base64-encoded <username>:<password> here>
Accept-Charset: utf-8
Host: localhost
Content-Type: text\xml
```

7.21.6 Data manipulation methods

In this topic, you can find examples of REST requests that can be used to manipulate document or object data in a Kentico CMS instance using the REST service. The URLs in this topic are listed as relative, so e.g. if your instance is running at *http://localhost/KenticoCMS*, you would perform a request listed as *~/rest* in format *http://localhost/KenticoCMS/rest*.

Validation of inserted or updated data

Please note that when inserting or updating object data via the REST service, no validation is performed on the side of Kentico CMS. You therefore need to ensure appropriate validation on the side of your application to prevent unwanted behavior.

Document manipulation methods

The following URLs can be requested by the respective type of request in order to manipulate Kentico CMS documents. You can update, delete and insert documents (respectively) using these Http methods. Both XML and JSON formats are supported when performing these requests.

Http POST method

- Creates new document in the given culture, website and alias path location:
  - **~/rest/content/site/<sitename>/<culture>/document/<aliaspath>**, e.g. *~/rest/content/site/corporatesite/en-us/document/news*
  - **~/rest/content/currentsite/<culture>/document/<aliaspath>**, e.g. *~/rest/content/currentsite/en-us/document/news*

Http PUT method

- Updates the specified document:
  - **~/rest/content/site/<sitename>/<culture>/document/<aliaspath>**, e.g. *~/rest/content/site/corporatesite/en-us/document/news*
  - **~/rest/content/currentsite/<culture>/document/<aliaspath>**, e.g. *~/rest/content/currentsite/en-us/document/news*
- Publishes the specified document:
  - **~/rest/content/site/<sitename>/<culture>/publish/<aliaspath>**, e.g. *~/rest/content/site/corporatesite/en-us/publish/news/mynewsitem*
  - **~/rest/content/currentsite/<culture>/publish/<aliaspath>**, e.g. *~/rest/content/currentsite/en-us/publish/news/mynewsitem*
- Performs check-out of the specified document:
  - **~/rest/content/site/<sitename>/<culture>/checkout/<aliaspath>**, e.g. *~/rest/content/site/corporatesite/en-us/checkout/news/mynewsitem*
  - **~/rest/content/currentsite/<culture>/checkout/<aliaspath>**, e.g. *~/rest/content/currentsite/en-us/checkout/news/mynewsitem*
- Performs check-in of the specified document:
  - **~/rest/content/site/<sitename>/<culture>/checkin/<aliaspath>**, e.g. *~/rest/content/site/corporatesite/en-us/checkin/news/mynewsitem*
  - **~/rest/content/currentsite/<culture>/checkin/<aliaspath>**, e.g. *~/rest/content/currentsite/en-us/checkin/news/mynewsitem*
- Archives the specified document:
  - **~/rest/content/site/<sitename>/<culture>/archive/<aliaspath>**, e.g. *~/rest/content/site/corporatesite/en-us/archive/news/mynewsitem*
  - **~/rest/content/currentsite/<culture>/archive/<aliaspath>**, e.g. *~/rest/content/currentsite/en-us/archive/news/mynewsitem*
- Moves the specified document to the next workflow step:
  - **~/rest/content/site/<sitename>/<culture>/movetonextstep/<aliaspath>**, e.g. *~/rest/content/site/corporatesite/en-us/movetonextstep/news/mynewsitem*
  - **~/rest/content/currentsite/<culture>/movetonextstep/<aliaspath>**, e.g. *~/rest/content/currentsite/en-us/movetonextstep/news/mynewsitem*
- Moves the specified document to the previous workflow step:
  - **~/rest/content/site/<sitename>/<culture>/movetopreviousstep/<aliaspath>**, e.g. *~/rest/content/site/corporatesite/en-us/movetopreviousstep/news/mynewsitem*
  - **~/rest/content/currentsite/<culture>/movetopreviousstep/<aliaspath>**, e.g. *~/rest/content/*

currentsite/en-us/movetopreviousstep/news/mynewsitem

Http DELETE method

- Deletes the specified document:
  - **~/rest/content/site/<sitename>/<culture>/document/<aliaspath>**, e.g. *~/rest/content/site/corporatesite/en-us/document/news*
  - **~/rest/content/currentsite/<culture>/document/<aliaspath>**, e.g. *~/rest/content/currentsite/en-us/document/news*

Please note

If you specify a *NodeID* and a *DocumentID* is not specified, the operation is considered as new culture version creation. If you want to create a linked document, you have to specify a *NodeLinkedNodeID* in the request.

Examples of document manipulation requests

Below, you can find examples of document manipulation requests. For testing purposes, you can use Fiddler to send the requests to your web application and see the results.

The following example creates a sample document in the content tree.

```
POST http://localhost/CMS/rest/content/site/CorporateSite/en-us/document/Home
HTTP/1.1
User-Agent: Fiddler
Authorization: Basic <enter Base64-encoded <username>:<password> here>
Host: localhost
Content-Type: text\xml
Content-Length:165

<CMS_MenuItem>
  <NodeClassID><enter class ID here></NodeClassID>
  <DocumentName>Services</DocumentName>
  <DocumentPageTemplateID><enter template ID here></DocumentPageTemplateID>
</CMS_MenuItem>
```

The following example updates the sample document created by the example above.

```
PUT http://localhost/CMS/rest/content/site/CorporateSite/en-us/document/Home/
Services HTTP/1.1
User-Agent: Fiddler
Authorization: Basic <enter Base64-encoded <username>:<password> here>
Host: localhost
Content-Type: text\xml
```

```
Content-Length:81

<CMS_MenuItem>
  <DocumentName>Services MODIFIED</DocumentName>
</CMS_MenuItem>
```

The following example creates a new language version of the document created by the first example in this section.

```
POST http://localhost/CMS/rest/content/site/CorporateSite/cs-cz/document/Home/
Services HTTP/1.1
User-Agent: Fiddler
Authorization: Basic <enter Base64-encoded <username>:<password> here>
Host: localhost
Content-Type: text\xml
Content-Length:103

<CMS_MenuItem>
  <NodeID>51939</NodeID>
  <DocumentName>Services CZ</DocumentName>
</CMS_MenuItem>
```

Object data manipulation methods

The following URLs can be requested by the respective type of request in order to manipulate Kentico CMS objects. You can update, delete and insert objects (respectively) using these Http methods. Both XML and JSON formats are supported when performing these requests.

HTTP POST method URLs

Creates a new object of the specified type.

- **~/rest/<objectType>**, e.g. *~/rest/cms.user*

Creates a new object of the specified type and assigns it to the current website.

- **~/rest/<objectType>/currentsite**, e.g. *~/rest/cms.user/currentsite*

Creates a new object of the specified type and assigns it to the specified website.

- **~/rest/<objectType>/site/<sitename>**, e.g. *~/rest/cms.user/site/corporatesite*

HTTP PUT method

Updates the object of the specified type with the specified ID.

- **~/rest/<objectType>/<id>**, e.g. *~/rest/cms.user/53*

Updates the object of the specified type with the specified code name and assigns it to the specified site.

- **~/rest/<objectType>/site/<sitename>/<objectcodename>**, e.g. *~/rest/cms.user/site/*

corporatesite/administrator

Updates the object of the specified type with the specified code name and assigns it to the current website.

- **~/rest/<objectType>/<currentsite>/<objectcodename>**, e.g. *~/rest/cms.user/currentsite/administrator*

Updates the global object of the specified type with the specified code name.

- **~/rest/<objectType>/global/<objectcodename>**, e.g. *~/rest/cms.user/global/administrator*

HTTP DELETE method

Deletes the object of the specified type with the specified ID.

- **~/rest/<objectType>/<id>**, e.g. *~/rest/cms.user/53*

Deletes the object of the specified type with the specified code name assigned to the specified website.

- **~/rest/<objectType>/site/<sitename>/<objectcodename>**, e.g. *~/rest/cms.user/site/corporatesite/administrator*

Deletes the object of the specified type with the specified code name assigned to the current website.

- **~/rest/<objectType>/<currentsite>/<objectcodename>**, e.g. *~/rest/cms.user/currentsite/administrator*

Deletes the globlal object of the specified type with the specified code name.

- **~/rest/<objectType>/global/<objectcodename>**, e.g. *~/rest/cms.user/global/administrator*

Examples of object manipulation requests

Below, you can find examples of object data manipulation requests. For testing purposes, you can use e.g. Fiddler to send the requests to your web application and see the results in the UI.

The following request creates a sample *cms.country* object based on the provided XML definition.

```
POST http://localhost/CMS/rest/cms.country HTTP/1.1
User-Agent: Fiddler
Authorization: Basic <enter Base64-encoded <username>:<password> here>
Host: localhost
Content-Type: text\xml
Content-Length: 271

<data><cms_country><CountryDisplayName>Test Country REST</
CountryDisplayName><CountryName>TestCountryREST</CountryName></
cms_country><CMS_State><StateDisplayName>Test State 1</
StateDisplayName><StateName>TestState1</StateName><StateCode>TST</StateCode></
CMS_State></data>
```

The following example updates the sample *cms.country* object created by the example above based on the provided XML data.

```
PUT http://localhost/CMS/rest/cms.country/TestCountryREST HTTP/1.1
User-Agent: Fiddler
Authorization: Basic <enter Base64-encoded <username>:<password> here>
Host: localhost
Content-Type: text\xml
Content-Length: 102

<data><cms_country><CountryDisplayName>Test Country MODIFIED</
CountryDisplayName></cms_country></data>
```

The following request creates a sample *cms.country* object based on the provided JSON definition.

```
POST http://localhost/CMS/rest/cms.country?format=json HTTP/1.1
User-Agent: Fiddler
Authorization: Basic <enter Base64-encoded <username>:<password> here>
Host: localhost
Content-Type: application/json
Content-Length:161

{ "CountryDisplayName":"Test JSON","CountryName":"TestJSON","cms.state":
[{"StateCode":"TST","StateName":"TestStateJSON","StateDisplayName":"Test State
JSON"}] }
```

The following example updates the sample *cms.country* object created by the example above based on the provided JSON data.

```
PUT http://localhost/CMS/rest/cms.country/TestJSON HTTP/1.1
User-Agent: Fiddler
Authorization: Basic <enter Base64-encoded <username>:<password> here>
Host: localhost
Content-Type: application/json
Content-Length:45

{ "CountryDisplayName":"Test JSON MODIFIED" }
```

The following example deletes the sample *cms.country* object created by either of the examples above.

```
DELETE http://localhost/CMS/rest/cms.country/TestCountryREST HTTP/1.1
User-Agent: Fiddler
Authorization: Basic <enter Base64-encoded <username>:<password> here>
Host: localhost
```

Null value macro

In case that you need to specify an empty value in a data manipulation request, you can use the **##null##** macro instead of an actual value. This is particularly useful for non-string fields (e.g. typically

foreign keys) where an empty string value would not produce the desired results.

The following example updates the **Home** page of the sample Corporate Site and ensures that it will not have any user specified in the **Document created by** field.

```
PUT http://localhost/CMS/rest/content/currentsite/en-us/document/home HTTP/1.1
User-Agent: Fiddler
Authorization: Basic <enter Base64-encoded <username>:<password> here>
Host: localhost
Content-Type: text\xml
Content-Length: 271

<CMS_File><DocumentCreatedByUserID>##null##</DocumentCreatedByUserID></CMS_File>
```

7.21.7 URL parameters

This topic contains a list of querystring parameters supported by the REST service. You can append these parameters to the request URLs in order to further specify the behavior of the methods. The information in the parentheses explains the expected data type of the parameter along with its default value (in **bold** font).

General parameters

- **Format** (json/atom10/rss20/**xml**)
  Sets the format of the request data. If specified for PUT/POST requests, this parameter has higher priority than the *Content-Type* parameter in the request header. For example, append *?format=json* to the request URL to use the JSON format.

- **Localize** (string) ***[Only available after applying hotfix 7.0.4 or newer]***
  If added to a data retrieval request, the system resolves all localization expressions inside the returned data. Specify the target language by entering the corresponding *culture code* into the parameter's value. Without this parameter, requests always return localization expressions in unresolved format. The *Localize* parameter is supported by both object and document retrieval requests.

  For example, append *?localize=fr-fr* to the request URL to resolve all localization expressions into their French value (or the value in the default UI culture if the expression is not defined in French).

- **Hash** (string)
  Allows you to authenticate the request without requiring an authentication header or Forms authentication. To generate a valid hash for a request:

  a. Go to **Site Manager -> Settings -> Integration -> REST**.
  b. Click **Generate authentication hash**.
  c. Enter the full absolute URL of the REST request, including the protocol, website domain name, virtual directory, REST path, and query string parameters.
  d. Click **Authenticate**.

  The system adds the authentication hash parameter to the URL. You can copy the URL and use it to perform the REST request without any other type of authentication.

Object retrieval parameters

You can add the following parameters to object retrieval requests:

- **ObjectData** (**true**/false)
  Indicates whether the request retrieves object data. To load only the metadata of an object, append *?objectdata=false&metadata=true* to the request URL.

- **MetaData** (true/**false**)
  Determines if the request retrieves the metadata of the object (type, list of properties / columns). To load the metadata, append *?metadata=true* to the request URL.

- **Binary** (true/**false**)
  Indicates whether the request retrieves binary data (e.g. the data of files uploaded into form fields). Binary data is retrieved in *Base64* format. To include the binary data in the response, append *?binary=true* to the request URL.

- **Children** (true/**false**)
  Indicates whether to include child objects in the result. To load child objects, append *?children=true* to the request URL.

- **MaxRelativeLevel** (int, **-1 = all levels**)
  If the *Children* parameter is true, this parameters sets the maximum depth of the exported object tree structure. For example, to load all child objects down to the second level of the object tree, append *?children=true&maxrelativelevel=2* to the request URL.

- **Bindings** (true/**false**)
  Determines if the retrieved data includes bindings to child objects and sites. Requests only return bindings if the *ObjectData* parameter is true. To load object data with its bindings, append *?bindings=true* to the request URL.

- **OtherBindings** (true/**false**)
  Determines if the retrieved data includes bindings to other objects (M:N relationships). Requests only return bindings if the *ObjectData* parameter is true. To load object data with bindings to other objects, append *?otherbindings=true* to the request URL.

- **Metafiles** (true/**false**)
  Determines if the retrieved data includes metafiles attached to the object. To load metafiles, append *?metafiles=true* to the request URL.

- **Relationships** (true/**false**)
  Determines if the retrieved data includes object relationships. To load relationships, append *?relationships=true* to the request URL.

- **Categories** (true/**false**)
  Determines if the retrieved data includes the object's category structure (if the object belongs to a category). To load the category structure, append *?categories=true* to the request URL.

- **Translations** (true/**false**)
  Indicates whether to include a translation table of foreign keys in the result. To load the translation table, append *?translations=true* to the request URL.

- **Hierarchy** (true/**false**)
  If true, the response data is exported in a hierarchical structure (if false, the children – bindings – parent structure is flat). To export the data in a hierarchical structure, append *?hierarchy=true* to the request URL.

Multiple object retrieval parameters

When retrieving a dataset containing multiple objects, the following parameters allow you to filter the items and change their order:

- **Where** (string, **empty by default**)
  SQL WHERE condition for filtering of the dataset. To filter retrieved data using a WHERE condition, append *?where=<text of the condition>* to the request URL.
  For example: *~/rest/cms.user?Where=UserIsEditor=1*

- **OrderBy** (string, **empty by default**)
  SQL ORDER BY clause for modifying the order of the items in the dataset. Append *?orderby=##default##* to use alphabetical order based on the object display name.

- **Columns** (string, **all columns by default**)
  Limits which data columns of the object are retrieved by the request. For example, to load only the *UserName* and *UserID* columns when retrieving users, append *?columns=UserName,UserID* to the request URL. If you add this parameter, the *Binary* parameter is ignored (you can choose whether to include binary columns by enumerating the corresponding columns).

- **TopN** (int, **all records by default**)
  SQL TOP N clause for filtering of the dataset. For example, to load only the first 10 retrieved records, append *?topn=10* to the request URL.

- **Offset** (int, **first record by default**)
  Sets the number of the first record that the request returns (according to the order of the dataset). This parameter allows you to implement paging of the data. For example, to load data starting from the third item in the dataset, append *?offset=2* to the request URL.

- **MaxRecords** (int, **all record by default**)
  Limits the maximum number of retrieved records. You can use this parameter in combination with the *Offset* parameter to load pages containing a specific range of records. For example, to load items 11–20 from the dataset, append *?offset=10&maxrecords=10* to the request URL.

Document retrieval parameters

The following parameters are supported for requests that retrieve document data. You may also use the filtering and ordering parameters listed in the *Multiple object retrieval parameters* section.

- **ClassNames** (string, **all classnames**)
  Limits which document types the request returns. Specified as a list of document type code names separated by semicolons. For example, to retrieve only *cms.article* documents, append *?classnames=cms.article* to the request URL.

- **CombineWithDefaultCulture** (**true**/false)
  Indicates if the request should load the default language versions of documents if they do not exist in the specified culture. To load the language culture versions of documents, append *?*

combinewithdefaultculture=true to the request URL.

- **SelectOnlyPublished** (**true**/false)
  Determines if the request loads only documents that are published on the live site. To also retrieve unpublished documents, append *?selectonlypublished=false* to the request URL.

- **Version** (**published**/last)
  Determines if the request returns the document versions that are published on the live site or the latest version that is being edited in CMS Desk (when using Workflow). To load the latest document versions, append *?version=last* to the request URL.

- **CoupledData** (**true**/false)
  Determines if the request retrieves the data stored in the fields of specific document types (coupled data). To load documents without coupled data, append *?coupleddata=false* to the request URL.

Document deletion parameters

- **DeleteAllCultures** (true/**false**)
  Indicates if the request also deletes all cultural versions of the specified document. To delete all culture versions of a document, append *?deleteallcultures=true* to the request URL.

- **DestroyHistory** (true/**false**)
  Indicates if the request also deletes the document's version history. To delete the version history together with the document, append *?destroyhistory=true* to the request URL.

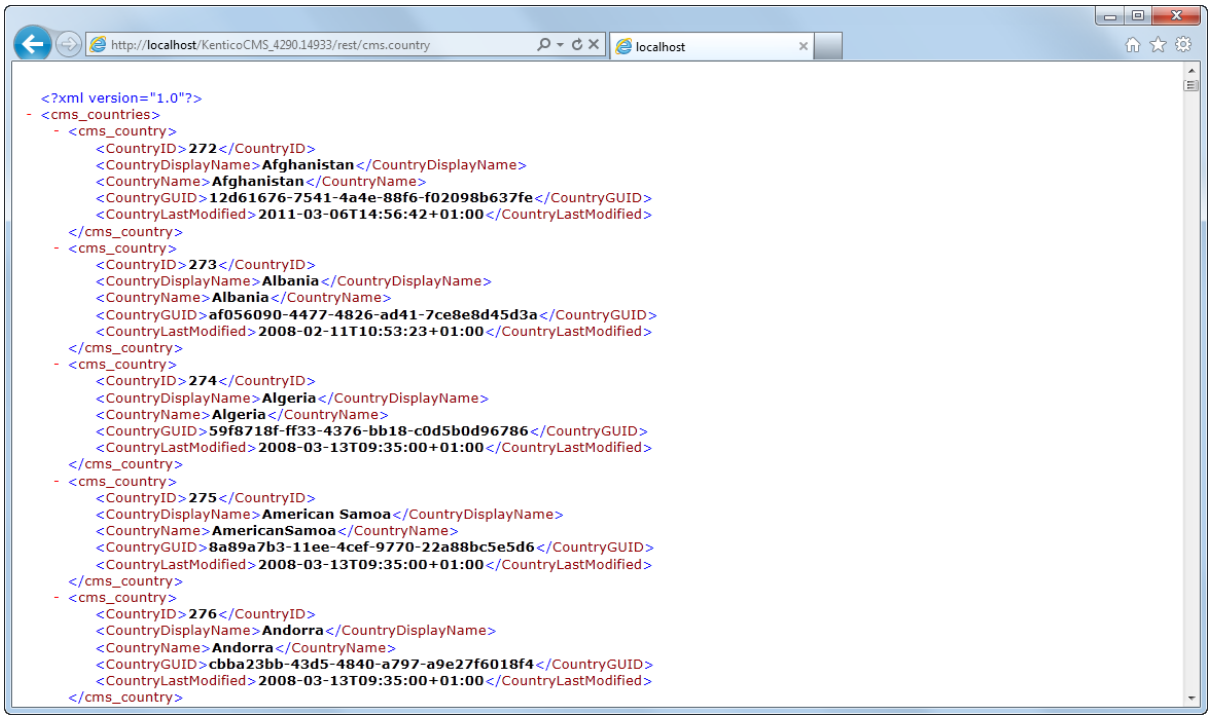
- **DeleteProduct** (true/**false**)
  Indicates if the request also deletes the e-commerce product associated with the specified product document. To delete e-commerce products along with documents, append *?deleteproduct=true* to the request URL.

7.21.8 Retrieved data examples

In this topic, you can see examples of the same data (all *cms.country* objects) obtained from the REST service in various formats.

XML

In the screenshot below, you can see data of all *cms.country* objects retrieved in XML format and displayed in Microsoft Internet Explorer. This result was obtained using the following URL: **~/rest/cms.country**



JSON

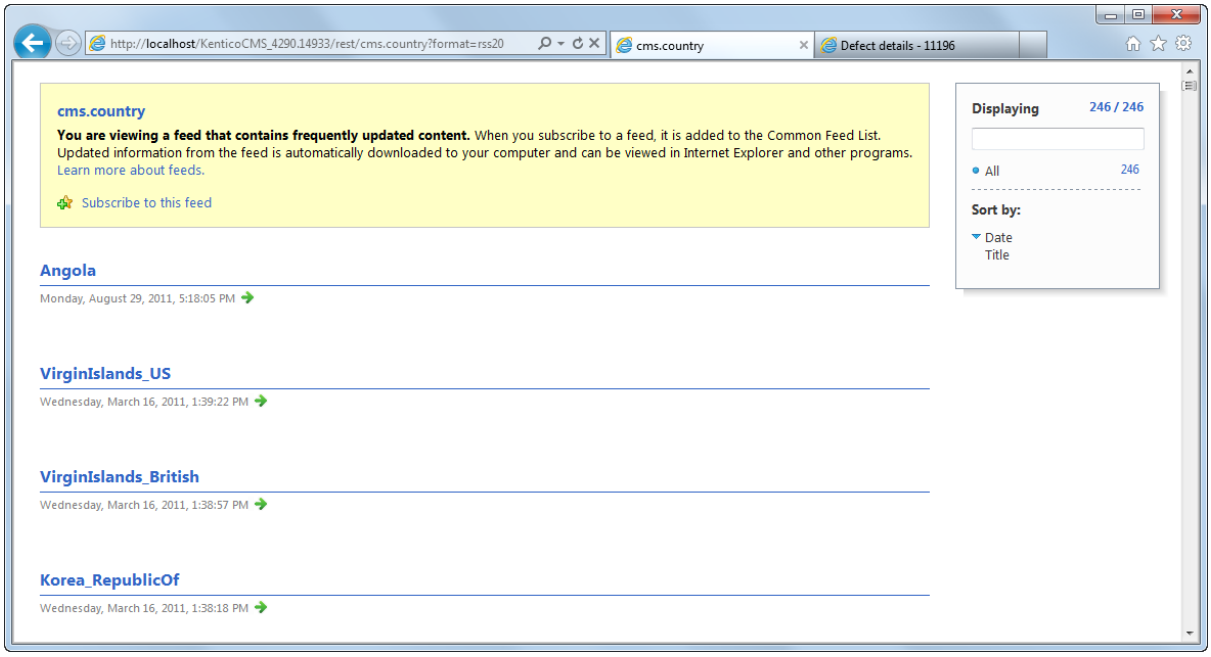
The code below is an extract from data of the *cms.country* objects retrieved in JSON format. This result was obtained using the following URL: **~/rest/cms.country?format=json**

```
{"cms_countries":
[
{"cms_country":
[
{"CountryDisplayName":"Afghanistan","CountryID":272,"CountryLastModified":"\/Date
(1299419802000)\/","CountryName":"Afghanistan","CountryGUID":"12d61676-7541-4a4e-
88f6-f02098b637fe"}
,{"CountryDisplayName":"Albania","CountryID":273,"CountryLastModified":"\/Date
(1202723603000)\/","CountryName":"Albania","CountryGUID":"af056090-4477-4826-ad41-
7ce8e8d45d3a"}
,{"CountryDisplayName":"Algeria","CountryID":274,"CountryLastModified":"\/Date
(1205397300000)\/","CountryName":"Algeria","CountryGUID":"59f8718f-ff33-4376-bb18-
c0d5b0d96786"}
,{"CountryDisplayName":"American Samoa","CountryID":275,"CountryLastModified":"\/
Date(1205397300000)\/","CountryName":"AmericanSamoa","CountryGUID":"8a89a7b3-11ee-
4cef-9770-22a88bc5e5d6"}
,{"CountryDisplayName":"Andorra","CountryID":276,"CountryLastModified":"\/Date
(1205397300000)\/","CountryName":"Andorra","CountryGUID":"cbba23bb-43d5-4840-a797-
a9e27f6018f4"}
,{"CountryDisplayName":"Angola","CountryID":277,"CountryLastModified":"\/Date
(1314631085000)\/","CountryName":"Angola","CountryGUID":"503673e4-fc0e-40a3-9ff9-
521f67680d3f"}
,{"CountryDisplayName":"Anguilla","CountryID":278,"CountryLastModified":"\/Date
(1205397300000)\/","CountryName":"Anguilla","CountryGUID":"fe1c087d-6157-4f94-
```

```
9745-39a49f981bfa"}

...

]
}
,{"TotalRecords":
[
{"TotalRecords":"246"}
]
}
]
}
```

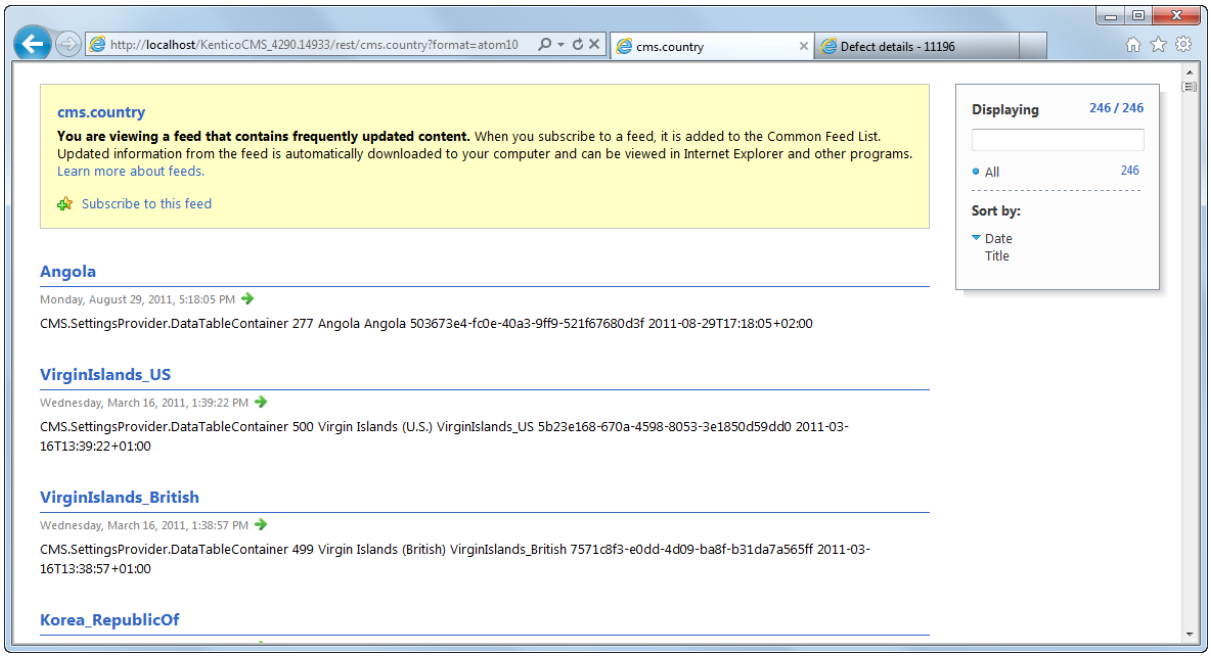
RSS 2.0

In the screenshot below, you can see all *cms.country* objects retrieved in RSS 2.0 format and displayed in Microsoft Internet Explorer. This result was obtained using the following URL: **~/rest/cms.country?format=rss20**



Atom 1.0

In the screenshot below, you can see all *cms.country* objects retrieved in Atom 1.0 format and displayed in Microsoft Internet Explorer. This result was obtained using the following URL: **~/rest/cms.country?format=atom10**

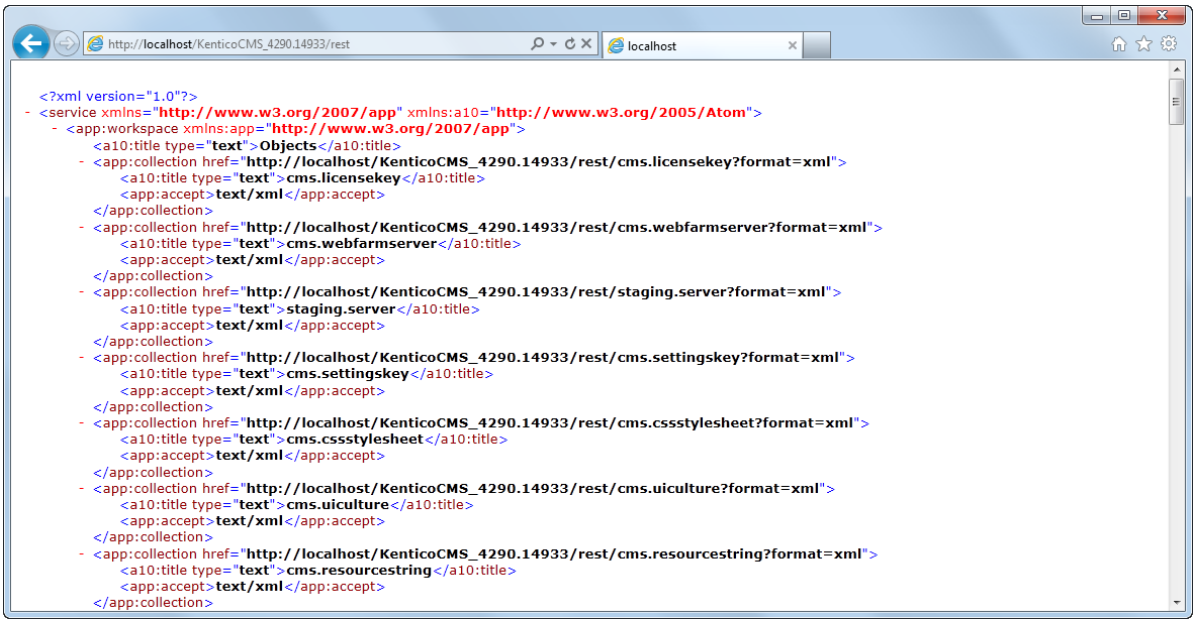


7.21.9 ODATA service documents

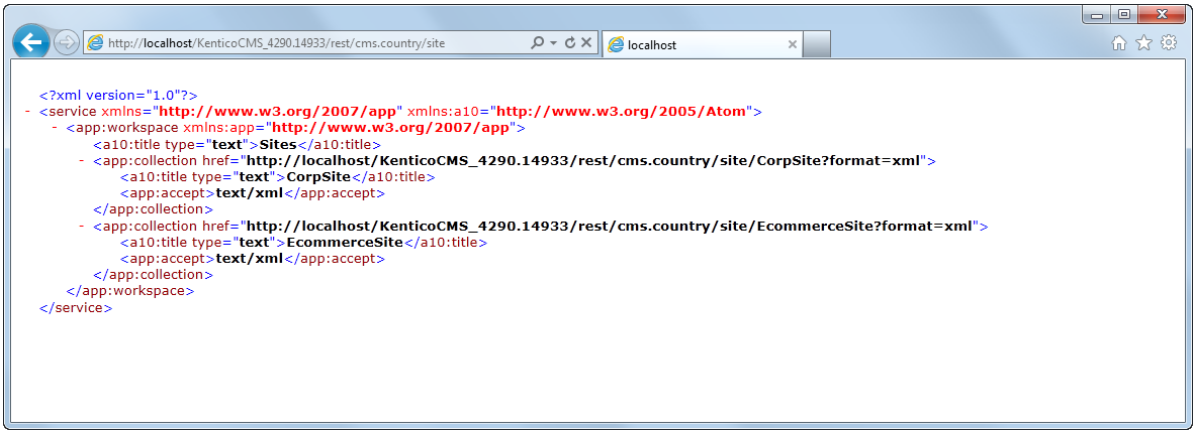
Kentico CMS REST service supports ODATA by providing service documents in a standard format for ODATA services. The service documents provide information about the types of data available via the REST service and about the URLs under which the data can be accessed.

The service documents are available under the following URLs:

- **~/rest -** exposes the service document. The document contains a list of all available object types and URLs under which objects of the type can be accessed.



- **~/rest/<objecttype>/site -** returns a list of all sites on which the specified object type is available to the currently authenticated user and URLs under which objects from the particular sites can be retrieved.



7.22 Scheduler

7.22.1 Overview

The Scheduler allows you to define when specified scheduled tasks should be executed. This is useful when you want some functionality to be performed automatically at a specific time or regularly over a certain time period. The scheduler is leveraged by many of the modules in Kentico CMS to execute tasks necessary for their functionality.

The Scheduler provides various configuration options that determine when scheduled tasks are executed and if they are executed by the application itself or by a dedicated Windows service. In the Configuring task execution topic, you can learn about these configuration possibilities. The Installing the Scheduler Windows service topic provides information on how the Windows service can be installed.

While Kentico CMS comes with a number of default scheduled tasks, it is possible to create custom ones to schedule execution of your own code. The Scheduling custom code topic contains an example of how this can be achieved. To learn about the user interface for management of scheduled tasks and actions that can be performed in it, please refer to the Scheduled tasks administration topic.

The Scheduler internals and API sub-chapter provides information about the database tables and classes used by the module and examples of how scheduled tasks can be managed using the API.

7.22.2 Configuring task execution

This topic explains how to configure execution of scheduled tasks.

Configuring execution by the Kentico CMS application or external Windows service

Scheduled tasks can either be executed by the Kentico CMS application itself, or by a dedicated Windows service. Executing tasks using the Windows service is recommended for resource-consuming tasks because their execution by the Windows service does not affect the performance of the Kentico CMS application.

For tasks to be executed by the Windows service, the service needs to be installed and running and the **Use external service** setting in **Site Manager -> Settings -> System** needs to be enabled. With these pre-conditions met, you can enable the **Use external service** option in each task's editing interface in **Site Manager -> Administration -> Scheduled tasks**. Tasks with the **Use external service** option disabled will still be executed by the application itself. If the **Use external service** setting in **Site Manager -> Settings -> System** is disabled, even tasks with this option enabled are processed by the application itself.

Scheduling reliability

Since task execution by the application runs within the ASP.NET process, tasks will not be executed by the application if it is not running. This happens when the process is recycled without being started again (after a long period of website inactivity).

If you want to run the scheduling reliably, it's recommended to execute these tasks using the Windows service.

If you still wish to have them executed by the application, you need to ensure that your website is always running. You can do that by using some utility or an external service that requests the home page of your website on a regular basis.

Resolving context macros when executing tasks by the Windows service

Tasks that are executed by the Windows service can't resolve macros dependent on application context (e.g. *{%ApplicationPath%}*). This happens due to the fact that the context is not available when tasks are executed outside the application. Therefore, it is recommended to execute such tasks by the application.

Configuring execution intervals

There are two types of settings which determine when tasks should be executed. First, you can configure a time period after which tasks should be **ready for execution**. This can be configured separately for each individual task. Then, you can configure a time period after which the application or the Windows service **checks if there are any tasks ready for execution**. All tasks that are ready for execution during this check are subsequently executed.

Task interval

Individual tasks have a specified time interval after which they should be considered as ready to be executed. This interval is defined separately for each scheduled task by means of the **Task interval** settings in the task's editing interface in **Site Manager -> Administration -> Scheduled tasks**.

Application scheduler interval

The **Application scheduler interval** setting in **Site Manager -> Settings -> System** determines the time interval after which the application checks if there are any tasks ready to be executed. This setting

only applies to tasks that are configured to be executed by the application itself, not by the Windows service. By default, the application performs the check at the end of each standard page request. This means that tasks are executed when user activity on your website generates requests; you can set the minimum time between the checks to any interval. The execution can be completely disabled by setting the value to 0.

Alternatively, the scheduler can be configured to process tasks regularly according to an automatic internal timer, regardless of website activity. In this case, the **Application scheduler interval** sets the precise interval between these checks. The values you are allowed to set range between 0 - 30 seconds, where 0 disables the checks altogether. You can do this by adding the **CMSUseAutomaticScheduler** key into the **/configuration/appSettings** section of your *web.config* file, as shown below.

```
<add key="CMSUseAutomaticScheduler" value="true" />
```

Service scheduler interval

The **Service scheduler interval** setting in **Site Manager -> Settings -> System** determines the time interval after which the Windows service checks if there are any tasks ready to be executed. This setting only applies to tasks that are configured to be executed by the Windows service, not by the application itself. The value sets the precise interval between the checks, the same as when using the automatic scheduler, but with higher reliability and less application load. The values are entered in seconds and the checks and execution can be completely disabled by setting the value to 0. The maximum value of the setting is 30.

Additional low-level settings

Additional low-level scheduler settings can be done by adding the keys listed in the Scheduler settings section of Appendix B - Web.config parameters.

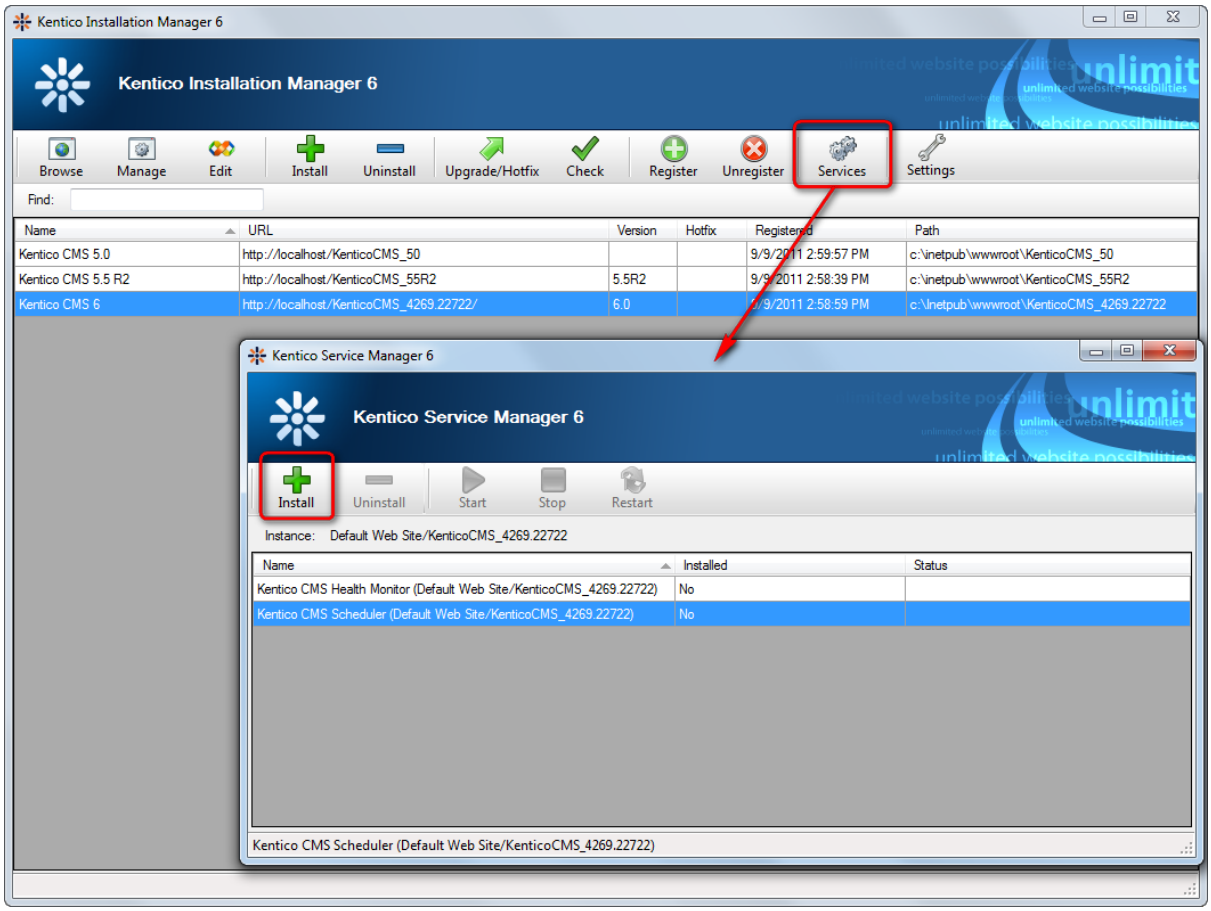
7.22.3 Installing the Scheduler Windows service

Kentico CMS comes with a dedicated Windows service for execution of scheduled tasks. This service can be used to execute specified tasks — typically the resource-consuming ones — instead of the application itself in order to optimize application performance.

The following text describes how the service can be installed. Please note that a specific configuration is also required for scheduled tasks to be executed by the Windows service. See Configuring task execution for more details.

Installing and uninstalling the Scheduler Windows service using Kentico CMS Service Manager

The easiest way to install or uninstall the Scheduler Windows service is to use the Kentico Service Manager utility. The utility can either be launched from Kentico CMS program group in Windows Start menu, or from Kentico Installation Manager, as shown in the screenshot below.



Installing the Scheduler Windows service from the command line

To install the Windows service from the command line, you need to use Installer Tool (InstallUtil.exe), which is a native part of the .NET Framework. The following steps describe installation of the Windows service using the Installer Tool.

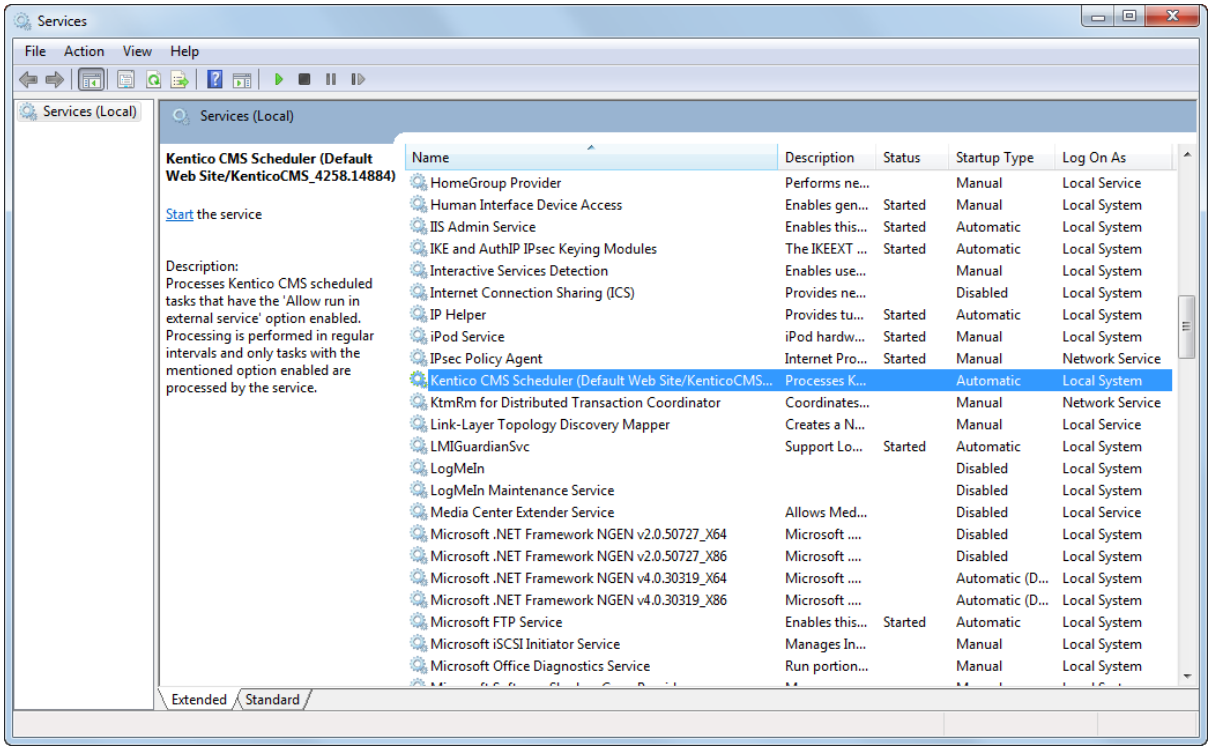
1. Open Windows command line (type *cmd* in the Start menu search box) and navigate to the .NET folder containing *InstallUtil.exe* (e.g. *c:\Windows\Microsoft.NET\Framework64\v4.0.30319*).

2. Execute *InstallUtil.exe* from Windows command line with parameters as shown below. The parameters mean the following:

- **/webpath** - path to the root folder of the Kentico CMS instance for that you want to install the Windows service.
- **second parameter** - path to the *SchedulerService.exe* file in the *Bin* folder inside application root (typically *c:\inetpub\wwwroot\KenticoCMS\bin\SchedulerService.exe*).
- **/LogToConsole** - optional parameter defining if installation progress will be logged to console.
- **/i** - if this parameter is used, the Installer Tool performs installation of the Windows service.

```
InstallUtil.exe /webpath="C:\inetpub\wwwroot\CMS" "c:
\inetpub\wwwroot\KenticoCMS\bin\SchedulerService.exe" /LogToConsole=true /i
```

3. To verify that the service is installed successfully, open Services management console (type *services.msc* into the Start menu search box). In the list of installed services, you should see the **Kentico CMS Scheduler (<CMSApplicationName web.config key value>)** service. The service is not running initially. To start it, click the **Start Service** button in the top toolbar, as highlighted in the screenshot below.



Uninstalling the Scheduler Windows service from the command line

Uninstalling the Scheduler Windows service can be performed exactly the same way as described above, while you only need to use the **/u** parameter instead of **/i** at the end of the command:

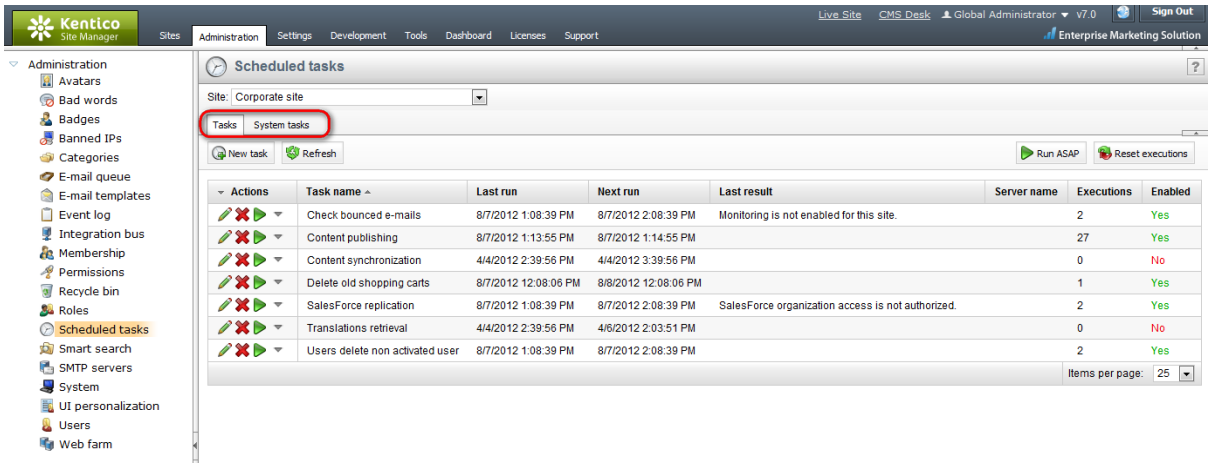
```
InstallUtil /webpath="C:\inetpub\wwwroot\CMS" "c:\Program Files (x86)
\KenticoCMS\6.0\Bin\SchedulerService.exe" /LogToConsole=true /u
```

After executing this command, the Scheduler Windows service for the instance of Kentico CMS specified by the **/webpath** parameter will be uninstalled.

7.22.4 Scheduled tasks administration

You can configure scheduled tasks in **Site Manager -> Administration -> Scheduled tasks**. The tasks are divided into two tabs:

- **Tasks tab** - here, you can manage, configure and add new scheduled tasks.
- **System tasks tab** - here, you can manage and configure system scheduled tasks. System tasks are temporary tasks created dynamically by the system to ensure its correct internal functionality. You cannot add new tasks under this tab as it contains system tasks only.



The **Site** drop-down list at the top is used for selecting a site. If a particular website is chosen, a list of site-specific tasks scheduled for the given site will be displayed in the grid below. Selecting the *(Global)* option allows you to work with tasks that are not related to a specific site. The tasks of the current site can also be edited in **CMS Desk -> Administration -> Scheduled tasks**.

Next to the site selection drop-down list, you can find the following two buttons that may be used to manage the selected site's timer:

- **Restart timer** - restarts the internal scheduling timer. This button is only available when the scheduler is configured to use the internal timer as described in Configuring task execution.
- **Run ASAP** - immediately executes all tasks that are ready to be executed.

New task, available only in **Tasks tab**, allows you to schedule new custom tasks. The process of creating a scheduled task is covered in more detail in the Scheduling custom code topic. The **Refresh** action updates the information displayed below in the list of tasks. Clicking **Reset executions** clears the execution counter for all tasks and starts it again from 0.

The three action icons on the left of each task serve the following purpose:

- **Edit** - allows the properties of the task (e.g. its execution interval) to be modified.
- **Delete** - removes the scheduled task.
- **Execute** - immediately runs the task. When a task is executed manually, the scheduled time of its next execution is automatically moved forward according to the scheduling interval.

If a icon is displayed in the **Actions column**, it indicates that the task should be executed by the Windows service, but was not executed for a long period of time. This typically happens when the service is not running and notifies you about the fact that it should be started in order for these task to be executed.

7.22.5 Scheduling custom code

The process of scheduling a custom task includes two steps:

- Writing the code that performs the required actions
- Registering a new scheduled task in the CMS

Writing the task code

You need to define each scheduled task as a class that implements the **CMS.Scheduler.ITask** interface. To integrate this type of class into the application, you can:

- Create a new assembly (Class library) in your web project and include the task class there. In this case, you must add the appropriate references to both the assembly and the main CMS project.
- Define the scheduled task in **App_Code** and load the class via the API. This ensures that the system automatically compiles the task and you do not need to use a separate assembly. The example below uses this approach.

Example

1. Open your web project in Visual Studio and add a new class into the **App_Code** folder (or **Old_App_Code** if the project is installed as a web application). For example, name the class **CustomTask.cs**.

2. Edit the class and add the following references:

[C#]

```
using CMS.EventLog;
using CMS.Scheduler;
```

3. (Optional) Wrap the class into the **Custom** namespace.

[C#]

```
namespace Custom
{
    /// <summary>
    /// Custom task class.
    /// </summary>
    public class CustomTask
    {

    }
}
```

4. Make the class implement the **ITask** interface.

[C#]

```
public class CustomTask: ITask
```

5. Define the **Execute** method in the class:

[C#]

```
/// <summary>
/// Executes the task.
/// </summary>
/// <param name="ti">Info object representing the scheduled task</param>
public string Execute(TaskInfo ti)
{
    string detail = "Executed from '~/App_Code/CustomTask.cs'. Task data:"
                    + ti.TaskData;

    // Logs the execution of the task in the event log.
    EventLogProvider.LogInformation("CustomTask", "Execute", detail);

    return null;
}
```

You must always include the **Execute** method when writing a scheduled task. The system calls this method whenever the given task is executed, so it needs to contain all code implementing the required functionality. In this example, the task only creates a record in the application's event log so that you can confirm it is being executed.

- The **TaskInfo** parameter of the method allows you to access the data fields of the corresponding scheduled task object. The sample code adds the content of the **TaskData** field into the details of the event log entry.
- The string returned by the method is displayed in the administration interface as the result of the task's most recent execution. You can leave it as *null* in this case.

Loading App_Code task classes

For tasks added to the *App_Code* folder, you must ensure that the system loads the appropriate class when executing the scheduled task. You can find additional information related to this topic in Registering custom classes in App_Code.

1. Create another class in the **App_Code** folder (or **Old_App_Code** if the project is installed as a web application). For example, name the class **ClassLoader.cs**.

2. Edit the class and add the following reference:

[C#]

```
using CMS.SettingsProvider;
```

3. Delete the default class declaration and its content. Instead, extend the **CMSModuleLoader** partial class and define a new attribute inheriting from *CMS.SettingsProvider.CMSLoaderAttribute*:

[C#]

```
[ClassLoader]
```

```
public partial class CMSModuleLoader
{
    /// <summary>
    /// Attribute class for loading custom classes.
    /// </summary>
    private class ClassLoader : CMSLoaderAttribute
    {

    }
}
```

4. Enter the following code into the **ClassLoader** attribute class:

[C#]

```
/// <summary>
/// Called automatically when the application starts.
/// </summary>
public override void Init()
{
    // Assigns a handler for the OnGetCustomClass event.
    ClassHelper.OnGetCustomClass += ClassHelper_OnGetCustomClass;
}

/// <summary>
/// Gets a custom class object based on the given parameters.
/// </summary>
private void ClassHelper_OnGetCustomClass(object sender, ClassEventArgs e)
{
    if (e.Object == null)
    {
        // Checks the name of the requested class.
        switch (e.ClassName)
        {
            // Gets an instance of the CustomTask class.
            case "Custom.CustomTask":
                e.Object = new Custom.CustomTask();
                break;
        }
    }
}
```

In the case of scheduled tasks, the value of the **ClassName** property of the **ClassHelper_OnGetCustomClass** handler's *ClassEventArgs* parameter matches the **Task class name** specified for the given task (*Custom.CustomTask* in this example). The handler checks the class name to identify which specific task is being executed and then passes on an instance of the appropriate class.

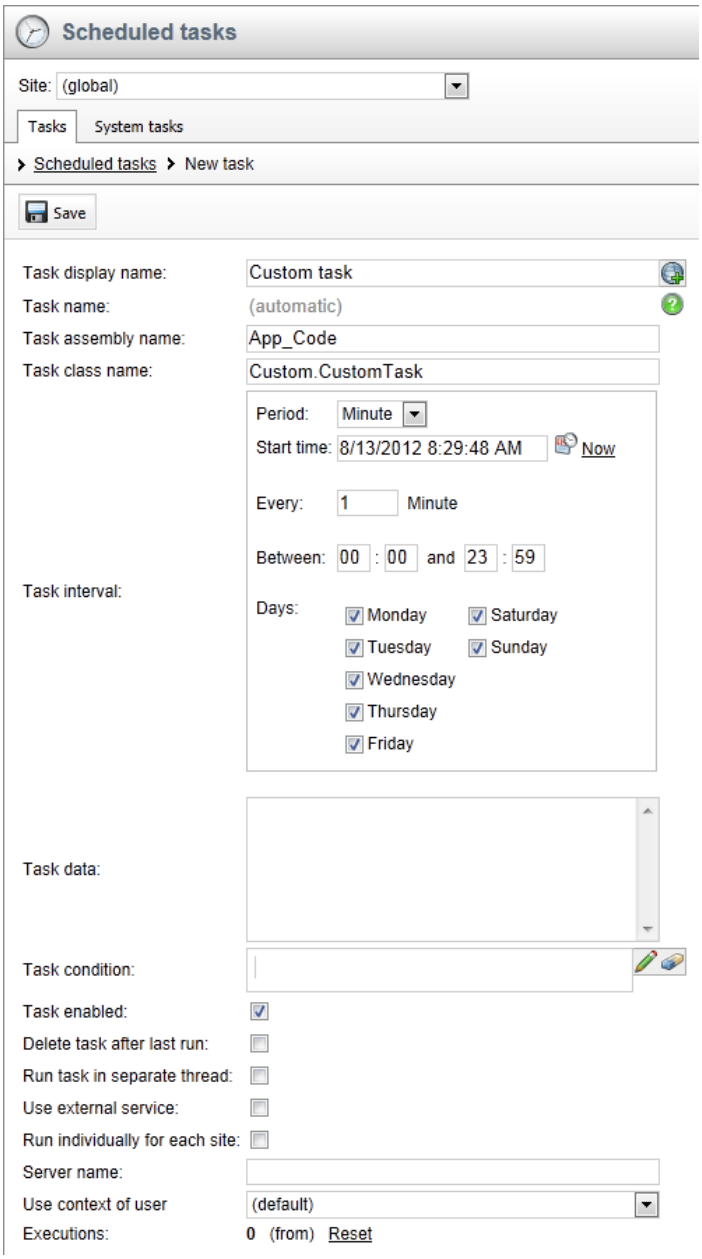
Registering new scheduled tasks

1. Go to **Site Manager -> Administration -> Scheduled tasks** and select the **Site** for which you wish to schedule the task *(global)* if it should be performed for all sites or affect global objects).

2. Click **New task** and fill in the properties of the task:

Task display name	Sets a name for the task that is shown in the administration interface.
Task name	Sets a unique name that serves as an identifier for the scheduled task, for example in the API. You can leave the *(automatic)* option to have the system generate an appropriate code name based on the display name.
Task assembly name	Specifies the name of the assembly where the task is implemented. Enter *App_Code* for tasks implemented in the App_Code folder.
Task class name	Specifies the exact class (including any namespaces) that defines the functionality of the scheduled task. Enter *Custom.CustomTask* to load the task class created in the example sections above.
Task interval	Sets the time interval between two executions of the task. This does not ensure that the task will be executed at the exact time, only that it will be considered ready for execution. The precise execution time depends on the general settings of the scheduler.
Task data	Additional data provided to the task's class. This field can be accessed from the code of the task, so you may use as a parameter to easily modify the task without having to edit its implementation.
Task condition	Allows you to enter an additional macro condition that must be fulfilled in order for the scheduler to execute the task. You can write any condition according to your specific requirements. For details about available macro options and syntax, refer to the Development -> Macro expressions chapter. Clicking on the edit icon () opens the Macro condition editor, which allows you to build conditions through a graphical interface. The **Clear condition** () action removes the current content of the condition field.
Task enabled	This field must be enabled in order for the task to be scheduled.
Delete task after last run	Indicates if the system should delete the task after its final run (applicable if the task is set to run only once).
Run task in separate thread	Indicates if the scheduler should execute the task in a separate thread to improve application performance. It is not possible to access context data (information about the current page, user, etc.) in the code of tasks that are executed in a separate thread.
Use external service	If enabled, the task is processed by the Scheduler Windows service instead of the web application. If the **Use external service** setting in **Site Manager -> Settings -> System** is disabled, even tasks with this option enabled are processed by the application itself. Only some of the

	default scheduled tasks support this option. The ones that do not have it available in their editing interfaces must be processed by the application itself. You cannot define the task in the *App_Code* folder if you wish to use the external service. To run a custom task externally, you must add a new assembly to your project and then define the task class there.
Run individually for each site	This option is only available for global tasks. If enabled, the scheduler executes the task repeatedly as a site-specific task, once for each running site in the system. The task automatically runs within the context of the corresponding site. This can be useful if you wish to manage a task in a single location instead of creating a separate one for every site.
Server name	Sets the name of the web farm server where the task is executed. This field is applicable only if your application is running in a Web farm environment. To add a new task for all web farm servers currently registered in the system, check the **Create tasks for all web farm servers** box below the field.
Use context of user	If the scheduled task needs to access data from the user context in its code (e.g. to check permissions), you can use this property to choose which user is provided. The scheduler always executes the task within the context of the selected user. In most cases, the user context does not affect the functionality of the task, and you can leave the *(default)* option — the context of a public user is used.
Executions	Shows how many times the task has been executed. You can reset this counter back to *0* by clicking **Reset**.

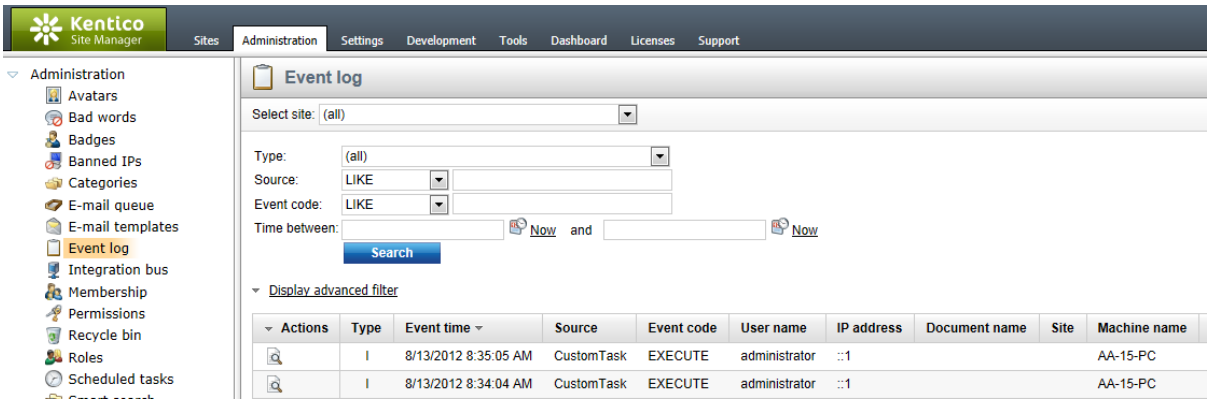


3. Click **Save**.

The task will now be executed regularly according to the specified interval.

Result

To check the result of the sample custom task, go to **Site Manager -> Administration -> Event log** and look for entries with *CustomTask* as their **Source**.



7.22.6 Scheduler internals and API

7.22.6.1 Overview

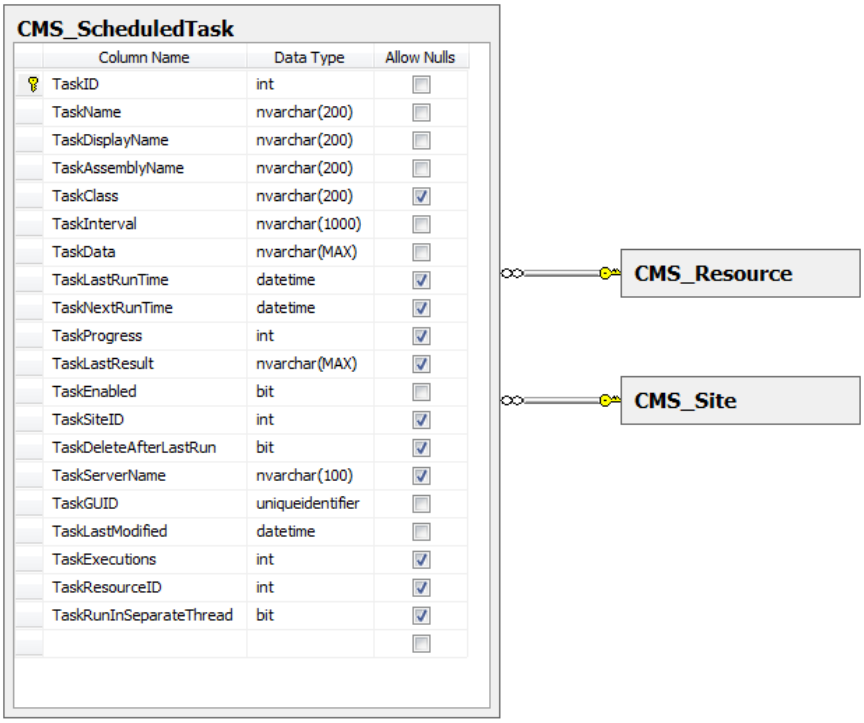
In this chapter, you will learn which database tables and API classes are used by the Scheduler. You will also see the most common API examples.

Further API-related information and examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all members of the related classes, please refer to Kentico CMS API Reference.

7.22.6.2 Database tables

The following database tables are used to store scheduler-related information:

- **CMS_ScheduledTask** - contains records representing scheduled tasks and their settings.



7.22.6.3 API classes

If you are not familiar with database table data management through Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

The classes of the Scheduler can be found in the **CMS.Scheduler** namespace.

CMS_ScheduledTask table API:

- **TaskInfo** - represents one scheduled task.
- **TaskInfoProvider** - provides management functionality for scheduled tasks.

Other classes:

- **SchedulingExecutor** - provides methods for executing scheduled tasks.
- **SchedulingHelper** - provides general functionality and settings for the scheduler.
- **SchedulingTimer** - can be used to manage the internal timer of the scheduler.
- **TaskInterval** - provides management functionality for the time intervals used by scheduled tasks.

7.22.6.4 API examples

7.22.6.4.1 Overview

These topics show examples of how the API of the Scheduler can be used:

- Managing scheduled tasks

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Administration\ScheduledTasks\Default.aspx.cs***.

The scheduler API examples use the following namespaces:

```
using System;
using System.Data;
using System.Collections;

using CMS.GlobalHelper;
using CMS.CMSHelper;
using CMS.Scheduler;
```

7.22.6.4.2 Managing scheduled tasks

The following example creates a scheduled task.

```
private void CreateScheduledTask()
{
    // Create new scheduled task object
    TaskInfo newTask = new TaskInfo();

    // Set the properties
    newTask.TaskDisplayName = "My new task";
    newTask.TaskName = "MyNewTask";
    newTask.TaskAssemblyName = "CMS.WorkflowEngine";
    newTask.TaskClass = "CMS.WorkflowEngine.ContentPublisher";

    // Create interval
    TaskInterval interval = new TaskInterval();

    // Set interval properties
    interval.Period = SchedulingHelper.PERIOD_DAY;
    interval.StartTime = DateTime.Now;
    interval.Every = 2;
```

```
    // Add some days to interval
    ArrayList days = new ArrayList();
    days.Add(DayOfWeek.Monday.ToString());
    days.Add(DayOfWeek.Sunday.ToString());
    days.Add(DayOfWeek.Thursday.ToString());

    interval.Days = days;

    newTask.TaskInterval = SchedulingHelper.EncodeInterval(interval);
    newTask.TaskSiteID = CMSContext.CurrentSiteID;
    newTask.TaskData = "<data></data>";
    newTask.TaskEnabled = true;
    newTask.TaskNextRunTime = SchedulingHelper.GetNextTime(newTask.TaskInterval,
DateTime.Now, DateTime.Now);

    // Save the scheduled task
    TaskInfoProvider.SetTaskInfo(newTask);
}
```

The following example gets and updates a scheduled task.

```
private bool GetAndUpdateScheduledTask()
{
    // Get the scheduled task
    TaskInfo updateTask = TaskInfoProvider.GetTaskInfo("MyNewTask",
CMSContext.CurrentSiteID);
    if (updateTask != null)
    {
        // Update the properties
        updateTask.TaskDisplayName = updateTask.TaskDisplayName.ToLower();

        // Save the changes
        TaskInfoProvider.SetTaskInfo(updateTask);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates scheduled tasks.

```
private bool GetAndBulkUpdateScheduledTasks()
{
    // Get the data
    DataSet tasks = TaskInfoProvider.GetAllTasks();
    if (!DataHelper.DataSourceIsEmpty(tasks))
    {
        // Loop through the individual items
        foreach (DataRow taskDr in tasks.Tables[0].Rows)
        {
```

```
            // Create object from DataRow
            TaskInfo modifyTask = new TaskInfo(taskDr);

            // Update the properties
            modifyTask.TaskDisplayName = modifyTask.TaskDisplayName.ToUpper();

            // Save the changes
            TaskInfoProvider.SetTaskInfo(modifyTask);
        }

        return true;
    }

    return false;
}
```

The following example deletes a scheduled task.

```
private bool DeleteScheduledTask()
{
    // Get the scheduled task
    TaskInfo deleteTask = TaskInfoProvider.GetTaskInfo("MyNewTask",
CMSContext.CurrentSiteID);

    // Delete the scheduled task
    TaskInfoProvider.DeleteTaskInfo(deleteTask);

    return (deleteTask != null);
}
```

The following example runs a scheduled task.

```
private bool RunTask()
{
    // Get the scheduled task
    TaskInfo runTask = TaskInfoProvider.GetTaskInfo("MyNewTask",
CMSContext.CurrentSiteID);

    if (runTask != null)
    {
        // Run task
        SchedulingExecutor.ExecuteTask(runTask);

        return true;
    }

    return false;
}
```

7.23 Social networking

7.23.1 Overview

Integration of Facebook, Google+, LinkedIn and Twitter social networks in Kentico CMS offers various features. A summary of features associated with each social network and available web parts and form controls is displayed in the provided table.

Social network	Feature	Available web parts	Available form controls
Facebook	Authentication	Facebook connect logon*	
		Facebook connect required data*	
	Company interactions	Facebook activity feed*	
	Users interactions	Facebook comments	
		Facebook facepile	
		Facebook like box	
		Facebook like button	
		Facebook send button	
		Facebook recommendations	
	Auto post		Facebook auto post*
Google+	Company interactions	Google+ activity feed*	
		Google+ badge	
	Users interactions	Google+ button	
LinkedIn	Authentication	LinkedIn logon*	
		LinkedIn required data*	
	Company Interactions	LinkedIn apply with*	
		LinkedIn company insider	
		LinkedIn company profile	
		LinkedIn member profile	
	Users interactions	LinkedIn recommend button	
		LinkedIn share button	
Twitter		Twitter feed*	
		Twitter follow button	
		Twitter tweet button	
	Auto post		Twitter auto post*

* Requires configuration of settings at **Site Manager -> Settings -> Social networks -> Facebook/ Google+/LinkedIn/Twitter**.

- **Authentication** - Provides an alternative way for users to log in to your site using their social network accounts.
- **Company interactions** - Provides means for displaying content and information from the company's

social network page.
- **Users interactions** - Provides users with useful buttons and features associated with their social network accounts.
- **Auto post** - Enables content editors to post on social networks through the site UI during the document creation and publishing process.

7.23.2 Autoposting

From Kentico CMS, you can post messages to Facebook and Twitter accounts directly from CMS Desk or the on-site editing interface.

You have the following options how to publish content from within the editing interface to Facebook or Twitter. Each of them is suitable for a different situation.

- If you want to post information about documents of one **specific document type**, either manually, or automatically at the time of publishing, choose Publishing to Facebook or Twitter using form controls.
- If you want to post information about a certain **section of the content tree**, regardless of the document type, choose Publishing to Facebook or Twitter using a workflow step.

In both these cases, you post to the chosen social network from within the interface where you edit a document. You can use macros in the message you post. For example, you can use macros to insert the URL of the document into the message.

Connecting the application to Facebook

Before you can post from Kentico CMS to Facebook, you must authorize the application to use your Facebook account. See Registering your application.

Connecting the application to Twitter

Before you can post tweets from Kentico CMS, you must authorize the application to use your Twitter account.

1. Go to http://dev.twitter.com and log in.

2. Create a new application.

3. Navigate to your application's **Settings** and select **Read and Write** under **Application Type -> Access**.

4. Switch back to **Details** and under **Your access token**, create your access token.

5. Paste the following information into Kentico CMS, **Site Manager -> Settings -> Social networks -> Twitter**.
   - Consumer key
   - Consumer secret
   - Access token
   - Access token secret

6. On Twitter, navigate to your application's **Settings** and select **Read and Write** under **Application**

Type -> Access.

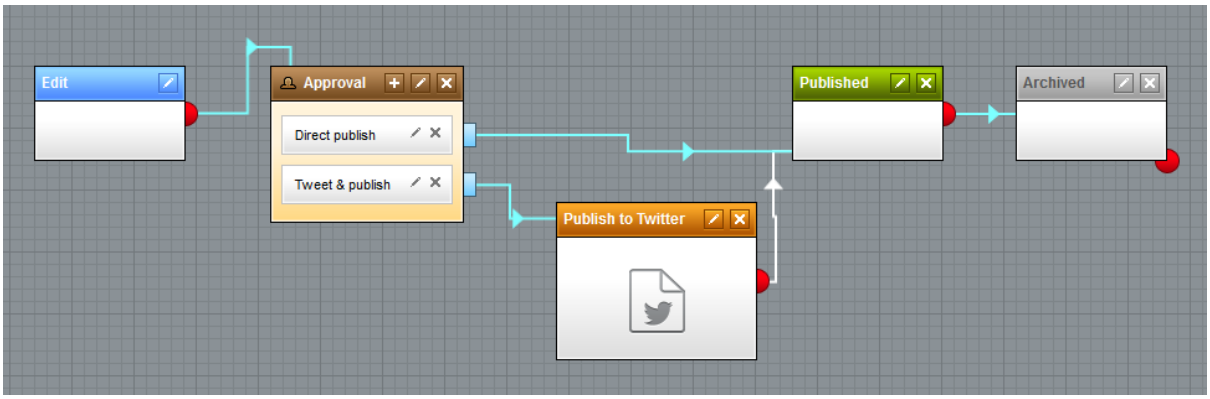
The system is now configured to post to Twitter.

Automatic publishing to Facebook or Twitter using a workflow step

You can post messages to social networks as part of a document's workflow process using the Publish to Facebook and Publish to Twitter workflow steps. You can use this approach to automatically publish information about documents in a section of your site's content. If you want to write a message and publish it to social networks directly from the document editing interface, consider Publishing to Facebook or Twitter using form controls.

1. Configure the application for connecting to Facebook or connecting to Twitter.

2. Create an advanced workflow with the **Publish to Facebook** or **Publish to Twitter** step.



An example workflow process with publishing to Twitter

3. Set the **Text** of the message which will be automatically posted to Facebook/Twitter in the **Workflow step properties**. You can use macros in the message.
   - For example, *Check out our new blog post at www.example.com{% GetDocumentURL() %}*. The resulting URL could then look like this: *www.example.com/blog/my-latest-blog-post.aspx*

4. Create a document using this defined workflow.
   - When advancing over the Publish to Facebook/Publish to Twitter step in the workflow, the predefined message will be automatically posted.

Publishing to Facebook or Twitter using form controls

By adding the **Facebook auto post** and **Twitter auto post** form controls to a document type, you can enable editors to post messages to social networks from the document editing interface. You can also allow choosing whether to post the message right away or after the document gets published.

1. Configure the application for connecting to Facebook or connecting to Twitter.

2. Insert the **Facebook auto post**/**Twitter auto post** form control into the required document type.
   - You can do this in **Site Manager -> Development -> Document types -> Edit** the chosen document type **-> Fields** tab **-> New attribute (✚)**. Choose **Text** as attribute type and choose a sufficient attribute size (at least 140 is recommended).
   - If you plan on using workflow on this document, add "***autopost***" as suffix to the column name.

3. Create a document of the modified type in CMS Desk.

4. Save the document.

5. Type a message into the **Post at Facebook**/**Post at Twitter** field. You can use macros.
   - Note that the message can be only 140 characters long on Twitter.

6. Click the **Post at Facebook**/**Post at Twitter** button, or check **Post at Facebook/Twitter after publishing** (if using workflow) and save the document.
   - The system posts your message on a corresponding social network.
   - If the Post after publishing checkbox is on, the system will post your message once you publish the document.

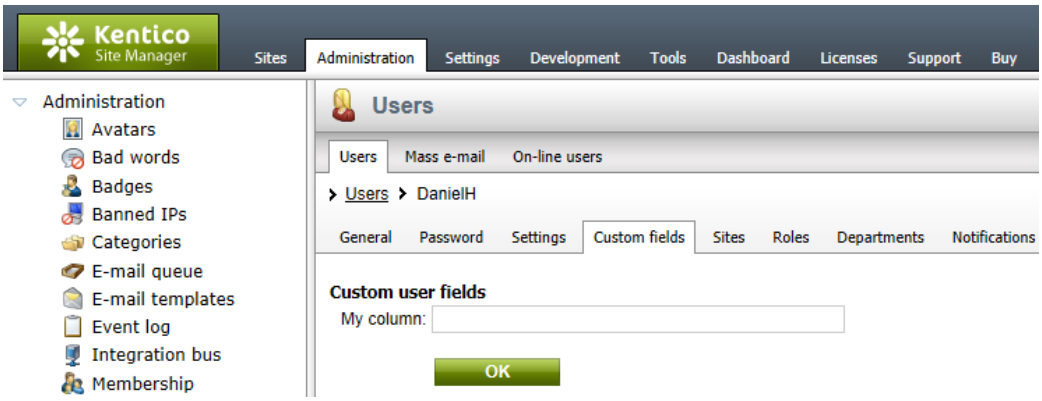
Posting a link to an article to Twitter and Facebook on the Form tab in CMS Desk

7.24 System tables and custom fields

7.24.1 Overview

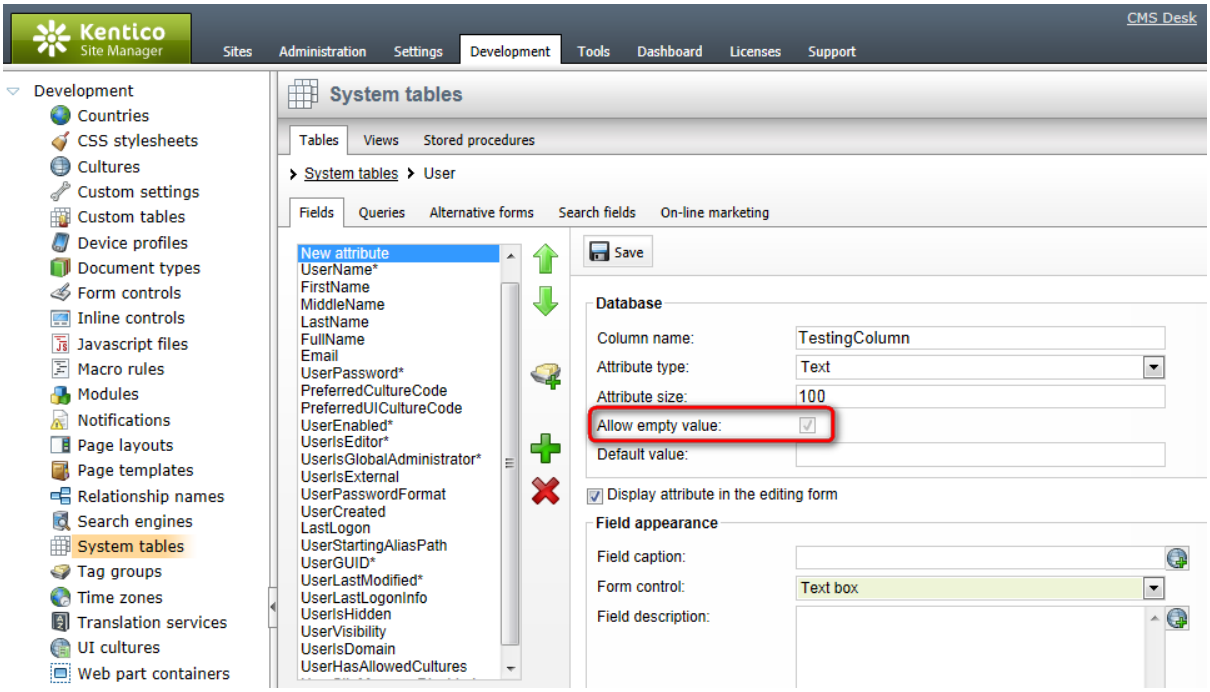
Kentico CMS allows you to modify some of the system tables and enhance them with custom attributes. You can edit them in **Site Manager -> Development -> System tables**.

If you add a new column to a system table, it is available on the **Custom fields** tab of the corresponding dialog. For example, if you add a custom column to the *CMS_User* table, it will be displayed in **Site Manager -> Administration -> Users -> Edit (🖉) *user*** on the **Custom fields** tab.



When creating a new field in a system table, the **Allow empty value** property is set to *TRUE* and cannot be edited. This is necessary to ensure that the default system procedures continue to work correctly.

However, you can change this value to *FALSE* to make the field required in alternative forms.



If you would like to learn how to add custom data to all documents, please refer to the Custom document data topic.

7.24.2 Custom document data

In some cases, you may need to add custom data to all documents. In this case, you can use the *NodeCustomData* or *DocumentCustomData* (culture specific) fields in the *CMS_Tree* and

CMS_Document database tables, respectively.

These fields are accessible through the following properties of the document (TreeNode):

- *TreeNode.NodeCustomData*
- *TreeNode.DocumentCustomData*

You can use these values in two ways:

1. You can use them as a single ntext block of text:

[C#]

```
TreeNode.NodeCustomData.Value = "my value";
```

2. You can use them as a collection of custom values that are stored as an XML document:

[C#]

```
TreeNode.NodeCustomData["myproperty1"] = "my value 1";
TreeNode.NodeCustomData["myproperty2"] = "my value 2";
```

If you would like to learn how to create a new document type or modify fields of an existing one, please refer to the Document types and transformations chapter in the Development section of this guide.

7.25 Team development

7.25.1 Overview

Development in a team of co-workers can be difficult, as two people might accidentally edit the same object at the same time and unintentionally overwrite each other's work. That's why a developer can lock a certain object for editing (check out) and unlock it (check in) after finishing the edits. When an object is checked out, other developers cannot modify it.

Object locking works similarly as content locking used for documents. Object locking is, however, used on various objects with modifiable code, such as CSS stylesheets or web part containers. A list of objects that currently support this function can be found in the Supported object types topic.

You can find the details about object locking, how this feature works and where it can be turned on/off in the Object locking topic.

All these objects can also be stored in the file system and managed by an external version control system (Git, Team Foundation Server, etc.). More information can be found in the Deployment and source control topic along with the instructions on how to deploy these objects to another system.

7.25.2 Supported object types

In the following table you can see the object types that support object locking and storing to the disk instead of a database (except for E-mail templates object type which does not support storing to disk).

The **Editing interface** column contains locations within Kentico CMS interface where the particular object type can be edited. If object locking is enabled, then the **Check out** button (or **Undo checkout** and **Check in** buttons, when the object is checked out) is situated above each code editing area.

Object type	Editing interface
CSS stylesheets	• Site Manager -> Development -> CSS stylesheets -> Edit • CMS Desk -> Content -> Edit -> Properties -> General -> Edit a CSS stylesheet • other interfaces containing the stylesheet selector (e.g., when editing a site, department, etc.)
E-mail templates*	• Site Manager -> Administration -> E-mail templates -> Edit • CMS Desk -> Administration -> E-mail templates
Page layouts	• Site Manager -> Development -> Page layouts -> Edit • CMS Desk -> Content -> Edit -> Design -> Edit layout
Page templates	• Site Manager -> Development -> Page templates -> Edit -> Layout • CMS Desk -> Content -> Edit -> Design -> Edit template -> Layout • other interfaces that allow editing of page templates
Transformations	• Site Manager -> Development -> Document types -> Edit -> Transformations -> Edit • Site Manager -> Development -> Custom tables -> Edit -> Transformations -> Edit • web part properties dialogs of web parts that have transformation properties
Web part containers	• Site Manager -> Development -> Web part containers -> Edit • CMS Desk -> Content -> Edit -> Design -> Configure a web part -> Edit a Web part container
Web part layouts	• Site Manager -> Development -> Web parts -> Edit -> Layout -> Edit • CMS Desk -> Content -> Edit -> Design -> Configure a web part -> Layout

* Does not support storing to disk.

7.25.3 Object locking

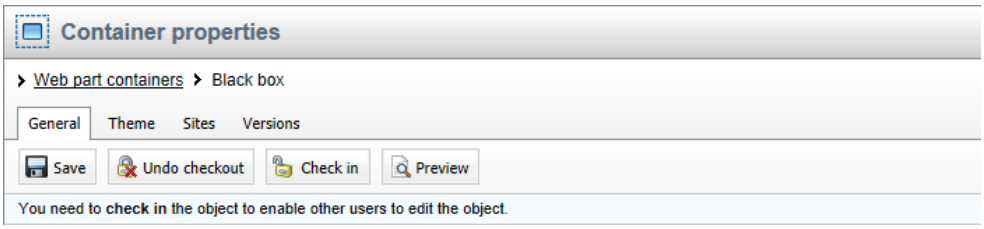
Object locking prevents developers from simultaneously overwriting each other's code modifications of objects. When the feature is enabled, only one developer (or more developers signed in under the same user name) is able to modify a certain object at a time.

Enabling object locking

The object locking feature is disabled by default. You can turn it **on** by checking the **Use check-in/ check-out for objects** option in the following location:

Site Manager -> Settings -> Versioning & Synchronization -> Object versioning

- When this option is enabled, then the **Check out** button (in some cases also the **Preview** button) is situated next to the **Save** button. After you click the **Check out** button, this button disappears and the **Undo checkout** and **Check in** buttons appear instead.



- When you disable this option, then only the **Save** button is present (in some cases also the **Preview** button).

How object locking works

Common interface for all objects that support this function:

Save - saves changes made to the object
Check out - locks an object, so that other users can not modify it at the same time.
Undo checkout - if versioning is ON: cancels the checkout and discards all changes made to the object (even the changes confirmed by the **Save** button) and also unlocks the object for other users.
- if versioning is OFF: discards changes made to the object since the last **Save** and unlocks the object for other users.

Check in - saves changes and unlocks the object for other users

If you want to edit an object, for example a CSS stylesheet (see the list of supported object types), you have to first click the **Check out** button situated above the code editing area. Until you click this button, the system will not allow you to make any modifications to the object.

When your object is checked out, you can modify it knowing, that you are the only one with this privilege. During the editing process you can save your work by clicking the **Save** button. Clicking this button also updates the system with any modifications and allows other users to view your changes to the object.

After you finish your edits, you can click the **Check in** button to confirm your changes and make the object available for other users. Or you can click the **Undo checkout** button to discard all changes made since the last checkout (even the changes confirmed by the **Save** button) and also to make the object available for other users.

Global administrator can Undo checkout on any object

There is not time limit or other restriction applied to the length of time for which you can keep an object checked out. Keep in mind though, that a **global administrator** can perform **Undo checkout** on any checked out object in the system.

Creating new versions

Every time you click the **Save** or **Check in** button, a new version of the object is created according to the rules specified in the Using object versioning topic and can be viewed on the objects' **Versions** tab.

Note that if you click the **Undo checkout** button, the version created by the **Save** button is discarded. The last version before the checkout is then set as the current version.

Typical scenarios

You want to edit an object:

1. Open the object's editing interface (for example **Site Manager -> Development -> CSS stylesheets -> Edit** (✎) a **Corporate Site**)
   - Notice, that you cannot edit the object at the moment.
2. Click **Check out**
   - The object is locked for other users. Only you can edit an object when it is checked out.
3. Edit the object
   - You can repeatedly save your work using the **Save** button.
4. Click **Check in**
   - The system confirms your modifications and unlocks the object.

You want to create a new object:

1. Click **New** <object> (where <object> can be a CSS stylesheet, page layout, transformation, container or layout)
   - The system displays the new object's dialog with only the **Save** button available.
2. Type in the object's name and all the required details
3. Edit the object's code
4. Click **Save**
   - The system saves your new object. According to the setting of option **Keep new object checked out** in **Site Manager -> Settings -> Versioning & Synchronization -> Object versioning** the object stays checked out or not.

You need to edit an object, that is currently checked out by another user:

You have several options, you can:
- wait until the user checks the object back in. You may have to refresh the object's page to see if the object is checked in again.
- contact the user (you can use the built-in interface: **CMS Desk -> My desk -> Messages**).
- ask global administrator to perform **Undo checkout**.

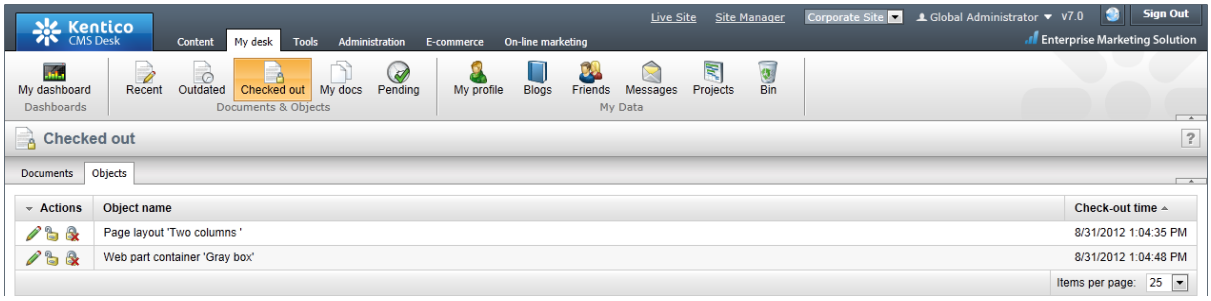
Viewing my checked-out objects

You can view all your currently checked out objects in one place:

CMS Desk -> My desk -> Checked out -> Objects

Each object's row provides three action buttons:

Edit ( ) - opens a window with the object's editing interface
Check in ( ) - checks the object in (unlocks it)
Undo checkout ( ) - discards all changes made to the object since the last checkout and unlocks the object



7.25.4 Deployment and source control

Kentico CMS provides a way to store virtual objects (listed in Supported object types) on the disk in addition to the database. Having code files on a local disk allows you to prepare your site for deployment, edit code in external editors, or manage objects using a source control system.

You can access the user interface for storing objects in the file system on the following page:

Site Manager -> Administration -> System -> Virtual objects

Deployment mode

To deploy your website to another system:

1. Click the **Store all virtual objects in file system** button.
   - The system saves all virtual objects (page layouts, page template layouts, transformations and web part layouts) on your disk.
   - The target folder is *~/CMSVirtualFiles*.

2. Compile the files in Visual Studio.
   - At this time, you can still edit the code of objects in the Kentico CMS user interface, but any changes will require you to compile the files again.

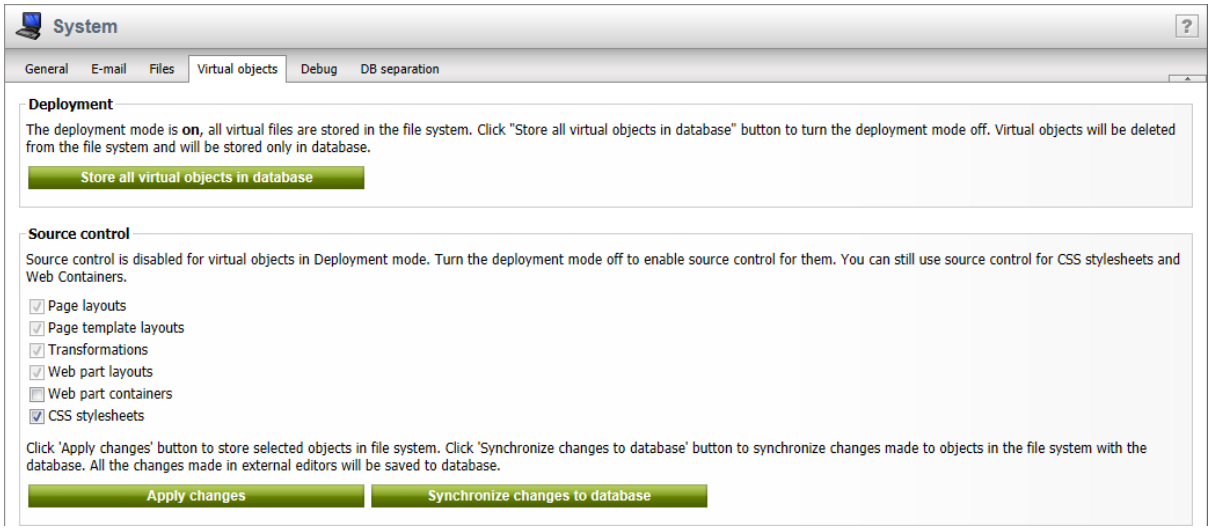
3. Move your website to the target environment.

When deployment mode is **OFF**:

- The location of the objects depends on Source control options.

When the deployment mode is **ON**

- All objects that require compilation are moved into the file system on the local disk. The target folder is *~/CMSVirtualFiles*.
- You can edit the code of objects in the user interface. Changes are saved into the files immediately.
- You can edit the object files in Visual Studio or another external editor.
- To move objects back into the database, click **Store all virtual objects in database** (deletes the files on the disk).
- You can configure the **Source control** options for objects that do not require compilation (Web part containers and CSS stylesheets).



Source control

The options in the **Source control** section allow you to select which objects are stored in the file system, where you can edit the code in external editors, or manage objects using a source control system.

- To store objects in the file system, check the boxes next to the required object types and click **Apply changes**. The file are saved in the *~/CMSVirtualFiles* folder.
- To move objects back into the database, uncheck the corresponding boxes and click **Apply changes**. Checked objects stay in the file system and unchecked objects are moved back into the database.
- Use the **Synchronize changes to database** button to copy the code from the files on the disk into the matching objects in the database.

Source control in Deployment mode

If deployment mode is **ON**, you cannot configure the source control options for objects that require compilation (only for Web part containers and CSS stylesheets).

When using source control, you can still edit the code of objects through the Kentico CMS user interface. If you edit an object, the system displays the code from the corresponding file. Saving the code in the UI writes the data into both the file system and the database.

Using deployment mode/source control on web application projects

When you enable deployment mode or source control on *web application* installations, the system cannot automatically integrate the created files into the Visual Studio project. You need to manually perform the following steps:

1. Edit your project in Visual Studio.
2. Click **Show all files** at the top of the Solution Explorer.
3. Right-click the **CMSVirtualFiles** folder and select **Include in Project**.
4. (For *Deployment mode* only) Right-click the **CMSVirtualFiles\WebPartLayouts** folder and select **Convert to Web Application**.
5. Build the **CMSApp** project.

The code files are now integrated in your web application project, and you can edit them in Visual Studio inside the *CMSVirtualFiles* folder.

7.26 UI cultures and localization

7.26.1 Overview

The user interface culture determines the language in which the administration interface is displayed, as well as other factors like numeric and date formats. UI cultures can be managed and added in **Site Manager -> Development -> UI Cultures**. The strings that contain the exact text displayed for specific UI cultures are stored in resource files in the **CMSResources** folder under your web project.

A UI culture can be set for each user in **Site Manager -> Administration -> Users** -> edit (✎) a user -> **General** -> **Preferred user interface culture**.

The default UI culture can be set by adding the following key to the <*appSettings*> section of your site's *web.config* file:

```
<add key="CMSDefaultUICulture" value="en-nz" />
```

The value of this key must be a valid culture code as in the example above (*en-nz* represents the **English - New Zealand** culture).

If you wish to change the default UI culture, you also need to rename the **~\CMSResources\CMS.resx** file to **CMS.en-us.resx** and the **CMS.en-nz.resx** file to **CMS.resx**. This is needed because the *CMS.resx* file is used when the *(default)* option is selected as a user's **Preferred user interface**

culture.

When the above mentioned key is used and the CMS.resx file contains the en-nz dictionary, the UI culture will be *en-nz* for users who have their **Preferred user interface culture** set to **(default)**.

Learn more about configuring a multilingual UI the Configuring multilingual and RTL UI.

Multilingual website content

Content of Kentico CMS websites (i.e. documents in the content tree) can also be multilingual. To learn about localization of website content, please refer to the Content management -> Multilingual content chapter of this guide.

7.26.2 Configuring multilingual and RTL UI

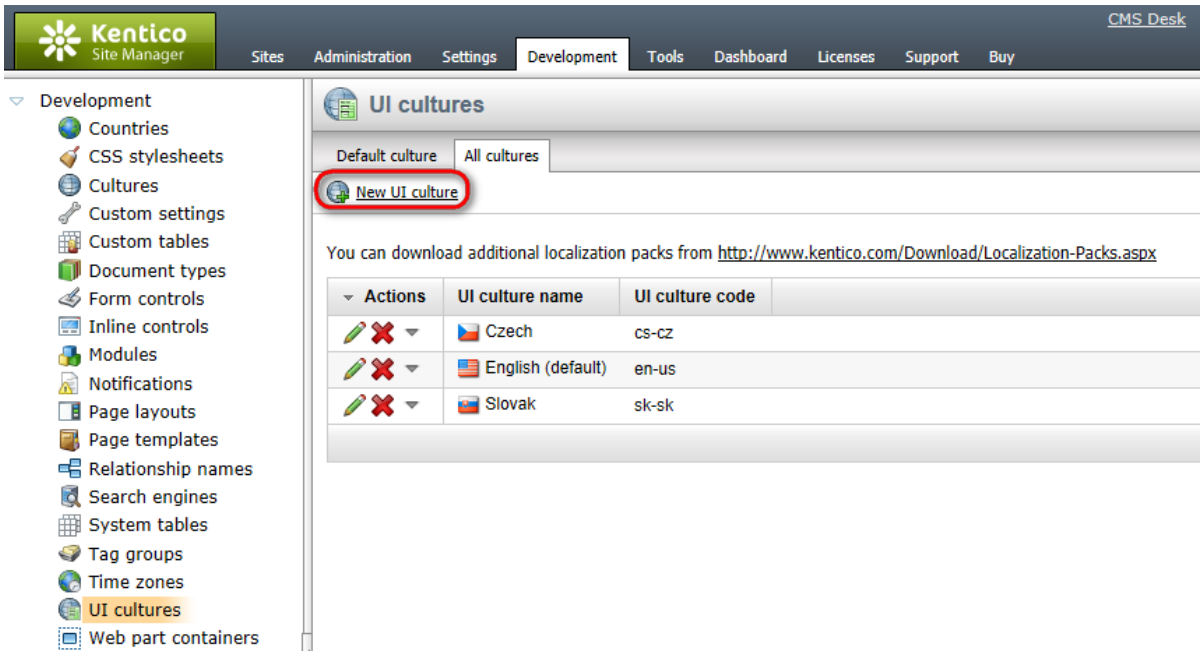
Kentico CMS allows you to manage content in any language, including double-byte (eastern) languages, such as Chinese, and right-to-left languages (such as Hebrew or Arabic). All content is stored and published in UNICODE.

Translating the administration interface

To have the administration interface displayed in a different language or at least with different culture settings (e.g. calendar and numeric format):

1. Go to **Site Manager -> Development -> UI cultures -> All cultures**.

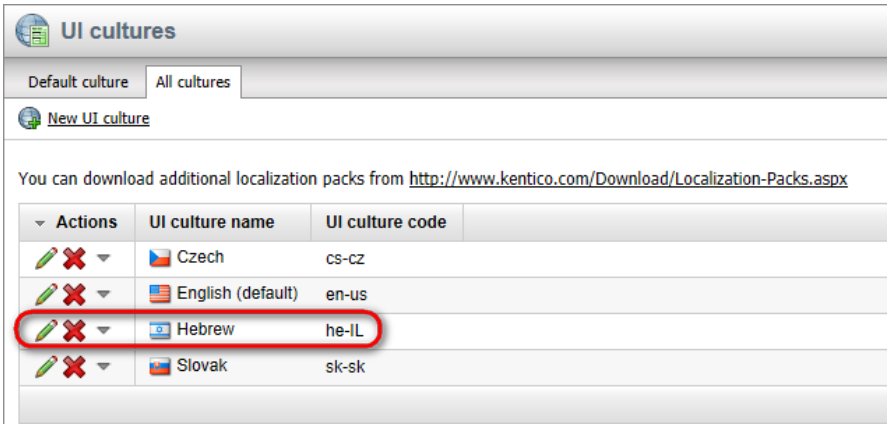
2. Add a **New UI Culture**.



3. Enter the following properties:

- **UI culture name**: Hebrew (example - you can use any other culture)
- **UI culture code**: he-IL (example - you can use any other culture code)

Click **OK**. If you switch back to the **All cultures** tab, the added culture is now shown in the list.



4. Create a copy of the **<web project>\CMSResources\cms.resx** file in the same folder and name it **cms.he-IL.resx** (**cms.<culture code>.resx** in general).

5. You can start translating the strings that are used to display text in the user interface. Localization packs that contain translated resource files are available for some languages and can be downloaded using the link on the **All cultures** tab.

Please note

After modifying a *.resx* file, you need to restart the application to apply the changes to the user interface:

1. Go to **Site Manager -> Administration -> System**
2. Click **Restart application**.

Modifying the default strings

If you want to modify some text in the user interface (including web part dialogs), you can create a **custom.resx** file and store your strings in this file. The key used to identify the string must be the same as in the **cms.resx** file. This procedure allows you to modify the strings without worrying that your changes will be overwritten during an upgrade to a newer version.

If you need to customize strings in a non-English resource file, your custom file must use a name like **custom.fr-fr.resx** for French.

Adding your own strings

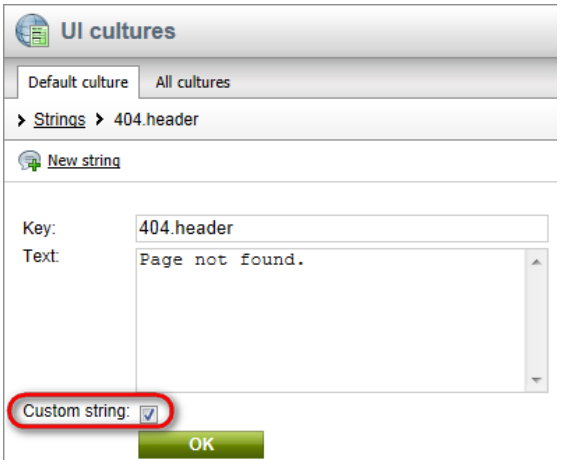
If you need to translate strings used on your website such as form labels, display names of objects or other static text into several languages:

1. Go to **Site Manager -> Development -> UI cultures -> Default culture** and click **New string**.

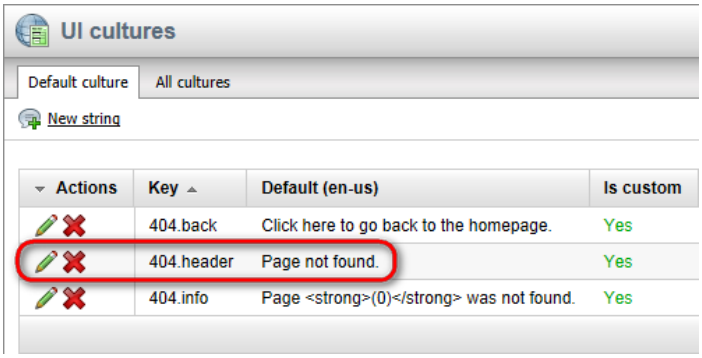
2. Enter the following properties:

- **Key**: 404.header (example - you can use any other key name)
- **Text**: Page not found. (example - you can use any other text)

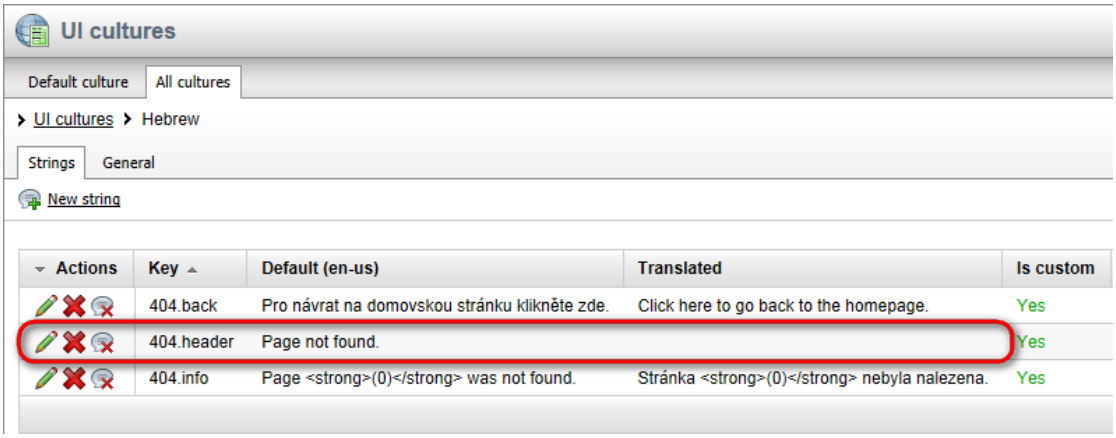
Please be sure to check the **Custom string** check box in this case, so that the string will automatically be exported with your website.



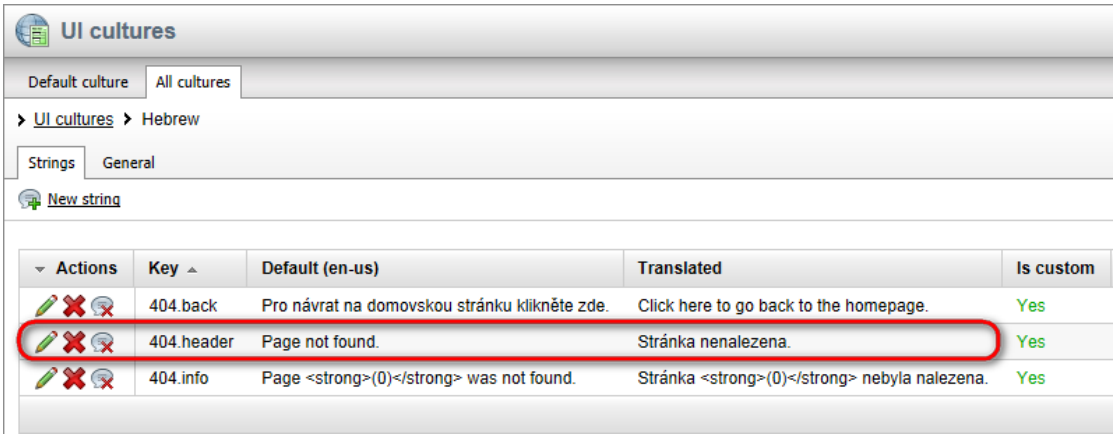
Click **OK**. A new string in the default culture is now displayed in the list.



3. To translate the newly created custom string into the desired language, switch to the **All cultures** tab and choose to **Edit** () the corresponding UI culture. A list of strings in the default language will be displayed.



4. **Edit** () the custom string created in step 1 and translate it into the desired language using the **Text** field of the string. Do not forget to check the **Custom string** check box and click **OK**. When you to edit this particular UI culture again, the string is now displayed with its translation.



You can create a new string in both the default and currently edited UI culture at the same time by using **New string** on this tab.

3. The resource string and its translation are now created and stored in the database. Please see the Localization expressions topic to see how you can insert localized strings into text fields throughout the interface of the CMS. If you need to retrieve the value of a resource string in your custom code, you can use the **CMS.GlobalHelper.ResHelper.GetString** method.

Resource string priority

When looking for a localized strings, the system uses the following priority:

1. database (Site Manager -> Development -> UI Cultures)
2. custom.resx
3. cms.resx

If there are duplicate strings with the same key in all three sources, the system will use the one stored in the database.

To change the priorities, you can add the following key to your web.config:

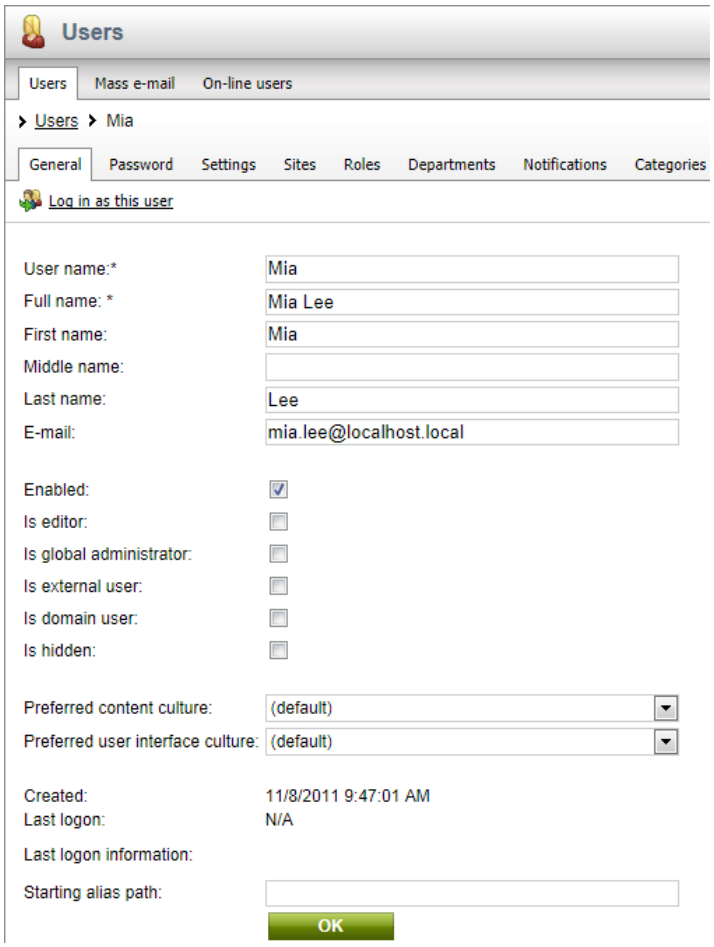
```
<add key="CMSUseSQLResourceManagerAsPrimary" value="false" />
```

When this key is added, the priorities are as follows:

1. custom.resx
2. cms.resx
3. database (Site Manager -> Development -> UI Cultures)

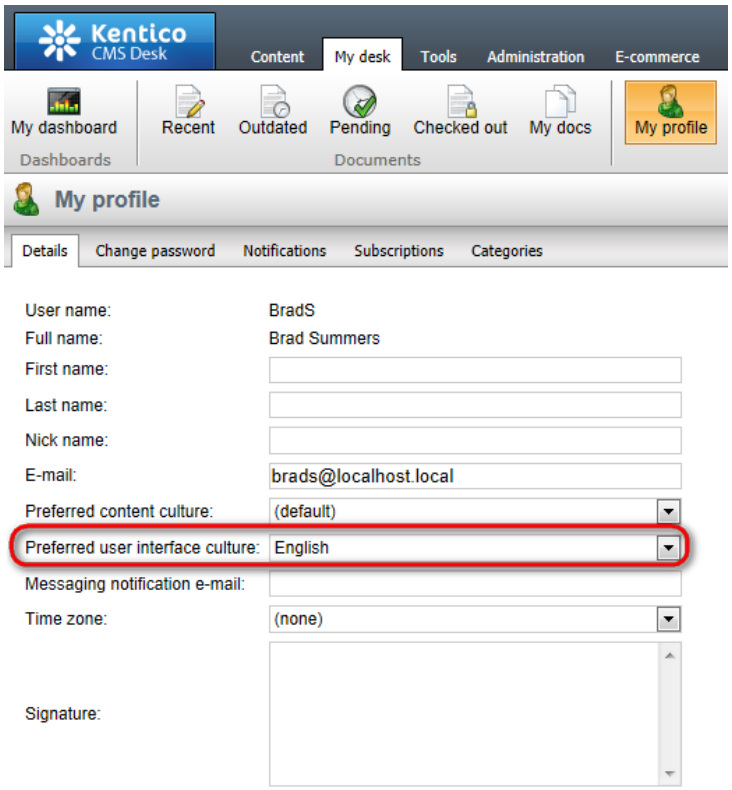
Applying a culture to the user interface

The user interface can be set to a specific culture for each user in the system. To do this, go to **CMS Desk / Site Manager -> Administration -> Users**, edit (✎) the given user and select the required value from the **Preferred user interface culture** drop-down list on the **General** tab.

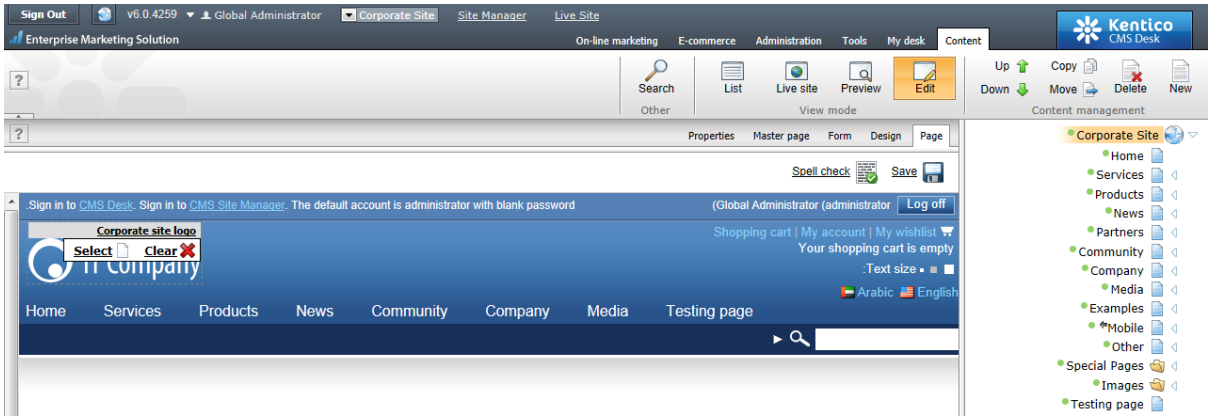


Users may also select their own user interface culture by going to **My Desk -> Account -> Details** and

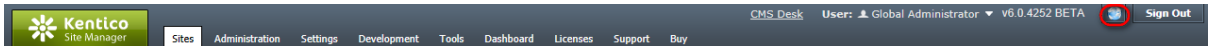
setting the **Preferred user interface culture**.



When the user signs out and then back in, the user interface will be displayed according to the new culture settings and with translated strings (once the translation is complete).



The culture of the user interface can also be selected for the current user by clicking the **Change UI culture** button located on the top right of the **CMS Desk** or **Site Manager** main header.



This selection also changes the **Preferred user interface culture** setting of the given user.

7.26.3 RTL languages

If the website is displayed in language that uses right-to-left direction of written text (typically arabic languages), the RTL CSS class is automatically assigned to the <body> element of all pages (<body class="RTL">). You may need to add right-to-left specific CSS style modifications to the stylesheet.

RTL in UI culture vs. content culture

The **user interface direction** (RTL or LTR) is driven by the **preferred UI culture** of the current user. The **content direction** is driven by the **preferred (content) culture** of the current user.

Example:

Original LTR style:

```
.xxx
{
text-align: left;
float: left;
border-right: solid 1px #cccccc;
}
```

RTL style:

```
.RTL .xxx
{
text-align: right;
float: right;
border-right: none;
border-left: solid 1px #cccccc;
}
```

Adding culture-specific fonts

If your culture uses specific fonts that are not available in the WYSIWYG editor, you need to configure it:

1. Open file <web project>\CMSAdminControls\CKeditor\config.js in notepad.
2. Add your font names on the following line:

```
CKConfig.FontNames = 'Arial;Comic Sans MS;Courier New;Tahoma;Times New Roman;
Verdana';
```

3. Save the file.
4. Clear the cache of your web browser (Internet Explorer: Tools -> Internet Options -> General -> Delete

files..., check the "delete all off-line content" box and click OK).
5. Close the browser and sign in to Kentico CMS Desk again. Now you should see the new font(s) in the WYSIWYG editor's font list.

7.26.4 Localization expressions

If you need to supply a localized value into a field or text area where localization expressions are supported, you can use expressions in the following formats:

Format	Description	Sample Value
Basic format: `{$key$}`	Displays value of the resource string with the specified key. Strings can be viewed and edited in **Site Manager -> Development -> UI Cultures** or in the *.resx* files under the **CMSResources** folder.	*{$myform.firstname$}*
In-place localization: `{$=default_string\|culture_code=translation\|culture_code=translation\|etc.$}`	Displays the strings defined in the expression. On the left, you can see an example that displays *Hallo* for German culture, *Ciao* for Italian and *Hello* for all other cultures (default value).	*{$=Hello\|de-de=Hallo\|it-it=Ciao$}*

See also: Development -> Macro expressions

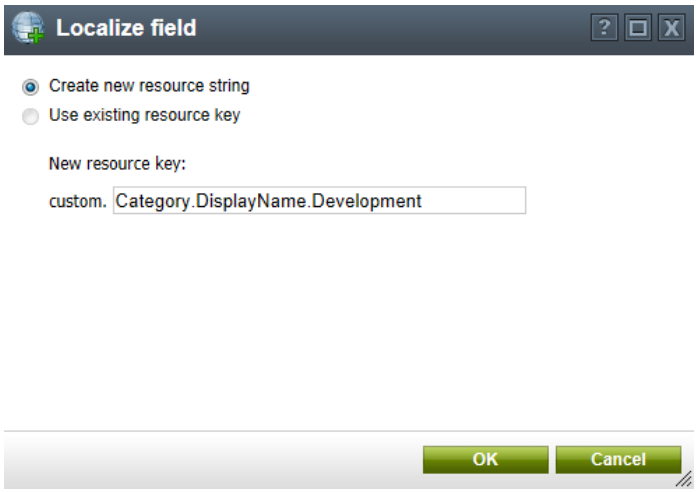
Localization dialogs

Fields in the Kentico CMS interface that contain the display names or descriptions of objects provide a more user friendly way to handle localization. This is achieved using action buttons and dialogs that allow you to insert resource strings and edit their text and translations directly from the given section of the UI.

To add a resource string to a field where this functionality is supported:

1. Click the **Localize** ( ) button.

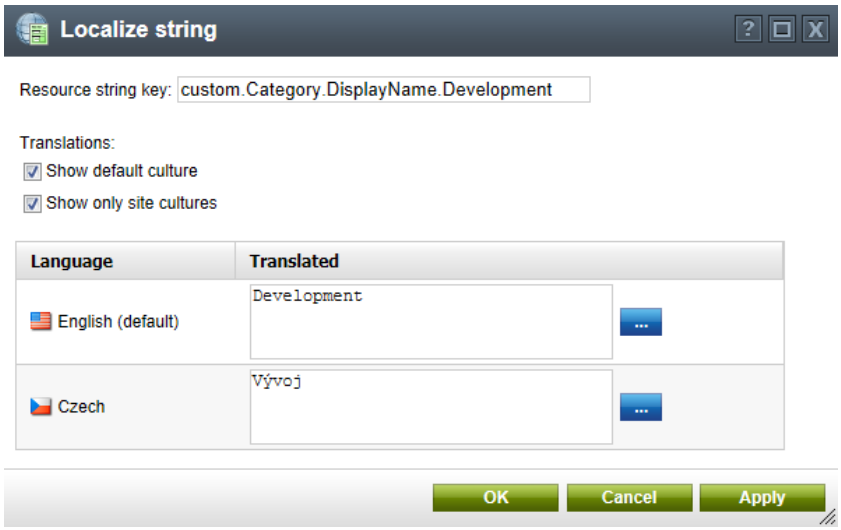
2. A dialog opens where you can create a new string or select an existing one for the field.



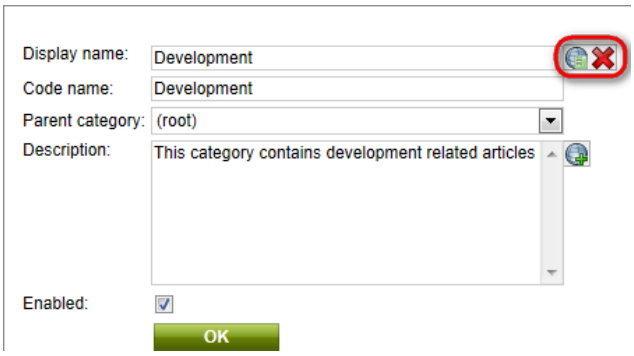
3. Edit the text of the resource string for individual cultures. You can use machine translation services to assist with the translation (if there are any configured for your application).

All cultures specified on the **Site Manager -> Development -> UI cultures -> All cultures** tab are available. If the **Show only site cultures** box is checked, only cultures assigned to the currently active site are visible. The **Show default culture** option can be used to ensure that the default UI culture configured for your installation of Kentico CMS is always displayed.

Please keep in mind that modifying the resource string assigned to a particular field also affects all other occurrences of the same string.



4. Click **OK** to confirm any changes made to the resource string.



5. Save the form by clicking **Save** at the top of the page to confirm that the string should be inserted into the field (some forms may use an **OK** confirmation button instead).

The field is now localized and it displays its value according to the current culture settings. You can edit the text of the resource string at any time by clicking the **Localize other languages** button. If you wish to remove the localization string from a field, click the **Remove localization** action.

Please note

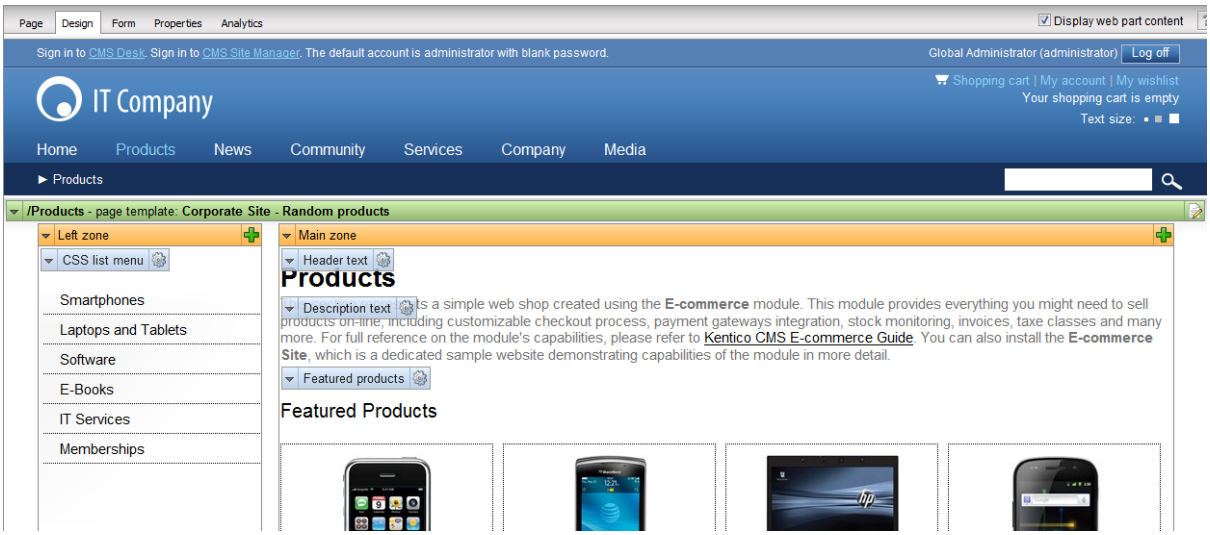
Resource strings created using localization dialogs are always stored in the database. The dialogs cannot be used to interact with strings contained in resource files (*.resx)* under the **CMSResources** folder.

7.27 Web parts

7.27.1 Overview

Web parts represent web page components that provide a combination of content and functionality. They are the basic building blocks of portal engine page templates. All web parts are registered as objects in the system and may be configured in various ways. More information can be found in the Web part management topic.

Using existing web parts, users with the appropriate permissions (**Design web site** from the **Module -> Design** permission matrix) can build or modify the structure of pages directly from a browser through the **CMS Desk -> Content -> Edit -> Design** interface. Individual web parts added onto a specific page are called web part *instances*. Each instance must be inserted into a specific web part zone, as defined by the layout of the given page's template.



To learn more about working with instances of web parts, please see the Using and configuring web parts topic in the **Portal engine development model** chapter.

From a developer's point of view, a web part is a user control (*.ascx* file) that inherits from an appropriate base class. You can easily create your own web parts according to the instructions in the Developing web parts topic.

A default Kentico CMS installation contains many built-in web parts. You can find complete documentation on all these web parts and their configuration options in the Kentico CMS Web Parts reference. In some cases however, you may find it necessary to change the behavior, design or functionality of one of them. The Modifying web parts sub-chapter covers several approaches how this can be done.

Also of interest may be Widgets, which work in a way similar to web parts. They are however intended for various types of users without website design permissions (such as content editors) and may also be used to allow personalization by users.

7.27.2 Web part management

Web parts can be managed in **Site Manager -> Development -> Web parts**. Here you can see all available web parts organized into categories in a tree structure.

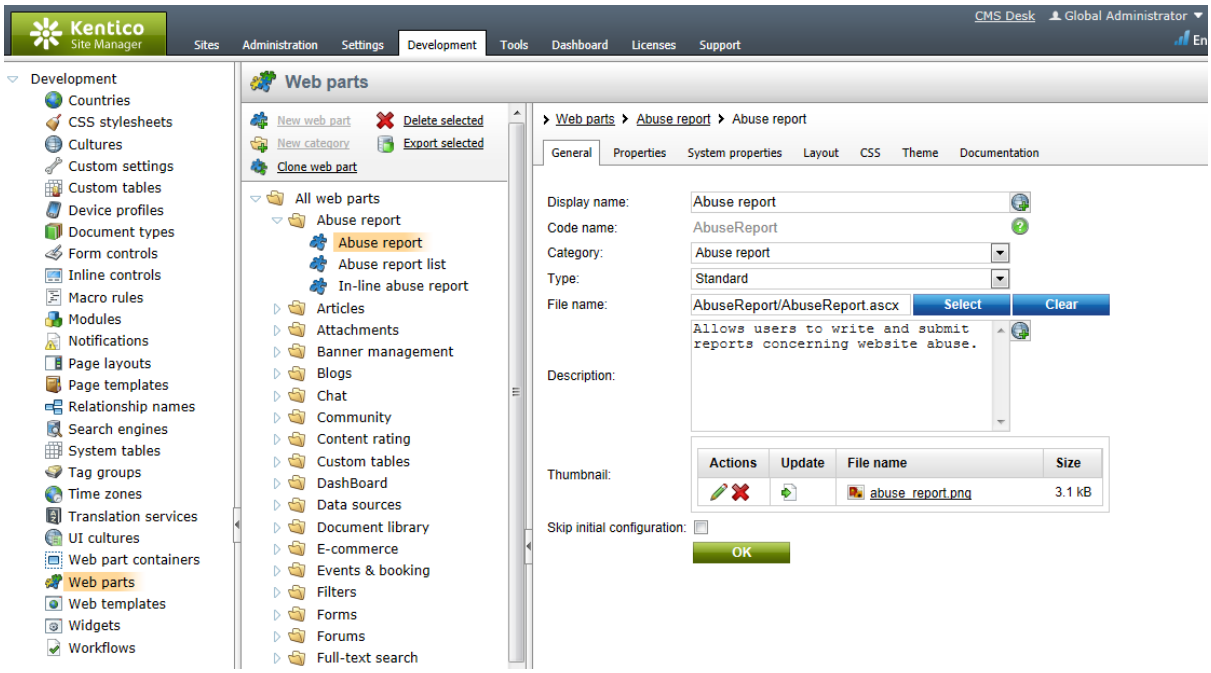
Each web part can be configured through the following properties on the **General** tab:

- **Display name** - name of the web part displayed to users in the administration interface (e.g. in the web part toolbar or selection dialog).
- **Code name** - this name serves as a unique identifier for the web part.
- **Category** - here you can choose the category of the web part catalog where the web part will be placed.
- **Type** - sets the type of the web part, which affects its behavior and properties. Different web part types are also marked with different colors and icons on the *Design* tab of *CMS Desk*. The following web part types are available:
  - **Standard** - typical web parts displaying some content.
  - **Data source** - do not display any content, only provide data to be displayed by another connected web part. More information related to data sources is available in the Data source web parts

sub-chapter.

- **Filter** - can be connected to a data source and enables users to limit the range of the data provided by it.
- **Placeholder** - used for the *General -> Layout -> Page placeholder* web part, which specifies an area where the content of sub-pages should be displayed.
- **Invisible** - are not displayed on the page at all and usually perform some type of background task.
- **Basic** - basic web parts without partial caching and AJAX UpdatePanel support.
- **Layout** - these web parts can be used to generate a specific layout for page content by defining additional web part zones. For more information, please see the Layout web parts sub-chapter.
- **Widget only** - these web parts are only intended to serve as base templates for widgets and are not available in the web part selection dialog on the *Design* tab of *CMS Desk*. Please keep in mind that changing a web part that is already on some page template to this type will not remove it or prevent it from functioning.
- **Wireframe** - special type of web parts used when defining wireframe schematics for pages. They are only for internal purposes and are not displayed on the live site. Wireframes can be created for pages on the *Wireframe* tab in the *Edit* mode of CMS Desk, either for dedicated wireframe documents or other document types that contain a wireframe definition.

- **File name** - contains a relative path to the user control that implements the web part. The path starts from the *~/CMSWebParts/* folder. It is recommended to organize web part source files on the file system in a way that matches the web part category structure. Example: *Search/cmscompletesearchdialog.ascx*
- **Description** - a text description of the web part that will be displayed in the selection catalog and as a tooltip in the web part toolbar.
- **Thumbnail** - used to upload an image that will represent the web part in the selection catalog and web part toolbar.
- **Skip initial configuration** - if checked, new instances of this web part will be placed directly onto the page without opening the property configuration dialog. This can be convenient, particularly in the case of web parts that are typically used with their default property values.



Web part properties

Since web parts often need to provide a wide range of functionality in order to cover different possible scenarios or preferences, their behavior can be configured through properties. A web part's properties are defined on the **Properties** tab and handled in the code of its source file.

The values of these properties can then be set for individual web part instances through the **Web part properties** dialog, which is displayed when a web part is added to a page or configured in CMS Desk. The user interface of each visible property is based on an object called a *form control*. This allows you to select from a large variety of built-in web part property types and provides almost unlimited customization options. Please see the Development -> Form controls chapter to learn more.

All standard web parts have the following **default properties**. These properties are loaded automatically when the web part is defined and their default values can be specified on the **System properties** tab. If you wish to modify more than the default value of one of these properties, such as its behavior and settings (e.g. for it not to be displayed in the web part configuration dialog) you can do so by overriding it on the **Properties** tab and configuring it differently.

Default

- **Web part control ID** - serves as an identifier for the web part. This ID must be unique within the context of each page template. The value of this property may only contain alphanumeric characters and the underscore character (_). It is recommended to keep the ID short to minimize the total size of the page's output code.
- **Web part title** - title of the web part displayed on the *Design* tab of CMS Desk and in On-site editing mode. If empty, the value of the *Web part control ID* property is used for this purpose.

Visibility

- **Visible** - indicates if the web part should be displayed.
- **Hide on subpages** - indicates if the web part should be hidden on sub-pages. If checked, the web part will not be displayed on documents that inherit the web part from a parent document.
- **Show for document types** - contains a list of document types on which the web part should be displayed. If the currently selected document uses the page template containing the web part, but its type is not specified by this property, the web part will be hidden. The document types in the list must be specified by their code names and separated by semicolons (;). If empty, the web part will be displayed on all document types.
- **Display to roles** - contains a list of roles to which the web part should be displayed. This may be used to implement documents with specific functionality for different types of users. The roles in the list must be specified by their code names and separated by semicolons (;). If empty, the web part will be displayed to all users.

Web part container

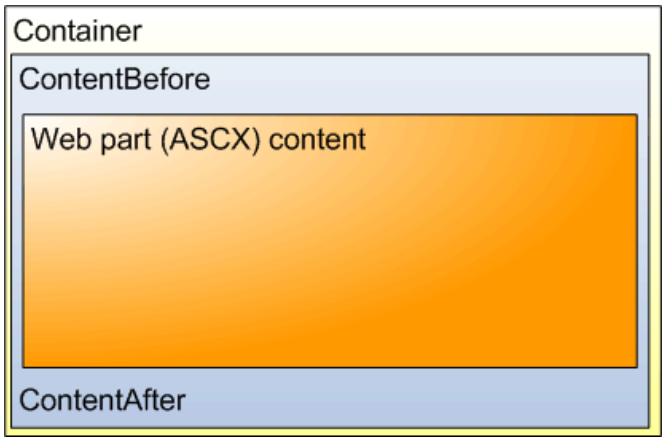
- **Web part container** - specifies the name of the Web part container (box) to be displayed around the web part. Only the containers defined in *Site Manager -> Development -> Web part containers* can be used. The currently selected container can be edited directly by using the **Edit** button.
- **Container title** - sets a title for the container. This title is displayed only if the *{%ContainerTitle%}* macro is used in the code of the container.
- **Container CSS class** - CSS class used for the web part's container. Applied only if the *{% ContainerCSSClass%}* macro is used as the value of a *Class* attribute somewhere in the code of the container.
- **Container custom content** - may be used to pass a text parameter to the specified web part container. Applied only if the *{%ContainerCustomContent%}* macro is used in the code of the container.

- **Hide container on subpages** - if enabled, the container will not be displayed around the web part on child documents that inherit it through visual inheritance. For example, this allows you to add a container for a master page only.

HTML Envelope

- **Content before** - HTML content placed before the web part. Can be used to display a header or add some encapsulating code, such as * * or *<table>* elements to achieve the required layout.
- **Content after** - HTML content placed after the web part. Can be used to display a footer or close the tags contained in the *ContentBefore* value, such as * * or *</table>* elements.

The structure of a web part, its content before/after sections and container is as follows:



Containers, unlike the *ContentBefore* and *ContentAfter* sections, are re-usable for other web part instances and may contain dynamically inserted values of web part properties.

AJAX

- **Use update panel** - indicates if an AJAX UpdatePanel container should be used for the web part. More information can be found in the AJAX support topic.

Time zones

- **Time zone** - specifies the type of time zone used for the content of the web part. The following types are available:
  - **Inherit** - inherits the time zone settings from the Page placeholder web part used to display the page template containing this web part (typically the one on the master page).
  - **Server** - server time zone settings will be used by the web part.
  - **Web site** - website time zone settings will be used by the web part.
  - **User** - time zone settings of individual users will be used by the web part.
  - **Custom** - some other time zone will be used based on the selection done in the *Custom time zone* property.
- **Custom time zone** - may be used to assign a custom time zone specifically for the content of this web part. If selected, the time zone will be used regardless of current user or website time zone settings.

Performance

- **Disable view state** - indicates if view state should be disabled for the web part.
- **Disable macros** - if checked, macros contained in the values of the web part's properties will no longer be resolved.
- **Partial cache minutes** - sets the number of minutes for which the output HTML code of the web part should remain cached. This process is similar to full-page caching, but only for the code of the web part specifically. If left empty or set to *0*, partial caching will be not be used for the web part.
- **Partial cache dependencies** - contains a list of cache keys on which the partial cache of the web part depends. When the specified cache items change, the partial cache of the web part is deleted. Each line may only contain a single item. If the *Use default cache dependencies* box is checked, the default dependencies will be used, which include all possible object changes that could affect the specific web part.

Output filter

- **Filter output HTML of web part** - if checked, the output code generated by the web part is processed by the output filters enabled through the properties below.
- **Resolve URLs** - indicates if the output filter should remove the "~" character in relative URLs and replace it with the root URL of the website (including the application's virtual directory if applicable).
- **Fix attributes** - if checked, the filter ensures that all attributes of HTML tags are in valid XHTML format.
- **Fix Javascript** - if checked, the filter ensures that the *type* and *language* attributes are included in all <*script*> tags.
- **Fix lower case** - if checked, the filter ensures that all HTML tags and attributes are generated in lower case.
- **Fix self closing tags** - if checked, the filter ensures that all HTML elements without closing tags are properly closed, e.g. <*br*> will be replaced by <*br* />.
- **Fix tags** - if checked, the filter replaces tags that are not XHTML valid with appropriate equivalents (<*strong*> instead of <*b*>, <*em*> instead of <*i*>).
- **Fix HTML5** - if checked, the filter replaces attributes that are obsolete in HTML5 with CSS classes named in format <*attribute name*>_<*attribute value*>. These classes need to be defined in the CSS stylesheet used by the page or specific web part. The affected attributes are: *cellpadding*, *cellspacing*, *width*, *height*, *border*, *align*, *valign*
- **Convert TABLE tags to DIV tags** - if enabled, <*table*> elements and their child <*tr*> and <*td*> tags are automatically converted to <*div*> elements with CSS classes assigned, named according to the replaced tag. These classes need to be defined in the CSS stylesheet used by the page or specific web part.

Web part layouts

Layouts allow you to customize the appearance of a web part or add further content into it. You may create or edit a web part's layouts on its **Layouts** tab. Designers can choose which layout they wish to use for each specific instance of the web part. More information and an example can be found in the Modifying web parts -> Customizing web part layout topic.

Web part CSS styles

You can define any CSS classes needed to correctly display a web part by editing it in on the **CSS** tab. If the styles require any files (such as images), you can add them on the **Theme** tab. This stylesheet will then be automatically requested on all pages where the given web part is placed (in addition to the website or page-specific stylesheet). For more information about page component CSS styles, please see the Development -> CSS stylesheets and design -> CSS for page components topic.

If you need to ensure that a certain CSS class is applied to the content of a web part, there are several options. You can simply use a container or the *Content before / after* properties to enclose the web part into a HTML element containing the appropriate class. Alternatively, you may define a new property for any web part using the **Column name** *CssClass* and **Attribute type** *Text*. Entering the name of a CSS class from your stylesheet as this property's value will automatically wrap the given web part instance into a <Div> element applying the specified class.

Web part documentation

You can add your own documentation to a web part on the **Documentation** tab. If you wish to document specific properties, you need to fill in the **Field description** on the **Properties** tab.

To generate full web part documentation in a printable format, enter *<website URL>/CMSPages/Dialogs/documentation.aspx?allwebparts=true* into your browser. It is recommended to use FireFox for correct formatting and page breaking.

7.27.3 Developing web parts

In cases where the web parts included in Kentico CMS by default do not meet the requirements of your specific scenarios, it is possible to develop new ones and register them in the system. This way, you can add any type of custom content or functionality to your pages through the portal engine.

All web parts are implemented as user controls. Standard web parts must inherit the **CMSAbstractWebPart** class from the **CMS.PortalControls** namespace. An exception to this rule are web parts that provide content editable on the Page tab of CMS Desk and in On-site editing mode (such as the default *Editable text* and *Editable image*), which need to inherit from the **CMSAbstractEditableWebPart** class instead.

Example

The following example will guide you through the process of creating a very simple "Hello world" web part that displays a label and a button. When the button is clicked, the current time will be displayed by the label.

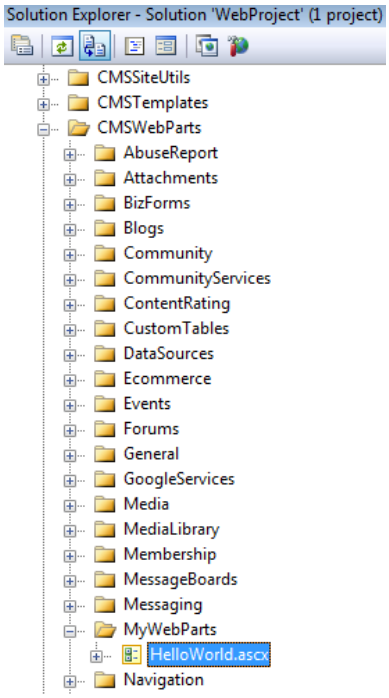
Creating the web part's code files

1. Open the web project in Visual Studio (or Visual Web Developer) using the **WebProject.sln** (or **WebApp.sln**) file or using **File -> Open -> Web site...** in Visual Studio.

2. Right-click the **CMSWebParts** folder in the **Solution Explorer** window and choose **New Folder**. Name the folder **MyWebParts**.

3. Right-click the **MyWebParts** folder and choose **Add New Item**.

4. Create a new **Web User Control** named ***HelloWorld.ascx***.

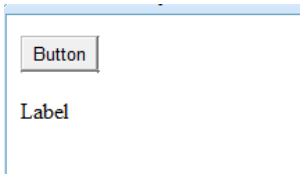


5. In the **Control** declaration, enter the full relative path of the control's code behind file into the **CodeFile** attribute (**CodeBehind** attribute on web application installations).

```
<%@ Control Language="C#" AutoEventWireup="true" CodeFile="~/CMSWebParts/
MyWebParts/HelloWorld.ascx.cs" Inherits="CMSWebParts_MyWebParts_HelloWorld" %>
```

6. Switch to the **Design** view and drag the following controls onto the form from the toolbox:

- **Button**
- **Label**



7. Double-click the Button control and add the following code to the **Button1_Click** method:

[C#]

```
Label1.Text = DateTime.Now.ToString();
```

[VB.NET]

```
Label1.Text = DateTime.Now.ToString()
```

8. Add the following line to the beginning of the code:

[C#]

```
using CMS.PortalControls;
```

[VB.NET]

```
Imports CMS.PortalControls
```

9. Change the following line:

[C#]

```
public partial class CMSWebParts_MyWebParts_HelloWorld : System.Web.UI.UserControl

to

public partial class CMSWebParts_MyWebParts_HelloWorld : CMSAbstractWebPart
```

[VB.NET]

```
Partial Class CMSWebParts_MyWebParts_HelloWorld
    Inherits System.Web.UI.UserControl

to

Partial Class CMSWebParts_MyWebParts_HelloWorld
    Inherits CMSAbstractWebPart
```

This ensures that the user control inherits from the correct base class and behaves as a web part.

10. Add the following code to the **Page_Load** method:

[C#]

```
Button1.Text = (string)this.GetValue("ButtonText");
```

[VB.NET]

(Visual Basic.NET doesn't create the **Page_Load** method automatically, so you need to add the whole method:)

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Me.Load
    Button1.Text = CType(My.GetValue("ButtonText"), String)
End Sub
```

This code sets the button text to the value configured for the web part through its **ButtonText** property in Kentico CMS Desk (this property will be defined later in the example). Please notice the use of the **GetValue** method (inherited from the parent class), which allows you to dynamically load values of the web part's properties. The method's parameter must match the *Column name* of the property that you wish to retrieve.

11. Save all changes. If your Kentico CMS project was installed as a web application, you must **Build** the project.

Registering the web part in the system

1. Open **Site Manager -> Development -> Web parts**.

2. Select the root of the tree (*All web parts*) and click **New category**.

3. Enter *My web parts* into the **Category display name** field, *MyWebParts* into the **Category name** field and click **Save**.

4. Select the new category and click **New web part**.

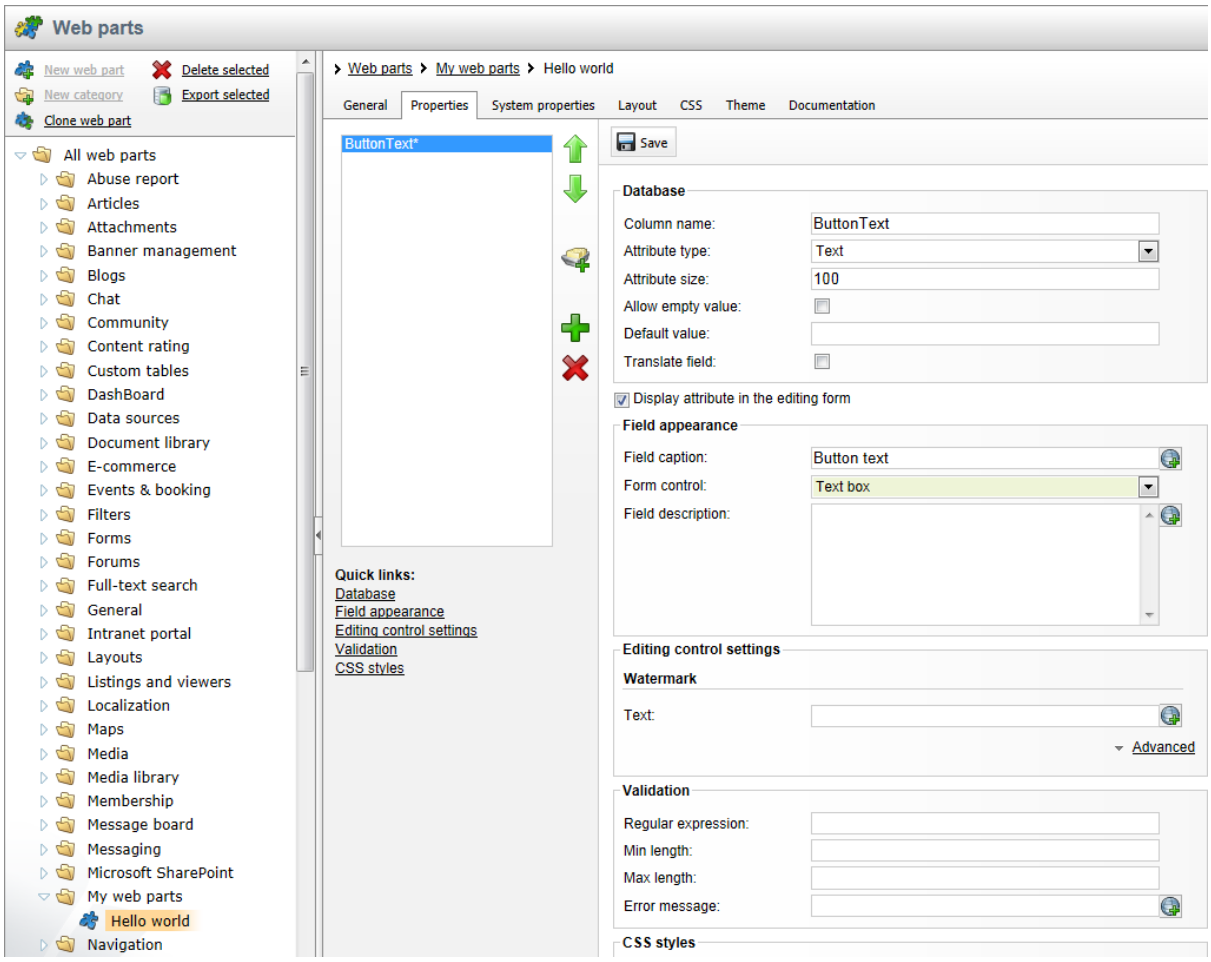
5. Choose to **Create a new web part** and enter the following values:

- **Display name**: Hello world
- **Code name**: HelloWorld
- **File path**: ~/CMSWebParts/MyWebParts/HelloWorld.ascx
- **Generate the code files**: false (the web part's source files were prepared in the previous steps, so there is no need to generate them)

6. Click **Save**.

7. Switch to the **Properties** tab and add (+) the following property:

- **Column name**: ButtonText
- **Attribute type**: Text
- **Attribute size**: 100
- **Field caption**: Button text
- **Form control**: Text box



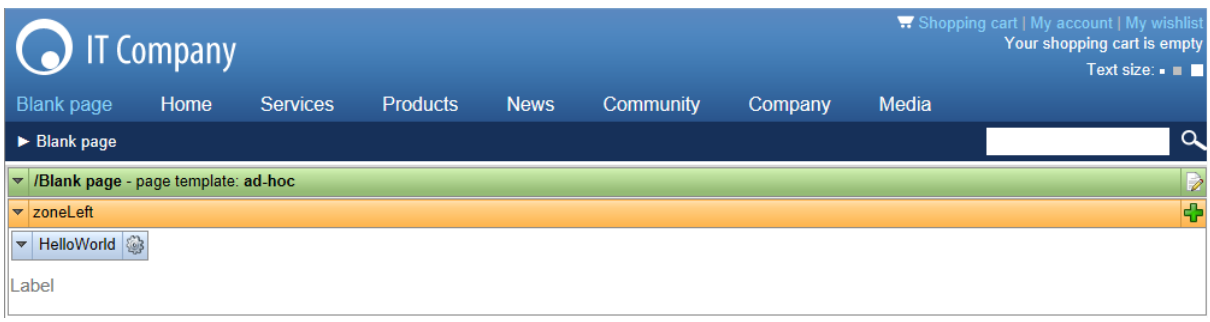
8. Click **Save**.

Adding an instance of the web part to a page

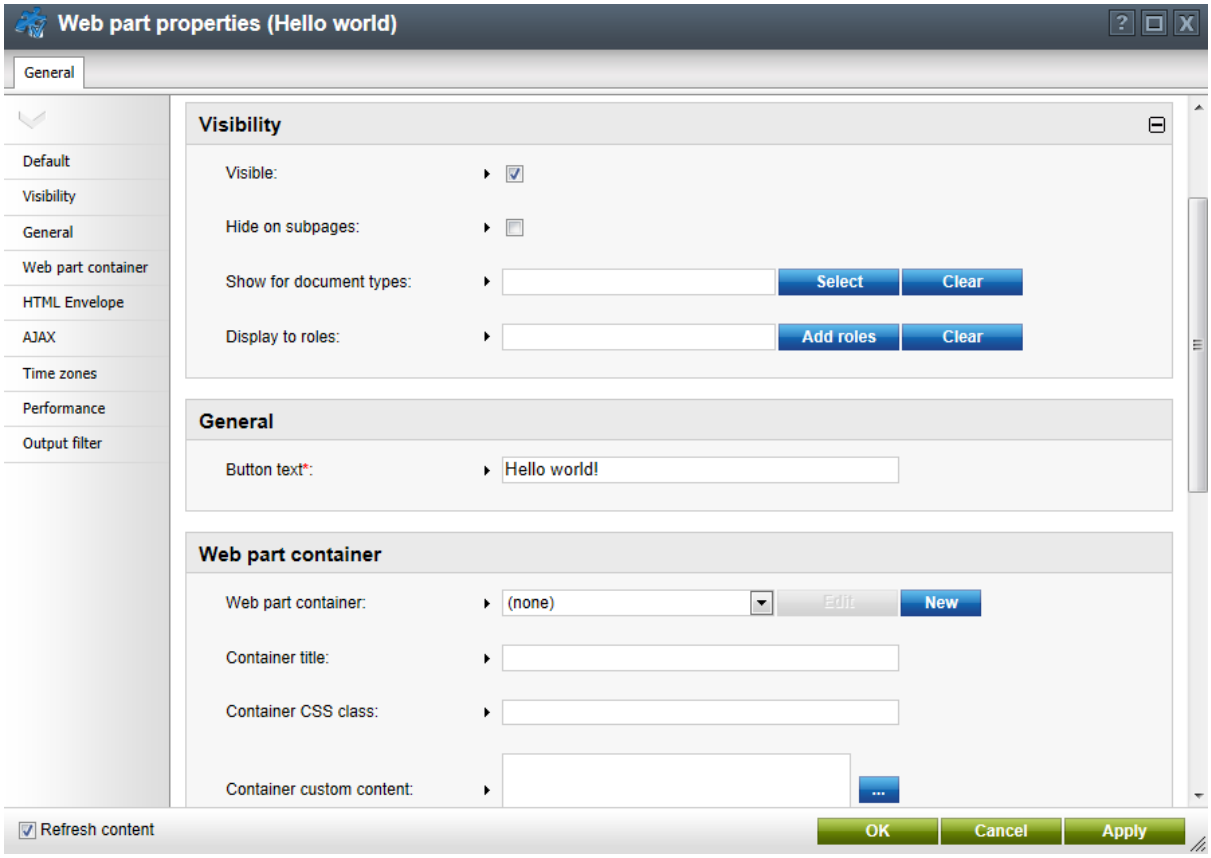
1. Switch to **CMS Desk**.

2. Create a new **blank page** using the **Simple** layout (or any other layout) under the root and switch to the **Design** tab.

3. Click **Add web part** (+) in the upper right corner of the web part zone and choose to add the **Hello world** web part:

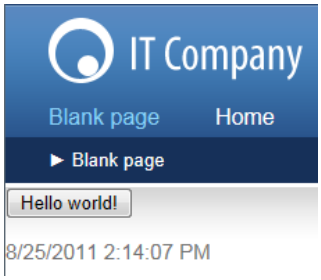


4. The Web part properties dialog of the **HelloWorld** web part will be displayed. Set the value of the **Button text** field to: *Hello world!*



5. Click **OK**.

Now switch to the **Live site** mode using the button in the main toolbar. You will see the button with text *Hello world!* When you click it, the label displays the current date and time:



You have learned how to create a simple web part. Please keep in mind that in many cases, it may be easier to achieve your goal by altering or extending one of the default web parts rather than developing an entirely new one. The Modifying web parts sub-chapter covers several ways how this can be done.

Tip: Displaying content in web parts

You can also use Kentico CMS Controls in your web parts (in the code of their ASCX control) to display content from Kentico CMS in a customized format.

Storing files related to web parts

If your web part consists of several files (such as other ASCX controls, images, js scripts, etc.), you should place these files in a sub-folder under the folder where your main web part ASCX file is placed. If the code name of the web part is **MyWebPart**, the sub-folder's name must be **MyWebPart_Files**. This will ensure that the additional files are included in the web part's export package and imported correctly when you move your website or distribute the web part to other developers.

Initializing Kentico CMS controls in your custom web parts

If you are using Kentico CMS controls in your web parts, it is recommended to initialize their properties using a combination of the **OnContentLoaded** and **SetupControl** methods. This is the way it is handled in all Kentico CMS web parts. You can view the code of any of the web parts located in *<project folder>/CMSWebParts* and use it as an example.

If you are using standard .NET controls or third party controls, this can be handled in the **PageLoad** method.

If a problem occurs (e.g. on postback), try loading the control dynamically. This can be achieved by converting the web part to a standard control and loading using the **General -> User control** web part).

Error message "The control collection cannot be modified during DataBind, Init, Load, PreRender or Unload phases."

If you get this error message you may need to modify the code of your web part, so that it doesn't display any content on the Design tab - for example:

[C#]

```
using CMS.PortalEngine;

public override void OnContentLoaded()
{
  base.OnContentLoaded();
  if ((this.PagePlaceholder.ViewMode == ViewModeEnum.Design)
```

```
     || (this.HideOnCurrentPage) || (!this.IsVisible))
    {
         this.Repeater1.DataSourceID = "";
         this.CMSRepeater1.StopProcessing = true;
    }
}
```

7.27.4 Modifying web parts

7.27.4.1 Modifying web parts

If you need to modify the behavior of a standard web part, you may choose from the following options, depending on your goals and requirements:

1. **You only need to set a web part's properties dynamically in your code**

You can create a user control, add the given web part to it and write additional code. See the Setting web part properties dynamically in your code topic.

2. **You need to modify the design (layout) of a web part**

You can use the custom web part layouts described in the Customizing web part layout topic.

3. **You need to modify the code of a web part**

You need to create a copy of a standard web part as described in the Modifying the code of standard web parts topic.

4. **You need to create a specialized version of a web part with different default property values**

You can use Web part inheritance to easily create a derived web part.

7.27.4.2 Setting web part properties dynamically in your code

In some cases, you may need to set the values of web part properties in your code, depending on some particular business rules. In such case, you need to create a new ASCX user control and place the original web part onto this user control. In the user control code, you can implement your custom logic and set the properties appropriately.

Example:

The following example shows how you can dynamically set the **WHERE condition (WhereCondition)** property of the **Repeater** web part based on if the current user is or is not authenticated. It uses the standard *News* document type with a custom boolean type field: **Show to public users (ShowToPublicUsers)**.

1. Open the web project in **Visual Studio**.

2. Create a **New folder** under the project root called *CMSGlobalFiles* (if it doesn't already exist). This location will ensure that your user controls can be exported along with the site when it is deployed to the

live server.

3. Create a new **Web User Control** under the **CMSGlobalFiles** folder and name it **NewsRepeater.ascx**.

4. Switch to the **Design** tab and drag and drop **CMSWebParts/Viewers/Documents/cmsrepeater.ascx** from the Solution Explorer onto your user control. You could alternatively use the CMSRepeater server control, but this is not the purpose of this example. Set its properties like this:

- **ID:** RepeaterWebPart1
- **ClassNames:** cms.news (document types)
- **Path:** /news/%
- **TransformationName:** cms.news.preview
- **SelectedItemTransformationName:** cms.news.default

5. Now add the following to the code behind of your user control inside the **CMSGlobalFiles_NewsRepeater** class:

[C#]

```
protected void Page_Init(object sender, EventArgs e)
{
    if (CMS.CMSHelper.CMSContext.CurrentUser.IsPublic())
    {
        // public user - show only public news
        this.RepeaterWebPart1.WhereCondition = "ShowToPublicUsers = 1";
        this.RepeaterWebPart1.ReloadData();
    }
}
```

This will set the **WhereCondition** property value dynamically depending on whether the current user is signed in. **Save** all changes. If your Kentico CMS project was installed as a web application, you must **Build** the project.

6. Go to **Site Manager -> Development -> Document types -> News**, add a **New attribute** () on the **Fields** tab and set its properties as shown below:

- **Attribute name**: ShowToPublicUsers
- **Attribute type**: Boolean (Yes/No)
- **Field caption**: Show to public users
- **Field type**: Check box

7. Go to **CMS Desk -> Content**, choose **Home**, switch to the **Design** tab and add () a new **General -> User control** web part to the *Main zone* zone. Set the **User control virtual path** property value to: *~/CMSGlobalFiles/NewsRepeater.ascx*

8. Edit some news document in the **/News** section of the website on the **Form** tab and check the **Show to public users** checkbox.

9. **Sign out** and view the home page. You should see only news items that you marked as **Show to public users**.

You have learned how to dynamically set web part properties based on your custom logic.

7.27.4.3 Customizing web part layout

Layouts allow you to customize the appearance of web parts or add content. Web part layouts are custom skins that replace the HTML code and ASPX markup of the web part.

- Each web part has its own list of layouts.
- You can select a different layout for every instance of the web part.

To select the layout for specific web part instances:

1. Go to CMS Desk.
2. Edit the document containing the web part instance on the **Design** tab.
3. Configure ( ) the web part instance.
4. Switch to the **Layout** tab of the web part configuration dialog.
5. Select an existing layout or write a *(New)* layout.
   - The *(Default)* layout uses the markup of the web part's source file.

You can also manage the layouts of web parts in **Site Manager -> Development -> Web parts** by editing web parts on the **Layout** tab.

Requirement

The **Control** declaration in the markup of the web part's user control file must contain the ***full relative path*** to the code behind file in the **CodeFile** attribute (**CodeBehind** attribute on web application installations). For example:

```
<%@ Control Language="C#" AutoEventWireup="true"
Inherits="CMSWebParts_Navigation_cmslistmenu"
CodeFile="~/CMSWebParts/Navigation/cmslistmenu.ascx.cs" %>
```

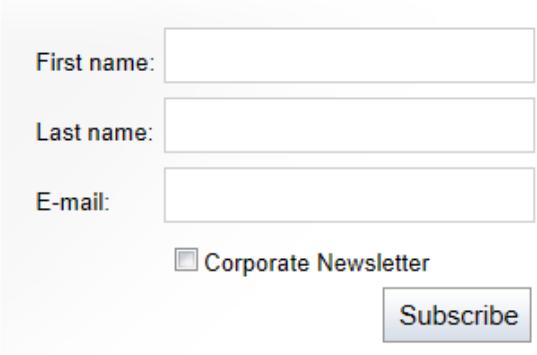
Do not remove controls from the layout

Keep all of the default controls inside the code of web part layouts to ensure that web parts work correctly. If you need to hide any of the controls, add the ***Visible="False"*** attribute.

Note: Your custom web part layouts may not work after upgrading to a new version of Kentico CMS (if the controls in the web part's markup changed). Please test your website carefully after upgrading, and update your custom layouts according to the controls in the default layout if necessary.

Example: Customizing the Newsletter subscription dialog

In this example, we will customize the newsletter subscription dialog layout. The standard layout looks like this (when no user is logged on):



Go to **CMS Desk -> Content** and navigate to the **Examples -> Web parts -> Newsletters -> Newsletter subscription** page (if you're using the sample Corporate Site).

Switch to the **Design** tab and configure ( ) the **Newsletter subscription** web part. Click the **Layout** tab and choose *(New)* from the drop-down list. Fill in the following values:

- **Display name:** Narrow layout
- **Code name:** NarrowLayout

Then enter the following **Layout code**:

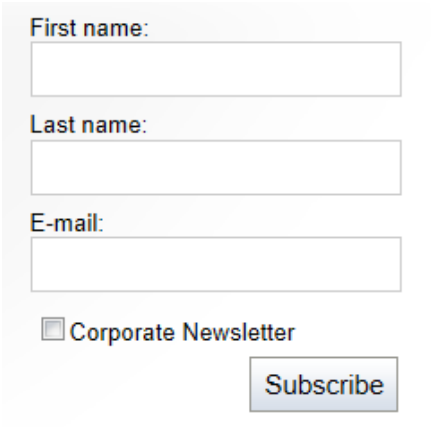
Please note: If you installed the Kentico CMS project as a web application, you need to rename the **CodeFile** attribute on the first line to *CodeBehind* for the code example to be functional.

```
<%@ Control Language="C#" AutoEventWireup="true"
Inherits="CMSWebParts_Newsletters_NewsletterSubscriptionWebPart" CodeFile="~/
CMSWebParts/Newsletters/NewsletterSubscriptionWebPart.ascx.cs" %>

<%@ Register Src="~/CMSFormControls/Inputs/SecurityCode.ascx"
TagName="SecurityCode" TagPrefix="cms" %>

<asp:Panel ID="pnlSubscription" runat="server" DefaultButton="btnSubmit"
CssClass="Subscription">
    <asp:Label runat="server" ID="lblInfo" CssClass="InfoMessage"
EnableViewState="false"
        Visible="false" />
    <asp:Label runat="server" ID="lblError" CssClass="ErrorMessage"
EnableViewState="false"
        Visible="false" />
    <div class="NewsletterSubscription">
        <table cellspacing="0" cellpadding="0" border="0" class="Table">
            <asp:PlaceHolder runat="server" ID="plcFirstName">
                <tr>
                    <td>
                        <cms:LocalizedLabel ID="lblFirstName" runat="server"
AssociatedControlID="txtFirstName"
                            EnableViewState="false" />
                    <br />
                        <asp:TextBox ID="txtFirstName" runat="server"
```

```
CssClass="SubscriptionTextbox"
                            MaxLength="200" />
                    </td>
                </tr>
            </asp:PlaceHolder>
            <asp:PlaceHolder runat="server" ID="plcLastName">
                <tr>
                    <td>
                        <cms:LocalizedLabel ID="lblLastName" runat="server"
AssociatedControlID="txtLastName"
                            EnableViewState="false" />
                    <br />
                        <asp:TextBox ID="txtLastName" runat="server"
CssClass="SubscriptionTextbox"
                            MaxLength="200" />
                    </td>
                </tr>
            </asp:PlaceHolder>
            <asp:PlaceHolder runat="server" ID="plcEmail">
                <tr>
                    <td>
                        <cms:LocalizedLabel ID="lblEmail" runat="server"
AssociatedControlID="txtEmail"
                            EnableViewState="false" />
                    <br />
                        <asp:TextBox ID="txtEmail" runat="server"
CssClass="SubscriptionTextbox"
                            MaxLength="400" />
                    </td>
                </tr>
            </asp:PlaceHolder>
            <asp:PlaceHolder runat="server" ID="plcNwsList">
                <tr>
                    <td>
                        <asp:CheckBoxList runat="server" ID="chklNewsletters"
CssClass="NewsletterList" />
                    </td>
                </tr>
            </asp:PlaceHolder>
            <asp:PlaceHolder runat="server" ID="plcCaptcha">
                <tr>
                    <td>
                        <cms:LocalizedLabel ID="lblCaptcha" runat="server"
AssociatedControlID="scCaptcha"
                            EnableViewState="false" />
                    <br />
                        <cms:SecurityCode ID="scCaptcha"
GenerateNumberEveryTime="false" runat="server" />
                    </td>
                </tr>
            </asp:PlaceHolder>
            <asp:PlaceHolder ID="pnlButtonSubmit" runat="server">
                <tr>
                    <td align="right">
                        <cms:LocalizedButton ID="btnSubmit" runat="server"
OnClick="btnSubmit_Click"
                            CssClass="SubscriptionButton"
EnableViewState="false" />
                    </td>
```

```
                </tr>
            </asp:PlaceHolder>
            <asp:PlaceHolder ID="pnlImageSubmit" runat="server">
                <tr>
                    <td align="right">
                        <asp:ImageButton ID="btnImageSubmit" runat="server"
OnClick="btnSubmit_Click"
                            EnableViewState="false" />
                    </td>
                </tr>
            </asp:PlaceHolder>
        </table>
    </div>
</asp:Panel>
```

Click **OK**. When you look at the page now, you will see that the dialog looks like this (when no user is logged on):



Layout styles

You can define any CSS classes used within the layout code by clicking the Add CSS styles link below the layout editor and adding the styles into the **CSS styles** field. If the styles require additional files (such as images), you can add them on the **Theme** tab, which is available when editing the layout in **Site Manager -> Development -> Web parts**.

For more information about page component CSS styles, please see the Development -> CSS stylesheets and design -> CSS for page components topic.

7.27.4.4 Modifying the code of standard web parts

This topic explains how you can create a copy of a standard web part and customize its behavior by modifying its code.

The following example uses the Forms -> On-line form web part as its base. It shows how you can send a custom e-mail when a form is submitted and display a confirmation message. Please note that this is only for demonstration purposes. There is a much easier way to set up notifications or autoresponders for on-line forms through the built-in functionality of the Forms module.

1. Create a copy of the *On-line form* web part in Kentico CMS. Go to **Site Manager -> Development -> Web parts** and select the **Forms -> On-line form** web part from the tree. Click the **Clone web part** action and enter the following values:

- **New object display name**: Form with custom e-mail
- **New object code name**: FormWithEmail

- **Clone web part to category**: Forms
- **Clone web part files**: yes (checked)
- **Cloned web part file name**: BizForms/formwithemail.ascx

Click **Clone**. The system creates a copy of the existing web part and its code files (ASCX and CS) using the specified names.

2. Open the web project in Visual Studio using the **WebProject.sln** (or **WebApp.sln**) file and edit **~/CMSWebParts/BizForms/formwithemail.ascx**.

Important!

If you installed Kentico CMS as a web application, the clone's code files will not be visible right away, since they are not included in the project. In this case:

1. Click **Show all files** at the top of the Solution Explorer.
2. Right-click the **formwithemail.ascx** file in the BizForms folder.
3. Select **Include in Project**.

3. Switch to the **Design** tab and drag and drop a **Label** control onto the page. Set its **ID** to *lblConfirmationMessage* and clear its **Text** property.

4. Edit the code behind file and locate the **viewBiz_OnAfterSave** handler, which is executed every time the form is saved. Insert the following code after the default content of the method:

[C#]

```
void viewBiz_OnAfterSave(object sender, EventArgs e)
{

...

    // Creates a new e-mail message.
    CMS.EmailEngine.EmailMessage msg = new CMS.EmailEngine.EmailMessage();

    msg.From = "mail@localhost.local"; // Enter any valid e-mail address.
    msg.Recipients = "mail@localhost.local"; // Use a valid e-mail address that
you can access.
    msg.Subject = "Custom form e-mail";
    msg.Body = "The value of the FirstName field: "
               + CMS.GlobalHelper.ValidationHelper.GetString(
               viewBiz.BasicForm.GetDataValue("FirstName"), "N/A");
```

```
    // Sends out the custom e-mail notification.
    CMS.EmailEngine.EmailSender.SendEmail(msg);

    // Sets the confirmation message shown in the label.
    lblConfirmationMessage.Text = "The e-mail has been sent.";
}
```

This code creates a new e-mail message, sends it out and adds text into the confirmation label. You can notice how the content of the e-mail dynamically retrieves a field value from the current form by using the **BasicForm.GetDataValue(string fieldName)** method, which may be called through the corresponding property of the BizForm control. Fill in valid e-mail addresses into the code to be able to try out the example.

Save all changes. Remember to **Build** the project if it is installed as a web application.

5. Go to **CMS Desk -> Content**, choose the **Home** page, switch to the **Design** tab and add the **Form with custom e-mail** web part to the *Main zone* zone. Set the **Form name** property to the **Contact Us** form using the **Select** button. This form should be available by default if you are working with the sample Corporate Site, and it includes the **FirstName** field used in the custom code.

6. Finally, open the live site and view the Home page. Enter some values into the form and submit it. You will see the additional confirmation message ("The e-mail has been sent.") and the specified address will receive the e-mail.

You have seen how to create a customized version of a standard web part. Using the same approach, you can modify the ASPX markup or code behind of any of the default web parts in order to alter their functionality to fit your specific requirements.

7.27.4.5 Web part inheritance

Web part inheritance allows you to create a web part that has the same properties and uses the same code as the original web part, but has different default property values. It means you can create a specialized web part from a general one.

For example: You can create a news list web part inherited from the Repeater web part that will display a list of news by default. The default values can later be modified to any other value, but the inherited (specialized) web part allows you to do things faster.

How to create an inherited web part

1. Go to **Site Manager -> Development -> Web parts**, select the **Listings and viewers** category and click **New web part**.

2. Choose **Inherit from an existing web part** and enter the following values:

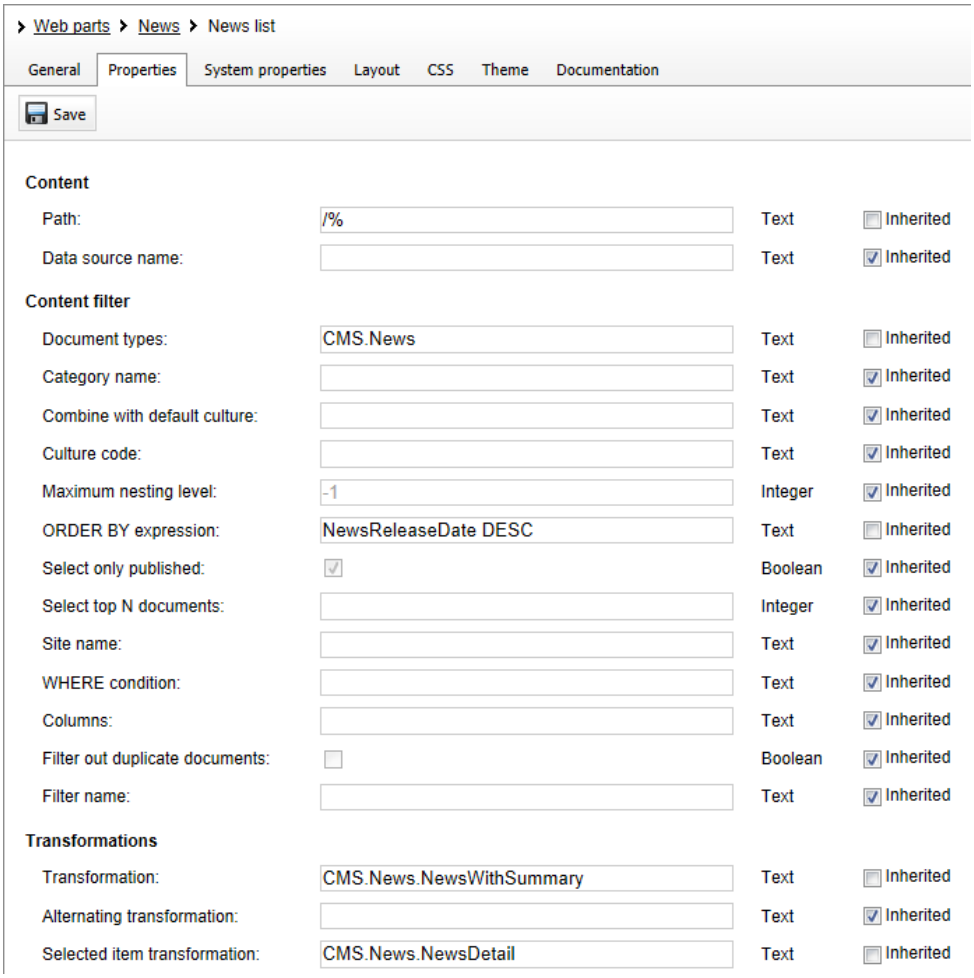
- **Display name:** Custom news list
- **Code name:** CustomNewsList
- **Inherit from:** Listings and viewers/Documents/Repeater

Click **Save**.

3. Switch to the **Properties** tab of the newly created web part. Here you can see the properties of the parent web part and you can override their default values by clearing the **Inherited** box and entering a new default value. Enter the following default values:

- **Path:** /%
- **Document types:** cms.news
- **ORDER BY expression:** NewsReleaseDate
- **Transformation:** cms.news.preview
- **Selected item transformation:** cms.news.default

Click **Save**.



4. Go to **CMS Desk**, choose the **Home** page in the content tree and switch to the **Design** tab. Add the News list web part to the page. It will now display all site news without any additional configuration.

7.27.4.6 Adding custom code to web parts (obsolete)

Obsolete feature

This feature is now obsolete. If you need to customize the behavior of some web part, please clone the web part and customize it.

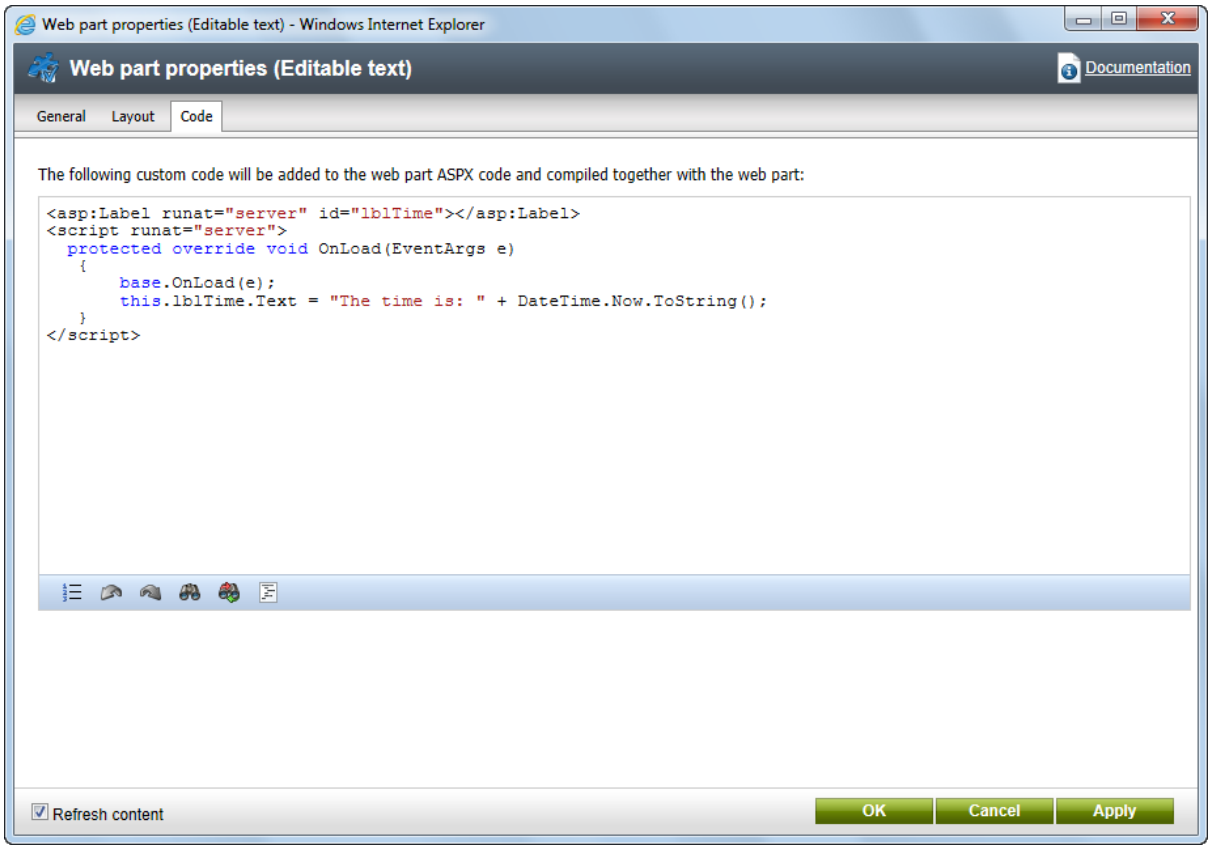
If you still need to use this feature (for backward compatibility), you need to enable it by adding the following parameter to your web.config file, to the **appSettings** section:

```
<add key="CMSShowWebPartCodeTab" value="true" />
```

Limitations when using the Code tab

If you add custom code on the Code tab, you will not be able to pre-compile the website and you will not be able to use the website in a medium trust environment. If this is an issue for you, you can use the alternative solutions described in this chapter.

If you need to modify the behavior or enhance the functionality of some web part only on a single page template, you can add custom code on the **Code** tab:



The following code displays the current date and time below the current web part:

[C#]

```
<asp:Label runat="server" id="lblTime"></asp:Label>
<script runat="server">
    protected override void OnLoad(EventArgs e)
    {
        base.OnLoad(e);
        this.lblTime.Text = "The time is: " + DateTime.Now.ToString();
    }
</script>
```

Programming language

Please note: You can only use the programming language in which the web part is written. If the web part was written in C#, you can use only C# here.

The following code sample sets the **WhereCondition** property of the web part dynamically at run-time:

[C#]

```
<script runat="server">
public override void OnContentLoaded()
{
this.SetValue("WhereCondition", "NewsTitle LIKE '%News%'");
base.OnContentLoaded();
}
</script>
```

Calling the original method from the inherited class

If you override some method of the web part, please be sure to always also call the original method (using base.Method in C# or MyBase.Method in VB). If you omit this, the web part may not work properly.

7.27.4.7 Web part binding (obsolete)

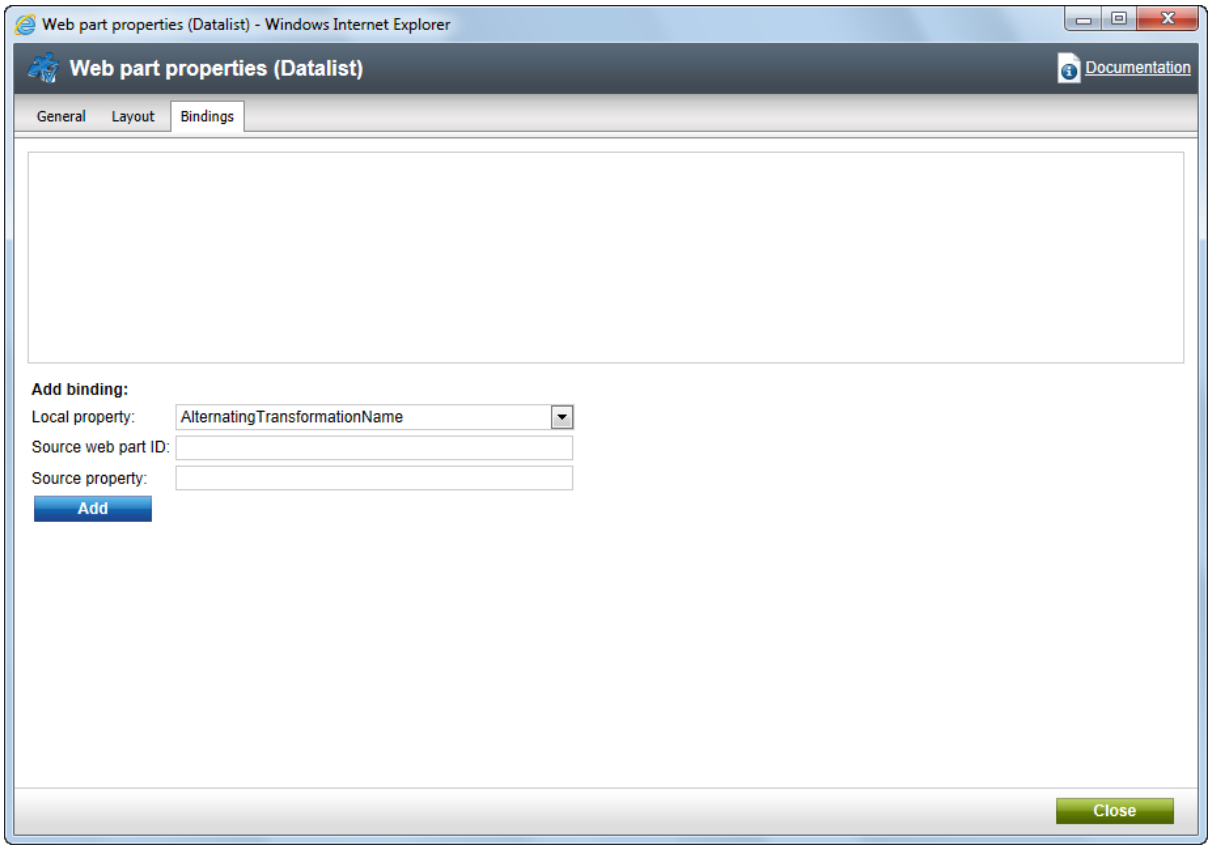
Obsolete feature

This feature is now obsolete. If you need to bind the behavior of a web part to another web part value, please use URL parameters or create a user control that contains both web parts and write custom code that will ensure the communication between the web parts (as described in Setting web part properties dynamically in your code).

If you need to use this feature for backward compatibility, you need to add the following parameter to your web.config file, to the **appSettings** section:

```
<add key="CMSShowWebPartBindingTab" value="true"/>
```

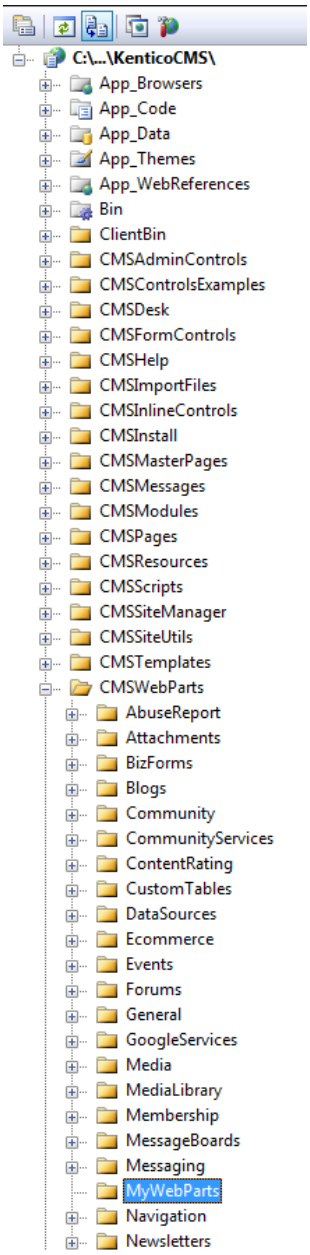
Web part binding allows you to connect two web parts. For example: you can have a web part containing a drop-down list with countries. When some value is selected, it is provided to another web part that displays a list of company offices in the selected country. You can manage web part binding on the **Binding** tab:



Example - creating a product selection dialog with drop-down list and product list

In this example, we will create two web parts - selector and viewer and bind their properties so that the product viewer reflects the product selector status.

1. Open the CMS project in Visual Studio and create a new folder **MyWebParts** under the **CMSWebParts** folder in the Solution Explorer:



2. Create a new user control (ASCX) under **MyWebParts** folder and call it **ProductSelector.ascx**. Switch to its HTML source and paste the following code:

```
<asp:DropDownList ID="DropDownList1" runat="server"
 OnSelectedIndexChanged="DropDownList1_SelectedIndexChanged" AutoPostBack="true">
  <asp:ListItem Value="Under">Under 500</asp:ListItem>
  <asp:ListItem Value="Over">Over 500</asp:ListItem>
</asp:DropDownList>
```

Switch to the code behind and add the following code to the beginning of the code:

[C#]

```
using System;
using CMS.PortalControls;
using CMS.GlobalHelper;
using CMS.DocumentEngine;
using CMS.CMSHelper;
using CMS.ExtendedControls;
```

Change the class inheritance like this:

[C#]

```
public partial class CMSWebParts_MyWebParts_ProductSelector : CMSAbstractWebPart
```

Add the following code inside the page class:

[C#]

```
protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
{
    this.SetValue("PriceRange", this.DropDownList1.SelectedValue.ToString());
    this.ReloadConsumers();
}
```

Please note that you need to set the new value of the web part property **PriceRange** and call the **ReloadConsumers** method.

3. Create a new user control (ASCX) under **MyWebParts** folder and call it **ProductViewer.ascx**. Switch to its HTML source and paste the following code:

```
<%@ Register Assembly="CMS.Controls" Namespace="CMS.Controls" TagPrefix="cc1" %>
<cc1:CMSRepeater ID="CMSRepeater1" runat="server" EnableViewState="true"
TransformationName="CorporateSite.Transformations.ProductList"
SelectedItemTransformationName="CMS.Smartphone.Default" Path="/%"
ClassNames="CMS.Smartphone">
</cc1:CMSRepeater>
```

The **CMSRepeater** control will display the product list.

Switch to the code behind and add the following code to the beginning of the code:

[C#]

```
using System;
using CMS.PortalControls;
using CMS.GlobalHelper;
```

```
using CMS.DocumentEngine;
using CMS.CMSHelper;
using CMS.ExtendedControls;
```

Change the class inheritance like this:

[C#]

```
public partial class CMSWebParts_MyWebParts_ProductViewer : CMSAbstractWebPart
```

Add the following code inside the page class:

[C#]

```
public override void OnContentLoaded()
{
    base.OnContentLoaded();
    SetupControl();
}
/// <summary>
/// Reloads data on request
/// </summary>

public override void ReloadData()
{
    base.ReloadData();
    this.SetupControl();
    this.CMSRepeater1.ReloadData(true);
}

/// <summary>
/// Initializes the control properties
/// </summary>

protected void SetupControl()
{
    if ((this.PagePlaceholder.ViewMode == CMS.PortalEngine.ViewModeEnum.Design) ||
(this.HideOnCurrentPage) || (!this.IsVisible))
    {
        // Stop processing in Design mode and if the control is invisible
        this.CMSRepeater1.StopProcessing = true;
    }

    else
    {
        // set CMSRepeater properties according to the selected value (price
range)
        string priceRange = ValidationHelper.GetString(this.GetValue
("PriceRange"), "Under");
        if (priceRange == "Over")
        {
            this.CMSRepeater1.WhereCondition = "SKUPrice > 500";
        }

        else
        {
            this.CMSRepeater1.WhereCondition = "SKUPrice <= 500";
        }

    }
}
```

4. Save the changes and go to **Site Manager -> Development -> Web parts**. Create a new category called **My Web Parts** and add a new web part:

- **Display name**: Product selector
- **Code name**: ProductSelector
- **File path**: MyWebParts/ProductSelector.ascx

5. Create another new web part:

- **Display name**: Product viewer
- **Code name**: ProductViewer
- **File path**: MyWebParts/ProductViewer.ascx

... and then switch to the **Properties** tab and create a new property:

- **Attribute name**: PriceRange
- **Attribute type**: Text
- **Attribute size**: 100
- **Allow empty value**: yes
- **Display attribute in the editing form**: no

6. Go to CMS Desk, create a new blank page and call it **Product list**.

7. Switch to the **Design** tab and add the **Product selector** web part to the page. Leave all its properties at the default values.

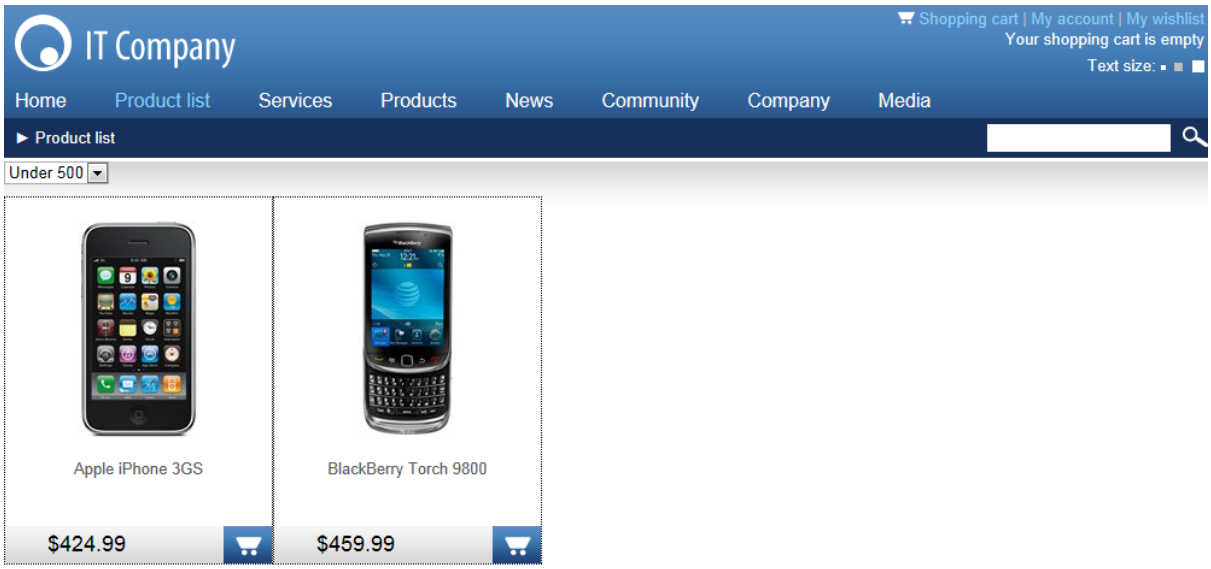
8. Next, add the **Product viewer** web part, set the following properties on the **General** tab:

- **Content before**:
- **Content after**:

and on the **Binding** tab, add a new web part binding using the following values:

- **Local property**: PriceRange
- **Source web part ID**: ProductSelector
- **Source property**: PriceRange

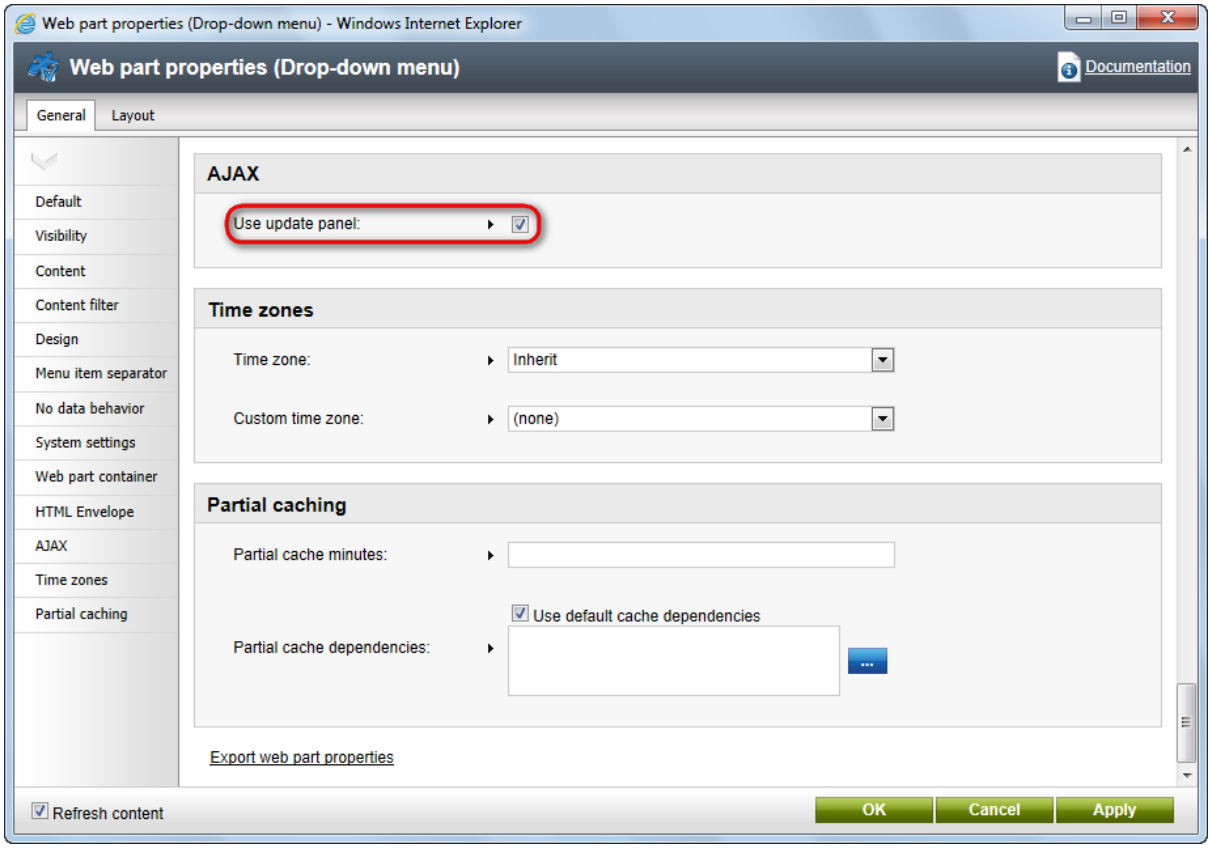
9. Go to the **Live site** mode and see the page. It should look like this:



When you change the drop-down list value, the product list below is automatically updated.

7.27.5 AJAX support

Your web parts can use the UpdatePanel, which wraps the web part into an AJAX UpdatePanel control. This can easily be done by enabling the **Use update panel** property of any web part.



No further modifications are necessary, but the following rules should be followed:

1. If the web part uses any dynamically loaded controls, their ID must be defined:

Incorrect:

```
Control ctrl = this.LoadControl("~/MyControl.ascx");
if (ctrl != null)
{
    Controls.Add(ctrl);
}
```

Correct:

```
Control ctrl = this.LoadControl("~/MyControl.ascx");
if (ctrl != null)
```

```
{
    ctrl.ID = "myControl";
    Controls.Add(ctrl);
}
```

2. When requesting PostBackEventReference, use Kentico's custom function instead of the default one:

Incorrect:

```
this.Page.ClientScript.GetPostBackEventReference(this, "");
```

Correct:

```
CMS.ExtendedControls.ControlsHelper.GetPostBackEventReference(this, "");
```

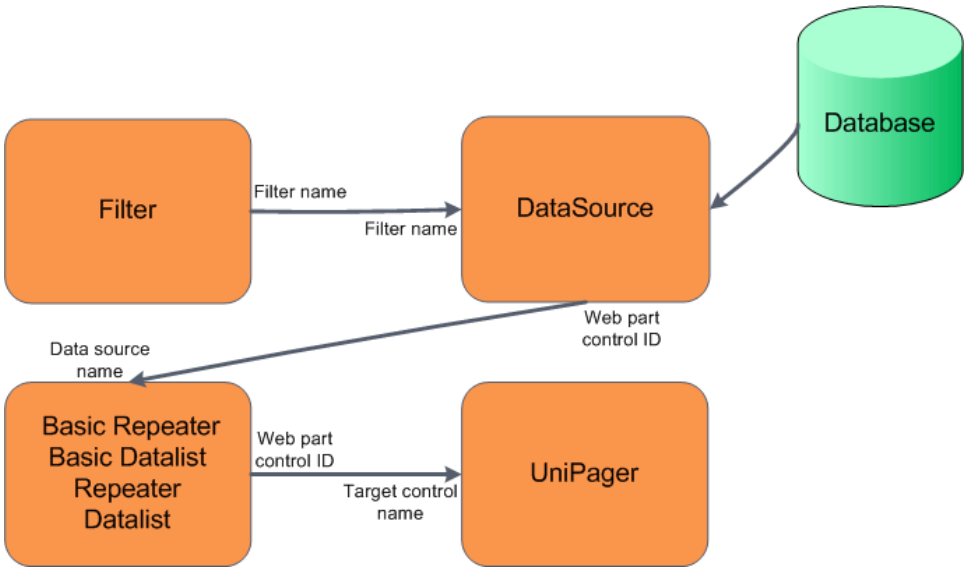
7.27.6 Data source web parts

7.27.6.1 Using Data source web parts

Data source web parts retrieve data from the database (or other sources) and send it to be displayed by other connected web parts. This allows you to distribute the following functions between multiple web parts:

- Loading data
- Displaying and formatting data
- Allowing users to filter the displayed data
- Paging

The diagram below shows how to link together a data source and other types of components. You can place these web parts anywhere on the same page without any change in functionality.



The line captions show which properties connect the web parts. The properties at the beginning and end of each line must have identical values.

Many listing web parts contain a built-in data source. For example, the default Repeater and Datalist provide their own data source for loading documents. If you wish to connect a separate data source to these web parts, the type of the data source web part must match the internal data source (e.g. the *Repeater* web part can only display data from a Documents data source).

To display items from any type of data source, use the Basic repeater or Basic datalist. These web parts do not load data on their own and require a connected data source web part to work correctly.

Example

The following example demonstrates the data source concept in practice. It creates a group of interconnected web parts that load product documents, display them on the website and also provide filtering and paging functionality for the data.

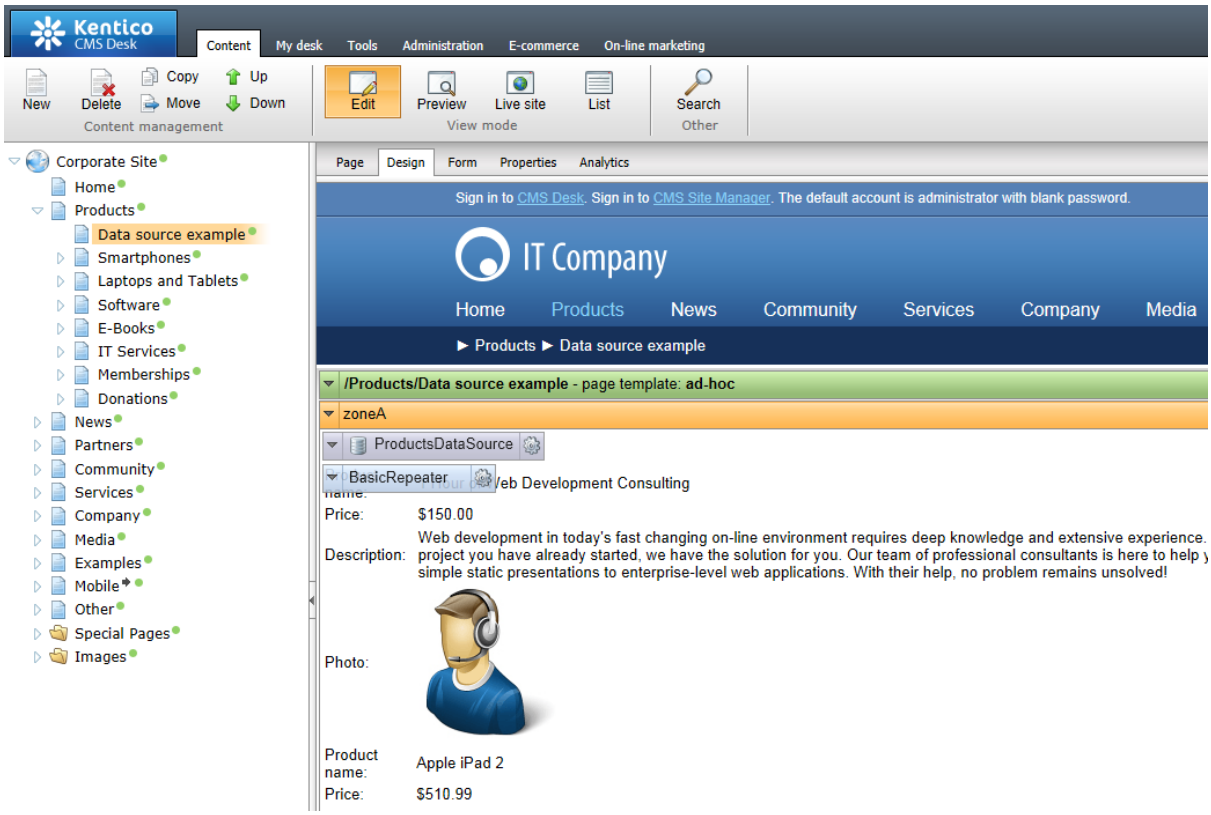
1. Sign in to CMS Desk and edit the sample *Corporate site*.

2. Create a new **Page (menu item)** document under the **Products** section. Choose the **Create a blank page** option in the page template selection dialog.

3. Add the **Products data source** web part to the web part zone on the page. Leave the default values for all of the web part's properties.

4. Add the **Basic repeater** web part to the same web part zone. Set the following properties:

- **Data source name**: ProductsDataSource (this is the default ID of the *Products data source* web part)
- **Transformation name**: CMS.Product.Default (this is a basic transformation for formatting product data)



Basic repeater displaying products loaded from the connected data source web part

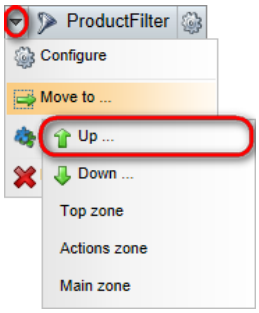
5. Add the **Product filter** web part to the same web part zone. Set the following property of the filter web part:

- **Filter name**: ProductFilter

This filter allows users to limit which products the page displays based on various criteria.

6. If you inserted the product filter web part using the zone's **Add web part** (✚) action, the system automatically adds the filter to the bottom of the zone. In this case, scroll down to the filter web part and move it above the repeater:

a. Right-click the web part.
b. Hover over **Move to** and click **Up**.



7. Configure (⚙) the properties of the *Products data source* web part and type the filter's name

(*ProductFilter*) into the **Filter name** property.
   - This connects the filter to the data source. The data source web part now loads a limited set of items according to the current filter settings.

8. Add the **Universal pager** web part to the zone with the following properties:

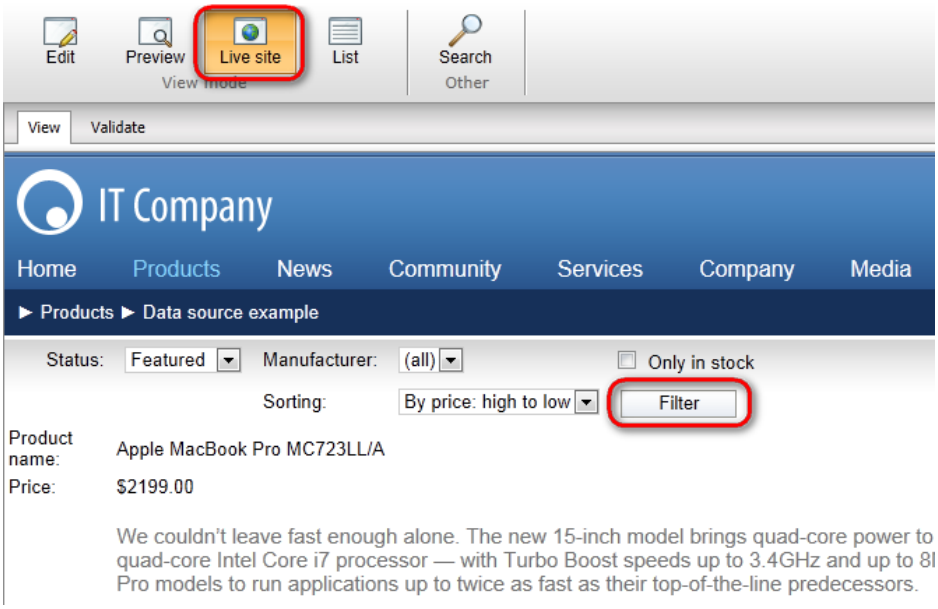
- **Target control name**: BasicRepeater
- **Page size**: 5
- **Group size**: 5
- **Pager layout transformation**: CMS.PagerTransformations.General-PagerLayout

This web part connects to the repeater and separates the displayed data into multiple shorter pages. Paging can be useful when displaying a large number of records.

Result

You can try out the functionality of the page by switching to **Live site** mode.

For example, use the filter to display only products that have the *Featured* **Status** and sort the items by price from high to low.



The pager below the data ensures that the list only includes up to 5 products and allows users to switch between pages.

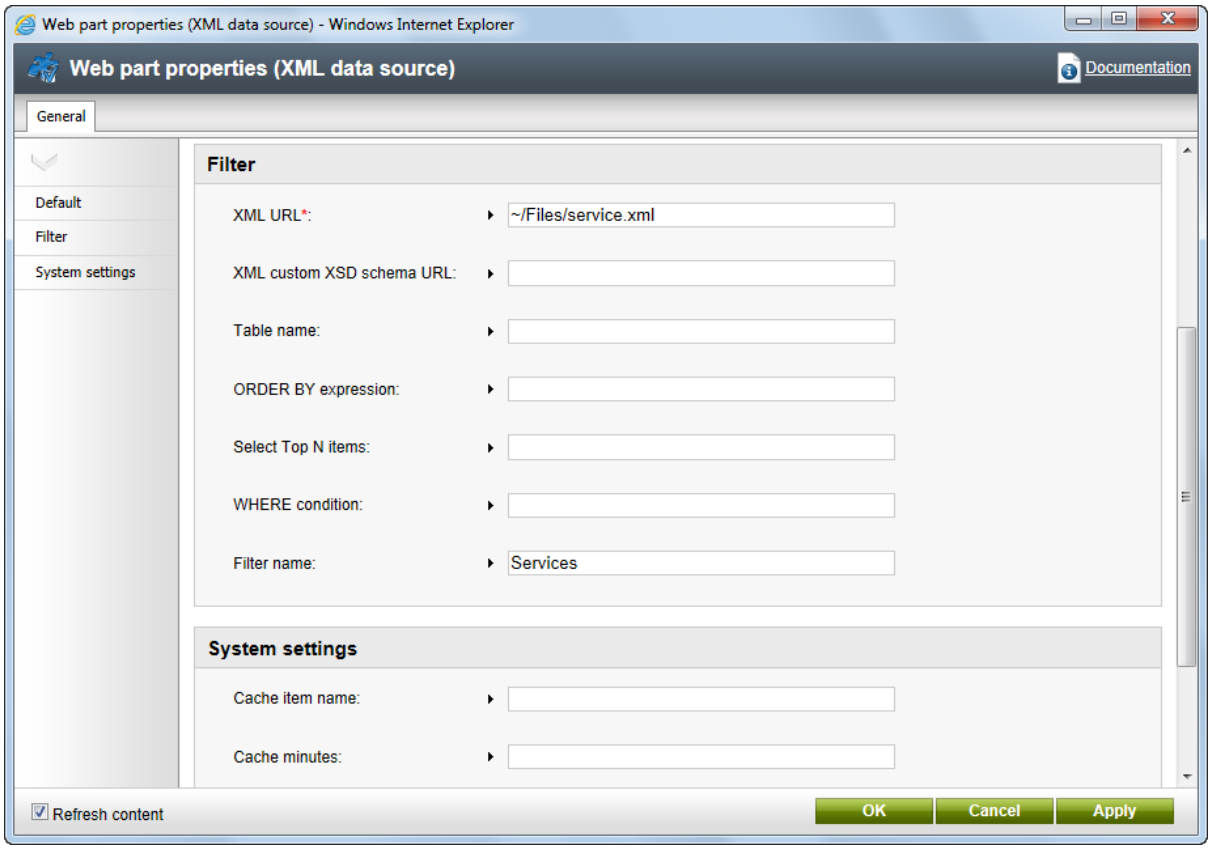
7.27.6.2 Problems with XML data sources

The XML data source web part provides data from an XML file specified by its **XML URL** property. It uses the Dataset.ReadXml() method to read the XML files. This method may separate data from the source XML file into more than one dataset table. In these cases, you need to specify which dataset table the web part should use through the **Table name** property.

Unfortunately, you cannot explicitly determine how the data source organizes XML data into tables. You can find out which table contains the data that you require using the **Debug** function in **Visual Studio**.

If the datasource separates the data into multiple tables and the **Table name** property is not specified appropriately, the web part does not provide any data.

In some cases, the data that you wish to load may be separated into multiple dataset tables. You cannot use the **Table name** property to specify more than one table, so the only way to resolve this issue is to modify the source XML structure (if possible).



7.27.6.3 Developing Data source web parts

When developing custom data source web parts, you need to create two separate user controls:

1. The first control ensures the loading of the data that will be provided by the data source. This control must inherit from **CMSBaseDataSource** (found in the *CMS.Controls* namespace).
2. The second user control contains the first control in its markup and serves as the source file for the actual web part. Must inherit from **CMSAbstractWebPart**.

Example

This example shows how to create a custom data source web part that provides user data from the CMS.

Open your web project in Visual Studio and create two user controls according to the sections below.

Warning

The code is only for demonstrational purposes. Set the values of the *Inherits* attribute in the user control declarations and the class names according to the name and location of your controls.

DataSourceControl.ascx

Leave the ASCX markup of the first user control empty.

DataSourceControl.ascx.cs

The **OnInit** override and **DataFilter_OnFilterChanged** handler method are needed to manage connected filters. If you are not planning to use filters with the data source web part, you do not need to implement these methods.

[C#]

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;

using CMS.Controls;
using CMS.SiteProvider;

/// <summary>
/// User data source. Loads all users according to a specified where condition.
/// </summary>
public partial class CMSGlobalFiles_DataSourceControl : CMSBaseDataSource
{
    #region "Properties"

    /// <summary>
    /// Gets or sets the data source.
    /// </summary>
    public override object DataSource
    {
        get
        {
            return mDataSource ?? (mDataSource = base.GetDataSource());
        }
        set
        {
            mDataSource = (DataSet)value;
        }
    }

    #endregion


    #region "Methods"

    /// <summary>
    /// Assigns a handler for the 'OnFilterChanged' event if there is a filter
attached to the data source.
    /// </summary>
    protected override void OnInit(EventArgs e)
```

```
    {
        if (SourceFilterControl != null)
        {
            SourceFilterControl.OnFilterChanged += new ActionEventHandler
(DataFilter_OnFilterChanged);
        }

        base.OnInit(e);
    }


    /// <summary>
    /// Handles the OnFilterChanged event.
    /// </summary>
    void DataFilter_OnFilterChanged()
    {
        // Clears the old data
        InvalidateLoadedData();

        // Raises the change event.
        this.RaiseOnFilterChanged();
    }


    /// <summary>
    /// Loads the data provided by the datasource.
    /// </summary>
    protected override object GetDataSourceFromDB()
    {
        // Initializes the data properties according to the filter settings.
        if (SourceFilterControl != null)
        {
            SourceFilterControl.InitDataProperties(this);
        }

        // Loads the data. The WhereCondition and OrderBy properties are inherited
from the parent class.
        DataSource = UserInfoProvider.GetFullUsers(WhereCondition, OrderBy);
        return DataSource;
    }

    #endregion
}
```

DataSourceWebPart.ascx

Warning: If your Kentico CMS project was installed as a web application, you need to rename the *CodeFile* property on the first line to *Codebehind*.

```
<%@ Control Language="C#" AutoEventWireup="true"
CodeFile="DataSourceWebPart.ascx.cs"
Inherits="CMSTestingSite_APIExamples_Controls_DataSourceWebPart" %>

<%@ Register src="DataSourceControl.ascx" tagname="DataSourceControl"
tagprefix="cms" %>
```

```
<cms:DataSourceControl ID="srcUsers" runat="server" />
```

DataSourceWebPart.ascx.cs

[C#]

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

using CMS.PortalControls;
using CMS.GlobalHelper;

public partial class CMSGlobalFiles_DataSourceWebPart : CMSAbstractWebPart
{
    /// <summary>
    /// Gets or sets the value of the WhereCondition web part property.
    /// </summary>
    public string WhereCondition
    {
        get
        {
            return ValidationHelper.GetString(this.GetValue("WhereCondition"),
"");
        }
        set
        {
            this.SetValue("WhereCondition", value);
            srcUsers.WhereCondition = value;
        }
    }


    /// <summary>
    /// Gets or sets the value of the OrderBy web part property.
    /// </summary>
    public string OrderBy
    {
        get
        {
            return ValidationHelper.GetString(this.GetValue("OrderBy"), "");
        }
        set
        {
            this.SetValue("OrderBy", value);
            srcUsers.OrderBy = value;
        }
    }


    /// <summary>
    /// Gets or sets the value of the FilterName web part property.
```

```
    /// </summary>
    public string FilterName
    {
        get
        {
            return ValidationHelper.GetString(this.GetValue("FilterName"), "");
        }
        set
        {
            this.SetValue("FilterName", value);
            srcUsers.SourceFilterName = value;
        }
    }


    /// <summary>
    /// Content loaded event handler.
    /// </summary>
    public override void OnContentLoaded()
    {
        base.OnContentLoaded();
        SetupControl();
    }


    /// <summary>
    /// Initializes the properties of the internal data source control.
    /// </summary>
    protected void SetupControl()
    {
        if (this.StopProcessing)
        {
            // Does nothing if the web part is disabled
        }
        else
        {
            this.srcUsers.WhereCondition = this.WhereCondition;
            this.srcUsers.OrderBy = this.OrderBy;

            // Sets the inner data source's name according to the web part's 'Web
part control ID' property.
            // This allows listing web parts to connect to the inner control.
            // The identifier property is named 'FilterName' because data sources
inherit the property from
            // a base class shared with filter objects.
            this.srcUsers.FilterName = (string)this.GetValue("WebPartControlID");

            // Connects the attached filter, as specified by the 'Filter name'
property of the web part
            this.srcUsers.SourceFilterName = this.FilterName;
        }
    }
}
```

If your Kentico CMS project was installed as a web application, you must **Build** the project.

Registering the web part

Register the **DataSourceWebPart.ascx** control as a new web part in Kentico CMS via the **Site Manager -> Development -> Web parts** interface. For details about this process, see the Developing web parts topic.

1. Type *Custom user data source* as the web part's **Display name**.

2. Set the **Type** of the web part to *Data source* on the **General** tab.

3. On the **Properties** tab, define the following properties for the web part:

- **Column name**: OrderBy
- **Attribute type**: Text
- **Attribute size**: 500
- **Allow empty value**: yes (checked)
- **Field caption**: SQL Order by clause
- **Form control**: Order by

- **Column name**: WhereCondition
- **Attribute type**: Text
- **Attribute size**: 500
- **Allow empty value**: yes (checked)
- **Field caption**: SQL Where condition
- **Form control**: Where condition

- **Column name**: FilterName
- **Attribute type**: Text
- **Attribute size**: 200
- **Allow empty value**: yes (checked)
- **Field caption**: Filter name
- **Form control**: Text box

Result

The custom data source web part is now ready and you can try out its functionality.

Go to **CMS Desk**, create a new page and place the following web parts into its design:

Web part	Instructions
Custom user data source	Type *UsersFilter* into the **Filter name** property.
Users filter	Leave the default property values. The **Web part control ID** is set to *UsersFilter* by default, which connects the filter to the custom data source.
Basic repeater	1. Enter the ID of the Custom user data source web part into the **Data source name** property: *CustomUserDataSource* 2. Set an appropriate **Transformation name** for displaying users, for example: *Community.Transformations.MembersList*

The page display a list of all users in the system. You can modify the list by configuring the *SQL Where condition* and *SQL Order by clause* properties of the data source. Visitors can dynamically narrow down the list directly on the page using the connected filter.

7.27.6.4 Developing custom filters

Filters are components that allow users to modify the data loaded and displayed by other website components. Filters provide two types of functionality:

- Limiting the range of items displayed in a list
- Changing the order of items in a list

Kentico CMS comes with a built-in set of filters for various types of data, but you can also create custom filters to fulfill specific requirements.

1. Implement filters as *user controls* that inherit from one of the following base classes:

- **CMSAbstractDataFilterControl** - works with document data sources
- **CMSAbstractMenuFilterControl** - works with navigation web parts
- **CMSAbstractQueryFilterControl** - works with custom table and query data sources

Select the appropriate base class according to the type of data that you need to filter. You can find these base classes in the **CMS.Controls** namespace.

Note

If you are connecting a **CMSAbstractQueryFilterControl** filter to a data source with a custom query, the query must contain the ##WHERE## and ##ORDERBY## expressions. When applied, the filter replaces the expressions with dynamically generated SQL code.

Query example:

```
SELECT ##TOPN## ##COLUMNS## FROM View_CONTENT_MenuItem_Joined
WHERE (##WHERE##) ORDER BY ##ORDERBY##
```

2. Add the filter control onto your website through the **Filter** web part.
   - You need to specify the path to the *.ascx* file in the **Filter control path** property.

3. Attach the filter web part to the data source web part of the matching type.

You can also use custom filters anywhere in your code as standard user controls, for example on ASPX templates or in other web parts. All necessary properties, such as the **FilterName** used to connect with the data source, are inherited from the base class.

Example

The example below demonstrates how to develop a custom filter. This sample filter works with *product documents*. It supports filtering according to product departments and allows the selection of several options that determine the order in which products are displayed. You can create filters for all types of

documents or other objects using the same approach.

1. Open the web project in Visual Studio and create a **New folder** under the root called *CMSGlobalFiles* (if it doesn't already exist).
   - The content of this folder can be exported along with your site when it is deployed to the live server.

2. Add a new **Web User Control** named *CustomProductFilter.ascx* into the **CMSGlobalFiles** folder and modify it to contain the following code:

```
<%@ Control Language="C#" AutoEventWireup="true"
CodeFile="CustomProductFilter.ascx.cs"
Inherits="CMSGlobalFiles_CustomProductFilter" %>

<table cellpadding="2">
  <tr>
    <td>
      <cms:LocalizedLabel ID="lblDepartment" runat="server" Text="Product
department" DisplayColon="true">
      </cms:LocalizedLabel>
    </td>
    <td>
      <cms:LocalizedDropDownList ID="drpDepartment" runat="server" Width="180" >
      </cms:LocalizedDropDownList>
    </td>
  </tr>
  <tr>
    <td>
      <cms:LocalizedLabel ID="lblOrder" runat="server" Text="Order by"
DisplayColon="true">
      </cms:LocalizedLabel>
    </td>
    <td>
      <cms:LocalizedDropDownList ID="drpOrder" runat="server" Width="180" >
      </cms:LocalizedDropDownList>
    </td>
  </tr>
  <tr>
    <td colspan="2">
      <cms:LocalizedButton ID="btnFilter" runat="server" Text="Apply Filter" />
    </td>
  </tr>
</table>
```

This creates the design of the filter's user interface. As you can see, it is composed of localized labels, drop-down lists and a button arranged in a simple table layout. Alternatively, it is possible to use a CSS-based layout applied through HTML elements (e.g. , , etc.).

When developing a filter for actual live deployment, it may be preferable to enter the captions of the child controls using localization strings through the **ResourceString** property, rather than directly as **Text**.

3. Switch to the code behind file of the user control, add the references shown below and set the control to inherit from the appropriate base class. This example uses the **CMSAbstractDataFilterControl** class, since the filter is intended for use with a document data source.

[C#]

```
using System;
using System.Data;
using System.Collections.Generic;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

using CMS.Controls;
using CMS.GlobalHelper;
using CMS.Ecommerce;

public partial class CMSGlobalFiles_CustomProductFilter :
CMSAbstractDataFilterControl
{

    ...

}
```

4. Add the following methods into the user control class:

[C#]

```
/// <summary>
/// Setup the inner child controls
/// </summary>
private void SetupControl()
{
    // Hide the filter if StopProcessing is enabled
    if (this.StopProcessing)
    {
        this.Visible = false;
    }

    // Initialize only if the current request is NOT a postback
    else if (!RequestHelper.IsPostBack())
    {
        // Loads product departments as filtering options
        InitializeDepartments();

        // Initialize the order by drop-down list
        IntializeOrder();
    }
}


/// <summary>
/// Ensures that the ordering options are loaded into the order by drop-down list
/// </summary>
private void IntializeOrder()
{
    // Initialize options
    this.drpOrder.Items.Add(new ListItem("Price - Ascending", "priceAsc"));
```

```
    this.drpOrder.Items.Add(new ListItem("Price - Descending", "priceDesc"));
    this.drpOrder.Items.Add(new ListItem("Product name", "name"));
}


/// <summary>
/// Loads all existing product departments as filtering options into the
department drop-down list
/// </summary>
private void InitializeDepartments()
{
    // Get all product departments
    DataSet departments = DepartmentInfoProvider.GetDepartments("",
"DepartmentDisplayName", 0, "DepartmentID, DepartmentDisplayName");

    // Check if there are any product departments
    if (!DataHelper.DataSourceIsEmpty(departments))
    {
        // Binds departments to the drop-down list
        this.drpDepartment.DataSource = departments;
        this.drpDepartment.DataTextField = "DepartmentDisplayName";
        this.drpDepartment.DataValueField = "DepartmentID";

        this.drpDepartment.DataBind();

        // Adds the default '(all)' value
        this.drpDepartment.Items.Insert(0, new ListItem("(all)", "##ALL##"));
    }
}
```

These methods ensure that the correct filtering options are loaded into the child drop-down lists.

5. Define the **SetFilter()** method as shown below:

[C#]

```
/// <summary>
/// Generates a WHERE condition and ORDER BY clause based on the current filtering
selection
/// </summary>
private void SetFilter()
{
    string where = null;
    string order = null;

    // Generate a WHERE condition based on the selected product department
    if (this.drpDepartment.SelectedValue != null)
    {
        int departmentId = ValidationHelper.GetInteger
(this.drpDepartment.SelectedValue, 0);

        if (departmentId > 0)
        {
            where = "SKUDepartmentID = " + departmentId;
        }
```

```
    }

    // Apply the selected ordering direction
    if (this.drpOrder.SelectedValue != "")
    {
        switch (this.drpOrder.SelectedValue)
        {
            case "priceAsc":
                order = "SKUPrice";
                break;

            case "priceDesc":
                order = "SKUPrice Desc";
                break;

            case "name":
                order = "SKUName";
                break;
        }
    }

    if (where != null)
    {
        // Set where condition
        this.WhereCondition = where;
    }

    if (order != null)
    {
        // Set orderBy clause
        this.OrderBy = order;
    }

    // Raise the filter changed event
    this.RaiseOnFilterChanged();
}
```

The custom filter control dynamically generates a WHERE condition and ORDER BY statement based on the current filter selection, which is then used to set the value of the **WhereCondition** and **OrderBy** members inherited from the base class. These members are read by the data source to which the filter is attached and inserted into the SQL query used to load data. Remember that the filter must inherit from the appropriate base class according to the type of the used data source.

6. Add two more methods that override the handlers of the **Init** and **PreRender** events:

[C#]

```
/// <summary>
/// Init event handler
/// </summary>
protected override void OnInit(EventArgs e)
{
    // Create child controls
    SetupControl();
```

```
    base.OnInit(e);
}


/// <summary>
/// PreRender event handler
/// </summary>
protected override void OnPreRender(EventArgs e)
{
    // Check if the current request is a postback
    if (RequestHelper.IsPostBack())
    {
        // Apply the filter to the displayed data
        SetFilter();
    }

    base.OnPreRender(e);
}
```

The handlers ensure that the appropriate private methods (defined previously in the example) are called during the correct stages of the page life cycle.

When the *Apply Filter* button is clicked, a postback occurs, which triggers the **SetFilter()** method during the **PreRender** event and the filter is applied to the displayed data. Notice the use of the **CMS.GlobalHelper.RequestHelper.IsPostBack()** method in the conditions. The **SetFilter()** method is only called when the current page request is a postback and the child controls are initialized only when this is not the case.

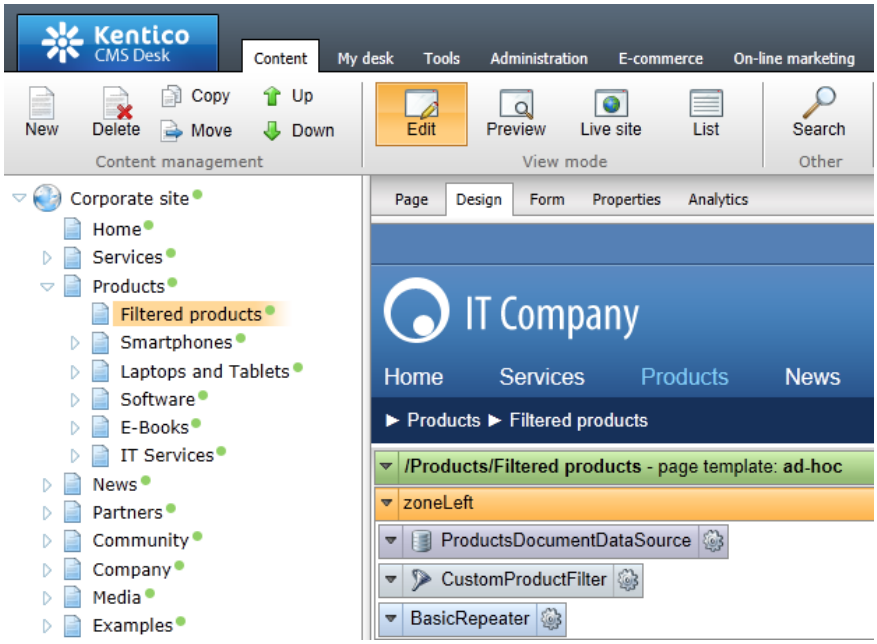
Result

The filter control is now finished and you can try out its functionality. It is recommended to test the filter on the **Corporate** or **E-commerce** sample sites, since they already contain examples of product documents.

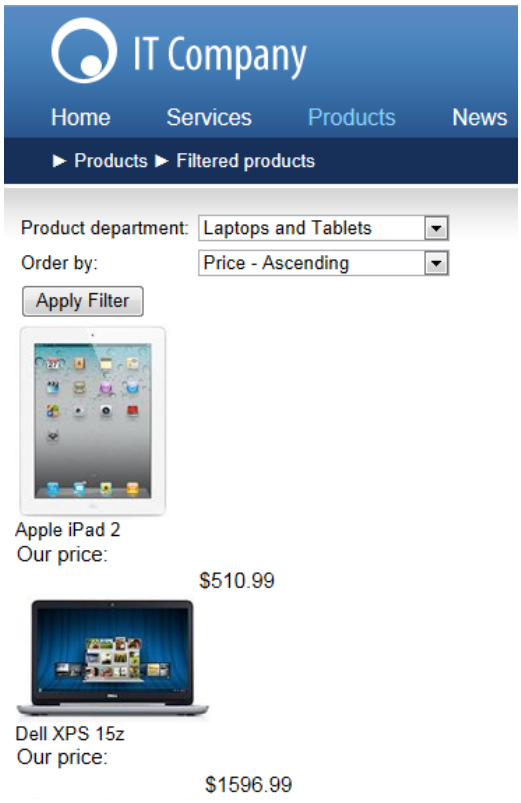
Go to **CMS Desk**, create a new page and place the following web parts into its design:

Web part	Instructions
Documents data source	1. Configure the web part to load product documents. 2. Fill in the **Filter name** property (e.g. *CustomProductFilter)*. **Note**: You can alternatively use the *Products data source* web part for this purpose.
Filter	1. Assign a **Filter name** that matches the data source (*CustomProductFilter*). 2. Enter the path of the .ascx file implementing your custom filter into the **Filter control path** property (*~/CMSGlobalFiles/CustomProductFilter.ascx*).
Basic repeater	1. Enter the ID of the Documents data source web part into the **Data source name** property. 2. Set an appropriate **Transformation name**, for example: *Ecommerce.Transformations.Product_SimplePreview*

You can find more information about how this type of web part combination works in Using Data source web parts.



If you view the page in **Live site** mode, it displays a list of products with the custom filter shown above. You can filter the products by their department and change the order in which they are displayed.



7.27.7 Layout web parts

7.27.7.1 Overview

Layout web parts are special types of web parts that allow users to define and modify the structure of portal engine page templates without the need to write or edit the code of the template's page layout. Each layout web part contains one or more child zones that are arranged in a specific way. The layout can be configured just like a standard web part using properties or even directly on the **Design** tab of CMS Desk.

Using this approach, it is possible to simply create a new page with one web part zone (e.g. using the **Create a blank page** option when adding a page to the content tree in **CMS Desk**) and then add the appropriate layout web part to separate the page into further zones as required.

In addition to defining flexible web part zones, some layout web parts also provide special functionality, such as collapsible page sections, zones with a fixed position or graphical interfaces that allow users to select which zone's content should be displayed. The following layout web parts (found under the **Layout** category) are delivered with Kentico CMS by default:

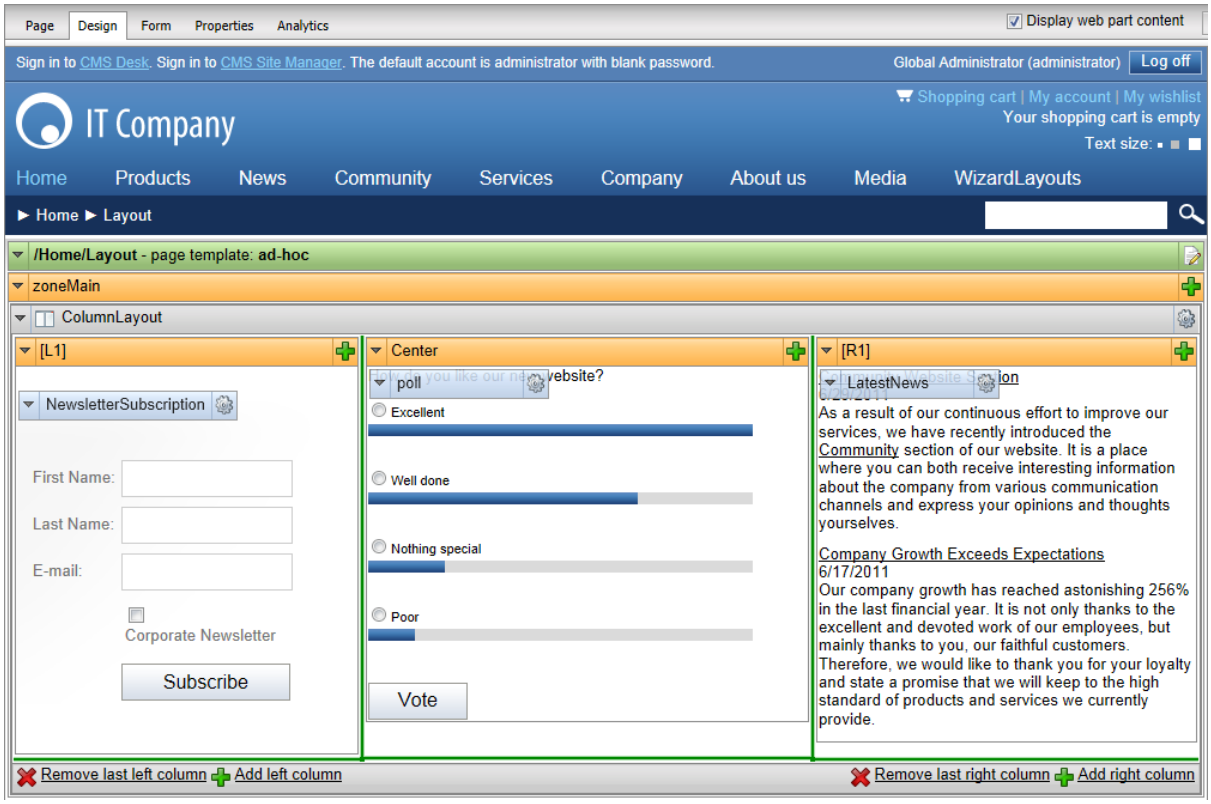
- **Accordion layout** - provides a layout divided into multiple panes that can be expanded or collapsed. Each pane defines a separate web part zone and users can click on the pane headers to select which one should be displayed.
- **Collapsible panel** - generates a web part zone inside a panel that can be collapsed or expanded by users.
- **Columns layout** - provides a CSS-based column layout for content. Each column defines a separate web part zone.
- **Rows layout** - provides a row layout for content. Each row contains a separate web part zone.
- **Table layout** - provides a table layout for content. Each cell in the table defines a separate web part zone.
- **Tabs layout** - creates an AJAX tab layout. The content of each tab can be added through a separate web part zone. Users can then click on the tabs to select which content should be displayed.
- **Web part zone** - adds a single web part zone to the page that can be used to group multiple web parts or assign additional formatting to them. The zone may also be configured to display its content at a fixed location on the screen that moves along with page scrolling.
- **Zones with effect** - provides a list of elements that define separate web part zones and allows additional JavaScript effects (e.g. using jQuery) to be applied to the content displayed by the zones. The scripts used to generate the effects must be specified through the web part's properties.
- **Wizard layout** - provides a layout divided into multiple steps. The wizard displays one step at a time and allows users to navigate through them in consecutive order. The content of each step is defined in its own web part zone. The layout may generate two additional zones as a header and footer for the wizard. It is also possible to separately add the header content of the wizard and the buttons used to move between steps through the *Wizard header* and *Wizard buttons* web parts.

For additional details about individual web parts, please refer to the Kentico CMS Web Parts reference.

If you wish to use a layout that cannot be achieved by the built-in web parts, it is possible to create new layout web parts or customize existing ones to fulfill the given requirements. Please refer to the Developing layout web parts topic to find an example of how a custom layout web part can be created and additional information about how web parts of this type work internally.

Editing the content of layout web parts

To add or edit the content of the zones generated by a layout web part, view the given page on the **Design** tab of CMS Desk. You can manage the zones inside the layout like any other standard web part zones and add the required web parts.



All default layout web parts have the **Allow design mode** property. If it is enabled, additional zone management actions will be available directly on the **Design** tab. This makes it possible to dynamically resize individual layout zones as necessary by dragging the green lines on the borders, as can be seen in the screenshot above. Layouts composed of multiple zones also allow designers to directly add or remove the sections that form the layout (columns, tabs etc.).

Once the layout web part and all of its child web parts are added and configured properly, the entered content will be displayed normally on the live site using the specified page structure.

If a layout web part is deleted while it still has some content in its child zones, the given web parts will remain saved in the template definition. You can use this fact to recover a layout if it is deleted accidentally. Simply add the layout web part again and its child content will automatically be inserted. For this reason, it is recommended to remove all child web parts contained in layout before deleting it, if you wish to permanently remove it from the page.

Adding layouts as widgets

It is also possible to add layouts to a page as Widgets. This can be used to give

editors or other users without design permissions additional flexibility when it comes to arranging page content. The zones generated by a layout widget will automatically match the type of the parent zone.

These layout widgets may also be placed onto Dashboards to help users organize their content. When adding content into a layout on a dashboard page, it is necessary to drag and drop new widgets from their default location to the appropriate zone generated by the layout, since it is not possible to add widgets into specific dashboard zones directly.

7.27.7.2 Developing layout web parts

Even though Kentico CMS is delivered with multiple layout web parts that may be used out-of-the-box, it is also possible to customize them or develop completely new web parts to define layouts that exactly match your specific requirements.

The development process is similar as with standard web parts, but there are several important differences. All layout web parts must inherit from the **CMSAbstractLayoutWebPart** base class (rather than the standard *CMSAbstractWebPart*), which can be found in the **CMS.PortalControls** namespace. The most important members inherited from this base class that can be used to implement the layout are described in the example below.

Example

The following example demonstrates how a custom layout web part can be developed. For the sake of simplicity, this sample layout will only provide a single web part zone with a configurable width and height. It is only intended for demonstration purposes and not for practical use. For further inspiration, you can view the code of the more advanced built-in layout web parts found in the **~/CMSWebParts/Layouts** folder of your web project.

1. Open the web project in Visual Studio, right click the **CMSWebparts/Layouts** folder and select **Add New Item**. Now create a **Web User Control** named *CustomZone.ascx*.

2. Next, edit the control's code behind file. Add references, set the control to inherit from the appropriate base class and define two properties according to the following code:

[C#]

```
using CMS.PortalControls;
using CMS.GlobalHelper;
using CMS.PortalEngine;

public partial class CMSWebParts_Layouts_CustomZone : CMSAbstractLayoutWebPart
{
    /// <summary>
    /// Property used to access the value of the Width property of the web part.
    /// </summary>
    public string Width
    {
        get
        {
```

```
            return ValidationHelper.GetString(this.GetValue("Width"), "");
        }
        set
        {
            this.SetValue("Width", value);
        }
    }

    /// <summary>
    /// Property used to access the value of the Height property of the web part.
    /// </summary>
    public string Height
    {
        get
        {
            return ValidationHelper.GetString(this.GetValue("Height"), "");
        }

        set
        {
            this.SetValue("Height", value);
        }
    }

}
```

The **Width** and **Height** public properties are used to handle the corresponding properties of the web part object that will be defined later in the example. The properties use the string type, since the width/height is assigned as a CSS style text value.

3. Add the following private method to the class:

[C#]

```
/// <summary>
/// Adds the layout's web part zone and envelope.
/// </summary>
private void insertZone()
{
    // Stores the Width and Height properties of the layout as a CSS style value.
    string style = null;

    // Width of the zone.
    if (!String.IsNullOrEmpty(Width))
    {
        style += " width: " + Width + ";";
    }

    // Height of the zone.
    if (!String.IsNullOrEmpty(Height))
    {
        style += " height: " + Height + ";";
    }

    // Adds a DIV element that will encapsulate the layout's web part zone.
    Append("<div");
```

```
    // Defines the DIV element's id attribute used to identify the zone's envelope
in Design mode.
    if (IsDesign)
    {
        Append(" id=\"", ShortClientID, "_zoneEnvelope\"");
    }

    // Defines the style attribute of the zone's envelope.
    if (!String.IsNullOrEmpty(style))
    {
        Append(" style=\"", style, "\"");
    }

    Append(">");

    // Adds the web part zone.
    CMSWebPartZone zone = AddZone(this.ID + "_zone", this.ID);

    Append("</div>");
}
```

The following members inherited from the base class are used:

- **Append(params string[] text)** - this method is a fundamental tool used to build the output code that defines the web part layout. When the method is called, the content of its parameters is added to the end of the current layout code. Any number of string parameters may be specified.
- **AddZone(string id, string title)** - creates a new web part zone, adds it to the layout and passes it back as the return value. If the layout is added as a widget, this method automatically ensures that the zones in the layout are created as widget zones of the same type as the parent zone.
- **IsDesign** - true if the web part is currently being viewed on the *Design* tab of CMS Desk.

The *InsertZone()* method is used to define the web part zone that will be displayed in the layout and surrounding code that allows its width and height to be configured. It will be called later by the code added in the next step of the example.

4. Every layout web part must override the **PrepareLayout()** virtual method, which is where the output code of the layout is generated. Use the following code to add the method:

[C#]

```
/// <summary>
/// Builds the output code that will generate the desired layout on the page.
/// </summary>
protected override void PrepareLayout()
{
    // Starts building the layout code.
    StartLayout();

    // Wraps the layout into a table with additional content if the web part is
edited in Design mode.
    if (IsDesign)
    {
        Append("<table class=\"LayoutTable\" cellspacing=\"0\">");
```

```
        if (this.ViewMode == ViewModeEnum.Design)
        {
            Append("<tr><td class=\"LayoutHeader\" colspan=\"2\">");

            // Adds a header container.
            AddHeaderContainer();

            Append("</td></tr>");
        }

        Append("<tr><td>");
    }

    // Calls the private method that adds the web part zone.
    insertZone();

    // Closes the Design mode table.
    if (IsDesign)
    {
        Append("</td>");

        // Generates dynamic resizers for the zone if the Allow design mode
property of the web part is enabled.
        if (AllowDesignMode)
        {
            // Adds a horizontal resizer.
            Append("<td class=\"HorizontalResizer\" onmousedown=\"",
GetHorizontalResizerScript("zoneEnvelope", "Width"), " return false;\"> </
td></tr><tr>");

            // Adds a vertical resizer.
            Append("<td class=\"VerticalResizer\" onmousedown=\"",
GetVerticalResizerScript("zoneEnvelope", "Height"), " return false;\"> </
td>");

            // Adds a combined horizontal and vertical resizer to the corner where
both resizer types intersect.
            Append("<td class=\"BothResizer\" onmousedown=\"",
GetHorizontalResizerScript("zoneEnvelope", "Width"), " ", GetVerticalResizerScript
("zoneEnvelope", "Height"), " return false;\"> </td>");
        }

        Append("</tr></table>");
    }

    // Saves the current status of the layout code and ensures that it is rendered
on the page.
    FinishLayout();
}
```

The following members inherited from the base class are used:

- **StartLayout()** - initializes the building of the layout code. You may start using the *Append()* method only after this method is called.
- **GetHorizontalResizerScript(string elementId, string widthPropertyName)** - returns a script that adds the functionality of a dynamic width resizer. The first parameter is used to specify the ID of the element that will be resized and the second sets the property that will be modified when the width is

changed via the resizer.
- **GetVerticalResizerScript(string elementId, string heightPropertyName)** - returns a script that adds the functionality of a dynamic height resizer. The parameters work the same way as for the horizontal resizer.
- **FinishLayout()** - this method finalizes the layout code of the web part. It must always be used as the last modification of the layout (i.e. once it is called, the *Append()* method will no longer work correctly).

As you can see, the method added in this step initializes the layout code, calls the method created in the previous step and then finalizes the output. In addition, it adds a table element around the web part zone, which is displayed only when the layout web part is viewed in design mode. If the **Allow design mode** property of the web part is enabled, this table also includes elements that can be dragged with the mouse to directly resize the layout's zone. It is not necessary to define the **AllowDesignMode** property for the control, since it is automatically inherited from the **CMSAbstractLayoutWebPart** base class.

Save the changes. If your Kentico CMS project was installed as a web application, you must also **Build** the project.

5. Next, register the web part in the system. Go to **Site Manager -> Development -> Web parts**, select the **Layouts** category and click **New web part**. Choose to **Create a new web part** and enter the following values:

- **Display name**: Custom zone layout
- **Code name**: CustomZone
- **File path**: ~/CMSWebParts/Layouts/CustomZone.ascx
- **Generate the code files**: false

Click **Save** and you will be redirected to the web part's **General** tab. Here, select the *Layout* option from the **Type** drop-down list and click **Save** again. All layout web parts must use this type in order to function correctly.

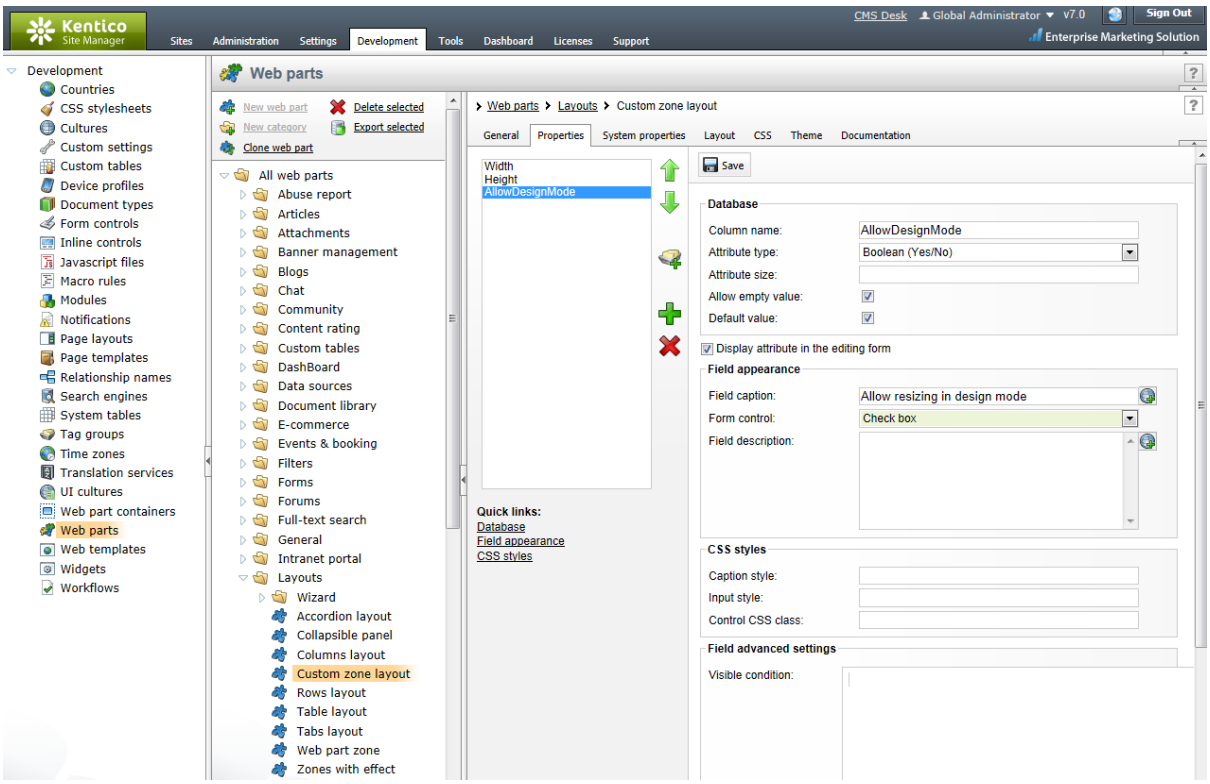
6. Switch to the **Properties** tab and add (+) three properties according to the information below. Click **Save** for each property before moving on to the next one.

- **Column name**: Width
- **Attribute type**: Text
- **Attribute size**: 100
- **Allow empty value**: yes (checked)
- **Field caption**: Zone width
- **Form control**: Text box

- **Column name**: Height
- **Attribute type**: Text
- **Attribute size**: 100
- **Allow empty value**: yes (checked)
- **Field caption**: Zone height
- **Form control**: Text box

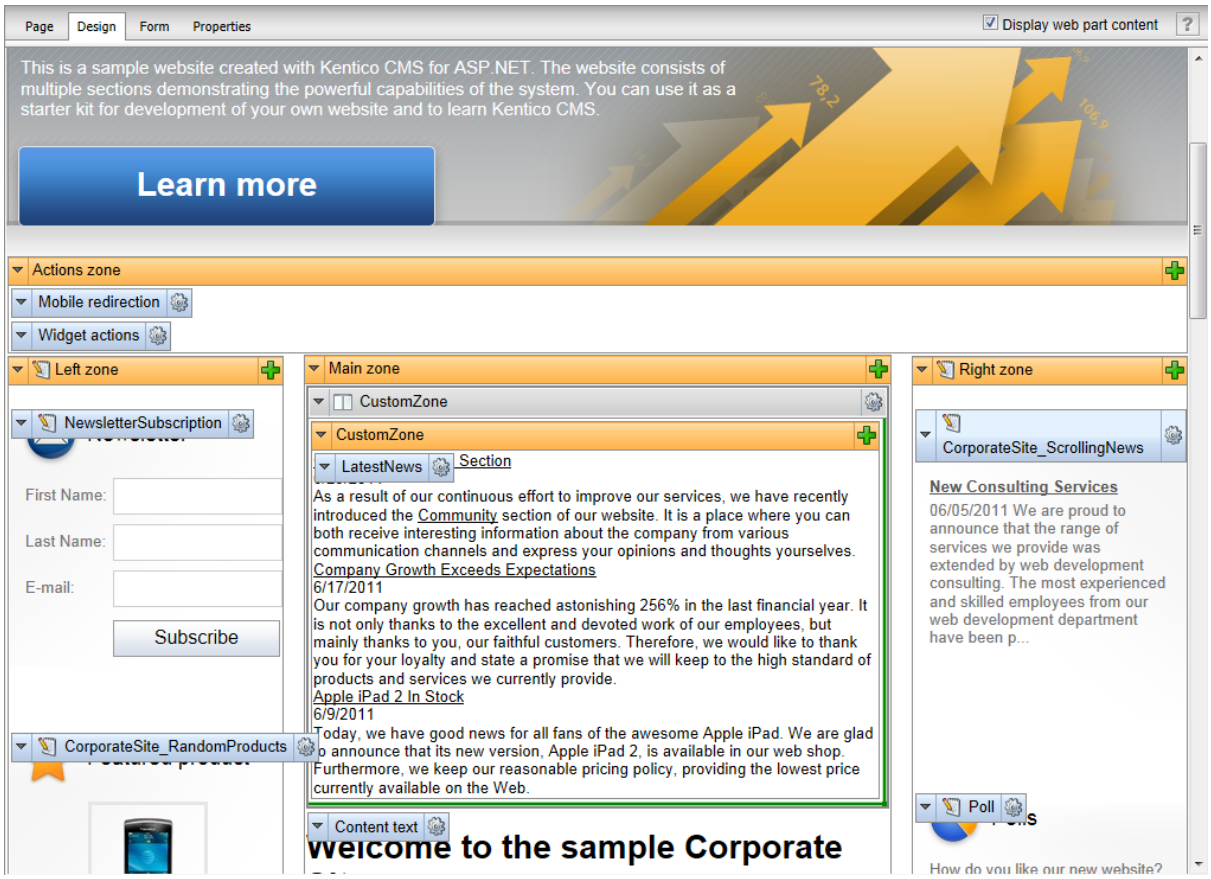
- **Column name**: AllowDesignMode
- **Attribute type**: Boolean (Yes/No)
- **Allow empty value**: yes (checked)
- **Default value**: yes (checked)

- **Field caption**: Allow resizing in design mode
- **Form control**: Check box



These properties will be available in the web part's configuration dialog. They are handled by the corresponding properties in the code of the user control implementing the web part.

7. The custom layout web part is now complete and ready to be used. You can try out its functionality by adding it onto one of your pages in CMS Desk.



You can use a similar approach (with more advanced layout code) to implement a custom web part to generate any required layout.

7.27.8 Web part containers

7.27.8.1 Containers overview

Containers are used as "boxes" for web parts. They consist of content that is displayed before and after the web part which means they are used as an envelope for web part content. They have three advantages over the **Content before** and **Content after** web part properties:

- They are re-usable for any number of web part instances.
- They can contain a title and dynamically inserted values of the enclosed web part's properties.
- They are objects in the system, which means they can be Exported and Imported.

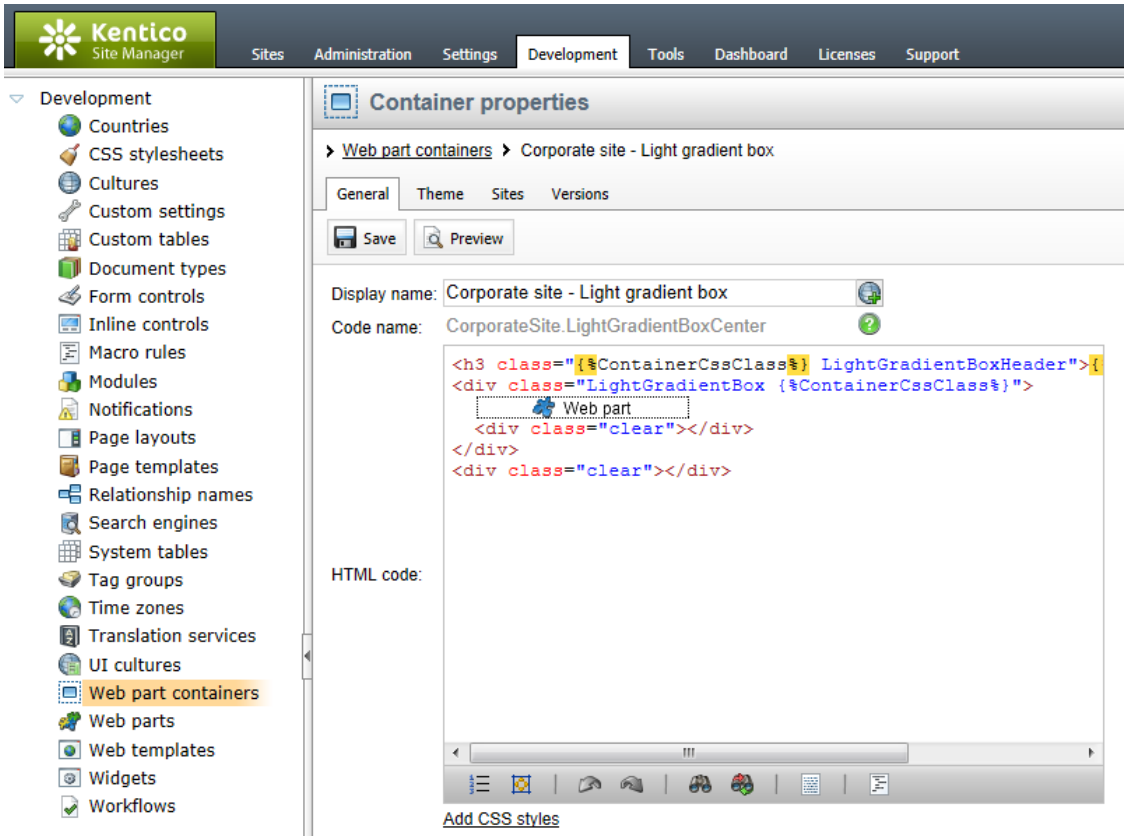
Containers may be assigned not only to web parts, but also widgets and entire web part zones.

The containers can be managed in **Site Manager -> Development -> Web part containers**. The following properties are available when editing ( ) or creating a web part container:

- **Display name** - the name of the container displayed to users in lists and selectors.
- **Code name** - a string identifier for the container that can be used by developers in the API and URLs. Unless there is a reason to set a particular value, you can leave the default *(automatic)* option, and the system will generate an appropriate code name.
- **HTML code** - this editor allows you to enter the HTML content that will be placed around the web

part. The position of the actual web part content is represented in the code by a placeholder, so you can add both opening and closing elements.

- **CSS styles** - this field becomes available if you click the Add CSS styles link. Here you can define any CSS classes used within the code of the container. The specified styles will be loaded on any page where the given container is used.



In the case of existing web part containers that are already placed somewhere on the website, you can use the **Preview** mode for convenient editing. This allows you to inspect the live site appearance of a container directly while working with its code. Please see the Design preview topic for additional details about the preview.

The **HTML code** field can contain dynamically loaded values of the enclosed web part's properties. You can insert them using expressions in the following format:

{%PropertyName%}

You will most often use the following expression to insert the container title:

{%ContainerTitle%}

These macro expressions are resolved even if macro resolving is disabled for the particular web part instance (through the **Disable macros** property).

Defining CSS styles

There are two locations where CSS classes used by the web part container can be defined:

- **In the general site stylesheet** - all CSS classes are stored in one file, but exporting the container to a site that uses a different stylesheet is more difficult.
- **In the CSS styles property of the container** - styles are stored separately for every container, which requires an additional resource request, but containers are automatically exported (including staging) with their CSS classes to other sites or Kentico CMS instances.

A stylesheet request link similar to the following will be added to the <head> element of any page that contains web part containers with styles defined in their **CSS styles** property:

<link href="/KenticoCMS/CMSPages/GetCSS.aspx?_containers=1;14" type="text/css" rel="stylesheet"/>

The value of the **_containers** URL parameter is dependent on the containers used on the page, the example above is for a page that contains the *Black box* and *Orange box* containers (identified by their *ContainerID* values).

If CSS minification is enabled, the request will use the following format instead:

<link href="/KenticoCMS/CMSPages/GetResource.ashx?_containers=1;14" type="text/css" rel="stylesheet"/>

Storing files related to web part containers

If your web part container code requires any additional files, such as images used by the classes defined in the **CSS styles** property, they must be stored in the *~/App_Themes/Components/Containers/<container code name>* folder so they can be exported/imported along with the container.

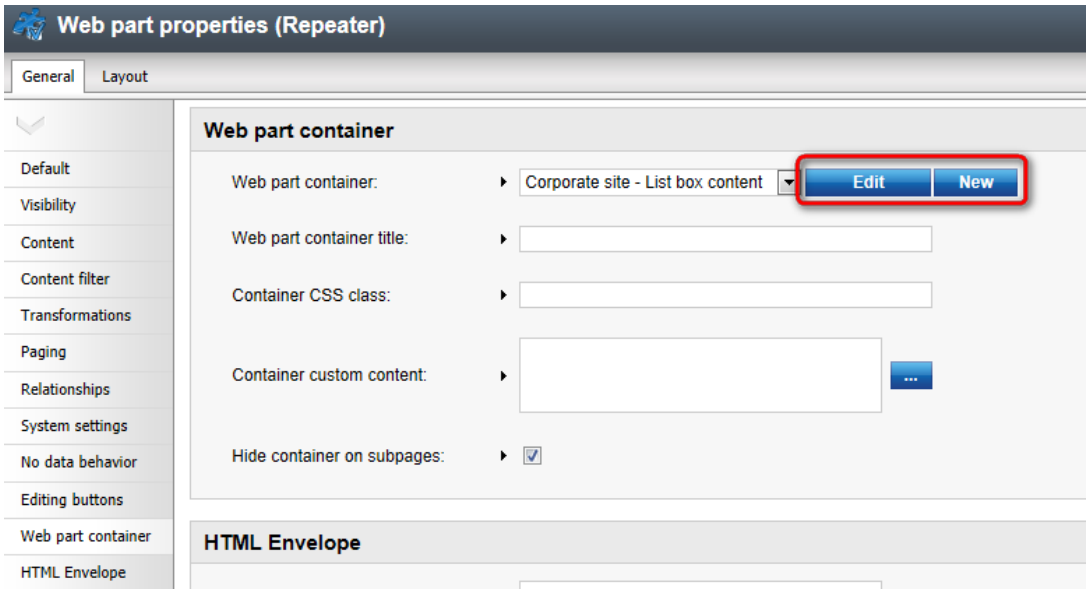
The content of this folder can be managed directly from the administration interface at **Site Manager -> Development -> Web part containers** by editing () the given container and switching the **Theme** tab.

If the used CSS classes are defined in the site stylesheet, any accompanying files should be stored in the *~/App_Themes/<stylesheet code name>* folder.

For more information about page component CSS styles, please see the Development -> CSS stylesheets and design -> CSS for page components topic.

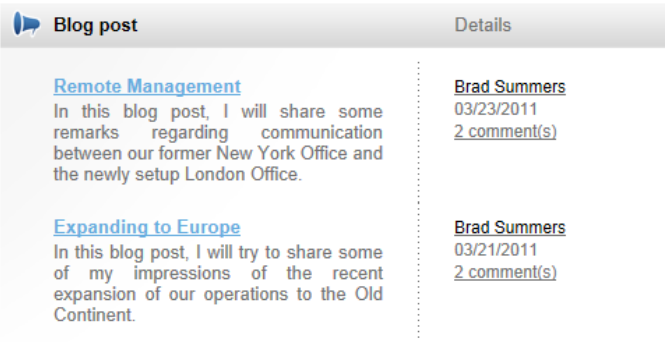
Using web part containers

Containers are assigned to web parts, widgets or zones simply by configuring () them and selecting a container through the **Web part container** property. The web part container editing form mentioned above can also be accessed directly from the **Web part properties** dialog by creating a container using the **New** button or modifying the selected one via **Edit**.



Here's an example of a web part **without a container**:

Here's an example of the same web part **with a container**:



Please see the Creating web part containers topic for a step-by-step guide on how a new container can be defined.

7.27.8.2 Creating web part containers

The following is an example of how a new web part container can be defined:

1. Go to **Site Manager -> Development -> Web part containers**.
2. Click **New container**.
3. Enter the following values:

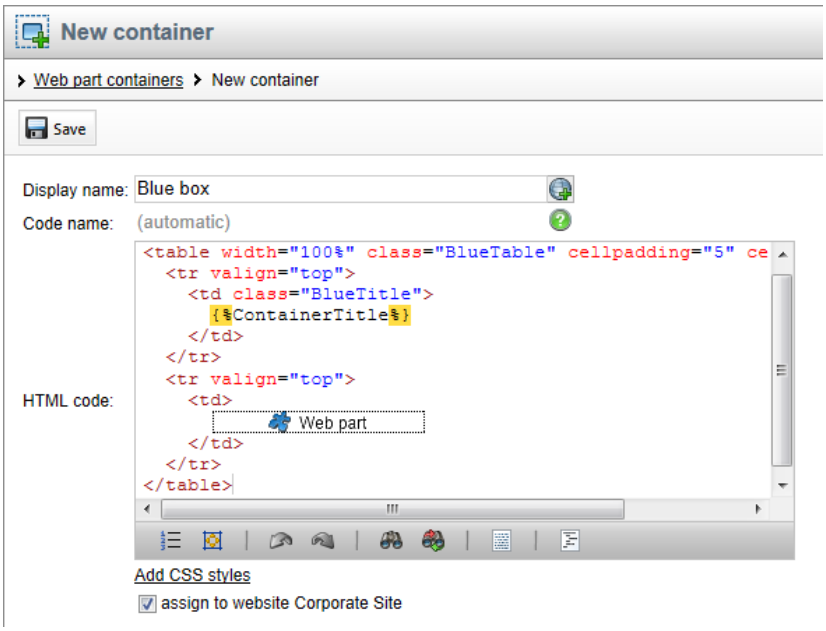
- **Display name**: Blue box
- **HTML code**:

```
<table width="100%" class="BlueTable" cellpadding="5" cellspacing="0">
  <tr valign="top">
    <td class="BlueTitle">
      {%ContainerTitle%}
    </td>
  </tr>
  <tr valign="top">
    <td>
      ?
    </td>
  </tr>
</table>
```

Working with the web part placeholder

The "?" character in the code above will determine the position of the web part placeholder when you paste it into the **HTML code** field.

If you ever need to set a container's HTML code using the API or in the database, you do not have to worry about the placeholder character, because the content before and after the placeholder is stored separately in two fields: **ContainerTextBefore** and **ContainerTextAfter**.



4. Click the Add CSS styles link and define the used CSS classes in the newly displayed field:

```
.BlueTable
{
    border: 1px solid #4a62e4;
}

.BlueTitle
{
    background-color:#4a62e4;
    font-weight:bold;
    color:white
}
```

5. Click **Save** to create the new container.

6. Switch to the **Sites** tab and click the **Add sites** button to assign the new container to the website where you wish to use it.

7. Go to **CMS Desk** and edit any page on the **Design** tab. **Configure** () any web part and set its properties according to the following:

- **Web part container**: Blue box
- **Container title**: My web part with a container

8. View the page in **Live site** mode. The web part will be surrounded by a blue border and it will have the specified title.

7.27.9 Web parts internals and API

7.27.9.1 Overview

In this chapter, you will learn which database tables and API classes are used to manage web parts. You will also see the most common API examples.

Further API-related information and examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all members of the related classes, please refer to Kentico CMS API Reference.

7.27.9.2 Database tables

The following database tables are used to store information about web parts:

- **CMS_WebPartCategory** - contains records representing web part categories.
- **CMS_WebPart** - contains records representing web parts and their properties.
- **CMS_WebPartLayout** - stores custom layouts for web parts.
- **CMS_WebPartContainer** - contains records representing web part containers.
- **CMS_WebPartContainerSite** - stores relationships between web part containers and sites. Each entry in this table indicates that a specific container may be used on a given site.

Web part instance storage

Data about the content of zones on individual page templates, including web parts and their property configuration, is stored in XML format in the **PageTemplateWebParts** column of the **CMS_PageTemplate** table.

7.27.9.3 API classes

If you are not familiar with database table data management through Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

The classes for managing web parts can be found in the **CMS.PortalEngine** namespace.

CMS_WebPartCategory table API:

- **WebPartCategoryInfo** - represents one web part category.
- **WebPartCategoryInfoProvider** - provides management functionality for web part categories.

CMS_WebPart table API:

- **WebPartInfo** - represents one web part.
- **WebPartInfoProvider** - provides management functionality for web parts.

CMS_WebPartLayout table API:

- **WebPartLayoutInfo** - represents a custom layout for a specific web part.
- **WebPartLayoutInfoProvider** - provides management functionality for web part layouts.

CMS_WebPartContainer table API:

- **WebPartContainerInfo** - represents one web part container.
- **WebPartContainerInfoProvider** - provides management functionality for web part containers.

CMS_WebPartContainerSite table API:

- **WebPartContainerInfo** - represents a relationship between a web part container and a site.
- **WebPartContainerInfoProvider** - provides management functionality for container-site relationships.

Other classes:

- **WebPartInstance** - represents a single instance of a web part.
- **WebPartZoneInstance** - represents an instance of a web part zone.

7.27.9.4 API examples

7.27.9.4.1 Overview

These topics show examples of how the API for managing web parts can be used:

- Managing web part categories
- Managing web parts
- Managing web part layouts
- Managing web part containers
- Web part containers and sites

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Development\Webparts\Default.aspx.cs***.

The web part API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.GlobalHelper;
using CMS.CMSHelper;
using CMS.PortalEngine;
```

7.27.9.4.2 Managing w eb part categories

The following example creates a web part category.

```
private bool CreateWebPartCategory()
{
    WebPartCategoryInfo root =
WebPartCategoryInfoProvider.GetWebPartCategoryInfoByCodeName("/");

    if (root != null)
    {
        // Create new web part category object
        WebPartCategoryInfo newCategory = new WebPartCategoryInfo();

        // Set the properties
        newCategory.CategoryDisplayName = "My new category";
        newCategory.CategoryName = "MyNewCategory";
        newCategory.CategoryParentID = root.CategoryID;

        // Save the web part category
```

```
        WebPartCategoryInfoProvider.SetWebPartCategoryInfo(newCategory);

        return true;
    }

    return false;
}
```

The following example gets and updates a web part category.

```
private bool GetAndUpdateWebPartCategory()
{
    // Get the web part category
    WebPartCategoryInfo updateCategory =
WebPartCategoryInfoProvider.GetWebPartCategoryInfoByCodeName("MyNewCategory");
    if (updateCategory != null)
    {
        // Update the properties
        updateCategory.CategoryDisplayName = updateCategory.CategoryDisplayName.ToLo

        // Save the changes
        WebPartCategoryInfoProvider.SetWebPartCategoryInfo(updateCategory);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates web part categories.

```
private bool GetAndBulkUpdateWebPartCategories()
{
    // Prepare the parameters
    string where = "CategoryDisplayName LIKE N'My new category%'";

    // Get the data
    DataSet categories = WebPartCategoryInfoProvider.GetCategories(where, null);
    if (!DataHelper.DataSourceIsEmpty(categories))
    {
        // Loop through the individual items
        foreach (DataRow categoryDr in categories.Tables[0].Rows)
        {
            // Create object from DataRow
            WebPartCategoryInfo modifyCategory = new WebPartCategoryInfo(categoryDr)

            // Update the properties
            modifyCategory.CategoryDisplayName =
modifyCategory.CategoryDisplayName.ToUpper();

            // Save the changes
```

```
            WebPartCategoryInfoProvider.SetWebPartCategoryInfo(modifyCategory);
        }

        return true;
    }

    return false;
}
```

The following example deletes a web part category.

```
private bool DeleteWebPartCategory()
{
    // Get the web part category
    WebPartCategoryInfo deleteCategory =
WebPartCategoryInfoProvider.GetWebPartCategoryInfoByCodeName("MyNewCategory");

    // Delete the web part category
    WebPartCategoryInfoProvider.DeleteCategoryInfo(deleteCategory);

    return (deleteCategory != null);
}
```

7.27.9.4.3 Managing w eb parts

The following example creates a web part.

```
private bool CreateWebPart()
{
    // Get parent category for web part
    WebPartCategoryInfo category =
WebPartCategoryInfoProvider.GetWebPartCategoryInfoByCodeName("MyNewCategory");

    if (category != null)
    {
        // Create new web part object
        WebPartInfo newWebpart = new WebPartInfo();

        // Set the properties
        newWebpart.WebPartDisplayName = "My new web part";
        newWebpart.WebPartName = "MyNewWebpart";
        newWebpart.WebPartDescription = "This is my new web part.";
        newWebpart.WebPartFileName = "whatever";
        newWebpart.WebPartProperties = "<form></form>";
        newWebpart.WebPartCategoryID = category.CategoryID;

        // Save the web part
        WebPartInfoProvider.SetWebPartInfo(newWebpart);

        return true;
```

```
    }

    return false;
}
```

The following example gets and updates a web part.

```
private bool GetAndUpdateWebPart()
{
    // Get the web part
    WebPartInfo updateWebpart = WebPartInfoProvider.GetWebPartInfo("MyNewWebpart");
    if (updateWebpart != null)
    {
        // Update the properties
        updateWebpart.WebPartDisplayName =
updateWebpart.WebPartDisplayName.ToLower();

        // Save the changes
        WebPartInfoProvider.SetWebPartInfo(updateWebpart);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates web parts.

```
private bool GetAndBulkUpdateWebParts()
{
    // Prepare the parameters
    string where = "WebPartName LIKE N'MyNewWebpart%'";

    // Get the data
    DataSet webparts = WebPartInfoProvider.GetWebParts(where, null);
    if (!DataHelper.DataSourceIsEmpty(webparts))
    {
        // Loop through the individual items
        foreach (DataRow webpartDr in webparts.Tables[0].Rows)
        {
            // Create object from DataRow
            WebPartInfo modifyWebpart = new WebPartInfo(webpartDr);

            // Update the properties
            modifyWebpart.WebPartDisplayName =
modifyWebpart.WebPartDisplayName.ToUpper();

            // Save the changes
            WebPartInfoProvider.SetWebPartInfo(modifyWebpart);
        }
```

```
            return true;
        }

        return false;
    }
```

The following example deletes a web part.

```
private bool DeleteWebPart()
{
    // Get the web part
    WebPartInfo deleteWebpart = WebPartInfoProvider.GetWebPartInfo("MyNewWebpart");

    // Delete the web part
    WebPartInfoProvider.DeleteWebPartInfo(deleteWebpart);

    return (deleteWebpart != null);
}
```

7.27.9.4.4 Managing web part layouts

The following example creates a web part layout.

```
private bool CreateWebPartLayout()
{
    // Get the web part
    WebPartInfo webpart = WebPartInfoProvider.GetWebPartInfo("MyNewWebpart");
    if (webpart != null)
    {
        // Create new web part layout object
        WebPartLayoutInfo newLayout = new WebPartLayoutInfo();

        // Set the properties
        newLayout.WebPartLayoutDisplayName = "My new layout";
        newLayout.WebPartLayoutCodeName = "MyNewLayout";
        newLayout.WebPartLayoutWebPartID = webpart.WebPartID;
        newLayout.WebPartLayoutCode = "This is the new layout of MyNewWebpart
webpart.";

        // Save the web part layout
        WebPartLayoutInfoProvider.SetWebPartLayoutInfo(newLayout);

        return true;
    }

    return false;
}
```

The following example gets and updates a layout.

```
private bool GetAndUpdateWebPartLayout()
{
    // Get the web part layout
    WebPartLayoutInfo updateLayout =
WebPartLayoutInfoProvider.GetWebPartLayoutInfo("MyNewWebpart","MyNewLayout");
    if (updateLayout != null)
    {
        // Update the properties
        updateLayout.WebPartLayoutDisplayName =
updateLayout.WebPartLayoutDisplayName.ToLower();

        // Save the changes
        WebPartLayoutInfoProvider.SetWebPartLayoutInfo(updateLayout);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates layouts.

```
private bool GetAndBulkUpdateWebPartLayouts()
{
    // Prepare the parameters
    string where = "WebPartLayoutCodeName LIKE N'MyNewLayout%'";

    // Get the data
    DataSet layouts = WebPartLayoutInfoProvider.GetWebPartLayouts(where, null);
    if (!DataHelper.DataSourceIsEmpty(layouts))
    {
        // Loop through the individual items
        foreach (DataRow layoutDr in layouts.Tables[0].Rows)
        {
            // Create object from DataRow
            WebPartLayoutInfo modifyLayout = new WebPartLayoutInfo(layoutDr);

            // Update the properties
            modifyLayout.WebPartLayoutDisplayName =
modifyLayout.WebPartLayoutDisplayName.ToUpper();

            // Save the changes
            WebPartLayoutInfoProvider.SetWebPartLayoutInfo(modifyLayout);
        }

        return true;
    }

    return false;
}
```

The following example deletes a web part layout.

```
private bool DeleteWebPartLayout()
{
    // Get the web part layout
    WebPartLayoutInfo deleteLayout =
WebPartLayoutInfoProvider.GetWebPartLayoutInfo("MyNewWebpart", "MyNewLayout");

    // Delete the web part layout
    WebPartLayoutInfoProvider.DeleteWebPartLayoutInfo(deleteLayout);

    return (deleteLayout != null);
}
```

7.27.9.4.5 Managing w eb part containers

The following example creates a web part container.

```
private void CreateWebPartContainer()
{
    // Create new web part container object
    WebPartContainerInfo newContainer = new WebPartContainerInfo();

    // Set the properties
    newContainer.ContainerDisplayName = "My new container";
    newContainer.ContainerName = "MyNewContainer";
    newContainer.ContainerTextBefore = "<div class=\"myNewContainer\">";
    newContainer.ContainerTextAfter = "</div>";

    // Save the web part container
    WebPartContainerInfoProvider.SetWebPartContainerInfo(newContainer);
}
```

The following example gets and updates a container.

```
private bool GetAndUpdateWebPartContainer()
{
    // Get the web part container
    WebPartContainerInfo updateContainer =
WebPartContainerInfoProvider.GetWebPartContainerInfo("MyNewContainer");
    if (updateContainer != null)
    {
        // Update the properties
        updateContainer.ContainerDisplayName =
updateContainer.ContainerDisplayName.ToLower();

        // Save the changes
        WebPartContainerInfoProvider.SetWebPartContainerInfo(updateContainer);

        return true;
    }
```

```
    return false;
}
```

The following example gets and bulk updates containers.

```
private bool GetAndBulkUpdateWebPartContainers()
{
    // Prepare the parameters
    string where = "ContainerName LIKE N'MyNewContainer%'";
    string orderBy = "";
    int topN = 0;
    string columns = "";

    // Get the data
    DataSet containers = WebPartContainerInfoProvider.GetContainers(where,
orderBy, topN, columns);
    if (!DataHelper.DataSourceIsEmpty(containers))
    {
        // Loop through the individual items
        foreach (DataRow containerDr in containers.Tables[0].Rows)
        {
            // Create object from DataRow
            WebPartContainerInfo modifyContainer = new WebPartContainerInfo
(containerDr);

            // Update the properties
            modifyContainer.ContainerDisplayName =
modifyContainer.ContainerDisplayName.ToUpper();

            // Save the changes
            WebPartContainerInfoProvider.SetWebPartContainerInfo(modifyContainer);
        }

        return true;
    }

    return false;
}
```

The following example deletes a web part container.

```
private bool DeleteWebPartContainer()
{
    // Get the web part container
    WebPartContainerInfo deleteContainer =
WebPartContainerInfoProvider.GetWebPartContainerInfo("MyNewContainer");

    // Delete the web part container
    WebPartContainerInfoProvider.DeleteWebPartContainerInfo(deleteContainer);
```

```
    return (deleteContainer != null);
}
```

7.27.9.4.6 Web part containers and sites

The following example assigns a web part container to a site.

```
private bool AddWebPartContainerToSite()
{
    // Get the web part container
    WebPartContainerInfo container =
WebPartContainerInfoProvider.GetWebPartContainerInfo("MyNewContainer");
    if (container != null)
    {
        int containerId = container.ContainerID;
        int siteId = CMSContext.CurrentSiteID;

        // Save the binding
        WebPartContainerSiteInfoProvider.AddContainerToSite(containerId, siteId);

        return true;
    }

    return false;
}
```

The following example removes a container from a site.

```
private bool RemoveWebPartContainerFromSite()
{
    // Get the web part container
    WebPartContainerInfo removeContainer =
WebPartContainerInfoProvider.GetWebPartContainerInfo("MyNewContainer");
    if (removeContainer != null)
    {
        int siteId = CMSContext.CurrentSiteID;

        // Delete the binding
        WebPartContainerSiteInfoProvider.RemoveContainerFromSite
(removeContainer.ContainerID, siteId);

        return true;
    }

    return false;
}
```

7.28 Widgets

7.28.1 Overview

Widgets introduce support for the personalization of pages. This feature allows users and editors to edit the structure of page templates. The personalized settings are saved within the system and can be invoked from the live site by authorized users or through the CMS Desk interface in the case of website editors and administrators.

From a designer's point of view, widgets are the basic building blocks of page templates in the same way as Web parts. Like web parts, they are placed into zones, which must be configured to contain a certain type of widgets. For more information about web development basics, such as page templates, please refer to the Web development overview -> Portal engine development model chapter of this guide. Widgets may also be used on pages based on ASPX page templates as long as they contain the appropriate zones and are configured as described in ASPX page template development model -> Adding portal engine functionality to ASPX templates.

There is a predefined set of widgets included in the default Kentico CMS installation. A description of these widgets and their properties can be found in the Kentico CMS Widgets reference. Because all widgets are based on existing web parts, you can also add your own widgets. Widgets can be created and managed at **Site Manager** -> **Development** -> **Widgets**. The details are given in the Developing widgets topic.

As mentioned above, widgets provide the same functionality as ordinary web parts. However, users with the appropriate permissions can modify the properties of widgets and their placement on the web page, add and remove widgets from their pages and so on. Users are divided into four different groups according to the possible ways they can use widgets:

- Site developers/Administrators - define the placement of widget zones and their default content, select the web part properties that should be available when configuring widgets and manage all available widgets.
- Page editors - define the content of widget zones created on page templates by the site developers/ administrators.
- Group administrators - define the content of widget zones located on the pages of their group via the live site.
- Website users - customize the content of widget zones on personalizable pages via the live site.

Widgets can additionally be added directly into editable text areas as described in the Inline widgets topic. Dashboards in the administration interface also utilize widgets as customization components. Please refer to the Modules -> Dashboards chapter for detailed information.

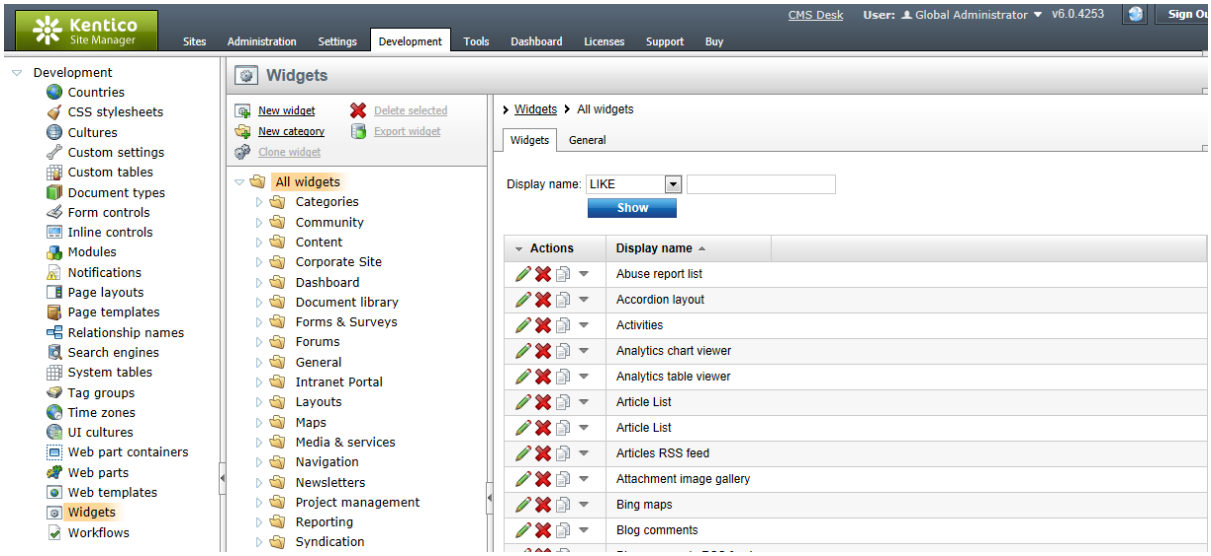
Security issues, such as the definition of where individual widgets can be used and permissions for different types of roles are discussed in the Security topic.

The Widget internals and API sub-chapter provides information about the database tables and classes used by widgets and examples of how widgets can be managed using the API.

7.28.2 Developing widgets

This topic leads you through the process of creating a widget. As a site developer, you can create your own widgets. Every widget is based on an existing web part, so if you require a widget with completely new custom functionality, you need to develop the appropriate web part first.

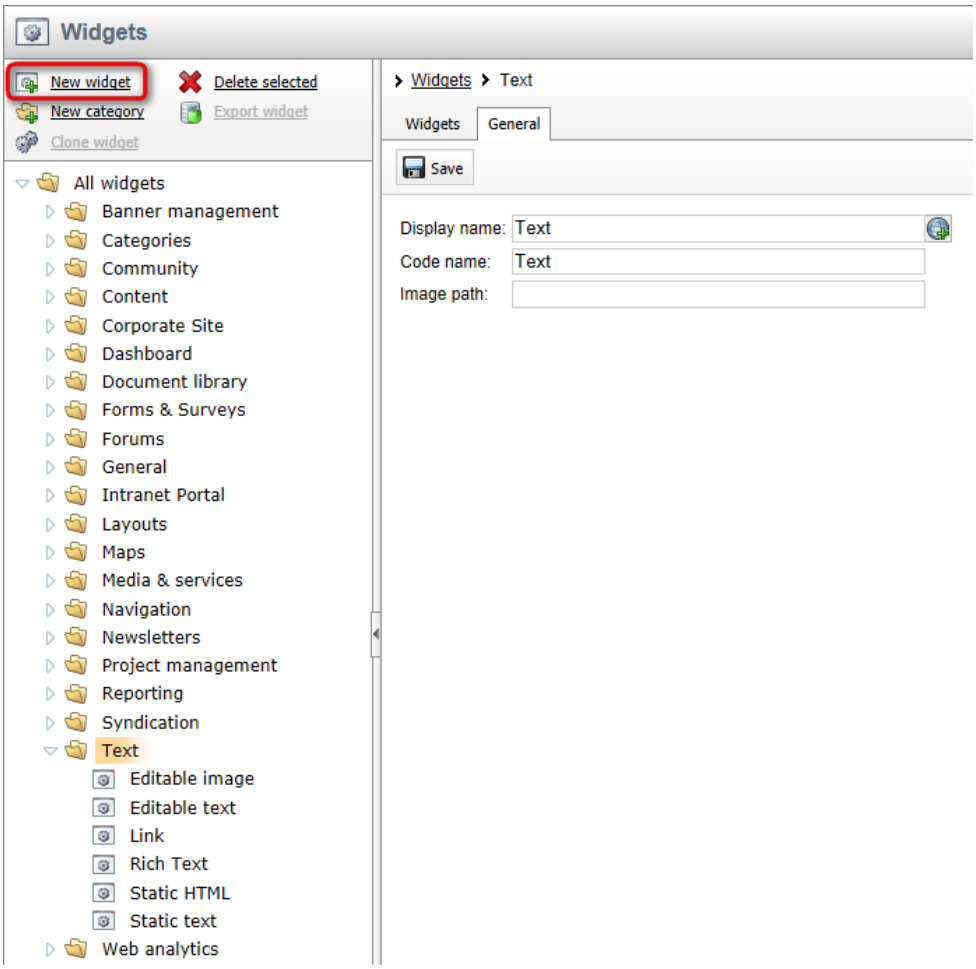
1. As mentioned above, it is not possible to create a widget without a reference to a web part. To start, switch to **Site Manager** -> **Development** -> **Widgets** to access the administration interface where widgets can be created. You should see a screen similar to the following one.



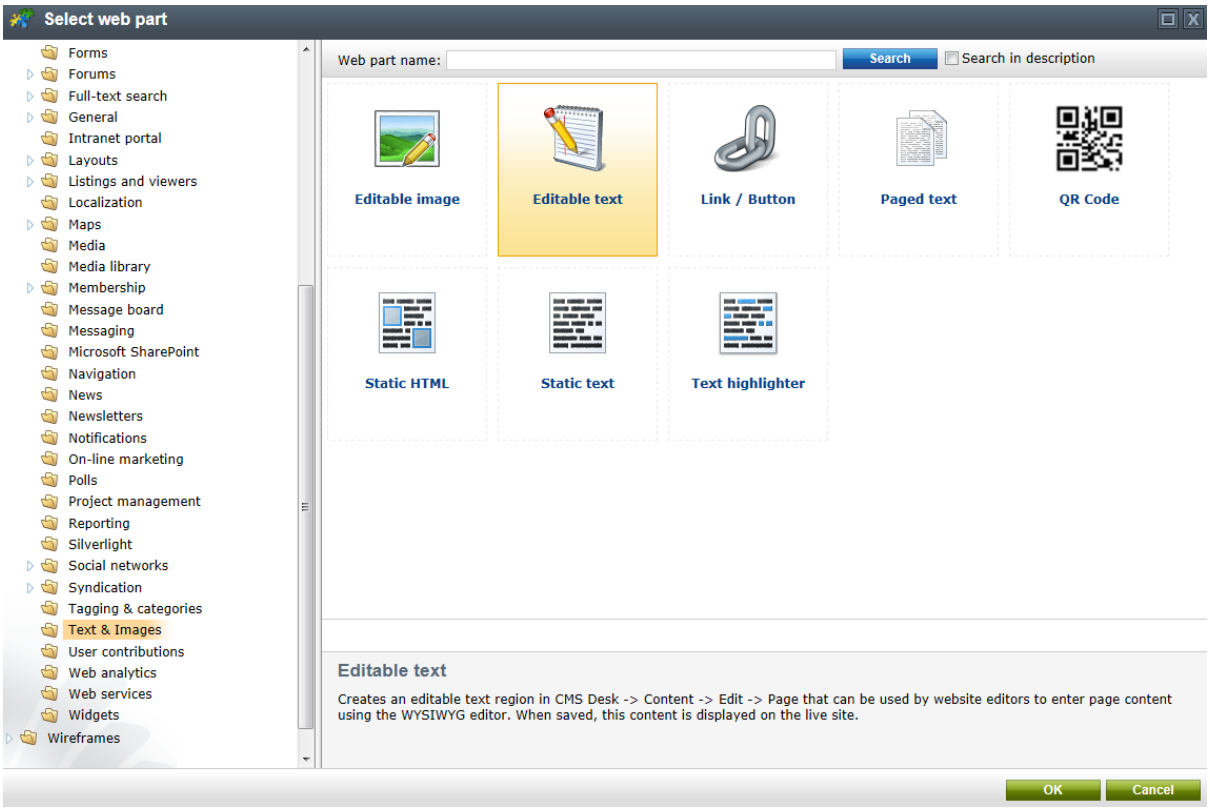
2. Widgets are grouped into categories. It is important to note that categories are used for the sake of organization only. The category under which a given widget is stored has no influence on who can use or modify the widget.

Clicking on a node causes the selected category to open and all widgets stored within that category are listed. You can also change the **Display** and **Code name** of a category on the **General** tab. The **Image path** property sets the path to the image that will be used as the category's icon in the **Add widget** dialog (the recommended image size is 16x16px).

Now select the category where the new widget should be stored, for example **Text**, and click the **New widget** link. Alternatively, you can create the new widget under any folder and change its location later when the properties of the widget are defined.



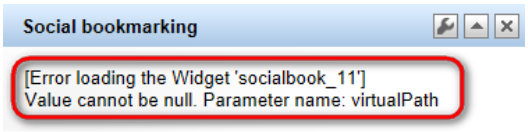
3. A dialog opens, where you can select the web part that will be used as a template for the new widget. In the example below, the **Editable text** web part from the **Text & Images** category is selected. Click the **OK** button to confirm the selection.



4. A new widget has just been created. You can manage widgets using the actions above the tree. A new widget can be created based on an existing one using **Clone widget**. If you wish to add a new category, use **New category**.

To delete a selected widget or an entire category (except the *All widgets* category), click **Delete selected**. Please bear in mind that when a widget is deleted, it is not automatically removed from page templates. Therefore, when pages containing a deleted widget are to be displayed, an error message will be displayed instead of the missing widget.

The example below shows the effect of deleting the **Social bookmarking** widget without removing it from the page templates.



A selected widget can also be exported by clicking on **Export widget**. For details regarding export see Exporting single objects.

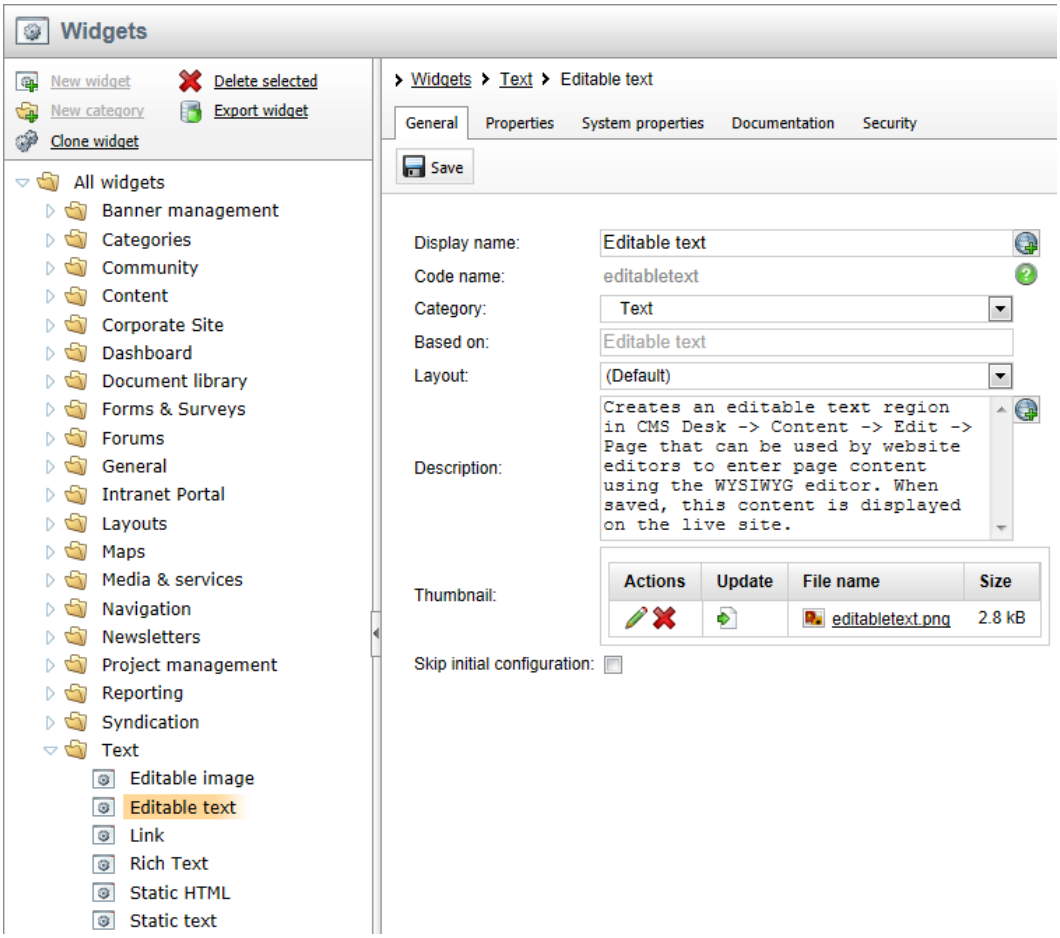
5. For each selected widget there are five tabs available. On the **General** tab there are all the basic settings:

- **Display name** - sets a name for the widget that will be used in the administration interface and widget selection dialog.
- **Code name** - sets a name that will serve as a unique identifier for the widget.
- **Category** - shows the category to which the widget belongs. You can change it here by selecting a

new category from the drop-down list.

- **Based on** - displays the name of the parent web part that the widget is derived from.
- **Layout** - allows you to choose one of the layouts defined for the parent web part. The selected layout will be applied to all instances of the widget across all sites. A layout may be used to modify the widget's appearance or design and even add further content into it. If you wish to use a different layout for certain instances of the same widget, you can make a clone of the widget for this purpose and have it use the alternative layout. For more information about web part layouts and how they can be created, please see the Customizing web part layout topic.
- **Description** - a text description that is displayed in the widget selection dialog and in widget documentation.
- **Thumbnail** - sets the image used as a graphical representation in the widget selection dialog.
- **Skip initial configuration** - if enabled, new instances of the widget will be added directly to the page without opening the property configuration dialog. This can make the widget easier to use and helps save time, particularly in the case of widgets that are usually left with their default property values.

Perform any changes required and click the **Save** button.

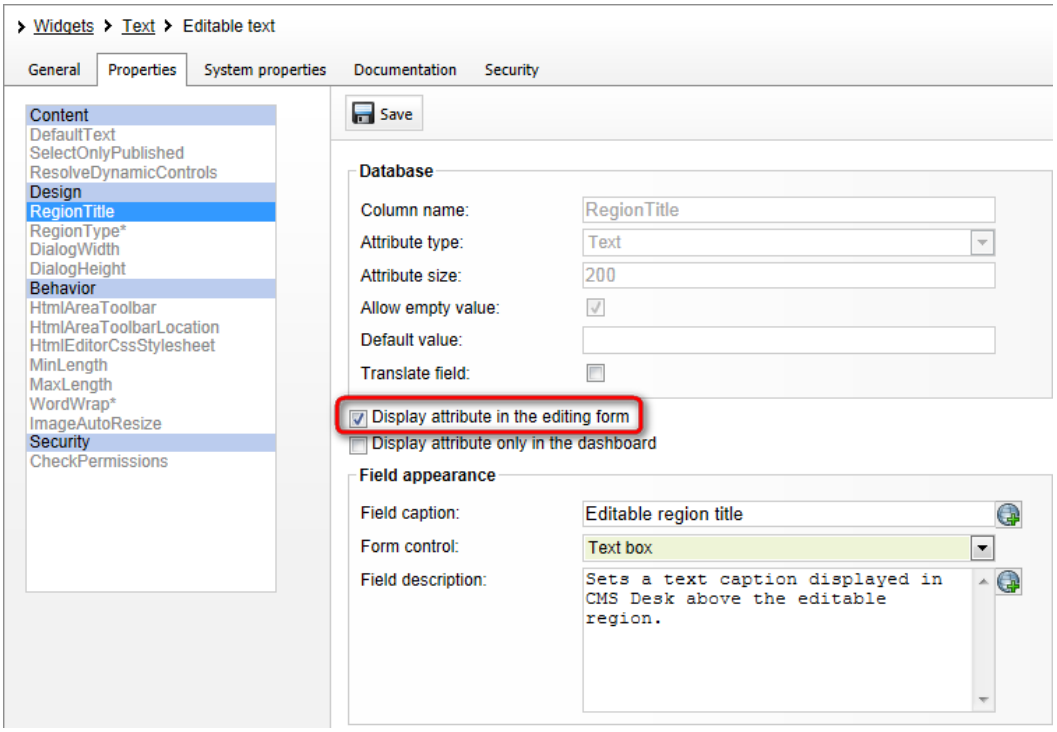


6. The **Properties** tab includes all the properties taken from the parent web part.

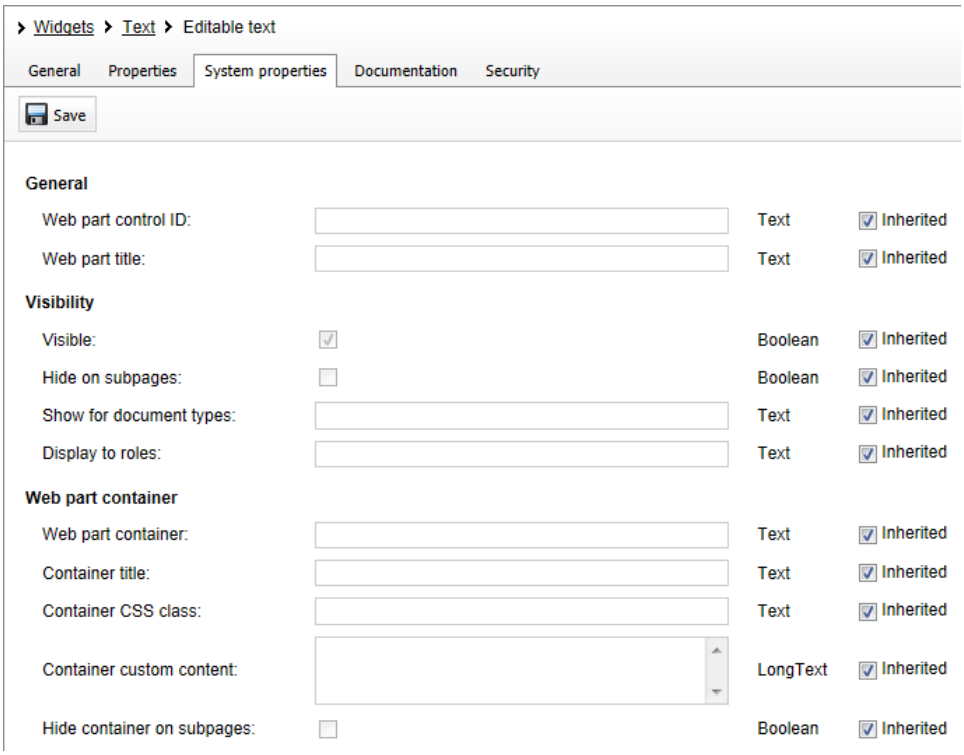
- Using the **Display attribute in the editing form** checkbox, you can choose if a given property should be available in the widget configuration dialog. You can notice that all properties are hidden by default.

- If the **Display attribute only in the dashboard** box is also checked, the property is only accessible when configuring the widget on one of the dashboard sections located in the CMS administration interface.

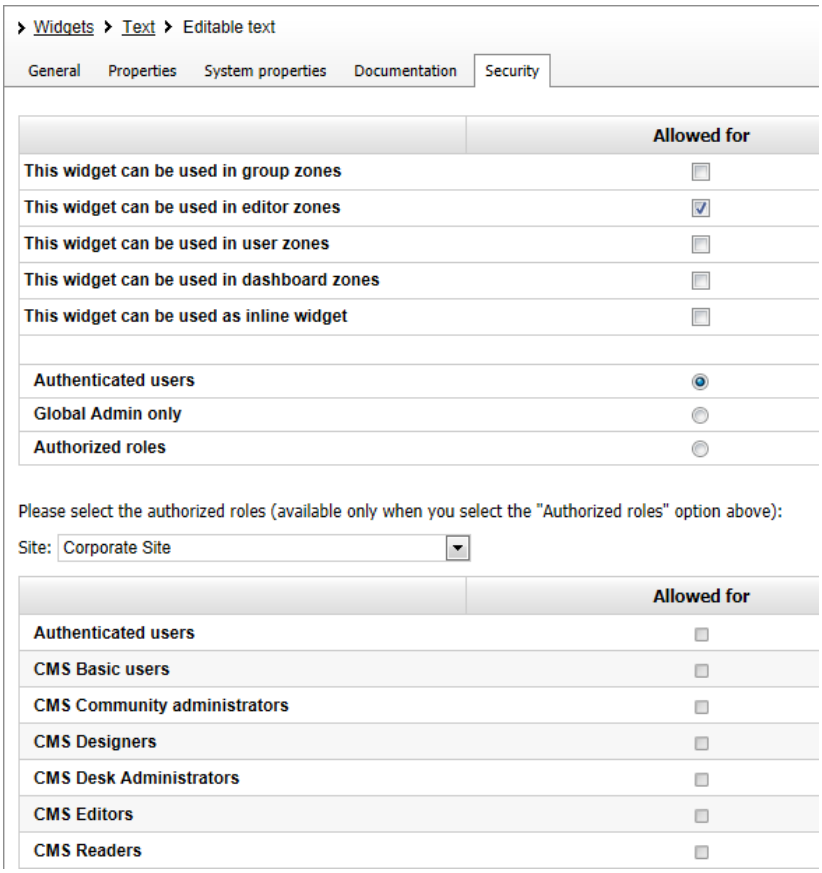
Changes must be confirmed by clicking **Save**.



7. The **System properties** tab contains the general properties that all widgets and web parts have in common. Their default values can be modified by clearing the **Inherited** checkbox and entering the desired value into the appropriate field. All changes must be confirmed by clicking **Save**.



8. On the **Security** tab, there are settings specifying where and by what type of users the widget can be used. The details are given in the widgets Security topic.



9. Once all settings are configured as needed, the widget is ready to be used on the website's pages.

7.28.3 Using widgets

Users can perform widget-related actions depending on their role on the website:

- Site developers and administrators - can determine which zones are widget zones (and their type) and which ones are standard (static) web part zones, and also define the default content of these zones. These tasks can be done in **CMS Desk -> Content -> Design**. They can also manage widgets in the administration interface in **Site Manager -> Development -> Widgets**. This includes developing new widgets, determining where and by whom they can be used and selecting the properties which should be personalizable (editable by users).
- Page editors - can define the content of Editor widget zones created on page templates by the site developers/administrators. This is accessible through **CMS Desk -> Content -> Edit -> Page**.
- Group administrators - can define the design of Group administrator widget zones on group page templates. Group administrators can add or remove widgets and set their properties just like page editors. This customization can be done on the live site.
- Website users - authenticated users of the website can customize the content of widget zones on personalizable pages on the live site. They can add or remove widgets and configure the widget properties. Only pages with templates that contain User widget zones can be personalized in this way.

Site developers, administrators and page editors can also use inline widgets. Please refer to the Inline widgets chapter for more details.

Another place where users can customize page content using widgets is on the dashboards located in the administration interface of Kentico CMS. Dashboard pages are based on special types of page templates and their widget functionality is very similar to that of User widget zones. More detailed information about widgets on dashboards can be found in the Modules -> Dashboards -> Managing dashboard content topic.

Site developers/Administrators

The site developers make the decisions regarding the layout of page templates. This also includes the decision whether a zone will be defined as a web part zone (static) or widget zone (customizable).

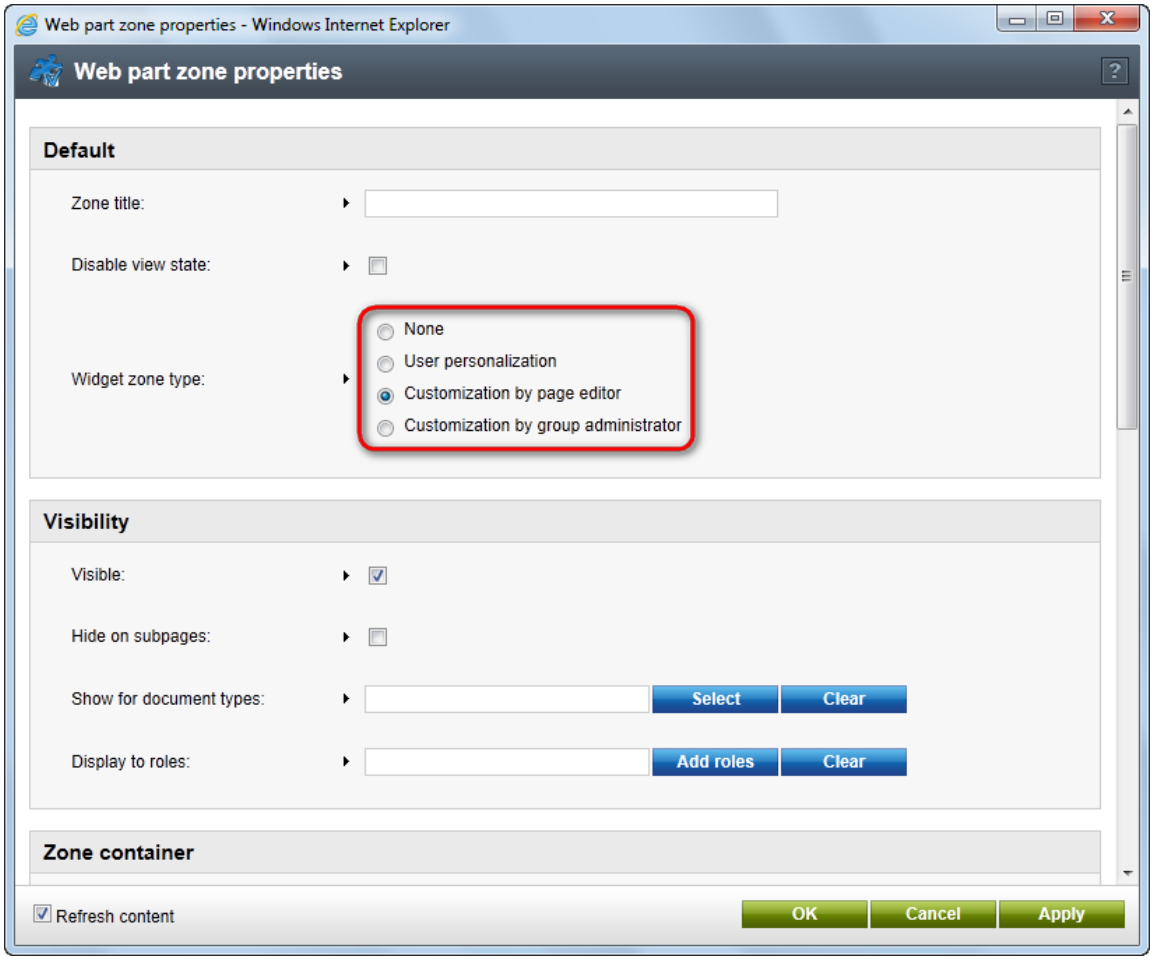
To change a web part zone into a widget zone, go to **CMS Desk -> Content -> Design**, expand the menu of the selected web part zone (▾) and click on the **Properties** item.

Using the **Web zone part properties** dialog you can define the **Widget zone type** property. The following options are available:

- **None** - a standard web part zone is used.
- **User personalization** - the zone will be a widget zone and website users will be able to personalize it on the live site. The zone will be marked with the *User zone* ( ) icon.
- **Customization by page editor** - the zone will be a widget zone and website editors will be able to customize it. The zone will be marked with the *Editor zone* ( ) icon.
- **Customization by group administrator** - the zone will be a widget zone and group administrators of the group that owns the page will be able to customize it. The zone will be marked with the *Group*

administrator zone (icon) icon.

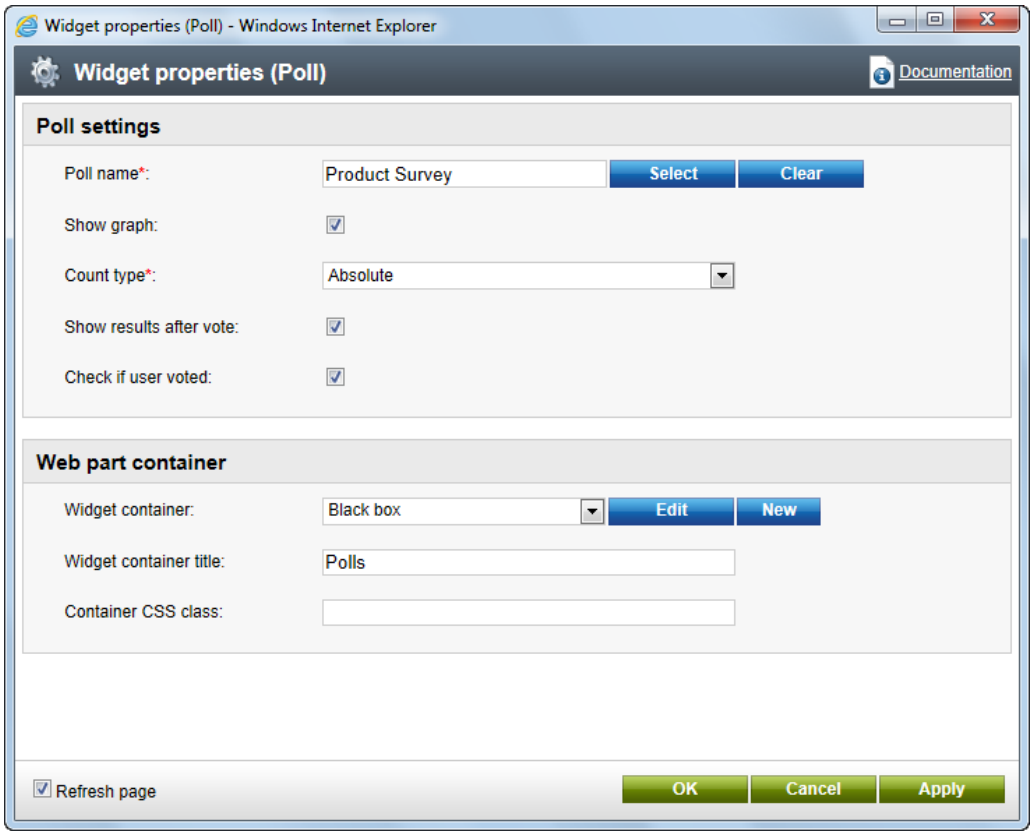
Please bear in mind that changing the widget zone type removes all current web parts or default widgets placed in that zone.



Having defined the zone as a widget type zone, you can now add widgets. Simply click on the **Add widget** (+) button on the right side of the zone.

You can select any widget available for the given type of widget zone (User, Editor, Group administrator).

A **Widget properties** dialog opens (unless the given widget is configured to skip it), where you can change any properties available for customization. Click **OK** to add the widget.



This way, developers can define the default content of widget zones. It is up to the users to personalize these zones further as described in the remaining sections of this topic.

Page editors

As a page editor, you can add, modify or delete any widget placed in editor type widget zones. This can be used to allow editors to quickly customize pages beyond simply entering content into static page templates.

To add a widget, just click on the **Add widget** ( ) button in the top left corner of an editor widget zone.

You can modify the properties of any widget placed in an editor zone by clicking the **Configure widget** ( ) button. You can also delete widgets using the **Delete widget** ( ) button. To move widgets between zones, hover over **Drag widget** ( ) and drag the widget to the desired place.

Group administrators

Group administrators (members of the group who are assigned to a group role that can manage the group) have basically the same options as page editors. They can personalize those pages that belong to their group that contain group administrator widget zones by adding, deleting and configuring widgets in these zones. This is done on the live site, as group administrators won't necessarily have access to the **CMS Desk -> Edit** interface. Other website users who have access to the group pages will be able to see the widgets, but won't be able to manage them.

For specific information about groups, please see the Modules -> Groups chapter of this guide.

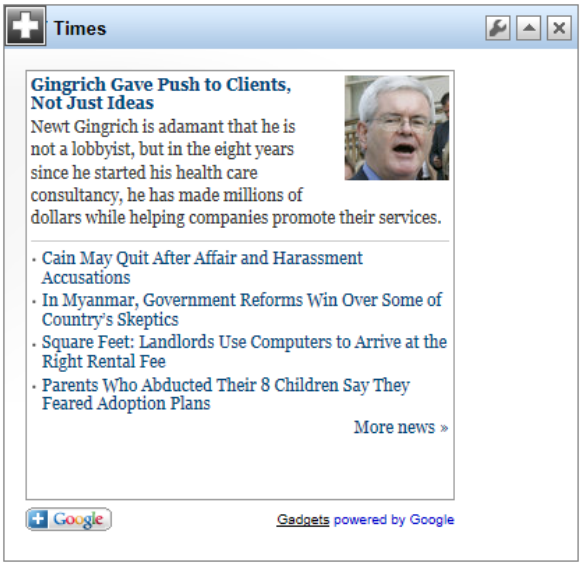
Note

Group ownership of a document can be configured in **CMS Desk -> Edit -> Properties** through the **Owned by group** field.

Website users

Authenticated website users can manipulate user widget zones on the live site and thus personalize their pages. A common use is placing user widget zones onto the home page, so that users can create their own personalized version of it.

In order to add a widget to a user widget zone on the live site, click on the **Add widget** ( ) button in the top left corner of the zone.



Any widget can be configured by clicking on the **Configure widget** ( ) button. You can also delete

widgets using the **Delete widget** ( ) button. To move widgets between zones or to different positions, click anywhere on the header of the given widget and drag it to the desired location. Any widget can be minimized using the **Minimize** ( ) button, or restored using the **Maximize** ( ) button.

Widget actions web part

You can place the Widget actions web part onto pages containing widget zones to allow users to:

- Add widgets
- Reset all widget zones to their default content

The web part only works with the type of widget zone specified in its **Widget zone type** property. When a user clicks the **Add widget** button, the web part places the widget into the zone specified by the **Widget zone** property. Users can drag the widget into another zone if the default placement is not suitable.

Tip

When creating pages with *editor* widget zones, you can enable the **Use main menu** property of the Widget actions web part. This causes the web part to display the action buttons on the main edit menu of the **Page** tab in CMS Desk, instead of in the page content.

Using macros in widget properties

For security reasons, the system does *not* resolve macro expressions entered by users into the properties of widget instances. However, you can pre-set macros as the default values of widget properties:

1. Go to **Site Manager -> Development -> Widgets**.
2. Select the given widget in the tree catalog and open the **Properties** tab.
3. Enter the required macro expression into the **Default value** field of the appropriate property.
4. Disable the **Display attribute in the editing form** option for the property.
5. Click **Save**.

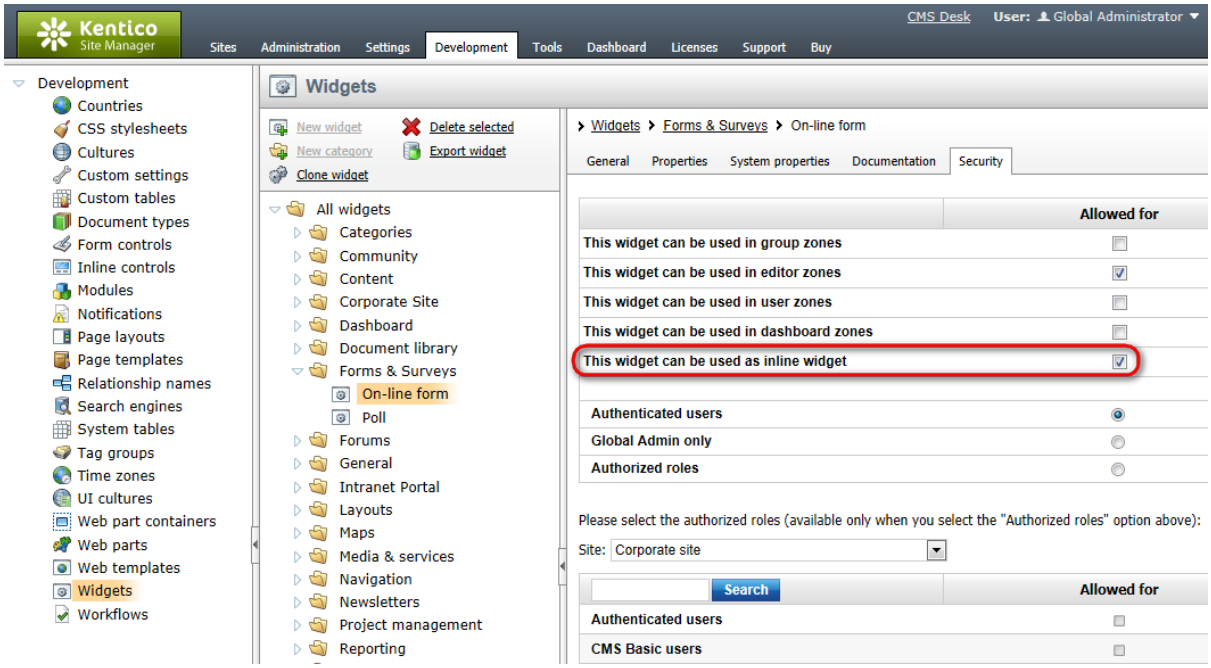
When the website renders a page containing the widget, the system resolves the macro and dynamically inserts the return value into the property.

7.28.4 Inline widgets

Inline widgets allow administrators and content editors to add widgets directly into the text of a page. These widgets are not contained in any widget zones, so they cannot be customized on the live site. Their main purpose is to give content editors, who do not have **Design** permissions, the option of customizing editable regions using widgets.

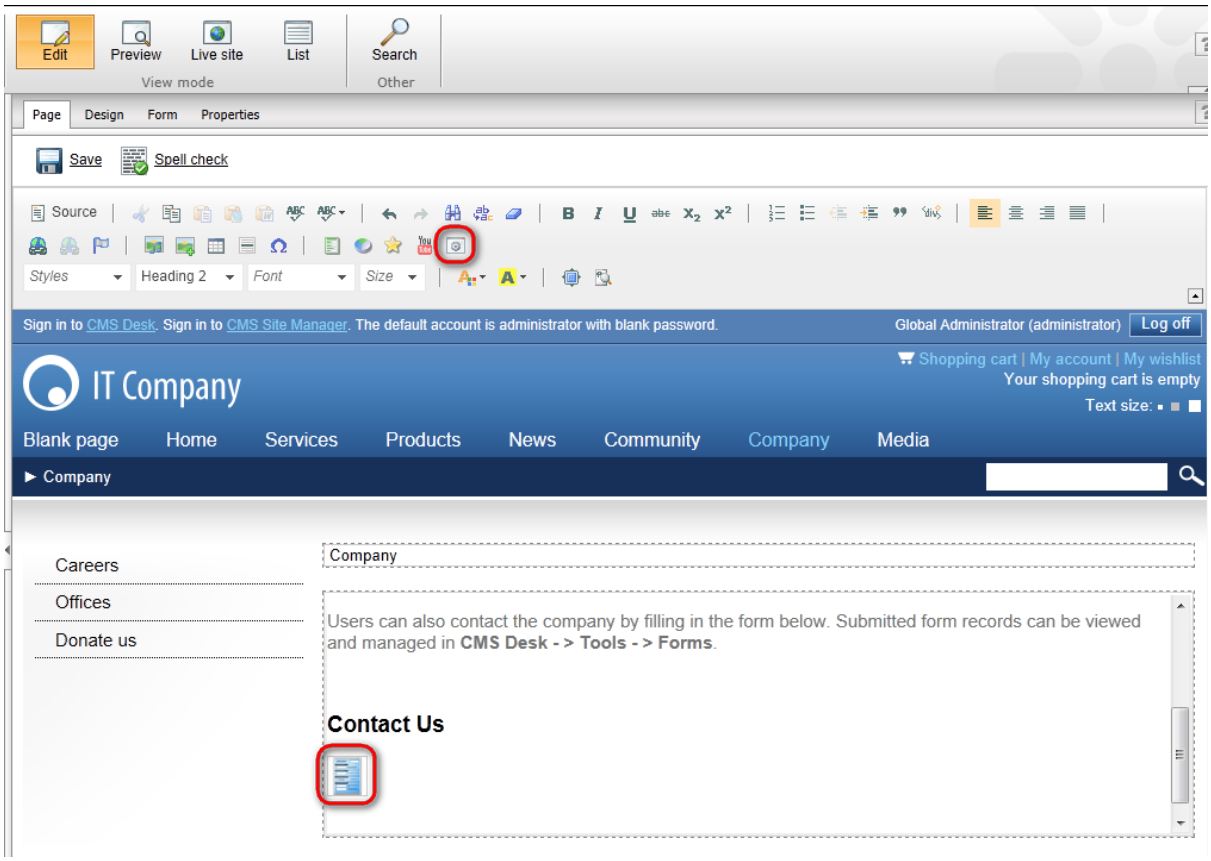
Inserting inline widgets into text

Inline widgets can be inserted into text by using the **Insert/Edit widget** ( ) button on toolbar of the WYSIWYG editor. This opens the same selection dialog as when adding a widget to a widget zone. Only widgets that are allowed to be used inline can be selected. This option can be configured when editing a widget on its **Security** tab in **Site Manager -> Development -> Widgets**.

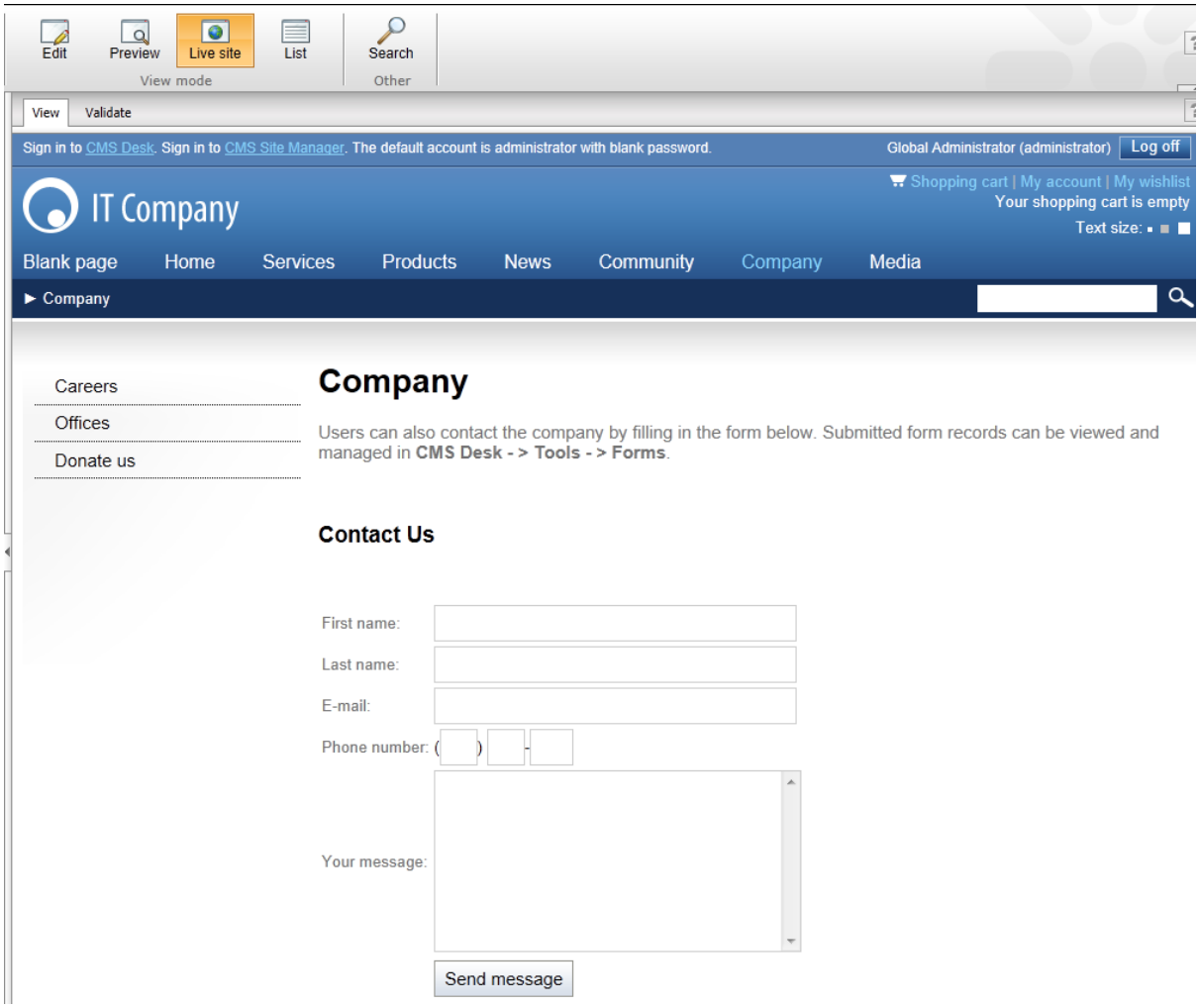


After you select the desired widget, the **Widget properties** dialog opens, where you can configure any of the widget's available properties. Once inserted into the text, an inline widget is represented by a placeholder image on the **Edit** tab of **CMS Desk** and is displayed fully on the **Preview** tab and the live site. Remember to **Save** the page once you are done.

The following images show an example of using an inline widget to add a form into text:



The Form is displayed fully on the **Live site**:



Configuring inline widgets

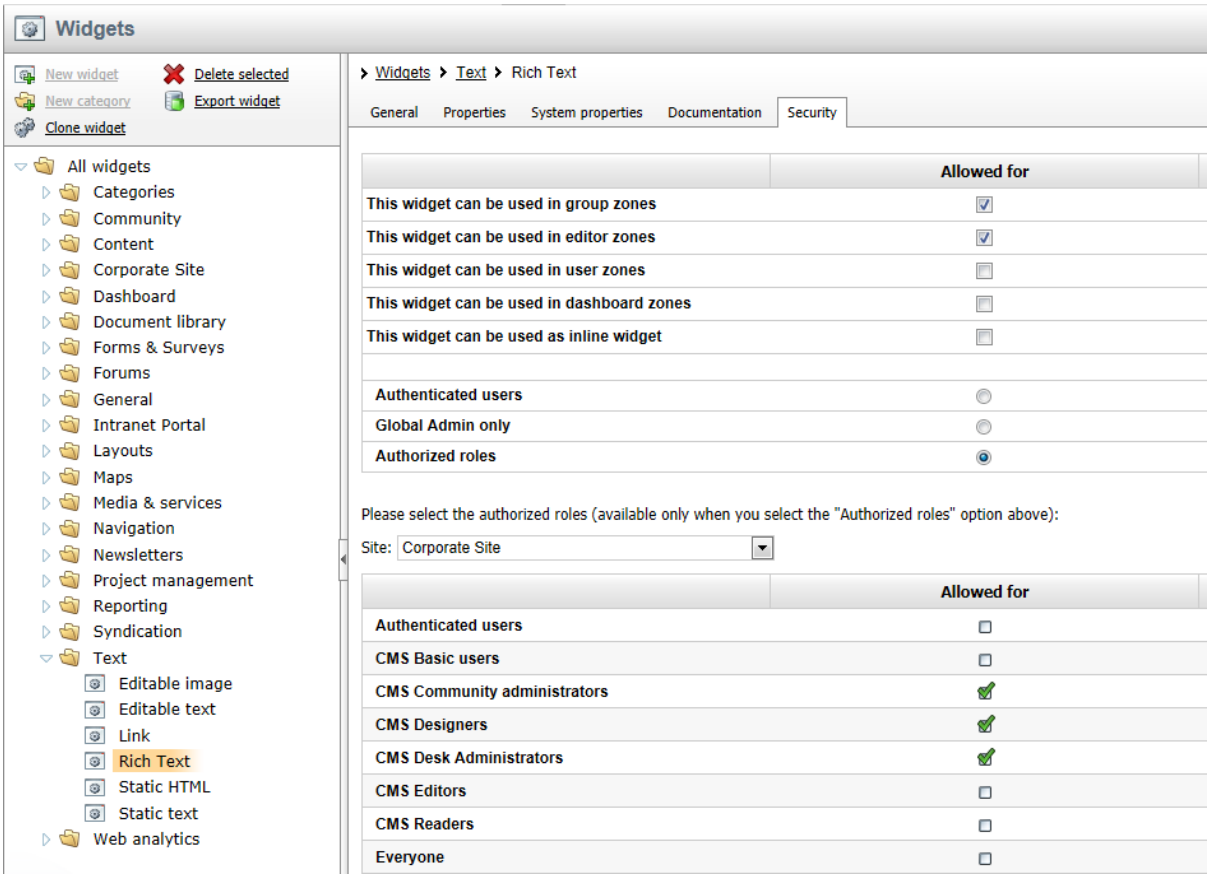
There are three different ways to open an inline widget's properties dialog:

- Double-click its placeholder image in the text
- Right-click its placeholder image in the text and select **Properties**
- Select its placeholder image in the text and click the **Insert/Edit widget** button ( ) on the WYSIWYG editor toolbar

7.28.5 Security

When a new widget is created from an existing web part, all the attributes are hidden in the editing form by default. It is up to the widget creator to select the attributes which should be available for customization using the check boxes on the **Properties** tab.

The security options are defined on the **Security** tab when selecting a widget from the content tree in **Site Manager -> Development -> Widgets**. By default, all widgets are forbidden for all zone types and are allowed for authorized roles only. However, no authorized role is selected by default. It is up to the developers to allow a widget for specific types of zones and users.



Widgets can be allowed for the following locations:

- **In group, editor or user zones** - the widget may be used in the respective type of widget zone.
- **In dashboard zones** - the widget may be used as a component on dashboards.
- **As inline widget** - the widget can be inserted into editable text regions via the WYSIWYG editor.

In addition, the permissions of a widget must also be set for one of the following types of users:

- **Authenticated users** - all logged in users are allowed to use the widget.
- **Global Admin only** - only users designated as global administrators are allowed to use the widget.
- **Authorized roles** - only members of the roles selected in the section below are allowed to use the widget.

Please keep in mind that changing the security settings will only affect new widgets. If a user was allowed to add a widget and an administrator later removes this permission, the user can still see the widget on their page. However, once deleted, the widget cannot be added back to the page without allowing it on the **Security** tab of that particular widget.

7.28.6 Widgets internals and API

7.28.6.1 Overview

In this chapter, you will learn which database tables and API classes are used to manage widgets. You will also see the most common API examples.

Further API-related information and examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all members of the related classes, please refer to Kentico CMS API Reference.

7.28.6.2 Database tables

The following database tables are used to store information about widgets:

- **CMS_WidgetCategory** - contains records representing widget categories.
- **CMS_Widget** - contains records representing widgets and their configuration.
- **CMS_WidgetRole** - stores relationships between widgets and roles. Each entry in this table indicates that a specific widget can be used by users in a given role.

Widget instance storage

Data about widget instances and their configuration is also stored in the database, but the location depends on the type of the widget.

The content of zones on page templates, including default widgets and their property configuration, is stored in XML format in the **PageTemplateWebParts** column of the **CMS_PageTemplate** table. Widget instances placed onto individual documents by page editors are stored using the same format in the **DocumentWebParts** column of the **CMS_Document** table.

The data about widget instances contained in personalized user and dashboard widget zones, which depends on the current context (user and site), is saved in the **CMS_Personalization** table.

Inline widgets are saved as control macro expressions within the text content of the document onto which they are placed. For standard editable regions on pages, this content is stored in the **DocumentContent** column of the **CMS_Document** table. If the inline widget is inserted into a document field via the **Form** tab of **CMS Desk**, it will be saved in the table that stores documents of the given type (e.g. **CONTENT_News** for news documents).

7.28.6.3 API classes

If you are not familiar with database table data management through Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

The classes for managing widgets can be found in the **CMS.PortalEngine** namespace.

CMS_WidgetCategory table API:

- **WidgetCategoryInfo** - represents one widget category.
- **WidgetCategoryInfoProvider** - provides management functionality for widget categories.

CMS_Widget table API:

- **WidgetInfo** - represents one widget object.
- **WidgetInfoProvider** - provides management functionality for widgets.

CMS_WidgetRole table API:

- **WidgetRoleInfo** - represents a relationship between a widget and a role.
- **WidgetRoleInfoProvider** - provides management functionality for widget-role relationships.

CMS_Personalization table API:

- **PersonalizationInfo** - represents a personalized version of a page for a specific site and user.
- **PersonalizationInfoProvider** - provides management functionality for personalization objects.

Other classes:

- **WebPartInstance** - can be used to represent a single instance of a widget.
- **WebPartZoneInstance** - represents an instance of a web part (widget) zone.

7.28.6.4 API examples

7.28.6.4.1 Overview

These topics show examples of how the API for managing widgets can be used:

- Managing widget categories
- Managing widgets
- Managing widget security

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Development\Widgets\Default.aspx.cs***.

The widget API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.GlobalHelper;
using CMS.CMSHelper;
using CMS.SiteProvider;
using CMS.PortalEngine;
using CMS.FormEngine;
```

7.28.6.4.2 Managing w idget categories

The following example creates a widget category.

```
private void CreateWidgetCategory()
{
    // Create new widget category object
    WidgetCategoryInfo newCategory = new WidgetCategoryInfo();

    // Set the properties
    newCategory.WidgetCategoryDisplayName = "My new category";
    newCategory.WidgetCategoryName = "MyNewCategory";

    // Save the widget category
    WidgetCategoryInfoProvider.SetWidgetCategoryInfo(newCategory);
}
```

The following example gets and updates a widget category.

```
private bool GetAndUpdateWidgetCategory()
{
    // Get the widget category
    WidgetCategoryInfo updateCategory =
WidgetCategoryInfoProvider.GetWidgetCategoryInfo("MyNewCategory");
    if (updateCategory != null)
    {
        // Update the properties
        updateCategory.WidgetCategoryDisplayName =
updateCategory.WidgetCategoryDisplayName.ToLower();

        // Save the changes
        WidgetCategoryInfoProvider.SetWidgetCategoryInfo(updateCategory);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates widget categories.

```
private bool GetAndBulkUpdateWidgetCategories()
{
    // Prepare the parameters
    string where = "WidgetCategoryName LIKE N'MyNewCategory%'";
    string orderBy = "";
    int topN = 0;
    string columns = "";

    // Get the data
    DataSet categories = WidgetCategoryInfoProvider.GetWidgetCategories(where,
orderBy, topN, columns);
    if (!DataHelper.DataSourceIsEmpty(categories))
    {
        // Loop through the individual items
        foreach (DataRow categoryDr in categories.Tables[0].Rows)
        {
            // Create object from DataRow
            WidgetCategoryInfo modifyCategory = new WidgetCategoryInfo(categoryDr);

            // Update the properties
            modifyCategory.WidgetCategoryDisplayName =
modifyCategory.WidgetCategoryDisplayName.ToUpper();

            // Save the changes
            WidgetCategoryInfoProvider.SetWidgetCategoryInfo(modifyCategory);
        }

        return true;
    }

    return false;
```

```
}
```

The following example deletes a widget category.

```
private bool DeleteWidgetCategory()
{
    // Get the widget category
    WidgetCategoryInfo deleteCategory =
WidgetCategoryInfoProvider.GetWidgetCategoryInfo("MyNewCategory");

    // Delete the widget category
    WidgetCategoryInfoProvider.DeleteWidgetCategoryInfo(deleteCategory);

    return (deleteCategory != null);
}
```

7.28.6.4.3 Managing w idgets

The following example creates a widget.

```
private bool CreateWidget()
{
    // Get parent web part and category for widget
    WebPartInfo webpart = WebPartInfoProvider.GetWebPartInfo("AbuseReport");
    WidgetCategoryInfo category = WidgetCategoryInfoProvider.GetWidgetCategoryInfo
("MyNewCategory");

    // Widgets cannot be created from an inherited web part
    if ((webpart != null) && (webpart.WebPartParentID == 0) && (category != null))
    {
        // Create new widget object
        WidgetInfo newWidget = new WidgetInfo();

        // Set the properties from the parent web part
        newWidget.WidgetName = "MyNewWidget";
        newWidget.WidgetDisplayName = "My new widget";
        newWidget.WidgetDescription = webpart.WebPartDescription;

        newWidget.WidgetProperties = FormHelper.GetFormFieldsWithDefaultValue
(webpart.WebPartProperties, "visible", "false");

        newWidget.WidgetWebPartID = webpart.WebPartID;
        newWidget.WidgetCategoryID = category.WidgetCategoryID;

        // Save new widget
        WidgetInfoProvider.SetWidgetInfo(newWidget);

        return true;
    }
```

```
    return false;
}
```

The following example gets and updates a widget.

```
private bool GetAndUpdateWidget()
{
    // Get the widget
    WidgetInfo updateWidget = WidgetInfoProvider.GetWidgetInfo("MyNewWidget");
    if (updateWidget != null)
    {
        // Update the properties
        updateWidget.WidgetDisplayName = updateWidget.WidgetDisplayName.ToLower();

        // Save the changes
        WidgetInfoProvider.SetWidgetInfo(updateWidget);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates widgets.

```
private bool GetAndBulkUpdateWidgets()
{
    // Prepare the parameters
    string where = "WidgetName LIKE N'MyNewWidget%'";
    string orderBy = "";
    int topN = 0;
    string columns = "";

    // Get the data
    DataSet widgets = WidgetInfoProvider.GetWidgets(where, orderBy, topN,
columns);
    if (!DataHelper.DataSourceIsEmpty(widgets))
    {
        // Loop through the individual items
        foreach (DataRow widgetDr in widgets.Tables[0].Rows)
        {
            // Create object from DataRow
            WidgetInfo modifyWidget = new WidgetInfo(widgetDr);

            // Update the properties
            modifyWidget.WidgetDisplayName =
modifyWidget.WidgetDisplayName.ToUpper();

            // Save the changes
            WidgetInfoProvider.SetWidgetInfo(modifyWidget);
        }
```

```
        return true;
    }

    return false;
}
```

The following example deletes a widget.

```
private bool DeleteWidget()
{
    // Get the widget
    WidgetInfo deleteWidget = WidgetInfoProvider.GetWidgetInfo("MyNewWidget");

    // Delete the widget
    WidgetInfoProvider.DeleteWidgetInfo(deleteWidget);

    return (deleteWidget != null);
}
```

7.28.6.4.4 Managing widget security

The following example assigns a specific role to a widget (the widget will be usable by users belonging to the given role).

```
private bool AddWidgetToRole()
{
    // Get role, widget and permission object
    RoleInfo role = RoleInfoProvider.GetRoleInfo("CMSDeskAdmin",
CMSContext.CurrentSiteID);
    WidgetInfo widget = WidgetInfoProvider.GetWidgetInfo("MyNewWidget");
    PermissionNameInfo permission =
PermissionNameInfoProvider.GetPermissionNameInfo("AllowedFor", "Widgets", null);

    // If all exist
    if ((role != null) && (widget != null) && (permission != null))
    {
        // Add role to widget
        WidgetRoleInfoProvider.AddRoleToWidget(role.RoleID, widget.WidgetID,
permission.PermissionId);

        return true;
    }

    return false;
}
```

The following example removes the relationship between a specific role and a widget.

```
private bool RemoveWidgetFromRole()
{
    // Get role, widget and permission object
    RoleInfo role = RoleInfoProvider.GetRoleInfo("CMSDeskAdmin",
CMSContext.CurrentSiteID);
    WidgetInfo widget = WidgetInfoProvider.GetWidgetInfo("MyNewWidget");
    PermissionNameInfo permission =
PermissionNameInfoProvider.GetPermissionNameInfo("AllowedFor", "Widgets", null);

    // If all exist
    if ((role != null) && (widget != null) && (permission != null))
    {
        // Remove role from widget
        WidgetRoleInfoProvider.RemoveRoleFromWidget(role.RoleID, widget.WidgetID,
permission.PermissionId);

        return true;
    }

    return false;
}
```

The following example sets the security level for a widget (allows it to be used by all authenticated users).

```
private bool SetSecurityLevel()
{
    // Get widget object
    WidgetInfo widget = WidgetInfoProvider.GetWidgetInfo("MyNewWidget");

    // If widget exists
    if (widget != null)
    {
        // Set security access type
        widget.AllowedFor = SecurityAccessEnum.AuthenticatedUsers;

        WidgetInfoProvider.SetWidgetInfo(widget);

        return true;
    }

    return false;
}
```

7.29 Wireframing

7.29.1 Overview

When creating or redesigning a website, you can plan out its structure, appearance, and functionality using wireframe schematics. Wireframes allow you to build approximate representations of pages directly in the site's content tree. CMS Desk users can view the wireframes and use them as a visual

guide while developing the website. Preparing wireframes is much easier and quicker than implementing real pages.

Topics:

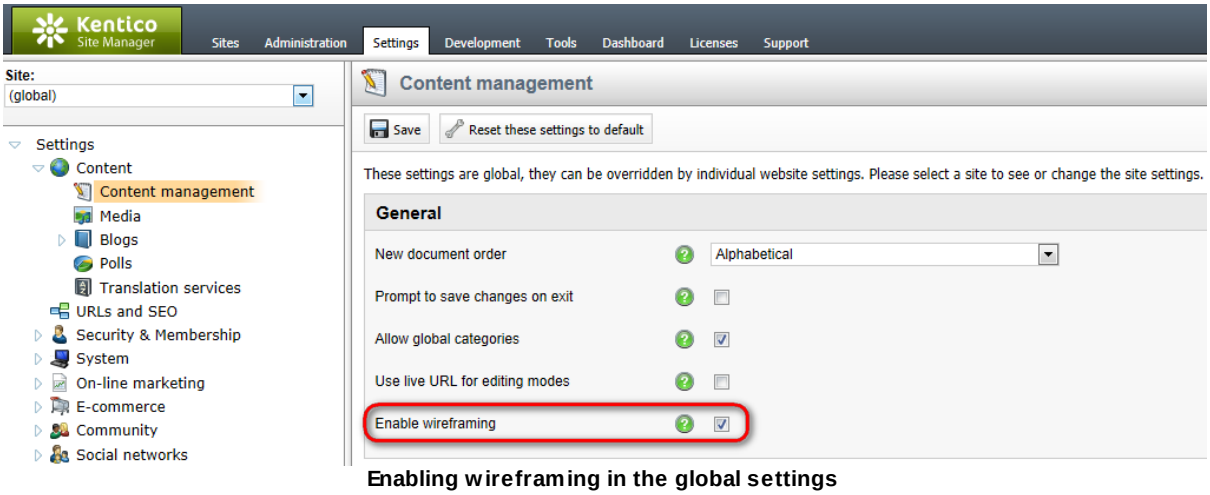
- Managing wireframes
- Setting up wireframe templates
- Configuring content inheritance
- Developing wireframe components
- Security

7.29.2 Managing wireframes

Users create, edit and view wireframes in the **CMS Desk -> Content** interface, directly in the content tree alongside the website's pages and other documents.

Enabling wireframes

Administrators can enable or disable wireframing for specific websites (or globally) through the **Enable wireframing** setting in **Site Manager -> Settings -> Content -> Content management**.



Enabling wireframing in the global settings

The system displays all existing wireframes even if this setting is disabled. However, users cannot make any modifications or create new wireframes.

Creating dedicated wireframe documents

The *Wireframe* (CMS.Wireframe) document type allows you to add wireframes into the website's content tree. The only purpose of wireframe documents is to provide visual guides in CMS Desk. They are not actual pages and the system does not include them in the website's navigation or sitemap. Visitors cannot access wireframe documents on the live site.

1. Add a new document in the CMS Desk content tree and select the **Wireframe** document type.

2. Type a name for the wireframe document into the **Page name** field.

3. Choose a page template for the wireframe. Click **Save**.

4. Design the content of the wireframe schematic on the document's **Wireframe** tab.

5. (Optional) Open the **Form** tab and configure the properties of the wireframe:

- **Comments** - allows you to type any required notes, such as a description of the wireframe or instructions for other users.
- **Exclude from search** - by default, wireframe documents are ignored by all forms of search (both the Smart search module and the SQL search). Uncheck the box to enable search for the wireframe.
- **Inherit content** - here you can set the wireframe's content inheritance options.

Users can now view the wireframe in the content tree and make further modifications if needed.

Combining wireframes with standard documents

The system allows you to integrate wireframes into documents of any other type. This can be useful in various scenarios, for example if you are creating an actual page (document) based on a wireframe or when using wireframes to plan a new design for existing pages.

- **Converting a dedicated wireframe to another document type**

  Warning: This process is irreversible. You cannot convert combined documents back to dedicated wireframe-only documents.

  1. Select the source wireframe document in the content tree and open the **Form** tab.
  2. Click **Convert to another document type** in the header actions.
  3. Select the target document type. The wireframe must be placed under a document that supports the new document type as a child. For example, pages cannot be placed under wireframe documents by default.
  4. Choose a template (for pages) or enter values into the form fields of the new document type.
  5. Click **Save**.

The document keeps the original wireframe definition on the **Wireframe** tab, but otherwise behaves exactly like a new document of the specified type. Users may refer to the wireframe at any time.

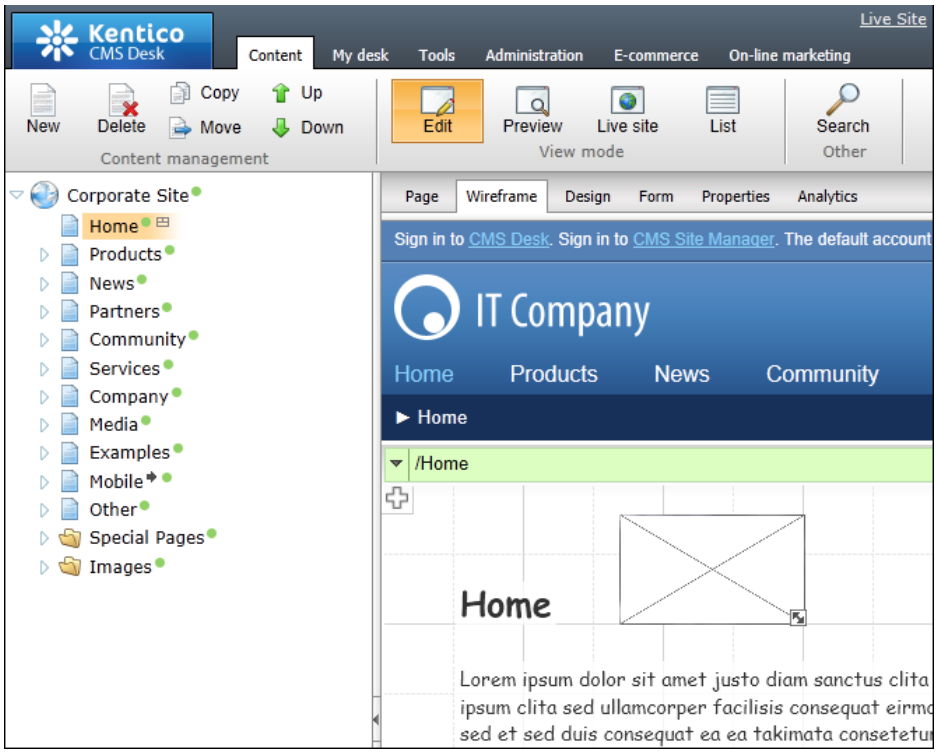
- **Inserting a wireframe into an existing document**

  1. Select the document in the content tree and use one of the following options:
     - Switch to the **Properties -> General** tab, scroll down to the **Advanced** section and click **Add wireframe to this page**.
     - Right-click the page template header on the **Design** tab and select **Add wireframe to this page** in the context menu.

2. Choose a template for the wireframe.
3. Click **Save**.

The document gains the **Wireframe** tab in addition to the standard content editing and configuration interface. Inserting a wireframe into a page does not affect its live site appearance in any way.

The result of these procedures is a combined document with both regular content and a wireframe.

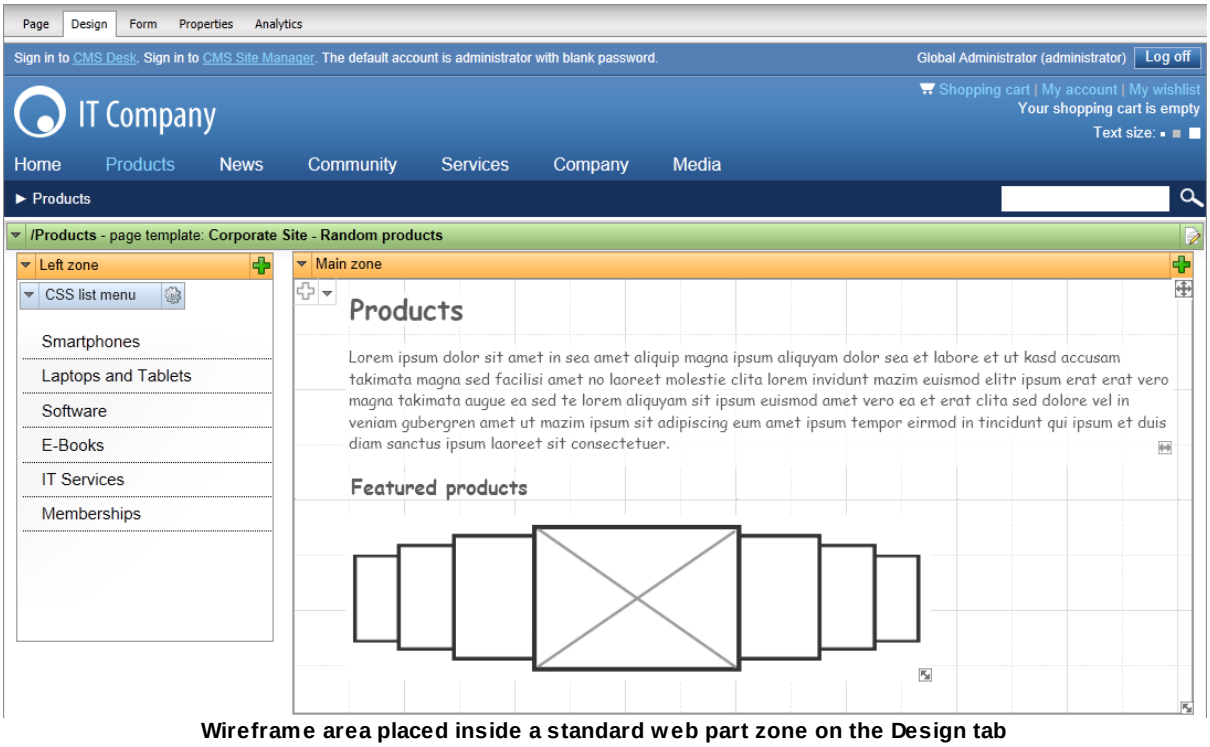


Wireframe inserted into the Home page

- You can manage the wireframe content of combined documents on the **Wireframe** tab, just like for dedicated wireframe documents.
- The system displays a status icon () next to documents that contain a wireframe. You can disable this icon in **Site Manager -> Settings -> Content -> Content management** via the **Display wireframe icon** setting.
- To configure the content inheritance settings of the document's wireframe or enter comments, go to **Properties -> Wireframe** instead of the **Form** tab.
- There are several ways to delete the document's wireframe if you no longer need it:
  - Open the **Properties -> General** tab, scroll down to the **Advanced** section and click **Remove wireframe**.
  - Right-click the page template header on the **Design** or **Wireframe** tab and select **Remove wireframe** in the context menu.
  - Click **Remove wireframe** on the **Properties -> Wireframe** tab.

Adding wireframes to the main page design

You can also place wireframes directly into the content of regular pages by adding the Wireframe area web part into a zone on the **Design** tab. This defines a sub-section of the page that works just like the *Wireframe* tab. You can use this approach to create wireframes for specific zones on pages that are already partially implemented.



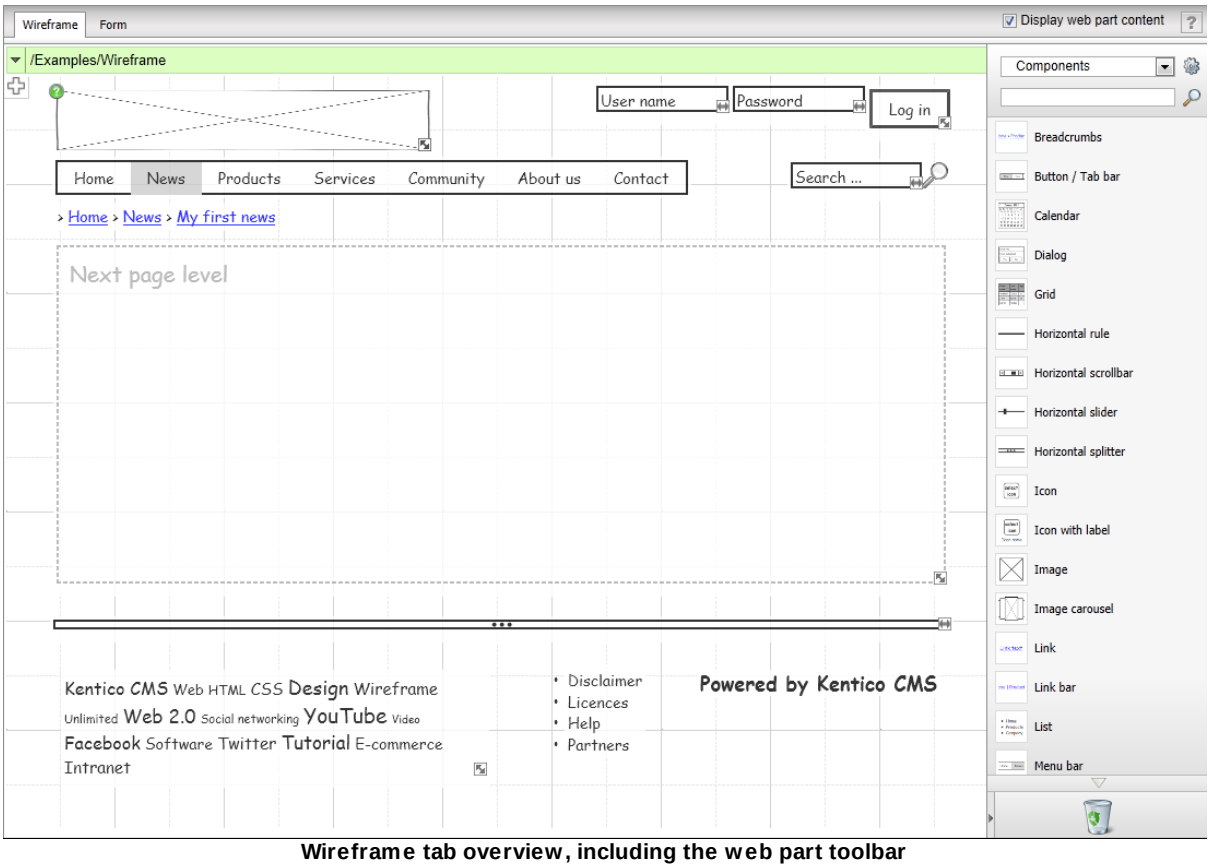
Wireframe area placed inside a standard web part zone on the Design tab

Such pages do not display their wireframe content on the live site, but users can view the wireframe components while editing the document in CMS Desk and in preview mode.

Defining wireframe content

To edit a wireframe, select the corresponding document in the CMS Desk content tree and open the **Wireframe** tab. The wireframe displays a grid, where you can place any number of components. Wireframe components are a special type of web parts. Kentico CMS contains a set of default wireframing components and you can also develop your own. When users add content to a wireframe, the system stores it on the assigned template.

Note: Wireframes do not support workflow. Dedicated wireframes always behave as unpublished documents.



Wireframe tab overview, including the web part toolbar

Adding components

You can add components (web parts) onto the wireframe by dragging items from the web part toolbar. To choose which components are offered in the toolbar, select a sub-category in the drop-down list. You can also filter the items by entering their name (or its part) into the search textbox (🔎). Alternatively, click the ✚ button in the corner of the wireframe zone and choose a specific component through the **Select web part** dialog.

Web part toolbar settings

Every user may configure the toolbar according to their preferences. There are four available positions and the toolbar can also be completely disabled.

To configure the toolbar settings, click the **Settings** (⚙) icon next to the category selector. The same options are also in **CMS Desk -> My Desk -> My profile -> Details**, so even users who have disabled their toolbar can change them.

Editing content

Double-click components to quickly modify their primary content. Depending on the type of the component, you can edit text, select an image, or specify a color. Some components have multiple editable sections.

Editing the options displayed by a menu component

When editing text values, you can use the following formatting syntax:

- **[a]**Text**[/a]** - displays the text as a standard hyperlink
- **[i]**Text**[/i]** - displays the text in italics
- **[b]**Text**[/b]** - makes the text bold
- **[u]**Text**[/u]** - underlines the text
- **[d]**Text**[/d]** - displays the text as a disabled element (grayed out)
- **{color:*Red*}**Text**{color}** - displays the text in red. You can apply any other CSS styles in format: *{cssProperty:Value}Text{cssProperty}*
- **\n** - inserts a line break (for values that have a single-line editing field)

Multiple formatting tags may be applied to the same text.

You can also insert various types of elements into the text by adding transformation expressions:

Element	Text expression	Output
Checkbox (checked)	**[x]**	☑
Checkbox (unchecked)	**[ ]**	☐
Radio button (selected)	**(o)**	◉
Radio button (unselected)	**( )**	○
Minus icon	**[-]**	⊟
Plus icon	**[+]**	⊞
Up arrow	**[^]**	▲
Down arrow	**[v]**	▼
Up/down icon (spinner)	**[^v]**	⇕
Left arrow	**[<]**	◀
Right arrow	**[>]**	▶
Directory icon	**[dir]**	
File icon	**[file]**	
Document icon	**[doc]**	
Indentation space	**[_]**	
	The number of underscore characters determines the indentation size.	

	For example: [____]

Moving components

You can relocate components by dragging them to any position in the grid. Some components with advanced functionality must be moved using a special drag icon ( ). If a component extends beyond the borders of the grid after being relocated, the wireframe increases the grid dimensions to fit the component. You can disable automatic zone resizing by holding down the CTRL key while dragging, which crops the component if necessary (i.e. hides the sections that do not fit into the zone).

Grouping wireframe components

To create a group of multiple components that you can move as a single unit:

1. Add the **Wireframe area** component from the **Layouts** sub-category.
2. Place the required components into the new sub-section of the grid.
3. Drag the Wireframe area to relocate the entire section.

Resizing items

You can dynamically change the dimensions of many types of wireframe components by dragging the arrow icon in their bottom right corner.

Advanced actions

Right-click components to access their context menu:

- **Configure** - allows you to set the component's web part properties. The available properties depend on the particular component. All wireframe web parts have the *Comments* property, which allows you to enter annotations or any other notes. The component displays the comment in the grid as the tooltip of an additional help icon ( ).
- **Bring forward** - moves the component to the foreground of the wireframe, in front of all other components that occupy the same space.
- **Send backward** - moves the component to the background, behind other components that share the same space.
- **Duplicate web part** - creates an exact copy of the component in the wireframe.

Removing components

There are two ways to delete components from the wireframe:

- Right-click the component and select **Remove** in the context menu.
- Drag the component into the "trash bin" area ( ) on the web part toolbar.

7.29.3 Setting up wireframe templates

The system stores the content of wireframes on page templates. Wireframe templates are standard portal engine page templates. Each template is defined by a layout, which contains one or more zones

representing separate wireframe grids. Individual zones save the content, positions, and configuration of the wireframe components that users place onto the grid.

When creating a new wireframe, users must always assign a template. There are two kinds of wireframe templates:

- **Ad-hoc** - every wireframe has its own unique ad-hoc page template. To start with a blank ad-hoc template when creating a new wireframe, choose the *Create a blank page* option in the template selection dialog. The system automatically deletes ad-hoc templates if their associated wireframe is removed.
- **Re-usable** - allow you to store predefined wireframe content. Individual wireframes cannot use these templates directly. When a user selects a re-usable template for a new wireframe, the system automatically creates an ad-hoc copy of the template for the given wireframe. This allows users to modify the wireframe without changing the default content of the re-usable template.

Note

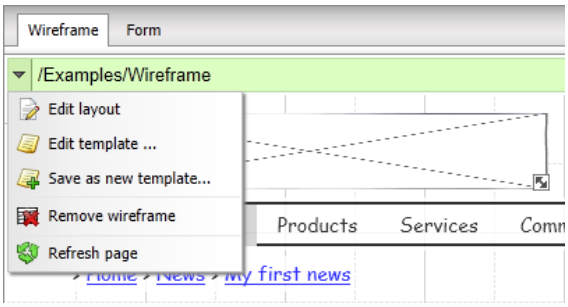
Combined documents with both standard page content and a wireframe have two separate templates assigned — one for the wireframe and the other for the actual page. These two templates are completely unrelated.

To manage the template of a wireframe:

1. Select the wireframe document in the CMS Desk content tree and open the **Wireframe** tab.

2. Right-click the green template header at the top of the wireframe grid and choose one of the template-related options in the context menu:

- **Edit layout** - allows you to modify the layout code of the wireframe's template.
- **Edit template** - opens a new window where you can configure the properties of the template.
- **Save as new template** - allows you to save the current template as a new re-usable page template, which users can then select when creating wireframes.



Page template context menu on the Wireframe tab

Creating a re-usable wireframe template

By preparing custom templates, you can store frequently used patterns of wireframe content. Users can then select an appropriate template as a starting point for new wireframes, instead of building each wireframe completely from scratch.

1. Go to **Site Manager -> Development -> Page templates**.

2. Select the **Wireframes** category and click **New template** above the catalog tree.

3. Enter a name and description for the template and click **Save**.

The template's **General** tab opens:

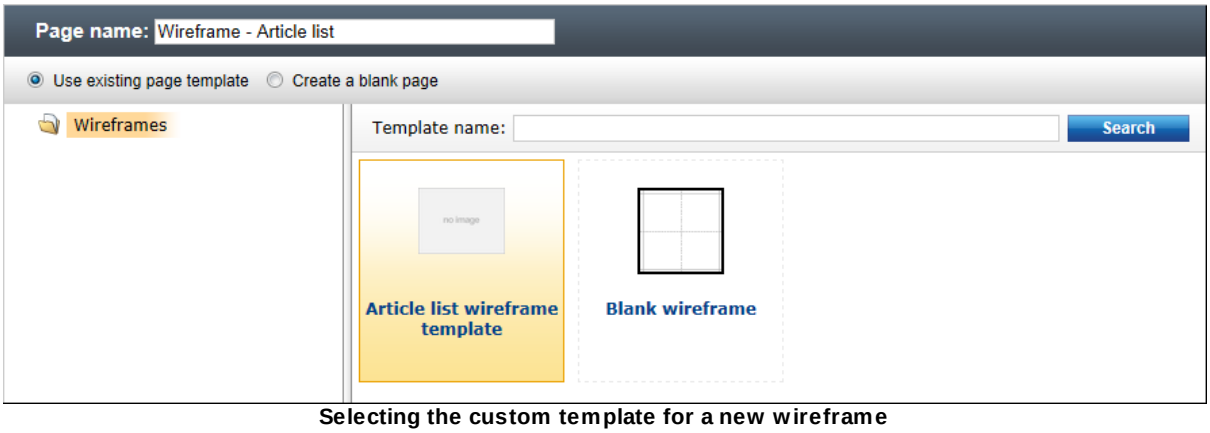
- The **Template type** must be set to *Portal page* for wireframe templates. This is the default option for new templates.
- The **Clone as ad-hoc for new documents** checkbox does not affect the behavior of wireframe templates. The system always clones templates as ad-hoc when creating wireframes, even if this setting is disabled for the assigned template.

4. Switch to the **Layout** tab and enter the template's layout code. Include at least one wireframe zone.

5. Open the **Design** tab and define the template's wireframe content.

6. On the **Sites** tab, add all websites for which you wish to make the template available.

Users can now select the new template from the catalog when creating wireframes on the specified sites.



Selecting the custom template for a new wireframe

Changing the wireframe template default category

By default, all wireframes must use templates from the *Wireframes* category. To allow templates from another category:

1. Go to **Site Manager -> Development -> Document types**.
2. Edit the **Wireframe** (CMS.Wireframe) document type.
3. On the **General** tab, select a different category in the **Root page template category** property.
4. Click **Save**.

When creating wireframes, users can now select templates from the specified category and its sub-categories.

Wireframe template layouts

You can create layouts for wireframe templates in the same way as standard portal page layouts. The only difference is that you need to define the web part zones in the layout as wireframe zones. You can do this by including an additional attribute according to the selected **Layout type**.

- With **ASCX** type layouts, zones are inserted as controls. To designate a zone as a wireframe grid, add the *WireFrame="true"* property to the corresponding control:

```
<cms:CMSWebPartZone runat="server" ZoneID="zoneA" Wireframe="true" />
```

- In the case of **HTML** layouts, web part zones are represented by special macro expressions. Wireframe zones must have the *(Wireframe)true* parameter in addition to their *(id)*:

```
{^ WebPartZone|(id)zoneA|(Wireframe)true ^}
```

You can combine standard web part zones and wireframes zones within a single template layout.

7.29.4 Configuring content inheritance

Like regular pages, wireframes provide support for master pages and visual inheritance. This allows them to display content from parent documents in addition to their own specific components. Content inheritance offers several advantages, such as viewing wireframes within the exact context of the website and being able to manage shared wireframe content in a single location.

To configure the inheritance preferences for dedicated *Wireframe* documents:

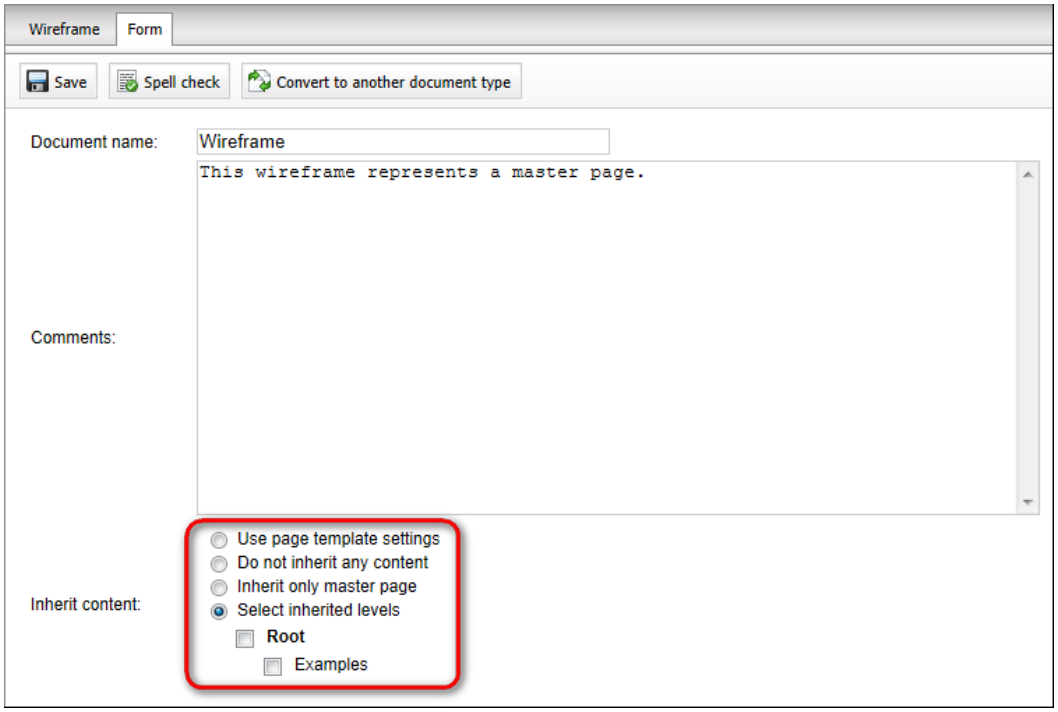
1. Select the wireframe document in the content tree in CMS Desk.

2. Open the **Form** tab.

3. Select one of the options in the **Inherit content** section.

Use page template settings	The content inheritance is determined by the settings of the wireframe's template. To manage the template's configuration: 1. Switch to the **Wireframe** tab. 2. Right-click the green template header at the top of the wireframe grid and select **Edit template**. 3. Set the **Inherit content** property on the **General** tab of the **Page template properties** dialog.
Do not inherit any content	The wireframe does not inherit any content from parent documents in the content tree.
Inherit only master page	The wireframe inherits content from the first master page above the

	document in the content tree. If there are multiple master pages, the wireframe only inherits from the closest one in the hierarchy.
Select inherited levels	You can select exactly from which parent documents the wireframe inherits via the checkboxes below.



Wireframe inheritance options

4. Click **Save**.

Wireframe inheritance for combined documents

For combined documents that have both standard content and a wireframe definition, you can find the inheritance settings on the **Properties -> Wireframe** tab instead.

The wireframe inheritance is completely separate from the content inheritance configured for the actual page (on the *Properties -> Template* tab). They do not affect each other in any way and you can set them up differently.

There are two possible types of inheritance for wireframes:

- Nesting within parent wireframes
- Loading content from portal engine pages through standard visual inheritance

Wireframe nesting

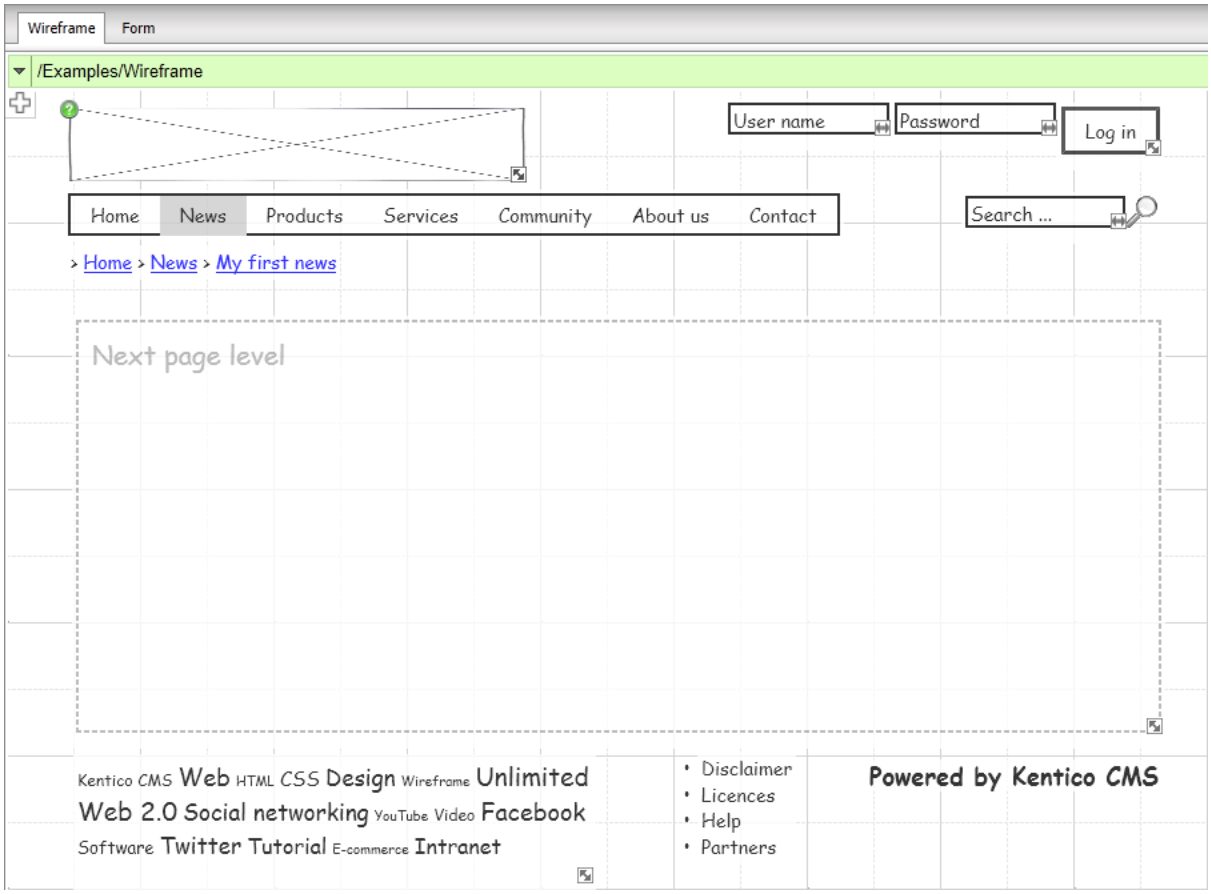
In most cases, real websites utilize some form of content inheritance. Wireframes can accurately represent pages with inheritance by loading content from parent wireframes in the content tree. The

system then displays the child wireframe inside a designated area within the content of the parent.

To set up a wireframe inheritance scenario on your website:

1. Create the wireframe representing the parent page. Inheritance works for both dedicated *Wireframe* documents and wireframes inserted into a different document type.

2. Add the **Next page level** component from the *Wireframes/Layouts* web part category to the parent wireframe.



Example of the Next page level component on a parent wireframe

The **Next page level** component must always be placed on parent wireframes. It defines an area inside the grid that serves as a placeholder for the content of child wireframes. You can resize the area to any required dimensions.

3. Create any number of wireframe documents under the parent. The child wireframes do not need to be located directly on the level below the parent. You can skip levels when inheriting content.

4. Open the **Form** (or **Properties -> Wireframe**) tab of the child wireframe documents and configure the **Inherit content** preferences to include the required parents.

Users can customize the child wireframes only in the exact area defined by the **Next page level** component. The wireframe displays the remaining components from the parent around this area as fixed content.

Child wireframe with inherited content

Inheriting content from standard pages

The second type of inheritance allows wireframes to load web content from parent portal engine pages. By displaying wireframes within the actual pages of the current website, you can create accurate visual representations.

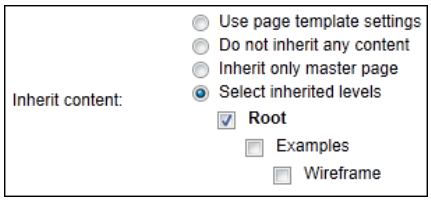
Requirements:

- The parent cannot have a wireframe definition of its own, because wireframe-to-wireframe inheritance has a higher priority.
- The Page placeholder web part must be placed on the parent page, otherwise the system is not able to display the content of child documents correctly (including wireframes).

Example:

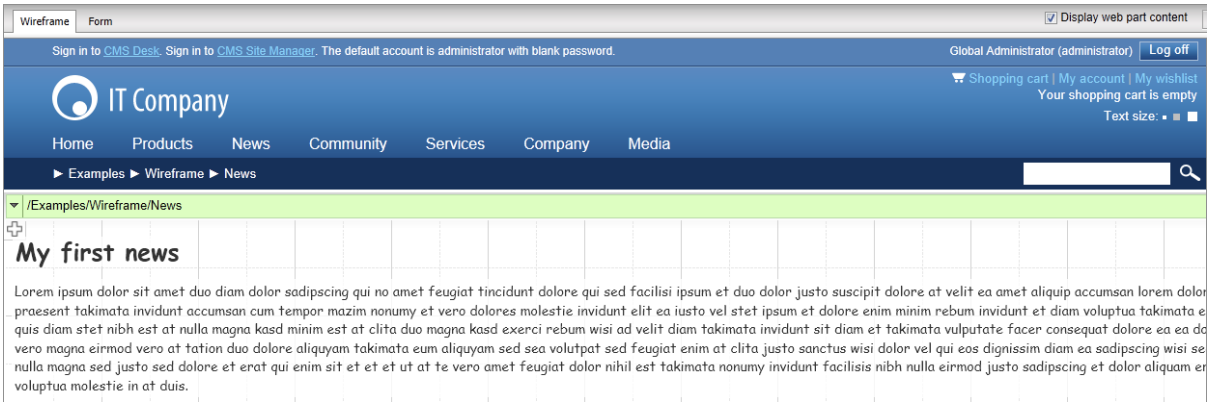
1. Open the sample Corporate Site in CMS Desk and select the **Examples/Wireframe/News** document in the content tree.

2. Switch to the **Form** tab, choose **Select inherited levels** in the **Inherit content** section and make sure that only the **Root** document is checked.



3. Click **Save**.

When viewed on the **Wireframe** tab, the document inherits the header and footer content defined on the website's master page.



Wireframe inheriting web content from the master page

You may use both types of inheritance for wireframes at the same time. If you set the document to also inherit from its direct parent, it loads the wireframe content and shows it inside the master page.

7.29.5 Developing wireframe components

All wireframe components are defined as web parts of a specific type. The system allows you to develop new web parts and modify existing ones, so you can create any custom wireframing tools that your website requires. For more Information about web part development in general, refer to Development -> Web parts -> Developing web parts.

Example

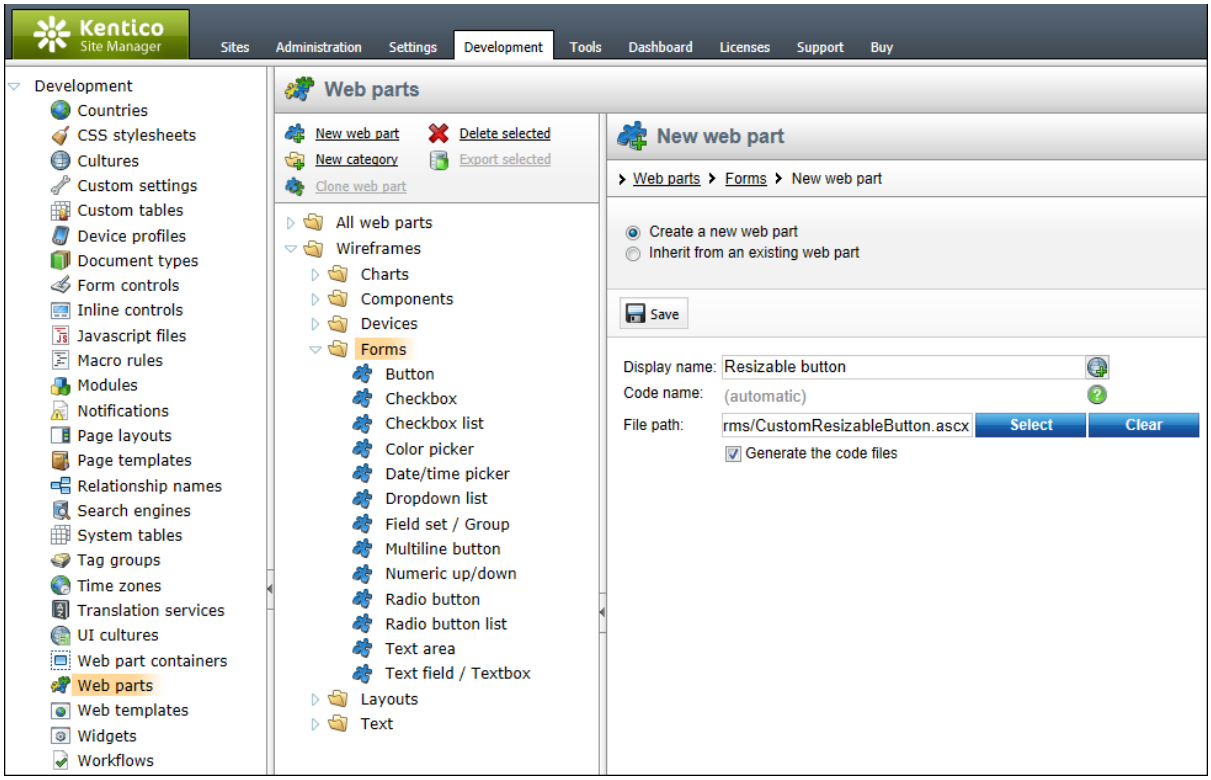
This example demonstrates how to create a custom wireframe component representing a button with a resizable width. You can develop more complex components by following the same basic principles. For additional inspiration, you can inspect the code of the default wireframe web parts found in the **~/CMSWebParts/Wireframes** folder of your web project.

Registering a new wireframe web part

1. Go to **Site Manager -> Development -> Web parts**, expand the **Wireframe** category at the bottom of the catalog tree and select the **Forms** sub-category.

2. Click **New web part** above the tree. In the **New web part** dialog, choose **Create a new web part** and enter the following values:

- **Display name**: Resizable button
- **File path**: ~/CMSWebParts/Wireframe/Forms/CustomResizableButton.ascx
- **Generate the code files**: yes (checked)



Creating a new web part

Click **Save**.

3. The new web part's **General** tab opens. Set the following options:

- **Type**: *Wireframe*; all web parts that you wish to use as wireframe components must use this type to function correctly.
- **Skip initial configuration**: *Yes (checked)*; allows users to add the web part to wireframe grids without having to configure its properties.

Click **Save**.

4. Switch to the **Properties** tab and add (+) a single property:

- **Column name**: Text
- **Attribute type**: Text
- **Attribute size**: 100
- **Default value**: OK
- **Field caption**: Button caption
- **Form control**: Text box

The purpose of the *Text* property is to store the value of the button's text caption. Click **Save**.

5. Open the **CSS** tab and add the following class definition:

```
.WireframeResizableButton
{
  border: solid 3px #000;
  padding: 5px 15px 5px 15px;
  display: block;
  overflow: hidden;
  text-align: center;
  min-width: 50px;
}
```

The *WireframeResizableButton* class will be used to style the button component. Click **Save**.

Implementing the component's functionality

6. Open the web project in Visual Studio and navigate to the **CMSWebParts/Wireframe/Forms** folder. It contains the *CustomResizableButton.ascx* file and its *ascx.cs* code behind, which were generated by the system when you created the web part.

7. Edit *CustomResizableButton.ascx* and add the following controls into the markup:

```
<cms:EditableWebPartProperty runat="server" id="txtElem"
CssClass="WireframeResizableButton" PropertyName="Text" Type="TextBox" />
<cms:WebPartResizer runat="server" id="resElem" HorizontalOnly="true"
RenderEnvelope="true" />
```

These are two instances of server controls from the *CMS.PortalControls* namespace that display content and provide the wireframe editing functionality.

Control	Description
EditableWebPartProperty	Renders a text value in a way similar to a standard literal control. Users can edit the value by double-clicking the text in the wireframe. • The **PropertyName** specifies the name of the web part property that is affected by the editing. Notice that the value matches the *Column name* of the web part's only property (Text). • Through the **Type** property, you can choose what kind of editing interface the control displays. The appearance of the control in the wireframe grid is typically defined using CSS classes, such as the *WireframeResizableButton* class in the current example.
WebPartResizer	Adds a resizer that allows users to easily change the dimensions of the component directly in the wireframe. The **HorizontalOnly** property is set to *true* in the example, so the component only supports width resizing in this case.

Other available control options:

EditableWebPartImage	Displays image content. Users can select the image through a dialog by double-clicking the current output.
EditableWebPartList	Represents a list of items formatted in a certain way, such as a menu or table. Users enter the value as text, with each item separated by a new line.
EditableWebPartColor	Displays an area in a specified color. Users can double-click the component to select the color.

8. Edit the code behind file. Because it was generated specifically for a web part, the control already inherits from the appropriate base class and also contains two default regions and several basic method definitions.

9. Add a **Text** property into the *Properties* region, which handles the corresponding property of the web part object (defined in step 4). The overall code of the class should now match the following:

[C#]

```
public partial class CMSWebParts_Wireframe_Forms_CustomResizableButton :
CMSAbstractWebPart
{
    #region "Properties"

    /// <summary>
    /// Accesses the value of the web part's Text property.
    /// </summary>
    public string Text
    {
        get
        {
            return ValidationHelper.GetString(this.GetValue("Text"), "");
        }
        set
        {
            this.SetValue("Text", value);
        }
    }

    #endregion

    #region "Methods"

    /// <summary>
    /// Content loaded event handler.
    /// </summary>
    public override void OnContentLoaded()
    {
        base.OnContentLoaded();
        SetupControl();
    }
```

```
    /// <summary>
    /// Initializes the control properties.
    /// </summary>
    protected void SetupControl()
    {
        if (this.StopProcessing)
        {
            // Do not process.
        }
        else
        {

        }
    }

    /// <summary>
    /// Reloads the control data.
    /// </summary>
    public override void ReloadData()
    {
        base.ReloadData();

        SetupControl();
    }

    #endregion
}
```

10. Implement the **SetupControl** method:

[C#]

```
/// <summary>
/// Initializes the control properties.
/// </summary>
protected void SetupControl()
{
    if (StopProcessing)
    {
        // Do not process.
    }
    else
    {
        // Loads and assigns the button caption.
        txtElem.Text = Text;

        // Sets up width resizing.
        string width = WebPartWidth;
        if (!String.IsNullOrEmpty(width))
        {
            txtElem.Style += String.Format("width: {0};", width);
        }
```

```
        // Connects the button control to the resizer.
        resElem.ResizedElementID = txtElem.ClientID;
    }
}
```

This method ensures that the web part displays the component correctly according to its current configuration.

- First, it checks that processing (visibility) is enabled for the web part.
- If true, the method loads the button's caption text from the **Text** property and assigns it to the **EditableWebPartProperty** control.
- The method retrieves the width value of the button via the **WebPartWidth** property (inherited from the *CMSAbstractWebPart* parent class) and adds it to the button's styles.
- The method binds the resizer to the button control through its *ClientID*.

Handling the width and height of wireframe components

In this example, the width property is not included in the configuration dialog of the web part and is handled exclusively through the resizer.

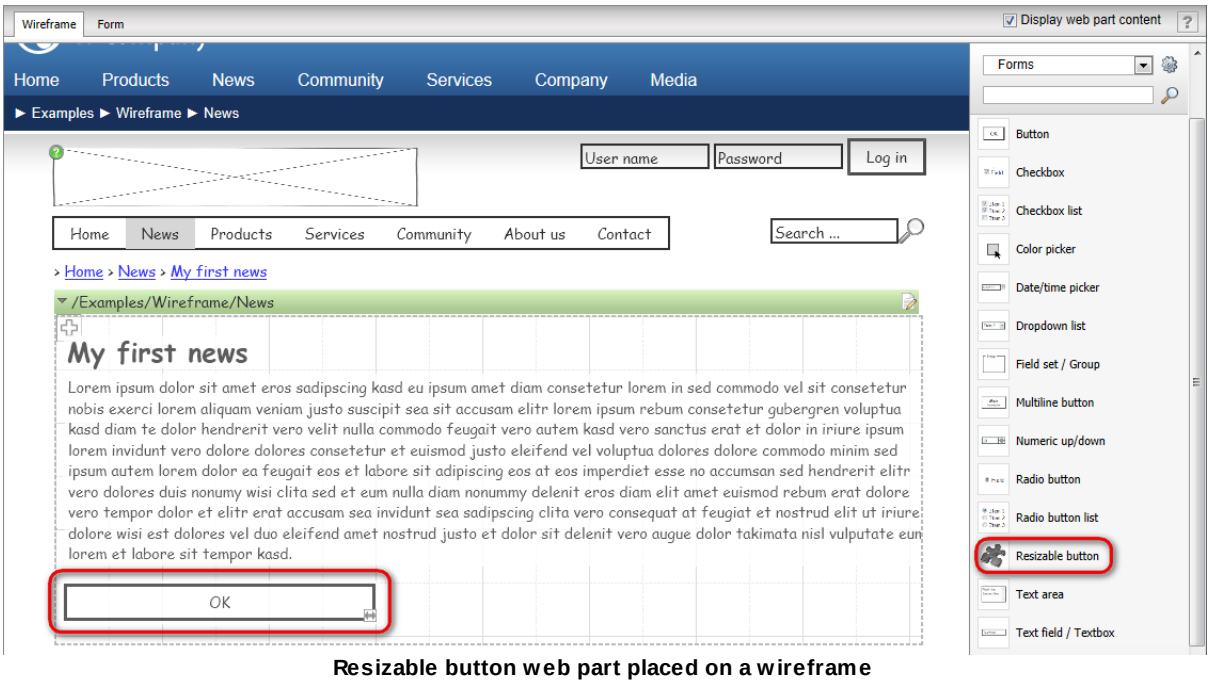
To register the width as a standard configuration option, define a property for the web part with *WebPartWidth* as its column name.

When creating components that also support vertical resizing, the **WebPartHeight** property (also inherited from the parent class) can be handled in the same way.

11. Save both files. If your Kentico CMS project was installed in the web application format, you must also **Build** the project. The wireframe component is now fully functional and ready to be used.

Result

Add the **Resizable button** component to any wireframe on the website (e.g. the *Examples -> Wireframe* document on the sample Corporate Site). You can dynamically change the width of the button by dragging the arrow icon in its bottom right corner. Double-clicking the button's caption allows you to edit the text.



Resizable button web part placed on a wireframe

7.29.6 Security

Only editors and other users with access to the CMS Desk interface can view and manage wireframes. The system does not display wireframes or wireframing components on the live website.

Permissions

To configure wireframing permissions for specific roles, go to **Site Manager** / **CMS Desk -> Administration -> Permissions**.

Permissions for	Permission	Description
Module -> Design	Wireframing	Allows users to edit the content of wireframes on the **Wireframe** tab and create new wireframe documents. Users without this permission can view existing wireframes, but are not allowed to make any modifications.
	Design web site	Allows users to edit documents on the **Design** tab of **CMS Desk**. Users only need this permission for managing wireframe zones inside standard page content.
Module -> Content	-	Users need the permissions for the Content module to access the CMS Desk content tree and read or create documents in general (including wireframes).
Document type -> Wireframe	-	You can assign the basic content permissions specifically for the *Wireframe* document type (instead of the entire *Content* module).

UI personalization

By configuring UI personalization for particular roles, you can customize the visibility of individual elements in the wireframing interface.

1. Go to **Site Manager** / **CMS Desk -> Administration -> UI personalization**.
2. Select the appropriate **Site** and **Role**.
3. Choose the *Content* **Module**.
4. Configure the checkboxes as required:

- **Wireframe**
- **Design**
  - **Edit layout**, **Edit template**, **Save as new template** - sets the visibility of the management actions offered in the page template context menu.
  - **Web part properties -> General** - users must have this UI element allowed to access the property configuration dialog of wireframe components.
  - **Add web parts** - sets the visibility of the add button () in the corner of the wireframe grid. Users can insert components from the toolbar even without this element.
  - **Remove web parts** - context menu option for deleting individual components from wireframes.
- **Form**
- **Properties -> Wireframe**
- **New -> Page template selection -> Use existing page template** - required to create new wireframe documents.

Members of the selected role can now see only the specified UI elements.

Part VIII

Modules

8 Modules

8.1 Overview

This section provides reference on in-box modules in Kentico CMS. Information on the particular modules and their API examples can be found in sub-categories of the section. General information on the whole API examples feature can be found in the API examples chapter in the **API programming and Kentico CMS internals** section of this guide.

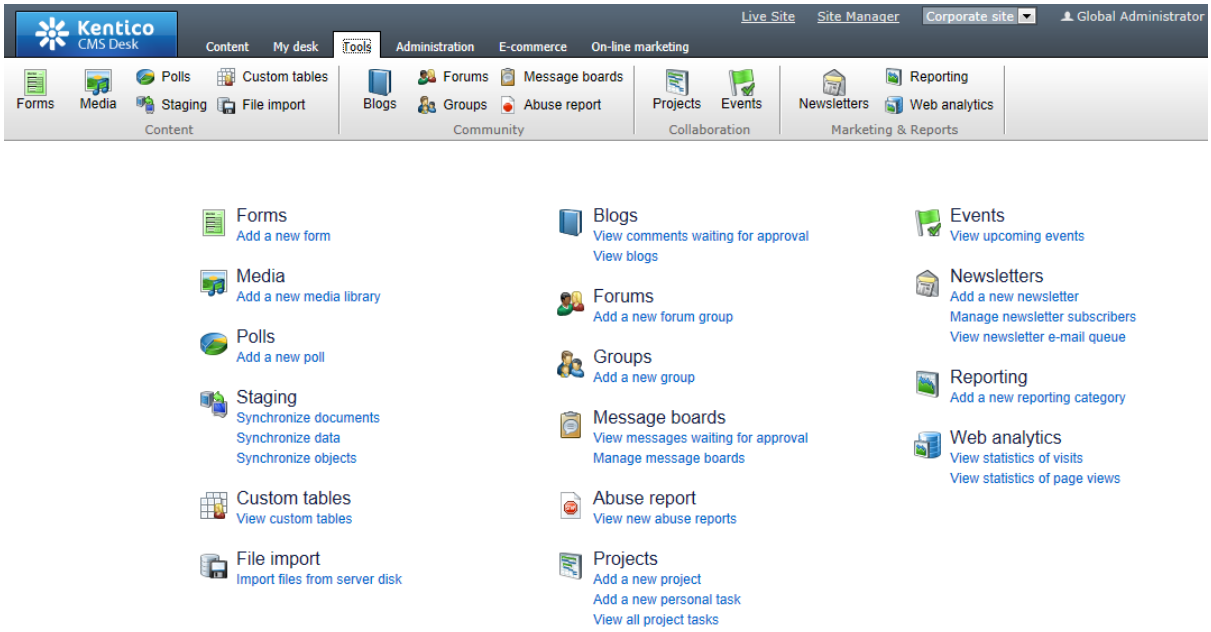
This topic describes where you can easily access Kentico CMS modules and how to customize the user interface.
In this example, you can learn how to develop a simple module and integrate it into the Kentico CMS user interface.

Please note: Not all modules might be available in your Kentico CMS installation because some modules are available only with certain licenses.

8.2 Accessing modules

Tools user interface

CMS Desk -> Tools is where you can easily access Kentico CMS modules. The interface consists of a ribbon and panel menu and contains links to these modules. Besides, the panel menu provides shortcuts to selected functionality of the modules. The UI is fully customizable, enabling you to access both built-in and custom modules:



Customizing the Tools user interface

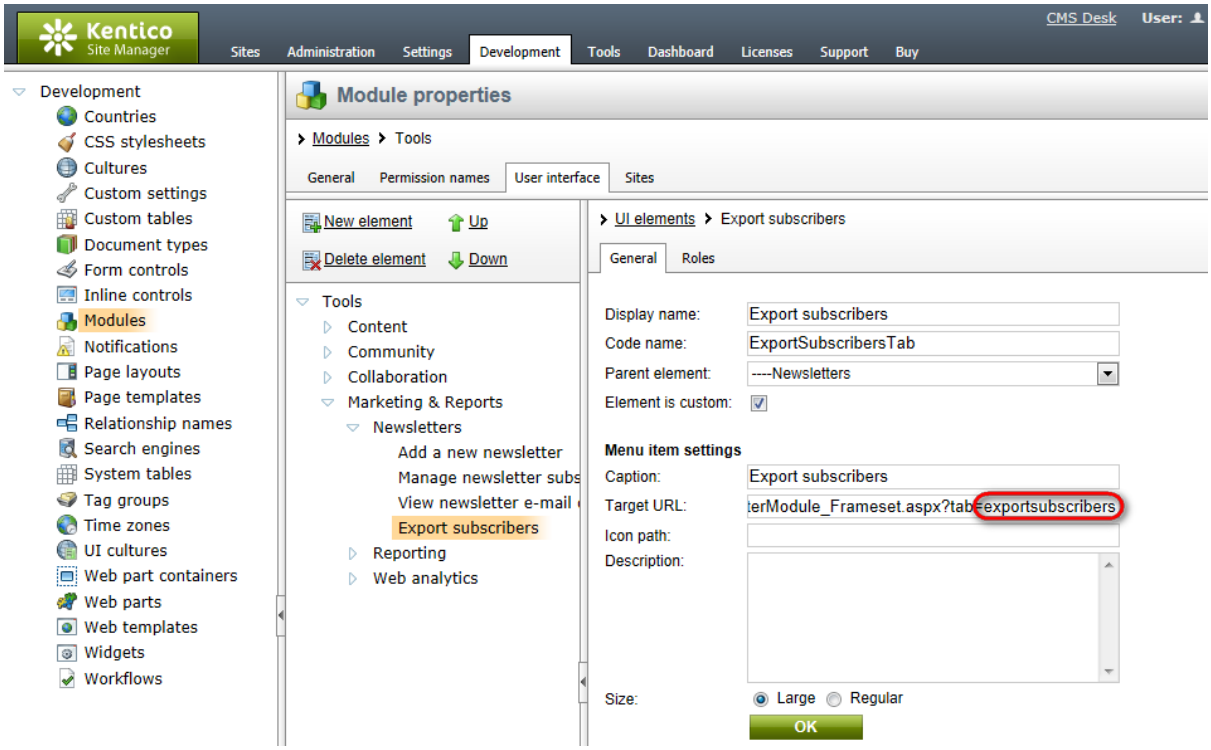
How to do it in general

To customize this user interface, you need to choose a module and for its selected functionality add a link to the panel menu. Specifically, you will create a new UI element under the chosen module, representing a shortcut to the tab where the desired functionality is available.

Example

1. Sign in to **Site Manager -> Developement -> Modules -> Edit** (✎) **Tools** and switch to the **User interface** tab.

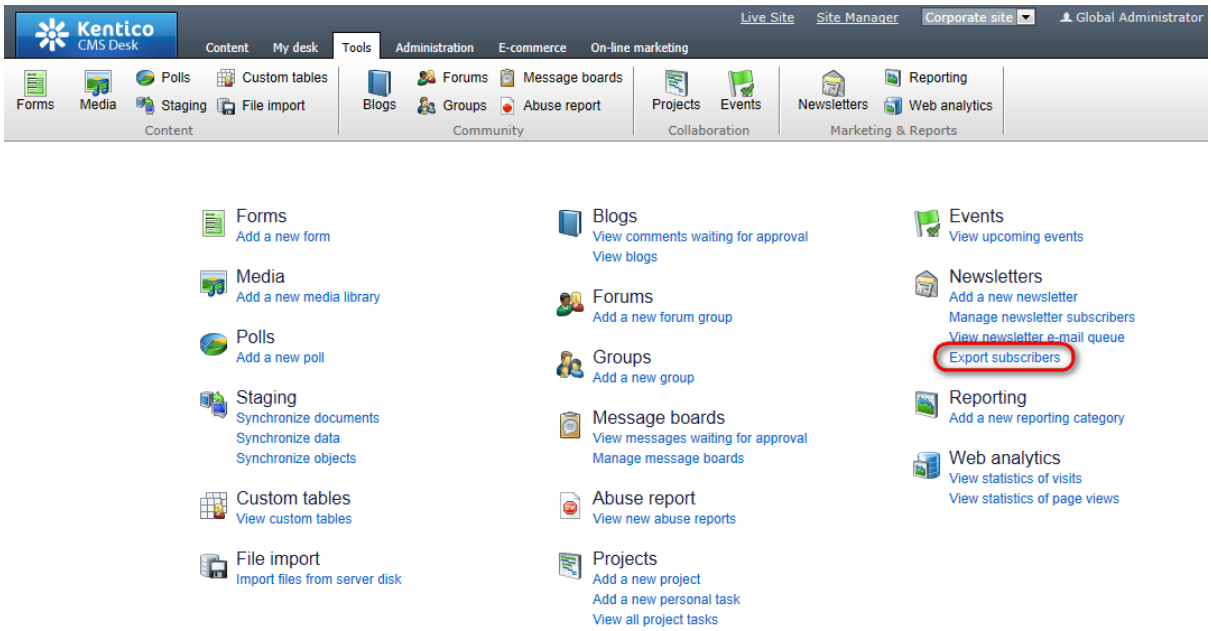
2. In the Tools UI elements tree, navigate to the module of your interest, click **New element** (📄) and proceed as described in the UI personalization chapter in the Membership, permissions and security section of this guide.



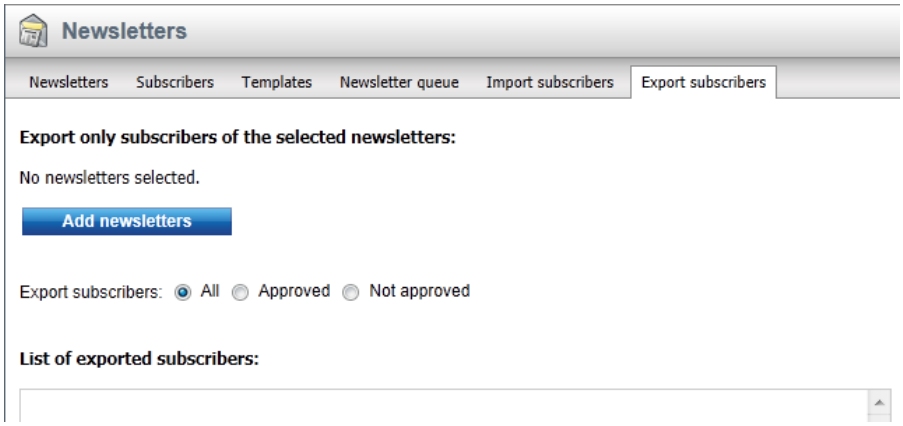
> **Please note**
>
> The value of the **Target URL** property must contain a *tab* query parameter whose value is the code name of the tab to which you want to link.
> The code name of the tab is the same as the code name of the particular UI element.

3. Now you may want to manage permissions to see the new UI element. If you decide to do so, switch to the **Roles** tab and grant selected roles with permission to see the given UI element. For more details, please refer again to the UI personalization chapter.

4. Finally, go to **CMS Desk -> Tools**. As you can see, the link you have just created is available among other links of the given module.



If you now click the link, you will be redirected to a tab containing the given functionality.



Please note

To add a new custom module to the Tools user interface, you will need to take the same steps as described in this example. However, you will need to add the new UI element directly under the **Tools** section. More details can be found in the Developing custom modules topic.

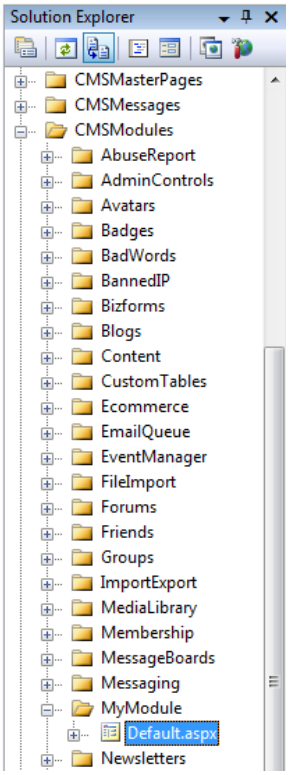
8.3 Developing custom modules

Here you will learn how to create a simple module with a single button that displays the current date and time. The procedure can be used for adding any kind of custom module that you develop. You will also learn how to control access to the module's functions using permissions and integration of this module into the UI using UI personalization.

1. Open Kentico CMS web project in Visual Studio. You can do that either using the *WebProject.sln* file in your website root or using the **File -> Open -> website** menu in Visual Studio.

2. Create a new folder *MyModule* under the *CMSModules* folder.

3. Create a new page *default.aspx* under the *CMSModules/MyModule* folder:



4. Switch to code behind of the page and change the following line:

[C#]

```
public partial class CMSModules_MyModule_Default : System.Web.UI.Page
```

to:

[C#]

```
public partial class CMSModules_MyModule_Default : CMSToolsPage
```

It ensures that the module can be used only by users with access to Kentico CMS Desk.

5. Add the following code at the beginning of the page:

[C#]

```
using CMS.CMSHelper;
using CMS.UIControls;
```

6. Add the following code to the *Page_Load* method:

[C#]

```
if (!CMSContext.CurrentUser.IsAuthorizedPerResource("cms.mymodule", "read"))
{
    RedirectToAccessDenied("cms.mymodule", "Read");
}
```

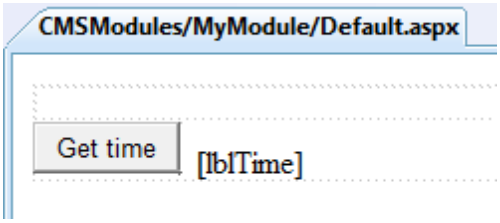
It checks if the current user has the **Read** permission for the *myprojects.mymodule* module.

7. Switch to the **Design** mode and add a **Button** control on the page. Set its properties:

- **ID**: btnGetTime
- **Text**: Get time

8. Add a **Label** control on the page, next to the button. Set its properties:

- **ID**: lblTime
- **Text**: (clear the value)



9. Double-click the button and add the following code inside the **btnGetTime_Click** method:

[C#]

```
if (CMSContext.CurrentUser.IsAuthorizedPerResource("cms.mymodule", "gettime"))
{
    lblTime.Text = DateTime.Now.ToString();
}
else
{
```

```
    lblTime.Text = "You're not authorized to get the current date and time.";
}
```

It checks if the current user has the *gettime* permission for the *cms.mymodule* module and if so, it displays the current date and time.

10. Run the project and sign in to **Site Manager**. Go to **Development -> Modules** and click **New module**. Enter the following values:

- **Module display name**: My module
- **Module code name**: cms.mymodule; for the UI element to be site-related, it is important to enter the value in the *cms.<elementname>* format, where *<elementname>* is the code name of the UI element created in step 16; see this topic for more details.

Click **OK**. The module was registered in the system.

Please note: In the system, modules are represented by *ResourceInfo* and *ResourceInfoProvider*.

11. Go to the **Permission names** tab and click **New permission**. Enter the following values:

- **Permission display name**: Read
- **Permission code name**: read
- **Display in matrix**: enable

Click **OK**.

12. Add another permission:

- **Permission display name**: Get time
- **Permission code name**: gettime
- **Display in matrix**: enabled

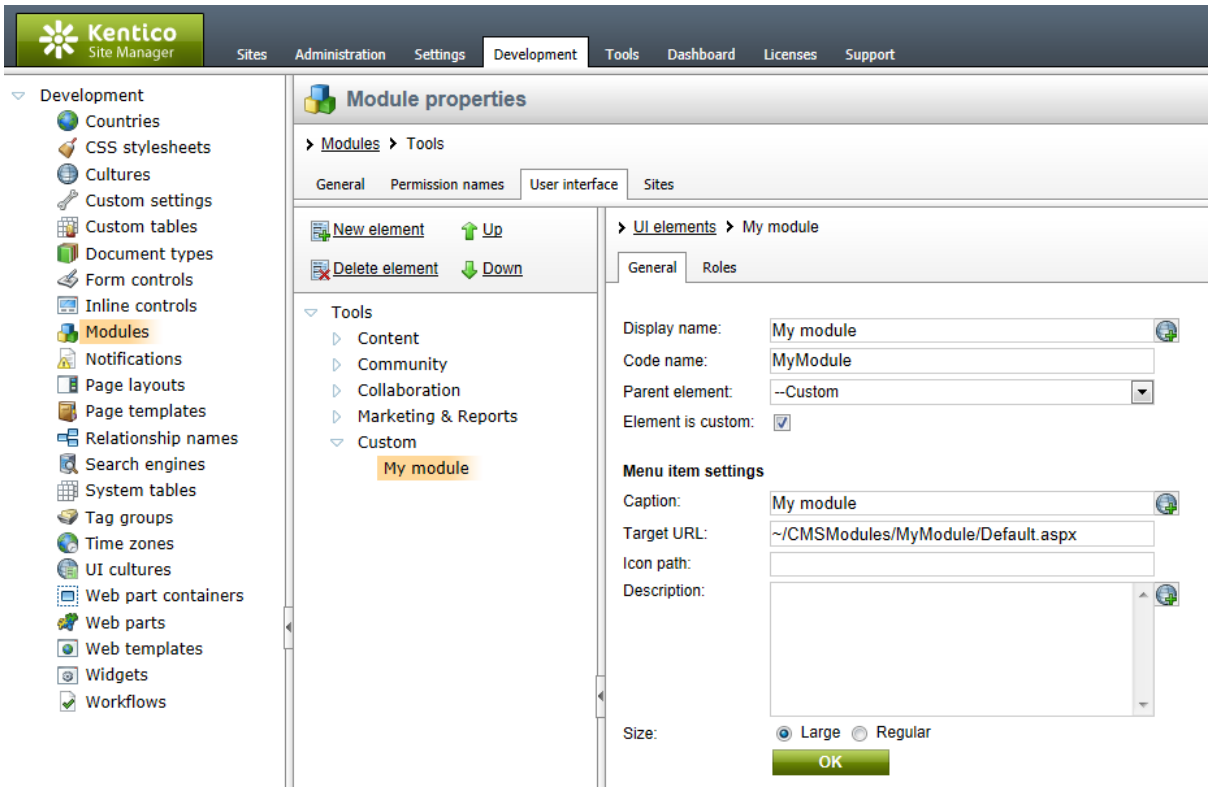
Click **OK**.

13. Go to the **Sites** tab and enable the new module for appropriate sites. Now that the module is registered, you need to display it in the user interface. For the purposes of this example, we will place it into the Tools menu. However, the module can also be placed into other sections of the UI as listed here.

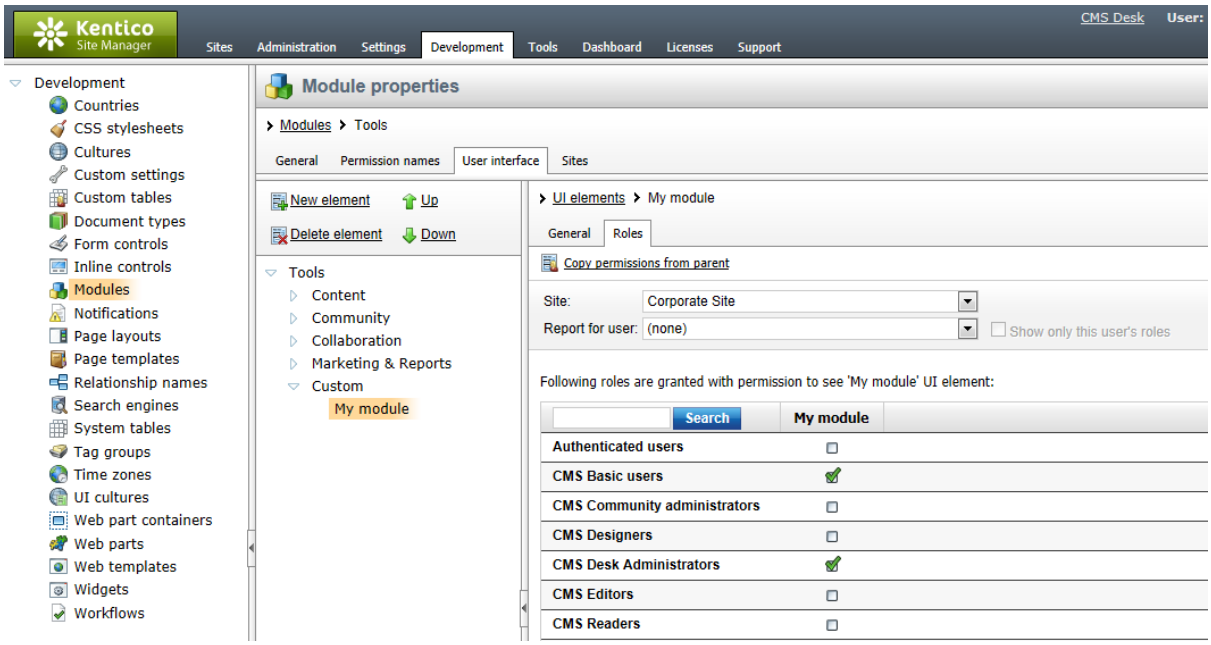
14. Go back to **Development -> Modules** and **Edit** the **Tools** module. Switch to its **User interface** tab, select **Custom** in the Tools tree and click **New element**. Enter the following details:

- **Display name**: My module
- **Code name**: MyModule
- **Element is custom**: enabled
- **Caption**: My module
- **Target URL**: ~/CMSModules/MyModule/Default.aspx

Click **OK**.

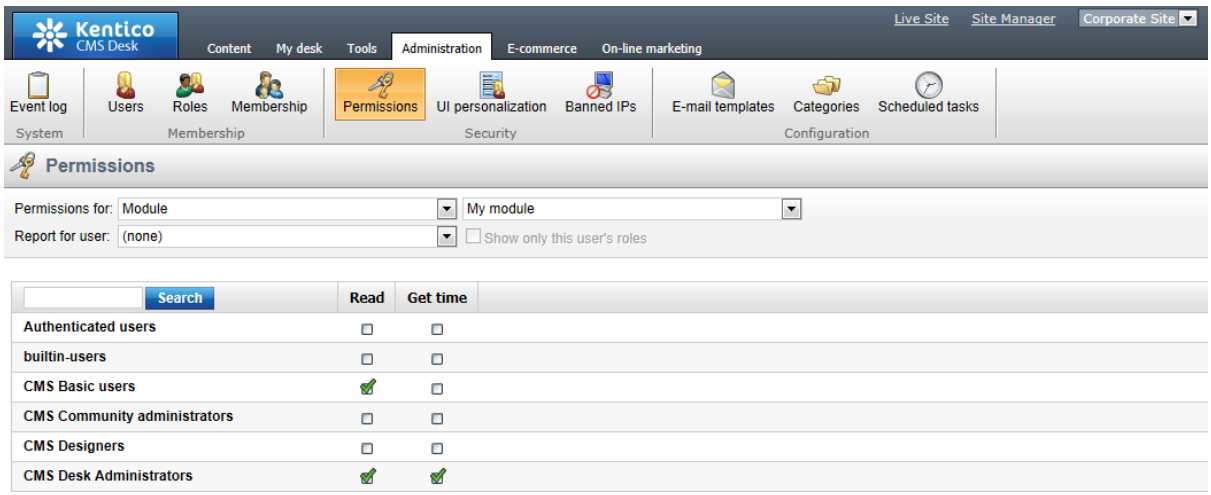


15. We will use UI personalization to enable access to this module only to members of certain roles. For this to work, UI personalization needs to be enabled. With the new UI element still selected, switch to the **Roles** tab and enable the element for members of the **CMS Basic users** and **CMS Desk Administrators** roles.

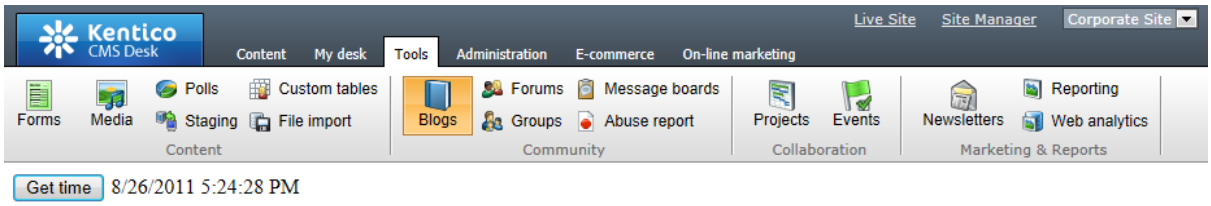


16. Go to **CMS Desk -> Administration -> Permissions** and choose **Permission for**: *Module* / *My*

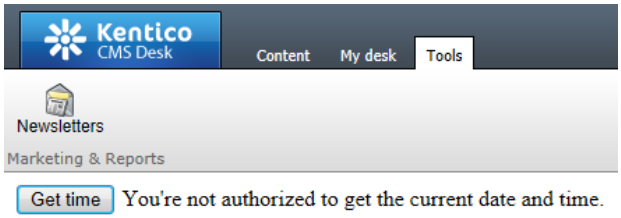
module. Grant the *Read* permission to the **CMS Basic users** role and both the *Read* and *Get time* permissions to the **CMS Desk Administrators** role:



17. Now go to **CMS Desk -> Tools** logged in as *administrator* (with blank password). As you can see, the **My module** item is now available in the menu. Click it and you will see your form created in steps 3-9. When you click the **Get time** button, the current date and time is displayed.



18. Finally, try logging in as *Andy* (with blank password) and go back to **CMS Desk -> Tools**. Andy is a member of the **CMS Basic users** role. In step 16, you configured that members of this role have only the *Read* permission necessary for the module to be displayed (ensured in step 6), but not the *Get time* permission necessary for the button functionality. Try clicking it, you will see the access denied message that you defined in step 9.



Exporting custom modules

The following folders will be included in the export package when the custom module is selected for export. It is therefore recommended that your modules data is stored within these folders:

- ~/App_Code/CMSModules/<module_name>
  (or *~/Old_App_Code/...* if you installed Kentico CMS as a web application)
- ~/App_Data/CMSModules/<module_name>
- ~/CMSModules/<module_name>

The <module_name> value needs to be the same as the code name of the module in the administration interface, so for example for a module named *CMS.Test*, the folders would be:

- ~/App_Code/CMSModules/CMS_Test
- ~/App_Data/CMSModules/CMS_Test
- ~/CMSModules/CMS_Test

8.4 A/B testing

8.4.1 Overview

For information about **A/B testing**, please see the Modules -> Website optimization -> A/B testing chapter of this guide.

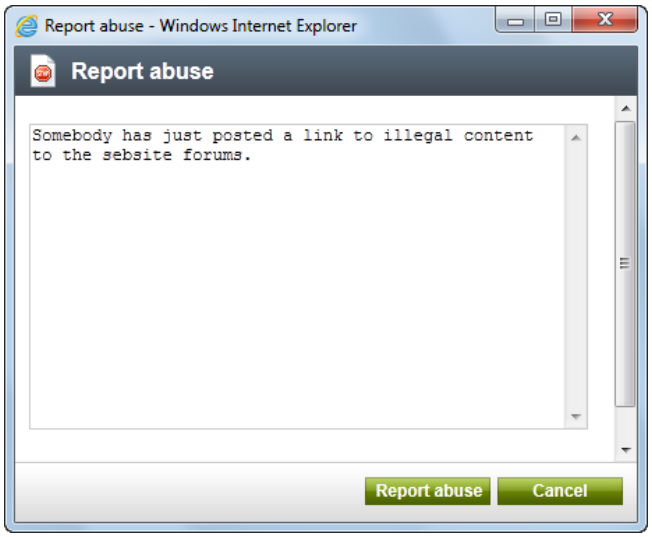
8.5 Abuse report

8.5.1 Overview

The Abuse report module enables site visitors to report various forms of website abuse. This can be anything from flame-type forum posts, rude comments on blog posts or message boards, spamming, links to illegal content or any other form of website abuse coming from user input.

The whole functionality is based on website users reporting these forms of website abuse to website administrators via one of the provided web parts - **Abuse report** and **In-line abuse report**. Both web parts are described in the Available web parts chapter. The Using the In-line abuse report web part in transformations chapter demonstrates how the **In-line abuse report** web part can be used in transformations to provide the abuse report functionality together with dynamically loaded content.

Abuse report functionality is also embedded in web parts of the Forums and Message boards modules. Advice on how to use the embedded functionality can be found in the Integration with other modules chapter.



Website administrators can read and manage submitted reports in **CMS Desk -> Tools -> Abuse report**. This is where a list of submitted reports is displayed, with the possibility to get redirected to the source of reported abuse and take an appropriate action. All abuse report management possibilities provided in this part of the administration interface are described in the Abuse reports management chapter. A user who wants to manage abuse reports must be in a role granted with appropriate permissions for the module - please consult the Security chapter for more details on the module's permissions.

The Abuse report internals and API sub-chapter provides information about the database tables and classes used by the module, as well as examples of how abuse reports can be managed using Kentico CMS API.

The Abuse report module is just one of the modules which can help you deal with unwanted user behavior on your site. If you want to keep your site free of rude language or simply filter user input based on contained keywords, the Bad words module may come in handy. If you are facing a problem with notorious spammers or have another reason why an IP address should be prevented from accessing your site, you can deny access to a particular IP address using the Banned IPs module.

8.5.2 Available web parts

This page gives just a brief overview of the Abuse report web parts. Detailed descriptions of each web part and its properties can be found in Kentico CMS Web Parts Reference.

Abuse report

The **Abuse report** web part gives site visitors the possibility to report website abuse such as using indecent expressions, offensive language, etc. In the web part selection dialog, you can find it under the **Abuse report** category. It can be placed into any web part zone on any page of your website, while you will typically add it to pages where user input is possible, e.g. pages with forums, message boards, blogs, user contributions, etc.

You only have to set the following two specific properties in order for the web part to work properly:

Confirmation text	Text message that will be displayed to the site visitor after submitting

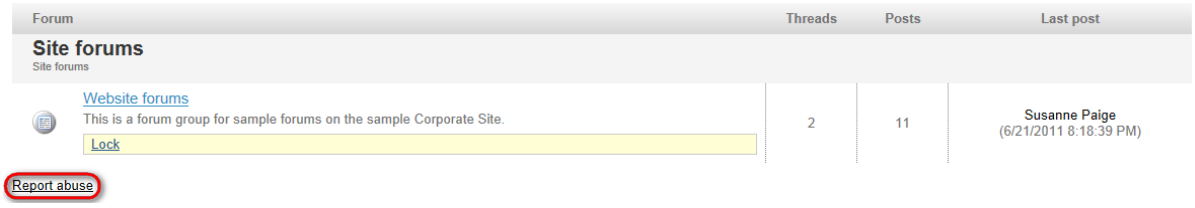
	the abuse report.
Title of abuse report	Title of the report that will be displayed in the list of abuse reports in **CMS Desk -> Tools -> Abuse report**. Using this property, you can distinguish reports submitted at different pages of your website from each other.

In the screenshot below, you can see the web part added below the forum group on the **/Community/Forums** page of the sample Corporate Site. When a user types in some report text and clicks the **Report abuse** button, a report is logged in **CMS Desk -> Tools -> Abuse report**. Read the Abuse reports management topic for information on how reports can be managed in this section of CMS Desk.

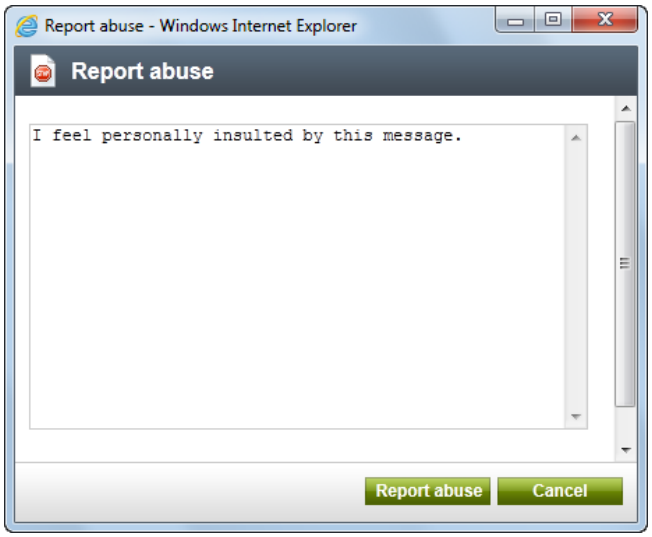
In-line abuse report

If you want to use only a tiny clickable link instead of the whole **Abuse report** web part, you can use the **In-line abuse report** web part. The web part appears only as the *Report abuse* link, as you can see in the screenshot below. Properties of the two web parts are identical.

The **In-line abuse report** web part can also be used in transformations, as explained in the Using the In-line abuse report web part in transformations topic.



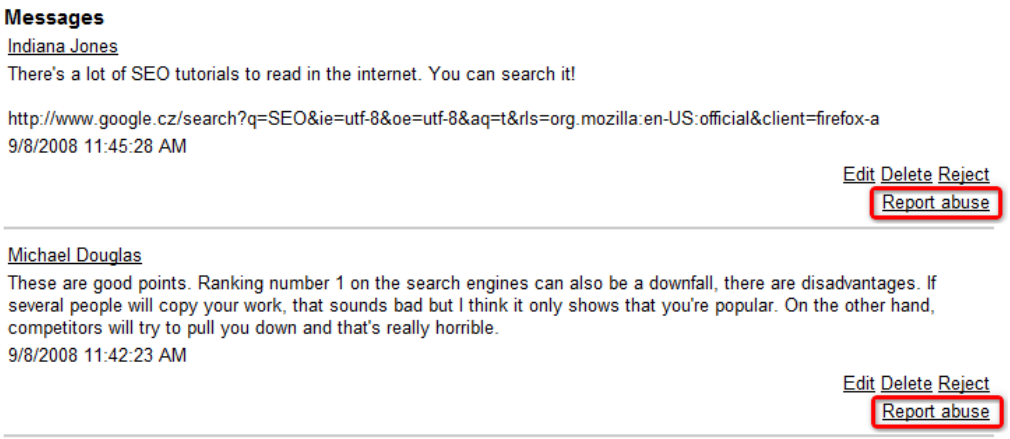
When a site visitor clicks the link, a dialog pops-up, letting the site visitors report abuse to the site administrators. Reports submitted via this web part are also logged in **CMS Desk -> Tools -> Abuse report** and can be managed there as described in the Abuse reports management topic.



8.5.3 Using the In-line abuse report web part in transformations

The Inline abuse report web part appears as a link with the *Report abuse* text. After clicking the link, a dialog appears, letting the site visitor send abuse report to the site administrators. Apart from using it as a standard web part, you can also use it in transformations to have the *Report abuse* link displayed with dynamically loaded content.

To see how the **In-line abuse report** web part is used, you can go to the **Examples -> Web parts -> Message boards -> Message board** page of the sample Corporate Site. As you can see in the screenshot below, the *Report abuse* link is included with each message on the board.



This is achieved using the In-line abuse report web part in the transformation used for displaying board messages. If you go to properties ( ) of the **Message board** web part and edit the **Message transformation** (the *community.transformations.MessageBoard* transformation should be selected by default), you should see the same code as below.

```
<%@ Register Src="~/CMSModules/MessageBoards/Controls/MessageActions.ascx" TagName="
<%@ Register Src="~/CMSModules/AbuseReport/Controls/InlineAbuseReport.ascx" TagNam
```

```
<div class="CommentDetail">
    <asp:Panel ID="pnlRating" runat="server" CssClass="CommentRating" />
    <table width="100%">
        <tr>
            <td class="CommentUserName" style="width: 100%">
              <%# IfEmpty(Eval("MessageURL"), Eval("MessageUserName", true), "<a
href=\"" + Eval("MessageURL", true) + "\" target=\"_blank\">" + Eval
("MessageUserName", true) + "</a>")%>
            </td>
        </tr>
        <tr>
            <td class="CommentText">
              <%# CMS.GlobalHelper.TextHelper.EnsureLineEndings(Convert.ToString(Eva
            </td>
        </tr>
        <tr>
            <td class="CommentDate">
              <%# GetDateTime(Eval("MessageInserted")) %>
            </td>
        </tr>
        <tr>
            <td align="right" class="CommentAction">
              <cms:MessageActions ID="messageActions" runat="server" />
            </td>
        </tr>
        <tr>
            <td align="right" class="CommentAction">
                    <cms:AbuseReport ID="ucInlineAbuseReport" runat="server"
ReportObjectType="board.message" ReportObjectID='<%# Eval("MessageID") %>'
ReportTitle='<%# "Message board abuse report: " + Eval("MessageText") %>'
CMSPanel-SecurityAccess="AuthenticatedUsers" />
            </td>
        </tr>
    </table>
</div>
<hr style="border: 1px solid #CCCCCC;"/>
```

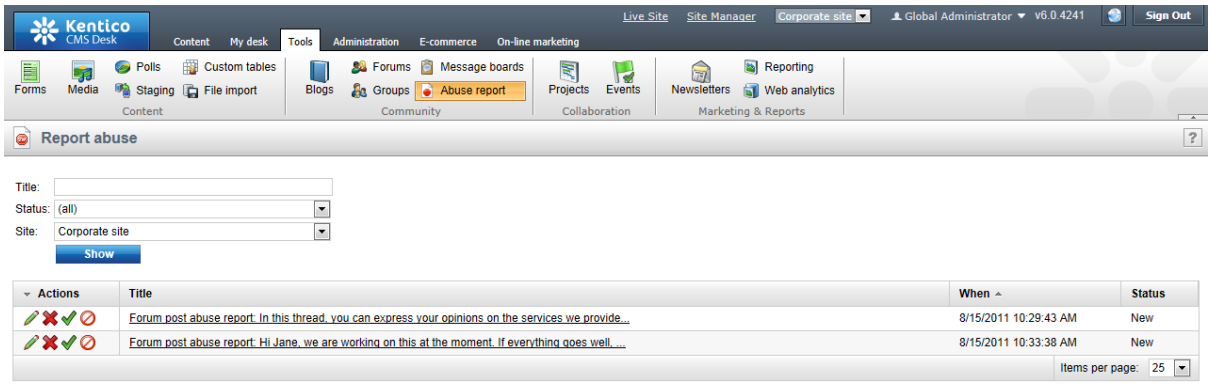
Let's take a closer look at the highlighted parts of the code. The highlighted line at the top is here to register the web part (actually control) so that it can be used in transformation code. As you can see, the web part code file is *~/CMSModules/AbuseReport/Controls/InlineAbuseReport.ascx*.

The second highlighted section is the actual web part, identified by the tag name and prefix defined in the *Register* tag. The *ReportObjectType="board.message"* and *ReportObjectID='<%# Eval ("MessageID") %>* parameters ensure that it will be possible to view details of a particular message (source of reported abuse) from the administration interface. These parameters pass the ID of the particular board message along with the report, so that the message can be identified and a link to it created. The *ReportTitle='<%# "Message board abuse report: " + Eval("MessageText") %>* parameter defines the title of the report displayed in the administration interface. The last parameter - *CMSPanel-SecurityAccess="AuthenticatedUsers"* - ensures that the link will be displayed only to authenticated users.

This way, you can use the web part in transformations of any document types you need. The only limitation is that the object details functionality is available only with **board messages**, **forum posts** and **blog comments**. Abuse reporting is still possible with other object types, but no object details are passed with the report and therefore can't be displayed in the administration interface.

8.5.4 Abuse reports management

Abuse reports can be viewed and managed in **CMS Desk -> Tools -> Abuse report**.



The top part of the page is a filter. By default, you get all reports displayed in the list below it. Using the filter, you can display only those reports that match the specified criteria. Available filtration parameters are **Title**, **Status** and **Site** from which the report was sent. To filter the reports, enter the appropriate parameters and click the **Show** button. A list of reports matching the criteria will be displayed.

The **Status** column shows which status is the report currently in. You can use the statuses to mark reports, so that **New**, **Solved** and **Rejected** reports can be distinguished from each other. By switching a report to a status, only the status changes - no other action happens. In other words, status switching is here just for your convenience, but you need to perform a suitable action manually.

The following actions are available for each of the listed reports:

- **Edit -** if clicked, the user will be redirected to report properties page, where you can edit the report's properties
- **Delete -** deletes the report
- **Mark as solved -** switches the report to the *Solved* status; used to mark reports for that the necessary actions have been taken
- **Mark as rejected -** switches the report to the *Rejected* status; used to mark reports that were not considered being cases of website abuse

If you click a title of a report, you will be redirected to the page that the report was sent from. If you mouse-over it, report description entered by the sender of the report will be displayed in form of a tooltip.

Editing a report

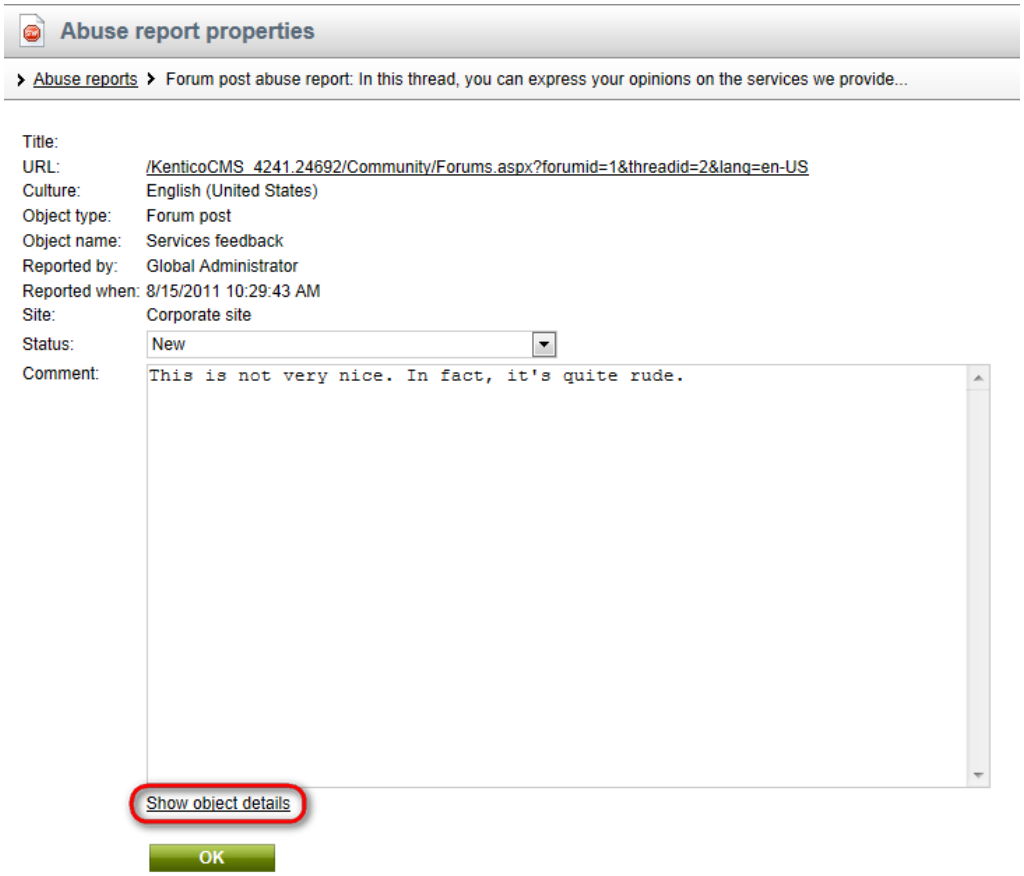
When editing ( ) a report, the following information is displayed:

Title	Title of the abuse report.
URL	URL of the page from which the report was sent. Click it to get redirected to that page.
Culture	Website culture from that the report was sent.

Object type	Type of object that was the cause of this report. If blank, the report was sent from some document.
Object name	Code name of the object that was the cause of this report.
Reported by	Site user who submitted the report.
Reported when	Time when the report was submitted.
Site	Website from that the report was submitted.
Status	Abuse report status, the following are possible: • **New** - the report is new and has not been solved yet • **Solved** - necessary actions have already been taken • **Rejected** - the report was not considered being a case of website abuse
Comment	Comment of the report entered by the reporting user.

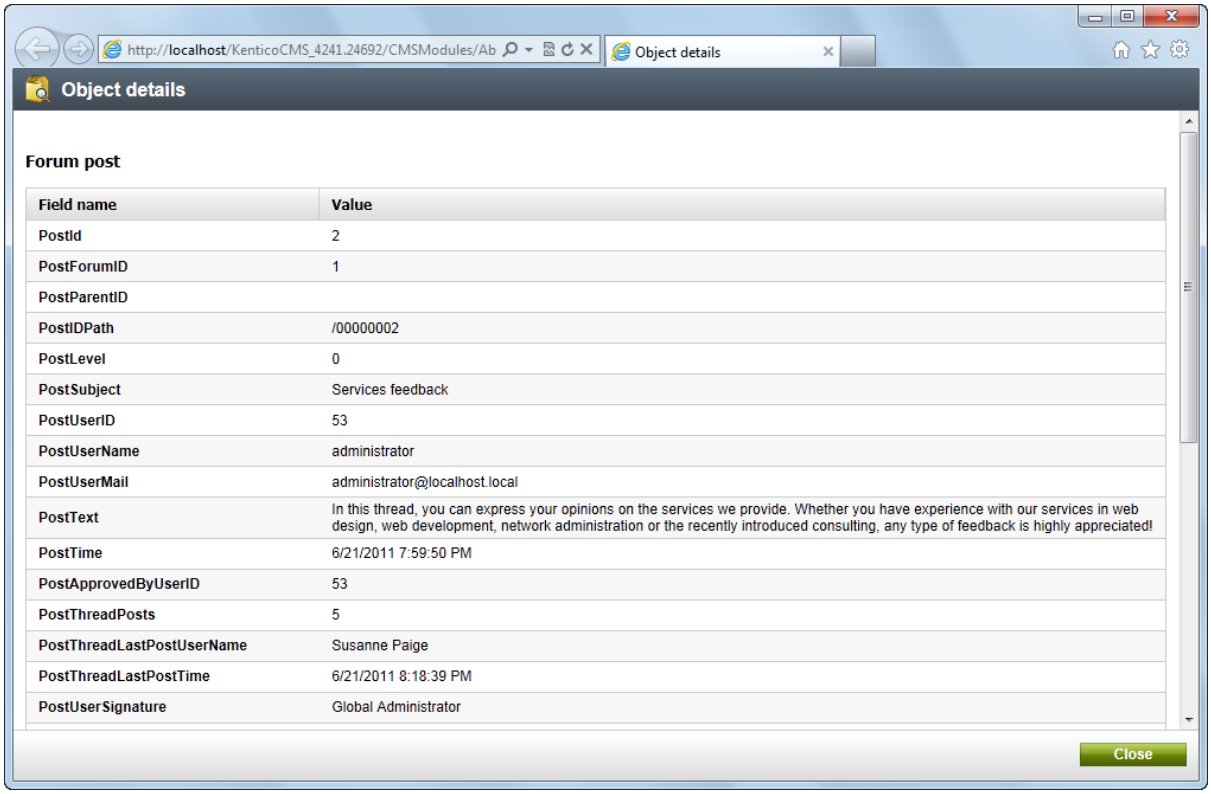
Object details

If you edit (✎) a report related to a **blog comment**, **board message** or **forum post**, the **Show object details** link is displayed below report comment.



Clicking the link opens a new window with details about the source of the report (the board message,

blog comment or forum post). This functionality is ensured automatically by the web parts listed in the Integration with other modules topic. You can also achieve it in transformations of these object types using the **In-line abuse report** web part, as described in the Using the In-line abuse report web part in transformations topic.



8.5.5 Integration with other modules

Apart from dedicated web parts described in the Available web parts chapter, the abuse report functionality is also embedded in the following web parts:

- **Blogs -> Comment view**
- **Community -> Group forum list**
- **Forums -> Forum group**
- **Forums -> Forum (Single forum - General)**
- **Forums -> Forum (Single forum - Tree layout)**
- **Forums -> Forum (Single forum - Flat layout)**

Each of these web parts has the **Abuse report** section in its properties. The section contains the following properties:

- **Who can report abuse** - using this property, you can determine to whom will the *Report abuse* link be displayed and who will therefore be able to submit abuse reports; the following options are available:
  - **All users** - the link will be displayed to anyone who views a page with the web part
  - **Authenticated users** - the link will be displayed only to authenticated users
  - **Authorized roles** - the link will be displayed only to users who are members of the roles specified by the *Authorized roles* property

- **Nobody** - the link will not be displayed at all
- **Authorized roles** - if the *Who can report abuse* option is set to *Authorized roles*, you can use this property to specify the roles to whose members the abuse report link will be displayed

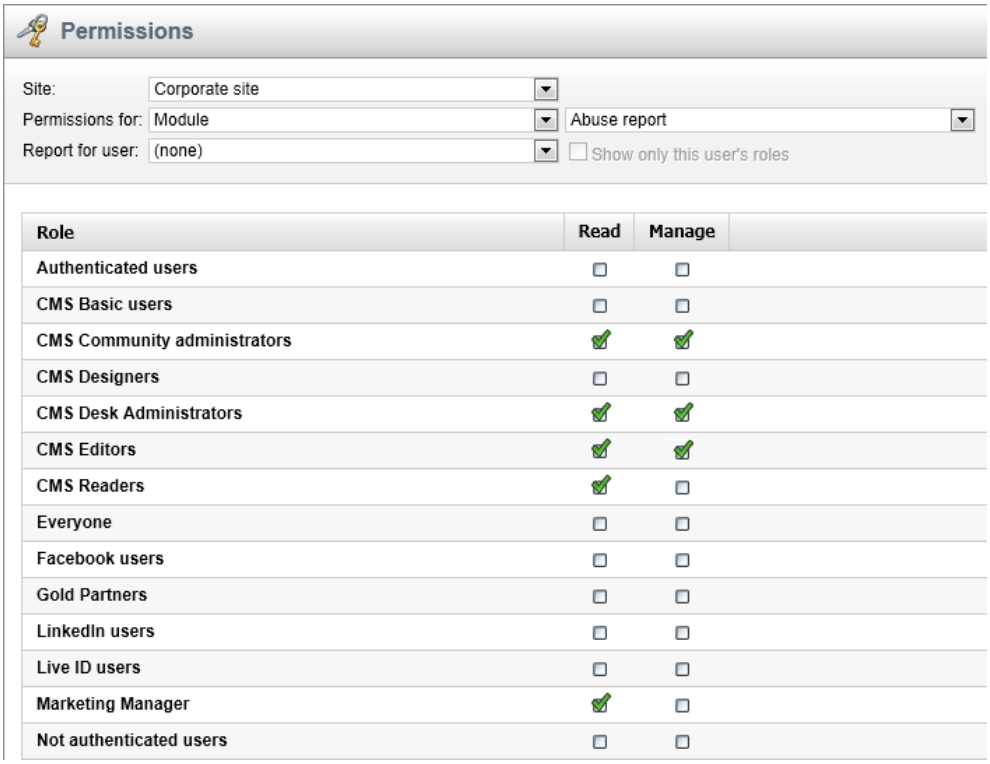
For a complete reference on Kentico CMS web parts and their properties, please consult Kentico CMS Web Parts Reference.

8.5.6 Security

Permissions for the Abuse report module can be set in **Site Manager -> Administration -> Permissions**. Select the **Modules -> Abuse report** permission matrix and grant appropriate permissions to particular roles.

The following permissions can be granted to the roles:

- **Manage** - members of the role are allowed to edit, delete, mark as solved and reject abuse reports
- **Read** - members of the role are allowed to view the abuse reports list



8.5.7 Abuse report internals and API

8.5.7.1 Overview

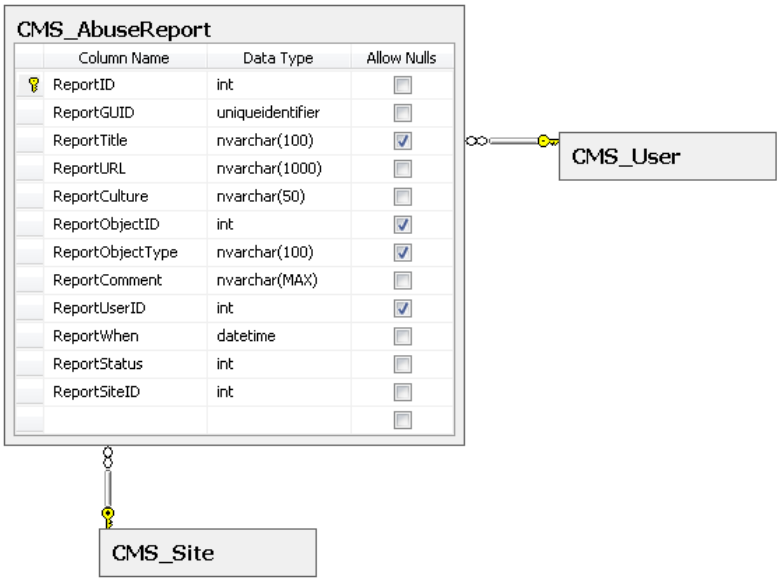
In this chapter, you will learn which database tables and API classes are used in the Abuse report module. You will also see the most common API examples.

Advanced API examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all methods from the module classes, please refer to Kentico CMS API Reference.

8.5.7.2 Database tables

The Abuse report module uses the following database table:

- **CMS_AbuseReport** - contains records representing submitted abuse reports.



8.5.7.3 API classes

If you are not familiar with database table data management through Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

The classes of the Abuse report module can be found in the **CMS.SiteProvider** namespace.

CMS_AbuseReport table API:

- **AbuseReportInfo** - represents one abuse report object.
- **AbuseReportInfoProvider** - provides management functionality for abuse reports.

8.5.7.4 API examples

8.5.7.4.1 Overview

These topics show examples of how the API of the Abuse report module can be used:

- Managing abuse reports

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Tools\AbuseReport\Default.aspx.cs***.

The Abuse report API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.GlobalHelper;
using CMS.UIControls;
using CMS.CMSHelper;
using CMS.SiteProvider;
```

8.5.7.4.2 Managing abuse reports

The following example creates an abuse report.

```
private bool CreateAbuseReport()
{
    // Create new abuse report object
    AbuseReportInfo newReport = new AbuseReportInfo();

    // Set the properties
    newReport.ReportTitle = "MyNewReport";
    newReport.ReportComment = "This is an example abuse report.";

    newReport.ReportURL = URLHelper.GetAbsoluteUrl(URLHelper.CurrentURL);
    newReport.ReportCulture = CMSContext.PreferredCultureCode;
    newReport.ReportSiteID = CMSContext.CurrentSiteID;
    newReport.ReportUserID = CMSContext.CurrentUser.UserID;
    newReport.ReportWhen = DateTime.Now;
    newReport.ReportStatus = AbuseReportStatusEnum.New;

    // Save the abuse report
    AbuseReportInfoProvider.SetAbuseReportInfo(newReport);

    return true;
}
```

The following example gets and updates the abuse report created by the code example above.

```
private bool GetAndUpdateAbuseReport()
```

```
{
    string where = "ReportTitle LIKE N'MyNewReport%'";

    // Get the report
    DataSet reports = AbuseReportInfoProvider.GetAbuseReports(where, null);

    if (!DataHelper.DataSourceIsEmpty(reports))
    {
        // Create the object from DataRow
        AbuseReportInfo updateReport = new AbuseReportInfo(reports.Tables[0].Rows
[0]);

        // Update the properties
        updateReport.ReportStatus = AbuseReportStatusEnum.Solved;

        // Save the changes
        AbuseReportInfoProvider.SetAbuseReportInfo(updateReport);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates multiple abuse reports selected from database based on a WHERE condition.

```
private bool GetAndBulkUpdateAbuseReports()
{
    // Prepare the parameters
    string where = "ReportTitle LIKE N'MyNewReport%'";

    // Get the data
    DataSet reports = AbuseReportInfoProvider.GetAbuseReports(where, null);
    if (!DataHelper.DataSourceIsEmpty(reports))
    {
        // Loop through the individual items
        foreach (DataRow reportDr in reports.Tables[0].Rows)
        {
            // Create object from DataRow
            AbuseReportInfo modifyReport = new AbuseReportInfo(reportDr);

            // Update the properties
            modifyReport.ReportStatus = AbuseReportStatusEnum.Rejected;

            // Save the changes
            AbuseReportInfoProvider.SetAbuseReportInfo(modifyReport);
        }

        return true;
    }
```

```
    return false;
}
```

The following example deletes the abuse report created by the first code example on this page.

```
private bool DeleteAbuseReport()
{
    string where = "ReportTitle LIKE N'MyNewReport%'";

    // Get the report
    DataSet reports = AbuseReportInfoProvider.GetAbuseReports(where, null);

    if (!DataHelper.DataSourceIsEmpty(reports))
    {
        // Create the object from DataRow
        AbuseReportInfo deleteReport = new AbuseReportInfo(reports.Tables[0].Rows
[0]);

        // Delete the abuse report
        AbuseReportInfoProvider.DeleteAbuseReportInfo(deleteReport);

        return true;
    }

    return false;
}
```

8.6 Alternative forms

8.6.1 Overview

The Alternative forms module enables you to create alternatives of various already existing forms. The alternative forms can then be used instead of the default ones in the system's administration interface or on the live site. You can even create multiple alternative forms for a single object and use each of them in a different situation.

Alternative forms can be created for:

- Forms - you can create alternative forms for existing forms and use them on the live site instead of the default form, while you can conveniently switch between various alternative forms in properties of the **On-line form** web part. Using alternative forms, you can also replace the default forms for creating or editing forms in the system's administration interface.
- Custom tables - using alternative forms, you can only replace a custom table's default form for adding or editing custom table items in the system's administration interface.
- Document types - you can create alternative forms for each document type. A typical usage of this feature is in the User contributions module, where you can provide users with a form for creation or editing of user-contributed documents on the live site which is different from the one used in the user interface. Using alternative forms, it is also possible to replace the default forms for adding or editing of documents in **CMS Desk -> Content -> Edit** (after clicking the **New** button or on the document's **Form** tab, respectively).

- System tables - you can create alternative forms for the system tables. A typical example is the **User** system table, as it has a dedicated alternative form to display a user profile, another form for profile editing, and yet another one for user registration. Using alternative forms, it is also possible to replace the default forms for adding or editing system tables data via the administration interface.

The module has no dedicated user interface, there only is the **Alternative forms** tab available when editing ( ) one of the objects listed above. On this tab, you can create and manage alternative forms of the currently edited item. In the Creating an alternative form topic, you can see an example of how an alternative form for a form can be created and used on the live site.

Some actions (typically creation of a new item or editing of an existing one) are associated with a reserved alternative form code name. If there is a form defined in the system which has the special code name, it is used instead of the default form when the respective action is performed. Please see the Automatically used alternative forms topic for more details.

Data about users of the system are stored in the **User** and **User - Settings** system tables. When creating an alternative form for creation or editing of users, it is possible to have fields from both of the system tables included in a single alternative form. For more information, please see the Joining two classes into one form topic.

8.6.2 Creating an alternative form

This example shows how to create an alternative form of the existing **Contact us** form on the sample Corporate Site.

Alternative forms of document types, system tables and custom tables can be created exactly the same way as described in this example. You only need to access the **Alternative forms** tab in the appropriate sections of the UI:

- **Site manager -> Development -> Custom tables**
- **Site manager -> Development -> Document types**
- **Site manager -> Development -> System tables**

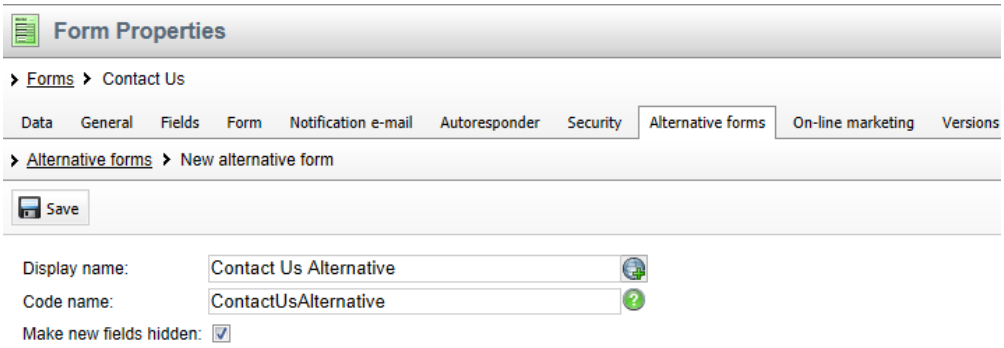
There, **Edit** ( ) the particular item, switch to its **Alternative forms** tab and follow the instructions below, starting from the second step.

1. Go to **CMS Desk -> Tools -> Forms** and **Edit** ( ) the **Contact us** form. The **Form Properties** dialog opens.

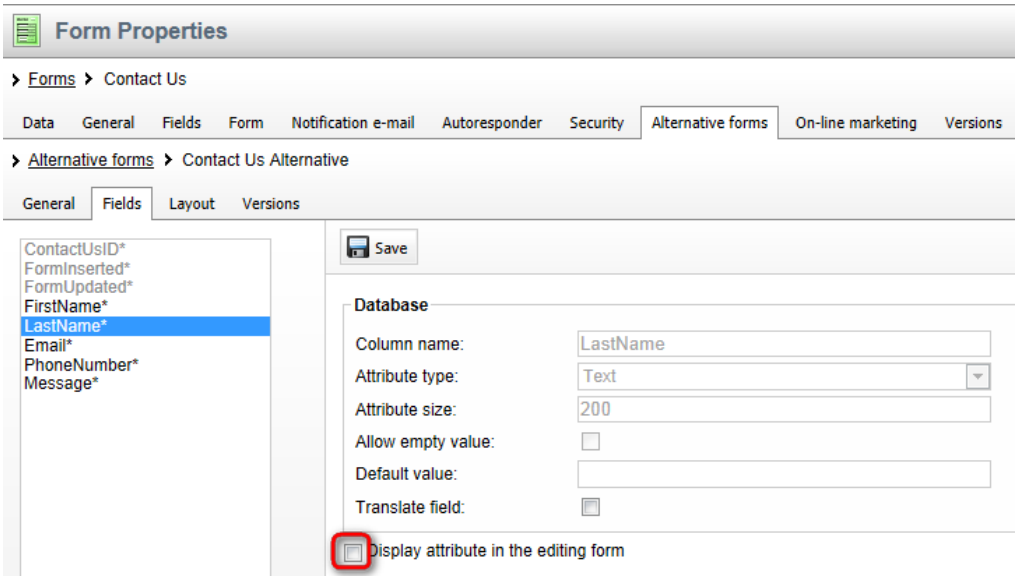
2. Switch to the **Alternative forms** tab and click the **Create new form** button.

3. Enter the following details and click **Save**.

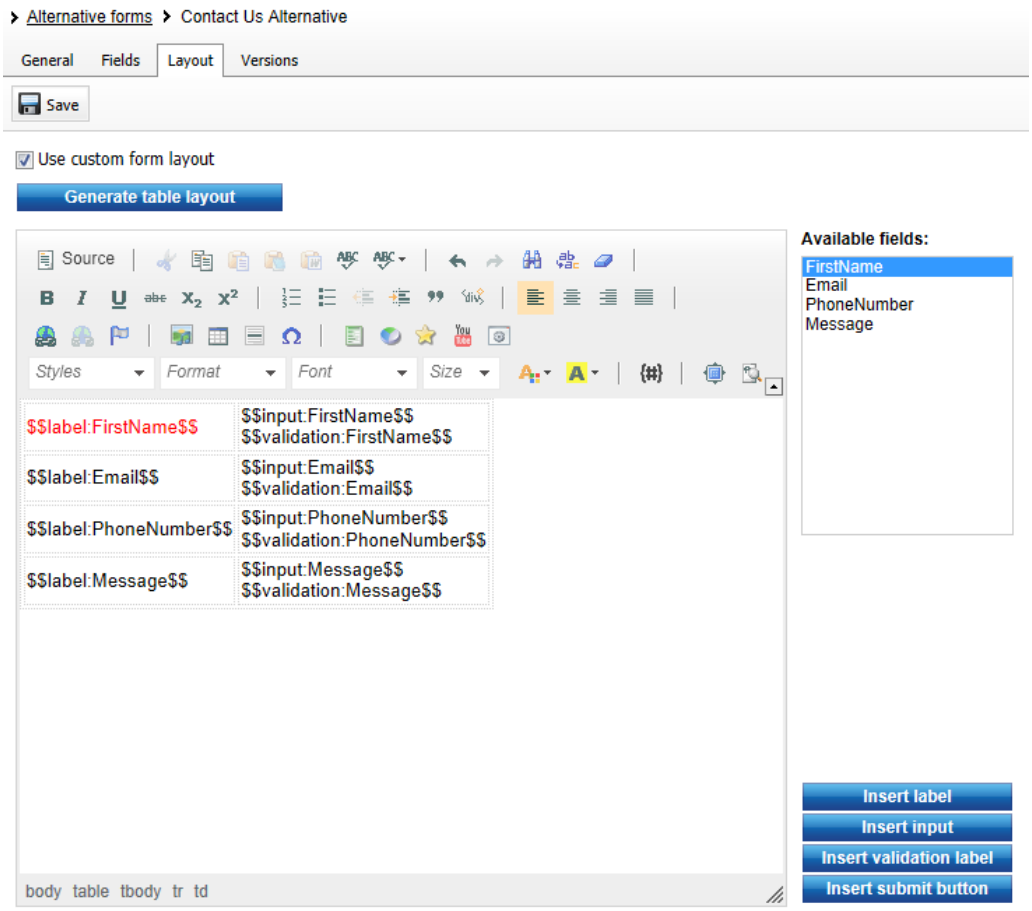
- **Display name**: Contact Us Alternative
- **Code name**: ContactUsAlternative
- **Make new fields hidden**: yes (this ensures that any new fields added to the main form are not displayed in the alternative form by default)



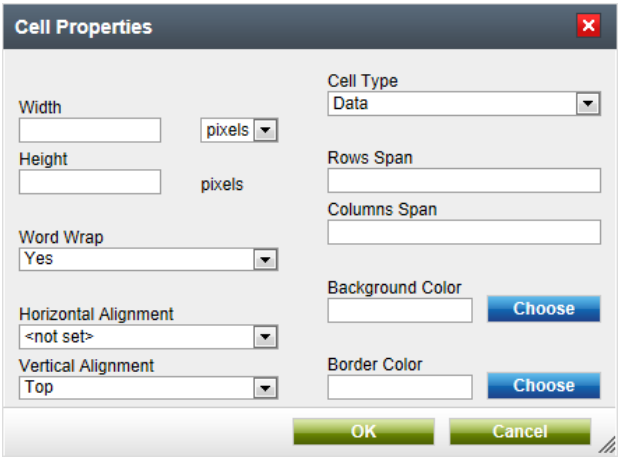
4. Switch to the **Fields** tab. All fields present in the original form are also available here and you can modify their configuration. For example, disable the **LastName** field. Select the field in the list on the left and uncheck **Display attribute in the editing form**. Confirm this change by clicking **Save**. Using the same approach, you can modify any field in the form according to your needs.



5. You can also modify the layout of the form. Switch to the **Layout** tab and check the **Use custom layout** checkbox. The layout editor appears. Notice that the **LastName** field that we disabled in the previous step is not offered in the **Available fields** listbox. Click the **Generate table layout** button. The system generates a default table layout in the editing area below. Highlight the *$$label:FirstName$$* text in the first row and change its color to red using the WYSIWYG editor.

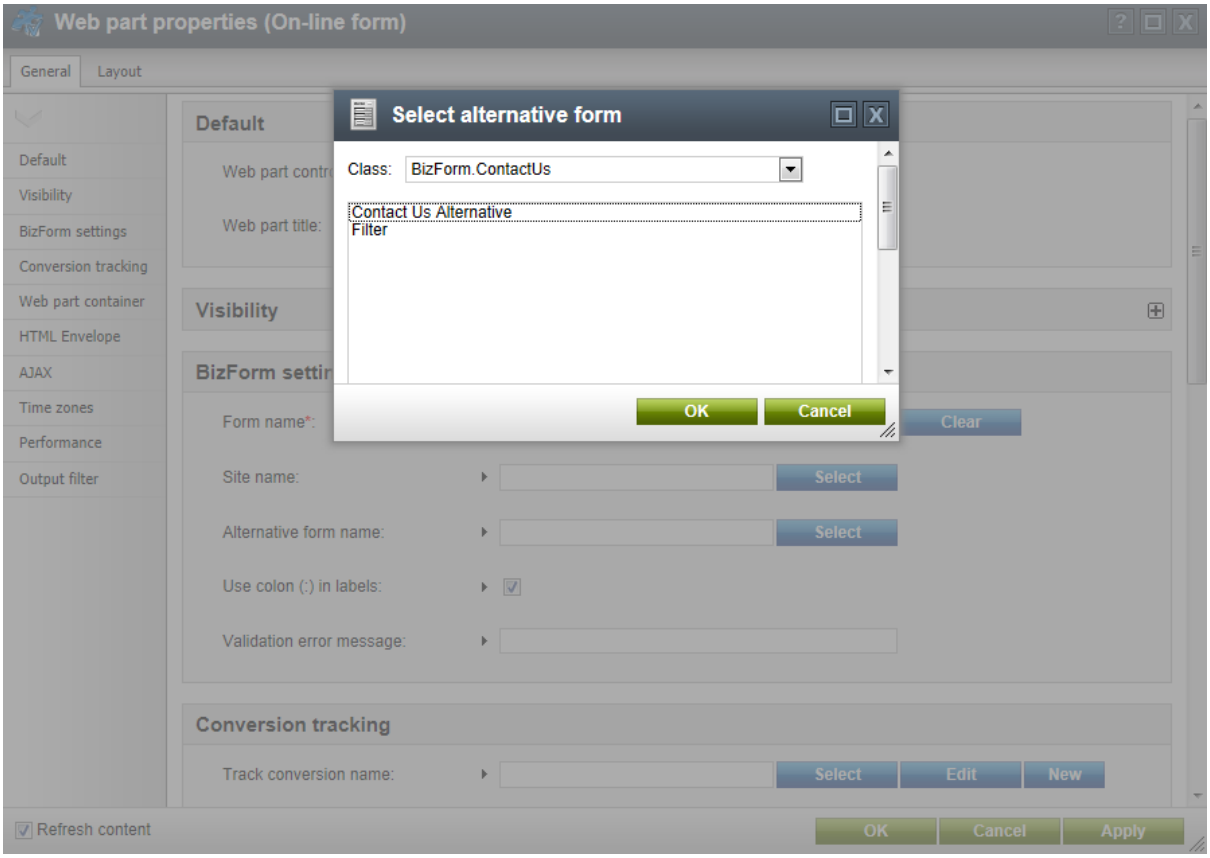


Then select the rows of the table's first column, right-click and choose **Cell -> Cell properties** from the context menu. In the displayed dialog, choose **Vertical alignment: Top** and click **OK**. This aligns the labels with their fields. Click **Save**.

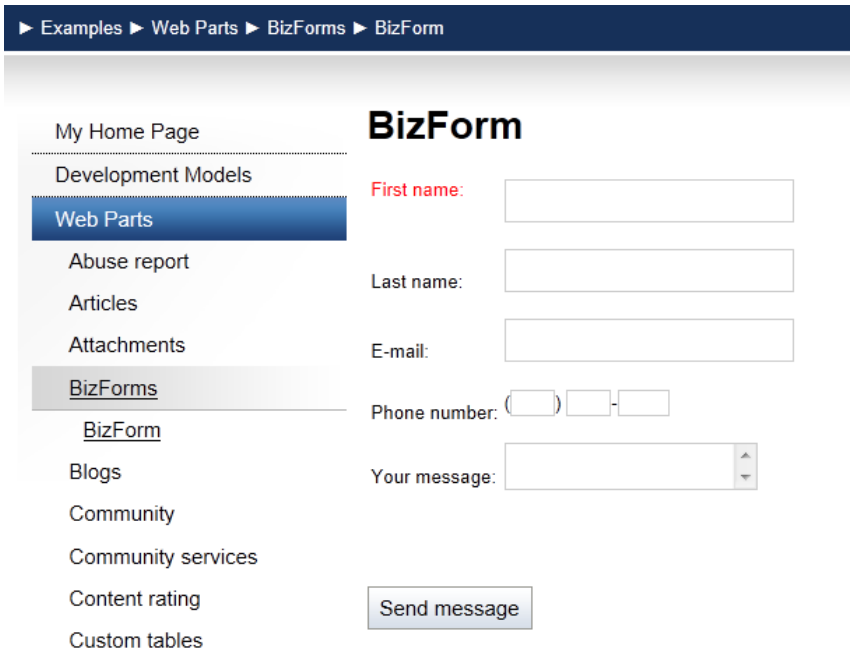


6. Let's take a look at what we've created. Switch to **CMS Desk -> Content**. From the content tree, select **Examples -> Web parts -> Forms -> On-line form**. As you can see, the original version of the **Contact Us** form is present on the page. We will edit the web part's properties so that the alternative from is used instead.

7. In **Edit** mode, switch to the page's **Design** tab and configure ( ) the **On-line form** web part's properties. Click the **Select** button next to the **Alternative form name** property. In the next dialog, choose the *BizForm.ContactUs* class and select the **Contact Us Alternative** option. Click **OK** in both dialogs to close them and confirm the changes.



8. Now when you switch to the **Live site** mode, you should see the modified version as in the following screenshot:

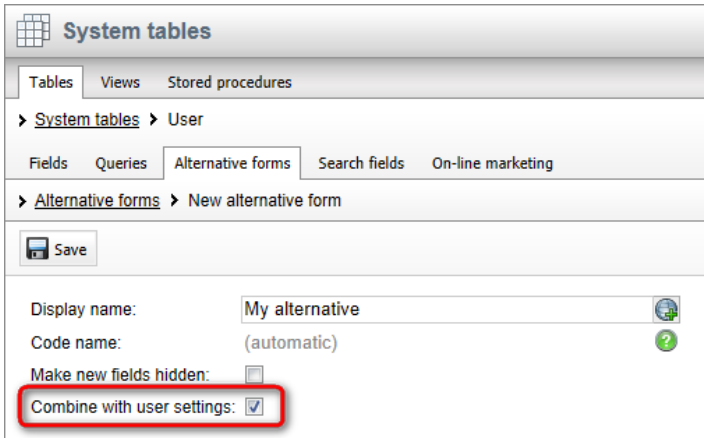


You have learned how to create an alternative form to an existing form and use it on your website.

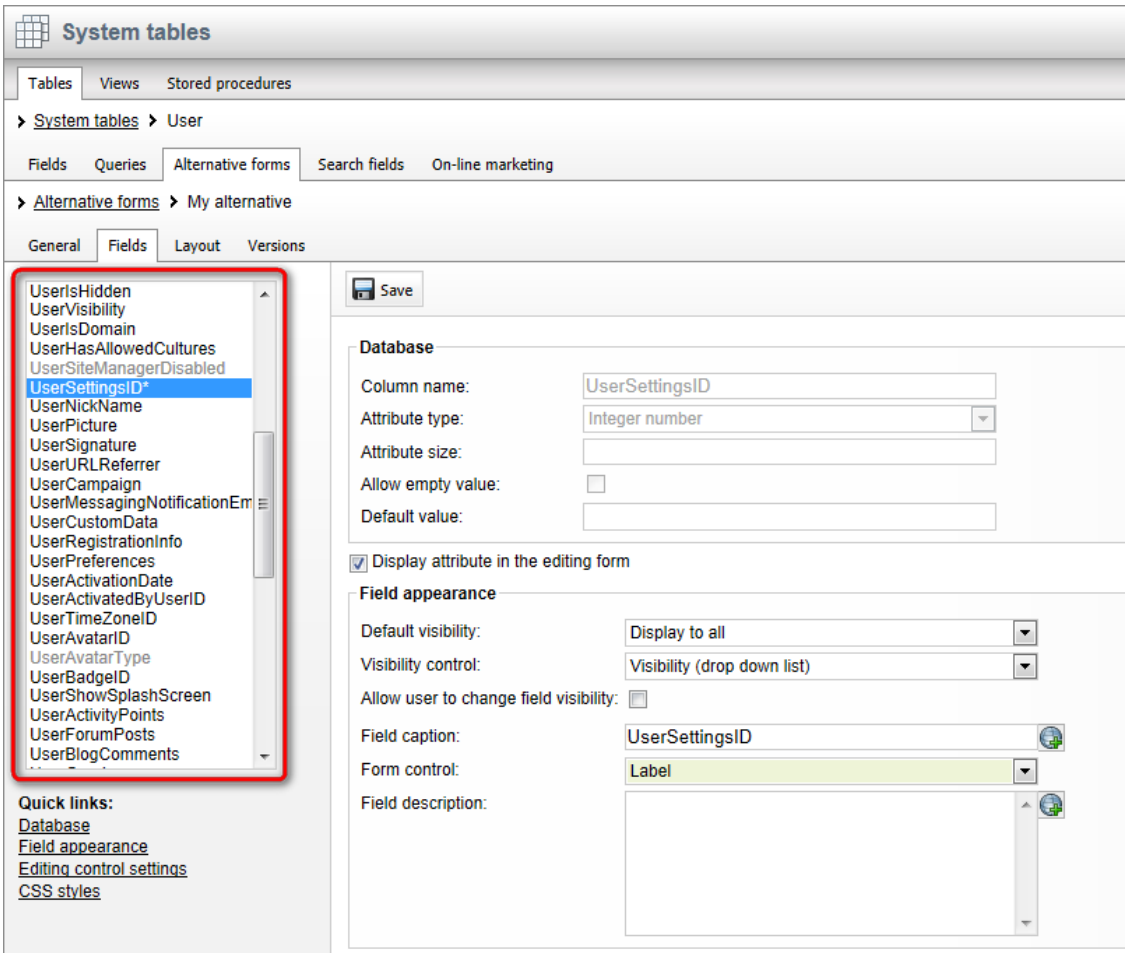
8.6.3 Joining two classes into one form

It is also possible to join two classes into one alternative form. This option is currently available only for the **User** and **User - Settings** system tables.

1. When creating an alternative form for the **User** system table, you have the option to check the **Combine with user settings** checkbox.



2. Now if you switch to the **Fields** tab, you will see that in addition to the original fields contained in the **Users** table, fields from the **User - Settings** table are also available in the list, as you can see in the following screenshot:



8.6.4 Automatically used alternative forms

If you create an alternative form and give it one of the reserved code names listed below, the alternative form will be automatically used when performing the corresponding action in the system's administration interface. For example, if you create an alternative form for a document type and give it the *insert* code name, the form will be used when documents of the type are created in **CMS Desk -> Content**.

The following table shows the reserved code names and the respective actions for which the particular forms are used:

Alternative form code name	Usage	Supported by
filter	Filter displayed above a list of items. See Creating filter forms for more details.	Forms, Custom tables
insert	Form used when creating a new item.	Forms, Custom tables, document types, system tables
newculture	Form used when creating a new culture version of a document.	document types
update	Form used when editing an existing item.	Forms, Custom tables, document types, system tables

8.6.5 Creating filter forms

By creating an alternative form named **filter** for a form or custom table, you can create a filter that will be used when a large number of records is displayed in:

- CMS Desk -> Tools -> Forms -> edit (✎) a form -> Data
- CMS Desk -> Tools -> Custom tables -> edit (✎) a custom table
- Site Manager -> Development -> Custom tables -> edit (✎) a custom table -> Data

The number of records required for the filter to be displayed is 25 by default. You can change this value by adding the following key into the *appSettings* section of your *web.config* file:

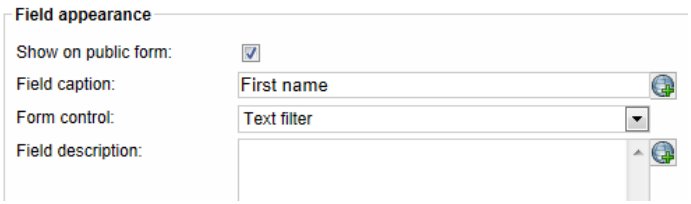
```
<add key="CMSDefaultListingFilterLimit" value="10" />
```

Filtering is possible based on all fields that store the following types of values:

- Text
- Boolean (Yes/No)
- Integer numbers
- Long integer numbers
- Decimal numbers
- Date & time

The required fields need to be displayed in the alternative form and an appropriate form control must be assigned to each field. A filter form control is available for each data type, including the following options:

- **Text filter**
- **Boolean filter**
- **Number filter**
- **Date & time filter**



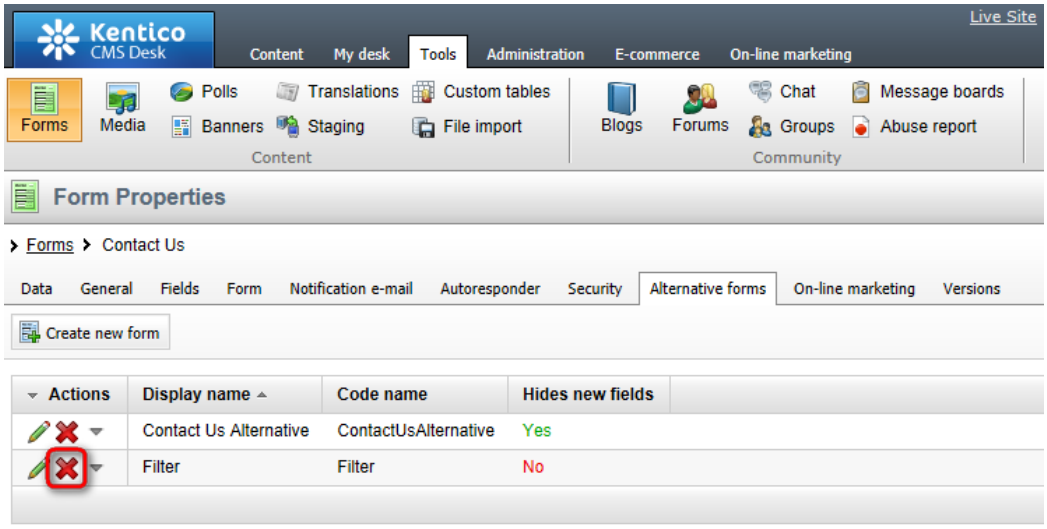
Creating a form filter

The following example will show you how to create a filter for the **Contact Us** form on the sample **Corporate Site**. The procedure demonstrated in the example can be used for any other form. Creation of custom table filters is also performed the same way.

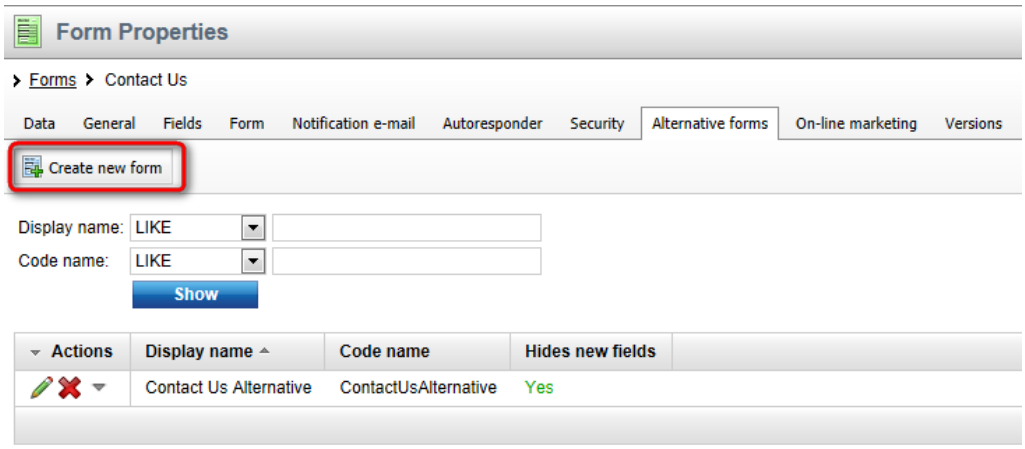
Please note that in a standard installation, the **Contact Us** form already has a pre-defined **filter** alternative form. If you are an experienced user, you can inspect its setting instead of going through the following example.

1. Sign in to **CMS Desk** and go to the **Tools -> Forms** section. Edit (✎) the **Contact Us** form and

switch to the **Alternative forms** tab of its editing interface. As mentioned in the previous paragraph, there should already be a pre-defined **filter** form. Click the **Delete** (✖) action next to it so that you can go through the rest of this example and create your own filter from scratch.



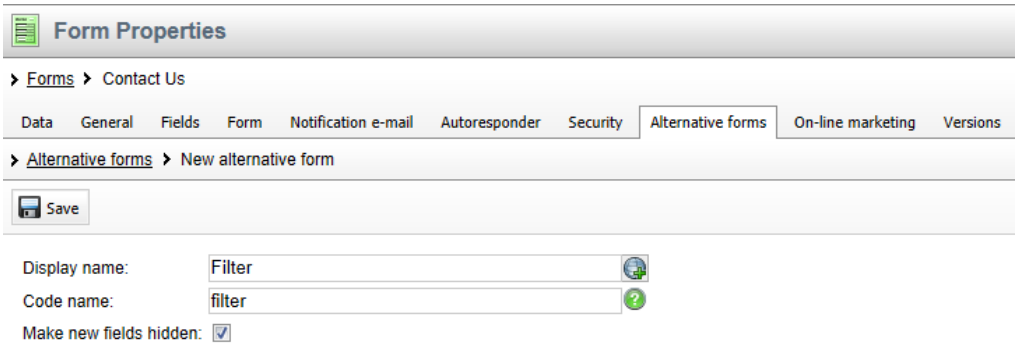
2. Once you have the form deleted, click **Create new form** to create a new one.



3. In the **New alternative form** dialog, enter the following details:

- **Display name**: Filter
- **Code name**: filter
- **Make new fields hidden**: yes (checked)

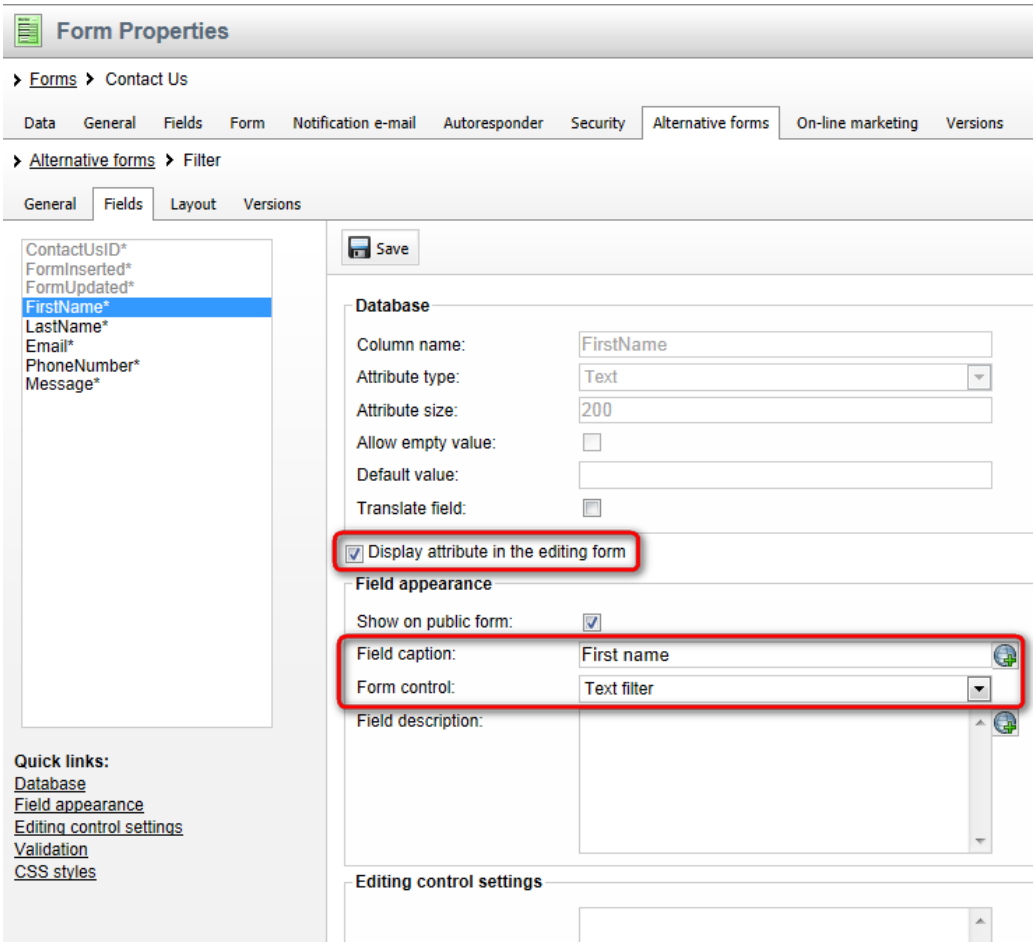
Click **Save**.



4. Now that the form is created, let's go to the most essential part of its configuration — switch to the **Fields** tab. In the listbox on the left, you should see all fields defined for the form. The first three of them are system fields that we do not want in the filter, so select the fourth one - **FirstName**. In the right area, scroll down to the **Field appearance** section and adjust the following values:

- **Display attribute in the editing form**: *enabled*; ensures that the field will be included in the filter's form.
- **Field caption**: First name; caption displayed next to the filtering field.
- **Form control**: *Text filter*; this is the filter form control available for text fields.

Click **Save** when you are finished.



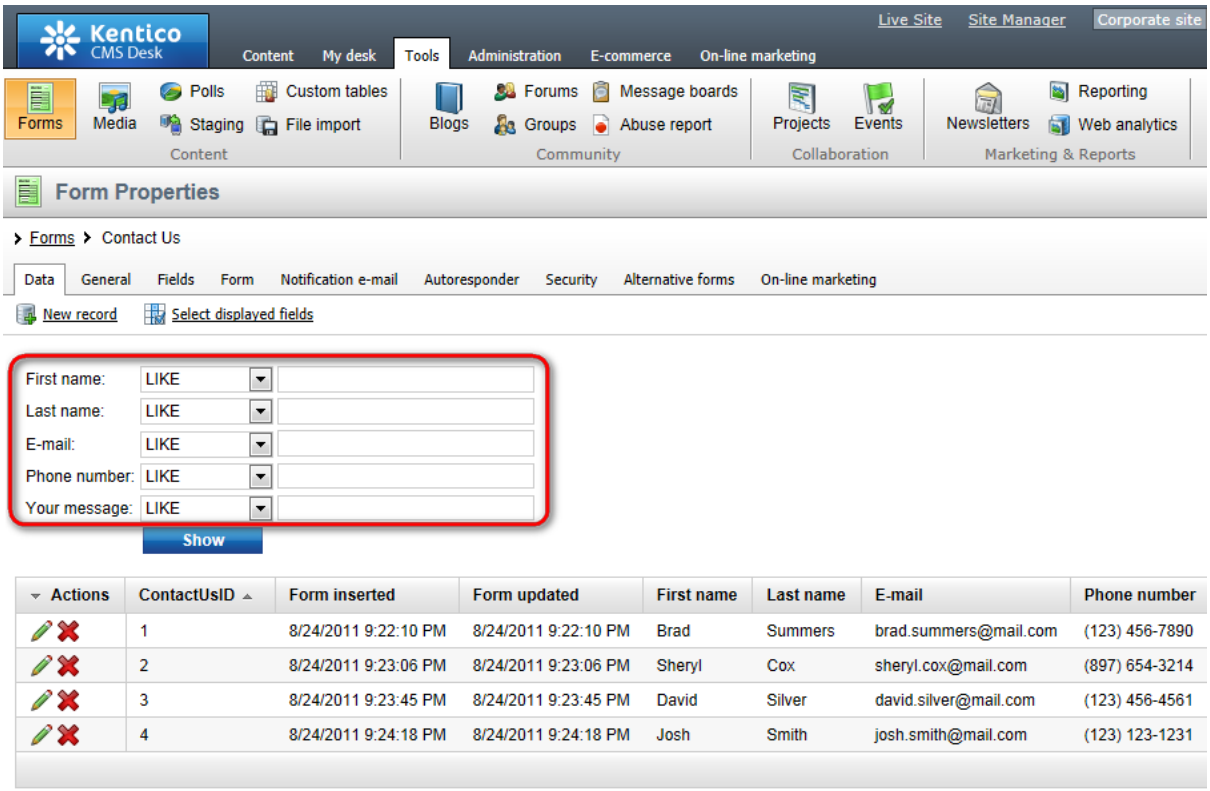
5. Repeat the configuration explained in step 4 for the **LastName**, **Email**, **PhoneNumber** and **Message** fields. This ensures that all the fields are included in the filter.

6. Once you have the filter created, you can go back to the **Data** tab of the form's editing interface to verify that it really works. A filter should be displayed on the **Data** tab as can be seen in the screenshot below. Try filtering based on various parameters.

Please note

The default number of 25 records has to be present in the list in order for the filter to be displayed.

Therefore to see the filter, you either need to create the required number of records, or use the **CMSDefaultListingFilterLimit** *web.config* key to lower the filter limit accordingly.



8.7 Avatars

8.7.1 Overview

The Avatars module enables users to have an image associated with their account. This image is called an **avatar** and is displayed on the user's public profile, next to their name in forums posts, blog comments etc. An avatar serves as a graphical representation of a user, and is used to personalize that user's contributions on websites. More information about users in general can be found in the Development -> Membership, permissions and security chapter of this guide.

Users can choose an avatar from a gallery of predefined avatars (if this option is enabled) or create a custom avatar by uploading their own image from a file on their local disk. Unlike predefined avatars, custom avatars can't be selected by other users and are deleted from the system if the user who uploaded them changes their avatar. All standard image formats are supported, including animations.

Kentico CMS also offers the option of retrieving images from the Gravatar hosting service for avatars. Please see the Gravatars topic to learn how this functionality can be configured.

Community groups and workgroups may also have avatars. These are displayed on the group's profile and can benefit the group by providing a way for it to be identified better and faster, etc. To read more about groups, please refer to the Modules -> Groups chapter of this guide.

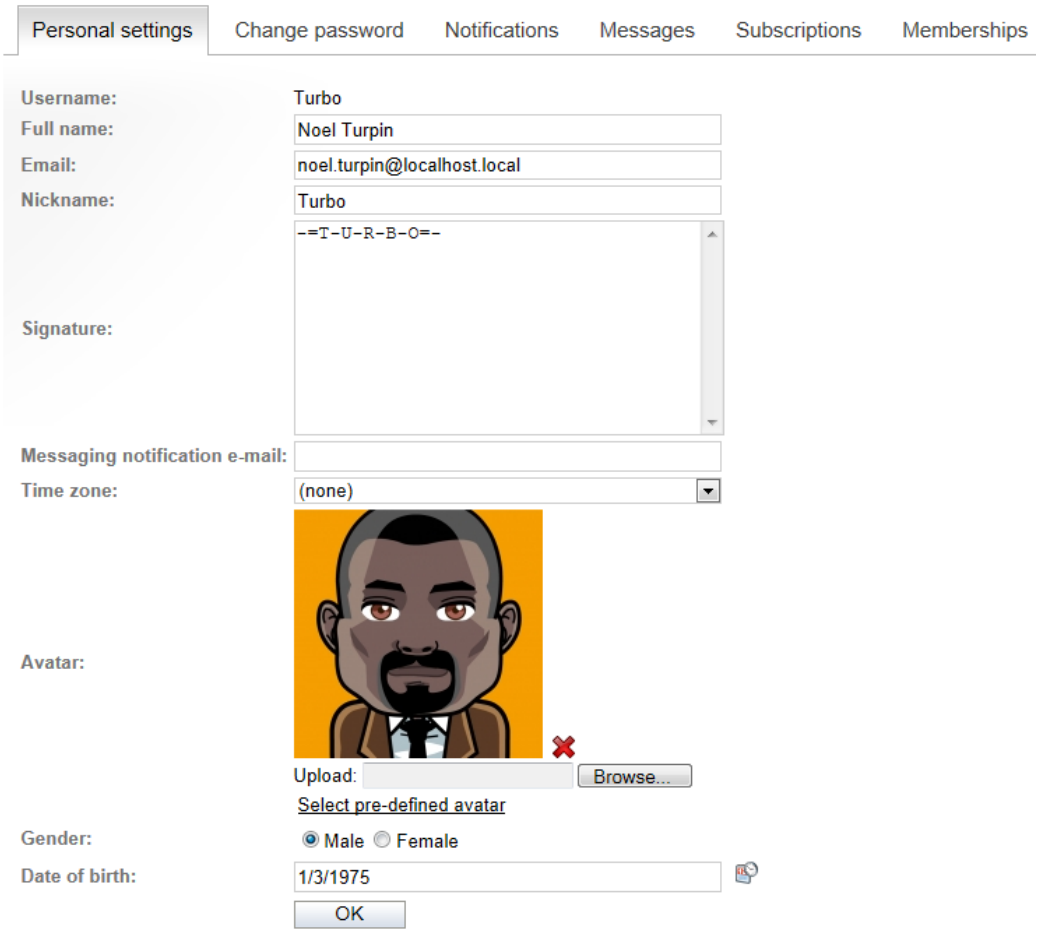
There are several ways for users or group administrators to add or change avatars. See the Changing user avatars and Changing group avatars topics for more details. Administrators can manage all locally stored avatars as described in the Managing avatars topic.

When displaying lists of users or groups on a website, you may often wish to display the matching avatar images alongside the items. This can be achieved by ensuring that the avatar is included in the transformation used to render the objects. Please see the Displaying avatars in transformations topic for details and examples.

The Avatars internals and API sub-chapter provides information about the database tables and classes used by the module, examples of how avatars can be managed using the API and how they can be displayed in transformations.

8.7.2 Changing user avatars

When a new user registers on a site, a default avatar will be assigned to them. After that, they can change their avatar on a page containing the My account web part. This can be done on the **Personal settings** tab, as you can see in the screenshot below.



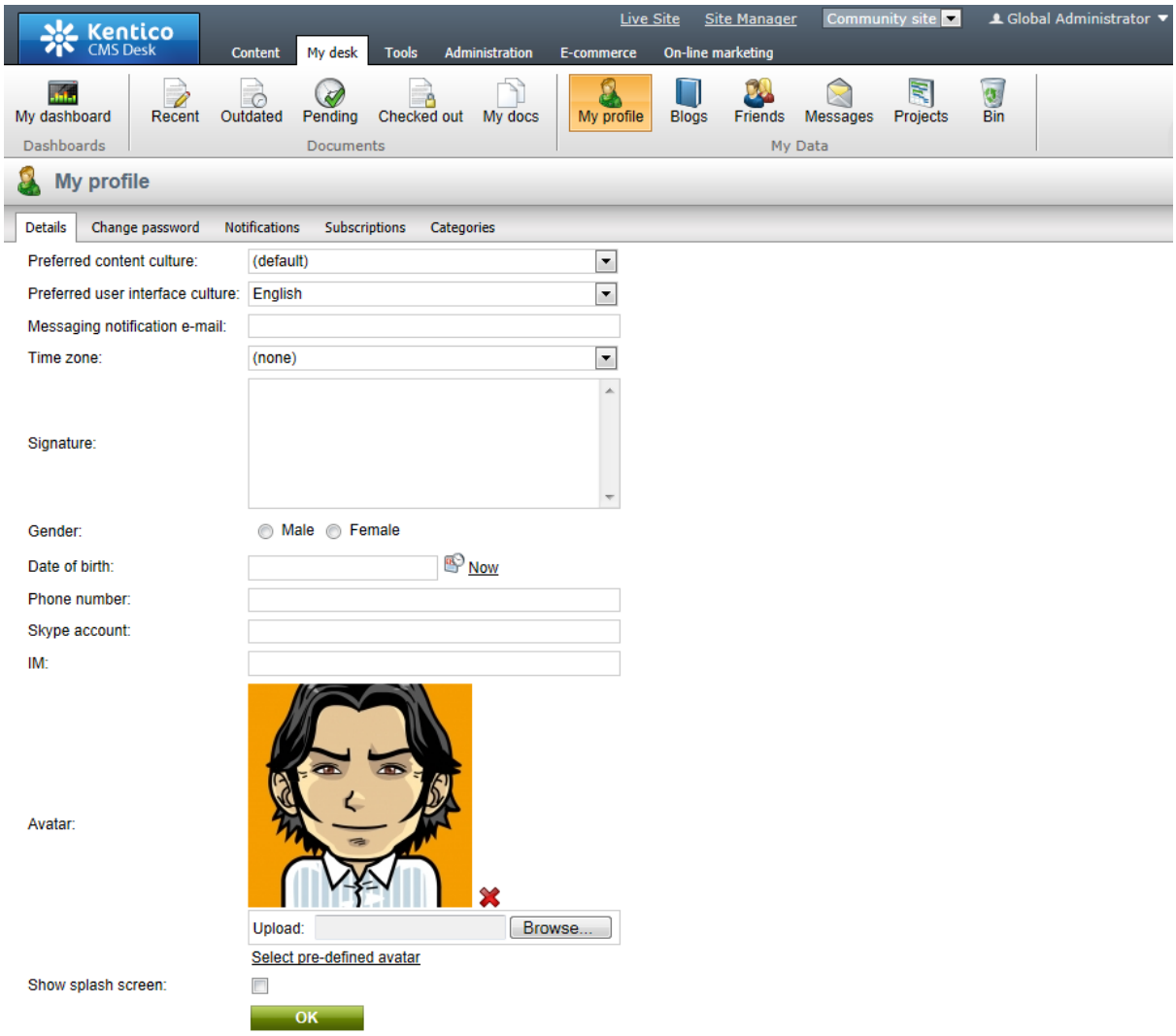
Users can **Delete** (✖) the avatar or **Upload** a custom one from a file. If predefined avatars are enabled in the site's settings, users can also click the **Select pre-defined avatar** link, which displays a gallery of predefined avatars.

You can find a live example of this on our **Community starter site**. Just log-in as one of the predefined users (e.g. *Turbo* with blank password) and click the **Edit my profile** link in the **Shortcuts** menu on the right.

Changing your user avatar in CMS Desk

Users with access to **CMS Desk** can change their avatars in **My Desk -> My profile**. It can be done in the same way as described above. You can **Delete** (✖) the avatar or **Upload** your own avatar from a file.

If shared (predefined) avatars are enabled in the site's settings, you can also click the **Select pre-defined avatar** link, which displays a gallery of pre-defined avatars from which you can easily pick one by clicking it and clicking **OK**.

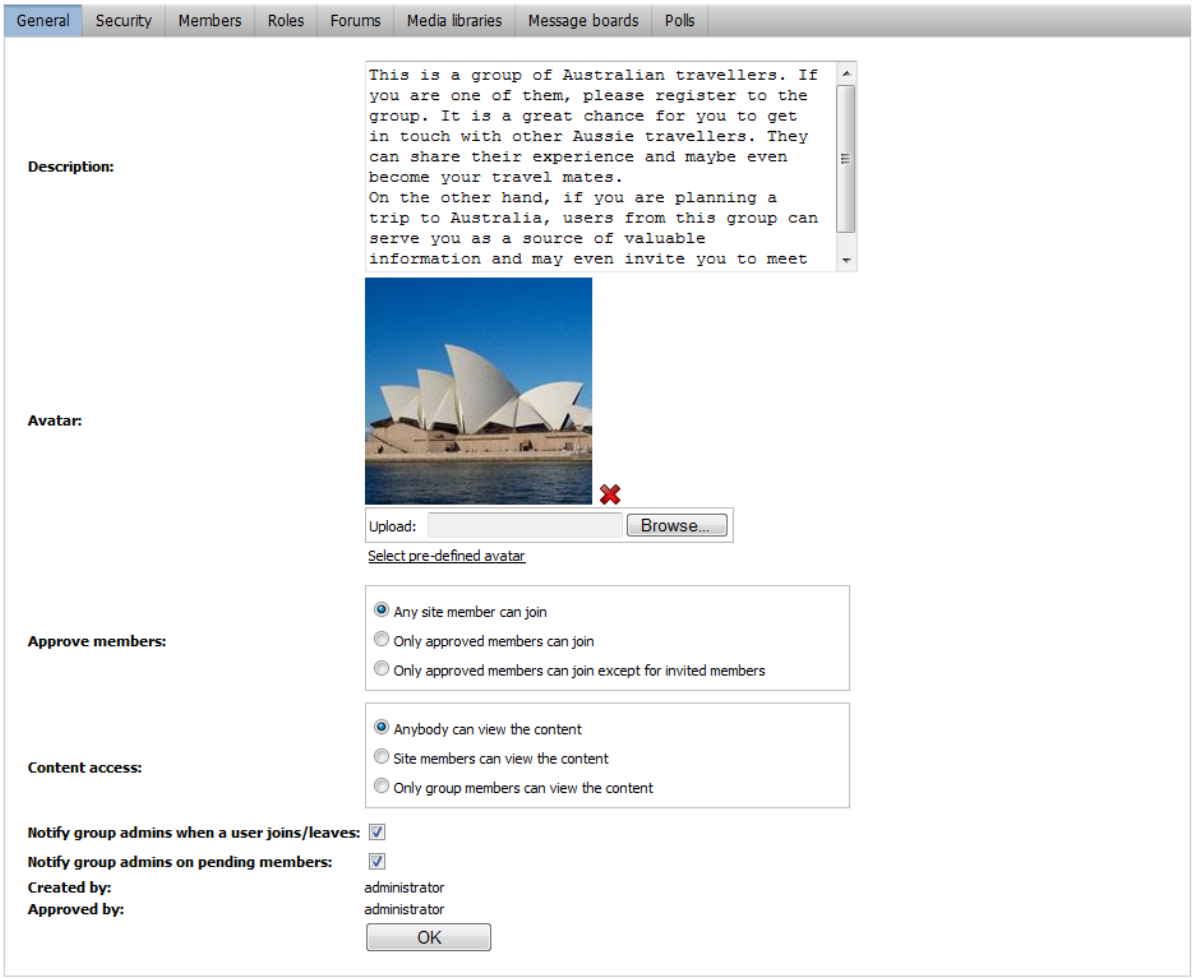


8.7.3 Changing group avatars

When a new group is created, the default avatar will be assigned to it. After that, group administrators can change the group's avatar using the **Group profile** web part.

Users can **Delete** (✖) the avatar or **Upload** a custom one from a file. If predefined avatars are enabled in site settings, users can also click the **Select pre-defined avatar** link, which displays a gallery of predefined avatars.

You can find a live example of this on the **Community starter site**. Sign-in as a group administrator (e.g. *Josh* with blank password, he is the *Australian travellers* group admin) and click **Groups** in the main menu. You should see the **Australian travellers** group under the **My groups** section. Click it and then use the **Manage the group** link in the **Shortcuts** section on the right of the page. As shown in the screenshot below, you can edit the group's avatar on the **General** tab of the management interface.

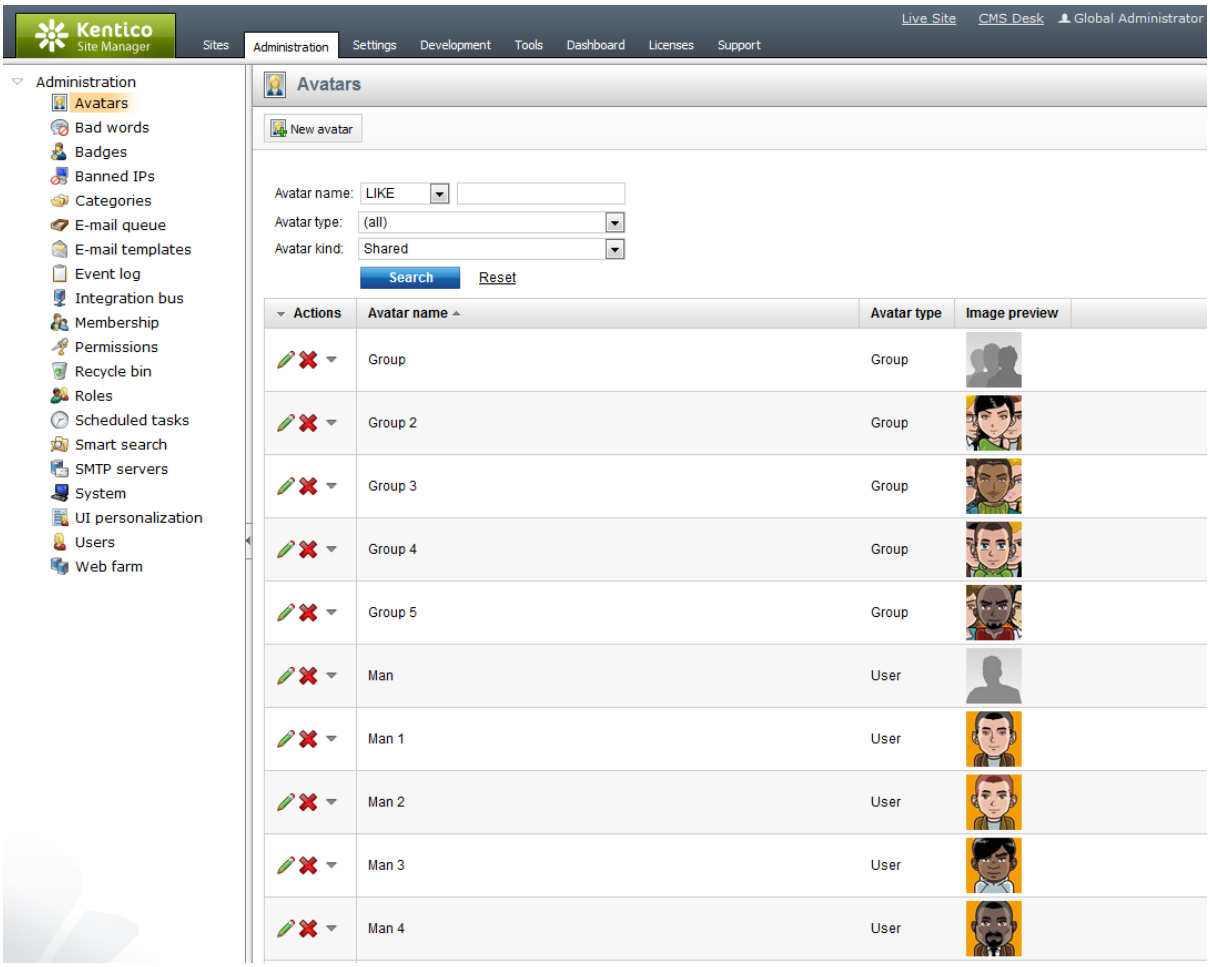


8.7.4 Managing avatars

The administration interface for managing locally stored avatars can be found in **Site Manager -> Administration -> Avatars**.

You can limit which avatars should be displayed using the filter above the list. Possible filtering parameters are **Avatar name**, **Avatar type** (user or group avatar, avatars of type *all* can be used for both) and **Avatar kind** (shared avatars are the predefined ones, while custom avatars are those that users uploaded from a file). Click **Search** to display only avatars matching the selected criteria.

You can **Edit** ( ) or **Delete** ( ) the listed avatars.



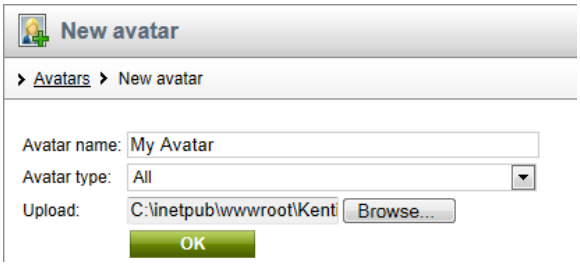
Creating predefined avatars

1. New **predefined avatars** can be created using the **New avatar** at the top part of the page. Click it.

2. You will be asked to enter the following details:

- **Avatar name** - the name of the avatar.
- **Avatar type** - choose if the avatar can be used for users, groups or both.
- **Upload** - click the **Browse** button to select an image file on your local machine.

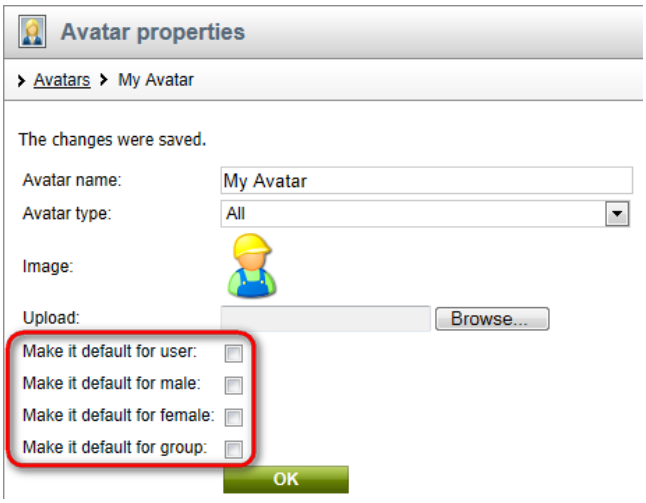
When entered, click **OK** to proceed.



3. The avatar is now created and if you go back to the list of avatars, you should see the avatar present in the list. However, you can set the following extra properties of the avatar now or any time later when editing the avatar:

- **Make it default for user** - if checked, this avatar will be the default avatar for users with an unspecified gender (including public visitors).
- **Make it default for male** - if checked, this avatar will be the default avatar for male users.
- **Make it default for female** - if checked, this avatar will be the default avatar for female users.
- **Make it default for group** - if checked, this avatar will be the default avatar for groups.

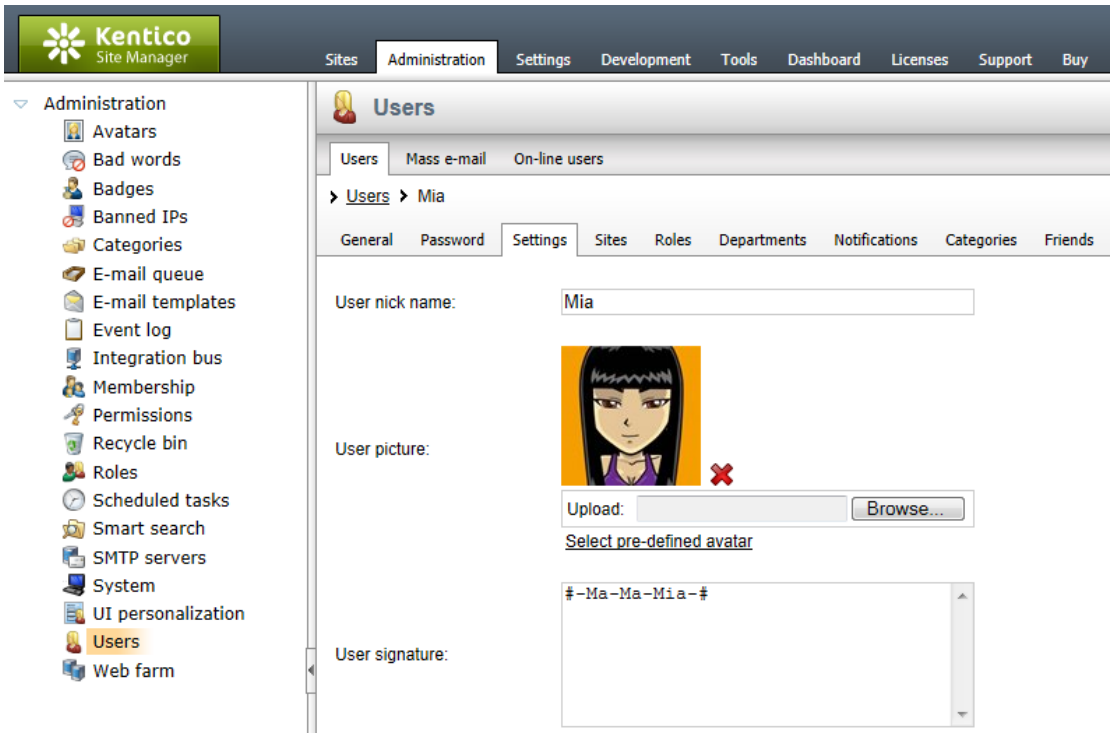
Default avatars will be assigned to new users or groups automatically when the user or group is created.



Administrating user avatars

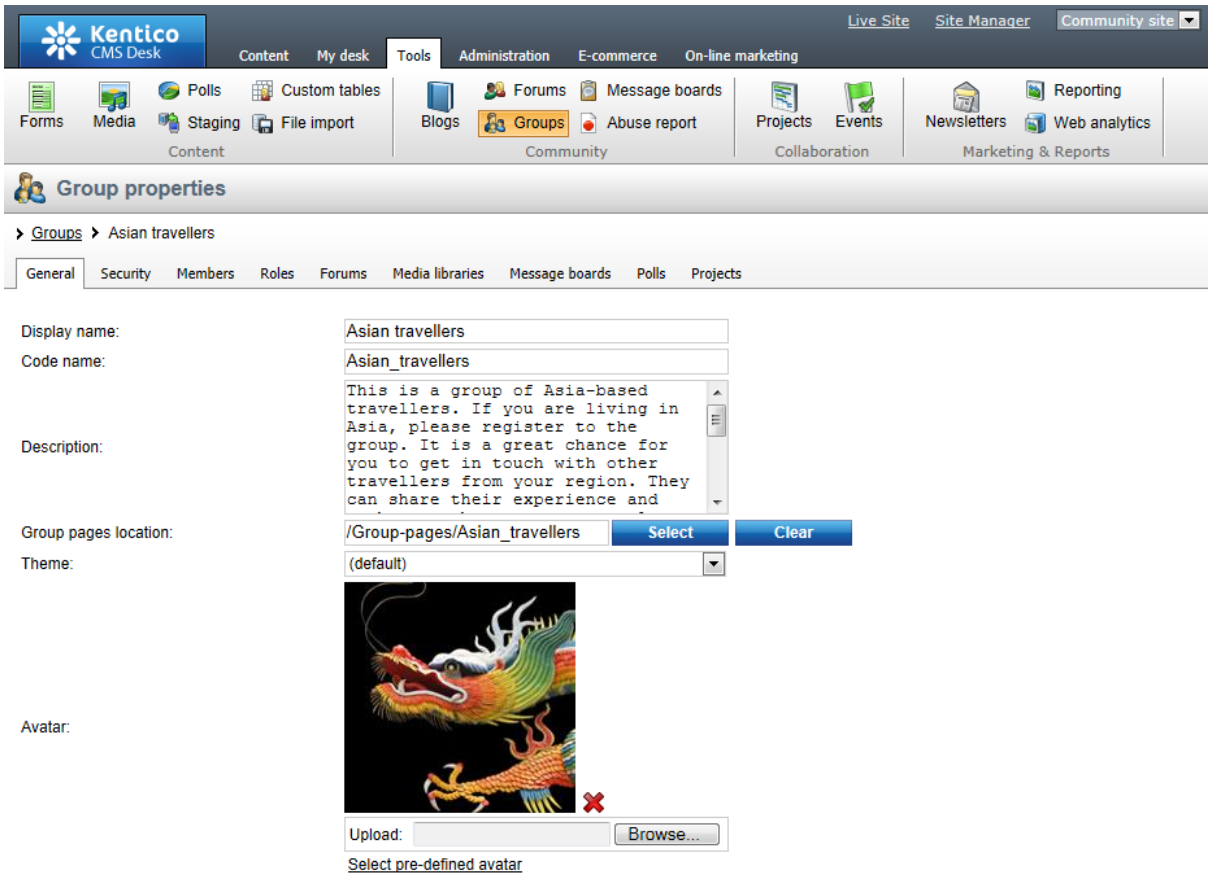
Site administrators can change the avatar of any user. If you go to **Administration -> Users** in **CMS Desk** or **Site Manager**, choose to **Edit** (🖉) a user in the list and switch to the **Settings** tab, you should see the given user's avatar in the **User picture** field as shown in the screenshot below. You can **Delete** (✖) the avatar or **Upload** your own avatar from a file.

If default avatars are enabled in site settings, you can also click the **Select pre-defined avatar** link, which displays a gallery of pre-defined avatars from which you can easily pick one by clicking it and clicking **OK**.



Changing group avatars in CMS Desk

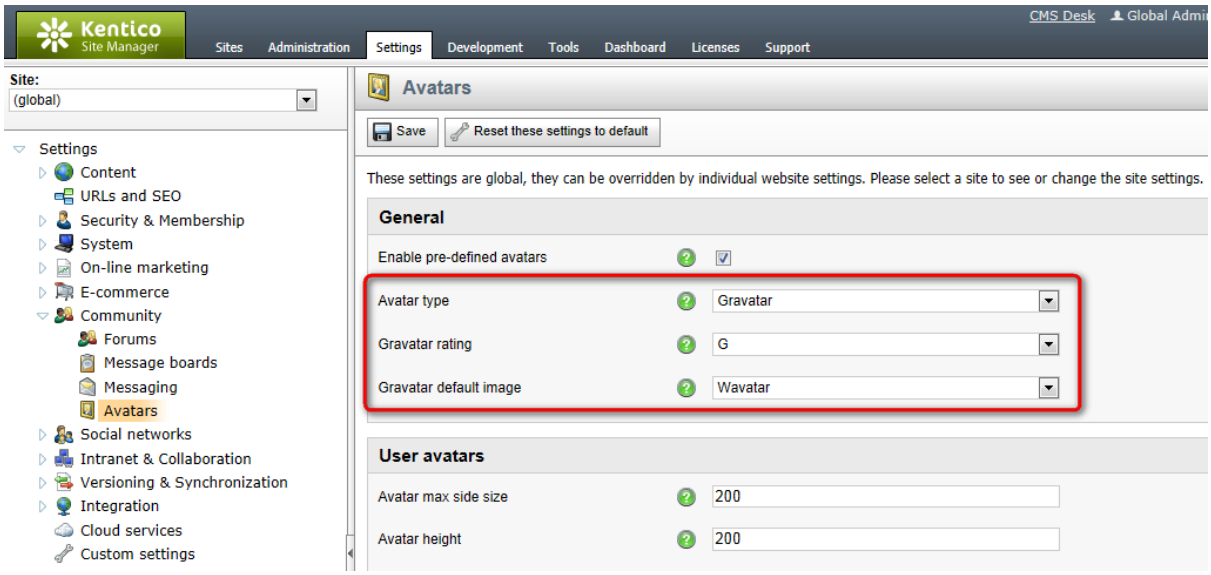
Site administrators can change the avatar of any group. If you go to **CMS Desk -> Tools -> Groups** and choose to **Edit** (🖉) some of the groups, you should see the group's avatar in the **Avatar** section, as depicted in the screenshot below. A Group avatar can be **Deleted** (✖), a new one can be **Uploaded** from a file or selected from a gallery of predefined avatars (if this is enabled in site settings).



8.7.5 Gravatars

In addition to standard locally hosted avatar images, Kentico CMS also supports Gravatars (globally recognized avatars) for users. Gravatar is an avatar hosting service (http://gravatar.com/) that allows users to register their e-mail address and upload an associated image. This image is then automatically provided as an avatar on all websites with Gravatar support where the user can be identified through the given e-mail address. With Kentico CMS sites, this address is taken from the settings of registered users. In the case of public (unauthenticated) visitors, the system checks the optional e-mail value that can be entered when leaving comments or posting in forums.

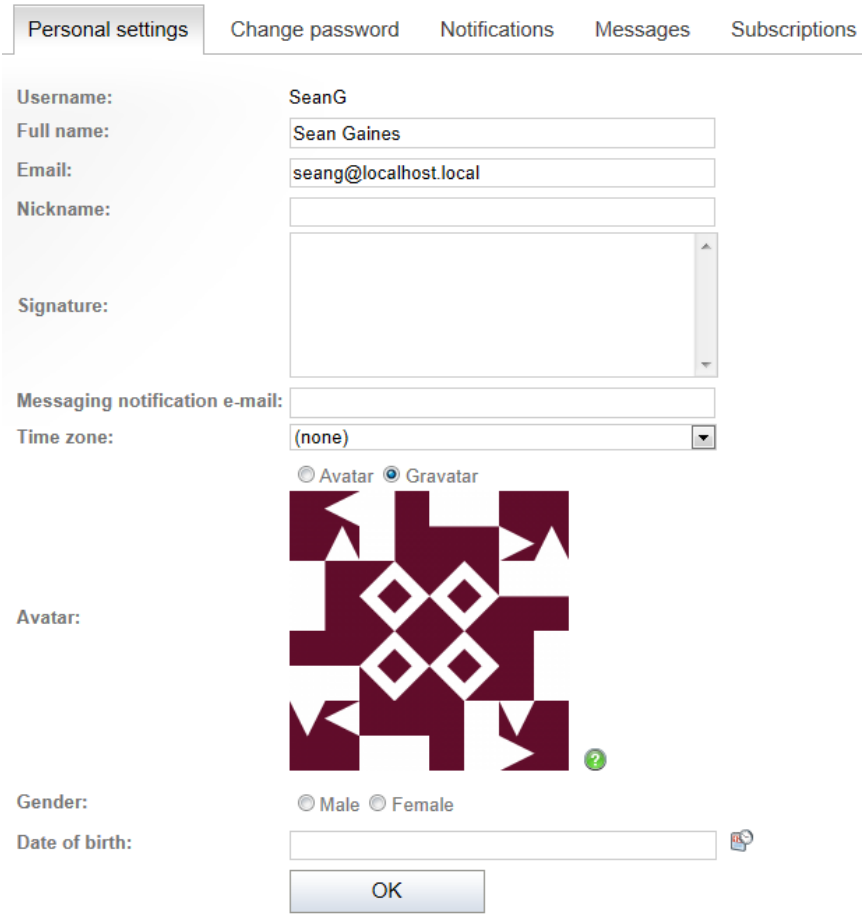
Gravatars are not enabled by default. If you wish to use them on your site, it is necessary to adjust the avatar settings in **Site Manager -> Settings -> Community -> Avatars**.



The **Avatar type** setting determines what kind of user avatars will be allowed on the website. The following options may be selected:

- **Avatar** - only the standard avatars stored locally in the website's database or file system will be available and Gravatars will not be supported. Users will be able to either upload their own image or choose a predefined one.
- **Gravatar** - with this option, Gravatars will be used for all users and visitors. When displaying avatars, the system will automatically check the e-mail addresses of users and if a matching Gravatar is found, it will be loaded and displayed. Users who do not have a Gravatar registered and public visitors without a specified e-mail address will have a default image assigned. Please note that switching to this option will change the avatars of users who have uploaded a custom image locally for the website. Enabling Gravatars has no effect on group avatars.
- **User can choose** - selecting this value allows both possible avatar options. Registered users will be able to choose whether they want to use a Gravatar or a local avatar. Gravatar support will be enabled for public users.

When using Gravatars, users will only be able to view the avatar image in their profile, with no option of changing it locally. If the **User can choose** option is selected, it will be possible for users to switch between the two modes by selecting the appropriate radio button as shown below.



All Gravatar images are rated according to the maturity level of their content. You can set the maximum rating that should be allowed on your website by configuring the **Gravatar rating** setting. The following ratings are available:

- **G** - suitable for display on all websites with any audience type.
- **PG** - may contain rude gestures, provocatively dressed individuals, the lesser swear words, or mild violence.
- **R** - may contain such things as harsh profanity, intense violence, nudity, or hard drug use.
- **X** - may contain hardcore sexual imagery or extremely disturbing violence.

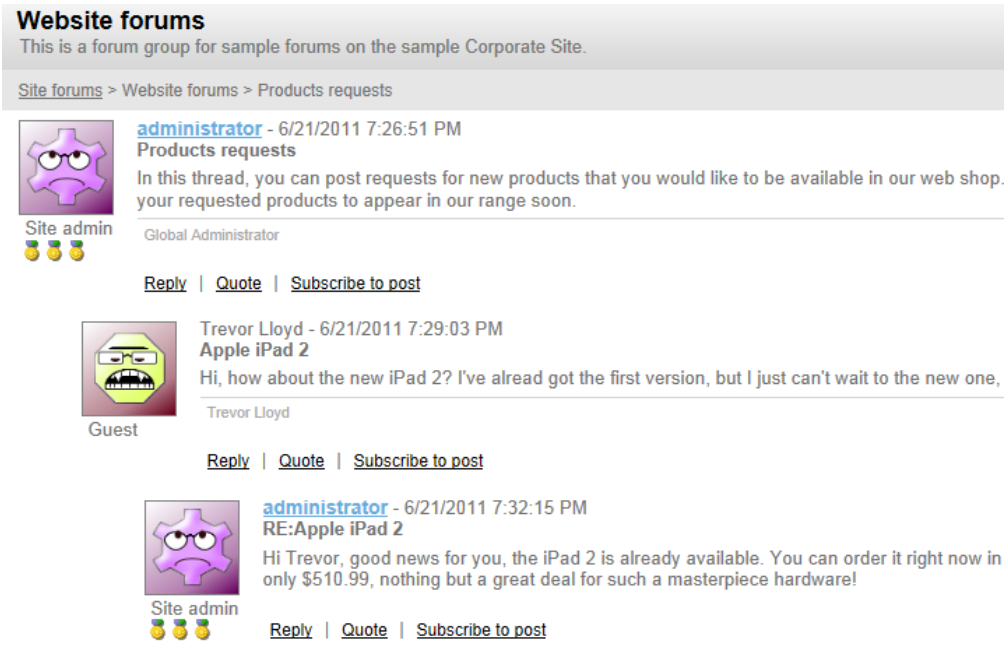
If a user's Gravatar does not meet the site's rating requirements, it is replaced by the default image.

Gravatars are user-rated

Please note that the rating of a Gravatar image is entered by its owner. Because of this, Kentico CMS cannot guarantee that inappropriate images will be filtered out if their rating is inaccurate.

The **Gravatar default image** setting determines what type of image should be displayed as the avatar of users who do not have a valid Gravatar registered for their e-mail address. If the *(none)* option is selected, the local avatar image specified as default for users in **Site Manager -> Administration ->**

Avatars is used. The remaining options provide various types of default Gravatar image sets. In most cases, a different image is generated based on the e-mail address of the given user, so even users without a Gravatar account will have a unique avatar image assigned.



Displaying Gravatars in transformations

If you wish to display Gravatars in member lists or on any other type of pages where user avatars are presented, it may be necessary to modify your transformations.

Not all transformation methods used to generate avatar image tags support displaying of Gravatars. Please read the Displaying avatars in transformations topic for more information.

8.7.6 General avatar settings

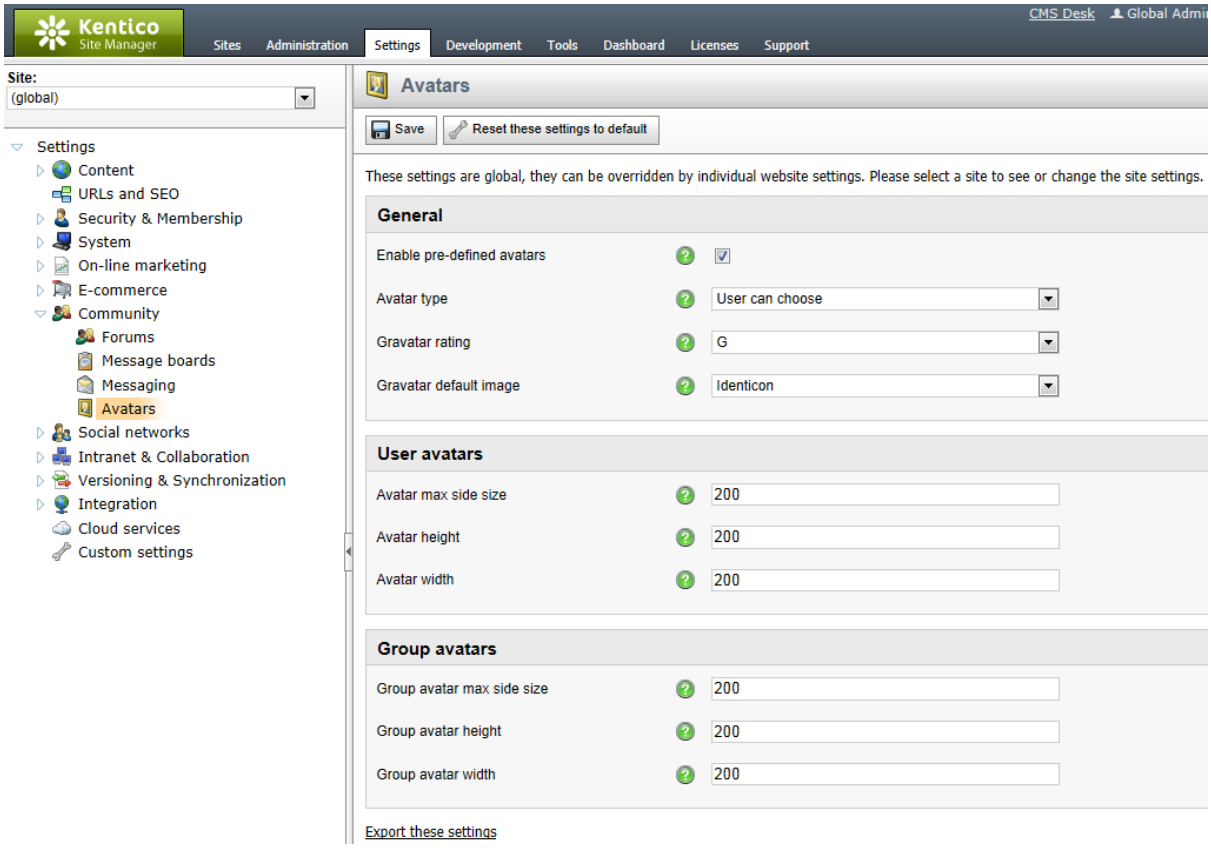
In addition to configuring Gravatar support, the settings in **Site Manager -> Settings -> Community -> Avatars** may also be used to adjust other preferences, such as automatic resizing of avatar images. The following general settings are available:

- **Enable pre-defined avatars** - if checked, it will be possible to select one of the shared, predefined avatars when choosing a user or group's avatar image. If disabled, only custom uploaded avatars will be allowed. This setting does not affect Gravatars.

- **Avatar max side size** - sets the maximum allowed size of *user* avatar images. If one or both dimensions of the uploaded image are larger, it will be resized so that the larger dimension matches the entered value. If *0* is entered, the *Avatar height* and *Avatar width* settings will be used instead.
- **Avatar height** - if the *Avatar max side size* setting is set to *0*, avatar images will be resized to this height.
- **Avatar width** - if the *Avatar max side size* setting is set to *0*, avatar images will be resized to this

width.

- **Group avatar max side size** - sets the maximum allowed size of *group* avatar images. If one or both dimensions of the uploaded image are larger, it will be resized so that the larger dimension matches the entered value. If *0* is entered, the *Group avatar height* and *Group avatar width* settings will be used instead.
- **Group avatar height** - if the *Group avatar max side size* setting is set to 0, images will be resized to this height.
- **Group avatar width** - if the *Group avatar max side size* setting is set to 0, images will be resized to this width.



8.7.7 Displaying avatars in transformations

User avatars

User avatars can be displayed in transformations by using the *<%# GetUserAvatarImage(...) %>* method. It can be called with four different sets of parameters as described below (including examples):

- **GetUserAvatarImage(int maxSideSize, string alt)**

```
<%# GetUserAvatarImage(50, HTMLEncode(GetNotEmpty("UserNickname;UserName"))) %>
```

This returns an image tag displaying the avatar contained in the *AvatarGuid* field of the currently

transformed user, with a maximum *sidesize* of 50 px. and *Alt tag* equal to the user's *Nickname* or *Username.* The *AvatarGuid* field must be accessible for this method to work.

This overload does not support Gravatars!

The *GetUserAvatarImage* method cannot display Gravatars when called with only these two parameters. Please use one of the overloads that includes the **userID** parameter for this purpose.

- **GetUserAvatarImage(object userID, int maxSideSize, int width, int height, object alt)**

```
<%# GetUserAvatarImage(Eval("UserID"), 200, 0, 0, Eval("UserName")) %>
```

This returns an image tag displaying the current avatar of the user with ID *UserID*, with a maximum *sidesize* of 200 px. and an *Alt* tag equal to the user's *Username*.

- **GetUserAvatarImage(object avatarID, object userID, int maxSideSize, object alt)**

```
<%# GetUserAvatarImage(Eval("UserAvatarID"), Eval("UserID"), 50, Eval("UserName"))
%>
```

This returns an image tag displaying the avatar with ID *UserAvatarID*, of user with ID *UserID*, with a maximum *sidesize* of 50 px. and *Alt tag* equal to the user's *Username*.

- **GetUserAvatarImage(object avatarID, object userID, int maxSideSize, int width, int height, object alt)**

```
<%# GetUserAvatarImage(Eval("UserAvatarID"), Eval("UserID"), 0, 40, 45, Eval
("UserName")) %>
```

This returns an image tag displaying the avatar with ID *UserAvatarID*, of user with ID *UserID*, with a maximum *width* of 40 px., maximum *height* of 45 px. and *Alt tag* equal to the user's *Username*.

Group avatars

Group avatars can by displayed using the same approach, with the difference that the *<%# GetGroupAvatarImage(...) %>* method is called instead:

- **GetGroupAvatarImage(int maxSideSize, object alt)**

```
<%# GetGroupAvatarImage(50, Eval("GroupDisplayName", true)) %>
```

This returns an image tag displaying the group avatar specified by the *AvatarGuid* field of the currently transformed group, with a maximum *sidesize* of 50 px. and *Alt tag* equal to the group's *Display name.*

The *AvatarGuid* field must be accessible for this method to work.

- **GetGroupAvatarImage(object avatarID, int maxSideSize, object alt)**

```
<%# GetGroupAvatarImage(Eval("GroupAvatarID"), 50, Eval("GroupDisplayName", true))
%>
```

This returns an image tag displaying the group avatar with ID *GroupAvatarID*, with a maximum *sidesize* of 50 px. and *Alt tag* equal to the group's *Display name*.

- **GetGroupAvatarImage(object avatarID, int maxSideSize, int width, int height, object alt)**

```
<%# GetGroupAvatarImage(Eval("GroupAvatarID"), 0, 40, 45, Eval("GroupDisplayName",
true)) %>
```

This returns an image tag displaying the group avatar with ID *GroupAvatarID*, with a maximum *width* of 40 px., maximum *height* of 45 px. and *Alt tag* equal to the group's *Display name*.

Image tag

The following is an example of an image tag generated by the *<%# GetUserAvatarImage(...) %>* method. As you can see, the method uses the **~/CMSModules/Avatars/CMSPages/GetAvatar.aspx** page to get the source image.

```
<img src="/KenticoCMS/CMSModules/Avatars/CMSPages/GetAvatar.aspx?
maxsidesize=50&amp;avatarguid=42f9d97e-b0c8-43a2-8a35-62dfec5ae64a" alt="Andy"
class="AvatarImage" />
```

When displaying Gravatars, the image is retrieved from the http://www.gravatar.com/ website as shown in the example below:

```
<img src="http://www.gravatar.com/avatar/9b20d3ee7a6ba98550babc044da91bfb?
s=80&amp;r=g&amp;d=wavatar" alt="User avatar" class="AvatarImage" />
```

8.7.8 Avatars internals and API

8.7.8.1 Overview

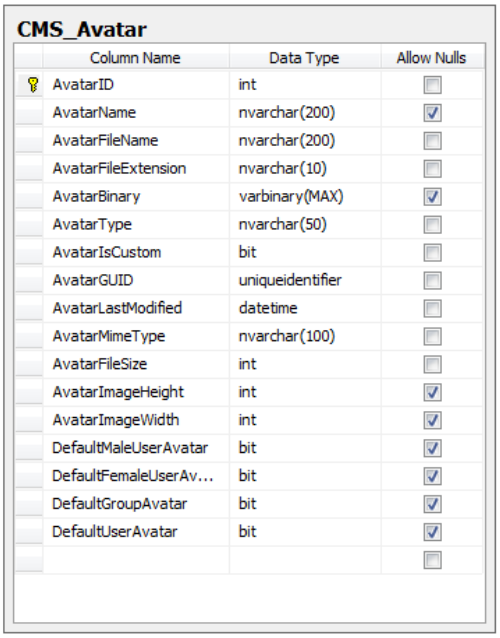
In this chapter, you will learn which database tables and API classes are used in the Avatars module. You will also see the most common API examples.

Further API-related information and examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all members of the module's classes, please refer to Kentico CMS API Reference.

8.7.8.2 Database tables

The following database table is used to store information about avatars:

- **CMS_Avatar** - contains records representing avatars and their configuration. Avatar images are stored in binary format.



8.7.8.3 API classes

If you are not familiar with database table data management through Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

The classes of the Avatars module can be found in the **CMS.SiteProvider** namespace.

CMS_Avatar table API:

- **AvatarInfo** - represents one avatar object.
- **AvatarInfoProvider** - provides management functionality for avatars.

8.7.8.4 API examples

8.7.8.4.1 Overview

These topics show examples of how the API of the Avatars module can be used:

- Managing avatars
- Managing user avatars

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Community\Avatars\Default.aspx.cs***.

The avatar API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.GlobalHelper;
using CMS.CMSHelper;
using CMS.SiteProvider;
```

8.7.8.4.2 Managing avatars

The following example creates an avatar.

```
private void CreateAvatar()
{
    // Create new avatar object
    AvatarInfo newAvatar = new AvatarInfo(Server.MapPath("~\\CMSAPIExamples\\Code\
\Community\\Avatars\\Files\\avatar_man.jpg"));

    // Set the properties
    newAvatar.AvatarName = "MyNewAvatar";
    newAvatar.AvatarType = AvatarInfoProvider.GetAvatarTypeString
(AvatarTypeEnum.All);
    newAvatar.AvatarIsCustom = false;

    // Save the avatar
    AvatarInfoProvider.SetAvatarInfo(newAvatar);
}
```

The following example gets and updates an avatar.

```
private bool GetAndUpdateAvatar()
{
    // Get the avatar
    AvatarInfo updateAvatar = AvatarInfoProvider.GetAvatarInfo("MyNewAvatar");
    if (updateAvatar != null)
    {
        // Update the properties
```

```
        updateAvatar.AvatarName = updateAvatar.AvatarName.ToLower();

        // Save the changes
        AvatarInfoProvider.SetAvatarInfo(updateAvatar);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates avatars.

```
private bool GetAndBulkUpdateAvatars()
{
    // Prepare the parameters
    string where = "AvatarName LIKE N'MyNewAvatar%'";

    // Get the data
    DataSet avatars = AvatarInfoProvider.GetAvatars(where, null);
    if (!DataHelper.DataSourceIsEmpty(avatars))
    {
        // Loop through the individual items
        foreach (DataRow avatarDr in avatars.Tables[0].Rows)
        {
            // Create object from DataRow
            AvatarInfo modifyAvatar = new AvatarInfo(avatarDr);

            // Update the properties
            modifyAvatar.AvatarName = modifyAvatar.AvatarName.ToUpper();

            // Save the changes
            AvatarInfoProvider.SetAvatarInfo(modifyAvatar);
        }

        return true;
    }

    return false;
}
```

The following example deletes an avatar.

```
private bool DeleteAvatar()
{
    // Get the avatar
    AvatarInfo deleteAvatar = AvatarInfoProvider.GetAvatarInfo("MyNewAvatar");

    // Delete the avatar
    AvatarInfoProvider.DeleteAvatarInfo(deleteAvatar);
```

```
    return (deleteAvatar != null);
}
```

8.7.8.4.3 Managing user avatars

The following example assigns an avatar to a specific user.

```
private bool AddAvatarToUser()
{
    // Get the avatar and user objects
    AvatarInfo avatar = AvatarInfoProvider.GetAvatarInfo("MyNewAvatar");
    UserInfo user = UserInfoProvider.GetUserInfo(CMSContext.CurrentUser.UserName);

    if ((avatar != null) && (user != null))
    {
        user.UserAvatarID = avatar.AvatarID;

        // Save edited object
        UserInfoProvider.SetUserInfo(user);

        return true;
    }

    return false;
}
```

The following example removes a user's avatar.

```
private bool RemoveAvatarFromUser()
{
    // Get the user
    UserInfo user = UserInfoProvider.GetUserInfo(CMSContext.CurrentUser.UserName);

    if (user != null)
    {
        user.UserAvatarID = 0;

        // Save edited object
        UserInfoProvider.SetUserInfo(user);

        return true;
    }

    return false;
}
```

8.8 Bad words

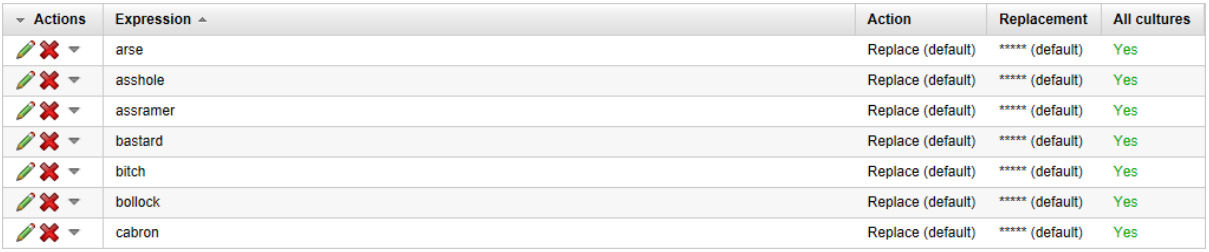
8.8.1 Overview

The Bad words module can be used as a filter for unwanted input from website users. This can be anything from rude language to spam or links to illegal content, basically anything that is detectable by presence of some keyword in input text.

The Bad words module is capable of filtering input submitted via the following modules:

- Forums
- Blogs
- Messaging
- Message boards
- Chat

For the module to work, you first need to enable it as described in Enabling the module. Filtering is then performed based on keywords - so called bad words - defined by site administrators in **Site manager -> Administration -> Bad words**. The procedure of defining bad words is described in the Defining a bad word topic.



Actions	Expression	Action	Replacement	All cultures
	arse	Replace (default)	***** (default)	Yes
	asshole	Replace (default)	***** (default)	Yes
	assramer	Replace (default)	***** (default)	Yes
	bastard	Replace (default)	***** (default)	Yes
	bitch	Replace (default)	***** (default)	Yes
	bollock	Replace (default)	***** (default)	Yes
	cabron	Replace (default)	***** (default)	Yes

If a user inputs some text containing one or more of these words, a predefined action is performed. Available actions range from removing the word, replacing it with a pre-defined string or just reporting abuse to website administrators by means of the Abuse report module. All actions and their effects are described in the Possible actions topic.

By default, input from all users except global administrators is filtered. However, you may grant the **Use bad words** permission to some roles, which will disable filtering of input submitted by its members. More information on this option can be found in the Security topic.

The Bad words internals and API sub-chapter provides information about the database tables and classes used by the module, as well as examples of how bad words can be managed and bad word checks performed using Kentico CMS API.

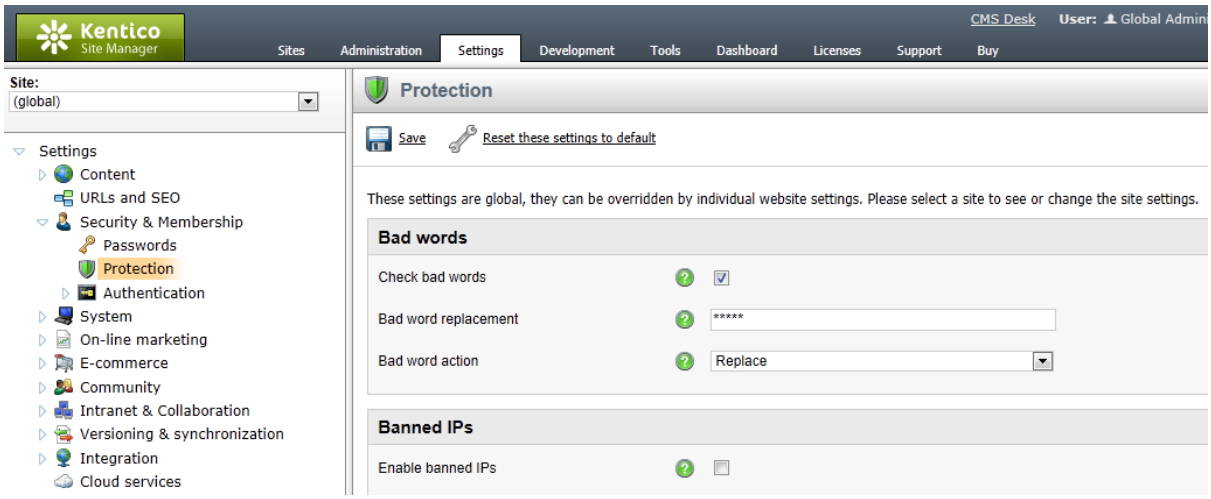
The Bad words module is just one of the modules which can help you deal with unwanted user input. Besides detecting unwanted input based on keywords, you may also let your site visitors report such content via the Abuse report module. The last resort in dealing with unwanted content may be blocking access to your site from a certain IP address. This can be achieved using the Banned IPs module.

8.8.2 Enabling the module

For the module to be functional, you first have to go to **Site Manager -> Settings & Membership -> Protection** and enable the **Check bad words** option.

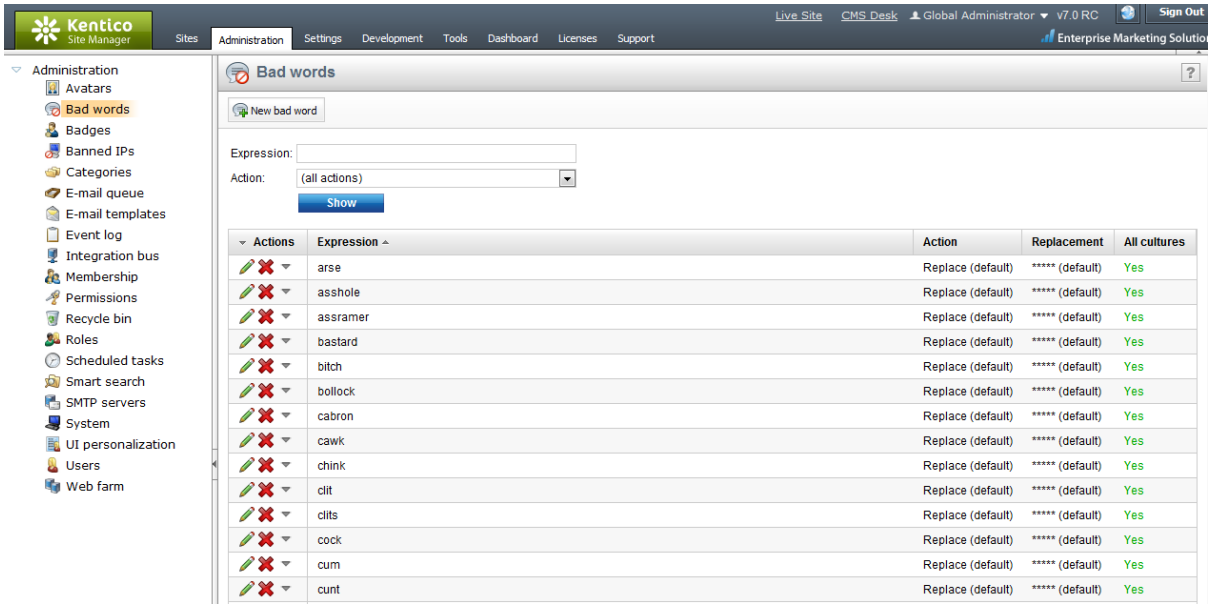
On this page, you can also define the default action that will be assigned to newly created bad words if the *Use default settings* option is enabled in the bad word creation dialog (see Defining a bad word for more details). The default action can be selected from the **Bad word action** drop-down list.

Default replacement string can be entered into the **Bad word replacement** field. It will be used in case that a bad word is detected, the *Replace* action should be performed on its detection, but has no replacement text defined (i.e. has the *Use default settings* option enabled for the *Replace by* field).



8.8.3 Defining a bad word

The Bad words module's user interface is located in **Site Manager -> Administration -> Bad words**. This is where all existing bad words are listed and where new bad words can be defined.



The top part of the page is a filter. Using it, you can display only those bad words that match the specified criteria. You can filter by the **Expression** and by the **Action** that should be taken on its detection. After specifying the filtering criteria and clicking the **Show** button, only those items that

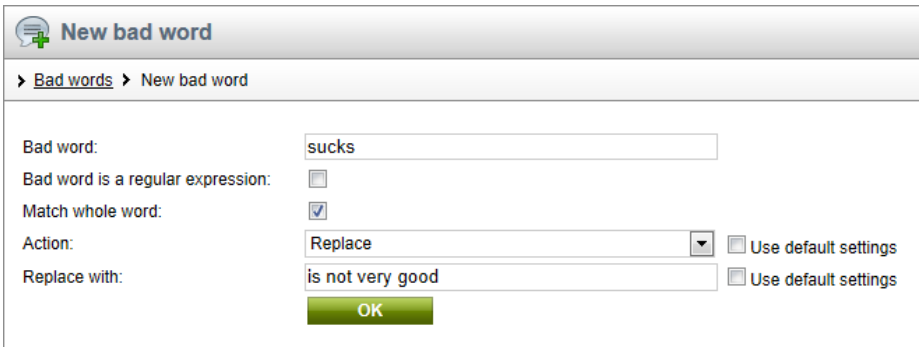
match the specified criteria will be displayed in the list.

Adding a new bad word

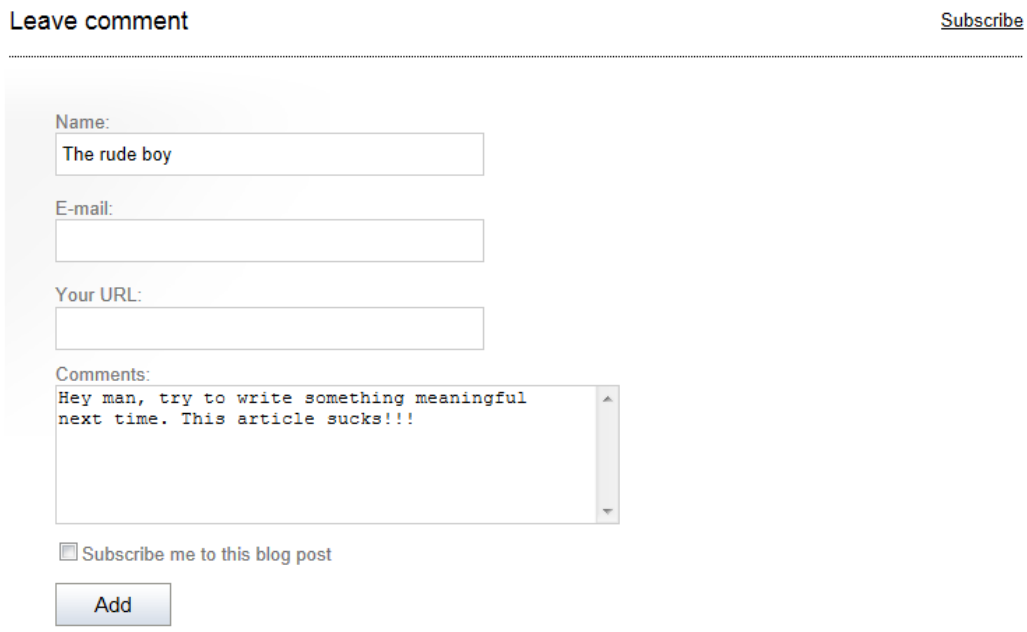
1. To add a new bad word, click **New bad word** ( ) at the top of the page. You will be redirected to the **New bad word** dialog. The following details can be entered:

- **Bad word** - the string that should not appear in input text.
- **Bad word is a regular expression** - if checked, the string entered into the previous field will searched as a regular expression.
- **Match whole word** - if enabled, only whole word occurrences of the expression will be identified. If disabled, even words with substrings matching the expression will be identified (e.g. an occurrence of the word "document" would be reported if "cum" was defined as a bad word with this option disabled).
- **Action** - action that will be taken in case that the bad word is detected. See the Possible actions topic for more details.
  Use default settings - if enabled, global value will be used as set in *Site Manager -> Settings -> Security & Protection -> Bad words -> Bad word action*.
- **Replace with** - in case the *Replace* action is selected, the substitute for the bad word is defined here.
  Use default settings - if enabled, global value will be used as set in *Site Manager -> Settings -> Security & Protection -> Bad words -> Bad word replacement*.

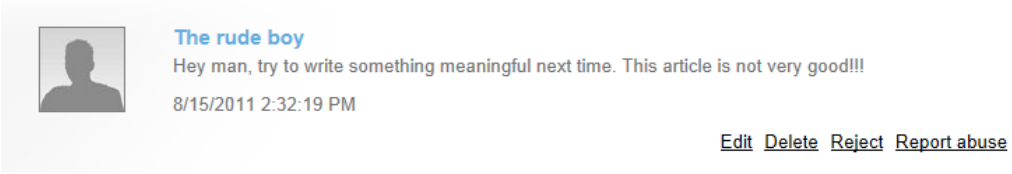
Fill in the required details and click **OK**.



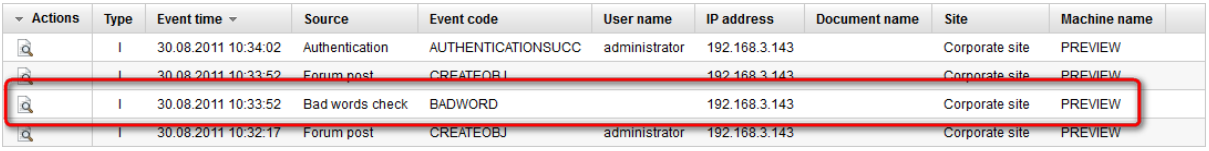
2. Let's try the functionality now. Go to the live Corporate Site, enter the Blogs section and open some of the blog posts. Enter a comment as in the following screenshot and click **Add**.



3. As you can see, the last word has been replaced with its polite substitute that has been defined earlier.



4. You can also check the **Event log** in **Site Manager -> Administration**. An event is always logged automatically when user input containing a bad word is detected. The **Event code** is **BADWORD** in such case.



Actions	Type	Event time	Source	Event code	User name	IP address	Document name	Site	Machine name
	I	30.08.2011 10:34:02	Authentication	AUTHENTICATIONSUCC	administrator	192.168.3.143		Corporate site	PREVIEW
	I	30.08.2011 10:33:52	Forum post	CREATEOBJ		192.168.3.143		Corporate site	PREVIEW
	I	30.08.2011 10:33:52	Bad words check	BADWORD		192.168.3.143		Corporate site	PREVIEW
	I	30.08.2011 10:32:17	Forum post	CREATEOBJ	administrator	192.168.3.143		Corporate site	PREVIEW

8.8.4 Possible actions

In case that a bad word is detected, one of the following actions can be taken:

- **Remove** - the bad word is removed from the entered text with no substitution.
- **Replace** - the bad word is removed and replaced with a predefined string.
- **Report abuse** - a report is created in **CMSDesk -> Tools -> Abuse report**.
- **Request moderation** - the post is submitted to approval before being published; this happens even in case that the forum, message board or blog is not moderated by default.
- **Deny** - a warning message will be displayed to the user when they try to post the inadequate text, listing which words should be removed.

The following table shows which actions are available for each of the supported modules:

	Blog comments	Forums	Messaging	Message boards
Remove	•	•	•	•
Replace	•	•	•	•
Report abuse	•	•		•
Request moderation	•	•		•
Deny	•	•	•	•

In case that there is **more than one bad word detected** in input text and the words have **different actions** set, the actions will be taken according to their hierarchy. *Remove*, *Replace* and *Report abuse* actions are independent, i.e. the actions can be taken simultaneously. *Request moderation* is stronger than *Report abuse* and *Deny* is the strongest of all.

The following list shows which actions will be taken in some specific cases in order for you to easily understand the actions hierarchy:

- **Remove and Replace** - the first one will be removed and the second one will be replaced with the defined substitute.
- **Remove, Replace and Report abuse** - the first one will be removed, the second one replaced and an abuse report will be logged.
- **Remove, Replace and Request moderation** - the first one will be removed, the second one will be replaced and the post will have to be approved.
- **Report abuse and Request moderation** - the post will have to be approved and no abuse report will be logged.
- **Deny and any other** - the text is left as it is, but produces an error message.

Replace action in fields with limited length

If the **Replace** action has to be performed in a text entered into a field with limited length, it may happen that the text with replaced bad words would exceed the maximal length (when the replacement text is longer than the original bad word). In this case, the bad word is replaced with as many asterisks as its length.

When there are multiple bad words in the text, replacement starts from the beginning of the string. If replacement of the first bad word wouldn't cause exceeding of the maximal length, it is replaced with the replacement text. The same applies to the subsequent bad words and asterisks start to be used with the first bad word whose replacement would cause exceeding of the maximal length.

8.8.5 Multilingual support

You can define in which cultures will a certain bad word be filtered. If you choose to **Edit** () a bad word in the list in **Site Manager -> Administration -> Bad words** and switch to its **Cultures** tab (the tab is not available in the **New bad word** dialog), you will be offered with the following two options:

- **The word is not allowed in all cultures** - the bad word will be filtered in all website cultures
- **The word is not allowed only in following cultures** - the bad word will be filtered only in cultures added to the list below

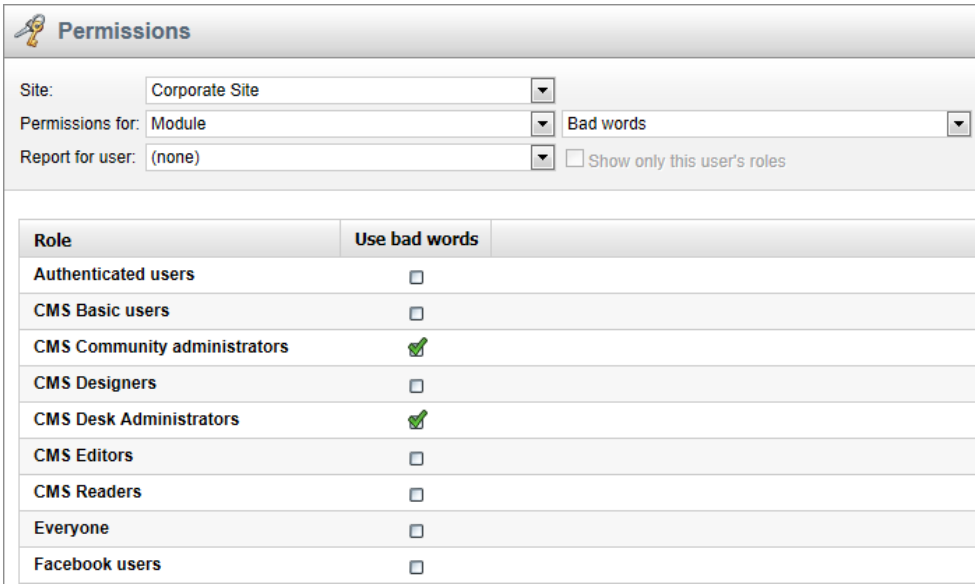
Using the **Add cultures** button, you can add cultures to the list. The **Remove selected button** removes all cultures selected by the check-boxes from the list.

8.8.6 Security

The Bad words module has only one permission to grant to roles:

- **Use bad words** - allows members of the role to use bad words, i.e. bad words filtering will not be performed to submissions made by members of the role

This permission can be granted to particular roles in **Site Manager -> Administration -> Permissions**, after selecting the **Modules -> Bad words** permission matrix.



8.8.7 Bad words internals and API

8.8.7.1 Overview

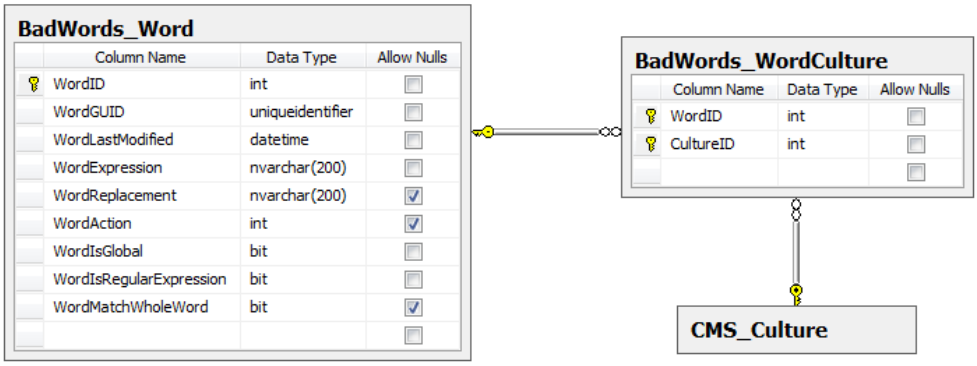
In this chapter, you will learn which database tables and API classes are used by the Bad words module. You will also see the most common API examples.

Advanced API examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all methods from the module classes, please refer to Kentico CMS API Reference.

8.8.7.2 Database tables

The Bad words module uses the following database tables:

- **BadWords_Word** - contains records representing defined bad words.
- **BadWords_WordCulture** - contains relationships between bad words and cultures. Each record indicates that the given bad word should be checked in the respective culture.



8.8.7.3 API classes

If you are not familiar with database table data management through Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

The classes of the Bad words module can be found in the **CMS.SiteProvider** namespace.

BadWords_Word table API:

- **BadWordInfo** - represents one bad word object.
- **BadWordInfoProvider** - provides management functionality for bad words.
- **BadWordsHelper** - provides management functionality for bad words. Compared to *BadWordInfoProvider*, this class provides extra methods for bad word actions and replacements.

Others:

- **BadWordActionEnum** - enumeration of actions that can be taken on detection of a bad word in checked text.

8.8.7.4 API examples

8.8.7.4.1 Overview

These topics show examples of how the API of the Bad words module can be used:

- Managing bad words
- Performing bad word checks

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Administration\BadWords\Default.aspx.cs***.

The Bad words API examples use the following namespaces:

```
using System;
using System.Data;
using System.Collections;

using CMS.GlobalHelper;
using CMS.UIControls;
using CMS.SiteProvider;
```

8.8.7.4.2 Managing bad w ords

The following example creates a bad word.

```
private bool CreateBadWord()
{
    // Create new bad word object
    BadWordInfo newWord = new BadWordInfo();

    // Set the properties
    newWord.WordExpression = "testbadword";
    newWord.WordAction = BadWordActionEnum.ReportAbuse;
    newWord.WordIsGlobal = true;
    newWord.WordIsRegularExpression = false;

    // Save the bad word
```

```
    BadWordInfoProvider.SetBadWordInfo(newWord);

    return true;
}
```

The following example gets and updates the bad word created by the code example above.

```
private bool GetAndUpdateBadWord()
{
    // Prepare the parameters
    string where = "[WordExpression] = 'testbadword'";

    // Get the data
    DataSet words = BadWordInfoProvider.GetBadWords(where, null);

    if (!DataHelper.DataSourceIsEmpty(words))
    {
        // Get the bad word's data row
        DataRow wordDr = words.Tables[0].Rows[0];

        // Create object from DataRow
        BadWordInfo modifyWord = new BadWordInfo(wordDr);

        // Update the properties
        modifyWord.WordAction = BadWordActionEnum.Replace;
        modifyWord.WordReplacement = "testpoliteword";

        // Save the changes
        BadWordInfoProvider.SetBadWordInfo(modifyWord);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates multiple bad words selected from database based on a WHERE condition.

```
private bool GetAndBulkUpdateBadWords()
{
    // Prepare the parameters
    string where = "[WordExpression] = 'testbadword'";

    // Get the data
    DataSet words = BadWordInfoProvider.GetBadWords(where, null);
    if (!DataHelper.DataSourceIsEmpty(words))
    {
        // Loop through the individual items
        foreach (DataRow wordDr in words.Tables[0].Rows)
```

```
        {
            // Create object from DataRow
            BadWordInfo modifyWord = new BadWordInfo(wordDr);

            // Update the properties
            modifyWord.WordAction = BadWordActionEnum.Replace;
            modifyWord.WordReplacement = "testpoliteword";

            // Save the changes
            BadWordInfoProvider.SetBadWordInfo(modifyWord);
        }

        return true;
    }

    return false;
}
```

The following example deletes the bad word created by the first code example on this page.

```
private bool DeleteBadWord()
{
    // Prepare the parameters
    string where = "[WordExpression] = 'testbadword' ";

    // Get the data
    DataSet words = BadWordInfoProvider.GetBadWords(where, null);

    if (!DataHelper.DataSourceIsEmpty(words))
    {
        foreach (DataRow wordDr in words.Tables[0].Rows)
        {
            // Create object from DataRow
            BadWordInfo deleteWord = new BadWordInfo(wordDr);

            // Delete the bad word
            BadWordInfoProvider.DeleteBadWordInfo(deleteWord);
        }

        return true;
    }

    return false;
}
```

8.8.7.4.3 Performing bad word checks

The following example checks the declared custom string for presence of a single bad word.

```
private bool CheckSingleBadWord()
```

```
{
    // Prepare parameters for selection of the checked bad word
    string where = "[WordExpression] = 'testbadword' ";

    // Get the data
    DataSet words = BadWordInfoProvider.GetBadWords(where, null);

    if (!DataHelper.DataSourceIsEmpty(words))
    {
        // Get DataRow with bad word
        DataRow wordDr = words.Tables[0].Rows[0];

        // Get BadWordInfo object
        BadWordInfo badWord = new BadWordInfo(wordDr);

        // String to be checked for presence of the bad word
        string text = "This is a string containing the sample testbadword, which
can be created by the first code example on this page.";

        // Hashtable that will contain found bad words
        Hashtable foundWords = new Hashtable();

        // Modify the string according to found bad words and return which action
should be performed
        BadWordActionEnum action = BadWordInfoProvider.CheckBadWord(badWord, null,
null, ref text, foundWords);

        if (foundWords.Count != 0)
        {
            switch (action)
            {
                case BadWordActionEnum.Deny:
                    // Some additional actions performed here ...
                    break;

                case BadWordActionEnum.RequestModeration:
                    // Some additional actions performed here ...
                    break;

                case BadWordActionEnum.Remove:
                    // Some additional actions performed here ...
                    break;

                case BadWordActionEnum.Replace:
                    // Some additional actions performed here ...
                    break;

                case BadWordActionEnum.ReportAbuse:
                    // Some additional actions performed here ...
                    break;

                case BadWordActionEnum.None:
                    // Some additional actions performed here ...
                    break;
```

```
            }

            return true;
        }

        apiCheckSingleBadWord.ErrorMessage = "Bad word 'testbadword' is not
present in the checked string.";
        return false;
    }

    return false;
}
```

The following example checks the declared custom string for presence of a all defined bad words.

```
private bool CheckAllBadWords()
{
    // String to be checked for presence of bad words
    string text = "This is a string containing the sample testbadword, which can
be created by the first code example on this page. It also contains the word
asshole, which is one of the default bad words defined in the system.";

    // Hashtable that will contain found bad words
    Hashtable foundWords = new Hashtable();

    // Modify the string according to found bad words and return which action
should be performed
    BadWordActionEnum action = BadWordInfoProvider.CheckAllBadWords(null, null,
ref text, foundWords);

    if (foundWords.Count != 0)
    {
        switch (action)
        {
            case BadWordActionEnum.Deny:
                // Some additional actions performed here ...
                break;

            case BadWordActionEnum.RequestModeration:
                // Some additional actions performed here ...
                break;

            case BadWordActionEnum.Remove:
                // Some additional actions performed here ...
                break;

            case BadWordActionEnum.Replace:
                // Some additional actions performed here ...
                break;

            case BadWordActionEnum.ReportAbuse:
                // Some additional actions performed here ...
```

```
                break;

            case BadWordActionEnum.None:
                // Some additional actions performed here ...
                break;
        }

        return true;
    }

    return false;
}
```

8.9 Banned IPs

8.9.1 Overview

The Banned IPs module is useful when you want to prevent users with certain IP addresses from accessing or using your website in a certain way. This typically happens when a user posts offensive material on a website (e.g. on a forum), harasses site members or behaves in some other unwanted way. IP banning can also be used to restrict access to your websites from certain areas of the world. These bans can be set either for individual websites or globally for all websites in the system.

To learn how to enable the Banned IPs module, please see the Enabling IP banning topic.

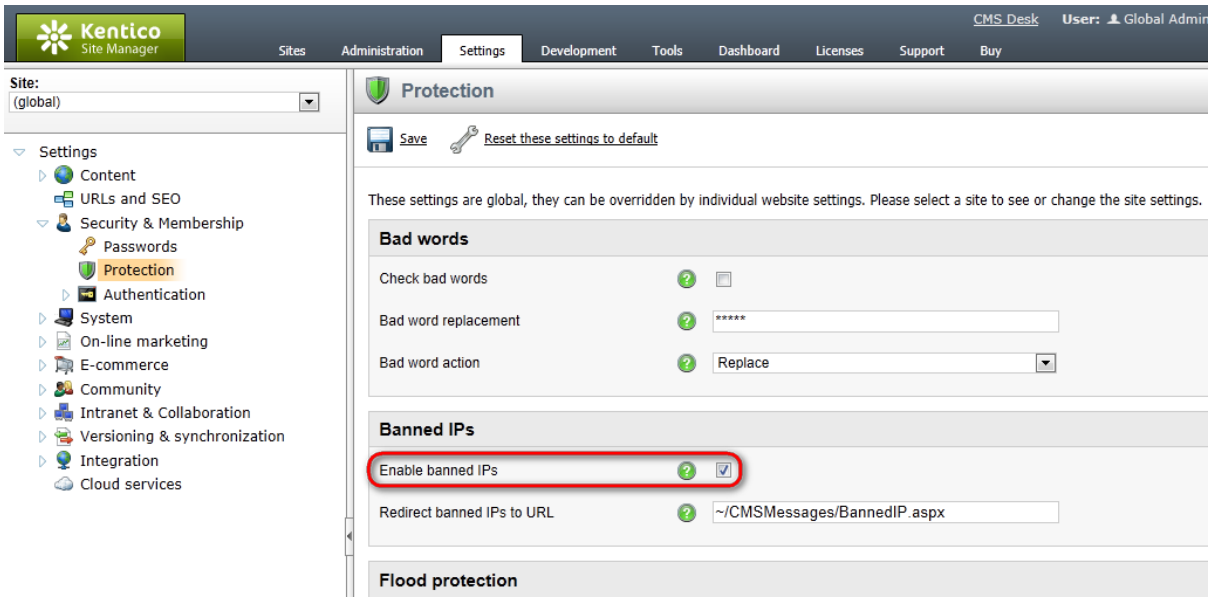
Adding banned IPs can easily be performed by site administrators. The Banning an IP address topic describes how. To find out how to determine which IP a user you want to ban is using, please see the Finding an IP address topic.

The Banned IPs internals and API sub-chapter provides information about the database tables and classes used by the module and examples of how banned IPs can be managed using the API.

IP banning is usually used as a last resort and there are other modules which can help you deal with offensive content and keep your website clean. Please refer to the Modules -> Abuse report and Modules -> Bad words chapters of this guide to read more. If you only wish to temporarily kick a user from a website, the On-line users module provides a way to do so.

8.9.2 Enabling IP banning

For the bans to take effect, go to **Site Manager -> Settings -> Security & Membership -> Protection**, select the appropriate site, check the **Enable banned IPs** check-box and click **Save**. The bans should take effect now.



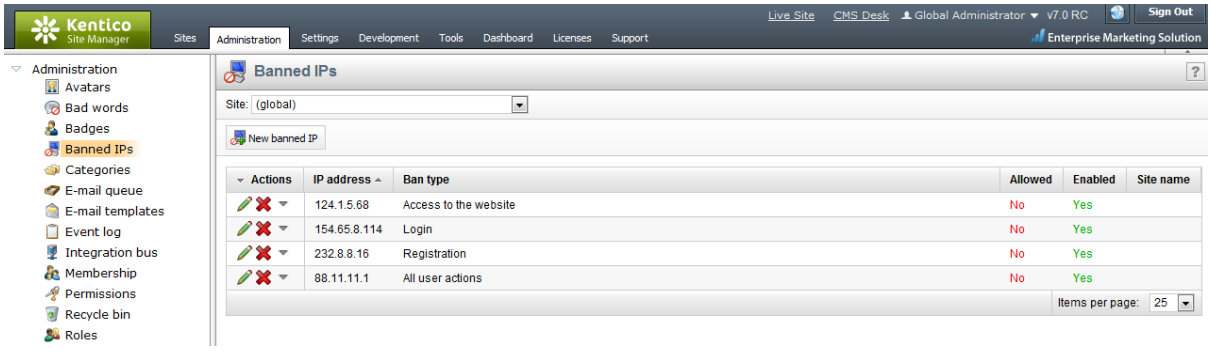
Users attempting to perform an action from an IP address that is banned will get a page with the following message displayed in their browsers (the HTTP response code of the page will be 403.6).

This is the default banned IPs redirect page, which can be found at *~/CMSMessages/BannedIP.aspx*. You can create your own page and set its URL in **Site Manager -> Settings -> Security & Membership -> Protection -> Redirect banned IPs to URL**.

8.9.3 Banning an IP address

Banned IPs can be managed in **Site Manager -> Administration -> Banned IPs**. Alternatively, you can access the same page from **CMS Desk -> Administration**, but you can manage banned IPs only for the currently edited site.

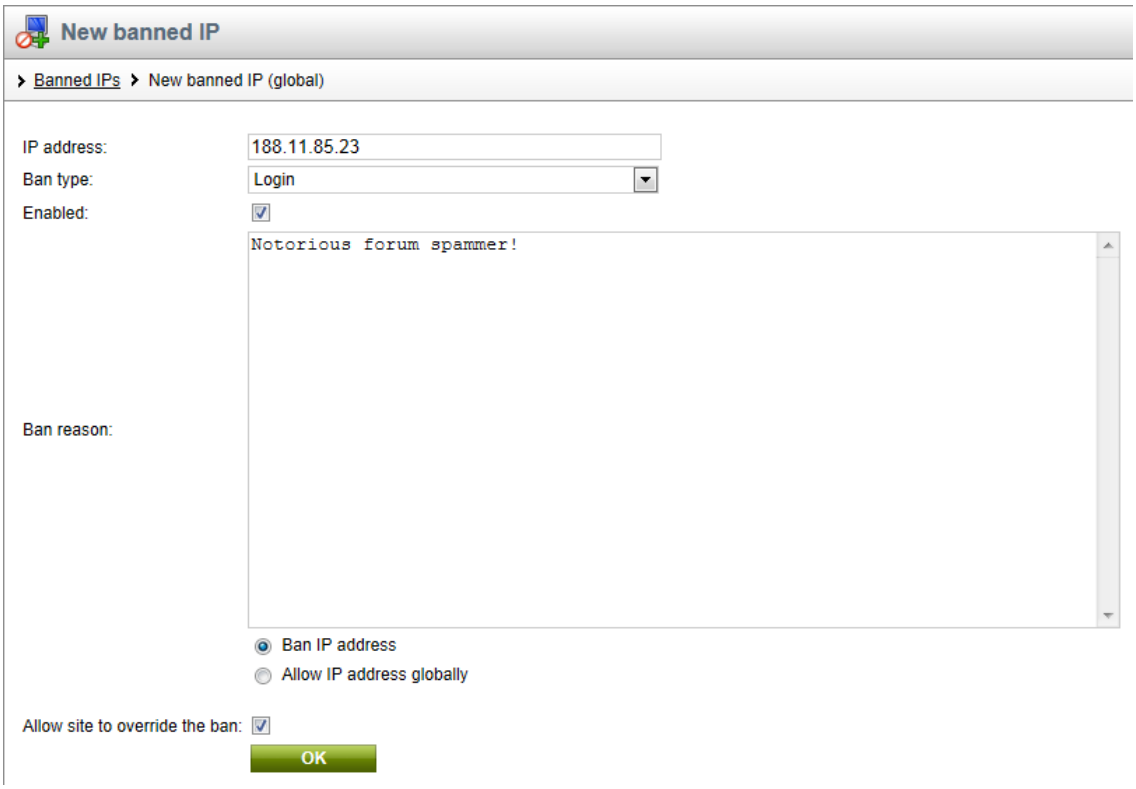
From the **Site** drop-down list, choose the site you want to add the IP for and click **New banned IP**.



Enter the required details:

IP Address	IP address to be banned.

	The asterisk (*) wildcard can be used as a substitute for any number in the IP address, substituting for all values from 0 to 255. Example: 192.168.1.*
Ban type	Type of the ban. The following types are available: • **Access to the website** - users cannot access the site from the entered address at all. • **Login** - users cannot log in from the entered address. • **Registration** - users cannot register from the entered address. • **All user actions** - users can enter the site from this IP, but they are not allowed to register or log in, and they are not allowed to add any content to the site (e.g., blog comments, board messages, etc.).
Enabled	If unchecked, the ban takes no effect.
Ban Reason	Text description of why the IP was banned.
Ban IP address	If selected, this IP address will be banned.
Allow IP address for this site if the IP address is banned globally	If selected, this IP address will be allowed for the selected site even if it is banned globally.
Allow site to override the ban	If checked, this ban can be overridden by bans set for particular sites; can be set only for global bans.



When you have all details entered, click **OK**.

Now if you go back to the list of banned IPs, you should see the newly created record present in the list.

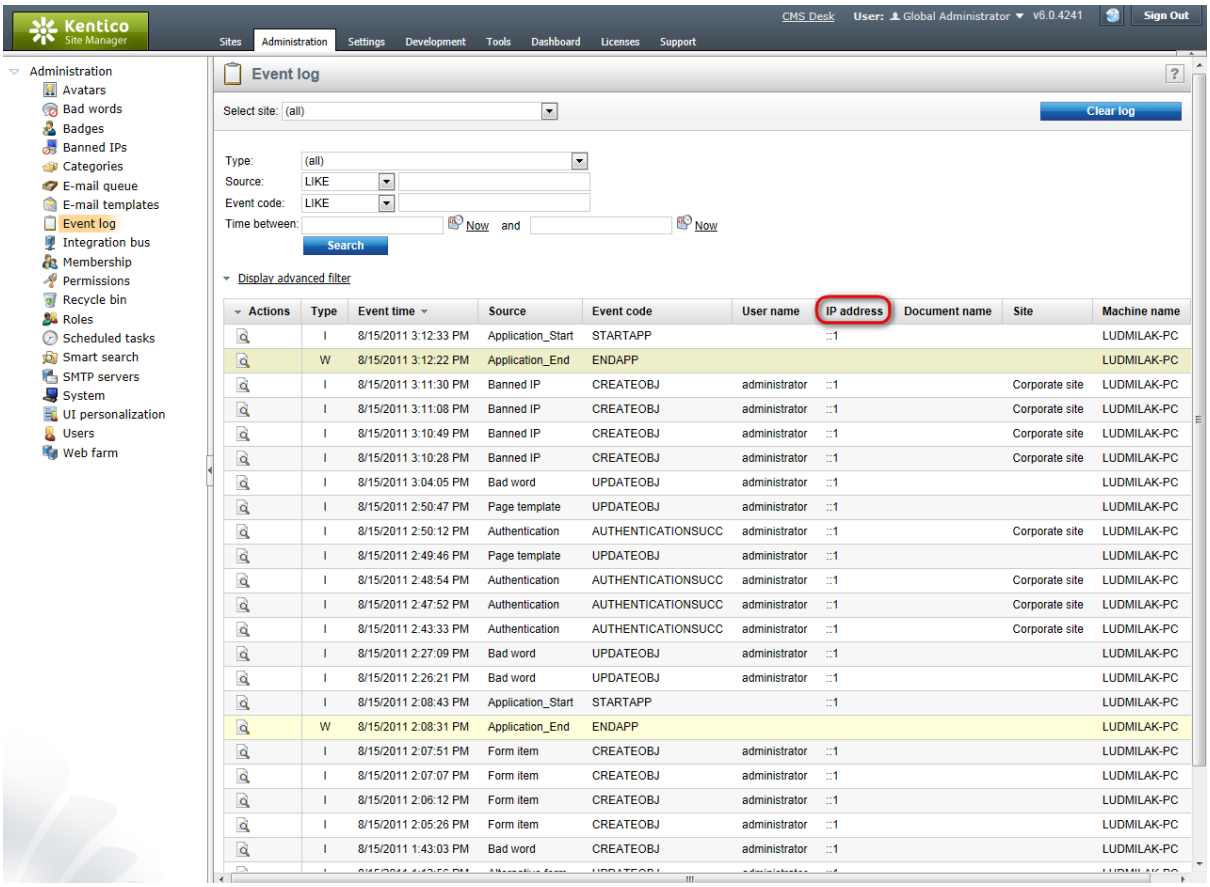
Editing an existing ban

If you want to edit the record in the future, just go to the list and click the **Edit** ( ) icon next to the record. The same page will be displayed as when creating a new record, but with details entered. To make the desired changes, just change the appropriate values and click **OK**.

8.9.4 Finding an IP address

There are several ways to locate user IP addresses.

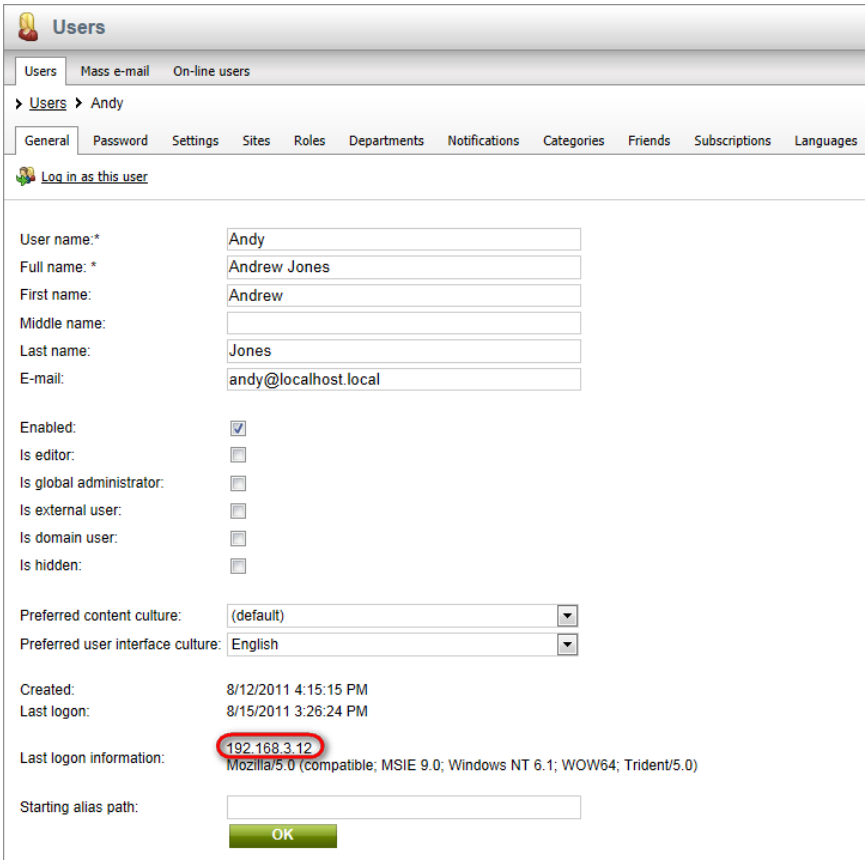
If you wish to find the IP address of a user responsible for a certain event, go to **Site Manager -> Administration -> Event log** and find the event you are interested in. The IP address of the user who caused the event has its own column in the displayed table and can also be viewed if you choose to click **Display event** ( ). This can also be done at **CMS Desk -> Administration -> Event Log**, but here you can only view events from the currently edited site.



For example, if you want to find a user who often uses Bad words, enter *BADWORD* into the **Event code** field of the event log filter and make sure **Contains** is selected. This displays a list of all Bad word violations and all relevant information including user names and their IP addresses.

To find the IP address used by a specific user when they last logged on, go to **Site Manager -> Administration -> Users**. Alternatively, you can access this information from **CMS Desk -> Administration -> Users**, but you can only see the users of the currently edited site.

Choose to **Edit** () the user whose IP address you wish to find. It can be seen under **General -> Last logon information**. This can be useful if you get multiple abuse reports about some user and want to quickly find their IP address.

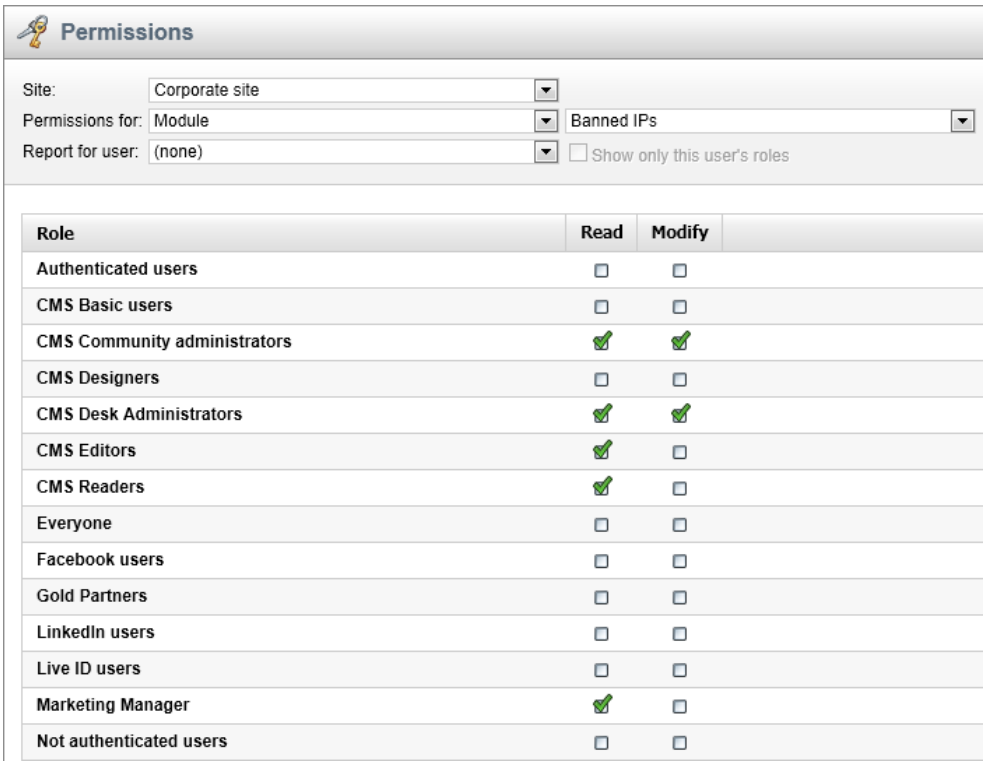


8.9.5 Security

To limit access to the Banned IPs module, go to **Site Manager -> Administration -> Permissions** and grant roles with appropriate permissions according to your needs.

The following permissions can be assigned to the roles:

- **Modify** - members of the role are allowed to add, edit and delete banned IPs
- **Read** - members of the role are allowed to view the banned IPs list



8.9.6 Banned IPs internals and API

8.9.6.1 Overview

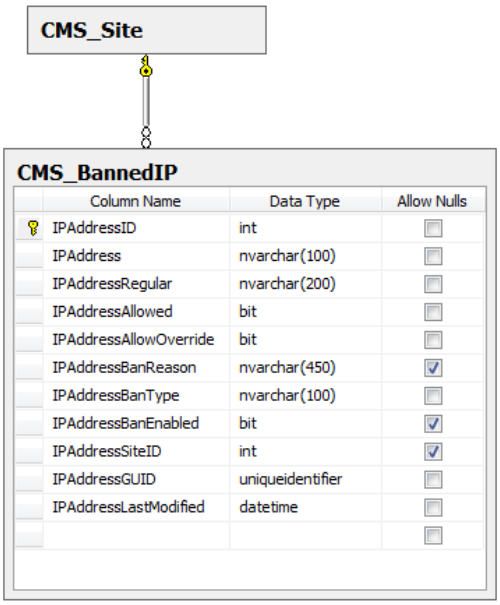
In this chapter, you will learn which database tables and API classes are used in the Banned IPs module. You will also see the most common API examples.

Further API-related information and examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all members of the module's classes, please refer to Kentico CMS API Reference.

8.9.6.2 Database tables

The following database table is used to store information about banned IPs:

- **CMS_BannedIP** - contains records representing banned IP addresses.



8.9.6.3 API classes

If you are not familiar with database table data management through Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

The classes of the Banned IPs module can be found in the **CMS.SiteProvider** namespace.

CMS_BannedIP table API:

- **BannedIPInfo** - represents one banned IP address.
- **BannedIPInfoProvider** - provides management functionality for banned IPs.

8.9.6.4 API examples

8.9.6.4.1 Overview

The following topic shows examples of how the API of the Banned IPs module can be used:

- Managing banned IPs

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

> If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Administration\BannedIP\Default.aspx.cs***.

The banned IP API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.GlobalHelper;
using CMS.CMSHelper;
using CMS.SiteProvider;
```

8.9.6.4.2 Managing banned IPs

The following example adds a banned IP.

```
private void CreateBannedIp()
{
    // Create new banned ip object
    BannedIPInfo newIp = new BannedIPInfo();

    // Set the properties
    newIp.IPAddress = "MyNewIp";
    newIp.IPAddressBanReason = "Ban reason";
    newIp.IPAddressAllowed = true;
    newIp.IPAddressAllowOverride = true;
    newIp.IPAddressBanType = BannedIPInfoProvider.BanControlEnumString
(BanControlEnum.AllNonComplete);
    newIp.IPAddressBanEnabled = true;

    // Save the banned IP
    BannedIPInfoProvider.SetBannedIPInfo(newIp);
}
```

The following example gets and updates a banned IP.

```
private bool GetAndUpdateBannedIp()
{
    // Prepare the parameter
    string where = "IPAddress LIKE N'MyNewIp'";

    //Get the data
    DataSet ips = BannedIPInfoProvider.GetBannedIPs(where, null);

    if (!DataHelper.DataSourceIsEmpty(ips))
    {
        // Get the first banned ip
        BannedIPInfo modifyIp = new BannedIPInfo(ips.Tables[0].Rows[0]);
```

```
        // Update the properties
        modifyIp.IPAddress = modifyIp.IPAddress.ToLower();

        // Save the changes
        BannedIPInfoProvider.SetBannedIPInfo(modifyIp);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates banned IPs.

```
private bool GetAndBulkUpdateBannedIps()
{
    // Prepare the parameters
    string where = "IPAddress LIKE N'MyNewIp%'";

    // Get the data
    DataSet ips = BannedIPInfoProvider.GetBannedIPs(where, null);
    if (!DataHelper.DataSourceIsEmpty(ips))
    {
        // Loop through the individual items
        foreach (DataRow ipDr in ips.Tables[0].Rows)
        {
            // Create object from DataRow
            BannedIPInfo modifyIp = new BannedIPInfo(ipDr);

            // Update the properties
            modifyIp.IPAddress = modifyIp.IPAddress.ToUpper();

            // Save the changes
            BannedIPInfoProvider.SetBannedIPInfo(modifyIp);
        }

        return true;
    }

    return false;
}
```

The following example removes a banned IP.

```
private bool DeleteBannedIp()
{
    string where = "IPAddress LIKE N'MyNewIp%'";

    // Get DataSet
    DataSet ips = BannedIPInfoProvider.GetBannedIPs(where, null);
```

```
    if (!DataHelper.DataSourceIsEmpty(ips))
    {
        // Get the first banned ip
        BannedIPInfo deleteIp = new BannedIPInfo(ips.Tables[0].Rows[0]);

        // Delete the banned ip
        BannedIPInfoProvider.DeleteBannedIPInfo(deleteIp);

        return true;
    }

    return false;
}
```

The method in the following example returns *true* if the specified IP address is banned for the current site, otherwise it returns *false*.

```
private bool CheckBannedIp()
{
    // Prepare the parameter
    string where = "IPAddress LIKE N'MyNewIp'";

    // Get DataSet
    DataSet ips = BannedIPInfoProvider.GetBannedIPs(where, null);

    if (!DataHelper.DataSourceIsEmpty(ips))
    {
        // Get the first banned ip
        BannedIPInfo checkIp = new BannedIPInfo(ips.Tables[0].Rows[0]);

        if (!BannedIPInfoProvider.IsAllowed(checkIp.IPAddress,
CMSContext.CurrentSiteName, BanControlEnum.AllNonComplete))
        {
            return true;
        }
    }

    return false;
}
```

8.10 Banner management

8.10.1 Overview

The Banner management module provides means to create, manage and display advertisements or other content in the form of banners. It also allows you to monitor statistics, namely impressions (banner views) and clicks. The banners are organized into categories, of which there are two types:

- **Global banner category** - shared across multiple websites.
- **Site banner category** - bound to a particular site and not accessible from the other sites.

The module incorporates the Banner rotator web part, which you can use to display banners from a specified banner category. When a user views a page where you placed the Banner rotator, the web part chooses a banner from the category and displays it to the visitor. The banner is rendered as a link to a URL which can be entered in the banner's properties.

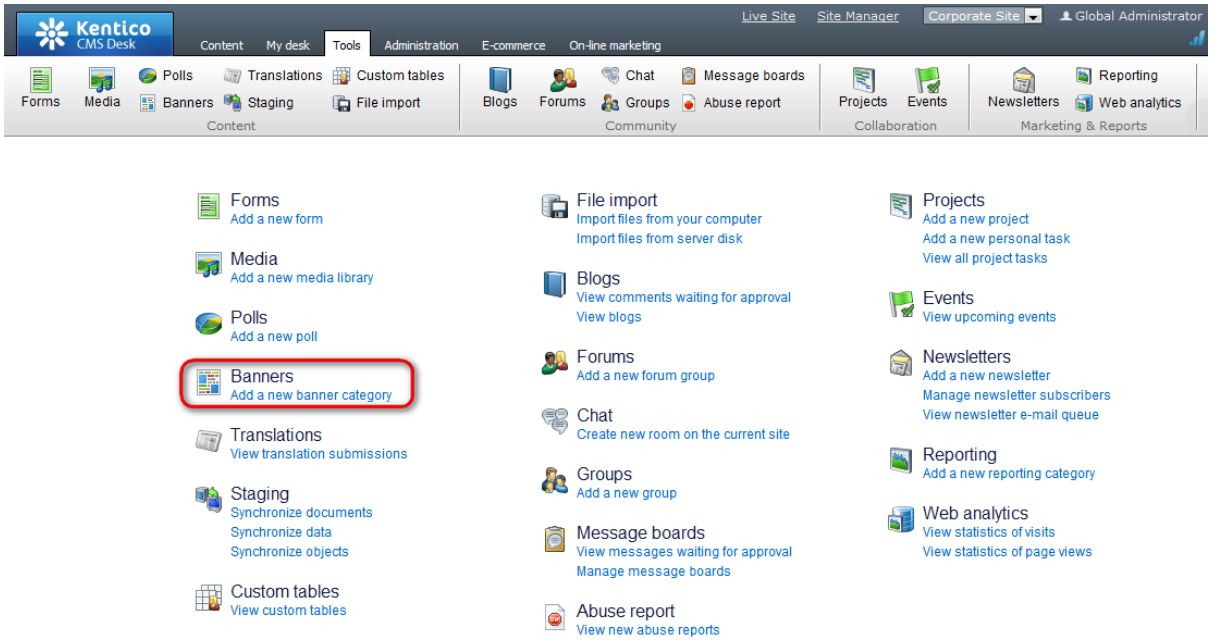
The Banner management module utilizes the built-in Web analytics and Reporting modules to log and display how many times a particular banner has been shown on a page (number of impressions) and how many times visitors have clicked the banner. You can find the following topics in this chapter:

- The Managing banners topic teaches you how to create and manage banners and their categories.
- The Placing banners on a page topic shows you how to display banners on the live site.
- The Statistics topic explains how you can view and administer the statistics that the module records.
- The Example of use topic guides you through the whole process of managing banners on your site.

8.10.2 Managing banners

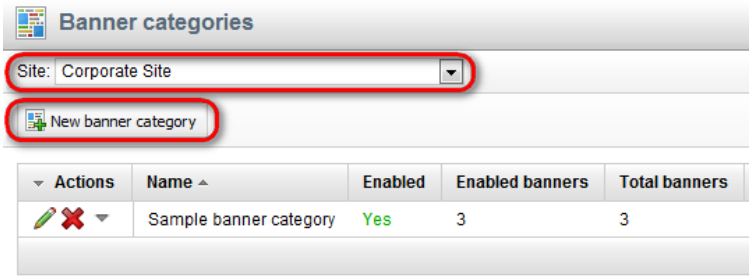
This topic will show you how to create and edit banner categories and the difference between global and site-bound categories. You will also learn about creating and editing banner categories.

Banners and their categories can be managed in **CMS Desk -> Tools -> Banners**.



Creating a new category

First, choose whether the new category will be site-bound or global by selecting an option from the **Site** drop-down list.



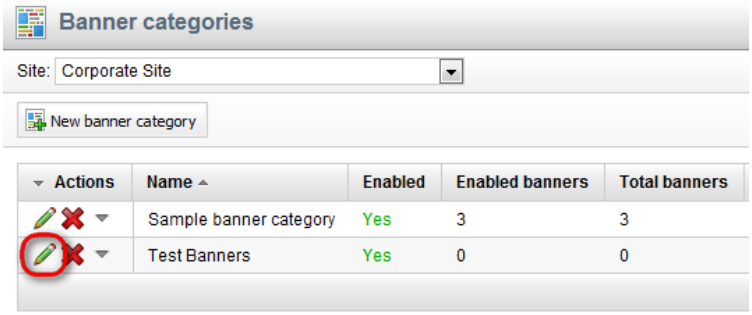
Then, click **New banner category** and fill in the following fields:

- **Display name** - the name of the banner category displayed in the user interface.
- **Code name** - the name of the banner category used in code.
- **Enabled** - turn the checkbox on to allow the banners from this category to be displayed on the live site.

After filling in the required fields, click **Save ()**.

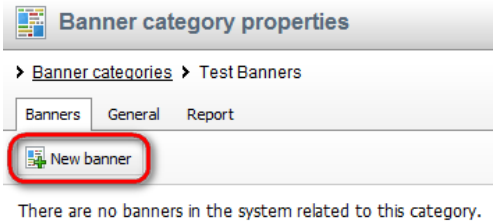
Editing an existing category

To edit a banner category, first select a site, then click the **Edit ()** icon next to the category you want to modify. The category properties can be edited on the **General** tab.



Creating a new banner

To create a new banner, **Edit ()** a category in which you want to create the banner. Then, on the **Banners** tab, click **New banner**.



You will be presented with a form offering a set of banner properties. The following properties are common for all types of banners:

- **Display name** - the name of the banner displayed in the user interface.

- **Code name** - the name of the banner used in code.
- **URL** - URL of the resource you want the banner to link to.
- **Weight** - decimal number that indicates the frequency in which the banner will be displayed on the live site. See the Placing banners on a page and Example topic for more information.
- **Open in new window** - indicates if the banner link will be opened in a new window.
- **Enabled** - Determines whether the banner will be displayed on the live site or not.
- **Valid from** - time from which the banner will be displayed on the live site.
- **Valid to** - time after which the banner will stop being displayed on the live site.
- **Impressions left** - determines how many times the banner will be displayed on the live site at most:
  - *Number of impressions is not limited* **-** choose this option if you don't want to limit the number of times a banner can be displayed.
  - *Allow only specific number of impressions* - allows you to specify how many times will the banner be displayed on the live site before the web part stops displaying it. Specify the number and click *Add*.
- **Clicks left** - determines how many times the banner will be clicked on the live site at most.
  - *Number of clicks is not limited* - choose this option if you want the number of times a banner can be clicked to be unlimited.
  - *Allow only specific number of clicks* - by choosing this option, you can specify the number of times the banner can be clicked on the live site before the web part stops displaying it. Enter the number and click *Add*.

You can choose from three banner types using the **Banner type** drop-down list:

- **Plain text** - the banner will consist of plain text only.
- **HTML** - you will be able to edit this type of banner using the built-in WYSIWYG editor.
- **Image** - the banner will have the form of an image which you'll be able to select using the standard *Select image* dialog.

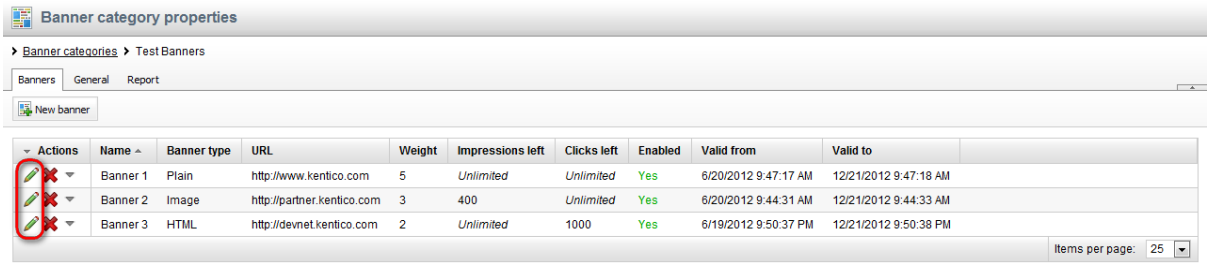
You will be offered additional configuration options depending on the selected type of banner:

Banner type	Configuration options
Plain text	• **Banner content** - plain text, which will represent the banner. You can also enter HTML tags.
HTML	• **Banner content** - any HTML content, which can be edited using the WYSIWYG editor.
Image	• **Banner image** - allows you to select an image from the *content tree*, a media library, or enter a URL of any image on the web. • **Image title** - text which will be rendered into the title attribute of the image tag. • **Alternative text** - text which will be rendered into the alt attribute of the image tag. • **Image CSS class** - CSS class of the image tag. • **Image CSS style** - in-line CSS style that will be rendered for the image tag.

Click **Save** once you have set the banner properties.

Editing an existing banner

A banner can be edited by clicking the **Edit ( )** icon in the list of banners under a selected banner category.



For more information about the way banners are displayed and how these attributes influence whether a banner is shown, refer to the Displaying banners section in the Placing banners on a page topic.

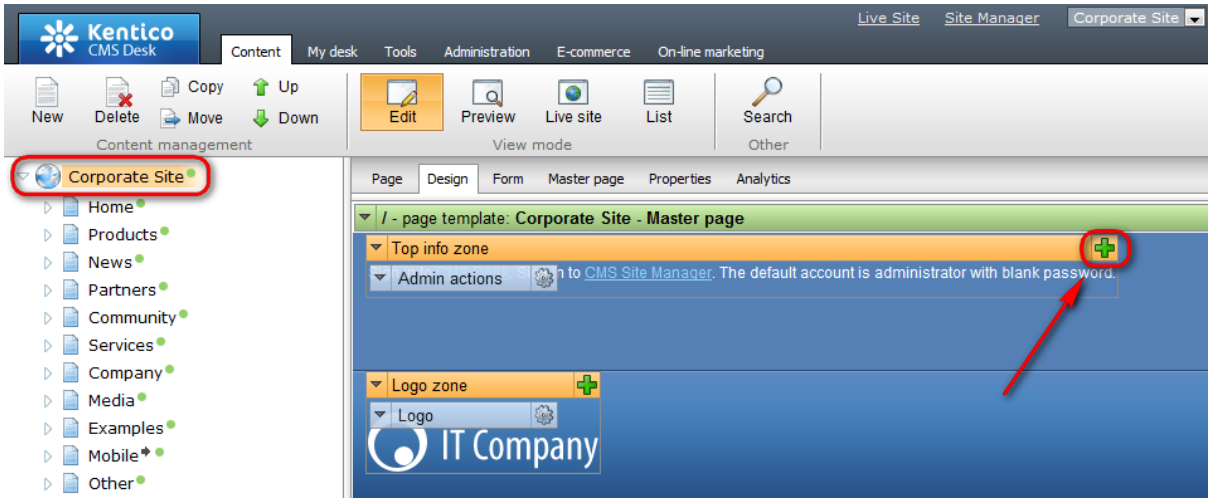
8.10.3 Placing banners on a page

In this chapter, you will learn how to display banners on your website. You will see a complete example of using the Banner management module and its associated web part to show your visitors the most relevant and up-to-date ads.

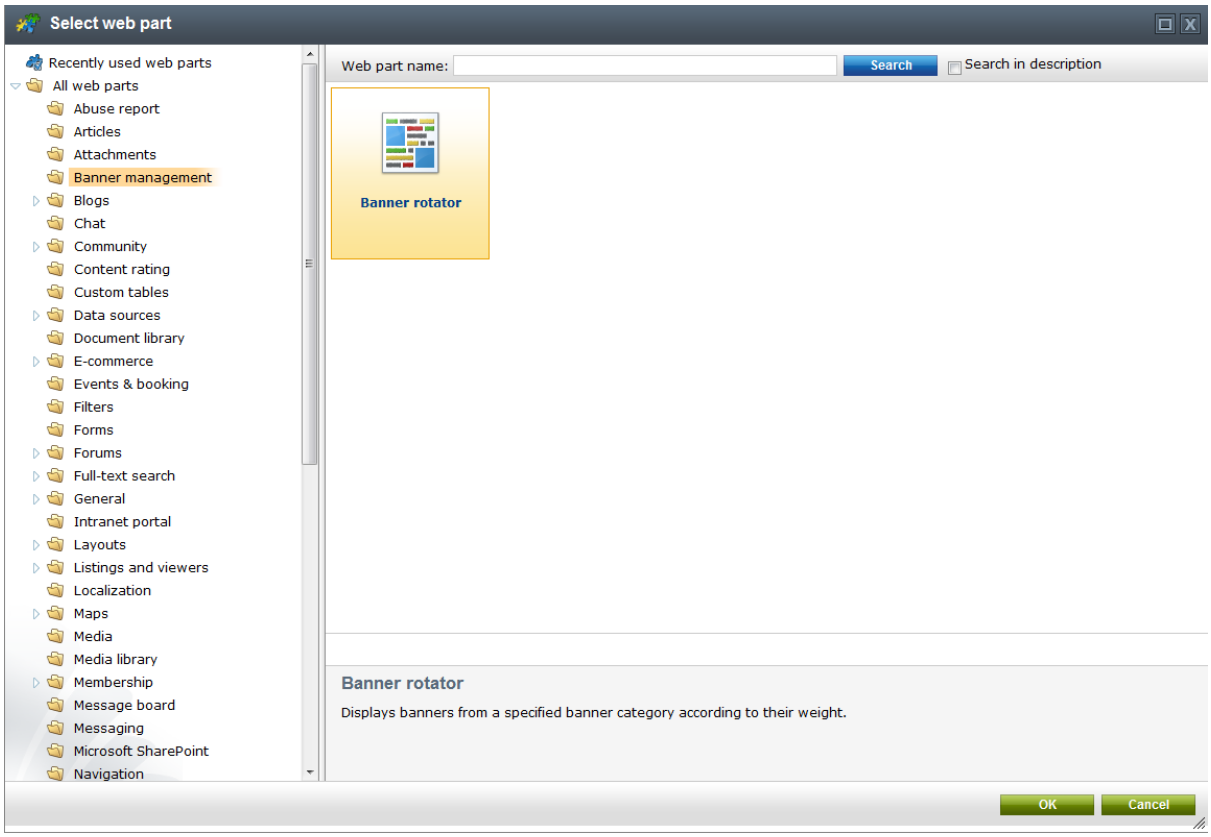
Banner rotator web part

The Banner rotator displays banners from a specified category. When you place the web part on a page, it renders the value stored in the Banner content property of the banner, encapsulated in an **<a>**(anchor) element.

To place the web part on your page, navigate to **CMS Desk -> Content -> Edit -> Design tab** and choose the root document of the website in the **content tree**. Then, click on the **Add web part (✚)** button of the **Top info** web part zone.

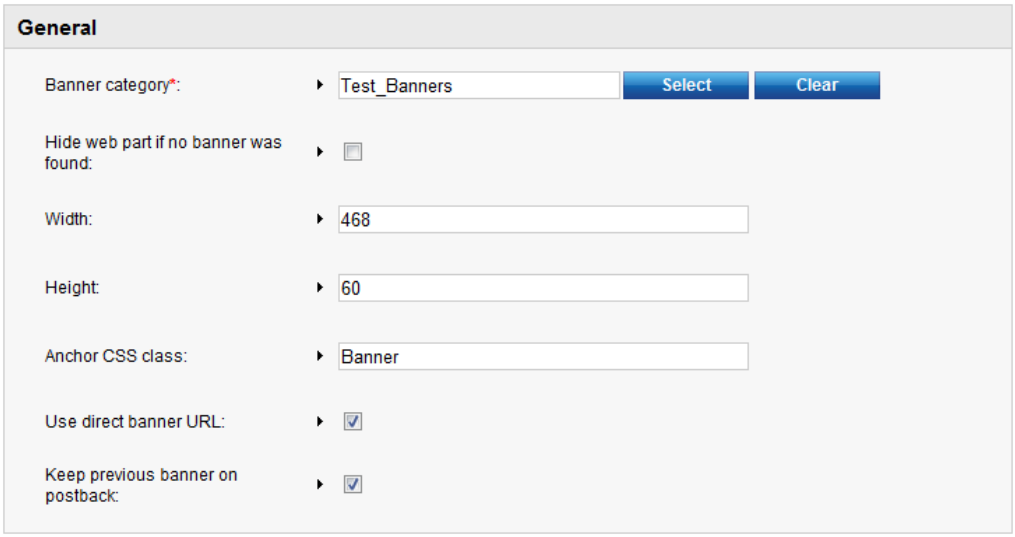


In the **Select web part** dialog, choose **All web parts -> Banner management -> Banner rotator**. Click **OK** to confirm the selection.

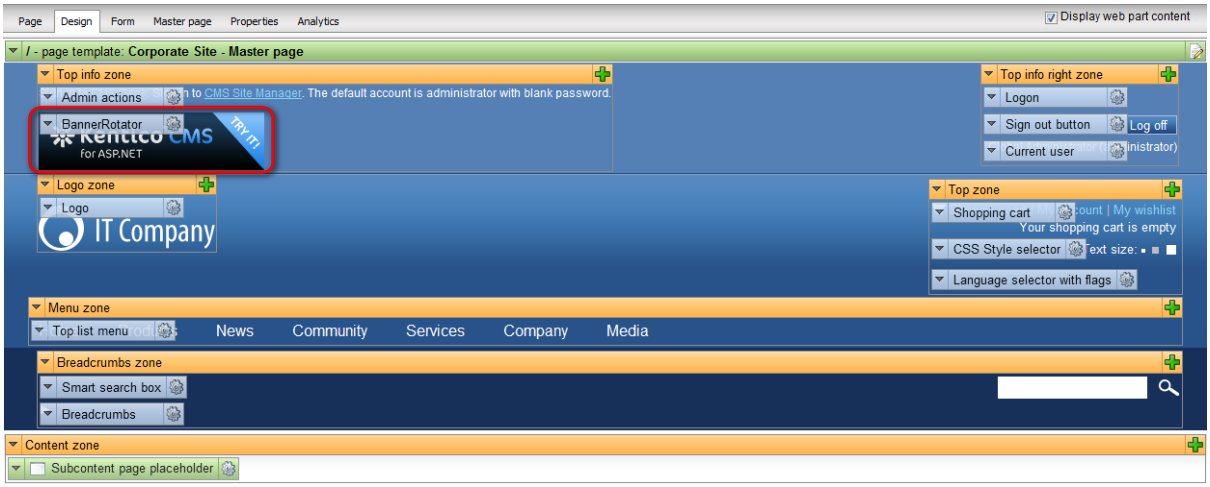


In the **Web part properties** dialog, you can configure the way the banner is displayed by adjusting the web part's properties:

- **Category** - allows you to select the category from which you want to display banners.
- **Hide web part if no banner was found** - when turned on, hides the web part if no banner is found, i.e., if there are no banners in the source category or if the category gets deleted.
- **Width -** defines the width of the banner. The value will be rendered as an in-line style of the encapsulating anchor.
- **Height -** defines the height of the banner. The value will be rendered as an in-line style of the encapsulating anchor.
- **Anchor CSS class** - specifies the CSS class of the anchor element.
- **Use direct banner URL** - determines whether the **href** attribute of the anchor will point to the actual banner URL instead of the default redirection handler. However, the **href** attribute's value will be changed to the redirection handler when the site visitor clicks the link, so as to log the click into the Web analytics log.
- **Keep previous banner on postback** - when turned on, makes sure the same banner is displayed when a postback is made on the page, e.g., when the visitor makes a search or logs in. Also, impressions made on a postback won't count into statistics.



The following image shows a banner management web part placed on the root page of the sample Corporate site.



You can find more information on working with web parts in Kentico CMS in Development -> Web parts.

Banner rotator widget

A widget for displaying banners is also shipped with the Banner management module. In contrast to the web part, its group of settings is limited to the following:

- Category
- Width
- Height

To learn more about widgets and their role on websites, refer to the Development -> Widgets chapter.

Displaying banners

When a user requests a page that contains the Banner rotator web part, the web part selects a banner from the specified category and displays it.

The following factors are taken into account when selecting a banner:

- The **Enabled** property of a banner must be true in order for the banner to be selected.
- Current time must be between the **Valid from** and **Valid to** properties' values.
- The **Impressions left** attribute must have a value greater than zero.
- The **Clicks left** attribute has to be greater than zero.

Another parameter that plays part in the process of selecting banners is the **Weight** property. The higher the weight of a banner, the more often that particular banner is selected by the rotator. The number is decimal, which means you can set a new banner between existing ones without changing their values. For example, if you have two banners with values 4 and 5, you can add a new one with weight set to 4.5.

8.10.4 Statistics

The Banner management module offers logging of banner impressions (views) and clicks via the built-in Web analytics and Reporting modules, so that you can see which banners draw most attention from your site's visitors. You can view the overall statistics of a banner category or display a detailed report about a single banner. This chapter describes how you can view and adjust these reports.

Category report

To view reports for a banner category, navigate to **CMS Desk -> Tools -> Banners**, **Edit (** **)** the banner category and switch to the **Report** tab. You can do the following in the tab:

- **Save** the report for later use. You can find the saved reports in *CMS Desk -> Tools -> Banner management -> Banner impressions and clicks by category -> Saved reports* tab.
- **Print** the report.
- **Subscribe** to the report to have it sent in specified intervals via e-mail.
- Click the **Update** button to redraw the report for the period specified in the *From* and *To* dates.
- Right-click the table to **Subscribe** to the report via e-mail.

Banner report

On the **Banners tab**, **Edit (** **)** a banner and switch to the **Report** tab to view statistics for the particular banner. You can do the following in the tab:

- **Save** the report for later use. You can find the saved reports in *CMS Desk -> Tools -> Banner management -> Banner impressions and clicks by category -> Saved reports* tab.
- **Print** the report.
- **Subscribe** to the report to have it sent in specified intervals via e-mail.
- Click the **Update** button to redraw the report for the period specified in the *From* and *To* dates.
- Right-click the graph to export the report to **Excel,** **CSV** and **XML**, or to **Subscribe** to the report via e-mail.

Tip

Additional reports for the Banner management module can be found in **CMS Desk -> Tools -> Reporting**, in the **Web analytics** report category.

For more information about the Reporting module, see its documentation.

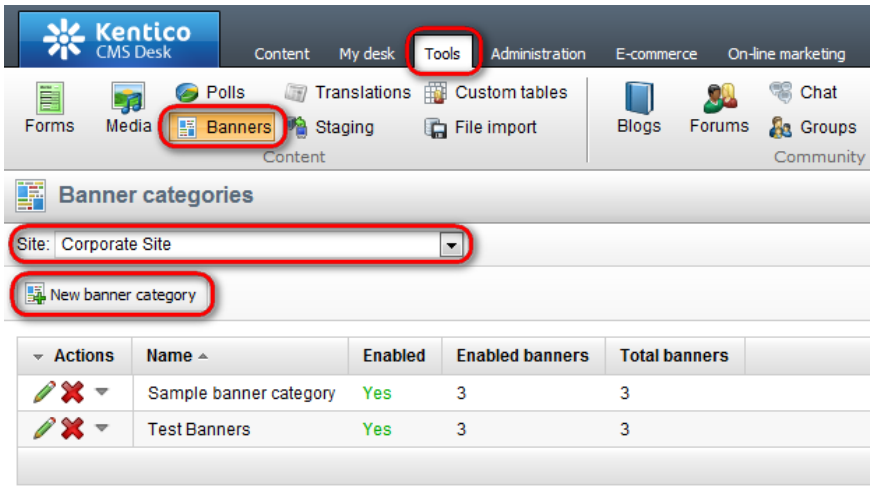
8.10.5 Example of use

This example will show you the whole process of banner management, from creating a banner category and filling it with banners, to displaying the banners on a site. The example will use the sample Corporate site.

Creating banners

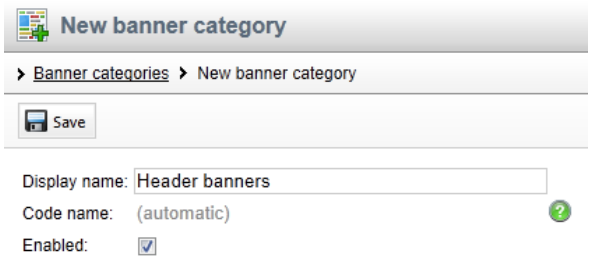
1. Log in to **CMS Desk**, switch to the **Tools** tab and select **Banners**.

2. In the **Site** drop-down list, select **Corporate site** and click **New banner category**.

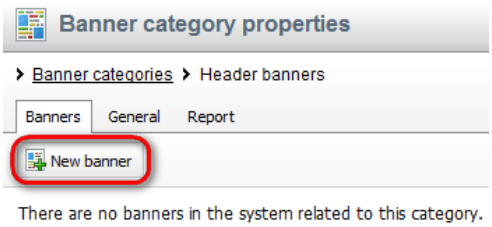


3. Set the banner category properties as follows:

- **Display name:** Header banners
- **Enabled:** True

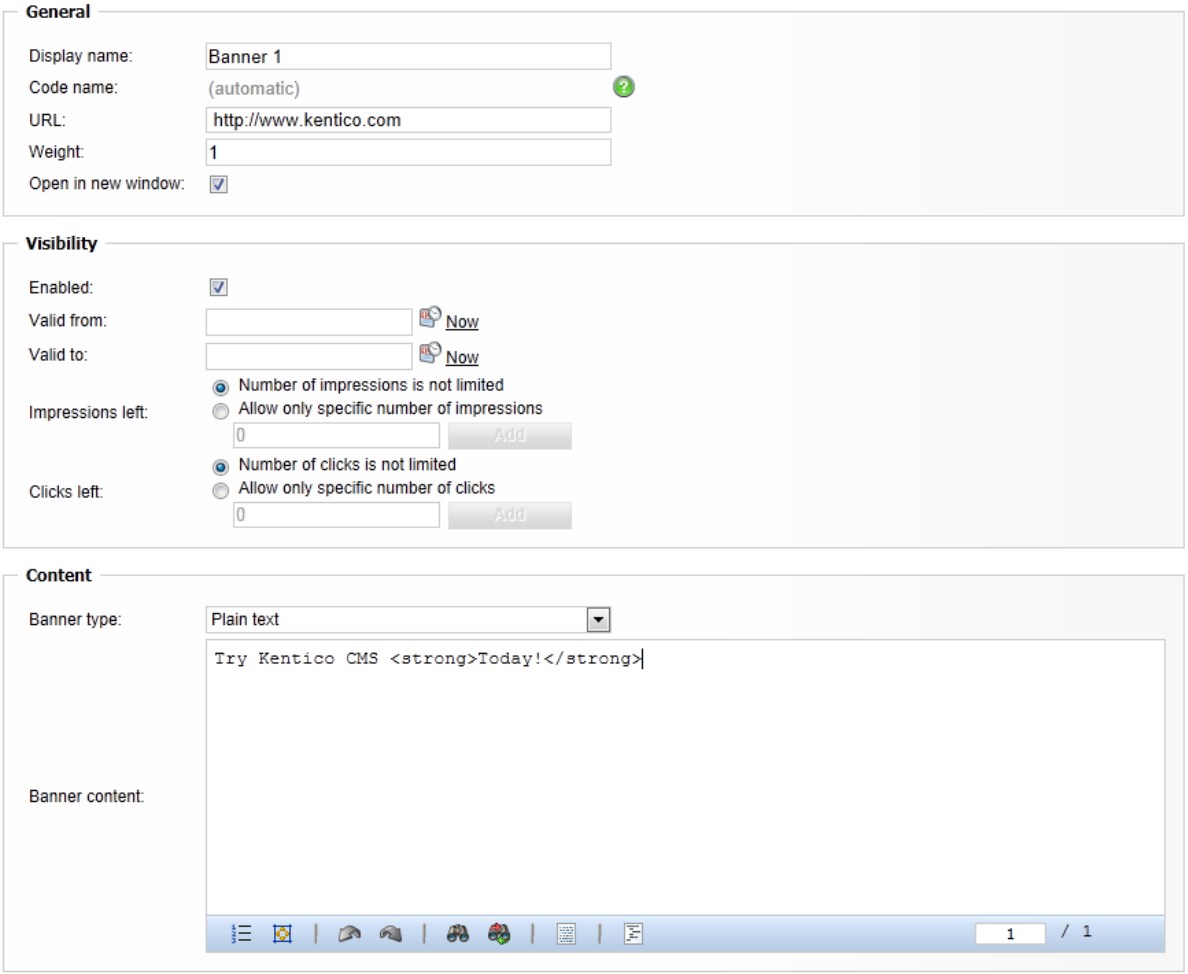


4. Click **Save** and then **New banner**.



5. Set the new banner's properties as follows:

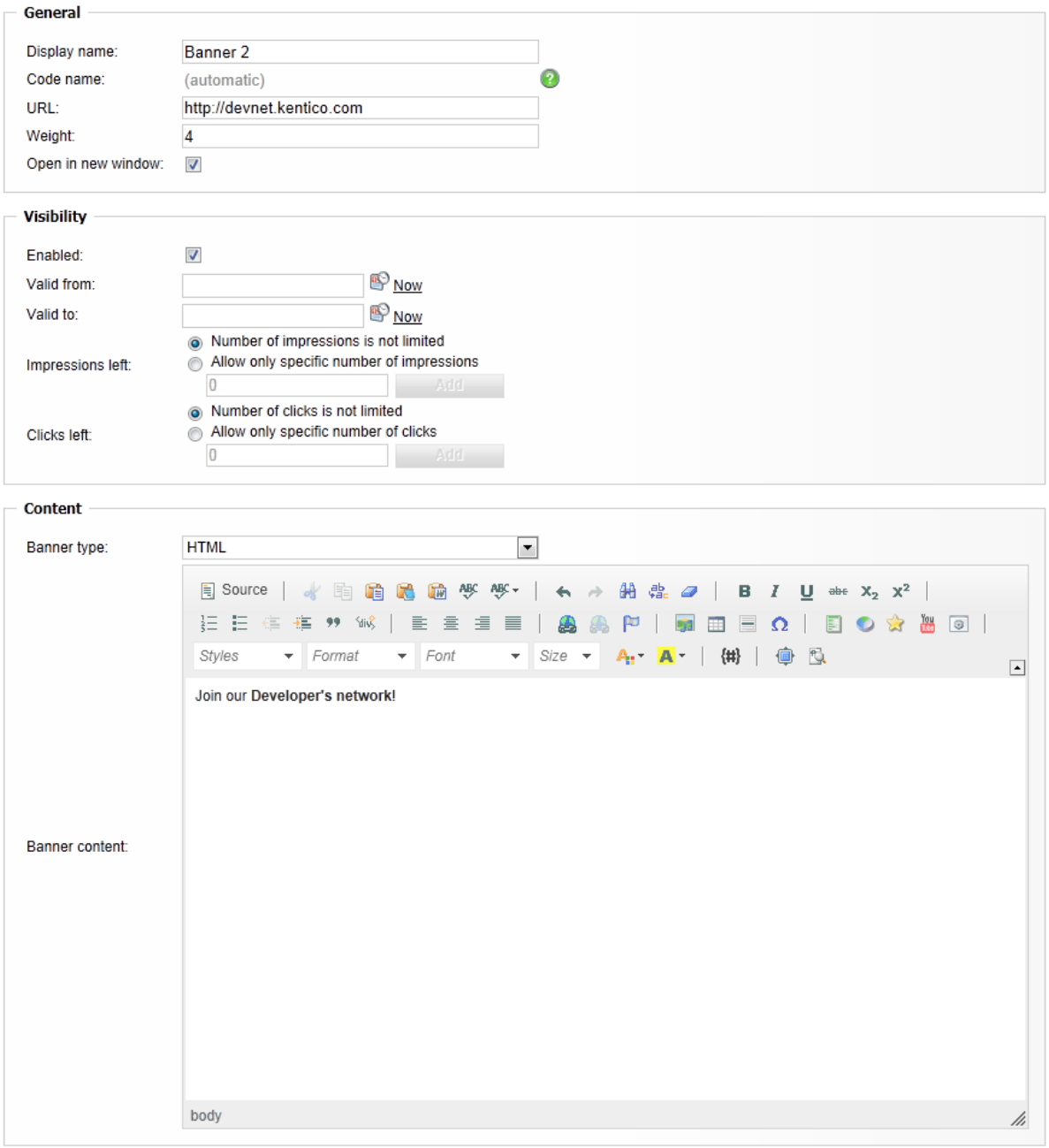
- **Display name:** Banner 1
- **URL:** http://www.kentico.com
- **Weight:** 1
- **Open in new window:** True
- **Enabled:** True
- **Banner type:** Plain text
- **Banner content:** Any text you like



6. Click **Save**.

7. Return to the list of banners, click **New banner** again and set its properties as follows:

- **Display name:** Banner 2
- **URL:** http://devnet.kentico.com
- **Weight:** 4
- **Open in new window:** True
- **Enabled:** True
- **Banner type:** HTML
- **Banner content:** Any formatted text you like



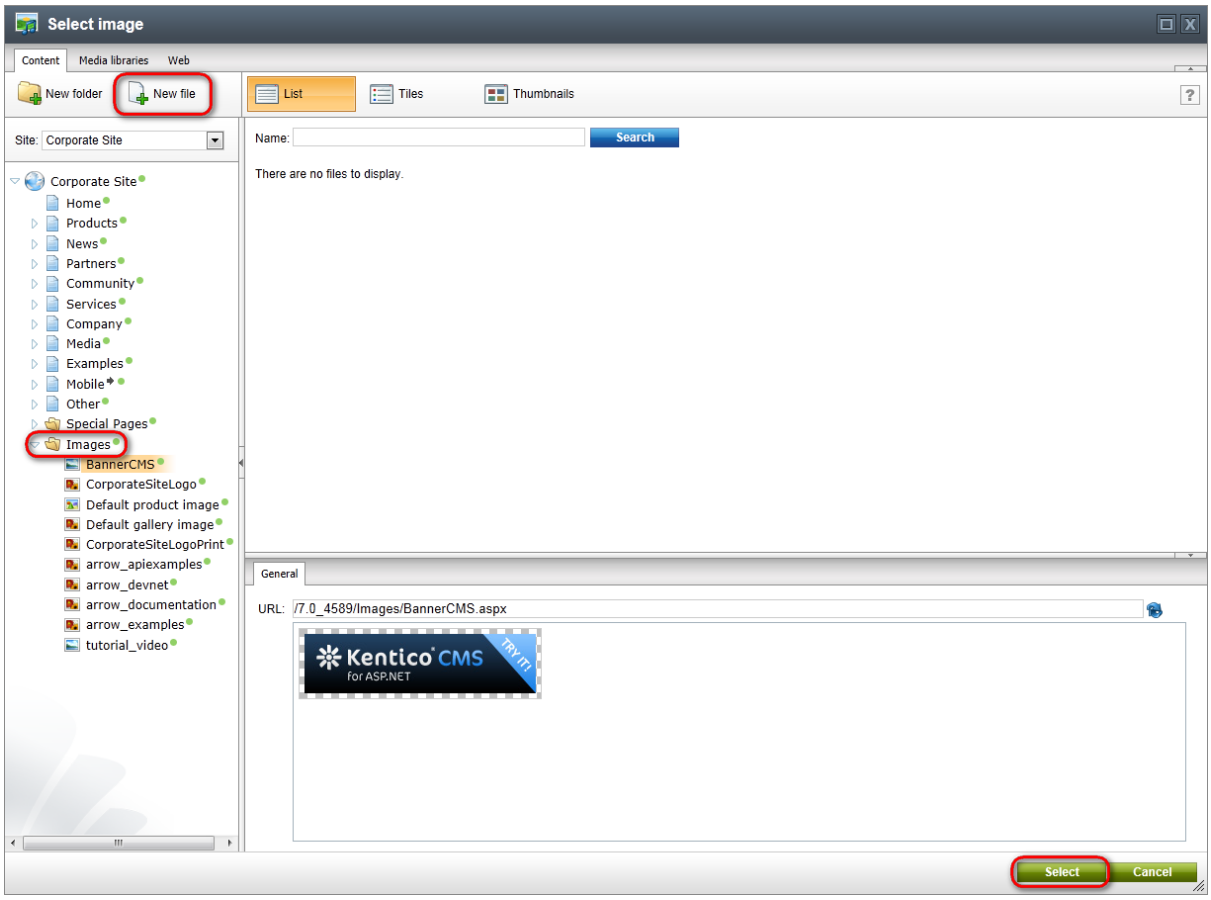
8. Click **Save**.

9. Return to the list of banners, click **New banner** the last time and set its properties as follows:

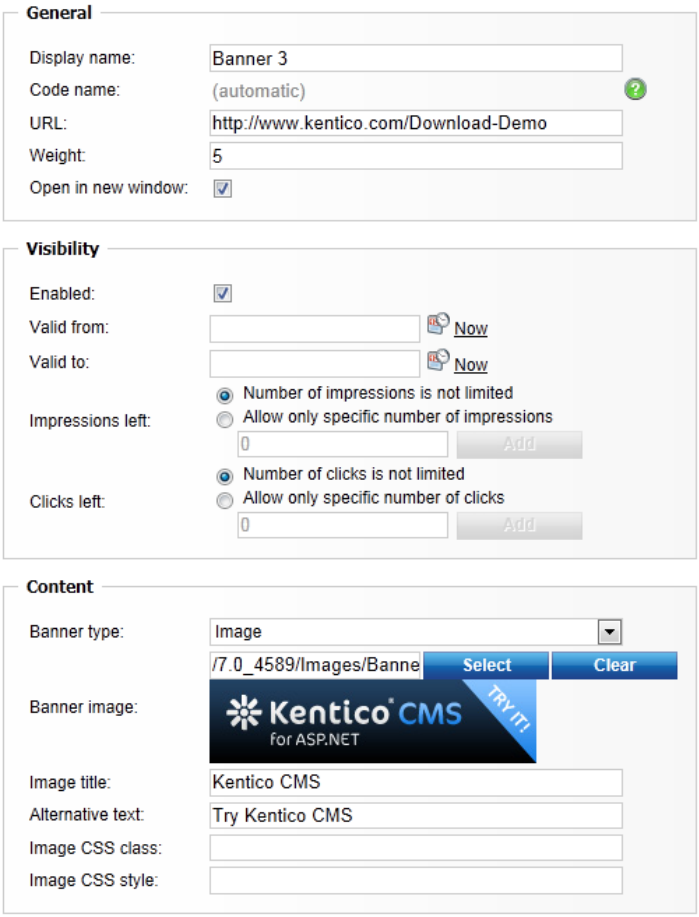
- **Display name:** Banner 3
- **Code name:** banner3
- **URL:** http://www.kentico.com/Download-Demo
- **Weight:** 5
- **Open in new window:** True
- **Enabled:** True
- **Banner type:** Image

10. Click **Select** next to the **Banner image** property. This will open the **Select image** dialog.

11. Upload a file of your choice into the **Images** folder and click **Select**.



12. Fill in the **Image title** and **Alternative text** fields as shown in the following picture.



13. Click **Save**.

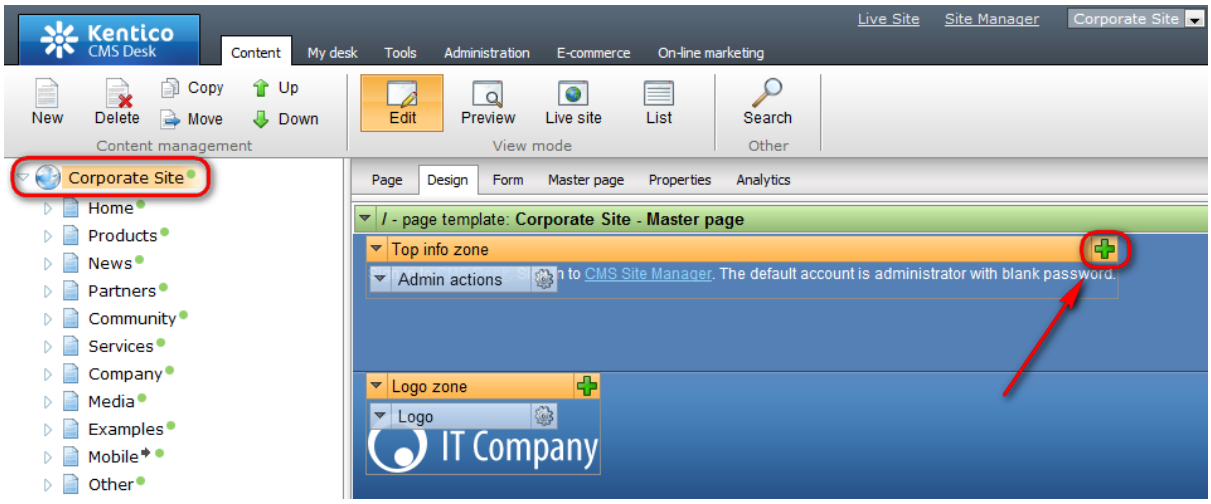
Placing banners on a page

You have created a set of banners that will rotate in the header of the Corporate site. Now place the Banner rotator web part on the Corporate site's master page as described in the following steps:

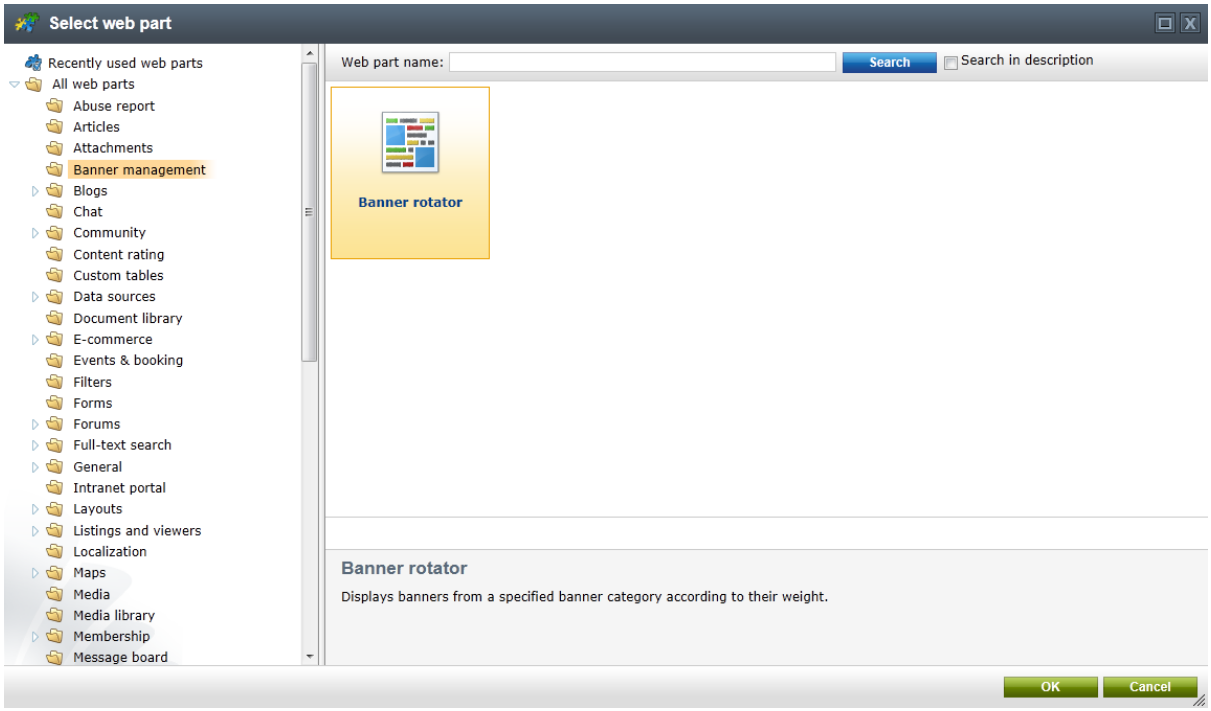
1. In **CMS Desk**, switch to the **Content** tab and choose **Edit**.

2. Select the root of the Corporate site and switch to **Design** tab.

3. Click **Add web part (+)** in the **Top info** web part zone.

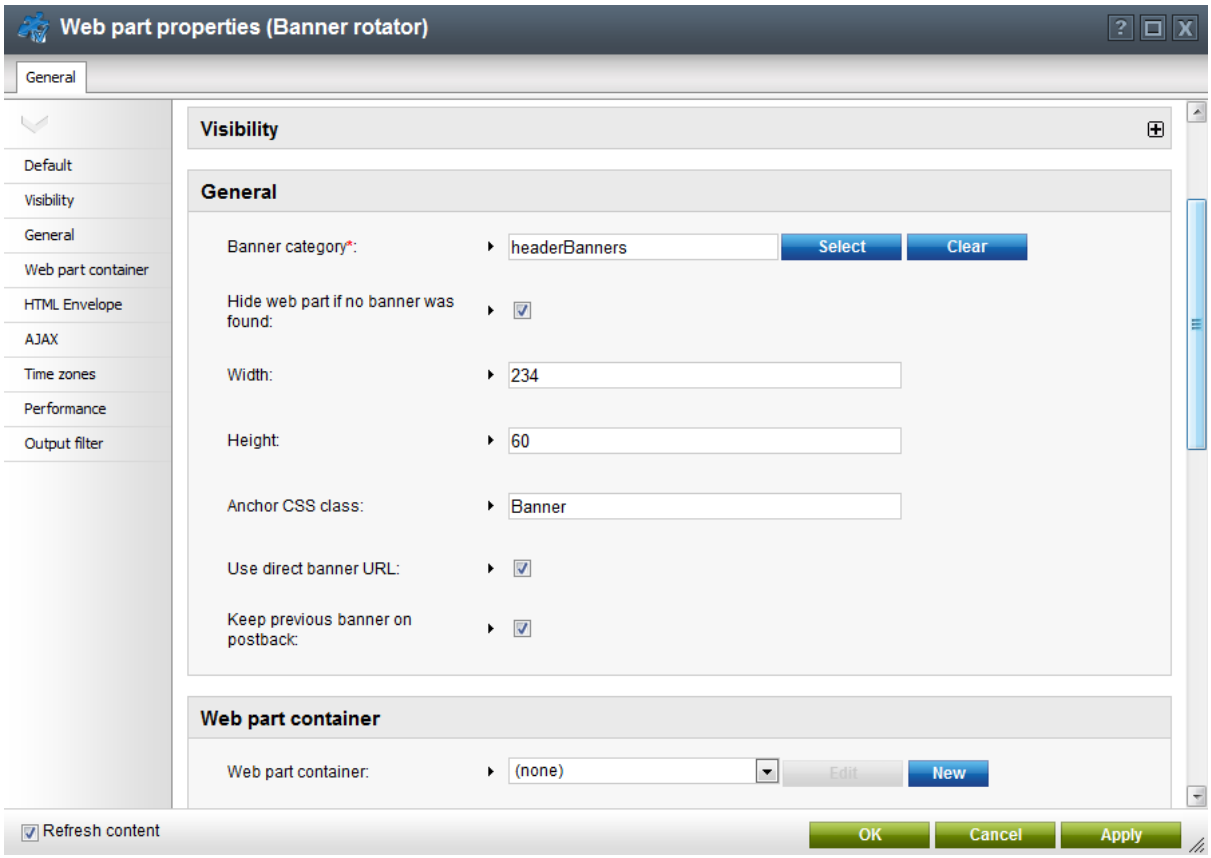


4. In the **Select web part** dialog, choose the **All web parts -> Banner management -> Banner rotator** web part.



5. In the **Web part properties** dialog, select the **Header banners** category created earlier.

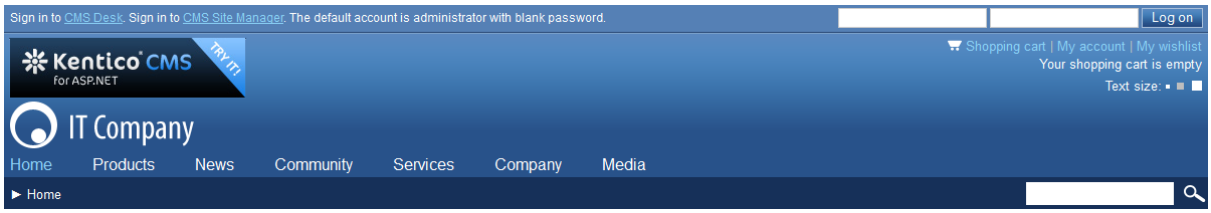
6. Enter the banner's dimensions and optionally set the rest of the properties.



7. Click **OK** to insert the web part.

The result

You have created three sample banners and configured the Banner rotator web part to display them to visitors in the header of the Corporate site.



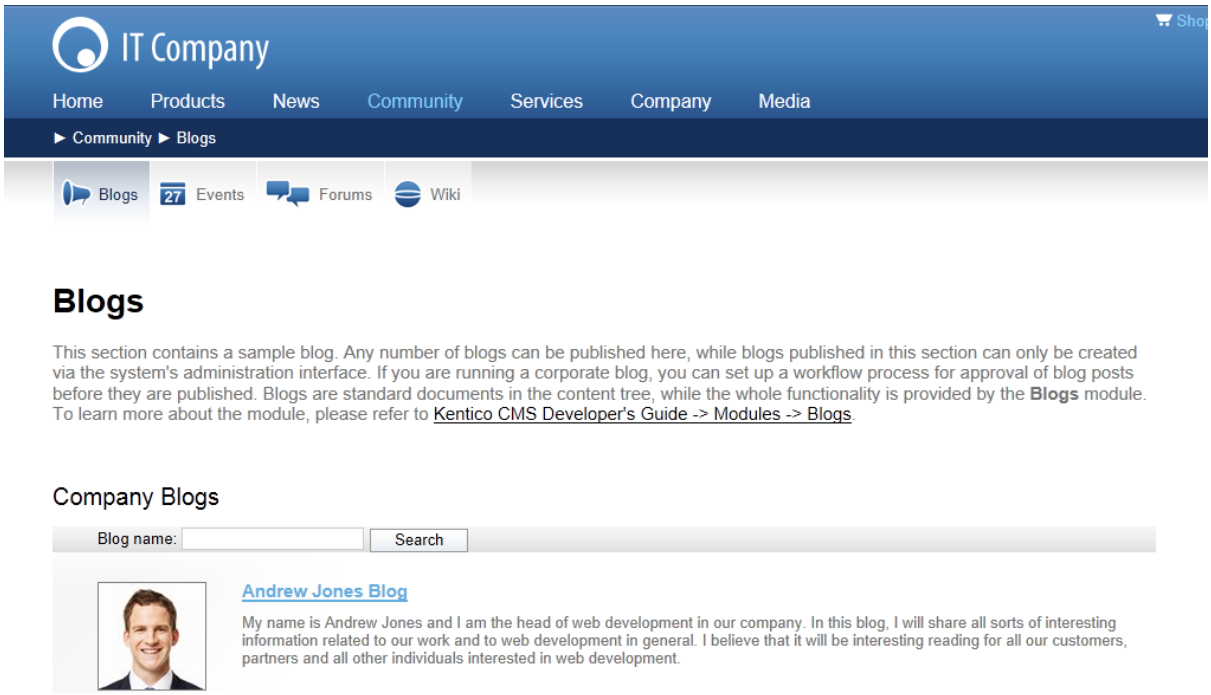
When you switch to the live site, you will be presented with one of the three banners. When you refresh the page several times, the other banners will appear with frequency proportional to their **Weight**. For example, out of 100 page views, the banners' displaying frequency will be as follows:

- **Banner 1** has a weight of 1, therefore it will be displayed approximately 10 times.
- **Banner 2** has a weight of 4, therefore it will be displayed approximately 40 times.
- **Banner 3** has a weight of 5, therefore it will be displayed approximately 50 times.

8.11 Blogs

8.11.1 Overview

The Blogs module allows you to publish a personal or company blog, which is a frequent, chronological publication of one's thoughts and web links. You can publish multiple blogs on the same site and there can be multiple editors for each blog.



The Blogs module fully leverages the standard content management engine, so every blog post you create is a standard document that can be displayed on the website, searched, etc. You can also configure permissions and workflow for every blog as you're used to do with other content.

- To see an example of blog, please refer to a sample blog on the Corporate Site sample website.
- To learn how to create a new blog on the sample Corporate Site, please refer to the Adding a blog to your site topic.
- To learn how to add posts to your blog, please refer to the Adding posts to your blog topic.
- To learn how comments can be moderated, please follow the Moderating comments topic.
- To learn how to configure the layout and design of your blog, please follow the Blog layout and design topic.
- To learn how to enable users to perform selected tasks using the User contributions module web parts, please refer to the On-site management via User contributions topic.
- If you would like to adjust the settings and security of the Blogs module, please refer to the Settings and Security topics respectively.
- To learn how to deal with blog post documents using standard API for documents, please refer to the Blogs internals and API chapter.

Subchapters on trackbacking and on a programming interface enabling co-operation of the Blogs module with external programs, MetaWeblog API, and topics related to Blog comments notifications (Who can be notified, User subscriptions and E-mail templates) are also available in the Blogs chapter.

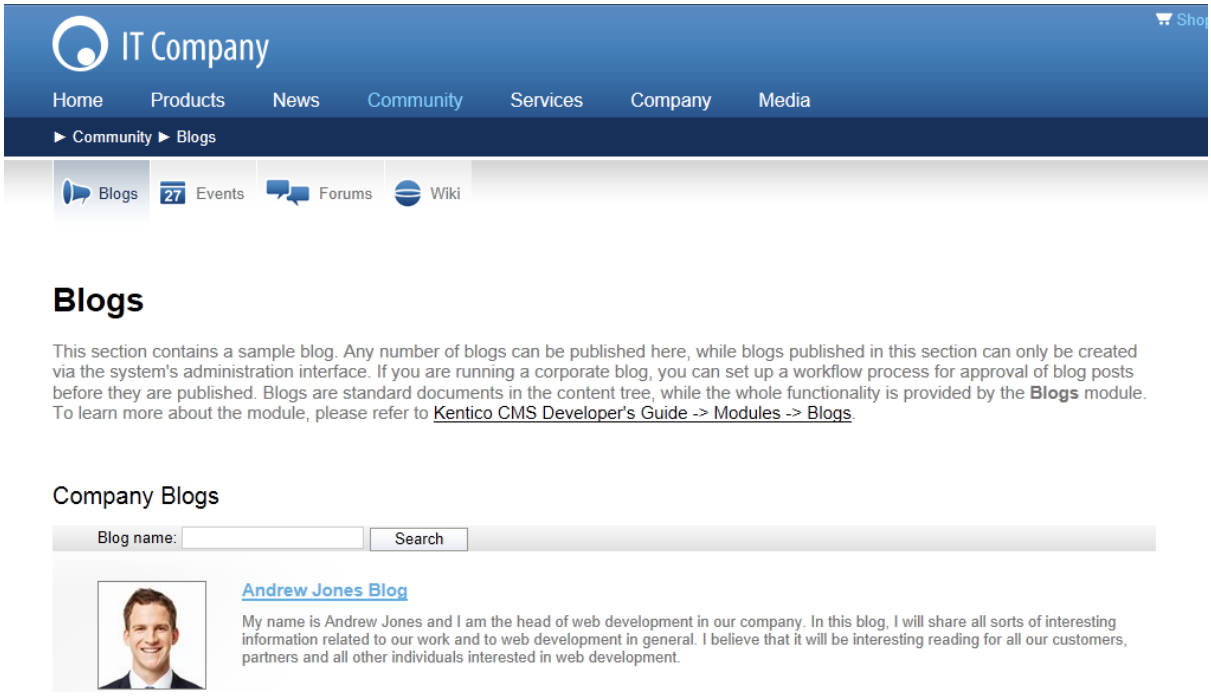
If you would like to compare the Corporate Site sample blog with a more detailed example of blogging, please switch to the Community Site sample website where you can find not only blogs with blog posts but also a tag cloud. The Tag cloud web part is used to display key words, called tags that are associated with a document, and is convenient for simple marking of documents according to various criteria, e.g. your interests.

Kentico CMS Community Site Guide contains some additional examples and tutorials related to blogs:

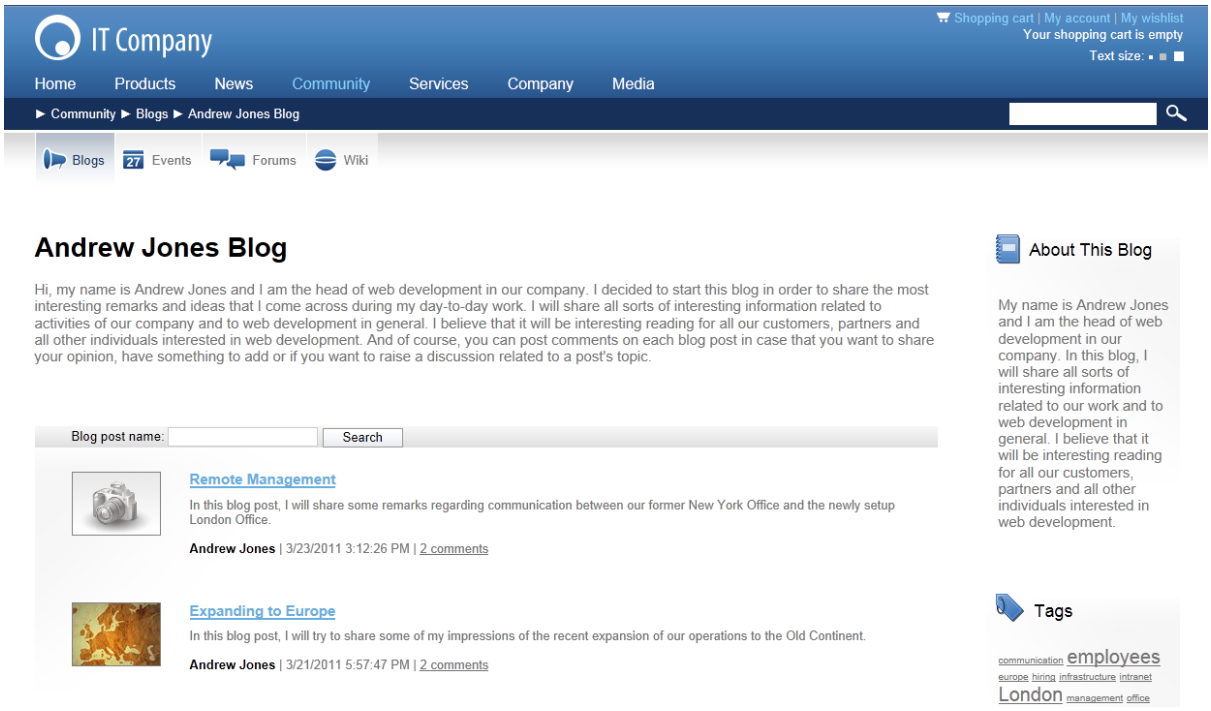
You will learn not only how to create a page on blogs (learn here how to do it) but also how to create a special page where users are redirected when they want to create a new blog (as explained here), a page that displays a list of all blogs on the site (as explained here) or how to create a page where only blog posts corresponding to the selected tag from the tag cloud are displayed (as explained here).

8.11.2 Sample blog

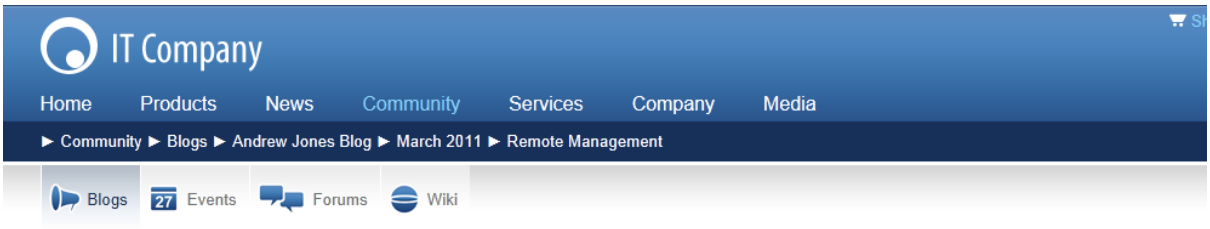
To see a sample blog, on the sample Corporate Site switch to the live site and from the main menu, choose **Community -> Blogs**. The **Blogs** page displays a list of blogs, each of which can be accessed by clicking on the particular blog link.



After clicking the chosen blog link, a new page with a list of blog posts and some additional information is displayed. As you can see, the additional information such as blog description, tags, favorite websites, recent posts, RSS feed link and post archive is displayed on the right of the page.

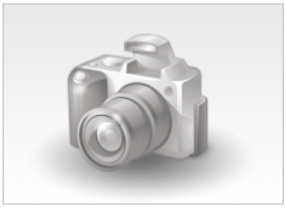


Finally, when you click on some particular blog post link, you can see the text plus the **Comments** section that allows you to read and add comments to the blog post. Additional information related to the current blog is also available.



Remote Management

In this blog post, I will share some remarks regarding communication between our former New York Office and the newly setup London Office.



As you have learned from my previous blog post, we have recently set up a new office in London, UK. I was involved in the whole setup process from its very beginning, but after the initial phase, I am back home in the US. Due to the fact that I am the head of web development in Our Company, it is within my responsibilities to oversee all work carried out by the London Office as well. And here comes the question about how to do this effectively while being on the other side of the Atlantic Ocean.

The first thing that is essential for successful remote management is trust. You simply have to believe that the people in the other office are competent enough and that you do not have to micro-manage their work. I paid special attention to choose the right people during the hiring process, so the trust I was talking about is definitely in place.

Not less important is to establish suitable communication channels. Traditional e-mails are a must, while using Skype for instant messaging and voice chat is a great and cost-free option. But what really did the job in our case was our new intranet based on Kentico Intranet Solution. This is a full-featured intranet solution based on Kentico CMS, the ASP.NET web content management system. It showed its true potential by enabling us to create an internal knowledge base to share our know-how, manage projects by means of its integrated project management features, have dedicated website sections for particular departments, etc. And it is all web-based, so we have a single intranet solution accessible from anywhere used by the both of our offices.

| 3/23/2011 3:12:26 PM | 2 comments
Filed under: communication, employees, intranet, London, management

Comments

Andrew Jones

Hello Paul, yes, I strongly recommend giving it a try. If you already have experience with Kentico CMS, it will be a piece of cake for you to set it up. It has loads of Intranet-specific features out of the box. And with the endless customization possibilities the CMS provides, any functionality that you can imagine is achievable.

6/29/2011 9:04:40 PM

Edit Delete Reject Report abuse

Paul Woods

Hi Andrew, great to know about the Intranet Solution by Kentico. I knew they were producing a brilliant CMS, but I didn't realize it could be used for intranet purposes. I'll definitely try it out and see if it suits our purposes. We are planning to update our intranet soon as the one we've got now is just terrible and not sufficient anymore.

6/29/2011 9:04:08 PM

Edit Delete Reject Report abuse

Leave comment Subscribe

Name:
Global Administrator

E-mail:
administrator@localhost.local

Your URL:

Comments:

Subscribe me to this blog post

8.11.3 Adding a blog to your site

In this example, you will learn how to create a new blog on the sample Corporate Site, under the Blogs section.

1. Sign in to **CMS Desk**. In the **Content** section, navigate to **Community -> Blogs** and click **New**. Choose to create a new **Blog**. Enter the following details:

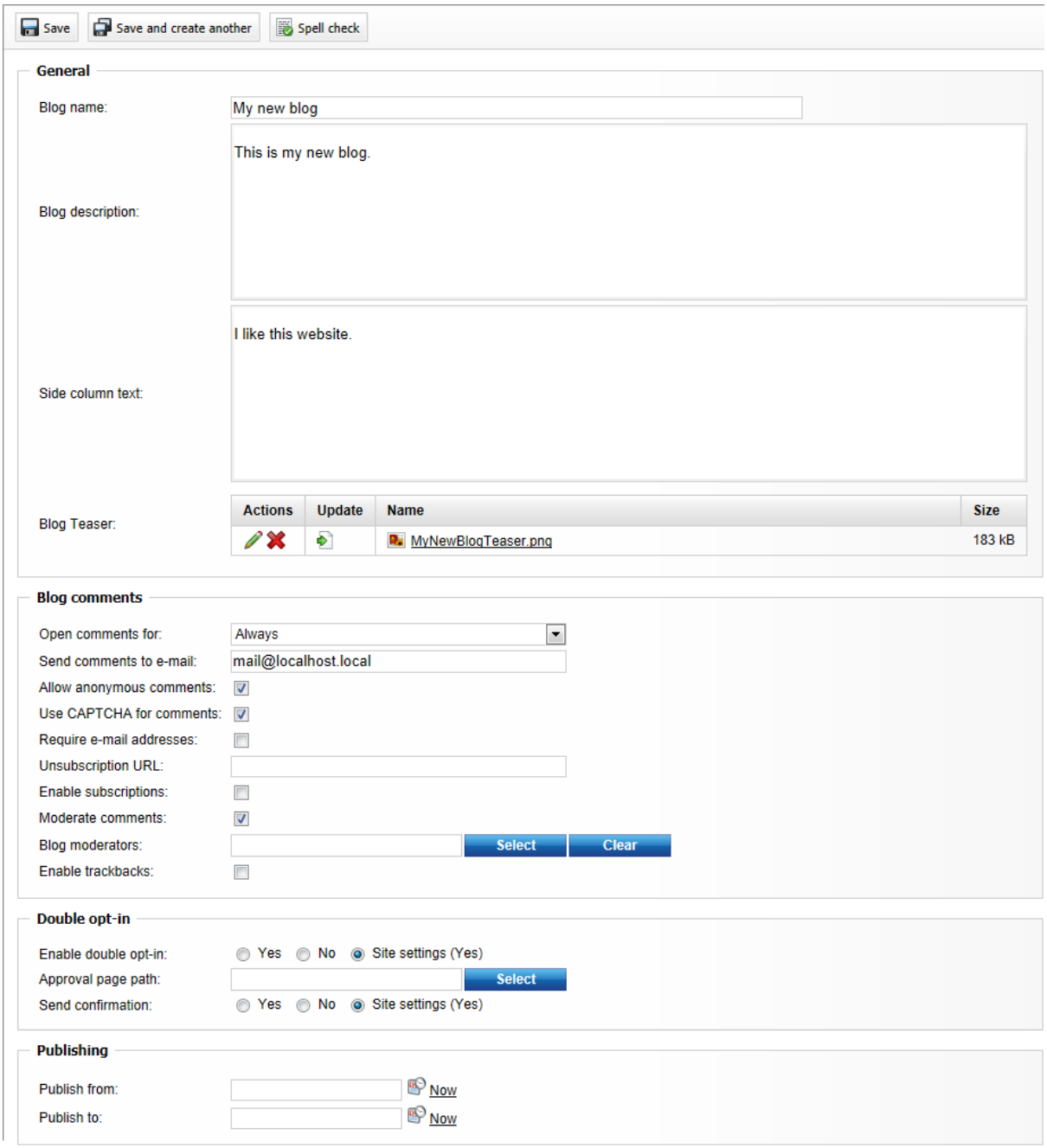
- **Blog name**: My new blog
- **Blog description**: This is my new blog.
- **Side column text**: I like this website.
- **Open comments for**: Always
- **Send comments to e-mail**: enter your e-mail address
- **Allow anonymous comments**: yes
- **Use CAPTCHA for comments**: yes (this will require a verification of the number displayed in the picture to avoid spam posts)
- **Moderate comments**: yes

The following fields are also available, but you don't need to enter them for the purpose of this example:

- **Blog Teaser** - blog teaser image displayed in blog lists.
- **Require e-mail address** - indicates if e-mail address is required when adding a blog comment.
- **Unsubscription URL** - URL of a page with the Blog post unsubscripiton web part, which handles blog post unsubscription requests from links contained in blog post notifications.
- **Enable subscriptions** - indicates if subscriptions to notifications about new blog comments are enabled.
- **Blog moderators** - users who can moderate blog post comments.
- **Enable trackbacks** - indicates if the trackbacking feature is enabled.
- **Publish from** - date and time from when the blog will be published on the site.
- **Publish to** - date and time until when the blog will be published on the site.

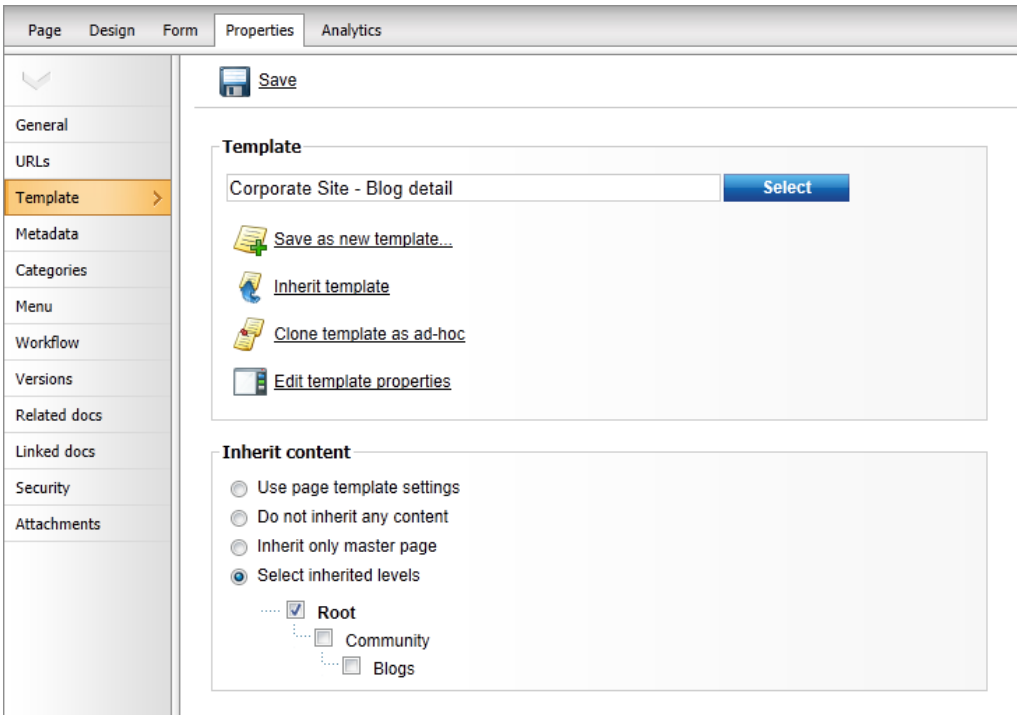
There are also settings related to blog comments subscriptions. To learn more about these settings and the functionality they affect, refer to the User subscriptions topic.

- **Enable double opt-in** - indicates if double opt-in should be enabled for blog post comments. When enabled, users are required to confirm their subscription by clicking a link that is sent to them in an e-mail.
- **Approval page path** - path to the page that contains the Blog post subscription confirmation web part. The subscription confirmation link that will be sent to users will point to this page.
- **Send confirmation** - indicates if an e-mail confirmation should be sent to user after they approve a subscription. If double opt-in is disabled, these confirmation e-mails are always sent.



Click **Save**. You have just created your blog. However, there are two additional steps you need to do:

2. Go to the **Properties -> Template** dialog, click **Select** and choose the **Corporate Site - Blog detail** page template. Set the **Inherit content** value to **Select inherited levels** and choose only the **Root**. It ensures that the page displays the master page and then your blog content.



Click **Save**.

Adding a blog on the live site

You can enable users to create blogs directly on the live site by adding the **New blog** web part to your site. Like this, users can only add the name and description of their new blog. All other blog properties, listed earlier in this article, will be taken from the web part's properties.

8.11.4 Adding posts to your blog

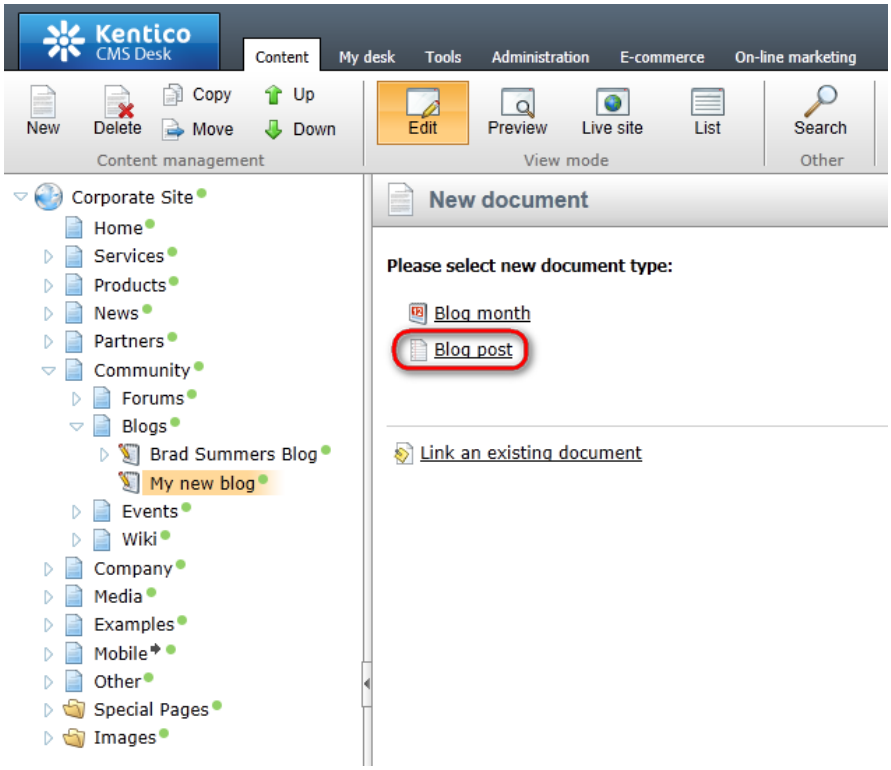
You can add new posts to your blog in two ways:

1. From the CMS Desk -> My Desk -> Blogs dialog

This dialog displays the list of blogs you are owner of (you can set the owner in the **Properties -> General** dialog of the blog document). Here you can click the **New post** ( ) icon and you will be displayed with form for inserting the post:

2. In the Content section

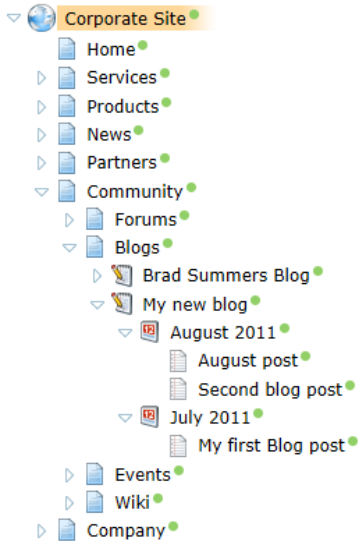
Choose the **Content** tab in **CMS Desk**, select the blog's main document and then click **New**. Choose to create a new **Blog post**:



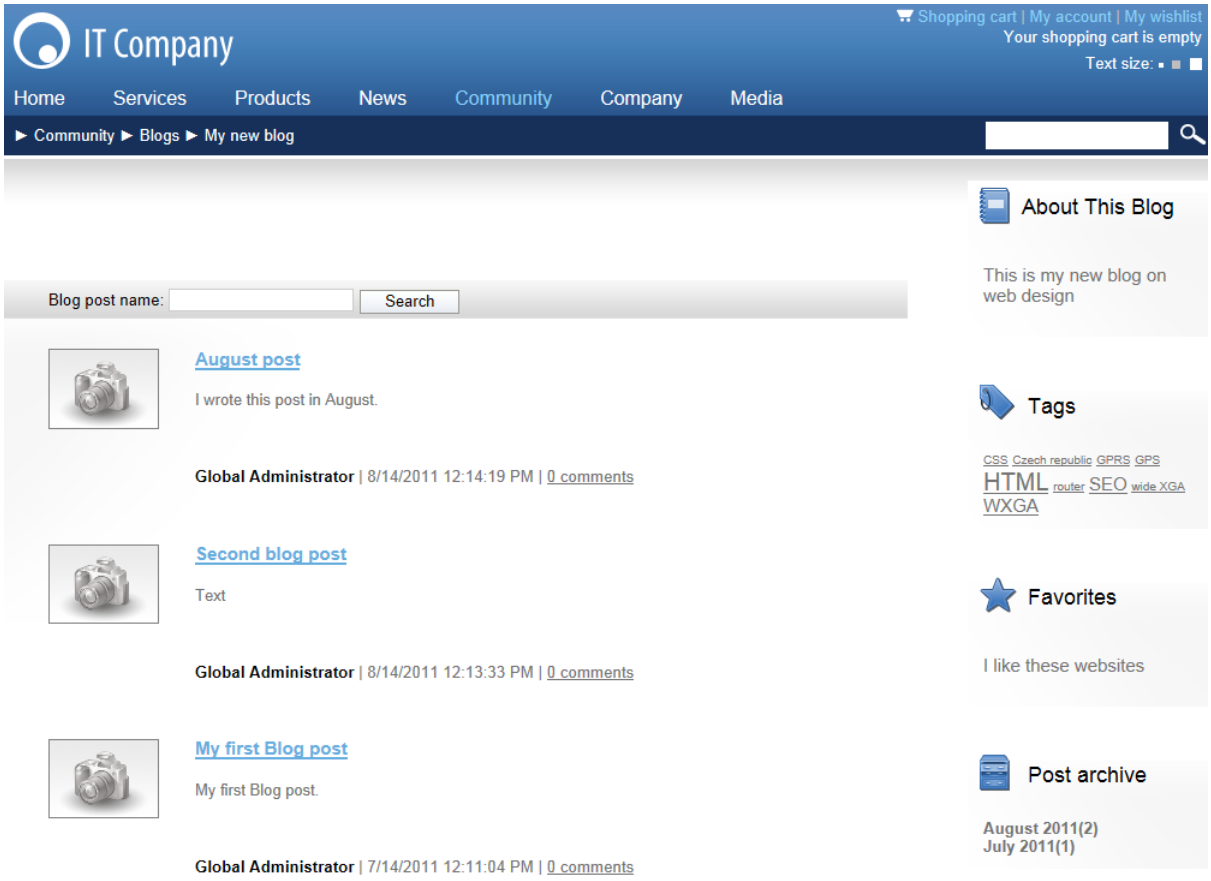
Enter the post content and click **Save**.

Document structure

In both cases, the blog post will be automatically saved under the appropriate month. If the month is not created yet, it's created automatically, so you do not need to care about creating a month. The months allows you to easily organize the blog posts. The structure of **Blog**, **Blog month** and **Blog post** documents looks like this:



When you sign out and go to your blog, you will see a page like this:



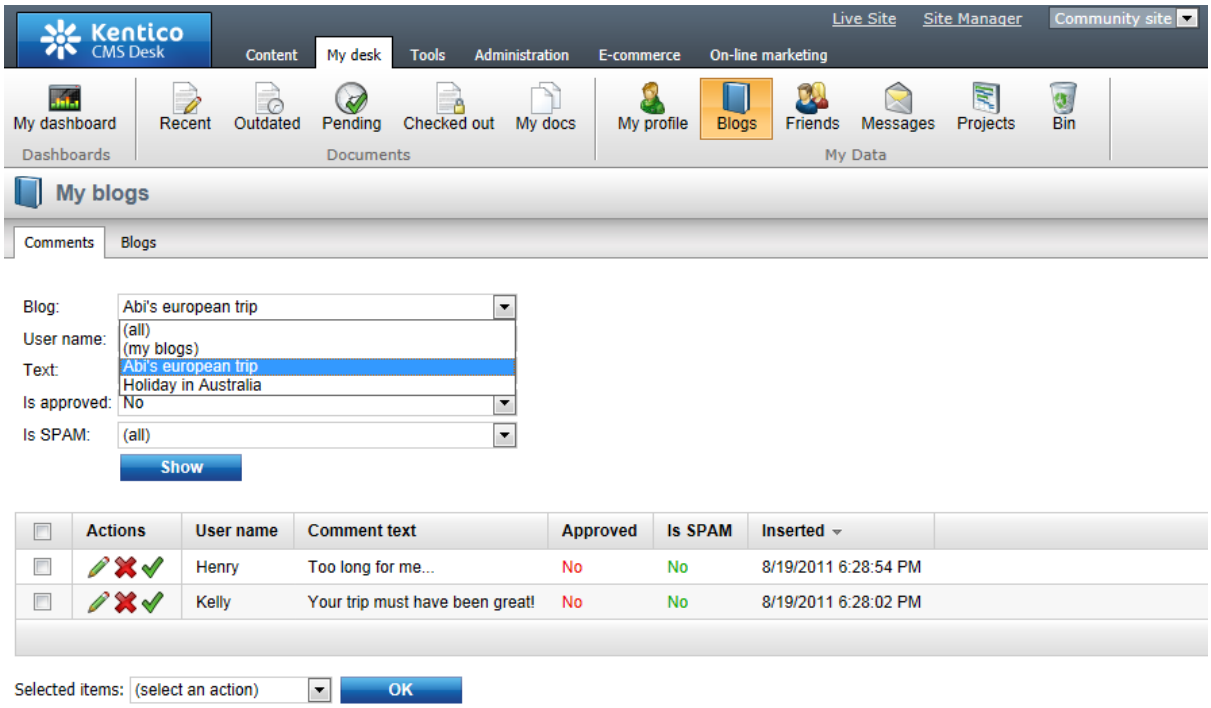
8.11.5 Moderating comments

The comments can be moderated in tree ways:

1. In the My Desk -> Blogs section

In this section, currently logged-in users can manage comments at all their own blogs. Neither the **Read** nor **Manage comments** permissions for the Blogs module are necessary for the user to perform blog comments management in this section.

Comments can be approved by clicking the **Approve** (✔) icon. More comments can be approved at once when you select them by the check-boxes, choose the **Approve** (✔) action in the **Selected items** drop-down list and click **OK**.

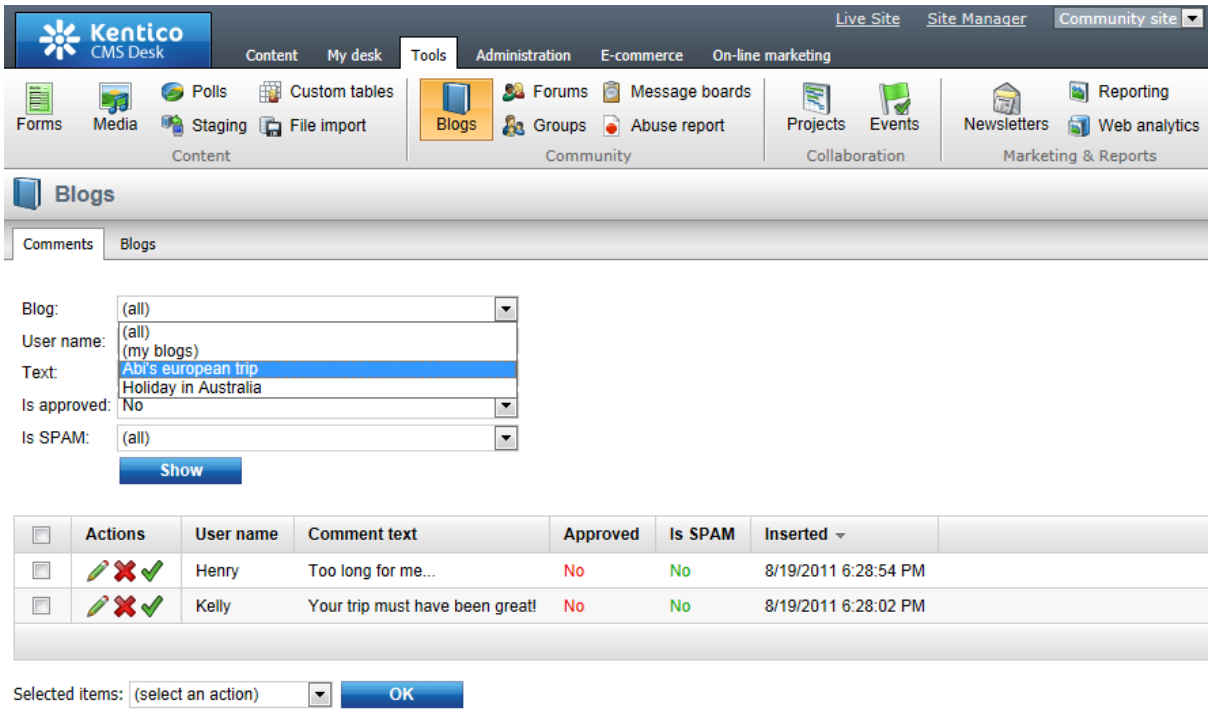


2. In the CMS Desk -> Tools -> Blogs section

In this section, management of blog comments on the site is possible based on the permissions:

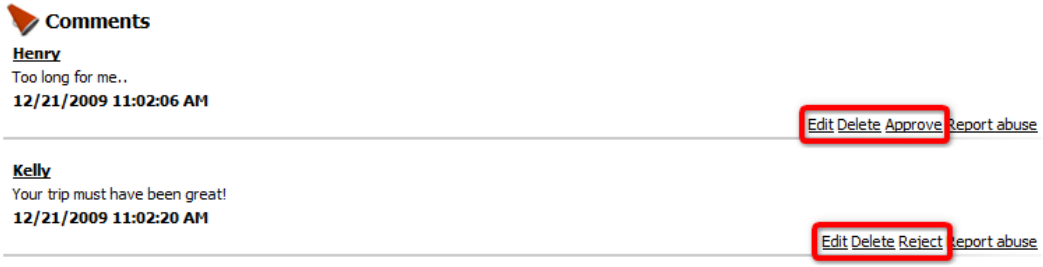
Global administrators and users with the **Manage comments** permission for the Blogs module can manage comments of all blogs on the current site. Blog **moderators** and blog **owners** can manage comments of their own blogs and blogs for that they are moderators. These permissions are reflected by the **Blog** drop-down list, which displays only those blogs where comments can be managed by the current user.

Comments can be approved by clicking the **Approve** (✔) icon. More comments can be approved at once when you select them by the check-boxes, choose the **Approve** (✔) action in the **Selected items** drop-down list and click **OK**.



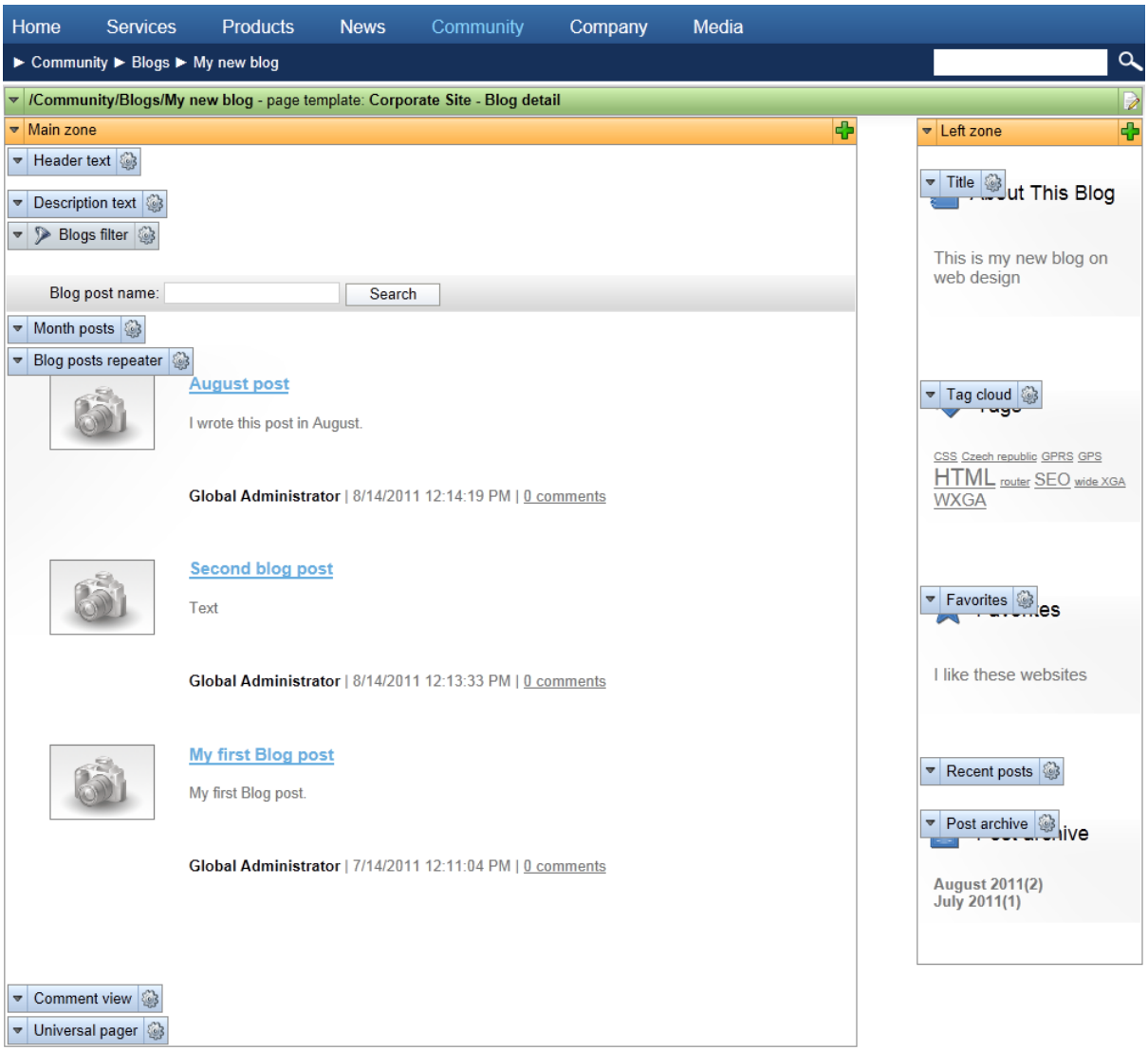
3. On-site management

When you're signed in and are entitled to manage the comments for the given blog, you are displayed with **Edit**, **Delete**, **Approve** and **Reject** buttons. These buttons are displayed to **blog owners**, **moderators** of the current blog, **global administrators** and users with the **Manage comments** permissions for the Blogs module.



8.11.6 Blog layout and design

You can configure the blog layout and design by changing the page template of the blog document. Locate your blog in the **CMS Desk -> Content** section and choose the **Design** tab. You will see a page like this:



The blog is displayed using the Blog page template. The web parts ensure displaying of the blog content, comments and boxes in the right-hand column.

There are two repeaters to display the posts:

- **rptMonthPosts** - this repeater is used to display posts in the given month
- **rptAllPosts** - this repeater is used to display the latest N posts when the blog is displayed without selecting a particular month

If you need to modify the layout of the blog posts, you need to modify the transformations. By default, the following transformations are used:

- **cms.blog.PostPreview** - the view in the list of posts
- **cms.blogpost.Default** - the detailed view

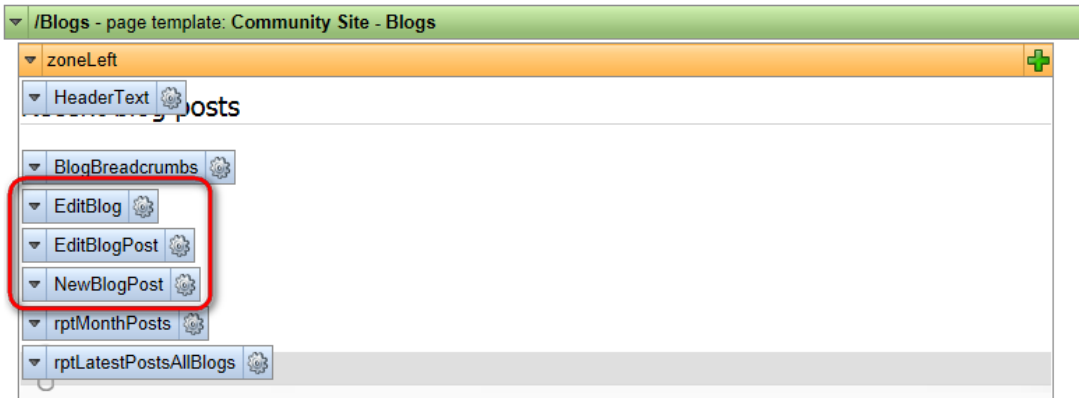
8.11.7 On-site management via User contributions

You can enable users to perform the following tasks using the User contributions module web parts:

- **Edit blog properties**
- **Edit or delete blog posts**
- **Add new blog posts**

A live example of this can be found on the sample **Community Starter site**, on the **Blogs** title page. If you go to CMS Desk's **Design** tab and view the page, you will see the following web parts, which enable the above mentioned functionalities:

- **EditBlog** - Edit contribution web part set up for enabling blog properties editing
- **EditBlogPost** - Edit contribution web part set up for enabling blog post editing
- **NewBlogPost** - Contribution list web part set up for enabling adding of blog posts



These web parts are inherited by the underlaying pages, which means they will be displayed when a blog or blog post is displayed. The following sections explain how to enable users to perform these tasks on the live site.

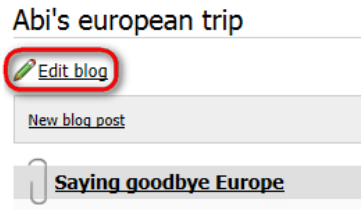
Edit blog properties

1. Create an **alternative form** for the **Blog** document type, which will contain the fields that you want to let users modify. If you are not familiar with the Alternative forms concept, please refer to the Alternative forms chapter first.

2. Add the **Edit contribution** web part to the blogs section title page and set its following properties:

- **Show for document types** - CMS.Blog
- **Alternative form name** - code name of the alternative form created in step 1
- **Edit button label** - Edit blog
- **Allow editing by users** - Document owner; this is the setting that makes the most sense as blog properties should be edited only by the blog's owner

If you go to the live site and display some blog placed under the blogs section title page, you should see the **Edit blog** link as highlighted in the screenshot below. After clicking the link, the alternative form will be displayed, letting blog owners edit the properties of the blog.



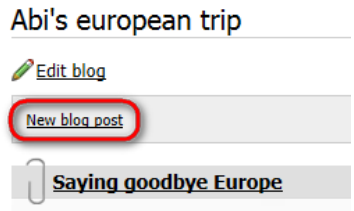
Add new blog posts

1. Create an **alternative form** for the **Blog post** document type, which will contain the fields that you want to let users to specify when creating the blog post. If you are not familiar with the Alternative forms concept, please refer to the Alternative forms chapter first.

2. Add the **Contributions list** web part to the blogs section title page. You only need to set the following properties, as the web part will not be used for displaying blog posts, but only for their inserting.

- **Show for document types** - CMS.Blog
- **Allowed new document types** - CMS.BlogPost
- **Alternative form name** - code name of the alternative form created in step 1
- **New item button label** - New blog post
- **Allow insert** - enable
- **Allow editing by users** - Document owner; this is the setting that makes the most sense as blog posts should be edited only by the blog's owner

If you go to the live site and display some blog placed under the blogs section title page, you should see the **New blog post** link as highlighted in the screenshot below. After clicking the link, the alternative form will be displayed, letting blog owners add new blog posts directly from the live site.



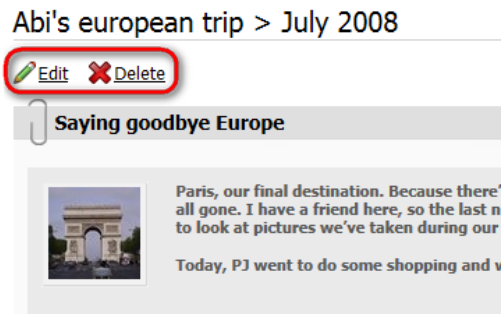
Edit or delete blog posts

1. Create an **alternative form** for the **Blog post** document type, which will contain the fields that you want to let users modify. If you are not familiar with the Alternative forms concept, please refer to the Alternative forms chapter first.

2. Add the **Edit contribution** web part to the blogs section title page and set its following properties:

- **Show for document types** - CMS.BlogPost
- **Alternative form name** - code name of the alternative form created in step 1
- **Edit button label** - Edit
- **Allow delete** - enable
- **Allow editing by users** - Document owner; this is the setting that makes the most sense as blog posts should be edited only by the blog's owner

If you go to the live site and display some blog post placed under the blogs section title page, you should see the **Edit** and **Delete** links as highlighted in the screenshot below. After clicking the **Edit** link, the alternative form will be displayed, letting blog owners edit the blog post. Using the **Delete** link, they can delete the blog post.



8.11.8 Settings

Settings of the Blogs module can be adjusted in **Site Manager -> Settings -> Content -> Blogs**. The following settings are available:

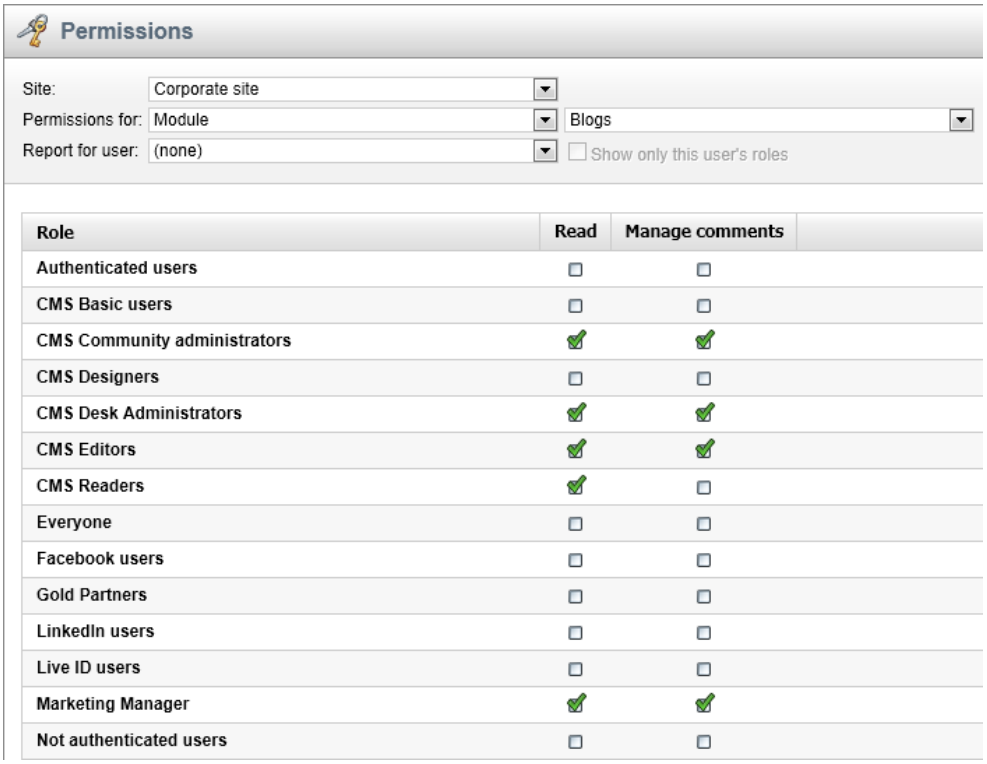
- **Send blog e-mails from** - e-mail address that will be used as the Sender ('From') address of notification e-mails.
- **Enable blog post trackbacks** - If checked, specified blog posts are pinged after the new blog post is saved; turn off this setting if you are creating your site on the production server to avoid creating trackbacks to your production server.
- **Use external service for trackbacks** - indicates if the external Scheduler Windows service should be used to process scheduled tasks which handle trackbacks.

- **Blog unsubscription URL** - URL of a page on which the Blog post unsubscription web part is placed; this is a special web part used for handling unsubscriptions from receiving notifications about new blog posts.
- **Enable double opt-in for blog post comments** - indicates if double opt-in should be enabled for blog post comments. When enabled, users are required to confirm their subscription by clicking a link that is sent to them in an e-mail.
- **Double opt-in approval page path** - path to the page that contains the Blog post subscription confirmation web part. The subscription confirmation link that will be sent to users will point to this page.
- **Double opt-in interval (hours)** - amount of time in hours, during which a user has to confirm their subscription request.
- **Send double opt-in confirmation** - indicates if an e-mail confirmation should be sent to user after they approve a subscription. If double opt-in is disabled, these confirmation e-mails are always sent.

8.11.9 Security

The Blogs module leverages the standard security used for all documents, as blogs are actually documents in the content tree. The Blogs module also has the following permissions in **Site Manager -> Administration -> Permissions**:

- **Manage comments** - allows members of the role to manage blog post comments
- **Read** - allows members of the role to view blog posts and blog post comments via the administration interface



To manage blog posts, you need to be one of the following:

- global administrator
- owner of the blog
- blog moderator
- granted with the **Manage comments** permission for the Blogs module in the **Administration -> Permissions** dialog

The blogs in the **My Desk -> Blogs** dialog are displayed only to the blog document owner.

8.11.10 Trackbacks

8.11.10.1 Trackbacks overview

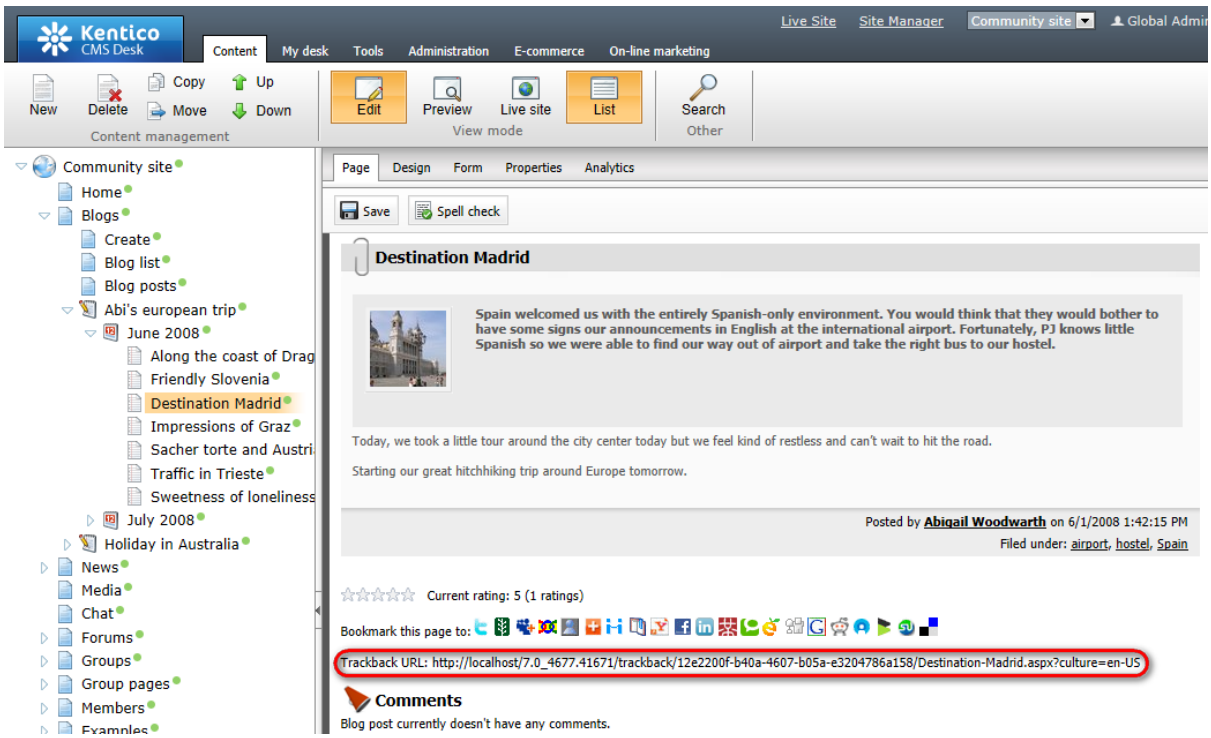
The **Trackbacks** feature allows sending of blog post links to other blogs when a new blog post is created. The link is typically, but not always, added as a blog comment in the target blog, as implemented in Kentico CMS. This can be performed not only within the same site, but also to completely different sites created not only with Kentico CMS.

Trackbacks are typically used to let readers of a topic-related blog posts know about your new blog post. You can find detailed information on trackbacks on the following Wikipedia page: http://en.wikipedia.org/wiki/Trackback

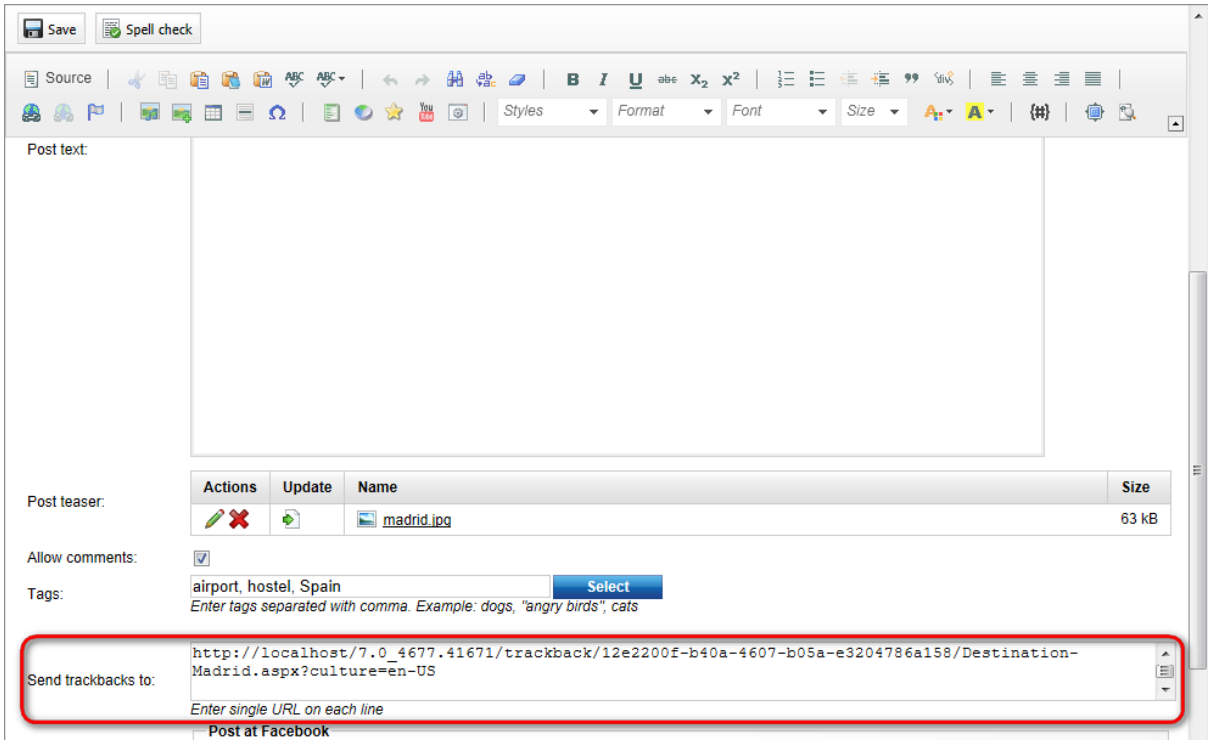
Read here to learn how to enable trackbacks on your site.

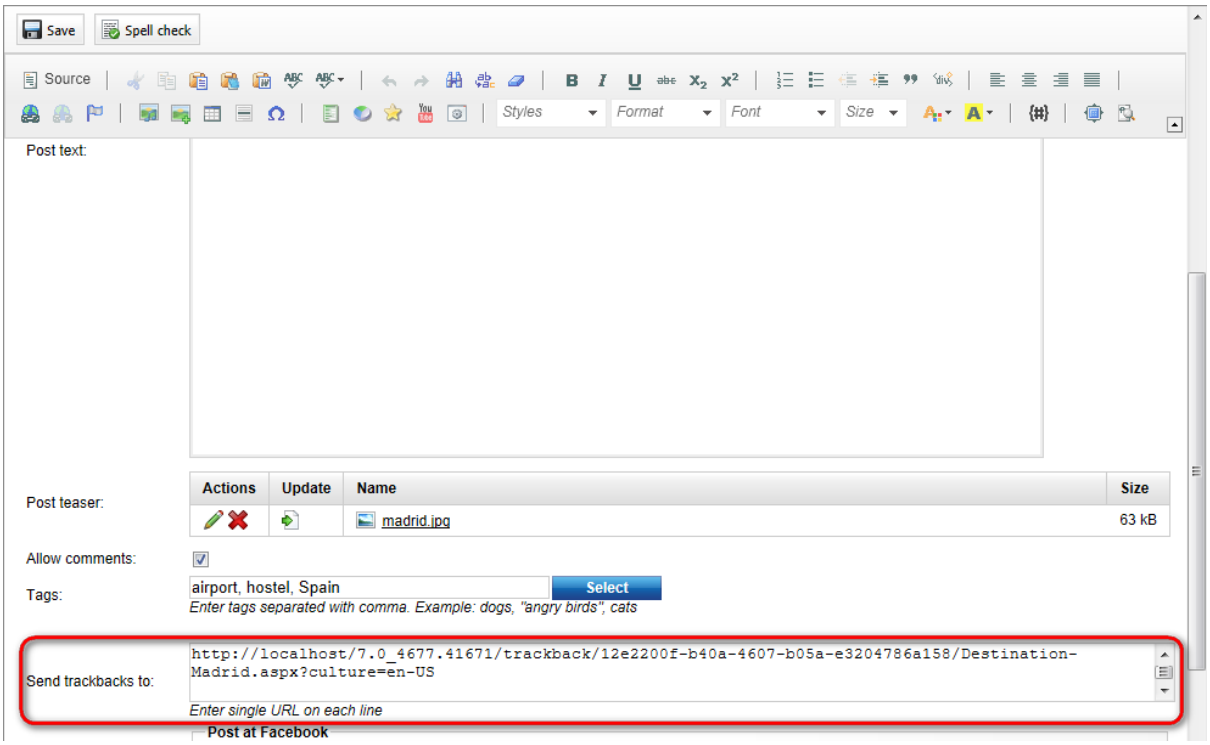
How it works

1. There is a Trackback URL published below the blog post. These URLs can be found on various blogs at sites developed not only with Kentico CMS.

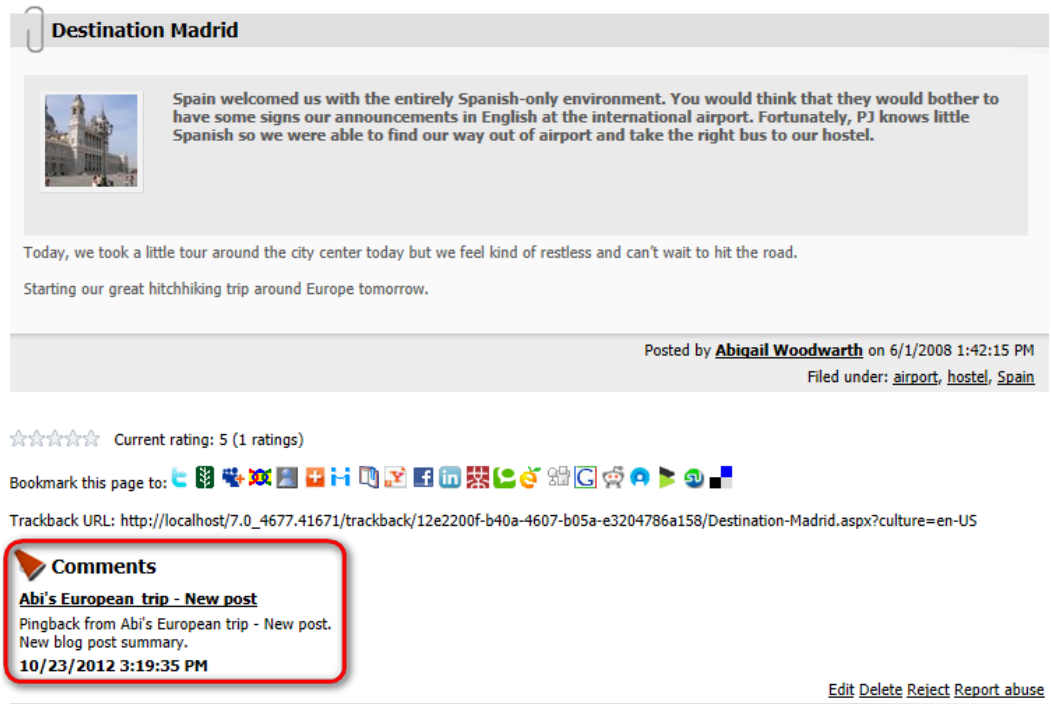


2. When users are creating a blog post, they can add such URLs into the **Send trackbacks to** field. More than one trackback URL can be entered, each on a new line.





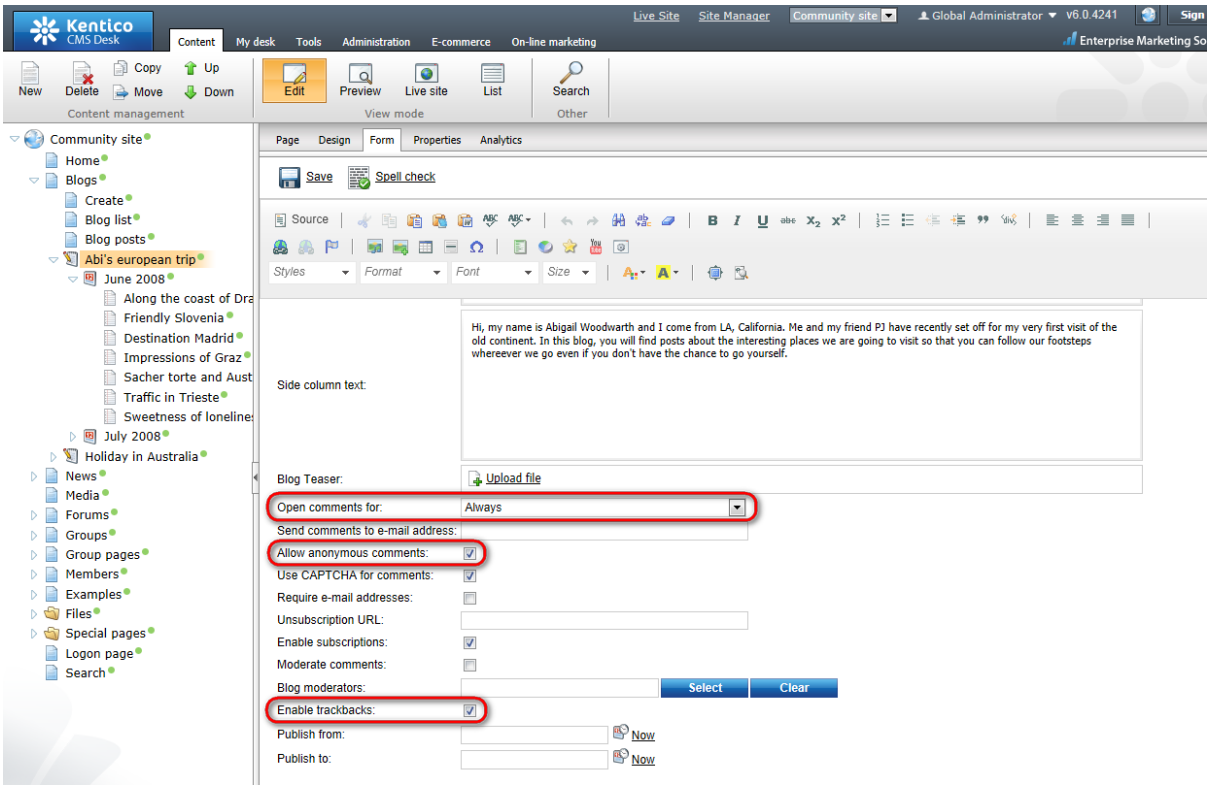
3. When the blog post is submitted, trackback ping is sent to the trackback URLs. If everything is configured correctly at the other blogs, a blog comment is added with the blog post link, title and summary, as you can see in the screenshot below.



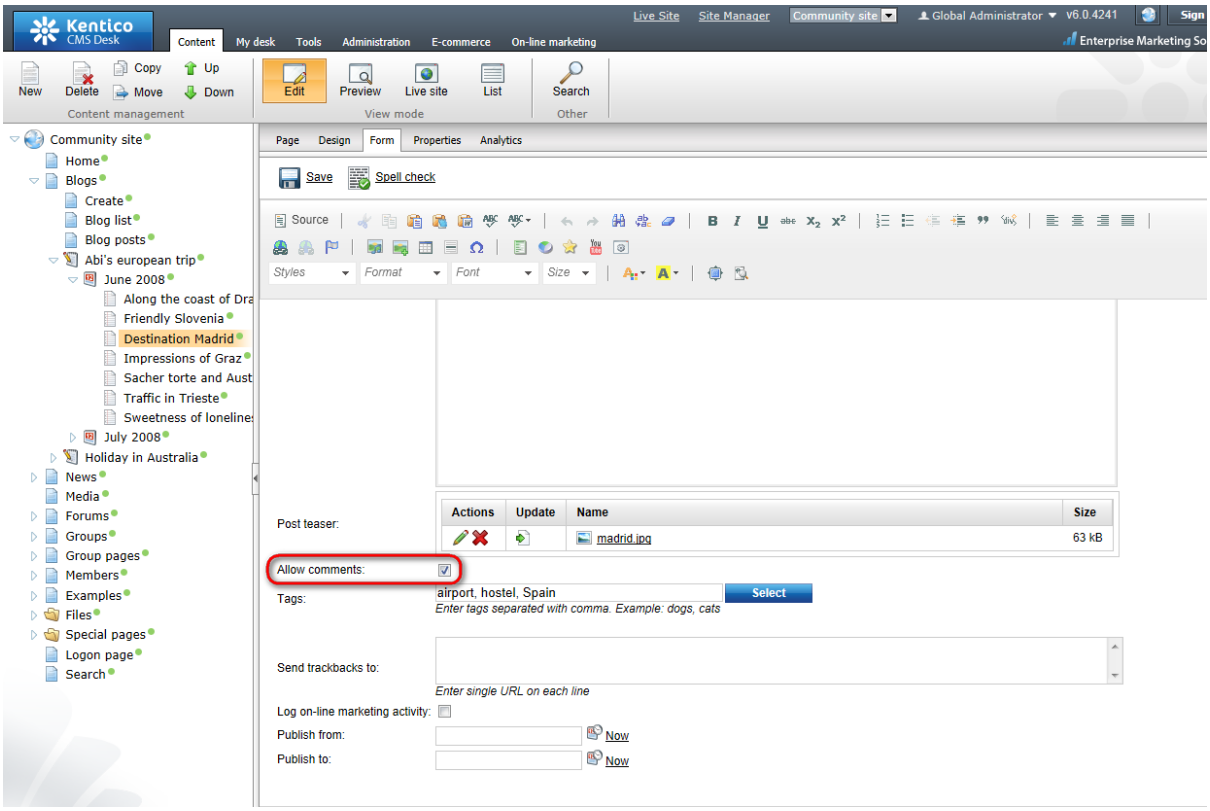
8.11.10.2 Enabling trackbacks

For the trackbacks to be functional in a particular blog, you need to enable the following options on the blog's **Form** tab:

- **Enable trackbacks** - this option enables displaying of the trackback URL with each blog post and inserting of trackbacks as blog comments
- **Allow anonymous comments** - trackback comments are in fact anonymous comments, so this option needs to be enabled for the comments to be inserted
- **Open comments for** - this option defines how long after the insertion of the blog post will inserting of blog comments be possible; trackbacks can also be inserted only within this period



The **Allow comments** option must also be enabled on the pinned blog post's **Form** tab so that the comment can be inserted.



8.11.11 Blog comments notifications

8.11.11.1 Who can be notified

When a new blog comment is added, notification e-mails can be sent to:

- **Blog owners**
- **Blog moderators**
- **Subscribers**

The text below explains to whom notification e-mails are sent under specific conditions:

New comment

1. has been added by the blog owner, blog moderator, user with the **Manage permission** or global administrator

- the comment is marked as APPROVED
- the e-mail is sent to: the blog owner (if their e-mail address is set in blog properties) and the subscribers

2. has been added by anybody else
a) the blog is moderated

- the comment is marked as NOT APPROVED
- the e-mail is sent to: the blog owner (if their e-mail address is set in blog properties) and the moderators

b) the blog is not moderated

- the comment is marked as APPROVED
- the e-mail is sent to: the blog owner (if their e-mail address is set in blog properties) and the subscribers

Comment editing

1. the comment has been switched from NOT APPROVED to APPROVED

- the e-mail is sent to: the subscribers

2. other comment changes

- no e-mails are sent

8.11.11.2 User subscriptions

Users can subscribe to receiving notifications about new blog comments at a blog post. It can be done in two ways:

Enabling subscriptions

You can enable users to subscribe to posts in a blog by checking the **Enable subscriptions** box on the **Form** tab of the corresponding blog document.

Enabling double opt-in

To make users confirm that the e-mail address they're subscribing with really exists and belongs to them, you can enable the double opt-in functionality. With double opt-in enabled, subscribers will be sent an e-mail with a confirmation link. They will have to click the link to create their subscription.

To enable double opt-in globally for all blogs:

1. Go to **Site Manager -> Settings -> Content -> Blogs**.

2. Turn on the **Enable double opt-in for blog post comments** setting.

3. Specify how long you want the confirmation links to be valid in the **Double opt-in interval** setting.

The following two steps are optional. If you omit them, the system will use the default approval page located in *~/CMSModules/Blogs/CMSPages/SubscriptionApproval.aspx*.

4. Place the **Blog post subscription confirmation** web part on a page. Adjust its properties according to your needs.

5. Enter the page's path into the **Double opt-in approval page path** setting.

Double opt-in should now be configured for all blogs. You can override the settings for each existing blog by selecting the corresponding blog document, switching to the **Form** tab and adjusting the settings in the **Double opt-in** category.

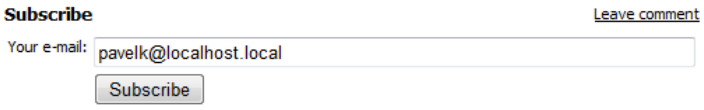
The functionality uses the **Blogs - Subscription request** e-mail template. You can insert the confirmation link into the template using the *{%SubscriptionLink%}* macro.

Subscribing to a blog post

Users can subscribe when leaving a comment, by checking the **Subscribe me to this blog post** check box. In this case, notifications will be sent to the e-mail address specified in the **E-mail** field above.

Users can also subscribe without leaving any comment, by clicking the **Subscribe** link at the top of the **Leave comment** form.

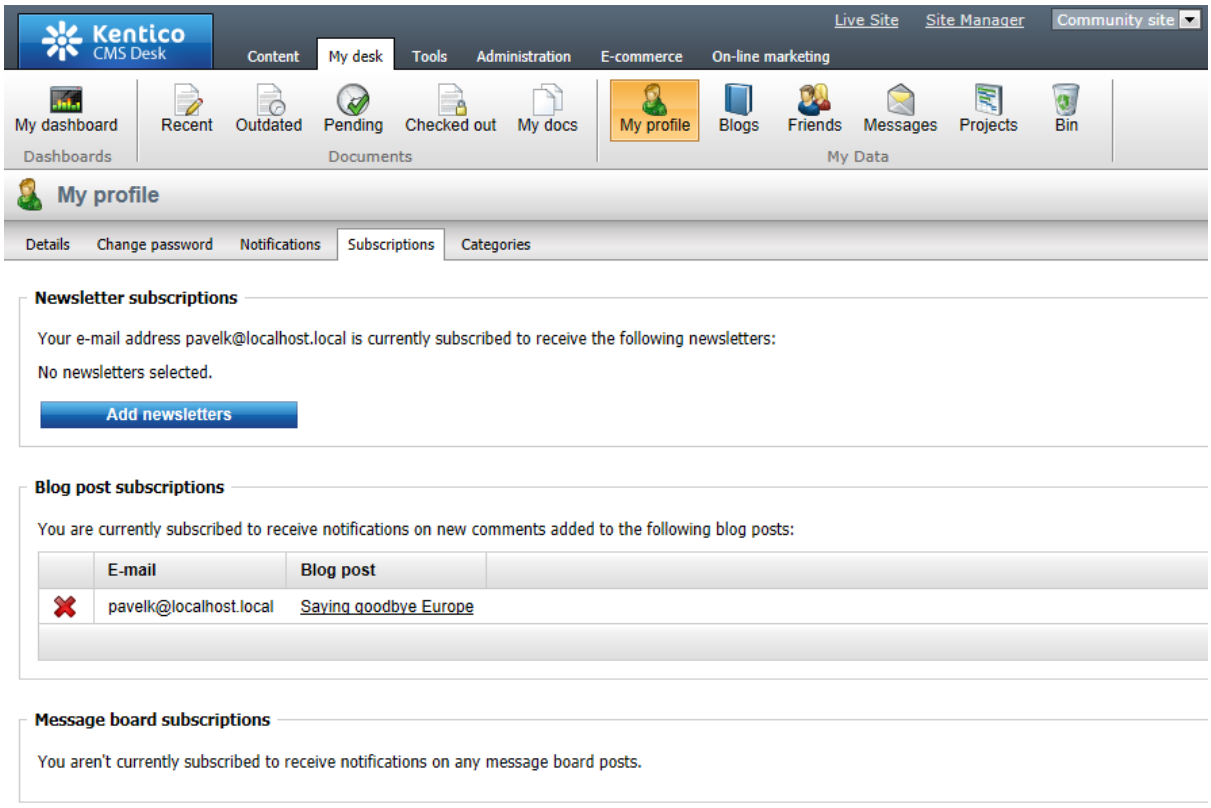
This displays a subscription form. The subscription can be created by entering the e-mail address to the **Your e-mail** field and clicking the **Subscribe** button.



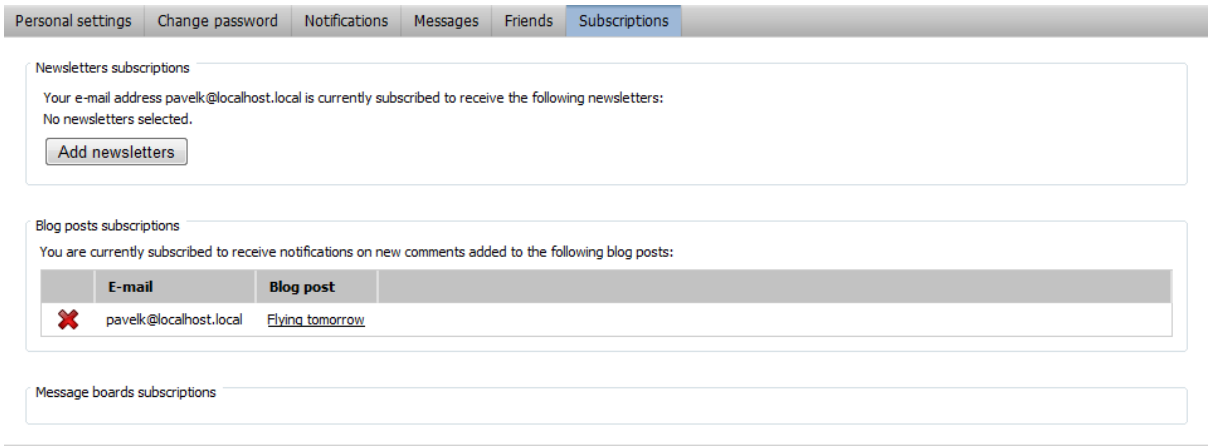
Managing subscriptions

Users can view and manage their subscriptions in the following two places:

1. Users with access to **CMS Desk** can manage their subscriptions on the **My Desk -> My profile -> Subscriptions** tab.



2. On the live site, users can manage their subscriptions in the **My account** web part. The **Display my subscriptions** property of the web part must be enabled for this to be possible.

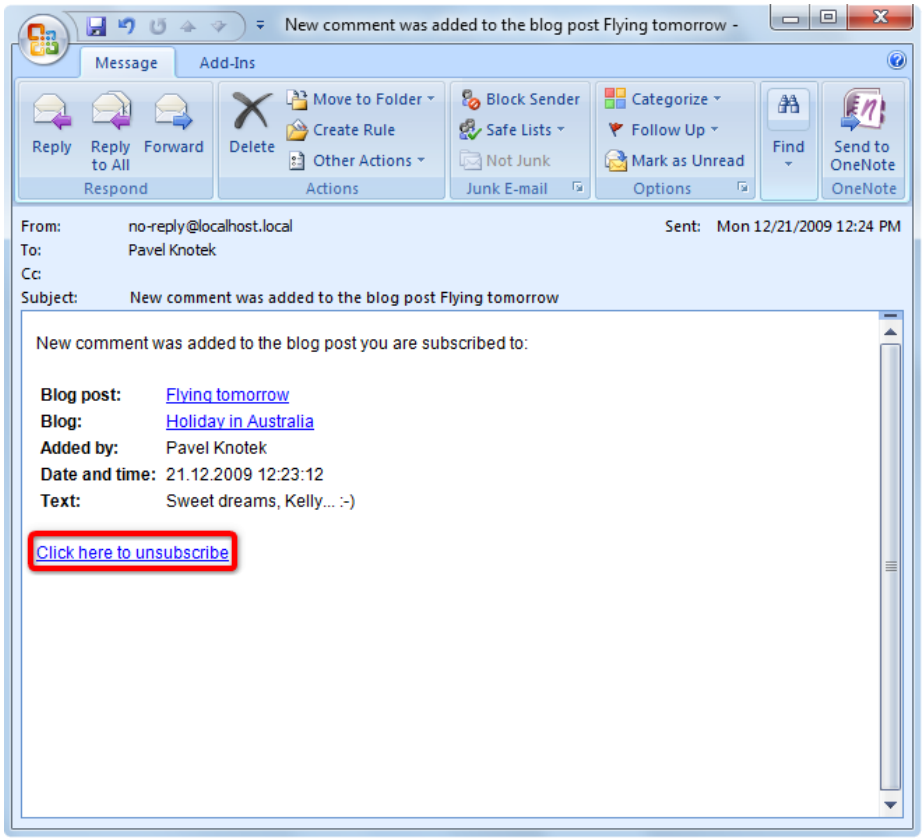


In both these locations, users can ✖ **Delete** the subscription. If double opt-in is enabled and a subscription hasn't been yet confirmed by clicking the link in the e-mail the user received, they can also ✔ **Approve** the subscription.

Unsubscription link configuration

Subscribers can also unsubscribe by clicking the unsubscription link, which is present in each

notification e-mail.



For this to work, you have to do the following two things:

1. Place the **Blog post unsubscription** web part onto a page. It is recommended to create a special page for this purpose, as you can see at **Community site -> Special-pages -> Blog unsubscribe**. You can set only one specific property of the web part - **Confirmation text** - this is the text that will be displayed after successful unsubscription.

2. Set the URL of the page created in step 1 as the **Unsubscription URL** property of the blog. This can be done two ways:

- In **Site Manager -> Settings -> Content -> Blogs**, by settings the **Blog unsubscription URL** property. This is the default value that will be used by default, if no other URL is set.
- If some different URL is set in the option mentioned above, you can set the value of the **Unsubscription URL** property on the blog's **Form** tab. This value overrides the one set in **Site Manager -> Settings -> Content -> Blogs**.

8.11.11.3 E-mail templates

There are three different e-mail templates that can be used when sending notifications about new blog comments, depending on the recipient of the notification:

- **Blog owners** - e-mails are based on the **Blogs - Notification to blog owner** e-mail template
- **Blog moderators** - e-mails are based on the **Blogs - Notification to blog moderators** e-mail

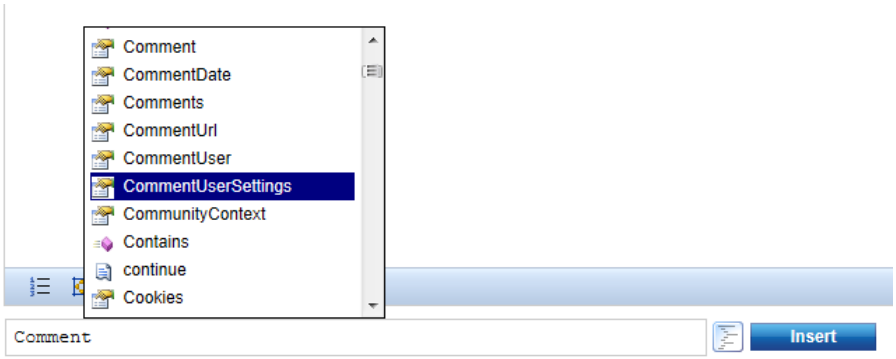
template
- **Subscribers** - e-mails are based on the **Blogs - Notification to blog post subscribers** e-mail template

The following macros can be used in the e-mail templates:

Data macros

- **Blog.XXX** - where *XXX* represents a column of the *CONTENT_Blog* table or the *CMS_View_Tree_Joined* view
- **BlogPost.XXX** - where *XXX* represents a column of the *CONTENT_Blog* table or the *CMS_View_Tree_Joined* view
- **Comment.XXX** - where *XXX* is a column of the *CONTENT_BlogComment* table
- **CommentUser.XXX** - where *XXX* is a column of the *CMS_User* table
- **CommentUserSettings.XXX** - where *XXX* is a column of the *CMS_UserSettings* table

Example: {%CommentUser.Email%}



Source macros

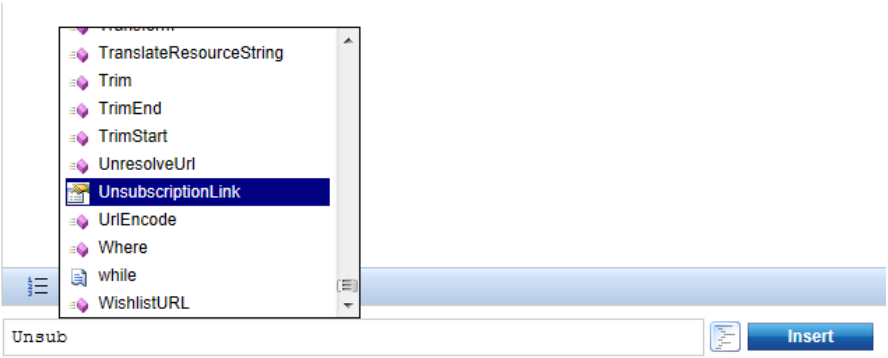
a) Links

- **BlogPostLink** - blog post link
- **BlogLink** - blog link
- **UnsubscriptionLink** - unsubscription link

b) Others - these macros are present due to backward compatibility purposes

- **UserFullName** - the same result as *{%Comment.CommentUserName%}*
- **CommentUrl** - the same result as *{%Comment.CommentUrl%}*
- **Comments** - the same result as *{%Comment.CommentText%}*
- **CommentDate** - the same result as *{%Comment.CommentDate%}*
- **BlogPostTitle** - the same result as *{%BlogPost.BlogPostTitle%}*

Example: *{%BlogPostLink%}*



8.11.12 MetaWeblog API

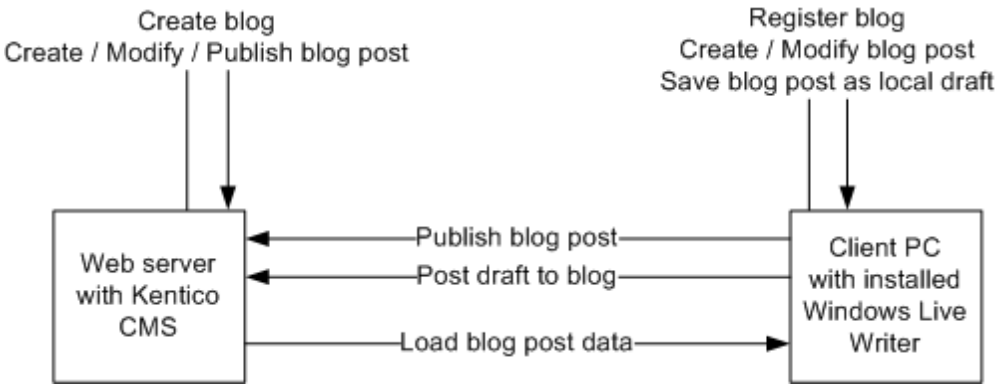
8.11.12.1 Overview

MetaWeblog API is a programming interface that enables external programs to get and set the text and attributes of blog posts. The API uses the XML-RPC protocol for communication between client applications and the blog server.

MetaWeblog API integration enables bloggers to write their blog posts using an application like Windows Live Writer (WLW) or Microsoft Word 2007 (or higher) on their local computer, even when they are offline; please note that in examples in this chapter WLW in version 14 is used. Then it enables transfer of these blog posts to the Kentico CMS web application, either by publishing them directly on the live site or just saving them as a draft to be published later.

How it works

The diagram below shows which actions can be performed in both Kentico CMS web application and Windows Live Writer, and the blog post transfer possibilities between the two applications.



The following actions need to be taken in order to use Windows Live Writer to write Kentico CMS blog posts:

1. User creates a blog in Kentico CMS (either via CMS Desk or on the live site).
2. User installs Windows Live Writer and registers their new blog in it.
3. Now the user is ready to create new and modify existing blog posts directly from WLW and transfer the changes into Kentico CMS.

In the **MetaWeblog API** subchapter, you can also learn how to publish a new blog post created with Windows Live Writer to a Kentico CMS Blog, how to manage blog posts and different cultural versions of your blog and finally, how to adjust the settings and security options related to **MetaWeblog API**.

Please note

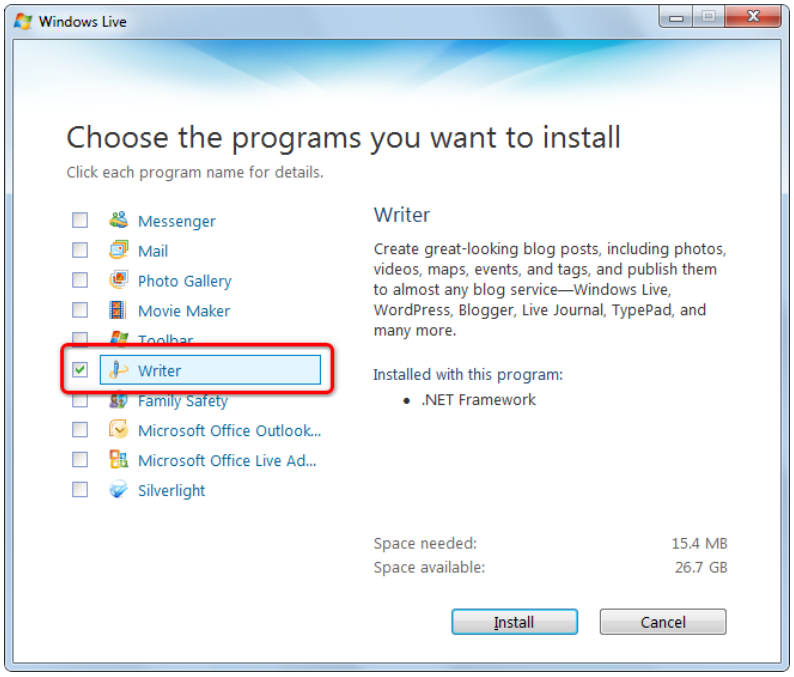
MetaWeblog API cannot be used by users authenticated by third-party authentication services (LiveID, OpenID etc.).

8.11.12.2 Windows Live Writer installation

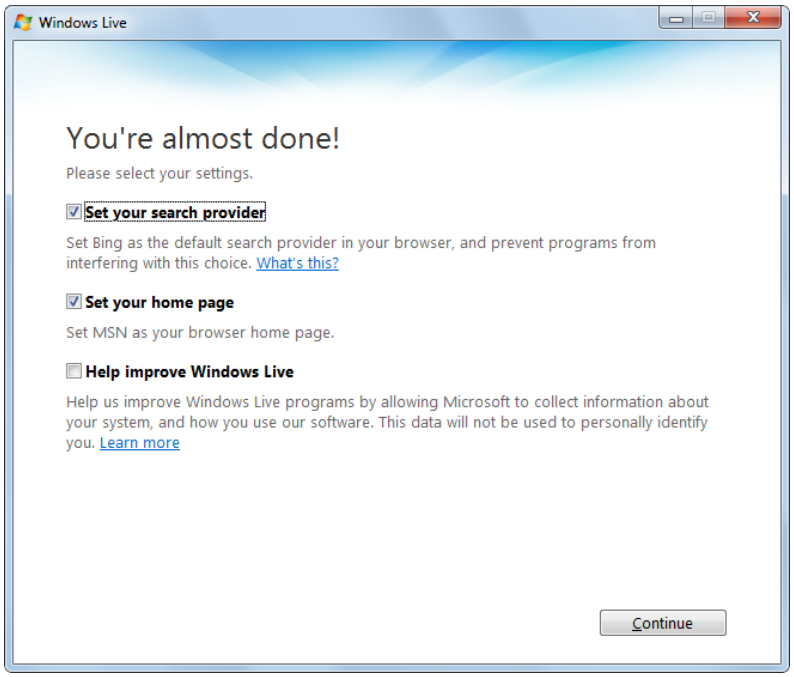
The following steps need to be taken in order to install Windows Live Writer on your machine:

1. Download Windows Live installer from http://download.live.com/writer.

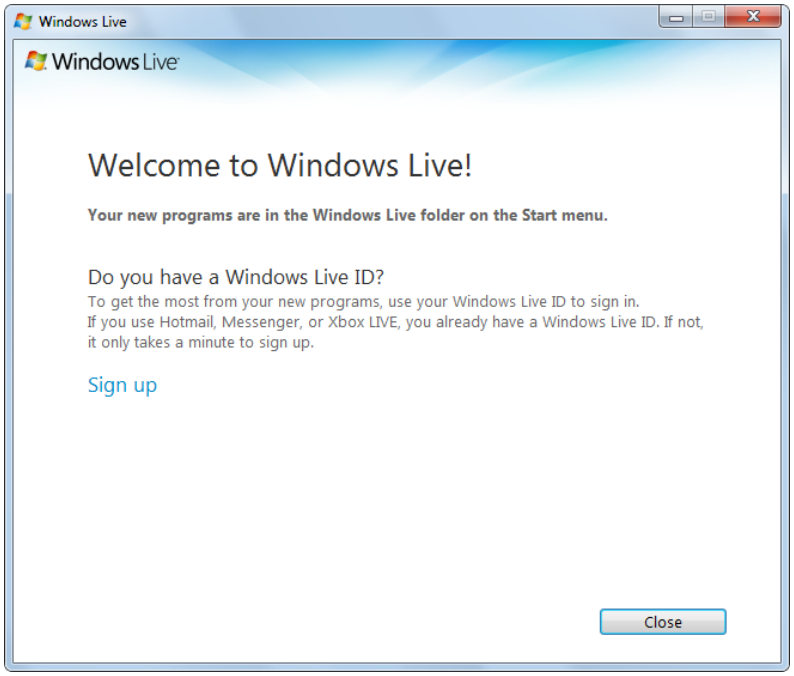
2. Execute the installer. Choose the **Writer** option from the list of available programs. Other options are not directly related to the Windows Live Writer. Choose as you wish and click **Install**.



3. When the installation finishes, it offers you some additional settings, which are not directly related to Windows Live Writer. Choose as you wish and click **Continue**.



4. You will get to the final screen, where you are offered to sign up for a Live ID. This again is not directly related to Windows Live Writer, so do as you wish and finally click **Close** to finish the installer.

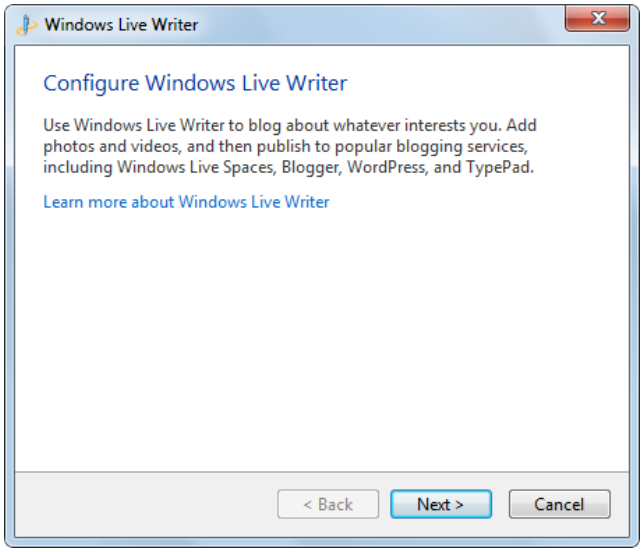


8.11.12.3 Registering blog account

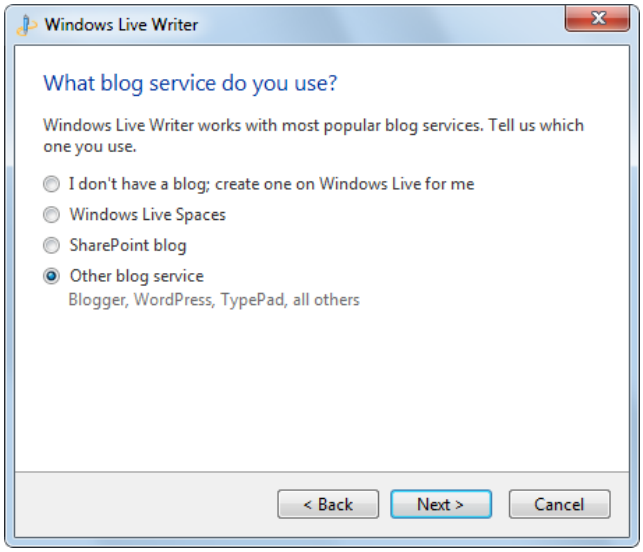
When you run Windows Live Writer for the first time, a wizard will appear, letting you configure connection to your on-line web service. After the initial setup, the wizard is still accessible under the **Tools -> Accounts -> Add** or **Blogs -> Add blog account** menu options.

The following text describes how to connect Windows Live Writer to a blog in a Kentico CMS instance.

1. The first step of the wizard is just informational. Besides some very basic info, it offers you the *Learn more about Windows Live Writer* link, which redirects you to some in-depth info about the program. Follow it if you wish and click **Next** to proceed further.



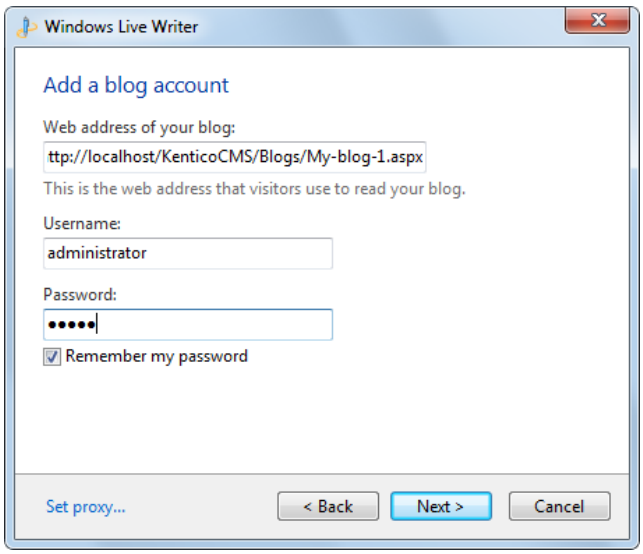
2. This step requires you to select the type of blog service. Select **Other blog service** and click **Next**.



3. Now you need to enter the URL of the blog and the logon credentials of the blog owner. Enter the following details:

- **Web address of your blog**: full Live URL of the blog's title page, e.g. *http://localhost/KenticoCMS_0322/Blogs/My-blog-1.aspx*
- **Username**: user name of the blog owner, e.g. *administrator*
- **Password**: blog owner's password (blank passwords are not supported), e.g. *12345*
- **Remember my password**: enable this option if you don't want to enter the password each time you use the application

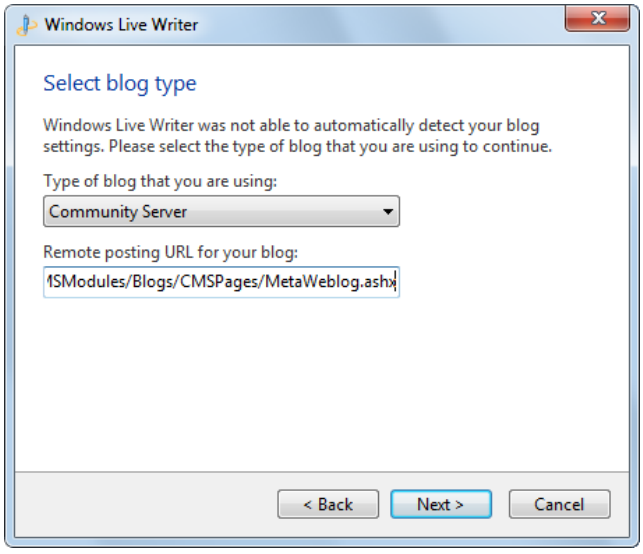
Click **Next** to proceed.



4. In the fourth step, you need to select the type of the blog and the posting URL of the blog. Enter the following details:

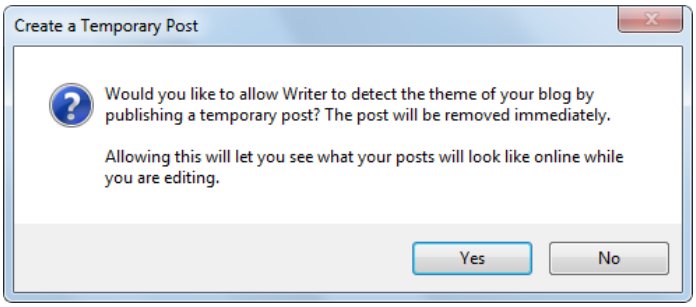
- **Type of blog that you are using**: *Community Server* or *MetaWeblog API* (the first one provides some extra functionality in WLW)
- **Remote posting URL for your blog**: URL of the MetaWeblog API handler, which is always in format *http://<application path>/CMSModules/Blogs/CMSPages/MetaWeblog.ashx* (e.g. *http://localhost/KenticoCMS/CMSModules/Blogs/CMSPages/MetaWeblog.ashx*)

Click **Next**.

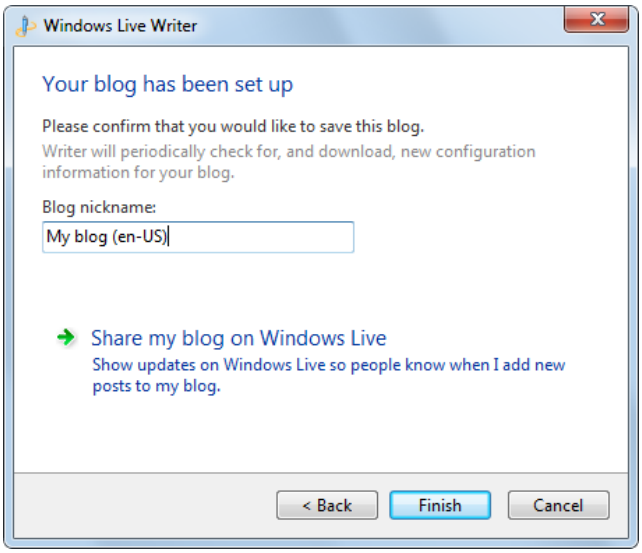


5. Now a dialog pops up, asking you if you want to detect the theme used for the blog so that you can see how the blog post will look like live when you are editing it. Click **Yes** to perform this action. If you click **No**, you can still do it later manually by choosing the **View -> Refresh theme** option from

Windows Live Writer menu.



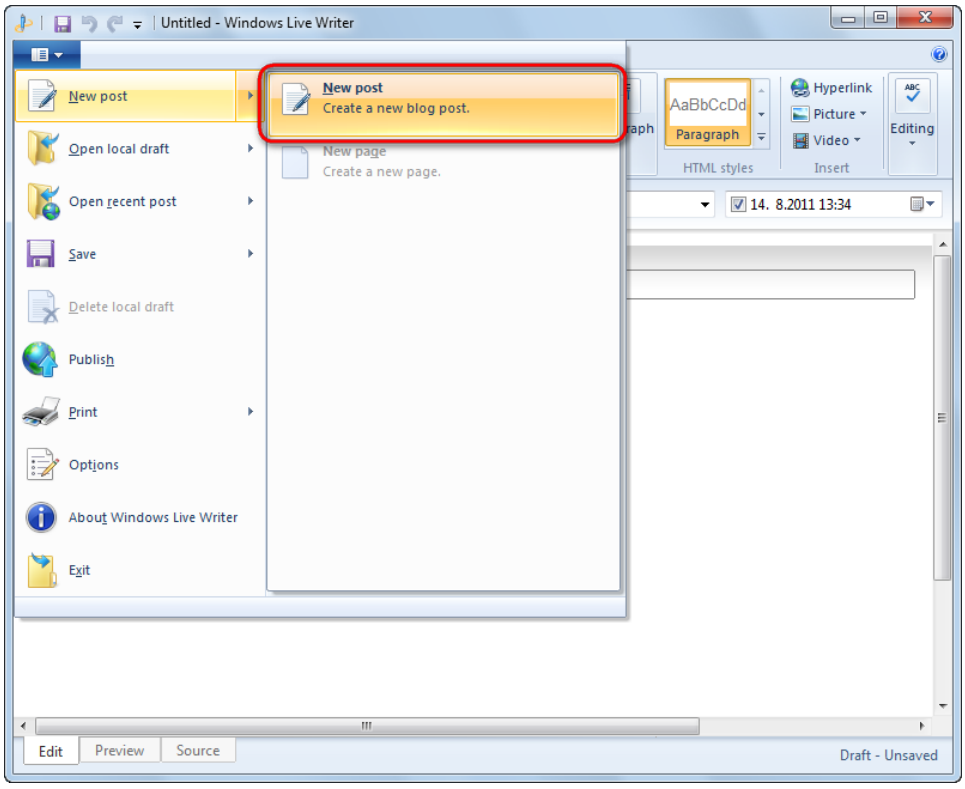
6. In the final step, you just need to enter the **Blog nickname**, which is just a name under which the blog will appear in WLW user interface. Enter whatever you wish and click **Finish**. You blog is now registered and you can start blogging from within Windows Live Writer.



8.11.12.4 Publishing first blog post

In the following example, you will learn how to create a new blog post with Windows Live Writer and publish it to the Kentico CMS blog registered in the previous example.

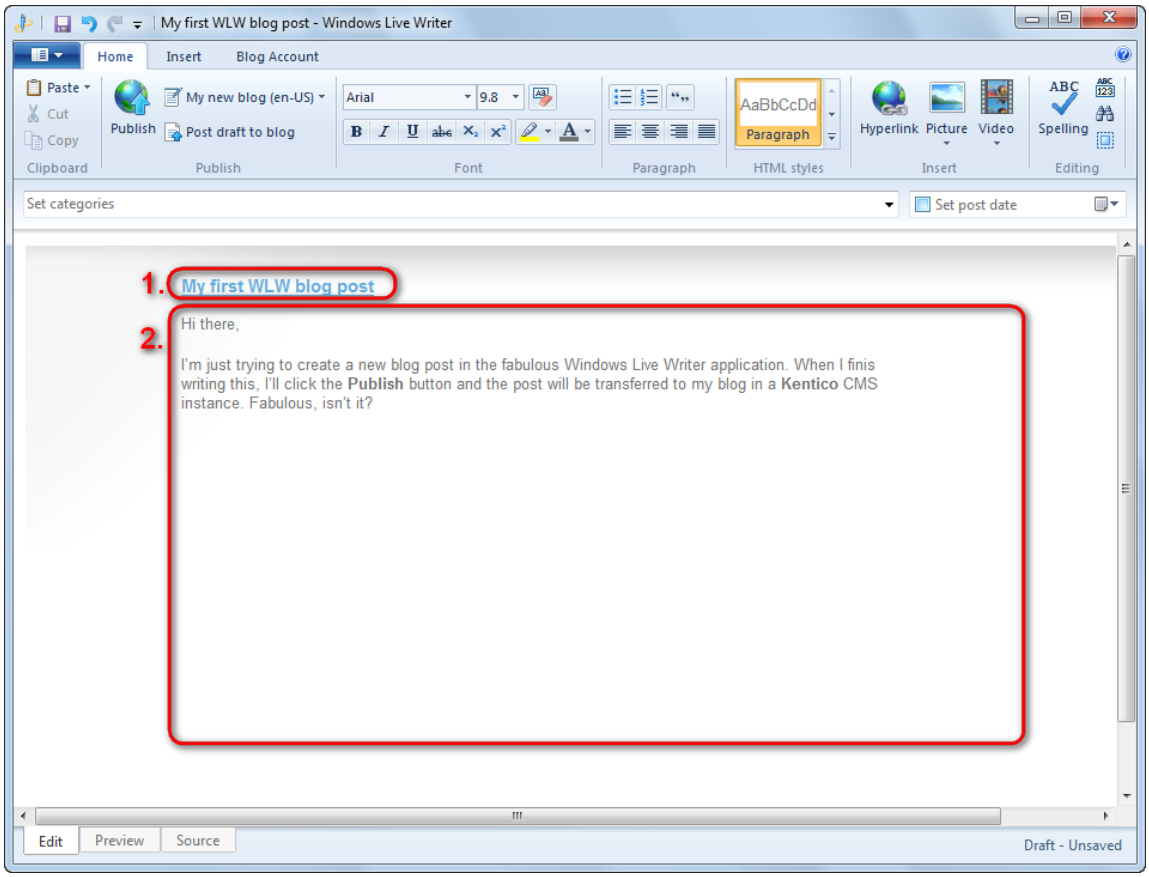
1. Open Windows Live Writer and select **New post** from the **File** menu (or click **Ctrl+N**).



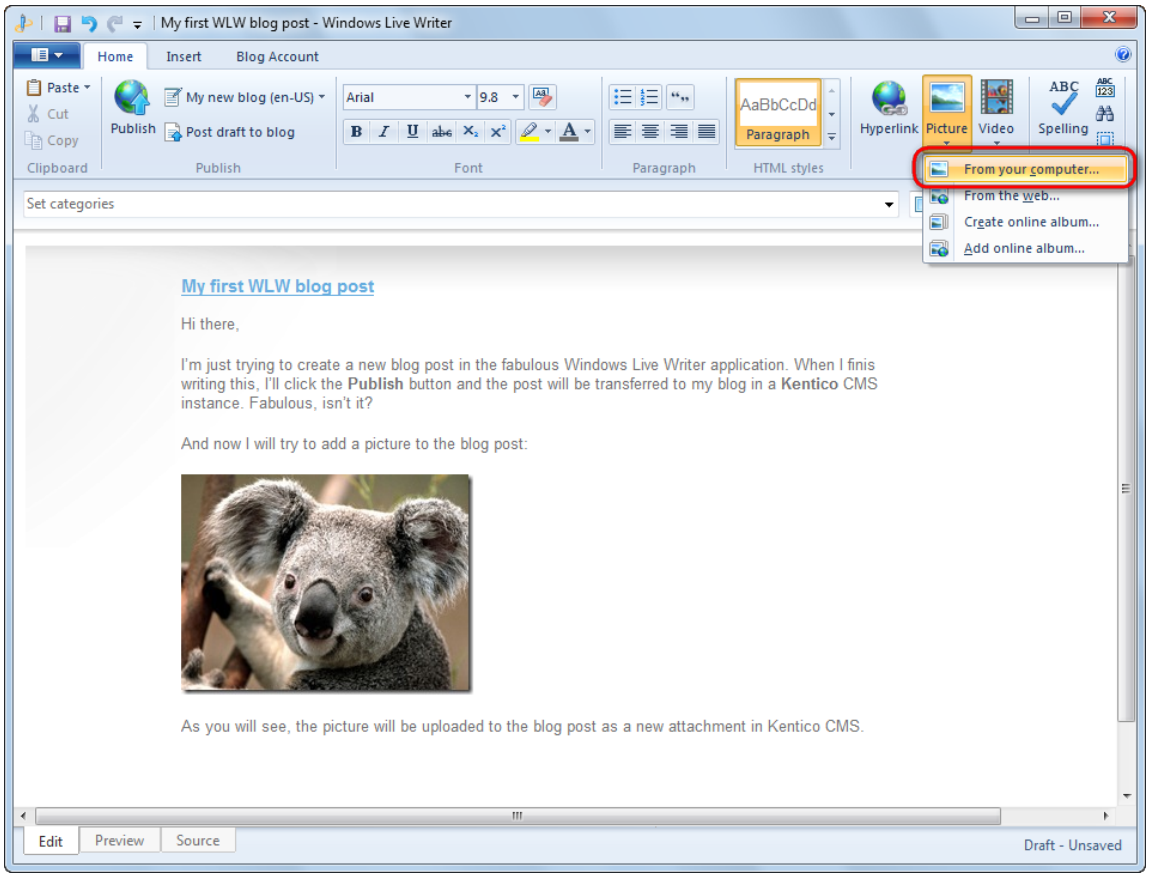
2. The gray rectangle at the top (1.) is where you enter the title of the blog post. What you enter here will be used as a value of the **Post title** field in your Kentico CMS blog post.

The text area below (2.) is where you enter the text of the blog post itself. This will be used for the **Post text** field, and in case you enabled the automatic summary (learn here how to do it), its beginning will be used for the **Post summary** field too.

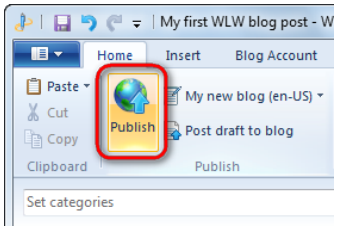
Enter some title and blog post text.



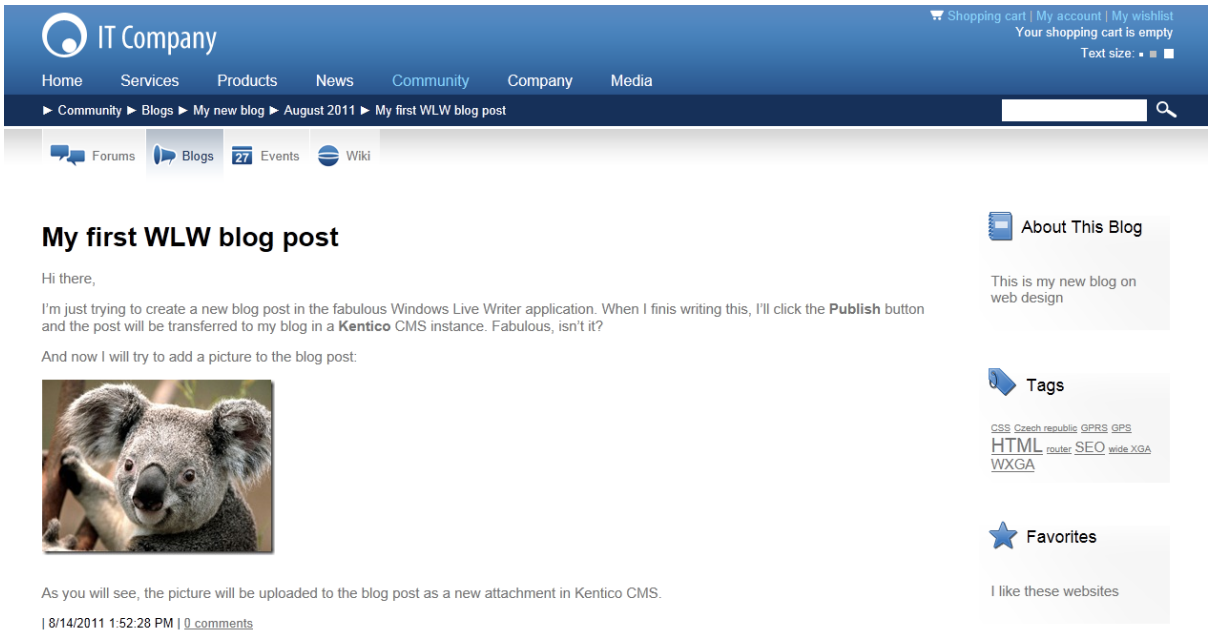
3. Now let's try adding an image. It is quite easy with Windows Live Writer, just click the **Picture...** option in the **Insert** menu on the right and select some image from your local disk. You should achieve that your blog post looks somewhat similar to the screenshot below.



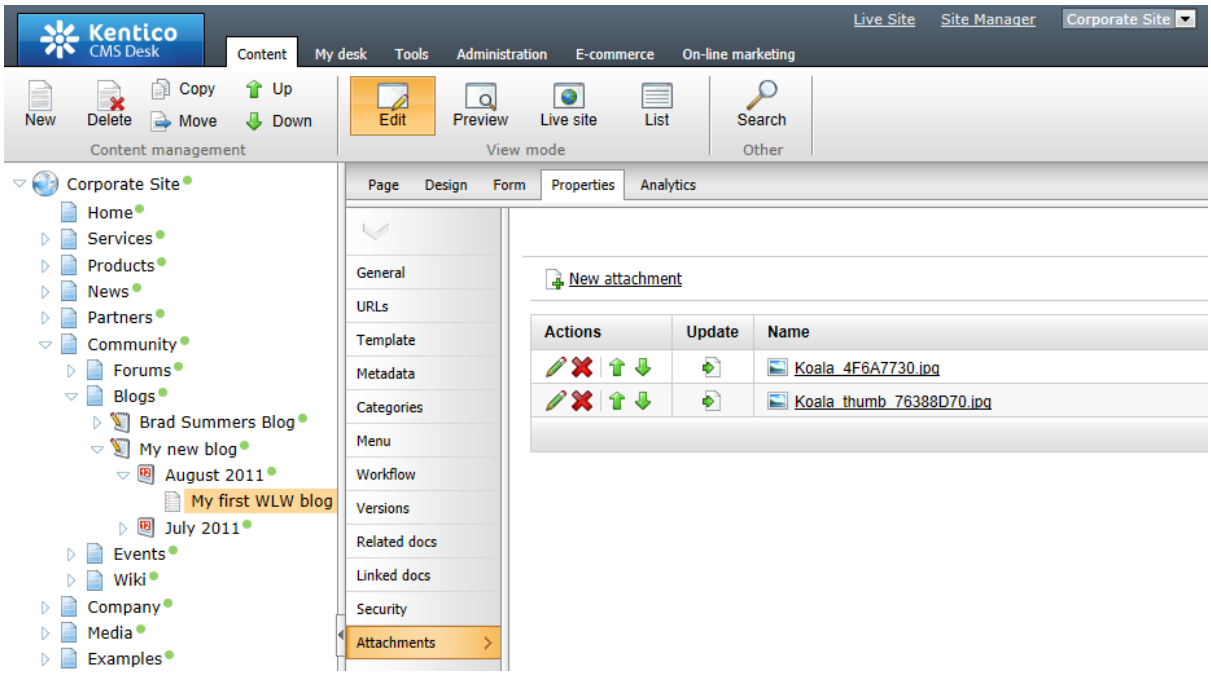
4. Now that your blog post is ready, we can publish it to the live site. Click the **Publish** button in the top toolbar.



5. When the publishing finishes, your blog will be displayed in your browser and you will see the new post published.



6. If you view the blog post in CMS Desk and switch to its **Properties -> Attachments** tab, you will see that the image has been uploaded as an attachment of the post. There are two versions of the image - the first one is the original full-size image, while the second one is its thumbnail used in the post text (e.g. in the screenshot above).

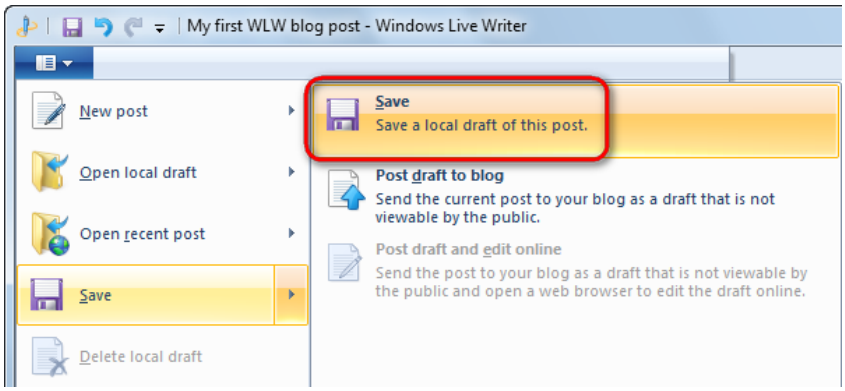


8.11.12.5 Managing blog posts

This chapter describes some additional options that you have when creating or editing a blog post.

Save local draft

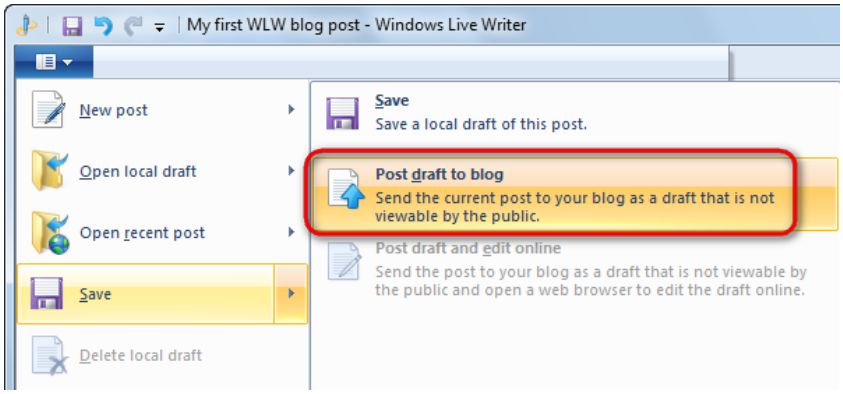
Current local version of the blog post is saved as blog post local draft; each blog post can have only one local draft. This action doesn't propagate any changes into the CMS.



Post draft to blog

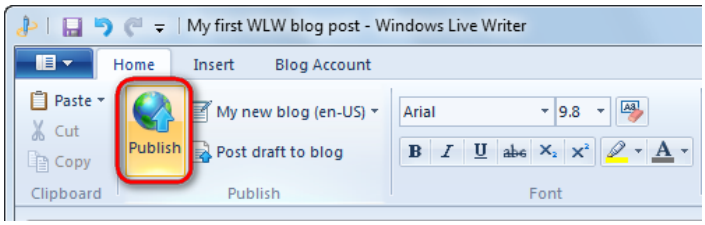
Current local version of the blog post is posted to blog, but its local draft is not removed. Depending on workflow settings, the following two situations can occur in the CMS:

a) **Blog post document is under workflow** - new minor version of the blog post is created (posted version is not published)
b) **Blog post document is not under workflow** - blog post is published immediately; this means that this action is identical to the *Publish* action, i.e. the *Post draft to blog* action makes sense only in case that workflow is enabled for blog post documents



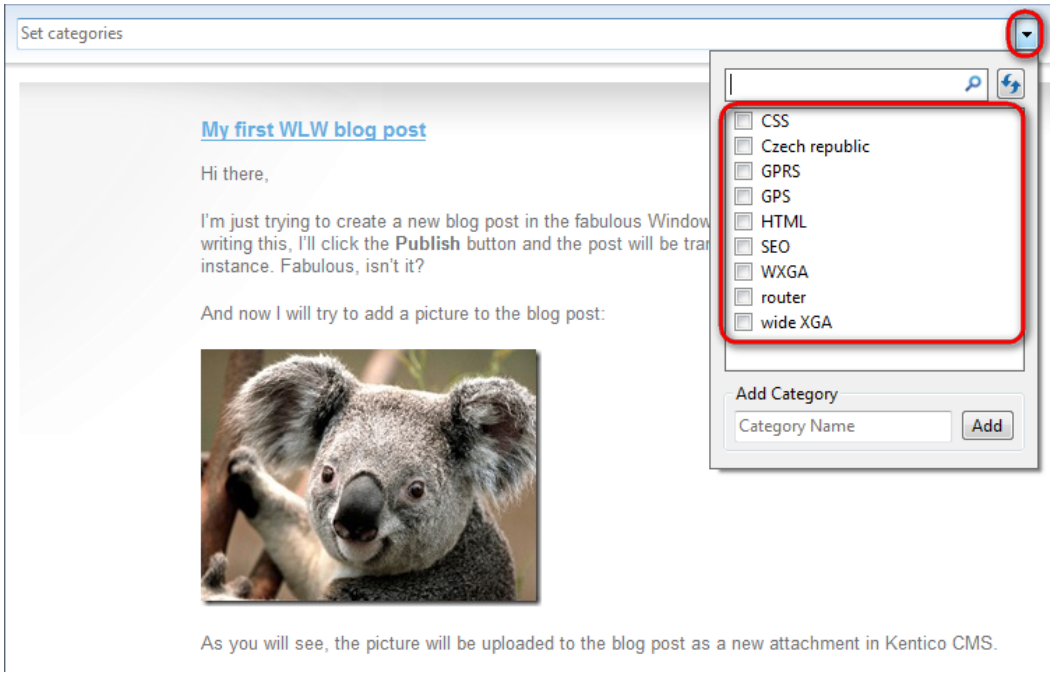
Publish

Current local version of the blog post is posted to blog and published immediately. Local draft of the published blog post is automatically removed.



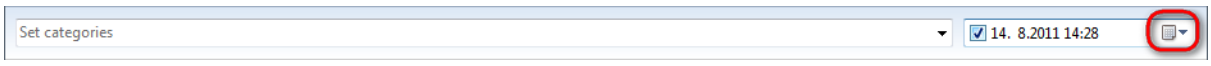
Tagging

The tag selector in WLW enables you to tag the blog post with tags from the parent blog's Kentico CMS tag group.The tag group of the blog post itself is not reflected here at all.



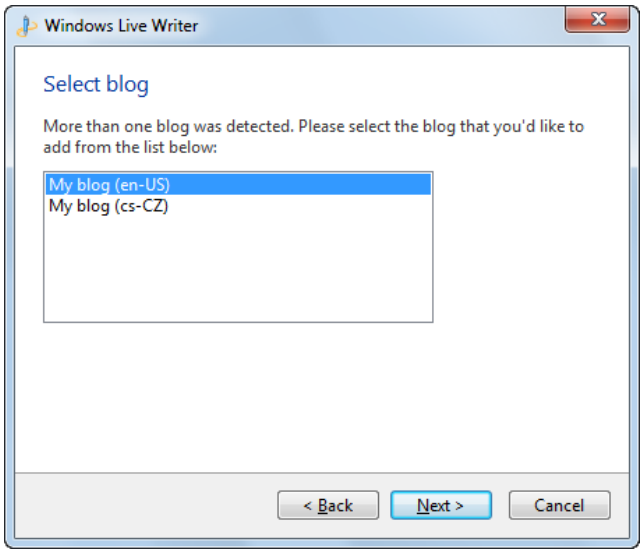
Publish date

You can specify a date here, which will be propagated into the **Publish from** field on the **Form** tab in CMS Desk. The value determines the date and time when the blog post gets published on the live site. It will be converted to the user's time zone before it is set as the **Publish from** property of the blog post in Kentico CMS. If no time is set here, the blog post is published immediately.



8.11.12.6 Multilingual support

In case that your blog exists in more than one cultural version, you will get the following dialog displayed after blog type selection (step 4) when registering your blog account.

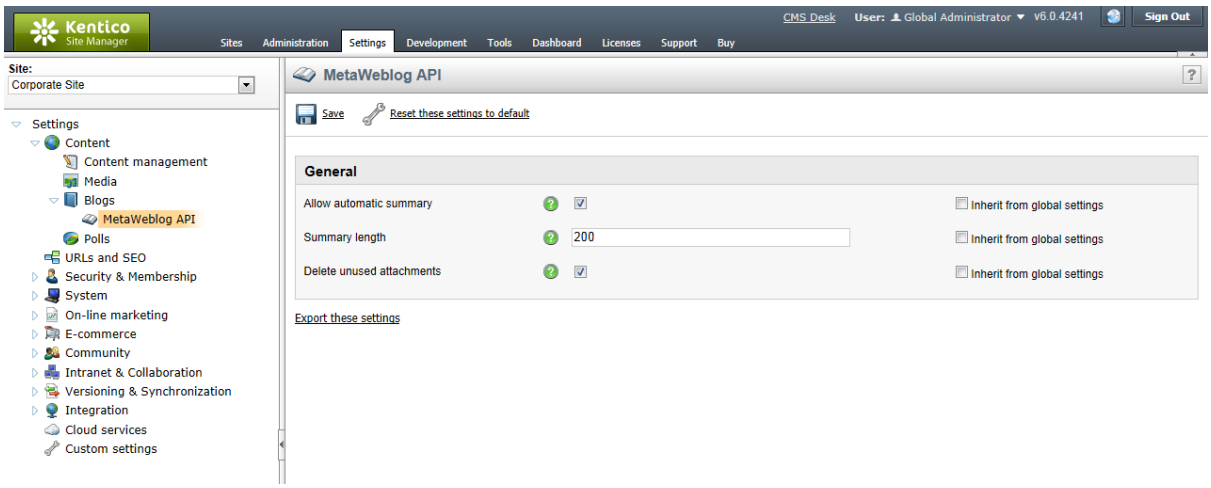


In the dialog, you need to choose which cultural version will the account be registered for. This means that you need to register **one dedicated account for each cultural version** of your blog.

8.11.12.7 Settings

Settings related to MetaWeblog API can be performed in **Site Manager -> Settings -> Content -> Blogs -> MetaWeblog API**. The following settings can be configured:

- **Allow automatic summary**: if enabled, blog post summary will be created automatically from the beginning of the blog post text, and will be as long as the value of the Summary length property; this is applied only when a post is created - when editing an existing post, the summary is not updated
- **Summary length**: length of automatic blog post summary in characters
- **Delete unused attachments**: if enabled, all blog post attachments which are not used in post text or have not been uploaded via WLW (i.e. have been uploaded via Kentico CMS user interface) will be deleted when an existing blog post is modified and then published from WLW



8.11.12.8 Security

Managing blog posts

The following text explains what conditions need to be met in order to perform the listed actions in Windows Live Writer. The conditions need to be valid for the user account used to access the blog (as entered in step 3 here).

To **view a blog post**, the user needs to:

- be the owner of the blog post
- or have the **Read** document permission for the blog post granted (as explained here)
- or be a global administrator

To **modify a blog post without workflow**, or to **modify a blog post under workflow without custom workflow steps**, the user needs to:

- be the owner of the blog post
- or have the **Modify** document permission for the blog post granted (as explained here)
- or be a global administrator

To **modify a blog post under workflow with custom workflow steps**, the user needs to be:

- member of the role which can approve the document in the custom workflow step (see Defining a workflow for more details)
- or a global administrator

To **delete a blog post**, the user needs to:

- be the owner of the blog post
- or have the **Delete** document permission for the blog post granted (as explained here)
- or be a global administrator

Banned IPs

If the user's IP is on the Banned IPs list, the user will not be able to transfer any data between Kentico CMS and the client application (WLW,Microsoft Word 2007+, ...).

8.11.13 Blogs internals and API

8.11.13.1 Overview

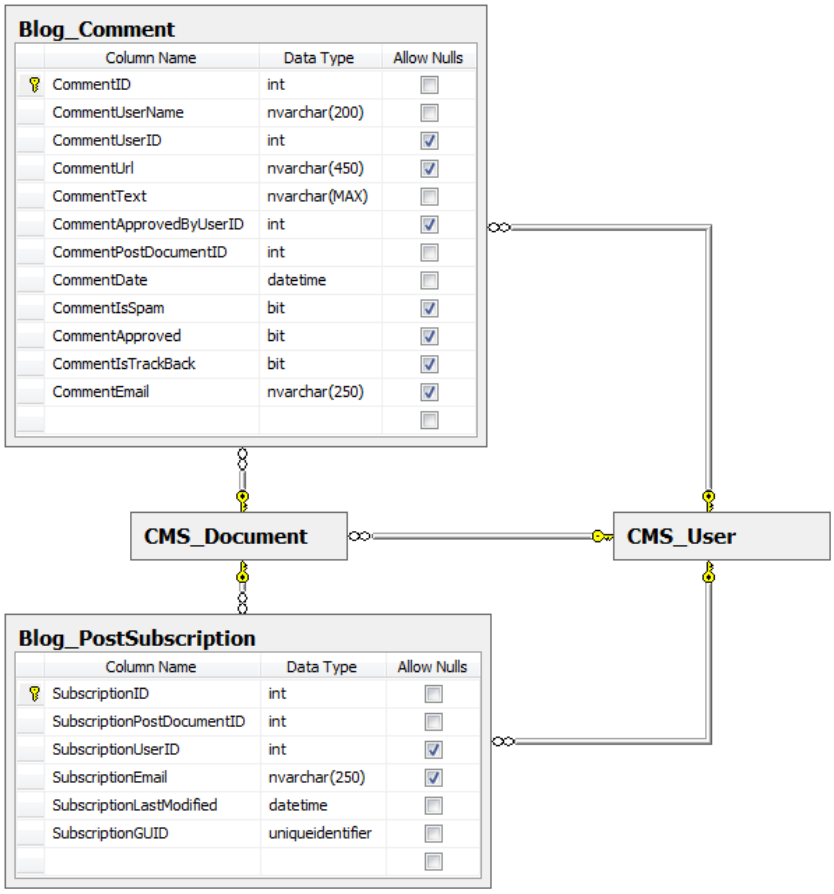
In this chapter, you will learn which database tables and API classes are used in the Blogs module. You will also see the most common API examples.

Advanced API examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all methods from the module classes, please refer to Kentico CMS API Reference.

8.11.13.2 Database tables

The following database tables are used in the Blogs module:

- **Blog_Comment** - contains records representing blog comments.
- **Blog_PostSubscription** - contains records representing blog post subscriptions.



8.11.13.3 API classes

If you are not familiar with the database table data management by Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

> **Please note**
>
> The Blogs module classes use the **CMS.Blogs** namespace.

Blog_Comment table API:

- **BlogCommentInfo** - represents one blog comment.
- **BlogCommentInfoProvider** - provides management of blog comments.

Blog_PostSubscription table API:

- **BlogPostSubscriptionInfo** - represents one blog post subscription.
- **BlogPostSubscriptionInfoProvider** - provides management of blog post subscriptions.

8.11.13.4 API examples

8.11.13.4.1 Overview

Please note

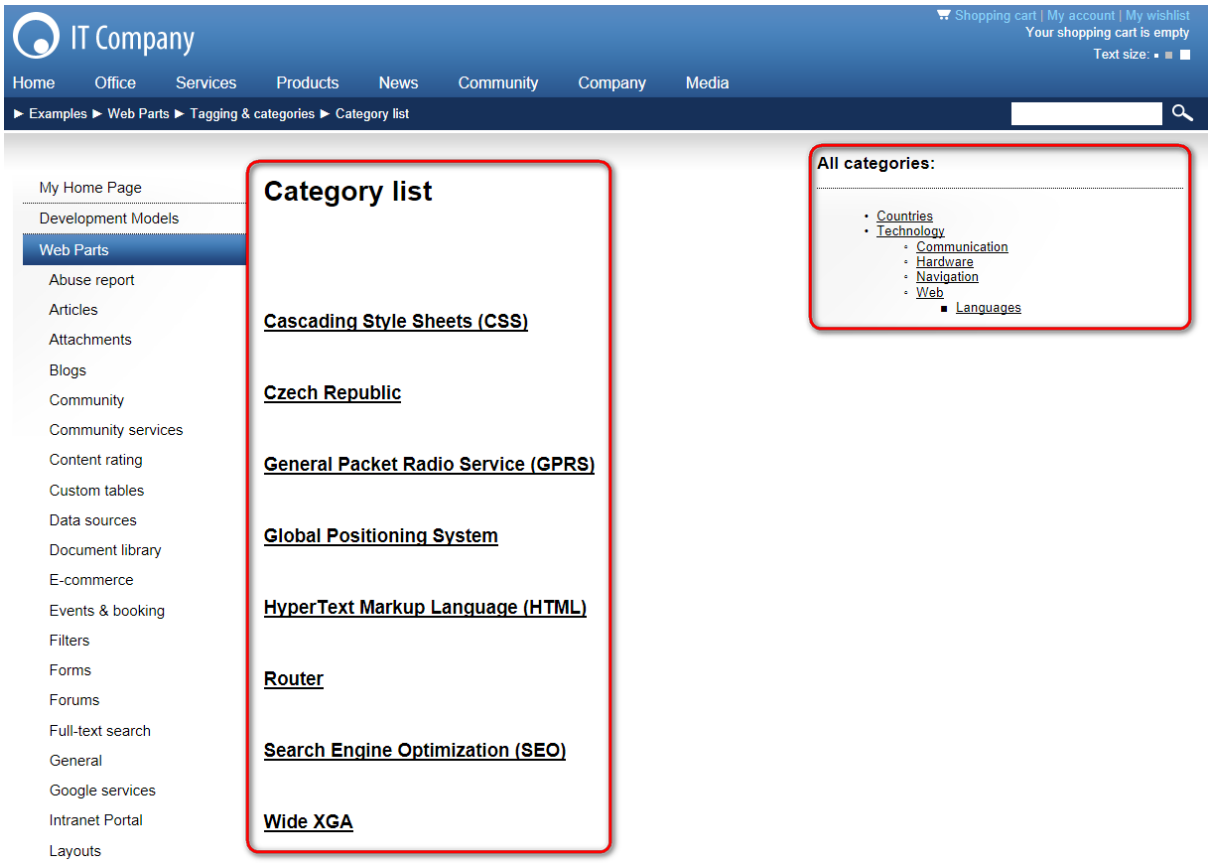
All API examples can be simulated in the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Community\Blogs\Default.aspx.cs***.

8.12 Categories

8.12.1 Overview

The Categories module enables users to sort their documents based on categories. It is thus a different approach to sorting documents from the one described in the Tags chapter.



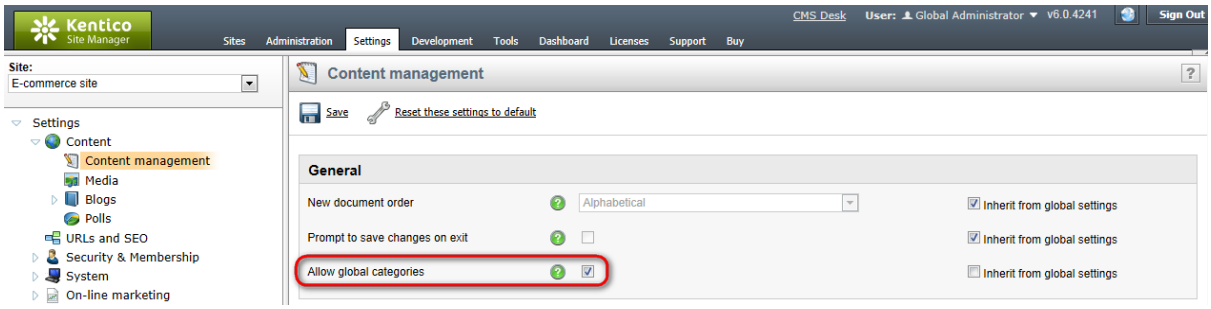
There are three types of categories from the point of view of the user:

- **Global categories** - such categories can be used across all available sites. However, you need to allow global categories for the sites first.
- **Site-specific categories** - such categories can be used only on the given site.
- **Personal categories** - global categories belonging to a particular user. The user can use these categories across all available sites.

To learn how to create and edit categories, please refer to the Managing categories topic.
To learn how to put your documents under categories, please refer to the Assigning categories to documents topic.
The security matters related to the Categories module are described in the Security topic.

8.12.2 Allowing global categories

The use of global categories can be enabled in **Site Manager -> Settings -> Content -> Content management** in the **General** section. To enable global categories, from the **Site** drop-down list choose ***(global)*** to make global settings or one of the available sites to make site-specific settings and set the **Allow global categories** option to TRUE.



8.12.3 Managing categories

Here you will learn how to manage categories and which parts of the UI serve this purpose.

Categories in Kentico CMS can be managed in the following locations:

1. Site Manager -> Administration

Because this is a global administration interface, the **Site** selector allows you to perform the categories settings globally or for a selected site. If you choose ***(global)***, you are managing global categories only. If you choose a particular site, you are managing categories of this site, optionally together with global categories (on condition that global categories are allowed for the given site).

2. CMS Desk -> Administration

As this is a site-specific administration interface, only current site-specific configuration is available. Both global and site-specific categories can be managed from here (on condition that global categories are allowed for the given site).

3. CMS Desk -> My desk

If they switch to **My profile** and navigate to the **Categories** tab, the user can manage their personal categories from this location.

4. On the live site

The user can manage their personal categories also on the live site. Specifically, they need to navigate to the **My account** page and switch to the **Categories** tab. However, this tab is displayed only if the **My account** web part is properly configured; more details on how to configure this web part can be found in E-commerce Guide -> Integration with your existing Kentico CMS website -> Web parts -> My account.

5. User properties

The administrator can manage personal categories of individual users in **Administration -> Users -> Edit (**  **)** ***user*** on the **Categories** tab.

Managing categories

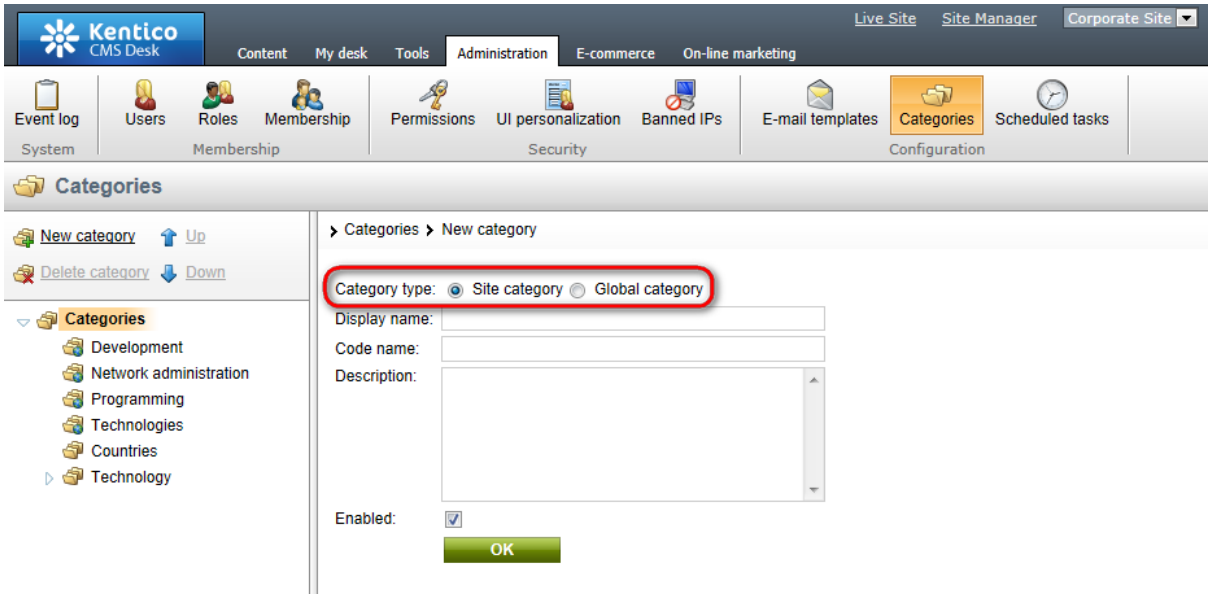
You can create a category using  **New category** and you can delete one using  **Delete category**. Categories can also be moved  **Up** and moved  **Down** in the Categories tree; however, global and

site-specific categories are sorted separately in the tree.

Creating categories

If you are creating a category from **Site Manager -> Administration** or **CMS Desk -> Administration** and global categories are allowed, the **Category type** selector may be visible on the Categories page, depending on under which category you are creating the new one. This is because both global and site-specific categories can be put under the **Categories** root and under any global category. However, under a site-specific category only categories of this type can be put.

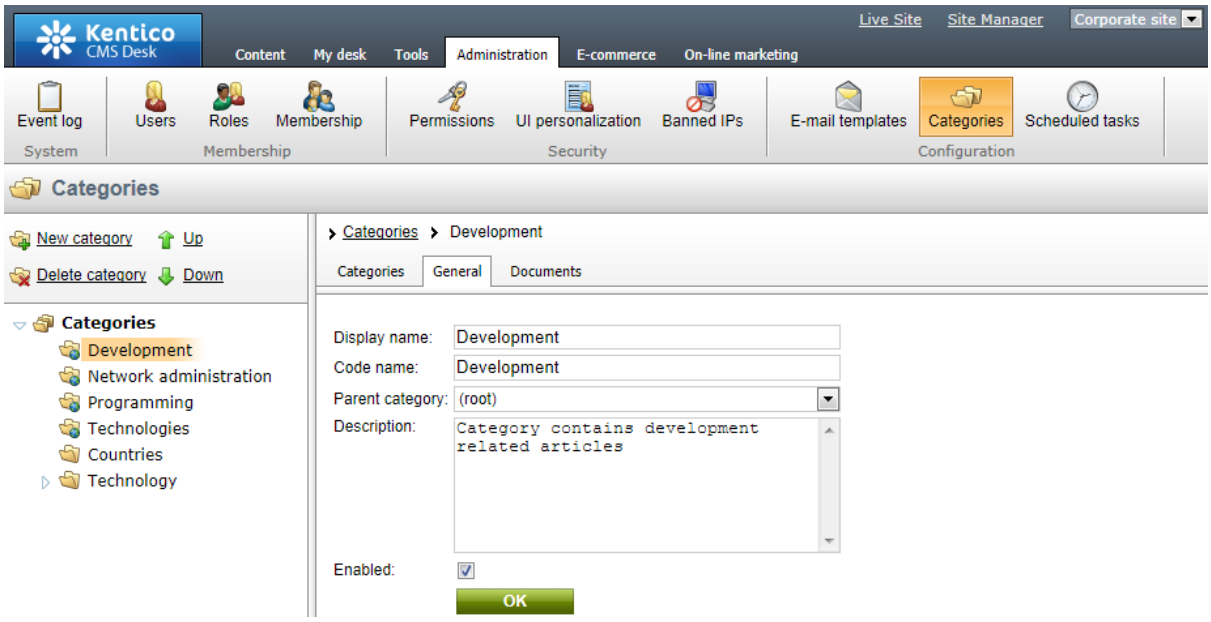
Besides, to view the **Category type** selector and thus be able to create global categories, at least one of the roles of which you are a member must have the **Modify global categories** permission assigned.



If you are creating a personal category, i.e. from **CMS Desk -> My Desk**, from user properties or from the **My account** section on the live site, the **Category type** selector is never visible because the user can use their personal categories across all available sites.

Editing categories

If you need to edit a category, select it from the **Categories** tree and configure its settings as required:



General tab

On this tab you can edit the following properties:

- **Display name** - the display name of the category.
- **Code name** - the code name of the category.
- **Parent category** - the parent category under which this child category is put. Use the drop-down list to move the category under a different category.
- **Description** - the description text of the category.
- **Enabled** - indicates if the category is enabled in the system. Documents cannot be assigned to categories that are not enabled.

Localization expressions, i.e. **Localize other languages** (), **Remove localization** () and **Localize** (), are described in detail in the Localization expressions topic in the Development -> Multilingual and international support section.

Documents tab

On this tab you can view a list of documents which fall under the given category, together with some important document details, e.g. document name, document type, document modification stamp etc.

Categories tab

On this tab you can **Edit ()** and **Delete ()** categories and you can also view categories details, i.e. their child categories, categories status and the number of assigned documents.

8.12.4 Assigning categories to documents

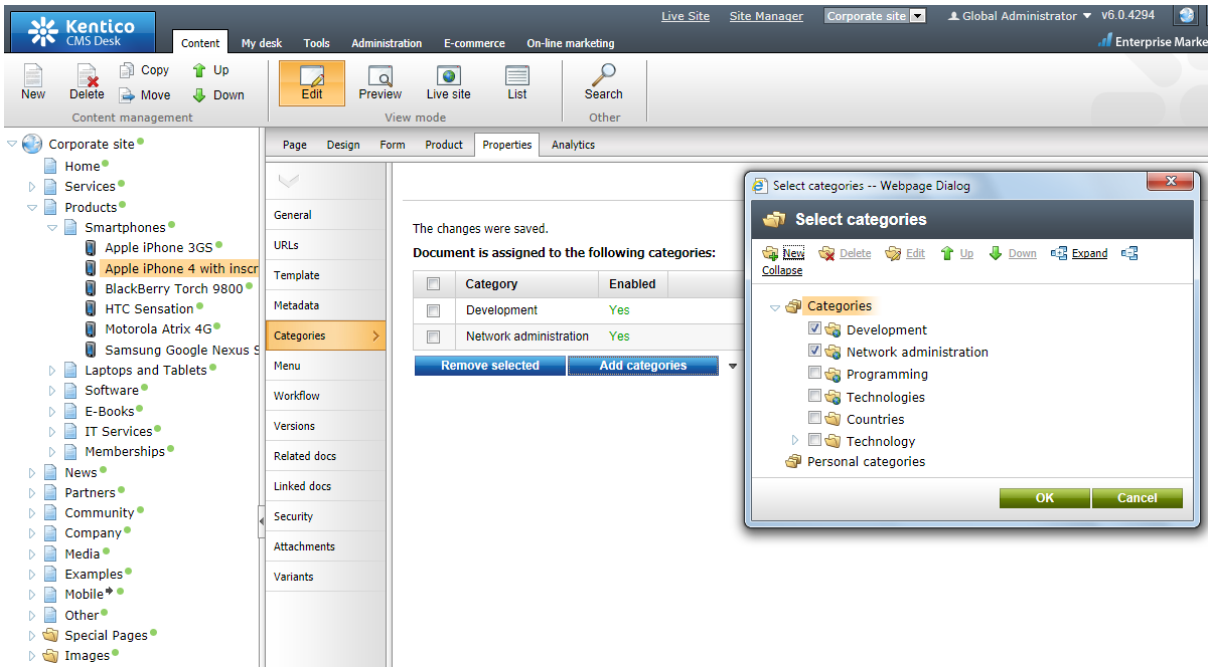
Here you will learn how to assign categories to a document.

1. To add categories to a document, you need to navigate to **CMS Desk -> Content**,

2. select a document from the **Content** tree

3. and on its **Properties** tab, switch to **Categories**. If any categories are already assigned to the document, a list of these categories is displayed, together with the **Remove selected** and **Add categories** buttons. If no categories are assigned, only the latter button is displayed.

4. Now you need to click the **Add categories** button and in a dialog window which pops up select the categories from the **Categories** tree. Only categories allowed for the current site together with personal categories of the user are displayed. Please note that the dialog window allows you not only to select categories but also perform standard categories editing tasks, as described in the Managing categories topic.



5.To save the changes, click **OK**.

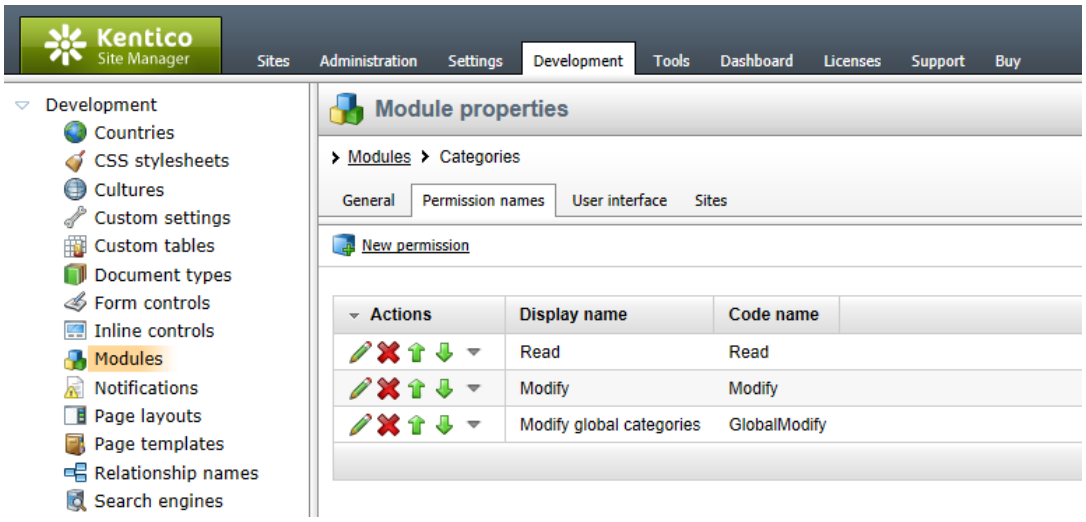
Please note

The **Multiple categories selector** can be used also with a document type. This gives you the option to assign categories to the document in **CMS Desk -> Content** on the **Form** tab or on the live site using User contributions.

8.12.5 Security

To prevent users from accessing and modifying categories, you will need to assign Categories module permissions. This procedure is described in more detail in the Membership, permissions and security -> Permissions chapter in the Development section of this guide.

The Categories module has the following permissions:



- **Read** - allows to read categories.
- **Modify** - allows to create and modify categories.
- **Modify global categories** - allows to create and modify global categories.

Example

To allow members of a particular role to read site-specific categories, you will need to assign this role the **Read** permission.

To allow members of a particular role to create and modify site-specific categories, you will need to assign this role the following permissions:

- **Read**
- **Modify**

To allow members of a particular role to read global categories, you will need to assign this role the **Read** permission:

To allow members of a particular role to create and modify global categories, you will need to assign this role the following permissions:

- **Read**
- **Modify global categories**

This can be configured in the Categories module permissions matrix in **Site Manager -> Administration -> Permissions**, as described in detail in the Permissions chapter in the Development -> Membership, permissions and security section of the Developer's Guide.

8.12.6 Categories internals and API

8.12.6.1 Overview

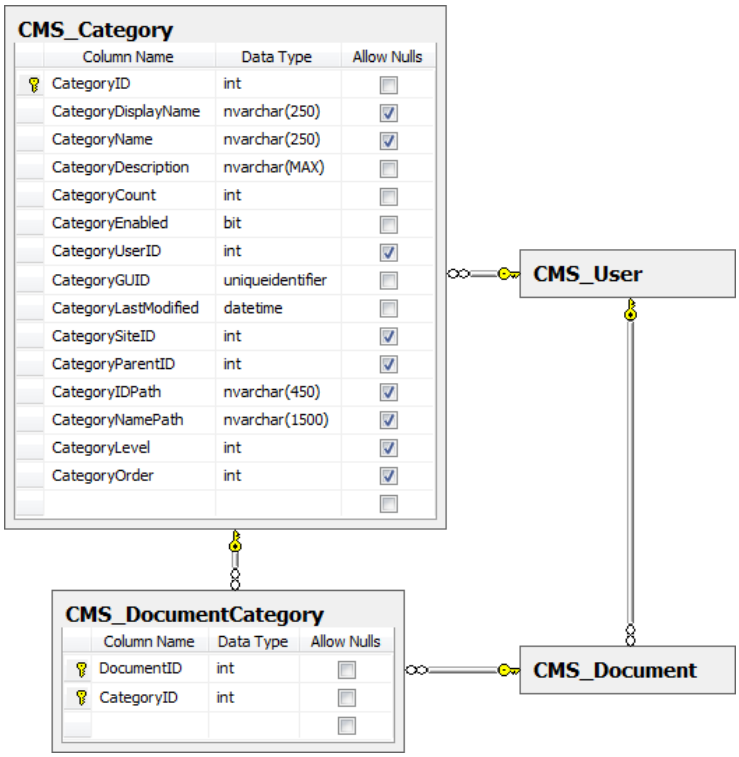
In this chapter, you will learn which database tables and API classes are used in the Categories module. You will also see the most common API examples.

Advanced API examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all methods from the module classes, please refer to Kentico CMS API Reference.

8.12.6.2 Database tables

The following database tables are used in the Categories module:

- **CMS_Category** - contains records representing categories.
- **CMS_DocumentCategory** - contains relationships between documents and categories indicating that particular documents are added to a particular category.



8.12.6.3 API classes

If you are not familiar with the database table data management by Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

The Categories module classes use the **CMS.SiteProvider** namespace.

Category table API:

- **CategoryInfo** - represents one category.
- **CategoryInfoProvider** - provides management of categories.

DocumentCategory table API:

- **DocumentCategoryInfo** - represents relationships between one document and one category expressing that the particular document is added to the particular category.
- **DocumentCategoryInfoProvider** - provides management of relationships between documents and categories.

8.12.6.4 API examples

8.12.6.4.1 Overview

Please note

All API examples can be simulated in the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Documents\Categories\Default.aspx.cs***.

8.13 Chat

8.13.1 Quick start

This topic will show you how to enable your website visitors to chat with each other in a few steps.

This step-by-step guide consists of two parts - first, you will create chat rooms, then you will place web parts on a page to enable the chat on the live site.

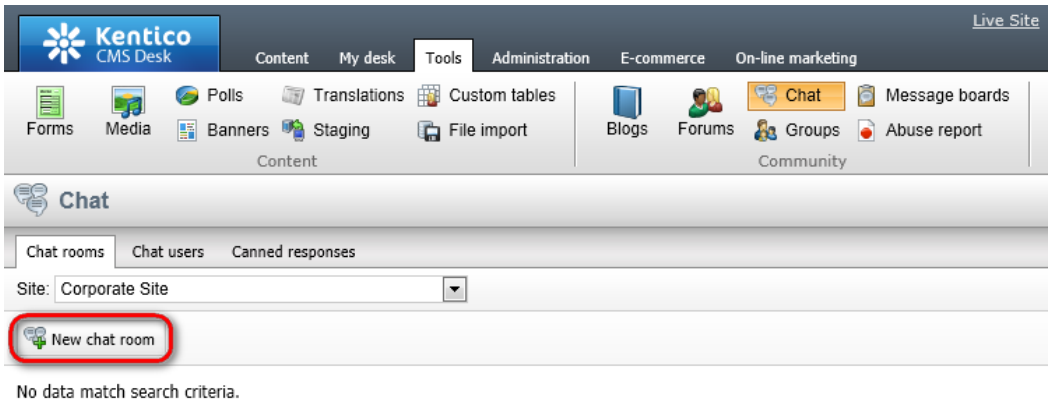
For basic information about the Chat module, refer to the Basics topic.

Creating chat rooms

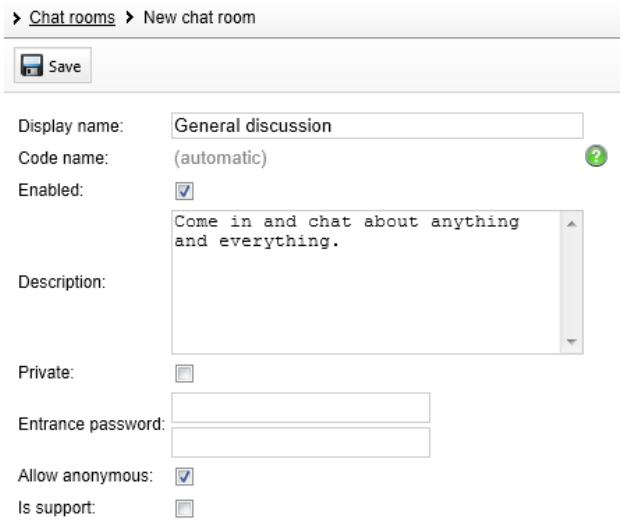
First, you need to create rooms that users will chat in. Follow these steps to create a chat room:

1. Log in to **CMS Desk** and navigate to **Tools -> Chat**.

2. Click **New chat room**.



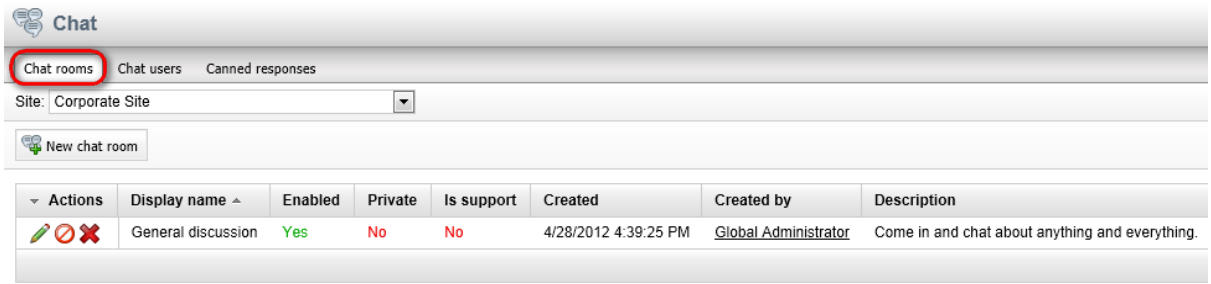
3. Fill in the **Display name** and **Description**.



The other settings determine the type of the room. You can leave them unchanged. To learn more, refer to the Chat room types and Configuring rooms topics.

4. Click **Save** to create the room.

5. Click **Chat rooms** to be redirected to the list of existing rooms, which now contains the room you just created.



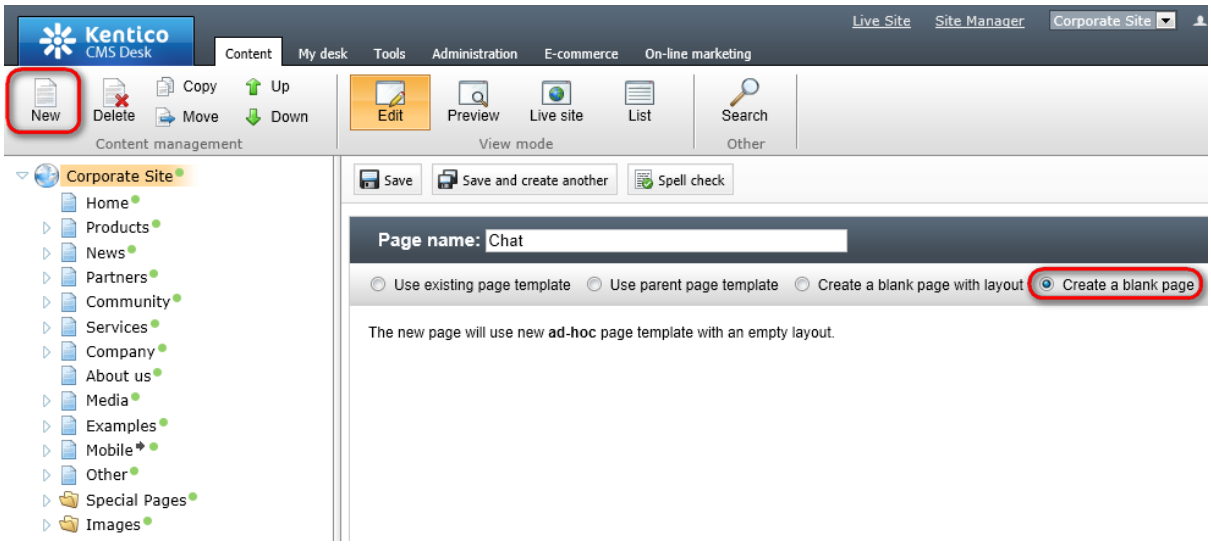
You have created a room, which users can join and chat with others. Repeat the procedure any number of times to create other rooms.

To learn more about chat room management, refer to the Managing chat rooms chapter.

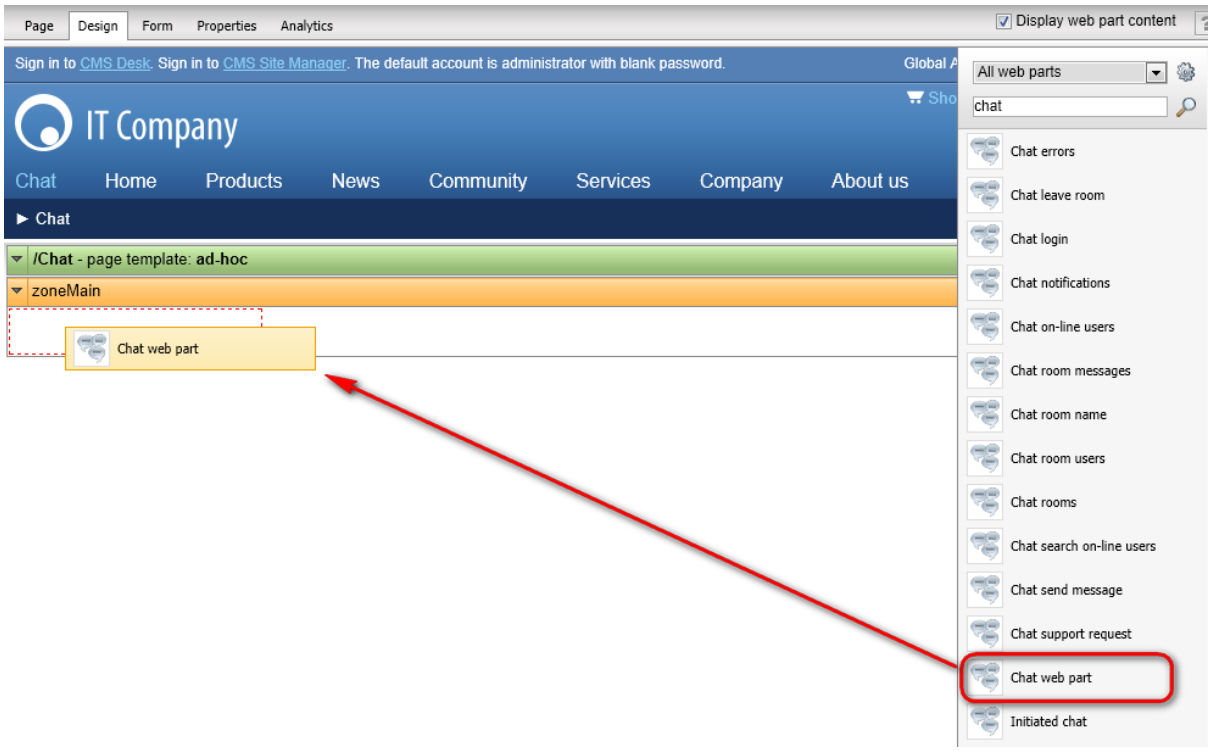
Setting up chat on site

1. In **CMS Desk**, switch to **Content**.

2. Create a new document of type **Page (menu item)**. Choose to **Create a blank page**.



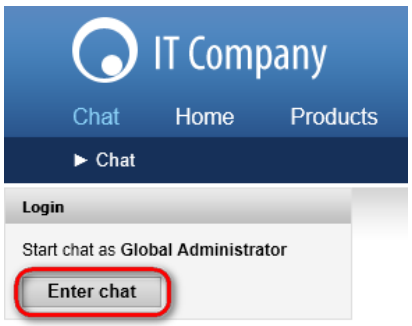
3. Search for "Chat" in the web part toolbar and drag the **Chat web part** into the **zoneMain** web part zone.



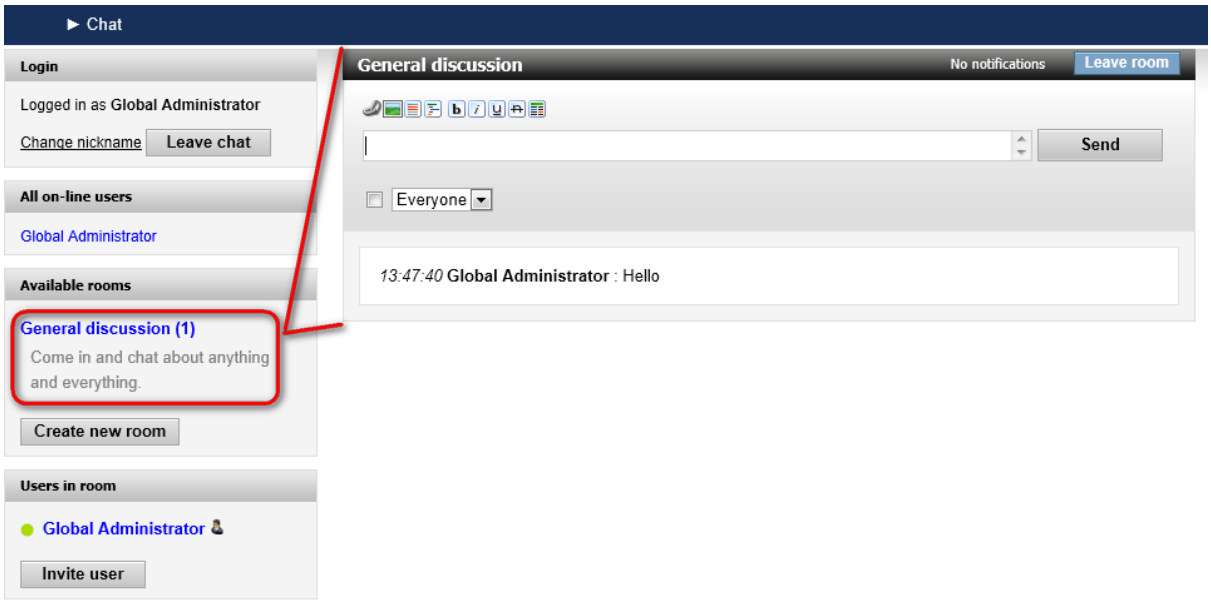
4. In the **Web part properties** dialog, leave all the settings at their default values and click **OK**. Chat should now be correctly configured.

5. Switch to the live site and navigate to the **Chat** page.

6. Click **Enter chat**.



You can now click a room in the list of **Available rooms** and start chatting with other users in that room.



To learn how to use the chat, refer to the Using chat chapter.

For information on how to manage users of the chat, refer to the Managing chat users chapter.

If you wish to learn about other ways of setting up the chat or customize the looks and behavior of the chat, refer to the Setting up chat on site chapter.

8.13.2 Overview

8.13.2.1 Basics

The Chat module allows visitors of a website to communicate with each other in real time by means of text messages. Chat users can gather in chat rooms of various types, as well as talk one-on-one without interruption of other users. Also included is an option to set up live customer support chat featuring operator notifications in CMS Desk and predefined (canned) responses. Other features include graphic smileys, BBCode tags and integration with the Bad words module.

The module comes as a set of web parts that can be either placed on a single page, allowing all the features to be accessed at one place, or spread across multiple pages to separate individual components (e.g. the log-on page, chat room list, etc.).

Administrators and site editors can manage the module through its user interface located in **CMS Desk -> Tools -> Chat**.

Chat rooms

Chat rooms connect people who share a common interest or background. When a user joins a chat room, they can send text messages which all other users in the room will be able to read and respond to. Kentico CMS offers a set of chat room types, including private and public rooms or password protected rooms.

To view the complete list of chat room types along with their detailed descriptions, refer to the Chat room

types topic.

Chat users

By default, any visitor of your website can enter the chat, regardless of whether they have a user account set up on the site. Every chat user only needs to choose a nickname which will represent them among the chatting community. The name with which the user logs in to the site will not be shown to other chat users.

For information about how to administer chat users, see the Managing chat users chapter.

Support chat

Chat in Kentico CMS contains a special type of chat rooms aimed at providing on-line help and support to customers. These support rooms have one or more registered users acting as consultants, while site visitors can join and ask questions. The consultant is notified about new messages in CMS Desk and is able to open a message dialog through there.

More information can be found in the Support chat chapter.

8.13.2.2 Prerequisites

To use the Chat module, you need to have **Windows Communication Foundation** (WCF) installed and properly configured. Also, there are some licensing limitations to consider.

Windows Communication Foundation

For description of the whole process of installing and configuring WCF, visit Installation and deployment -> Additional configuration tasks -> Configuring Windows Communication Foundation.

Licensing

The Chat module is included in the **Base license** with **Social Networking package**, **Ultimate License** and **Kentico EMS**.

8.13.2.3 Technical overview

The majority of chat actions (such as sending messages, joining a room, etc.) on the live site are performed using AJAX without reloading the page. HTML content is generated almost exclusively by JavaScript on the client side. The server provides the client with data in JSON format.

Windows Communication Foundation

On the server side operate two WCF services - **ChatService** and **ChatSupportService**. The client side communicates with those services using HTTP requests and receives data in the JSON format.

JavaScript code for communication with the web services is generated automatically by WCF. You can check if it is generated correctly by visiting the following URL:

```
<your website URL>/CMSModules/Chat/CMSPages/ChatService.svc/JS
```

If there isn't any JavaScript generated on the page, WCF may not be correctly configured. See the Prerequisites topic for configuration information and links to the official troubleshooting page.

jQuery Templates

The client side receives objects with data and generates HTML. To do so, it uses a jQuery plug-in called **Templates**. Using the templating system on the client side allows chat to work without requesting pages and still offer flexibility similar to Kentico macro language (K#). jQuery Templates documentation and syntax description can be found at https://github.com/BorisMoore/jquery-tmpl.

Templates are saved as transformations with their type set to jQuery. With the module comes a container document type called **Chat – Transformations**, which holds transformations needed for chat to work. JavaScript objects are passed to the jQuery Template engine along with the transformation text.

To learn more about the templates and how to use them to customize the chat module, refer to the Writing transformations for chat topic.

BBCode parser

The client side also provides BBCode resolving - a JavaScript resolver is used to parse the BBCode tags.

8.13.3 Managing chat rooms

8.13.3.1 Chat room types

This topic describes the various types of chat rooms available in the Kentico CMS Chat module. The types are cumulative, meaning that a chat room can be of more than one type, while inheriting the properties of all the selected types.

Types can be selected when creating a new room or editing an existing one. You can find more information about the editing interface in the Configuring chat rooms topic.

Global and site-related rooms

Similarly to most objects throughout the CMS, a chat room can be made available on all websites in a particular instance of the application, or its usage can be limited to a specific site. This property needs to be specified before you create a chat room, by selecting **(global)** or a desired site, in the **Site** drop-down list.

Public and private rooms

A public chat room can be seen in the list of rooms and joined by anyone. Contrary to that, joining a private room requires an invitation from a user who is an administrator of that particular room. This parameter can be turned on or off anytime on the chat room editing page, by checking or un-checking the **Is private** check-box.

Anonymous access

Governed by the **Allow anonymous** check-box on the chat room editing page, this property determines whether an anonymous user can join the room. A chat user is considered anonymous when she is not logged in to the site via a user name and password.

One-on-one chat

This type of room is automatically created when a user requests chat with another user by clicking her name in the list of on-line users.

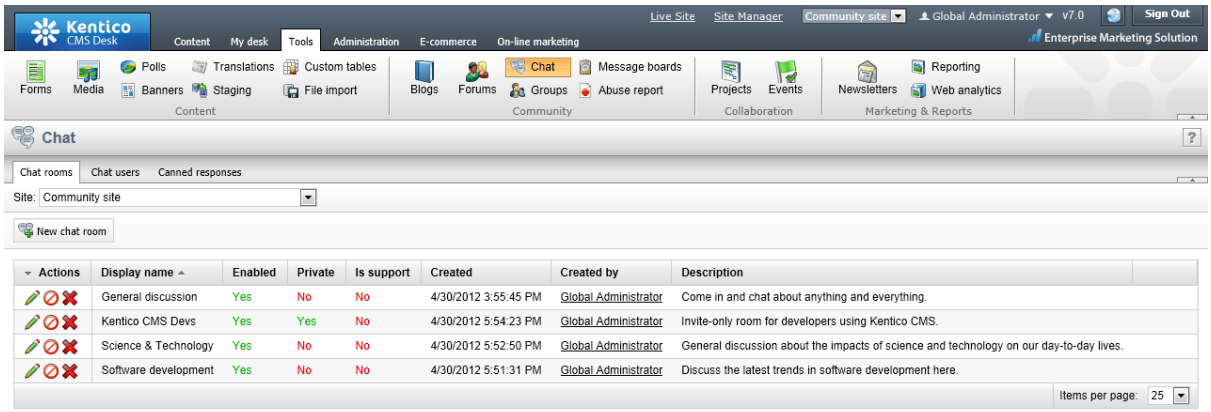
Support chat

Support chat is a special type of chat room where customer support personnel can talk to site visitors. Requests for support can be accessed through CMS Desk, provided you have the appropriate permission. You can create a room of this type by checking the **Is support** check-box on the room editing page. For more information about this feature, refer to the Support chat chapter.

8.13.3.2 Configuring rooms

In this topic, you will learn how to create chat rooms, enable or disable, and reconfigure existing rooms.

Chat rooms can be managed in **CMS Desk -> Tools -> Chat**. On the **Chat rooms** tab, you can see the list of existing rooms.



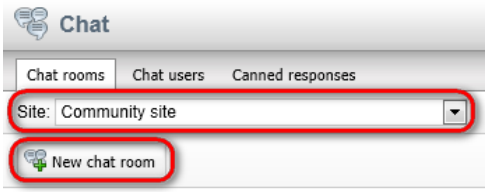
Please note

The page lists all types of rooms, including rooms that accommodate one-on-one chat and support rooms.

Creating a new room

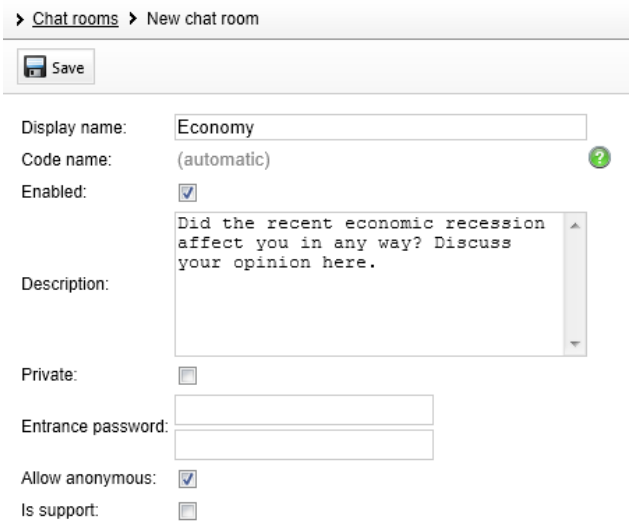
To create a new room, select whether you want it to be global (i.e., available on all sites) or bound to a

specific site, then click **New chat room**.



The **New chat room** form offers a set of properties that the new room can have:

- **Display name** - the name that will be displayed in the user interface and in the list of rooms on the live site.
- **Code name** - the name that will be used in code.
- **Description** - text which will be displayed alongside the display name of the room in both the user interface and the live site.
- **Private** - property that indicates whether users will need an invitation in order to join the room.
- **Entrance password** - string of characters which the users will be prompted to enter when trying to join the room.
- **Allow anonymous** - check-box that determines whether users which are not logged in to the site will have access to the room.
- **Is support** - setting that allows to receive notifications about new messages and chat in the room through CMS Desk, thus enabling customer support from a single location. You can find more information in the Support chat chapter.



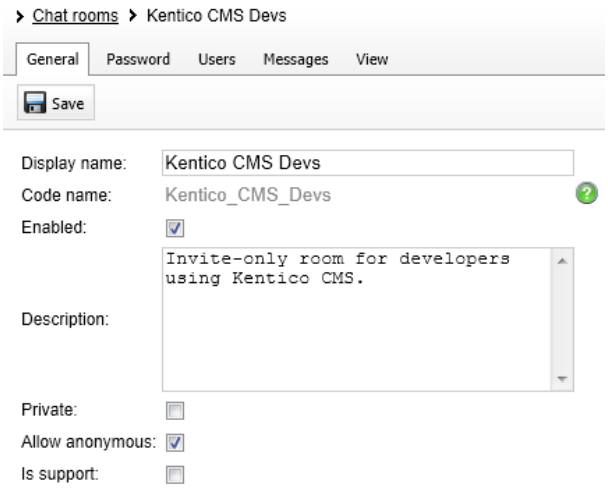
To create the room, click **Save**.

Room actions

The list of chat rooms provides buttons for room management in the **Actions** column.

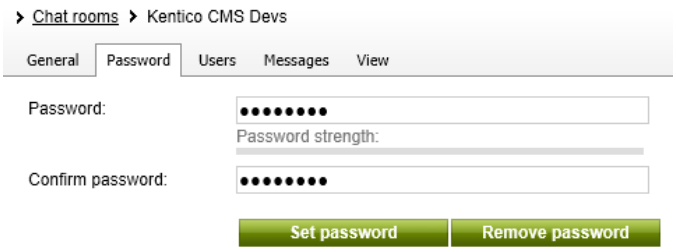
- **Edit** - opens the room editing page, where you can adjust room properties and manage users that are joined in the room, as well as messages posted in the room.
- **Disable -** renders a room unavailable to the live site users and disallows them to send any messages to the room.
- **Enable** - re-enables a disabled room.
- **Delete** - Schedules a room for deletion. Rooms are not deleted immediately, but rather disabled and flagged as scheduled to deletion. A scheduled task will delete the room after a certain amount of time. Nevertheless, the room won't be accessible by any user.

The **Edit** button takes you to the room editing form.



On the **General** tab, you can modify the room's properties, which are described in the preceding section of this topic.

The **Password** tab allows you to set, change or clear the room's password.



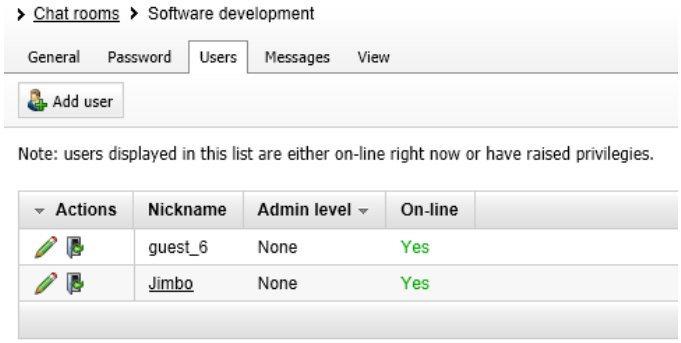
The **Users** tab lists users who are present in the room. To learn about management of users in a particular room, see the Users tab topic.

To learn more about the **Messages** and **View** tabs, see the Messages & View tabs topic.

8.13.3.3 Users tab

The **Users** tab of the chat room editing interface allows management of users who are present in the respective room. It allows you to kick users, add users to the room and assign them administrator rights.

The tab features a list of users that are currently present in the chat room and users that have elevated rights for the room. Off-line users with no special permissions do not appear on the list.

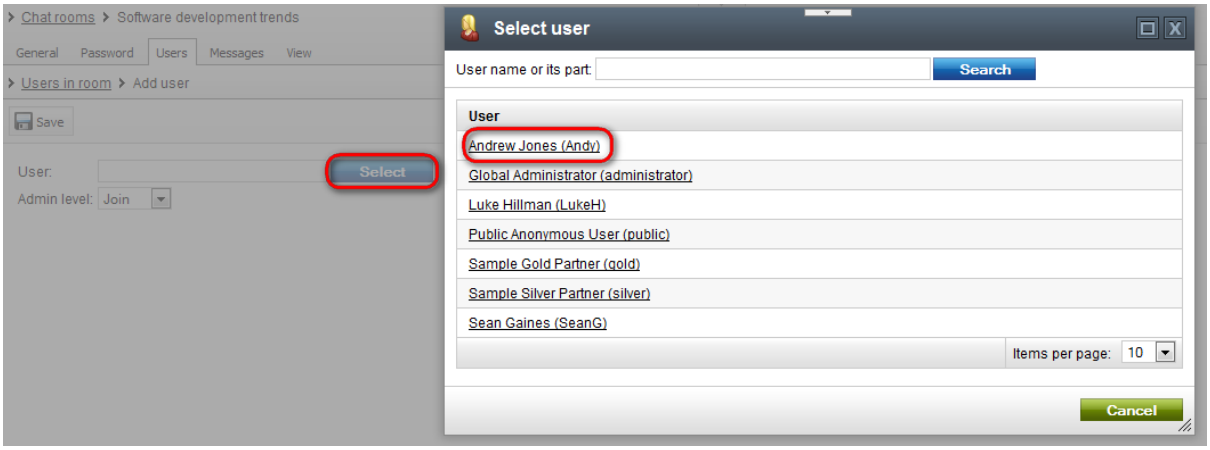


Adding a user

To add a user to the room, click **Add user**. Then click **Select** to bring up a dialog listing users registered on the current site. You can select the user you wish to add by clicking the appropriate line.

Before adding the user to the room, you need to assign them certain permissions. For this purpose, the **Admin level** drop-down list offers the following options:

- **None** - the user won't have any special permission.
- **Join** - the user will receive permission to join the room, but no special management rights. This option is available only in private rooms.
- **Admin** - the user will receive rights to administer the room. A room administrator can adjust the properties of the room, moderate messages and manage users present in that room.



User actions

The following action buttons are available in the list of room users.

- **Edit** - allows to modify the user's permissions and properties.
- **Kick** - removes the user from the room.
- **Revoke access** - available only in private rooms, removes the user's right to join the room.

The **Edit** button takes you to the **Editing admin rights** page. On this page, you can change the permissions the user has.

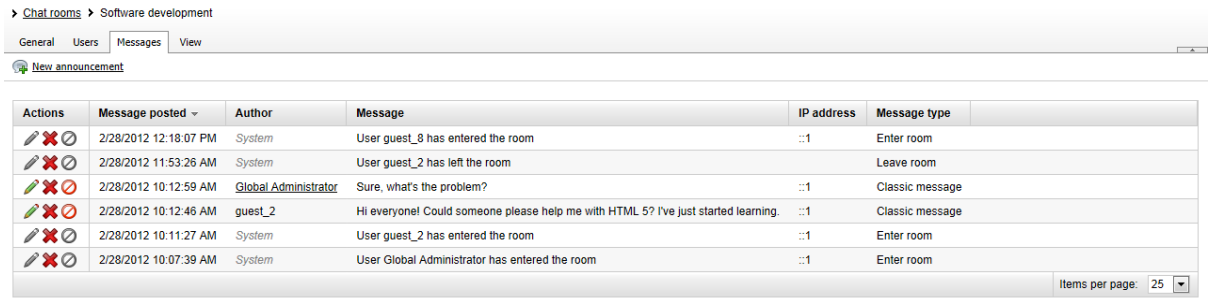
You can click the user's name to display a dialog that allows you to change the user's chat nickname. In that dialog, you can click **Edit** to modify the user's settings, similarly to the interface located in **Administration -> Users**.

8.13.3.4 Messages & View tabs

The **Messages** tab of the room editing page allows you to manage messages sent in the edited room, while the **View** tab displays the room as live site users would see it.

Messages tab

On the **Messages** tab, you can view and manipulate messages that were sent in the chat room. You can also send announcements to the room. The announcements will appear in the same formatting as system messages.



You can sort and filter messages by their type. The following list presents types of messages authored by chat users:

- **Classic message** - messages posted by users on the live site.
- **Whisper** - private messages posted to the room. On the live site, only the user selected as recipient can see the message.
- **Announcement** - messages sent from CMS Desk.

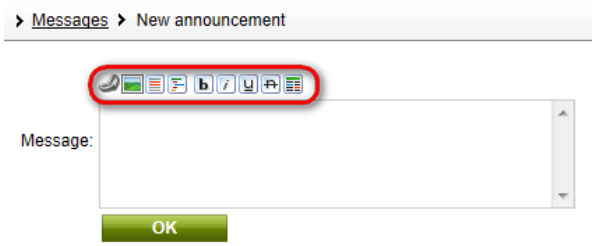
In addition, there are several types of messages that are automatically generated by the system:

- **Enter room** - appears when a user joins a room.
- **Leave room** - appears when a user leaves a room.
- **Change nickname** - appears when a user changes her nickname.
- **Kick** - informs that a user has been kicked from a room.
- **Invitation** - informs that a user has invited another user into a room.

- **Support greeting** - displayed only in support chat, appears to visitors who request support for the first time in a session.
- **Permanent leave room** - displayed when a user leaves a private room and cancels her invitation.
- **Permanent kick** - displays when a user is revoked access to a room.

Adding an announcement

To add a new announcement, click **New announcement**. Enter the message text into the text area and click **OK** to submit the message. The text of the message can contain BB code, which you can insert via the image buttons above the text area.



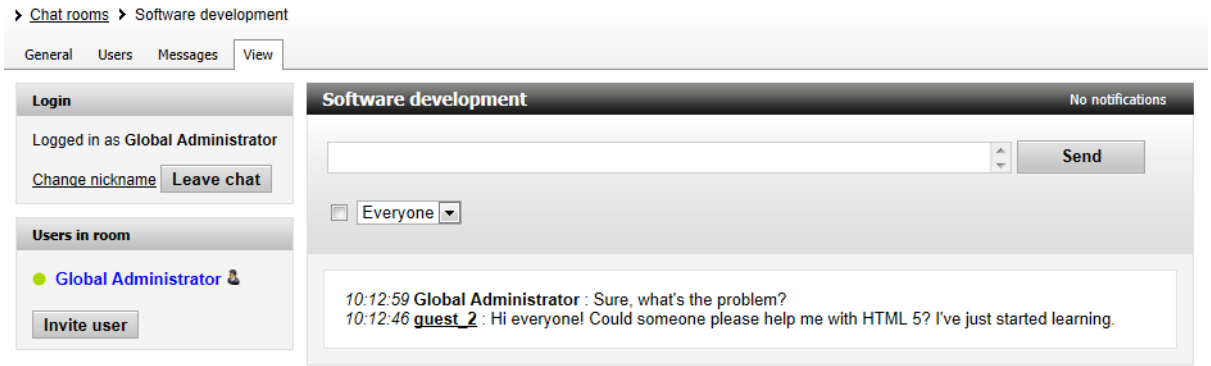
Message actions

The list of messages provides the following actions to be performed on messages:

- **Edit** - takes you to the message editing page, which is identical to the **New message** form. There, you can modify the text of the message.
- **Delete** - permanently deletes the message.
- **Reject** - hides the message from live site users without deleting it.
- **Approve** - takes back the Reject action and makes the message available for all users in the room to see.

View tab

The **View** tab lets you see the chat room as it would appear to you on the live site, including the list of on-line users, messages and notifications.



8.13.4 Managing chat users

8.13.4.1 Overview

This topic provides insight into the system of organizing users in the Chat module. It describes how chat users are stored in the database, the difference between anonymous and registered users and the two types of on-line state.

Chat users are an entity separate from CMS users. They are stored in a dedicated database table as a combination of their nickname and a foreign key binding them to CMS users. The binding, however, is optional - absence of the binding makes the user anonymous.

User types

There are two types of chat users - anonymous and registered. Anonymous users are site visitors, who either don't posses a user account on a site, or they do have an account but are not logged in. To enter the chat, users only need to choose a nickname. Registration is not necessary.

If a registered user is logged in, the nickname they choose when entering the chat is tied to their account. As a result, the nickname is reserved for that user.

On-line users

The on-line state of chat users is monitored on two levels - site level and room level. A user who enters the chat starts pinging the server on a regular basis, thus keeping the server aware that they are on-line. When the user goes off-line (i. e. stops pinging the server), their on-line state is changed, rather than the user record being deleted altogether.

When the user enters a room, the client side of the application includes the information in the packets sent to the server, thus notifying the server that they are on-line in a certain chat room. When a user with elevated permissions leaves the room, they are not removed from the room, only their on-line state changes. Users without permissions are removed from rooms immediately after leaving them or leaving the chat. More information about room permissions can be found in the Permissions topic.

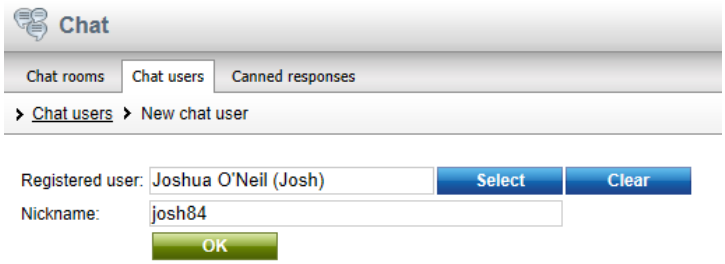
8.13.4.2 Creating and editing users

Chat users can be managed in **CMS Desk -> Tools > Chat -> Users**. The page lists registered users who entered chat on the current site.

Adding a new user

To add a new user to the chat, click **New chat user**. Select a registered user, enter a nickname for

the user and click **OK**.



The system doesn't allow adding anonymous users, since real users wouldn't then be able to choose nicknames added this way.

Editing users

You can click **Edit** () in the Actions column to bring up the user editing page. On the page, you can change the user's nickname. Additionally, you can click **Edit**, which opens a new window allowing to adjust the user's properties.

8.13.4.3 Permissions

The Chat module offers a set of permissions intended for allowing or denying various actions to certain users.

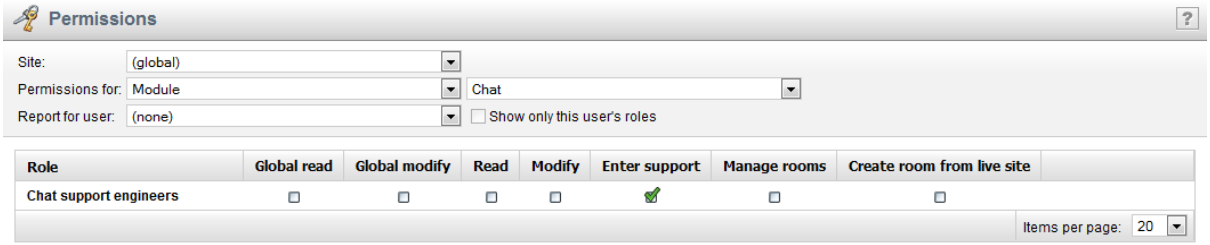
Module permissions

The Chat module offers standard module permissions adjustable in **Site Manager -> Administration -> Permissions**. The following permissions control allowed actions in CMS Desk:

- **Global read** - allows users in the given role to display the lists of global chat rooms, chat users and canned responses, but doesn't permit any modifications. The permission is limited to global objects, it doesn't apply to objects bound to a specific site.
- **Global modify** - gives full control of global chat rooms, chat users and canned responses. The permission is limited to global objects, it doesn't apply to objects bound to a specific site.
- **Read** - allows users in the given role to display the lists of site-bound chat rooms, users and canned responses, but doesn't permit any modifications. The permission doesn't apply to global objects.
- **Modify** - gives full control of site-bound chat rooms, users and canned responses. The permission doesn't apply to global objects.
- **Enter support** - displays an additional toolbar with the options to go on-line as a support person and manage canned responses. More information can be found in the Support chat chapter.

The following permissions are applicable to users using chat on the live site:

- **Manage rooms** - gives users of the given role the right to modify chat room properties (e.g., create, delete, set passwords, send invitations, etc.) and administer users (e.g., define room administrators or kick users).
- **Create room from live site** - allows users to create a new chat room using a button in the list of chat rooms.



Room permissions

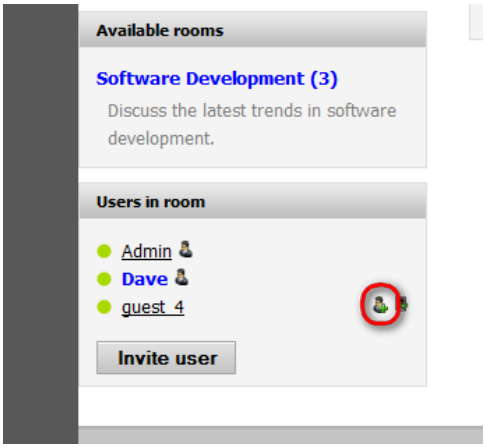
There are also a few permissions which are specific for the Chat module. These can be assigned to a user for a particular room. This means that a single user can have elevated rights in one room, while being a regular user without any special permissions in another.

The following list presents users with different permissions and the actions they are allowed to perform:

- **Regular user** - doesn't have any special rights. Such user can only post messages to public rooms.
- **Invited user** - can post messages to a private room they are invited to.
- **Room administrator** - can post messages, reject messages, kick users and define other administrators. An administrator can also edit the room's properties or disable the room to make it unavailable to be joined.
- **Room creator** - has the same rights as an administrator, however, this permission can be taken away from the user only in the CMS Desk interface by a user with the **Modify** permission for the Chat module.

Room permissions can be set while adding or editing a room user in **CMS Desk -> Tools -> Chat -> ...select a room... -> Users**. See the Users tab topic in the **Managing chat rooms** chapter for details.

Users can be assigned rights also on the live site, using the **Add room admin** button in the list of room users.



You can remove administrator rights from a user in a similar fashion - by clicking the **Delete room admin** button.

Invitations to users can be sent via the **Invite user** button in the list of room users.

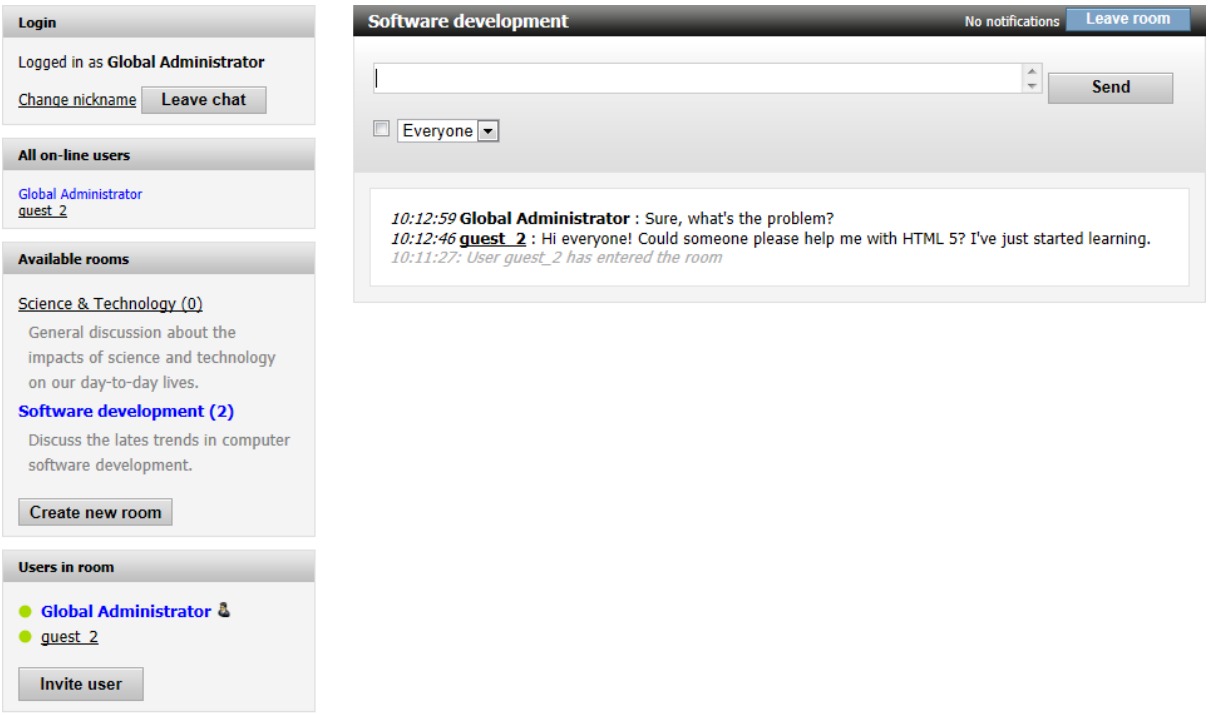
8.13.5 Setting up chat on site

8.13.5.1 Available web parts

This topic lists all web parts that accommodate the different functions of the Chat module on the live site.

Chat web part

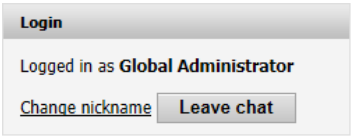
The Chat web part combines all the web parts described in the following text into one. It provides a ready-to-use solution to be placed on a page without additional configuration.



Other web parts in this topic can be placed either all together on a single page, or separately on multiple pages while the web parts ensure redirection between the pages.

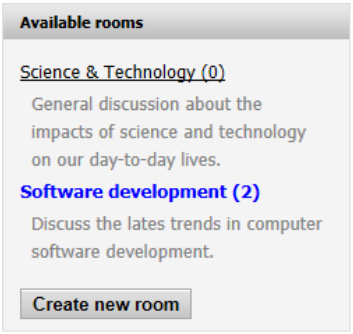
Chat login

Enables users to log into chat. When logged in, displays the login status, a link to change nickname and the Leave button.



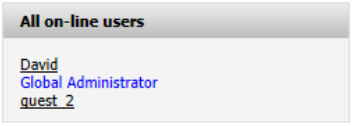
Chat rooms

Displays a list of available rooms and a button to create a new one if the current user is allowed to do so.



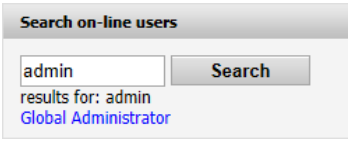
Chat on-line users

Provides a list of users currently on-line in the chat. Additionally, it allows to initiate one-on-one chat with an on-line user.



Chat search on-line users

Allows to find a particular user among the users currently on-line.

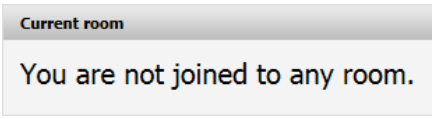


Chat room users

Displays the list of users who joined a particular chat room along with their on-line state.

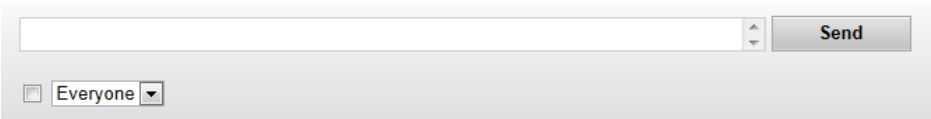
Chat room name

Displays the name of the room the user is currently in.



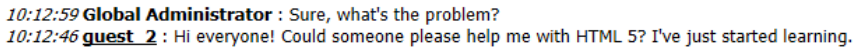
Chat send message

Allows users to send a message to the current chat room.



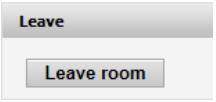
Chat room messages

Displays the messages posted to the current chat room.



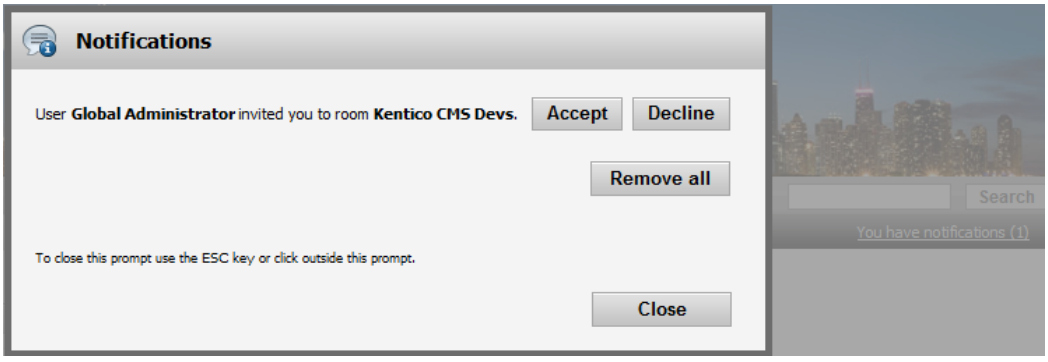
Chat leave room

Displays a button to leave the current room.



Chat notification

Displays notifications about various events, such as receiving an invitation to a room.



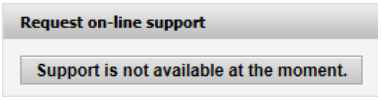
Chat errors

Displays warnings and errors that may occur while chatting. If the web part is not present, all errors are displayed as JavaScript alerts.

Support chat web parts

Chat support request

Displays a button that users can use to request chat with a support person.

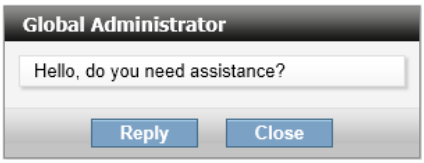


Initiated chat

Allows support personnel to start chatting with the visitor who is viewing the page with this web part.

Automatically initiated chat

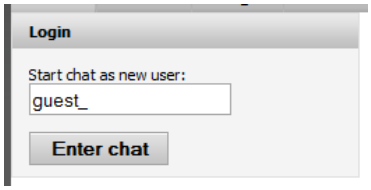
Allows to define a message, with which it will automatically address the visitor who is viewing the page with this web part.



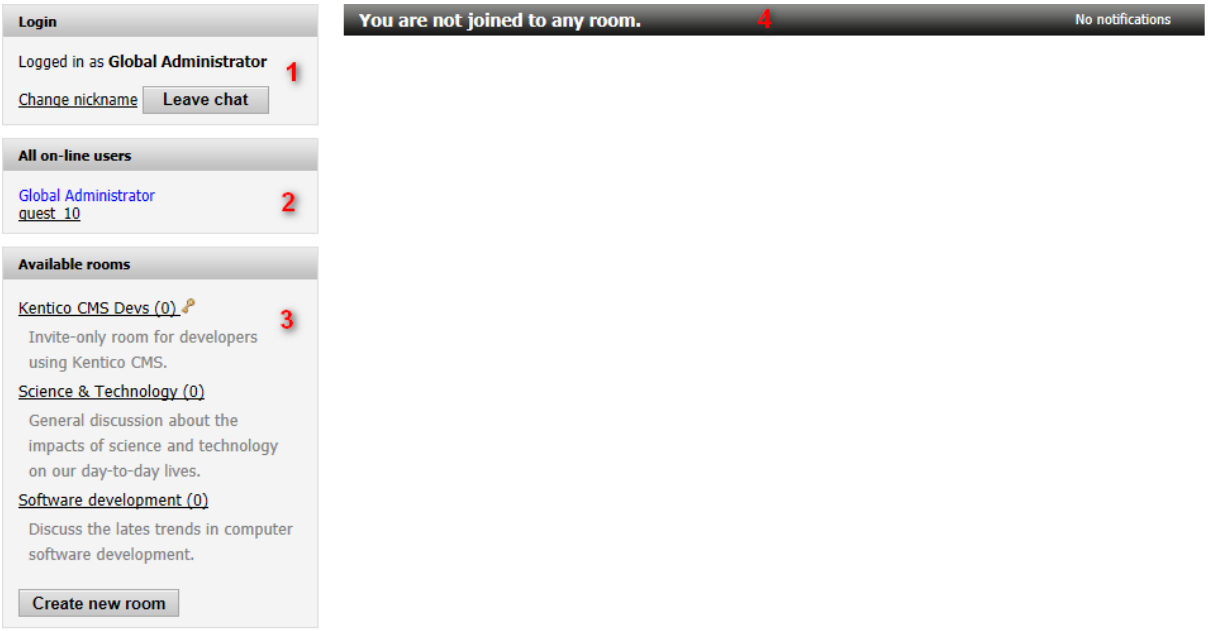
8.13.5.2 Layout overview

This topic describes the components and the default layout of the chat web part.

The Chat web part groups all the available components into a single ready-to-use web part. When you access a page with the web part for the first time, you will see a log-in box.



Once logged in, you will be presented with the following screen.



1. **Login information** - displays your nickname, a link allowing to change the nickname, and a button to log out.
2. **On-line users** - shows the list of users that are on-line in the site's chat. Allows to initiate a one-on-one chat session with a user.
3. **Available rooms** - displays a list of chat of rooms that you can join.
4. **Notification area** - contains the name of the room you are in and displays a link which brings up the notifications dialog.

When you click a room in the list of rooms (3.), additional controls appear:

1. **Users in room** - lists all users who are on-line in the room or have elevated permissions for the room.
2. **Send message** - allows to type and send a message.
3. **Recipient** - defines who will see the message. A message can be addressed to everyone in the room or you can select a specific user to send it to. Checking the check-box prevents changing the recipient back to everyone after sending a message to a specific user.
4. **Messages area** - displays messages that are addressed to everyone, and private messages intended for the logged-in user.

8.13.5.3 Example setup

The Chat module with its web parts allows you to set up chat functionality on your site in many different configurations, or modes. The most basic use cases include single page mode and redirection mode, which are described in this topic.

Single page mode

Web parts belonging to the Chat module can be placed on a page one by one, thus compiling the functionality of the Chat web part from multiple components, allowing for customization of appearance and complexity. For example, you can leave out certain components in order to limit the set of features users of the chat can utilize. The following text explains how to set up multiple-room chat within a single page.

> **Tip**
>
> Chat web parts do not need any configuration in order to function properly. You can

achieve standard behavior by leaving the web parts' properties at their default values. The default values are specified in the module's settings. On the other hand, the web parts and settings allow for a high level of customization.

Start by creating a new blank page. Then place the following web parts on the page. As mentioned previously, you can leave all their properties at their default values.

- **Chat login** - this web part will serve as the starting point for users who wish to enter the chat, as well as an exit point for logged-in users.
- **Chat rooms** - this component will display the list of rooms that the currently logged-in user can join.
- **Chat send message** - this web part will display a text-box to allow sending messages.
- **Chat messages** - this web part will display the most recent messages in a room.
- **Chat leave room** - this web part will display a button that will allow users to leave the room they are in.
- **Chat notifications** - this web part will display system notifications, such as room invitations.

Redirection mode

Many web parts that belong to the Chat module have the **Redirect URL** property. This attribute defines the URL of the page where the chat user should be redirected when performing the action the web part is intended for. For example, in the Chat room list web part, you would set the Redirect URL to the page that contains the Chat room messages web part.

Let's presume that you want to set up chat which will comprise the following pages:

- Logon page
- Page with list of chat rooms
- Chat room page with messages

The following is a step-by-step tutorial how to set up chat on the pages. The procedure is presented on the sample Community site.

Please note that unless instructed otherwise, you can leave all the web parts' properties at their defaults.

Preparation

1. Create a new blank page called **Chat**. This will be the starting page, which the users will use to enter the chat.

2. Under the Chat page, create two blank pages called **Room list**, and **Room**.

3. On the **Room list** page, select the **Properties -> Template** tab, then click **Edit template properties** and select **Inherit all** in the **Inherit content** field. Then save the template.

4. Repeat the previous step on the **Room** page.

Basic setup

1. On the **Chat** page, add the **Chat login** web part. Set its properties as follows:

- **Redirect URL after entering chat**: ~/Chat/Room-list.aspx
- **Redirect URL after logout**: ~/Chat.aspx

This will ensure that users will be redirected to the correct page when logging in and out.

2. Place the **Chat notifications** web part on the **Chat** page.

This will allow users to see notifications about invitations to rooms and private conversations, as well as other system messages addressed specifically to the current user.

3. Place the **Page placeholder** web part on the **Chat** page.

The web part, along with the template settings defined previously, will cause the Chat login web part to appear on all subpages, allowing users to log out at any time.

4. On the **Room list** page, add the **Chat rooms** web part. Set the following property:

- **Redirect URL after join**: ~/Chat/Room.aspx

When a user clicks a room in the list, the web part tries to join the user to the room. If that action is successful, the system redirects the user to the specified URL.

5. Place the **Chat send message** web part on the **Room** page.

This will allow sending messages to the room and its members.

6. Place the **Chat room messages** web part on the **Room** page.

This web part will display messages from other users in the room, as well as system messages and announcements.

7. Place the **Chat leave room** web part on the **Room** page. Set the following property:

- **Redirect URL after leaving**: ~/Chat/Room-list.aspx

This will provide user a way to exit the room and return to the page with the list of rooms.

Extended setup

After completing the preceding procedure, you will have implemented a functioning multi-page chat with all the basic features. However, this configuration can be extended with additional features. The process is described in the following step-by-step tutorial.

1. Place the **Chat on-line users** web part on the **Room list** page.

This will enable users to see the list of users on-line in the chat regardless of the room they are in, if any. They will also be able to start a private conversation with any of the on-line users.

2. Add the **Chat room name** web part on the **Room** page.

This web part will display the name of the room the user is in.

3. Place the **Chat room users** web part on the **Room** page.

This will allow users in a room to see who else is on-line in the particular chat room and initiate a one-to-one chat session with them.

8.13.5.4 Web part grouping

All chat web parts share a common property - **Group name**. This property allows you to have multiple instances of the same web part in one chat implementation, with each of the instances acting on its own.

For example, you can enable chat users to contribute to two chat rooms at the same time.

1. Place the **Chat send message** web part on a page. Set its **Chat room** property to an existing room. Set its **Group name** to **Group1**.

2. Place the **Chat messages** web part on the same page. Set its properties as for the previous web part.

3. Add the same pair of web parts once again, but set their **Room name** property to another existing room and their **Group name** to **Group2**.

When you view the page on the live site, you can see that each pair of web parts is tied to a different chat room and you can use them both at the same time.

8.13.6 Customizing appearance

8.13.6.1 Customizing CSS styles

The Chat web part has its CSS styles defined by default. If you want the individual web parts to look like the components of the Chat web part, you just need to wrap them in the **Gray box with header** web part container.

8.13.6.2 Writing transformations for chat

Chat web parts use **jQuery templates** to display their content. This includes on-line users lists, room lists, messages and other elements. In Kentico CMS, the templates are stored as ordinary transformations that have their type set to jQuery.

You can find all transformations that the Chat module needs in **Site Manager -> Development -> Document types -> Edit (**  **) *Chat - Transformations* -> Transformations**.

A jQuery transformation consists of HTML code and **template tags** that dynamically insert values of the object that is currently being processed. There are also tags that control the flow of the transformation.

Displaying data

An example transformation could look like this:

```
<div>
    <strong>${Nickname}</strong>: ${MessageText}
```

```
</div>
```

The example can be used to display a chat message. It displays the sender's nickname in bold and the text of the message.

The template tags are highlighted in *italics*. In this example, the tags are used to display data. Tags that display data should always start with a dollar sign (*$*), followed by the name of the variable that you want to display enclosed in brackets.

To learn what data you can display, refer to the Reference topic.

Creating conditions

You can also add conditional template tags that will control what content will be displayed based on a true/false condition.

A condition begins with the *{{if}}* tag. As an argument, you can enter any statement that returns a boolean value (*true* or *false*). The following are examples of a valid conditional statement:

- IsSystem
- Nickname == 'administrator'
- ! IsCurrentUser

An *if* condition can have two branches - one that gets rendered when the condition is true, and one indicated by the tag *{{else}}* that gets rendered when the condition is false. You should always close the body of the condition with the *{{/if}}* closing tag.

In the following example, we'll add the recipient's nickname in case the message is a whisper. Otherwise, the transformation will display only the standard nickname and text combination.

```
<div>
    {{if Whisper}}
        <strong>${Nickname} -> ${Recipient}</strong>: ${MessageText}
    {{else}}
        <strong>${Nickname}</strong>: ${MessageText}
    {{/if}}
</div>
```

Conditional tags support multiple conditions joined by logical operators. The syntax is similar to that of C#, as you can see in the following example:

```
{{if IsSystem || Nickname == 'administrator' }}
    <div>
        <strong>System message:</strong> ${MessageText}
    </div>
{{/if}}
```

Integrating macro expressions

You can insert K# macro expressions into templates and combine them with template tags.

By using **localization macros** in transformations, you can ensure that all elements will be displayed in the correct language. The elements that can be translated include:

- system messages
- notifications
- buttons

All of these elements are already prepared in the form of resource strings.

The following code example shows how to inform that a message has been rejected using a resource string.

```
{{if Rejected}}
    <span class="Rejected">{$chat.messagerejected$}</span>
{{/if}}
```

You can find the complete list of resource strings belonging to the Chat module in the Reference topic.

Inserting action commands

Users can perform actions on objects, provided they have permission to do so. For example, you can allow room administrators to reject messages directly in the list, or you can allow them to disable or delete rooms.

The following example shows how you can insert a command control for the **reject message** action. You need to insert the action command into the *onclick* attribute with the same syntax as if you were displaying a variable.

The *if* condition ensures that the action command will be rendered only if the current chat user has permission to reject the message, i.e., if the user is an administrator or a creator of the room.

```
{{if RejectMessage}}
    <a href="#" onclick="${RejectMessage}">Reject</a>
{{/if}}
```

For a complete list of available commands, refer to the Reference topic.

8.13.6.3 Reference

This topic lists variables and commands that you can use in chat transformations. Each section of this topic is dedicated to a single type of object, i.e., messages, rooms, users, notifications, support chat objects and errors.

To learn how to use these values in transformations, refer to the Writing transformations for chat topic.

You can also take inspiration in the default transformations within the **Chat - Transformations** document type.

Messages list

The following table lists data variables that are available when displaying messages. You can either display the variables' values or evaluate them in conditional statements.

Variable name	Data type	Description
AuthorID	int	Contains the database ID of the chat user who sent the message. If the message is a system message, the value is *null*.
IsOneOnOne	bool	Returns true if the message is part of a one-on-one chat session.
IsRejected	bool	Returns true if the message has been rejected.
IsSupport	bool	Indicates if the message is part of a support chat session.
LastModified	date/time	Contains the time when the message was modified.
MessageID	int	Returns the database ID of the message.
MessageText	string	Contains the text of the message.
Modified	bool	Indicates if the message has been modified.
Nickname	string	Contains the nickname of the sender. If the message is a system message, the value is *null*.
PostedTime	date/time	Contains the time when the message was sent.
Recipient	string	Returns the nickname of the message's recipient. If the message is addressed to all users in a particular room, i.e., the recipient is not specified, the value is *null*.
RecipientID	int	Contains the database ID of the recipient, if specified.
SystemMessage Type	int	Contains the type of the message represented by an integer number as follows: • 0 - classic message • 1 - whisper • 2 - leave room • 3 - enter room • 4 - change nickname • 5 - kick • 6 - user invited • 7 - support greeting • 8 - permanently leave room • 9 - permanent kick • 10 - announcement • 11 - declined chat request

The following table lists commands that you can use, for example, in *onclick* events.

Command name	Description
SelectRecipient	Selects the sender of the message in the Recipient drop-down list as the recipient

	of the next message.
SelectPrevRecipient	If the message is a whisper message, this command selects the recipient of the message as the recipient of the next message.
RejectMessage	Rejects the message.

Rooms list

The following table lists data variables that are available when displaying the list of rooms. You can either display the variables' values or evaluate them in conditional statements.

Variable name	Data type	Description
AllowAnonymous	bool	Indicates whether the room allows anonymous users.
CanManage	bool	Indicates whether the current chat user can manage the room.
ChatRoomID	int	Contains the database ID of the room.
Description	string	Contains the room's description.
DisplayName	string	Contains the name of the room.
HasPassword	bool	Indicates whether users are required to enter a password to join the room.
IsCurrentRoom	bool	Indicates whether the current chat user is present in the room. The value takes into account the group that the web part displaying the list is in.
IsPrivate	bool	Indicates whether the room is private.

The following table lists commands that you can use, for example, in *onclick* events.

Command name	Description
Abandon	Removes the current user from the room. Revokes access if the room is private.
Delete	Disables the room and schedules it for deletion.
Delete	Opens the room editing dialog box.

Important!

When rendering rooms, you should always wrap them in an anchor tag with "JoinRoom" as the class attribute value. This will allow users to join the room upon clicking.

Example:

```
<a class="JoinRoom">${DisplayName}</a>
```

Room name

When you want to display the name of the current room in the Chat room name web part, you need to create a separate transformation. You can display the room name in the transformation using the *RoomName* variable.

Users list

The following table lists data variables that are available when displaying the **list of users in a chat room**. You can either display the variables' values or evaluate them in conditional statements.

Variable name	Data type	Description
ChatUserID	int	Contains the database ID of the chat user.
IsAdmin	bool	Indicates whether the user can manage the room.
IsAnonymous	bool	Indicates whether the user is anonymous.
IsChatAdmin	bool	Indicates whether the user has the Manage rooms permission.
IsCreator	bool	Indicates whether the user is the room's creator.
IsCurrentUser	bool	Indicates whether the user is the currently logged in user.
IsOnline	bool	Indicates whether the user is currently on-line.
Nickname	string	Contains the user's nickname.

The following table lists commands that you can use, for example, in *onclick* events.

Command name	Description
AddAdmin	Gives the user administrator rights for the room.
DeleteAdmin	Takes administrator rights away from the user.
KickUser	Kicks the user from the room.
KickUserPerm	Permanently kicks the user from the room. Removes the **Join** permission from the user.
OneOnOneChat	Initiates a one-on-one chat session with the user.

The following table lists data variables that are available when displaying the **list of users that are on-line in chat**. You can either display the variables' values or evaluate them in conditional statements.

Variable name	Data type	Description
ChatUserID	int	Contains the database ID of the chat user.
IsAnonymous	bool	Indicates whether the user is anonymous.
Nickname	string	Contains the user's nickname.

Important!

When listing users who are on-line in chat, you need to include the *${OnClick}* command in an OnClick event. The command ensures that other users will be able to initiate one-on-one chat with the user.

The transformation where you use this command will also be used when inviting users to a room. In this case, the command will make sure that the user gets selected.

Notifications

The following table lists data variables that are available when displaying **notifications**. You can either display the variables' values or evaluate them in conditional statements.

Variable name	Data type	Description
IsOneOnOne	bool	Indicates whether the notification is a one-on-one chat invitation.
KickTime	int	Contains the information about how long a user will be kicked from a room. This variable is only available when the **NotificationType** variable has a value of *4*.
NotificationDateTime	date/time	Contains the time when the notification was created.
NotificationID	int	Contains the database ID of the notification.
NotificationType	int	Contains the integer representation of the notification type. The possible values are the following: • 0 - invitation • 1 - invitation declined • 2 - invitation accepted • 3 - nickname automatically changed • 4 - kick • 5 - permanent kick • 6 - administrator added • 7 - administrator deleted
ReadDateTime	date/time	Contains the time when the notification was read. If the message has not been read, the value is *null*.
RoomID	int	Contains the database ID of the room of concern. If the notification isn't related to room activity, the value is *null*.
RoomName	string	Contains the name of the room of concern. If the notification isn't related to room activity, the value is *null*.
SenderNickname	string	Contains the nickname of the user who performed the action that caused the notification to appear, e.g., the user who invited another user to a room.

The following table lists commands that you can use, for example, in *onclick* events.

Command name	Description
AcceptEvent	Accepts an invitation. Applicable only when the **NotificationType** variable has a value of *0*.
DeclineEvent	Declines an invitation. Applicable only when the **NotificationType** variable has a value of *0*.
CloseEvent	Closes (deletes) a notification. Applicable for all notification types except for invitations (**NotificationType** is not *0*).

Support request

When writing a transformation for the Chat support request web part, you can use the **LiveSupport** variable, which indicates whether a support person is on-line.

Initiated chat

Transformations for initiated chat must have the following format:

```
{{if MessagesTemplate}}
    <code used for messages>
{{else}}
    <code used for the whole web part>
{{/if}}
```

In the code used for messages, you can use the **Text** variable to insert the text of the message.

In the code for the web part, you can use the following variables and commands:

- Accept - command that accepts the initiated chat and opens the support chat window.
- Reject - command that rejects the initiated chat.
- Initiator name - string variable containing the name of the initiator.
- Messages - string that contains all messages.

Errors

When writing a transformation for errors, you can use the **Message** string variable, which contains the error message. It is recommended to insert it in the following form so as to resolve potential HTML code.

```
{{html Message}}
```

When writing a transformation for the Delete all button in the list of errors, you can use the **DeleteAll** command, which deletes the errors.

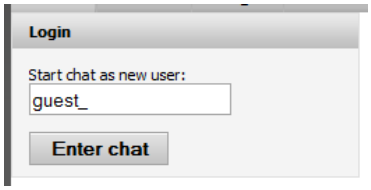
8.13.7 Using chat

8.13.7.1 Logging in

This topic explains the process of logging in to chat on the live site and the differences of this process for anonymous and registered users.

Anonymous users

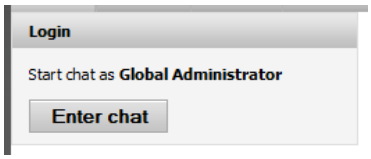
When an anonymous user views a page that contains the Chat web part or the Chat login web part, she is presented with the following login box.



The text box provides an option to choose a nickname. It is automatically pre-filled with text defined in the **Guest prefix** settings key. If the user chooses not to change the predefined nickname, the system will append a number to the prefix and log the user in. Users cannot append any custom text to the guest prefix.

Registered users

When a registered and logged-in CMS user accesses a page that contains the Chat web part or the Chat login web part, she is presented with the following box:



The login box allows the user to enter the chat with the nickname specified in their account properties. If the user hasn't specified a nickname, their full name is used. Note that the chat nickname can be changed after logging in using the **Change nickname** link.

Related settings

The following are settings from the **Community -> Chat** category, which modify the behavior of the login box.

- **Guest prefix** - indicates the prefix anonymous users will be offered to include in their chat nickname.
- **Allow anonymous users globally** - determines whether anonymous users will be allowed to join chat. Note that even if the setting is on, you still can restrict anonymous access to individual rooms.
- **Force anonymous unique nicknames** - indicates if anonymous users should be forced to choose unique (unoccupied) nicknames.

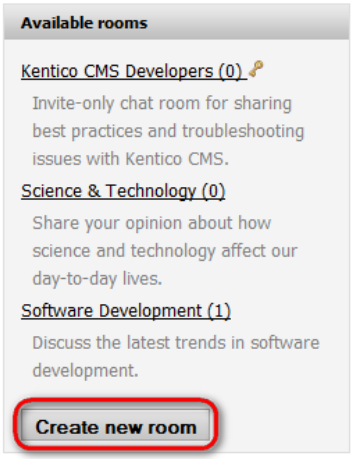
8.13.7.2 Chat rooms

This topic describes functions that relate to chat rooms.

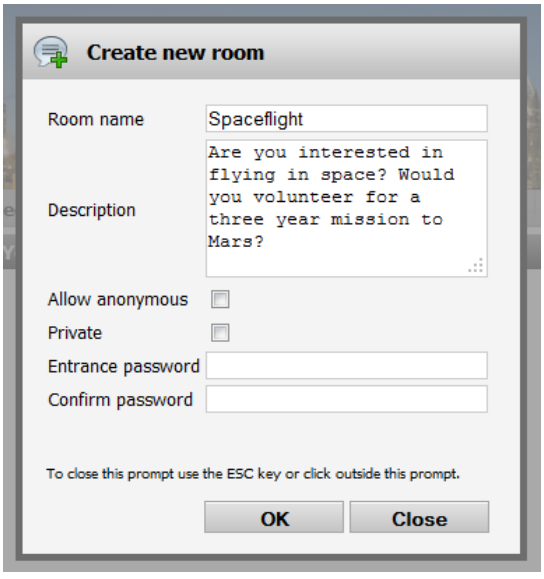
Creating a room

Chat rooms can only be created by registered users who have the appropriate module permission - **Create room from live site** or **Manage**. Anonymous users cannot create new rooms.

A new room can be created using the **Create new room** button in the bottom of the list of rooms.



After clicking the button, the user is presented with a dialog offering the same set of properties as in the room management UI. The only difference is that she cannot create support chat rooms.



The room will be created after clicking **OK**.

Joining a room

The list of rooms displays rooms that the current user is allowed to join. For example, if the current user is anonymous, she can only see chat rooms that have the **Allow anonymous** property set to true.

Special types of rooms are marked with icons:

- - password protected room
- - private room

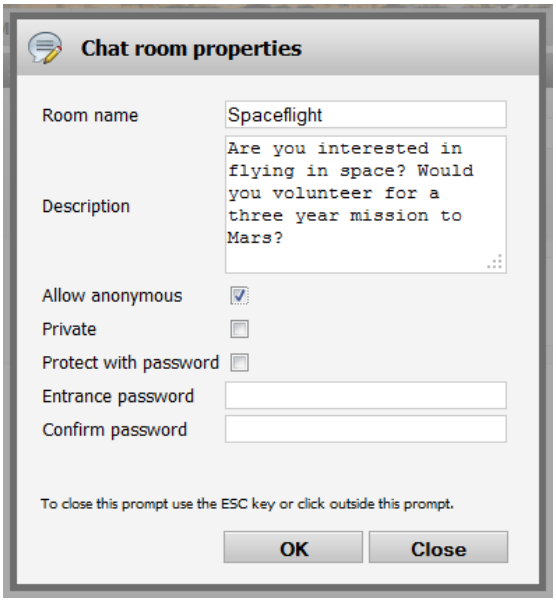
Clicking a room will make the user join the room. If a room is password protected, the user will first be asked to enter the password via a dialog.

Managing a room

Room creators and administrators have the right to edit parameters of the room, add and remove other administrators and kick users. These actions are provided by action buttons in the list of rooms and the list of users in the room. The buttons appear when moving the mouse over the particular room or user and only in case the current user has permission to perform the actions.

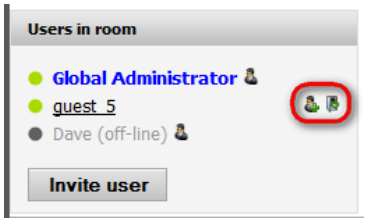
Room actions

- **Edit** - displays a dialog allowing to change the properties of the room.



- **Disable** - removes all users and makes the room unavailable for other users.
- **Abandon** - removes the current user from the room and discards her invitation. Available only for private rooms.

User actions

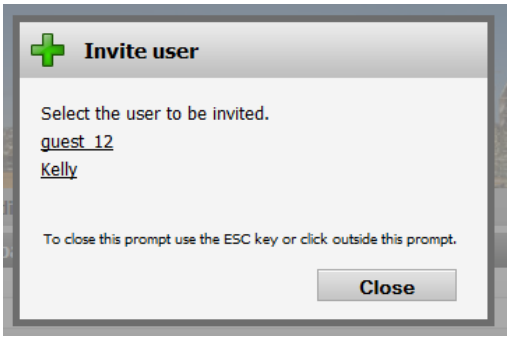


- **Add room admin** - gives the user room administrator permissions.
- **Delete room admin** - takes administrator permissions away from the user.
- **Kick** - prevents the user from accessing the room for a certain period of time, specified by the **Kick lasting time** setting.
- **Revoke access** - kicks the user and removes their invitation, thus preventing them from accessing the room further.

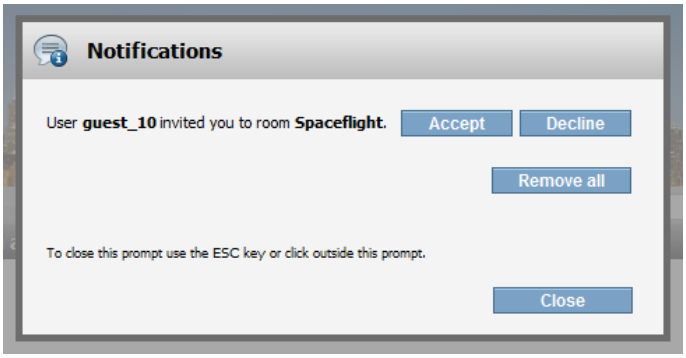
Inviting a user

The list of room users offers a button to invite users to the room. In a public room, the button is visible to all users in the room. In a private room, only administrators can see the button and invite users.

Clicking this button will bring up a dialog allowing to select the user to be invited. The dialog lists only users who are on-line in the chat but not present in the room.



When a user is selected, they will receive a notification about the invitation and will be offered the option to accept and join the room or decline the invitation.



Leaving a room

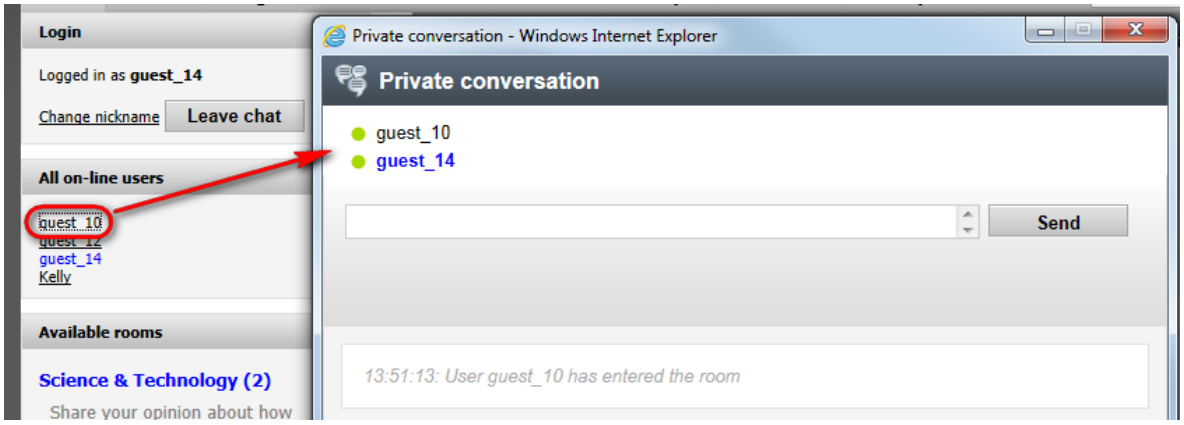
To leave a room, users need to click the **Leave room** button, by default located in the upper right corner of the Chat web part layout.

Starting private conversation

When a chat user clicks another user's name either in the list of on-line chat users or in the list of room users, a private conversation window pops up. At the same time, a new private room is automatically

created and the clicked user is invited to join the room. Since the room is private, other users cannot see it in the list of rooms.

After the invited user accepts the invitation, they automatically join the private room, which is then displayed in the private conversation window.



8.13.7.3 Chat messages

This topic presents the various types of chat messages and ways to send them.

Message types

Users of the Chat module may encounter the following types of messages:

- **Classic** - messages posted on the live site into a room.

  10:23:19 **guest_10** : Hello!

- **Whisper** - messages posted in the room, but only to a particular user designated as recipient.

  15:51:19 **From guest_5** : OK, looking forward!
  15:51:06 **To guest_5** : Meet me tonight at my place.

- **System** - messages generated by the system, informing about users joining and leaving rooms, and sent invitations.

  10:20:56: User guest_12 has entered the room
  9:20:10: User guest_14 has left the room

- **Announcement** - messages sent from the CMS Desk UI.

  10:25:31: Users not adhering to rules will be banned!

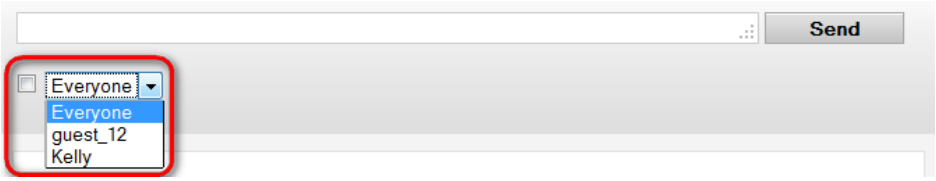
When a user joins a room, system messages posted earlier are not displayed to them by default. This behavior can be adjusted in **Site manager -> Settings -> Community -> Chat -> First load of messages**.

Sending messages

Once joined in a room, users can start sending messages. A message typed into the textbox can be sent either by clicking the **Send** button or by pressing **Enter**.

A new line can be inserted by pressing **Shift + Enter** at the same time.

Users can send messages either to all users present in the room, or to a selected user. Recipient of the message can be specified via the drop-down list below the message text box.



If user selects **Everyone**, the message will appear to all users in the room.

If a particular user is selected, then only the selected user will be able to see the message. The following example shows how such message appears to the sender:

10:33:13 **To Kelly** : This is a private message.

The recipient will see the message as follows:

10:33:13 **From guest_10** : This is a private message.

After a private message is sent, the selector will revert the selected recipient back to Everyone. To prevent this and keep sending messages to a specific user, the sender should check the check-box on the left-hand side of the selector.

8.13.8 Support chat

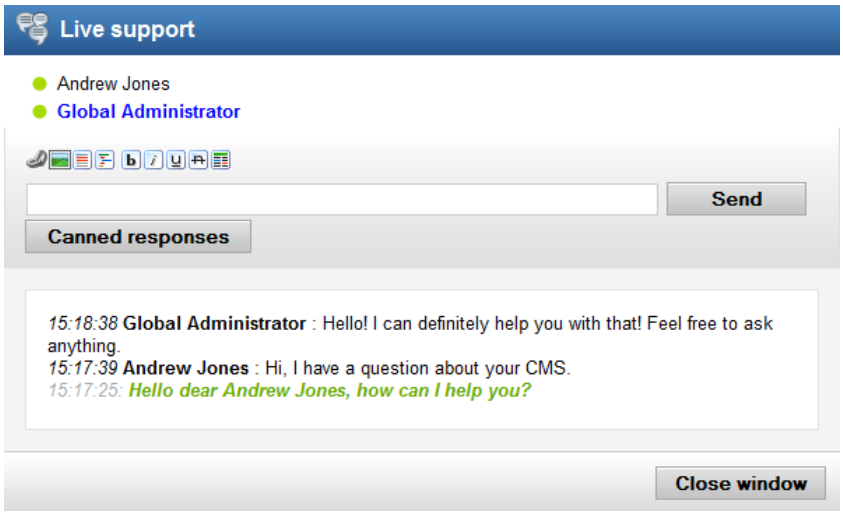
8.13.8.1 Overview

Support chat offers means for customer care departments of companies to provide help to their customers and answer their questions, which can be asked directly from the website, in real time.

This part of the Chat module is composed of several components:

- Support chat web parts - Chat support request, Initiated chat and Automatically initiated chat
- Notification and settings area

- Live support window



The available web parts differ in the way the chat is initiated:

- The **Chat support request** web part displays a button that the visitors can use to initiate chat with support personnel.
- The **Initiated chat** web part provides a way for support personnel to start chatting with on-line visitors from the interface in **Administration -> Users -> On-line users**.
- The **Automatically initiated chat** web part automatically sends the visitor a message after a specified period of time.

Regardless of the web part used to start the conversation, when a visitor sends the first message, the system creates a new support room and lights up a notification in CMS Desk. On-line support personnel then can enter the room and start attending to the customer.

8.13.8.2 Setup

This topic describes how to set up user accounts for support personnel and place the support chat on the live site.

Support chat permissions

With the Chat module comes a global role - **Chat support engineers**. The role already has the permission required for handling support requests from site visitors.

The permission - called **Enter support** - allows the notification area to be displayed and the support engineers to take rooms.

Setting up web parts

To allow site visitors to chat with the support personnel, you need to place one of the following web parts on a page. The following text describes the specifics of the individual web parts.

The web parts render all content based on transformations. The default transformations are defined in settings, so you can leave their properties dedicated to transformations blank. To learn how to write transformations for chat, refer to the Writing transformations for chat topic.

Chat support request

Displays a control that visitors click to start chatting with a support person. In this case, the site visitor initiates the chat and the support person receives a notification in the user interface. See the Notifications topic for details.

If no support person is on-line, the request will be sent by e-mail or won't be available at all. You can choose between these options in **Settings -> Community -> Chat -> Support chat**.

Initiated chat

When you place this web part on a page, you will be able to start chat with visitors who are present on the page.

You will be able to chat only with registered users who are logged on. To do that, navigate to Administration -> Users -> On-line users and click the Initiate chat button to start chatting with the user.

Please note that you need to enable the on-line users feature first.

Automatically initiated chat

This web part automatically displays a pre-defined message to all visitors after a specified amount of time that they spend on a page. When the message appears, they have two options:

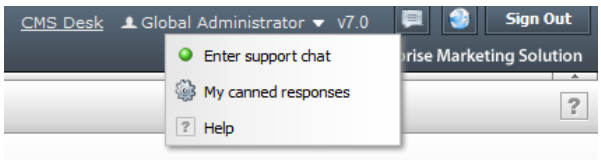
- Reply - opens the support chat window. The visitor can now reply to the message, which then lights up a notification for support personnel in the user interface.
- Close - closes the message without replying. This prevents the message from appearing again for an hour.

The message won't appear if no support person is on-line.

Going on-line

When you're logged in to **CMS Desk** or **Site Manager** and have the **Enter support** permission, you can see the following icon in the top right-hand part of the user interface:

When you click the icon, a context menu appears:



The context menu offers the following functions:

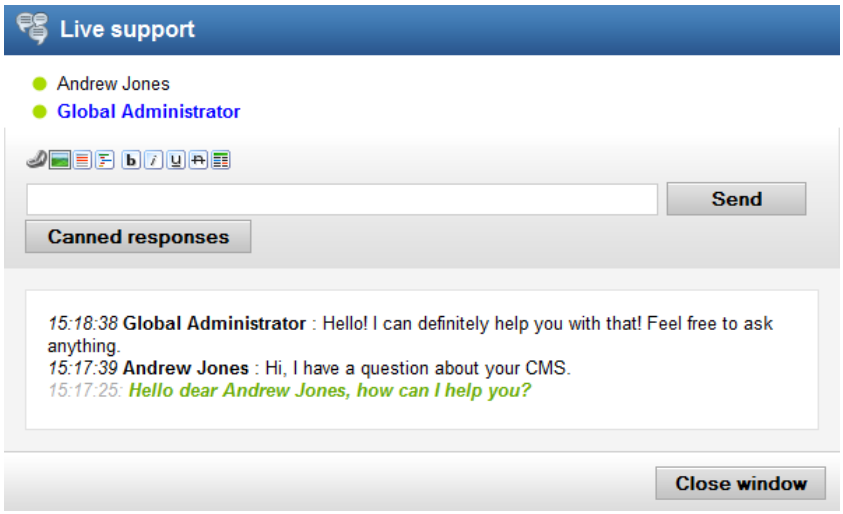
- Enter support chat - brings you on-line and replaces itself with a button to go off-line.
- My canned responses - see Canned responses.

8.13.8.3 Notifications

When a user requests support on the live site and sends a message, all support personnel who are on-line will receive a notification.

When a user sends a message to the room that has been created for her, the room selector will appear in the notification area. The room will be added to the list of currently active support rooms. From the drop-down list, you can select the room you wish to take and provide support in.

Clicking a room will open a window similar to what appears to visitors requesting support.



After a support person opens (takes) the room, notifications will appear only to her.

You can hide the room selector using the icon. You can show it again by clicking **Support chat** () and then clicking **Show active rooms**.

8.13.8.4 Canned responses

Canned responses are pre-defined bits of text that are commonly used when communicating with customers. Support engineers usually define greetings and answers to frequently asked questions as canned responses in order to shorten the time needed to satisfy the customer.

The Chat module allows to define canned responses on two levels in terms of availability to users - publicly - for all support engineers, and privately - only for a particular user.

Public canned responses can be managed in the Chat UI in **CMS Desk -> Tools -> Chat -> Canned responses**.

Creating canned responses

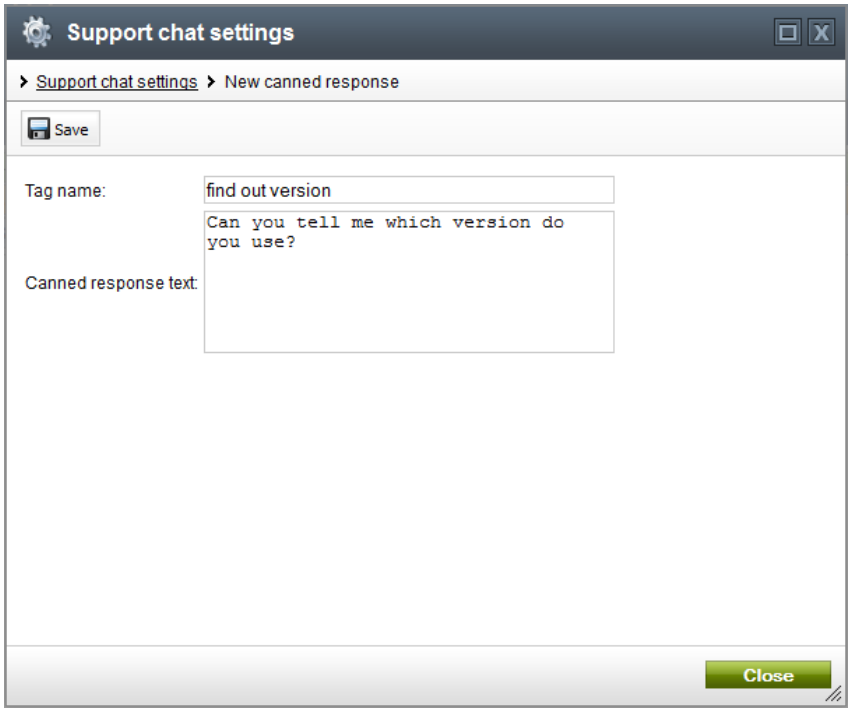
To create a canned response, first select whether you want it to be global, i.e. available on all sites, or

select a particular site in the **Site** drop-down list. Then click **New canned response**.

A canned response consists of a tag name and the response text itself.

To create a personal canned response, click **Support chat settings**.

In the windows that pops up, click **New canned response**. You can then input the tag name and canned response text and click **Save** to save the response.

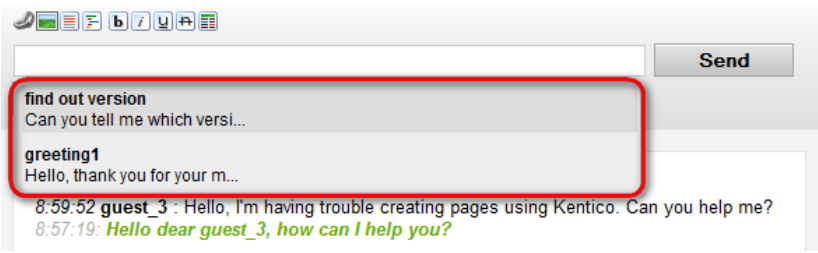


Editing and deleting canned responses

You can edit both public and private canned responses by clicking the **Edit** action button in their respective list. You can delete responses using the **Delete** button.

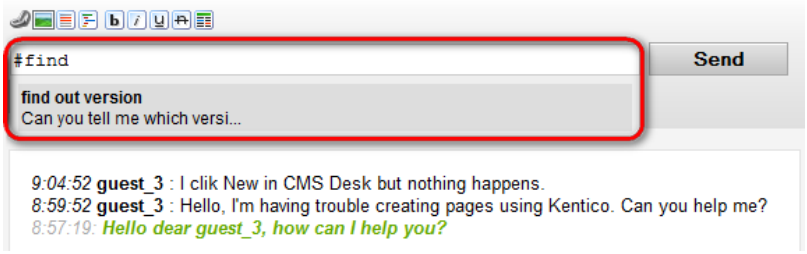
Inserting a canned response

To insert a canned response using the mouse, click the Canned responses button, then click one of the canned responses in the list that appears.



To insert a canned response using the keyboard, type "#" into the message text box. Then select the

desired canned response using the arrow keys. You can find a particular response by typing the first characters of the tag name. Press **Enter** to insert the selected response.



Macros in canned responses

Support chat allows inserting macro expressions into canned responses. Entered macro expressions are resolved upon opening the support room window. Supported are the standard macro types, including custom macros.

8.13.9 Cleanup

In order to minimize the amount of storage space that the Chat module's data consumes, it incorporates a set of scheduled tasks, which delete old and inactive records on a regular basis.

Delete old chat records

This scheduled task deletes chat messages, rooms and anonymous users which haven't logged any activity for the amount of time specified in settings. The age of the data that is to be deleted can be specified in **Settings -> Community -> Chat -> Days of inactivity needed to delete chat records**.

The order of performed deleting operations is as follows:

1. **Delete messages** - messages that were sent earlier than the specified amount of time.
2. **Delete rooms** - rooms that meet **all** of the following criteria will be deleted:
   a. is private
   b. contains no messages
   c. was created earlier than the specified amount of time
   d. doesn't contain any registered (non-anonymous) user that has elevated room permissions
   e. doesn't contain any anonymous user that has been on-line in the room during specified amount of time
3. **Delete users** - users meeting **all** of the following conditions will be deleted:
   a. haven't sent any message in the specified amount of time
   b. haven't been on-line in the specified amount of time

Clean chat on-line users

This scheduled task removes anonymous users who stopped sending pings to the server without going off-line (e.g. their connection was interrupted or they simply closed the browser window).

Clean chat rooms scheduled to delete

When you delete a room, it becomes disabled and scheduled to be deleted instead of being physically

deleted from the database right away.

This scheduled task is by default executed once a day to delete the rooms marked for deletion.

This behavior ensures that users present in such room have a chance to get notified about the fact that the room they are in is getting deleted.

8.14 Contact management

8.14.1 Overview

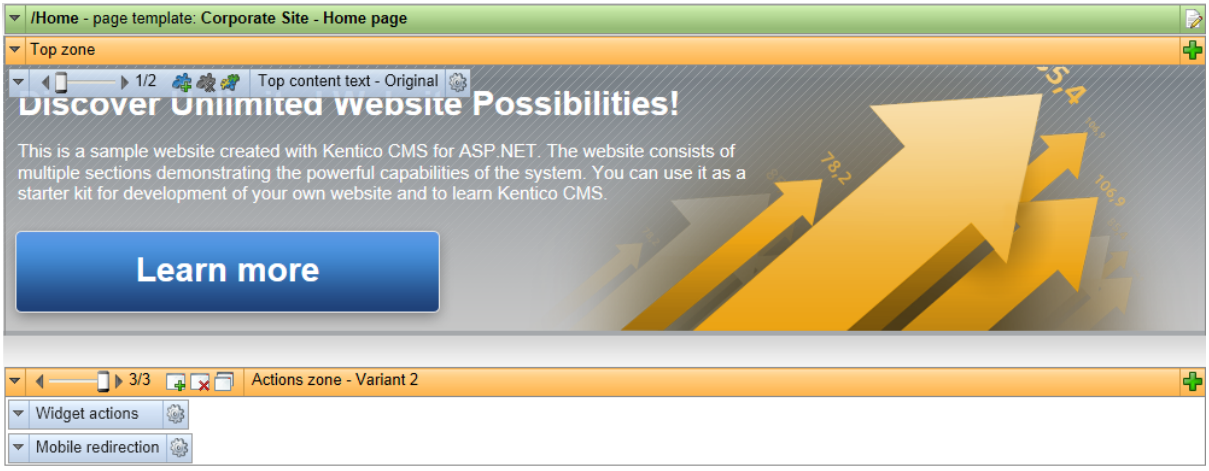
For more information about the **Contact management** module please refer to the Kentico CMS Online Marketing Guide.

8.15 Content personalization

8.15.1 Overview

Content personalization is an on-line marketing feature that can significantly increase the flexibility of a website. It allows you to create pages that display different content depending on the circumstances in which they are viewed. This way, pages can be custom-built for different types of visitors, dynamically reflect previous actions performed on the website by a given user or have their content determined by any other site variables.

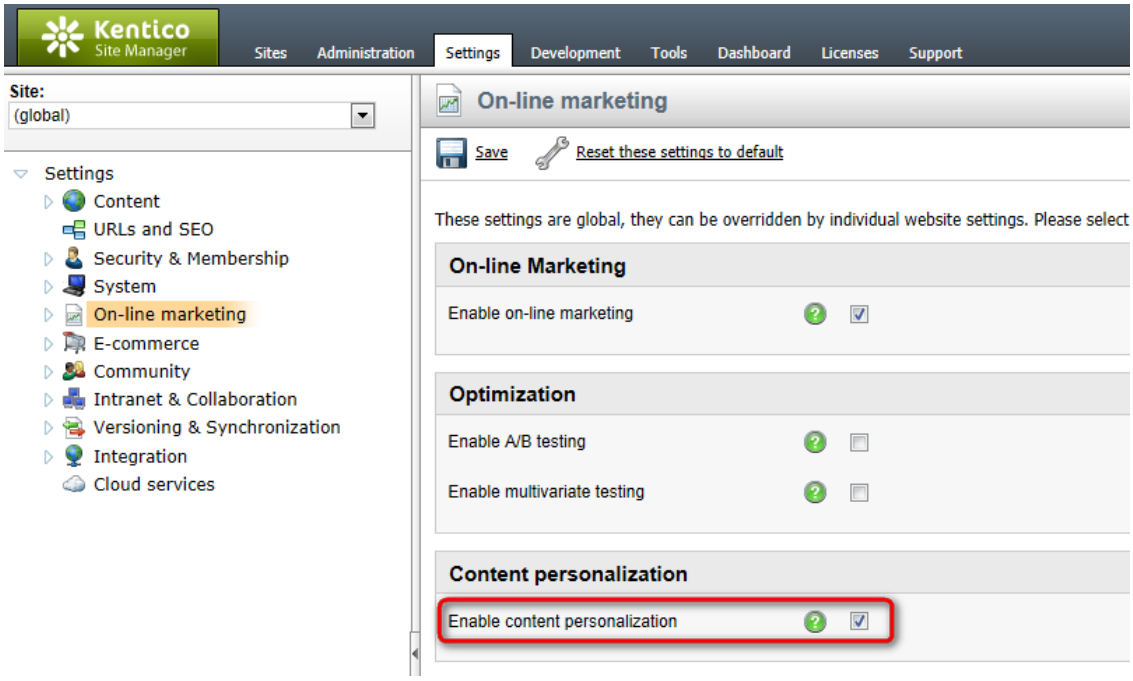
Personalization is applied through the basic components that form the content of pages, which includes Web parts, entire web part zones and Widgets added into page editor zones. The first necessary step is to define different versions of these objects called *variants*. The details of how personalization variants can be created and configured for objects are described in Managing personalization variants.



Each variant is dedicated to a particular scenario, which is specified through a macro-based condition. The condition determines in which situations the variant should be displayed. When a visitor selects the page on the live site, the variants of all personalized objects will have their conditions dynamically resolved according to the current context and the appropriate variant will be displayed for each object. Please see the Variant conditions topic to learn more about possible condition options and how they are processed.

Enabling content personalization

To start using content personalization on a website, go to **Site Manager -> Settings -> On-line marketing** and check **Enable content personalization**. This will allow users with the appropriate permissions to start creating variants of objects, which will then be processed and displayed accordingly on the live site.



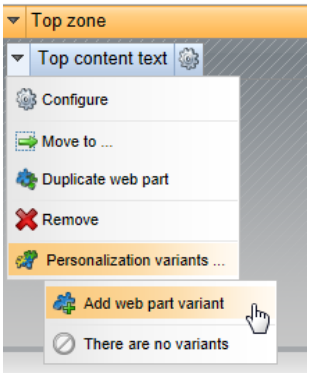
If you disable content personalization at a later time, existing variants on your website will not be deleted, but it will not be possible to manage them or define new ones, and the original objects will always be shown on the live site.

8.15.2 Managing personalization variants

Once content personalization is enabled, variants of objects can be created through the **CMS Desk -> Content -> Edit** interface, which you would normally use to define standard page content. There are three possible types of objects that you can personalize:

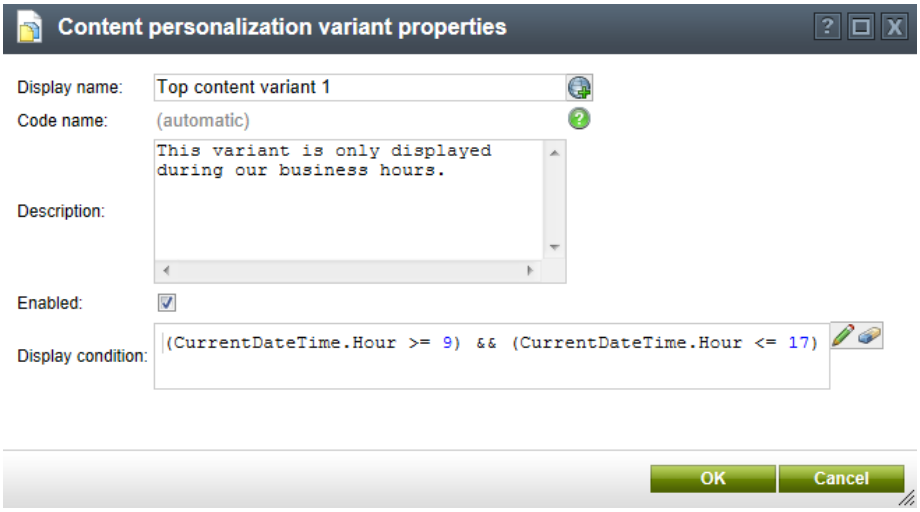
Web parts

To add the first content personalization variant to a web part, open its context menu by right clicking its header (or through the ▼ icon), hover over the **Personalization variants** option and then select **Add web part variant** from the second level of the menu.



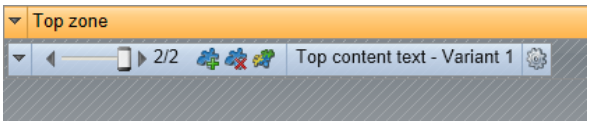
A dialog will be displayed where you can set the properties of the variant:

- **Display name** - this name will be displayed in lists of content personalization variants in the administration interface.
- **Code name** - sets a code name that serves as an identifier of the variant.
- **Description** - can be used to enter a text description for the variant. To make the variant easier to use and maintain, you can add an explanation about the scenario for which the variant is intended, describe the differences from the original object, etc.
- **Enabled** - indicates if the variant should be considered as a possible content option. When an object's variants are processed to determine which one should be displayed, disabled variants are skipped (even if the requirements set by their condition are met).
- **Display condition** - enter the condition that must be fulfilled in order for the variant to be displayed. Please see the Variant conditions topic, to learn more about these conditions and how they are processed.



When the values are confirmed, a standard web part configuration dialog will be displayed. The variant is simply another instance of the original web part. By default, it will have the same values in its properties as are set for the original, but you can change them as required.

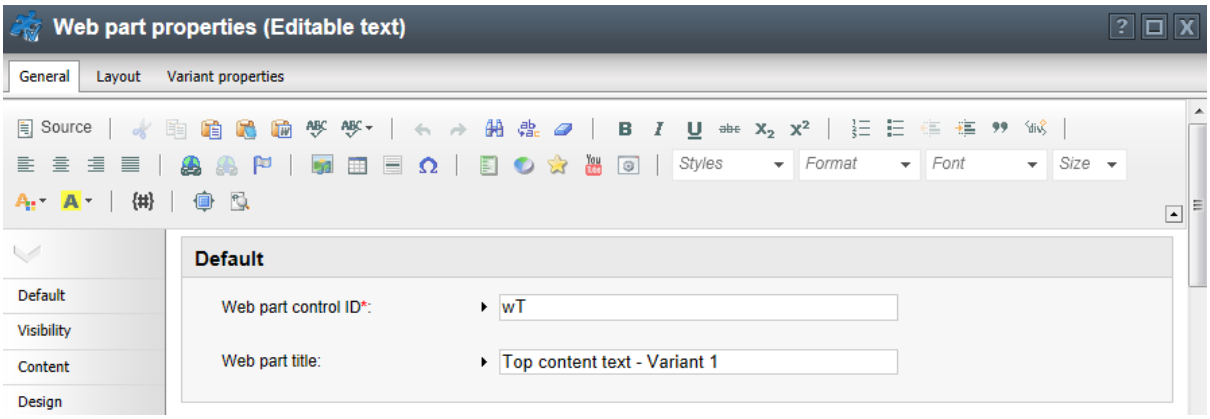
After the first variant is created, a slider will be added to the header of the given web part as shown below:



This slider can be used to switch between individual variants as needed (including the original). The content specified by the currently selected variant will be displayed on the **Design** and **Page** tabs, as well as in **Preview** mode. The conditions are only resolved on the actual live site. This means that you can easily check how the page will look with different active variants without having to fulfill the required conditions. Simply set the slider(s) to the appropriate variants.

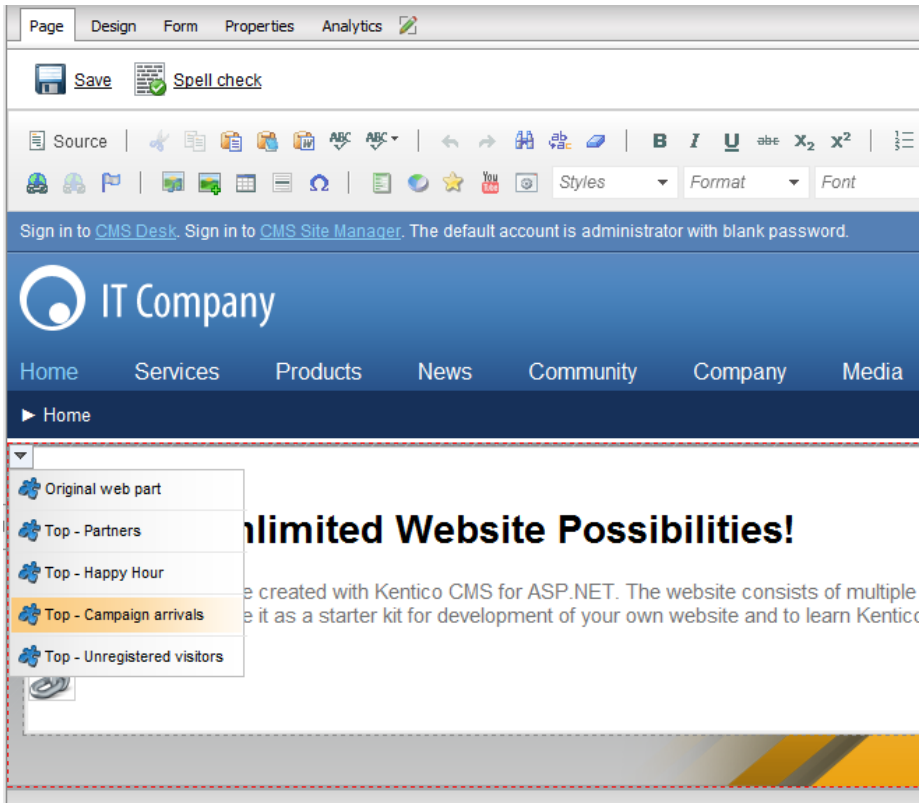
If you wish to view the content of variants while cycling through the slider on the **Design** tab, make sure that the **Display web part content** checkbox on the right side of the **Edit** mode header is checked. To make orientation easier, it is recommended to assign an appropriate **Web part title** for each variant, so that you can identify which one you are working with straight from the header text.

The buttons on the right of the slider allow you to add new variants (), remove the currently selected one () or open a list of all variants defined for the given web part (). At any time, you can **Configure** ( ) the properties of the variant currently chosen on the slider.



A variant can be edited just like any other web part. Notice that there is an additional tab available in the configuration dialog called **Variant properties**, which contains the same options that were offered when creating a new variant (as described above). It is also possible to specify a unique Web part layout for each variant. You only need to switch to the **Layout** tab, select or create the required layout and it will be applied to the currently edited variant.

If you need to enter content into a variant of a web part that provides an editable region (Editable text or Editable image), you can do so on the **Page** tab as usual. To quickly switch between different variants on this tab, hover over the given personalized section of the page and a menu icon (▾) will be displayed. If you expand the menu, a list of all available variants will be shown and you can select the one that you wish to edit or view.



When one of the menu options is clicked, the page will be reloaded with the content of the chosen variant.

The variants of a personalized web part are stored on the given document's page template, so they will be present on all pages that use the same template. If you delete (✖) a web part, all of its variants will be removed along with it. These same principles also apply to web part zone variants.

Web part zones

If you wish to define personalized content that uses a completely different type of web part or multiple web part instances, you can add variants of an entire web part zone. This allows you to set the properties of the zone in a specific way for each variant, and more importantly add (or remove) any child web parts that you need. Each zone variant may contain any type or amount of web parts, regardless of what is placed inside other variants or the original zone.

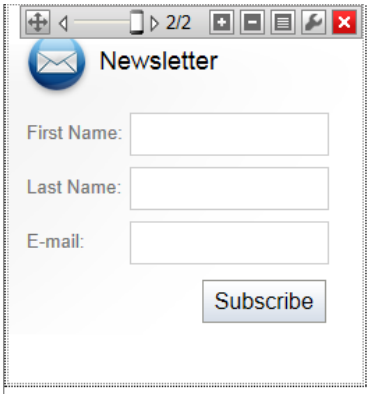
Variants of zones can be created and managed on the **Design** tab in the same way as with web parts (the buttons on the slider use different icons for zones). The only difference is that you need to work with zone objects rather than specific web parts. When a new zone variant is added, the content of the original is automatically copied into it, so you do not have to rebuild the zone from scratch if you only need to make small modifications.

The zone variant currently selected on the slider may be edited by double clicking the header of the zone or through the **Properties** option in the zone's context menu. To add content into a variant, use the **Add web part** button in the top-right corner of the zone as usual. Most of the functionality described above for web part personalization is also available for zones.

Creating variants for individual web parts inside a personalized zone is not supported. The same also applies in the opposite direction: it is not possible to personalize a zone that already contains personalized web parts. Moreover, personalization is only available for standard web part zones, and cannot be done for widget zones.

Widgets

Content personalization can also be leveraged by editors, who do not have access to the **Design** tab, via widgets. If the feature is enabled, variants can be added and managed through buttons on the pop-up menu of individual widgets on the **Page** tab. When at least one variant is created for a widget, a slider will become available just like for web parts or zones.



The following actions can be used:

- **Add new variant** - creates a new variant for the given widget. When creating the first variant for a widget on a page that has a multivariate test defined, this action will open a menu where you can choose which type of variant should be created.
- **Remove variant** - removes the variant currently selected on the slider (not available for the original).
- **Variant list** - opens a dialog that contains a list of all variants defined for the widget and allows their management.

Personalization of editor widgets works the same way as for web parts. Each variant is a widget of the same type as the original, but its properties may be configured differently. This functionality is only available for widgets placed into zones set for **Customization by page editors** and cannot be done for *group administrator* or *user* widgets.

Since editor widgets are categorized as page content, their variants are bound to the specific document and are not included on the page template. Like with web parts, removing a widget ( ) also deletes all of its variants. Using the **Reset to default** action provided by a Widget actions web part also deletes all widget variants from the page.

Variant overview

You can access a list of all content personalization variants defined on a given document (page), by selecting it from the content tree in **CMS Desk -> Content -> Edit** and going to **Properties -> Variants**. The variants of all three object types are included here.

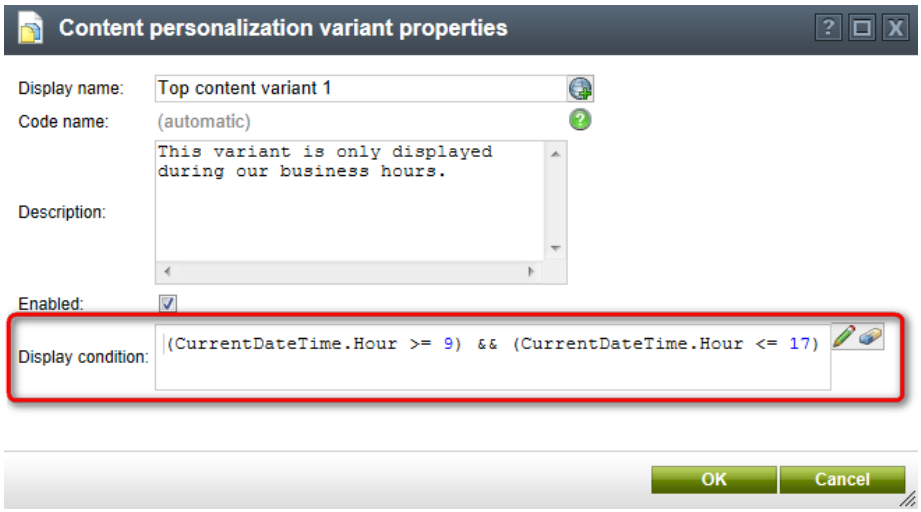
Multivariate testing and Content personalization

Please note that it is not possible to create personalization variants of web parts, zones or widgets that are already included in a Multivariate test.

8.15.3 Variant conditions

When setting up a personalized page, the most important part of the process is to properly define the conditions that indicate when individual variants should be displayed. By utilizing macro expressions, you can write conditions for virtually any type of scenario according to your specific requirements. Keep in mind that the result of a condition's expression must be a logical (boolean) value in order for it to work correctly. For details about available macro options and syntax, please refer to the Development -> Macro expressions chapter.

A variant's condition is specified in its **Display condition** property.

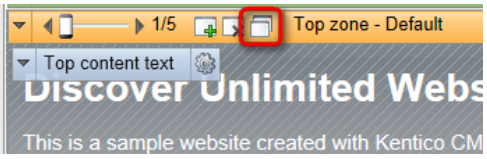


Through the edit icon ( ), you can use the macro condition editor, which provides a graphical interface

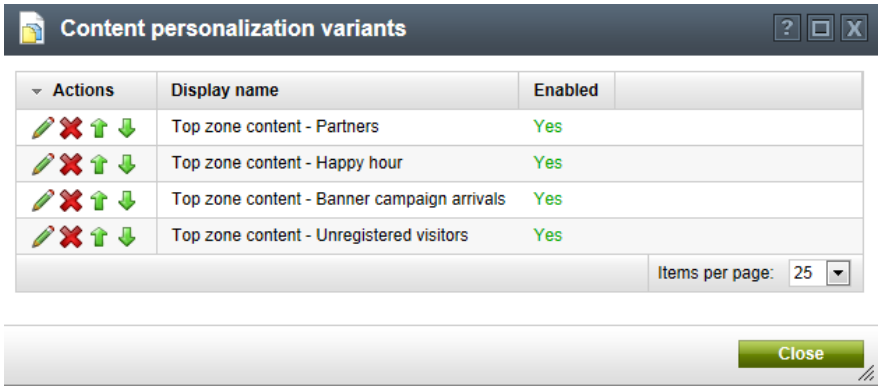
that makes it easier to build complex conditions. It allows non-technical users to create conditions based on predefined macro rules. Please see the Macro conditions topic for further details.

On the live site, each personalized object will only have one of its variants displayed at any given moment. Variants are processed in the order of their priority and the first **enabled** variant whose condition is fulfilled in the current context will be selected. The original object is displayed in cases where the conditions of all variants are resolved as *false*.

By default, the priority of variants depends on the order in which they were created, as can be seen on the personalization slider. Apart from the original object, which is always first, variants with a higher priority can be found further on the left of the slider.



If you wish to change the order in which an object's variants should be processed, click the **Variant list** ( ) button among the actions on the slider. A dialog containing a list of the variants will be opened as shown below.



Here, you can reorganize the variants as necessary through the **Up** ( ) or **Down** ( ) actions.

Condition example

The example below demonstrates how you can build a macro that causes the given variant to be displayed only during a specific part of the day, or to users with special authorization.

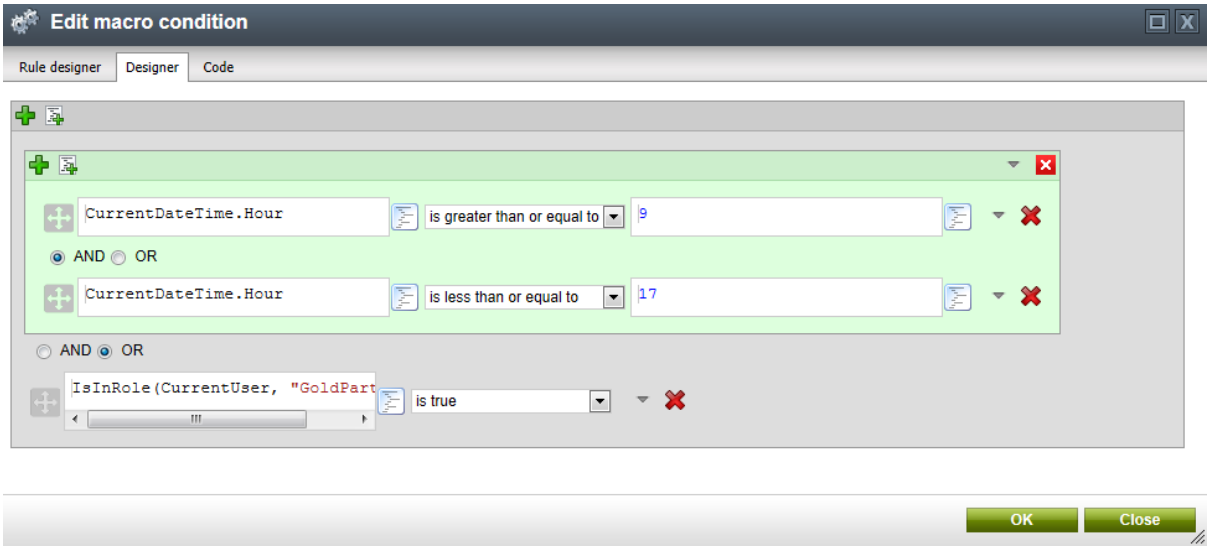
1. Create a content personalization variant for an object, open its properties dialog and click the edit icon ( ) next to the **Display condition** field to open the macro condition editor.

2. Switch to the **Designer** tab and remove ( ) any existing expressions. Then click the **Add Group** ( ) action and use **Add expression** ( ) to create two expression fields inside the new group. Fill in the expressions according to the following instructions:

- Enter *CurrentDateTime.Hour* into the field on the left, select the **is greater than or equal to** operator and write *9* on the right.
- Enter *CurrentDateTime.Hour* into the field on the left, select the **is less than or equal to** operator and write *17* on the right.

Leave the **AND** option selected between the two expressions.

3. Add another expression to the main area and select the **OR** option between it and the group. Enter *CurrentUser.IsInRole("GoldPartners")* into the expression's field and select **is true** as the operator.



4. Click **OK** and the code of the condition will be inserted into the **Display condition** field:

(CurrentDateTime.Hour >= 9 && CurrentDateTime.Hour <= 17) || CurrentUser.IsInRole("GoldPartners")

This condition ensures that the variant is only displayed to ordinary users if the current time is between 9 AM and 5 PM. Users who belong to the *GoldPartners* role will be able to see the content specified by the variant at any time.

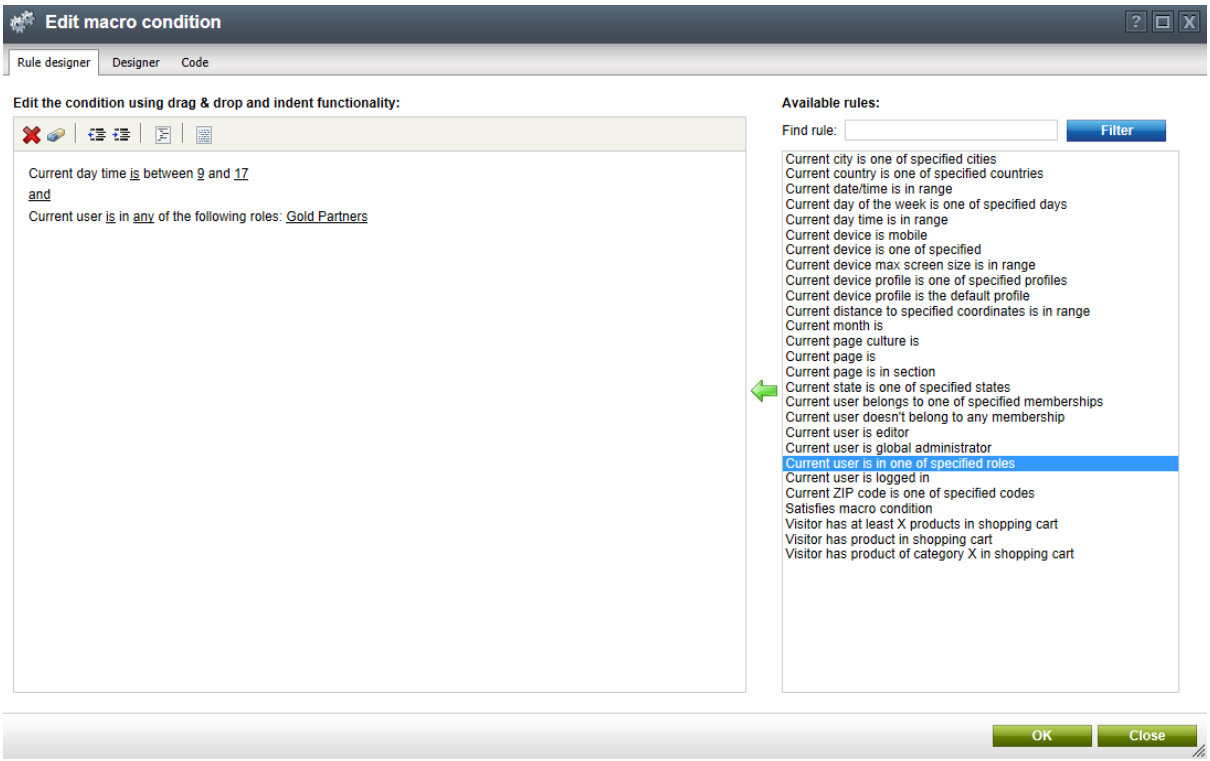
Using macro rules

Alternatively, you can also define the same condition through macro rules. Click the **Clear condition** ( ) action next to the field and edit ( ) the condition again.

1. On the **Rule designer** tab, select the **Current day time is in range** macro rule and use the **Add rule** ( ) button to insert it into the condition.

2. Do the same for the **Current user is in one of specified roles** rule and leave the **and** operator between the two clauses in the designer area.

3. Click on the underlined parameters in the text of the rules to set the time and role values that were used for the original condition.



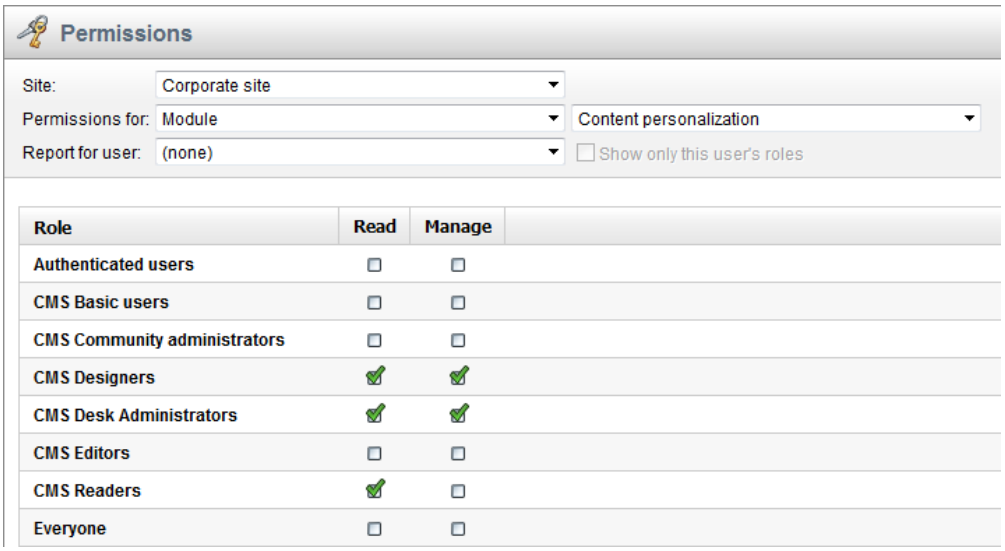
This time, clicking **OK** inserts the variant's display condition as a user-friendly expression composed of macro rule clauses.

8.15.4 Security

Users must have the appropriate permissions assigned to be able to view and manage content personalization variants in the administration interface. These permissions can be set by going to **CMS Site Manager** (or **CMS Desk**) **-> Administration -> Permissions**.

To configure the security options for content personalization, select the *Content personalization* module. The following two permissions can be assigned:

- **Read** - allows members of the selected roles to view the content of personalization variants, their properties and variant lists in the *CMS Desk* administration interface. No special permissions are required to view personalized content on the live site.
- **Manage** - allows members of the selected roles to create, edit and delete personalization variants of objects.



Additionally, the **Design web site** permission for the *Design* module is needed for users to be able to manage the variants of web parts and zones on the **Design** tab of **CMS Desk**.

To work with variants of editor widgets on the **Page** tab, the **Modify** permission for the *Content* module is required. Also, the security settings defined for specific widgets are checked, as described in Development -> Widgets -> Security.

8.16 Content rating

8.16.1 Overview

The Content rating module can be used to give live site users the possibility of rating any document on your website. This may come in handy in case that you want your website users to express their opinion about the quality of published content. The module has no dedicated user interface - the overall rating of each document can be viewed in the **Rating** field on the **CMS Desk -> Content -> Edit -> Properties -> General** tab.

The module's main part is the **Content rating** web part. The web part can be placed on any page and it enables users to rate content of the currently displayed document. More information about the web part and a step-by-step example of its usage can be found in the Using the Content rating web part topic.

It is also possible to include the *~/CMSAdminControls/ContentRating/RatingControl.ascx* control in your transformations in order to ensure the rating possibility with each transformed item. The Displaying ratings in transformations topic will show you how to achieve this.

The content rating functionality is also integrated in web parts of the Message boards module. This enables live site users to submit a rating along with a comment posted to the message board. To learn more, please proceed to the Integration with Message boards topic.

The Content rating internals and API sub-chapter provides information about the database tables and classes used by the module, as well as examples of how ratings can be managed from your code using

Kentico CMS API.

8.16.2 Using the Content rating web part

The **Content rating** web part is stored in the **Content rating** web part category. The web part can be added to any web part zone on any page of your website, enabling users of the website to rate the content of the currently displayed document. It has three default appearance modes:

Stars	Radio-buttons	Drop-down list
 Current rating: 3 (1 ratings)	 1 2 3 4 5 Current rating: 4 (1 ratings)	 Current rating: 2.5 (2 ratings)

It is also possible to create other appearance modes by creating your **custom controls**. These must be placed in *~/CMSAdminControls/ContentRating/Controls/* and inherit from *ExtendedControls.AbstractRatingControl.cs*, just like the default ones.

Below is a list of specific properties of the web part:

- **Rating value** - this property can be used to set the displayed rating value explicitly. A value from the <0,1> interval can be used, while *0* represents the lowest rating and *1* represents the highest rating on the scale defined by the Max rating value. If blank, the real rating calculated as an average value of particular ratings is displayed.
- **Rating type** - appearance of the web part. *Stars*, *Radio-buttons* or *Drop-down list* can be chosen, as depicted above.
- **Max rating value** - size of the rating scale. For example, if 7 is entered, rating will be possible on a scale from 1 to 7.

- **Show results** - if checked, overall rating results will be displayed. If unchecked, users can rate, but don't see the results.
- **Result message** - message showing overall rating results. The {0} macro shows overall rating (for one decimal rounding, you can use {0:0.#}). {1} displays the total number of votes.
- **Message after rating** - text message displayed after a user submits their rating. Macros that can be used: {0} your rating, {1} overall rating, {2} overall number of votes.

- **Check permissions** - if checked, permissions set by the *Allow for public* and *Hide to unauthorized roles* properties will be checked.
- **Allow zero value** - if checked, users are allowed to submit ratings with zero value (no rating value selected).
- **Error message** - message displayed to users who try to submit a rating with zero value. Applied only if the *Allow zero value* option is disabled.
- **Anonymous users can rate** - if checked, rating will be allowed to public unauthorized users. If unchecked, only authenticated users will be allowed to rate.
- **Check if user rated** - if checked, users will be allowed to vote only once. To indicate that the user already voted, Kentico CMS stores a **DocRated** cookie in your web browser, the cookie contains **NodeID**s of the rated documents separated by the tube character (e.g. *|4|61|229|230|228|369|*).
- **Hide to unauthorized users** - if checked, the web part will be hidden to unauthorized users.

Example: Adding news ratings

You can place the web part on any page of your website to enable users to rate the content of this particular page. In the following example, we will add the web part to news items on the sample **Corporate Site**. A similar result can also be achieved using transformations, as described in Displaying ratings in transformations.

1. Sign in to **CMS Desk** as *administrator* (blank password by default). Switch to the **Design** mode and select **News** from the content tree.

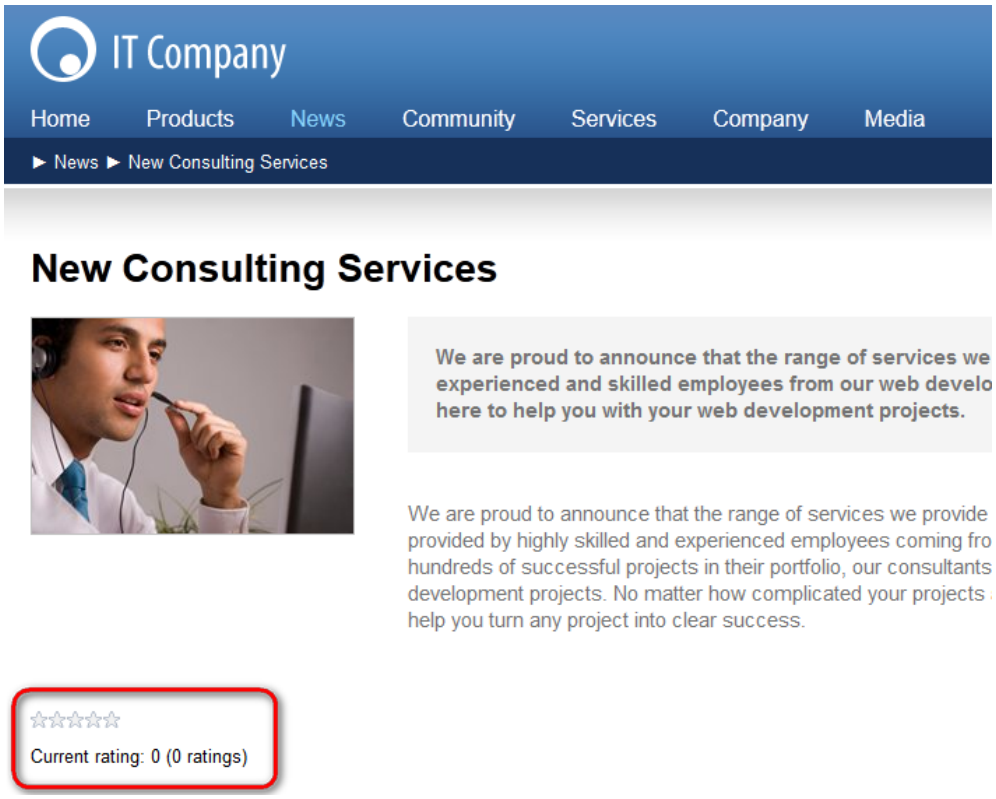
2. Click the **Add web part** (+) icon at the top right corner of the **Main zone** web part zone. Select the **Content rating\Content rating** web part and click **OK**.

3. In the **Web part properties** window, just configure the following property:

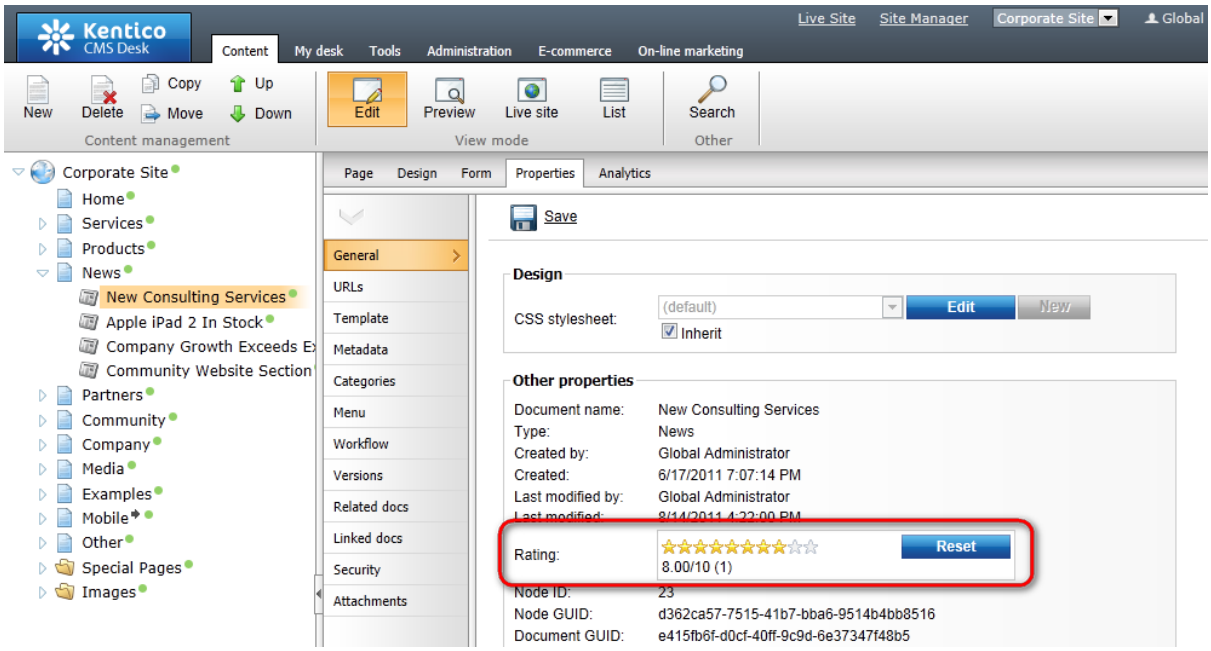
- **Show for document types**: *cms.news*; this ensures that the web part will be displayed only for the particular news items and not for the list of news on the title page of the *News* section.

Leave the rest of the properties at their default values and click **OK**.

4. Now go to the live site and view the **New Consulting Services** news item. You should see the web part below the news text as in the screenshot below. Now try rating!



5. If you switch back to **CMS Desk** and view **Properties** of the news item, you should see the **Rating** property on the **General** tab. This property reflects the current rating of the selected document. All ratings are recalculated to 10 step scale and displayed as stars here, no matter what the settings of the web part are. You can reset the value using the **Reset** button.



8.16.3 Displaying ratings in transformations

Rating controls can also be displayed in Transformations. To do that, the following code needs to be placed in your transformation, ensuring that the *~/CMSAdminControls/ContentRating/RatingControl.ascx* control will be displayed along with the transformed item:

```
<%@ Register Src="~/CMSAdminControls/ContentRating/RatingControl.ascx"
TagName="RatingControl" TagPrefix="cms" %>

<cms:RatingControl ID="elemRating" runat="server" Enabled="true"
RatingType="Stars" ExternalValue='
<%# Convert.ToString(CMS.GlobalHelper.ValidationHelper.GetDouble(Eval
("DocumentRatingValue"), 0)/((CMS.GlobalHelper.ValidationHelper.GetDouble(Eval
("DocumentRatings"), 0) == 0?1:CMS.GlobalHelper.ValidationHelper.GetDouble(Eval
("DocumentRatings"), 1)))) %>' />
```

With this control, you can use similar properties as with the **Content rating** web part, as described in the Using the Content rating web part topic:

- **bool Enabled** - indicates if rating is possible via the control. If disabled, the control is visible, but rating is not possible.
- **bool CheckPermissions** - the same as the **Check permissions** web part property.
- **bool HideToUnauthorizedUsers** - the same as the **Hide to unauthorized users** web part property.
- **bool CheckIfUserRated** - the same as the **Check if user rated** web part property.
- **bool AllowForPublic** - the same as the **Anonymous users can rate** web part property.
- **string ErrorMessage** - the same as the **Error message** web part property.
- **string MessageAfterRating** - the same as the **Message after rating** web part property.
- **string ResultMessage** - the same as the **Result message** web part property.

- **bool ShowResultMessage** - the same as the **Show result message** web part property.
- **string RatingType** - the same as the **Rating type** web part property.
- **string ExternalValue** - similar as the **Rating value** property.
- **bool AllowZeroValue** - the same as the **Allow zero value** property.
- **int MaxRatingValue** - the same as the **Max rating value** property.

Ratings submitted via this control are added to the ratings of the currently displayed document, the same as if you rated via the **Content rating** web part.

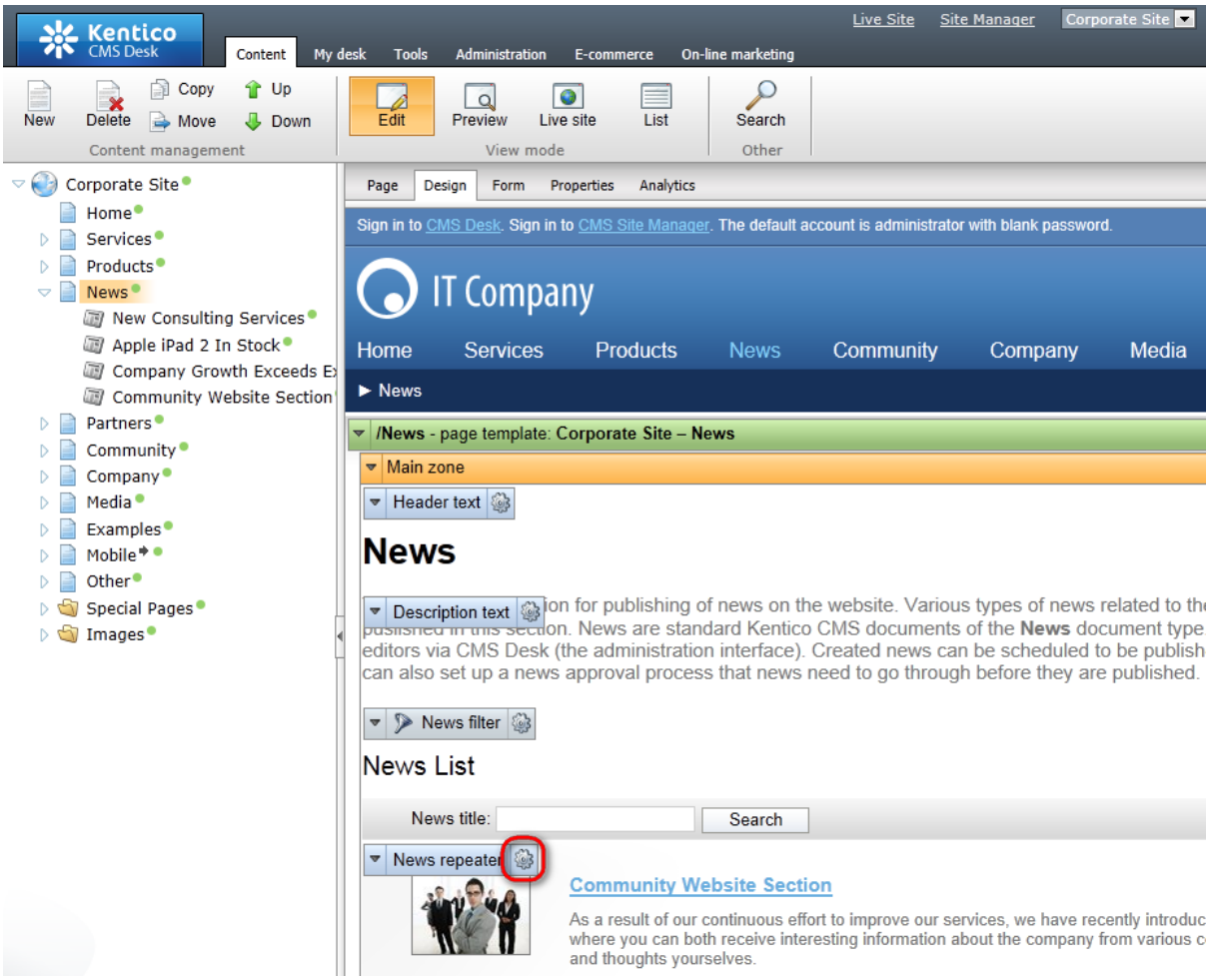
Please note

It is only possible to add controls into transformations that use the **ASCX** type. The rating control will not be rendered correctly by the other transformation types.

Example: Adding news ratings

In the following example, you will learn how to add the rating control to your pages via transformations. We will use the sample Corporate Site and add the rating functionality to news items displayed in the **News** section. A similar result can be achieved using the **Content rating** web part, as described in the Using the Content rating web part topic.

1. Log on to **CMS Desk** and on the **Content** tab, select the **News** page from the content tree. View the page in Design mode and click the **Configure** (⚙) icon of the **NewsRepeater** web part.



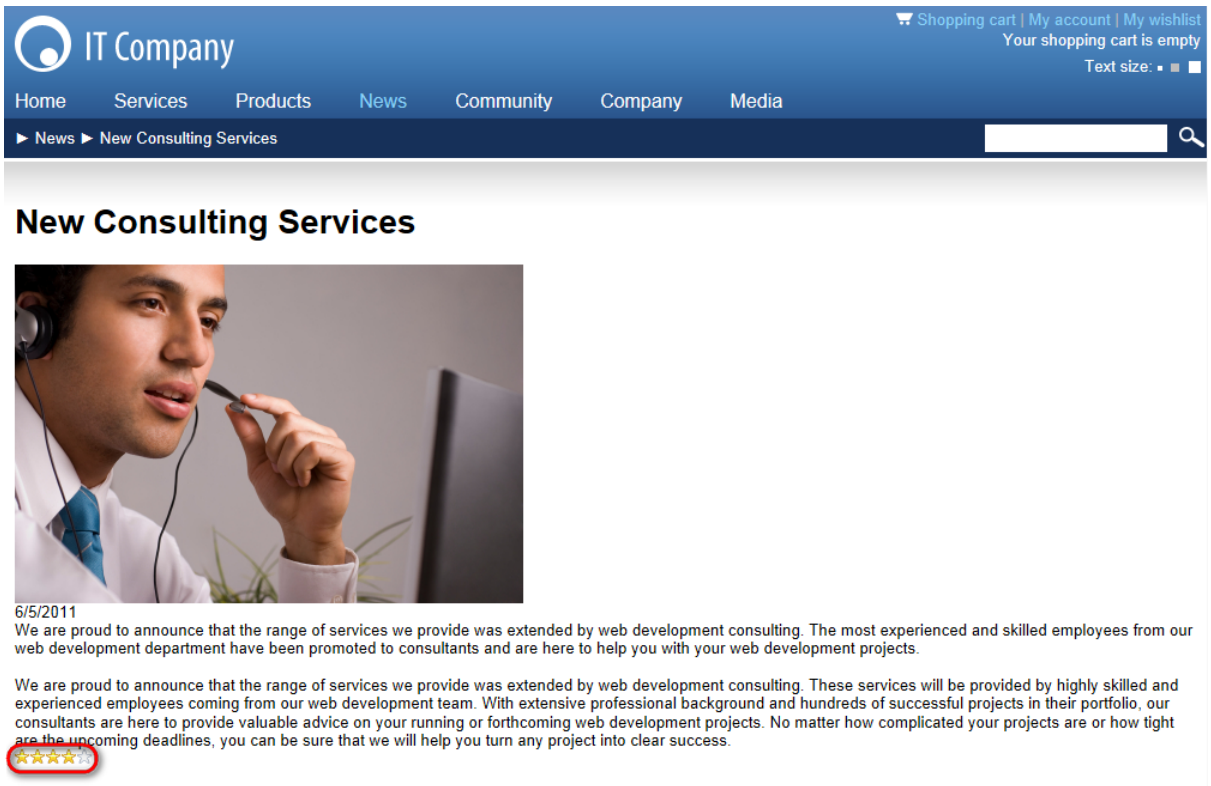
2. In the **Web part properties** dialog which pops up, click the **Edit** button next to the **Selected item transformation** property. Replace the original transformation with the following code, which is the original code with the highlighted parts added.

```
<%@ Register Src="~/CMSAdminControls/ContentRating/RatingControl.ascx"
TagName="RatingControl" TagPrefix="cms" %>

  <div class="newsItemDetail">
    <h1>
      <%# Eval("NewsTitle") %></h1>
    <div class="NewsSummary">
      <%# IfEmpty(Eval("NewsTeaser"), "", GetImage("NewsTeaser")) %>
      <div class="NewsContent">
        <div class="Date">
          <%# GetDateTime("NewsReleaseDate", "d") %></div>
        <div class="TextContent">
          <%# Eval("NewsSummary") %></div>
      </div>
      <div class="Clearer"> </div>
    </div>
```

```
    <div class="NewsBody">
      <div class="TextContent">
        <%# Eval("NewsText") %></div>

<cms:RatingControl ID="elemRating" runat="server" Enabled="true"
RatingType="Stars" ExternalValue='
      <%# Convert.ToString(CMS.GlobalHelper.ValidationHelper.GetDouble(Eval
("DocumentRatingValue"), 0)/((CMS.GlobalHelper.ValidationHelper.GetDouble(Eval
("DocumentRatings"), 0) == 0?1:CMS.GlobalHelper.ValidationHelper.GetDouble(Eval
("DocumentRatings"), 1)))) %>' />

    </div>
  </div>
```

Click **Save & close** to save the changes.

3. Now if you go to the live site, browse to the **News** section and display detail of some news item, you should see the rating control present below the news text, as can be seen in the screenshot below.

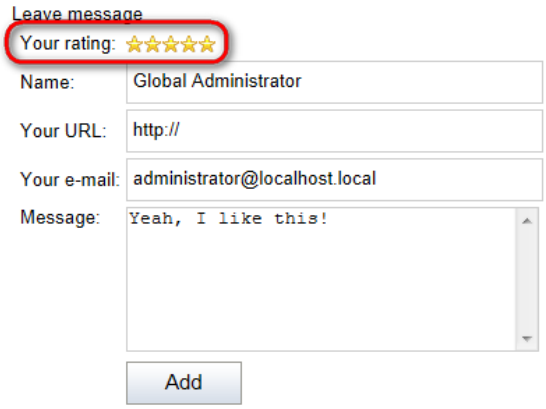


8.16.4 Integration with Message boards

Web parts of the Message boards module - the **Message board** and **Group message board list** web parts - have content rating features embedded. This enables live site users to submit a rating along with a comment posted to the message board. To enable content rating features of these web parts, you have to configure the following properties:

- **Enable content rating** - if checked, content rating features of the web part will be enabled.
- **Rating type** - appearance of the web part. *Stars*, *Radio-buttons* or *Drop-down list* can be chosen
- **Max rating value** - size of the rating scale; e.g. if 7 is entered, rating will be possible on a scale from 1 to 7

When leaving a board message, the rating control (determined by the **Rating type** property) will be displayed above the **Name** field. Users can select their rating and after sending the message, the rating will be applied.



8.16.5 Content rating internals and API

8.16.5.1 Content rating database tables and API classes

The Content rating module uses the following database table:

- **CMS_Document** - this table is used for storing information about documents in the content tree; its *DocumentRatingValue* column stores the total sum of all ratings of a particular document, while the *DocumentRatings* column stores the total number of ratings of the document

The Content rating API is provided by the following **CMS.DocumentEngine** namespace classes:

- **TreeProvider** - among other methods for document manipulation, this class provides methods for management of documents' ratings

The following topics show how methods from these classes can be used to manage abuse reports:

- Adding document rating
- Getting document rating
- Modifying document rating
- Resetting document rating

The API programming and Kentico CMS internals section of this guide contains more API related information, so please refer to it if required.

For detailed API documentation, such as a list of all methods from the classes above, please refer to Kentico CMS API Reference.

8.16.5.2 Adding document rating

The following code example explains how content rating can be added to a document using the API.

[C#]

```
using CMS.DocumentEngine;
using CMS.CMSHelper;

int nodeId = 241;
double rating = 0.5f;

// Gets the document
TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);
TreeNode node = tree.SelectSingleNode(nodeId, CMSContext.PreferredCultureCode);

// Adds the rating
// If the 3rd parameter is true, information that the document has been rated is
stored in a cookie. It is possible to check if the user rated the document.
TreeProvider.AddRating(node, rating, false);
```

8.16.5.3 Getting document rating

The following code example demonstrates how you can get rating of a document using the API.

[C#]

```
using CMS.DocumentEngine;
using CMS.CMSHelper;

int nodeId = 241;
double rating = 0.0f;

// Get the document
TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);
TreeNode node = tree.SelectSingleNode(nodeId, CMSContext.PreferredCultureCode);

// Get its rating
rating = node.DocumentRatingValue / node.DocumentRatings;
```

8.16.5.4 Modifying document rating

The following code example shows how rating of a document can be modified using the API.

[C#]

```
using CMS.DocumentEngine;
using CMS.CMSHelper;
```

```
int nodeId = 241;
double rating = 1.0f;
int numOfRatings = 2;

// Get the document
TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);
TreeNode node = tree.SelectSingleNode(nodeId, CMSContext.PreferredCultureCode);

// Modify its rating
TreeProvider.SetRating(node, rating, numOfRatings);
```

8.16.5.5 Resetting document rating

The following code example shows how to reset content rating of a document using the API.

[C#]

```
using CMS.DocumentEngine;
using CMS.CMSHelper;

int nodeId = 241;

// Get the document
TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);
TreeNode node = tree.SelectSingleNode(nodeId, CMSContext.PreferredCultureCode);

// Reset its rating
TreeProvider.ResetRating(node);
```

8.17 Custom tables

8.17.1 Overview

The Custom tables module allows you to create your own tables in the system database. You can manage data in custom tables without using Microsoft SQL Server Management Studio or any other database management tool. You will find this useful especially when you need to store a large number of items in a flat data structure. Standard Kentico documents are not as efficient as custom tables for storing items in a flat structure. Refer to the Storing data effectively topic for more information on the advantages of using custom tables over Documents.

You can also use custom tables when developing custom modules. Custom tables allow you to store the module data and access it using API.

The module comes with four web parts that you can use to display custom table data on a live site. To change the way the data in custom tables is being presented in the web parts, you can use transformations.

There is no form web part for acquiring data from live site visitors and storing them into custom tables availabe—custom tables aren't designed for this use. You can use Online forms for acquiring visitor information on the live site.

Finally, the module comes with an API, which enables you to handle custom table data in your own code.

The Custom tables module interacts with two other Kentico CMS modules. Data stored in the tables can be searched using the Smart search module. You can also create Alternative forms for the tables. These forms can be used instead of the default forms for creating or editing a custom table in the administration interface, as described in Modules -> Alternative forms -> Automatically used alternative forms.

Topics:

8.17.2 Creating custom tables

You can find the custom tables module at two locations of Kentico interface:

- **Site Manager -> Development -> Custom tables** allows you to create new custom tables, given you are the site's global administrator. You can also view and manipulate data in existing tables using this interface.
- **CMS Desk -> Tools -> Custom tables** allows you to view and manipulate data in existing custom tables only.

Note that you can create new custom tables using the wizard only. There is currently no simple way of creating custom tables using the API. This is due to the need to create related objects and settings—these tasks are handled by the wizard.

Creating a table using the New custom table wizard

In the following example, you will create a new custom table using the **New custom table wizard**.

Starting the New custom table wizard

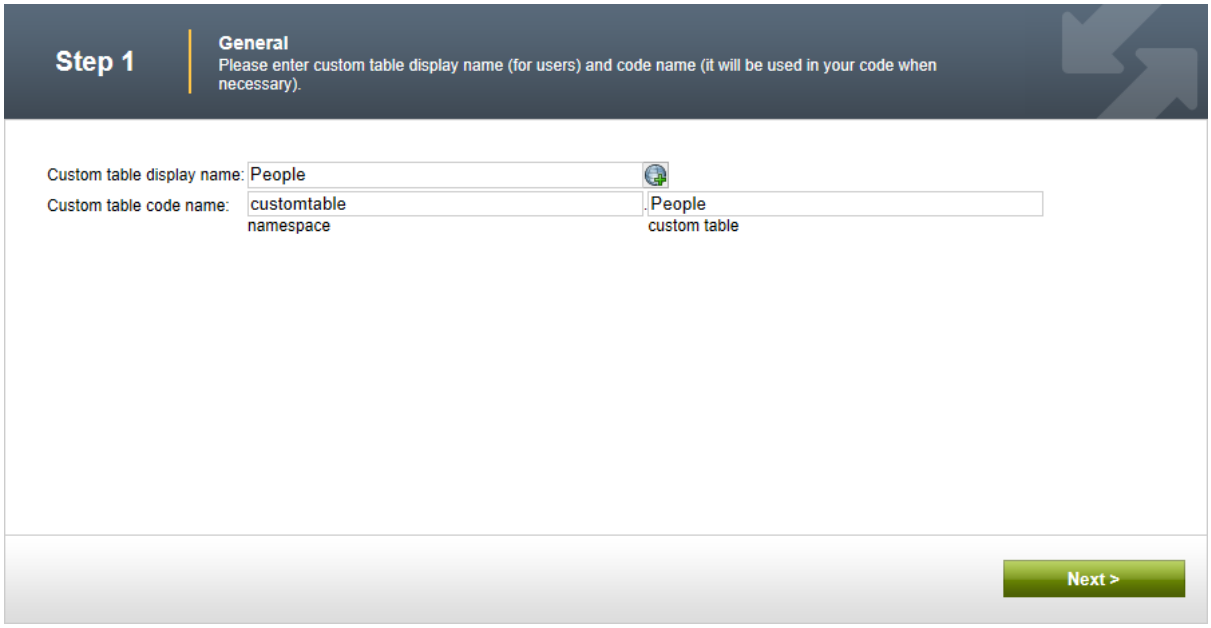
1. Navigate to **Site Manager -> Development -> Custom tables**.

2. Click **New custom table** (). The **New custom table wizard** starts.

Step 1

1. Enter the following details:

- **Custom table display name:** People
  - display name is used in Kentico CMS user interface.
- **Custom table code name:** customtable.People
  - display name is used in website code — always preceded by a namespace, which allows you to have different tables of the same name in different contexts.



2. Click **Next** to continue.

Step 2

In this step, you are given a choice from the two following options:

- **Create a new database table** - choose this option to create a brand new table in the system's database.
  - **Database table name** - the actual name of the table in the system database. A name in the *<namespace>_<code name>* format is pre-filled automatically.
  - **Primary key name** - primary key column name, pre-filled with the *ItemID* value.

- **Use an existing database table** - choose this option if you already have a physical table in the system database and want to register it in the system.
  - **Database table name** - the actual name of the table in the system database. The drop-down list offers only database tables which are not part of the default Kentico CMS database schema and are not yet registered for any object.
  - **Primary key name** - primary key column name. The original table's primary key column is used automatically.

The following check-boxes allow you to choose the default fields that you want to include in the custom table:

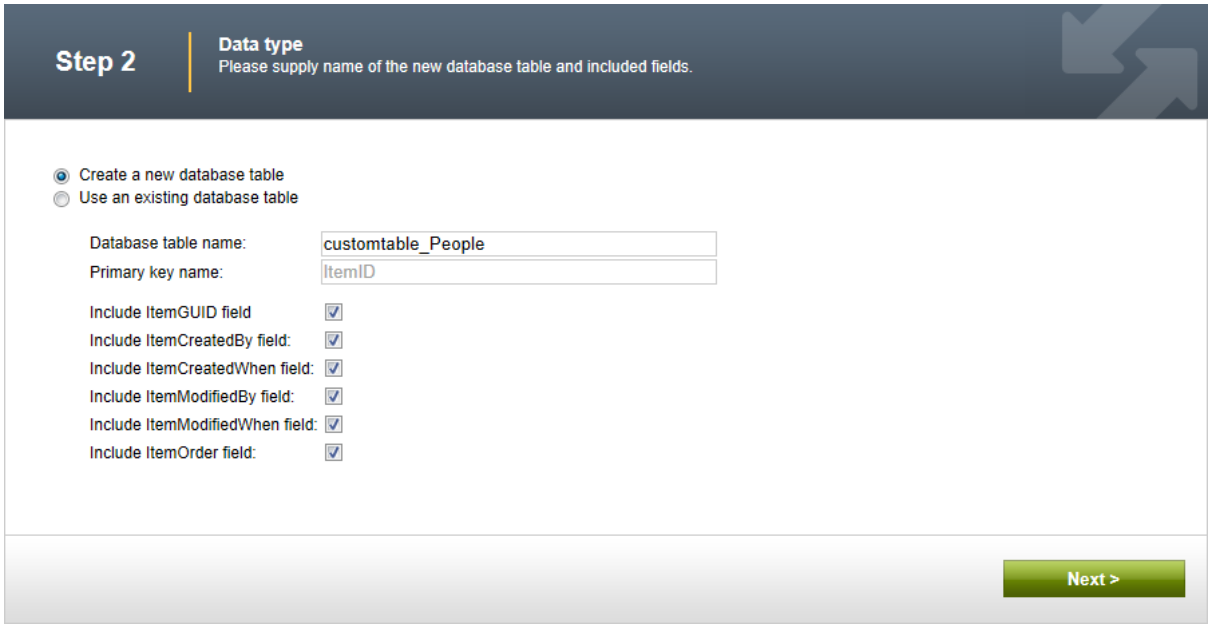
- **Include ItemGUID field** - globally unique ID of the particular custom table data record.
- **Include ItemCreatedBy field** - user name of the user who created the item.
- **Include ItemCreatedWhen field** - date and time of when the item was created.

- **Include ItemModifiedBy field** - user name of the user who last modified the item.
- **Include ItemModifiedWhen field** - date and time of last modification.
- **Include ItemOrder field** - order of the item when table content list is displayed. The lower number, the earlier position in the list.

1. Choose **Create a new database table**.

2. Leave the value *customtable_People* in the **Database table name field**.

3. Make sure all the check-boxes are turned on.



4. Click **Next** to continue.

Step 3

Third step contains the field editor. The field editor lets you define which columns will be included in the database table.

1. Create a field using the **New attribute** () button with the following properties:

- **Column name:** FirstName
- **Attribute type:** Text
- **Attribute size:** 100
- **Field caption:** First name
- **Form control:** Text box

2. Click the  **Save** button.

3. Create a second field using the **New attribute** () button with the following properties:

- **Column name:** LastName
- **Attribute type:** Text

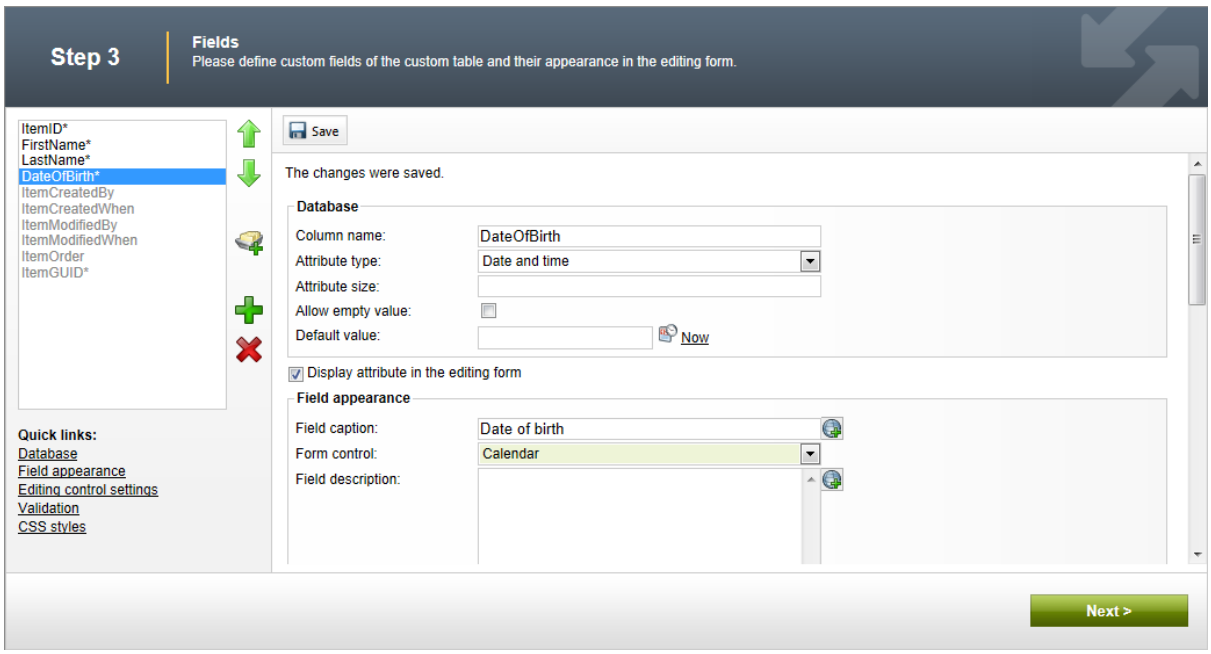
- **Attribute size:** 100
- **Field caption:** Last name
- **Form control:** Text box

4. Click the **Save** button.

5. Create a third field using the **New attribute** (+) button with the following properties:

- **Column name:** DateOfBirth
- **Attribute type:** Date and time
- **Field caption:** Date of birth
- **Form control:** Calendar
- **Editing control settings -> Edit time:** unchecked

6. Click the **Save** button.



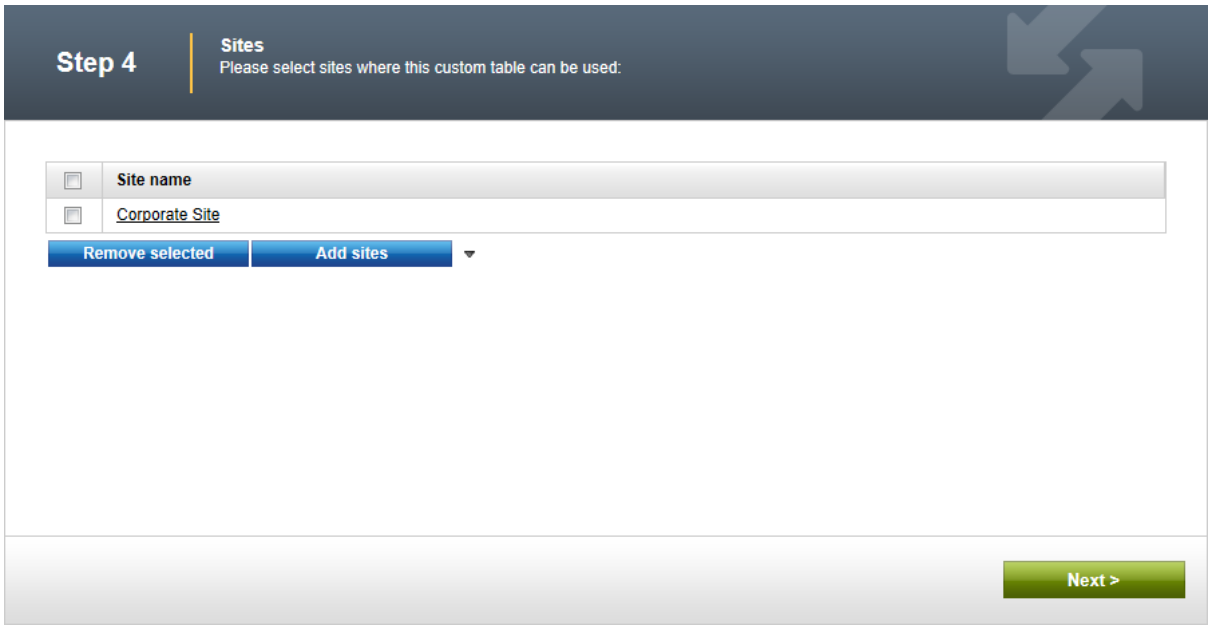
7. Once you are finished, click the **Next** button.

Step 4

Select on which sites the custom table will be available.

1. Click **Add sites**. A dialog box opens.

2. Choose **Corporate Site** (or any site that you want to add the custom table to) in the pop-up dialog.



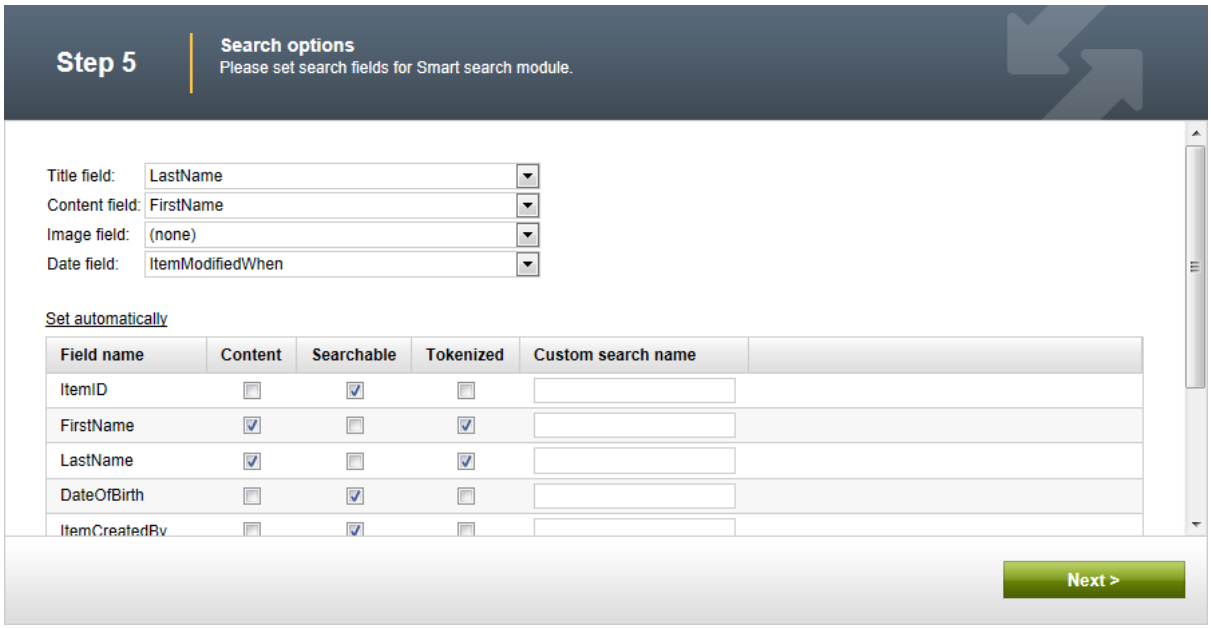
3. Click **Next** to proceed to the following step.

Step 5

Configure the indexing of data in the custom table by the Smart Search module. The four drop-down lists at the top determine which fields of the table will be displayed in search results.

1. Change the values to the following:

- **Title field**: LastName
- **Content field**: FirstName
- **Image field**: none
- **Date field**: ItemModifiedWhen

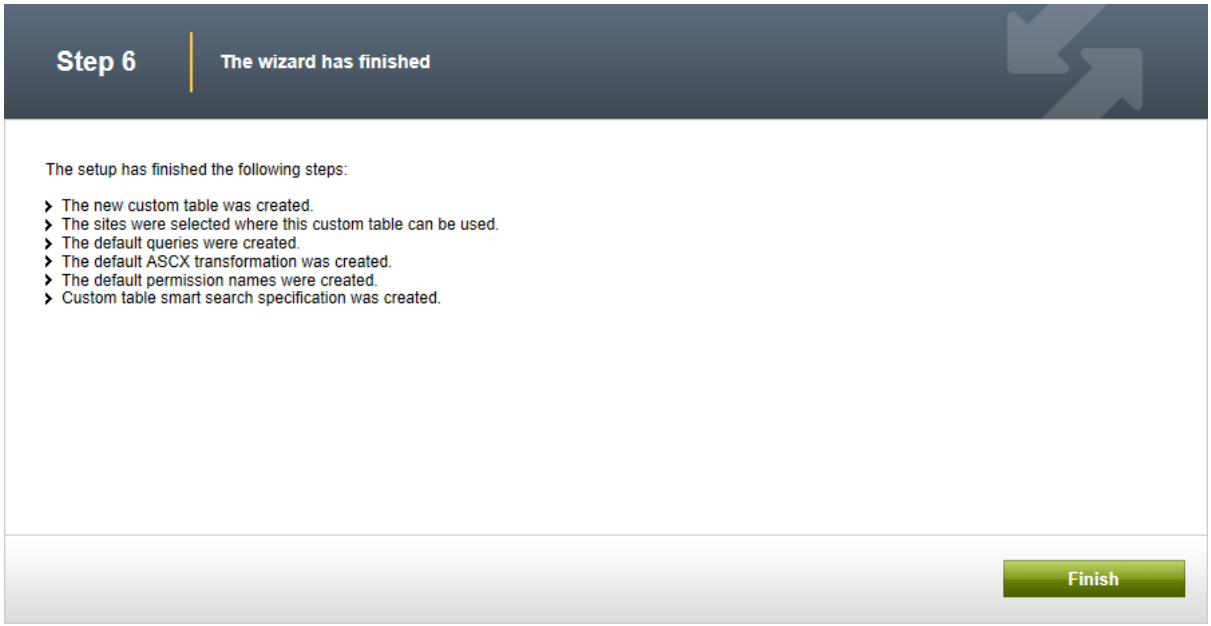


2. Leave the rest of the settings at default values and click **Next**.

For more information on settings in this step, please refer to Modules -> Smart search -> Object settings.

Step 6

The last step gives you an overview of the tasks executed by the wizard.



End the wizard by clicking **Finish**. You will be redirected back to the list of custom tables in **Site Manager -> Development -> Custom tables**.

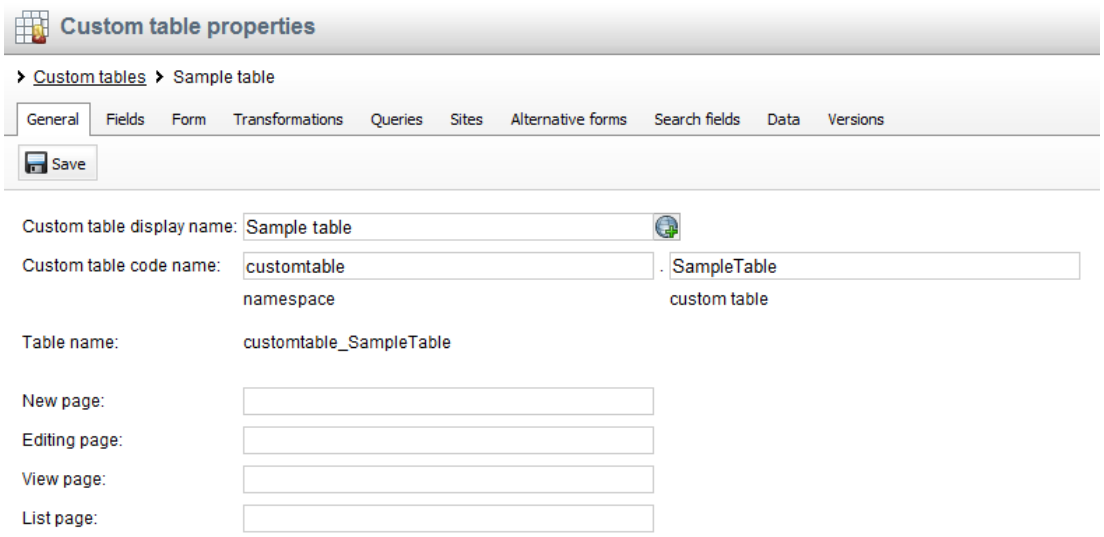
8.17.3 Editing custom tables

This topic covers editing existing custom tables. If you want to learn more about creating custom tables first, refer to the Creating custom tables topic.

Editing existing custom tables

1. Navigate to **Site Manager -> Development -> Custom tables**.

2. **Edit** () the custom table that you want to manage.



3. **Save** the changes you made.

The interface is divided into the following tabs:

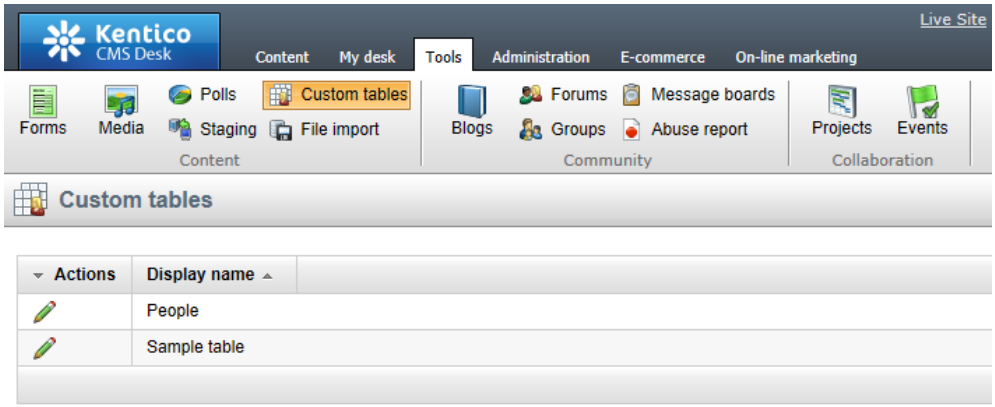
- **General** - this is where you can modify the display and code name of the custom table, as well as define URLs of custom pages that can be used instead of the default pages for adding, editing, viewing and listing of data items stored in the table
- **Fields** - on this tab, you can find the field editor that can be used to manage fields (columns) of the table
- **Form** - this tab allows you to create a custom form layout that will be used for adding and editing

data items
- **Transformations** - this is where transformations for the custom table can be created and managed
- **Queries** - database queries for manipulation with data in the table can be created and managed on this tab
- **Sites** - on this tab, you can define websites for which the custom table will be available
- **Alternative forms** - alternative forms for adding and editing custom table data can be created on this tab; more info can be found in Modules -> Alternative forms -> Automatically used alternative forms
- **Search fields** - on this tab, you can define how data stored in the custom table will be indexed by the Smart search module

You can find more information on particular tabs in the **Kentico CMS Context Help**. Access the context help by clicking the ? icon in the top right corner of each tab.

8.17.4 Managing data in custom tables

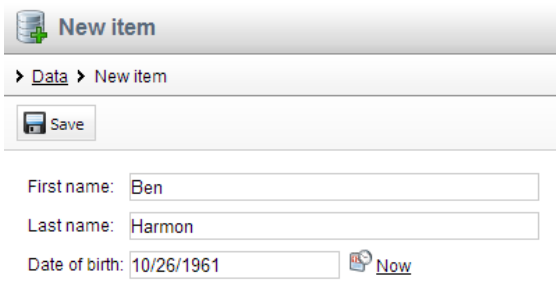
As a content editor, you can manage data stored in custom tables in **CMS Desk -> Tools -> Custom tables.** As an administrator, you can access the same interface through **Site Manager -> Development -> Custom tables -> edit (✎) -> Data** as well.



Adding data into custom tables

This example uses the **People** custom table that you have created if you followed the Creating custom tables chapter. The procedure is the same for any other custom table (except for the fields, which are always specific for a particular table).

1. Navigate to **CMS Desk -> Tools -> Custom tables**.
2. Click the **Edit** (✎) icon next to the **People** table.
3. Click the **New item** button.
4. Fill in any values.



5. Click **Save** to store the data into the custom table.

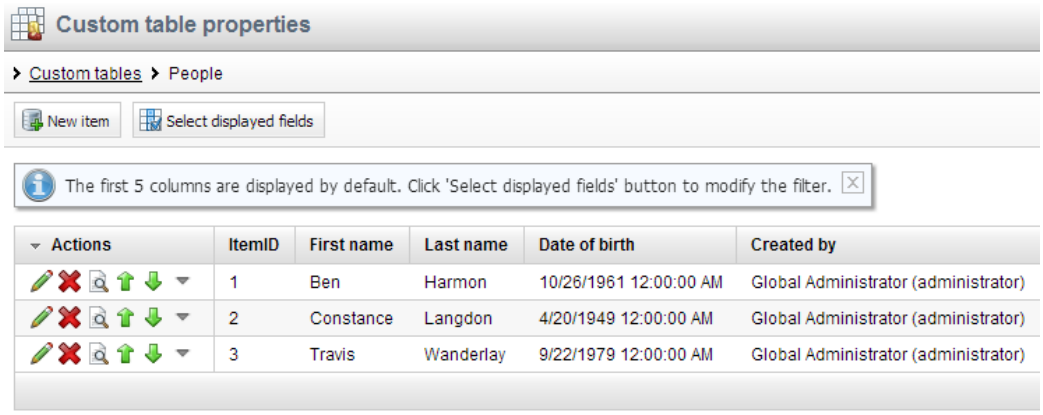
6. Use the **Create another** button and create at least two more records.

Viewing data in custom tables

This example uses the **People** custom table that you have created if you followed the Creating custom tables chapter. The procedure is the same for any other custom table (except for the fields, which are always specific for a particular table).

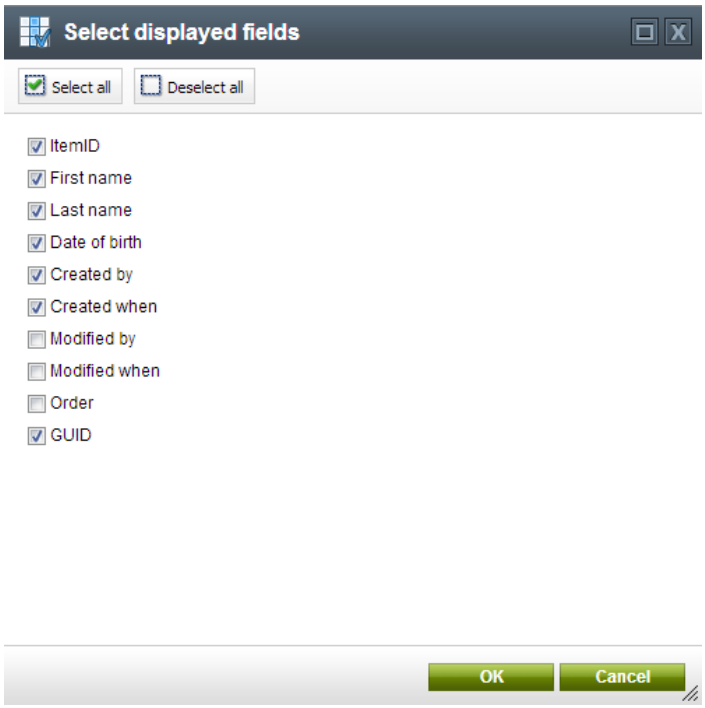
1. Navigate to **CMS Desk -> Tools -> Custom tables**.

2. **Edit** (✎) the **People** custom table. The system displays the list of entries in the custom table.

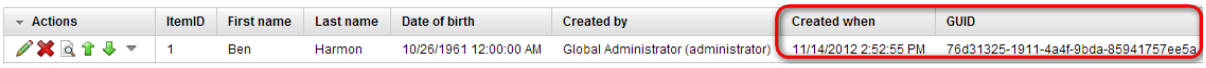


3. Click the **Select displayed fields** button. The system displays the **Select displayed fields** dialog box.

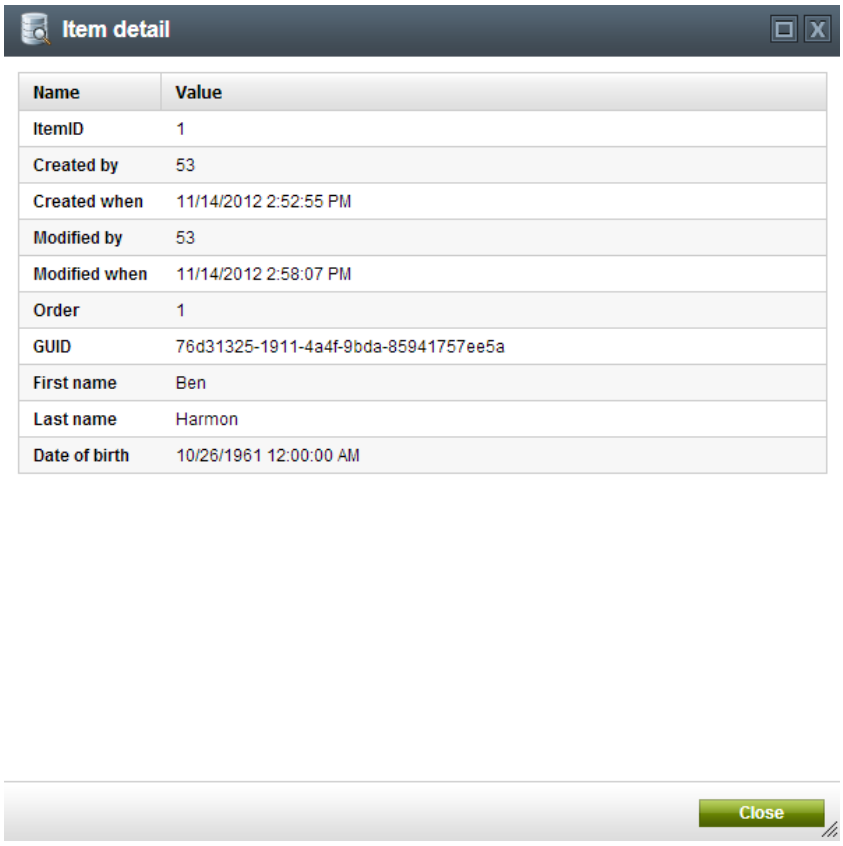
4. Turn the check-boxes on for the fields that you want to display.



5. Click **OK**. You can now see the modified view of the entries.



6. Click the **View** ( ) icon next to any of the data records. The system displays a dialog box with detailed information about the entry.



Note that you can also create custom filter for listing custom table data using alternative forms.

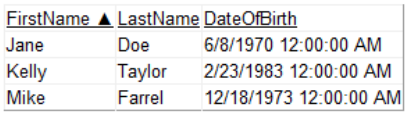
8.17.5 Displaying custom table data using web parts

The Custom tables module comes with three web parts that you can use for displaying custom table data. This page gives just a brief overview of these web parts. Detailed description of each web part and its properties can be found in Kentico CMS Web Parts Reference. Live site examples of these web parts can be found on the sample Corporate Site, under **Examples -> Webparts -> Custom tables**.

The custom table web parts that are available to you:

Custom table datagrid

This web part displays a grid with selected columns of data records from a custom table. You can modify the design of the web part using standard ASP.NET skins. If you want to handle the data by transformations, use one of the web parts below.



FirstName ▲	LastName	DateOfBirth
Jane	Doe	6/8/1970 12:00:00 AM
Kelly	Taylor	2/23/1983 12:00:00 AM
Mike	Farrel	12/18/1973 12:00:00 AM

Custom table datalist

In the following picture, you can see the basic appearance of the Custom table datalist web part. The

web part displays the custom table data in a multi-column/multi-row list. You can change the appearance using transformations. You can find more information about using transformations with custom tables web parts in the Transformations for custom tables topic.

First name:	Kelly	**First name:**	Mike
Last name:	Taylor	**Last name:**	Farrel
Date of birth:	2/23/1983 12:00:00 AM	**Date of birth:**	12/18/1973 12:00:00 AM

Custom table repeater

In the following picture, you can see the basic appearance of the Custom table repeater web part. You can change the appearance using transformations. You can find more information about using transformations with custom tables web parts in the Transformations for custom tables topic.

First name:	Kelly
Last name:	Taylor
Date of birth:	2/23/1983 12:00:00 AM

First name:	Mike
Last name:	Farrel
Date of birth:	12/18/1973 12:00:00 AM

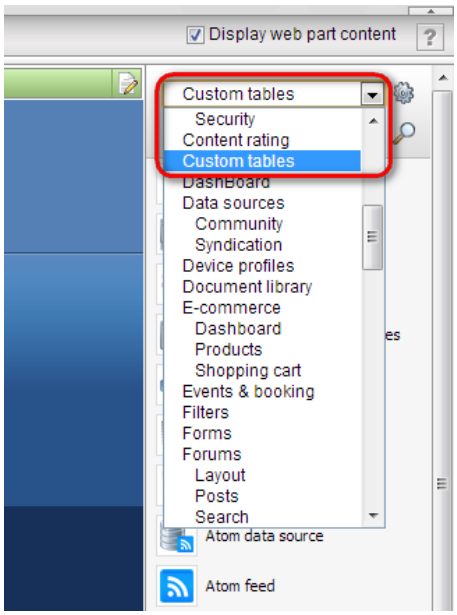
First name:	Jane
Last name:	Doe
Date of birth:	6/8/1970 12:00:00 AM

Custom table repeater with effect

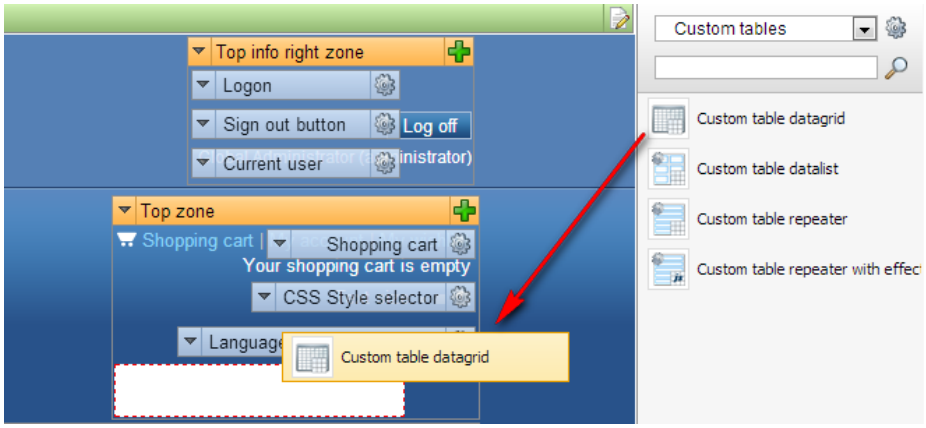
Renders data retrieved from a custom table you specify according to the assigned transformations and applies an additional javascript effect to the displayed records. You have to specify the effect through the web part's properties. You can find more information about using transformations with custom tables web parts in the Transformations for custom tables topic.

Placing the web parts using the web part toolbar

1. Navigate to **CMS Desk -> Content -> Edit**.
2. Choose the page on which you want to place the web part.
3. Switch to the **Design** tab.
4. In the **Web part toolbar**, select **Custom tables**.



5. Drag and drop the web part that you want to use to a **content area** of your choice.



The **Web part properties** dialog box opens.

Specifying the web part properties

1. Specify the following properties:

- **Web part control ID** - Enter a unique identifier for the web part.
- **Web part title -** Enter the title for the web part that is displayed on the **Design tab** and during **on-site editing**.
- **Custom table** - Select the custom table from which you want to display the data.

2. Learn more about the rest of the properties that you can specify for each of the web parts in Kentico CMS Web Parts Reference.

3. Click on **OK** to save the changes.

ItemID ▲	ItemCreatedBy	ItemCreatedWhen	ItemModifiedBy	ItemModifiedWhen	ItemOrder	FirstName	LastName	DateOfBirth
1	53	11/14/2012 2:52:55 PM	53	11/14/2012 2:58:07 PM	1	Ben	Harmon	10/26/1961 12:00:00 AM
2	53	11/15/2012 9:33:25 AM	53	11/15/2012 9:33:25 AM	2	Constance	Langdon	4/20/1949 12:00:00 AM
3	53	11/15/2012 9:34:27 AM	53	11/15/2012 9:34:27 AM	3	Travis	Wanderlay	9/22/1979 12:00:00 AM

Note that when on the **Design** tab, you can always drag and drop the web part elsewhere.

Placing the web parts using add web part button

1. Navigate to **CMS Desk -> Content -> Edit**.

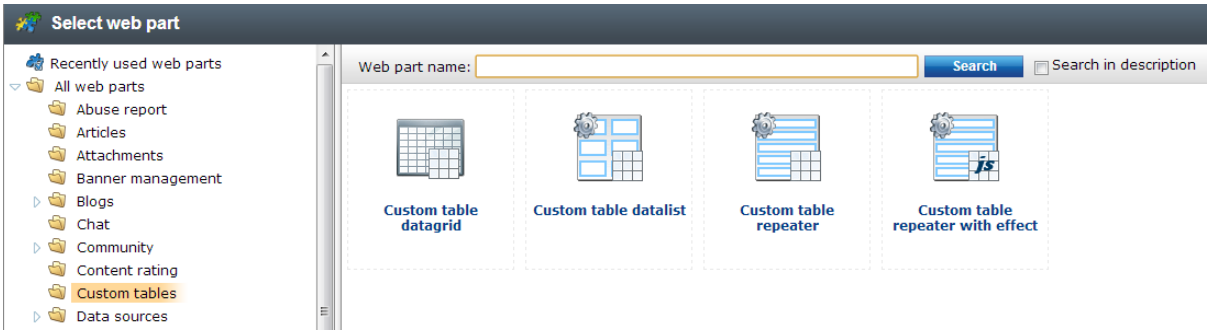
2. Choose the page on which you want to place the web part.

3. Switch to the **Design** tab.

4. Click the **Add web part** () button in the **content area** of your choice. The **Select web part** dialog box opens.

5. Select **Custom tables** in the left content tree.

6. Select the web part that you want to place.



7. Click on **OK**. The **Web part properties** dialog box opens.

Specifying the web part properties

1. Specify the following properties:

- **Web part title** - Enter a unique identifier for the web part.
- **Web part title -** Enter the title for the web part that is displayed on the **Design tab** and during **on-site editing**.
- **Custom table** - Select the custom table from which you want to display the data.

2. Learn more about the rest of the properties that you can specify for each of the web parts in Kentico CMS Web Parts Reference.

3. Click on **OK** to save the changes.

Note that when on the **Design** tab, you can always drag and drop the web part elsewhere.

8.17.6 Transforming custom tables

In the following example, you will learn you can change the way custom table data is displayed on your site by modifying default transformation code.

This example makes use of the Custom table repeater transformation on the sample Corporate Site. The **People** custom table that you have created if you followed the Creating custom tables topic and populated with data in the Managing data in custom tables topic is used throughout the example as well. You can use any other custom table as well.

Applying the transformation

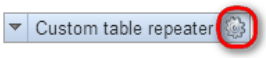
1. Navigate to **CMS Desk**.

2. Switch to **Edit** mode.

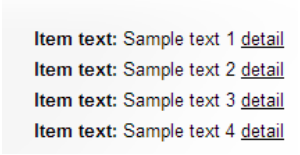
3. In the content tree, select **Examples -> Web Parts -> Custom tables -> Custom table repeater**.

4. Switch to the **Design** tab. The web part displays the content of the **Sample table** custom table by default.

5. **Configure** () the properties of the Custom table repeater web part.



Example of the web part:



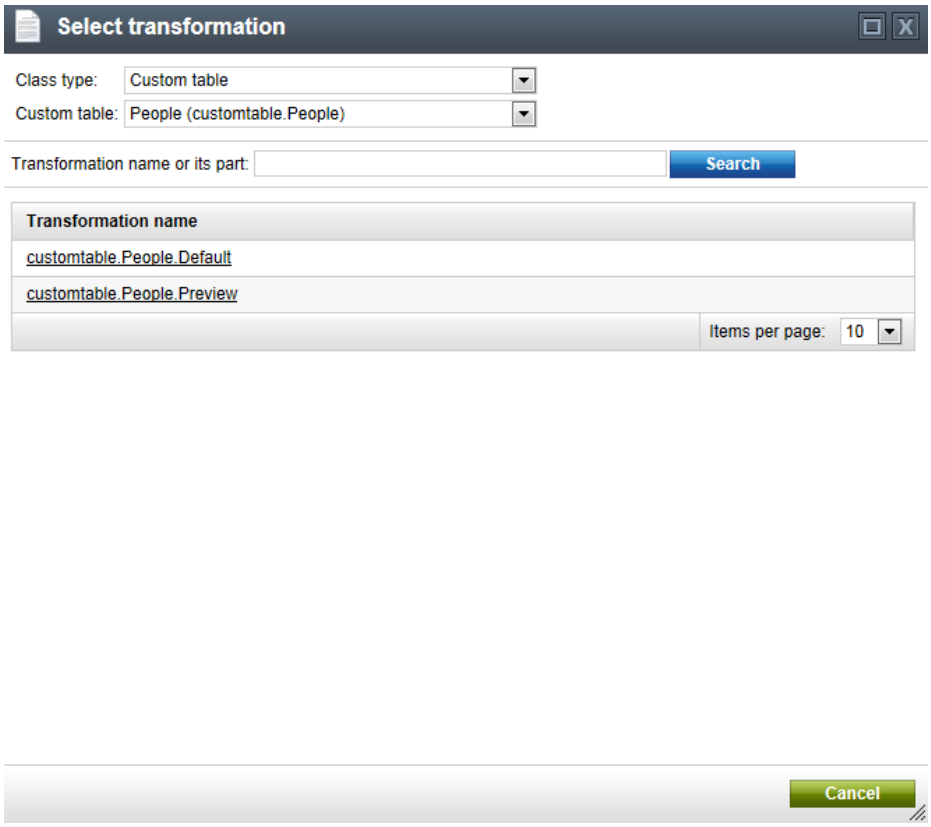
The **Web part properties** dialog box opens.

6. Under **Content**, in the **Custom table** drop-down list, select the **People** table.

7. Under **Transformations**, in the **Transformation name** property, click **Select**. The **Select transformation** dialog box opens.

8. Change the **Class type** to *Custom table* and select *People* as the **Custom table** value.

9. Click on **customtable.People.Default** to select the transformation.

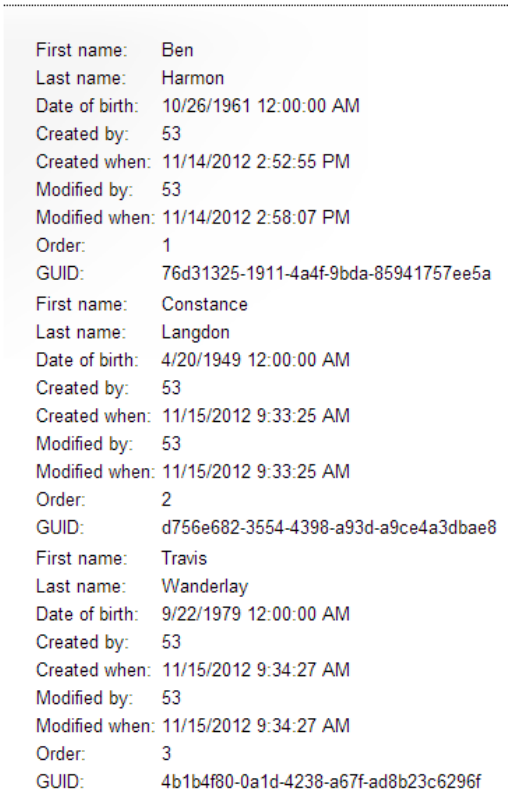


Click **OK** to confirm and close the **Web part properties** dialog.

Modifying the transformation

1. Switch to the **Live site** mode. You can see the repeater displaying the content of the **People** custom table.

Example of the web part:



On an actual website, you probably would not want to display all the system fields. To change this, you will need to edit the code of the used transformation.

2. Switch to the **Edit** mode.

3. Switch to the **Design** tab.

4. **Configure** () the repeater web part's properties again.

5. Under **Transformations**, in the **Transformation name** property, click **Edit**.

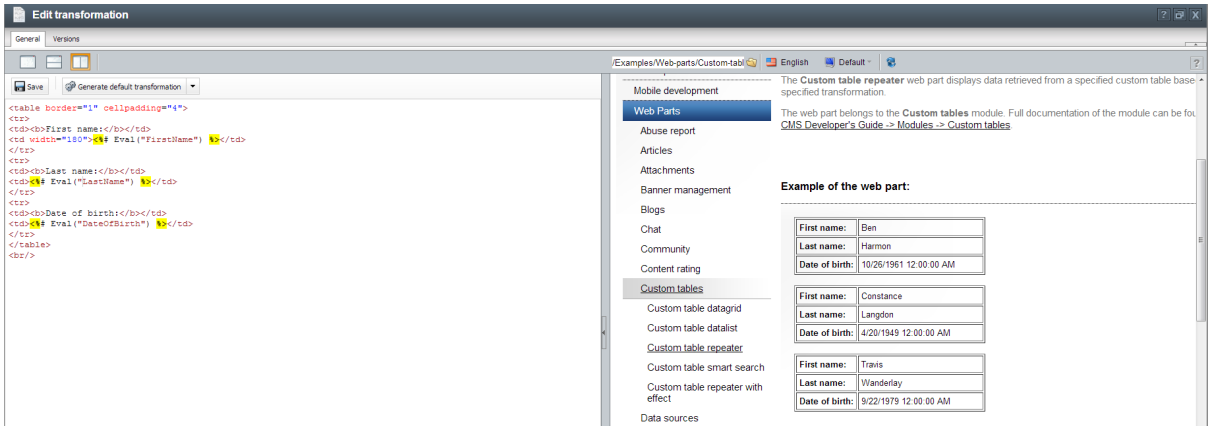
6. For the purpose of this tutorial, replace the entire transformation content with the following sample code. You can notice that it is just a modification of the former **Default** transformation, with the unwanted fields deleted and highlighted tags and properties added.

```
<table border="1" cellpadding="4">
<tr>
<td><b>First name:</b></td>
<td width="180"><%# Eval("FirstName") %></td>
</tr>
<tr>
<td><b>Last name:</b></td>
<td><%# Eval("LastName") %></td>
</tr>
```

```
<tr>
<td><b>Date of birth:</b></td>
<td><%# Eval("DateOfBirth") %></td>
</tr>
</table>
<br/>
```

7. **Save** the modifications confirm the change.

8. Click on **Preview**. The **Edit transformation** window opens. You can see a split view that allows you to see how the data is displayed by the current web part while editing the transformation code.



9. **Close** the dialog box.

10. Click on **OK** to confirm the changes.

11. Switch to the **Live site** to see the changes you made to the transformation.

8.17.7 Setting permissions

You can configure the permissions of the custom tables module on two levels — globally for all custom tables belonging to the current site and separately for each particular custom table.

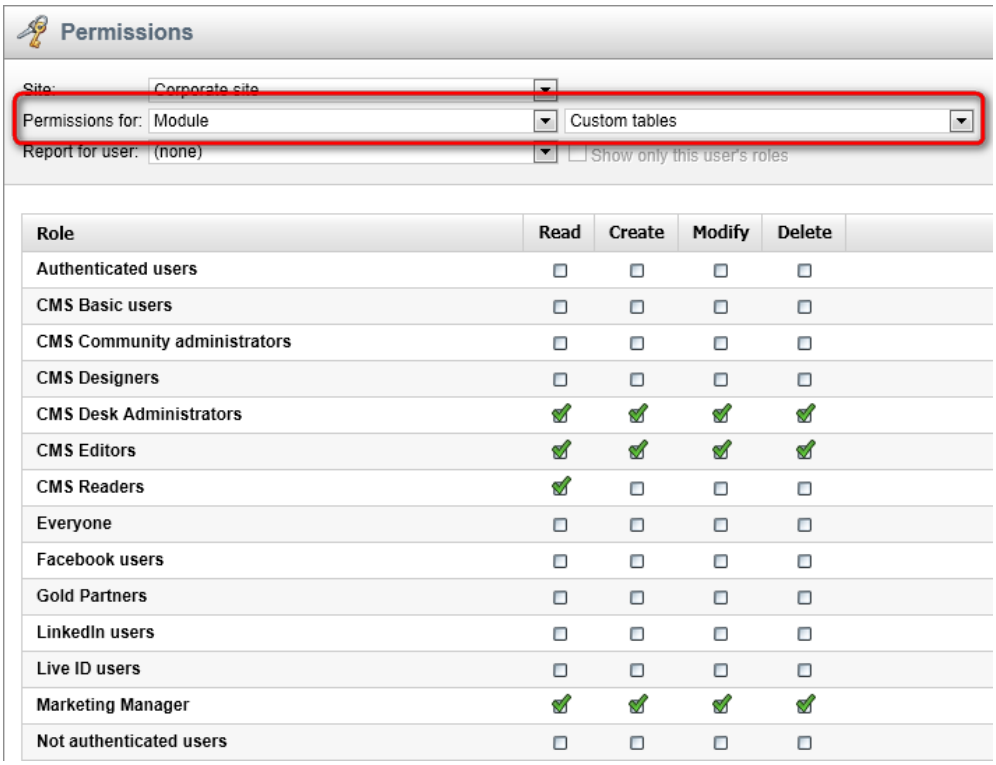
Configuring permissions for custom tables belonging to a certain site

1. Navigate to **Site Manager -> Administration -> Permissions.**

2. In the first **Permissions for** drop-down list choose **Module** and **Custom tables** in the second.

You can grant the following permissions on this level:

- **Create** - members of the role are allowed to create data in any custom table.
- **Delete** - members of the role are allowed to delete data in any custom table.
- **Modify** - members of the role are allowed to modify data in any custom table.
- **Read** - members of the role are allowed to read any custom table data.



Once you grant the permissions to the specific roles, you may want to assign users to the roles as described in the Role management topic.

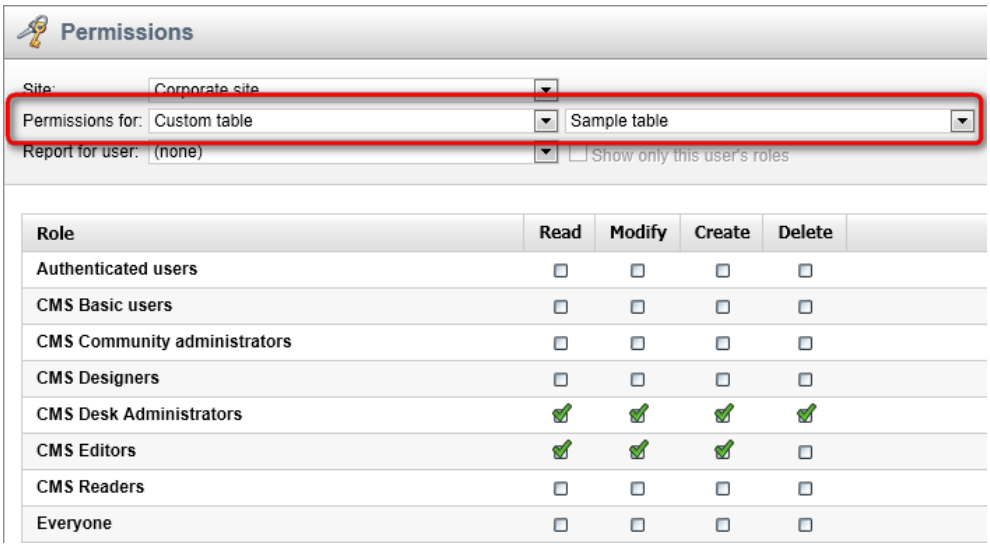
Configuring permissions for a particular custom table

1. Navigate to **Site Manager -> Administration -> Permissions.**

2. In the first **Permissions for** drop-down list choose **Custom table** and the specific custom table in the second.

You can grant the following permissions on this level:

- **Create** - members of the role are allowed to add new records into the selected table
- **Delete** - members of the role are allowed to delete records from the selected table
- **Modify** - members of the role are allowed to modify existing records in the selected table
- **Read** - members of the role are allowed to read data stored in the selected table



Once you grant the permissions to the specific roles, you may want to assign users to the roles as described in the Role management topic.

8.17.8 Custom tables internals and API

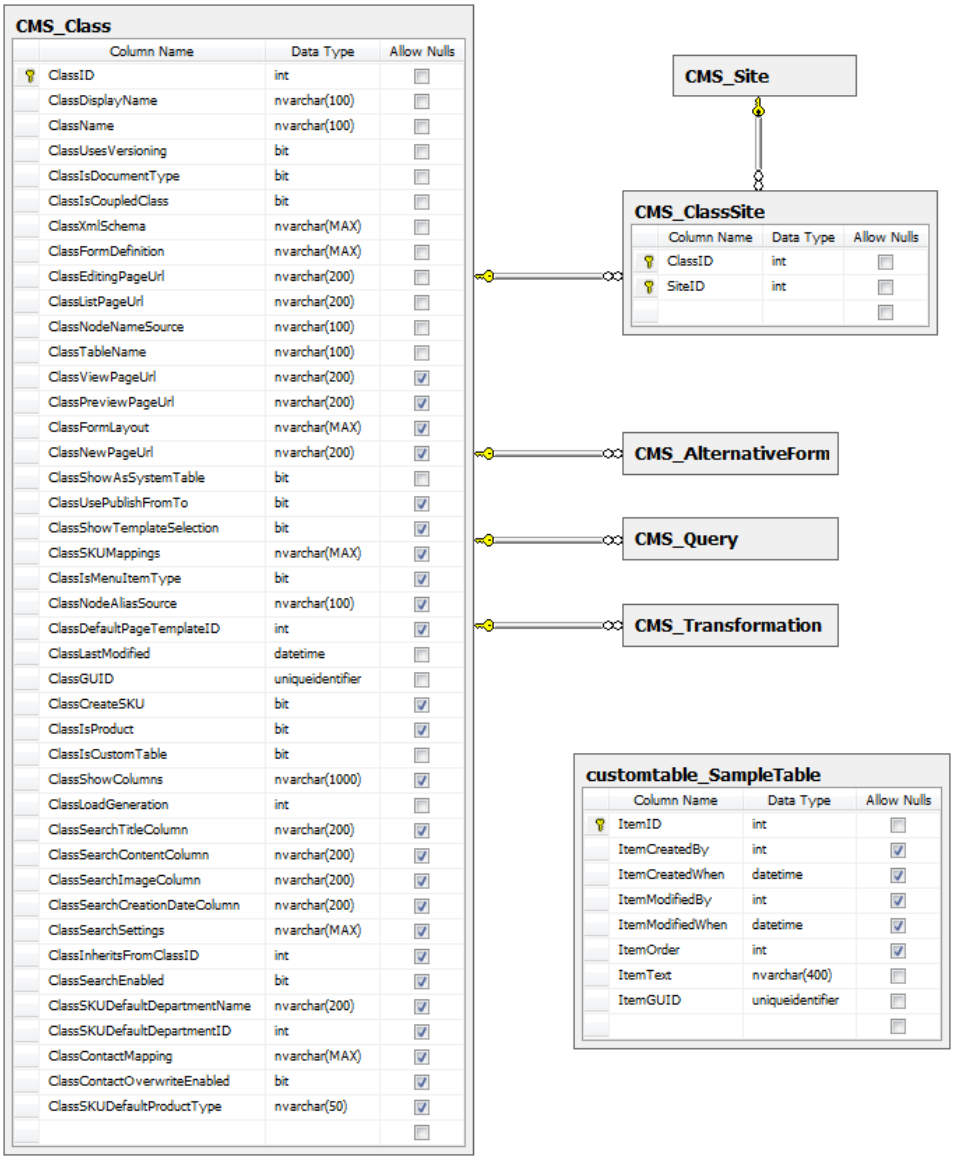
In this chapter, you will learn which database tables and API classes are used in the Custom tables module. You will also see the most common API examples.

Advanced API examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all methods from the module classes, please refer to Kentico CMS API Reference.

8.17.8.1 Database tables

The Custom tables module uses the following database tables:

- **CMS_Class** - custom tables are stored as classes in the system database, this database table is used for storing information about classes.
- **CMS_ClassSite** - this table is used to assign classes (and therefore also custom tables) to particular sites in the system.
- **CMS_Query** - contains records representing SQL queries for operations with data in custom tables, along with other queries used by the system.
- **CMS_Transformation** - contains records representing transformations for custom tables data, as well as other transformations used by the system.
- **CMS_AlternativeForm** - contains records representing alternative forms for particular custom tables, together with other alternative forms used by the system.
- **<prefix>_<table code name>** - particular custom tables are stored as standard tables in the system database. You can distinguish them by giving them a descriptive prefix, while the *customtable_* prefix is used by default.



8.17.8.2 API classes

If you are not familiar with database table data management through Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

> **Please note**
>
> The classes of the Custom tables module can be found in the **CMS.SiteProvider** namespace.

Custom tables API:

- **CustomTableItem** - represents one custom table object.
- **CustomTableItemProvider** - provides management functionality for custom tables.

8.17.8.3 API examples

These topics show examples of how the API of the Custom tables module can be used:

- Managing custom table data

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Development\CustomTables\Default.aspx.cs***.

The Custom tables API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.GlobalHelper;
using CMS.UIControls;
using CMS.CMSHelper;
using CMS.SiteProvider;
using CMS.SettingsProvider;
```

8.17.8.3.1 Managing custom table data

Use the following example to create a custom table item.

```
private bool CreateCustomTableItem()
{
    // Creates new Custom table item provider
    CustomTableItemProvider customTableProvider = new CustomTableItemProvider
(CMSContext.CurrentUser);

    // Prepares the parameters
    string customTableClassName = "customtable.sampletable";

    // Checks if Custom table 'Sample table' exists
    DataClassInfo customTable = DataClassInfoProvider.GetDataClass
(customTableClassName);
    if (customTable != null)
    {
        // Creates new custom table item
        CustomTableItem newCustomTableItem = CustomTableItem.New
```

```
(customTableClassName, customTableProvider);

        // Sets the ItemText field value
        newCustomTableItem.SetValue("ItemText", "New text");

        // Inserts the custom table item into database
        newCustomTableItem.Insert();

        return true;
    }

    return false;
}
```

Using the following example, you can get and update the custom table item that you created in the previous code example.

```
private bool GetAndUpdateCustomTableItem()
{
    // Creates a new Custom table item provider
    CustomTableItemProvider customTableProvider = new CustomTableItemProvider
(CMSContext.CurrentUser);

    string customTableClassName = "customtable.sampletable";

    // Checks if Custom table 'Sample table' exists
    DataClassInfo customTable = DataClassInfoProvider.GetDataClass
(customTableClassName);
    if (customTable != null)
    {
        // Prepares the parameters
        string where = "ItemText LIKE N'New text'";
        int topN = 1;
        string columns = "ItemID";

        // Gets the data set according to the parameters
        DataSet dataSet = customTableProvider.GetItems(customTableClassName,
where, null, topN, columns);

        if (!DataHelper.DataSourceIsEmpty(dataSet))
        {
            // Gets the custom table item ID
            int itemID = ValidationHelper.GetInteger(dataSet.Tables[0].Rows[0][0],
0);

            // Gets the custom table item
            CustomTableItem updateCustomTableItem = customTableProvider.GetItem
(itemID, customTableClassName);

            if (updateCustomTableItem != null)
            {
                string itemText = ValidationHelper.GetString
```

```
(updateCustomTableItem.GetValue("ItemText"), "");

                // Sets new values
                updateCustomTableItem.SetValue("ItemText", itemText.ToLowerCSafe
());

                // Saves the changes
                updateCustomTableItem.Update();

                return true;
            }
        }
    }

    return false;
}
```

Use this example to get a custom table item and move it **down** in the order of items in the custom table.

```
private bool GetAndMoveCustomTableItemDown()
{
    // Creates new Custom table item provider
    CustomTableItemProvider customTableProvider = new CustomTableItemProvider
(CMSContext.CurrentUser);

    string customTableClassName = "customtable.sampletable";

    // Checks if Custom table 'Sample table' exists
    DataClassInfo customTable = DataClassInfoProvider.GetDataClass
(customTableClassName);
    if (customTable != null)
    {
        // Prepares the parameters
        string where = "ItemText LIKE N'New text'";
        int topN = 1;
        string columns = "ItemID";

        // Gets the data set according to the parameters
        DataSet dataSet = customTableProvider.GetItems(customTableClassName,
where, null, topN, columns);

        if (!DataHelper.DataSourceIsEmpty(dataSet))
        {
            // Gets the custom table item ID
            int itemID = ValidationHelper.GetInteger(dataSet.Tables[0].Rows[0][0],
0);

            // Moves the item down
            customTableProvider.MoveItemDown(itemID, customTableClassName);

            return true;
```

```
        }
    }

    return false;
}
```

Use this example to get a custom table item and move it **up** in the order of items in the custom table.

```
private bool GetAndMoveCustomTableItemUp()
{
    // Creates new Custom table item provider
    CustomTableItemProvider customTableProvider = new CustomTableItemProvider
(CMSContext.CurrentUser);

    string customTableClassName = "customtable.sampletable";

    // Checks if Custom table 'Sample table' exists
    DataClassInfo customTable = DataClassInfoProvider.GetDataClass
(customTableClassName);
    if (customTable != null)
    {
        // Prepares the parameters
        string where = "ItemText LIKE N'New text'";
        int topN = 1;
        string columns = "ItemID";

        // Gets the data set according to the parameters
        DataSet dataSet = customTableProvider.GetItems(customTableClassName,
where, null, topN, columns);

        if (!DataHelper.DataSourceIsEmpty(dataSet))
        {
            // Gets the custom table item ID
            int itemID = ValidationHelper.GetInteger(dataSet.Tables[0].Rows[0][0],
0);

            // Moves the item up
            customTableProvider.MoveItemUp(itemID, customTableClassName);

            return true;
        }
    }

    return false;
}
```

Using the following example, you can get multiple custom table items based on a WHERE condition and then update t

```
private bool GetAndBulkUpdateCustomTableItems()
{
```

```
    // Creates new Custom table item provider
    CustomTableItemProvider customTableProvider = new CustomTableItemProvider
(CMSContext.CurrentUser);

    string customTableClassName = "customtable.sampletable";

    // Checks if Custom table 'Sample table' exists
    DataClassInfo customTable = DataClassInfoProvider.GetDataClass
(customTableClassName);
    if (customTable != null)
    {
        // Prepares the parameters

        string where = "ItemText LIKE N'New text%'";

        // Gets the data
        DataSet customTableItems = customTableProvider.GetItems
(customTableClassName, where, null);
        if (!DataHelper.DataSourceIsEmpty(customTableItems))
        {
            // Loops through the individual items
            foreach (DataRow customTableItemDr in customTableItems.Tables[0].Rows)
            {
                // Creates object from DataRow
                CustomTableItem modifyCustomTableItem = CustomTableItem.New
(customTableItemDr, customTableClassName);

                string itemText = ValidationHelper.GetString
(modifyCustomTableItem.GetValue("ItemText"), "");

                // Sets new values
                modifyCustomTableItem.SetValue("ItemText", itemText.ToUpper());

                // Saves the changes
                modifyCustomTableItem.Update();
            }

            return true;
        }
    }

    return false;
}
```

Use this example to delete the custom table item that you created in the first code example.

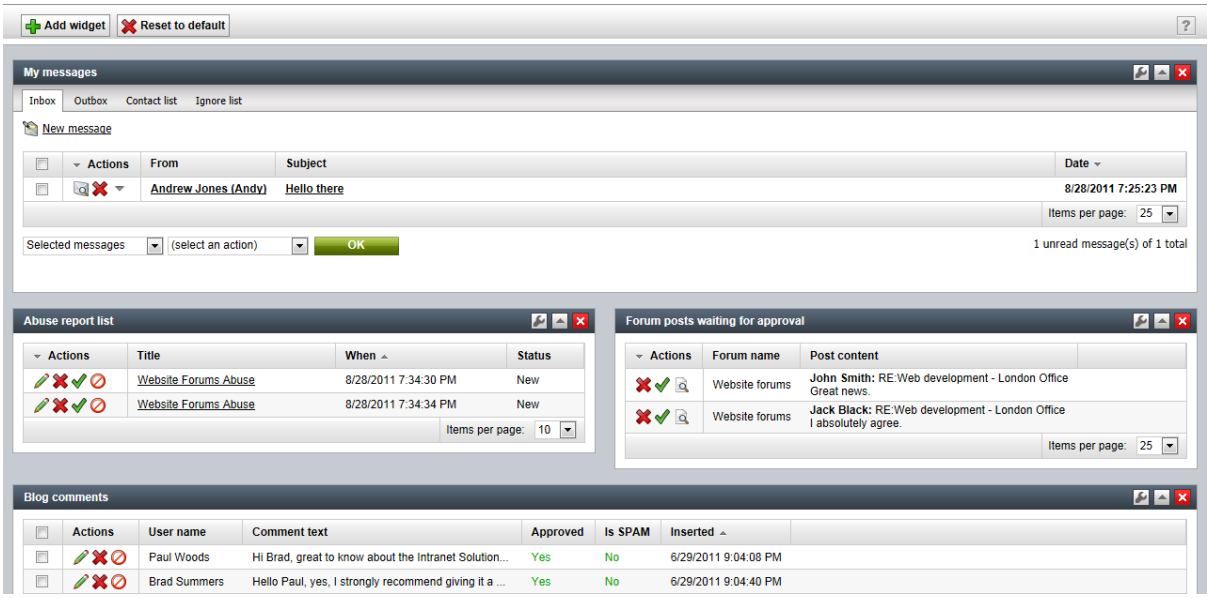
```
private bool DeleteCustomTableItem()
{
    // Creates new Custom table item provider
    CustomTableItemProvider customTableProvider = new CustomTableItemProvider
(CMSContext.CurrentUser);
```

```
    string customTableClassName = "customtable.sampletable";

    // Checks if Custom table 'Sample table' exists
    DataClassInfo customTable = DataClassInfoProvider.GetDataClass
(customTableClassName);
    if (customTable != null)
    {
        // Prepares the parameters
        string where = "ItemText LIKE N'New text%'";

        // Gets the data
        DataSet customTableItems = customTableProvider.GetItems
(customTableClassName, where, null);
        if (!DataHelper.DataSourceIsEmpty(customTableItems))
        {
            // Loops through the individual items
            foreach (DataRow customTableItemDr in customTableItems.Tables[0].Rows)
            {
                // Creates object from DataRow
                CustomTableItem deleteCustomTableItem = CustomTableItem.New
(customTableItemDr, customTableClassName);

                // Deletes custom table item from database
                deleteCustomTableItem.Delete();
            }

            return true;
        }
    }

    return false;
}
```

8.18 Dashboards

8.18.1 Overview

Dashboards are sections of the Kentico CMS administration interface that can be customized by individual users directly through the browser. Typically, a user will personalize their dashboard to contain frequently used tools or sources of information, which they can then easily access from a single location without the need to navigate through the interface.



The structure of every dashboard is based on a portal engine page template of a specific type, which designates the parts of the dashboard that are always present and cannot be changed by users. This also includes defining customizable areas (zones) and any default content. The components that can be placed on dashboards by users are called widgets. These are the same objects that allow the personalization of pages directly on the live site or on the **Page** tab of **CMS Desk**.

The personalizable content of a dashboard, meaning the widgets that are placed on it and their configuration, is distinct for every user. If the dashboard is located in the **CMS Desk** interface, its widget content is also unique for every different site.

Detailed information about defining dashboard page templates and using widgets on dashboards may be found in the Managing dashboard content topic.

By default, dashboards can be found in the following sections of the interface:

- **CMS Site Manager -> Dashboard**
- **CMS Desk -> My Desk -> My Dashboard**
- **CMS Desk -> Tools -> Web analytics -> Dashboard**

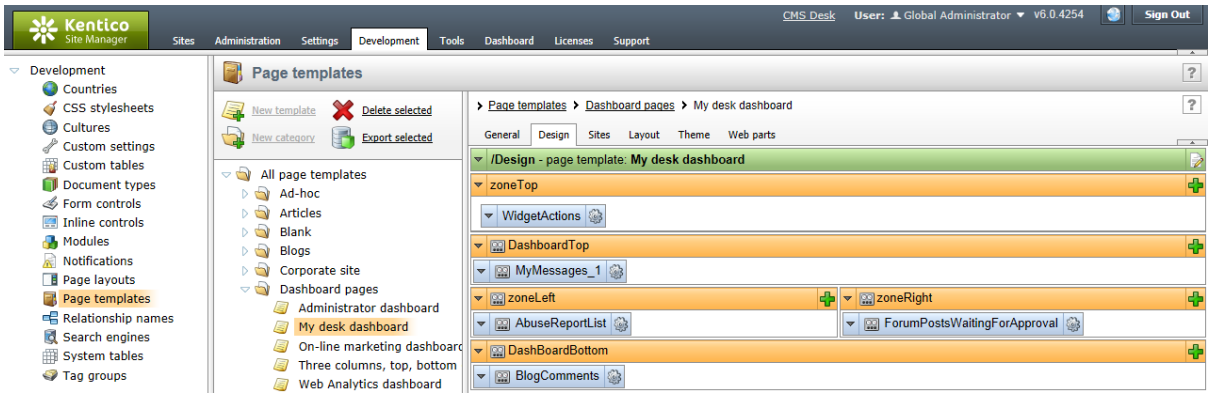
If required, a dashboard page may be created at a custom location according to the procedure described in the Adding a new dashboard to the interface topic.

8.18.2 Managing dashboard content

There are several levels on which dashboard content can be specified.

Setting the layout of zones, fixed content and default widgets

The general design and initial content of a dashboard is determined by the page template that it uses. Dashboard templates can be managed at **Site Manager -> Development -> Page templates**, like any other type of page templates. A dashboard template must have its **Template type** property set to *Dashboard page* on the **General** tab. Configuration of the actual template content can be performed on the **Design** tab.



The placement, size and amount of zones on the template may be altered by editing the code of its page layout via the **Edit layout** ( ) button or on the **Layout** tab.

There are two possible types of zones that can be defined on a dashboard template:

- **Web part zones** - the content of zones of this type may only be managed here on the *Design* tab of the page template. Web parts located here function as fixed content on the dashboard page itself. The *Widget zone type* property of these zones must be set to *None*.
- **Dashboard widget zones** - zones of this type may only contain widgets. Any widgets added here on the *Design* tab serve as the default content of the zone. Individual users may change and configure the widget content of the zone for their own personalized version of the dashboard. The *Widget zone type* property of these zones must be set to *Dashboard*.

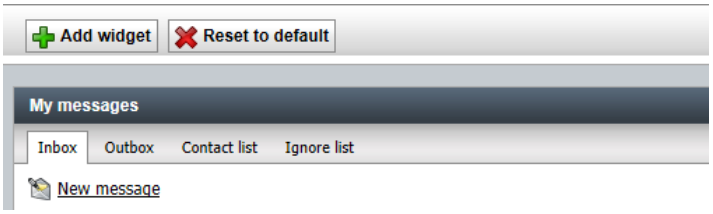
Every dashboard template should contain the Widget actions web part somewhere in its layout in order to allow users to add new widgets or reset the zones of the dashboard to their default widget content. The web part must also be configured ( ) to have its **Widget zone type** property set to *Dashboard*. The **Widget zone ID** property may be used to specify the dashboard zone to which new widgets will be added (the ID of the desired zone must be entered). Once created, users can drag the widget to another zone as required.

The settings made on the **Sites** tab of a dashboard page template do not affect on which websites dashboards with this template can be placed. They are only used to ensure that the page template will be exported/imported along with the assigned sites.

To learn about how a page template is assigned to a specific dashboard section, please read the Adding a new dashboard to the interface topic.

Using widgets on dashboards

Users of the administration interface with access to a dashboard section can manipulate widgets in the available zones and thus create their own personalized version of the dashboard. Widgets are added using the **Add widget** button generated by the **Widget actions** web part. At any time, all the zones on the dashboard and the widgets they contain can be returned to their default state by clicking **Reset to default**.



Individual widgets are enclosed in containers with a header and a set of action buttons. To manage the widgets on the dashboard, the appropriate actions must be used.

The simplest action available is provided by the **Minimize widget** ( ) button, which hides the content of a widget so that only its header is visible. The **Maximize widget** ( ) action can be used to restore the widget to its previous state.

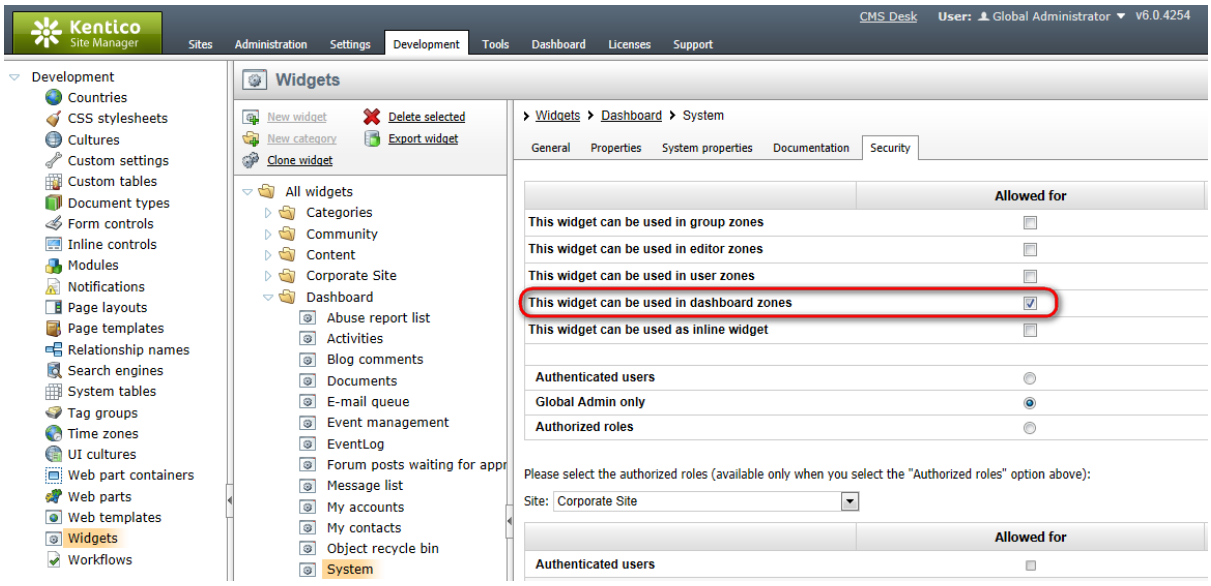
Clicking the **Configure widget** ( ) button opens the **Widget properties** dialog where the widget's properties can be modified. Every widget has its own set of properties depending on its function (detailed information can be found in the Widgets reference guide). The **Widget title** property is present for all types of widgets, as it sets the text displayed in the header of the widget on the dashboard page.

The location of a widget can be changed easily by using its drag-and-drop functionality. To pick up a widget, simply hover over its header, hold down the mouse button and then drag it to the desired position. This works both for modifying the order of widgets within a zone and moving them to a different zone.

The **Remove widget** ( ) action deletes the widget from the dashboard completely.

Configuring widgets for dashboards

In order for a specific widget to be usable as a dashboard component, it must first be configured in **Site Manager -> Development -> Widgets**, specifically by checking the **This widget can be used in dashboard zones** option on its **Security** tab.



If a widget is also allowed in other types of zones, for example user zones on the live site, it may in certain cases be useful to set some of its properties to be available only when configuring the widget on a dashboard. This can be achieved on the **Properties** tab using the **Display attribute only in the dashboard** checkbox included among the settings of the given property.

8.18.3 Adding a new dashboard to the interface

If the dashboards built into the administration interface by default do not meet your specific requirements, it is possible to integrate new dashboard pages.

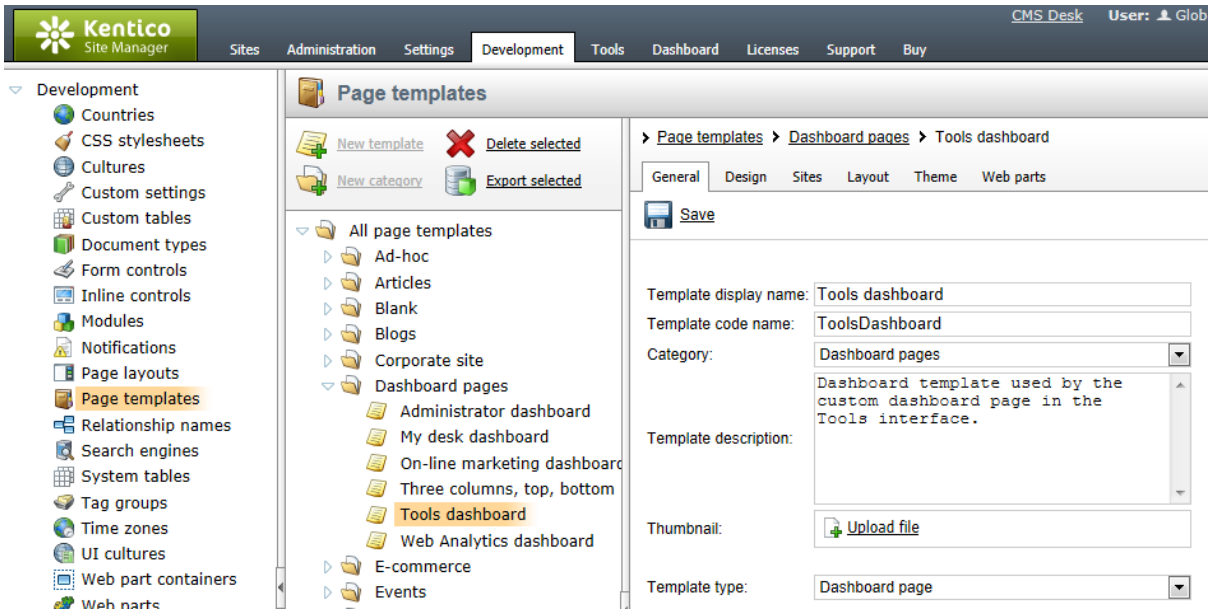
The following example demonstrates how a custom dashboard can be added to the **Tools** tab of the **CMS Desk** administration interface. The same principles can be applied when creating a dashboard in other locations.

Creating the dashboard page template

1. First, create the page template that the dashboard page will be based on. Go to **CMS Site Manager -> Development -> Page templates**, select the **Dashboard pages** category from the tree and click **New template**. Enter the following values into the available fields:

- **Template display name**: Tools dashboard
- **Template code name**: ToolsDashboard

Click **OK**, select *Dashboard page* from the **Template type** drop-down list of the new template and **Save** the change.



2. Switch to the **Design** tab and click the **Edit layout** ( ) button. Copy the following code into the layout to define some web part/widget zones for the template:

```
<table border="0" width="100%" cellspacing="0" cellpadding="0">
  <tr>
    <td colspan="2">
        <div class="DashboardActions PageTitleHeader">
           <cc1:CMSWebPartZone ID="zoneTop" runat="server" />
        </div>
    </td>
  </tr>
  <tr>
    <td colspan="2">
      <cc1:CMSWebPartZone ID="DashboardTop" runat="server" />
    </td>
  </tr>
  <tr valign="top">
    <td style="width:50%">
      <cc1:CMSWebPartZone ID="DashboardLeft" runat="server" />
    </td>
    <td style="width:50%">
      <cc1:CMSWebPartZone ID="DashboardRight" runat="server" />
    </td>
  </tr>
</table>
```

Click **Save**.

3. Now open the **Design** tab again, expand the menu ( ) of the **DashboardTop** zone and select the **Properties** item. Switch the **Widget zone type** property from *None* to *Dashboard* and click **OK**. Repeat this step for the **DashboardLeft** and **DashboardRight** zones.

4. Next, click the **Add web part** ( ) button of the **ZoneTop** zone, select **Widgets -> Widget actions** from the web part catalog and press **OK**. Configure the following properties of the web part:

- **Widgets -> Widget zone type**: Dashboard
- **Widgets -> Widget zone ID**: Leave empty; designates the zone where new widgets should be created when the *Add widget* button is clicked. By default, the first available zone will be used (*DashboardTop* in this case), but the ID of any other dashboard zone can be specified if required.

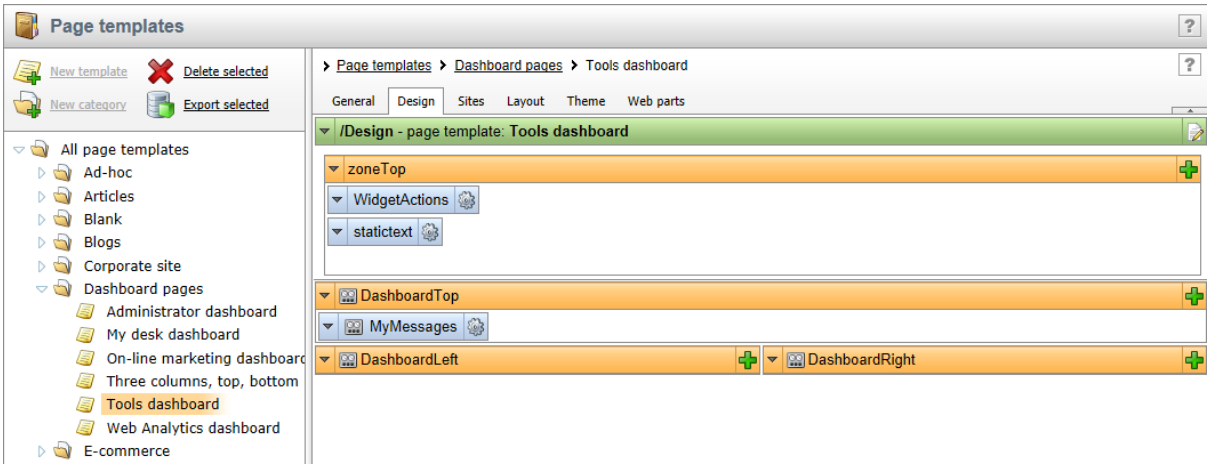
Leave the remaining properties in their default state and click **OK**.

5. Add another web part to the same zone. This time select **Text & Images -> Static text** and set the following properties:

- **Content -> Text**: This is a custom dashboard page.
- **HTML Envelope -> Content before**: <h2>
- **HTML Envelope -> Content after**: </h2>

Click **OK**.

6. Place a widget into the **DashboardTop** zone, for example **Community -> My messages** and leave the remaining two zones empty. This sets the default content of the dashboard that individual users can later configure and expand.



7. Switch to the **Sites** tab and assign the template to any websites that should carry over the new dashboard page when they are exported.

Adding the necessary UI element

8. The **CMS Desk -> Tools** section where the new dashboard will be located utilizes *User interface elements* that can be managed via the CMS. Navigate to **Site Manager -> Development -> Modules** and edit ( ) the **Tools** module. On the **General** tab, notice that the **Module code name** is *CMS.Tools*. This will be used later in the example.

9. Switch to the **User interface** tab, add a **New element** under the root of the Tools tree and enter the values shown below:

- **Display name**: Dashboards
- **Code name**: Tools.Dashboards
- **Element is custom**: Yes (checked)

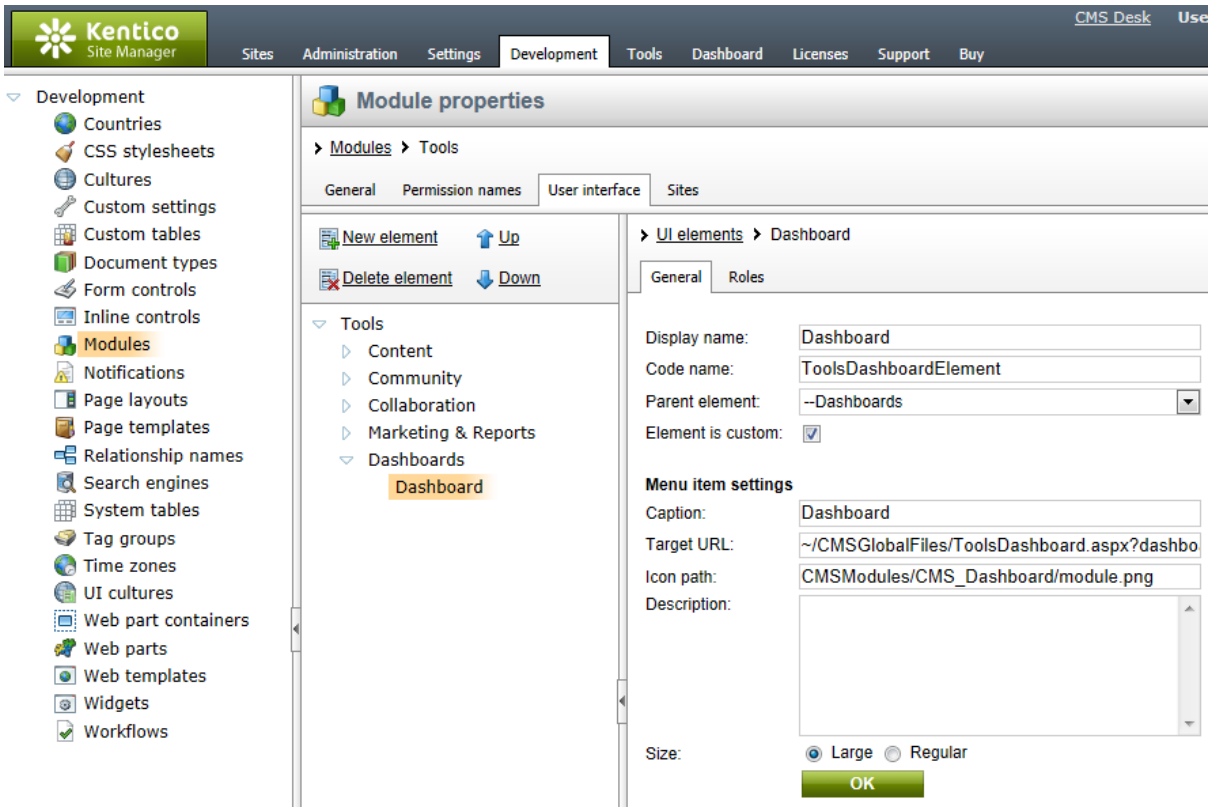
- **Caption**: Dashboards

Click **OK**. This element will only serve as a parent category in the tools menu, under which the actual dashboard will be placed. Now click **New element** again to add an item under the **Dashboards** element and fill in the form according to the following information:

- **Display name**: Dashboard
- **Code name**: ToolsDashboardElement
- **Element is custom**: Yes (checked)

- **Caption**: *Dashboard*; this sets the text caption displayed for the element in the actual interface. In real-world scenarios, it may be preferable to specify the caption using a localization string in the appropriate macro expression format: *{$<string key>$}*
- **Target URL**: *~/CMSGlobalFiles/ToolsDashboard.aspx?dashboardName=Tools&templateName=ToolsDashboard&{hash}*
  Sets the URL of the page with the content of the UI element. The actual source file used in the URL above will be created later in this example. When creating links to dashboard pages, it is necessary to understand and correctly specify the query string parameters:
  - **dashboardName** - sets a name for the dashboard to ensure uniqueness in cases where multiple dashboards use the same page template. The content of a dashboard is unique for every user and site (when located in CMS Desk). However, if two or more dashboards share a page template and the *dashboardName* parameters in the URLs used to access them have the same value, changes made to one of the dashboards will also be applied to the others (for the given user and site).
  - **templateName** - sets the code name of the page template that the dashboard will be based on. The type of the assigned template must be *Dashboard page*. As you can see, this example uses the template created in the previous steps.
  - **hash** - the hash code will be generated automatically by the used menu control.
- **Icon path**: *CMSModules/CMS_Dashboard/module.png*; sets the path to the image used to represent the element in the actual interface.
- **Size**: Large

Click **OK** to create the new UI element once all the fields are filled in correctly.



Adding dashboards without using UI elements

Not all sections of the administration interface can be modified via UI elements (please refer to UI personalization -> How does it work for more information). If you wish to add a dashboard page to these other locations, the code of the used navigation components must be modified manually. Keep in mind that the dashboard page needs to be accessed through a URL with the query string parameters mentioned above in the description of the **Target URL** property.

Creating the dashboard page's source file

10. Now the **.aspx** file of the dashboard page needs to be developed and added to your web project. Open your website in Visual Studio and create a **New folder** under the root called *CMSGlobalFiles* (if it doesn't already exist). Right-click the folder and select **Add New Item**. Choose to create a **Web Form** and name it *ToolsDashboard.aspx*. As you can see, this is the file specified in the **Target URL** of the UI element created before. This location ensures that the file will be exported along with any site that includes global folders in its export package.

11. Modify the page code to match the following:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="ToolsDashboard.aspx.cs"
Theme="Default" EnableEventValidation="false"
Inherits="CMSGlobalFiles_ToolsDashboard" %>
```

```
<%@ Register Src="~/CMSModules/Widgets/Controls/Dashboard.ascx"
TagName="Dashboard" TagPrefix="cms" %>

<%=DocType%>

<html xmlns="http://www.w3.org/1999/xhtml" <%=XmlNamespace%>>
<head id="Head1" runat="server" enableviewstate="false">
    <title id="Title1" runat="server">Dashboard</title>
    <asp:Literal runat="server" ID="ltlTags" EnableViewState="false" />
    <style type="text/css">
        body
        {
            margin: 0px;
            padding: 0px;
            height: 100%;
            font-family: Arial;
            font-size: small;
        }
    </style>
</head>
<body class="<%=BodyClass%>">
    <form id="form1" runat="server">

      <cms:Dashboard ID="ucDashboard" runat="server" ShortID="d" />

    </form>
</body>
</html>
```

The **Dashboard** user control that you registered and placed onto the form handles the entire functionality of the dashboard. It processes the query string parameters from the URL used to access the page and displays the corresponding dashboard according to the specified dashboard name, page template and context-related data such as the current site and user.

12. Switch to the code behind of the web form and add the following references to the beginning of the code:

[C#]

```
using CMS.UIControls;
using CMS.CMSHelper;
```

13. Next, set the **CMSGlobalFiles_ToolsDashboard** class to inherit from the **DashboardPage** class and modify it to contain the following code:

[C#]

```
public partial class CMSGlobalFiles_ToolsDashboard : DashboardPage
{
    protected override void OnPreInit(EventArgs e)
    {
```

```
        base.OnPreInit(e);

        // Must be equal to the code name of the module containing the
corresponding UI element
        ucDashboard.ResourceName = "CMS.Tools";

        // Must be equal to the code name of the corresponding UI element
        ucDashboard.ElementName = "ToolsDashboardElement";

        ucDashboard.PortalPageInstance = this as PortalPage;
        ucDashboard.TagsLiteral = this.ltlTags;
        ucDashboard.DashboardSiteName = CMSContext.CurrentSiteName;

        ucDashboard.SetupDashboard();
    }

    protected void Page_Load(object sender, EventArgs e)
    {

        // Security access and UI personalization checks for the current user

    }
}
```

In the handler of the *PreInit* event, certain properties of the **Dashboard** user control are assigned and its **SetupDashboard()** method is called. Note that the values of the **ResourceName** and **ElementName** properties must be set according to the module and code name of the UI element created in previous steps of this example.

When adding a dashboard to the interface of an actual live website, the **Page_Load** method should contain code checking that the current user has the *Read* permission for the given part of the interface and that the UI element is not disabled via UI personalization settings.

14. Save both files. If your CMS project was installed as a web application, **Build** your web project.

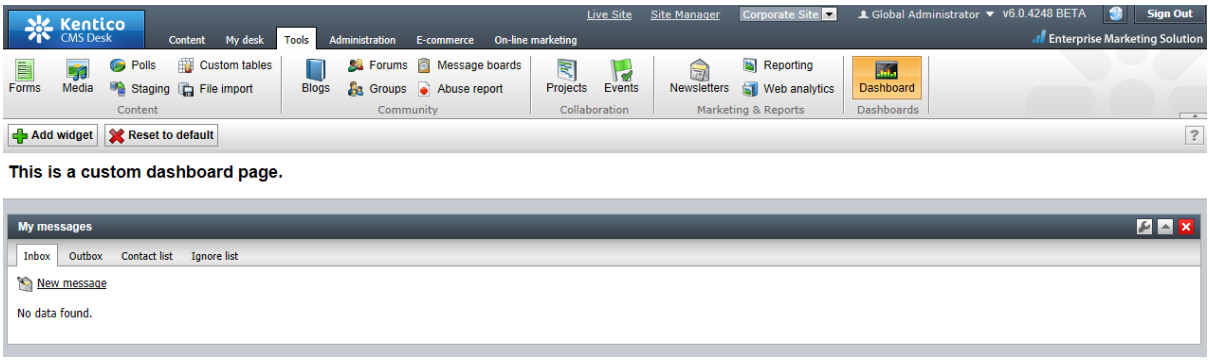
Creating dashboards in the Site Manager interface

It is not necessary to create a new aspx file for dashboards located in the Site Manager administration interface. You can simply reuse the default **~/CMSSiteManager/Dashboard.aspx** file, since there is no matching UI element to be configured and the security checks should always be the same.

When adding the link to the dashboard in Site Manager navigation components, remember to appropriately set the **dashboardName** and **templateName** query string parameters in the URL.

Result

15. If you now open **CMS Desk** and switch to the **Tools** tab, you will see that there is a new **Dashboard** item in the menu. If you select it, a fully functional dashboard page based on the created page template will be displayed.



8.19 Document library

8.19.1 Overview

The Document library module enables live site users to upload and manage files stored in a pre-defined location in the content tree. The module has no dedicated user interface. Its whole functionality is provided by two web parts: **Document library** and **Group document library**. Both web parts and their specific properties are described in the Document library web parts topic. The module also comes with the Document library widget, which is based on the **Document library** web part.

Document library files are stored in the content tree as standard *CMS.File* documents. This means that even though the module has no dedicated user interface, actions made on the live site are reflected in the content tree. Like this, the uploaded documents can also be managed by website editors or administrators in **CMS Desk -> Content**. Only flat file structure of document libraries is currently supported, i.e. you can't store files in sub-folders but only under the main document. More information about *CMS.File* documents, as well as other ways how files can be stored in Kentico CMS, can be found in Content management -> File management.

A live example of use of this module can be found on the **Intranet Portal** sample website, specifically on its */Documents* page. The process of creating a similarly structured document library can be found in the Creating a document library topic. The Managing files in document libraries topic explains how the module can be used from the live site user's point of view. Finally, the Security topic explains how standard document-level permissions are applied to documents in the library and how they can be configured.

8.19.2 Document library web parts

The whole functionality of the Document library module is ensured by the following two web parts:

- **Document library -> Document library** - designed for general use anywhere on your website; it displays all documents from a specified location except those whose *Owner* is a community group
- **Community -> Group document library** - designed for use within a context of a community group; displays documents whose *Owner* is the specified community group along with other documents stored under the specified path

The web parts have the following specific properties:

- **Library path** - path to the document under which the library documents are stored, e.g. */My-document-library/My-images*; only *CMS.File* documents from the first level under the entered path are displayed; using the **Set permissions** button, you can directly configure the document's permissions, as described in Security
- **Page size** - number of documents listed by the web part on a single page
- **Document form** - Alternative form that will be used for document properties editing (after choosing **Properties** from the context menu); if blank, the *CMS.File.DocumentLibrary* form will be used; if this form is not available, the *CMS.File* document type's default form will be used
- **Group name** - available only with the *Group document library* web part; documents owned by the specified group stored under the *Library path* will be displayed along with other documents under the same path
- **Combine with default culture** - if *No*, the web part only displays documents from the current culture that are stored under the *Library path*. If *Yes*, documents from the default culture are displayed along with documents from the current culture. In this case, the default culture documents are marked with the culture's flag and the **Translate** option is available in their context menu
- **Check permissions** - if enabled, only documents for which the current user has the **Read** permission are displayed by the web part; if disabled, all documents are displayed

8.19.3 Document library widget

The module also comes with the **Document library** widget. The widget is stored under the **Document library** widget category and it is an editor widget, i.e. it can only be added to editor widget zones of page templates by website editors in CMS Desk.

Compared to the Document library web parts, the **Document library** widget only has a limited set of properties to configure:

- **Library path** - path to the document under which the library documents are stored, e.g. */My-document-library/My-images*; only *CMS.File* documents from the first level under the entered path are displayed; using the **Set permissions** button, you can directly configure the document's permissions, as described in Security
- **Page size** - number of documents listed by the web part on a single page
- **Combine with default culture** - if *No*, only documents from the current culture that are stored under the *Library path* will be displayed; if *Yes*, documents from the default culture will be displayed

along with documents from the current culture, while default culture documents will be marked with the culture's flag and the **Translate** option is available in their context menu

Besides the fact that only a limited set of properties is available with the widget, its functionality is identical to the **Document library** web part.

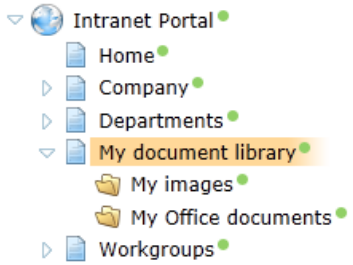
8.19.4 Creating a document library

This topic explains how a document library can be created on your website. We will create a sample page with two **Document library** web parts, each of which will display files from a separate folder. Like this, we will actually have two document libraries on a single page, each for storing different types of files. The example is based on the **Intranet Portal** sample website. However, the procedure is analogical when performed on any other website.

The first step when creating a document library is to choose a location in the content tree where the files will be stored. A single document library web part can display documents from only a single location. As we want two separate libraries on a single page, we will create a page in the content tree where two **Document library** web parts will be placed. Under the page, we will create two separate folders where the actual files will be stored.

1. Go to **CMS Desk -> Content**, select the root of the content tree and click **New**. Choose the **Page (menu item)** document type. In the following dialog, enter *My document library* into the **Page name** field and choose the **Create a blank page** option. Click **Save**.

With the new document selected in the content tree, click **New** again. This time, choose the **Folder** document type and enter *My images* as its **Document name**. Repeat the same once again to create another folder, while this one should be called *My Office documents*. The final result should look as in the screenshot below.



2. Now that you have the page and folders created, we may proceed to adding the **Document library** web parts. Select the **My document library** document, switch to the **Design** mode and click the **Add web part** (✚) icon of the only web part zone on the page. In the **Select web part** pop-up dialog,

choose the **Document library** web part, which is stored in the **Document library** web part category. Configure its properties as follows:

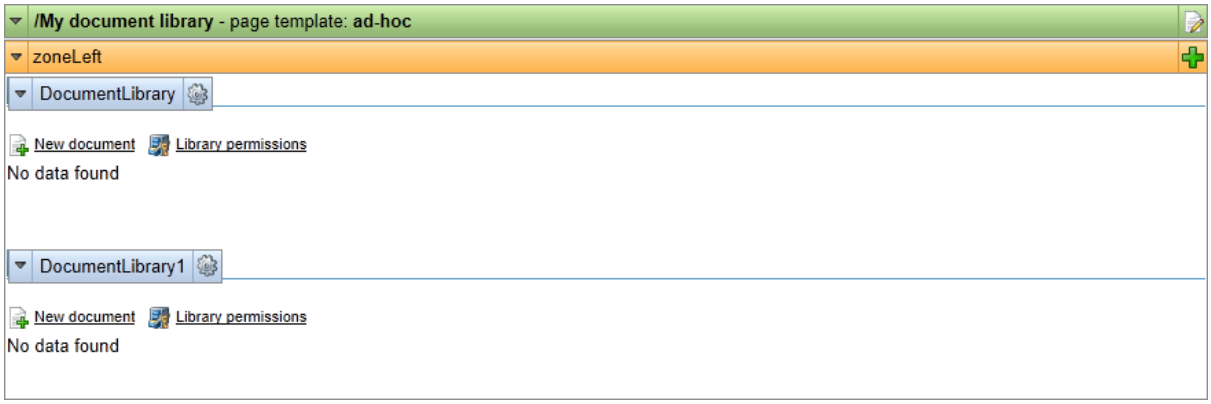
- **Library path**: /My-document-library/My-images
- **Page size**: 15
- **Document form**: leave the field blank
- **Web part container**: Intranet simple container (with header line)
- **Container title**: My images
- **Content after**:

Click **OK**.

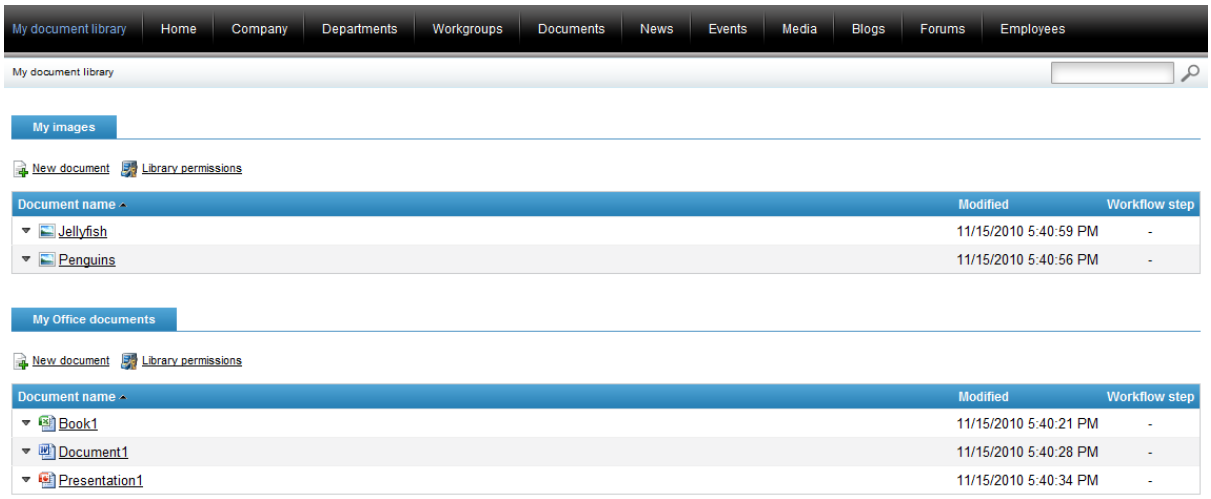
3. Repeat step 2 and add another **Document library** web part to the page. This time, use the following values of its properties:

- **Library path**: /My-document-library/My-Office-documents
- **Page size**: 15
- **Document form**: leave the field blank
- **Web part container**: Intranet simple container (with header line)
- **Container title**: My Office documents

Click **OK**. The web part zone should now look as in the following screenshot if viewed in the **Design** mode.



4. The document libraries are now configured and ready to be used by live site users. If viewed on the live site after uploading some files into the libraries, the page should look as in the screenshot below.



Please proceed to the Managing files in document libraries topic to learn more about how live site users can upload and manage document library files.

8.19.5 Managing files in document libraries

This topic explains particular actions that can be performed by live site users in a document library.

Permissions for particular actions

All of the actions listed below **are not always available**. They are displayed based on document permissions granted to the current user or their roles. Please consult the Security topic for more details on which permissions are required for each of the listed actions.

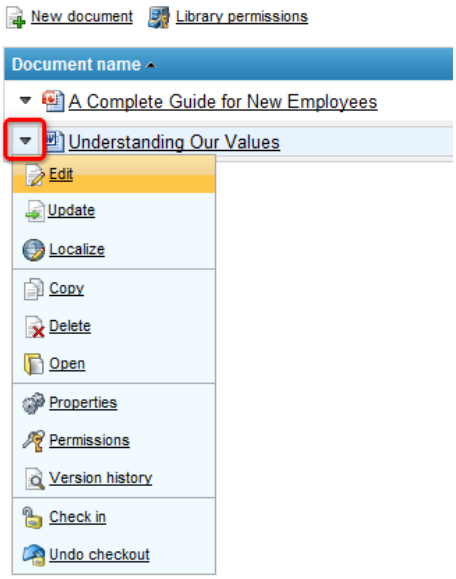
When a new document library is created, there are no files in it, so live site users can only see the web part with the following two links at the top.

- New document
- Library permissions

Once some files are uploaded in the document library, live site users can perform the following actions with the documents:

- Edit
- Update
- Localize
- Copy
- Delete
- Open
- Properties
- Permissions
- Version history
- Workflow actions

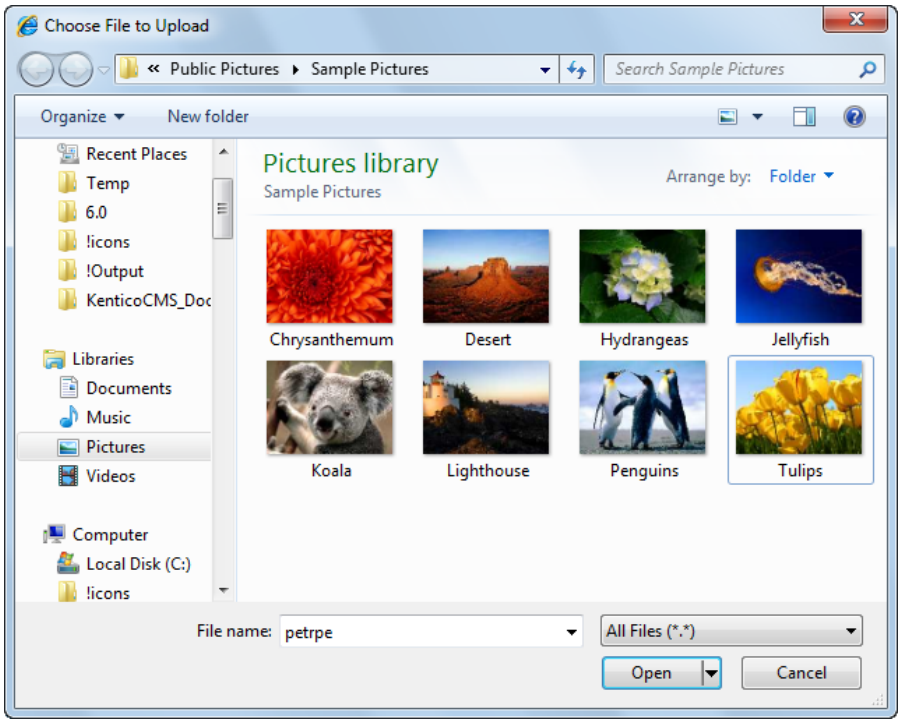
These actions can be executed from a context menu, which is accessible by right-clicking anywhere in a document's row, or by left-clicking the down-arrow to the left of a document's name (as highlighted in the screenshot below).



The following text explains particular actions that can be performed in a document library:

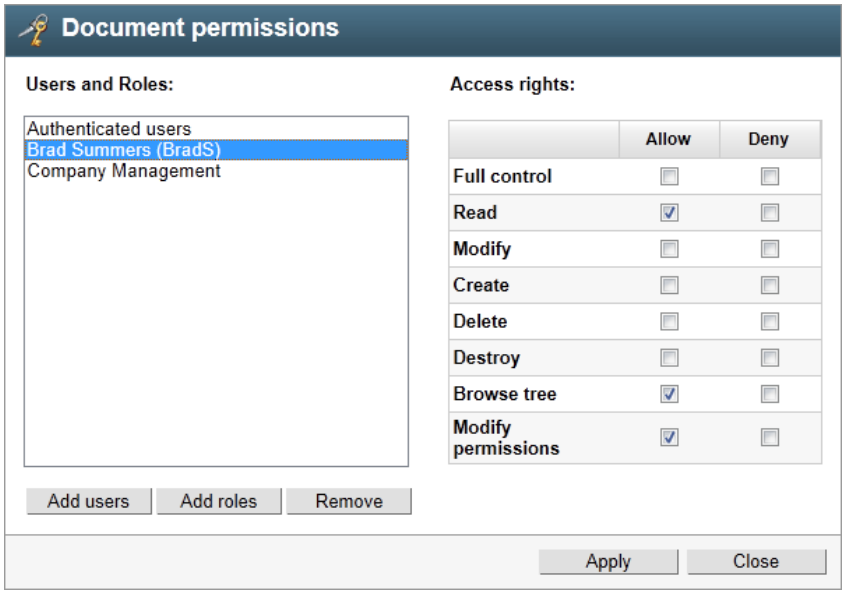
New document

Clicking the **New document** link in **Internet Explorer** or **Mozilla Firefox** directly opens the browser's file selection dialog. By selecting a file from a local disk and clicking **Open**, the file will be uploaded into the document library.



Library permissions

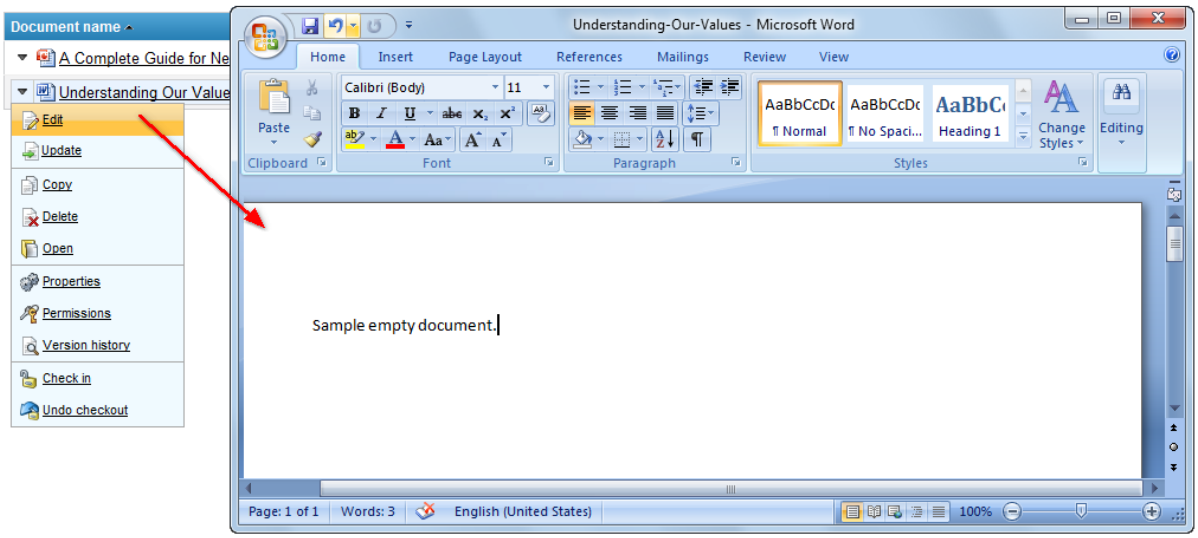
Clicking the **Library permissions** link opens a dialog for document library permissions configuration. Please refer to the **Configuring document-level permissions on the live site** section of the Security topic for more details.



Edit

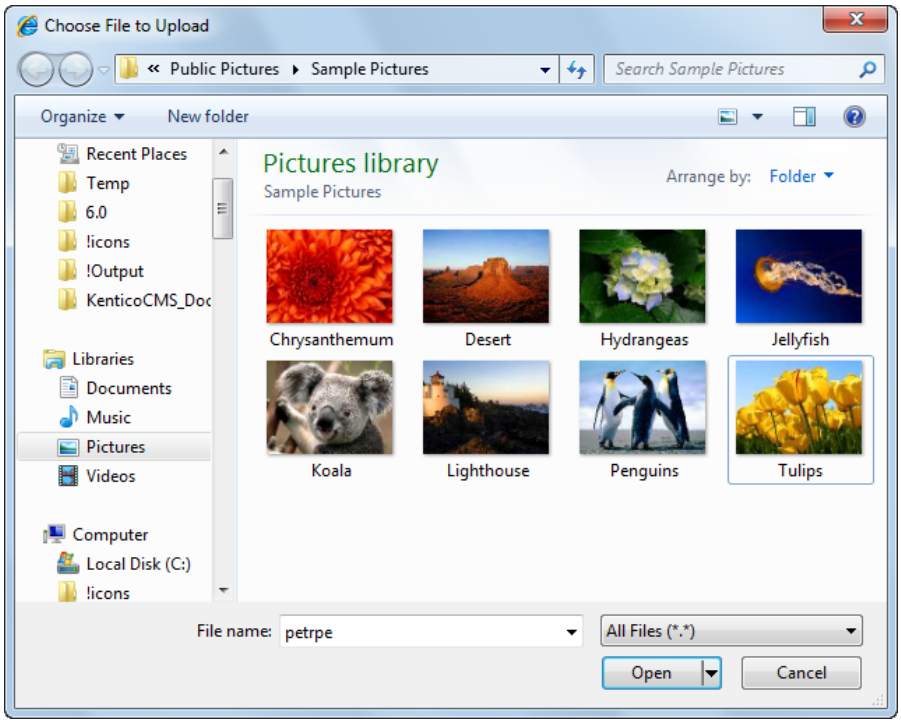
The **Edit** action is only available if WebDAV integration is enabled. It opens the file in a client

application for direct editing (for example, *.docx* documents get opened in *MS Word*, as you can see in the screenshot below). WebDAV editing is only possible if there is an application installed on the client machine that supports WebDAV editing of the particular file type.



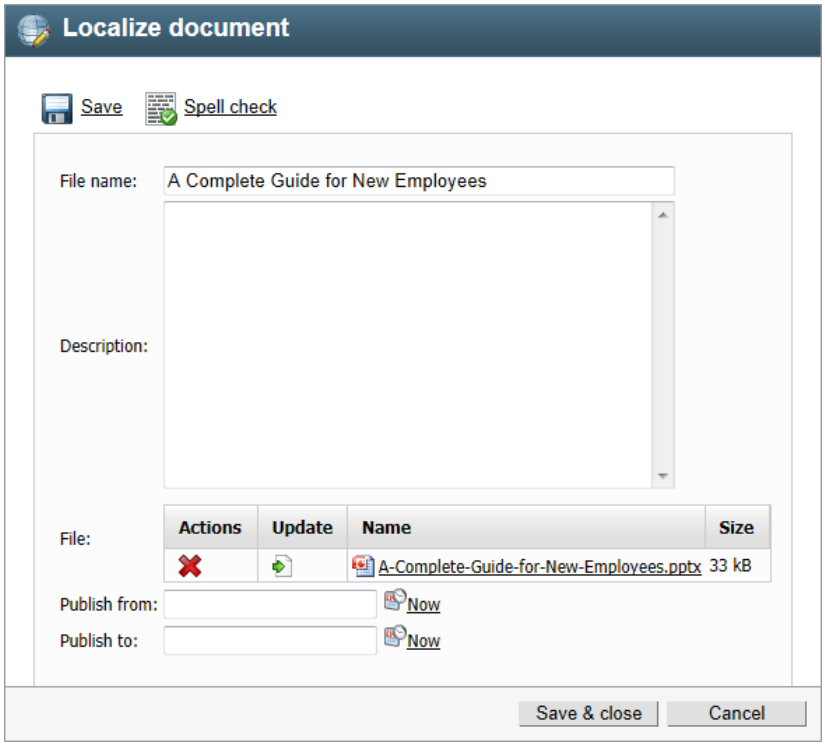
Update

Selecting the **Update** action opens the browser's file selection dialog. Selecting a file from a local disk and clicking **Open** in the dialog replaces the original file with the selected one. In **Apple Safari** and **Opera**, there is one extra dialog displayed before the browser's file selection dialog. Please refer to the New document paragraph above for more details.



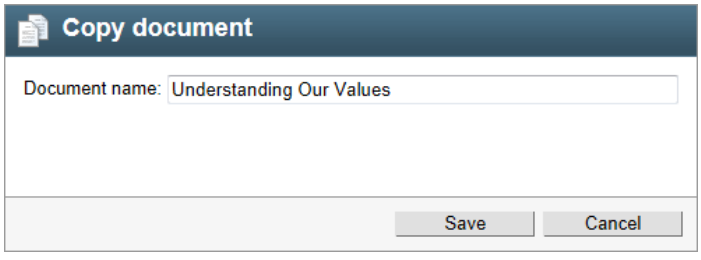
Localize

The **Localize** option is only displayed with documents from the default culture when the **Combine with default culture** property of the web part is enabled. Clicking the action opens the dialog depicted in the screenshot below, which serves for creation of a new language version of the selected document in the current culture.



Copy

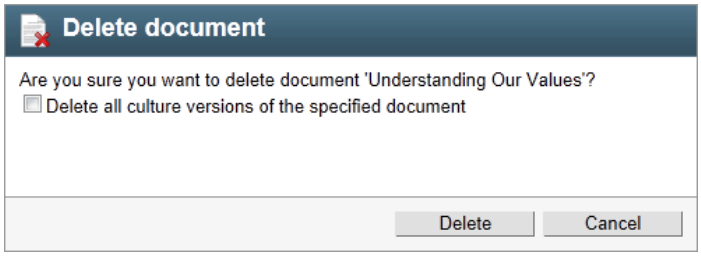
The **Copy** action creates a copy of the file in the same document library. As only flat structure of document libraries is currently possible, a different file name needs to be entered for the copy, while the " - Copy" suffix is offered to be added to the file name automatically. When copying a linked document (marked with the flag), the copy will be a standard document, not a linked one.



Delete

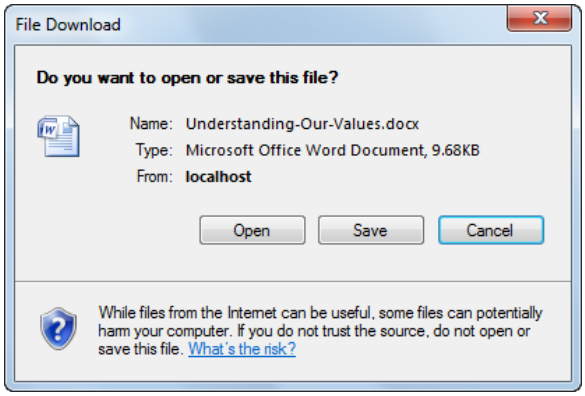
Clicking the **Delete** action opens the dialog depicted below. Using the dialog, you can remove the document from the document library. On multilingual websites where more that one language version of

the document exists, the **Delete all culture versions of the specified document** check-box is displayed in the dialog. If you delete the document with the check-box enabled, all cultural versions of the document will be deleted.



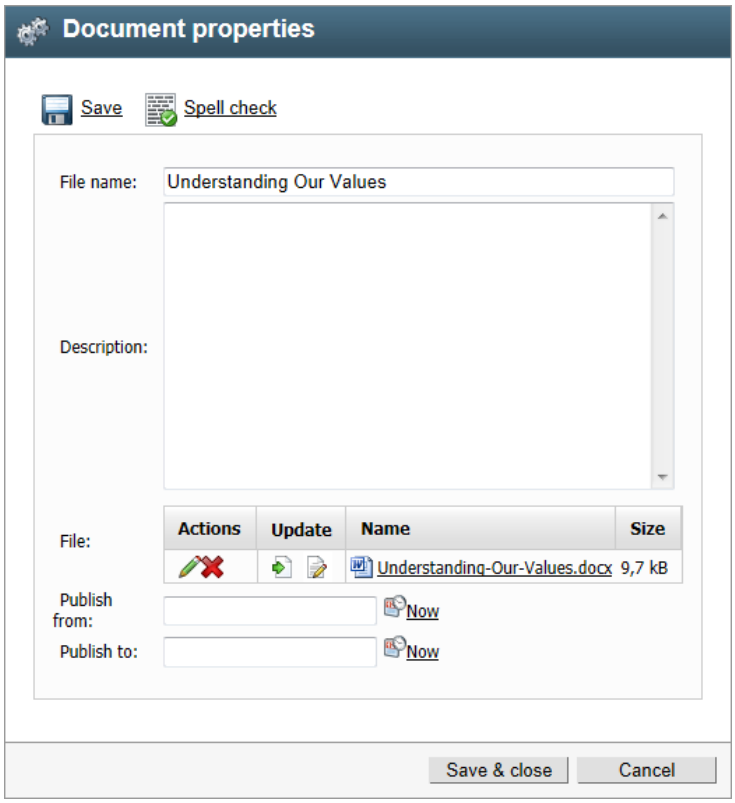
Open

The **Open** action works similarly as a standard file download link. The web browser's standard file download dialog is displayed and you can either open the document or save it to a local disk. The same behavior can be expected when you click the document's name in the library listing.

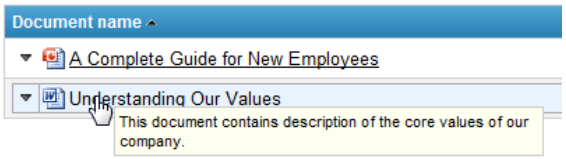


Properties

The **Properties** action opens a dialog where properties of the *CMS.File* document can be configured the same way as in **CMS Desk -> Content -> Edit -> Form**. The dialog uses the Alternative form specified in the **Document form** property of the web part. If no form is specified, the *CMS.File.DocumentLibrary* form is used. If even this form is not available, *CMS.File* document type's default form is used.

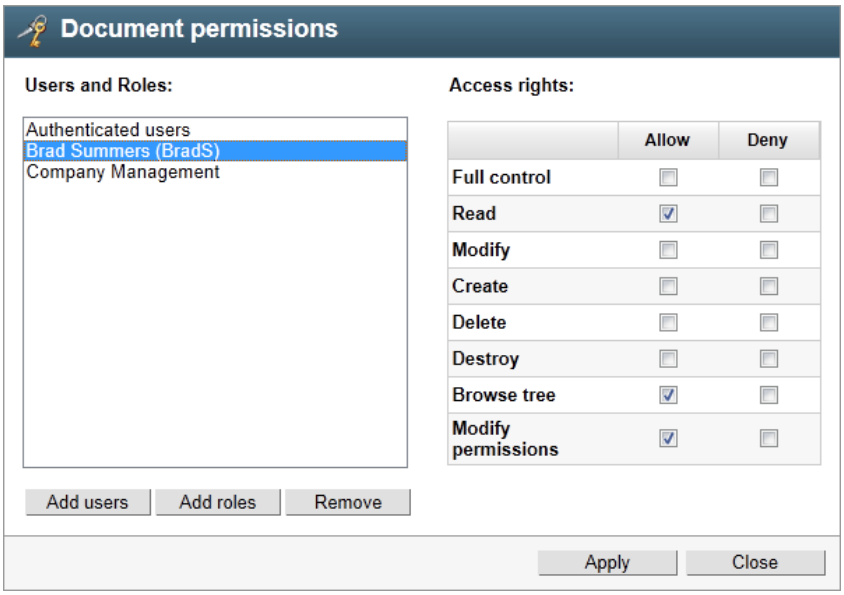


The text filled in the **Description** field is displayed in a tooltip when the mouse pointer is placed above the document.



Permissions

The **Permissions** action opens a dialog for configuration of permissions for the particular document, similarly as in **CMS Desk -> Content -> Edit -> Properties -> Security**. Please refer to the **Configuring document-level permissions on the live site** section of the Security topic for more details.



Version history

Clicking the **Version history** action displays an overview of the document's versions, similarly as in **CMS Desk -> Content -> Edit -> Properties -> Versions**. The following actions are available with each of the listed versions:

- **View -** displays an overview of the document's content with the possibility of side-by-side comparison of versions
- **Rollback -** rolls back any changes made since the particular version of the document
- **Delete** - deletes the old version

You can also use the **Clear history** button to clear all versions except the latest one. For the action to be available, the **Destroy** permission for this document must be granted to the user.

Workflow actions

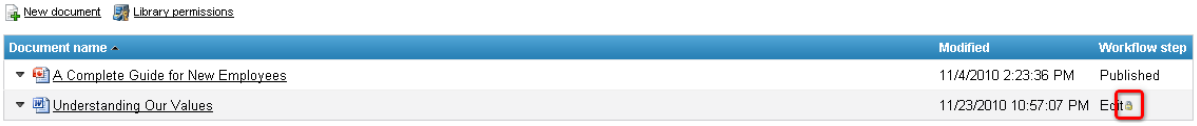
If there is a workflow configured for documents in a document library, standard workflow actions are displayed at the bottom of the context menu, depending on workflow configuration and the current

workflow step of the document:

- **Submit to approval** - submits the document for approval to the sub-sequent workflow step
- **Approve**[1] - approves the document and switches it to the sub-sequent workflow step
- **Reject**[1] - rejects the document and switches it back to the previous workflow step
- **Archive** - switches the document to the Archived step
- **Check out**[2] - checks the document out for editing; while checked out, the document can't be edited by other users; checked-out documents are marked with the icon in the document library listing (as can be seen in the screenshot below)
- **Check in**[2] - checks the modified document in after editing, so that other users can edit it again
- **Undo checkout**[2] - takes back the check-out, while changes made to the document are not saved

[1] for these actions to be available, the user must also be in one of the roles that are allowed to approve/reject the document in the current workflow step or have the **Manage workflow** permissions for all content
[2] these actions are only available if the workflow applied to the document is configured to use check-in/check-out

Document libraries are typically used with Versioning without workflow. In this case, document in the libraries are in the **Published** workflow step all the time and a new version of the document is created with each modification. Please refer to Content management -> Workflow and versioning for more information on workflows in Kentico CMS.



8.19.6 Security

The Document library module leverages standard document permissions. The following table explains which permissions are required to perform particular actions in the document library. These permissions can be granted to roles globally for all content or for the *CMS.File* document type, or to particular users or roles on document-level of the library's parent document or each particular document in the library.

Action	Read	Modify	Create	Delete	Destroy	Manage workflow	Modify permissions
New document	•	•	•				
Library permissions	•						•
Edit	•	•					
Update	•	•					
Localize	•	•	•				
Copy	•	•	•				
Delete	•			•			

Open	•						
Properties	•	•					
Permissions	•						•
Version history	•	•			1		
Submit to approval	•	•					
Approve[2]	•	•				2	
Reject[2]	•	•				2	
Archive	•	•				•	
Check out[3]	•	•					
Check in[3]	•	•					
Undo checkout[3]	•	•					

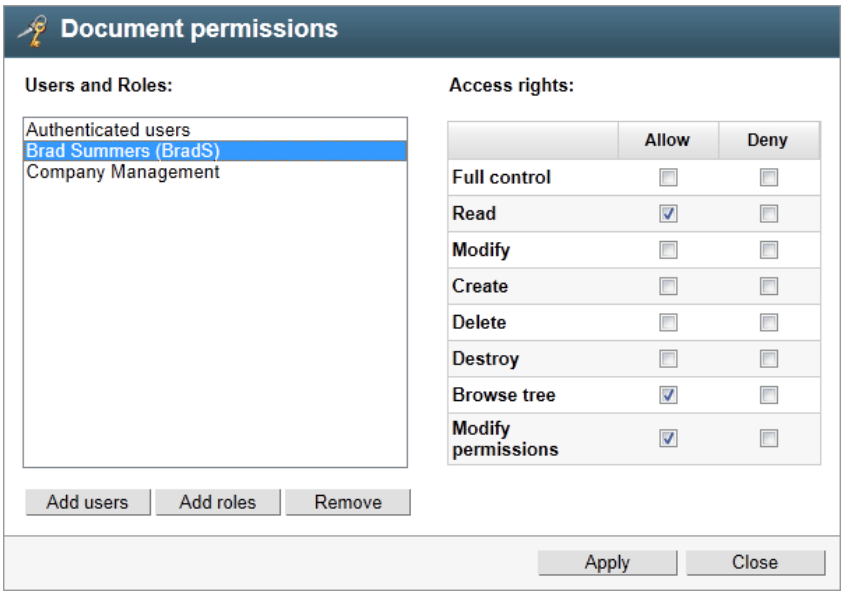
[1] The **Destroy** permission is required for the user to be able to delete particular versions or the whole version history.
[2] For these actions to be available, the user must also be in one of the roles that are allowed to approve/reject the document in the current workflow step or have the **Manage workflow** permissions for all content.
[3] These actions are only available if the workflow applied to the document is configured to use check-in/check-out.

Configuring document-level permissions on the live site

Document-level permissions can be configured directly on the live site. They can be configured either globally for the document library's parent document, which results in the permissions being inherited by the child documents in the library, or separately for each particular document in the library. Permissions can be granted to users or roles. Permissions for group document libraries can also be granted to group members and group roles.

The **Library permissions** action opens a dialog for configuration of the library's parent document permissions, i.e. the permissions that can be inherited by its child documents (the actual documents stored in the library). This dialog is identical to the **Permissions** section of the **CMS Desk -> Content -> Edit -> Properties -> Security** dialog for the document. Permissions configured via the web part on the live site will be reflected in this dialog.

By choosing the **Permissions** action from the context menu of a document in the library, the same dialog gets displayed, while this time, permissions are configured just for the particular document. Here again, the permissions configured on the live site will be reflected in the **CMS Desk -> Content -> Edit -> Properties -> Security** dialog for the document.



Permissions and workflow

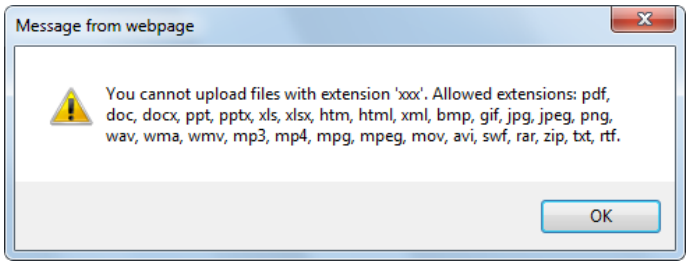
Document libraries reflect workflows applied to documents stored in them. Unless the current user has the **Modify** permission for a document, the currently published version of the document is always displayed to the user. If the document is currently archived or not published, the document is not displayed to the user at all. If the current user does have the **Modify** permission, the current version of the document (in the current workflow step) is displayed to them.

Please refer to Content management -> Workflow and versioning for more information on workflows in Kentico CMS.

Allowed file extensions

When uploading a new document into the document library using the **New document** link or updating a document using the **Update** action (both in the context menu and in the **Properties** dialog), only files with extensions defined in **Site Manager -> Settings -> System -> Files -> Upload extensions** or in the **Allowed extensions** property of the *FileAttachment* field of the *CMS.File* document type can be uploaded.

Each attempt to upload a file with an extension that is not allowed will result in the error message as displayed in the screenshot below.



8.20 E-commerce

8.20.1 Overview

For information about the **E-commerce** module please refer to Kentico CMS E-commerce Guide.

8.21 E-mail queue

8.21.1 Overview

The E-mail queue module was designed to enable the sending of large amounts of e-mails, e.g. when sending newsletter issues to subscribers, without the risk of losing any of the e-mails due to errors. If an e-mail is not delivered successfully, it remains in the queue so that it can be resent later. The e-mails in the queue are sent out **automatically**, no manual sending is necessary as this is handled by the **Send queued e-mails** scheduled task.

During the mail-out, e-mails in the queue are distributed to the SMTP servers defined in system. Please see the SMTP server configuration topic to learn how you can register and configure SMTP servers in Kentico CMS.

E-mails in the queue can be monitored or manually managed through the administration interface. Please see the Administrating the e-mail queue topic for more information.

To utilize the e-mail queue, authorized users have the option of Sending mass e-mails to large amounts of users. Certain other modules, such as Newsletters, also make use of the e-mail queue. All other e-mails sent by the system can be configured to either use the e-mail queue or be sent directly to the SMTP server, as is described in the Settings topic.

8.21.2 Administrating the e-mail queue

The e-mail queue administration interface is located in **Site Manager -> Administration -> E-mail queue**. Using the **Site** drop-down list, you can choose which site you want to display the e-mail queue for. If *(global)* is selected, e-mails sent from the administration interface will be displayed. If *(all)* is selected, all e-mails in all queues will be displayed. The administration interface is divided into three tabs:

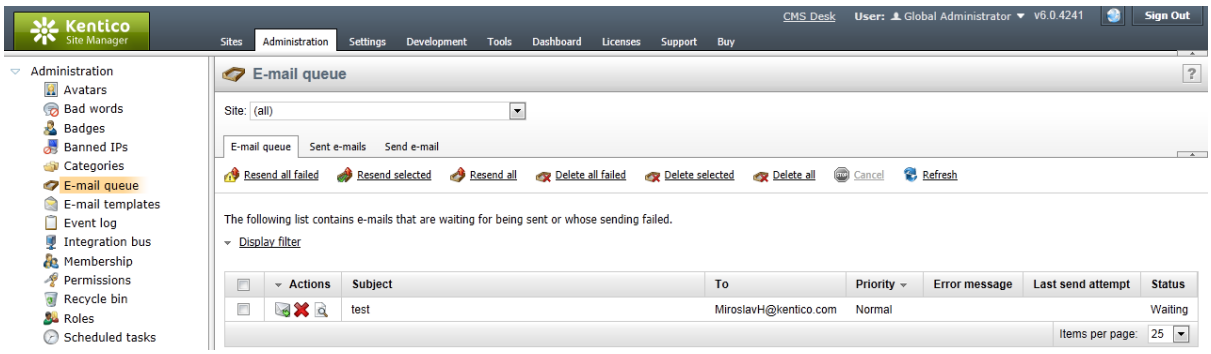
E-mail queue tab

This tab displays the actual e-mail queue. E-mails that are waiting to be sent or that haven't been sent successfully (displayed with an Error message) are displayed here. You can **Resend** ( ) the mail, **Delete** ( ) it or **View** ( ) its details. There is also a number of links above the list:

- **Resend all failed** - resends all e-mails in the queue that were unsuccessfully sent; new e-mails that have not yet been sent will not be resent
- **Resend selected** - resends all e-mails selected by the check-boxes in the list
- **Resend all** - resends all e-mails in the list

- **Delete all failed** - deletes all e-mails in the queue that were unsuccessfully sent; new e-mails that have not yet been sent will not be deleted
- **Delete selected** - deletes all e-mails selected by the check-boxes in the list
- **Delete all** - deletes all e-mails in the list

- **Refresh** - refreshes the content of the e-mail queue

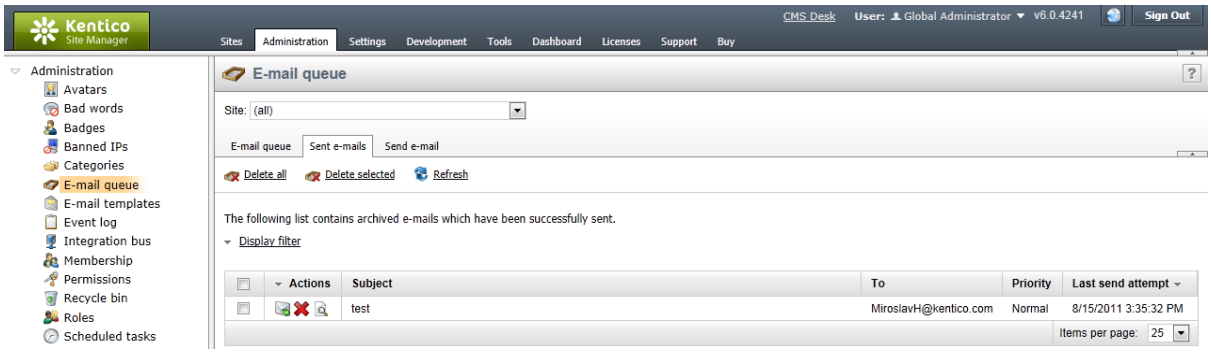


Sent e-mails tab

This tab displays a list of e-mails that have been successfully sent via the e-mail queue. You can set how long the e-mails will stay in this list through the **Site Manager -> Settings -> System -> E-mails -> Archive e-mails (days)** property.

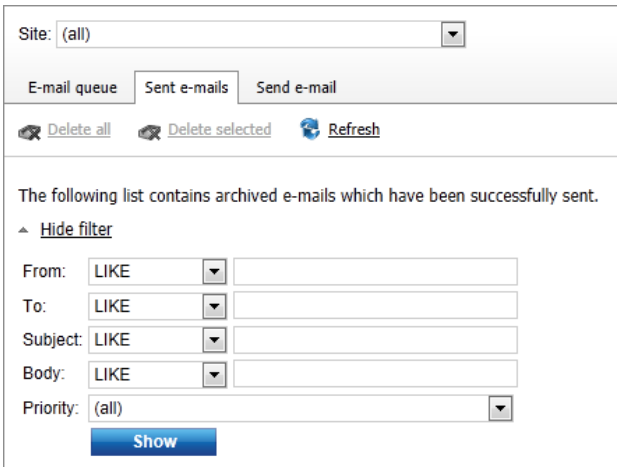
You can **Resend** ( ) the mail, **Delete** ( ) it or **View** ( ) its details. There are also three links at the top of the page:

- **Delete all** - deletes all e-mails in the list
- **Delete selected** - deletes e-mails selected by the check-boxes in the list
- **Refresh** - refreshes the content of the list of sent e-mails



E-mail filter

When there is a large number of e-mails in the e-mail queue, you can easily limit which e-mails will be displayed using the filter. It can be displayed by clicking the **Display filter** link above the e-mail lists on both tabs.



Send e-mail tab

From this tab, you can easily send a single e-mail to a specified recipient (not only the site users). For sending e-mails to multiple users, we recommend using mass e-mails instead of this tab.

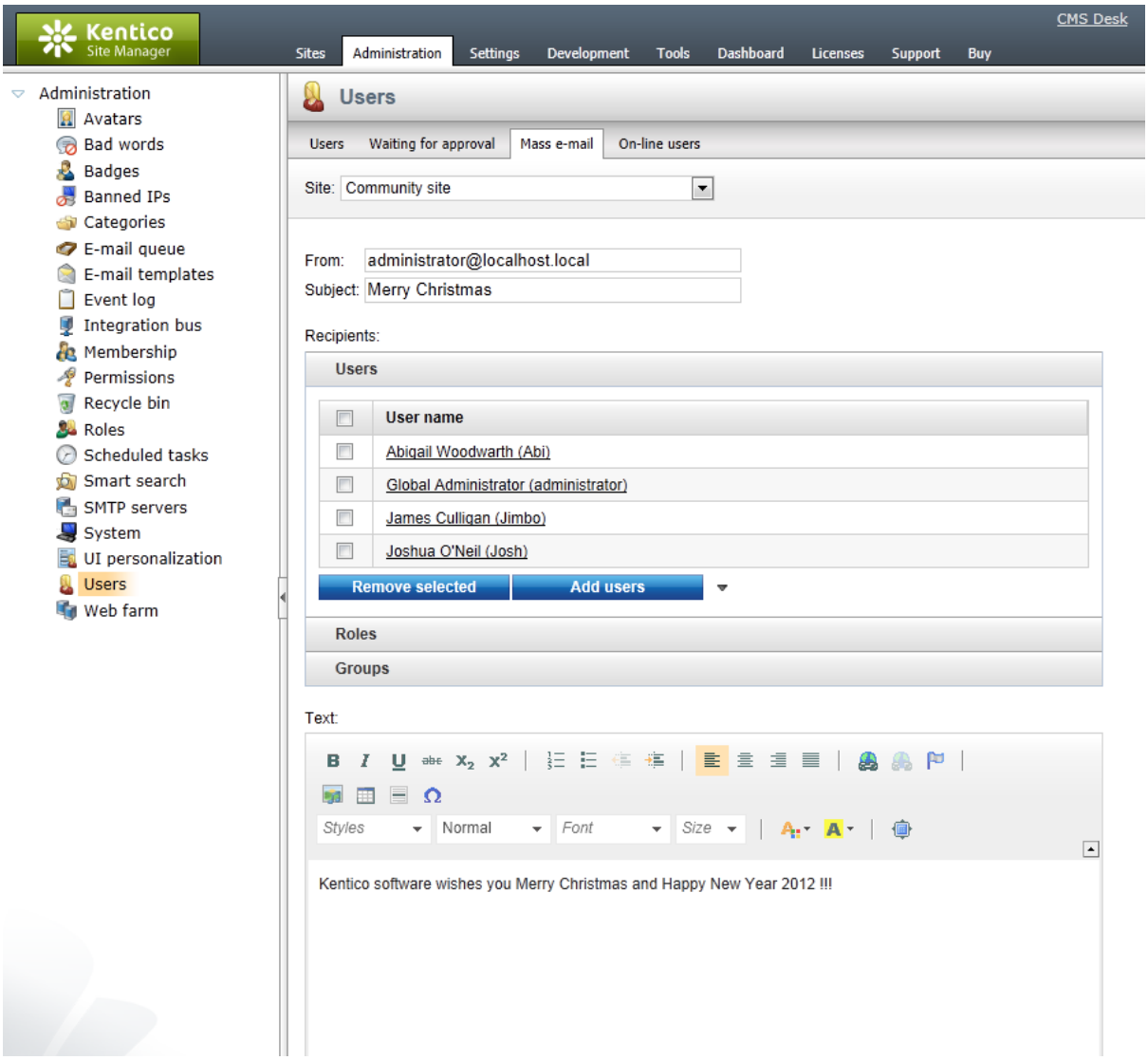
8.21.3 Sending mass e-mails

As the e-mail queue allows the sending of massive amounts of e-mails, we've implemented the Mass e-mail feature. This feature enables you to send the same e-mail to a large amount of users.

You can do this at **Site Manager -> Administration -> Users -> Mass e-mail**.

Using the **Site** drop-down list, you can select which site the recipients are related to. Based on this choice, users can be selected either directly, or according to the roles or group that they are members of by using the **Add users** buttons in the **Recipients** section. If you choose the *(all)* option from the site drop-down list, only global roles (those that are not limited to a single site) will be available and the **Groups** section will be hidden since groups are always site-related.

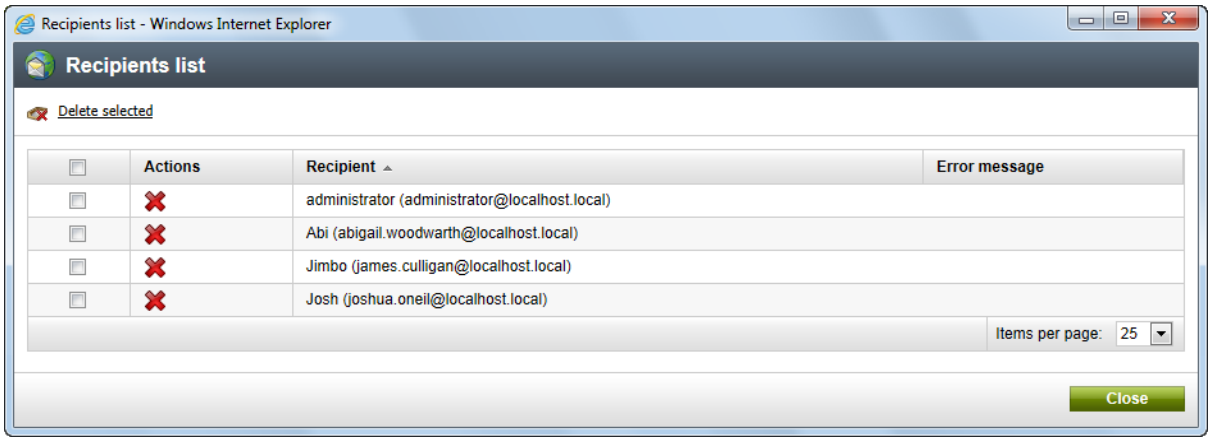
You can also add an attachment to the mail using the **Attach file** link.



This tab is also available at **CMS Desk -> Administration -> Users -> Mass e-mail**, but only users of the current site can be selected.

In the e-mail queue, you will see the **Show details** link instead of a recipient's e-mail address for mass e-mails, as you can see in the screenshot below.

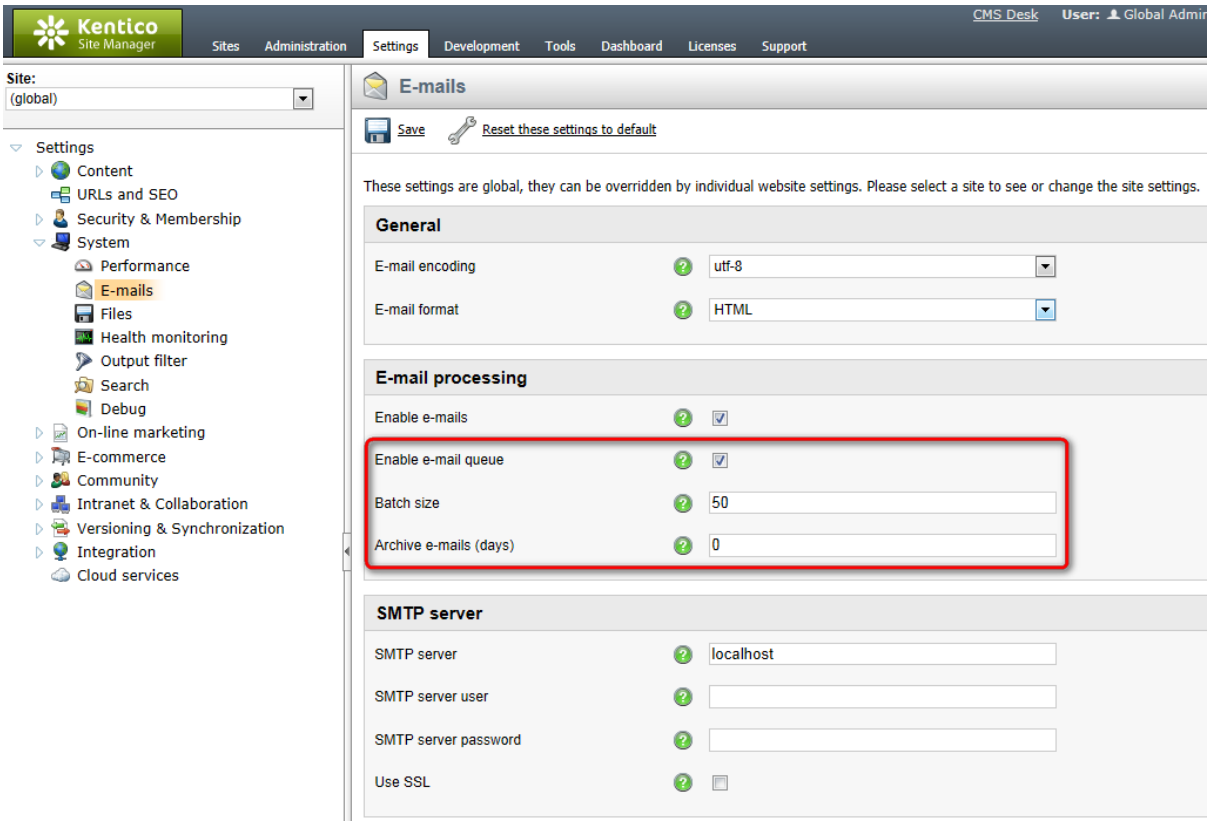
After clicking the link, a window will appear as in the screenshot below. You can delete recipients from the list, so that the mass e-mail will not be delivered to them. This can be done using the **Delete** (✖) icons or by selecting more recipients using the checkboxes and clicking **Delete selected**.



8.21.4 Settings

E-mail settings are located at **Site Manager -> Settings -> System -> E-mails**. Among other e-mail related settings, the following three can be used to enable the e-mail queue and configure its functionality:

- **Enable e-mail queue** - if checked, the e-mail queue will be used when sending e-mails. There are certain exceptions for priority e-mails, like those containing forgotten passwords for example, that skip the queue and are sent directly.
- **Batch size** - sets the maximum number of e-mails that can be transferred from the e-mail queue to an SMTP server in one batch. If the specified value is smaller than the total amount of e-mails to be sent from the queue, a new batch is prepared and assigned to the next available server. This process is repeated until all e-mails are mailed out. The setting affects all sites in the system, so it is only available if the *(global)* option is selected from the *Site* drop-down list.
- **Archive e-mails (days)** - the number of days for which e-mails sent via the e-mail queue will remain stored on the **Sent e-mails** tab. If set to 0, e-mails will not be archived.



8.22 Events

8.22.1 Overview

The Events module allows you to manage events and their attendees. It can be used to schedule various types of event like meetings, conferences, training sessions, social events, etc. Events can be displayed on your website in a calendar and you can let your website users to register for the events. Registered attendees can be managed in the module's user interface.

In the CMS, events are represented as standard documents stored in the content tree. The **Event (booking system)** document type (code name **CMS.BookingEvent**) is used for the events. Once you have some events in the content tree, you can use web parts of the module to display them on the live site and enable registration.

Web parts of the module are stored in the **Events & booking** web part category. The **Event calendar** web part appears as a standard calendar, while events located in a configured path are displayed in respective day fields. It is possible to enable registration for the events by means of the **Event registration** web part, which enables registration for the event currently selected in the calendar.

In the Publishing events topic, you can find an example demonstrating how these two web parts can be used together in a typical scenario. The example also shows how events to be displayed can be created. Another example is located in the Using the Event calendar with other document types topic — this one shows how the **Event calendar** web part can be used to display other document types than the default **Event (booking system)**. The E-mail invitations topic in the same part of the guide provides information on invitation e-mails sent to new attendees when they perform registration to an event.

User interface of the Events module can be found in **CMS Desk -> Tools -> Events**. This is where you can see an overview of all events, view registered attendees and perform other related actions. Management of event attendees is described in the Managing attendees topic. Access to this part of the user interface, as well as to creating documents, can be limited by means of permissions. Their configuration is explained in the Security topic.

The Events internals and API sub-chapter provides information about the database tables and classes used by the module, as well as examples of how booking events and their attendees can be managed using Kentico CMS API.

8.22.2 Publishing events

Events managed by the Events module are stored in the content tree as documents of the **Event (booking system)** document type (code name **CMS.BookingEvent**). The document type has the following fields:

Event name	Name of the event.
Event summary	Text summarizing the event.
Event details	Detailed text providing full details about the event.
Event location	Location of the event.
Start date	Date and time of the event's start.
End date	Date and time of the event's end. This value is optional and if not entered, the event will be displayed as one-day event occurring on the date specified by the **Start date** value.
All day event	Indicates if the event occurs during the whole day.
Capacity	Maximal number of registered attendees.
Allow registration over capacity	Indicates if the number of registered attendees can be higher than the defined capacity.
Open from	Date and time when attendee registration starts.
Open to	Date and time when attendee registration ends.
Log on-line marketing activity	Indicates if registration should be logged as an activity by the Contact management module.
Publish from	Date and time from when the event should be published on the website.
Publish to	Date and time until when the event should be published on the website.

In a typical scenario, events are displayed using the **Event calendar** web part, while registration to the events is ensured by the **Event registration** web part. You can see an example of such setup on the **Events** page of the sample **Corporate Site**.

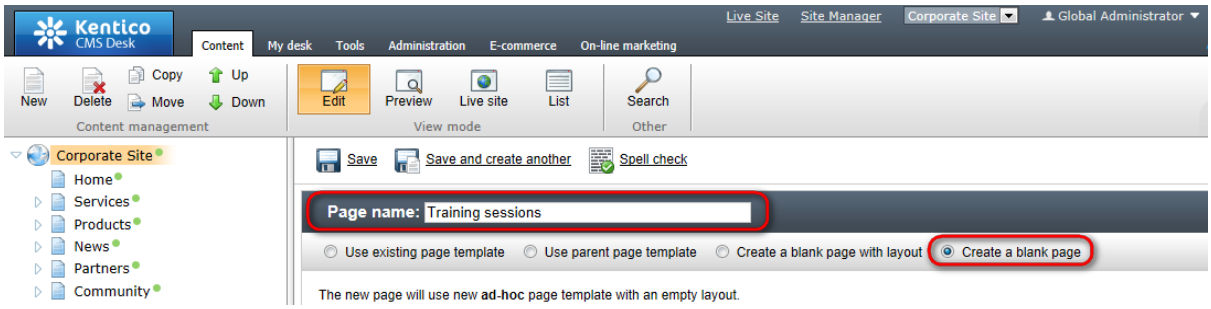
Example

In the following example, you will learn how to schedule new events and create a page where the events will be displayed in a calendar, with the possibility to register for them. For the purpose of this example, we will create a sample **Training sessions** page on the sample **Corporate Site** and use it to schedule

a few events.

Part 1: Creating the page for events

1. To get started, log on to **CMS Desk** and create a new **Page (menu item)** in the root of the content tree. When creating it, enter *Training sessions* as its **Page name** and choose the **Create a blank page** option. Finally, click **Save**.



2. Once the page is created, view it on the **Design** tab and click the **Add web part** (✚) icon in the top right corner of the only web part zone on the page. In the web part selection dialog, choose the **Events & booking -> Event calendar** web part. Configure its properties as follows:

- **Path**: */Training-sessions/%*; path where displayed event documents will be stored.
- **Event start field**: *EventDate*; document field that determines the event's start date and time.
- **Event end field**: *EventEndDate*; document field that determines the event's end date and time.
- **Document types**: *cms.bookingevent*; document type used to store the displayed events in the content tree.
- **Skin ID**: *EventCalendar*; skin applied to the calendar.
- **Hide default day number**: true (checked); this needs to be enabled to avoid doubled numbers for each day, because the number is already included in the transformations that will be selected.
- **Transformation**: *CorporateSite.Transformations.CalendarEventItem;* you can use any suitable transformation, this one and the two below are chosen just to match the whole website's style.
- **No event transformation**: *CorporateSite.Transformations.CalendarNoEvent*
- **Event detail transformation name**: *CorporateSite.Transformations.CalendarEventDetail*
- **Content before**: * *; this just ensures that the calendar will have limited width so that it doesn't span over the whole web part zone.
- **Content after**: * *

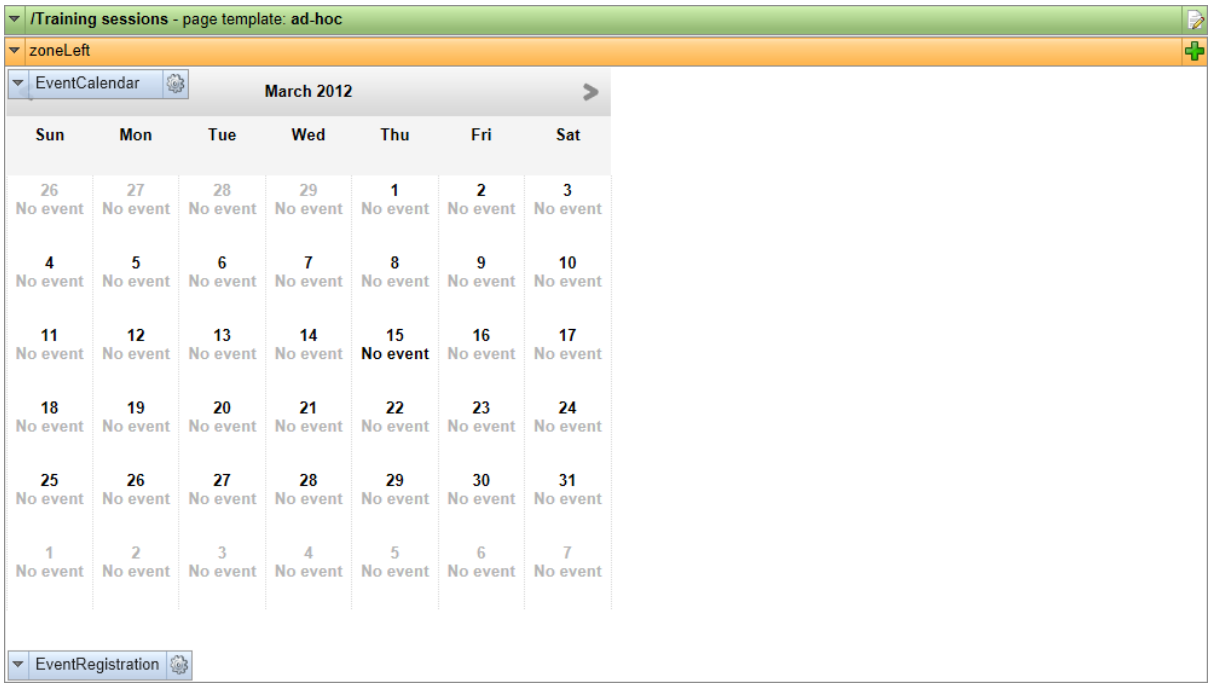
Leave the rest of the properties at their default values and click **OK**.

3. Click the **Add web part** (✚) icon again and add the **Events & booking -> Event registration** web part. Configure its properties the following way:

- **Show for document types**: *CMS.BookingEvent*; this ensures that the web part will be displayed only when an event is selected in the calendar above, letting you register for the selected event.
- **Registration title**: leave the field blank as we will ensure the title in the web part container.
- **Web part container**: Corporate Site - Light gradient box
- **Container title**: Registration

Again, leave the rest of the properties at their default values and click **OK**.

4. The page is now ready. On the **Design** tab, it should look as in the following screenshot.



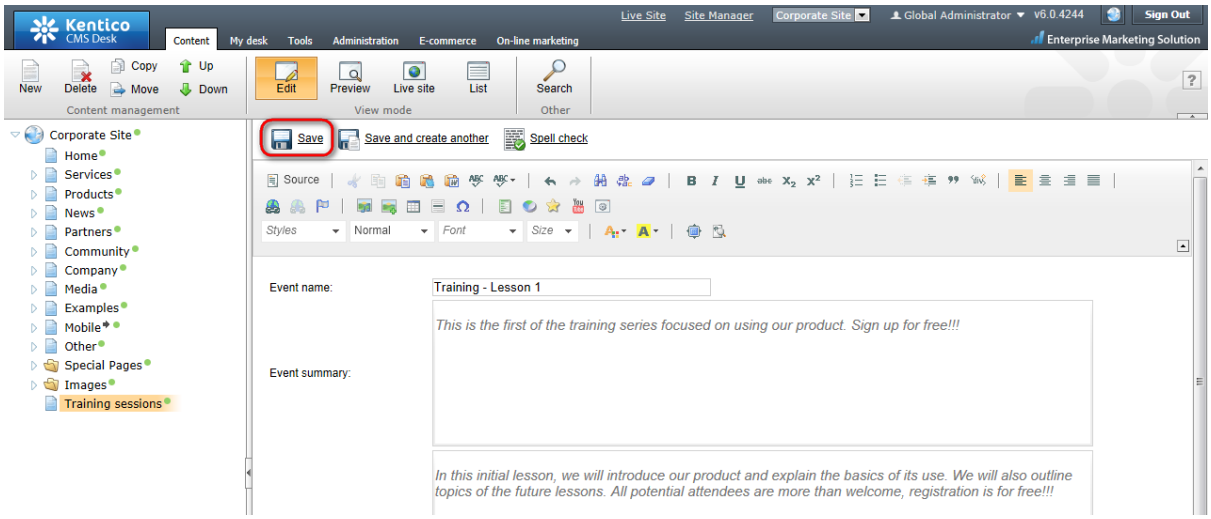
Part 2: Creating the events

The page is now ready to display events. But as you can see in the screenshot above, there are no events to be displayed yet. To see the required functionality working, we need to create the events now.

5. First, we will create a one-day event. Still in **CMS Desk**, select the **Training sessions** document in the content tree. Click the **New** button above and choose the **Event (booking system)** document type. Enter the following details into the form:

- **Event name**: *Training - Lesson 1*
- **Event summary**: *This is the first of the training series focused on using our product. Sign up for free!!!*
- **Event details**: *In this initial lesson, we will introduce our product and explain the basics of its use. We will also outline topics of the future lessons. All potential attendees are more than welcome, registration is for free!!!*
- **Event location**: *London Office*
- **Start date**: enter a future date and time in the current month so that you can instantly see it in the calendar.
- **End date**: enter a later time on the same day as entered above.
- **All day event**: *disabled*
- **Capacity**: *50*

Leave the rest of the fields blank or at default values and click **Save**.

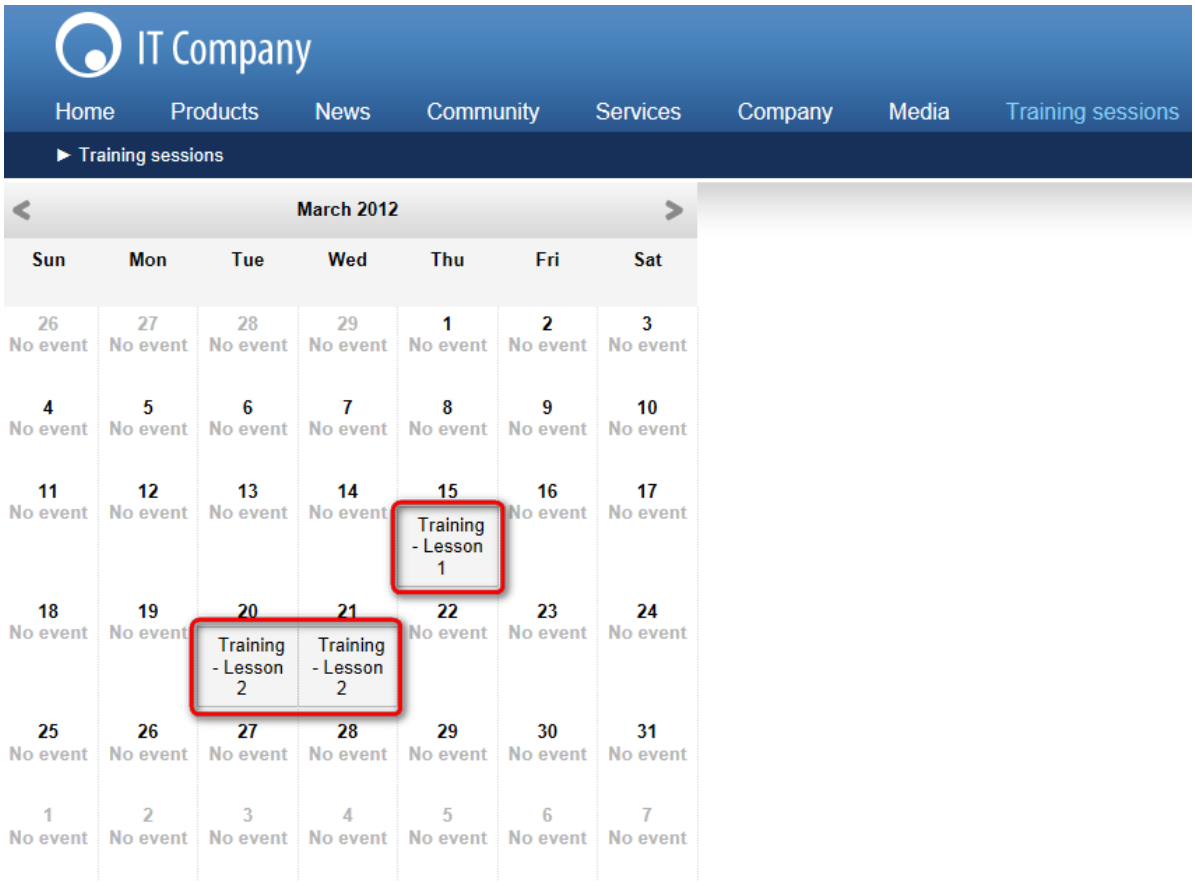


6. Now let's repeat the previous step and create a multi-day event. Select the **Training sessions** document again, click the **New** button and choose the **Event (booking system)** document type. This time, enter the following details:

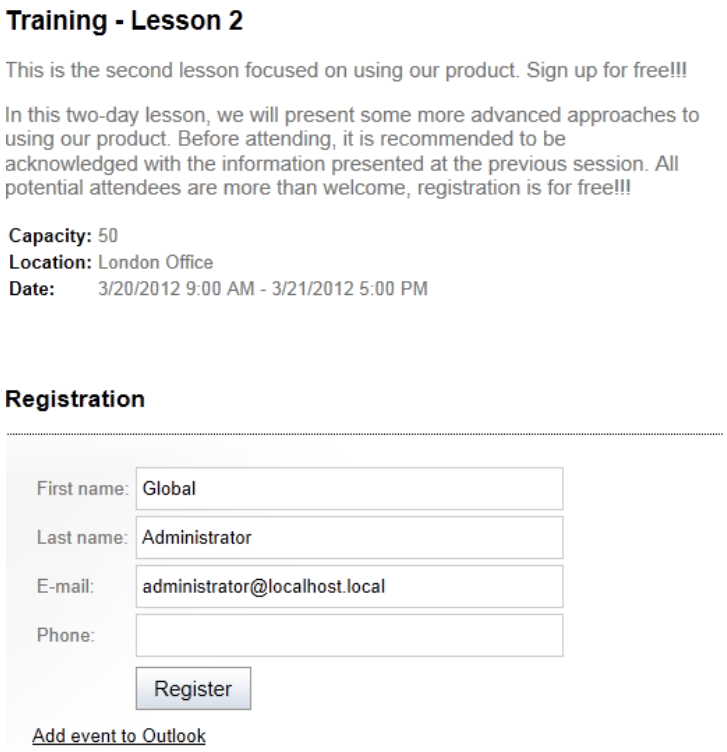
- **Event name**: *Training - Lesson 2*
- **Event summary**: *This is the second lesson focused on using our product. Sign up for free!!!*
- **Event details**: *In this two-day lesson, we will present some more advanced approaches to using our product. Before attending, it is recommended to be acknowledged with the information presented at the previous session. All potential attendees are more than welcome, registration is for free!!!*
- **Event location**: *London Office*
- **Start date**: enter a future date in the current month so that you can instantly see it in the calendar.
- **End date**: enter a date one day after the *Start date* so that a multi-day event is created.
- **All day event**: *disabled*
- **Capacity**: *50*

Leave the rest of the fields blank or at default values and click **Save**.

7. Switch to the live site. If you navigate to the newly created page, you should see the events in the calendar as in the screenshot below.



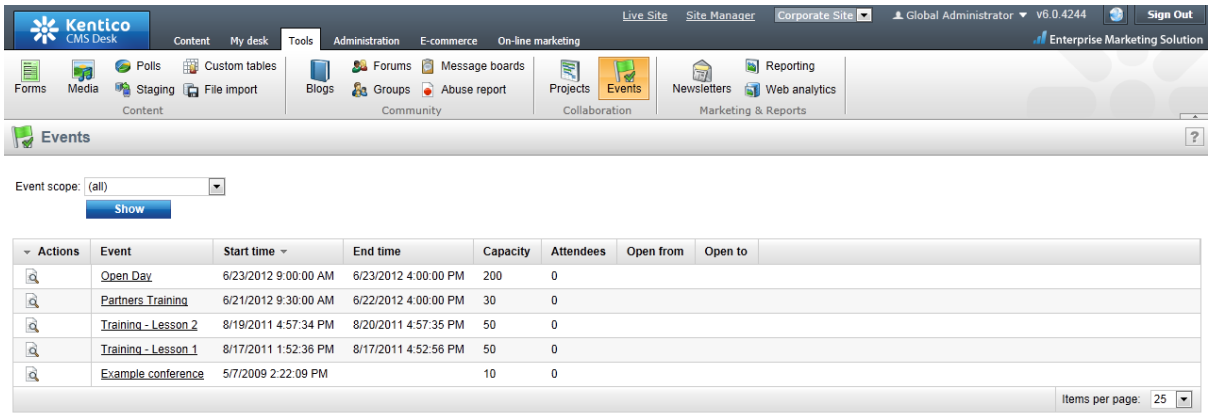
8. If you click the date field where an event is displayed, you should get its details displayed below the calendar. You should also see the registration form, enabling you to register for the event.



Try filling in the registration form a few times. You will be able to check the submitted registrations in **CMS Desk -> Tools -> Events**, as described in the Managing attendees topic.

8.22.3 Managing attendees

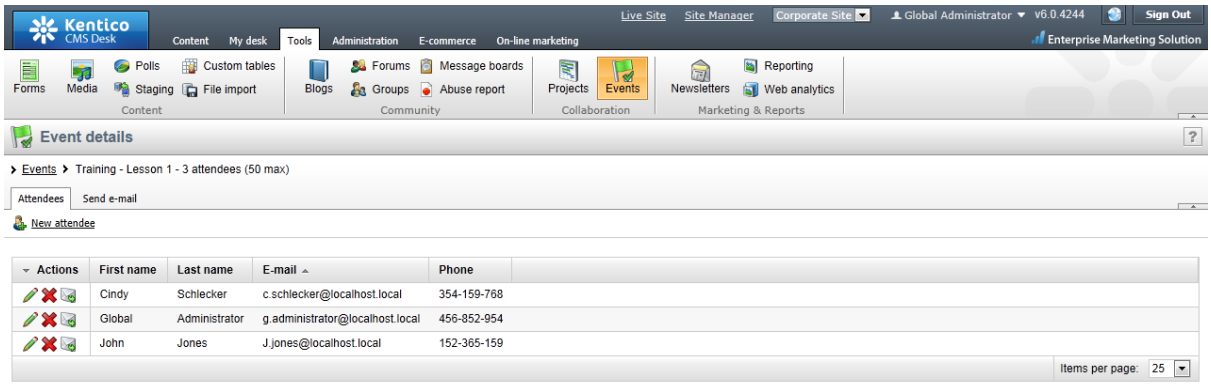
Events can be managed in **CMS Desk -> Tools -> Events**. Here, you can see a list of all events defined on the currently edited site, their capacity, current numbers of registered attendees, etc.



By clicking the event name in the **Event** column, you get redirected to the event document's editing interface in **CMS Desk -> Content -> Edit**. If you click the **View** ( ) icon next to an event, you get redirected to the event's editing interface where attendees can be managed.

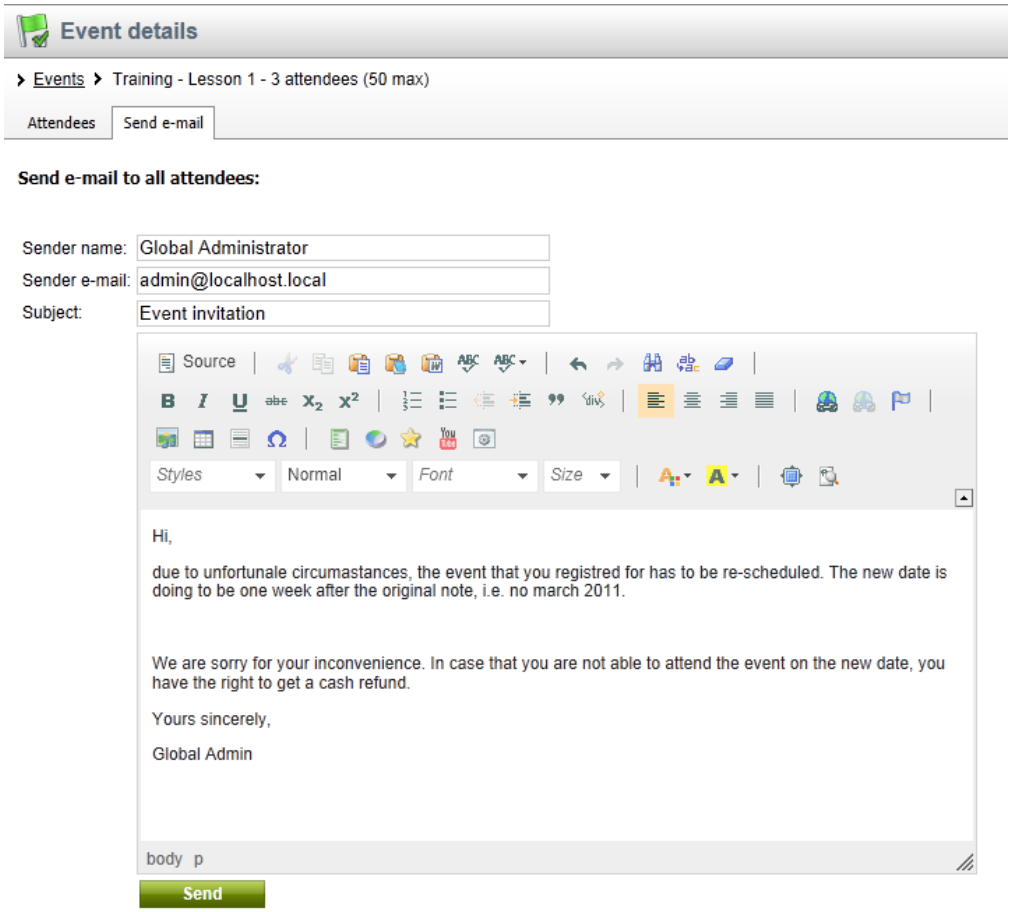
It is split into two tabs, while the **Attendees** tab will be displayed initially. Here, you can see a list of all attendees registered for the event. New attendees can be added to the event manually after clicking **New attendee** above the list. The following actions can be performed with each attendee:

- **Edit** - redirects you to the registration editing form where you can modify details submitted by the attendees on registration.
- **Delete** - clicking this icon removes the attendee from the event.
- **Resend invitation e-mail** - clicking this icon sends the invitation e-mail to the respective attendee. See E-mail invitations for more details.



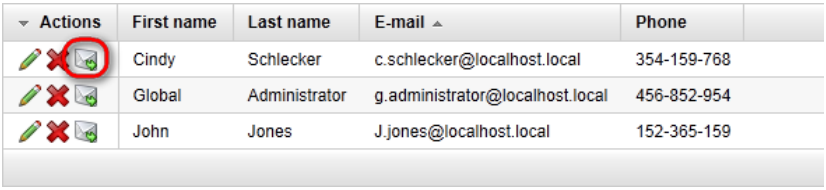
Sending e-mails to all attendees

On the **Send e-mail** tab, you can send an e-mail to all attendees currently registered for the event. This may come in handy in case of changes in the event schedule, when sending additional information about the event, etc.

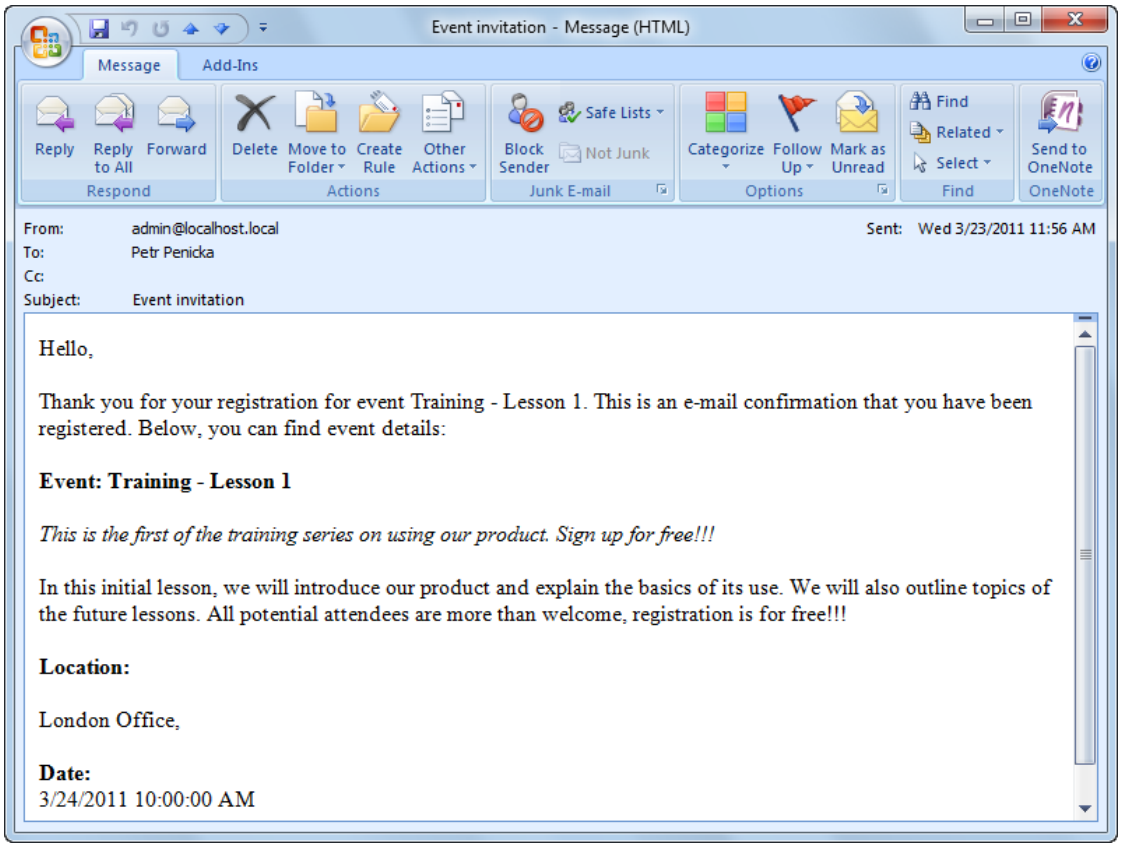


8.22.4 E-mail invitations

When an attendee registers for an event by means of the **Event registration** web part, an automatic e-mail notification is sent to their e-mail address. It is also possible to send the invitation again by clicking the **Resend invitation e-mail** icon in the respective line of the event attendees list in **CMS Desk -> Tools -> Events**.



The notification e-mail is based on the **Booking system - Event invitation** e-mail template. You can see a sample e-mail message based on this template in the screenshot below. The e-mail template can be customized in **Site Manager -> Administration -> E-mail templates**, as described in the Development -> E-mail templates chapter of this guide.

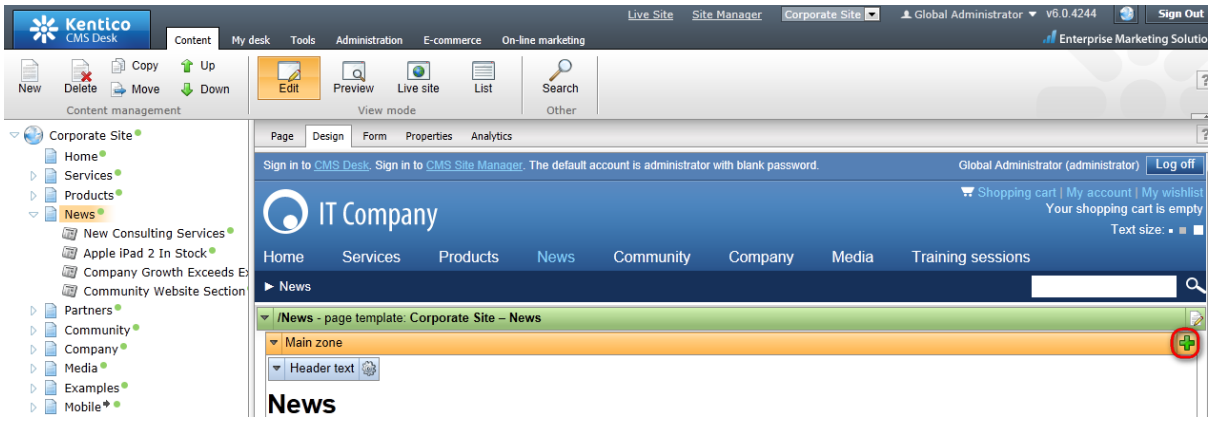


8.22.5 Using the Event calendar with other document types

The **Event calendar** web part can not only display **Event (booking system)** documents, but also documents of any other type that have a date-time field. All you have to do is specify the date-time field in the **Date field** property of the web part. This way, you can for example display news in a calendar according to their publish date.

In the following example, you will learn how to use the **Event calendar** web part to display news from the **News** section of the sample **Corporate Site**.

1. Sign in to **CMS Desk** and on the **Content** tab, select the **News** document in the content tree. View it on the **Design** tab and click the **Add web part** (✚) icon of the **Main zone** web part zone.



2. Choose the **Event calendar** web part and click **OK**. In the web part properties dialog, configure its properties as follows:

- **Show for document types**: *CMS.MenuItem*; this ensures that the calendar will only be displayed in the list of news and not when a particular news item is viewed.
- **Path**: */News/%*; path to the news documents to be displayed.
- **Event start field**: *NewsReleaseDate*; news will be displayed in respective calendar days based on the value in this field.
- **Document types**: *CMS.News*; this ensures that only news documents stored in the path will be displayed.
- **Skin ID**: *EventCalendar*; skin applied to the calendar.
- **Transformation**: click the New button and create the following transformation: *<a href="<%# GetDocumentUrl() %>"><%# Eval("NewsTitle", true) %>
*
- **Content before**: * *; this just ensures that the calendar will have limited width so that it doesn't span over the whole web part zone.
- **Content after**: * *

Leave the rest of the properties at their default values and click **OK**.

3. Now if you go to the live site and view the **News** page, you should see the calendar below the list of news. If you browse to the appropriate month where the sample news are published (or if you create your own sample news with a current date), you should see the news in the calendar. After clicking the news title, you should be redirected to the particular news item's detailed view.

June 2011

Sun	Mon	Tue	Wed	Thu	Fri	Sat
29 No event	30 No event	31 No event	1 No event	2 No event	3 No event	4 No event
5 New Consulting Services	6 No event	7 No event	8 No event	9 Apple iPad 2 In Stock	10 No event	11 No event
12 No event	13 No event	14 No event	15 No event	16 No event	17 Company Growth Exceeds Expectations	18 No event
19 No event	20 No event	21 No event	22 No event	23 No event	24 No event	25 No event
26 No event	27 No event	28 No event	29 Community Website Section	30 No event	1 No event	2 No event
3 No event	4 No event	5 No event	6 No event	7 No event	8 No event	9 No event

8.22.6 Security

Security settings of the Events module can be configured at two levels:

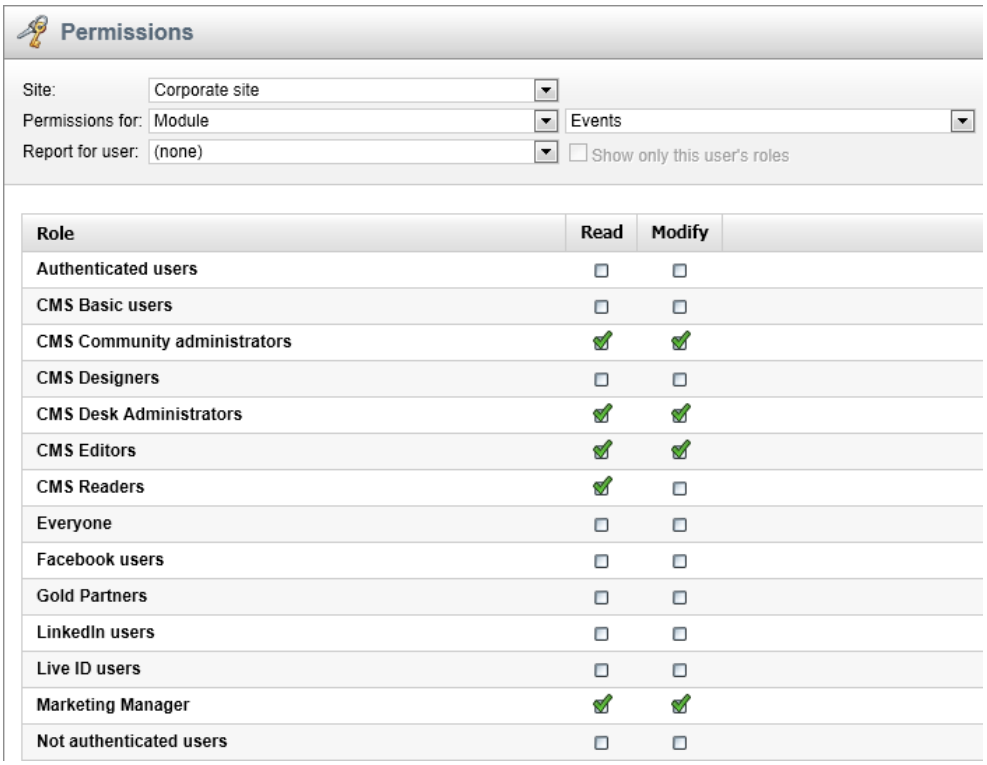
Management of events

Since the events are standard documents, the users who manage them need to have appropriate **document permissions** configured on the three levels of the document permissions hierarchy, as described in Development -> Membership -> Permissions -> Document permissions.

Management of attendees

Event attendees can be managed in **CMS Desk -> Tools -> Events**. In the **Administration -> Permissions** section of both **CMS Desk** and **Site Manager**, permissions for this module need to be granted to the roles that should be able to manage event attendees. You can assign the following permissions to the roles:

- **Modify** - allows users to modify (add, update, delete) lists of attendees and attendee details, as well as re-send invitation messages and send e-mails to all attendees.
- **Read** - allows users to read the lists of events and attendees of particular events.



8.22.7 Events internals and API

8.22.7.1 Overview

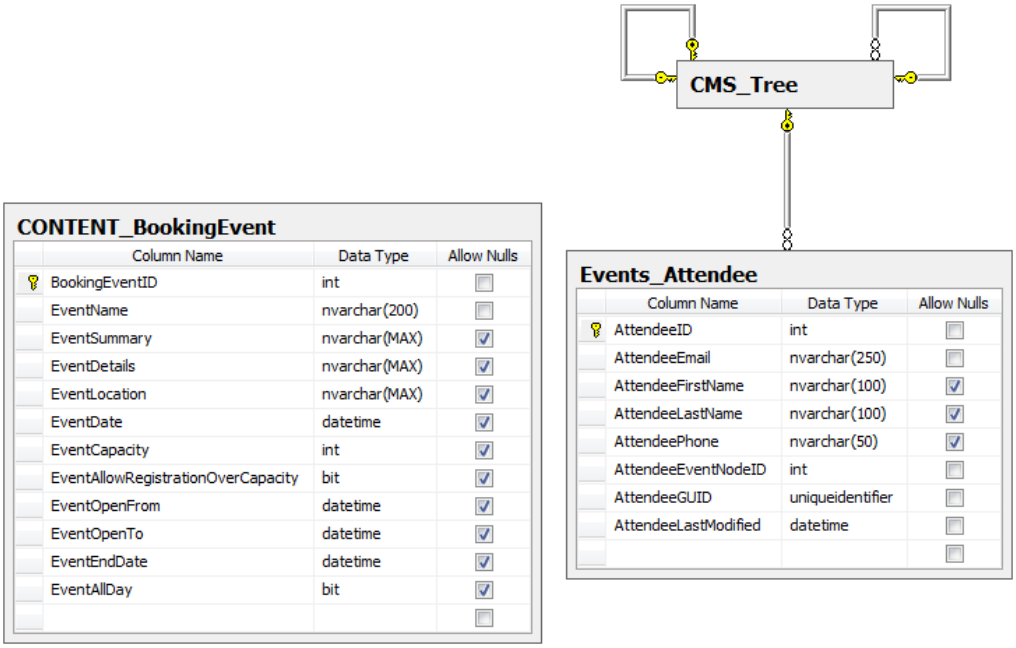
In this chapter, you will learn which database tables and API classes are used in the Events module. You will also see the most common API examples.

Advanced API examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all methods from the module classes, please refer to Kentico CMS API Reference.

8.22.7.2 Database tables

The following database tables are used in the Events module:

- **CONTENT_BookingEvent** - the *CMS.BookingEvent* is a standard Kentico CMS document type, therefore, it has this associated database table where records representing particular booking events are stored.
- **Events_Attendee** - contains records representing booking event attendees. It is bound to a record in the *CMS_Tree* table which represents the booking event's document in the content tree.



8.22.7.3 API classes

If you are not familiar with the database table data management by Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

Classes for management of documents can be found in the **CMS.DocumentEngine** namespace. Classes for management of event attendees are located in the **CMS.EventManager** namespace.

Events management API

Because booking events are standard documents stored in the content tree, their management is carried out the same way as management of any other documents using the following classes:

- **TreeNode** - represents one content tree node.
- **TreeProvider** - provides functionality manipulation with tree nodes.

For more information on documents management API, please refer to Content management -> Content management internals and API.

Events_Attendee table API:

- **EventsAttendeeInfo** - represents one event attendee object.
- **EventsAttendeeInfoProvider** - provides functionality for management of event attendees.

8.22.7.4 API examples

8.22.7.4.1 Overview

These topics show examples of how the Events module API can be used:

- Managing events
- Managing attendees

Please note

All API examples can be simulated in the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Tools\Events\Default.aspx.cs***.

The Events module API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.GlobalHelper;
using CMS.UIControls;
using CMS.CMSHelper;
using CMS.SiteProvider;
using CMS.EventManager;
using CMS.DocumentEngine;
using CMS.WorkflowEngine;
using CMS.PortalEngine;
```

8.22.7.4.2 Managing events

The following example creates an event.

```
private bool CreateEvent()
{
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get root document
    TreeNode root = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/", null,
true);

    // Create a new CMS Page (menu item) document
    TreeNode node = TreeNode.New("CMS.MenuItem", tree);

    // Set values
    node.DocumentName = "MyNewDocument";
```

```
    node.DocumentCulture = CMSContext.PreferredCultureCode;

    // Get page template
    PageTemplateInfo template = PageTemplateInfoProvider.GetPageTemplateInfo
("cms.empty");
    if (template != null)
    {
        node.SetDefaultPageTemplateID(template.PageTemplateId);
    }

    // Insert the document
    DocumentHelper.InsertDocument(node, root.NodeID, tree);

    // Create new Event (booking system) document
    TreeNode eventNode = TreeNode.New("CMS.BookingEvent", tree);

    // Set values
    eventNode.DocumentName = "MyNewEvent";
    eventNode.DocumentCulture = CMSContext.PreferredCultureCode;
    eventNode.SetValue("EventSummary", "My event summary");
    eventNode.SetValue("EventDetails", "My event details");
    eventNode.SetValue("EventLocation", "My location");
    eventNode.SetValue("EventDate", DateTime.Now);
    eventNode.SetValue("EventCapacity", 100);

    // Get page template
    PageTemplateInfo eventTemplate = PageTemplateInfoProvider.GetPageTemplateInfo
("cms.empty");
    if (eventTemplate != null)
    {
        eventNode.SetDefaultPageTemplateID(eventTemplate.PageTemplateId);
    }

    // Insert the Event (booking system) document
    DocumentHelper.InsertDocument(eventNode, node.NodeID, tree);

    return true;
}
```

The following example gets and updates an event.

```
private bool GetAndUpdateEvent()
{
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get event document
    TreeNode node = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/
MyNewDocument/MyNewEvent", null, true);

    if (node != null)
    {
        // Update value
```

```
        node.SetValue("EventDetails", "My event details were updated.");
        node.SetValue("EventCapacity", 200);
        DocumentHelper.UpdateDocument(node, tree);

        return true;
    }

    return false;
}
```

The following example deletes an event.

```
private bool DeleteEvent()
{
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get event document
    TreeNode eventNode = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/
MyNewDocument/MyNewEvent", null, true);

    // Get events parent document
    TreeNode node = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/
MyNewDocument", null, true);

    if (eventNode != null && node != null)
    {
        // Delete event document
        DocumentHelper.DeleteDocument(eventNode, tree, true, true, true);

        // Delete document
        DocumentHelper.DeleteDocument(node, tree, true, true, true);

        return true;
    }

    return false;
}
```

8.22.7.4.3 Managing attendees

The following example creates an attendee.

```
private bool CreateAttendee()
{
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get event document
    TreeNode eventNode = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/
MyNewDocument/MyNewEvent", null, true);
```

```
    if (eventNode != null)
    {
        // Create new attendee object
        EventAttendeeInfo newAttendee = new EventAttendeeInfo();

        // Set the properties
        newAttendee.AttendeeEmail = "MyNewAttendee@localhost.local";
        newAttendee.AttendeeEventNodeID = eventNode.NodeID;
        newAttendee.AttendeeFirstName = "My firstname";
        newAttendee.AttendeeLastName = "My lastname";

        // Save the attendee
        EventAttendeeInfoProvider.SetEventAttendeeInfo(newAttendee);

        return true;
    }

    return false;
}
```

The following example gets and updates an attendee.

```
private bool GetAndUpdateAttendee()
{
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get event document
    TreeNode eventNode = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/
MyNewDocument/MyNewEvent", null, true);

    if (eventNode != null)
    {
        // Get the attendee
        EventAttendeeInfo updateAttendee =
EventAttendeeInfoProvider.GetEventAttendeeInfo(eventNode.NodeID,
"MyNewAttendee@localhost.local");
        if (updateAttendee != null)
        {
            // Update the properties
            updateAttendee.AttendeeEmail = updateAttendee.AttendeeEmail.ToLower();

            // Save the changes
            EventAttendeeInfoProvider.SetEventAttendeeInfo(updateAttendee);

            return true;
        }
    }

    return false;
}
```

The following example gets and bulk updates attendees.

```
private bool GetAndBulkUpdateAttendees()
{
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get event document
    TreeNode eventNode = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/
MyNewDocument/MyNewEvent", null, true);

    if (eventNode != null)
    {
        // Prepare the parameters
        string where = "AttendeeEmail LIKE N'MyNewAttendee%'";

        // Get the data
        DataSet attendees = EventAttendeeInfoProvider.GetEventAttendees
(eventNode.NodeID, where, null, null, 0);

        if (!DataHelper.DataSourceIsEmpty(attendees))
        {
            // Loop through the individual items
            foreach (DataRow attendeeDr in attendees.Tables[0].Rows)
            {
                // Create object from DataRow
                EventAttendeeInfo modifyAttendee = new EventAttendeeInfo
(attendeeDr);

                // Update the properties
                modifyAttendee.AttendeeEmail =
modifyAttendee.AttendeeEmail.ToUpper();

                // Save the changes
                EventAttendeeInfoProvider.SetEventAttendeeInfo(modifyAttendee);
            }

            return true;
        }
    }

    return false;
}
```

The following example deletes an attendee.

```
private bool DeleteAttendee()
{
    TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);

    // Get event document
    TreeNode eventNode = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/
MyNewDocument/MyNewEvent", null, true);
```

```
    if (eventNode != null)
    {
        // Get the attendee
        EventAttendeeInfo deleteAttendee =
EventAttendeeInfoProvider.GetEventAttendeeInfo(eventNode.NodeID,
"MyNewAttendee@localhost.local");

        // Delete the attendee
        EventAttendeeInfoProvider.DeleteEventAttendeeInfo(deleteAttendee);

        return (deleteAttendee != null);
    }

    return false;
}
```

8.23 Event log

8.23.1 Overview

The Event log module stores information about all events that occur in the system. A log of these events can then be displayed and inspected. It is useful in case that some unwanted behavior occurs in the system and you want to find out where the problem originates or some other details about it.

The event log can be viewed in the **Administration -> Event log** section of both **CMS Desk** and **Site Manager**. In **CMS Desk**, only events related to the currently edited website can be viewed, while in **Site Manager**, you can view all events that occurred in the whole system. More details about these interfaces can be found in the Viewing logged events topic. Access to the interfaces and actions performed there can be restricted by means of permissions. To learn more, please see the Security topic.

In the Event log internals and API sub-chapter, you can find information about database tables and classes that are used for event logging, as well as several examples of how events can be logged and how the log can be managed using Kentico CMS API.

8.23.2 Viewing logged events

The event log can be viewed in the **Administration -> Event log** section of both **CMS Desk** and **Site Manager**. In **CMS Desk**, only events related to the currently edited website can be viewed, while in **Site Manager**, you can view all events that occurred in the whole system.

In the top row, you can select from the following options:

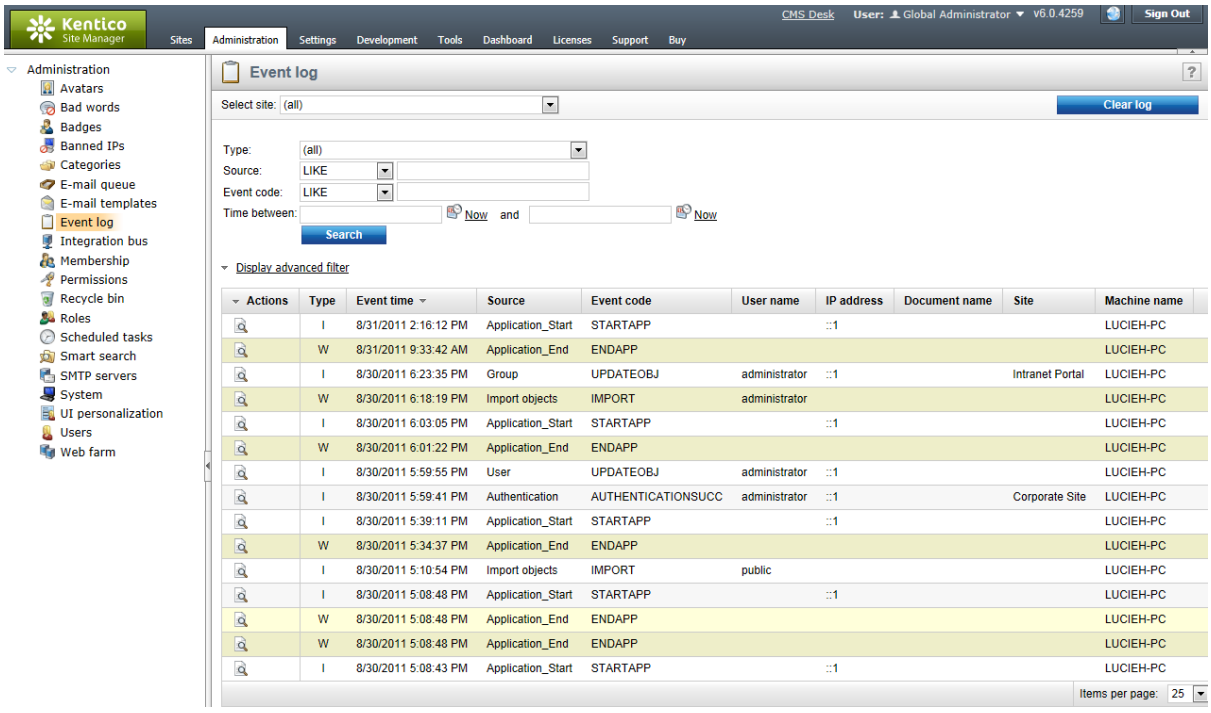
Select site	Using this drop-down list, you can select what events will be displayed. You can either select a particular website to display only event that occurred on the website, or you have the following extra options: • **(all)** - displays all events that occurred in the whole system.

	• **(only global events)** - displays only global, i.e. not website specific events. This option is only available in **Site Manager**. In **CMS Desk**, only events related to the current website are displayed in the log.
Clear log	Deletes all records currently displayed in the log.

The filter above the list enables you to display only records that match specified criteria. The **Display advanced filter** and **Display simplified filter** links switch the filter between simple and advanced mode, while the advanced one offers more filtering criteria to be specified.

The following details are displayed with each logged event. Some extra information is logged but not displayed in the grid. It can be viewed after clicking the **Display event** ( ) icon, which displays a pop-up window with full details about the event. More information can be found below.

Type	Type of the event. There are three types of events that can occur: • **Information** - standard event. • **Warning** - standard event with higher importance, e.g. application restart. • **Error** - critical error event, e.g. an unfinished operation, unhandled exception, etc.
Event time	Time when the event occurred.
Source	Source module where the event occurred.
Event code	Code of the event. These codes depend on the type of performed action.
User ID	ID of the user who performed the action that raised the event.
User name	Username of the user who performed the action that raised the event. If blank, the event was not raised by a user action, but was raised by the system itself.
IP address	IP address of the user who performed the action. If blank, the event was not raised by a user action, but was raised by the system itself.
Document name	Name of the document to which the event is related. If blank, the event was not document-related.
Site	Name of the website where the event occurred.
Machine name	Name of the server where the event occurred. Useful e.g. when running the system in a web farm.

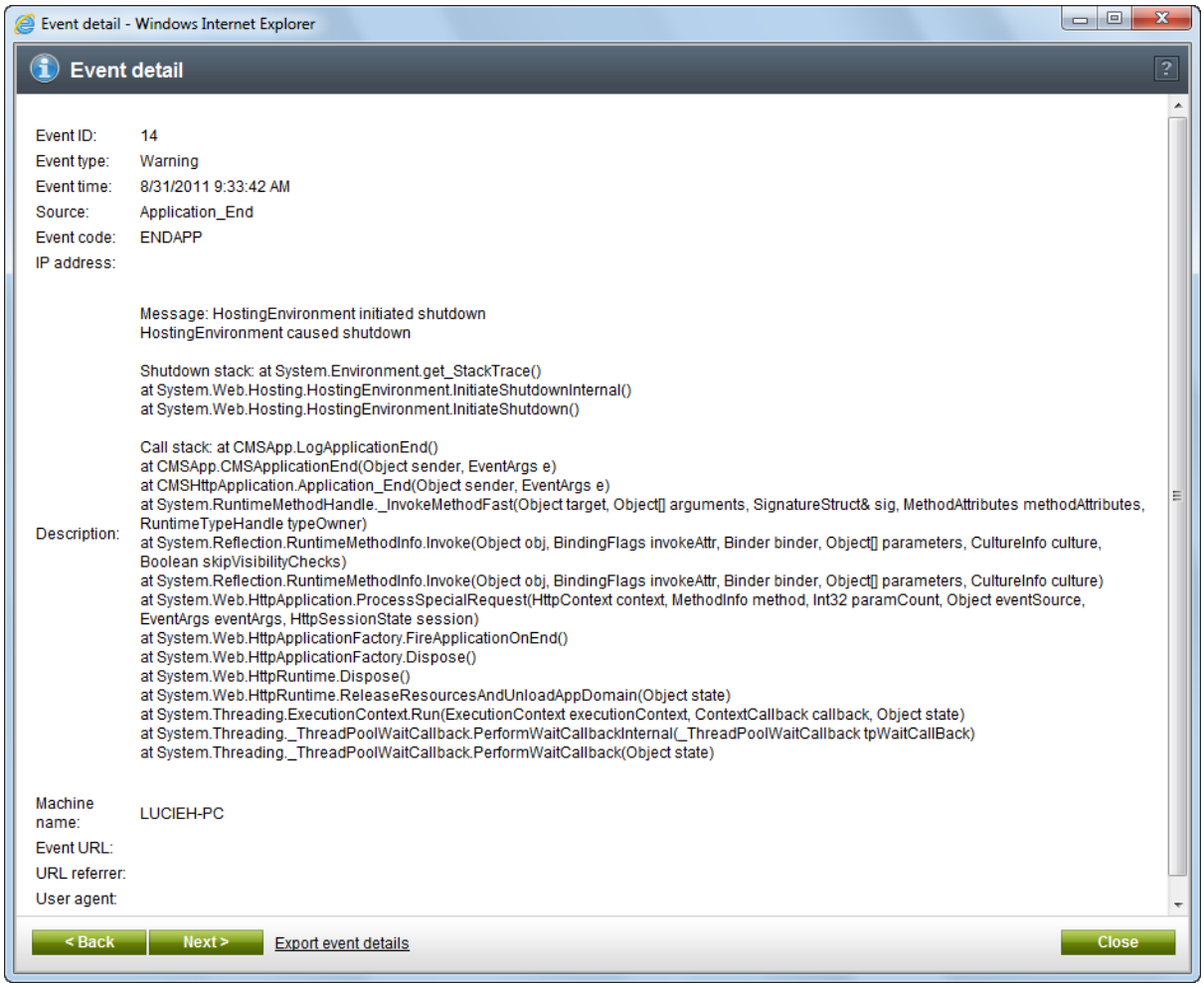


Event details

If you click the **Display event** ( ) icon in an event's row, a pop-up window with full details about the event is displayed. The following details are displayed in the window, while some of them may not be displayed if not applicable.

Event ID	Identifier of the event.
Event type	Type of the event. There are three types of events that can occur: • **Information** - standard event. • **Warning** - standard event with higher importance, e.g. application restart. • **Error** - critical error event, e.g. an unfinished operation, unhandled exception, etc.
Event time	Time when the event occurred.
Source	Source module where the event occurred.
Event code	Code of the event. These codes depend on the type of performed action.
User ID	ID of the user who performed the action that raised the event.
User name	Username of the user who performed the action that raised the event. If blank, the event was not raised by a user action, but was raised by the system itself.
IP address	IP address of the user who performed the action. If blank, the event was not raised by a user action, but was raised by the

	system itself.
Document name	Name of the document to which the event is related. If blank, the event was not document-related.
Description	Text describing the event.
Site name	Name of the website where the event occurred.
Machine name	Name of the server where the event occurred. Useful e.g. when running the system in a web farm.
Event URL	URL of the page where the event occurred.
URL referrer	URL of the page from which the event was raised.
User agent	User agent of the browser used when the event was raised.



8.23.3 Related settings

The following settings in **Site Manager -> Settings** are related to the Event log module:

Settings category	Description

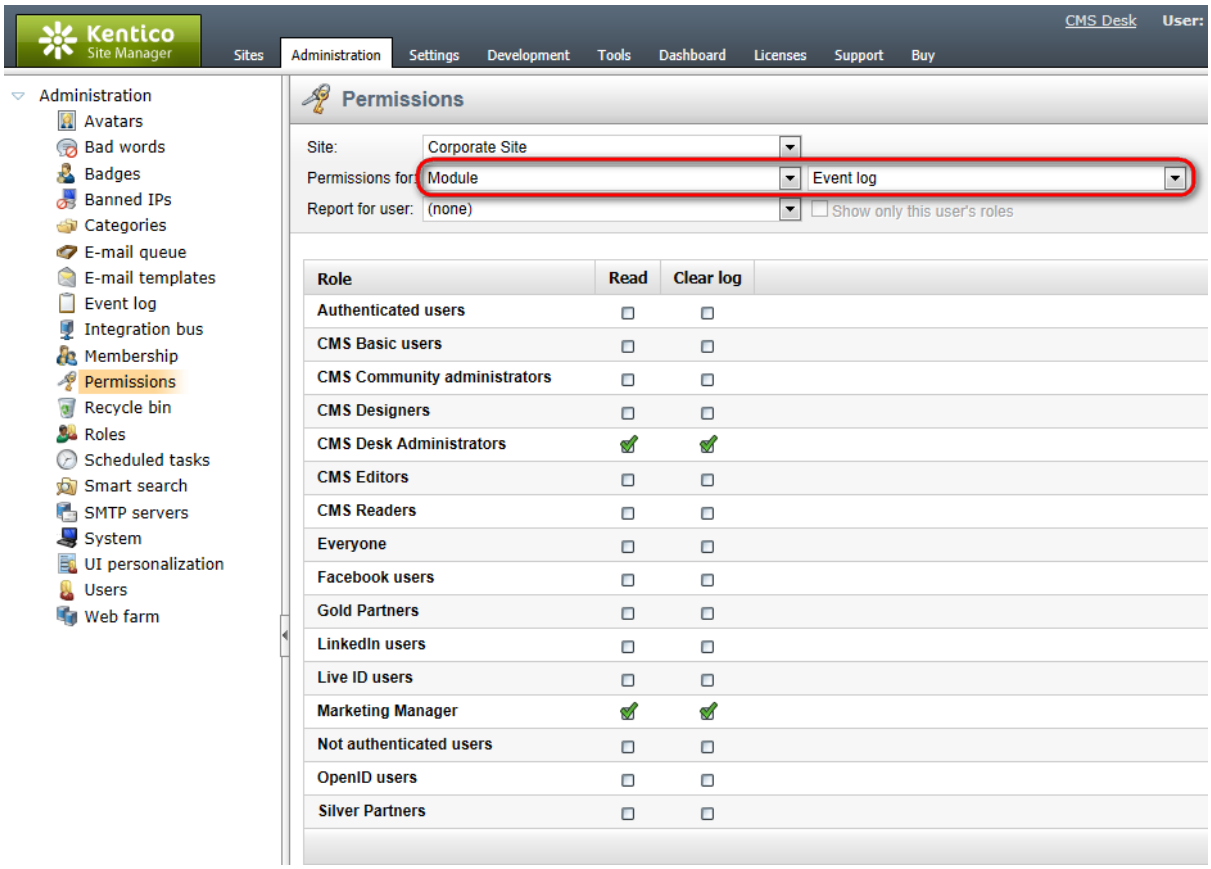
Content -> Log page not found exception	If enabled, page not found (404) exceptions will be logged as a warning in the event log.
System -> Event log size	Number of events stored in the event log. When exceeded by 10% (or a different percentage set by means of the *CMSLogKeepPercent* web.config key), the percentage of the oldest events is deleted from the log in a batch. If 0, no events are logged.
System -> Log metadata changes	If enabled, changes of object and document metadata (i.e. when an object or document is created, edited or deleted) are logged in the Event log.
System -> Error notification e-mail address	If an e-mail address is entered, e-mail notifications about errors logged in the event log will be sent to the address.

There are also several special keys that can be added to the *appSettings* section of the *web.config* file that provide an extra level of event logging settings in combination with the settings listed above. For more information on the keys, please refer to Appendix B - Web.config parameters.

8.23.4 Security

Access to the Event log and actions that can be performed in it can be restricted by means of permissions. There is a dedicated **Module -> Event log** permission matrix in the **Administration -> Permissions** section of **CMS Desk** and **Site Manager**. In the matrix, you can grant the following permissions to particular roles:

- **Read** - allows members of the role to view the event log and details of particular logged events.
- **Clear log** - allows members of the role to clear the event log (i.e. use the **Clear log** button).



8.23.5 Event log internals and API

8.23.5.1 Overview

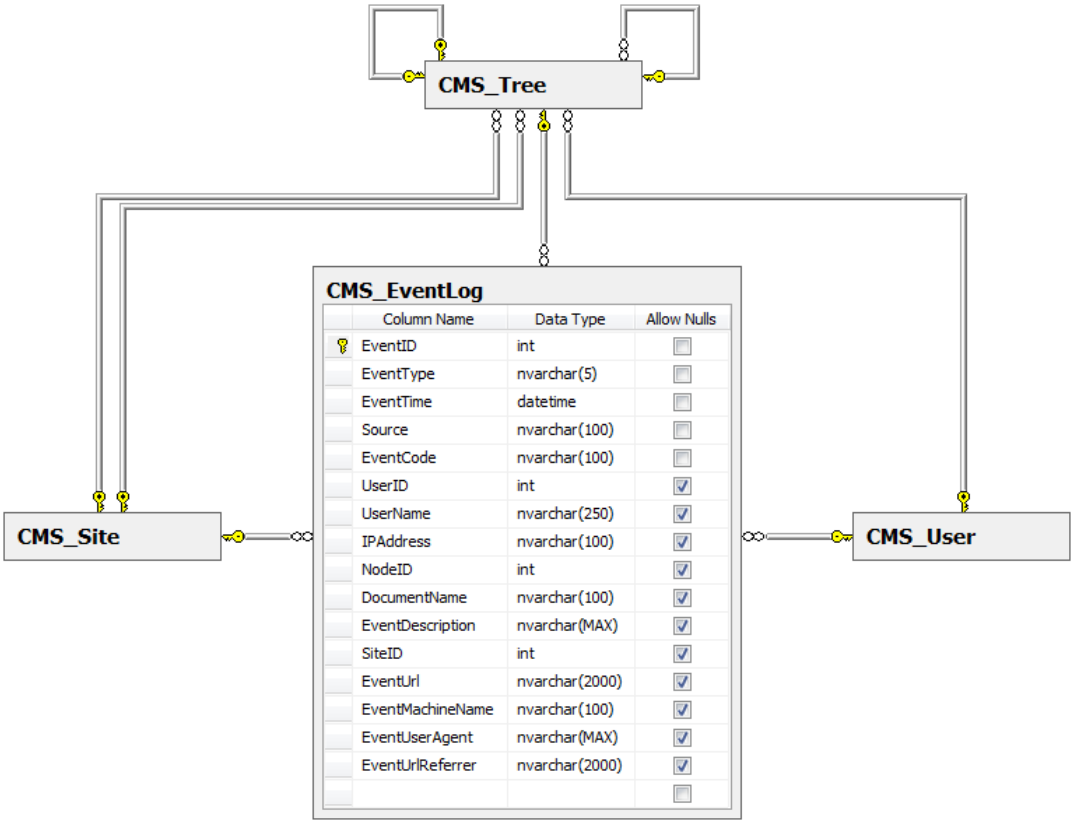
In this chapter, you will learn which database tables and API classes are used in the Events module. You will also see the most common API examples.

Advanced API examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all methods from the module classes, please refer to Kentico CMS API Reference.

8.23.5.2 Database tables

The Event log module uses the following database table:

- **CMS_EventLog** - contains records representing logged events.



8.23.5.3 API classes

If you are not familiar with the database table data management by Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

The classes of the Event log module use the **CMS.EventLog** namespace.

CMS_EventLog table API:

- **EventLogInfo** - represents one log event object.
- **EventLogInfoProvider** - provides functionality for management of log events.

8.23.5.4 API examples

8.23.5.4.1 Overview

These topics show examples of how the Event log module API can be used:

- Managing log events

Please note

All API examples can be simulated in the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Administration\EventLog\Default.aspx.cs***.

The Event log module API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.GlobalHelper;
using CMS.UIControls;
using CMS.CMSHelper;
using CMS.SiteProvider;
using CMS.EventLog;
```

8.23.5.4.2 Managing log events

The following example logs an event in the event log.

```
private bool LogEvent()
{
    // Create new event object
    EventLogInfo newEvent = new EventLogInfo();

    // Set the properties
    newEvent.EventType = "I";
    newEvent.EventDescription = "My new logged event.";
    newEvent.EventCode = "APIEXAMPLE";
    newEvent.EventTime = DateTime.Now;
    newEvent.Source = "API Example";
    newEvent.SiteID = CMSContext.CurrentSiteID;

    // Create new instance of event log provider
    EventLogProvider eventLog = new EventLogProvider();

    // Log the event
    eventLog.LogEvent(newEvent);

    return true;
}
```

The following example gets and updates the event created by the example above.

```
private bool GetAndUpdateEvent()
{
    // Create new instance of event log provider
    EventLogProvider eventLog = new EventLogProvider();

    // Get top 1 event matching the where condition
    string where = "EventCode = 'APIEXAMPLE'";
    int topN = 1;
    DataSet events = eventLog.GetAllEvents(where, null, topN, null);

    if (!DataHelper.DataSourceIsEmpty(events))
    {
        // Create the object from DataRow
        EventLogInfo updateEvent = new EventLogInfo(events.Tables[0].Rows[0]);

        // Update the properties
        updateEvent.EventDescription = updateEvent.EventDescription.ToLower();

        // Save the changes
        eventLog.SetEventLogInfo(updateEvent);

        return true;
    }

    return false;
}
```

The following example gets multiple events based on a WHERE condition and bulk updates them.

```
private bool GetAndBulkUpdateEvents()
{
    // Create new instance of event log provider
    EventLogProvider eventLog = new EventLogProvider();

    // Get events matching the where condition
    string where = "EventCode = 'APIEXAMPLE'";
    DataSet events = eventLog.GetAllEvents(where, null);

    if (!DataHelper.DataSourceIsEmpty(events))
    {
        // Loop through the individual items
        foreach (DataRow eventDr in events.Tables[0].Rows)
        {
            // Create the object from DataRow
            EventLogInfo updateEvent = new EventLogInfo(eventDr);

            // Update the properties
            updateEvent.EventDescription = updateEvent.EventDescription.ToUpper();

            // Save the changes
            eventLog.SetEventLogInfo(updateEvent);
        }
```

```
        return true;
    }

    return false;
}
```

The following example clears the event log.

```
private bool ClearLog()
{
    // Create new instance of event log provider
    EventLogProvider eventLog = new EventLogProvider();

    // Clear event log for current site
    eventLog.ClearEventLog(CMSContext.CurrentUser.UserID,
CMSContext.CurrentUser.UserName, HTTPHelper.GetUserHostAddress(),
CMSContext.CurrentSiteID);

    return true;
}
```

8.24 File import

8.24.1 Overview

The file import module allows you to import files including their entire folder structure from the disk to the Kentico CMS content repository (content tree). When files are uploaded this way, they are stored as **CMS.File** (or **CMS.Folder**) documents. The module allows large amounts of files to be selected, which eliminates the need to create documents and upload files in the content tree manually one-by-one.

The module's user interface can be found in **CMS Desk -> Tools -> File import**. The label at the top shows the path to the file import folder, which is where all files that you wish to import must be located. If the folder doesn't exist, you may need to create it on the disk. The default file import folder is **~/CMSImportFiles**. You can define your custom file import path in **Site Manager -> Settings -> System -> Files -> File import folder**.

For a more specific example of using this module to import files, please continue in the Importing files topic.

The Security topic shows how permissions can be set for this module. The use of this module should usually only be allowed for site administrators, as it doesn't check certain security settings from **Site Manager -> Settings -> System -> Files**, such as allowed file extensions.

The file import module isn't the only way files can be uploaded to the Kentico CMS system. For other options and information about general file management, please refer to the Content management -> File Management chapter of this guide.

8.24.2 Importing files

The following is a step-by-step guide through the process of importing files:

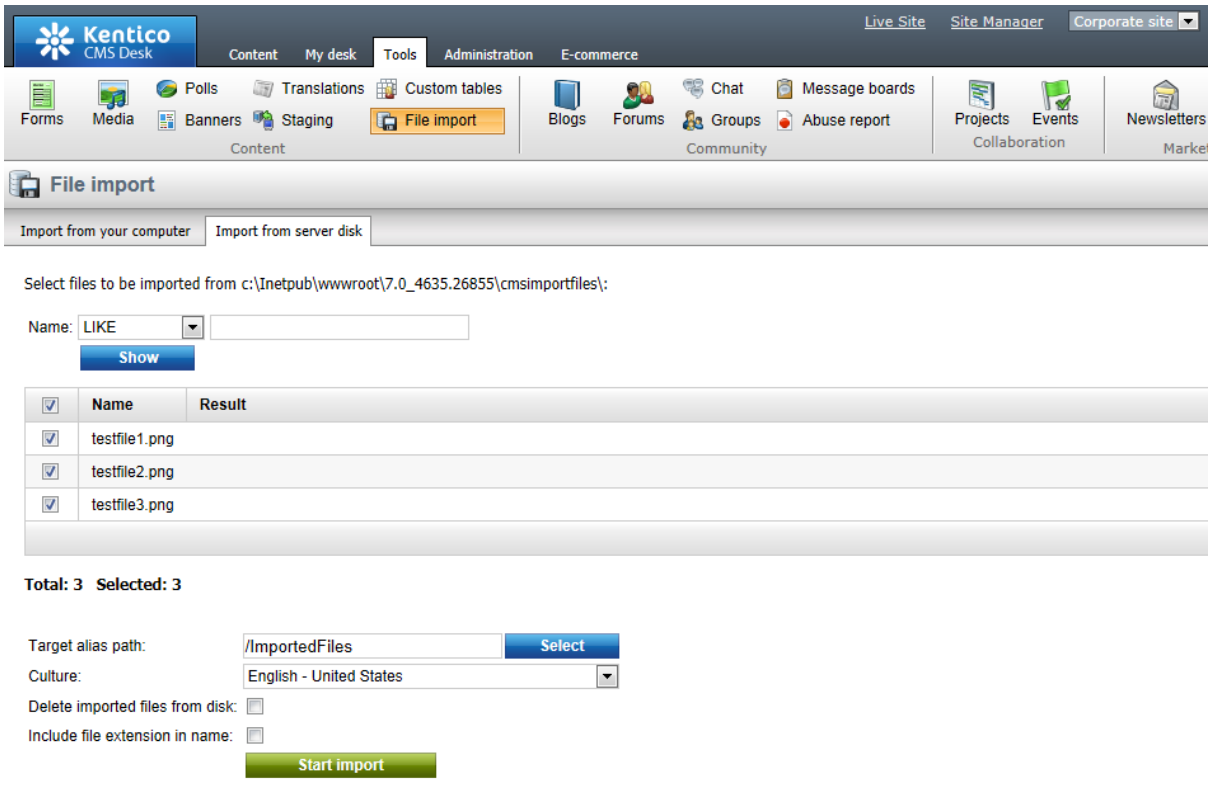
1. Copy your files into the specified import folder.

2. Go to **CMS Desk -> Tools -> File import**. You should see a list of files that are currently in the import folder.

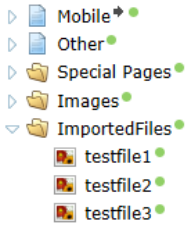
3. Choose which files should be imported using the checkbox next to each file and specify the following properties:

- **Target alias path**: path within the content tree where the files will be imported
- **Culture**: culture to which the uploaded files (documents) will be assigned
- **Delete imported files from disk**: if enabled, files in the import folder will be deleted after a successful import
- **Include file extension in name**: if enabled, the file extension is included in the *Document name* of the newly imported files

Click **Start import**.



4. Now if you switch to the **Content** tab and locate the alias path that you specified in the previous step, you should see the files uploaded.



Please note

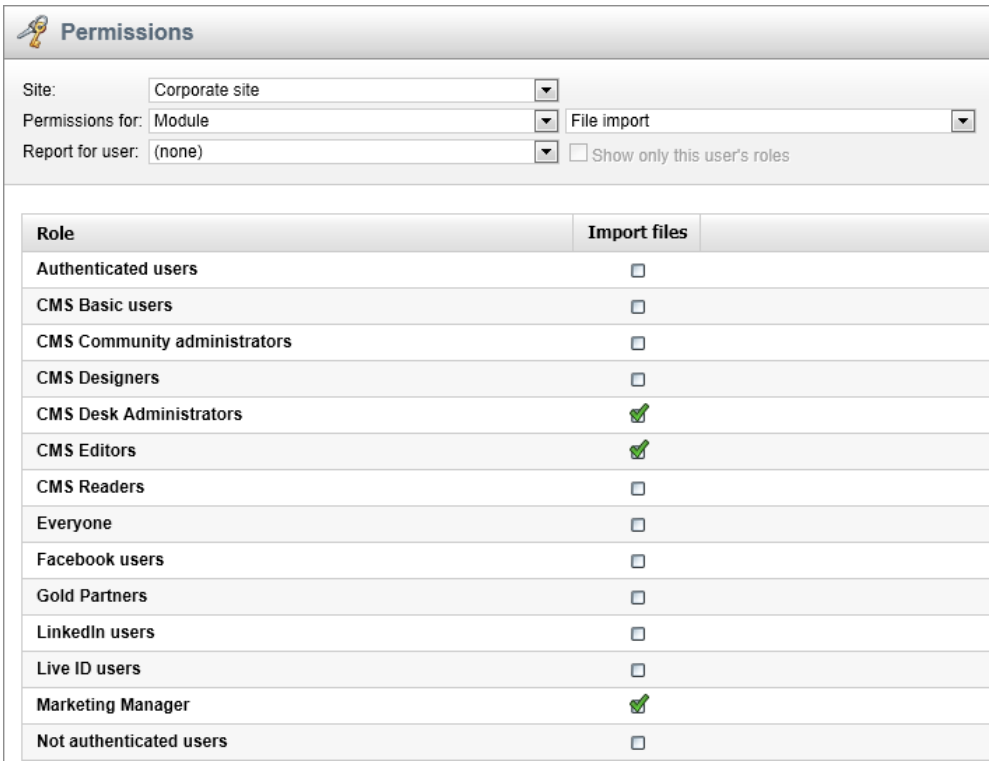
If you are importing files from your computer (on the **Import from your computer** tab), you need to:

1. Click the **Select files** button to specify the files that you wish to import.
2. Click the **Upload** button to upload the selected files to the Kentico CMS content repository (content tree).

8.24.3 Security

Access rights to the file import module can be configured in **CMS Site Manager -> Administration -> Permissions**, after you select the **Modules -> File import** permission matrix. This can also be done in **CMS Desk -> Administration -> Permissions**, but only for the current site. The File import module only has one single permission:

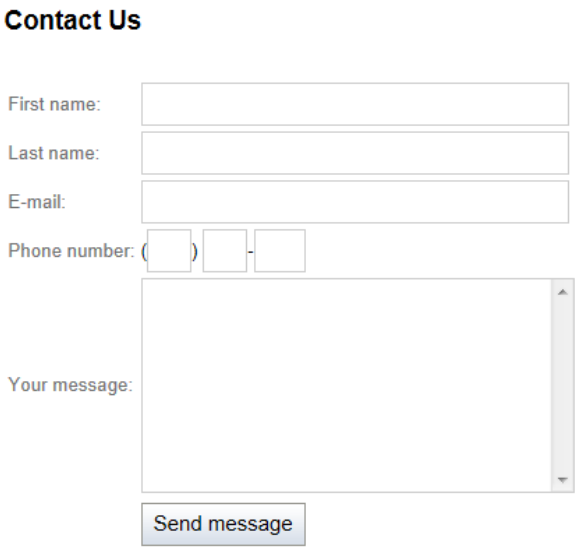
- **Import files** - members of the specified roles are allowed to import files using the File import module



8.25 Forms

8.25.1 Overview

The Forms module enables non-technical users (content editors) to easily create and publish on-line forms. These forms can be used to gather structured data from website users. A typical example of such form is the **Contact Us** form, which can be found on the **Company** page of the sample Corporate Site. The form is depicted in the screenshot below.



The administration interface of the module is located in **CMS Desk -> Tools -> Forms**. This is where new forms can be created, as described in the Creating a new form topic. Once a form is created via the user interface, it can be added to a live site page. This can be done either via the WYSIWYG editor, or by means of the **On-line form** web part or widget. Both options are described in the Displaying a form on the live site topic.

Each form has its own separate database table where submitted data is stored. The submitted data can be viewed and managed in **CMS Desk -> Tools -> Forms**. Data records can even be exported to a **Microsoft Excel** spreadsheet in this part of the user interface. Please refer to the Managing form data topic for more information on which actions can be performed with the submitted data.

You may want to notify the person who is responsible for management of forms data when a new record is submitted. For this purpose, you can configure the form to send automatic notification e-mails about new records. The user who submitted the new record may also be notified that the record was submitted successfully and that it will be processed soon. This can be achieved by configuring autoresponder e-mails. Both options are described in the Notification and autoresponder e-mails sub-chapter. Macros can be used largely with notification and autoresponder e-mails, as well as when you want to localize field captions in the form. Please refer to the Using macros with forms topic to learn more.

Access to forms may be restricted, so that only members of selected roles can perform certain operations with the forms themselves or with the data in them. The Security topic gives you an overview of how to configure forms permissions both globally and separately for each particular form.

Layout of each form is fully customizable. The Defining custom form layout topic explains how it can be achieved. You may also customize behavior of the forms module with your custom code. These possibilities are explained in the Forms internals and API sub-chapter, along with an overview of where form records are stored and which API classes contain methods for their management. It also provides a series of code examples showing how methods from these classes may be used in your custom code.

8.25.2 Creating a new form

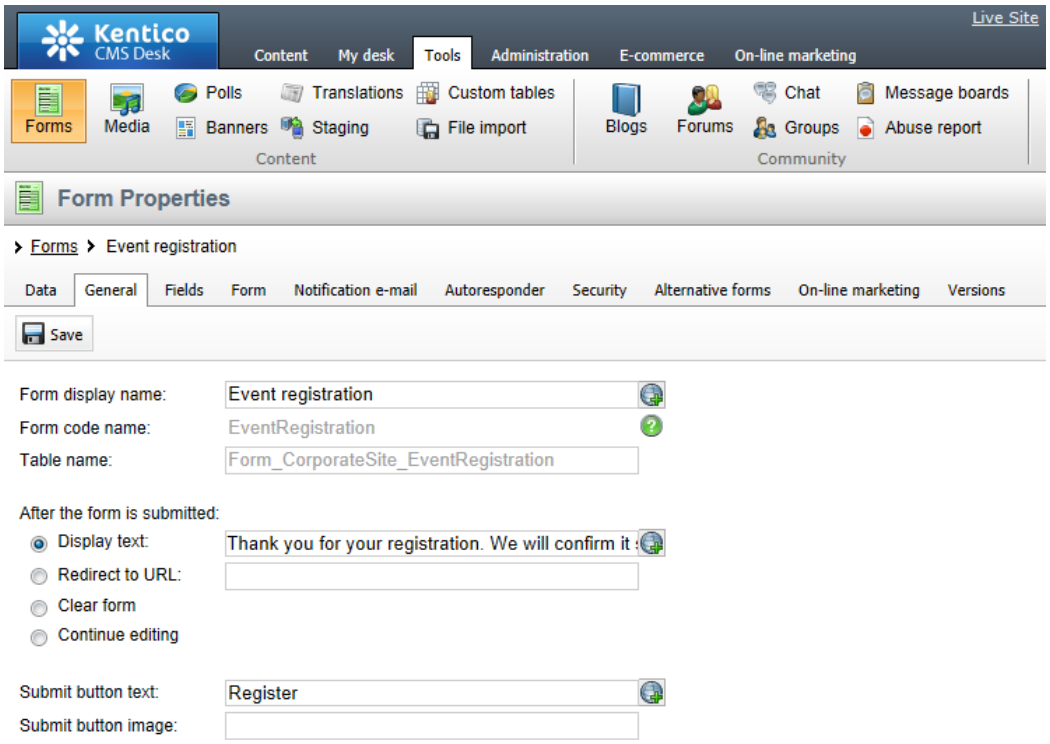
In this topics, we will create a new sample form via the Forms module's user interface. This topic doesn't explain all options that are available in the user interface. For a detailed description of each option, please refer to the built-in context help, which is accessible by clicking the ? icon in the top-right corner of the user interface.

1. Go to **CMS Desk -> Tools -> Forms** and click the **New form** button. Type *Event registration* into the **Form display name** field.

Click **Save**.

2. You will be redirected to the **General** tab of the new form's editing interface. Specify the following values:

- **After the form is submitted**: Select the **Display text** option and enter the following text into the field next to it: *Thank you for your registration. We will confirm it shortly by e-mail.*
- **Submit button text:** Register



Click **Save**.

3. Now we will define the form's fields. Go to the **Fields** tab. Add the following fields using the **Add attribute** (+) button. For each field, enter the values shown below, click **Save** and repeat the procedure until you have all the listed fields defined.

- **Column name**: FirstName
- **Show on public form**: enabled
- **Field caption**: First name
- **Field type**: TextBox
- **Maximum length**: 100
- **Allow empty value**: disabled

- **Column name**: LastName
- **Show on public form**: enabled
- **Field caption**: Last name
- **Field type**: TextBox
- **Maximum length**: 100
- **Allow empty value**: disabled

- **Column name**: Phone
- **Show on public form**: enabled
- **Field caption**: Phone
- **Field type**: U.S. phone number
- **Maximum length**: 14
- **Allow empty value**: enabled

- **Column name**: Email
- **Show on public form**: enabled

- **Field caption**: E-mail
- **Field type**: E-mail
- **Maximum length**: 100
- **Allow empty value**: disabled

- **Column name**: Presentations
- **Show on public form**: enabled
- **Field caption**: Presentations you want to visit
- **Field type**: Multiple choice
- **Editing control settings -> Data source**: Select the **Options** radio button and enter the following into the textbox below:
  ASP.NET;ASP.NET
  ATLAS;ATLAS
  WPF;Windows Presentation Foundation
- **Allow empty value**: enabled

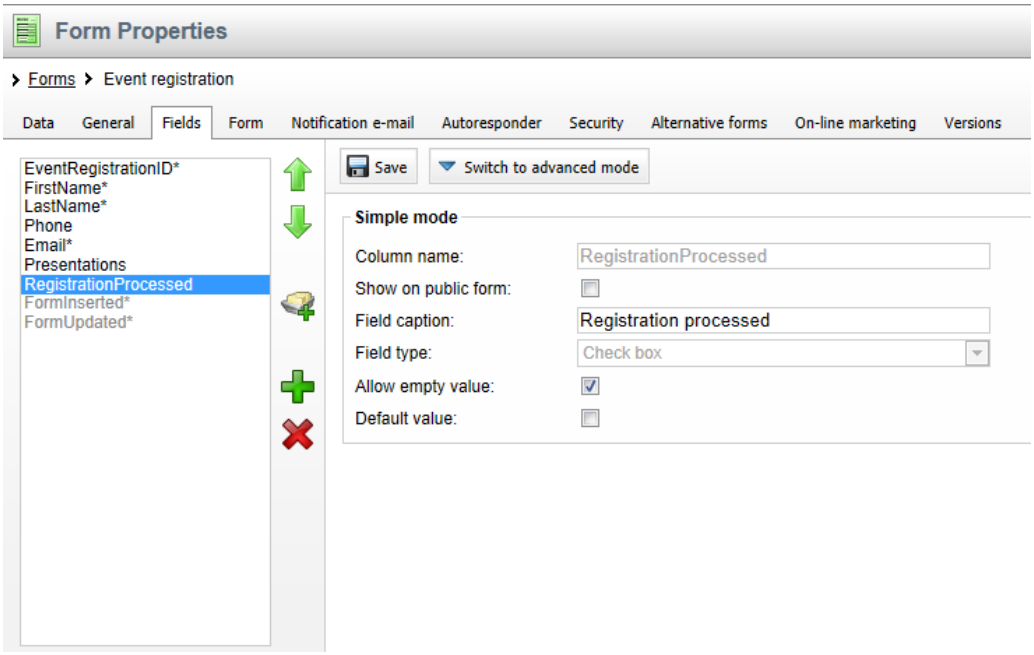
Each form field and its functionality is based on an object called a *form control*, which is selected through the **Field type**. This gives you almost unlimited form flexibility and customization options.

When editing the properties of a field, you can switch between **Simple** and **Advanced** mode using the appropriate button at the top of the dialog. In advanced mode, the full configuration options of the field editor are available. Simple mode offers a more straightforward editing dialog and is recommended for non-technical users. In simple mode, only form controls that are configured to have **High priority** are offered among the **Field type** options.

Please see the Development -> Form controls chapter for detailed information about form fields and their types.

4. The last item will be used only by site editors to mark the processed registration forms. This will be done exclusively via the user interface. Therefore, it has the **Show on public form** property disabled.

- **Column name**: RegistrationProcessed
- **Show on public form**: disabled
- **Field caption**: Registration processed
- **Field type**: Check box
- **Allow empty value**: enabled



5. With all the fields defined, the form is ready to be published on the live site. This can be done several different ways, all of which are described in the Displaying a form on the live site topic.

Please note

If you click the **Switch to advanced mode** link below the field editor, an extended user interface will be displayed. There you can configure additional options such as input validation rules or CSS classes used for the fields.

8.25.3 Displaying a form on the live site

In the Creating a new form topic, you have learned how to create a new form via the module's user interface. This topic will give you an overview of how you can add the form to the live site.

Content editors can add a form to any page with editable text regions. It can be done either using the **Insert Form** ( ) button on the WYSIWYG editor toolbar. A form can also be added directly into text by typing a macro in the following format: *%%control:BizFormControl?BizFormCodeName%%*. The *BizFormCodeName* part of the macro needs to be replaced with the code name of the particular form.

Website designers or developers can add a form to any web part or widget zone by adding the **On-line form** web part or widget. In this case, code name of the form needs to be entered in the **Form name** property of the web part or widget.

The examples below explain both options of adding forms to your live site pages.

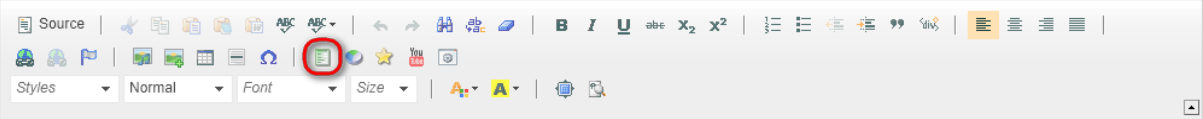
Example 1: Adding a form via the WYSIWYG editor

In this example, we will add the *Event registration* form, created in the Creating a new form topic, to a

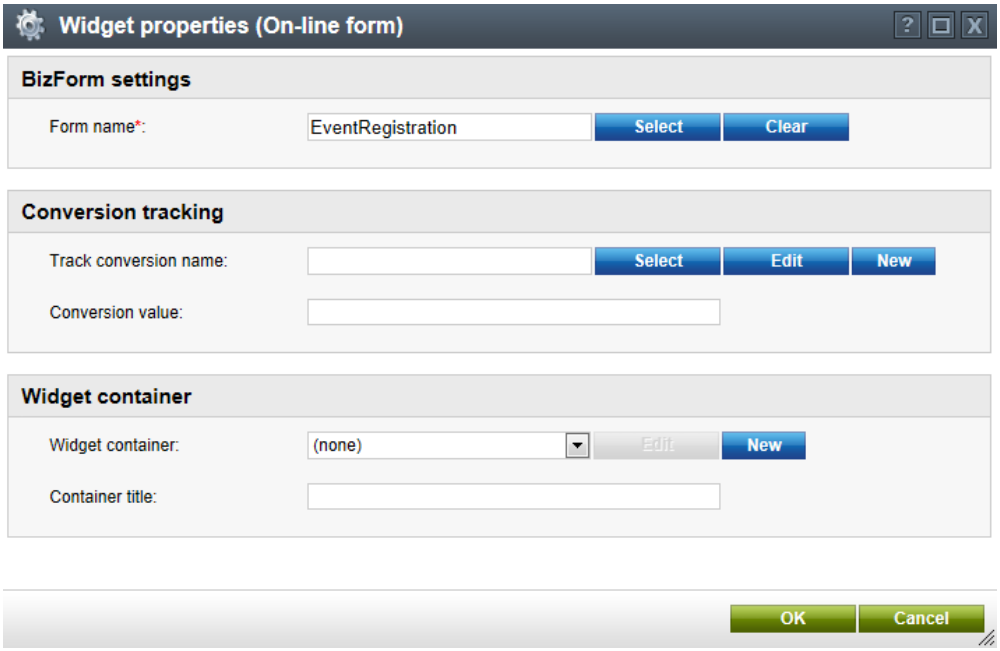
page using the **Insert Form** ( ) button on the WYSIWYG editor toolbar. Any other form can be added to a page exactly the same way.

1. Go to **CMS Desk -> Content** and select some page with editable regions (the **Editable text** web part). View the page in **Edit** mode on the **Page** tab.

2. Click the **Insert Form** ( ) button on the WYSIWYG editor toolbar, which will create an On-line form inline widget.



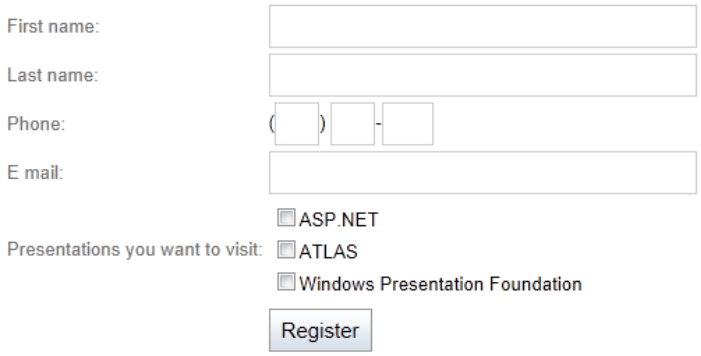
3. The configuration dialog of the widget will be opened. Click the **Select** button next to the **Form name** field and choose the **Event registration** form.



Click **OK**.

A placeholder image will be pasted into the text. The properties of the On-line form inline widget can be edited at any time by double clicking this image.

4. Click **Save** and switch to the **Live site** mode. You will see the form on the page.



5. Try entering some values and submitting the filled-in form. After doing that, you may proceed to the Managing form data topic, where management of submitted data is explained.

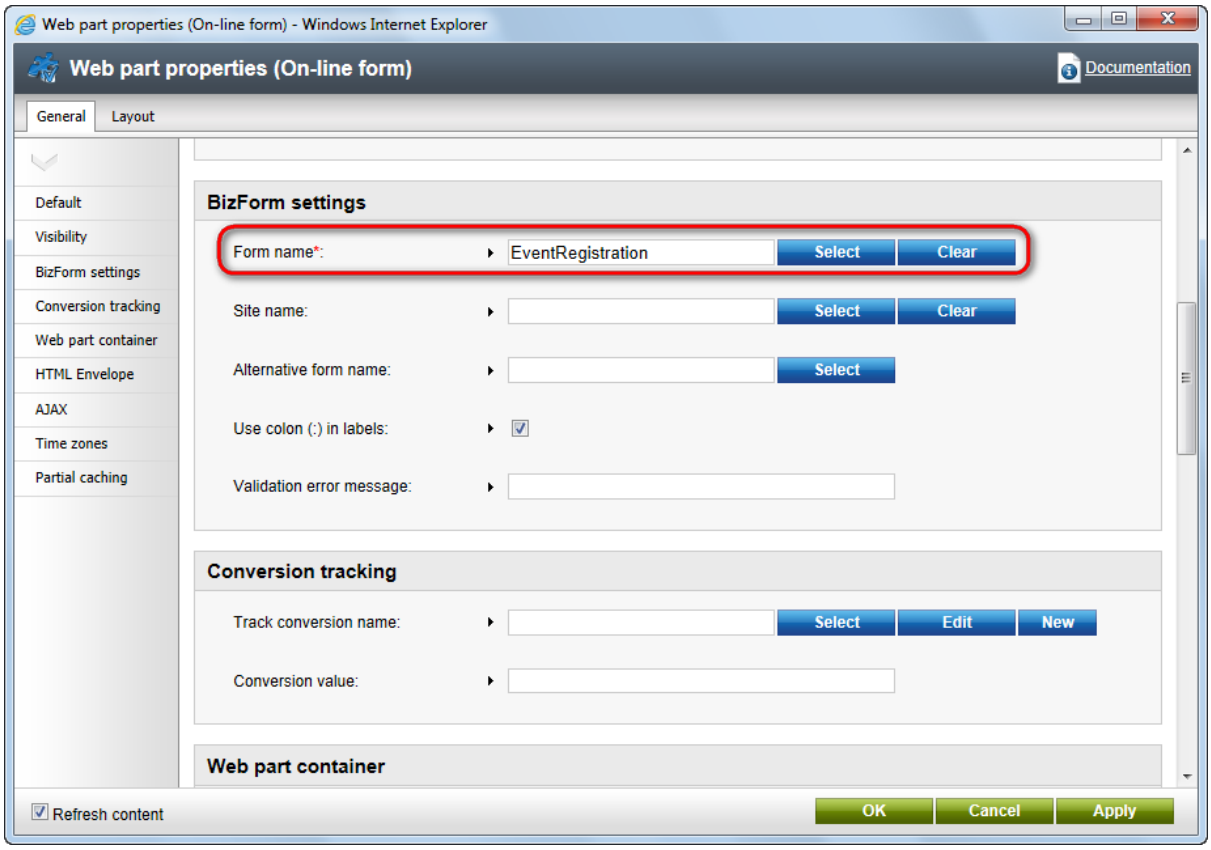
Example 2: Adding a form using the On-line form web part

This example demonstrates how to add the *Event registration* form, created in the Creating a new form topic, to a page using the **On-line form** web part. Any other form can be added to a page exactly the same way. The same applies to using the **On-line form** widget - the only difference is that the widget can only be added to a widget zone, not a web part zone.

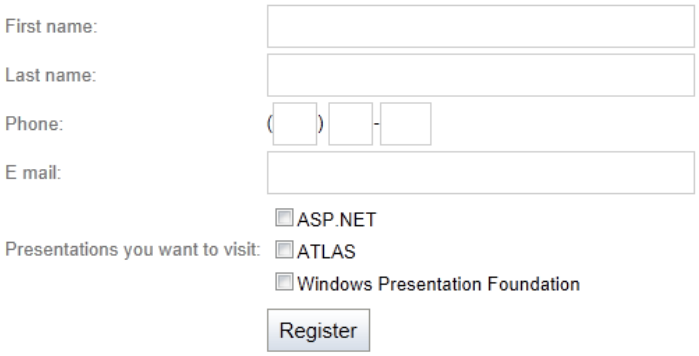
1. Go to **CMS Desk -> Content** and select ant page with a web part zone. View the page in the **Edit -> Design** mode.

2. Click the **Add web part** ( ) button at the top-right corner of the zone and choose the **Forms -> On-line form** web part. Click **OK**.

3. In the web part properties dialog, you only need to enter *EventRegistration* (the code name of the form) into the **Form name** property and click **OK**.



4. Click **Save** and switch to the **Live site** mode. You will see the form on the page.



5. Try entering some values and submitting the filled-in form. After doing that, you may proceed to the Managing form data topic, where management of submitted data is explained.

8.25.4 Managing form data

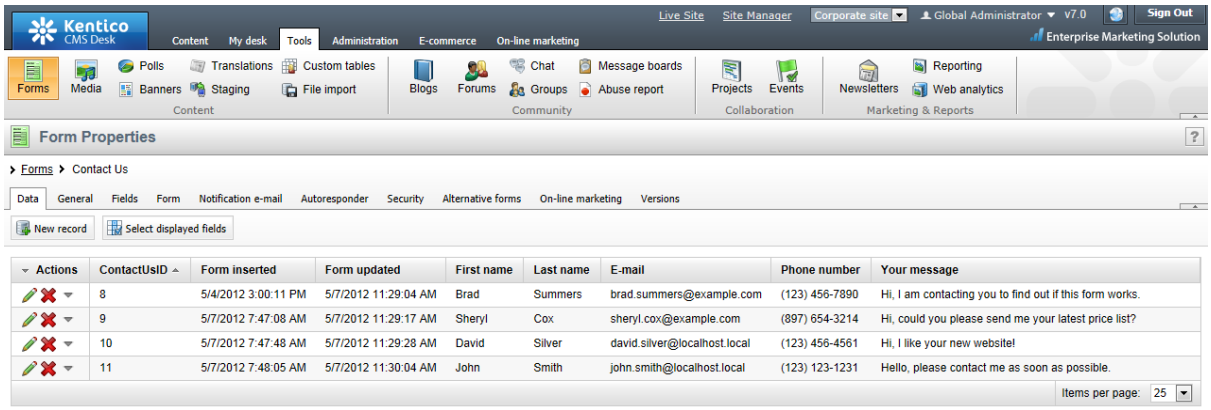
Form records (data submitted by website users) can be viewed and managed in **CMS Desk -> Tools -> Forms**. In the list of forms, you need to click the **Edit** (🖉) icon next to a particular form. You will get redirected to the **Data** tab of the form's editing interface, where particular records are listed.

Each of the records listed in the grid has the following actions available:

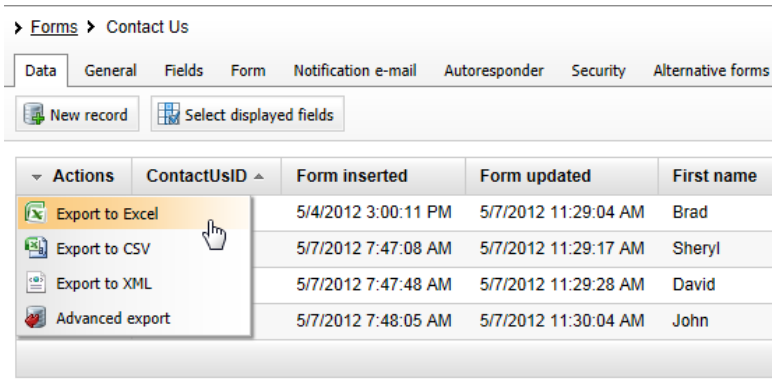
- **Edit record ( )** - allows you to alter the record.
- **Delete record ( )** - allows you to delete the record.

You can also perform the following actions using the buttons in the tab's header:

- **New record** - used to manually add records to the current form.
- **Select displayed fields** - enables you to select which of the form's fields should be displayed in the grid.



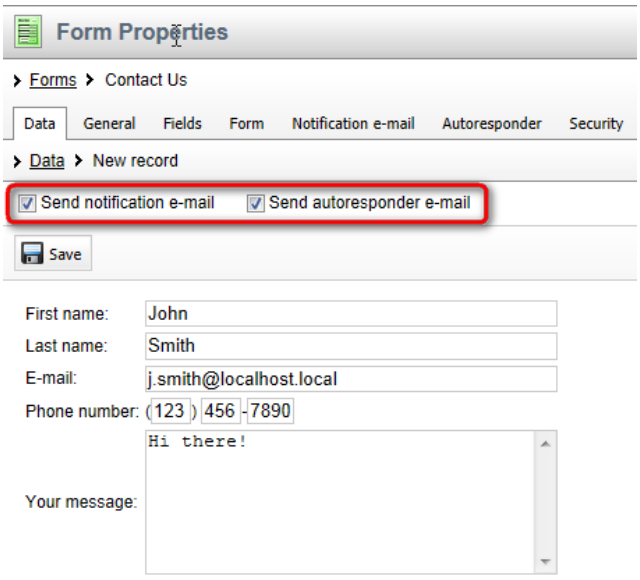
A form's records may be exported into an external file using either the XLSX (Excel), CSV or XML format. This can be done by clicking the ▾ icon in the header of the **Actions** column in the grid and then selecting the appropriate option from the context menu.



The **Advanced export** option opens a dialog where you can specify which rows and columns should be included in the exported data. For detailed information about exporting data to files, please refer to the Modules -> UI data export chapter.

In cases where a large number of records is displayed, you can create a filter for limiting which records should be displayed. To learn more about this possibility, please refer to Modules -> Alternative forms -> Creating filter forms.

When editing ( ) or creating ( ) a record via the administration interface, the form is displayed the same way as on the live site, letting you enter or change the values. Additionally, you can decide if notification and autoresponder e-mails should be sent when you save the record.



8.25.5 Notification and autoresponder e-mails

8.25.5.1 Notification and autoresponder e-mails

The Forms module allows you to send two types of e-mails automatically when a new record is added:

- Notification e-mail - e-mails notifying the person responsible for form data management (content editor, administrator, ...) about the new submitted record.
- Autoresponder - e-mail to the person who submitted the new record, typically confirming that the record has been received and will be processed.

When a record is added on the live site, the e-mails are sent out based on settings described in the following topics. When a record is added or modified via the administration interface, the person who is entering the record can decide whether these e-mails will be sent, as described in Managing form data.

8.25.5.2 Notification e-mails

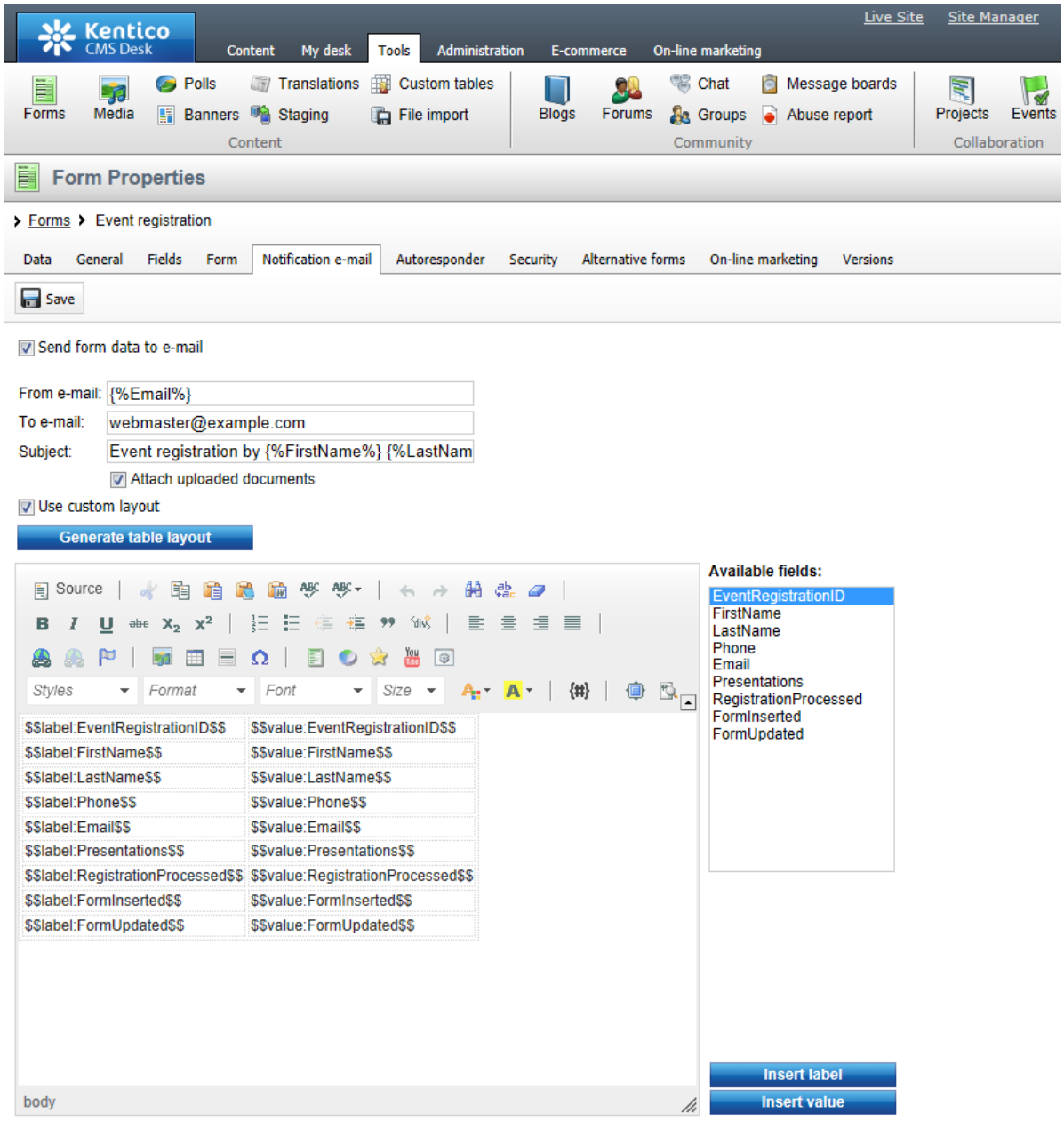
Notification e-mails can be configured on the **Notification e-mail** tab of a form's editing interface. To enable them, you first need to enable the **Send form data to e-mail** option. Then you need to configure the following settings:

- **From e-mail -** the e-mail address from which the notifications will be sent. A typical option is to use the address of the user who submitted the form. This can be done by entering a macro expression, which retrieves the value from the field in the form where users enter their e-mail address, as described in the Using macros with forms topic.
- **To e-mail -** e-mail address where notification e-mails should be sent. Typically the address of the person responsible for management of form records. You can specify multiple addresses separated by semicolons.
- **Subject** - subject of the notification e-mails.
- **Attach uploaded documents** - enable this option if you want to attach files submitted via the form (if there are any) to the notification e-mails.
- **Use custom layout** - if disabled, the body of the notification e-mails will contain all field names with the entered values, each on a single line. If enabled, you can define a custom layout for the

notifications in the text area displayed below.

- **Generate table layout** - this button generates a table containing all field names in the left column and their values in the right column. Macros for particular field names and values can also be entered separately by selecting a field from the *Available fields* listbox and clicking the *Insert label* or *Insert value* buttons.

Macros may be utilized in the values of any of these fields. Once all the options are configured, click **Save** to save your configuration.



8.25.5.3 Autoresponder e-mails

Autoresponder e-mails can be configured on the **Autoresponder** tab of a form's editing interface.

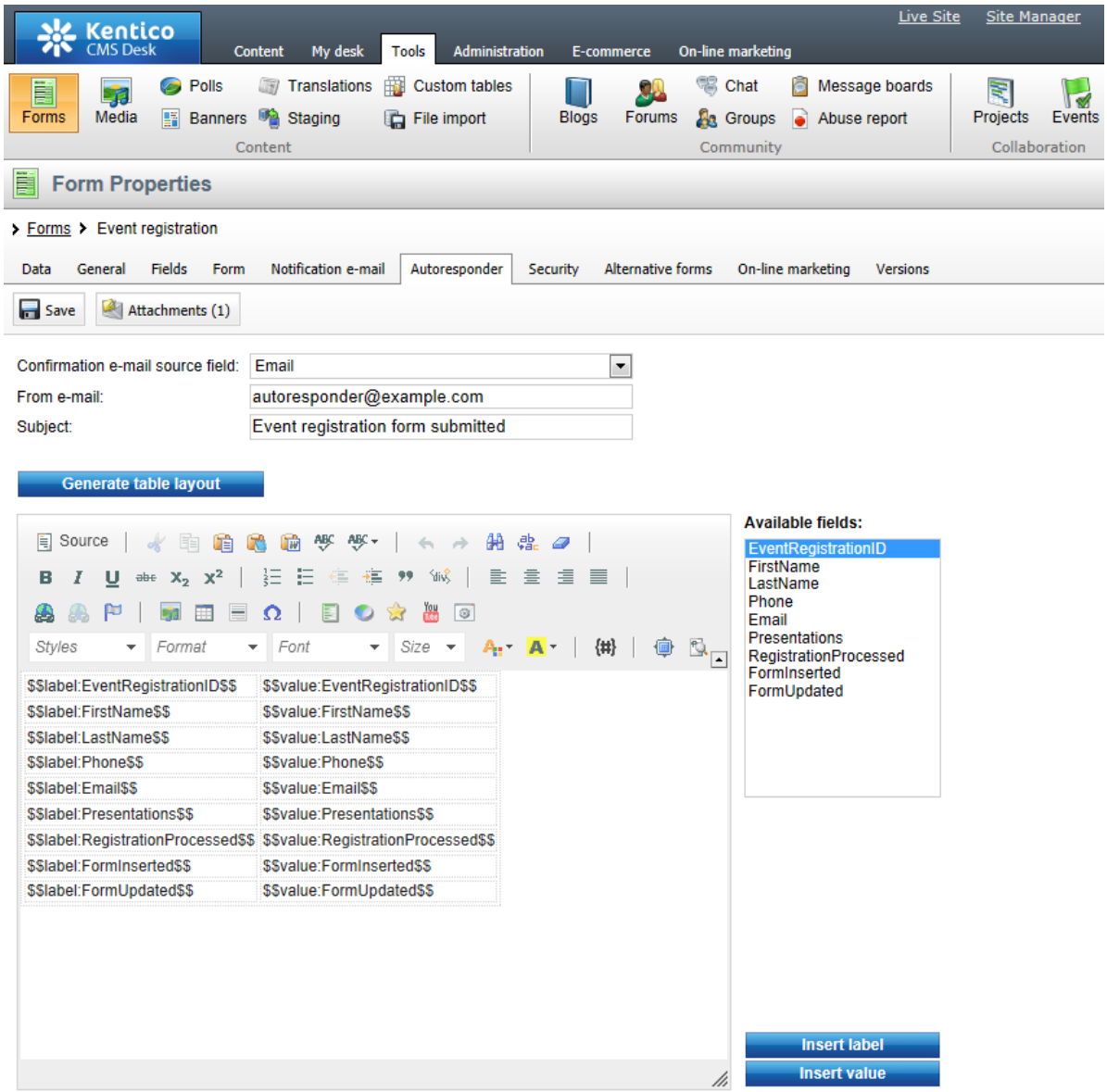
When you navigate to the tab for the first time, only the **Confirmation e-mail source field** drop-down list will be visible. Expand it and choose the form field where users will enter their e-mail address, so that the system knows where to send the automatic response. This means that autoresponder e-mails can only be used with forms where such a field is present (however, this should be the case with most real-world on-line forms).

Once a field is chosen, the following additional options will be displayed:

- **From e-mail** - e-mail address from which the autoresponder e-mails will be sent (the *From* field of the e-mail message).
- **Subject** - subject of the autoresponder e-mails.

You can then define the body content of the e-mails through the text area and WYSIWYG editor. The **Generate table layout** button generates a table with all field names in the left column and their values in the right column. Particular field names and their values can also be entered separately by selecting a field from the **Available fields** listbox and clicking the **Insert label** or **Insert value** buttons

Macros may be utilized in the values of these fields, as described in the Using macros with forms topic.



The **Attachments** button in the header of the tab allows you to add files to the autoresponder e-mail, such as a detailed event agenda, white papers, etc. You can see the current number of attachments in the button's caption. Clicking the button opens a dialog where you can manage the attachments. The **Paste to text** action may be used to insert image attachments directly into the body of the e-mail (to the current cursor position).

When everything is configured as required, click **Save** to save the autoresponder settings.

8.25.6 Defining custom form layout

If you're not satisfied with the standard table layout of the form, the forms module allows you to define custom form layout. This can be achieved on the **Form** tab of a form's editing interface in **CMS Desk -> Tools -> Forms**.

Example

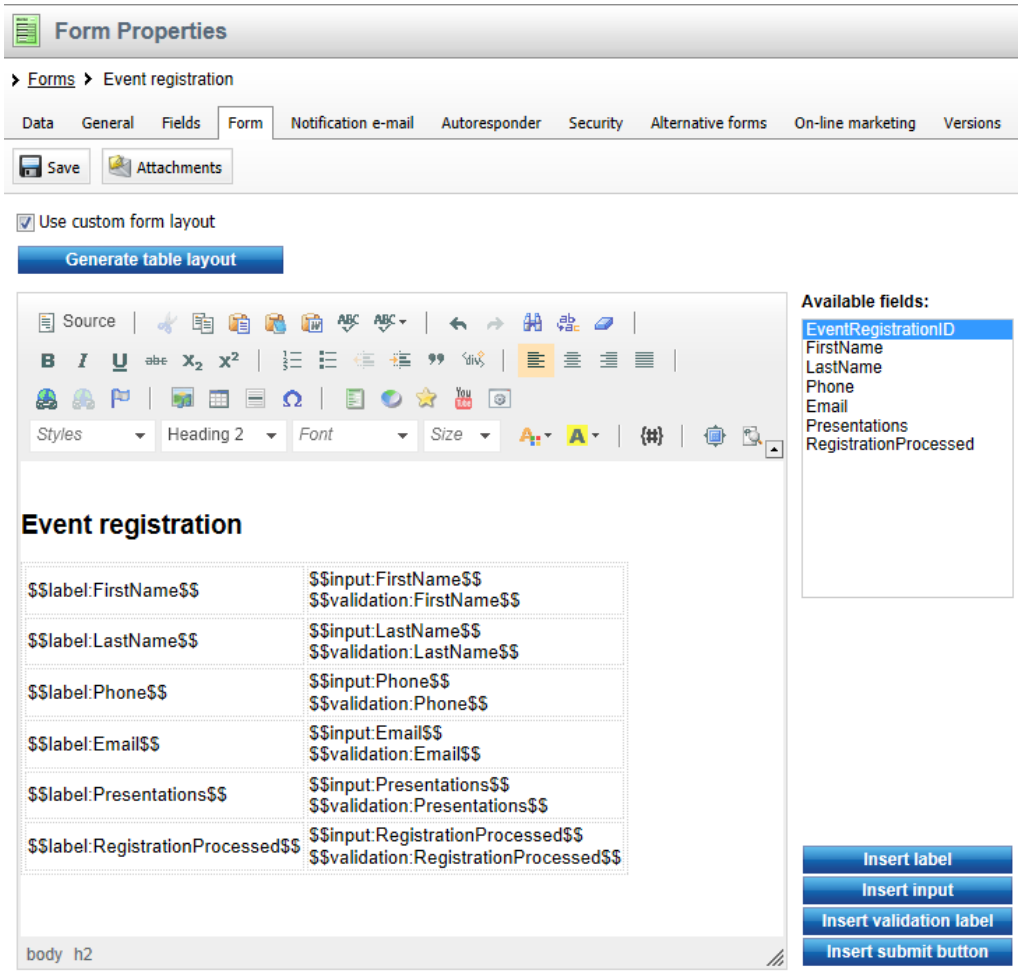
To modify the default layout of the sample *Event registration* form created in the Creating a new form topic, please take the following steps:

1. Go to **CMS Desk -> Tools -> Forms** and click the **Edit** () icon next to the form.

2. In its editing interface, go to the **Form** tab and check the **Use custom form layout** checkbox.

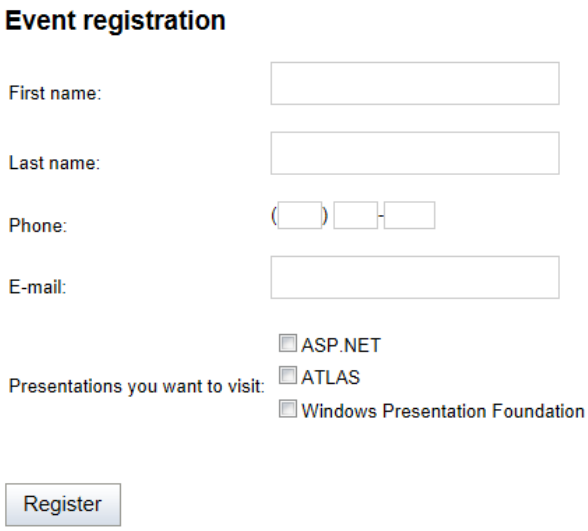
3. Click the **Generate table layout** button. This will generate a table containing all fields of the form, with appropriate labels, input controls and validation controls. Alternatively, you may use the **Insert label**, **Insert input** and **Insert validation label** buttons on the right to achieve the same result. This gives you the option to include just the fields that you want to appear in the form.

4. With the table generated, you may try altering the appearance of the form using actions on the WYSIWYG editor toolbar. For the purpose of this example, add a heading saying *Event registration* above the form, select it and apply the **Heading 2** format through the WYSIWYG editor toolbar. You may also delete the first line of the table (*EventRegistrationID*), which does not need to be displayed on the public form. The result should look as in the screenshot below.



5. If you wish to include images in the layout, you can upload the files by clicking the **Attachments** button in the tab's header. The form's attachments can then be pasted into the layout.

6. Click **Save** to save the new layout. Now if you add the form to the live site, it should look as in the following screenshot:



8.25.7 Using macros with forms

Values of the fields that you need to fill in when configuring a form can be obtained dynamically from the submitted forms. The following table shows which macro types can be used for particular values of form configuration.

	Localization	Context	Query String	Cookie	Custom	Path	Context (Data) macros
General tab - **Form display name**	•						
General tab - **Display text**	•						•
General tab - **Redirect to URL**		•					•
General tab - **Submit button text**	•						
Fields tab **- Default value**	•	•	•	•	•	•	
Fields tab **- Field caption**	•						
Fields tab **- Table layout**	•						
Notification e-mail tab - all fields	•	•	•	•	•		•
Autoresponder tabs - all fields	•	•	•	•	•		•

Context (data) macros

You can use values from current context when configuring a form. This can be achieved using a data macro in the **{%column_name%}** format. The *column_name* part of the macro is the value of the **Column name** property of a particular form filed. When the form is submitted, the macros are

automatically resolved and replaced with particular data from the form.

These macros can be used in the following fields:

- **General tab**
  - Display text
  - Redirect to URL
- **Notification e-mails**
  - From e-mail
  - To e-mail
  - Subject
  - E-mail body
- **Autoresponder e-mails**
  - From e-mail
  - Subject
  - E-mail body

Example 1

When configuring notification e-mails for the sample form created in the Creating a new form topic, you may use the following values:

- **From e-mail:** *{%Email%}*;
- **Subject:** *Event registration by {%FirstName%} {%LastName%}*;

This will result in the e-mail having the user's e-mail address as the sender address, which enables the recipient to easily reply to the e-mail. The subject of the e-mail will have the first and last name of the user, which will help identifying the sender of the e-mail.

Example 2

If you entered the **Display text** value on the **General** tab of the same form like this:

- *Dear {%FirstName%}, thank you for your message. We will contact you shortly.*

The text will be resolved as the following when "Jane" was entered in the **First name** field by the user:

- *Dear Jane, thank you for your message. We will contact you shortly.*

Form text localization

If you need to display the form on a multi-lingual website, you can localize field captions and other text strings using localization macros, e.g.:

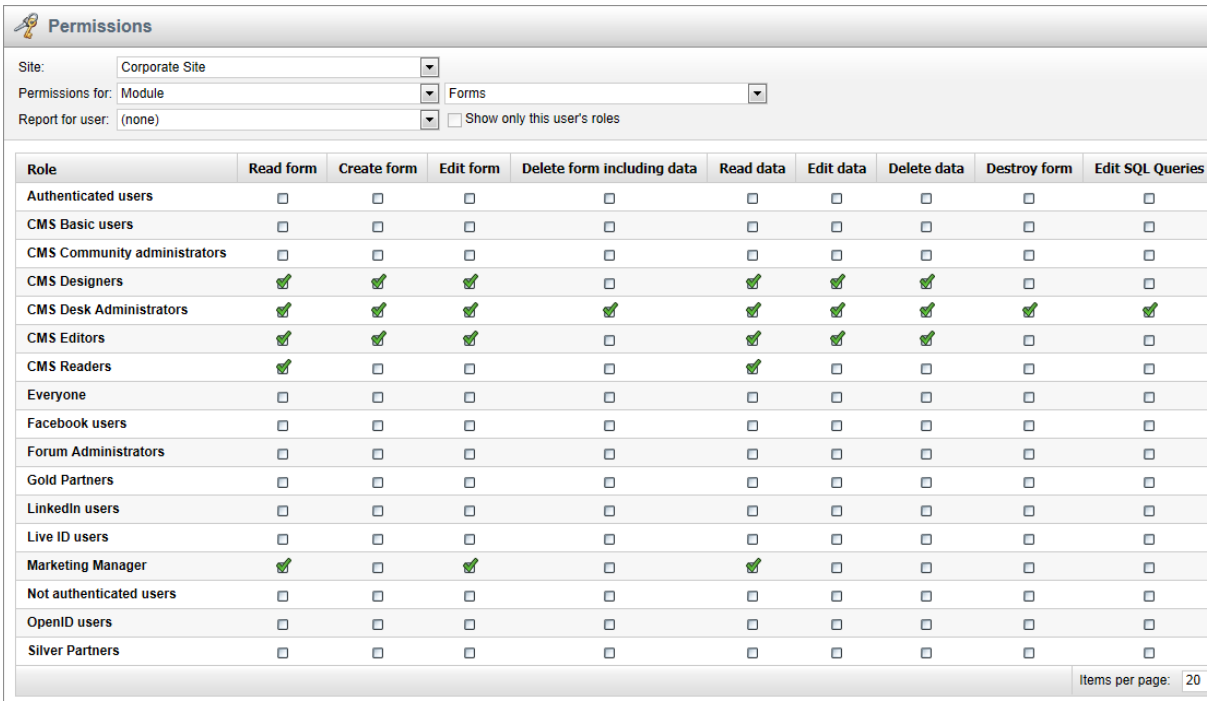
- {$myform.fullname$}
- {$=Hello|de-de=Hallo|it-it=Ciao$}

You can find more details in Development -> Multilingual and international support -> Localization expressions.
Detailed overview of all macros in Kentico CMS can be found in Development -> Macro expressions.

8.25.8 Security

Access to the Forms module can be managed in **CMS Desk -> Administration -> Permissions**, after you select the **Module -> Forms** permission matrix. The Forms module has the following permissions:

- **Read form** - members of the roles are allowed to view form configuration, fields and layout (not the actual records).
- **Create form** - members of the roles are allowed to create new forms.
- **Edit form** - members of the roles are allowed to edit form configuration, fields and layout (not the actual records).
- **Delete form including data** - members of the roles are allowed to delete forms including stored records.
- **Read data** - members of the roles are allowed to view form records.
- **Edit data** - members of the roles are allowed to create and edit form records
- **Delete data** - members of the roles are allowed to delete existing form records
- **Destroy form** - members of the roles are allowed to delete the version history of forms.
- **Edit SQL Queries** - some types of fields (form controls) offer the possibility of specifying an SQL query that will be used to retrieve the offered options. Users who belong to the specified roles will be allowed to write the code of these queries (please note that this can be a security risk).

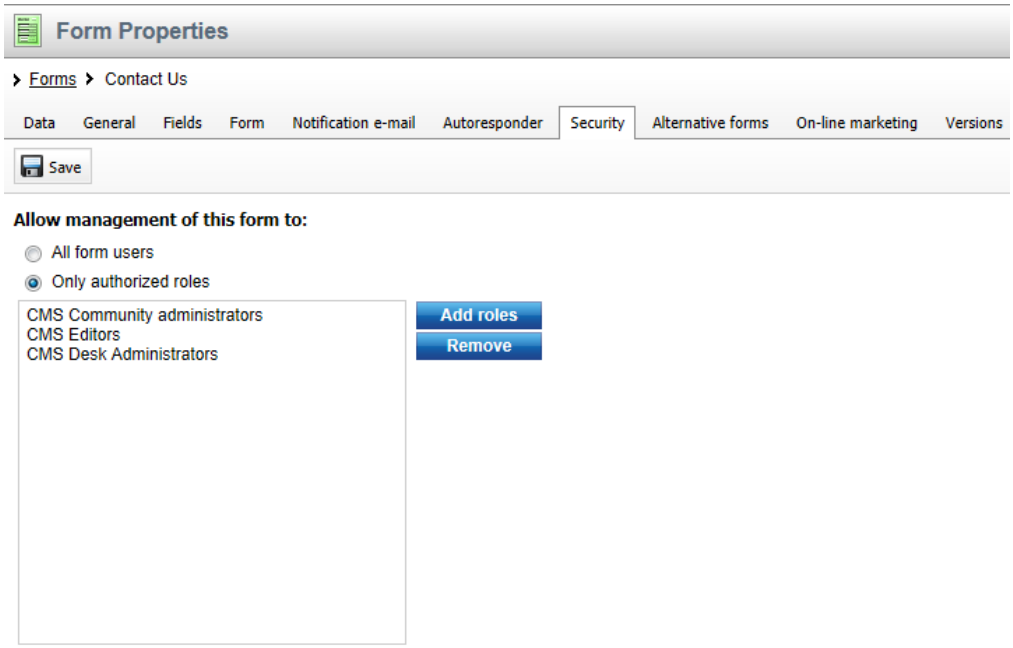


Security for individual forms

The roles which are authorized to read and modify a form and its data can also be specified for individual forms. To do this, edit ( ) a particular form in **CMS Desk -> Tools -> Forms** and switch to its **Security** tab. The following two options are available:

- **All form users** - all users with access to the *Tools -> Forms* section will be allowed to manage the form.
- **Only authorized roles** - only members of the roles added to the box will be allowed to manage the form.

Please note: General module permissions for the Forms module (described above) must be granted to the role first. Then, you can further customize access to particular forms using the form-level settings. The fact that a role has permissions to access a particular form is not sufficient — the form-level settings only define if the particular form will be listed in **Tools -> Forms**.



8.25.9 Form file settings

Files may be submitted as part of form records, typically when a form contains an uploader type field (specifically the *Upload file* field type/form control). In **Site Manager -> Settings -> System -> Files**, you can adjust the following settings related to files submitted through forms and their storage in the file system:

Storage	
Custom form files folder	Folder where files uploaded via forms are stored. You can use: • **physical disk path**: e.g. c:\myfiles\ • **virtual path**: ~/UploadedFiles • **UNC path**: \\server\folder If no value is entered, the files are stored under the *~/<site code name>/BizFormFiles/* folder.
Use site-specific subfolders for custom form files folder	This setting is only applied when a **Custom form files folder** is configured. If enabled, attachment files will be stored in a sub-folder named according to the code name of the site where the form is placed, i.e. *<custom form files folder>/<site code name>*. It is useful for better orientation in files when multiple sites are running in the system.
Security	

Upload extensions	Specifies which extensions are allowed for uploaded files in general, including forms. You can restrict the types of uploaded files by entering a limited list of extensions separated by semicolons, for example: *gif; jpg;doc;pdf* This allows you to block users from uploading potentially dangerous files, such as ASPX scripts. If no value is specified, uploading will be allowed for all file types. Allowed extensions can also be set for individual uploader form fields. Each field of this type has an **Extensions** setting, which can be configured on the **Fields** tab of the given form. You may either have the field inherit from the site settings or specify a different list of extensions.

8.25.10 Forms internals and API

8.25.10.1 Database tables and API classes

The Forms module uses the following database tables:

- **Form_<site code name>_<table name>** - each form has its own database table. for example: *BizForm_CorporateSite_EventRegistration*. The table contains all fields that you specified on the form's **Fields** tab and you can modify stored data using direct access to the table (there are no dependencies).
- **CMS_Class** - each form has an associated record in the CMS_Class table. You can recognize form records by the *BizForm.* prefix in the *ClassName* column.

The Forms API is provided by the following classes from the **CMS.FormEngine** namespace:

- **BizFormInfo**, **BizFormInfoProvider** - these classes provide functionality for managing forms.

The following topics show examples of how these classes can be used:

- Creating a new record
- Updating a record
- Deleting a record
- Customization possibilities

The API programming and Kentico CMS internals section of this guide contains more API related information, so please refer to it if required.

For detailed API documentation, such as a list of all methods from the classes above, please refer to Kentico CMS API Reference.

8.25.10.2 Creating a new record

The sample code below demonstrates how to add a new record to a form using the API.

[C#]

```
using CMS.SettingsProvider;
```

```
using CMS.FormEngine;

...

string bizFormName = "ContactUs";
string siteName = "CorporateSite";

// Read From definition
BizFormInfo bfi = BizFormInfoProvider.GetBizFormInfo(bizFormName, siteName);

if (bfi != null)
{
    // Read data type definition
    DataClassInfo dci = DataClassInfoProvider.GetDataClass(bfi.FormClassID);

    if (dci != null)
    {
        BizFormItemProvider bProvider = new BizFormItemProvider();
        // Create a new record in memory
        BizFormItem formRecord = BizFormItem.New(dci.ClassName, bProvider);

        // Insert some data
        formRecord.SetValue("FirstName", "Alice");
        formRecord.SetValue("LastName", "Cooper");
        formRecord.SetValue("Email", "alice@email.com");
        formRecord.SetValue("Message", "Hello world");
        formRecord.SetValue("PhoneNumber", "(123) 123-4567");
        formRecord.SetValue("FormInserted", DateTime.Now);
        formRecord.SetValue("FormUpdated", DateTime.Now);

        // Insert the new record in the database
        formRecord.Insert();

        // Update number of entries in BizFormInfo
        BizFormInfoProvider.RefreshDataCount(bfi.FormName, bfi.FormSiteID);
    }
}
```

8.25.10.3 Updating a record

This code example shows how an existing form record can be modified.

[C#]

```
using System.Data;

using CMS.FormEngine;
using CMS.SettingsProvider;
using CMS.DataEngine;
using CMS.GlobalHelper;

...

        string bizFormName = "ContactUs";
```

```
        string siteName = "CorporateSite";

        // Read Form definition
        BizFormInfo bfi = BizFormInfoProvider.GetBizFormInfo(bizFormName,
siteName);

        if (bfi != null)
        {
            // Read data type definition
            DataClassInfo dci = DataClassInfoProvider.GetDataClass
(bfi.FormClassID);

            if (dci != null)
            {
                // Get all Form data
                GeneralConnection genConn = ConnectionHelper.GetConnection();
                DataSet ds = genConn.ExecuteQuery(dci.ClassName + ".selectall",
null, null, null);

                if (!DataHelper.DataSourceIsEmpty(ds))
                {
                    // Get ID of the first record
                    int formRecordID = ValidationHelper.GetInteger(ds.Tables[0]
.Rows[0][0], 0);

                    BizFormItemProvider bProvider = new BizFormItemProvider();

                    // Get the record with ID of the first row record
                    BizFormItem formRecord = bProvider.GetItem(formRecordID,
dci.ClassName);

                    if (formRecord != null)
                    {
                        // Set new field values
                        formRecord.SetValue("FirstName", "Bob");
                        formRecord.SetValue("LastName", "Marley");
                        formRecord.SetValue("Email", "bob@email.com");
                        formRecord.SetValue("Message", "Good job:)");
                        formRecord.SetValue("FormUpdated", DateTime.Now);

                        // Save updates in the database
                        formRecord.Update();

                        lblInfo.Text = "The first record was updated
successfully.";
                    }
                }
                else
                {
                    lblInfo.Text = "No data found.";
                }
            }
        }
```

8.25.10.4 Deleting a record

This code example shows how a form record can be deleted.

[C#]

```
using System.Data;

using CMS.FormEngine;
using CMS.SettingsProvider;
using CMS.DataEngine;
using CMS.GlobalHelper;

...

        string bizFormName = "ContactUs";
        string siteName = "CorporateSite";

        // Get Form definition
        BizFormInfo bfi = BizFormInfoProvider.GetBizFormInfo(bizFormName,
siteName);

        if (bfi != null)
        {
            // Get data type definition
            DataClassInfo dci = DataClassInfoProvider.GetDataClass
(bfi.FormClassID);
            if (dci != null)
            {
                // Get all Form data
                GeneralConnection genConn = ConnectionHelper.GetConnection();
                DataSet ds = genConn.ExecuteQuery(dci.ClassName + ".selectall",
null, null, null);

                if (!DataHelper.DataSourceIsEmpty(ds))
                {
                  // Get ID of the first record
                  int formRecordID = ValidationHelper.GetInteger(ds.Tables[0].Rows
[0][0], 0);

                  BizFormItemProvider bProvider = new BizFormItemProvider();
                  // Get the record with specified ID
                  BizFormItem formRecord = bProvider.GetItem(formRecordID,
dci.ClassName);

                  if (formRecord != null)
                  {
                      // Delete the first record
                      formRecord.Delete();

                      // Update number of entries in BizFormInfo
                      BizFormInfoProvider.RefreshDataCount(bfi.FormName,
bfi.FormSiteID);

                      lblInfo.Text = "The record was deleted successfully.";
                  }
                }
```

```
                else
                {
                    lblInfo.Text = "No data found.";
                }
            }
        }
```

8.25.10.5 Customization possibilities

This topic provides information on how forms behavior can be modified with your custom code.

Event Handling

You can run custom actions when a form record is created, updated or deleted. There are two ways how you can handle these events:

1. Handling BizForm control events

In this case, you can place the **BizForm** control into the markup of a user control or web part and specify the code name of the required Form in its **FormName** property. Then, you can handle the BizForm events, such as **OnAfterSave**, **OnBeforeValidate**, etc.

In these handlers, you can retrieve the values of form fields using the **BasicForm.GetFieldValue(string fieldName)** method, which may be called through the corresponding property of the BizForm control. For example:

[C#]

```
BizForm.BasicForm.GetFieldValue("FirstName");
```

The data of the entire form may be accessed as a data container through the **BizForm.BasicForm.Data** property. This allows you to insert values into fields programatically using the **SetValue(string fieldName, object value)** method. For example:

[C#]

```
BizForm.BasicForm.Data.SetValue("TimeStamp", DateTime.Now);
```

Please note that field values need to be set at the appropriate time in the BizForm control's life cycle. A suitable place is in the handler of the **OnBeforeSave** event.

If you wish to set up special behavior in the form during the editing process, individual controls that make up the fields, labels and validation error messages in the form can be accessed through the hash tables provided by the **BizForm.BasicForm.FieldControls**, **BizForm.BasicForm.FieldLabels** and **BizForm.BasicForm.FieldErrorLabels** properties.

If necessary, you can stop further processing (saving) of the form at any point by setting the **BizForm.StopProcessing** property to *true*.

An example of this type of customization can be found in Development -> Web parts -> Modifying web parts -> Modifying code of standard web parts.

2. Handling global data events

You can write a custom data handler as described in the API programming and Kentico CMS internals -> Data handler (CustomDataHandler class) topic. This type of handler will allow you to run custom code whenever a database record is inserted, updated or deleted. You can use the *dataItem.ClassName* property to check if the record being updated matches the code name of your form (you can find it in the **Form code name** field on the **General** tab of a form's editing interface).

Custom field controls

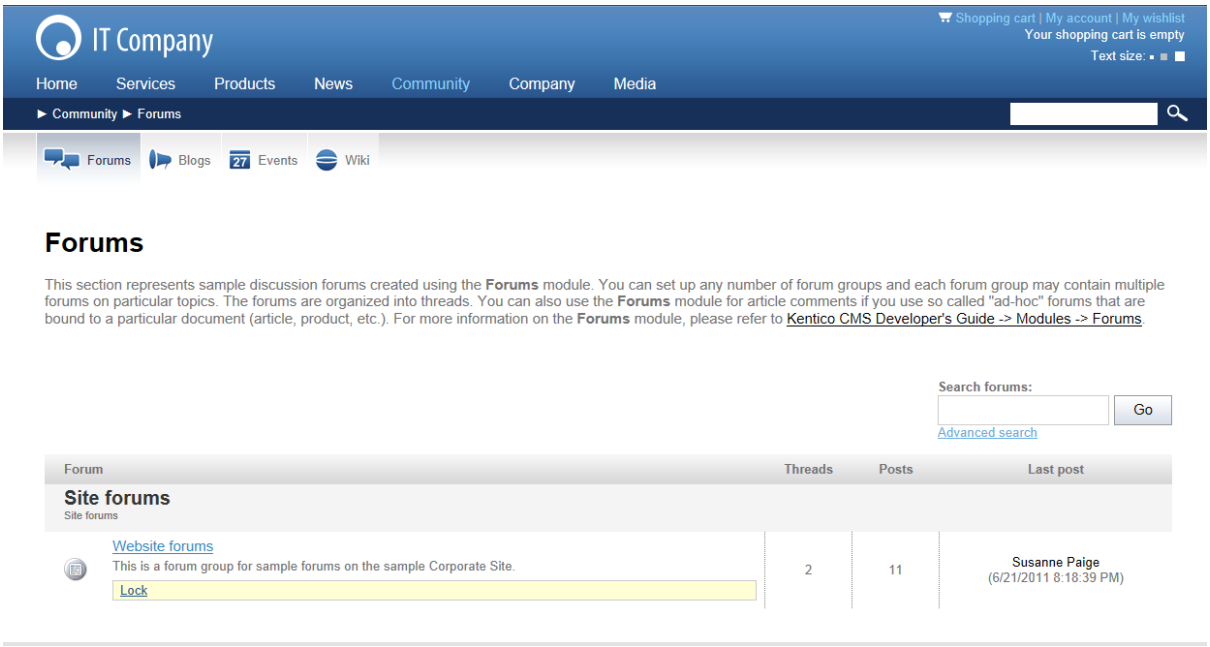
You can create your own field controls as described in the Form controls chapter. If you want to make a form control available in the field editor of the Forms module, you need to check the **Show control in Forms** box in the given control's editing dialog (in **Site Manager -> Development -> Form controls**) and also choose a **Default data type**. To ensure that the form control is also offered in the simple mode of the form field editor, it is also necessary to enable its **High priority** flag.

8.26 Forums

8.26.1 Overview

This module allows you to integrate full-featured forums into your website. The forums are highly configurable and allow you to:

- Organize forums into forum groups
- Open/lock forums
- Perform SQL or index-based full-text searches of forum content
- Manage all forum posts and threads of a particular forum
- Choose to require and/or display e-mails of the forum users
- Subscribe to receive notifications about all posts added to a forum or thread by e-mail
- Create moderated forums (posts needs to be approved before they are displayed on-line)
- Configure standard forum functions such as post attachments, BBCode support, adding posts to favorites, user avatars and more
- Set forums to use Friendly URLs
- Use your own custom forum layouts
- Enable a forum only for authenticated users
- Specify user roles that are allowed to use various forum functions via Security settings



There are two basic types of forums:

- Pre-defined forums - created by the administrator and then displayed on the website.
- Ad-hoc forums (article comments) - created for a single document when a visitor posts the first comment to the given document.

The Message boards module provides another option that allows users to post comments on your website. Please refer to the Modules -> Message boards chapter to learn more about it.

The Forums module can be managed in **CMS Desk -> Tools -> Forums**. Further settings can be found at **Site Manager -> Settings -> Community -> Forums** and are described in more detail in the Settings topic.

The Forums internals and API sub-chapter provides information about the database tables and classes used by the module and examples of how forums can be managed using the API.

Kentico CMS Community Site Guide contains some additional forum examples and tutorials:

- Part 1 -> Forums: Examples of the functionality and customization of groups.
- Part 2 -> Creating the Forums section: A step-by-step tutorial on how to create a sample forum section of a website.

You will need to have the sample Community Site installed to follow Kentico CMS Community Site Guide.

8.26.2 Creating a forum group

All forums are a part of some forum group. Pre-defined forums need to be created within some particular forum group and ad-hoc forums are automatically placed into the **AdHoc forum group** upon creation. A **forum group** usually contains forums related to the same topic. For example:

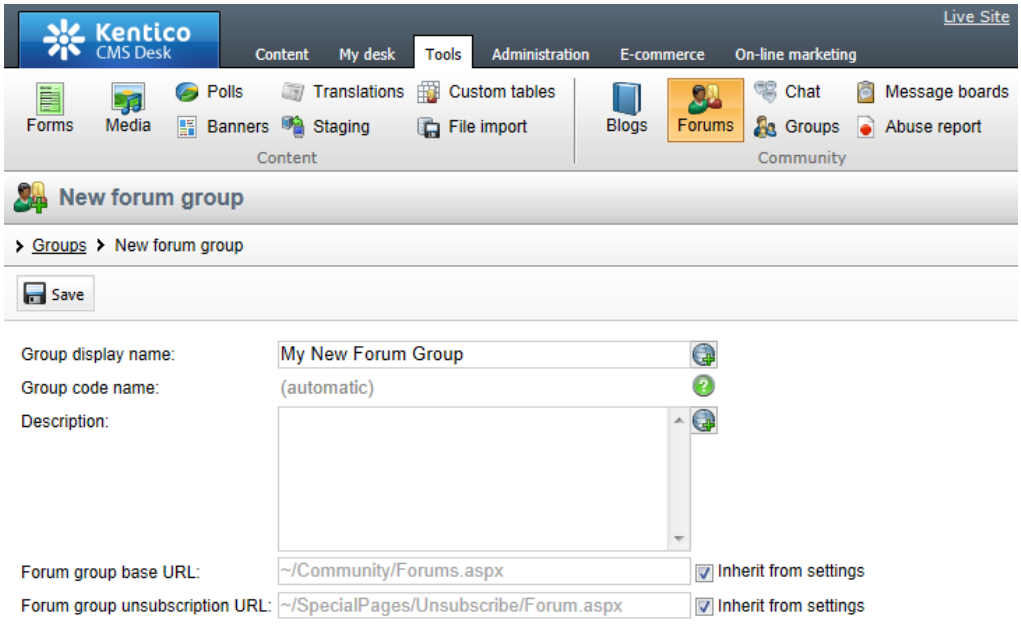
- Computers (forum group)
  - Announcements (forum)
  - Technical questions (forum)
  - FAQ's (forum)
- Web design (forum group)
  - CSS (forum)
  - XHTML (forum)

Creating a new forum group

Go to **CMS Desk -> Tools -> Forums** and click the **New forum group** button. Fill in the following fields:

- **Group display name** - sets the name displayed for the forum group on the website.
- **Group code name** - assigns a unique name to the forum group that can be used to identify it, e.g. in the API. You can leave the *(automatic)* option to have the system generate an appropriate code name based on the display name.
- **Description** - may be used to enter a text description of the forum group, which will be displayed on the website.
- **Forum group base URL** - enter the URL of the page where the forum group will be placed, for example: *~/MyForum.aspx*. This is used by the system when creating absolute URLs leading to the given forum group.
- **Forum group unsubscription URL** - sets the URL of the page where users can unsubscribe from forums under the given group.

Click **Save** to confirm the creation of the group.



Continue to the Creating a pre-defined forum topic to learn how to create individual forums within a forum group or the Adding an ad-hoc forum to the web topic to find out more about ad-hoc forums.

8.26.3 Creating a pre-defined forum

Pre-defined forums need to be created within a forum group before you can publish them on the website. See the Creating a forum group topic to learn how to create a forum group.

Creating a new forum

Go to **CMS Desk -> Tools -> Forums** and click **Edit** () for some forum group. Go to the Forums tab and click  **Add forum**. Enter the following details:

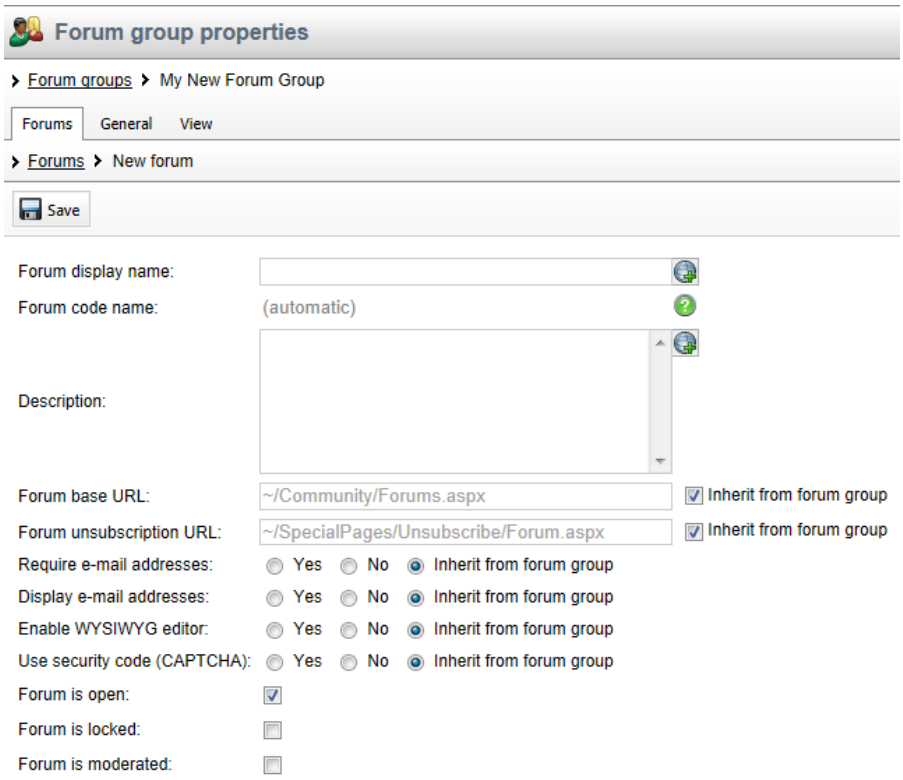
- **Forum display name** - the name of the forum that will be displayed on your website.
- **Forum code name** - assigns a unique name to the forum that can be used to identify it, e.g. in the API. You can leave the *(automatic)* option to have the system generate an appropriate code name based on the display name.
- **Description** - the description displayed on your website.
- **Forum base URL** - URL displayed when the user accesses the forum; *e.g. ~/MyForums.aspx.*
- **Forum unsubscription URL** - URL of the page where users can unsubscribe from the given forum.

- **Require e-mail address** - indicates if e-mail address should be required from the post author.
- **Display e-mail addresses** - indicates if e-mail address of the post author should be displayed to other site visitors.
- **Enable WYSIWYG editor** - indicates if the visitors can use the WYSIWYG editor for entering text.
- **Use security code (CAPTCHA)** - indicates if the user needs to retype the security code displayed as an image. This feature can help avoid spam in the forums.

- **Forum is open** - indicates if the forum is visible and can be accessed.
- **Forum is locked** - if checked, users will not be able to add new posts to the forum, but it will remain accessible for viewing.
- **Forum is moderated** - indicates if the posts need to be approved by a forum moderator.

The **Inherit from forum group** checkboxes or radio button options displayed next to the properties are used to automatically load values from the settings of the parent forum group. To configure unique properties for a forum, simply clear the appropriate checkbox and enter the desired value.

Click  **Save** to add the forum.

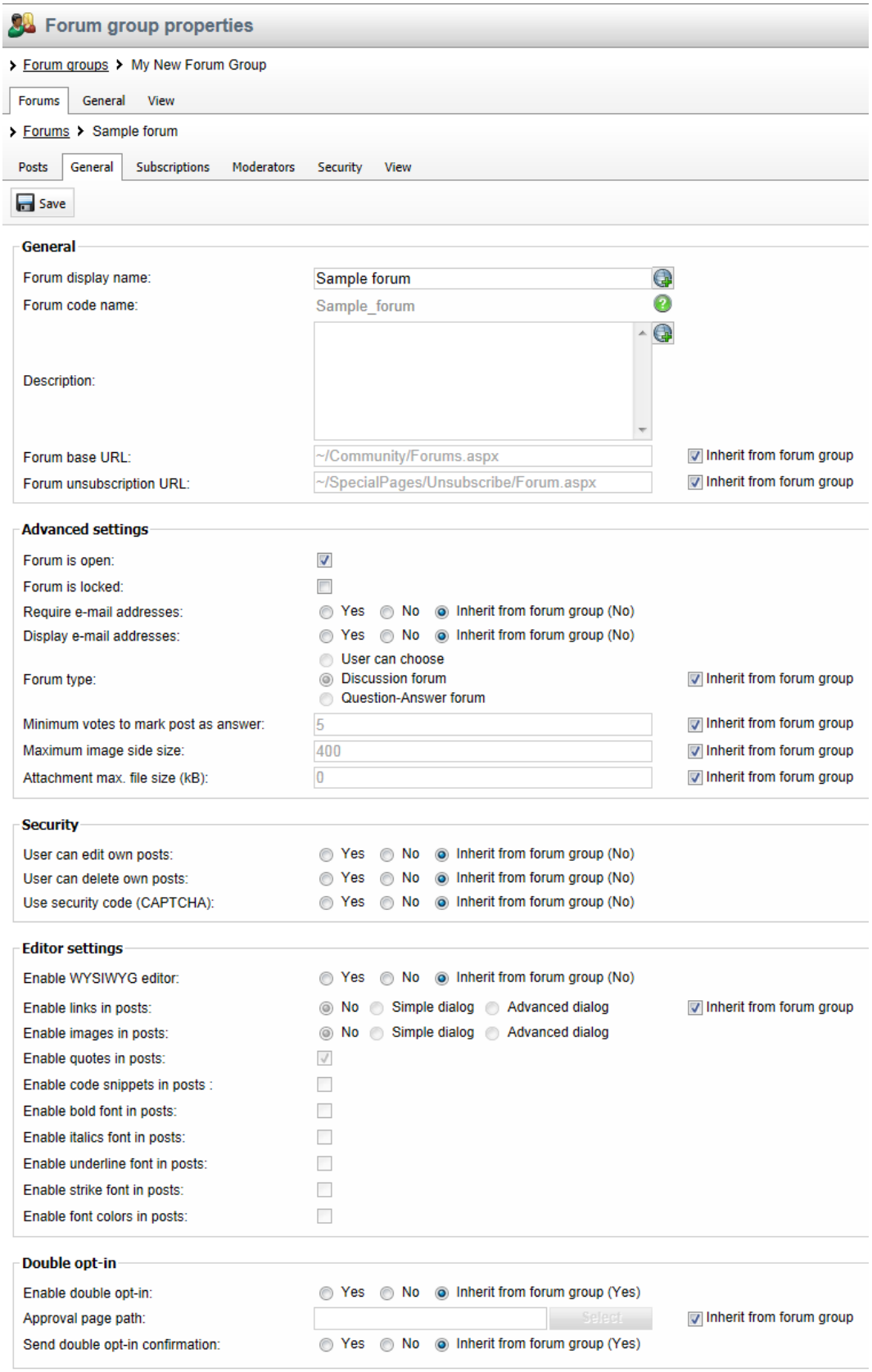


Now that the forum is created, it may be configured in more detail. Switch to its **General** tab, where you can edit the options that were entered when the forum was created, as well as the following additional properties:

- **Forum type** - determines the overall type of the forum. The following three types may be selected:
  - **User can choose** - when creating a new thread, users can choose from the following two types.
  - **Discussion forum** - news threads use a standard discussion format where users reply to previous posts.
  - **Question-Answer forum** - in forums of this type, the initial post of a thread is usually a question and the replies are attempts to answer it. The forum includes a voting feature that allows users to mark individual replies as helpful answers. Once a post receives the amount of votes specified in the *Minimum votes to mark post as answer* property, it will be designated as a valid answer and displayed accordingly.
- **Minimum votes to mark post as answer** - determines the minimum amount of votes that a post must receive before it is marked as an answer in *Question-Answer* type forums.
- **Maximum image side size** - sets the maximum side size in pixels of images inserted into forum posts. If a larger picture is included in a forum post, it will be resized so that its larger side is equal to the entered value.
- **Attachment max. file size (kB)** - sets the maximum file size of forum post attachments in kB.

- **User can edit own posts** - indicates whether users are allowed to edit their own existing posts.
- **User can delete own posts** - indicates whether users are allowed to delete their own existing posts.

The properties in the **Editor settings** section determine which BB code macros may be used by forum users to format the text of forum posts. Further details may be found in the BBCode support topic.



For information about the remaining tabs, please refer to the following topics:

- Managing forum posts
- Subscriptions
- Forum moderation
- Security

Once everything is configured as necessary, the forum can be placed on a website. The Publishing a pre-defined forum on the website topic describes how.

8.26.4 Publishing a pre-defined forum on the website

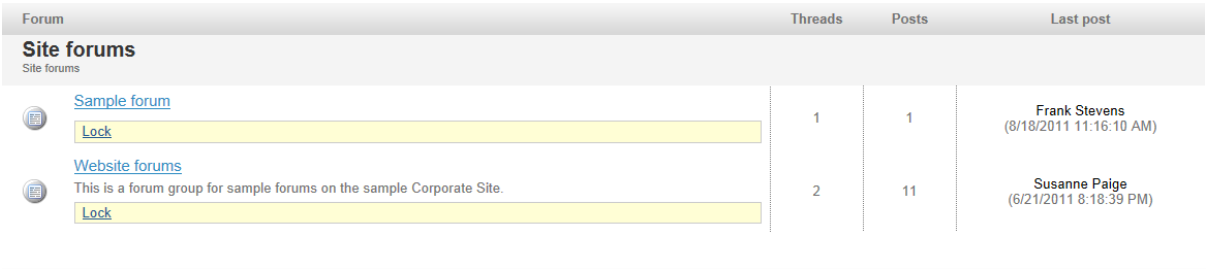
When you want to publish a forum on your website, you can use the built-in web parts on your page templates. You can find more details about each web part in the Kentico CMS Web Parts reference.

Web parts and ASPX page templates

If you are using ASPX page templates, you can simply drag and drop the controls that implement the forum web parts located in the *~/CMSWebParts/Forums* folder onto your page and use them in a similar way.

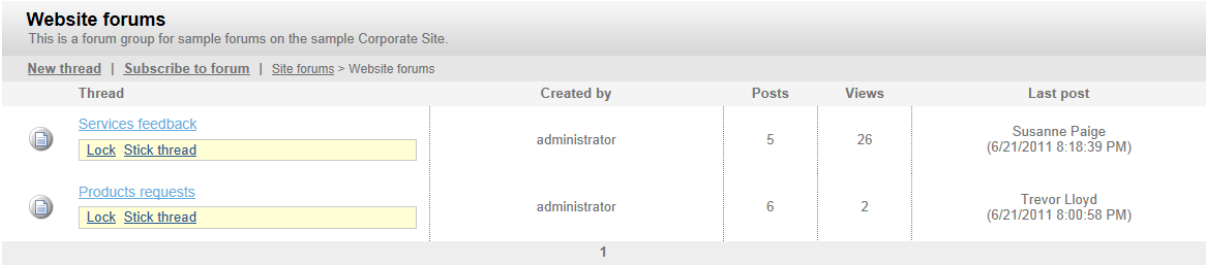
Publishing a forum group on the website

You can publish an entire forum group on the website using the **Forum group** (ForumGroup.ascx) web part. All you need to do is to select the appropriate **Group name** value in the web part properties. The default forum looks like this:

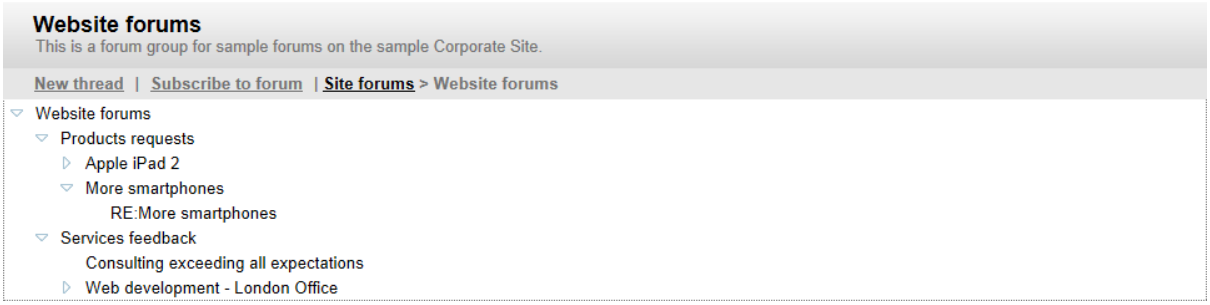


The forum threads in the selected forum can be displayed in two ways:

1. As a list of threads
- set the **Forum layout** property to **Flat**



2. As a tree of threads and posts
- set the **Forum layout** property to **Tree**



8.26.5 Adding an ad-hoc forum to the web

Ad-hoc forums are useful if you want to enable users to add comments to some page or article, but you do not want to create the forum for each document manually. You may also consider using Message boards for this purpose if you don't need a structured forum.

Ad-hoc forums can be added using the **Forum (Single forum - General)** web part.

If you place the **Forum (Single forum - General)** web part to a page and set its **Forum name** property to *ad-hoc forum* (or *ad_hoc_forum* if you are using ASPX templates), the forum will be displayed on the website, but it will be actually created after some visitor adds a first post to the forum. After that, the forum will be created in the **AdHoc forum group** forum group in **CMS Desk -> Tools -> Forums**.

Ad-hoc forums are uniquely identified by the document they belong to.

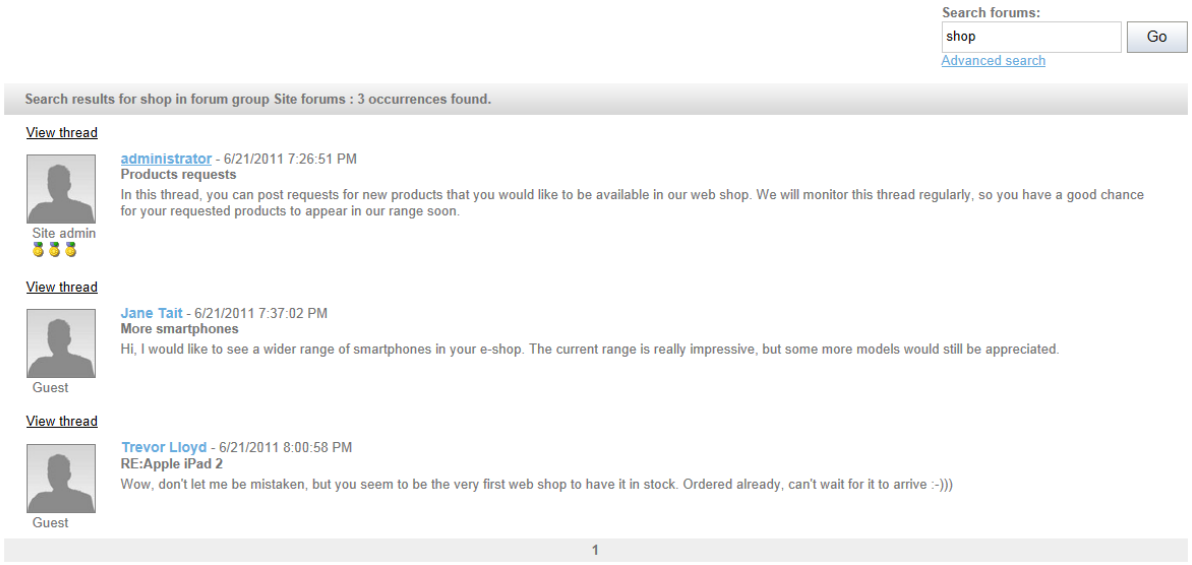
The following two web parts are **deprecated**, and they are available only because of **backward compatibility** with versions prior to 4.0:

- **Forum (Single forum - Flat layout)** - this web part ensures backward compatibility with the former **Forum thread list** web part
- **Forum (Single forum - Tree layout)** - this web part ensures backward compatibility with the former **Forum tree** web part

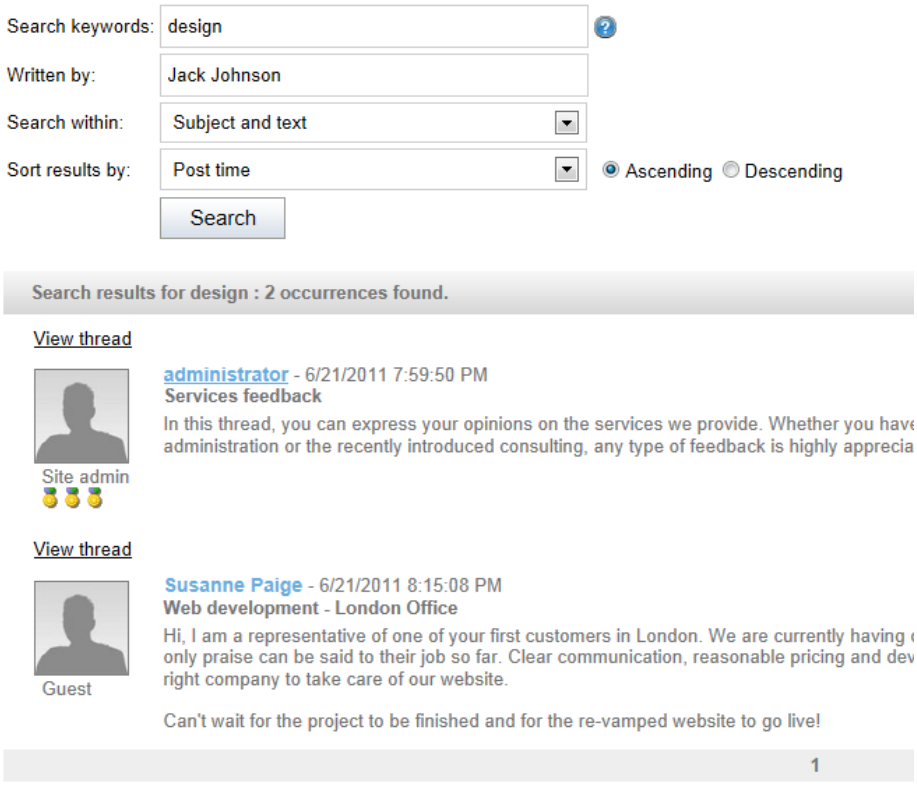
8.26.6 Adding forum searching

If you want to let users search forum posts for some given text, you can use the **Forum search box** (ForumSearch.ascx) and **Forum search results** (ForumSearchResults.ascx) web parts. You will typically place them both on the same page. However, if you need to place the forum search box on a different page, you can set its **Redirect to URL** property to the page where you have the **Forum search**

results web part, such as `~/SearchForum.aspx`.



You can also use the **Forum search - advanced dialog** (ForumExtendedSearchDialog.ascx) web part, which offers a larger dialog that allows users to specify multiple search parameters.



If you wish to use the basic search box by default, but want to let users have the option of using the advanced dialog as well, you can do so by entering the path to the page containing the **Forum search - advanced dialog** web part into the **Advanced search path** property of the **Forum search box** web

part. The **Advanced search** link will then be displayed as seen in the first image.

These web parts use the SQL search engine.

If you prefer index-based searching, this can be provided by various Smart search web parts from the **Full-text search** category if they have a forums type search index set. Please refer to the Modules -> Smart search chapter for more details.

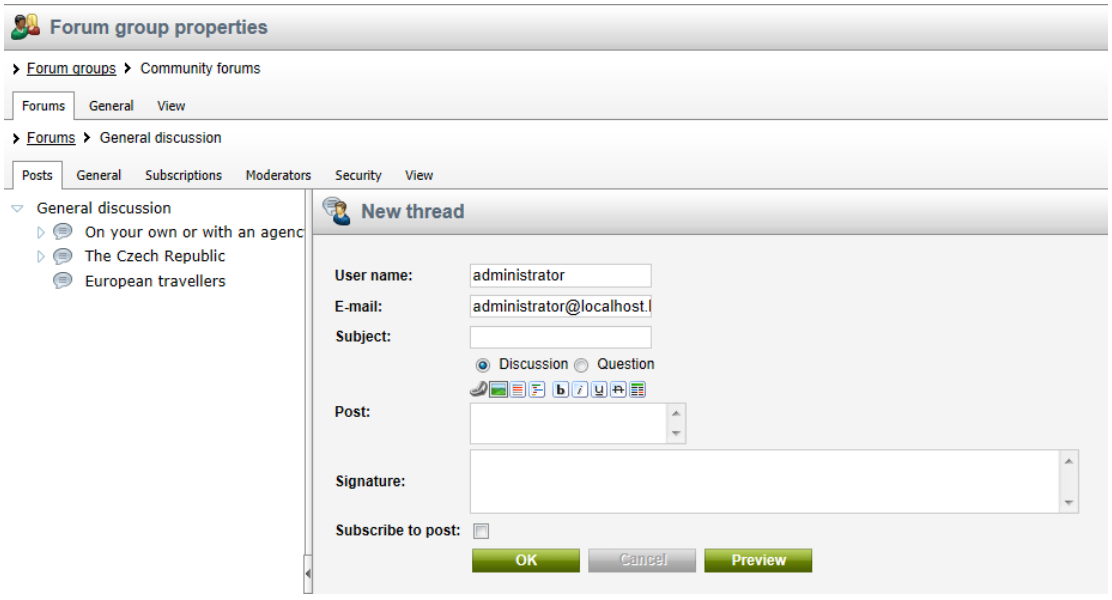
8.26.7 Managing forum posts

Threads and posts can be managed on the **Posts** tab when editing the forum that contains them.

Creating a new thread

When you navigate to the tab, you can initially see the **New thread** page, letting you add a new thread to the current forum group. The following details need to be filled in:

- **User name**: user name which will be used for the thread author
- **E-mail**: e-mail of the thread author
- **Subject**: subject of the initial post of the thread
- **Post**: text of the initial post of the thread
- **Signature**: signature under the initial post
- **Subscribe to post**: if enabled, notification e-mails about new posts in the thread will be sent to the e-mail address specified in the *E-mail* field

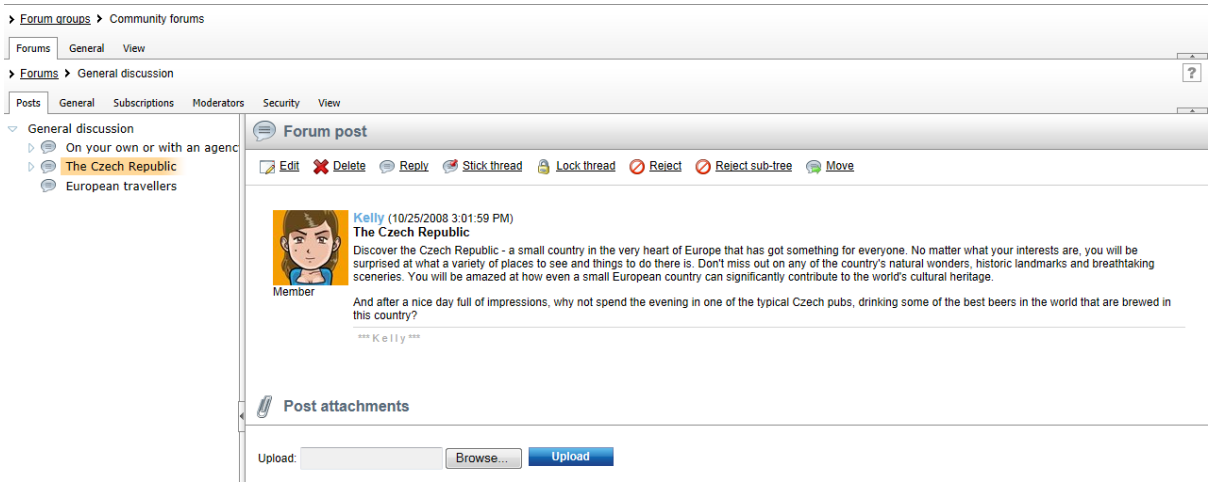


Managing particular posts

If you select a post from the tree on the left, you can perform the following actions:

- **Edit** - opens the post for editing - you can edit the post text, user's name, signature, etc.
- **Delete** - deletes the post
- **Reply** - opens a dialog using which you can send a reply to the post

- **Stick thread** - sticks this thread so that it will be displayed at the beginning of the forum
- **Split thread** - moves the post and its replies to a new thread
- **Lock thread** - locks the thread so that no more posts can be added to it
- **Reject** - rejects the post so that it is not displayed and needs approval to be displayed
- **Reject sub-tree** - performs the Reject action for the current posts and all its children
- **Move** - moves a thread to a new forum

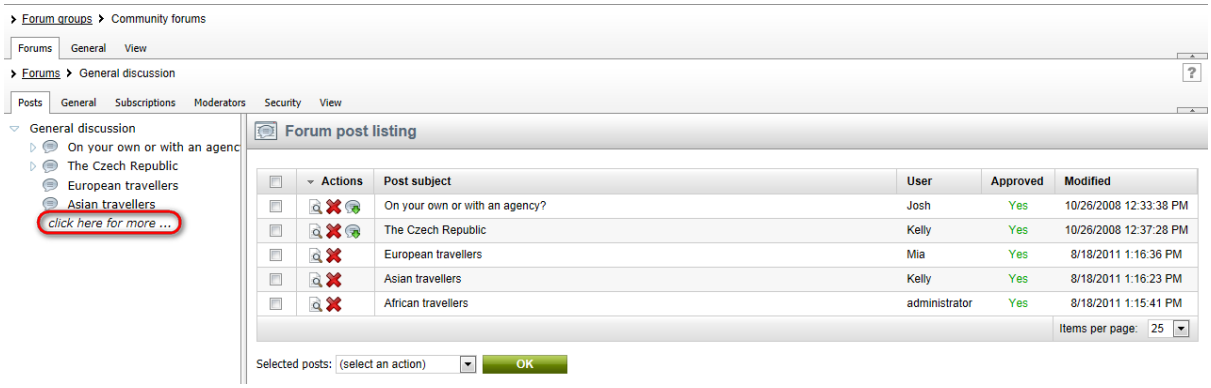


Forum post listing

In case that there are too many threads in the forum, you can limit the number of displayed threads using the **Max forum post nodes** settings key in **Site Manager -> Settings -> Community -> Forums**. If you use this key and there are more threads in the forum than the set value, the **Click here for more...** link is displayed at the bottom of the tree. After clicking this link, the **Forum post listing** page is displayed in the main area.

The **Forum posts listing** page initially displays all threads in the forum, while the following actions can be performed:

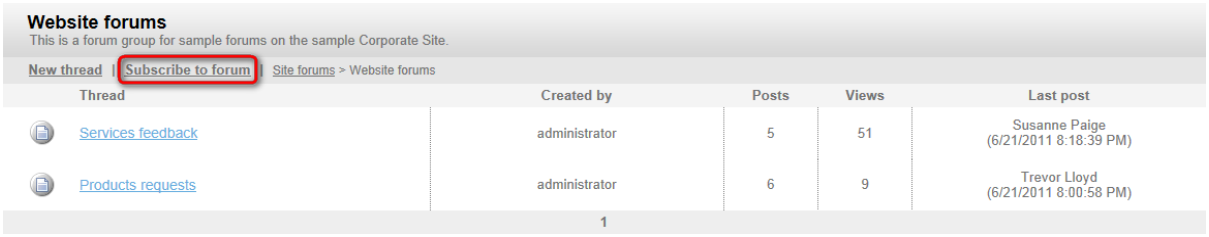
- **View** - displays the thread's initial post in the main area (this action is identical to clicking the post in the tree)
- **Delete** - deletes the whole thread including all child posts
- **Sub-posts** - displays sub-posts of the post in the grid in the main area
- **Parent post** - displays a list of sub-posts of the parent post in the main area (goes up one level)



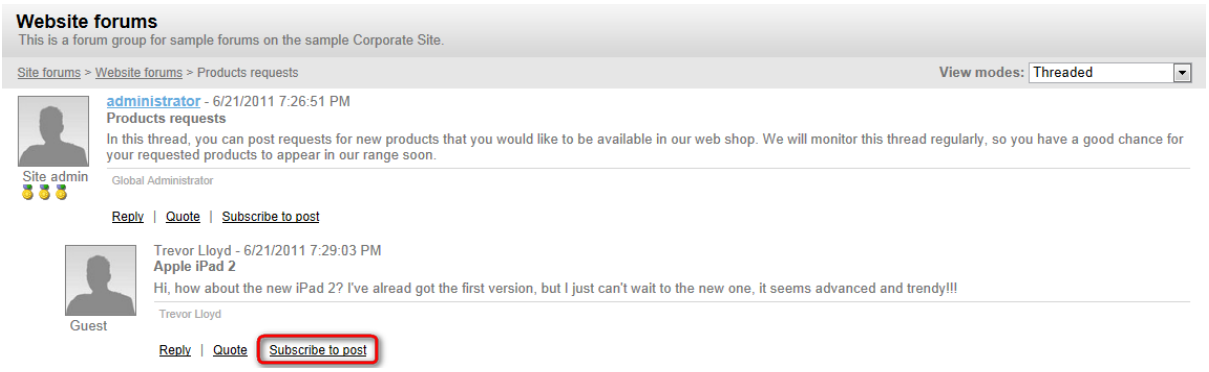
8.26.8 Subscriptions

Subscriptions allow users to receive e-mail notifications when a new post is added to a selected forum or as a reply to a specific post. Users may subscribe directly while browsing through forums on the live site.

When viewing the thread list of a forum, the **Subscribe to forum** link displayed at the top provides a way to subscribe to the entire forum (all of its threads).



To subscribe to a specific post and the replies under it, users can click on the **Subscribe to post** button shown below any given post.



Both of these actions open a dialog where users can submit their e-mail address.

Please note

The subscription functionality may be disabled through the properties of the web part used to publish the forum or forum group.

Additionally, the Security settings of specific forums may be configured to only allow subscription for certain types of users or roles.

Enabling double opt-in

To make users confirm that the e-mail address they're subscribing with really exists and belongs to them, you can enable the double opt-in functionality. With double opt-in enabled, subscribers will be sent an e-mail with a confirmation link. They will have to click the link to create their subscription.

To enable double opt-in globally for all forums:

1. Go to **Site Manager -> Settings -> Community -> Forums**.

2. Turn on the **Enable double opt-in for forums** setting.

3. Specify how long you want the confirmation links to be valid in the **Double opt-in interval** setting.

The following two steps are optional. If you omit them, the system will use the default approval page located in *~/CMSModules/Forums/CMSPages/SubscriptionApproval.aspx*.

4. Place the **Forum subscription confirmation** web part on a new page. Adjust its properties according to your needs.

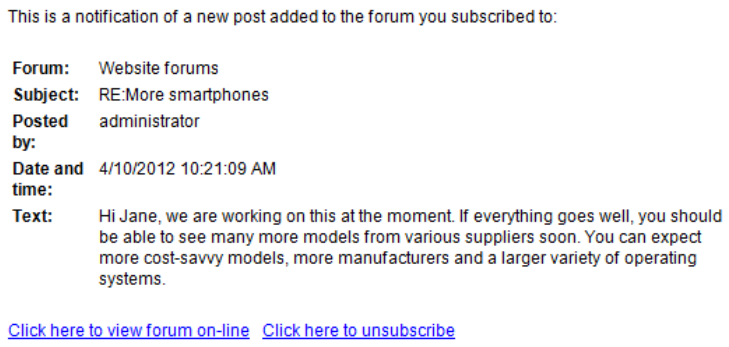
5. Enter the page's path into the **Double opt-in approval page path** setting.

Double opt-in should now be configured for all forums. You can override the settings for each forum group in its properties in **CMS Desk -> Tools -> Forums**. You can also define specific settings for particular forums.

The functionality uses the **Forums - Subscription request** e-mail template. You can insert the confirmation link into the template using the *{%SubscriptionLink%}* macro.

Unsubscription

Users can cancel their forum subscriptions at any time by clicking on the unsubscription link included in every notification e-mail about a new post (when using the default e-mail template).


This is a notification of a new post added to the forum you subscribed to:

Forum:	Website forums
Subject:	RE:More smartphones
Posted by:	administrator
Date and time:	4/10/2012 10:21:09 AM
Text:	Hi Jane, we are working on this at the moment. If everything goes well, you should be able to see many more models from various suppliers soon. You can expect more cost-savvy models, more manufacturers and a larger variety of operating systems.

Click here to view forum on-line Click here to unsubscribe

The link leads to a page that automatically processes the unsubscription request and deletes the appropriate subscription (it is identified using a parameter passed in the query string of the link's URL).

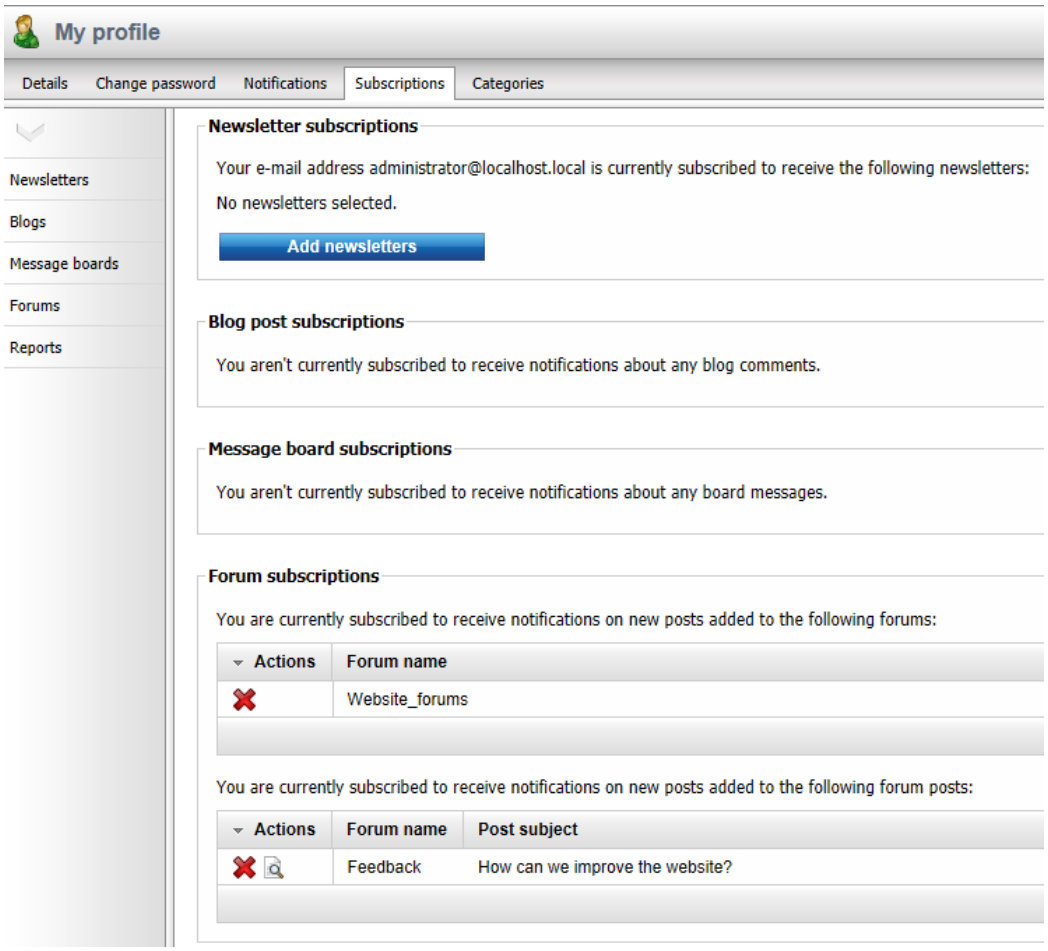
This is either handled by the *~/CMSPages/Unsubscribe.aspx* system page, or by a custom unsubscription page created for the website. If you wish to use a custom page, it is necessary to place the Forum unsubscription web part onto it, which ensures all the required functionality. The URL of this page then needs to be entered into the **Forum unsubscription URL** field of the given forum or forum group (the value can also be inherited from the website's **Forum unsubscription URL** setting, which can be specified in **Site Manager -> Settings -> Community-> Forums**).

Subscription e-mails and the Forum base URL

It is important to set a correct **Forum base URL** value in the forum configuration so that the subscription notifications and other related e-mails contain valid links leading to the appropriate forum.

When entering the value, you can use the relative URL of the page where the forum is located, for example: *~/Community/Forums.aspx*

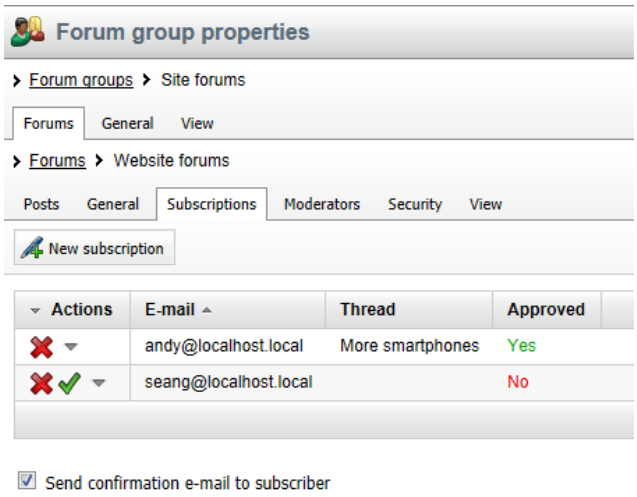
Alternatively, users with access to CMS Desk can view all of their active forum subscriptions in **My Desk -> My profile -> Subscriptions** and manually **Unsubscribe** (✖) as needed.



For live site users, the same options can be provided by the My account web part, as long as its **Display my subscriptions** and **Display forum subscriptions** properties are enabled.

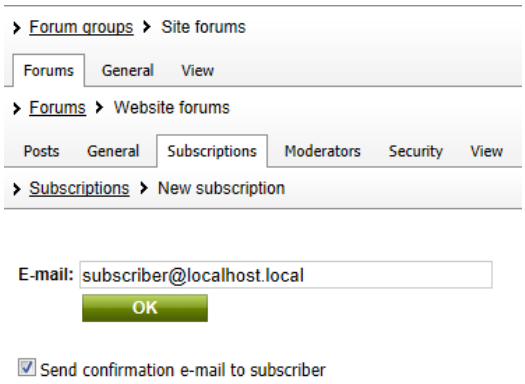
Managing subscriptions

When editing a forum in **CMS Desk -> Tools -> Forums**, administrators can manage all of the given forum's subscriptions on the **Subscriptions** tab.



The **Delete** (✖) action may be used to remove existing subscriptions. If this is done with the **Send confirmation e-mail to subscriber** checkbox enabled, a notification e-mail will be sent to the given address, informing about the unsubscription.

The **New subscription** button provides a way to subscribe a user to the whole forum. When clicked, the dialog depicted in the screenshot below is opened. All that needs to be done is enter the e-mail address of the user that should be subscribed and then click **OK**.



Like when removing subscriptions, the checkbox at the bottom of the dialog indicates if a confirmation e-mail should be sent to the address of the new subscriber. These confirmation e-mails are sent automatically to users who subscribe on-site.

E-mail templates

The post notification e-mails are sent to the subscribed users based on the **Forums - New post** e-mail template, which can be edited in **Site Manager -> Administration -> E-mail templates**. The notifications informing users about their forum subscriptions or unsubscriptions are based on the **Forums - Subscription confirmation** and **Forums - Unsubscription confirmation** templates respectively. If you have double opt-in enabled for forums, the e-mail that is sent upon requesting subscription is based on the **Forums - Subscription request** template.

You can use the following specific context macros in forum subscription e-mail templates:

- **{% ForumDisplayName %}** - display name of the related forum.
- **{% ForumName %}** - code name of the related forum.
- **{% ForumDescription %}** - resolves into the description text of the forum.
- **{% GroupDisplayName %}** - display name of the related forum group.
- **{% GroupName %}** - code name of the related forum group.
- **{% GroupDescription %}** - description text of the forum group.
- **{% PostSubject %}** - the subject of the new forum post that caused the notification to be sent.
- **{% PostText %}** - resolves into the HTML content of the new post.
- **{% PostTextPlain %}** - resolves into the plain text version of the new post's content.
- **{% PostUsername %}** - name of the user who created the post.
- **{% PostTime %}** - the date and time when the new post was added.
- **{% Link %}** - resolves into a link to the new post.
- **{% UnsubscribeLink %}** - generates a link that can be used to unsubscribe from the given forum.

The two macros below are available for (un)subscription notifications:

- **{% Subject %}** - if the user is subscribed to a specific post, this resolves into the subject of the given post. Otherwise it returns an empty string.
- **{% Separator %}** - if the user is subscribed to a specific post, this returns the value of the forums.confirmationtemplateseparator resource string (" - " by default). This can be placed between the name of the forum and the subject of the post to cover both possible subscription scenarios.

You can use the following macro when double opt-in is enabled:

- **{% SubscribeLink %}** - contains the URL of the page used to confirm subscriptions.

You can also access the following related objects and their properties (e.g. *{% ForumPost.PostViews %}*:

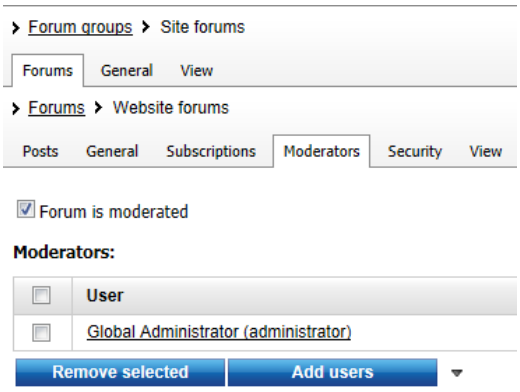
- **{% ForumPost %}** - *ForumPostInfo* object of the post to which the user is subscribed (if applicable).
- **{% Forum %}** - *ForumInfo* object representing the forum to which the user is subscribed.
- **{% ForumGroup %}** - *ForumGroupInfo* object of the forum group containing the given forum.
- **{% Subscriber %}** - *ForumSubscriptionInfo* object representing subscription of the recipient to the given forum. Only available for the *New post* template.

8.26.9 Forum moderation

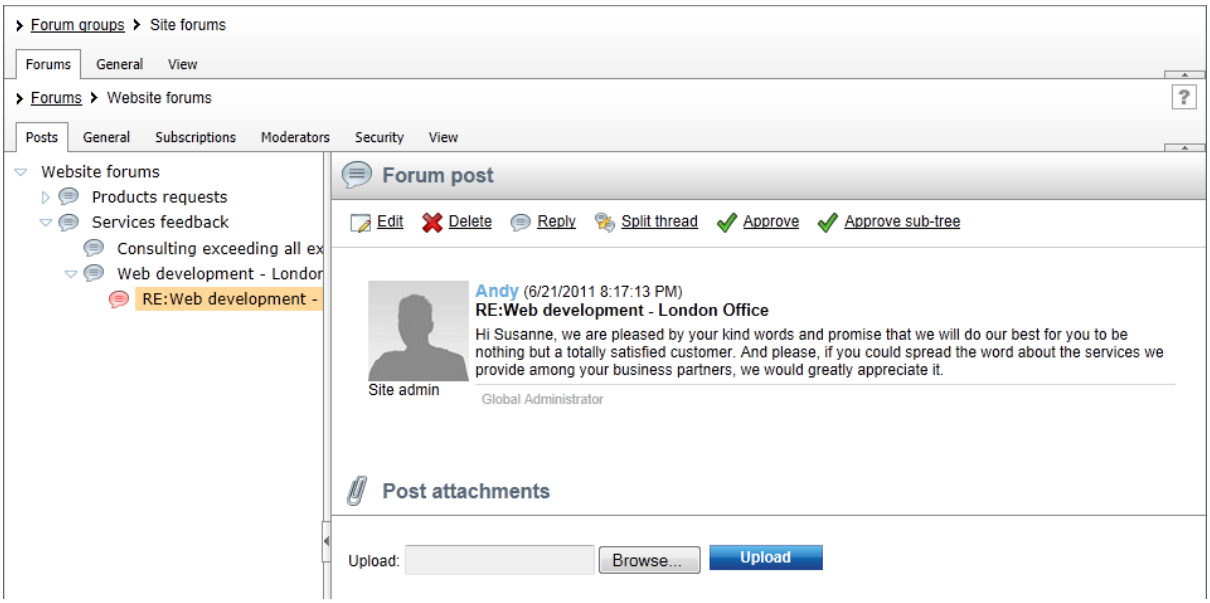
You can configure a forum so that it is moderated. It means that all posts must be approved by one of the forum moderators before they are published in the forum.

Configuring forum moderation

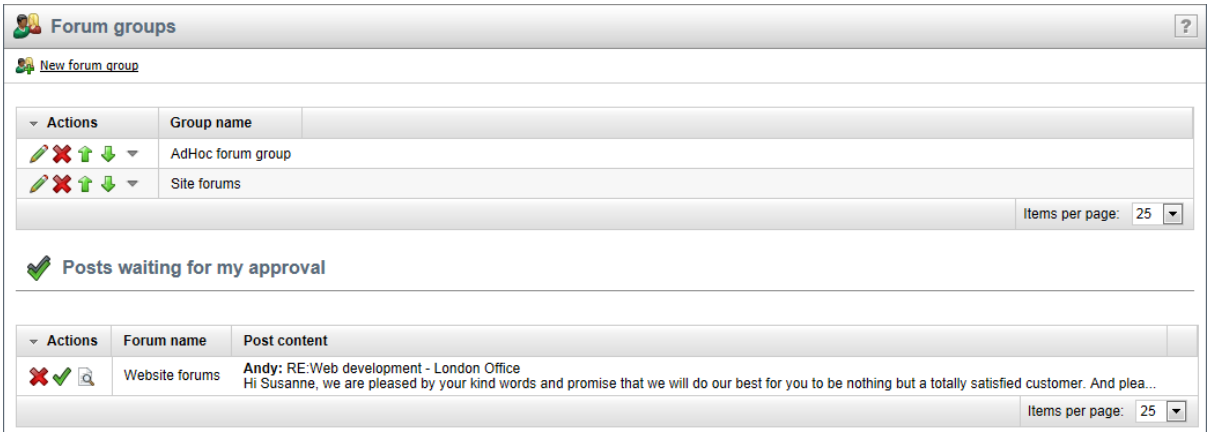
You can enable forum moderation and configure the list of moderators under the **Forum properties -> Moderators** tab. Even when this option is disabled, approval of posts might be required, e.g. when a Bad word is detected or when a post has been rejected.



If you're a moderator, you can moderate the forum posts either **directly on the website** or on the **Forum properties -> Posts** tab:



... or you can find the list of all posts waiting for your approval on the **Forum groups** dialog:



The moderators automatically receive an e-mail notification when a new item is waiting for approval. The e-mail template can be modified in **Site Manager -> Administration -> E-mail templates -> Forums - Moderator notification**. To insert dynamic values into the template, you may use the same macro expressions as are described for the *Forums - New post* notification in the Subscriptions topic.

8.26.10 Forum post attachments

Users can attach files to forum posts. To enable users to do this, you have to assign the **Attach files** permission on the forum's **Security** tab to the desired user roles or all authenticated users. See the Security topic for more details. There are also some additional settings related to attachments:

In **Site Manager -> Settings -> Community -> Forums**, you can make the following setting:

- **Forum attachments allowed extensions** - you can specify which file extensions can be attached to forum posts. Extensions should be entered without dots and separated by semicolons. If blank, all extensions are allowed.

In web part properties of the **Forum group** web part, you can make the following settings:

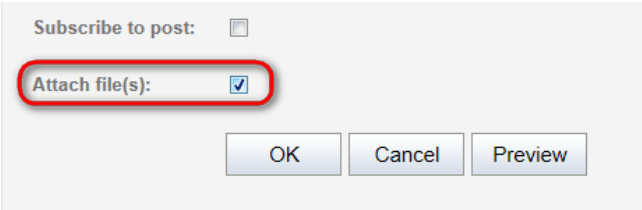
- **Display attachment image** - if checked, attached images will be displayed as images, not only as links.
- **Attachment image maximal side size** - if an attached image is larger than the entered value, it will be resized so that its larger side's size is equal to the entered value.

When editing a forum in **CMS Desk -> Tools -> Forums**, you can set the following property on the **General** tab:

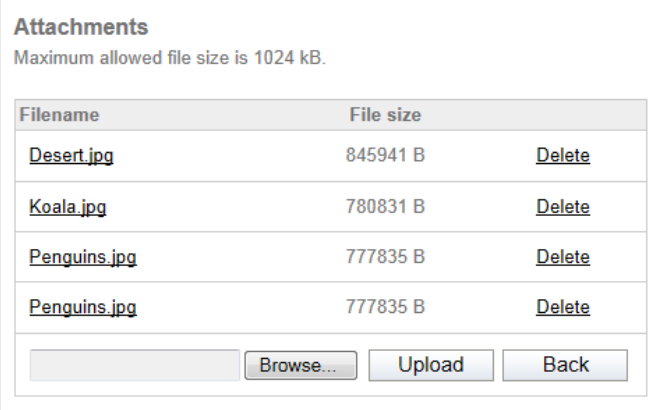
- **Attachment max. file size (kB)** - you can define the maximum size of an attached file in kB.

Adding attachments to a forum post

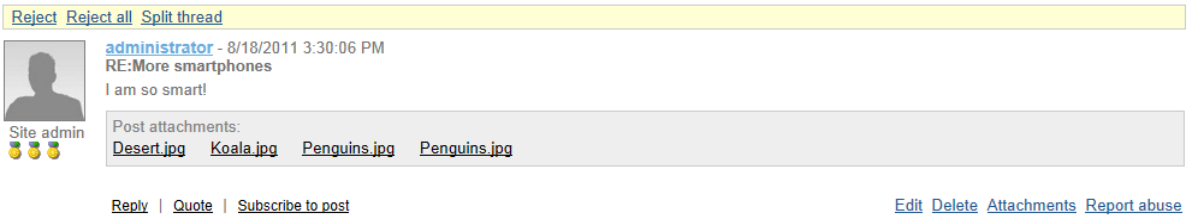
When adding a post to a forum, users can check the **Attach file(s)** check box at the bottom of the post adding dialog.



After clicking **OK**, a new dialog will be displayed, where users can upload the attachments. When all desired images are uploaded, users can click the **Back** button, which will take them back to the forum.



Back in the forum thread, links to uploaded attachments will be displayed with the post, as you can see in the screenshot below.



8.26.11 BBCode support

Bulletin Board code (the *BBCode* abbreviation is commonly used) is a lightweight markup language designed to let users format text of their messages. It is used in many forums on the web, not just on websites created with Kentico CMS. Its tags are similar to HTML tags and are entered in square brackets.

Users can use BBCode in their forum posts only if it is enabled. You have to allow use of BBCode when editing ( ) a forum or forum group, on the **General** tab of its editing interface in **CMS Desk -> Tools -> Forums**.

The following table explains particular BBCode tags. The first column shows icons on the BB editor toolbar. Clicking one of these icons encloses the selected text within the respective tags. The second column shows examples of text enclosed within the tags. The **Description** column explains what effects the tags have on the formatting. The last column lists respective properties on the **General** tab that have to be enabled in order for the tags to be functional.

Icon	Example	Description	Property on the General tab
	[url]http://example.org[/url] [url=http://example.com]Example[/url]	Makes the text a link leading to the URL. One of the following three options can be selected: **No** - the macros are not functional	Enable links in posts

		and the icon is not displayed **Simple dialog** - clicking the icon displays a simple dialog where the URL of the link can be entered manually **Advanced dialog** - clicking the icon displays the Insert link dialog where the URL can be selected in various ways	
	[img]http://www.imagesxy.com/img.bmp[/img] [img=200x50]http://www.imagesxy.com/img.bmp[/img] [img=200]http://www.imagesxy.com/img.bmp[/img]	Displays an image located at the URL. The optional parameter resizes the image. It can be added either in format *<width>x<height>* or *<max side size>*. One of the following three options can be selected: **No** - the macros are not functional and the icon is not displayed **Simple dialog** - clicking the icon displays a simple dialog where the URL of the link can be entered manually **Advanced dialog** - clicking the icon displays the Insert image dialog where the image can be selected in various ways	Enable image macros in posts
	[quote]quoted text[/quote] [quote=Administrator]quoted text[/quote]	Displays text in a grey box; used for quotations. The optional parameter displays *Administrator wrote:* and the quoted text on a new line.	Enable quote macros in posts
	[code]code example[/code]	Displays text in monospaced format; used for code snippets.	Enable code snippet macros in posts
	[b]bold text[/b] [strong]bold text[/strong]	Makes the text bold.	Enable bold font macros in posts
	[i]italicized text[/i] [em]italicized text[/em]	Makes the text italic.	Enable italics font macros in posts
	[u]underlined text[/u]	Underlines the text.	Enable underline font macros in posts
	[s]strikethrough text[/s]	Strikes the text through.	Enable strike font macros in posts
	[color=red]Red Text[/color] [color=#f00]Red Text[/color] [color=f00]Red Text[/color] [color=#ff0000]Red Text[/color] [color=ff0000]Red Text[/color]	Sets the text color.	Enable font color macros in posts

URLs in BBCode macros

All URLs in macros (URL, IMG) are validated as a URL to avoid XSS and resolved into their absolute URL equivalents. The following URL formats can be used:

- **www.google.com** – URL starting with www.
- **http://devnet.kentico.com** – URL starting with protocol
- **~/CMSDesk/default.aspx** – Virtual path
- **../default.aspx** – Relative URLs
- **/KenticoCMS/default.aspx** – Server relative URL

API for BBcode macros

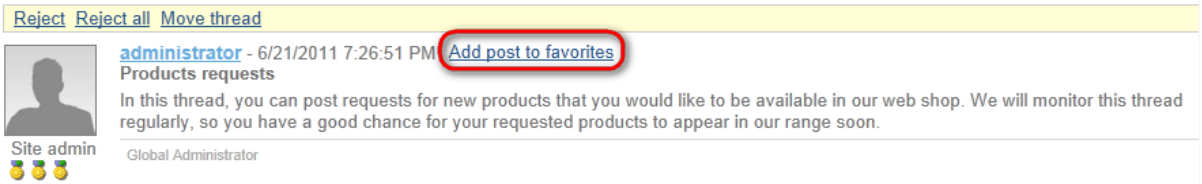
There is an easy way to resolve macros in ASPX or codebehind code. To resolve all the macros (recommended), use the following method:

string **CMS.CMSHelper.CMSContext.ResolveDiscussionMacros**(string **inputText**)

Or you can use the **CMS.GlobalHelper.DiscussionMacroHelper** class to resolve macros with particular settings using the **ResolveMacros** method from an object of this class.

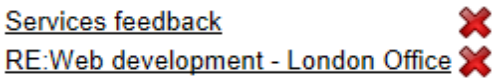
8.26.12 Forum favorites

If you enable the **Enable favorites** web part property of the **Forum group** web part, forum posts will be displayed with the **Add post to favorites** link, as you can see in the screenshot below.



If a user clicks this link, the post will be added to their favorite posts.

If you place the **Forum favorites** web part to some page, the web part will display links to favorite posts of the current user. You can see the appearance of the web part with two favorite posts in the screenshot below.



8.26.13 Friendly URLs

Forums display content based on the current values of the **forumid** and **threadid** query string parameters. By default, the URL of a forum thread might look like the following:

- **<domain>/Forums.aspx?forumid=3&threadid=12**

A friendly URL of the same forum thread looks like the following:

- **<domain>/Forums/f4/t13/Frequently-asked-questions.aspx**

Having your forum URLs in this format is a good practice when it comes to SEO (Search Engine Optimization).

Enabling friendly URLs

For this to be enabled, you have to do the following two tasks:

1. Set the following properties of the **Forum group** web part:

- **Use friendly URLs** - check to enable the Friendly URLs feature.
- **Friendly base URL** - enter the base part of the URL, which is displayed after the domain name, e.g. **/Forums** for the example above and for the document aliases listed below.
- **URL extension** - extension that will be used at the end of the friendly URL.

2. Assign the following document aliases to the document containing the Forum group web part. The **/Forums** part of each alias must be equal to the value in the **Friendly base URL** property of the Forum group web part:

- **/Forums/f{forumid}/{whatever}**
- **/Forums/f{forumid}/fp{fpage}/{whatever}**
- **/Forums/f{forumid}/t{threadid}/{whatever}**
- **/Forums/f{forumid}/fp{fpage}/t{threadid}/{whatever}**
- **/Forums/f{forumid}/t{threadid}/tp{tpage}/{whatever}**
- **/Forums/f{forumid}/fp{fpage}/t{threadid}/tp{tpage}/{whatever}**

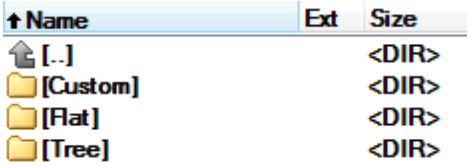
Single forum friendly URLs

If you want to enable friendly URLs for a single forum, the **Forum (Single forum - General)** web part should be set the same way as described in step 1, but only the following two document aliases need to be added to the document:

- **/Forums/t{threadid}/{whatever}**
- **/Forums/t{threadid}/tp{tpage}/{whatever}**

8.26.14 Customizing forum design

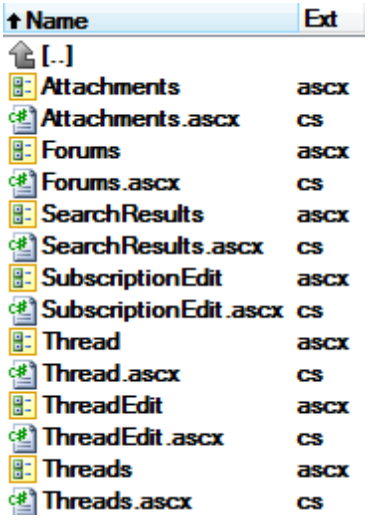
Forum layouts are stored in **<web project>\CMSModules\Forums\Controls\Layouts**. The folder contains three sub-folders by default.



The **Flat** and **Tree** folders contain controls for the identically named layouts. The **Custom** folder can be used for defining custom layouts.

To **create a custom layout**, you need to create a sub-folder inside the **Custom** folder. The name of this sub-folder will be used as the name of the layout. The sub-folder has to contain controls with the same

names as included in the Flat or Tree folders. You can see a screenshot of these files below. When writing the controls' code, make sure that they inherit from **ForumViewer**.



8.26.15 Security

The security model of the Forums module has two parts:

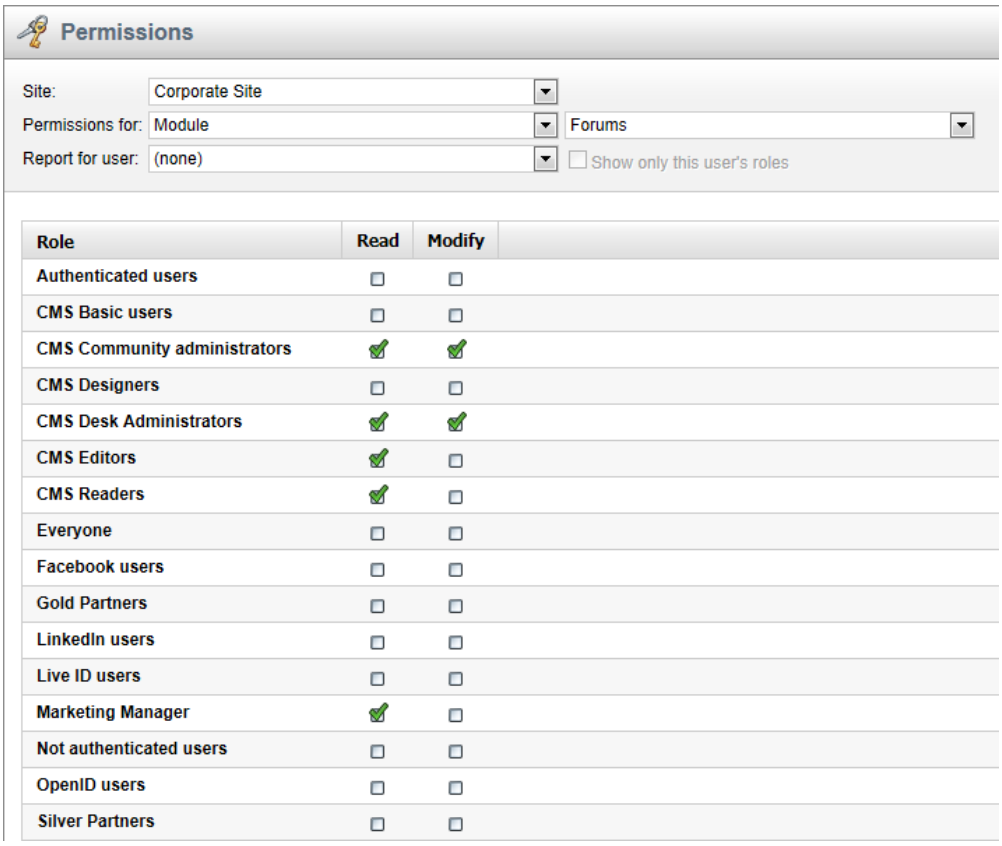
1. Security of the Forums module administration interface.
2. Security of the forums published on the website.

Forums module administration interface

Access to the Forums module administration interface in **CMS Desk -> Tools** can be managed in the **Modules -> Forums** permission matrix at **CMS Site Manager -> Administration -> Permissions** (or **CMS Desk -> Administration -> Permissions**; only applies to the current site):

- **Modify** - allows users to modify forum settings
- **Read** - allows users to only read forum settings

Users without any permissions who are moderators of at least one forum are allowed to access the **Posts waiting for my approval** dialog only.



Security of forums published on a website

If you **Edit** (🖉) some forum and switch to its **Security** tab, the permission matrix displayed in the screenshot below will be displayed.

Columns of the matrix represent the following actions:

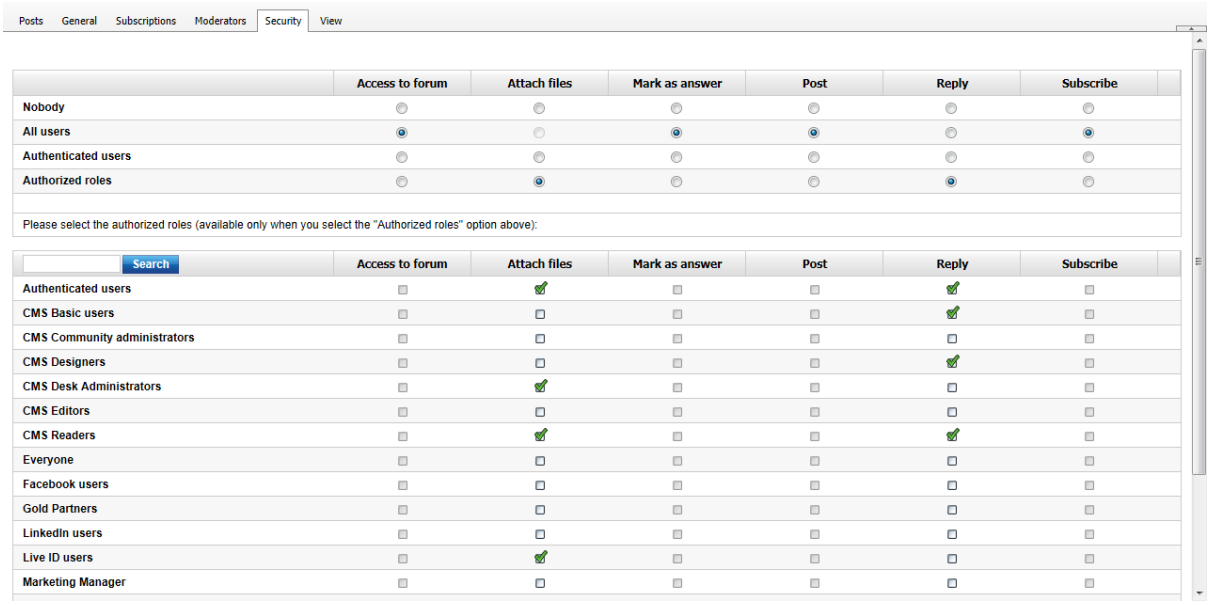
- **Access to forum** - defines who can enter the forum and view posts
- **Attach files** - defines who can attach files to forum posts
- **Mark as answer** - defines who can mark posts as answers in Question - Answer forums
- **Post** - defines who can add posts to the forum
- **Reply** - defines who can reply to forum posts
- **Subscribe** - defines who can subscribe for receiving notifications about new posts in the forum

Rows in the top part of the matrix have the following meanings:

- **Nobody** - the action can't be performed by anyone
- **All users** - anybody can perform the action
- **Authenticated users** - only signed-in registered users can perform the action
- **Authorized roles** - only members of roles specified in the lower part of the matrix can perform the action

Below the permission matrix, there is one more check-box:

Allow user to change the name - if checked, users can change their name displayed with a forum post when entering the post; if unchecked, their user name will be used



The following properties of the **Forum group** web part are also related to forum security:

- **Hide forum to unauthorized users** - indicates whether forums for which the user has no permissions are visible for them in the list of forums in a forum group
- **Redirect unauthorized users** - determines whether to redirect unauthorized users to the logon page or whether to display only an info message
- **Access denied page URL** - URL where the user is redirected when trying to access forum for which they are not authorized

8.26.16 Settings

Settings of the Forums module are located in **Site Manager -> Settings -> Community-> Forums**. The following settings are available:

- **Send forum e-mails from** - sets the e-mail address that will be used as the sender for forum notifications and confirmation e-mails.
- **Forum base URL** - this setting should contain the relative URL of the page where the website's forums are located, for example *~/Community/Forum.aspx*. Forum groups and individual forums can either inherit this value from the settings or use their own. Having a correct base URL is important for generating valid links in forum-related e-mails.
- **Forum attachments allowed extensions** - allows you to specify a list of file extensions that will be allowed for forum post attachments. The extensions should be entered without dots and separated by semicolons. If blank, all extensions are allowed.
- **Max forum post nodes** - determines the maximum number of forum post nodes displayed in the forum post tree when editing a forum in *CMS Desk -> Tools -> Forums*. If there are more nodes than allowed by this value, the *click here for more...* link will be displayed at the bottom of the tree, letting you display a list of all nodes in the main area.

- **Forum unsubscription URL** - sets the URL of the page that unsubscribes users from receiving notifications about new forum posts. The Forum unsubscription web part must be placed on the page in order for the unsubscriptions to work. The value of this setting can be inherited by forum groups and through them by particular child forums. If left empty, the *~/CMSPages/Unsubscribe.aspx*

system page is used.
- **Enable double opt-in for forums** - indicates if double opt-in should be enabled for forum objects. When enabled, users are required to confirm their subscription by clicking a link that is sent to them in an e-mail.
- **Double opt-in approval page path** - path to the page that contains the Forum subscription confirmation web part. The subscription confirmation link that will be sent to users will point to this page.
- **Double opt-in interval (hours)** - amount of time in hours, during which a user has to confirm their subscription request.
- **Send double opt-in confirmation** - indicates if an e-mail confirmation should be sent to user after they approve a subscription. If double opt-in is disabled, these confirmation e-mails are always sent.

8.26.17 Forums internals and API

8.26.17.1 Overview

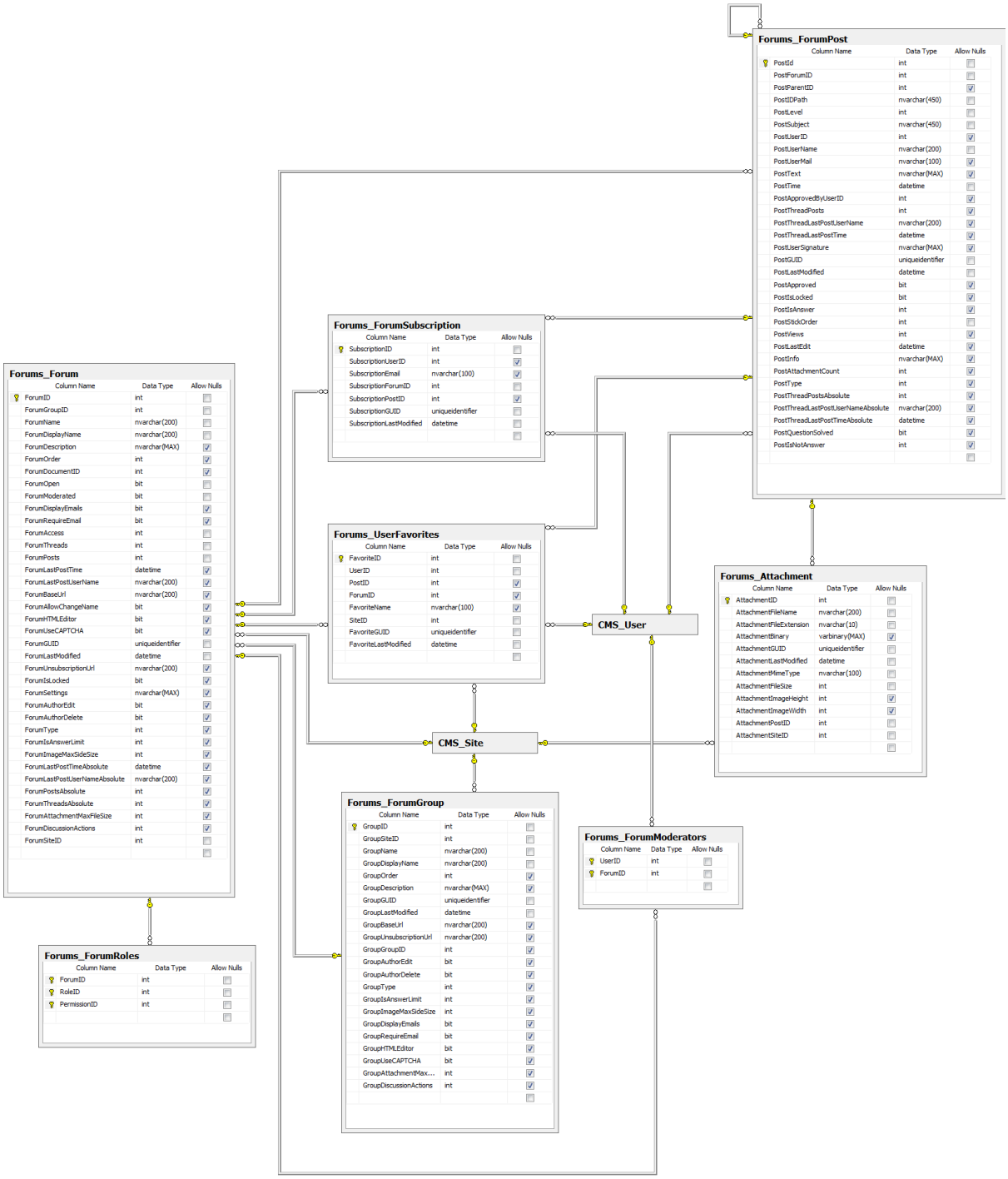
In this chapter, you will learn which database tables and API classes are used in the Forums module. You will also see the most common API examples.

Further API-related information and examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all members of the module's classes, please refer to Kentico CMS API Reference.

8.26.17.2 Database tables

The following database tables are used to store information about forums and their content:

- **Forums_ForumGroup** - contains records representing forums groups and their settings.
- **Forums_Forum** - contains records representing individual forums and their settings.
- **Forums_ForumPost** - contains records of forum posts and their content.
- **Forums_ForumModerators** - stores relationships between forums and users. Each entry in this table indicates that a user is a moderator of a specific forum.
- **Forums_ForumRoles** - stores relationships between forums and roles as well as a permission type. Each entry in this table indicates that users of the given role have the specified permission for a forum.
- **Forums_ForumSubscription** - stores information about user subscriptions to forums.
- **Forums_Attachment** - contains records representing forum post attachments.
- **Forums_UserFavorites** - stores relationships between users and the forums or posts marked by them as favorite.



8.26.17.3 API classes

If you are not familiar with database table data management through Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

The classes of the Forums module can be found in the **CMS.Forums** namespace.

Forums_ForumGroup table API:

- **ForumGroupInfo** - represents one forum group object.
- **ForumGroupInfoProvider** - provides management functionality for forum groups.

Forums_Forum table API:

- **ForumInfo** - represents one forum object.
- **ForumInfoProvider** - provides management functionality for forums.

Forums_ForumPost table API:

- **ForumPostInfo** - represents one forum post.
- **ForumPostInfoProvider** - provides management functionality for forum posts.

Forums_ForumModerator table API:

- **ForumModeratorInfo** - represents a relationship between a user and a forum, indicating that the user is the forum's moderator.
- **ForumModeratorInfoProvider** - provides management functionality for user-forum moderator relationships.

Forums_ForumRoles table API:

- **ForumRoleInfo** - represents a relationship between a forum, role and permission.
- **ForumRoleInfoProvider** - provides management functionality for forum-role relationships.

Forums_ForumSubscription table API:

- **ForumSubscriptionInfo** - represents a subscription of a user to a forum.
- **ForumSubscriptionInfoProvider** - provides management functionality for forum subscriptions.

Forums_Attachment table API:

- **ForumAttachmentInfo** - represents one forum post attachment.
- **ForumAttachmentInfoProvider** - provides management functionality for forum post attachments.

Forums_UserFavorites table API:

- **ForumUserFavoritesInfo** - represents a relationship between a user and a forum or post marked as favorite.
- **ForumUserFavoritesInfoProvider** - provides management functionality for forum favorites.

Other classes:

- **ForumContext** - provides forum-related information for the current user.

8.26.17.4 API examples

8.26.17.4.1 Overview

These topics show examples of how the API of the Forums module can be used:

- Managing forum groups
- Managing forums
- Managing forum posts

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Community\Forums\Default.aspx.cs***.

The forum API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.GlobalHelper;
using CMS.CMSHelper;
using CMS.SiteProvider;
using CMS.Forums;
```

8.26.17.4.2 Managing forum groups

The following example creates a forum group.

```
private void CreateForumGroup()
{
    // Create new forum group object
    ForumGroupInfo newGroup = new ForumGroupInfo();

    // Set the properties
    newGroup.GroupDisplayName = "My new group";
    newGroup.GroupName = "MyNewGroup";
    newGroup.GroupSiteID = CMSContext.CurrentSiteID;
    newGroup.GroupAuthorDelete = true;
    newGroup.GroupAuthorEdit = true;
    newGroup.GroupDisplayEmails = true;
```

```
    // Save the forum group
    ForumGroupInfoProvider.SetForumGroupInfo(newGroup);
}
```

The following example gets and updates a forum group.

```
private bool GetAndUpdateForumGroup()
{
    // Get the forum group
    ForumGroupInfo updateGroup = ForumGroupInfoProvider.GetForumGroupInfo
("MyNewGroup", CMSContext.CurrentSiteID);
    if (updateGroup != null)
    {
        // Update the properties
        updateGroup.GroupDisplayName = updateGroup.GroupDisplayName.ToLower();

        // Save the changes
        ForumGroupInfoProvider.SetForumGroupInfo(updateGroup);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates forum groups.

```
private bool GetAndBulkUpdateForumGroups()
{
    // Prepare the parameters
    string where = "GroupName LIKE N'MyNewGroup%'";
    string orderBy = "";
    string columns = "";
    int topN = 10;

    // Get the data
    DataSet groups = ForumGroupInfoProvider.GetGroups(where, orderBy, topN,
columns);
    if (!DataHelper.DataSourceIsEmpty(groups))
    {
        // Loop through the individual items
        foreach (DataRow groupDr in groups.Tables[0].Rows)
        {
            // Create object from DataRow
            ForumGroupInfo modifyGroup = new ForumGroupInfo(groupDr);

            // Update the properties
            modifyGroup.GroupDisplayName = modifyGroup.GroupDisplayName.ToUpper();

            // Save the changes
```

```
            ForumGroupInfoProvider.SetForumGroupInfo(modifyGroup);
        }

        return true;
    }

    return false;
}
```

The following example deletes a forum group.

```
private bool DeleteForumGroup()
{
    // Get the forum group
    ForumGroupInfo deleteGroup = ForumGroupInfoProvider.GetForumGroupInfo
("MyNewGroup", CMSContext.CurrentSiteID);

    // Delete the forum group
    ForumGroupInfoProvider.DeleteForumGroupInfo(deleteGroup);

    return (deleteGroup != null);
}
```

8.26.17.4.3 Managing forums

The following example creates a forum.

```
private bool CreateForum()
{
    // Get the forum group
    ForumGroupInfo group = ForumGroupInfoProvider.GetForumGroupInfo("MyNewGroup",
CMSContext.CurrentSiteID);

    if (group != null)
    {
        // Create new forum object
        ForumInfo newForum = new ForumInfo();

        // Set the properties
        newForum.ForumDisplayName = "My new forum";
        newForum.ForumName = "MyNewForum";
        newForum.ForumGroupID = group.GroupID;
        newForum.ForumSiteID = group.GroupSiteID;
        newForum.AllowAccess = SecurityAccessEnum.AllUsers;
        newForum.AllowAttachFiles = SecurityAccessEnum.AuthenticatedUsers;
        newForum.AllowPost = SecurityAccessEnum.AllUsers;
        newForum.AllowReply = SecurityAccessEnum.AllUsers;
        newForum.AllowSubscribe = SecurityAccessEnum.AllUsers;
        newForum.ForumOpen = true;
        newForum.ForumModerated = false;
```

```
        newForum.ForumThreads = 0;
        newForum.ForumPosts = 0;

        // Save the forum
        ForumInfoProvider.SetForumInfo(newForum);

        return true;
    }

    return false;
}
```

The following example gets and updates a forum.

```
private bool GetAndUpdateForum()
{
    // Get the forum
    ForumInfo updateForum = ForumInfoProvider.GetForumInfo("MyNewForum",
CMSContext.CurrentSiteID);
    if (updateForum != null)
    {
        // Update the properties
        updateForum.ForumDisplayName = updateForum.ForumDisplayName.ToLower();

        // Save the changes
        ForumInfoProvider.SetForumInfo(updateForum);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates forums.

```
private bool GetAndBulkUpdateForums()
{
    // Prepare the parameters
    string where = "ForumName LIKE N'MyNewForum%'";
    string orderBy = "";
    string columns = "";
    int topN = 10;

    // Get the data
    DataSet forums = ForumInfoProvider.GetForums(where, orderBy, topN, columns);
    if (!DataHelper.DataSourceIsEmpty(forums))
    {
        // Loop through the individual items
        foreach (DataRow forumDr in forums.Tables[0].Rows)
        {
            // Create object from DataRow
```

```
            ForumInfo modifyForum = new ForumInfo(forumDr);

            // Update the properties
            modifyForum.ForumDisplayName = modifyForum.ForumDisplayName.ToUpper();

            // Save the changes
            ForumInfoProvider.SetForumInfo(modifyForum);
        }

        return true;
    }

    return false;
}
```

The following example deletes a forum.

```
private bool DeleteForum()
{
    // Get the forum
    ForumInfo deleteForum = ForumInfoProvider.GetForumInfo("MyNewForum",
CMSContext.CurrentSiteID);

    // Delete the forum
    ForumInfoProvider.DeleteForumInfo(deleteForum);

    return (deleteForum != null);
}
```

8.26.17.4.4 Managing forum posts

The following example creates a forum post.

```
private bool CreateForumPost()
{
    // Get the forum
    ForumInfo forum = ForumInfoProvider.GetForumInfo("MyNewForum",
CMSContext.CurrentSiteID);
    if (forum != null)
    {
        // Create new forum post object
        ForumPostInfo newPost = new ForumPostInfo();

        // Set the properties
        newPost.PostUserID = CMSContext.CurrentUser.UserID;
        newPost.PostUserMail = CMSContext.CurrentUser.Email;
        newPost.PostUserName = CMSContext.CurrentUser.UserName;
        newPost.PostForumID = forum.ForumID;
        newPost.PostTime = DateTime.Now;
        newPost.PostApproved = true;
```

```
        newPost.PostText = "This is my new post";
        newPost.PostSubject = "My new post";

        // Save the forum post
        ForumPostInfoProvider.SetForumPostInfo(newPost);

        return true;
    }

    return false;
}
```

The following example gets and updates a forum post.

```
private bool GetAndUpdateForumPost()
{
    // Prepare the parameters
    string where = "PostSubject LIKE N'My new post%'";
    string orderBy = "";
    string columns = "";
    int topN = 10;

    // Get the data
    DataSet posts = ForumPostInfoProvider.GetForumPosts(where, orderBy, topN,
columns);
    if (!DataHelper.DataSourceIsEmpty(posts))
    {
        ForumPostInfo updatePost = new ForumPostInfo(posts.Tables[0].Rows[0]);

        // Update the properties
        updatePost.PostSubject = updatePost.PostSubject.ToLower();

        // Save the changes
        ForumPostInfoProvider.SetForumPostInfo(updatePost);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates forum posts.

```
private bool GetAndBulkUpdateForumPosts()
{
    // Prepare the parameters
    string where = "PostSubject LIKE N'My new post%'";
    string orderBy = "";
    string columns = "";
    int topN = 10;
```

```
    // Get the data
    DataSet posts = ForumPostInfoProvider.GetForumPosts(where, orderBy, topN,
columns);
    if (!DataHelper.DataSourceIsEmpty(posts)
    {
        // Loop through the individual items
        foreach (DataRow postDr in posts.Tables[0].Rows)

        {
            // Create object from DataRow
            ForumPostInfo modifyPost = new ForumPostInfo(postDr);

            // Update the properties
            modifyPost.PostSubject = modifyPost.PostSubject.ToUpper();

            // Save the changes
            ForumPostInfoProvider.SetForumPostInfo(modifyPost);
        }

        return true;
    }

    return false;
}
```

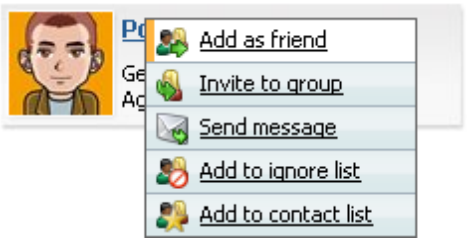
The following example deletes a forum post.

```
private bool DeleteForumPost()
{
    // Prepare the parameters
    string where = "PostSubject LIKE N'My new post%'";
    string orderBy = "";
    string columns = "";
    int topN = 10;

    // Get the data
    DataSet posts = ForumPostInfoProvider.GetForumPosts(where, orderBy, topN,
columns);
    if (!DataHelper.DataSourceIsEmpty(posts))
    {
        // Get the forum post
        ForumPostInfo deletePost = new ForumPostInfo(posts.Tables[0].Rows[0]);

        // Delete the forum post
        ForumPostInfoProvider.DeleteForumPostInfo(deletePost);

        return true;
    }

    return false;
}
```

8.27 Friends

8.27.1 Overview

The Friends module is one of the social-networking features of Kentico CMS. It allows you to add the Facebook-style friendship functionality to your site. The module enables registered website users to request another user's friendship. When the friendship gets approved by the second user, the two users become friends. The process of requesting a friendship is described in the Requesting friendships topic. Requests are sent to users based on e-mail templates, which are listed and described in the Related e-mail templates topic.



Management of friendships can be performed either by the users themselves, or by the site administrators. Both options are described in the Friends management topic. Users can manage only their own friendships, while users with appropriate permissions (described in the Security topic) can manage friendships of all users in the system.

The module comes with several web parts. Using these, you can add the friendship management functionality to the live site. All web parts related to the module are listed in the Available web parts topic.

By default, becoming friends brings just some improvements to the way the users can communicate and publish details about themselves, as described in the Examples of use topic. However, the friendship functionality may be leveraged largely in your custom code, enabling you to use these relationships between users for your specific purposes.

In the Friends internals and API sub-chapter, you can find information about database tables and classes that are used by the Friends module, as well as several examples of how friendships can be managed using Kentico CMS API.

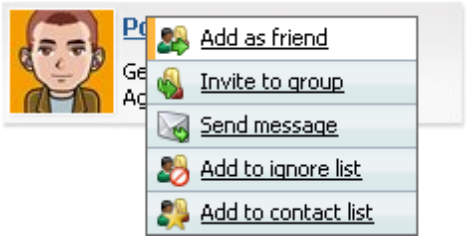
8.27.2 Requesting friendships

How to make the request?

There are several ways how users can request other users' friendship:

Context menus

As described in Development -> Document types and transformations -> Context menus in transformations, the **Users viewer** web part may be equipped with a context menu when an appropriately written transformation is used. This context menu contains the **Add as friend** option, enabling registered website users to request this user's friendship.

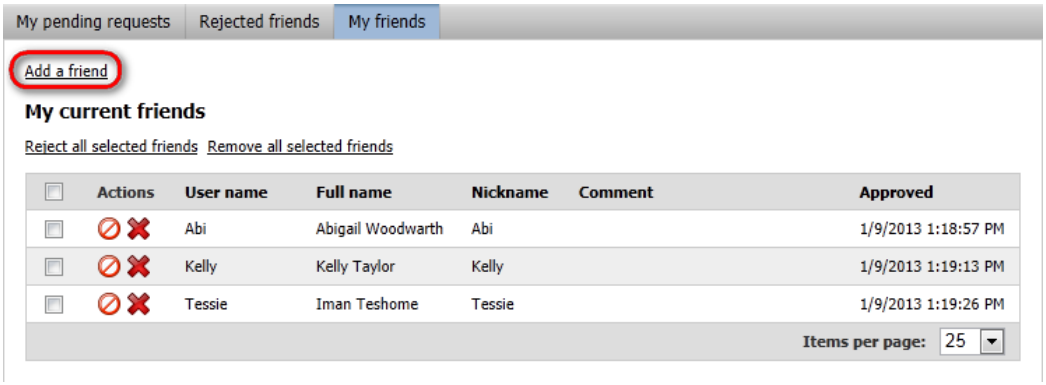


Request friendship and My friends web parts

The **Request friendship** web part is just a tiny link with customizable text. Clicking it opens the **Add a new friend** pop-up dialog.

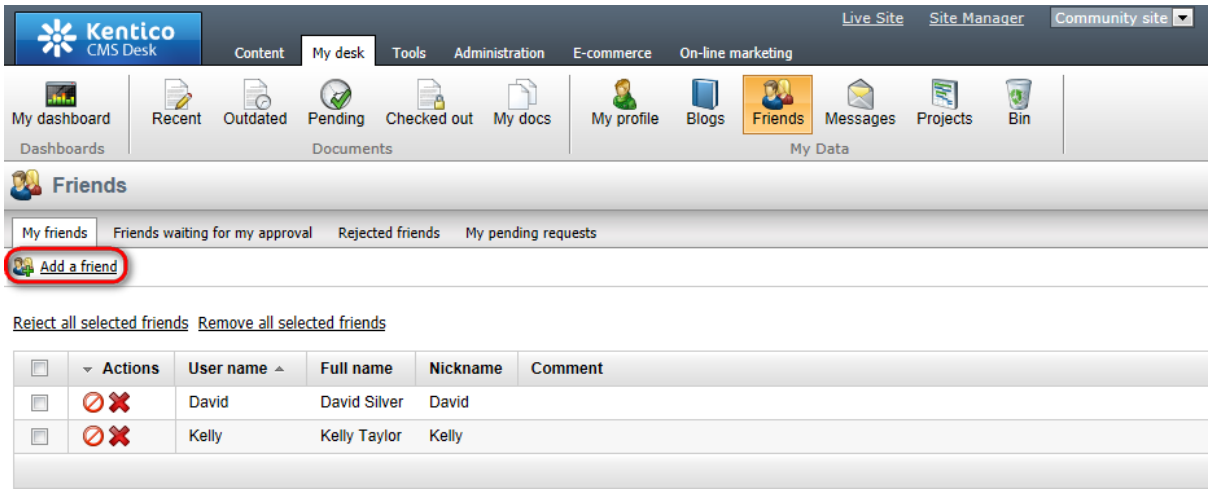
Request friendship

Requesting friendships on the live site is also possible via the **My friends** web part. It contains the **Add a friend** link, which opens the same **Add a new friend** dialog mentioned above.



CMS Desk -> My Desk -> Friends

Users with access to **CMS Desk** can request friendships from within **My desk -> Friends**. Here again, they can find the **Add a friend** button, which opens the **Add a new friend** dialog.

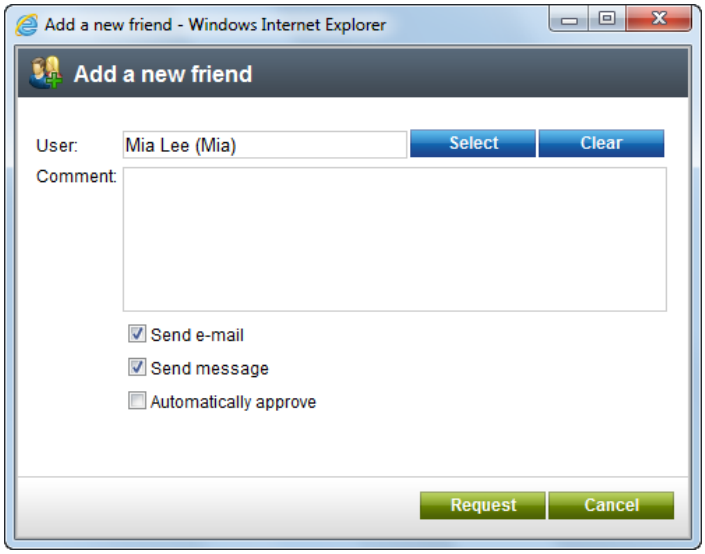


Add a new friend dialog

After clicking one of the options mentioned above, the **Add a new friend** dialog appears. In the dialog, you can specify the following:

- **User** - the user whose friendship you are requesting. This options is not available when requesting a friendship from the context menu as friendship of the user whose context menu was displayed is requested in this case.
- **Comment** - comment sent to the user along with the friendship request.
- **Send e-mail** - the user will be notified by a standard e-mail message.
- **Send message** - the user will be notified by a Messaging module message.
- **Automatically approve** - global administrators and users with the **Manage** permission for the Friends module can use this check-box, which creates the requested friendship without the other user's approval; not available on the live site.

In case that the **Send e-mail** or **Send message** options are enabled, the **Friends - Friend request** e-mail template mentioned in the section below is used for the notification e-mail or Messaging module message. When the **Automatically approve** option is enabled along with the previously mentioned ones, the **Friends - Friend approval** template is used.



8.27.3 Related e-mail templates

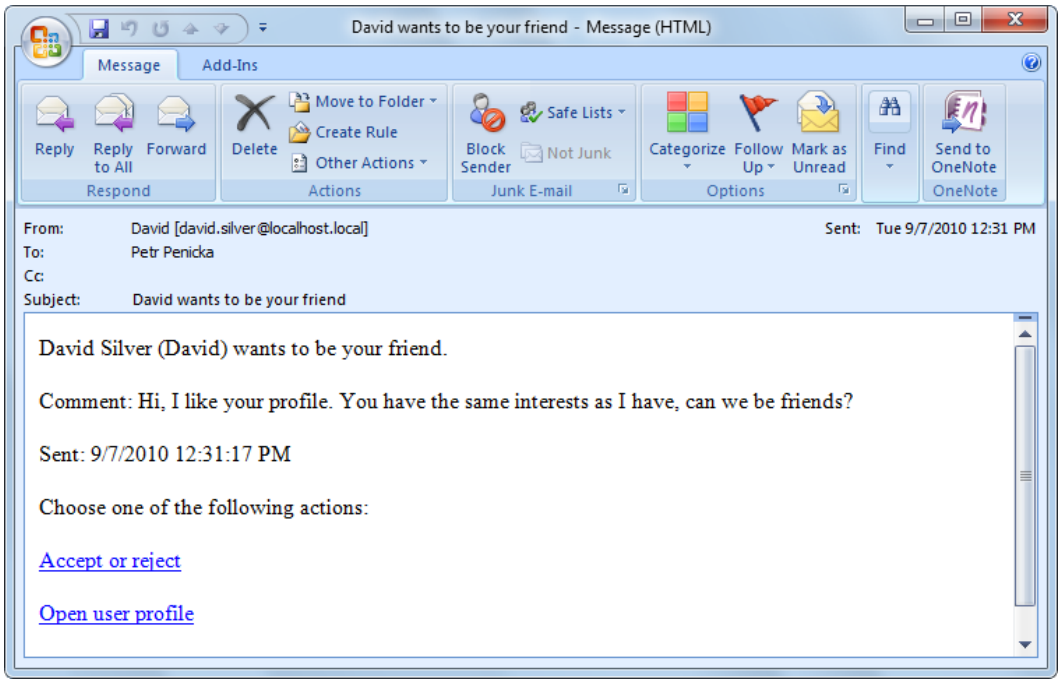
In **Site manager -> Administration -> E-mail templates**, you can find the following e-mail templates related to the Friends module:

- **Friends - Friend approval** - template for e-mail or message confirming that a user approved your friendship request.
- **Friends - Friend rejection** - template for e-mail or message confirming that a user rejected your friendship request.
- **Friends - Friend request** - template for e-mail or message notifying you about the fact that some user requested your friendship.

If you choose to **Edit** (✎) some of the templates, you will see two large text fields:

- **Text** - text of the template used for e-mails.
- **Plain text** - text of the template used for Messaging module messages.

Links in these e-mails need to be handled by a special page containing the **Friendship management** web part, which is a special web part that ensures handling of these requests.



Macros in friendship e-mail templates

In both fields, you can use the following specific context macros to include dynamic values in their text:

- **{% managementurl %}** - returns the frienship management page URL. See **Friend management path** in Settings for more details.
- **{% profileurl %}** - returns URL of the user who requested the friendship.
- **{% formattedsendername %}** - returns the name of the user who is requesting the friendship in format *username (fullname)*.
- **{% formattedrecipientname %}** - returns the name of the user whose friendship is being requested in format *username (fullname)*.

You can also access the following objects and their properties (e.g. *{% Sender.UserName %}*):

- **{% Sender %}** - *UserInfo* object of the user who is requesting the friendship.
- **{% Recipient %}** - *UserInfo* object of the user whose friendship is being requested.
- **{% Friendship %}** - *FriendInfo* object containing information about the current friendship.

Besides these special ones, you can also use all other standard macro expressions in the templates. See the Development -> Macro expressions chapter of this guide for more information about macro expressions in Kentico CMS.

8.27.4 Friends management

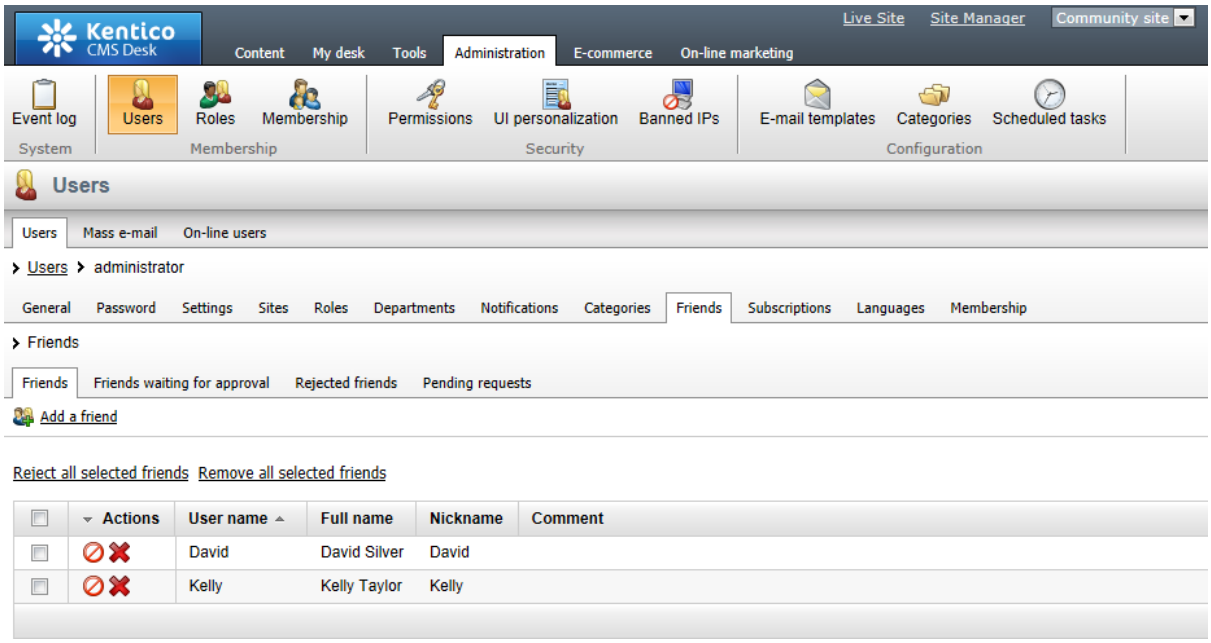
Users with access to **CMS Desk -> My desk -> Friends** can manage their own friendships from within this section of the administration interface.

Site administrators can manage all users' friends in both **CMS Desk** and **Site Manager**, in the **Administration -> Users** section, after choosing **Edit** (✎) next to some user and switching to their **Friends** tab.

In both cases, the four tabs described below are available. On the live site, the same functionality can be provided by the web parts described in the Available web parts topic.

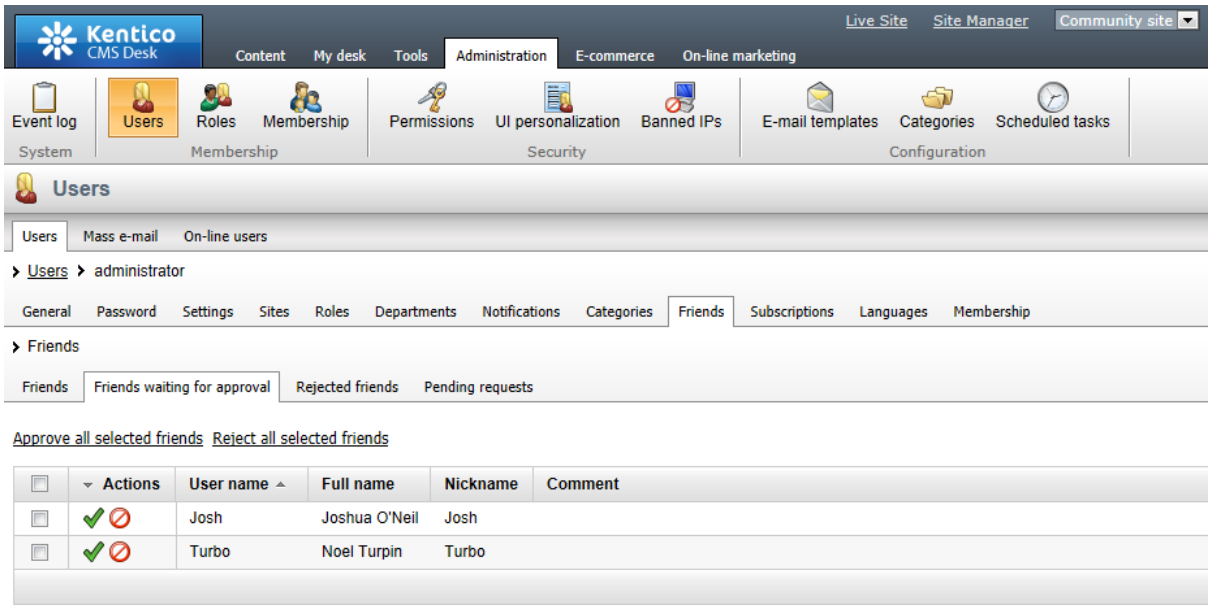
Friends (My friends)

On this tab, you can see a list of all your current friends. New friendships can be requested using **Add a friend**. You also can **Remove** (✖) or **Reject** (⊘) friends in the list. The difference between the two is that rejected friends can't request your friendship anymore while they are in the rejected status, while removed friends can request the friendship again. By checking some of the check-boxes and clicking one of the **Reject all selected friends** and **Remove all selected friends** buttons, you can reject or remove more friends at once.



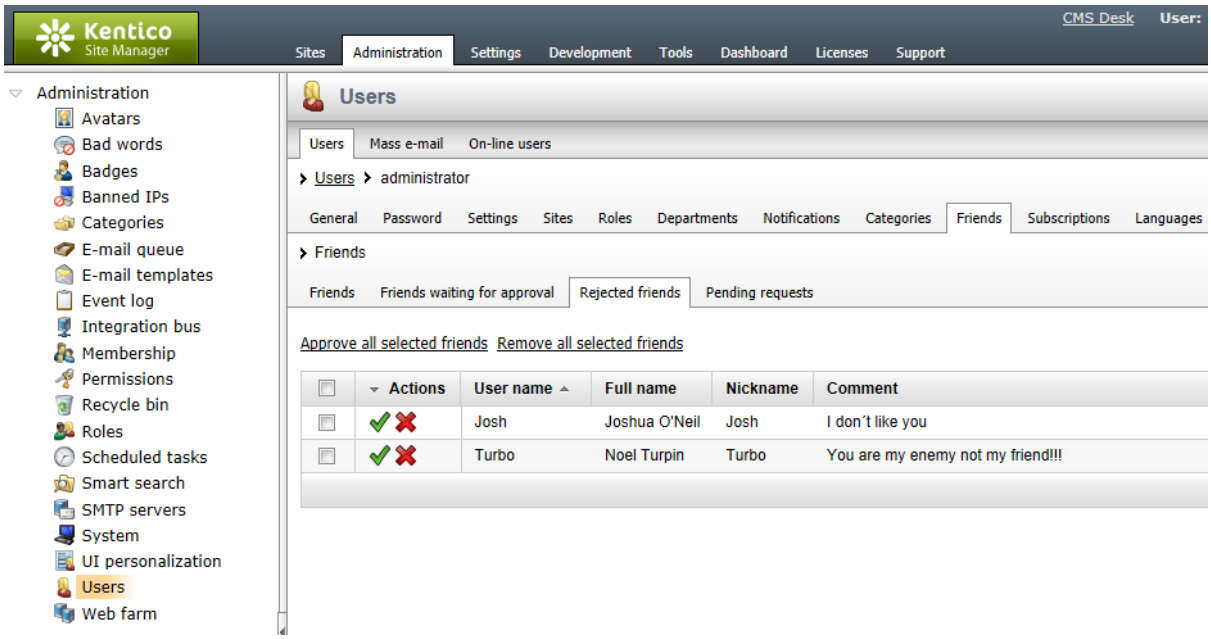
Friends waiting for approval (Friends waiting for my approval)

On this tab, you can see a list of all users who requested your friendship. You can either **Approve** (✔) or **Reject** (⊘) their request. By checking some of the check-boxes and clicking one of the **Approve all selected friends** and **Reject all selected friends** links, you can approve or reject more requests at once.



Rejected friends

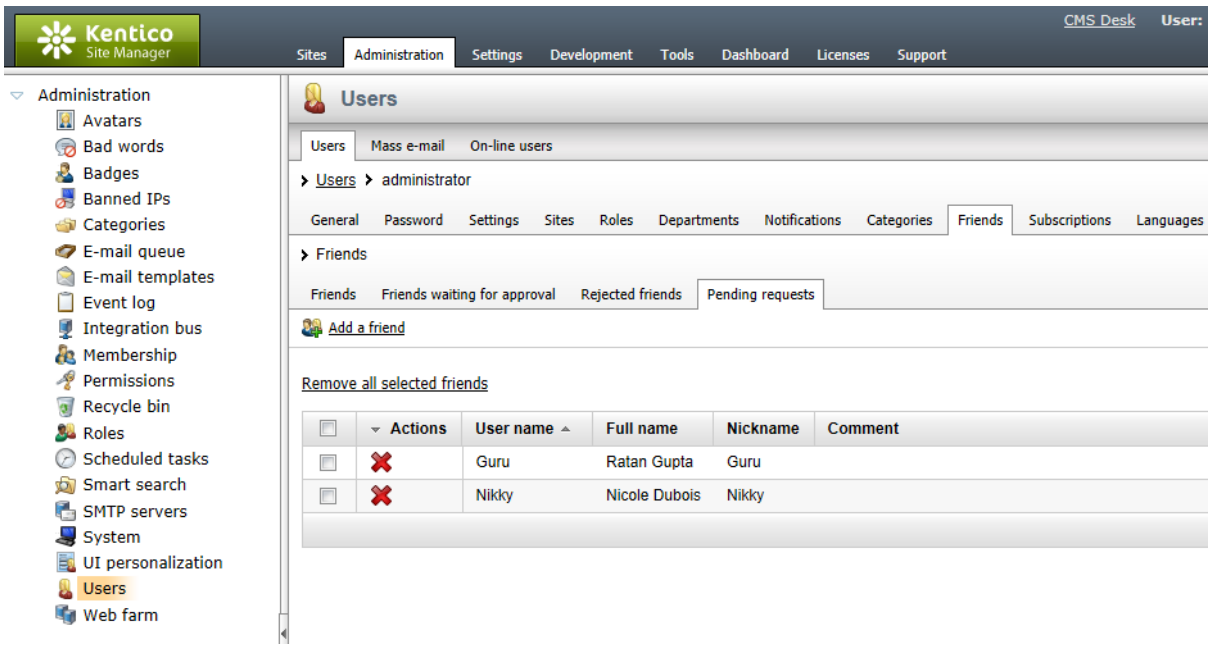
On this tab, you can see a list of all users whose friendship you rejected. Once in the rejected status, the user can't request your friendship anymore. You can either **Approve** (✔) some user's friendship, which makes them your friend, or **Remove** (✖) the user from the list of rejected, which enables them to request your friendship again. By checking some of the check-boxes and clicking one of the **Approve all selected friends** or **Remove all selected friends**, you can approve or remove more users at once.



Pending requests (My pending requests)

On this tab, you can see a list of friends whose friendship you requested. New friendships can be requested using **Add a friend**.You can cancel a request by clicking the **Remove** (✖) icon or you can

select more users using the check-boxes and click the **Remove all selected friends** link, which cancels more friendship requests at once.

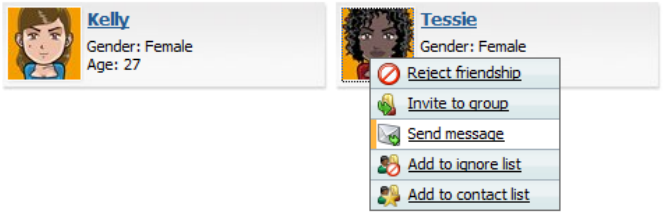


8.27.5 Available web parts

The Friends module comes with the following web parts. All of them are located in the **Community -> Friends** web part category. This page brings just a brief overview of them. For a detailed description of each web part's properties please see Kentico CMS Web parts reference.

Friends viewer

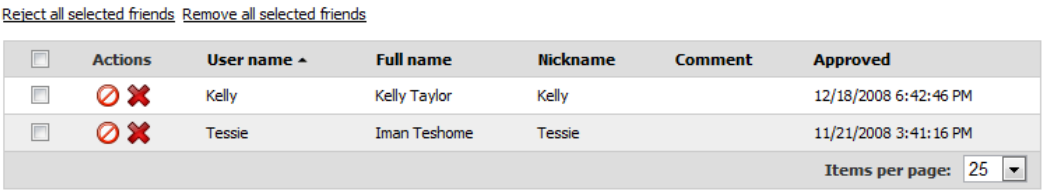
This web part can typically be used on user's profile page for viewing their friends. If you right-click the user's avatar image, you will be offered several tasks, as you can see in the screenshot below.



The following web parts are actually on-site equivalents of the friends management administration interface described in the Friends management topic:

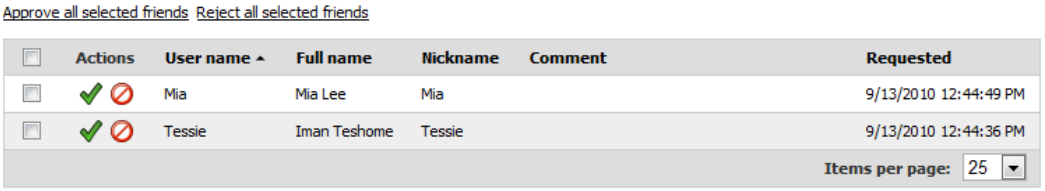
Friends list

This web part displays a list of all friends of the current user.



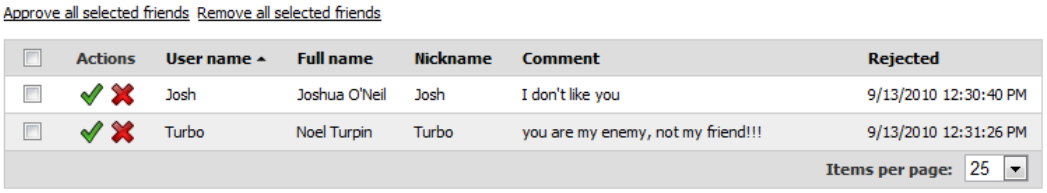
Friends waiting for approval

This web part displays a list of all users waiting for the current user's friendship approval.



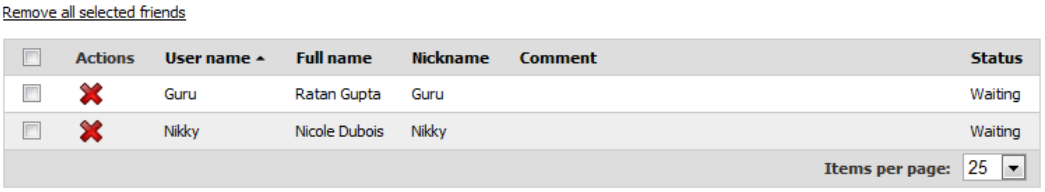
Rejected friends

This web part displays a list of all users whose friendship the current user rejected. These users can't request the current user's friendship while they are in this list.



My pending requests

This web part displays a list of all users whose friendship the current user requested.

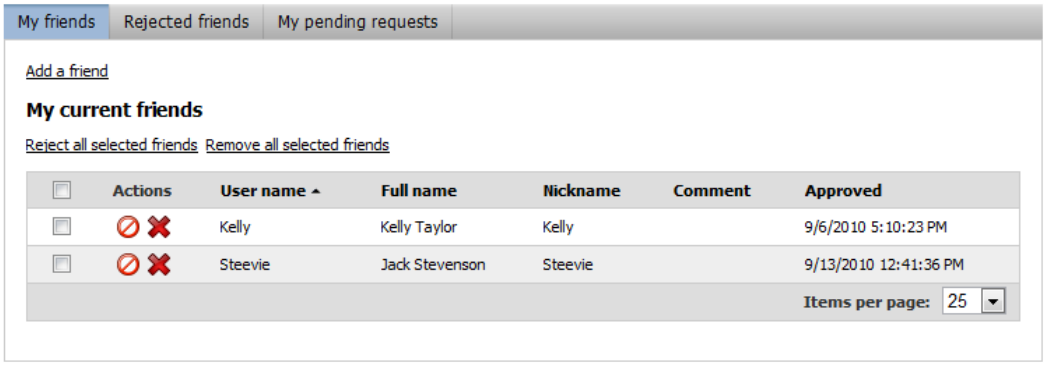


Request friendship

This web part displays only a link that, when clicked, opens the **Add a new friend** dialog.

My friends

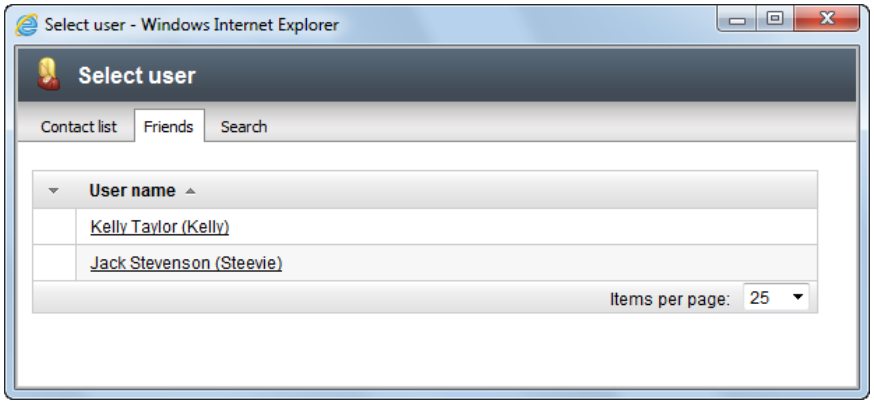
This web part combines the five previously mentioned web parts in one single web part. It's functionality is fully equal to the UI in **CMS Desk -> My Desk -> Friends**.



8.27.6 Examples of use

Messaging module

When selecting a recipient of a messaging module message, you can conveniently switch to the **Friends** tab and select one of your friends just by clicking their user name.



Fields visibility

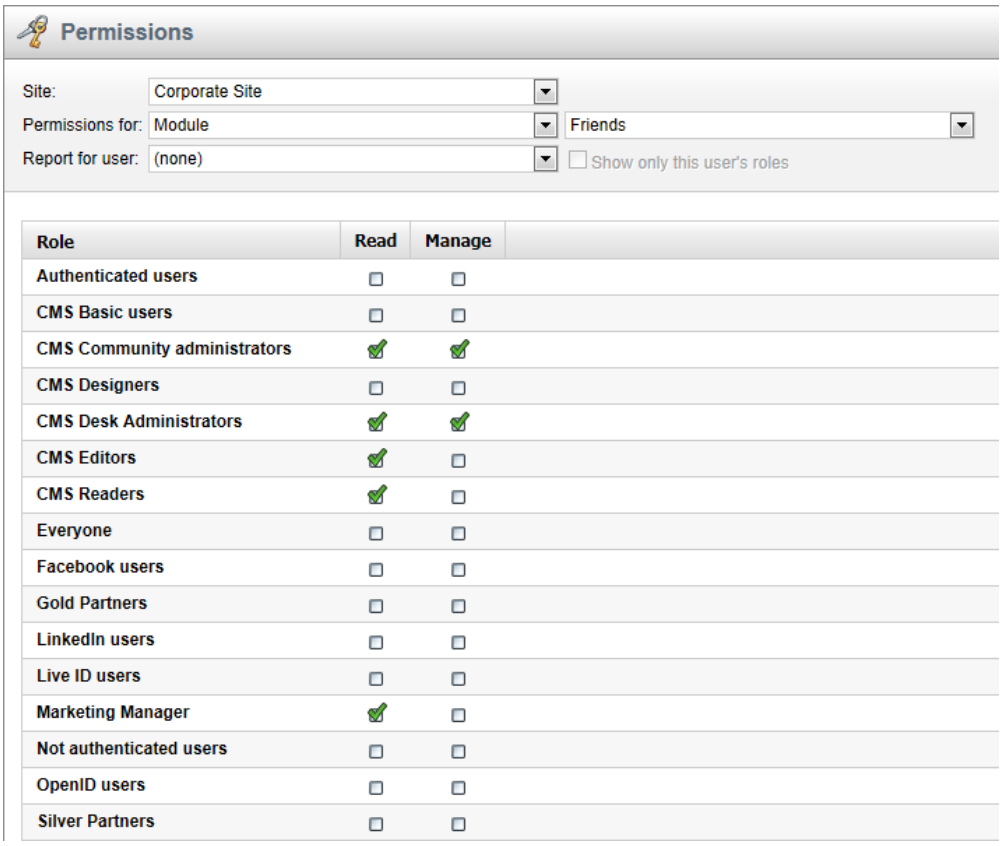
Users can set the visibility of some fields on their public profile to **Display to friends**, which makes the field visible only to their friends. For more information on this feature, please refer to Development -> Membership, security and permissions -> Custom fields visibility.

8.27.7 Security

Permissions of the Friends module can be set in both **Site manager** and **CMS Desk**, in the **Administration -> Permissions** section. There, you need to select the **Modules -> Friends** permission matrix.

The following permissions can be assigned to particular roles:

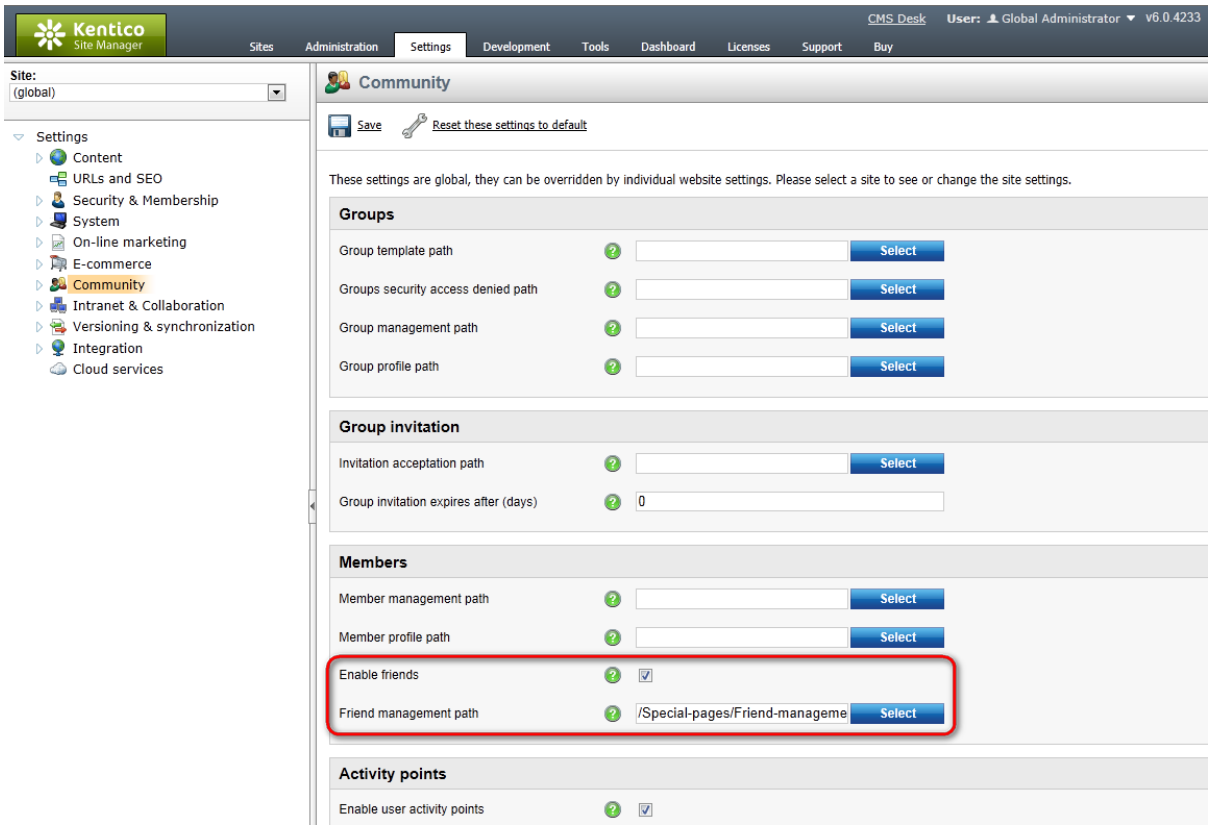
- **Manage** - members of the role can manage Friends of particular users in *CMS Desk -> Administration -> Users*
- **Read** - members of the role can view friends of particular users in *CMS Desk -> Administration -> Users*



8.27.8 Settings

The following two settings available in **Site Manager -> Settings -> Community** are related to the Friends module:

Enable friends	Indicates if the Friends functionality should be available in other modules on the live site.
Friend management path	Node alias path of the page on that the **Friendship management** web part is placed. This is a special web part that handles friendship management actions from friendship request e-mails. If some user receives an e-mail with friendship request, there are links for friendship approval or rejection included in the mail. When the user clicks one of the links, they are redirected to this page and the user guid and type of action is transferred in form of querystring parameters. The **Friendship management** web part processes the received parameters and performs the necessary tasks for friendship approval or rejection. On the **Community Starter Site**, the *Special-pages/Friend-management* page serves exactly this purpose. More information on how such page can be created can be found in Community Site Guide -> Part 2 -> Creating the Special pages -> Creating the friend management page.



8.27.9 Friends internals and API

8.27.9.1 Overview

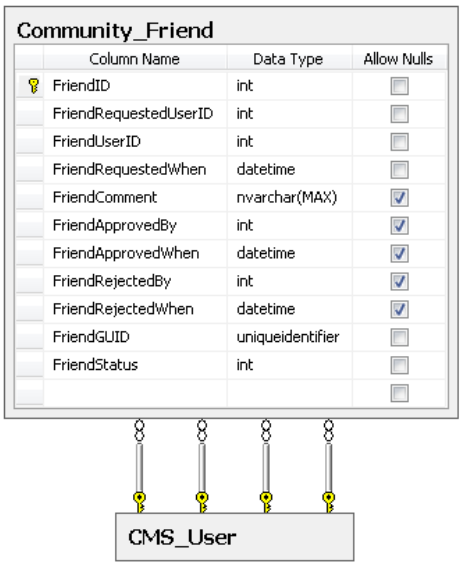
In this chapter, you will learn which database tables and API classes are used in the Abuse report module. You will also see the most common API examples.

Advanced API examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all methods from the module classes, please refer to Kentico CMS API Reference.

8.27.9.2 Database tables

The Friends module uses the following database tables:

- **Community_Friend** - contains friendship records, while each row in the table represents a friendship between two users.
- **CMS_User** - contains records representing user accounts.



8.27.9.3 API classes

If you are not familiar with database table data management through Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

The classes of the Friends module can be found in the **CMS.Community** namespace.

CommunityFriend table API:

- **FriendInfo** - represents one friendship object.
- **FriendInfoProvider** - provides friendship management functionality.

8.27.9.4 API examples

8.27.9.4.1 Overview

The following topic show examples of how the API of the Friends module can be used:

- Managing friendships

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project***

folder>\CMSAPIExamples\Code\Community\Friends\Default.aspx.cs.

The Friends API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.GlobalHelper;
using CMS.UIControls;
using CMS.CMSHelper;
using CMS.SiteProvider;
using CMS.Community;
```

8.27.9.4.2 Managing friendships

The following example demonstrates how a friendship request can be created.

```
private bool RequestFriendship()
{
    // First create a new user which the friendship request will be sent to
    UserInfo newFriend = new UserInfo();
    newFriend.UserName = "MyNewFriend";
    newFriend.FullName = "My new friend";
    newFriend.UserGUID = Guid.NewGuid();

    UserInfoProvider.SetUserInfo(newFriend);

    // Create new friend object
    FriendInfo newFriendship = new FriendInfo();

    // Set the properties
    newFriendship.FriendUserID = CMSContext.CurrentUser.UserID;
    newFriendship.FriendRequestedUserID = newFriend.UserID;
    newFriendship.FriendRequestedWhen = DateTime.Now;
    newFriendship.FriendComment = "Sample friend request comment.";
    newFriendship.FriendStatus = FriendshipStatusEnum.Waiting;
    newFriendship.FriendGUID = Guid.NewGuid();

    // Save the friend
    FriendInfoProvider.SetFriendInfo(newFriendship);

    return true;
}
```

The following example approves the friendship requested in the previous code example.

```
private bool ApproveFriendship()
{
```

```
    // Get the users involved in the friendship
    UserInfo user = CMSContext.CurrentUser;
    UserInfo friend = UserInfoProvider.GetUserInfo("MyNewFriend");

    if (friend != null)
    {
        // Get the friendship with current user
        FriendInfo updateFriendship = FriendInfoProvider.GetFriendInfo
(user.UserID, friend.UserID);
        if (updateFriendship != null)
        {
            // Set its properties
            updateFriendship.FriendStatus = FriendshipStatusEnum.Approved;
            updateFriendship.FriendRejectedBy = 0;
            updateFriendship.FriendApprovedBy = user.UserID;
            updateFriendship.FriendApprovedWhen = DateTime.Now;

            // Save the changes to database
            FriendInfoProvider.SetFriendInfo(updateFriendship);

            return true;
        }
    }

    return false;
}
```

The following example rejects the friendship requested in the first code example on this page.

```
private bool RejectFriendship()
{
    // Get the users involved in the friendship
    UserInfo user = CMSContext.CurrentUser;
    UserInfo friend = UserInfoProvider.GetUserInfo("MyNewFriend");

    if (friend != null)
    {
        // Get the friendship with current user
        FriendInfo updateFriendship = FriendInfoProvider.GetFriendInfo
(user.UserID, friend.UserID);

        // Set its properties
        updateFriendship.FriendStatus = FriendshipStatusEnum.Rejected;
        updateFriendship.FriendApprovedBy = 0;
        updateFriendship.FriendRejectedBy = user.UserID;
        updateFriendship.FriendRejectedWhen = DateTime.Now;

        // Save the changes to database
        FriendInfoProvider.SetFriendInfo(updateFriendship);

        return true;
    }
```

```
    else
    {
        return false;
    }
}
```

The following example demonstrates how multiple friendships can be get based on a WHERE condition and bulk updated.

```
private bool GetAndBulkUpdateFriends()
{
    // Get the user
    UserInfo friend = UserInfoProvider.GetUserInfo("MyNewFriend");

    if (friend != null)
    {
        // Prepare the parameters
        string where = "FriendRequestedUserID = " + friend.UserID;

        // Get the data
        DataSet friends = FriendInfoProvider.GetFriends(where, null);
        if (!DataHelper.DataSourceIsEmpty(friends))
        {
            // Loop through the individual items
            foreach (DataRow friendDr in friends.Tables[0].Rows)
            {
                // Create object from DataRow
                FriendInfo modifyFriend = new FriendInfo(friendDr);

                // Update the properties
                modifyFriend.FriendStatus = FriendshipStatusEnum.Approved;
                modifyFriend.FriendRejectedBy = 0;
                modifyFriend.FriendApprovedBy = CMSContext.CurrentUser.UserID;
                modifyFriend.FriendApprovedWhen = DateTime.Now;

                // Save the changes
                FriendInfoProvider.SetFriendInfo(modifyFriend);
            }

            return true;
        }
    }

    return false;
}
```

The following example shows how friendships can be selected based on a WHERE condition and bulk deleted.

```
private bool DeleteFriends()
{
```

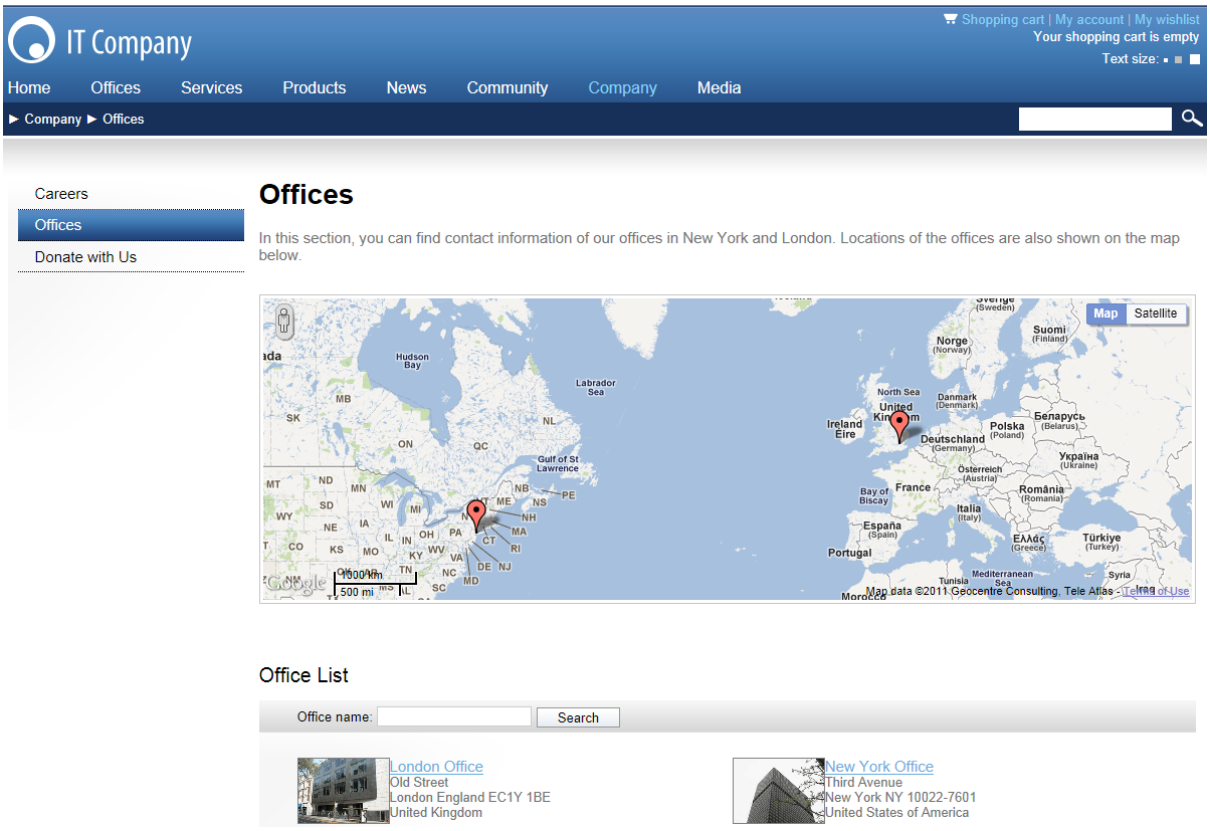
```
    // Get the user
    UserInfo friend = UserInfoProvider.GetUserInfo("MyNewFriend");

    if (friend != null)
    {
        // Prepare the parameters
        string where = "FriendRequestedUserID = " + friend.UserID;

        // Get all user's friendships
        DataSet friends = FriendInfoProvider.GetFriends(where, null);
        if (!DataHelper.DataSourceIsEmpty(friends))
        {
            // Delete all the friendships
            foreach (DataRow friendDr in friends.Tables[0].Rows)
            {
                FriendInfo deleteFriend = new FriendInfo(friendDr);

                deleteFriend.Delete();
            }
        }
        else
        {
            // Change the info message
            apiDeleteFriends.InfoMessage = "The user 'My new friend' doesn't have
any friends. The user has been deleted.";
        }

        // Finally delete the user "My new friend"
        UserInfoProvider.DeleteUser(friend);

        return true;
    }

    return false;
}
```

8.28 Geo mapping

8.28.1 Overview

The Geo mapping module allows you to display maps on your site. You can display also content on these maps.

The module has no administration interface and consists only of nine web parts and three widgets derived from the static group of these web parts. The web parts and widgets can be found under the **Maps** category in the web part and widget selection dialogs.



You can see the web parts configured and working on the sample Corporate Site, under the **/Examples/Webparts/Maps** node of the content tree.

What purpose it has

The module can be used for many scenarios in which a geographical position is the key information:

- show your offices
- show your stores
- show your partners
- show real estates you offer
- etc.

You can use it to display virtually any content that has a location. The only requirement is that you tag your content with longitude and latitude.

How to get longitude and latitude for a given place

You can use an on-line service that allows you to enter the country, city and street and shows you the respective longitude and latitude. One such service is available for example at http://world.maporama.com.

Available providers

You can choose from the following map providers:

- Bing Maps (originally Live Maps)
- Google Maps

Each of these providers has its dedicated web parts and a widget. Please click the particular map provider to learn more.

8.28.2 Bing maps

The **Bing maps web parts** enable displaying a map with defined location markers using the Bing maps service. They use the **Bing maps API**, which is described in detail in the following location: http://www.microsoft.com/maps/isdk/ajax/.

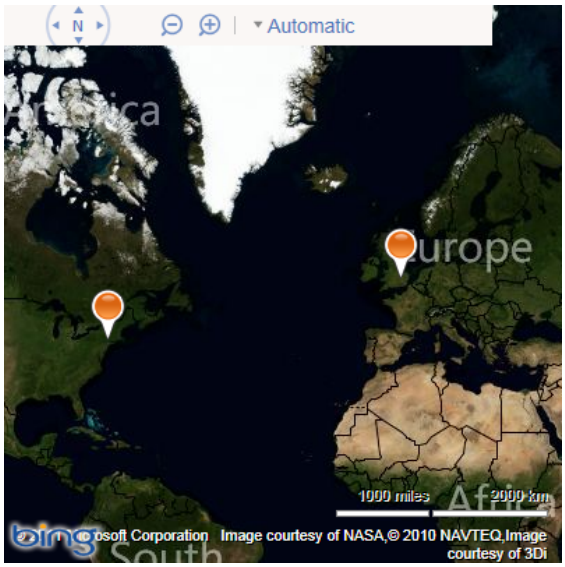
The maps show the location markers and are zoomed according to the settings made by the **Large view scale** property. For a document-related map and for a map with a data source, the center is on the coordinates specified by the **Default latitude** and **Default longitude** properties of the corresponding web part. However, a static map has its center on the location marker.

Static Bing maps

The **Static Bing maps** web part is used. Geographical location is displayed based on manually entered coordinates. Only one location can be displayed at a time. Learn here how to display a location marker on the map.
In the screenshot below, you can see how the web part looks on page load.

Document-related Bing maps

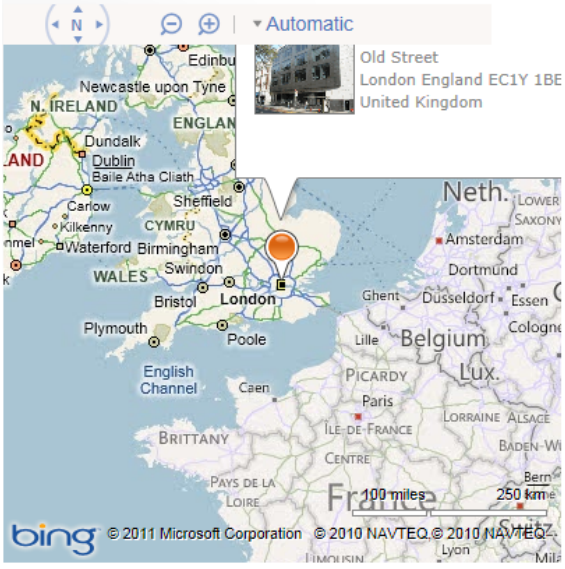
The **Bing maps** web part is used. Geographical location is displayed dynamically based on the properties of the document. More locations can be displayed at a time. Learn here how to display location markers on the map.
In the screenshot below, you can see how the web part looks on page load.



Bing maps with a data source

The **Basic Bing maps** web part in combination with a data source web part is used. Geographical location is displayed dynamically based on the data source, e.g. documents data source, query data source, xml data source etc. More locations can be displayed at a time. Learn here how to display location markers on the map.

The screenshot below shows how any of the web parts looks when one of the location markers is clicked. Regardless of its type, the map is zoomed according to the value of the **Detailed view scale** property and a tooltip is displayed. It shows the heading loaded from a document type field specified in the **Tooltip field** property and the rest of the content loaded using a transformation specified in the **Transformation** property.



If you enable the **Enable keyboard shortcuts** property, the following shortcuts can be used to control the map:

- R - switch to Road view.
- A - switch to Aerial view.
- H - switch to Aerial view with labels.
- O - switch to Bird's eye view.
- B - switch to Bird's eye view with labels.

- Plus Sign (+) - zoom the map in.
- Minus Sign (-) - zoom the map out.

Detailed descriptions of all properties of the web parts can be found in the Kentico CMS Web Parts Reference or directly in the web part properties dialog after clicking the **Documentation** link at the top right corner of the dialog.

Please note

Certain Bing maps services, e.g. location translations, require that you have your Bing maps account. As a Bing map account holder, you can create keys to use these services.

The **Bing maps widget** is derived from the **Static Bing maps** web part. It provides the same functionality and look as this web part, while only a limited set of properties can be configured.

8.28.3 Google maps

The **Google maps web parts** enable displaying a map with defined location markers using the Google maps service. They use the **Google maps API v3**, which is described in detail in the following location: http://code.google.com/intl/cs-CZ/apis/maps/documentation/v3/controls.html.

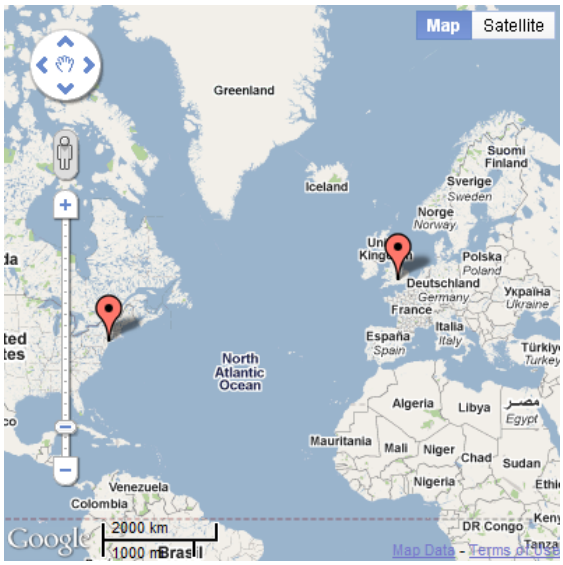
The maps show the location markers and are zoomed according to the settings made by the **Large view scale** property. For a document-related map and for a map with a data source, the center is on the coordinates specified by the **Default latitude** and **Default longitude** properties of the corresponding web part. However, a static map has its center on the location marker.

Static Google maps

The **Static Google maps** web part is used. Geographical location is displayed based on manually entered coordinates. Only one location can be displayed at a time. Learn here how to display a location marker on the map.
In the screenshot below, you can see how the web part looks on page load.

Document-related Google maps

The **Google Maps** web part is used. Geographical location is displayed dynamically based on the properties of the document. More locations can be displayed at a time. Learn here how to display location markers on the map.
In the screenshot below, you can see how the web part looks on page load.



Google maps with a data source

The **Basic Google maps** web part in combination with a data source web part is used. Geographical location is displayed dynamically based on the data source, e.g. documents data source, query data source, xml data source etc. More locations can be displayed at a time. Learn here how to display location markers on the map.

The screenshot below shows how any of the web parts looks when one of the location markers is

clicked. Regardless of its type, the map is zoomed according to the value of the **Detailed view scale** property and a tooltip is displayed. It shows the heading loaded from a document type field specified in the **Tooltip field** property and the rest of the content loaded using a transformation specified in the **Transformation** property.

If you enable the **Enable keyboard shortcuts** property, the following shortcuts can be used to control the map:

- **Up Arrow** - move a little north.
- **Down Arrow** - move a little south.
- **Left Arrow** - move a little west.
- **Right Arrow** - move a little east.

- **Page Up** - move a large step north.
- **Page Down** - move a large step south.
- **Home** - move a large step west.
- **End** - move a large step east.

- **Plus Sign (+)** - zoom the map in.
- **Minus Sign (-)** - zoom the map out.

Detailed descriptions of all properties of the web parts can be found in the Kentico CMS Web Parts Reference or directly in the web part properties dialog after clicking the **Documentation** link at the top right corner of the dialog.

The **Google maps widget** is derived from the **Static Google maps** web part. It provides the same functionality and look as this web part, while only a limited set of properties can be configured.

8.28.4 Example: Static maps

This example will show you how to display one object, e.g. an office, on a map. The **Static Google maps** web part will be used. However, the procedure is identical for the Static Bing maps web part, too. You can use any Kentico CMS sample site for the purpose of this example.

How to do it in general

To display one object on a map, you will need to add a static map web part to a page and configure its properties. Specifically, you will need to fill in **two fields for the geographical coordinates** - one for **longitude** and one for **latitude** - or, alternatively, a **location field** in human-readable form, e.g. an address. This will ensure that the location marker is displayed on the map.

Example

1. Sign in as an administrator to **CMS Desk -> Content** and click the root of the content tree.

2. Click **New** and choose to create a new **Page (menu item)**. Enter *Office* for **Page name**, choose the **Create a blank page with layout** option and select the **Simple** layout. Click **Save**.

3. Now switch to the **Design** tab, add the **Maps/Static/Static Google maps** web part to the **zoneLeft** web part zone and set the following properties:

- **Latitude:** 40.76 (The field containing the latitude position.)
- **Longitude:** -73.98 (The field containing the longitude position.)
- **Tooltip:** New York Office (The field used for the heading of tooltips.)
- **Marker content:** 1290 Avenue of the Americas, New York, 10104, NY (The field containing details of the displayed object. You can use the built-in WYSIWYG editor to edit the details and you can also insert an image.)
- **Large view scale:** 3 (The zoom used on page load.)
- **Icon URL** - if you don't want to use the default location marker icon, you can **alternatively** choose your custom location marker icon. Click the **Select** button to open the **Insert link** dialog and set your custom location marker icon URL. For more details on how to insert a URL, please refer to the WYSIWYG editor -> Insert link chapter.
- **Detailed view scale:** 10 (The zoom used if the location marker is clicked.)
- **Width:** 500 (In pixels.)
- **Height:** 400 (In pixels.)

Geolocation values are valid in the following intervals:

- **Latitude**: from -90 to 90
- **Longitude**: from -180 to 180

Alternatively, you can leave the **Latitude** and **Longitude** fields undefined or set to zero and use the **Location or address** field instead to enter the location value in a human-readable form, e.g. an address.

Important!

Using geolocation values in a human-readable form involves some processing overhead and can even lead to inaccurate results. To ensure maximum performance and accuracy, it is therefore recommended to use the latitude and longitude coordinates when defining a geographical location.

Please note

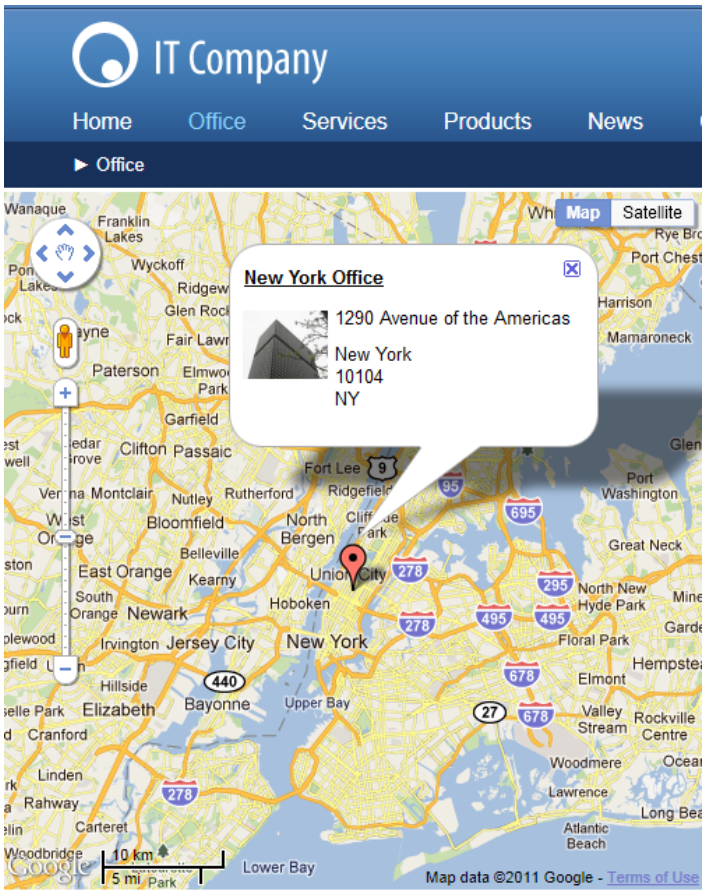
You can enable geolocation translations, i.e. addresses to coordinates, on the server side by checking the **Use server processing** checkbox. If you decide to do so, please check that the maps services query limits (IP-based) are sufficient for your needs.

To ensure maximum efficiency, maps services results are cached if the **Use server processing** option is enabled. What is more, separate cache keys are used for the live site mode and for other modes.

Click **OK**.

4. Sign out and view the page. It will look like this:

You can see one baloon on the map - this is the location marker. If you mouse-over the balloon, you will see the office name. If you click the balloon, you will see the detailed view:



8.28.5 Example: Document-related maps

This example will show you how to display a list of offices and their location on a map. The **Google maps** web part will be used in the example. However, the procedure is identical for Bing maps web part, too.

The Corporate Site sample website will be used and you will create a page for offices directly under the website root. However, if you don't have this site installed, you can use any other available site.

How to do it in general

To display content on a map, you need to have this content stored in the content tree as standard documents. These documents can be of any document type as long as the document type contains **two fields for the geographical coordinates** - one for **longitude** and one for **latitude** - or, alternatively, a human-readable **location field**, e.g. an address.

Then you need to configure properties of the map web part to display documents stored within the desired location, and specify which fields of the document type contain the coordinates or human-readable location. This ensures that the location markers are displayed.

Step 1 - Creating a new page with offices

1. Sign in as an administrator to **CMS Desk -> Content** and click the root of the content tree.

2. Click **New** and choose to create a new **Page (menu item)**. Enter *Offices* for **Page name**, choose the **Create a blank page with layout** option and select the **Two columns** layout. Click **Save**.

3. First you need to add a list of offices. Switch to the **Design** tab and add the **Listings and viewers/Documents/Datalist** web part to the **zoneLeft** web part zone. Set the following properties:

- **Path**: /{0}/%
- **Document types**: CMS.Office
- **Transformation**: CMS.Office.Simple
- **Selected item transformation**: CMS.Office.Default
- **Repeat columns**: 1
- **Repeat direction**: Vertical
- **Repeat layout**: Table
- **Content before**:
- **Content after**:

Click **OK**.

Step 2 - Geo-coding your information

4. Now you will create two documents of the **Office** type under the document created in Step 1. Click the *Offices* page and click **New**. Choose to create a new **Office** and enter the following values:

- **Office name**: New York Office
- **Address line 1**: 1290 Avenue of the Americas
- **City**: New York
- **ZIP code**: 10104
- **State**: NY
- **Country**: U.S.A.
- **Phone**: 123456789
- **E-mail**: ny@example.com
- **Latitude**: 40.76
- **Longitude**: -73.98
- **Office photo** - please choose an image from your disk.

- **Icon URL** - if you don't want to use the default location marker icon for this particular document on the map, you can **alternatively** set URL of your custom location marker icon. Click the **Select** button to open the **Insert link** dialog and set your custom location marker icon URL. For more details on how to insert a URL, please refer to the WYSIWYG editor -> Insert link chapter.

> **Please note**
>
> The **Office** document type already contains the **Latitude** and **Longitude** fields. These are the fields that you will later specify in the map web part properties as the **source for the geographical position** of the location markers. If you use a custom document type, you will need to define these fields manually. They must be of the **decimal number** type. You can give them any names you want and only need to specify them in the web part properties.

Values are valid in the following intervals:

- **Latitude**: from -90 to 90
- **Longitude**: from -180 to 180

Click **Save**.

5. Create another office:

- **Office name**: San Francisco Office
- **Address line 1**: 835 Market Street
- **City**: San Francisco
- **ZIP code**: 94103
- **State**: CA
- **Country**: U.S.A.
- **Phone**: 123456789
- **E-mail**: sf@example.com
- **Latitude**: 37.78
- **Longitude**: -122.41
- **Office photo** - please choose an image from your disk.

- **Icon URL** - if you don't want to use the default location marker icon for this particular document on the map, you can **alternatively** set URL of your custom location marker icon. Click the **Select** button to open the **Insert link** dialog and set your custom location marker icon URL. For more details on how to insert a URL, please refer to the WYSIWYG editor -> Insert link chapter.

Click **Save**.

Step 3 - Displaying the content on the map

6. If you view the *Offices* page now, it displays only a list of offices. Switch to the **Design** tab and add the **Maps/Documents/Google maps** web part to the **zoneRight** web part zone.

First, you need to configure which documents should be displayed on the map. Set the following properties:

- **Path:** /{0}/%
- **Document types:** CMS.Office

It ensures that all offices in the current site section will be shown. Now you need to specify the transformation used for the text displayed in the balloon:

- **Transformation:** CMS.Office.Preview

Next, you will set the values that specify how the map is displayed:

- **Default latitude:** 39.27 (Latitude of the map center when the large view map is displayed on page load.)
- **Default longitude:** -98.20 (Longitude of the map center when the large view map is displayed on

page load.)
- **Latitude field:** OfficeLatitude (The field of the document containing the latitude position.)
- **Longitude field:** OfficeLongitude (The field of the document containing the longitude position.)
- **Large view scale:** 3 (The zoom used on page load.)
- **Detailed view scale:** 10 (The zoom used if a location marker is clicked.)
- **Width:** 500 (In pixels.)
- **Height:** 400 (In pixels.)
- **Tooltip field:** OfficeName (The field used for the heading of tooltips.)
- **Icon field** - contains the code name of the source field whose content will be used as URL for your custom location marker icon; i.e. *OfficeIconURL* for the purpose of this example. If not set for a particular document or left empty, the default location marker icon will be used.

Alternatively, you can leave the **Default latitude** and **Default longitude** fields for the large view map center undefined or set to zero and use the **Default location or address** field instead to enter the location value in a human-readable form, e.g. an address. Similarly, you can do this for the **Latitude field** and **Longitude field** fields related to the location markers using the alternative **Location field** field.

Important!

Using geolocation values in a human-readable form involves some processing overhead and can even lead to inaccurate results. To ensure maximum performance and accuracy, it is therefore recommended to use the latitude and longitude coordinates when defining a geographical location.

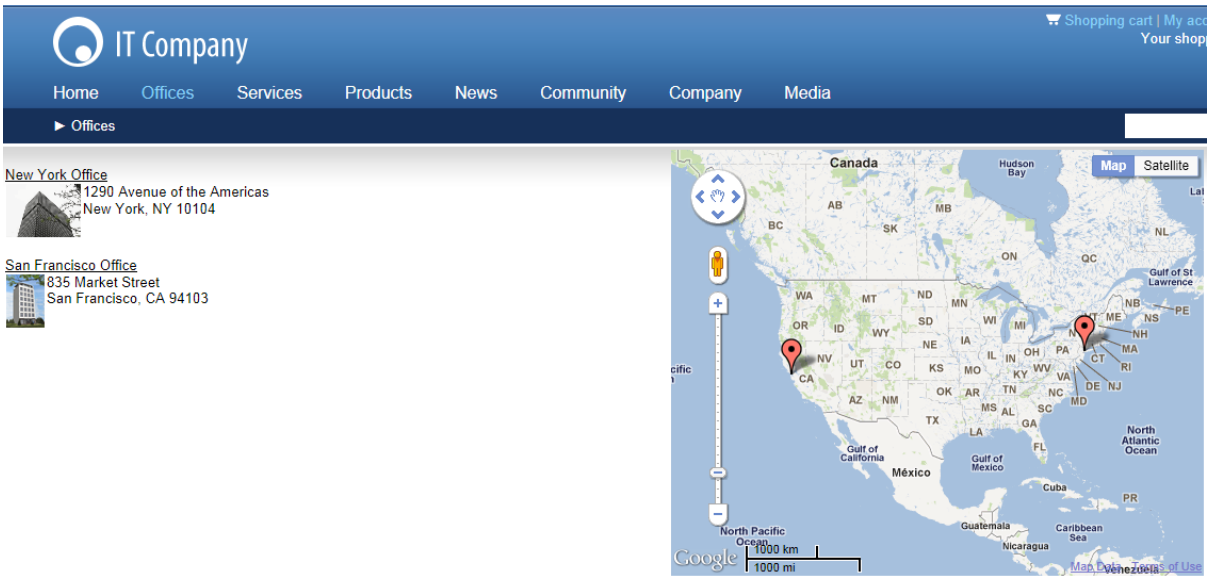
Please note

You can enable geolocation translations, i.e. addresses to coordinates, on the server side by checking the **Use server processing** checkbox. If you decide to do so, please check that the maps services query limits (IP-based) are sufficient for your needs.

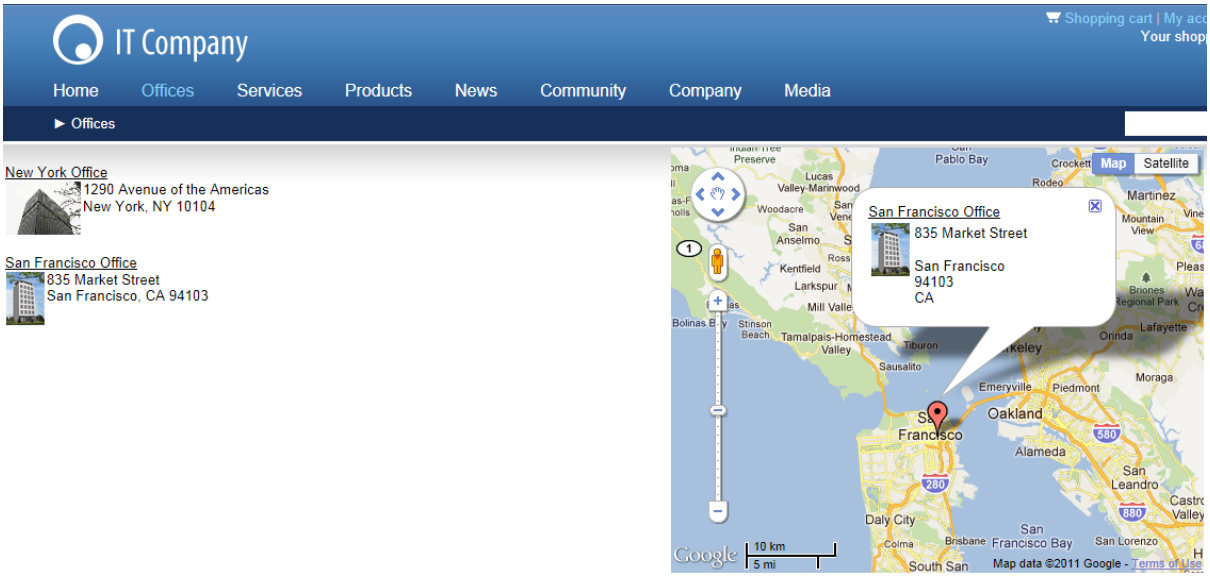
To ensure maximum efficiency, maps services results are cached if the **Use server processing** option is enabled. What is more, separate cache keys are used for the live site mode and for other modes.

Click **OK**.

7. Sign out and view the page. It will look like this:



You can see two balloons on the map - these are the location markers. If you mouse-over a balloon, you will see the office name. If you click a balloon, you will see the office detailed view:



8.28.6 Example: Maps with a data source

This example will show you how to display a list of offices and their location on a map using a data source. The **Basic Google maps** web part in combination with the **Documents data source** web part will be used in the example. However, the procedure is identical for Basic Bing maps web part, too.

The Corporate Site sample website will be used and you will create a page for offices directly under the website root. However, if you don't have this site installed, you can use any other available site.

How to do it in general

To display content on a map, you need to have a data source web part configured that will provide this content for the map web part. For the purpose of this example, documents content will be used. It will be stored in the content tree as standard documents. These documents can be of any document type as long as the document type contains **two fields for the geographical coordinates** - one for **longitude** and one for **latitude** - or, alternatively, a human-readable **location field**, e.g. an address.

When you have the documents created and stored within the desired location, you will need to configure properties of the data source and map web parts to display them on a map. You will also need to specify which fields of the document type contain the coordinates or human-readable location. This ensures that location markers are displayed.

Step 1 - Creating a new page with offices

1. Sign in as an administrator to **CMS Desk -> Content** and click the root of the content tree.

2. Click **New** and choose to create a new **Page (menu item)**. Enter *Offices* for **Page name**, choose the **Create a blank page with layout** option and select the **Two columns** layout. Click **Save**.

3. First you need to add a list of offices. Switch to the **Design** tab and add the **Listings and viewers/ Documents/Datalist** web part to the **zoneLeft** web part zone. Set the following properties:

- **Path**: /{0}/%
- **Document types**: CMS.Office
- **Transformation**: CMS.Office.Simple
- **Selected item transformation**: CMS.Office.Default
- **Repeat columns**: 1
- **Repeat direction**: Vertical
- **Repeat layout**: Table
- **Content before**:
- **Content after**:

Click **OK**.

Step 2 - Geo-coding your information

4. Now you will create two documents of the **Office** type under the document created in Step 1. Click the *Offices* page and click **New**. Choose to create a new **Office** and enter the following values:

- **Office name**: New York Office
- **Address line 1**: 1290 Avenue of the Americas
- **City**: New York
- **ZIP code**: 10104
- **State**: NY
- **Country**: U.S.A.
- **Phone**: 123456789
- **E-mail**: ny@example.com
- **Latitude**: 40.76
- **Longitude**: -73.98
- **Office photo** - please choose an image from your disk.

- **Icon URL** - if you don't want to use the default location marker icon for this particular document on

the map, you can **alternatively** set URL of your custom location marker icon. Click the **Select** button to open the **Insert link** dialog and set your custom location marker icon URL. For more details on how to insert a URL, please refer to the WYSIWYG editor -> Insert link chapter.

Please note

The **Office** document type already contains the **Latitude** and **Longitude** fields. These are the fields that you will later specify in the map web part properties as the **source for the geographical position** of the location markers. If you use a custom document type, you will need to define these fields manually. They must be of the **decimal number** type. You can give them any names you want and only need to specify them in the web part properties.

Values are valid in the following intervals:

- **Latitude**: from -90 to 90
- **Longitude**: from -180 to 180

Click **Save**.

5. Create another office:

- **Office name**: San Francisco Office
- **Address line 1**: 835 Market Street
- **City**: San Francisco
- **ZIP code**: 94103
- **State**: CA
- **Country**: U.S.A.
- **Phone**: 123456789
- **E-mail**: sf@example.com
- **Latitude**: 37.78
- **Longitude**: -122.41
- **Office photo** - please choose an image from your disk.

- **Icon URL** - if you don't want to use the default location marker icon for this particular document on the map, you can **alternatively** set URL of your custom location marker icon. Click the **Select** button to open the **Insert link** dialog and set your custom location marker icon URL. For more details on how to insert a URL, please refer to the WYSIWYG editor -> Insert link chapter.

Click **Save**.

Step 3 - Configuring the data source

6. If you view the *Offices* page now, it displays only a list of offices. Switch to the **Design** tab and add the **Data sources/Documents data source** web part to the **zoneRight** web part zone.

You only need to configure which documents should be used by this web part to be displayed on the map. Set the following properties:

- **Path:** /{0}/%
- **Document types:** CMS.Office

It ensures that all offices in the current site section will be shown.

Click **OK**.

Step 4 - Displaying the content on the map

7. On the **Design** tab, add the **Maps/Basic/Basic Google maps** web part to the **zoneRight** web part zone.

First, you will need to configure which data source should be used. Set the following properties:

- **Data source name:** DocumentsDataSource

Now you need to specify the transformation used for the text displayed in the balloon:

- **Transformation:** CMS.Office.Preview

Finally, you will set the values that specify how the map is displayed:

- **Default latitude:** 39.27 (Latitude of the map center when the large view map is displayed on page load.)
- **Default longitude:** -98.20 (Longitude of the map center when the large view map is displayed on page load.)
- **Latitude field:** OfficeLatitude (The field of the document containing the latitude position.)
- **Longitude field:** OfficeLongitude (The field of the document containing the longitude position.)
- **Large view scale:** 3 (The zoom used on page load.)
- **Detailed view scale:** 10 (The zoom used if a location marker is clicked.)
- **Width:** 500 (In pixels.)
- **Height:** 400 (In pixels.)
- **Tooltip field:** OfficeName (The field used for the heading of tooltips.)
- **Icon field** - contains the code name of the source field whose content will be used as URL for your custom location marker icon; i.e. *OfficeIconURL* for the purpose of this example. If not set for a particular document or left empty, the default location marker icon will be used.

Alternatively, you can leave the **Default latitude** and **Default longitude** fields for the large view map center undefined or set to zero and use the **Default location or address** field instead to enter the location value in a human-readable form, e.g. an address. Similarly, you can do this for the **Latitude field** and **Longitude field** fields related to the location markers using the alternative **Location field** field.

> **Important!**
>
> Using geolocation values in a human-readable form involves some processing overhead and can even lead to inaccurate results. To ensure maximum performance and accuracy, it is therefore recommended to use the latitude and longitude coordinates when defining a geographical location.

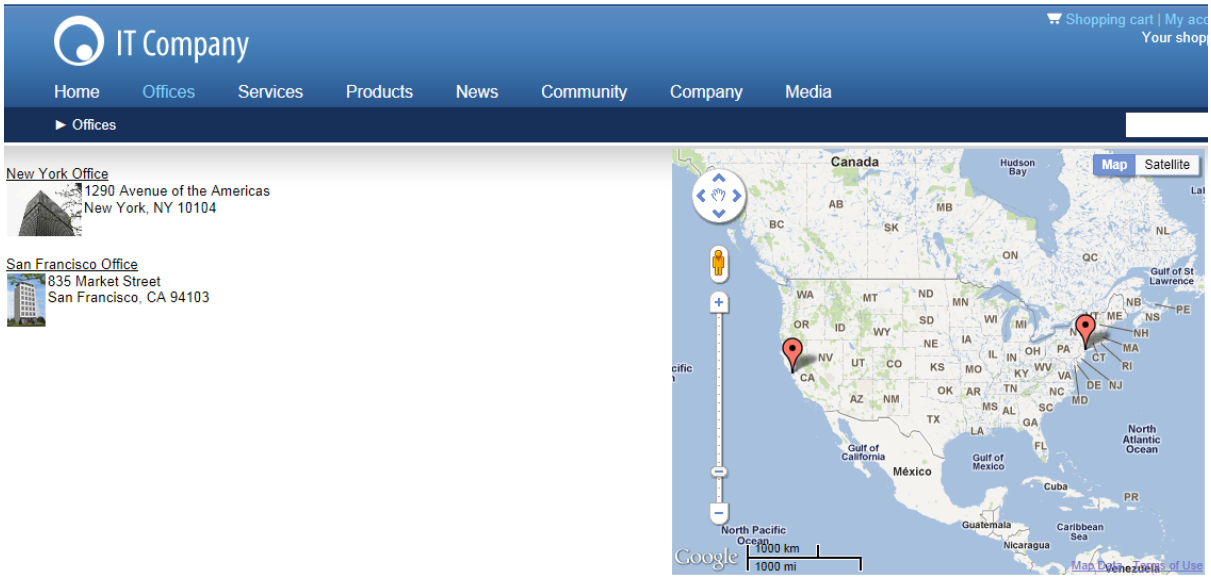
Please note

You can enable geolocation translations, i.e. addresses to coordinates, on the server side by checking the **Use server processing** checkbox. If you decide to do so, please check that the maps services query limits (IP-based) are sufficient for your needs.

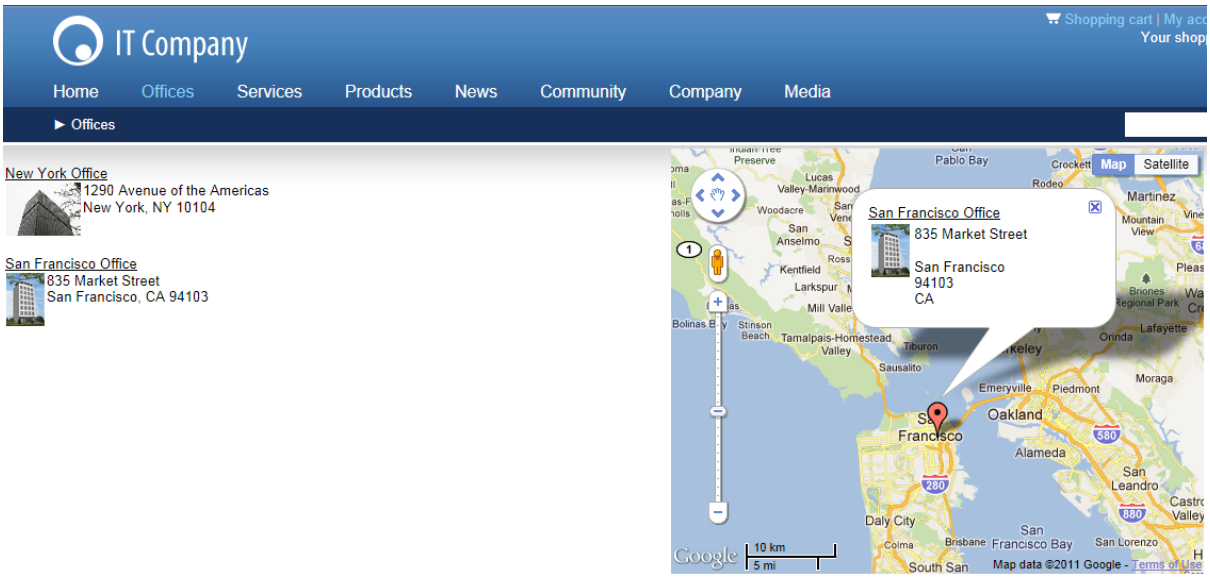
To ensure maximum efficiency, maps services results are cached if the **Use server processing** option is enabled. What is more, separate cache keys are used for the live site mode and for other modes.

Click **OK**.

8. Sign out and view the page. It will look like this:



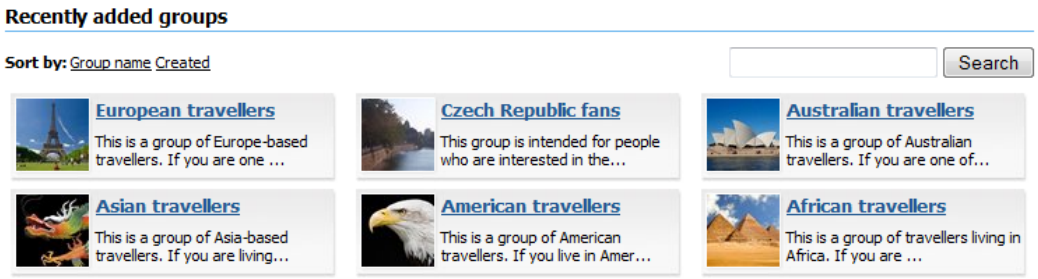
You can see two baloons on the map - these are the location markers. If you mouse-over a balloon, you will see the office name. If you click a balloon, you will see the office detailed view:



8.29 Groups

8.29.1 Overview

The groups module allows site members who share an interest in a certain topic or field to access related information and share their own experiences on a subsection of your website. Site users can create new groups or join already existing ones. Groups may contain their own documents section, forums, message boards, media libraries, polls and projects and have the option of defining group roles. Groups can also be useful for managing access control.



Site administrators can manage groups of a given site through the CMS Desk interface. Learn more in the Groups management topic.

Both site administrators and users who are in authorized roles can edit the content and various settings of groups. Learn more about this in the Editing a group topic. Further settings available only to global administrators are described in the Settings topic.

To allow users to interact with the groups module, you have to place some group web parts, which can be found in the **Community** web part category, somewhere on your site. An example of this that describes how to enable users to create groups can be found in the Enabling users to create groups topic. A full list of group web parts and their function and properties can be found in the Kentico CMS Web Parts reference.

The Groups internals and API sub-chapter provides information about the database tables and classes used by the module and examples of how groups can be managed using the API.

Kentico CMS Community Site Guide contains some additional group examples and tutorials:

- Part 1 -> Groups: Examples of the functionality and customization of groups.
- Part 2 -> Creating the Groups section: A step-by-step tutorial on how to create a sample group pages section of a website.

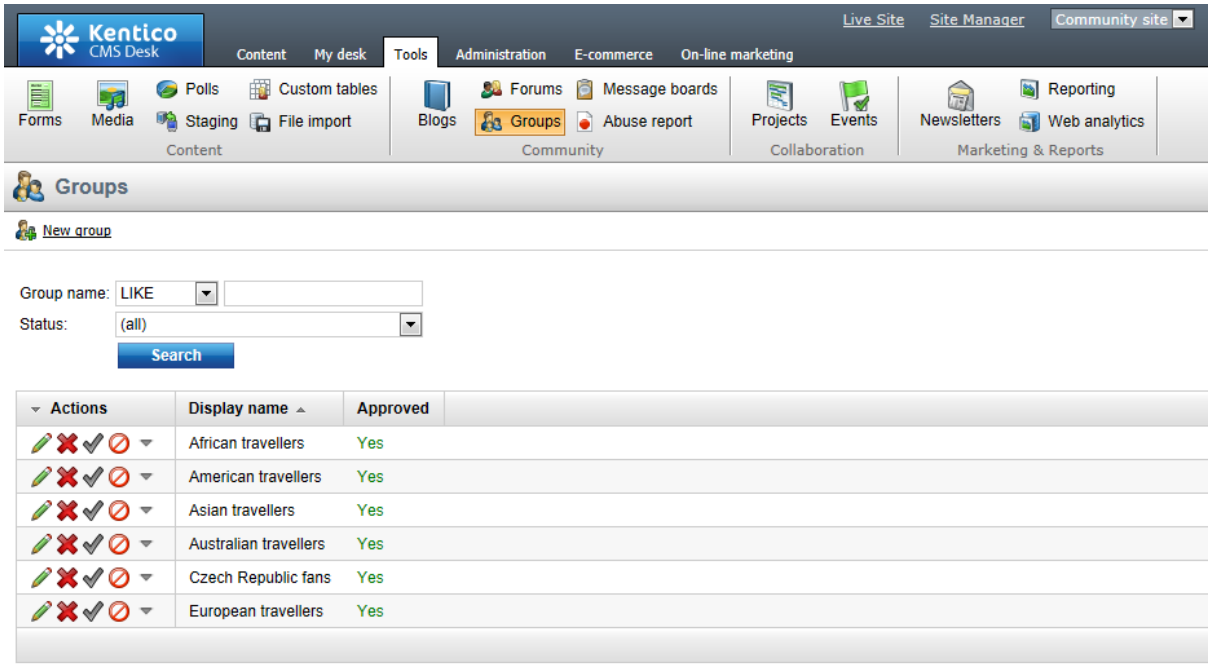
You will need to have the sample Community Site installed to follow Kentico CMS Community Site Guide.

8.29.2 Groups management

In the screenshot below, you can see the groups management interface located in **CMS Desk -> Tools -> Groups**. On this page, you can see a list of all groups on the site. You can filter displayed groups using the filter above the list. Filtration is possible by the group's **display name** and approval **Status**.

Even though groups are typically created by site users on the live site, you can create new groups in this section of the administration interface too. It can be done by clicking **New group** at the top part of the page. Groups in the list can be **Edited** () or **Deleted** (). The Editing a group topic describes editing in more detail.

If an administrator's approval is needed after a user creates a new group, the approval can be done by clicking the **Approve** () icon. By clicking the **Reject** () icon, groups can be switched back to the state they were in before they were approved. If you do this to an existing group, the group will not be displayed on the live site.



All Kentico CMS documents can be set to be owned by a group. To do this select a document from the content tree, switch to **Edit** mode and go to **Properties -> General -> Owned by group**. Then click **Change** and choose a group from the drop-down list. This property is used to identify which group a

document belongs to and is used by various group context sensitive web parts to display the correct content. It also influences the editing permissions of Group administrator widget zones. See Enabling users to create groups for more information.

8.29.3 Editing a group

There are two ways how group properties can be edited:

- On-site management using the Group profile web part; this is typically used by group administrators
- Administration interface in **CMS Desk -> Tools -> Groups**, after choosing to **Edit** () a group; this is typically used by site administrators

Both of the two approaches provide the same tabs layout, with the difference that there are some extra settings in the administration interface compared to on-site management. These will be marked as **AI only** in the descriptions below.

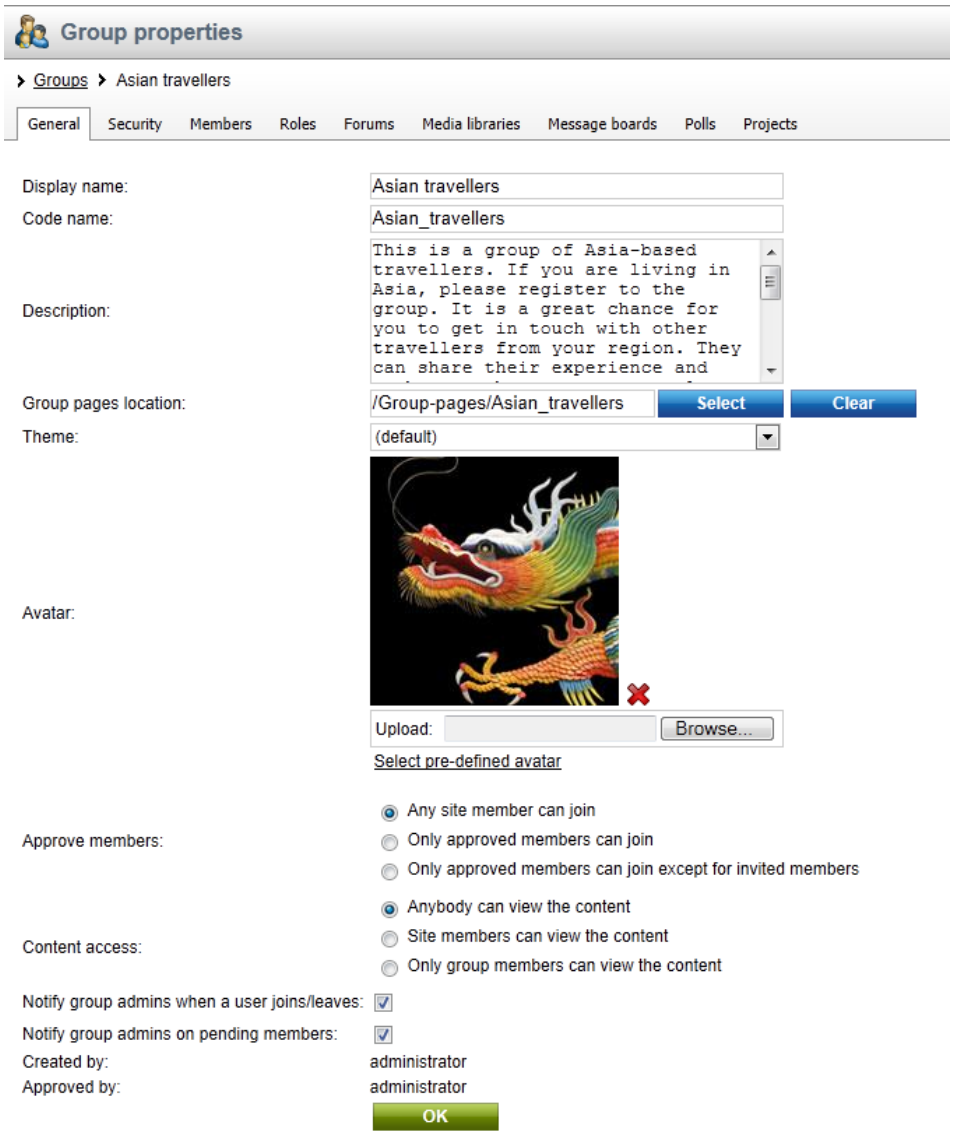
General tab

On this tab, you can set the general properties of the group:

- **Display name** - name of the group displayed on the site and in the administration interface; *AI only*
- **Code name** - name of the group used in site code; *AI only*
- **Description** - text describing the group
- **Group pages location** - node alias path of the location where group pages of the group are stored
- **Theme** - allows the selection of one of the website CSS stylesheets that will be used by the pages of the group
- **Avatar** - group avatar image

- **Approve members** - determines if users can join the group with or without a group admin's approval; the last option allows invited members to join without approval
- **Content access** - determines who can view content of the group pages
- **Notify group admins when a user joins/leaves** - if checked, group administrators will receive a notification e-mail when a user joins/leaves the group
- **Notify group admins on pending members** - if checked, group administrators will receive a notification e-mail when a user requests to join the group and admin's approval is needed

- **Created by** - displays who created the group
- **Approved by** - displays who approved the group to be created on the site



Security tab

On this tab, you can use the matrix to set permissions for group pages. The following permissions can be assigned:

- **Create pages** - users can create group pages
- **Delete pages** - users can delete group pages
- **Edit pages** - users can edit group pages

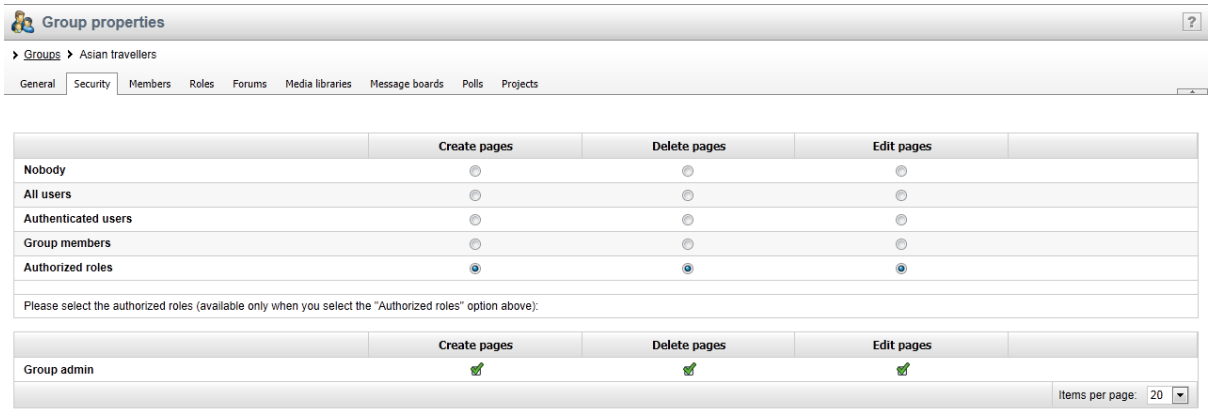
These permissions can be assigned to:

- **Nobody** - nobody can perform the action
- **All users** - all users can perform the action
- **Authenticated users** - only signed-in users can perform the action, i.e. anonymous public users cannot perform it

- **Group members** - only group members can perform the action, i.e. authenticated non-group members and anonymous users cannot perform it
- **Authorized roles** - only members of the group roles selected below can perform the action

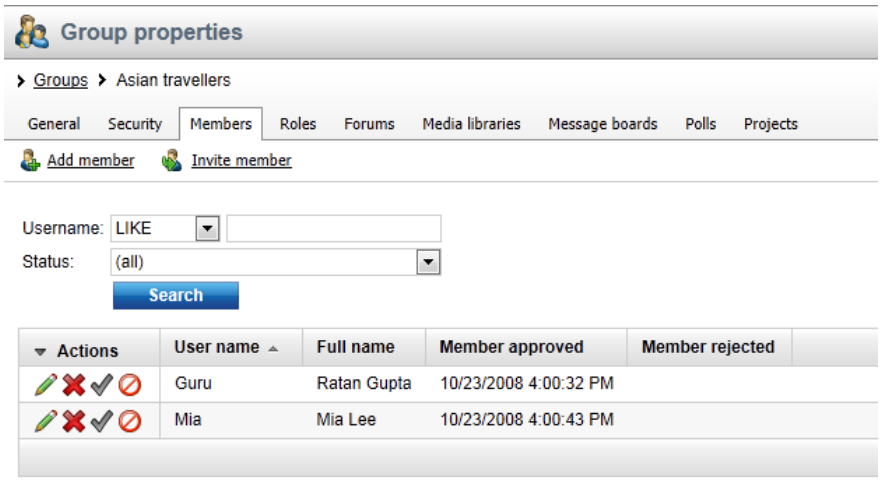
Group admin's permissions

Group administrators can perform any of these actions, even if they don't have the permissions assigned.



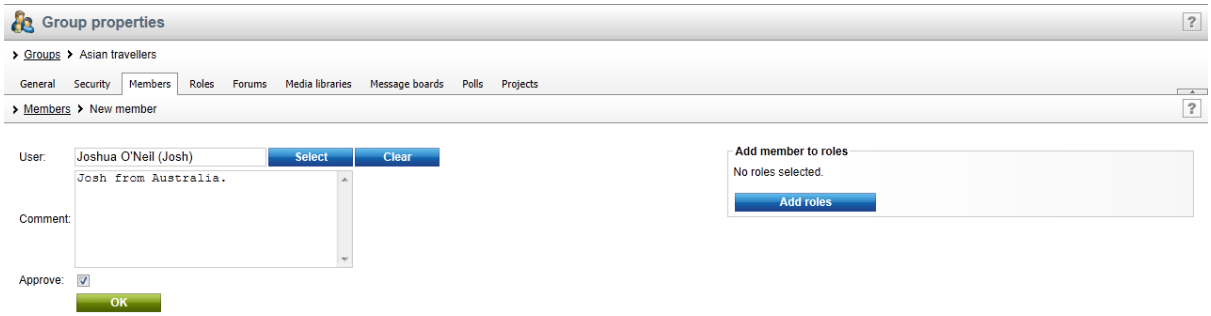
Members tab

On this tab, you can see a list of all members of the group. You can **Edit** (✎) or **Delete** (✖) users in the list. You can also **Approve** (✔) members' requests for joining the group or **Reject** (⊘) them from the group. Once rejected, the user cannot request to join the group until they are approved again.



Clicking **Add member**, you can add users to the group directly, without sending an invitation to them. This is possible only in the administration interface. On the live site, only **Invite member** is displayed. When adding a user to a group, you have the following options:

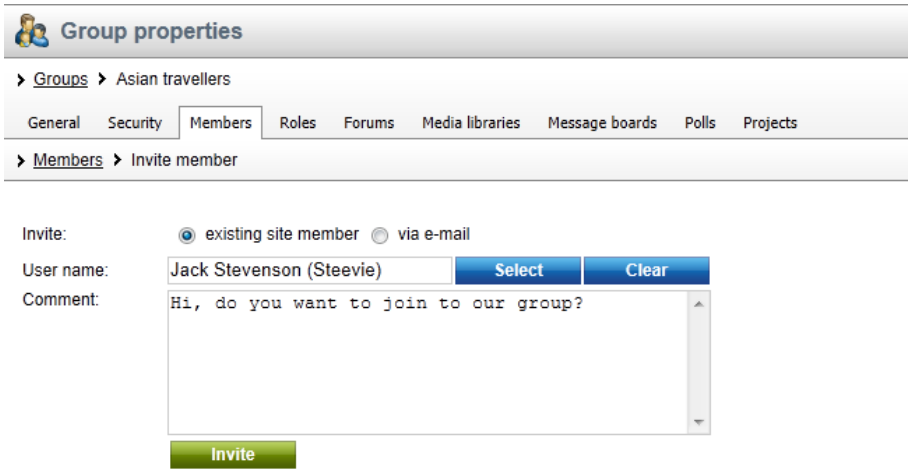
- **User** - select an existing site user who you want to add to the group
- **Comment** - text comment that you can add to the user; this comment is not sent to the user, it is only displayed in the administration interface
- **Approve** - if checked, the user will be automatically approved; if not, a user will need a group admin's approval before they become members of the group
- **Add roles** - you can use this button to assign the user to group roles; a dialog where you can select from a list of available group roles is opened



After clicking **Invite member**, the dialog displayed in the screenshot below will be displayed. There are two ways of invitation:

- **Invite existing site member** - after selecting an existing site user in the **User name** field, an invitation e-mail will be sent to the user's e-mail address. The text entered into the **Comment** field will be included in the e-mail. Users can then join the group either by clicking a link in the e-mail, or via the **My sent invitations** web part.

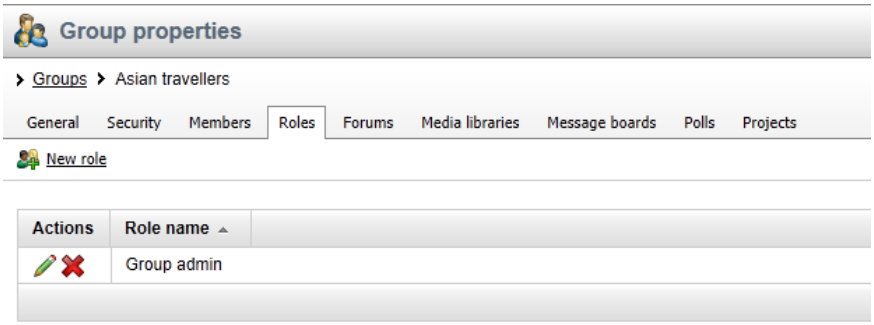
- **Invite via e-mail** - this way, you can send the invitation to any e-mail address that you enter into the **E-mail** field. In this case, user will be required to register to the site after clicking the join link in the e-mail. Text entered to the **Comment** field will be included in the e-mail.



Roles tab

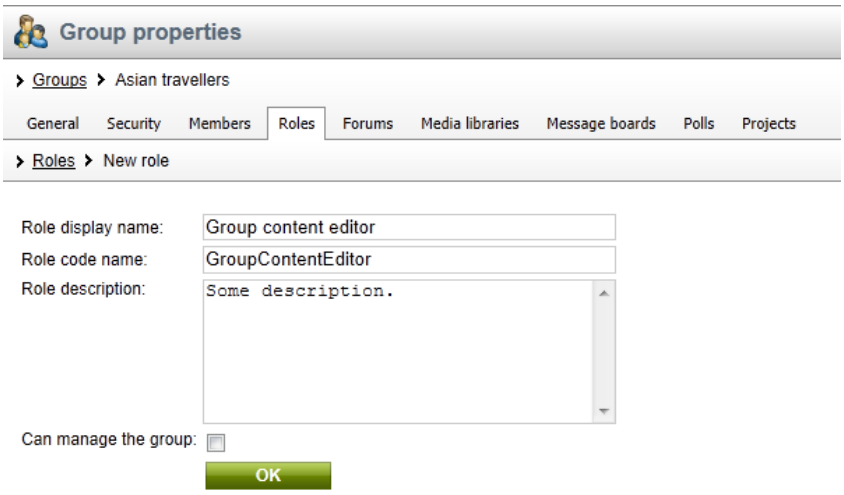
On this tab, you can see a list of roles defined for the group. These roles are applicable only in the context of the group. Don't confuse them with website roles, which can be set in **Site Manager -> Administration -> Roles**.

Roles in the list can be **Edited** () and **Deleted** ().

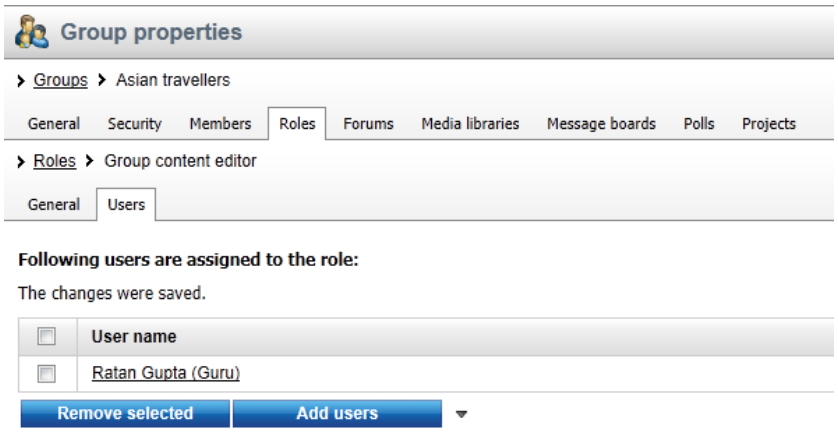


If you click  **New role** above the list, you can define a new role for the group. The following properties can be entered:

- **Role display name** - name of the role that will be used on the live site and in the administration interface
- **Role code name** - name of the role used in your code; *AI only*
- **Role description** - text description of the role
- **Can manage the group** - indicates if members of the role can manage the group by means of the **Group profile** web part



When **Editing** () a role, two tabs are offered. On the **General** tab, you can change the details entered when creating the role, as described above. On the **Users** tab, you can see a list of all members assigned to the role. These can be removed from the role by selecting the checkbox to the left of users in the list and clicking the **Remove selected** button. New users can be added to the role using the **Add users** button.



Forums tab

On this tab, you can create and manage the group's forums. As these forums are standard Kentico CMS forums set into the context of the group, please refer to the Modules -> Forums chapter of this guide for more information on their management.

Media libraries tab

On this tab, you can create and manage the group's media libraries. As these are standard Kentico CMS media libraries set into the context of the group, please refer to the Modules -> Media libraries chapter of this guide for more information on their management.

Message boards tab

On this tab, you can manage the group's message boards. As these are standard Kentico CMS message boards set into the context of the group, please refer to the Modules -> Message boards chapter of this guide for more information on their management.

Polls tab

On this tab, you can manage the group's polls. As these are standard Kentico CMS polls set into the context of the group, please refer to the Modules -> Polls chapter of this guide for more information on their management.

Projects tab

On this tab, you can manage the group's projects. As these are standard Kentico CMS projects set into the context of the group, please refer to the Modules -> Project management chapter of this guide for more information on their management.

8.29.4 Enabling users to create groups

You can enable site users to create new groups by placing the Community -> Groups -> Group registration web part on your site. You have to set the following properties of the web part:

- **Template source alias path** - alias path of the document that will be used, together with the documents stored under it, as a template for groups created by the web part. If left empty, the value of the *Site Manager -> Settings -> Community -> Group template path* field will be used.
- **Template target alias path** - alias path where the documents copied from the *Template source alias path* will be loaded when a group is created.
- **Automatically create forum** - if checked, a forum group and a General discussion forum are automatically added under the created group.
- **Automatically create media library** - if checked, a media library is automatically added under the created group.
- **Automatically create smart search indexes** - if checked, a smart search index is automatically created for the documents of the created group, as well as for the new forum if the *Automatically create forum* property is enabled.
- **Group profile URL path** - alias path of the page containing the group profile. The *{groupname}* wildcard can be used to substitute for the name of the current group.
- **Combine with default culture** - if checked, the default culture will be used when creating group pages under a culture where the source or target nodes were not found.
- **Group name label text** - text that will be displayed in the form before the field where the group name is entered.
- **Text after successful registration** - text displayed when a group is successfully created.
- **Text after successful registration with approving** - text displayed when a group is successfully created, but requires an administrator's approval to be published on the web.

- **Require approval** - if checked, the group will have to be approved by a site administrator before it is published on the site.
- **Redirect to URL** - URL where the user will be redirected after creating the group.
- **Hide form after registration** - if checked, the form will be hidden after creating the group.

Group pages templates

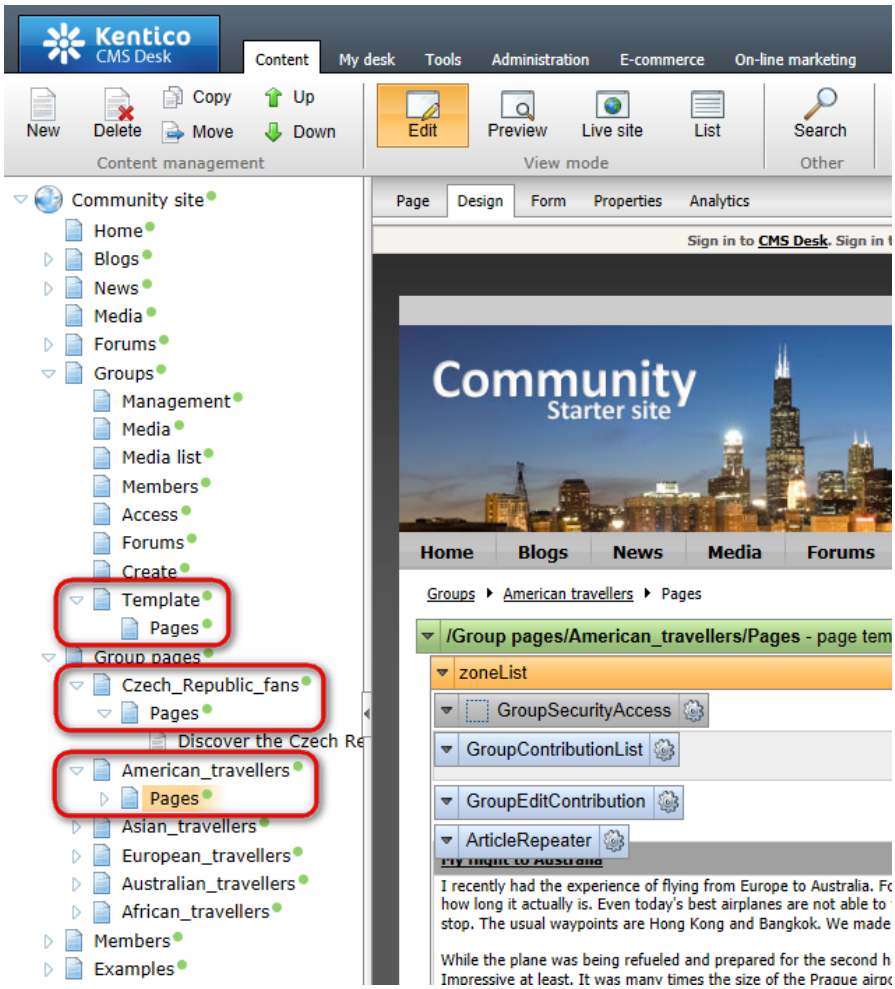
Each group has its own section on the website where its content is stored - so called group pages. When adding the **Group registration** web part to your site, you have to specify the **Template source alias path** and **Template target alias path** properties. These two properties are essential when creating the **group pages section** of each group.

The page specified by the **Template source alias path** and all its sub-pages are copied to the location specified by the **Template target alias path**.

To get a better idea of how this works, you can take a look at our sample **Community Starter site**. On the site, the **Group registration** web part is configured the following way:

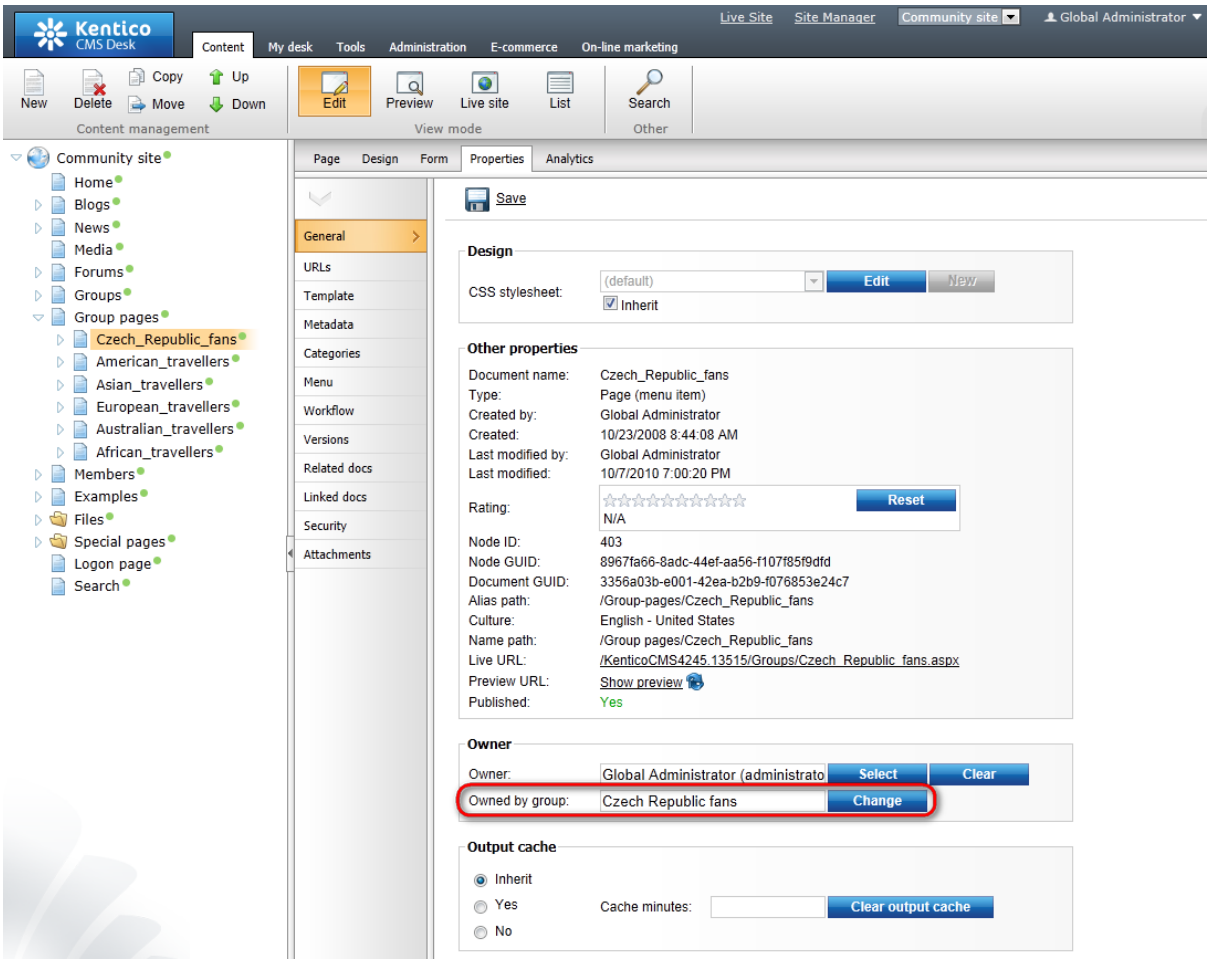
- **Template source alias path**: /Groups/Template
- **Template target alias path**: /Group-pages

As you can see in the screenshot below, there is the **/Groups/Template** page with one sub-page: **Pages**. When a new group is created, its title page is created under **/Group-pages** and the **Pages** page is created under it. As you have probably noticed, the web parts placed on the title page are identical to those placed on the **Template** page. Web parts on the **Pages** page are also identical to the source **Pages** page. Under **Pages**, all group documents will be stored.



All Kentico CMS documents can be set to be owned by a group. This can be done by signing into CMS Desk, selecting a document from the content tree, switching to **Edit** mode and going to **Properties -> General -> Owned by group** and choosing a group from the drop-down list displayed after clicking the **Change** button. This property is used to identify which group a document belongs to and is used by various group context sensitive web parts to display the correct content. It also influences the editing permissions of Group administrator widget zones.

In **Site Manager -> Settings -> Community**, you can enable or disable the **Use parent community group for new documents** option. It you have it enabled, all newly created documents will inherit the group assignment from their parent document.



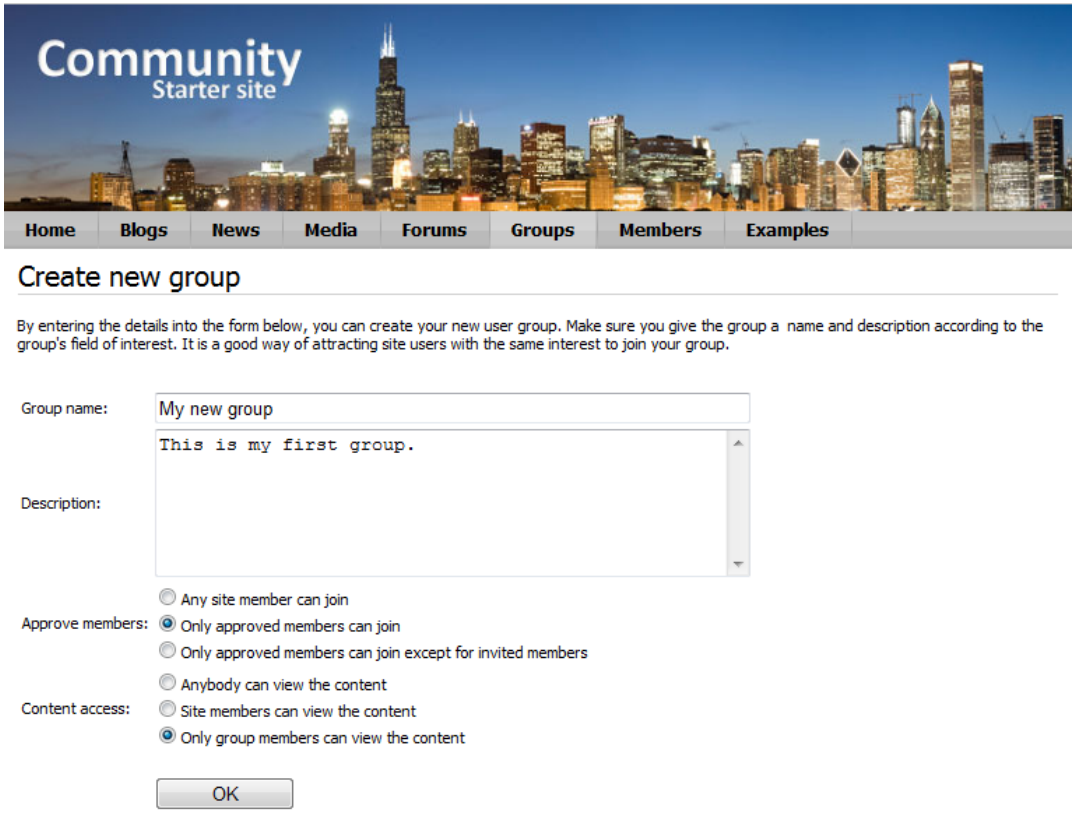
Please refer to Community Site Guide -> Part 2 -> Creating the Groups section for a step-by-step tutorial on how to create a sample group pages section of a website.

8.29.5 How site users create a new group

When a user wants to create a new group on the live site using the **Group registration** web part, they have to fill in the following details:

- **Group name** - name of the group displayed on the site and in the administration interface
- **Description** - text describing the group
- **Approve members** - determines if users can join the group with or without the group admin's approval; the last options allows invited members to join without the approval
- **Content access** - determines who can view content of the group pages

After clicking **OK**, the group will be created and group pages added to the site. In case that site administrator's approval is needed, these actions will be performed after the approval.



8.29.6 Related e-mail templates

In **Site manager -> Administration -> E-mail templates**, you can find the E-mail templates used for the automatic group-related e-mail notifications. Messages based on the following templates are sent to users when they attempt to join a group or invite a new member:

- **Groups - Member joined confirmation** - sent to users who join a group that does not require membership approval.
- **Groups - Member joined waiting for approval** - sent to users who apply to groups where approval is needed.
- **Groups - Member approved** - informs users that their group membership has been approved.
- **Groups - Member rejected** - sent to users if their attempt to join a group was rejected by the group's administrators.
- **Groups - Member invitation** - sent to users who are invited to a group by an existing member.
- **Groups - Member accepted invitation** - used to inform group members that their invitation was accepted by the recipient.

The templates below are used for notifications sent to group or community administrators:

- **Groups - Member join** - notifies group administrators when a new member joins their group.
- **Groups - Member waiting for approval** - lets administrators know that a new group member is waiting for their approval.
- **Groups - Member leave** - notifies group administrators when a member leaves their group.
- **Groups - Waiting for approval** - sent to community administrators when a new group is created (if it requires approval) .

Any of these templates can be edited as needed, so you may fully customize the content of the notifications. You can use the following context macros to include dynamic values in their text. Please note that some of the macros are only available in certain templates, e.g. invitation values and objects can only be accessed in templates related to invitation notifications.

- **{% AcceptionURL %}** - resolves into the URL of a page where users can accept an invitation to join a group. See *Invitation acceptation path* in Settings for more details.
- **{% InvitedBy %}** - returns the name of the user who sent the invitation.

You can also access the following objects and their properties (e.g. *{% MemberUser.FullName %}*):

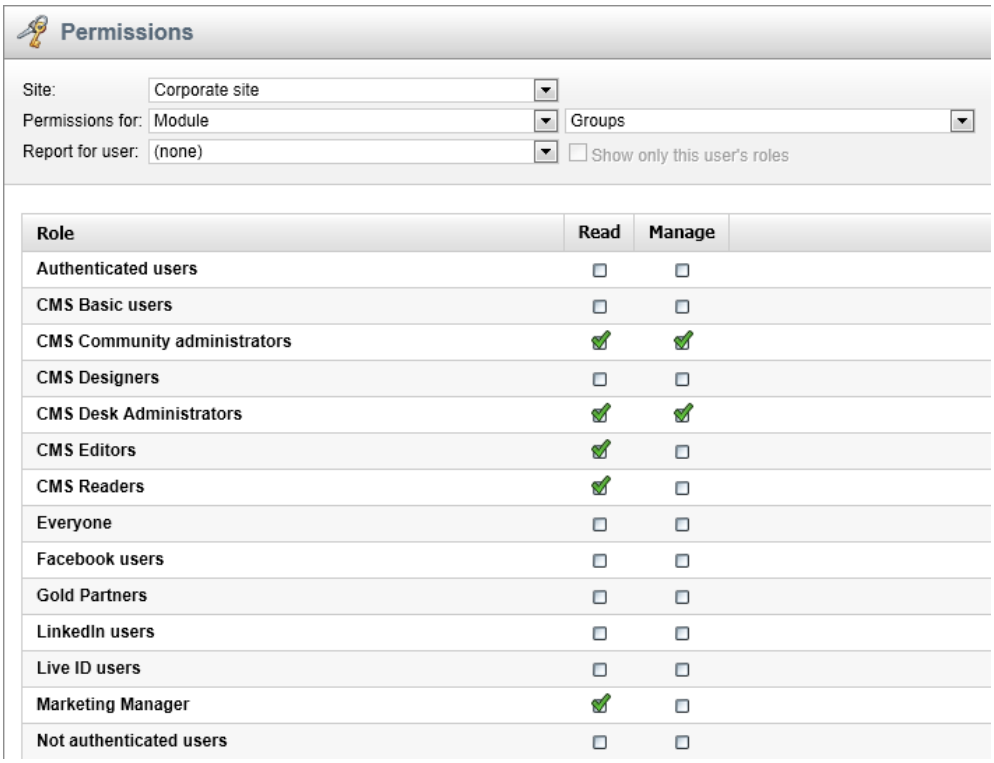
- **{% Group %}** - *GroupInfo* object of the group related to the notification.
- **{% MemberUser %}** - *UserInfo* object of the user who is attempting to join the group.
- **{% Invitation %}** - *InvitationInfo* object that can be used to access the properties of the group invitation.
- **{% Sender %}** - *UserInfo* object representing the user who accepted the invitation (usable in the *Member accepted invitation* template).
- **{% Recipient %}** - *UserInfo* object of the user who originally sent the invitation (usable in the *Member accepted invitation* template)..
- **{% GroupMember %}** - *GroupMemberInfo* object representing the new member who accepted the invitation to join the group.

In addition to the special ones listed above, you can also use all other standard macro expressions in the templates. See the Development -> Macro expressions chapter of this guide for more information about macro expressions in Kentico CMS.

8.29.7 Security

Permissions of the Groups module can be set in **Site Manager -> Administration -> Permissions**. The following permissions can be assigned to members of the particular roles:

- **Manage** - allows managing of groups via the administration interface
- **Read** - allows viewing group settings, but does not allow any changes to be made to them



Group pages permissions

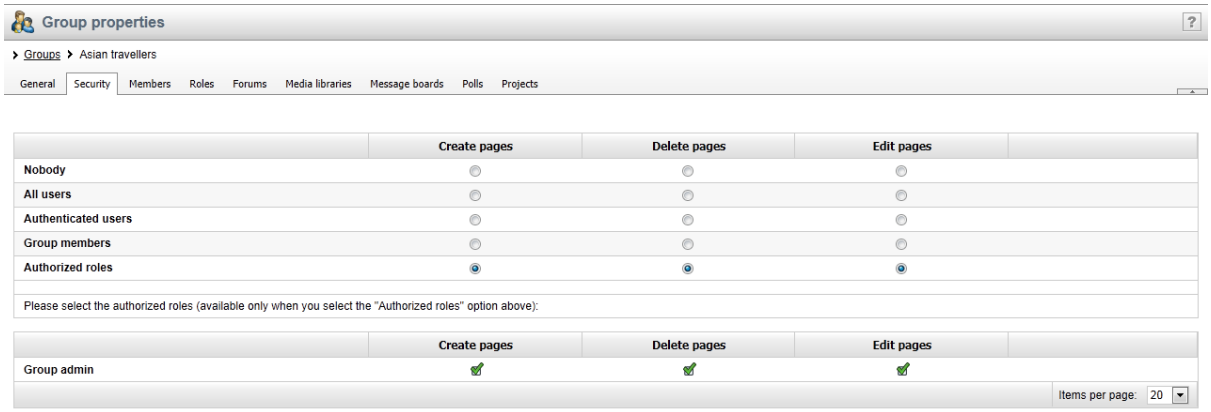
- **Create pages** - users can create group pages
- **Delete pages** - users can delete group pages
- **Edit pages** - users can edit group pages

These permissions can be assigned to:

- **Nobody** - nobody can perform the action
- **All users** - all users can perform the action
- **Authenticated users** - only signed-in users can perform the action, i.e. anonymous public users can't perform it
- **Group members** - only group members can perform the action, i.e. authenticated non-group members and anonymous users can't perform it
- **Authorized roles** - only members of the group roles selected below can perform the action

Group admin's permissions

Group administrators can perform any of these actions, even if they haven't the permissions assigned.



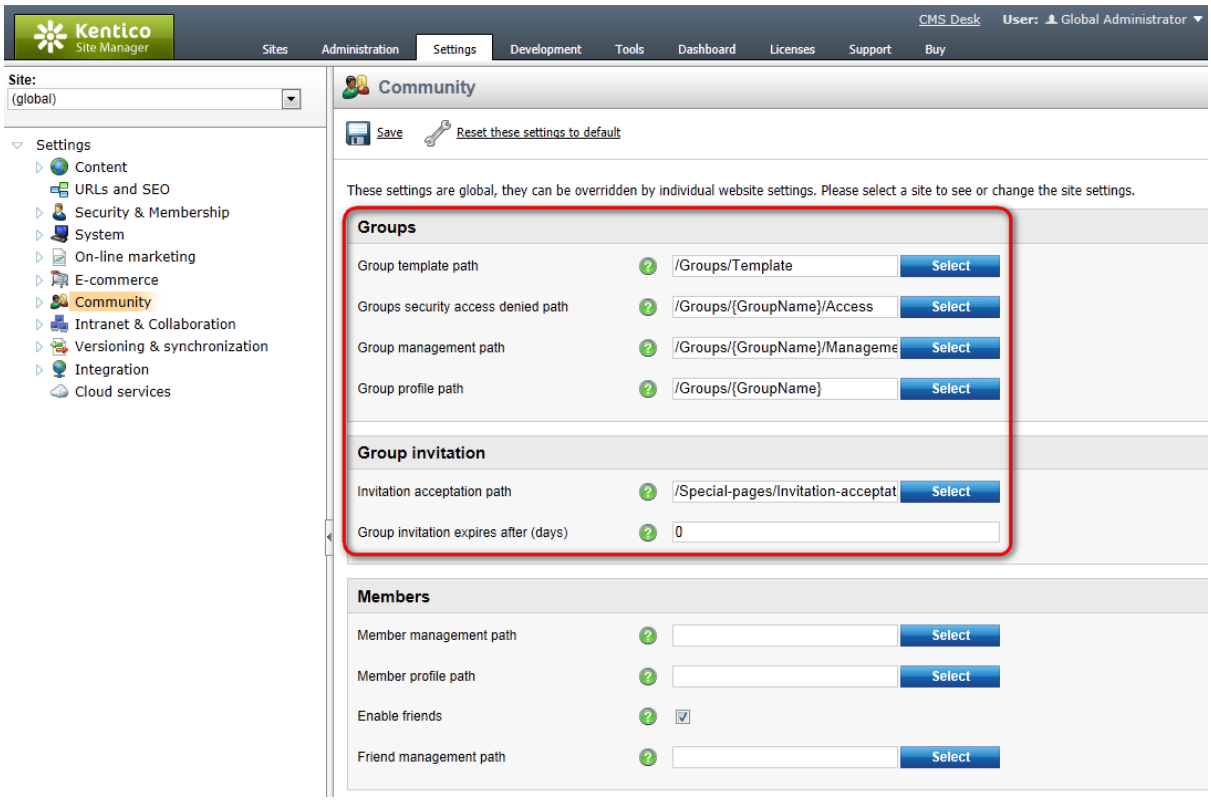
8.29.8 Settings

Settings of the groups module are located in **Site Manager -> Settings -> Community**. The following settings can be adjusted:

- **Group template path** - alias path of the document that will be used, together with the documents stored under it, as a template for newly created groups; e.g. */Groups/Template*
- **Groups security access path** - alias path of a document to which users will be redirected when they try to access pages of a group to which they don't have permissions; this page should contain the Group security message web part; e.g. */Groups/{GroupName}/Access*
- **Group management path** - alias path of the group management page, containing the Group profile web part; e.g. */Groups/{GroupName}/Management*
- **Group profile path** - alias path of the group profile page; e.g. */Groups/{GroupName}*

- **Invitation acceptation path** - alias path of the document containing the Group invitation web part; this is a special web part handling requests for joining a group when a user clicks the joining link in a group invitation e-mail; e.g. */Special-pages/Invitation-acceptation*.
- **Group invitation expires after (days)** - when some user receives a group invitation e-mail, the link for joining the group included in the e-mail will be active for the number of days entered here. After the specified duration, the link will no longer be functional; when 0 is entered, the link will be functional permanently.

- **Use parent community group for new documents** - indicates if new documents should inherit the value of the *Owned by group* property from their parent document.



8.29.9 Groups internals and API

8.29.9.1 Overview

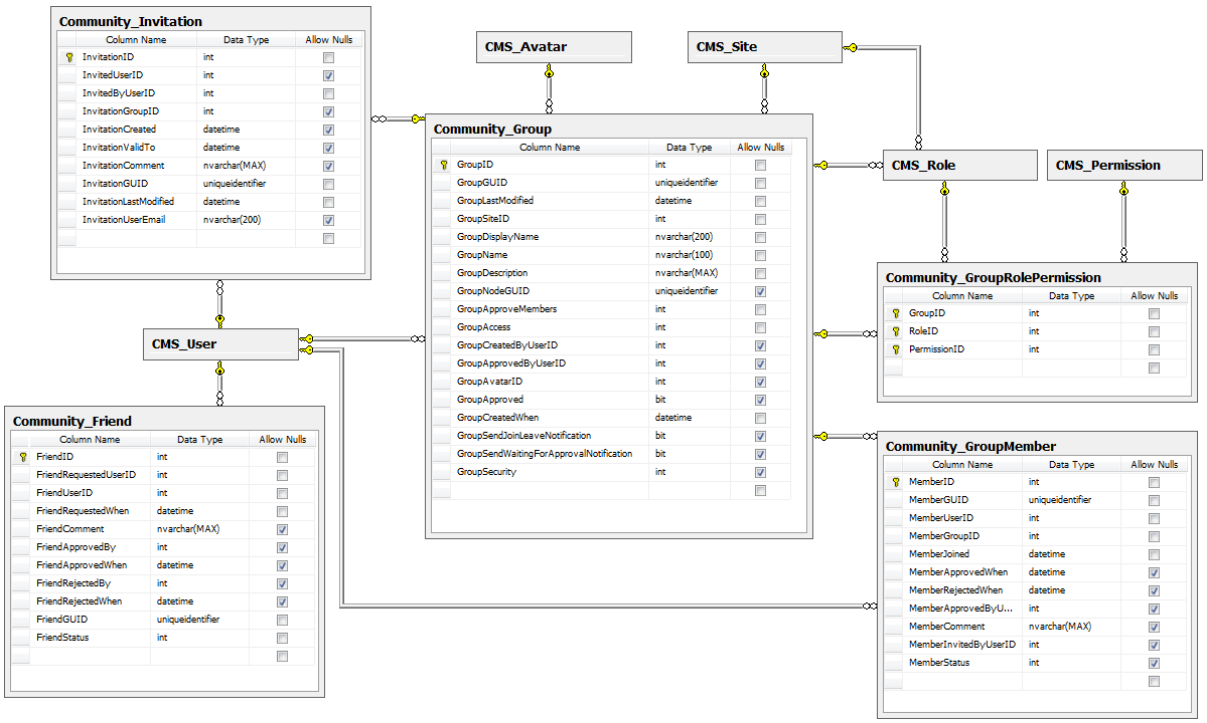
In this chapter, you will learn which database tables and API classes are used in the Groups module. You will also see the most common API examples.

Further API-related information and examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all members of the module's classes, please refer to Kentico CMS API Reference.

8.29.9.2 Database tables

The following database tables are used to store information about groups:

- **Community_Group** - contains records representing groups and their settings.
- **Community_GroupMember** - stores relationships between groups and users and other group membership data.
- **Community_GroupRolePermission** - stores relationships between groups and roles as well as a permission type. Each entry in this table indicates that users belonging to the given role have the specified permission for a group.
- **Community_Invitation** - stores invitations for group membership.



Group role storage

Group roles are stored along with standard roles in the **CMS_Role** table. Records representing group roles have a value in their **RoleGroupID** field that is equal to the ID of the group that they belong under, while standard roles have it set to *NULL*.

8.29.9.3 API classes

If you are not familiar with database table data management through Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

The classes of the Groups module can be found in the **CMS.Community** namespace.

Community_Group table API:

- **GroupInfo** - represents one group object.
- **GroupInfoProvider** - provides management functionality for groups.

Community_GroupMember table API:

- **GroupMemberInfo** - represents one group member.

- **GroupMemberInfoProvider** - provides management functionality for group members.

Community_GroupRolePermission table API:

- **GroupRolePermissionInfo** - represents a relationship between a group, role and permission.
- **GroupRolePermissionInfoProvider** - provides management functionality for group-role relationships.

Community_Invitation table API:

- **InvitationInfo** - represents an invitation to become a member of a group.
- **InvitationInfoProvider** - provides management functionality for group invitations.

Other classes:

- **CommunityContext** - this class can be used to provide group-related data of the current request.

8.29.9.4 API examples

8.29.9.4.1 Overview

These topics show examples of how the API of the Groups module can be used:

- Managing groups
- Managing group members

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Community\Groups\Default.aspx.cs***.

The group API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.GlobalHelper;
using CMS.CMSHelper;
using CMS.SiteProvider;
using CMS.Community;
```

8.29.9.4.2 Managing groups

The following example creates a group.

```
private void CreateGroup()
{
    // Create new group object
    GroupInfo newGroup = new GroupInfo();

    // Set the properties
    newGroup.GroupDisplayName = "My new group";
    newGroup.GroupName = "MyNewGroup";
    newGroup.GroupSiteID = CMSContext.CurrentSiteID;
    newGroup.GroupDescription = "";
    newGroup.GroupApproveMembers = GroupApproveMembersEnum.AnyoneCanJoin;
    newGroup.GroupAccess = CMS.SiteProvider.SecurityAccessEnum.AllUsers;
    newGroup.GroupApproved = true;
    newGroup.GroupApprovedByUserID = CurrentUser.UserID;
    newGroup.GroupCreatedByUserID = CurrentUser.UserID;
    newGroup.AllowCreate = SecurityAccessEnum.GroupMembers;
    newGroup.AllowDelete = SecurityAccessEnum.GroupMembers;
    newGroup.AllowModify = SecurityAccessEnum.GroupMembers;
    newGroup.GroupNodeGUID = Guid.Empty;

    // Save the group
    GroupInfoProvider.SetGroupInfo(newGroup);
}
```

The following example gets and updates a group.

```
private bool GetAndUpdateGroup()
{
    // Get the group
    GroupInfo updateGroup = GroupInfoProvider.GetGroupInfo("MyNewGroup",
CMSContext.CurrentSiteName);
    if (updateGroup != null)
    {
        // Update the properties
        updateGroup.GroupDisplayName = updateGroup.GroupDisplayName.ToLower();

        // Save the changes
        GroupInfoProvider.SetGroupInfo(updateGroup);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates groups.

```
private bool GetAndBulkUpdateGroups()
{
    // Prepare the parameters
    string where = "GroupName LIKE N'MyNewGroup%'";
```

```
    // Get the data
    DataSet groups = GroupInfoProvider.GetGroups(where, null);
    if (!DataHelper.DataSourceIsEmpty(groups))
    {
        // Loop through the individual items
        foreach (DataRow groupDr in groups.Tables[0].Rows)
        {
            // Create object from DataRow
            GroupInfo modifyGroup = new GroupInfo(groupDr);

            // Update the properties
            modifyGroup.GroupDisplayName = modifyGroup.GroupDisplayName.ToUpper();

            // Save the changes
            GroupInfoProvider.SetGroupInfo(modifyGroup);
        }

        return true;
    }

    return false;
}
```

The following example deletes a group.

```
private bool DeleteGroup()
{
    // Get the group
    GroupInfo deleteGroup = GroupInfoProvider.GetGroupInfo("MyNewGroup",
CMSContext.CurrentSiteName);

    // Delete the group
    GroupInfoProvider.DeleteGroupInfo(deleteGroup);

    return (deleteGroup != null);
}
```

8.29.9.4.3 Managing group members

The following example adds a user as a member of a group.

```
private bool CreateGroupMember()
{
    // Get the group
    GroupInfo group = GroupInfoProvider.GetGroupInfo("MyNewGroup",
CMSContext.CurrentSiteName);

    if (group != null)
    {
```

```
        // Get the user
        UserInfo user = UserInfoProvider.GetUserInfo
(CMSContext.CurrentUser.UserName);

        if (user != null)
        {
            // Create new group member object
            GroupMemberInfo newMember = new GroupMemberInfo();

            //Set the properties
            newMember.MemberGroupID = group.GroupID;
            newMember.MemberApprovedByUserID = CurrentUser.UserID;
            newMember.MemberApprovedWhen = DateTime.Now;
            newMember.MemberInvitedByUserID = CurrentUser.UserID;
            newMember.MemberUserID = user.UserID;
            newMember.MemberJoined = DateTime.Now;
            newMember.MemberComment = "New member added by API example.";

            // Save the member
            GroupMemberInfoProvider.SetGroupMemberInfo(newMember);
        }

        return true;
    }

    return false;
}
```

The following example gets and updates a group member.

```
private bool GetAndUpdateGroupMember()
{
    // Get the group
    GroupInfo group = GroupInfoProvider.GetGroupInfo("MyNewGroup",
CMSContext.CurrentSiteName);

    if (group != null)
    {
        // Get the user
        UserInfo user = UserInfoProvider.GetUserInfo
(CMSContext.CurrentUser.UserName);

        if (user != null)
        {
            // Get the group member
            GroupMemberInfo updateMember =
GroupMemberInfoProvider.GetGroupMemberInfo(user.UserID, group.GroupID);
            if (updateMember != null)
            {
                // Update the properties
                updateMember.MemberComment = updateMember.MemberComment.ToLower();

                // Save the changes
```

```
                GroupMemberInfoProvider.SetGroupMemberInfo(updateMember);

                return true;
            }
        }
    }

    return false;
}
```

The following example gets and bulk updates group members.

```
private bool GetAndBulkUpdateGroupMembers()
{
    // Prepare the parameters
    string where = "MemberUserID =" + CMSContext.CurrentUser.UserID;

    // Get the data
    DataSet members = GroupMemberInfoProvider.GetGroupMembers(where, null);
    if (!DataHelper.DataSourceIsEmpty(members))
    {
        // Loop through the individual items
        foreach (DataRow memberDr in members.Tables[0].Rows)
        {
            // Create object from DataRow
            GroupMemberInfo modifyMember = new GroupMemberInfo(memberDr);

            // Update the properties
            modifyMember.MemberComment = modifyMember.MemberComment.ToUpper();

            // Save the changes
            GroupMemberInfoProvider.SetGroupMemberInfo(modifyMember);
        }

        return true;
    }

    return false;
}
```

The following example removes a member from a group.

```
private bool DeleteGroupMember()
{
    // Get the group
    GroupInfo group = GroupInfoProvider.GetGroupInfo("MyNewGroup",
CMSContext.CurrentSiteName);

    if (group != null)
    {
        // Get the user
```

```
        UserInfo user = UserInfoProvider.GetUserInfo
(CMSContext.CurrentUser.UserName);

        if (user != null)
        {
            // Get the group member
            GroupMemberInfo deleteMember =
GroupMemberInfoProvider.GetGroupMemberInfo(user.UserID, group.GroupID);
            if (deleteMember != null)
            {
                // Save the changes
                GroupMemberInfoProvider.DeleteGroupMemberInfo(deleteMember);

                return true;
            }
        }
    }

    return false;
}
```

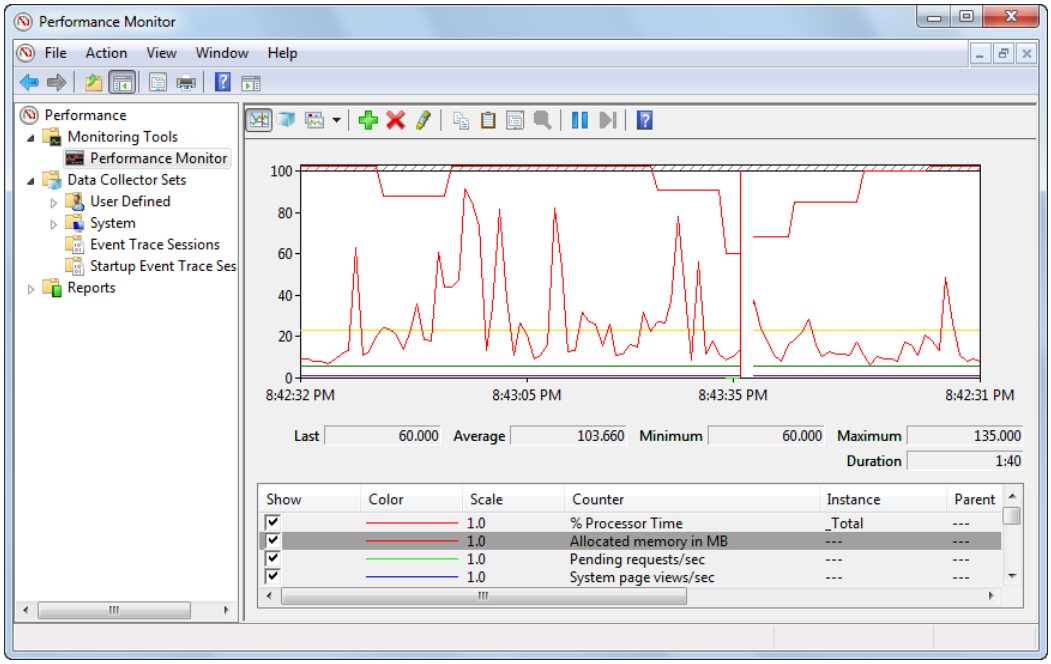
The following example creates an invitation to a group for a user.

```
private bool CreateInvitation()
{
    // Get the group
    GroupInfo group = GroupInfoProvider.GetGroupInfo("MyNewGroup",
CMSContext.CurrentSiteName);

    if (group != null)
    {
        // Create new invitation
        InvitationInfo newInvitation = new InvitationInfo();

        // Set properties
        newInvitation.InvitationComment = "Invitation created by API example.";
        newInvitation.InvitationGroupID = group.GroupID;
        newInvitation.InvitationUserEmail = "admin@localhost.local";
        newInvitation.InvitedByUserID = CMSContext.CurrentUser.UserID;
        newInvitation.InvitationCreated = DateTime.Now;
        newInvitation.InvitationValidTo = DateTime.Now.AddDays(1);

        // Save object
        InvitationInfoProvider.SetInvitationInfo(newInvitation);

        return true;
    }

    return false;
}
```

8.30 Health monitoring

8.30.1 Overview

The Health monitoring module enables website administrators to monitor and record load and performance of a Kentico CMS instance. Monitoring itself is not integrated in Kentico CMS user interface — the module only stores monitored values about the system in Windows performance counters and therefore provides support for monitoring using an external application, e.g. the built-in Performance monitor in Microsoft Windows 7, Windows Vista or Windows Server 2008 R2. The module can be used only if the Kentico CMS instance is running in Full Trust environment.



As mentioned above, monitored values are stored in Windows performance counters. The Performance counters overview topic provides overview of default performance counters that can be used out-of-the-box, as well as information on counters definition XML files, categories where the counters are stored, etc. While Kentico CMS comes with a pre-defined set of default counters, you are not limited to use just the default ones - in the Adding custom counters topic, you can learn how to implement custom performance counters tailored for your specific purposes.

The first thing to do when enabling Health monitoring of an instance of Kentico CMS is registration of the performance counters in Windows. This can be done in three different ways, each of which is described in the Registering performance counters topic. Once the counters are registered, it is also necessary to enable Health monitoring in Kentico CMS settings, as explained in Enabling Health monitoring. When counters are registered and Health monitoring enabled, you can start using an external application to monitor the values. Basics of monitoring using the built-in Performance monitor are explained in the Monitoring using Performance monitor topic.

While most values are monitored and written to the counters by the Kentico CMS application itself, monitoring of some of them requires database access. To optimize performance in this case, it is possible to install a dedicated Windows service and let it handle monitoring of these values instead of the application. To learn more about the installation of Windows service, please refer to the Installing Health monitoring Windows service topic.

8.30.2 Performance counters overview

This topic provides an overview of default Kentico CMS performance counters, their definition XML file and categories where the counters are stored.

Application name web.config key

Before getting to the counters, it is important to explain the following key in the *appSettings* section of the instance's *web.config* file:

```
<add key="CMSApplicationName" value="Default Web Site/CMS" />
```

This key is added to the *web.config* file automatically during installation. In case of IIS installation, path to the instance in IIS is used as its value. In case of Visual Studio web server installation, name of the target web project root folder is used. The value must be less than 60 characters long. In general, the value of the key is used by Kentico CMS Windows services to identify the Kentico CMS instance. Specifically for Health monitoring, the value is used to identify performance counters to which monitored values about the instance should be written.

Important!

In case of Visual Studio web server installation, it is possible that multiple instances running on a single server may have identical values of the key (if the instances are installed into folders with the same names). In this case, you need to ensure that the keys have different values. Otherwise, values from these instances may be written to the same counters.

Performance counter categories

Kentico CMS performance counters are stored in two categories:

- **Kentico - General (<CMSApplicationName>)** - contains general counters monitoring the Kentico CMS instance as a whole. It contains single-instance counters, which means each of the counters can be added to the to the list of monitored counters just once. If a counter is present in this category as well as in the Sites category, its value in this category is a sum of values of all site instances of the counter in the Sites category.
- **Kentico - Sites (<CMSApplicationName>)** - contains site specific counters monitoring particular websites running in the Kentico CMS instance. It contains multi-instance counters, i.e. each of the counters can be added to the list of monitored counters multiple times — once for each website running in the instance. These counters are used only if the *Enable site counters* option is enabled in *Site Manager -> Settings -> System -> Health monitoring*.

The <*CMSApplicationName*> part of the category names is the value of the CMSApplicationName *web.config* key explained above. In case of IIS installation, the value is reversed in the category names so that website name is stated first and the IIS path after it. For example, if the IIS path is **Default Web**

Site/CMS, you would have **CMS/Default Web Site** in the name of the category. This should provide better orientation in the category list in Performance Monitor.

Performance counters definition XML

In ~*AppData**CMSModules**HealthMonitoring*, you can find the **counters.xpc** file. This is a file in XML format which contains definitions of default performance counters for the respective instance of Kentico CMS. It contains definitions of both General and Site counters. When performance counters are registered in Windows, this file is accessed to get the list of counters to be registered.

Apart from this default file, the whole folder structure under ~*AppData**CMSModules*\ is also searched for other files with the **.xpc** extension when counters are registered. The other files can contain definitions of additional performance counters and when found, the counters are registered as well.

The following code is an extract from the **counters.xpc** file:

```
<?xml version="1.0" encoding="utf-8"?>
<Counters>
   <Counter Key="allocatedmemory" Name="Allocated memory in MB" Description="The
   size of allocated memory in megabytes." Type="NumberOfItems32" Enabled="True"
   OnlyGlobal="True" />
   <Counter Key="pendingrequestspersecond" Name="Pending requests/sec"
   Description="The number of pending requests per second." Type="NumberOfItems32"
    Enabled="True" OnlyGlobal="True" PerSecond="True" />
...
</Counters>
```

As you can see in the code extract above, each counter element has the following attributes:

- **Key** – counter key used for its identification.
- **Name** – name of the counter displayed in Performance Monitor or another monitoring tool.
- **Type** – type of the counter, all types are listed and explained at: http://msdn.microsoft.com/en-us/library/system.diagnostics.performancecountertype.aspx.
- **Enabled** – indicates if the counter should be enabled.
- **OnlyGlobal** – indicates if the counter will be included in the *General* category (true) or in both *General* and *Sites* categories (false).

Default performance counters

The following table lists the default counters that are pre-defined in the **counters.xpc** file and registered in Windows by default:

Counter name	Category	Description	Values written by
Allocated memory in MB	Global	The size of memory allocated by the application in Megabytes.	Application
Cache expired items/sec	Global	The number of expired cache items per second.	Application
Cache removed items/sec	Global	The number of items removed from cache per second.	Application

Cache underused items/sec	Global	The number of underused cache items per second.	Application
Content page views/ sec	Global and Sites	The number of content pages viewed per second.	Application
E-mails in queue	Global and Sites	The number of e-mails in E-mail queue.	Application or Windows service[1]
Error e-mails in queue	Global and Sites	The number of e-mails in E-mail queue whose sending failed.	Application or Windows service[1]
Errors	Global	The number of errors in event log since last application restart.	Application
File downloads/sec	Global and Sites	The number of files downloaded per second.	Application
Non-page requests/ sec	Global	The number of non-page requests per second.	Application
On-line users – total	Global and Sites	The total number of on-line users.	Application
On-line users – authenticated	Global and Sites	The number of authenticated on-line users.	Application
On-line users – anonymous	Global and Sites	The number of anonymous on-line users.	Application
Pages not found/sec	Global and Sites	The number of not found pages (404 error) per second.	Application
Pending requests/sec	Global	The number of pending page requests per second.	Application
Robots.txt views/sec	Global and Sites	The number of *robots.txt* page requests per second. **Important:** For values to be written to this counter, IIS must be configured to handle *.txt* extensions. This can be ensured by following a part of the procedure used when configuring custom URL extensions. The procedure is different for IIS 6 (perform all steps on the page) and for IIS 7 or higher (perform only step 1 of Required configuration).	Application
Running SQL queries	Global	The number of running SQL queries.	Application
Running threads	Global	The number of running threads.	Application
Scheduled tasks running	Global	The number of running scheduled tasks.	Application
Scheduled tasks in queue	Global	The number of scheduled tasks in queue.	Application or Windows service[1]
System page views/ sec	Global	The number of system pages viewed per second.	Application

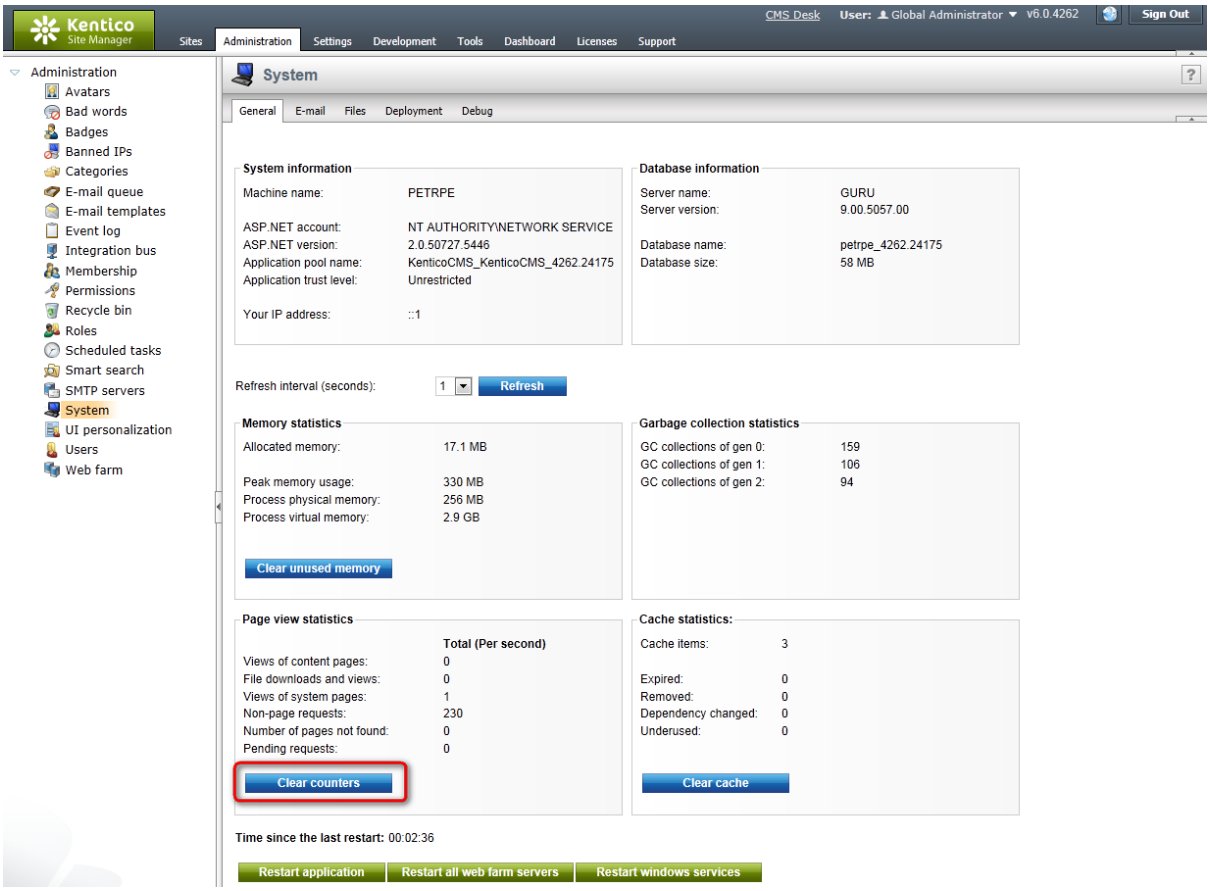
Warnings	Global	The number of warnings in event log since last application restart.	Application

[1] The Windows service is used to write values to these counters only if it is installed and if the **Use external service** option is enabled in **Site Manager -> Settings -> System -> Health monitoring**.

It is also possible to define custom counters to monitor other system values according to your specific needs. For more information on this, please refer to the Adding custom counters topic.

Clearing counter values

By clicking the **Clear counters** button in **Site Manager -> Administration -> System**, it is possible to clear values stored in all counters registered for the current Kentico CMS instance. Due to the fact that Health monitoring is only functional in Full Trust environment, the button is only present in this section if the application is running in Full Trust environment.

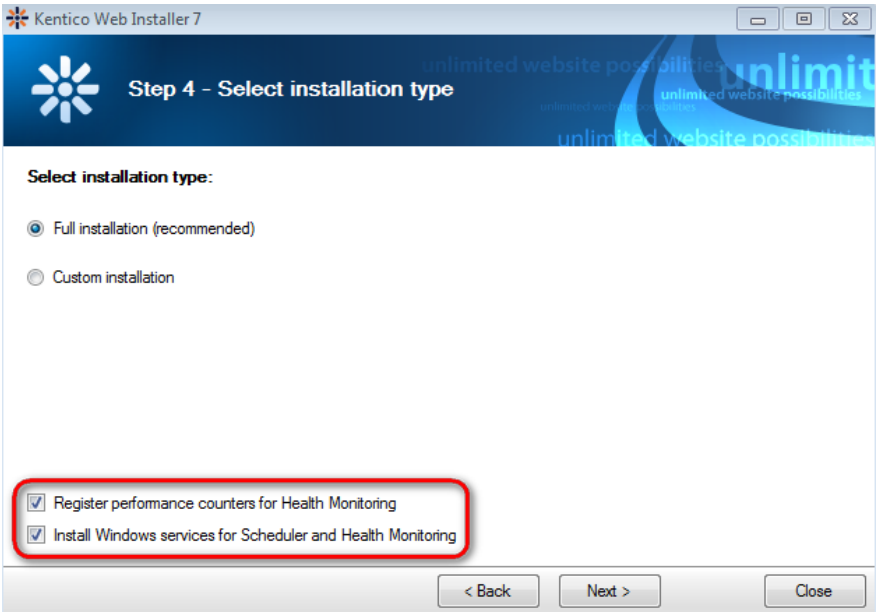


8.30.3 Registering performance counters

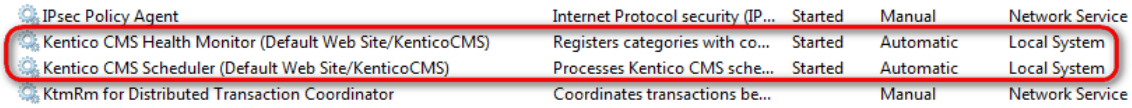
There are three ways how performance counters for a Kentico CMS instance can be registered in Windows.

Registration of counters in Web Installer

The first option of registering performance counters for an instance of Kentico CMS is in Step 4 of the Web Installer. In this step, you can see the **Register performance counters for Health Monitoring** check-box. If you enable it, performance counters will be registered for the currently installed instance of Kentico CMS.



Check the **Install Windows services for Scheduler and Health Monitoring** option to install the **Kentico CMS Scheduler** and **Kentico CMS Health Monitor** services. Installing the service can optimize your application's performance.



Automatic registration of counters by the Windows service

When the Windows service is launched or restarted, it checks if performance counter categories for the respective Kentico CMS instance are registered. If it detects that they are not registered, it performs their registration automatically.

When the Windows service is running, it also monitors creation, editing and deletion of **.xpc** files located anywhere in the folder structure under ~*App_Data**CMSModules*\. When an **.xpc** file in this location is created, edited or deleted, the Windows services performs reload of all counters in both counter categories, ensuring that registered counters match the counters defined in the **.xpc** files.

Manual registration/removal of counters using the Windows service

It is also possible to register performance counters manually. This can be done by going to the **Bin** folder of your current Kentico CMS project folder (typically C:\inetpub\wwwroot\KenticoCMS\bin) and executing the **HealthMonitoringService.exe** file from Windows command line with the following parameters:

```
HealthMonitoringService.exe /webpath="<disk path to web project root>" /
createcounters
```

Similarly, you can use the **HealthMonitoringService.exe** file to remove already registered counters. This is done by executing the file with parameters as follows:

```
HealthMonitoringService.exe /webpath="<disk path to web project root>" /
deletecounters
```

Performance counters in Windows registry

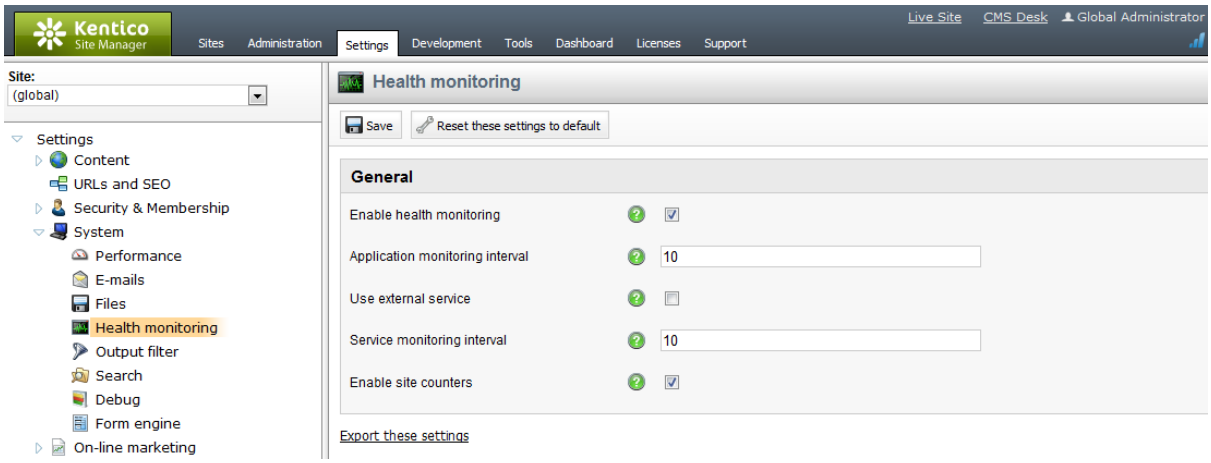
Registration of performance counters is technically performed by adding specific keys to Windows registry. In the registry, they are located in *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services*. The keys represent individual registered counter categories and have the same names as the respective registered counter categories. By deleting the registry keys, you perform removal of the respective counter categories so that they are no longer registered.

8.30.4 Enabling Health monitoring

Once you have Kentico CMS performance counters registered in Windows as described in the Registering performance counters topic, you need to adjust Health monitoring settings in order for monitored values to be written to the counters and to specify how exactly it will be performed.

This can be done in **Site Manager -> Settings -> System -> Health monitoring**, where the following settings can be adjusted:

Enable health monitoring	Indicates if Health monitoring is enabled. i.e. if monitored values are written to both General and Site performance counters related to this instance of Kentico CMS. If disabled, no values are written to any of these performance counters.
Application monitoring interval	Time interval (in seconds). In this periodic interval, the application reads monitored values and writes them to performance counters.
Use external service	Indicates if external Windows service should be used to read and write monitored values to the *Scheduled tasks in queue*, *E-mails in queue* and *Error e-mails in queue* performance counters. As these counters require database access to get their values, using the external service may optimize your application's performance.
Service monitoring interval	Time interval (in seconds). In this periodic interval, the external Windows service reads monitored values and writes them to performance counters. If you are using the Health Monitoring Windows service and change this value, it is necessary to restart the Windows service in order for the new value to be used.
Enable site counters	Indicates if values are written to site specific performance counters. If disabled, values are written to general counters only.

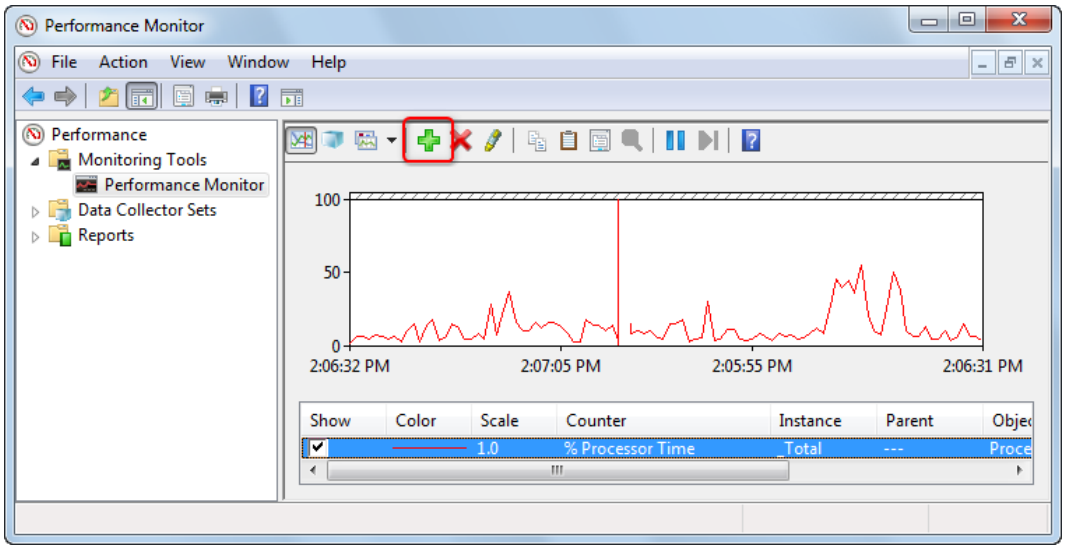


8.30.5 Monitoring using Performance monitor

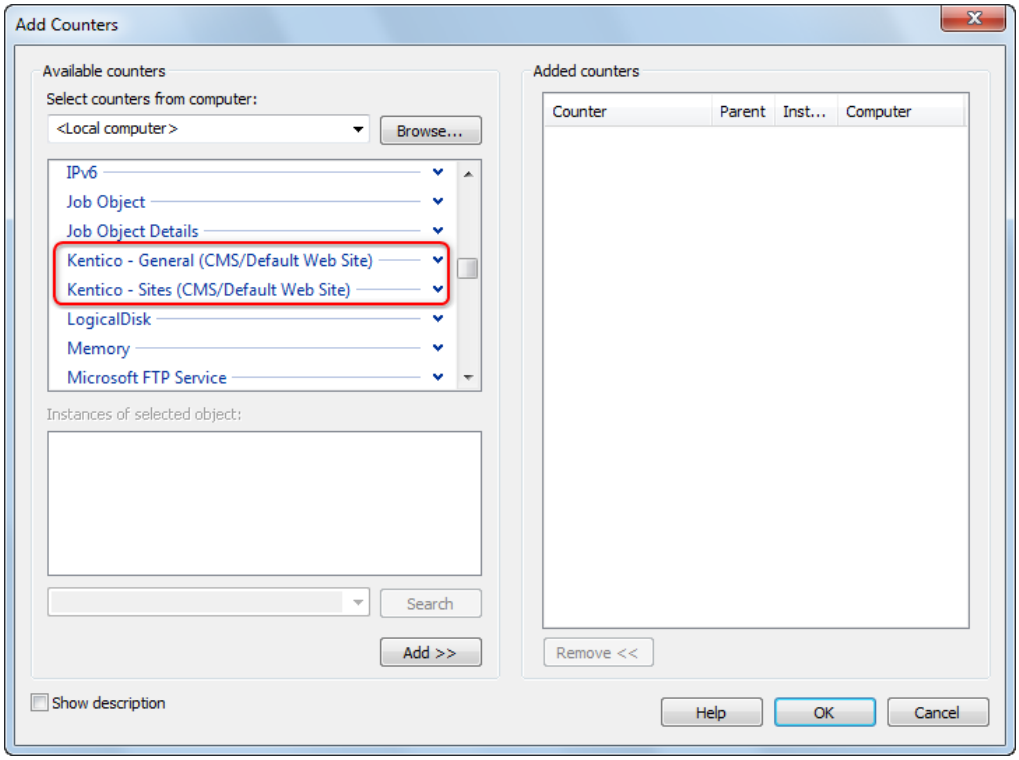
When you have performance counters for your instance of Kentico CMS registered in Windows and Health monitoring settings adjusted, you can start monitoring values written to the counters. There is a number of applications that can be used for this purpose. In this topic, we will use the built-in Performance monitor that is a native part of Windows 7, Windows Server 2008 R2 and Windows Vista.

To execute Performance monitor, type *perfmon* in Windows Start menu search box and press *Enter*.

Performance monitor will be launched. In the tree on the left, select **Monitoring Tools -> Performance monitor**. As you can see, only the default **% Processor Time** counter is monitored initially. To add the Kentico CMS counters, click the **Add** (✚) icon in the toolbar at the top, as highlighted in the following screenshot.



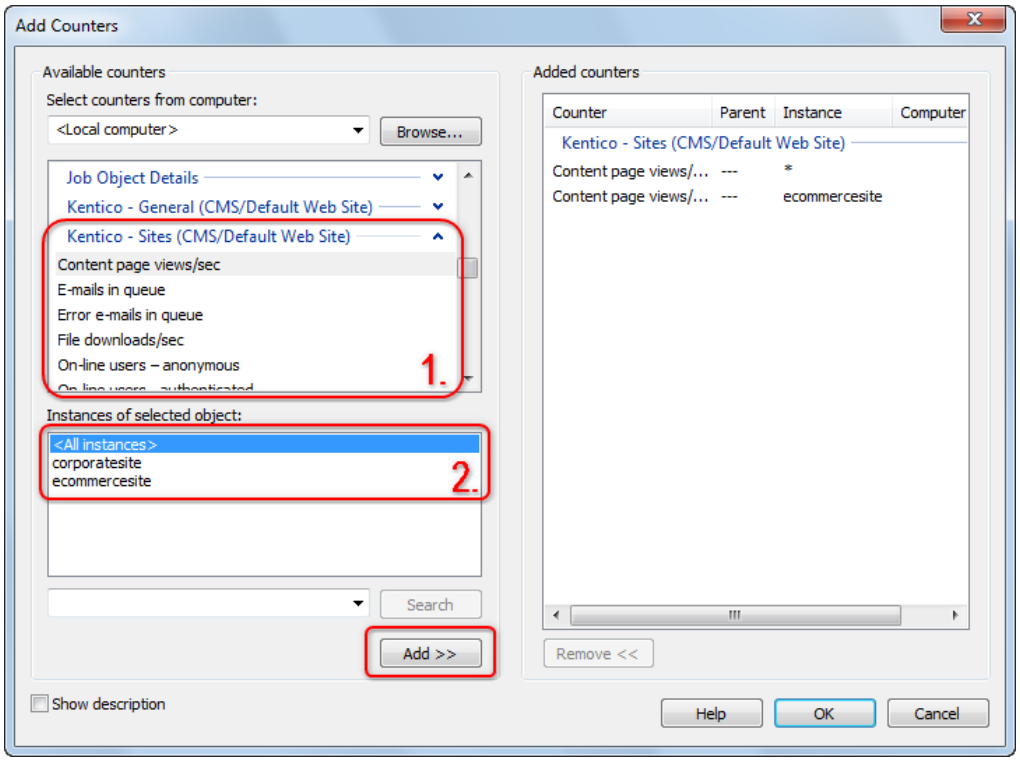
The **Add Counters** window pops up. From the **Select counters from computer** drop-down list, choose **<Local computer>**. In the section below, you will see a list of all counter categories currently registered in Windows. Among them, you will find two counter categories for each Kentico CMS instance with registered performance counters — **Kentico - General (<IIS path>/<IIS website>)** and **Kentico - Sites (<IIS path>/<IIS website>)**.



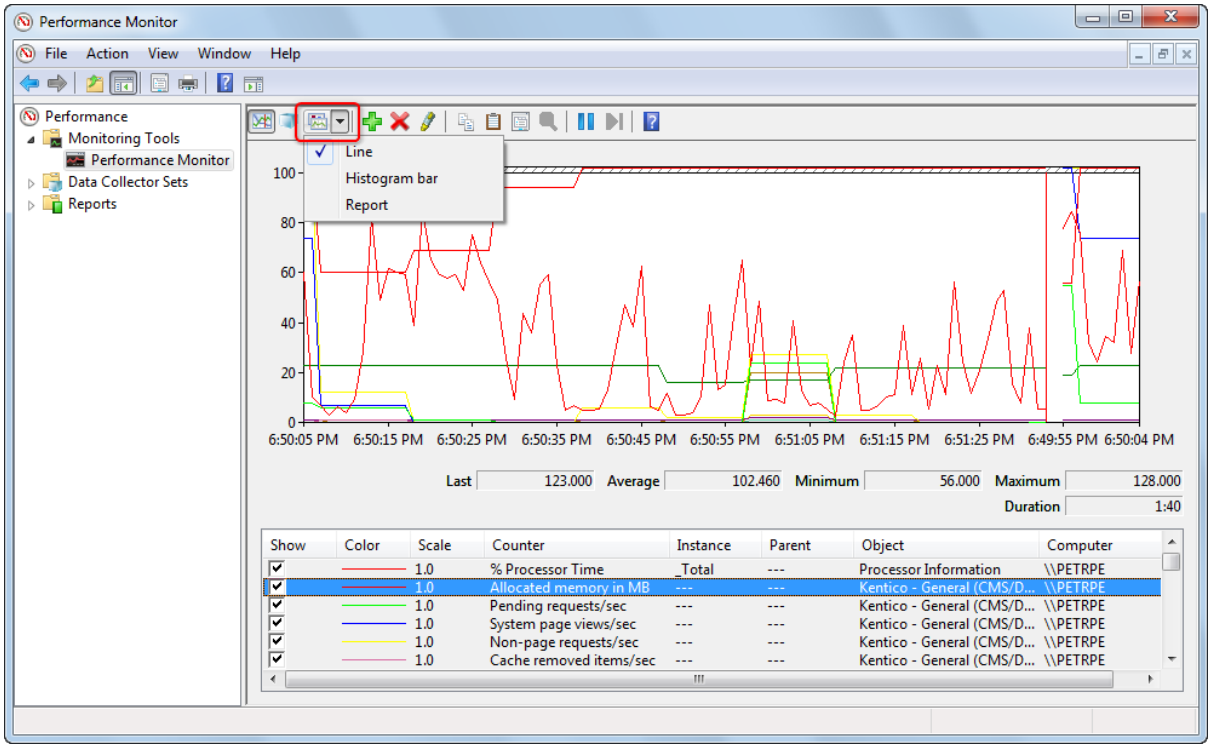
If you click the down-arrow next to a category name, you will expand all counters present in the category (1. in the screenshot below). You can either add all counters in a category by selecting the category and clicking **Add >>**, or just individual counters by selecting the counter and clicking the **Add >>** button.

When a counter from the **Site** category is selected, you can add it either for an individual website, or for all websites running in the instance. To add it for all websites, select the counter, then choose **<All instances>** in the **Instances of selected objects** section (2. in the screenshot below) and click **Add >>**. To add it for an individual website, select the counter, then choose the website's code name in the **Instances of selected objects** section and click **Add >>**.

Added counters will be listed in the **Added counters** section on the right. Once you have all counters added, click **OK** to return back to the monitoring UI.



Back in **Monitoring Tools -> Performance Monitor**, you can see that values in counters are now shown in a graph and reflect the real activity of the Kentico CMS instance. You can switch between different ways how monitored values are displayed using the **Graph type** drop-down list highlighted in the screenshot below.



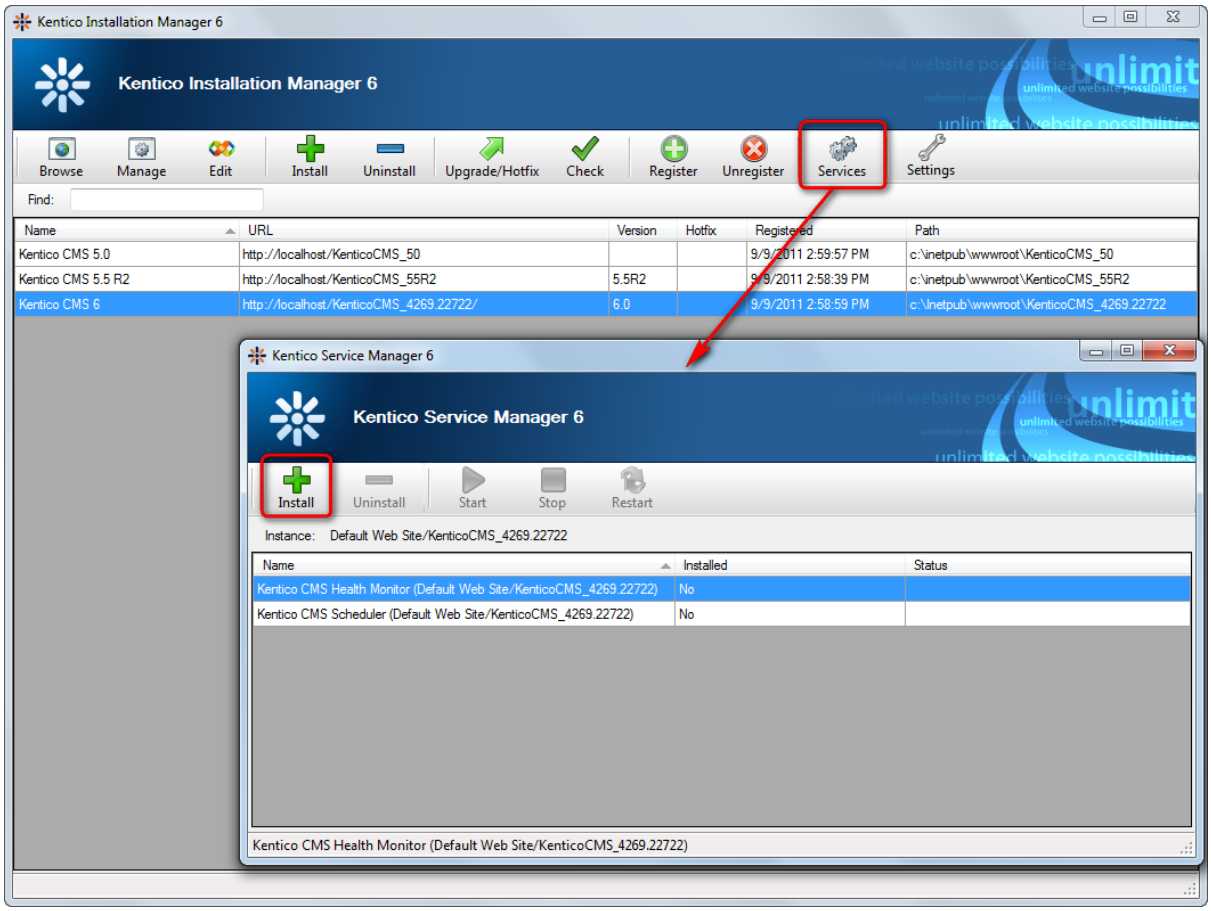
8.30.6 Installing Health monitoring Windows service

Kentico CMS comes with a dedicated Windows service for Health monitoring purposes. This service can be used to additionally register performance counters and to write values to the *Scheduled tasks in queue*, *E-mails in queue* and *Error e-mails in queue* performance counters. As these three counters require database access to get their values, using the Windows service for their monitoring instead of the application itself can provide better performance.

The following text describes how the service can be installed. Please note that a specific configuration is also required for the service to write values to the mentioned performance counters. See Enabling Health monitoring for more details.

Installing and uninstalling the Health monitoring Windows service using Kentico CMS Service Manager

The easiest way to install or uninstall the Health monitoring Windows service is to use the Kentico Service Manager utility. The utility can either be launched from Kentico CMS program group in Windows Start menu, or from Kentico Installation Manager, as shown in the screenshot below.



Installing the Health monitoring Windows service from the command line

To install the Windows service from the command line, you need to use Installer Tool (InstallUtil.exe), which is a native part of the .NET Framework. The following steps describe installation of the Windows service using the Installer Tool.

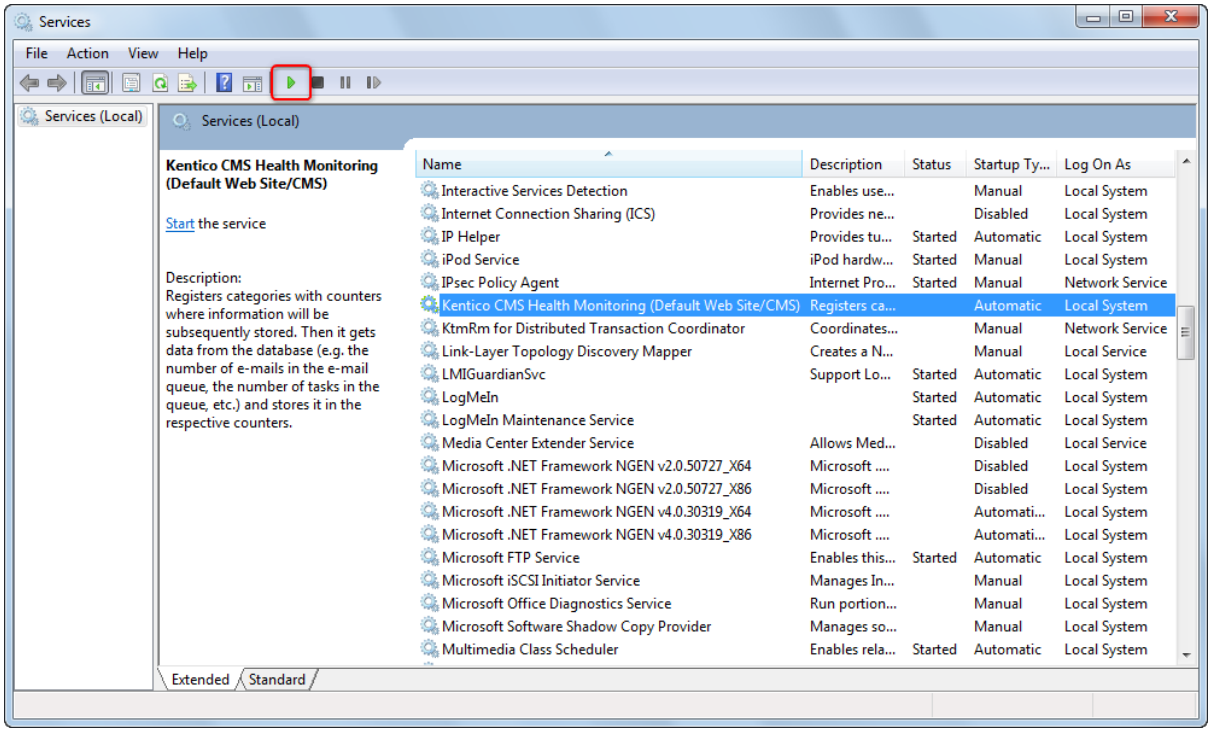
1. Open Windows command line (type *cmd* in the Start menu search box) and navigate to the .NET folder containing *InstallUtil.exe* (e.g. *c:\Windows\Microsoft.NET\Framework64\v4.0.30319*).

2. Execute *InstallUtil.exe* from Windows command line with parameters as shown below. The parameters mean the following:

- **/webpath** - path to the root folder of the Kentico CMS instance for that you want to install the Windows service.
- **second parameter** - path to the *HealthMonitoringService.exe* file in the *Bin* folder inside application root (typically *c:\inetpub\wwwroot\KenticoCMS\bin\HealthMonitoringService.exe*).
- **/LogToConsole** - optional parameter defining if installation progress will be logged to console.
- **/i** - if this parameter is used, the Installer Tool performs installation of the Windows service.

```
InstallUtil /webpath="C:\inetpub\wwwroot\KenticoCMS" "c:
\inetpub\wwwroot\KenticoCMS\bin\HealthMonitoringService.exe" /LogToConsole=true /i
```

3. To verify that the service is installed successfully, open Services management console (type *services.msc* into the Start menu search box). In the list of installed services, you should see the **Kentico CMS Health Monitoring (<CMSApplicationName web.config key value>)** service. The service is not running initially. To start it, click the **Start Service** button in the top toolbar, as highlighted in the screenshot below.



Uninstalling the Health monitoring Windows service from the command line

Uninstalling the Health monitoring Windows service can be performed exactly the same way as described above, while you only need to use the **/u** parameter instead of **/i** at the end of the command:

```
InstallUtil /webpath="C:\inetpub\wwwroot\KenticoCMS" "c:
\inetpub\wwwroot\KenticoCMS\bin\HealthMonitoringService.exe" /LogToConsole=true /u
```

After executing this command, the Health monitoring Windows service for the instance of Kentico CMS specified by the **/webpath** parameter will be uninstalled.

8.30.7 Adding custom counters

Besides the default performance counters listed in the Performance counters overview topic, it is also possible to implement custom performance counters to monitor other values according to your specific needs.

In the following example, you will learn how to implement a custom performance counter to monitor the number of accesses to the **Home.aspx** page of any website running in the system. The counter will be available in both the **General** and the **Sites** counter categories, enabling you to monitor access to the **Home.aspx** page of each individual website or globally for all websites in the Kentico CMS instance.

1. Create a new XML file where the custom counter's definition will be located. Give it the name e.g. *MyCounters.xpc* and add it to ~*App_Data**CMSModules**HealthMonitoring* (or any other folder under ~*App_Data**CMSModules*\). Its content should be as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<Counters>
  <Counter Key="requestshomepage" Name="Home.aspx page requests" Description="The
number of Home.aspx page requests." Type="NumberOfItems32" Enabled="True"
OnlyGlobal="False" />
</Counters>
```

2. We need to create a new module class in **App_Code** (or **Old_App_Code** if you installed the project as a web application), e.g. ~*App_Code**Custom**CustomCounterModule.cs*, which will extend the **CMSModuleLoader** partial class. The code extract below shows what the class should look like for the purpose of this example.

At first, a private attribute *mTotalHomePageRequests* is added to the class, which will hold the value logged in the counter. Its value is made accessible by the *TotalHomePageRequests* public property. The code within the *Init()* method, which is executed on each application start, ensures registration of the method in the *OnLogCustomCounter* event. When the event occurs, the *HealthMonitoringLogHelper_OnLogCustomCounter* method is executed.

```
using System;

using CMS.GlobalHelper;
using CMS.SettingsProvider;
```

```
using CMS.SiteProvider;

[CustomCounterModuleLoader]
public partial class CMSModuleLoader
{
    /// <summary>
    /// Counter of total Home.aspx page requests.
    /// </summary>
    private static CMSPerformanceCounter mTotalHomePageRequests = null;

    /// <summary>
    /// Counter of total home page requests.
    /// </summary>
    public static CMSPerformanceCounter TotalHomePageRequests
    {
        get
        {
            if (mTotalHomePageRequests == null)
            {
                mTotalHomePageRequests = new CMSPerformanceCounter();
            }

            return mTotalHomePageRequests;
        }
    }


    /// <summary>
    /// Module registration
    /// </summary>
    private class CustomCounterModuleLoaderAttribute : CMSLoaderAttribute
    {
        /// <summary>
        /// Initializes the module
        /// </summary>
        public override void Init()
        {
            CMS.CMSHelper.HealthMonitoringLogHelper.OnLogCustomCounter += new
CMS.CMSHelper.HealthMonitoringLogHelper.LogCustomCounterHandler
(HealthMonitoringLogHelper_OnLogCustomCounter);
        }

        /// <summary>
        /// Handles custom event of health monitoring.
        /// </summary>
        /// <param name="counter">Counter</param>
        /// <returns>CMS performance counter</returns>
        private static CMSPerformanceCounter
HealthMonitoringLogHelper_OnLogCustomCounter(Counter counter)
        {
            if (counter.Key.ToLower() == "requestshomepage")
            {
                return TotalHomePageRequests;
            }

            return null;
        }
```

```
    }
}
```

3. Add the *OnLoad* method below to *~\CMSPages\PortalTemplate.aspx.cs*. The code ensures that the counter value is incremented each time a page with the */home* node alias path is accessed.

```
    protected override void OnLoad(EventArgs e)
    {
        // Increment Home.aspx page requests counter
        if (CMS.CMSHelper.CMSContext.CurrentPageInfo.NodeAliasPath.Equals("/home",
StringComparison.InvariantCultureIgnoreCase))
        {
            CMSModuleLoader.TotalHomePageRequests.Increment
(CMS.CMSHelper.CMSContext.CurrentSiteName);
        }

        base.OnLoad(e);
    }
```

4. At this point, the process is different depending on if the Windows service was running at the time when you performed step 1 of this example.

4.a If it was running, the counters should be already registered in both categories as the Windows service performs automatic reload of counters when an **.xpc** file in the folder structure under *~\AppData\CMSModules* is created, edited or deleted.

4.b If the Windows service was not running when you performed step 1, you will need to register the counters manually by executing the Windows service with respective parameters, as explained in the Registering performance counters topic. To do it, open Windows command line, navigate to the **Bin** folder inside the Kentico CMS installation folder (typically *C:\Program Files\Kentico CMS\<version>\Bin*) and execute the **HealthMonitoringService.exe** file with the following parameters:

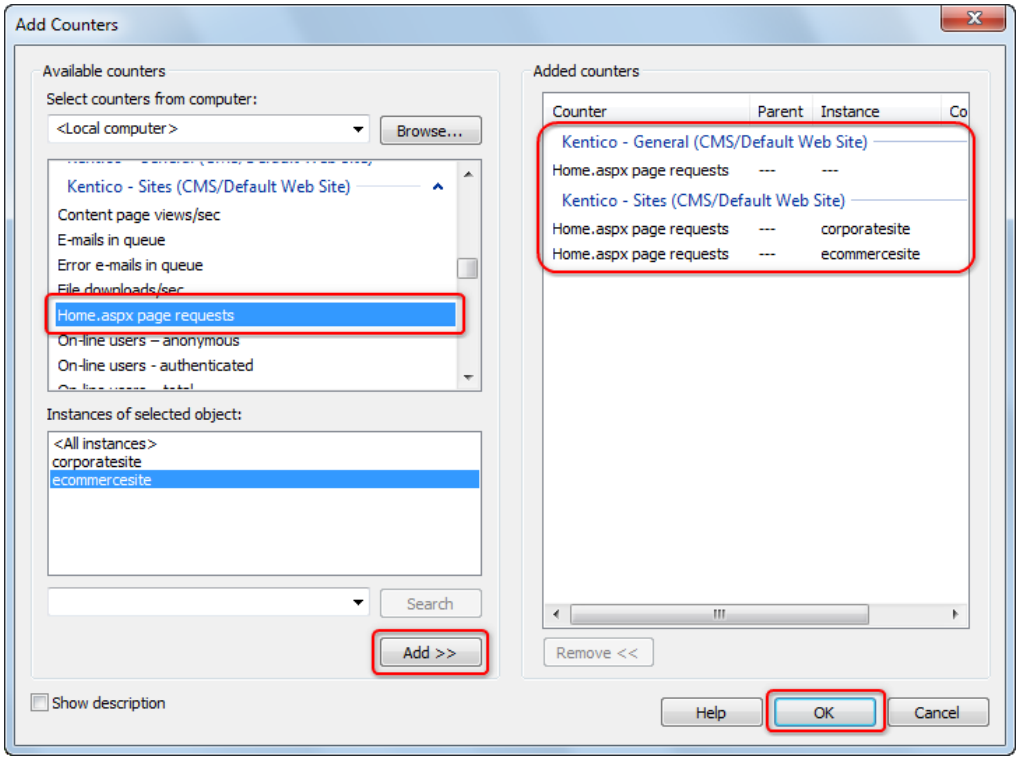
```
HealthMonitoringService.exe /webpath="<disk path to web project root>" /
createcounters
```

This action will reload all counters in both categories, including the newly added one.

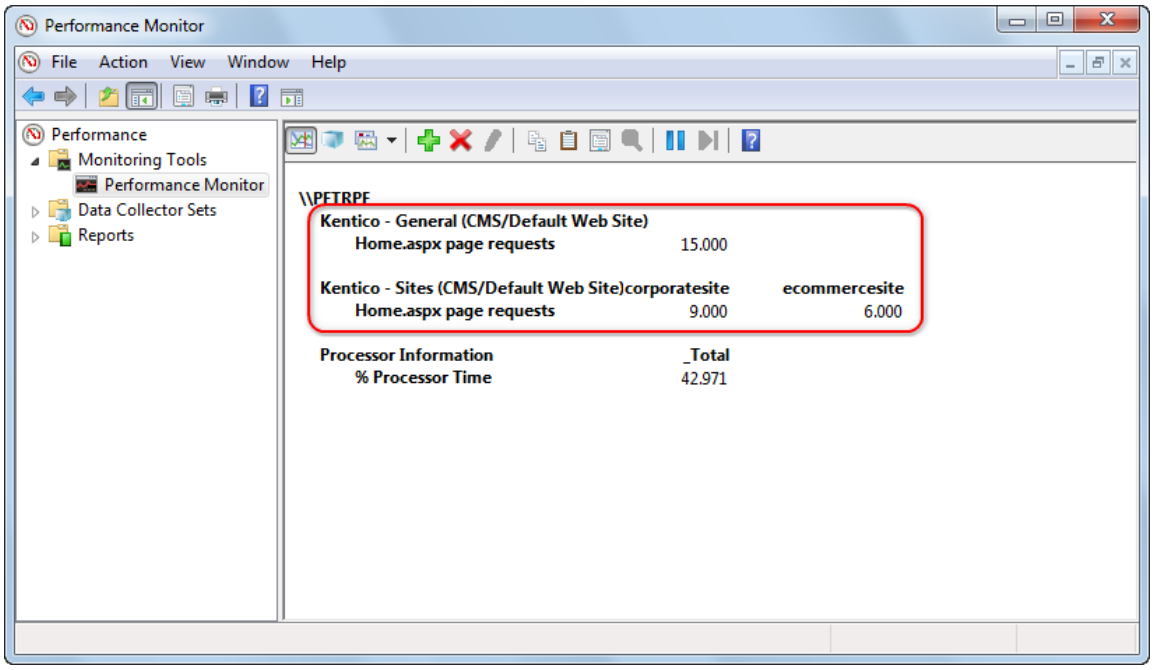
> **Custom counters and Health Monitoring Windows service**
>
> Values of all custom counters can be logged only by the application itself, i.e. it is not possible for values of custom counters to be logged by the Windows service. The only purpose of using the service in this step is to register the added counter, as described in the Registering performance counters topic.

5. Verify that you have Health monitoring settings adjusted correctly in **Site Manager -> Settings -> System -> Health monitoring**, launch Performance monitor and open the dialog for adding monitored counters (as explained in Monitoring using Performance monitor). You should see the new counter

present in both counter categories of the current Kentico CMS instance. Add the counter using the **Add >>** button, once from the **General** category and once for each website (if you have more than in your Kentico CMS instance) from the **Sites** category. Then click **OK**.



6. Before trying out the functionality, go to **Site Manager -> Administration -> System** and restart the application using the **Restart application** button. Now try accessing the **Home.aspx** page multiple times and see how the value gets incremented after each access.



8.31 Image gallery

8.31.1 Overview

The Image gallery module is used for effortless creating of image gallery pages. By means of this module, you can display images which are saved in the content tree as documents. The module encompasses four page templates and three web parts suitable for creating image galleries.

However, using the Image gallery module is not the only way to display images on your web pages. You can achieve similar functionality by using the Media libraries module. The Media libraries module provides an alternative way of creating a gallery and is suitable if large numbers of stored images are to be displayed or if you want to make audio, video and other file types accessible from your web pages. Another option consists in attaching an image to a document using the Attachment image gallery web part.

In the Image gallery chapter, you can learn what web parts are available in the Image gallery module (learn here how to do it), what page templates can be used (as explained here), you can learn how to import files or even the whole folder structure from the disk to Kentico CMS content repository (as referenced here) and you can also learn how to use transformations to alter the appearance of your gallery (as explained here).

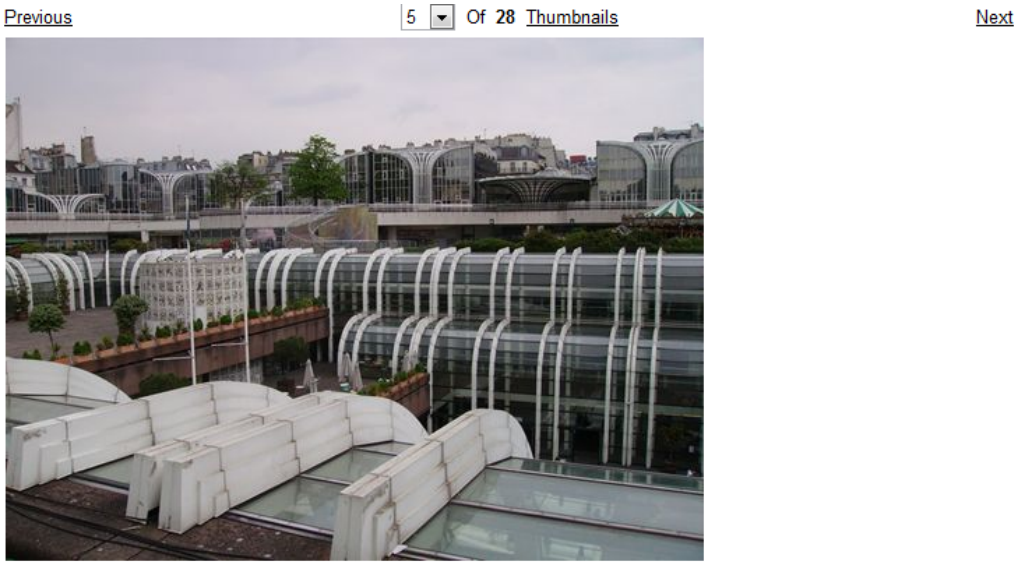
8.31.2 Available web parts

There are three web parts suitable for creating image galleries. From the **Select web part** dialog window, you can select the **Image gallery**, **Lightbox gallery** and **Content slider** web parts in the **Listings and viewers** category.

Image gallery

This is the basic web part for image galleries. In its initial view, it displays a set of picture thumbnails:

After clicking one of the thumbnails, the detail view will be displayed:



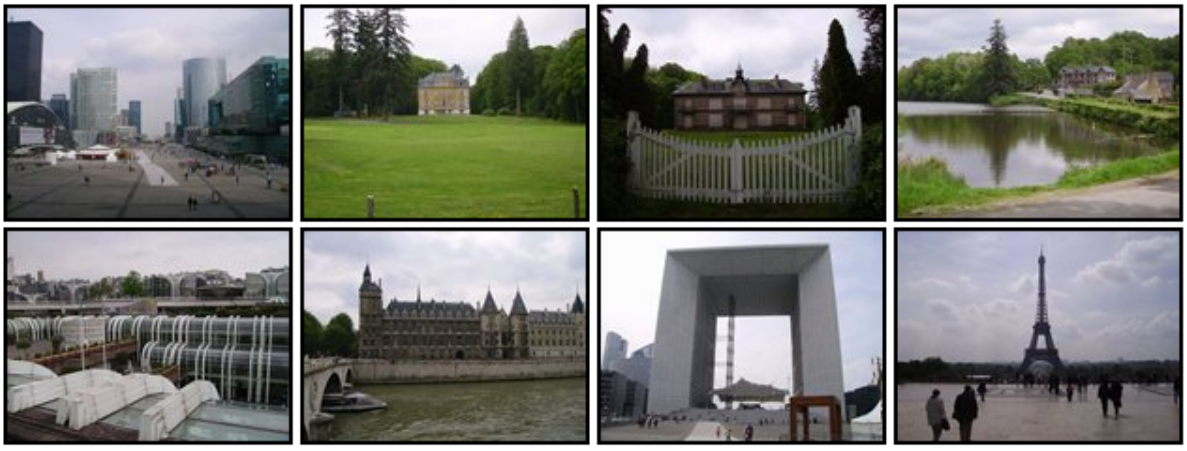
Besides the usual parameters common to all web parts, these properties can be set to customize the appearance of the gallery:

Transformations	
Detail transformation	Transformation used for displaying a selected image.
Thumbnail transformation	Transformation used for displaying gallery thumbnails.
Layout	

Number of columns	Number of thumbnail columns in the thumbnail view.
Rows per page	Number of thumbnail rows per page in the thumbnail view.
Repeat direction	Determines the direction in which items should be displayed when multiple columns are used - can be either vertical or horizontal.
Paging	
Paging mode	Paging parameter transfer type: Query string - the paging parameter is transferred through URL Postback - the actual page is transferred through ViewState, no URL parameter is used
Query string key	Name of the URL parameter containing the page number.
Show first and last buttons	If checked, buttons leading to the first and last pages of the gallery will be displayed.
Show buttons on top	If checked, paging buttons will be shown above the thumbnails. Otherwise, they will be displayed below them.

Lightbox gallery

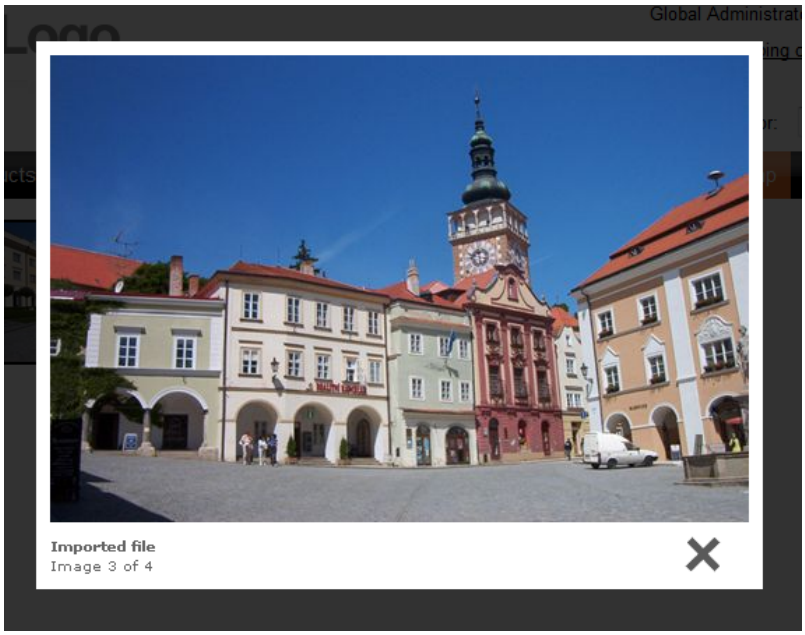
This web part's thumbnail view is similar to that of the **Image gallery** web part:



Displaying results 1-8 (of 28)

|< < 1 - 2 - 3 - 4 > >|

After clicking one of the thumbnails, the whole page will be grayed out and a lightbox with the selected image will be displayed on the top of it, as you can see in the screenshot below:



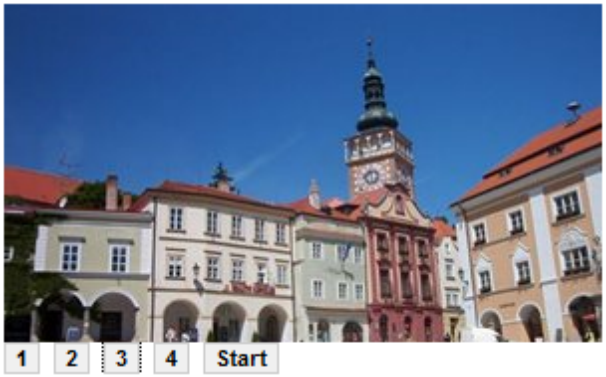
Here is a list of properties specific to the **Lightbox gallery** web part:

Transformations	
Transformation	Transformation used for displaying the list of thumbnails.
Alternating transformation	Transformation used for even items in the thumbnail view.
Selected item transformation	Transformation used in the detail view mode.
Item separator	Separator displayed between thumbnails.
Nested controls ID	Sets the nested controls IDs. Use ';' as a separator; Example: myRepeaterID;myDatalistID;myRepeaterID2 This property replaces the previously used NestedRepeaterID and NestedDataListID properties. If you are still using these properties, no changes to functionality will occur, but it is advisable to rewrite your code to use the new property instead.
Paging	
Enable paging	Indicates if paging is enabled. If unchecked, all thumbnails in the gallery will be displayed on a single page.
Paging mode	Paging parameter transfer type: Query string - the paging parameter is transferred through URL Postback - the actual page is transferred through ViewState, no URL parameter is used
Pager position	Determines position of the pager. Available options are *Bottom*, *Top* and *Top and bottom.*
Page size	Number of thumbnails displayed per page.

Query string key	Name of the URL parameter containing the page number.
Show first and last buttons	If checked, buttons leading to the first and last page of the gallery will be displayed.
LightBox Configuration	
Always visible navigation buttons	Indicates whether the navigation buttons are always visible, not only on mouse over.
Frame width	Width of the LightBox frame.
Frame height	Height of the LightBox frame.
Path to external CSS file	URL path to the external CSS file required by LightBox.
Overlay opacity	Opacity of LightBox background. Enter values ranging from 0 (transparent) to 1 (opaque black).
Animate	Enables LightBox animation.
Load delay	Load delay time (in miliseconds). If you are using automatic resizing, this value indicates how long the lightbox will take to reach the element size. If you have problems with displaying lightbox content, try to use a higher value.
Resize speed	Defines the speed of resizing images. Choose values ranging from 1 (slowest) to 10 (fastest).
Border size	Size of the image border.
Loading image	Image displayed while loading the LightBox image.
Close button image	Image of the Close button.
Previous button image	Image of the Previous button.
Next button image	Image of the Next button.
Group name	Name of the LightBox group.

Content slider

The **Content slider** is a web part that can be used for displaying various document types, hence it is also very suitable for displaying images. Contrary to the previous two web parts, the **Content slider** provides no thumbnail view. It displays a full sized image slide show with a pager below. The pager allows for browsing through the images using the numbered buttons. After clicking any of these buttons, the slide show stops and the **Start** button appears. This button launches the slide show again.



Specific properties of the **Content slider** web part:

Transformations	
Transformation	Transformation used for displaying the list of thumbnails.
Alternating transformation	Transformation used for even items in the thumbnail view.
Item separator	Separator displayed between thumbnails.
Nested controls ID	Sets the IDs of nested controls as specified in the transformation code.
Div options	
Width (px)	Width of the scrolling text area in pixels.
Height (px)	Height of the scrolling text area in pixels.
Style	Style assigned to the DIV tag of the area.
JavaScript options	
FadeIn time (milliseconds)	Fade in time of the image.
FadeOut time (milliseconds)	Fade out time of the image.
Break time (milliseconds)	Time for which the image will be displayed.
Auto start	If checked, the slide show will automatically start from the beginning.

8.31.3 Available page templates

There are three basic page templates that can be used for image galleries. In **CMS Desk -> Content**, create a new **Page (menu item)** using a page template. From the **Use page template** dialog window, you can select the **Image gallery**, **Lightbox gallery**, **Sliding gallery** and **List of galleries** templates in the **Images** category.

List of galleries

This page template is used for displaying a list of all galleries under a selected path. For each gallery, it displays a thumbnail with a gallery name above it. By clicking one of the thumbnails, you will be redirected to the main page of the gallery.

The **Teaser image** of each gallery's menu item is used as the thumbnail in the list of galleries. To change some of the thumbnails, select the appropriate gallery's menu item in the content tree and switch to **CMSDesk -> Edit -> Form**. On the displayed page, select a new **Teaser image** and click **Save**. The selected image will now be displayed as a thumbnail of the gallery in the list of galleries.

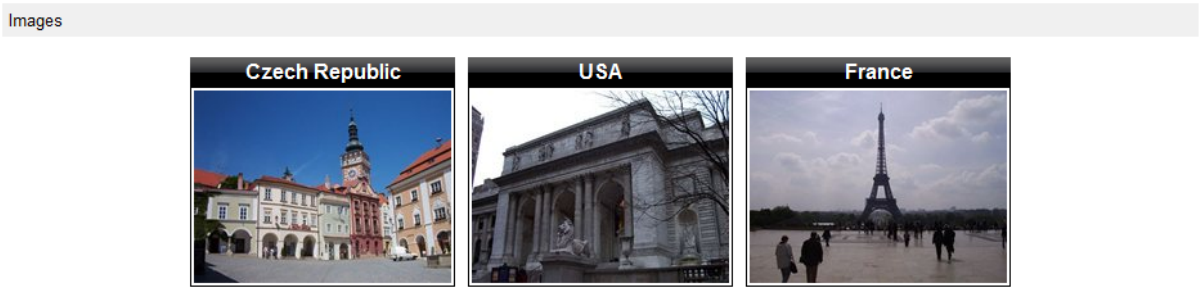


Image gallery

This is a basic page template used for displaying image galleries. It uses the **Image gallery** web part for displaying images under a given path in the content tree. See the Available web parts chapter for detailed info.

Lightbox gallery

This page template uses the **Lightbox gallery** web part for displaying images. See the Available web parts chapter for detailed info.

Sliding gallery

Displays images using the **Content slider** web part.

8.31.4 Importing images

Images used in galleries are imported the same way as any other files, as described in the File import chapter.

8.31.5 Transformations

Use of transformations is essential for all **Image gallery** web parts. You can view and alter transformations in the **Transformations** section of a web part's property configuration dialog. For each transformation property, you can **Select** a predefined transformation from a list, **Edit** the current transformation or create a **New** one by means of the respective buttons.

Here are some examples of how you can alter the appearance of the **Image gallery** web part. This is the default thumbnail transformation code of the **Image gallery** web part:

```
<a href="?imagepath=<%# System.Web.HttpUtility.UrlEncode(DataBinder.Eval
(Container, "DataItem.NodeAliasPath").ToString()) %>">
<%# IfEmpty(Eval("FileAttachment"), "no image", "<img alt=\"" + Eval("FileName") +
"\" src=\"" + GetFileUrl("FileAttachment") + "?maxsidesize=180\" border=\"0\" /
```

```
>") %>
</a>
```

And this is how the gallery created using this transformation looks like:

In the following example, we will add a red border around each thumbnail.

```
<div style="border: solid 3px Red">
<a href="?imagepath=<%# System.Web.HttpUtility.UrlEncode(DataBinder.Eval
(Container, "DataItem.NodeAliasPath").ToString()) %>">
<%# IfEmpty(Eval("FileAttachment"), "no image", "<img alt=\"" + Eval("FileName") +
"\" src=\"" + GetFileUrl("FileAttachment") + "?maxsidesize=180\" border=\"0\" /
>") %>
</a>
</div>
```

This is how the result looks like:

This example shows how to display the file name and date and time of creation for each thumbnail in the gallery:

```
<a href="?imagepath=<%# System.Web.HttpUtility.UrlEncode(DataBinder.Eval
(Container, "DataItem.NodeAliasPath").ToString()) %>">
<%# IfEmpty(Eval("FileAttachment"), "no image", "<img alt=\"" + Eval("FileName") +
"\" src=\"" + GetFileUrl("FileAttachment") + "?maxsidesize=180\" border=\"0\" /
>") %>
</a>
<%# Eval("DocumentName") %> <br/>
<%# GetDateTime("DocumentCreatedWhen") %>
```

Transformations for the Lightbox gallery web part

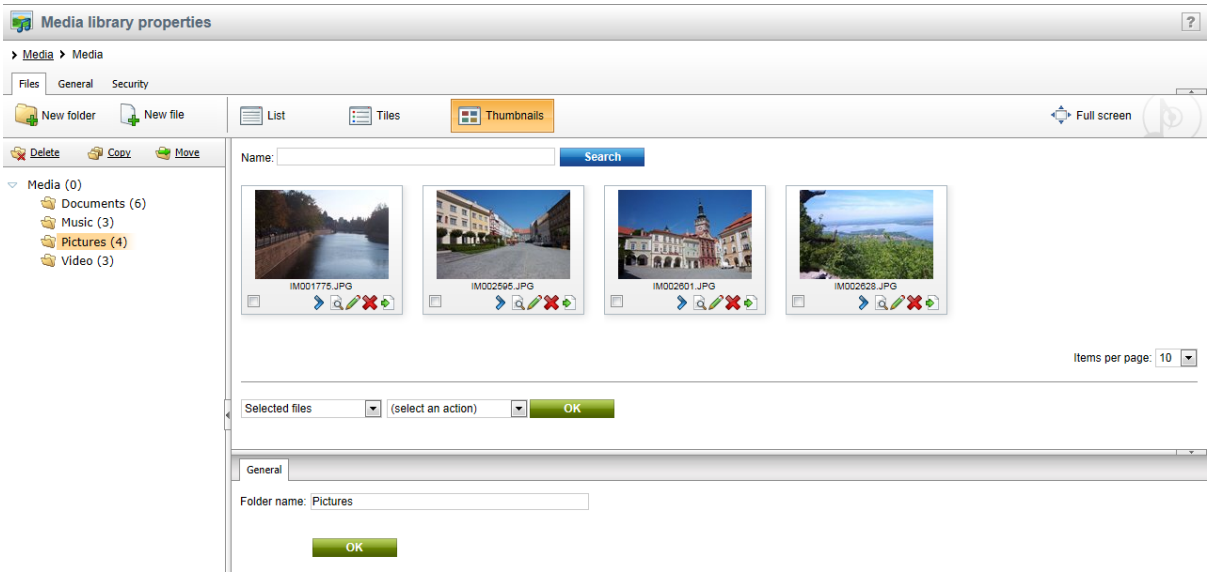
When writing a custom transformation for the **Lightbox gallery** web part, it is necessary to use the 'rel' and 'rev' parameters as highlighted in the transformation code below. The 'title' parameter is used to determine the description of the image displayed in the lightbox.

```
<a href="<%# GetDocumentUrl() %>" rel="lightbox[group]" rev="<%# Eval
("NodeAliasPath") %>"
title="<%# Eval("FileDescription") %>"><img src="<%# GetFileUrl
("FileAttachment") %>?maxsidesize=150"
alt="<%# Eval("FileName") %>" /></a>
```

8.32 Media

8.32.1 Overview

The Media libraries module enables storing of various files, such as photos, sound, videos, package files, presentation files, etc. This means that not only media files, but also other types of files can be stored in media libraries. Media libraries can either be global or related to a particular group.



However, the Media libraries module is not the only way to store files of various kinds within the system. Please refer to the Content management -> File management chapter of this guide for more details.

- To learn how to create media libraries, please refer to the Creating media library topic.
- To learn how to upload files to your media libraries, how to manage these files and what file types are supported, please refer to the Media library content chapter.
- To learn how to use the Media library web part and the built-in WYSIWYG editor in your media libraries, please refer to the Displaying files from media library chapter.
- To learn how to adjust your media library settings and how to configure your media library custom storage, custom file types and the maximal uploaded file size, please refer to the Settings chapter.
- To learn how to assign media library permissions and how to deal with secured and non-secured libraries on your site, please refer to the Security chapter.

More examples of the use of the Media libraries module are available in Kentico CMS Community Site Guide, as describe here.

8.32.2 Community site examples

More examples of the use of the Media libraries module are available in Kentico CMS Community Site Guide. Please note that these practical examples do not concern the whole functionality of the module but focus on its use in a broader context of the Community Site sample website:

- See Part 1 -> Media libraries -> Current functionality in Kentico CMS Community Site Guide: A brief description of the functionality of the Media libraries module.
- See Creating custom media libraries in the same section of the Community Site Guide: An example of creating a new global media library and publishing it on the site.
- See Publishing more than one global media library in the same section of the Community Site Guide: An example of how to publish more than one global media library on the site.
- See Community Site Guide -> Part 2 -> Pre-development tasks -> Creating a sample Media library in Kentico CMS Community Site Guide: An example of creating a sample media library.
- See Creating the Blogs section -> Creating the Media page in the same section of the Community Site Guide: An example of creating the Media page using the Media gallery web part under the Blogs section.
- See Creating the Groups section -> Creating the Media page in the same section of the Community

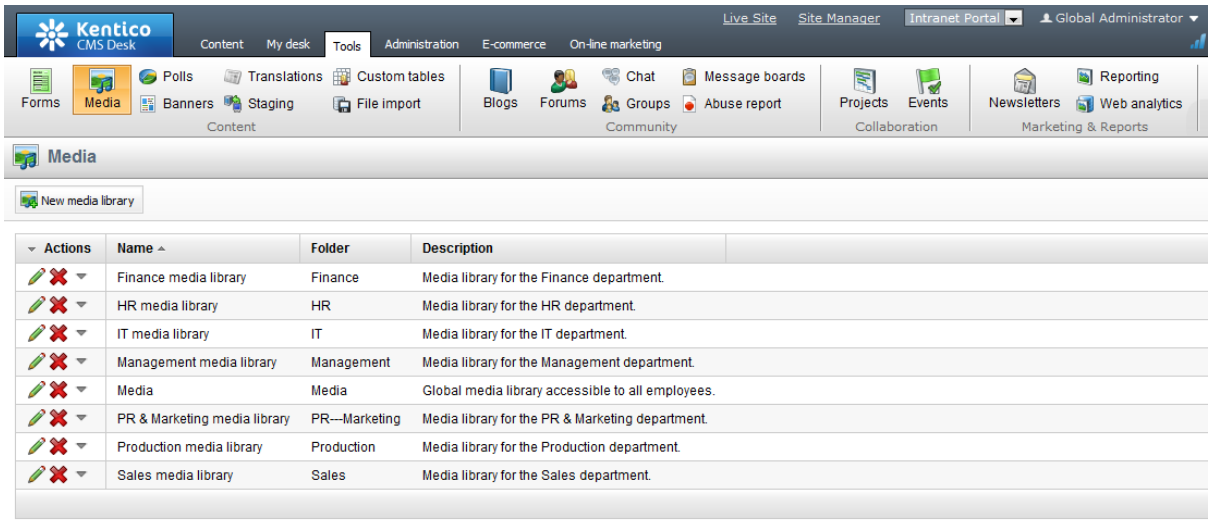
Site Guide: An example of creating the Media page using the Media gallery web part under the Groups section.

8.32.3 Creating media library

There are two ways how media libraries can be created:

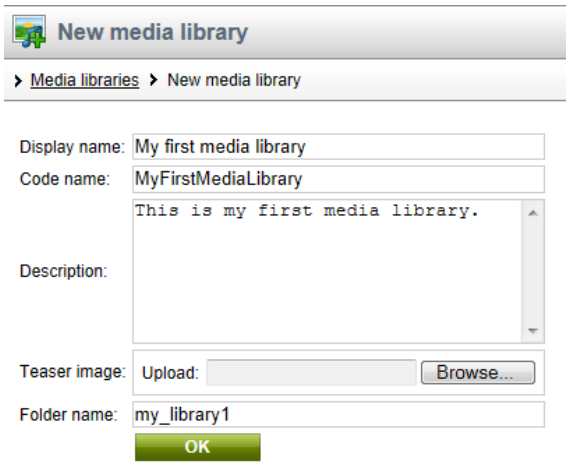
- **Live site** - group administrators can create group media libraries on the live site using the **Group profile** web part.
- **Administration interface** - site administrators can create global media libraries in **CMS Desk -> Tools -> Media libraries** or group media libraries in **CMS Desk -> Tools -> Groups -> Edit (✎) -> Media libraries**.

1. In all of these cases, media libraries can be created by clicking **New media library** above the list of libraries, as you can see in the screenshot below.

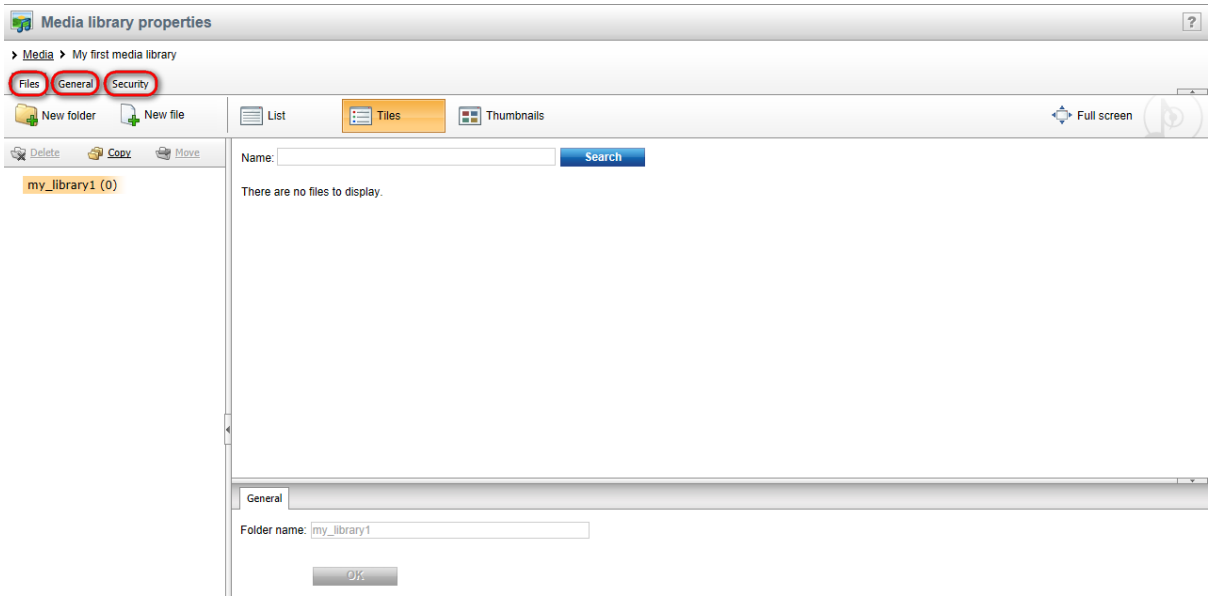


2. After clicking the link, the following details need to be entered:

- **Display name** - the name of the media library displayed in the administration interface and on the live site.
- **Code name** - the name of the media library used by developers in the code.
- **Description** - the text describing the media library.
- **Teaser image** - the image used as the media library teaser.
- **Folder name** - the name of the folder where files will be stored. This folder will be created under **<web project>\<site name>\media** or custom location.



3. Click **OK** to create the library. After doing so, you will be redirected to the **Media library properties** interface - just as if you clicked the **Edit** (✎) icon in the list of libraries.



The **Files** tab allows the management of the media library content as described in more detail in the Media library content subchapter. The **General** tab is used to edit media library general properties that you entered in step 2. Finally, the **Security** tab enables you to adjust permissions to work with folders and files in the given media library. For more details on how to adjust permissions in media libraries, please refer to the Media library permissions topic.

8.32.4 Media library content

8.32.4.1 Overview

In this subchapter you will learn how to deal with content in a media library.

- To learn how to upload files into media libraries, please refer to the Uploading files topic.
- To learn how to manage files stored in a media library, please refer to the Files and folders

management topic.
- For a brief overview of file types that can be played or viewed on your site by default, please refer to the Supported file types topic.
- To learn about file size supported in media libraries, please refer to the Supported file size topic.
- To learn about names of files uploaded to media libraries and about these files thumbnails, please refer to the File naming conventions topic.

8.32.4.2 Uploading files

There are three ways how files can be uploaded into a media library:

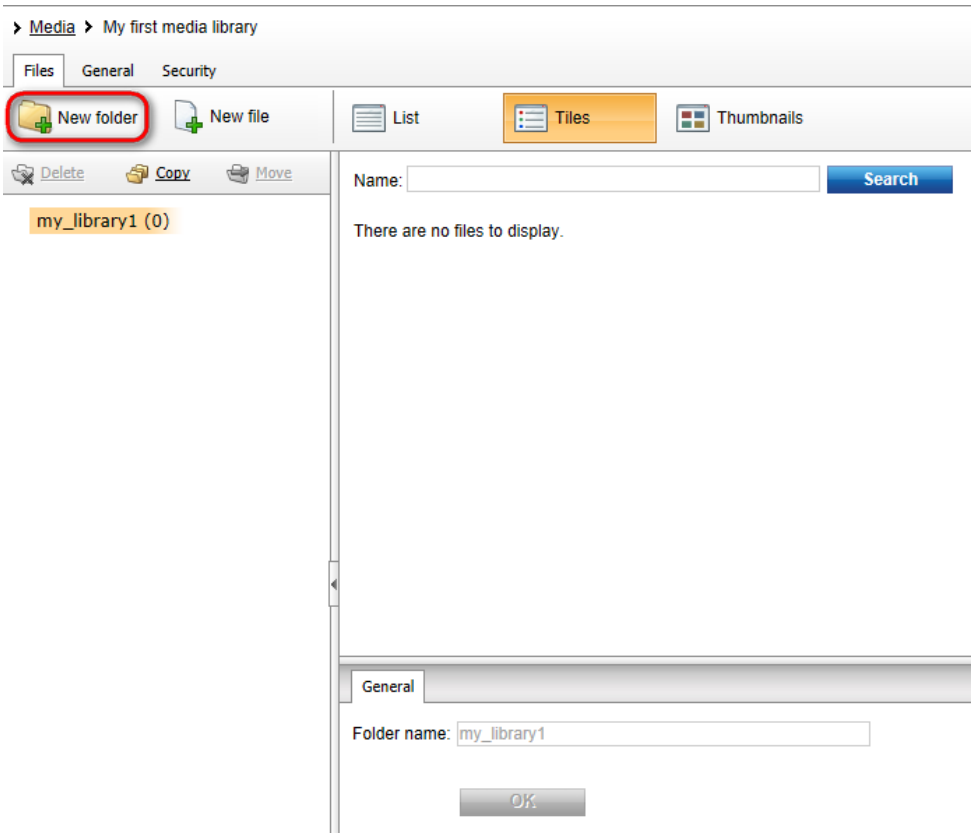
- **By means of the administration interface**
- **On the live site using the Media gallery or Media file uploader web parts**
- **Externally, directly into the file system - e.g. via FTP**

In the following examples, you will learn how to add files into media libraries via the administration interface, on the live site and using FTP.

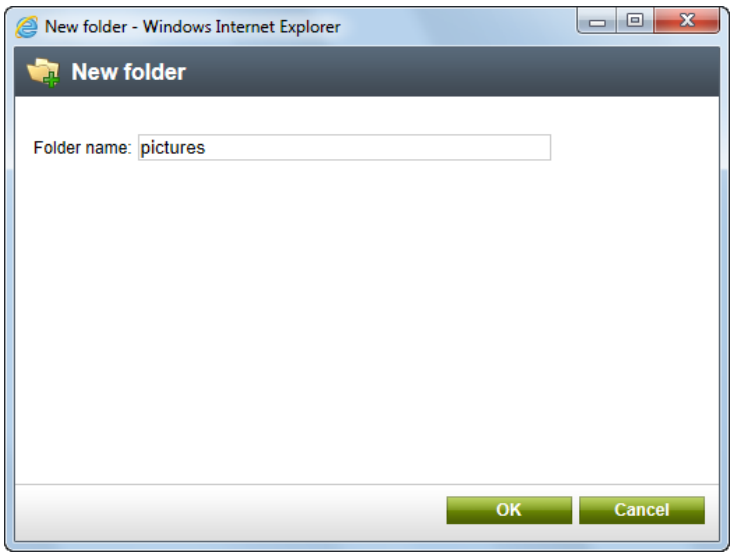
Uploading files via the administration interface

If you choose to **Edit** () a media library in **CMS Desk -> Tools -> Media libraries**, you can manage files in the library on the **Files** tab.

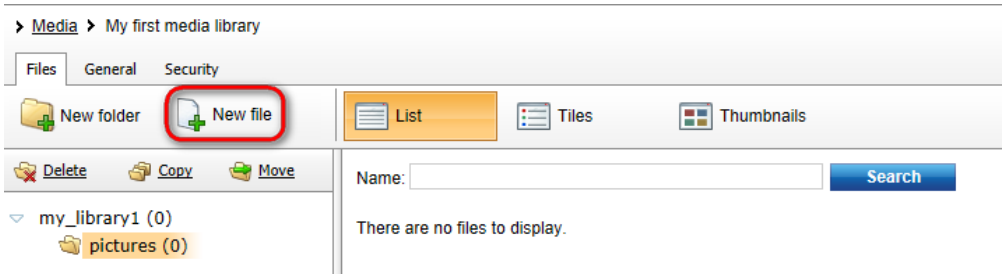
1. It is a good idea to keep your files organized in folders within the media library. To create a new folder, click the  **New folder** button in the top left corner of the page.



2. Enter the name of the folder, e.g. *pictures*, and click **OK**.

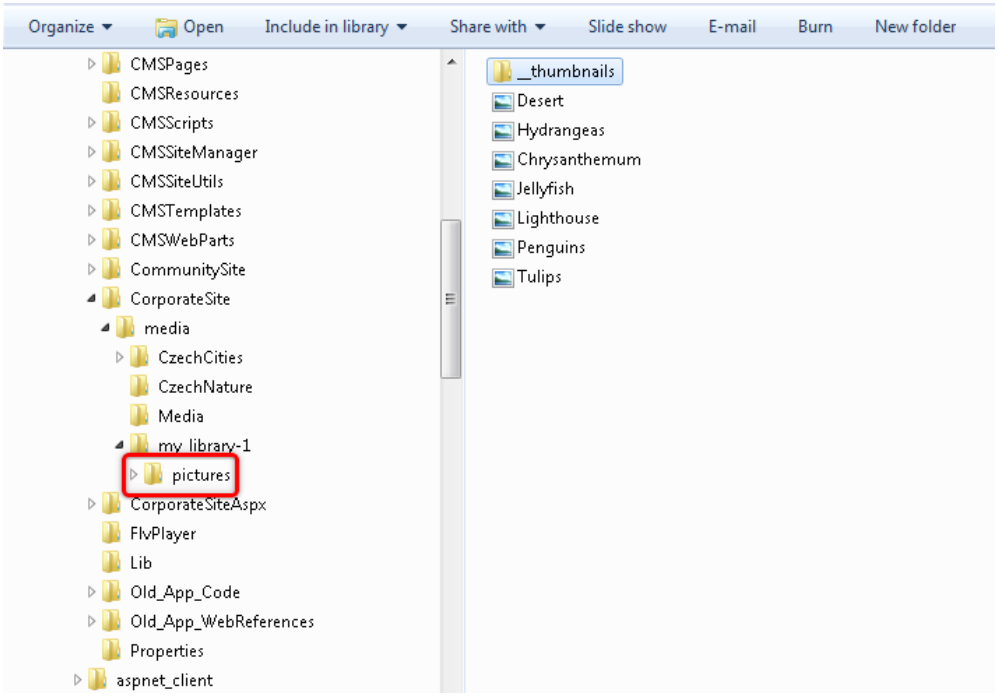


3. The folder is now created. You can upload files into the selected folder by clicking the **New file** button.



4. A dialog appears, letting you upload the file. Just select the right folder and the file to be uploaded, then click the **Open** button.

5. Now if you go to the site folder in your file system, you will find the media library under the **media** folder, as you can see the screenshot below. The location of the folder may be customized as described here.

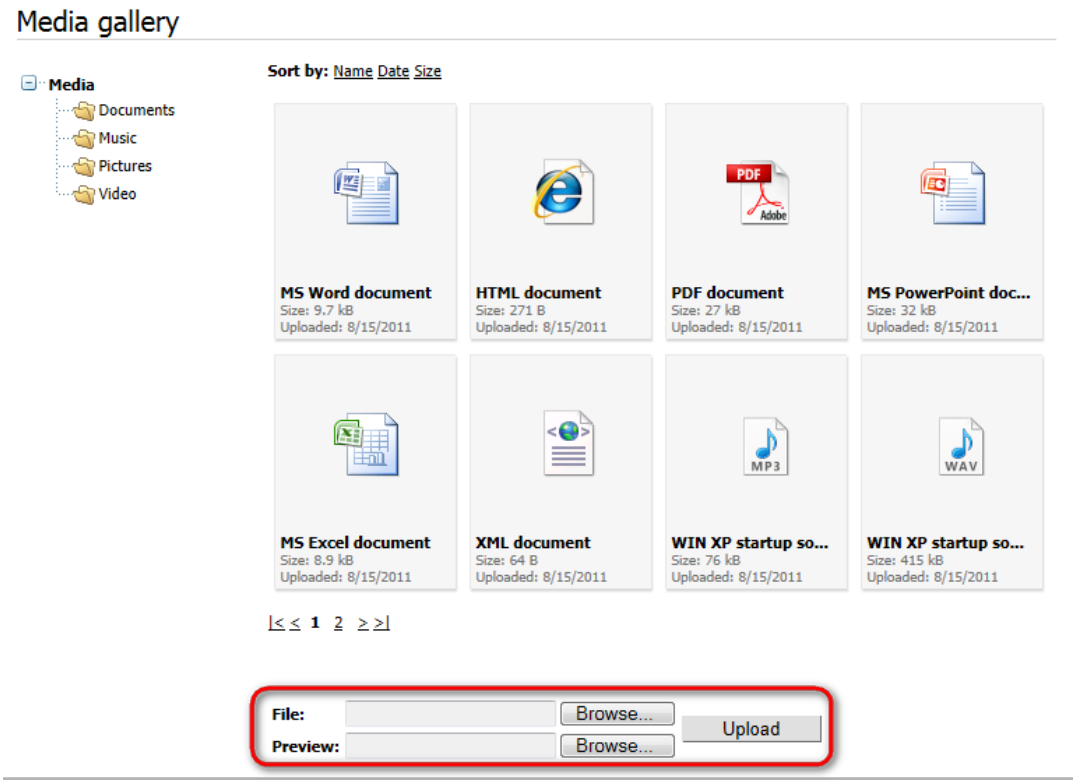


On-site upload via the Media gallery and Media file uploader web parts

The **Media gallery** web part has the following two properties enabling on-site file upload:

- **Allow upload** - enables on-site upload of files.
- **Allow upload thumbnail** - enables on-site upload of file thumbnails.

If these properties are enabled, the controls highlighted in the screenshot below will be displayed in the web part, letting users upload files and thumbnails:



The **Media file uploader** web part is also used for uploading files to your media libraries; it has the following property that enables on-site upload of file thumbnails:

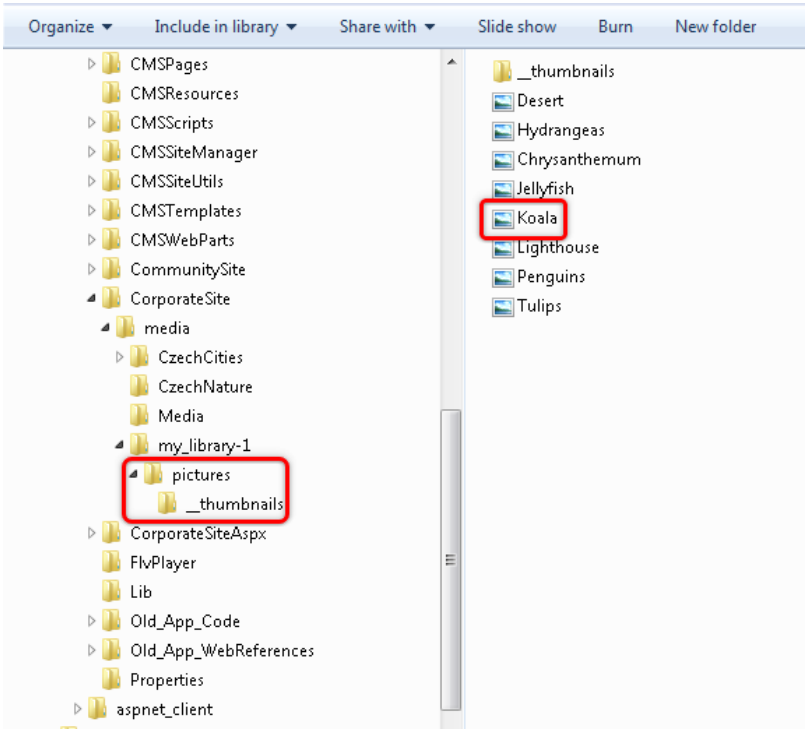
- **Enable upload thumbnail** - enables on-site upload of file thumbnails.

If the property is enabled, the users are allowed to upload both files and their thumbnails. Otherwise, only the control enabling users to browse for and upload files will be available:

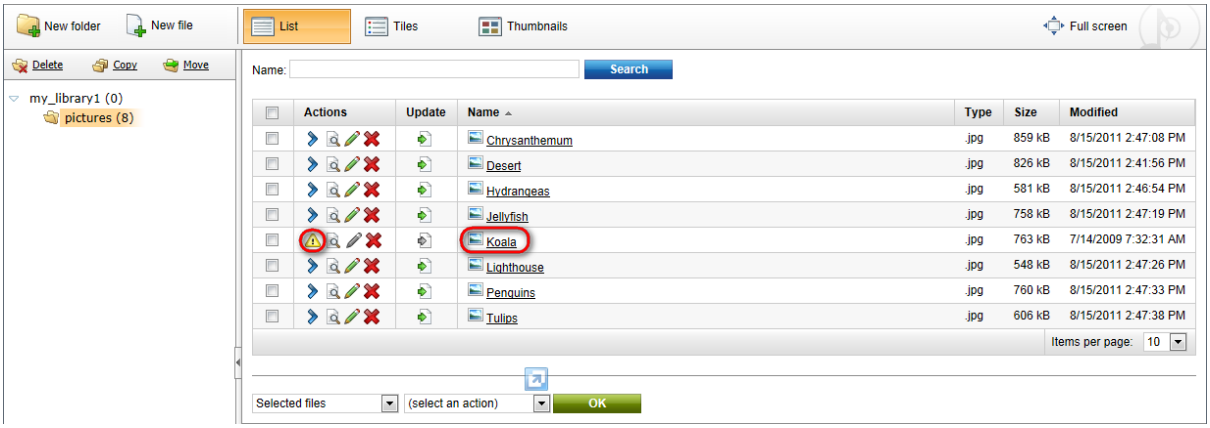
External upload

Files can also be uploaded **externally**, without any use of Kentico CMS administration interface or on-site web parts, e.g. using **FTP**. To do this, you simply need to copy the files into the appropriate folder of the media library.

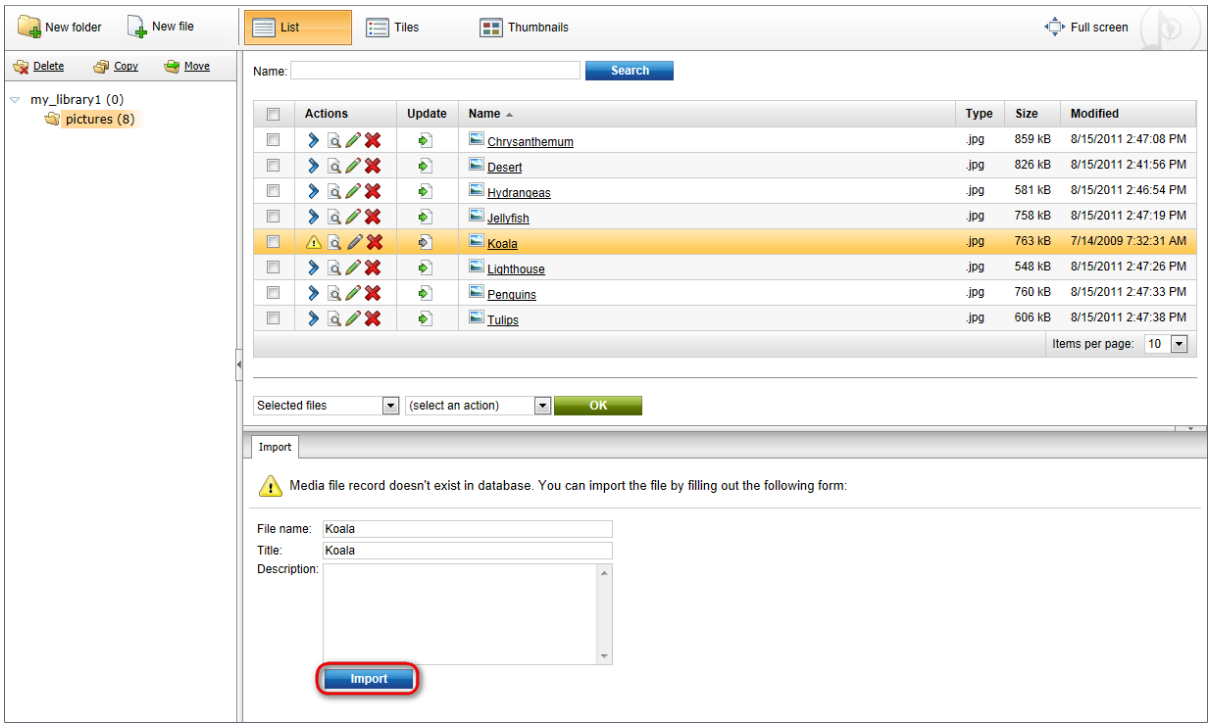
1. For the purposes of this example, try uploading another file into the **pictures** folder by copying it using Windows Explorer or any other file manager.



2. If you go back to the Media library administration interface, you will see the file present in the list. As you can notice, this file is marked with the warning (⚠) icon indicating that the file has not been imported yet. This means that the file **has not yet been registered in the database**. Such files can't be used on the site and have to be registered first. This happens only to externally uploaded files, files uploaded via the administration interface are registered in the database automatically.



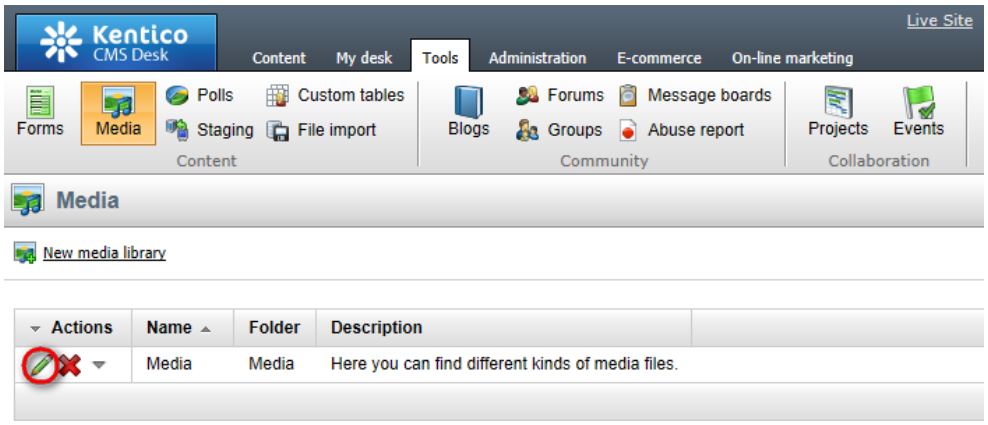
To register the file, you have to click the **Import** button, which creates the file's record in the database. Please note that the files are not physically copied into the database, they remain in the file system only and a record linking to the physical file is created in the database.



Multiple files can also be imported conveniently as a batch. See Batch operations in the **Files and folders management** topic for more details.

8.32.4.3 Files and folders management

To manage files stored in a media library, you need to go to **CMS Desk -> Tools -> Media libraries** and click the **Edit** (✎) icon next to the appropriate media library.



The library editing interface will be displayed on the **Files** tab where the files can be managed.

Folder operations

The following actions can be performed with the folders in the media library:

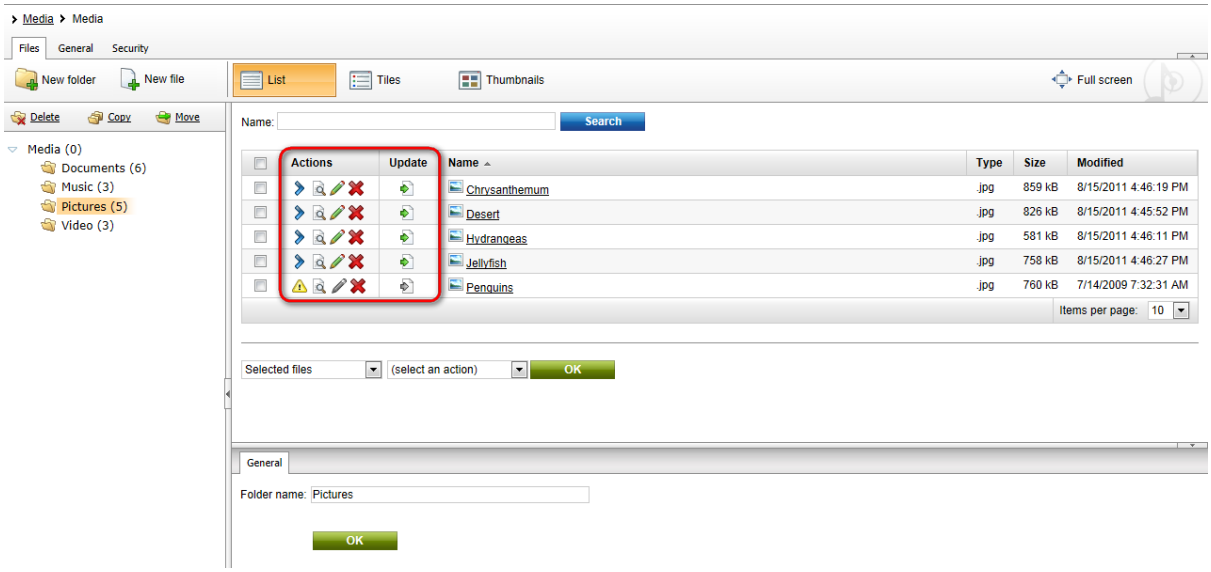
- **New folder** - creates a new folder under the currently selected folder.

- **Delete** - deletes the currently selected folder.
- **Copy** - uploads the folder and its content to another location and keeps the original folder.
- **Move** - uploads the folder into another location and deletes the original folder.

Single file operations

You can perform the following operations by clicking the icons next to particular files:

- **New file** - uploads a file from the local or network file system to the currently selected folder.
- **Select** - selects the file and opens its editing interface in the bottom section.
- **View** - downloads the file and opens it in your default application for the file type.
- **Edit** - if you click the icon next to an image, the image gets opened in the built-in image editor. If the icon is clicked next to a non-image file, the metadata editor is opened.
- **Delete** - deletes the file.
- **Update** - replaces the file with another file. The original file gets deleted and replaced by the new one.
- **Import** - this icon is displayed only with files that have been uploaded to the library folder externally (e.g. via FTP) and have not yet been registered in the database. By clicking the icon, you register the file in the database as described here.



Batch file operations

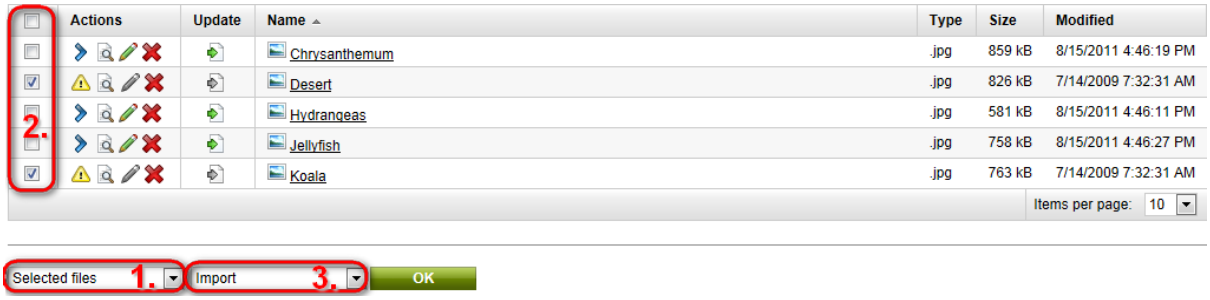
You can also perform the **Copy**, **Move**, **Delete** and **Import** operations for multiple files at once. To perform a batch operation, take the following steps:

1. Use the first drop-down list (**1.** in the screenshot) to select if you want to perform the operation for:

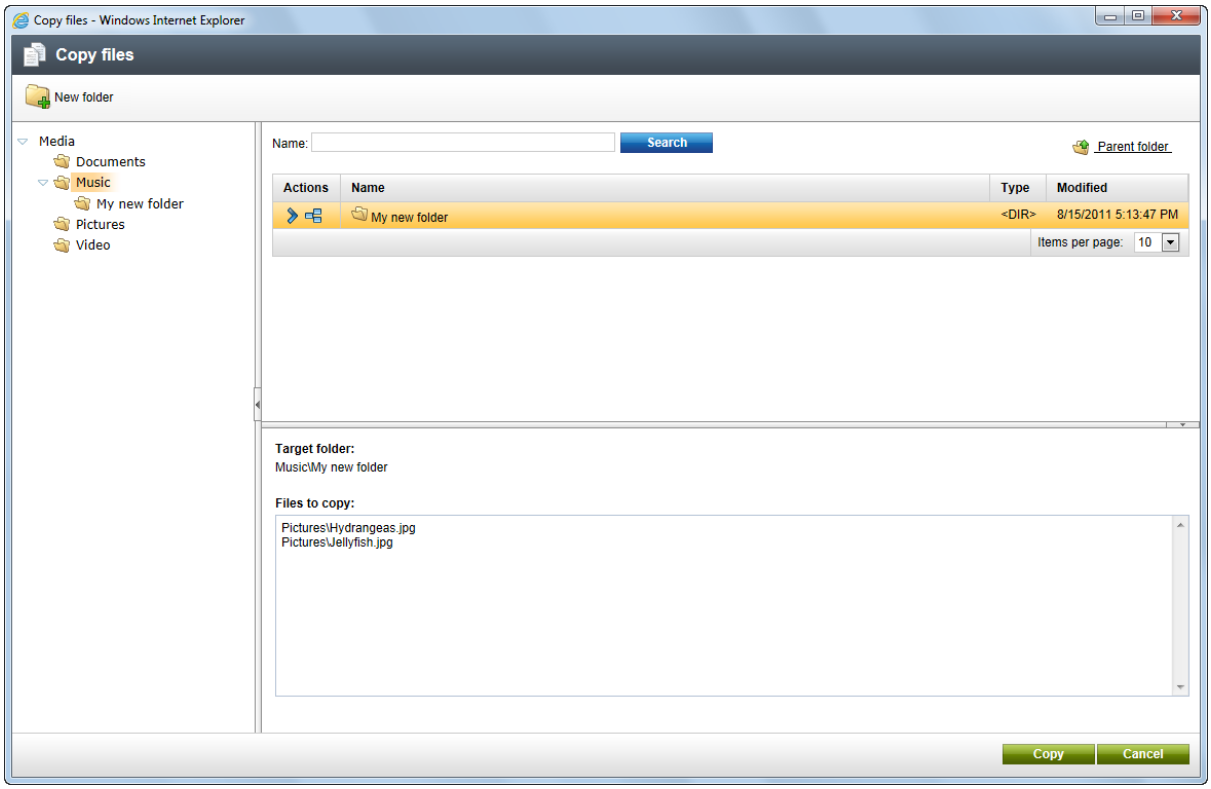
- **All files** - performs the operation for all files in the current folder.
- **Selected files** - performs the operation only for files selected by the check-boxes ( ).

2. If you selected **Selected files** in the previous step, use the check-boxes (**2.** in the screenshot) to select the files with which to perform the operation.

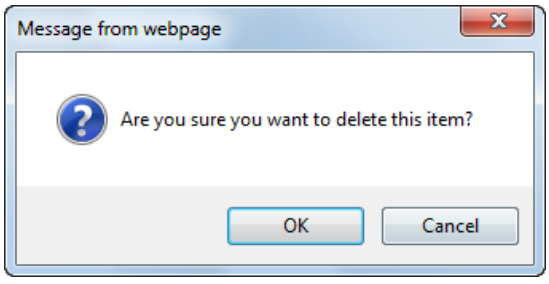
3. Choose the operation to perform using the second drop-down list (**3.** in the screenshot) and click **OK**.



4.a If you are performing the **Copy** or **Move** operation, the following dialog is displayed. Choose the target folder (or create a new one using the **New folder** button if needed) and click the **Copy/ Move** button.



4.b If you are performing the **Delete** operation, you will be prompted to confirm the deletion. Click **OK** to delete the files.

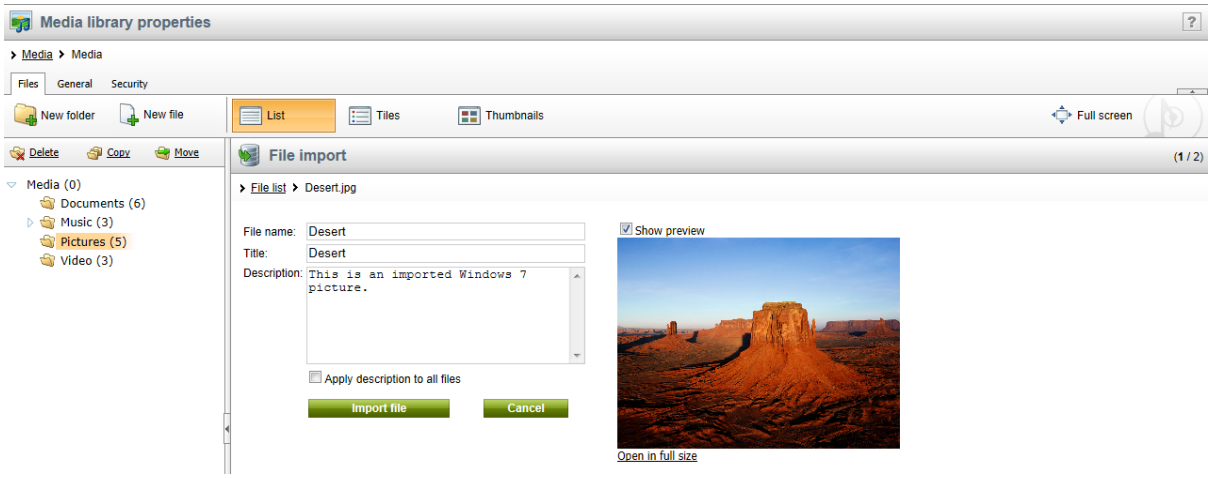


4.c If you are performing a batch **Import** operation, the dialog as in the following screenshot will appear for each imported file. Please specify the following details for each imported file:

- **File name** - the name of the file displayed in the administration interface.
- **Title** - the name of the file displayed on the live site.
- **Description** - the text describing the image. You can leave blank.

Click the **Import file** button to proceed to the following file.

You can also enable the **Apply description to all files** option, which imports all files in one click on the **Import all files** button. All files will have the entered **Description** and the **File name** and **Title** fields will contain the names of the physical files without extension.



Important!

Because of ASP.NET architecture, **site restart** will occur when:

- a media library is deleted
- a group containing a media library is deleted
- one of the following actions is performed when editing a library in **CMS Desk -> Tools -> Media libaries -> Files** or on the live site:
  - folder is deleted
  - folder is renamed
  - folder is moved
  - large number of files is deleted (100 by default, this can be set in the *<system.web>* section of your web.config by the following key: *<compilation debug="true" numRecompilesBeforeAppRestart="100">*)

Because of this, it is highly recommended to allow performing of these actions only to system administrators or to the least possible number of users. The recommended practice is for the site administrators to pre-define the folder structure of the libraries when they are created and not to allow users to further modify it.

8.32.4.4 Supported file types

The following file types are supported and can be played or displayed on your site by default:

- **Images** - bmp, gif, png, wmf, jpg, jpeg, tiff, tif.
- **Audio** - wav, wma, mp2, mp3, mid, midi, mpga.
- **Video** - avi, mp4, mpg, mpeg, wmv, qt, mov, rm.
- **Flash** - swf.

Custom file types can also be defined, as described in this chapter.

Please note

All other file types will be recognized as **documents**, which means that they can't be played as videos, displayed as pictures, etc., but they can still be stored in the library and downloaded by users on the live site.

8.32.4.5 Supported file size

Media libraries in Kentico CMS support the upload of large files (audio files, video files etc.) However, if you would like to enable this functionality for your media libraries, you need **to increase the maximal uploaded file size**, as described in this topic.

8.32.4.6 File naming conventions

When a file is being uploaded to a media library, users can upload the file itself and its thumbnail. The **thumbnail** will be displayed in the file list in the **Media gallery** web part. Thumbnails are typically used if you want to upload some non-image file and attach an image which will be displayed as the file's thumbnail in the file list. For image files, the thumbnail can but doesn't have to be done, as the image itself will be used for this purpose.

When the thumbnail image or the image itself is larger than the required thumbnail image size, its resized copy will be created with the required size. Both thumbnails and resized copies are stored in a **hidden folder** in the root of the media library folder. The name of the hidden folder can be set in **Site Manager -> Settings -> Content -> Media -> Media file hidden folder**.

Format of media library files on the disk

- **File** - <file name>.<file extension>.
- **Resized file** - <hidden folder>/<file name>_<file extension>_<width>_<height>.<extension>.

- **Thumbnail** - <hidden folder>/<file name>_<file extension><thumbnail suffix>.<thumbnail extension>.
- **Resized thumbnail** - <hidden folder>/<file name>_<file extension><thumbnail suffix>_<width>_<height>.<thumbnail extension>.

Example

If you upload the following files:

- **Uploaded file** - *MyImage.jpg*.
- **Uploaded thumbnail** - *MyPhoto.bmp*.

with the following settings defined in Site Manager -> Settings -> Content -> Media:

- **Media file hidden folder** - *__thumbnails*.
- **Media file thumbnail suffix** - *_preview*.

the uploaded files will be stored in the following locations in the media library folder, with the following names:

- **File** - *MyImage.jpg*.
- **Resized file** - *__thumbnails/MyImage_jpg_100_200.jpg*.

- **Thumbnail** - *__thumbnails/MyImage_jpg_preview.bmp*.
- **Resized thumbnail** - *__thumbnails/MyImage_jpg_preview_20_30.bmp*.

8.32.5 Displaying files from media library

8.32.5.1 Overview

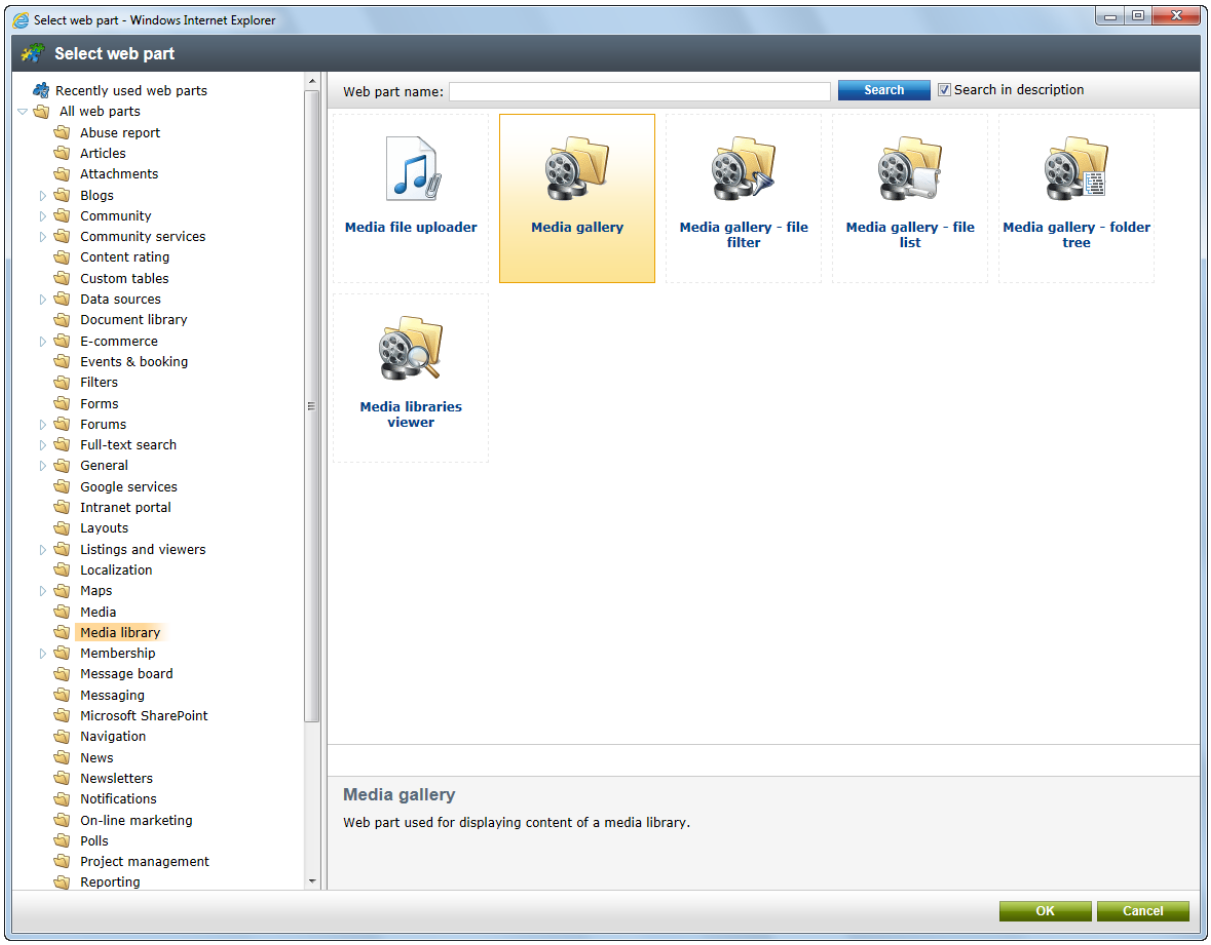
In this subchapter you will learn how to display files from a media library.

- If you would like to learn how to display media gallery content on your site, please follow the Using Media gallery web part topic.
- If you would like to learn about integration of media library files into text using the built-in WYSIWYG editor, please refer to the Using WYSIWYG editor topic.
- If you would like to learn how to use files from various sources (from Document attachments, Content and Media libraries) in your document types, please refer to the Using the Media selection control topic.

8.32.5.2 Using Media gallery web part

The **Media gallery** web part can be used to display content of media libraries on the live site.

1. If you need to use the **Media gallery** web part, navigate to **Content -> Edit**. Switch to the **Design** tab and click any of the available **Add web part** () icons on the page you have selected from the content tree of the current site. Now select **Media library -> Media gallery** and click **OK**.

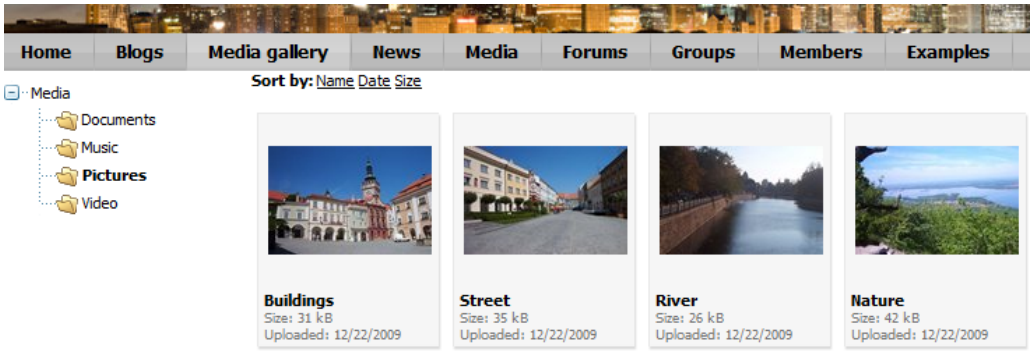


2. In the web part properties window, you only have to choose the media library that you want to display:

- **Media library** - *Media*.

Click **OK**.

3. If you now switch to the live site, and select **Pictures** for instance, you should see the web part displaying content of the selected media library.



8.32.5.3 Using WYSIWYG editor

If you are editing a text using the WYSIWYG editor, you can easily insert an image, video, flash etc. from your media library into the text via the **Insert image or media** dialog (please refer to the Insert image or media chapter for more details) or you can insert a link to any file from your media library via the **Insert link** dialog (please refer to the Insert link chapter for more details).

8.32.5.4 Using the Media selection control

The **Media selection** form control can be used to enable users to select files on the **Form** tab of a document. It allows to use any file from any source (from Document attachments, Content and Media libraries) in your documents. To follow an example which demonstrates how this can be achieved, please refer to the topic of the same name in the **File management** chapter in the **Content management** section of this guide.

8.32.6 Settings

8.32.6.1 Overview

In this subchapter you will learn how to adjust general media library settings.

- If you would like to adjust the settings of the Media libraries module, please refer to the Media library settings topic.
- If you would like to learn how to customize the location of all libraries of a particular site, please refer to the Configuring custom storage for media library topic.
- If you would like to read a more detailed explanation of how to enable any media types to be recognized by the system, please refer to the Configuring custom file types topic.
- If you would like to learn how to configure maximal uploaded file size, please refer to the Configuring maximal uploaded file size topic.

8.32.6.2 Media library settings

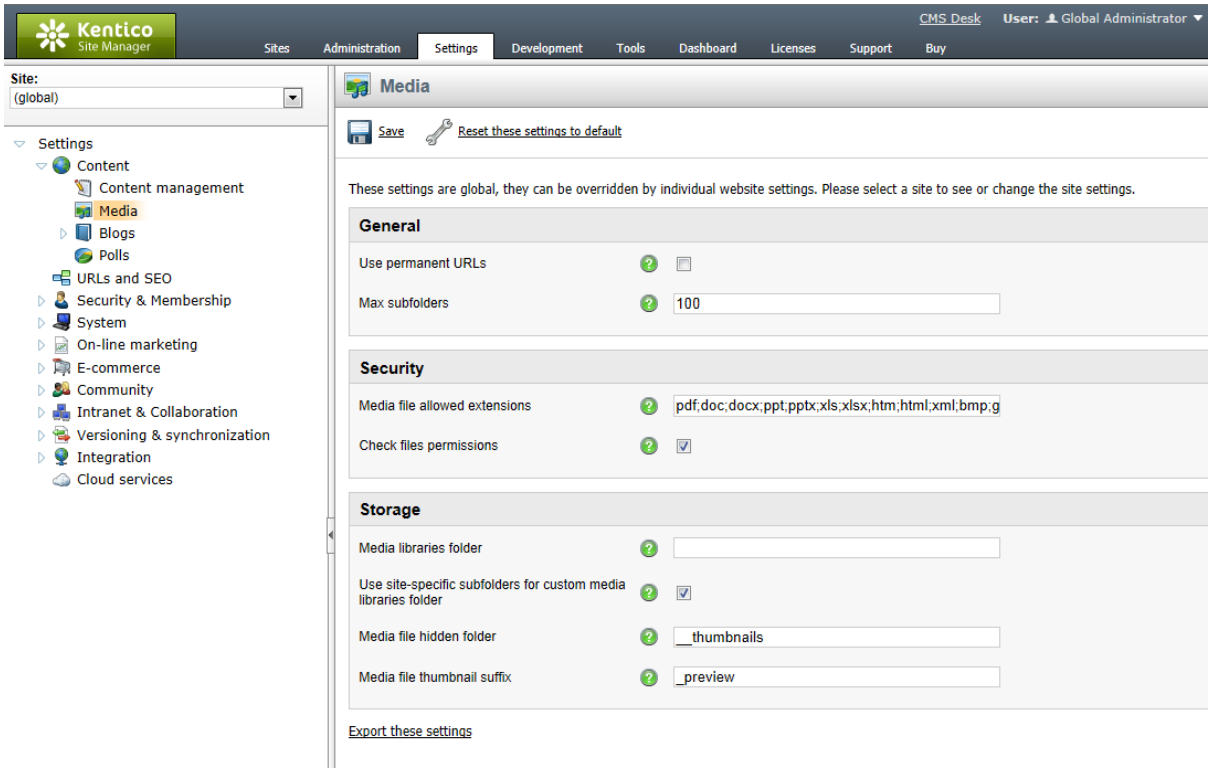
Settings related to the Media libraries module are located in **Site Manager -> Settings -> Content -> Media**. The following settings are related to the module:

General	
Use permanent URLs	If true, URLs of media library files will be generated in permanent format (*~/getmedia/<file guid>/<file name>.<file extension>.<files friendly URL extension>*), otherwise a direct path to the file will be used (e.g.: *~/MySite/Media/MyLibrary/MyImage.jpg*).
Max subfolders	The maximum number of folders that can be displayed under an expanded folder node in the tree view.
Security	
Media file allowed extensions	Extensions of files which can be uploaded to media libraries; should be divided by semicolons.
Check files permissions	Indicates if the See media library content permission is checked when retrieving media files using permanent URLs.
Storage	

Custom media libraries folder	The folder where media library files are stored. If no value is entered, the default location is **~/<site code name>/media**. If you would like to learn how to configure custom storage for your media library, please refer to this topic.
Use site-specific subfolders for custom media libraries folder	This setting is applied only when a *Custom media libraries folder* is configured. If enabled, media library files will be stored in a sub-folder named using the site code name under the custom files folder, i.e. *<custom media libraries folder>/<site code name>*. It is useful for better orientation in files when multiple sites are running in the system.
Media file hidden folder	The name of the folder where thumbnails of media files will be stored. This folder is hidden in the file system by default and thumbnails are generated within it.
Media file thumbnail suffix	The suffix added to thumbnail files. Thumbnail files names are in the following format: *<file name>_<file extension><thumbnail suffix>.<thumbnail extension>*

Please note

If the **Custom media libraries folder** settings are inherited from global settings and the **Use site-specific subfolders for custom media libraries folder** property is set to TRUE, only the folder of the site you are about to delete will be removed from the custom folder.



8.32.6.3 Configuring custom storage for media library

The default location of all libraries of a particular site is **~/<sitename>/media**. You can customize the location by entering a path in **Site Manager -> Settings -> Content -> Media -> Custom media libraries folder**.

You can enter the folder location in the following formats:

- **physical path** - e.g. *c:\Libraries.*
- **virtual path** - e.g. *~/Libraries.*
- **UNC path** - e.g. \\<*server name*>*Libraries.*

If you enable the **Use site-specific subfolders for custom media libraries folder** option, media library files will be stored in a sub-folder named as the site code name under the custom files folder, i.e. <*custom media libraries folder*>/<*site code name*>.

Please note

In case that you are running the system on a **web farm** and have **the same UNC root defined** on all servers, it is necessary to add the following key into your *web.config* in order for the files stored in the libraries not to be transferred when synchronizing the web farm content:

```
<add key="CMSWebFarmSynchronizeMediaFiles" value="false"/>
```

8.32.6.4 Configuring custom file types

You can allow custom media types by a simple modification to the site's **web.config** and **MediaControl.ascx** inline controls. Any media types can be enabled to be recognized by the system in case that you have the right player for the file type.

The following example shows how to enable .flv videos in the system.

1. Add the following keys into the **appSettings** section of your **web.config** file. Add your custom extensions for the particular document types here for the files to be recognized as image/audio/video by the system, as you did with the highlighted **flv** extension.

```
<add key="CMSImageExtensions" value="bmp;gif;ico;png;wmf;jpg;jpeg;tiff;tif" />
<add key="CMSAudioExtensions" value="wav;wma;mp2;mp3;mid;midi;mpga" />
<add key="CMSVideoExtensions" value="avi;mp4;mpg;mpeg;wmv;qt;mov;rm;flv" />
```

2. Add the **flv** extension to the enumeration in **Site Manager -> Settings -> Content -> Media -> Media file allowed extensions**, which will allow upload of these files into media libraries, and click the **Save** button.

You may also want to add the extension to the **Site Manager -> Settings -> System -> Files -> Upload extensions** enumeration, which will allow upload of these files as CMS.File documents and document attachments, and click the **Save** button.

3. Download a player for your media type. In this example, you will use the JW FLV Player available here: http://www.longtailvideo.com/players/jw-flv-player/.

4. Extract the **swfobject.js** and **player.swf** files and put them in a folder under your website folder, e.g. **~/FlvPlayer**. This path will be used in the **CreateFlvObject()**, which you will create in step 6.

5. Open the **~/CMSInlineControls/MediaControl.ascx.cs** inline control in Visual Studio and replace the **Page_PreRender** method with the following code. It is the original code of the method with the first condition added. The added code ensures that when the file extension is **.flv**, the **CreateFlvObject()** method is called.

[C#]

```
protected void Page_PreRender(object sender, EventArgs e)
{
    if ((this.Type != null) && (this.Type.TrimStart('.').ToLower() == "flv"))
    {
        CreateFlvObject();
    }
    else if (MediaHelper.IsFlash(this.Type))
    {
        CreateFlash();
    }
    else if (ImageHelper.IsImage(this.Type))
    {
        CreateImage();
    }
    else
    {
        CreateMedia();
    }
}
```

6. Now you need to add the **CreateFlvObject()** private method to the control. In this example, it looks like the code sample below.

The method creates a new DIV tag for the FLV player, with ID generated in the divID variable. It also handles the player URL in a special way - the URL must be absolute and must end with *flv*. Support for the Autoplay and Loop properties of the player is also ensured.

[C#]

```
private void CreateFlvObject()
{
    string playerID = Guid.NewGuid().ToString("N");
    string flvUrl = URLHelper.GetAbsoluteUrl(this.Url)
                .Replace("?", "%3F")
```

```
                .Replace("=", "%3D")
                .Replace("&", "%26");

    string flashVars = "file=" + flvUrl + "&provider=video&";
    flashVars += "controlbar=" + (AVControls ? "bottom" : "none") + "&";
    flashVars += "autostart=" + (AutoPlay ? "true" : "false") + "&";
    flashVars += "repeat=" + (Loop ? "always" : "none");

    string player = "";
    player += "<object id=\"" + playerID + "\"";
    player += "classid=\"clsid:D27CDB6E-AE6D-11cf-96B8-444553540000\"";
    player += "codebase=\"http://download.macromedia.com/pub/";
    player += "shockwave/cabs/flash/swflash.cab#version=9.0.115\"";
    player += "width=\"" + Width + "\" height=\"" + Height + "\">";
    player += "<param name=bgcolor value=\"#FFFFFF\">";
    player += "<param name=movie value=\"" + ResolveUrl("~/FlvPlayer/player.swf")
+ "\">";
    player += "<param name=allowfullscreen value=\"true\">";
    player += "<param name=allowscriptaccess value=\"always\">";
    player += "<param name=\"flashvars\" value=\"" + flashVars + "\">";
    player += "<embed name=\"" + playerID + "\" ";
    player += "type=\"application/x-shockwave-flash\" ";
    player += "pluginspage=\"http://www.macromedia.com/go/getflashplayer\" ";
    player += "width=\"" + Width + "\" height=\"" + Height + "\" ";
    player += "bgcolor=\"#FFFFFF\" ";
    player += "src=\"" + ResolveUrl("~/FlvPlayer/player.swf") + "\" ";
    player += "allowfullscreen=\"true\" ";
    player += "allowscriptaccess=\"always\" ";
    player += "flashvars=\"" + flashVars + "\">";
    player += "</embed>";
    player += "</object>";

    this.ltlMedia.Text = player;
}
```

7. To make this work also on the live site using media library transformations, you now need to open the **~/CMSModules/MediaLibrary/Controls/LiveControls/MediaFilePreview.ascx.cs** file and replace the following line of code:

```
else if (CMS.GlobalHelper.MediaHelper.IsVideo(mfi.FileExtension))
```

with this piece of code:

```
else if (mfi.FileExtension.TrimStart('.').ToLower() == "flv") {
    this.ltlOutput.Text = CreateFlvObject(url);
}
else if (CMS.GlobalHelper.MediaHelper.IsVideo(mfi.FileExtension))
```

8. Similarly, you will need to add the **CreateFlvObject(string url)** private method to the control. In this example, it looks like the code sample below.

The method creates a new DIV tag for the FLV player, with ID generated in the divID variable. It also handles the player URL in a special way - the URL must be absolute and must end with *flv*. Please note that the Autoplay and Loop properties of the player are defined firmly in the code.

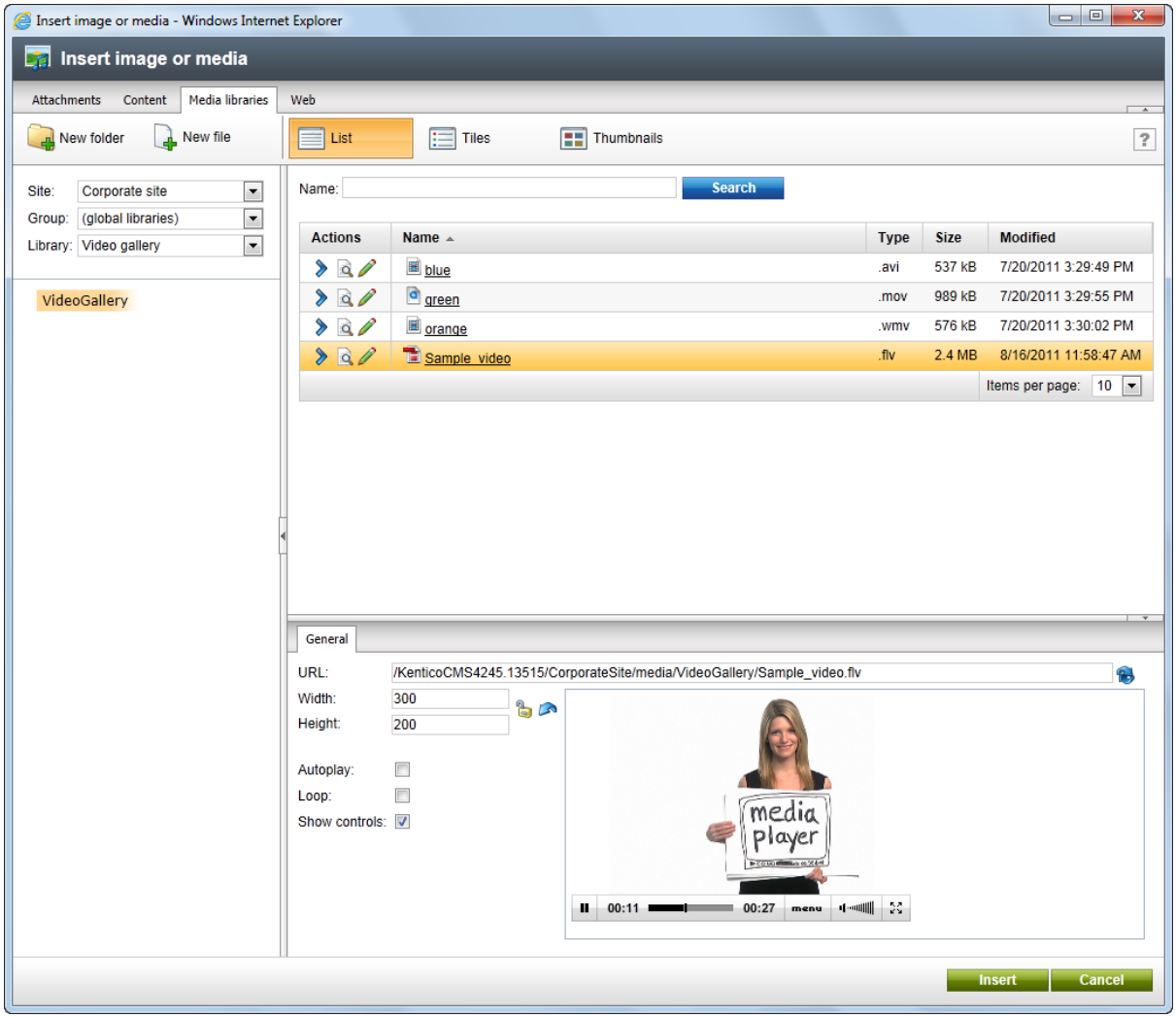
```
private string CreateFlvObject(string url)
{
    string playerID = Guid.NewGuid().ToString("N");
    string flvUrl = URLHelper.GetAbsoluteUrl(url)
                .Replace("?", "%3F")
                .Replace("=", "%3D")
                .Replace("&", "%26");

    bool avControls = true;
    bool autoPlay = false;
    bool loop = false;

    string flashVars = "file=" + flvUrl + "&provider=video&";
    flashVars += "controlbar=" + (avControls ? "bottom" : "none") + "&";
    flashVars += "autostart=" + (autoPlay ? "true" : "false") + "&";
    flashVars += "repeat=" + (loop ? "always" : "none");

    string player = "";
    player += "<object id=\"" + playerID + "\"";
    player += "classid=\"clsid:D27CDB6E-AE6D-11cf-96B8-444553540000\"";
    player += "codebase=\"http://download.macromedia.com/pub/";
    player += "shockwave/cabs/flash/swflash.cab#version=9.0.115\"";
    player += "width=\"" + Width + "\" height=\"" + Height + "\">";
    player += "<param name=bgcolor value=\"#FFFFFF\">";
    player += "<param name=movie value=\"" + ResolveUrl("~/FlvPlayer/player.swf")
+ "\">";
    player += "<param name=allowfullscreen value=\"true\">";
    player += "<param name=allowscriptaccess value=\"always\">";
    player += "<param name=\"flashvars\" value=\"" + flashVars + "\">";
    player += "<embed name=\"" + playerID + "\" ";
    player += "type=\"application/x-shockwave-flash\" ";
    player += "pluginspage=\"http://www.macromedia.com/go/getflashplayer\" ";
    player += "width=\"" + Width + "\" height=\"" + Height + "\" ";
    player += "bgcolor=\"#FFFFFF\" ";
    player += "src=\"" + ResolveUrl("~/FlvPlayer/player.swf") + "\" ";
    player += "allowfullscreen=\"true\" ";
    player += "allowscriptaccess=\"always\" ";
    player += "flashvars=\"" + flashVars + "\">";
    player += "</embed>";
    player += "</object>";

    return player;
}
```

If you need to change the behavior of the player, please modify these lines of code of the added method.

```
bool avControls = true;
```

```
bool autoPlay = false;
bool loop = false;
```

9. Now .flv files are recognized as videos by the system. The custom player is used for the files on the live site and also in the **Insert image or media** dialog, as you can see in the screenshot below.



8.32.6.5 Configuring maximum uploaded file size

The default maximum uploaded file size setting on IIS6/7 is 30MB. To enable uploads of larger files, which is essential for the functionality that the media libraries were designed for, you need to add the following keys to your **web.config** file:

IIS 6

All you need to do is to set the value of the following property, which is already present in your **web.config**, to the desired value, while the value is entered in **kiloBytes**:

```
<httpRuntime maxRequestLength="2000000" />
```

IIS 7

You need to do the same setting as described above for IIS 6. Then, you need to add the following highlighted code to the end of your **web.config**. The value of the **maxAllowedContentLength** property is entered in **Bytes**:

```
...

<system.webServer>
<security>
    <requestFiltering>
      <requestLimits maxAllowedContentLength="2147483648"/>
    </requestFiltering>
</security>
  <validation validateIntegratedModeConfiguration="false" />
  <modules>
    <add name="ScriptModule" preCondition="integratedMode"
type="System.Web.Handlers.ScriptModule, System.Web.Extensions, Version=3.5.0.0,
Culture=neutral, PublicKeyToken=31bf3856ad364e35" />
    <add name="XHtmlModule" type="CMS.CMSOutputFilter.OutputFilterModule,
CMS.OutputFilter" />
  </modules>
  <handlers>
    <remove name="WebServiceHandlerFactory-Integrated" />
    <add name="ScriptHandlerFactory" verb="*" path="*.asmx"
preCondition="integratedMode"
type="System.Web.Script.Services.ScriptHandlerFactory, System.Web.Extensions,
Version=3.5.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" />
    <add name="ScriptHandlerFactoryAppServices" verb="*" path="*_AppService.axd"
preCondition="integratedMode"
type="System.Web.Script.Services.ScriptHandlerFactory, System.Web.Extensions,
Version=3.5.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" />
    <add name="ScriptResource" preCondition="integratedMode" verb="GET,HEAD"
path="ScriptResource.axd" type="System.Web.Handlers.ScriptResourceHandler,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35" />
  </handlers>
</system.webServer>
</configuration>
```

8.32.7 Security

8.32.7.1 Overview

In this subchapter you will learn how to adjust media library security settings.

- If you would like to learn how to deal with secured and non-secured libraries on your site, please refer to the Secured vs. Non-secured libraries topic.
- If you would like to learn how to set media library permissions, please refer to the Media library

permissions topic.

8.32.7.2 Media library permissions

Media library module permissions

The following permissions can be set in **Site Manager -> Administration -> Permissions**, when you choose the **Modules -> Media libraries** permissions matrix:

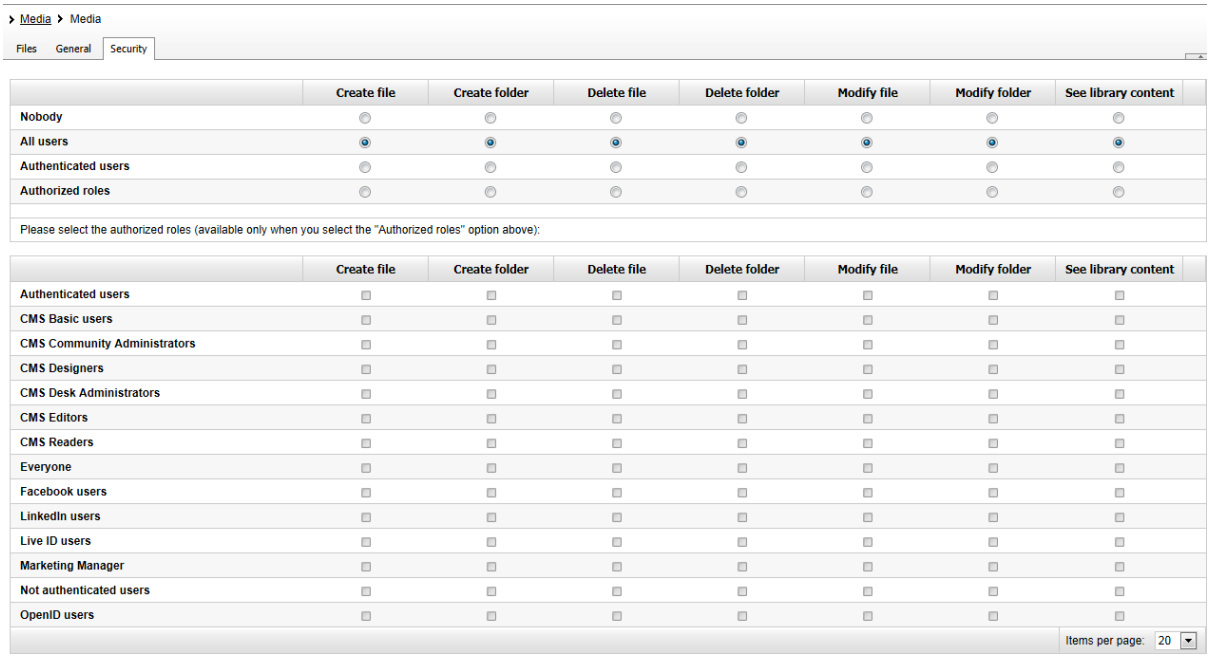
- **Manage** - allows to create, delete and edit media libraries and manage media library content via the administration interface.
- **Read** - allows seeing media library content and its properties when editing a media library, but does not allow to make any changes to it.
- **Destroy** - allows users to destroy the media library version history.

Role	Read	Manage	Destroy
Authenticated users	☐	☐	☐
CMS Basic users	☐	☐	☐
CMS Community administrators	☑	☑	☐
CMS Designers	☐	☐	☐
CMS Desk Administrators	☑	☑	☐
CMS Editors	☑	☐	☐
CMS Readers	☑	☐	☐
Everyone	☐	☐	☐
Facebook users	☐	☐	☐
Gold Partners	☐	☐	☐
LinkedIn users	☐	☐	☐
Live ID users	☐	☐	☐
Marketing Manager	☑	☐	☐
Not authenticated users	☐	☐	☐
OpenID users	☐	☐	☐
Silver Partners	☐	☐	☐

Media library permissions

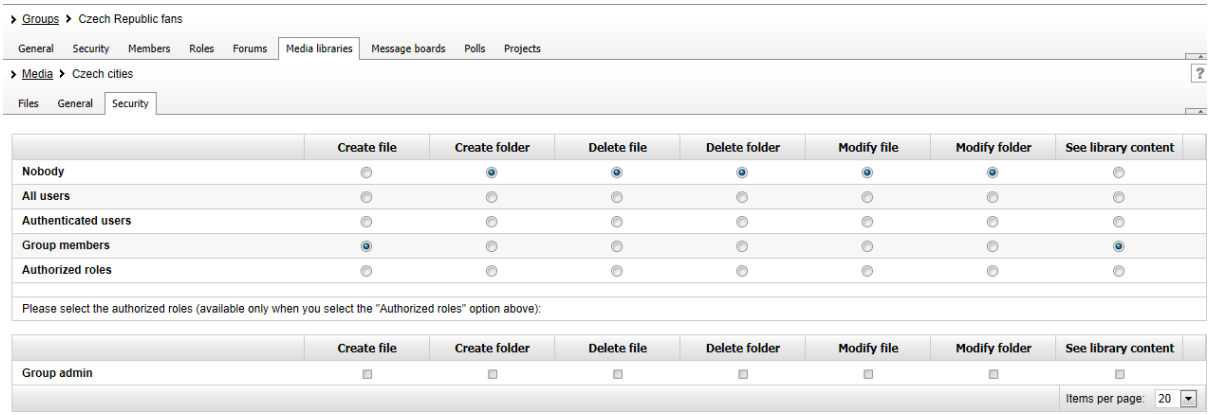
On the **Security** tab of each media library, you can assign permissions for particular actions. This can be useful if settings for the Media library module permissions are not sufficient for your needs and you want to restrict users from performing certain actions in certain media libraries. If this is the case, permissions for particular actions can be assigned to:

- **Nobody** - nobody can perform the action.
- **All users** - all users including anonymous site visitors can perform the action.
- **Authenticated users** - only signed-in site members can perform the action.
- **Authorized roles** - only members of roles selected by the check-boxes below can perform the action.



When editing media library of some group, permissions for particular actions can be assigned to:

- **Nobody** - nobody can perform the action.
- **All users** - all users including anonymous site visitors can perform the action.
- **Authenticated users** - only signed-in site members can perform the action.
- **Group members** - only members of the group can perform the action.
- **Authorized roles** - only members of group roles selected by the check-boxes below can perform the action.



The following table shows which permissions need to be assigned to allow users to perform particular actions. Global administrators can perform all of these actions for all general and group media libraries on the site. Group administrators can perform all of these actions for group media libraries of groups where they are group administrators.

8.32.7.3 Secured vs. Non-secured libraries

Media libraries on your site can be secured or non-secured. To ensure the required functionality, several settings need to be done as described below.

Please note

By default, files in media libraries are NOT secured and can be accessed directly by anybody who knows the exact link to the file. If you want to prevent this behavior, please set up your media library as a secured one.

Secured libraries

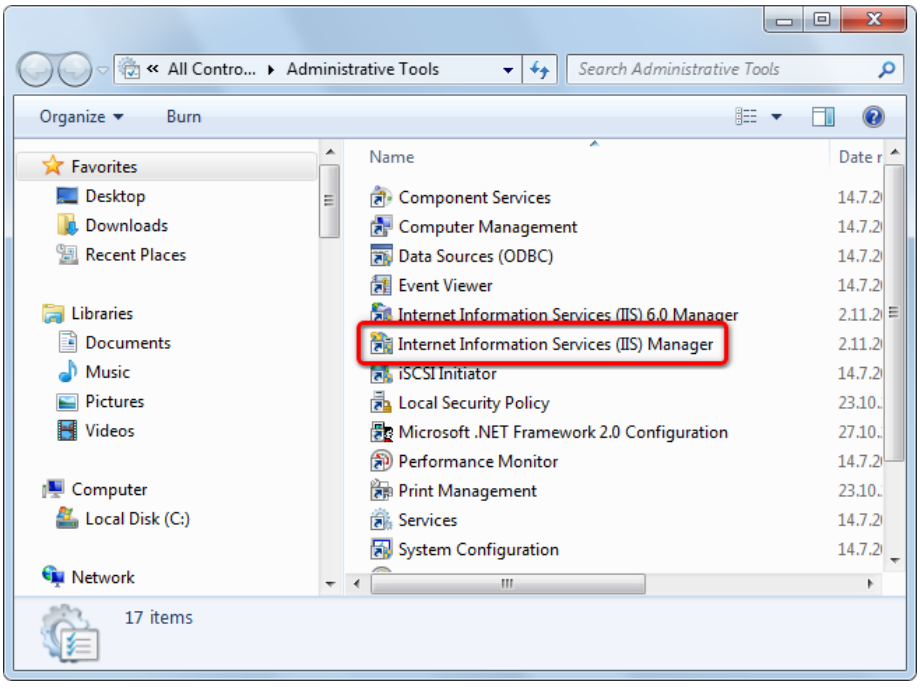
Secured media libraries allow viewing of their content only to members of authorized roles or only to authenticated users, based on the settings made on the library's **Security** tab. Secured libraries are also slower than the non-secured ones as permission checking involves some processing overhead.

To set up a media library to behave as a secured library, you have to take the following steps:

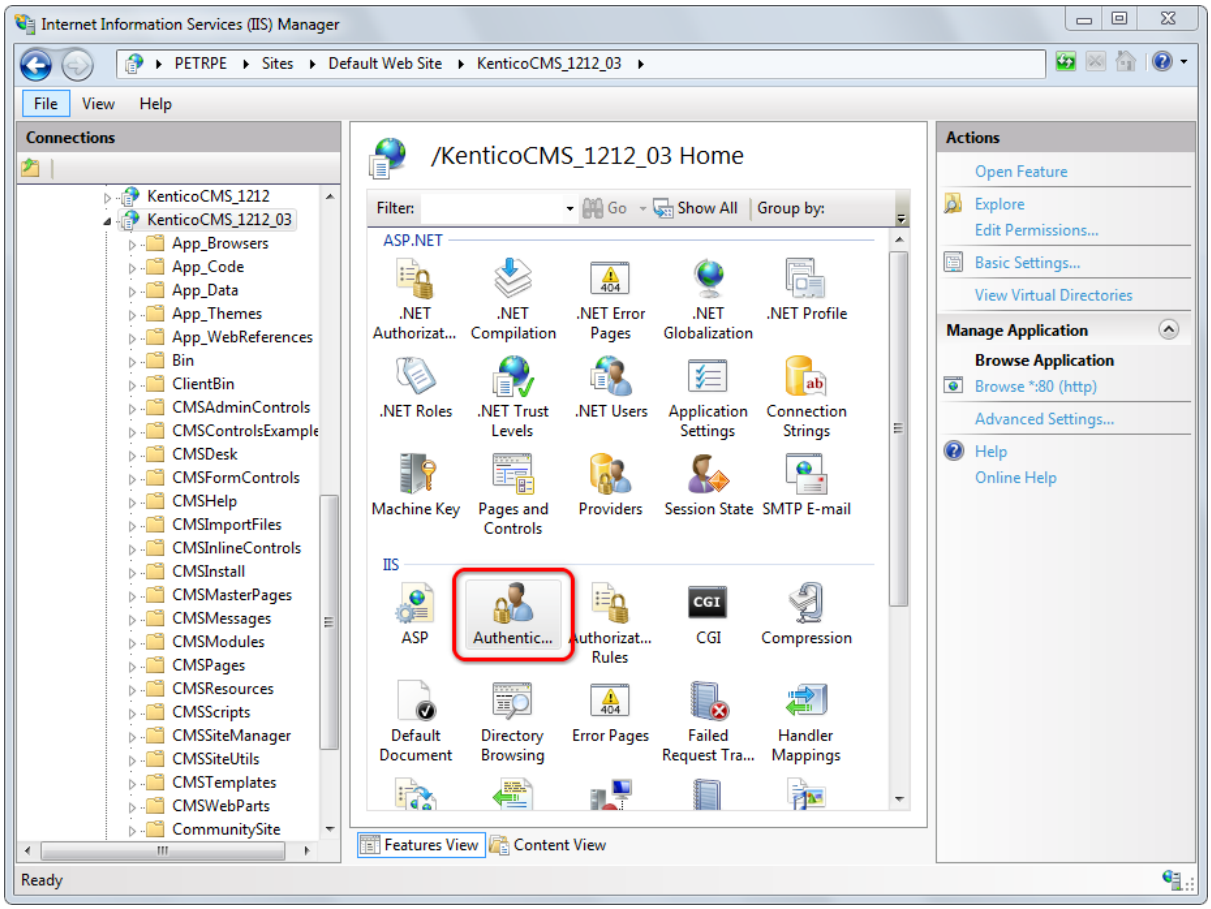
1. IIS setup

You have to **set up your IIS** so that files can't be downloaded directly from the library by typing the link to the file like *<site url>/media/file.jpg* into the browser.

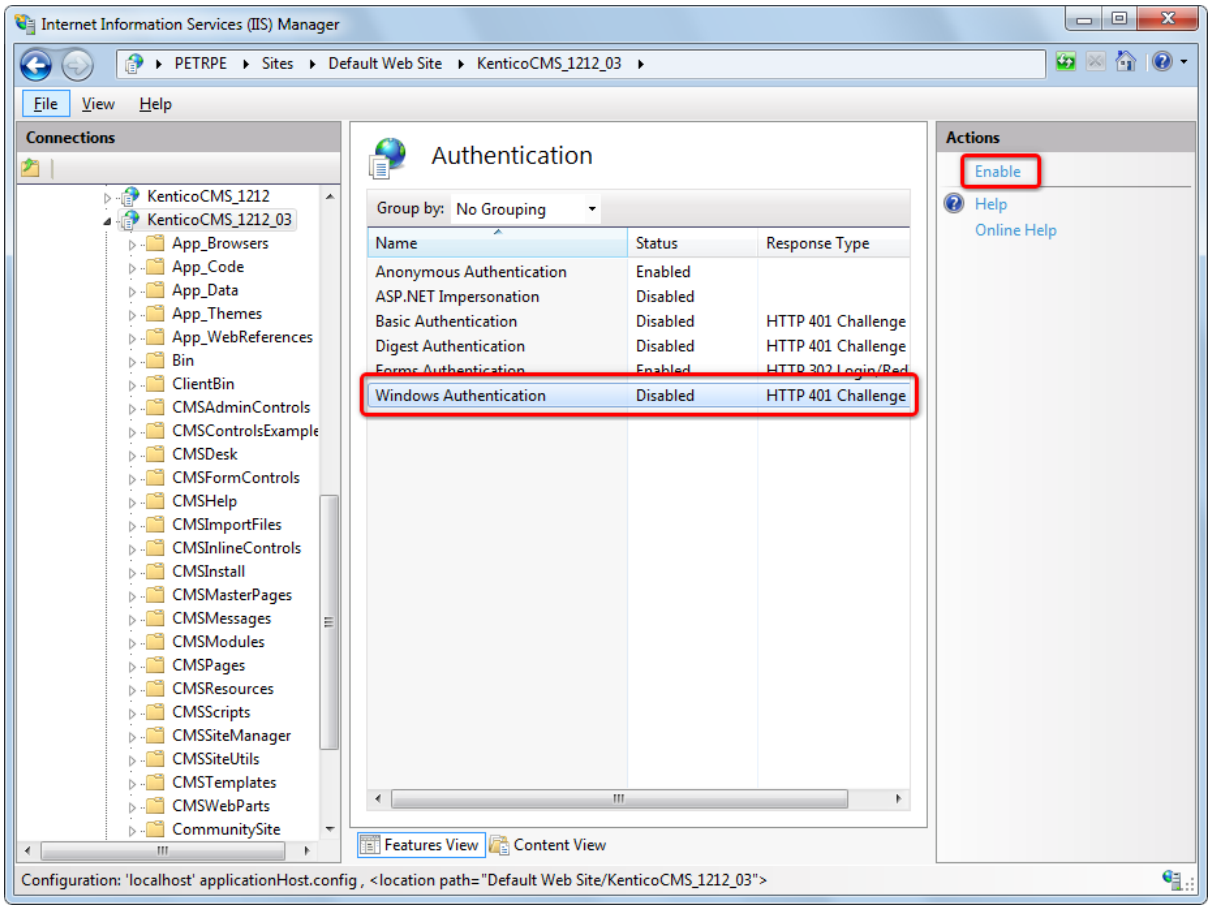
a) Go to **Start** -> **Control Panel** -> **Administrative Tools** and start the **Internet Information Services (IIS) Manager**.



b) Locate and select the media library folder in the IIS tree, then click on the **Authentication** icon.

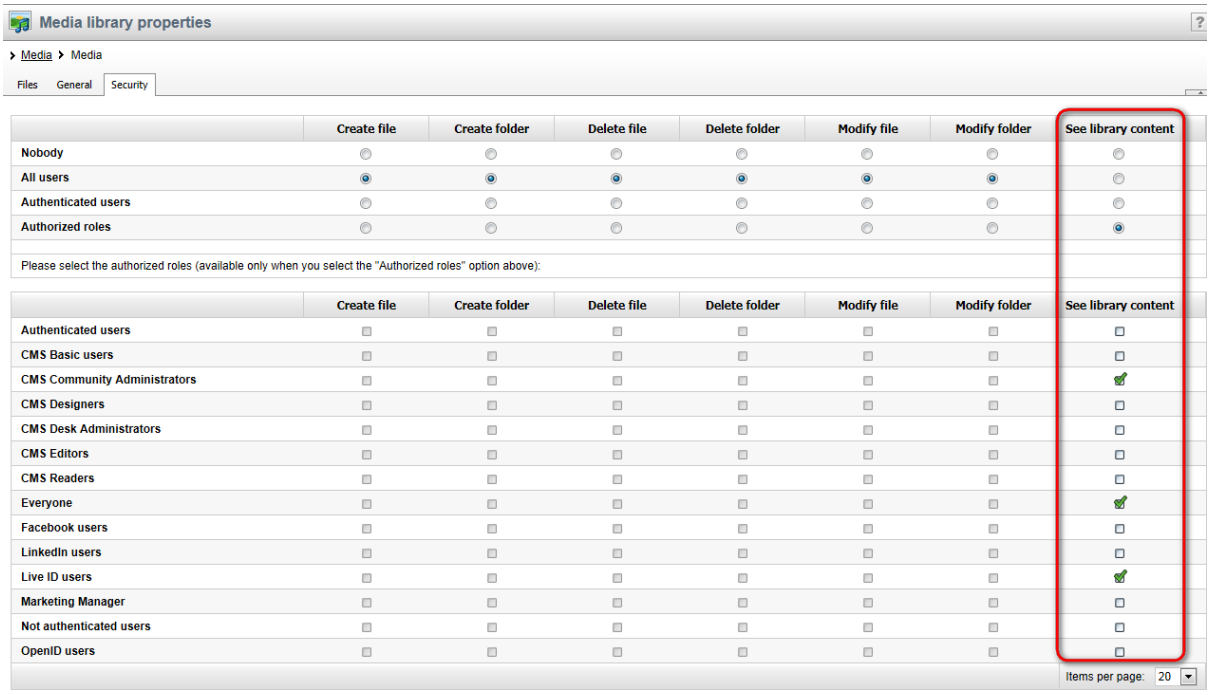


c) Disable the **Anonymous Authentication** by clicking the **Disable** link in the **Actions** list. If enabled, disable also the **Windows Authentication**.



2. Media library security settings

You have to assign the **See library content** permission to the appropriate roles or all authenticated users on the media library's **Security** tab.



You also have to make sure that the **Check files permissions** option is enabled in **Site Manager -> Settings -> Content -> Media**. With this option disabled, permission checks would not be performed.

3. Media gallery web part settings

a) You need to enable the **Use secure links** web part property.

b) When writing your transformations for the **Media gallery** web part, you should stick to the following rules:

File previews and **file details** should be displayed using the following control:

```
<cc1:MediaFilePreview ID="filePreview" runat="server" maxsidesize="117" />
```

Download links should be obtained using the following method:

```
<%# MediaLibraryFunctions.GetMediaFileUrl(Eval("FileLibraryID") ,Eval("FilePath"),
Eval("FileGUID"), Eval("FileName"), GetDataControlValue<bool>("UseSecureLinks")) %
>
```

You can see an example of a real-world use of this web part, including the defined transformations, on the **Community Site** sample website, in the **Media** section.

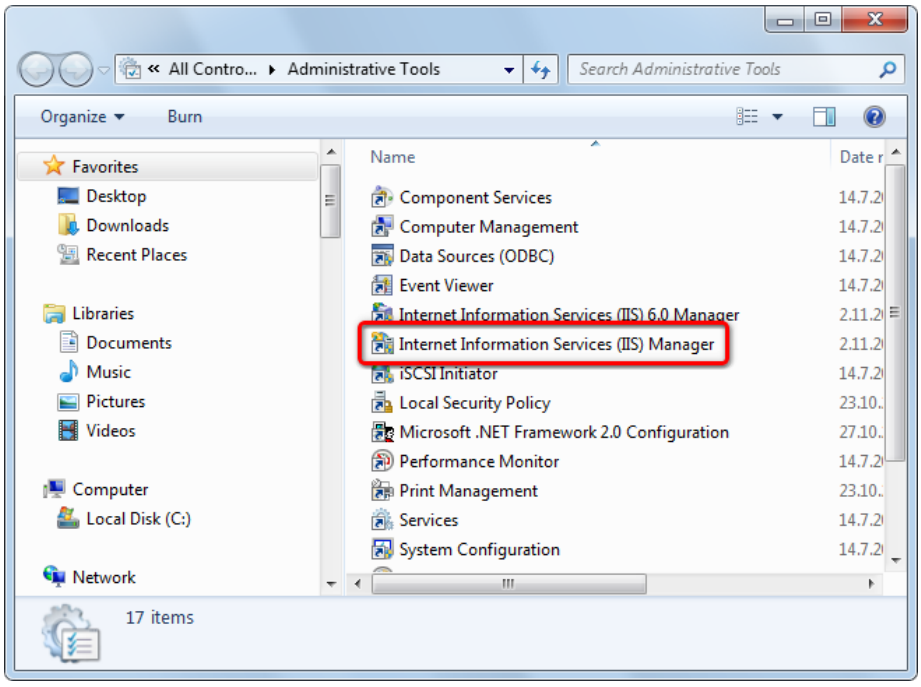
Non-secured libraries

Content of non-secured media libraries is accessible to all site users or visitors. These libraries are also faster than the secured ones, as no permissions need to be checked.

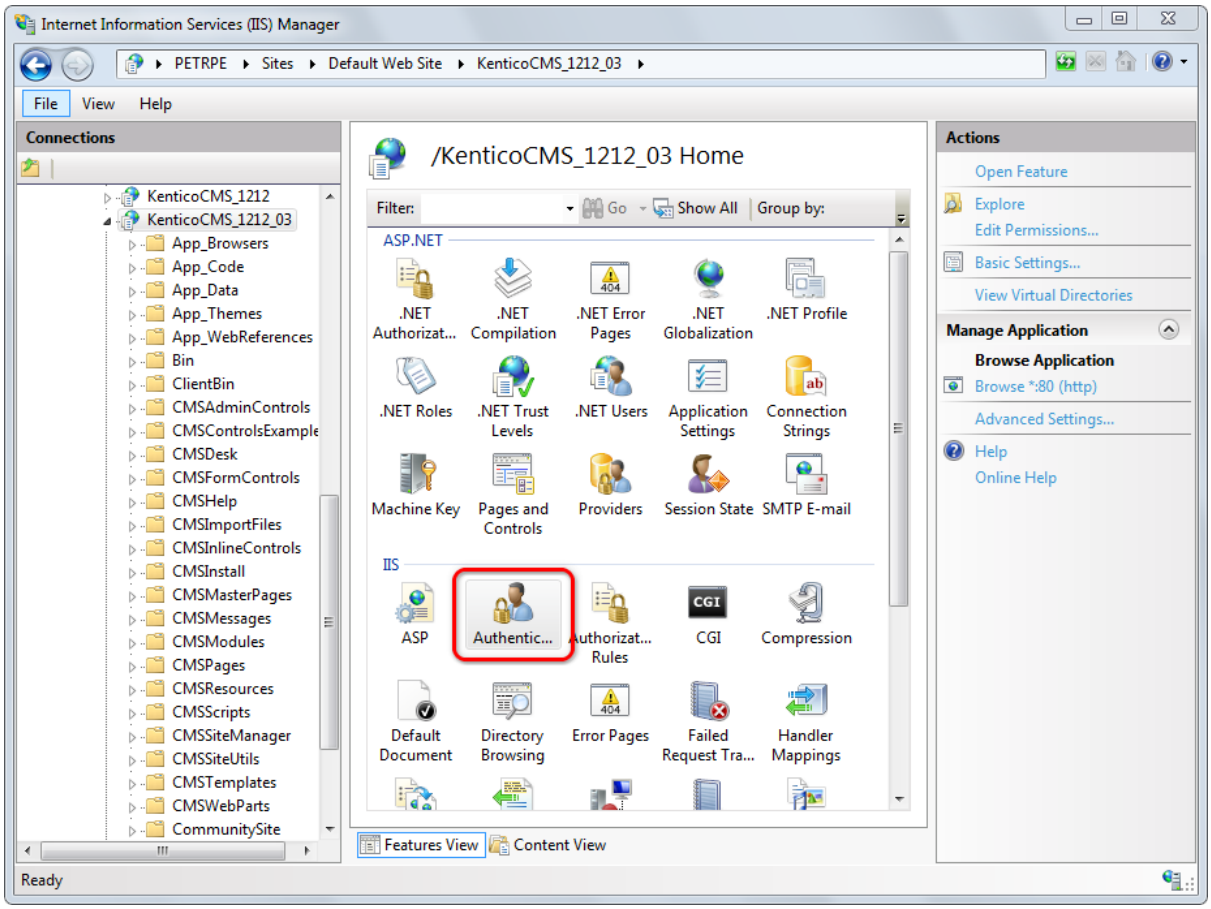
1. IIS setup

You have to **set up your IIS** so that files in the library can be accessed directly by anonymous users.

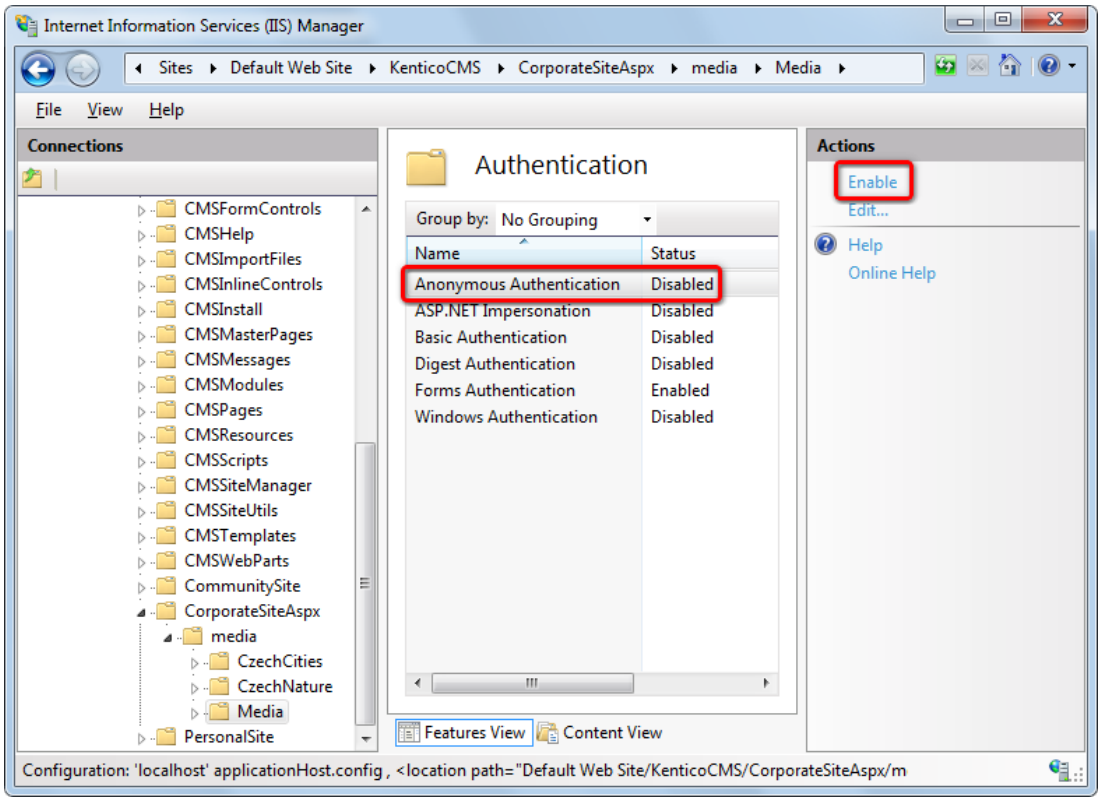
a) Go to **Start** -> **Control Panel** -> **Administrative Tools** and start the **Internet Information Services (IIS) Manager**.



b) Locate and select the media library folder in the IIS tree, then click on the **Authentication** icon.

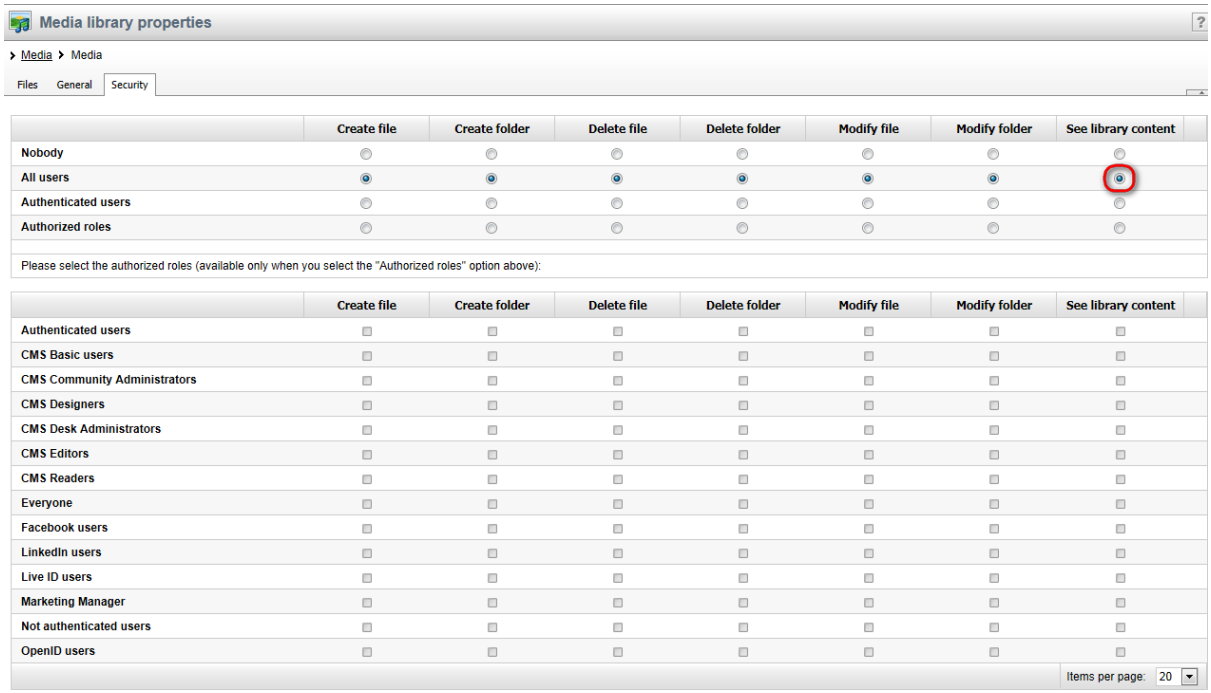


c) Enable the **Anonymous Authentication** by clicking the **Enable** link in the **Actions** list.

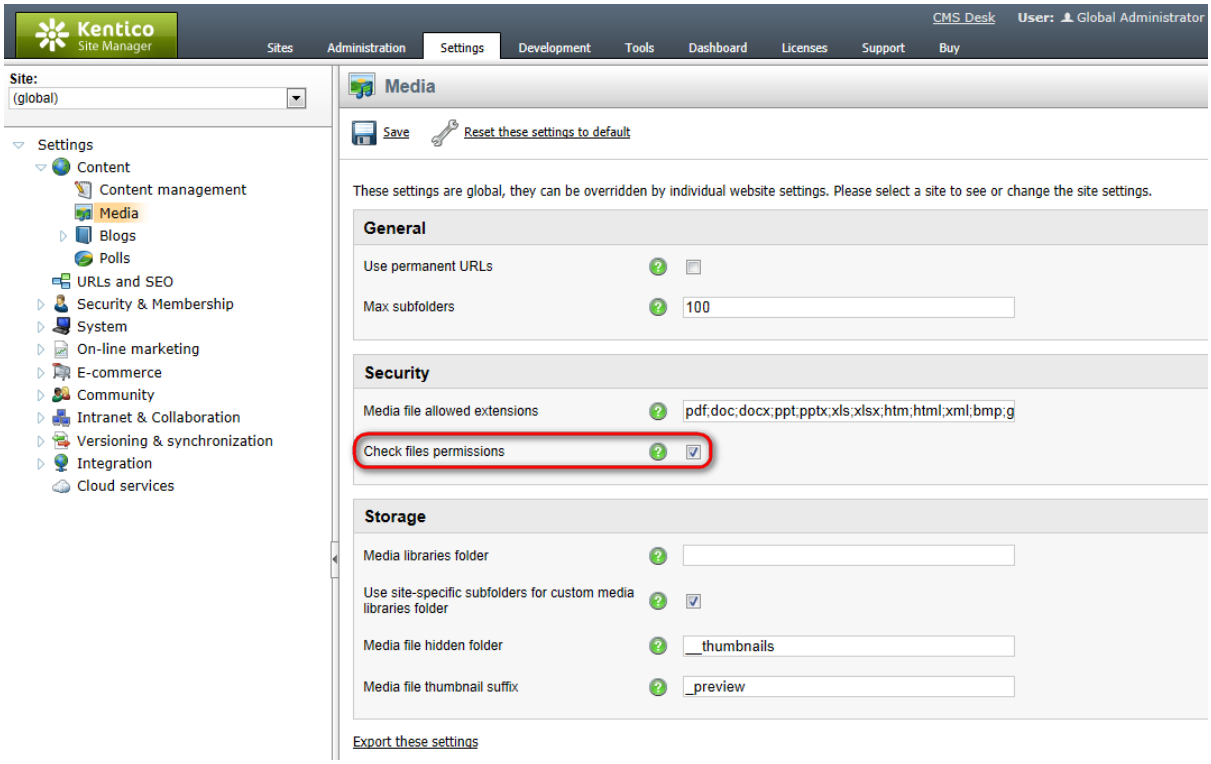


2. Media library security settings

You have to assign the **See library content** permission to **All users** on the media library's **Security** tab.



Alternatively, if all media libraries on the site are non-secured, you can disable the **Check files permissions** option in **Site Manager -> Settings -> Content -> Media**. This disables all permission checks for all media libraries on the site, which enables all users to see the libraries' content.



3. Media gallery web part settings

a) The **Use secured links** web part property should be disabled for the file requesting to be faster.

b) When writing your transformations for the **Media gallery** web part, you should stick to the following rules:

Image previews and **image details** should be obtained using the following method because they need to be resized:

```
<cc1:MediaFilePreview ID="filePreview" runat="server" maxsidesize="117" />
```

Other **file types previews** and **details** can be obtained using direct links. **Download links** can be obtained directly too.

8.32.8 Media internals and API

8.32.8.1 Overview

In this chapter, you will learn which database tables and API classes are used in the Media libraries module. You will also see the most common API examples.

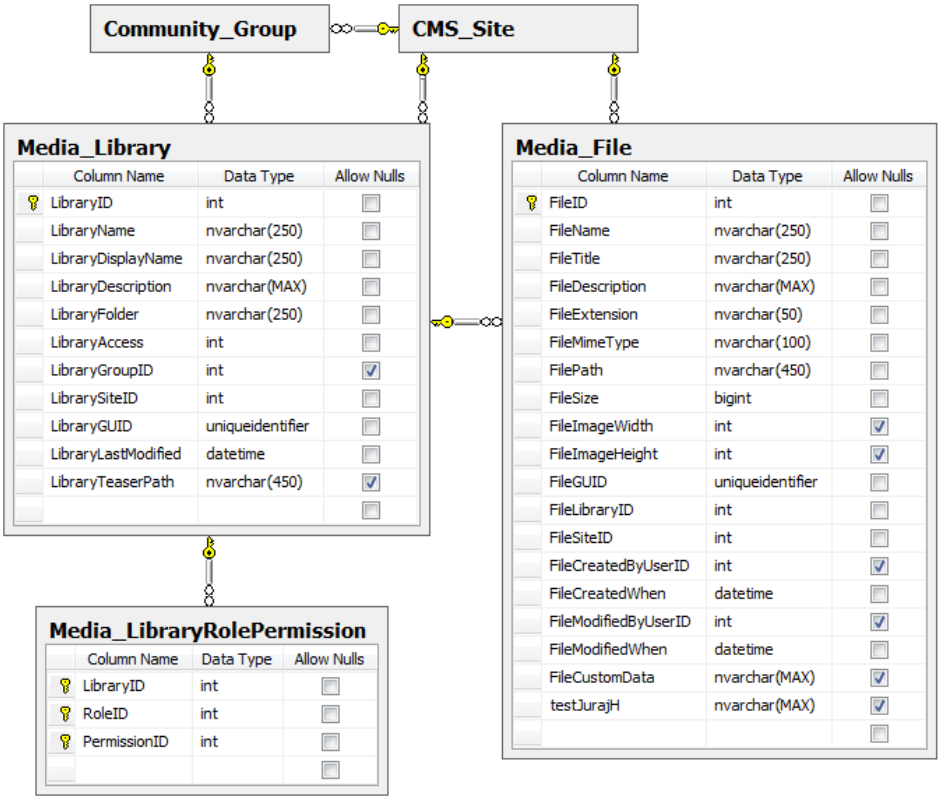
Advanced API examples can be found in the API programming and Kentico CMS internals section of this

guide. For detailed API documentation, such as a list of all methods from the module classes, please refer to Kentico CMS API Reference.

8.32.8.2 Database tables

The following database tables are used in the Media libraries module:

- **Media_Library** - contains records representing media libraries.
- **Media_File** - contains records representing media files.
- **Media_LibraryRolePermission** - represents that particular roles have particular permissions in particular media libraries.



8.32.8.3 API classes

If you are not familiar with the database table data management by Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

> **Please note**
>
> The Media libraries module classes use the **CMS.MediaLibrary** namespace.

Media_Library table API:

- **MediaLibraryInfo** - represents one media library.
- **MediaLibraryInfoProvider** - provides management of media libraries.

Media_File table API:

- **MediaFileInfo** - represents one media file.
- **MediaFileInfoProvider** - provides management of media files.

Media_LibraryRolePermission table API:

- **MediaLibraryRolePermissionInfo** - represents that a particular role has a particular permission in a particular media library.
- **MediaLibraryRolePermissionInfoProvider** - provides management of relationships between media libraries, roles and permissions.

8.32.8.4 API examples

8.32.8.4.1 Overview

Please note

All API examples can be simulated in the API examples user interface; please refer to the API examples chapter for more details.

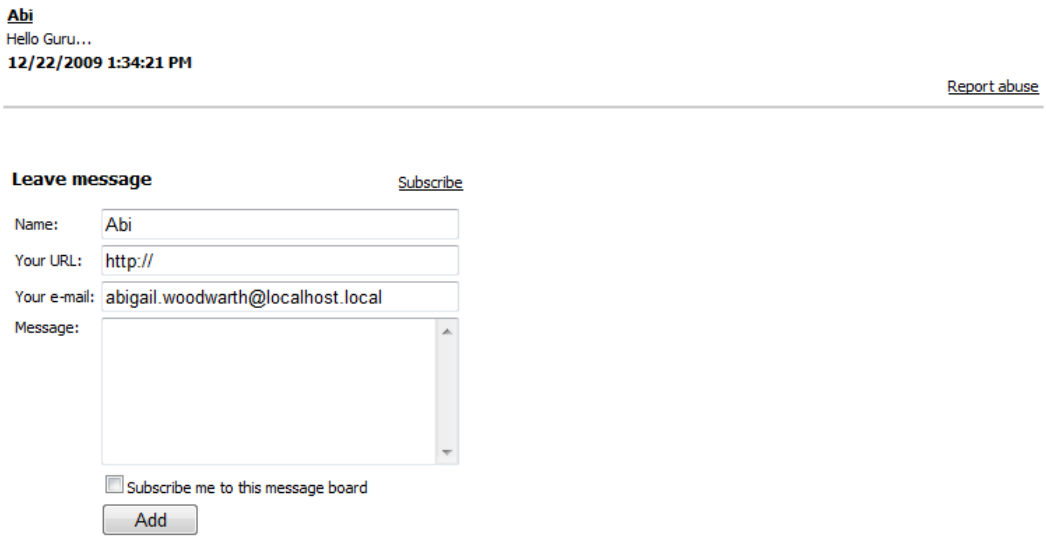
If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Tools\MediaLibrary\Default.aspx.cs***.

8.33 Message boards

8.33.1 Overview

The Message boards module enables site visitors to comment on the content of a particular page. Each message board is related to a document on which it is placed, and optionally to a user or group. Messages can be moderated by board moderators. Users can subscribe to receiving notifications about new messages and rate the content of the document.

The module thus provides similar functionality like the Forums module. However, unlike structured forums, message boards are flat and simpler.



- To learn how to use the Message board web part, please refer to the Using the Message board web part topic.
- To learn how to administrate Message boards, please refer to the Administrating message boards topic.
- To learn how to edit message boards, please refer to the Editing message boards topic.
- To learn how to set Board base URL, please refer to the Setting Board base URL topic.
- Module settings are described in the Settings topic.
- Module security is described in the Security topic.
- Message board notifications are described in the Message board notifications subchapter.
- The internals and API of the Message boards module are described in the Message boards internals and API subchapter.

8.33.2 Using the Message board web part

Message board can be placed on any page (document) on your website. It allows site visitors to send instant comments on the content of the particular page. You can have unlimited number of message boards on one page.

In the following example, you will see the basic process of adding a message board to your site. Our goal will be to add a Message board to the **Your first news** document on the sample **Corporate site**.

1. Log in to CMS Desk as *administrator*. From the content tree, select **News**.

2. Click the **Add web part** ( ) icon at the top right corner of **Main zone** web part zone and in the web part selection dialog, choose **Message board -> Message board** and click **OK**.

3. In the web part properties window, you can set a number of properties. As there is quite a lot of them, we will leave default values and set only those listed below. If you want exact explanation of the meanings of each of the properties, please refer to Kentico CMS Web Parts Reference or click the **Documentation** ( ) link at the top right corner of the web part properties window.

- **Message transformation** - transformation used for displaying board messages; you can use *Community.Transformations.MessageBoard*, which is a default predefined transformation for board messages

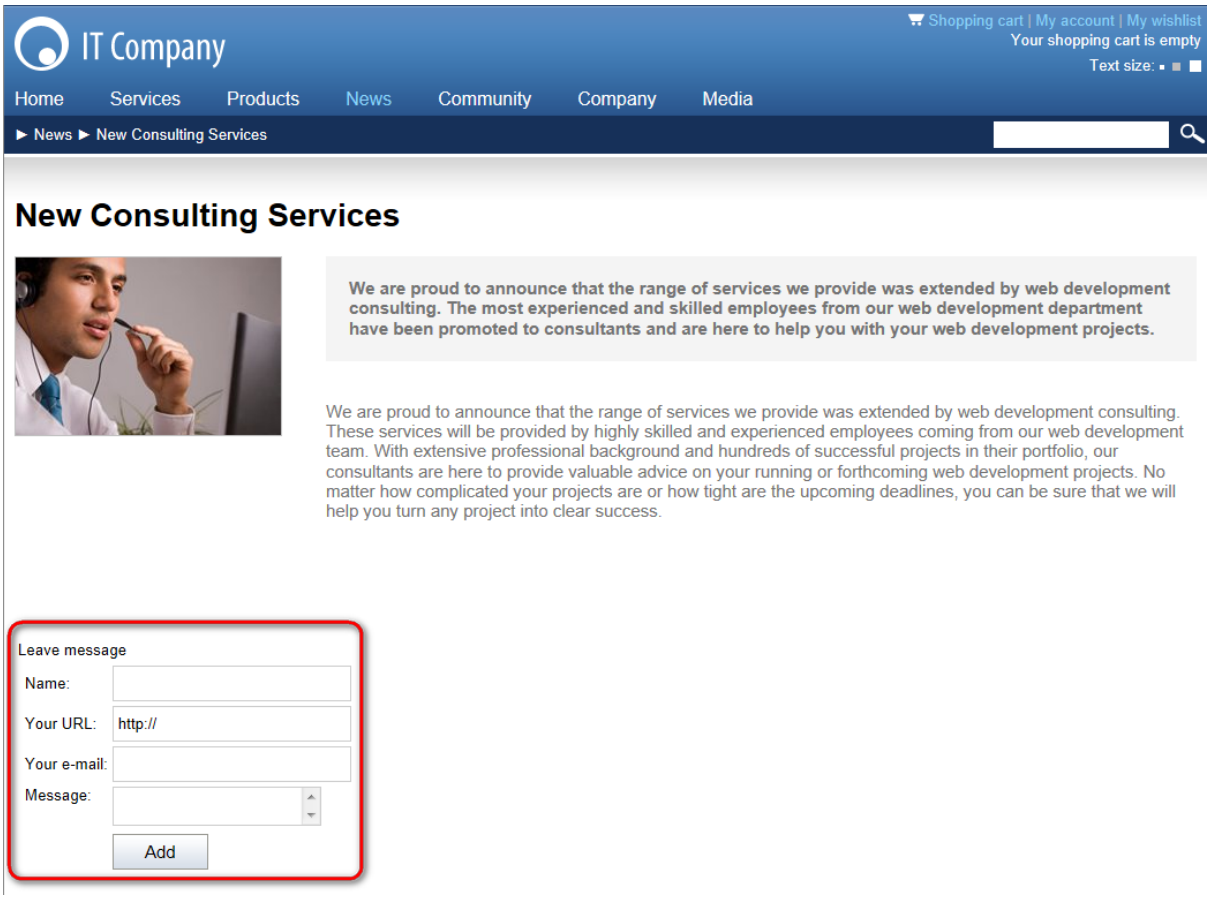
- **No messages text** - text message that will be displayed to site visitors when there are no messages in the board
- **Display name** - display name of the message board, it will be used in the administration interface to identify the message board, so it is advisable to use some well-descriptive name under that you will recognize the message board; it is useful to use macros, such as *{%SiteName%} - my first message board*

Click **OK** when you are finished entering the values.

Board default settings

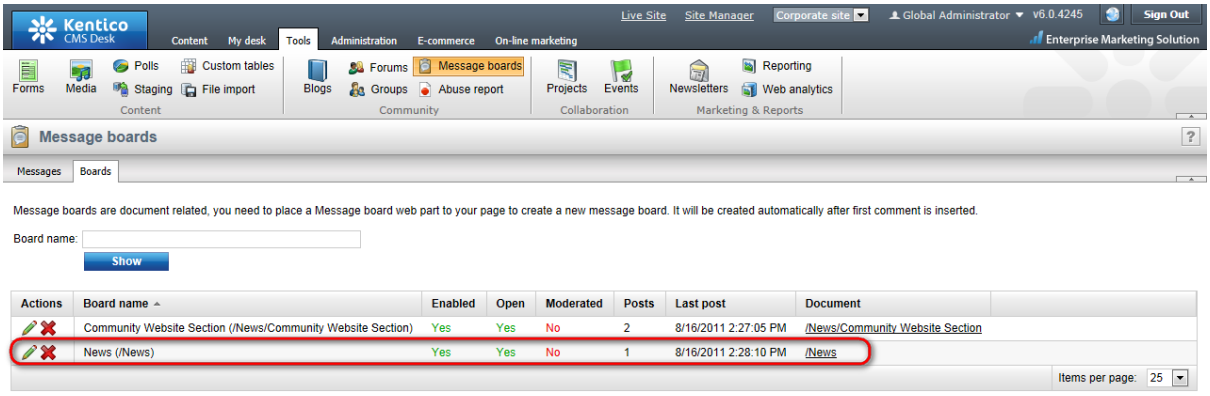
Properties in the **New board settings** section will be used when the message board is created (after first message or subscription). Thereafter, changes made to the web part properties will take no effect. To change some of these properties, you will have to edit the message board in **CMS Desk -> Tools -> Message boards -> Boards** tab.

4. If you switch to the **Live Site** mode, you should see the form for entering board messages and the 'No messages text' displayed above it, as you can see in the following screenshot.



5. If you go to the administration interface located in **CMS Desk -> Tools -> Message boards**, you won't see the message board yet as message boards are created in the administration interface section

after first message is added to it, **not after adding the web part** to the site. Try adding some message now. After doing so, go to **CMS Desk -> Tools -> Message boards**. You should see the new message board in the list as in the following screenshot.



8.33.3 Administrating message boards

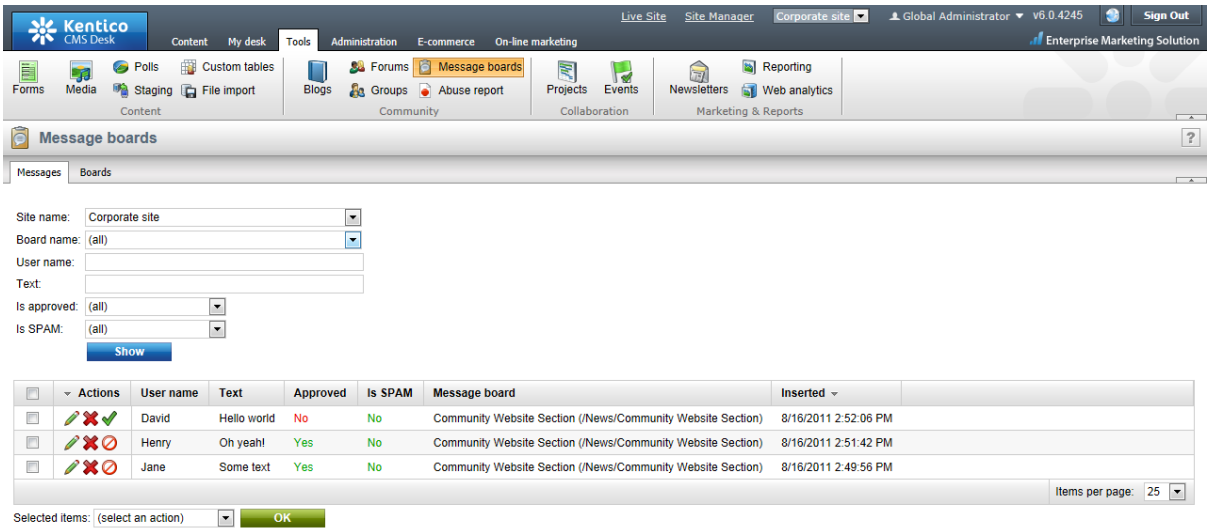
Message boards administration can be performed in **CMS Desk -> Tools -> Message boards**. The section is divided into two tabs.

Messages tab

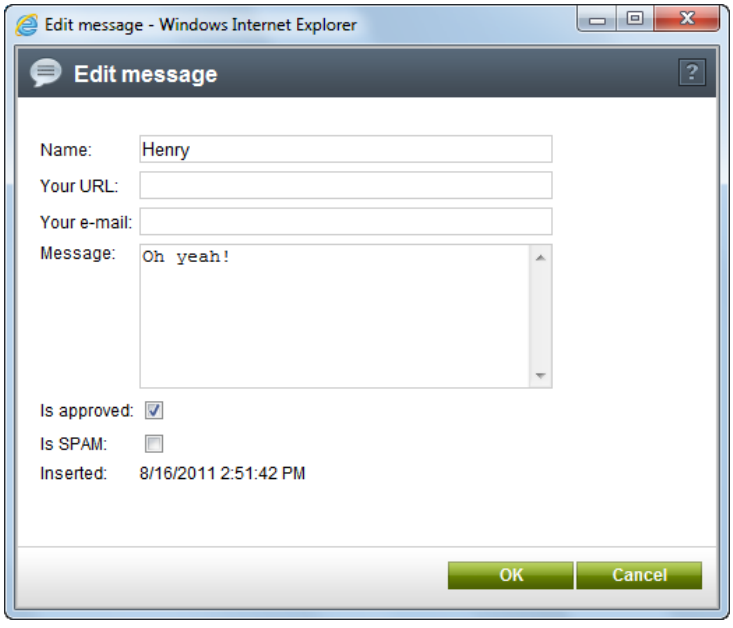
On this tab, moderators can manage board messages. By default, there are only messages requiring approval displayed when you access the page, so that the moderator sees only new messages that need to be approved or rejected. Until the messages are approved, they won't be displayed on the message board. Rejected messages won't be displayed either.

Using the filter above the list, you can determine which messages you want to display. The following filtering parameters are available:

- **Site name** - only messages from the selected site will be displayed
- **Board name** - only messages from the selected message board will be displayed
- **User name** - only messages posted by the user specified here will be displayed; you can also enter only a part of the user name
- **Text** - only messages containing the entered text will be displayed
- **Is approved** - you can choose whether to display only approved or disapproved messages
- **Is SPAM** - you can choose whether to display only messages marked or not marked as SPAM



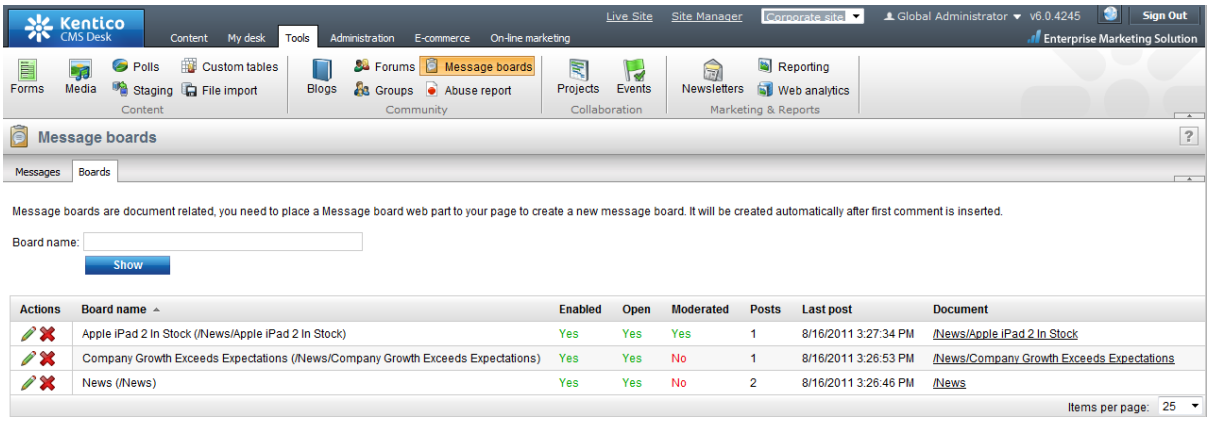
Messages in the list can be **Approved** (✔), **Rejected** (⊘), **Deleted** (✖) or **Edited** (✎). If you choose to edit a message, the following window pops up, letting you make changes to it. You can also select more messages using the check-boxes and perform one of these actions for all of them using the **Selected items** drop-down list and clicking the **OK** button.



Boards tab

On this tab, you can see a list of all message boards on the current site. Using the **Board name** field above the list, you can filter the displayed message boards. You don't need to enter the exact name, you can enter only a part of it and the list will display all message boards with name containing the entered expression. It is a good idea to give your message boards well-descriptive names so that you can tell one from another and search efficiently.

Message boards can be **Edited** (✎) and **Deleted** (✖) on this tab.



8.33.4 Editing message boards

You can edit message board properties at **CMS Desk -> Tools -> Message boards**, on the **Boards** tab. If you click the **Edit** ( ) icon next to some message board, its editing interface will be displayed.

The administration interface reflects the **Message board** web part properties. When you add the web part to a page, you can set its properties in the web part properties dialog. When the first message is added to the board, the message board is created in this section of the administration interface. At this point, changes made to the web part properties have no effect and you have to make all changes only in this section!

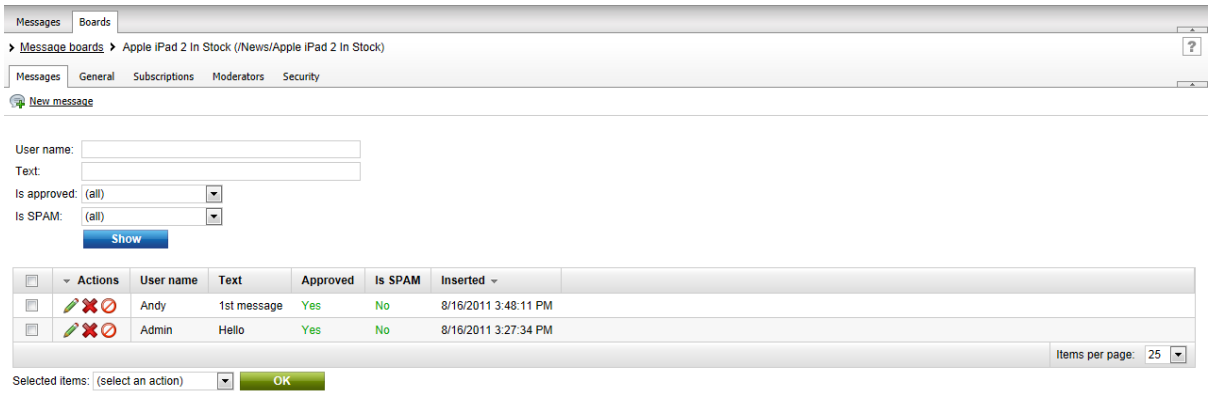
The administration interface for editing message boards is divided into five tabs:

Messages tab

This tab displays a list of all messages on the message board. Using **New message**, you can add new messages to the board directly from the administration interface. Below is a filter, letting you display only messages matching specified criteria. The following filtering parameters are available:

- **User name** - only messages posted by the user specified here will be displayed; you can also enter only a part of the user name.
- **Text** - only messages containing the entered text will be displayed.
- **Is approved** - you can choose whether to display only approved or disapproved messages.
- **Is SPAM** - you can choose whether to display only messages marked or not marked as SPAM.

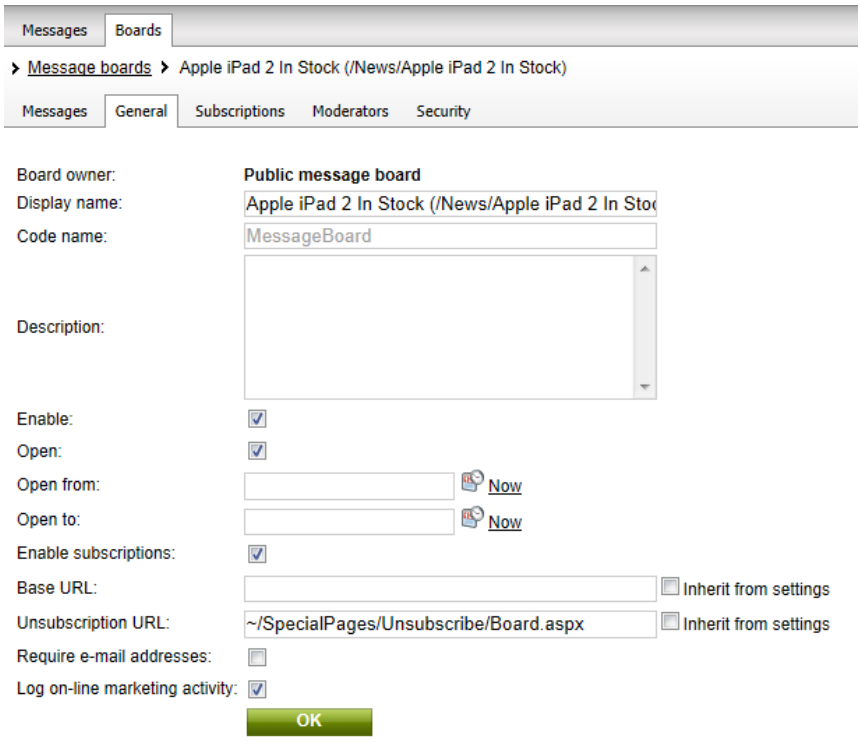
Messages can be **Edited** ( ), **Deleted** ( ), **Approved** ( ) or **Rejected** ( ). You can also select more messages using the check-boxes and perform one of these actions for all of them using the **Selected items** drop-down list and clicking the **OK** button.



General tab

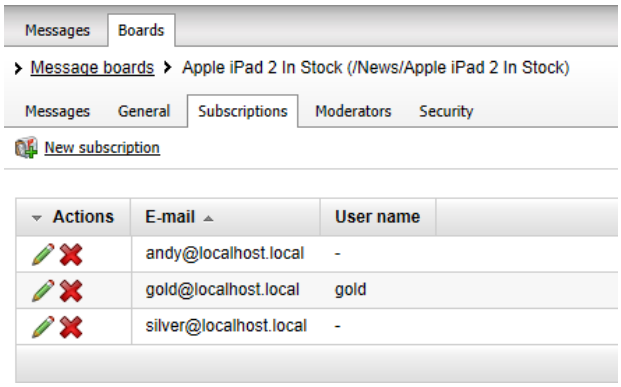
On the general tab, you can specify the following properties of the message board:

- **Display name** - name that is displayed in the user interface.
- **Code name** - name that is used in code.
- **Description** - text describing the message board.
- **Enable** - if unchecked, the message board will be hidden from the live site; if checked, the message board will work normally.
- **Open** - if checked, users can add messages to the message board; if unchecked, messages are displayed but cannot be added.
- **Open from / to** - using these fields, you can define the time interval during which users can add new messages.
- **Enable subscriptions** - if checked, users can subscribe to receiving notifications about new messages on the board.
- **Base URL** - URL used as the URL base of links to message boards in notification e-mails; if empty, value from **Site Manager -> Settings -> Community -> Message boards -> Board base URL** will be used; if that setting is also empty, message boards cannot be placed on pages with wildcard URLs.
- **Unsubscription URL** - URL of the page containing the **Message board unsubscription** web part. The web part handles notification unsubscription requests. If the value is not set, the value in **Site Manager -> Settings -> Community -> Message boards -> Board unsubscription URL** will be used.
- **Require e-mail addresses** - if checked, users are required to enter their e-mail address when posting board messages.
- **Enable double opt-in** - indicates if double opt-in should be enabled for message boards. When enabled, users are required to confirm their subscription by clicking a link that is sent to them in an e-mail.
- **Approval page path** - path to the page that contains the **Message board subscription confirmation** web part. The subscription confirmation link that will be sent to users will point to this page.
- **Send double opt-in confirmation** - indicates if an e-mail confirmation should be sent to user after they approve a subscription. If double opt-in is disabled, these confirmation e-mails are always sent.



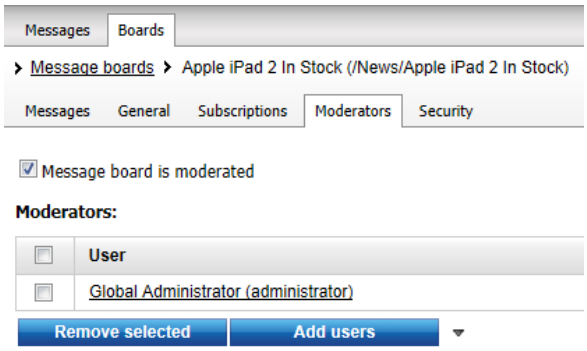
Subscriptions tab

On this tab, you can see a list of subscriptions to receiving notifications about new board messages. You can create new subscriptions using the **New subscription** button. Displayed subscriptions can be filtered by **E-mail** address and **User name**. You can also **Edit** (🖉) or **Delete** (✖) subscriptions in the list.



Moderators tab

On this tab, you can make the message board moderated by checking the **Message board is moderated** check-box. In such case, new messages will be displayed only after approval by some of the moderators listed in the **Moderators** list-box below. Moderators can be **Added** or **Removed** using the corresponding buttons.



Security tab

On this tab, you can set security-related properties of the message board.

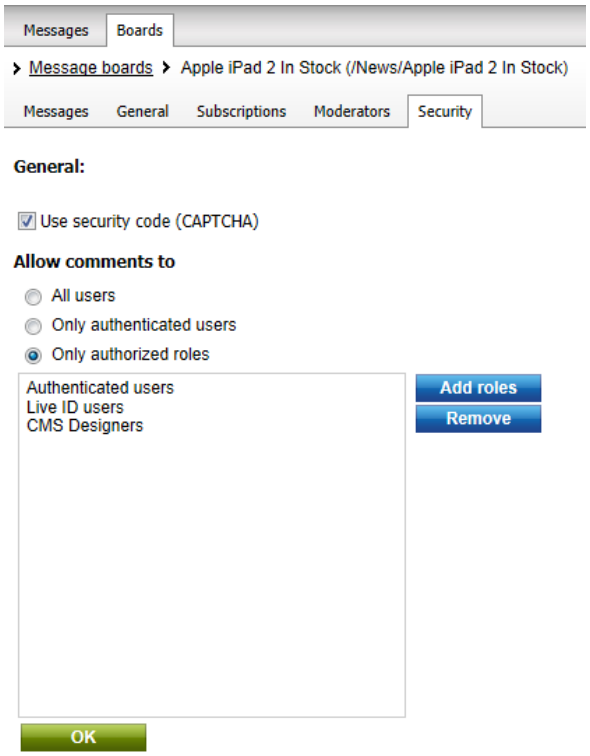
If the **Use security code (CAPTCHA)** check-box is checked, users will have to retype the CAPTCHA security code before adding a new message.

The **Allow comments to** section can be used to define which users can add new messages to the board. The following options are available:

- **All users** - everyone can add messages to the board
- **Only authenticated users** - only signed-in users can add messages to the board
- **Only authorized roles** - only members of the roles in the list-box below can add messages to the board

The following two options can be set only for group message boards:

- **Only group members** - only members of the group can add messages to the board
- **Only group admin** - only administrators of the group can add messages to the board



8.33.5 Setting Board base URL

Board base URL is the URL which will be used as the URL base for unsubscription links in notification e-mails. It can be set in two ways:

- In **Site Manager -> Settings -> Community -> Message boards -> Board base URL**; from here, it can be inherited by the web parts.
- Directly in **Message board web part properties.**

The following rules should be followed when creating user and group message boards placed on these pages in order for the unsubscription links to work correctly:

1. When you create a **user message board**, which is a message board placed on a user's profile, it is recommended to set the **Board base URL** directly in web part properties, for example like this:

```
~/Members/Profile.aspx
```

You can find a live example of this setting on the Community Site sample website.

2. When you create a **group message board**, which is a board placed on a group's profile, it is recommended to set the **Board base URL** directly in web part properties, for example like this:

```
~/Groups/Profile.aspx
```

3. When you create a public message board, which is a board placed on any document without a wildcard URL, the Board base URL property needn't be set.

8.33.6 Settings

Settings of the **Message boards** module are located in **Site Manager -> Settings -> Community -> Message boards**. Among other community-related settings, the following settings are related to the **Message boards** module:

- **Send message board e-mails from** - e-mail address that will appear in the From field of notification messages about new message board messages.
- **Board base URL** - global board base URL that can be inherited by message boards; it can be used by board notification e-mails and message board viewers.

- **Board unsubscription URL** - URL of the site on which the **Message board unsubscription** web part is placed; this web part handles requests for unsubscription from notifications about new message board messages.
- **Enable double opt-in for message boards** - indicates if double opt-in should be enabled for message boards. When enabled, users are required to confirm their subscription by clicking a link that is sent to them in an e-mail.
- **Double opt-in approval page path** - path to the page that contains the **Message board subscription confirmation** web part. The subscription confirmation link that will be sent to users will point to this page.
- **Double opt-in interval (hours)** - amount of time in hours, during which a user has to confirm their subscription request.
- **Send double opt-in confirmation** - indicates if an e-mail confirmation should be sent to user after they approve a subscription. If double opt-in is disabled, these confirmation e-mails are always sent.

8.33.7 Security

Message board

Based on the **Access** and **Message board owner** properties of the **Message board** web part, you can determine who will be allowed to add new messages to the board.

Changing the values

Remember that once the message board is created (after inserting first message or subscribing), you cannot make changes to these settings in the **New board settings** section of web part properties. You can only modify values of the **Access** property on the **Security** tab when editing the corresponding message board in **CMS Desk -> Tools -> Message boards**.

The following table explains **who can add messages to the board** under particular configurations. The difference between **User** and **Public boards** is that **Public boards** are always related to a document, while **User boards** are always related to a document and a user.

Public boards are typically used when you want multiple users to post messages to it. **User boards**

are typically used on user profiles, as you can see on the **Community starter site** sample website, on the **Members -> Profile** page.

Message board owner	Access	Anonymous user	Authenticated user	Authorized role	Owner	Owner in authorized role
Public board	All users	Yes	Yes	Yes	Yes	Yes
Public board	Authenticated users		Yes	Yes	Yes	Yes
Public board	Authorized roles			Yes		Yes
Public board	Owner				Yes	Yes
User	All users	Yes	Yes	Yes	Yes	Yes
User	Authenticated users		Yes	Yes	Yes	Yes
User	Authorized roles			Yes		Yes
User	Owner				Yes	Yes

When a board is in the **User x Owner** configuration, the following conditions need to be met in order for the current user to be able to post messages:

- the page must be accessed with *userid* or *username* parameter in querystring
- the current user must be the same as the one whose *userid* or *username* is passed in querystring
- the current user must not be hidden (configured by the *Is hidden* option when editing the user)

This can be typically used on user profiles, where messages to such board can be posted only by the owner of the profile, while other users can only read these messages. An example of such board is the **MessageBoardAnnouncements** board on the **/Members/Profile** page of the sample Community Site.

Group message board

The Group message board is always related to some group, hence only the **Access** property can be set. You can see a typical usage of this web part on the **Community starter site** sample website, on the **Groups -> Profile** page.

Access	Anonymous user	Authenticated user	Authorized role	Group member	Group member in authorized role	Group admin
All users	Yes	Yes	Yes	Yes	Yes	Yes
Authenticated users		Yes	Yes	Yes	Yes	Yes
Authorized roles			Yes		Yes	Yes
Group members				Yes	Yes	Yes
Group admin						Yes

When the **Group members** option is set, the following conditions need to be met in order for the current

user to be able to post messages:

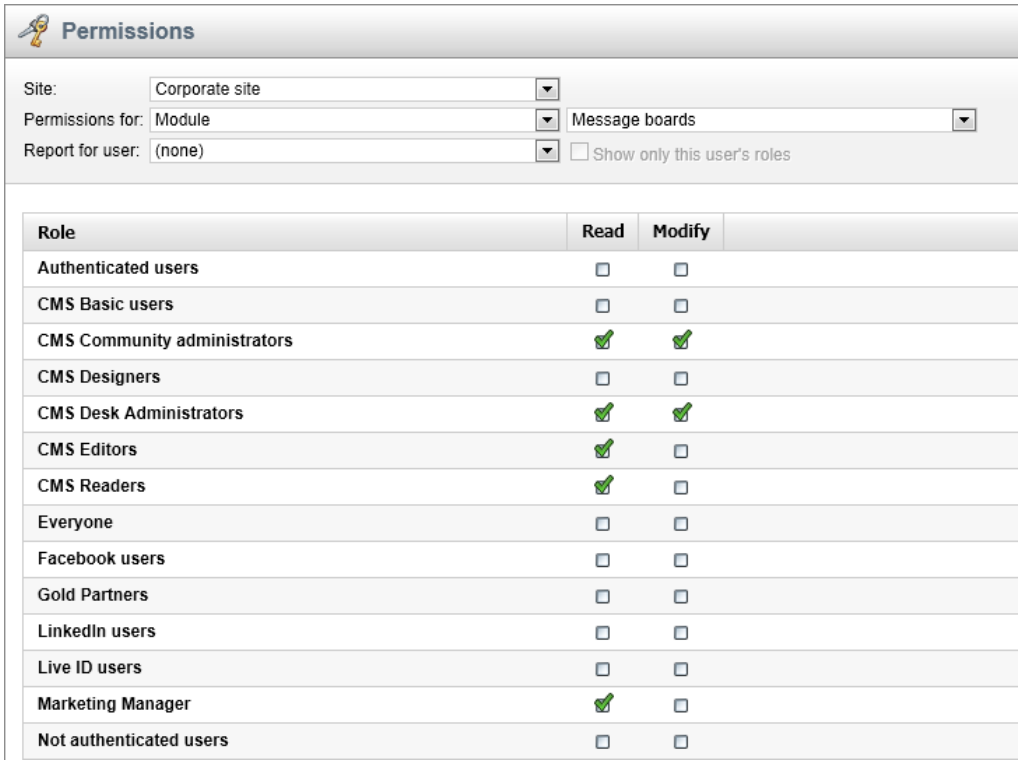
- the page must be accessed with *groupid* parameter in querystring
- the current user must be member of the group whose *groupid* is in the querystring
- the current user must not be hidden (configured by the *Is hidden* option when editing the user)

This can be typically used on group profiles, where messages to such board can be posted only by members of the group, while other users can only read these messages. An example of such board is the **GroupMessageBoard** board on the **/Group-pages/<group name>** page of the sample Community Site.

Permissions

Permissions for access to **Message boards** administration interface can be set in **Site Manager -> Administration -> Permissions**. You have to select the **Modules -> Message boards** permission matrix.

- **Modify** - members of the roles are allowed to edit message board settings, delete the boards and manage message board posts
- **Read** - selected role members are allowed to read the records and configuration of particular message boards, but are not allowed to modify them



8.33.8 Message board notifications

8.33.8.1 Overview

The Message boards module enables users to subscribe to receiving notifications about messages that have been added to your message board.

- To learn who can receive notifications when a new message is added or during message editing, please refer to the Who can be notified topic.
- To learn how to let users subscribe to receiving notifications about new messages added to the message board, please refer to the User subscriptions topic.
- E-mail templates used when sending notifications to subscribers and board moderators are described in the E-mail templates topic.

8.33.8.2 Who can be notified

When a new message is added or edited, notifications can be sent to:

- **Board moderators**
- **Subscribers**

The following text explains to whom notifications are sent under specific conditions.

New message inserted

1. Message has been added by a board moderator, global administrator, board owner (in case of a user board), user with the **Modify** permission, or group administrator (in case of group boards):

- the message is marked as APPROVED
- the e-mail is sent to subscribers

2. Message has been added by anybody else
a) board is moderated

- the message is marked as NOT APPROVED
- the e-mail is sent to moderators

b) board is not moderated

- the message is marked as APPROVED
- the e-mail is sent to subscribers

Existing message edited

1. the message is switched from NOT APPROVED to APPROVED

- the e-mail is sent to subscribers

2. other message changes

- no e-mail is sent

8.33.8.3 User subscriptions

You can let users subscribe to receiving notifications about new messages added to a message board. This topic describes the different ways in which you can set up and manage message board subscriptions.

Enabling subscriptions

You can enable your site visitors to subscribe to a message board by doing one of the following:

- In case of a new board, i.e. one that hasn't been posted to yet, open the properties of the particular **Message board** web part and set the **Enable subscriptions** property to true.
- In case of an existing board, go to **CMS Desk -> Tools -> Message boards -> Message boards**, edit your board and on the **General** tab, check the **Enable subscription** box, then click **Save**.

Enabling double opt-in

To make users confirm that the e-mail address they're subscribing with really exists and belongs to them, you can enable the double opt-in functionality. With double opt-in enabled, subscribers will be sent an e-mail with a confirmation link. They will have to click the link to create their subscription.

To enable double opt-in globally for all message boards:

1. Go to **Site Manager -> Settings -> Community -> Message boards**.

2. Turn on the **Enable double opt-in for message boards** setting.

3. Specify how long you want the confirmation links to be valid in the **Double opt-in interval** setting.

The following two steps are optional. If you omit them, the system will use the default approval page located in *~/CMSModules/MessageBoards/CMSPages/SubscriptionApproval.aspx*.

4. Place the **Message board subscription confirmation** web part on a new page. Adjust its properties according to your needs.

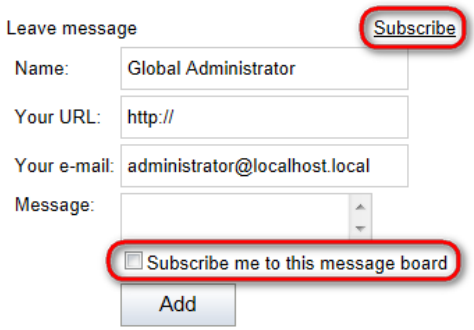
5. Enter the page's path into the **Double opt-in approval page path** setting.

Double opt-in should now be configured for all message boards. You can override the settings for each existing message board in its properties in **CMS Desk -> Tools -> Message boards**, or for not yet existing message boards in the properties of the respective **Message board** web part.

The functionality uses the **Boards - Subscription request** e-mail template. You can insert the confirmation link into the template using the *{%SubscriptionLink%}* macro.

Subscribing to a board

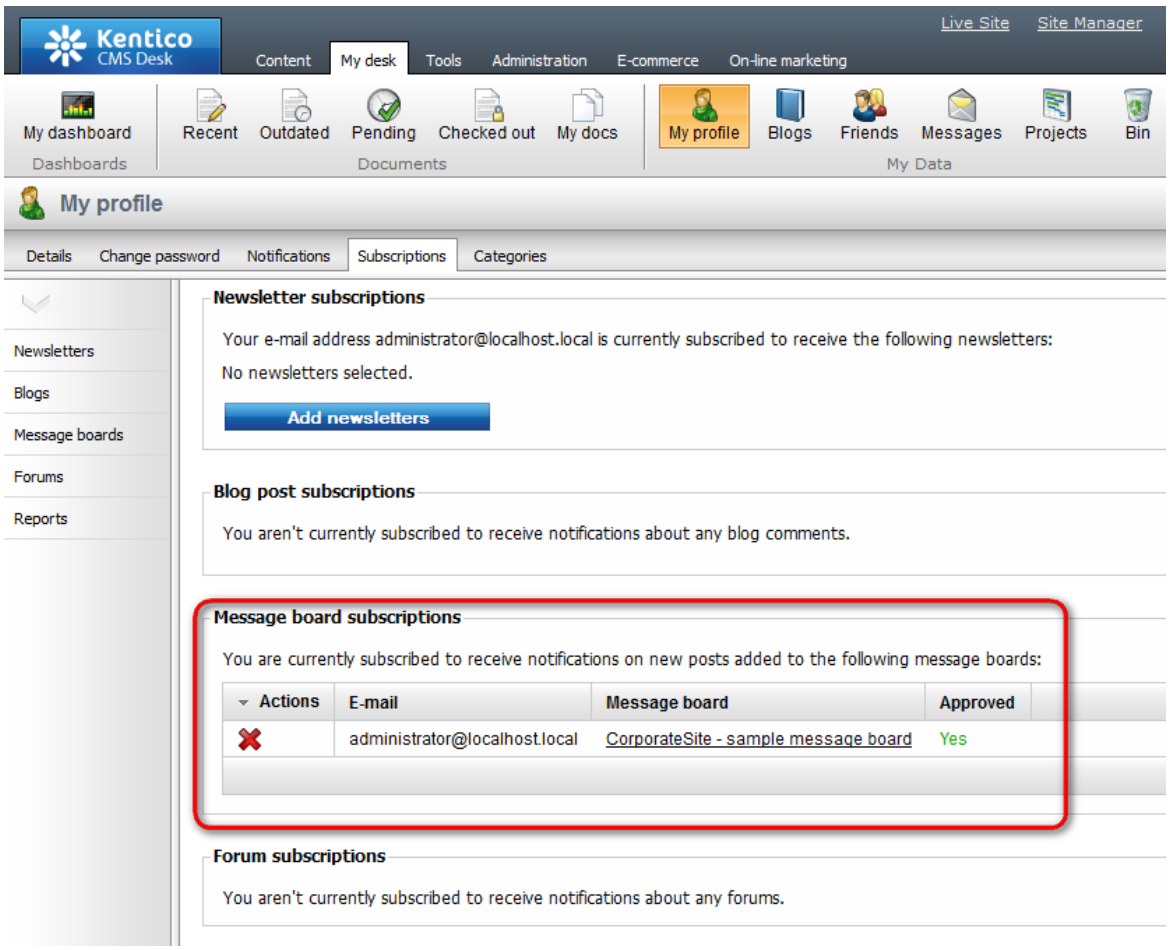
Users can subscribe in two ways. They can either check the **Subscribe to message board** checkbox, which subscribes them along with adding the message. Or, they can click the **Subscribe to board** link, which displays only one **E-mail** field. After entering their address and clicking the **Subscribe** button, they can subscribe to notifications for this board without leaving any message.



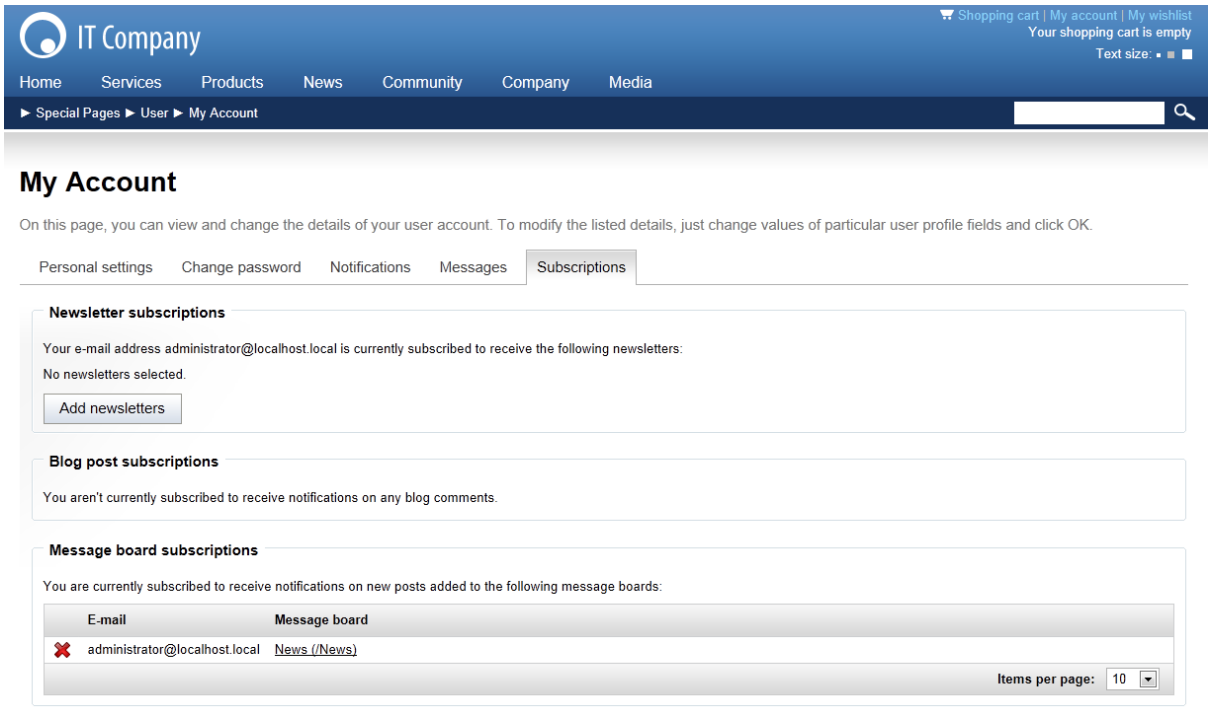
Managing subscriptions

Users can view their subscriptions and potentially **unsubscribe** using the **Delete** (✖) icon at the following two places:

- Users with access to **CMS Desk** can view their subscriptions on the **My Desk -> My profile -> Subscriptions** tab.



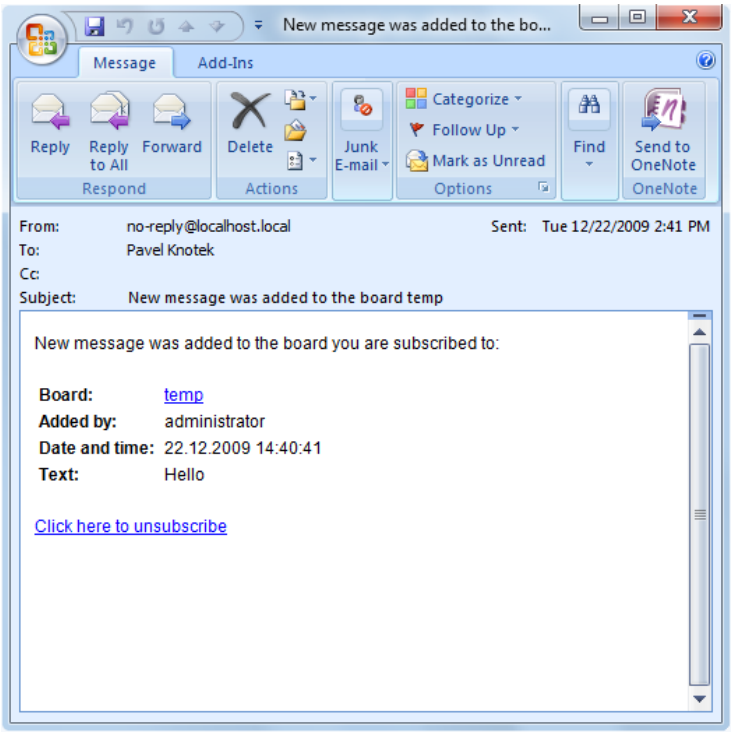
- On the live site, users can view their subscriptions in the **My account** web part. The **Display my subscriptions** property of the web part must be enabled for this to be possible.



In both these locations, users can **Delete** the subscription. If double opt-in is enabled and a subscription hasn't been yet confirmed by clicking the link in the e-mail the user received, they can also **Approve** the subscription.

Unsubscription link configuration

Each message board notification e-mail contains an unsubscription link (when using the default e-mail template). By clicking the link, users can unsubscribe from receiving notifications about new messages.



For the unsubscription links to work, you have to do the following tasks:

1. Place the **Message board unsubscription** web part on a page. It is recommended to create a special page for this purpose, as you can see at **Corporate site -> Special pages -> Unsubscribe -> Board** or **Community site -> Special-pages -> Board unsubscribe**. You can set only one property of the web part - **Confirmation text**. This is the text that will be displayed after successful unsubscription.

2. Set the URL of the page created in step 1 as the **Unsubscription URL** property of the message board. This can be done in three ways:

- Enter the URL into the **Board unsubscription URL** field in **Site Manager -> Settings -> Community -> Message boards**. This URL will be used by default when no URL is entered in web part properties of the message board.
- When adding the **Message board** web part, you can set set its **Unsubscription URL** property. This setting overrides the option in **Site Manager -> Settings -> Community -> Message boards**.
- When the message board is created, you can edit the **Unsubscription URL** property on the **CMS Desk -> Tools -> Message boards -> Boards** tab **-> Edit (**) the message board **-> General** tab.

8.33.8.4 E-mail templates

There are two different e-mail templates that can be used when sending notifications to subscribers and board moderators:

- **Subscribers** - e-mails are based on the **Boards - Notification to board subscribers** template.
- **Moderators** - e-mails are based on the **Boards - Notification to board moderators** template.

There is also a template that is used when double opt-in is enabled - **Boards - Subscription request**.

The following macros can be used in the notification e-mails:

Data macros

- **Board.XXX** - where XXX are columns of the *Board_Board* table.
- **Message.XXX** - where XXX are columns of the *Board_Message* table.
- **MessageUser.XXX** - where XXX are columns of the *CMS_User* table.
- **MessageUserSettings.XXX** - where XXX are columns of the *CMS_UserSettings* table.

Source macros

- **DocumentLink** - link to the document where the board is placed.
- **SubscriptionLink** - link used to confirm the subscription when double opt-in is enabled.
- **UnsubscriptionLink** - unsubscription link.

8.33.9 Message boards internals and API

8.33.9.1 Overview

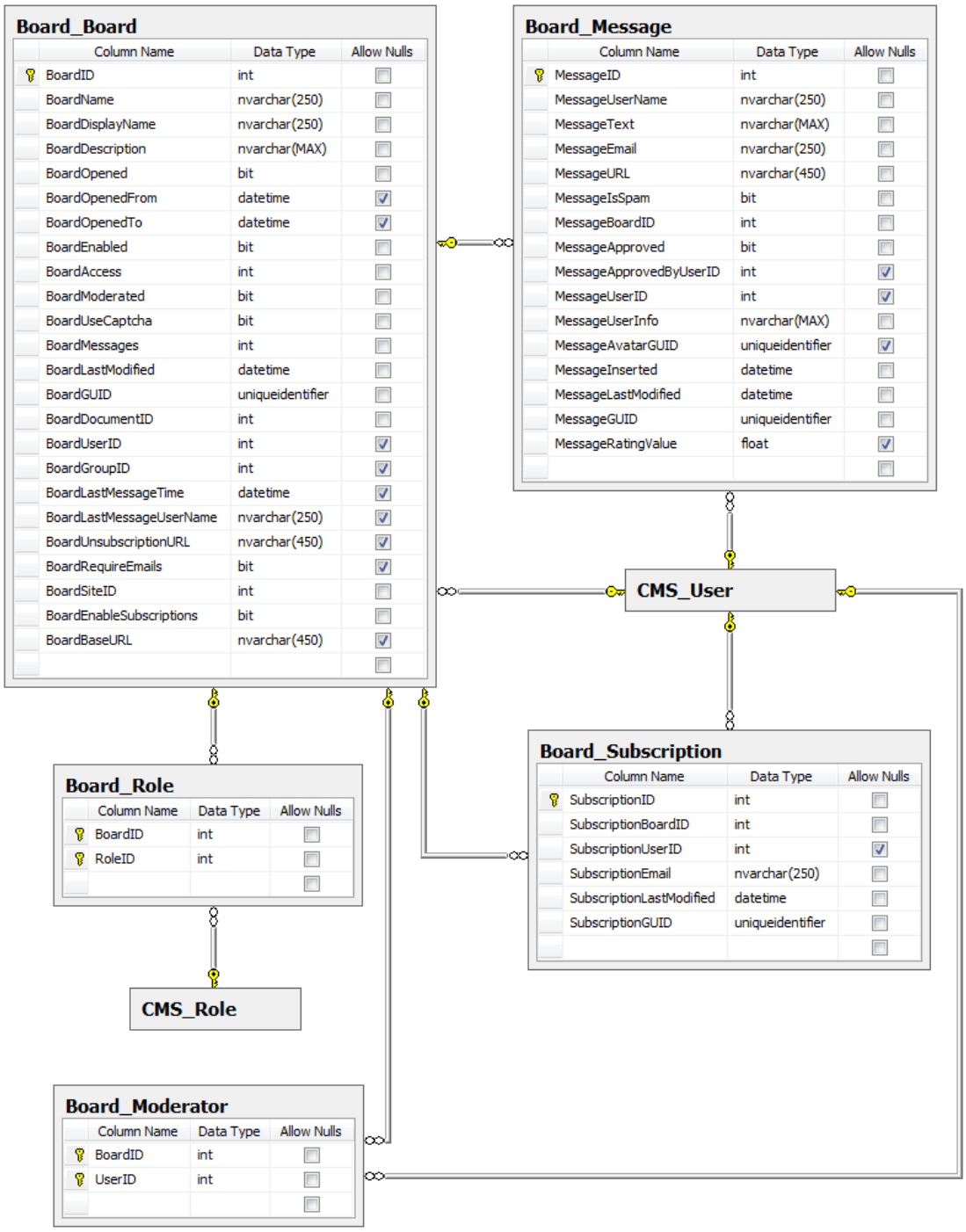
In this chapter, you will learn which database tables and API classes are used in the Message boards module. You will also see the most common API examples.

Advanced API examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all methods from the module classes, please refer to Kentico CMS API Reference.

8.33.9.2 Database tables

The following database tables are used in the Message boards module:

- **Board_Board** - contains records representing message boards.
- **Board_Message** - contains records representing messages.
- **Board_Subscription** - contains records representing message board subscriptions.
- **Board_Role** - contains relationships between roles and message boards indicating that particular roles are allowed to add comments to a particular board.
- **Board_Moderator** - contains relationships between users and message boards indicating that a particular user can moderate messages of a particular board.



8.33.9.3 API classes

If you are not familiar with the database table data management by Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

The Message boards module classes use the **CMS.MessageBoard** namespace.

Board_Board table API:

- **BoardInfo** - represents one message board.
- **BoardBoardInfoProvider** - provides management of message boards.

Board_Message table API:

- **BoardMessageInfo** - represents one message.
- **BoardMessageInfoProvider** - provides management of messages.

Board_Subscription table API:

- **BoardSubscriptionInfo** - represents one message board subscription.
- **BoardSubscriptionInfoProvider** - provides management of message board subscriptions.

Board_Role table API:

- **BoardRoleInfo** - represents a relationship between one role and one message board expressing that a particular role is allowed to add comments to a particular board.
- **BoardRoleInfoProvider** - provides management of relationships between roles and message boards.

Board_Moderator table API:

- **BoardModeratorInfo** - represents a relationship between one user and one message board expressing that a particular user can moderate messages of a particular message board.
- **BoardModeratorInfoProvider** - provides management of relationships between moderators and message boards.

8.33.9.4 API examples

8.33.9.4.1 Overview

Please note

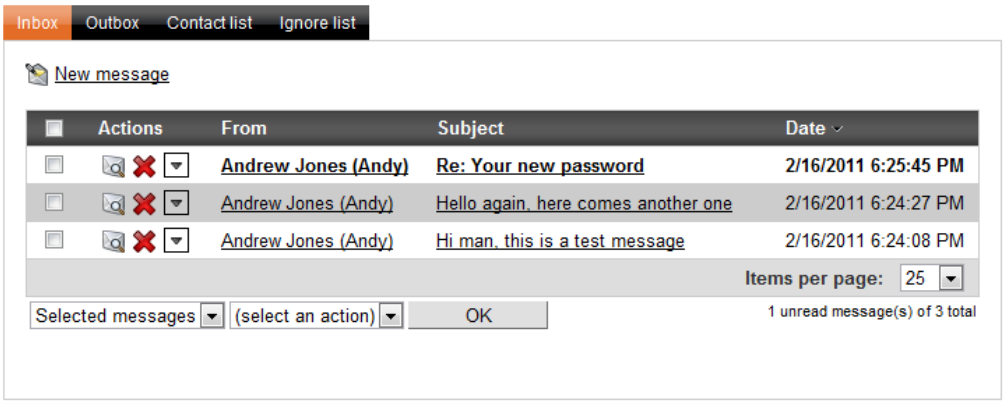
All API examples can be simulated in the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Community\MessageBoards\Default.aspx.cs***.

8.34 Messaging

8.34.1 Overview

The **Messaging** module allows users of your website to communicate via text messages. Its purpose is to provide an internal way of communication with other users of the website, both on the live site and in the user interface.



Users with access to Kentico CMS user interface can send, receive and manage their messages in **CMS Desk -> My desk -> Messages**. Functionality provided by this section of the user interface is described in the My messages topic. If you want to try to send out a testing message, you may proceed to the Sending a new message topic. On the live site, identical functionality can be provided by a single **My messages** web part, or separately by other web parts which come with the module. All Messaging module web parts are described in the Adding the messaging functionality to the live site topic.

It is possible to let Kentico CMS notify users about new messages by means of automatic notification e-mails. Information related to this topic can be found in the E-mail notifications topic. The Security topic explains messaging possibilities of unregistered anonymous website visitors.

The Messaging internals and API sub-chapter provides an overview of database tables and API classes used by the module. It also provides examples of how methods from the classes can be used to handle messaging in your custom code.

There are other ways how you can enable users of your website to communicate with each other. The Forums module enables you to add standard discussion forums to the live site. The Message boards modules provides similar functionality as the forums, with the difference that message boards are not structured and are typically used to let users add comments on the content of a particular page.

8.34.2 My messages

The **CMS Desk -> My desk -> Messages** section of the administration interface is divided into the following four tabs:

Inbox

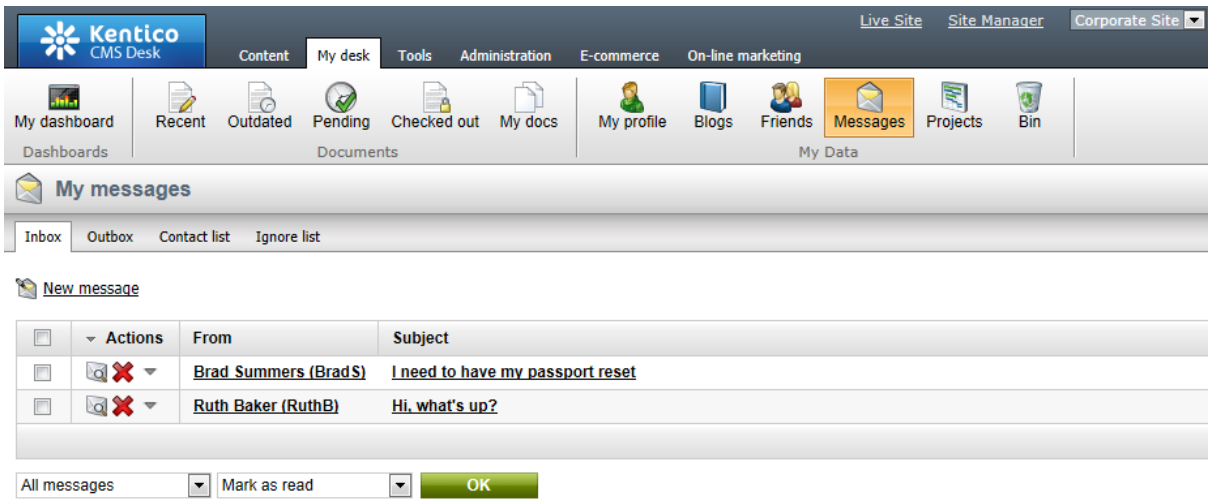
On this tab, the current user can see a list of all their received messages. The following actions are available for each message:

- **View** - displays the message. Alternatively, you can view a message by clicking its **Subject** or its sender in the **From** column.
- **Delete** - deletes the message.

Other actions are available if you unfold the drop-down menu by clicking the icon in a message's row:

- **Reply** - opens a dialog where a reply to the message can be written and sent out.
- **Forward** - opens a dialog where you can forward the message to other users.
- **Mark as read** - marks the message as read (already opened), which is recognizable by a non-bold font in the listing.
- **Mark as unread** - marks the message as unread (not opened yet), which is recognizable by a bold font in the listing.

Certain actions can be performed to more messages at once, as explained in Bulk actions for Inbox and Outbox below.

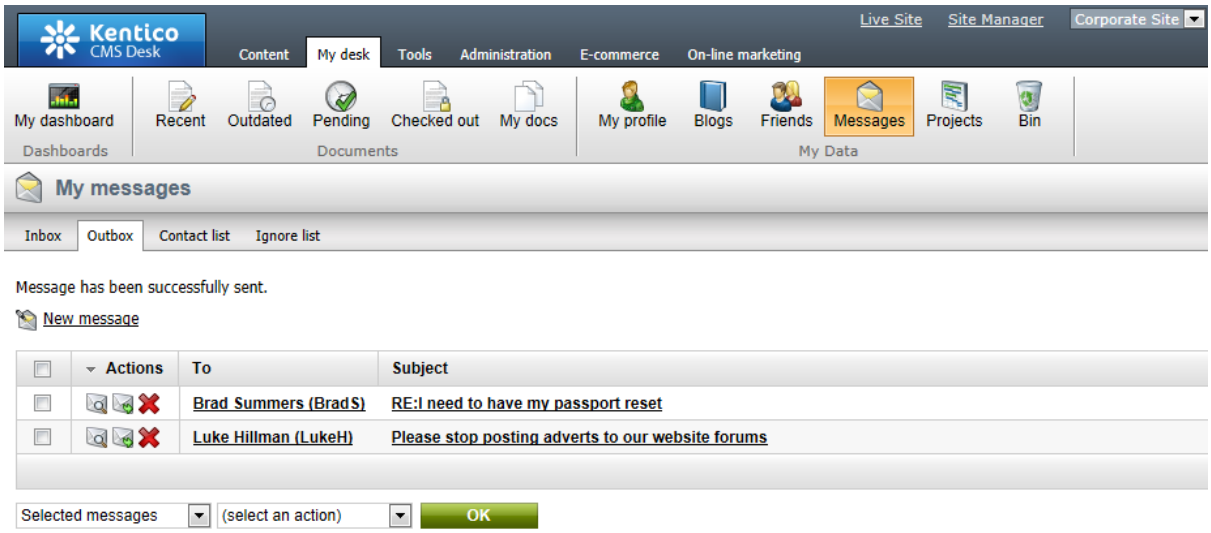


Outbox

This tab lists all messages sent out by the current user. These actions are available for each message on this tab:

- **View** - displays the message. Alternatively, you can view a message by clicking its **Subject** or its recipient in the **To** column.
- **Forward** - opens a dialog where you can forward the message to other users.
- **Delete** - deletes the message.

The Delete action can be performed to more messages at once, as explained in Bulk actions for Inbox and Outbox below.

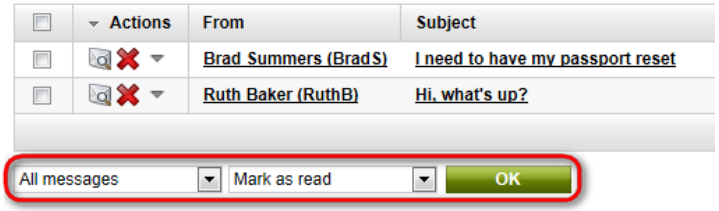


Bulk actions in Inbox and Outbox

In **Inbox** and **Outbox**, you can perform certain actions with multiple messages at once. For this purpose, you need to adjust the two drop-down lists below the listing. In the first one, you can choose from the following options:

- **Selected messages** - the action will only be performed to messages selected by the check-boxes in the respective rows.
- **All messages** - the action will be performed to all listed messages.

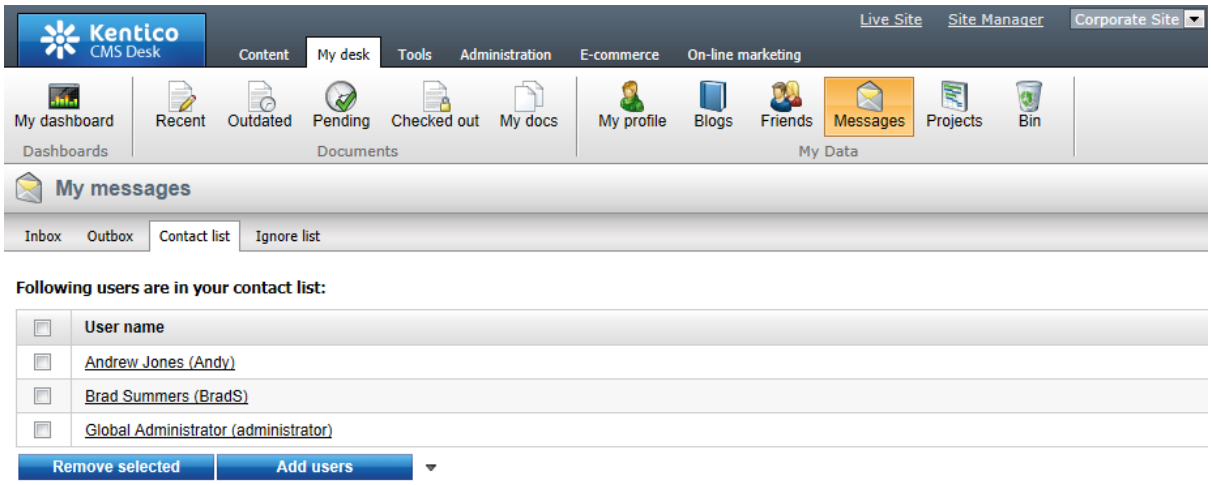
After choosing from the first drop-down list, you need to select the action to be taken from the second one and click **OK**. In **Inbox**, the **Delete**, **Mark as read** and **Mark as unread** actions are available. I **Outbox**, you can only use the **Delete** action for multiple files at once.



Contact list

This tab represents the current user's contact list, i.e. a list of users they communicate with. Having a user in the contact list makes it easier to select the user as a recipient when creating a new message.

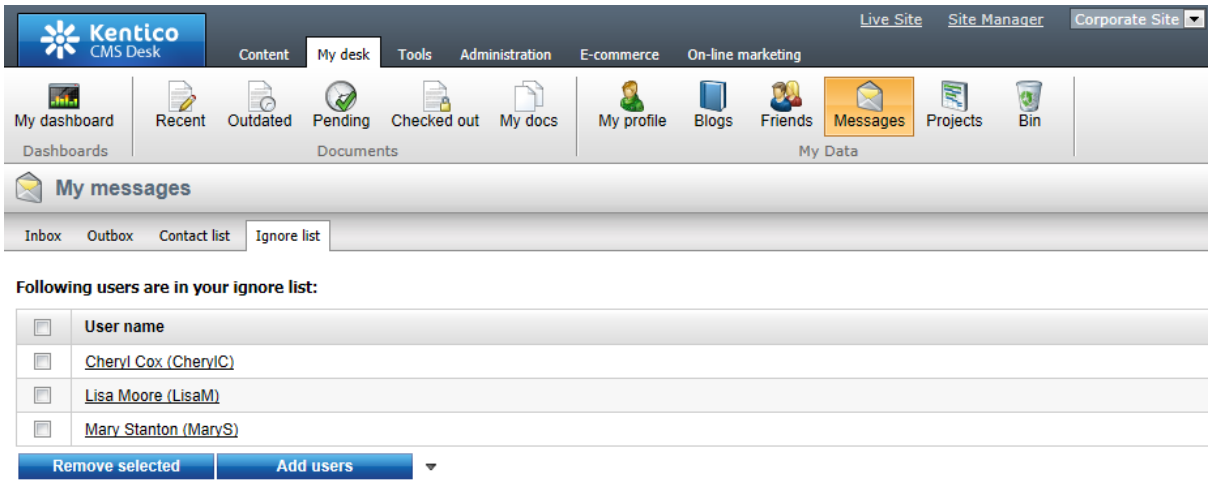
The **Add users** button opens a pop-up dialog where you can select users to be added to the list. The **Remove selected** button removes users selected by the check-boxes from the list.



Ignore list

This tab represents the current user's ignore list, i.e. a list of users from which no messages will be received. If a user who is in your ignore list sends you a message, the message will simply not be delivered.

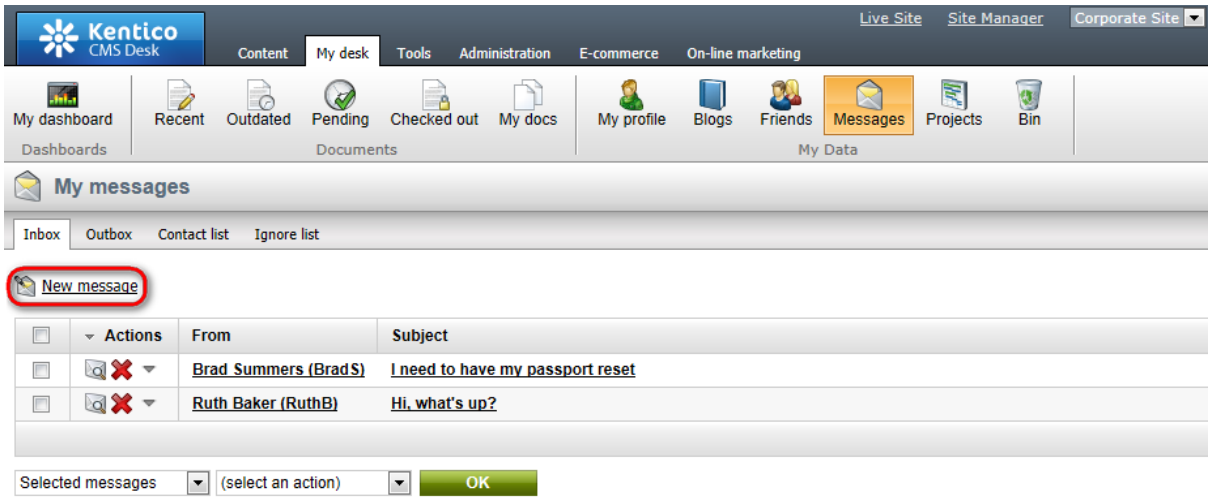
The **Add users** button opens a pop-up dialog where you can select users to be added to the list. The **Remove selected** button removes users selected by the check-boxes from the list.



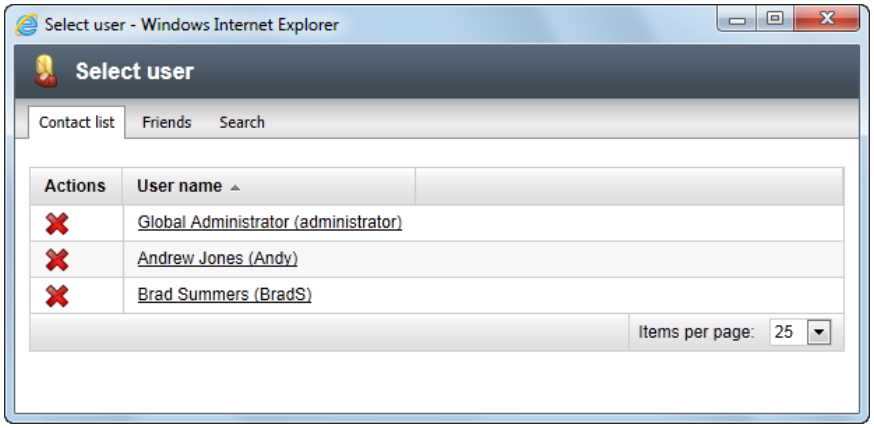
8.34.3 Sending a new message

New messages can be created and sent out in **CMS Desk -> My Desk -> Messages**, on both the **Inbox** and **Outbox** tabs in this part of the UI.

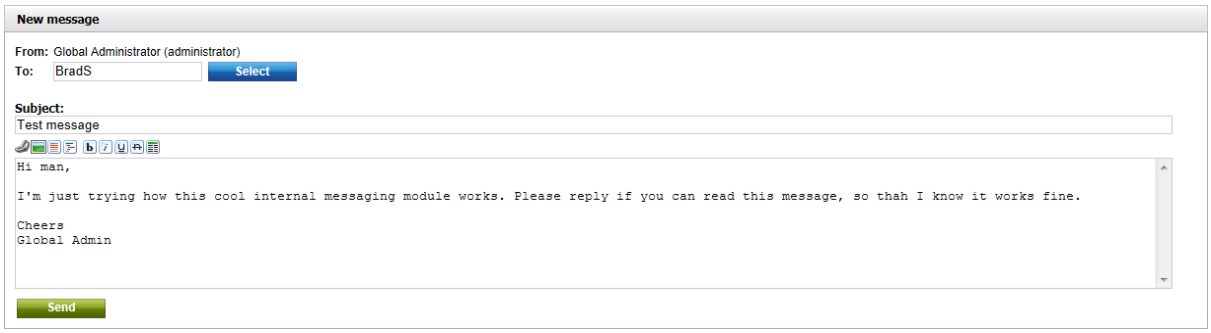
1. On each of these tabs, the first thing to do is to click **New message** above the messages listing.



2. Clicking the link opens the **New message** dialog where new messages can be created and sent out. When creating a message, you first need to select its recipient. By clicking the **Select** button next to the **To** field, you open a pop-up dialog where the recipient can be selected from the contact list, from the your friends, or searched among all visible website users.



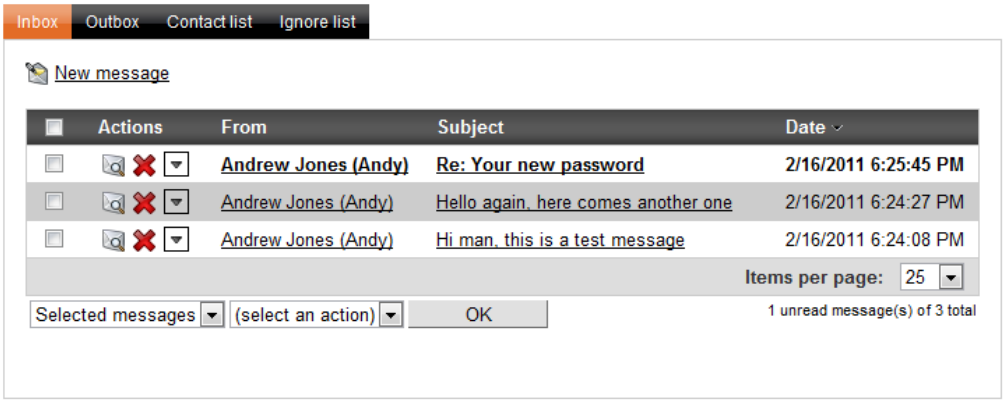
3. Then you need to fill in the **Subject** field and the actual text of the message into the large text area. Entered text can be formatted using BBCode. This is possible either by entering BBCode tags directly, or using the icons above the main text area. Supported BBCode tags are listed and explained in Modules -> Forums -> BBCode support.



4. Finally, the message can be sent out using the **Send** button.

8.34.4 Adding the messaging functionality to the live site

The same functionality that is described in the My messages topic can be added to the live site as well. It is possible using the web parts stored under the **Messaging** web part category. This page gives just a brief overview of the messaging web parts. Detailed descriptions of each web part and its properties can be found in Kentico CMS Web Parts Reference.



The **My messages** web part, shown in the screenshot above, is an all-in-one messaging web part. It provides exactly the same functionality as the user interface in **CMS Desk -> My Desk -> Messages**. This web part encapsulates the rest of the messaging web parts in one web part. If you need to add just a particular part of the messaging functionality,you can use one of the other messaging web parts:

- **Contact list** - allows users to add other users into their contact list or remove them from the list
- **Ignore list** - allows users to define which users they don't want to receive any messages from
- **Inbox** - shows a list of received messages and allows replying to them and deleting them
- **Messaging info panel** - displays links to inbox, outbox, etc.
- **Outbox - s**hows a list of sent messages and allows deleting them
- **Send message** - allows users to send messages

These web parts can be used separately on any page of the website, providing the same functionality as when the **My messages** web part is used. You can also choose which of these web parts will be included in the **My messages** web part. This can be done using its **Display inbox**, **Display outbox**, **Display contact list** and **Display ignore list** properties.

8.34.5 E-mail notifications

Users can be notified about new messages received via the Messaging module by means of notification e-mails. The e-mails are based on the **Messaging - Notification email** e-mail template.

For this to work, the target e-mail address must be filled into a user's **Messaging notification e-mail** field. This can be done several ways:

- On the live site, it can be entered via the **My profile** web part.
- Users with access to the system's user interface can enter it in **CMS Desk -> My desk -> My profile**.
- Global administrators can edit this field of each user in **Site Manager -> Administration -> Users -> edit (✎) a user -> Settings**.

When the e-mail address is entered into the field, a notification e-mail is sent to the address whenever the user receives a new message. This does not apply to messages received from users in the recipient's **Ignore list**.

Please note

Your instance of Kentico CMS must be configured to use an SMTP server in order for e-mails to be sent, as described in Installation and deployment -> Additional configuration tasks -> SMTP server configuration.

Macros in messaging e-mail template

In text of the **Messaging - Notification email** template, you can access the following specific objects and their properties (e.g. *{% Sender.UserName %}*) using context macros to include dynamic values in their text:

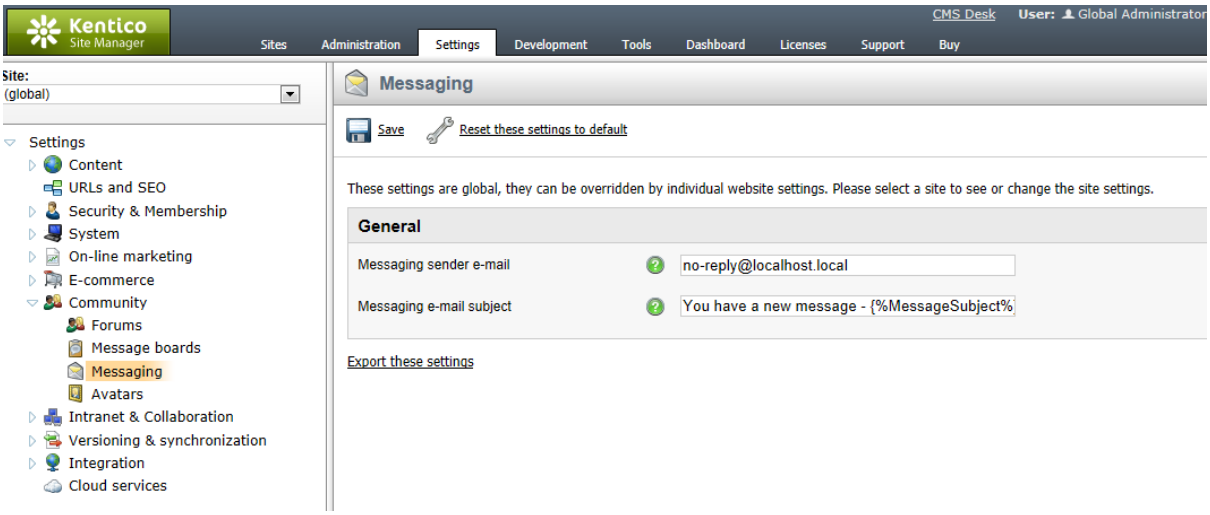
- **{% Sender %}** - *UserInfo* object of the message sender.
- **{% Recipient %}** - *UserInfo* object of the message recipient.
- **{% Message %}** - *MessageInfo* object of the message.

Besides these special ones, you can also use all other standard macro expressions in the templates. See the Development -> Macro expressions chapter of this guide for more information about macro expressions in Kentico CMS.

Related settings

In **Site manager -> Settings -> Community -> Messaging**, you can find the following settings related to the notification e-mails:

- **Messaging sender e-mail** - e-mail address that will be used as the sender address (*From* field) of the notification e-mails.
- **Messaging e-mail subject** - entered text will be used as content of the *Subject* field of notification e-mails.



8.34.6 Security

Even **unregistered anonymous users** can send messages to registered users of your website. This is possible only from the **Send message** web part, the **My messages** web part can be used only by registered users.

To allow this functionality, go to properties of the **Send message** web part and check the **Allow anonymous users** check-box. By checking the **Allow anonymous users to select recipient** check-box below, you can give anonymous users permission to view a list of all registered users and select a user from this list.

8.34.7 Messaging internals and API

8.34.7.1 Overview

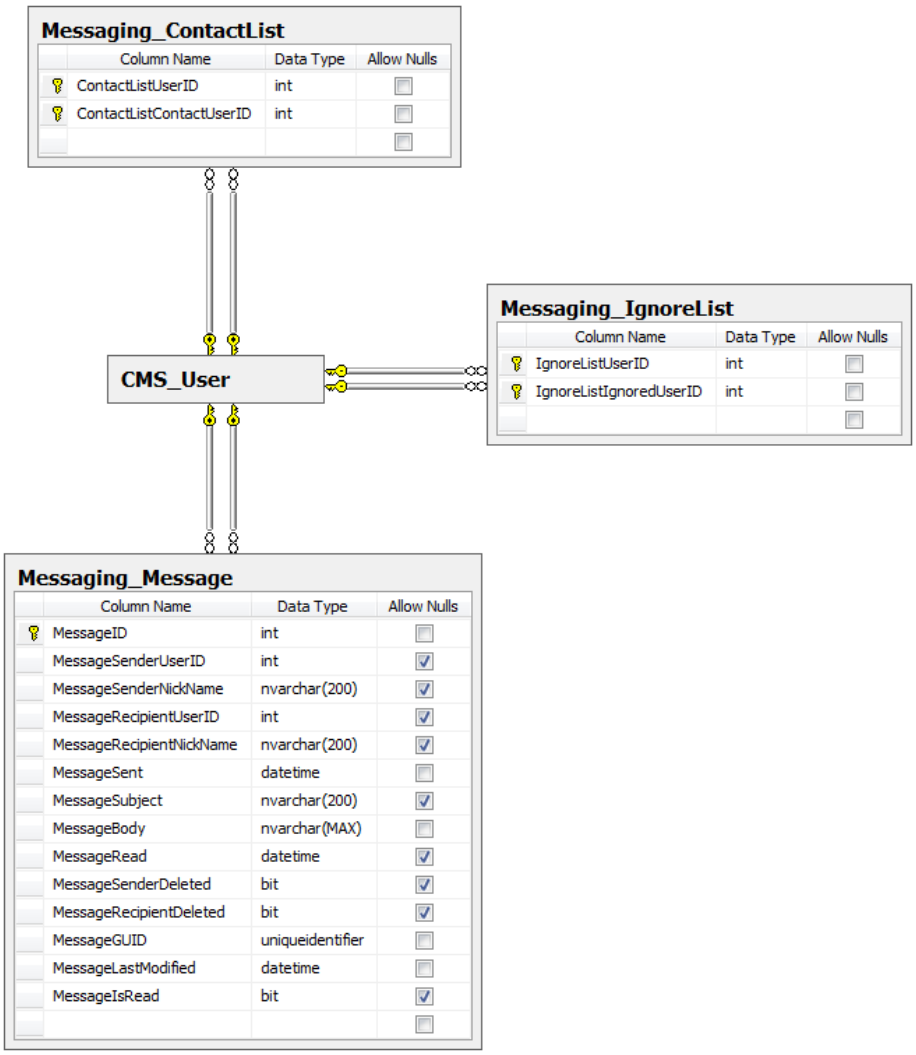
In this chapter, you will learn which database tables and API classes are used in the Messaging module. You will also see the most common API examples.

Advanced API examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all methods from the module classes, please refer to Kentico CMS API Reference.

8.34.7.2 Database tables

The Messaging module uses the following database tables:

- **Messaging_Message** - contains records representing messages.
- **Messaging_ContactList** - contains relationships between pairs of users. Each record indicates that the first user has the second user in their contact list.
- **Messaging_IgnoreList** - contains relationships between pairs of users. Each record indicates that the first user has the second user in their ignore list.



8.34.7.3 API classes

If you are not familiar with database table data management through Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

> **Please note**
>
> The classes of the Messaging module can be found in the **CMS.Messaging** namespace.

Messaging_Message table API:

- **MessageInfo** - represents one message object.
- **MessageInfoProvider** - provides management functionality for abuse reports.

Messaging_ContactList table API:

- **ContactListInfo** - represents one contact list object.
- **ContactListInfoProvider** - provides management functionality for contact lists.

Messaging_IgnoreList table API:

- **IgnoreListInfo** - represents one ignore list object.
- **IgnoreListInfoObject** - provides management functionality for ignore lists.

8.34.7.4 API examples

8.34.7.4.1 Overview

These topics show examples of how the API of the Messaging module can be used:

- Managing messages
- Managing contact lists
- Managing ignore lists

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Community\Messaging\Default.aspx.cs***.

The Messaging API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.GlobalHelper;
using CMS.UIControls;
using CMS.CMSHelper;
using CMS.SiteProvider;
using CMS.Messaging;
```

8.34.7.4.2 Managing messages

The following example creates a message.

```
private bool CreateMessage()
{
    // Create new message object
    MessageInfo newMessage = new MessageInfo();
```

```
    // Set the properties
    newMessage.MessageSubject = "API example message";
    newMessage.MessageBody = "Hello! This is a sample message created by Kentico
CMS API.";

    // Get sender and recipient of the message
    UserInfo sender = CMSContext.CurrentUser;
    UserInfo recipient = UserInfoProvider.GetUserInfo("administrator");

    // Check if both sender and recipient exist
    if ((sender != null) && (recipient != null))
    {
        newMessage.MessageSenderUserID = sender.UserID;
        newMessage.MessageSenderNickName = sender.UserNickName;
        newMessage.MessageRecipientUserID = recipient.UserID;
        newMessage.MessageRecipientNickName = recipient.UserNickName;
        newMessage.MessageSent = DateTime.Now;

        // Save the message
        MessageInfoProvider.SetMessageInfo(newMessage);

        return true;
    }

    return false;
}
```

The following example gets and updates the message created by the code example above.

```
private bool GetAndUpdateMessage()
{
    // Prepare the parameters
    string where = "[MessageSubject] = 'API example message'";

    // Get the data
    DataSet messages = MessageInfoProvider.GetMessages(where, null, 1, null);
    if (!DataHelper.DataSourceIsEmpty(messages))
    {
        // Get the message from the DataSet
        MessageInfo modifyMessage = new MessageInfo(messages.Tables[0].Rows[0]);

        // Update the properties
        modifyMessage.MessageBody = modifyMessage.MessageBody.ToUpper();

        // Save the changes
        MessageInfoProvider.SetMessageInfo(modifyMessage);

        return true;
    }

    return false;
```

```
}
```

The following example gets and bulk updates multiple messages selected from database based on a WHERE condition.

```
private bool GetAndBulkUpdateMessages()
{
    // Prepare the parameters
    string where = "[MessageSubject] = 'API example message'";

    // Get the data
    DataSet messages = MessageInfoProvider.GetMessages(where, null);
    if (!DataHelper.DataSourceIsEmpty(messages))
    {
        // Loop through the individual items
        foreach (DataRow messageDr in messages.Tables[0].Rows)
        {
            // Create object from DataRow
            MessageInfo modifyMessage = new MessageInfo(messageDr);

            // Update the properties
            modifyMessage.MessageBody = modifyMessage.MessageBody.ToLower();

            // Save the changes
            MessageInfoProvider.SetMessageInfo(modifyMessage);
        }

        return true;
    }

    return false;
}
```

The following example deletes all messages created by the first code example on this page.

```
private bool DeleteMessage()
{
    // Prepare the parameters
    string where = "[MessageSubject] = 'API example message'";

    // Get the message
    DataSet messages = MessageInfoProvider.GetMessages(where, null);

    if (!DataHelper.DataSourceIsEmpty(messages))
    {
        foreach (DataRow messageDr in messages.Tables[0].Rows)
        {
            // Create message object from DataRow
            MessageInfo deleteMessage = new MessageInfo(messageDr);
```

```
            // Delete the message
            MessageInfoProvider.DeleteMessageInfo(deleteMessage);
        }

        return true;
    }

    return false;
}
```

8.34.7.4.3 Managing contact lists

The following example adds the **cmseditor** user to the current user's contact list.

```
private bool AddUserToContactList()
{
    // Gets "cmseditor" UserInfo object
    UserInfo user = UserInfoProvider.GetUserInfo("cmseditor");

    if (!ContactListInfoProvider.IsInContactList(CMSContext.CurrentUser.UserID,
user.UserID))
    {
        // Adds "cmseditor" to the current user's contact list
        ContactListInfoProvider.AddToContactList(CMSContext.CurrentUser.UserID,
user.UserID);

        return true;
    }

    return false;
}
```

The following example removes the **cmseditor** user from the current user's contact list.

```
private bool RemoveUserFromContactList()
{
    // Gets "cmseditor" UserInfo object
    UserInfo user = UserInfoProvider.GetUserInfo("cmseditor");

    if (ContactListInfoProvider.IsInContactList(CMSContext.CurrentUser.UserID,
user.UserID))
    {
        // Removes "cmseditor" from the current user's contact list
        ContactListInfoProvider.RemoveFromContactList
(CMSContext.CurrentUser.UserID, user.UserID);

        return true;
    }

    return false;
```

```
}
```

8.34.7.4.4 Managing ignore lists

The following example adds the **cmseditor** user to the current user's ignore list.

```
private bool AddUserToIgnoreList()
{
    // Gets "cmseditor" UserInfo object
    UserInfo user = UserInfoProvider.GetUserInfo("cmseditor");

    if (!IgnoreListInfoProvider.IsInIgnoreList(CMSContext.CurrentUser.UserID,
user.UserID))
    {
        // Adds "cmseditor" to the current user's ignore list
        IgnoreListInfoProvider.AddToIgnoreList(CMSContext.CurrentUser.UserID,
user.UserID);

        return true;
    }

    return false;
}
```

The following example removes the **cmseditor** user from the current user's ignore list.

```
private bool RemoveUserFromIgnoreList()
{
    // Gets "cmseditor" UserInfo object
    UserInfo user = UserInfoProvider.GetUserInfo("cmseditor");

    if (IgnoreListInfoProvider.IsInIgnoreList(CMSContext.CurrentUser.UserID,
user.UserID))
    {
        // Removes "cmseditor" from the current user's ignore list
        IgnoreListInfoProvider.RemoveFromIgnoreList(CMSContext.CurrentUser.UserID,
user.UserID);

        return true;
    }

    return false;
}
```

8.35 Multivariate testing

8.35.1 Overview

For information about **Multivariate testing**, please see the Modules -> Website optimization -> Multivariate testing chapter of this guide.

8.36 Newsletters

8.36.1 Overview

The Newsletters module allows the creation and management of highly configurable newsletters. Newsletters are used to regularly send e-mails about a certain topic to users who agree to receive them by subscribing.

When a newsletter issue is sent, e-mails are created and personalized (if applicable) for every individual subscriber. This procedure is performed by the Newsletter queue and any e-mails lost due to errors are automatically re-sent. This is handled by the **Send queued newsletters** scheduled task. Once this process is complete, the e-mails are sent either directly to the SMTP server or to the E-mail queue if the given newsletter is configured to do so (later topics in this chapter describe how this can be done). Using the e-mail queue is recommended for newsletters with large amounts of subscribers to ensure that all e-mails are delivered correctly.

The newsletters can be of two types:

- Static newsletters - every issue is edited and sent manually. The newsletters are based on predefined templates.
- Dynamic newsletters - issues are sent out to all subscribers automatically at a specified interval. The content is dynamically taken from a specified page, which is usually updated between newsletter issues.

Users can manage their subscriptions and find information about newsletters directly on the pages of a website if it contains the appropriate web parts. Please refer to the Integrating newsletters into the site topic to see what web parts are available.

Newsletters make use of templates to give all issues and related notification e-mails a unified appearance by utilizing design elements such as a company logo or footer. Templates may also contain various macros that are resolved into data matching individual recipients during the mail merge. For more details, please see the Newsletter templates topic.

The subscribers of all newsletters are stored in the system and can be monitored and managed as shown in the Subscriber management topic. The Subscriber import and export topic describes how tasks affecting large amounts of subscribers can be performed.

Newsletters also offer several tools that help track, monitor and optimize issues, which can be very useful when carrying out e-mail marketing campaigns. More information about this is given in the On-line marketing chapter.

The Troubleshooting topic contains a list of solutions that can be used to resolve common problems with newsletter e-mails.

8.36.2 Creating a static newsletter

In this topic, you will learn how to create a static (template-based) newsletter:

1. Go to **CMS Desk -> Tools -> Newsletters** and click **New newsletter**. Enter the following details:

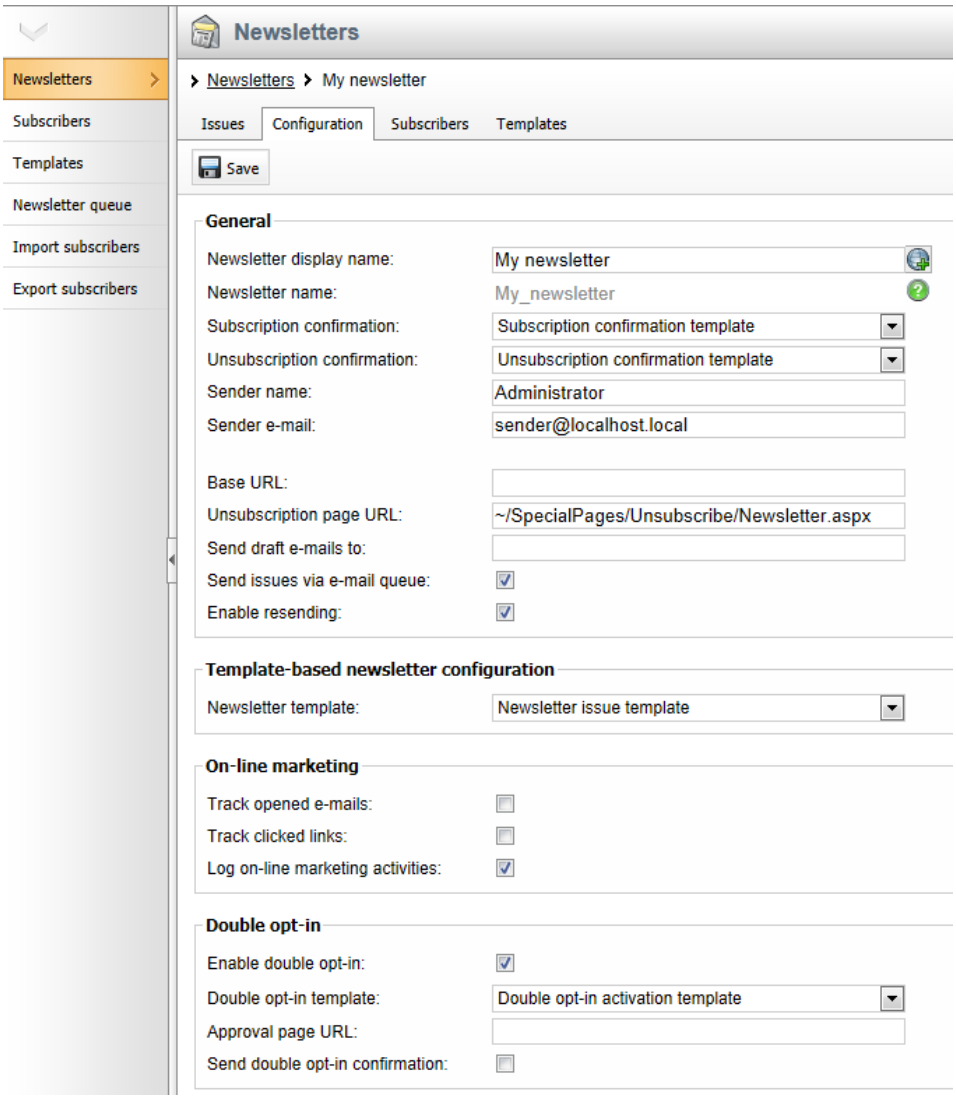
- **Newsletter display name**: *My newsletter*; this is the name displayed to users.
- **Newsletter name**: leave the *(automatic)* option; this name is used internally as an identifier in web part properties, URLs and the API. The system will generate an appropriate one based on the display name.
- **Subscription confirmation**: Subscription confirmation template
- **Unsubscription confirmation**: Unsubscription confirmation template
- **Sender name**: Enter your full name
- **Sender e-mail**: Enter your e-mail address

Choose the **Template-based newsletter** option and use the *Newsletter issue template***.** Click **Save**. The newsletter will be created and you will be redirected to its **Configuration** tab, where the following additional options can now be set:

- **Base URL** - here you can specify the base URL of your website (protocol, domain name and virtual directory), which is used to convert relative links to absolute URLs in newsletter issues. It may be necessary to set this property in order for the unsubscription links to work properly. It's also useful if you encounter any issues with links in newsletter e-mails, e.g. if you are using a different URL for your editing environment than for the live website.
  Example: *https://www.example.com*
- **Unsubscription page URL** - enter *~/SpecialPages/Unsubscribe/Newsletter.aspx*. This page on the sample Corporate site contains a Newsletter unsubscription web part, which ensures the required functionality.
- **Send draft e-mails to** - the addresses specified here are pre-entered by default when sending draft newsletter issues for testing purposes. Multiple addresses must be separated by semicolons. Draft e-mails are not included in tracking statistics (e-mail opening and link clicking).
- **Send issues via e-mail queue** - if enabled, newsletter issues will be sent to the SMTP server through the E-mail queue. This is recommended for newsletters with a large number of recipients. If disabled, the issues will be sent directly to the SMTP server.
- **Enable resending** - if enabled, it will be possible to edit newsletter issues and manually send them again even after they have been mailed out to subscribers. You can disable this property to ensure that subscribers do not accidentally receive the same content multiple times. Issues that utilize A/B testing are an exception and can never be re-sent.

- **On-line marketing** - the properties in this section are related to tracking of the newsletter's e-mails and keeping marketing statistics. Please see the On-line marketing chapter for further details.

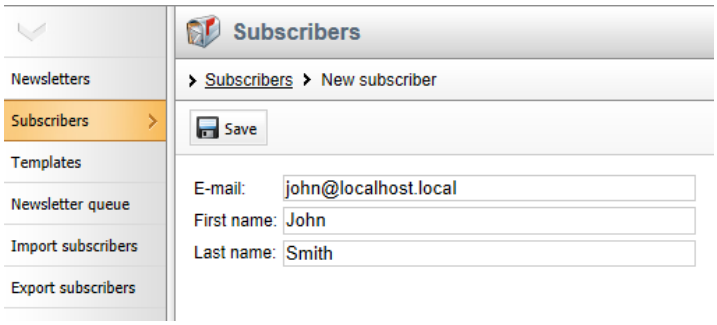
- **Double opt-in** - the properties in this section are related to double opt-in subscription. Please refer to the Double opt-in topic for more information.



Click **Save** again.

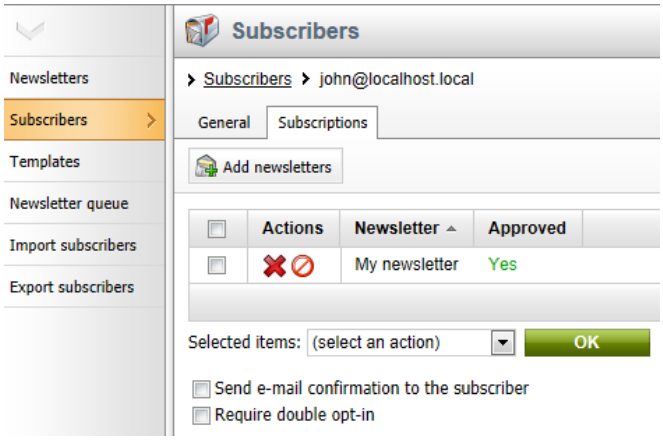
2. Now we will create a new subscriber. Select the main **Subscribers** tab in the left menu and click the **New subscriber** button. Enter the following details:

- **E-mail** - subscriber's e-mail address. If you wish to check the newsletter e-mail once it is sent, enter a valid address that you can access.
- **First name** - subscriber's first name.
- **Last name** - subscriber's last name.



Click **Save** to add the subscriber.

3. Now we will assign this subscriber to our previously created newsletter. Switch to the **Subscriptions** tab, click the **Add newsletters** button, check the box next to **My newsletter** in the selection dialog and click **OK**.



If you check the **Send e-mail confirmation to the subscriber** box before adding a newsletter to a subscriber, an e-mail informing about the subscription will be sent to them. If you check the **Require double opt-in** box, subscriptions to newsletters that have double opt-in enabled will be inactive (rejected) until the user confirms them or the **Approve** (✔) action is manually selected in the list of subscriptions.

4. Your newsletter is configured. You can now create new issues as is described in the Authoring static newsletter issues topic.

8.36.3 Authoring static newsletter issues

Now that you have created a static newsletter with a subscriber as described in the Creating a static newsletter topic, it is possible to write and send individual issues. The following steps describe how this can be done:

1. Go to the **Newsletters** tab in the left menu and edit (✎) **My newsletter**. Click **Create new issue** on the **Issues** tab. A wizard will guide you through the process of creating a new newsletter issue.

2. In Step 1, start by entering the following **Subject**: *Welcome to issue #1 of My newsletter*

3. Then copy the following text into the **content** editable region:

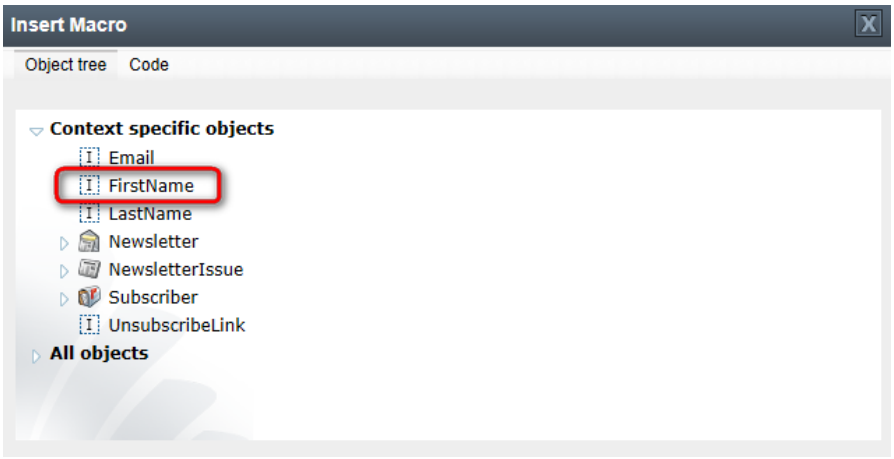
Dear ,

welcome to the first issue.

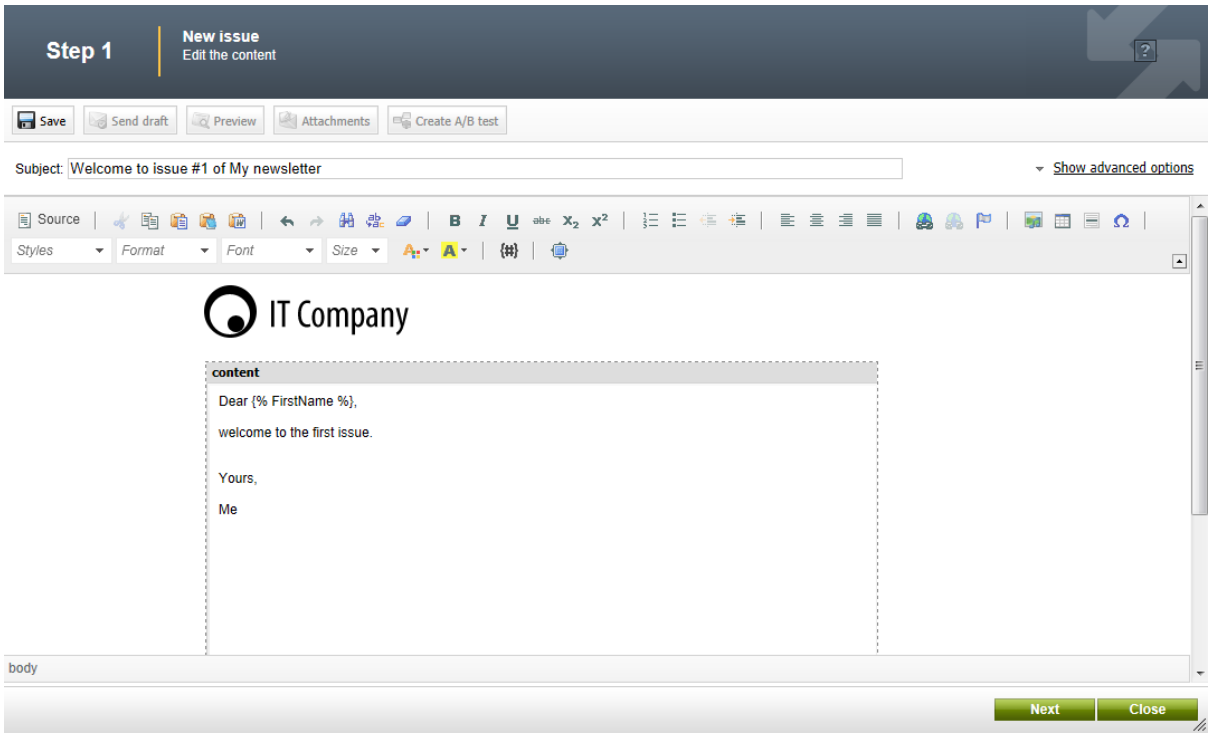
Yours,

Me

Place the cursor after the word *Dear* and click the **Insert macro** ({#}) button on the toolbar of the WYSIWYG editor. In the displayed dialog, choose the **FirstName** option under the **Context specific objects** section.



The *{% FirstName %}* macro expression will be placed into the text. This macro will automatically be replaced by the e-mail recipient's first name during the mail merge. The issue should now look like this:



Newsletter templates

The overall structure of this issue is defined by the *Newsletter issue template*, which can be edited under the **Templates** section of the left menu. Templates are described in the Newsletter templates topic.

Some newsletters may have multiple issue templates assigned. In this case, there will be an additional step in the wizard where you can choose the template that you wish to use for the new issue.

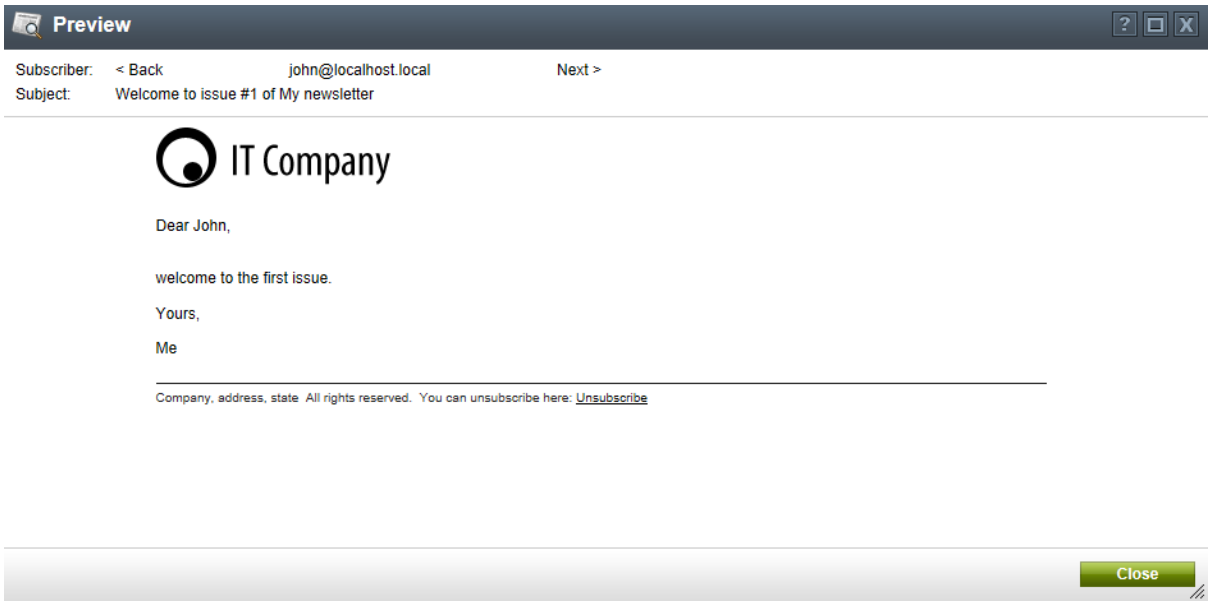
It is also possible to change the template by clicking **Show advanced properties** next to the **Subject** field and selecting a different option through the **Template** selector.

4. Click the **Save** button at the top. Once the issue is saved, you can manage it using the actions described below:

- **Send draft** - allows you to send a test draft of the newsletter issue to any e-mail address. Draft e-mails do not correctly resolve newsletter macros (such as unsubscription links), since the recipients are not linked to subscribers in the system.
- **Preview** - opens a dialog window where you can view how the issue will appear to specific subscribers.
- **Attachments** - allows you to attach files to the issue, such as images, an event agenda, white papers, etc. The attachments will be included in the issue e-mails when they are sent out. Please note that other attachments may already be included in the newsletter template on which the issue is based.
- **Create A/B test** - may be used to prepare an A/B test for the issue. This allows you to create

several different versions of the issue, evaluate them on a test group of subscribers and then send out the most successful one. Clicking this button adds the first testing variant of the issue. Please see the On-line marketing -> A/B testing topic to learn more.

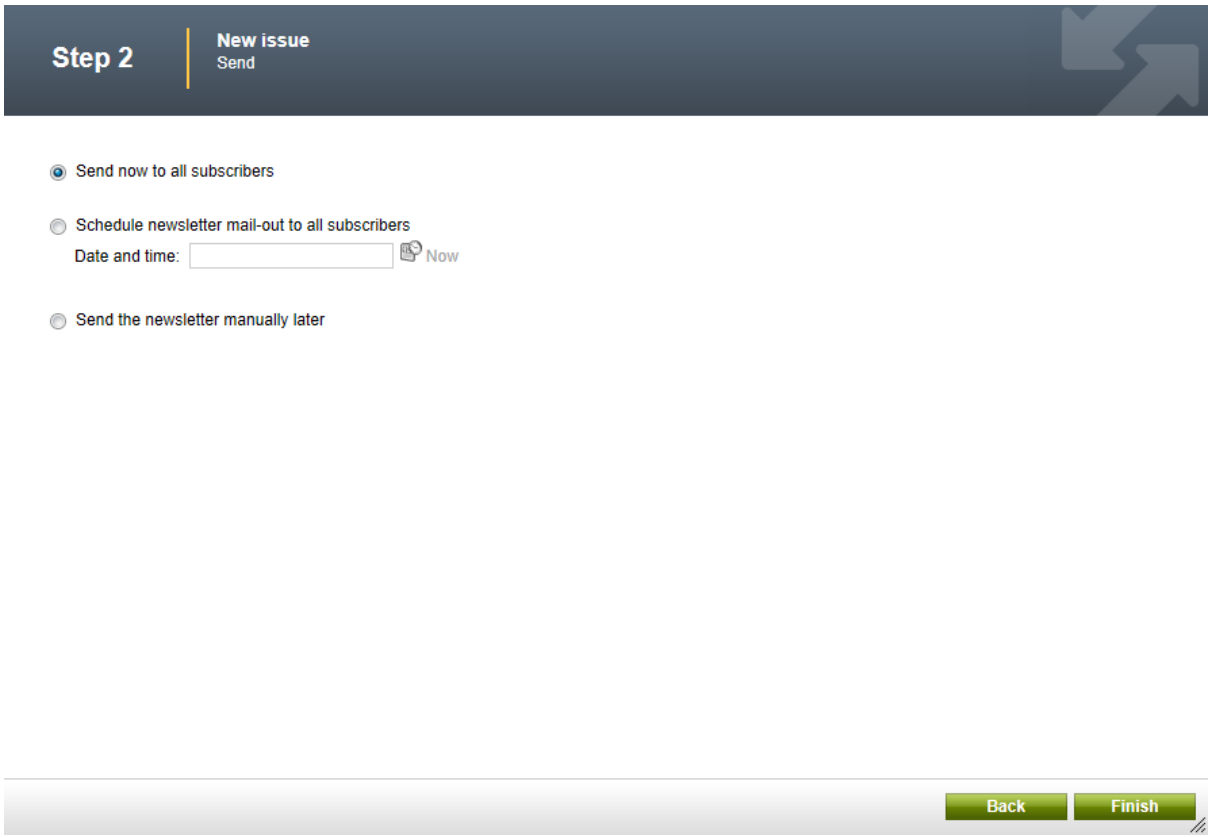
Try clicking the **Preview** button. This will open a dialog where you can see how the content of the issue will appear for each subscriber.



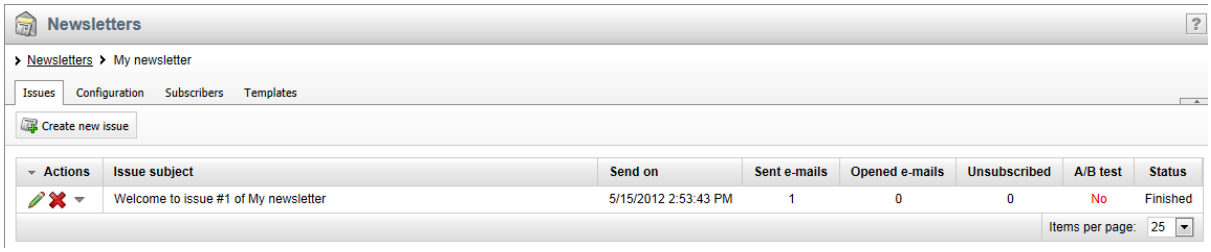
Close the preview dialog.

5. Now click **Next**. This automatically saves the content of the issue and moves you to the final step of the wizard. Here you can choose when the newsletter issue should be sent out:

- **Send now to all subscribers** - the newsletter issue is sent out immediately to all subscribers.
- **Schedule newsletter mail-out** - the issue will be sent out on the specified date and time.
- **Send the newsletter manually later** - the issue will not be sent for now and you can decide on the mail-out time later.



Select **Send now to all subscribers** and click **Finish**. On the **Issues** tab, you can review the statistics and details of the newsletter's issues.



Issues that are scheduled to be sent out in the future can be modified by clicking the **Edit** ( ) action. This allows you to work with their content and other settings.

6. If you experience problems receiving the newsletter issue e-mail, please follow the instructions in the Troubleshooting topic.

8.36.4 Creating a dynamic newsletter

Dynamic newsletters are based the content of a specified page and they are sent out automatically on a regular basis using the built-in scheduling system. When the mailout of a dynamic newsletter is scheduled, a new scheduled task called *Send dynamic newsletter: <newsletter name>* is created for the current site, which reads the content of the given source page and ensures that the issues are sent out according to the set time interval.

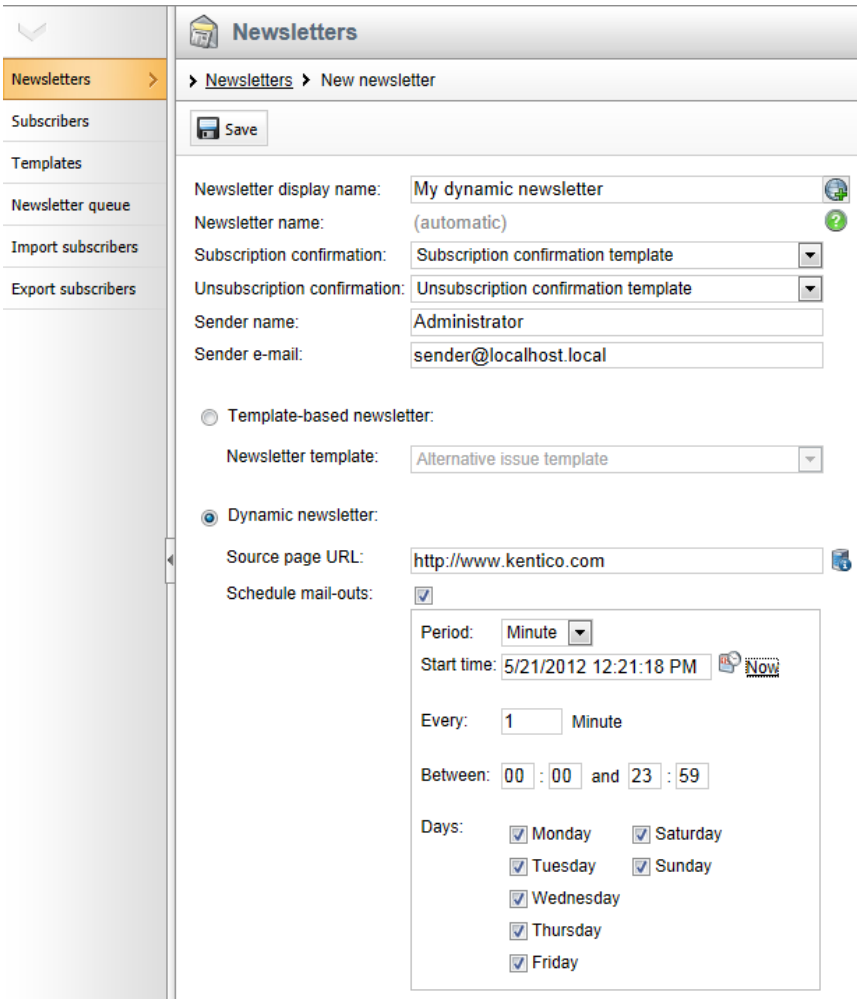
The following steps will guide you through the creation of a dynamic newsletter:

1. Go to **CMS Desk -> Tools -> Newsletters** and click **New newsletter**. Enter the following details:

- **Newsletter display name**: My dynamic newsletter
- **Newsletter name**: leave the *(automatic)* option; the system will generate an appropriate one based on the display name.
- **Subscription confirmation**: Subscription confirmation template
- **Unsubscription confirmation**: Unsubscription confirmation template
- **Sender name**: Enter your full name
- **Sender e-mail**: Enter your e-mail address

Choose **Dynamic newsletter** and enter the following details:

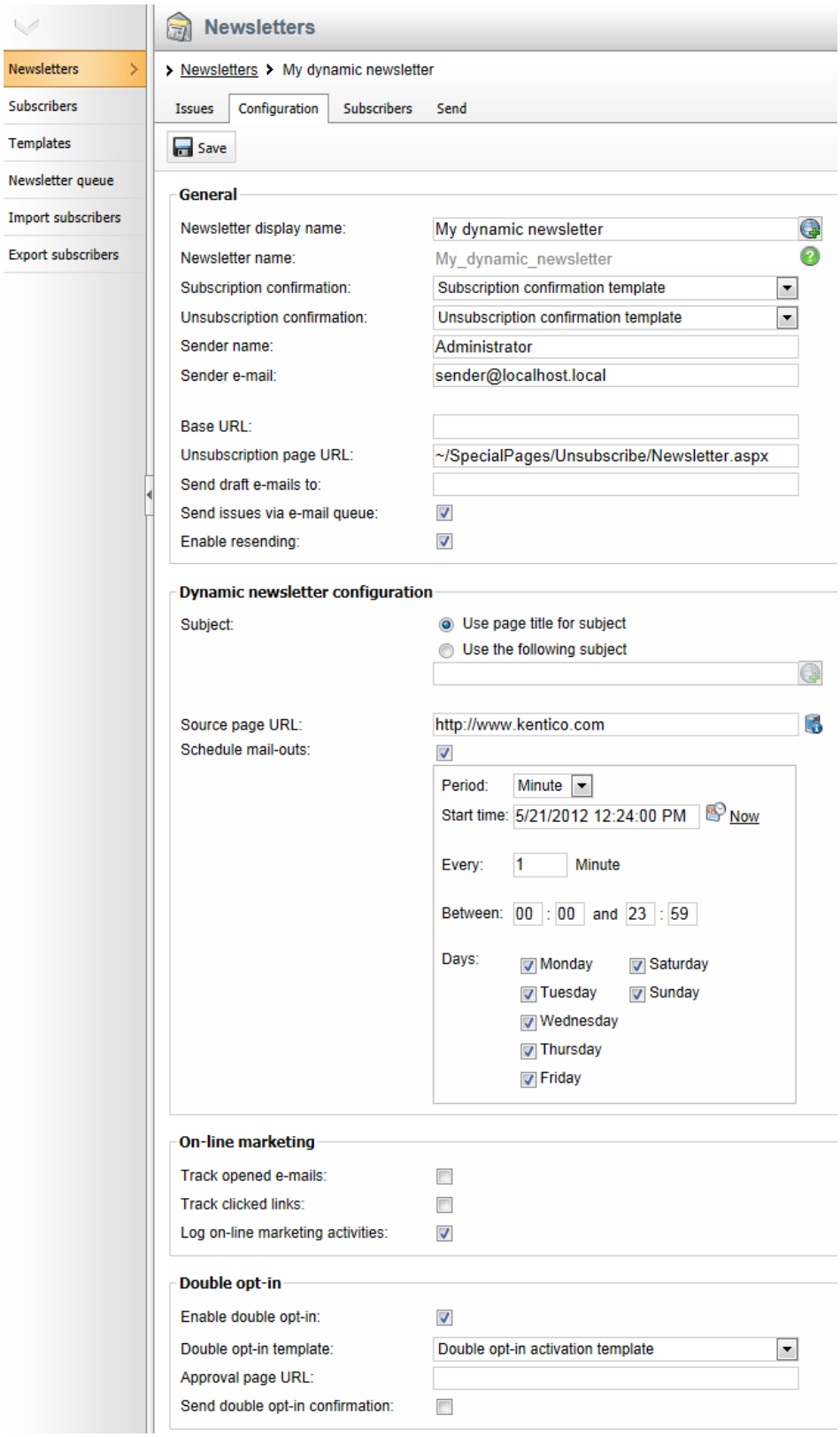
- **Source page URL**: *http://www.kentico.com/*; this is the URL of the page from which the newsletter will take its content.
- **Schedule mail-outs**: Enabled
  - **Period**: Minute
  - **Start time**: Use the date-time picker to select the current date and time (click **Now**)
  - **Every**: 1 minute
  - **Between**: 00:00 and 23:59
  - **Days**: Check all days



Click **Save**. The newsletter will be created and you will be redirected to its **Configuration** tab, where the following additional options can now be set:

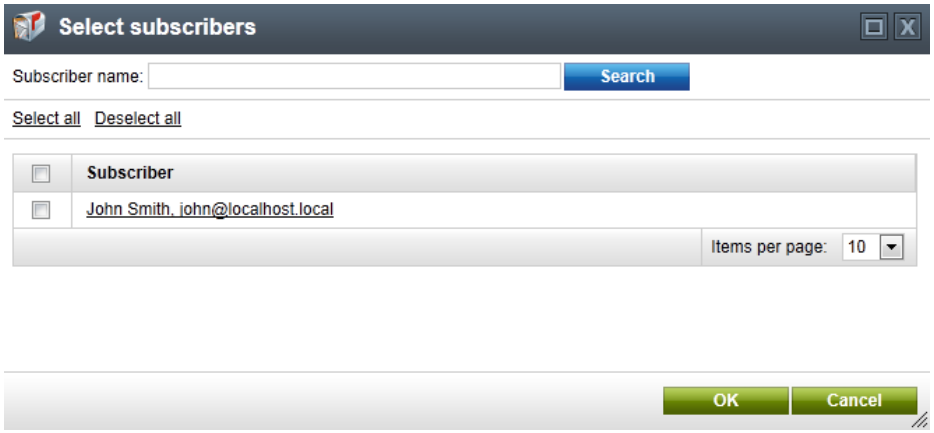
- **Base URL** - here you can specify the base URL of your website (protocol, domain name and virtual directory), which is used to convert relative links to absolute URLs in newsletter issues. It may be necessary to set this property in order for the unsubscription links to work properly. It's also useful if you encounter any issues with links in newsletter e-mails, e.g. if you are using a different URL for your editing environment than for the live website.
  Example: *https://www.example.com*
- **Unsubscription page URL** - enter *~/SpecialPages/Unsubscribe/Newsletter.aspx*. This page on the sample Corporate site contains a Newsletter unsubscription web part, which ensures the required functionality.
- **Send draft e-mails to** - the addresses specified here are pre-entered by default when sending draft newsletter issues for testing purposes. Multiple addresses must be separated by semicolons. Draft e-mails are not included in tracking statistics (e-mail opening and link clicking).
- **Send issues via e-mail queue** - if enabled, newsletter issues will be sent to the SMTP server through the E-mail queue. This is recommended for newsletters with a large number of recipients. If disabled, the issues will be sent directly to the SMTP server.
- **Enable resending** - if enabled, it will be possible to manually send the newsletter's issues again even after they have been mailed out to subscribers. You can disable this property to ensure that subscribers do not accidentally receive the same content multiple times.

- **Subject** - sets the subject of the dynamic newsletter e-mails. It can either use the page title of the content, or be entered manually by selecting *Use the following subject*.
- **On-line marketing** - the properties in this section are related to tracking of the newsletter's e-mails and keeping marketing statistics. Please see the On-line marketing chapter for further details.
- **Double opt-in** - the properties in this section are related to double opt-in subscription. Please refer to the Double opt-in topic for more information.

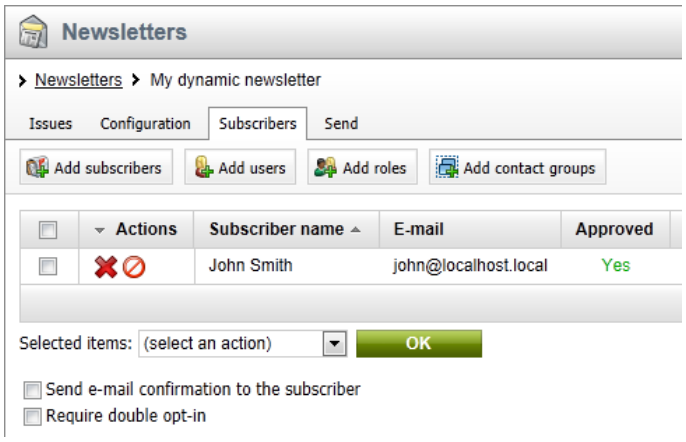


Click **Save** again.

2. Now go to the **Subscribers** tab of this newsletter and add the subscriber created in the Creating a static newsletter topic to this newsletter by clicking **Add subscribers**. The following dialog will appear:

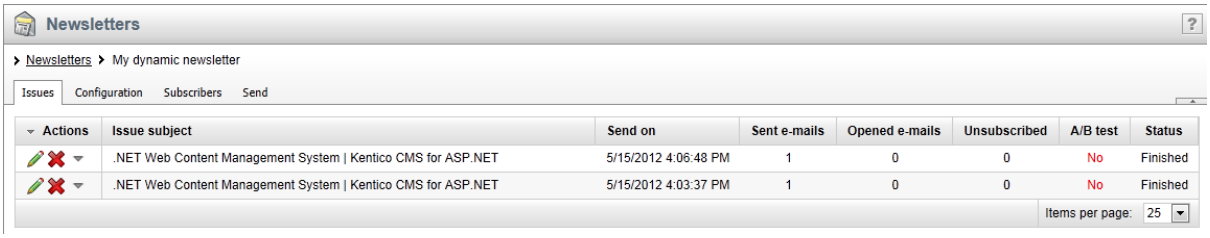


This dialog can be used to select from existing subscribers of all newsletters. Make the selection and click **OK**. Alternatively, the **Add users**, **Add roles** or **Add contact group** buttons may be used to select from the users, roles or on-line marketing contact groups defined for the current website.



If you add a subscriber with the **Send e-mail confirmation to the subscriber** box checked, a notification e-mail will be sent to the selected users, informing them about the changes in their subscriptions. If you check the **Require double opt-in** box (only available if the newsletter has double opt-in enabled), the added subscriptions will be inactive (rejected) until the users confirm them or the **Approve** (✔) action is manually selected in the list of subscriptions.

3. Go to the **Issues** tab. Here, you will see a list of sent issues. You may need to wait up to 2 minutes until the first issue is sent out. You can refresh the list by clicking the **Issues** tab again.



4. Check your mail box, you should receive the content of the given page by e-mail:

This way, you can send out any page from your website. You can create a new page specifically for the purposes of a dynamic newsletter that will display e.g. new articles added to your website during the last month. Please note that some e-mail clients may not be able to process and display pages exactly like a standard browser. It may be necessary to simplify the source page's content or adjust its CSS stylesheet.

Dynamic newsletter unsubscription

Because the content of a dynamic newsletter is loaded from a web page, it may be difficult to include a standard unsubscription link. To allow users to easily cancel their

subscription, you can instead add the Unsubscription request web part onto the newsletter's source page.

Subscribers will then be able to enter their address and receive a special e-mail with an appropriate unsubscription link.

Blocking dynamic newsletter mail-out

If you want to block the mail-out of the page (e.g. if there are no new articles), you can either disable the **Schedule mail-outs** property of the dynamic newsletter on its **Configuration** tab or you can set the title of the source page to ##DONOTSEND## and the newsletter will not be sent. The title of a page can easily be changed by selecting the corresponding document from the content tree in **CMS Desk -> Content -> Edit** and editing the **Page title** field on the **Properties -> Metadata** tab.

8.36.5 Double opt-in

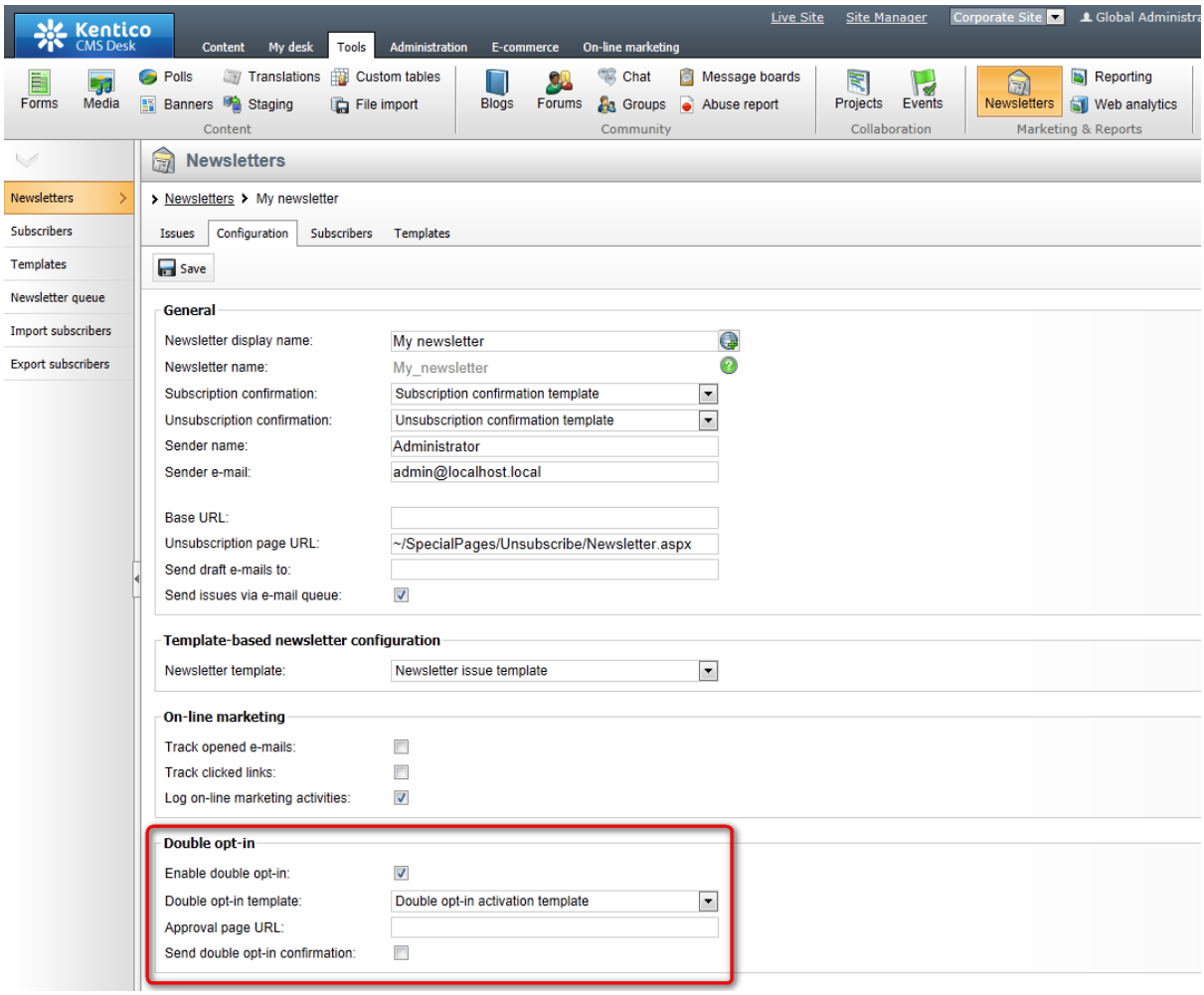
Double opt-in functionality, also referred to as confirmed opt-in, provides an additional security layer to newsletter subscription. It confirms that the entered e-mail address actually exists and prevents users from being unknowingly subscribed to a newsletter by someone else, either intentionally or by mistake.

When double opt-in is disabled for a newsletter, anyone may simply submit an e-mail address for subscription, without any steps being taken to ensure that the address actually belongs to that user. With double opt-in, new subscribers are not immediately activated and do not receive newsletter issues. Instead, an automatic e-mail is sent to them, containing an activation link. Upon clicking the link, users are redirected to a special page and their subscription is confirmed.

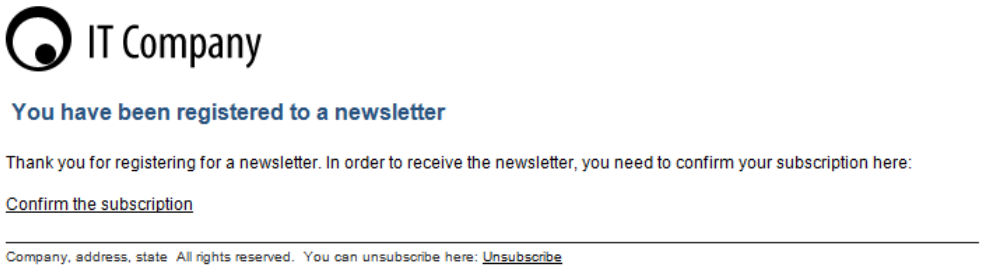
Enabling double opt-in

To configure a newsletter to use double opt-in, open the **CMS Desk -> Tools -> Newsletters** interface, edit () the appropriate newsletter and switch to its **Configuration** tab. Now check the **Enable double opt-in** option in the bottom section of properties and set the remaining ones below it:

- **Double opt-in template** - selects the template used for the subscription activation e-mails that are sent to users. Only templates of the *Double opt-in* type may be selected. Please see the Newsletter templates topic to learn about creating templates.
- **Approval page URL** - sets the URL of the page where users can confirm their subscription to the newsletter. The *Subscription approval* web part must be placed on the specified page to ensure the required functionality. This URL is used by the *Activation link* field, which is typically inserted into the Double opt-in template. If left empty, the value of the *Site Manager -> Settings -> On-line marketing -> Newsletters -> Newsletter double opt-in approval page URL* field is used. If this setting is empty as well, the default system page is used.
- **Send double opt-in confirmation** - if checked, a confirmation e-mail will be sent to users after they successfully activate their subscription.



Click the **Save** button to confirm any changes. Now any users who subscribe to the newsletter will receive an e-mail similar to the following (depending on the used double opt-in template):



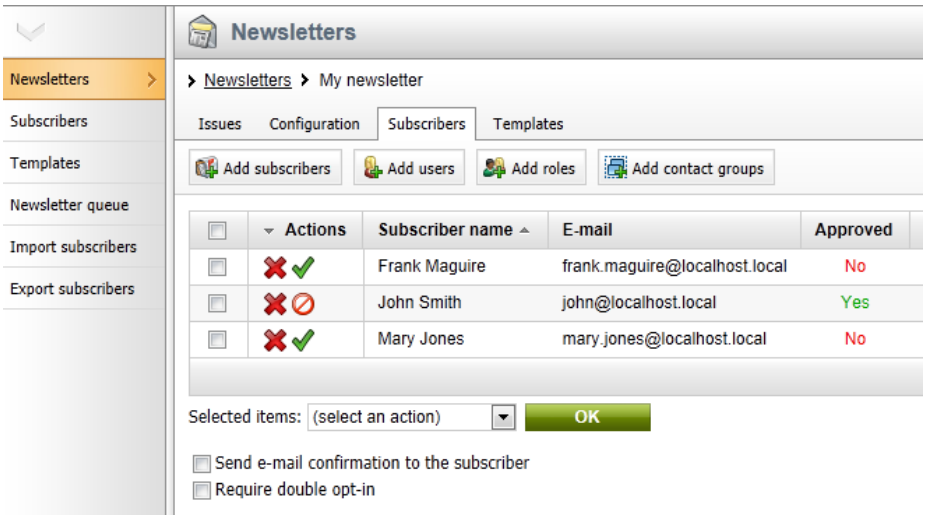
When a user clicks the **Confirm the subscription** link, they will be redirected to the default **~/CMSModules/Newsletters/CMSPages/SubscriptionApproval.aspx** page, where their subscription will be activated and a basic confirmation message will be displayed. The identifier of the exact subscription is passed to the page as a parameter in the query string of the URL in the activation link.

Approval will only be possible for a limited amount of time after the initial subscription (12 hours by default). You can set the length of this time interval for all newsletters through the **Site Manager -> Settings -> On-line marketing -> Newsletters -> Double-opt in interval** setting. If a user does not

activate their subscription within the specified number of hours, the link in the confirmation e-mail will expire, and the user will have to subscribe again.

If you wish to use a custom activation page, simply create a new page on your website according to the specific requirements and place the Subscription approval web part on it. Remember to enter the URL of this page into the **Approval page URL** property of individual newsletters, or in the **Newsletter double opt-in approval page URL** field of the website settings in **Site Manager -> Settings -> On-line marketing -> Newsletters** to assign the page to all newsletters that do not have an approval page specified.

Alternatively, newsletter administrators may also activate subscriptions manually using the **Approve** ( ) action or **Reject** ( ) existing ones.



The Subscriber management topic provides more information about this subject.

8.36.6 Integrating newsletters into the site

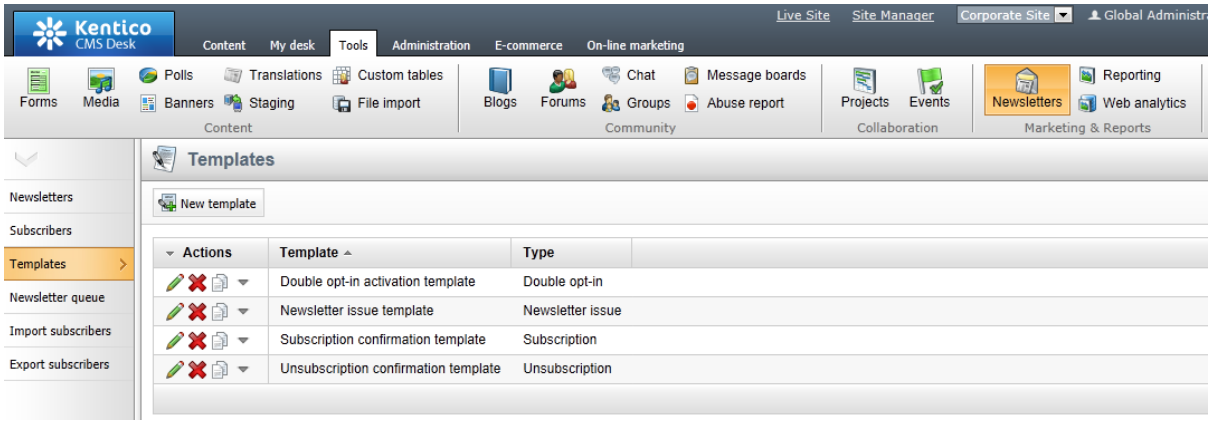
You can integrate newsletters into your website using the following web parts:

- Newsletter subscription - displays a dialog that may be used by the website's users to subscribe to newsletters.
- Custom subscription form - provides a newsletter subscription form like the *Newsletter subscription* web part, but with the option of using a different layout and appearance. The customized subscription form must be defined as an alternative form of the *Newsletter - Subscriber* system table, which can be edited in *Site Manager -> Development -> System tables*.
- Newsletter unsubscription - unsubscribes users from a newsletter and displays a confirmation message. Users are typically directed to the page containing this web part through the "**Unsubscribe**" link in newsletter issues. Identifiers of the exact newsletter and subscriber are read from parameters passed in the query string of the URL.
- Newsletter archive - displays archived issues of a specified newsletter.
- Unsubscription request - this web part allows newsletter subscribers to request an unsubscription e-mail (based on the *Newsletters - Unsubscription request* e-mail template, which can be edited in *Site Manager/CMS Desk -> Administration -> E-mail templates*) by submitting their e-mail address. The unsubscription link in the sent e-mail will only be valid for the amount of hours specified in the *Site Manager -> Settings -> On-line marketing -> Newsletters -> Double-opt in interval* setting.

- My subscriptions - this web part can be used by users to add or remove their newsletter subscriptions.
- My account - if configured to display newsletter subscriptions, this web part can be used by users to add or remove their subscriptions.
- Subscription approval - approves subscriptions to newsletters with double opt-in enabled and displays a confirmation message. Subscribers are typically directed to the page containing this web part through a link in the double opt-in activation e-mail. An identifier of the exact subscription is read from a parameter passed in the query string of the URL.

8.36.7 Newsletter templates

The e-mails sent by the Newsletter module are defined by templates (with the exception of dynamic newsletter issues). These templates can be managed in **CMS Desk -> Tools -> Newsletters -> Templates**.



There are four types of templates:

- **Double opt-in** - template for e-mail messages sent to users to confirm their subscription to a newsletter that has double opt-in enabled.
- **Newsletter issue** - this is a template that defines the layout and design of static newsletter issues. Templates of this type typically contain editable regions where newsletter authors can enter the content of individual issues.
- **Subscription** - template for the confirmation e-mail message sent when a user subscribes to the newsletter.
- **Unsubscription** - template for the e-mail message sent when a user unsubscribes.

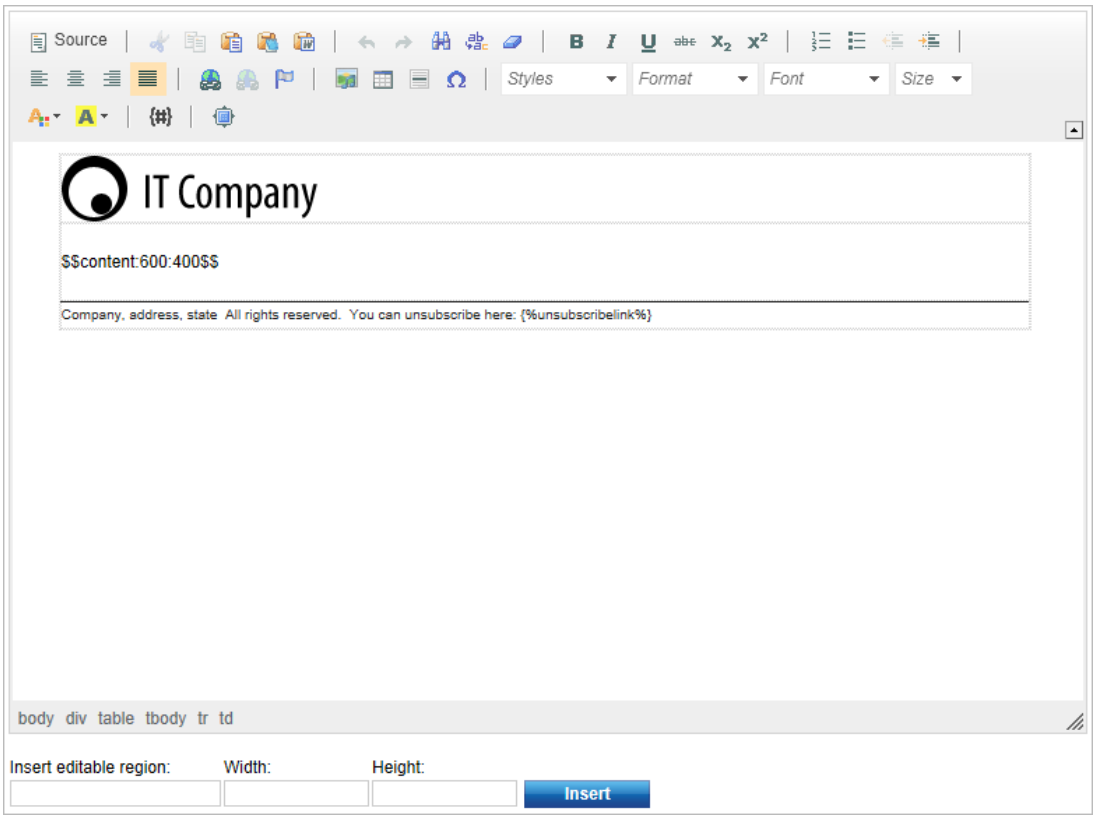
Editing templates

When editing a template, the following configuration can be made on the **General** tab:

- **Display name** - sets the name of the template that is displayed in the newsletter interface.
- **Code name** - sets the name of the template that is used as an identifier, for example in URLs or the API.
- **Thumbnail** - allows you to upload an image that will be displayed as a representation of the template in the selection dialog (when creating new newsletter issues). Only available for *Newsletter issue* templates.
- **Subject** - can be used to set the subject of the e-mails based on this template. This field is not available for *Newsletter issue* templates, since the e-mail subject is entered locally when writing

individual issues.

- **Header (HTML)** - the opening HTML code of the e-mail message, including the <html> element.
- **Body** - defines the main layout of the e-mails that use this template. Here you can enter static text, use the WYSIWYG editor and insert macros or newsletter fields. *Newsletter issue* templates may also contain editable regions where editors can enter the content of individual newsletter issues. These can be inserted using the *Insert editable region* section below the editing area. Regions are inserted as expressions in format: *$$regionName:width:height$$*



- **Footer (HTML)** - the closing HTML code.
- **CSS stylesheet** - may be used to define CSS classes that will be applied to the e-mails based on this template.

You can insert the following newsletter fields anywhere inside the body of the template:

- **{%ActivationLink%}** - only available for *Double opt-in* templates. It is resolved into a link to the subscription approval page, as defined by the *Approval page URL* property of the given newsletter.
- **{%Email%}** - resolves into the e-mail address of the e-mail recipient.
- **{%FirstName%}** - resolves into the first name of the recipient.
- **{%LastName%}** - resolves into the last name of the recipient.
- **{%UnsubscribeLink%}** - resolves into a link to the unsubscription page, as defined by the *Unsubscription page URL* property of the given newsletter.

To easily add these expressions into the body, click the **Insert macro** ({#}) button on the WYSIWYG editor toolbar and select the appropriate option under the **Context specific objects** section.

Note: The newsletter field expressions only work in e-mails sent to actual subscribers. The macros do not resolve correctly when sending draft e-mails for newsletter issues, since the recipients are not linked

to subscribers in the system.

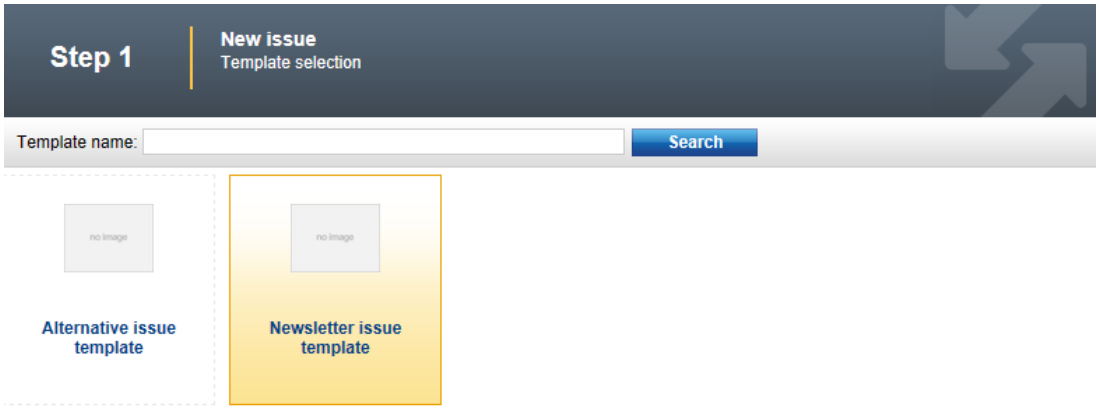
Newsletter templates also support all other types of macro expressions.

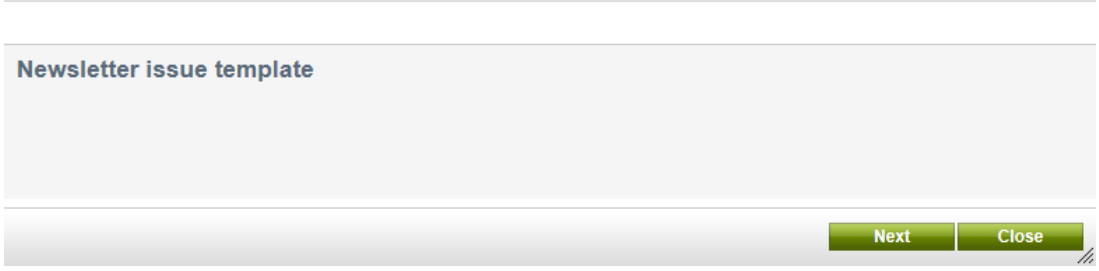
You can attach files to newsletter templates and manage them as needed by clicking the **Attachments** button in the header of the editing page. When e-mails based on the given template are sent out, the attachments will be included. For example, if you wish to display a logo image somewhere in the template layout and have it stored directly in the e-mails, you can upload it as an attachment and then insert it into the body content. In the case of newsletter issue templates, further attachments may also be added when creating specific issues.

Assigning issue templates to newsletters

In addition to the main template configured for a static newsletter, it is also possible to add any number of alternative templates that may be selected for specific issues. When editing a *Newsletter issue* type template, you can assign it to individual newsletters on the **Newsletters** tab. Alternatively, newsletter-template bindings can be configured from the opposite side of the relationship by editing a static newsletter on its **Templates** tab.

Once a newsletter has multiple templates allowed, authors of issues can choose which one they wish to use in the first step of the New issue wizard.

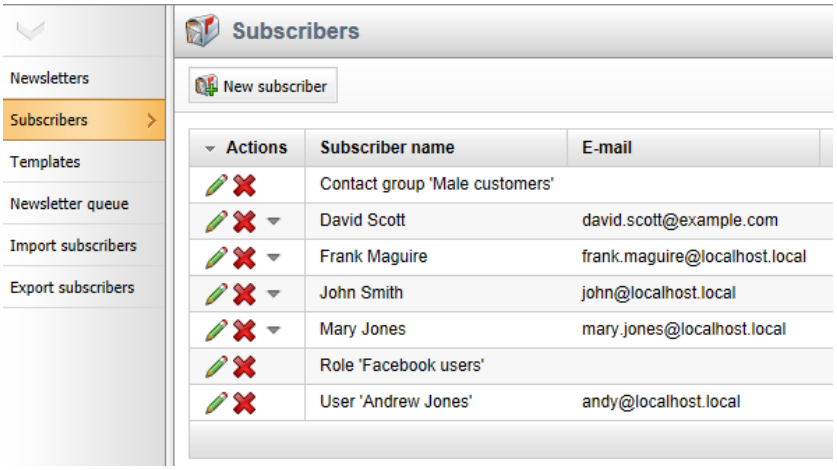




The template of an existing issue may also be changed through the **Template** selector available among the *advanced options* when editing the given issue on the **Content** tab.

8.36.8 Subscriber management

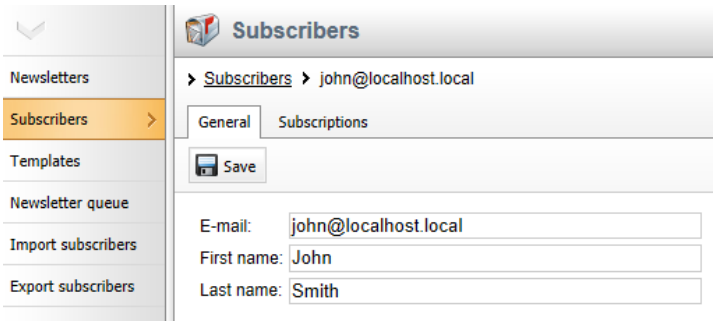
You can manage subscribers at **CMS Desk -> Tools -> Newsletters -> Subscribers**.



On this page, you can see a list of all subscribers belonging to the website's newsletters. You can filter these according to their name or e-mail by entering the desired value into the appropriate filter field and clicking the **Show** button.

By clicking the **Delete** (✖) action icon next to a subscriber record, you can remove the subscriber from the list and consequently from all newsletters to which they are subscribed.

Through the **Edit** (✎) icon, you can modify a subscriber's details and manage their newsletter subscriptions. On the **General** tab, you have the same options as when creating a new subscriber.



Customizing subscriber fields

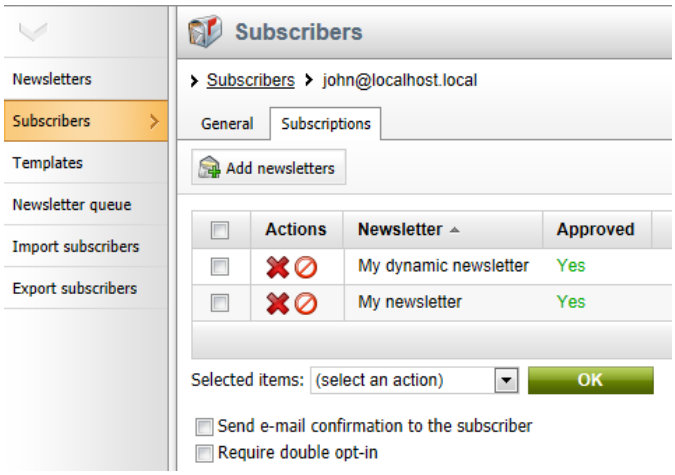
If you wish to store additional data for newsletter subscribers, you can add your own custom fields. To do this, go to **Site Manager -> Development -> System** tables and edit the **Newsletter - Subscriber** table. Then you can define any required fields on the **Fields** tab.

> Please note that the default fields must remain unchanged to ensure that the module continues to work correctly.
>
> The values of custom fields may be entered manually on the **General** tab, filled in by users who subscribe through an appropriately configured Custom subscription form web part, or populated via the API.

On the **Subscriptions** tab, you can see a list of newsletters to which the edited subscriber is subscribed. Additional ones can be added by using the **Add newsletters** button. Individual newsletter subscriptions may be managed via the following actions:

- **Remove** - unsubscribes the subscriber from the selected newsletter.
- **Approve** - manually approves the subscription to the selected newsletter, causing issues to be sent.
- **Reject** - rejected subscriptions remain in the list, but issues of the selected newsletter are not sent. Users who subscribe to newsletters with double opt-in enabled are rejected by default until they confirm their subscription.

To apply these actions to multiple newsletters, mark them using the boxes on the left choose an action from the drop-down list below and click **OK**.



If you check the **Send e-mail confirmation to the subscriber** box before performing an action, a notification e-mail will be sent to the edited subscriber, informing about the changes in their subscription. If you check the **Require double opt-in** box before adding newsletters, subscriptions to newsletters that have double opt-in enabled will be inactive (rejected) until the user confirms them or the **Approve** ( ) action is manually selected in the list of newsletters.

Subscribers of individual newsletters

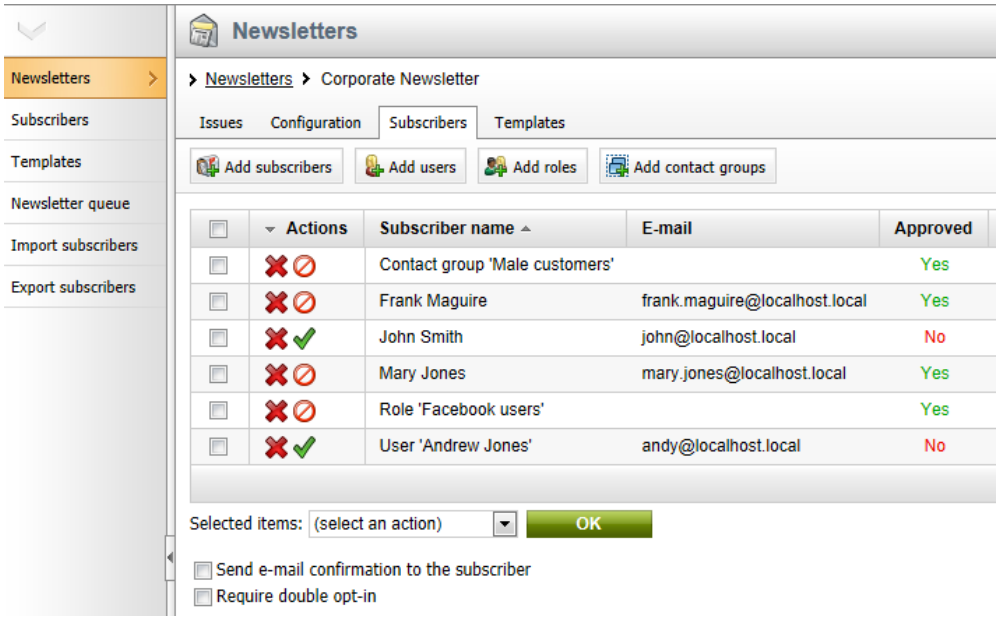
Alternatively, subscribers of individual newsletters can be managed by selecting **Newsletters** in the left menu and editing ( ) the given newsletter on the **Subscribers** tab. Here you can find a list similar to the one on the main Subscribers tab, but it only contains subscribers of the current newsletter. If necessary, subscribers in the list can be filtered by their e-mail, name or approval status. Subscribers can be added by using the following buttons:

- **Add subscribers** - allows selection from a list of all subscribers in the system.
- **Add users** - allows selection from the registered users of the current site.
- **Add roles** - allows selection from the roles of the current site (newsletter issues will be sent to all members of the selected roles).
- **Add contact group** - allows selection from all on-line marketing contact groups defined for the current site (all contacts that belong to the selected groups will be added as subscribers).

Individual subscriptions can be managed via the following actions:

- **Remove** - unsubscribes the address, user, role or contact group from the edited newsletter.
- **Approve** - manually approves the given subscription, causing it to receive newsletter issues.
- **Reject** - rejected subscriptions remain in the list, but they do not receive newsletter issues. Users who subscribe to newsletters with double opt-in enabled are rejected by default until they confirm their subscription.

To apply these actions to multiple subscriptions, mark them using the boxes on the left choose an action from the drop-down list below and click **OK**.



If you check the **Send e-mail confirmation to the subscriber** box before performing an action, a notification e-mail will be sent to the affected users, informing them about the changes in their subscriptions. If you check the **Require double opt-in** box (only available if the newsletter has double opt-in enabled), added subscriptions will be inactive (rejected) until the users confirm them or the **Approve** (✔) action is manually selected in the list of subscriptions.

Duplicate subscriber addresses

Even though all regular newsletter subscribers need to have a unique e-mail address, it is still possible to have a single address subscribed to a newsletter multiple times through users, roles or contact groups.

> In these cases, the duplicate addresses are filtered out from the newsletter queue, so that each e-mail address only receives the newsletter's issues once.

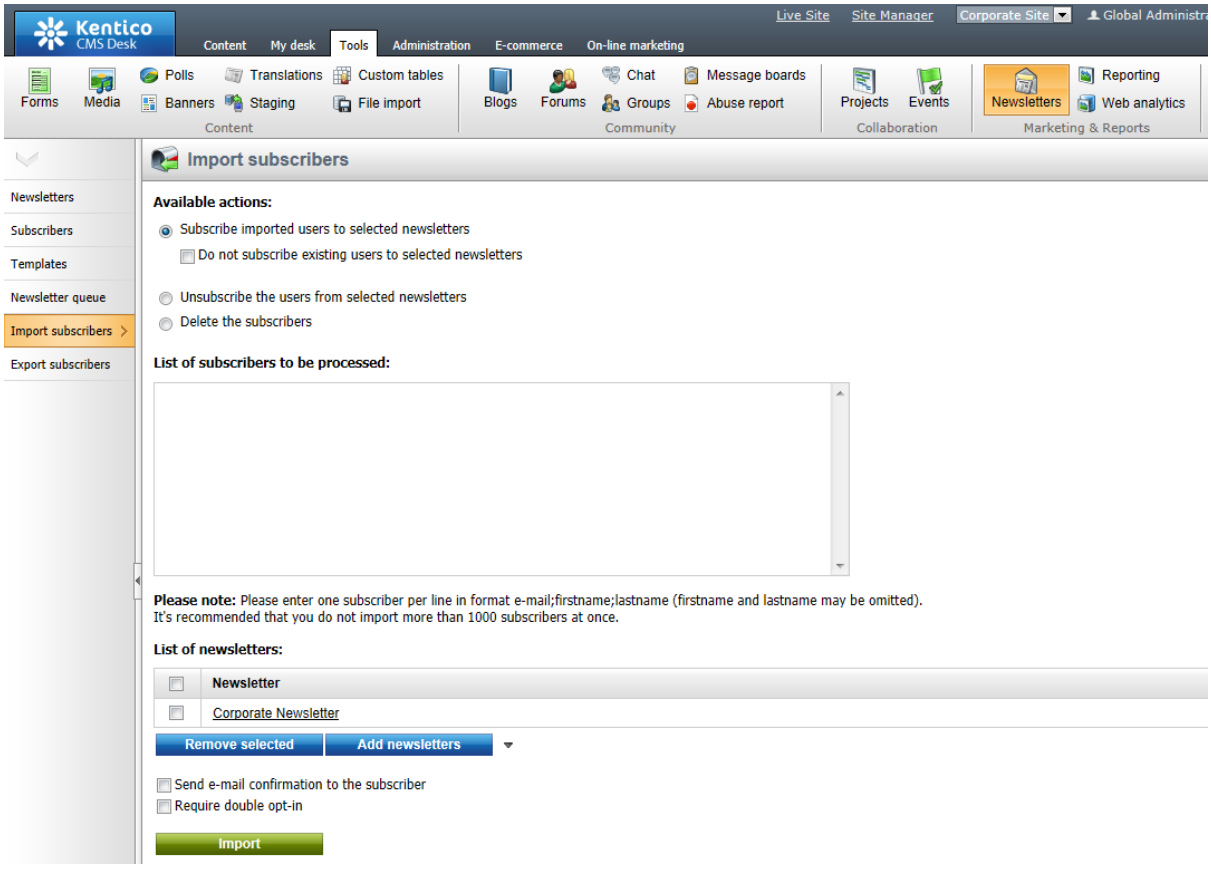
If you need to add or modify a large amount of subscribers, the **Import subscribers** tab described in the Subscriber import and export topic provides an easier way.

8.36.9 Subscriber import and export

When handling a large amount of subscribers, creating or modifying them one by one would be very slow and inefficient. The import and export functionality provides a way to perform mass actions or get subscriber data using specially formatted text.

Importing subscribers

You can import or modify large amounts of subscribers using the **CMS Desk -> Tools -> Newsletters -> Import subscribers** dialog.



You need to prepare a list of subscribers in the following format:

- **email;firstname;lastname**

Copy the list into the **List of subscribers to be processed** text area.

- Each record must be on a new line.
- The name sections are optional.

The following examples are all valid:

```
david.scott@localhost.local;David;Scott
mary.jones@example.com;;Jones
frank.maguire@localhost.local;Frank
monica@localhost.local
```

Note

Importing more than 1000 subscribers at once is not recommended.

If you need to import more subscribers in a single operation, apply hotfix 7.0.28 or newer. With the hotfix, the system performs the import process asynchronously, allowing you to import any number of subscribers.

Select one of the following actions using the radio-buttons above the text area:

- **Subscribe imported users to the selected newsletters** - the imported users will be subscribed to the selected newsletters.
  - **Do not subscribe existing users to selected newsletters** - if checked, the import only updates the names of existing subscribers with matching e-mail addresses (does not add the subscribers to the selected newsletters).
- **Unsubscribe the users from selected newsletters** - removes the imported users from the selected newsletters.
- **Delete the subscribers** - deletes subscribers matching the imported data from the system.

Click **Add newsletters** to select the target newsletters for the subscribe or unsubscribe actions.

If you check **Send e-mail confirmation to the subscriber** before performing the import, a notification e-mail will be sent to the entered addresses, informing them about the changes in their subscriptions. If you check **Require double opt-in**, subscriptions to newsletters that have double opt-in enabled will be inactive (rejected) until the users confirms them.

Clicking **Import** takes the subscribers from the list and performs the selected action. If one of the lines contains an invalid entry, the import is not processed for any of the records and an error is displayed.

Exporting subscribers

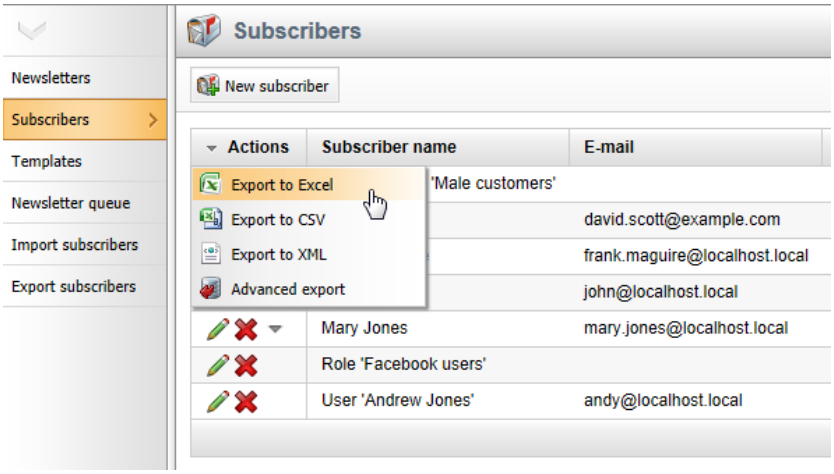
If you need to export a list of subscribers to some other application, you can do so using the **CMS Desk -> Tools -> Newsletters -> Export subscribers** dialog.

You can choose if you want to export all subscribers (do not set any newsletters) or only subscribers of the newsletters specified using the **Add newsletters** button. The subscribers are exported in format:

- **email;firstname;lastname**

The **Export subscribers** radio buttons allow the output to be limited according to the approval status of the subscriptions. Clicking **Export** generates the output, which can be copied from the textbox to your application.

Alternatively, you can also export subscriber data into other file formats (XLSX, CSV, XML) using the general export feature, which is available for all lists of data in Kentico CMS. This can be done by clicking the ▾ icon in the **Actions** column header of the list on the **Subscribers** tab, either in the main menu of the Newsletters module or when editing a specific newsletter. For more details, please refer to the Modules -> UI data export chapter of this guide.



8.36.10 On-line marketing

8.36.10.1 Overview

When using newsletters for e-mail marketing campaigns, it is important to determine their overall effectiveness and optimize issues according to the results. This can be achieved by tracking e-mails and monitoring the reactions of recipients. Kentico CMS offers several features that provide feedback and statistics about how successful your newsletters are and which actions subscribers take when reading issues, including the following:

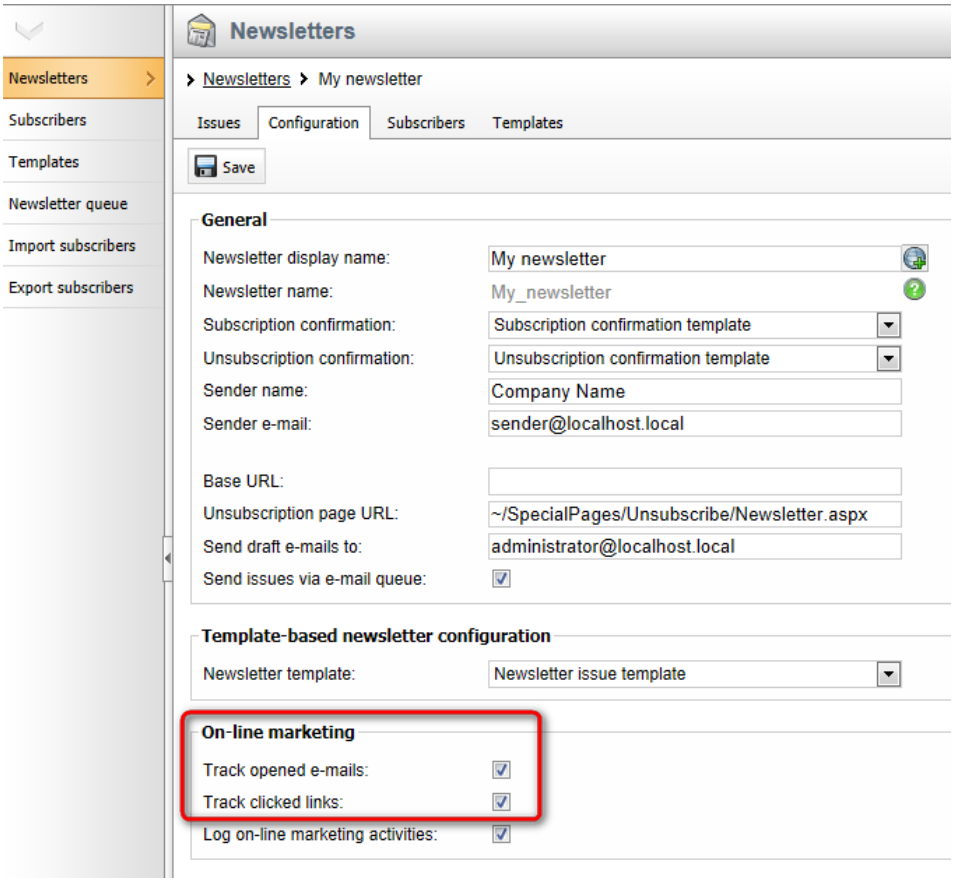
- E-mail tracking - measures how many newsletter issues were actually opened by subscribers and the clickthrough rate of any links placed into the content of the e-mails.
- Bounced e-mail monitoring - allows you to identify and block subscribers with invalid e-mail addresses to which newsletter issues cannot be delivered.
- A/B testing - provides a way to create several different versions of each newsletter issue and evaluate them based on the e-mail tracking statistics measured for a test group of subscribers. This allows you to send the most successful variant to the remainder of the subscribers.

Please note that these features are only available if the **Enable on-line marketing** setting is enabled in **Site Manager -> Settings -> On-line marketing**.

8.36.10.2 E-mail tracking

Two types of tracking can be used for the e-mails that deliver newsletter issues to subscribers. This functionality may be enabled for individual newsletters through the **CMS Desk -> Tools -> Newsletters** interface by editing the selected newsletter on the **Configuration** tab:

- **Track opened e-mails** - if enabled, the e-mails used to send out the issues of the newsletter will be tracked and statistics will be kept about the amount of e-mails that are opened by subscribers.
- **Track clicked links** - if enabled, statistics will be kept about the amount of clicks subscribers perform on hyperlinks placed in the e-mail issues of the newsletter.



Please note that tracking cannot be done retroactively for e-mails that have already been sent before the appropriate settings were enabled. Also, since draft e-mails are intended only for testing purposes, they are not included in tracking statistics (only the e-mails sent to actual subscribers are measured).

Logging newsletter actions as activities

You can also decide if you want the actions related to a specific newsletter to be included in the site's on-line marketing activity statistics. This is done by setting the **Log newsletter actions as on-line marketing activities** property on the **Configuration** tab.

Activities include the following types of events: Subscription, Unsubscription, E-mail opening, Link clickthrough.

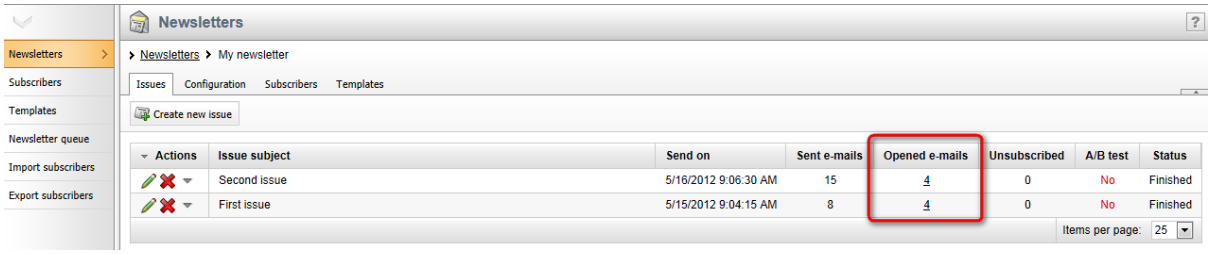
You can then keep track of the logged activities in the **CMS Desk -> On-line marketing -> Activities** interface.

Opened e-mails

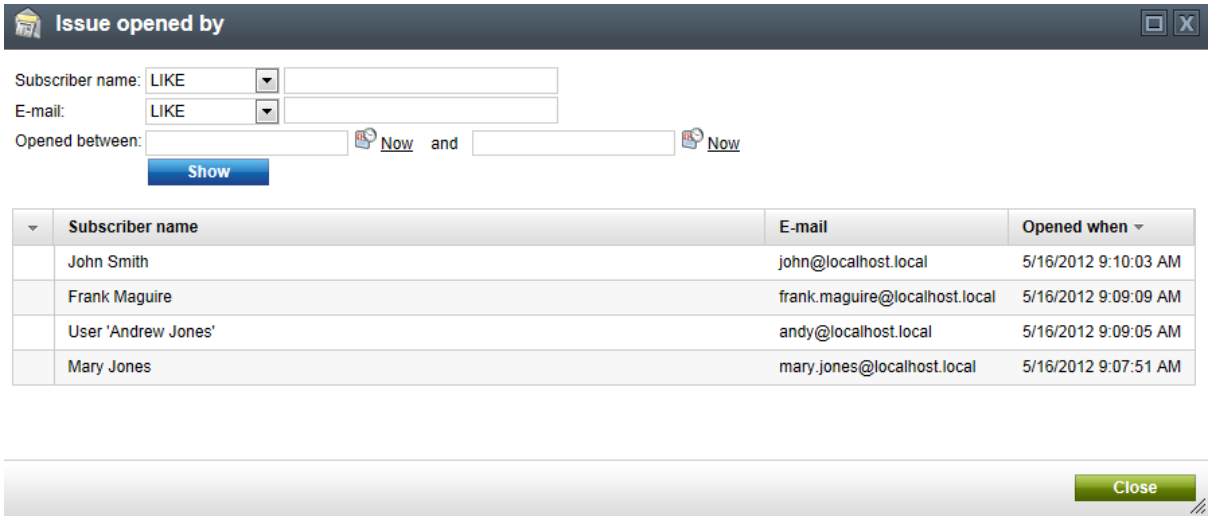
This feature allows you to monitor how many newsletter e-mails are actually opened by subscribers. Open rate is one of the key metrics for judging an e-mail campaign's success.

Tracking of this type is achieved by embedding a small, invisible image into the content of e-mails. When this image is requested from the server for the first time, the system marks the e-mail as received and opened for the given subscriber. As a result, mails will only be counted as opened if the e-mail client of the subscriber allows the loading of images. Alternatively, if the recipient clicks on a link contained in the e-mail and **Track clicked links** is enabled for the newsletter, the e-mail will be recognized as opened even if images are blocked. Because of these factors, the indicated number of opened e-mails may be slightly lower than the actual amount.

If tracking is enabled, the statistics of newsletter issues can be viewed when editing a specific newsletter on the **Issues** tab. The amount of e-mails that were received and opened by subscribers can be compared side by side with the total number of sent e-mails.



If the number is greater than zero, it can be clicked to display a dialog containing a detailed list of all subscribers who opened the e-mail.

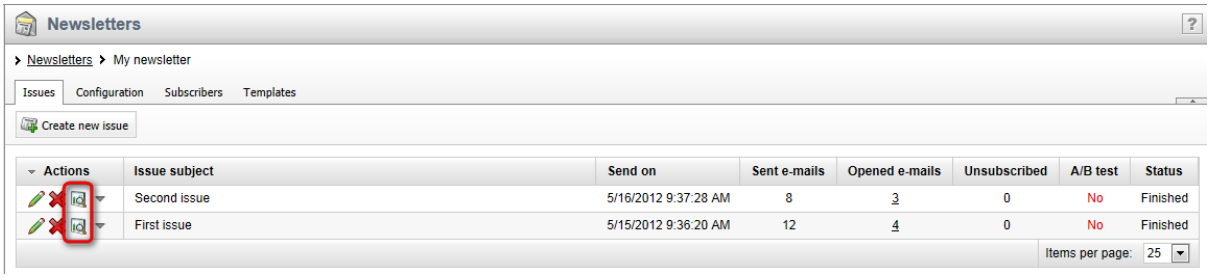


The subscribers can be filtered according to their name, e-mail address or the time interval during which they opened the e-mail.

Clicked links

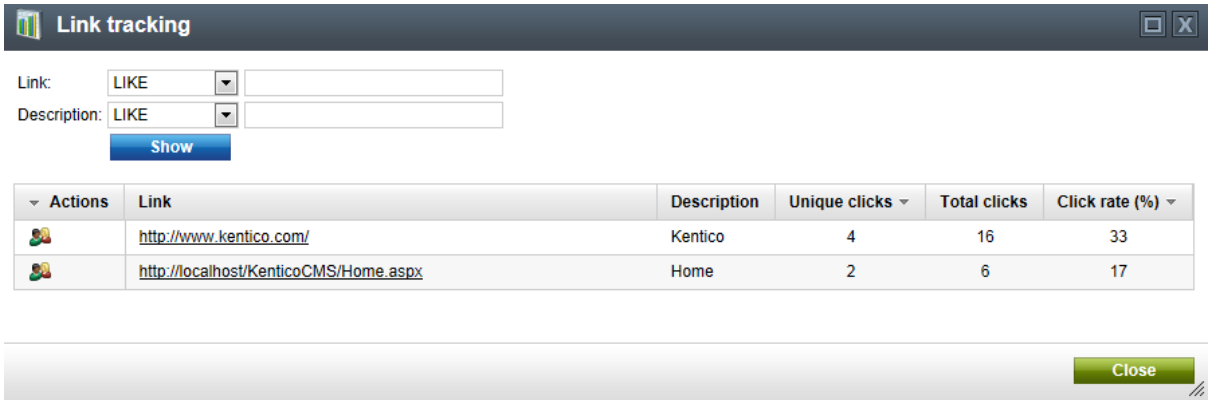
Another way to measure the effectiveness of an e-mail campaign is to track which links are clicked by the recipients and how many times, also known as the *clickthrough rate*. If this type of tracking is enabled, all links in newsletter issues are converted to tracking links when they are sent out and the information is stored. This applies to both template-based and dynamic newsletters.

Link statistics of individual newsletter issues can be viewed when editing a specific newsletter on the **Issues** tab. The **View tracking statistics** ( ) action is available for all issues that contain tracked hyperlinks.



When clicked, this action opens a modal dialog that lists all links in the given issue and displays the following information about them:

- **Unique clicks** - shows how many different subscribers clicked on the link.
- **Total clicks** - indicates how many times the link was clicked, including multiple clicks from the same user.
- **Click rate (%)** - displays the clickthrough rate of the link as a percentage. Calculated as the number of unique clicks divided by the total amount of sent e-mails.



Further details can be displayed for each link by using the **View participating subscribers** ( ) action. This opens another dialog containing the names and addresses of the subscribers that used the given link and their respective click count.

Excluding links from tracking

The newsletter unsubscription links generated by the *{%UnsubscribeLink%}* expression are not tracked. Links to local anchors within the content of the e-mail are also excluded.

The tracking of any link in a newsletter issue may be manually disabled by editing the source and inserting the **tracking="false"** attribute into the **<a>** tag of the given hyperlink, for example:

Kentico CMS

When editing the content of a newsletter issue, you can easily switch to its source using the **Source** button on the toolbar of the WYSIWYG editor.

8.36.10.3 Bounced e-mail monitoring

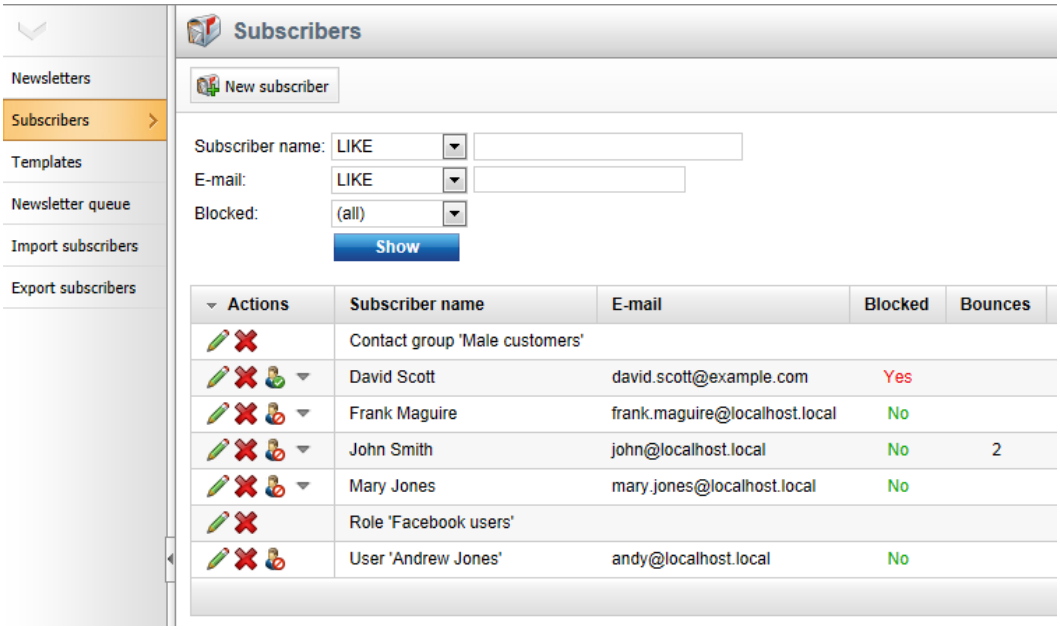
When an e-mail cannot be delivered successfully for some reason, an automatic reply informing about the problem is returned (bounced) back to the sender. Tracking bounced e-mails allows the system to identify addresses that do not correctly receive newsletter issues. Removing invalid addresses from your mailing lists saves bandwidth and improves the accuracy of your subscription statistics.

To enable this feature, go to **Site Manager -> Settings -> On-line marketing -> Newsletters** and check the **Monitor bounced e-mails** box. It is also necessary to properly fill in the remaining settings in the **Bounced e-mails** and **POP3 settings** categories (more information about individual fields is given in the Settings topic).

Once this is done, the **Check bounced e-mails** scheduled task periodically checks the specified mailbox for newsletter e-mail bounces, analyzes them and increases the bounce counter of subscribers as necessary. Once a bounced e-mail is processed, it is deleted from the mailbox. By default, this task is executed once per hour. Some mail servers may be configured to store e-mails even after they are downloaded, which causes bounces to be counted multiple times (every time the scheduled task is executed), so please adjust the settings of the e-mail server if you experience issues of this type.

If the amount of bounces from a subscriber reaches the **Bounced e-mail limit**, the system will automatically block the address from receiving any further newsletter issues.

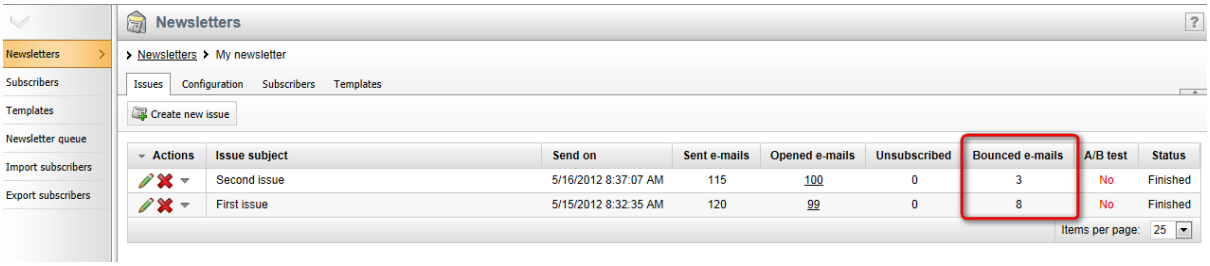
Bounce statistics can be viewed in various parts of the **CMS Desk -> Tools -> Newsletters** interface. On the **Subscribers** tab, the amount of bounces associated with individual subscribers is displayed.



The **Blocked** filter may be used to display only blocked or active subscribers. The **Block** ( ) or **Unblock** ( ) actions may be used to manually change the status of subscribers. When a subscriber is unblocked, their bounce counter is reset to zero. To perform these actions for individual subscribers assigned through a role or contact group, edit ( ) the given role or contact group and switch to the **Users** or **Contacts** tab respectively.

The same options are also available when viewing the subscribers of a specific newsletter in **Newsletters -> edit ( ) newsletter -> Subscribers**. The bounce count of a subscriber is shared for all newsletters on the given site.

On the **Issues** tab of a newsletter, the total amount of bounces per issue is tracked. The amount of sent e-mails can be viewed and compared with the number of bounces.



Monitoring bounced e-mails under medium trust

If your application is running in a medium trust environment, the bounced e-mails feature will not be functional by default.

For more details and a possible solution, please refer to Configuration for Medium Trust environment.

8.36.10.4 A/B testing

When preparing a newsletter issue, it can be difficult to predict exactly how subscribers will react to various important elements, such as the e-mail subject or the primary call to action. In some cases, you may also wish to find out which option out of several alternatives will have the most positive effect. Optimizing your newsletter issues via A/B testing provides a possible solution to these problems.

It allows you to create multiple different versions of each issue referred to as *variants*. These variants are then mailed out to a *test group* of subscribers, typically a relatively small percentage of the newsletter's full mailing list. This way, the best version of the issue can be identified based on the tracking statistics measured for the test group, and then sent to the remainder of the subscribers. The winning variant can either be selected automatically by the system according to specified criteria or manually after evaluating the results of the test.

Please note that A/B testing works best for newsletters that have a large number of subscribers. With a small testing group, the results may be heavily affected by random factors and will not be statistically significant.

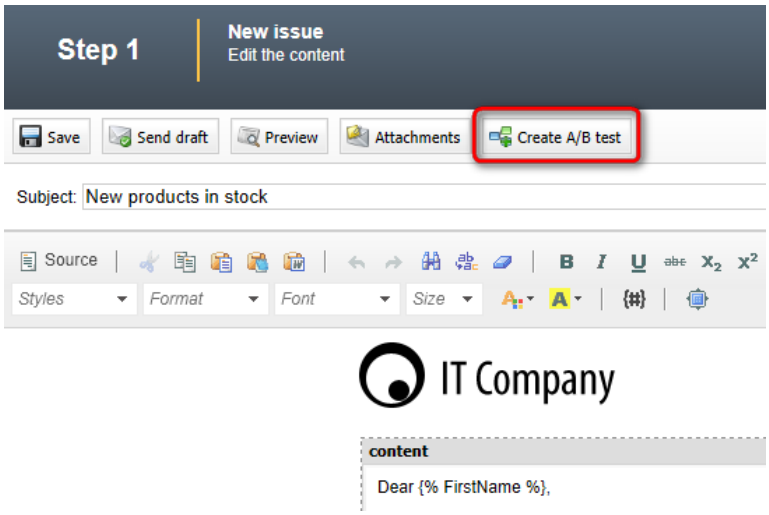
Prerequisites

A/B testing variants of newsletter issues are evaluated based on the actions performed by recipients, so both types of E-mail tracking need to be enabled for the given newsletter on its **Configuration** tab (**Track opened e-mails** and **Track clicked links**). This also requires the **Enable on-line marketing** setting to be enabled in **Site Manager -> Settings -> On-line marketing**.

Additionally, A/B testing is only supported for template-based (static) newsletters.

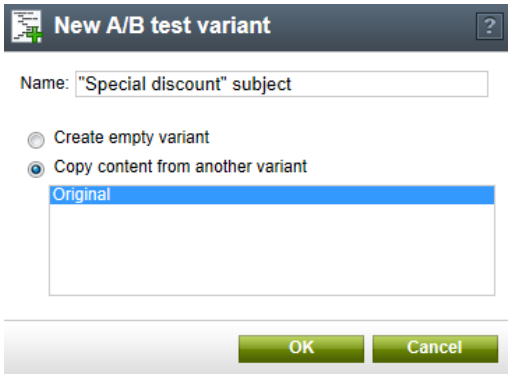
Creating A/B tests for newsletter issues

If the conditions described in the section above are met, A/B tests can be defined through the **CMS Desk -> Tools -> Newsletters** interface either directly when creating a new newsletter issue via the wizard, or when editing an existing issue on the **Content** tab. In the case of new issues, it is necessary to **Save** the content before the advanced actions become available. To add the first testing variant for the issue, click the **Create A/B test** button offered among the header actions.



This opens a dialog where the new variant can be defined. Start by filling in the **Name**, which will be used to identify the variant while working with the A/B test. You can then choose one of two possible options that determine the initial content of the variant:

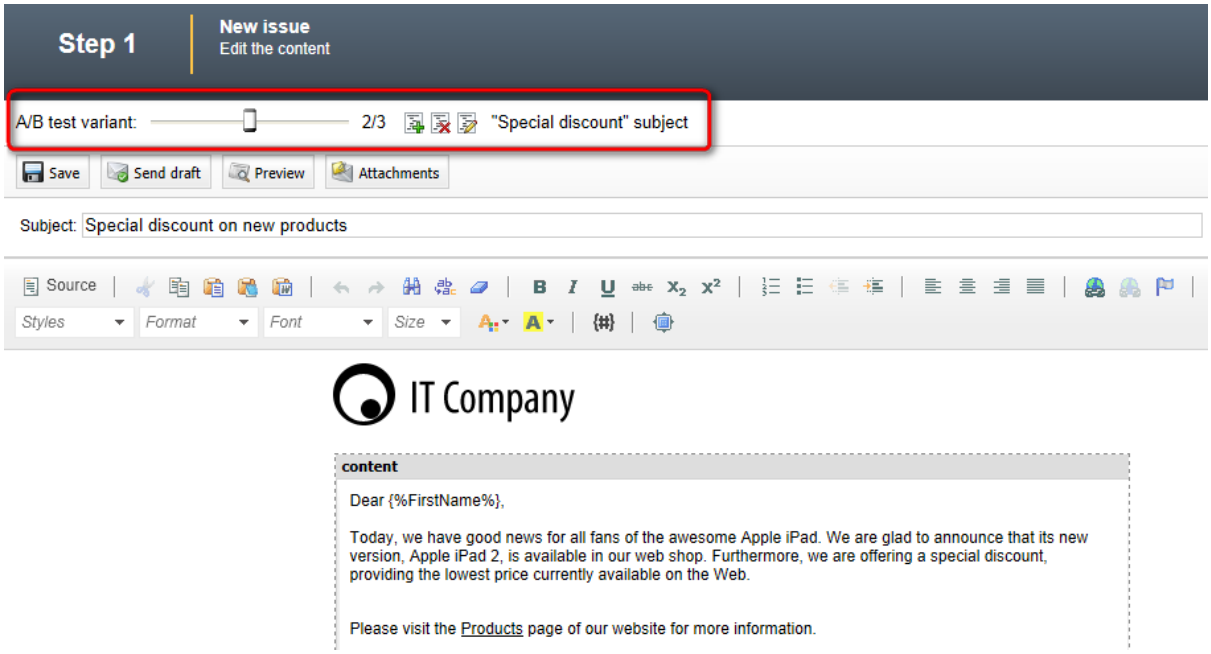
- **Create empty variant** - select this option if you wish to create the issue variant from scratch. The variant will use the main template set for the newsletter on the *Configuration* tab and the content of its editable regions will be empty.
- **Copy content from another variant** - if selected, the content of an existing variant (or the original issue) will be used as a starting point that you can modify as required. Choose the source from the list of issue variants in the list below. The variant will use the same template as the source issue and all editable region content will also be copied. This option makes it easy to create variants for testing small changes, such as a different e-mail subject or text headline.



Now that the issue has an A/B testing variant defined, a slider will be displayed at the top of the content editing page. You can use it to switch between individual variants, including the original issue. The name of the currently selected variant is shown next to the slider. You may manage the A/B test variants using the following buttons:

- **Add variant** - creates another variant of the issue. You can add any number of variants.
- **Remove variant** - deletes the variant currently selected through the slider.
- **Edit properties** - allows you to change the name of the currently selected variant. If required, you may also rename the *original* issue.

The settings and content of the currently selected variant can be modified just like when editing a standard issue. Each variant may have a different subject, issue template, editable region content etc. This allows you to test any variables that you need.

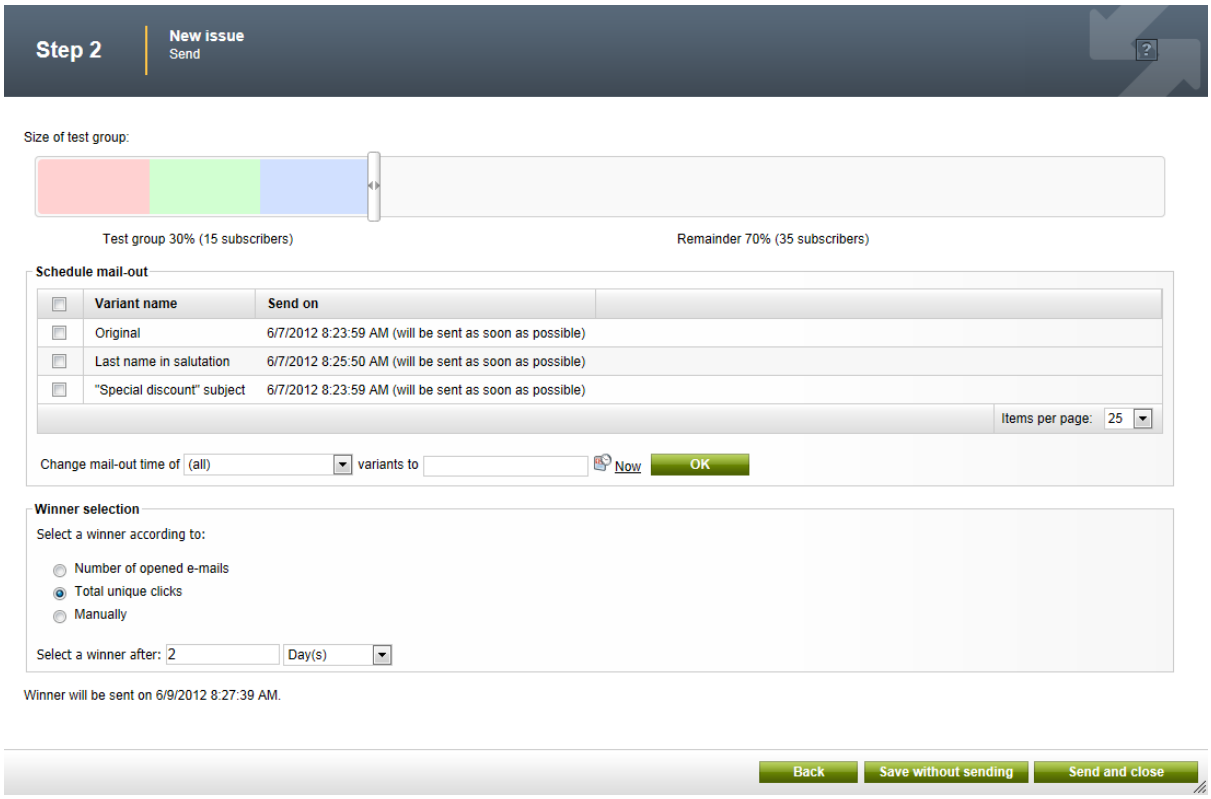


Sending A/B test issues

When all of the issue's variants are defined as required, the next step is to configure and schedule how the test should be sent out and evaluated. This can be done either in the send step of the new issue wizard or when editing an existing issue on the **Send** tab.

The size of the subscriber test group can be defined using the slider in the upper part of the page. By moving the slider's handle, you can increase or decrease the number of subscribers that will receive the variants of the newsletter during the testing phase. The test group is automatically balanced so that each variant is sent to the same amount of subscribers. Because of this, the overall test group size will always be a multiple of the total number of variants created for the issue.

The remaining subscribers who are not part of the test group will receive the variant that achieves the best results (i.e. the winner) after the testing process is complete.



Using a full test group

It is even possible to set up a scenario where the test group includes 100% of all subscribers. In this case, the A/B test simply provides a way to evenly distribute different versions of the issue between the subscribers and the selection of the winner is only done for statistical purposes.

In the **Schedule mail-out** section below the slider, you can specify when individual issue variants should be sent to the subscribers from the corresponding portion of the test group. To schedule the mail-out, enter the required date and time into the field below the list (you may use the **Calendar** selector or the **Now** link) and then click **OK**. This can either be done for a specific variant, *all* variants or only those selected through the checkboxes in the list. If the mail-out time is the same for multiple variants, they will be sent in sequence with approximately 1 minute intervals between individual variants.

The configuration made in the **Winner selection** section determines how the winning variant of the A/B test will be chosen. You can select one of the following options:

- **Number of opened e-mails** - the system will automatically choose the variant with the highest number of opened e-mails as the winner. This type of testing focuses on optimizing the first impression of the newsletter, i.e. the subject of the e-mails and the sender name or address, not the actual content.
- **Total unique clicks** - the winner will be chosen automatically according to the amount of link clicks measured for each variant. Each link placed in the issue's content will only be counted once per subscriber, even when clicked multiple times. This option is recommended if the primary goal of your newsletter is to encourage subscribers to follow the links provided in the issues.

- **Manually** - the winner of the A/B test will not be selected automatically. Instead, the author of the issue (or other authorized users) can monitor the results of the test and choose the winning variant manually at any time.

When using an automatic selection option (one of the first two), it is also necessary to enter the duration of the testing period through the **Select a winner after** settings below. This way, you can specify how long the system should wait after the last variant is sent out before it chooses a winner and mails it to the remaining subscribers.

Once everything is configured as required, you can confirm that the variants should be sent according to their mail-out scheduling time by clicking the **Send** button (or **Send and close** in the new issue wizard). If you only wish to save the configuration of the A/B test without actually starting the mail-out, use the **Save** (**Save without sending**) button instead.

Evaluating A/B tests

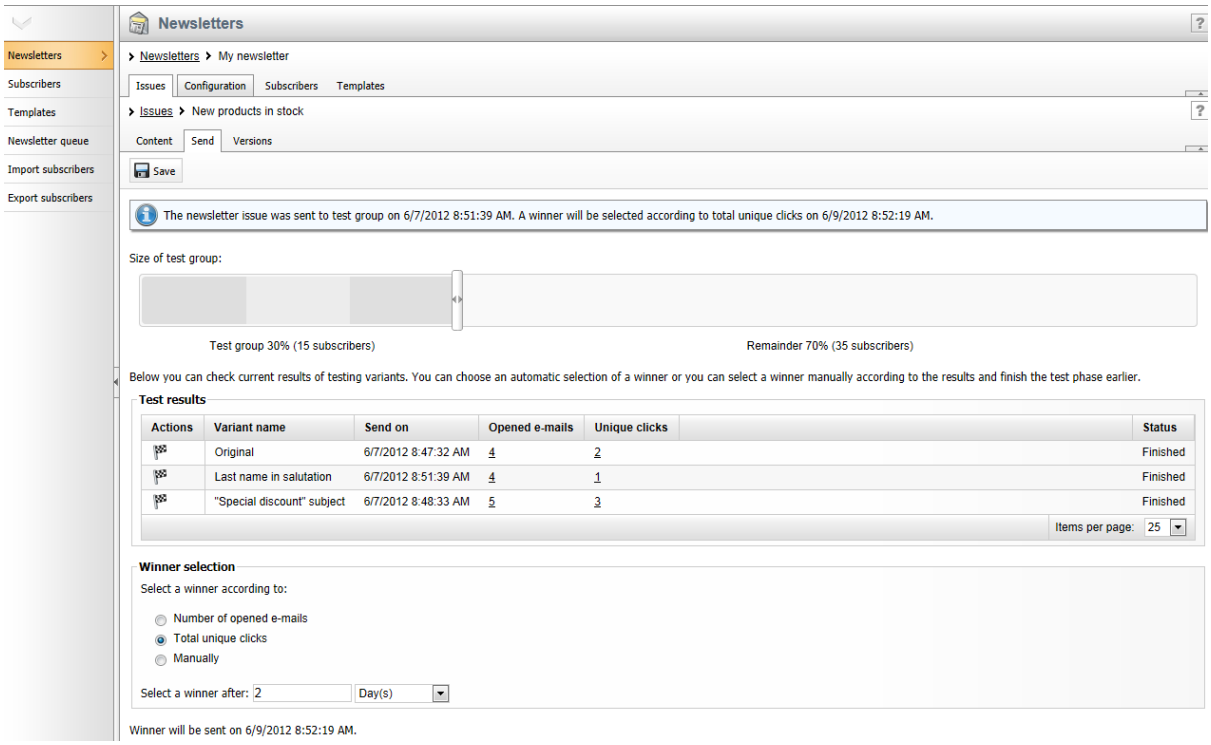
The testing phase begins after the first variant is sent out. If you need to make any changes to the configuration of the A/B test or the content of its variants, you can do so by editing the issue.

On the **Content** tab, you may modify the variants that have not yet been mailed, but the slider actions will be disabled. This means that it is no longer possible to add, remove or rename variants.

If you switch to the **Send** tab, the test group slider will now be locked. However, you can view the e-mail tracking data measured for individual variants in the **Test results** section. The current tracking statistics are shown for each variant, specifically the number of *opened e-mails* and amount of *unique link clicks* performed by subscribers. By clicking on these numbers, you can open a dialog with the details of the corresponding statistic for the given variant. It is also possible to reschedule the sending of variants that have not been mailed yet using the selector and date-time field below the list.

The **Select as winner** action allows you to manually choose a winner (even when using automatic selection). It opens a confirmation dialog where you can schedule when the winning issue variant should be sent to the remaining subscribers. If you specify a date in the future, you will still have the option of choosing a different winner during the interval before the mail-out.

The winner selection criteria may also be changed at any point while the testing is still in progress.

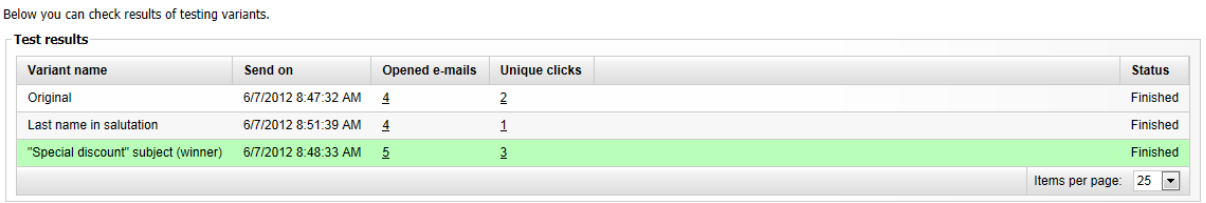


Special cases with tied results

If a draw occurs at the end of the testing phase (i.e. the top value in the tested statistic is achieved by multiple issue variants), the selection of the winner will be postponed and evaluated again after one hour.

In certain situations, you may need to choose the winner manually even when using automatic selection, e.g. if you are testing the number of opened e-mails and all subscribers in the test group view the received issue.

Once the test is concluded and the winner is decided, the given variant is highlighted by a green background. At this point, the winning issue is mailed out to the remaining subscribers who were not included in the test group and no further actions are possible except for viewing the statistics of the variants.



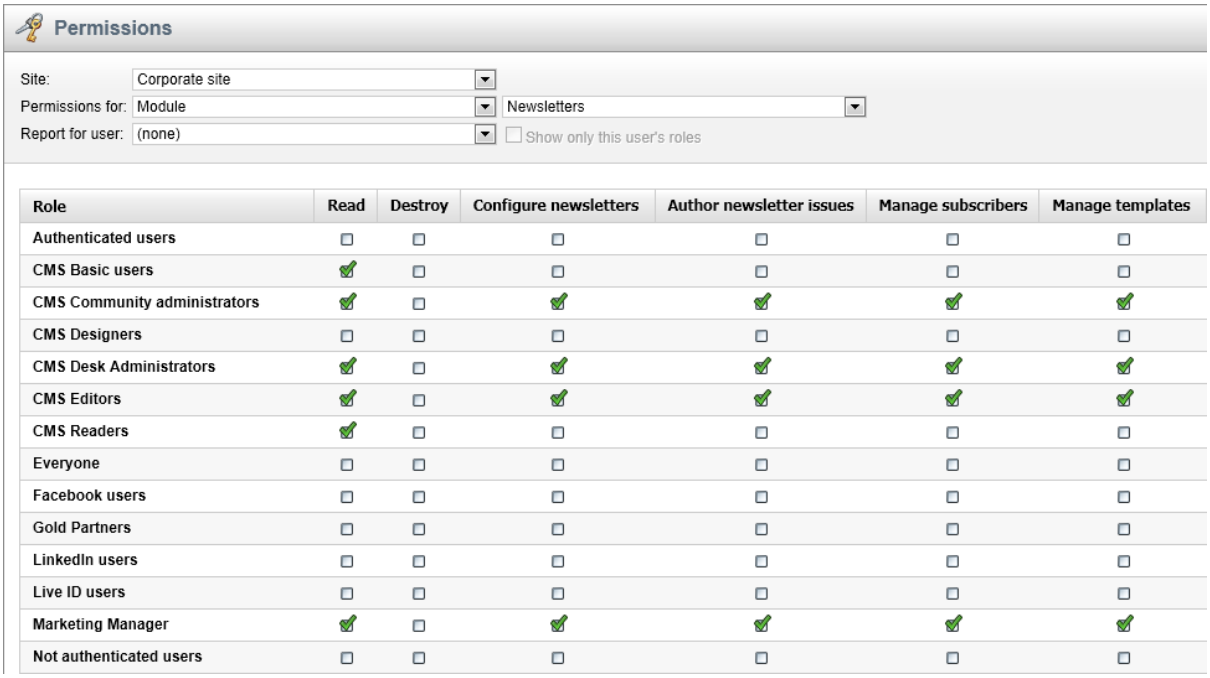
The overall statistics of the A/B tested issue, including the e-mails used to deliver the winning variant to the subscribers outside of the test group, can be monitored in the usual way on the **Issues** tab of the newsletter. When viewing the opened e-mail or clickthrough data in the detail dialogs, you may use the additional **Variants** filter to display either the total *(all)* values for the entire issue, or only those of

specific variants. The statistics of the winning variant include both the corresponding portion of the test group and the remainder of the subscribers who received it after the completion of the testing phase.

8.36.11 Security

You can control access to the Newsletters module from **CMS Site Manager (or CMS Desk) -> Administration -> Permissions**, after selecting the **Module -> Newsletters** permission matrix. The Newsletters module has the following permissions:

- **Read** - members of the selected roles are allowed to view all data in the *CMS Desk -> Tools -> Newsletters* interface.
- **Destroy** - allows members of the roles to delete the version history of newsletter objects.
- **Configure newsletters** - members of the selected roles are allowed to configure newsletter settings.
- **Author newsletter issues** - members of the selected roles are allowed to create and edit newsletter issues.
- **Manage subscribers** - members of the selected roles are allowed to add and remove newsletter subscribers.
- **Manage templates** - members of the selected roles are allowed to create, edit and delete newsletter templates.



8.36.12 Settings

Site-level or global newsletter settings can be configured at **Site Manager -> Settings -> On-line marketing -> Newsletters**. The following options are available:

General

- **Newsletter unsubscription page URL** - sets the URL of the page where users can unsubscribe from a newsletter. The *Newsletter unsubscription* web part must be must be placed on the specified page to ensure the required functionality. The value of this setting is inherited by individual

newsletters if their *Unsubscription page URL* property is not set. If empty, the *~/CMSPages/Unsubscribe.aspx* default system page is used.

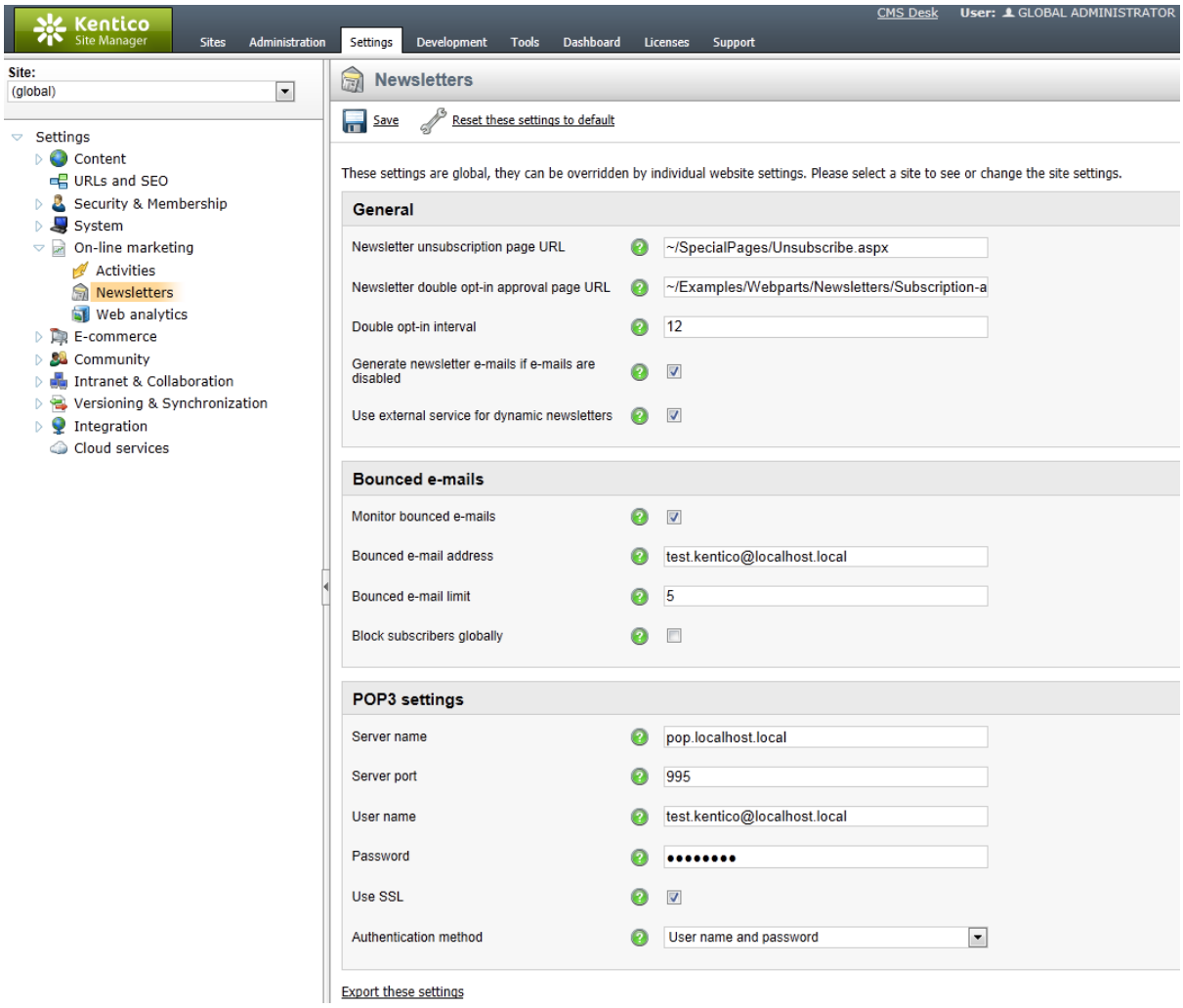
- **Newsletter double opt-in approval page URL** - sets the URL of the page where users can confirm their subscription to a newsletter with double opt-in enabled. The *Subscription approval* web part must be placed on the specified page to ensure the required functionality. The value of this setting is inherited by individual newsletters if their *Approval page URL* property is not set. If empty, the *~/CMSModules/Newsletters/CMSPages/SubscriptionApproval.aspx* default system page is used.
- **Double opt-in interval** - sets the length (in hours) of the time interval during which users will be allowed to confirm their subscription to a newsletter that uses double opt-in. If a user does not activate their subscription within the specified number of hours, the link in their confirmation e-mail will expire and become invalid. The same time limit is also applied to unsubscription requests submitted through the *Unsubscription request* web part.
- **Generate newsletter e-mails if e-mails are disabled** - if enabled, newsletter e-mails will be generated and saved into the E-mail queue even if the sending of e-mails is disabled in *Site Manager -> Settings -> System -> E-mails*.
- **Use external service for dynamic newsletters** - if enabled, the scheduled tasks that are created to ensure the mail-out of new dynamic newsletters will be set to be processed by the external scheduling service by default. Please note that this does not change the settings of existing dynamic newsletter tasks. If you wish to configure these to use the external scheduler Windows service, you will have to enable their *Use external service* property manually in the *Site Manager -> Administration -> Scheduled tasks* interface (remember to select the appropriate site, these tasks are not global).

Bounced e-mails

- **Monitor bounced e-mails** - indicates if bounced e-mails should be tracked for newsletter subscribers. Bounced e-mails are received whenever there is a problem with the delivery of a newsletter issue to a subscriber.
- **Bounced e-mail address** - sets the address to which bounced e-mails are sent when the delivery of a newsletter issue to a subscriber fails. If set, this address is used in the *From* field of newsletter issue e-mails.
- **Bounced e-mail limit** - sets the amount of bounced e-mails that can be counted for a subscriber before the system blocks them from receiving further newsletter issues. This limit is set for all newsletters under the selected site. If *0* is entered, subscribers will never be blocked automatically, but their bounced e-mail count will still be tracked and displayed in the *CMS Desk -> Tools -> Newsletters* interface.
- **Block subscribers globally** - if checked, bounces will be added to all subscribers that have the same e-mail address. This is applied across all sites in the system. Please note that this setting does not ensure consistency between the bounce counts of all subscribers with a shared address, only that new bounces will be added to all of them.

POP3 settings

- **Server name** - sets the domain name or IP address of the mail server where the bounced e-mails are stored. POP3 is used to check the server and monitor bounced e-mails.
- **Server port** - specifies the number of the port used to connect to the mail server.
- **User name** - sets the user name used for authentication against the mail server.
- **Password** - sets the password used for authentication against the mail server.
- **Use SSL** - indicates whether the connection to the mail server should be secured using SSL.
- **Authentication method** - allows the selection of the authentication method used for the connection to the mail server. Options include basic user name and password authentication and several challenge-response mechanisms. The *Auto* option uses APOP if the server supports it and plain text user name and password authentication otherwise.



8.36.13 Troubleshooting

Problems with sending e-mails

If you are not correctly receiving newsletter e-mails, please check the following:

1. The newsletter issues are sent out using a scheduled task that is executed every 1 minute by default. You may need to wait for up to 2 minutes before you receive newsletter issue e-mail. The scheduled task status can be checked at **CMS Desk -> Administration -> Scheduled tasks**.

2. Go to **CMS Desk -> Tools -> Newsletter -> Newsletter queue**. If some e-mail failed, you will find the error message here. After you resolve the technical issue, you can resend all failed e-mails by clicking **Resend all failed**.

3. If your newsletters are configured to use the e-mail queue, go to **CMS Site Manager -> Administration -> E-mail queue**. Any e-mails lost due to errors can be monitored and re-sent here.

4. Make sure you are using the correct e-mail addresses.

5. Make sure the newsletter issues aren't being blocked by some anti-spam software.

6. Go to **Site Manager -> Settings -> System -> E-mails** or **Administration -> SMTP servers** and make sure your SMTP servers are configured correctly. You can find additional details on SMTP server configuration in the SMTP server configuration topic. You can try sending a test e-mail in **Site Manager -> Administration -> System -> E-mail**.

7. Newsletter e-mail debugging might be helpful when solving problems with newsletter e-mails. To enable newsletter debugging, add the following keys to the */configuration/appSettings* section of your project's *web.config* file:

```
<add key="CMSLogEmails" value="true"/>
<add key="CMSDebugEmails" value="true"/>
```

The first key enables you to log all the sent e-mails. You can find the logged e-mails in **<web root>/AppData/logemails.log**. The log contains each e-mail's recipient and subject.

The second key enables you to log all the sent e-mails into event log without actually sending them to the recipients. This is helpful when you need to test the functionality but do not want the testing e-mails to actually reach the recipient. You can view the event log in **Administration -> Event log** section of both **CMS Desk** and **Site Manager**. The e-mails are logged as an **Information** type of event.

Additionally, with the second key enabled, a *Sending failed for <recipient's e-mail address>* error is generated at random to simulate an error when sending an e-mail.

Problems with unsubscription links

If your unsubscription links are not resolving correctly, check that the **Base URL** property of the given newsletter is set correctly according to the domain name of your website. To configure this property:

1. Go to **CMS Desk -> Tools -> Newsletters**.
2. **Edit** () the newsletter and switch to its **Configuration** tab.

Note: Newsletters only generate valid unsubscription links when sending issues to actual subscribers. Unsubscription links do not work in draft e-mails, since the recipients are not linked to subscribers in the system.

Problems with role or contact group subscribers

If you have a user role set as a subscriber for your newsletter, it is highly recommended to send the newsletter issues via the e-mail queue in order for the issues to be successfully sent to all members of the role. To set the newsletter to use the newsletter queue:

1. Go to **CMS Desk -> Tools -> Newsletter**.
2. **Edit** () your newsletter and switch to the **Configuration** tab.
3. Enable the **Send issues via e-mail queue** property.
4. Click  **Save**.

Problems with event log spam *[This option is only available after applying hotfix 7.0.18 or newer]*

By default, the system logs an event log entry for every sent newsletter issue. When mailing out issues to a very large number of subscribers, this may lead to a cluttered event log or even website performance problems.

You can disable logging of newsletter issue changes (including send events) by adding the following key to the */configuration/appSettings* section of your project's *web.config* file:

```
<add key="CMSLogNewsletterIssueEvents" value="false" />
```

8.36.14 Newsletters internals and API

8.36.14.1 Overview

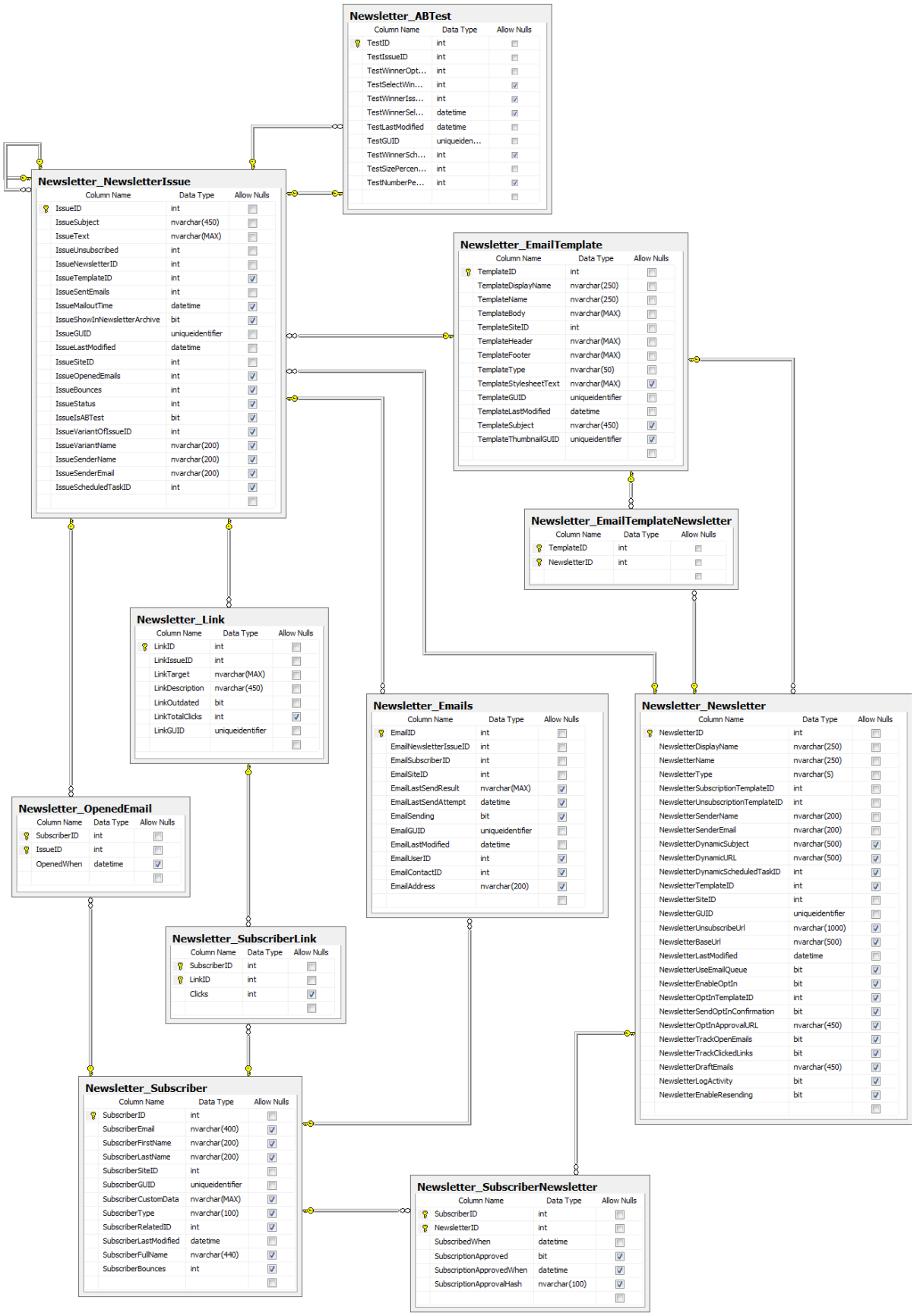
In this chapter, you will learn which database tables and API classes are used by the Newsletters module. You will also see the most common API examples.

Advanced API examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all methods from the module classes, please refer to Kentico CMS API Reference.

8.36.14.2 Database tables

The following database tables are used for the newsletters module:

- **Newsletter_Newsletter** - contains records representing newsletters and their configuration.
- **Newsletter_NewsletterIssue** - used to store individual newsletter issues, including their content.
- **Newsletter_Subscriber** - contains records representing newsletter subscribers.
- **Newsletter_SubscriberNewsletter** - stores relationships between subscribers and newsletters. This table also contains data that determines whether the subscriber is approved for the given newsletter.
- **Newsletter_EmailTemplate** - used to store the templates on which newsletter issues and notification messages are based (all types).
- **Newsletter_EmailTemplateNewsletter** - stores relationships between newsletter issue templates and static newsletters. Each record indicates that a certain template may be used for the issues of a specific newsletter.
- **Newsletter_Emails** - this table stores newsletter e-mails that could not be sent out successfully (i.e. the newsletter queue). This allows them to be re-sent at a later time.
- **Newsletter_OpenedEmail** - stores relationships between subscribers and newsletter issues. Each record indicates that a subscriber has opened a specific issue.
- **Newsletter_Link** - contains records representing links inside the content of newsletter issues.
- **Newsletter_SubscriberLink** - stores relationships between subscribers and newsletter links. Each record indicates that a subscriber has clicked on a specific link.
- **Newsletter_ABTest** - stores records that contain the A/B testing configuration of newsletter issues. Individual A/B test variants of issues are saved as separate items in the *Newsletter_NewsletterIssue* table.



8.36.14.3 API classes

If you are not familiar with database table data management through Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

The classes of the newsletters module can be found in the **CMS.Newsletter** namespace.

Newsletter_Newsletter table API:

- **NewsletterInfo** - represents one newsletter object.
- **NewsletterInfoProvider** - provides management functionality for newsletters.

Newsletter_NewsletterIssue table API:

- **IssueInfo** - represents one newsletter issue.
- **IssueInfoProvider** - provides management functionality for newsletter issues.

Newsletter_Subscriber table API:

- **SubscriberInfo** - represents one newsletter subscriber.
- **SubscriberInfoProvider** - provides management functionality for subscribers.

Newsletter_SubscriberNewsletter table API:

- **SubscriberNewsletterInfo** - represents a relationship between a subscriber and a newsletter.
- **SubscriberNewsletterInfoProvider** - provides management functionality for subscriber-newsletter relationships.

Newsletter_EmailTemplate table API:

- **EmailTemplateInfo** - represents one newsletter e-mail template.
- **EmailTemplateInfoProvider** - provides management functionality for newsletter templates.

Newsletter_EmailTemplateNewsletter table API:

- **EmailTemplateNewsletterInfo** - represents a relationship between a newsletter and an issue template.
- **EmailTemplateNewsletterInfoProvider** - provides management functionality for newsletter-template relationships.

Newsletter_Emails table API:

- **EmailQueueItem** - represents an issue e-mail in the newsletter e-mail queue.
- **EmailQueueManager** - provides management functionality for the newsletter queue.

Newsletter_OpenedEmail table API:

- **OpenedEmailInfo** - represents a relationship which indicates that a subscriber has opened a newsletter issue.
- **OpenedEmailInfoProvider** - provides management functionality for subscriber-issue relationships.

Newsletter_Link table API:

- **LinkInfo** - represents a link in a newsletter.
- **LinkInfoProvider** - provides management functionality for newsletter links.

Newsletter_SubscriberLink table API:

- **SubscriberLinkInfo** - represents a relationship between a subscriber and a newsletter link (i.e. the amount of clicks that the given subscriber performed, if any).
- **SubscriberLinkInfoProvider** - provides management functionality for subscriber-link relationships.

Newsletter_ABTest table API:

- **ABTestInfo** - represents the A/B test configuration of a newsletter issue.
- **ABTestInfoProvider** - provides management functionality for newsletter issue A/B tests.

8.36.14.4 API examples

8.36.14.4.1 Overview

These topics show examples of how the Newsletters module API can be used:

- Managing email templates
- Managing newsletters
- Managing subscribers
- Managing issues

Please note

All API examples can be simulated in the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\OnlineMarketing\Newsletters\Default.aspx.cs***.

The Newsletters module API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.GlobalHelper;
```

```
using CMS.CMSHelper;
using CMS.SiteProvider;
using CMS.Newsletter;
```

8.36.14.4.2 Managing email templates

The following example creates a subscription template.

```
private void CreateSubscriptionTemplate()
{
    // Creates a new subscription template object
    EmailTemplateInfo newTemplate = new EmailTemplateInfo();

    // Sets the properties
    newTemplate.TemplateDisplayName = "My new subscription template";
    newTemplate.TemplateName = "MyNewSubscriptionTemplate";
    newTemplate.TemplateType = EmailTemplateType.Subscription;
    newTemplate.TemplateBody = "My new subscription template body";
    newTemplate.TemplateHeader = "<html xmlns=\"http://www.w3.org/1999/
xhtml\"><head><title>Newsletter</title><meta http-equiv=\"content-type\"
content=\"text/html; charset=UTF-8\" /></head><body>";
    newTemplate.TemplateFooter = "</body></html>";
    newTemplate.TemplateSiteID = CMSContext.CurrentSiteID;

    // Saves the subscription template
    EmailTemplateInfoProvider.SetEmailTemplateInfo(newTemplate);
}
```

The following example creates an unsubscription template.

```
private void CreateUnsubscriptionTemplate()
{
    // Creates a new unsubscription template object
    EmailTemplateInfo newTemplate = new EmailTemplateInfo();

    // Sets the properties
    newTemplate.TemplateDisplayName = "My new unsubscription template";
    newTemplate.TemplateName = "MyNewUnsubscriptionTemplate";
    newTemplate.TemplateType = EmailTemplateType.Unsubscription;
    newTemplate.TemplateBody = "My new unsubscription template body";
    newTemplate.TemplateHeader = "<html xmlns=\"http://www.w3.org/1999/
xhtml\"><head><title>Newsletter</title><meta http-equiv=\"content-type\"
content=\"text/html; charset=UTF-8\" /></head><body>";
    newTemplate.TemplateFooter = "</body></html>";
    newTemplate.TemplateSiteID = CMSContext.CurrentSiteID;

    // Saves the unsubscription template
    EmailTemplateInfoProvider.SetEmailTemplateInfo(newTemplate);
}
```

The following example creates an issue template.

```
private void CreateIssueTemplate()
{
    // Creates a new issue template object
    EmailTemplateInfo newTemplate = new EmailTemplateInfo();

    // Sets the properties
    newTemplate.TemplateDisplayName = "My new issue template";
    newTemplate.TemplateName = "MyNewIssueTemplate";
    newTemplate.TemplateType = EmailTemplateType.Issue;
    newTemplate.TemplateBody = "<p>My new issue template body</
p><p>$$content:800:600$$</p>";
    newTemplate.TemplateHeader = "<html xmlns=\"http://www.w3.org/1999/
xhtml\"><head><title>Newsletter< title><meta http-equiv=\"content-type\"
content=\"text/html; charset=UTF-8\" /></head><body>";
    newTemplate.TemplateFooter = "</body></html>";
    newTemplate.TemplateSiteID = CMSContext.CurrentSiteID;

    // Saves the issue template
    EmailTemplateInfoProvider.SetEmailTemplateInfo(newTemplate);
}
```

The following example gets and updates an issue template.

```
private bool GetAndUpdateIssueTemplate()
{
    // Gets the issue template
    EmailTemplateInfo updateTemplate =
EmailTemplateInfoProvider.GetEmailTemplateInfo("MyNewIssueTemplate",
CMSContext.CurrentSiteID);
    if (updateTemplate != null)
    {
        // Updates the properties
        updateTemplate.TemplateDisplayName =
updateTemplate.TemplateDisplayName.ToLower();

        // Saves the changes
        EmailTemplateInfoProvider.SetEmailTemplateInfo(updateTemplate);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates issue templates.

```
private bool GetAndBulkUpdateIssueTemplates()
{
```

```
    // Prepares the parameters
    string where = "TemplateName LIKE N'MyNewIssueTemplate%'";

    // Gets the data
    DataSet templates = EmailTemplateInfoProvider.GetEmailTemplates(where, null);
    if (!DataHelper.DataSourceIsEmpty(templates))
    {
        // Loops through individual items
        foreach (DataRow templateDr in templates.Tables[0].Rows)
        {
            // Creates an object from the DataRow
            EmailTemplateInfo modifyTemplate = new EmailTemplateInfo(templateDr);

            // Updates the properties
            modifyTemplate.TemplateDisplayName =
modifyTemplate.TemplateDisplayName.ToUpper();

            // Saves the changes
            EmailTemplateInfoProvider.SetEmailTemplateInfo(modifyTemplate);
        }

        return true;
    }

    return false;
}
```

The following example deletes a subscription template.

```
private bool DeleteSubscriptionTemplate()
{
    // Gets the subscription template
    EmailTemplateInfo deleteTemplate =
EmailTemplateInfoProvider.GetEmailTemplateInfo("MyNewSubscriptionTemplate",
CMSContext.CurrentSiteID);

    // Deletes the subscription template
    EmailTemplateInfoProvider.DeleteEmailTemplateInfo(deleteTemplate);

    return (deleteTemplate != null);
}
```

The following example deletes an unsubscription template.

```
private bool DeleteUnsubscriptionTemplate()
{
    // Gets the unsubscription template
    EmailTemplateInfo deleteTemplate =
EmailTemplateInfoProvider.GetEmailTemplateInfo("MyNewUnsubscriptionTemplate",
CMSContext.CurrentSiteID);
```

```
    // Deletes the unsubscription template
    EmailTemplateInfoProvider.DeleteEmailTemplateInfo(deleteTemplate);

    return (deleteTemplate != null);
}
```

The following example deletes an issue template.

```
private bool DeleteIssueTemplate()
{
    // Gets the issue template
    EmailTemplateInfo deleteTemplate =
EmailTemplateInfoProvider.GetEmailTemplateInfo("MyNewIssueTemplate",
CMSContext.CurrentSiteID);

    // Deletes the issue template
    EmailTemplateInfoProvider.DeleteEmailTemplateInfo(deleteTemplate);

    return (deleteTemplate != null);
}
```

8.36.14.4.3 Managing newsletters

The following example creates a static newsletter.

```
private bool CreateStaticNewsletter()
{
    EmailTemplateInfo subscriptionTemplate =
EmailTemplateInfoProvider.GetEmailTemplateInfo("MyNewSubscriptionTemplate",
CMSContext.CurrentSiteID);
    EmailTemplateInfo unsubscriptionTemplate =
EmailTemplateInfoProvider.GetEmailTemplateInfo("MyNewUnsubscriptionTemplate",
CMSContext.CurrentSiteID);
    EmailTemplateInfo myNewIssueTemplate =
EmailTemplateInfoProvider.GetEmailTemplateInfo("MyNewIssueTemplate",
CMSContext.CurrentSiteID);

    if ((subscriptionTemplate != null) && (unsubscriptionTemplate != null) &&
(myNewIssueTemplate != null))
    {
        // Creates a new static newsletter object
        NewsletterInfo newNewsletter = new NewsletterInfo();

        // Sets the properties
        newNewsletter.NewsletterDisplayName = "My new static newsletter";
        newNewsletter.NewsletterName = "MyNewStaticNewsletter";
        newNewsletter.NewsletterType = NewsletterType.TemplateBased;
        newNewsletter.NewsletterSubscriptionTemplateID =
subscriptionTemplate.TemplateID;
```

```
        newNewsletter.NewsletterUnsubscriptionTemplateID =
unsubscriptionTemplate.TemplateID;
        newNewsletter.NewsletterTemplateID = myNewIssueTemplate.TemplateID;
        newNewsletter.NewsletterSenderName = "Sender name";
        newNewsletter.NewsletterSenderEmail = "sender@localhost.local";
        newNewsletter.NewsletterSiteID = CMSContext.CurrentSiteID;

        // Saves the static newsletter
        NewsletterInfoProvider.SetNewsletterInfo(newNewsletter);

        return true;
    }

    return false;
}
```

The following example gets and updates a static newsletter.

```
private bool GetAndUpdateStaticNewsletter()
{
    // Gets the static newsletter
    NewsletterInfo updateNewsletter = NewsletterInfoProvider.GetNewsletterInfo
("MyNewStaticNewsletter", CMSContext.CurrentSiteID);
    if (updateNewsletter != null)
    {
        // Updates the properties
        updateNewsletter.NewsletterDisplayName =
updateNewsletter.NewsletterDisplayName.ToLower();

        // Saves the changes
        NewsletterInfoProvider.SetNewsletterInfo(updateNewsletter);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates static newsletters.

```
private bool GetAndBulkUpdateStaticNewsletters()
{
    // Prepares the parameters
    string where = "NewsletterName LIKE N'MyNewStaticNewsletter%'";

    // Gets the data
    DataSet newsletters = NewsletterInfoProvider.GetNewsletters(where, null, 0,
null);
    if (!DataHelper.DataSourceIsEmpty(newsletters))
    {
```

```
        // Loops through the items
        foreach (DataRow newsletterDr in newsletters.Tables[0].Rows)
        {
            // Creates an object from the DataRow
            NewsletterInfo modifyNewsletter = new NewsletterInfo(newsletterDr);

            // Updates the properties
            modifyNewsletter.NewsletterDisplayName =
modifyNewsletter.NewsletterDisplayName.ToUpper();

            // Saves the changes
            NewsletterInfoProvider.SetNewsletterInfo(modifyNewsletter);
        }

        return true;
    }

    return false;
}
```

The following example creates a dynamic newsletter.

```
private bool CreateDynamicNewsletter()
{
    EmailTemplateInfo subscriptionTemplate =
EmailTemplateInfoProvider.GetEmailTemplateInfo("MyNewSubscriptionTemplate",
CMSContext.CurrentSiteID);
    EmailTemplateInfo unsubscriptionTemplate =
EmailTemplateInfoProvider.GetEmailTemplateInfo("MyNewUnsubscriptionTemplate",
CMSContext.CurrentSiteID);

    if ((subscriptionTemplate != null) && (unsubscriptionTemplate != null))
    {
        // Creates a new dynamic newsletter object
        NewsletterInfo newNewsletter = new NewsletterInfo();

        // Sets the properties
        newNewsletter.NewsletterDisplayName = "My new dynamic newsletter";
        newNewsletter.NewsletterName = "MyNewDynamicNewsletter";
        newNewsletter.NewsletterType = NewsletterType.Dynamic;
        newNewsletter.NewsletterSubscriptionTemplateID =
subscriptionTemplate.TemplateID;
        newNewsletter.NewsletterUnsubscriptionTemplateID =
unsubscriptionTemplate.TemplateID;
        newNewsletter.NewsletterSenderName = "Sender name";
        newNewsletter.NewsletterSenderEmail = "sender@localhost.local";
        newNewsletter.NewsletterDynamicURL = "http://www.google.com";
        newNewsletter.NewsletterDynamicSubject = "My new dynamic issue";
        newNewsletter.NewsletterSiteID = CMSContext.CurrentSiteID;

        // Saves the dynamic newsletter
        NewsletterInfoProvider.SetNewsletterInfo(newNewsletter);
```

```
        return true;
    }

    return false;
}
```

The following example gets and updates a dynamic newsletter.

```
private bool GetAndUpdateDynamicNewsletter()
{
    // Gets the dynamic newsletter
    NewsletterInfo updateNewsletter = NewsletterInfoProvider.GetNewsletterInfo
("MyNewDynamicNewsletter", CMSContext.CurrentSiteID);
    if (updateNewsletter != null)
    {
        // Updates the properties
        updateNewsletter.NewsletterDisplayName =
updateNewsletter.NewsletterDisplayName.ToLower();

        // Saves the changes
        NewsletterInfoProvider.SetNewsletterInfo(updateNewsletter);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates dynamic newsletters.

```
private bool GetAndBulkUpdateDynamicNewsletters()
{
    // Prepares the parameters
    string where = "NewsletterName LIKE N'MyNewDynamicNewsletter%'";

    // Gets the data
    DataSet newsletters = NewsletterInfoProvider.GetNewsletters(where, null, 0,
null);
    if (!DataHelper.DataSourceIsEmpty(newsletters))
    {
        // Loops through the items
        foreach (DataRow newsletterDr in newsletters.Tables[0].Rows)
        {
            // Creates an object from the DataRow
            NewsletterInfo modifyNewsletter = new NewsletterInfo(newsletterDr);

            // Updates the properties
            modifyNewsletter.NewsletterDisplayName =
modifyNewsletter.NewsletterDisplayName.ToUpper();
```

```
            // Saves the changes
            NewsletterInfoProvider.SetNewsletterInfo(modifyNewsletter);
        }

        return true;
    }

    return false;
}
```

The following example deletes a dynamic newsletter.

```
private bool DeleteDynamicNewsletter()
{
    // Gets the dynamic newsletter
    NewsletterInfo deleteNewsletter = NewsletterInfoProvider.GetNewsletterInfo
("MyNewDynamicNewsletter", CMSContext.CurrentSiteID);

    // Deletes the dynamic newsletter
    NewsletterInfoProvider.DeleteNewsletterInfo(deleteNewsletter);

    return (deleteNewsletter != null);
}
```

The following example deletes a static newsletter.

```
private bool DeleteStaticNewsletter()
{
    // Gets the static newsletter
    NewsletterInfo deleteNewsletter = NewsletterInfoProvider.GetNewsletterInfo
("MyNewStaticNewsletter", CMSContext.CurrentSiteID);

    // Deletes the static newsletter
    NewsletterInfoProvider.DeleteNewsletterInfo(deleteNewsletter);

    return (deleteNewsletter != null);
}
```

8.36.14.4.4 Managing subscribers

The following example creates a subscriber.

```
private bool CreateSubscriber()
{
    // Creates a new subscriber object
    SubscriberInfo newSubscriber = new SubscriberInfo();
```

```
    // Sets the properties
    newSubscriber.SubscriberFirstName = "Name";
    newSubscriber.SubscriberLastName = "Surname";
    newSubscriber.SubscriberFullName = "Name Surname";
    newSubscriber.SubscriberEmail = "subscriber@localhost.local";
    newSubscriber.SubscriberSiteID = CMSContext.CurrentSiteID;

    // Saves the subscriber
    SubscriberInfoProvider.SetSubscriberInfo(newSubscriber);

    return true;
}
```

The following example gets and updates a subscriber.

```
private bool GetAndUpdateSubscriber()
{
    // Gets the subscriber
    SubscriberInfo updateSubscriber = SubscriberInfoProvider.GetSubscriberInfo
("subscriber@localhost.local", CMSContext.CurrentSiteID);
    if (updateSubscriber != null)
    {
        // Updates the properties
        updateSubscriber.SubscriberFullName =
updateSubscriber.SubscriberFullName.ToLower();

        // Saves the changes
        SubscriberInfoProvider.SetSubscriberInfo(updateSubscriber);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates subscribers.

```
private bool GetAndBulkUpdateSubscribers()
{
    // Prepares the parameters
    string where = "SubscriberEmail LIKE N'subscriber@localhost.local%'";

    // Gets the data
    DataSet subscribers = SubscriberInfoProvider.GetSubscribers(where, null);
    if (!DataHelper.DataSourceIsEmpty(subscribers))
    {
        // Loops through the items
        foreach (DataRow subscriberDr in subscribers.Tables[0].Rows)
        {
            // Creates an object from the DataRow
            SubscriberInfo modifySubscriber = new SubscriberInfo(subscriberDr);
```

```
            // Updates the properties
            modifySubscriber.SubscriberFullName =
modifySubscriber.SubscriberFullName.ToUpper();

            // Saves the changes
            SubscriberInfoProvider.SetSubscriberInfo(modifySubscriber);
        }

        return true;
    }

    return false;
}
```

The following example subscribes a subscriber to newsletter.

```
private bool SubscribeToNewsletter()
{
    // Gets the subscriber and newsletter
    SubscriberInfo subscriber = SubscriberInfoProvider.GetSubscriberInfo
("subscriber@localhost.local", CMSContext.CurrentSiteID);
    NewsletterInfo newsletter = NewsletterInfoProvider.GetNewsletterInfo
("MyNewStaticNewsletter", CMSContext.CurrentSiteID);

    if ((subscriber != null) && (newsletter != null))
    {
        // Subscribes to 'My new static newsletter'
        SubscriberInfoProvider.Subscribe(subscriber.SubscriberID,
newsletter.NewsletterID, DateTime.Now);

        return true;
    }

    return false;
}
```

The following example unsubscribes a subscriber from newsletter.

```
private bool UnsubscribeFromNewsletter()
{
    // Gets the subscriber and newsletter
    SubscriberInfo subscriber = SubscriberInfoProvider.GetSubscriberInfo
("subscriber@localhost.local", CMSContext.CurrentSiteID);
    NewsletterInfo newsletter = NewsletterInfoProvider.GetNewsletterInfo
("MyNewStaticNewsletter", CMSContext.CurrentSiteID);

    if ((subscriber != null) && (newsletter != null))
    {
        // Unubscribes from 'My new static newsletter'
```

```
        SubscriberInfoProvider.Unsubscribe(subscriber.SubscriberID,
newsletter.NewsletterID);

        return true;
    }

    return false;
}
```

The following example deletes a subscriber.

```
private bool DeleteSubscriber()
{
    // Gets the subscriber
    SubscriberInfo deleteSubscriber = SubscriberInfoProvider.GetSubscriberInfo
("subscriber@localhost.local", CMSContext.CurrentSiteID);

    // Deletes the subscriber
    SubscriberInfoProvider.DeleteSubscriberInfo(deleteSubscriber);

    return (deleteSubscriber != null);
}
```

8.36.14.4.5 Managing issues

The following example creates a static issue.

```
private bool CreateStaticIssue()
{
    // Gets the newsletter
    NewsletterInfo newsletter = NewsletterInfoProvider.GetNewsletterInfo
("MyNewStaticNewsletter", CMSContext.CurrentSiteID);

    if (newsletter != null)
    {
        // Creates a new static issue object
        IssueInfo newIssue = new IssueInfo();

        // Sets the properties
        newIssue.IssueSubject = "My new static issue";
        newIssue.IssueNewsletterID = newsletter.NewsletterID;
        newIssue.IssueSiteID = CMSContext.CurrentSiteID;
        newIssue.IssueText = "<?xml version=\"1.0\" encoding=\"utf-16\"?
><content><region id=\"content\">Issue text</region></content>";
        newIssue.IssueUnsubscribed = 0;
        newIssue.IssueSentEmails = 0;
        newIssue.IssueTemplateID = newsletter.NewsletterTemplateID;
        newIssue.IssueShowInNewsletterArchive = false;

        // Saves the static issue
```

```
        IssueInfoProvider.SetIssueInfo(newIssue);

        return true;
    }

    return false;
}
```

The following example gets and updates a static issue.

```
private bool GetAndUpdateStaticIssue()
{
    // Gets the newsletter
    NewsletterInfo newsletter = NewsletterInfoProvider.GetNewsletterInfo
("MyNewStaticNewsletter", CMSContext.CurrentSiteID);

    if (newsletter != null)
    {
        // Prepares the parameters
        string where = "IssueNewsletterID = " + newsletter.NewsletterID;

        // Gets the data
        DataSet issues = IssueInfoProvider.GetIssues(where, null);

        if (!DataHelper.DataSourceIsEmpty(issues))
        {
            // Creates an object from the DataRow
            IssueInfo updateIssue = new IssueInfo(issues.Tables[0].Rows[0]);

            if (updateIssue != null)
            {
                // Updates the properties
                updateIssue.IssueSubject = updateIssue.IssueSubject.ToLower();

                // Saves the changes
                IssueInfoProvider.SetIssueInfo(updateIssue);

                return true;
            }
        }
    }
    return false;
}
```

The following example gets and bulk updates static issues.

```
private bool GetAndBulkUpdateStaticIssues()
{
    // Gets the newsletter
    NewsletterInfo newsletter = NewsletterInfoProvider.GetNewsletterInfo
```

```
("MyNewStaticNewsletter", CMSContext.CurrentSiteID);

    if (newsletter != null)
    {
        // Prepares the parameters
        string where = "IssueNewsletterID = " + newsletter.NewsletterID;

        // Gets the data
        DataSet issues = IssueInfoProvider.GetIssues(where, null);
        if (!DataHelper.DataSourceIsEmpty(issues))
        {
            // Loops through the items
            foreach (DataRow issueDr in issues.Tables[0].Rows)
            {
                // Creates object from the DataRow
                IssueInfo modifyIssue = new IssueInfo(issueDr);

                // Updates the properties
                modifyIssue.IssueSubject = modifyIssue.IssueSubject.ToUpper();

                // Saves the changes
                IssueInfoProvider.SetIssueInfo(modifyIssue);
            }

            return true;
        }
    }

    return false;
}
```

The following example creates a dynamic issue.

```
private bool CreateDynamicIssue()
{
    // Gets the newsletter
    NewsletterInfo newsletter = NewsletterInfoProvider.GetNewsletterInfo
("MyNewDynamicNewsletter", CMSContext.CurrentSiteID);

    if (newsletter != null)
    {
        // Generates a dynamic issue
        EmailQueueManager.GenerateDynamicIssue(newsletter.NewsletterID);

        return true;
    }

    return false;
}
```

The following example gets and updates a dynamic issue.

```
private bool GetAndUpdateDynamicIssue()
{
    // Gets the newsletter
    NewsletterInfo newsletter = NewsletterInfoProvider.GetNewsletterInfo
("MyNewDynamicNewsletter", CMSContext.CurrentSiteID);

    if (newsletter != null)
    {
        // Prepares the parameters
        string where = "IssueNewsletterID = " + newsletter.NewsletterID;

        // Gets the data
        DataSet issues = IssueInfoProvider.GetIssues(where, null);

        if (!DataHelper.DataSourceIsEmpty(issues))
        {
            // Creates an object from the DataRow
            IssueInfo updateIssue = new IssueInfo(issues.Tables[0].Rows[0]);

            if (updateIssue != null)
            {
                // Updates the properties
                updateIssue.IssueSubject = updateIssue.IssueSubject.ToLower();

                // Saves the changes
                IssueInfoProvider.SetIssueInfo(updateIssue);

                return true;
            }
        }
    }
    return false;
}
```

The following example gets and bulk updates dynamic issues.

```
private bool GetAndBulkUpdateDynamicIssues()
{
    // Gets the newsletter
    NewsletterInfo newsletter = NewsletterInfoProvider.GetNewsletterInfo
("MyNewDynamicNewsletter", CMSContext.CurrentSiteID);

    if (newsletter != null)
    {
        // Prepares the parameters
        string where = "IssueNewsletterID = " + newsletter.NewsletterID;

        // Gets the data
        DataSet issues = IssueInfoProvider.GetIssues(where, null);
        if (!DataHelper.DataSourceIsEmpty(issues))
        {
            // Loops through the individual items
```

```
            foreach (DataRow issueDr in issues.Tables[0].Rows)
            {
                // Creates an object from the DataRow
                IssueInfo modifyIssue = new IssueInfo(issueDr);

                // Updates the properties
                modifyIssue.IssueSubject = modifyIssue.IssueSubject.ToUpper();

                // Saves the changes
                IssueInfoProvider.SetIssueInfo(modifyIssue);
            }

            return true;
        }
    }
    return false;
}
```

The following example deletes a dynamic issue.

```
private bool DeleteDynamicIssue()
{
    // Gets the newsletter
    NewsletterInfo newsletter = NewsletterInfoProvider.GetNewsletterInfo
("MyNewDynamicNewsletter", CMSContext.CurrentSiteID);

    if (newsletter != null)
    {
        // Prepares the parameters
        string where = "IssueNewsletterID = " + newsletter.NewsletterID;

        // Gets the data
        DataSet issues = IssueInfoProvider.GetIssues(where, null);

        if (!DataHelper.DataSourceIsEmpty(issues))
        {
            // Creates an object from the DataRow
            IssueInfo deleteIssue = new IssueInfo(issues.Tables[0].Rows[0]);

            // Deletes the dynamic issue
            IssueInfoProvider.DeleteIssueInfo(deleteIssue);

            return (deleteIssue != null);
        }
    }
    return false;
}
```

The following example deletes a static issue.

```
private bool DeleteStaticIssue()
```

```
{
    // Gets the newsletter
    NewsletterInfo newsletter = NewsletterInfoProvider.GetNewsletterInfo
("MyNewStaticNewsletter", CMSContext.CurrentSiteID);

    if (newsletter != null)
    {
        // Prepares the parameters
        string where = "IssueNewsletterID = " + newsletter.NewsletterID;

        // Gets the data
        DataSet issues = IssueInfoProvider.GetIssues(where, null);

        if (!DataHelper.DataSourceIsEmpty(issues))
        {
            // Creates an object from the DataRow
            IssueInfo deleteIssue = new IssueInfo(issues.Tables[0].Rows[0]);

            // Deletes the static issue
            IssueInfoProvider.DeleteIssueInfo(deleteIssue);

            return (deleteIssue != null);
        }
    }
    return false;
}
```

8.37 Notifications

8.37.1 Overview

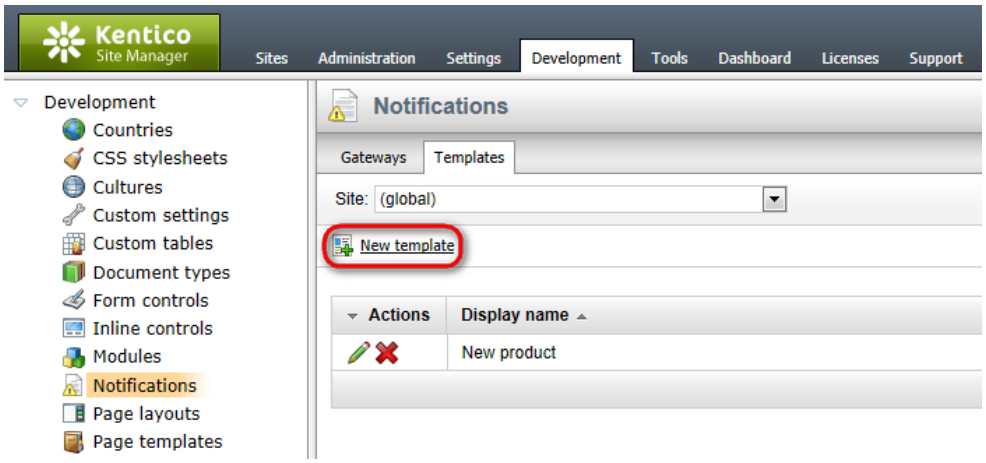
The Notifications module enables sending of notification messages using notification providers. Kentico CMS comes with a built-in e-mail notification gateway. However, you can create your custom gateways, such as an SMS or ICQ provider.

Please note that for one event, users can obtain notification messages from multiple providers. There are three built-in events (document created, document updated, document deleted). However, you can create and use your custom ones.

- To learn how to create notification message templates, please refer to the Creating a notification message template topic.
- To learn how to allow site visitors or CMS Desk administrators to subscribe to receiving notifications, please refer to the Subscribing users to content changes notifications topic.
- To learn how to manage users' notifications and choose providers from which the users will receive their notification messages, please refer to the Managing users' notifications topic.
- To learn how to create your own notification gateway, please refer to the Custom notification gateway subchapter.
- Module settings are described in the Settings topic.
- Module security is described in the Security topic.
- The internals and API of the Notifications module are described in the Notifications internals and API subchapter.

8.37.2 Creating a notification message template

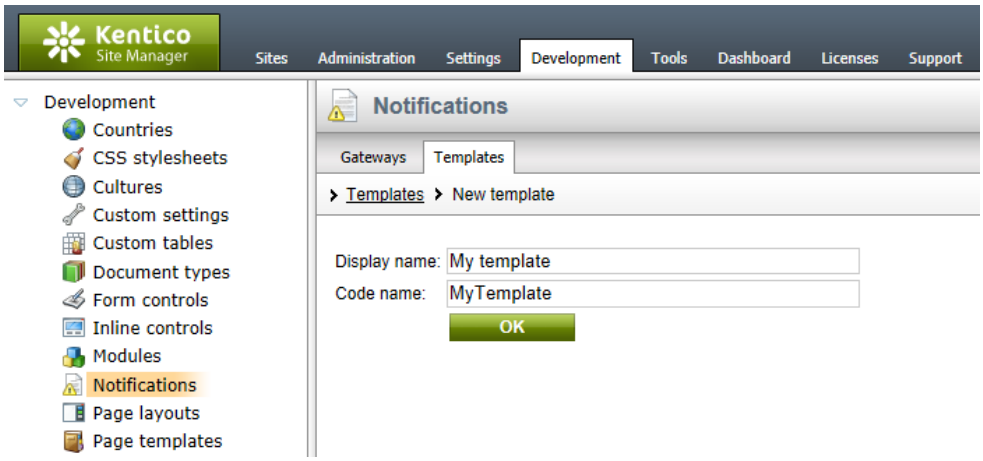
1. Go to **Site Manager -> Development -> Notifications** and switch to the **Templates** tab. Using the **Sites** drop-down list, you can select which site will be the template used by. If you choose **(global)**, the template will be available for all sites. When selected, click **New template**. You will be redirected to the **New template** form.



2. On the **New template** form, enter the following details:

- **Display name** - display name of the template
- **Code name** - code name of the template

Click **OK**. The template will be created.



3. Now you should see the same form that you've just filled, but in the **General** tab, while another tab called **Text** is also available. Switch to the **Text** tab.

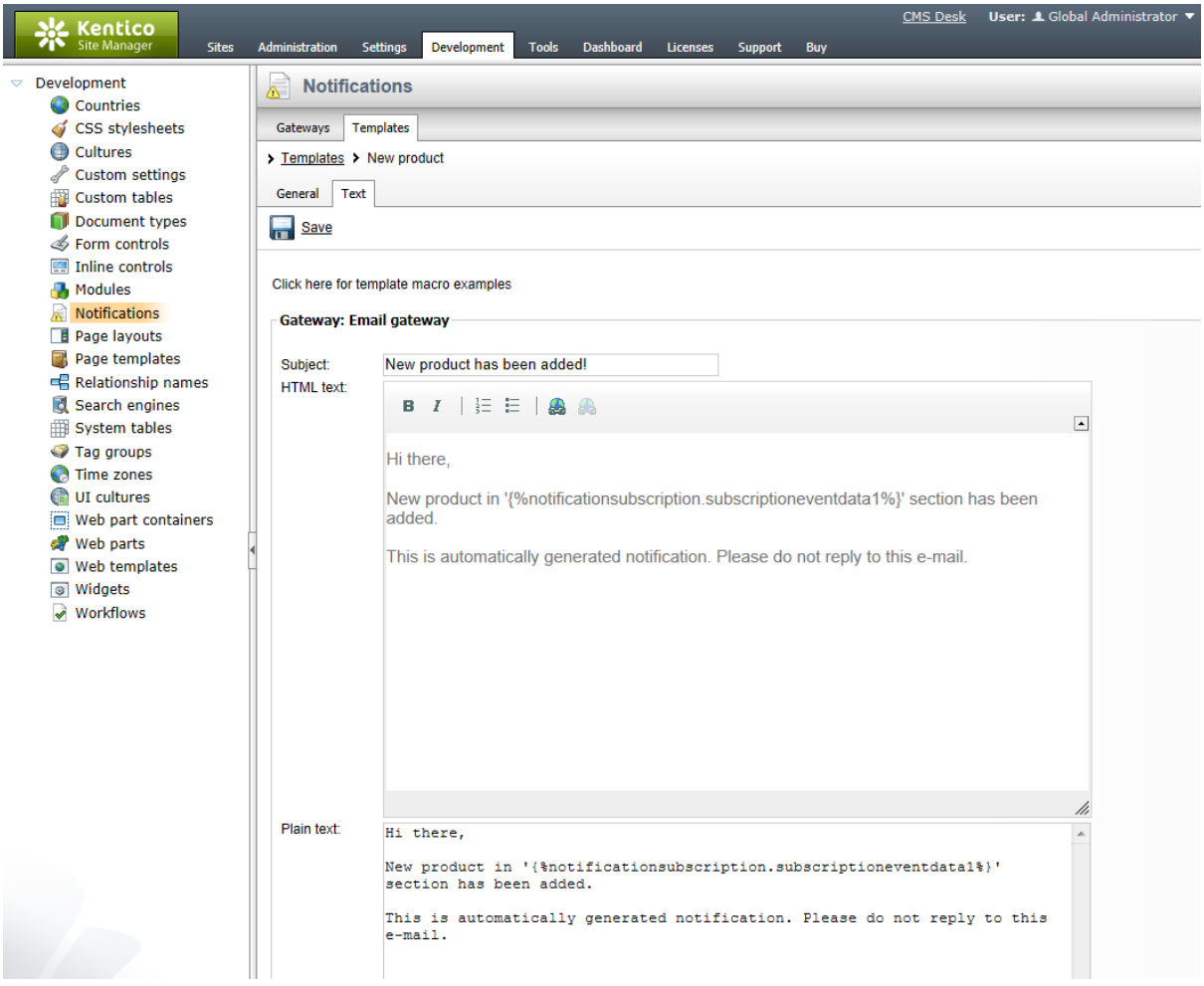
On this tab, the text of the notification message can be defined. You should see **sections for each of the defined gateways**. Texts can be set separately for each gateway, so that for one event, you can have different texts sent via e-mail and via SMS. Each of the sections contains the gateway name and some of the following three fields, depending on the gateway settings:

- **Subject** - message subject text

- **HTML text** - text of the message in HTML format
- **Plain text** - text of the message in plain text format

The following macros can be used in your notification templates:

- **{%notificationsubscription.SubscriptionID%}** - displays the value of specified data column from the *Notification_Subscription* table; this represents subscription information
- **{%notificationgateway.GatewayID%}** - displays the value of the specified data column from the *Notification_Gateway* table; this represents the notification gateway which performs the sending of the notification message
- **{%notificationuser.UserID%}** - displays the value of the specified user data column from the *CMS_Usertable* table; this represents the user the notification message is being sent to
- **{%notificationcustomdata.XXX%}** - displays the value of the specified data column from a custom data source. You can use the Document, Tree and SKU columns from *View_CMS_Tree_Joined* and also the document type's table (e.g. *CONTENT_article* for *CMS.article* document type).
- **{%documentlink%}** - displays the link to the document (for content event notifications only)

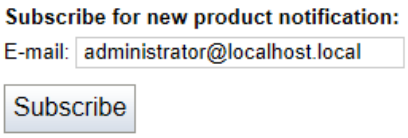


When you have entered appropriate text for each available gateway, click **Save**.

8.37.3 Subscribing users to content changes notifications

The **Content subscription** web part is a pre-configured version of the **Notification subscription** web part. It can be conveniently used to let site visitors subscribe to notifications about the following three events:

- **Document has been created**
- **Document has been updated**
- **Document has been deleted**



In the following example, you will learn how to add the **Content subscription** web part to your site and set it up. We will use the Corporate site sample website as the starting point. Let's presume that we want to enable site visitors to receive an e-mail whenever a new product is added.

Please note

In order for you to see the full functionality, it is necessary to have a SMTP server configured correctly. For more information on how to do this, please refer to the SMTP server configuration chapter.

1. Sign in to **CMS Desk** as the global administrator (login *administrator* with blank password by default). From the content tree on the left, select the **Products** title page and switch to the **Design** tab.

2. Add the **Content subscription** web part to the **Left zone** web part zone, below the **CSS list menu**. In the **Web part properties** dialog, set the following properties:

- **Site name** - (current site)
- **Path** - /Products/%
- **Document type** - CMS.Product
- **Event description** - Enter your e-mail address to receive notifications about new products:

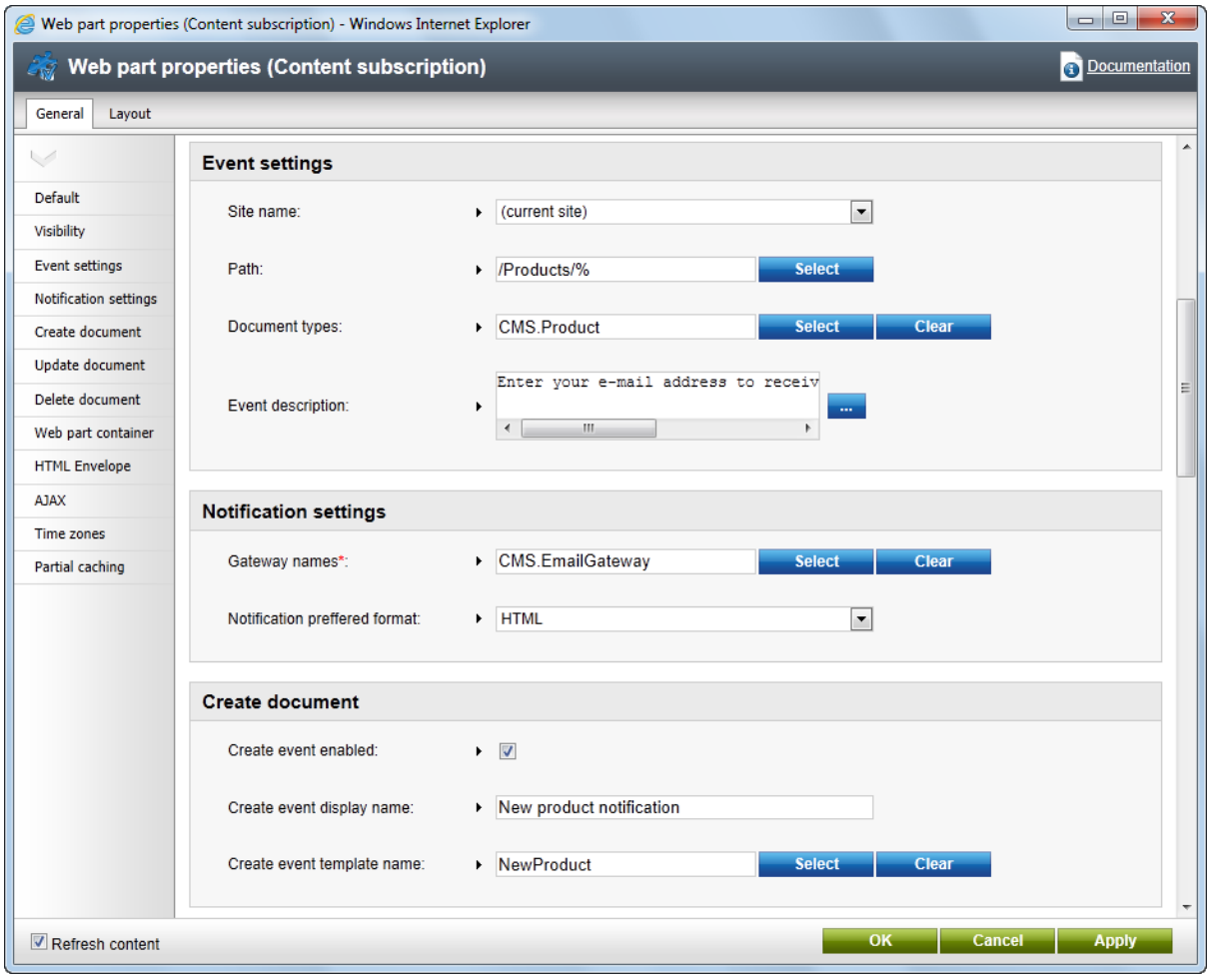
- **Gateway names** - CMS.EmailGateway
- **Notification preferred format** - HTML

- **Create event enabled** - true
- **Create event display name** - New product notification
- **Create event template name** - MyTemplate

and click **OK**.

Please note

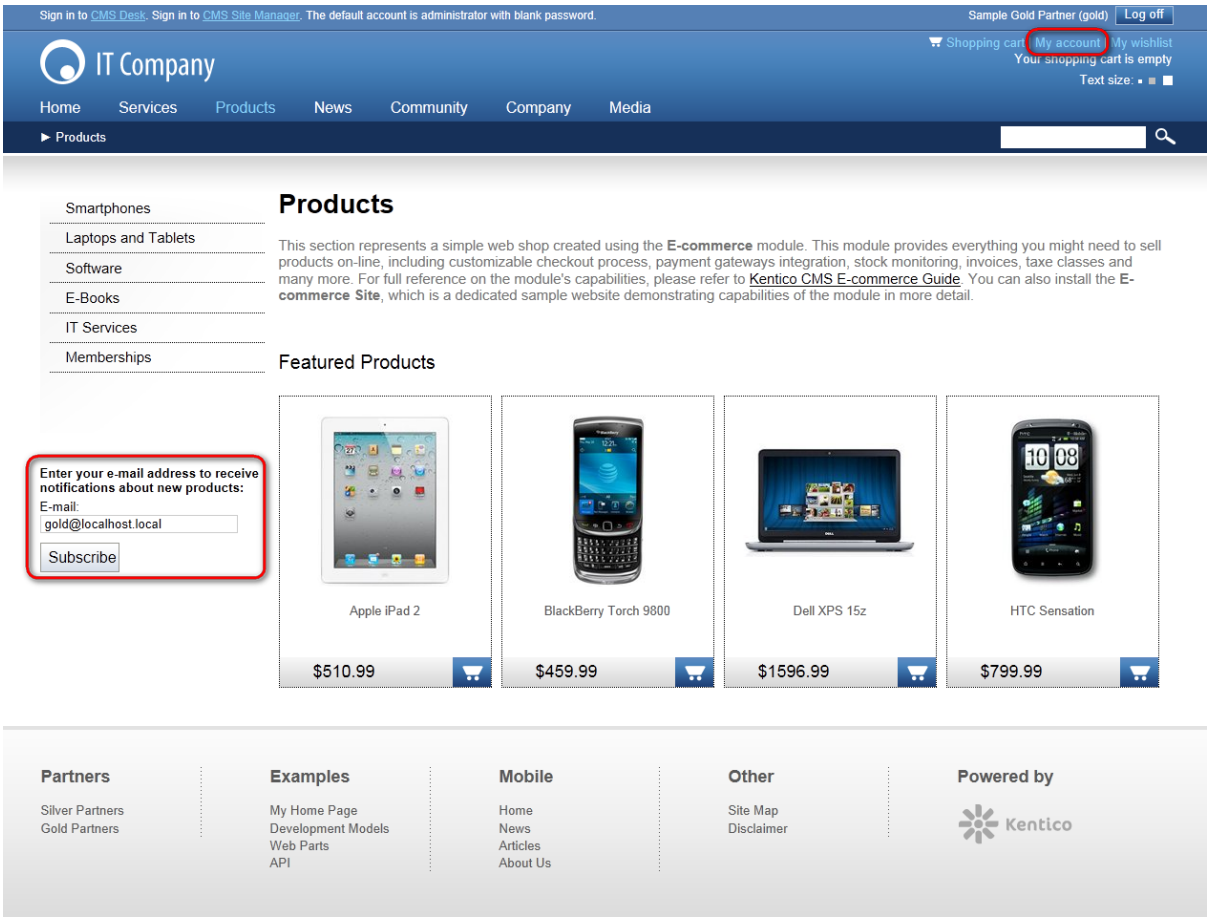
It is important that the template you have recently created according to the *Creating a notification message template* topic is global or assigned to the current site.



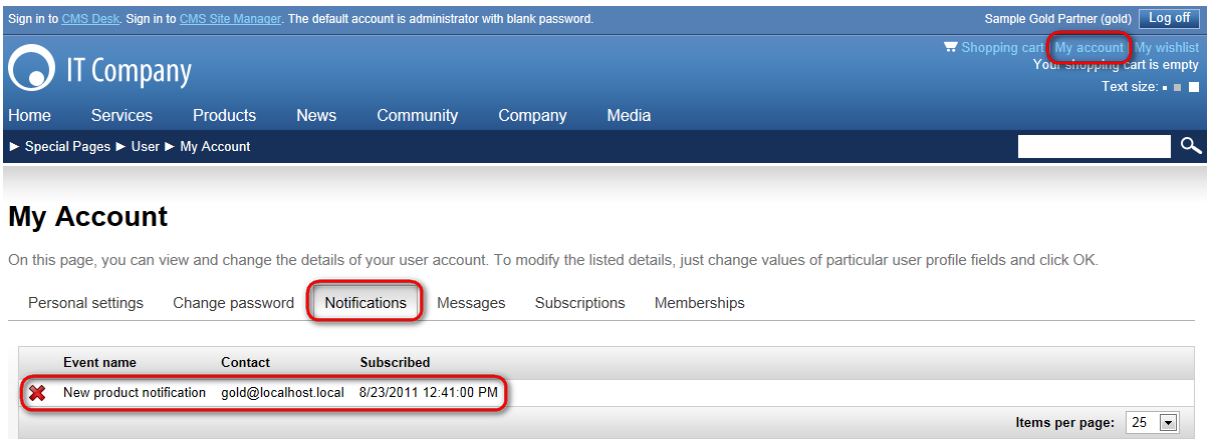
3. Let's switch to the **site visitor's** perspective now. Sign out of **CMS Desk**.

As notifications are available only for registered users, the web part is not visible unless you log in. Use the **My account** link at the top of the page to display a logon form and sign in; use e.g. user name *Gold* with blank password. Once signed in, switch to the **Products** page. You should see the **Content subscription** web part underneath the tree menu as highlighted in the screenshot below.

If you want to receive notifications about new products on this page, you can just type in your e-mail address into the **E-mail** field and click **Subscribe**. In case that you want to verify the functionality later on, enter your own e-mail address so that you can check your inbox for notification e-mails.



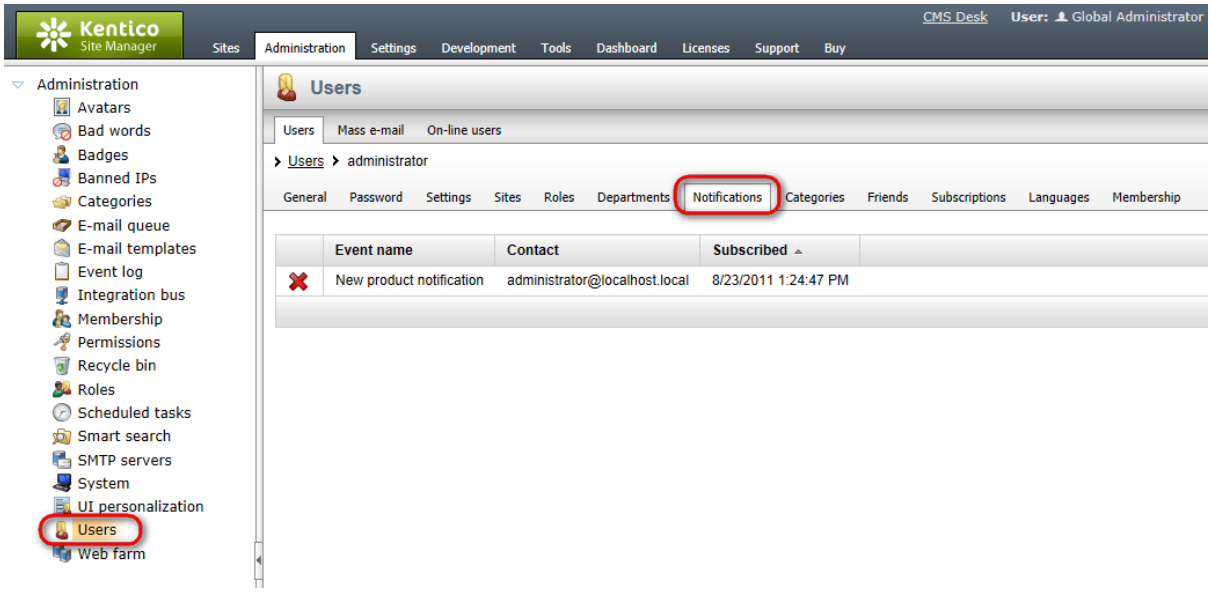
4. You can immediately verify your subscription by going to the **My account** section again. If you switch to the **Notifications** section of the **My account** web part, you should see the notification subscription present.



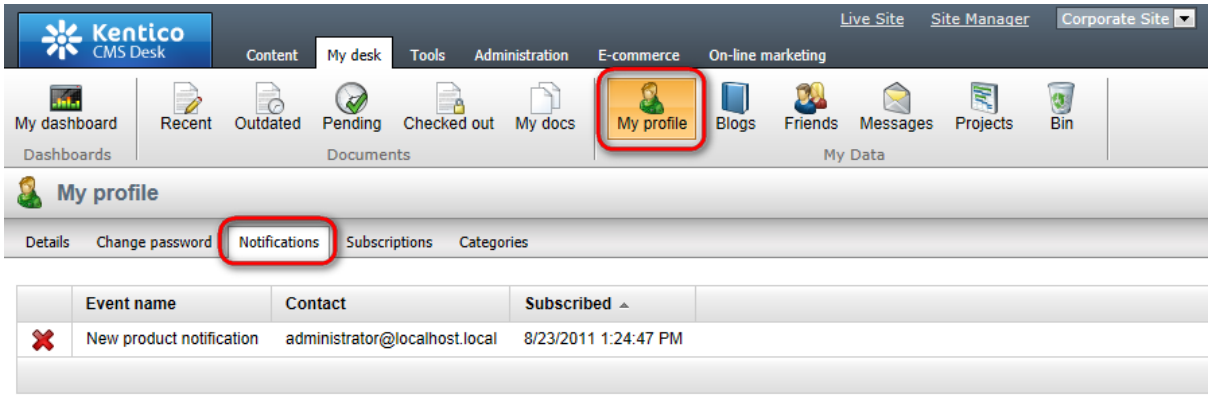
5. Now you can verify that the whole setup works. Log in to **CMS Desk** as the *administrator* again. From the content tree, select **Products** and click the **New** icon above the content tree. Choose to create some new product, e.g. a Cell phone, enter some sample data about the cell phone and click **Save**. Check your e-mail inbox in a few minutes. You should have received a new notification message about the newly added product.

8.37.4 Managing users' notifications

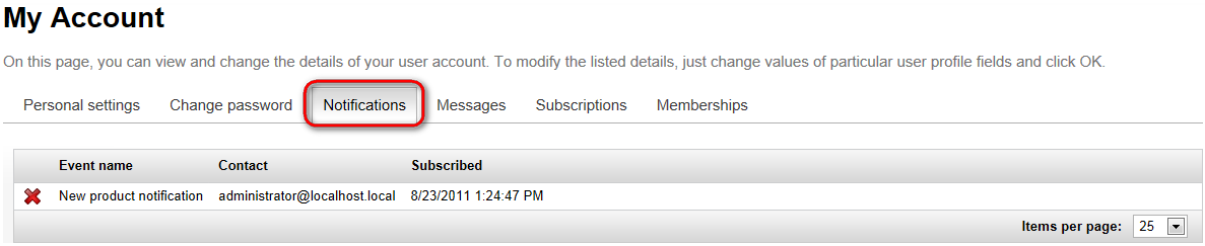
Site administrators can manage subscriptions of particular users in **Site Manager -> Administrator -> Users**. If you choose to **Edit** (✎) a user and switch to the **Notifications** tab, you will see all notifications that the current user is subscribed to. You can unsubscribe the user from a notification by clicking the **Delete** (✖) icon next to it.



CMS Desk users can manage their own subscriptions the same way in **CMS Desk -> My Desk -> Account -> Notifications**.



Site visitors registered to the site can manage their subscriptions the same way using the **My account** web part, on its **Notifications** tab.



8.37.5 Custom notification gateway

8.37.5.1 Overview

In this chapter, you will get familiar with the process of creating a custom notification gateway and using it on your site. The general description is supplied by code examples of a sample e-mail notification gateway.

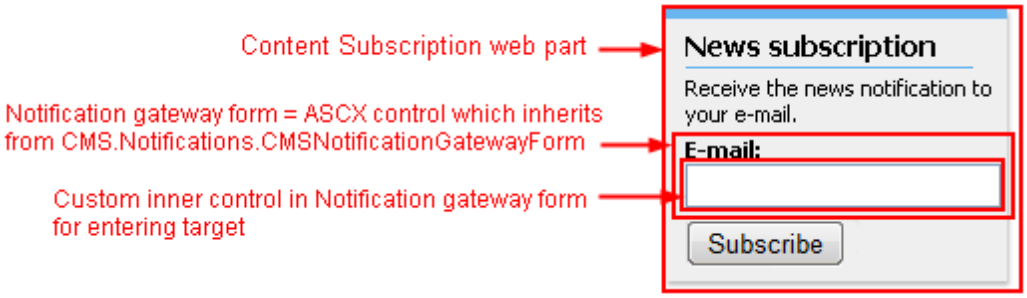
General steps

Here is a general overview of the steps that need to be taken when implementing a custom notification gateway:

1. Create a custom notification gateway form with your custom inner control(s) for entering the target where the notification messages should be sent.
2. Create a custom notification gateway class which handles loading of correct notification gateway form and sending notification messages.
3. Register the custom notification gateway in **Site Manager -> Development -> Notifications** and create a template for notification messages.
4. Use the gateway with the **Content subscription** web part on the live site.

8.37.5.2 Custom notification gateway form

In the picture below, you can see how the default **E-mail notification gateway form** is contained within the **Content subscription** web part.



The following steps need to be taken to create a custom notification gateway form:

1. Create a new web user control (*.ascx) and place it into your site folder which is located in the root of your web project. You can place it anywhere in your web project, but since the control is located in the site folder, it will be included in your site's export packages.
2. Set the control's class to inherit from the ***CMS.Notifications.CMSNotificationGatewayForm*** class.

3. There are two methods and one property you will need to override to reach the required functionality:
   - **object Value** - gets or sets the value from the custom inner control (e.g. gets or sets the text of the inner TextBox in the picture above)
   - **string Validate()** – validates the inner control's value and returns an error message if the value doesn't meet the requirements of the notification gateway (e.g. the TextBox value is validated for e-mail address format for the **E-mail notification gateway**)
   - **void ClearForm()** – your custom code inside this method should set the inner control to the default state; example: text of the TextBox should be cleared (set to the empty string)

Example - E-mail notification gateway form

The following code samples show how a custom e-mail notification gateway form can be implemented:

EmailNotificationForm.ascx

Please note: If you installed the Kentico CMS project as a web application, you need to rename the *CodeFile* property on the first line to *Codebehind* for the code example to be functional.

```
<%@ Control Language="C#" AutoEventWireup="true" CodeFile="EmailNotificationForm.asc
    Inherits="CMSModules_Notifications_Controls_NotificationSubscription_EmailNotifi
<asp:Label ID="lblEmail" runat="server" />
<asp:TextBox ID="txtEmail" runat="server" CssClass="EmailNotificationForm" EnableVie
```

EmailNotificationForm.ascx.cs

[C#]

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

using CMS.Notifications;
using CMS.GlobalHelper;
using CMS.CMSHelper;
using CMS.EventLog;

public partial class
CMSModules_Notifications_Controls_NotificationSubscription_EmailNotificationForm :
CMSNotificationGatewayForm
{
    #region "Variables"

    private bool mEnableMultipleEmails = false;
```

```
    #endregion


    #region "Properties"

    /// <summary>
    /// Gets or sets the e-mail(s) from/to textbox.
    /// </summary>
    public override object Value
    {
        get
        {
            return this.txtEmail.Text.Trim();
        }
        set
        {
            this.txtEmail.Text = ValidationHelper.GetString(value, "");
        }
    }


    /// <summary>
    /// Gets or sets the value which determines whether to allow multiple e-mails
separated with semicolon.
    /// </summary>
    public bool EnableMultipleEmails
    {
        get
        {
            return this.mEnableMultipleEmails;
        }
        set
        {
            this.mEnableMultipleEmails = value;
        }
    }

    #endregion


    protected void Page_Load(object sender, EventArgs e)
    {
        this.lblEmail.Text = (this.EnableMultipleEmails ? ResHelper.GetString
("general.emails") : ResHelper.GetString("general.email")) + ResHelper.Colon;

        // Fill in the default e-mail
        if ((!RequestHelper.IsPostBack()) && (CMSContext.CurrentUser != null) &&
(!String.IsNullOrEmpty(CMSContext.CurrentUser.Email)))
        {
            this.txtEmail.Text = CMSContext.CurrentUser.Email;
        }
    }
```

```
    #region "Public methods"

    /// <summary>
    /// Checks whether the input is correct e-mail address (or multiple e-mail
addresses).
    /// </summary>
    public override string Validate()
    {
        if (this.EnableMultipleEmails)
        {
            if (!ValidationHelper.AreEmails(this.txtEmail.Text.Trim()))
            {
                return ResHelper.GetString("notifications.emailgateway.formats");
            }
        }
        else
        {
            if (!ValidationHelper.IsEmail(this.txtEmail.Text.Trim()))
            {
                return ResHelper.GetString("notifications.emailgateway.format");
            }
        }

        return String.Empty;
    }


    /// <summary>
    /// Clears the e-mail textbox field.
    /// </summary>
    public override void ClearForm()
    {
        this.txtEmail.Text = "";
    }

    #endregion
}
```

8.37.5.3 Custom notification gateway class

The following steps need to be taken to create a custom notification gateway class:

1. Create a new library (assembly) as a part of your solution and a new class inside this library.
2. Set your class to inherit from the **CMS.Notifications.CMSNotificationGateway** abstract class.
3. There are two methods you will need to override to reach the required functionality:

- **void SendNotification()** – sends a single notification; it is automatically called after the specified event is raised
- **CMSNotificationGatewayForm GetNotificationGatewayForm()** – loads and returns notification gateway form for the notification gateway

4. Compile the library.
5. Ensure the library file (*.dll) is included in the **/Bin** directory.

Example - E-mail notification gateway class

The following code sample shows how a custom e-mail notification gateway class can be implemented:

EmailNotificationGateway.cs

[C#]

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Web.UI;

using CMS.EventLog;
using CMS.EmailEngine;
using CMS.GlobalHelper;
using CMS.SiteProvider;
using CMS.SettingsProvider;
using CMS.Notifications;

namespace EmailNotification
{
    /// <summary>
    /// Base class for e-mail notification gateway.
    /// </summary>
    public class EmailNotificationGateway : CMSNotificationGateway
    {
        /// <summary>
        /// Returns the e-mail gateway form.
        /// </summary>
        public override CMSNotificationGatewayForm GetNotificationGatewayForm()
        {
            try
            {
                Control ctrl =
this.NotificationSubscriptionControl.Page.LoadControl("~/CMSModules/Notifications/
Controls/NotificationSubscription/EmailNotificationForm.ascx");
                ctrl.ID = ValidationHelper.GetIdentificator("notif" +
this.NotificationGatewayObj.GatewayName);

                return (CMSNotificationGatewayForm)ctrl;
            }
            catch (Exception ex)
            {
                try
                {
                    // Log the event
                    EventLogProvider eventLog = new EventLogProvider();
                    eventLog.LogEvent("EmailGateway", "EXCEPTION", ex);
                }
                catch
                {
                    // Unable to log the event
```

```
                }
            }
            return null;
        }


        /// <summary>
        /// Sends the notification via e-mail.
        /// </summary>
        public override void SendNotification()
        {
            try
            {
                if (this.NotificationSubscriptionObj != null)
                {
                    // Get template text
                    NotificationTemplateTextInfo templateText =
NotificationTemplateTextInfoProvider.GetNotificationTemplateTextInfo
(this.NotificationGatewayObj.GatewayID,
this.NotificationSubscriptionObj.SubscriptionTemplateID);
                    if (templateText != null)
                    {
                        // Get the site name
                        string siteName = null;
                        SiteInfo si = SiteInfoProvider.GetSiteInfo
(this.NotificationSubscriptionObj.SubscriptionSiteID);
                        if (si != null)
                        {
                            siteName = si.SiteName;
                        }

                        // Create message object
                        EmailMessage message = new EmailMessage();

                        // Get sender from settings
                        message.From = SettingsKeyProvider.GetStringValue(siteName
+ ".CMSSendEmailNotificationsFrom");

                        // Do not send the e-mail if there is no sender specified
                        if (message.From != "")
                        {
                            // Initialize message
                            message.Recipients =
this.NotificationSubscriptionObj.SubscriptionTarget;

                            // Body
                            if
((this.NotificationSubscriptionObj.SubscriptionUseHTML) &&
                                (this.NotificationGatewayObj.GatewaySupportsHTMLTe
xt) &&
                                (templateText.TemplateHTMLText != ""))
                            {
                                // HTML format, set Body property, resolve macros
if possible
```

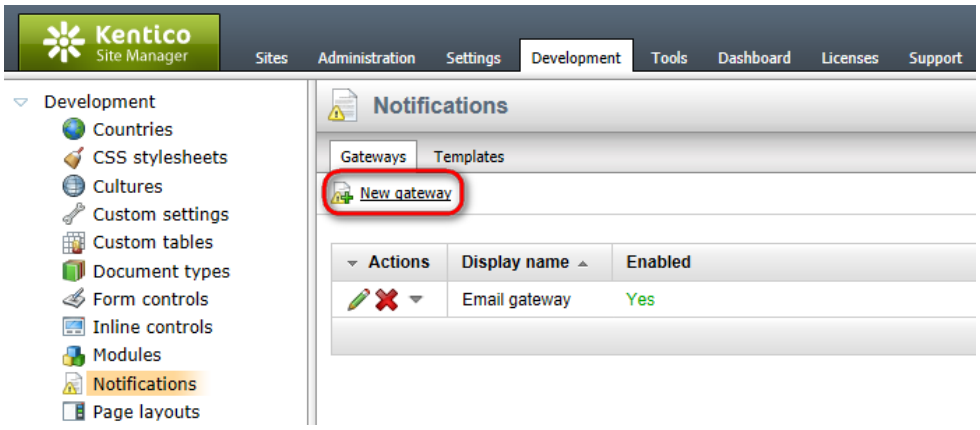
```
                                message.EmailFormat = EmailFormatEnum.Html;
                                if (this.Resolver != null)
                                {
                                    message.Body = this.Resolver.ResolveMacros
(templateText.TemplateHTMLText);
                                }
                                else
                                {
                                    message.Body = templateText.TemplateHTMLText;
                                }
                            }
                            else
                            {
                                // Plaintext format, set PlainTextBody property,
resolve macros if possible
                                message.EmailFormat = EmailFormatEnum.PlainText;
                                if (this.Resolver != null)
                                {
                                    message.PlainTextBody =
this.Resolver.ResolveMacros(templateText.TemplatePlainText);
                                }
                                else
                                {
                                    message.PlainTextBody =
templateText.TemplatePlainText;
                                }
                            }

                            // Subject, resolve macros if possible
                            if (this.Resolver != null)
                            {
                                message.Subject = this.Resolver.ResolveMacros
(templateText.TemplateSubject);
                            }
                            else
                            {
                                message.Subject = templateText.TemplateSubject;
                            }

                            // Send email via Email engine API
                            EmailSender.SendEmail(siteName, message);
                        }
                    }
                }
            }
            catch (Exception ex)
            {
                try
                {
                    // Log the event
                    EventLogProvider eventLog = new EventLogProvider();
                    eventLog.LogEvent("EmailGateway", "EXCEPTION", ex);
                }
                catch
                {
```

```
                    // Unable to log the event
                }
            }
        }
    }
}
```

8.37.5.4 Registering a custom gateway

When you have developed a custom notification gateway, you need to take the following steps to register your gateway in the system:

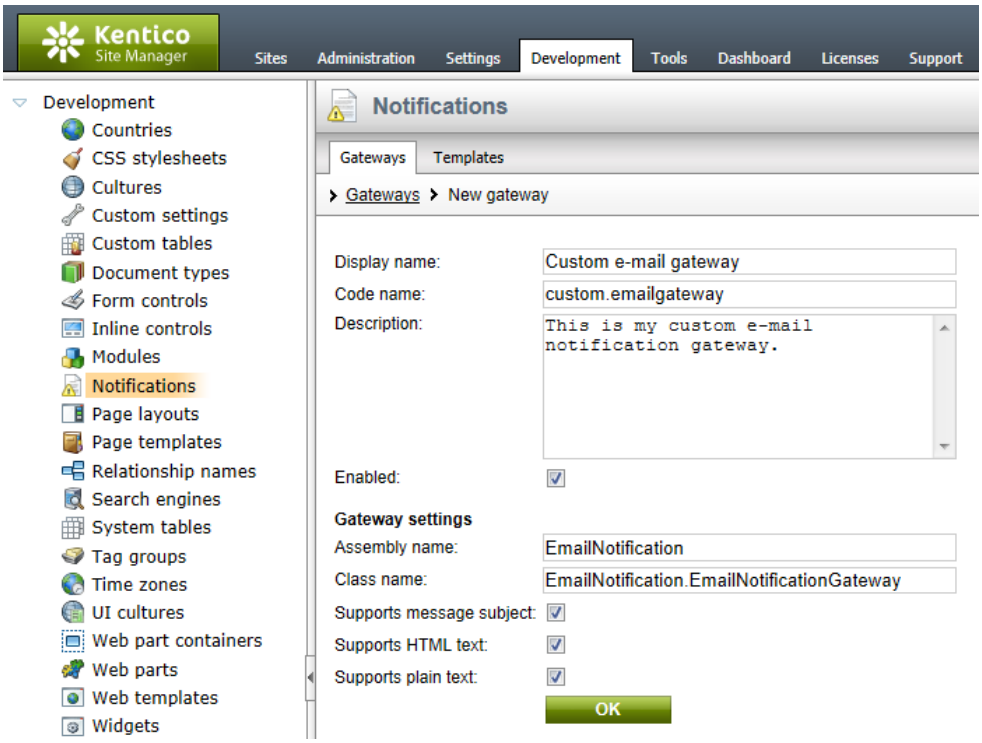
1. Go to **Site Manager -> Development -> Notifications**. At the top of the **Gateways** tab, click **New gateway**. The **New gateway form** will be displayed.



2. Enter the following details into the **New gateway** form:

- **Display name** - display name of the notification gateway
- **Code name** - code name of the gateway
- **Description** - text describing the gateway
- **Enabled** - if unchecked, the notification gateway is not functional - this can be useful if you want to temporarily disable the gateway so that no messages will be sent, e.g. when you are performing some administration tasks; if checked, the gateway works normally

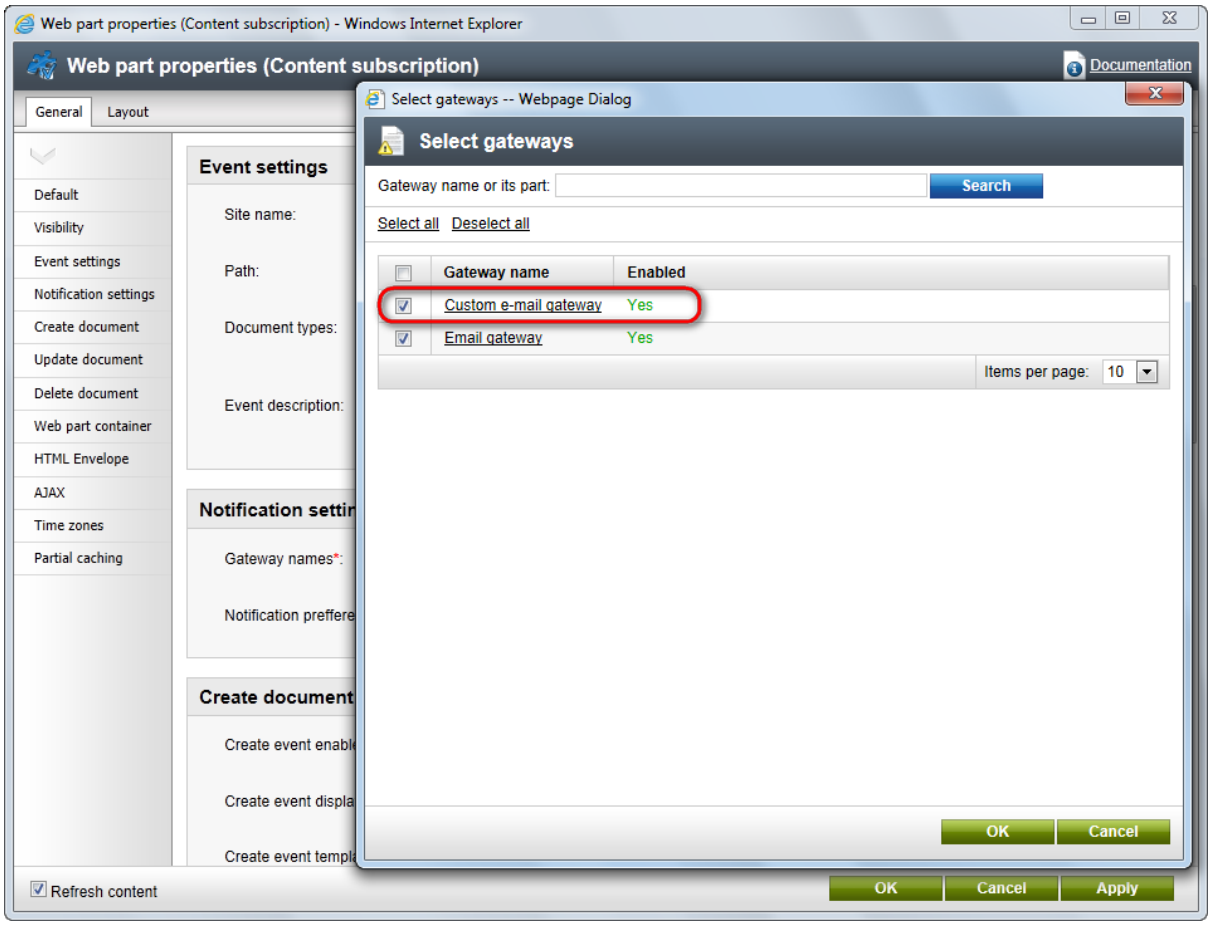
- **Assembly name** - name of the assembly in that the gateway code is stored
- **Class name** - name of the class containing the gateway code; must be entered including the assembly name, as you can see in the screenshot below
- **Supports message subject** - enable if the gateway's message format supports message subjects
- **Supports HTML text** - enable if the gateway supports messages in HTML format (e.g. for e-mails)
- **Supports plain text** - enable if the gateway supports plain text format (e.g. for SMS)



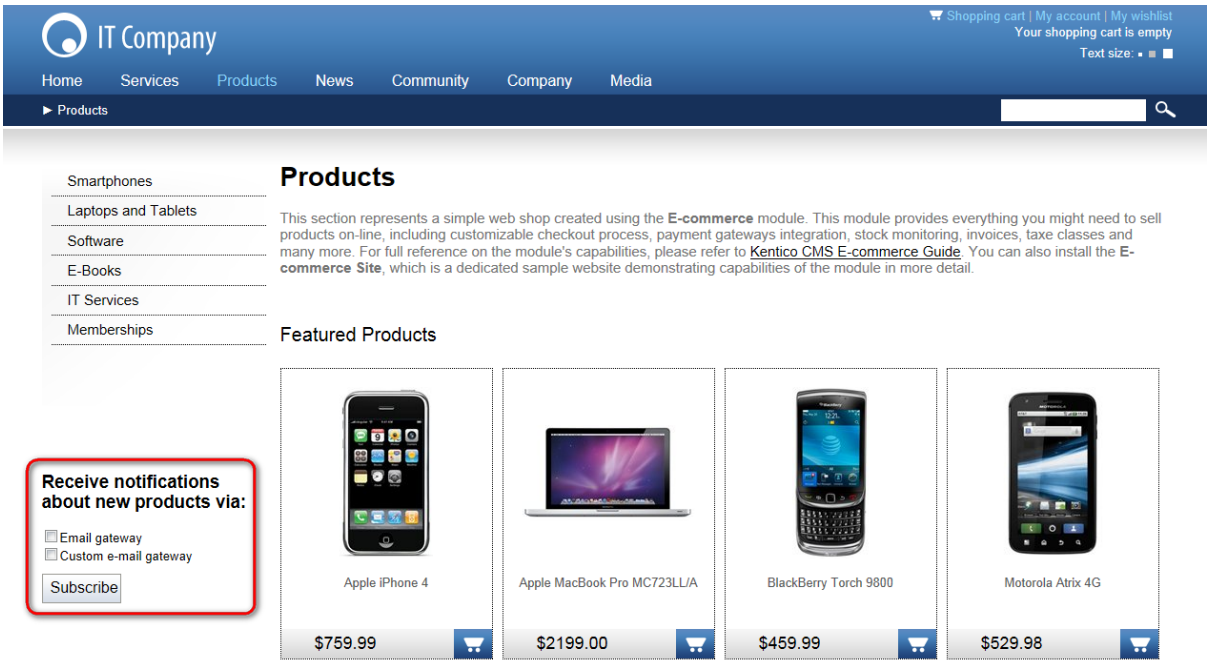
3. After entering all the details, click **OK** to confirm the changes you have made.

8.37.5.5 Using the gateway on your site

On the live site, you can enable users to subscribe using the **Content subscription** web part, just as described here. You only need to enable your gateway in the **Gateway names** property of the web part.



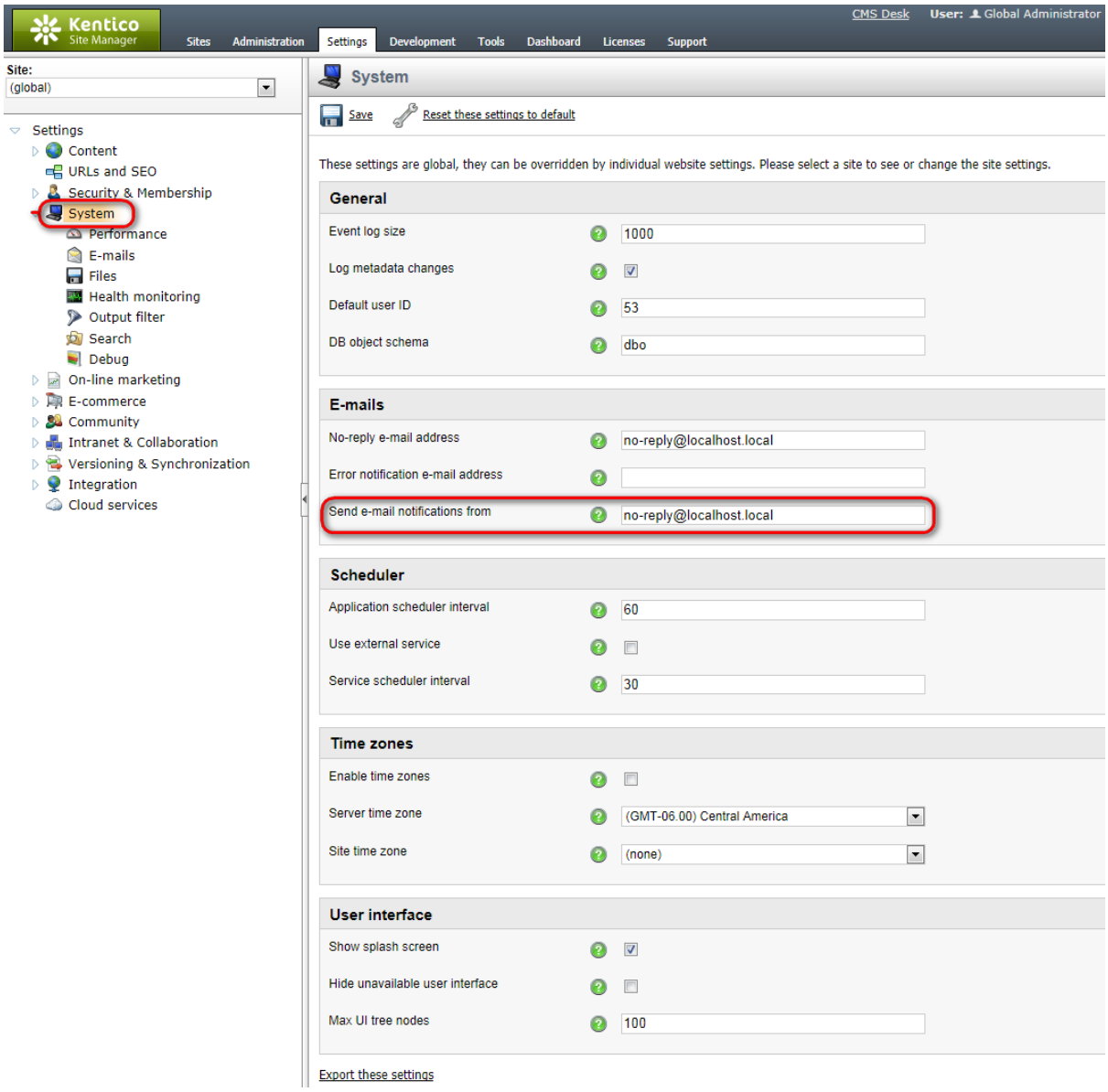
On the live site, the web part displays a check-box for each gateway when multiple gateways are selected. If you enable a particular check-box, the gateway's form is displayed, as you can see in the screenshot below.



8.37.6 Settings

The **Notifications** module has only one setting to be done in **Site Manager -> Settings -> System**:

- **Send e-mail notifications from** - sets the e-mail address that will be used as the sender address (the **From** field) of notification e-mails



8.37.7 Security

The **Notifications** module has no permissions to be set in **CMS Desk/Site Manager -> Administration -> Permissions**.

Subscription to notifications is **allowed only for registered users**. This is why the **Content subscription** and **Notification subscription** web parts are **hidden to public anonymous users** by default.

8.37.8 Notifications internals and API

8.37.8.1 Overview

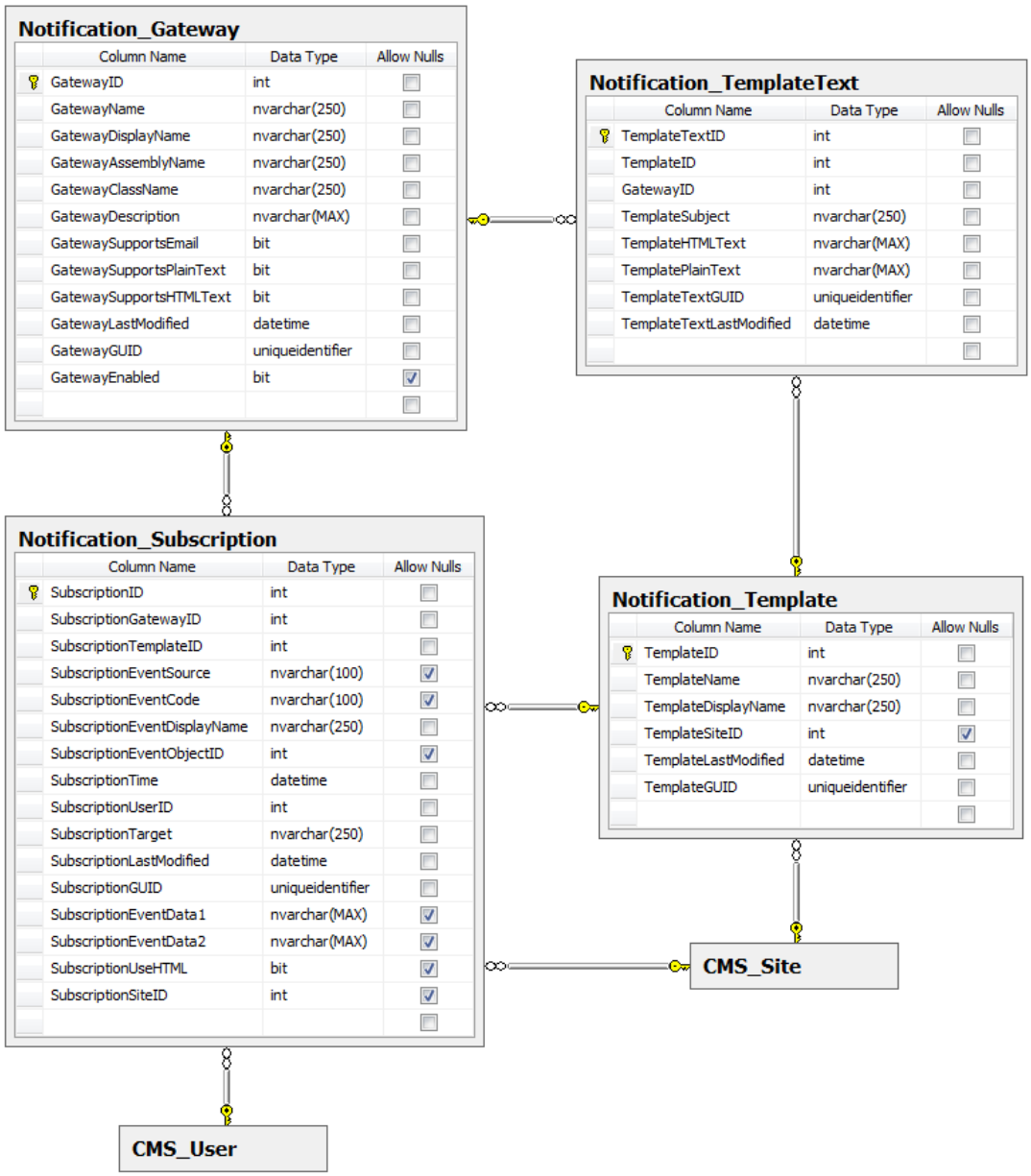
In this chapter, you will learn which database tables and API classes are used in the Notifications module. You will also see the most common API examples.

Advanced API examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all methods from the module classes, please refer to Kentico CMS API Reference.

8.37.8.2 Database tables

The following database tables are used in the Notifications module:

- **Notification_Gateway** - contains records representing notification gateways.
- **Notification_Template** - contains records representing notification templates.
- **Notification_TemplateText** - contains records representing notification template texts.
- **Notification_Subscription** - contains records representing notification subscriptions.



8.37.8.3 API classes

If you are not familiar with the database table data management by Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

> **Please note**
>
> The Notifications module classes use the **CMS.Notifications** namespace.

Notification_Gateway table API:

- **NotificationGatewayInfo** - represents one notification gateway.
- **NotificationGatewayInfoProvider** - provides management of notification gateways.

Notification_Template table API:

- **NotificationTemplateInfo** - represents one notification template.
- **NotificationTemplateInfoProvider** - provides management of notification templates.

Notification_TemplateText table API:

- **NotificationTemplateTextInfo** - represents one notification template text.
- **NotificationTemplateTextInfoProvider** - provides management of notification template texts.

Notification_Subscription table API:

- **NotificationSubscriptionInfo** - represents one notification subscription.
- **NotificationSubscriptionInfoProvider** - provides management of notification subscriptions.

Other classes:

- **NotificationSender** - used for asynchronous sending of notification messages.
- **CMSNotificationGateway** - the base class for notification gateways. All custom notification gateways need to inherit from this class.
- **CMSEmailNotificationGateway** - a built-in e-mail notification gateway.

8.37.8.4 API examples

8.37.8.4.1 Overview

These topics show examples of how the Notifications module API can be used:

- Managing notification templates

Please note

All API examples can be simulated in the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Development\Notifications\Default.aspx.cs***.

The Notifications module API examples use the following namespaces:

```
using System;
```

```
using System.Data;

using CMS.GlobalHelper;
using CMS.UIControls;
using CMS.CMSHelper;
using CMS.SiteProvider;
using CMS.Notifications;
```

8.37.8.4.2 Managing notification templates

The following example creates a notification template.

```
private bool CreateNotificationTemplate()
{
    // Create new notification template object
    NotificationTemplateInfo newTemplate = new NotificationTemplateInfo();

    // Set the properties
    newTemplate.TemplateDisplayName = "My new template";
    newTemplate.TemplateName = "MyNewTemplate";
    newTemplate.TemplateSiteID = CMSContext.CurrentSiteID;

    // Create the notification template
    NotificationTemplateInfoProvider.SetNotificationTemplateInfo(newTemplate);

    return true;
}
```

The following example gets and updates a notification template.

```
private bool GetAndUpdateNotificationTemplate()
{
    // Get the notification template
    NotificationTemplateInfo updateTemplate =
NotificationTemplateInfoProvider.GetNotificationTemplateInfo("MyNewTemplate",
CMSContext.CurrentSiteID);
    if (updateTemplate != null)
    {
        // Update the property
        updateTemplate.TemplateDisplayName =
updateTemplate.TemplateDisplayName.ToLower();

        // Update the notification template
        NotificationTemplateInfoProvider.SetNotificationTemplateInfo
(updateTemplate);

        return true;
    }

    return false;
```

```
}
```

The following example gets and bulk updates notification templates.

```
private bool GetAndBulkUpdateNotificationTemplates()
{
    // Prepare the parameters
    string where = "TemplateName LIKE N'MyNewTemplate%'";
    where = SqlHelperClass.AddWhereCondition(where, "TemplateSiteID = " +
CMSContext.CurrentSiteID, "AND");

    // Get the data
    DataSet templates = NotificationTemplateInfoProvider.GetTemplates(where,
null);
    if (!DataHelper.DataSourceIsEmpty(templates))
    {
        // Loop through the individual items
        foreach (DataRow templateDr in templates.Tables[0].Rows)
        {
            // Create object from DataRow
            NotificationTemplateInfo modifyTemplate = new NotificationTemplateInfo
(templateDr);

            // Update the property
            modifyTemplate.TemplateDisplayName =
modifyTemplate.TemplateDisplayName.ToUpper();

            // Update the notification template
            NotificationTemplateInfoProvider.SetNotificationTemplateInfo
(modifyTemplate);
        }

        return true;
    }

    return false;
}
```

The following example deletes a notification template.

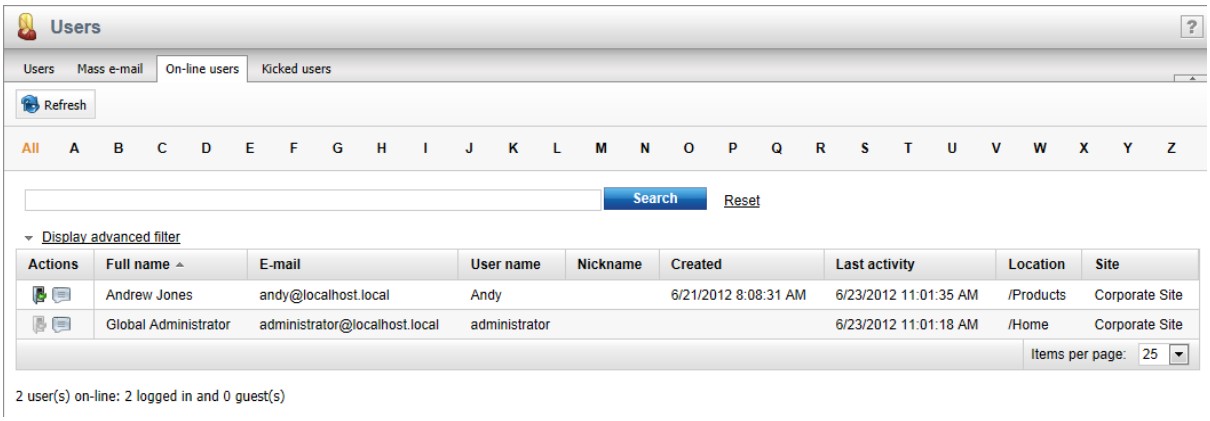
```
private bool DeleteNotificationTemplate()
{
    // Get the notification template
    NotificationTemplateInfo deleteTemplate =
NotificationTemplateInfoProvider.GetNotificationTemplateInfo("MyNewTemplate",
CMSContext.CurrentSiteID);

    // Delete the notification template
    NotificationTemplateInfoProvider.DeleteNotificationTemplateInfo
(deleteTemplate);
```

```
    return (deleteTemplate != null);
}
```

8.38 On-line users

8.38.1 Overview

The On-line users module allows you to monitor users currently connected to the website. This can be useful for various site administration purposes. In addition to providing information, the module also provides a way to temporarily kick a specific user off the website.



The module identifies a new user when a new session between a client browser and the server is started. The user is considered off-line if the session expires or when the user logs off. This means that a user is still considered on-line for some time when they close their web browser without signing off.

To utilize the module on your site, you need to perform the following simple steps:

1. Enable and configure on-line user monitoring in the system. Learn how in the Enabling the On-line users module topic.

2. Keep track of users on the On-line users tab or display information about them on your website using the On-line users web part.

The On-line users internals and API sub-chapter provides information about the database tables and classes used by the module and examples of how on-line users can be managed using the API.

For more general information about users and site membership, please refer to the Development -> Membership, permissions and security chapter of this guide.

8.38.2 Enabling the On-line users module

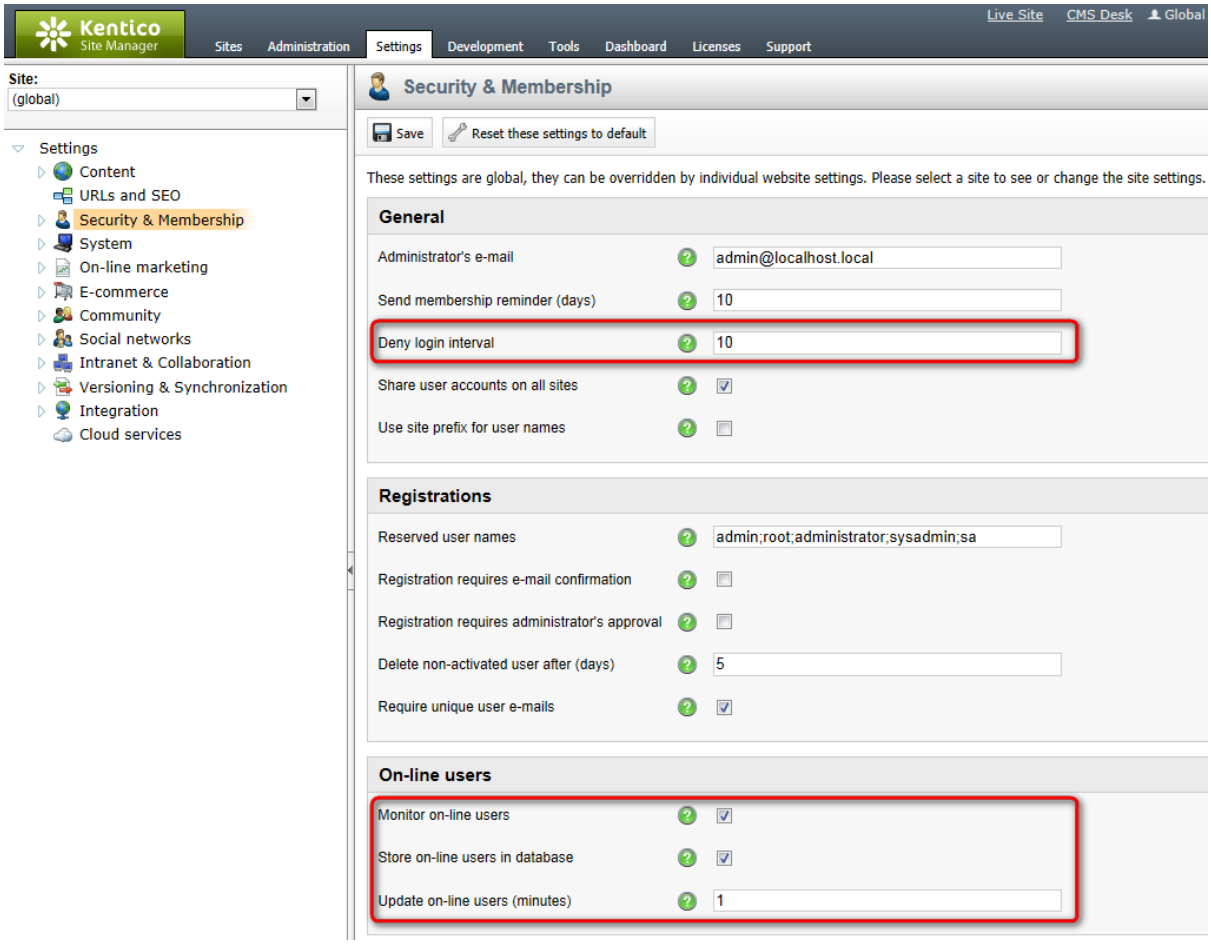
The On-line users module can be enabled in **Site Manager -> Settings -> Security & Membership** by checking the **Monitor on-line users** check-box.

If you are running the system on a web farm, you also have to check the **Store on-line users in**

database check-box. This causes information about on-line users to be saved in the database and also allows the system to provide more detailed information about guests (anonymous users).

The **Deny login interval** property determines how long users will not be able to log-in after they are kicked. The value is entered in minutes.

Finally, the **Update on-line users (minutes)** property defines how often information about users accessing the site is reloaded. The value is entered in minutes. When running the system on a web farm, you need to enter the same value which is set for the **Remove expired sessions** scheduled task (you can read the value in **Site Manager -> Administration -> Scheduled tasks -> edit (🖉) Remove expired sessions -> Task interval -> Every: X minute**).



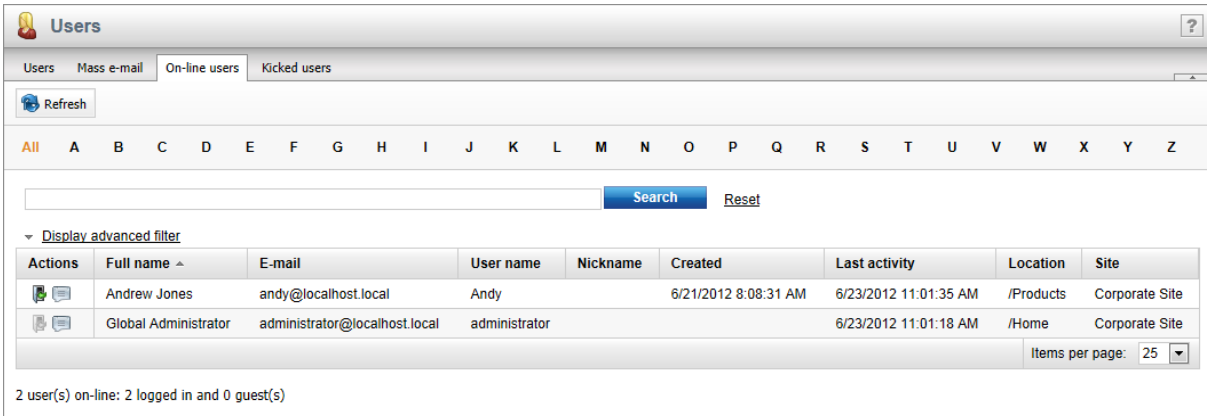
8.38.3 On-line users tab

If you go to **Site Manager/CMS Desk -> Administration -> Users** and switch to the **On-line users** tab, you can see a list of all users that are currently accessing the website. By clicking one of the letters at the top of the page, you can view only those users whose user names begin with the selected letter.

Below the letters, you can find a filter. This filter can further limit displayed users according to the specified criteria. The filter can work in two modes between which you can switch by clicking the **Display advanced filter** or **Display simplified filter** links respectively.

Simplified filter

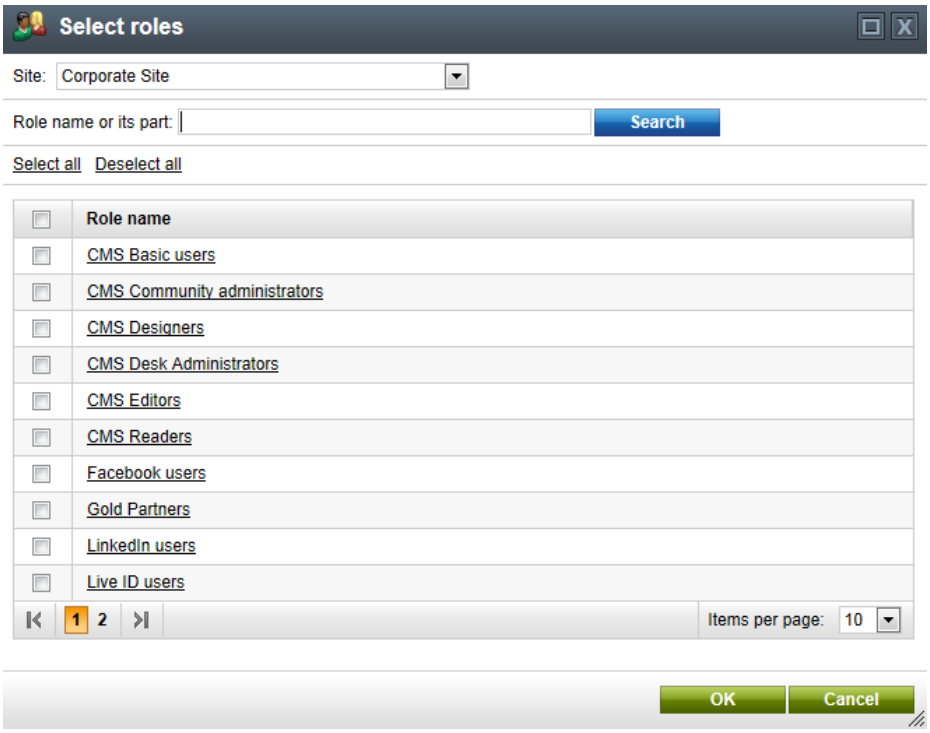
This option offers only one field. Enter an expression into the field to find users according to their **User Name**, **Full Name**, **E-mail Address** or **Nickname**.



Advanced filter

The advanced filter offers searching for users based on specific **User Name**, **Full Name**, **E-mail Address** or **Nickname** values.

You can also filter on-line users by the roles to which they belong. To specify roles into either the **In roles** or **Not in roles** fields, use the **Add roles** button to open the role selection dialog. Click **OK** to confirm the selection.

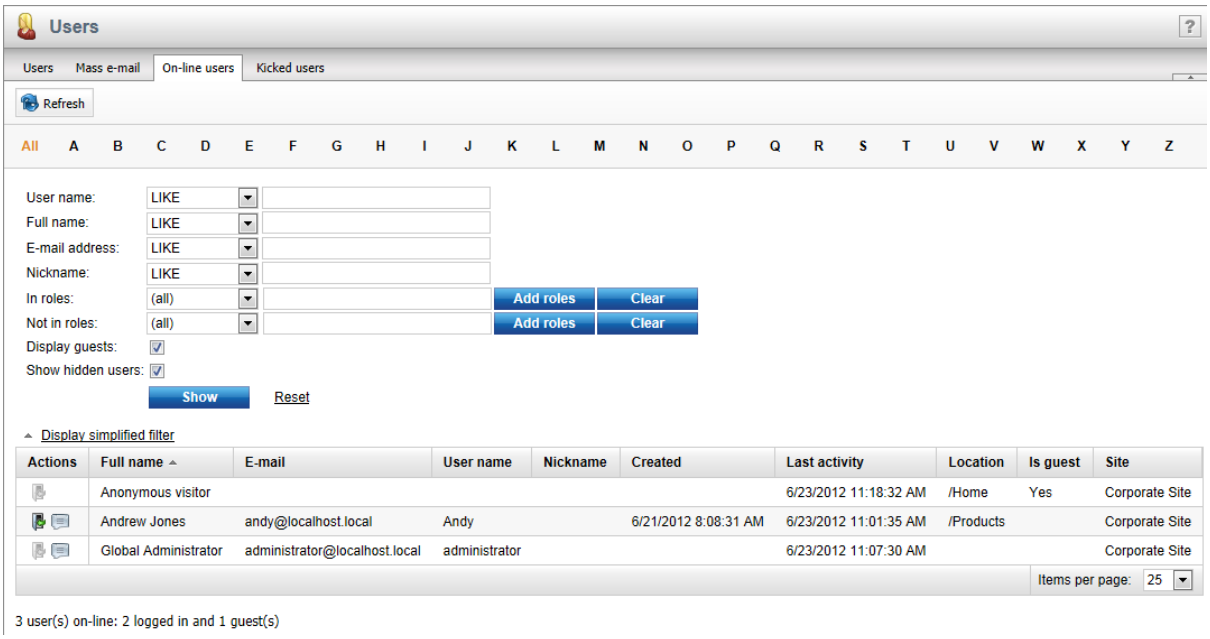


If you have selected more than one user role, you can also select if the displayed users should (not) be members of **All** or **Any** of the selected roles.

The **Display guests** option indicates whether individual visitors who are not logged in should be displayed. This is only possible if the on-line user data is stored in the database. If Contact management is being used on the website, the system also provides the names and e-mail addresses of guests based on the available contact data.

The **Show hidden users** checkbox determines whether the list should include users who are flagged as hidden (e.g. administrators or other internal user accounts).

When you have entered all filtering data, click the **Show** button. Only those users that match the specified criteria are listed.

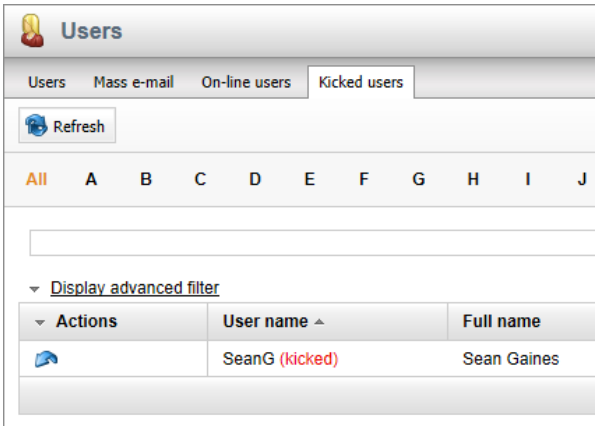


Kicking a user

You can kick users off the website by clicking the **Kick** ( ) action next to one of the listed users. This means that the user is logged out of the website and will not be able to log back in for the duration specified in the **Site Manager -> Settings -> Security & Membership -> Deny login interval** setting.

After doing so, a (kicked) label is added in red letters after the given user's name. When the on-line user data is being stored in the database, you can view all users who are currently kicked on the **Kicked users** tab. Otherwise the kicked users are included in the main list of on-line users.

To revoke a kick, click the **Take back** ( ) action.



If the reason for kicking a user is serious enough, you may want to consider using the Banned IPs module to permanently block the user from accessing the website.

Chatting with on-line users

Clicking the **Initiate chat** action next to an on-line user allows you to directly communicate with the given person through a chat window. To work correctly, support chat must be enabled for the website and the page that the user is currently viewing needs to contain the the Initiated chat web part. Please see the Modules -> Chat -> Support chat chapter for more information.

8.38.4 On-line users web part

The module comes with the On-line users web part. In the web part selection dialog, you can find the web part in the **Membership -> Users** category. The image below displays the output of the web part enclosed in the **Orange box** web part container.

The web part displays a summary defined by the **Additional info text** property, followed by a list of users that are currently on-line. Users are displayed based on the transformation specified in the **Transformation name** property.

The web part has the following related properties:

Transformation name	Sets the transformation used to display the on-line users. You can use the default **CMS.Root.OnLineUsers** transformation, which displays user names separated by spaces.
Path	If you enter an alias path here, only users that are accessing pages found under the specified path will be displayed.
Select top N	Sets the maximum number of users that can be selected and displayed.

Additional info text	Sets the text which will be displayed above the list of on-line users. You can use the following formatting macros that will be resolved into the appropriate number: {0} - number of all connected users {2} - number of connected registered users {1} - number of connected anonymous users
No users on-line text	Sets the text that will be displayed if no users are currently on-line.
Columns	Lists which columns should be loaded from the **CMS_User** or **CMS_UserSettings** tables along with user records. Column names need to be separated by commas (,). Specifying a list without unnecessary columns may significantly improve performance. These columns may be used in the code of the specified transformation to display data related to the on-line users.

8.38.5 On-line users internals and API

8.38.5.1 Overview

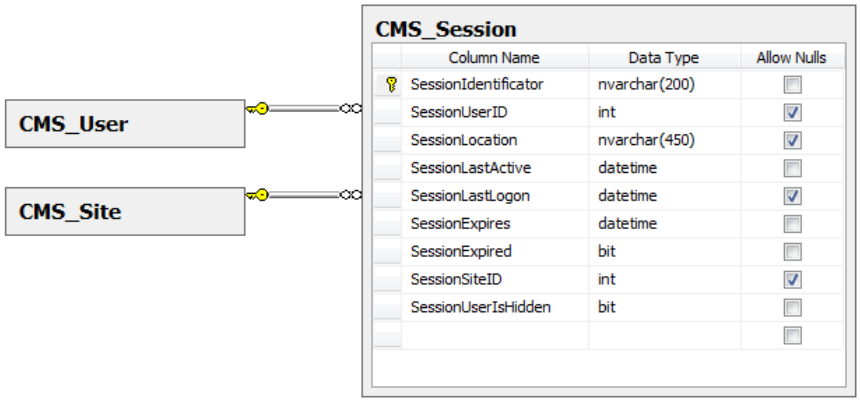
In this chapter, you will learn which database tables and API classes are used in the On-line users module. You will also see the most common API examples.

Further API-related information and examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all members of the module's classes, please refer to Kentico CMS API Reference.

8.38.5.2 Database tables

The following database table is used to store information about on-line users:

- **CMS_Session** - contains records representing users that are currently connected to a website. On-line users are only stored in the database if the *Site Manager -> Settings -> Security & Membership -> Store on-line users in database* setting is enabled. This is necessary when running the system on a web farm.



8.38.5.3 API classes

The functionality for managing sessions, which includes the monitoring of on-line users, is provided by the **SessionManager** class found in the **CMS.CMSHelper** namespace.

User and site objects can be handled via the corresponding Info and Provider classes from the **CMS.SiteProvider** namespace. If you are not familiar with this type of database table management, we strongly recommend that you refer to the Database table API topic first.

CMS_User table API:

- **UserInfo** - represents one user object.
- **UserInfoProvider** - provides management functionality for users.

CMS_Site table API:

- **SiteInfo** - represents one site object.
- **SiteInfoProvider** - provides management functionality for sites.

8.38.5.4 API examples

8.38.5.4.1 Overview

The following topic shows examples of how the API of the On-line users module can be used:

- Managing on-line users

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Administration\Membership\Default.aspx.cs***.

The on-line user API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.GlobalHelper;
using CMS.CMSHelper;
using CMS.SiteProvider;
```

8.38.5.4.2 Managing on-line users

The following example gets a dataset of on-line users and updates their properties.

```
private bool GetOnlineUsers()
{
    string where = "";
    int topN = 10;
    string orderBy = "";
    string location = "";
    string siteName = CMSContext.CurrentSiteName;
    bool includeHidden = true;
    bool includeKicked = false;

    // Get DataSet of on-line users
    DataSet users = SessionManager.GetOnlineUsers(where, orderBy, topN, location,
siteName, includeHidden, includeKicked);
    if (!DataHelper.DataSourceIsEmpty(users))
    {
        foreach (DataRow userDr in users.Tables[0].Rows)
        {
            // Create object from DataRow
            UserInfo modifyUser = new UserInfo(userDr);

            // Update the properties
            modifyUser.FullName = modifyUser.FullName.ToUpper();

            // Save the changes
            UserInfoProvider.SetUserInfo(modifyUser);
        }

        return true;
    }

    return false;
}
```

The following example checks if a specified user is currently on-line.

```
private bool IsUserOnline()
{
    bool includeHidden = true;

    // Get user and site objects
    UserInfo user = UserInfoProvider.GetUserInfo("MyNewUser");
    SiteInfo site = SiteInfoProvider.GetSiteInfo
(CMSContext.CurrentSiteName);

    if ((user != null) && (site != null))
    {
        // Check if user is online
        return SessionManager.IsUserOnline(site.SiteName, user.UserID,
includeHidden);
    }

    return false;
```

```
}
```

The following example kicks a user from a site.

```
private bool KickUser()
{
    // Get the user
    UserInfo kickedUser = UserInfoProvider.GetUserInfo
(CMSContext.CurrentUser.UserID);

    if (kickedUser != null)
    {
        // Kick the user
        SessionManager.KickUser(kickedUser.UserID);

        return true;
    }

    return false;
}
```

8.39 Polls

8.39.1 Overview

The Polls module allows you to create, edit and publish polls on your websites created with Kentico CMS. There are two types of poll in Kentico CMS:

- **Global poll** - the poll is shared across more websites.
- **Site poll** - the poll has a relationship with a particular site. Such poll is not accessible from other sites.

The module can display voting statistics in the form of graph, which represents either the absolute number of votes or percentage. Besides, the module enables you to control whether single or multiple answers will be allowed on the page and you have the possibility to deny multiple votes for one site visitor.

By placing polls on the pages of your website, you give site visitors the chance to vote for certain options. These options can be related directly to the current page, to the contents of the whole website or simply to anything on which you would like visitors of your web pages to express their opinion. Polls in Kentico CMS are customizable and the module supports integration with the built-in WYSIWYG editor.

- To learn how to perform administrative tasks related to polls, please refer to the Managing polls topic.
- To learn how to make your polls available from your web pages, please refer to the Publishing polls topic.
- If you would like to see an example of how to integrate polls on your site, please refer to the Adding a poll to your site topic.
- To learn how to make your polls in more than one cultural version, please refer to the Multilingual support topic.
- If you would like to learn about the security possibilities of the Polls module, please refer to the Security topic.

A practical example of the use of the Polls module can be found in Kentico CMS Personal Site Guide. You can see a step-by-step tutorial on how to add a poll to your personal website in Personal Site Guide -> Adding web parts -> Adding a poll.

You will need to have the Personal Site sample website installed to follow the Personal Site Guide.

Several more practical examples of the use of the Polls module are available in Kentico CMS Community Site Guide; please note that these examples do not concern the whole functionality of the module but focus on its use in a broader context of the Community Site sample website:

- See e.g. Part 2 -> Pre-development tasks -> Creating a sample poll in this guide: You will learn how to create a sample poll that you will publish on the Home page of you website later on.
- See also e.g. Creating the Home page in the same section of the guide: Here you will learn how to

publish the previously created sample poll on the Home page your website.

You will need to have the Community Site sample website installed to follow the Community Site Guide.

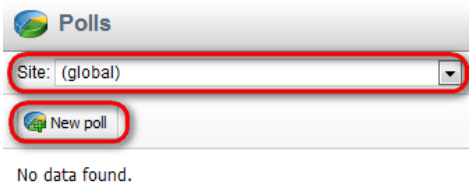
8.39.2 Managing polls

In this topic, you will learn how to create a new poll, edit poll properties, define poll answers, define who can vote and you will also learn how to share a global poll between sites, and how to preview a poll.

By default, both global and site polls are allowed in Kentico CMS; to learn how to customize these settings, please refer to the Security topic. Both types of polls can be managed in **CMS Desk -> Tools -> Polls**.

Creating a new poll

First, you need to choose whether the new poll will be site-based or global. This can be done using the **Site** drop-down list:



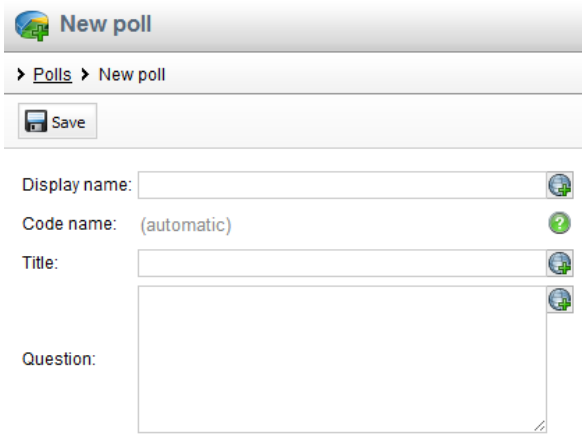
(global and this site)	Both global and current site polls are listed in the poll list. The option does not allow to create a new poll.
(global)	Only global polls are listed in the poll list. The option allows to create a new global poll.
current site	Only current site polls are listed in the poll list. The option allows to create a new site poll.

Please note

The **Site** drop-down list may be disabled for a particular user due to security reasons. For more details, please refer to this topic.

Now click **New poll** and enter the following values:

Display name	The name of the poll displayed to poll administrators.
Code name	The name of the poll used in the code.
Title	The title of the poll displayed in the poll view; optional field.
Question	The poll question displayed in the poll view.

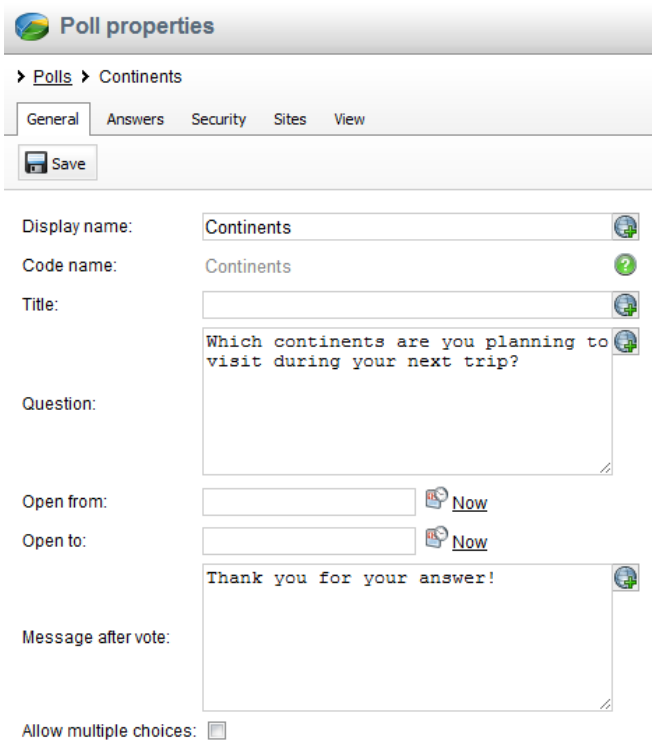


After entering the required values, **Save** the changes you have made.

Editing poll properties

On the **General** tab, you can define more poll details by entering the following values:

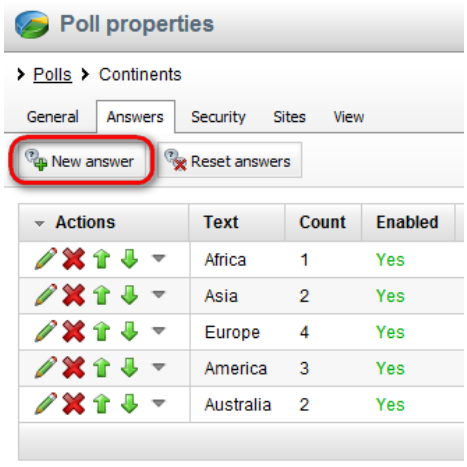
Display name	The name of the poll displayed to poll administrators.
Code name	The name of the poll used in the code.
Title	The title of the poll displayed in the poll view; optional field.
Question	The poll question displayed in the poll view; optional field.
Open from	Indicates when the voting is opened; optional field.
Open to	Indicates when the voting is closed; optional field.
Message after vote	The message displayed after vote; optional field.
Allow multiple choices	Indicates if a visitor can select more than one option.



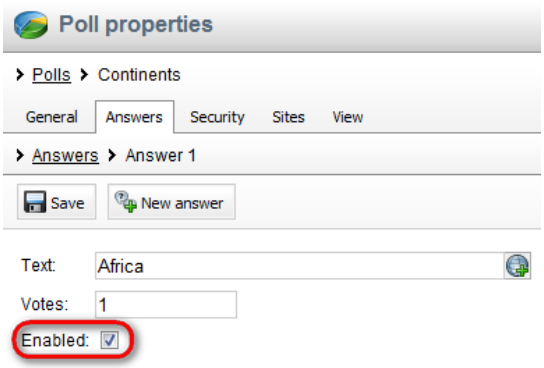
Save the changes you have made to the poll.

Defining answers

Then you can define a list of available answers on the **Answers** tab. To create a new answer, click **New answer** and enter the answer text. Your poll answers can be **Edited** (), **Deleted** (), **Moved Up** () and **Moved down** ().



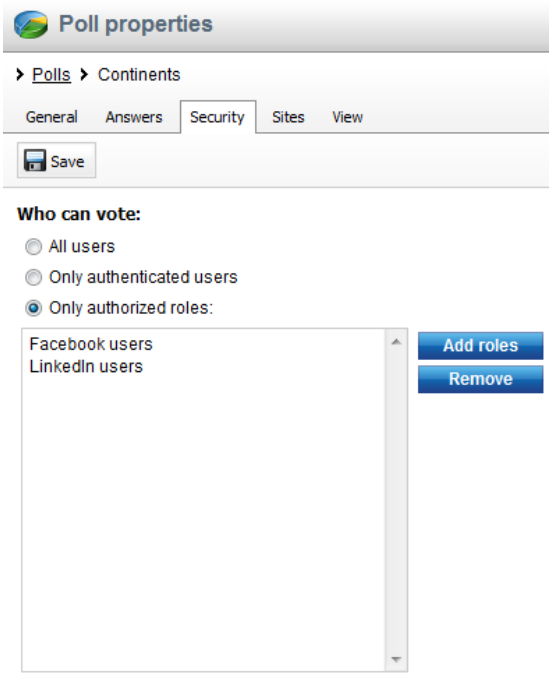
You can choose if the given answer should be enabled, i.e. displayed in the list of options. This is useful if you need to remove an answer from the poll while keeping the number of votes in the history. The disabled option is then not calculated into the displayed results.



You can also reset answers in your poll by clicking **Reset answers**.

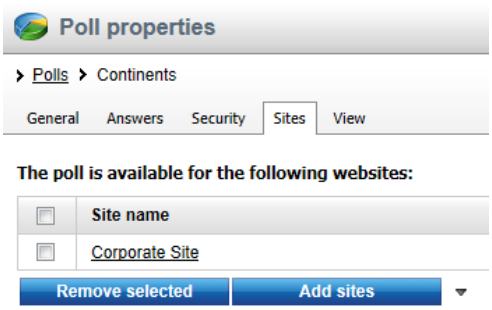
Defining who can vote

On the **Security** tab, you can choose which users can vote in your poll. For more details on who is authorized to vote in your poll, please refer to the Security topic:



Sharing a global poll between sites

On the **Sites** tab, you can choose on which sites a particular global poll will be available. It will be offered to content editors and developers of the selected websites so that they can put it into the text. By default, the site where you created the poll is added.

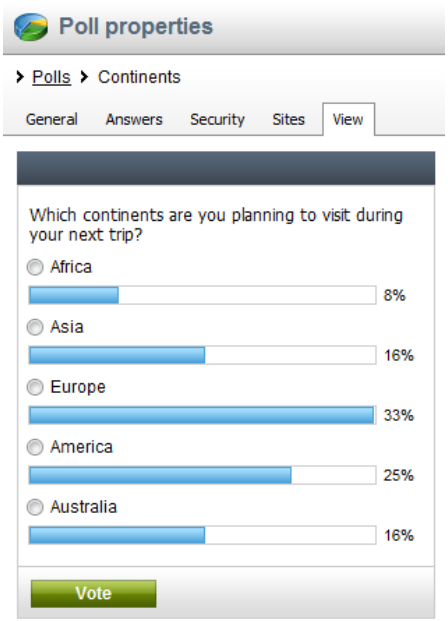


Please note

The **Sites** tab is visible only in global poll management.

Previewing the poll

You can preview the poll on the **View** tab. The actual poll on your website may use a different design depending on the design of this website. It may also behave differently depending on the web part settings (in case you publish it using a web part).



8.39.3 Publishing polls

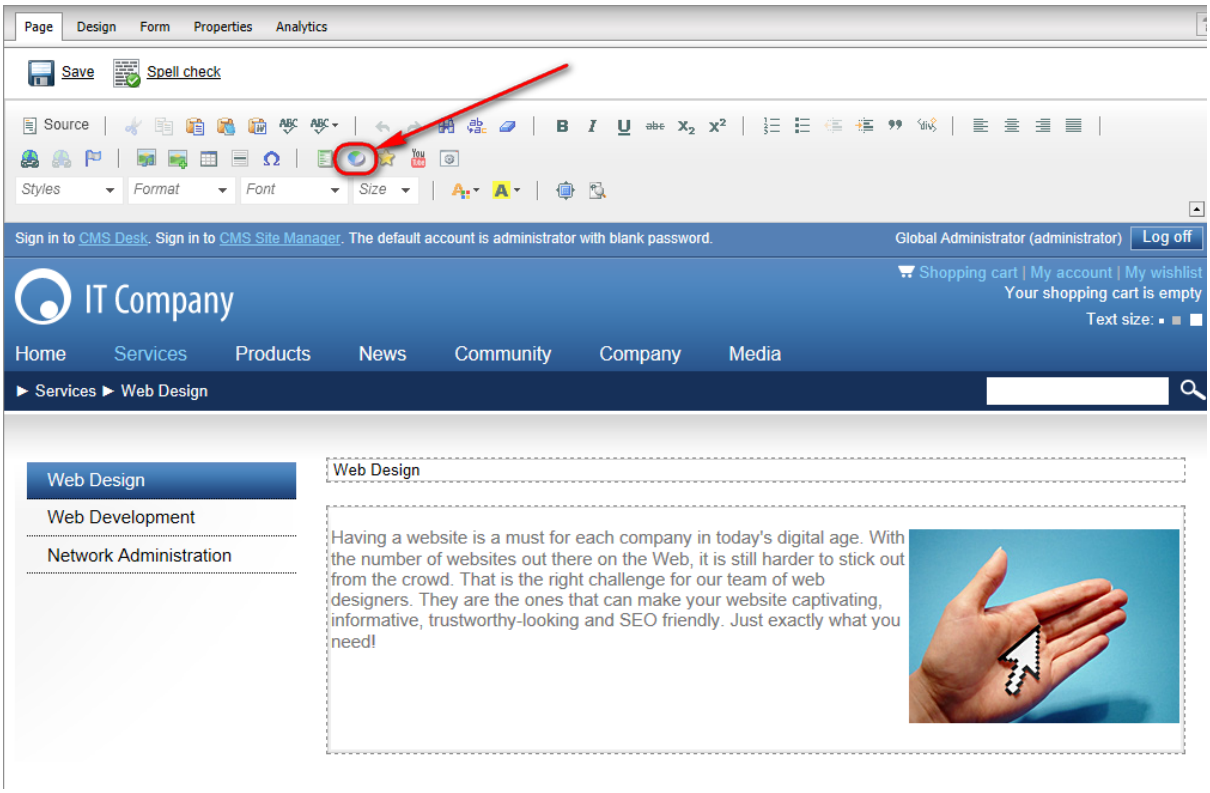
There are two ways how you can publish a poll on your website:

- Content editors can place a poll into editable regions of a page using the **Insert Poll** button on the WYSIWYG editor toolbar.
- Developers can place a poll on the page using the **Polls -> Poll** web part.

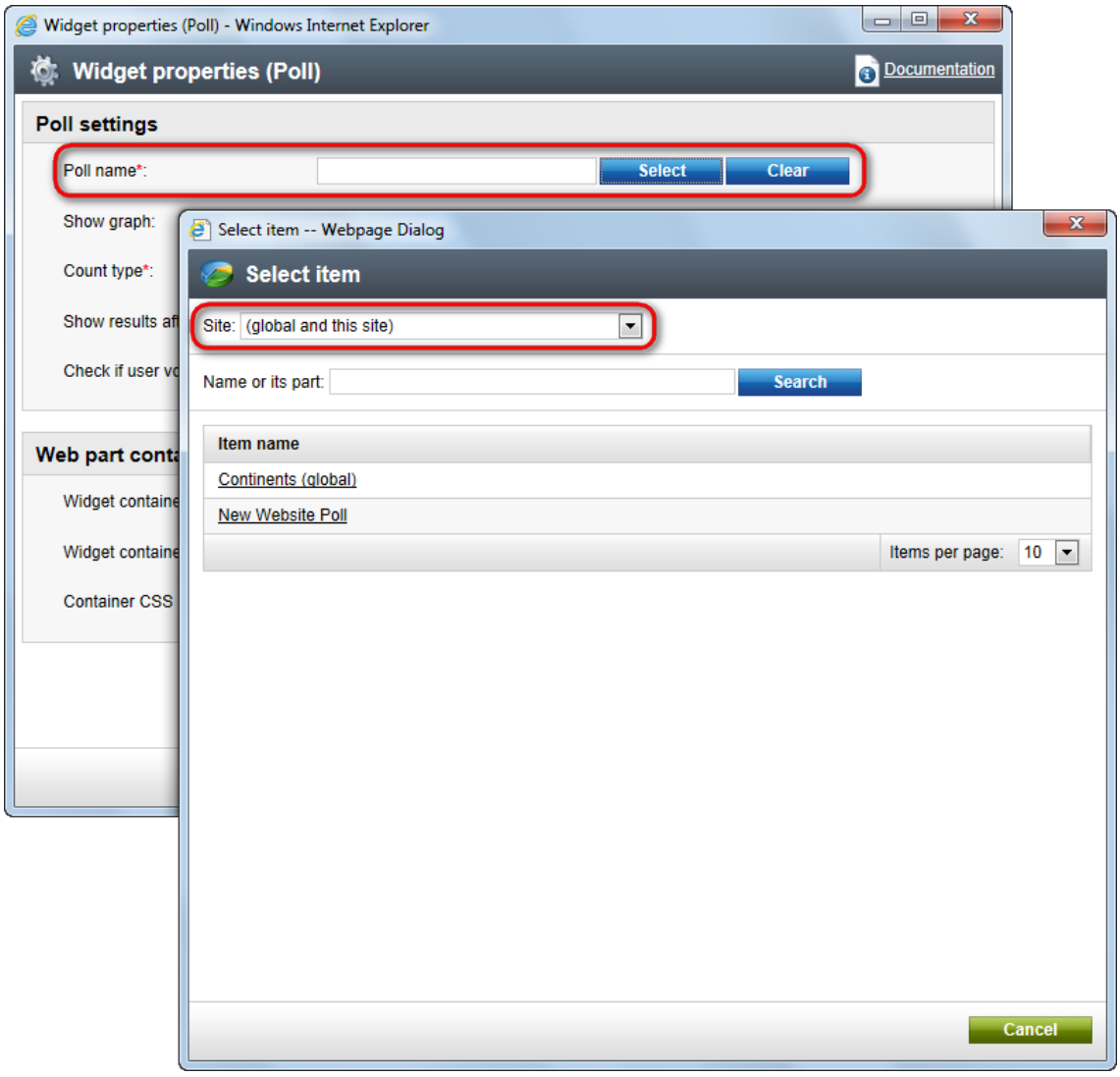
- Users with appropriate permissions can insert a poll into a widget zone of a page using the **Forms & Surveys -> Poll** widget.

Publishing polls for content editors

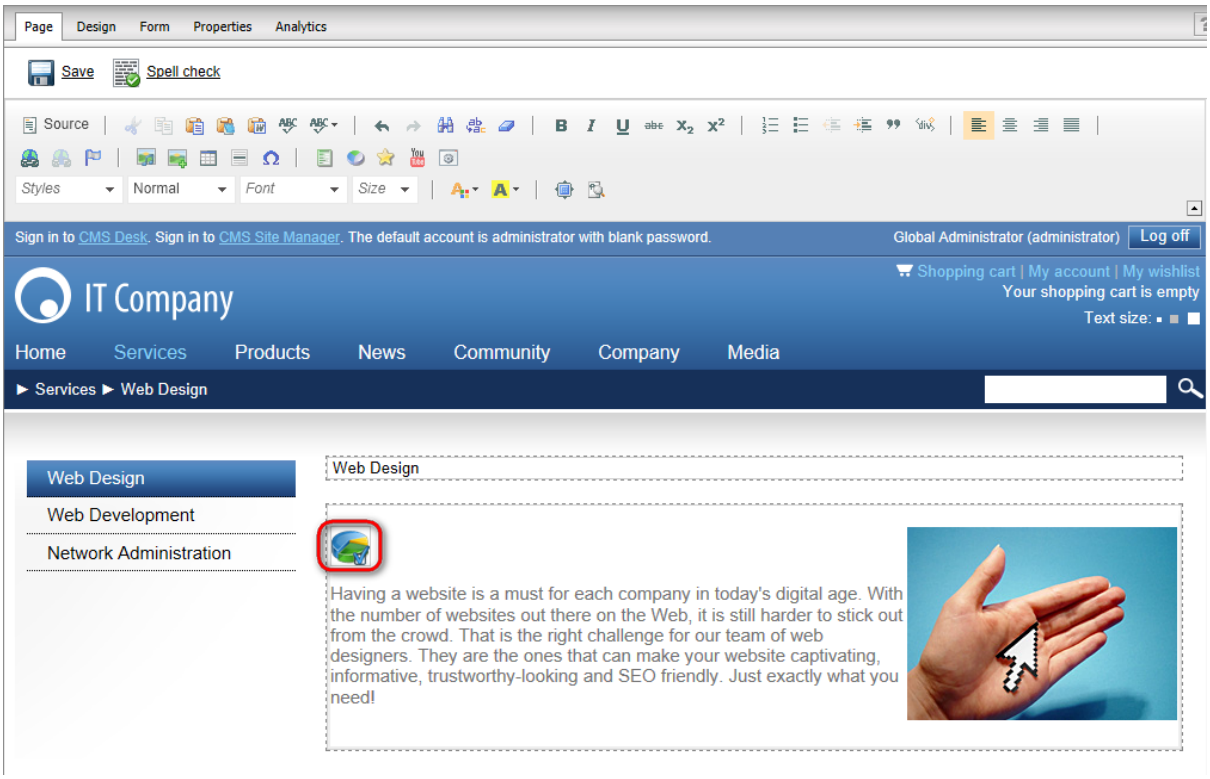
After defining your poll, you need to go to **CMS Desk -> Content** and navigate to the page where you want to have the poll displayed. Switch to the **Edit -> Page** tab and edit the page content using the built-in WYSIWYG editor. You will insert the poll by clicking the **Insert Poll** ( ) button on the WYSIWYG editor toolbar.



The poll will be inserted as an Inline widget. First, the Poll widget's configuration dialog will be displayed, where you can set its properties. Most importantly, the poll that should be displayed must be selected via the **Poll name** property. To do this, click the **Select** button and open the **Select item** dialog window. Here you may first need to set the **Site** property if available, and then select one of the available polls.



Once configured, the poll will be placed on the page, represented by a placeholder image in the editable text region. The properties of the poll widget can be edited at any time by double-clicking the placeholder image.

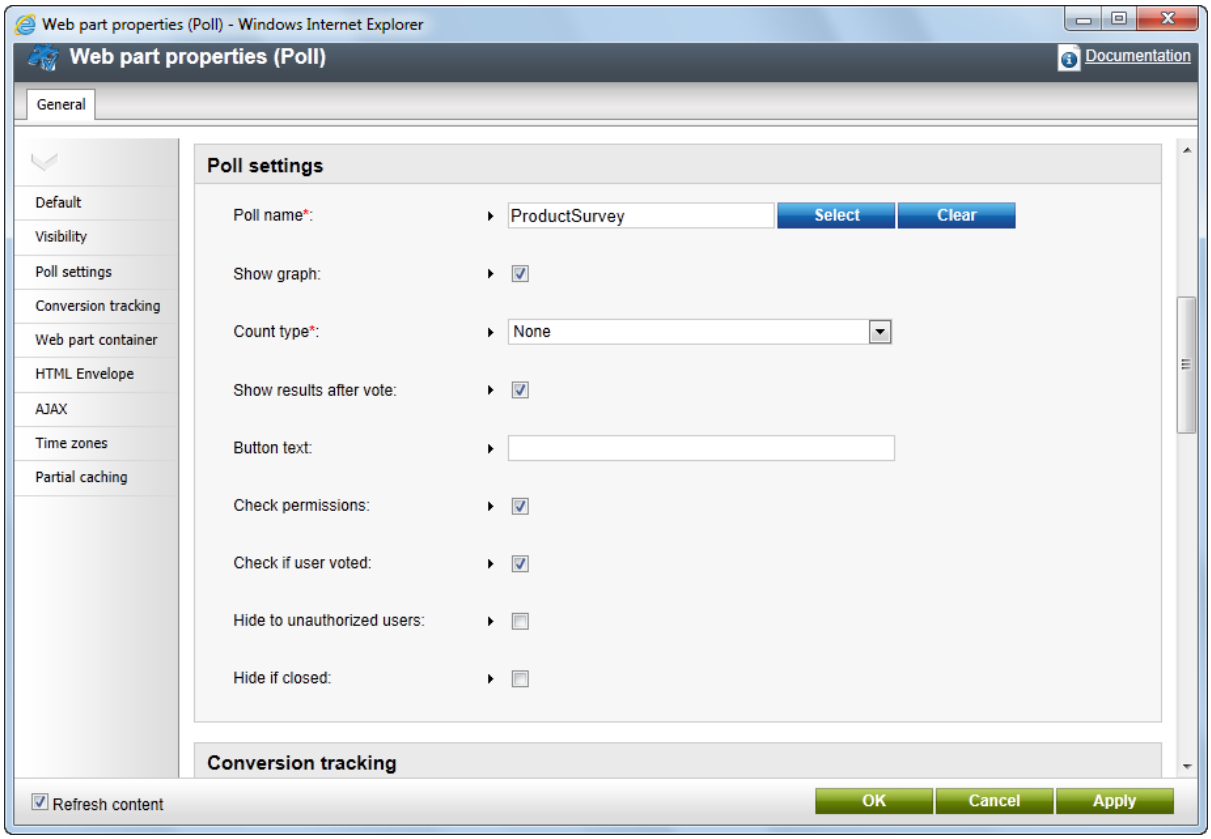


The placeholder image will be replaced by the actual poll on the live site.

Publishing polls for developers

If you are a developer, you can go to **CMS Desk -> Content -> Edit** and from the content tree choose the page where you want to put the poll. Switch to the **Design** tab and add the **Polls -> Poll** web part into a zone on the page.

You need to enter the code name of the poll. Then you can configure some additional settings of the poll that are described in more detail in the Kentico CMS Web Parts reference.



If you are using **ASPX page templates**, you need to drag-and-drop the *~/CMSWebParts/Polls/Poll.ascx* user control (web part) onto your ASPX page.

Please note

If both global and site poll have the same code name, the value of the **Poll name** property will be returned in the ***.<pollname>*** format when selecting the global poll from the **Select item** dialog.
When entering the value manually, to put the global poll on the page you need to enter the value in the same format.

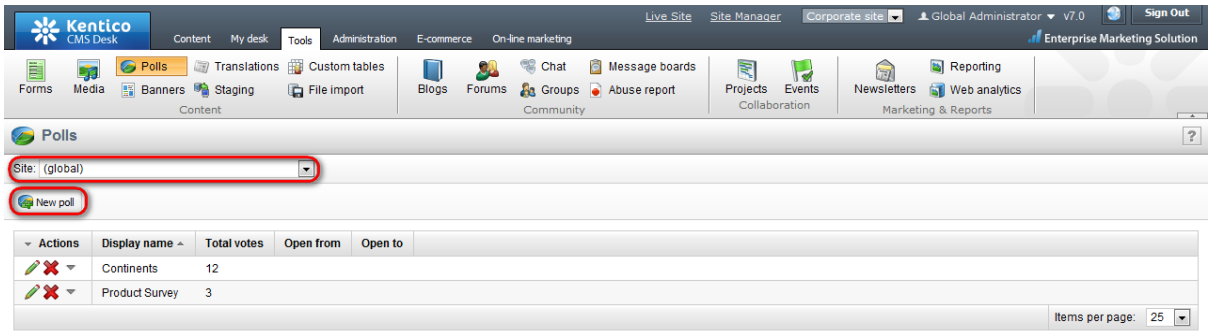
Tip: Global polls not showing up in the list

If global polls are not showing up in the list of polls, you may need to enable them in **Site Manager -> Settings -> Content -> Polls**; more details can be found in the Security topic.
You may also need to enable the particular poll for the given website. This can be done in **CMS Desk -> Tools -> Polls**; more details can be found in the Managing polls topic.

8.39.4 Adding a poll to your site

Here you will learn how to create a new poll and publish it on a page of your website. Please note that in this example, we will be creating a global poll. However, you would proceed analogically if you needed to create a site poll.

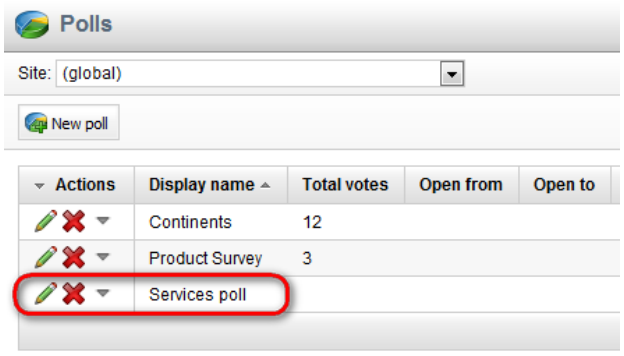
1. First go to **CMS Desk -> Tools -> Polls**, set the **Site** property to *(global)* and click **New poll**.



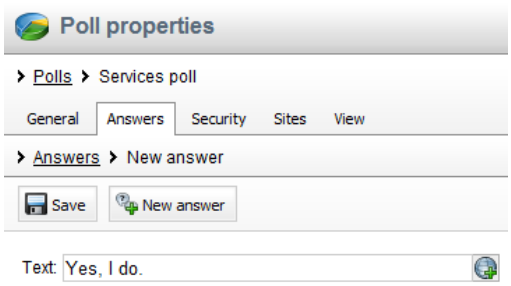
2. Enter the following information for the new poll.

- **Display name:** *Services poll*
- **Code name:** *ServicesPoll*
- **Title:** *Services poll*
- **Question:** *Do you like our services?*

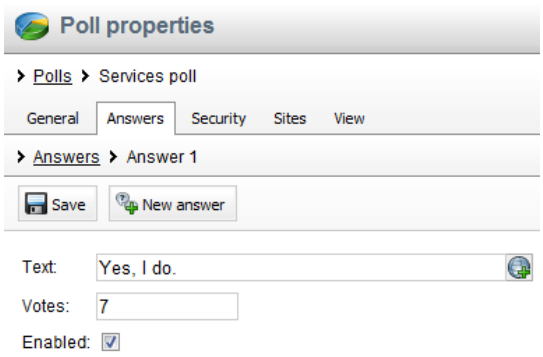
Click **OK** to save the changes. If you switch back to the **CMS Desk -> Tools -> Polls** tab, the poll you have just created will be displayed in the list.



3. Now you need to define some answers for your new poll. Choose to **Edit** () the poll, in the **Poll properties** dialog switch to the **Answers** tab and click **New answer**. Enter *Yes, I do.* into the **Text** text box and click **Save**.

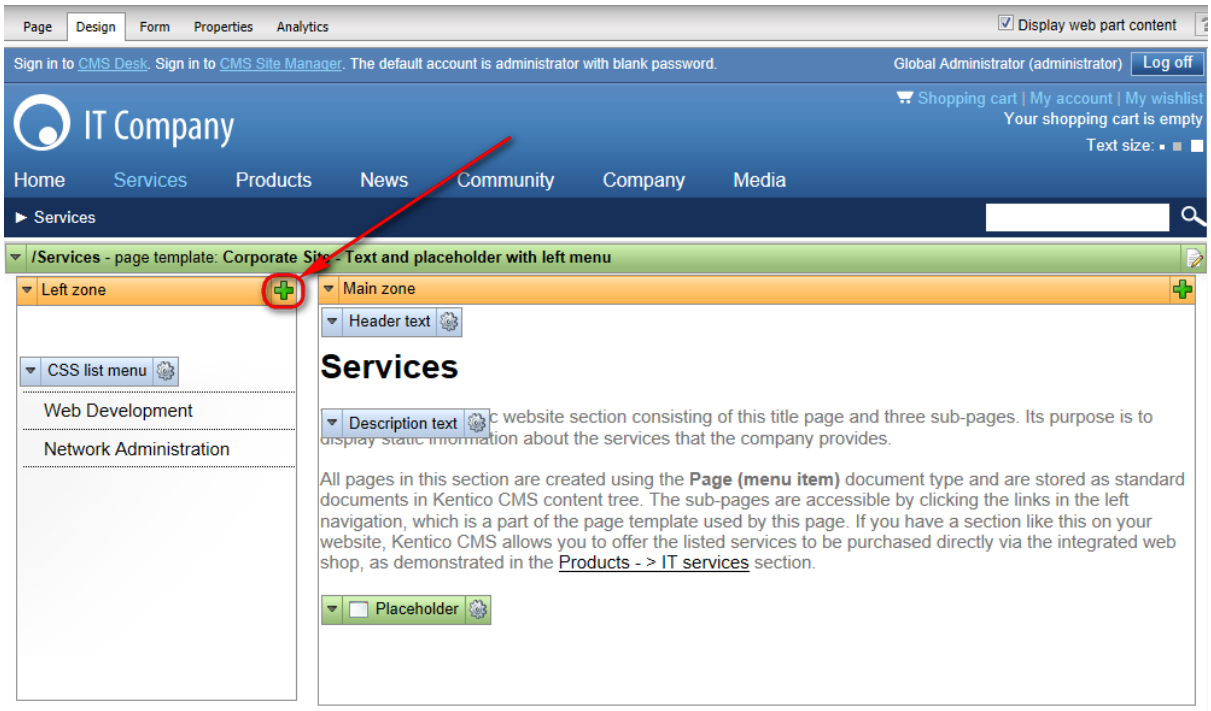


To make the answer available on the live site, check the **Enabled** check box. Besides, you can also enter the initial value for votes. For the purpose of this example, enter 7 and **Save** the changes.

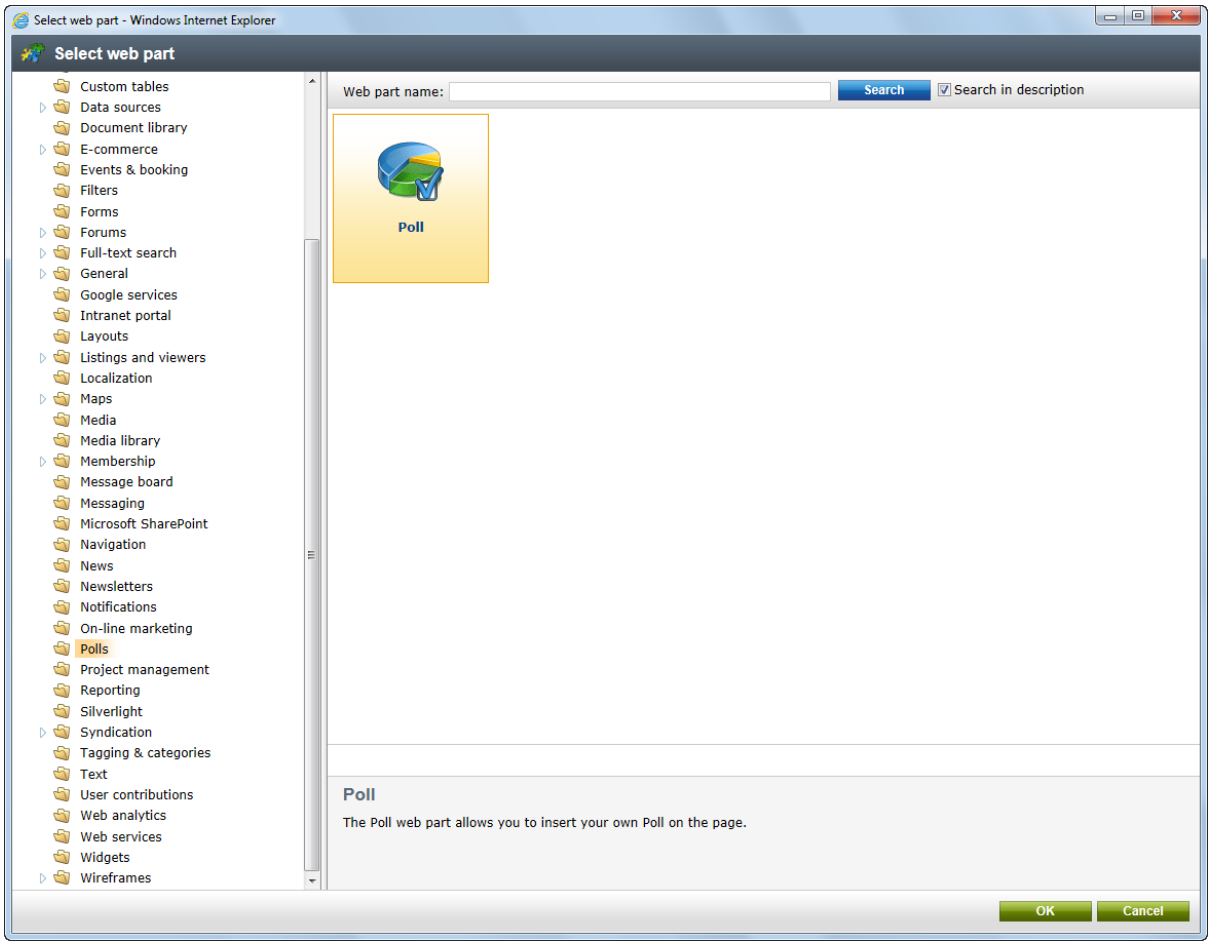


Repeat this step entering *No, I don't.* as an alternative answer and 3 as its initial value for votes. Optionally you can define more answers for your poll.

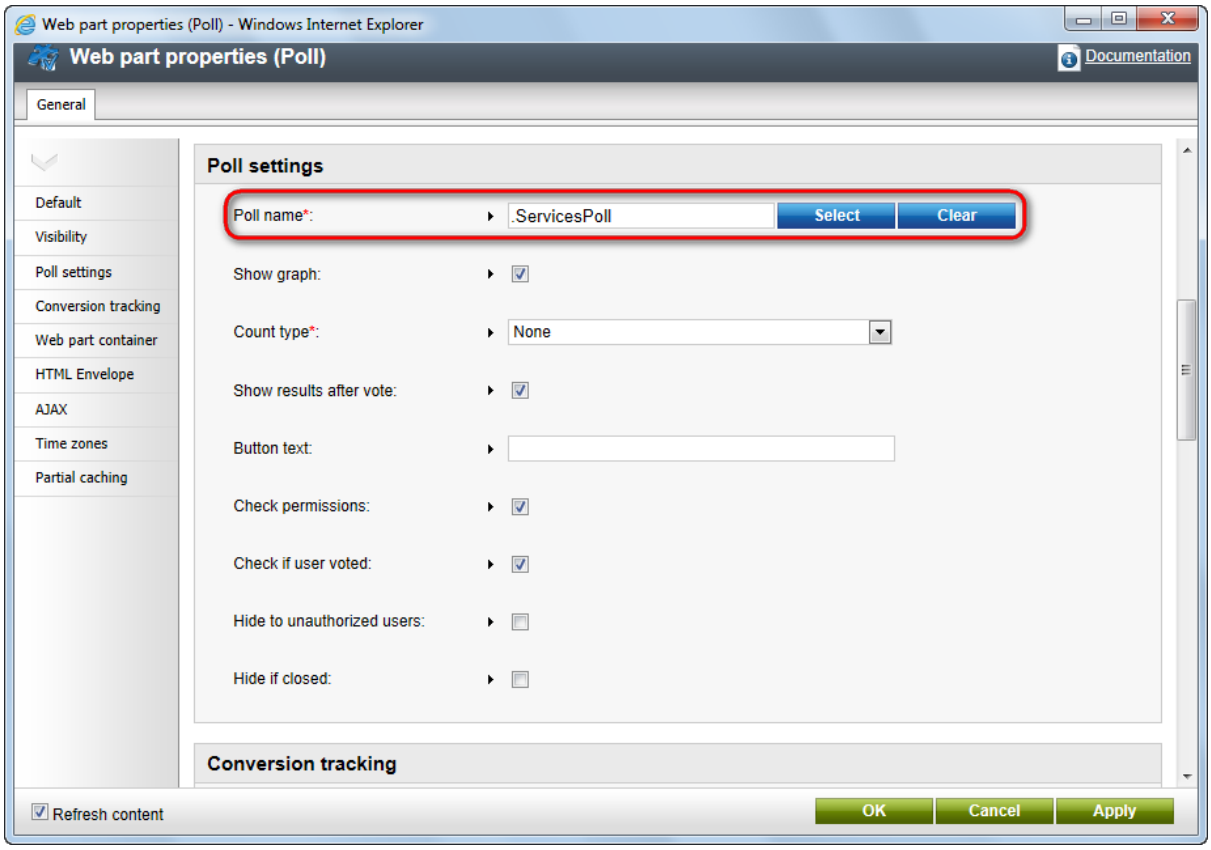
4. Now you are ready to publish your new poll on the website. Go to **CMS Desk -> Content**, navigate to the page where you want to add your poll and switch to the **Design** tab. Click **Add web part** (+) in one of the available web part zones



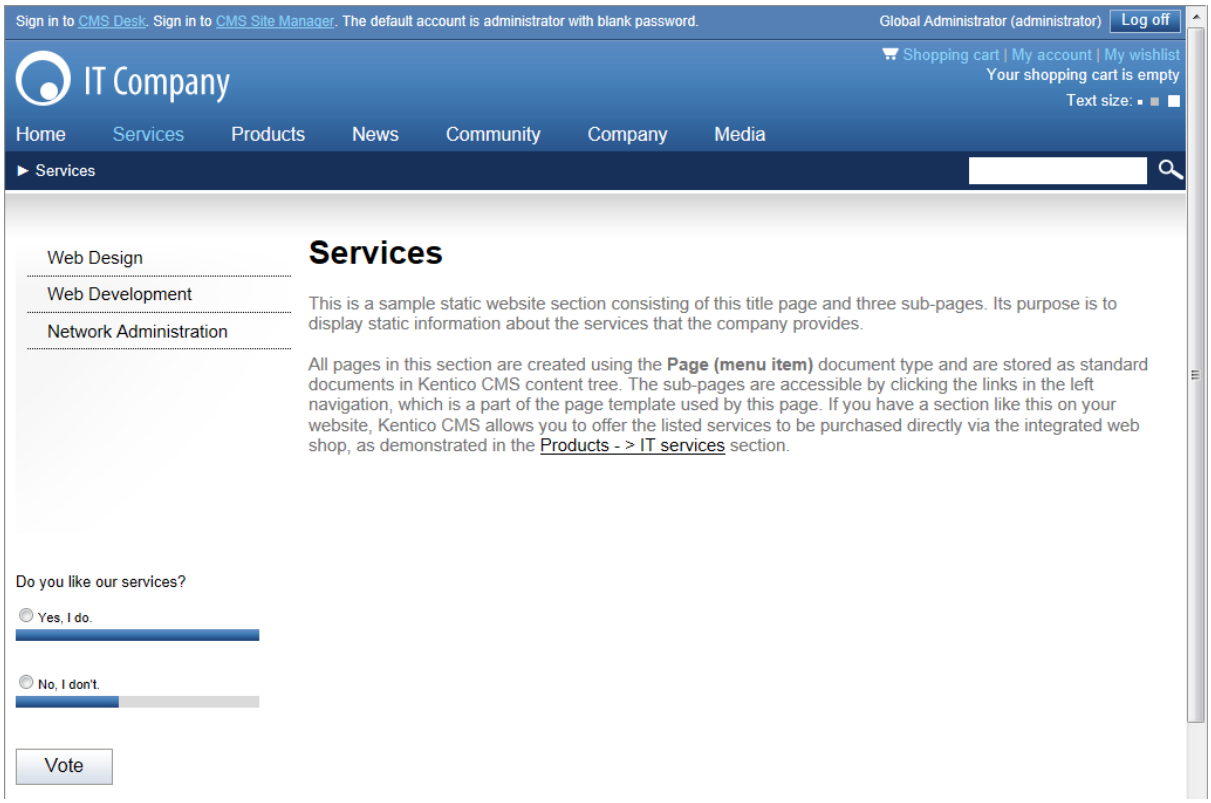
and in the **Select web part** dialog choose the **Polls -> Poll** web part.



5. In the **Web part properties (Poll)** dialog, make sure *.ServicesPoll* is selected as **Poll name** and click **OK**. Please note that in this dialog you can configure also some additional settings of the poll; this is described in more detail in the Kentico CMS Web Parts reference.



If you now switch to the live site, you will see that the page you have just finished editing contains your poll.



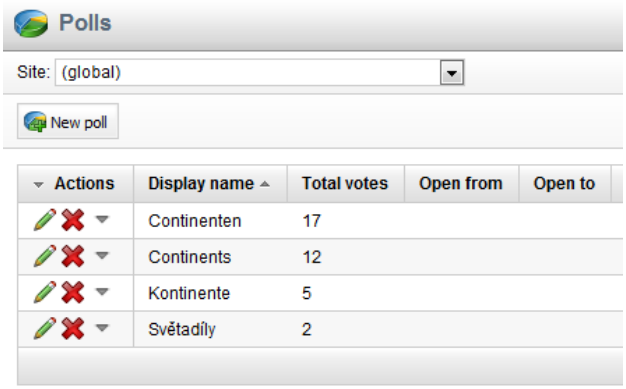
Please note

You can also add a poll to your web page using the built-in WYSIWYG editor as described in more detail in the Publishing polls topic. However, this option gives you fewer possibilities of further adjustment of your poll.

8.39.5 Multilingual support

When you use the Polls module on a multilingual website, you can choose from two options:

- You can create **a new poll for every language** - this is useful if you wish to track votes for different cultures/countries of your website.



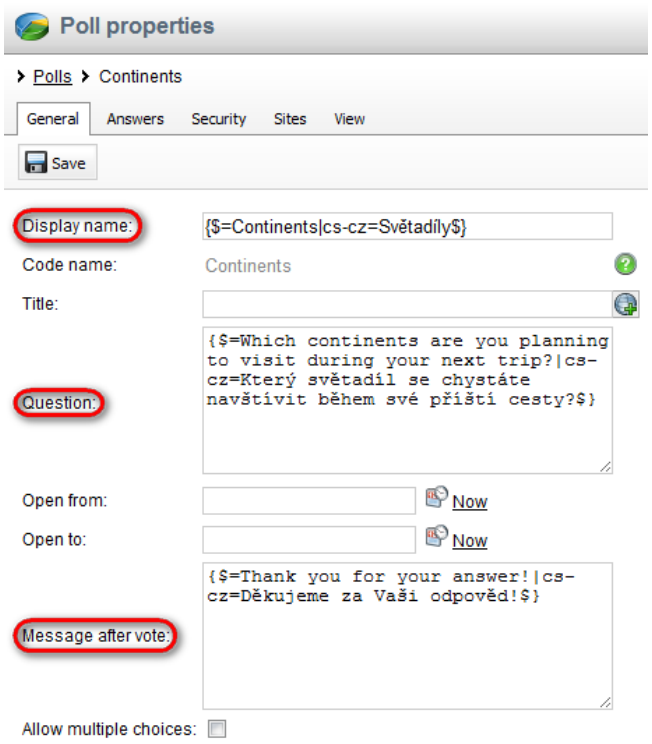
or

- You can create **a single poll for all languages** using in-place localization expressions - in this case, all votes are tracked together. In order to learn how to localize strings of the poll, please refer to the Localization Expressions topic. More detailed information on these expressions can be found in Development -> Macro expressions.

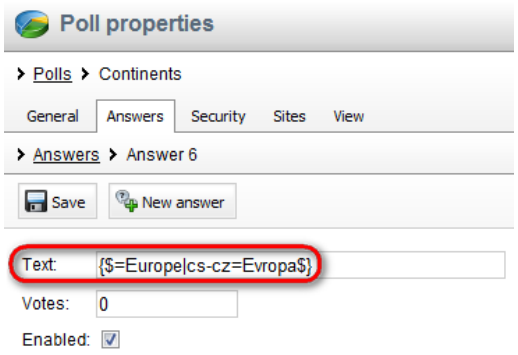
If you want to create a single localized poll (or adjust an existing one to create a single localized poll), you will need to take steps similar to those described in the Managing polls topic. The main difference consists in entering localization expressions into the corresponding text fields. For example, if you would like to have your poll both in the website default language and in Czech, the macro must be entered in the *{$=Default string|cs-cz=Czech string$}* format.

On the **General** tab of the **Poll properties** dialog, use localization expressions for the following fields:

- **Display name**
- **Question**
- **Message after vote**



Save the changes. Now switch to the **Answers** tab to localize the answers of your poll. You will use the localization expression only in the **Text** text field.



When you have entered the new value, click **Save** again.

> **Please note**
>
> To culturally localize your polls, you can also use basic localization macros. These are entered in the *{$string.key$}* format. The system uses the *ResHelper.GetString ("string.key")* method and replaces the macros with the appropriate resource strings. To learn more about this type of localization, please refer to the Configuring multilingual and RTL UI topic.

8.39.6 Security

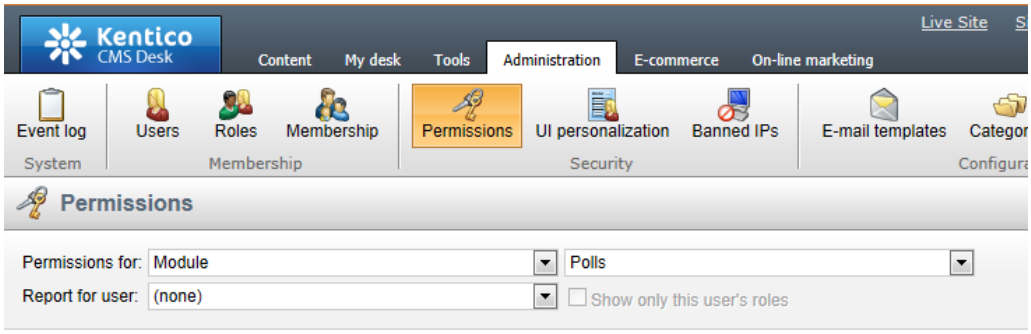
Here you will learn how to configure your CMS to enable users to access the Polls module, to allow global polls and you will also learn how to configure which users are authorized to vote in the poll.

Access to the Polls module

Access to the Polls module can be configured in **CMS Desk -> Administration -> Permissions**. To view the Polls module permissions matrix, you now need to select *Module* and *Polls* from the **Permissions for:** drop-down lists.

The following permissions can be assigned to particular roles:

- **Read global** - members of the given role can view global polls and their configuration, but are not allowed to make any changes to them.
- **Modify global** - members of the given role can create, edit and delete global polls.
- **Read** - members of the given role can view site polls and their configuration, but are not allowed to make any changes to them.
- **Modify** - members of the given role can create, edit and delete site polls.



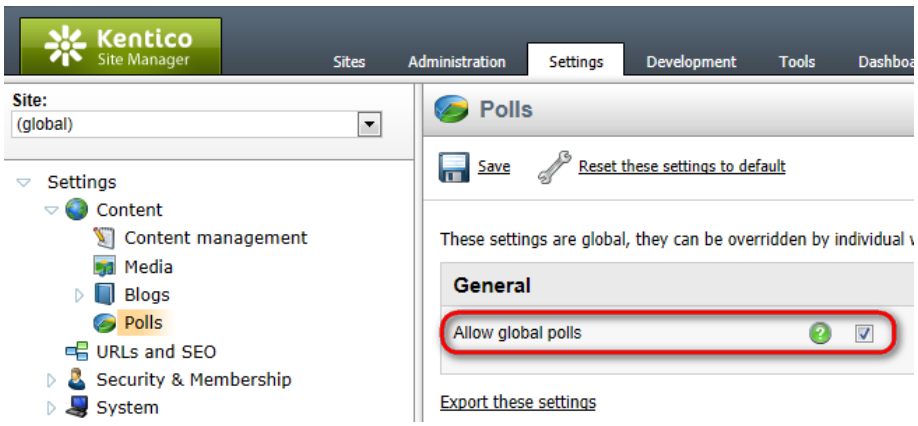
Role	Read global	Modify global	Read	Modify
Authenticated users	☐	☐	☐	☐
CMS Basic users	☐	☐	☐	☐
CMS Community administrators	☑	☑	☑	☑
CMS Designers	☐	☐	☐	☐
CMS Desk Administrators	☑	☑	☑	☑
CMS Editors	☑	☐	☑	☐
CMS Readers	☑	☐	☑	☐
Everyone	☐	☐	☐	☐
Facebook users	☐	☐	☐	☐
Gold Partners	☐	☐	☐	☐
LinkedIn users	☐	☐	☐	☐
Live ID users	☐	☐	☐	☐
Marketing Manager	☑	☐	☑	☐
Not authenticated users	☐	☐	☐	☐

Please note

You can configure access to the Polls module also in **Site Manager -> Administration -> Permissions**. However, as this is a global interface, here you will need to select also the site for which your Polls module permissions assignment applies.

Access to global polls

By default, both global and site polls are allowed in Kentico CMS. However, if you need to restrict access to global polls on a particular website, you can disallow these polls in **Site Manager -> Settings -> Content -> Polls**.

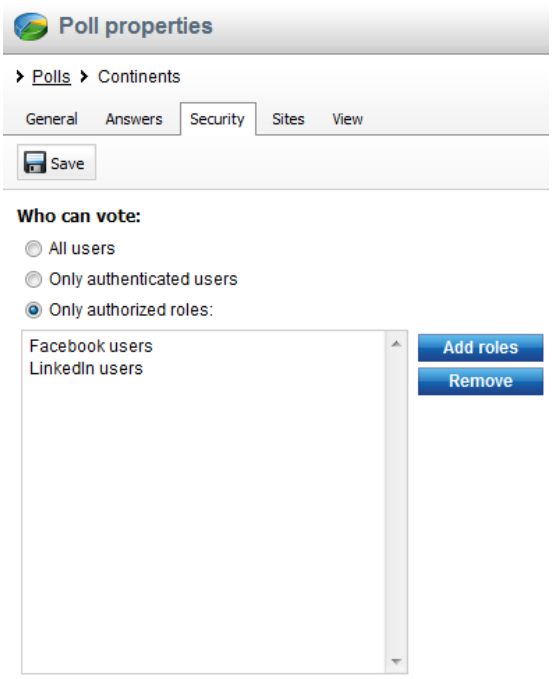


For detailed information on how to configure these settings, please refer to the Website settings topic.

Access to voting

You can configure the poll and specify which users are authorized to vote on the **Security** tab of the **Poll properties** dialog. Here you can choose one of the following options:

- **All users** - any visitor can vote.
- **Only authenticated users** - only site members who sign in can vote.
- **Only authorized roles** - only authenticated members of chosen roles can vote.



Please note

To ensure that the same visitor (using the same browser) cannot vote twice in the same poll, the Polls module uses cookies.

Other security settings

Developers can customize the behavior of the **Polls** web part using the following web part properties:

- **Check permissions** - indicates if permissions for voting specified for the given poll should be checked.
- **Hide to unauthorized users** - hides the web part if the user is not authorized to vote.

8.39.7 Polls internals and API

8.39.7.1 Overview

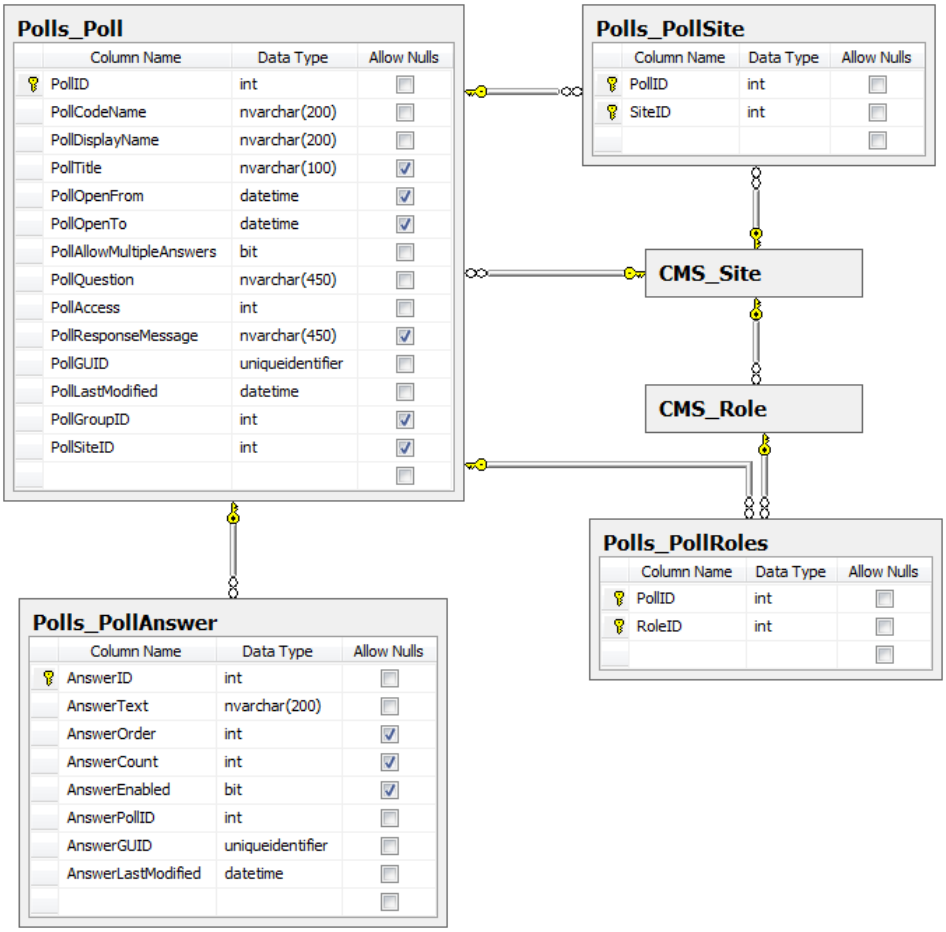
In this chapter, you will learn which database tables and API classes are used in the Polls module. You will also see the most common API examples.

Advanced API examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all methods from the module classes, please refer to Kentico CMS API Reference.

8.39.7.2 Database tables

The following database tables are used in the Polls module:

- **Polls_Poll** - contains records representing polls.
- **Polls_PollSite** - contains records representing sites where the poll can be used (M:N).
- **Polls_PollAnswer** - contains records representing poll answers (1:N).
- **Polls_PollRoles** - contains records representing roles authorized to vote (M:N).



8.39.7.3 API classes

If you are not familiar with the database table data management by Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

> **Please note**
>
> The Polls module classes use the **CMS.Polls** namespace.

Polls_Poll table API:

- **PollInfo** - represents one poll.
- **PollInfoProvider** - provides management of polls.

Polls_PollSite table API:

- **PollSiteInfo** - represents one site where the poll can be used.
- **PollSiteInfoProvider** - provides management of sites where the poll can be used.

Polls_PollAnswer table API:

- **PollAnswerInfo** - represents one poll answer.
- **PollAnswerInfoProvider** - provides management of poll answers.

Polls_PollRoles table API:

- **PollRolesInfo** - represents one role authorized to vote.
- **PollRolesInfoProvider** - provides management of roles authorized to vote.

8.39.7.4 API examples

8.39.7.4.1 Overview

These topics show examples of how the Polls module API can be used:

- Managing polls
- Managing answers

Please note

All API examples can be simulated in the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<your web project folder>\CMSAPIExamples\Code\Tools\Polls\Default.aspx.cs***.

The Polls module API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.GlobalHelper;
using CMS.UIControls;
using CMS.CMSHelper;
using CMS.SiteProvider;
using CMS.Polls;
```

8.39.7.4.2 Managing polls

The following example creates a poll.

```
private bool CreatePoll()
{
    // Create new poll object
    PollInfo newPoll = new PollInfo();

    // Set the properties
    newPoll.PollDisplayName = "My new poll";
    newPoll.PollCodeName = "MyNewPoll";
    newPoll.PollTitle = "My title";
    newPoll.PollQuestion = "My question";
    newPoll.PollResponseMessage = "My response message.";
    newPoll.PollAllowMultipleAnswers = false;
    newPoll.PollAccess = 0;

    // Save the poll
    PollInfoProvider.SetPollInfo(newPoll);

    return true;
}
```

The following example gets and updates a poll.

```
private bool GetAndUpdatePoll()
{
    // Get the poll
    PollInfo updatePoll = PollInfoProvider.GetPollInfo("MyNewPoll");

    if (updatePoll != null)
    {
        // Update the properties
        updatePoll.PollDisplayName = updatePoll.PollDisplayName.ToLower();

        // Save the changes
        PollInfoProvider.SetPollInfo(updatePoll);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates polls.

```
private bool GetAndBulkUpdatePolls()
{
```

```
    // Prepare the parameters
    string where = "PollCodeName LIKE N'MyNewPoll%'";

    // Get the data
    DataSet polls = PollInfoProvider.GetPolls(where, null);

    if (!DataHelper.DataSourceIsEmpty(polls))
    {
        // Loop through the individual items
        foreach (DataRow pollDr in polls.Tables[0].Rows)
        {
            // Create object from DataRow
            PollInfo modifyPoll = new PollInfo(pollDr);

            // Update the properties
            modifyPoll.PollDisplayName = modifyPoll.PollDisplayName.ToUpper();

            // Save the changes
            PollInfoProvider.SetPollInfo(modifyPoll);
        }

        return true;
    }

    return false;
}
```

The following example adds a poll to site.

```
private bool AddPollToSite()
{
    // Get the poll
    PollInfo poll = PollInfoProvider.GetPollInfo("MyNewPoll");

    if (poll != null)
    {
        int pollId = poll.PollID;
        int siteId = CMSContext.CurrentSiteID;

        // Save the binding
        PollSiteInfoProvider.AddPollToSite(pollId, siteId);

        return true;
    }

    return false;
}
```

The following example removes a poll from site.

```
private bool RemovePollFromSite()
```

```
{
    // Get the poll
    PollInfo removePoll = PollInfoProvider.GetPollInfo("MyNewPoll");

    if (removePoll != null)
    {
        // Remove poll from site
        PollSiteInfoProvider.RemovePollFromSite(removePoll.PollID,
CMSContext.CurrentSiteID);

        return true;
    }

    return false;
}
```

The following example deletes a poll.

```
private bool DeletePoll()
{
    // Get the poll
    PollInfo deletePoll = PollInfoProvider.GetPollInfo("MyNewPoll");

    // Delete the poll
    PollInfoProvider.DeletePollInfo(deletePoll);

    return (deletePoll != null);
}
```

8.39.7.4.3 Managing answers

The following example creates an answer.

```
private bool CreateAnswer()
{
    // Get the poll
    PollInfo poll = PollInfoProvider.GetPollInfo("MyNewPoll");

    if (poll != null)
    {
        // Create new answer object
        PollAnswerInfo newAnswer = new PollAnswerInfo();

        // Set the properties
        newAnswer.AnswerPollID = poll.PollID;
        newAnswer.AnswerText = "My new answer";
        newAnswer.AnswerEnabled = true;
        newAnswer.AnswerCount = 0;

        // Save the answer
```

```
        PollAnswerInfoProvider.SetPollAnswerInfo(newAnswer);

        return true;
    }

    return false;
}
```

The following example gets and updates an answer.

```
private bool GetAndUpdateAnswer()
{
    // Get the answer
    PollInfo updatePoll = PollInfoProvider.GetPollInfo("MyNewPoll");

    if (updatePoll != null)
    {
        DataSet answers = PollAnswerInfoProvider.GetAnswers(updatePoll.PollID, 1,
null);

        if (!DataHelper.DataSourceIsEmpty(answers))
        {
            PollAnswerInfo updateAnswer = new PollAnswerInfo(answers.Tables[0]
.Rows[0]);

            // Update the properties
            updateAnswer.AnswerText = updateAnswer.AnswerText.ToLower();

            // Save the changes
            PollAnswerInfoProvider.SetPollAnswerInfo(updateAnswer);

            return true;
        }
    }

    return false;
}
```

The following example gets and bulk updates answers.

```
private bool GetAndBulkUpdateAnswers()
{
    PollInfo updatePoll = PollInfoProvider.GetPollInfo("MyNewPoll");

    if (updatePoll != null)
    {
        // Get the data
        DataSet answers = PollAnswerInfoProvider.GetAnswers(updatePoll.PollID);
        if (!DataHelper.DataSourceIsEmpty(answers))
        {
```

```
            // Loop through the individual items
            foreach (DataRow answerDr in answers.Tables[0].Rows)
            {
                // Create object from DataRow
                PollAnswerInfo modifyAnswer = new PollAnswerInfo(answerDr);

                // Update the properties
                modifyAnswer.AnswerText = modifyAnswer.AnswerText.ToUpper();

                // Save the changes
                PollAnswerInfoProvider.SetPollAnswerInfo(modifyAnswer);
            }

            return true;
        }
    }

    return false;
}
```

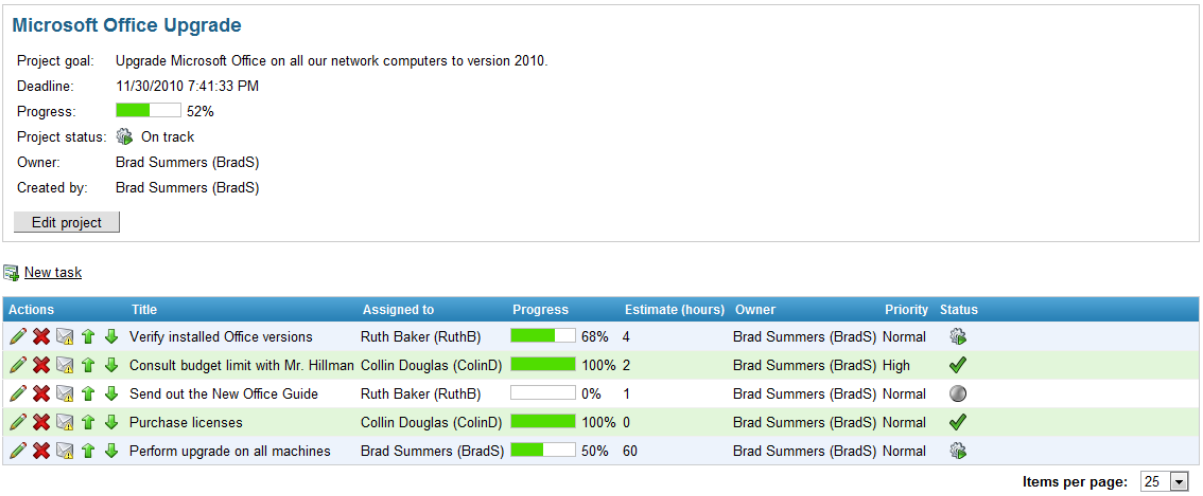
The following example deletes an answer.

```
private bool DeleteAnswer()
{
    // Get the poll
    PollInfo updatePoll = PollInfoProvider.GetPollInfo("MyNewPoll");

    if (updatePoll != null)
    {
        // Get the answer
        DataSet answers = PollAnswerInfoProvider.GetAnswers(updatePoll.PollID, 1,
null);

        if (!DataHelper.DataSourceIsEmpty(answers))
        {
            PollAnswerInfo deleteAnswer = new PollAnswerInfo(answers.Tables[0]
.Rows[0]);

            // Delete the answer
            PollAnswerInfoProvider.DeletePollAnswerInfo(deleteAnswer);

            return (deleteAnswer != null);
        }
    }

    return false;
}
```

8.40 Project management

8.40.1 Overview

Project management is a module that provides a system that helps organize work and keep track of its progress. This is achieved by creating objects within the Kentico CMS system called **projects**, that represent a certain segment of work with a specific goal, time frame, set of conditions or various other properties. Individual assignments called **tasks**, which are usually (but not necessarily) categorized under a certain project, can be given to users within the system.



Projects can be created, monitored and managed on any page (document) in the site content tree that contains the appropriate project management web parts. The document where a project is created is important, as the project is bound to that page and can only be viewed or edited on that specific page. This functionality can be used to categorize projects by having several different pages with project management web parts, which can then be individually configured.

All projects and their tasks can be administered at **CMS Desk -> Tools -> Project Management**, regardless of the document where they were created. Here, administrators can also implement custom priority and status options to be used by projects and tasks. More detailed information about administering and using projects and tasks is given in the Managing projects and tasks topic.

Project management is also integrated into the groups module. Group projects function just as described above, but they belong to a certain group in addition to being bound to the specific group page where they were created. Group projects only operate within the scope of their group, which means that tasks can only be assigned to group members, project security settings can be configured for group roles instead of site-related roles etc. This can be used to create groups for handling certain types of projects. Projects associated with a group are not displayed in the **CMS Desk -> Tools -> Project Management** interface. They can instead be managed by group or global administrators under the **Projects** tab when editing a group.

Users can manage tasks that they own or those that are assigned to them on pages containing specific project management web parts or widgets. Tasks are displayed without regard to the project they belong under or the page or group where they were created, the only thing that matters is that they are relevant to the current user (and that any permission requirements are fulfilled). If the displaying web part/widget is configured to do so, even tasks from other sites can be viewed. These web parts and widgets also allow users to create new personal tasks that aren't categorized under any group or project.

To ensure that users are aware of task assignments or any changes in tasks related to them, project management uses a system of notification e-mails based on templates, that are automatically sent when required.

An overview of the web parts and widgets that allow users to interact with the project management module on the live site can be found in the Project management web parts and widgets topic.

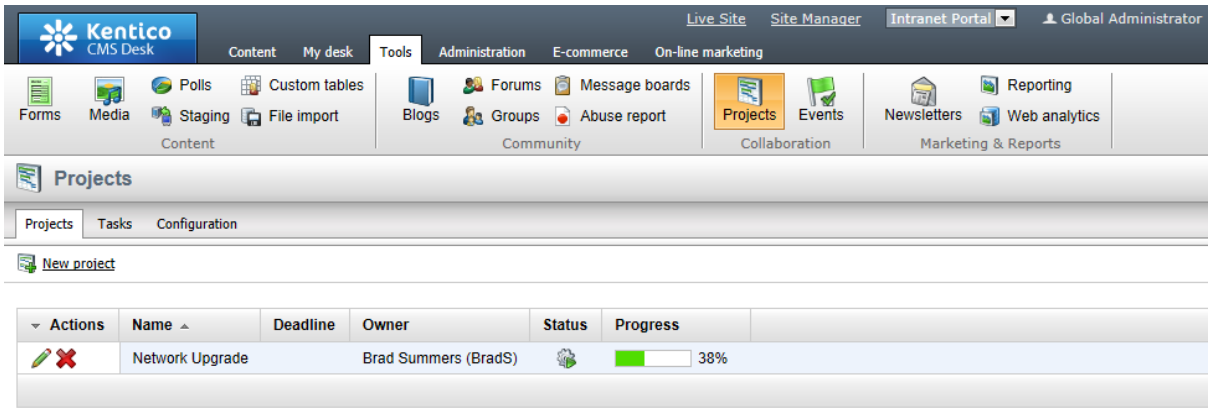
Permissions for project/task management are determined by many factors. Please refer to the Security topic for more information.

8.40.2 Managing projects and tasks

There are several ways how projects and their tasks can be used and managed:

Using the main project management interface

All projects that do not belong to a group can be managed by users with sufficient permissions at **CMS Desk -> Tools -> Project management -> Projects**.

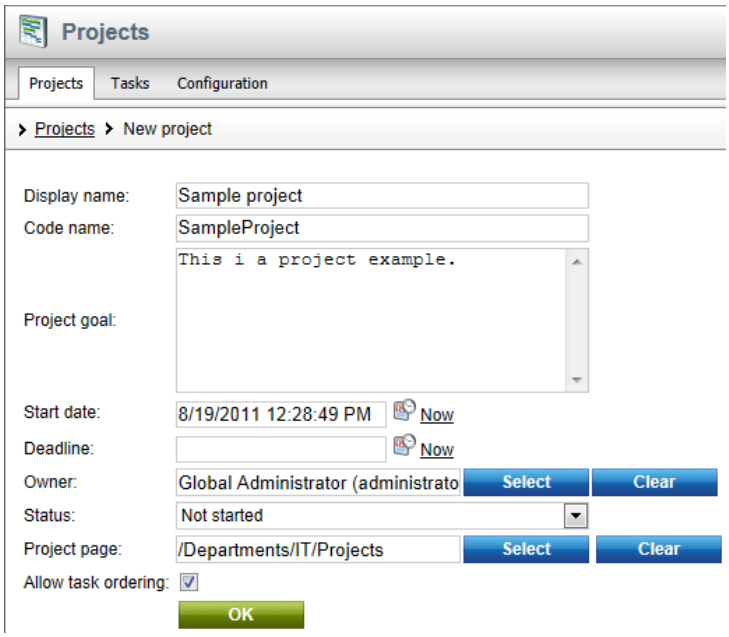


Here, existing projects are displayed in a list where they may be **Edited** ( ) or **Deleted** ( ).

Create a new project using **New project** and fill in the following properties:

- **Display name** - name of the project that is displayed on the live site and in the user interface.
- **Code name** - code name of the project.
- **Project goal** - text description of the purpose of the project.
- **Start date** - specifies the date and time when work on the project should start.
- **Deadline** - specifies the date and time before which the project should be complete.
- **Owner** - allows the selection of the user who will be set as the project owner. The owner is usually the user who ensures that the project is completed successfully. By default, the user who created the project is entered.
- **Status** - allows one of the predefined project statuses to be selected. Statuses are used to indicate which life cycle step the project is currently in and can be managed on the *Configuration* tab of this interface.
- **Project page** - allows the selection of a document from the current site to which the project will be bound.
- **Allow task ordering** - if enabled, the order of tasks under the project can be changed using arrow

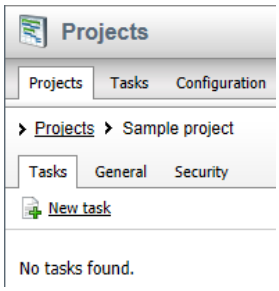
buttons.



Click the **OK** button.

This will open the editing interface of the new project, where three tabs are available. The **General** tab allows the properties that were set when the project was created to be modified and the **Security** tab is where permissions for the project can be configured (learn more in the Security topic).

The **Tasks** tab contains a list of all tasks under the project, which will currently be empty.

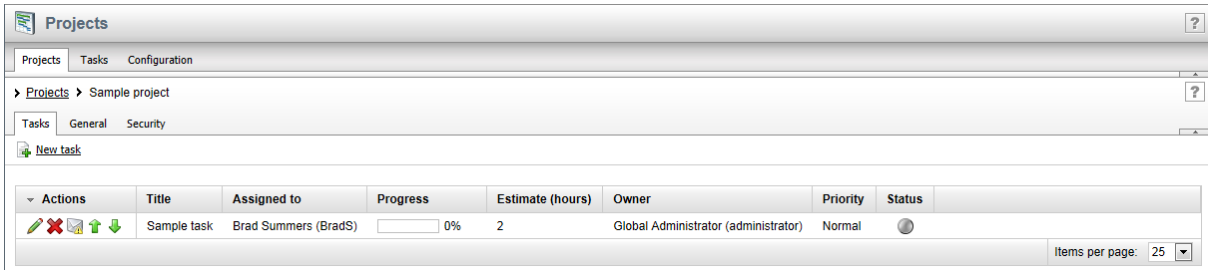


Click **New task** and enter the following properties:

- **Title** - name of the task that is displayed on the live site and in the user interface.
- **Owner** - allows the selection of the user who will be set as the task owner. The owner is usually the user who checks that the task was completed correctly. By default, the user who created the task is entered.
- **Progress** - a percentage representing the amount of progress that has been made on the task.
- **Estimate** - is used to set an approximate amount of hours that the completion of the task should take.
- **Deadline** - specifies the date and time before which the task should be complete.
- **Status** - allows one of the predefined task statuses to be selected. Statuses are used to indicate which life cycle step the task is currently in.

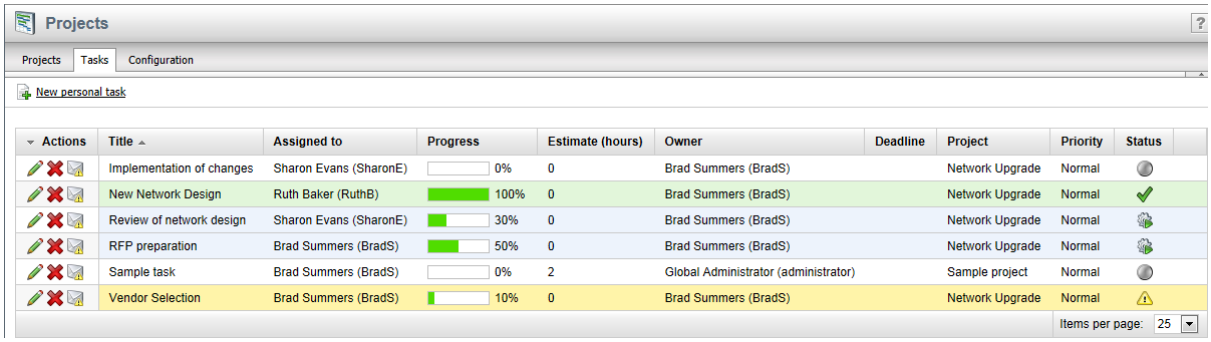
- **Priority** - allows one of the predefined task priorities to be selected.
- **Is private** - if enabled, the task will only be displayed to its owner, the user to whom it is assigned and to users who have permissions to manage the project that the task belongs under (if applicable).
- **Assigned to** - allows the selection of the user who should complete this task. A notification e-mail is automatically sent to the specified user when the task is created or modified.
- **Description** - text containing the goal of the task and any other relevant information.

Click **OK**.

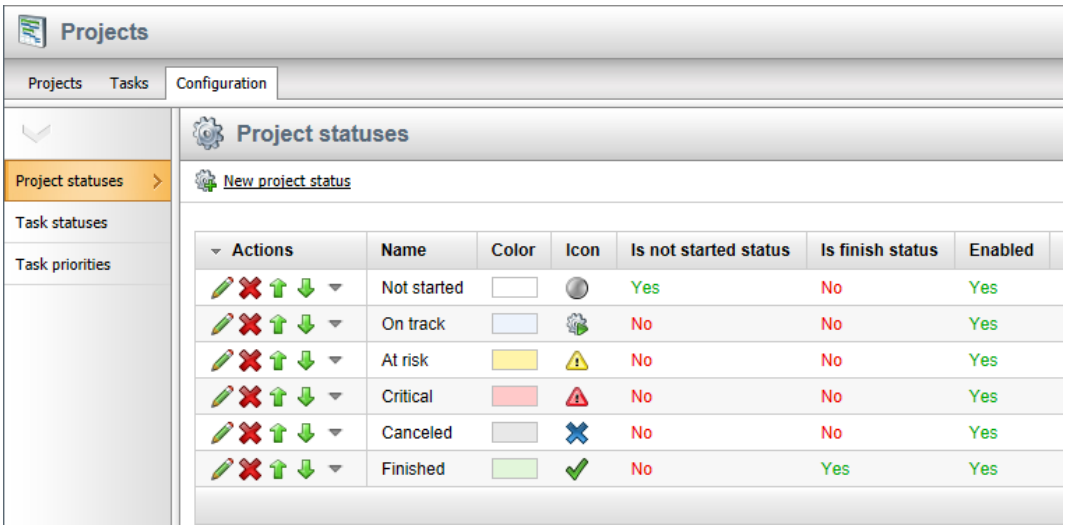


You can manage existing tasks using the appropriate action buttons on the left side: **Edit** () or **Delete** (). The **Send reminder** () button allows an e-mail to be sent to the user to whom the task is assigned, reminding them about the task. If the **Allow task ordering** property of the edited project is enabled, the **Move up** () and **Move down** () buttons can be used to change the order of the tasks.

The **Tasks** tab on the top level of the project management interface is very similar to to the list of tasks shown when editing a project, except that all tasks are displayed regardless of the project they belong under, including personal tasks without any project.



The **Configuration** tab is where the statuses and priorities that can be set for projects or tasks may be defined. It may only be accessed by users belonging to roles with the **Manage configuration** permission for the project management module (more information can be found in the Security topic). All the usual actions such as **Edit** () and **Delete** () are available.

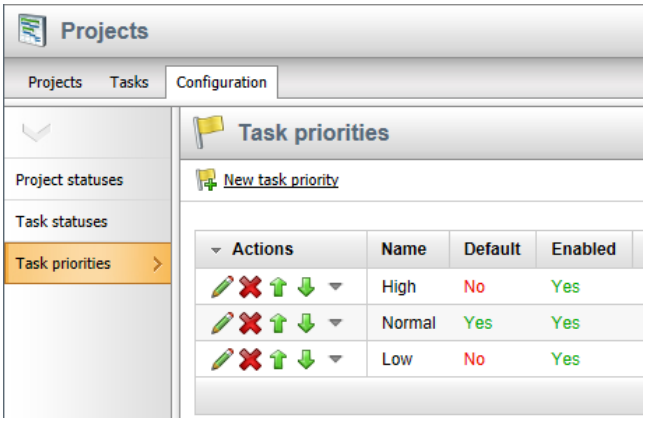


When creating or editing a **status**, the following properties can be set:

- **Display name** - name of the status that is used in the status selection list.
- **Code name** - code name of the status.
- **Status color** - determines the color of projects/tasks that have this status selected. The color selector can be used here.
- **Status icon** - contains the path to an image used as the icon of the status.
- **Is not started status** - if enabled, projects/tasks with this status are marked as not yet started.
- **Is finish status** - if enabled, projects/tasks with this status are marked as finished. A status cannot be both *not started* and *finished*.
- **Enabled** - if enabled, the status can be selected for projects.

The *not started* or *finished* flag given to a project or task by its status can have an effect on whether it is displayed by certain project management web parts or widgets, depending on their configuration.

The **Task priorities** tab is where priorities, which can be selected to determine how urgent a task is, may be managed.



The following properties are available for task priorities:

- **Display name** - name of the priority that is used in the task priority selection list.
- **Code name** - code name of the task priority.

- **Default** - if enabled, the priority is selected by default when a new task is created. Only one priority can be set as default.
- **Enabled** - if enabled, the priority can be selected for tasks.

E-mail notifications

As mentioned above, there are several types of notification e-mails that are sent to users informing them about events related to tasks that they are involved in. These e-mails are sent from the address specified in the value of the **Site Manager -> Settings -> Intranet & Collaboration -> Projects -> Send project management e-mails from** setting and their format can be customized at **Site Manager -> Administration -> E-mail templates**. The e-mail templates listed below are available:

- **Project Management - New task** - sent when a new task is assigned to the user.
- **Project Management - Changed task** - sent when an assigned or owned task is in some way modified.
- **Project Management - Overdue task** - sent when an assigned task reaches its deadline before it is finished.
- **Project Management - Task reminder** - sent when the **Send reminder** ( ) button is used for an assigned task.

You can use the following specific context macros in project management notification templates:

- **{% TaskURL %}** - URL of the page where the given task can be edited.
- **{% ProjectTaskDescriptionPlain %}** - resolves into a plain text version of the task's description (all HTML tags are removed or converted to text equivalents).
- **{% ReminderMessage %}** - text of the message entered by the user who sent the reminder.
- **{% ReminderMessagePlain %}** - plain text version of the reminder message.

You can also access the following related objects and their properties (e.g. *{% Project.ProjectDeadline %}*, *{% Owner.FullName %}* etc.):

- **{% Project %}** - *ProjectInfo* object of the project under which the given task belongs.
- **{% ProjectTask %}** - *ProjectTaskInfo* object representing the task to which the notification is related.
- **{% ProjectTaskStatus %}** - *ProjectTaskStatusInfo* object of the task's current status.
- **{% Owner %}** - *UserInfo* object representing the task's owner.
- **{% Assignee %}** - *UserInfo* object of the user to whom the task is assigned.

Besides these special ones, you can also use all other standard macro expressions in the templates. See the Development -> Macro expressions chapter of this guide for more information about macro expressions in Kentico CMS.

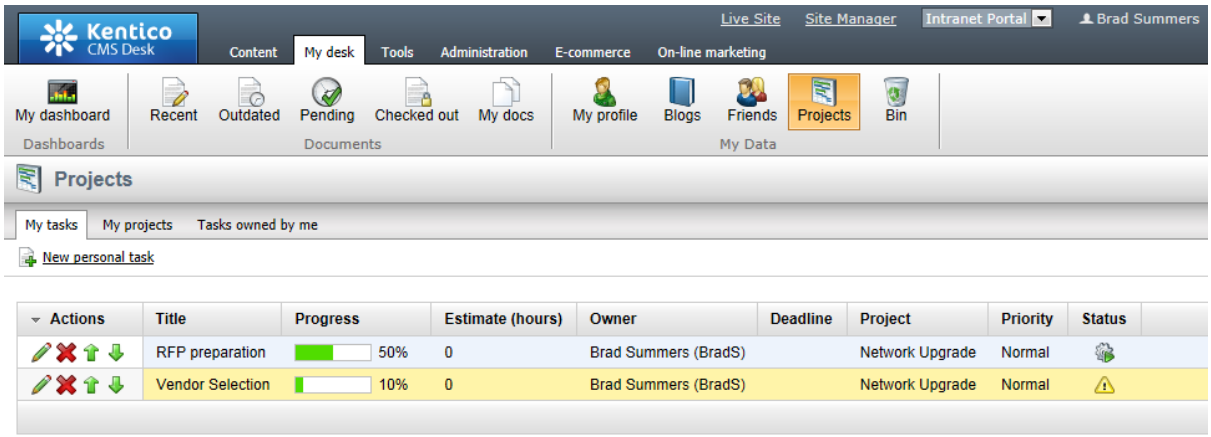
Using the Group management interface

Group projects can be administered by users, who have permissions to manage a given group, by editing the group on its **Projects** tab either at **CMS Desk -> Tools -> Groups** or on-site using the **Group profile** web part. Exactly the same options are available as when using the **Projects** tab of the main project management interface described above, except that the context of the current group is used e.g. for selecting users to whom a task should be assigned.

Using CMS Desk -> My Desk

Users with access to **CMS Desk** can use the **My Desk -> Projects** interface to manage projects and

tasks that they are involved in.



The current user can see a list of all tasks assigned to them on the **My tasks** tab, all the projects that they have permissions to access on the **My projects** tab and all tasks that they own on the **Tasks owned by me** tab. Projects and tasks can be managed using the actions described above.

8.40.3 Project management web parts and widgets

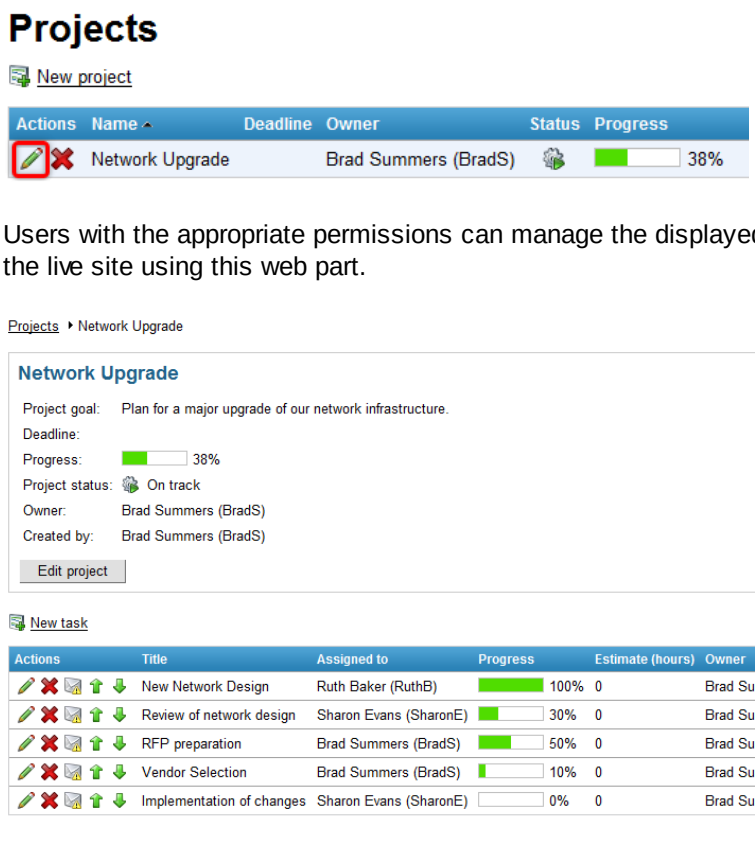
The web parts described in this topic can be placed on pages to create an interface that allows users to work with projects and tasks directly on the live site. They can all be found in the **Project management** web part category. Additional information about using these web parts on the live site is given in the Using project management on the live site topic.

Project list

The Project list web part displays all projects that are bound to the document it is placed on and allows new ones to be created. If the document is owned by a group, only projects that belong under that group are shown.

Specific properties:

- **Show finished projects** - if enabled, projects that have a status designating a finished project are also displayed.
- **Project can be managed by** - can be used to set the *Create*, *Modify* and *Delete* permissions for the displayed projects. The selection made here overrides the security settings of individual projects. The following options can be selected:
  - *Module administrators* - each displayed project uses its individual security settings.
  - *All users* - all site users receive permissions for the displayed projects.
  - *Authenticated users* - only signed-in users receive permissions for the displayed projects.
  - *Group members* - only members of the group that the current document belongs to receive permissions for the displayed projects.
  - *Authorized roles* - only members of the roles selected in the *Authorized roles* property receive permissions for the displayed projects.
- **Authorized roles** - can be used to select the roles that should have permissions for the displayed projects if *Authorized roles* is selected in the *Project can be managed by* property.



Users with the appropriate permissions can manage the displayed projects and their tasks directly on the live site using this web part.

My projects

This web part displays a list of all projects that the current user has access permissions for. Only projects that are bound to an existing document are shown, since the projects in the list also serve as links to that document. Documents that have projects bound to them usually also contain the **Project list** web part and if this is the case, the editing interface of the linked project is automatically opened.

Specific properties:

- **Show finished projects** - if enabled, projects that have a status designating a finished project are also displayed.

The **My projects** widget based on this web part is included in the project management module by default.

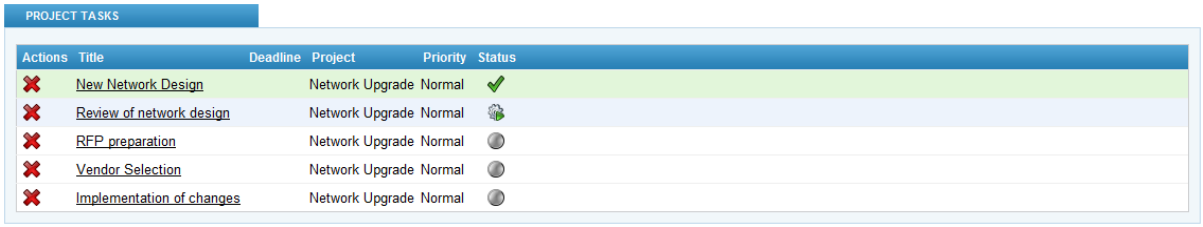
Project tasks

This web part displays a list of tasks from a selected set of projects. Users can only view tasks from

those projects that they have access permissions for. Depending on the security settings of the selected projects, tasks can also be edited directly by clicking on their title.

Specific properties:

- **Projects** - allows the projects whose tasks should be displayed to be selected. All projects from the current site that do not belong under any group are available.
- **Show overdue tasks** - if enabled, tasks that have passed their deadline are displayed.
- **Show on time tasks** - if enabled, tasks that haven't yet reached their deadline are displayed.
- **Show private tasks** - if enabled, private tasks are displayed. They are only visible by those users who are allowed to see them.
- **Show finished tasks** - if enabled, tasks that have a status designating a finished task are displayed.
- **Show status as** - allows the way that task statuses are displayed to be selected. Possible options are *Icon*, *Text* and *Icon & Text*.
- **Allow task actions** - if enabled, users with the appropriate permissions are allowed to delete the displayed tasks.

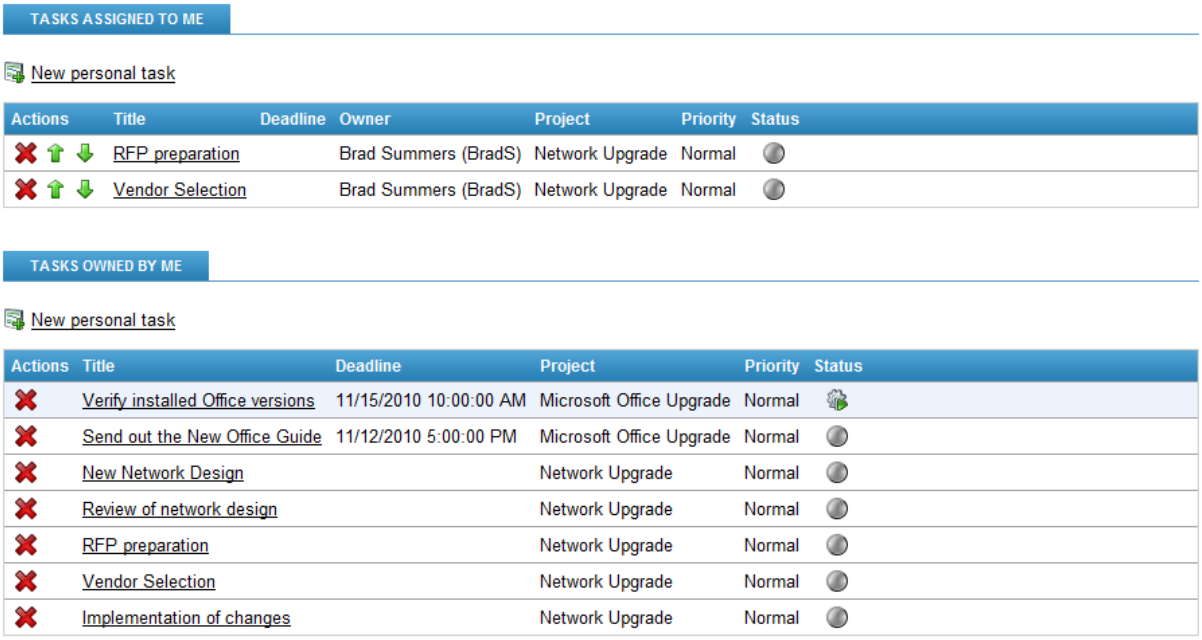


Tasks owned by me / Tasks assigned to me

These two web parts display a list of tasks that are either owned by or assigned to the current user. Personal tasks (those not belonging to a project) are always displayed, but project tasks are only shown if the user has permissions to access the given project. The tasks in the list can be edited directly by clicking their title.

Specific properties:

- **Show overdue tasks**, **Show on time tasks**, **Show private tasks**, **Show finished tasks**, **Show status as** - are the same as for the *Project tasks* web part described above.
- **Allow task actions** - if enabled, the displayed tasks can be deleted and new personal tasks (not categorized under any project) can be created using the web part.
- **Site** - can be used to select if tasks should be displayed from *(all sites)*, or only the *(current site)*.



The **Tasks owned by me** and **Tasks assigned to me** widgets based on these two web parts are included in the project management module by default.

Task info panel

The Task info panel can be used to display a message containing task related information, usually showing the amount of tasks that are assigned to the current user serving as a link to a document where the details of the tasks can be viewed.

Specific properties:

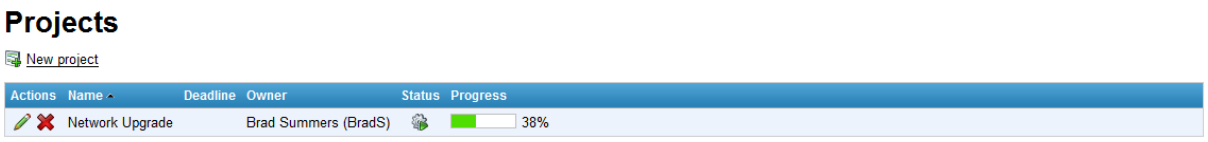
- **Task detail page URL** - contains the URL of the document that is linked when the displayed message is clicked. If left empty, the value is taken from the *Site Manager -> Settings -> Intranet & Collaboration -> Projects -> Task detail page* field. In the case that neither of these locations contain a value, the message will not work as a link.
- **Info text** - contains the text of the displayed message. It may contain the *{0}* formatting macro expression, which will be resolved into the number of tasks assigned to the current user. Sample value: *{0} active task(s)*
- **Include not started tasks** - if enabled, tasks that have a status designating a task that has not been started yet are counted by the web part.
- **Include finished tasks** - if enabled, tasks that have a status designating a finished task are counted by the web part.

8.40.4 Using project management on the live site

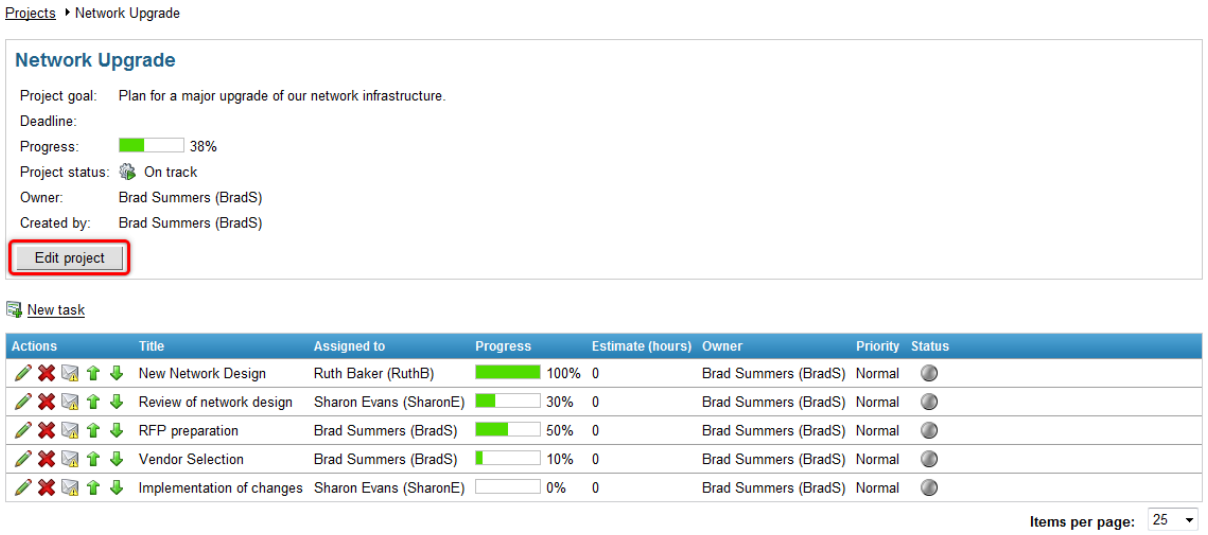
Users can work with projects and tasks on the live site through Project management web parts and widgets. The actions that are available to users depend on project security settings and the configuration of specific web parts or widgets.

The **Project list** web part allows projects, that are bound to the document it is placed on, to be managed in a similar fashion as in the administration interface, which is described in the Managing projects and

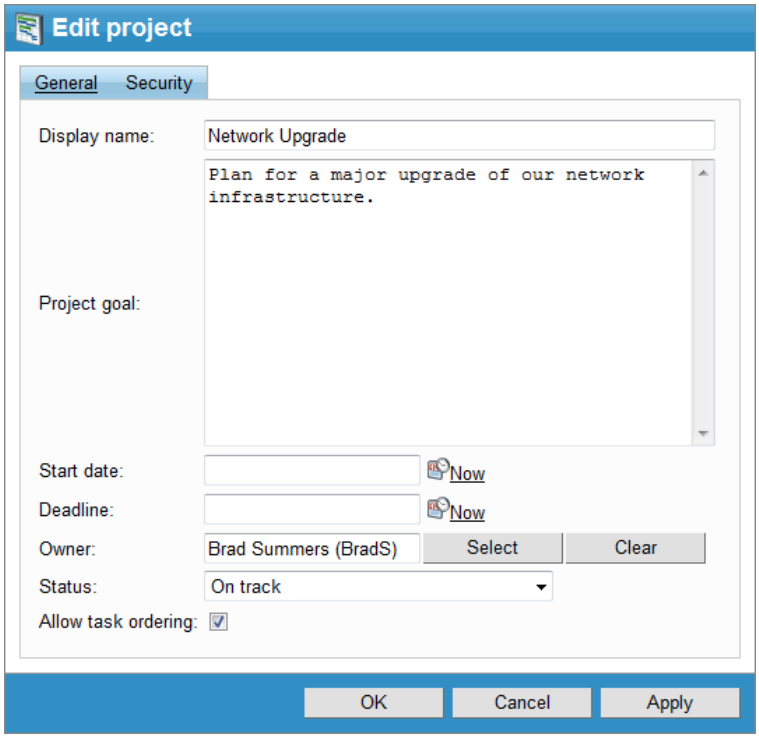
tasks topic.



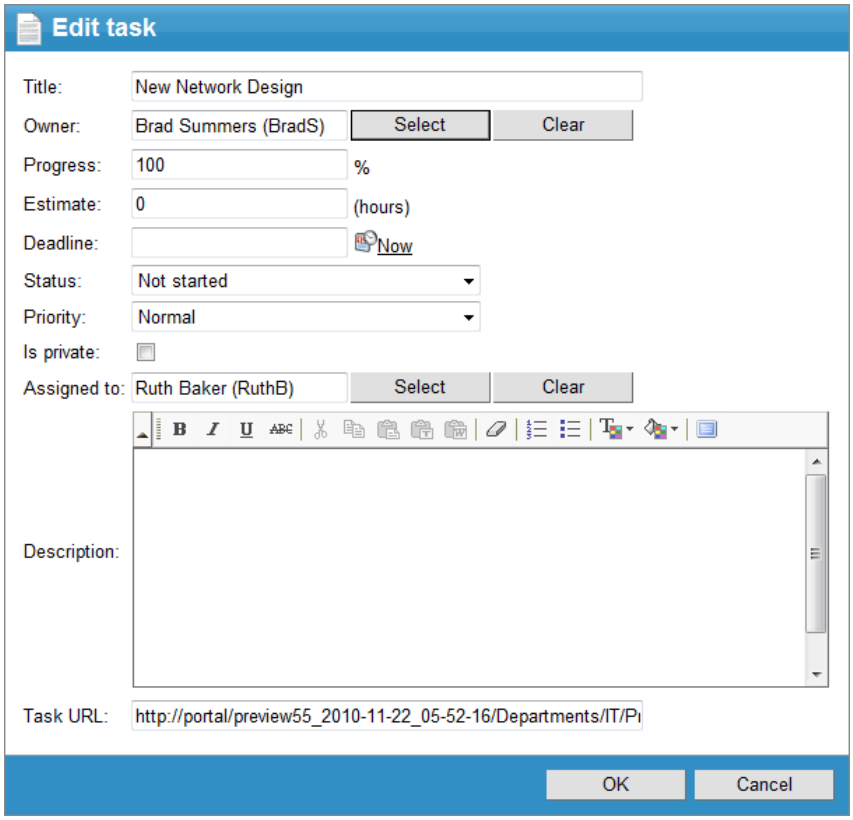
When a project is edited (), its details and the tasks it contains are displayed. Configuration of the project itself can be achieved by clicking the **Edit project** button.



This causes the following dialog to appear:



Editing ( ) a task on the live site is done through the dialog depicted below:

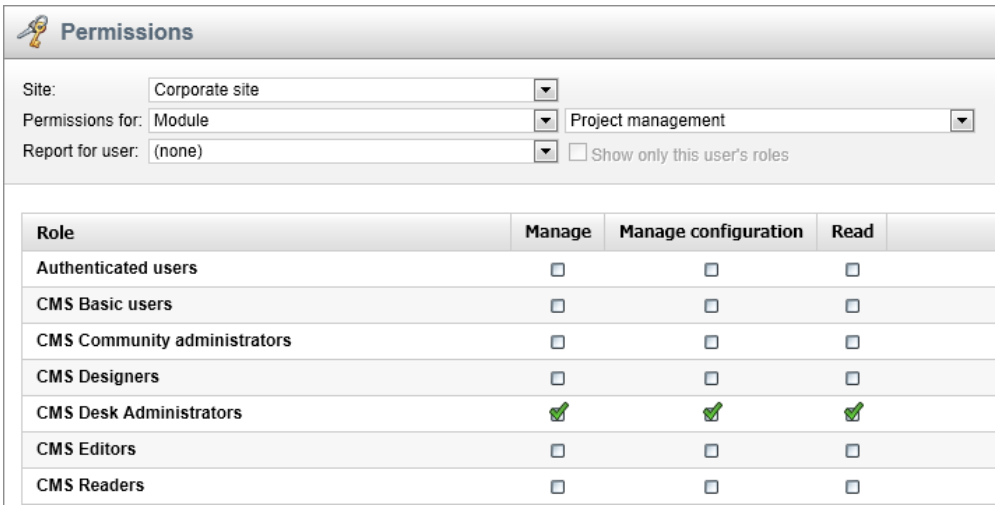


In addition to the task properties described in the Managing projects and tasks topic, the **Task URL** is shown, which can be used to link to the edited task.

Certain other web parts and widgets, such as **Tasks assigned to me**, may be used to edit displayed tasks in the same way.

8.40.5 Security

Permissions for administering the project management module can be set by going to **CMS Site Manager** (or **CMS Desk**) **-> Administration -> Permissions** and selecting *Module* and then *Project Management* through the **Permissions for** drop-down lists.



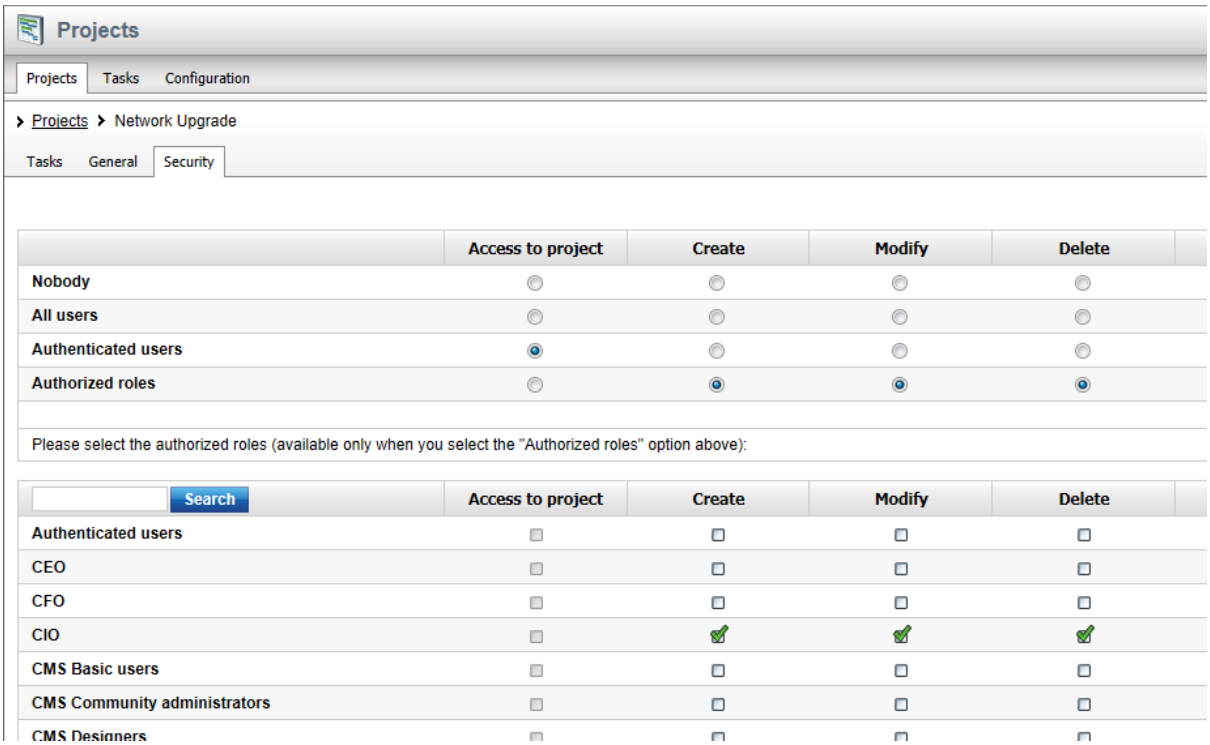
The following permissions can be assigned to members of the selected roles:

- **Manage** - members of the selected roles are allowed to create, modify and delete data in the project management interface, except for the *Configuration* tab.
- **Manage configuration** - members of the selected roles are allowed to view and modify data, such as statuses and priorities, on the *Configuration* tab of the project management interface.
- **Read** - members of the selected roles are allowed to view all data in the project management interface, except for the *Configuration* tab.

To access and use the management interface for group projects, users must either have the **Read** and **Manage** permissions from the *Groups* **Permission matrix**, or belong to a group role that can manage the given group.

Project permissions

Permissions for individual projects can be configured on the **Security** tab when editing a project. This affects which users can view and manage the project and its tasks using project management web parts.



The following permissions are available:

- **Access to project** - users with this permission can view and access the project.
- **Create** - users with this permission can create new tasks under the project.
- **Modify** - users with this permission can modify the tasks under the project.
- **Delete** - users with this permission can delete the tasks under the project.

These permissions can be assigned to:

- **Nobody** - nobody can perform the action.
- **All users** - all users can perform the action.
- **Authenticated users** - only signed-in users can perform the action, i.e. anonymous public users cannot perform it.
- **Group members** - only members of the group can perform the action; this option is only available for projects that belong under a group.
- **Authorized roles** - only members that are assigned to the roles selected below can perform the action; for group projects, only the roles defined for the given group can be selected.

Project security exceptions

1. The user who is set as the owner of a project automatically has full permissions for that project.

2. A task can be modified by users that it is assigned to or owned by regardless of the security settings of the project that the task belongs under.

3. Users who belong to roles that have the **Manage** permission for the project management module have full permissions for all projects that do not belong to a group.

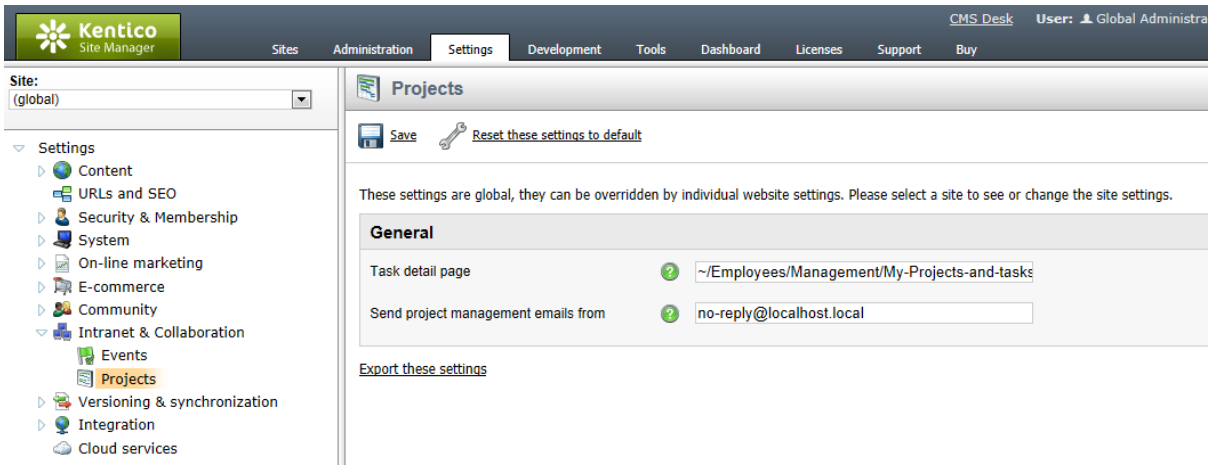
4. Users who belong to roles that can manage a group have full permissions for all projects under the given group.

5. The **Project list** web part has its own security properties that can be used to override the permissions set for the projects that it displays.

8.40.6 Settings

Additional settings of the project management module are located in **Site Manager -> Settings -> Intranet & Collaboration -> Projects**. The following can be configured:

- **Task detail page** - URL of a page that displays detailed information about the tasks assigned to users. This page should contain the *Tasks assigned to me* web part. The value of this setting is used for generating links to personal tasks in project management e-mail notifications and as the URL that is linked to by the *Task info panel* web part if its *Task detail page URL* property is empty. Sample value: *~/Employees/Management/My-Projects-and-tasks.aspx*
- **Send project management e-mails from** - sets the e-mail address from which automatic project management notification e-mails are sent when tasks are created or modified.



8.40.7 Project management internals and API

8.40.7.1 Overview

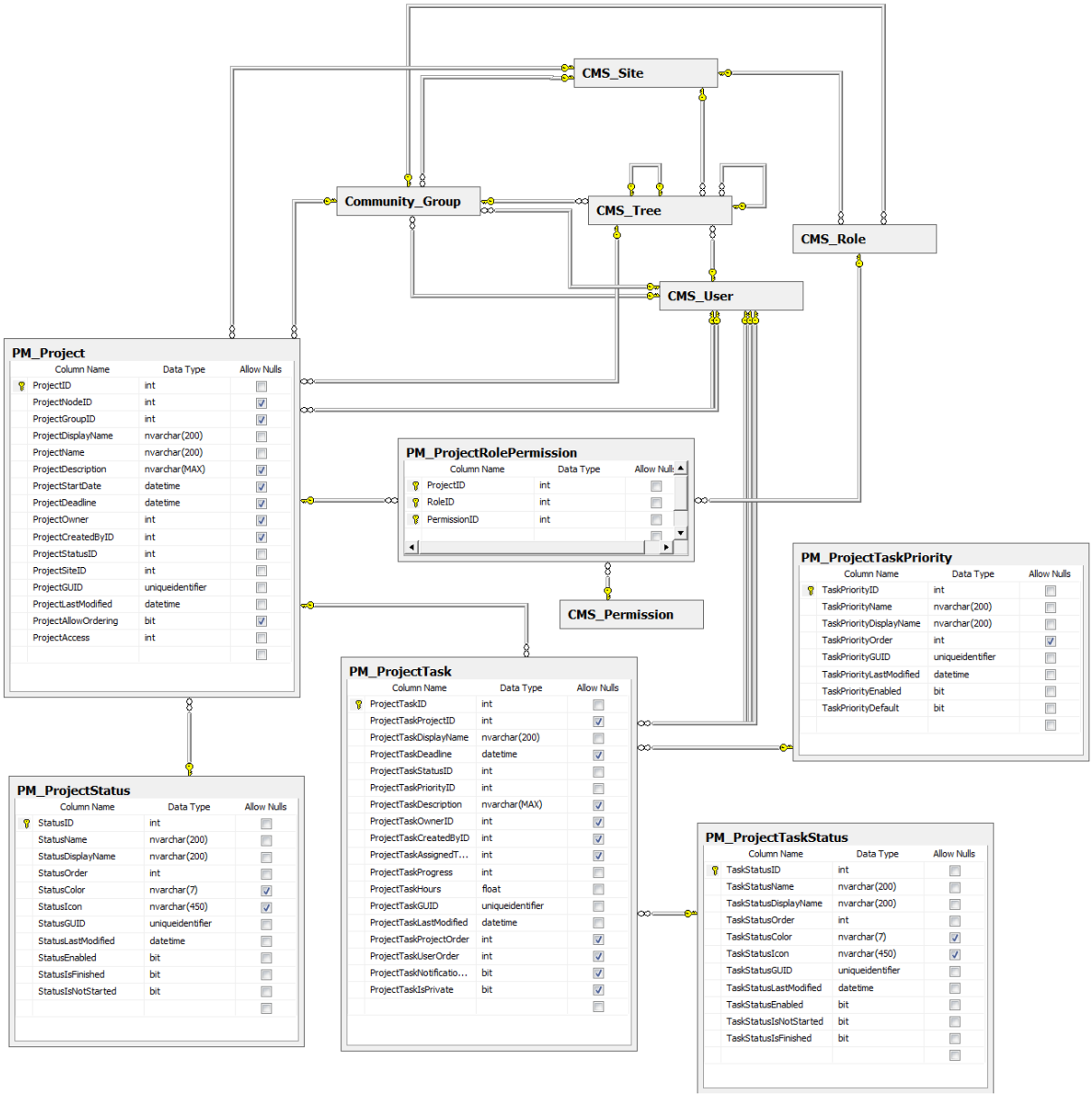
In this chapter, you will learn which database tables and API classes are used by the Project management module. You will also see the most common API examples.

Further API-related information and examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all members of the module's classes, please refer to Kentico CMS API Reference.

8.40.7.2 Database tables

The following database tables are used to store project management data:

- **PM_Project** - contains records representing projects.
- **PM_ProjectStatus** - stores possible project statuses.
- **PM_ProjectRolePermission** - stores relationships between projects and roles as well as a permission type. Each entry in this table indicates that users belonging to the given role have the specified permission for a project.
- **PM_ProjectTask** - contains records representing project management tasks.
- **PM_ProjectTaskPriority** - stores task priorities.
- **PM_ProjectTaskStatus** - stores possible task statuses.



8.40.7.3 API classes

If you are not familiar with database table data management through Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

The classes of the Project management module can be found in the **CMS.ProjectManagement** namespace.

PM_Project table API:

- **ProjectInfo** - represents one project.
- **ProjectInfoProvider** - provides management functionality for projects.

PM_ProjectStatus table API:

- **ProjectStatusInfo** - represents one project status.
- **ProjectStatusInfoProvider** - provides management functionality for project statuses.

PM_ProjectRolePermission table API:

- **ProjectRolePermissionInfo** - represents a relationship between a project, role and permission.
- **ProjectRolePermissionInfoProvider** - provides management functionality for project-role relationships.

PM_ProjectTask table API:

- **ProjectTaskInfo** - represents one task.
- **ProjectTaskInfoProvider** - provides management functionality for tasks.

PM_ProjectTaskPriority table API:

- **ProjectTaskPriorityInfo** - represents a task priority.
- **ProjectTaskPriorityInfoProvider** - provides management functionality for task priorities.

PM_ProjectTaskStatus table API:

- **ProjectTaskStatusInfo** - represents a task status.
- **ProjectTaskStatusInfoProvider** - provides management functionality for task statuses.

Other classes:

- **Reminder** - this class is used to send project management e-mail reminders.

8.40.7.4 API examples

8.40.7.4.1 Overview

These topics show examples of how the API of the Project management module can be used:

- Managing projects
- Managing tasks
- Managing project security

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Tools\ProjectManagement\Default.aspx.cs***.

The project management API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.GlobalHelper;
using CMS.CMSHelper;
using CMS.ProjectManagement;
using CMS.SiteProvider;
```

8.40.7.4.2 Managing projects

The following example creates a project.

```
private void CreateProject()
{
    ProjectStatusInfo status = ProjectStatusInfoProvider.GetProjectStatusInfo
("NotStarted");

    if (status != null)
    {
        int currentUserID = CMSContext.CurrentUser.UserID;

        // Create new project object
        ProjectInfo newProject = new ProjectInfo();

        // Set the properties
        newProject.ProjectDisplayName = "My new project";
        newProject.ProjectName = "MyNewProject";
        newProject.ProjectStatusID = status.StatusID;
        newProject.ProjectSiteID = CMSContext.CurrentSiteID;
```

```
        newProject.ProjectOwner = currentUserID;
        newProject.ProjectCreatedByID = currentUserID;

        // Save the project
        ProjectInfoProvider.SetProjectInfo(newProject);
    }
}
```

The following example gets and updates a project.

```
private bool GetAndUpdateProject()
{
    // Get the project
    ProjectInfo updateProject = ProjectInfoProvider.GetProjectInfo("MyNewProject",
CMSContext.CurrentSiteID, 0);
    if (updateProject != null)
    {
        // Update the properties
        updateProject.ProjectDisplayName =
updateProject.ProjectDisplayName.ToLower();

        // Save the changes
        ProjectInfoProvider.SetProjectInfo(updateProject);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates projects.

```
private bool GetAndBulkUpdateProjects()
{
    // Prepare the parameters
    string where = "ProjectName LIKE N'MyNewProject%'";
    string orderBy = "";
    int topN = 0;
    string columns = "";

    // Get the data
    DataSet projects = ProjectInfoProvider.GetProjects(where, orderBy, topN,
columns);
    if (!DataHelper.DataSourceIsEmpty(projects))
    {
        // Loop through the individual items
        foreach (DataRow projectDr in projects.Tables[0].Rows)
        {
            // Create object from DataRow
            ProjectInfo modifyProject = new ProjectInfo(projectDr);
```

```
            // Update the properties
            modifyProject.ProjectDisplayName =
modifyProject.ProjectDisplayName.ToUpper();

            // Save the changes
            ProjectInfoProvider.SetProjectInfo(modifyProject);
        }

        return true;
    }

    return false;
}
```

The following example deletes a project.

```
private bool DeleteProject()
{
    // Get the project
    ProjectInfo deleteProject = ProjectInfoProvider.GetProjectInfo("MyNewProject",
CMSContext.CurrentSiteID, 0);

    // Delete the project
    ProjectInfoProvider.DeleteProjectInfo(deleteProject);

    return (deleteProject != null);
}
```

8.40.7.4.3 Managing tasks

The following example creates a project management task.

```
private bool CreateProjectTask()
{
    ProjectInfo project = ProjectInfoProvider.GetProjectInfo("MyNewProject",
CMSContext.CurrentSiteID, 0);
    ProjectTaskStatusInfo status =
ProjectTaskStatusInfoProvider.GetProjectTaskStatusInfo("NotStarted");
    ProjectTaskPriorityInfo priority =
ProjectTaskPriorityInfoProvider.GetProjectTaskPriorityInfo("Normal");

    if ((project != null) && (status != null) && (priority != null))
    {
        // Create new project task object
        ProjectTaskInfo newTask = new ProjectTaskInfo();

        int currentUserID = CMSContext.CurrentUser.UserID;

        // Set the properties
        newTask.ProjectTaskDisplayName = "My new task";
```

```
        newTask.ProjectTaskCreatedByID = currentUserID;
        newTask.ProjectTaskOwnerID = currentUserID;
        newTask.ProjectTaskAssignedToUserID = currentUserID;
        newTask.ProjectTaskStatusID = status.TaskStatusID;
        newTask.ProjectTaskPriorityID = priority.TaskPriorityID;
        newTask.ProjectTaskProjectID = project.ProjectID;

        // Save the project task
        ProjectTaskInfoProvider.SetProjectTaskInfo(newTask);

        return true;
    }

    return false;
}
```

The following example gets and updates a task.

```
private bool GetAndUpdateProjectTask()
{
    // Prepare the parameters
    string where = "ProjectTaskDisplayName LIKE N'My new task%'";
    string orderBy = "";
    int topN = 0;
    string columns = "";

    // Get the data
    DataSet tasks = ProjectTaskInfoProvider.GetProjectTasks(where, orderBy, topN,
columns);
    if (!DataHelper.DataSourceIsEmpty(tasks))
    {
        // Get the project task
        ProjectTaskInfo updateTask = new ProjectTaskInfo(tasks.Tables[0].Rows[0]);
        if (updateTask != null)
        {
            // Update the properties
            updateTask.ProjectTaskDisplayName =
updateTask.ProjectTaskDisplayName.ToLower();

            // Save the changes
            ProjectTaskInfoProvider.SetProjectTaskInfo(updateTask);

            return true;
        }
    }

    return false;
}
```

The following example gets and bulk updates tasks.

```
private bool GetAndBulkUpdateProjectTasks()
```

```
{
    // Prepare the parameters
    string where = "ProjectTaskDisplayName LIKE N'My new task%'";
    string orderBy = "";
    int topN = 0;
    string columns = "";

    // Get the data
    DataSet tasks = ProjectTaskInfoProvider.GetProjectTasks(where, orderBy, topN,
columns);
    if (!DataHelper.DataSourceIsEmpty(tasks))
    {
        // Loop through the individual items
        foreach (DataRow taskDr in tasks.Tables[0].Rows)
        {
            // Create object from DataRow
            ProjectTaskInfo modifyTask = new ProjectTaskInfo(taskDr);

            // Update the properties
            modifyTask.ProjectTaskDisplayName =
modifyTask.ProjectTaskDisplayName.ToUpper();

            // Save the changes
            ProjectTaskInfoProvider.SetProjectTaskInfo(modifyTask);
        }

        return true;
    }

    return false;
}
```

The following example deletes a task.

```
private bool DeleteProjectTask()
{
    // Prepare the parameters
    string where = "ProjectTaskDisplayName LIKE N'My new task%'";
    string orderBy = "";
    int topN = 0;
    string columns = "";

    // Get the data
    DataSet tasks = ProjectTaskInfoProvider.GetProjectTasks(where, orderBy, topN,
columns);
    if (!DataHelper.DataSourceIsEmpty(tasks))
    {
        // Get the project task
        ProjectTaskInfo deleteTask = new ProjectTaskInfo(tasks.Tables[0].Rows[0]);

        // Delete the project task
        ProjectTaskInfoProvider.DeleteProjectTaskInfo(deleteTask);
```

```
        return (deleteTask != null);
    }

    return false;
}
```

8.40.7.4.4 Managing project security

The following example configures the security settings of a project.

```
private bool SetSecurity()
{
    // Get the project
    ProjectInfo project = ProjectInfoProvider.GetProjectInfo("MyNewProject",
CMSContext.CurrentSiteID, 0);

    if (project != null)
    {
        // Set properties
        project.AllowCreate = SecurityAccessEnum.AllUsers;
        project.AllowDelete = SecurityAccessEnum.Owner;
        project.AllowRead = SecurityAccessEnum.AuthorizedRoles;

        ProjectInfoProvider.SetProjectInfo(project);

        return true;
    }

    return false;
}
```

The following example assigns a project permission to a role.

```
private bool AddAuthorizedRole()
{
    // Get the project
    ProjectInfo project = ProjectInfoProvider.GetProjectInfo("MyNewProject",
CMSContext.CurrentSiteID, 0);
    RoleInfo role = RoleInfoProvider.GetRoleInfo("CMSDeskAdmin",
CMSContext.CurrentSiteID);
    PermissionNameInfo permission =
PermissionNameInfoProvider.GetPermissionNameInfo
("AccessToProject","ProjectManagement",null);

    if ((project != null) && (role != null) && (permission != null))
    {
        // Add relationship
        ProjectRolePermissionInfoProvider.AddRelationship(project.ProjectID,
role.RoleID, permission.PermissionId);
```

```
        return true;
    }

    return false;
}
```

The following example removes a project permission from a role.

```
private bool RemoveAuthorizedRole()
{
    // Get the project
    ProjectInfo project = ProjectInfoProvider.GetProjectInfo("MyNewProject",
CMSContext.CurrentSiteID, 0);
    RoleInfo role = RoleInfoProvider.GetRoleInfo("CMSDeskAdmin",
CMSContext.CurrentSiteID);
    PermissionNameInfo permission =
PermissionNameInfoProvider.GetPermissionNameInfo("AccessToProject",
"ProjectManagement", null);

    if ((project != null) && (role != null) && (permission != null))
    {
        // Remove relationship
        ProjectRolePermissionInfoProvider.RemoveRelationship(project.ProjectID,
role.RoleID, permission.PermissionId);

        return true;
    }

    return false;
}
```

8.41 Reporting

8.41.1 Overview

The Reporting module allows you to create internal reports to watch the activity in the Kentico CMS system and on the website, such as recently created documents, expired documents, website visits, user registration etc. The data the module uses is gathered from the database by using SQL queries and can be displayed in highly customizable reports including tables and graphs.

Report categories are used to group related reports together. See the Managing report categories topic for more details.

The following topics provide more details and examples on how reports can be created and used:

- Creating new reports
- Defining report parameters
- Saving reports
- Displaying reports on the website
- Report subscriptions

The Security topic describes how permissions can be set up for the reporting module.

The Reporting internals and API sub-chapter provides information about the database tables and classes used by the module and examples of how reports can be accessed using the API.

Reports are also used by the Web analytics module to display measured data, which you can learn more about in Modules -> Web analytics.

Reports can be managed in **CMS Desk -> Tools -> Reporting**.

Important!

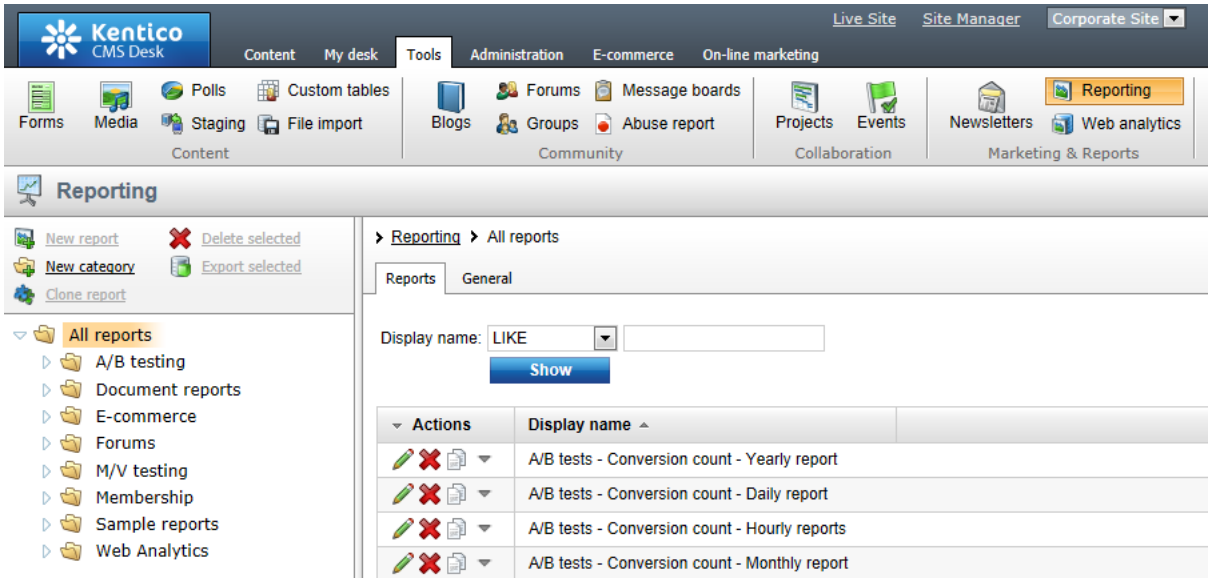
The graphs provided by the module are based on MS Charting Controls for ASP.NET.

If your application uses the 3.5 SP1 version of the .NET Framework and you plan to host it in a medium trust environment, it is necessary to ensure that the MS Chart package is installed on the server. The installation executable can be downloaded from the Microsoft website: http://www.microsoft.com/download/en/details.aspx?id=14422

The required assemblies are already included in .NET Framework 4.0, so the installation is not necessary if your application uses this version.

8.41.2 Managing report categories

All reports are organized into categories in a hierarchical tree. It is recommended to keep reports that monitor related actions in one category. You can manage the categories in **CMS Desk -> Tools -> Reporting**.



Example

The following steps describe how to create a sample report and how to display it on your website:

To start, you need to create a report category. Go to **CMS Desk -> Tools -> Reporting**, select the root of the reporting tree (the **All reports** category by default) click **New category** and enter *User reports* as the **Category display name**. You can leave the code name as *(automatic)*. Then click **OK**.

The tutorial is continued in the example section of the Creating new reports topic.

8.41.3 Creating new reports

8.41.3.1 Creating new reports

To create new reports, select a report category from the tree in **CMS Desk -> Tools -> Reporting** and click **New report**. Existing reports can also be deleted (), cloned () or exported ().

When editing a report on its **General** tab, you can configure the following properties:

Report display name	Sets the name of the report displayed in the administration interface.
Report code name	A unique name that serves as an identifier of the report, for example in the API or URLs.
Report category	Shows the category under which the report belongs. You can move the report to a different category by selecting one from the drop-down.
Allow public users to see this report	Indicates if the report should be visible by public users if it is published on the website using a reporting web part.
Connection string	Sets the database connection string used by the report's components (graphs, tables and values) when loading data. You can override this value for individual components by editing their details. Only users who have the **Set connection string** permission for the *Reporting* module are allowed to change this property's value. The system retrieves the list of connection strings from the <*connectionStrings*> section of the application's web.config file. The *(default)* option represents the *CMSConnectionString* added by the application's initial database installer. You can check the **Inherit** box to load the value from the **Default report connection string** setting configured in **Site Manager -> Settings -> Security & Membership**. You can use reporting connection strings for the following scenarios: • Retrieving data from a Separated on-line marketing database • Restricting the database-level permissions of reporting queries via a connection string with a limited database user
Enable subscription	If enabled, users with the *Subscribe* permission for the *Reporting* module are allowed to subscribe to the report and its components (graphs, tables or values). This setting is also available when configuring the details of individual

	components. Subscribing to specific components is only possible if both settings are enabled.

Enter the layout of the report using the built-in WYSIWYG editor. To retrieve and display information from the CMS database, add the following objects into the report's layout:

- Tables
- Graphs
- HTML graphs
- Values

Additionally, macro expressions as described in Development -> Macro expressions are supported in the report layout.

You can view the output of the report on the **View** tab.

Localizing strings in reports

If you need to create a single report in multiple languages, follow the instructions given in the Localization Expressions topic.

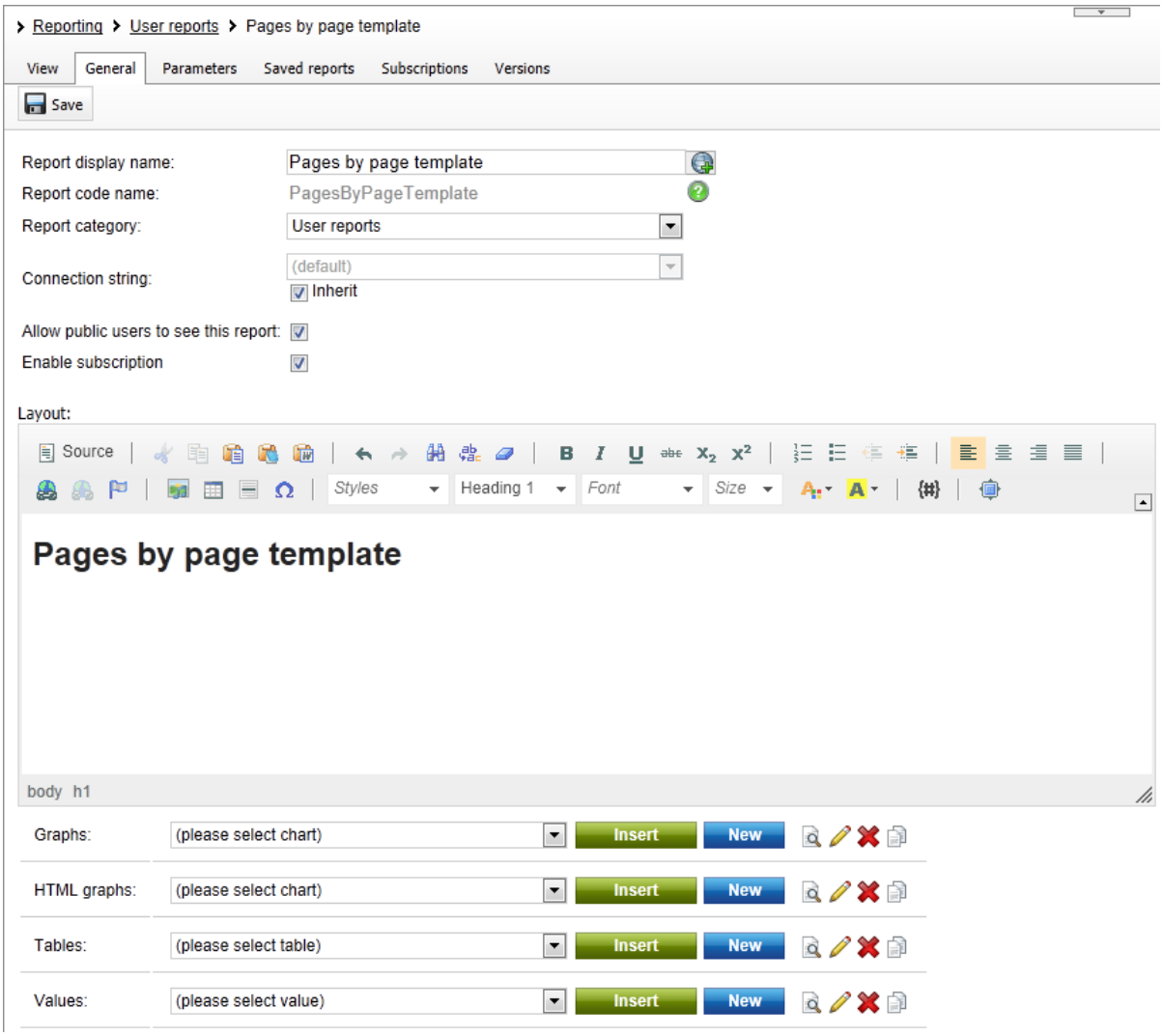
Example

1. Click **New report** and enter the following values:

- **Report display name:** Pages by page template
- **Report code name:** PagesByPageTemplate

Click **OK**. Now you can edit the layout of the report and insert tables, graphs and values.

2. The **General** tab of the report editing interface opens. Enter the following text into the **Layout** text area: *Pages by page template*. Select the text and use the WYSIWYG editor to set its **Format** to *Heading 1*.



Click **Save**.

Continued in the example section of the Creating new tables topic.

8.41.3.2 Creating new tables

Tables allow you to retrieve data from the Kentico CMS database using an SQL query. You can display the data by placing the table into the layout of a report.

The following properties can be used to configure tables:

Default	
Display name	The name of the table shown in the user interface.
Code name	Name used in your code.
Enable export	If enabled, users who view the table are able to export the displayed data to external files using the Microsoft Excel

	(XLSX), CSV or XML format. The data export feature may be accessed by right-clicking the table in the report, which opens a context menu with possible export actions.
Enable subscription	If enabled, users will be able to subscribe to the currently edited report table. To allow subscriptions, it is also necessary to have the **Enable subscription** box checked on the **General** tab of the given report.
Query	
Query	Here you can add the SQL query used to retrieve data to be displayed by the table.
Is stored procedure	Indicates if the query is a stored procedure or not.
Connection string	Sets the database connection string used by the table's query. Only users who have the **Set connection string** permission for the *Reporting* module are allowed to change this value. The system loads the list of connection strings from the <*connectionStrings*> section of the application's web.config file. The *(default)* option represents the *CMSConnectionString* added by the application's initial database installer. You can check the **Inherit** box to load the **Connection string** value set for the parent report.
No record text	Text to be displayed if the query doesn't return any data.
Skin	
Skin ID	ID of the .NET skin (stored in the *.skin* files in *~/AppThemes/<theme name>*) which will be used for the table.
Paging	
Enable paging	If enabled, paging will be enabled when the report table is displayed. The paging can be configured by the two properties below.
Page size	Number of table rows per page.
Paging mode	Type of paging controls displayed below the table. The following options are available: • **Previous-next buttons** - displays buttons leading to the previous and next page • **Page numbers** - displays page numbers leading to the corresponding pages • **Previous-next-first-last buttons** - displays buttons leading to the first, last, previous and next page • **Page numbers-first-last buttons** - displays page numbers leading to the corresponding pages and buttons leading to the first and last page

Tables are entered into the report layout editor as an expression in the following format:

%%control:ReportTable?<report code name>.<table code name>%%

This is done automatically when the **Insert** button is used.

Writing queries for tables

The queries you write for tables are standard SQL queries that pull data from the Kentico CMS database. You can find the description of Kentico CMS database tables and views in **Kentico CMS Database Reference** that is a part of the standard installation.

For information about documents, you can use the *View_CMS_Tree_Joined* table that returns published versions of all documents.

Table column names

The table column names use the same names as the column names from the returned data set. If you need to use user friendly names, you can use the following syntax in the query:

```
SELECT PageTemplateDisplayName AS [Template Name], ...
```

Example

1. Click **New** in the **Tables** section below the layout editor. Enter the following values:

- **Display name:** Pages by Page Template
- **Code name:** PagesByPageTemplate
- **Query:**

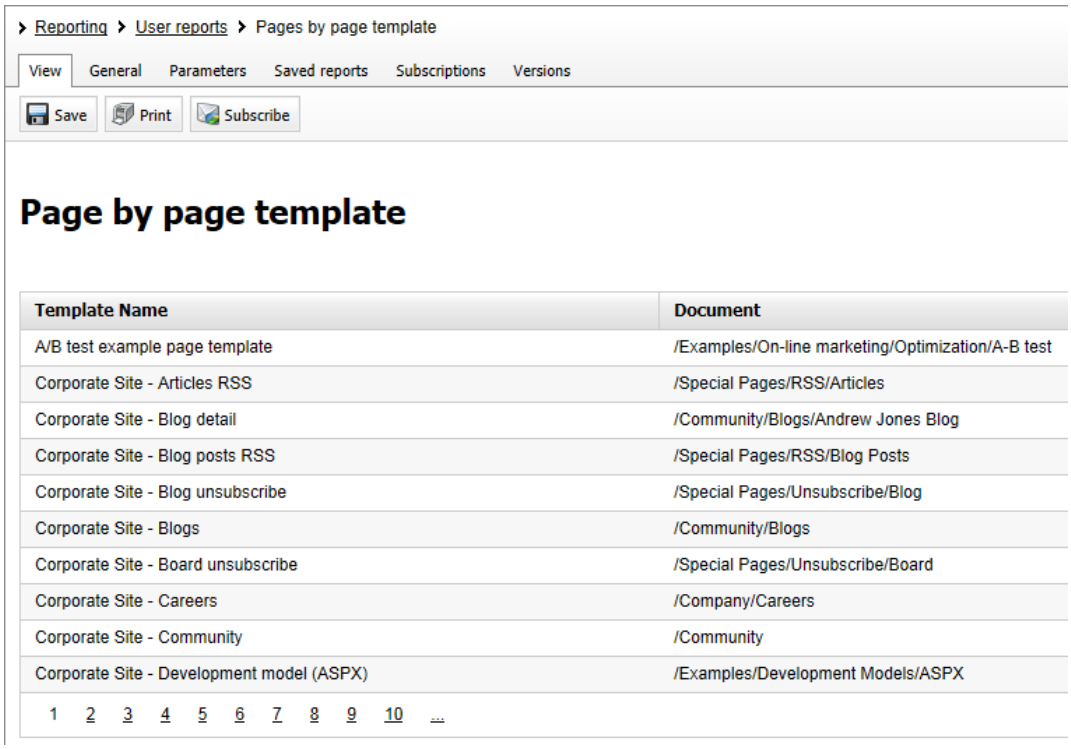
```
SELECT PageTemplateDisplayName AS [Template Name], DocumentNamePath AS [Document]
FROM View_CMS_Tree_Joined
LEFT JOIN CMS_PageTemplate
ON CMS_PageTemplate.PageTemplateID = View_CMS_Tree_Joined.DocumentPageTemplateID
WHERE PageTemplateDisplayName IS NOT NULL AND PageTemplateIsReusable = 1
ORDER BY PageTemplateDisplayName
```

- **Is stored procedure:** no
- **SkinID**: leave empty
- **Enable paging**: enabled
- **Page size**: 10
- **Page mode**: Page numbers

Click **OK**.

2. Place the cursor in the layout editor on a new line under the title, select the table from the drop-down list in the **Tables** section and click **Insert**. A string like *%%control:ReportTable?PagesByPageTemplate.PagesByPageTemplate%%* is added to the text area.

Click **Save** and switch to the **View** tab. You will see a report like this:



Continued in the example section of the Creating new graphs topic.

8.41.3.3 Creating new graphs

Graphs can be placed into the layout of a report. They allow you to retrieve data from the Kentico CMS database and display it in various types of visual formats.

The following properties can be used to configure graphs:

Default:

Display name	Display name of the graph shown in the user interface.
Code name	Code name of the graph.
Enable export	If enabled, users who view the graph are able to export the displayed data to external files using the Microsoft Excel (XLSX), CSV or XML format. The data export feature may be accessed by right-clicking the graph in the report, which opens a context menu with possible export actions.

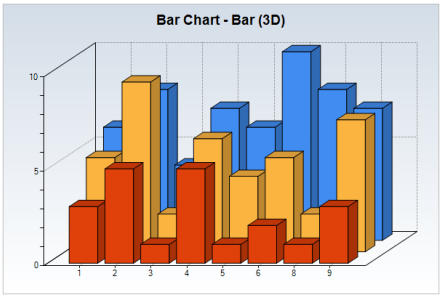
Enable subscription	If enabled, users will be able to subscribe to the currently edited report graph. To allow subscriptions, it is also necessary to have the **Enable subscription** box checked on the **General** tab of the given report.

Query:

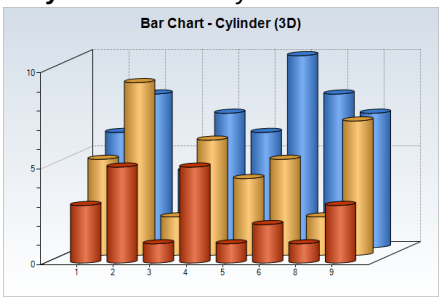
Query	Database query that extracts the data that will be displayed in the graph. It must return at least two columns: first one for categories, the other columns are used for values.
Is stored procedure	Indicates if the specified query is a stored procedure.
Connection string	Sets the database connection string used by the graph's query. Only users who have the **Set connection string** permission for the *Reporting* module are allowed to change this value. The system loads the list of connection strings from the <*connectionStrings*> section of the application's web.config file. The *(default)* option represents the *CMSConnectionString* added by the application's initial database installer. You can check the **Inherit** box to load the **Connection string** value set for the parent report.
No record text	Text to be displayed if the query doesn't return any data.

Chart type:

Graph type	The following graph types are available: **Bar chart** - bar graph, accepts multiple values and displays them next to each other. **Bar stacked chart** - bar graph, accepts multiple values and displays them on top of each other. **Pie chart** - pie graph, accepts only one column for values. **Line chart** - line graph, accepts multiple values and displays them as separate lines.
Drawing style	The following chart styles are available: **Bar chart**: • **Bar** - uses rectangular column bars

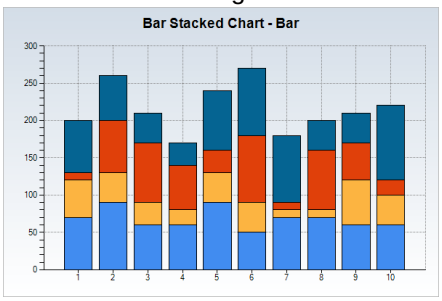


- **Cylinder** - uses cylinders

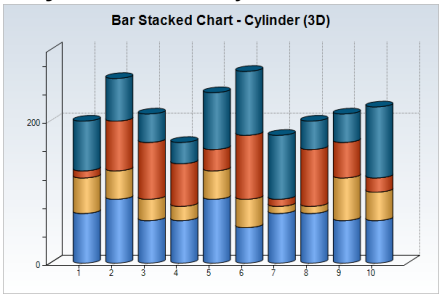


Bar stacked chart:

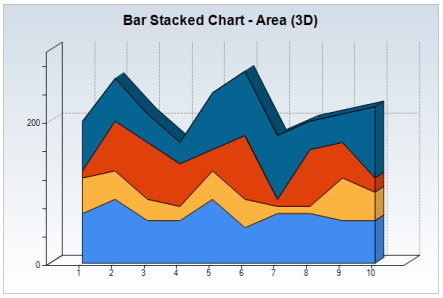
- **Bar** - uses rectangular bars



- **Cylinder** - uses cylinders

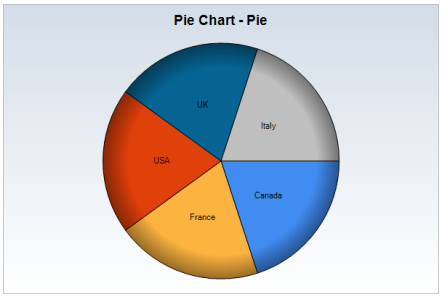


- **Area** - uses a line chart with the space under the lines filled with the respective color

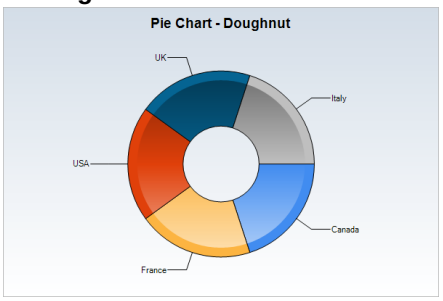


Pie Chart:

- **Pie** - uses a standard circular chart divided into sectors



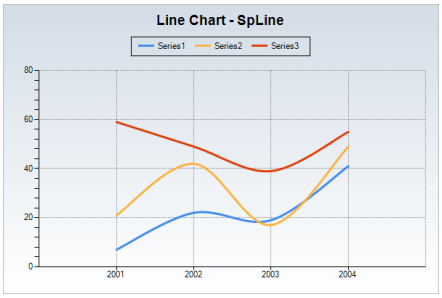
- **Doughnut** - uses a circular chart with a blank center



Line chart:

- **Line** - uses straight lines to connect values

- **SpLine** - uses smooth curved lines

	
Overlay	If enabled, charts with multiple values display them behind each other with the lower values in the front. Only available for **Bar charts** displayed in 3D.
100% stacked	If enabled, values are displayed as a percentage of their category's column. Only available for **Bar stacked charts**.
Orientation	Determines if bars are displayed horizontally or vertically. Only available for **Bar** and **Bar stacked charts** with the **Bar** drawing style.
Drawing design	Determines the aesthetic design of **Pie charts**.
Label style	Determines the style of 'pie piece' descriptions for **Pie charts**.
Doughnut radius	Determines the width of **Doughnut** style **Pie charts**. Larger numbers decrease the size of the center. Only available for **Pie charts** with the **Doughnut** drawing style.
Collect pie slices	Items that represent a smaller percentage of the **Pie chart** than the value entered here will be added together and displayed as a single item labeled *Others*. This ensures that pie charts remain legible, even if there are many items with very small values.
Show as 3D	If enabled, charts are displayed in 3D.
Rotate X	Rotates the chart around its X axis. Accepts values from -90 to 90. Only available if **Show as 3D** is enabled.
Rotate Y	Rotates the chart around its Y axis. Accepts values from -180 to 180. Only available if **Show as 3D** is enabled.
Width	Determines the width of the chart image.
Height	Determines the height of the chart image.
Show Grid	Shows a thin dotted line grid in the graph chart, Not available for **Pie charts.**

Title:

Title	Title of the chart.
Title font	Determines font properties of the chart title.
Title color	Determines the color of the chart title. Accepts standard HTML color names and hexadecimal color codes or the color selector can be used.

Title position	Determines the position of the chart title.

Legend:

Background color	Determines the background color of the legend. Accepts standard HTML color names and hexadecimal color codes or the color selector can be used.
Border color	Determines the border color of the legend. Accepts standard HTML color names and hexadecimal color codes or the color selector can be used.
Border size	Determines the size of the legend border.
Border style	Determines the style of the legend border.
Position	Determines the position of the legend in the chart.
Legend inside	If enabled, the legend is displayed inside the chart area.
Fixed legend	Allows a custom description to be set for the value in the legend that will be used instead of the name of the source column. This field is only usable for charts that display one type of series, i.e. each item has a single value. It also cannot be used for **Pie charts**. It is possible to enter static text or use a macro that resolves into the currently selected value of a report parameter in format **{%<parameter name>%}**. For example, if the report has a parameter named *CampaignName* that allows users to display the statistics of a selected marketing campaign, *{%CampaignName%}* could be entered into this field, and the legend value description would automatically contain the name of the currently displayed campaign.
Legend title	Sets the text caption of the legend.

X-axis:

X axis title	Title of the horizontal axis in the chart.
Title color	Determines the color of the X axis title. Accepts standard HTML color names and hexadecimal color codes or the color selector can be used.
X axis angle	Determines the declination angle of X axis descriptions. Setting this parameter to 90 causes upright descriptions. Accepts values from -90 to 90.
X axis format	Can be used to specify the format of item descriptions on the X axis that are in numerical or date-time format. **Numeric formatting**:

	Numbers can be formatted using .NET Custom numeric format strings enclosed in curly brackets. Examples: *{Item #}* - X axis descriptions will be displayed as *Item 1*, *Item 2* etc. *{0.00}* - numeric X axis descriptions will be displayed with precision of two decimal places. **Date and time formatting**: The format can be set using single-letter .NET Standard date and time format specifiers without quotes. In addition, any custom formatting can be defined using expressions enclosed in curly brackets. For example, *{yyyy - MM - dd - hh:mm}* would specify a date and time format like: 2010 - 08 - 19 - 12:30
Title font	Determines font properties of the X axis title.
Position	Determines the position of the X axis title.
Axis label font	Determines font properties of X axis descriptions.
X axis interval	Sets the interval between X axis descriptions.
Use X axis sorting	If enabled, values are connected in the order they appear in on the X axis, otherwise they are connected in the order they have in the returned dataset. Only used by **Line charts** and **Bar Stacked** charts with the **Area** drawing style.

Y-axis:

Y axis title	Title of the vertical axis in the chart.
Title color	Determines the color of the Y axis title. Accepts standard HTML color names and hexadecimal color codes or the color selector can be used.
Y axis angle	Determines the declination angle of Y axis descriptions. Setting this parameter to 90 causes upright descriptions. Accepts values from -90 to 90.
Y axis format	Can be used to specify the format of value descriptions on the Y axis that are in numerical or date-time format. The same formatting options can be used as in the **X axis format** field described above.
Use X axis settings	If enabled, X axis settings are used for the **Title font**, **Position** and **Axis label font** properties.
Title font	Determines font properties of the Y axis title.
Position	Determines the position of the Y axis title.
Axis label font	Determines font properties of Y axis descriptions.

Series:

Primary background color	Determines the primary color of series items in the chart. Accepts standard HTML color names and hexadecimal color codes or the color selector can be used. Not available for **Line charts**.
Secondary background color	Determines the secondary color of series items in the chart. Accepts standard HTML color names and hexadecimal color codes or the color selector can be used. Not available for **Line charts**.
Gradient	Gradient of the colors of the series items in the chart. Transitions from **Primary** to **Secondary background colors**. Not used when displayed in 3D. Not available for **Line charts**.
Border color	Determines the border color for series items in the chart. Not available for **Line charts**.
Border size	Determines the border size for series items in the chart. Not available for **Line charts**.
Border style	Determines the border style for series items in the chart. Not available for **Line charts**.
Display item value	If enabled, values are displayed above series items.
Item value format	Sets the format of the text displaying the values of series items in the chart. This overrides the **Display item value** property. Standard MS chart keywords can be placed into this field, such as for example: • **#VALX** - displays the current value of the X axis. • **#VALY** - displays the current value of the Y axis. • **#AXISLABEL** - displays the current X axis label. • **#INDEX** - displays a number determined by the order of the series item on the X axis, starting from 0. • **#SER** - displays the name of the current series, i.e. the type of the value. If the current value is numerical, the displayed format can be modified by adding the following parameters after the keyword: • **{P}** - displays the number as a percentage. • **{C}** - displays the number as a monetary amount in the currency of the current language culture. specified in the browser; please be aware that this does **not** convert the value, it only influences the format. • **{F}** - displays the number with a floating point, this is the default parameter. • **{E}** - displays the number in exponential format. The number of digits after the decimal point can be specified within the curly brackets.

	For example **#VALY{F2}** displays Y axis values with a floating point and a precision of 2 decimal places.
Line color	Determines the line color used in **Line charts**. Accepts standard HTML color names and hexadecimal color codes or the color selector can be used.
Line size	Determines the line size used in **Line charts**.
Line style	Determines the style used in **Line charts**. Not used when displayed in 3D.
Symbols	Determines the symbols used for values in **Line charts**.
Item tooltip	Determines the content and format of the tooltip that is displayed when hovering over a series item in the chart. This field supports both Kentico CMS macro expressions and standard MS chart keywords as described in the **Item value format** property.
Item link	Causes the series items in the chart to serve as links to the specified URL when clicked. The same macro expressions and keywords as described in the **Item tooltip** property can be used here as well.
Values as percent	If checked, graphs with multiple types of series (several values per item) will convert item values into a percentage out of the sum of all values for that item. For example, if an item has two values, *3* and *9*, they would be converted to *25* and *75* respectively. When using this setting, it is necessary to set the **Chart Area -> Scale max** property to at least *100* to ensure that all types of data are displayed correctly. Not available for **Pie charts**, since these already display one type of value as a percentage.

Chart area:

Primary background color	Determines the primary background color of the chart area. Accepts standard HTML color names and hexadecimal color codes or the color selector can be used.
Secondary background color	Determines the secondary background color of the chart area. Accepts standard HTML color names and hexadecimal color codes or the color selector can be used.
Gradient	Gradient of the chart area background colors. Transitions from **Primary** to **Secondary background colors**.
Border color	Determines the border color of the chart area. Accepts standard HTML color names and hexadecimal color codes or the color selector can be used.

Border size	Determines the size of the chart area border.
Border style	Determines the style of the chart area border.
Scale min	Sets the minimum Y axis value that is required for an X axis category to be displayed. Not used by **Pie charts**.
Scale max	Sets the maximum value that is displayed on the Y axis. Not used by **Pie charts**.
Ten powers	If large values are present in the chart, they are divided by appropriate ten powers and the division ratio is displayed with the y-axis title. Not used by **Pie charts**.
Reverse Y axis	If enabled, the vertical axis is reversed. Not used by **Pie charts**.
Border skin style	Determines the skin of the chart area border.

Plot area:

Primary background color	Determines the primary background color of the plot area. Accepts standard HTML color names and hexadecimal color codes or the color selector can be used.
Secondary background color	Determines the secondary background color of the plot area. Accepts standard HTML color names and hexadecimal color codes or the color selector can be used.
Gradient	Gradient of the plot area background colors. Transitions from **Primary** to **Secondary background colors**.
Border color	Determines the border color of the plot area. Accepts standard HTML color names and hexadecimal color codes or the color selector can be used.
Border size	Determines the size of the plot area border.
Border style	Determines the style of the plot area border.

Graphs are entered into the report layout editor as an expression in the following format:

%%control:ReportGraph?<report code name>.<graph code name>%%

This is done automatically when the **Insert** button is used.

Writing queries for pie charts

The queries for pie chart graphs must return two columns: the item categories and their values. The graph automatically calculates the displayed size of the given category.

Writing queries for bar graphs

The queries for bar chart graphs must return at least two columns: the item categories

and their values. If you specify more than two columns, the additional columns will be displayed side-by-side (**Bar charts**), in front of each other (**Bar charts** with the **Overlay** setting enabled), on top of each other (**Bar stacked charts**) or they will divide one column by percentage (**Bar stacked charts** with the **100% stacked** setting enabled).

Writing queries for line charts

The queries for line chart graphs must return at least two columns: the item categories and their values. If you specify more than two columns, the additional columns will be displayed as separate lines.

Example

1. Switch back to the **General** tab. Click the **New** button in the **Graphs** section below the layout editor. Enter the following values:

- **Display name:** Favorite Page Templates
- **Code name:** FavoritePageTemplates
- **Query:**

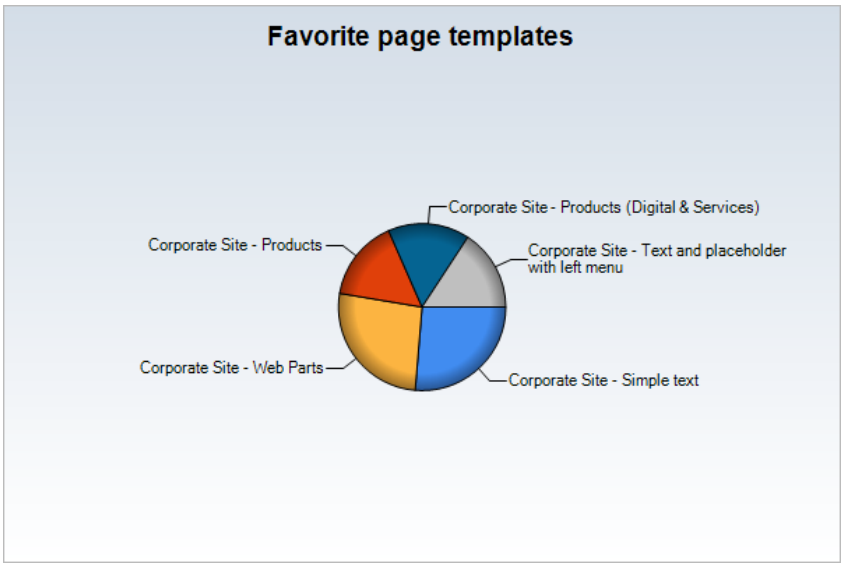
```
SELECT TOP 5 PageTemplateDisplayName AS [Template Name], COUNT
(PageTemplateDisplayName) AS [Usage]
FROM View_CMS_Tree_Joined
LEFT JOIN CMS_PageTemplate
ON CMS_PageTemplate.PageTemplateID = View_CMS_Tree_Joined.DocumentPageTemplateID
WHERE PageTemplateDisplayName IS NOT NULL AND PageTemplateIsReusable = 1
GROUP BY PageTemplateDisplayName
ORDER BY COUNT(PageTemplateDisplayName) DESC
```

- **Graph type**: Pie chart
- **Title**: Favorite page templates
- **Series -> Display item value**: Disabled (unchecked)

Click **OK**.

2. Now place the cursor in the layout editor on a new line under the table, select the graph from the drop-down list in the **Graphs** section and click **Insert**. A string like *%%control:ReportGraph?PagesByPageTemplate.MostFavoritePageTemplates%%* is added to the text area.

Click **Save** to save the changes and switch to the **View** tab. You will see a graph similar to this in the report:



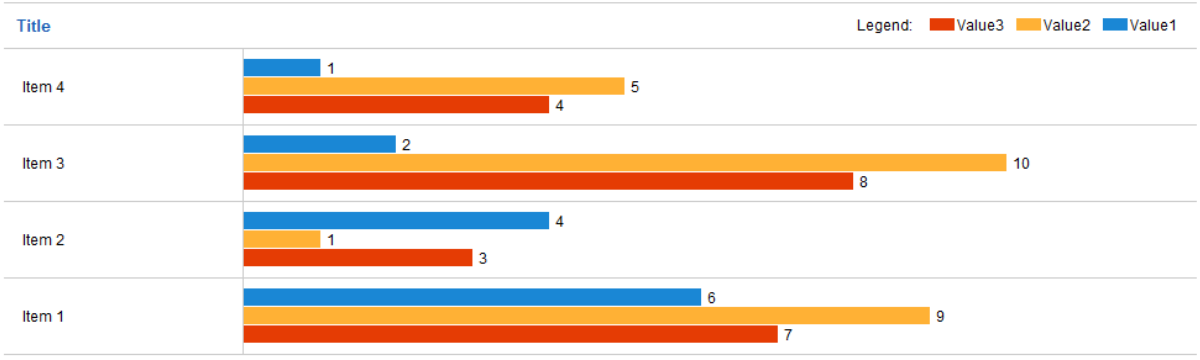
Continued in the example section of the Creating new values topic.

8.41.3.4 HTML graphs

In addition to the image-based graphs described in the previous topic, data can be visually represented in HTML graphs. Graphs of this type are composed purely out of HTML code (table and DIV elements). As a result, they can be dynamically scaled according to the amount of data that needs to be displayed, unlike an image with a predefined size.

HTML graphs always use a horizontal bar layout, which can easily be extended to display any number of items. In most cases where scaling is not an issue, it is recommended to use standard graphs, since they offer more customization options and graphical flexibility.

Like other reporting tools, HTML graphs retrieve the data to be displayed using queries. The queries must return at least two columns: the first column is used for items and the others for their values. If more than two columns are specified, the values of these additional columns will be displayed below each other as differently colored bars.



By default, data is displayed in descending order, i.e. with the newest items at the top of the graph.

When creating or editing a HTML graph, the following properties are available:

Default:

Display name	Display name of the graph shown in the user interface.
Code name	Code name of the graph.
Enable export	If enabled, users who view the graph are able to export the displayed data to external files using the Microsoft Excel (XLSX), CSV or XML format. The data export feature may be accessed by right-clicking the graph in the report, which opens a context menu with possible export actions.
Enable subscription	If enabled, users will be able to subscribe to the currently edited HTML graph. To allow subscriptions, it is also necessary to have the **Enable subscription** box checked on the **General** tab of the given report.

Query:

Query	Database query that extracts the data that will be displayed in the graph. It must return at least two columns: first one for categories, the other columns are used for values.
Is stored procedure	Indicates if the specified query is a stored procedure.
Connection string	Sets the database connection string used by the graph's query. Only users who have the **Set connection string** permission for the *Reporting* module are allowed to change this value. The system loads the list of connection strings from the <*connectionStrings*> section of the application's web.config file. The *(default)* option represents the *CMSConnectionString* added by the application's initial database installer. You can check the **Inherit** box to load the **Connection string** value set for the parent report.
No record text	Text to be displayed if the query doesn't return any data.

Title:

Title	Sets the title of the graph.

Legend:

Legend title	Sets the text caption of the legend.
Display legend	Indicates if the legend should be displayed.

Series:

Item name format	Can be used to specify the format of item descriptions on the X axis that are in numerical or date-time format. **Numeric formatting**: Numbers can be formatted using .NET Custom numeric format strings. Examples: *Item #* - X axis descriptions will be displayed as *Item 1*, *Item 2* etc. *{0.00}* - numeric X axis descriptions will be displayed with precision of two decimal places. **Date and time formatting**: The format can be set using single-letter Standard date and time format specifiers without quotes. In addition, any custom formatting can be defined. For example, *yyyy - MM - dd - hh:mm* would specify a date and time format like: 2010 - 08 - 19 - 12:30
Item value format	Sets the format of the text displaying the values of series items on the Y axis of the graph. This field supports all types of Kentico CMS macro expressions. The following context macros should be used most frequently: • **{%ser%}** - resolves into the name of the current item's series, i.e. the type of the value. • **{%xval%}** - resolves into the X axis value of the current item. • **{%yval%}** - resolves into the Y axis value of the current item. • **{%pval%}** - resolves into the percentage that the value of the current item represents out of the sum of all values in the graph. If there are multiple types of Y axis values, they are all included in the total sum. The format of the values can additionally be specified using the macro parameters listed in the topic linked above. **Examples**: • *{%ser%} = {%yval%}* - displays the name of the series and its value (e.g. *Visits = 287*). • *{%pval|(todouble)0.0|(format){0:0.0}%}%* - displays the item's percentage value rounded to one decimal place (e.g. *5.1%*).
Item tooltip	Determines the content and format of the tooltip that is displayed when hovering over a series item in the graph. The macro expressions described in the **Item value format** property can also be used in this field.
Item link	Causes the series items in the graph to serve as links to the specified

	URL when clicked. The macro expressions described in the **Item value format** property can also be used in this field.

HTML graphs are entered into the report layout editor as an expression in the following format:

%%control:ReportHtmlGraph?<report code name>.<graph code name>%%

This is done automatically when the **Insert** button is used.

HTML graphs on the live site

Because of the way they are constructed, HTML graphs are only displayed correctly in the administration interface. The necessary styles will not be applied on the live site.

Therefore, it is recommended to avoid publishing reports containing HTML graphs on your website (more information about this process can be found in the Displaying a report on the website topic).

8.41.3.5 Creating new values

A value is an object that you can place into the layout of a report, which can be used to display a single scalar value returned by a query in a specified string format.

The following properties of a value can be configured:

Default	
Display name	The name of the item in the list
Code name	Name used in your code
Enable subscription	If enabled, users will be able to subscribe to the currently edited report value. To allow subscriptions, it is also necessary to have the **Enable subscription** box checked on the **General** tab of the given report.
Query	
Query	Here you can add the SQL query used to retrieve data to be displayed by the value.
Is stored procedure	Indicates if the query is a stored procedure or not.
Connection string	Sets the database connection string used by the value's query. Only users who have the **Set connection string** permission for the *Reporting* module are allowed to change this value. The system loads the list of connection strings from the <*connectionStrings*> section of the application's web.config file. The *(default)* option represents the *CMSConnectionString* added by the application's initial database installer.

	You can check the **Inherit** box to load the **Connection string** value set for the parent report.
Format	
Formatting string	You can format the displayed value using standard .NET expressions. For example: • **{0}** - displays the value • **{0:F1}** - displays the value as a floating point number with one digit displayed after the decimal point

Values are entered into the report layout editor as an expression in the following format:

%%control:ReportValue?<report code name>.<value code name>%%

This is done automatically when the **Insert** button is used.

Writing queries for scalar value

The queries for scalar values may return any number of columns and rows, but the only value that will be displayed is the value in the first column of the first row of the result set.

Example

1. Switch back to the **General** tab. Click the **New** button in the **Values** section below the layout editor. Enter the following values:

- **Display name:** Number of pages with page template
- **Code name:** PagesWithTemplate
- **Query:**

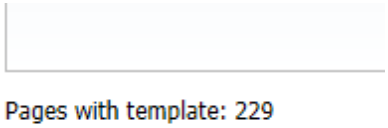
```
SELECT COUNT(DocumentID)
FROM View_CMS_Tree_Joined
WHERE DocumentPageTemplateID IS NOT NULL
```

- **Is stored procedure**: no
- **Formatting string**: Pages with template: {0}

Click **OK**.

2. Place the cursor in the layout editor under the graph, select the new value from the drop-down list in the **Values** section and click **Insert**. A string like *%%control:ReportValue?PagesByPageTemplate.PagesWithTemplate%%* is added to the text area.

Click **Save** to confirm the changes and switch to the **View** tab. You will see a text like this:



Continued in the example section of the Defining report parameters topic.

8.41.4 Defining report parameters

Reports may be filtered using parameters. You can define custom parameters on the **Parameters** tab of the **Report properties** dialog.

Context Parameters

In your queries, you can use parameters that provide information about the current context when the report is viewed, such as current user, current site, etc. Here's a list of all available context variables:

- @CMSContextCurrentUserID
- @CMSContextCurrentUserName
- @CMSContextCurrentUserDisplayName
- @CMSContextCurrentSiteID
- @CMSContextCurrentSiteName
- @CMSContextCurrentSiteDisplayName
- @CMSContextCurrentDomain
- @CMSContextCurrentTime
- @CMSContextCurrentURL
- @CMSContextCurrentNodeID
- @CMSContextCurrentCulture
- @CMSContextCurrentDocumentID
- @CMSContextCurrentAliasPath
- @CMSContextCurrentDocumentName
- @CMSContextCurrentDocumentNamePath

For example, if you want to display a list of all expired documents of the current website, you can use a query like this:

```
SELECT DocumentNamePath AS [Document path]
FROM View_CMS_Tree_Joined
WHERE DocumentPublishTo < @CMSContextCurrentTime AND NodeSiteID =
@CMSContextCurrentSiteID
```

Displaying Parameter Values in the Report

If you need to display the parameter values in the report, you can place the following macro expression in the report text:

{%parametername%}

For example:

List of documents expired on or before {%CMSContextCurrentTime%}

displays:

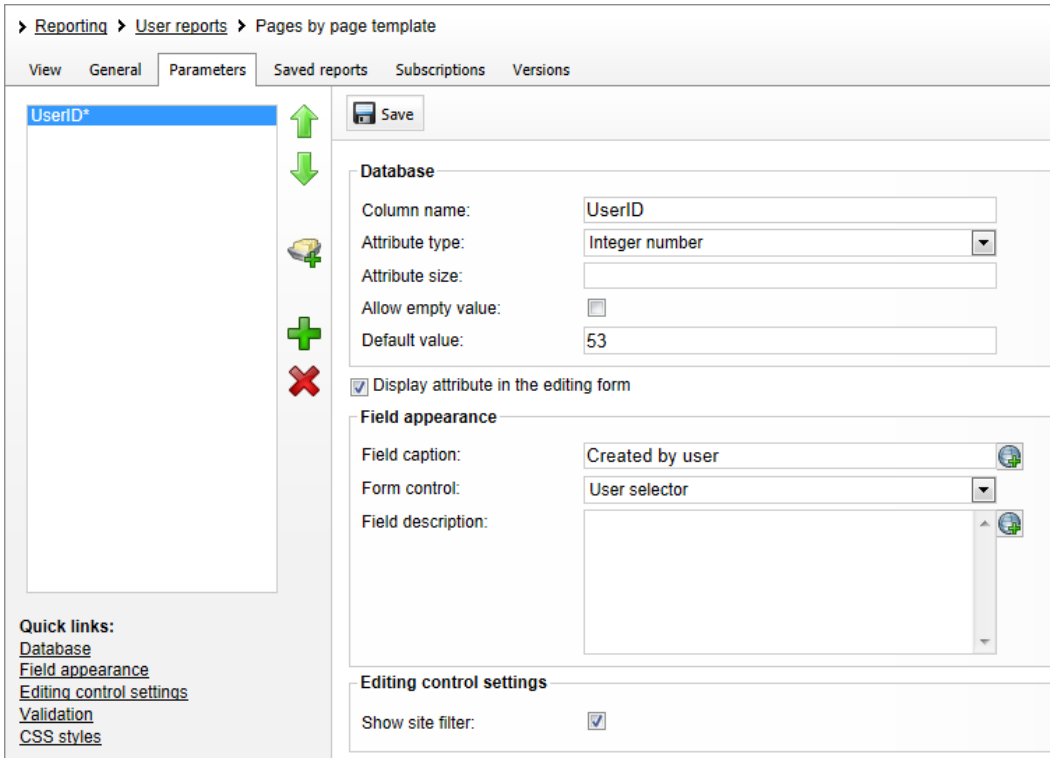
List of documents expired on or before 8/12/2007 12:06:49 PM

You can use this syntax for both custom report parameters and context parameters.

Example

1. Switch to the **Parameters** tab, click **New attribute** (+) and enter the following values:

- **Column name:** UserID
- **Attribute type:** Integer number
- **Default value:** 53
- **Field caption:** Created by user
- **Form control:** User selector



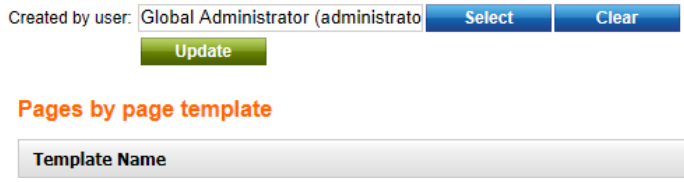
Click **Save**.

2. Now we need to add this parameter to our queries. For the purposes of this example, we will modify only the table query. Switch to the **General** tab, select the **Pages by page template** table and click **Edit**. Now modify the table SQL query like this:

```
SELECT PageTemplateDisplayName AS [Template Name], DocumentNamePath AS [Document]
FROM View_CMS_Tree_Joined
```

```
LEFT JOIN CMS_PageTemplate
ON CMS_PageTemplate.PageTemplateID = View_CMS_Tree_Joined.DocumentPageTemplateID
WHERE PageTemplateDisplayName IS NOT NULL AND DocumentCreatedByUserID = @UserID
ORDER BY PageTemplateDisplayName
```

As you can see we added the parameter to the WHERE condition of the query. All parameters that you define can be used in the query using the @<parametername> expression. Click **OK** and go to the **View** tab. You will see that the report now has a filter like this:



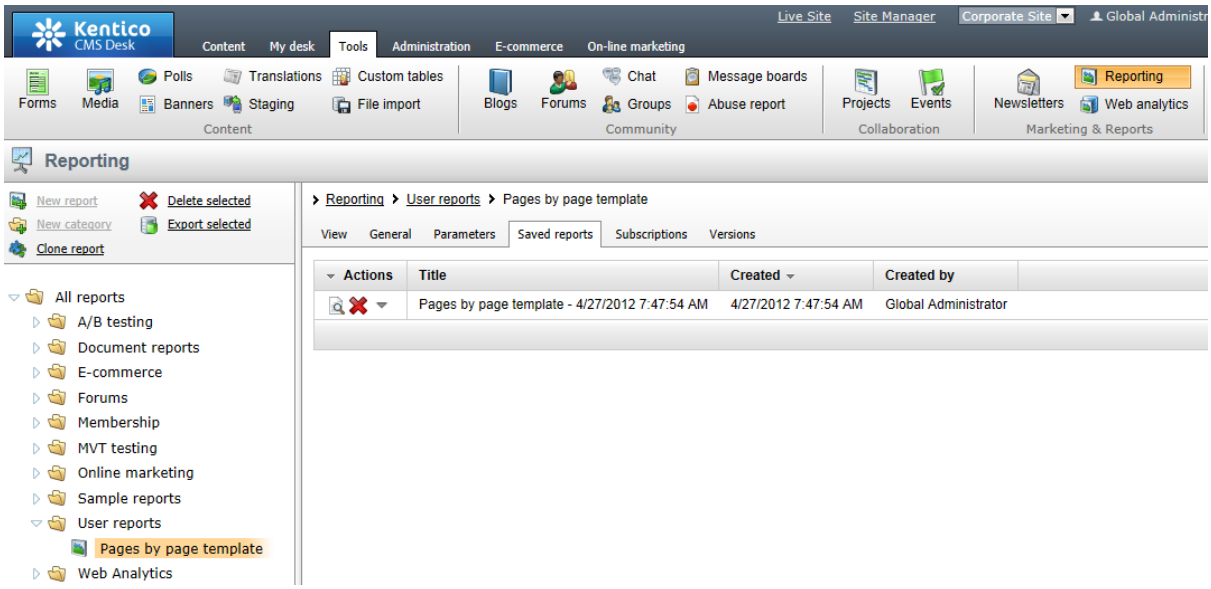
The table now only displays template names of documents that were created by the user specified in the filter.

Continued in the Saving reports topic.

8.41.5 Saving reports

When viewing a report on the **View** tab of its editing dialog, there are several ways to store the displayed data.

You can archive the entire report into the system history using the **Save** button. To view saved reports at a later time, switch to the **Saved reports** tab. If the data displayed in the report changes, the saved reports will not be affected.



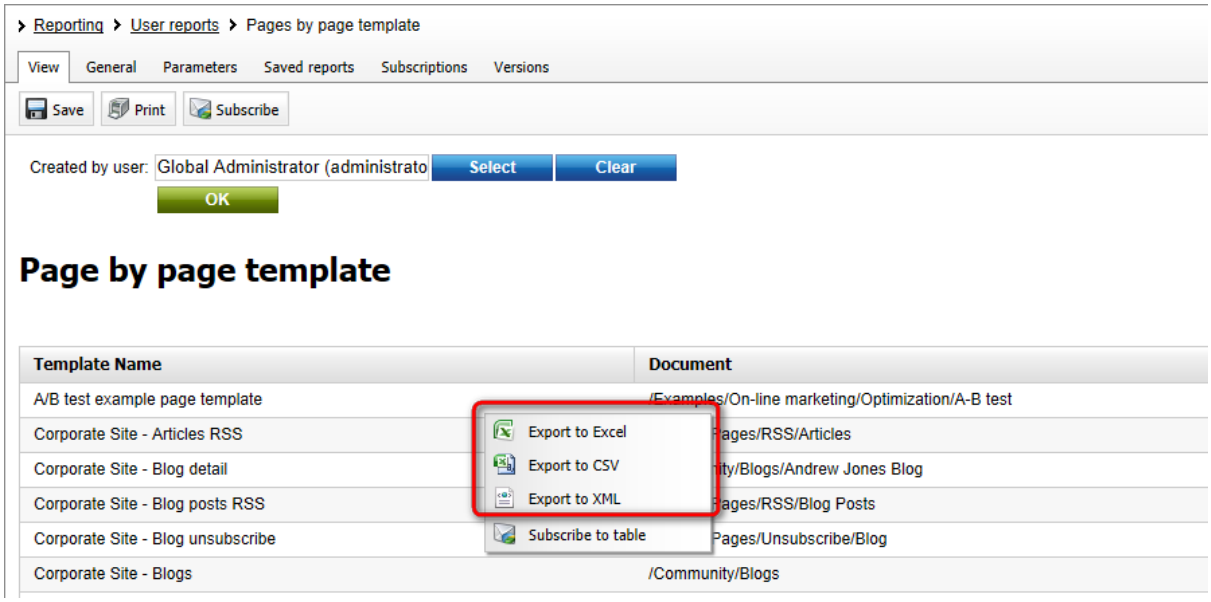
You can open the details of saved reports using the **Edit** ( ) action.

This tutorial is concluded in the example section of the Displaying reports on the website topic.

Exporting report data to files

Additionally, the data displayed in a report on the **View** tab may be exported into external files using various formats. This can be done by right-clicking on a graph or table in the given report, which will open a context menu offering the following options:

- **Export to Excel** - exports the data displayed by the given object to an XLSX spreadsheet.
- **Export to CSV** - exports data to a CSV file.
- **Export to XML** - exports data to an XML file.



After you select an action from the menu, your browser's standard file download dialog will pop up, letting you open or save the file with the exported data just like when downloading any other type of file. For more details on the data export feature, please refer to the Modules -> UI data export chapter of this guide.

Please note that the data export function may be disabled for some tables and graphs, depending on the configuration of the given reporting object.

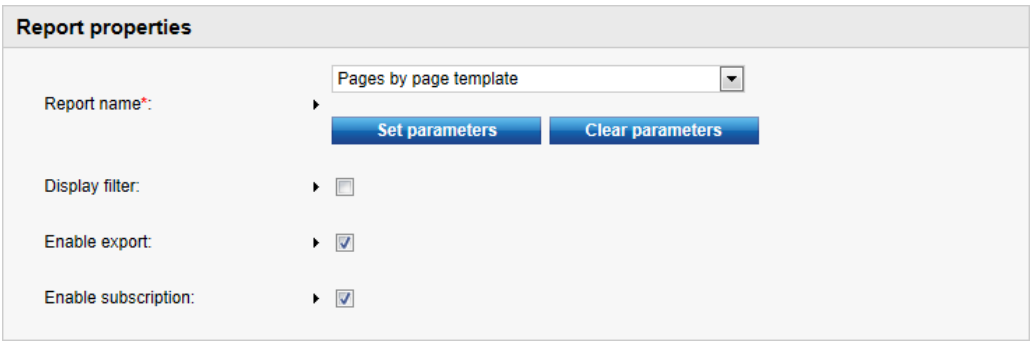
8.41.6 Displaying reports on the website

If you want to display a report on the website or include it on a custom page in the CMS administration interface, you can use the Reporting -> Report web part. All you need to configure are the following properties:

- **Report name** - select the required report.
- **Display filter** - indicates if a parameter filter should be displayed on the page (if the report has some parameters specified).
- **Enable export** - if enabled, users will be able to export the data displayed in the report's charts and tables to other formats (Excel, CSV or XML). The export feature may be accessed using a context menu that can be opened by right clicking the rendered chart/table. Please note that data export will

not be allowed if it is disabled for the given chart/table using the properties in the main reporting interface.
- **Enable subscription** - indicates whether authenticated users should be allowed to subscribe to the components (graphs, tables or values) in the displayed report. Subscription can be done by right clicking on the appropriate item and selecting the *Subscribe to* option in the context menu. In addition to this property, subscription also needs to be enabled for the given report and the specific graph, table or value in the main reporting interface. Please see the Report subscriptions topic for more information.

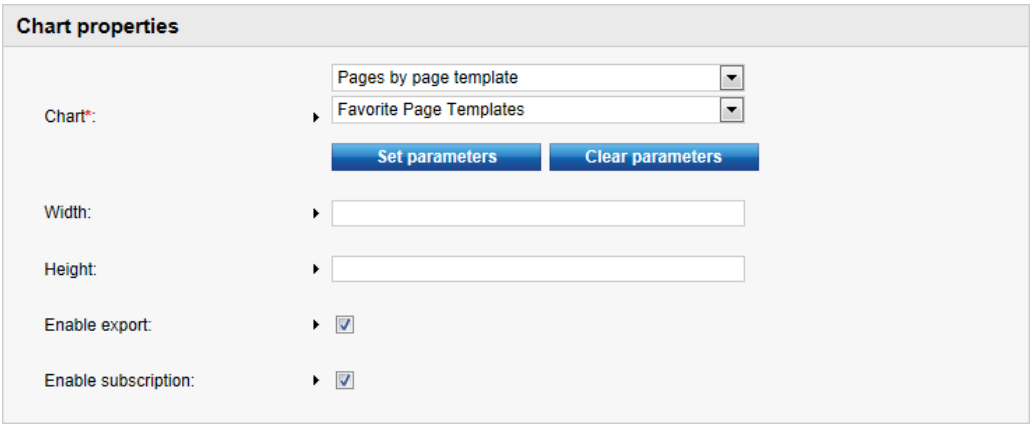
If the selected report has some parameters defined, their values can be entered here by using the **Set parameters** button. However, if the **Display filter** property is enabled, these settings will be ignored, as the parameters will be be configurable on the live site using the filter.



If you only wish to display a certain graph, table or value from the report, this can be done by using one of the other web parts from the **Reporting** category:

- **Report chart**
- **Report table**
- **Report value**

The exact chart/table/value must be specified in the respective property of the web part.



The first drop-down list is used to select the report, the second one specifies the chart, table or value. Values of the report's parameters can be specified using the **Set parameters** button.

These web parts are also available as Widgets.

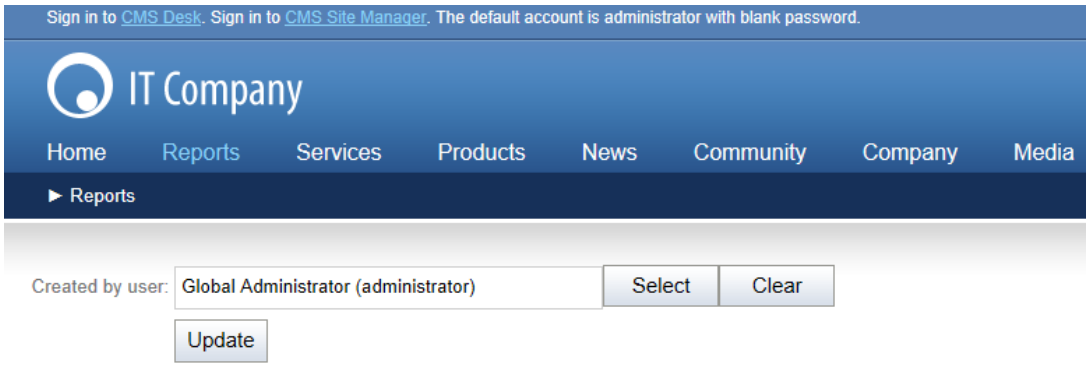
Example

1. To display this report on your website, switch to the **Content** tab of CMS Desk. Select the root of your website and create a new **Page (menu item)** type document. Name it *Reports*, choose the **Create a blank page** option and click **Save**.

2. Now add the **Reporting/Report** web part to the page's web part zone on the **Design** tab and set its properties to the following:

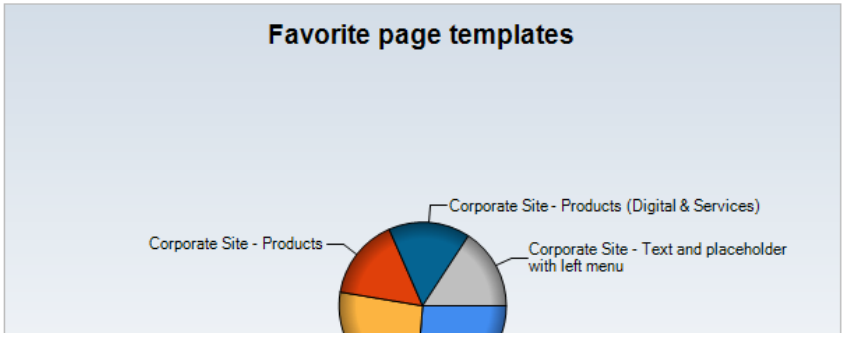
- **Report name** - select *Pages by page template*
- **Display filter** - enabled

3. Now go to **Live site** mode and you should see something similar to the following:



Pages by page template

Template Name	Document
A/B test example page template	/Examples/On-line marketing/Optimization/A-B test
Corporate Site - Articles RSS	/Special Pages/RSS/Articles
Corporate Site - Blog detail	/Community/Blogs/Brad Summers Blog
Corporate Site - Blog posts RSS	/Special Pages/RSS/Blog Posts
Corporate Site - Blog unsubscribe	/Special Pages/Unsubscribe/Blog
Corporate Site - Blogs	/Community/Blogs
Corporate Site - Board unsubscribe	/Special Pages/Unsubscribe/Board
Corporate Site - Careers	/Company/Careers
Corporate Site - Community	/Community
Corporate Site - Development model (ASPX)	/Examples/Development Models/ASPX
1 2 3 4 5 6 7 8	



As you can see, the report is displayed just like on the **View** tab of the **Report Properties** dialog and the parameter filter can be used by site visitors.

8.41.7 Report subscriptions

If a user is interested in the content of a specific report or wants to follow the progress of its data, they can use the subscription feature of the reporting module. Subscribers will regularly receive e-mails with the up-to-date status of the given report. This way, users can easily keep track of the data provided by the report simply by checking their e-mail, without having to access the website or its administration interface.

Requirements

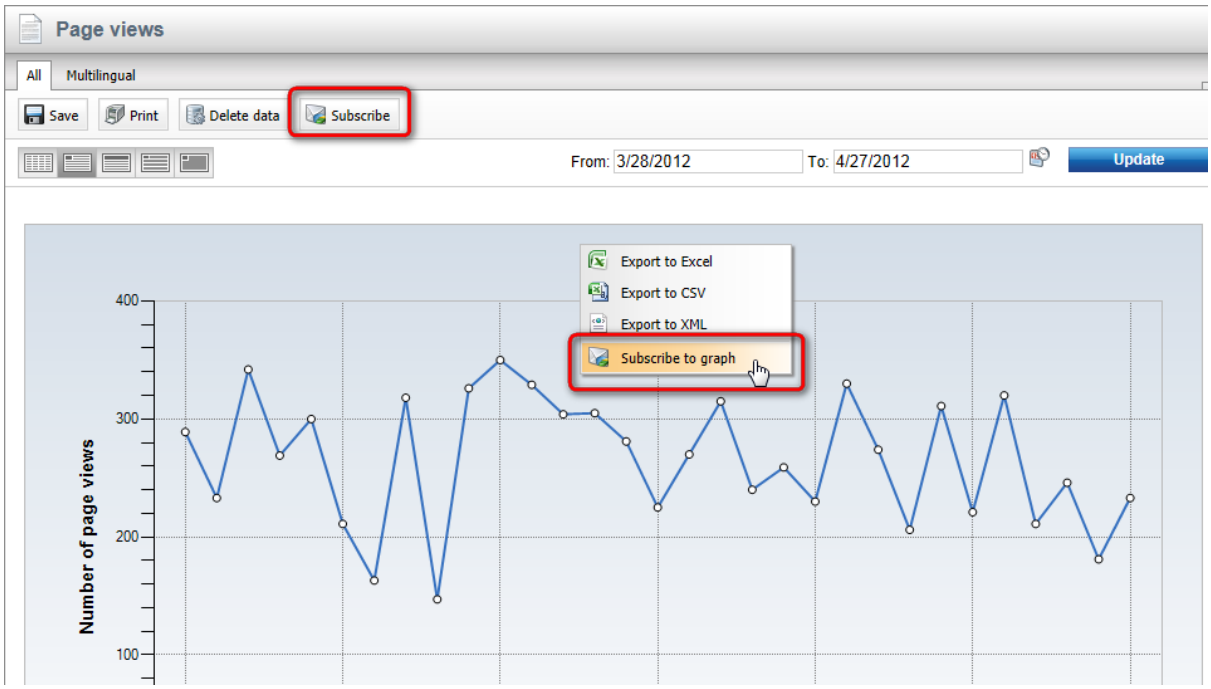
Each report has settings that determine whether subscriptions to it should be allowed. On the **General** tab of the report editing interface, the **Enable subscription** checkbox applies to the entire report. The same setting is available when configuring the details of individual components (graphs, tables or values) . If both settings are enabled, it is possible to subscribe to specific components. For reports published on the website's pages, subscription also needs to be allowed through the **Enable subscription** property of the web part or widget used to display the given report.

Additionally, the subscription feature is only available for users who belong to a role that has the **Subscribe** or **Modify** permission for the *Reporting* module.

Subscribing to reports

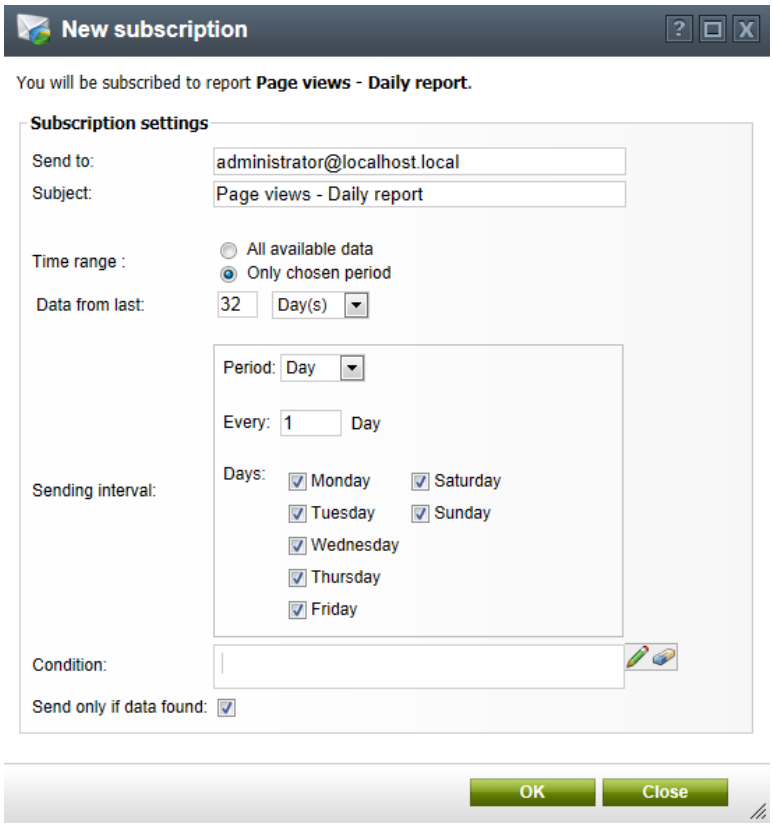
If all of the conditions above are met, users can subscribe to entire reports displayed in the administration interface by clicking on the **Subscribe** action in the header of the given page. This can either be done on a report's **View** tab in **CMS Desk -> Tools -> Reporting** or in any other sections of the UI that provide reports, such as Web analytics.

Subscriptions to individual components can be created by right-clicking on the appropriate item in the report and selecting the **Subscribe to** option in the offered context menu. This way of subscribing is also available for authenticated users when viewing reports on the live site.



Both of these action open a subscription dialog, where the following details can be configured:

- **Send to** - sets the e-mail address to which the subscription e-mails will be sent. By default, the address of the current user is loaded, but a different one may be specified.
- **Subject** - sets the subject that will be used for the subscription e-mails.
- **Time range** - this setting is only available for reports that offer the possibility of displaying data from a specific time range, selected using *From* and *To* parameters, e.g. most web analytics and on-line marketing reports in the system. Users can choose to either have *All available data* included in every subscription e-mail, or only a certain range of the most recent data according to the date and time when the e-mail is sent. With the second option, the exact time range can be specified through the *Data from last* field below.
- **Sending interval** - the interval settings define how often and when exactly the report subscription e-mails should be sent.
- **Condition** - may be used to enter an additional macro condition that must be fulfilled in order for the subscription e-mails to be sent. You can write any condition according to your specific requirements. For detailed information about macro conditions, please see the Development -> Macro expressions -> Macro conditions topic.
- **Send only if data found** - if checked, subscription e-mails will only be sent if the report contains data at the given time (the subscriber will not receive empty reports).

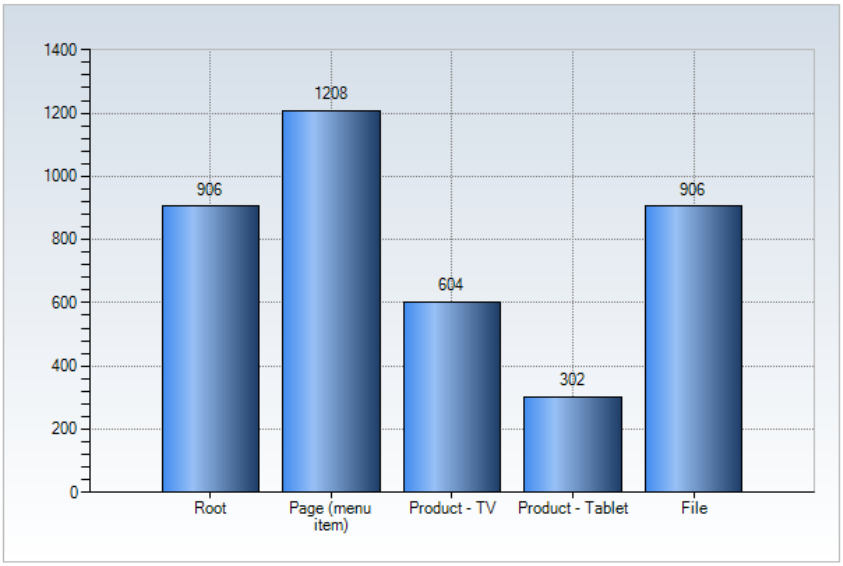


If the report has any parameters that affect how its data is displayed, the report e-mails will always be sent according to the values set before the subscribe action was used.

Unsubscription

Users can cancel their subscription to a report at any time by clicking on the unsubscription link included in every e-mail (when using the default e-mail template).

Full report subscription



Class name	Random value
Root	1068
Page (menu item)	1424
Product - TV	712
Product - Tablet	356
File	1068

Report value: 20

Click here to unsubscribe

The link leads to the *~/CMSModules/Reporting/CMSPages/Unsubscribe.aspx* system page, where the user can confirm that they wish to unsubscribe from the given report. The appropriate subscription to be removed is automatically identified using parameters passed in the query string of the link's URL.

Alternatively, users with access to CMS Desk can view their report subscriptions in **My Desk -> My profile -> Subscriptions** and manually **Unsubscribe** (✖) as needed. All report subscriptions created by the current user are listed here, even if the target e-mail address belongs to someone else.

Report subscriptions

You are currently subscribed to receive the following reports:

Actions	Subject	E-mail
✖ ✎	A/B test - Conversions by variations - Daily report	administrator@localhost.local
✖ ✎	Pages by page templates	administrator@localhost.local
✖ ✎	Sample report - Table	private@localhost.local

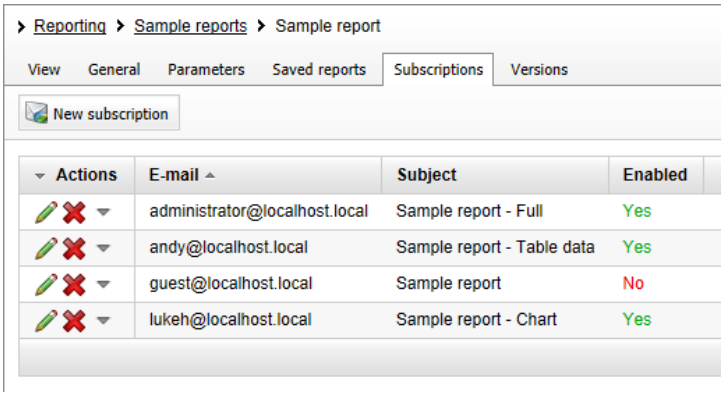
For live site users, the same option can be provided by the My account web part, as long as its **Display**

my subscriptions and **Display report subscriptions** properties are enabled.

After unsubscribing through any of the described ways, a notification e-mail is automatically sent to the given address.

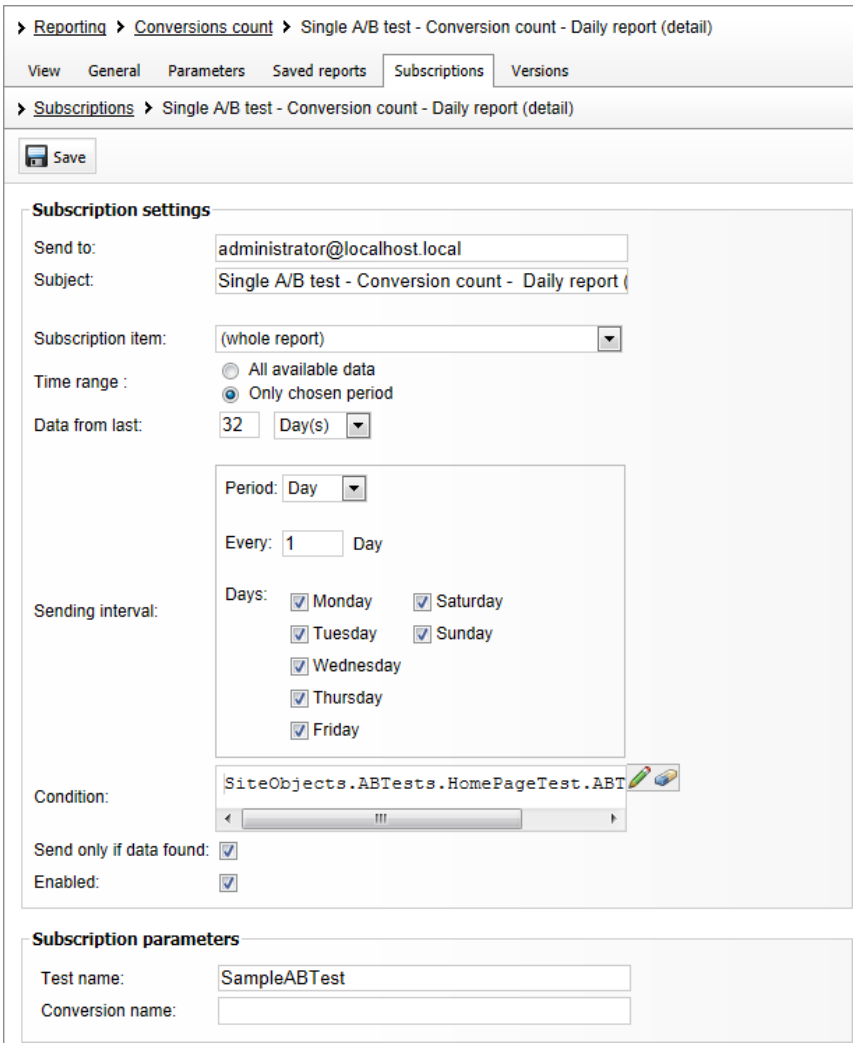
Managing report subscriptions

When editing a report in **CMS Desk -> Tools -> Reporting**, it is possible to view and manage all of its subscriptions on the **Subscriptions** tab. The **Edit** icon can be used to modify the details of existing subscriptions. Individual subscriptions may be manually removed using the **Delete** action and new ones can be added to the report by clicking the **New subscription** button in the tab's header.



Creating or editing a subscriptions here offers the standard subscription dialog as described above, but with several additional options.

- The **Subscription item** selector may be used to choose whether the subscription e-mails should include the full content of the report or only a specific graph, table or value component.
- Individual subscriptions may be manually disabled by unchecking the **Enabled** property.
- If the currently edited report has parameters, the **Subscription parameters** section below the main settings provides a way to enter the values that should be used for the reports sent to the subscriber. Only parameters that are configured to be visible when viewing the report can be edited (i.e. those that have the **Display attribute in the editing form** checkbox enabled on the **Parameters** tab).



Time range parameters

Some reports offer the possibility of displaying data from a specific time span selected through parameters (e.g. most web analytics and on-line marketing reports in the system). This functionality is ensured by two parameters with **Column names** set to *FromDate* and *ToDate* respectively on the **Parameters** tab.

These two parameters are not displayed in the **Subscription parameters** section. Instead, subscriptions to reports that contain them provide the **Time range** and **Data from last** settings, which may be used to specify the appropriate values.

Every authenticated user may also configure the options described above for their own report subscriptions by going to **My Desk -> My profile -> Subscriptions** or a page containing the My account web part**,** and editing () the appropriate subscription.

Scheduling subscription mailout

To ensure that the report subscription e-mails are sent correctly and at the appropriate time, the system uses a global scheduled task called **Report subscription sender**. When executed, this task goes through all enabled reporting subscriptions and checks their settings. If the time interval requirements and condition of a subscription are fulfilled, an e-mail with the current content of the given report is sent out.

By default, the task is executed every minute. If necessary, you can change its configuration in **Site Manager -> Administration -> Scheduled tasks**. However, if you wish to use report subscriptions, the task must always be enabled and scheduled frequently.

Important!

To be able to send e-mails, the system needs to be connected to a working SMTP server. To learn how this can be done, please see the SMTP server configuration topic.

E-mail templates

The content of the reporting e-mails sent to subscribers is based on e-mail templates. If you wish to customize the e-mails in some way, you can edit the templates in **Site Manager -> Administration -> E-mail templates**. The following templates are available:

- **Reporting - Subscription template** - defines the content of the main subscription e-mails used to send the report status.
- **Reporting - Subscription confirmation** - used for the automatic notification e-mails sent to the recipient when a new subscription is created.
- **Reporting - Unsubscription confirmation** - used for the notifications that users receive after unsubscribing (or when the subscription is removed by an administrator).

Reports in plain text

If your system is configured to send e-mails in plain text format, the content of report subscription e-mails will be limited. In this case, image based graphs are included as attachments and the data from tables is sent in an a CSV file. HTML graphs are not represented at all.

You can choose the preferred e-mail format using the **E-mail format** setting that can be found in **Site Manager -> Settings -> System -> E-mails**.

The following context macros may be used in report subscription e-mail templates:

- **{% SubscriptionBody %}** - this macro is resolved into the content of the subscribed report or component.
- **{% DefaultSubscriptionCSS %}** - used to add the CSS styles that are required to properly display report content in the e-mail. This should always be included in the *<head>/<style>* element in the HTML version of the subscription template.
- **{% ItemName %}** - for subscriptions to individual reporting components, this macro resolves into the name and type (graph, table or value) of the given item. In the case of full report subscriptions, it

returns an empty string.
- **{% UnsubscriptionLink %}** - generates a link that can be used to cancel the given subscription.

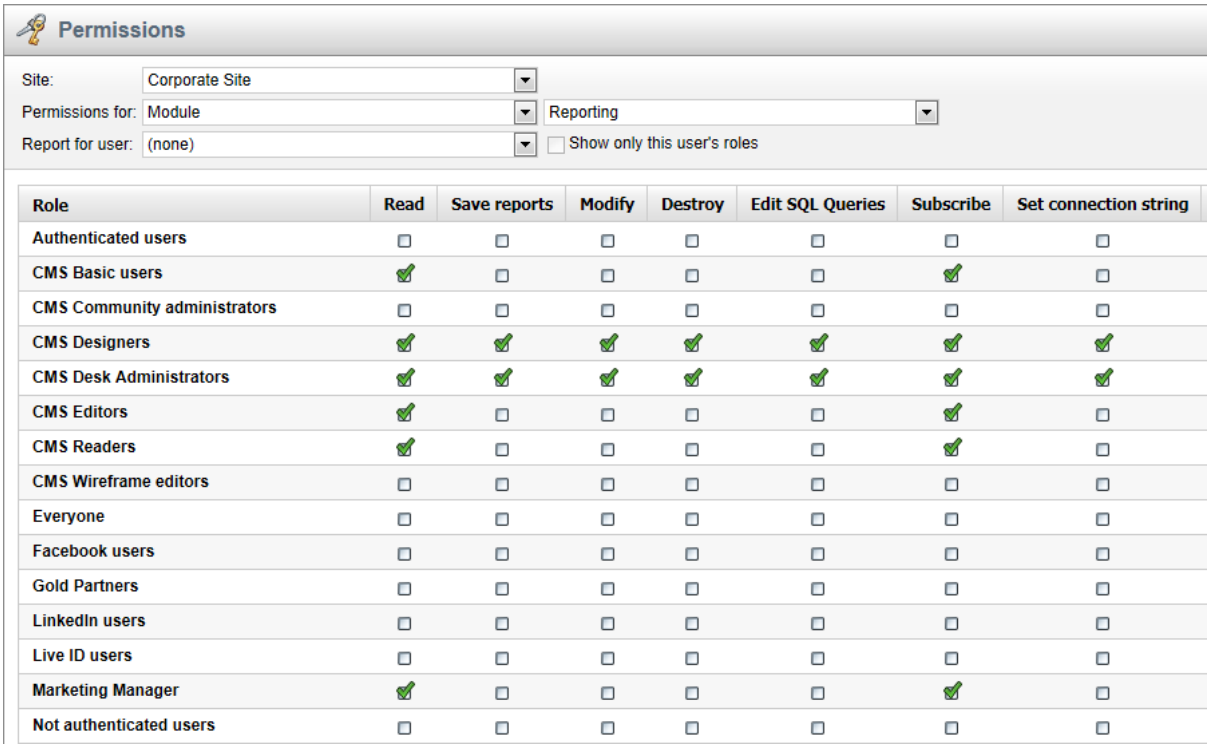
It is also possible to access the following related objects and their properties (e.g. *{% Report.ReportDisplayName %}*):

- **{% Report %}** - *ReportInfo* object representing the report to which the user is subscribed.
- **{% ReportSubscription %}** - *ReportSubscriptionInfo* object of the subscription for which the e-mail is being sent.

8.41.8 Security

You can configure access to the Reporting module in **Site manager / CMS Desk -> Administration -> Permissions**. Choose the permission matrix for **Module** -> **Reporting** and assign the available permissions to the appropriate user roles.

Permission	Description
Read	Allows users to view existing reports.
Save reports	Allows users to save reports into the report archive.
Modify	Allows users to create, modify and delete reports. This also grants permission to subscribe to reports.
Destroy	Allows users to delete the version history of reporting objects.
Edit SQL queries	Allows editing of the queries used to retrieve data for reporting components (graphs, tables and values). This permission is needed to create new reports, but it can be a security risk, since it allows users to run queries against the website's database.
Subscribe	Allows users to subscribe to reports and their components. Subscription also needs to be allowed for individual reports through their properties. Unauthenticated (public) users cannot subscribe to reports.
Set connection string	Allows users to change the connection string property of reports and their components. The report uses the specified connection string to access the database when loading data.



Role	Read	Save reports	Modify	Destroy	Edit SQL Queries	Subscribe	Set connection string
Authenticated users	☐	☐	☐	☐	☐	☐	☐
CMS Basic users	☑	☐	☐	☐	☐	☑	☐
CMS Community administrators	☐	☐	☐	☐	☐	☐	☐
CMS Designers	☑	☑	☑	☑	☑	☑	☑
CMS Desk Administrators	☑	☑	☑	☑	☑	☑	☑
CMS Editors	☑	☐	☐	☐	☐	☑	☐
CMS Readers	☑	☐	☐	☐	☐	☑	☐
CMS Wireframe editors	☐	☐	☐	☐	☐	☐	☐
Everyone	☐	☐	☐	☐	☐	☐	☐
Facebook users	☐	☐	☐	☐	☐	☐	☐
Gold Partners	☐	☐	☐	☐	☐	☐	☐
LinkedIn users	☐	☐	☐	☐	☐	☐	☐
Live ID users	☐	☐	☐	☐	☐	☐	☐
Marketing Manager	☑	☐	☐	☐	☐	☑	☐
Not authenticated users	☐	☐	☐	☐	☐	☐	☐

Permission matrix of the Reporting module

Making reports available on the live site

The **Allow public users to see this report** property available on the **General** tab when **Editing** (✎) reports indicates if the system displays the given report on the live site to non-authenticated (public) users. If false, the report and all of its components (graphs, tables or values) are always hidden from public users.

Specifying connection strings for reports

You can restrict the database-level permissions of the SQL queries used for reporting by registering custom connection strings and assigning them to reports.

1. Prepare a user account for your Kentico CMS database with the required security configuration.

2. Edit your application's *web.config* and add a new connection string into the *<configuration><connectionStrings>* section. Enter authentication information (the *user id* and *password*) for the appropriate database user account.

```
<add name="CMSReadOnlyConnectionString" connectionString="Persist Security
Info=False;database=DBName;server=ServerName;user id=DBUser;password=password;
Current Language=English;Connection Timeout=240;" />
```

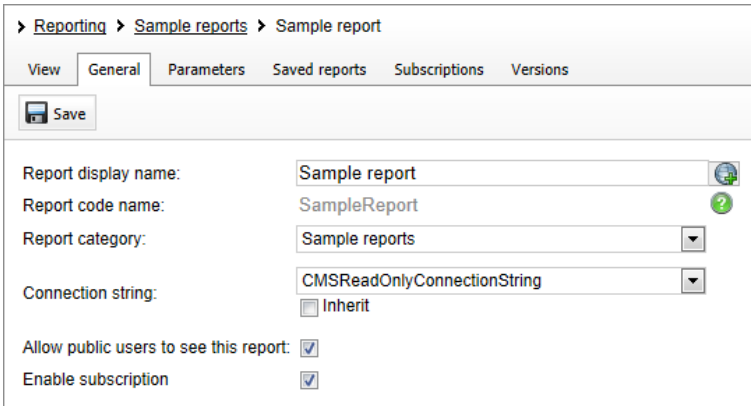
3. Go to **Site Manager -> Settings**, select the **Security & Membership** category and choose the **Default report connection string** (in the **Reporting** section).

Connection string names

The setting loads the list of connection strings from the web.config and displays their *name* attribute values. The *(default)* option represents the *CMSConnectionString* added by the application's initial database installation.

The system assigns the specified connection string to newly created reports. By default, all existing reports also inherit the connection string value from this setting.

4. Open the **CMS Desk -> Tools -> Reporting** interface and edit any reports for which you wish to set a non-default connection string. On the **General** tab, uncheck the **Inherit** box below the **Connection string** property and select a different option. You can also override the connection string value for individual reporting components (graphs, tables and values).



Assigning a connection string to a specific report

The system now uses the assigned connection string when executing the queries of reports. This limits the functionality of the queries according to the database permissions of the user account specified in the connection string. Only users who have the **Set connection string** permission for the *Reporting* module are allowed to change the connection strings of individual reports.

8.41.9 Reporting internals and API

8.41.9.1 Overview

In this chapter, you will learn which database tables and API classes are used in the Reporting module. You will also see the most common API examples.

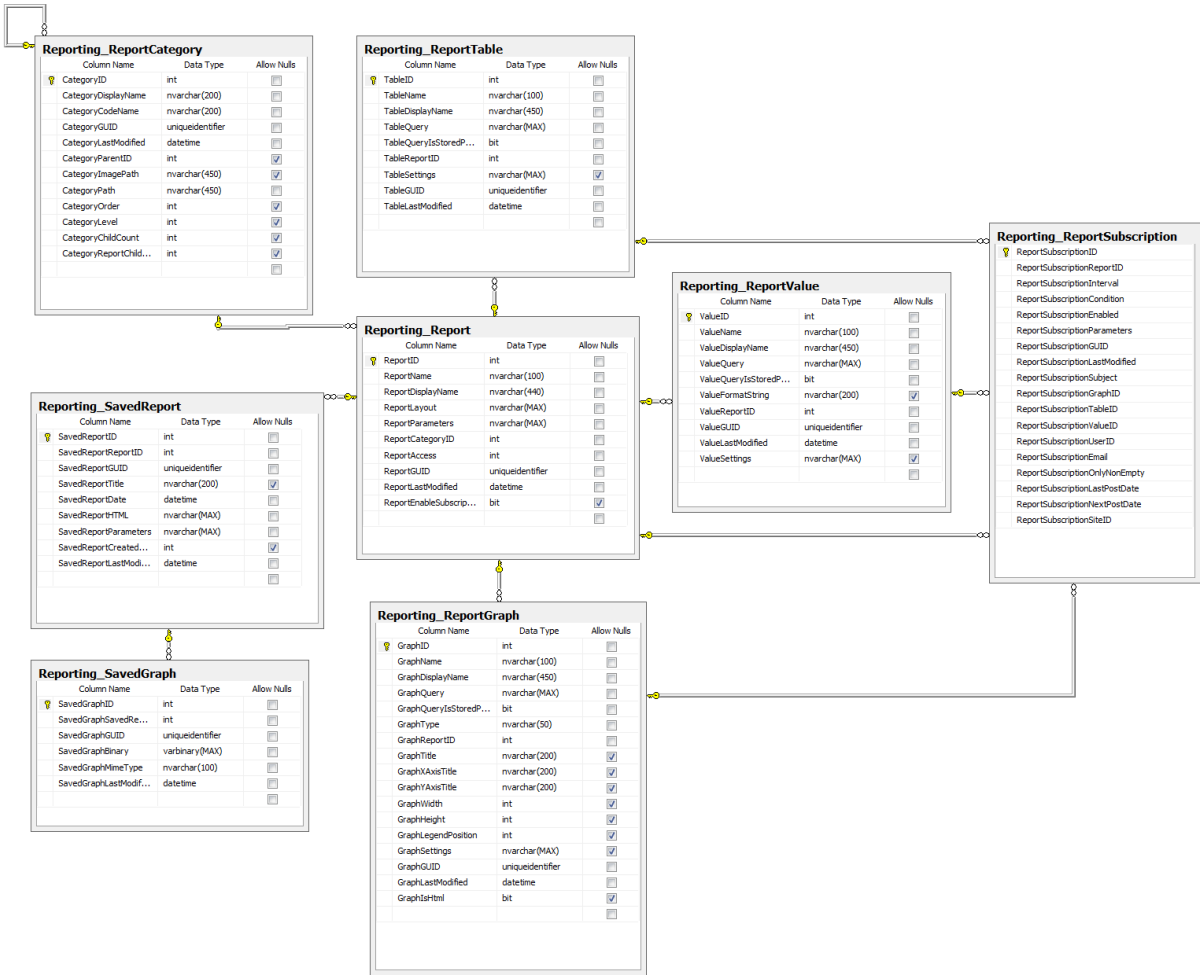
Further API-related information and examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all members of the module's classes, please refer to Kentico CMS API Reference.

8.41.9.2 Database tables

The following database tables are used to store information about reports:

- **Reporting_ReportCategory** - contains records representing report categories.

- **Reporting_Report** - contains records representing reports and their content.
- **Reporting_ReportGraph** - contains records representing report graphs.
- **Reporting_ReportTable** - contains records representing report tables.
- **Reporting_ReportValue** - contains records representing report values.
- **Reporting_SavedReport** - contains records representing saved reports in the system.
- **Reporting_SavedGraph** - stores graphs contained in saved reports (in binary format).
- **Reporting_ReportSubscription** - contains records representing report subscriptions.



8.41.9.3 API classes

If you are not familiar with database table data management through Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

The classes of the Reporting module can be found in the **CMS.Reporting** namespace.

Reporting_ReportCategory table API:

- **ReportCategoryInfo** - represents one report category object.
- **ReportCategoryInfoProvider** - provides management functionality for report categories.

Reporting_Report table API:

- **ReportInfo** - represents one report object.
- **ReportInfoProvider** - provides management functionality for reports.

Reporting_ReportGraph table API:

- **ReportGraphInfo** - represents one report graph.
- **ReportGraphInfoProvider** - provides management functionality for report graphs.

Reporting_ReportTable table API:

- **ReportTableInfo** - represents one report table.
- **ReportTableInfoProvider** - provides management functionality for report tables.

Reporting_ReportValue table API:

- **ReportValueInfo** - represents one report value object.
- **ReportValueInfoProvider** - provides management functionality for report values.

Reporting_SavedReport table API:

- **SavedReportInfo** - represents one saved report.
- **SavedReportInfoProvider** - provides management functionality for saved reports.

Reporting_SavedGraph table API:

- **SavedGraphInfo** - represents one saved graph.
- **SavedGraphInfoProvider** - provides management functionality for saved graphs.

Reporting_ReportSubscription table API:

- **ReportSubscriptionInfo** - represents a subscription to a report.
- **ReportSubscriptionInfoProvider** - provides management functionality for reporting subscriptions.

8.41.9.4 API examples

8.41.9.4.1 Overview

These topics show examples of how the API of the Groups module can be used:

- Managing report categories
- Managing reports
- Managing report graphs
- Managing report tables

- Managing report values
- Adding elements to the layout of a report

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Tools\Reporting\Default.aspx.cs***.

The reporting API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.GlobalHelper;
using CMS.CMSHelper;
using CMS.SiteProvider;
using CMS.Reporting;
```

8.41.9.4.2 Managing report categories

The following example creates a report category.

```
private void CreateReportCategory()
{
    // Create new report category object
    ReportCategoryInfo newCategory = new ReportCategoryInfo();

    // Set the properties
    newCategory.CategoryDisplayName = "My new category";
    newCategory.CategoryCodeName = "MyNewCategory";

    // Save the report category
    ReportCategoryInfoProvider.SetReportCategoryInfo(newCategory);
}
```

The following example gets and updates a report category.

```
private bool GetAndUpdateReportCategory()
{
    // Get the report category
    ReportCategoryInfo updateCategory =
ReportCategoryInfoProvider.GetReportCategoryInfo("MyNewCategory");
    if (updateCategory != null)
    {
```

```
        // Update the properties
        updateCategory.CategoryDisplayName =
updateCategory.CategoryDisplayName.ToLower();

        // Save the changes
        ReportCategoryInfoProvider.SetReportCategoryInfo(updateCategory);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates report categories.

```
private bool GetAndBulkUpdateReportCategories()
{
    // Prepare the parameters
    string where = "CategoryCodeName LIKE N'MyNewCategory%'";

    // Get the data
    DataSet categories = ReportCategoryInfoProvider.GetCategories(where, null);
    if (!DataHelper.DataSourceIsEmpty(categories))
    {
        // Loop through the individual items
        foreach (DataRow categoryDr in categories.Tables[0].Rows)
        {
            // Create object from DataRow
            ReportCategoryInfo modifyCategory = new ReportCategoryInfo
(categoryDr);

            // Update the properties
            modifyCategory.CategoryDisplayName =
modifyCategory.CategoryDisplayName.ToUpper();

            // Save the changes
            ReportCategoryInfoProvider.SetReportCategoryInfo(modifyCategory);
        }

        return true;
    }

    return false;
}
```

The following example deletes a report category.

```
private bool DeleteReportCategory()
{
    // Get the report category
    ReportCategoryInfo deleteCategory =
```

```
ReportCategoryInfoProvider.GetReportCategoryInfo("MyNewCategory");

    // Delete the report category
    ReportCategoryInfoProvider.DeleteReportCategoryInfo(deleteCategory);

    return (deleteCategory != null);
}
```

8.41.9.4.3 Managing reports

The following example creates a report.

```
private bool CreateReport()
{
    // Get the report category
    ReportCategoryInfo category = ReportCategoryInfoProvider.GetReportCategoryInfo
("MyNewCategory");
    if (category != null)
    {
        // Create new report object
        ReportInfo newReport = new ReportInfo();

        // Set the properties
        newReport.ReportDisplayName = "My new report";
        newReport.ReportName = "MyNewReport";
        newReport.ReportCategoryID = category.CategoryID;
        newReport.ReportAccess = ReportAccessEnum.All;
        newReport.ReportLayout = "";
        newReport.ReportParameters = "";

        // Save the report
        ReportInfoProvider.SetReportInfo(newReport);

        return true;
    }

    return false;
}
```

The following example gets and updates a report.

```
private bool GetAndUpdateReport()
{
    // Get the report
    ReportInfo updateReport = ReportInfoProvider.GetReportInfo("MyNewReport");
    if (updateReport != null)
    {
        // Update the properties
        updateReport.ReportDisplayName = updateReport.ReportDisplayName.ToLower();
```

```
        // Save the changes
        ReportInfoProvider.SetReportInfo(updateReport);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates reports.

```
private bool GetAndBulkUpdateReports()
{
    // Prepare the parameters
    string where = "ReportName LIKE N'MyNewReport%'";
    string orderby = "";
    int topN = 0;
    string columns = "";

    // Get the data
    DataSet reports = ReportInfoProvider.GetReports(where, orderby, topN,
columns);
    if (!DataHelper.DataSourceIsEmpty(reports))
    {
        // Loop through the individual items
        foreach (DataRow reportDr in reports.Tables[0].Rows)
        {
            // Create object from DataRow
            ReportInfo modifyReport = new ReportInfo(reportDr);

            // Update the properties
            modifyReport.ReportDisplayName =
modifyReport.ReportDisplayName.ToUpper();

            // Save the changes
            ReportInfoProvider.SetReportInfo(modifyReport);
        }

        return true;
    }

    return false;
}
```

The following example deletes a report.

```
private bool DeleteReport()
{
    // Get the report
    ReportInfo deleteReport = ReportInfoProvider.GetReportInfo("MyNewReport");
```

```
    // Delete the report
    ReportInfoProvider.DeleteReportInfo(deleteReport);

    return (deleteReport != null);
}
```

8.41.9.4.4 Managing report graphs

The following example creates a graph and assigns it to a report.

```
private bool CreateReportGraph()
{
    // Get report object by report code name
    ReportInfo report = ReportInfoProvider.GetReportInfo("MyNewReport");

    // If report exists
    if (report != null)
    {
        // Create new report graph object
        ReportGraphInfo newGraph = new ReportGraphInfo();

        // Set the properties
        newGraph.GraphDisplayName = "My new graph";
        newGraph.GraphName = "MyNewGraph";
        newGraph.GraphQuery = "SELECT TOP 10 DocumentName, DocumentID FROM
CMS_Document";
        newGraph.GraphReportID = report.ReportID;
        newGraph.GraphQueryIsStoredProcedure = false;
        newGraph.GraphType = "bar";

        // Save the report graph
        ReportGraphInfoProvider.SetReportGraphInfo(newGraph);

        return true;
    }

    return false;
}
```

The following example gets and updates a report graph.

```
private bool GetAndUpdateReportGraph()
{
    // Get the report graph
    ReportGraphInfo updateGraph = ReportGraphInfoProvider.GetReportGraphInfo
("MyNewGraph");
    if (updateGraph != null)
    {
        // Update the properties
        updateGraph.GraphDisplayName = updateGraph.GraphDisplayName.ToLower();
```

```
        // Save the changes
        ReportGraphInfoProvider.SetReportGraphInfo(updateGraph);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates report graphs.

```
private bool GetAndBulkUpdateReportGraphs()
{
    // Prepare the parameters
    string where = "GraphName LIKE N'MyNewGraph%'";

    // Get the data
    DataSet graphs = ReportGraphInfoProvider.GetGraphs(where, null);
    if (!DataHelper.DataSourceIsEmpty(graphs))
    {
        // Loop through the individual items
        foreach (DataRow graphDr in graphs.Tables[0].Rows)
        {
            // Create object from DataRow
            ReportGraphInfo modifyGraph = new ReportGraphInfo(graphDr);

            // Update the properties
            modifyGraph.GraphDisplayName = modifyGraph.GraphDisplayName.ToUpper();

            // Save the changes
            ReportGraphInfoProvider.SetReportGraphInfo(modifyGraph);
        }

        return true;

    }
    return false;
}
```

The following example deletes a report graph.

```
private bool DeleteReportGraph()
{
    // Get the report graph
    ReportGraphInfo deleteGraph = ReportGraphInfoProvider.GetReportGraphInfo
("MyNewGraph");

    // Delete the report graph
    ReportGraphInfoProvider.DeleteReportGraphInfo(deleteGraph);
```

```
    return (deleteGraph != null);
}
```

8.41.9.4.5 Managing report tables

The following example creates a table and assigns it to a report.

```
private bool CreateReportTable()
{
    // Get report object by report code name
    ReportInfo report = ReportInfoProvider.GetReportInfo("MyNewReport");

    // If report exists
    if (report != null)
    {
        // Create new report table object
        ReportTableInfo newTable = new ReportTableInfo();

        // Set the properties
        newTable.TableDisplayName = "My new table";
        newTable.TableName = "MyNewTable";
        newTable.TableQuery = "SELECT TOP 10 DocumentName, DocumentID FROM
CMS_Document";
        newTable.TableReportID = report.ReportID;
        newTable.TableQueryIsStoredProcedure = false;

        // Save the report table
        ReportTableInfoProvider.SetReportTableInfo(newTable);

        return true;
    }

    return false;
}
```

The following example gets and updates a report table.

```
private bool GetAndUpdateReportTable()
{
    // Get the report table
    ReportTableInfo updateTable = ReportTableInfoProvider.GetReportTableInfo
("MyNewTable");
    if (updateTable != null)
    {
        // Update the properties
        updateTable.TableDisplayName = updateTable.TableDisplayName.ToLower();

        // Save the changes
        ReportTableInfoProvider.SetReportTableInfo(updateTable);
```

```
        return true;
    }

    return false;
}
```

The following example gets and bulk updates report tables.

```
private bool GetAndBulkUpdateReportTables()
{
    // Prepare the parameters
    string where = "TableName LIKE N'MyNewTable%'";

    // Get the data
    DataSet tables = ReportTableInfoProvider.GetTables(where, null);
    if (!DataHelper.DataSourceIsEmpty(tables))
    {
        // Loop through the individual items
        foreach (DataRow tableDr in tables.Tables[0].Rows)
        {
            // Create object from DataRow
            ReportTableInfo modifyTable = new ReportTableInfo(tableDr);

            // Update the properties
            modifyTable.TableDisplayName = modifyTable.TableDisplayName.ToUpper();

            // Save the changes
            ReportTableInfoProvider.SetReportTableInfo(modifyTable);
        }

        return true;
    }

    return false;
}
```

The following example deletes a report table.

```
private bool DeleteReportTable()
{
    // Get the report table
    ReportTableInfo deleteTable = ReportTableInfoProvider.GetReportTableInfo
("MyNewTable");

    // Delete the report table
    ReportTableInfoProvider.DeleteReportTableInfo(deleteTable);

    return (deleteTable != null);
}
```

8.41.9.4.6 Managing report values

The following example creates a value and assigns it to a report.

```
private bool CreateReportValue()
{
    // Get report object by report code name
    ReportInfo report = ReportInfoProvider.GetReportInfo("MyNewReport");

    // If report exists
    if (report != null)
    {
        // Create new report value object
        ReportValueInfo newValue = new ReportValueInfo();

        // Set the properties
        newValue.ValueDisplayName = "My new value";
        newValue.ValueName = "MyNewValue";
        newValue.ValueQuery = "SELECT COUNT(DocumentName) FROM CMS_Document";
        newValue.ValueQueryIsStoredProcedure = false;
        newValue.ValueReportID = report.ReportID;

        // Save the report value
        ReportValueInfoProvider.SetReportValueInfo(newValue);

        return true;
    }

    return false;
}
```

The following example gets and updates a report value.

```
private bool GetAndUpdateReportValue()
{
    // Get the report value
    ReportValueInfo updateValue = ReportValueInfoProvider.GetReportValueInfo
("MyNewValue");
    if (updateValue != null)
    {
        // Update the properties
        updateValue.ValueDisplayName = updateValue.ValueDisplayName.ToLower();

        // Save the changes
        ReportValueInfoProvider.SetReportValueInfo(updateValue);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates report values.

```
private bool GetAndBulkUpdateReportValues()
{
    // Prepare the parameters
    string where = "ValueName LIKE N'MyNewValue%'";

    // Get the data
    DataSet values = ReportValueInfoProvider.GetValues(where, null);
    if (!DataHelper.DataSourceIsEmpty(values))
    {
        // Loop through the individual items
        foreach (DataRow valueDr in values.Tables[0].Rows)
        {
            // Create object from DataRow
            ReportValueInfo modifyValue = new ReportValueInfo(valueDr);

            // Update the properties
            modifyValue.ValueDisplayName = modifyValue.ValueDisplayName.ToUpper();

            // Save the changes
            ReportValueInfoProvider.SetReportValueInfo(modifyValue);
        }

        return true;
    }

    return false;
}
```

The following example deletes a report value.

```
private bool DeleteReportValue()
{
    // Get the report value
    ReportValueInfo deleteValue = ReportValueInfoProvider.GetReportValueInfo
("MyNewValue");

    // Delete the report value
    ReportValueInfoProvider.DeleteReportValueInfo(deleteValue);

    return (deleteValue != null);
}
```

8.41.9.4.7 Adding elements to the layout of a report

The following example inserts elements into the layout of a report.

```
private bool InsertElementsToLayout()
{
    // Get report object by report code name
```

```
    ReportInfo report = ReportInfoProvider.GetReportInfo("MyNewReport");

    // If report exists
    if (report != null)
    {
        ReportGraphInfo graph = ReportGraphInfoProvider.GetReportGraphInfo
("MyNewGraph");
        if (graph != null)
        {
            report.ReportLayout += "<br/>%%control:Report" +
ReportItemType.Graph + "?" + report.ReportName + "." + graph.GraphName + "%%<br/
>";
        }

        ReportTableInfo table = ReportTableInfoProvider.GetReportTableInfo
("MyNewTable");
        if (table != null)
        {
            report.ReportLayout += "<br/>%%control:Report" +
ReportItemType.Table + "?" + report.ReportName + "." + table.TableName + "%%<br/
>";
        }

        ReportValueInfo value = ReportValueInfoProvider.GetReportValueInfo
("MyNewValue");
        if (value != null)
        {
            report.ReportLayout += "<br/>%%control:Report" +
ReportItemType.Value + "?" + report.ReportName + "." + value.ValueName + "%%<br/
>";
        }

        ReportInfoProvider.SetReportInfo(report);

        return true;
    }

    return false;
}
```

8.42 SharePoint integration

8.42.1 Overview

Kentico CMS allows you to access data stored on a SharePoint (Windows SharePoint Services 3.0 (WSS) or MOSS - Microsoft Office SharePoint Server) server and display them on your site. This can be done using the web parts of the SharePoint module.

The Usage examples sub-chapter contains sample tutorials on using the SharePoint web parts for various tasks.

Default logon settings, used to access a SharePoint server, can be configured as described in the Settings topic.

What it can do

- Get list and list item data from SharePoint and provide it to Kentico CMS.
- Download documents or images.

What it can't do

- Modify SharePoint data.
- Display Kentico CMS data in SharePoint.

How it works

SharePoint web parts use pre-generated proxy classes of selected SharePoint web services to get data from the server. The following services are used:

- *http://server/_vti_bin/Lists.asmx* - methods for working with lists
- *http://server/_vti_bin/Imaging.asmx* - methods for working with picture libraries
- *http://server/_vti_bin/Copy.asmx* - methods for retrieving file content

The returned data is in CAML (XML) format, which needs to be further processed to display the data in a meaningful way. XSLT or ASCX transformations may be used for this purpose. For ASCX, the CAML response must be transformed to an ASP Dataset.

8.42.2 Web parts

Web parts for connecting to SharePoint are stored in the **Microsoft SharePoint** category in the web part catalog. This includes the following four web parts:

- **SharePoint datasource**
- **SharePoint datagrid**
- **SharePoint datalist**
- **SharePoint repeater**

The following text contains information about the specific properties of these web parts.

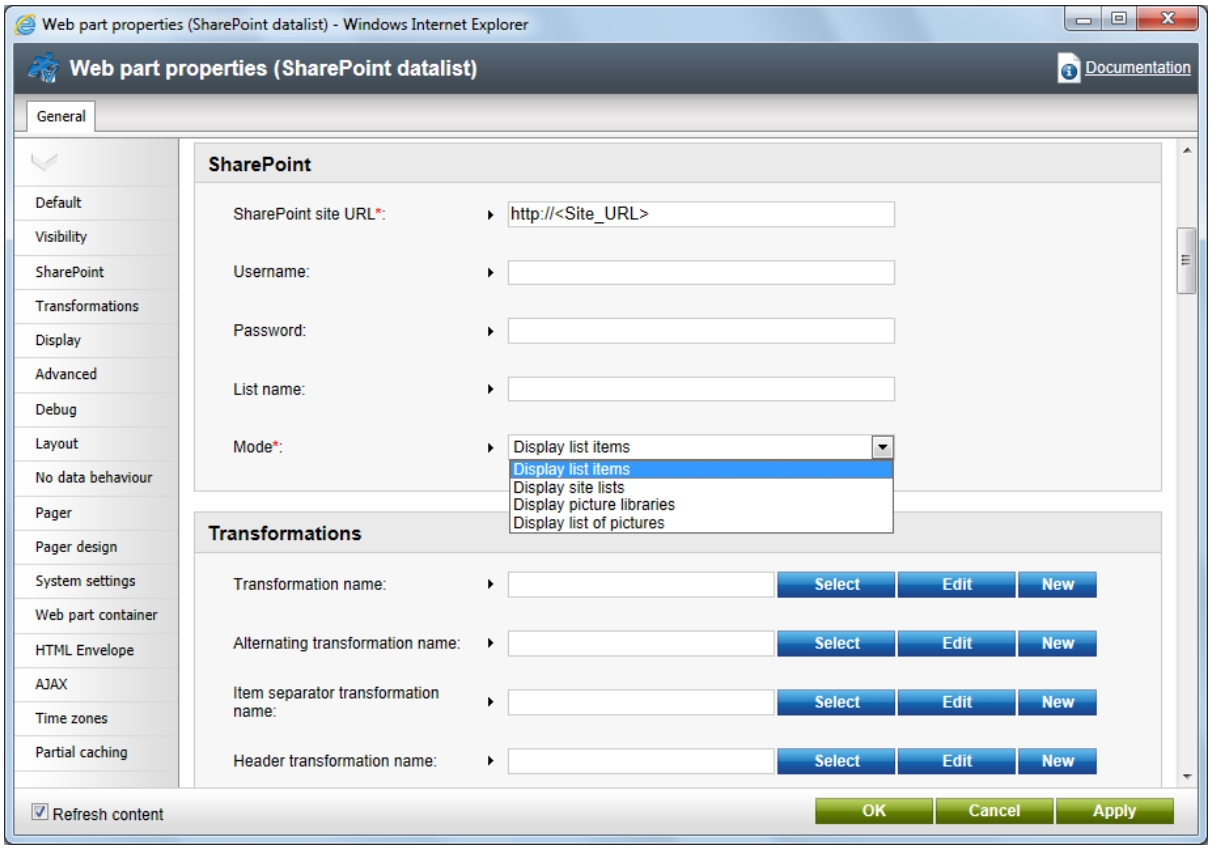
SharePoint

The following properties can be configured in the **SharePoint** section. All the web parts listed above have these settings in common.

- **Mode** - determines what the web part will do (what data will be retrieved from the SharePoint server) and therefore which web service should be used.
- **SharePoint site URL** - specifies the URL of the SharePoint site that should be used. Based on the value of the *Mode* property, the full URL of the required web service is determined by this option (it is not configured directly).
- **Username, Password** - you can specify the username and password for authentication directly in the web part properties, or you can inherit global values from *Site Manager -> Settings -> Integration -> Microsoft SharePoint.* If you want to use the credentials from the Site Manager settings, leave the fields empty.

- **List name** - if you use the *Display list items* or *Display list of pictures* options, you must specify the list from which the items should be loaded. In the other two modes, you can leave it empty. Please note: this field is case sensitive.

Just these settings should make the **SharePoint datagrid** display data or the **SharePoint datasource** to provide them. The other two web parts still need to have a transformation specified.



Transformations

The **SharePoint datalist** and **SharePoint repeater** need to have a transformation specified in order to display the retrieved data.

There is a number of pre-defined transformations which you can use almost right-off and of course modify them according to your needs. The transformations are stored under the **SharePoint - Transformations** document type (can be edited in **Site Manager -> Development -> Document types**). Their names are self-explaining - they correspond to what they should be used for.

The only thing you need to do to get them to work is edit () them and **replace the sample server or field names with the real names** used on the SharePoint server.

Advanced

In the **Advanced** section, you can specify (limit) in more detail what you want to retrieve. These settings are applied only in the “Get list items” mode.

- **Row limit** - sets the maximum number of retrieved items.
- **Query** - filters or sorts the returned items. Its purpose is similar to that of a WHERE condition in SQL, but it has completely different syntax based on CAML. For more information on how to construct such queries, search SharePoint documentation or use Google with keywords like "SharePoint query".
- **View fields** - directly specifies which fields (columns) of a SharePoint list should be retrieved.

Display

In this section, you can control how the retrieved data will be processed and displayed.

- **Selected item querystring key name** - if entered, items will be selected based on the presence of this key in the URL query string.
- **Selected item field name** - allows you to specify the field that is used to select items. The default approach is to select items by ID — the web part tries to get the query string parameter specified in the *Selected item querystring key* property and uses its value, e.g. If you entered id into the property above, *aspx?id=2* will select the item with *2* in its ID field. Note: the original field name is used here — without the *ows_* prefix. This field is case sensitive.
- **Selected item field type** - specifies the type of the *ItemID* field. For IDs, you should use the *Counter* type, for strings, use the *Text* type. These two should be sufficient for most cases, but you can search SharePoint documentation for other types.
- **Use dataset** - if enabled, the retrieved XML data will be converted to a dataset that can be used with standard ASCX transformations. If disabled, the data will not be modified and can be converted directly to HTML using XSLT.
- **Dataset fields** - if you want to use the dataset format, but you don't need all fields, you can specify which fields should be included in the dataset by entering their names separated by semicolons. This property can be particularly useful if you want to use caching.

Please note: Properties in the **Display** section are not available for the **SharePoint datagrid** web part.

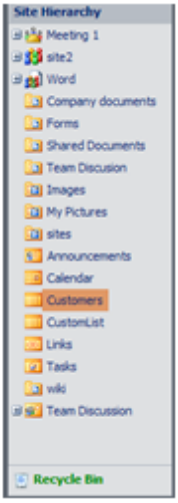
Debug

Because it is sometimes hard to tell or remember how SharePoint addresses list fields, we implemented the **Show raw response** function. It displays all retrieved SharePoint data in its raw XML form. You can look for fields which contain valuable information and can be displayed. Note: if the data is fetched from the cache, the response can't be printed.

8.42.3 Usage examples

8.42.3.1 Site hierarchy

In this example, you will see how to display site lists which resemble the site hierarchy panel in the SharePoint site. It is quite simple. In fact, it only requires setting the *Display site lists* mode.



This can be done by adding one of the SharePoint web parts (e.g. **Microsoft SharePoint -> SharePoint repeater**) and configuring them the following way:

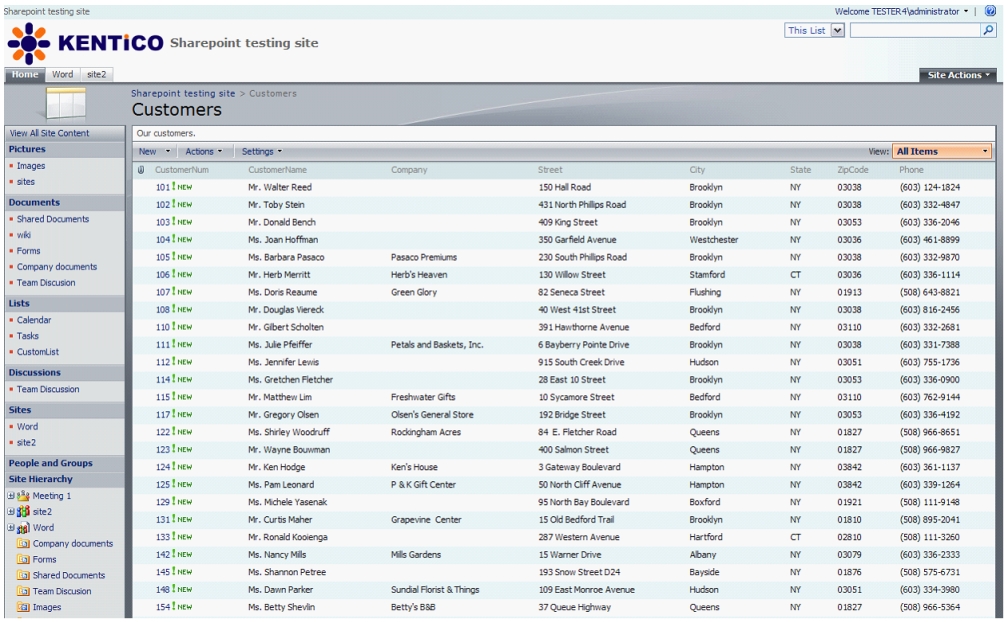
- **SharePoint site URL**: URL of your SharePoint server.
- **Mode**: Display site lists.
- **Username, password**: enter your username and password or leave the fields empty if you have them configured in *Site manager -> Settings -> Integration -> Microsoft SharePoint*.
- **Transformation**: there is a prepared transformation for displaying a site hierarchy called *SiteHierarchy* under the *SharePoint – Transformations* document type. For correct behavior, you must only change the name of the SharePoint server.

Leave default values for the rest of the properties and click **OK**. You should see a result similar to the screenshot below - it simply creates a link to each of the lists on the SharePoint server.

Announcements(2)
Calendar(3)
Company documents(7)
Customers(51)
CustomList(4)
Empty(0)
EmptyList(0)
Forms(0)
Images(5)
Kentico Annual Reports(4)
|< < **1** 2 > >|

8.42.3.2 List items

The *Display list items* mode enables you to display items from any list from the SharePoint server.



Let's say that in SharePoint, there is a custom list called *Customers*, storing information about company customers in fields like *Customer name*, *Street*, *City*, *Phone*, etc. Let's say that we want to display data from this list on our Kentico CMS website.

1. This can be done by adding one of the SharePoint web parts (e.g. **Microsoft SharePoint -> SharePoint repeater**) and configuring them the following way:

- **SharePoint site URL**: URL of your SharePoint server.
- **Mode**: Display list items.
- **Username, password**: enter your username and password or leave the fields empty if you have it configured in *Site manager -> Settings -> Integration -> Microsoft SharePoint.*
- **Transformation**: there is a prepared transformation for item lists called **ListItems** under the **SharePoint – Transformations** document type; for correct behavior, you must only change the name of the SharePoint server.
- **Show raw response**: enabled; this is because we will need to inspect the retrieved data and change the transformation to suit our needs.

Leave the default values for the rest of the properties and click **OK**. You should see a result similar to the screenshot below.

```
(2009-06-26 13:24:54)
(2009-06-26 13:24:54)
(2009-06-26 13:24:54)
(2009-06-26 13:24:54)
(2009-06-26 13:24:54)
<rs:data ItemCount="49" xmlns:rs="urn:schemas-microsoft-com:rowset">
        <z:row ows_Attachments="0" ows_LinkTitle="101" ows_CustomerName="Mr. Walter Reed" ows_Street="150 Hall Road" ows_City="Brooklyn" ows_State="NY" ow
        <z:row ows_Attachments="0" ows_LinkTitle="102" ows_CustomerName="Mr. Toby Stein" ows_Street="431 North Phillips Road" ows_City="Brooklyn" ows_Stat
        <z:row ows_Attachments="0" ows_LinkTitle="103" ows_CustomerName="Mr. Donald Bench" ows_Street="409 King Street" ows_City="Brooklyn" ows_State="NY"
        <z:row ows_Attachments="0" ows_LinkTitle="104" ows_CustomerName="Ms. Joan Hoffman" ows_Street="350 Garfield Avenue" ows_City="Westchester" ows_Sta
        <z:row ows_Attachments="0" ows_LinkTitle="105" ows_CustomerName="Ms. Barbara Pasaco" ows_Company="Pasaco Premiums" ows_Street="230 South Phillips
        <z:row ows_Attachments="0" ows_LinkTitle="106" ows_CustomerName="Mr. Herb Merritt" ows_Company="Herb's Heaven" ows_Street="130 Willow Street" ows_
        <z:row ows_Attachments="0" ows_LinkTitle="107" ows_CustomerName="Ms. Doris Reaume" ows_Company="Green Glory" ows_Street="82 Seneca Street" ows_Cit
```

2. That's not exactly what we want. We would like to see the customers' name and city. We must look into the raw output and search for those fields. We find them there under the *ows_CustomerName* and *ows_Street* attributes.

Now that we know the names of the attributes, we need to edit the transformation code or create a new

transformation.

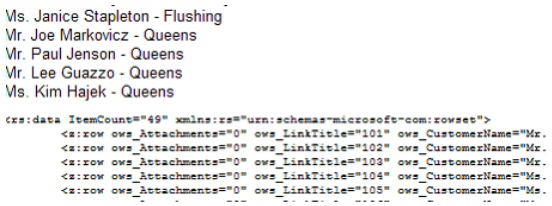
Replace the original **Transformation**:

```
<%# Eval("ows_Title") %> (<%#  Eval("ows_Created")%>) <br />
```

With something like this:

```
<%# Eval("ows_CustomerName") %> - <%#  Eval("ows_City")%> <br />
```

The output looks better now, it displays the desired values.



3. Now we would like to be able to click each item and see more detail about it. We must prepare the **Selected item transformation**. The transformation may look like the following code sample:

```
<strong><%# Eval("ows_CustomerName") %></strong><br/>
<%# Eval("ows_City")%> <br/>
<%# Eval("ows_Street")%> <br/>
<%# Eval("ows_Phone")%> <br/>
```

4. In web part properties, we must enter the field name of the field by which we want to select items. In this case, it is the *ID* field. In raw response output, it can be seen as *ows_ID*. So enter the following values:

- **Selected item querystring key name**: id
- **Selected item field name**: ID
- **Selected item field type**: Counter

Finally we must alter the original **Transformation** to suit the values we just entered, i.e. to create a link to customer details using the *id* parameter in query string. It can look like this.

```
<a href="<%# CMS.GlobalHelper.URLHelper.AddParameterToUrl(CMSContext.RawUrl,"id",
(string)Eval("ows_ID")) %>">
<%# Eval("ows_CustomerName") %> - <%# Eval("ows_City")%> </a><br />
```

The final result looks like this - a simple text list of customers with links:

Mr. Gregory Olsen - Brooklyn
Ms. Shirley Woodruff - Queens
Mr. Wayne Bouwman - Queens
Mr. Ken Hodge - Hampton
Ms. Pam Leonard - Hampton
Ms. Michele Yasenak - Boxford

After clicking a customer name, a detail of the customer is displayed.

Mr. Wayne Bouwman
Queens
400 Salmon Street
(508) 966-9827

```
<rs:data ItemCount="1" xmlns:rs="urn:schemas-microsoft-com:rowset">
        <z:row ows_Attachments="0" ows_LinkTitle="123" ows_CustomerName="Mr. Wayne Bouwman" ows_Street="400 Salmon Street"
</rs:data>
```

5. Now let's configure the **Advanced** settings. We will want a maximum of 10 customers to be displayed and we want to display only customers from *Queens*.

Enter number 10 into the **Row limit** field and use this code as the **Query**:

```
<Query>
<Where><Eq><FieldRef Name="City" /><Value Type="Text">Queens</Value></Eq></Where>
</Query>
```

The <*Eq*> tag means equals; field name is *City*, and its value of type *Text* should be equal to *Queens*.

Click **OK**, you should see the filtered output now as in the following screenshot:

Ms. Shirley Woodruff - Queens
Mr. Wayne Bouwman - Queens
Ms. Betty Shevlin - Queens
Ms. Deborah Wolfe - Queens
Ms. Kelly Smith - Queens
Mr. Paul Griffin - Queens
Mr. Joe Markovicz - Queens
Mr. Paul Jenson - Queens
Mr. Lee Guazzo - Queens
Ms. Kim Hajek - Queens

```
<rs:data ItemCount="10" xmlns:rs="urn:schemas-microsof
        <z:row ows_Attachments="0" ows_LinkTitle="122"
        <z:row ows_Attachments="0" ows_LinkTitle="123"
```

8.42.3.3 Picture libraries

In this example, we use SharePoint to retrieve only a picture library type list. For setup, you must only set the following properties:

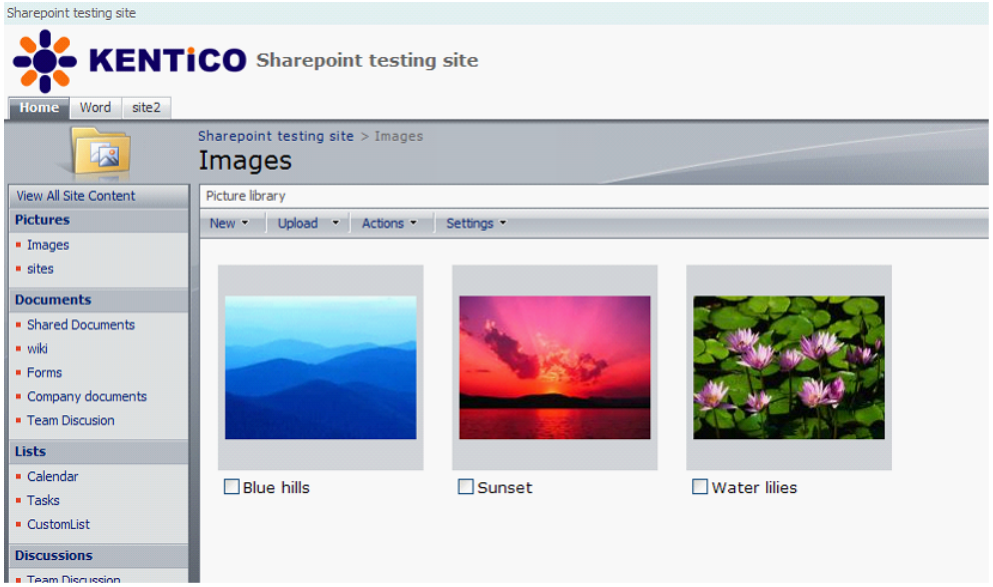
- **SharePoint site URL**: URL of your SharePoint server.
- **Mode**: Display picture libraries.
- **Username, password**: enter your username and password or leave the fields empty if you have it configured in *Site manager -> Settings -> Integration -> Microsoft SharePoint*.
- **Transformation**: there is a prepared transformation for picture libraries called **PictureLibLists** under the **SharePoint – Transformations** document type. For correct behavior, you must only change the name of the SharePoint server.

Leave default values for the rest of the properties and click **OK**. You should see a result similar to the screenshot below - it shows a list of picture libraries with links to the SharePoint server.

Empty
Images
My Pictures
sites

8.42.3.4 List of pictures

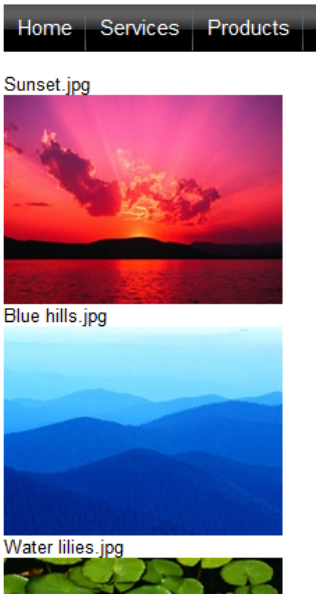
The *Display list of pictures* mode enables you to display pictures from picture libraries on a SharePoint server.



On the SharePoint server, we have a picture library named *Images* and 3 pictures in it. We want to display these pictures in our Kentico CMS site. For setup, you must enter the following properties of the web part (e.g. SharePoint repeater):

- **SharePoint site URL**: URL of your SharePoint server.
- **Mode**: Display list of pictures.
- **Username, password**: enter your username and password or leave the fields empty if you have it configured in *Site manager -> Settings -> Integration -> Microsoft SharePoint*.
- **List name***:* Images
- **Transformation**: there is a prepared transformation for picture lists called *Pictures* under the *SharePoint – Transformations* document type. For correct behavior, you must only change the name of the SharePoint server.

Leave default values for the rest of the properties and click **OK**. You should see a result similar to the screenshot below.



The transformation code looks like this:

```
<%# SharePointFunctions.SplitSharePointField((string)Eval("ows_FileLeafRef"),1) %
><br/>
<img src="<%# SharePointFunctions.GetSharePointFileUrl("tester4",
SharePointFunctions.SplitSharePointField((string)Eval("ows_FileRef"),1)) %
>&maxsidesize=200" /> <br />
```

The images are downloaded through the **GetSharePointFile.aspx** page, same as documents. So we create *<img>* tags with *src* attributes pointing to this page in the transformation. For ease of use, there is the **GetSharePointFileUrl** function in the **SharePointFunctions** class, which takes two parameters and returns the URL which downloads the file. The first parameter is server name (or site url), the second parameter is the location of the file in SharePoint. The final URL looks like this:

- *~/ CMSModules/Sharepoint/CMSPages/GetSharePointFile.aspx?server=server_name&name=Images/Sunset.jpg*

Please note: Credentials from settings are always used for downloading the file when the **GetSharePointFile** page is used.

There is another useful function in the **SharePointFunctions** class called **SplitSharePointField**. SharePoint for some reason creates combined attribute values for some fields. It looks like this - *ows_FileLeafRef="2;#Blue hills.jpg"*. We would often like to use only a part of such a value. The **SplitSharePointField** function serves this purpose. The first argument is a combined value and the second one is the index of the part we want in the output.

8.42.3.5 GetSharePointFile page

The **GetSharePointFile.aspx** page is a special page located under the **~/CMSModules/Sharepoint/CMSPages** directory. The correct URL to access the page looks like the following:

- *~/ CMSModules/Sharepoint/CMSPages/GetSharePointFile.aspx?*

server=server_name&name=Images/Sunset.jpg

It downloads the specified file from the SharePoint server and sends it further to the user. Credentials configured in **Site Manager -> Settings -> Integration -> Microsoft SharePoint** are used for authentication by the web service. You can control the Content-Disposition HTTP header using the *disposition* query parameter.

For images, it supports *maxsidesize*, *width* and *height* parameters to control the image size. Kentico CMS cache is used for images, you can configure it in **Site Manager -> Settings -> System -> Performance -> Cache images (minutes)**.

You can also specify servers, from which files can be retrieved. Enter the servers, delimited by semicolons, into the setting **Site Manager -> Settings -> Integration -> Microsoft SharePoint -> Allowed servers**.

Important!

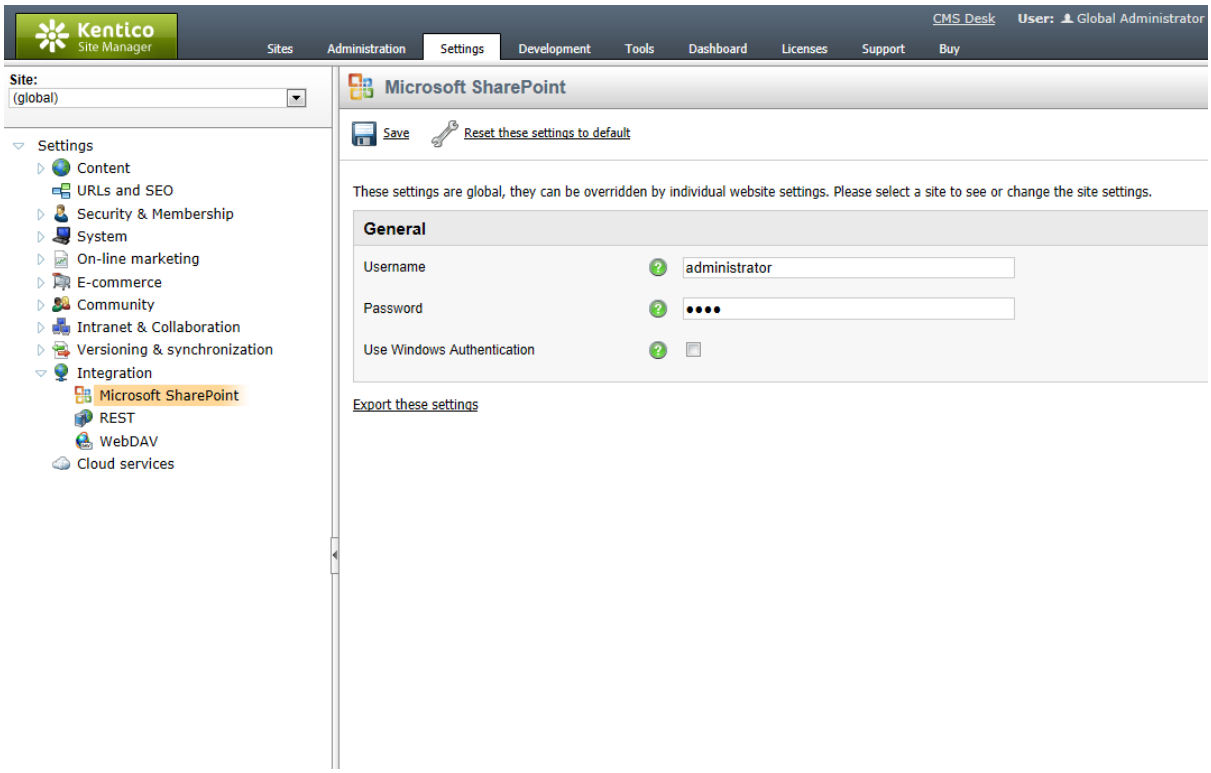
Because the settings configured in **Site Manager -> Settings -> Integration -> Microsoft SharePoint** are automatically used for authentication by the **GetSharePointFile.aspx** page and since the URL to access the page can be entered manually, it is highly recommended to enter the credentials of a user that is authorized to access only the files you want to display on your website.

8.42.4 Settings

Default logon credentials used to access the SharePoint server can be configured in **Site Manager -> Settings -> Integration -> Microsoft SharePoint**. If you configure a **Username** and **Password** here, it will be used by default if you leave the *Username* and *Password* fields **blank** in the properties of SharePoint web parts.

If you enable the **Use Windows Authentication** option, the current user's windows domain credentials will be used to access the SharePoint server.

The address of the SharePoint server itself needs to be specified in the properties of SharePoint web parts.



Important!

The settings configured here are automatically used for authentication by the GetSharePointFile.aspx page and since the URL to access the page can be entered manually, it is highly recommended to enter the credentials of a user that is authorized to access only the files you want to display on your website.

8.43 Smart search

8.43.1 Overview

The Smart search module allows index-based searching through the content of websites and various types of data within the system. It is based on **Lucene.Net** (version 2.1.0) which is a source code, class-per-class, API-per-API port of the Java **Lucene** search engine to the C# and .NET platform.

The module uses **indexes** to store information about the website content. When a users sends a search request, the system searches through the appropriate indexes, which results in **significantly better performance** compared to linear SQL query search. For more information on **how the module works**, please refer to the How it works topic.

To set up the smart search functionality on your website, you need to perform the following three steps:

1. Enable Smart search indexing in the system.

2. Create search indexes. Assign the indexes to your site (and culture) and define their exact content.

See the Managing indexes sub-chapter for more details.

3. Add Smart search web parts to the pages of your site. You can find an overview of these web parts, including explanations of the most important web part properties, in the Available web parts and Using the Smart search filter topics.

The following topics provide additional information about search related issues:

- Search syntax
- Related scheduled tasks
- Searching attachments
- Search results in transformations
- Security

In older versions, Kentico CMS supported only linear SQL search functionality. To keep the system backward compatible, this functionality is still available. It is now referred to as **SQL Search** and you can find further information about it in the SQL Search overview topic.

The Smart search internals and API sub-chapter provides information about the database tables and classes used by the module and examples of how smart search indexes can be managed using the API and how search results can be displayed in transformations.

8.43.2 How it works

Information about the content specified for smart search indexes is stored in physical **index files** on the local disk. The index files are located in the **~/App_Data/CMSModules/SmartSearch/<Index code name>** folder within your web project directory.

Documents, forums and other objects in Kentico CMS are reflected in the index file as **index documents**. The data structure of the index documents is much more suitable for being searched through, resulting in significantly higher search performance compared to linear SQL search.

The index documents contain the same fields as the corresponding Kentico CMS objects, based on the search settings of individual object types. Depending on these settings, the **Index writer** creates representations of objects in the index files. When an object included in the index is created, removed or has one of its fields modified, the system automatically schedules an **Indexing task**, which update the corresponding index document. The **Index searcher** searches through the index file and returns the relevant results.

Example

The following model scenario explains the life cycle of a document in a search index file:

1. A user creates a new document.
2. Upon the document's creation, the system logs a new indexing task in the database.
3. The Smart search either runs the indexing task immediately or processes it later using a scheduled task.
4. When executed, the indexing task adds the new document to the appropriate search indexes. The system indexes the document's content based on the search field settings defined for the given

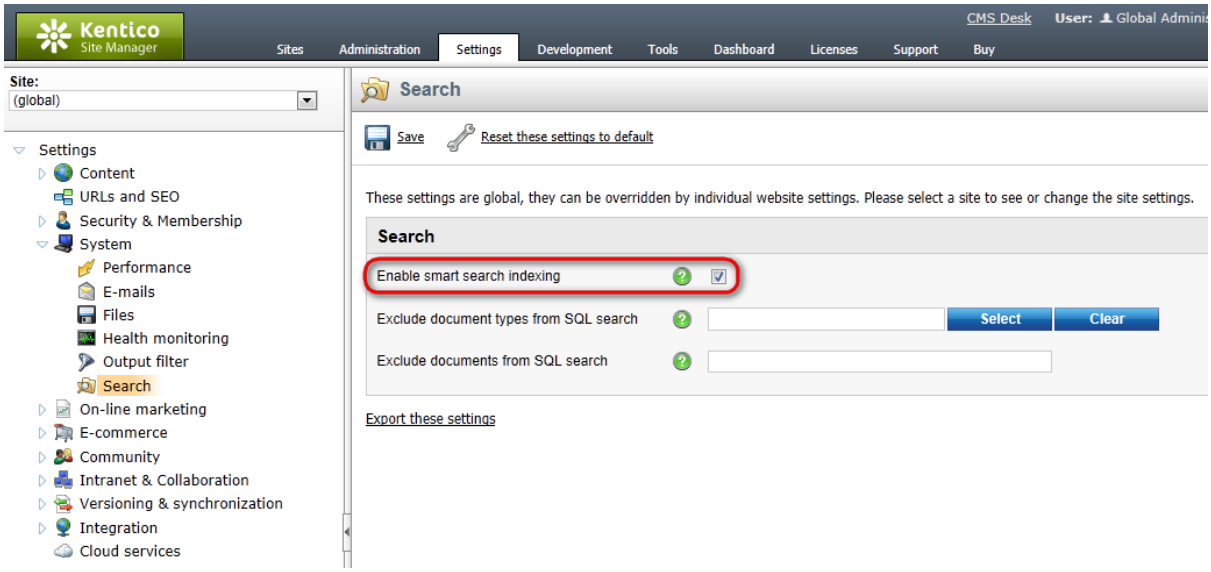
document type.

5. A user arrives on the website and sends a search request via a Smart search web part.
6. The web part searches through the assigned indexes and returns results based on the found data.

8.43.3 Enabling Smart search indexing

To enable Smart search indexing for all websites in the system:

1. Go to **Site Manager -> Settings -> System -> Search**.
2. Check **Enable smart search indexing** box.
3. Click **Save**.



Note: Your application must have the **write permission** for the **~/App_Data** folder on the server's file system. This folder stores the search index files, so the system cannot create and update indexes without the required permissions. See the Disk permission problems chapter to learn how to grant the write permission for the folder.

Scheduling the search indexing process

By default, the smart search creates and executes indexing tasks immediately whenever content covered by a search index is created or modified.

You can disable automatic running of indexing tasks upon creation by adding the **CMSProcessSearchTasksByScheduler** key to the */configuration/appSettings* section of your application's web.config file:

```
<add key="CMSProcessSearchTasksByScheduler" value="true" />
```

When you set this key to *true*:

- The system only logs the search indexing tasks into the database without running them. You need to process the tasks periodically, for example using the **Execute search tasks** scheduled task.
- You cannot manually **Rebuild** indexes unless you also run the process that executes the indexing tasks.

8.43.4 Managing indexes

8.43.4.1 Creating an index

Before you can use the Smart search module, you need to prepare the appropriate indexes for storing information about the website's content in an efficiently searchable format.

The following types of search indexes are available:

- Documents - stores information about the content of documents in the content tree.
- Documents crawler - directly indexes the HTML output of documents (pages).
- Forums - stores information about the content of discussion forums in the system.
- Custom tables - indexes records stored in custom tables.
- Users - stores information about users in the system.
- General - stores information about system objects of a specified type.
- Custom - allows you to use your own custom-coded search index. Stores any kind of data depending on the implementation.

The following steps describe how to create smart search indexes for your website:

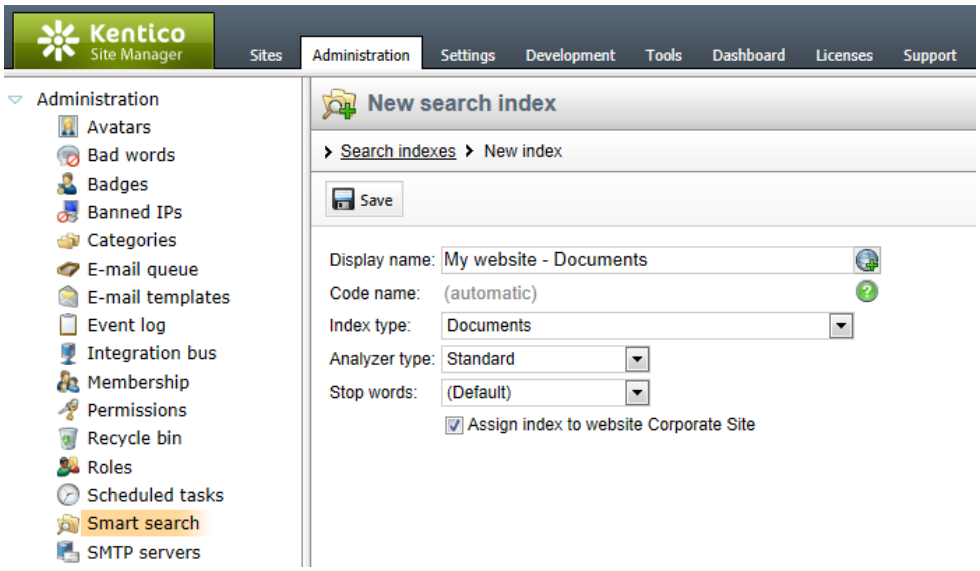
1. Go to **Site Manager -> Administration -> Smart search**.

2. Click **New index**. The **New search index** dialog opens.

3. Fill in the following details for the index:

Display name	Name of the index displayed in the administration interface.
Code name	Name of the index used as a unique identifier, typically in web part properties or in the API. You can leave the default *(automatic)* option to have the system generate a code name based on the display name. **Warning**: This name is also used for the physical index file. The fully qualified name of the file must be less than 260 characters long, including the directory path.
Index type	Determines what type of content is stored in the index. The following index types are available: • **Custom index** - indexes any kind of data depending on the implementation. • **Custom tables** - indexes records in custom tables. • **Documents** - indexes the content of documents in the content tree. • **Documents crawler** - indexes the HTML output of the website's documents (pages). • **Forums** - indexes the content of discussion forums. • **General** - indexes system objects of a specified type. General indexes allow you to search through any objects within the CMS. • **Users** - indexes details about users in the system (fields of the

	CMS_User system table).
Analyzer type	Sets the type of text analyzer that the index uses to process (tokenize) content. The following analyzer types are available: • **Custom** - allows you to assign a custom-written analyzer. This provides a way to perform text tokenization according to your own specific requirements. If selected, you need to specify the names of the assembly and class where the custom analyzer is implemented. See Using custom analyzers for more information. • **Keyword** - returns the entire text stream as a single token. This is useful for structured data fields like zip codes or IDs. • **Simple** - divides text at non-letter characters. • **Standard** - grammar-based analyzer (stop words, shortcuts, ...). This option is very efficient for English, but may not produce satisfactory results with other languages. • **Starts with** - tokenizes all prefixes contained in words, which allows searching for words that start with the search keyword. Divides text at whitespace characters. For example, searching for *test* returns words such as *test*, *tests*, *tester*, etc. • **Stop** - uses a predefined collection of stop words to divide text. • **Subset** - tokenizes all possible substrings in words. Divides text at whitespace characters. Indexes with this analyzer type return results for all words that contain the search keyword. For example, searching for *net* returns words such as *net*, *Internet*, *network*, etc. • **White space** - divides text at whitespace characters.
Stop words	Selects the stop word dictionary for *Stop* or *Standard* analyzers. Stop words (e.g., 'and', 'or') are excluded from the index content and the analyzer uses them to divide text into tokens. You can edit the content of the dictionaries or add new ones. The application stores the dictionaries as text files in the *~\App_Data\CMSModules\SmartSearch_StopWords* folder.
Assign index to website <sitename>	Check this box to automatically assign the new index to the currently active site.



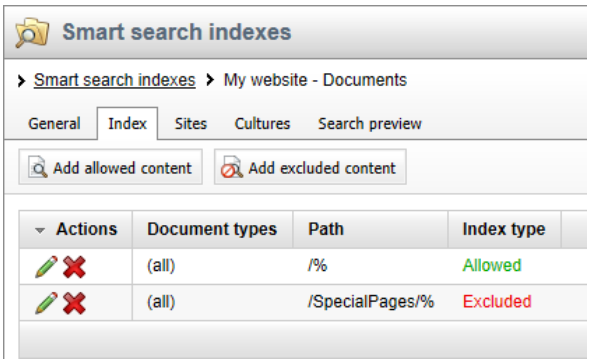
Creating a new document index

4. Click **Save** to create the search index. The **General** tab of the index's editing interface opens, where you can edit the same properties that you configured when creating the index.

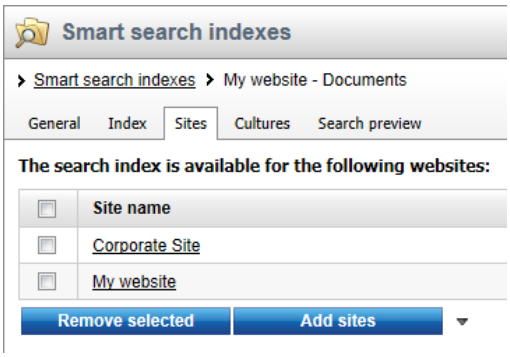
Additionally, you can set the **Batch size** property for the index, which sets the maximum amount of records that the system retrieves in a single database query when rebuilding (or creating) the index. This allows you to optimize indexing performance. The default value is 10. Increasing the value reduces the amount of queries required for large numbers of records, which may improve performance, but also increases memory consumption. The optimal value depends on the type (size) of the indexed objects and on the resources available in your hosting environment. When indexing large objects (e.g. documents), it is recommended to set a reasonably small batch size.

5. Switch to the **Index** tab and define which documents, forums, custom tables, users or other objects should be included in the index. The options available on the Index tab depend on the type of the index. You can find detailed information about setting the content of individual index types in the following topics:

- Defining document index content
- Defining forums index content
- Defining custom tables index content
- Defining user index content
- Defining general index content
- Defining custom index content

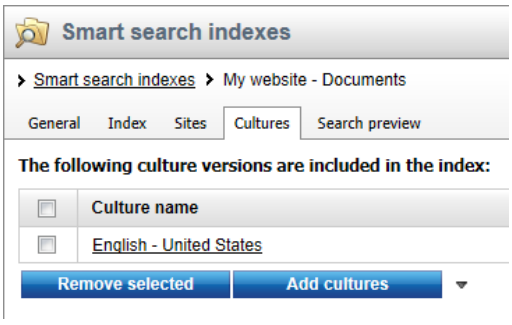


6. Open the **Sites** tab and assign the index to the websites where you wish to use it. You can implement multi-site search functionality by assigning the index to more than one website.

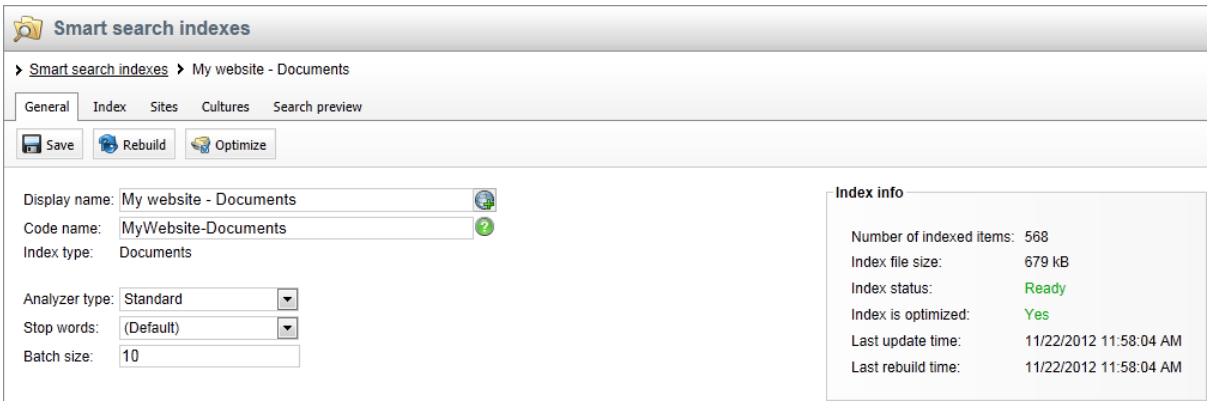


Note: If the index includes global objects that are not site-specific, the selection made on the Sites tab does not affect the index's content. However, the index is only available for use (through Smart search web parts) on the assigned sites.

7. If you are creating a *Documents* or *Documents crawler* type index, switch to the **Cultures** tab. Here you need to select which language versions of the website's documents should be indexed.
   - You must assign at least one culture in order for the index to be functional.
   - If you have a multi-site index, you can select the cultures separately for each site.



8. Go back to the **General** tab and **Rebuild** the index.
   - The **Index info** box on the right side of the General tab displays current information about the status and properties of the index.



Once the system finishes building the index, you can start using it on your website. The **Search preview** tab allows you to test the functionality of the index.

Maintaining search indexes

You can manage existing search indexes using the actions available on the **General** tab of the index editing interface.

The system automatically updates search indexes to reflect all changes made to the indexed content. Over time, these updates can make indexes less efficient, particularly in the case of large indexes.

To restore optimal search performance for an index, defragment it by clicking the **Optimize** action. You can enable the **Optimize search indexes** scheduled task to have the system automatically optimize all smart search indexes once per week.

The **Rebuild** action deletes the current index file and indexes all specified content again.

- Use the rebuild action to apply changes made to the index's configuration. This includes modifications of the analyzer settings (*Analyzer type*, *Stop words*), all options on the *Index*, *Sites* or *Cultures* tabs, and adjustments of the search field settings for the indexed objects.
- The system automatically optimizes the index after a successful rebuild.

Note

Clicking the **Rebuild** action does not always guarantee that the index starts rebuilding immediately. The process may be delayed if another index is already being rebuilt or if the rebuilding tasks are configured to be handled by the scheduler.

8.43.4.2 Defining document index content

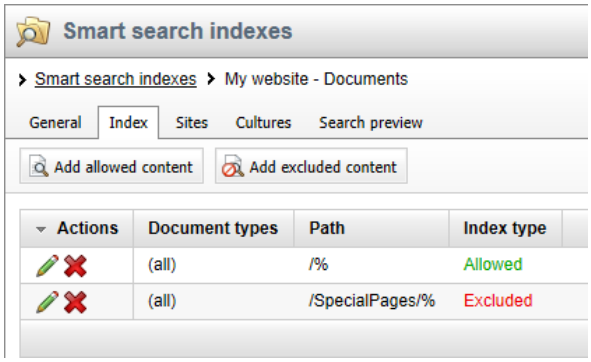
There are two types of search indexes available for the documents in a website's content tree.

Documents type indexes include information from general document fields such as metadata, the text content of certain web parts placed on page (menu item) documents, as well as the selected fields of

individual documents types (described in the Settings for particular object types topic). Data from other documents or objects displayed by web parts is not indexed. For example, the content of news documents displayed by a **Repeater** web part placed on an indexed document will not be added to the index etc.

Documents crawler type indexes directly parse the HTML output generated by documents, which means that all text located on or associated with a document is searchable. This allows document content to be searched more accurately than using a **Documents** type index. However, building and updating a **Documents crawler** index may require more time and resources, particularly in the case of large indexes and complex documents. The crawler accesses documents under a specific user account (the ***administrator*** account by default). You can set the details of the crawler account by adding keys to your project's web.config file. All keys related to smart search indexes are listed in the Smart search settings section of Appendix B - Web.config parameters.

The process used to define which documents on the site should be indexed is the same for both document index types. Specify allowed or excluded content on the **Index** tab of the index's editing interface by clicking **Add allowed content** or **Add excluded content** respectively.



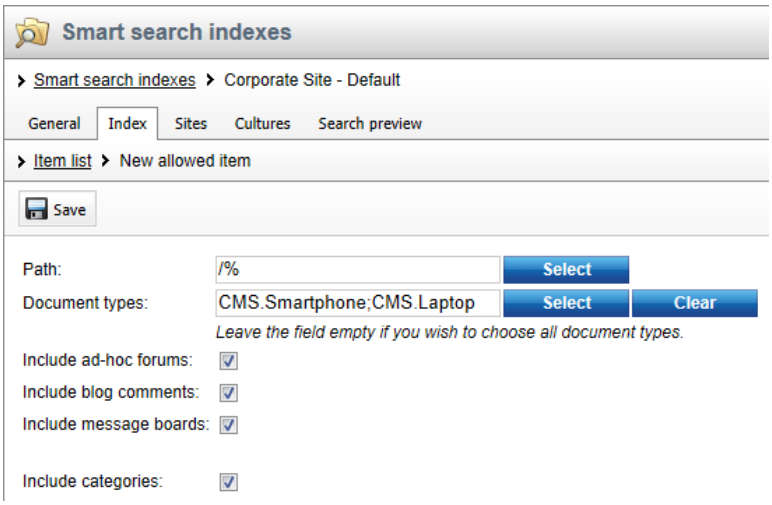
Adding allowed content

Allowed content defines which of the website's documents are included in the index. Specify documents using a combination of the following options:

- **Path** - path expression identifying the documents that should be indexed.
- **Document types** - allows you to limit which document types are included in the index.

The following properties define types of additional content that you can include in *Documents* search indexes. They are not available for *Documents crawler* indexes:

- **Include ad-hoc forums** - if checked, the index also includes the content of ad-hoc forums placed on the specified documents (if there are any).
- **Include blog comments** - if checked, the index also includes blog comments posted for the blog post documents.
- **Include message boards** - if checked, the index also includes message boards placed on the specified documents.
- **Include categories** - if enabled, the index stores the display names of Categories assigned to the specified documents. This means that the search results also include documents that belong to categories whose name matches the search expression.



Examples:

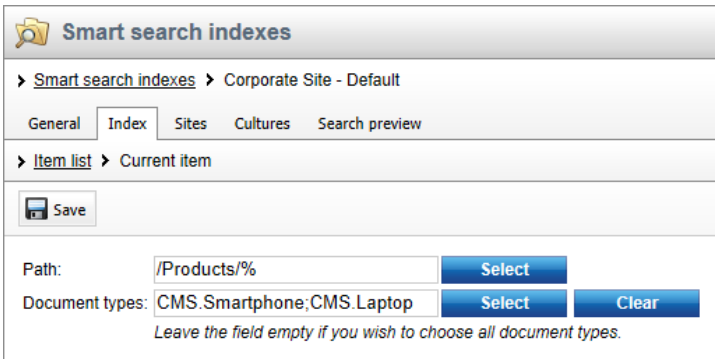
Allowed content settings	Result
• **Path**: /% • **Document types**: empty	Indexes all documents on the site.
• **Path:** /Partners • **Document types**: empty	Only indexes the */Partners* page, without the child pages placed under it.
• **Path**: empty • **Document types**: CMS.News	Indexes all documents of the *CMS.News* document type on the entire site. In this case, an empty path field value is equal the **/%** expression.
• **Path**: /Products/% • **Document types**: CMS.Smartphone;CMS.Laptop	Indexes all documents of the *CMS.Smartphone* and *CMS.Laptop* document types found under the */Products* section.

Adding excluded content

Excluded content allows you to remove documents or entire website sections from the allowed content. For example, if you allow **/%** and exclude **/Special-pages/%** at the same time, the index will include all documents on the site except for the ones found under the **/Special-pages** node.

You can specify the following options:

- **Path** - path expression identifying the documents that should be excluded.
- **Document types** - allows you to limit which document types are excluded from the index.



Examples:

Excluded content settings	Result
• **Path:** /Partners • **Document types**: empty	Excludes the */Partners* page from the index. Child pages are not excluded.
• **Path**: empty • **Document types**: CMS.News	Excludes all documents of the *CMS.News* document type from the index. In this case, an empty path field value is equal the ***/%*** expression.
• **Path**: /Products/% • **Document types**: CMS.Smartphone;CMS.Laptop	Excludes all documents of the *CMS.Smartphone* and *CMS.Laptop* document types found under the */Products* section from the index.

Excluding individual documents from all indexes

You can also exclude specific documents from all smart search indexing:

1. Go to **CMS Desk** and select the given document in the content tree.
2. In **Edit** mode, open the **Properties -> Navigation** tab.
3. Enable the **Exclude from search** property.
4. Click **Save**.

Customizing how document crawler indexes process page content (API)

By default, the system converts the HTML output of documents to plain text (stripped of all HTML tags, JavaScript and whitespace formatting) before saving it to document crawler indexes. If you wish to index the content of any tags or exclude parts of the page output, you can customize how the crawlers process the HTML.

You need to implement your custom functionality in a handler for the **OnHtmlToPlainText** event of the **CMS.SiteProvider.SearchHelper** class. This event occurs whenever a document search crawler processes the HTML output of a page.

To assign a method as the handler for the *OnHTMLToPlainText* event, add a new class to the **~/App_Code** folder of your web project (or **~/Old_App_Code** on web application installations). You can define the content of the class as shown below:

[C#]

```
using CMS.SettingsProvider;
using CMS.SiteProvider;

[DocumentCrawlerContentLoader]
public partial class CMSModuleLoader
{
    /// <summary>
    /// Attribute class for assigning event handlers.
    /// </summary>
    private class DocumentCrawlerContentLoaderAttribute : CMSLoaderAttribute
    {
        /// <summary>
        /// Called automatically when the application starts.
        /// </summary>
        public override void Init()
        {
            // Assigns a handler for the OnHtmlToPlainText event
            SearchHelper.OnHtmlToPlainText += new
SearchHelper.HtmlToPlainTextHandler(SearchHelper_OnHtmlToPlainText);
        }


        static string SearchHelper_OnHtmlToPlainText(string plainText, string
originalHtml)
        {
            // Add your custom HTML processing actions and return the result as a
string
        }
    }
}
```

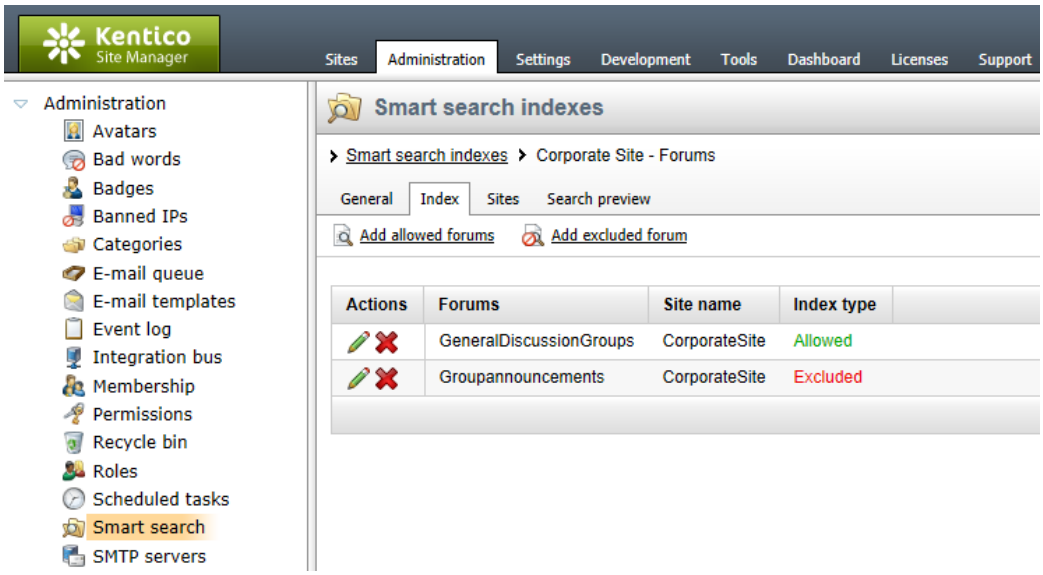
The *OnHTMLToPlainText* event provides the following string parameters to the handler:

- **plainText** - contains the page output already stripped of all tags and converted to plain text.
- **originalHTML** - allows you to access the raw page code without any modifications.

8.43.4.3 Defining forums index content

On the **Index** tab of a **forum index**, you can define which forums from the system will be indexed. This is done by defining allowed and excluded forums.

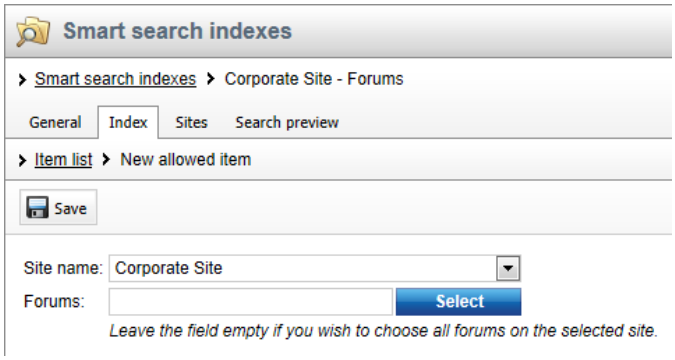
The dialogs for defining new allowed/excluded content can be accessed using **Add allowed forums** and **Add excluded forum**.



Adding allowed forums

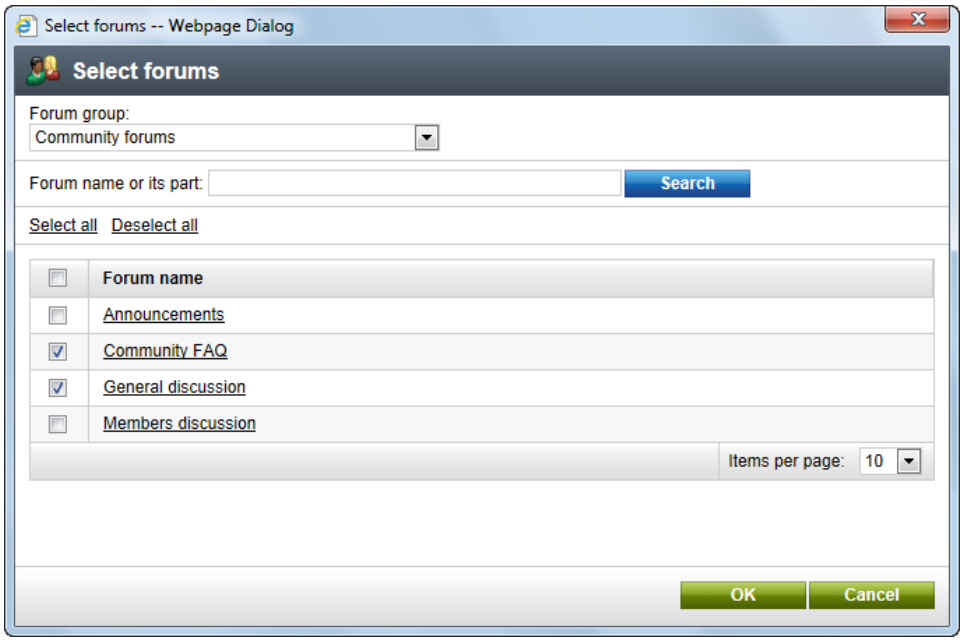
1. Click **Add allowed forums**.

2. In the following dialog, first use the **Site name** drop-down to choose the site whose forums will be indexed. If you select **(all)**, all forums on all sites in the system will be indexed. Only websites assigned to the index on the **Sites** tab are available for selection.



3. If you selected a particular site in the previous step, click the **Select** button next to the **Forums** field. The dialog depicted in the screenshot below will be displayed.

Use the **Forum group** drop-down to select a forum group. Its child forums will be listed below. To include a forum in the index, enable (☑) the appropriate check-boxes. Click **OK** to save the settings.



Alternatively, you can manually enter the code names of forums into the **Forums** field, separated by semicolons. All forums on the selected site can be added to the index this way, including group forums. The asterisk character (*) can be used as a wildcard for any number of characters. For example, entering **community** adds all forums that contain the string *community* in their code name to the index.

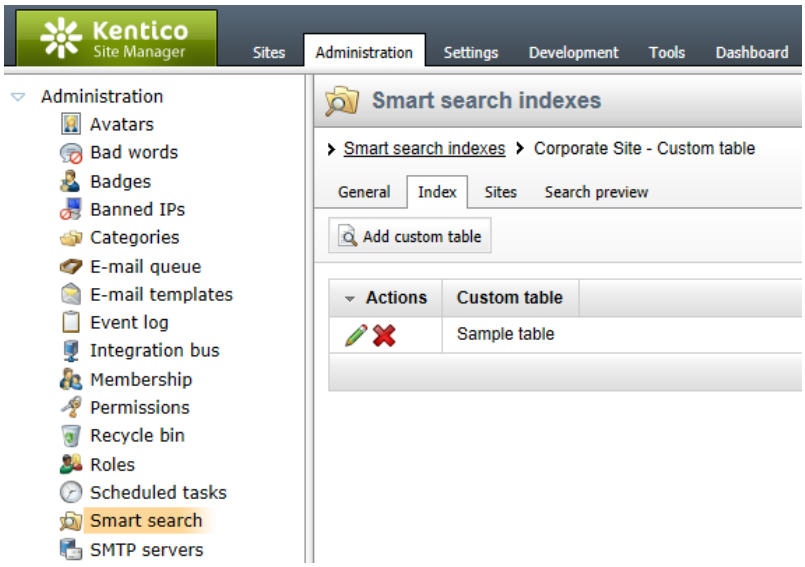
Adding excluded forums

Excluded forums make sense only when you have all forums defined as allowed. By defining a forum as excluded, it will not be indexed and all forums except for the excluded one will be indexed.

You can define an excluded forum using **Add excluded forums**, while the procedure is the same as when adding allowed forums.

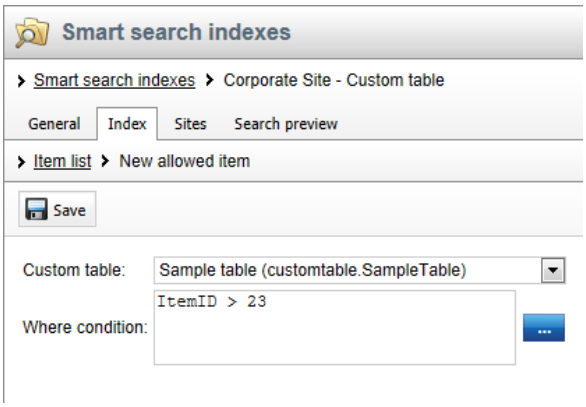
8.43.4.4 Defining custom tables index content

When editing () a custom table index on the **Index** tab in **Site Manager -> Administration -> Smart search**, you can see a list of custom tables included in the index. Custom tables can be added to the index using the **Add custom table** button. You can also **Edit** () the way listed custom tables are indexed or **Delete** () them from the list.



When adding a new custom table to the index or editing an existing one, you have the following options:

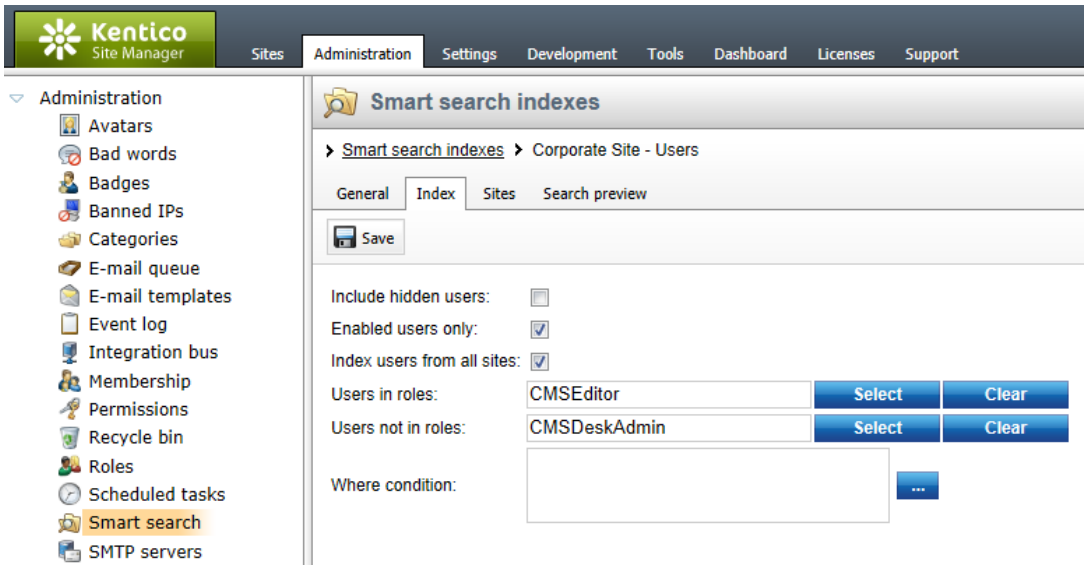
- **Custom table** - custom table to be indexed.
- **Where condition** - WHERE clause of the queries run against the custom table when building the index. This can be used to limit which records (rows) in the table should be included.



8.43.4.5 Defining user index content

When editing (✎) a user index on the **Index** tab in **Site Manager -> Administration -> Smart search**, you can limit which users will be indexed. The dialog allows you to set the following limitations:

- **Include hidden users** - if enabled, hidden users will be indexed.
- **Enabled users only** - if enabled, only enabled users will be indexed.
- **Index users from all sites** - if enabled, users from all sites will be indexed. If disabled, only users from the sites assigned on the *Sites* tab will be indexed.
- **Users in roles** - if entered, only users from the entered roles will be indexed.
- **Users not in roles** - if entered, only users who are not in the entered roles will be indexed.
- **Where condition** - WHERE clause of the queries run against the *View_CMS_User* view in order to retrieve users when building the index. This can be used to limit which users should be included.



You can also define which user fields are indexed:

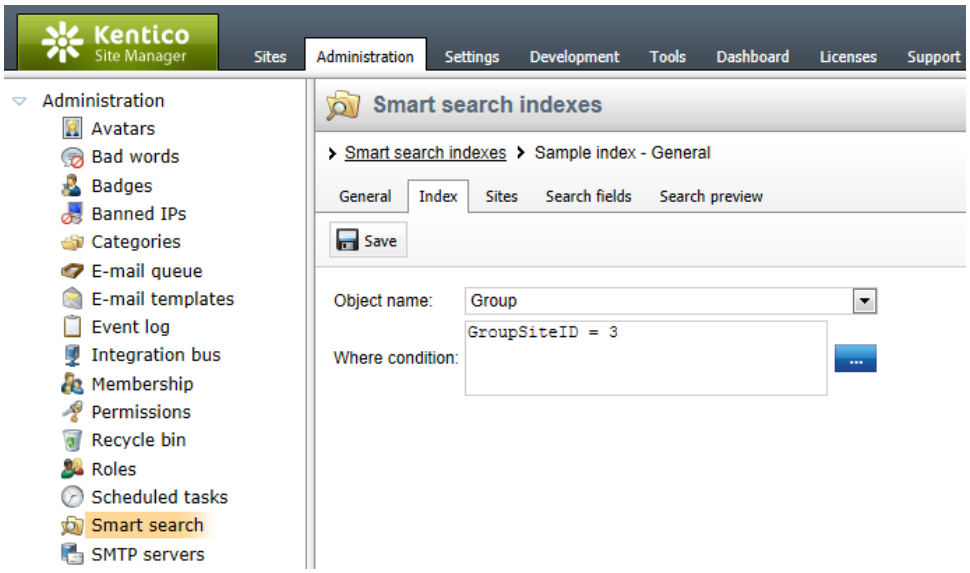
1. Go to **Site Manager -> Development -> System tables**.
2. Edit (🖉) the **User** (**CMS_User**) system table.
3. Configure the field settings on the **Search fields** tab.

8.43.4.6 Defining general index content

General indexes allow searching through any type of objects used within the Kentico CMS system. This includes items you may recognize from various sections of the administration interface, module related objects and more. Specific examples could be web parts, page templates, groups, sites etc.

While editing an index, its content can be specified on the **Index** tab by defining the following two properties:

- **Object name** - sets the type of objects that should be searched for by the index. The index will store information representing objects in the system of the specified type. When an object is created, removed or has one of its fields modified, the index will automatically be updated to reflect these changes. Since there is a very large amount of object types in the CMS system, it will in most cases be necessary to choose the *(more items...)* option and use the full dialog to make your selection.
- **Where condition** - allows a custom WHERE clause to be set for the queries used to retrieve data when building the index. This can be used to limit which records should be included.



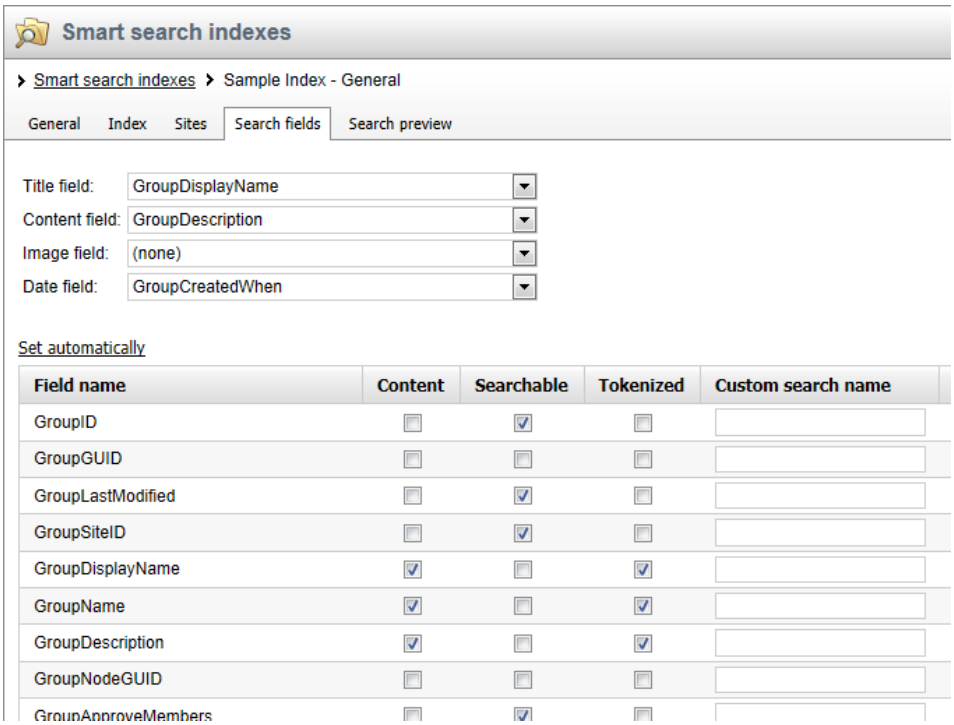
Once the object type is selected, it is necessary to configure which fields should be included in the index on the **Search fields** tab. The settings here affect how objects can be searched for.

Using the drop-down lists at the top, you can specify what kind of data should be displayed in search results:

- **Title field** - specifies which field will be used as the title of result items.
- **Content field** - specifies which field will be used for the content extract of result items.
- **Image field** - specifies which field will be used for the image of result items.
- **Date field** - specifies which field will be used for the date and time displayed with result items.

The rows of the main table contain the fields available for the specified type of objects. These fields correspond with the columns of the database table used to store objects of the given type. The following options can be set for individual fields:

- **Content** - if checked, the content of the field will be indexed and searchable the standard way. Otherwise, objects will not be included in the search results even if they contain the searched for expression in this field.
- **Searchable** - if checked, the content of the field will be searchable by entering an expression in format: *<field code name>:<searched phrase>*. Please refer to the Search syntax topic for more information about field searches.
- **Tokenized** - indicates if the content of the field should be processed by the analyzer when indexing. The general rule is to use this for *Content* fields and not for *Searchable* fields.
- **Custom search name** - relevant for *Searchable* fields. The specified value is used as a substitute for the field code name in the *<field code name>:<searched phrase>* search expression. If a value is entered, the original code name can't be used.



The configuration of search fields is global for objects of the given type. If there are multiple general indexes for a single object type (i.e. using the same **Object name**), changing the search field settings for one index will also affect the others.

You can use the **Set automatically** link to apply the default configuration for the specific object type. Any changes must be confirmed by clicking the **OK** button below the table.

General indexes and Sites

The content of general indexes is not affected by the selection made on the **Sites** tab. It only determines on which websites the index will be available for use (through smart search web parts).

If you wish to configure a general index to search only through objects assigned to a specific site, we recommend using the **Where condition** property on the **Index tab**.

For example, a general index with the *Group* **Object name** and *GroupSiteID = 3* set in its **Where condition** would only index groups created under the site with a SiteID equal to 3.

This approach is of course only possible for site-bound object types.

8.43.4.7 Defining custom index content

When creating a custom index, you do not define the content on the **Index** tab. Instead, you must implement all functionality of the index by writing code. In the administration interface, you only need to specify the names of the assembly and class that contain the custom index logic.

To define a custom index, create a class that implements the **CMS.Siteprovider.ICustomSearchIndex** interface.

To integrate this type of class into the application, you can:

- Create a new assembly (Class library) in your web project and include the index class there. When using this approach, you must add the appropriate references to both the assembly and the main CMS project. You can find a sample search index in the **CustomSearchIndex** project located in your Kentico CMS installation directory (typically *C:\Program Files\KenticoCMS\<version>\CodeSamples\CustomSearchIndex*).
- Define the custom index in **App_Code** and load the class via the API. This ensures that the system automatically compiles the index class and you do not need to use a separate assembly. The example below demonstrates this approach.

Writing custom index code

The following example shows how to create a custom index that searches the content of text files:

1. Open your web project in Visual Studio and add a new class into the **App_Code** folder (or **Old_App_Code** if the project is installed as a web application). For example, name the class **TextFileIndex.cs**.

2. Edit the class and add the following references:

[C#]

```
using CMS.EventLog;
using CMS.SiteProvider;
using CMS.SettingsProvider;
using CMS.GlobalHelper;
using CMS.IO;

using Lucene.Net.Index;
using Lucene.Net.Documents;
```

3. Make the class implement the **ICustomSearchIndex** interface.

[C#]

```
public class TextFileIndex : ICustomSearchIndex
```

4. Define the **Rebuild** method inside the class. You must always include this method when writing a custom index. It fills the index with data, which determines what kind of searches the index provides. The system calls the method when building the index for the first time and on each subsequent rebuild.

[C#]

```
/// <summary>
```

```
/// Fills the index with content.
/// </summary>
/// <param name="srchInfo">Info object representing the search index</param>
public void Rebuild(SearchIndexInfo srchInfo)
{
    // Checks whether the index info object is defined.
    if (srchInfo != null)
    {
        // Gets an index writer object for the current index.
        IndexWriter iw = srchInfo.GetWriter(true);

        // Checks the whether writer is defined.
        if (iw != null)
        {
            try
            {
            // Gets an info object of the index settings.
            SearchIndexSettingsInfo sisi = srchInfo.IndexSettings.Items
[SearchHelper.CUSTOM_INDEX_DATA];

            // Gets the search path from the Index data field.
            string path = Convert.ToString(sisi.GetValue("CustomData"));

            // Checks whether the path is defined.
            if (!String.IsNullOrEmpty(path))
            {
                // Gets all text files from the specified directory.
                string[] files = Directory.GetFiles(path, "*.txt");

                // Loops through all files.
                foreach (string file in files)
                {
                    // Gets the current file info.
                    FileInfo fi = FileInfo.New(file);

                    // Gets the text content of the current file.
                    string text = fi.OpenText().ReadToEnd();

                    // Checks that the file is not empty.
                    if (!String.IsNullOrEmpty(text))
                    {
                    // Converts the text to lower case.
                    text = text.ToLower();
                    // Removes diacritics.
                    text = TextHelper.RemoveDiacritics(text);

                    // Creates a new Lucene.Net search document for the current
text file.
                    Document doc = SearchHelper.CreateDocument
(SearchHelper.CUSTOM_SEARCH_INDEX, Guid.NewGuid().ToString(),
SearchHelper.INVARIANT_FIELD_VALUE, fi.CreationTime,
SearchHelper.INVARIANT_FIELD_VALUE);

                    // Adds a content field. This field is processed when the
search looks for matching results.
                    SearchHelper.AddField(doc, SearchHelper.CONTENT_FIELD, text,
false, true);
```

```
                    // Adds a title field. The value of this field is used for the
search results title.
                    SearchHelper.AddField(doc, SearchHelper.CUSTOM_TITLE,
fi.Name, true, false);

                    // Adds a content field. The value of this field is used for
the search result excerpt.
                    SearchHelper.AddField(doc, SearchHelper.CUSTOM_CONTENT,
TextHelper.LimitLength(text, 200), true, false);

                    // Adds a date field. The value of this field is used for the
date in the search results.
                    SearchHelper.AddField(doc, SearchHelper.CUSTOM_DATE,
fi.CreationTime, true, false);

                    // Adds a url field. The value of this field is used for link
urls in the search results.
                    SearchHelper.AddField(doc, SearchHelper.CUSTOM_URL, file,
true, false);

                    // Adds an image field. The value of this field is used for
the images in the search results.
                    //SearchHelper.AddField(doc, SearchHelper.CUSTOM_IMAGEURL,
"textfile.jpg", true, false);

                    // Adds the document to the index.
                    iw.AddDocument(doc);
                    }
                }

                // Optimizes the index.
                iw.Optimize();
            }
            }

            // Logs any potential exceptions.
            catch (Exception ex)
            {
                EventLogProvider.LogException("CustomTextFileIndex", "Rebuild", ex);
            }
            // Always close the index writer.
            finally
            {
                iw.Close();
            }
        }
    }
}
```

You need to write the code of the *Rebuild* method according to the specific purpose of the index, but all indexes use the following general steps:

a. Get an instance of the **Lucene.Net.IndexWriter** object for the search index.
b. Define search **Documents** (*Lucene.Net.Documents.Document* objects) and their fields for the items that you wish to add to the index.
c. Call the **AddDocument** method of the *IndexWriter* for every search document.
d. Optimize the index using the **Optimize** method of the *IndexWriter* once you have added all required

search documents.
e. Call the **Close** method of the *IndexWriter*.

Index data parameters

The **SearchIndexInfo** parameter of the *Rebuild* method allows you to access the data fields of the corresponding search index object. The sample code loads the content of the **Index data** field and uses it to define the path to the searched text files.

When writing your own custom indexes, you can use this field as a string parameter for any required purpose. This way, you may easily modify the behavior of the index directly from the administration interface without having to edit its code.

Loading App_Code index classes

For indexes defined in the *App_Code* folder, you need to ensure that the system loads the appropriate class when building the index. You can find additional information related to this topic in Registering custom classes in App_Code.

1. Create another class in the **App_Code** folder (or **Old_App_Code** if the project is installed as a web application). For example, name the class **ClassLoader.cs**.

2. Edit the class and add the following reference:

[C#]

```
using CMS.SettingsProvider;
```

3. Delete the default class declaration and its content. Instead, extend the **CMSModuleLoader** partial class and define a new attribute inheriting from *CMS.SettingsProvider.CMSLoaderAttribute*:

[C#]

```
[ClassLoader]
public partial class CMSModuleLoader
{
    /// <summary>
    /// Attribute class for loading custom classes.
    /// </summary>
    private class ClassLoader : CMSLoaderAttribute
    {

    }
}
```

4. Enter the following code into the **ClassLoader** attribute class:

[C#]

```
/// <summary>
/// Called automatically when the application starts.
/// </summary>
public override void Init()
{
    // Assigns a handler for the OnGetCustomClass event.
    ClassHelper.OnGetCustomClass += ClassHelper_OnGetCustomClass;
}


/// <summary>
/// Gets a custom class object based on the given parameters.
/// </summary>
private void ClassHelper_OnGetCustomClass(object sender, ClassEventArgs e)
{
    if (e.Object == null)
    {
        // Checks the name of the requested class.
        switch (e.ClassName)
        {
            // Gets an instance of the TextFileIndex class.
            case "TextFileIndex":
                e.Object = new TextFileIndex();
                break;
        }
    }
}
```

For custom indexes, the value of the **ClassName** property of the **ClassHelper_OnGetCustomClass** handler's *ClassEventArgs* parameter matches the **Class name** specified for the given index on its **Index** tab (*TextFileIndex* in this example). The handler checks the class name to identify which index is being rebuilt and then passes on an instance of the appropriate class.

Registering custom search indexes

1. Open Kentico CMS, go to **Site Manager -> Administration -> Smart Search** and click **New Index**. Fill in the following properties:

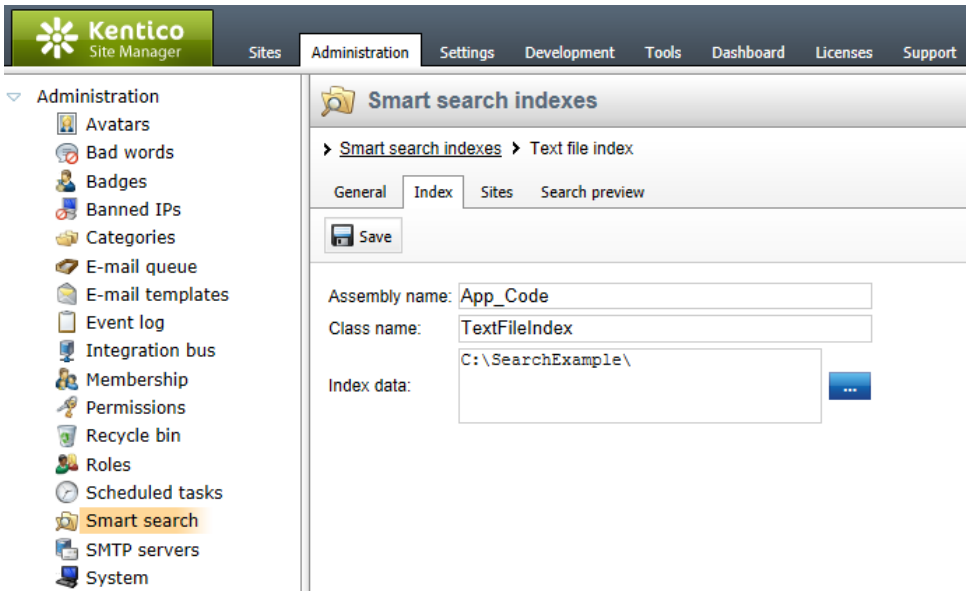
- **Display name**: Text file index
- **Index type**: Custom Index

Click **Save**.

2. Switch to the **Index** tab and enter the names of the assembly and class where the custom index is implemented:

- **Assembly name**: App_Code
- **Class name**: TextFileIndex

3. Type any required parameters into the **Index data** field. In this example, you need to specify the file system path of the folder containing the text files that the index will search. You can create a new folder for this purpose, e.g. *C:\SearchExample* and add some text files into it.



Click **Save**.

4. Go to the **General** tab and **Rebuild** the index.

Result

The index is now fully functional. To test the index, switch to **Search preview** tab and try searching for any words from the text files created in the *SearchExample* folder.

8.43.4.8 Using a custom analyzer

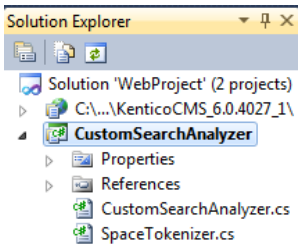
In case the selection of built-in indexing analyzers is insufficient, a custom-written or third-party analyzer may be specified for a search index. This gives you the option of performing tokenization according to your particular requirements.

A custom analyzer may be used with any type of smart search index, but its code files must first be created and added to your web project. The class defining a custom analyzer must inherit from the **Lucene.Net.Analysis.Analyzer** class.

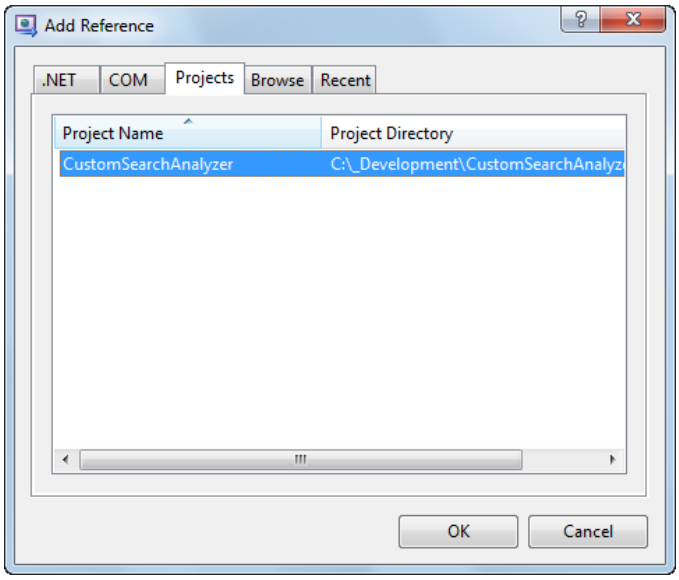
The following example demonstrates how a custom analyzer can be assigned to a smart search index. This sample analyzer divides text into tokens only at space characters and is purely for demonstrative purposes.

1. Copy the **CustomSearchAnalyzer** project from your Kentico CMS installation directory (by default *C:\Program Files\KenticoCMS\<version>\CodeSamples\CustomSearchAnalyzer*) to a development folder (not under the web project).

2. Open your CMS web project using the **WebProject.sln** file (**WebApp.sln** if you installed the project as a web application). Click **File -> Add -> Existing Project** and select the **CustomSearchAnalyzer.csproj** file in the folder where you copied the **CustomSearchAnalyzer** project. Your Solution Explorer window should now look like this:



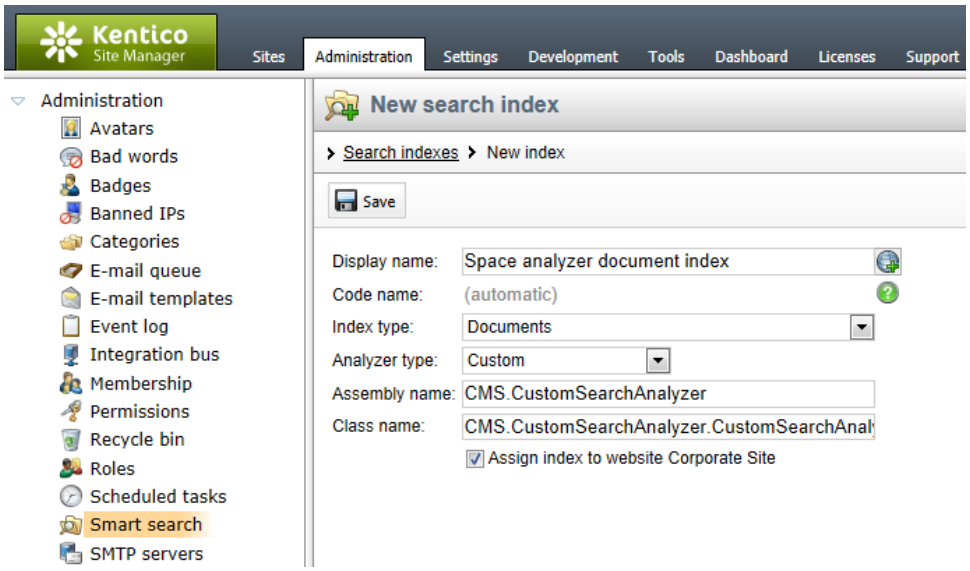
3. Right-click the CMS website object (or the **CMSApp** project if your installation is a web application) and choose **Add Reference**. Switch to the **Projects** tab, select the **CustomSearchAnalyzer** project and click **OK** to add the project reference.



4. Right-click your solution and select **Rebuild Solution**.

5. Open Kentico CMS, go to **Site Manager -> Administration -> Smart Search** and click **New Index**. Fill in the following properties:

- **Display name**: Space analyzer document index
- **Analyzer type**: Custom
- **Assembly name**: CMS.CustomSearchAnalyzer
- **Class name**: CMS.CustomSearchAnalyzer.CustomSearchAnalyzer
- **Index type**: Documents



This defines the new index and specifies the assembly and class names where the custom analyzer is implemented in the **CustomSearchAnalyzer** project that you added to your solution. Click **Save**.

6. Next, switch to the **Index** tab, click **Add allowed content**, enter */%* into the **Path** field to index all documents and click **Save**.

7. Switch over to the **Cultures** tab and add the culture(s) used by your website.

8. Finally, go to the **General** tab and **Rebuild** the index. Once complete, the index should be functional. If you wish, you can test the index and its analyzer by switching to the **Search preview** tab and trying to search for words from the content of the website's documents.

Defining custom analyzers in App_Code

If you do not wish to add a new project to the application, you may alternatively create the required classes under the **App_Code** folder, and register the custom analyzer class using the same approach described in the Defining custom index content topic.

In this case, you need to change the *Assembly* and *Class name* set for the analyzer accordingly.

8.43.5 Settings for particular object types

Documents and other objects in Kentico CMS are often complex data structures with many different fields. In most cases, not all of these fields will be relevant to the search that you are trying to implement. Additionally, it is always a good idea to avoid indexing unnecessary fields to keep your indexes as small (and fast) as possible. For this reason, most types of objects have the option of adjusting search settings for fields.

The sections below describe how these settings may be configured for particular object types.

Document types, Custom tables and Users

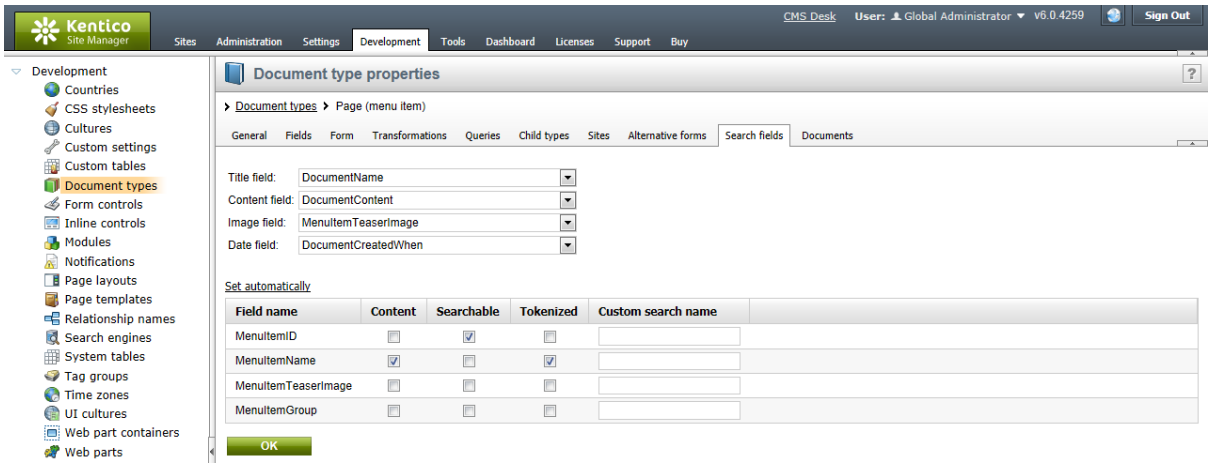
Settings for document types, custom tables and users are almost identical. You can find them on the **Search fields** tab in the editing interfaces at **Site Manager -> Development -> Document types / Custom tables / System tables -> Edit ( ) User**.

In the top part of the tab, you can specify how the objects will be displayed in search results:

- **Title field** - specifies which field will be used as the title of result items.
- **Content field** - specifies which field will be used for the content extract of result items.
- **Image field** - specifies which field will be used for the image of result items (not available for users, Avatar images are used for users by default).
- **Date field** - specifies which field will be used for the date and time displayed with result items.

The rows of the table in the bottom part of the page represent fields as defined on the **Fields** tab. The following options can be set for individual fields:

- **Content** - if checked, the content of the field will be indexed and searchable the standard way.
- **Searchable** - if checked, the content of the field will be searchable by entering an expression in format: *<field code name>:<searched phrase>*. Please refer to the Search syntax topic for more information about field searches. This option must also be enabled in order for the field to be usable in Smart search filters.
- **Tokenized** - indicates if the content of the field should be processed by the analyzer when indexing. This allows the search to find results that match individual tokens (subsets) of the field's value. If disabled, items will only be considered a match if the full value of the field exactly matches the search expression. The general rule is to use this for *Content* fields and not for *Searchable* fields.
- **Custom search name** - relevant for *Searchable* fields. The specified value is used as a substitute for the field code name in the *<field code name>:<searched phrase>* search expression. If a value is entered, the original code name can't be used.



Please note

Documents crawler type search indexes directly index the HTML output of documents and as a result are not affected by the field settings of document types.

E-commerce SKUs (and general document fields)

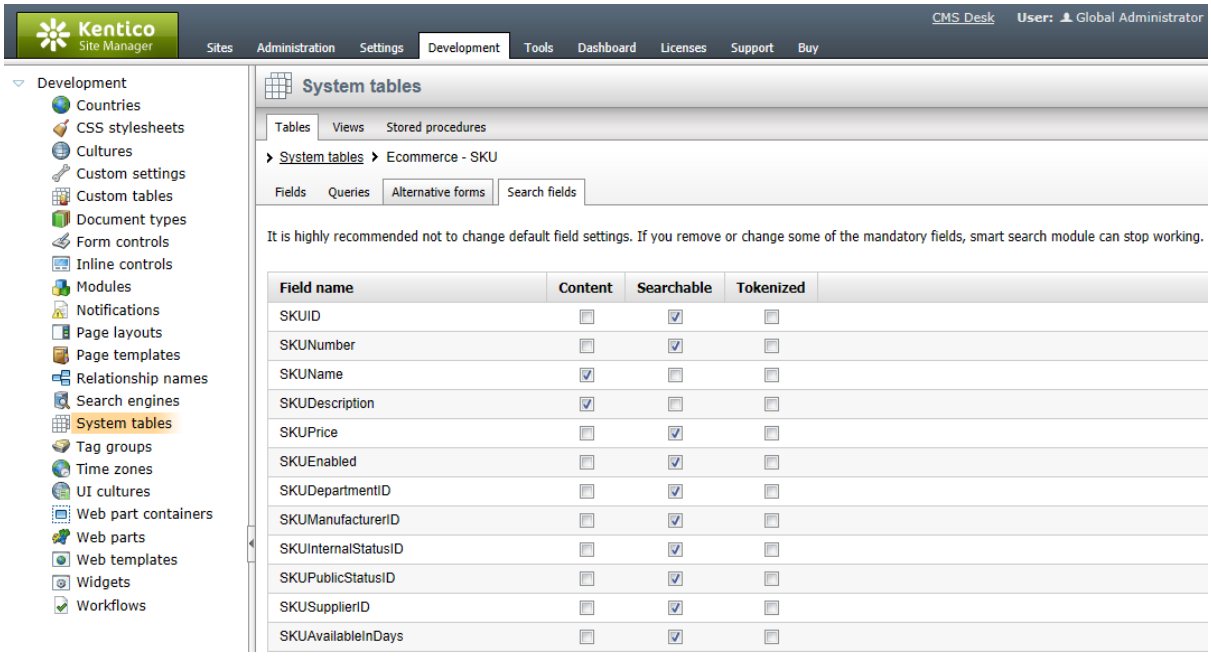
Similar settings are available for **E-commerce SKUs** (products). These can be configured in **Site Manager -> Development -> System tables** by editing (✎) the **Ecommerce - SKU** table on the **Search fields** tab. Here, SKU fields are joined together with general document fields, such as those used to store the content of editable regions on the page (*DocumentContent*) or the content of text web parts (*DocumentWebParts*). The search settings of general fields affect all documents, even those that are not products.

In this case, only the following options are available:

- **Content** - if checked, content of the field will be indexed and searchable the standard way.
- **Searchable** - if checked, content of the field will be searchable by entering an expression in format: *<field code name>:<searched phrase>.* Search requests of this type only search through the given field and not through the other fields.
- **Tokenized** - indicates if the content of the field should be processed by the analyzer when indexing. The general rule is to use this for *Content* fields and not for *Searchable* fields.

Important!

It is highly recommended that you do not modify settings other than those of your custom fields. If you modify the settings of some of the default fields, the functionality of searching through SKUs may get broken.



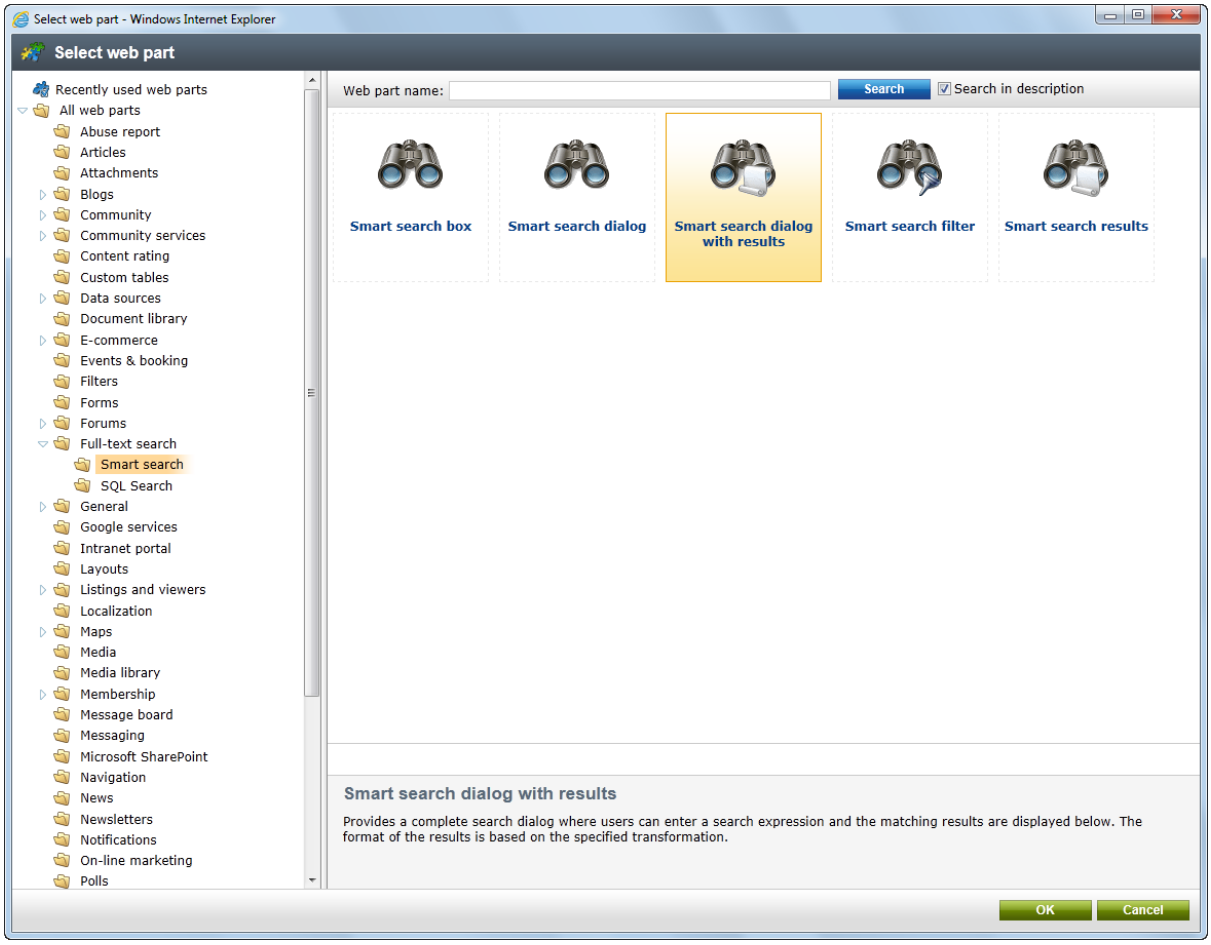
General objects

For objects searchable by *General* indexes, these settings are available directly on the **Search fields** tab when editing an index in **Site Manager -> Administration -> Smart search** (as described in Managing indexes -> Defining general index content).

8.43.6 Available web parts

The Smart search module comes with a set of web parts that can be used to build a search interface on the pages of your website. Only the most important web part properties are mentioned here. For a complete list and explanations of the web part properties, please refer to the Kentico CMS Web Parts reference or click the help icon (?) in the top right corner of the web part properties window.

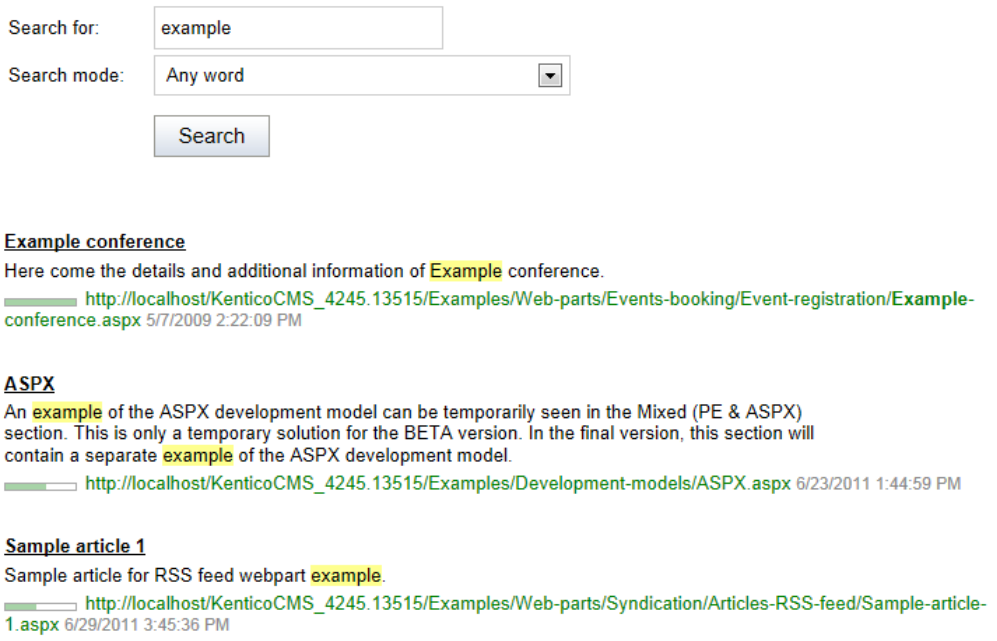
Smart search web parts are located in the **Full-text search -> Smart search** category.



The following web parts are included with the Smart search module by default:

Smart search dialog with results

This is an all-in-one web part that allows both searching and displaying of the results.



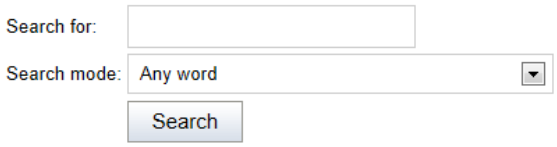
The following properties are the most important for setting up the smart search:

Property Name	Description
Indexes	Determines which index will be searched. Multiple indexes can be entered and then searched at the same time.
Transformation name	Name of the transformation used to display search results. There are two default transformations suitable for this purpose: • **CMS.Root.SmartSearchResults** • **CMS.Root.SmartSearchResultsWithImages**
Search options	Sets the level of syntax that is allowed in search expressions: • **Basic** - users are allowed to input special syntax, but cannot search specific fields. • **None** - users can only enter text, everything is processed as a part of the search expression. • **Full** - all search options can be used, including field searching. More information can be found in the Search syntax topic.
Search condition	Sets a condition that is added to any submitted search expressions. The condition is built using the smart search syntax, i.e. special symbols (+ -) and field conditions. For example: *+articleid:[(int)25 TO (int)150]*
Search results order	Defines the order in which search results are displayed.

	You can specify one or more search fields (separated by commas) according to which the results will be sorted. The ##SCORE## macro can be used to order results by their score (relevance). The default order is ascending, you can change this using the DESC keyword (e.g. articleid DESC). If you encounter the "field <fieldname> does not appear to be indexed" error when using multiple indexes, try specifying the type of the field using the following syntax: *(date)documentcreatedwhen*

Smart search dialog

This web part needs to be placed on a page together with the **Smart search results** web part. The combined functionality of the two web parts is nearly identical to a Smart search dialog with result. The main difference between them is that these two web parts can be placed separately at different locations on the page.



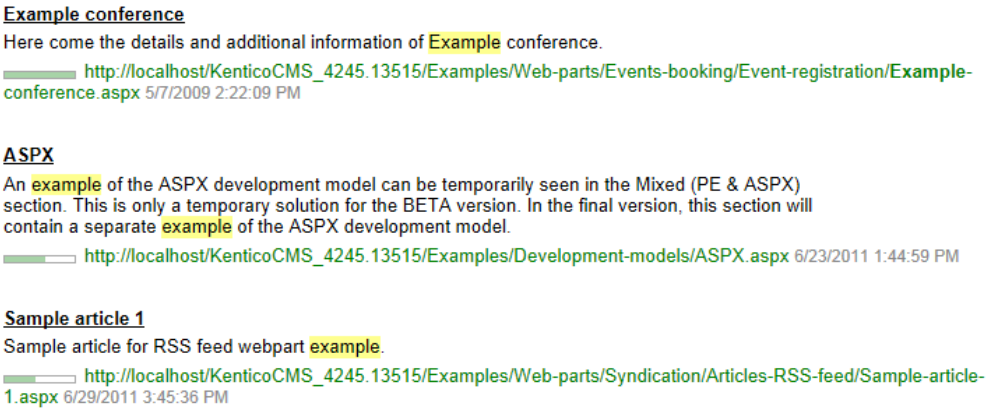
The web part also supports a special mode that can be enabled through the **Show only search button** property. In this case, only the search submit button is displayed without the search textbox and mode selector. This is intended for scenarios that utilize Smart search filters to specify all of the search parameters. When combined with a textbox filter, it also provides a way to separate the textbox from the search button.

Smart search box

This web part is similar to the Smart search dialog, with the difference that users cannot select the Search mode, which is hard-set in its web part properties. It is useful where limited space is available for the web part, e.g. in menus, etc. Additionally, it can redirect users to a different result page, where the appropriate **Smart search results** or **Smart search dialog with results** web part is located.

Smart search results

This web part is used to display results of a search request sent from a **Smart search box** or **Smart search dialog** web part. It can be placed either on the same page as one of the two web parts or on a different one. If it is placed on a different page, the page needs to be specified by their **Search results page URL** property. The web part can be configured using the same properties described for the Smart search dialog with results.



Smart search filter

This web part allows users to set parameters that affect the scope of the search or the order of the displayed results. It may also be used as a separate search textbox. You can find a detailed description of how this works in the Using the Smart search filter topic.

8.43.7 Using the Smart search filter

The **Smart search filter** web part allows users to limit the range of objects that will be searched (conditional filter), or define the order of the search results. It is designed to be connected to the **Smart search dialog** or **Smart search dialog with results**. You can connect more than one filter to these web parts.

The behavior of the smart search filter is primarily defined by the properties explained below. You can find descriptions of the web part's other properties in the Kentico CMS Web Parts reference by clicking on the help icon (?) in the top right corner of the web part properties window.

Property Name	Description
Search webpart ID	ID of the Smart search dialog or Smart search dialog with results web part to which the filter should be connected.
Filter mode	Determines what type of user interface element will be used to set the filtering options. Possible choices are a drop-down list, checkboxes, radio buttons or a text box.
Filter auto postback	Indicates whether the search results should automatically be refreshed (via postback) whenever a different filtering option is selected by a user. Not applicable when using the text box **Filter mode**.
Values	This property is used to specify the options that will be available for selection in the filter. Please see the **Defining filtering options** section below for details.
Query name	Name of the query which can be used instead of the *Values* property to

	dynamically create the filtering options. The query must return the appropriate values depending on the type of the filter.
	
	For example, for a standard conditional filter, three columns need to be loaded, which will then be used in the following order: *<index field name>,<value of the field>,<displayed text>*
	
	The following is a sample query that loads all document types as conditional filtering options:
	
	`SELECT TOP 1000 '+classname', ClassName, ClassDisplayName`
	`FROM CMS_Class`
	`WHERE ClassIsDocumentType = 1`

Filter clause	Sets a clause that overrides the logical values specified for filtering options. Possible choices are: • **None** - no clause is added and the original logical values set for individual filtering options are used. • **Must** - indicates that the conditions in all filtering options must be fulfilled (adds the + symbol). • **Must not** - indicates that the conditions in all filtering options must not be fulfilled (adds the - symbol). Conditions are inverted compared to the *Must* option.
Filter is conditional	If true, the filter limits the scope of the objects that are searched (where condition). If false, the filter determines the order in which the search results are displayed (order by condition).

Examples of how filters work can be found on the sample Corporate Site, specifically on the **Examples -> Web parts -> Full-text search -> Smart search -> Smart search filter** and **Faceted search** pages.

Defining filtering options

The most important part of a Smart search filter web part's configuration is the definition of the options that should be offered to users. In most cases, this will be done through the **Values** property of the web part. If you wish to use the **Query name** property to load the options dynamically, the rules described below also apply to the results retrieved by the query.

The format used when entering the options depends on the type of the filter:

- **Conditional filters:**

In the case of conditional filters, define one option per line in format:

<index field name>;<value of the field>;<displayed text>

The logical meaning of each filtering option must also be specified. If the + symbol is added before the option, then only objects whose value in the given field matches the value specified in the second part of the definition will be included in the search. If the - symbol is added, only results that do **not** match the

value will be returned.

When entering integer or double type fields as filter options, the type of the value needs to be specified in the following way:
*+DocumentCreatedByUserID;**(int)**53;Administrator*

Examples:

- ;;All
- +classname;cms.smartphone;Smartphones
- +_created;[{%CurrentDateTime.AddDays(-7)|(tosearchdatetime)%} TO {%CurrentDateTime|(tosearchdatetime)%}];Past week
- +_content;product;Results related to products

- **Search result order filters:**

When creating filters that change the order of the results (i.e. the **Filter is conditional** property is disabled), the option definition format described above is also used, but the second part that contains the field value should be left empty. The order will be determined by the values in the specified field.

Examples:

- ##SCORE##;;Score
- documentcreatedwhen;;Creation date
- SKUPrice DESC;;Price descending

- **Text box filter mode:**

Because text box filters do not offer any selection options, the **Values** property is instead used to determine which index fields should be searched when a user enters an expression into the text box. Multiple fields can be specified, each one on a new line. Like with standard conditional filters, the + and - symbols determine whether the search should return results that contain the text box value in the given field, or only those that do not.

For example, a text box filter could have *+DocumentTags* in its Value definition. Visitors would then be able to enter the names of document tags into this text box and perform a standard search. The retrieved results would be filtered to contain only those documents that are marked by the specified tags.

If you wish to create a text box filter that behaves like a regular search box, use *+_content* as its only field name.

A text box filter may also be used to determine the order of the search results. In this case, the **Filter is conditional** property must be disabled and the **Values** property should be left empty. The name of the field used for ordering can then be entered by users into the text box. This scenario is not recommended for use by regular visitors on the live site.

Important!

Please note that in order for the filter to work correctly, the data fields used in the option definitions must be set as **Searchable** in the smart search field configuration of the given object type.

It is also possible to create filtering options for the fields that are marked as **Content** by using *_content* as the field name. Such a condition would be fulfilled if the value is found in any of the content fields.

Information about the search field configuration of objects can be found in the Settings for particular object types topic.

Filtering without search text (Faceted search)

In some cases, you may wish to implement a faceted search based purely on filters. This allows users to get search results simply by selecting filtering options, without the need to enter and submit search text. To achieve this result, **Configure** () the properties of the web part used to display the search results (Smart search results or Smart search dialog with results) according to the following instructions:

1. First, make sure that the **Search text required** property is disabled.
2. Unless you wish to force users to use a filtering option that works with standard content fields (those that have *_content* as their field name), also leave the **Block field-only search** property disabled.
3. Then enable **Search on each page load**, which ensures that the results are automatically displayed whenever the page is loaded, even without input from a search dialog.
4. For additional convenience, enable the **Filter auto postback** property of the used Smart search filter web part(s). Users are then able to instantly refresh the results simply by changing the filtering options.

8.43.8 Search syntax

A detailed description of Lucene query parser syntax can be found at http://lucene.apache.org/core/old_versioned_docs/versions/2_9_0/queryparsersyntax.html.

Based on the level of allowed syntax specified by the **Search options** property of **Smart search dialog with results** or **Smart search results** web parts, users can enter search queries with restrictions as described below:

- **Basic** - all Lucene search query syntax (as described on the page linked above) are recognized *except for field searching*.
- **None** - no query syntax is recognized. All text entered by users is processed as a part of the searched for expression.
- **Full** - all search query syntax is processed, including field searching.

Search syntax errors

If special search expressions are allowed by the **Search options** property, a parsing

error occurs if a user enters the query syntax incorrectly.

By default, the standard "no results" message is shown to the user in this case, and the error is entered into the application's Event log. If you wish to have syntax errors displayed in place of the results directly on the live site, you can enable the **Show parsing errors** property of the given smart search results web part.

Field search

Field searching may be used to define additional conditions in search expressions. All conditions must start with either the + or - symbol. The + symbol indicates that only results which fulfill the field condition should be returned. The - symbol has the opposite meaning, only results that do **not** contain the specified value in the given field will be retrieved.

For example:

- +network +NewsReleaseDate:[20080101 TO 20091231]

When searching for this expression using a document index, only news documents containing the word *network*, released in the year 2008 or 2009 will be returned.

Keep in mind that only fields set as **Searchable** in the field configuration of the given object type may be included in field searches. If there is no field name specified before a value in the expression (such as the word *network* in the example above), the system will search the index fields marked as **Content**. Information about the search field configuration of objects can be found in the Settings for particular object types topic.

Additionally, field searches will only be processed correctly if the search dialog web part has its **Search mode** set to *Any words*.

Unless the web part used to display the search results has its **Block field-only search** property enabled, it is also possible to perform direct field searches without any standard content keywords. This allows users to find records simply by entering an exact field value:

- *DocumentNodeID:(int)17* - returns the document with a nodeID equal to *17*.
- *NewsTitle:"New features"* - returns the news document titled *New features*.
- *SKUDepartmentID:(int)4* - returns all products that belong to the department that has *4* as its ID.

Using field search queries provides a great deal of flexibility, but it is not very convenient for regular website visitors. If you wish to allow users to limit the scope of searches through conditions on the live site, it is recommended to prepare appropriate Smart search filters.

Searching numeric fields

When performing field searches, the values specified are understood as strings by default. If you know that you are searching in **integer** or **double** fields, you will need to specify the type of the field the following way:

- NewsID:**(int)**22
- SKUPrice:**(double)**255.0
- DocumentNodeID:[**(int)**1 TO **(int)**100]

Searching date and time fields

Due to the same reason, search in **DateTime** fields also requires special syntax in format **<field name>:yyyymmddhhmm**. So for example:

- DocumentCreatedWhen:200812230101
- DocumentCreatedWhen:[200902020101 TO 200906020101]

If you need to specify date and time using a macro, you will need to use the **(tosearchdatetime)** parameter to convert the returned value to a format suitable for Lucene.

- {%CurrentDateTime|(tosearchdatetime)%}

You can also use the **(add)** parameter, which adds the specified amount of **minutes** to the value.

- {%CurrentDateTime|(add)-1440|(tosearchdatetime)%}

Field search with Stop and Simple analyzers

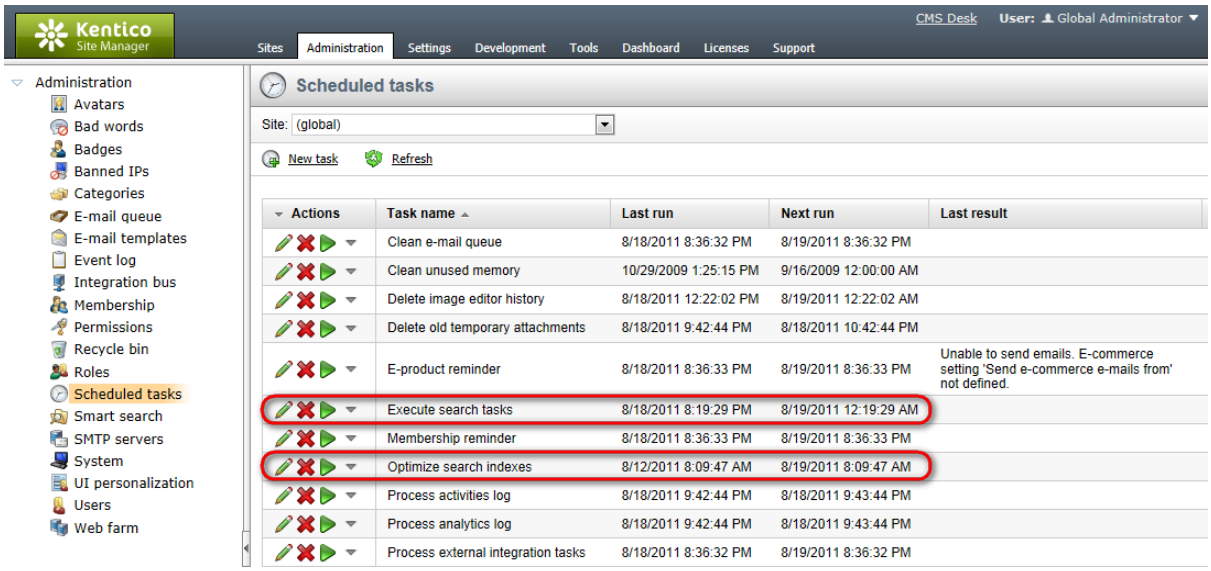
Indexes created by Stop and Simple analyzers cannot be searched using the standard field search format. This is by design, but you can use a workaround with a range query containing identical boundaries:

- newsid:[(int)22 TO (int)22]

8.43.9 Related scheduled tasks

The following two scheduled tasks are related to the Smart search module:

Scheduled task name	Description
Optimize search indexes	Allows you to set up automatic optimization of search indexes. Optimization defragments indexes, which can help maintain their performance (only significant in the case of very large indexes). This task is *disabled* by default.
Execute search tasks	Runs all existing search indexing tasks. Performed every 4 hours by default. **Note**: By default, the smart search automatically creates and executes indexing tasks whenever the indexed content changes. If you wish to run search indexing only through this scheduled task, disable automatic indexing task execution via the **CMSProcessSearchTasksByScheduler** web.config key. See Enabling Smart search indexing.



Refer to the Development -> Scheduler chapter for general information about scheduled tasks.

8.43.10 Searching attachments

Users can search through the content of attachment files uploaded to the database. The system searches attachments using the Microsoft SQL Server full-text search.

Note: You need to have your SQL server configured to support full-text search in files.

Once you have the SQL server configured, enable attachment searching through the following properties of the **Smart search dialog with results** or **Smart search results** web part:

Property Name	Description
Search in attachments	If enabled, the web part searches document attachments.
WHERE condition	WHERE condition used to limit the scope of the attachment search for the web part. In addition to specifying which documents have their attachments searched, you can also use the columns of the **CMS_Attachment** table to search only attachments of a specific type, for example: *AttachmentExtension = '.txt'*
ORDER BY expression	ORDER BY expression that determines the order of documents retrieved by the attachment search in the results.

When users perform a search and the system finds a match in the attachment of a document, the given document is added to the search results. The attachment results are always interlaced with the other results provided by the specified smart search indexes. This behavior is by design and cannot be modified.

The attachment search is performed by the SQL server, so it is not affected by the settings and

restrictions of the used search indexes. To limit the attachment search scope, enter an appropriate value into the **WHERE condition** property of the used web part. For example, if you have a search results web part using a document index that is limited to the */News/%* section of your website, you need to add the following **WHERE condition** to ensure that the attachment search is also restricted to these documents: *NodeAliasPath LIKE '/News/%'*

Please note

The search only returns documents if they are directly connected to the matching attachment through one of the following methods:

- Attachment files added to documents through fields that have their **Field type** set to ***File*** or ***Document attachments*** in the document type definition.
- Attachments uploaded in CMS Desk on the **Properties -> Attachments** tab of documents.

8.43.11 Search results in transformations

You can use the following default transformations to display search results using the **Smart search dialog with results** and **Smart search results** web parts:

- **CMS.Root.SmartSearchResults**
- **CMS.Root.SmartSearchResultsWithImages**

Search results are returned in a so-called **search dataset**. No matter how the fields are named in the found objects, the search dataset always contains the following fields that are automatically mapped to the corresponding object fields:

score	Expresses the relevance of the found document in a numeric value ranging from 0 to 1, where higher values indicate higher relevance.
title	This field is mapped to the field specified by the **Title field** drop-down list on the **Search fields** tab for the particular object.
content	This field is mapped to the field specified by the **Content field** drop-down list on the **Search fields** tab for the particular object.
created	This field is mapped to the field specified by the **Date field** drop-down list on the **Search fields** tab for the particular object.
image	This field is mapped to the field specified by the **Image field** drop-down list on the **Search fields** tab for the particular object.

In transformations, you can get the values from the dataset by using the **Eval("<field name>")** function: *Eval("score")*, *Eval("title")*, etc.

You can also use the following functions in your transformations:

- **SearchResultUrl(bool absolute)** - returns the URL of the page where the details of the search result can be found. The parameter indicates if the returned URL should be absolute.

Search result URLs for general index results

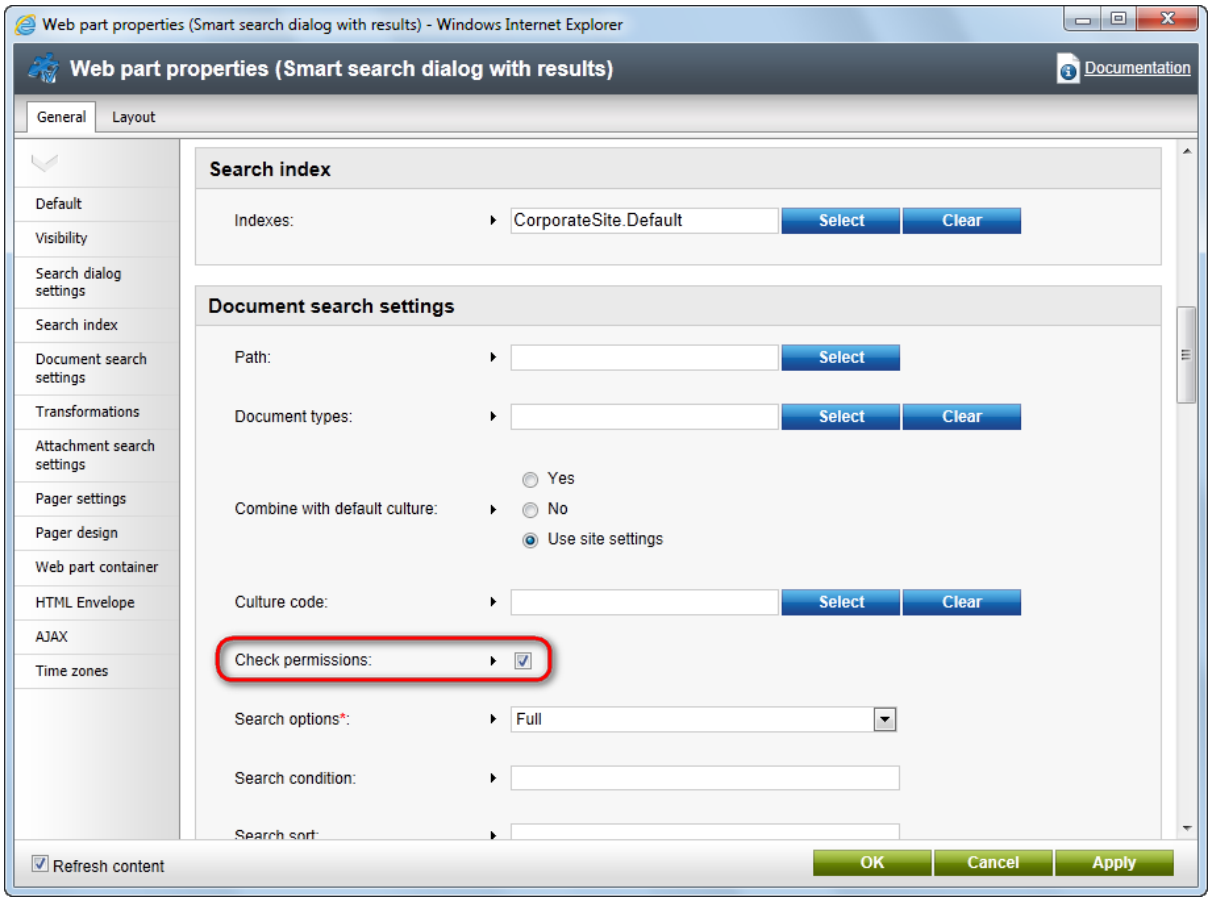
This method does not return valid URLs for search results produced by a general index, since the indexed objects are not documents and there is no default page where the details are displayed.

A custom transformation function must be written and used in order to generate the correct URL of a custom page displaying the appropriate information.

- **SearchHighlight(string text, string startTag, string endTag)** - wraps the text entered in the first attribute into the tags specified by the other two attributes.
- **GetSearchImageUrl(string noImageUrl, int maxSideSize)** - returns the URL of the current search result image. The first attribute specifies the URL returned if no image is found, the second one specifies the maximum side size to which the image will be resized.
- **GetSearchValue(string columnName)** - returns the value of the specified field for the current search result. This may be used to access both the general fields of the given object type (document, custom table etc.) and any other fields included in the search index.

8.43.12 Security

Smart search web parts have the **Check permissions** property. If this option is enabled, pages for which the current user does not have read permissions will not be displayed in search results.



8.43.13 SQL search overview

The SQL search is an obsolete search engine for documents in the content tree of websites (pages). We highly recommend using the index-based Smart search instead. The system continues supporting the SQL search for the following reasons:

- Backward compatibility with older versions
- Searching files uploaded as document attachments

This search engine uses standard SQL queries to search for expressions:

- The system automatically generates queries for the data of individual document types. To override the search query for a document type, create a new query named **searchtree** in *Site Manager -> Development -> Document types -> Edit document type -> Queries*.
- The **cms.root.searchdocuments** query searches fields that are shared by all documents (such as the document name).
- The **cms.root.searchattachments** query searches files uploaded as document attachments. To search attachments, you need to configure the system as described in Configuration of full-text search in files.

Adding the SQL search onto your website

To integrate the SQL search into website pages, use the web parts in the **Full-text search** category

(SQL search dialog, SQL search box, etc.) or the **CMSSearchDialog** and **CMSSearchResults** server controls (in your ASP.NET code).

Excluding documents from search

You can **exclude document types** by setting the **Exclude document types from SQL search** value in **Site Manager -> Settings -> System -> Search**. You need to enter the code name, such as *cms.article*. You can enter multiple document types separated by semicolons (;).

You can **exclude website sections** from the search by setting the **Exclude documents from SQL search** value in **Site Manager -> Settings -> System -> Search**:

- To exclude a single document, enter the alias path, for example: */news/news1*
- To exclude entire website sections, enter a path expression, for example: /news/%.

You can enter multiple values separated by semicolons (;).

To directly exclude individual documents from the SQL search:

1. Select the document in the content tree in CMS Desk.
2. Go to **Edit -> Properties -> Navigation**.
3. Check **Exclude from search**.
4. Click **Save**.

Modifying the search result format

If you wish to modify the format of the SQL search results, modify the **searchresults** transformation in **Site Manager -> Development -> Document Types -> edit ( ) Root -> Transformations**.

Development of Custom Search Provider

If you need to integrate a custom search engine or make additional modifications to the search results returned by the SQL search engine, you can develop your own search provider. You can find more details in API programming and Kentico CMS internals-> Custom providers -> Custom Search Provider.

8.43.14 Smart search internals and API

8.43.14.1 Overview

In this chapter, you will learn which database tables and API classes are used in the Smart search module. You will also see the most common API examples.

Further API-related information and examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all members of the module's classes, please refer to Kentico CMS API Reference.

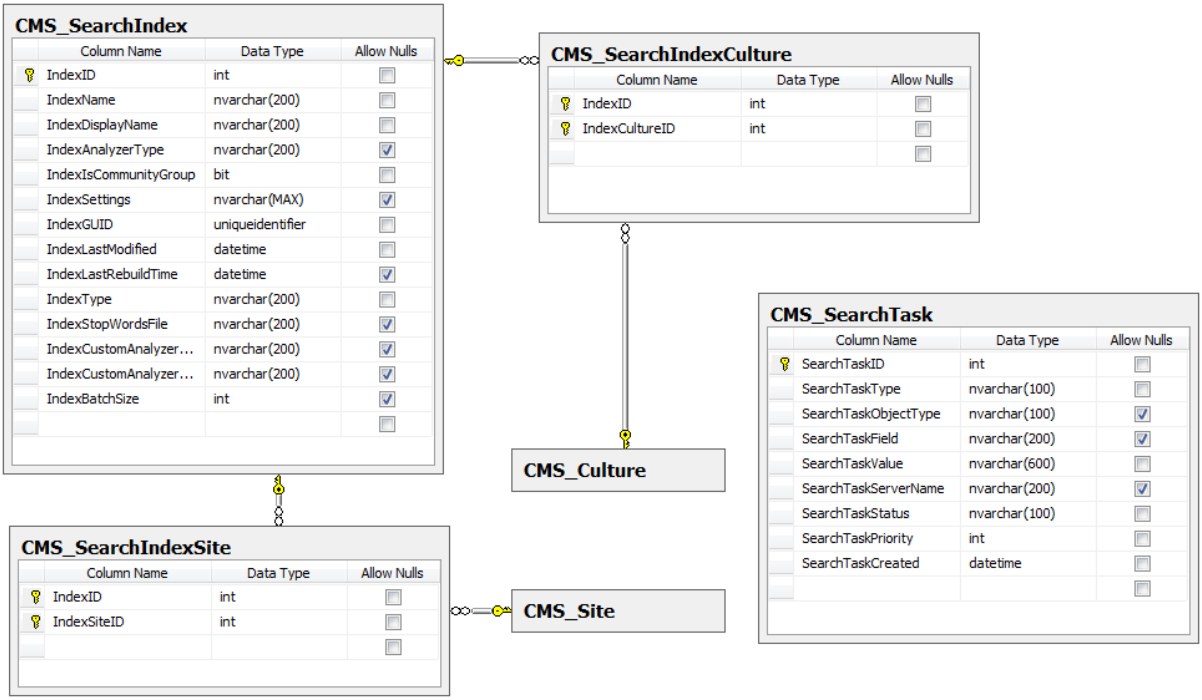
8.43.14.2 Database tables

The following database tables are used to store information for the smart search module:

- **CMS_SearchIndex** - contains records representing smart search indexes.
- **CMS_SearchIndexSite** - stores relationships between smart search indexes and sites. Each entry

in this table indicates that an index is assigned to a site.

- **CMS_SearchIndexCulture** - stores relationships between indexes and cultures. Each entry indicates that documents of the specified culture should be included in the given index.
- **CMS_SearchTask** - stores smart search indexing tasks.



8.43.14.3 API classes

If you are not familiar with database table data management through Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

The classes of the Smart search module can be found in the **CMS.SiteProvider** namespace.

CMS_SearchIndex table API:

- **SearchIndexInfo** - represents one search index.
- **SearchIndexInfoProvider** - provides management functionality for search indexes.

CMS_SearchIndexSite table API:

- **SearchIndexSiteInfo** - represents a relationship between a search index and a site.
- **SearchIndexSiteInfoProvider** - provides management functionality for site-index relationships.

CMS_SearchIndexCulture table API:

- **SearchIndexCultureInfo** - represents a relationship between a search index and a culture.
- **SearchIndexCultureInfoProvider** - provides management functionality for index-culture relationships.

CMS_SearchTask table API:

- **SearchTaskInfo** - represents one indexing task.
- **SearchTaskInfoProvider** - provides management functionality for smart search indexing tasks.

Other classes:

- **SearchHelper** - provides general smart search functionality and data.

- **CMS.SettingsProvider.SearchIndexSettingsInfo** - represents the settings that can be configured for a search index.
- **CMS.SettingsProvider.SearchIndexSettings** - used to manage *SearchIndexSettingsInfo* objects and assign them to indexes.
- **CMS.SettingsProvider.SearchSettingsInfo** - represents the search field settings of an object.
- **CMS.SettingsProvider.SearchSettings** - provides management functionality for *SearchSettingsInfo* objects.

8.43.14.4 API examples

8.43.14.4.1 Overview

These topics show examples of how the API of the Smart search module can be used:

- Managing search indexes
- Managing search index sites
- Managing search index cultures
- Performing indexing and search actions

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Administration\Search\Default.aspx.cs***.

The smart search API examples use the following namespaces:

```
using System;
using System.Data;
using System.Configuration;
```

```
using CMS.GlobalHelper;
using CMS.CMSHelper;
using CMS.SiteProvider;
using CMS.SettingsProvider;
using CMS.DocumentEngine;
```

8.43.14.4.2 Managing search indexes

The following example creates a smart search index.

```
private void CreateSearchIndex()
{
    // Create new search index object
    SearchIndexInfo newIndex = new SearchIndexInfo();

    // Set the properties
    newIndex.IndexDisplayName = "My new index";
    newIndex.IndexName = "MyNewIndex";
    newIndex.IndexIsCommunityGroup = false;
    newIndex.IndexType = PredefinedObjectType.DOCUMENT;
    newIndex.IndexAnalyzerType = AnalyzerTypeEnum.StandardAnalyzer;
    newIndex.StopWordsFile = "";

    // Save the search index
    SearchIndexInfoProvider.SetSearchIndexInfo(newIndex);
}
```

The following example creates and configures the settings of a search index.

```
private bool CreateIndexSettings()
{
    // Get the search index
    SearchIndexInfo index = SearchIndexInfoProvider.GetSearchIndexInfo
("MyNewIndex");
    if (index != null)
    {
        // Create new index settings
        SearchIndexSettingsInfo indexSettings = new SearchIndexSettingsInfo();

        // Set setting properties
        indexSettings.IncludeBlogs = true;
        indexSettings.IncludeForums = true;
        indexSettings.IncludeMessageCommunication = true;
        indexSettings.ClassNames = ""; // for all document types
        indexSettings.Path = "/%";
        indexSettings.Type = SearchIndexSettingsInfo.TYPE_ALLOWED;
        indexSettings.ID = Guid.NewGuid();

        // Save index settings
        SearchIndexSettings settings = new SearchIndexSettings();
```

```
        settings.SetSearchIndexSettingsInfo(indexSettings);
        index.IndexSettings = settings;

        // Save to database
        SearchIndexInfoProvider.SetSearchIndexInfo(index);

        return true;
    }

    return false;
}
```

The following example gets and updates a search index.

```
private bool GetAndUpdateSearchIndex()
{
    // Get the search index
    SearchIndexInfo updateIndex = SearchIndexInfoProvider.GetSearchIndexInfo
("MyNewIndex");
    if (updateIndex != null)
    {
        // Update the properties
        updateIndex.IndexDisplayName = updateIndex.IndexDisplayName.ToLower();

        // Save the changes
        SearchIndexInfoProvider.SetSearchIndexInfo(updateIndex);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates search indexes.

```
private bool GetAndBulkUpdateSearchIndexes()
{
    // Prepare the parameters
    string where = "IndexName LIKE N'MyNewIndex%'";

    // Get the data
    DataSet indexes = SearchIndexInfoProvider.GetSearchIndexes(where, null);
    if (!DataHelper.DataSourceIsEmpty(indexes))
    {
        // Loop through the individual items
        foreach (DataRow indexDr in indexes.Tables[0].Rows)
        {
            // Create object from DataRow
            SearchIndexInfo modifyIndex = new SearchIndexInfo(indexDr);

            // Update the properties
```

```
            modifyIndex.IndexDisplayName = modifyIndex.IndexDisplayName.ToUpper();

            // Save the changes
            SearchIndexInfoProvider.SetSearchIndexInfo(modifyIndex);
        }

        return true;
    }

    return false;
}
```

The following example deletes a search index.

```
private bool DeleteSearchIndex()
{
    // Get the search index
    SearchIndexInfo deleteIndex = SearchIndexInfoProvider.GetSearchIndexInfo
("MyNewIndex");

    // Delete the search index
    SearchIndexInfoProvider.DeleteSearchIndexInfo(deleteIndex);

    return (deleteIndex != null);
}
```

8.43.14.4.3 Managing search index sites

The following example assigns a smart search index to a site.

```
private bool AddSearchIndexToSite()
{
    // Get the search index
    SearchIndexInfo index = SearchIndexInfoProvider.GetSearchIndexInfo
("MyNewIndex");
    if (index != null)
    {
        int indexId = index.IndexID;
        int siteId = CMSContext.CurrentSiteID;

        // Save the binding
        SearchIndexSiteInfoProvider.AddSearchIndexToSite(indexId, siteId);

        return true;
    }

    return false;
}
```

The following example removes a search index from a site.

```
private bool RemoveSearchIndexFromSite()
{
    // Get the search index
    SearchIndexInfo removeIndex = SearchIndexInfoProvider.GetSearchIndexInfo
("MyNewIndex");
    if (removeIndex != null)
    {
        int siteId = CMSContext.CurrentSiteID;

        // Get the binding
        SearchIndexSiteInfo indexSite =
SearchIndexSiteInfoProvider.GetSearchIndexSiteInfo(removeIndex.IndexID, siteId);

        // Delete the binding
        SearchIndexSiteInfoProvider.DeleteSearchIndexSiteInfo(indexSite);

        return true;
    }

    return false;
}
```

8.43.14.4.4 Managing search index cultures

The following example assigns a culture to a smart search index.

```
private bool AddCultureToSearchIndex()
{
    // Get the search index and culture
    SearchIndexInfo index = SearchIndexInfoProvider.GetSearchIndexInfo
("MyNewIndex");
    CultureInfo culture = CultureInfoProvider.GetCultureInfo("en-us");

    if ((index != null) && (culture != null))
    {
        // Save the binding
        SearchIndexCultureInfoProvider.AddSearchIndexCulture(index.IndexID,
culture.CultureID);

        return true;
    }

    return false;
}
```

The following example removes a culture from a search index.

```
private bool RemoveCultureFromSearchIndex()
{
```

```
    // Get the search index
    SearchIndexInfo removeIndex = SearchIndexInfoProvider.GetSearchIndexInfo
("MyNewIndex");
    CultureInfo culture = CultureInfoProvider.GetCultureInfo("en-us");

    if ((removeIndex != null) && (culture != null))
    {
        // Get the binding
        SearchIndexCultureInfo indexCulture =
SearchIndexCultureInfoProvider.GetSearchIndexCultureInfo(removeIndex.IndexID,
culture.CultureID);

        // Delete the binding
        SearchIndexCultureInfoProvider.DeleteSearchIndexCultureInfo(indexCulture);

        return true;
    }

    return false;
}
```

8.43.14.4.5 Performing indexing and search actions

The following example creates a task that rebuilds a smart search index.

```
private bool RebuildIndex()
{
    // Get the search index
    SearchIndexInfo index = SearchIndexInfoProvider.GetSearchIndexInfo
("MyNewIndex");

    if (index != null)
    {
        // Create rebuild task
        SearchTaskInfoProvider.CreateTask(SearchTaskTypeEnum.Rebuild,
index.IndexType, null, index.IndexName);

        return true;
    }

    return false;
}
```

The following example performs a search through the content of an index and retrieves the results in a dataset.

```
private bool SearchText()
{
    // Get the search index
    SearchIndexInfo index = SearchIndexInfoProvider.GetSearchIndexInfo
```

```
("MyNewIndex");

    int numberOfResults = 0;

    if (index != null)
    {
        // Set the properties
        string searchText = "home";
        string path = "/%";
        string classNames = "";
        string cultureCode = "EN-US";
        string defaultCulture = CultureHelper.DefaultCulture.IetfLanguageTag;
        Lucene.Net.Search.Sort sort = SearchHelper.GetSort("##SCORE##");
        bool combineWithDefaultCulture = false;
        bool checkPermissions = false;
        bool searchInAttachments = false;
        string searchIndexes = index.IndexName;
        int displayResults = 100;
        int startingPosition = 0;
        int numberOfProcessedResults = 100;
        UserInfo userInfo = CMSContext.CurrentUser;
        string attachmentWhere = "";
        string attachmentOrderBy = "";

        // Get search results
        DataSet ds = SearchHelper.Search(searchText, sort, path, classNames,
cultureCode, defaultCulture, combineWithDefaultCulture, checkPermissions,
searchInAttachments, searchIndexes, displayResults, startingPosition,
numberOfProcessedResults, userInfo, out numberOfResults, attachmentWhere,
attachmentOrderBy);

        // If found at least one item
        if (numberOfResults > 0)
        {
            return true;
        }
    }

    return false;
}
```

The following example creates a task that updates the content of a search index.

```
private bool UpdateIndex()
{
    // Tree provider
    TreeProvider provider = new TreeProvider
(CMS.CMSHelper.CMSContext.CurrentUser);

    // Get document of specified site, aliaspath and culture
    TreeNode node = provider.SelectSingleNode
(CMS.CMSHelper.CMSContext.CurrentSiteName, "/", "en-us");
```

```
    // If node exists
    if ((node != null) && (node.PublishedVersionExists) &&
(SearchIndexInfoProvider.SearchEnabled))
    {
        // Edit and save document node
        node.NodeDocType += " changed";
        node.Update();

        // Create update task
        SearchTaskInfoProvider.CreateTask(SearchTaskTypeEnum.Update,
PredefinedObjectType.DOCUMENT, SearchHelper.ID_FIELD, node.GetSearchID());

        return true;
    }

    return false;
}
```

8.44 Syndication (RSS, Atom, XML)

8.44.1 Overview

The Syndication module allows you to create syndication feeds of your site's content. The whole functionality is provided by a set of web parts stored under the **Syndication** web part category. These web parts generate feeds which conform to the RSS, Atom or general XML format. The name of the feed format generated by each web part is included in the name of the web part.

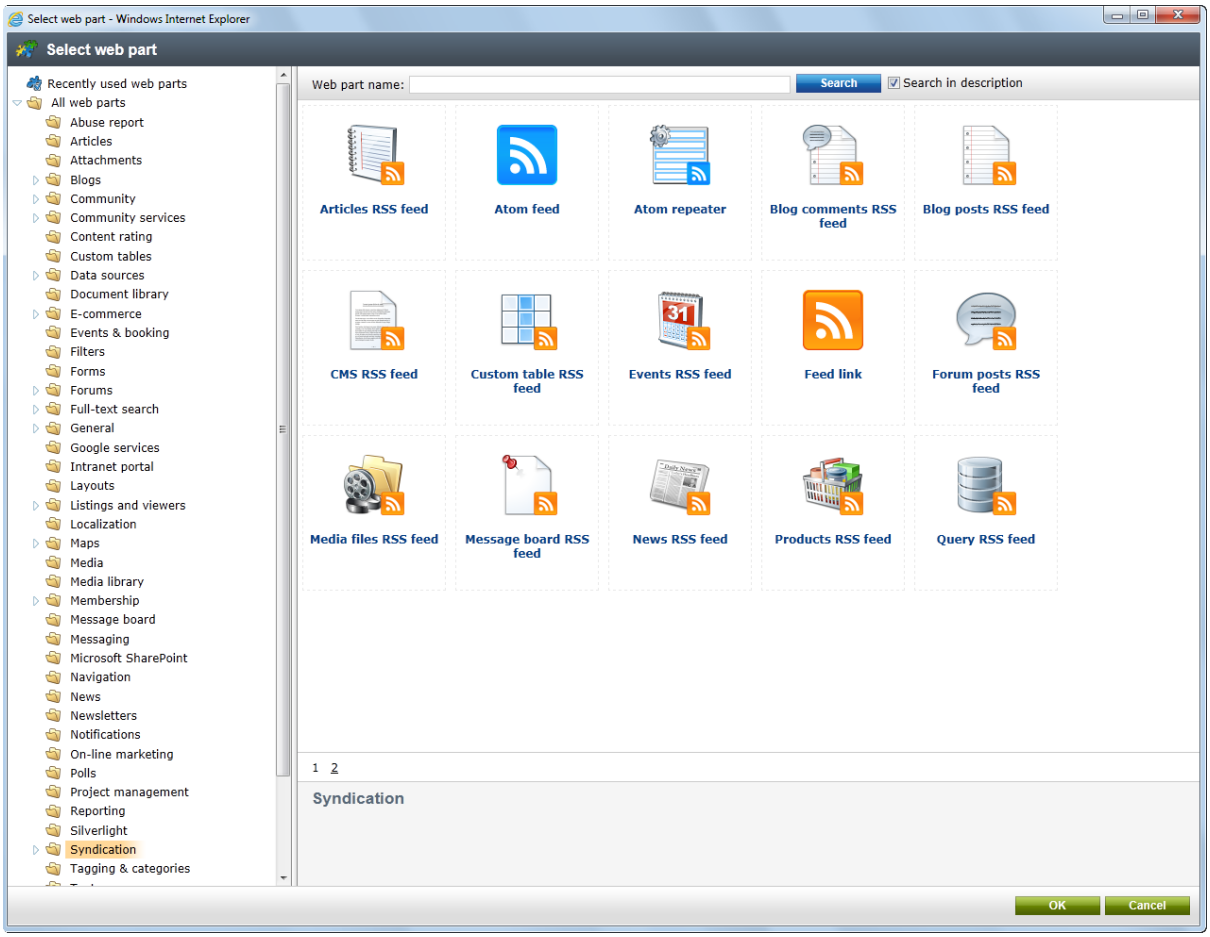
Detailed overview of the web parts can be found in the Syndication web parts and widgets topic. Use of transformations is essential with these web parts. Information related to use of transformations in syndication web parts can be found in the Syndication transformations topic.

We have also prepared three examples of typical usage:

- Example 1: Creating a **document RSS feed** using the *CMS RSS feed* web part.
- Example 2: Creating an **object RSS feed** using the *RSS feed* web part and a *data source*.
- Example 3: Creating a **dedicated page with an RSS feed** using the *RSS repeater* web part and a *data source*.
- Example 4: Displaying content of an **external RSS feed** on your website using the *RSS data source* and *Basic repeater* web parts.

Live usage examples of these web parts can be found on the sample **Corporate Site**, under the ***/Examples/Web-parts/Syndication/*** node.

All specific web part properties are explained in Kentico CMS Web Parts Reference or after clicking the **Documentation** ( ) link in the top right corner of the *Web part properties* dialog.



Creating RSS feeds manually

Due to backward compatibility with Kentico CMS versions prior to 5.5, it is still possible to create a page with an RSS feed manually. Even though it is not recommended to take this approach, you can learn more about it in this chapter.

8.44.2 Syndication web parts and widgets

Syndication web parts are stored under the **Syndication** category in the web part catalog. There is a large number of them, but most of them are similar and can be grouped into several categories:

From the functional point of view, there are two basic types of syndication web parts - repeaters and feeds:

- **Syndication feeds** show the feed icon with an optional text. The icon and text are clickable and when clicked, a querystring parameter with the feed ID is appended to its URL and the page renders the feed.
- **Syndication repeaters** change the page on which they are placed to a feed. If you place one of the syndication repeaters on a page, it changes the page response type to *application/xml*, transforms retrieved data to the required feed format and uses the transformed data as the content of the feed. This means that the page is no longer a standard HTML page - it becomes a syndication feed.

The web parts can also be divided in terms of what data they are used for. From this perspective, you can divide them as follows:

Basic feed web parts

These are universal web parts which can create a feed of data provided by a connected data source. Without a data source, the web parts are not functional. The data provided by the data source can be of any type, it just needs to be handled properly by the used transformation.

- **XML repeater**
- **RSS repeater**
- **Atom repeater**
- **RSS feed**
- **Atom feed**

Document feed web parts

These web parts are based on the **RSS feed** web part, but have a built-in data source for the **document types** in their names. This means that you don't need a connected data source - all you need is included in a single web part. They are a kind of "all-in-one" solutions for frequently used document types.

- **CMS RSS Feed**
- **Articles RSS Feed**
- **Blog posts RSS Feed**
- **Events RSS Feed**
- **News RSS Feed**

The **CMS RSS Feed** can even be used for **documents of any type**, which makes it a universal web part for any document feed.

Object feed web parts

Similarly to object feeds described above, object feeds also have a built-in data source, letting you create feeds from **Kentico CMS objects** by adding just a single "all-in-one" web part.

- **Blog comments RSS feed**
- **Custom tables RSS feed**
- **Media files RSS feed**
- **Message board RSS feed**
- **Products RSS feed**
- **Forum posts RSS feed**

Other syndication web parts

Custom feeds can be created using the following two web parts:

- **Query RSS feed** - generates a feed based on a custom database query and transformation.
- **Web service RSS feed** - transforms a dataset provided by a web service into an RSS feed.

The last web part does not crate any feed, but can be used as a link to a feed:

- **Feed link** - displays the RSS icon with a link leading to a URL set in its web part properties. Typically used for links to feeds created by syndication repeaters.

Syndication widgets

The default installation of Kentico CMS contains a set of widgets which are derived from the web parts listed above. They provide the same functionality and look, while only a limited set of properties can be configured. The following widgets are available:

- **Articles RSS Feed**
- **Blog comments RSS feed**
- **Blog posts RSS Feed**
- **Events RSS Feed**
- **Forum posts RSS feed**
- **Media files RSS feed**
- **News RSS Feed**
- **Products RSS feed**

Further information

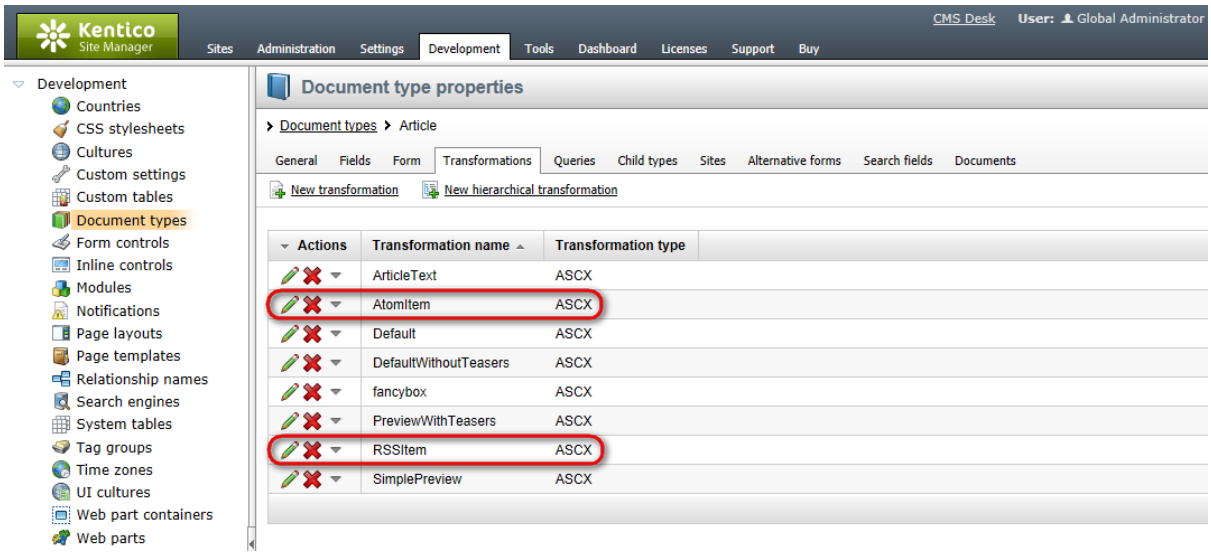
All specific properties of these web parts or widgets are explained in Kentico CMS Web Parts Reference or after clicking the **Documentation** ( ) link in the top right corner of the *Web part (widget) properties* dialog.

8.44.3 Syndication transformations

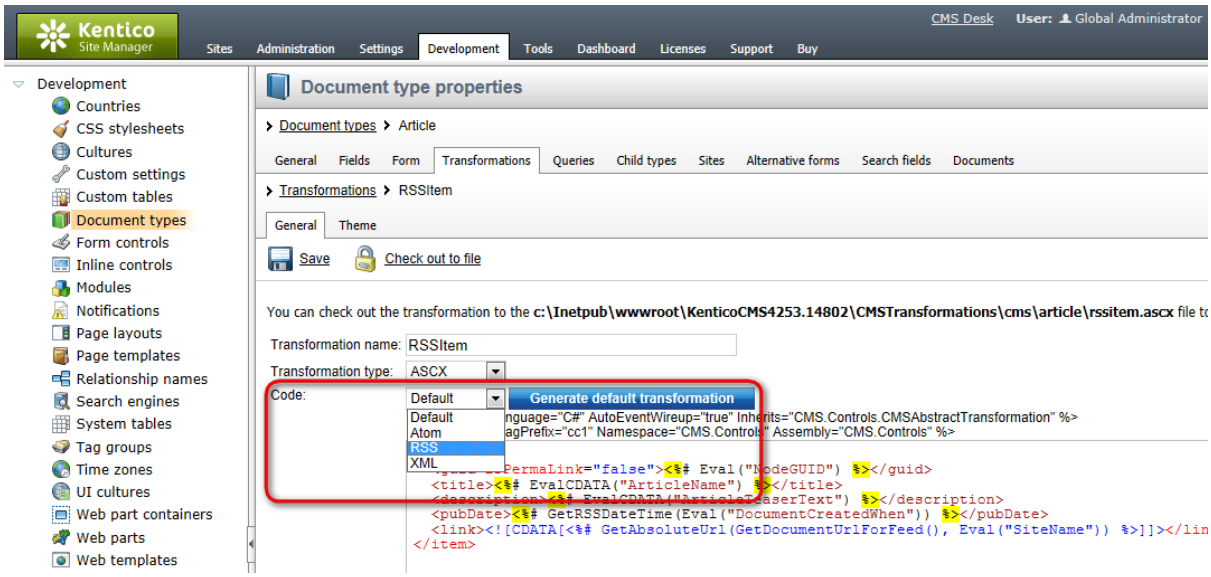
Use of transformations is of the highest importance when creating feeds, as all syndication web parts render the feeds using them. They need to be set in the **Transformation** property of each feed web part.

Default transformations

All document types in Kentico CMS have the **RSSItem** and **AtomItem** transformations. When you create a new document type, these transformations are created for it automatically.

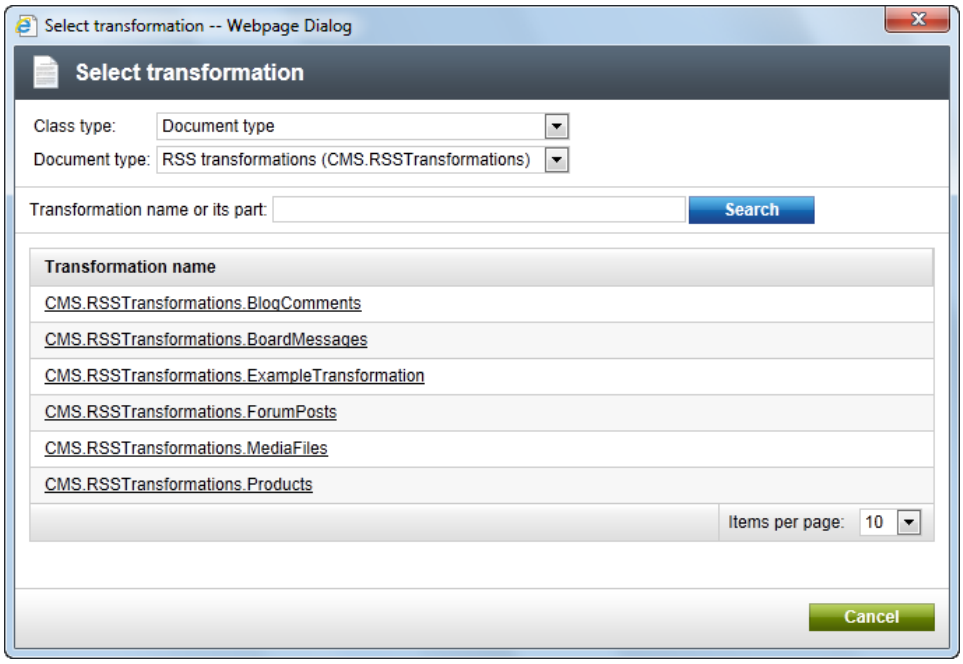


The default RSS, Atom and XML transformations can also be generated when editing or creating transformations of a document type or a custom table. To do this, you only need to select the required format from the **Code** drop-down list and click the **Generate default transformation** button.



RSS transformations document type

Other feed transformations are found in a special document type called **RSS transformations (CMS.RSSTransformations)**. This is a special document type used just to store transformations for **Kentico CMS objects**. Transformations for blog comments, board messages, forum posts, media library files and products are contained in the document type.



CDATA in feed transformations

If you need to include CDATA sections in your feeds, you may utilize the **EvalCDATA** method. This method works in a similar manner as the commonly used **Eval** method. The difference is that the retrieved string is wrapped in a CDATA section and all occurences of the **]]>** character sequence in the string are escaped.

Escaping is performed the way that each occurrence of **]]>** is replaced with **]]]]><![CDATA[>** and the whole text is then wrapped in standard **<![CDATA[**"*wrapped text*"**]]>** enclosure.

The method has two overrides:

- **EvalCDATA("FieldName")** - the first one has only one parameter - the actual name of the retrieved field.
- **EvalCDATA("FieldName", true)** - the second override has one extra boolean parameter, which determines if the wrapping will be performed. If the second parameter is set to *false*, value of the field will be retrieved, escaping of the **]]>** sequence will be performed, but the whole retrieved text will not be wrapped in the standard **<![CDATA[**"*wrapped text*"**]]>** enclosure.

The second override with the second parameter set to *false* is particularly useful if you use the method within another CDATA section. Because nesting of CDATA sections is not possible, you will only want to have the retrieved string escaped, but not wrapped in the enclosure. The code example below is a transformation extract where the method is used exactly this way.

```
...

<description>
  <![CDATA[
    <strong><%# EvalCDATA("CommentUserName",false) %></strong><br /><%# EvalCDATA
("CommentText",false) %>
```

```
  ]]>
</description>

...
```

8.44.4 Usage examples

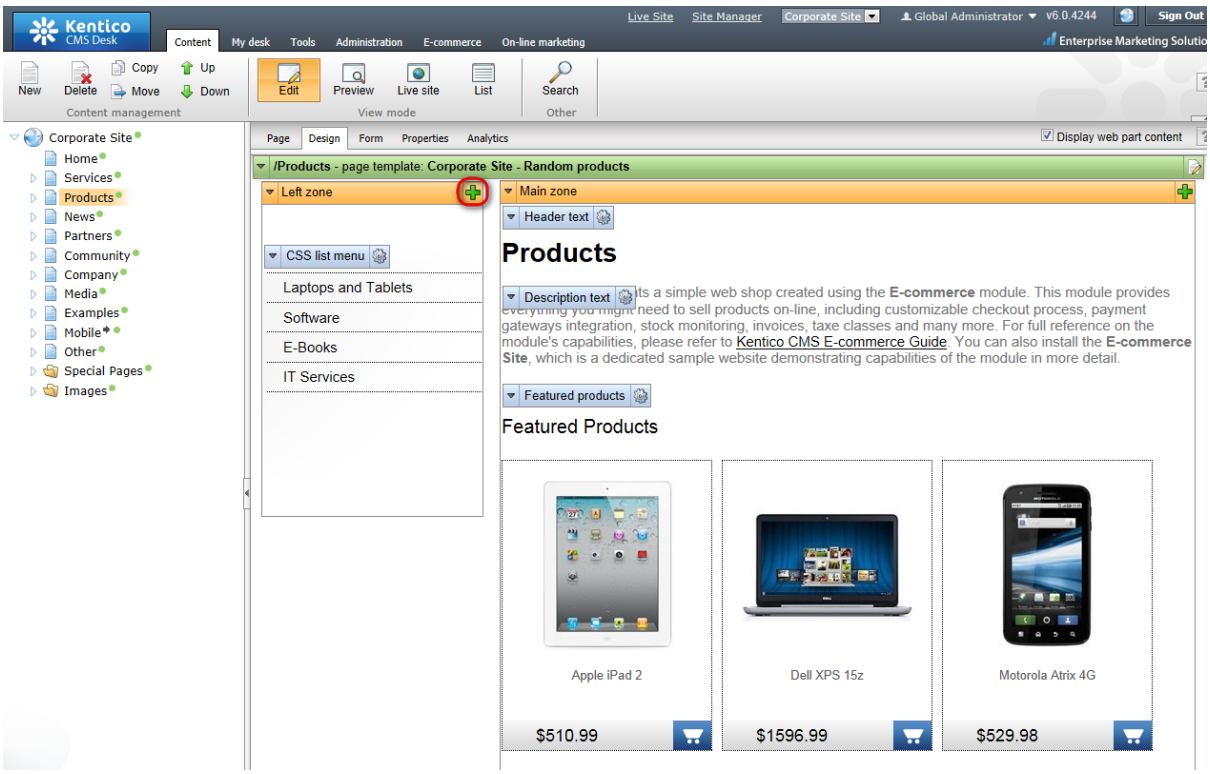
8.44.4.1 CMS RSS feed

The **CMS RSS feed** web part can be used to easily create RSS feeds from documents in the content tree. It has a **built-in Documents data source**, which means that the web part can work separately without any connected data source and that it can be used to create feeds from documents of any type.

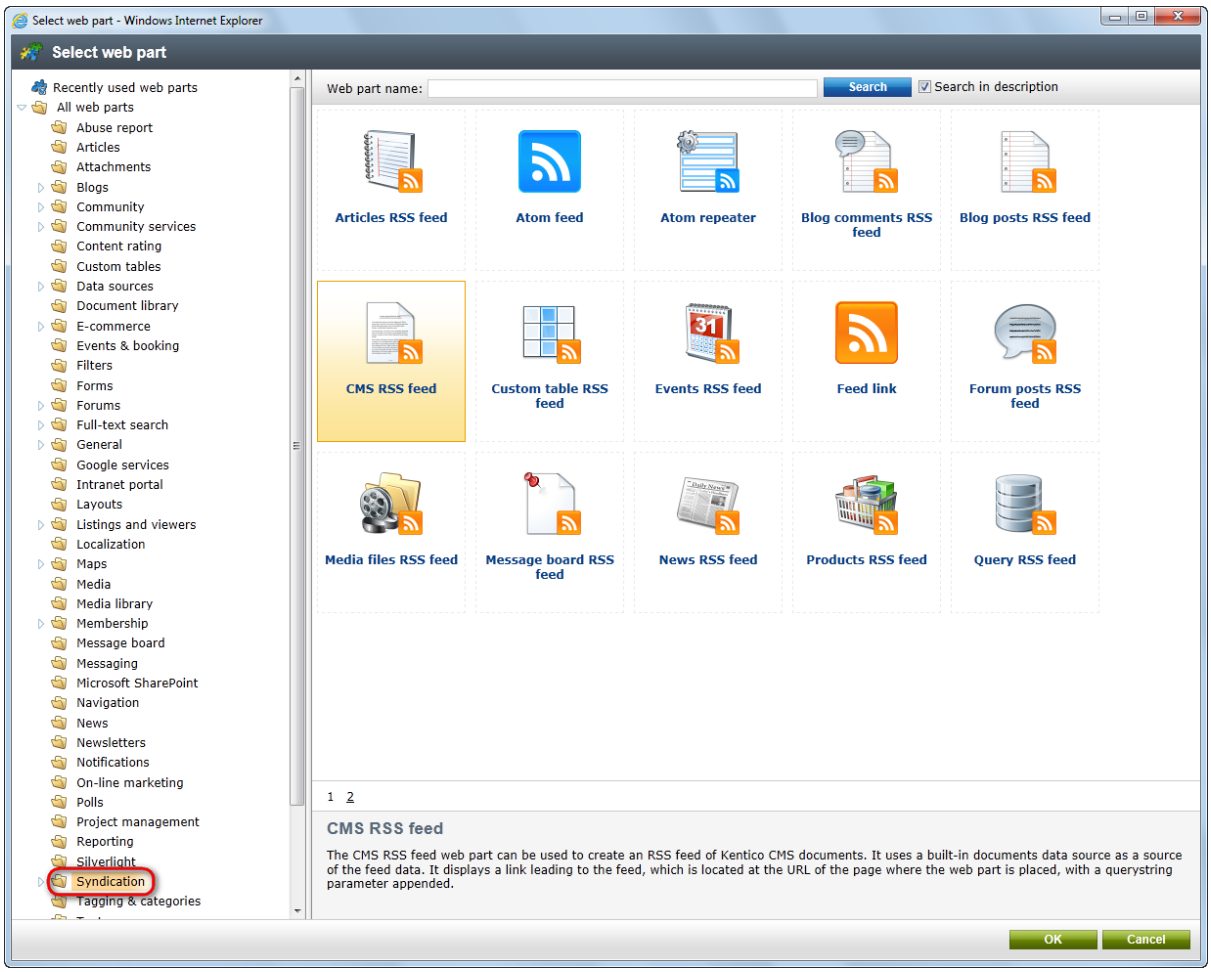
Apart from this universal web part, there are other web parts in the **Syndication** category which are pre-configured for one particular document type. See the listing in the Syndication web parts and widgets chapter.

In the following example, you will learn how to use this web part to create a feed link on the **/Products** page of the sample **Corporate site**. We will configure it so that it provides an RSS feed with products stored in all three categories under the page. The same procedure can be used on any other page and for documents of any other type, you just need to specify your required path, document type and transformation in the web part properties.

1. Sign in to CMS Desk and select the **/Products** page from the content tree. Switch to the **Design** tab and click the **Add web part** (+) icon of the *Left zone* web part zone.



2. In the **Select web part** dialog, select the **Syndication** category. Choose the **CMS RSS feed** web part and click **OK**.

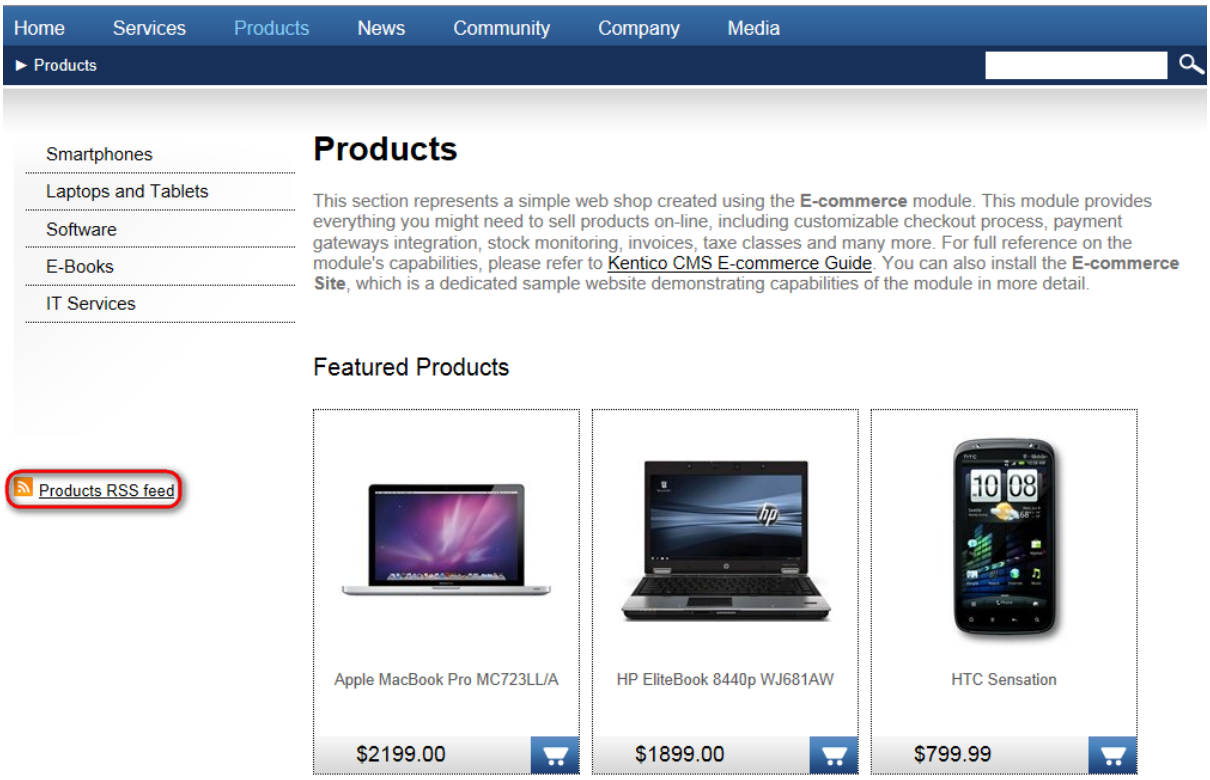


3. The **Web part properties** dialog opens. To achieve the required behavior, use the following configuration:

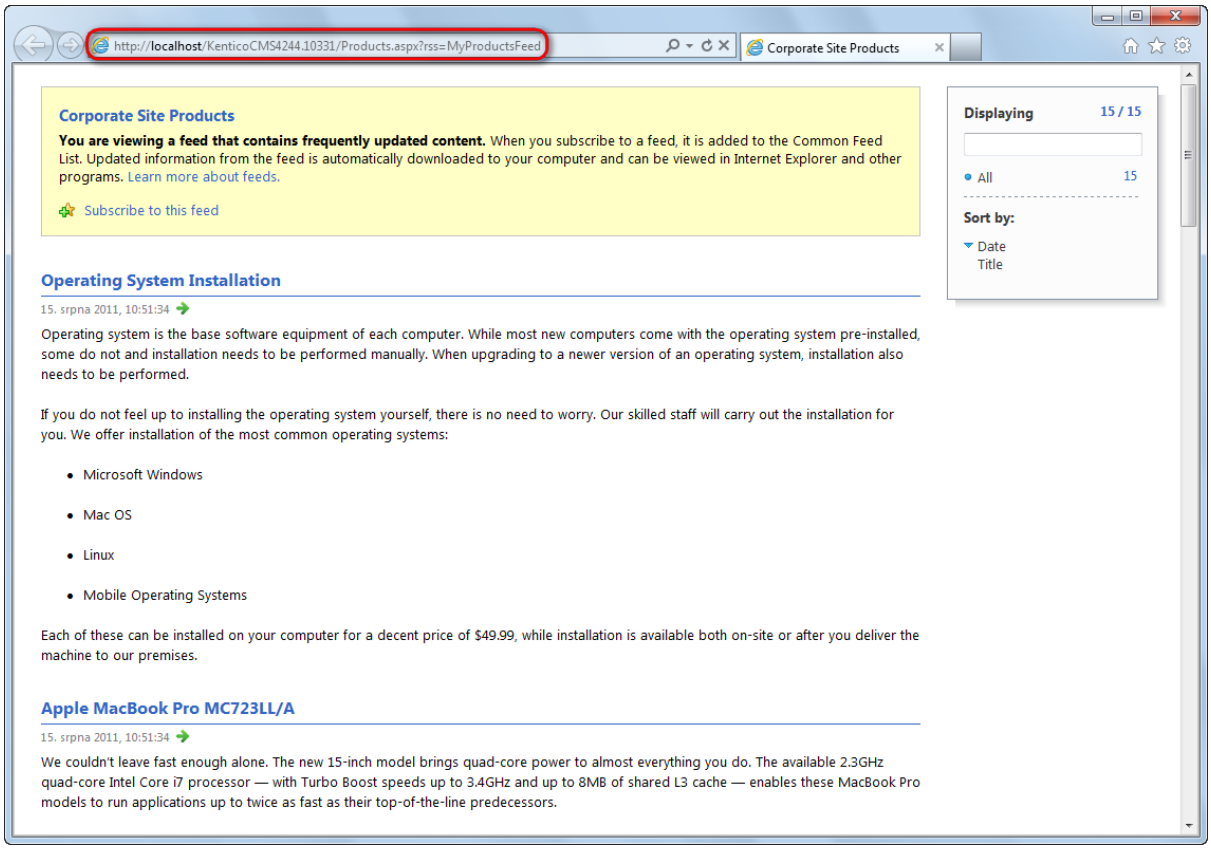
- **Link text**: *Products RSS feed*
- **Feed name**: *MyProductsFeed*
- **Feed title**: *Corporate Site Products*
- **Feed description**: *This is a sample RSS feed of products on the Corporate Site.*
- **Path**: */Products/%/%*
- **Transformation name**: *CMS.RSSTransformations.Products*

Leave default values for the rest of the properties and click **OK**.

4. Sign out of CMS Desk and browse to the **Products** page. You should see the RSS icon and link as highlighted in the screenshot below.



5. Click the icon or link, your browser will detect that you are accessing an RSS feed and display its content. As you can see, the URL of the feed is actually the URL of the page where the web part is placed with the **?rss=MyProductsFeed** querystring appended (this can be modified using the *Feed querystring key* and *Feed name* web part properties). The same URL can be used to access the feed directly both from a browser or a dedicated RSS reader program.



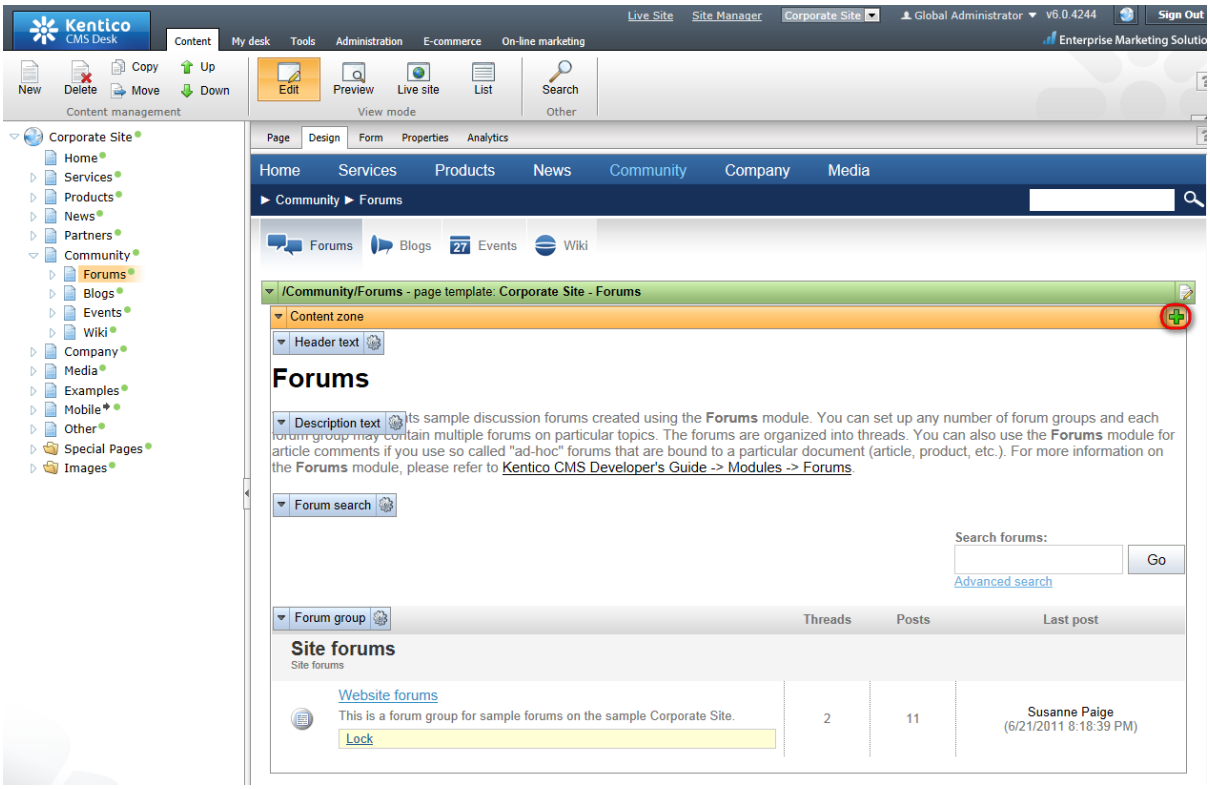
8.44.4.2 RSS feed + Data source

By connecting the **RSS feed** web part to a **data source**, you can create an RSS feed of the data provided by the data source. The **Atom feed** web part can be used exactly the same way, but the rendered feed is in the *Atom* format. You can use this approach to create feeds from Kentico CMS objects (forum posts, board comments, media files, ...).

The following example demonstrates how to create an RSS feed containing forum posts on the sample Corporate site. You can achieve this using the **RSS feed** and **Forum posts data source** web parts. The same procedure can be used for any other data. You only need to use a properly configured data source and process the provided data using a suitable transformation.

> **Please note**
>
> The result of the following example can also be achievable using a single web part - the **Forum posts RSS feed** web part, which has a built-in *Forum posts data source*.
>
> Similarly, there are all-in-one web parts for other frequently used document types and objects (see the list in Syndication web parts and widgets).

1. Sign in to CMS Desk and select the **Community -> Forums** page from the content tree. Switch to the **Design** tab and click the **Add web part** (✚) icon of the *Content zone* web part zone.

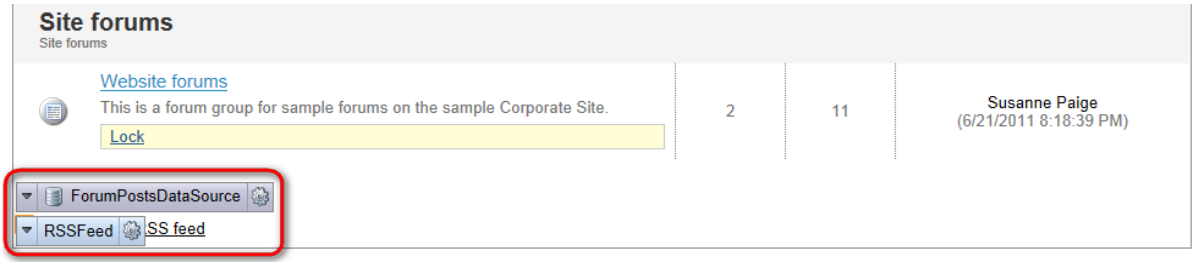


2. Select the **Data sources/Forum posts data source** web part and click **OK**. Leave all its properties at the default values, which will result in the data source providing all forum posts from all forum groups on the site. Click **OK**. The data source will be added to the bottom of *Content zone*.

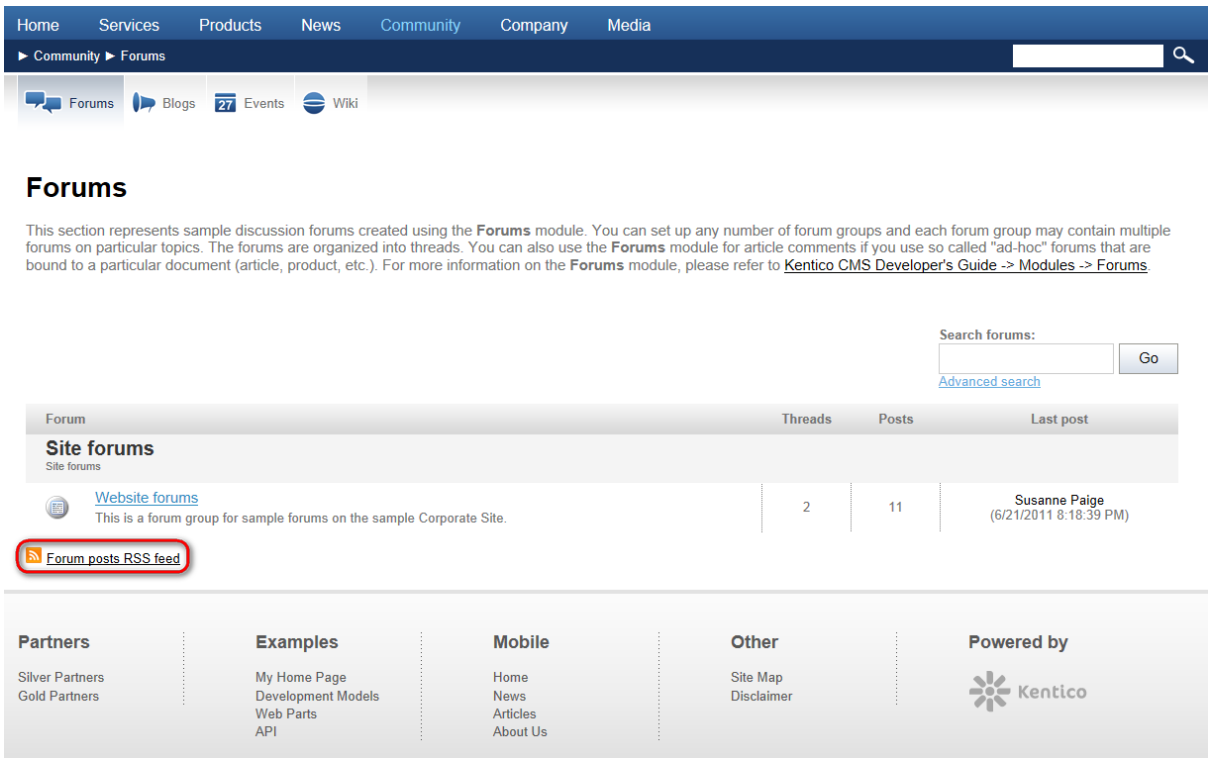
3. Click the **Add web part** () icon again and select the **Syndication/RSS feed** web part. Click **OK** and enter the following in the *Web part properties* dialog:

- **Link text**: *Forum posts RSS feed*
- **Feed name**: *MyForumPostsFeed*
- **Feed title**: *Corporate Site Forum Posts*
- **Feed description**: *This is a sample feed of all forum posts on the Corporate Site.*
- **Data source name**: *ForumPostsDataSource*
- **Transformation name**: *CMS.RSSTransformations.ForumPosts*

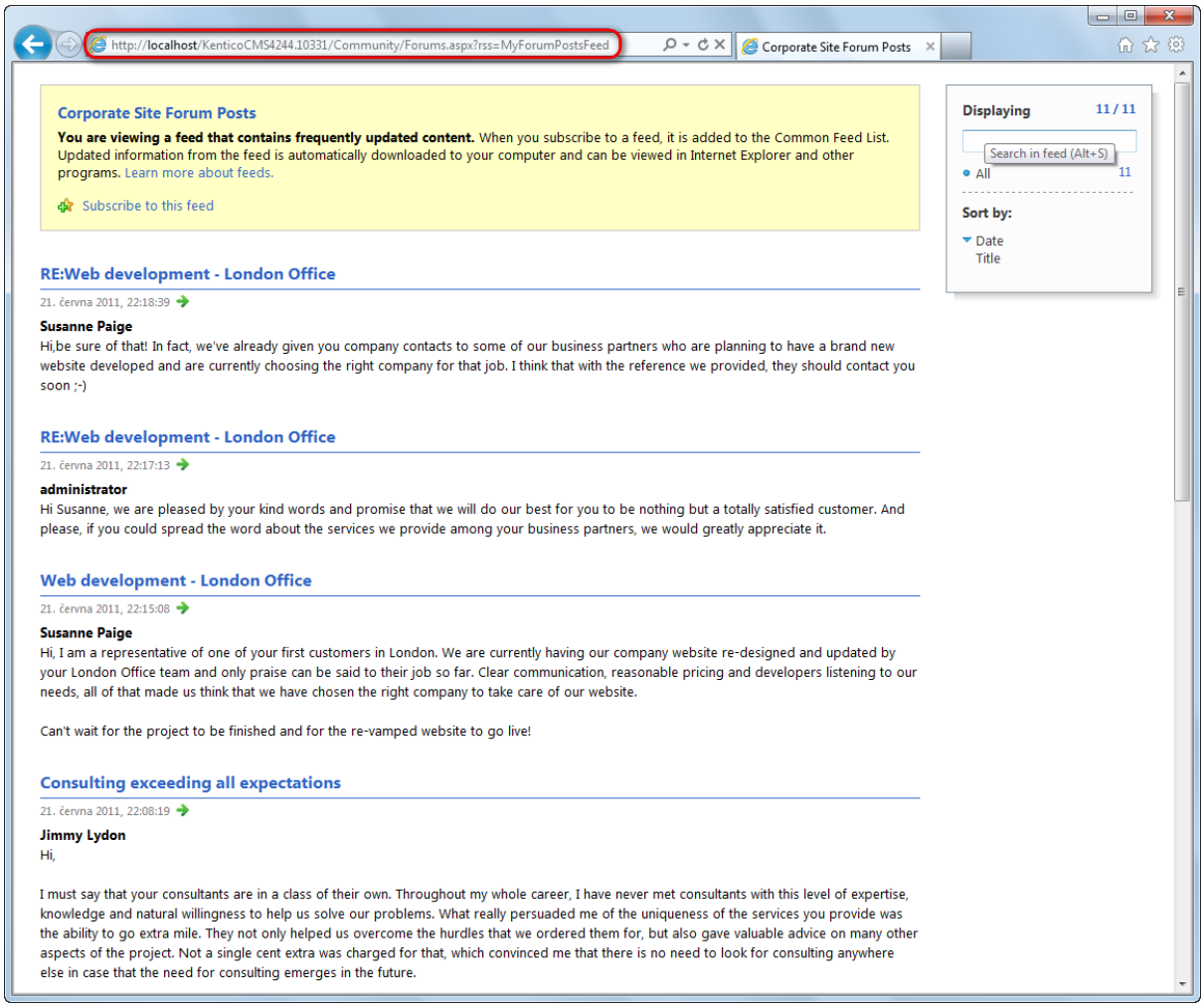
Leave defaults for the rest of the properties and click **OK**. The two web parts should now be at the bottom of the web part zone and fully configured for the required behavior.



4. Sign out of CMS Desk and browse to the **Forums** page. You should see the RSS icon and link as highlighted in the screenshot below.



5. Click the icon or link, your browser will detect that you are accessing an RSS feed and display its content. As you can see, the URL of the feed is actually the URL of the page where the web part is placed with the **?rss=MyForumPostsFeed** querystring appended (this can be modified using the *Feed querystring key* and *Feed name* web part properties). The same URL can be used to access the feed directly both from a browser or a dedicated RSS reader program.



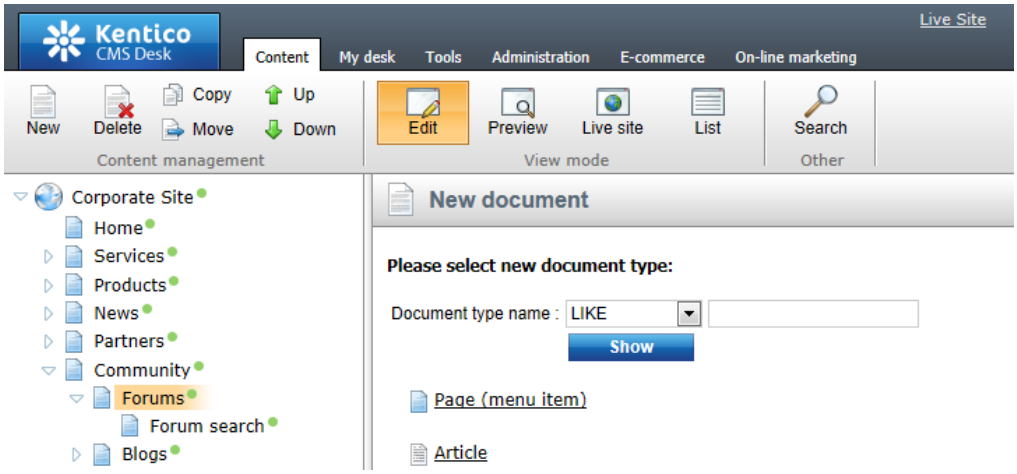
8.44.4.3 RSS repeater + Data source

The **RSS repeater** web part allows you to transform pages into RSS feeds. When a page containing the web part is accessed, the response type is changed to *application/xml* and the page renders the content of the feed. This is useful if you want your feed to have a dedicated URL, without the need for a querystring parameter. The **Atom repeater** and **XML repeater** web parts can be used exactly the same way, but the rendered feed is in the *Atom* or *XML* format.

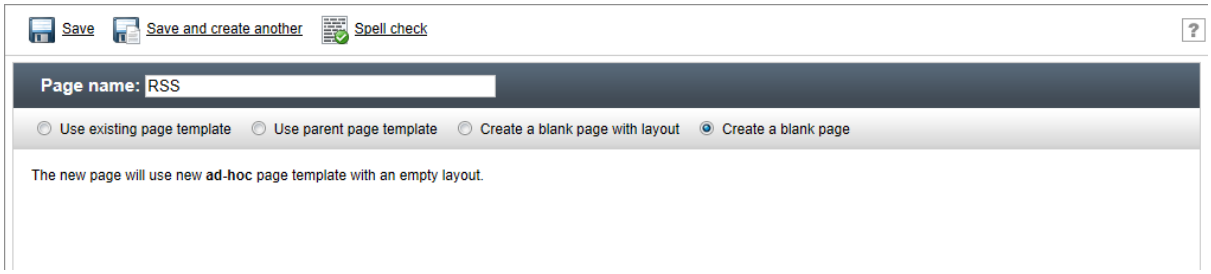
The following example demonstrates how to create an RSS feed containing the data of forum posts on the sample Corporate site. The example uses a dedicated feed page containing the **RSS repeater** web part.

> **Please note**
>
> The result of the following example can also be achieved using a single web part - the **Forum posts RSS feed** web part, which has a built-in Forum posts data source. Similarly, there are all-in-one web parts for other frequently used document types and objects (see the list in Syndication web parts and widgets).

1. Sign in to CMS Desk and select the **Forums** page from the content tree. Click **New** and choose the **Page (menu item)** document type.



2. Choose the **Create a blank page** option. Enter *RSS* as the **Page name** and click **Save**.



3. The page opens in **Design** mode. Click the **Add web part** (✚) icon of the only web part zone on the page. Select the **Data sources/Forum posts data source** web part and click **OK**. Leave all properties at the default values, which causes the data source to load all forum posts from all forum groups on the site. Click **OK**.

4. Click the **Add web part** (✚) icon again and select the **Syndication/RSS repeater** web part. Click **OK** and enter the following in the *Web part properties* dialog:

- **Feed name**: *MyForumPostsFeed*
- **Feed title**: *Corporate Site Forum Posts*
- **Feed description**: *This is a sample feed of all forum posts on the Corporate Site.*
- **Data source name**: *ForumPostsDataSource*
- **Transformation name**: *CMS.RSSTransformations.ForumPosts*

Leave defaults for the rest of the properties and click **OK**.

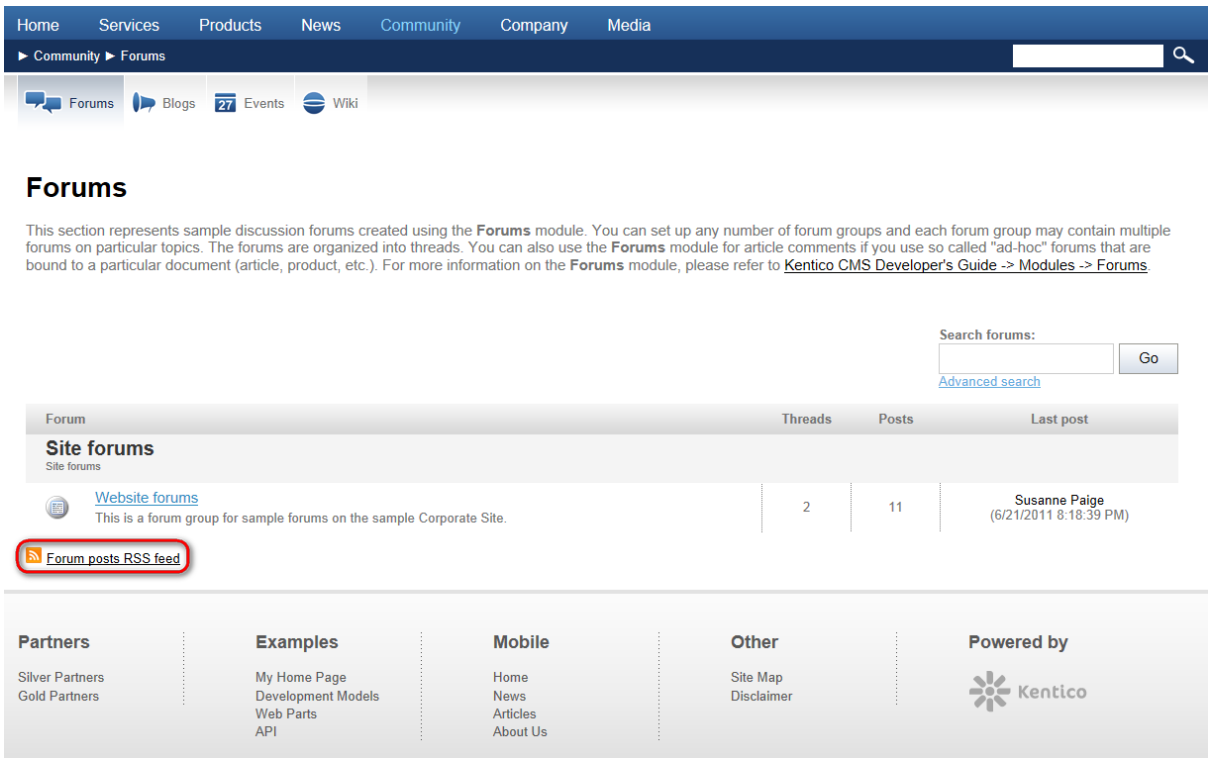
5. The RSS feed is now ready. To allow visitors to access the feed, create a link using the **Feed link** web part, which displays the RSS icon with a link leading to a URL specified in its properties.

Select the **/Forums** page in the content tree, switch to the **Design** mode and click the **Add web part** ( ) icon of the *Content zone* web part zone. Select **Syndication/Feed link** and click **OK**. Enter the following properties:

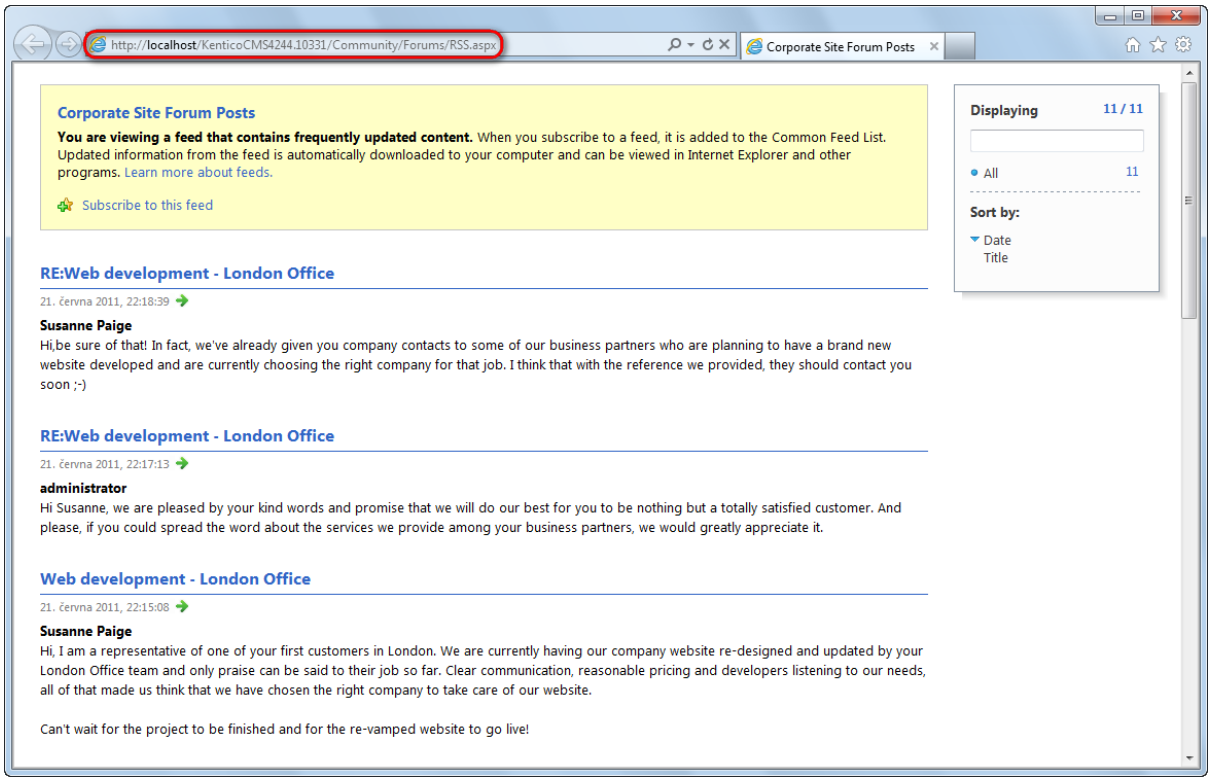
- **Link text**: *Forum posts RSS feed*
- **Feed URL**: *~/Community/Forums/RSS.aspx*
- **Feed title**: *My Forum Posts Feed*

Leave defaults for the rest of the properties and click **OK**.

6. Sign out of CMS Desk and browse to the **Forums** page. You should see the RSS icon and link.



7. Click the icon or link. Your browser detects that you are accessing an RSS feed and displays the content. The URL of the feed matches the URL of the page containing the **RSS repeater** web part. You can use the same URL to access the feed from both browsers and dedicated RSS reader programs.

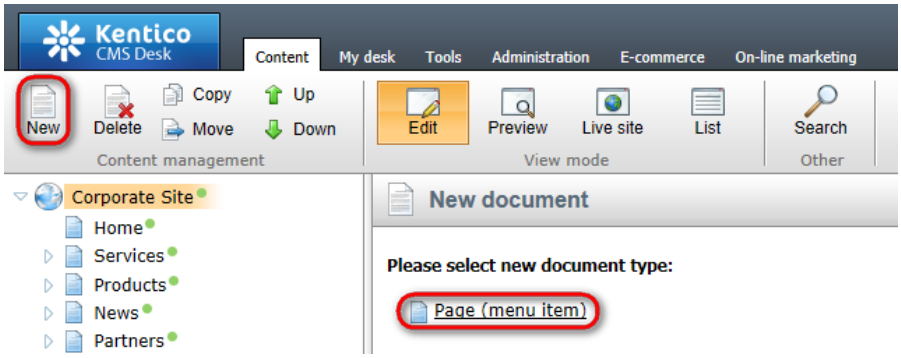


8.44.4.4 External RSS feed

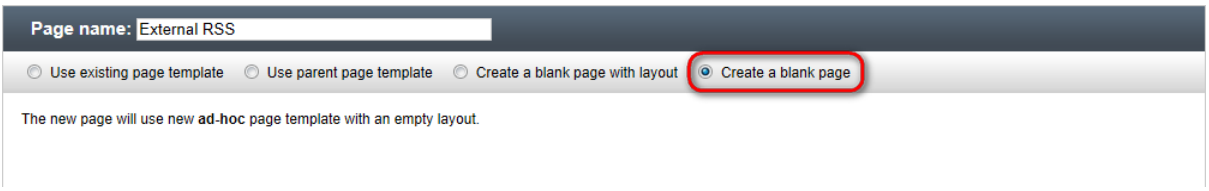
Apart from creating RSS feeds from the CMS content, it is also possible to use external RSS feeds (i.e. feeds published on other websites) as a source of data and display the data on your website.

In the following example, you will learn how to use the **RSS data source** and **Basic repeater** web parts to display content of an external feed on your website. For the purpose of the example, we will use the main blog posts feed from Kentico DevNet: http://devnet.kentico.com/CMSPages/BlogRss.aspx

1. Sign in to CMS Desk and select the root of your website's content tree. Click **New** above the content tree and choose to create a document of the **Page (menu item) type**.



2. In the following dialog, choose to **Create a blank page**, enter *External RSS* into the **Page name** field and click **Save**.



3. The page will get created and you will see it selected in the content tree. View the page on the **Design** tab and click the **Add web part** () icon in the top right corner of the only web part zone on the page. In the web part selection dialog, choose the **Data sources -> RSS data source** web and click **OK**. In its **Web part properties** dialog which pops-up, enter the following values:

- **Web part control ID**: *KenticoBlogsRSS*; this ID will be used in step 4 to connect the repeater that will display the data.
- **RSS feed URL**: *http://devnet.kentico.com/CMSPages/BlogRss.aspx*; this is the URL of the external RSS feed that you want to get data from.

Leave the rest of the properties at their default values and click **OK**.

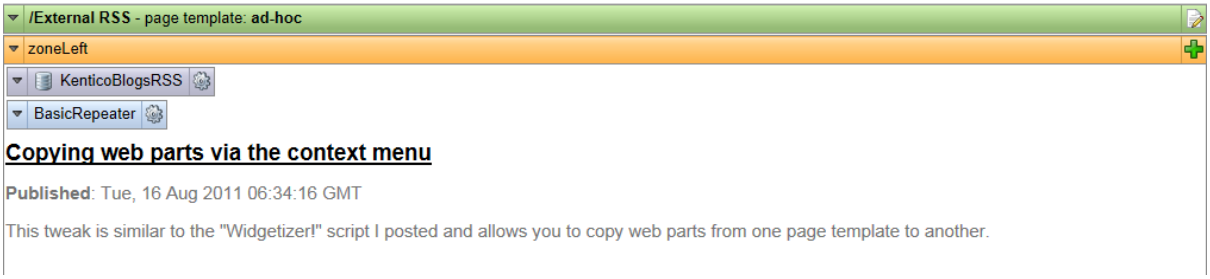
4. Click the **Add web part** () icon again. This time, choose the **Listings and viewers -> Basic repeater** web part and configure its properties as follows:

- **Data source name**: *KenticoBlogsRSS*;
- **Transformation name**: click **New** and create a new transformation with the code listed below.

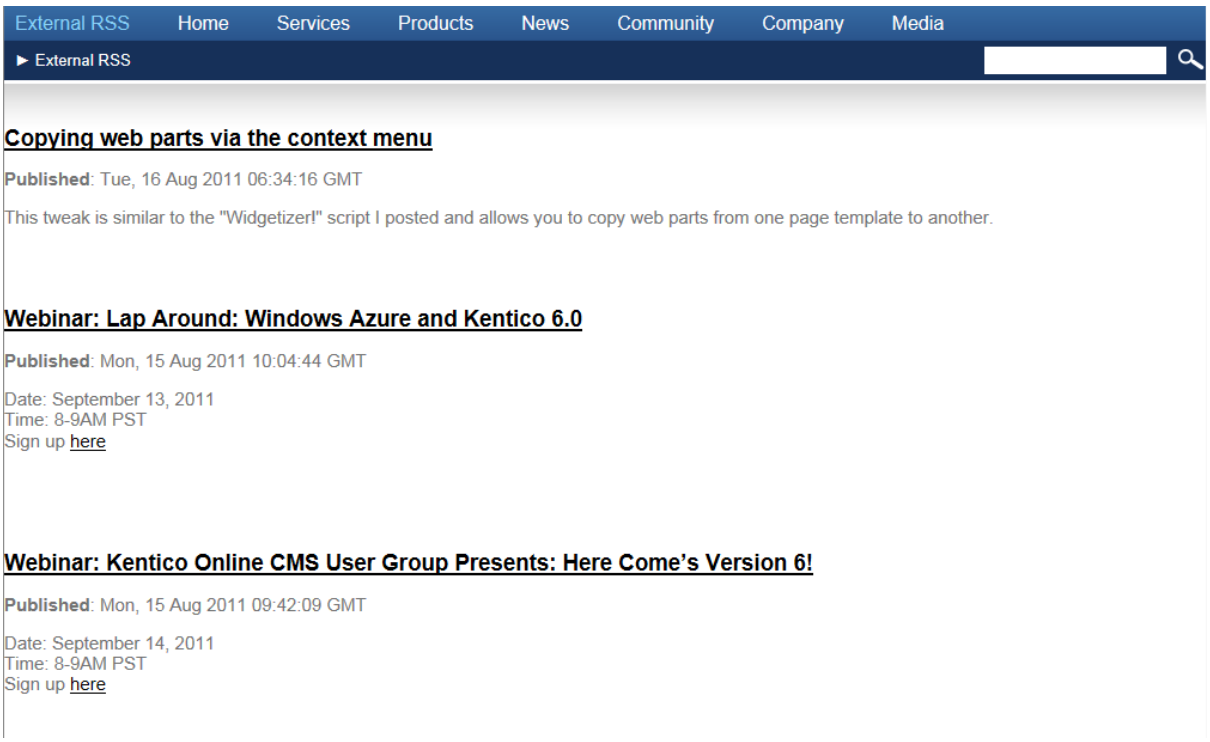
The following is a sample transformation code which transforms the *link*, *pubdate* and *description* elements of each record provided by the feed. Content of the elements is retrieved using the **Eval(string columnName)** method, while the element name is used in the parameter. This way, you can get data from any elements in any external RSS feed that you want to get the data from.

```
<h2>
  <a href="<%# Eval("link")%>">
  <%# Eval("title") %>
  </a>
</h2>
<p>
  <strong>Published</strong>: <%# Eval("pubdate") %>
</p>
<p>
  <%# Eval("description") %>
</p>
<br/>
```

Leave the rest of the web part properties at their default values and click **OK**. On the **Design** tab, the page should now look as in the following screenshot.



5. As visible in the screenshot above, the repeater is already displaying some data. To see the result the way your site visitors will see it, switch to the live site and navigate to the new page. You will see that the page really displays content of the RSS feed, as can be seen in the screenshot below.



8.44.5 Creating RSS feed pages manually (obsolete)

Due to backward compatibility with Kentico CMS versions prior to 5.5, it is also possible to create a dedicated page with an RSS feed. This way of creating RSS feeds is **obsolete** now, as syndication web parts provide a much more convenient way of creating RSS feeds.

The default installation contains a simple **CMSPages\NewsRss.aspx** page which shows how to build your own RSS feed. It works with news items, but you can modify the code so that it displays a different type of documents.

The following code example shows the code of the *NewsRss.aspx* page.

Please note: If you installed the Kentico CMS project as a web application, you need to rename the *CodeFile* property on the first line to *Codebehind* for the code example to be functional.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="RSS.aspx.cs"
    Inherits="RSSNews" %>
<%@ Register Assembly="CMS.Controls" Namespace="CMS.Controls"
    TagPrefix="cc1" %><?xml version="1.0" encoding="utf-8" ?>
<rss version="2.0">
 <channel>
  <title>News RSS</title>
    <link><![CDATA[<%=HttpContext.Current.Request.Url.AbsoluteUri.Remove
(HttpContext.Current.Request.Url.AbsoluteUri.Length -
HttpContext.Current.Request.Url.PathAndQuery.Length) +
HttpContext.Current.Request.ApplicationPath%>]]></link>
  <description>News RSS Feed</description>
  <cc1:cmsrepeater ID="NewsRepeater" runat="server" OrderBy="NewsReleaseDate DESC"
 ClassNames="cms.news"
   TransformationName="cms.news.rssitem"
SelectedItemTransformationName="cms.news.rssitem"
   Path="/news/%"></cc1:cmsrepeater>
 </channel>
</rss>
```

As you can see, the page contains only RSS elements with dynamic code. The RSS items are rendered using a *CMSRepeater* control with appropriate transformation.

The code behind looks like this:

[C#]

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.ContentType = "text/xml";
}
```

This code changes the output content type to XML.

How to create an RSS feed page for a different document type

If you want to display articles instead of news in your RSS feed, you will need to follow these steps:

1. Create a new ASPX page called articles_rss.aspx.

2. Copy and paste all code from the *NewsRss.aspx* file except for the <%@ Page %> directive.

3. Change the following properties of the CMSRepeater control:

- **OrderBy**: DocumentModifiedWhen DESC
- **ClassName**: cms.article
- **TransformationName**: cms.article.rssitem
- **SelectedItemTransformationName**: cms.article.rssitem
- **Path**: /%
- **WhereCondition**: leave empty

4. Add the same line of code as used in the *NewsRss.aspx.cs* code behind file (*Response.ContentType*

= *"text/xml"*) to articles_rss.aspx.cs.

5. Create the trasformation **cms.article.rssitem** like this in **Site Manager -> Development -> Document Types -> ... edit Article ... -> Transformations**:

```
<item>
     <guid isPermaLink="true"><![CDATA[<%# GetAbsoluteUrl(GetDocumentUrl()) %>]]
></guid>
     <title><![CDATA[<%# Eval("ArticleTitle") %>]]></title>
     <description><![CDATA[<%# Eval("ArticleText") %>]]></description>
     <pubDate><%# Convert.ToDateTime(Eval("DocumentModifiedWhen")).ToString("r") %
></pubDate>
     <link><![CDATA[<%# GetAbsoluteUrl(GetDocumentUrl()) %>]]></link>
</item>
```

This code renders the particular items.

8.45 System integration bus

8.45.1 Overview

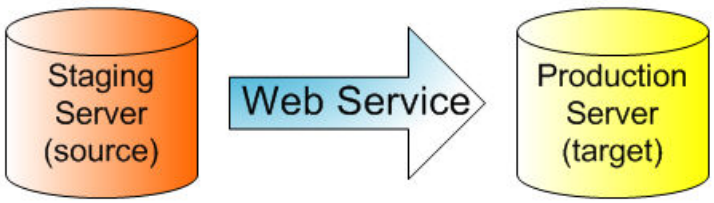
The System integration bus module enables integration of Kentico CMS with external systems. Detailed information about the module, implementation of custom integration connectors, management of integration tasks and usage examples of the module can be found in Kentico CMS Integration Guide.

8.46 Staging

8.46.1 Overview

The **Staging** module allows you to easily transfer changes made to documents or objects in a Kentico CMS instance on one server to another instance on another server. It is also possible to perform complete synchronization of all documents and objects this way. This is particularly useful if you need to separate website content in development, editing (staging) and live (production) environment.

All documents stored in CMS Desk's content tree can be synchronized using the module. However, not all objects in a Kentico CMS system can be synchronized this way. The What can be synchronized topic gives you an overview of which data can be synchronized using the module.



For the synchronization to be possible, you need to configure Kentico CMS instances on particular servers as described in the Staging configuration topic. Each instance can be configured as a source server or as a target server. Changes from the source server are transferred to the target servers, as depicted in the diagram above. Every site can use different target servers for content staging. You can

even synchronize between different sites in the same Kentico CMS instance (in the same database).

It is also possible to have servers configured as both a target and a source server at the same time, so that changes can be transferred in both directions. This possibility and the configuration required for it is described in the Bi-directional staging topic.

Once all instances are configured, you can start synchronizing the changes. All changes are logged as staging tasks in **CMS Desk -> Tools -> Staging**. The Synchronizing the content topic explains how staging tasks can be transferred to the other servers, as well as all other operations possible in this section of the user interface. Synchronization can also be performed automatically, based on a pre-defined scheduled task named **Content synchronization**. The Automatic synchronization topic explains this scheduled task can be enabled.

The synchronized data is transferred over a secured web service. Management of content staging tasks and performing of the actual synchronization can be allowed only to certain roles. The Security topic provides more information about permissions that need to be granted to roles in order to perform these actions.

Another way of transferring your content from one Kentico CMS instance to another is to export the data from one instance and import them to another one. The Managing sites -> Export and import chapter of this guide gives you an extensive overview of the built-in export and import functionality.

In the Staging internals and API sub-chapter, you can find information about database tables and classes that are used by the Staging module, as well as several examples of how staging can be performed using Kentico CMS API.

8.46.2 What can be synchronized

The Staging module **supports** synchronization of the following data:

- Document data - documents in the website content tree. Timeout defined for documents that are part of a workflow is not synchronized for consistency reasons.
- Document attachments - if a document contains attachments or file fields, the files are synchronized together with the document.
- Document relationships - relationships between documents are synchronized if the relationship type and both documents exist on the target server. Synchronization is NOT supported for relationships between documents on different sites.
- Workflow process - only published document versions are synchronized to the target server and both servers need to have the same workflow schemas defined.
- Custom tables and their data.
- Media libraries and the files and folders in them
- ACLs (document-level permissions)
- **Global objects** - all global objects except for the cases mentioned in the list below.

The Staging module **does not support** synchronization of the following data:

- Abuse reports
- Accounts
- Activities
- Blog comments (blogs and blog posts are synchronized as documents)
- Booking event attendees
- Contacts (the state of contacts in automation processes is also not synchronized)

- Event log
- Export history
- Forms data (the actual form objects are synchronized)
- Forum posts (forum objects are synchronized)
- Friends
- Message board messages (message board objects are synchronized)
- Messaging module messages
- Physical files associated with global objects (for example web part source file, form control files, ASPX page template files)
- Web templates

Staging of Media library content

Kentico CMS supports staging of **physical files and folders** stored in the Media library file system. Staging tasks related to Media library files are logged in CMS Desk -> Tools -> Staging -> Objects.

You can limit the maximum size of synchronized media library files as described in the Staging large files topic.

Please note

Changes in media library files and folders are logged as content staging tasks only when performed via the UI. If you make changes directly in the library folder in the file system, e.g., upload or update files using FTP, the changes are not logged.

Also, if you make a change to a file via the UI and then update the file using FTP, the current file (the one updated using FTP) will be transferred to the target server, even if the staging task was created before the file upload. This happens because binary data of the files are loaded in synchronization time, not when the synchronization task is logged.

8.46.3 Staging configuration

Configuration of the Staging module consists of the following three parts:

1. Microsoft WSE configuration - renaming of a *.dll* library on all servers.
2. Source server configuration - configuration of the server from which changes will be transferred to the target servers.
3. Target server configuration - configuration of the server to which changes will be transferred from the source servers.
4. Configuring servers for staging macro expressions

Apart from this configuration, you also need to ensure that all instances use the same settings (document types, templates, web parts ...), code files and that both servers use the same version of Kentico CMS.

Microsoft WSE configuration

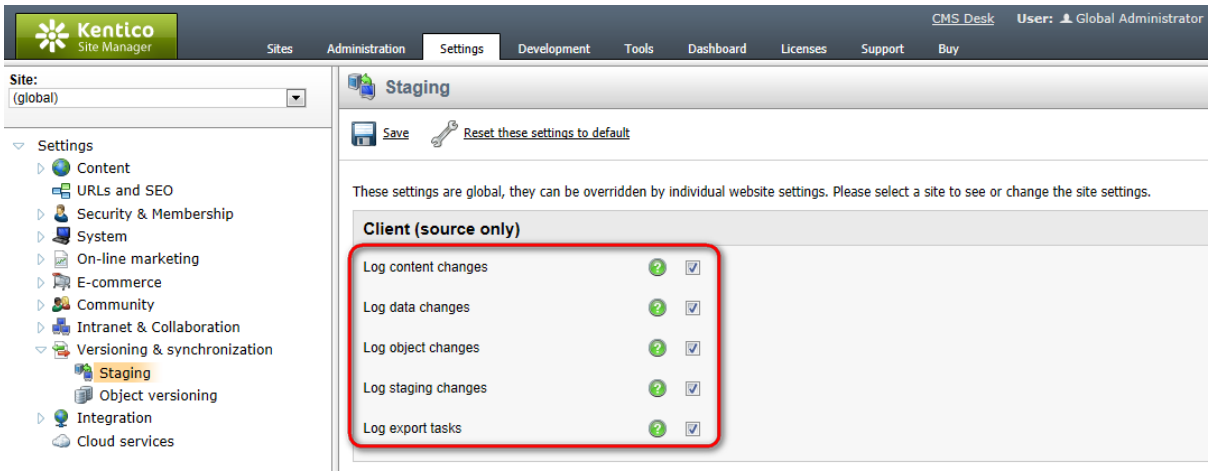
To enable content staging, you first need to open the **bin** folder under your web project and rename the **Microsoft.Web.Services3.dll.rename** file to **Microsoft.Web.Services3.dll**. This needs to be done on both source and target servers.

Source server configuration

1. To configure a Kentico CMS instance as a source server, you first need to enable logging of content staging tasks using the following settings in **Site Manager -> Settings -> Versioning & synchronization -> Staging**.

- **Log content changes** - if enabled, synchronization tasks are automatically logged when content (a document) is modified.
- **Log data changes** - if enabled, synchronization tasks are automatically logged when custom tables data are modified.
- **Log object changes** - if enabled, synchronization tasks are automatically logged when an object is modified.
- **Log staging changes** - if enabled, synchronization tasks are logged for changes made by synchronization from another server to this server. See Bi-directional staging for more details.
- **Log export tasks** - if enabled, synchronization tasks are logged when an object is deleted (incremental update support).

With these settings enabled, all changes to the corresponding content are logged as content staging tasks. These tasks can then be transferred to the target servers and performed there to synchronize the content.

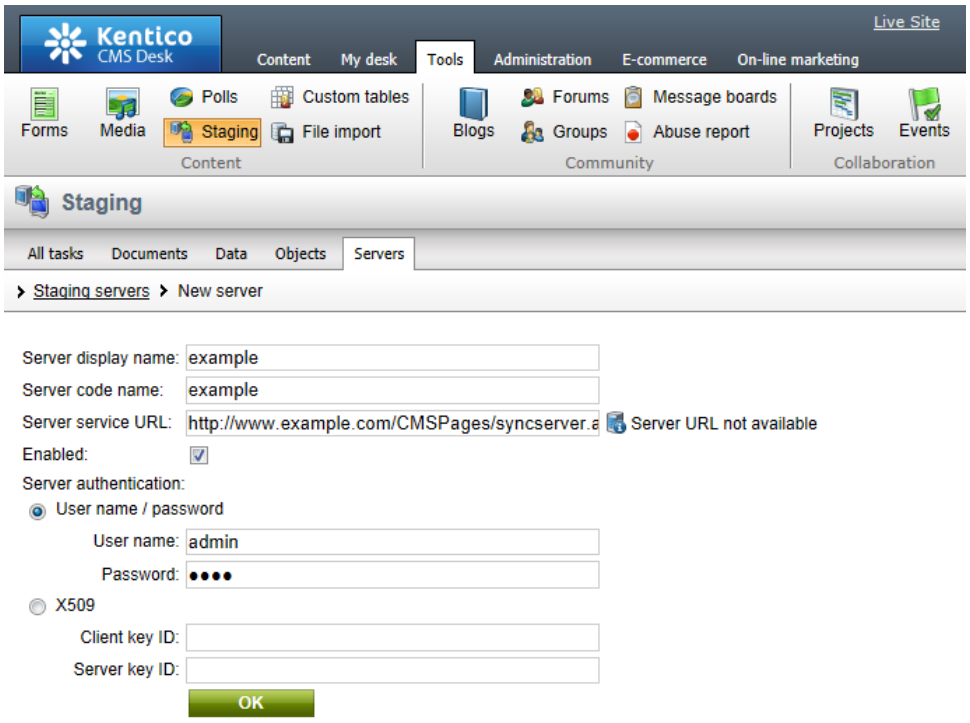


2. Then, you need to specify the target server(s). Go to **CMSDesk –> Tools –> Staging -> Servers**.

This is where you can manage the list of target servers. To add a new server, click **New server** (), enter the server properties and save them:

- **Server display name** - server display name displayed to the users in the administration interface.
- **Server code name** - server name used in website code.
- **Server URL** - staging service URL that points to the content staging web service of the target server; the web service page is located at *~/CMSPages/syncserver.asmx*, so the URL should look like this: *http://www.example.com/CMSPages/syncserver.asmx.* The **Check Server availability** button () checks if the server with the entered URL is available and displays the result.

- **Enabled** - if checked, the target server is enabled, it means that synchronization tasks are automatically generated for the server and that synchronization is enabled for this server; you can temporarily disable the server by un-checking the box, e.g. in case of server maintenance.
- **Server authentication** - server authentication settings; you should set the same parameters that are configured for your target server (described below); the default user name is **admin** and the default password is **pass**; if you want to use X509 authentication, please consult the Using X.509 authentication topic.



Please note

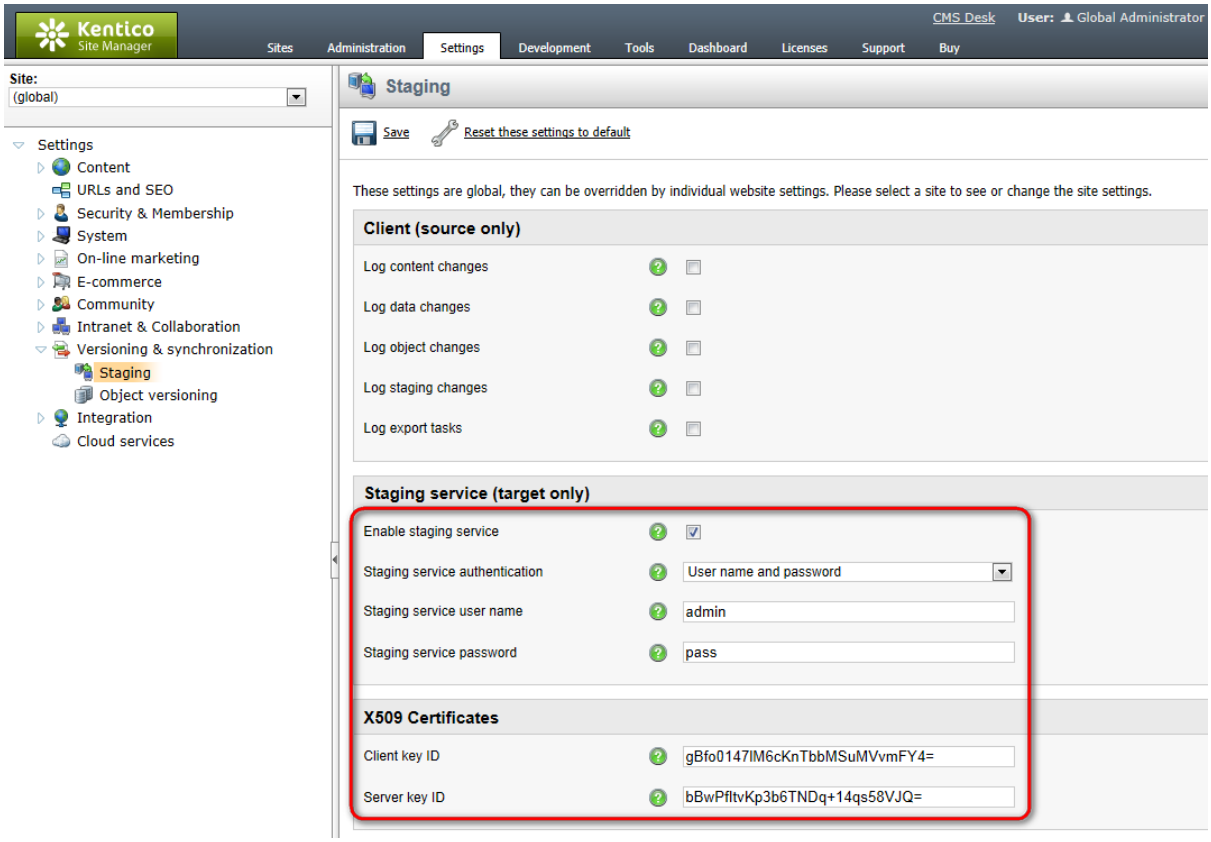
Tasks are logged only when there is at least one target server created and enabled. If there is no server created or if no server is running, no tasks are logged.

Target server configuration

On the target server, the staging service is disabled by default. For it to work, you need to set the following values in **Site Manager -> Settings -> Versioning & synchronization -> Staging**:

- **Enable staging service** - if checked, the staging service is enabled for the given site;
- **Staging service authentication** - staging service authentication type; it is recommended that you choose *USERNAME* authentication to configure staging first, test synchronization and then optionally configure the site for *X509* certificates
  - *USERNAME* – username/password authentication (fast, recommended for the data without high security requirements)

- *X509* – X509 certificate authentication (more secure, slower, requires certificates); more details can be found in the Using X.509 authentication topic
- **Staging service username** and **password** - username and password for the *USERNAME* authentication
- **Server key ID** and **Client key ID** - certificate keys for the *X509* authentication



Configuring servers for staging macro expressions

The system uses signatures to ensure the security of macro expressions. Macro signatures contain the user name of the macro's author and a hash of the given expression.

The hash function used to create the signatures appends a salt to the input. The default salt is the application's main database connection string, so the signatures are only valid in the environment where the macros were originally saved. To allow macros to work correctly on all staging servers, you need to set the same custom hash salt for all servers:

- Add the **CMSHashStringSalt** key to the *appSettings* section of the web.config file on all staging servers. You can use any string as the value, but the salt should be random and at least 16 characters long. For example, a randomly generated GUID is a strong salt:

```
<add key="CMSHashStringSalt" value="e68b9ad6-a461-4707-8e3e-ece73f03dd02" />
```

The best option is to set the hash salt value before you start creating content for your website. Changing the salt causes all current hash values to become invalid. To fix existing macro expressions in the

system after changing the hash salt, you need to re-sign the macros. See Macro security for more information.

Warning

In addition to macro signatures, the system uses the **CMSHashStringSalt** value for other hash functions. Changing the hash salt on a website that already has defined content may break dialog links and images on your website.

If you encounter such problems, you need to re-save the given content (the system then creates the hashes using the new salt).

8.46.4 Bi-directional staging

Bi-directional content staging, i.e. content staging where servers are not exclusively source or target ones, but can act both ways, is achievable with some extra settings. The advantage of such configuration is that you can make changes not only on the source server, but also on the other server (s), while the changes are transferred to the rest of the servers too.

There are two types of bi-directional content staging, depending on the number of servers between which you want to synchronize:

- Simple - used for synchronization between 2 servers only
- Advanced - used for synchronization between more than 2 servers

Both configuration procedures are described in the text below, click the respective link to scroll directly to the beginning of the description.

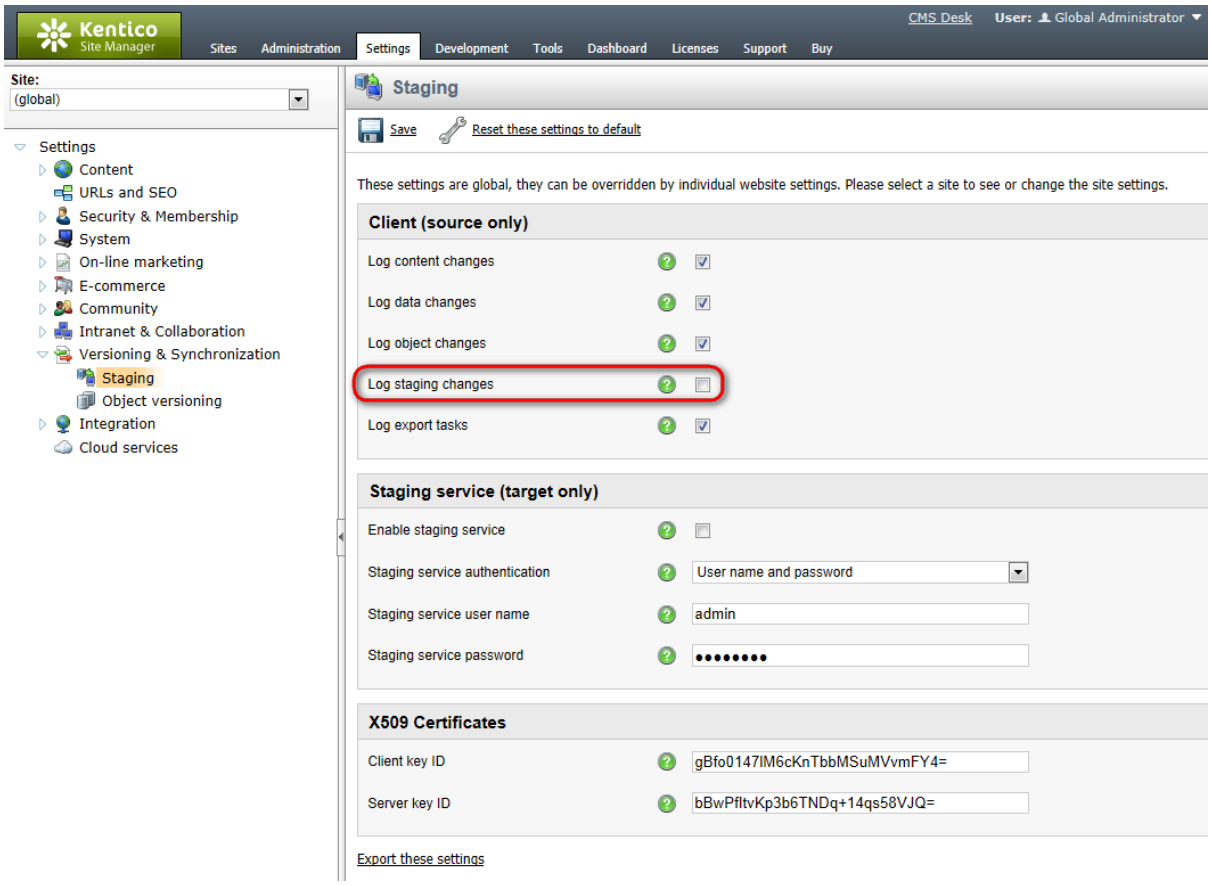
Simple bi-directional staging

If you want to perform content staging bi-directionally between **two servers**, i.e. you want to transfer changes made on Server 1 to Server 2 and also in the other direction from Server 2 to Server 1, you need to configure both servers as a source and a target server at the same time, as described in the Staging configuration topic.

On top of it, you also need to **disable** the **Log staging changes** option in **Site Manager -> Settings -> Versioning & Synchronization -> Staging** on both servers.

If this option is enabled, changes made to the system via content staging synchronization (i.e. transferred from Server 1 to Server 2) are logged as new synchronization tasks on the target server (on Server 2). Now if you perform synchronization in the reversed direction, i.e. back from Server 2 to Server 1, these tasks are also performed and logged back on Server 1 as new staging tasks. This goes on and on in a never-ending loop, which results in the tasks remaining and not being deleted on both servers.

In order to prevent this behavior, you need to disable the **Log staging changes** option on both servers so that the staging changes are not logged as new synchronization tasks.

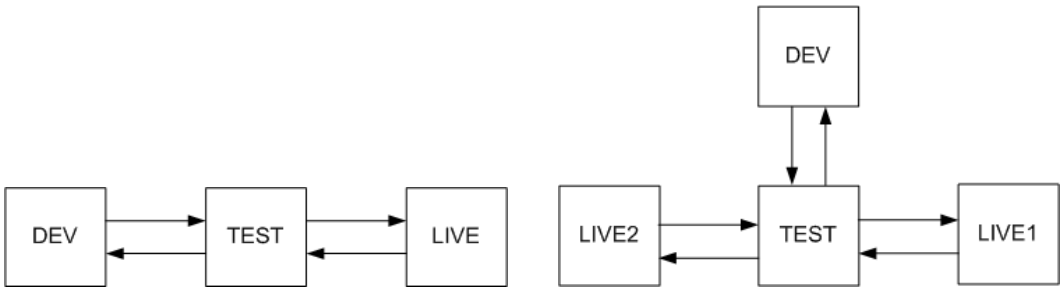


Advanced bi-directional staging

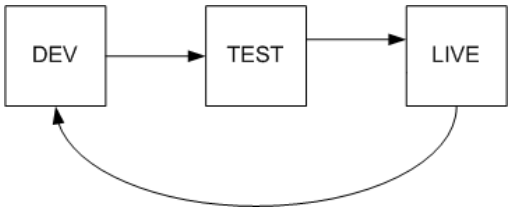
Bi-directional content staging is also possible on more than two servers. In this case, a list of servers where a staging task has already been processed is transferred with each task. This prevents redundant staging tasks from being logged for servers where the changes have already been performed.

The rule is that there has to be **only one path between any two servers in both directions**. The following figures show examples of supported server topologies. The rectangles represent particular servers, while the arrows indicate the flow of content staging synchronization tasks.

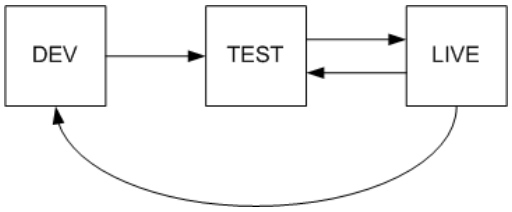
The first option is to have the servers connected using **star topology**.



Another possible topology is the **circle topology**.



Other topologies are not supported. The following diagram shows an unsupported environment. It doesn't follow the rule which says that there should be only one path between any two servers. There are two paths from the LIVE server to the TEST server, which breaks the rule.

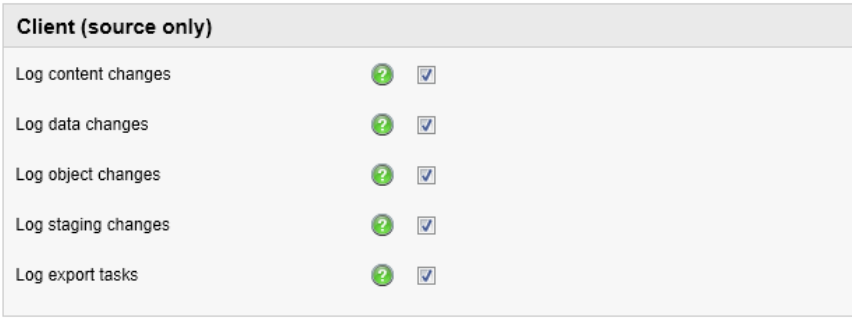


Configuring advanced bi-directional staging

Follow the instructions below to configure bi-directional staging for three servers—development server, test server and live server, using star topology.

1. On each of the servers that you want to be part of the bi-directional staging, navigate to **Site Manager -> Settings -> Versioning & Synchronization -> Staging**.

2. Configure the servers as **Source** by turning all the check-boxes on.



3. Configure the servers as **Target**, using the **user name and password** authentication.

Configuring Development server

1. Into the *AppSettings* section of the *web.config* file of the server, enter the following value to name the server **Dev**.

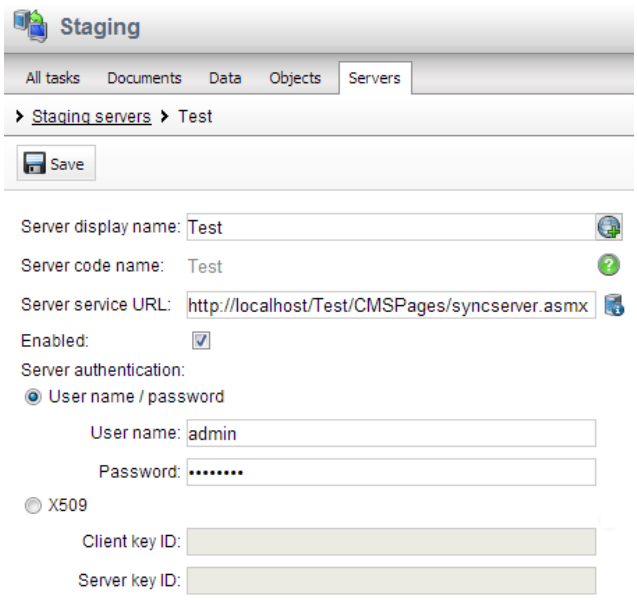
```
<add key="CMSStagingServerName" value="Dev" />
```

2. Navigate to **CMS Desk -> Tools -> Staging**.

3. Switch to the **Servers** tab and add **New server** ( ).

4. Configure the new server to target the **Test** server. You will configure the **Test** server later.

Make sure the value of **Server code name** is set to **Test**, which is the value that you will use in the **Test** server's *web.config* file.



Configuring Test server

1. Into the *AppSettings* section of the *web.config* file of the server, enter the following value to name the server **Test**.

```
<add key="CMSStagingServerName" value="Test" />
```

2. Navigate to **CMS Desk -> Tools -> Staging**.

3. Switch to the **Servers** tab and add **New server** ( ).

4. Configure the new server to target the **Dev** server that you created previously.

Make sure the value of **Server code name** is set to **Dev**, which is the value used in the **Dev** server's *web.config* file.

5. Add another **New server** ( ).

6. Configure the new server to target the **Live** server. You will configure the **Live** server later.

Make sure the value of **Server code name** is set to **Live**, which is the value that you will use in the **Live** server's *web.config* file.

Configuring Live server

1. Into the *AppSettings* section of the *web.config* file of the server, enter the following value to name the server **Live**.

```
<add key="CMSStagingServerName" value="Live" />
```

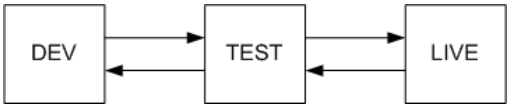
2. Navigate to **CMS Desk -> Tools -> Staging**.

3. Switch to the **Servers** tab and add **New server** ( ).

4. Configure the new server to target the **Test** server that you created previously.

Make sure the value of **Server code name** is set to **Test**, which is the value that you will use in the **Test** server's *web.config* file.

You have now set up bi-directional staging between three servers.



8.46.5 Synchronizing the content

All changes made to the documents and objects are tracked in the database, in the synchronization log. You can view the changes in **CMS Desk –> Tools –> Staging**.

Please note

It is recommended that both the target and source sites have the same *Code name* prior to synchronizing content between them. You can change the value by going to *Site manager* and *Editing* ( ) the site.

The interface for viewing changes and performing synchronization is divided into the following tabs:

All tasks tab

On the **All tasks** tab, you can see a list of all content staging tasks, i.e. all changes made to the system that can be synchronized on the target server.

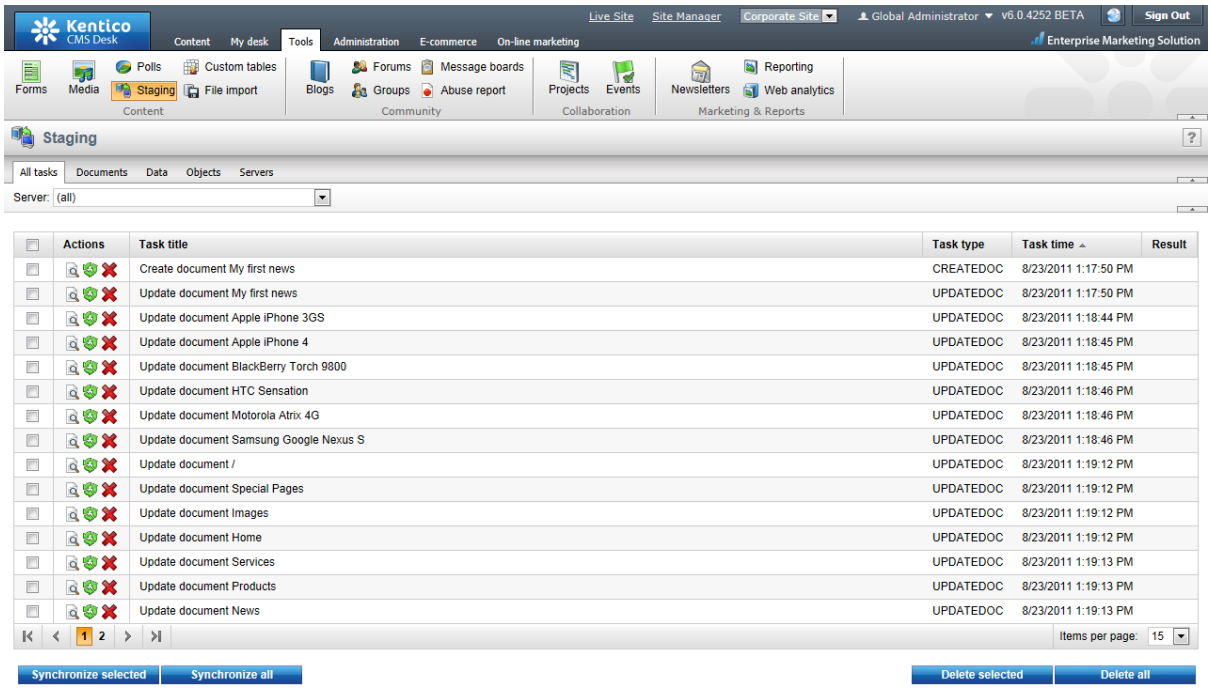
Using the **Server** drop-down, you can choose the target server that you want to synchronize. By choosing **(all)**, you perform synchronization for all available target servers. Then you can perform one of the following actions using the buttons at the bottom:

- **Synchronize selected** - performs synchronization for all tasks selected by the check-boxes (☑) on the target server;
- **Synchronize all** - performs synchronization for all listed tasks on the target server; in case you made any changes to content on the target server in the meantime, these changes will be overwritten

- **Delete selected** - deletes all tasks selected by the check-boxes (☑) on the target server
- **Delete all** - deletes all listed tasks on the target server

You can also perform the following actions separately with particular staging tasks:

- **View** - opens a new window with detailed information about the staging task
- **Synchronize** - performs the synchronization task on the target server
- **Delete** - deletes the synchronization task from the list



Documents tab

In the screenshot below, you can see the **Documents** tab. By clicking the website root in the content tree on the left, you can view a list of all changes (synchronization tasks) made to all the documents of your website. By clicking a particular document, you can view only the changes made to it.

You can perform the same actions as on the **All tasks** tab, as described above.

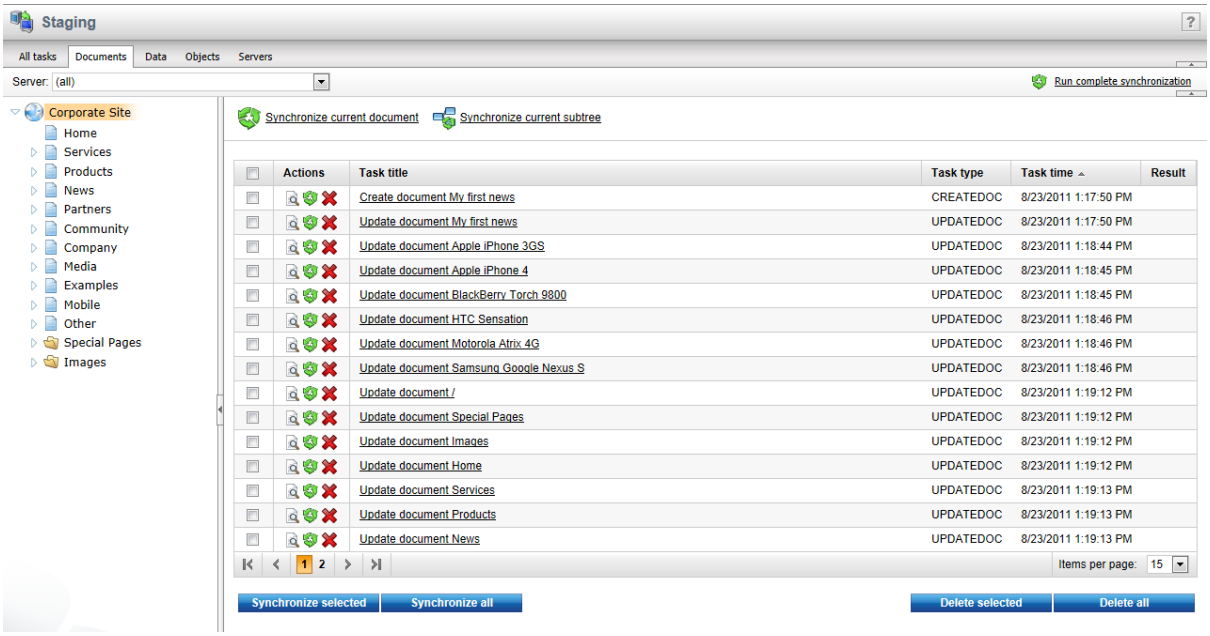
You can also perform the following manual actions. These actions are manual because they are not

related to the listed tasks and they can be performed even if there are no synchronization tasks logged:

- **Run complete synchronization** - performs complete synchronization of all documents in the content tree
- **Synchronize current document** - synchronizes the currently selected document
- **Synchronize current sub-tree -** synchronizes all documents in the selected sub-tree

The following types of tasks are logged for documents. You can see the type in the **Task type** column:

- **CREATEDOC** - document was created
- **UPDATEDOC** - document was modified
- **DELETEDOC** - document was deleted
- **DELETEALLCULTURES** - all cultural versions of the document were deleted
- **PUBLISHDOC** - document was published
- **ARCHIVEDOC** - document was archived
- **REJECTDOC** - document was rejected
- **MOVEDOC** - document was moved to another location in the content tree



Data tab

In the screenshot below, you can see the **Data** tab. This is where changes made to data in **custom tables** are logged.

You can perform the same actions as on the **All tasks** tab, as described above.

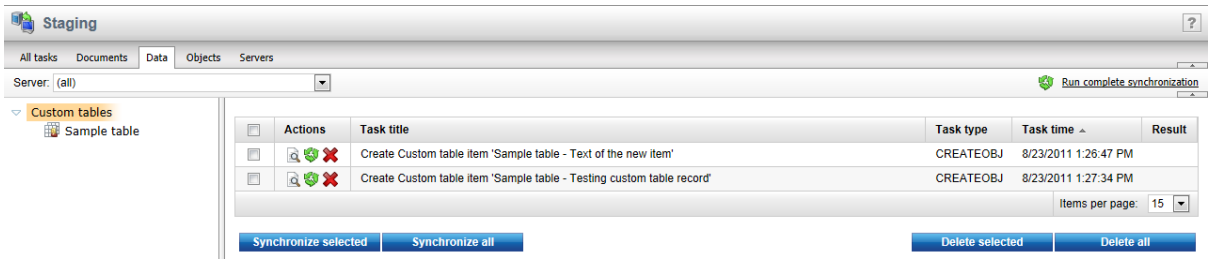
You can also perform the following manual actions. These actions are manual because they are not related to the listed tasks and they can be performed even if there are no synchronization tasks logged:

- **Run complete synchronization** - performs complete synchronization of all data in all custom tables
- **Synchronize current sub-tree -** synchronizes all data in the selected custom table

The following types of tasks are logged for custom tables data. You can see the type in the **Task type** column:

- **CREATEOBJ** - new item was added to a table
- **UPDATEOBJ** - an item in a table was updated
- **DELETEOBJ** - an item in a table was deleted

Please note: Staging tasks for custom table items are logged based on their *ItemGUID* columns. Changes made to items that do not have this column (typically custom tables imported from older versions of Kentico CMS) are not logged. You can edit custom tables in CMS Desk -> Tools -> Custom tables. You will see a warning message with a link letting you generate the GUIDs for the respective tables.



Objects tab

In the following screenshot, you can see the **Objects** tab. This is where changes made to objects are logged.

The main categories are **website** and **Global objects**. The first category contains object changes connected to the current website, whilst the second one contains object changes for global objects.

You can perform the same actions as on the **All tasks** tab, as described above.

When you select a particular object category, you get also the following manual action offered. This action is manual because it is not related to the listed tasks and can be performed even if there are no synchronization tasks logged:

- **Synchronize current sub-tree -** synchronizes all objects in the selected category

The following types of tasks are logged for objects. You can see the type in the **Task type** column:

- **CREATEOBJ** - new object was created
- **UPDATEOBJ** - object was modified
- **DELETEOBJ** - object was deleted
- **ADDTOSITE** - object was assigned to a site; applicable only to site-related objects
- **REMOVEFROMSITE** - object was removed from a site; applicable only to site-related objects

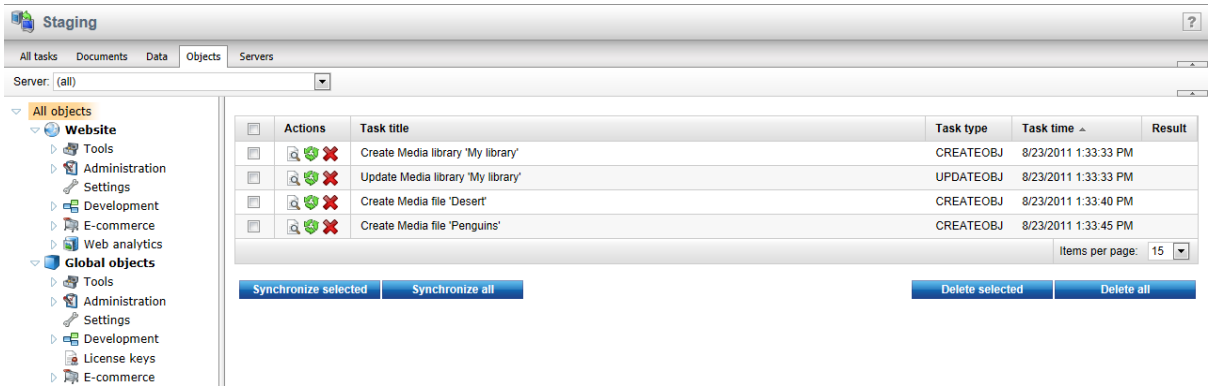
The following types of tasks are logged for folders in media libraries:

- **CREATEFOLDER** - folder was created
- **RENAMEFOLDER** - folder was renamed
- **COPYFOLDER** - folder was copied
- **MOVEFOLDER** - folder was moved
- **DELETEFOLDER** - folder was deleted

Please note

Global metadata changes such as changes to document types, custom tables and system tables produce staging tasks for all staging servers of all sites. In such case, it is recommended to synchronize such changes to all servers of all sites at the same time to prevent overwriting of such metadata and losing the data by synchronizing the older tasks later.

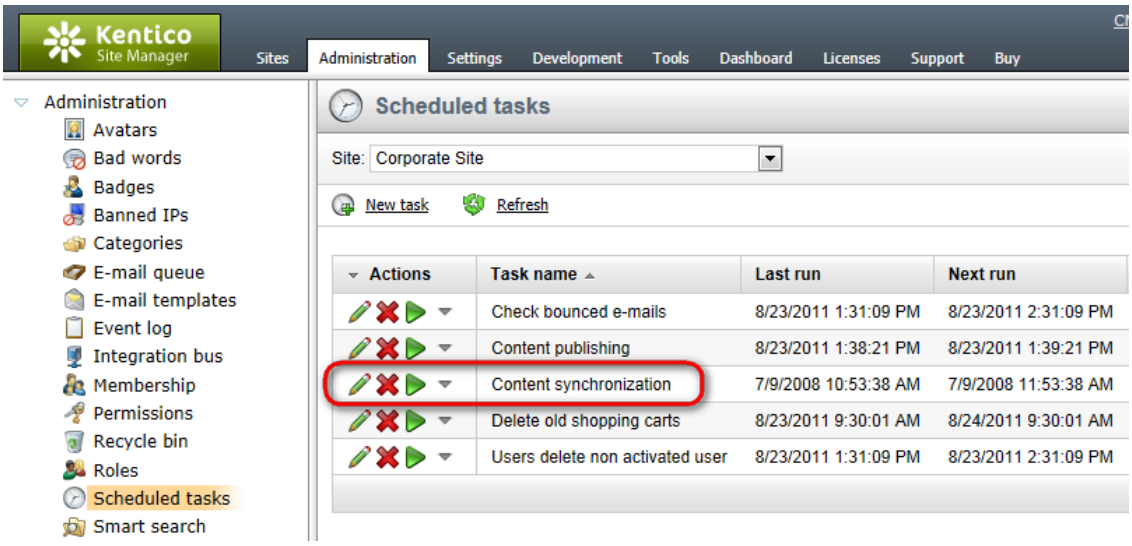
You can use the *<add key="CMSStagingTreatServerNamesAsInstances" value="true" />* key to make sure that once the global task is synchronized, it is deleted from all other servers with the same name to prevent such possibility. The default value is false since staging can use multiple target instances targeted with the same names.



8.46.6 Automatic synchronization

Automatic synchronization is available **only for document changes**. Object changes cannot be synchronized this way. Manual synchronization remains the only way to perform object synchronization tasks.

If you want the changes to be synchronized from the staging to the live site on a regular basis without waiting for the administrator's approval, you can configure the scheduled task called **Content synchronization** in **Site Manager -> Administration -> Scheduled tasks**. This task is disabled by default, so you need to enable it and configure the synchronization interval.



More information about the built-in scheduler in Kentico CMS can be found in the Development -> Scheduler chapter of this guide.

8.46.7 Using X.509 authentication

In order to use **X.509** authentication, install your own certificates or you use our sample ones.

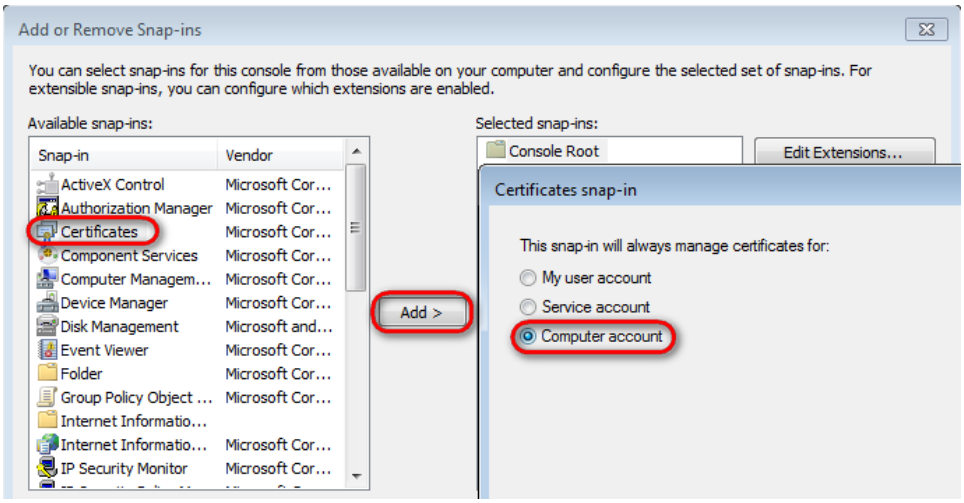
Using the sample certificates

Kentico CMS is delivered with sample client and server private certificates. In order to install them, do the following on the source server and on the target server:

Server certificate

Install the server certificate:

1. Choose **Start -> Run**, type **mmc** and press **Enter**.
2. In the console window, choose **File -> Add/Remove Snap-in**.
3. Choose **Certificates** and click **Add**.
4. Choose **Computer account** and click **Next**.



5. Choose **Local computer (the computer this console is running on)** and press **Finish**.

6. Close the **Add or Remove Snap-ins** window by clicking **OK**.

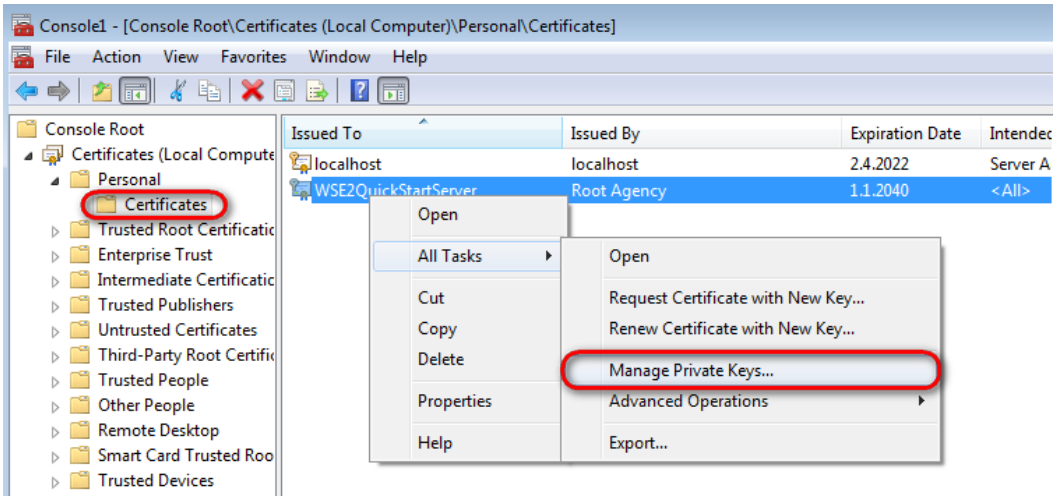
7. Unfold **Certificates (Local Computer)** under the console root, right-click **Personal** and choose **All Tasks -> Import...** The **Certificate Import Wizard** starts.

8. Import the *Server private.pfx* file located in **C:\Program Files\Kentico CMS\<version>\SampleCertificates**.

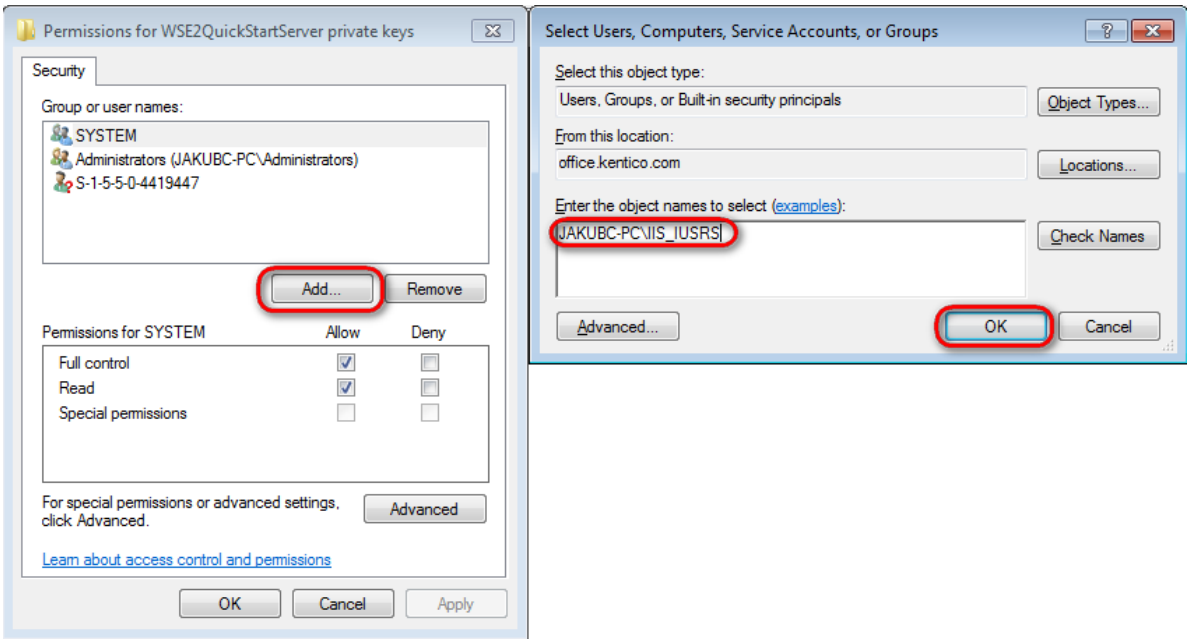
9. Enter the following password for the sample certificate: **wse2qs**.

10. Don't change any other settings and finish the **Certificate Import Wizard**.

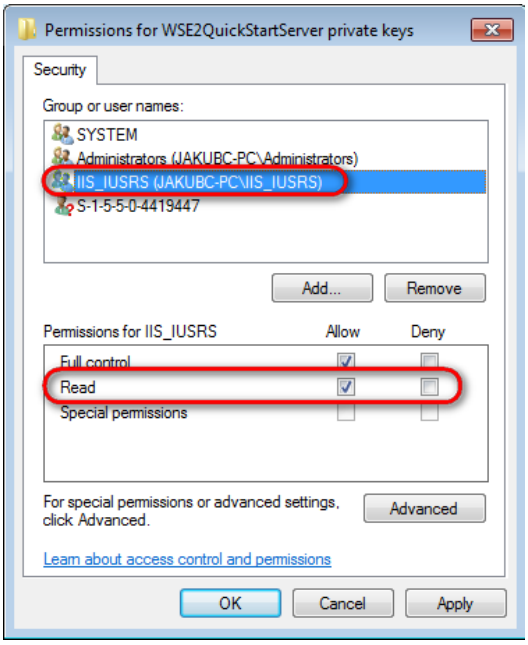
11. Now, grant **Read** permission to the certificate file for the ASP.NET account. Do that by right-clicking the imported *WSE2QuickStartServer* certificate and choosing **All tasks -> Manage private keys**.



12. Click on **Add...**, fill in the name of the account and click **OK**.



13. Make sure the account's permission is set to **Read - Allow** and click **OK**.



Client certificate

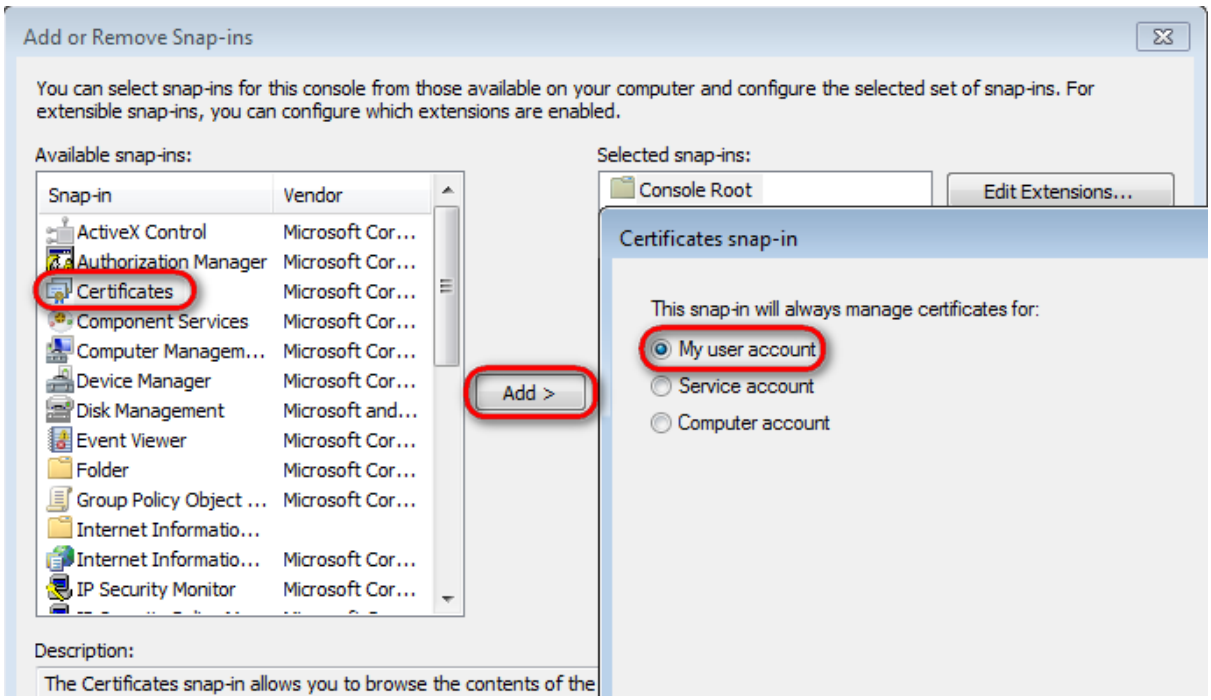
Install the client certificate:

1. Choose **Start -> Run**, type **mmc** and press **Enter**.

2. In the console window, choose **File -> Add/Remove Snap-in**.

3. Choose **Certificates** and click **Add**.

4. Choose **My user account** and click **Finish**.



5. Close the **Add or Remove Snap-ins** window by clicking **OK**.

6. Unfold **Certificates - Current User** under the console root, right-click **Personal** and choose **All Tasks -> Import...** The **Certificate Import Wizard** starts.

8. Import the *Client private.pfx* file located in **C:\Program Files\Kentico CMS\<version>\SampleCertificates**.

9. Enter the following password for the sample certificate: **wse2qs**.

10. Don't change any other settings and finish the **Certificate Import Wizard**.

11. Now you need to grant the **Read** permissions for the certificate file to the ASP.NET account. For the *Client certificate,* do that using the **WseCertificate3.exe** tool that can also be found in **C:\Program Files\Kentico CMS\<version>\Sample Certificates** folder.

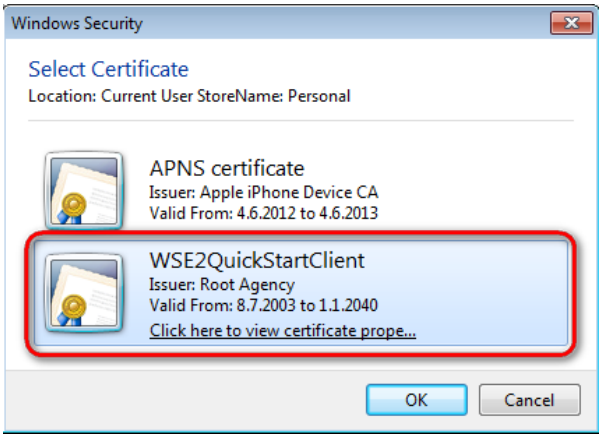
12. Run the **WseCertificate3.exe** tool.

13. Choose **Current User** in the **Certificate Location** field.

14. Choose **Personal** in the **Store Name** field.

15. Click **Open Certificate**.

16. Choose the *WSE2QuickStartClient* certificate and click **OK**.



17. Click **View Private Key File Properties...** and grant the **Read** permission for this file to the ASP.NET account.

18. Click on **Add...**, fill in the name of the account and click **OK**.

19. Make sure the account's permission is set to **Read - Allow** and click **OK**.

Important: Sample certificates

Using the sample certificates is not secure and it's also very slow. It's highly recommended that you use your own certificate issued by a certification authority.

Using your own certificates

If you're using your own certificates (highly recommended), you will need to update the following values in **Site Manager -> Settings -> Versioning & Synchronization -> Staging**:

- Client key ID
- Server key ID

To get these IDs, you can use the **WseCertificate3.exe** tool located in **C:\Program files\KenticoCMS\<version>\SampleCertificates**.

1. Run the **WseCertificate3.exe** tool.

2. Choose **Local Computer** in the **Certificate Location** field.

3. Choose **Personal** in the **Store Name** field.

4. Click **Open certificate** and select either client or the server certificate. In the **Key identifiers group** you can now see the certificate key, **Windows key identifier (Base64)** should be used within Kentico CMS settings.

Tip

If you encounter problems with content staging when using SLL (X.509), you may try adding the following key to your *web.config* file:

```
<add key="CMSStagingAcceptAllCertificates" value="true" />
```

This key ensures that all certificates will be accepted. If set to false, only certificates issued by a certification authority will be accepted.

Configuring staging for the use of certificates

Now that you have installed and configured the certificates, configure staging inside Kentico CMS to use the certificates for authentication.

Target server

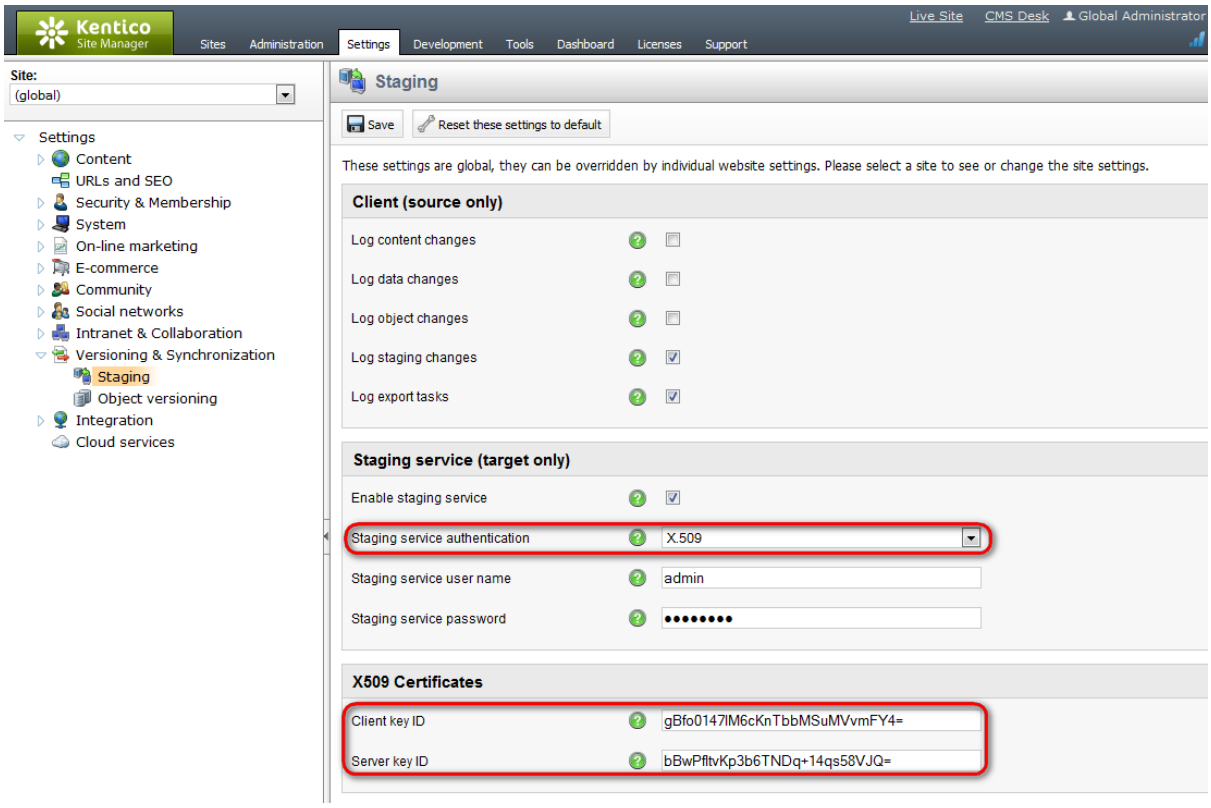
On the target server, change the staging service authentication service as follows:

1. Navigate to **Site manager -> Settings -> Versioning & Synchronization**.

2. Under the **Staging service (target only)** settings group, change the form to X.509.

3. **Save** the settings.

4. Note that you need to copy the **Client** and **Server key ID** to your source server's settings.



Source server

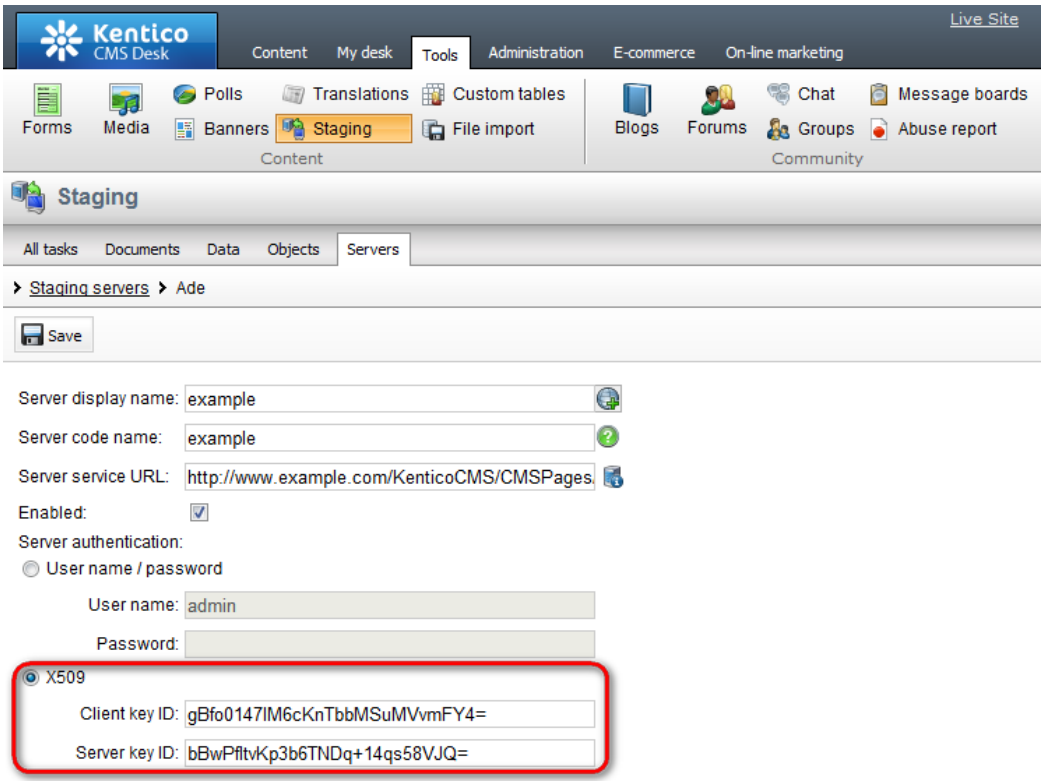
On the source server, change the settings as follows:

1. Navigate to **CMS Desk -> Tools -> Staging**.

2. Choose the **Servers** tab and **Edit ( )** the server you want to configure.

3. Change **Server authentication** to **X509** and copy the **Client** and **Server key ID's** from the target server.

4. **Save** the configuration.



You have now successfully configured staging in Kentico CMS for the use of certificates in authentication.

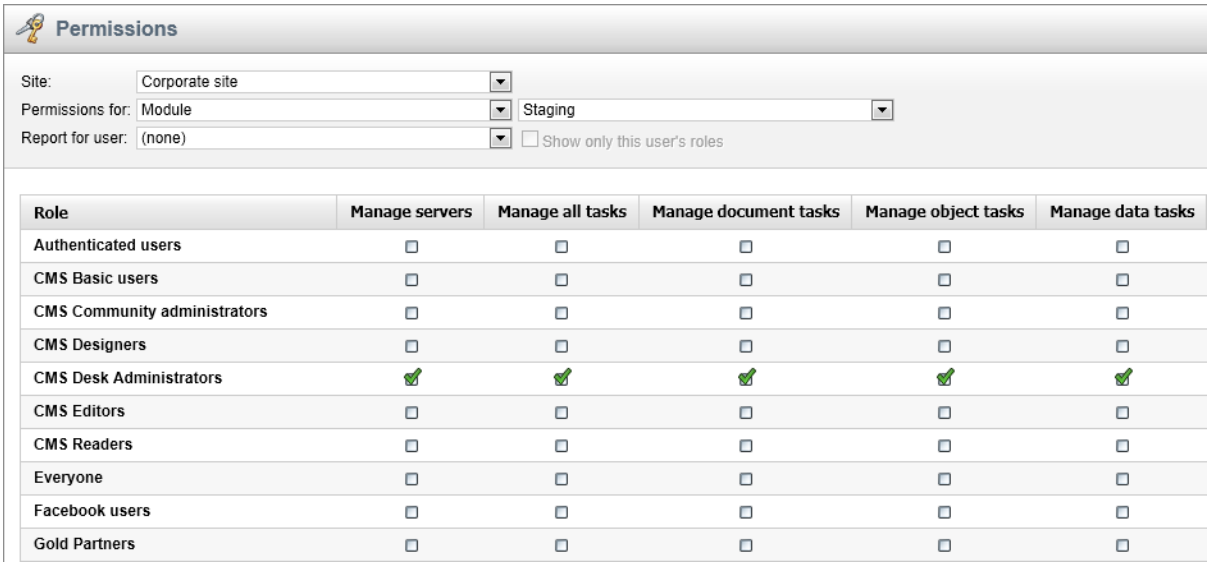
8.46.8 Security

You can control access to the Staging module in **Administration -> Permissions**. You need to select:

- **Site**: your site
- **Permission type**: Modules
- **Permission matrix**: Staging

In the permission matrix, you can grant the following permissions to the particular roles:

- **Manage all tasks** - displays the **All tasks** tab and allows synchronization and management of synchronization tasks on it
- **Manage data tasks** - displays the **Data** tab and allows synchronization and management of synchronization tasks on it
- **Manage document tasks** - displays the **Documents** tab and allows synchronization and management of synchronization tasks on it
- **Manage object tasks** - displays the **Objects** tab and allows synchronization and management of synchronization tasks on it
- **Manage servers** - allows members of the roles to manage target server configurations on the **Servers** tab



8.46.9 Staging large files

Kentico allows you to modify the size of files in Media libraries that you want to stage. You can set the size in order to avoid too large files from being synchronized. You may also want to modify the values in order to stage files larger than the values set by default.

Changing maximum content and request size

1. On both the **target and source server**, open the web.config file located in the root of the website.

2. Modify the value of the *maxRequestLength* property. The value is in **kilobytes**.

```
...

<system.web>
    ...

    <httpRuntime maxRequestLength="2097151" ... />

    ...
</system.web>

...
```

3. Add the following code into the *system.webServer* part of the web.config file. The value of *maxAllowedContentLength* is in **bytes**.

Note that there are two *system.webServer* parts in the web.config file by default. The first one is related to WebDav configuration. Enter the code into the second part.

```
...
```

```
<system.webServer>
    ...

    <security>
      <requestFiltering>
        <requestLimits maxAllowedContentLength="2147483648" />
      </requestFiltering>
    </security>

    ...
</system.webServer>

...
```

By default, IIS limits this value to 30000000 bytes (28.6 megabytes). Synchronizing files larger than the set value results in an *HTTP status 404: Not Found* error.

4. Save the web.config file.

Limiting maximum media library file size

To limit the maximum size of a file staged from the media library:

1. On the **source server**, open the web.config file located in the root of the website.

2. Add the following key into the file. The *value* is entered in kilobytes.

Note that files above the limit aren't synchronized and the tasks including the files leave the task list. The system transfers database definitions of the files only. You can find a record of the file not being synchronized in the Event log.

```
<add key="CMSMediaFileMaxStagingSize" value="102400" />
```

3. Save the web.config file.

8.46.10 Synchronization using API

In special cases, you may want to use Kentico CMS API to perform your own synchronization process. There are several methods you can use to work with synchronization.

DocumentEngine.DocumentSynchronizationHelper.LogDocumentChange	Creates synchronization tasks for the specified document or a set of documents under the specified path. The method returns the resulting tasks in a list. Loop through the list to synchronize the tasks.
CMS.Synchronization.SynchronizationHelper.LogObjectChange	Creates synchronization tasks for the specified object under the current site. The method returns the resulting tasks in a list. Loop through the list to synchronize the tasks.
CMSStaging.StagingHelper.RunSyn	Runs the synchronization of specified task for the specified

chronization	server or servers.

Here is an example code how to synchronize the content of the document "/Home" to server "Staging.Target1":

[C#]

```
using CMS.CMSHelper;
using CMS.DocumentEngine;
using CMS.SettingsProvider;
using CMS.Synchronization;

TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);
// Get the base document record
TreeNode node = tree.SelectSingleNode(CMSContext.CurrentSiteName, "/Home", "en-
us", false, null, false);
if (node != null)
{
    // Get the server
    ServerInfo si = ServerInfoProvider.GetServerInfo("Staging.Target1",
CMSContext.CurrentSiteID);
    if (si != null)
    {

        // Log the synchronization task
        List<ISynchronizationTask> tasks =
DocumentSynchronizationHelper.LogDocumentChange(node, TaskTypeEnum.UpdateDocument,
tree, false, tree, si.ServerID, null, false);

        // Loop through the list of tasks
        foreach (TaskInfo ti in tasks)
        {
            // Run the task synchronization
            StagingHelper.RunSynchronization(ti.TaskID, si.ServerID);
        }
    }
}
```

You can also use low level operations from TaskInfoProvider, SynchronizationInfoProvider and ServerInfoProvider to achieve synchronization. Though the previous operations are enough to perform any simple synchronization action.

The following example shows how you can synchronize the administrator user account. Synchronization of any other objects is done the same way using the API.

[C#]

```
using CMS.SiteProvider;
using CMS.Synchronization;
using CMS.CMSHelper;

// Gets the object
UserInfo userObj = UserInfoProvider.GetUserInfo("administrator");
```

```
if (userObj != null)
{
    // Gets the server
    ServerInfo si = ServerInfoProvider.GetServerInfo("Staging.Target1",
CMSContext.CurrentSiteID);
    if (si != null)
    {
        // Log the synchronization task
        List<ISynchronizationTask> tasks = SynchronizationHelper.LogObjectChange
(userObj.ObjectType, 0, userObj.UserlastModified, TaskTypeEnum.Object, true,
false, false, false ,false,  CMSContext.CurrentSiteID);

        // Loop through the list of tasks
        foreach (TaskInfo ti in tasks)
        {
            // Run the task synchronization
            StagingHelper.RunSynchronization(ti.TaskID, si.ServerID);
        }
    }
}
```

8.46.11 Staging internals and API

8.46.11.1 Overview

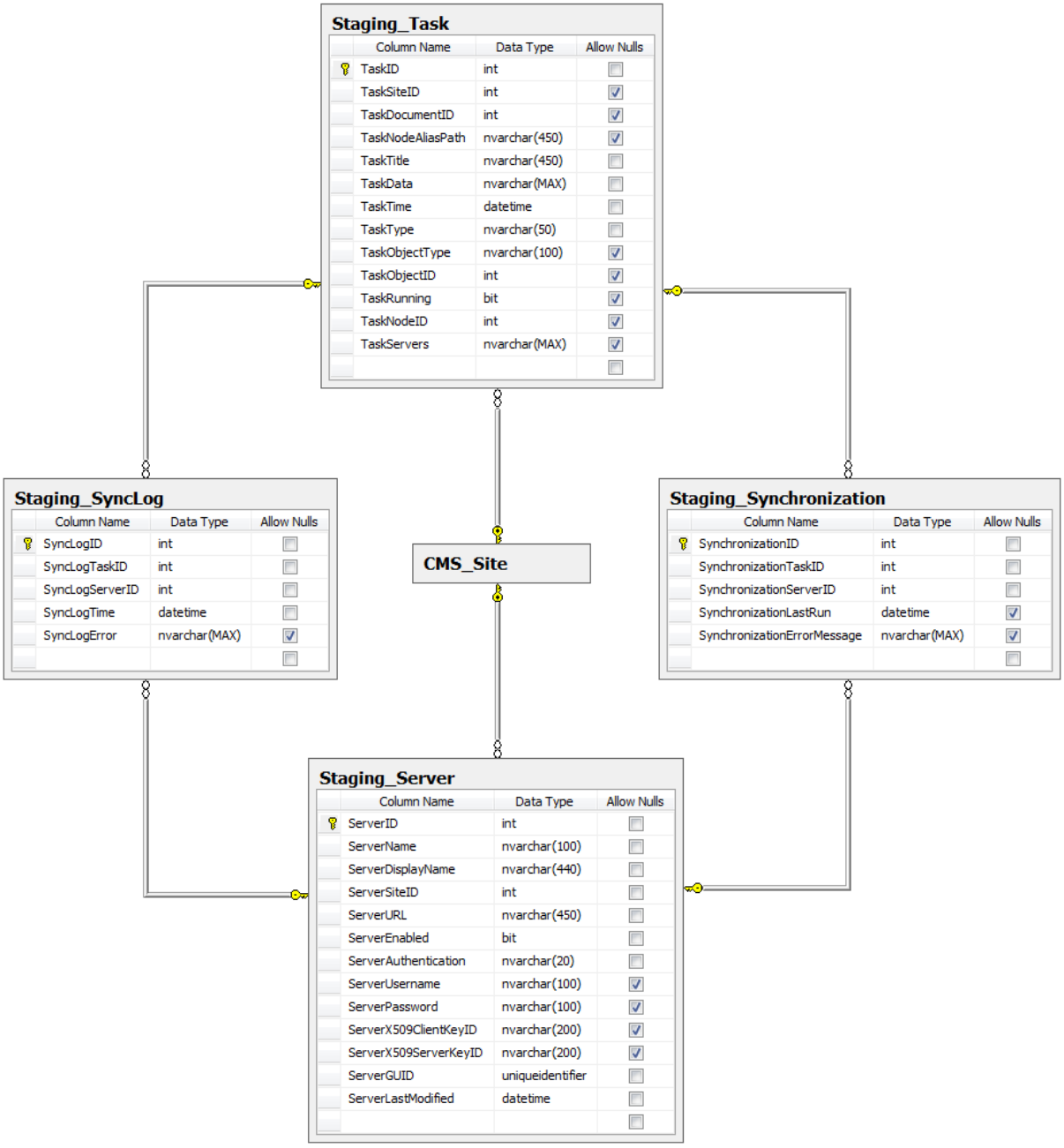
In this chapter, you will learn which database tables and API classes are used in the Staging module. You will also see the most common API examples.

Advanced API examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all methods from the module classes, please refer to Kentico CMS API Reference.

8.46.11.2 Database tables

The Staging module uses the following database tables:

- **Staging_Task** - contains records representing staging synchronization tasks.
- **Staging_SyncLog** - contains records representing performed synchronizations, while only those that produced an error are stored in this table.
- **Staging_Synchronization** - contains records representing performed synchronizations.
- **Staging_Server** - contains records representing staging servers.



8.46.11.3 API classes

If you are not familiar with database table data management through Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

> **Please note**
>
> The Staging module classes can be found in the **CMS.Synchronization** namespace.

Staging Server table API:

- **ServerInfo** - represents one staging server.
- **ServerInfoProvider** - provides functionality for management of staging servers.

Staging_Task table API:

- **TaskInfo** - represents one staging synchronization task.
- **TaskInfoProvider** - provides functionality for management of staging synchronization tasks.

8.46.11.4 API examples

8.46.11.4.1 Overview

In the following topics, you can find examples of you can use the Staging module API:

- Managing staging servers
- Managing staging tasks

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Tools\Staging\Default.aspx.cs***.

The Staging API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.CMSHelper;
using CMS.GlobalHelper;
using CMS.Synchronization;
using CMS.UIControls;
```

8.46.11.4.2 Managing staging servers

The following example demonstrates how a staging server can be created.

```
private bool CreateStagingServer()
{
    // Create new staging server object
    ServerInfo newServer = new ServerInfo();
```

```
    // Set the properties
    newServer.ServerDisplayName = "My new server";
    newServer.ServerName = "MyNewServer";
    newServer.ServerEnabled = true;
    newServer.ServerSiteID = CMSContext.CurrentSiteID;
    newServer.ServerURL = "http://localhost/KenticoCMS/CMSPages/SyncServer.asmx";
    newServer.ServerAuthentication = ServerAuthenticationEnum.UserName;
    newServer.ServerUsername = "admin";
    newServer.ServerPassword = "pass";

    // Save the staging server
    ServerInfoProvider.SetServerInfo(newServer);

    return true;
}
```

The following example gets and updates the staging server created by the code example above.

```
private bool GetAndUpdateStagingServer()
{
    // Get the staging server
    ServerInfo updateServer = ServerInfoProvider.GetServerInfo("MyNewServer",
CMSContext.CurrentSiteID);
    if (updateServer != null)
    {
        // Update the properties
        updateServer.ServerDisplayName = updateServer.ServerDisplayName.ToLower();

        // Save the changes
        ServerInfoProvider.SetServerInfo(updateServer);

        return true;
    }

    return false;
}
```

The following example demonstrates how to get multiple staging servers based on a WHERE condition and bulk update them.

```
private bool GetAndBulkUpdateStagingServers()
{
    // Prepare the parameters
    string where = "ServerName LIKE N'MyNewServer%'";

    // Get the data for the current site
    DataSet servers = ServerInfoProvider.GetSiteServers(CMSContext.CurrentSiteID,
where, null, -1, null, false);
    if (!DataHelper.DataSourceIsEmpty(servers))
    {
```

```
        // Loop through the individual items
        foreach (DataRow serverDr in servers.Tables[0].Rows)
        {
            // Create object from DataRow
            ServerInfo modifyServer = new ServerInfo(serverDr);

            // Update the properties
            modifyServer.ServerDisplayName =
modifyServer.ServerDisplayName.ToUpper();

            // Save the changes
            ServerInfoProvider.SetServerInfo(modifyServer);
        }

        return true;
    }

    return false;
}
```

The following example deletes the staging server created by the first code example on this page.

```
private bool DeleteStagingServer()
{
    // Get the staging server
    ServerInfo deleteServer = ServerInfoProvider.GetServerInfo("MyNewServer",
CMSContext.CurrentSiteID);

    // Delete the staging server
    ServerInfoProvider.DeleteServerInfo(deleteServer);

    return (deleteServer != null);
}
```

8.46.11.4.3 Managing staging tasks

The following example gets all staging tasks for a particular server and performs their synchronization.

```
private bool GetAndSynchronizeTasks()
{
    // Get server
    ServerInfo server = ServerInfoProvider.GetServerInfo("MyNewServer",
CMSContext.CurrentSiteID);

    if (server != null)
    {
        // Get tasks for the server
        DataSet tasks = TaskInfoProvider.SelectTaskList(CMSContext.CurrentSiteID,
server.ServerID, null, null);
```

```
        if (!DataHelper.DataSourceIsEmpty(tasks))
        {
            foreach (DataRow taskDr in tasks.Tables[0].Rows)
            {
                // Create task info object from data row
                TaskInfo task = new TaskInfo(taskDr);

                // Synchronize the task
                if (!string.IsNullOrEmpty(StagingHelper.RunSynchronization
(task.TaskID, server.ServerID)))
                {
                    apiGetAndSynchronizeTasks.ErrorMessage = "Synchronization
failed.";
                    return false;
                }
            }

            return true;
        }

        apiGetAndSynchronizeTasks.ErrorMessage = "No tasks found.";
    }

    return false;
}
```

The following example deletes all staging synchronization tasks for a particular server.

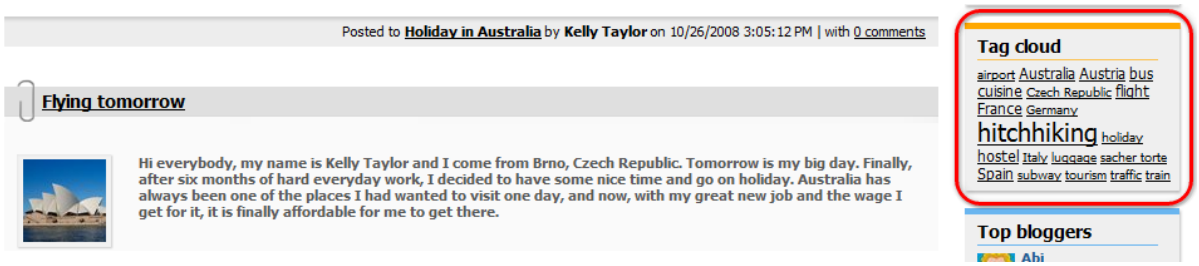
```
private bool DeleteTasks()
{
    // Get server
    ServerInfo server = ServerInfoProvider.GetServerInfo("MyNewServer",
CMSContext.CurrentSiteID);

    if (server != null)
    {
        // Get tasks for the server
        DataSet tasks = TaskInfoProvider.SelectTaskList(CMSContext.CurrentSiteID,
server.ServerID, null, null);

        if (!DataHelper.DataSourceIsEmpty(tasks))
        {
            foreach (DataRow taskDr in tasks.Tables[0].Rows)
            {
                // Create task info object from data row
                TaskInfo deleteTask = new TaskInfo(taskDr);

                // Delete the task
                TaskInfoProvider.DeleteTaskInfo(deleteTask);
            }

            return true;
```

```
        }

        apiDeleteTasks.ErrorMessage = "No tasks found.";
    }

    return false;
}
```

8.47 Tags

8.47.1 Overview

The Tags module enables users to tag documents with key words, called tags, depending on their content. Tags are associated with the documents and are convenient for their simple marking according to various criteria, e.g. your interests.



Please note that analogous to the Categories module, the Tags module enables you to categorize your documents.

- To learn how to tag individual documents, please refer to the Tagging documents topic.
- To learn how to display documents associated with the selected tag, please refer to the Using the Tag cloud web part topic.
- To learn how to create, alter or delete a tag group, please refer to the Managing tag groups topic.
- The internals and API of the Tags module are described in the Tags internals and API subchapter.

8.47.2 Tagging documents

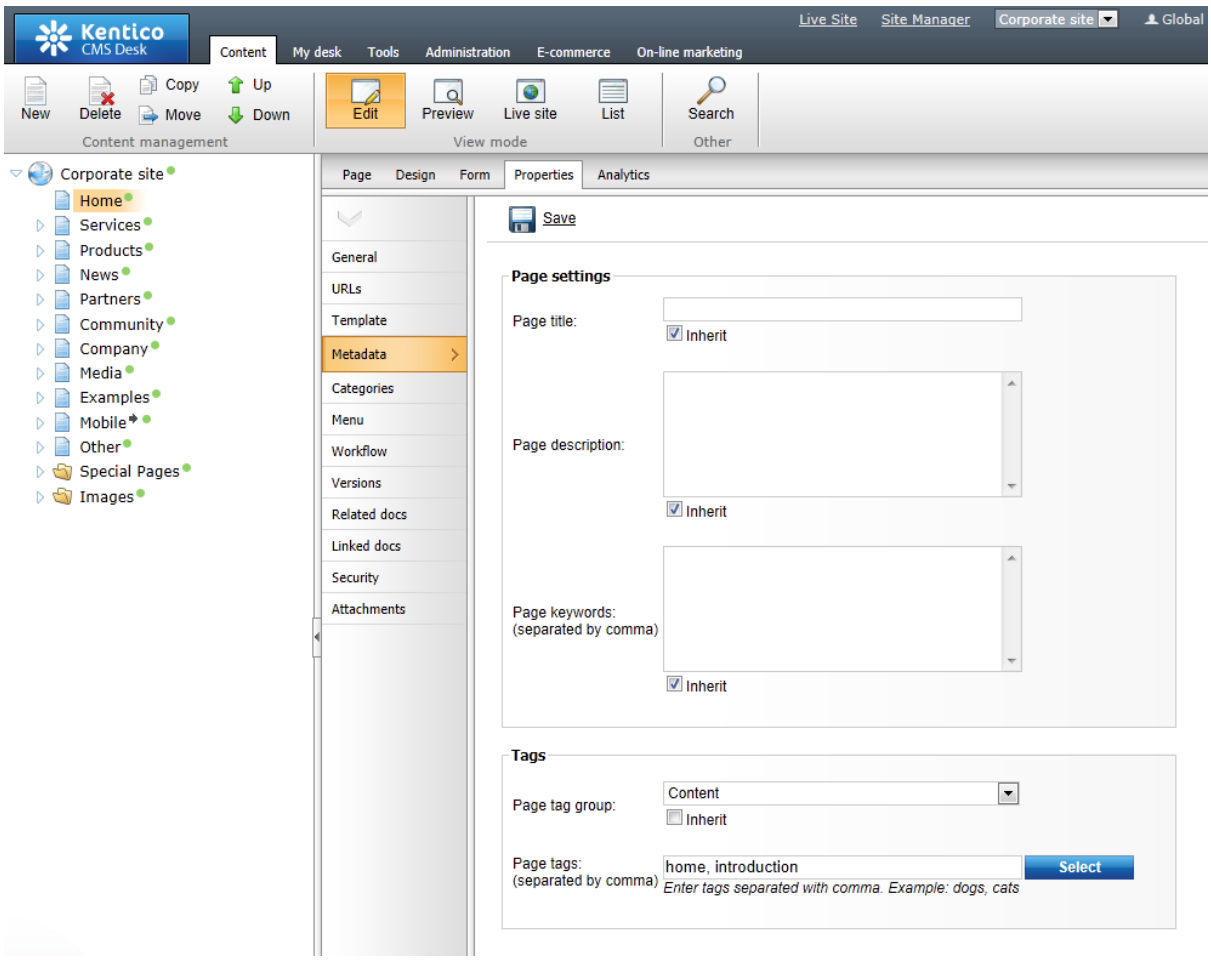
If you need to learn how to tag a document, the following examples describe this process. There are two ways of tagging documents:

- You can add tags to individual documents by using the **Properties** tab of the **CMS Desk -> Content -> Edit** interface (as described here).
- Or you can use the **Fields** tab in **Site Manager -> Development -> Document types -> Edit (✎) <*document_type*>** (as described here).

Tagging documents using Properties

1. Go to **CMS Desk**, switch to the **Edit** mode and select the document that you wish to tag from the content tree.

2. Switch to its **Properties -> Metadata** tab.



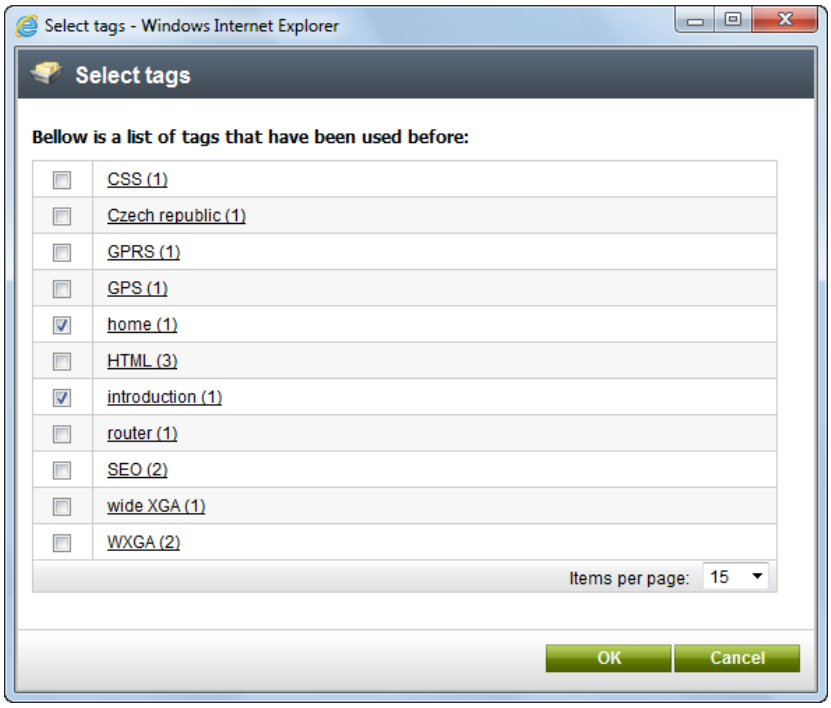
3. Choose a tag group using the **Page tag group** drop-down list. If you check the **Inherit** check-box, tag group will be inherited from the document's parent.

Please note

There needs to be **at least one tag group defined** for you to be able to tag a document. If there are no tag groups created, you will not be able to tag a document as each tag needs to belong to some tag group. You can learn about creating tag groups in the next chapter.

It is also recommended to select a tag group for the root of the content tree so that it could be inherited by the pages located under it.

4. Enter the tags into the **Page tags** field. Tags can be entered either manually or can be selected using the **Select** button. This button displays a list of all tags in the selected tag group. Check the appropriate tags' check-boxes and click **OK**.



5. Click **Save**. The entered tags will be saved and attached to the document.

Tag formats

When entering more than one tag into the **Page tags** field, the tags should be separated with a comma or a blank space. A combination of both in a single entry is also valid. The following examples are all valid entries for adding three tags - *tag1*, *tag2* and *tag3*:

tag1, tag2, tag3
tag1 tag2 tag3
tag1, tag2 tag3

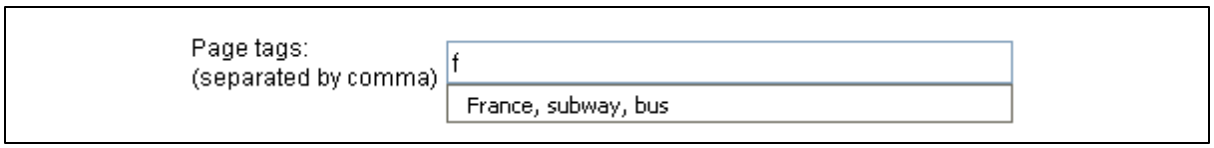
In case that you are entering a tag consisting of more than one word, you should enclose it within quotation marks. Multiple long tags can also be entered and can be also divided by both blank spaces and commas:

"long tag1", tag, "long tag2"
"long tag1" tag "long tag2"

Quotation marks can also be used for tags containing special characters that couldn't be used otherwise:

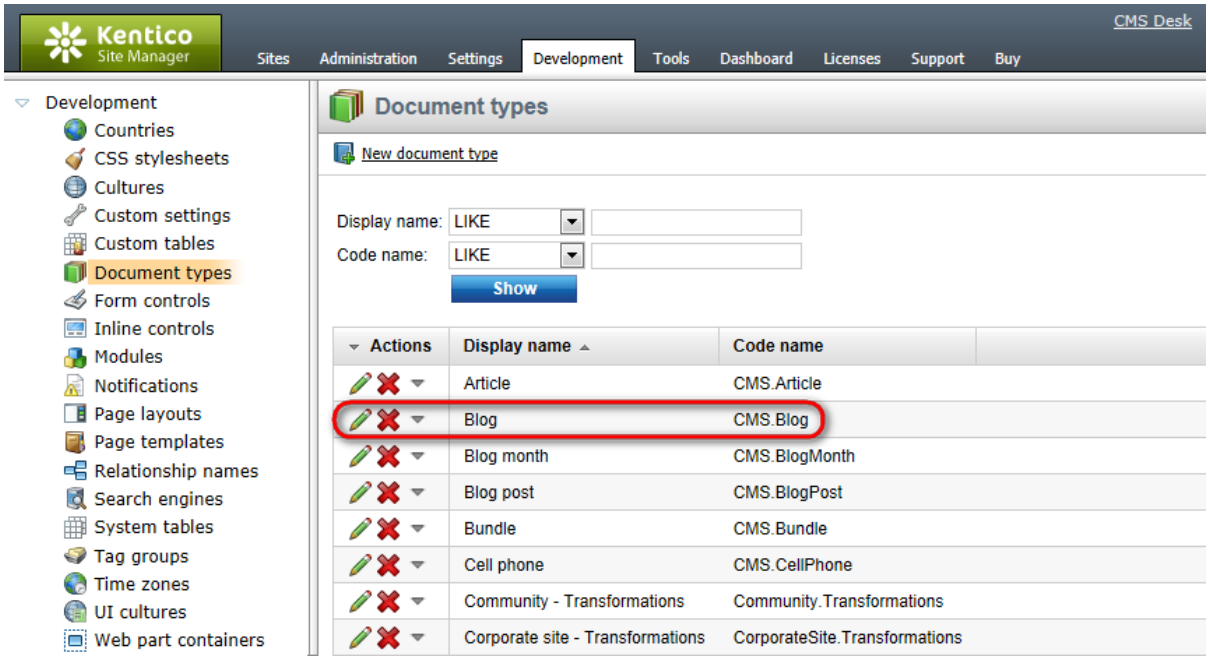
"tag@1", "tag#2", "long, strange: tag@#$"

The page tags field has also an insinuation function implemented. This functions offers you tags from the selected tag group while you are writing:



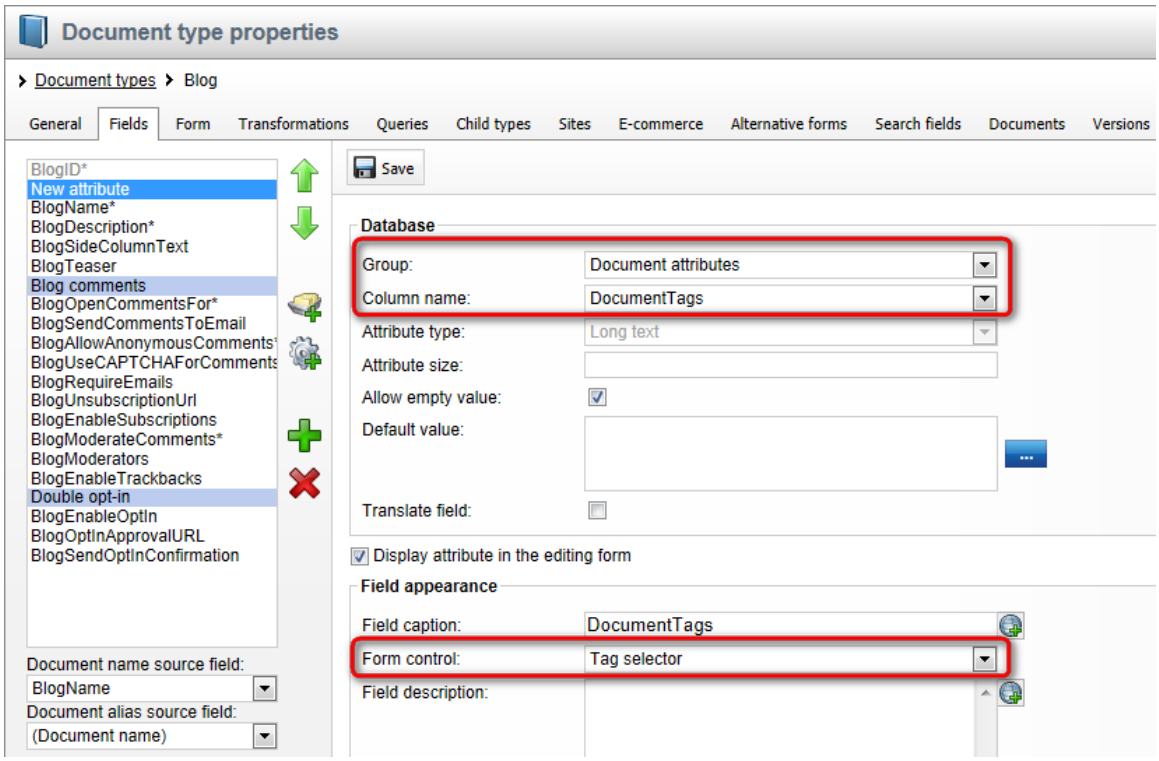
Tagging documents using Fields

1. Go to **Site Manager -> Development -> Document types** and choose to **Edit** ( ) a document type.



2. Switch to the **Fields** tab, click the **New system attribute** ( ) icon to create a new system attribute and select the following values from the corresponding drop-down lists:

- **Group:** Document attributes
- **Column name:** DocumentTags
- **Form control:** Tag selector



Click **Save** to confirm the changes.

Please note

If you need to create a custom document type with a tag selector, the tag selector must be bound to a system field.

8.47.3 Using the Tag cloud web part

The **Tag cloud** web part is used to display tags that are associated with the current document. Tags are displayed in the form of links and those with higher occurrence are displayed in higher font size. If the site visitor clicks on some of the links, a list of all documents tagged with the given tag will be displayed.

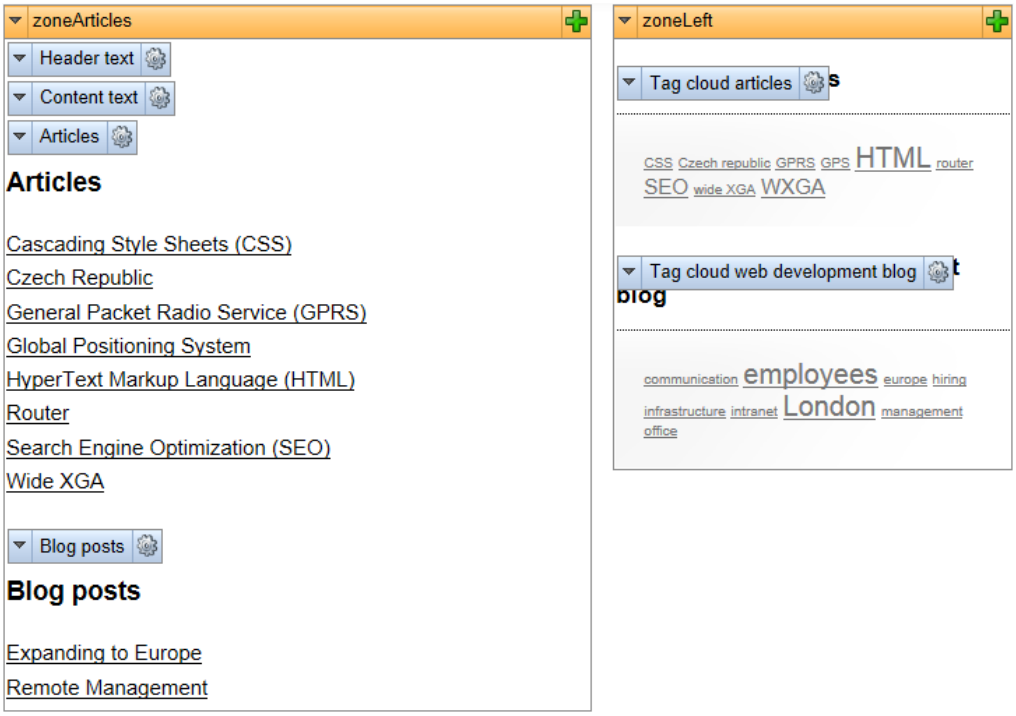
The web part has the following specific properties:

Tag filter	
Site name	Code name of the website from which you want to display the tags. If you leave the value empty, the content is retrieved from the current website.
Tag group name	Name of the tag group from which the tags will be displayed.
Select top N tags	Number of tags which will be displayed. Top tags according to the order specified by the *ORDER BY expression* property will be displayed.
ORDER BY expression	ORDER BY part of the SQL query used to retrieve the tags.
Tag cloud settings	
Document list URL	URL of the page on which the documents list will be displayed after clicking some of the tags (see below for an example).
Query string parameter name	Name of the parameter by means of which the tag ID will be transferred.
Tag separator	Separator that will be placed between tags in the tag cloud.
Minimal tag font size	Size of tags with the lowest occurrence.
Maximal tag font size	Size of tags with the highest occurrence.
Document filter	
Use document filter	Indicates if the document filter will be used. If true, the web part will display only tags used in the documents specified by the properties below.
Path	Path of the documents the tags of which will be displayed by the web part.
Combine with default culture	Indicates if tags from the default language version of the document should be displayed if the document is not translated to the current language.
Culture code	Culture version of the displayed tags.
Maximum nesting level	Maximum nesting level. It specifies the number of sub-levels in the content tree of the document specified by the *Path* property that should be included in the displayed content (i.e. whose tags will be displayed).
Select only published	Indicates if only tags from published documents should be displayed. Applied only when 'Site name' or 'Alias path' property is defined. If disabled, all tags from the selected tag group will be displayed.
Where condition	WHERE part of the SQL query used to retrieve tags.

Common usage

The Tag cloud is designed to "work in pair" with some repeater web part. When a tag link is clicked in the tag cloud, a list of all documents tagged with this tag is displayed in the repeater. The repeater can be placed either on the same page as the tag cloud or on some other page to that the site visitor will be redirected. You can see an example of this behavior on the Corporate Site sample website under

Examples -> Web parts -> Tagging & Categories -> Tag cloud.



In order for the two web parts to cooperate correctly, you have to correctly set some of their properties:

Tag cloud:

- The placement of the repeater is defined by the Tag cloud's **Document list URL** property. In case that the repeater is placed on the same page as the tag cloud, the value should be left blank. In case that it is placed on some other page, you should enter the **alias path** of that page.
- ID of the clicked tag is transferred to the repeater in form of a query string parameter. The name of the parameter can be set using the **Query string parameter name** property. The repeater displays the appropriate list of documents based on the value that it gets by via this parameter.

Repeater:

- Set the value of the **Path** parameter to the location in the content tree where the documents are stored.
- Set the value of the **Document types** parameter to the document type(s) that is (are) to be displayed.
- Select the transformations that you want to use for the **Transformation** and **Selected items transformation**.
- Finally, use the following code as a value for the repeater's **WHERE condition** parameter. The **tagid** value should be replaced by the name set in the Tag cloud's **Query string parameter name**:

```
({?tagid|(toint)?} = 0 AND '{?tagname?}'='')
```

```
OR (DocumentID IN (SELECT DocumentID FROM CMS_DocumentTag WHERE TagID = {?tagid|
(toint)?}))
OR (DocumentID IN (SELECT DocumentID FROM CMS_DocumentTag WHERE TagID IN (SELECT
TagID FROM CMS_Tag WHERE TagName = '{?tagname?}' AND TagGroupID = {?groupid|
(toint)?}))))
```

The first line ensures displaying of all documents when **no query string parameters** (tagid nor tagname nor taggroupid) are received. The second line ensures displaying of documents based on the received **tagid** (or differently named parameter) from a **Tag cloud** web part. The third line ensures displaying of documents based on the received **tagname** and **groupid** from a **blog post's** Filed under section.

If you would like to learn more on the use of the **Tag cloud** web part, please refer to Community Site Guide -> Part 2 -> Pre-development tasks -> Creating the tag groups. This is a step-by-step tutorial on how to create a tag group.

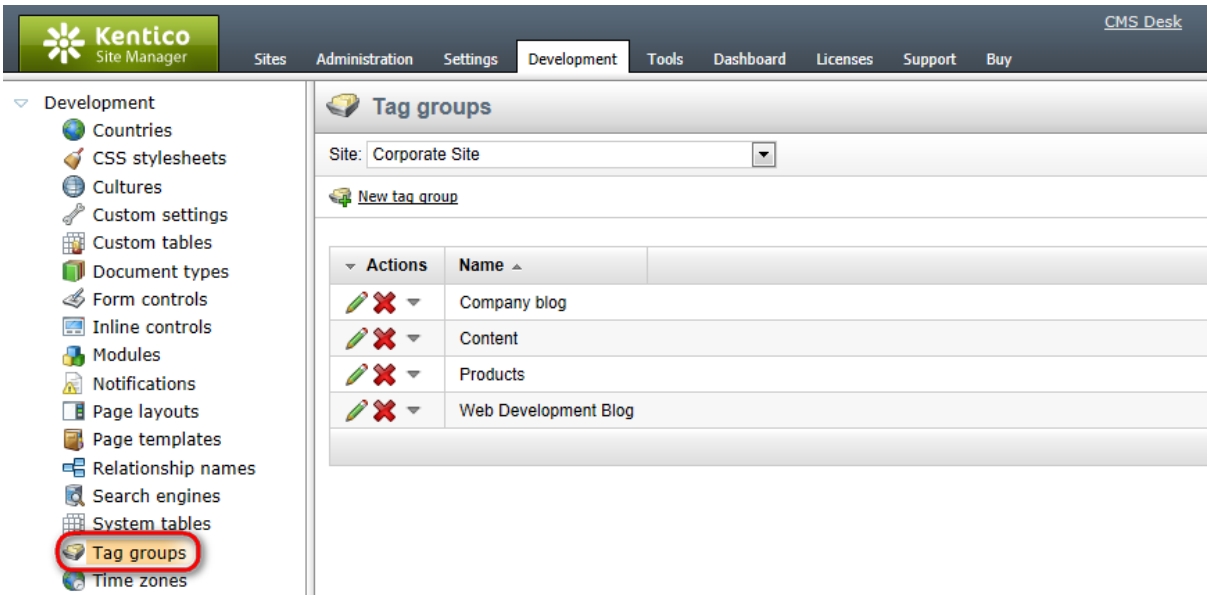
If you would like to see examples of how the **Tag cloud** web part is added to the site, please refer to:

- Creating the Blogs section -> Creating the Blogs page in the same section of Community Site Guide
- Creating the Blog posts page
- Creating the News page

You will need to have the Community Site sample website installed to follow Kentico CMS Community Site Guide.

8.47.4 Managing tag groups

Tags are divided into tag groups, which are topic-related groups of tags. No global tag groups are defined and each tag group is bound to a particular site. Tag groups can be created and managed by the global administrator in **Site Manager -> Development -> Tag groups**. You can **Edit** (✎) or **Delete** (✖) the listed tag groups in this section.



Please note

If you want to allow users to tag documents, at least one tag group must be defined.

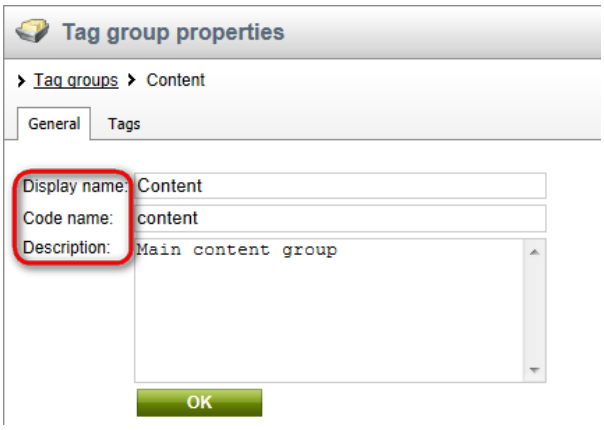
The default tag group **Content** is available for all Kentico CMS sample sites. It is inherited and should not under standard circumstances be deleted from the system.

Creating tag groups

In this example, you will learn how to create a new tag group.

1. To create a new tag group, go to **Site Manager -> Development -> Tag groups** and click **New tag group**. The following properties will have to be entered:

- **Display name** - display name of the tag group
- **Code name** - code name of the tag group
- **Description** - description text of the tag group



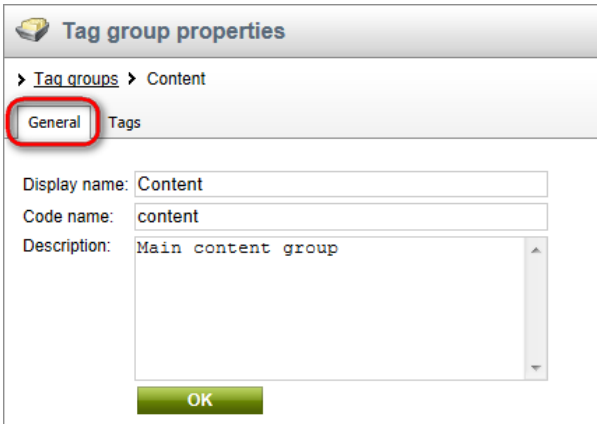
2. Click **OK** to confirm the values you have entered. The tag group will be created and become visible in the list.

Editing tag groups

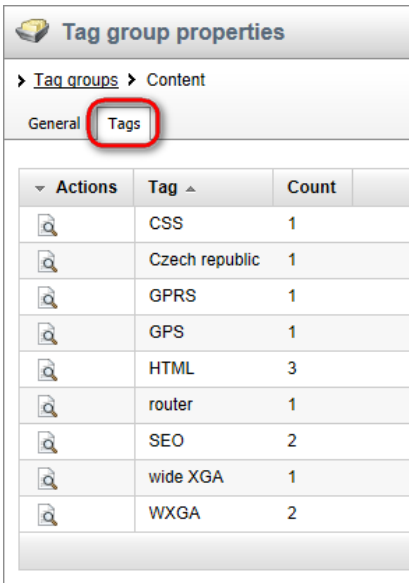
In this example, you will learn how to edit an existing tag group.

1. To edit an existing tag group, go to **Site Manager -> Development -> Tag groups** and choose to **Edit** ( ) one of the listed tag groups.

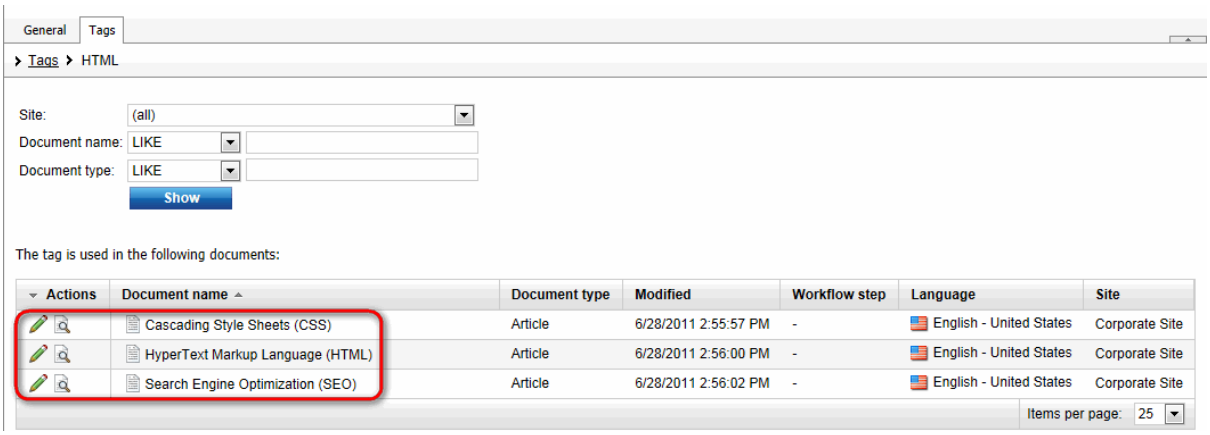
2. If you would like to change some of its properties, go to the **General** tab and change either the **Display name**, **Code name** or **Description** of the tag group.



3. If you would like to display the list of all tags in the selected tag group, switch to the **Tags** tab:



4. If you would like to display a list of links to documents tagged with a tag, click the icon next to the particular tag.



Please note

The global administrator cannot delete a tag from the document in **Site Manager**. This must be done by the user in **CMS Desk**.

8.47.5 Tags internals and API

8.47.5.1 Overview

In this chapter, you will learn which database tables and API classes are used in the Tags module. You will also see the most common API examples.

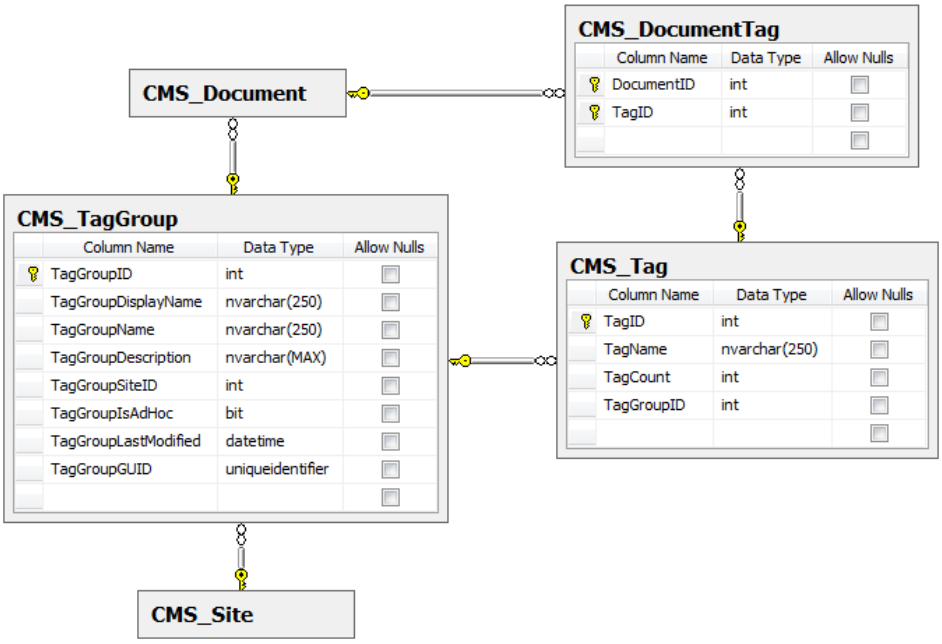
Advanced API examples can be found in the API programming and Kentico CMS internals section of this

guide. For detailed API documentation, such as a list of all methods from the module classes, please refer to Kentico CMS API Reference.

8.47.5.2 Database tables

The following database tables are used in the Tags module:

- **CMS_TagGroup** - contains records representing tag groups.
- **CMS_DocumentTag** - contains relationships between documents and tags indicating that particular documents are tagged with particular tags.
- **CMS_Tag** - contains records representing tags.



8.47.5.3 API classes

If you are not familiar with the database table data management by Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

The Tags module classes use the **CMS.SiteProvider** namespace.

CMS_TagGroup table API:

- **CMSTagGroupInfo** - represents one tag group.
- **CMSTagGroupInfoProvider** - provides management of tag groups.

CMS_DocumentTag table API:

- **CMSDocumentTagInfo** - represents a relationship between one document and one tag expressing that a particular document is tagged with a particular tag.
- **CMSDocumentTagInfoProvider** - provides management of relationships between documents and tags.

CMS_Tag table API:

- **CMSTagInfo** - represents one tag.
- **CMSTagInfoProvider** - provides management of tags.

8.47.5.4 API examples

8.47.5.4.1 Overview

Please note

All API examples can be simulated in the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Documents\Tags\Default.aspx.cs***.

8.48 Time zones

8.48.1 Overview

The Time zones module enables the configuration of time zones for the physical location of the server, for particular websites and even for particular users. This can be useful if your site has an international audience and you want the date and time displayed on your site to be correct for users from across the world.

Before it can be used, the module must be enabled as described in the Enabling the module topic.

To learn how to configure individual time zones in the system, please see the Managing time zones topic. Time zones can also be set to adjust their time according to Daylight saving time.

Time zones are currently supported in:

- **CMS Desk -> Content**
- **CMS Desk -> My desk**
- **CMS Desk -> Tools ->** Events
- **Web parts** of the Blogs, Forums, Message boards, Messaging and Smart search modules; more information can be found in the Use in web parts topic

The Time zones internals and API sub-chapter provides information about the database tables and classes used by the module and examples of how time zones can be managed using the API and how

correct time can be displayed by your code.

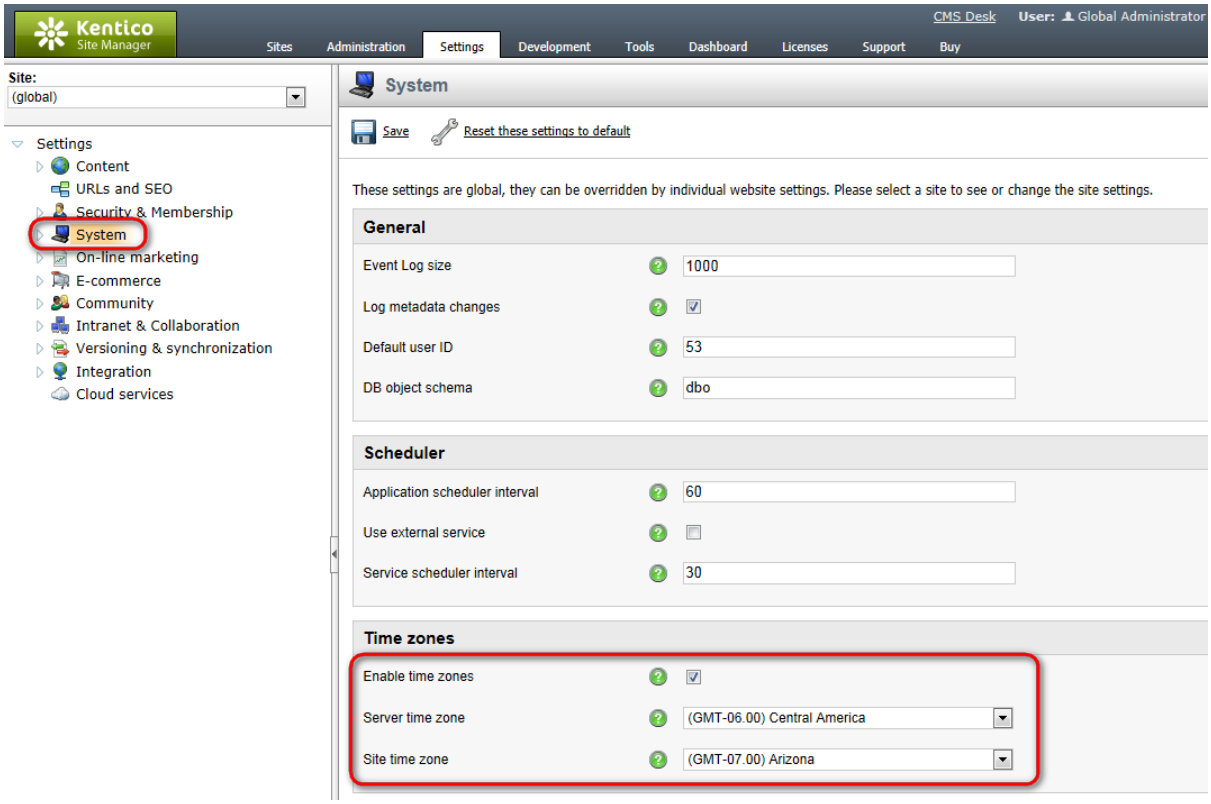
A typical example of use is displaying the time of forum posts when you have a global community – while the server may be located in New York (GMT -5:00), visitors coming from Paris (GMT +1:00) may see their new posts were added at 8am, while they would expect to see 2pm according to their current time.

Another example is a website of a global company that runs on a server in New York, but contains content for a French office. In this case, French visitors may wonder why the current time displayed by the server is 8am while it's 2pm in Paris. That's when you use the built-in support for multiple time zones.

8.48.2 Enabling the module

To enable the Time zones module, go to **Site Manager -> Settings -> System** and check the **Enable time zones** check-box. Below it, you can find the following two drop-down lists:

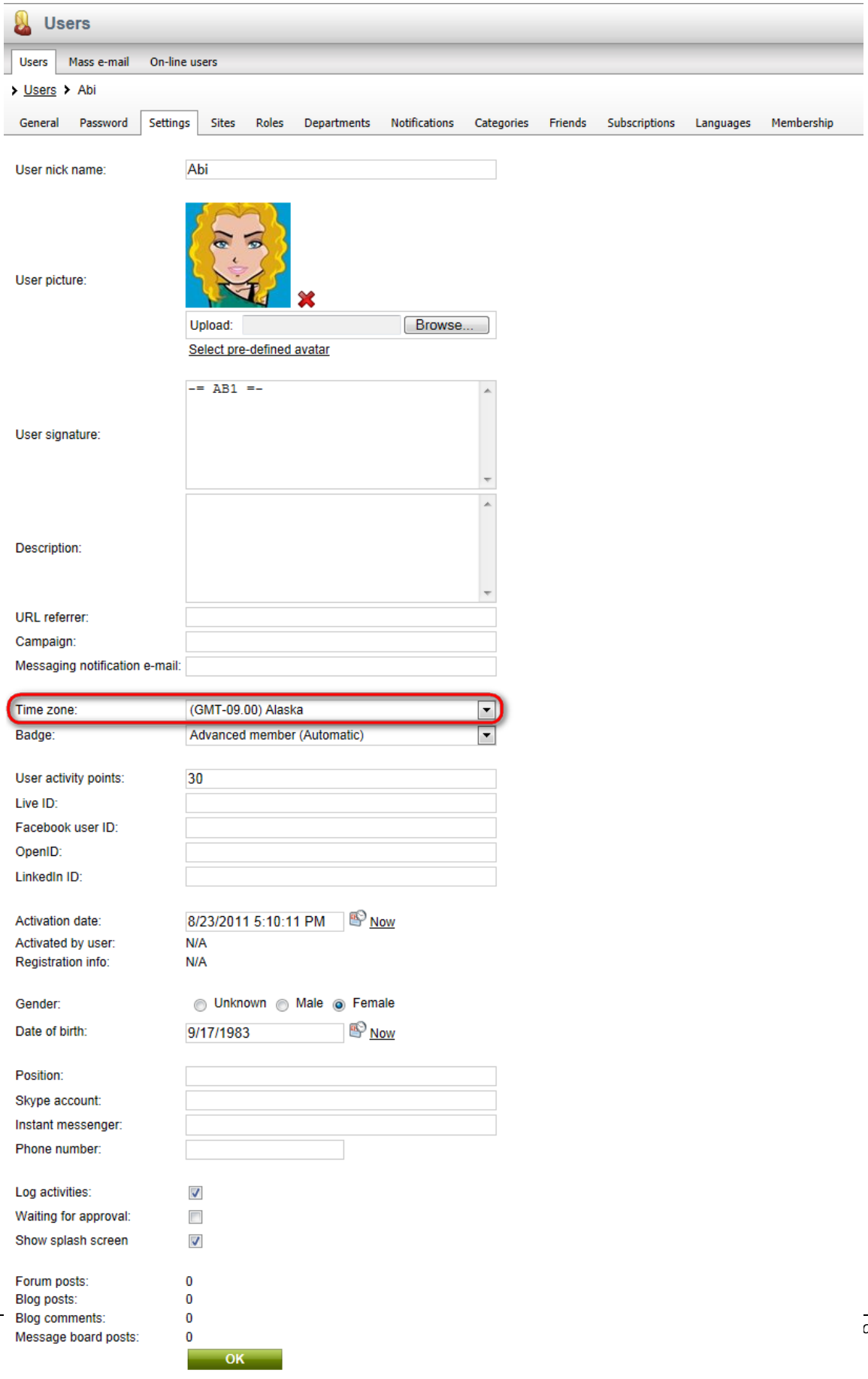
- **Server time zone** - time zone of the physical server location
- **Site time zone** - default time zone of the website selected by the **Site** drop-down list in the top-left part of the page; this time zone can also be set globally and inherited by sites that don't have it set differently



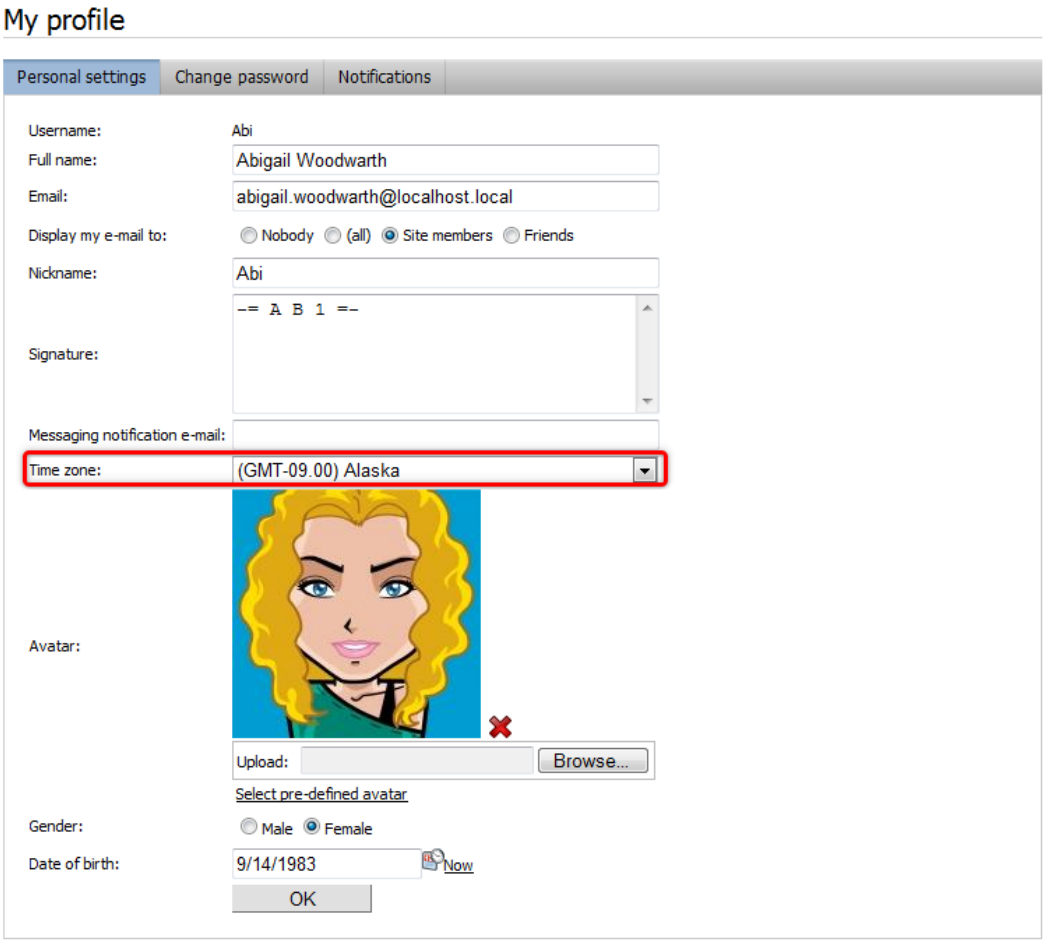
8.48.3 Setting user's time zone

Each user can have their own time zone settings. Where applicable, these time zone settings are used instead of the website's default time zone. A user's time zone can be set in both **CMS Desk** and **Site**

Manager, on the **Administration -> Users -> Edit user -> Settings** tab. On the tab, the selection can be done using the **Time zone** drop-down list.

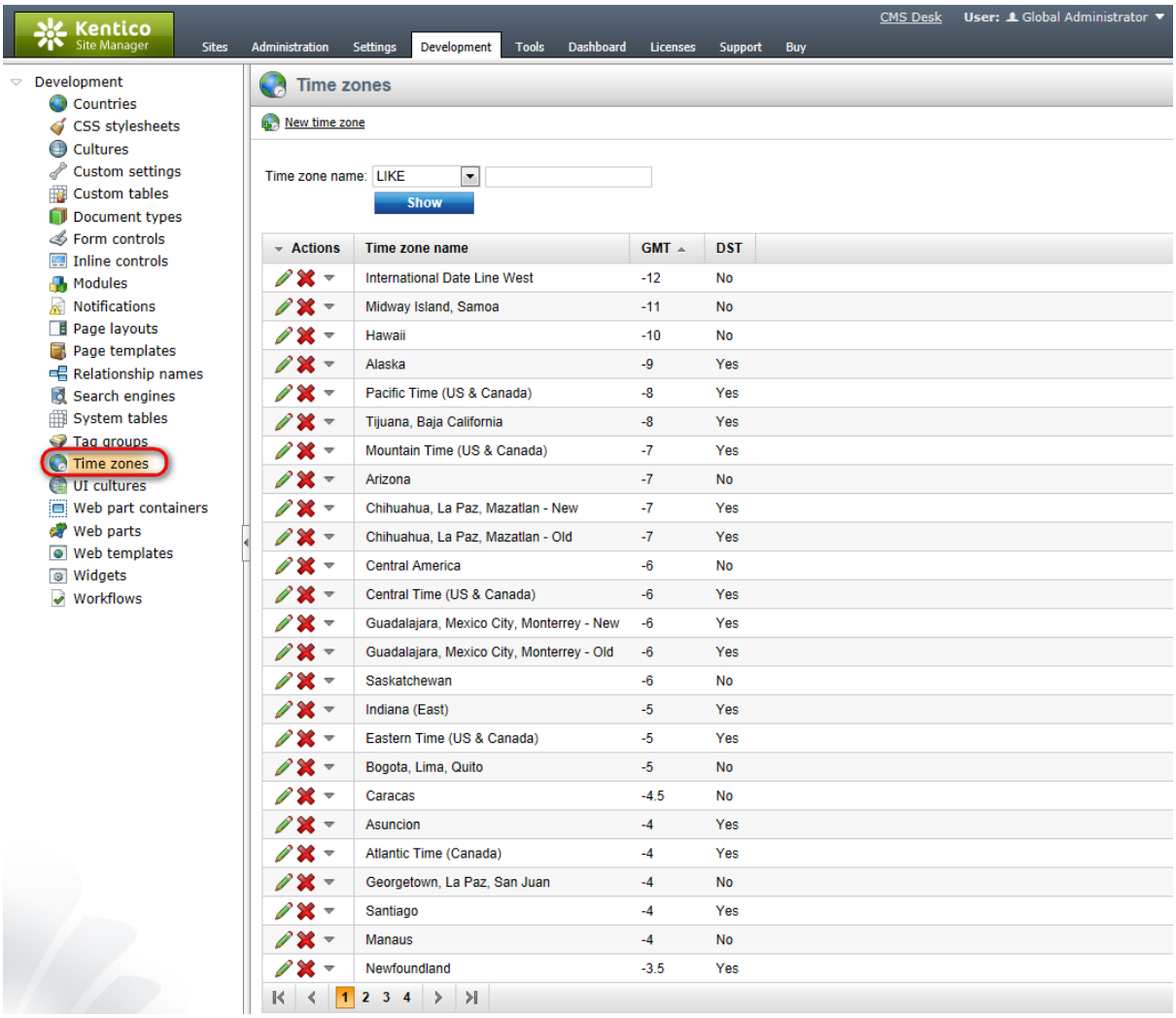

Users
Users
Mass e-mail
On-line users
Users
Abi
General
Password
Settings
Sites
Roles
Departments
Notifications
Categories
Friends
Subscriptions
Languages
Membership
User nick name:
Abi
User picture:
Upload:
Browse...
Select pre-defined avatar
User signature:
-= AB1 =-
Description:
URL referrer:
Campaign:
Messaging notification e-mail:
Time zone:
(GMT-09.00) Alaska
Badge:
Advanced member (Automatic)
User activity points:
30
Live ID:
Facebook user ID:
OpenID:
LinkedIn ID:
Activation date:
8/23/2011 5:10:11 PM
Now
Activated by user:
N/A
Registration info:
N/A
Gender:
Unknown
Male
Female
Date of birth:
9/17/1983
Now
Position:
Skype account:
Instant messenger:
Phone number:
Log activities:
Waiting for approval:
Show splash screen
Forum posts:
0
Blog posts:
0
Blog comments:
0
Message board posts:
0
OK

Users can also select their time zone on the live site if you place the **My Account** or **My Profile** web part on one of your pages. If they have access to **CMS Desk**, they can do this in **My Desk -> Account -> Details**. This is done the same way as above, by using the **Time zone** drop-down list.



8.48.4 Managing time zones

Time zones can be managed in **Site Manager -> Development -> Time zones**.



On this page, you can see a list of defined time zones. All time zones are displayed by default, but you can filter displayed items using the filter above the list. The only possible filtering parameter is the time zone **Display name**. If you type in the searched value and click the **Show** button, only those items that match the entered expression will be displayed in the list.

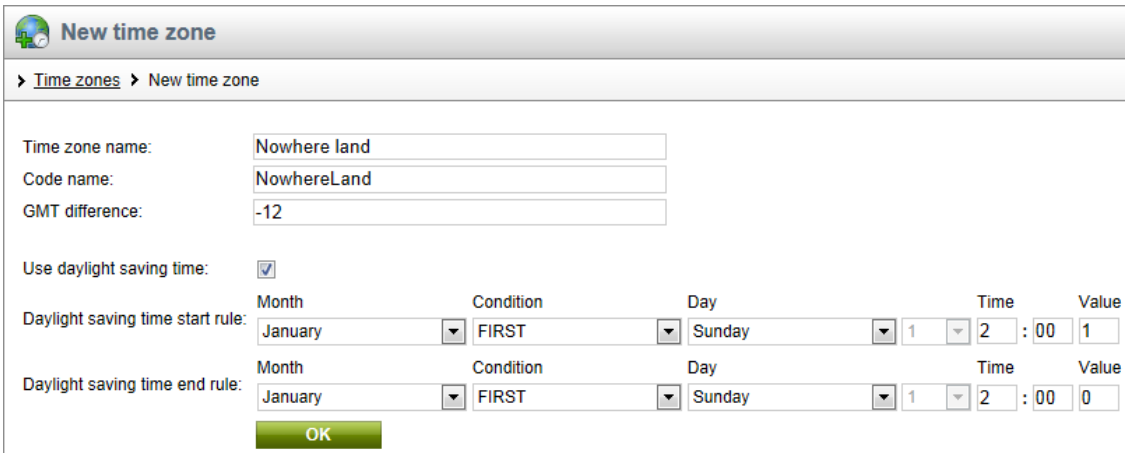
Time zones in the list can be **Edited** (✎) or **Deleted** (✖). You can also create **New time zone**.

Creating a new time zone

In the following example, you will learn how to create a new time zone:

1. Go to **Site Manager -> Development -> Time zones** and click **New time zone**.

2. Fill in the details that you can see in the following screenshot:



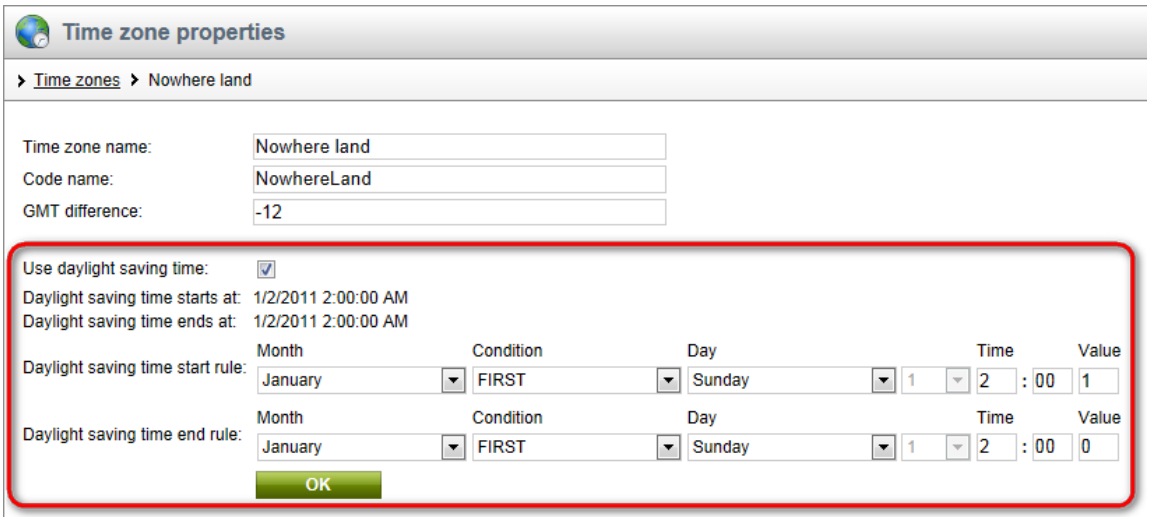
3. Click **OK**. You have just created the time zone. Now if you switch back to the time zones list, you should see the new time zone present among the records.

8.48.5 Daylight saving time

When creating a time zone or modifying some of the existing ones, you may come across the need to specify the daylight saving time (DST). This is a convention of setting clocks so that afternoons have more daylight and mornings have less of it. The amount of time advance and dates of change vary from country to country, however, it is usually a one hour advance at the beginning of spring and the advance is rolled back in autumn.

For more information about DST, please read this Wikipedia article: http://en.wikipedia.org/wiki/Daylight_saving_time

Daylight saving time can be set separately for each of the time zones. It can be set when creating a new time zone or when editing an existing one.



1. The first thing you need to do is to check the **Use daylight saving time** check-box. This enables DST for the time zone.

2. Now when you have DST enabled, you have to set the **DST start rule** and **DST end rule** for the current time zone. First, select the month in which the change will be carried out using the **Month** drop-down list.

3. Here comes the complicated part. You have to specify on which day of the selected month the change will be carried out. This is done by the **Condition** drop-down list and the two **Day** drop-down lists.

The following table explains the meanings of possible options for the **Condition** parameter:

FIRST	Day of the week can be selected. If you select Monday, the time advance will occur on the first Monday of the selected month.
LAST	Day of the week can be selected. If you select Monday, the time advance will occur on the last Monday of the selected month.
>=	Day of the week and day number can be selected. If you select Monday and 15, the time advance will occur on the first Monday after the 15th day of the selected month.
<=	Day of the week and day number can be selected. If you select Monday and 15, the time advance will occur on the last Monday before the 15th day of the selected month.
=	Day number can be selected. If you select 15, the time advance will occur on the 15th of the selected month.

4. Set the time when the change will occur on the specified date using the **Time** fields.

5. The last thing is to set the time difference between the standard time and DST. It should be entered into the **Value** field and represents the **difference from standard time in hours**. Use this value for the **DST start rule** and '**0**' for the **DST end rule**.

6. Click **OK** to save the settings.

8.48.6 Use in web parts

Web parts of the Blogs, Forums, Message boards, Messaging and Smart Search modules have the **Time zones** section in their **web part properties**, where the applied time zone can be set. The section contains the following two properties:

Time zone	• **Inherit** - inherits the time zone setting from the Page placeholder web part used to display the current page template (typically the one on the master page). • **Server** - server time zone settings will be used by the web part. • **Web site** - website time zone settings will be used by the web part. • **User** - time zone settings of individual users will be used by the web part. • **Custom** - some other time zone will be used based on the selection done in the *Custom time zone* property.
Custom time zone	If you select one of the time zones, it will be used by the content of this web part, regardless of current user or website time zone settings.

In the case of the **Calendar** and **Event calendar** web parts, these web part properties take no effect. Instead, displaying of the correct time zone needs to be ensured in the used transformation, as described in the Displaying correct time in your code chapter.

8.48.7 Time zones internals and API

8.48.7.1 Overview

In this chapter, you will learn which database tables and API classes are used in the Time zones module. You will also see the most common API examples.

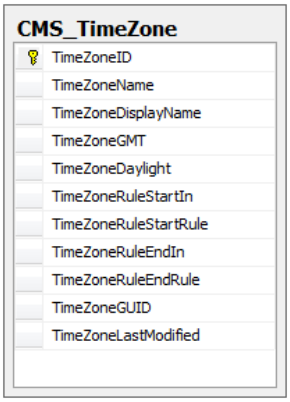
The Displaying correct time in your code topic demonstrates how the correct time can be displayed in transformations and by user controls depending on the current time zone settings.

Further API-related information and examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all members of the module's classes, please refer to Kentico CMS API Reference.

8.48.7.2 Database tables

The following database tables are used to store information for the time zones module:

- **CMS_TimeZone** - contains records representing time zones.



8.48.7.3 API classes

If you are not familiar with database table data management through Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

The classes of the Time zones module can be found in the **CMS.SiteProvider** namespace.

CMS_TimeZone table API:

- **TimZoneInfo** - represents one time zone object.
- **TimeZoneInfoProvider** - provides management functionality for time zones.

Other classes:

- **TimeZoneHelper** - can be used to convert the time according to a specified time zone or get other types of time zone data.

8.48.7.4 API examples

8.48.7.4.1 Overview

This topics shows examples of how the API of the Time zones module can be used:

- Managing time zones

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Development\TimeZones\Default.aspx.cs***.

The time zone API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.GlobalHelper;
using CMS.CMSHelper;
using CMS.SiteProvider;
```

8.48.7.4.2 Managing time zones

The following example creates a time zone.

```
private void CreateTimezone()
{
    // Create new timezone object
    TimeZoneInfo newTimezone = new TimeZoneInfo();

    // Set the properties
    newTimezone.TimeZoneDisplayName = "My new timezone";
    newTimezone.TimeZoneName = "MyNewTimezone";
    newTimezone.TimeZoneGMT = -12;
    newTimezone.TimeZoneDaylight = true;
    newTimezone.TimeZoneRuleStartRule = "MAR|SUN|1|LAST|3|0|1";
    newTimezone.TimeZoneRuleEndRule = "OCT|SUN|1|LAST|3|0|0";
    newTimezone.TimeZoneRuleStartIn = TimeZoneInfoProvider.CreateRuleDateTime
(newTimezone.TimeZoneRuleStartRule);
    newTimezone.TimeZoneRuleEndIn = TimeZoneInfoProvider.CreateRuleDateTime
(newTimezone.TimeZoneRuleEndRule);

    // Save the timezone
    TimeZoneInfoProvider.SetTimeZoneInfo(newTimezone);
}
```

The following example gets and updates a time zone.

```
private bool GetAndUpdateTimezone()
{
    // Get the timezone
    TimeZoneInfo updateTimezone = TimeZoneInfoProvider.GetTimeZoneInfo
("MyNewTimezone");
    if (updateTimezone != null)
    {
        // Update the properties
        updateTimezone.TimeZoneDisplayName =
updateTimezone.TimeZoneDisplayName.ToLower();

        // Save the changes
        TimeZoneInfoProvider.SetTimeZoneInfo(updateTimezone);
```

```
        return true;
    }

    return false;
}
```

The following example gets and bulk updates time zones.

```
private bool GetAndBulkUpdateTimezones()
{
    // Prepare the parameters
    string where = "TimeZoneName LIKE N'MyNewTimezone%'";

    // Get the data
    DataSet timezones = TimeZoneInfoProvider.GetTimeZones(where, null);
    if (!DataHelper.DataSourceIsEmpty(timezones))
    {
        // Loop through the individual items
        foreach (DataRow timezoneDr in timezones.Tables[0].Rows)
        {
            // Create object from DataRow
            TimeZoneInfo modifyTimezone = new TimeZoneInfo(timezoneDr);

            // Update the properties
            modifyTimezone.TimeZoneDisplayName =
modifyTimezone.TimeZoneDisplayName.ToUpper();

            // Save the changes
            TimeZoneInfoProvider.SetTimeZoneInfo(modifyTimezone);
        }

        return true;
    }

    return false;
}
```

The following example deletes a time zone.

```
private bool DeleteTimezone()
{
    // Get the timezone
    TimeZoneInfo deleteTimezone = TimeZoneInfoProvider.GetTimeZoneInfo
("MyNewTimezone");

    // Delete the timezone
    TimeZoneInfoProvider.DeleteTimeZoneInfo(deleteTimezone);

    return (deleteTimezone != null);
}
```

The following example gets the time according to the current user's time zone settings.

```
private bool ConvertTime()
{
    // Get user
    UserInfo user = UserInfoProvider.GetFullUserInfo
(CMSContext.CurrentUser.UserID);

    // If user exist
    if (user != null)
    {
        // Get converted time
        System.DateTime convertedTime = TimeZoneHelper.ConvertUserDateTime
(System.DateTime.Now, user);

        return true;
    }

    return false;
}
```

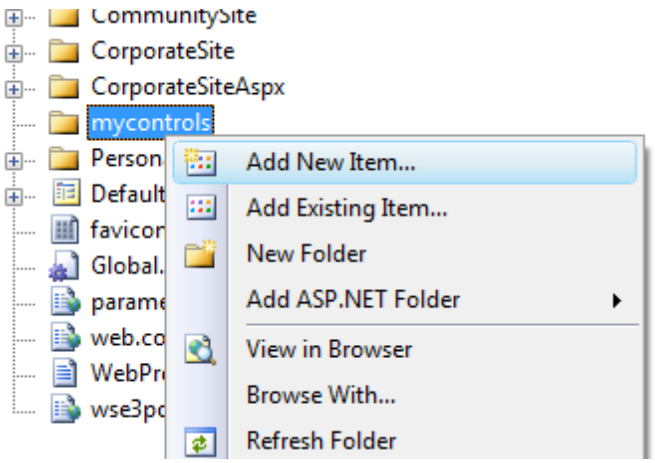
8.48.7.5 Displaying correct time in your code

The following methods can be used in transformation code to display the correct time according to time zone settings.

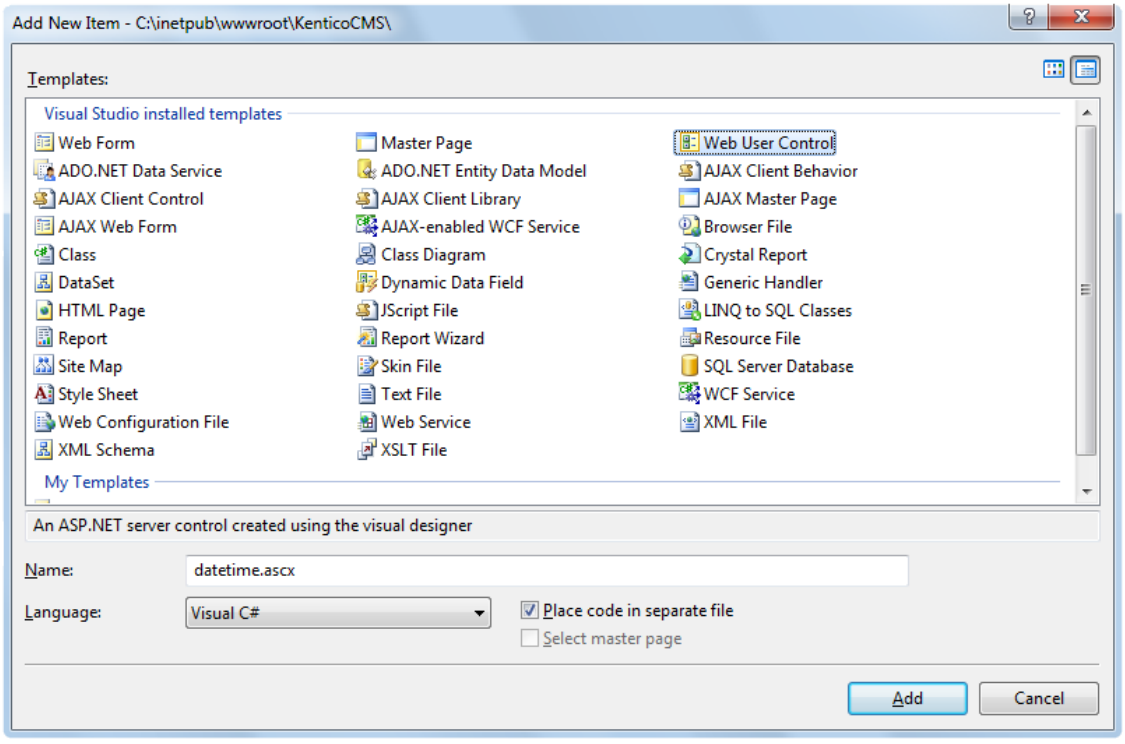
<%# GetDateTime (DateTime.Now) %>	Returns date-time value according to web part time zone settings.
<%# GetSiteDateTime (DateTime.Now) %>	Returns date-time according to site time zone settings.
<%# GetUserDateTime (DateTime.Now) %>	Returns date-time according to the current user's time zone settings.
<%# GetCustomDateTime (DateTime.Now, "GreenwichMeanTime") %>	Returns date-time according to the time zone given in the second parameter.

In the following example, you will learn how to use the **General -> User control** web part to display the current date and time on your site, which will reflect the time zone settings of the web part.

1. Open the web project in Visual Studio and create a new subfolder in the project folder. Name it *mycontrols*. Right click the folder and click Add new item.



2. Create a new WebUserControl in the mycontrols folder, name it *datetime.ascx*.



3. Edit the user control on the Design tab. Drag and drop a **Label** control onto the form.

4. Add the following code to the **PageLoad** method of the user control:

[C#]

```
Label1.Text = CMS.CMSHelper.CMSContext.ConvertDateTime(DateTime.Now, this)
.ToString();
```

5. Add the **General -> User control** web part somewhere to some page of your website. Set the

following properties of the web part:

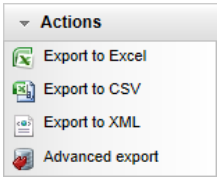
- **User control virtual path:** *~/mycontrols/datetime.ascx*
- **Time zone**: *Custom*
- **Custom time zone**: any time zone of your choice

and click **OK**. If you switch to the live site now, you should see the web part displaying the current date and time in the selected time zone. Now you can also try changing the value of the **Custom time zone** property and verify that the time displayed on the live site changes according to it.

8.49 UI data export

8.49.1 Overview

The UI data export functionality allows you to export data shown in various listings throughout the whole UI to XLSX, CSV or XML files. The export functionality is accessible in a context menu after clicking the ▾ icon in the **Actions** column header.



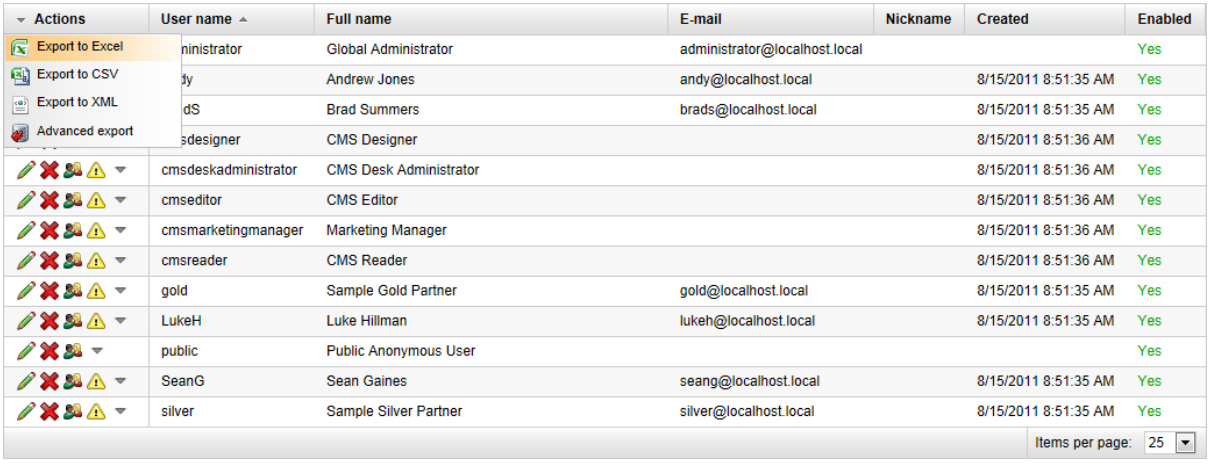
The Exporting UI data topic in this chapter provides you with information on the basics of this functionality. Export to the three formats mentioned above can either be performed in a single click with pre-defined options, or using the **Advanced export** dialog, which allows detailed configuration of export options. The Advanced export topic explains the options available in the dialog.

You can pre-define templates that will be used when exporting data to Excel files, which enables you to include graphics and additional text in the files. The Excel export templates topic explains how these templates can be created and where they should be uploaded to be used for specific object types. The CSV delimiters topic explains the use of appropriate separation characters in exported CSV files, necessary for appropriate displaying of these file in spreadsheet editors.

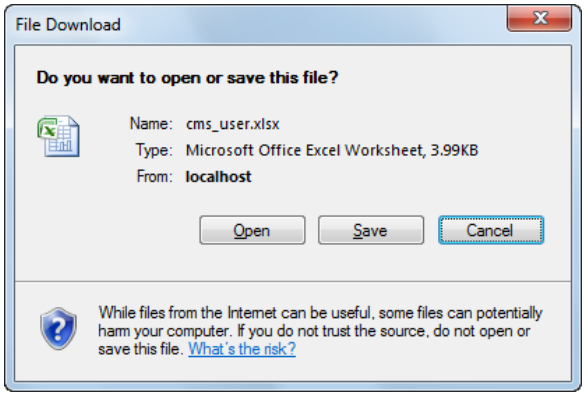
8.49.2 Exporting UI data

The UI data export functionality allows you to export data shown in various listings throughout the whole UI to XLSX, CSV or XML files. To achieve this, click the ▾ icon in the **Actions** column header. A drop-down menu will be displayed, offering the following options:

- **Export to Excel** - exports data shown on the current page of the listing to an XLSX spreadsheet.
- **Export to CSV** - exports data shown on the current page of the listing to a CSV file.
- **Export to XML** - exports data shown on the current page of the listing to an XML file.
- **Advanced export** - opens a dialog where export to the three formats mentioned above can be performed based on detailed settings. For more details, please refer to the Advanced export topic.



After executing an action from the drop-down menu, your browser's standard file download dialog pops-up, letting you open or save the file with exported data just as if you were downloading any other file.



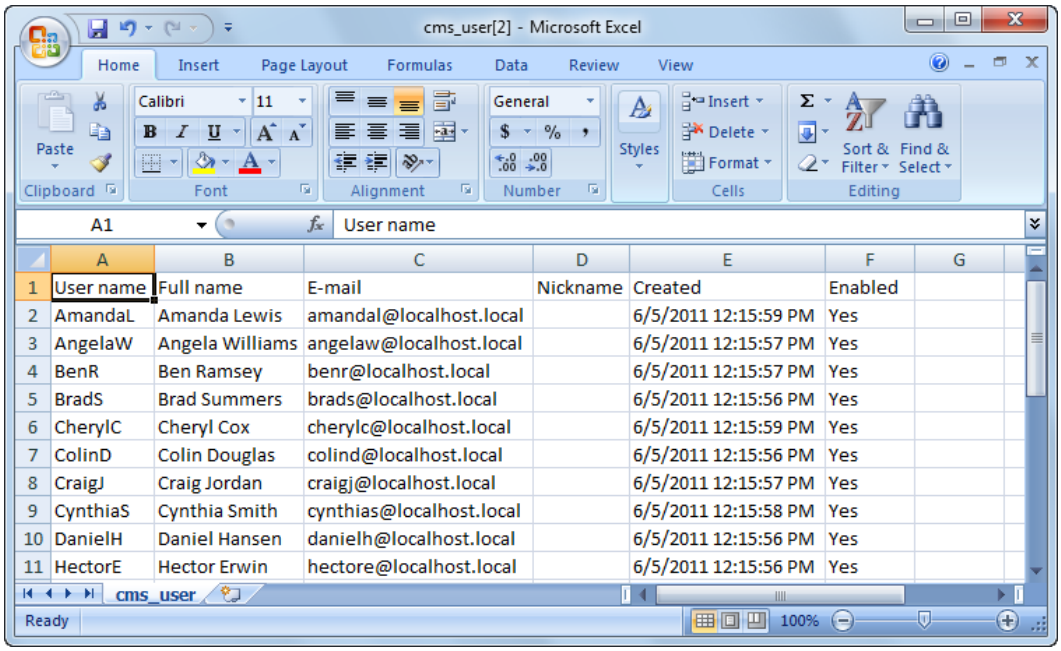
Export to Excel

Data exported to an XLSX spreadsheet are displayed just as in the listing. Spreadsheet columns represent the columns of the listing, while rows represent particular records shown in the rows of the listing.

Maximal exported string length

Please note that maximal length of text exported into a single cell of an Excel spreadsheet is **32767 (2^15 - 1) characters**. Longer strings are trimmed to match this length.

In the screenshot below, you can see the default appearance of an exported XLSX file. However, it is also possible to use customized templates for Excel export so that you can for example add graphics to the header of the spreadsheet. More details on this option can be found in the Excel export templates topic.



	A	B	C	D	E	F	G
1	User name	Full name	E-mail	Nickname	Created	Enabled	
2	AmandaL	Amanda Lewis	amandal@localhost.local		6/5/2011 12:15:59 PM	Yes	
3	AngelaW	Angela Williams	angelaw@localhost.local		6/5/2011 12:15:57 PM	Yes	
4	BenR	Ben Ramsey	benr@localhost.local		6/5/2011 12:15:57 PM	Yes	
5	BradS	Brad Summers	brads@localhost.local		6/5/2011 12:15:56 PM	Yes	
6	CherylC	Cheryl Cox	cherylc@localhost.local		6/5/2011 12:15:59 PM	Yes	
7	ColinD	Colin Douglas	colind@localhost.local		6/5/2011 12:15:56 PM	Yes	
8	CraigJ	Craig Jordan	craigj@localhost.local		6/5/2011 12:15:57 PM	Yes	
9	CynthiaS	Cynthia Smith	cynthias@localhost.local		6/5/2011 12:15:58 PM	Yes	
10	DanielH	Daniel Hansen	danielh@localhost.local		6/5/2011 12:15:56 PM	Yes	
11	HectorE	Hector Erwin	hectore@localhost.local		6/5/2011 12:15:56 PM	Yes	

Export to CSV

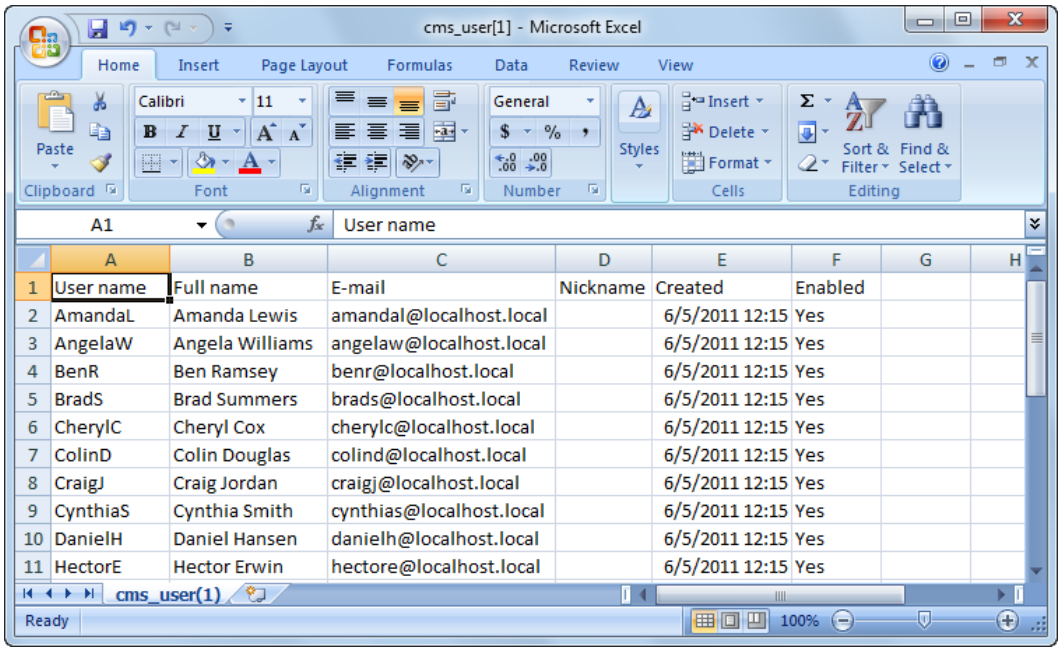
CSV is an abbreviation for Comma-separated values. It is a file format that stores tabular data in text form — each line represents one row of data, while particular values (columns) in each row are separated by a comma (,) or a semicolon (;). If a column value contains the delimiter, the whole value is wrapped in double quotation marks in CSV (e.g. the value *one,two* is exported as *"one,two"*). If a column value contains double quotation marks, they are backslashed in CSV (e.g. the value *"one* is exported as *\"one*).

A comma is used as a column delimiter by default if you select the **Export to CSV** action, while you can choose between the comma and the semicolon in the **Advanced export** dialog. See the CSV delimiters topic for more information.

The plain text form of exported data stored in a CSV file looks as follows:

```
User name,Full name,E-mail,Nickname,Created,Enabled
AmandaL,Amanda Lewis,amandal@localhost.local,,5/6/2011 12:43:26 PM,Yes
AngelaW,Angela Williams,angelaw@localhost.local,,5/6/2011 12:43:25 PM,Yes
BenR,Ben Ramsey,benr@localhost.local,,5/6/2011 12:43:24 PM,Yes
BradS,Brad Summers,brads@localhost.local,,5/6/2011 12:43:24 PM,Yes
CherylC,Cheryl Cox,cherylc@localhost.local,,5/6/2011 12:43:26 PM,Yes
...
```

If the correct delimiter is used, the file can also be opened in a spreadsheet editor such as Microsoft Excel.



	A	B	C	D	E	F	G	H
1	User name	Full name	E-mail	Nickname	Created	Enabled		
2	AmandaL	Amanda Lewis	amandal@localhost.local		6/5/2011 12:15	Yes		
3	AngelaW	Angela Williams	angelaw@localhost.local		6/5/2011 12:15	Yes		
4	BenR	Ben Ramsey	benr@localhost.local		6/5/2011 12:15	Yes		
5	BradS	Brad Summers	brads@localhost.local		6/5/2011 12:15	Yes		
6	CherylC	Cheryl Cox	cherylc@localhost.local		6/5/2011 12:15	Yes		
7	ColinD	Colin Douglas	colind@localhost.local		6/5/2011 12:15	Yes		
8	CraigJ	Craig Jordan	craigj@localhost.local		6/5/2011 12:15	Yes		
9	CynthiaS	Cynthia Smith	cynthias@localhost.local		6/5/2011 12:15	Yes		
10	DanielH	Daniel Hansen	danielh@localhost.local		6/5/2011 12:15	Yes		
11	HectorE	Hector Erwin	hectore@localhost.local		6/5/2011 12:15	Yes		

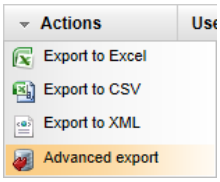
Export to XML

The code sample below illustrates how exported data are stored in XML files. There is always the *NewDataSet* root element, which contains a number of *Table* elements. Each of the *Table* elements represents a single data record. Its sub-elements represent particular columns of the table, have names identical to the physical database column names and contain the respective values.

```
<?xml version="1.0" encoding="windows-1250"?>
<NewDataSet>
  <cms_user>
    <UserName>AmandaL</UserName>
    <FullName>Amanda Lewis</FullName>
    <Email>amandal@localhost.local</Email>
    <Nickname />
    <Created>5/6/2011 12:43:26 PM</Created>
    <Enabled>Yes</Enabled>
  </cms_user>
  <cms_user>
    <UserName>AngelaW</UserName>
    <FullName>Angela Williams</FullName>
    <Email>angelaw@localhost.local</Email>
    <Nickname />
    <Created>5/6/2011 12:43:25 PM</Created>
    <Enabled>Yes</Enabled>
  </cms_user>
  ...
</NewDataSet>
```

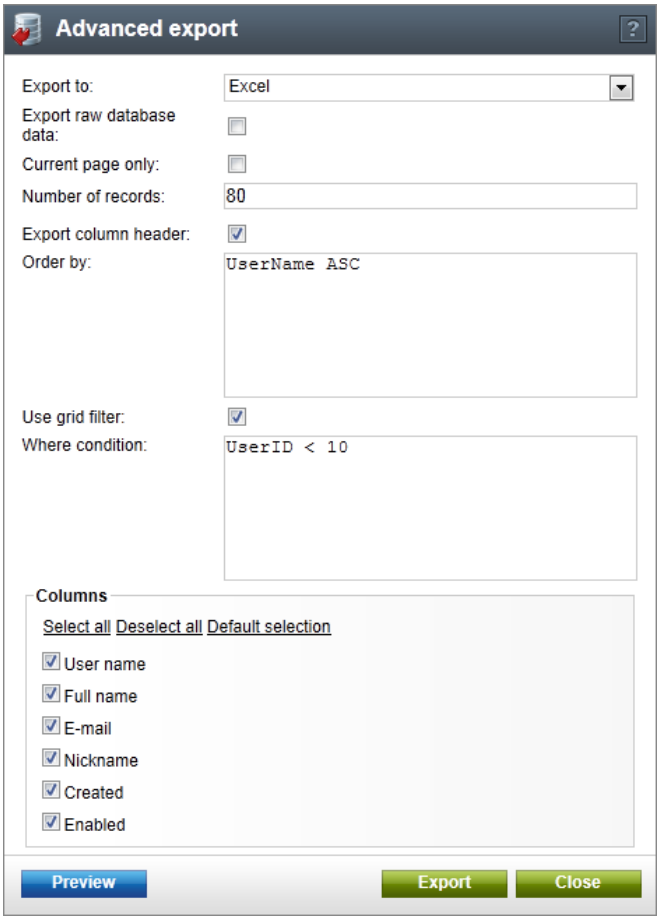
8.49.3 Advanced export

Clicking the **Advanced export** option can raise two different versions of the **Advanced export** dialog, depending on the type of user performing the operation.



Advanced export - global administrator

When the **Advanced export** option is selected by a global administrator, the dialog depicted in the following screenshot is displayed.



The following options can be adjusted in the dialog:

Export to	Type of file to which the data will be exported. You can choose from the following: • **Excel** - exports data to an XLSX spreadsheet.

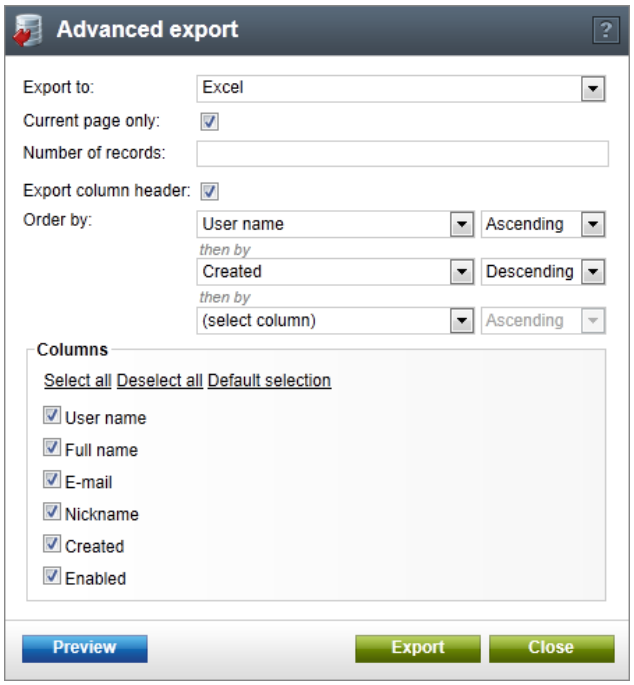
	• **CSV** - exports data to a CSV file. • **XML** - exports data to an XML file.
Delimiter	Displayed only when export to CSV is selected. Determines the value separation character used by the CSV file. The character can either be a comma (,) or a semicolon (;). See the CSV delimiters topic for more details.
Export raw database data	The format of data displayed in listings is not always identical to actual data stored in respective database columns. E.g. the **Enabled** column in user listing shows *Yes/No* values, while boolean *True/False* values are actually stored in the **UserEnabled** column of the **CMS_User** table. With this option enabled, raw data will be exported exactly as stored in the database and column headers will be identical to the database column names. If disabled, column headers and the actual data will be exported as shown in the listing.
Current page only	If enabled, only records displayed on the current page of the listing will be exported. If disabled, all records on all pages of the listing will be exported. The number of exported records in both cases can be limited by means of the **Number of records** value below.
Number of records	Specifies the number of data records that will be exported. Empty value results in all records being exported. The specified number of exported records can be limited by enabling the **Current page only** option above.
Export column header	If enabled, column headers will be included in the exported file. This option is not available when exporting to XML as column header names are used as names of the respective XML elements.
Order by	Standard SQL ORDER BY expression. Exported items will be ordered according to this expression in the exported file. Please note that you can select only columns that have a corresponding physical column in the database.
Use grid filter	If enabled, only data matching the filter above the listing will be exported. If disabled, all data will be exported regardless of the filter settings.
Where condition	Standard SQL WHERE condition. Only items matching this condition will be included in the export file.
Columns	In this section, you can use the check-boxes to choose which columns will be included in the export file. Bulk column selection can be performed using the following link buttons: • **Select all** - selects all columns listed in this section. • **Deselect all** - deselects all columns listed in this section. • **Default selection** - selects columns that were selected initially when the dialog was opened. If the **Export raw database data** option is enabled, listed column names are identical to the actual database columns name. If the option

	is disabled, column names are shown as in the listing.

When you have the options adjusted, you can click **Export** to get the data exported to the chosen type of file. By clicking the **Preview** button, you get a maximum number of 100 records exported (unless you set a smaller number by means of the **Current page only** or **Number of records** options). This is useful in case that you are exporting large amounts of records and want to get a quick preview of the exported file without waiting for all records to be exported.

Advanced export - standard user

When the **Advanced export** option is selected by a standard user who is not a global administrator, a dialog with simplified and more secure options (compared to the global administrator version described above) is displayed to them.



The following options can be adjusted in the simplified dialog:

Export to	Type of file to which the data will be exported. You can choose from the following: • **Excel** - exports data to an XLSX spreadsheet. • **CSV** - exports data to a CSV file. • **XML** - exports data to an XML file.
Delimiter	Displayed only when export to CSV is selected. Determines the value separation character used by the CSV file. The character can either be a comma (,) or a semicolon (;). See the CSV delimiters topic for more details.
Current page only	If enabled, only records displayed on the current page of the listing will be exported. If disabled, all records on all pages of the listing will be

	exported. The number of exported records in both cases can be limited by means of the **Number of records** value below.
Number of records	Specifies the number of data records that will be exported. Empty value results in all records being exported. The specified number of exported records can be limited by enabling the **Current page only** option above.
Export column header	If enabled, column headers will be included in the exported file. This option is not available when exporting to XML as column header names are used as names of the respective XML elements.
Order by	In this section, you can determine how exported records will be ordered in the exported file. Initially, there are two drop-down lists displayed. In the first one, all columns that have a corresponding physical column in a database are offered. The second one determines if values will be exported in ascending or descending order according to values in the selected column. Once you make your choice, another two drop-down lists are displayed below, allowing you to choose how records will be ordered in case of identical values in the column chosen above. After selecting, two other drop-downs are displayed below again. In the end, you can get as many pairs of drop-down lists as the number of offered columns.
Columns	In this section, you can use the check-boxes to choose which columns will be included in the export file. Bulk column selection can be performed using the following link buttons: • **Select all** - selects all columns listed in this section. • **Deselect all** - deselects all columns listed in this section.

8.49.4 Excel export templates

It is possible to customize the default appearance of exported XLSX templates. On each export, the CMS searches for a *Template.xlsx* file in the following locations:

1. **~\App_Data\CMSModules\DataExport\<site code name>\<object type>\Template.xlsx**
2. **~\App_Data\CMSModules\DataExport\<object type>\Template.xlsx**
3. **~\App_Data\CMSModules\DataExport\<site code name>\Template.xlsx**
4. **~\App_Data\CMSModules\DataExport\Template.xlsx**

The *DataExport* folder is not present under *~\App_Data\CMSModules* by default, so you have to create it and all its required sub-folders manually in case that you want to use custom templates. The *<object type>* folder name should be identical to the name of the exported file, e.g. *cms_user* for user listings as the exported file name is *cms_user.xlsx* (the actual object type name is *cms.user*, but dots are replaced with underscores in the file names). The *<site code name>* folder is only searched when exporting site-related objects, not for global objects that can be shared among all websites in the system.

The CMS searches for the templates with the priorities as stated above. This means that when exporting listings of an object type on a specified website, the path stated in 1. is searched first. If the

Template.xlsx file is not found there, it searches location 2, and so on. This allows you to have dedicated templates for each object type and website in your system.

Please note

If the *Template.xlsx* file is not found in any of the locations, the default template is used. The same happens if the template is opened for editing at the time of export or if the current user doesn't have the Read and Write permissions for the template file (on operating system level).

Custom data export folder

Excel export templates can also be stored at a different location than the default ~*AppData**CMSModules**DataExport*. The custom location can be defined by adding the following key to the *AppSettings* section of the *web.config* file:

```
<add key="CMSDataExportTemplateFolder" value="\\server1\MyDataExportTemplates" />
```

As the value of the key, you can either use a local disk path (e.g. *C:\MyDataExportTemplates*) or a UNC path (e.g. *\\server1\MyDataExportTemplates*). Using the UNC path may be useful in cases when you want to share the same templates between several Kentico CMS instances running on separate servers.

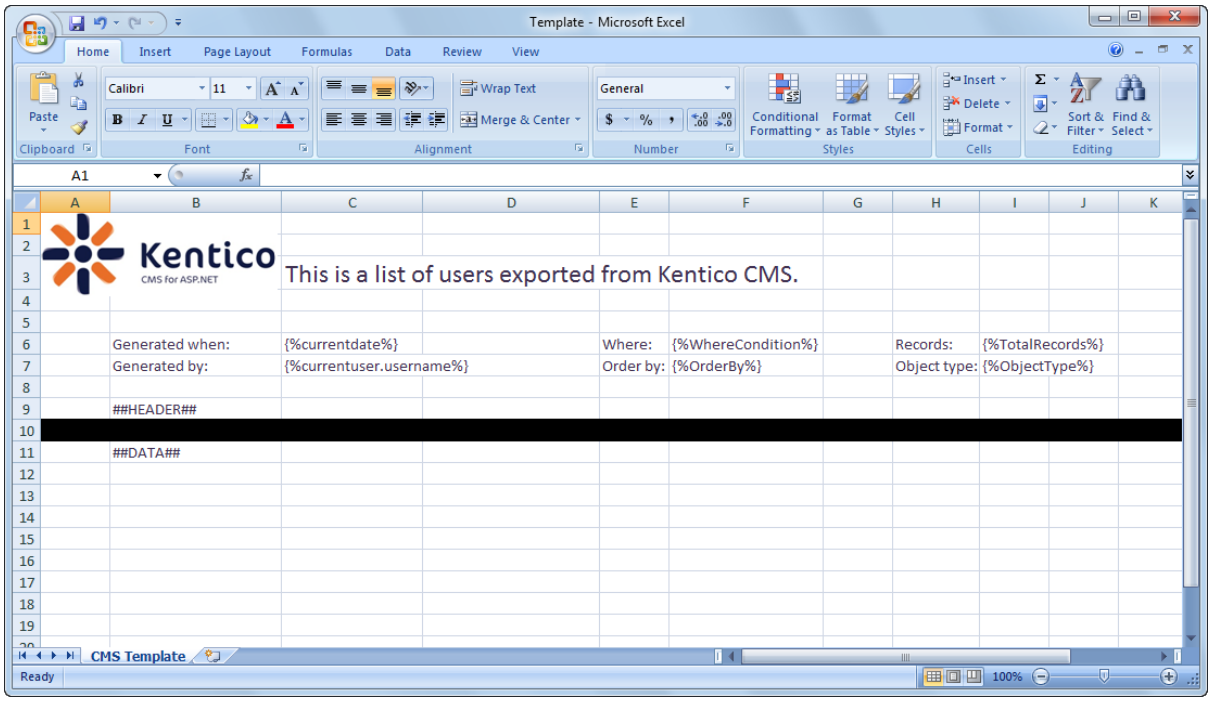
Template format

The template spreadsheet can contain any graphics, text or any other pre-filled data in it, while the following macros can be used in any cells. On export, the macros are replaced with the actual exported data:

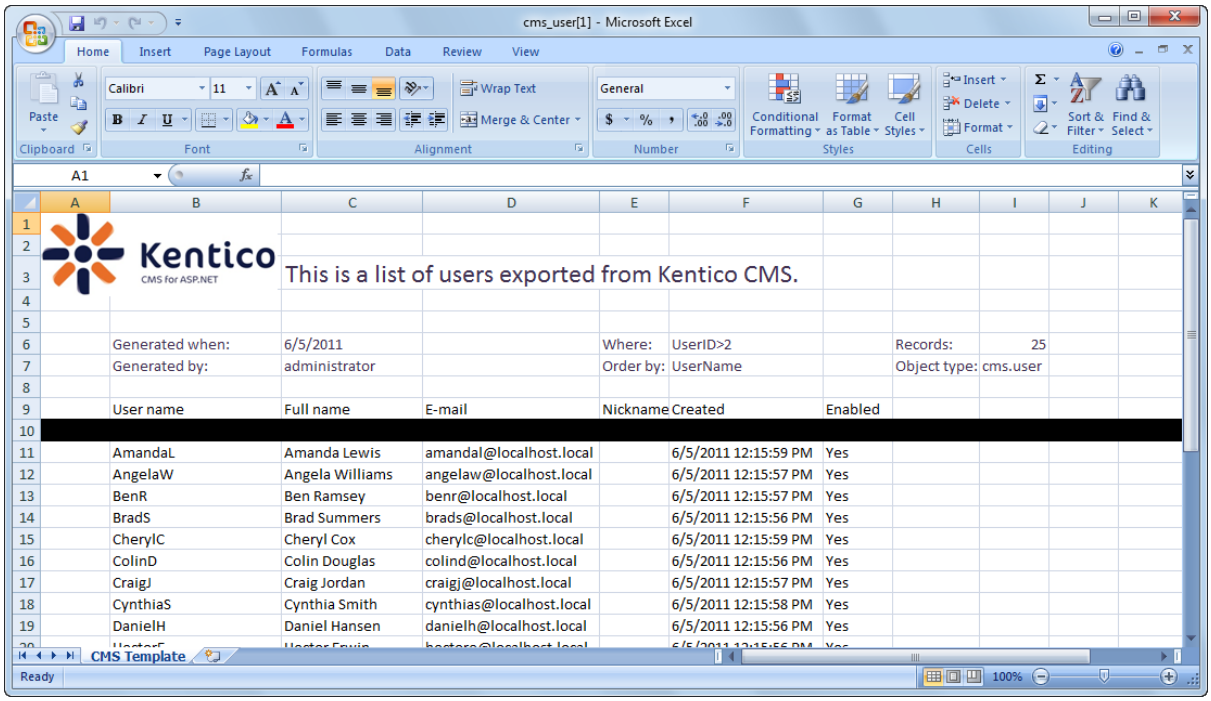
- **##HEADER##** - replaced with the header row. When advanced export is used, it is only replaced if the **Export header row** option is enabled.
- **##DATA##** - replaced with the actual exported data.
- **##TABLE##** - replaced with both the header row and the exported data.
- **{%WhereCondition%}** - replaced with the WHERE condition used for export (only relevant when exporting via the **Advanced export** dialog).
- **{%OrderBy%}** - replaced with the ORDER BY expression used to order exported items (either the expression configured in the **Advanced export** dialog or the default column according to which records are sorted when exporting using the **Export to Excel** action).
- **{%TotalRecords%}** - replaced with the total number of exported records.
- **{%ObjectType%}** - replaced with the type of exported object (e.g. *cms.userlist*).

You can also use all standard Context (data) macros the same way as you are used to within Kentico CMS user interface.

So for example, if you create a *Template.xlsx* file as in the screenshot below and upload it to ~*App_Data**CMSModules**DataExport**cms_user*\ ...



... the XLSX file with exported users will look as you can see below.

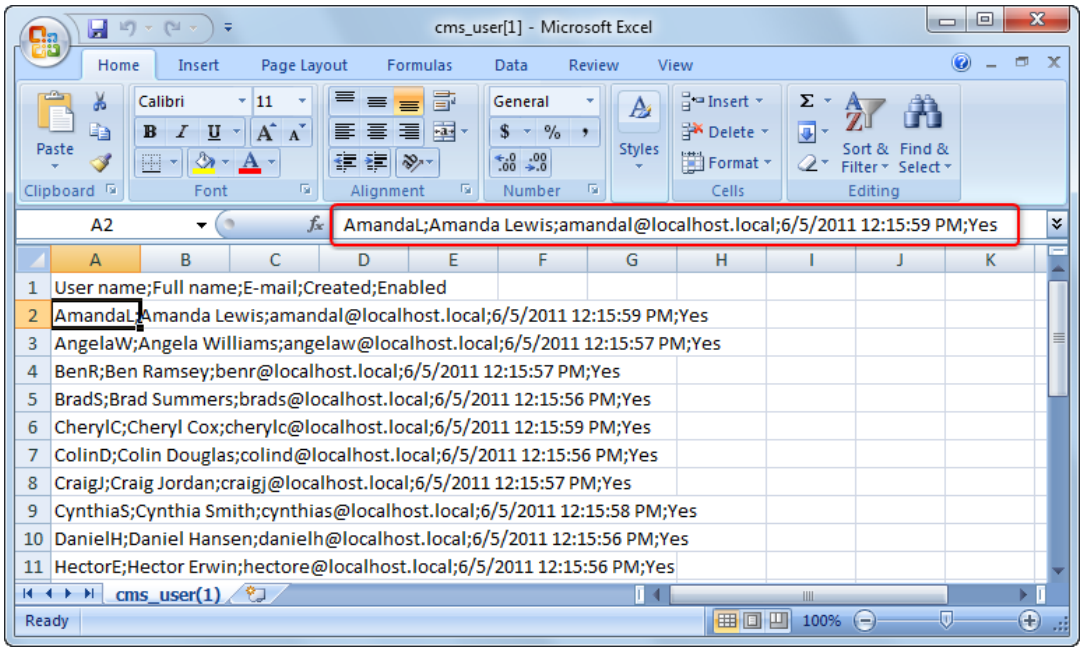


8.49.5 CSV delimiters

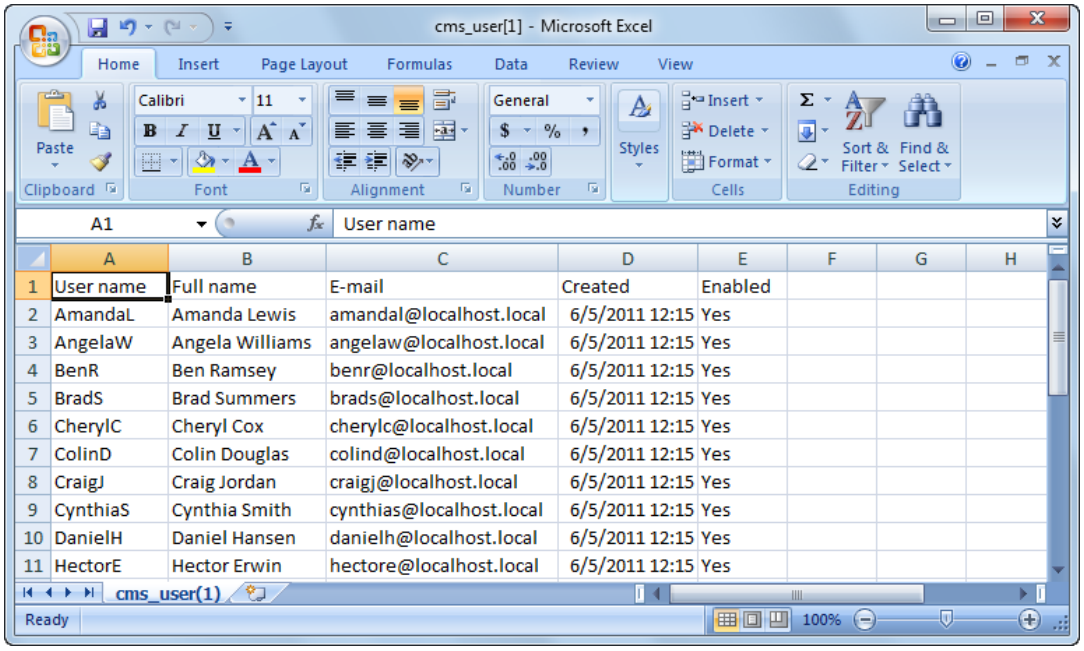
CSV is an abbreviation for Comma-separated values. It is a file format that stores tabular data in text form — each line represents one row of data, while particular values (columns) in each row are separated by a comma (,) or a semicolon (;). The comma is used as a column delimiter by default if you select the

Export to CSV action, while you can choose between the comma and the semicolon in the **Advanced export** dialog. The choice of the correct delimiter depends on your operating system's regional settings, as described below.

If you use the inappropriate delimiter, data from each row will be displayed in a single cell as the displaying software (e.g. Microsoft Excel) will not be able to identify the boundaries between individual values.



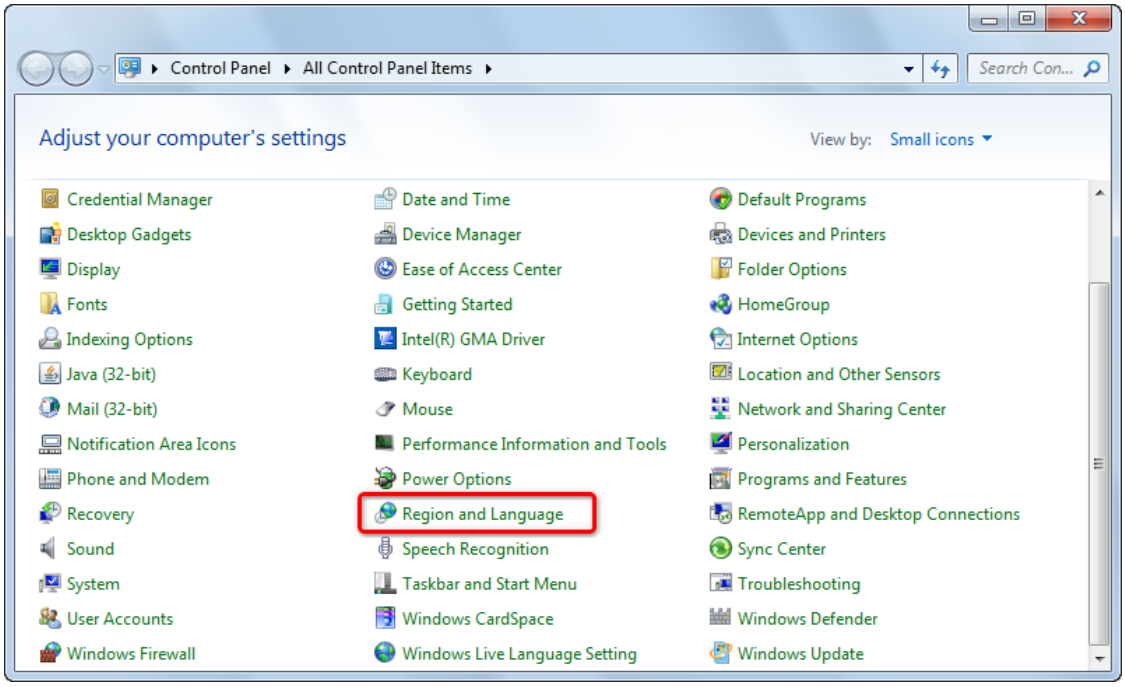
If the appropriate delimiter is used, data from each column will be displayed in individual cells as expected.



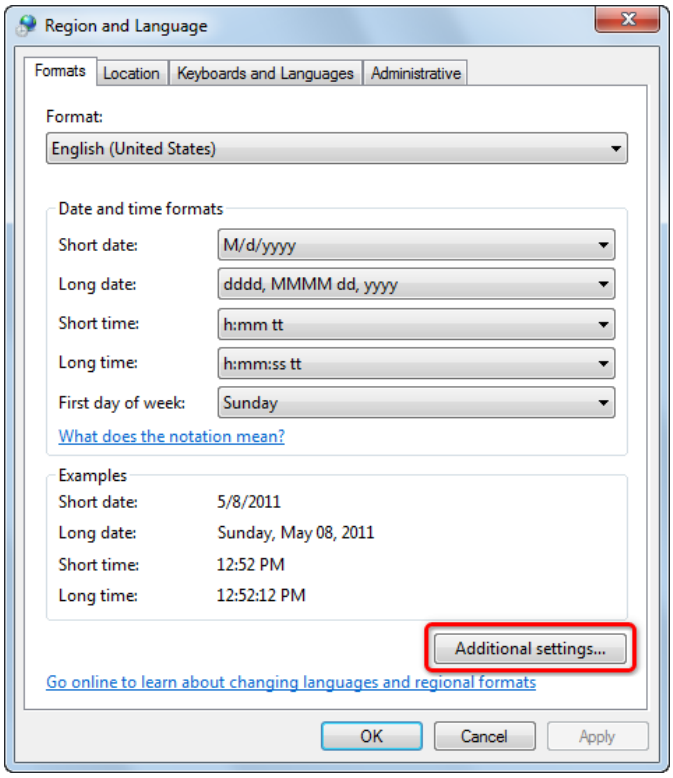
Delimiter settings on operating system level

To find out which delimiter you should use in your environment or to configure your system to use the other one than the one currently used, you need to:

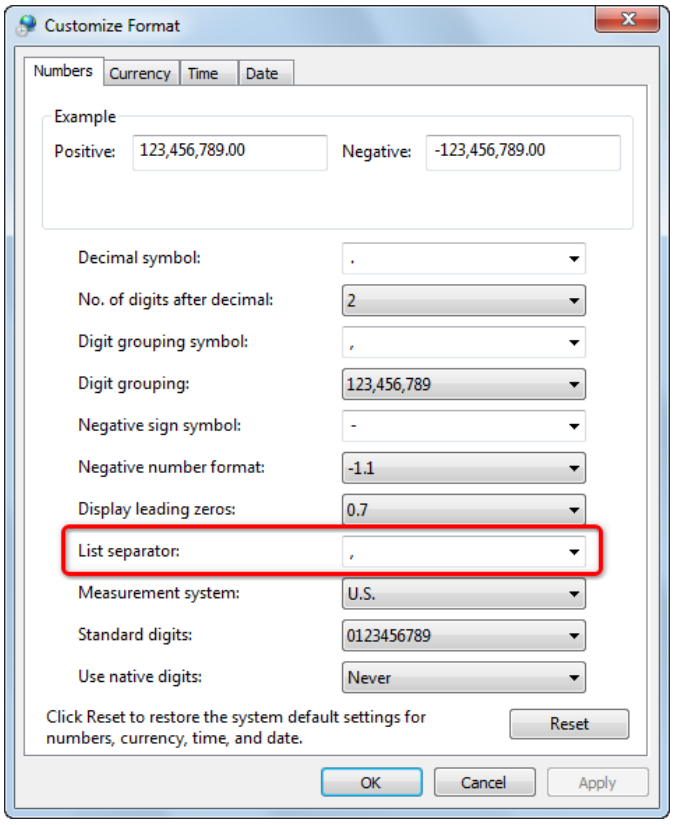
1. Go to **Start menu -> Control Panel** and open the **Region and Language** settings category.



2. In the **Region and Language** dialog, click **Additional settings...** on the **Formats** tab.



3. The **Customize Format** dialog will pop up. On its **Numbers** tab, you can choose the delimiter in the **List separator** field. The separator chosen here is the one that you should use when exporting listings data in order to get it displayed correctly.



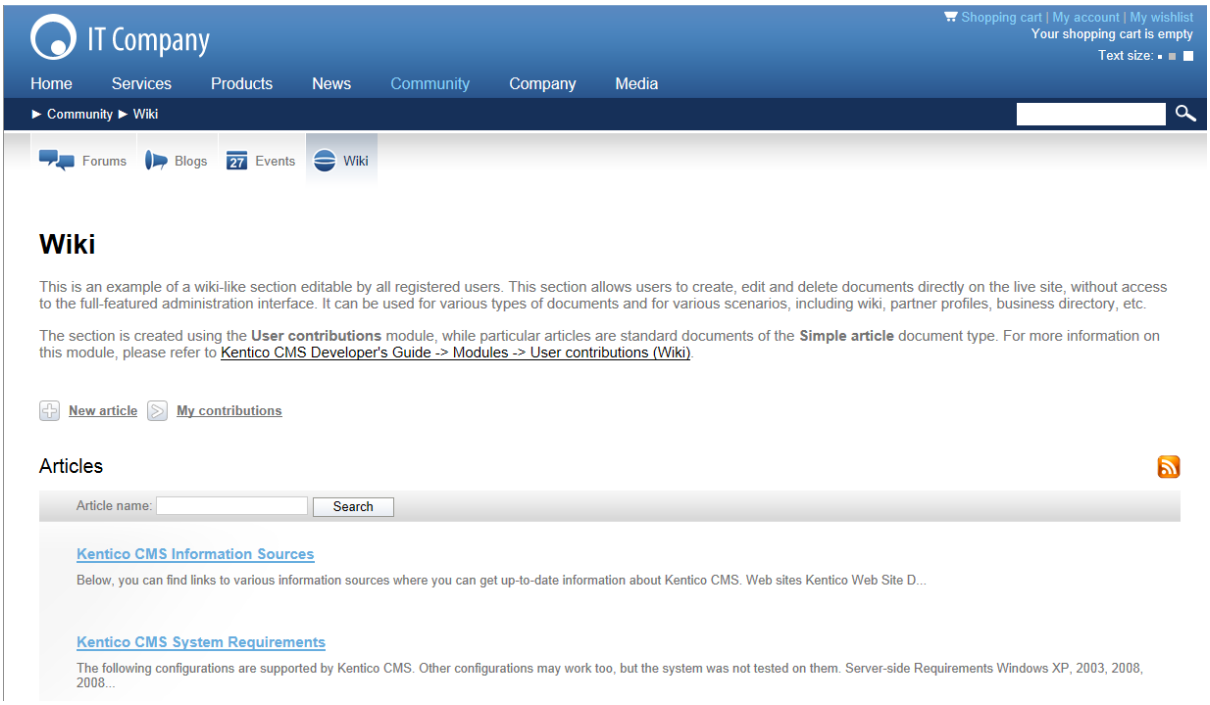
Important!

If you change the value in the **List separator** field, make sure to confirm both the **Customize Format** and the **Region and Language** dialogs by clicking **OK** or **Apply**. Otherwise, the changes will not take effect and the original settings will be preserved.

8.50 User contributions (Wiki)

8.50.1 Overview

The User contributions module allows you to create a content editing interface for site members. It means that chosen website visitors can create, edit and delete content, even if they are not editors and cannot access CMS Desk or On-site editing mode.



There are several scenarios where you can use this module, for example:

1. Community news

You can create a list of news and allow community members to add news without going to CMS Desk.

2. Partner directory

Your business partners can manage their profile on your website and the list of their reference project.

3. Business directory

You can create a business directory for some town or industry and let the business owners to manage their own profile.

Implementation

You can use the following web parts to implement user contributions:

- **Contribution list** - this web part allows you to display a list of contributions (documents) and the **New document** link.
- **Edit contribution** - this web part allows you to edit an existing document.

You can find the description of both web parts in the Kentico CMS Web Parts reference.

- If you would like to see an example of how to create a new section for publishing community news, please refer to the Example: Publishing community news topic.
- If you would like to see an example of how to create a list of partners who will be able to edit their profile on a special page after they have signed in, please refer to the Example: Editing partner profile topic.

- If you would like to learn about the security possibilities of the User contributions (Wiki) module, please refer to the Security topic.
- A brief reference on the relationship between the User contributions (Wiki) module and API can be found in the User contributions and API topic.

You will need to have the E-commerce Site sample website installed to follow Kentico CMS E-commerce Guide.

Several more practical examples of the use of the User contributions (Wiki) module are available in Kentico CMS Community Site Guide; please note that these examples do not concern the whole functionality of the module but focus on its use in a broader context of the Community Site sample website:

- See Community Site Guide -> Part 2 -> Creating the Blogs section -> Creating the Blogs page: An example of the use of the User contributions (Wiki) module web parts in the context of creating the Blogs section.
- See Creating the Groups section -> Preparing the Group pages section -> Creating the Pages page in the same section of Kentico CMS Community Site Guide: An example of the use of the User contributions (Wiki) module web parts in the context of creating the Groups section.

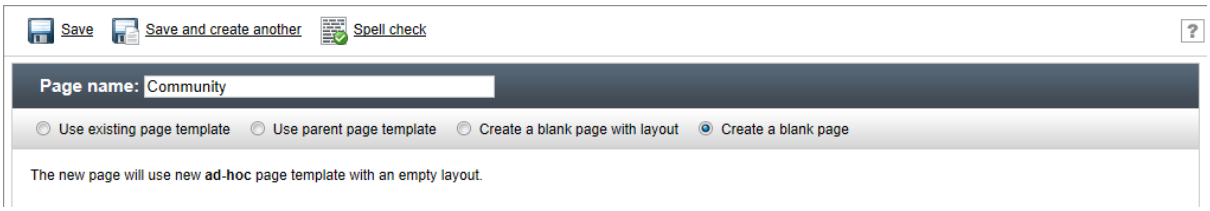
8.50.2 Example: Publishing community news

In this example, you will create a new section for publishing community news. Each registered site member can create new news items and edit/delete their previously posted news. The example assumes that you are using the **Corporate Site** sample website. In this topic you will learn how to:

- Create the community news section
- Add the New document button
- Add the editing support
- Create a testing user
- Create your first news item
- Approve the news

Creating the community news section

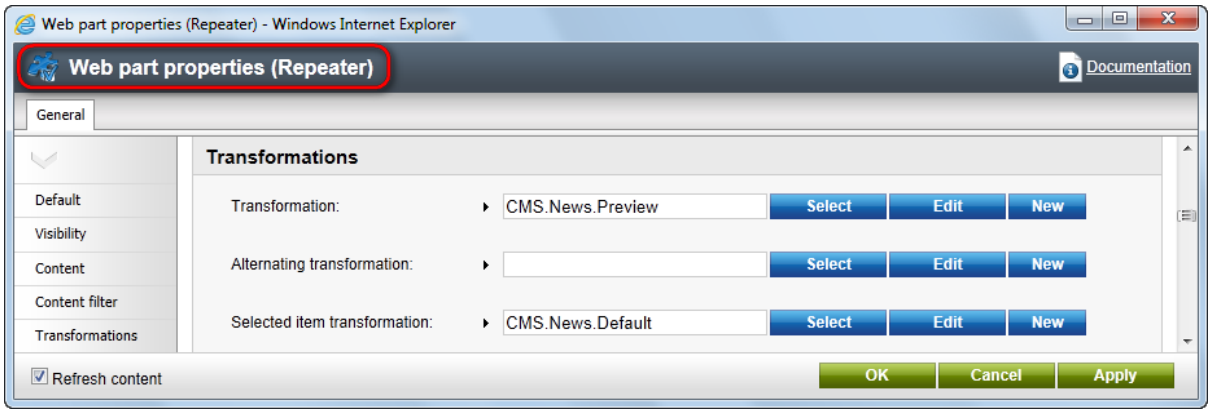
1. Sign in to **CMS Desk** as the administrator, go to the **Content** section and click **Examples** in the content tree. Click **New** and choose to create a new **Page (menu item)**. Enter the name **Community** and choose the **Create a blank page** option.



2. Click **Save**. Switch to the **Design** tab of the newly created page and add the **Listings and Viewers/Repeater** web part. Set the following properties:

- **Web part control ID:** repeaterNews
- **Document types:** cms.news
- **Transformation:** CMS.News.preview

- **Selected item transformation:** CMS.News.default



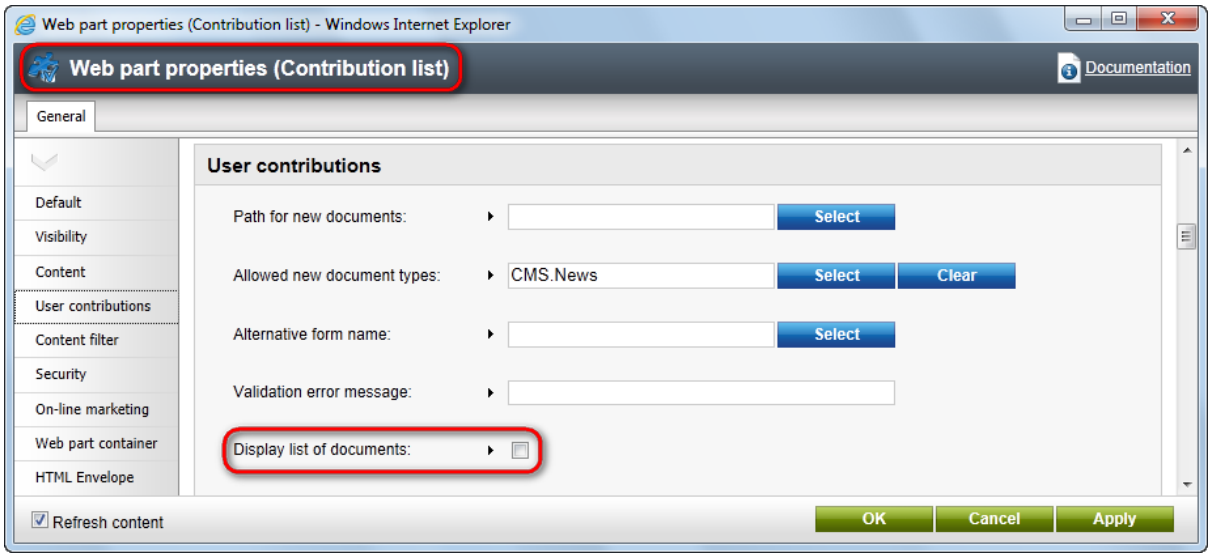
3. Click **OK**.

Adding the New document button

1. Add a new web part **User contributions/Contribution list**. It will display the **New document** link. Set the following values:

- **Show for document types:** cms.menuitem
  - this ensures that the web part is displayed only when the list of news is displayed, not on the news detail page.
- **Allowed new document types:** CMS.News
  - this means that the users will be allowed to create only news items under this section.
- **Display list of documents:** no (unchecked)
  *- this ensures that the web part displays only the **New document** link, without displaying the list of documents.*
- **Allow editing by users:** Authenticated
  - this means that only authenticated users will be able to edit/delete the document. Authenticated users are those who are signed-in to the website (not to the administration interface).

Leave the other values as they are by default.



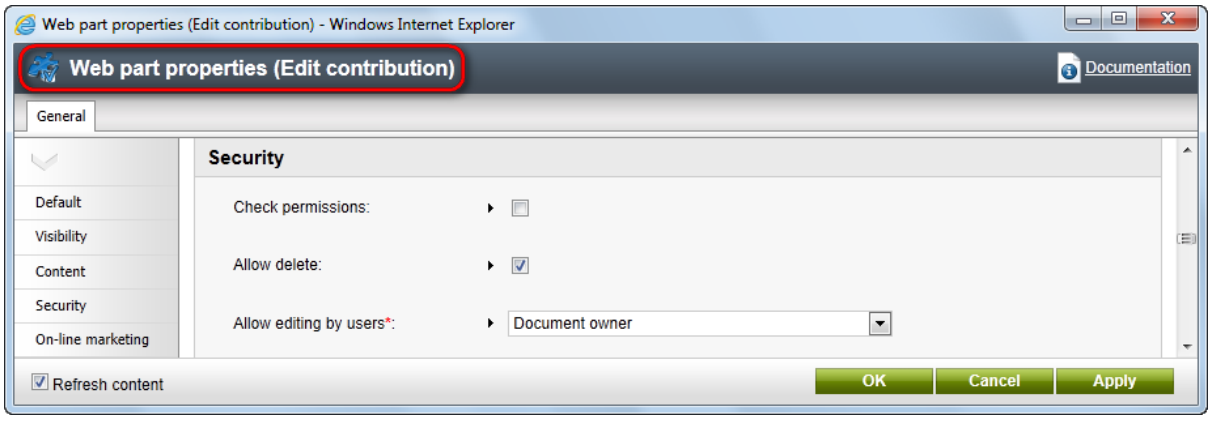
2. Click **OK**.

Adding the editing support

1. Add a new web part **User contributions/Edit contribution**. It will display the **Edit/Delete** icons when some news item is chosen.

- **Show for document types:** cms.news
  - this ensures that the web part is displayed only on the news detail page.
- **Allow editing by users:** Document owner

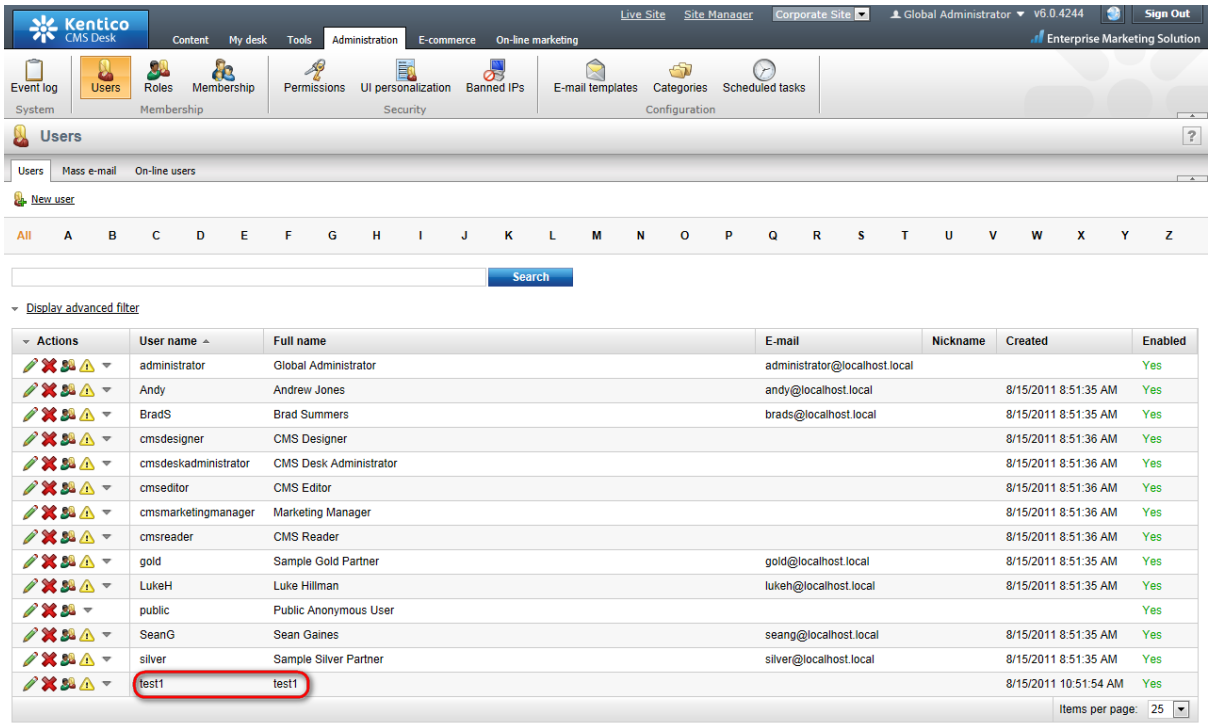
Please note that the **Edit contribution** web part is missing the **Allow insert/edit** properties.



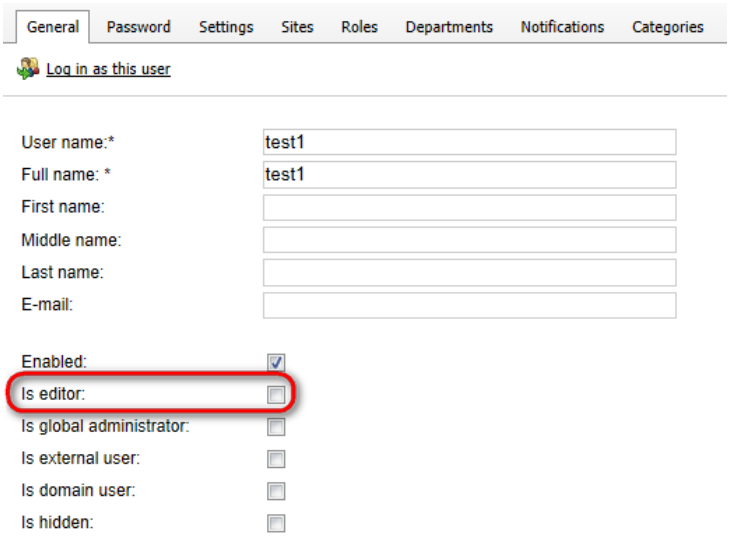
2. Click **OK**.

Creating a testing user

1. Go to **CMS Desk -> Administration -> Users** and click the **New user** link to create a new user with user name **test1**. Please note that to create a user, at least the values of the following two compulsory fields: **User name** and **Full name** must be entered.



2. For the purpose of this example, choose no (uncheck the box) in the **Is editor** field - **test1** is a common site member without access to Kentico CMS Desk.

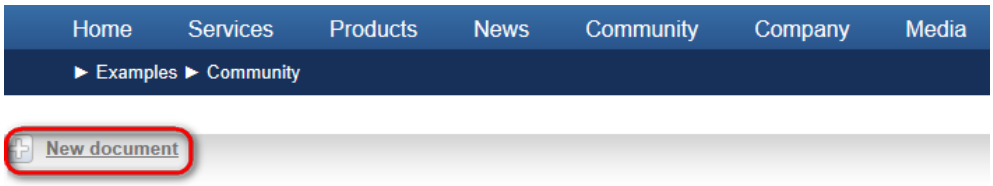


Click **OK** to save the changes.

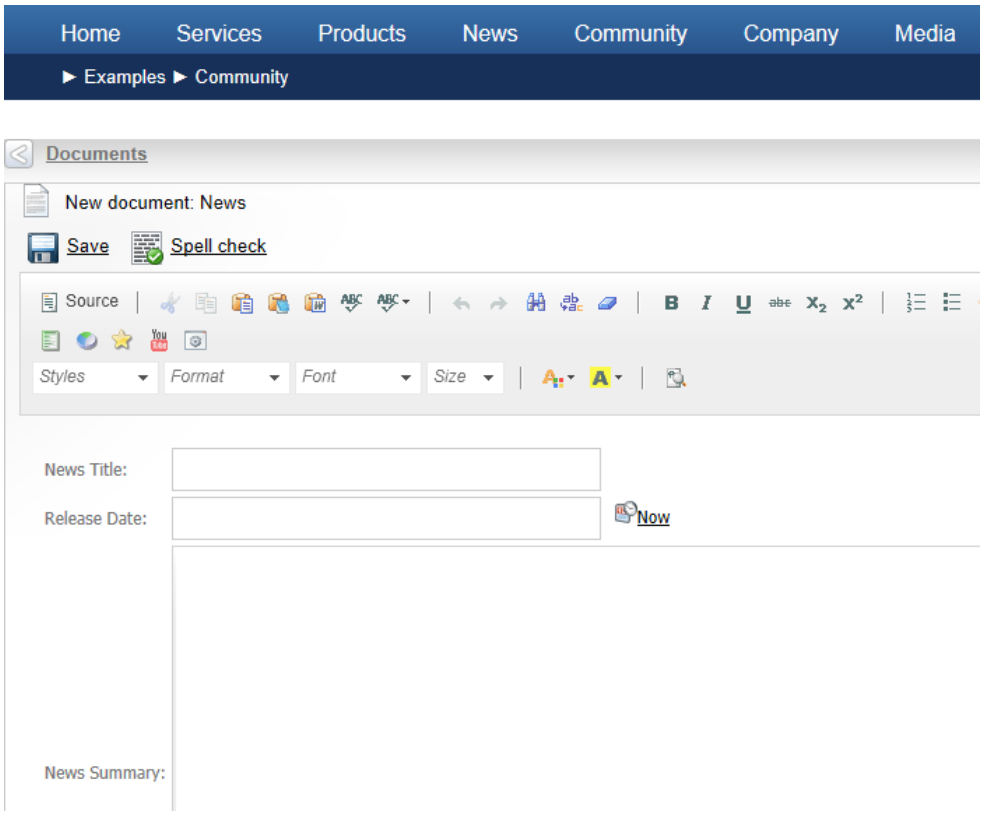
Creating your first news item

1. Sign out and go to the **Examples -> Community** section. You will see a blank page since the user contributions are now enabled only for site members. In order to sign in, click the **My account** link at the top and sign in as user **test1**. Then go back to the **Examples -> Community** section. You will see the

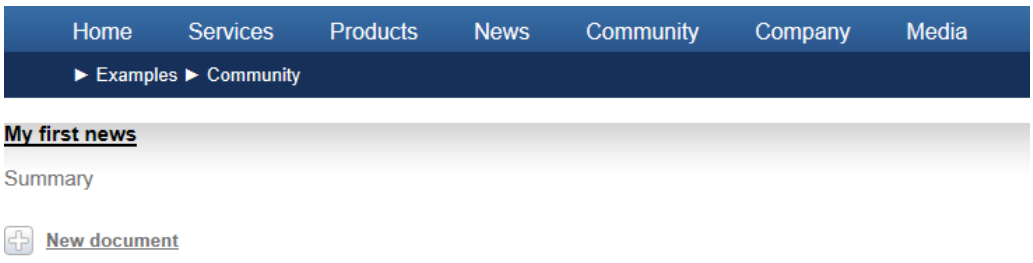
New document link:



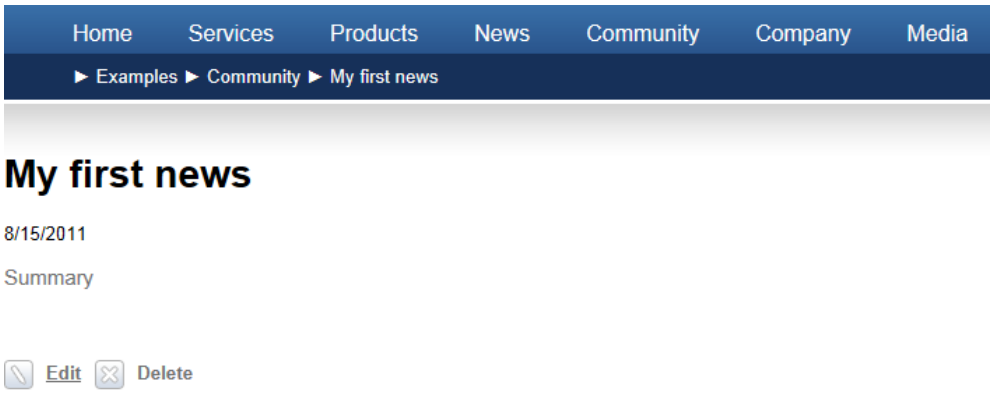
2. Click this link and you will be displayed with the news editing form:



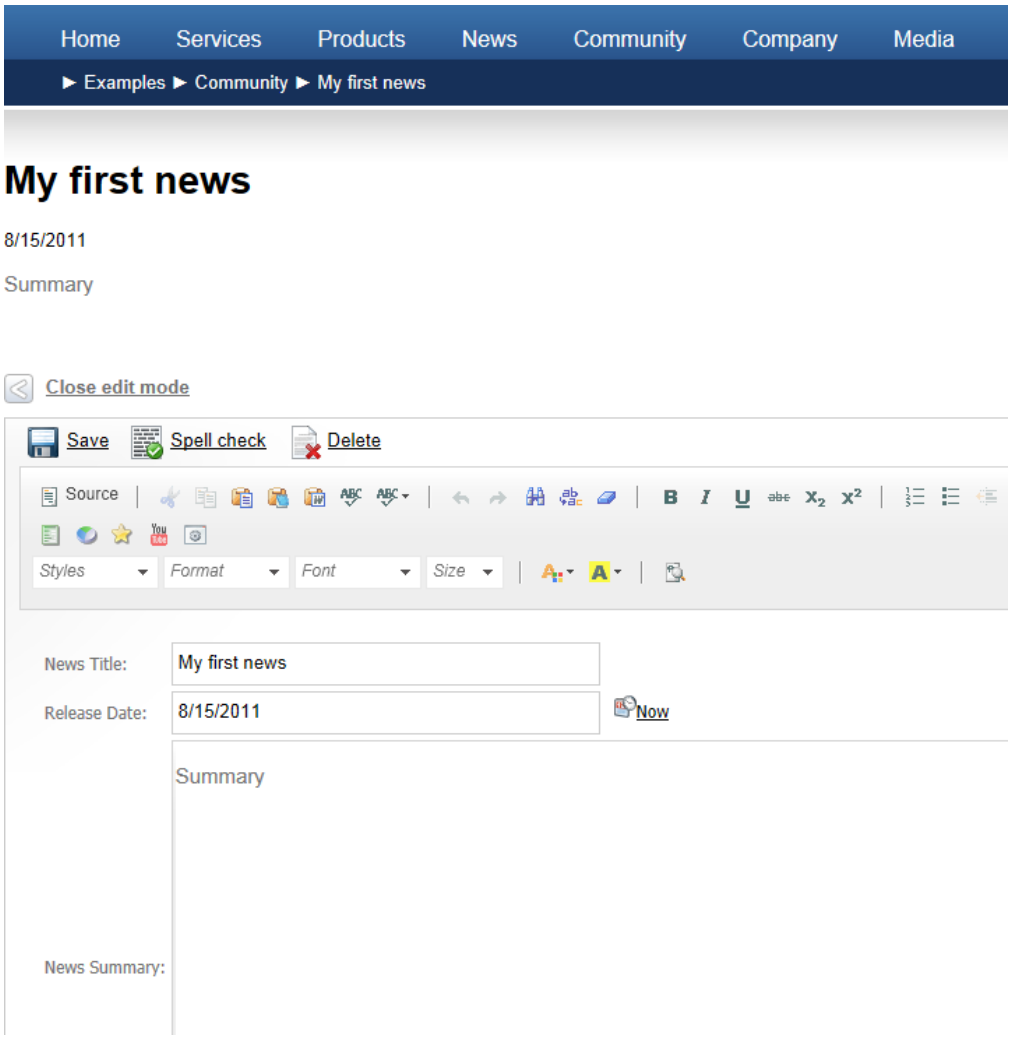
3. Enter some content and click **Save**. Now select **Examples -> Community** from the main menu at the top. You will see the **Community** page with your first news item:



4. Click the news item link to see the details page. The **Edit** and **Delete** links are be displayed below the news item:



5. Choose to **Edit** the news item and you will be displayed with the editing form where you can edit its content:



6. Make some modifications and click **Save**. Click the **Close edit mode** link to see the updated

page.

Approval process

If you need to enforce some approval process for published news, you can simply set up workflow for the given site sections and all news will need to be approved by a site manager. You can find more details on workflow configuration in the Workflow overview chapter.

In this example, you have learned how to create a site section where community members can create and edit content without having access to Kentico CMS Desk.

8.50.3 Example: Editing partner profile

In this example, you will create a list of partners who will be able to edit their profile on the **My profile** page after they have signed in. This example assumes that you are using the Corporate Site sample website.

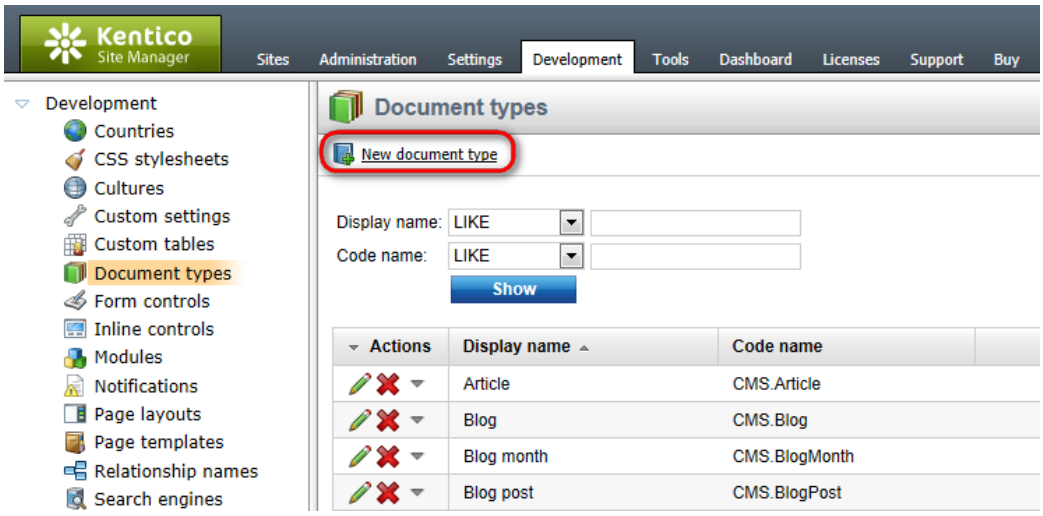
- Create the partner document type
- Create the partner list
- Create the partner user account
- Create the partner profile document
- Create the partner profile editing page
- Test the profile editing page

Creating the partner document type

Please note

A detailed description of how to create a new document type can be found in the Document types and transformations -> Defining a new document type topic in the Development section of the Developer's Guide.

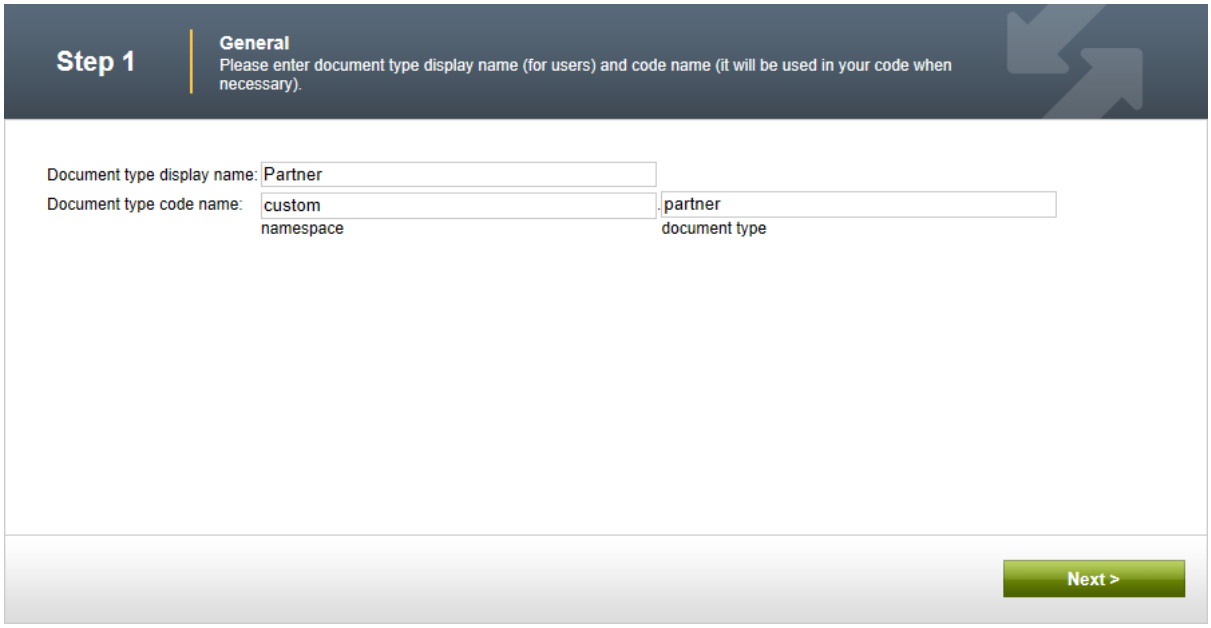
Before you create the page, it is necessary that you create a new document type **Partner**. Sign in to **Site Manager** and go to **Development -> Document types** where you need to click **New document type**.



Step 1: General

Enter the following values and click the **Next** button.

- **Document type display name:** Partner (this name will be displayed to the users)
- **Document type code name:** custom.Partner

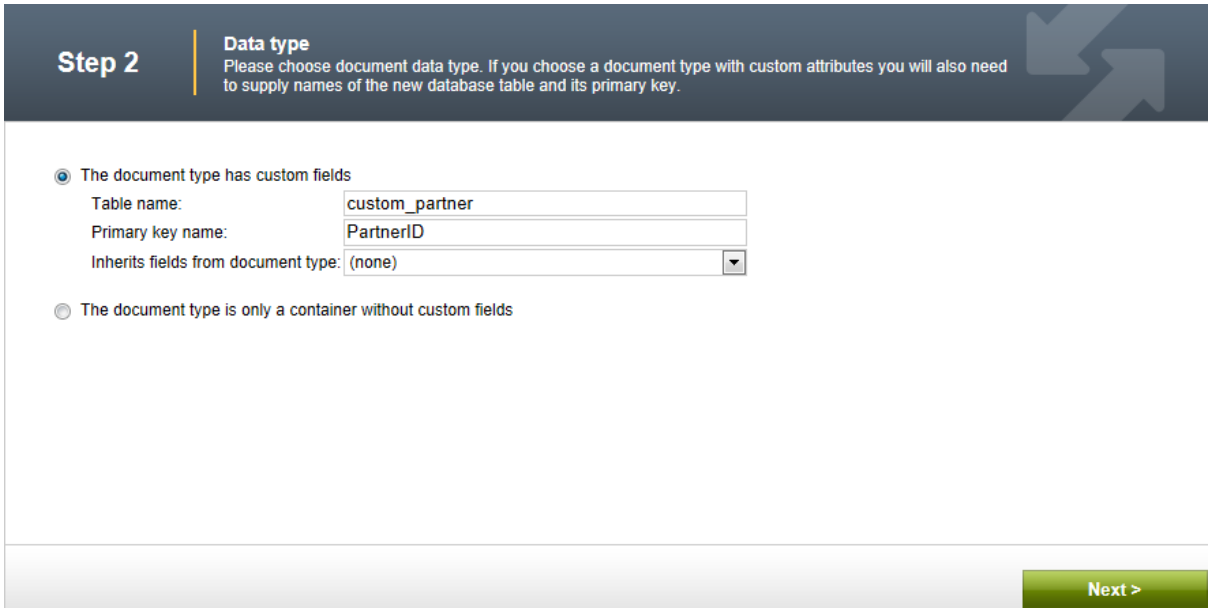


Please note that *custom* in the **Document type code name** field is your namespace to distinguish your document types from system types that use the *cms* namespace. You will use this value in web part properties later.

Step 2: Data type

You need to choose the name of the database table that will be used for storing partner details. You also need to enter the name of the primary key in this table. Choose the option **The document type has custom fields** and enter the following values:

- **Table name:** custom_Partner
- **Primary key name:** PartnerID



Click **Next**.

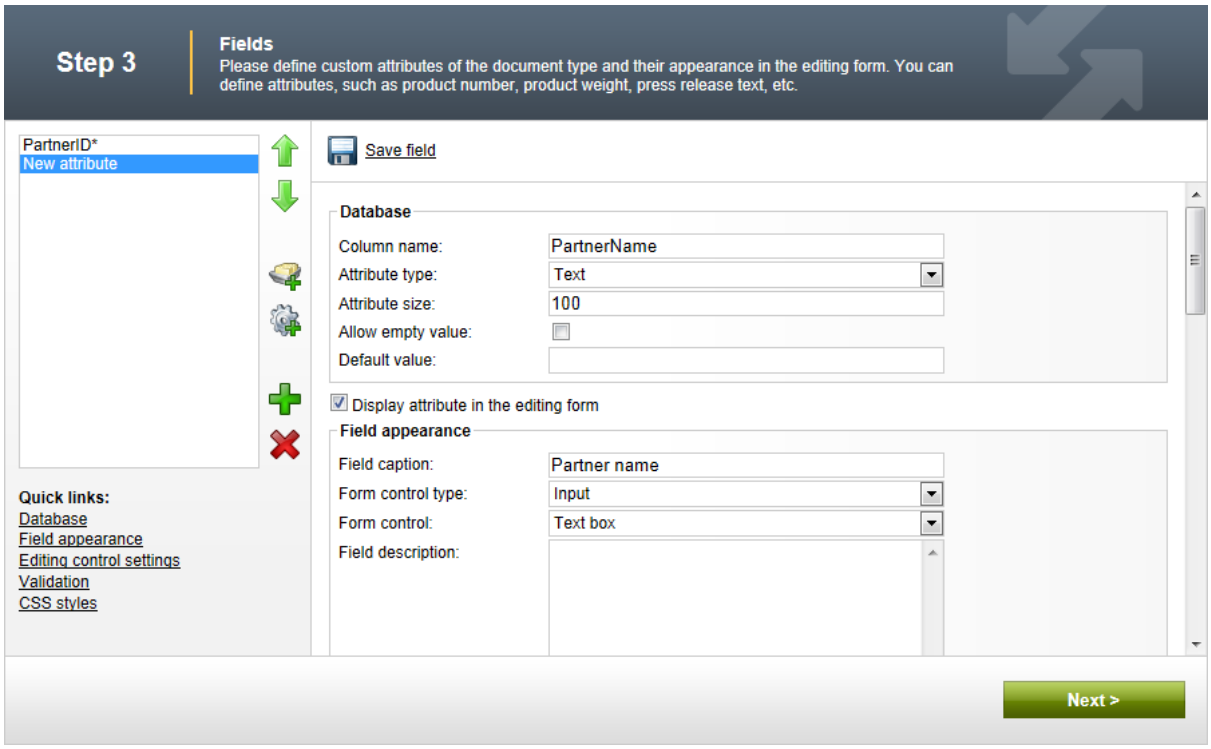
Step 3: Fields

Now you need to define the columns of the table or fields. Click **New attribute** () to create a new field, enter the following values and click **OK**.

- **Attribute name:** PartnerName
- **Attribute type:** Text
- **Attribute size:** 100
- **Field caption:** Partner name
- **Field type:** TextBox

- **Attribute name:** PartnerProfile
- **Attribute type:** Long Text
- **Field caption:** Partner profile
- **Field type:** HTML area (Formatted Text)

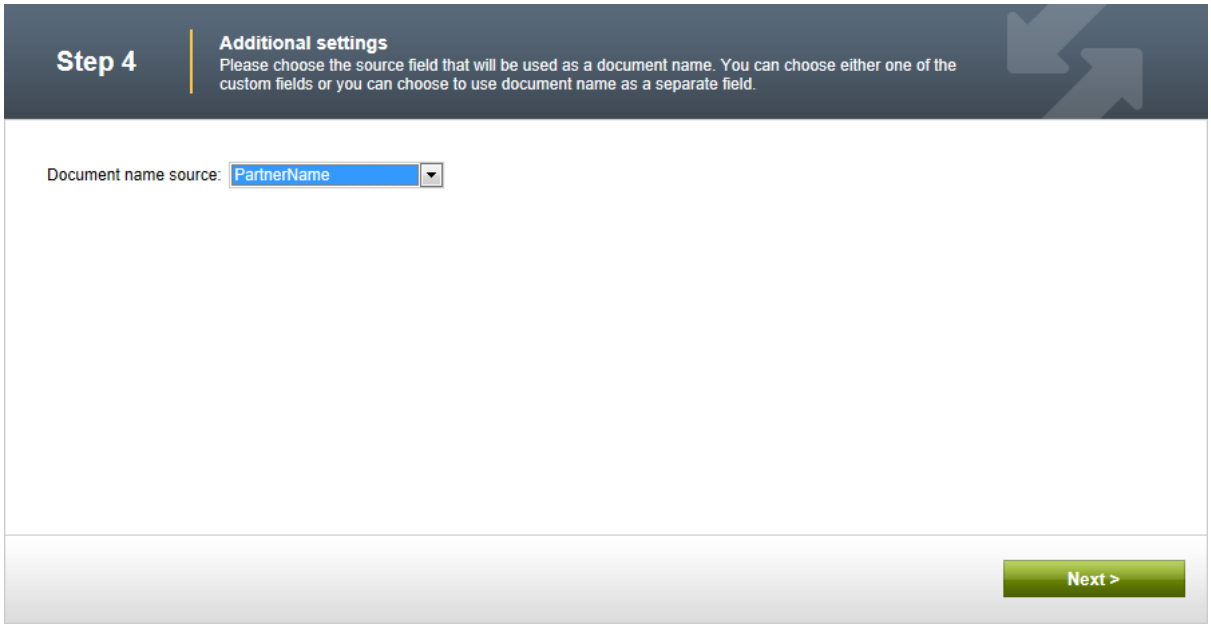
When you have defined both the fields, click **Next** again.



Step 4: Additional Settings

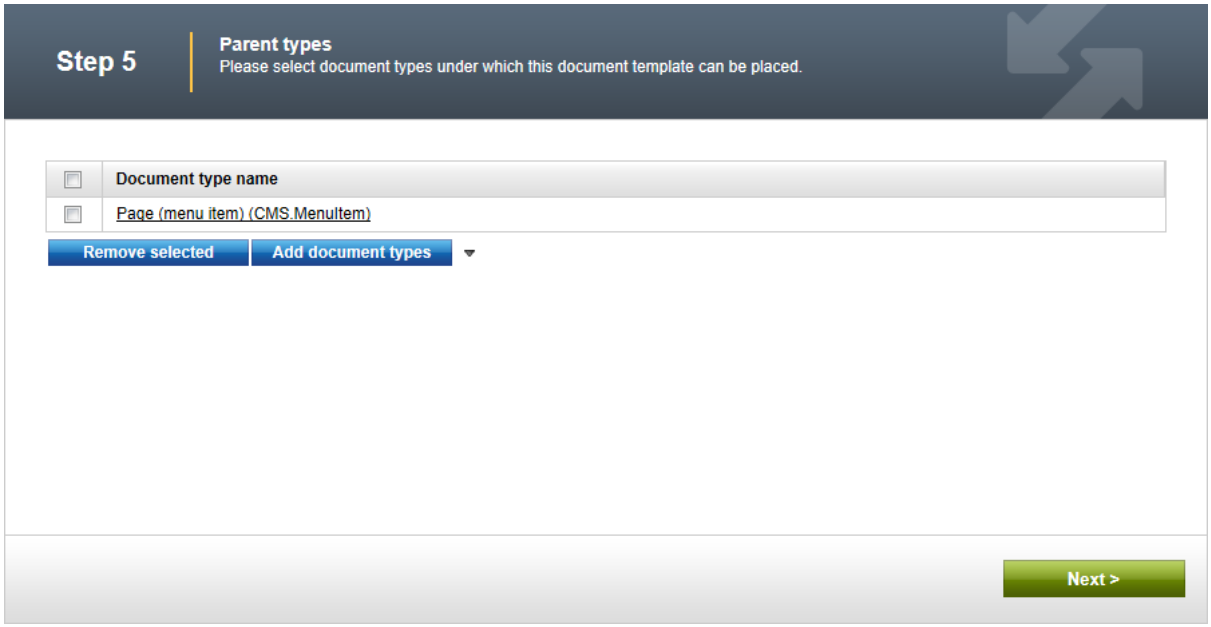
Now you need to choose the field that will be used as document name. Choose the *PartnerName* field option from the **Document name source** drop-down list and click **Next**.

- **Document name source:** PartnerName



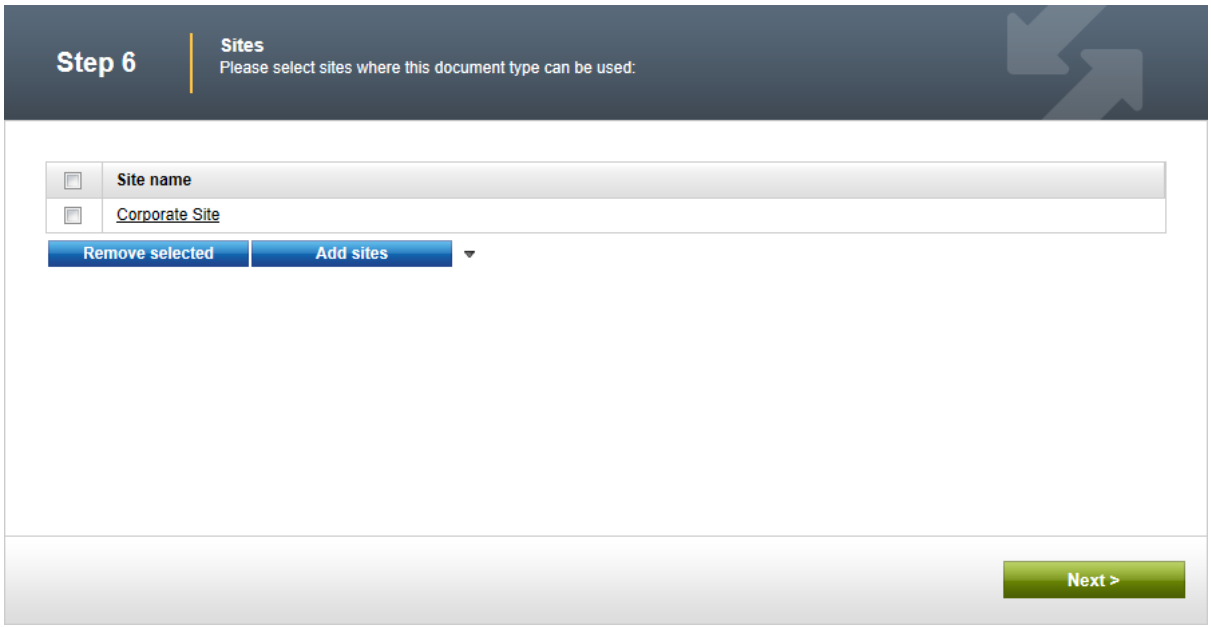
Step 5: Parent types

In this step, you need to select the document types under which the partner documents will be displayed. Check only the **Page (menu item)** value, which means the users will be able to create computer documents only under some page, not under article or news document in the content tree. Click the **Next** button.



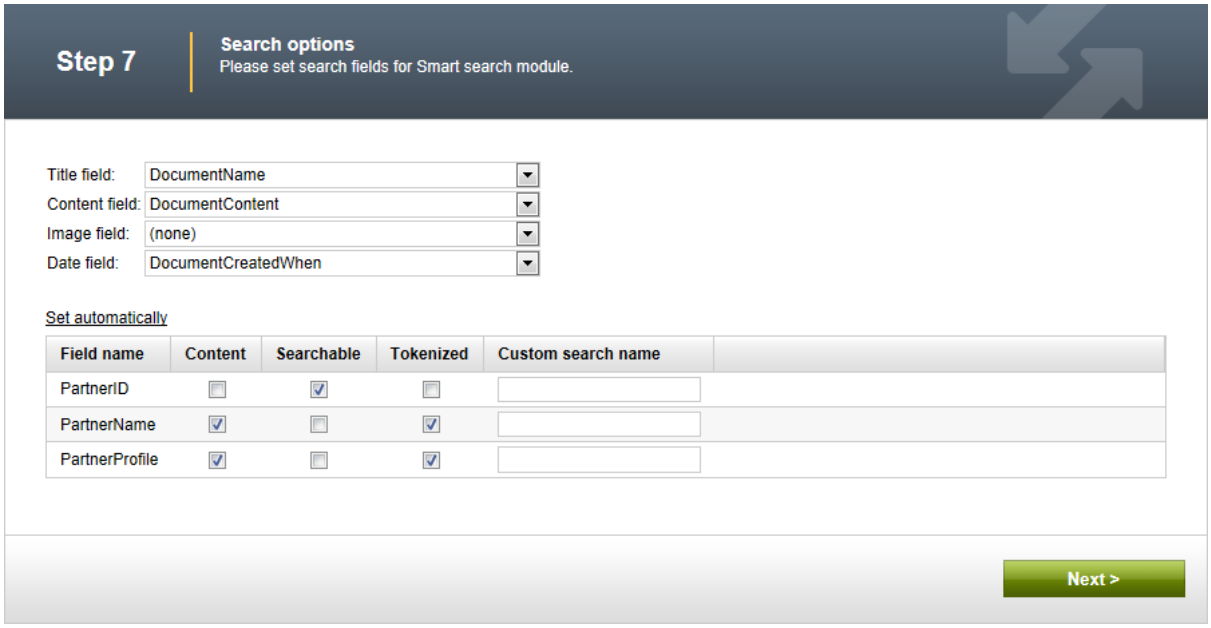
Step 6: Sites

Here you need to choose which websites will use this document type. Choose your current website and click **Next**.



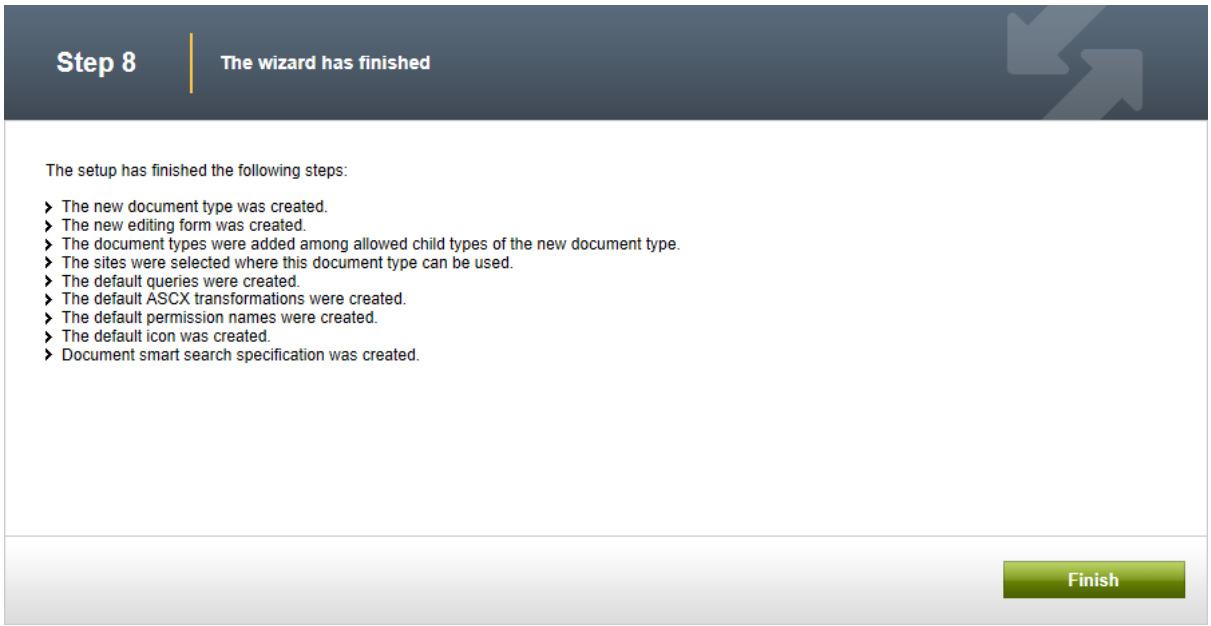
Step 7: Search options

In this step, you are asked to specify how documents of this type will be indexed and displayed in the search results. For more information on these settings, please refer to the Settings for particular object types topic. Make your choice and click **Next**.



Step 8: The wizard has finished

Step 8 is the last step in the process of creating the partner document type - the wizard has finished the configuration of the this document type.



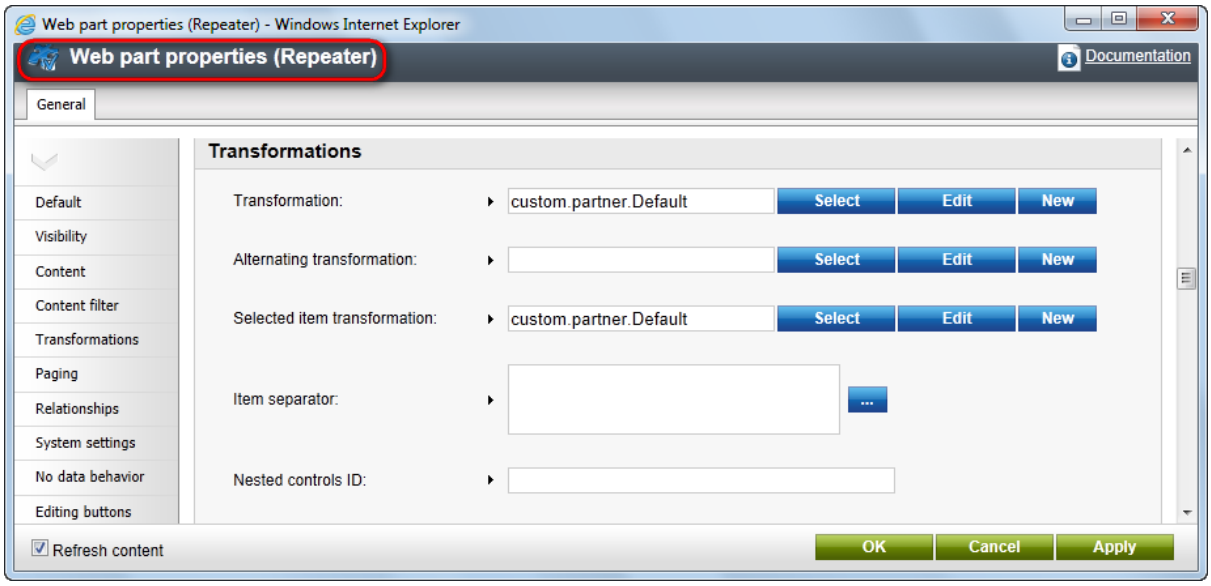
Creating the partner list

1. Go to **CMS Desk -> Content**, choose **Examples** from the content tree, click **New** and choose to

create a new **Page (menu item)** document. Call the page **Partner directory**, choose the **Create a blank page** option and click **Save**.

2. Switch to the **Design tab** of the newly created page and add the **Listings and viewers/Repeater** web part. Set the following properties:

- **ID:** repeaterPartners
- **Document types:** custom.partner
- **Transformation:** custom.Partner.Default
- **Selected item transformation:** custom.Partner.Default

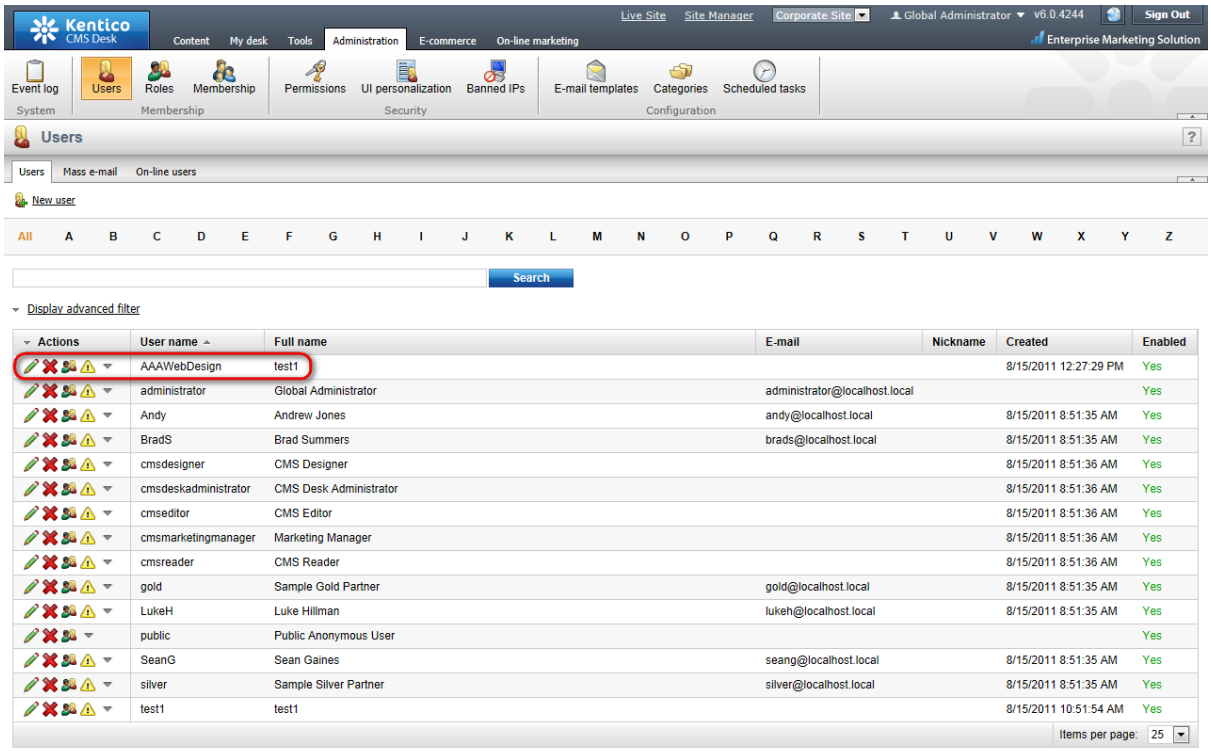


Please note that the default transformation generated by the system will be used. You can later modify the transformation so that it meets your design and layout requirements.

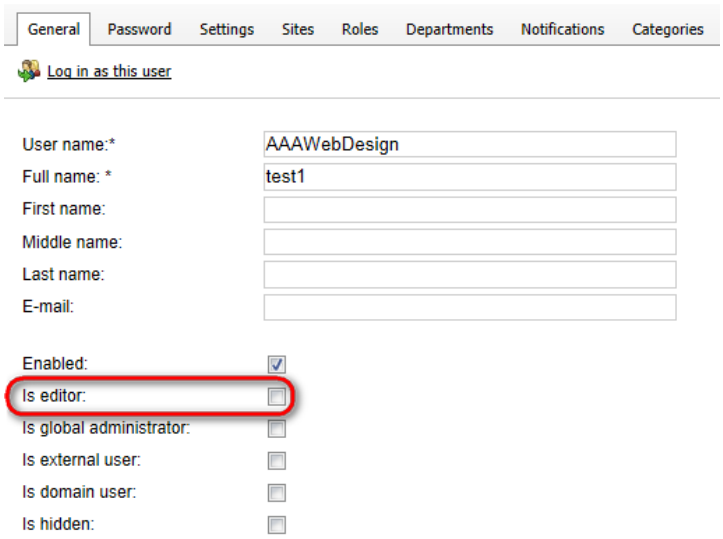
3. Click **OK**. Switch to the **Page** view and you will see an empty page now.

Creating the partner user account

1. Go to **CMS Desk -> Administration -> Users** and by clicking the **New user** link create a new user with user name **AAAWebDesign**. Assign the user to the **CMS Basic users** role.



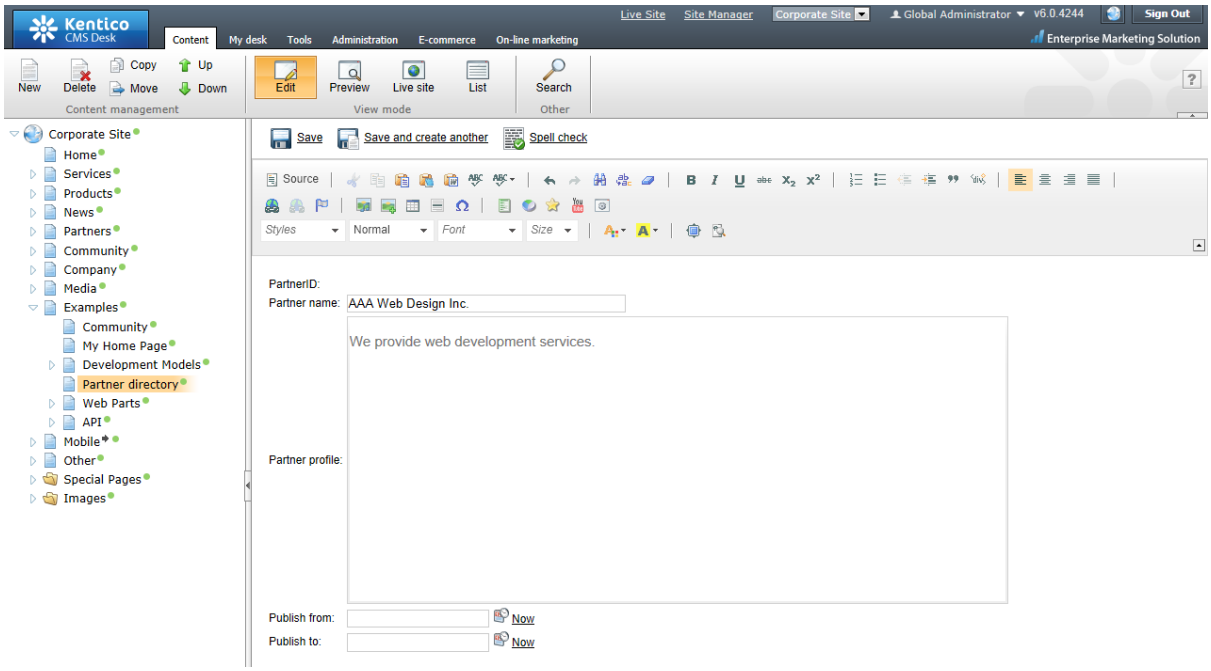
2. For the purpose of this example, choose no (uncheck the box) in the **Is editor** field - **AAA Web Design** is a common site member without access to Kentico CMS Desk. Click **OK** to save the changes.



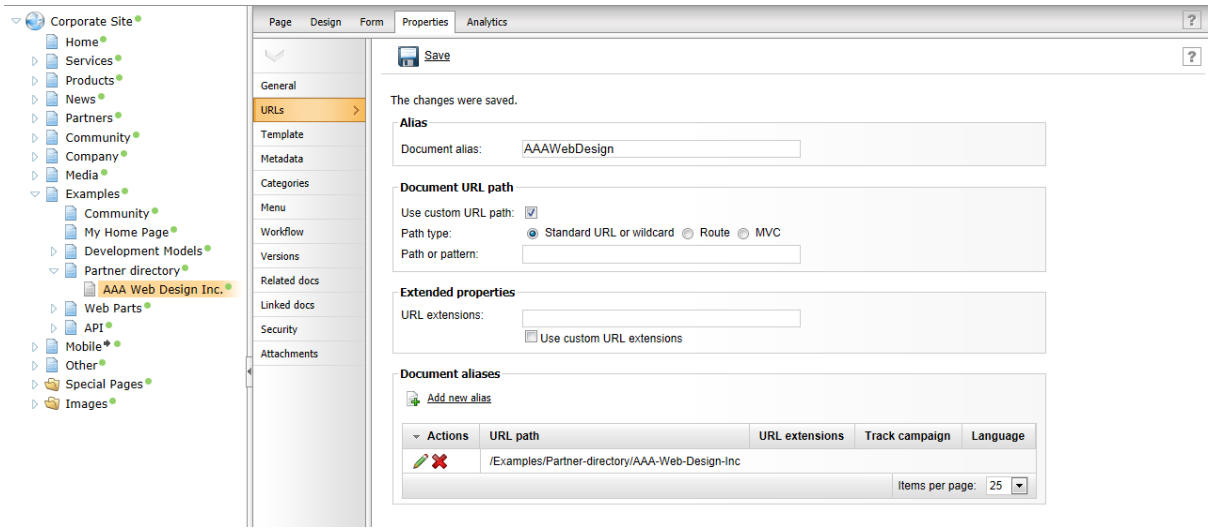
Creating the partner profile document

1. Go to **CMS Desk -> Content**, from the content tree, choose **/Examples/Partner directory** and click **New**. Choose to create a new **Partner** and enter the following values:

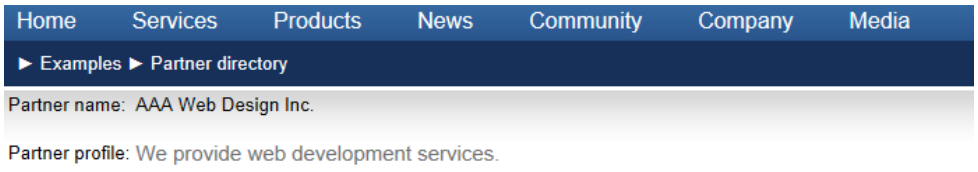
- **Partner name:** AAA Web Design Inc.
- **Partner profile:** We provide web development services.



2. Click **Save**. Go to the **Properties -> URLs** tab of the newly created document, set the **Document alias** to **AAAWebDesign** and click **Save**. You need to use the same alias as the user name since you will be using them to match the users to their user profiles. The alias path of the document will always be **`/Examples/Partner-directory/<user name>`**.



3. When you display the page in the **Live site** mode now, you will see a page like this:



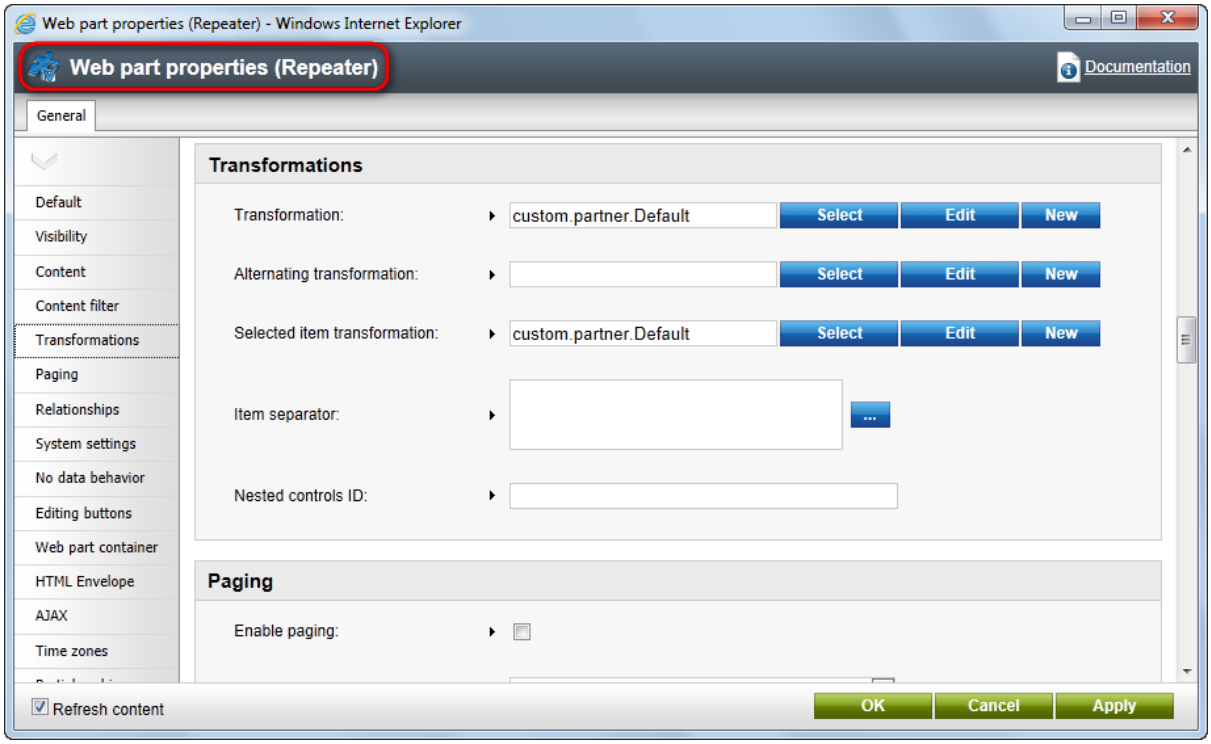
Creating the partner profile editing page

Now you will create a page that will be available only to partners and that will allow them to edit their partner profile.

1. Go to **CMS Desk -> Content**, from the content tree, choose **/Examples** and click **New**. Choose to create a new **Page (menu item)**. Call the page *My profile*, choose the **Create a blank page** option and click **Save**.

2. Switch to the **Design tab** of the newly created page and add the **Listings and viewers/Repeater** web part. It will display the current user's partner profile. Enter the following values and click **OK**.

- **ID:** repeaterPartners
- **Path:** /Examples/Partner-directory/{%UserName%}
  - this ensures that the page displays the profile that matches the current user. The {%UserName%} macro is resolved as the user name of the current site visitor.
- **Document types:** custom.partner
- **Transformation:** custom.Partner.Default
- **Selected item transformation:** custom.Partner.Default
- **Hide if no record found:** no (unchecked)
- **No record found text:** No partner profile was found for your user account.
- **Content before:** <h1>Your Partner Profile</h1>

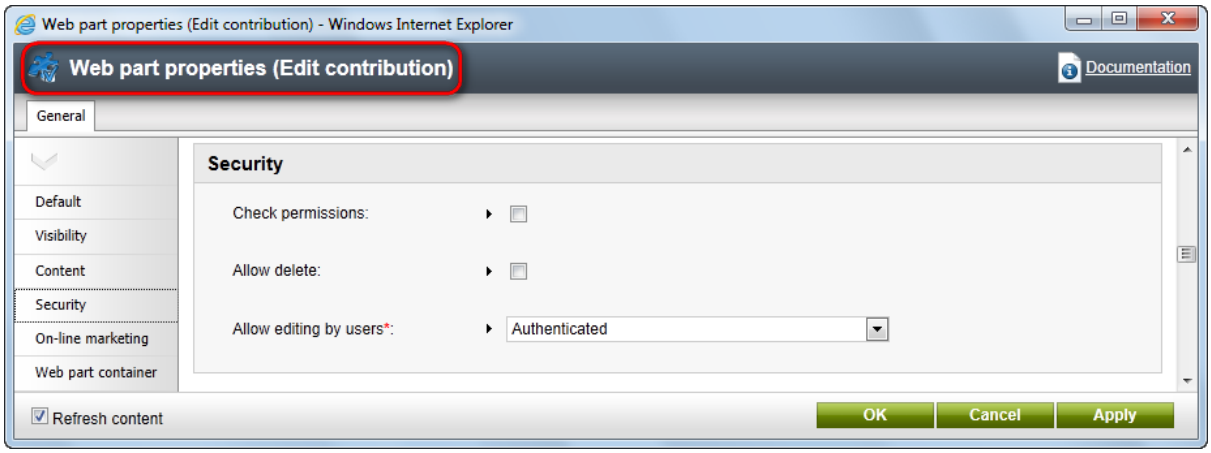


3. Add the **User contributions/Edit contribution** web part, enter the following values and click **OK**.

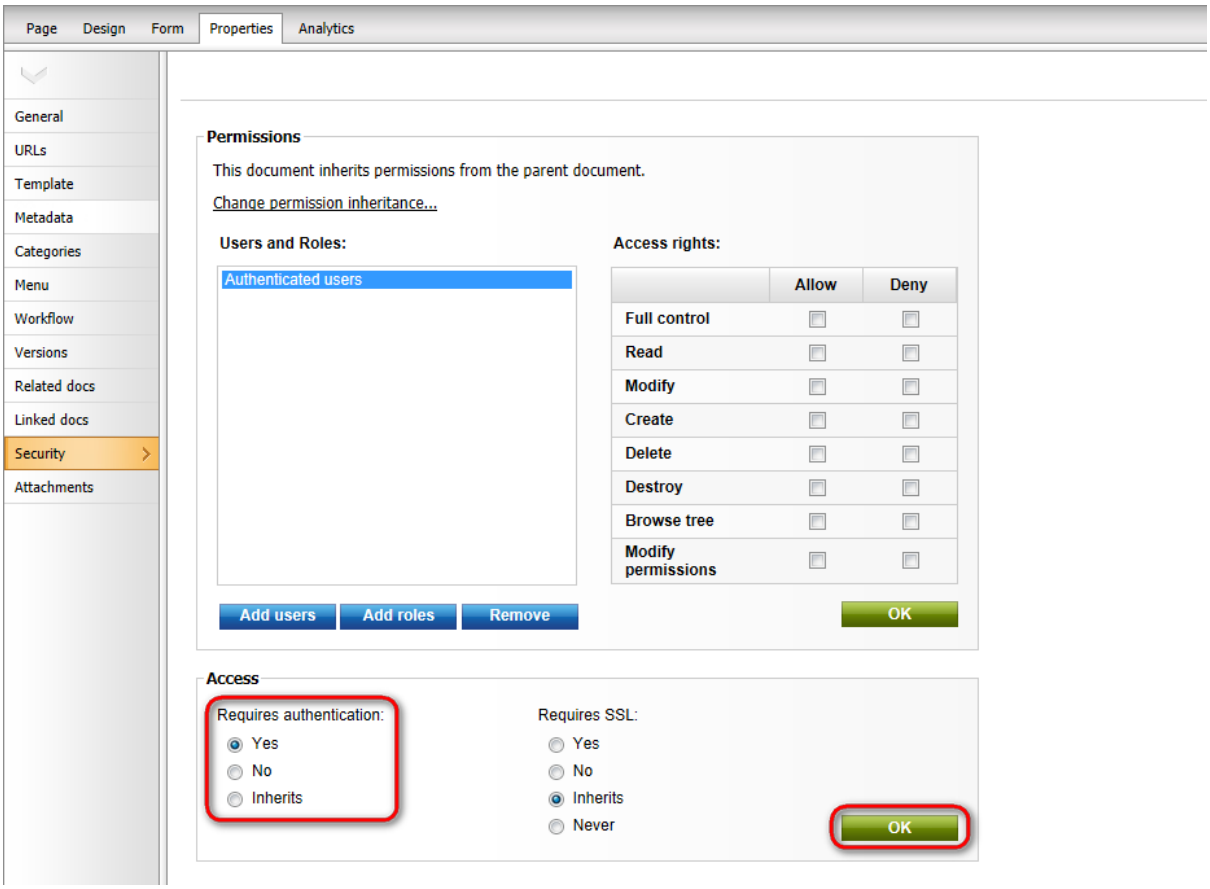
- **Path:** /Examples/Partner-directory/{%UserName%}
  - this ensures that the form edits the profile that matches the current user. The {%UserName%} macro is resolved as the user name of the current site visitor.

- **Allow delete:** no (unchecked)
  - this option hides the Delete button since you do not want the partners to delete their profile accidentally.
- **Allow editing by users:** Authenticated
  - you want to allow editing only to authenticated users; at the same time, you do not require the user to be a document owner.

Please note that the **Edit contribution** web part is missing the **Allow insert/edit** properties.

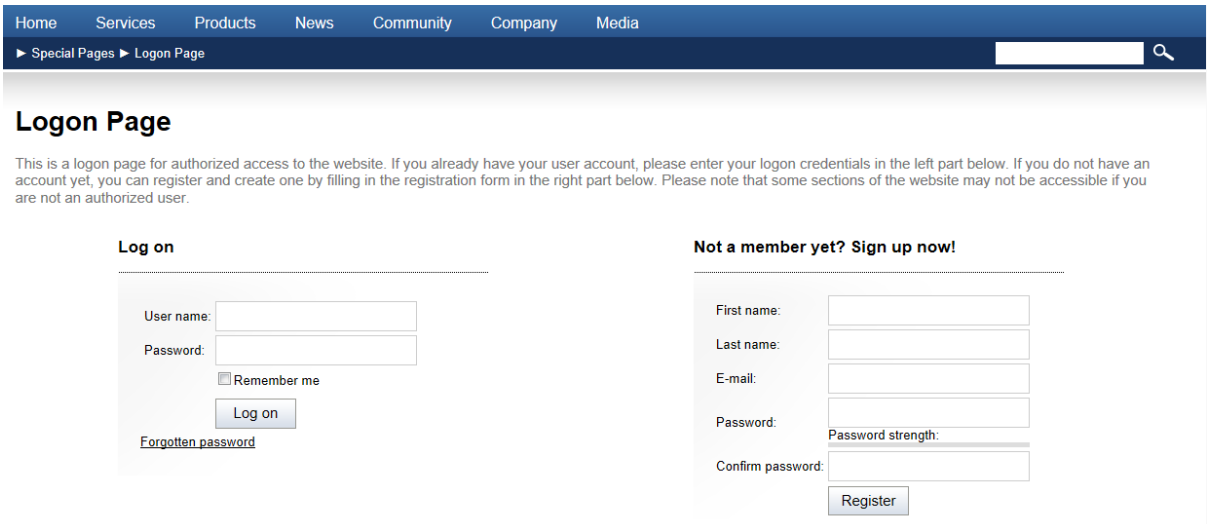


4. As the last step, go to the **Properties -> Security** dialog, set the **Requires authentication** value to **Yes** and click **OK** - the reason is that you want to allow access to this page only to authenticated partners so that you know their user name and can display their profile.

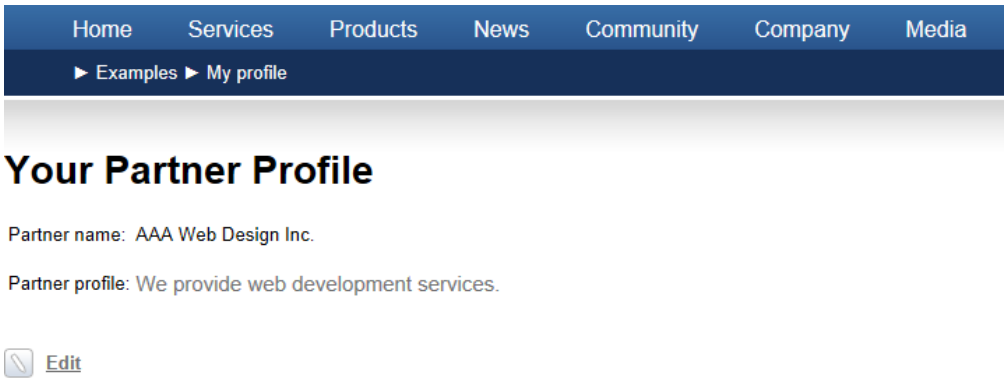


Testing the profile editing page

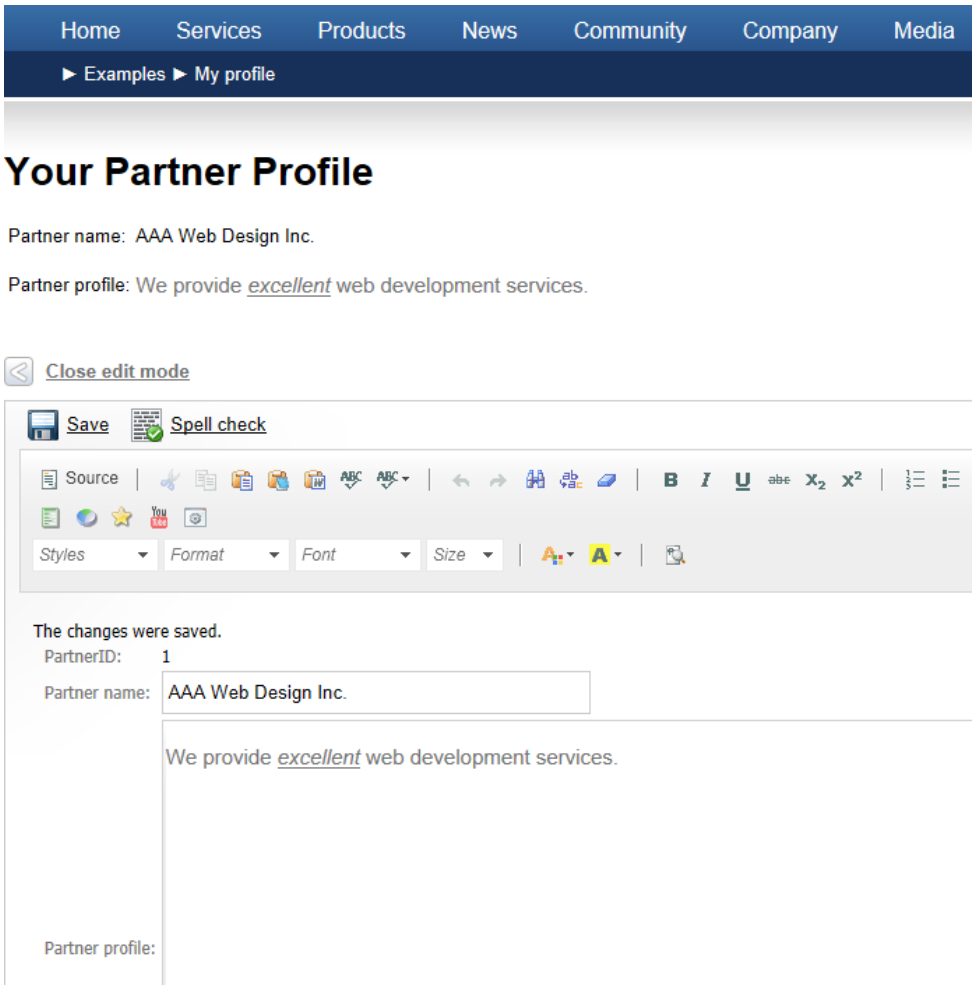
1. Sign out and go to **Examples -> My profile**. You will be displayed with logon form.



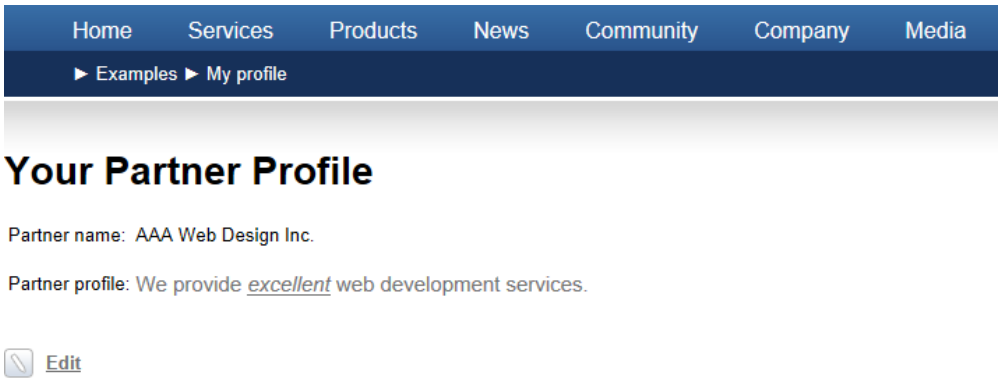
2. Sign in as user **AAAWebDesign** and you will see your partner profile with the **Edit** link.



3. Click **Edit** to open the edit dialog window, modify some values and click **Save**:



4. Go to **Examples -> Partner directory** and you will see the updated profile.



In this topic, you have learnt how to allow users (partners) to edit a single document that matches their user name. It was a little different in comparison to the previous example (Example: Publishing community news), since in this case, you created the document first and only then you mapped it to the user by matching the document alias path and user name. In this example, the partner was not allowed to create new documents and you did not use the document owner security option.

Hint 1: Simplifying the process of creating a new partner

In such cases, it is useful to create a custom module (see Custom modules) that will contain a custom form for creating new partners. Your code will ensure creating the user, the default partner profile and setting the document alias path and user name to the same value. You will need to use Kentico CMS to create the user account (see the Managing users API examples) and the partner profile document (see the Creating documents API examples).

Hint 2: Assigning multiple users to the same partner profile

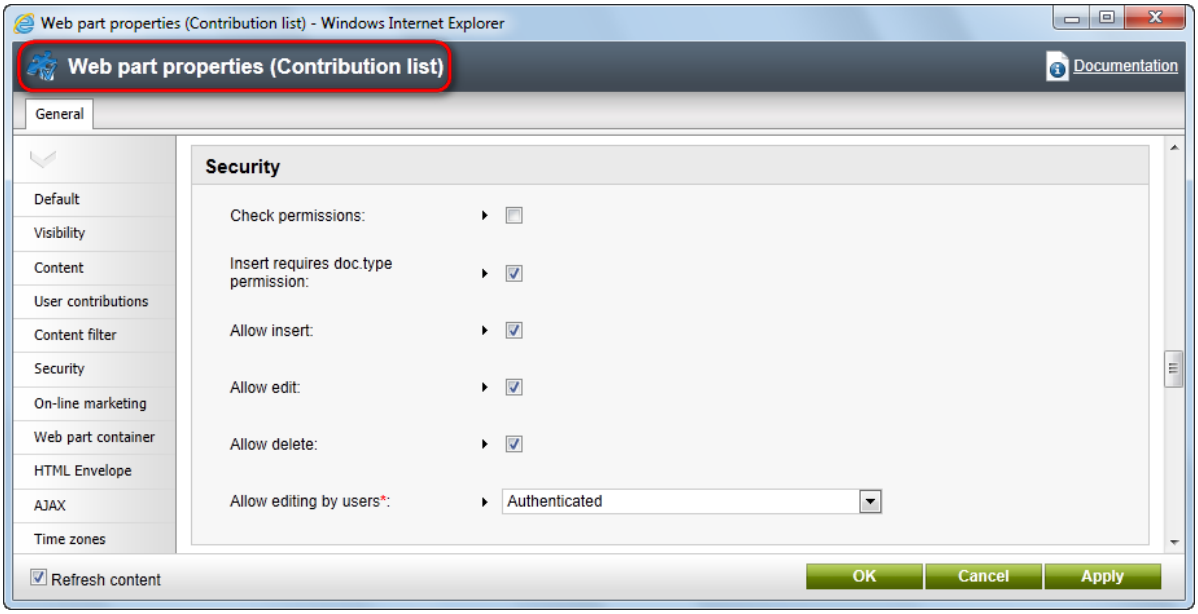
If the user name/document alias path mapping does not fit your needs or if you need to assign multiple users to the same partner profile, you can use another option: Go to **Site Manager -> Development -> System tables -> User** and create a custom text field **PartnerProfileAliasPath**. You will use this custom field in the user profile to specify the partner profile that can be edited by the given user. Then, the **Path** property in the **Edit contribution** web part will be set to value ***{%PartnerProfileAliasPath%}***. Please note that you will find your new custom field on the **Custom Fields** tab, not on the **General** tab as the default attributes.

8.50.4 Security

The User contributions (Wiki) web parts use the following properties to configure their security options:

- **Check permissions** - if you choose this option, appropriate permissions to read/modify/create/delete documents using the User contributions (Wiki) web parts need to be granted to the users. See the Permissions chapter for more details on document permissions.
- **Insert requires doc.type permission** - indicates if document type permissions are required to create a new document.
- **Allow insert/edit/delete** - indicates if the respective buttons should be displayed.

- **Allow editing by users** - you can choose between:
  - **All** - any user who comes to the page with the web part can use it to edit documents
  - **Authenticated** - any authenticated user (site member) can edit the documents; you can use this value in combination with the *NodeOwner = {%CMSContext.CurrentUser.UserID%}* value in the *WHERE condition* property for the web part to display only documents created by the current user (and therefore allow editing of these documents only to them)
  - **Document owners** - only owner of the parent document under which the user contribution documents are stored can edit them



8.50.5 User contributions and API

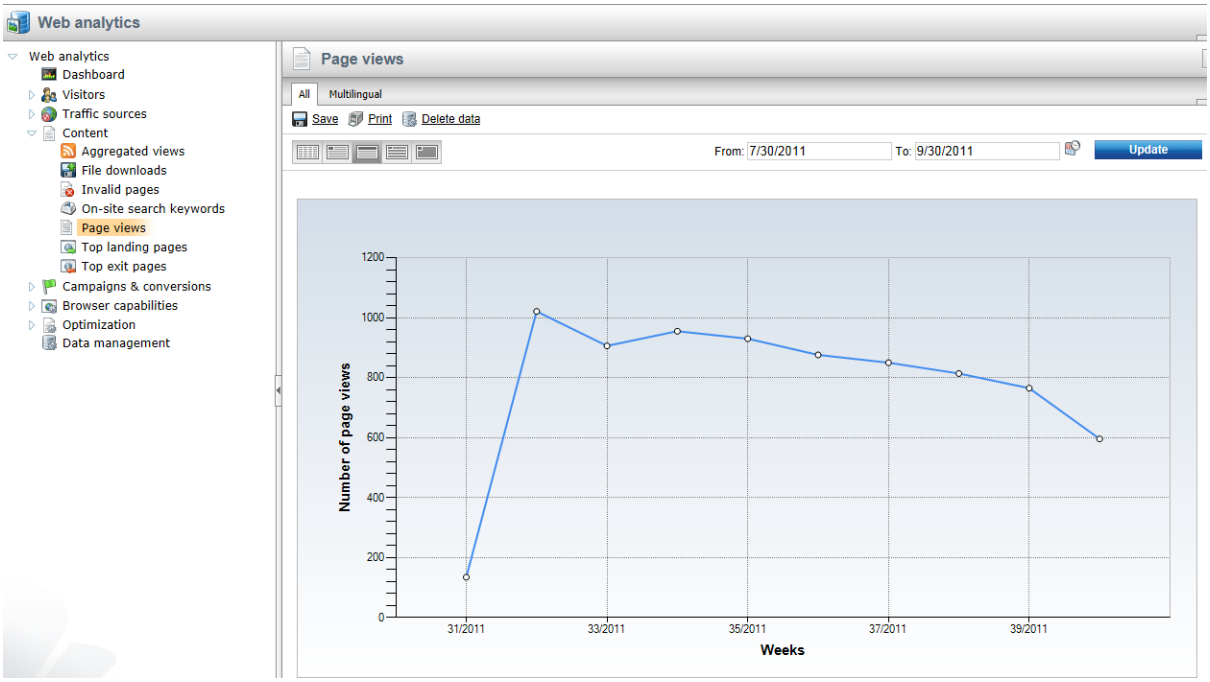
User contributions (Wiki) rely on standard Kentico CMS content management features. As such, they do not have a special programming interface and you need to use the document management API described in the Content management internals chapter.

8.51 Web analytics

8.51.1 Overview

The Web analytics module allows you to track and analyze metrics of your website such as visits, page views, file downloads, traffic sources and much more. Web analytics only measure activity on the live site. The statistics of the pages belonging to the Kentico CMS administration interface (CMS Desk and Site Manager) are not tracked.

You can access the web analytics interface in **CMS Desk -> Tools -> Web Analytics**. Various types of statistics are available in the tree menu, which track events that occur on your website. When you select a statistic, the page displays the corresponding data in a report.



Please read the Web analytics reports topic to find out more about individual statistics and learn about the functionality provided by the built-in reports.

Even though the module provides many statistics and tracking options out-of-the-box, you may also wish to track custom events that are unique for your website. For instructions on how to do this, please refer to the Adding custom analytics topic.

How web analytics work

When a tracked event (such as a page view, file download, etc.) occurs, the system stores a record in the application's memory. The application then periodically writes the memory records into log files in the *~/App_Data/CMSModules/WebAnalytics* folder. The names of the files use the following format:

<event type>_<date>_<time>.log

Disabling memory logging for analytics

The intermediate memory storage step improves the performance of the web analytics logging on high-traffic websites. The logging does not use up a significant amount of memory, but you can configure the system to directly log analytics records into physical files if you need to fully minimize the application's memory usage.

Add the following key to the *appSettings* section of the application's web.config file:

```
<add key="CMSWebAnalyticsUseMemoryStorage" value="false" />
```

The **Process analytics log** global scheduled task reads the content of all analytics log files every minute (or other scheduled interval) and imports the processed data into the database. You can view the status of the scheduled task and configure its settings in **Site Manager -> Administration -> Scheduled tasks**.

This data is then loaded from the database and presented in an easy to read format using web analytics reports (defined through the Reporting module).

You can manage the analytics data directly from the interface — either delete old data or generate sample data for evaluation purposes. Please see the Managing analytics data topic for further information.

Limitations

The Web analytics module only tracks content and events related to pages managed by Kentico CMS. It cannot track other content, such as html files or media files that are not served by the CMS.

Delay in the displaying of results

You may notice a delay between the time that an event occurs on the website and the time the web analytics statistics are updated. All tracked events are first stored in temporary files on the local file system and need to be processed by a scheduled task.

8.51.2 Enabling web analytics

The Web analytics module is disabled by default. To enable logging of analytics for your website:

1. Go to **Site Manager -> Settings -> On-line marketing -> Web Analytics**.
2. Check **Enable web analytics**.
3. **Save** the settings.

The other settings in this category enable or disable tracking of individual types of statistics and events. See the Configuration options topic for details.

Disk Permissions

The Web analytics module requires your ASP.NET account to have the *Modify* permission for the *~/App_Data* folder on your disk.

See the Disk permissions problems chapter for the names of the account under various operating systems.

Selecting the logging method *[Only available after applying hotfix 7.0.17 or newer]*

The system provides two different ways of logging web analytics:

Processing analytics on every web request	Using JavaScript
Advantages: • Provides statistics for all types of web requests.	**Advantages**: • Filters out statistics generated by non-human tools such as RSS readers and web crawlers. • Provides more accurate and relevant data for most public-facing websites.
Disadvantages: • Results may be skewed by non-human traffic such as web crawlers, RSS readers and other bots (typically irrelevant statistics).	**Disadvantages**: • Ignores users with browsers that do not support JavaScript or have it disabled (typically less than 1% of all visitors). • If your website uses custom analytics, JavaScript logging does *not* affect their functionality. This may lead to inconsistency with the default analytics. • May not work correctly on pages that run custom JavaScripts (if conflicts occur). The logging is compatible with all default Kentico CMS scripts.

In most cases, it is recommended to use JavaScript logging for web analytics. Switching to JavaScript logging does not reduce the performance of the website.

To enable JavaScript logging:

1. Go to **Site Manager -> Settings -> On-line marketing -> Web Analytics**.
2. Enable the **Log via JavaScript snippet** setting.
3. Save the settings.

Note: This setting also determines how the system logs on-line marketing activities related to website traffic.

8.51.3 Web analytics reports

The data logged by the web analytics module can be viewed and analyzed using reports displayed in the **CMS Desk -> Tools -> Web analytics** interface.

The reports are organized in a tree menu that can be used for navigation.

The first item in the tree is a Dashboard page, which may be personalized to display unique content for individual users. This way, users can quickly access graphs or tables containing data of the statistics that they need most frequently, view side-by-side comparisons, etc. This can be done by adding (✚) the Analytics chart viewer or Analytics table viewer widgets from the **Reporting** category, and configuring their properties to display the desired graph or table from a specific report.

Reports that track the statistics and events described in the table below are included in the web analytics tree menu by default:

Visitors	
Countries	This statistic shows from which countries visitors access the website. The countries are recognized according to the IP addresses of visitors, which may not be 100% reliable in all cases, but the overall statistics for a high number of visits should provide correct results.
Visitors	Displays the number of unique visits on the website that occurred over the specified time period. A single visit includes any number of page views or other actions performed by a specific user during one session. Also displayed is the ratio between the number of new and returning visitors. A visitor is considered as returning after being inactive for the amount of minutes specified in the **Site Manager -> On-line marketing -> Web analytics -> Visitor idle time (minutes)** setting. Visitors are recognized according to the presence of a browser cookie. Visitors may additionally be tracked according to their IP address. To enable this functionality, enter a value greater than *0* into the **Remember visitors by IP (minutes)** setting. This ensures that visitors will be remembered for the specified number of minutes even if their browser does not save cookies.
Registered users	Displays the total amount of new users that registered on your website during the specified time period, as well as the names and IDs of individual users.
Search crawlers	Displays the number of pages that were visited by search engine web crawlers (robots). Statistics are also provided for individual types of crawlers. Tracking is only done for crawlers whose user agent is specified for one of the search engines registered in the system. Please see Monitoring traffic from search engines for more information.
Mobile devices	Displays the number of visitors who accessed the website using a mobile device. The report also provides a comparison between the number of mobile and standard visits, as well as detailed statistics for individual types of mobile devices.
Traffic sources	
All traffic sources	Displays the number of page views that the website received, categorized according to the type of the traffic source. Three different types of sources are tracked: • **Direct** - page views gained when the URL of one of the website's pages is entered directly into the browser (i.e. no referrer was passed) . • **Referring sites** - page views gained through links from external websites (not including search engine result pages). • **Search engines** - views generated by search engines.

	References from the website's local pages (e.g. navigation menus) are not included in this statistic.
Referrals	Lists the full URLs of external pages from which visitors followed links to your website. You can also see the number of page views gained from each referring URL. **Note**: After applying hotfix 7.0.25 or newer, referrals do not include search engine result pages. The system excludes all URLs that match the **Domain rule** of one of the search engines defined in *Site Manager -> Development -> Search engines*.
Referring sites	Displays the total amount of page views gained through links from external websites. You can view the statistics for individual website domains. **Note**: After applying hotfix 7.0.25 or newer, referring sites do not include search engines. The system excludes all websites whose URLs match the **Domain rule** of one of the search engines defined in *Site Manager -> Development -> Search engines*.
Search engines	Displays the amount of page views generated by visitors who found the website through a search engine, as well as the statistics of individual engines. See Monitoring traffic from search engines for more details.
Search keywords	Displays which keywords were entered into search engines in order to find the website and the amount of page views generated by individual keywords.
Content	
Aggregated views	This statistic tracks access to pages via links in RSS or Atom feeds created using the Syndication module. On the **All** tab, the total number of pages viewed this way is displayed, as well as the statistics of individual documents. The **Multilingual** tab contains the same type of data, but page views are categorized and displayed separately according to the content culture of the given pages.
File downloads	Displays the number of files downloaded by website visitors and the statistics for individual files. On the **All** tab, the total statistics are shown for all files regardless of their assigned content culture. The **Multilingual** tab contains the same type of data, but the number of downloads for files belonging to different content cultures is tracked and displayed separately. Please note that only files stored as documents in the website's content tree are tracked.
Invalid pages	Tracks page requests that contain the website's domain name, but specify a path to a page that does not exist. The total amount of invalid

	requests logged during the specified time period can be viewed, and the paths and statistics of individual requests are also displayed.
On-site search keywords	Displays the total number of searches that were performed using the website's local search functionality and the keywords that were entered.
Page views	Monitors how many times the website's pages were viewed by visitors. On the **All** tab, the displayed data includes the total amount of views for the entire website, and specific information for individual pages. If a page is available in multiple content cultures, the view count of all its versions is added together and tracked as a single page. The **Multilingual** tab contains the same type of data, but the views of pages that belong to different content cultures are tracked and displayed separately. Only pages that are served by Kentico CMS are included in the statistics.
Top landing pages	Displays which pages are the first ones viewed by visitors when they start their browsing session on the website. On the **All** tab, all culture versions of particular landing pages are tracked together as a single page. The **Multilingual** tab may be used to view separate statistics for individual page versions that belong to different content cultures.
Top exit pages	Keeps track of the final pages that were visited by users when their browsing session ended. On the **All** tab, all culture versions of particular exit pages are tracked together as a single page. The **Multilingual** tab may be used to view separate statistics for individual page versions that belong to different content cultures.
Campaigns & conversions	
Campaigns	This category contains reports used to track the progress of marketing campaigns and evaluate their results. Further information can be found in the Campaigns sub-chapter.
Conversions	Displays the number of conversions that were performed on the entire website by users over the specified time period. The statistics of individual conversions are also included. To learn more about conversions, please see the Conversions sub-chapter.
Browser capabilities	
Browser types	This statistic shows what type of browsers are used by visitors to view the website. The name and version number of each visitor's browser is logged, for example: *FireFox3.6*
Please note: The statistics below can only be tracked for visitors who access a page that contains the **Analytics browser capabilities** web part.	
Flash support	Tracks if the browsers used by visitors support viewing of Flash animations and videos.

Java support	Tracks if the browsers used by visitors support Java applets.
Operating system	This statistic logs which operating systems are used by the website's visitors.
Silverlight support	Tracks if the browsers used by visitors support Microsoft Silverlight.
Screen colors	Tracks the color depth that can be displayed in the visitors' browsers.
Screen resolution	Logs the screen resolution used by the website's visitors.

Tracking browser capabilities

If you wish to log detailed browser information other than the type and version, it is necessary to use the Web analytics -> Analytics browser capabilities web part. You can choose which statistics should be tracked by configuring the properties of the web part.

This web part utilizes JavaScript to collect the necessary data, which may in some cases interfere with other scripts on the page, so it is up to the website's developers to determine where it should be located.

To provide the most accurate statistics, it is recommended to place the web part on a page that most visitors will pass through, such as the website's default page or a frequently used landing page.

The reports in the **Optimization** category are used to display the statistics of page optimization tests. Information about these sections can be found in the chapters dedicated to the A/B testing and Multivariate testing modules.

Viewing a web analytics report

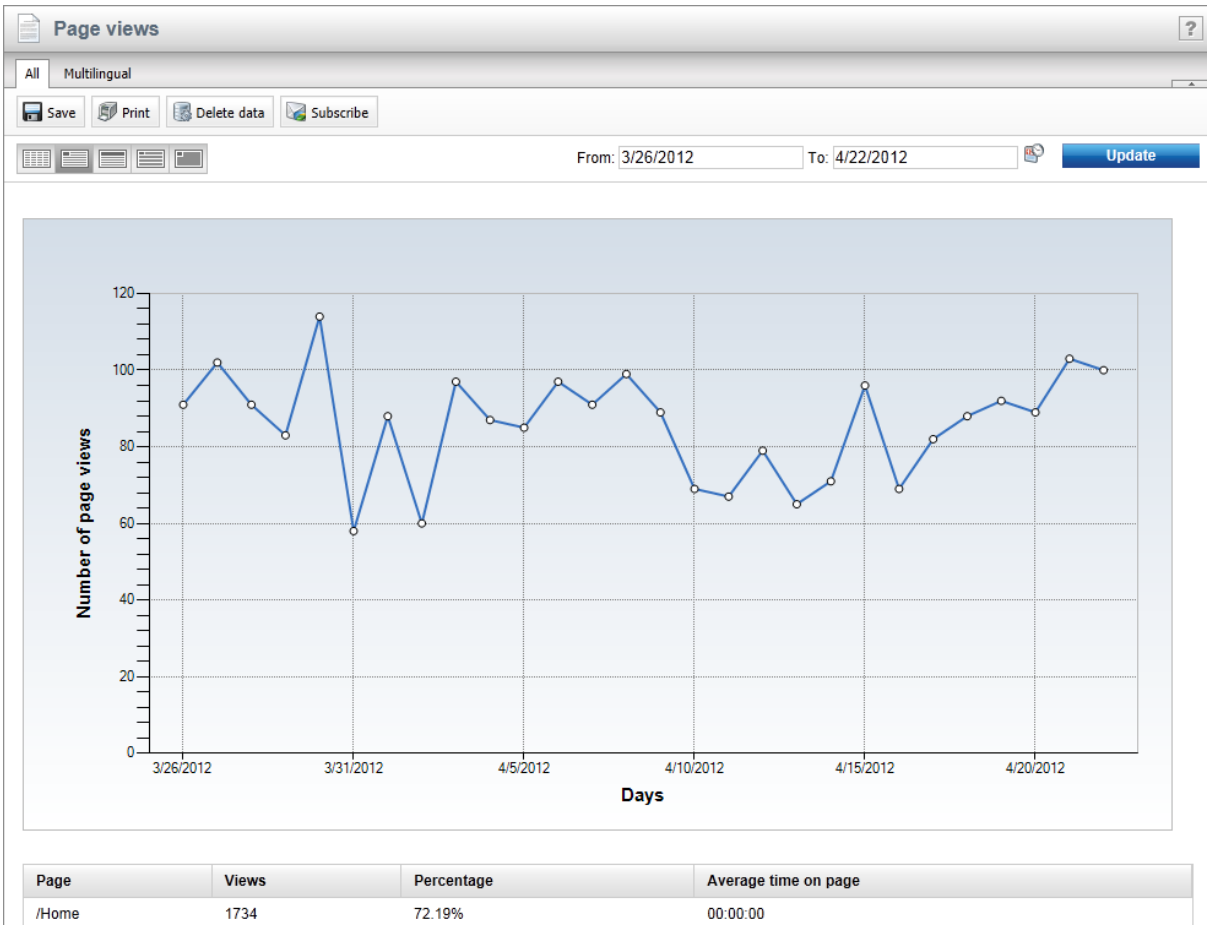
When a statistic is selected from the tree menu in the web analytics interface, its report will be displayed on the right. For most statistics, the data is organized into a graph showing the hits measured over time and a table containing detailed information for individual objects.

The **From** and **To** fields on the right can be used to enter a time period for the report. Only hits that were logged for the statistic during the specified interval will be included in the displayed data.

The following options allow you to choose which unit of time should be used in the statistic's report:

- **Hour**
- **Day**
- **Week**
- **Month**
- **Year**

This selection determines the length of time which is represented by individual units on the horizontal axis of the report's graphs (if there are any) and the precision that can be specified in the **From** and **To** fields.



The following actions may also be performed for every report:

- **Save** - saves the report in its current state (according to the selected time interval). To view saved reports at a later time, go to *CMS Desk -> Tools -> Reporting*, select the matching report under the *Web analytics* category and switch to the *Saved reports* tab.
- **Print** - allows the report to be printed. The available options depend on the used browser.
- **Delete data** - clears all data measured for the given statistic. Please note that this permanently removes all of the statistic's hits from the database. This action is only available for users who have the *Manage data* permission for the Web analytics module. It is also possible to delete data for all statistics at once, as described in the Managing analytics data topic.
- **Subscribe** - opens a new dialog where you can subscribe to the report. Subscribing allows users to periodically receive e-mails with the up-to-date content of the given report. It is also possible to subscribe to a specific reporting component (graph or table) by right clicking on it and selecting the *Subscribe to* option in the displayed context menu.

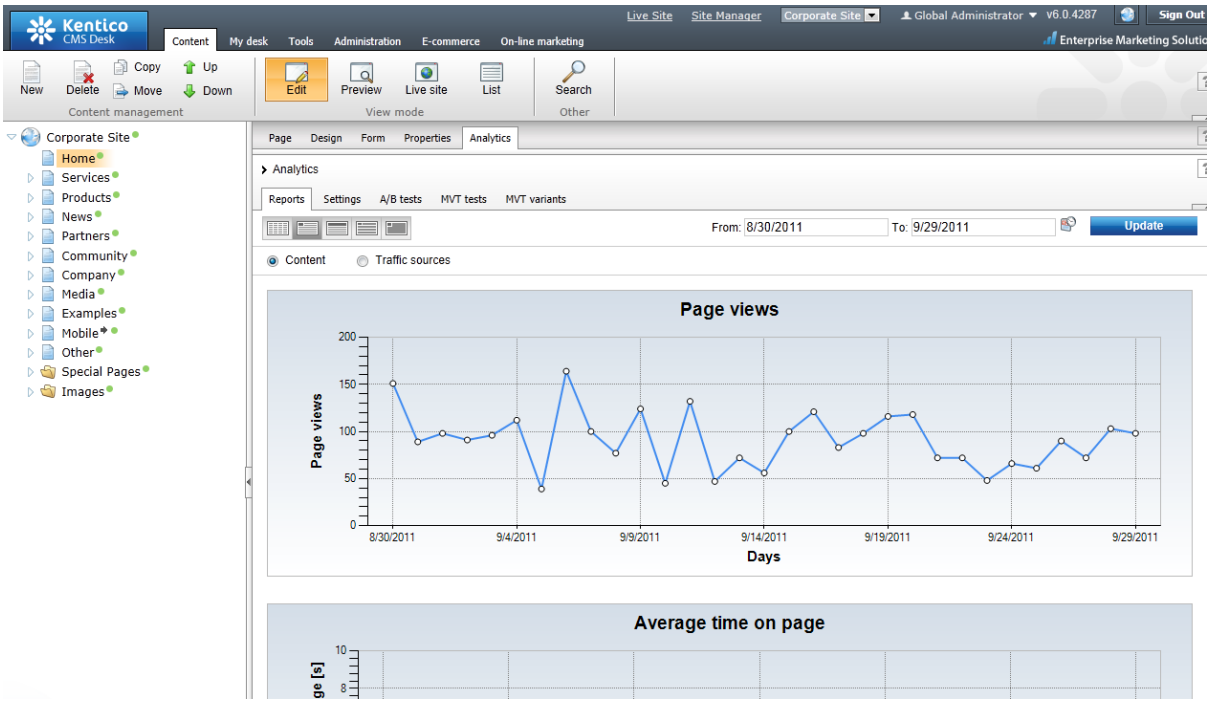
The data displayed in the reports may also be exported into external files using various formats. This can be done by right clicking on a graph or table in the given report, which will open a context menu offering the following export options:

- **Export to Excel** - exports the data displayed by the given object to an XLSX spreadsheet.
- **Export to CSV** - exports data to a CSV file.
- **Export to XML** - exports data to an XML file.

After you select an action from the menu, your browser's standard file download dialog will pop up, letting you open or save the file with the exported data just like when downloading any other type of file. For more details on the data export feature, please refer to the Modules -> UI data export chapter of this guide.

Reports for individual pages

It is also possible to view the values of web analytics statistics measured for specific website pages. To do this, go to **CMS Desk -> Content -> Edit** and select the document representing the page that you wish to examine. Then switch to the **Analytics** tab and select **Reports**.



A time interval can be specified for the reports here and their data may be exported in the same way as when using the main web analytics interface.

The reports for each page are divided into two categories:

- **Content**:

Page views	Shows how many times the given page was accessed by the website's visitors during the specified time interval.
Average time on page	Displays the average time that visitors spend viewing the given page (measured in seconds).
Landing page	Counts how many times the given document served as a landing page for the website's visitors. A landing page is the first page viewed when a visitor starts their browsing session on the website.
Exit page	Counts how many times the document was an exit page for a visitor. An exit page is defined as the last page visited by a user before their

	browsing session ended.

- **Traffic sources**:

Traffic sources	Displays the statistics of the traffic sources that generated the document's page views. Four different types of sources are tracked: • **Direct** - page views gained when the URL of the page was entered directly into the browser (i.e. no referrer was passed). • **Referring local pages** - views gained as a result of references (links) from other pages on the given website (e.g. through a navigation menu). • **Referring sites** - views gained through links from external websites (not including search engine result pages). • **Search engines** - views generated by visitors who found the page using a search engines.
Search engines	This table contains the names of the search engines that visitors used to find the given page and the amount of page views generated by each engine.
Search keywords	Contains the keywords that were entered into search engines in order to find the page and the page view statistics of individual keywords.
Referring sites	Displays the domain names of external sites from which visitors were linked to the given page. You can view the statistics for individual sites. **Note**: After applying hotfix 7.0.25 or newer, referring sites do not include search engines. The system excludes all websites whose URLs match the **Domain rule** of one of the search engines defined in *Site Manager -> Development -> Search engines*.
Referring local pages	Contains the alias paths of the local website's pages from which visitors were linked to the given page and the amount of views gained from each page.
Crawlers	Shows how many times and how recently the page was visited by search engine web crawlers (robots).Tracking is only done for crawlers whose user agent is specified for one of the search engines registered in the system.

8.51.4 Conversions

8.51.4.1 Overview

The web analytics module provides a way to track actions that can be performed by your website's visitors and record them as *conversions*. This is typically done for desired events that somehow benefit the website, such as the registration of a new user, a product order, subscription to a newsletter or similar. Conversions are represented in the system by corresponding tracking objects, which are described in more detail in Managing conversions.

Once an action is defined as a conversion, a *conversion hit* is logged whenever it occurs. Additionally, a numerical value may be stored along with each hit to indicate its importance. To learn how you can

assign conversions to individual types of actions and ensure that they are logged correctly, please refer to the Logging actions as conversions topic.

Once you start tracking conversions on the live site, you can compare the recorded statistics with the values of other web analytics metrics, such as the total amount of visitors. This allows you to evaluate the website and adjust it as necessary.

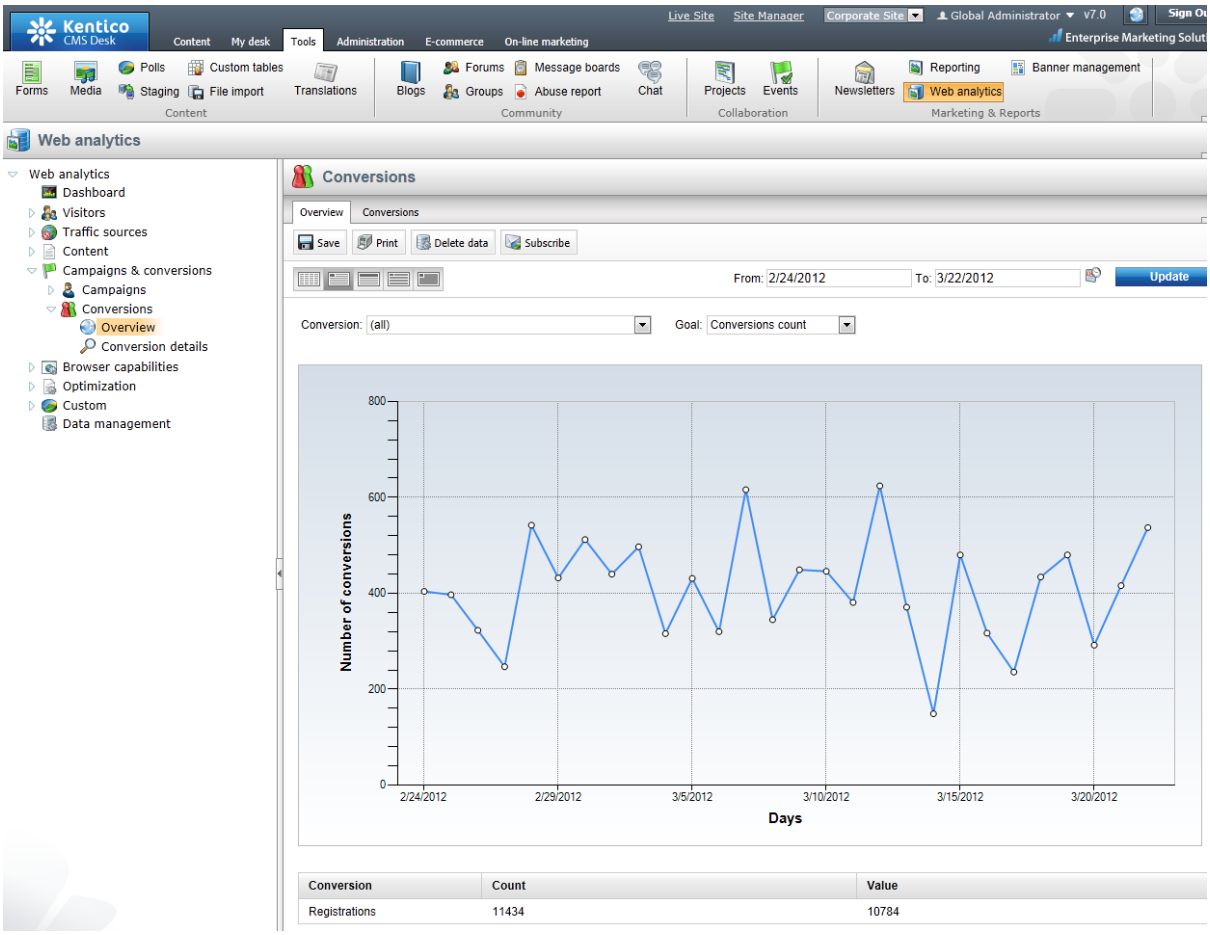
While simply tracking all conversions that occur on the website can be useful, in many cases you may also require additional information about the context in which the given actions occurred. For this reason, conversion tracking is integrated with several other web analytics and on-line marketing features. When used together with Campaign tracking, conversions allow you to record actions only for visitors who arrive on the website in a specific way (e.g. as a result of a marketing campaign).

If you wish to optimize your site to increase its conversion rate (i.e. make it more user friendly to get better results), you may utilize A/B or Multivariate testing. These features allow you to accurately measure how changes made to the content or structure of your website's pages affect the behavior of users.

8.51.4.2 Managing conversions

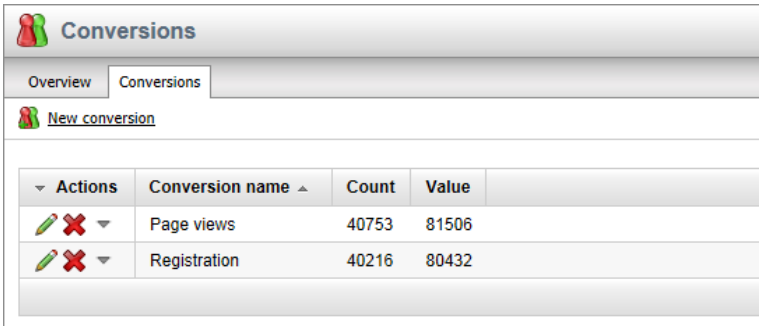
To access the management interface dedicated to conversions, go to **CMS Desk -> Tools -> Web analytics**, expand the **Campaigns & conversions** category in the tree on the left and select **Conversions**. This section of the UI may alternatively be reached by navigating to **On-line marketing -> Conversions**.

The **Overview** contains a web analytics report displaying the conversions that were logged on the website over the specified time period.



This report only provides a general overview of the conversion statistics. Additional information about conversions that were logged under special circumstances is available in the **Conversion details** report and in the specialized reports under the **Campaigns** and **Optimization -> A/B tests** or **MVT tests** categories.

If you switch to the **Conversions** tab, you can view a list of all conversions defined for the current site and manage them as necessary.



To define a new conversion, click **New conversion** and fill in the following properties in the displayed dialog:

- **Conversion display name** - the name of the conversion displayed in the administration interface.

- **Conversion code name** - sets a code name that serves as a unique identifier for the conversion.
- **Conversion description** - can be used to enter text describing the conversion's purpose.

The entered data may be modified at any time by editing () the given conversion object on the **General** tab.

The **Campaigns** tab allows you to configure which campaigns should track the currently edited conversion as part of their statistics. By default, campaigns log all possible conversions, so it is only necessary to assign campaigns that are configured to work with a limited set of conversions. This type of configuration is not available for A/B or multivariate tests, which automatically keep track of all conversions performed on the website.

As you can see, the objects representing conversions are very simple and do not require any advanced configuration. It is however necessary to assign the conversions to the appropriate actions to ensure that they are logged correctly. This can be done through various other parts of the Kentico CMS administration interface as described in the Logging actions as conversions topic.

8.51.4.3 Logging actions as conversions

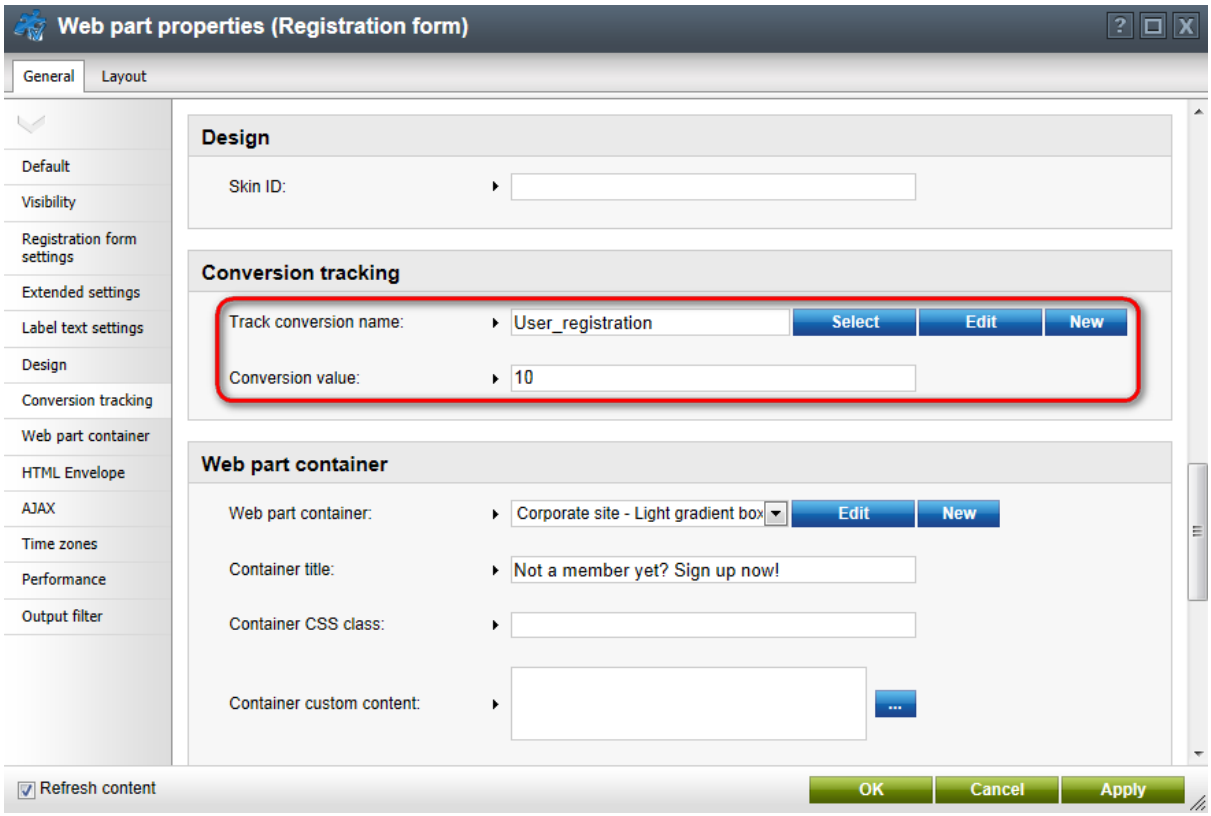
This topic describes the possible options that may be used to ensure that the system logs specific events as conversions. When assigning a conversion through the user interface, two types of fields are provided.

The first is a standard conversion selector. You can either enter the *code name* of a conversion into the text box or click the **Select** button to choose from a list of conversions defined for the current site. If you enter a name that does not match any conversion in the system, a conversion with this name will automatically be added. The **New** and **Edit** buttons allow you to create a new conversion or modify the properties of the selected one directly from the given part of the user interface.

The second field is optional and provides a way to set a number that will be recorded along with the conversion when the tracked action is performed. This may be used to indicate the relative importance of the conversion, the profit generated by a single conversion hit or similar. The values are cumulative, i.e. when a conversion hit is logged, the specified value will be added to the total sum previously recorded for the conversion. You may insert a Macro expression into this field to dynamically retrieve a value from the current site context. For examples of conversion value macros, please see the sections below dedicated to individual types of actions.

Web part and widget actions

Many of the default Kentico CMS Web parts and Widgets that allow users to perform important actions come with built-in support for conversion tracking. To configure a specific web part or widget instance to log actions as conversions, open its properties dialog and enter the appropriate values into the **Track conversion name** and **Conversion value** properties.



Below you can find a list of all types of actions that can be tracked as conversions through web parts:

Action	Web part(s)
User registration	In this case, the conversion will be logged when a visitor successfully completes their registration using the given web part. There are multiple web parts that allow users to register on the website: • Registration form • Custom registration form • Facebook Connect logon and Facebook Connect required data • LinkedIn logon and LinkedIn required data • Windows LiveID and LiveID required data • OpenID logon and OpenID required data
Newsletter subscription	Newsletter subscriptions may be tracked as conversions through the Newsletter subscription or Custom subscription form web part. This can also be done for the widget based on the web part.
Shopping cart actions	The Shopping cart web part may be used to track two types of events: • **Registration** - occurs when a customer registers on the website through the shopping cart. • **Order** - logged when a customer successfully completes an order. You can assign conversions to these actions for specific instances of

	the shopping cart web part through the corresponding **conversion name** properties. The conversion values of these events can be configured for the entire website via the e-commerce website settings described in a dedicated section below.
Filling in an on-line form	A conversion can be logged when a user submits a form displayed by the On-line form web part.
Voting in a poll	The Poll web part may be used to log a conversion whenever a user votes in the displayed poll.

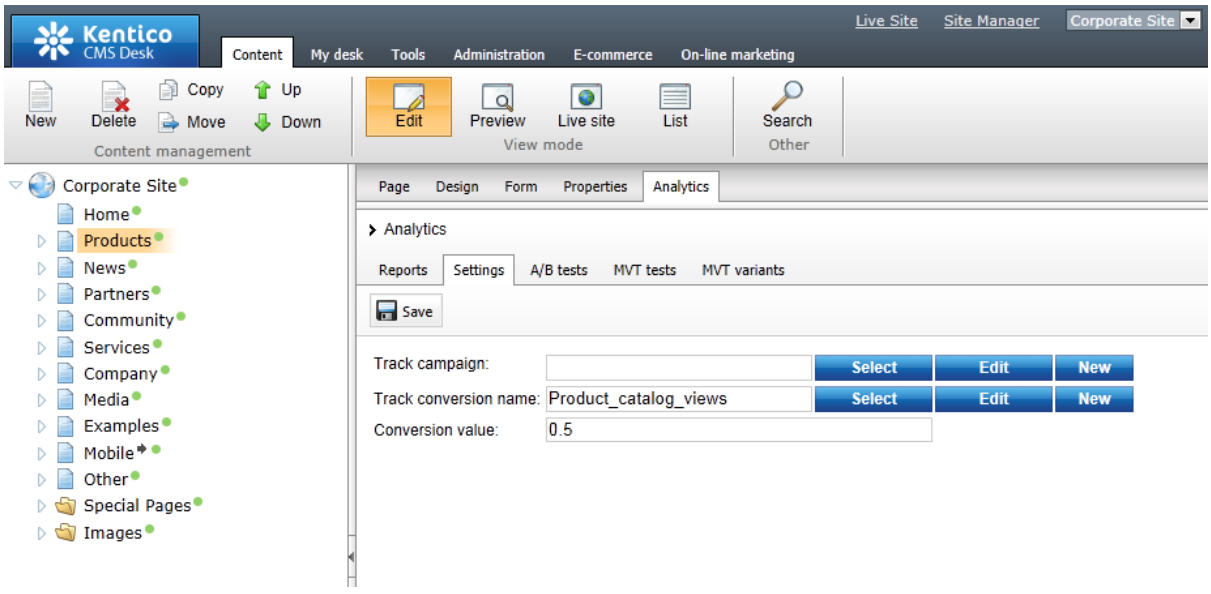
Entering conversion value macros into web part properties

The **Conversion value** properties of web parts only support numeric (decimal) values, so it is not possible to specify a macro expression directly. However, you can enter macros via the **Edit value** dialog that can be opened by clicking the ▸ icon next to the given property.

In the case of widgets, macros entered by users into properties are not resolved at all. If necessary, macro expressions can be pre-set as the default values of widget properties by administrators. This can be done by editing the given widget in **Site Manager -> Development -> Widgets** on the **Properties** tab.

Page views of specific documents

You can also use conversions to keep track of the amount of hits received by individual pages. To configure this behavior for a page, go to **CMS Desk -> Content -> Edit**, select the document representing the given page from the content tree and switch to its **Analytics -> Settings** tab. To assign a conversion and associated value, fill in the **Track conversion name** and **Conversion value** fields as described above.



The specified conversion will be logged every time the given page is requested by the website's users.

E-commerce actions

Conversions may be configured for e-commerce actions that occur on the entire website using the settings in **Site Manager -> Settings -> E-commerce**. There are three types of events that can be tracked:

- **Registration** - occurs when a customer registers on the website through the checkout process.
- **Order** - logged when a customer successfully places a product order.
- **Add to shopping cart** - occurs when a user adds a product to a shopping cart on the website.

You can assign a different conversion and value to each of these actions through the appropriate **conversion name** and **conversion value** settings. The registration and order conversion name settings can be overridden for individual instances of the shopping cart web part through their corresponding properties.

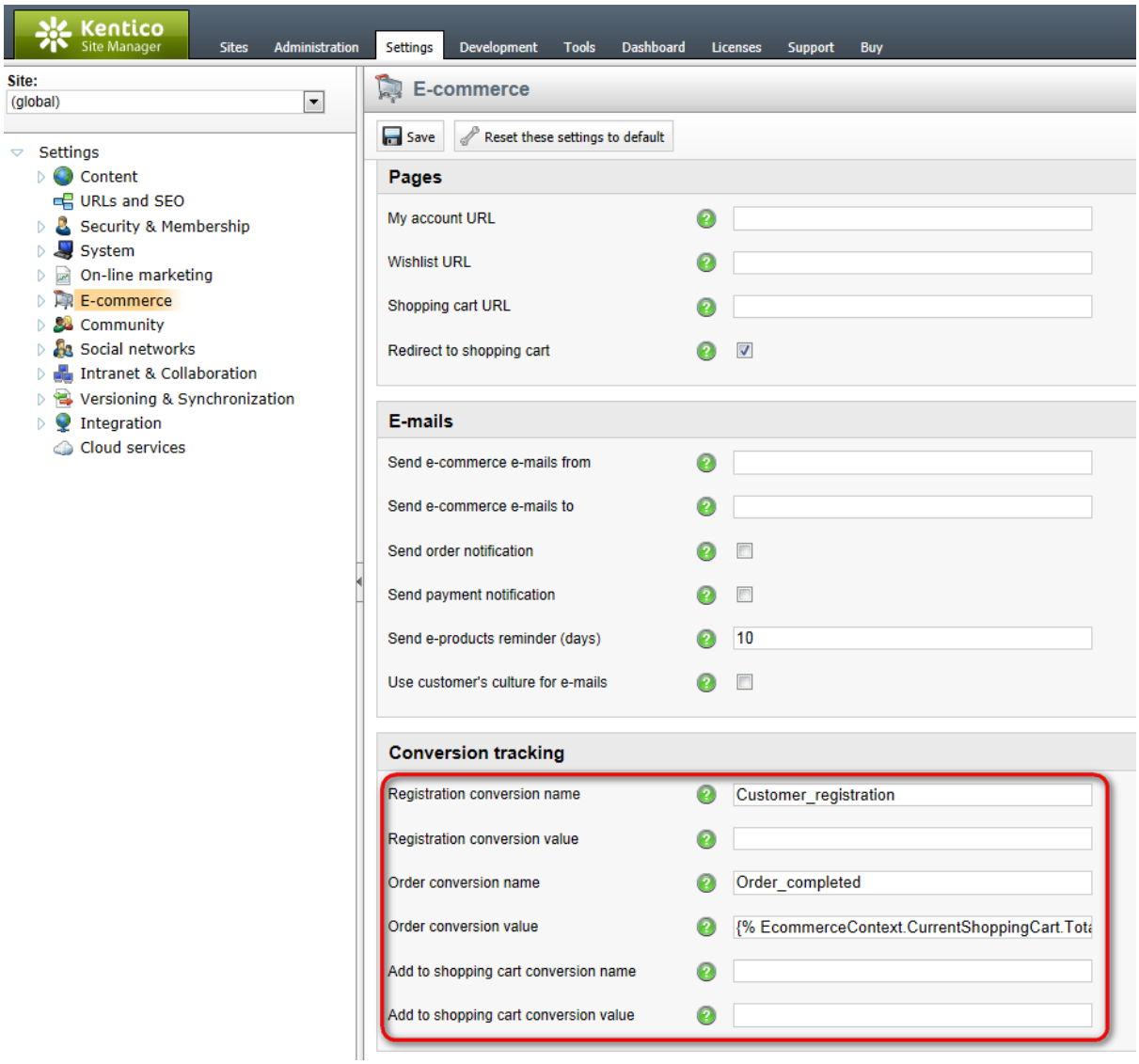
If you wish to log the conversion value dynamically based on item prices, you may use macro expressions, for example:

Order conversion value: *{% EcommerceContext.CurrentShoppingCart.TotalPrice %}*

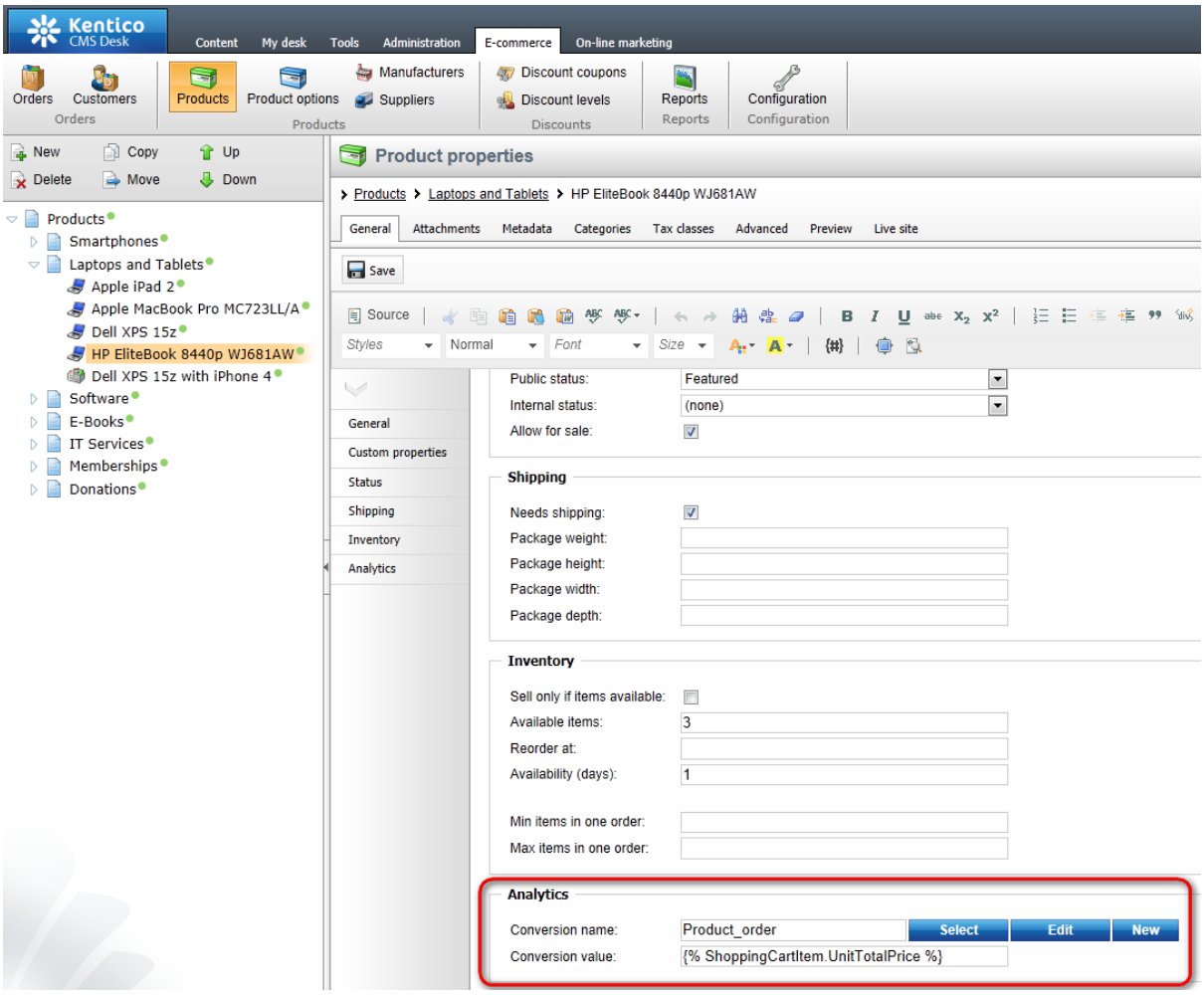
This macro is resolved into the total price of all items contained in the order, including tax and shipping. With this configuration, each *Order* conversion will automatically store the price of the given order as its value.

Add to shopping cart conversion value: *{% ShoppingCartItem.UnitTotalPrice %}*

With this macro as the conversion value, the *Add to shopping cart* conversion will log the price (including tax) of the specific product that was added to the shopping cart.



The settings described above allow you to track entire orders, but in some cases you may wish to log a separate conversion hit every time a product of a specific type is purchased. This can be done by navigating to the product list in **CMS Desk -> E-commerce -> Products** and selecting the particular product (or when editing a product document in the main content tree on the **Form** tab).



Here you can assign a conversion through the **Conversion name** property at the bottom of the product's editing page.

The **Conversion value** field can be used to specify an appropriate value that will be recorded for each conversion hit. In addition to numeric values, you may enter macro expressions here, for example: *{% ShoppingCartItem.UnitTotalPrice %}.* This macro allows the conversion to log the price of the given product as its value. The advantage of a macro is that it retrieves the price dynamically, including tax and any potential discounts applied by the given customer.

Please note that these properties are not available for global products, since each site has its own separate set of conversions.

You can also use the same approach to configure different conversion settings for individual product options (via **E-commerce -> Product options -> edit Category -> Options**).

Logging conversions using the API

If you need to track some other type of action that is not included among the options listed above, you can write your own custom code and use the API to log conversions. This allows you to monitor any type of activity performed by users on your website and view the results using the various conversion

reports available in the web analytics interface.

To log a conversion via the API, you can use code similar to the following:

[C#]

```
using CMS.WebAnalytics;
using CMS.CMSHelper;

...

string siteName = CMSContext.CurrentSiteName;
string aliasPath = CMSContext.CurrentAliasPath;

// Checks that web analytics and conversion tracking are enabled in the site's
settings.
// Also confirms that the current IP address, alias path and URL extension are not
excluded from web analytics tracking.
if (AnalyticsHelper.IsLoggingEnabled(siteName, aliasPath)
    && AnalyticsHelper.TrackConversionsEnabled(siteName))
{
    // Logs the conversion according to the specified parameters.
    HitLogProvider.LogConversions(siteName, CMSContext.PreferredCultureCode,
ConversionName, 0, 1, ConversionValue);
}
```

There are several possible ways to include this type of code in your website's functionality. When tracking activity on a specific page, you may use a custom user control or web part to ensure that the code is executed as required. If you wish to log actions that may occur anywhere on the website, you may utilize global event handlers.

As shown above, conversions can be logged using the **HitLogProvider** class from the **CMS.WebAnalytics** namespace, specifically the following method:

LogConversions(string siteName, string culture, string objectName, int objectId, int count, double value)

- **siteName** - sets the code name of the site for which the conversion should be logged.
- **culture** - sets the culture code under which the conversion should be logged.
- **objectName -** used to specify the code name of the conversion that should be logged.
- **objectId** - used to specify the ID of the conversion. This parameter may be set to *0* if a valid code name is passed via the *objectName*.
- **count** - sets the amount of conversion hits that should logged. This parameter is optional and the default value (*1*) is used if it is not specified.
- **value** - specifies the value that will be logged for the conversion.

In addition to logging a general conversion, this method checks if the current user has passed through a page with a running A/B or Multivariate test, or has arrived on the website through a Campaign. If this is the case, then the conversion is also automatically logged within the appropriate context and included in the statistics of the given test or campaign.

8.51.5 Campaigns

8.51.5.1 Overview

One of the most common techniques used to bring new traffic to a website are on-line marketing campaigns. When attempting to determine if a campaign was cost-effective or when measuring its exact results, simply tracking referring sites or URLs may not always provide sufficient data.

In these cases, the campaign tracking support provided by the web analytics module may be used, which allows you to accurately monitor traffic generated by individual campaigns. This can include banners, marketing e-mails or any other method used to present links to your website. In addition to logging the amount of visits, this feature lets you keep records of any important actions performed on the website by users who arrive as a result of a campaign.

How the tracking works

There are several ways to identify visitors who come to your site through a campaign link (further details can be found in the Managing campaigns topic). When this happens, the system updates the visit statistics and stores a cookie named **Campaign** in the user's browser, which saves the name of the corresponding campaign tracking object as its value. Only one campaign may be assigned to a user at a given time and the current value is overwritten if the user returns to the site through a different campaign.

Until the cookie expires (its duration is 24 hours), any actions performed by the visitor that are defined as conversions will be logged as part of the statistics kept for the campaign. Please refer to the Conversions chapter for information about how the tracking of individual actions can be configured.

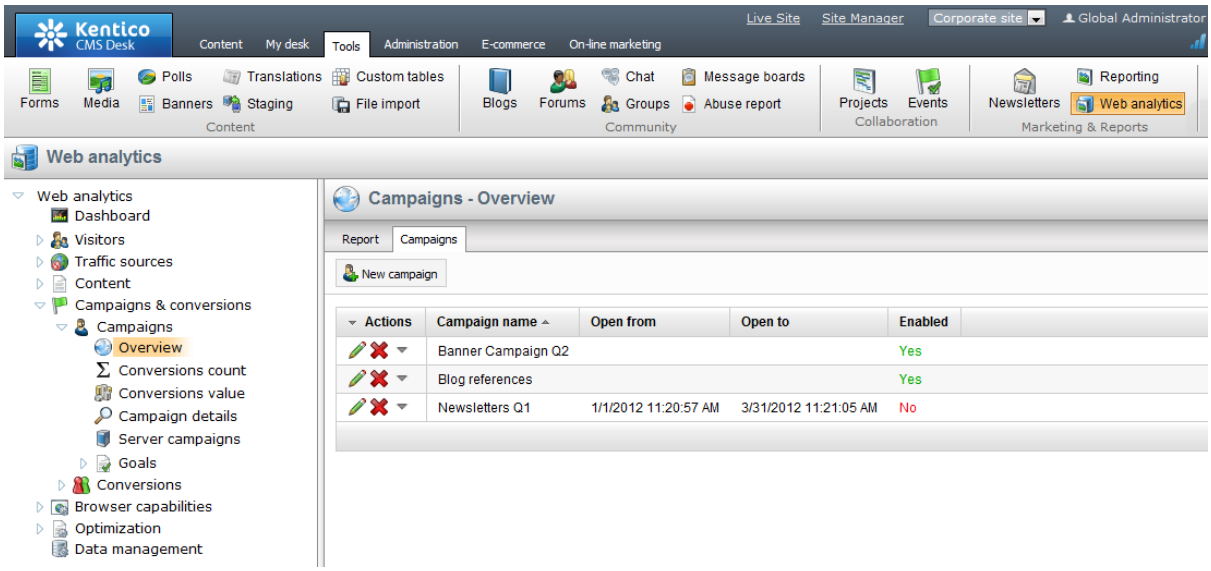
If the visitor registers on the website while the cookie is still valid, the name of the currently active campaign will be permanently stored in their user settings (administrators may view the value in **Administration -> edit ( ) user -> Settings -> Campaign**). This data field only serves to provide information about users and is not used to track activity on the website.

Campaign statistics and results can be viewed using special reports available in the web analytics interface, which are described in the Evaluating campaigns topic.

8.51.5.2 Managing campaigns

Each marketing campaign is represented in the system by a corresponding object. To configure these objects, go to **CMS Desk -> Tools -> Web analytics** and expand the **Campaigns & conversions -> Campaigns** category in the tree. Now select one of the displayed items and switch to the **Campaigns** tab on the right side of the page (alternatively, this interface may be accessed through **CMS Desk -> On-line marketing -> Campaigns**).

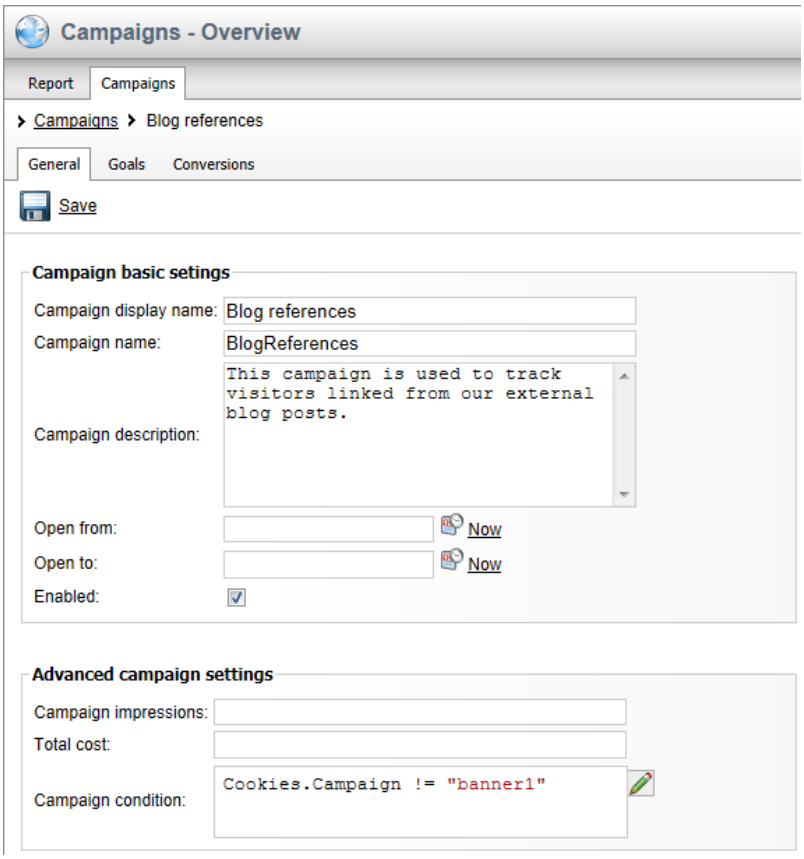
Here you can view the list of all campaigns defined for the current website and manage them as needed.



To create a new campaign, click **New campaign** and fill in the following properties:

- **Campaign display name** - the name of the campaign displayed in the administration interface (in campaign lists and reports).
- **Campaign name** - sets a code name that serves as a unique identifier for the campaign. It is also stored as the value of the *Campaign* browser cookie used to identify visitors who came to the website through the given campaign.
- **Campaign description** - can be used to enter a text description for the campaign in order to give information about its purpose, goals, etc.
- **Open from/to** - sets the time interval during which the campaign should be active. The *Calendar* button () can be used to select an exact date and time.
- **Enabled** - this property may be used to manually start or stop the campaign. Visitor statistics and actions will not be logged for disabled campaigns.

Click **Save**. The **General** tab of the campaign's editing interface will now be displayed, where you can modify the fields listed above at any time. Several additional properties are also available.



The **Campaign impressions** and **Total cost** values may be used to help evaluate the campaign and calculate its goals (for more information, please see Evaluating campaigns).

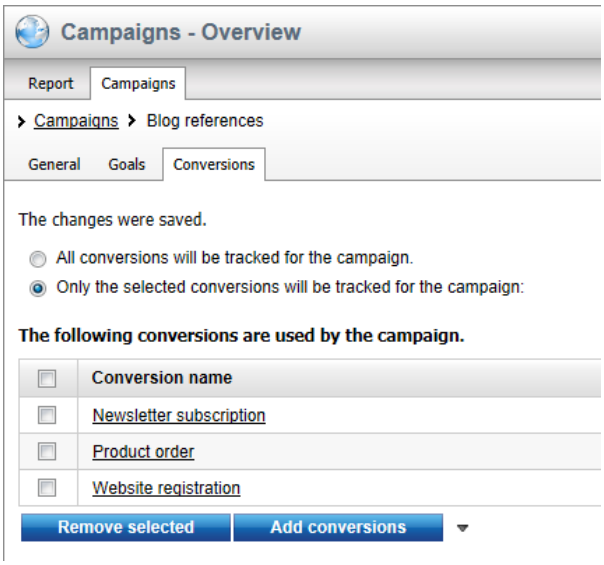
An important property is the **Campaign condition**, which allows you to specify a macro condition that must be fulfilled in order for visitors to be included in the campaign. Through the edit icon (), you can use the Macro condition editor, which provides a way to build conditions through a graphical interface.

For example: *Cookies.Campaign != "banner1"*

This sample condition checks the value of the *Campaign* cookie and prevents the edited campaign from being assigned to visitors who are already being tracked by a different campaign named **banner1**. You can write any other type of condition according to your specific requirements. For details about available macro options and syntax, please refer to the Development -> Macro expressions chapter.

If you switch to the **Conversions** tab, you can configure which conversions should be tracked by the campaign. There are two possible options:

- **All conversions will be tracked for the campaign** - if chosen, any conversion hit logged on the entire website will be included in the campaign's statistics.
- **Only the selected conversions will be tracked for the campaign** - this option allows you to limit which conversions should be tracked. Specific conversions may be assigned to the campaign by clicking the *Add conversions* button and choosing from the list in the displayed dialog.



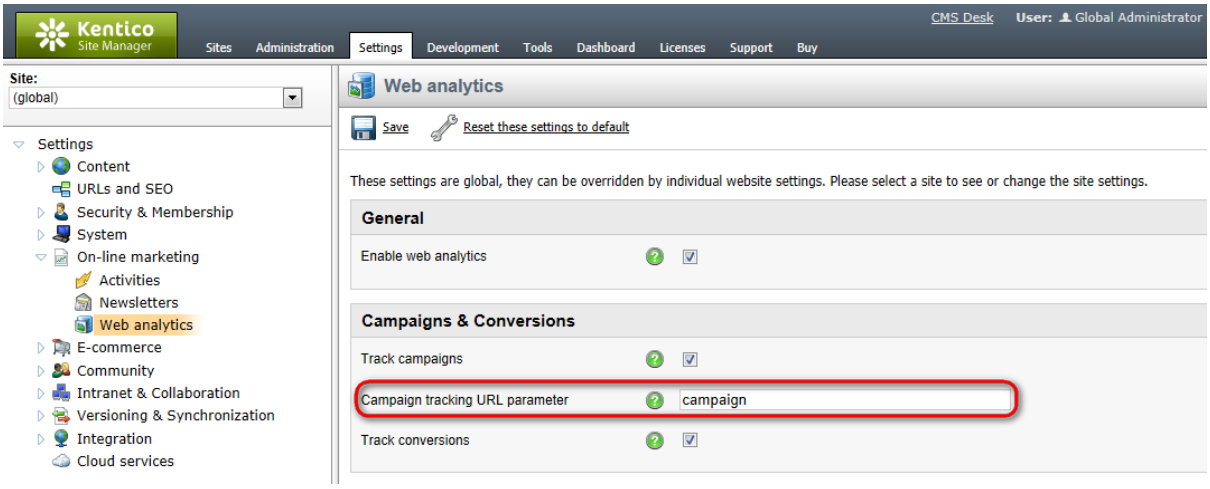
In both cases, only conversions performed by visitors who arrive on the website as a result of the given campaign will be logged.

Configuring campaign tracking

In order to correctly identify visitors who access the website through a marketing campaign and ensure that they are assigned to the appropriate campaign tracking object, you can use one of the three approaches described below.

1. URL parameter

This method utilizes a URL query string parameter to indicate that the traffic source is a campaign. All you need to do is enter the name of the parameter that you wish to use into the **Site Manager -> Settings -> On-line marketing -> Web analytics -> Campaign tracking URL parameter** setting (e.g. *campaign*).



Then, ensure that all link URLs used in the campaign's marketing materials contain this parameter, with the name of the appropriate campaign object as the value. For example:

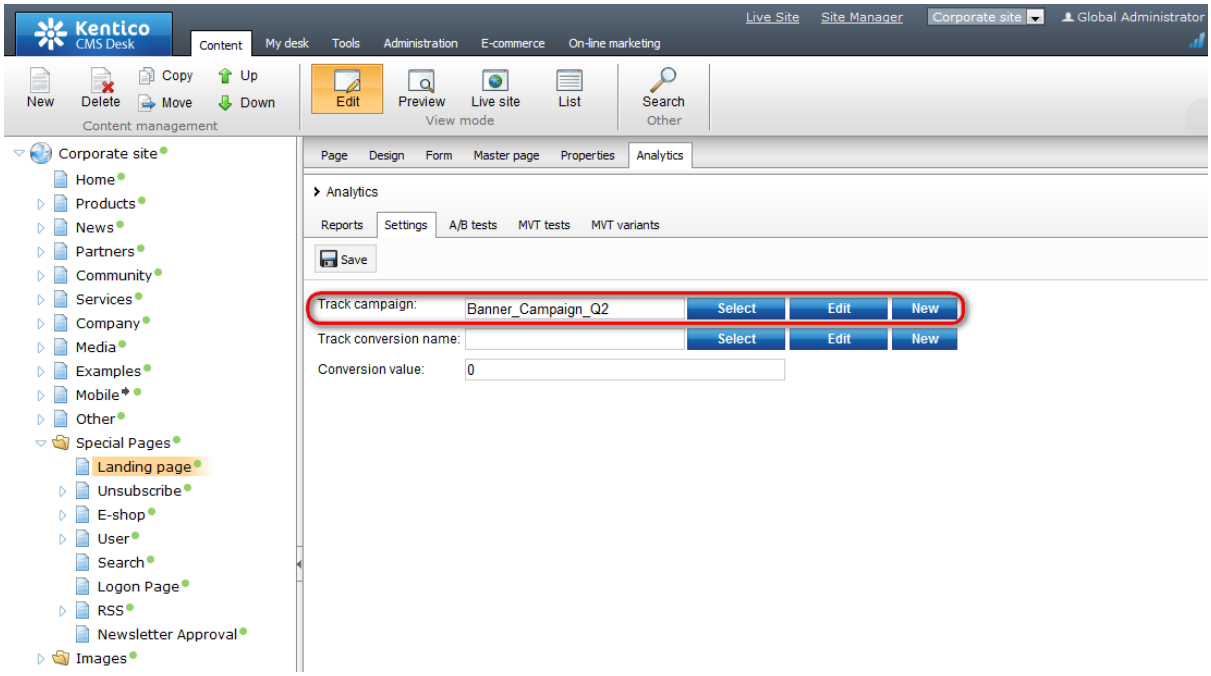
http://www.mywebsite.com/Home.aspx?campaign=banner1
http://www.mywebsite.com/News.aspx?campaign=newsletterJune

This type of campaign tracking is not limited to a specific document and works for all pages on the website.

2. Document specific campaigns

Alternatively, you can configure a document to behave as a campaign landing page. When this type of page is viewed by a user, a specified campaign will automatically be assigned. A visit will also be logged for the campaign every time the document is requested.

To enable this functionality for a page, go to **CMS Desk -> Content -> Edit**, select the given document from the content tree and switch to its **Analytics -> Settings** tab. Here, enter the name of a campaign into the **Track campaign** field. The **Select** button allows you to choose from a list of existing campaign objects available for the website. **Edit** and **New** may be used to manage campaigns directly from this dialog.

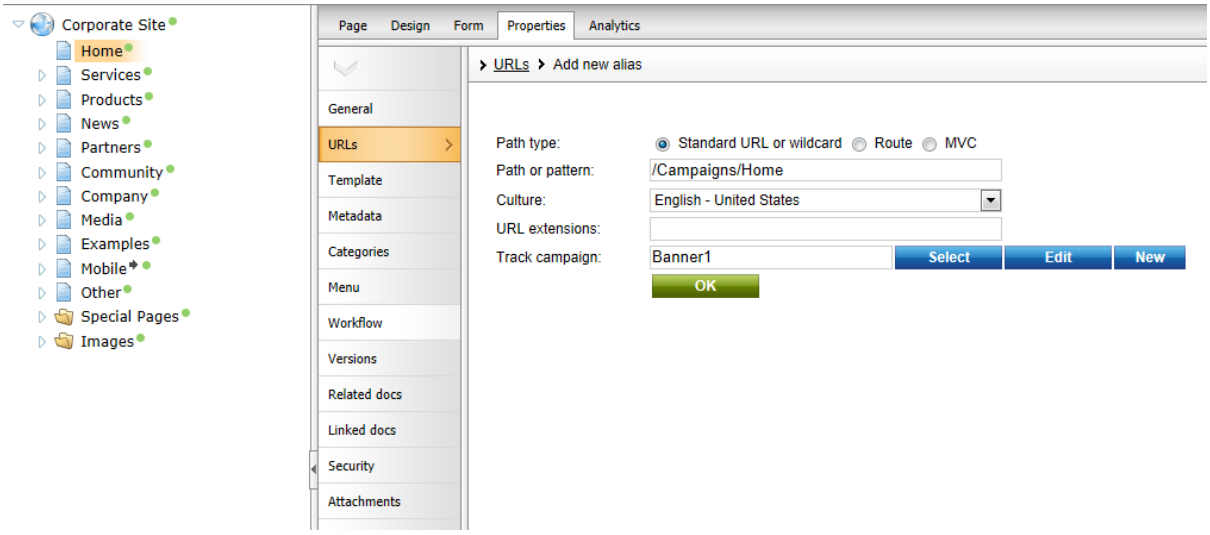


To ensure accurate campaign tracking, it is recommended to create a special page that will not be available in the website's standard navigation, since the campaign will be assigned to all users who view this page (and repeated views by the same user will also be logged as campaign visits). When using this approach, all of the campaign's links need to be directed at this special landing page.

3. Campaigns for document aliases

You can also define campaign tracking for a page's specific document alias. This approach is useful if you wish for the campaign to link visitors to an easily accessible page, but want to avoid using a query string parameter. The campaign will only be assigned to users who access the document through this alias.

To add a document alias to a page, select it from the content tree in **CMS Desk** and go to **Properties -> URLs**. Here, click **Add new alias** and configure its settings in the displayed dialog. Select **Standard URL or wildcard**, enter a suitable path into the **Path or pattern** field and then specify the name of the campaign in the **Track campaign** property.



Click **OK** to create the alias. Now you can use the URL of this alias in the campaign's links, for example: *http://www.mywebsite.com/Campaigns/Home.aspx*

The campaign tracking methods are listed in the order of their priority. So if a campaign is specified for a document through **Settings -> Analytics**, but the page is accessed via a URL containing a tracking parameter, the campaign specified via the parameter will be assigned to the given visitor.

Automatic campaign creation

When a visitor comes to the website through an undefined campaign (typically specified via the tracking URL parameter), it will be ignored by default and no campaign tracking will be performed.

If you wish to change this behaviour, edit your application's web.config file and add the following key to the **/configuration/appSettings** section:

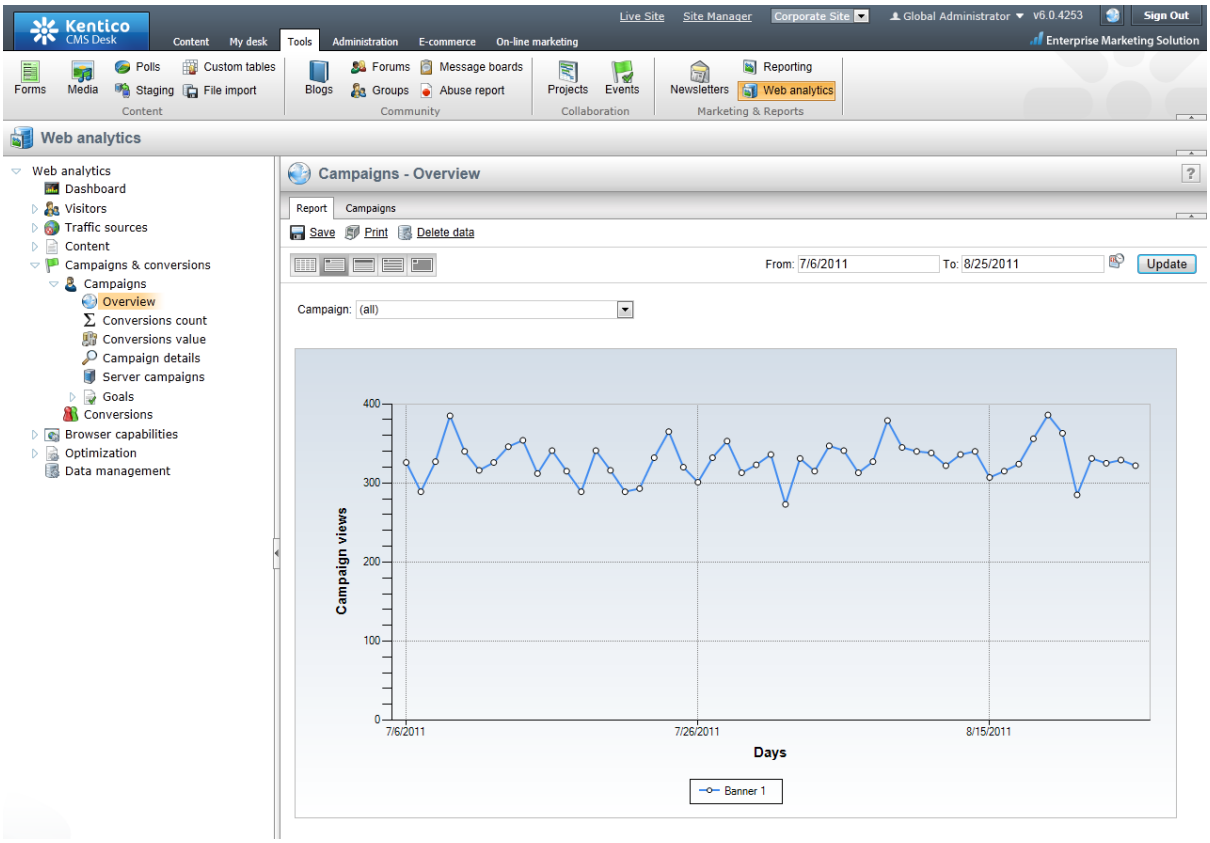
```
<add key="CMSEnableAutomaticCampaignCreate" value="true" />
```

Once this is done, unknown campaigns will automatically be added to the list of campaign objects for the given website.

Campaigns created this way will be enabled by default and active for an unlimited time interval, but they can be configured using the same approach as mentioned above at any time.

8.51.5.3 Evaluating campaigns

The statistics logged for campaigns may be viewed using various types of web analytics reports. To access these reports, go to **CMS Desk -> Tools -> Web analytics** and expand the **Campaigns & conversions -> Campaigns** category.



Campaign reports use the same basic format as other web analytics reports, so please see the Web analytics reports topic for information about the actions that can be used to specify which data should be displayed. The following types of reports are available for campaigns:

Overview	Displays the amount of hits that the website received as a result of the selected campaign(s). It contains a line chart that shows the number of visits recorded over time and a table with the total amount of page views generated by the given campaigns during the entire time period.
Conversions count	Displays the number of conversion hits that were performed by users who arrived on the website through the selected campaign. This report includes two types of charts, one that shows the progress of the amount of conversion hits recorded during the specified time period and another with detailed statistics for individual units of time according to the selected report type (hours, days, months etc.).
Conversions value	Displays the sum of the conversion values generated by users who arrived on the website through the selected campaign.

	This report includes two types of charts, one that shows the progress of the conversion values recorded during the specified time period and another with detailed statistics for individual units of time according to the selected report type.
Campaign details	Displays the values of the following campaign metrics: • **Visits** - the amount of hits that the website received as a result of the given campaign. • **Conversions** - the number of conversions that were performed by users who arrived on the site through the campaign. • **Conversions rate** - the percentage of visitors that performed a conversion. This can be higher than *100%* if the average visitor generated more than one conversion. • **Conversions value** - the total sum of the values recorded for all of the campaign's conversions. • **Conversions value per visit** - the average conversion value contributed by the campaign's visitors. • **Total cost** - shows the total cost that was specified for the given campaign. • **ROI** - the rate of investment, calculated as the sum of the campaign's conversion values divided by the total cost. This statistic is accurate only if your conversion values are set to reflect the income generated by the matching conversions. You can either choose to view all campaigns defined for the website and compare their values, or select a specific campaign and analyze the statistics logged for individual types of conversions.
Server campaigns	May be used to compare the results of campaigns created for different sites in the system. You can choose to display the *Views*, *Conversion count* or *Conversion value* statistics for the campaigns belonging under the selected site.

Campaign goals

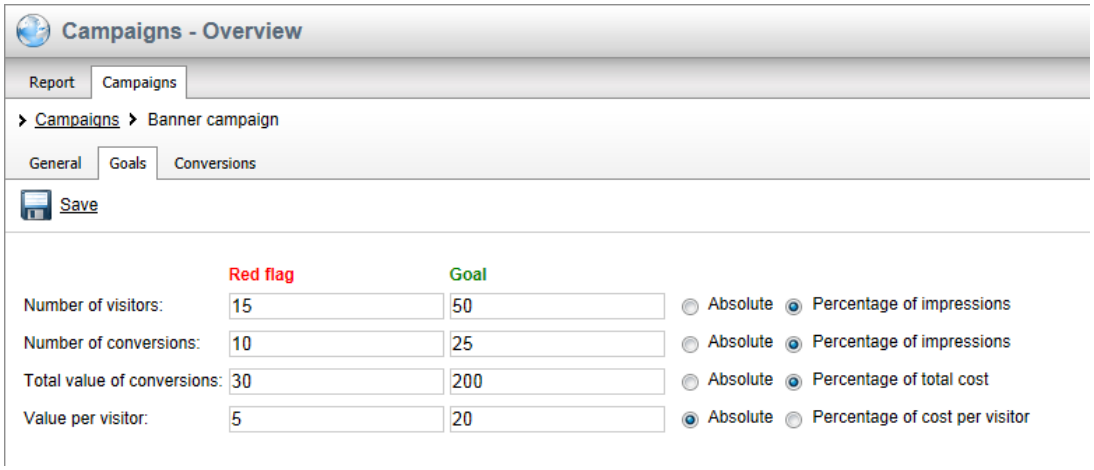
In addition to the data provided by the reports listed above, you can optionally specify goals that should be achieved by each campaign and then compare them with the actual results.

To enter a campaign's goals, select any of the reports, switch to the **Campaigns** tab, and edit () the given campaign. On the **General** tab, the following two properties are available among the advanced settings:

- **Campaign impressions** - this field can be used to specify how many people were targeted by the given marketing campaign. For example, if you sent marketing e-mails containing a link to the website to ten thousand people, the amount of impressions would be *10000*.
- **Total cost** - allows you to manually enter the total cost of the given marketing campaign. This can be used to determine whether the campaign was a success and when calculating the campaign's goals.

Now switch to the **Goals** tab, where you can configure the following types of target values for the campaign:

- **Number of visitors** - sets how many visitors should be brought to the website by the campaign. It can either be specified directly as an *Absolute* number, or as a percentage of the campaign's *Impressions* according to the value set on the *General* tab.
- **Number of conversions** - specifies the expected amount of conversions performed by users who visit the website as a result of the campaign. It can either be specified directly as an *Absolute* number, or as a percentage of the campaign's *Impressions*.
- **Total value of conversions** - sets a target number for the sum of all conversion values logged as a result of the campaign. It can either be specified directly as an *Absolute* number, or as a percentage of the campaign's *Total cost*.
- **Value per visitor** - this indicator allows you to specify the average conversion value that should be generated by a single campaign visitor. It is calculated as the campaign's *Total value of conversions* divided by its *Number of visitors*. The value can either be specified directly as an *Absolute* number, or as a percentage of the campaign's *Cost per visitor* (i.e. the *Total cost* divided by the *Number of Visitors*).

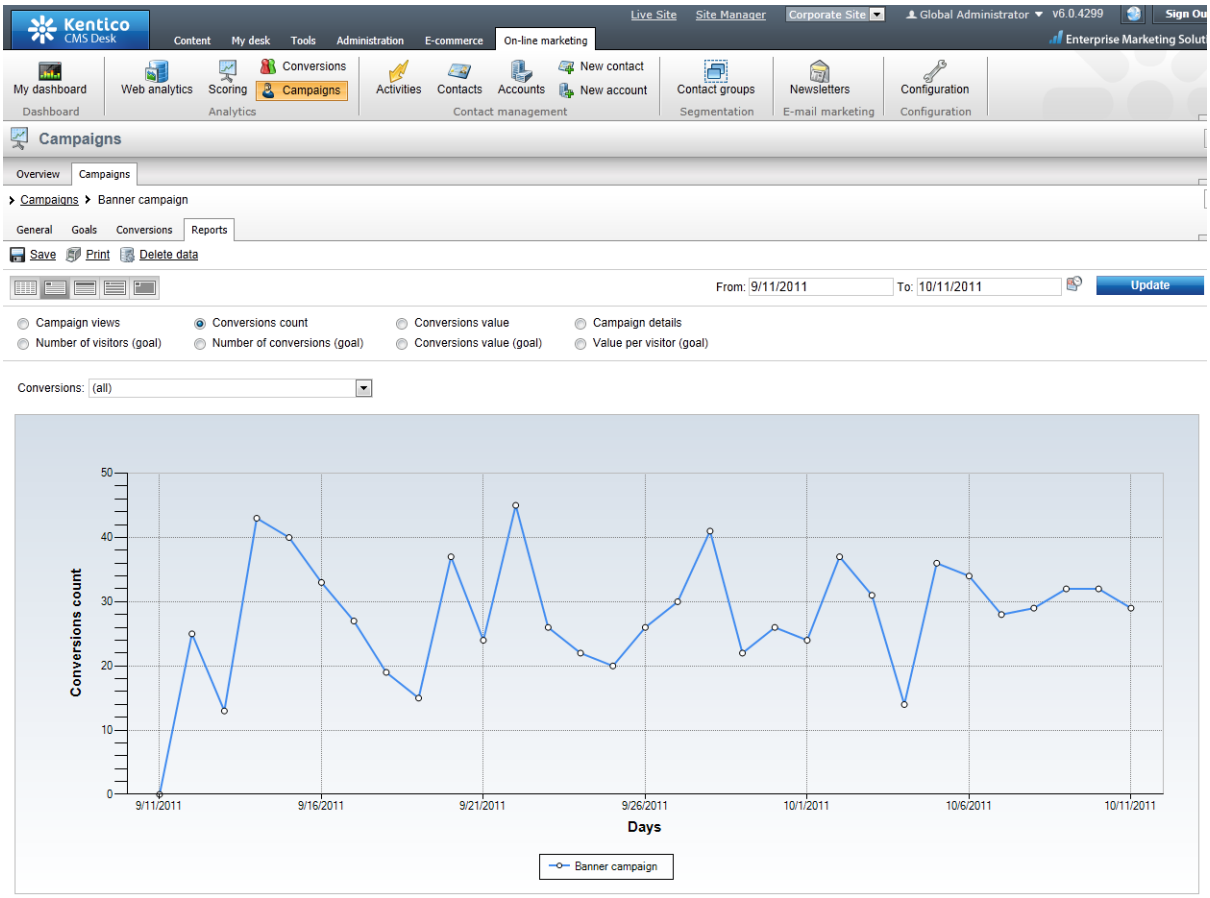


Each goal may have two values. The **Red flag** may be used to set a number that must be reached in order for the campaign to be at least partially successful (campaigns below this value are considered to have failed). The **Goal** value sets a target that should ideally be achieved by the campaign. The progress that individual campaigns make towards their goals can then be followed using the corresponding reports under the **Goals** sub-category in the web analytics tree.

The red flag and goal values configured for the selected statistic are displayed in the reports as red and green lines, so you can easily see the current status of the specified campaign.

Reports for individual campaigns

Any of the reports described above may also be viewed when editing a campaign in **CMS Desk -> On-line marketing -> Campaigns** on the **Reports** tab. You can choose a specific report via the radio buttons at the top of the page.

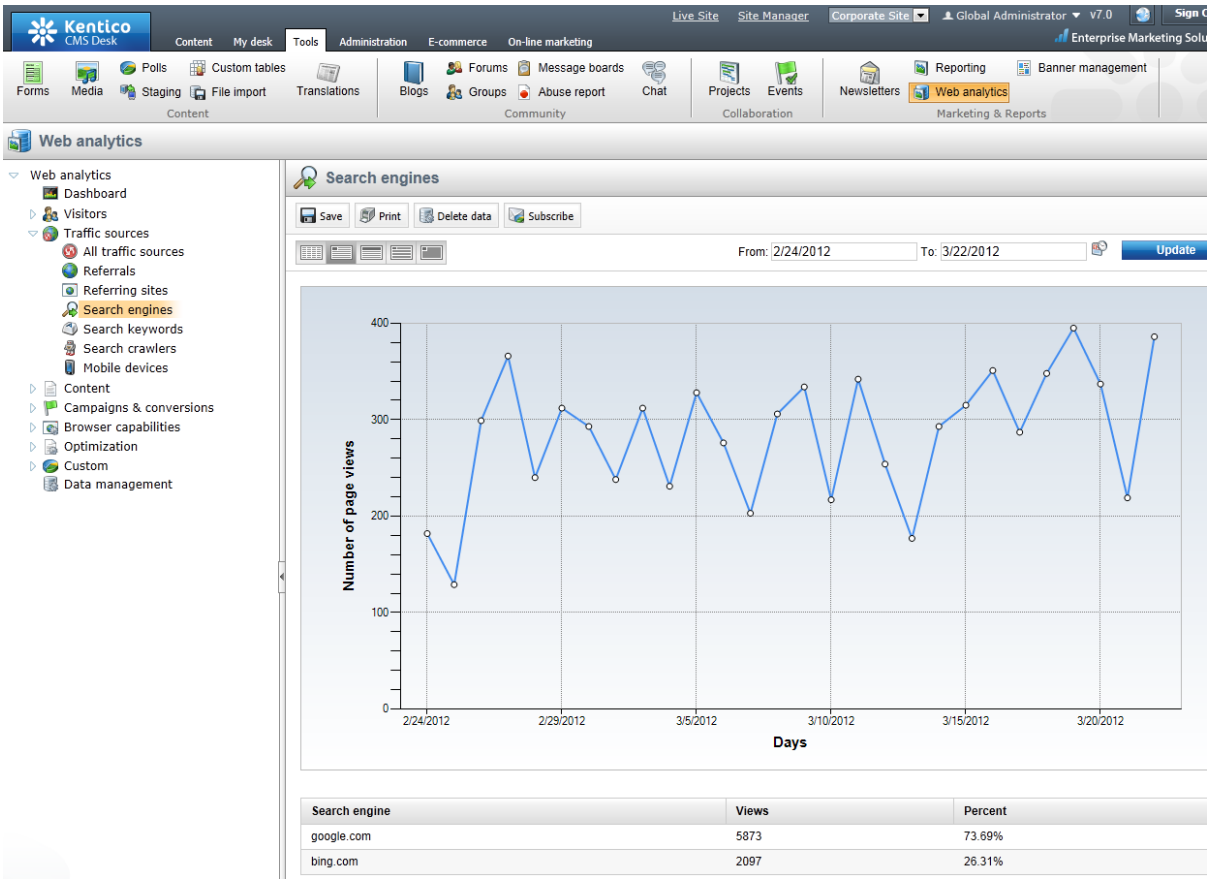


Only the data logged for the given campaign is displayed here.

8.51.6 Monitoring traffic from search engines

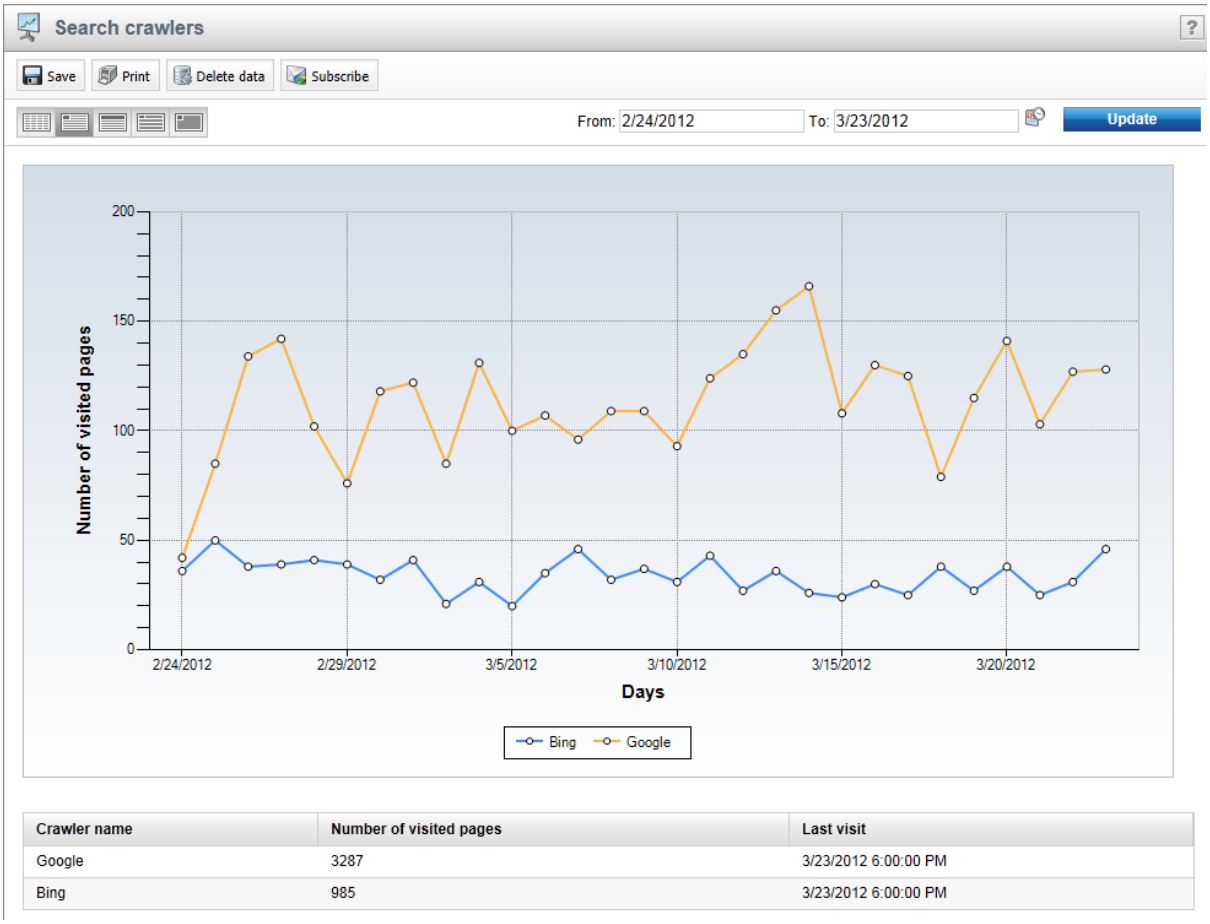
Search engine tracking allows you to monitor the amount of page views received from visitors who found the website using a search engine. Since search engines are the most common type of referring websites, they are tracked separately and with additional details.

The report containing this data can be accessed through the web analytics tree menu in **Traffic sources -> Search engines**. As shown below, it contains a graph displaying the total amount of page views generated by search engine traffic and a breakdown of the statistics for individual search engines.



Additionally, it is possible to view the exact keywords that were entered into the search engines using the **Search keywords** report (also located under the **Traffic sources** category).

Before they can provide search results that link users to your website, search engines need to index the site using web crawlers (robots). The web analytics module is also capable of tracking the activity of these crawlers on your website's pages, and the results can be reviewed in the **Search crawlers** report.



Crawler name	Number of visited pages	Last visit
Google	3287	3/23/2012 6:00:00 PM
Bing	985	3/23/2012 6:00:00 PM

The data provided in this report can be useful when performing Search engine optimization of your website. These statistics can also be checked for specific pages by selecting the corresponding document in the content tree of CMS Desk and viewing its **Traffic sources** reports on the **Analytics -> Reports** tab.

Managing search engines

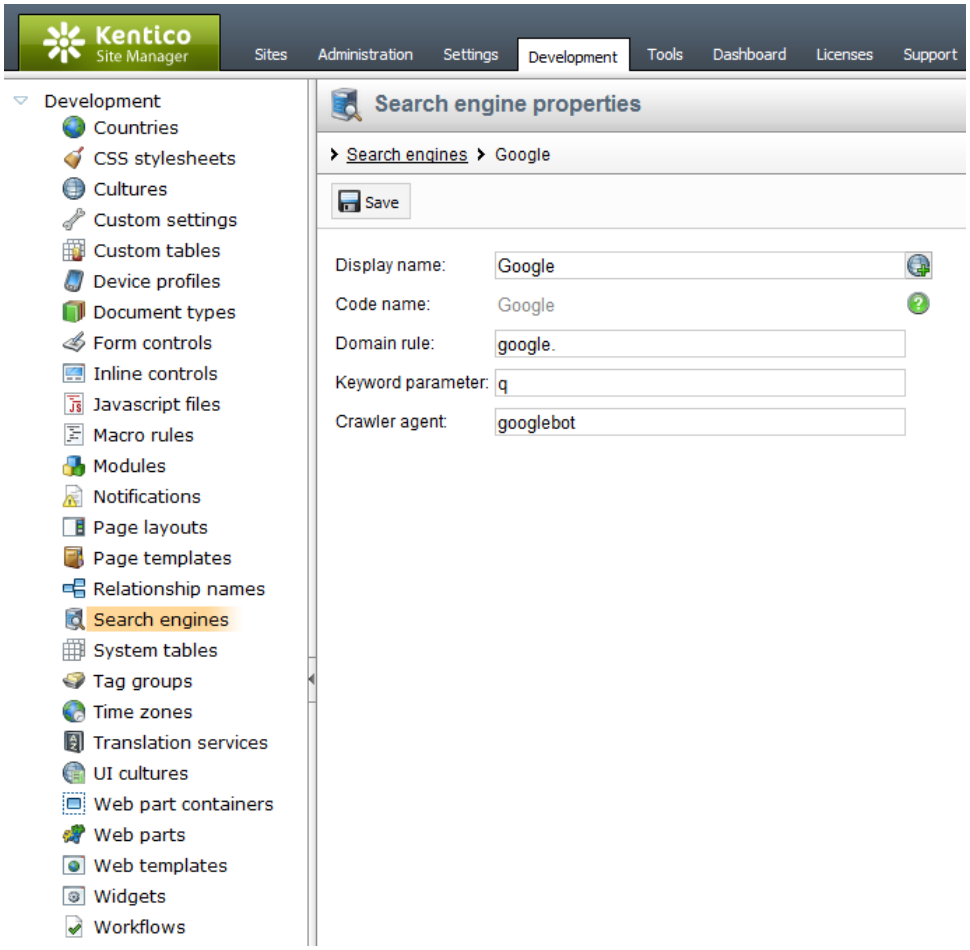
In order to correctly log incoming traffic from search engines, objects representing individual engines must be defined in the system. This can be done in **Site Manager -> Development -> Search engines**. By default, the list contains some of the most commonly used search engines, but any additional ones that you wish to track must be added manually.

When creating or editing ( ) a search engine object, the following properties can be specified:

- **Display name** - sets the name of the search engine used in the administration and web analytics interface.
- **Code name** - may be used to set a unique identifier for the search engine. You can leave the default *(automatic)* option to have the system generate an appropriate code name.
- **Domain rule** - this string is used to identify whether website traffic originates from the given search engine. In order to work correctly, this string must always be present in the engine's URL, for example *google.* for the Google search engine.
- **Keyword parameter** - specifies the name of the query string parameter that the given search engine uses to store the keywords entered when a search request is submitted. This is necessary so that

the system can log which search keywords visitors used to find the website.

- **Crawler agent** - sets the user agent that will be used to identify which web crawlers (robots) belong to this search engine. Examples of common crawler agents are *Googlebot* for Google, or *msnbot* for Bing. This property is optional and it is recommended to specify the crawler agent only for major search engines, or those that are relevant to your website.



If the site is accessed from an external website, the system parses the URL of the page that generated the request. First, the URL is checked for the presence of a valid **Domain rule** that matches the value specified for a search engine object. Then the query string of the URL is searched for the parameter defined in the corresponding **Keyword parameter** property to confirm that the referring link was actually generated by search results, not by a banner or other type of link. This allows the system to accurately track user traffic that is gained from search engine results.

Whenever a page is accessed (indexed) by a web robot with a user agent matching the **Crawler agent** property of one of the search engines registered in the system, a hit is logged in the **Search crawlers** web analytics statistic.

8.51.7 Adding custom web analytics

The system allows you to track custom events as web analytics and display the results in reports. The following example logs a basic button click event, but you can use the same approach to implement tracking for any type of event on your website.

Logging the event

This example logs the event through a user control:

1. Open your Kentico CMS web project in Visual Studio.

2. Create a **New folder** under the root named according to the code name of your site, e.g. *CorporateSite* (if it doesn't already exist).
   - This location ensures that the control will be exported along with your website if it is deployed to a different server.

3. Right-click the folder and select **Add New Item**. Create a **Web User Control** named *AnalyticsButton.ascx*.

4. Add the following code to the control:

```
<asp:Button id="btnLog" runat="server" Text="Add analytics hit"
onclick="btnLog_Click" />
```

This inserts a standard Button control, which will be used to add hits to the custom statistic.

5. Switch to the code behind file of the control and modify it according to the following:

Note: The name of the class will be different according to the code name of your site.

[C#]

```
using CMS.WebAnalytics;
using CMS.CMSHelper;

public partial class CorporateSite_AnalyticsButton : System.Web.UI.UserControl
{
    // Handler for the button's click event
    protected void btnLog_Click(object sender, EventArgs e)
    {
        // Checks if web analytics are enabled in the settings
        if (AnalyticsHelper.AnalyticsEnabled(CMSContext.CurrentSiteName))
        {
            // Adds a hit to a custom statistic
            HitLogProvider.LogHit("buttonclicked", CMSContext.CurrentSiteName,
null, CMSContext.CurrentUser.UserName, 0);
        }
    }
}
```

You can log custom events for web analytics via the API using the **HitLogProvider** class from the **CMS.WebAnalytics** namespace, specifically the following method:

LogHit(string codeName, string siteName, string culture, string objectName, int objectId, int count, double value)

- **codeName** - determines the code name of the custom statistic, *buttonclicked* in the case of this example. This name is also used in the code names of related reports.
- **siteName** - specifies the code name of the site for which the event is logged.
- **culture** - specifies the culture code under which the system logs the event.
- **objectName / objectId** - specifies the context in which the hit was logged. You can use the name or ID of an object. For example, when logging page views, you could store the alias path and document ID of the given page. In the example, the name of the user who clicked the button is logged as the related object.
- **count** - sets the amount of hits added to the statistic. This parameter is optional and the default value (1) is used if not specified.
- **value** - specifies a special value for the hit. This parameter is optional and can be used to assign weight to the hit according to some conditions. The value of a statistic is cumulative, which means that each hit adds to the total value logged for the given analytics record (specified by the **objectName** and **objectId**).

6. **Save** the user control's files.

When a user clicks the button, the API creates an analytics log. Once the system processes the log, the results containing the given user's name and the amount of clicks are stored in the database, where they can be used by the web analytics module.

Note *[Only applies after installing hotfix 7.0.17 or newer]*

Custom analytics do not automatically use JavaScript logging. If your website has JavaScript logging enabled, you may need to leverage JavaScript in the code to keep your custom statistics consistent with the default web analytics.

Integrating custom analytics into the website

Now you need to publish the user control on your website (this example uses the sample Corporate site) .

1. Open the Corporate site in CMS Desk in **Content -> Edit** mode.

2. Select the **Home** document from the content tree and switch to the **Design** tab.

3. Add (✚) the **User control** web part to the *Main zone* zone.

4. Enter *~/CorporateSite/AnalyticsButton.ascx* into the web part's **User control virtual path** property.
   - The page now displays the custom button.

5. View the live site version of the **Home** page and click the **Add analytics log** button several times to add hits to the custom statistic. Try doing this while logged in under different user accounts.

There are also other ways to add custom web analytics functionality to the pages of your website. For example:

- You can implement the control as a custom web part, add it to your pages and configure it through

the portal engine as necessary.
- If you are using the ASPX page template development model for your website, you can integrate the analytics logging code and any required interface elements directly into your page templates.

Using the Analytics custom statistics web part

If you wish to log a simple event when visitors access a specific page, you can do so without having to write any code using the Analytics custom statistics web part.

Once this web part is placed on a page, it automatically adds a hit to a custom statistic when the given page is viewed. You can specify the details of the statistic, such as its code name, the name of the related object or the hit value using the web part's properties. The web part automatically loads the site name and culture from the context of the page.

Creating reports for statistics

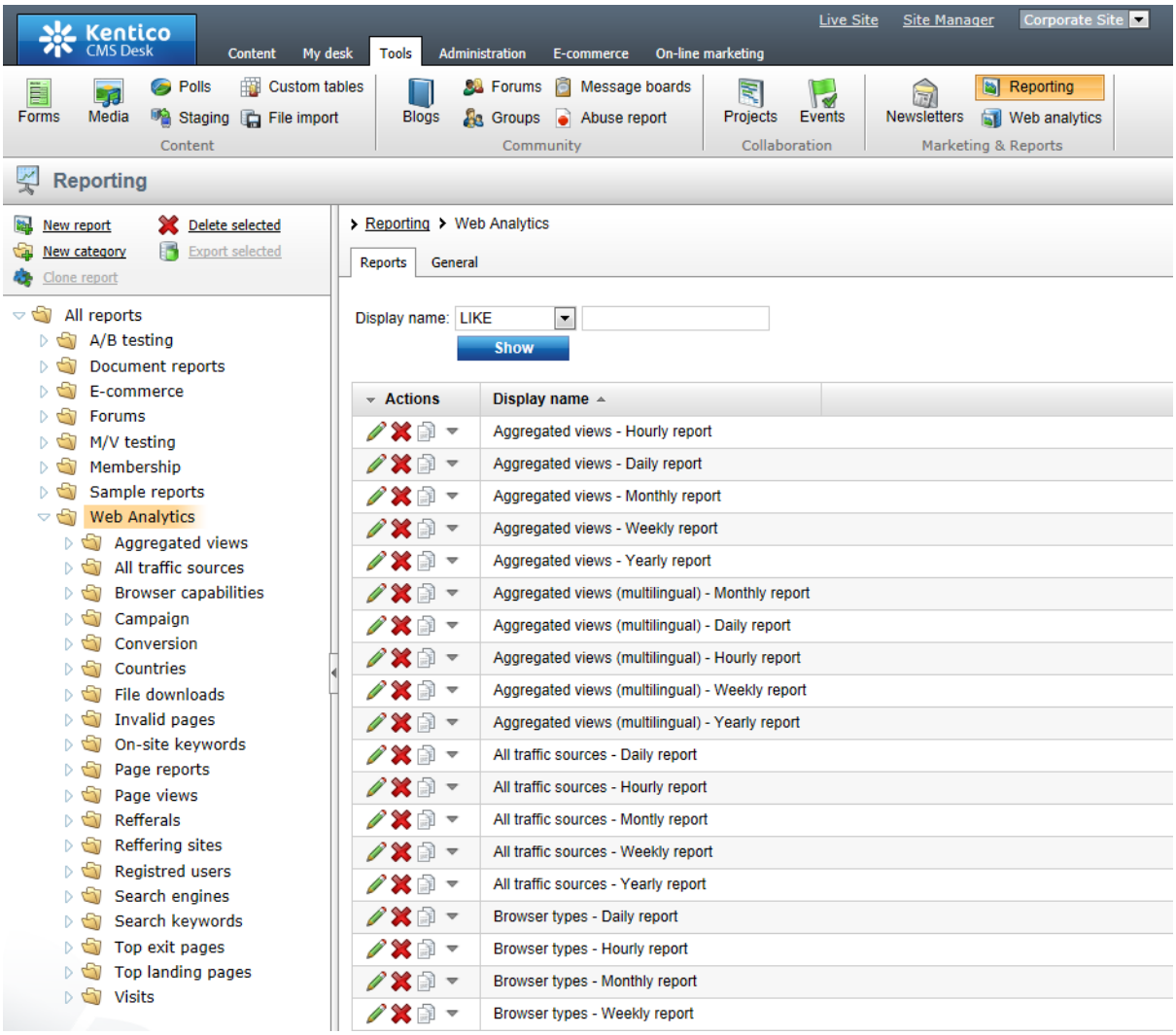
Before users can view the statistics of the button click event in **CMS Desk -> Tools -> Web Analytics**, you need to create reports for displaying the data of the custom statistic.

1. Go to **CMS Desk -> Tools -> Reporting** and expand the **Web Analytics** category in the tree on the left. The sub-categories contain reports used by the default statistics such as page views, traffic sources, campaigns, etc. Most statistics have five types of reports: hourly, daily, weekly, monthly and yearly.

The code names of the reports have to use a specific format:

- <statistic code name>.hourreport
- <statistic code name>.dayreport
- <statistic code name>.weekreport
- <statistic code name>.monthreport
- <statistic code name>.yearreport

In our example, the code name of the custom statistic is **buttonclicked**, as defined above in the code of the user control.



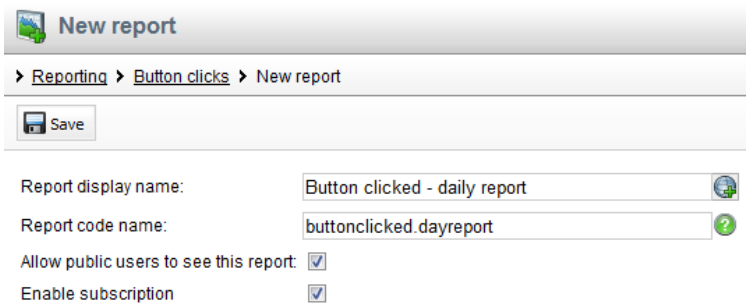
2. Add a **New category** under **Web Analytics** and enter the following names for it:

- **Category display name**: Button clicks
- **Category code name**: ButtonClicks

Click **Save**.

3. Now we will create a daily report for the new statistic. Click **New report** and enter the following values:

- **Report display name**: Button clicked - daily report
- **Report code name**: buttonclicked.dayreport



Click **Save**.

4. Switch to the **Parameters** tab of the new report and use the **New attribute** (+) button to create three essential parameters which will be used internally by the web analytics module. Click **Save field** for every parameter before moving on to the next one:

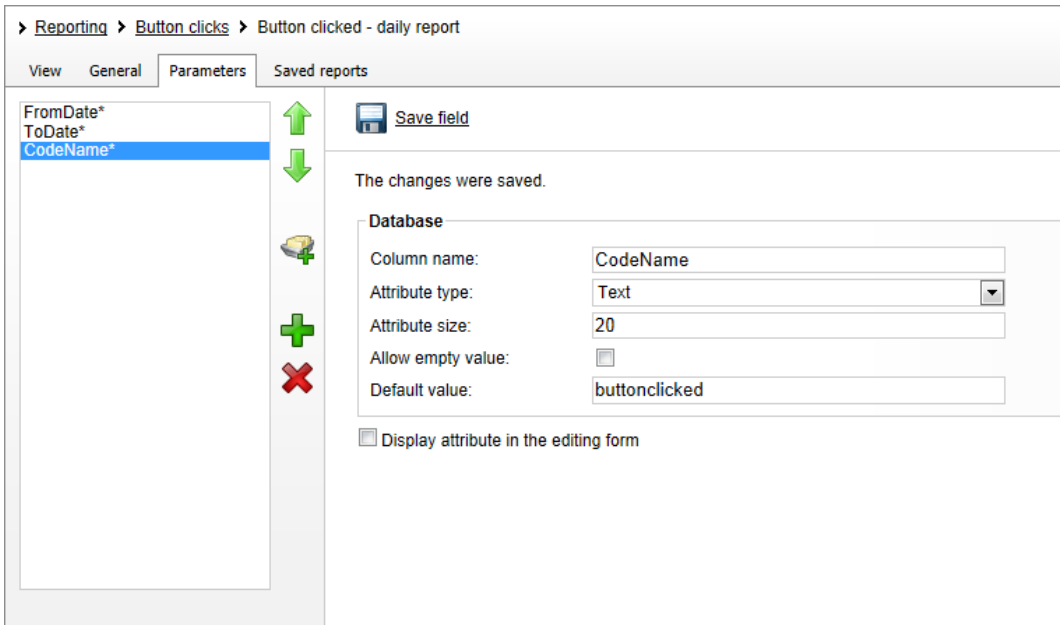
- **Column name:** FromDate
- **Attribute type**: Date and Time
- **Display attribute in the editing form**: disabled

- **Column name**: ToDate
- **Attribute type**: Date and Time
- **Display attribute in the editing form**: disabled

The two parameters above define a time interval used to limit which data of the statistic should be loaded. Only hits that occurred during the specified interval will be displayed in the report. The values of these parameters can be set by users when viewing the data of the statistic in the web analytics interface.

- **Column name**: CodeName
- **Attribute type**: Text
- **Attribute size**: 20
- **Default value**: buttonclicked (the code name of the displayed statistic in general)
- **Display attribute in the editing form**: disabled

This parameter is simply used to identify from which statistic the data should be loaded. Its value is usually not modified if the report is dedicated to a single statistic (like the one in this example).



5. Go to the **General** tab and click the **New** button in the **Tables** section below the layout editor to create a report table. Fill in its properties as shown below:

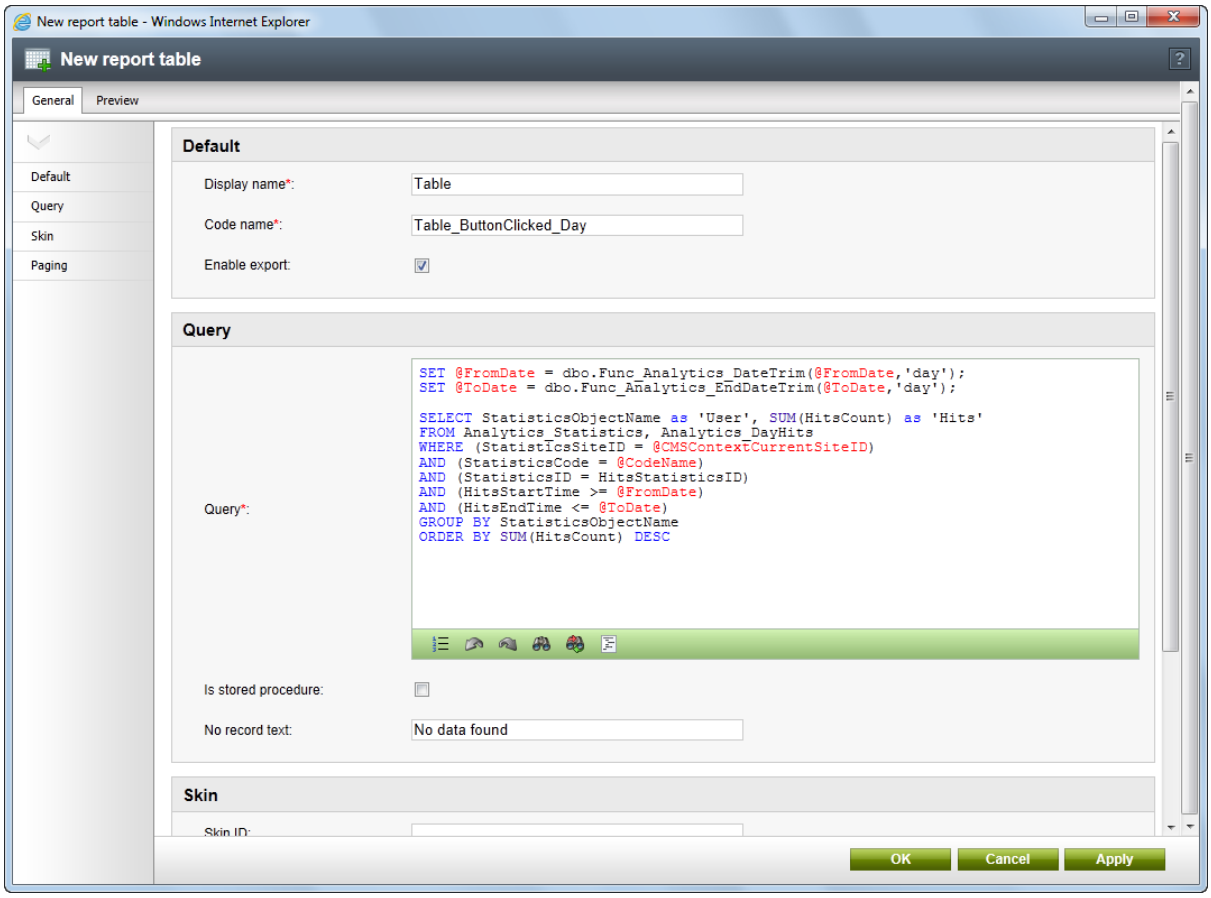
- **Display name**: Table
- **Code name**: Table_ButtonClicked_Day
- **Enable export**: true (checked)
- **Query**:

```
SET @FromDate = dbo.Func_Analytics_DateTrim(@FromDate,'day');
SET @ToDate = dbo.Func_Analytics_EndDateTrim(@ToDate,'day');

SELECT StatisticsObjectName as 'User', SUM(HitsCount) as 'Hits'
FROM Analytics_Statistics, Analytics_DayHits
WHERE (StatisticsSiteID = @CMSContextCurrentSiteID)
AND (StatisticsCode = @CodeName)
AND (StatisticsID = HitsStatisticsID)
AND (HitsStartTime >= @FromDate)
AND (HitsEndTime <= @ToDate)
GROUP BY StatisticsObjectName
ORDER BY SUM(HitsCount) DESC
```

Trimming the values of the FromDate and ToDate parameters

The SET statements included in the query are necessary to ensure that the specified time interval includes the first and last units of time (e.g. the exact days entered entered into the **From** and **To** fields of a daily report).

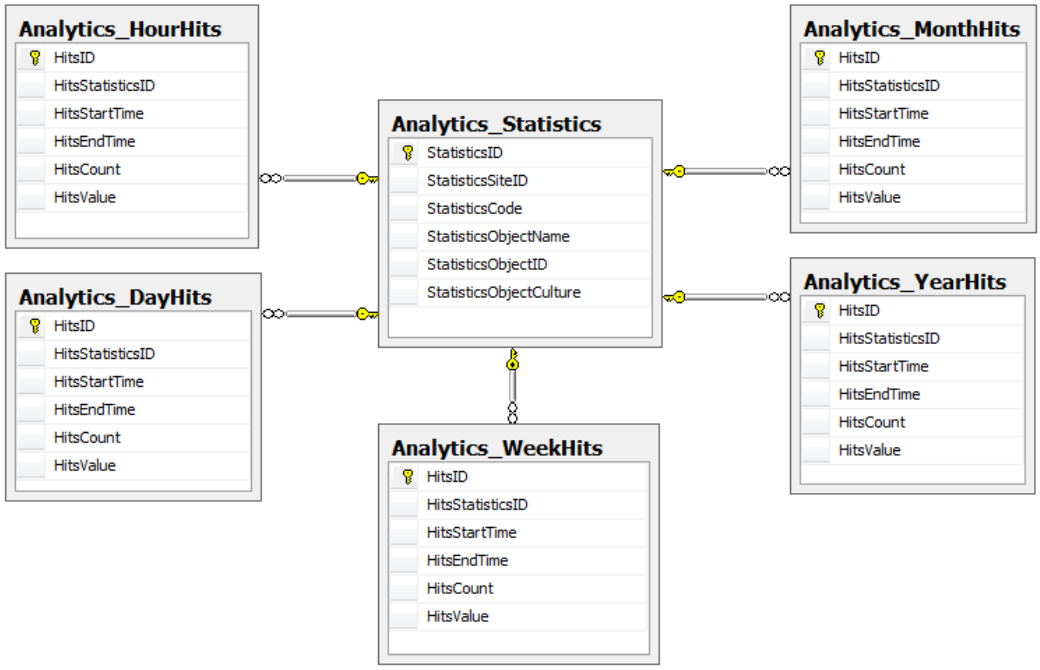
The second parameter of the trimming functions must match the time unit of the given report. The options are: *'hour'*, *'day'*, *'week'*, *'month'*, *'year'*.



There are six important database tables used by the web analytics module to keep track of statistics and their values. The *Analytics_Statistics* table stores records that represent the statistics of a tracked event within a certain context, i.e. related to a specific object, site and culture.

Five other tables are used to store the exact number of hits for the statistics in the *Analytics_Statistics* table (*Analytics_HourHits*, *Analytics_DayHits*, *Analytics_WeekHits*, *Analytics_MonthHits* and *Analytics_YearHits*). When a hit for a tracked statistic occurs, it is logged into all of these tables. The difference between them is in the unit of time used to separate hits into individual records. For example, a record in the *Analytics_HoursHits* table would contain the number of hits that were logged for a given statistic during one hour, while a single record in *Analytics_MonthHits* would count all hits that occurred over an entire month.

The diagram below represents the basic database structure used to store statistics and their hits:



Since the new report is a daily report, the data of the table is retrieved from the *Analytics_DayHits* table, together with the *Analytics_Statistics* table.

Click **OK** to create the table.

6. To complete the daily report, use the **Insert** button in the **Tables** section to add a macro representing the table into the layout of the report and click **Save**.

In a real-world scenario, the report would usually also contain a graph to display the data. Please refer to the Modules -> Reporting chapter to learn more about graphs and other reporting options.

You can create the reports for the remaining time intervals using the same approach. The only differences should be in the names of the reports and tables, and the code of the queries, where you need to load data from the appropriate tables — *Analytics_YearHits* in the yearly report, *Analytics_MonthHits* in the monthly report, etc. You can make this process easier by using the **Clone report** action from the menu above the report tree. Simply create copies of the daily report and then make the necessary adjustments.

Setting the user interface title for custom statistics

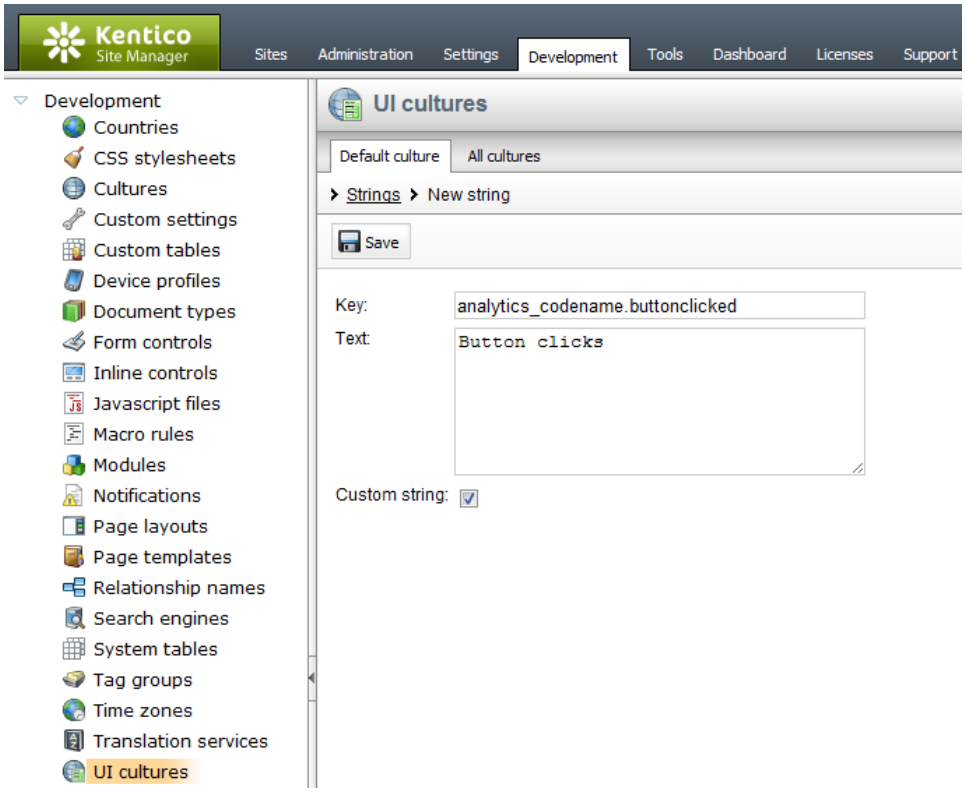
To ensure that the user interface displays an appropriate title for your custom statistic, you need to create a resource string:

1. Go to **Site Manager -> Development -> UI cultures**.

2. Click **New string** and enter the values below:

- **Key**: *analytics_codename.buttonclicked*; in general, the key of the string must always be in format *analytics_codename.<statistic code name>*.
- **Text**: *Button clicks*; you may enter any required text.

  - If you are using a multilingual UI, you can enter translations of the string for specific cultures via the *All cultures* tab.
- **Custom string**: yes (checked)



3. Click **Save**.

The system uses the string as the title for the matching statistic.

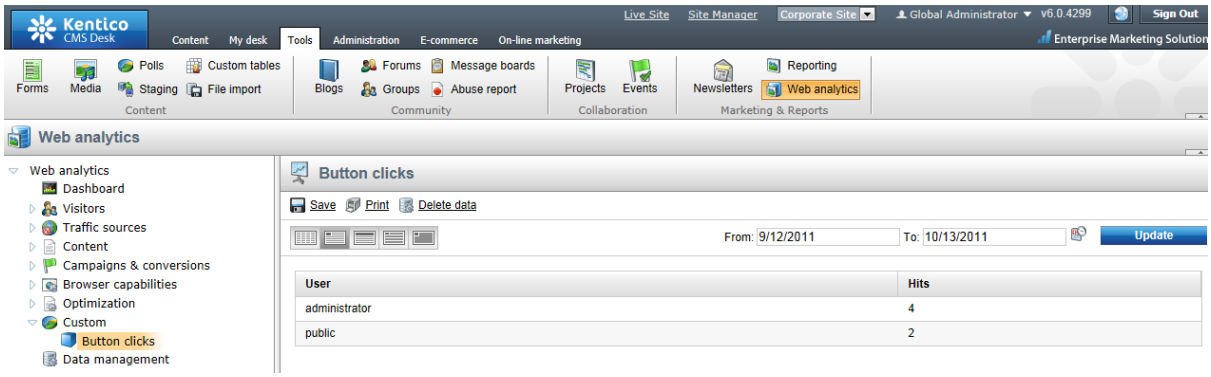
Adding icons for custom statistics

You can add a custom icon for your new statistic:

1. Open the *\App_Themes\Default\Images* folder in your Kentico CMS project's directory.
2. Extract the **CMSModules** folder from the **Images.zip** archive and place it into the *~/App_Themes/Default/Images/* directory.
3. Add your icon image file into the *CMSModules\CMS_WebAnalytics* folder.
   - Name the image file in format **<statistic code name>.<extension>** (for example *buttonclicked.png* in this scenario).
   - If there are any period characters ('.') in the code name, you can replace them with an underscore ('_') to create a valid file name.

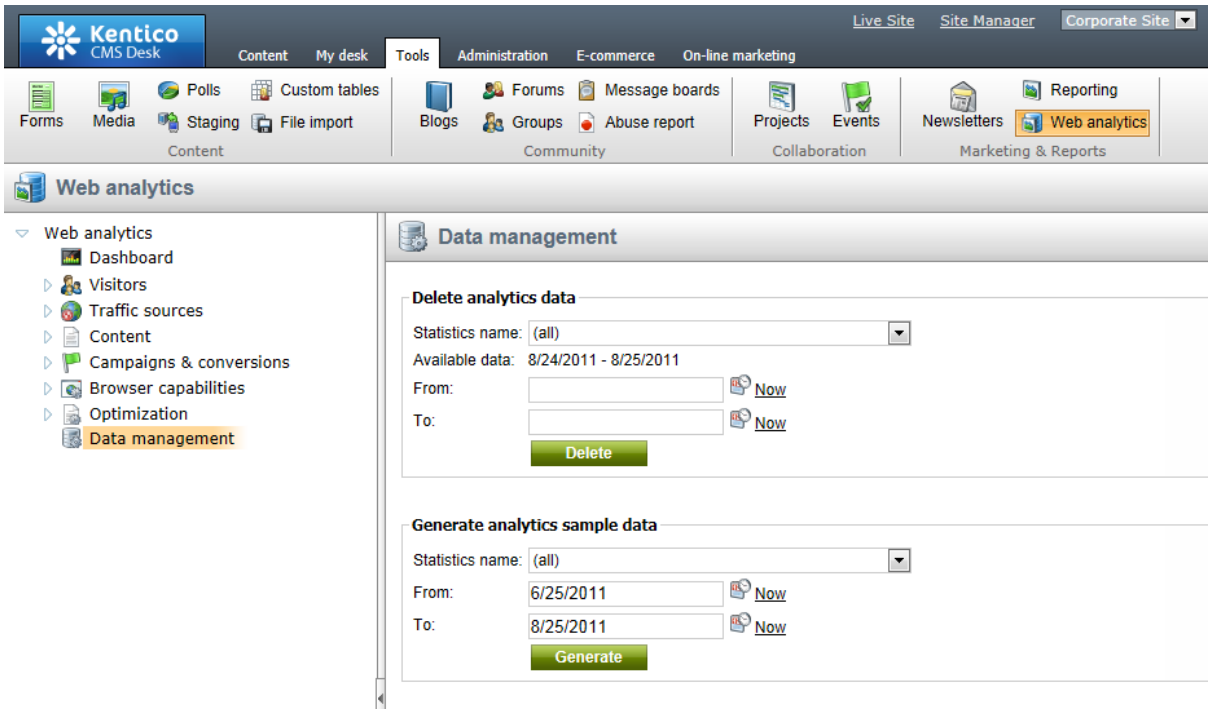
Result

Once there are some hits logged in the database for a custom statistic, the system automatically displays the statistic under the **Custom** category of the tree in **CMS Desk -> Tools -> Web Analytics**. The reports of the sample statistic show the number of hits logged for the tracked button click event.



8.51.8 Managing analytics data

It is possible to manually manage the data used to store how many hits are logged for individual web analytics statistics. To do this, access the **CMS Desk -> Tools -> Web analytics** interface and select the **Data management** item from the tree menu. By default, this UI element is only available for global administrators.



Data security

Performing the actions available in this section will permanently modify the analytics records of the website. Only use them if you are sure that doing so will not delete or overwrite valuable statistical data.

For security reasons, the actions can only be performed by users who have the **Manage data** permission for the Web analytics module.

Delete analytics data

When running a site with web analytics enabled, the amount of data that is stored may grow quite large over time, particularly when tracking many types of events or on high-traffic websites. The **Delete analytics data** action can be used to remove hits that were logged for the selected statistic during the time period specified in the **From** and **To** fields. You can either choose a specific statistic or clear all analytics data for the given time interval by selecting the *(all)* option from the **Statistics name** drop-down list.

Generate analytics sample data

This action logs randomly generated sample data for the selected statistics, which can be used to simulate hits measured over the given time period. The specific statistic and time interval may be configured the same way as when deleting data.

The purpose of this function is to allow testing or evaluation of the web analytics module and the various types of statistics that it provides, since a certain amount of time and traffic would otherwise be required to naturally log a realistic amount of data for the website.

Important!

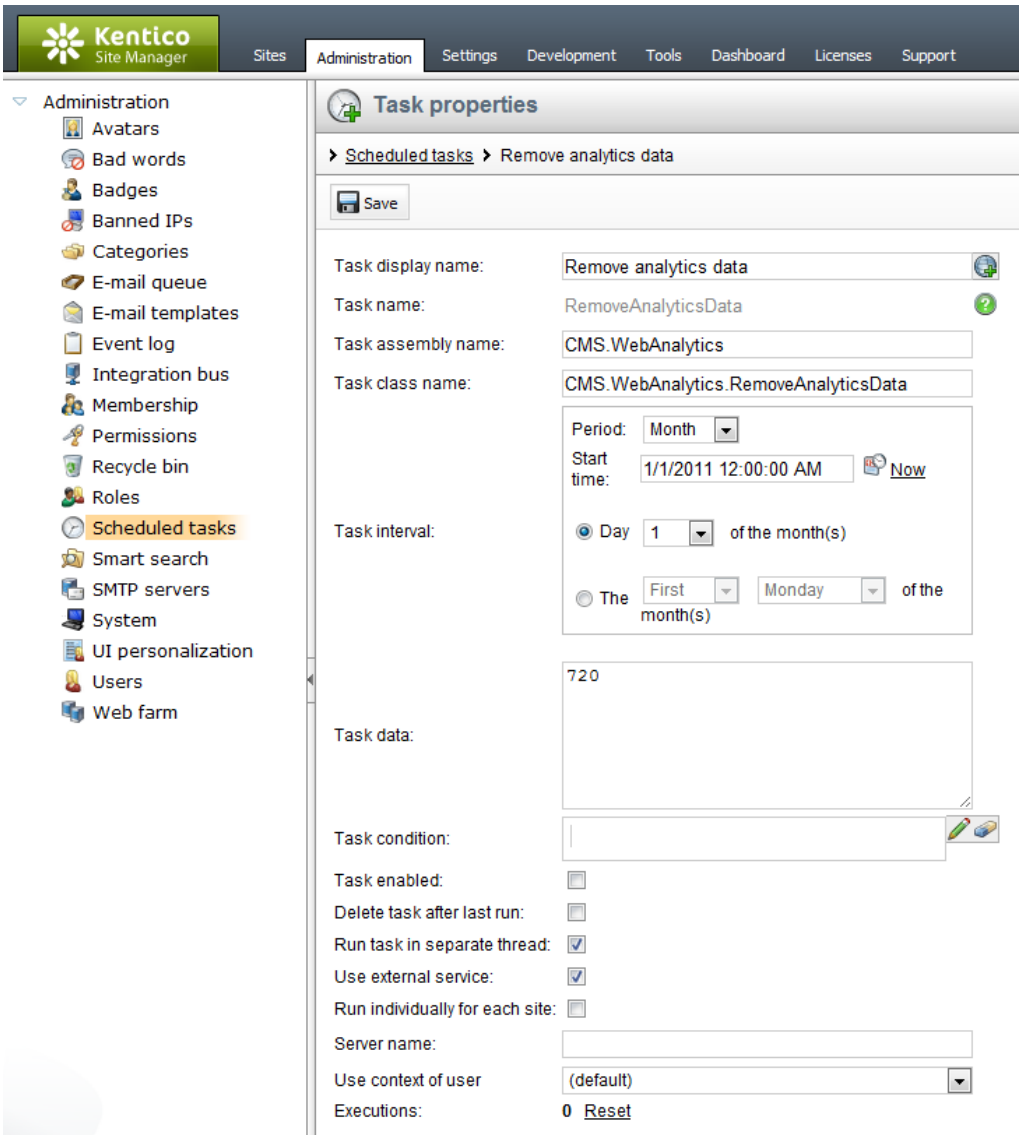
The data generation process is a highly resource-intensive operation, since it must create a very large amount of records in multiple database tables.

For this reason, it is necessary to set a reasonably short interval to prevent time out issues. This depends on the performance of the server used to host the site, but it is recommended to avoid generating more than six months of data in a single operation.

Automatic data cleanup

It is also possible to configure the system to automatically delete old web analytics data that is no longer needed. There is a global scheduled task available for this purpose called **Remove analytics data**. By default this task is disabled to prevent unwanted deletion of data.

To enable automatic data removal for web analytics, go to **Site Manager -> Administration -> Scheduled tasks**, edit () the mentioned task and check its **Task enabled** field.



You can configure how often the task should be executed using the **Task interval** settings. To specify how old data must be before the task is allowed to remove it, enter a number into the **Task data** field. This number indicates age in days, so the default value of 720 sets the task to delete only data that is approximately two years old. It is recommended to configure this task carefully to avoid losing valuable data.

8.51.9 Configuration options

You can configure the functionality of the web analytics module in **Site Manager -> Settings -> On-line marketing -> Web analytics**. The settings here allow you to enable or disable tracking of various types of events and specify other related configuration options.

Note: The system checks the values of the settings at the moment that tracked events occur. Changes do not affect previously logged data, only future events.

General

Enable web analytics	Enables or disables the entire web analytics module. If this setting is disabled, the system does not track any of the statistics listed below.
Log via JavaScript snippet	By default, the system logs analytics directly on each request. If you enable this setting, the analytics are logged via JavaScript. JavaScript logging ignores all browsers and devices that do not support JavaScript or have it disabled. This filters out non-human tools such as RSS readers and web crawlers from future analytics statistics. Switching to JavaScript logging does not affect the performance of the website.
Campaigns & Conversions	
Track campaigns	Indicates if campaign tracking should be enabled.
Campaign tracking URL parameter	Sets the name of the URL query string parameter which is used to indicate that the site was accessed as a result of a campaign. The value of the parameter stores the code name of the given campaign.
Track conversions	Indicates if tracking of conversions should be enabled. This includes both general conversions and those logged within a certain special context, such as for A/B or Multivariate tests, or as a result of a Campaign.
Views & Downloads	
Track file downloads	If enabled, the system tracks which files were downloaded by website visitors. Please note that only files stored as documents in the website's content tree are tracked.
Track invalid pages	Indicates if invalid page requests should be tracked. Invalid requests are those that contain the website's domain name, but specify a path to a page that does not exist.
Track page views	Indicates if the system should track how many times the website's pages were viewed by visitors. Only pages that are served by Kentico CMS are included in the statistics.
Track aggregated views	If enabled, access to pages via links contained in RSS or Atom feeds (created using the Syndication module) are tracked.
Track landing pages	Indicates if the system should track which pages are the first ones viewed by visitors when they start their browsing session on the website.
Track exit pages	If enabled, the system logs the final pages that were visited by users when their browsing session ended.
Track average time on page	If enabled, the average time that users spend on pages is tracked.
Visitors	
Track browser types	Indicates if the browser types and versions used by the website's visitors is tracked.
Track visits	Indicates if site visits should be tracked. A single visit includes any

	number of page views or other actions performed by a specific user during one session. First time users are logged as new visits. After a specified time period of inactivity, known users are logged as returning visits. Visitors are recognized using a browser cookie.
Visitor idle time (minutes)	Determines how long (in minutes) a visitor must be inactive before their presence on the website is logged as a returning visit. The default value is *1380* (23 hours). You can adjust this setting depending on how often visitors usually return to your site. For example, to track how frequently your site's visitors return over the course of a single day, you could set the value to *60* minutes or less.
Remember visitors by IP (minutes)	If the entered value is higher than *0*, the IP addresses of the website's visitors are stored in the memory for the specified number of minutes. You may use this setting in cases where there are problems identifying visitors using the standard approach, e.g. if many of your visitors have cookies disabled in their browser.
Track countries	If enabled, the countries from which visitors access the website is tracked. The countries are recognized according to the IP addresses of visitors, which may not be 100% reliable in all cases, but the overall statistics for a high number of visits should provide correct results.
Track registered users	Enables tracking of user registrations.
Track search crawlers	Indicates whether the activity of search engine web crawlers (robots) should be monitored. This type of tracking is performed even if the **Exclude search engines** setting is enabled.
Track mobile devices	If enabled, the system tracks whether visitors access the website using mobile devices.
Excluded	
Exclude search engines	If enabled, hits generated by search engine robots (crawlers) are not included in tracking statistics. This does not affect the *Search crawlers* statistic.
Excluded file extensions	Sets which file types should not be tracked as part of the **File downloads** statistics. The file types are specified using a list of extensions separated by semicolons (;), for example: *.jpg;.gif* Please note that it's necessary to include the period in the extension name.
Excluded URLs	Can be used to exclude websites or their sections from all types of web analytics tracking. To exclude a page and all underlying pages, enter its URL (or document alias). You can specify multiple URLs (or site sections) separated by a semicolon (;).
Excluded IP addresses	Hits generated by the client IP addresses listed here are not included in

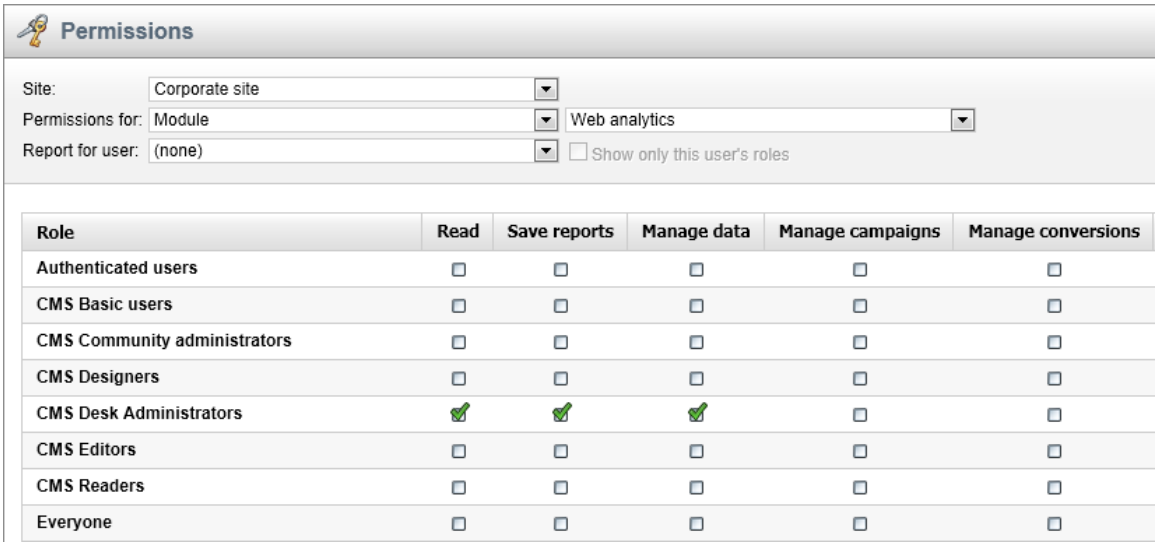
	web analytics tracking. If multiple addresses are entered, they must be separated by semicolons (;). The asterisk (*) wildcard can be used as a substitute for any number in an IP address, substituting for all values from 0 to 255. This can be used to exclude the IP addresses of the website's administrators or other special users so that they do not influence tracking results.
Traffic sources	
Track search engines	Indicates if traffic from search engines should be tracked.
Track search keywords	Indicates if the keywords that were entered into search engines in order to find the website should be logged.
Track on-site keywords	Indicates if the keywords that were used in the website's local search functionality should be logged.
Track referring local pages	If enabled, the system tracks which links visitors use to navigate within the website (i.e. menus or other types of links present on the site's pages).
Track referring pages by direct link	If enabled, the system tracks how many times website's pages are accessed directly through a URL entered into the browser.
Track referring sites	Indicates if the system should log the total amount of page views gained through links from external websites and the statistics for individual website domains.
Track referrals	Indicates if the system should log the full URLs of external pages from which visitors were linked to the website and the number of resulting page views.

8.51.10 Security

You can configure security options for the various types of actions provided by the web analytics module in **Site Manager/CMS Desk -> Administration -> Permissions**. Choose to display the *Module* permission type and then select *Web analytics*.

The following permissions can be given to the listed roles:

- **Read** - allows members of the specified roles to view web analytics reports and other parts of the *CMS Desk -> Tools -> Web analytics* interface. It is also required to access the analytics reports anywhere else in the UI, e.g. in *CMS Desk -> Content -> Analytics -> Reports*.
- **Save reports** - allows members of the specified roles to save web analytics reports. The saved reports can be viewed using the *CMS Desk -> Tools -> Reporting* interface.
- **Manage data** - allows members of the specified roles to manage the data logged for various statistics (i.e. delete or generate sample data for statistics).
- **Manage campaigns** - allows members of the specified roles to create or delete campaign tracking objects and edit their properties, including goals.
- **Manage conversions** - allows members of the specified roles to create or delete conversions and edit their properties.



8.51.11 Web analytics internals and API

8.51.11.1 Overview

In this chapter, you will learn which database tables and API classes are used in the Web analytics module.

Advanced API examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all methods from the module classes, please refer to Kentico CMS API Reference.

8.51.11.2 Database tables

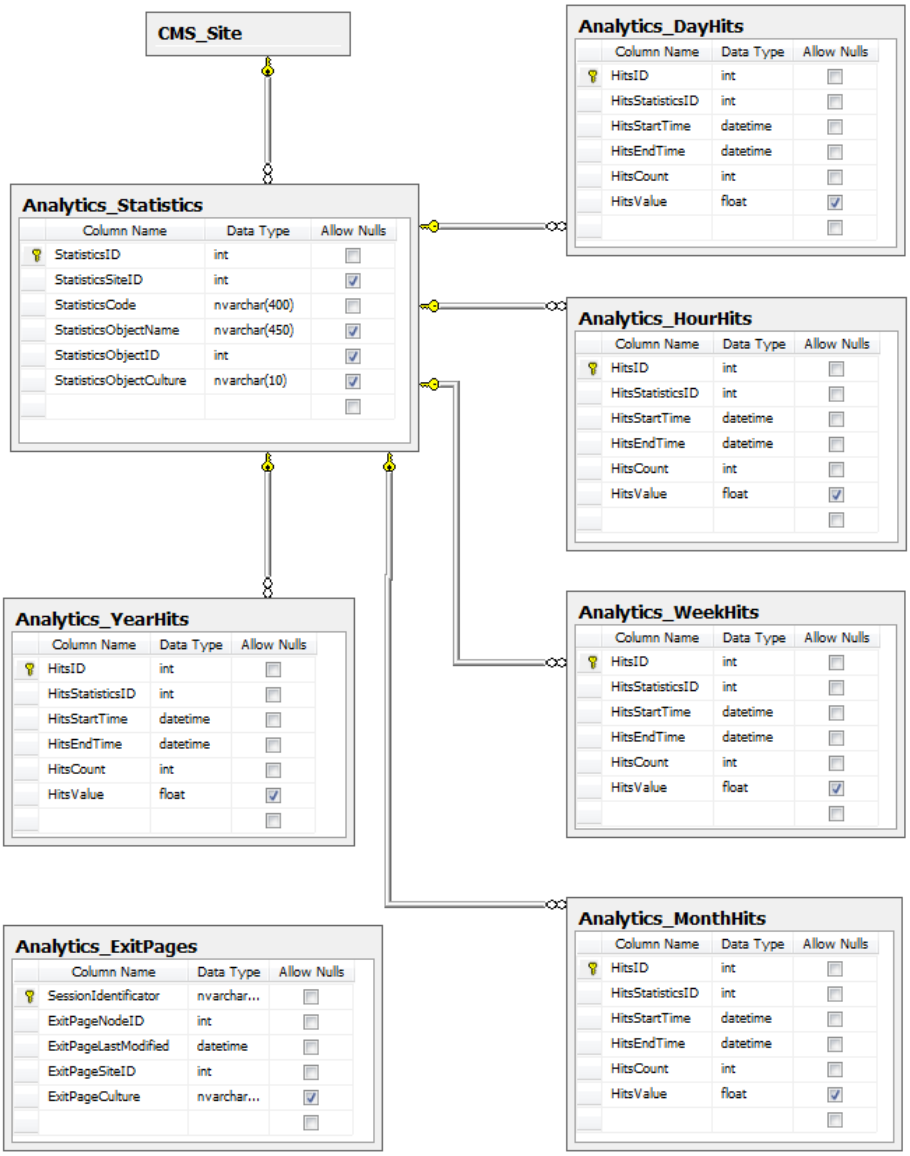
There are six important database tables used by the web analytics module to keep track of statistics and their values. The **Analytics_Statistics** table stores records that represent the statistics of a tracked event within a certain context, i.e. related to a specific object, site and culture.

Five other tables are used to store the exact number of hits for the statistics in the Analytics_Statistics table:

- **Analytics_HourHits**
- **Analytics_DayHits**
- **Analytics_WeekHits**
- **Analytics_MonthHits**
- **Analytics_YearHits**

When a hit for a tracked statistic occurs, it is logged into all of these tables. The difference between them is in the unit of time used to separate hits into individual records. For example, a record in the **Analytics_HoursHits** table would contain the number of hits that were logged for a given statistic during one hour, while a single record in **Analytics_MonthHits** would count all hits that occurred over an entire month.

The **Analytics_ExitPages** table is used to temporarily store exit page candidates for the *Top exit pages* statistic. The latest candidate recorded for a visitor when their session expires is then stored as the final exit page.



Additionally, the tables listed below are used to store campaign and conversion tracking objects:

- **Analytics_Campaign** - contains records representing campaigns and their settings.
- **Analytics_Conversion** - contains records representing conversions.
- **Analytics_ConversionCampaign** - stores relationships between campaigns and conversions. Each entry in this table indicates that a conversion should be included in the statistics of a certain campaign. Only campaigns that are restricted to a limited set of conversions have these relationships.

Campaign and conversion statistic storage

The data logged for campaigns and conversions is stored like all other web analytics

statistics. The main records are kept in the **Analytics_Statistics** table and the corresponding amount of hits for individual units of time are saved in the appropriate **Hits** table.

Page view statistics for campaigns always use the *campaign* **StatisticsCode** with the name of the given campaign stored in the **StatisticsObjectName** column.

The following code names are used for conversion statistics:

- **conversion** - general statistic used to store the overall conversion records. This statistic is always logged when a conversion is performed on the website.
- **campconversion;<campaign name>** - logged when a conversion is performed by a user associated with the particular campaign.
- **abconversion;<A/B test code name>;<Variant code name>** - logged when a conversion is performed by a user who viewed the given page variant of an A/B test.
- **mvtconversion;<MVT test code name>;<Combination name>** - logged when a conversion is performed by a user who viewed the given content combination on a page with a defined multivariate test.

All types of conversions use the **StatisticsObjectName** column to store the code name of the logged conversion.

8.51.11.3 API classes

If you are not familiar with the database table data management by Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

The classes of the web analytics module can be found in the **CMS.WebAnalytics** namespace.

Analytics_Statistics table API:

- **StatisticsInfo** - represents the statistics of a certain event within a specific context.
- **StatisticsInfoProvider** - provides management functionality for statistic records.

Analytics_<time interval>Hits table API:

- **HitsInfo** - represents the hits of a statistic during a specific time interval.
- **HitsInfoProvider** - provides management functionality for statistic hits.

Analytics_Campaign table API:

- **CampaignInfo** - represents one campaign tracking object.
- **CampaignInfoProvider** - provides management functionality for campaigns.

Analytics_Conversion table API:

- **ConversionInfo** - represents one conversion tracking object.
- **ConversionInfoProvider** - provides management functionality for conversions.

Analytics_ConversionCampaign table API:

- **ConversionCampaignInfo** - represents a relationship between a campaign and a conversion.
- **ConversionCampaignInfoProvider** - provides management functionality for campaign-conversion relationships.

Other classes:

- **AnalyticsHelper** - provides general web analytics functionality and data.
- **HitLogProvider** - contains methods used to create the analytics log files for statistics.
- **HitLogProcessor** - this class implements the scheduled task used to periodically process the analytics log, which transfers the information to the database.

8.52 WebDAV integration

8.52.1 Overview

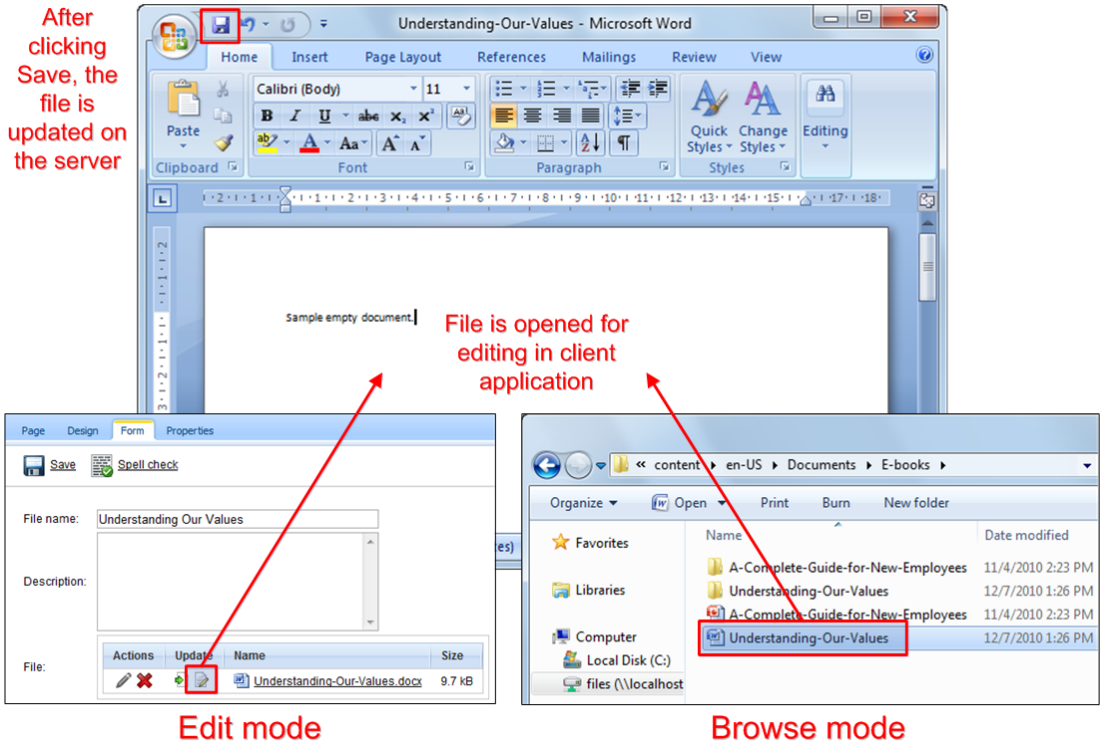
WebDAV stands for *Web-based Distributed Authoring and Versioning*. It is a set of extensions to the HTTP protocol which allows users to collaboratively edit and manage files on remote web servers. Detailed information, specifications and documentation related to this technology can be found on its official website: http://www.webdav.org/. Other valuable information is located on its Wikipedia page: http://en.wikipedia.org/wiki/WebDAV.

WebDAV integration in Kentico CMS enables users to open, edit and save files stored in the CMS in a client application installed on their local computer (e.g. *Microsoft Office Word*, *Excel*, *PowerPoint*, *MS Paint*, etc.). The advantage of this approach is that no local copy of the edited file needs to be created and uploaded back to the server manually after editing. Saving the document in the client application updates the file on the server automatically. This provides a much more straightforward process of editing attached documents, especially for non-experienced or non-technical end users.

The Requirements and limitations topic gives you an overview of which software and configuration is required for WebDAV integration to be functional both on the server- and the client-side. The Configuration for WebDAV topic explains how a website needs to be installed and configured in order to meet these requirements. Finally, WebDAV-related settings need to be adjusted in the CMS.

Once you have met all pre-requisites and performed the required configuration, you can use WebDAV in two modes:

- Edit mode - this mode represents integration of WebDAV editing in Kentico CMS user interface. Using this feature, files uploaded in the CMS can be opened for WebDAV editing by clicking the **Edit in client application** ( ) icon.
- Browse mode - this mode enables you to map a network drive in your operating system. Content of the network drive reflects the content of your website and you can edit website files just as if they were stored on a local drive.



To learn details about WebDAV editing in each of the two modes, click the name of the mode above to get redirected to its sub-chapter. Both modes also fully integrate with workflow and versioning support in Kentico CMS. To learn more about system behavior specifics in this case, please proceed to the Integration with workflow and versioning topic.

8.52.2 Requirements and limitations

The following requirements must be met on the server- and the client-side in order for WebDAV integration to be functional in Kentico CMS:

Server-side requirements

- **Windows authentication** must be enabled in IIS and the CMS must be configured to use this type of authentication.
- If you are using an older version of IIS than 7.0, the Kentico CMS website must be **installed in the root** of the server. This is necessary because Microsoft Office submits configuration requests to the site root and requires the server to respond properly. If the website is not installed to the root, Microsoft Office will open its associated documents as read-only.
- If you are using IIS 7.0 or higher, the website needn't be installed in the root - it can also be installed in a virtual directory.
- IIS WebDAV is not compatible with Kentico CMS WebDAV integration. If you have WebDAV enabled in your IIS, Kentico CMS WebDAV integration will not be functional.

The Configuration for WebDAV topic explains how the required configuration can be achieved.

Client-side requirements

- WebDAV can be used only in **Internet Explorer 6 or higher**. No other web browsers are currently supported.
- **Client applications with WebDAV support** for a particular file types need to be installed on the client machine. E.g. *Microsoft Office 2003* or higher or *Paint* and *Notepad* in *Windows 7* support editing of associated file types using WebDAV.
- **Matching (x86) or (x64) versions** of both *Internet Explorer* and *Microsoft Office* must be installed. In other words, to be able to open documents using WebDAV in *Internet Explorer (x86)*, *Microsoft Office (x86)* must be installed on the client computer. For *Internet Explorer (x64)*, you need to have *Microsoft Office (x64)* installed.
- The **WebClient** service needs to be installed and running in order for WebDav clients to be able to connect to the server. For more information on how to install and run the service on different versions of Windows, visit the Installing and running WebClient service topic.

Editable file size limit

By default, WebDAV editing is only functional with files smaller than 50 MB.

In case that a user needed to edit larger files using WebDAV, they would need to go to *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\WebClient\Parameters* in their Windows registry and adjust the *FileSizeLimitInBytes* key to the required value.

Forbidden characters in file names

WebDAV does not support uploading and editing of files whose names contain the following characters: **%, &, +**. Uploading and editing of such files using WebDAV may result in unwanted behavior and is not recommended.

8.52.3 Configuration for WebDAV

The following steps need to be taken in order to install a Kentico CMS website with WebDAV support enabled. The beginning of the procedure is different for IIS 7.0 or higher and the previous versions of IIS.

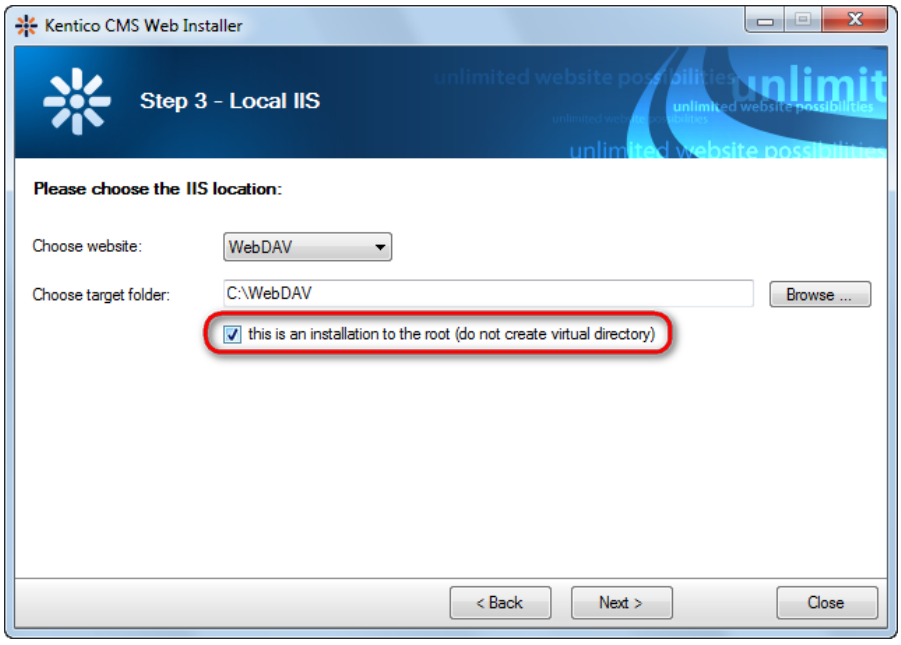
Step 1 on IIS versions prior to 7.0

Before you start, it is necessary to have a website created in the IIS root (i.e. not in a virtual directory). The website should have *Windows authentication* enabled in IIS.

1. Launch Kentico Web Installer. In step 2, choose **I want to use local IIS server**. In step 3, use the following configuration:

- **Choose web site**: choose the website that you have prepared in the IIS root
- **Choose target folder**: enter the path to the website's physical folder
- **This is an installation to the root (do not create virtual directory)**: enabled

Click **Next** and proceed through the rest of the wizard.

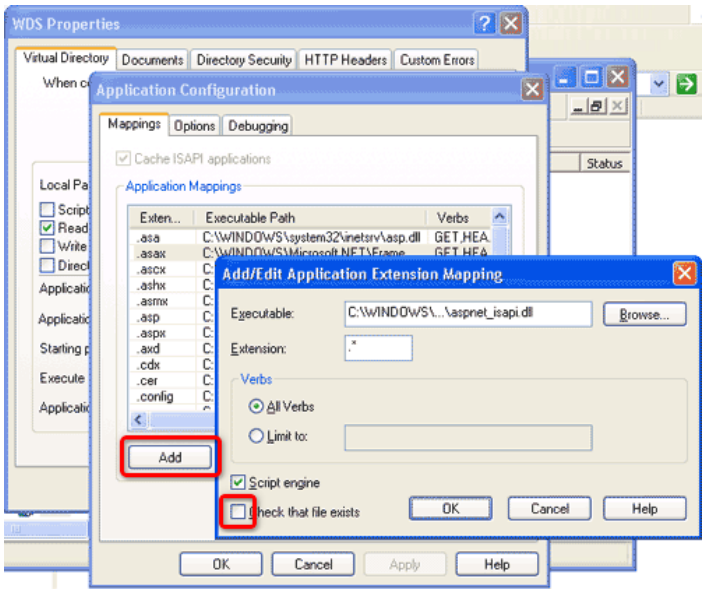


When using IIS versions older than 7.0, it is also necessary to correctly map the **aspnet_isapi.dll** file in IIS. The process of performing this configuration is described below.

Configuration for IIS 5 and earlier versions

Open the IIS Management console, right-click on your Kentico application in the tree and select **Properties**. On the **Home Directory** tab, click **Configuration** and then **Add** to create a new application extension mapping. Enter the following details into the dialog:

- **Executable**: <Windows install directory>\Microsoft.NET\Framework\<.Net framework version>\aspnet_isapi.dll
- **Extension**: .*
- **Verbs**: All Verbs
- **Check that file exists**: disabled

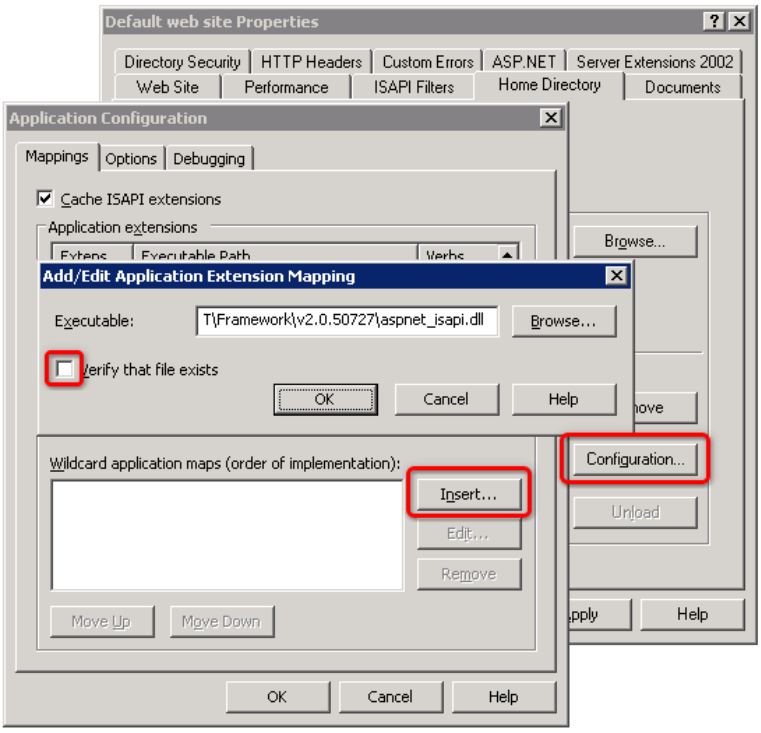


Click **OK** and return to the main management console. Now right-click the website under which your application is stored and select **Properties**. Switch to the **Home directory** tab, ensure that the **Read** box is checked and set the **Execute permissions** field to *Scripts only*.

Configuration for IIS 6

Open the IIS 6 Management console, right-click on your Kentico application in the tree and select **Properties**. On the **Home Directory** tab, click **Configuration** and then **Insert** to add a new wildcard application map. Enter the following details into the dialog:

- **Executable**: <Windows install directory>\Microsoft.NET\Framework\<.Net framework version>\aspnet_isapi.dll
- **Verify that file exists**: disabled



Click **OK** and return to the main management console. Now right-click the website under which your application is stored and select **Properties**. Switch to the **Home directory** tab, ensure that the **Read** box is checked and set the **Execute permissions** field to *Scripts only*.

Step 1 on IIS 7.0 or higher

On IIS 7.0 or higher, you do not need to have the website created in the IIS root - it can also be installed in a virtual directory.

1. Launch Kentico Web Installer and proceed through the steps with configuration according to your needs.

If you wish to run the website using an application pool set for *Classic* managed pipeline mode, please ensure that your **web.config** file contains the two **Isapi** extension handlers highlighted below once the **Web installer** finishes.

The handlers should be defined as follows on .NET 3.5:

```
...
<!-- WebDAV location BEGIN -->
<location path="cms/files">
  <system.web>
    <httpHandlers>
      <clear />
      <add verb="*" path="*" type="CMS.WebDAV.WebDAVHandler, CMS.WebDAV" />
    </httpHandlers>
    <httpRuntime executionTimeout="2400" maxRequestLength="2097151" />
  </system.web>
```

```
  <system.webServer>
    <handlers>
      <clear />
      <add name="aspnet_isapi 32-bit" path="*" verb="*" modules="IsapiModule"
      scriptProcessor="%windir%\Microsoft.NET\Framework\v2.0.50727
      \aspnet_isapi.dll" resourceType="Unspecified" requireAccess="Script"
      preCondition="classicMode,runtimeVersionv2.0,bitness32" />
      <add name="aspnet_isapi 64-bit" path="*" verb="*" modules="IsapiModule"
      scriptProcessor="%windir%\Microsoft.NET\Framework64\v2.0.50727
      \aspnet_isapi.dll" resourceType="Unspecified" requireAccess="Script"
      preCondition="classicMode,runtimeVersionv2.0,bitness64" />
      <add name="CMSWebDAVHandler" path="*" verb="*"
type="CMS.WebDAV.WebDAVHandler, CMS.WebDAV" />
    </handlers>
    <security>
      <requestFiltering>
        <requestLimits maxAllowedContentLength="2147483648" />
      </requestFiltering>
    </security>
  </system.webServer>
</location>
<!-- WebDAV location END -->
...
```

On .NET 4.0, the handlers should be defined the following way:

```
...
<!-- WebDAV location BEGIN -->
<location path="cms/files">
  <system.web>
    <httpHandlers>
      <clear />
      <add verb="*" path="*" type="CMS.WebDAV.WebDAVHandler, CMS.WebDAV" />
    </httpHandlers>
    <httpRuntime executionTimeout="2400" maxRequestLength="2097151" />
  </system.web>
  <system.webServer>
    <handlers>
      <clear />
      <add name="aspnet_isapi 32-bit" path="*" verb="*" modules="IsapiModule"
      scriptProcessor="%windir%\Microsoft.NET\Framework\v4.0.30319
      \aspnet_isapi.dll" resourceType="Unspecified" requireAccess="Script"
      preCondition="classicMode,runtimeVersionv4.0,bitness32" />
      <add name="aspnet_isapi 64-bit" path="*" verb="*" modules="IsapiModule"
      scriptProcessor="%windir%\Microsoft.NET\Framework64\v4.0.30319
      \aspnet_isapi.dll" resourceType="Unspecified" requireAccess="Script"
      preCondition="classicMode,runtimeVersionv4.0,bitness64" />
      <add name="CMSWebDAVHandler" path="*" verb="*"
type="CMS.WebDAV.WebDAVHandler, CMS.WebDAV" />
    </handlers>
    <security>
      <requestFiltering>
        <requestLimits maxAllowedContentLength="2147483648" />
      </requestFiltering>
    </security>
```

```
  </system.webServer>
</location>
<!-- WebDAV location END -->
...
```

If necessary, adjust the path in the **scriptProcessor** attributes of the handlers according to your current .NET framework version.

The rest of the configuration - common for all IIS versions

2. When the web installation is complete and the appropriate configurations have been made according to your IIS version and application pool mode, open the website using the link in the last step of the **Web installer**. When you access the URL, **Kentico CMS Database Setup** will be displayed. Follow the wizard as described in Installation and deployment -> Installation procedure -> Database setup.

3. After finishing the Database Setup, go to **Site Manager -> Settings -> Integration -> WebDAV** and configure the settings as described in the Settings topic.

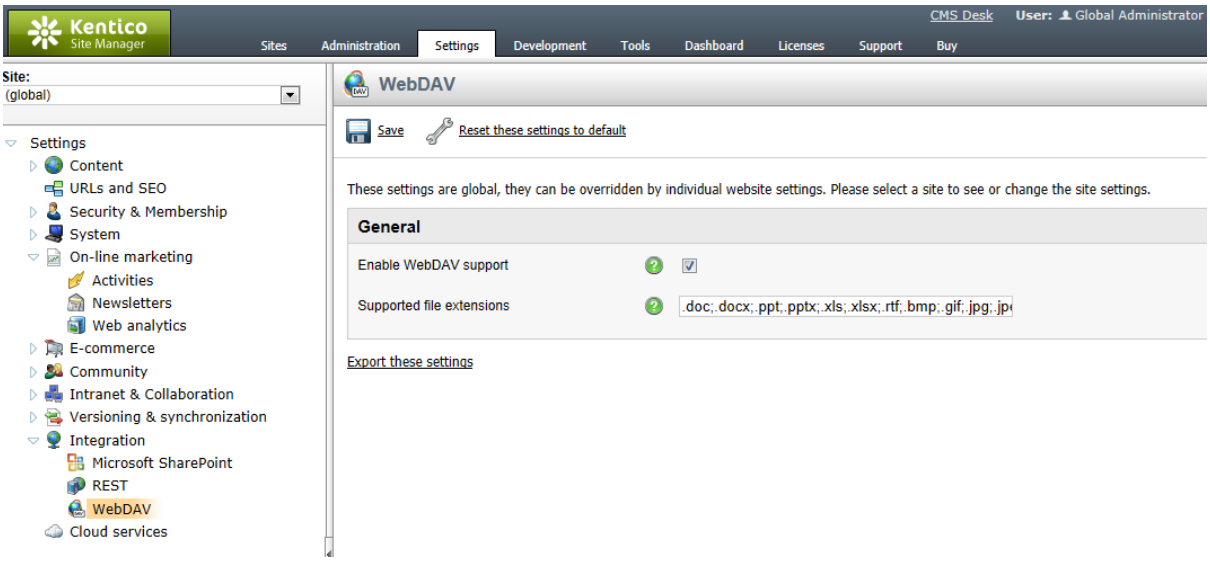
4. The last step is to enable **Windows authentication** for your website. Please follow the instructions in Development -> Membership -> Authentication -> Windows authentication -> Configuring Windows authentication.

With all these steps performed, WebDAV editing should be possible. Please refer to the Editing files using WebDAV topic to learn more about the editing possibilities.

8.52.4 Settings

Settings related to WebDAV integration can be found in **Site Manager -> Settings -> Integration -> WebDAV**. The following settings are available:

- **Enable WebDAV support** - this options enables/disables WebDAV in both Edit mode and Browse mode.
- **Supported file extensions** - list of file extensions that should be editable using WebDAV Edit mode. The **Edit in client application** ( ) icon is only displayed next to documents with this extension. Enter extensions with or without the leading dots, separated by semicolons; e.g. *.docx; .xlsx;.pptx;jpg;jpeg*. An appropriate client application with WebDAV support must be installed on the client machine for each listed extension in order for WebDAV editing of these files to be possible.



8.52.5 Integration with workflow and versioning

WebDAV editing fully integrates with workflow and versioning. The following text explains behavior specifics of the system when editing attachments or files in file fields of documents to which workflow and versioning is applied. The behavior is explained separately for each of the two supported WebDAV modes.

Edit mode

When there is a workflow defined for a document, the same rules apply when editing its file fields or attachments using WebDAV Edit mode as when editing the document's content. When its file fields or attachments are edited using WebDAV Edit mode, a new version of the document is created, check-in/check-out is performed etc., based on settings of the workflow applied to the document.

Browse mode

In Browse mode, documents under workflow or their attachments can only be edited by users that are allowed to edit the document in the current workflow step. Please note that because WebDAV only works with Windows Authentication, the account used to log on to Windows needs to be imported as a user account in Kentico CMS. If you allow editing in a workflow step to roles where the CMS user is member, Browse mode editing for the Windows account will be allowed. If a user tries to open a document in a workflow step that they can't approve, the document is opened as read-only.

When a document under workflow with check-out/check-in enabled is opened in Browse mode, it is checked out automatically. A checked out document can only be edited by the user who checked it out, other users can only open it as read-only. When it is saved and closed, automatic check-in is performed so that the document can be edited by other users again.

8.52.6 Edit mode

8.52.6.1 Edit mode overview

WebDAV Edit mode integrates WebDAV editing into the user interface of Kentico CMS. With Edit mode enabled, the **Edit in client application** ( ) icon is displayed next to files throughout the CMS. Clicking the icon opens the respective file for editing in a client application.

WebDAV Edit mode is integrated in various sections throughout the whole UI. The following topics will give you an overview of these sections:

- Editing files using WebDAV - explains the general process of WebDAV editing in a simple example, showing how to edit document attachments and files stored in a file field of a document (typically the *CMS.File* document type).
- Editing metafiles using WebDAV - explains WebDAV editing of metafiles (e.g. E-commerce product images, web part teaser images, etc.).
- Integration with WYSIWYG editor dialogs - explains how to use WebDAV editing in dialogs of the WYSIWYG editor.
- Integration with Media libraries - explains how to use WebDAV editing in the administration interface of the Media libraries module.

WebDAV editing has also been integrated into the following built-in modules to provide the functionality even to live site users:

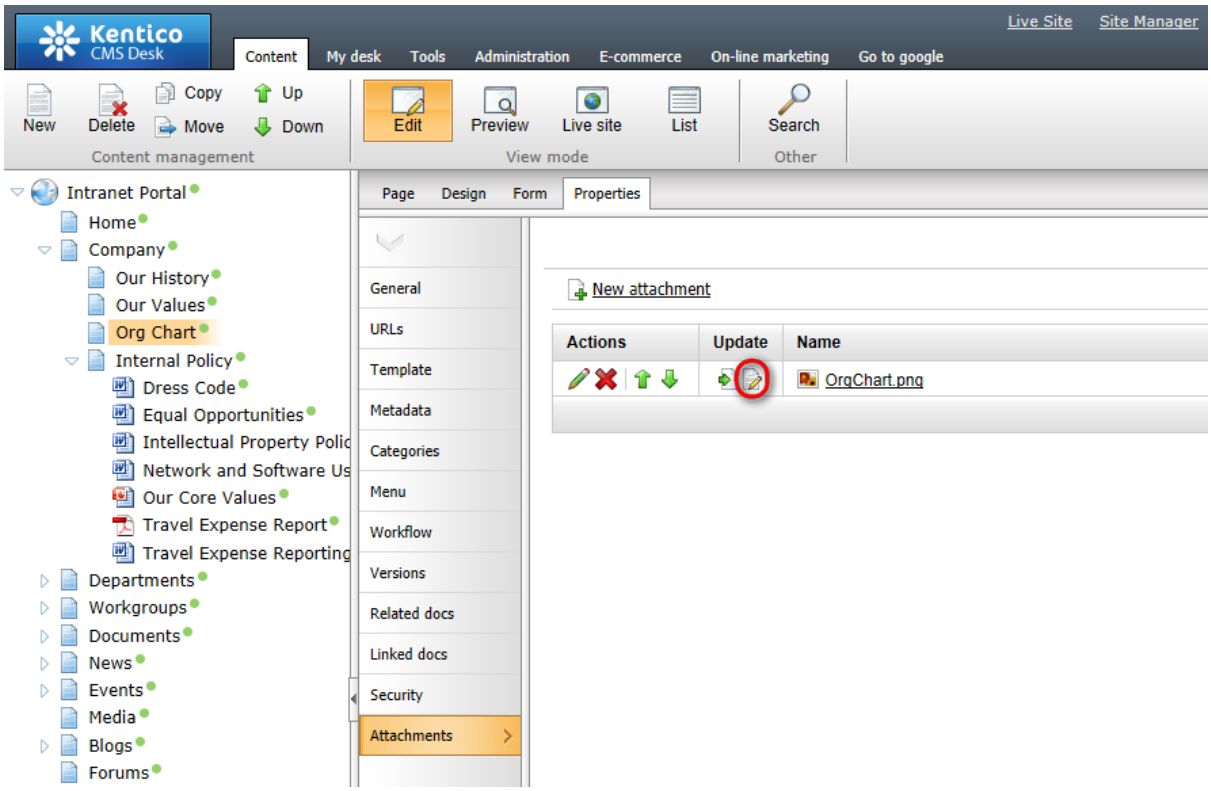
- Integration with Document library - explains how live site users can edit documents stored by means of the Document library module.
- Integration with User contributions - explains how live site users can edit files uploaded as part of documents submitted via the User contributions module.

In case that you want to learn more about WebDAV Browse mode, the other WebDAV editing mode supported by Kentico CMS, please proceed to the Browse mode overview topic in the following sub-chapter.

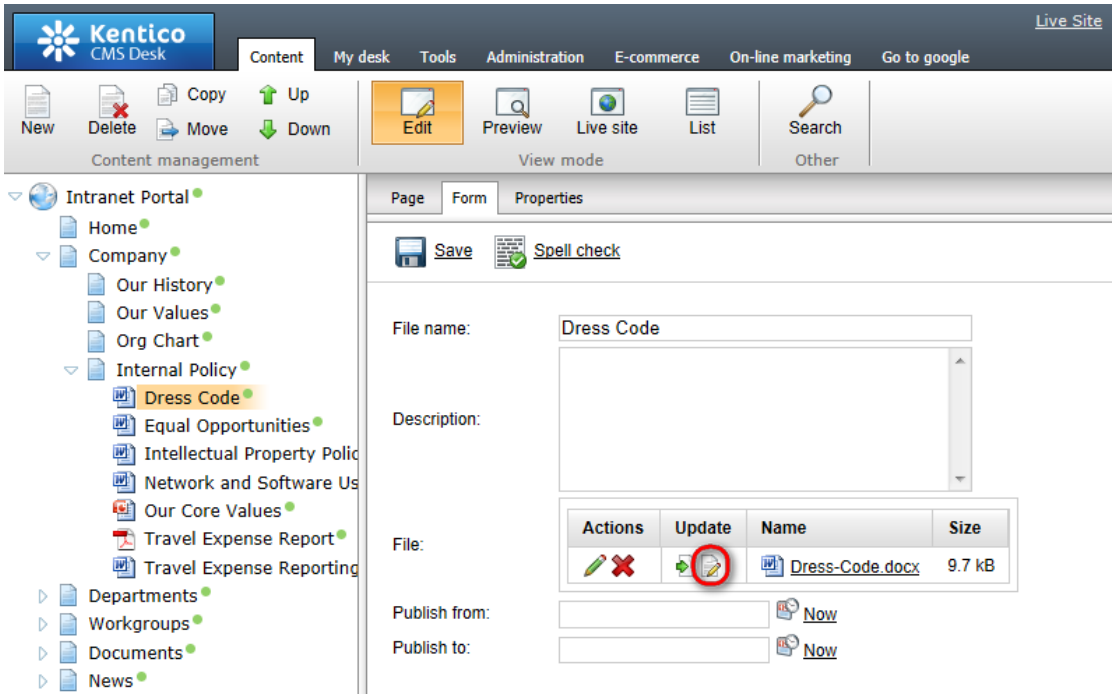
8.52.6.2 Editing files using WebDAV

After configuring your server as described in Configuration for WebDAV, users who meet the client-side requirements listed in the Requirements topic may start editing document attachments or documents using the File field (e.g. *CMS.File* document type) using WebDAV.

1. WebDAV editing of document attachments is possible in **CMS Desk -> Content -> Edit**, after selecting a document in the content tree and switching to its **Properties -> Attachments** tab. The **Edit in client application ( )** icon should be displayed next to each attachment whose extension is included in **Site Manager -> Settings -> Integration -> WebDAV -> Supported file extensions**.

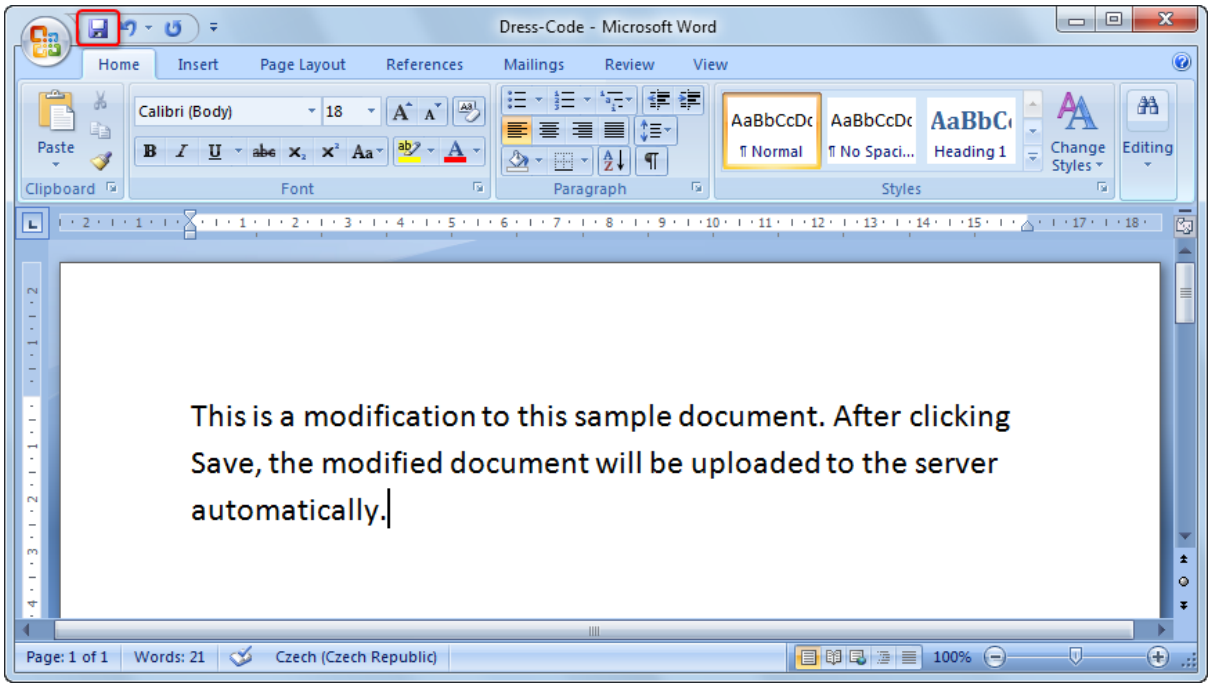


2. The same **Edit in client application ( )** icon is displayed on the **Form** tab of documents using the File field (e.g. *CMS.File* document type) where a file with a supported extension is uploaded.

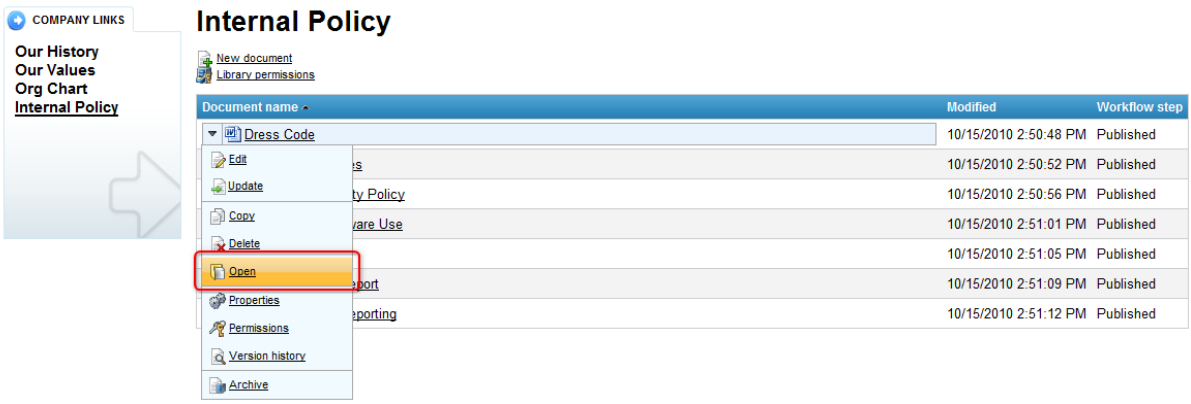


3. Clicking the icon opens the document in a client application associated with the particular file extension (e.g. *.docx* documents are opened in MS Office Word). Try editing the opened file and save

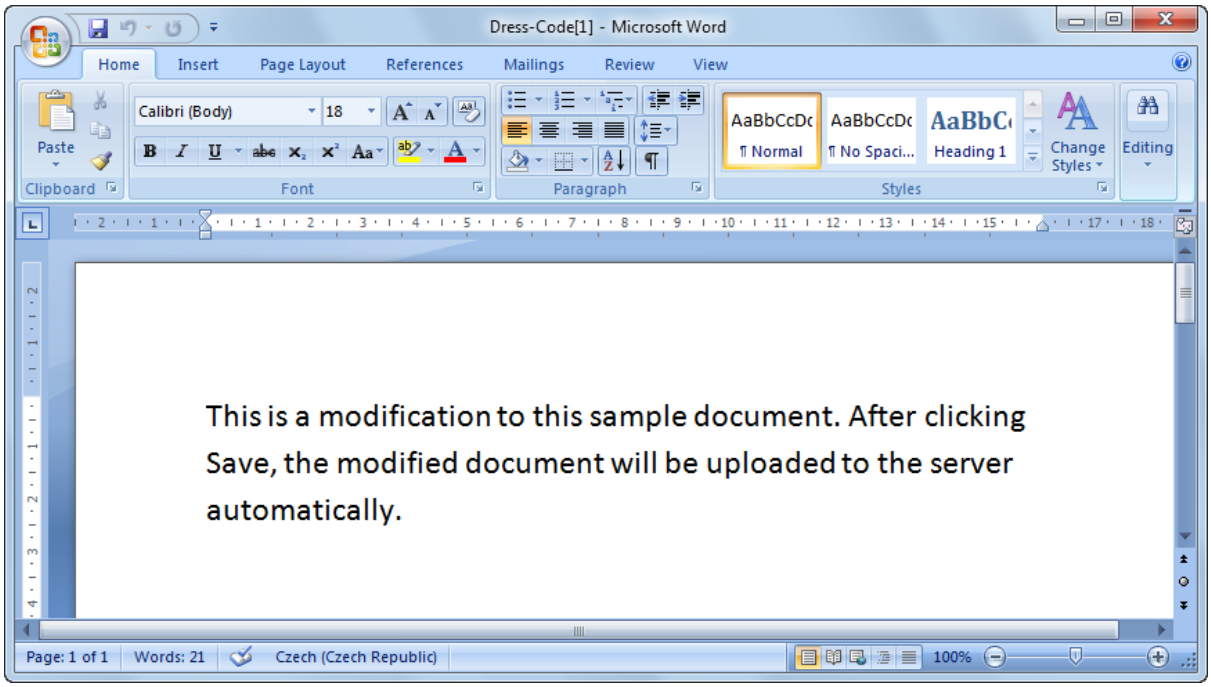
your changes using the **Save** button.



4. Now close the client application and try to open the document again, either from the UI or from the live site.



5. You should see that the file that you modified has been saved to the server and the version that you have just opened contains the modifications you made to it in the previous step.



You have learned how document attachments or documents using the File field can be edited using a client application when WebDAV integration is enabled. Editing of any other file types in any other client application (e.g. *Microsoft Office Excel*, *PowerPoint*, *Paint*, etc.) can be performed exactly the same way - all you need to do is click the **Edit in client application ( )** icon, edit the file and save it in the client application. Transfer of the updated file to the server is handled by the CMS automatically.

Required permissions

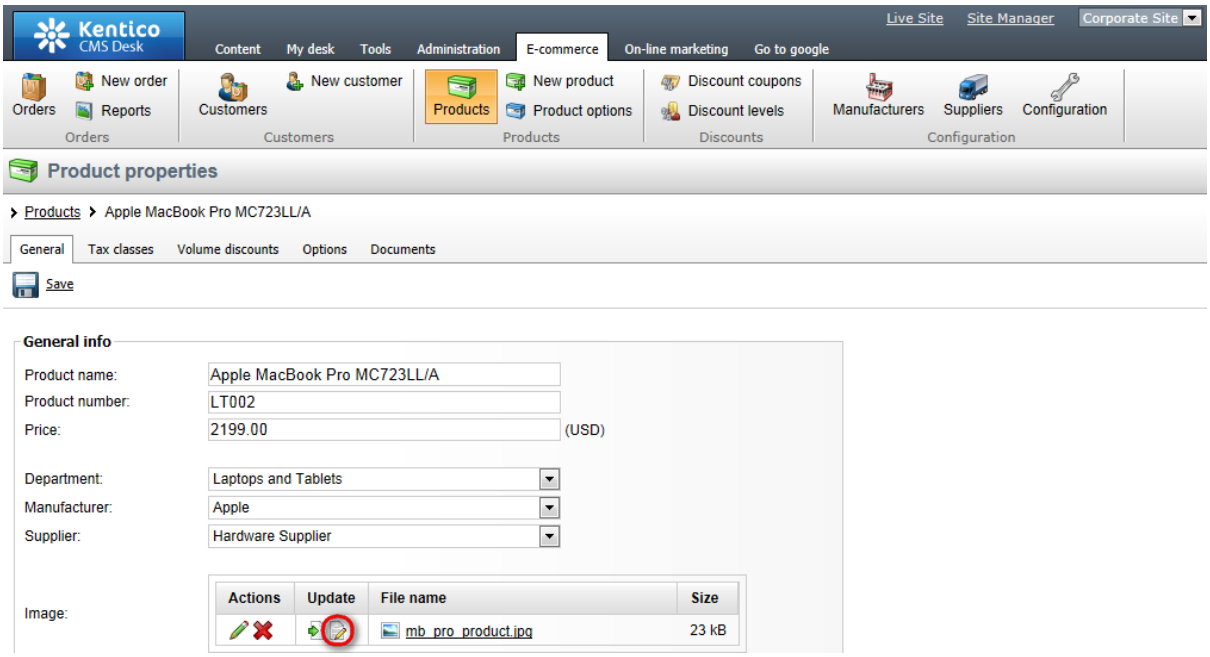
For the **Edit in client application ( )** icon to be enabled, the **Modify** permission for the current document must be granted to the current user (or one of their roles) on one of the three levels described in Development -> Membership -> Permissions -> Document permissions.

8.52.6.3 Editing metafiles using WebDAV

Metafiles in Kentico CMS are files stored together with system objects. Typical examples of metafiles are:

- E-commerce product images
- web part thumbnails
- widget thumbnails
- page template thumbnails
- etc.

All these files can be edited using WebDAV Edit mode — in the respective fields, you can find the **Edit in client application** ( ) icon, just as highlighted in the screenshot below. Clicking the icon opens the document for editing in a client application (if there is an application for the particular file type installed on the client machine and if the application supports WebDAV).



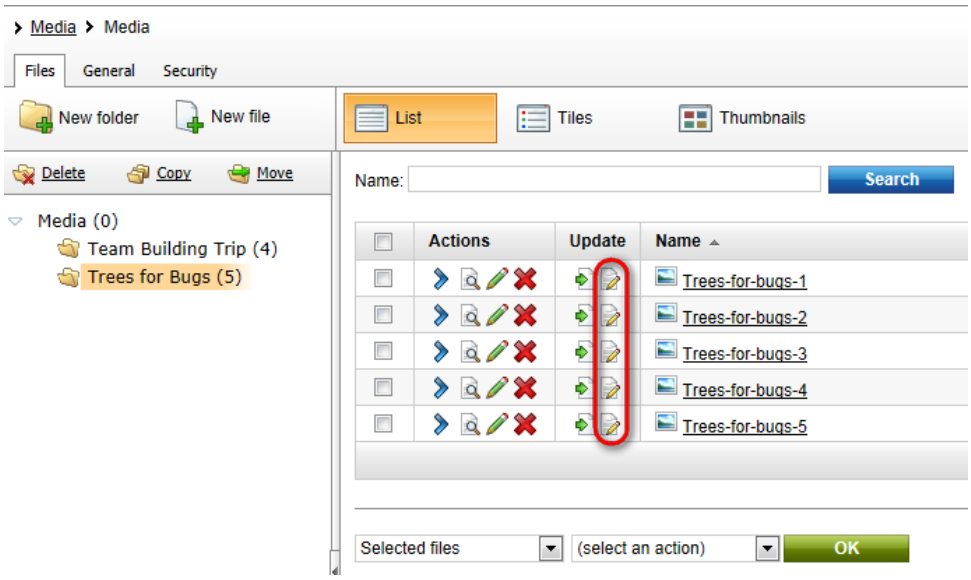
Required permissions

The following metafiles can be edited by site editors with appropriate module permissions:

Metafile	Permissions matrix	Required permissions
Forms metafiles	Module -> Forms	Read form, Edit form
E-commerce invoice	Module -> E-commerce	Read configuration, Modify configuration
E-commerce product images	Module -> E-commerce	Read products, Modify products
Newsletters (issue attachments)	Module -> Newsletters	Read, Author newsletter issues
Newsletters (issue template attachments)	Module -> Newsletters	Read, Manage templates
Reports (General tab -> Attachments)	Module -> Reporting	Read, Modify
All other metafiles can only be edited by **global administrators**.		

8.52.6.4 Integration with Media libraries

Media library files can be edited using WebDAV as well. In the libraries, the **Edit in client application** ( ) icon is displayed next to files with the supported file extensions (see Settings for more details). Clicking the icon opens the document for editing in a client application (if there is an application for the particular file type installed on the client machine and if the application supports WebDAV).

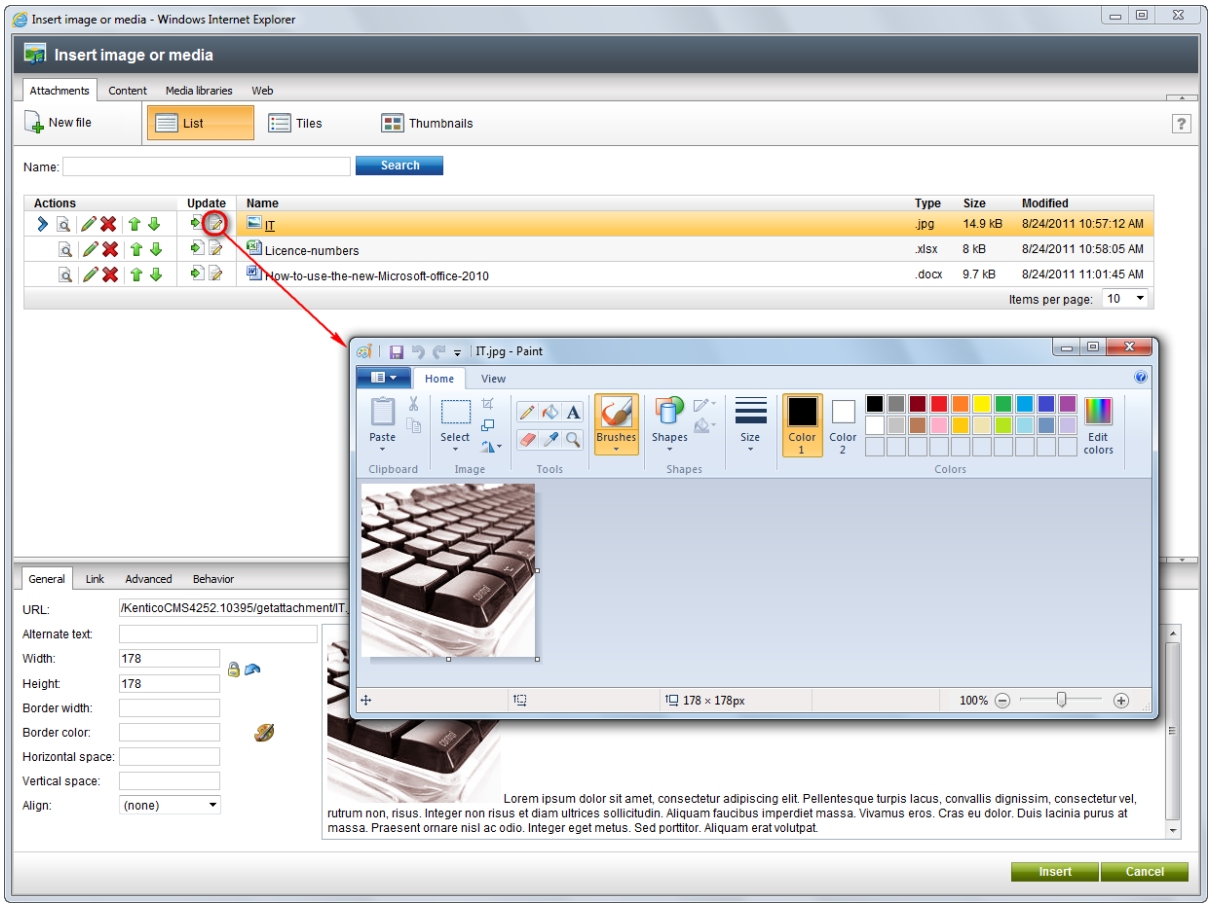


Required permissions

For the action to be available, the user must either have the **Manage** permission for the **Media libraries** module, or the **Modify file** permission in security configuration of a particular media library. See Modules -> Media libraries -> Security -> Media library permissions for more details.

8.52.6.5 Integration with WYSIWYG editor dialogs

WebDAV editing is also integrated in the Insert/Edit image or media and Insert/Edit link dialogs of the WYSIWYG editor. In these dialogs, the **Edit in client application ( )** icon is displayed on the **Attachments**, **Content** and **Media libraries** tabs, next to files with the supported file extensions (see Settings for more details).



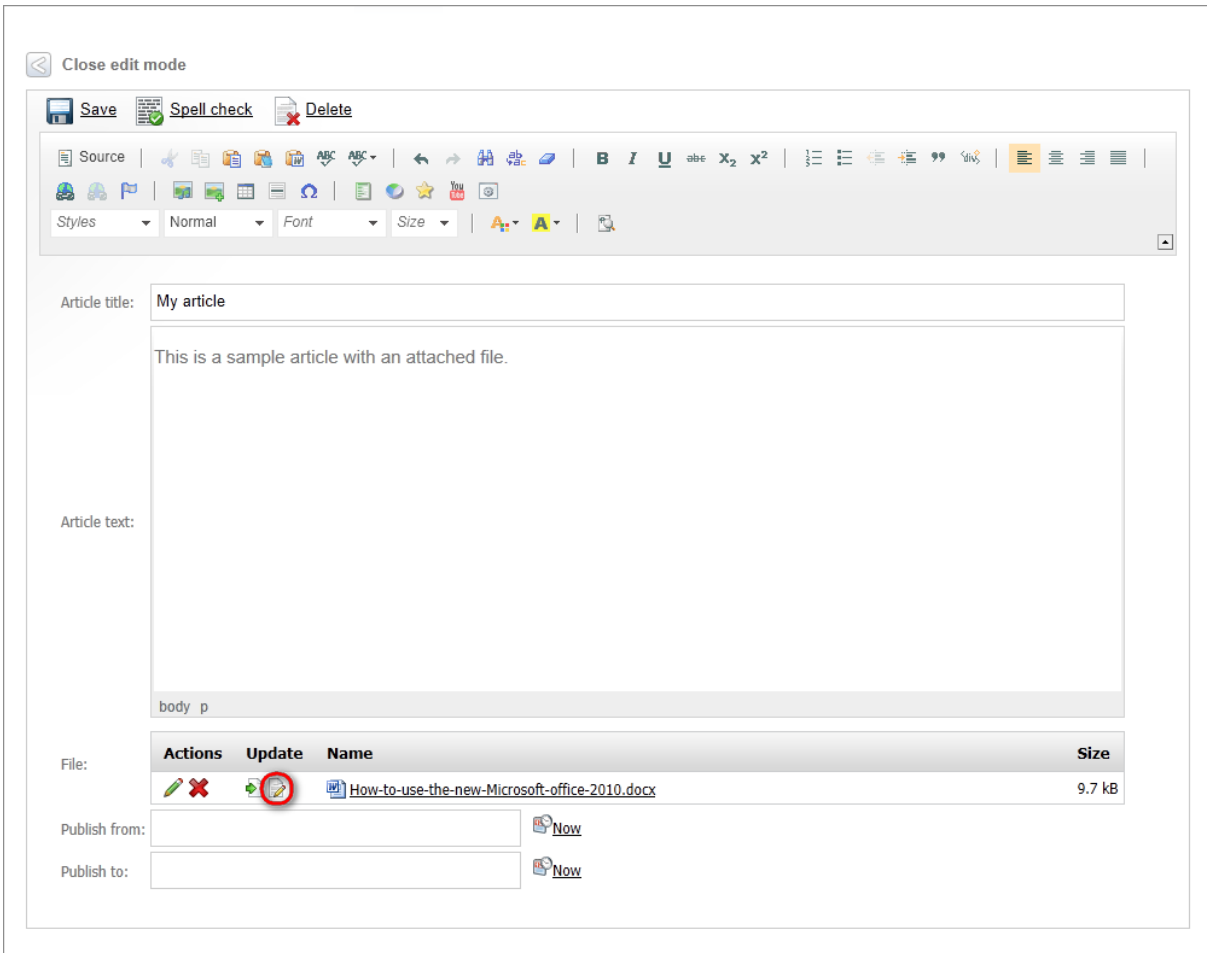
Required permissions

For the icon to be displayed on the **Attachments** tab, the **Modify** permission for the current document needs to be granted to the current user. On the **Content** tab, the icon is displayed only with documents for which the **Modify** permission is granted to the current user. Please refer to Development -> Membership -> Permissions -> Document permissions for more details on the three-level document permissions hierarchy in Kentico CMS.

For the action to be available on the **Media libraries** tab, the user must either have the **Manage** permission for the **Media libraries** module, or the **Modify file** permission in security configuration of a particular media library. See Modules -> Media libraries -> Security -> Media library permissions for more details.

8.52.6.6 Integration with User contributions

WebDAV editing is integrated into the User contributions module, which brings WebDAV editing to the live site. If a document type used for user contributions contains a field of the **File** or **Document attachment** type, the **Edit in client application (** **)** icon is displayed with files uploaded into these fields. Clicking the icon opens the document for editing in a client application (if there is an application for the particular file type installed on the client machine and if the application supports WebDAV).

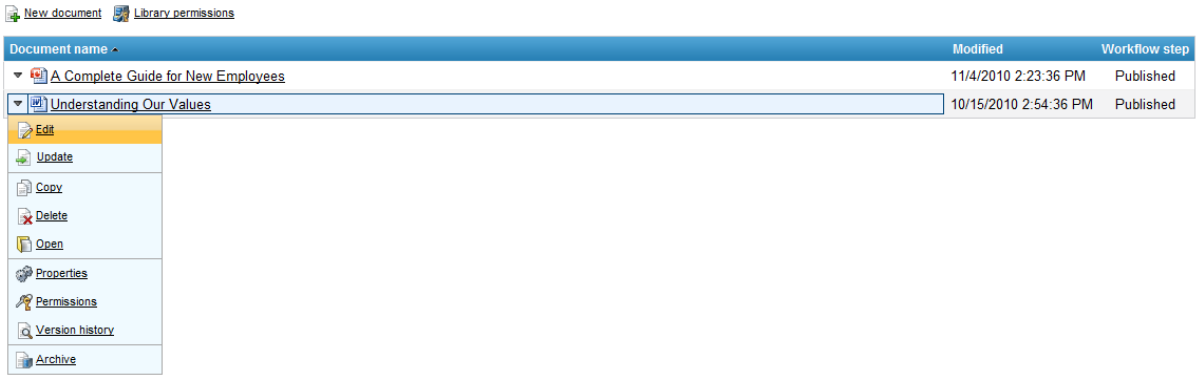


Required permissions

For the icon to be displayed in the **Contribution list** or **Edit contribution** web parts, the **Check permissions** property of the web part needs to be enabled and the current user must have appropriate permissions to edit the document, as described in Modules -> User contributions -> Security. When using the Groups module versions of the web parts, the icon is displayed either under the same conditions as described above or when the current user is the administrator of the respective group.

8.52.6.7 Integration with Document library

Files stored in a Document library can also be edited in a client application using WebDAV directly on the live site. There is the **Edit** action available in their drop-down menu, as can be seen in the screenshot below. Clicking the icon opens the document for editing in a client application (if there is an application for the particular file type installed on the client machine and if the application supports WebDAV).



Required permissions

For this action to be enabled, the document must have one of the supported extensions (see Settings for more details) and the **Modify** permission for the particular document must be granted to the current user. Please refer to Modules -> Document library -> Security for more details on permissions in document libraries.

8.52.7 Browse mode

8.52.7.1 Browse mode overview

WebDAV Browse mode enables you to **map a network drive** in your operating system that represents the content of your website. Files stored by the website can be found on the drive, letting you **open, edit and save** the files in a client application just as if they were stored on your local drive.

For to start using WebDAV Browse mode, you need to meet all requirements and performed the necessary configuration. After that, you only need to map the network drive in the operating system. The Mapping a WebDAV network drive topic explains two approaches how this can be achieved. When the network drive is mapped, you can start editing website files right away.

The network drive contains the following folders:

- attachments - contains document attachments and files stored in documents' file fields. They can be found in folders resembling the website's content tree structure.
- content - contains CMS.File documents stored in folders resembling the website's content tree structure.
- media - contains a folder for each media library on the website, while each folder contains the actual content of the respective library.
- groups - contains a folder for each group defined on the site, while each group folder contains the three folders mentioned above, containing only those attachments, content and media that belong to the particular group.

The **media** and **groups** folders may not be present, depending on if the Media libraries and Groups modules are installed. Click a folder name above in order to get more information about how files are stored in it. To get a quick overview of how Browse mode editing actually works, you may refer to the Example of Browse mode editing topic.

8.52.7.2 Mapping a WebDAV network drive

Once you have Kentico CMS installed and configured as described in the Configuration for WebDAV topic, you only need to map a network drive in your operation system that will be pointing to the web server. There are two ways how this can be done:

1. Mapping the network drive in Windows command line

1. Open Windows command line (press **[window]+R**, type *cmd* and click **OK**).

2. Enter the following command:

```
net use x: http://<your_domain>/cms/files
```

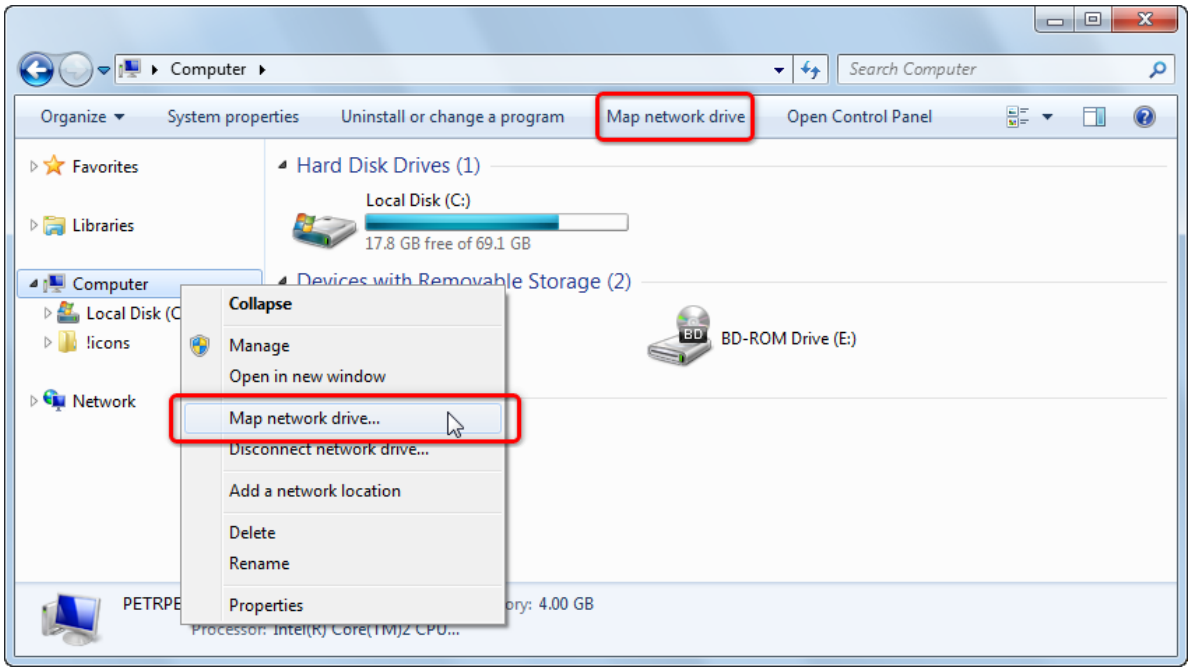
The *x:* part of the command determines which letter will be used for identification of the network drive in your system. Instead of *x*, you can use any letter that is not used for another drive yet.

The <*your_domain*> part of the command needs to be replaced with the domain name of your website. So, for example, if your website is running on *http://www.example.com*, you would need to enter *net use x: http://www.example.com/cms/files*. Similarly for *http://localhost/KenticoCMS*, you would need to enter *net use x: http://localhost/KenticoCMS/cms/files*.

2. Mapping the network drive in Windows UI

The following procedure is demonstrated on Windows 7. In other versions of Windows, it may be slightly different, while its principles and the entered values remain the same.

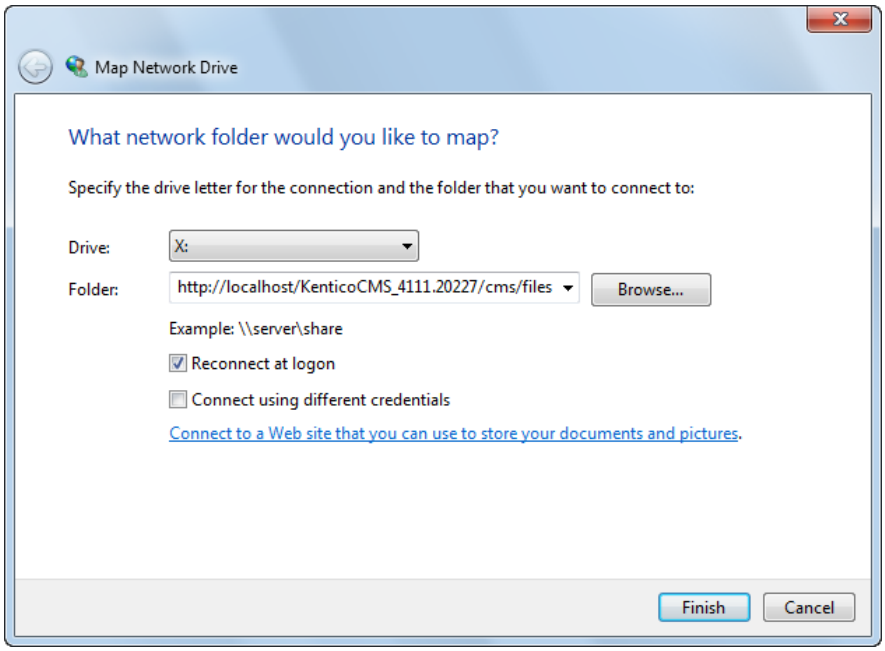
1. Open **Windows Explorer** and select **Computer** in the left menu. Right-click it and select **Map network drive**. Alternatively, the **Map network drive** action can be executed from the top menu, as highlighted in the screenshot below.



2. In the **Map Network Drive** dialog, adjust the following values:

- **Drive** - select the letter that will be assigned to the network drive in Windows.
- **Folder** - enter a URL in format *http://<your_domain>/cms/files* where the <*your_domain*> part should be replaced with the domain name of your website.
- **Reconnect at logon** - leave this option enabled if you want the drive to be connected next time your start Windows.
- **Connect using different credentials** - use this option in case that you want to use a different user account than your current Windows account. In such case, you will be asked to enter logon credentials after clicking **Finish**.

Click **Finish** and wait until the network drive gets connected.



Mapping a network place on Windows XP/Server 2003/Vista

When you experience problems while mapping a network drive on Windows XP, Windows Server 2003 or Windows Vista, you can map a network place instead:

1. Open the **Start** menu, right-click **My Computer** (or **Computer** on Vista) and choose **Map Network Drive** from the context menu.

2. In the dialog which pops up, click the link saying *Sign up for online storage or connect to a network server* (on Vista, the link says *Connect to a Web site that you can use to store your documents and pictures*). This will open the **Add Network Place Wizard**.

3. In the first step, click **Next**. In the second step, choose **Choose another network location** and click **Next**. In the third step, enter the URL in the *http://<your_domain>/cms/files* format and click **Next**.

4. In the fourth step, enter a name that will be used for the network location (e.g. *MyNetworkPlace*) and click **Next**. In the final step, click **Finish**.

Once finished, the network place should be mapped and content of the WebDAV network drive should be accessible through it. Please note that in this case, the content will not be accessible under a drive letter (e.g. *X:*\), but under the name of the network place specified in step 4 of the wizard (e.g. *MyNetworkPlace*\).

Accessing the network drive

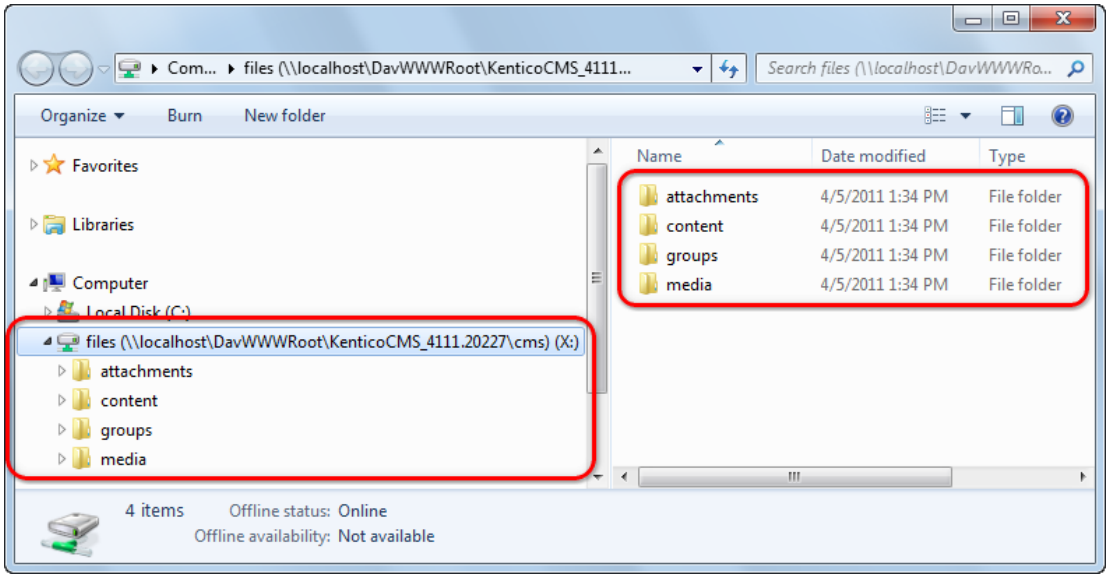
Once you have the network drive connected, it is recommended to check that you can access it. Open

Windows Explorer or any other file manager that you are using. The drive should be available among other drives in your system, under the letter that you assigned to it when you mapped it.

If you open the drive, you should see that it contains four folders. Each of them contains different types of files stored by your website:

- attachments - contains document attachments and files stored in documents' file fields. They can be found in folders resembling the website's content tree structure.
- content - contains CMS.File documents stored in folders resembling the website's content tree structure.
- media - contains a folder for each media library on the website, while each folder contains the actual content of the respective library.
- groups - contains a folder for each group defined on the site, while each group folder contains the three folders mentioned above, containing only those attachments, content and media that belong to the particular group.

Click a folder name above to learn more about content of the respective folder.



Unusually long response time

If you are experience unusually long delays when opening a WebDAV drive, copying files to or from it or switching between different WebDAV drives, please make sure you have the **Automatically detect settings** option disabled in **Internet Explorer -> Tools -> Internet Options -> Connections -> LAN Settings**.

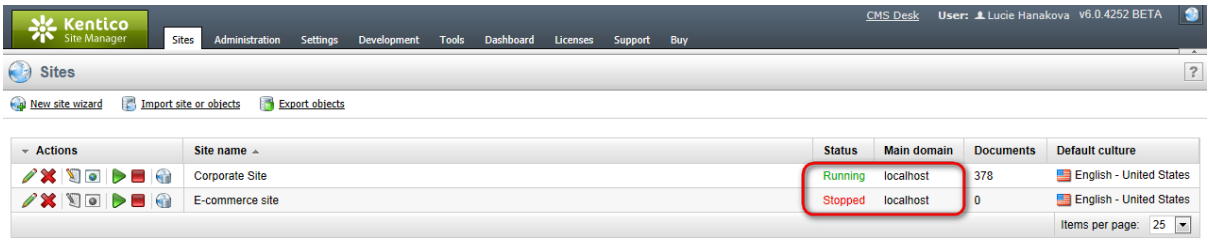
See the following article for more details: http://support.microsoft.com/kb/2445570

Multiple websites using the same domain

In a typical scenario, each website running in Kentico CMS uses a different domain. Therefore, the URL

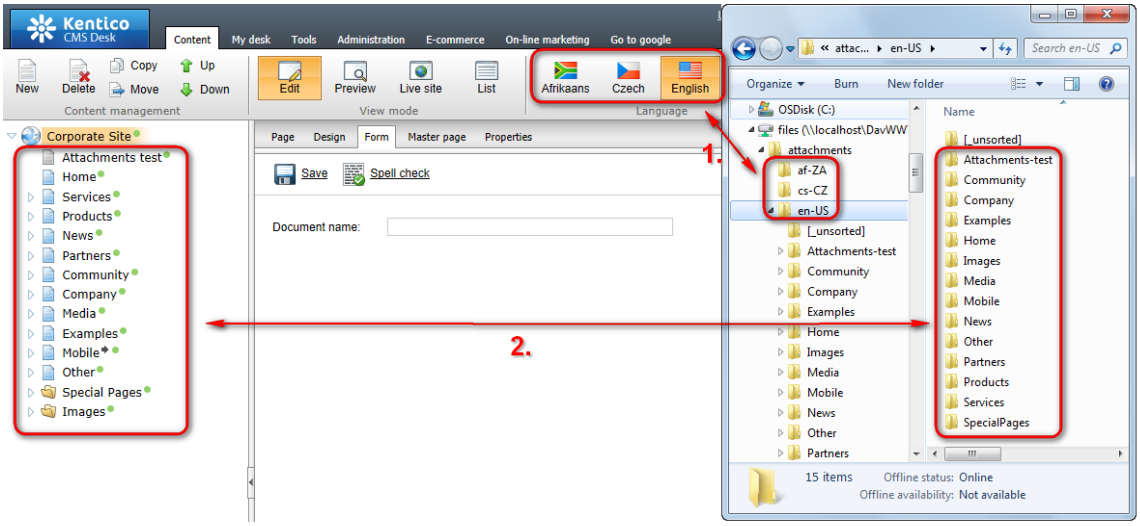
of the mapped network drive would be different for each site, letting you map a drive separately for each website.

However, if you have multiple websites using the same domain (typically *localhost* in a development environment), only one of these websites can be running at a time while all others are stopped. In this case, the network drive always displays content of the site that is currently running.

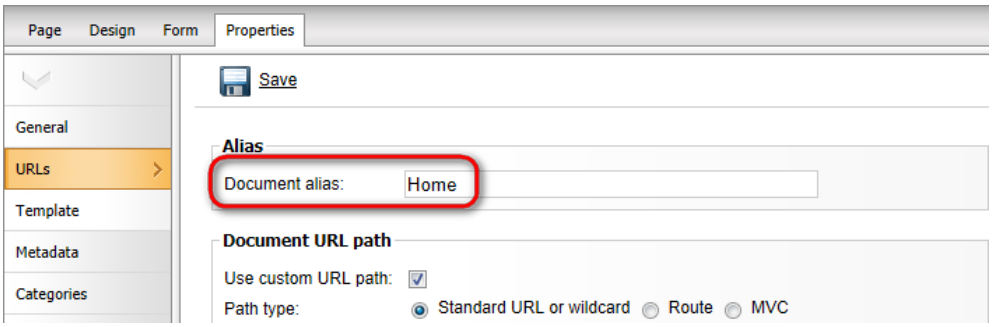


8.52.7.3 Attachments

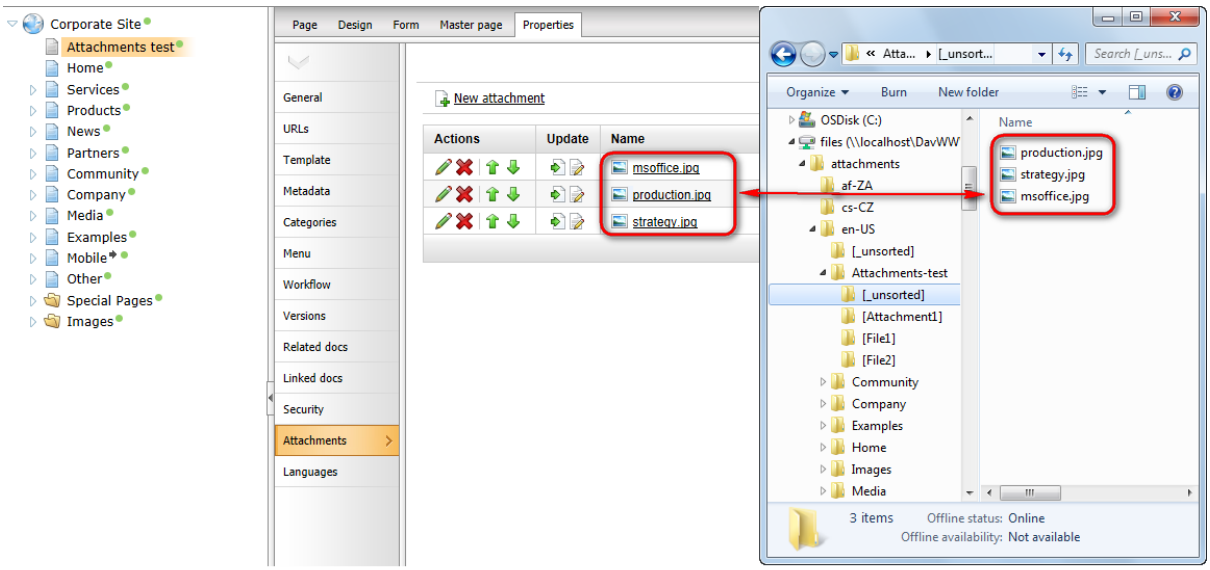
The **attachments** folder in the root of your WebDAV network drive contains document attachments of all documents in the website's content tree. Under the **attachments** folder, there is one folder for each culture used for the website (1. in the screenshot below). Under each of the folders, you can find one folder for each document in the content tree in the particular language version (2. in the screenshot below).



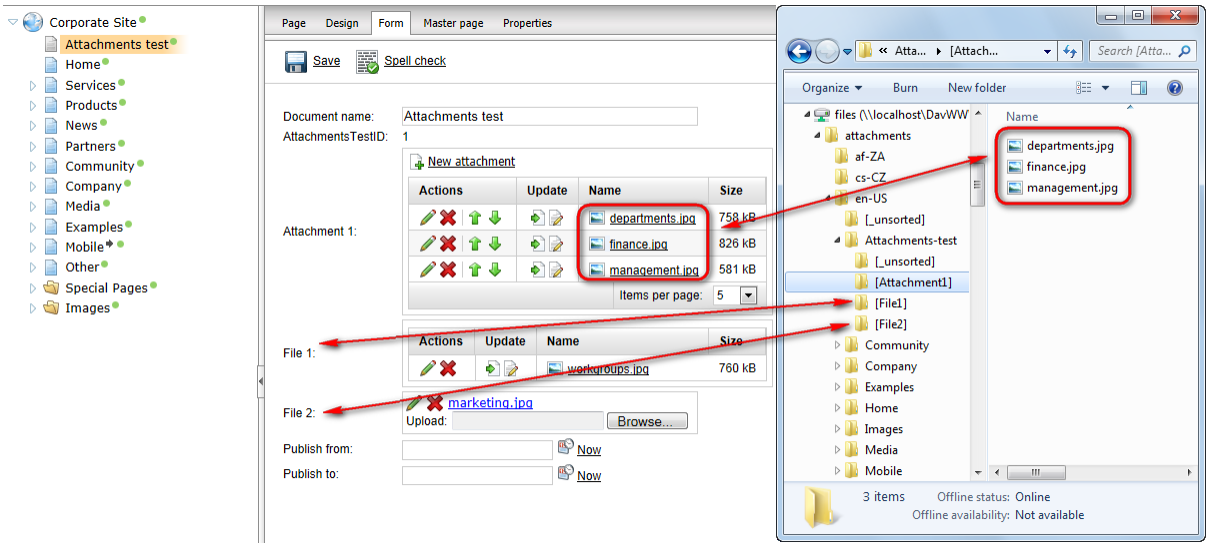
Please note that **folder names reflect node aliases** of particular documents, **not their document names**. This is necessary for unique identification of documents in URLs. Node aliases of documents can be viewed and modified on the **Properties -> URLs** tab, in the **Document alias** field.



The **[_unsorted]** folder does not represent any document in the content tree. It is present under each folder representing a document and contains its unsorted attachments. In the root, this folder contains unsorted attachments of the content tree root. Unsorted attachments are the ones that are added to documents on the **Properties -> Attachments** tab.



If you expand a folder representing a document that has some grouped attachment fields or file fields, you should see a folder for each of the fields. The folder has the same name as the respective field's DB column, wrapped in square brackets like **[MyAttachmentField]**. Under it, you can find all files uploaded into the field.



Possible actions and required permissions

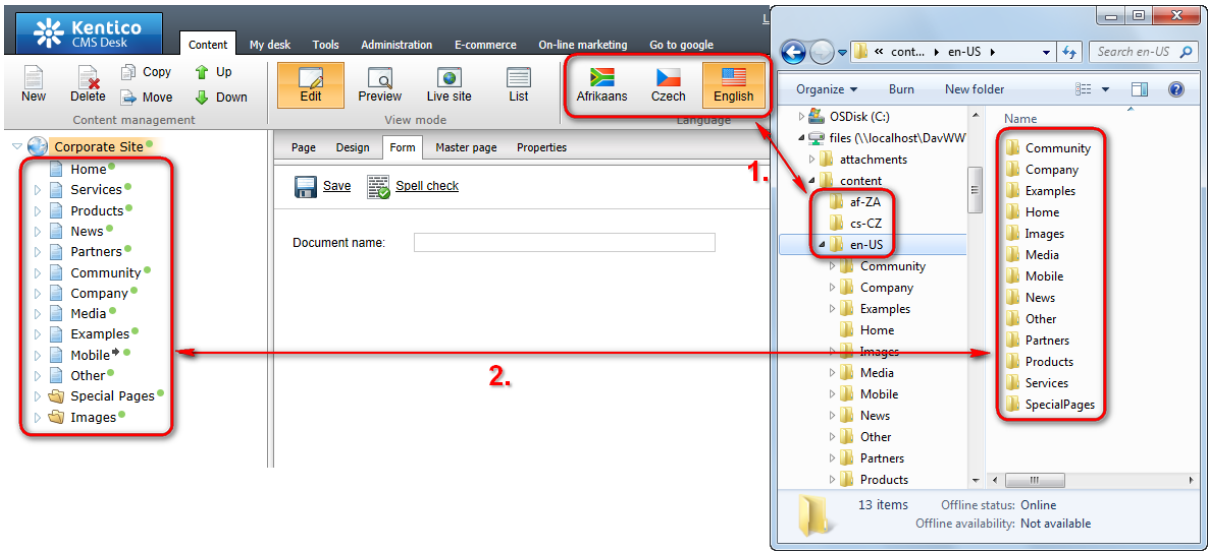
The following document permissions need to be granted to users (their roles) in order to perform respective actions with document attachments:

Action	Required permission
Browse folder structure	Browse tree
Display attachments & open them in read-only mode	Read
Upload attachments	Modify, while the file must have one of the extensions defined in the *Allowed extensions* value for the respective field, or globally in *Site Manager -> Settings -> System -> Files -> Upload extensions* in case that the field is configured to inherit from settings.
Edit attachments	Modify
Delete unsorted attachments	Modify
Delete grouped attachments	Modify, while the *Allow empty value* option must be enabled for the respective field.

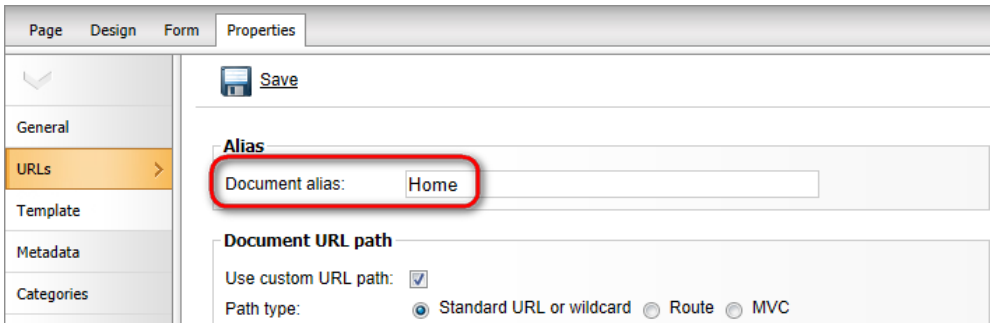
Please note that because WebDAV only works with Windows Authentication, the account used to log on to Windows needs to be imported as a user account in Kentico CMS. If you grant permissions to roles where the CMS user is member, Browse mode editing for the Windows account will be allowed.

8.52.7.4 Content

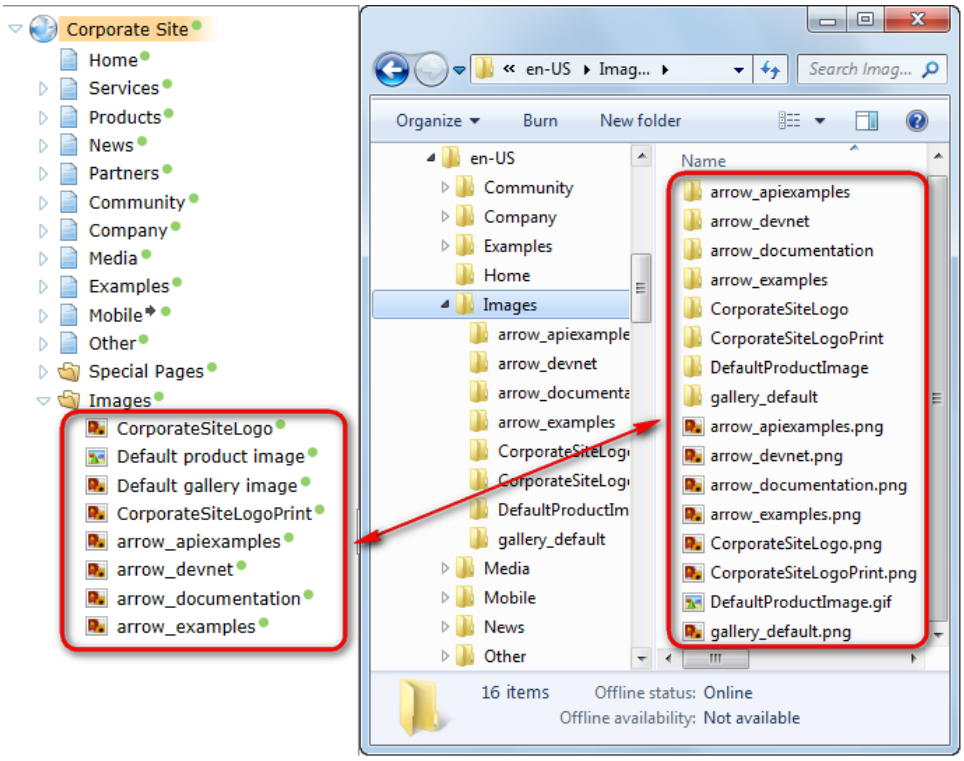
The **content** folder contains documents of the **CMS.File** document type (see Content management -> File management for more details). Under the **content** folder, there is one folder for each culture used for the website (1. in the screenshot below). Under each of the folders, you can find a folder for each document in the content tree in the particular language version (2. in the screenshot below).



Please note that **folder names reflect node aliases** of particular documents, **not their document names**. This is necessary for unique identification of documents in URLs. Node aliases of documents can be viewed and modified on the **Properties -> URLs** tab, in the **Document alias** field.



While standard documents are represented by a folder, *CMS.File* documents are here both as a folder and as a file. The file represents the actual physical file stored in the document's file field. The folder is here for the case that the *CMS.File* document had some child documents, as those would be located under it in this case (as the *Services_webdevelop* document under the *CompanyLogo* document in the screenshot below).



Possible actions and required permissions

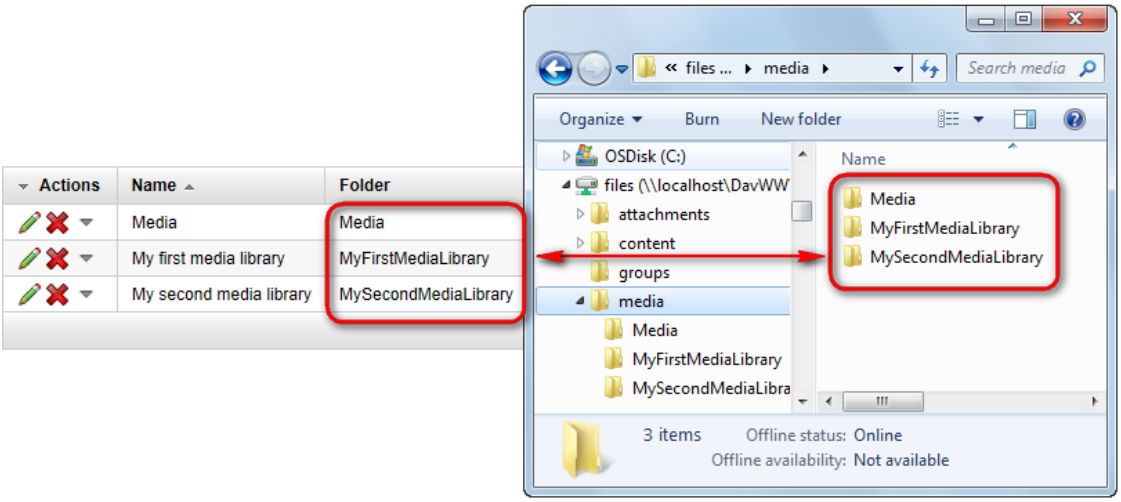
The following document permissions need to be granted to users (their roles) in order to perform respective actions with *CMS.File* documents:

Action	Required permission
Open a file in read-only mode	Read
Upload a file	Create, while the file must have one of the extensions defined in the *Allowed extensions* value for the *FileAttachment* field of the *CMS.File* document type, or globally in *Site Manager -> Settings -> System -> Files -> Upload extensions* in case that the field is configured to inherit from settings.
Edit a file	Modify
Delete a file	Delete, while the *Allow empty value* option must also be enabled for the *FileAttachment* field of the *CMS.File* document type. Deleted files are moved to the recycle bin (in the CMS, not in Windows).

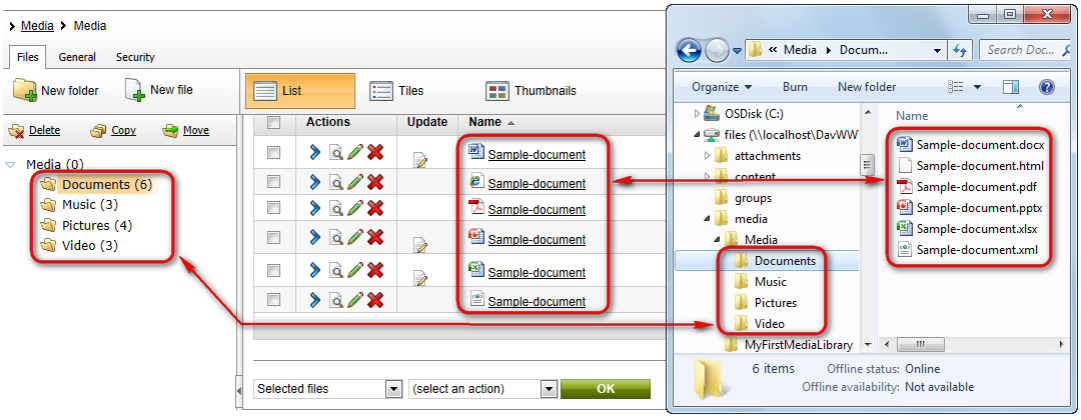
Please note that because WebDAV only works with Windows Authentication, the account used to log on to Windows needs to be imported as a user account in Kentico CMS. If you grant permissions to roles where the CMS user is member, Browse mode editing for the Windows account will be allowed.

8.52.7.5 Media

The **media** folder contains a folder for each media library on the website. The name of each folder is the same as the **Folder name** value entered when creating the respective media library.



Under each of the folders mentioned above, structure of the particular library is reflected the same way as in the CMS — files and folders are displayed in the same structure as in the actual library.



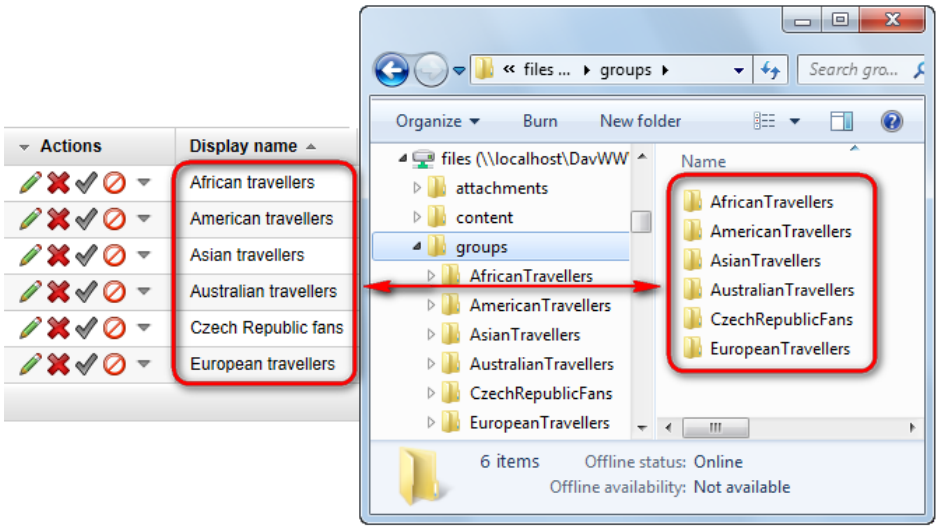
Possible actions and required permissions

Appropriate permissions need to be granted to users (their roles), either globally for the **Media libraries** module or separately for each particular library. Permissions required for individual actions are listed in the table in Modules -> Media libraries -> Security -> Media library permissions, while only the **Files** and **Folders** action groups are relevant for WebDAV Browse mode.

Please note that because WebDAV only works with Windows Authentication, the account used to log on to Windows needs to be imported as a user account in Kentico CMS. If you grant permissions to roles where the CMS user is member, Browse mode editing for the Windows account will be allowed.

8.52.7.6 Groups

The **groups** folder contains a sub-folder for each group on the website.



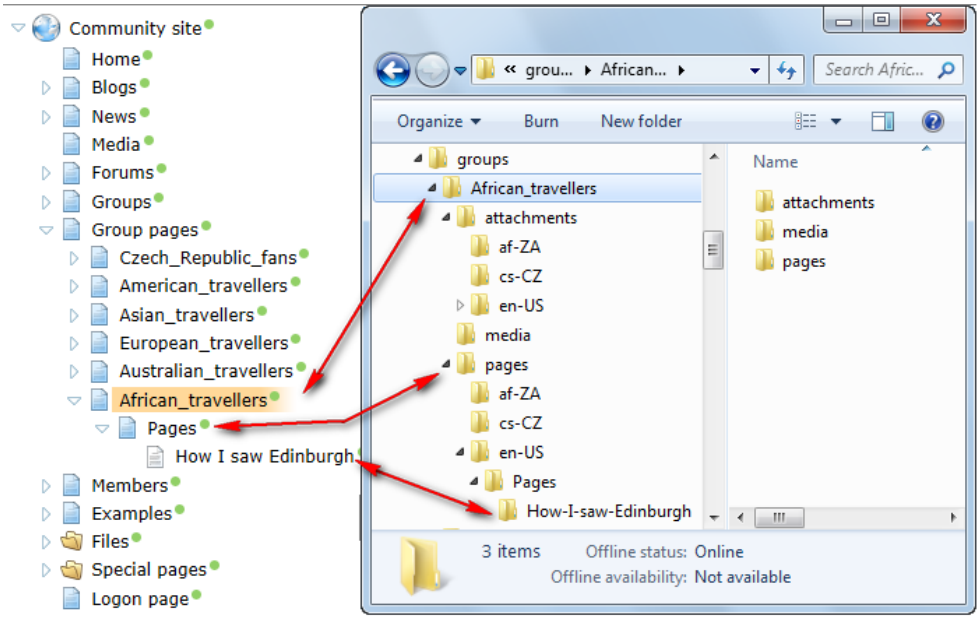
Under each group folder, you can find three folders: attachments, **pages** (similar behavior as the content folder in the network drive root) and media. The content and behavior of these folders is identical to the three folders of the same name that are located in the WebDAV network drive root. Click the folder names above to get redirected to pages dedicated to their descriptions. The following text will describe the main differences applied in this section.

Group pages attachments

The **attachments** folder contains a sub-folder for each website culture. Under each culture, you can find folders representing pages in the group pages location (configurable in **Site Manager -> Settings -> Community -> Group pages location**) that belong to the respective group.

In the screenshot below, you can see that the **en-US** folder actually represents the *en-US* version of the **African_travellers** document. The **[_unsorted]** folder contains unsorted attachments of the

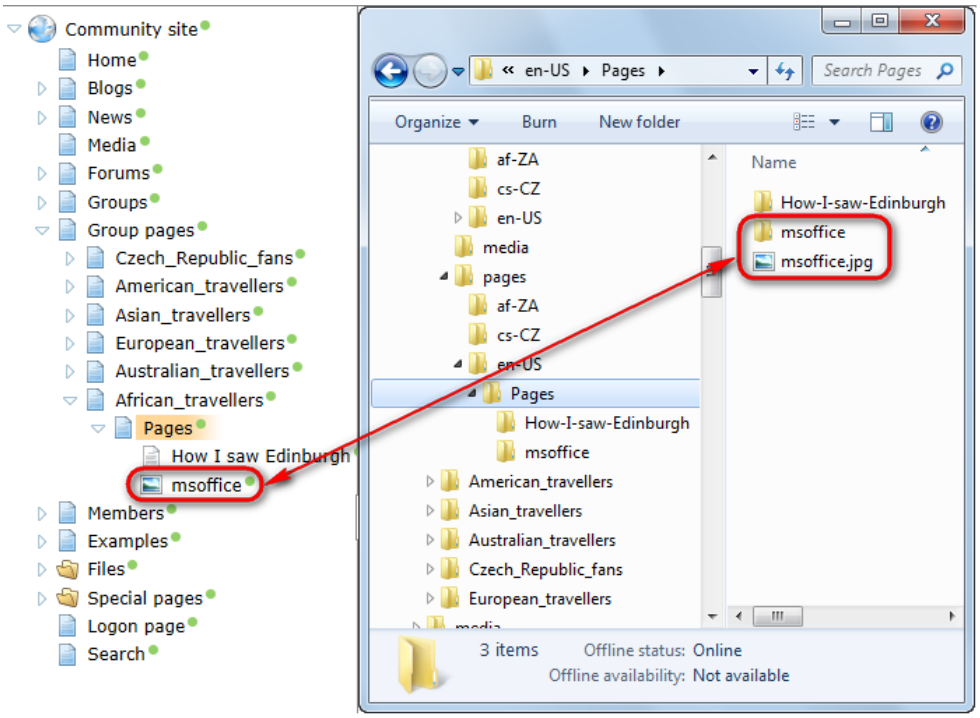
African_travellers document, while the **[MenuItemTeaserImage]** folder represents the **Menu item teaser image** file field of the **African_travellers** document. The same applies to the documents under it.



Group pages content

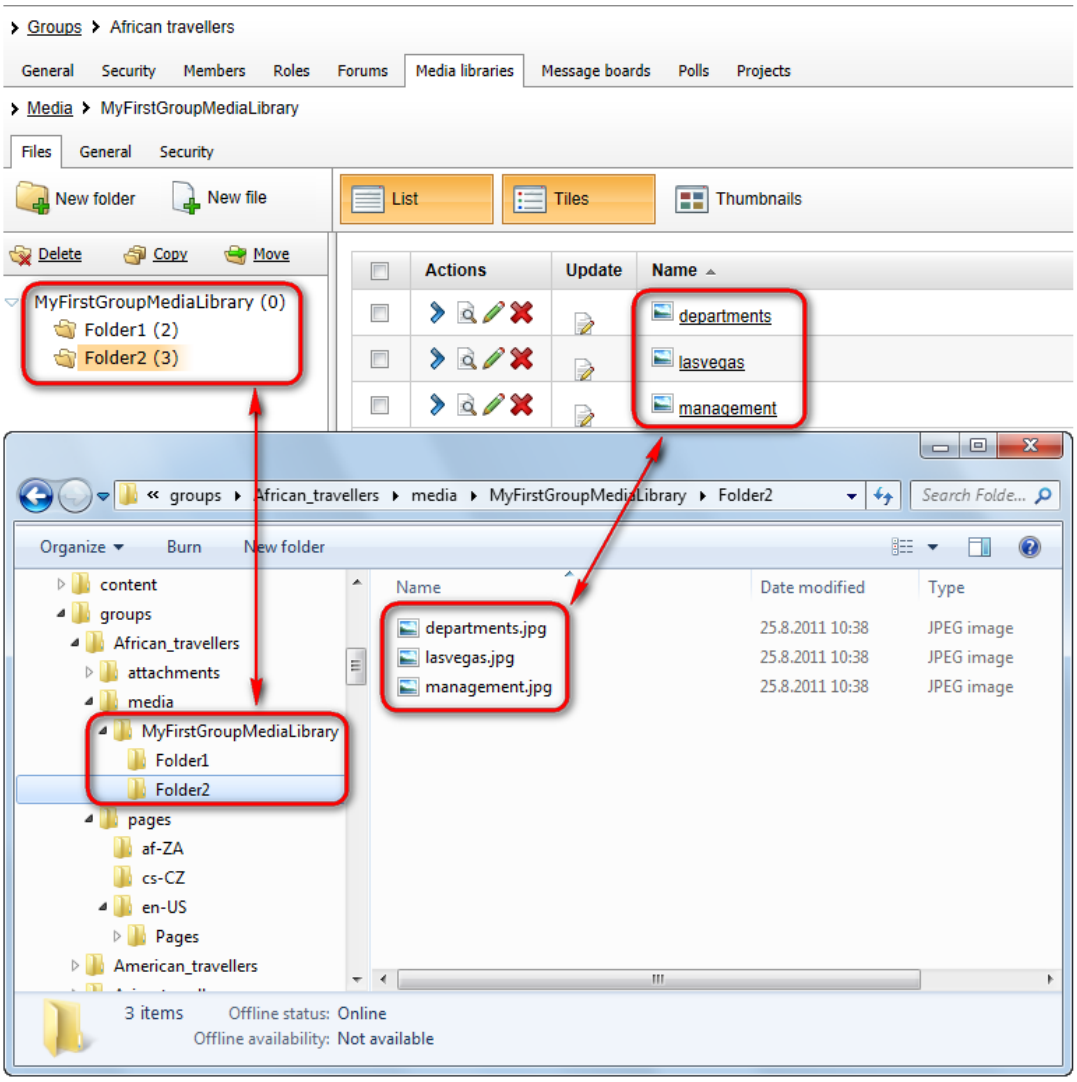
The **content** folder also contains a sub-folder for each website culture. Similarly to the content folder in the WebDAV network drive root, these folders are used to provide WebDAV editing of *CMS.File* documents. The only difference is that only documents belonging to the group found in the group pages location (configurable in **Site Manager -> Settings -> Community -> Group pages location**) can be found in each culture folder.

In the following screenshot, you can see the **msoffice.jpg** file in the content tree under the **Pages** document, as well as in **Windows Explorer** under the **Pages** folder. It is there twice. Once as a file, which represents the actual **.jpg** file uploaded in the document's file field, and as a folder, for the case that it had some child documents that would then be located under it.



Group media libraries

The **media** folder contains one folder for each media library of the respective group. Under the folder, the structure of the media library is reflected identically to the structure showed by the CMS — folders represent folders inside the library, while files represent the actual files stored in them.



Possible actions and required permissions

Only group administrators, community administrators or global administrators are allowed to browse the **groups** folder. Only those groups where the user is an administrator are displayed to them as a folder under the **groups** folder (community and global administrators are able to see all group folders). Therefore, only files of the appropriate groups can be edited by them.

Please note that permission requirements described in the Attachments, Content and Media topics apply in the respective folders in the network drive root. Therefore, users who do not belong to those mentioned in the previous paragraph can still get to the group files via these folders if they have the

required permissions.

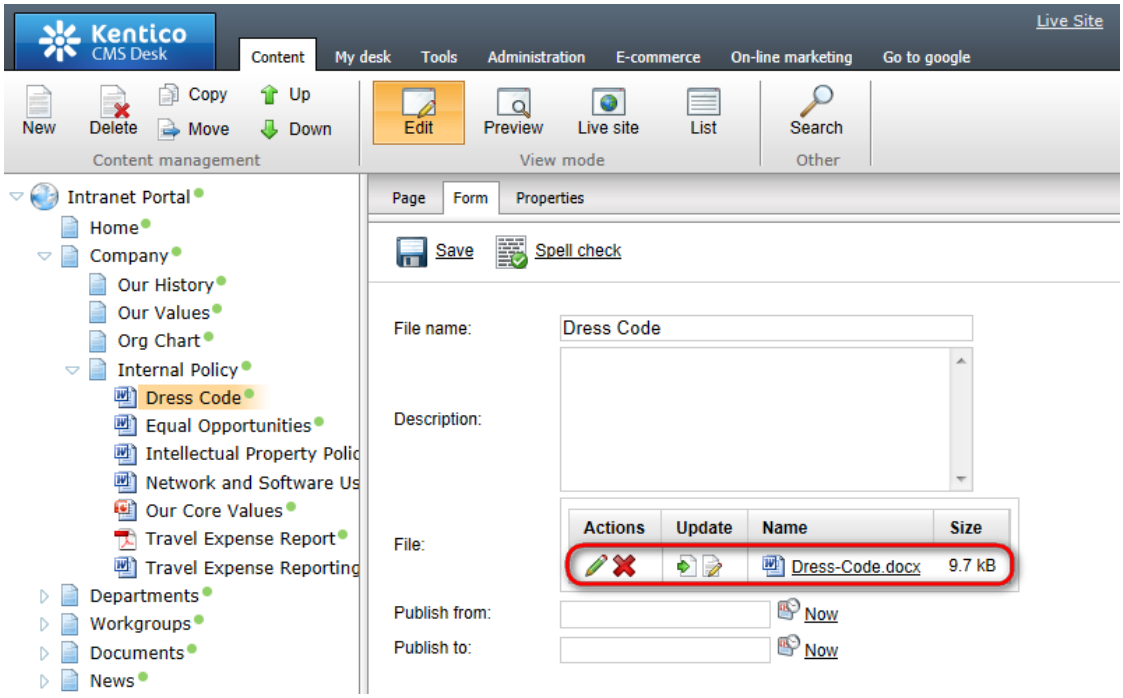
8.52.7.7 Example of Browse mode editing

The following example demonstrates how website file management using WebDAV Browse mode actually works. The sample Intranet Portal website is used for the purpose of the example.

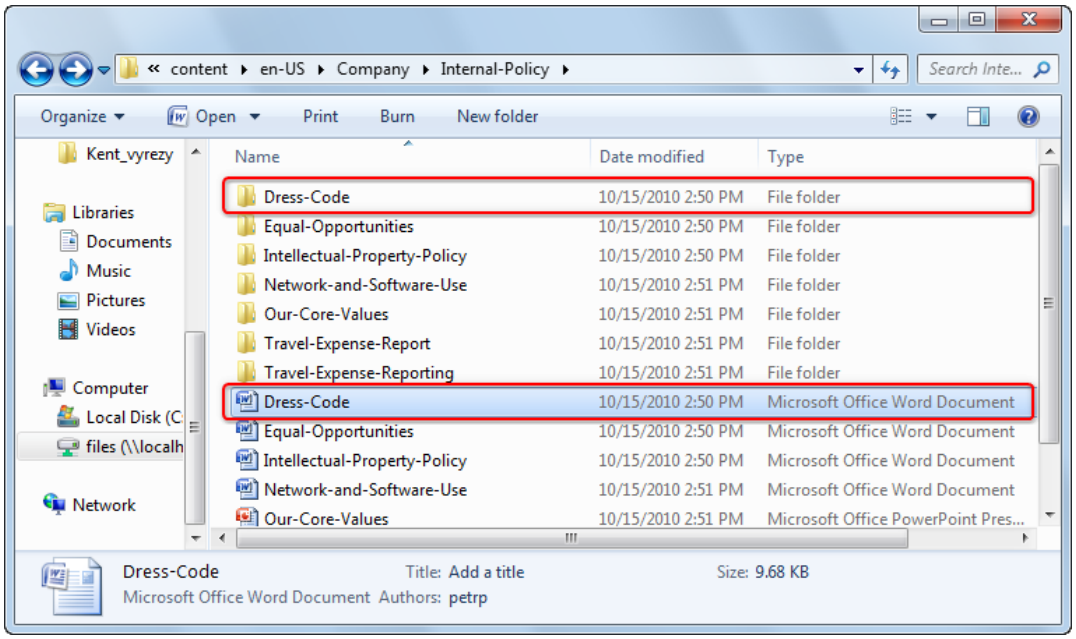
Before trying this example on your machine, please make sure that your system meets all requirements, that it is configured as described in the Configuration for WebDAV topic, that you have WebDAV settings adjusted properly and that you have mapped a WebDAV network drive.

Editing a CMS.File document

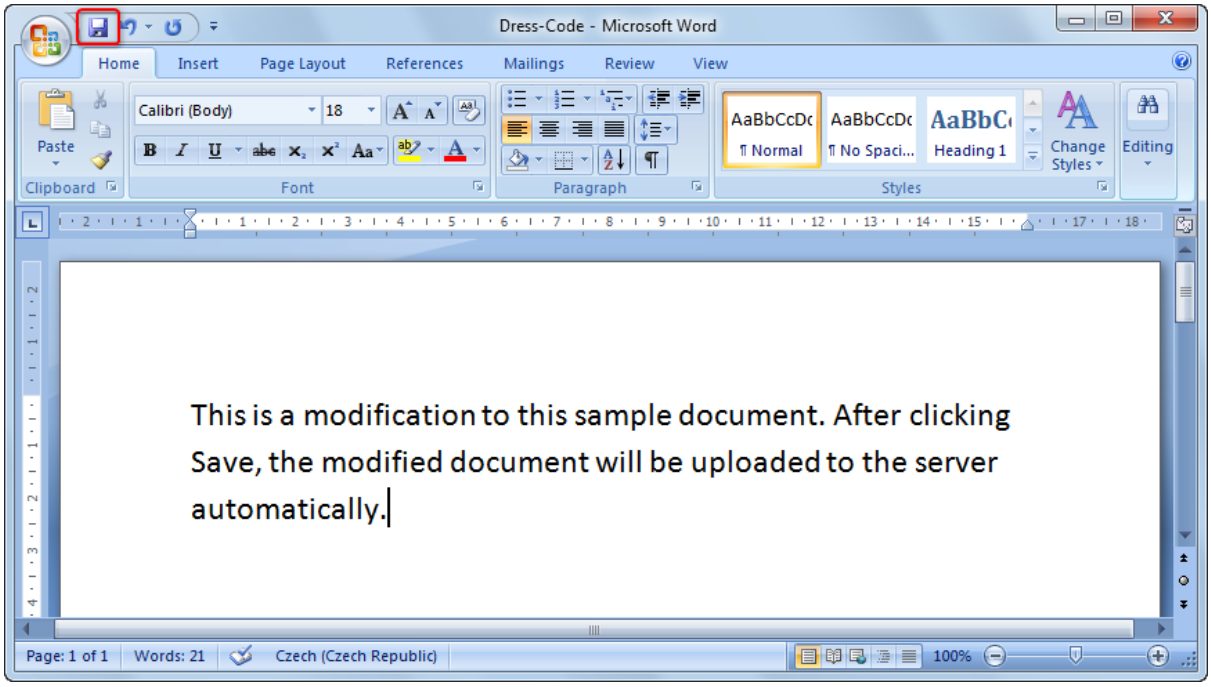
1. Go to **CMS Desk** and expand the **Company/Internal Policy** node in the content tree. From the documents under it, select **Dress Code** and switch to its **Form** tab. As it is a *CMS.File* document, you can see that it has a **Dress-Code.docx** file uploaded in its File field.



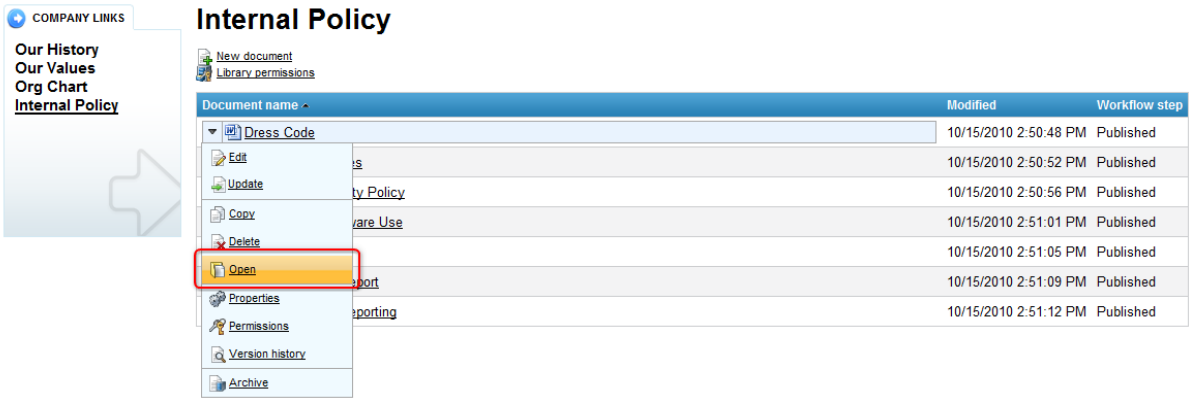
2. Let's take a look at the document on our network drive. Open **Windows Explorer** or any other file manager that you prefer. Open the WebDAV network drive and browse to *content\<your culture>\Company\Internal-Policy*. As you can see in the screenshot below, the **Dress-Code** document is represented twice in the folder — first as the **Dress-Code** folder and second as the **Dress-Code.docx** file. The folder represents the document in the content tree and would contain unsorted attachments if the document had some. The file represents the **Dress-Code.docx** file uploaded in the document's File field. Open the **Dress-Code.docx** file to see how it can be edited.



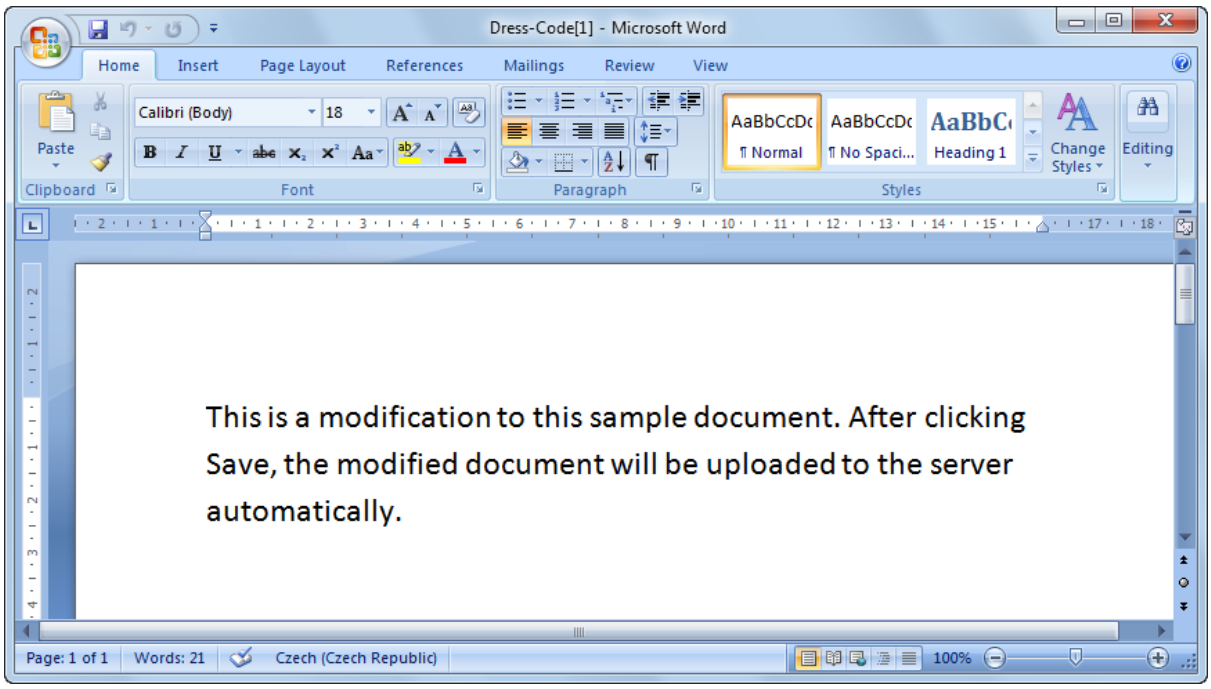
3. The **Dress-Code.docx** file gets opened in a client application (typically in Microsoft Office Word). Try editing the opened file and save your changes using the **Save** button.



4. When you have your modifications saved, try opening the file either from the UI or from the live site.

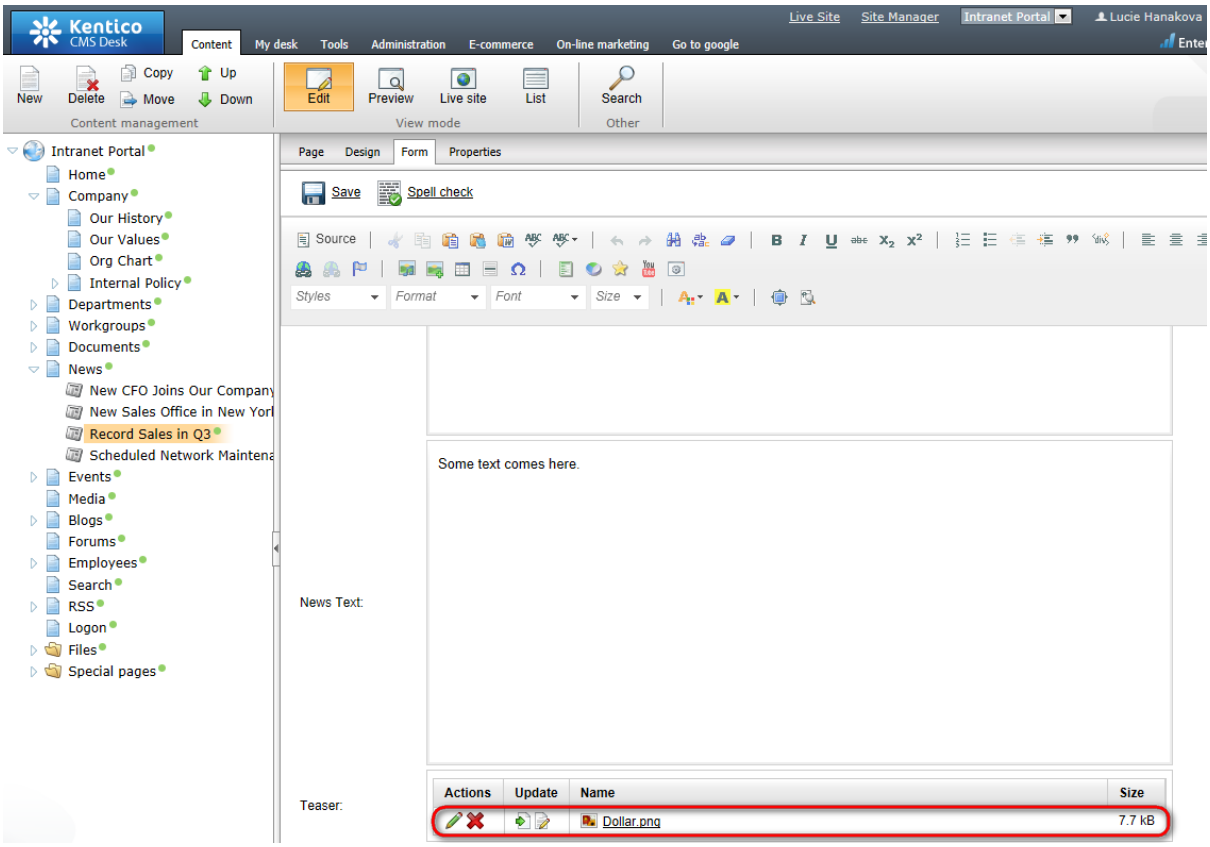


5. You should see that the opened file contains the modifications that you made to it.

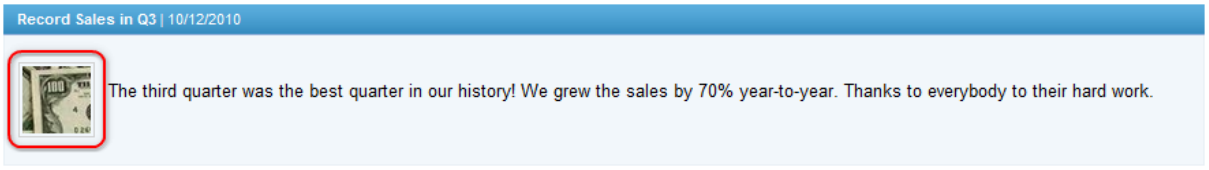


Updating a file in a File field

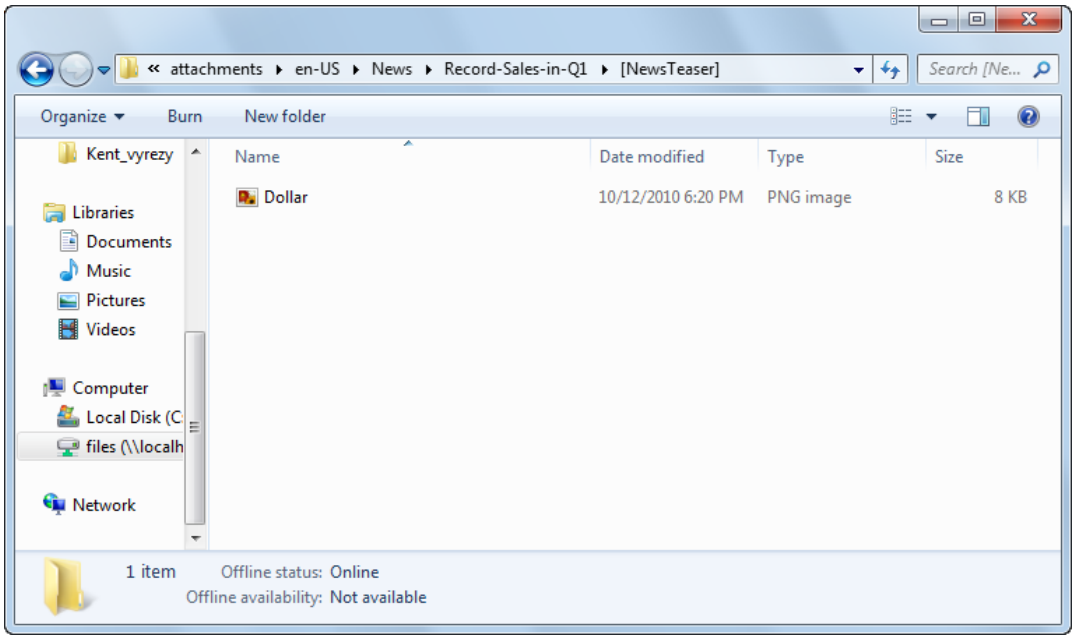
6. Next, we will try out how file uploaded in File fields can be replaced with other files. For this purpose, go back to **CMS Desk** and choose the **News/Record Sales in Q3** document in the content tree. On its **Form** tab, you can see a file uploaded in the **Teaser** File field.



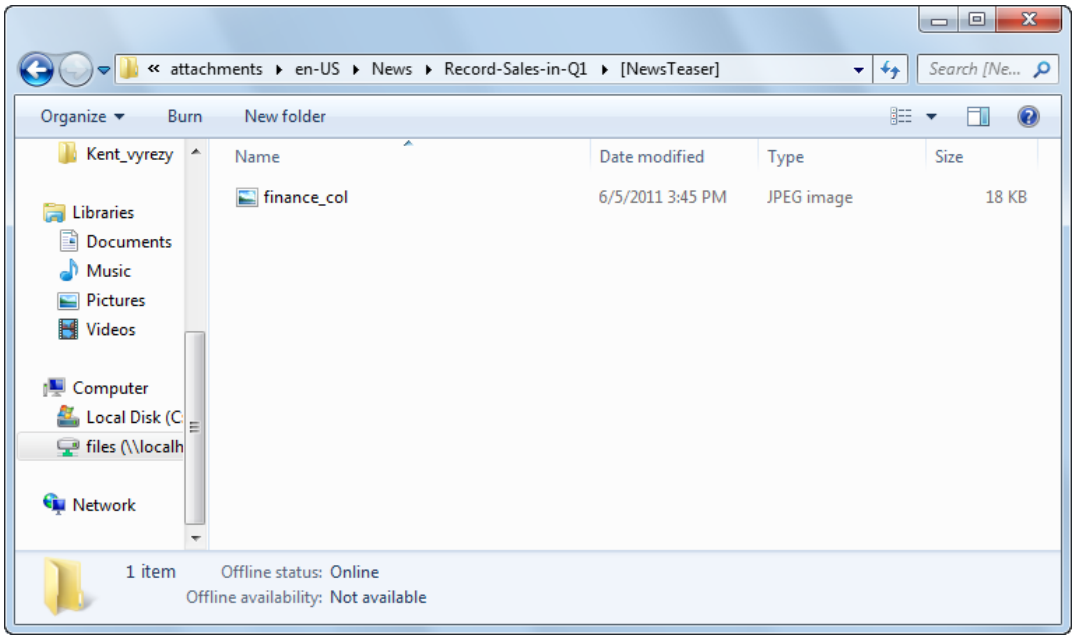
7. This file is displayed in the summary of the News item on the live site.



8. Open the WebDAV network drive and navigate to *attachments\en-US\News\Record-Sales-in-Q3\[NewsTeaser]*. You should see that the original **Dollar.png** file in the folder.

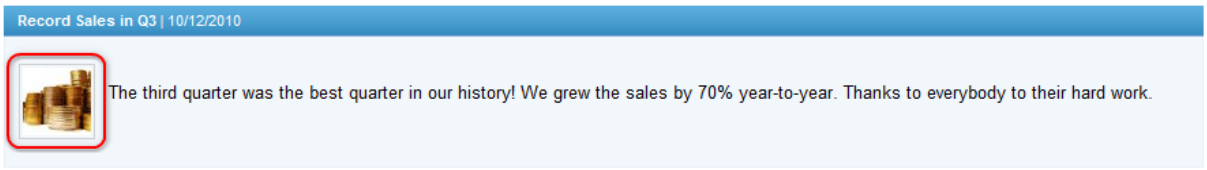


8. Try copying another image file into the folder. If you refresh the content of the folder (press F5) after a few seconds, you should see that the original file disappeared. This means that the original file in the File field has been replaced with the newly uploaded one.



9. The newly uploaded file should now also be visible in the UI ...

10. ... and on the live site.



8.52.7.8 Browse mode limitations and specifics

The following text lists specific behavior of the system when WebDAV Browse mode is used. These specifics are by design and can't be influenced or solved by a workaround.

Maximal path length

The maximal path length in Windows file systems is 250 characters. Therefore, if you have deep indentation in your content tree, folders and files that would have the path to them longer than 250 characters are not displayed on the network drive.

Windows cache delays

As explained in step 5 of the Example of Browse mode editing topic, you may experience delays when updating files in grouped attachment fields or file fields. The original file will be present along with the new one for about 1 minute. If you refresh displayed content of the folder after the interval, the file should disappear.

The same problems may be experienced when deleting a file without appropriate permissions. The file will not actually be deleted, but it will not appear in the folder for the 1 minute interval until Windows cache is refreshed.

Zero file-length limitations

It is not possible to upload or create document attachments and files using the file field based on files that have zero file length. On attempt to do so, the following behavior can be expected by-design:

When uploading a zero-length file under the **attachments** folder, a temporary attachment is created. It is not visible in the UI or on the network drive. If your try to upload a file to the same folder, an error message saying that the file already exists will be displayed (asking you if you want to overwrite it).

When uploading a zero-length file under the **content** folder, a standard *CMS.File* document is created, while it only has a temporary attachment in the file field. The document can be seen both in the UI (as a document in the content tree) and on the network drive (as a folder which can't be deleted using Browse mode). The temporary attachment can't be seen in either of the two. If you try to upload a file to the same folder, an error message saying that the file already exists will be displayed (asking you if you want to overwrite it).

Please note that in WebDAV Edit mode, uploading of zero-length files is possible without any problems or limitations.

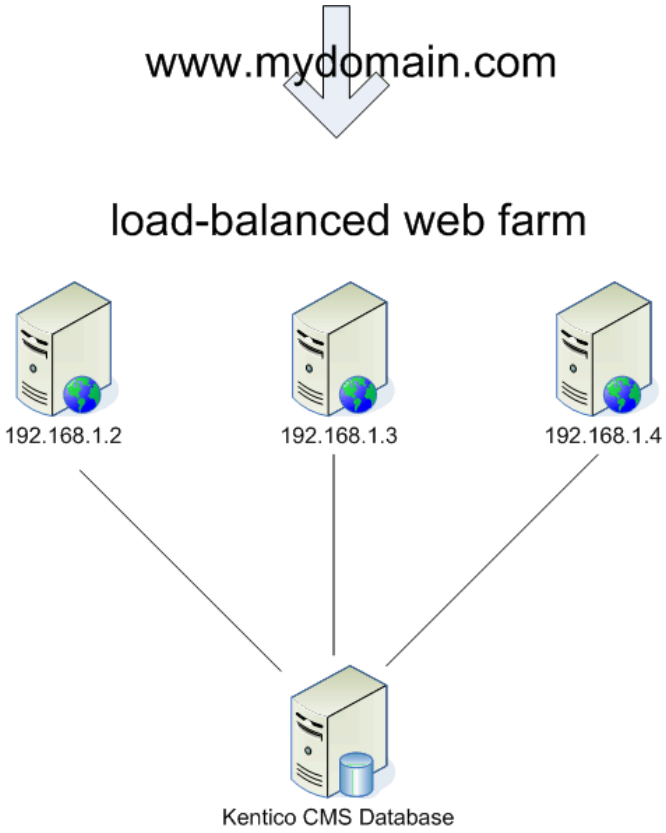
8.53 Web farm synchronization

8.53.1 Overview

Native web farm support in Kentico CMS provides the following features:

- It synchronizes the changes made to the site settings or content on one of the servers to all the other servers.
- It synchronizes the files uploaded to the site to all other servers. This is used only if you store the uploaded files on the disk or on both disk and in the database.

The following image shows the structure of the web farm and how the synchronization works:



If you change some settings or upload a file using server 192.168.1.2, the other servers do not know about it in a standard scenario. However, if you're using the **Web farm synchronization** module, it automatically creates a new synchronization task in the database and notifies the other servers so that they process their synchronization task. To learn more about how the synchronization works, refer to the Synchronization mechanisms topic.

To learn how to add web farm servers into the system and configure them, please see the Defining web farm servers topic.

Developers can write custom tasks to be used by this module. The Creating custom web farm synchronization tasks topic contains more information.

The Web farm synchronization internals and API sub-chapter provides information about the database

tables and classes used by the module and examples of how web farm synchronization can be managed using the API.

Using Kentico CMS on a web farm without the Web farm synchronization module

You can use Kentico CMS on a web farm even if you do not use the web farm synchronization module, especially if you do not store uploaded files in the file system. Then, the only limitation is that if you change the settings or page content on one of the servers, the other servers may keep using the old version of the settings in their memory/cache until the web application is restarted or cache content expires.

Please note: The web farm support **doesn't replace any load-balancing or web farm management tools**.

SSL in a web farm environment

If you use an SSL offload device or accelerator as part of your web farm, you may encounter problems with a redirection loop if your website is configured to require SSL for the administration interface or on specific pages.

For this type of scenario, it is necessary to add some custom code to your website, as described in the SSL (HTTPS) support topic in the **Membership, permissions and security -> Security** chapter of this guide.

8.53.2 Synchronization mechanisms

This topic describes the process of propagating changes in a web farm.

When you make changes to content on a server, those changes get logged into the database as a web farm task. The server then notifies other servers, which retrieve the task data from the database and make appropriate changes in their file system or memory. There are two ways servers can get notified about changes in the web farm.

URL notifications

Every Kentico CMS instance contains a page that a web farm server can send requests to when it generates one or more web farm tasks. The other web farm servers don't get involved in the process until they are notified about the changes. When this happens, each server fetches its tasks and processes them accordingly.

This is the default behavior in instances not running on Windows Azure.

Database notifications

In this approach, a column in the **CMS_WebFarmServer** database table is used. The column contains a time stamp, which indicates the time of the last change made on a server. On the application side, a

routine runs in a separate thread which continuously polls the database to find out whether a time stamp changed.

To enable this mechanism, add the following key to your web.config file. You don't need this key if your application runs on Windows Azure, as this mechanism is used on Azure by default.

```
<add key="CMSWebFarmUseDBUpdater" value="true" />
```

8.53.3 Defining web farm servers

You need to enter the **internal** URLs of all your servers into the system so that the web farm synchronization module knows which servers should be synchronized.

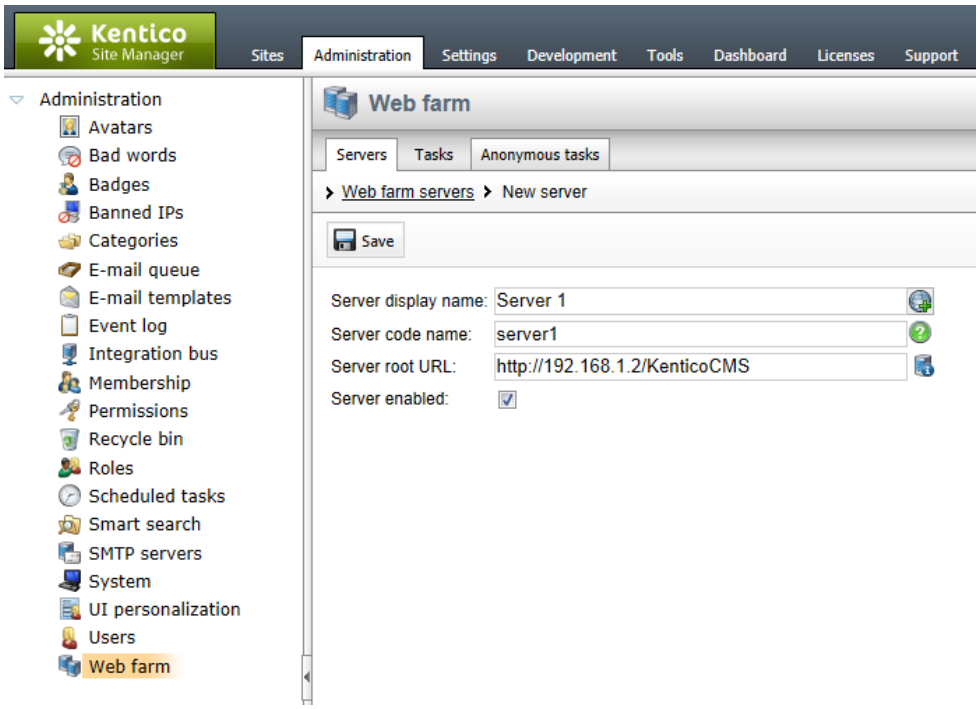
To add a server to the web farm:

1. Go to **Site Manager -> Administration -> Web farm**.
2. Click **New server**.
3. Fill in the following fields:

- **Server display name** - a descriptive name for the server displayed in the administration interface.
- **Server code name** - a unique code name of the server that will be used in the server's configuration file.
- **Server root URL** - the URL of the root of the website on the server, such as *http://192.168.1.2/KenticoCMS.* You can check the availability of the server by clicking *Check server availability* ().
- **Server enabled** - this checkbox may be used to manually enable or disable web farm synchronization for the server.

4. Click **Save** to register the server.

Repeat the process for every server in your web farm.



You also need to update the web.config file on each server and add the **CMSWebFarmServerName** key into the *appSettings* section:

```
<add key="CMSWebFarmServerName" value="<servername>" />
```

Where <*servername*> is the **code name** that the system created for the server (or the code name that you manually entered). Every server must contain only one such key with its own name.

After you have defined your web farm servers, you need to enable the web farm module itself in the settings:

1. Navigate to **Site Manager -> Settings -> Versioning & Synchronization -> Web farm**.
2. Turn on **Enable web farm**.
3. Click **Save**.

You can perform additional low-level settings of web farm synchronization by adding the keys listed in Web farm synchronization settings into the ***/configuration/appSettings*** section of your web.config file.

Web farm license keys

Please be sure to enter an appropriate license key for the internal URLs of the web farm servers.

For example, if you use the web farm for domain name ***example.com*** and access the servers internally through URLs like:

- http://192.168.1.2
- http://192.168.1.3
- ...

Enter separate license keys for ***example.com***, ***192.168.1.2*** and ***192.168.1.3*** in **Site Manager -> Licenses**. You only need to add the licenses on one server and restart the other instances using **Site Manager -> Administration -> System -> Restart application**.

Configuring servers for synchronizing macros

The system uses signatures to ensure the security of macro expressions. Macro signatures contain the user name of the macro's author and a hash of the given expression.

The hash function used to create the signatures appends a salt to the input. To ensure that macro expressions work correctly in a web farm environment, you need to configure all servers to use the same hash salt:

- Add the **CMSHashStringSalt** key to the *appSettings* section of the web.config file on all web farm servers. You can use any string as the value, but the salt should be random and at least 16 characters long. For example, a randomly generated GUID is a strong salt:

```
<add key="CMSHashStringSalt" value="e68b9ad6-a461-4707-8e3e-ece73f03dd02" />
```

The best option is to set the hash salt value before you start creating content for your website. Changing the salt causes all current hash values to become invalid. To fix existing macro expressions in the system after changing the hash salt, you need to re-sign the macros. See Macro security for more information.

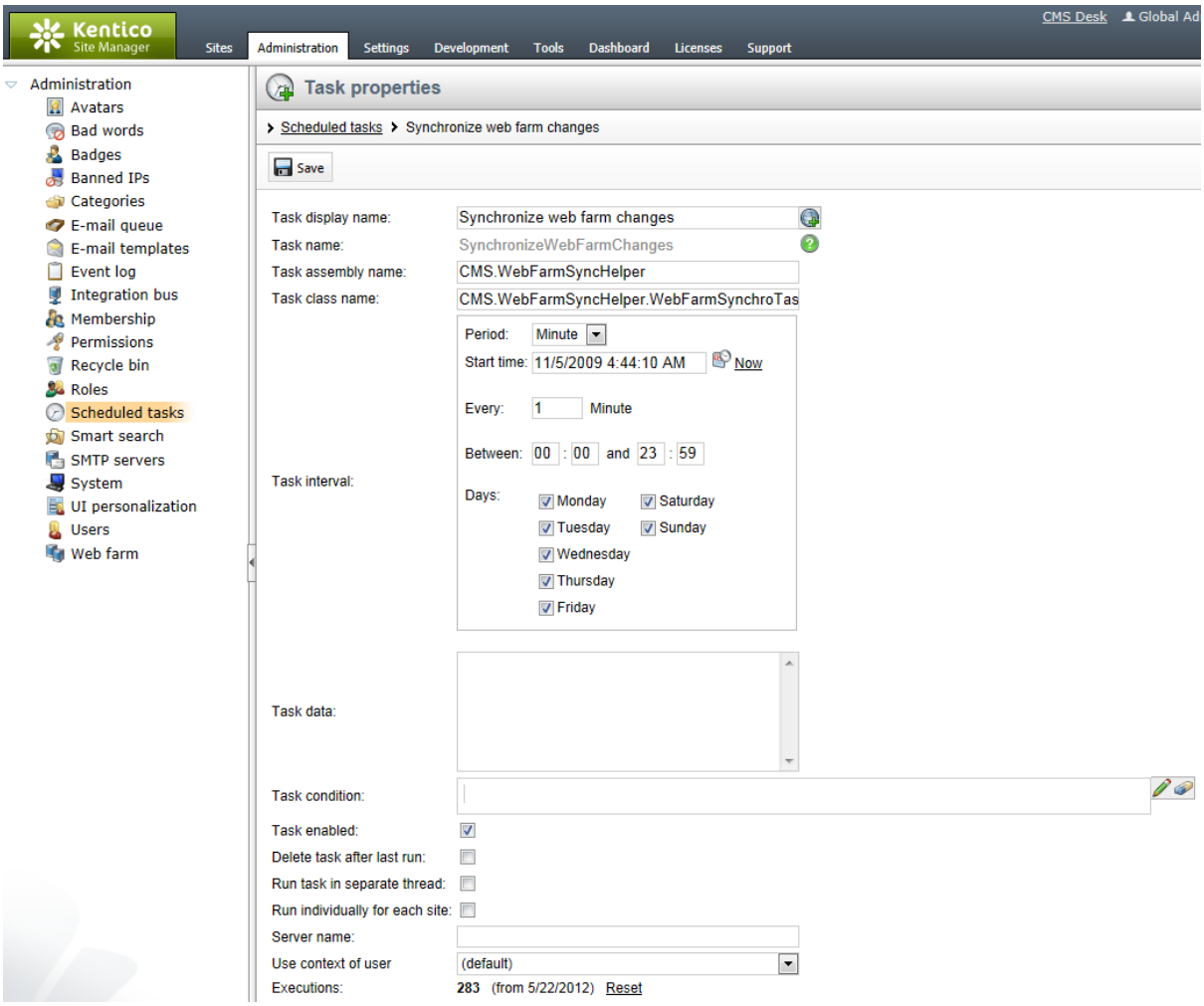
Setting the server synchronization interval

By default, web farm servers are updated with changes made on other servers once per request. Synchronizing this frequently may be impractical for high-traffic websites. To change the synchronization interval, set the value of the **CMSWebFarmUpdateWithinRequest** web.config key to false:

```
<add key="CMSWebFarmUpdateWithinRequest" value="false"/>
```

and use the **Synchronize web farm changes** scheduled task instead, which has a configurable execution interval:

1. Go to **Site Manager -> Administration -> Scheduled tasks**.
2. Select *(global)* in the **Site** selector at the top of the page.
3. Edit () the **Synchronize web farm changes** task.
4. Set an appropriate **Task interval**.
5. Check the **Task enabled** box.
6. Click  **Save**.



You also need to create the task for each server in the web farm. To create the tasks quickly:

Note: All web farm servers that you wish to use need to be registered in the system before performing the following steps.

1. Return to the list of scheduled tasks.
2. Click **New task**.
3. Fill in the same properties as those of the **Synchronize web farm changes** task (the **Task name** must be different)
4. Check **Create tasks for all web farm servers** below the **Server name** property.
5. Click **Save** to complete the process.

Your web farm now synchronizes according to the interval of the scheduled tasks.

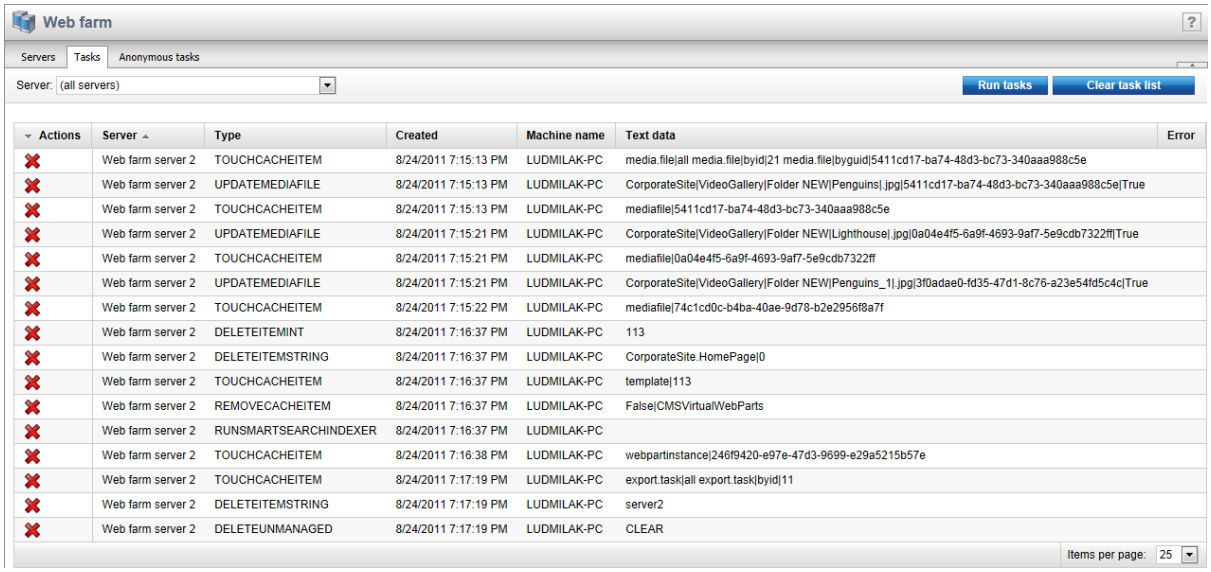
Monitoring web farm synchronization tasks

If synchronization doesn't work as expected, you can check the **Tasks** tab of the web farm administration interface. Here, you can view information about all synchronization tasks that are currently active (waiting to be processed).

You can filter the tasks using the **Server** drop-down list according to the server to which they are assigned, or select **(all servers)** to show all synchronization tasks in the entire web farm.

Click **Run tasks** to manually execute tasks, regardless of how the synchronization interval is currently configured.

You can remove all listed tasks using the **Clear task list** button, or individual tasks via the **Delete** ( ) action. Clearing the list is not necessary if the web farm is working correctly — the system automatically removes successfully processed tasks.



A similar user interface can be found on the **Anonymous tasks** tab. This tab displays a list of currently active (waiting to be processed) anonymous synchronization tasks. These tasks are logged by external applications (e.g. Windows services) to ensure that changes made by the external application are reflected in the web application cache. Tasks are logged as anonymous only if the application is NOT configured to run in a web farm. If it is configured to run in a web farm, these tasks are logged as standard synchronization tasks on the **Tasks** tab.

Creating web farm servers automatically

An alternative way of defining web farm servers is letting the system create them automatically on application start. You need to have web farm enabled via a setting or a web.config key on all the servers. Then add the following key

```
<add key="CMSWebFarmGenerateServers" value="true"/>
```

or turn on the following setting: **Versioning & Synchronization -> Web farm -> Generate servers dynamically**.

All servers with enabled web farm support will add themselves into the list of servers in Administration -> Web farm when the application starts. If a server doesn't have a server name defined in the web.config, it uses its machine name.

8.53.4 Creating custom web farm synchronization tasks

You can write custom tasks to be executed by web farm servers. This is done by adding the code of your tasks into the **App_Code** folder of your web project (or **Old_App_Code** if you installed Kentico CMS as a web application).

The following example shows how to create a custom synchronization task that logs information events into the event log of all servers in the web farm:

1. Open your web project in Visual Studio, expand the **App_Code** folder and add a new class into it called **WebFarmTaskLoader.cs**.

2. Add the following references:

[C#]

```
using CMS.SettingsProvider;
using CMS.EventLog;
using CMS.WebFarmSyncHelper;
```

3. Delete the default class declaration and its content. Instead extend the **CMSModuleLoader** partial class and define a new attribute for it as shown below:

[C#]

```
[WebFarmTaskLoader]
public partial class CMSModuleLoader
{
    /// <summary>
    /// Module registration
    /// </summary>
    private class WebFarmTaskLoaderAttribute : CMSLoaderAttribute
    {
        ...
    }
}
```

4. Enter the following code into the **WebFarmTaskLoaderAttribute** class:

[C#]

```
/// <summary>
/// Called automatically when the application starts
/// </summary>
public override void Init()
{
    // Hook the custom task event
    WebSyncHelper.OnProcessCustomTask += new WebSyncHelper.TaskHandler
(WebSyncHelper_OnProcessCustomTask);
}
```

```
static void WebSyncHelper_OnProcessCustomTask(WebFarmTaskTypeEnum taskType, string
target, string data, int taskId)
{
    // Log execution
    EventLogProvider ev = new EventLogProvider();
    ev.LogEvent("I", DateTime.Now, "CustomTask", "Execute", null, data);
}
```

The override of the **Init()** method is called automatically when the application starts. The code assigns a handler for the **OnProcessCustomTask** event, which is then used to log a record with *Execute* as its *event code* into the website's event log.

5. Repeat the steps above for all web servers in the web farm, so that the *Execute* event is logged for each one. **Build** their projects if they are installed as a web application.

6. Select one web farm server and expand the **Init()** method according to the following:

[C#]

```
/// <summary>
/// Called automatically when the application starts
/// </summary>
public override void Init()
{
    // Hook the custom task event
    WebSyncHelper.OnProcessCustomTask += new WebSyncHelper.TaskHandler
(WebSyncHelper_OnProcessCustomTask);


    string data = "Task from " + WebSyncHelperClass.ServerName;

    // Add your actions for the custom synchronization task
    if (WebSyncHelperClass.CreateTask(WebFarmTaskTypeEnum.Custom, "MyTask", data,
null))
    {
        // Log creation
        EventLogProvider ev = new EventLogProvider();
        ev.LogEvent("I", DateTime.Now, "CustomTask", "Create", null, data);
    }
}
```

This ensures that the custom task is created when the selected server is started and if successful, an event with *Create* as its event code will be logged for the given server.

7. Finally, save the changes (**Build** the project if you installed it as a web application). You can see the results by checking the event log for all web farm servers at **Site Manager -> Administration -> Event log**.

8.53.5 Web farm synchronization internals and API

8.53.5.1 Overview

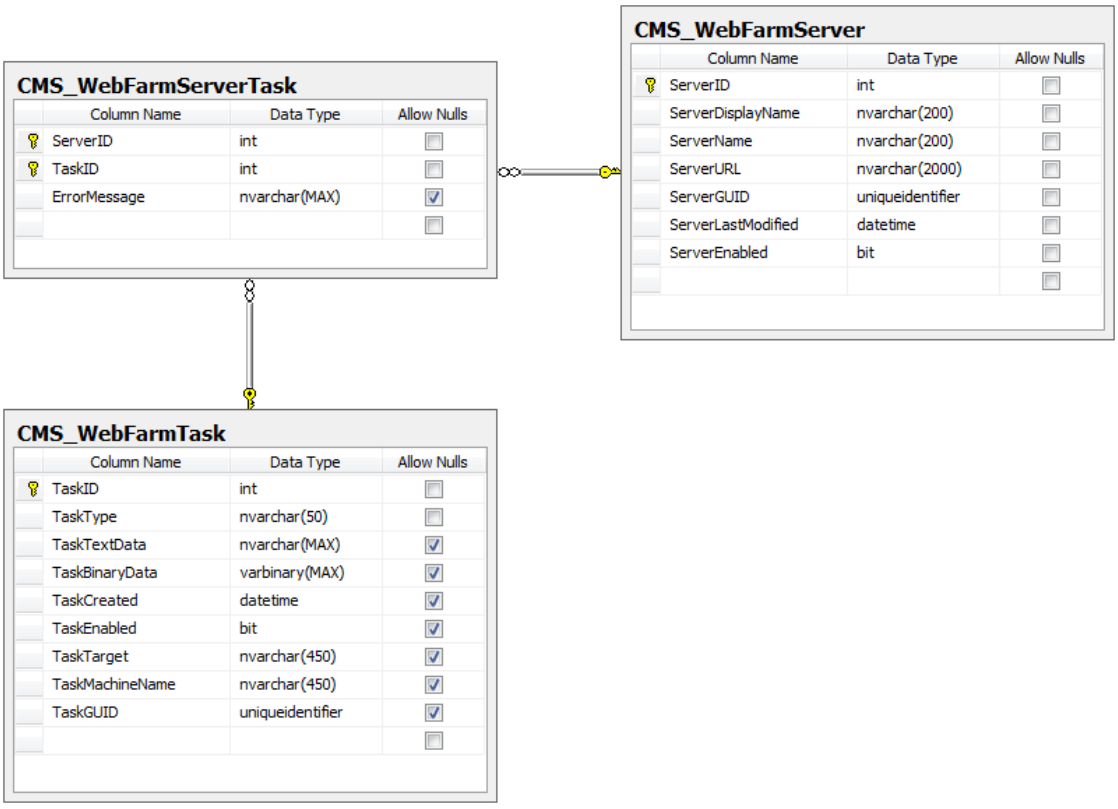
In this chapter, you will learn which database tables and API classes are used in the Web farm synchronization module. You will also see the most common API examples.

Further API-related information and examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all members of the module's classes, please refer to Kentico CMS API Reference, which is a part of your Kentico CMS installation and can be accessed through the programs folder in the Windows Start menu.

8.53.5.2 Database tables

The following database tables are used to store information for the web farm module:

- **CMS_WebFarmServer** - contains records representing web farm servers.
- **CMS_WebFarmTask** - stores web farm synchronization tasks.
- **CMS_WebFarmServerTask** - stores relationships between web farm servers and synchronization tasks. Each entry indicates that a task should be processed by a server. Successfully completed tasks are automatically deleted from the table.



8.53.5.3 API classes

If you are not familiar with database table data management through Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

Please note

The main classes of the web farm module can be found in the **CMS.WebFarmSync** namespace.

CMS_WebFarmServer table API:

- **WebFarmServerInfo** - represents one web farm server.
- **WebFarmServerInfoProvider** - provides management functionality for web farm servers.

CMS_WebFarmTask table API:

- **WebFarmTaskInfo** - represents one web farm synchronization task.
- **WebFarmTaskInfoProvider** - provides management functionality for synchronization tasks.

Other classes:

- **CMS.SettingsProvider.WebSyncHelperClass** - allows the creation of synchronization tasks and the management of general web farm settings.
- **CMS.WebFarmSyncHelper.WebSyncHelper** - provides functionality for processing web farm synchronization tasks.

8.53.5.4 API examples

8.53.5.4.1 Overview

These topics show examples of how the API of the Web farm module can be used:

- Managing web farm servers
- Managing synchronization tasks

Please note

All API examples can be simulated on your Kentico CMS website through the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Administration\WebFarm\Default.aspx.cs***.

The web farm API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.GlobalHelper;
using CMS.WebFarmSync;
```

```
using CMS.WebFarmSyncHelper;
using CMS.SettingsProvider;
```

8.53.5.4.2 Managing web farm servers

The following example creates a web farm server.

```
private void CreateWebFarmServer()
{
    // Create new web farm server object
    WebFarmServerInfo newServer = new WebFarmServerInfo();

    // Set the properties
    newServer.ServerDisplayName = "My new server";
    newServer.ServerName = "MyNewServer";
    newServer.ServerEnabled = true;
    newServer.ServerURL = "http://localhost/KenticoCMS";

    // Save the web farm server
    WebFarmServerInfoProvider.SetWebFarmServerInfo(newServer);
}
```

The following example gets and updates a web farm server.

```
private bool GetAndUpdateWebFarmServer()
{
    // Get the web farm server
    WebFarmServerInfo updateServer =
WebFarmServerInfoProvider.GetWebFarmServerInfo("MyNewServer");
    if (updateServer != null)
    {
        // Update the properties
        updateServer.ServerDisplayName = updateServer.ServerDisplayName.ToLower();

        // Save the changes
        WebFarmServerInfoProvider.SetWebFarmServerInfo(updateServer);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates web farm servers.

```
private bool GetAndBulkUpdateWebFarmServers()
{
    // Get the data
```

```
    DataSet servers = WebFarmServerInfoProvider.GetAllEnabledServers();
    if (!DataHelper.DataSourceIsEmpty(servers))
    {
        // Loop through the individual items
        foreach (DataRow serverDr in servers.Tables[0].Rows)
        {
            // Create object from DataRow
            WebFarmServerInfo modifyServer = new WebFarmServerInfo(serverDr);

            // Update the properties
            modifyServer.ServerDisplayName =
modifyServer.ServerDisplayName.ToUpper();

            // Save the changes
            WebFarmServerInfoProvider.SetWebFarmServerInfo(modifyServer);
        }

        return true;
    }

    return false;
}
```

The following example deletes a web farm server.

```
private bool DeleteWebFarmServer()
{
    // Get the web farm server
    WebFarmServerInfo deleteServer =
WebFarmServerInfoProvider.GetWebFarmServerInfo("MyNewServer");

    // Delete the web farm server
    WebFarmServerInfoProvider.DeleteWebFarmServerInfo(deleteServer);

    return (deleteServer != null);
}
```

8.53.5.4.3 Managing synchronization tasks

The following example creates a web farm synchronization task.

```
private void CreateTask()
{
    // Set the properties
    string taskTarget = "";
    string taskTextData = "MyWebFarmTask";
    byte[] taskBinaryData = null;
    WebFarmTaskTypeEnum webfarmTaskType = WebFarmTaskTypeEnum.ClearHashtables;

    // Create the web farm task
```

```
    WebSyncHelperClass.CreateTask(webfarmTaskType, taskTarget, taskTextData,
taskBinaryData);
}
```

The following example runs all tasks assigned to the current server (the one specified in the web.config file).

```
private void RunMyTasks()
{
    WebSyncHelper.ProcessMyTasks();
}
```

8.54 Website optimization

8.54.1 Overview

Making improvements on a website can be a difficult process, since it is often not possible to know ahead of time whether a change will have a positive effect, or which modification out of several options will bring the best results. A possible solution for these issues is optimization testing of pages.

Optimization testing allows you to create different versions of a page (or specific parts of a page) and evaluate them according to the behavior of the website's visitors. This way you can confirm which changes are actually beneficial and use the content that works best for the users who visit your website. The entire process is completely transparent for the site's visitors and does not require them to give any feedback manually. Testing is typically done for key sections of the website that receive the most traffic, such as the default home page (landing page).

There are two different techniques that you can use to optimize pages in Kentico CMS:

- A/B testing - divides incoming traffic between two or more different variants of a page. Each variant is defined as a separate document in the content tree. Results are tracked for each page variant as a whole, meaning that the combined effect of all changes made to the given page is measured.
- Multivariate testing (MVT) - allows you to make multiple modifications to the content of a single page and monitor the effect that specific changes have on the behavior of visitors.

Each type of testing has its advantages and disadvantages and is intended for different scenarios. A/B testing is usually more straightforward and easier to set up and use. It is most suitable for situations where you need to test a single variable element or a full redesign that changes the entire appearance of a page. In cases where you need to evaluate multiple variables on a single page or monitor the effect of individual modifications with a greater degree of detail, multivariate testing is the better option. Because MVT usually involves more tested variants, it may require more time (or site traffic) than A/B testing to get meaningful results.

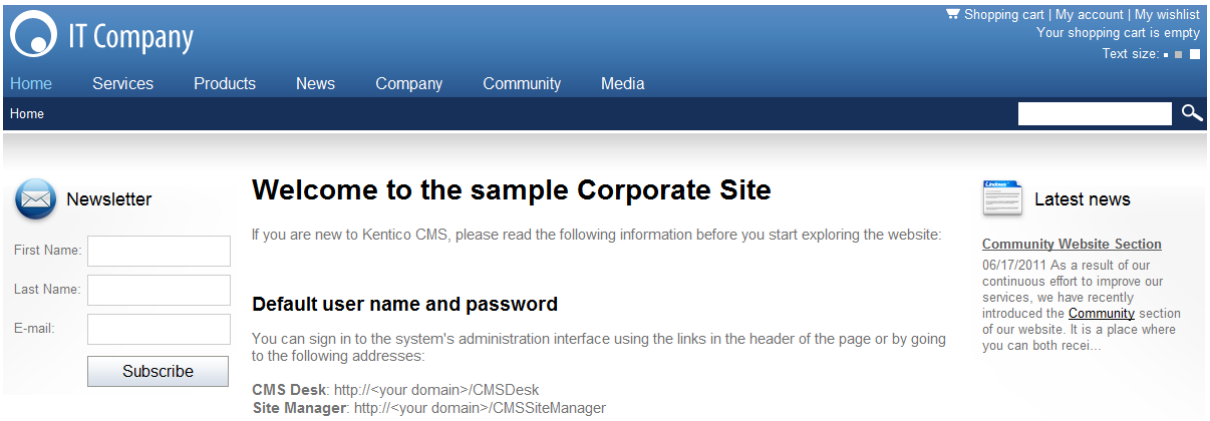
For both types of optimization testing, results are measured by tracking the activity of users after they access the tested page and view one of the different content versions. Actions that are desired from visitors are represented in the system as ***Conversions***. Typical examples of conversions are product orders, registrations, newsletter subscriptions, views of special pages etc. When a user performs the specific action tracked by a conversion, it is referred to as a ***Conversion hit***. To learn more about conversions and how they can be implemented on your website, please read the Modules -> Web analytics -> Conversions chapter.

8.54.2 A/B testing

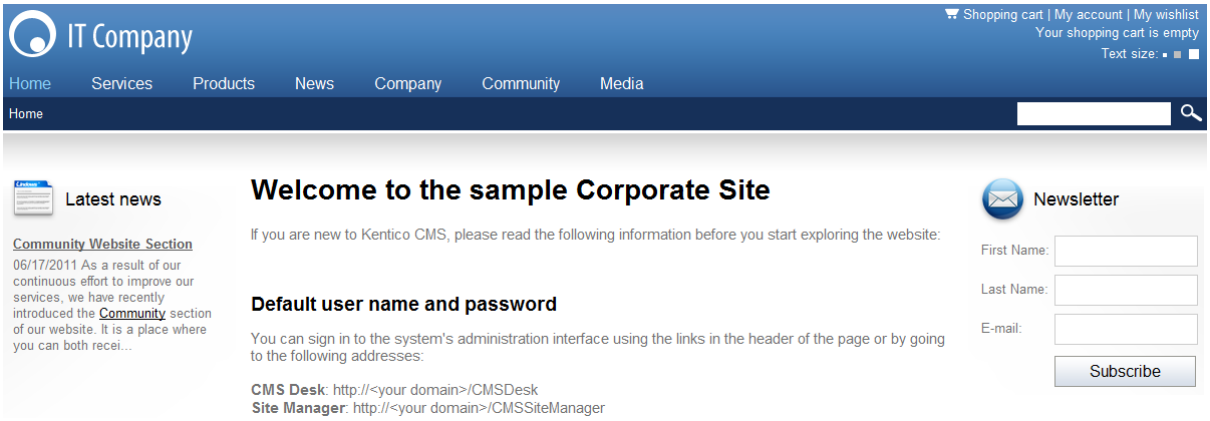
8.54.2.1 A/B testing overview

A/B testing is an on-line marketing technique used to optimize the pages of a website according to the reactions of visitors. This is achieved by creating one or more modified versions of a given page and running them all at the same time during a designated testing period. The system then divides traffic between individual page versions and tracks how the changes affect the user experience and activity of visitors.

The following images show two versions of a sample home page — the original and an alternative page with the position of the left and right content columns reversed.



Some visitors will be assigned to the second version and will see the following when they view the home page:



There are no limitations placed on the modifications that can be made to different page options. They may include anything from subtle changes to using a completely different page.

In Kentico CMS, the basic objects that provide this functionality are ***A/B tests***, which can be created for specific website pages. The page options defined for a test are called ***Page variants***. Each variant is linked to a document in the content tree of the website that can be designed and configured using the standard web development process. Usually, the ***Original page*** for which the test was created will be included as one of the variants, so that potential improvements can be compared with the baseline

statistics (those of the unmodified page).

For a practical scenario, imagine that you have an e-commerce page on your website where visitors can purchase products and you wish to fine-tune it to be more user friendly in order to increase the amount of sales. By utilizing A/B testing, you can create multiple versions of this page with any type of modifications such as a slightly altered layout, graphical design or text content. These alternative pages will then serve as variants of the A/B test. While the test is running, it will automatically display different page variants to different visitors and keep track of all product purchases as conversions. When the test is completed after a certain amount of time, you can evaluate which page variant encouraged more users to make purchases on the website and set the most successful one as the permanent page.

Please see the Creating an A/B test topic for detailed information about how an A/B test can be defined for a page. The data gathered by A/B tests is displayed using reports that allow you to view and analyze several types of conversion metrics, as described in Analyzing A/B test results.

With A/B testing, results are tracked for entire page variants, which means that the measured statistics reflect the combined effect of all changes made to each variant. If you wish to monitor how individual page modifications affect the behaviour visitors, you may use Multivariate testing, which is another website optimization feature provided by Kentico CMS.

How it works

When visitors navigate to a page that has a running A/B test defined, one of the page variants configured for the given test will be displayed to them. The variant is chosen randomly for every user. With a large enough visit sample size, each page variant should receive roughly the same amount of traffic during the course of the test.

A persistent cookie is stored in the visitor's browser, used to identify which variant was assigned by the given A/B test. The name of the cookie uses the following format: *CMSAB<A/B test code name>*. It saves the code name of the assigned page variant as its value. This cookie expires either within 30 days after the last visit on the tested page, or on the date when the test is configured to end.

Any conversions performed on the website by users who have passed through an A/B test page will be logged under the assigned page variant, which is taken from the value of the A/B testing cookie. The logging of conversion hits is provided by the Web analytics module. In addition to monitoring conversions, the cookie also ensures that returning visitors are always shown the page variant that was previously assigned to them, which helps avoid confusion by maintaining a consistent appearance of the tested page.

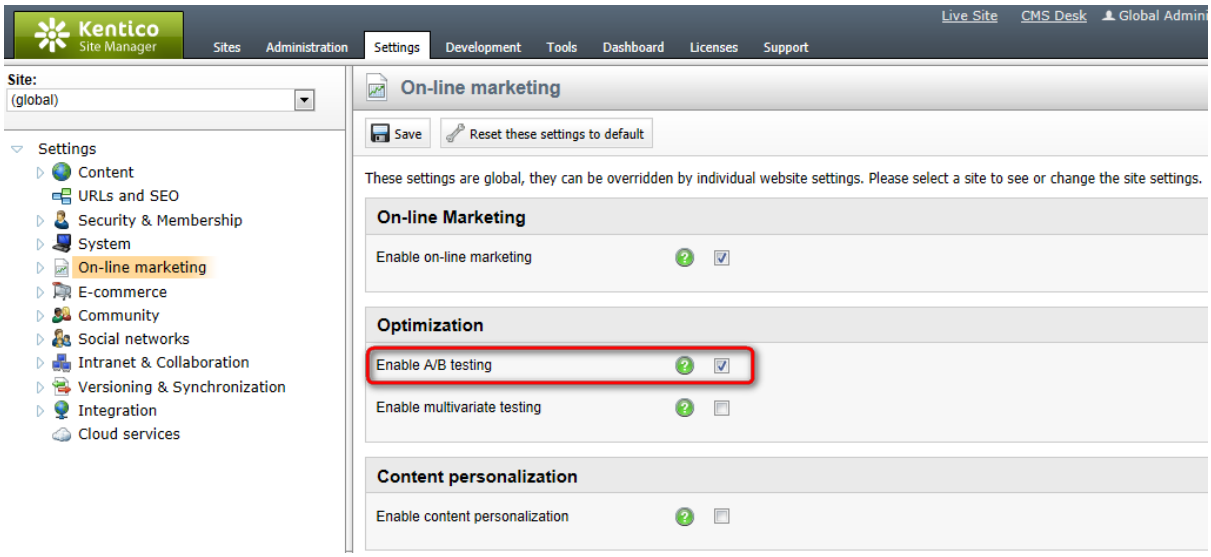
It is possible to run multiple A/B tests concurrently for different pages on the same website. Conversions will be logged for all tests defined for pages visited by a given user, according to the cookies present in the browser.

8.54.2.2 Creating an A/B test

The sections below describe the tasks that must be performed to successfully set up an A/B test on your website.

Enabling A/B testing

Before you can start creating A/B tests for pages, it is first necessary to enable the module by going to **Site Manager -> Settings -> On-line marketing** and checking the **Enable A/B testing** field.



Since web analytics are used to log conversion statistics for individual page variants, the **Enable web analytics** and **Track conversions** fields must also be enabled in the **On-line marketing -> Web analytics** sub-category of the settings. All other requirements for the correct functioning of the Web analytics module also need to be fulfilled, so please refer to Modules -> Web analytics if you encounter any problems with conversion tracking while performing an A/B test.

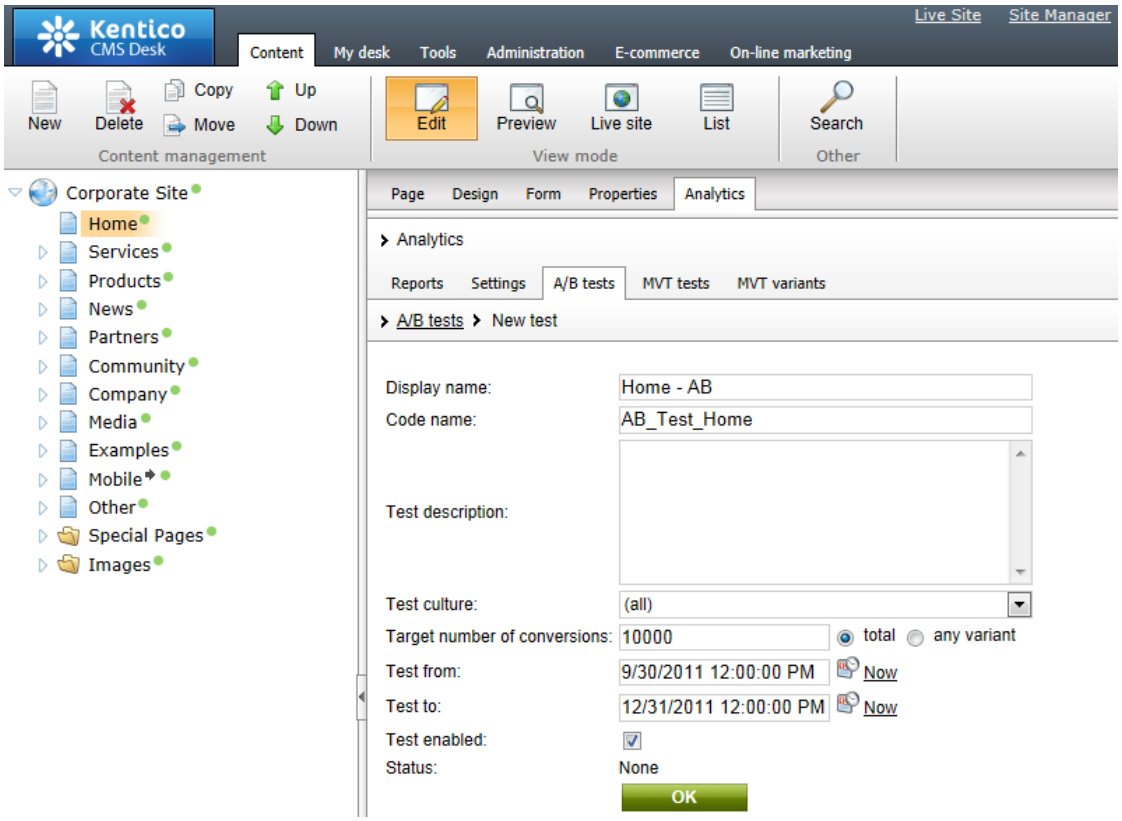
Defining an A/B test for a page

To start optimizing a page on your website via A/B testing, create and configure an *A/B test* object for it. To do this, open **CMS Desk**, select the appropriate document from the content tree and switch to its **Analytics -> A/B tests** tab. Here you can find a list of all A/B tests assigned to the currently selected document and manage them as required.

Click **New test** and fill in the A/B test's properties according to the information in the table below:

Display name	The name of the A/B test displayed in the administration interface.
Code name	Sets a code name that serves as an identifier for the test. It is also used in the name of the browser cookie used to store which of the test's page variants was assigned to a visitor.
Test description	Can be used to enter a text description for the test in order to give information about its purpose, goals, etc.
Test culture	Used to selects which culture version(s) of the document should be included in the test.
Target number of conversions	Sets the number of conversion hits that must be logged to complete the test. Once this number is reached, the A/B test will automatically stop working and switch to the **Finished** status. If the **total** option is selected, then the number will be

	compared with the total sum of the conversion hits logged for all test page variants. If **any variant** is chosen, then the test will be concluded when the specified number is reached by one variant (the one with the most conversion hits). Leaving this property empty or setting it to *0* means that there will be no conversion hit limit for the test.
Test from/to	Sets the time interval during which the test should occur. The **Calendar** button () can be used to select the exact date and time.
Test enabled	This property may be used to manually start or stop the test.
Status	Displays the current status of the A/B test. The following statuses are possible: • **Disabled** - indicates that the test is not active. The original tested page will always be displayed to visitors and conversions will not be logged for the test's variants. • **Not running** - indicates that the test is not active yet. Used when the test is enabled, but the *Test from* date is still in the future. • **Running** - indicates that the test is currently active. • **Finished** - this status is automatically assigned after the *Test to* date passes or when the *Current number of conversions* reaches the *Target number of conversions*. Tests with this status are no longer active. Each page may have only one A/B test running at a given time. It is however possible to have multiple finished or inactive (disabled) tests assigned to a page, which can be used to archive data from previous tests or when preparing future tests. Different culture versions of a page may each have a different test running at the same time.



Click **OK** when everything is configured. The test will now be assigned to the document.

Managing A/B tests through the Web analytics interface

If you wish to manage all tests assigned to different pages from a single location, go to **CMS Desk -> Tools (or On-line marketing) -> Web analytics**. Then expand the **Optimization -> A/B tests** category, select any of the contained items (reports) and switch to the **A/B tests** tab.

Here, all A/B tests defined on the current website and their page variants may be managed in the same way as described for individual documents. The only difference is that the **Original page** property is additionally available when editing a test, which can be used to assign tests to specific pages.

Adding page variants to an A/B test

Test variants are created the same way as any other pages, since each variant is represented by a separate document in the website's content tree and can be managed in **CMS Desk -> Content**. Because of this, you can utilize all page configuration and design options provided by Kentico CMS. Please refer to the Development -> Web development overview chapter if you require basic information about page development.

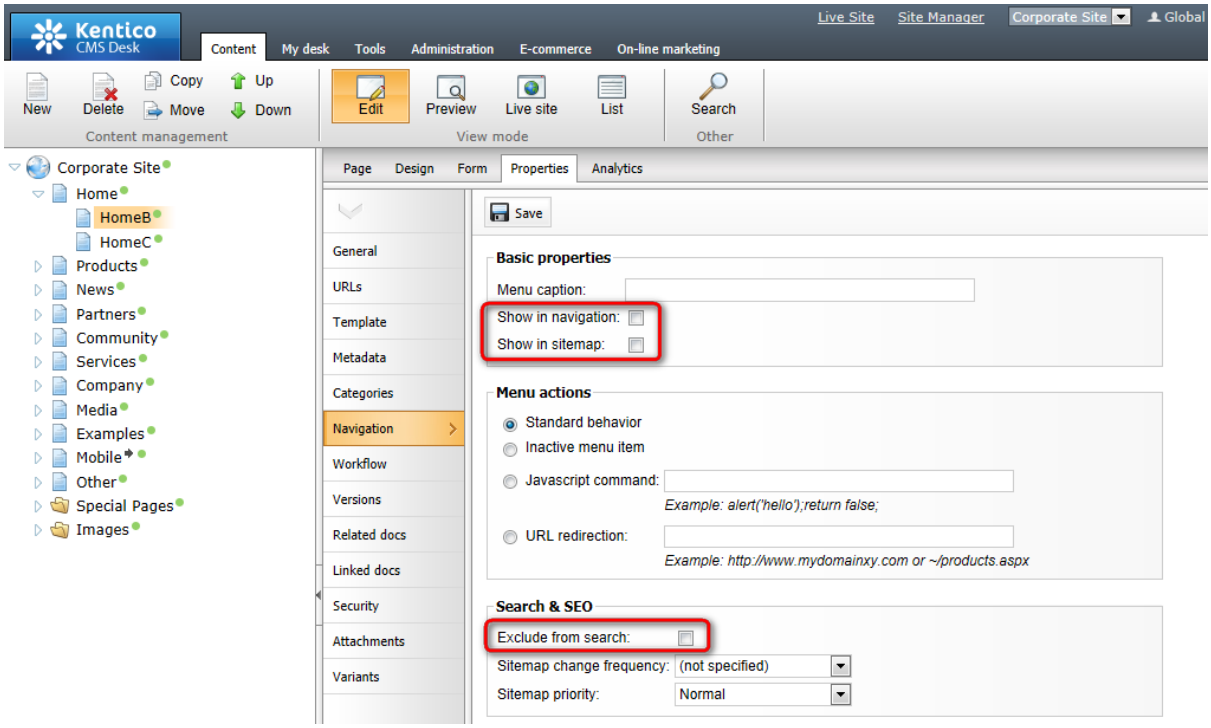
Usually, the page variants of an A/B test will all use the same basic concept, but have one key difference that sets them apart from the original page, such as:

- The position of an important page element.
- A different color scheme intended to highlight a part of the page.
- Alternative text content of headings or descriptions that could potentially be more interesting for visitors.

These are only basic examples of ideas that can be tested by individual variants. You can implement any other type of modifications according to your specific requirements.

You may also leverage existing functionality to help present all variants defined for an A/B test as a single page. The following two properties are available on the **Properties -> Navigation** tab when editing any document:

- **Show in navigation** - indicates if the page variant should be displayed by navigation controls and web parts (i.e. in the website's menus).
- **Show in site map** - indicates if the page variant should be included in the website's Google Sitemap and displayed by the Site map web part.



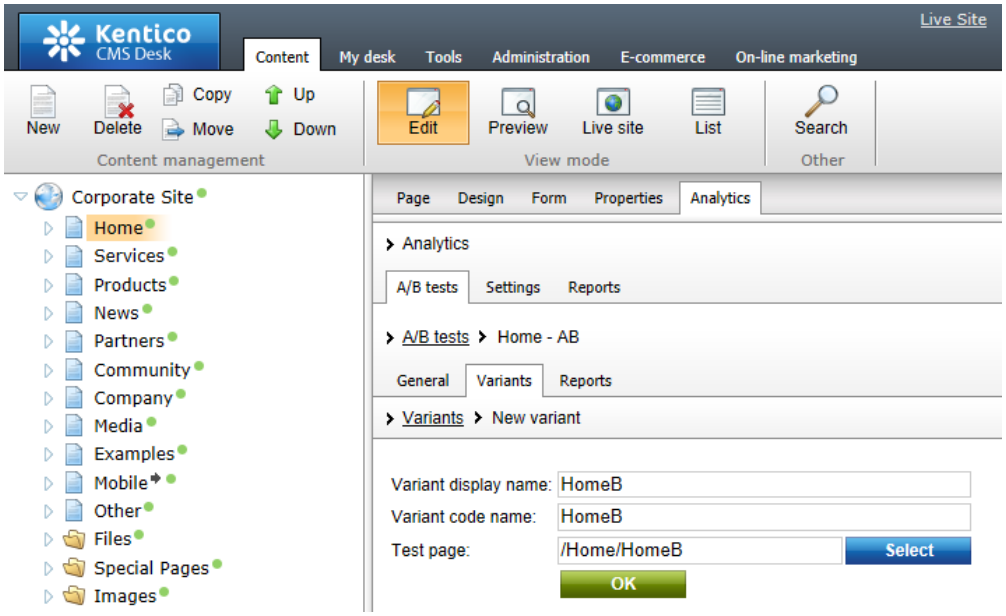
In most cases, it is recommended to have these properties disabled for page variants in order to prevent users from encountering multiple pages with nearly identical content. The appropriate variant will automatically be displayed to users when they view the test's original page, so standard navigation is not necessary.

Additionally, the **Exclude from search** flag can be enabled for page variants under the **Search & SEO** section. This will ensure that the given variant will not be returned in the results of searches performed on the website.

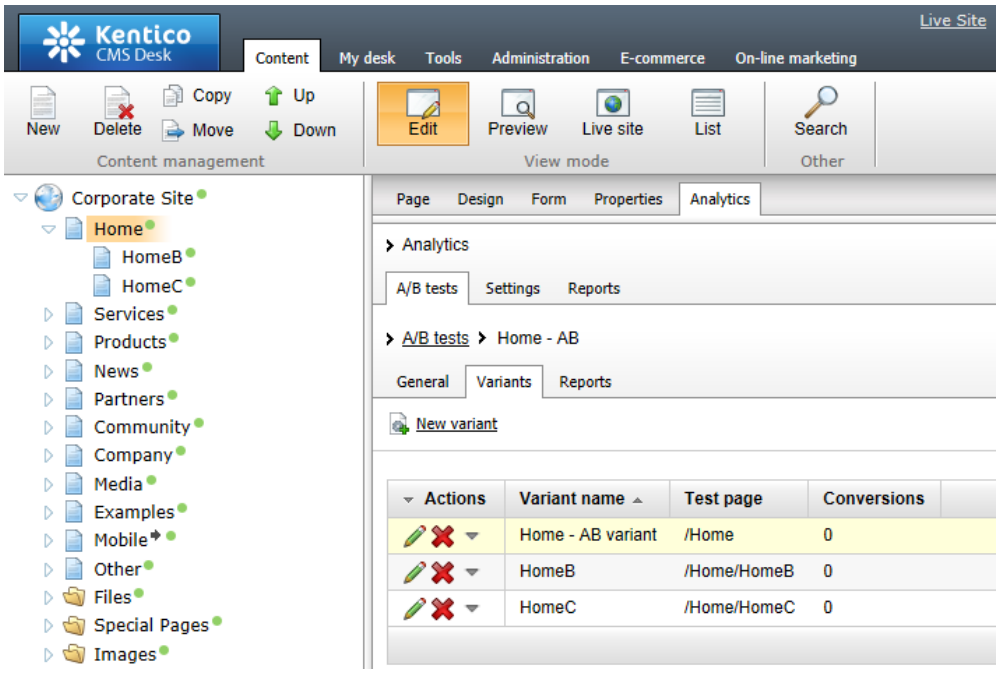
Once the modified versions of the original document are added to the content tree, they must be registered as variants of the given A/B test. To do this, select the original document, go to **Analytics ->**

A/B tests, edit (✎) the test and switch to its **Variants** tab. Click **New variant** and enter the following properties:

- **Variant display name** - the name of the variant displayed in the administration interface.
- **Variant code name** - sets a code name that serves as an identifier for the page variant. It is also used as the value of the A/B testing cookie stored for visitors assigned to this variant.
- **Test page** - must contain the path of the associated page (document). When users assigned to the variant access the original page of the given A/B test, they will see the page specified here instead. The *Select* button may be used to choose an existing page from the website's content tree.



Repeat this process for all documents that should be included as possible page options for the A/B test. The original document is automatically added as a variant when the A/B test is created, and highlighted in the list by a yellow background.

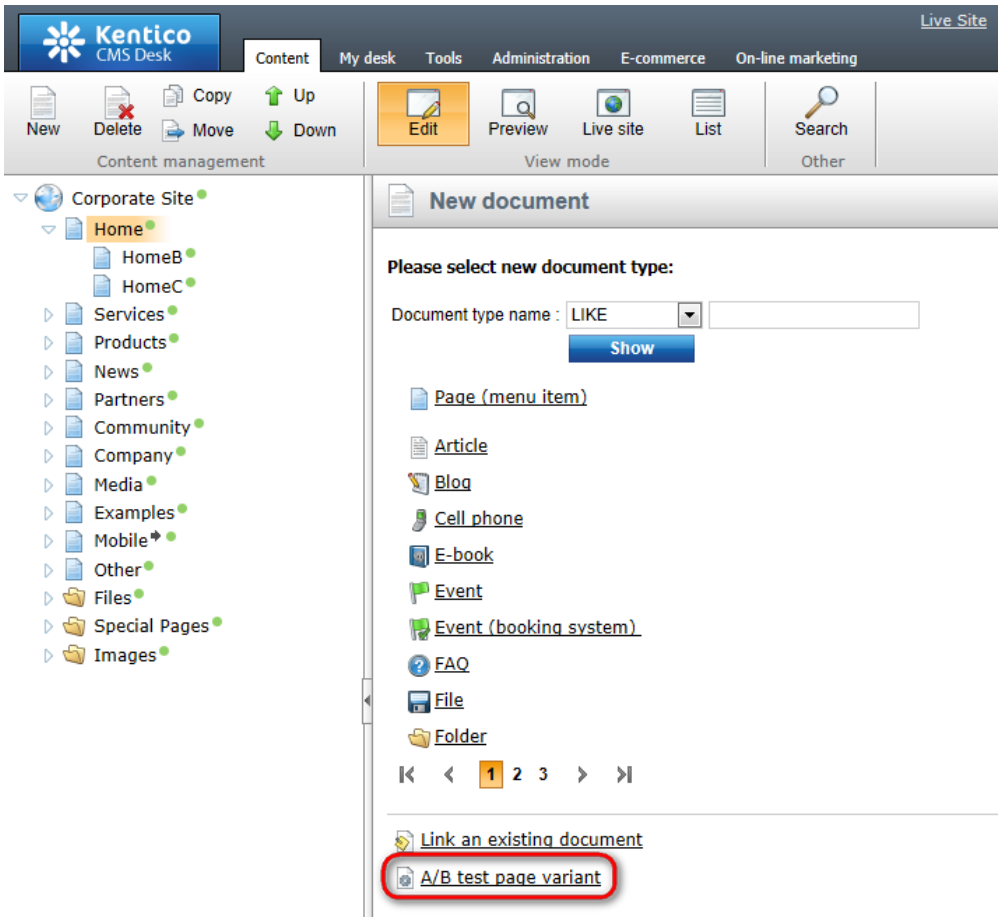


Important!

When editing a running A/B test, it is necessary to always keep the variant **Test page** paths up-to-date and ensure that the target documents actually exist in the content tree. Otherwise visitors may encounter a *Page not found* error if the path of their variant is not valid.

To preserve conversion statistics, variants remain in the system even if their associated document is deleted.

It is possible to save time when defining page variants by using an action that combines the two steps described above. Simply select the A/B test's original page in **CMS Desk -> Content** and click the **New** button in the menu above the content tree. Instead of selecting a document type, choose the **A/B test page variant** option below the list.



This can also be done by right-clicking the document in the content tree and selecting the **New -> A/B test page variant** item from the context menu. A dialog with the following options will be opened:

- **Document name** - sets the name of the new document that will be associated with the page variant.
- **Save under location** - sets the path of the document under which the page variant will be placed. The *Select* button may be used to choose a parent page directly.
- **Assign to A/B test** - selects the A/B test to which the variant will be added.
- **Show in navigation, Show in site map, Exclude from search** - these options may be used to directly set the corresponding properties of the new document. The default values will hide the page variant from all standard website navigation.

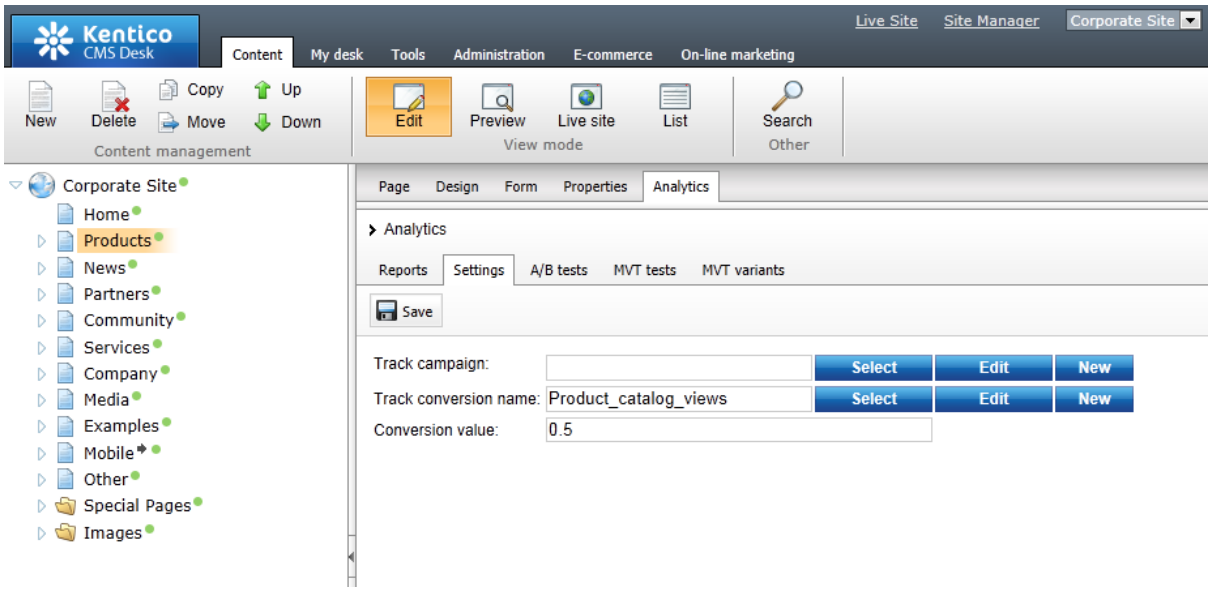
Fill in these properties accordingly and click **Save** to confirm the entered data. This will create a copy of the selected document and automatically add it as a page variant of the specified A/B test. Now you can implement the required modifications that will distinguish the variant from the original page. Please keep in mind that this copy will use the same page template as the original document, so the template must be *Cloned as ad-hoc* on the document's **Properties -> Template** tab if you wish to make any changes to the design or layout of the page.

Configuring conversions

Conversions allow you to track the behaviour and actions of the website's visitors, so they must be used in order to get A/B testing results. They work on a site-wide level and are not assigned to A/B tests in any way. All that needs to be done is define conversion objects and link them with the actions that you

wish to log for the test.

There are many ways to specify that an action should be logged as a conversion. For example, to have the system log a conversion hit whenever a specific page is viewed by a user, select the given document from the content tree in **CMS Desk** and switch to its **Analytics -> Settings** tab. Use the **Track conversion name** property to either create a conversion via the **New** button, or **Select** an existing one.



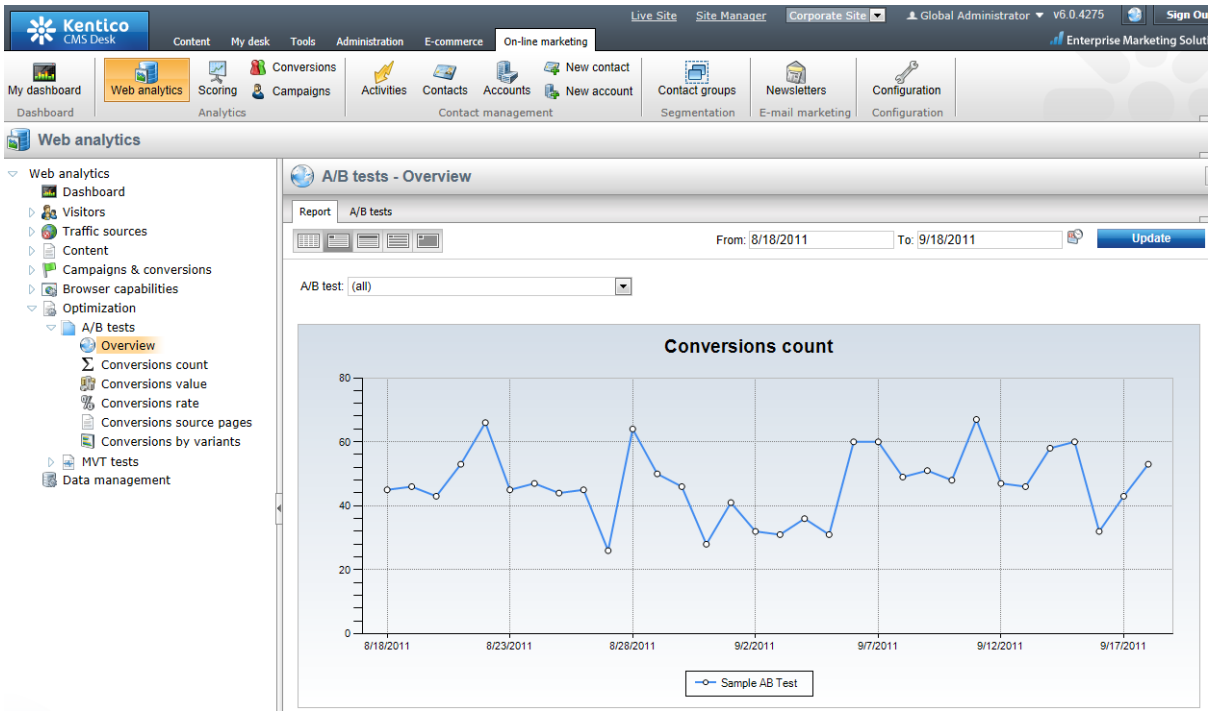
Additionally, many web parts and widgets that allow users to perform important actions (e.g. registration) also have the **Track conversion name** property, which can be used to specify a conversion in the same way. Further details about conversion management may be found in the Modules -> Web analytics -> Conversions chapter.

When an action designated as a conversion is performed anywhere on the website, the system checks if the given user has passed through a page with an A/B test (according to the presence of a cookie). If this is the case, the conversion hit will be logged for the page variant assigned to the user.

Once the A/B test starts running on the live website, you can monitor the conversion statistics for individual page variants using pre-defined reports, as described in the Analyzing A/B test results topic.

8.54.2.3 Analyzing A/B test results

The data gathered during the course of A/B testing may be viewed in reports that display various types of conversion metrics. To access the main interface containing these reports, go to **CMS Desk -> On-line marketing -> Web analytics** and expand the **Optimization -> A/B tests** category.



When viewing a report, the **From** and **To** fields on the right can be used to enter a time period. Only conversion hits that were logged for the selected **A/B test** during the specified interval will be included in the data.

The following options allow you to choose which unit of time should be used in the report:

- **Hour**
- **Day**
- **Week**
- **Month**
- **Year**

This selection determines the length of time which is represented by individual units in the report's graphs and the precision that can be specified in the **From** and **To** fields.

The **Conversions** drop-down list may be used to select which conversion's statistics should be displayed. Please note that the system logs *all* conversion hits generated by visitors who have passed through a page where an A/B test is running. If there are many conversions defined on your website, only those that can somehow be affected by the differences in the A/B test's page variants will contain relevant data.

Data in the reports is represented by two possible types of graphs. The line charts show the progress of a certain conversion statistic over time and display combined data for all of the A/B test's variants. The bar graphs contain details for individual units of time according to the selected report type (hours, days, months etc.). You may also view the conversion data in a table located between the graphs. Each row in the table shows the data logged for a specific page variant, both for the time period currently displayed by the report and the entire duration of the test.

The following reports are available for A/B testing:

Overview	This report can be used to view the progress of the primary metrics measured for the site's A/B tests from a single location.
Conversions count	Displays the number of conversion hits logged for the selected A/B test during the specified time interval. In the bar graph, the number of conversion hits is divided into categories that represent individual page variants. This allows you to compare the A/B test's variants and determine which one generated the most conversions (in absolute terms).
Conversions value	Displays the sum of the conversion values logged for the selected A/B test during the specified time interval. In the bar graph, the conversion values are divided into categories that represent individual page variants, which allows you to determine which variant generated the highest total conversion value. This way you can easily evaluate an A/B test's variants when using weighted conversions that have a different level of importance.
Conversions rate	Used to indicate how many visitors who access the tested page perform a conversion. The conversion rate is calculated as the amount of logged conversion hits divided by the total number of visitors on the variants of the tested page. If you select the *(all)* option from the **Conversions** drop-down list, then the rate will be measured for all possible conversions, i.e. as the percentage of visitors who generated at least one conversion hit of any type. The conversion rate in the bar graph is displayed for individual page variants. This allows you to compare the A/B test's variants and determine which one encouraged the highest share of its visitors to perform a conversion.
Conversions source pages	Displays hit statistics for individual conversions that were logged as part of the selected A/B test during the specified time interval. The data logged for the chosen conversion is categorized according to the page variants defined for the given A/B test. This allows you to easily determine which variant generated the most conversion hits of the selected type.
Conversions by variants	Displays details of the number of hits logged for each conversion by individual page variants defined for the selected A/B test. You can select the variant that you wish to evaluate from the **Variants** drop-down list. The hits logged for the chosen variant are divided into categories that match individual conversions. This allows you to easily measure which conversions are performed most commonly by visitors assigned to the selected page variant.

Reports for individual A/B tests

These reports can also be viewed when editing the original page of an A/B test in **CMS Desk -> Content -> Edit -> Analytics -> A/B testing -> Reports**.

The same options are available as described for the web analytics interface, but statistics are only displayed for the currently edited A/B test.

The following actions may also be performed for the reports:

- **Save** - saves the report in its current state (according to the selected time interval). To view saved reports at a later time, go to *CMS Desk -> Tools -> Reporting*, select the matching report under the *A/B testing* category and switch to the *Saved reports* tab. This action is only available for users who have the *Save reports* permission for the *Web analytics* module.
- **Print** - allows the report to be printed. The available options depend on the used browser.
- **Delete data** - can be used to clear all conversion hits logged during the specified time period for the selected A/B test. Please note that this permanently removes the data from the database. This action is only available for users who have the *Manage data* permission for the *Web analytics* module.
- **Subscribe** - opens a new dialog where you can subscribe to the report. Subscribing allows you to periodically receive e-mails with the up-to-date content of the given report. It is also possible to subscribe to a specific reporting component (graph or table) by right clicking on it and selecting the *Subscribe to* option in the displayed context menu.

The data displayed in the reports can be exported into external files using various formats. This can be done by right clicking on a specific graph, which opens a context menu offering the following export options:

- **Export to Excel** - exports the data displayed by the given object to an XLSX spreadsheet.
- **Export to CSV** - exports data to a CSV file.
- **Export to XML** - exports data to an XML file.

After you select an action from the menu, your browser's standard file download dialog will pop up, letting you open or save the file with the exported data just like when downloading any other type of file. For more details on the data export feature, please refer to the Modules -> UI data export chapter.

8.54.3 Multivariate testing

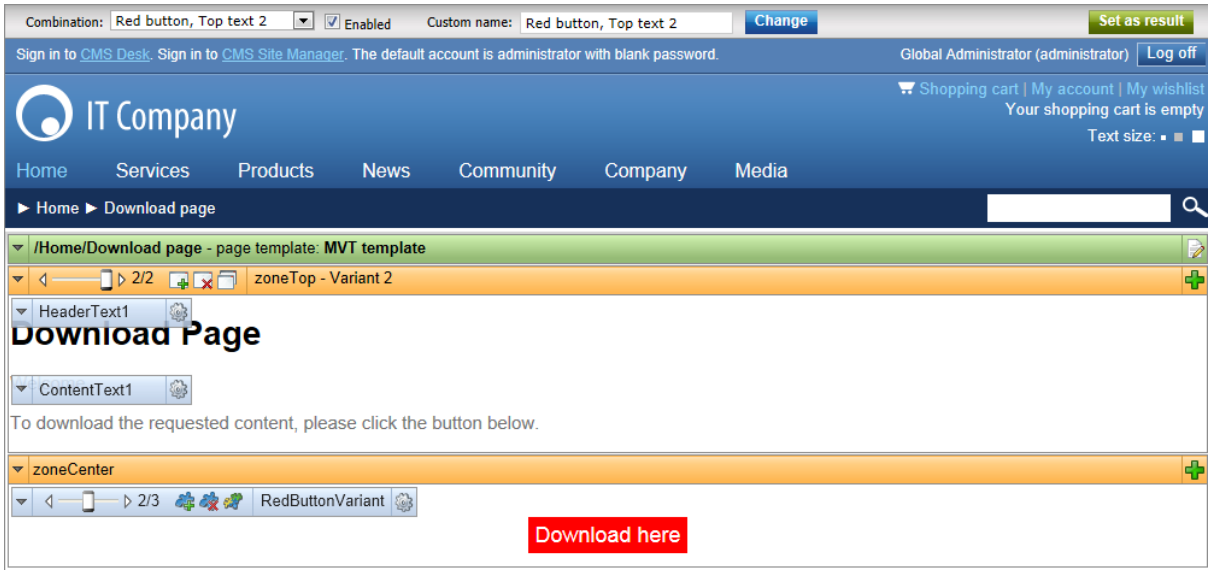
8.54.3.1 Multivariate testing overview

Multivariate testing (MVT) provides a way to optimize a website's pages based on the behaviour of its visitors. It allows you to define any number of elements on a page as variables and create different versions for each one. Once the test is started, users who view the page will see one of the possible versions of its content and their subsequent activity will then be tracked to determine which modifications produce the best results.

The basic objects used to manage this functionality are ***MVT tests***, which can be created for specific documents (pages) on the website. You can implement the changes that you wish to test by creating ***Variants*** of the standard design objects that make up the content of the given page. This includes Web parts, entire web part zones and Widgets in the page's editor zones.

Testing is not done for individual variants, but rather for ***Combinations*** of the variants on the page. Because scenarios with multiple object variables can lead to a very large amount of possible combinations, you can limit the scope of the test by disabling those that you are not considering as options. The default page is also available as a combination, so you can compare potential improvements with the baseline statistics (those of the page with its original, unmodified content).

The image below shows how a page with a defined multivariate test could look in the design interface. There are two different versions of the top zone's content and three possible variants of the web part that displays the download button in the bottom zone. That makes six total combinations of the page's content which can be included in the MVT test.



To learn how you can define an MVT test for a page and create different testing variants of its content, please see the Creating an MVT test topic. The data gathered by MVT tests can be viewed in various reports that allow you to analyze several types of conversion metrics, as described in Analyzing MVT test results.

With multivariate testing, results are tracked for specific combinations of the variants defined on a single page, which represent individual modifications. If you wish to monitor the aggregate effect of all changes made to an entire page as a single variable, you may use A/B testing, which is another optimization feature provided by Kentico CMS.

How it works

When a visitor navigates to a page that has a running MVT test on the live site, one of the enabled content combinations will be displayed. The combination is chosen randomly for every user. With a large enough visit sample size, each active combination should receive roughly the same amount of traffic during the course of the test.

A persistent cookie is stored in the visitor's browser, used to identify which combination was assigned to the user by the given MVT test. The name of the cookie uses the following format: *CMSMVT<MVT test code name>*. It saves the name of the assigned combination as its value. This cookie expires either within 30 days after the last visit on the tested page, or on the date when the test is configured to end.

Any conversions performed on the website by users who have passed through a page where an MVT

test is running will be logged for the given test under the assigned combination, which is taken from the value of the MVT testing cookie. The logging of conversion hits is provided by the Web analytics module. In addition to monitoring conversions, the cookie also ensures that returning visitors are always shown the same content combination that was previously assigned to them, which helps avoid confusion by maintaining a consistent appearance of the tested page.

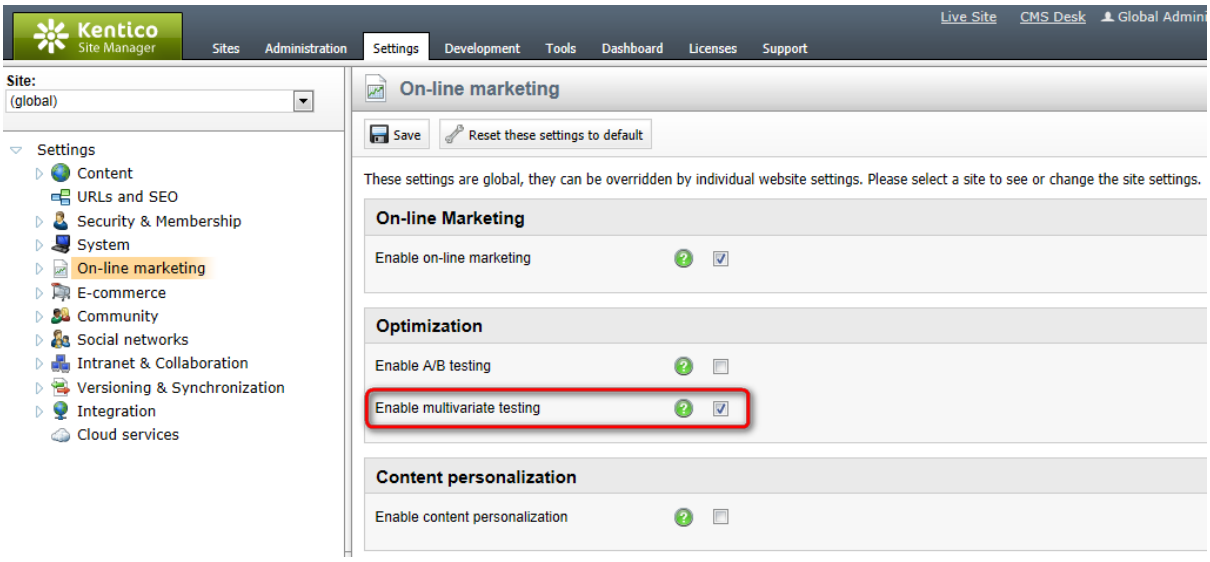
It is possible to run multiple MVT tests concurrently for different pages on the same website. Conversions will be logged for all tests defined on the pages visited by a given user, according to the cookies present in the browser.

8.54.3.2 Creating an MVT test

The sections below describe the tasks that must be performed to successfully set up a multivariate test on your website.

Enabling multivariate testing

Before you can start creating MVT tests for pages, it is first necessary to enable the module by going to **Site Manager -> Settings -> On-line marketing** and checking the **Enable multivariate testing** field.



Since web analytics are used to log conversion statistics for individual versions of tested pages, the **Enable web analytics** and **Track conversions** fields must also be enabled in the **On-line marketing -> Web analytics** sub-category of the settings. All other requirements for the correct functioning of the Web analytics module also need to be fulfilled, so please refer to Modules -> Web analytics if you encounter any problems with conversion tracking while performing an MVT test.

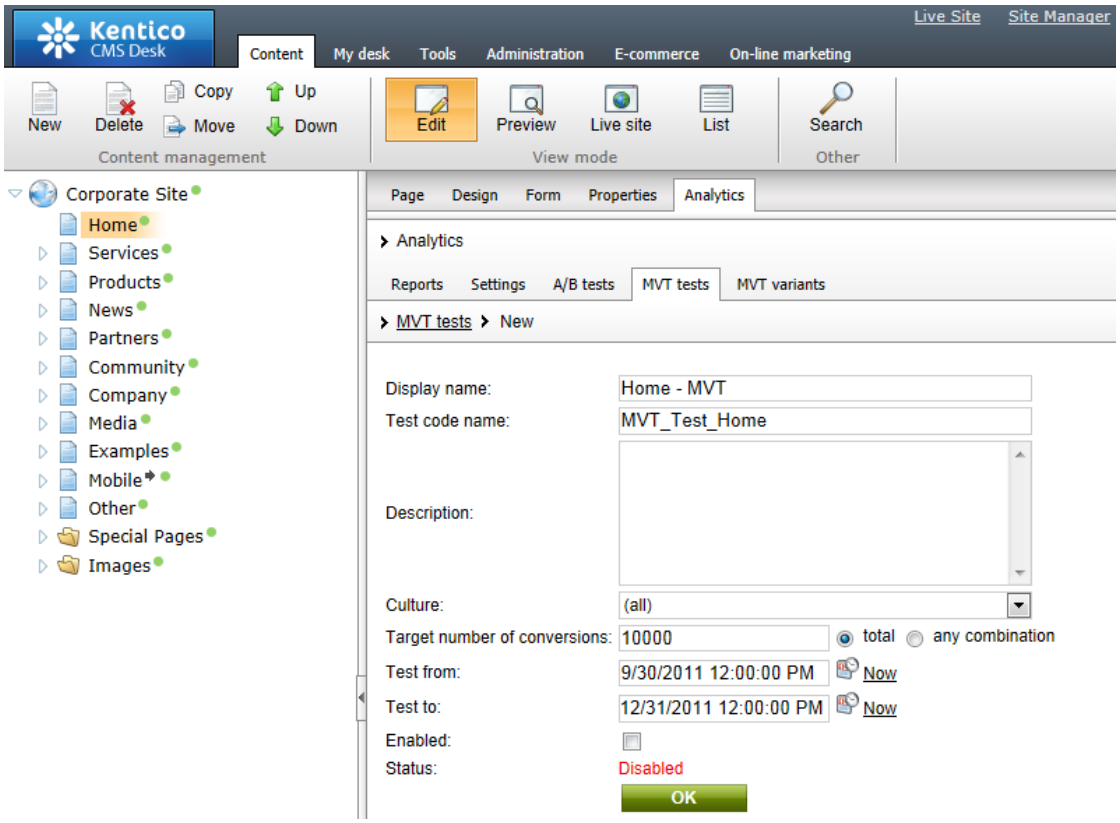
Defining an MVT test

To start optimizing a page on your website via multivariate testing, create and configure an *MVT test* object for it. To do this, open **CMS Desk**, select the appropriate document from the content tree and switch to its **Analytics -> MVT tests** tab. Here you can find a list of all MVT tests assigned to the currently selected document and manage them as required.

Click **New MVT test** and fill in the test's properties according to the information in the table below:

Display name	The name of the MVT test displayed in the administration interface.
Test code name	Sets a code name that serves as an identifier for the test. It is also used in the name of the browser cookie used to store which of the test's variant combinations was assigned to a visitor.
Description	Can be used to enter a text description for the test in order to give information about its purpose, goals, etc.
Culture	Used to select which culture version(s) of the document should be included in the test.
Target number of conversions	Sets the number of conversion hits that must be logged to complete the test. Once this number is reached, the MVT test will automatically stop working and switch to the **Finished** status. If the **total** option is selected, then the number will be compared with the total amount of conversion hits logged for all of the test's combinations. If **any combination** is chosen, then the test will be concluded when the specified number is reached by one combination (the one with the most conversion hits). Leaving this property empty or setting it to *0* means that there will be no conversion hit limit for the test.
Test from/to	Sets the time interval during which the test should occur. The **Calendar** button ( ) can be used to select the exact date and time.
Enabled	This property may be used to manually start or stop the test.
Status	Displays the current status of the MVT test. The following statuses are possible: • **Disabled** - indicates that the test is not active. The default content of the tested page will always be displayed to visitors and conversions will not be logged for the test's combinations. • **Not running** - indicates that the test is not active. Used when the test is enabled, but the *Test from* date is still in the future. • **Running** - indicates that the test is currently active. • **Finished** - this status is automatically assigned after the *Test to* date passes or when the *Current number of conversions* reaches the *Target number of conversions*. Tests with this status are no longer active. Each page may have only one MVT test running at a given time. It is however possible to have multiple finished or

	disabled tests assigned to a page, which can be used to archive data from previous tests or when preparing future tests. Different culture versions of a page may each have a different test running at the same time.



Click **OK** to confirm the configuration and the test will be assigned to the document.

Managing MVT tests through the Web analytics interface

If you wish to manage all tests assigned to different pages from a single location, go to **CMS Desk -> Tools (or On-line marketing) -> Web analytics**. Then expand the **Optimization -> MVT tests** category, select one of the contained items (reports) and switch to the **MVT tests** tab.

Here, all MVT tests defined on the current website may be managed in the same way as described for individual documents. The only difference is that the **Page** property is additionally available when editing a test, which can be used to assign tests to specific pages.

Creating testing variants on the page

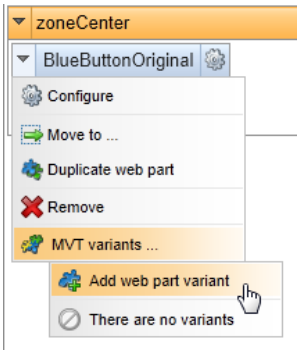
Once a page contains an MVT test, you can start creating the content options that you wish to evaluate. This is done by defining variants for the elements that make up the content of the page. It is possible to

use the following objects as variables:

- **Web parts** - each variant is another instance of the original web part. A variant's properties may be configured differently and an alternative Web part layout can be specified.
- **Web part zones** - in this case, variants are added as entire web part zones. Each zone variant may contain any type or number of child web parts as required. The basic properties of the zone may also be set differently. When a new variant is added to a zone, the content of the original is automatically copied into it, so you do not have to rebuild the zone from scratch if you only need to make small modifications. Please note that creating variants for individual web parts inside zone variants is not supported.
- **Editor widgets** - like with web parts, each variant is a widget of the same type as the original that provides the option to set different values for its properties.

If you are not familiar with the basics of page development in Kentico CMS, it is recommended to read through the Development -> Web parts and Widgets chapters of this guide before you continue.

To add an MVT variant to a web part or zone, open the page on the **Design** tab, expand the context menu of the given object by right clicking its header (or through the ▼ icon), hover over the **MVT variants** option and select **Add <object type> variant** from the second level of the menu.

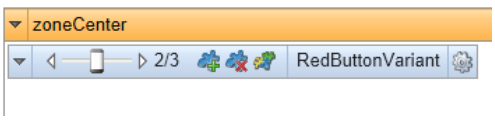


Then, fill in the following properties in the displayed dialog:

- **Display name** - the name used in lists of MVT variants in the administration interface.
- **Code name** - sets a code name that serves as an identifier of the variant.
- **Description** - can be used to enter a text description of the variant.
- **Enabled** - indicates if the variant should be considered as a possible testing option. If you disable an MVT variant, all testing combinations that include this variant will also be disabled.

After you enter and confirm these basic values for the new variant, a configuration dialog will be opened, just like when creating a standard web part or widget. Here, you can set up the available properties (according to the type of the given object) so that the variant generates the content that you wish to test. By default, the values set for the original will be loaded into the properties, so you only have to change the parts of the configuration that are unique for the given variant.

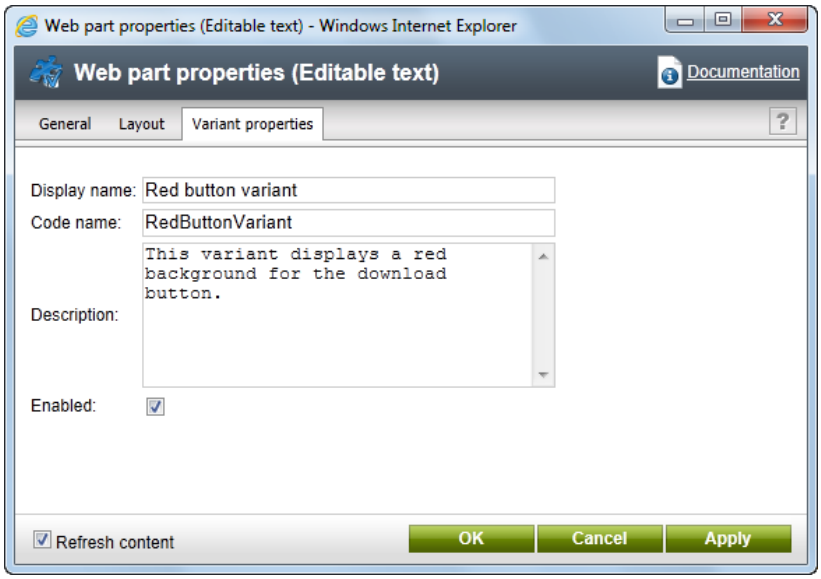
Each object for which you create testing variants will have a slider available in its header as shown below:



This slider can be used to switch between individual variants as needed (including the original). The content specified by the current selection on the page's variant sliders will be displayed in CMS Desk on the **Design** and **Page** tabs, and in **Preview** mode. You can also switch between different versions of the page's content by using the combination panel, which is described in the next section below.

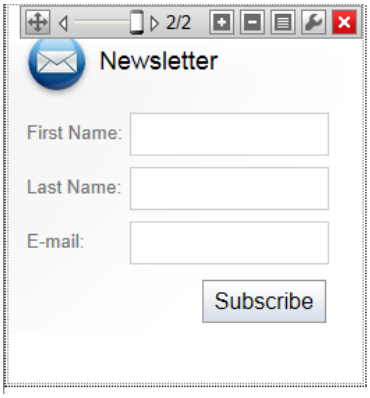
If you wish to view the content of variants while cycling through the slider on the **Design** tab, make sure that the **Display web part content** checkbox on the right side of the **Edit** mode header is checked. To make orientation easier, it is recommended to assign an appropriate **title** property for each web part or zone variant, so that you can identify which one you are working with straight from the header text.

The buttons on the right of the slider allow you to add new variants (), remove the currently selected one () or open a list () of all variants that the given object currently has. At any time, you can **Configure** () the properties of the variant currently chosen on the slider.



When editing a variant like this, you can access the properties that affect the content on the **General** tab. The specific settings related to the testing functionality of the specific variant (the same options that were offered when creating a new variant) can be changed on the additional **Variant properties** tab.

In the case of editor widgets, multivariate testing variants are handled using a very similar approach. The only difference is that editing is done on the **Page** tab of **CMS Desk** and the slider and actions for variant management are located on the pop-up menu of individual widgets.



You can access a list of all multivariate testing variants defined on a given document (page), by selecting it from the content tree in **CMS Desk -> Content -> Edit** and going to **Analytics -> MVT variants**. The variants of all three object types are included here, and they can be removed or edited as necessary.

There are some general rules that apply to all MVT variants:

- If you delete an object from the page, any variants that it might have will be removed along with it.
- Variants are not linked to a specific MVT test. Instead, they are either stored on the page template used by the given document (in the case of web parts and zones) or bound to the document itself (editor widget variants). This means that existing variants may be used by other MVT tests performed on the given page at another time. Also keep this in mind if you wish to Export a multivariate testing scenario to another website. Variants will be transferred along with documents and page templates, not MVT test objects.

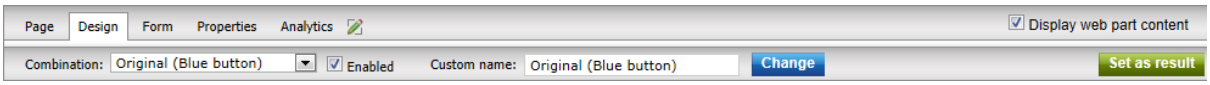
Multivariate testing and Content personalization

Please note that it is not possible to create multivariate testing variants of web parts, zones or widgets that already have variants defined for Content personalization.

Setting up testing combinations

Individual testing scenarios are represented by combinations of the MVT variants created on the page. By default, all possible combinations of the page's content will be included in a multivariate test, but you may perform some additional configuration to fine-tune your test as required.

Combinations can be managed for documents that have an MVT test defined via a panel located at the top of the page editing interface in CMS Desk. This panel is available on both the **Page** and **Design** tabs, and in **Preview** mode.



You can choose any of the available combinations through the **Combination** selector. Doing so will cause the page to display the content defined by the variants that make up the given combination. The selection made through the combination panel is linked with the positions set on the MVT sliders of variable objects on the page. If you switch to a different variant through an object's slider, the current combination will also change accordingly and vice versa.

Pages where there are several variants for multiple objects will have a large amount of possible combinations. For this reason, it is recommended to carefully choose which combinations should be included in the page's MVT test. You can include or exclude a combination by toggling the **Enabled** checkbox located on the panel next to the selector. There is a close relationship between the status of a combination and its variants. If an individual variant is disabled through its properties, all combinations that include it will also be disabled. If a disabled combination is enabled at a later point, all of its variants will be enabled automatically if needed.

A good way to evaluate potential combinations is to check them in the **Preview** mode of CMS Desk, and disable those where the variants clash visually, have conflicting logic or are otherwise incompatible.

We recommend leaving the *Default page* combination enabled, so you can compare any improvements with the baseline statistics of the original page.

Another best practice is to set a descriptive custom name for each active combination, so that you can easily differentiate between combinations in the editing interface and identify them in the test's result reports. To do this, simply select a combination through the panel, enter the appropriate text into the **Custom name** field and click the **Change** button. The default names assigned to new combinations are composed of a number indicating their order, followed by a list of the display names of all variants that are included in the given combination.

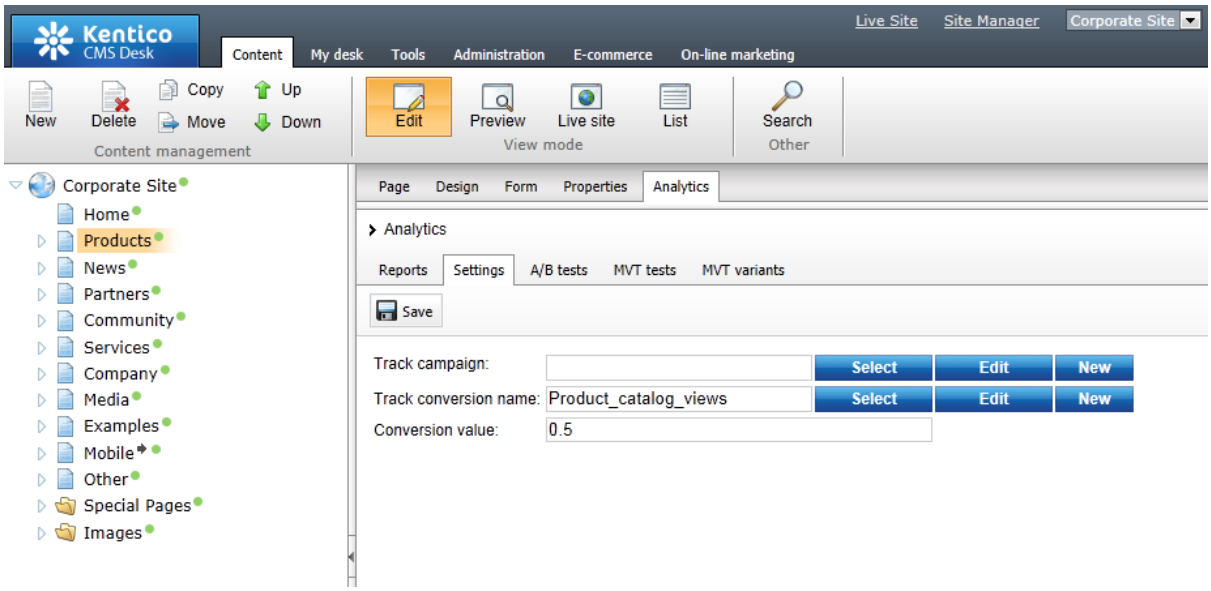
Once the page's variants and combinations are all configured as needed, you can enable your MVT test on the **Analytics -> MVT tests** tab of the document. Making further modifications on the page while an MVT test is running is not recommended, since this may affect the accuracy and relevance of the measured statistics. A warning will be displayed below the combination panel on pages where a test is running.

The **Set as result** action should only be used after the conclusion of the test, once you have analyzed the logged conversion data and identified the combination that provides the best results. The action allows you to easily set the winning combination as the permanent content of the page. Simply select the given combination via the combination panel, then click the button and confirm the action. This will replace the original web parts, zones and widgets with the variants included in the currently selected combination and remove all other MVT variants from the page.

Configuring conversions

Conversions allow you to track the behaviour and actions of the website's visitors, so they must be used in order to get MVT testing results. They work on a site-wide level and are not assigned to MVT tests in any way. All that needs to be done is define conversion objects and link them with the actions that you wish to log for the test.

There are many ways to specify that an action should be logged as a conversion. For example, to have the system log a conversion hit whenever a specific page is viewed by a user, select the given document from the content tree in **CMS Desk** and switch to its **Analytics -> Settings** tab. Use the **Track conversion name** property to either create a conversion via the **New** button, or **Select** an existing one.



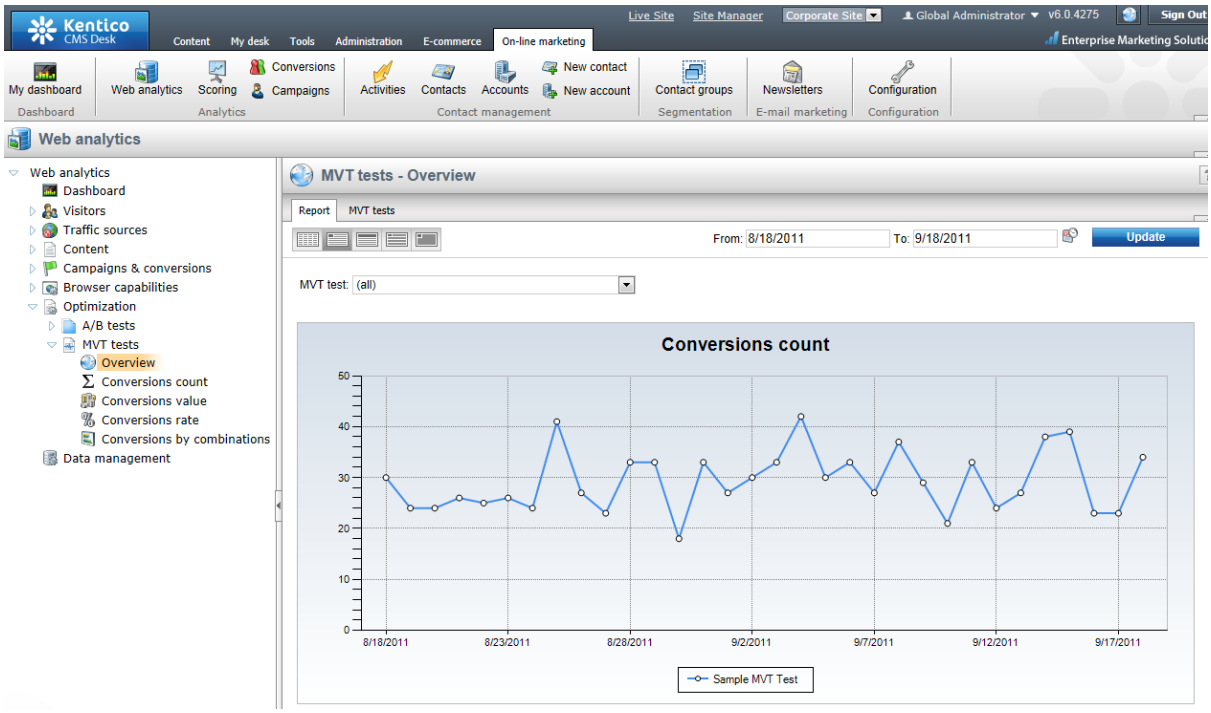
Additionally, many web parts and widgets that allow users to perform important actions (e.g. registration) also have the **Track conversion name** property, which can be used to specify a conversion in the same way. Further details about conversion management may be found in the Modules -> Web analytics -> Conversions chapter.

When an action designated as a conversion is performed anywhere on the website, the system checks if the given user has passed through a page with a running MVT test (according to the presence of a cookie). If this is the case, the conversion hit will be logged for the content combination that was assigned to the user.

Once you enable the MVT test and it starts running on the live site, the conversion statistics for individual combinations can be monitored using pre-defined reports, as described in the Analyzing MVT test results topic.

8.54.3.3 Analyzing MVT test results

The data gathered during the course of MVT testing may be viewed in reports that display various types of conversion metrics. To access the main interface containing these reports, go to **CMS Desk -> On-line marketing -> Web analytics** and expand the **Optimization -> MVT tests** category.



When viewing a report, the **From** and **To** fields on the right can be used to enter a time period. Only conversion hits that were logged for the selected **MVT test** during the specified interval will be included in the data.

The following options allow you to choose which unit of time should be used in the report:

- **Hour**
- **Day**
- **Week**
- **Month**
- **Year**

This selection determines the length of time which is represented by individual units in the report's graphs and the precision that can be specified in the **From** and **To** fields.

The **Conversions** drop-down list may be used to select which conversion's statistics should be displayed. Please note that the system logs *all* conversion hits generated by visitors who have passed through a page where an MVT test is running. If there are many conversions defined on your website, only those that can somehow be affected by the differences between the tested content combinations will have relevant data.

Data in the reports is represented by two possible types of graphs. The line charts show the progress of a certain conversion statistic over time and display combined data for all of the tested combinations. The bar graphs contain details for individual units of time according to the selected report type (hours, days, months etc.). You may also view the conversion data in a table located between the graphs. Each row in the table shows the data logged for a specific combination, both for the time period currently displayed by the report and the entire duration of the test.

The following reports are available for MVT testing:

Overview	This report can be used to view the progress of the primary metrics measured for the site's MVT tests from a single location.
Conversions count	Displays the number of conversion hits logged for the selected MVT test during the specified time interval. In the bar graph, the number of conversion hits is divided into categories that represent individual content combinations. This allows you to compare the tested combinations and determine which one generated the most conversions (in absolute terms).
Conversions value	Displays the sum of the conversion values logged for the selected MVT test during the specified time interval. In the bar graph, the conversion values are divided into categories that represent individual content combinations, which allows you to determine which one generated the highest total conversion value. This way you can easily evaluate the results of the MVT test when using weighted conversions that have a different level of importance.
Conversions rate	Used to indicate how many visitors who access the tested page perform a conversion. The conversion rate is calculated as the amount of logged conversion hits divided by the total number of visitors on the given page. If you select the *(all)* option from the **Conversions** drop-down list, then the rate will be measured for all possible conversions, i.e. as the percentage of visitors who generated at least one conversion hit of any type. The conversion rate in the bar graph is displayed for individual content combinations. This allows you to compare the tested combinations and determine which one encouraged the highest share of its visitors to perform a conversion.
Conversions by combinations	Displays details about the number of conversion hits logged for individual content combinations defined on the page associated with the selected MVT test. You can select the combination that you wish to evaluate from the **Combinations** drop-down list. If the MVT test is used for multiple culture versions of the page, you can also specify the culture. The hits logged for the chosen combination are divided into categories that match individual conversions. This allows you to easily measure which conversions are performed most commonly by visitors assigned to the selected content combination.

Reports for individual MVT tests

These reports can also be viewed when working with the tested page in **CMS Desk -> Content -> Edit** by editing the given MVT test in **Analytics -> MVT tests** and switching

to its **Reports** tab.

The same options are available as described for the web analytics interface, but statistics are only displayed for the currently edited MVT test.

The following actions may also be performed for the reports:

- **Save** - saves the report in its current state (according to the selected time interval). To view saved reports at a later time, go to *CMS Desk -> Tools -> Reporting*, select the matching report under the *M/V testing* category and switch to the *Saved reports* tab. This action is only available for users who have the *Save reports* permission for the *Web analytics* module.
- **Print** - allows the report to be printed. The available options depend on the used browser.
- **Delete data** - can be used to clear all conversion hits logged during the specified time period for the selected MVT test. Please note that this permanently removes the data from the database. This action is only available for users who have the *Manage data* permission for the *Web analytics* module.
- **Subscribe** - opens a new dialog where you can subscribe to the report. Subscribing allows you to periodically receive e-mails with the up-to-date content of the given report. It is also possible to subscribe to a specific reporting component (graph or table) by right clicking on it and selecting the *Subscribe to* option in the displayed context menu.

The data displayed in the reports can be exported into external files using various formats. This can be done by right clicking on a specific graph, which will open a context menu offering the following export options:

- **Export to Excel** - exports the data displayed by the given object to an XLSX spreadsheet.
- **Export to CSV** - exports data to a CSV file.
- **Export to XML** - exports data to an XML file.

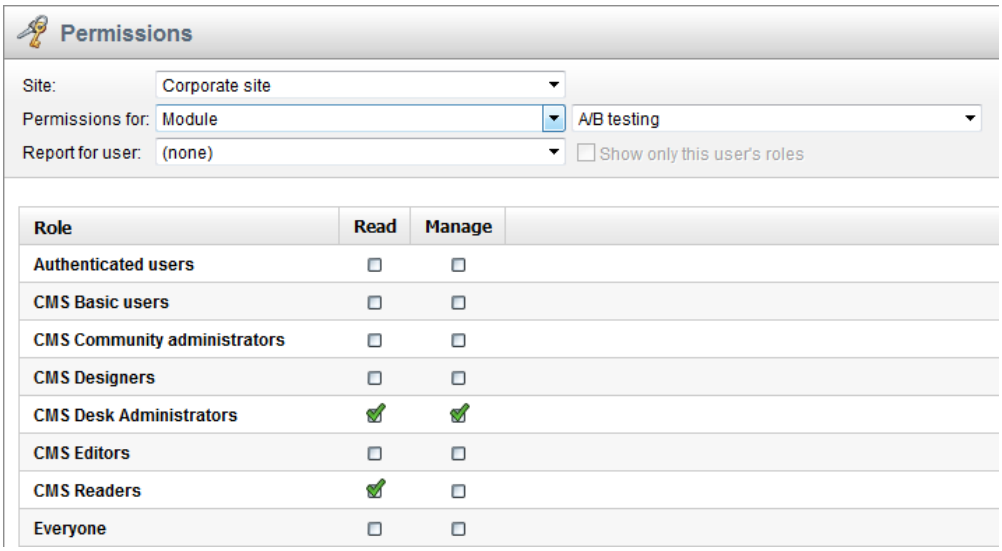
After you select an action from the menu, your browser's standard file download dialog will pop up, letting you open or save the file with the exported data just like when downloading any other type of file. For more details on the data export feature, please refer to the Modules -> UI data export chapter.

8.54.4 Security

Permissions from several different modules are required to perform A/B or MVT testing actions on the website. These permissions can be set by going to **CMS Site Manager** (or **CMS Desk**) **-> Administration -> Permissions**.

To configure the basic security options for optimization testing, select the *A/B testing* or *MVT testing* module respectively. The following two permissions can be assigned for either one of the modules:

- **Read** - allows members of the selected roles to view all parts of the A/B or MVT testing management interface and the corresponding reports.
- **Manage** - allows members of the selected roles to create, edit and delete tests and manage their variants. In this case of A/B testing, this means page variants. For MVT testing, this affects management of object (web part, zone or widget) variants on pages that have a test defined.



Additionally, permissions for the *Web analytics* module are required to access A/B or MVT testing reports in the web analytics interface and in **CMS Desk -> Content -> Analytics -> A/B tests / MVT tests -> Reports**. The following are available:

- **Read** - allows members of the specified roles to access the web analytics interface and view its reports (including those for A/B and MVT testing).
- **Save reports** - allows members of the specified roles to archive the reports via the *Save* action.
- **Manage data** - allows members of the specified roles to delete the conversion statistics logged for tests. This is done through the *Delete data* action available for reports.

Editing A/B testing page variant documents

Because every page variant is represented by a document in the content tree, the standard document permissions apply.

All permissions that can be configured for the *Content* module are checked, such as for creating, modifying or deleting documents. Also, the **Design web site** permission for the *Design* module is needed to edit page variants on the **Design** tab of **CMS Desk.**

Managing MVT object variants

The **Design web site** permission for the *Design* module is also needed for users to be able to manage the variants of web parts and zones on the **Design** tab of **CMS Desk**.

To work with variants of editor widgets on the **Page** tab, the **Modify** permission for the *Content* module is required. Also, the security settings defined for specific widgets are checked, as described in Development -> Widgets -> Security.

8.55 Modules internals and API

8.55.1 Overview

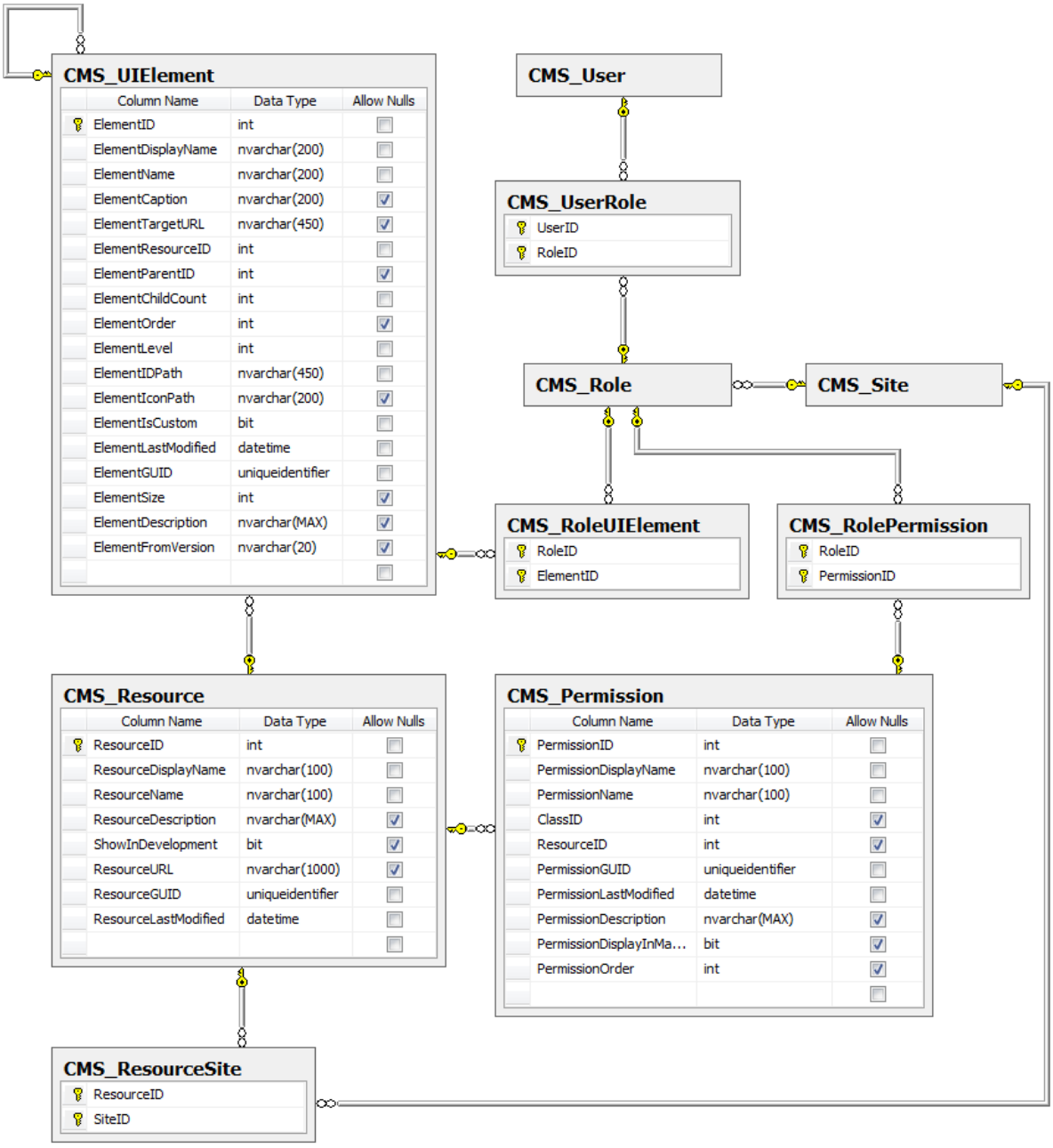
In this chapter, you will learn which database tables and API classes are used in Modules. You will also see the most common API examples.

Advanced API examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all methods from the module classes, please refer to Kentico CMS API Reference.

8.55.2 Database tables

The following database tables are used in Modules:

- **CMS_Resource** - contains records representing modules.
- **CMS_ResourceSite** - contains relationships between modules and sites indicating that particular modules are available for a particular site.
- **CMS_Permission** - contains records representing modules permissions.
- **CMS_RolePermission** - contains relationships between roles and permissions indicating that a particular role is granted with particular permissions.
- **CMS_UIElement** - contains records representing modules UI elements.
- **CMS_RoleUIElement** - contains relationships between roles and UI elements indicating that a particular role is granted with particular UI elements.



8.55.3 API classes

If you are not familiar with the database table data management by Info and Provider classes, we strongly recommend that you refer to the Database table API topic first.

> **Please note**
>
> The Modules classes use the **CMS.SiteProvider** namespace.

CMS_Resource table API:

- **CMSResourceInfo** - represents one module.
- **CMSResourceInfoProvider** - provides management of modules.

CMS_ResourceSite table API:

- **CMSResourceSiteInfo** - represents relationship between one module and one site expressing that the particular module is available for the particular site.
- **CMSResourceSiteInfoProvider** - provides management of relationships between modules and sites.

CMS_Permission table API:

- **CMSPermissionInfo** - represents one module permission.
- **CMSPermissionInfoProvider** - provides management of module permissions.

CMS_RolePermission table API:

- **CMSRolePermissionInfo** - represents relationship between one role and one permission expressing that the particular role is granted with the particular permission.
- **CMSRolePermissionInfoProvider** - provides management of relationships between roles and permissions.

CMS_UIElement table API:

- **CMSUIElementInfo** - represents one module UI element.
- **CMSUIElementInfoProvider** - provides management of module UI elements.

CMS_RoleUIElement table API:

- **CMSRoleUIElementInfo** - represents relationship between one role and one UI element expressing that the particular role is granted with the particular UI element.
- **CMSRoleUIElementInfoProvider** - provides management of relationships between roles and UI elements.

8.55.4 API examples

8.55.4.1 Overview

These topics show examples of how the Modules API can be used:

- Managing modules
- Managing permissions
- Managing UI elements

Please note

All API examples can be simulated in the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please install CMS API examples and refer to ***<web project folder>\CMSAPIExamples\Code\Development\Modules\Default.aspx.cs***.

The Modules API examples use the following namespaces:

```
using System;
using System.Data;

using CMS.GlobalHelper;
using CMS.UIControls;
using CMS.CMSHelper;
using CMS.SiteProvider;
```

8.55.4.2 Managing modules

The following example creates a module.

```
private bool CreateModule()
{
    // Create new module object
    ResourceInfo newModule = new ResourceInfo();

    // Set the properties
    newModule.ResourceDisplayName = "My new module";
    newModule.ResourceName = "MyNewModule";

    // Save the module
    ResourceInfoProvider.SetResourceInfo(newModule);

    return true;
}
```

The following example gets and updates a module.

```
private bool GetAndUpdateModule()
{
    // Get the module
    ResourceInfo updateModule = ResourceInfoProvider.GetResourceInfo
("MyNewModule");
    if (updateModule != null)
    {
        // Update the properties
        updateModule.ResourceDisplayName =
```

```
updateModule.ResourceDisplayName.ToLower();

        // Save the changes
        ResourceInfoProvider.SetResourceInfo(updateModule);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates modules.

```
private bool GetAndBulkUpdateModules()
{
    // Prepare the parameters
    string where = "ResourceName LIKE N'MyNewModule%'";

    // Get the data
    DataSet modules = ResourceInfoProvider.GetResources(where, null);
    if (!DataHelper.DataSourceIsEmpty(modules))
    {
        // Loop through the individual items
        foreach (DataRow moduleDr in modules.Tables[0].Rows)
        {
            // Create object from DataRow
            ResourceInfo modifyModule = new ResourceInfo(moduleDr);

            // Update the properties
            modifyModule.ResourceDisplayName =
modifyModule.ResourceDisplayName.ToUpper();

            // Save the changes
            ResourceInfoProvider.SetResourceInfo(modifyModule);
        }

        return true;
    }

    return false;
}
```

The following example adds a module to site.

```
private bool AddModuleToSite()
{
    /// Get the module
    ResourceInfo module = ResourceInfoProvider.GetResourceInfo("MyNewModule");
    if (module != null)
    {
```

```
        int moduleId = module.ResourceId;
        int siteId = CMSContext.CurrentSiteID;

        // Save the binding
        ResourceSiteInfoProvider.AddResourceToSite(moduleId, siteId);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates site modules.

```
private bool GetAndBulkUpdateSiteModules()
{
    int siteId = CMSContext.CurrentSiteID;
    string where = "ResourceName LIKE N'MyNewModule%'";

    // Get the data
    DataSet modules = ResourceInfoProvider.GetResources(where, null, 0, null,
siteId);
    if (!DataHelper.DataSourceIsEmpty(modules))
    {
        // Loop through the individual items
        foreach (DataRow moduleDr in modules.Tables[0].Rows)
        {
            // Create object from DataRow
            ResourceInfo modifyModule = new ResourceInfo(moduleDr);

            // Update the properties
            modifyModule.ResourceDisplayName =
modifyModule.ResourceDisplayName.ToLower();

            // Save the changes
            ResourceInfoProvider.SetResourceInfo(modifyModule);
        }

        return true;
    }

    return false;
}
```

The following example removes a module from site.

```
private bool RemoveModuleFromSite()
{
    // Get the module
    ResourceInfo removeModule = ResourceInfoProvider.GetResourceInfo
```

```
("MyNewModule");
    if (removeModule != null)
    {
        int siteId = CMSContext.CurrentSiteID;

        // Get the binding
        ResourceSiteInfo moduleSite = ResourceSiteInfoProvider.GetResourceSiteInfo
(removeModule.ResourceId, siteId);

        // Delete the binding
        ResourceSiteInfoProvider.DeleteResourceSiteInfo(moduleSite);

        return true;
    }

    return false;
}
```

The following example deletes a module.

```
private bool DeleteModule()
{
    // Get the module
    ResourceInfo deleteModule = ResourceInfoProvider.GetResourceInfo
("MyNewModule");

    // Delete the module
    ResourceInfoProvider.DeleteResourceInfo(deleteModule);

    return (deleteModule != null);
}
```

8.55.4.3 Managing permissions

The following example creates a permission.

```
private bool CreatePermission()
{
    // Get the resource
    ResourceInfo module = ResourceInfoProvider.GetResourceInfo("MyNewModule");
    if (module != null)
    {
        // Create new permission object
        PermissionNameInfo newPermission = new PermissionNameInfo();

        // Set the properties
        newPermission.PermissionDisplayName = "My new permission";
        newPermission.PermissionName = "MyNewPermission";
        newPermission.ResourceId = module.ResourceId;
```

```
        // Save the permission
        PermissionNameInfoProvider.SetPermissionInfo(newPermission);

        return true;
    }

    return false;
}
```

The following example gets and updates a permission.

```
private bool GetAndUpdatePermission()
{
    // Get the permission
    PermissionNameInfo updatePermission =
PermissionNameInfoProvider.GetPermissionNameInfo("MyNewPermission", "MyNewModule",
null);
    if (updatePermission != null)
    {
        // Update the properties
        updatePermission.PermissionDisplayName =
updatePermission.PermissionDisplayName.ToLower();

        // Save the changes
        PermissionNameInfoProvider.SetPermissionInfo(updatePermission);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates permissions.

```
private bool GetAndBulkUpdatePermissions()
{
    // Prepare the parameters
    string where = "PermissionName LIKE N'MyNewPermission%'";

    // Get the data
    DataSet permissions = PermissionNameInfoProvider.GetPermissionNames(where,
null, 0, null);
    if (!DataHelper.DataSourceIsEmpty(permissions))
    {
        // Loop through the individual items
        foreach (DataRow permissionDr in permissions.Tables[0].Rows)
        {
            // Create object from DataRow
            PermissionNameInfo modifyPermission = new PermissionNameInfo
(permissionDr);
```

```
            // Update the properties
            modifyPermission.PermissionDisplayName =
modifyPermission.PermissionDisplayName.ToUpper();

            // Save the changes
            PermissionNameInfoProvider.SetPermissionInfo(modifyPermission);
        }

        return true;
    }

    return false;
}
```

The following example adds a permission to role.

```
private bool AddPermissionToRole()
{
    // Get the permission
    PermissionNameInfo permission =
PermissionNameInfoProvider.GetPermissionNameInfo("MyNewPermission", "MyNewModule",
null);

    // Get the role
    RoleInfo role = RoleInfoProvider.GetRoleInfo("cmsdeskadmin",
CMSContext.CurrentSiteID);

    if ((permission != null) && (role != null))
    {
        // Create new role permission object
        RolePermissionInfo newRolePermission = new RolePermissionInfo();

        // Set the properties
        newRolePermission.PermissionID = permission.PermissionId;
        newRolePermission.RoleID = role.RoleID;

        // Add permission to role
        RolePermissionInfoProvider.SetRolePermissionInfo(newRolePermission);

        return true;
    }

    return false;
}
```

The following example removes a permission from role.

```
private bool RemovePermissionFromRole()
{
```

```
    // Get the permission
    PermissionNameInfo permission =
PermissionNameInfoProvider.GetPermissionNameInfo("MyNewPermission", "MyNewModule",
null);

    // Get the role
    RoleInfo role = RoleInfoProvider.GetRoleInfo("cmsdeskadmin",
CMSContext.CurrentSiteID);

    if ((permission != null) && (role != null))
    {
        // Get the role permission
        RolePermissionInfo deleteRolePermission =
RolePermissionInfoProvider.GetRolePermissionInfo(role.RoleID,
permission.PermissionId);

        // Remove permission from role
        RolePermissionInfoProvider.DeleteRolePermissionInfo(deleteRolePermission);

        return true;
    }

    return false;
}
```

The following example deletes a permission.

```
private bool DeletePermission()
{
    // Get the permission
    PermissionNameInfo deletePermission =
PermissionNameInfoProvider.GetPermissionNameInfo("MyNewPermission", "MyNewModule",
null);

    // Delete the permission
    PermissionNameInfoProvider.DeletePermissionInfo(deletePermission);

    return (deletePermission != null);
}
```

8.55.4.4 Managing UI elements

The following example creates a UI element.

```
private bool CreateUIElement()
{
    // Get the module
    ResourceInfo module = ResourceInfoProvider.GetResourceInfo("MyNewModule");
    if (module != null)
    {
```

```
        // Get the parent UI element
        UIElementInfo rootElement = UIElementInfoProvider.GetRootUIElementInfo
(module.ResourceId);
        if (rootElement == null)
        {
            // Create root UI element
            rootElement = new UIElementInfo();
            rootElement.ElementResourceID = module.ResourceId;
            rootElement.ElementDisplayName = module.ResourceDisplayName;
            rootElement.ElementName = module.ResourceName.ToLower().Replace(".",
"");
            rootElement.ElementIsCustom = false;

            // Save root UI element
            UIElementInfoProvider.SetUIElementInfo(rootElement);
        }

        // Create new UI element object
        UIElementInfo newElement = new UIElementInfo();

        // Set the properties
        newElement.ElementDisplayName = "My new element";
        newElement.ElementName = "MyNewElement";
        newElement.ElementResourceID = module.ResourceId;
        newElement.ElementIsCustom = true;
        newElement.ElementParentID = rootElement.ElementID;

        // Save the UI element
        UIElementInfoProvider.SetUIElementInfo(newElement);

        return true;
    }

    return false;
}
```

The following example gets and updates a UI element.

```
private bool GetAndUpdateUIElement()
{
    // Get the UI element
    UIElementInfo updateElement = UIElementInfoProvider.GetUIElementInfo
("MyNewModule", "MyNewElement");
    if (updateElement != null)
    {
        // Update the properties
        updateElement.ElementDisplayName =
updateElement.ElementDisplayName.ToLower();

        // Save the changes
        UIElementInfoProvider.SetUIElementInfo(updateElement);
```

```
        return true;
    }

    return false;
}
```

The following example gets and bulk updates UI elements.

```
private bool GetAndBulkUpdateUIElements()
{
    // Prepare the parameters
    string where = "ElementName LIKE N'MyNewElement%'";

    // Get the data
    DataSet elements = UIElementInfoProvider.GetUIElements(where, null);
    if (!DataHelper.DataSourceIsEmpty(elements))
    {
        // Loop through the individual items
        foreach (DataRow elementDr in elements.Tables[0].Rows)
        {
            // Create object from DataRow
            UIElementInfo modifyElement = new UIElementInfo(elementDr);

            // Update the properties
            modifyElement.ElementDisplayName =
modifyElement.ElementDisplayName.ToUpper();

            // Save the changes
            UIElementInfoProvider.SetUIElementInfo(modifyElement);
        }

        return true;
    }

    return false;
}
```

The following example adds a UI element to role.

```
private bool AddUIElementToRole()
{
    // Get the role
    RoleInfo role = RoleInfoProvider.GetRoleInfo("cmsdeskadmin",
CMSContext.CurrentSiteID);

    // Get the UI element
    UIElementInfo element = UIElementInfoProvider.GetUIElementInfo("MyNewModule",
"MyNewElement");

    if ((role != null) && (element != null))
```

```
    {
        // Create new role UI element object
        RoleUIElementInfo newRoleElement = new RoleUIElementInfo();

        // Set the properties
        newRoleElement.RoleID = role.RoleID;
        newRoleElement.ElementID = element.ElementID;

        // Save the role UI element
        RoleUIElementInfoProvider.SetRoleUIElementInfo(newRoleElement);

        return true;
    }

    return false;
}
```

The following example removes a UI element from role.

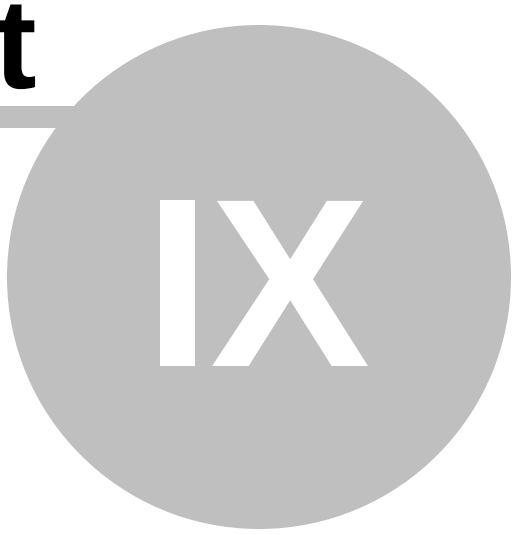
```
private bool RemoveUIElementFromRole()
{
    // Get the role
    RoleInfo role = RoleInfoProvider.GetRoleInfo("cmsdeskadmin",
CMSContext.CurrentSiteID);

    // Get the UI element
    UIElementInfo element = UIElementInfoProvider.GetUIElementInfo("MyNewModule",
"MyNewElement");

    if ((role != null) && (element != null))
    {
        // Get the role UI element
        RoleUIElementInfo deleteElement =
RoleUIElementInfoProvider.GetRoleUIElementInfo(role.RoleID, element.ElementID);

        // Delete the role UI element
        RoleUIElementInfoProvider.DeleteRoleUIElementInfo(deleteElement);

        return (deleteElement != null);
    }

    return false;
}
```

The following example deletes a UI element.

```
private bool DeleteUIElement()
{
    // Get the UI element
    UIElementInfo deleteElement = UIElementInfoProvider.GetUIElementInfo
```

```
("MyNewModule", "MyNewElement");

    // Delete the UI element
    UIElementInfoProvider.DeleteUIElementInfo(deleteElement);

    return (deleteElement != null);
}
```

Part



External utilities

9 External utilities

9.1 Kentico AD Import Utility

9.1.1 Overview

Introduction

Kentico Active Directory Import Utility is a standalone Windows application which allows importing of users and groups (roles) from Active Directory (AD) into Kentico CMS and assigning users to roles depending on AD settings. The application also provides the possibility of updating already imported users and roles so that their properties are the same as in the current AD.

What can it do?

- Import users from AD into Kentico CMS.
- Import roles (groups) from AD into Kentico CMS.
- Assign users to appropriate roles based on AD settings.
- Update already imported users and roles according to current AD.

What can't it do?

- Import from multiple ADs or domains at once.
- Import the tree structure of roles, since Kentico CMS does not support hierarchical roles.
- Since there is no hierarchy in the CMS roles, import cannot keep the tree structure of AD groups.

Terminology

- **Import profile** – XML file with import settings; this file can be created using the wizard mode, or even written manually; it is necessary to have an import profile prepared when you want to use the console mode of the tool
- **SAM Account Name** - logon name used to support clients and servers on older versions of the operating system, such as Windows NT 4.0, Windows 95, Windows 98, and LAN Manager.
- **UPN (User Principal Name)** - Internet-style login name for a user. It is based on the RFC 822 standard. The UPN is shorter than the distinguished name and easier to remember. By convention, the name should map to the user's e-mail name. The value set for this attribute is equal to the length of the user's ID and the domain name. (Sample UPN: username@subdomain.domain.tld)
- **Role or Group** - these two terms have an almost identical meaning, while "group" is used in AD terminology and "role" in Kentico CMS

Requirements

- **Kentico CMS 5.0** or higher
- **Ultimate edition** or any edition with the **Advanced package**

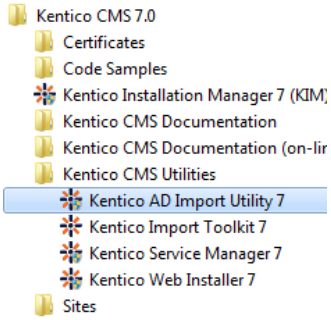
Launching the utility

There are two ways how the utility can be launched. You will typically use the AD import wizard, which is described in the Using the wizard chapter. If you want to schedule AD import to be performed on a

regular basis, you may utilize the second option - launching AD import from the command line. This approach is described in the AD import from command line chapter.

9.1.2 Using the wizard

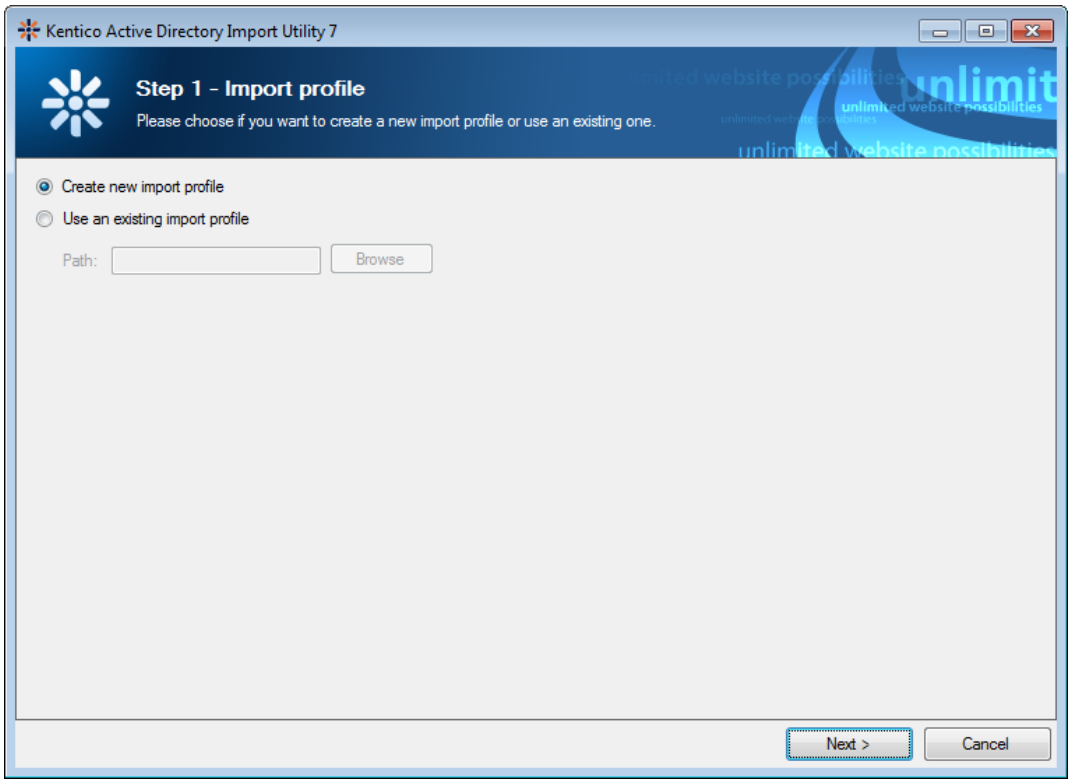
The AD Import Utility wizard can be executed either from **Start menu -> All programs -> Kentico CMS <version number> -> Kentico CMS Utilities**, or by launching the **ADImport.exe** file located in **<Kentico CMS installation folder>\Bin** (typically *c:\Program Files (x86)\KenticoCMS\<version number>\Bin*).



The following text describes particular steps of the wizard:

Step 1 – Import profile settings

In the first step, you need to choose if you want to create a new import profile or use an existing XML profile. If you select an existing profile, values will be pre-filled in the following steps based on the profile settings.



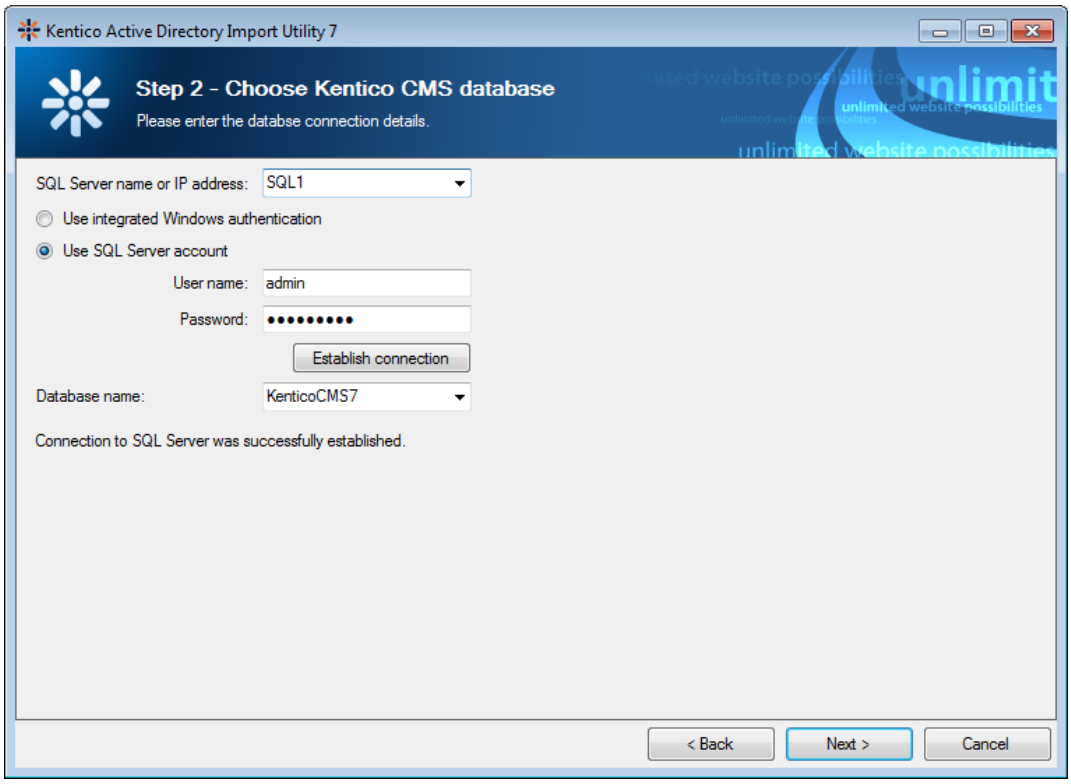
Step 2 – Kentico CMS DB Setup

In the second step, specify the target database of the CMS where the users and roles will be imported:

- **SQL Server name or IP address** - name or IP address of the server where the target database is stored.
- **Database name** - name of the target database.

- **Use integrated Windows authentication** - choose this option if you want to log on to the server using Windows authentication.
- **Use SQL Server account** - choose this option if you want to log on to the server using credentials filled in the fields below.

It is a good idea to test the specified connection using the **Test connection** button before proceeding to the next step.

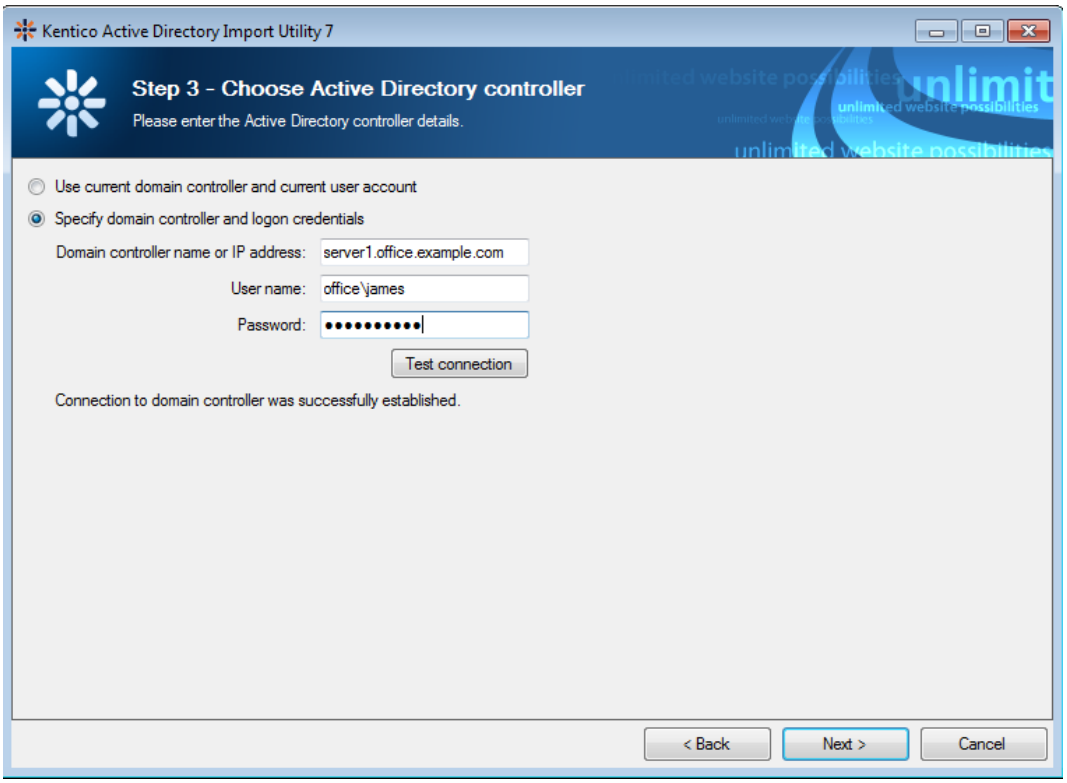


Step 3 – Active directory connection

In the third step, specify the source AD's domain controller. You have two options:

- **Use current user account** - uses the domain where the current user belongs.
- **Specify domain controller and logon credentials** - if you choose this option, you can enter the logon details manually into the fields below.

Here again, it is recommended to test the specified connection using the **Test connection** button.



Step 4 – Import settings

In this step, you can adjust some general settings of the import process:

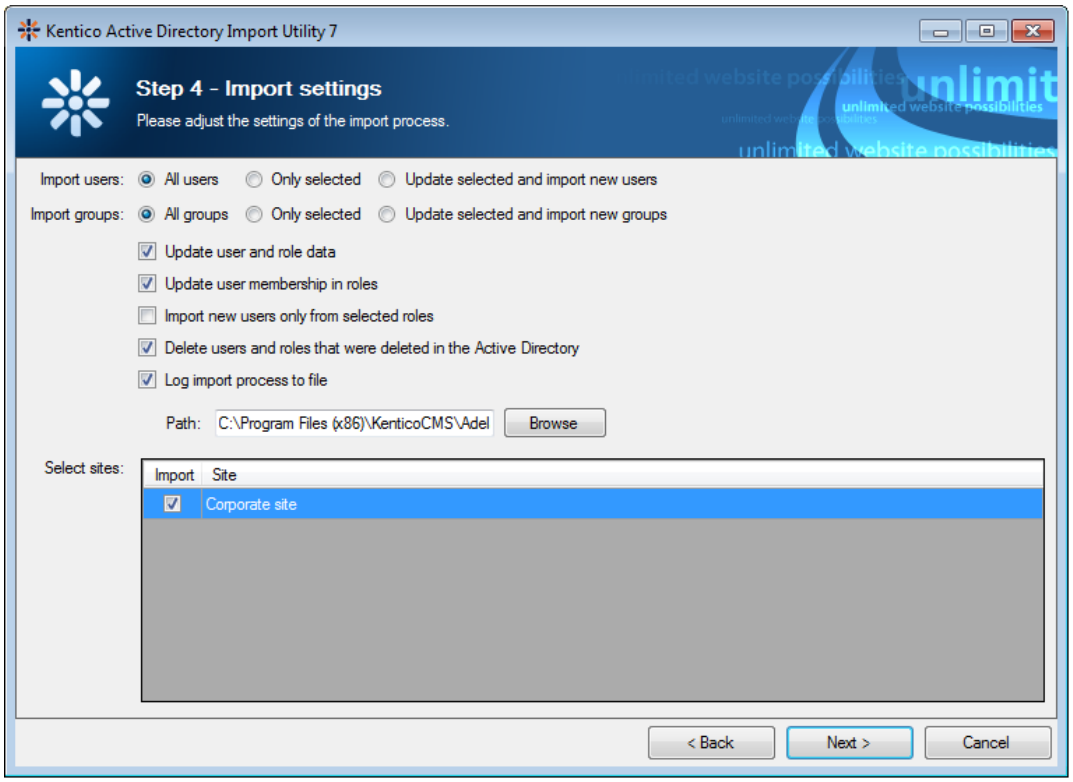
- **Import users/groups** - determines which users or groups (roles) will be pre-selected in Step 6, you have the following options:
  - **All** - all users or groups will be pre-selected.
  - **Only selected** - when using an existing import profile, selection stored in the profile will be used. Otherwise, nothing will be pre-selected.
  - **Only selected and new** - same as above, while new users and groups will be selected as well.

- **Update user and role data** - if enabled, properties of users and roles already imported from the AD will be updated in the CMS based on the current values in AD.
- **Update user membership in roles** - if enabled, membership of users imported from the AD will be updated in the CMS based on the current membership settings in AD.
- **Import new users only from selected roles** - if enabled, only those new users who belong to at least one role (group) selected in Step 6 or 8 of the wizard will be imported. Please note that enabling this option may override previous selection of users to be imported.
- **Delete users and roles that were deleted in the Active directory** - if enabled, users who were previously imported from the Active Directory but were deleted on the source server since then will be deleted in the CMS.
- **Log import process to file** - if enabled, you can specify a file where the import log will be stored.

- **Select sites** - choose the sites to which the imported users and roles will be assigned.

Please note

If you do not choose any site in this step, the rest of the wizard will leave out steps related to the import of roles (groups). This happens because it is currently not possible to import roles from AD into Kentico CMS as global objects and they must be assigned to a specific site.



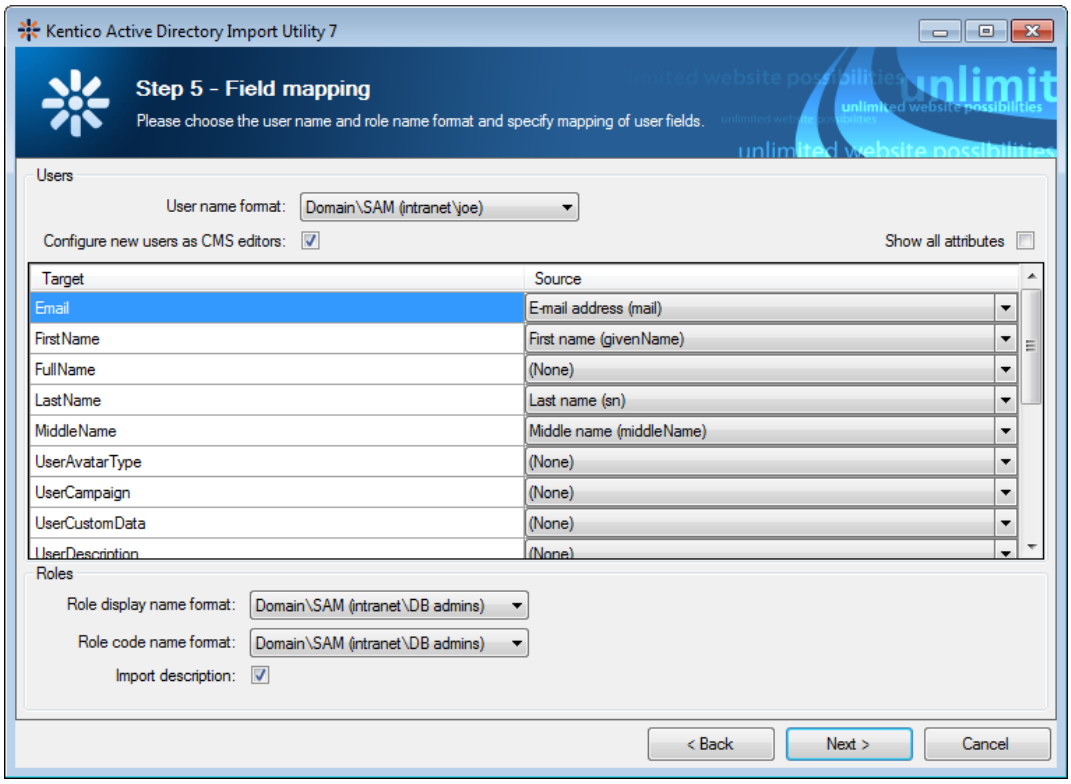
Step 5 – Import properties

In this step, you are asked to define user name and role name format and to bind AD user properties to CMS user properties. The following options can be defined:

- **User name format** - choose one of the three possilbe formats:
  - Domain\SAM (e.g. intranet\joe)
  - SAM account name (e.g. joe)
  - UPN (joe@intranet.mycompany.com)
- **Configure user as CMS editor** - turn on to enable the *Is editor* option for newly imported users. This option is located in *Administration -> Users -> user edit -> General.*
- **Target/Source** - you can choose which attributes from AD (Source) will be mapped to particular attributes of the *CMS_User* role.
- **Show all attributes** - turn on to enable all the attributes, including custom attributes, from your AD schema to be selected as *Source*. Note that you can import attributes of any data type, however, their values are always imported to *Target* as string.

- **Role display name format** - choose one of the two possible formats:

  - Domain\SAM (intranet\DB Admins)
  - SAM (DB Admins)
- **Role code name format** - choose one of the following formats:
  - Domain\SAM (intranet\DB Admins)
  - SAM (DB Admins)
  - Guid (16-byte number)
- **Import description** - indicates if role description should be imported from the AD.



Step 6 – Select users & groups to be imported

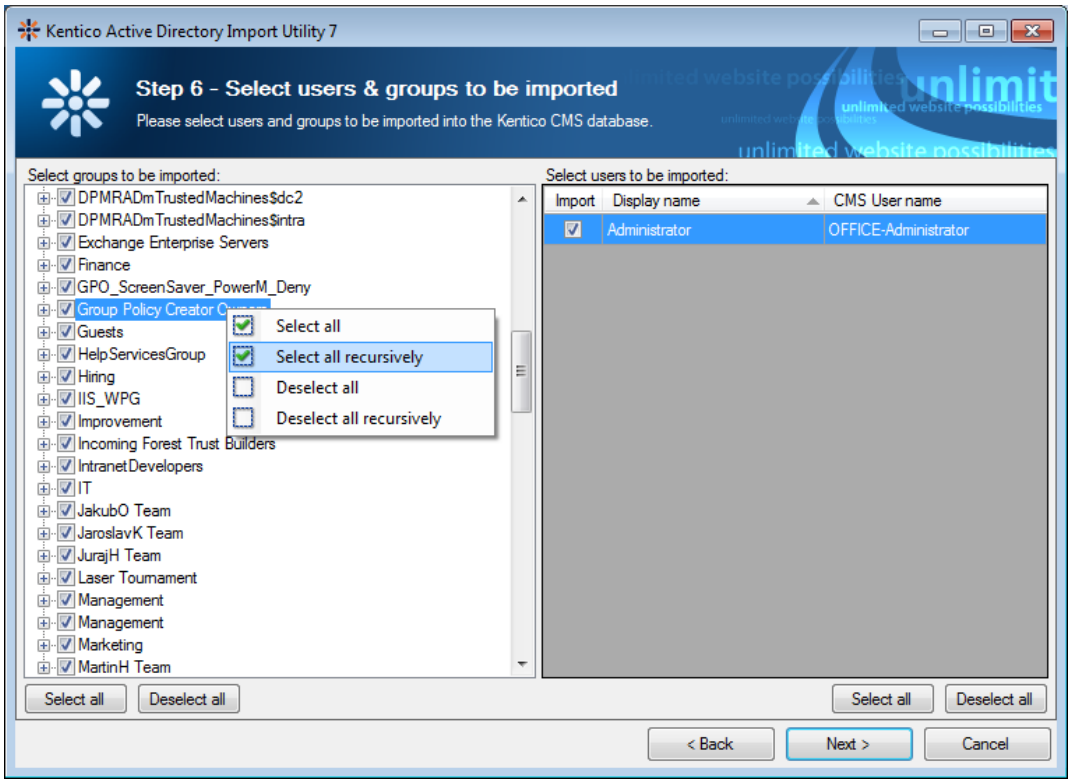
In the sixth step, you can select which roles and users will be imported. It will be possible to adjust the settings made here in the following two steps.

On the left, you can see all groups (roles) found on the source server. If you select a group, its members are displayed in the list on the right. You can define which users and roles will be imported using the appropriate check-boxes.

By right-clicking a group, you can display a context menu with the following actions:

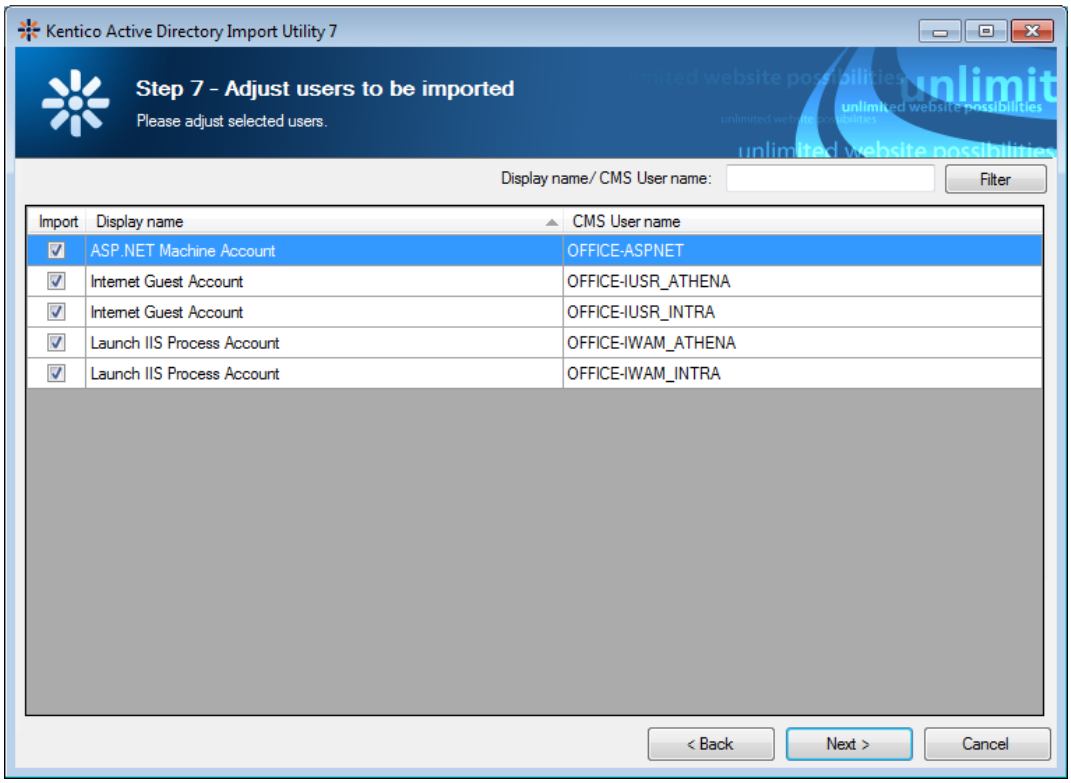
- **Select all** - selects all child groups directly under the selected group.
- **Select all recursively** - selects all child groups under the selected group until the last level.
- **Deselect all** - selects all groups directly under the selected group.
- **Deselect all recursively** - selects all group under the selected group until the last level.

All users in a role or all roles can be selected or deselected in one click using the **Select all** and **Deselect all** buttons.



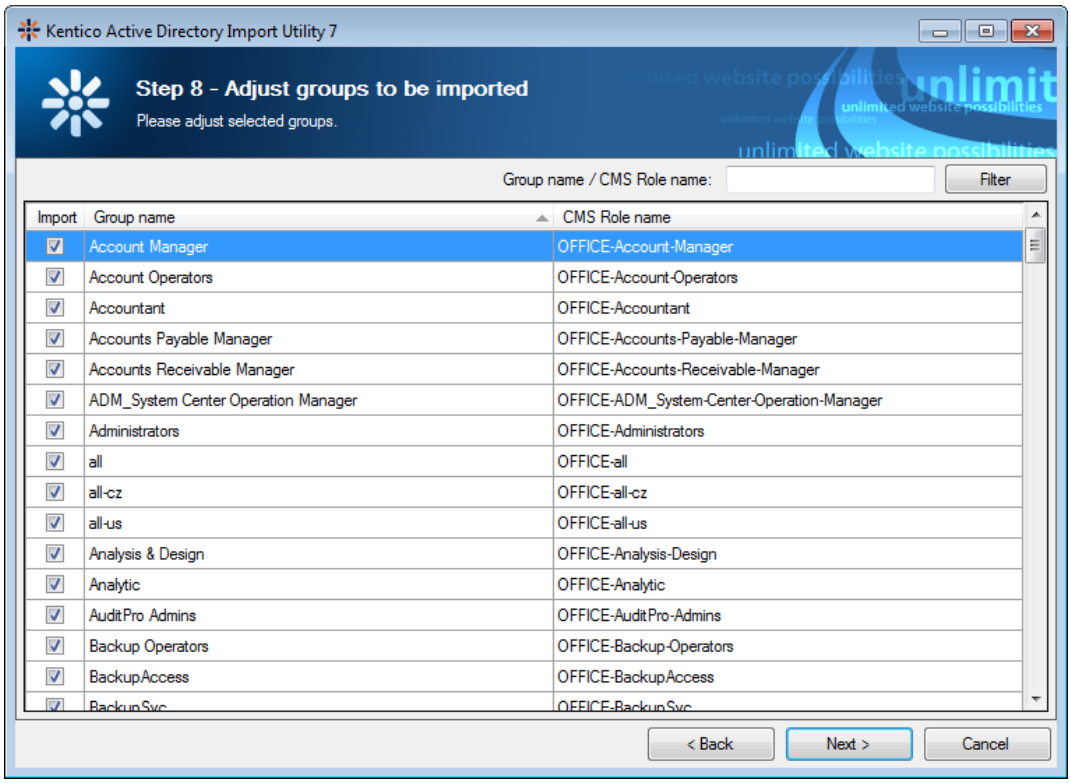
Step 7 – Adjust users to be imported

In this step, you can adjust the users to be imported. Users are selected based on settings made in the previous step, while you can adjust the selection using the check-boxes. You can also filter the listed users by *Display name* and *User name* using the filter above the list.



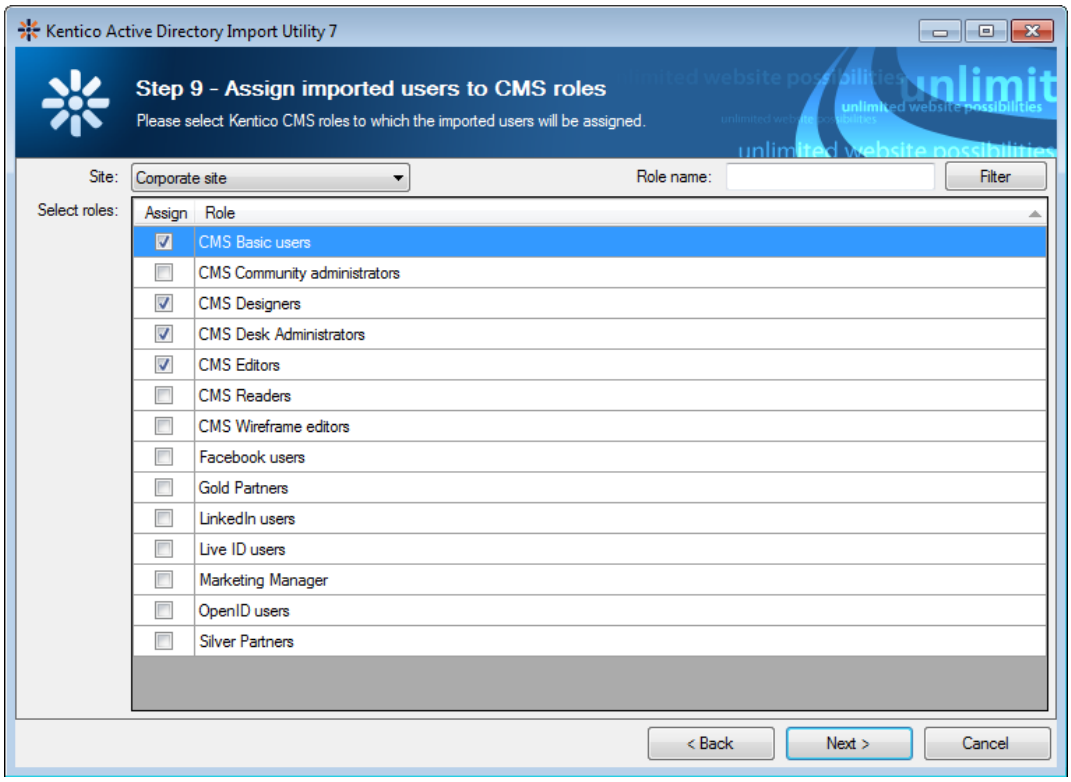
Step 8 – Adjust groups to be imported

This step is similar to the previous one, while groups (roles) to be imported can be adjusted here using the check-boxes. Listed groups can be filtered by *Group name* using the filter above the list.



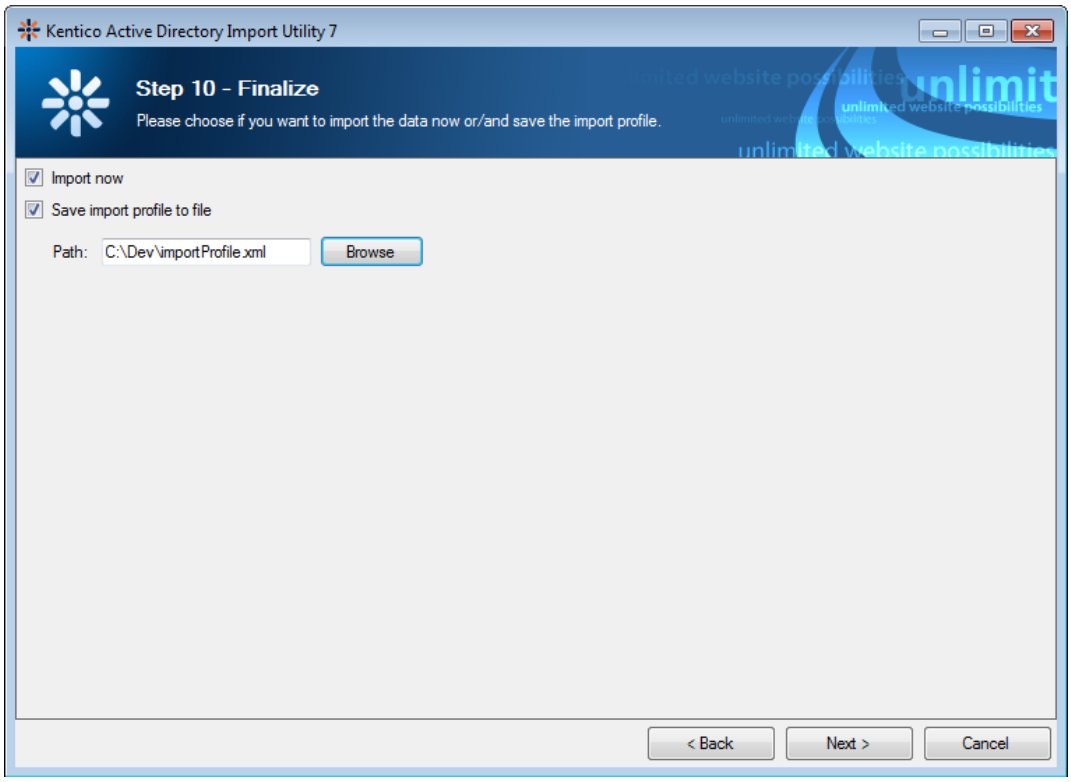
Step 9 – Assign to CMS roles

In the sixth step, you can select roles to which the imported users will be assigned. If you are importing to multiple sites, you first need to choose the site whose roles should be displayed using the **Site** drop-down.



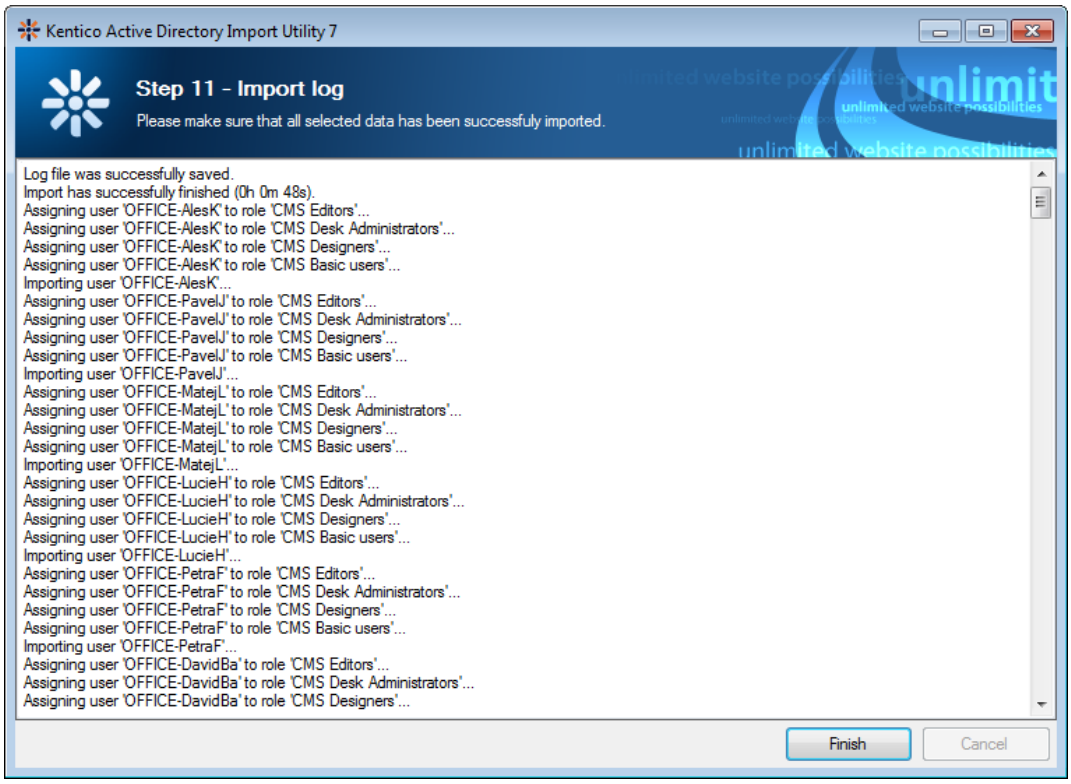
Step 10 - Finalize

Now you have your import profile configured. You can execute it immediately, save it into a file or perform both of these actions, depending on which of the **Import now** and **Save import profile to file** check-boxes is enabled.



Step 11 – Import log

The last step displays an import log, showing the progress of the import process. When the import finishes, you can close the wizard using the **Finish** button.



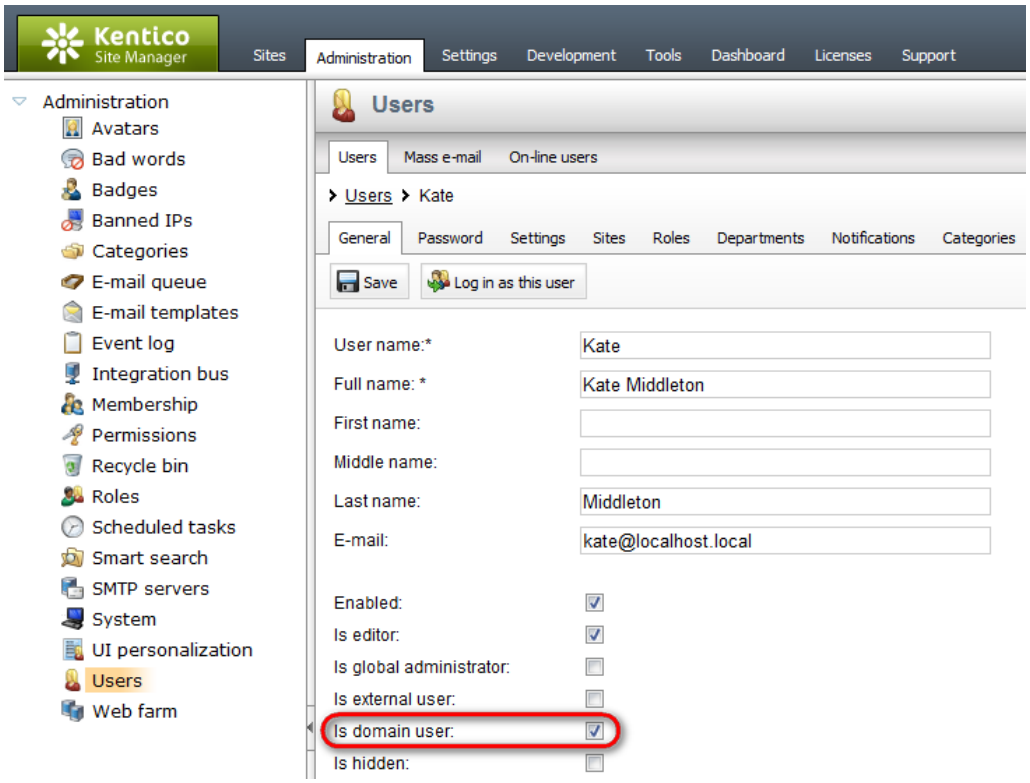
9.1.3 How to recognize imported users and roles

In the CMS, you can recognize users imported from AD by the **Is domain user** check-box on a user's **General** tab, as you can see in the screenshot below.

When editing roles, you can see the **Is domain role** check-box, which has the same meaning for roles.

These check-boxes reflect the values of the following Boolean fields in the database tables:

- **CMS_User -> UserIsDomain**
- **CMS_Role -> RoleIsDomain**



9.1.4 AD import from command line

Apart from the wizard described in the Using the wizard chapter, the AD Import Utility can also be launched from windows command line. It is achievable by executing the **ADImport.exe** file located in **<Kentico CMS installation folder>\Bin** (typically *C:\Program Files (x86)\KenticoCMS\<version number>\Bin*) using a special syntax.

You can launch the utility with the **-h** parameter, which displays help on using the utility from the command line:

```
ADImport -h
```

To perform the actual import, you need to have an import profile created via the wizard. With the import profile prepared, you need to execute the utility using the **ADImport /profile <profile file name>** syntax. When specifying the profile file name, an absolute or a relative path can be used. Please make sure that you use proper quotation when entering an absolute path containing special characters (e.g. blank spaces).

```
ADImport /profile my_profile.xml
ADImport /profile "C:\Temp\AD Import\my_profile.xml"
```

After executing the command, users or groups from Active Directory will be imported to your Kentico CMS instance based on settings contained in the specified import profile.

9.2 Kentico Import Toolkit

9.2.1 Overview

Kentico Import Toolkit is an external utility which enables import of data from external sources into Kentico CMS. As a source of data, you can use:

- **MS SQL database**
- **XML file**
- **CSV file**
- **XLSX file**

Data from these sources can be imported as:

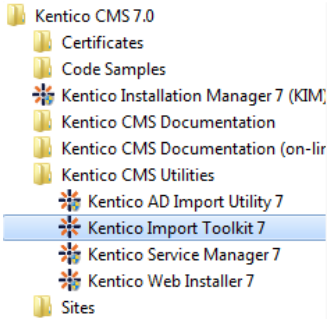
- **Documents**
- **Document attachments**
- **Objects**
- **Object attachments (metafiles)**
- **Custom table items**
- **On-line form items**
- **Resource strings**

To learn how to use the tool, proceed to the Initial steps topic and follow the step-by-step guide through the topics that follow.

9.2.2 Initial steps

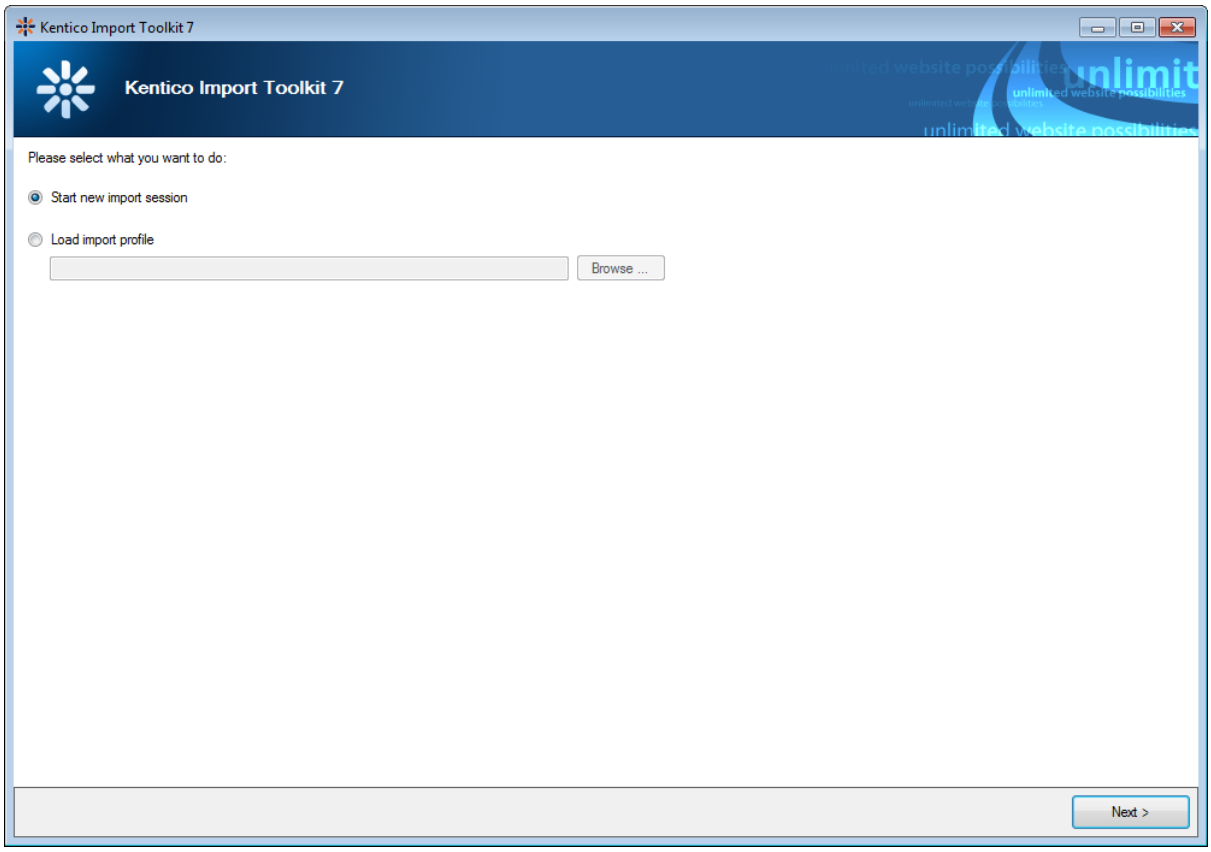
This topic describes the initial steps of the Kentico Import Toolkit wizard.

1. You can launch Kentico Import Toolkit from the Windows Start menu. In the Kentico CMS program group, open the Kentico CMS Utilities folder and choose **Kentico Import Toolkit**.



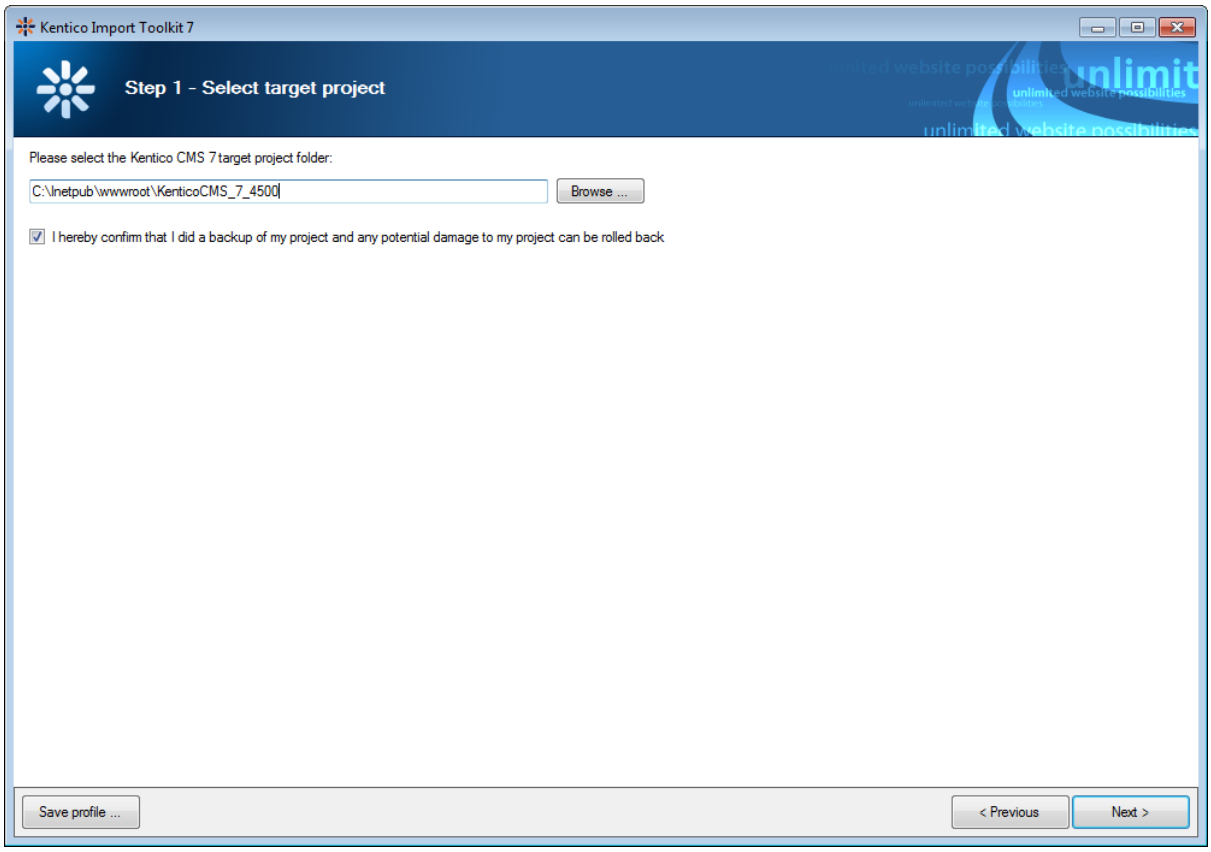
2. On the initial screen of the wizard, you can choose from the following options:

- **Start new import session** - starts a new import session where you can configure all the options of the import. In each of the configuration steps, you can click the *Save profile* button to save all the configuration done until that point. The configuration is saved as an *.iprofile* file in a destination of your choice.
- **Load import profile** - By selecting the *Load import profile* option, you can load an *.iprofile* file that you created when *importing a new session***.** All the configuration you have done prior to saving the configuration file will be pre-filled automatically.



Click **Next** to proceed when selected.

3. In **Step 1**, choose the root folder of the project into which you want to import your data. Either enter the path manually, or click **Browse** and search for the folder within your file system structure. In order to proceed, you also need to turn the check-box below the field on. By doing this, you confirm that you have performed a backup of your project so that you can roll back any potential damage caused by the import.



Click **Next** to proceed to the following step.

4. In **Step 2**, select the data type that will be used to store the imported data in Kentico CMS. You can make the selection by clicking the **Select data type to import** drop-down list and the optional drop-down lists below it:

- **Custom table items** - imports the data into a custom table selected in the *Import to custom table* drop-down list.
- **On-line form items** - imports the data as Form records of the on-line form selected in the *Import to on-line form* drop-down list.
- **Objects** - imports the data as Kentico CMS objects of a type selected in the *Import as* radio buttons. Three options are offered:
  - *Typical objects* - offers a set of typically used objects.
  - *All data objects* - includes all data objects except for many to many relationship objects.
  - *Many to many relationships* - provides a selection of objects with many to many relationship such as *contact list*, *group permissions* and *workflow step roles*.

  Additionally, you can reduce the selection of *objects* offered in the drop-down lists by adding optional keys to the **ImportToolkit.exe.config** file. You can find the configuration file in **<Kentico CMS installation folder>\Bin** (typically *C:\Program Files (x86)\KenticoCMS\<version number>\Bin*) . The keys accept object *ClassNames,* separated by semicolons, as their values. You can find *ClassNames, e.g.* in the *CMS_Class* table of your Kentico CMS database.

  Using the following key you can restrict the selection offered in the *Typical objects* drop-down list to *Product, Manufacturer* and *Supplier* only.

```
<add key="TypicalObjectTypes" value="CMS.Product;COM.Manufacturer;
COM.Supplier" />
```

By adding the following key, you can modify the selection of the *All data objects* drop-down list to contain only *Newsletter, Newsletter subscriber* and *Newsletter issue*.

```
<add key="AllObjectTypes" value="Newsletter.Newsletter;
Newsletter.Subscriber;Newsletter.Issue" />
```

You can limit the selection offered in the *Many to many relationships* drop-down list to *Workflow user, Workflow step and Workflow step user* with the following key.

```
<add key="MNRelationshipTypes" value="CMS.WorkflowUser;CMS.WorkflowStep;
CMS.WorkflowStepUser" />
```

- **Object attachments (metafiles)** - imports the data as metafiles of Kentico CMS objects. The *Attach to object type* drop-down list offers a selection of object types to which the metafiles will be attached.

  By adding the following key to the Import toolkit's configuration file, you can restrict the selection of the *Attach to object type* drop-down list to *Newsletter issue* and *Newsletter email template* only.

```
<add key="MetafileObjectTypes" value="Newsletter.Issue;
Newsletter.EmailTemplate />
```

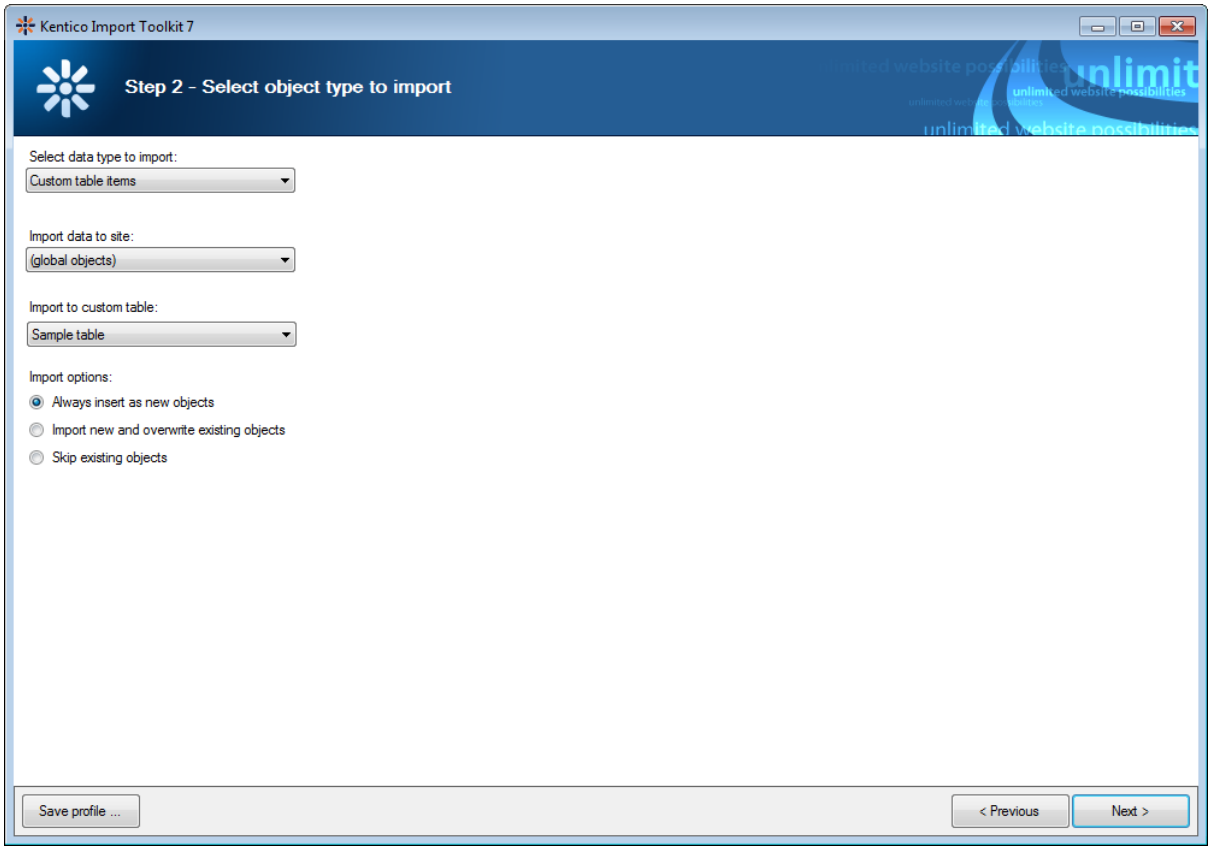
- **Documents** - imports the data as documents of the type selected in the *Import as* drop-down list. If you enable the *Automatically publish documents under workflow* option, imported documents will be automatically published when a workflow applies to them. By enabling the *Import as products* option, only product document types will be offered in the *Import as* drop-down list and E-commerce products will be created together with the product documents.
- **Document attachments** - imports the data as attachments of a document specified in Step 6 of the wizard.
- **Resource strings** - imports the data as resource strings into a culture selected in the *Import as* drop-down list.

With data types that can be site-related, the **Import data to site** drop-down list is displayed. Using this drop-down list, you can select a site to which the imported objects should be assigned. Where applicable, you can also choose **(global objects)** from the drop-down list to import the data as global objects not bound to a specific website.

Using the **Import options** radio buttons, you can determine the tool's behavior when it detects that some of the imported data already exists in Kentico CMS. Existing objects can either be detected automatically (based on code name or GUID), or by a WHERE condition specified in Step 6 of the wizard.

- **Always insert as new objects/documents** - all imported data will be inserted as new objects/documents, even if some of the data exists in Kentico CMS.

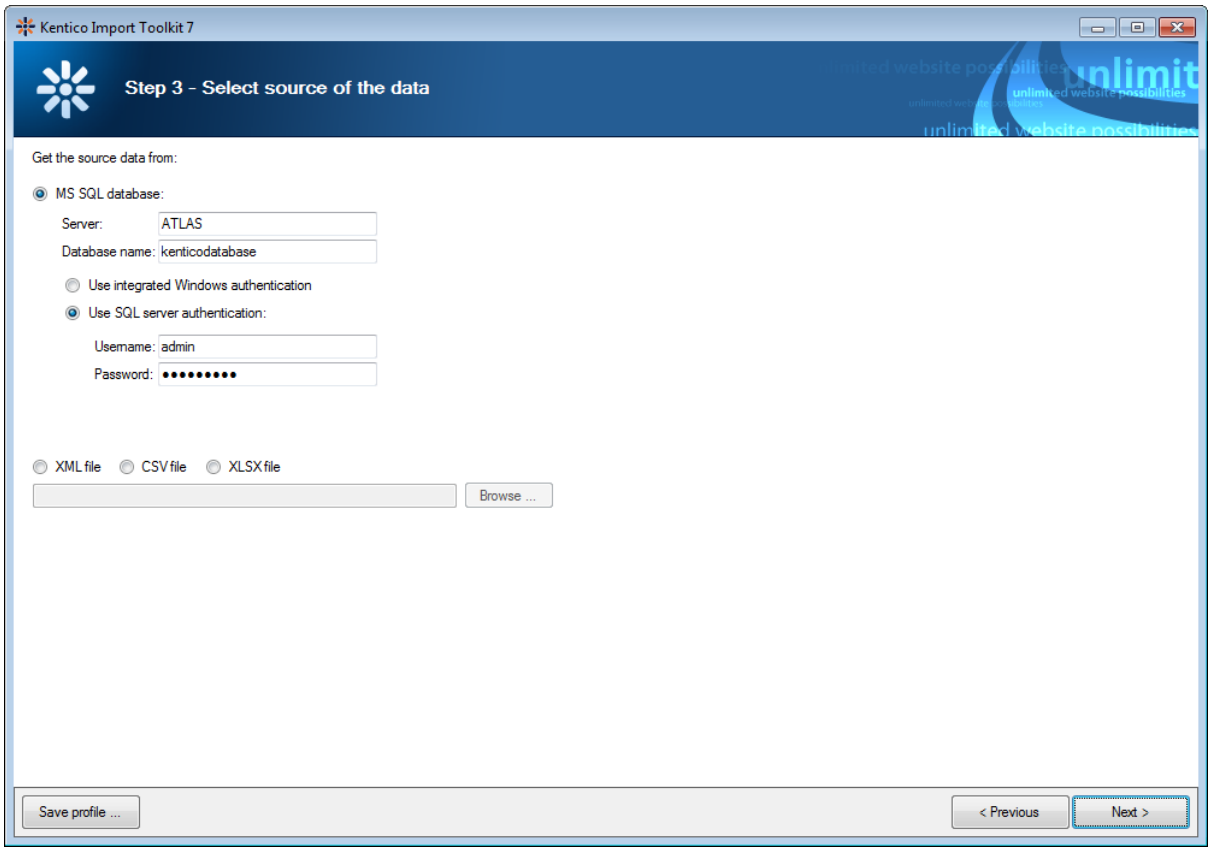
- **Import new and overwrite existing objects/documents** - all imported data will be inserted. Objects that already exist in Kentico CMS will be overwritten by the newly imported equivalents. This option must be selected if you wish to import new culture versions of an existing document (node).
- **Skip existing objects/documents** - existing objects/documents will be skipped and only new ones will be imported.



Once selected, click **Next** to proceed to the following step.

5. In **Step 3**, specify the source of the imported data. Using the radio buttons, you can choose to import from:

- MS SQL database - imports data from a specified MS SQL database.
  - **Server** - name of the database server containing the source database.
  - **Database name** - name of the source database.
  - **Use integrated Windows authentication** - the current user's Windows account will be used to log on to the database server.
  - **Use SQL server authentication** - logon credentials filled into the *Username* and *Password* fields below will be used to log on to the database server.
- XML file - imports data from an XML file specified in the field below.
- CSV file - imports data from a CSV file specified in the field below.
- XLSX file - imports data from an Excel spreadsheet specified in the field below.



Once selected, click **Next** to proceed to the following step.

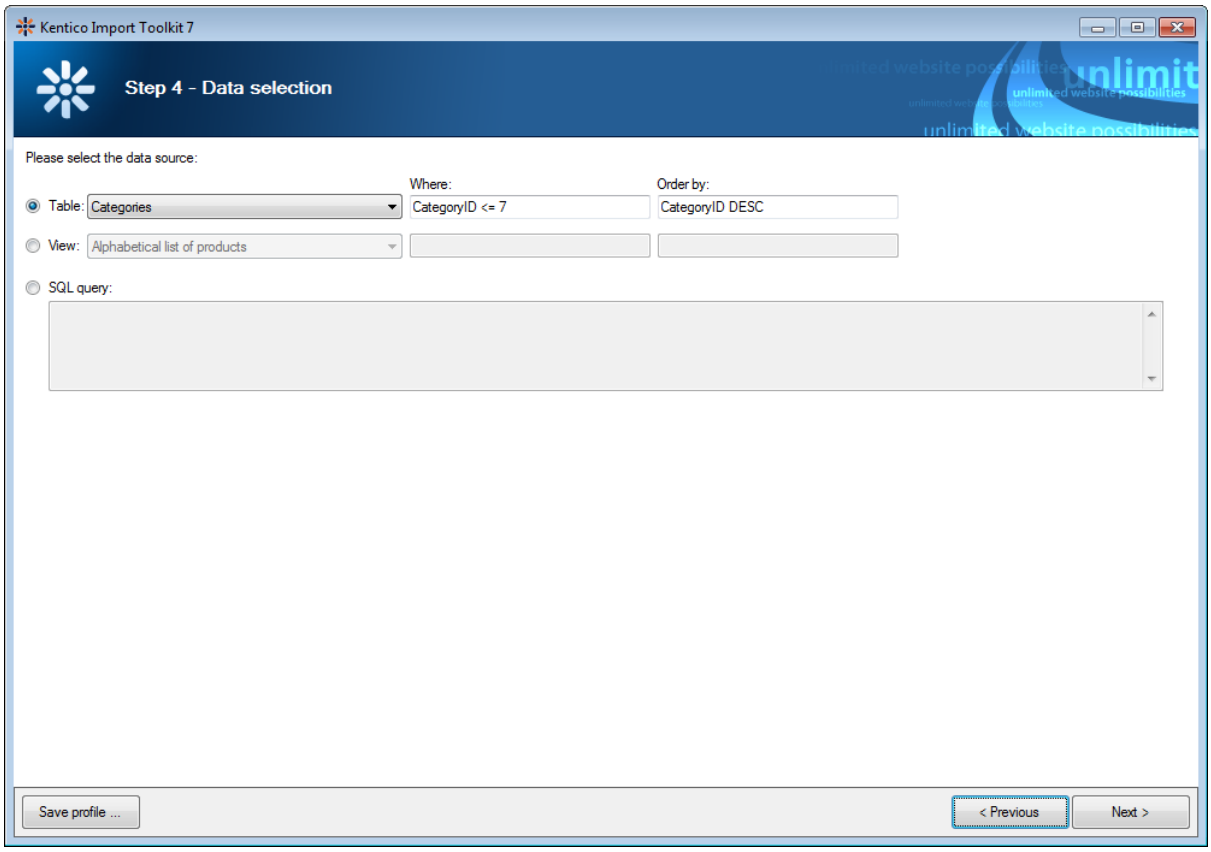
The rest of the wizard differs depending on the selected source of data. Click the links above to learn more about the following step once you select the respective source of data.

9.2.3 Data selection from MS SQL database

This topic contains information about the part of Kentico Import Toolkit wizard specific for import from an MS SQL database. The preceding steps of the wizard are explained in the Initial steps topic.

6. In **Step 4**, you can choose from the following options:

- **Table** - imports data from the table selected in the drop-down list. You can filter the imported data using a WHERE condition and sort them using an ORDER BY expression entered into the respective fields.
- **View** - imports data from the view selected in the drop-down list. You can filter the data you want imported using a WHERE condition and sort them using an ORDER BY expression entered into the respective fields.
- **SQL query** - imports data retrieved by a custom SQL query entered into the text area.



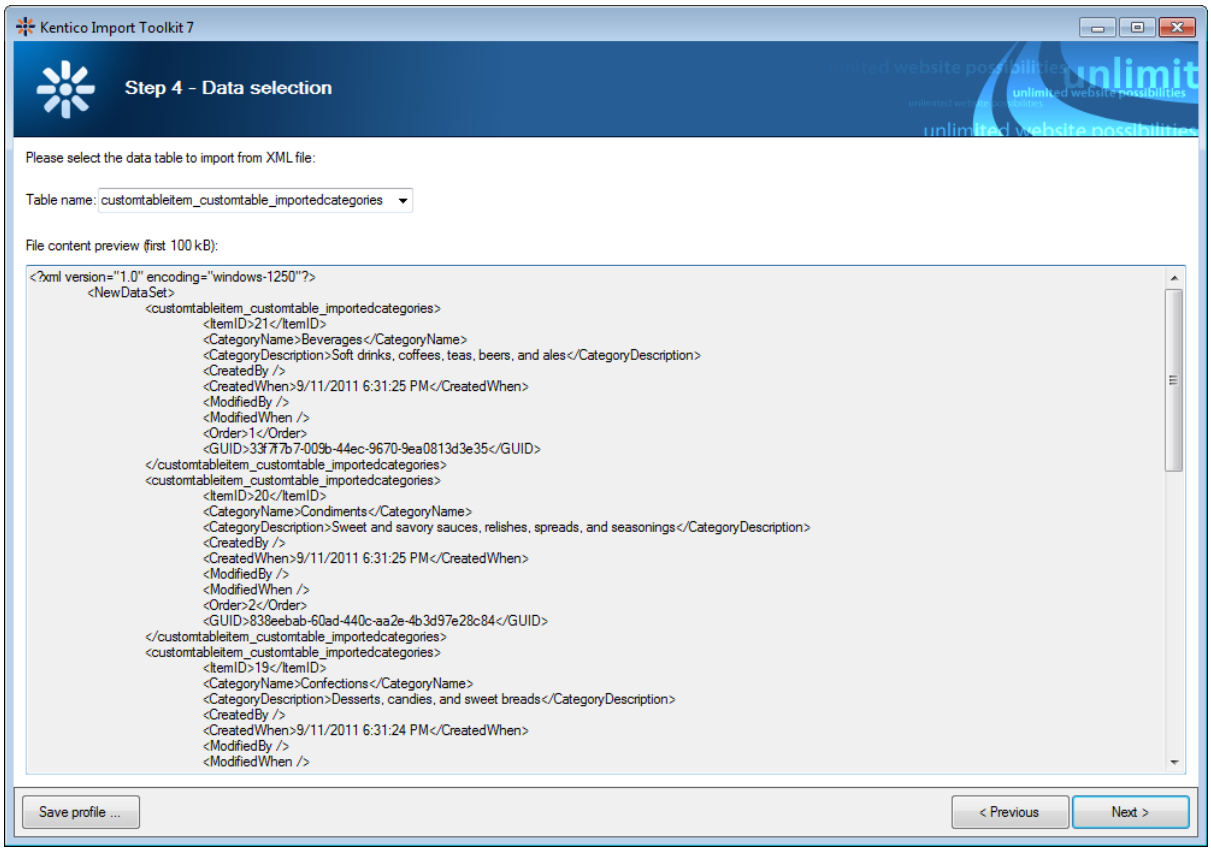
When specified, click **Next** to proceed to the following step.

For the rest of the wizard, which is common for all the imported data sources, see the Final steps topic.

9.2.4 Data selection from XML file

This topic contains information about the part of Kentico Import Toolkit wizard specific for import from an XML file. The preceding steps of the wizard are explained in the Initial steps topic.

6. In **Step 4**, use the **Table name** drop-down list to select which elements of the source XML file represent individual records to be imported. All its sub-elements will then be taken as individual data columns of the imported records.



Once selected, click **Next** to proceed to the following step.

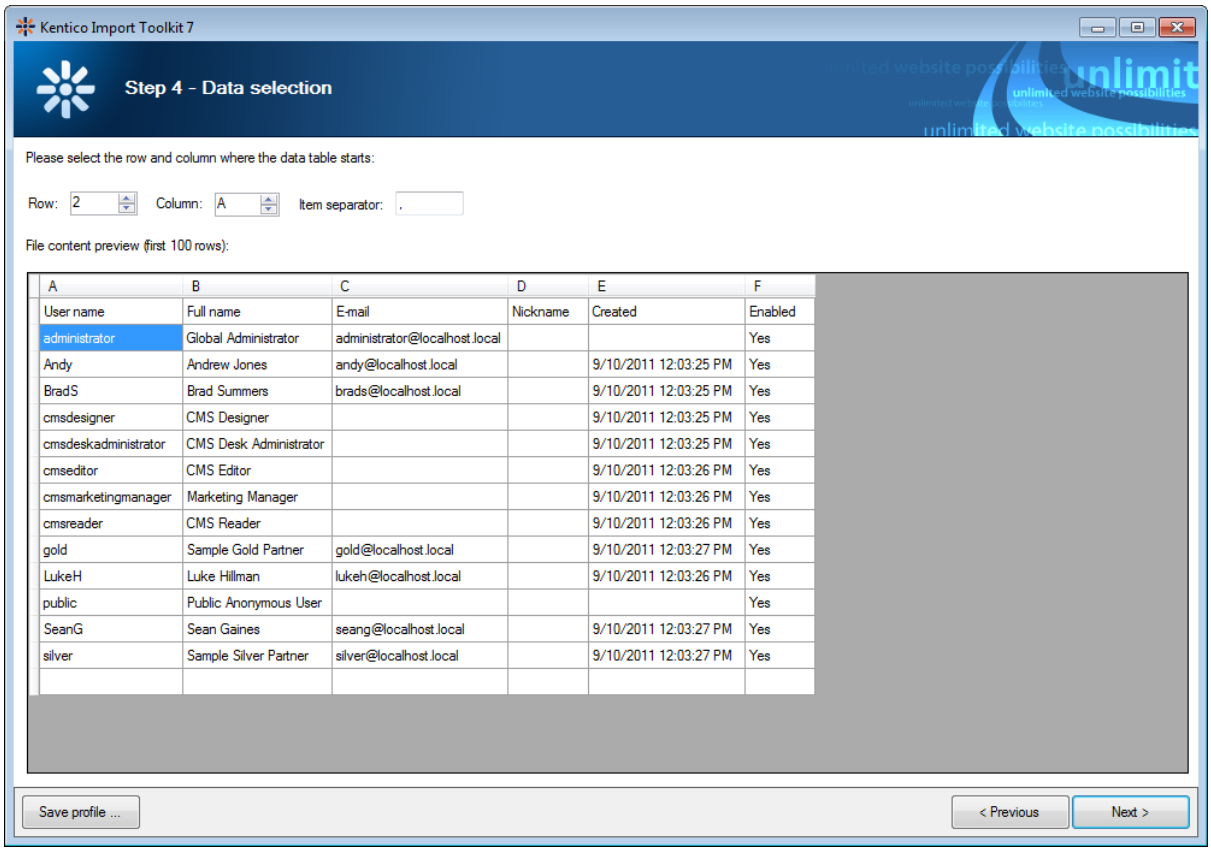
For the rest of the wizard, which is common for all the imported data sources, see the Final steps topic.

9.2.5 Data selection from CSV or XLSX file

This topic contains information about the part of Kentico Import Toolkit wizard specific for import from CSV or XLSX files. The preceding steps of the wizard are explained in the Initial steps topic.

6. In **Step 4**, choose which row and column of the file will be taken as the beginning of the data grid. This is useful for excluding non-data rows and columns (typically heading rows) from the imported data. The selection can either be done by specifying the row and column in the **Row** and **Column** fields above the grid, or simply by clicking the respective cell in the grid.

When importing from a CSV file, the **Item separator** filed is displayed next to the two fields mentioned above. Here, specify which character is used in the source CSV file as a separator between values in individual rows. A comma is used most typically, but you may come across CSV files that use different item separators, e.g., semicolons.



A	B	C	D	E	F
User name	Full name	E-mail	Nickname	Created	Enabled
administrator	Global Administrator	administrator@localhost.local			Yes
Andy	Andrew Jones	andy@localhost.local		9/10/2011 12:03:25 PM	Yes
BradS	Brad Summers	brads@localhost.local		9/10/2011 12:03:25 PM	Yes
cmsdesigner	CMS Designer			9/10/2011 12:03:25 PM	Yes
cmsdeskadministrator	CMS Desk Administrator			9/10/2011 12:03:25 PM	Yes
cmseditor	CMS Editor			9/10/2011 12:03:26 PM	Yes
cmsmarketingmanager	Marketing Manager			9/10/2011 12:03:26 PM	Yes
cmsreader	CMS Reader			9/10/2011 12:03:26 PM	Yes
gold	Sample Gold Partner	gold@localhost.local		9/10/2011 12:03:27 PM	Yes
LukeH	Luke Hillman	lukeh@localhost.local		9/10/2011 12:03:26 PM	Yes
public	Public Anonymous User				Yes
SeanG	Sean Gaines	seang@localhost.local		9/10/2011 12:03:27 PM	Yes
silver	Sample Silver Partner	silver@localhost.local		9/10/2011 12:03:27 PM	Yes

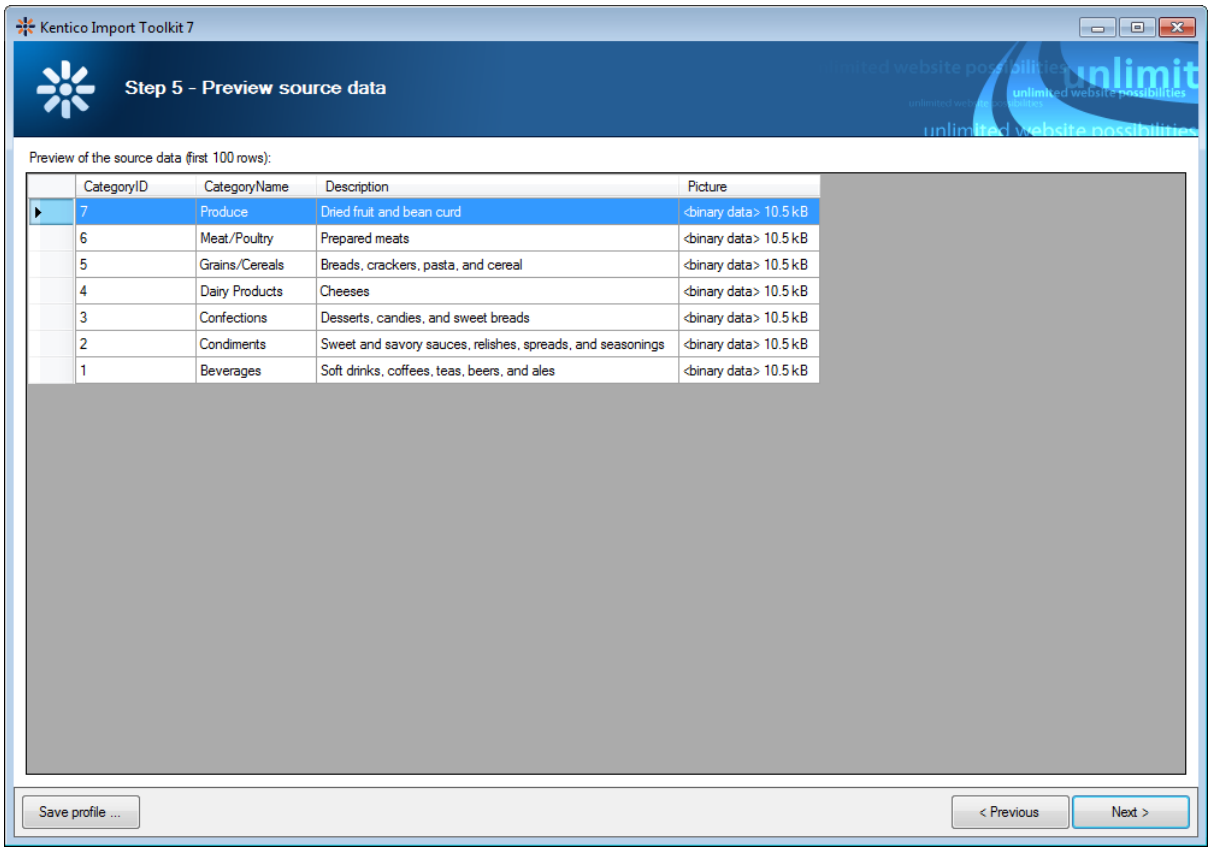
When selected, click **Next** to proceed to the following step.

For the rest of the wizard, which is common for all the imported data sources, see the Final steps topic.

9.2.6 Final steps

In this topic, you will find information on the final steps of the Kentico Import Toolkit wizard. To learn about the preceding steps, refer to the Initial steps topic and the topics about data selection for import from an MS SQL database, XML file or CSV and XLSX files.

7. In **Step 5**, you can see a preview of the data to be imported. This step is only informative and you can use it to check if you specified the source data correctly in the previous steps.



Click **Next** to proceed to the next step.

8. In **Step 6**, perform mapping of source fields to target fields. The utility automatically recognizes fields that appear to be equivalent, so if names of the fields or at least their suffixes match, most of the fields should be mapped automatically. You then only need to check and fine-tune the mappings. If you want to return to the default mappings after adjusting them, click **Reset to default mappings** above the grid.

The mappings grid contains the following columns:

- **Target field** - field of the target object/document into which the data should be imported. If you turn on the **Show advanced columns** check-box above the grid, additional columns that you typically wouldn't use as an import target are offered as well.
- **Required** - indicates if the target field is required.
- **Source field or expression** - source field from which data will be imported into the target field. You can use the following expressions in the text to get the values dynamically:
  - **=macroexpression** - evaluates the specified macro expression using the source data and uses its result in the value. For example, *={%ItemText%} - {%ItemOrder%}*.
  - **#<source>query** - executes a query against the source database and uses the result in the field. For example, *#<source>SELECT TOP 1 UserID FROM CMS_User WHERE UserName ='{% SourceUserName%}'* .
  - **#<target>query** - executes a query against the target database and uses the result in the field. For example, *#<target>SELECT TOP 1 ...* .
  - **#<file>** - you can specify a file using a URL or a disk path. Binary data of the file will be used as the value of the target field.
- **Default value** - value that is used if the source field does not contain any value (or when its value is NULL).

- **FK mapping file** - you can specify an *.fkmap* file from a previous export session which converts an ID field to a new field mapped in the previous session. See the following text for more details.

If you selected to skip or overwrite existing objects in Step 2 of the wizard, you can specify a WHERE condition in the field below the list. Based on this, the existing objects to be skipped or overwritten will be identified.

When importing the data as documents, you can see an extra field below the grid. Use the field to specify the path or NodeID of the document under which you want to create the imported documents. When importing the data as document attachments, there is also an extra field below the grid. In this case, you need to specify the path to the document to which you want to attach the imported attachments.

Importing new culture versions of an existing document

If you wish to use the import toolkit to add language versions to an existing node in the content tree, you also need to specify the appropriate NodeID using the extra field. This can be done through the following expression:

#<target>SELECT TOP 1 NodeID FROM View_CMS_Tree_Joined WHERE NodeAliasPath ='{%NodeAliasPath%}'

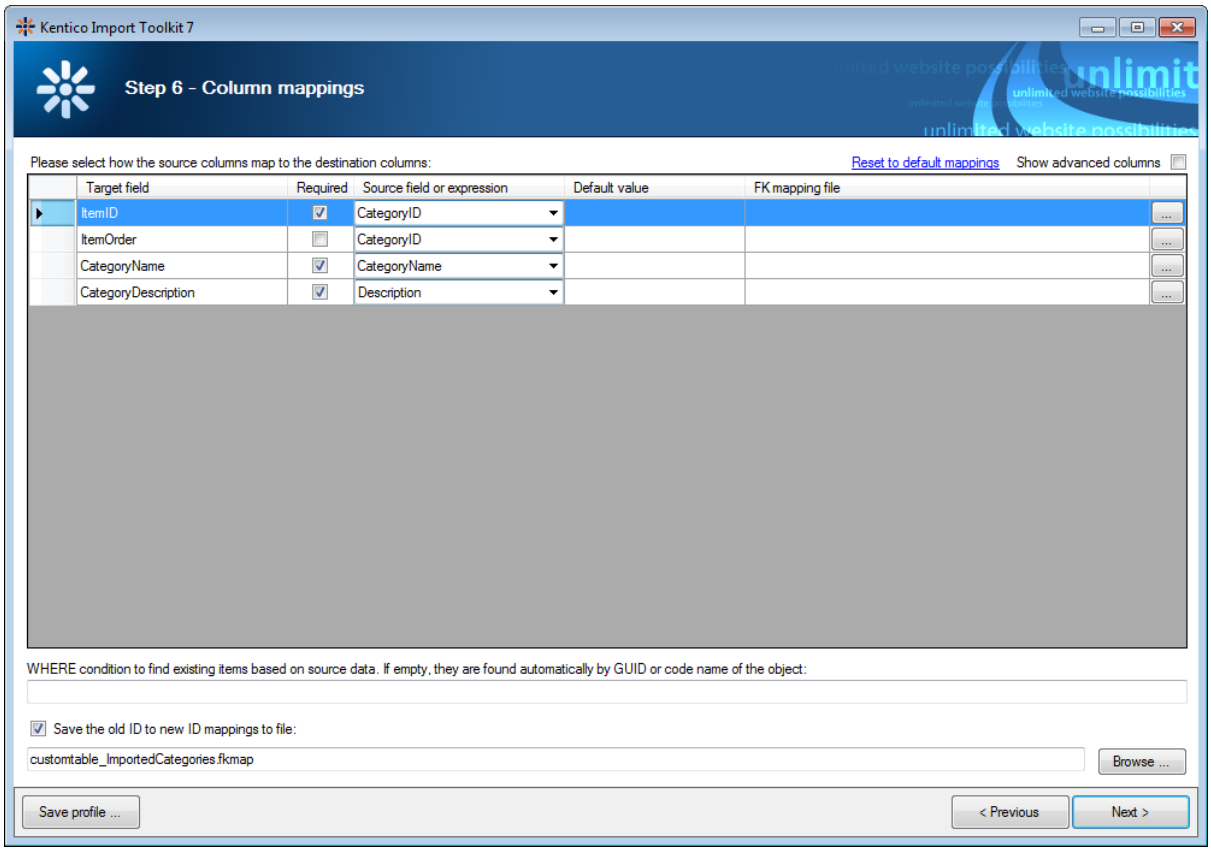
As described for the **Source field or expression** fields above, the *#<target>* query is used to select the NodeID of the document that should be updated from the target database.

The *{%NodeAliasPath%}* macro is resolved into the alias path of your document versions in the source data. Because all language versions of the same document share the same alias path, this allows you to find the appropriate NodeID in the target database.

Please note that the **Import new and overwrite existing documents** option must be selected in step 2 of the wizard in order for this to work correctly.

You can also enable the **Save the old ID to new ID mappings to file** check-box. This enables you to save mappings of source ID columns to target ID columns into an *.fkmap* file that can be used during another session in the FK mapping file column above. The following example explains the use of the file:

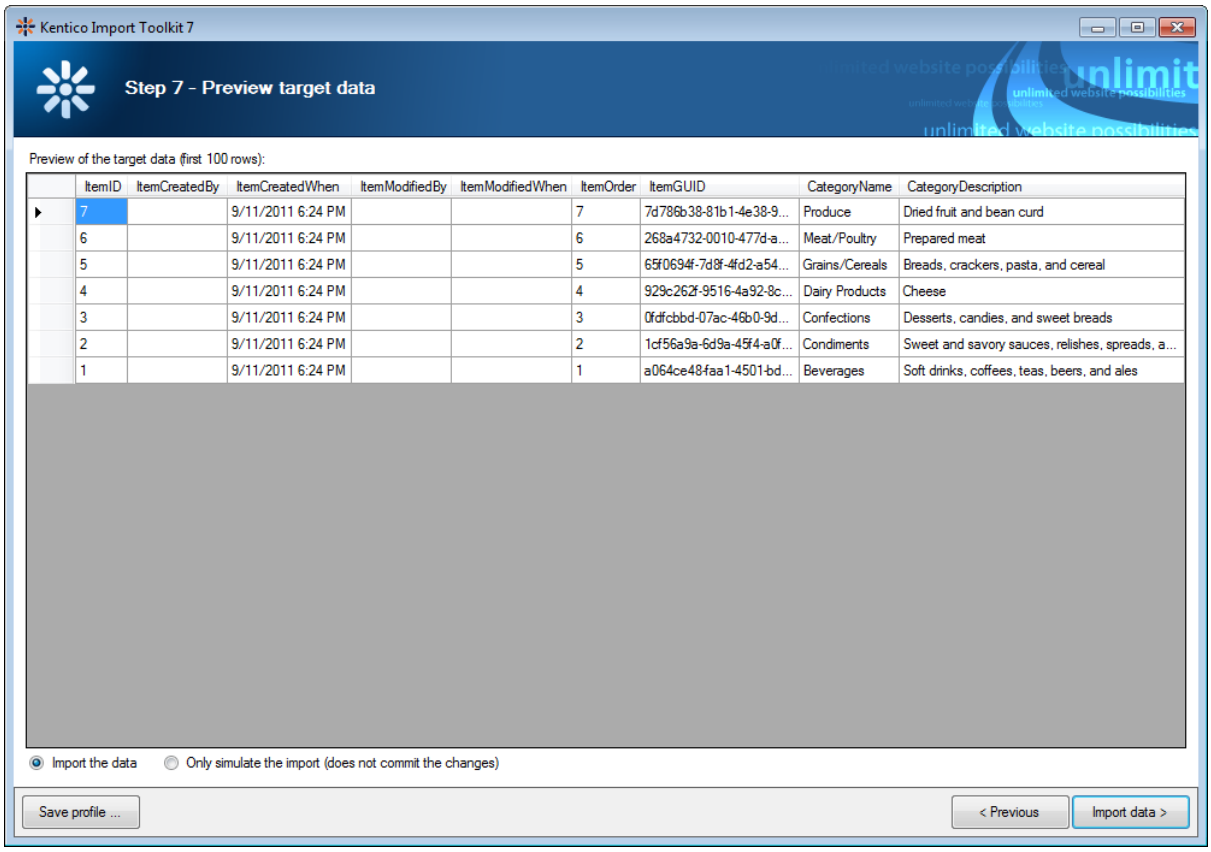
- You have a list of questions and a list of answers, where each answer is bound to a question with a foreign key.
- You want to import both lists and preserve their bindings.
- First, import the questions and save the ID mapping to a *questions.fkmap* file.
- Then import the answers and use the *questions.fkmap* file in the FK mapping file column of the field that contains the foreign key.



Once you have the mappings specified, click **Next** to proceed to the following step.

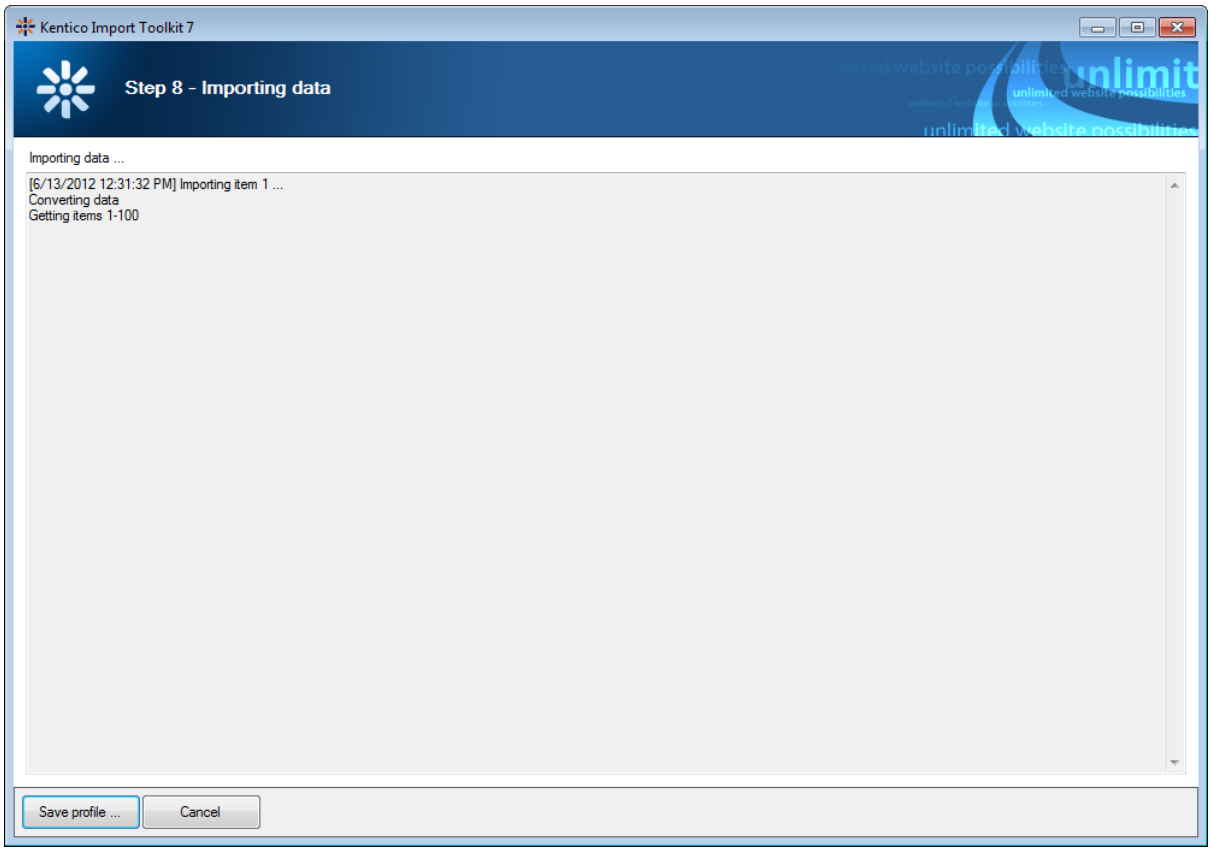
9. In **Step 7**, you can see a preview of the data as it will look when imported to the target. Before proceeding, you can choose how you want the data to be imported:

- **Import the data** - imports the specified data into Kentico CMS.
- **Only simulate the import (does not commit the changes)** - performs the import to validate that the data is correct, but does not commit the changes to the database.



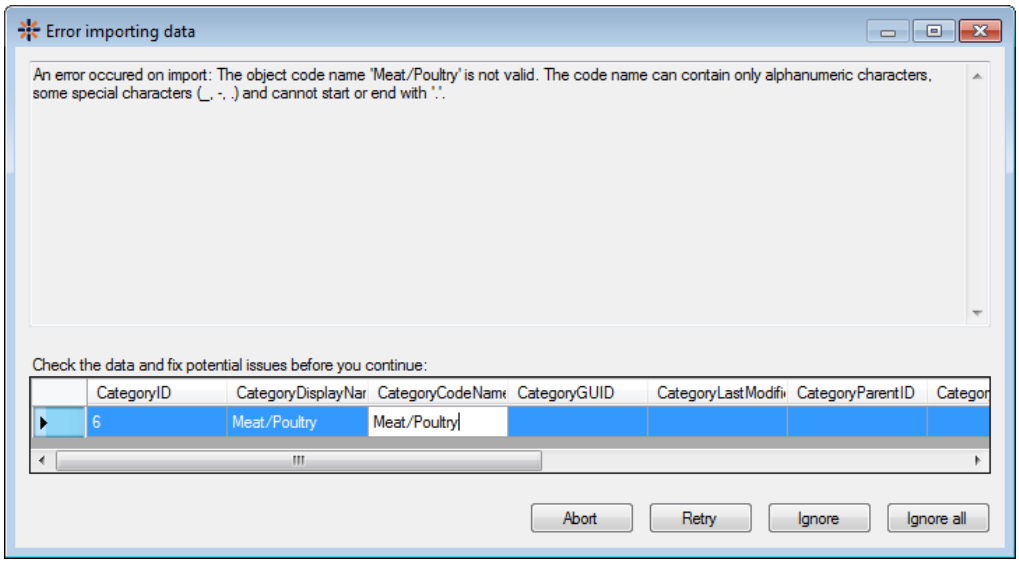
After making your choice, click the **Import data** button to perform the import.

10. A log will be displayed, showing you the progress of the import.



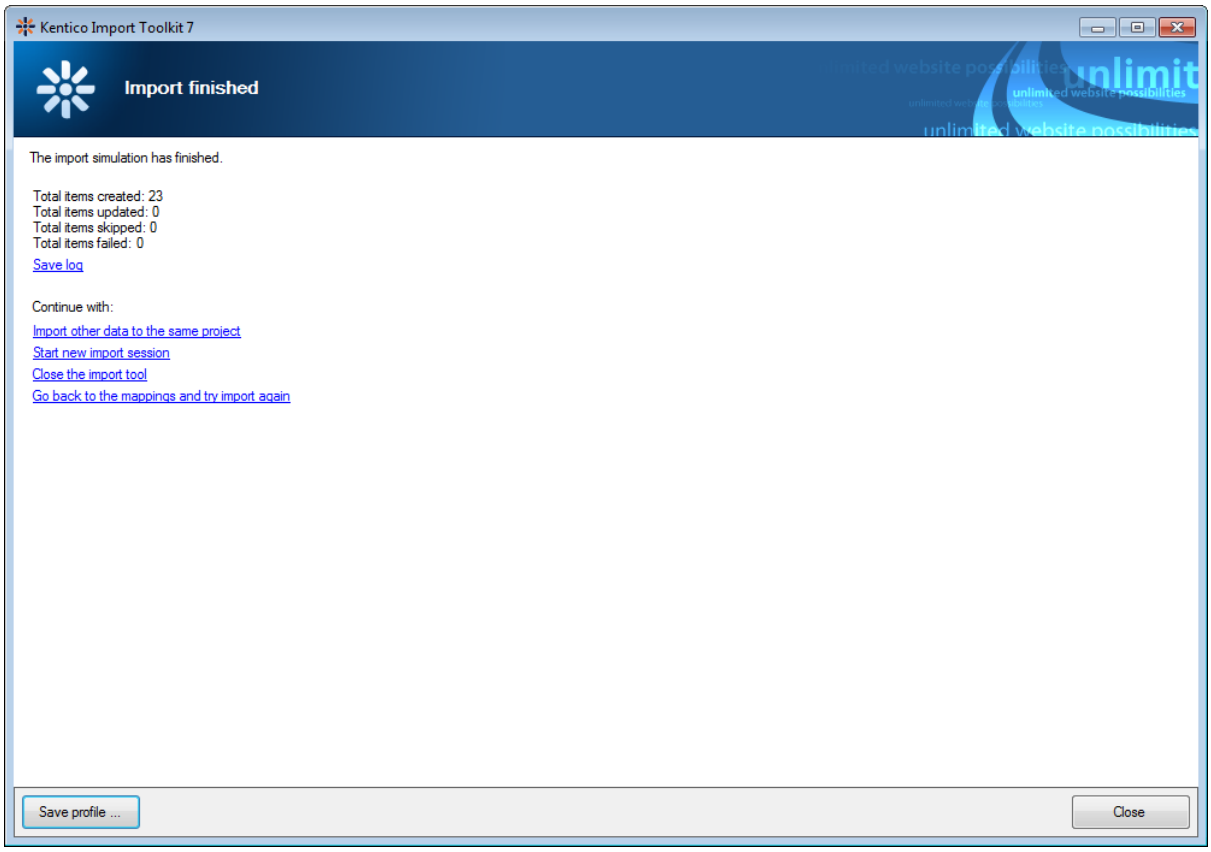
11. If an error occurs during the import, e.g., when there are restricted characters in imported data, the dialog that follows is displayed. The dialog displays an error message explaining the issue and provides you with the following options:

- **Abort** - aborts the whole import process.
- **Retry** - you can correct the data in the displayed row and click this button to import the corrected data — without the need of going through the import process again.
- **Ignore** - the current record will be skipped and not imported.
- **Ignore all** - all records containing errors will be skipped during the rest of the import.



12. The final step informs you of the results of the import. Imported data should already be visible in Kentico CMS at this point. You can perform the following actions:

- **Save log** - saves the import log created while the import was being performed.
- **Import other data to the same project** - redirects you to Step 2 of the wizard where you can select object type to import for the same target project.
- **Start new import session** - redirects you to Step 1 of the wizard where you can start a new import session.
- **Close the import tool** - closes the utility.
- **Go back to the mappings and try import again** - this options becomes available if you selected the *Only simulate import* option in Step 7 or if you chose to *Abort* the import process.



9.2.7 Import Toolkit from command line

Apart from using the wizard described in the Initial steps chapter, you can also run the Import Toolkit utility from the command line.
To perform the actual import, you need to have an import profile file. You can run the utility by executing the **ImportToolkit.exe** file located in **<Kentico CMS installation folder>\Bin** (typically *C:\Program Files (x86)\KenticoCMS\<version number>\Bin*) using a special syntax.

Important!

The profile file used in the command line needs to be created in the last step of the wizard—once the import has finished. This way the file will contain all the required data for a successful import.

You can launch the utility with an optional **-i** parameter, which ignores errors during the import:

```
ImportToolkit <path to your i.profile file> -i
```

You can specify either a relative or an absolute path to the *.iprofile* file. Make sure you use proper quotation when entering an absolute path containing special characters ,e.g., blank spaces:

```
ImportToolkit my_profile.xml
ImportToolkit "C:\Temp\AD Import\my_profile.xml"
```

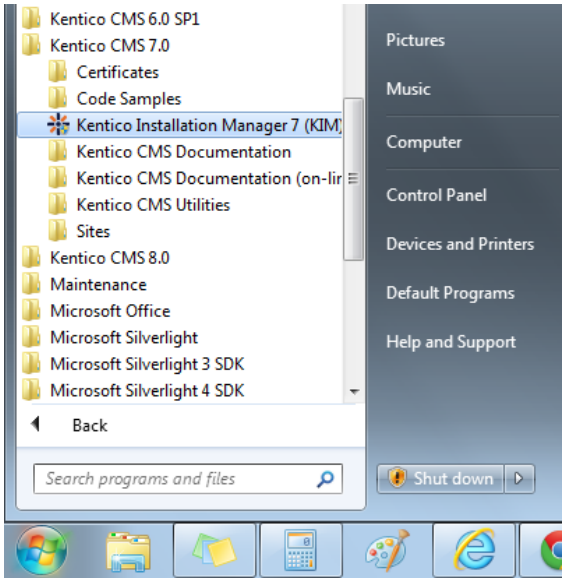
After executing the command, data will be imported to your Kentico CMS instance based on settings contained in the specified import profile.

9.3 Kentico Installation Manager

9.3.1 Overview

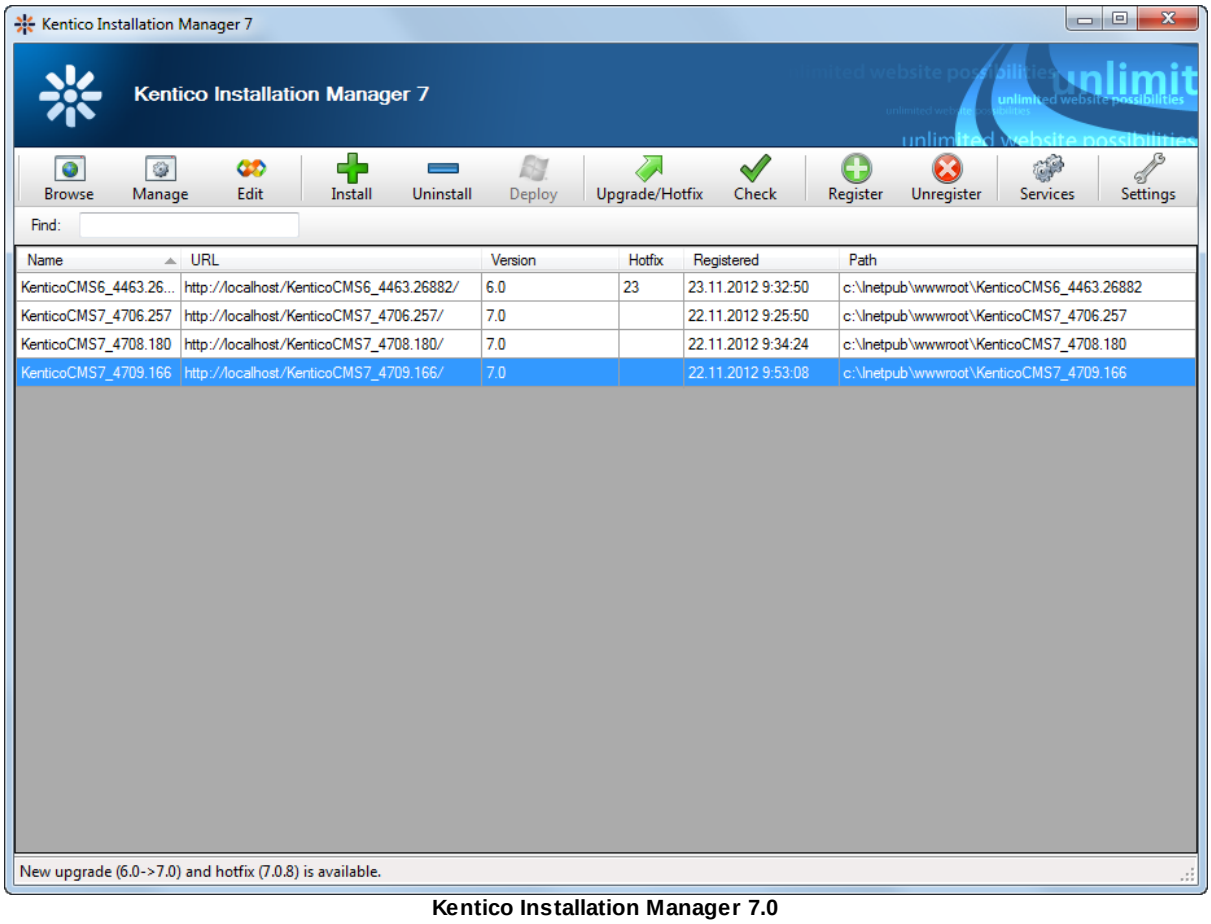
Kentico Installation Manager (also referred to as **KIM**) is an external utility which enables installation and management of Kentico CMS instances. You can manage all instances installed on a single machine using this tool, which is especially useful on machines with a large number of instances.

You can launch this utility by selecting **Kentico Installation Manager** from the Kentico CMS folder in Windows Start menu.



Running KIM 7.0 in Windows

KIM 7.0 interface



Kentico Installation Manager 7.0

Browse - opens the URL in the URL column of the selected instance in a new browser window.
Manage - opens the Site Manager of the selected instance in a new browser window.
Edit - opens the selected instance's web project in Visual Studio.
Install - opens Kentico Web Installer, which you can use to install a new Kentico CMS instance.
Uninstall - uninstalls the selected instance, optionally including a removal from the file system, IIS, database and from the instances list in KIM.
Deploy - deploys the selected instance to Windows Azure. See Deployment to Azure.
Upgrade/Hotfix - opens a dialog box where you can upgrade or hotfix the currently selected instances. See Upgrading and hotfixing for more details.
Check - checks if there are any new upgrades or hotfixes available for any of the managed instances. See Upgrading and hotfixing for more details.
Register - opens a dialog where you can register a new instance for management by KIM (add it to the list). See Instance registration for more details.
Unregister - unregisters the selected instance from KIM (removes it from the list). See Instance registration for more details.
Services - opens Kentico Service Manager, where you can install and manage Kentico CMS Windows services.
Settings - opens a dialog where you can adjust settings related to upgrading and hotfixing. See Upgrading and hotfixing for more details.

In cases where large numbers of instances are registered in the utility, you can use the **Find** text box below the toolbar to search for particular instances by the value stored in their **Name** column.

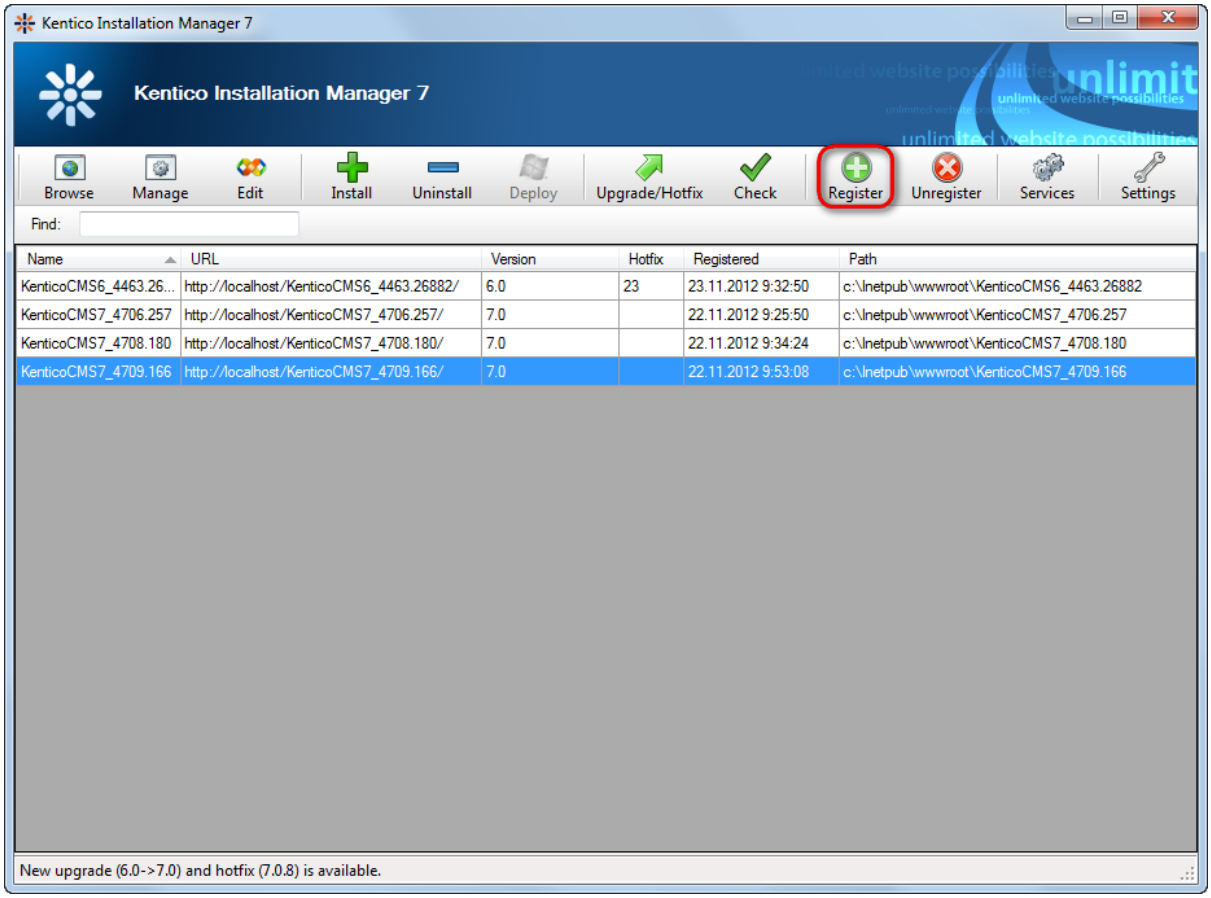
9.3.2 Instance registration

Kentico Installation Manager is able to register installed Kentico CMS instances automatically. In some cases though, the registration process may not complete successfully and you will have to register the instances manually. Such cases can include:

- You have Kentico CMS instances of versions prior to 6.0 already installed on your machine.
- You installed a Windows Azure project.

To register an instance in KIM:

1. Click **Register** on the main toolbar.

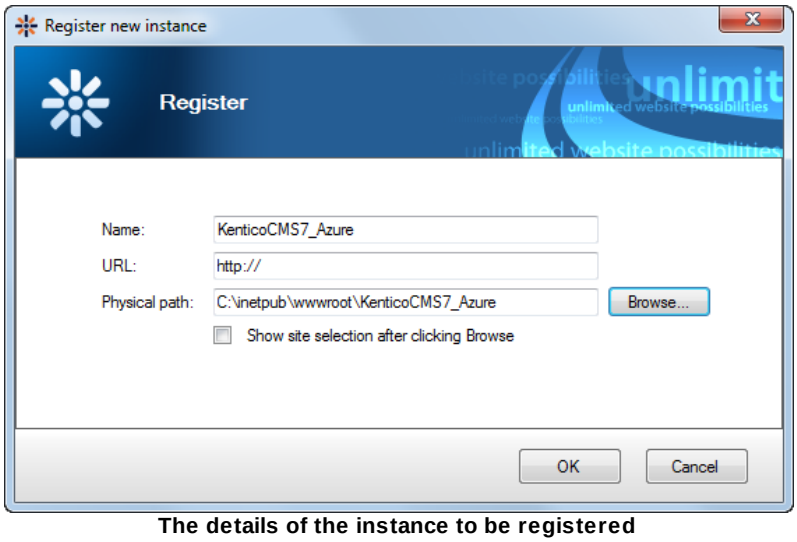


Registering a new KenticoCMS instance

A new dialog box appears.

2. Enter the following details:

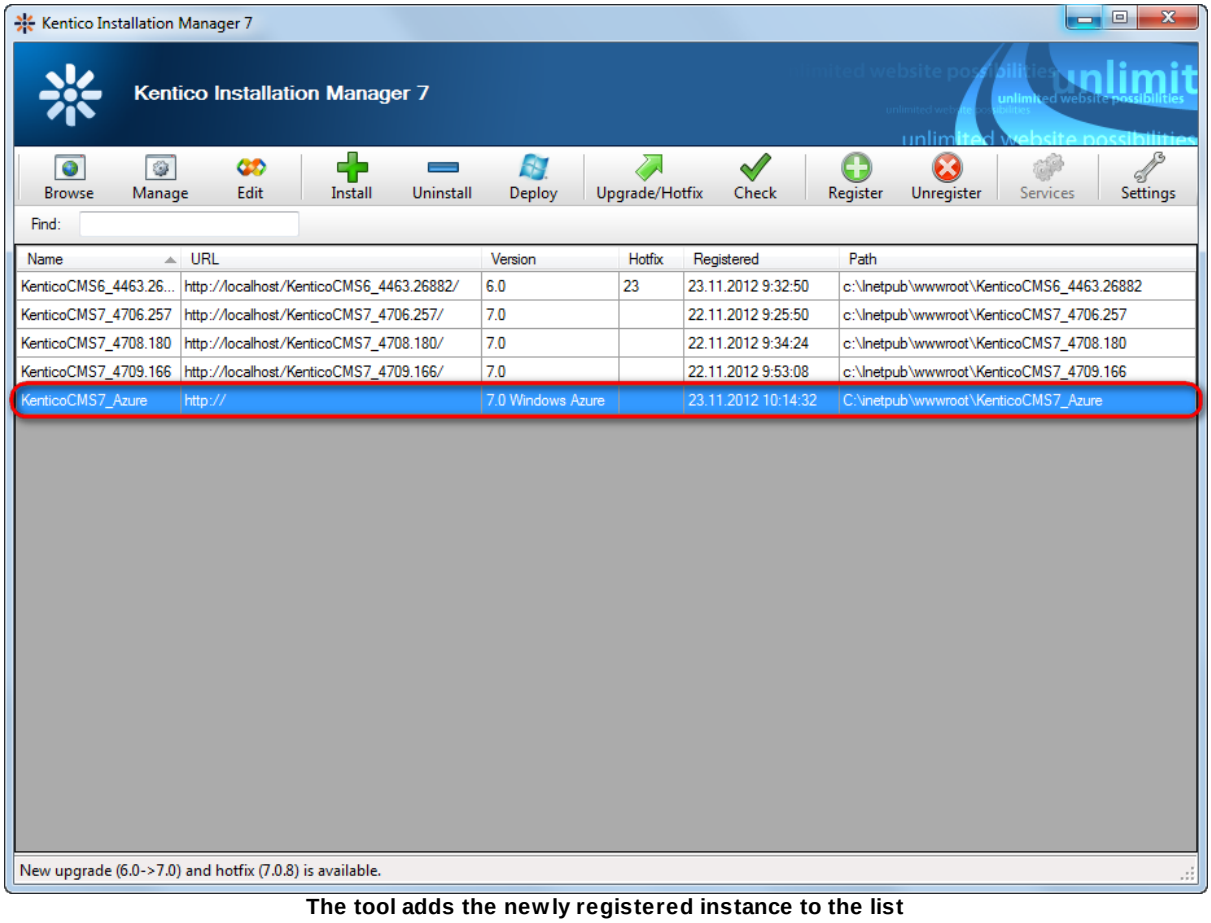
- **Name**: identifying name of the instance displayed in the **Name** column in the list of instances.
- **URL**: URL under which the instance is accessible.
- **Physical path**: path to the instance's web project root folder in the file system.
- **Show site selection after clicking Browse**: if enabled, the tool displays a dialog box with available web sites of the particular instance after clicking the **Browse** button (useful for multi-site instances).



The details of the instance to be registered

3. Click **OK**.

The KIM registers your instance and displays it in the list of managed instances.



The tool adds the newly registered instance to the list

Where is the information stored

Information about registered instances is stored in **c:\ProgramData\KIM\kim.xml**. Within the root <sites> element, it contains a number of <item> elements which represent particular registered instances. The <item> elements have the following sub-elements representing properties of the registered instances:

- **<guid>** - unique identifier of the instance. Its value is taken from the **CMSApplicationGuid** key in the *appSettings* section of the instance's *web.config* file.
- **<url>** - URL under which the instance is accessible.
- **<name>** - identifying name of the instance displayed in the **Name** column in the list of instances.
- **<version>** - version of Kentico CMS.
- **<path>** - path to the instance's web project root folder in the file system.
- **<created>** - date and time when the instance was registered in Kentico Installation Manager.
- **<hotfix>** - number of the latest hotfix applied to the instance.
- **<netversion>** - version of .NET Framework used by the instance. The number of .NET version does not correspond to the number stored here, they are encoded in the following way:

.NET version	Encoding number
2	2
3.5	4
4	8
4.5	16

- **<showlist>** - indicates if a dialog box with available web sites of the particular instance is displayed after clicking the **Browse** button.

Here is a sample extract from the xml file:

```
<?xml version="1.0"?>
<sites>
  <item>
    <guid>7d6b7a6e-5f3a-4403-97d0-970dcabba91d</guid>
    <url>http://localhost/KenticoCMS7</url>
    <name>Kentico CMS 7</name>
    <version>7.0</version>
    <path>c:\inetpub\wwwroot\KenticoCMS7</path>
    <created>2012-10-30T11:52:59</created>
    <hotfix>0</hotfix>
    <netversion>8</netversion>
    <showlist>1</showlist>
  </item>
  ...
</sites>
```

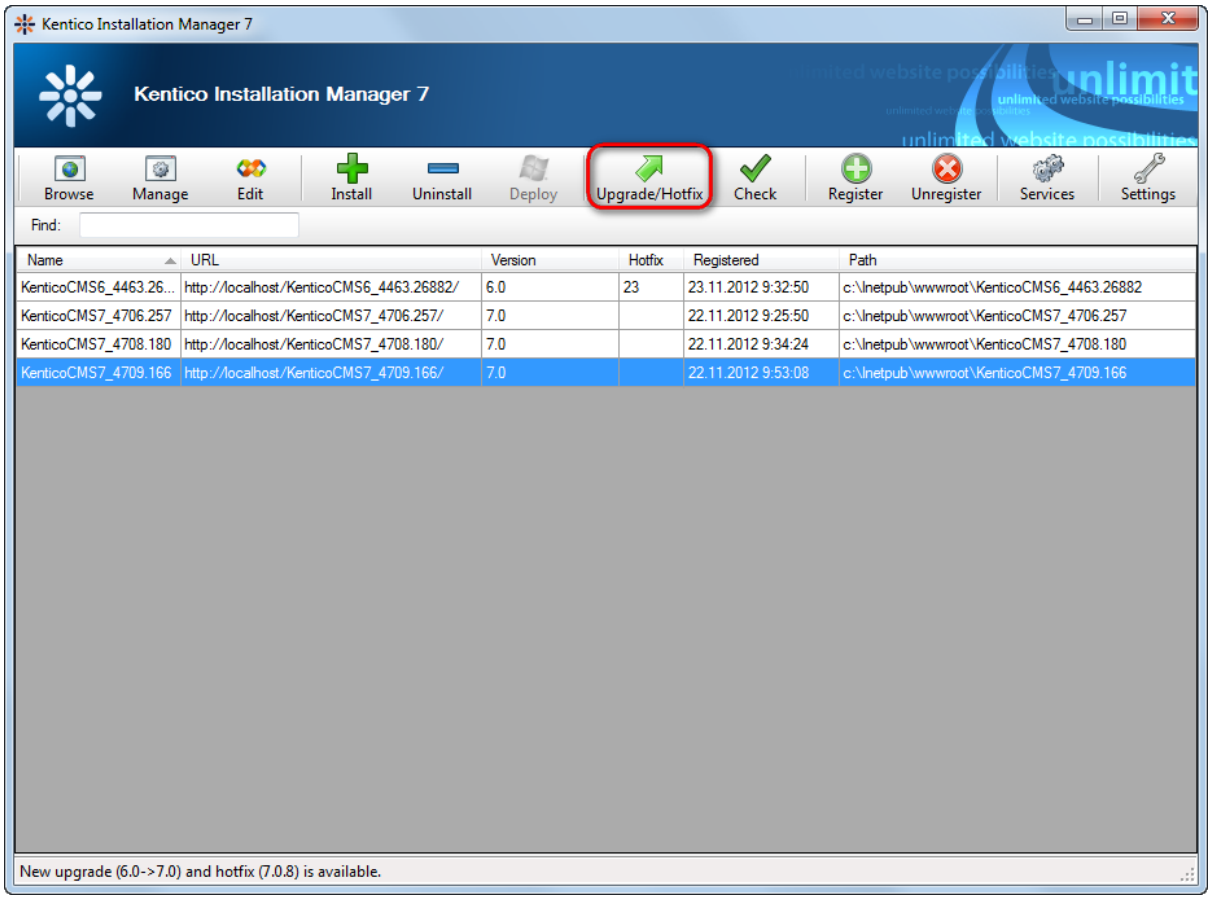
9.3.3 Upgrading and hotfixing

You can upgrade and hotfix Kentico CMS instances using Kentico Installation Manager. There are two actions related to these tasks on the main panel:

- **Upgrade/Hotfix** - opens a dialog box, where you can upgrade or hotfix the currently selected instance.
- **Check** - checks if there are any new upgrades or hotfixes available for any of the managed instances.

Upgrading and hotfixing

1. Select an instance (or multiple instances) you want to upgrade or hotfix.
2. Click **Upgrade/Hotfix**.

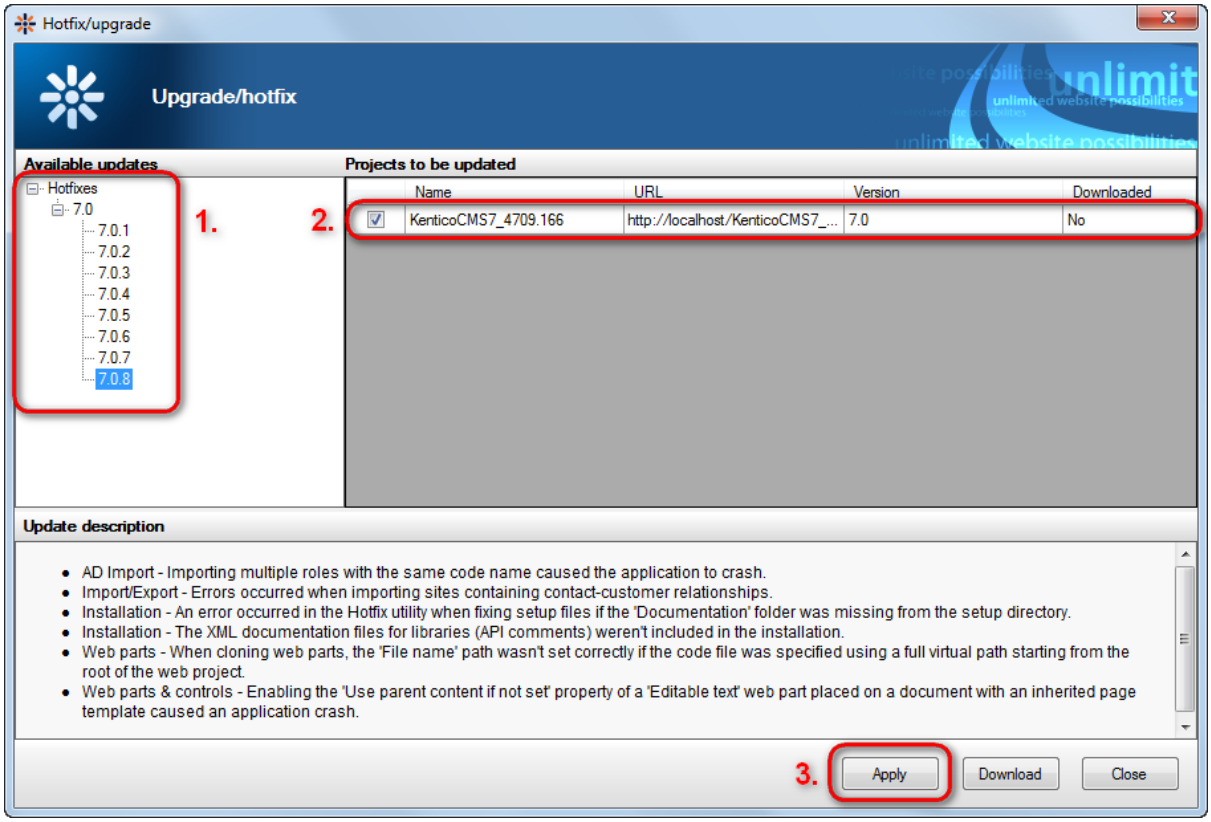


Upgrading/hotfixing an instance

KIM displays a dialog box with upgrades and hotfixes available for the selected instance.

3. Select an upgrade/hotfix from the tree on the left.
4. Select the instances you want to apply this upgrade/hotfix to.
5. Click on one of the following buttons:

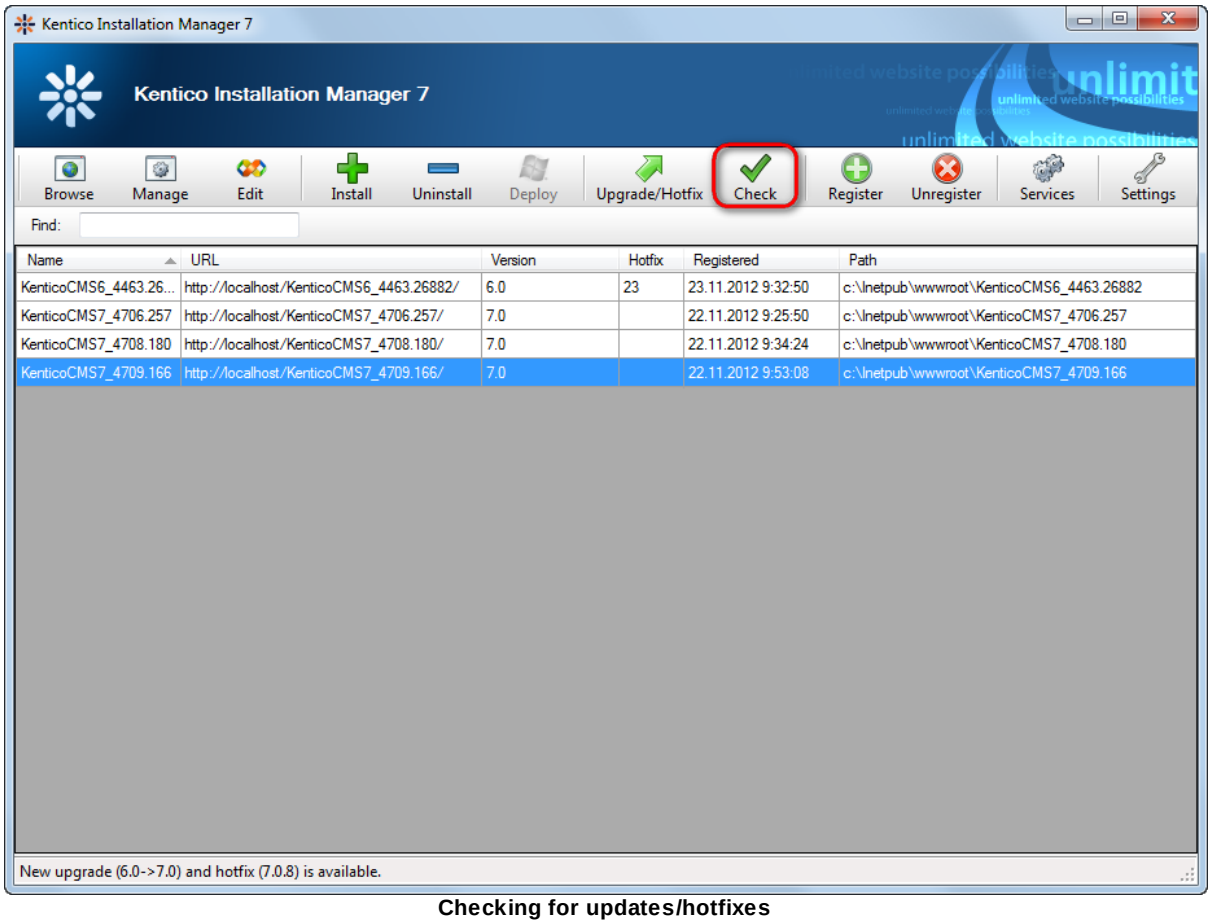
- **Apply** - the tool downloads the upgrade or hotfix and launches the Kentico Upgrade or Hotfix Utility.
- **Download** - the tool only downloads the upgrade or hotfix and stores it in *C:\ProgramData\KIM*. You can apply the upgrade/hotfix later.



Applying a hotfix on an instance

Checking for available upgrades and hotfixes

1. Click ✔ **Check**.



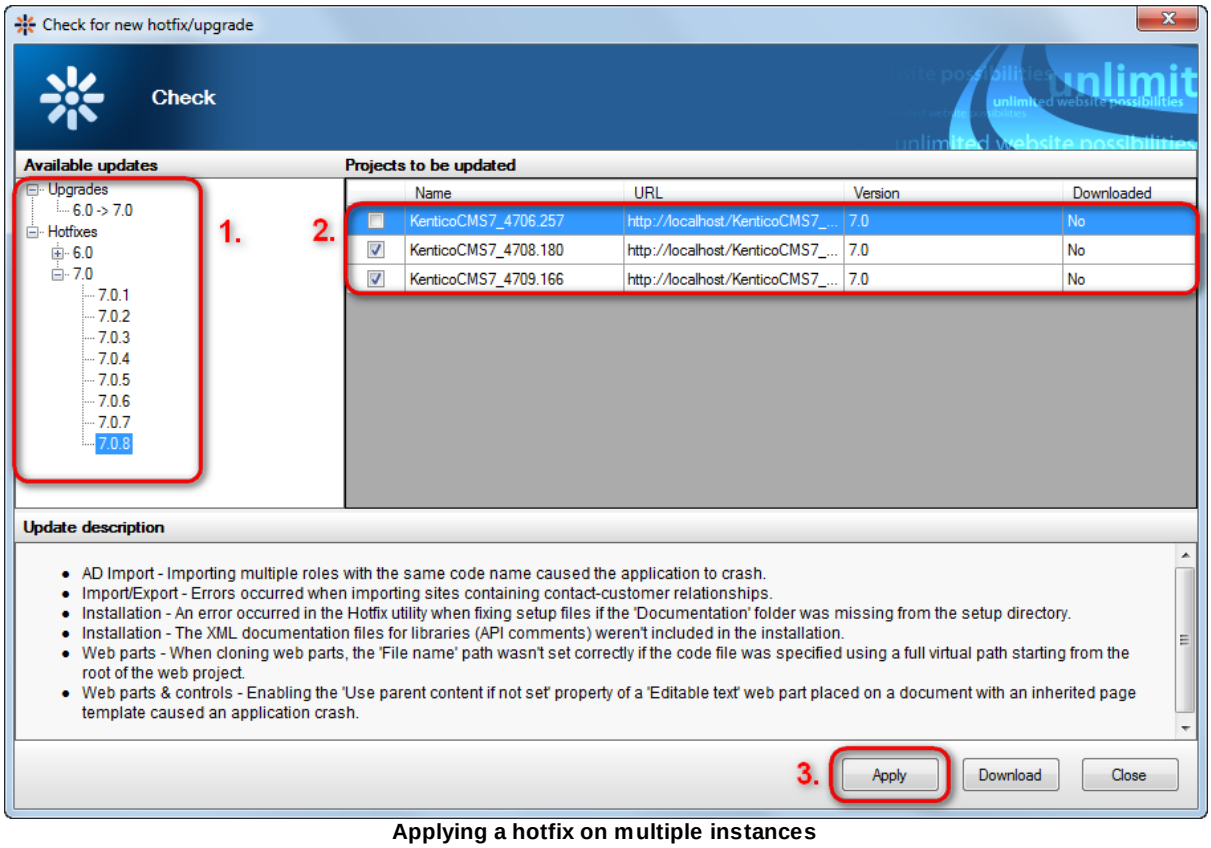
Checking for updates/hotfixes

KIM displays a dialog box with upgrades and hotfixes available for all instances included in the list.

2. Select an upgrade/hotfix from the tree on the left.

3. Select all the instances you want to apply this upgrade/hotfix to.

4. Click on one of the following buttons:
   - **Apply** - the tool downloads the upgrade or hotfix and launches the Kentico Upgrade or Hotfix Utility.
   - **Download** - the tool only downloads the upgrade or hotfix and stores it in *C:\ProgramData\KIM*. You can apply the upgrade/hotfix later.



Applying a hotfix on multiple instances

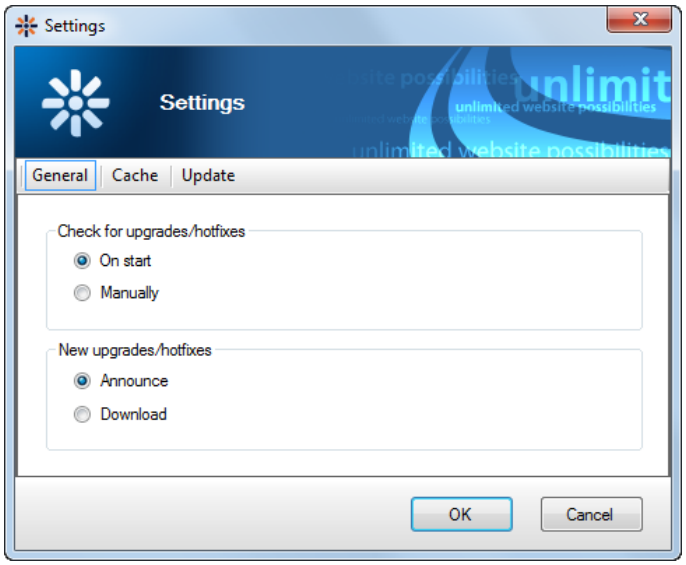
Settings

You can adjust the settings related to upgrading and hotfixing by clicking the **Settings** button on the main toolbar.

On the **General** tab, you can adjust the following settings:

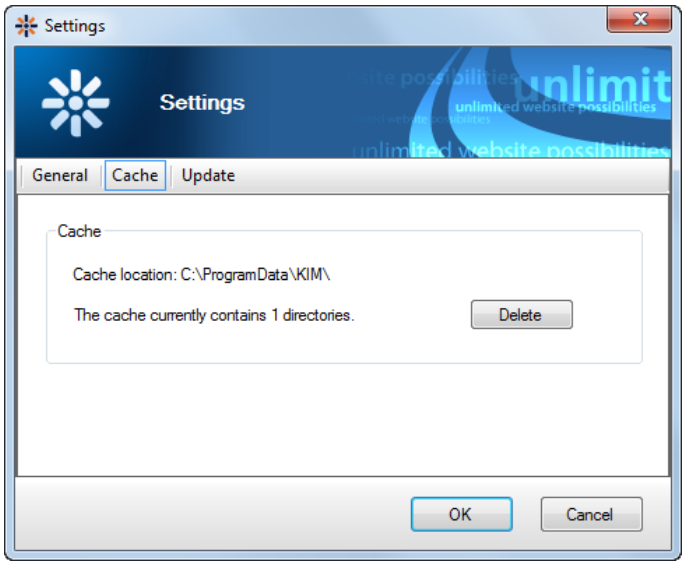
- **Check for upgrades/hotfixes** - determines when KIM checks if new hotfixes or upgrades are available for registered instances.
  - **On start** - KIM checks for updates/hotfixes automatically upon launching.
  - **Manually** - KIM checks only after selecting the **Upgrade/Hotfix** or **Check** action from the toolbar.

- **New upgrades/hotfixes** - determines what to do when new upgrades or hotfixes are found. Only applicable if **On start** is selected above.
  - **Announce** - KIM displays a notification message.
  - **Download** - KIM downloads all new upgrades and hotfixes automatically.



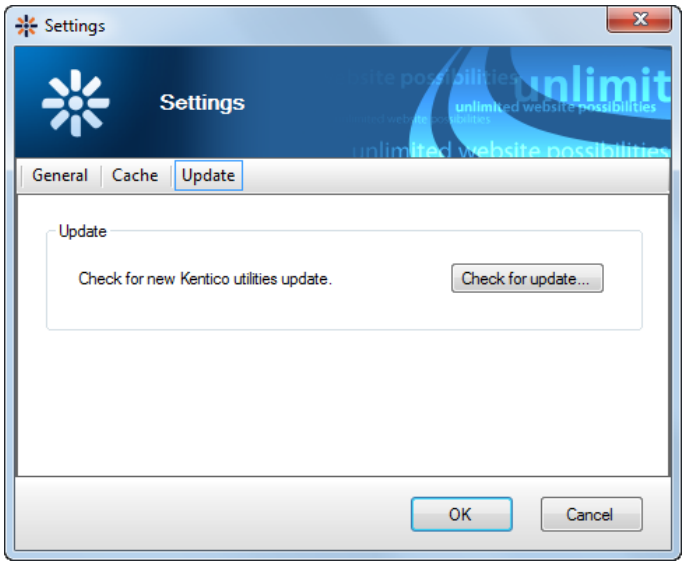
Configuring how should KIM check for updates

On the **Cache** tab, you can see the number of temporary upgrade and hotfix folders. These folders contain upgrade/hotfix data unpacked from downloaded upgrade/hotfix packages and are stored in *C:\ProgramData\KIM*.
You can delete these temporary folders by clicking the **Delete** button.



Deleting the cache files

On the **Update** tab, you can check for available updates of Kentico CMS **program files**.

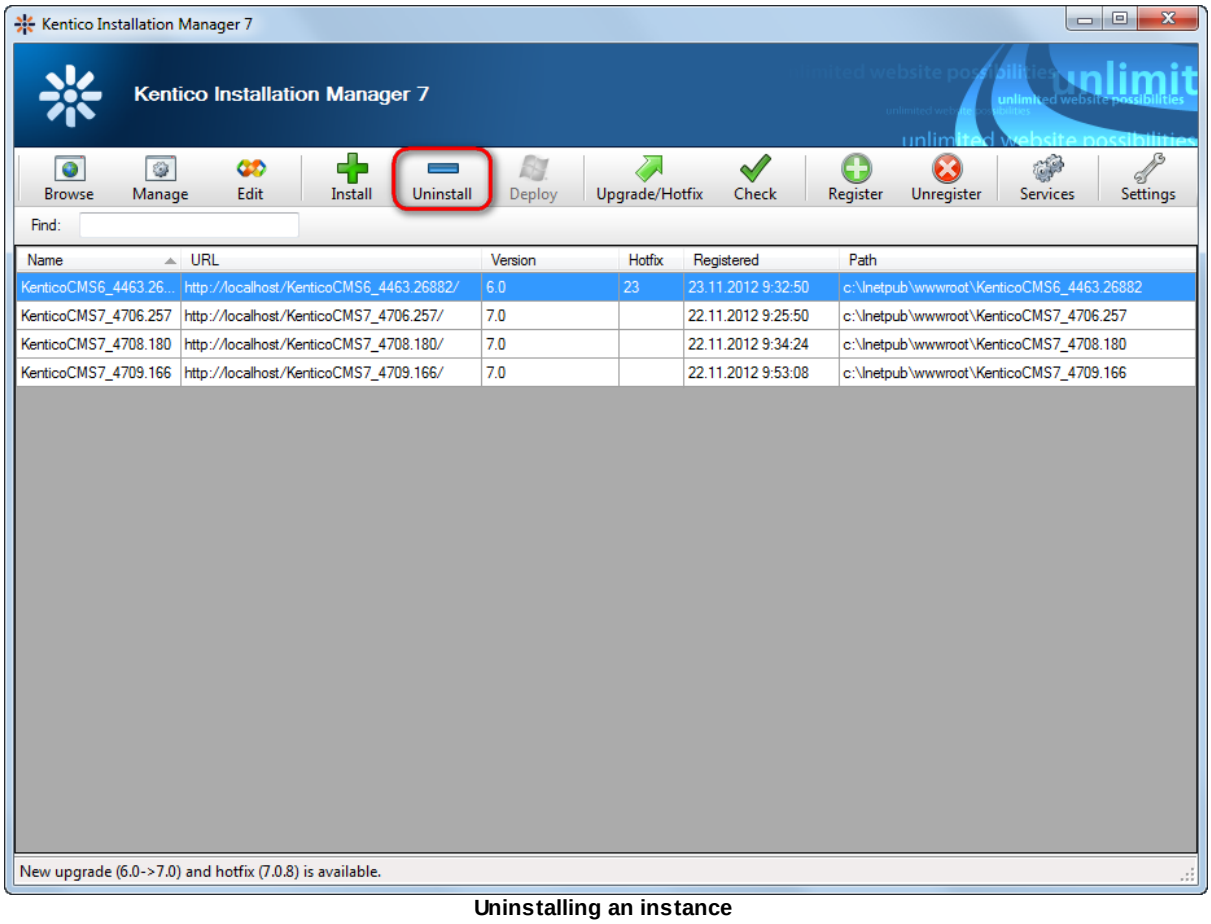


Updating the Kentico CMS program files

9.3.4 Uninstallation

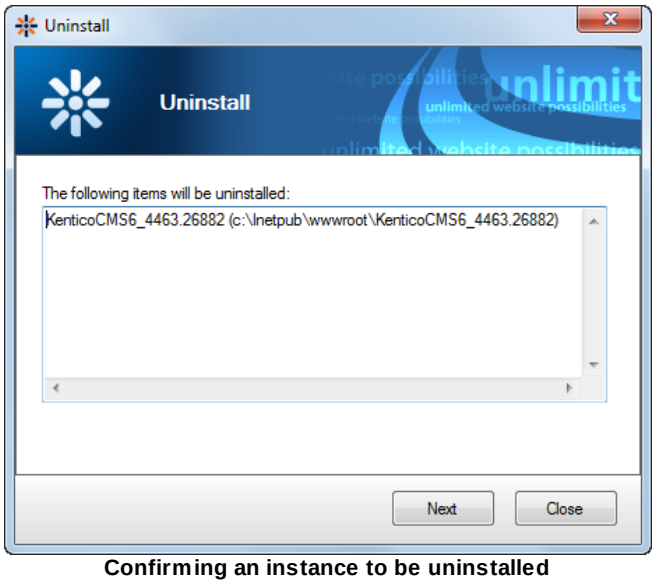
You can also use Kentico Installation Manager to uninstall Kentico CMS instances, including their project folders in the file system, websites in IIS and databases on a database server.

1. Select the instance (or multiple instances) that you want to uninstall and click **Uninstall** on the main toolbar.



Uninstalling an instance

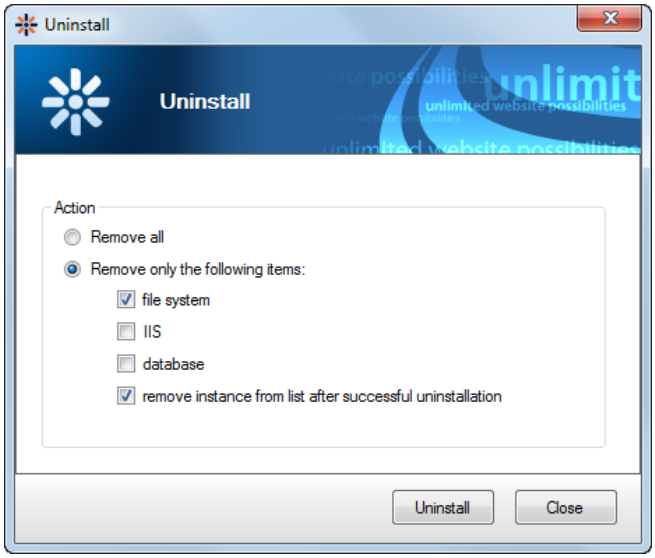
KIM displays a dialog box showing the name and path of the website to be uninstalled.



Confirming an instance to be uninstalled

2. Click **Next**.

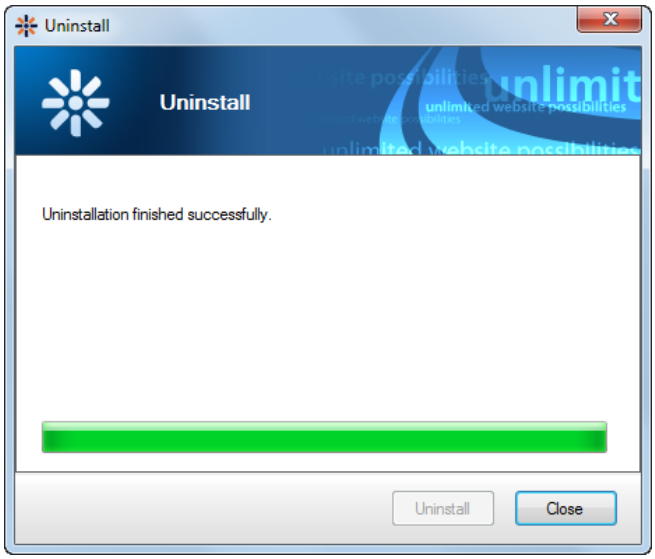
3. Choose which parts of the installation you want to uninstall:
   - **Remove all** - removes the instance completely, including its file system folder, IIS website, database and registration in KIM.
   - **Remove only the following items** - removes only the parts of the installation selected by the check-boxes below.
     - **file system** - deletes the instance's project folder and all its content.
     - **IIS** - removes the instance's website from IIS. If the website is installed in a custom application pool not shared with another website, the pool is deleted as well.
     - **database** - removes the instance's database from the database server.
     - **remove instance from list after successful uninstallation** - unregisters the instance from KIM so that it is no longer visible in the managed instances list.



Choosing the items to uninstall

4. Click **Uninstall**.

KIM displays a log, showing the progress of the uninstallation.



Succesful completion of the uninstallation process

9.3.5 Deployment to Azure

Using the **Deploy** button, you can easily deploy your Windows Azure instance of Kentico CMS on a Windows Azure server.

Before you can use this feature, you have to **create a certificate**, which will provide you with the access to the Windows Azure Service Management API functionality.
The page http://msdn.microsoft.com/en-us/library/windowsazure/gg432987.aspx will help you create a X.509 v3 certificate on your machine and create a personal information exchange certificate, which you have to upload to the Windows Azure via Windows Azure portal.

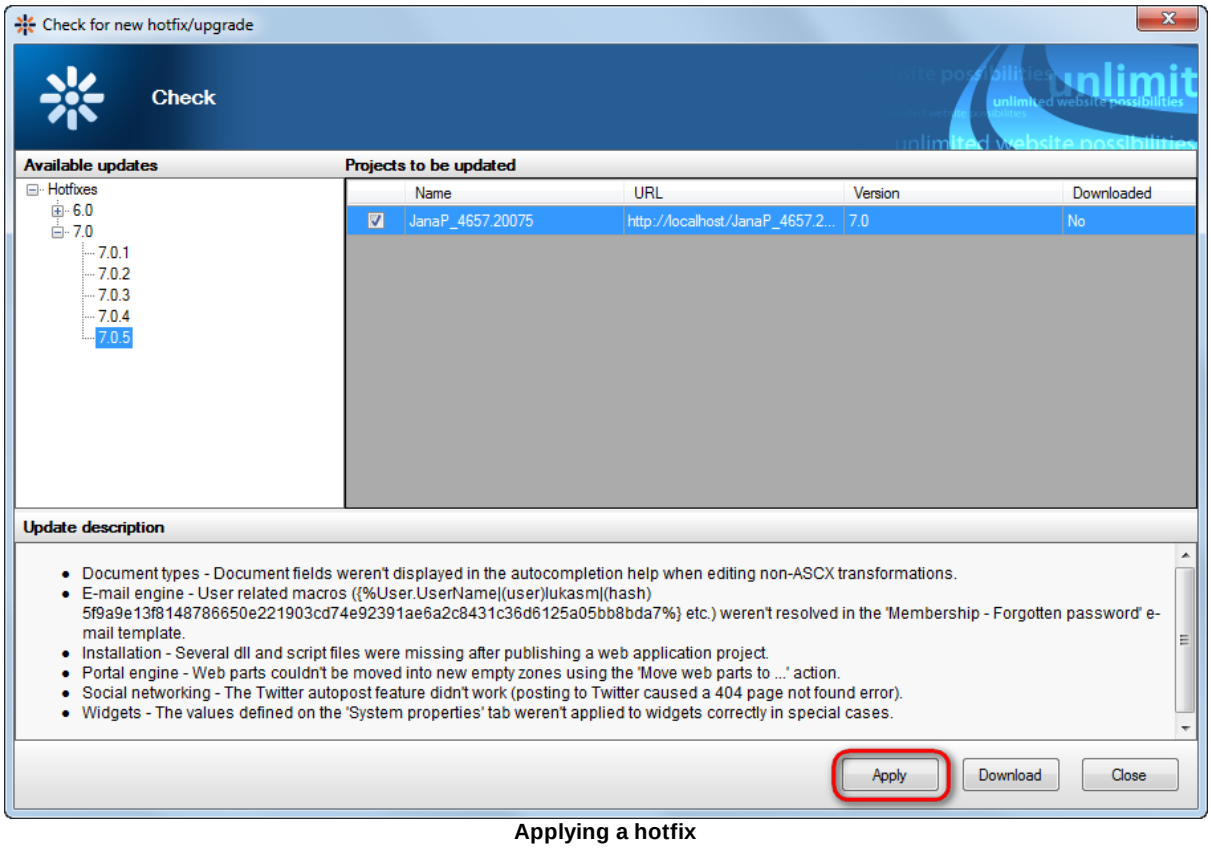
For general instructions on how to configure your Azure project, refer to the Windows Azure Deployment Guide.

9.4 Kentico Hotfix and Upgrade Utility

9.4.1 Overview

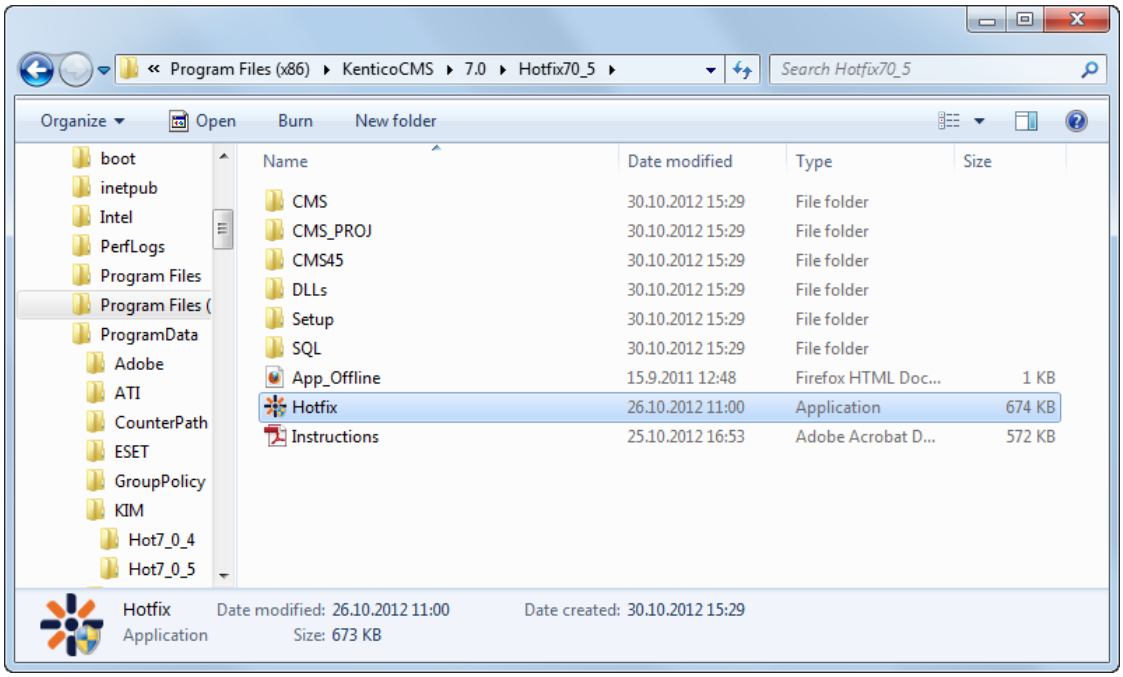
Kentico Hotfix Utility and Kentico Upgrade Utility are two identical external utilities that can be used to hotfix or upgrade Kentico CMS instances. They are included in Kentico CMS hotfix and upgrade packages as the *Hotfix.exe* and *Upgrade.exe* executables. The utilities can be launched in two ways:

- By clicking the **Apply** button in the Upgrade/Hotfix and Check dialogs of Kentico Installation Manager.



Applying a hotfix

- By executing the file unpacked from the hotfix package.



Running a hotfixing utility

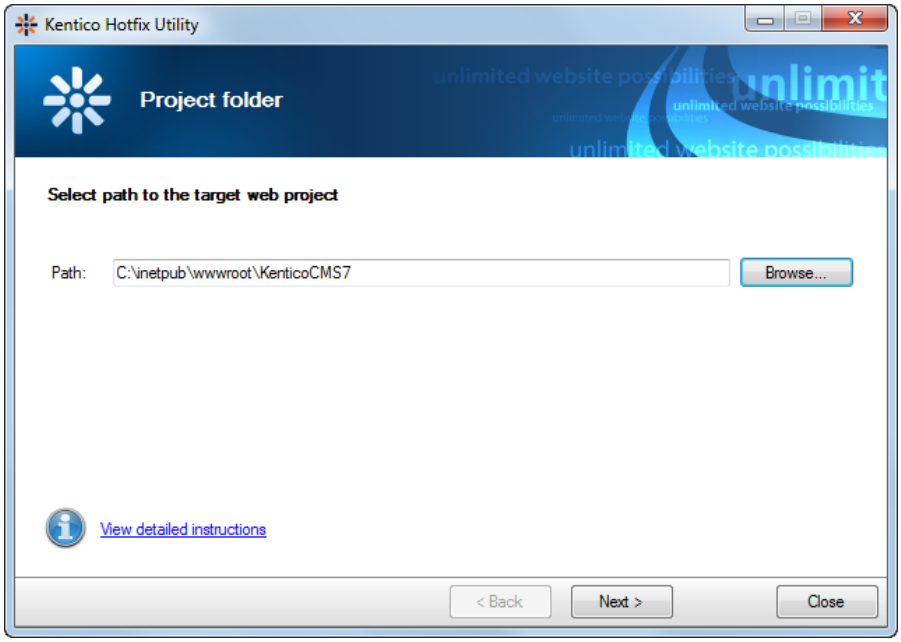
Please refer to the Using the wizard topic for a step-by-step guide through individual steps of the utility's

wizard.

9.4.2 Using the wizard

The hotfix or upgrade process consists of the following steps:

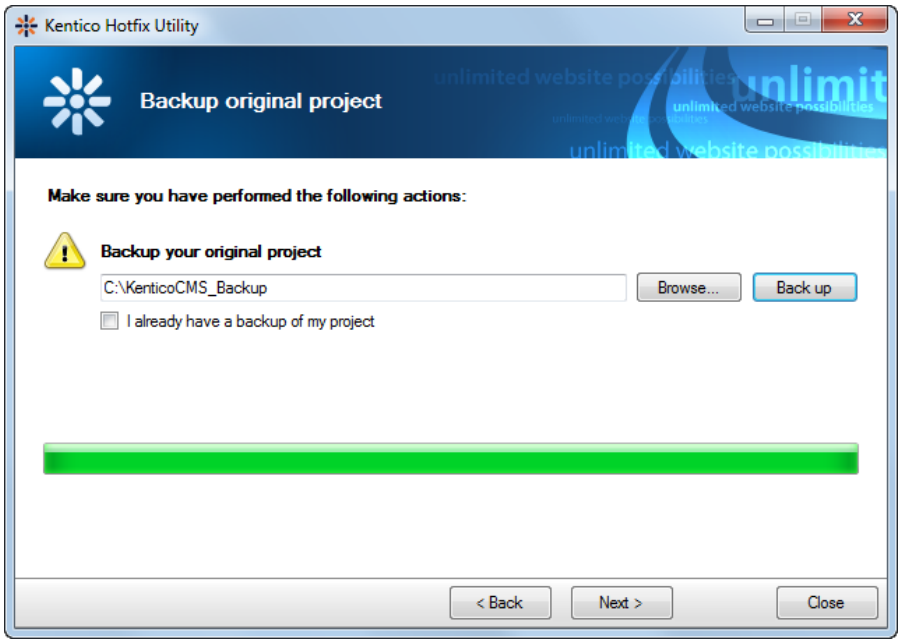
1. Specify the root folder of the Kentico CMS instance you want to upgrade or hotfix.
   - This is only necessary if you ran the utility manually (not from Kentico Installation Manager).



Setting the folder of a project to upgrade/hotfix

2. Click **Next**.

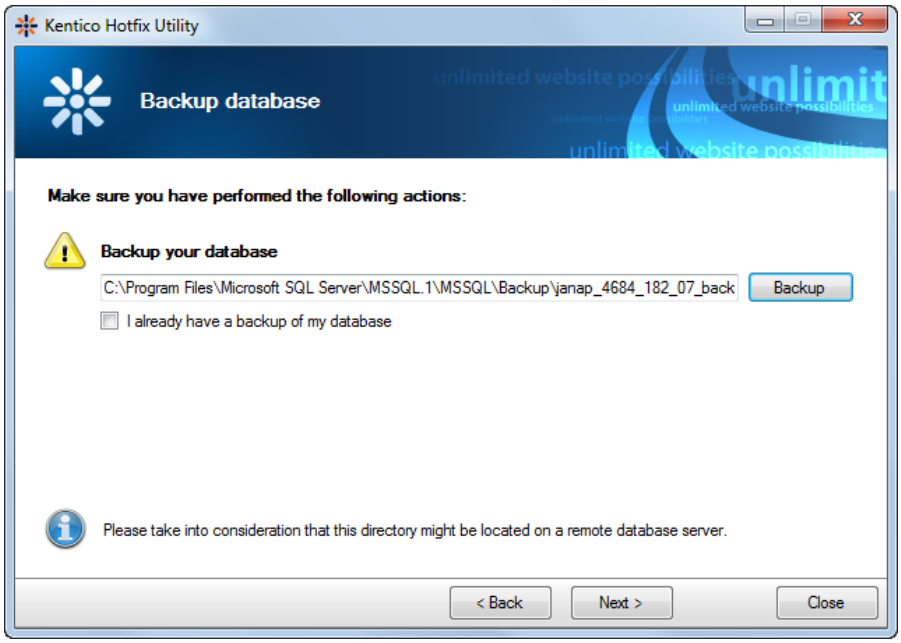
3. Specify a project backup folder and click **Backup** OR check the **I already have a backup of my project** check-box.



Creating a backup of a project

4. Click **Next**.

5. Specify a database backup folder and click **Backup** OR check the **I already have a backup of my database** check-box.

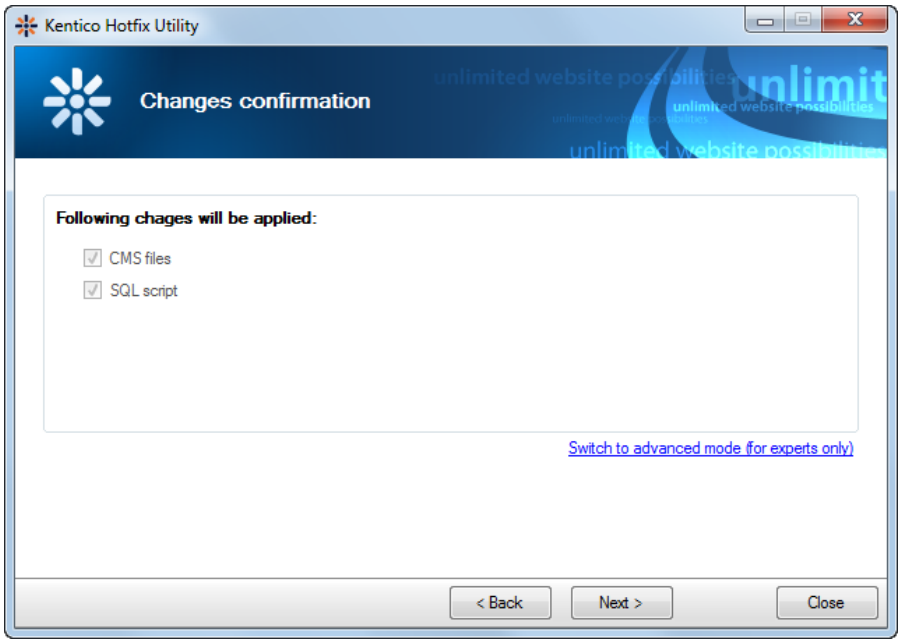


Creating a database backup

6. Click **Next**.

7. Keep the default settings (i.e., all options enabled) OR click the **Switch to advanced mode (for experts only)** link and specify, if the upgrade or hotfix should replace project files, setup files (i.e.,

external utilities) and execute SQL scripts..

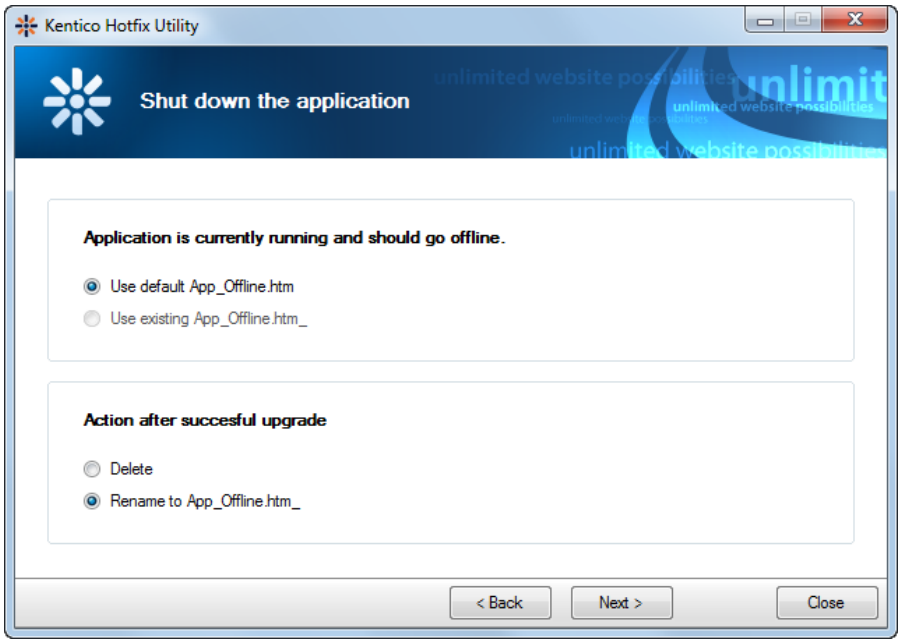


Configuring the upgrade/hotfix settings

8. Click **Next**.

9. Select one of the first set of options:
- **Use default App_Offline.htm** - the utility creates a new *App_Offline.htm* file in the root of the web project to put the instance into an offline mode (see http://msdn.microsoft.com/en-us/library/ff925031.aspx for more details on the use of the file).
- **Use existing App_Offline.htm_** - uses an existing *App_Offline.htm_* file and renames it to *App_Offline.htm*. The file must be present in the project's root folder for this option to be available. See the second option below for more details.

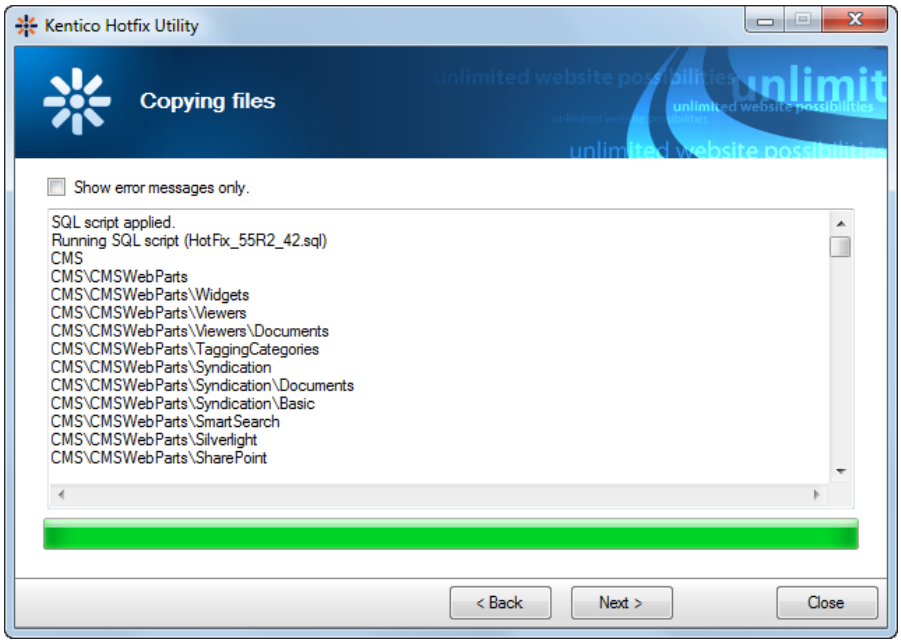
10. Select one of the second set of options, which lets you choose what to do with the *App_Offline.htm* file:
- **Delete** - the file will be deleted.
- **Rename to App_Offline.htm_** - the file will be renamed to *App_Offline.htm_* and usable on next upgrade or hotfix.



Shutting down the application

11. Click **Next**.

A log showing progress of the hotfix or upgrade is be displayed. You can enable the **Show error messages only** check-box to display only error messages in the log.

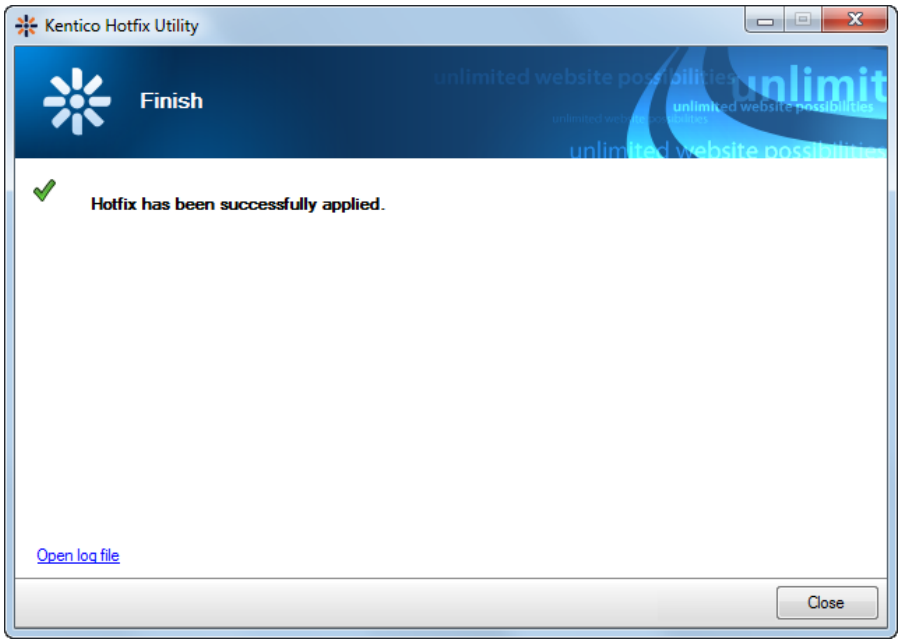


Upgrading/hotfixing process

12. Click **Next**, once the process finishes.

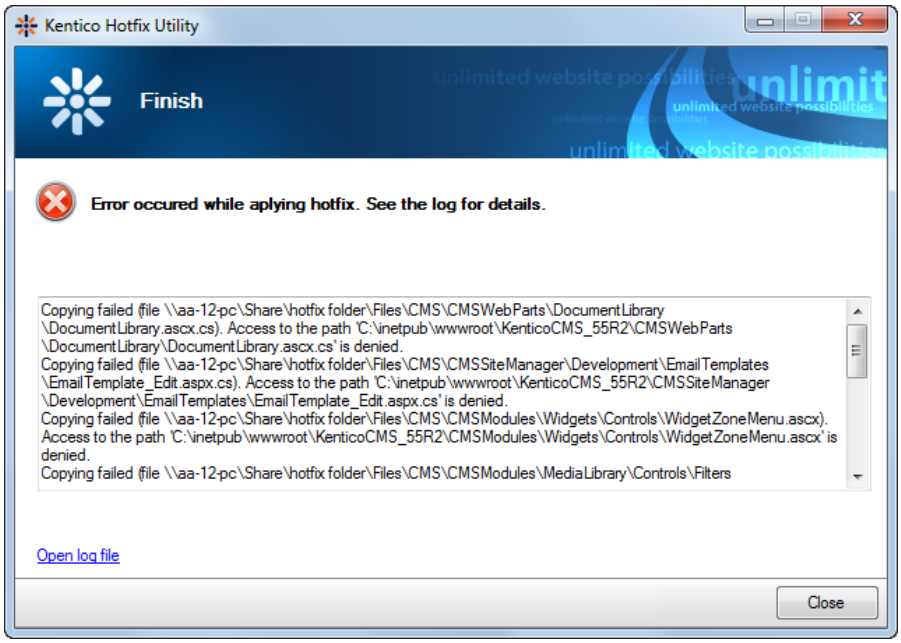
The utility lets you know if the hotfix or upgrade finished successfully. By clicking the **Open log file** link, you can open the upgrade process log that was displayed in the previous step.

Physically, the log is saved in *~\App_Data\log.txt*, so you can inspect it even after closing the utility.



Succesful hotfix application

If an error occurred while applying the hotfix or upgrade, the utility will notify you with a message and an error log.



Failed hotfix application

13. Click **Close** to finish and close the utility.

9.5 Kentico Service Manager

9.5.1 Overview

Kentico CMS Service Manager is an external Windows utility which allows management of external Windows services used by Kentico CMS. Using the utility, you can install, uninstall, start, stop and restart Windows services for individual instances of Kentico CMS. While installation and uninstallation can be performed from Windows command line and the services can be started, stopped and restarted from Windows Services manager (*services.msc*), this utility integrates these tasks into a single tool where Kentico CMS services can be managed separately from other services running in Windows.

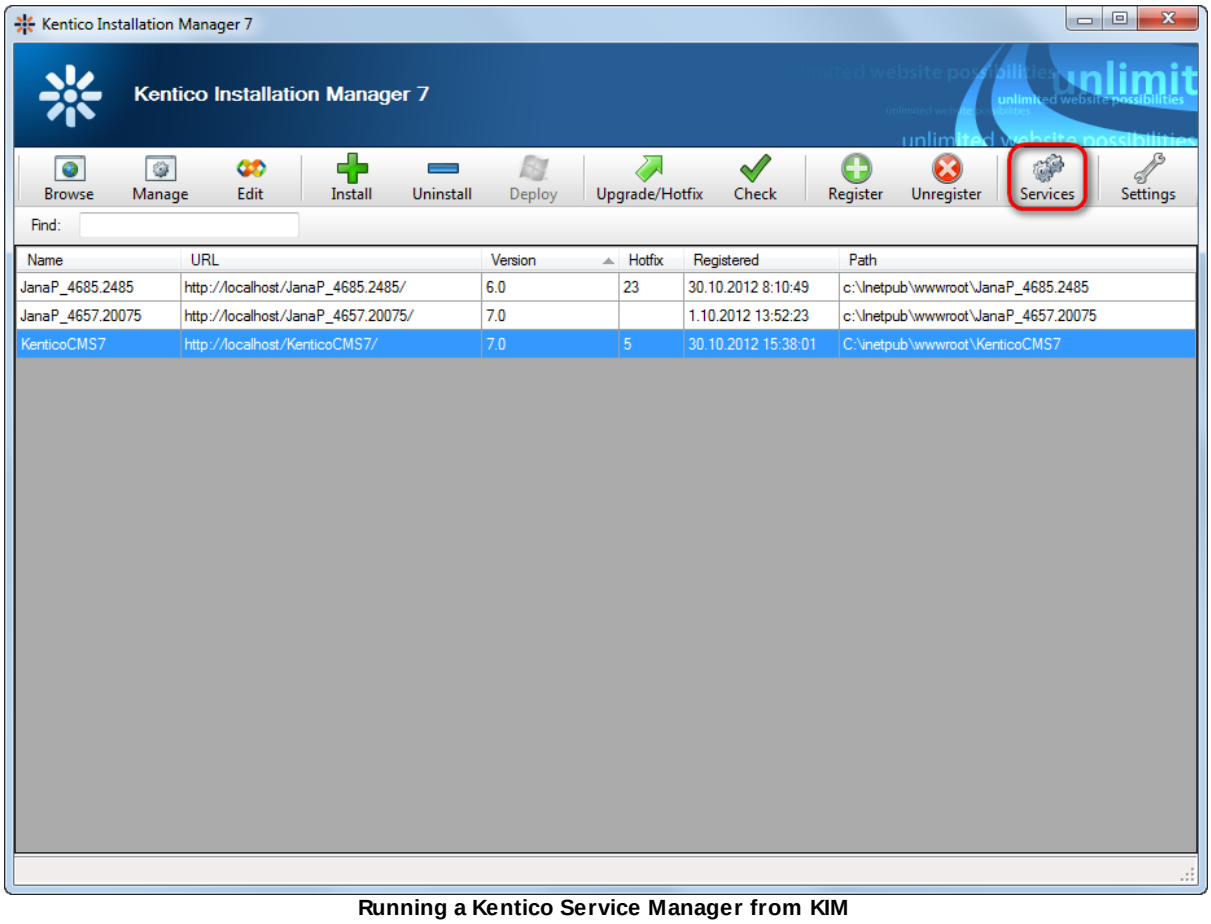
In the Using the utility topic, you can learn how services can be managed using the utility. Windows services for each particular Kentico CMS instance are defined in a dedicated XML file. Only services that are defined in the file can be managed by the Service Manager. To learn more about this file and its required format, please refer to the Services definition XML topic.

Launching the utility

Kentico Service Manager can be launched two ways.

A) First way:

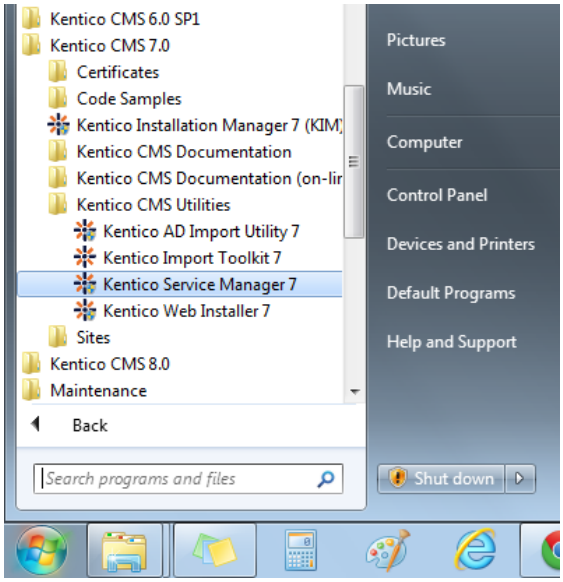
1. Run Kentico Installation Manager.
2. Select an instance of Kentico CMS.
3. Click the **Services** button on the toolbar.



Running a Kentico Service Manager from KIM

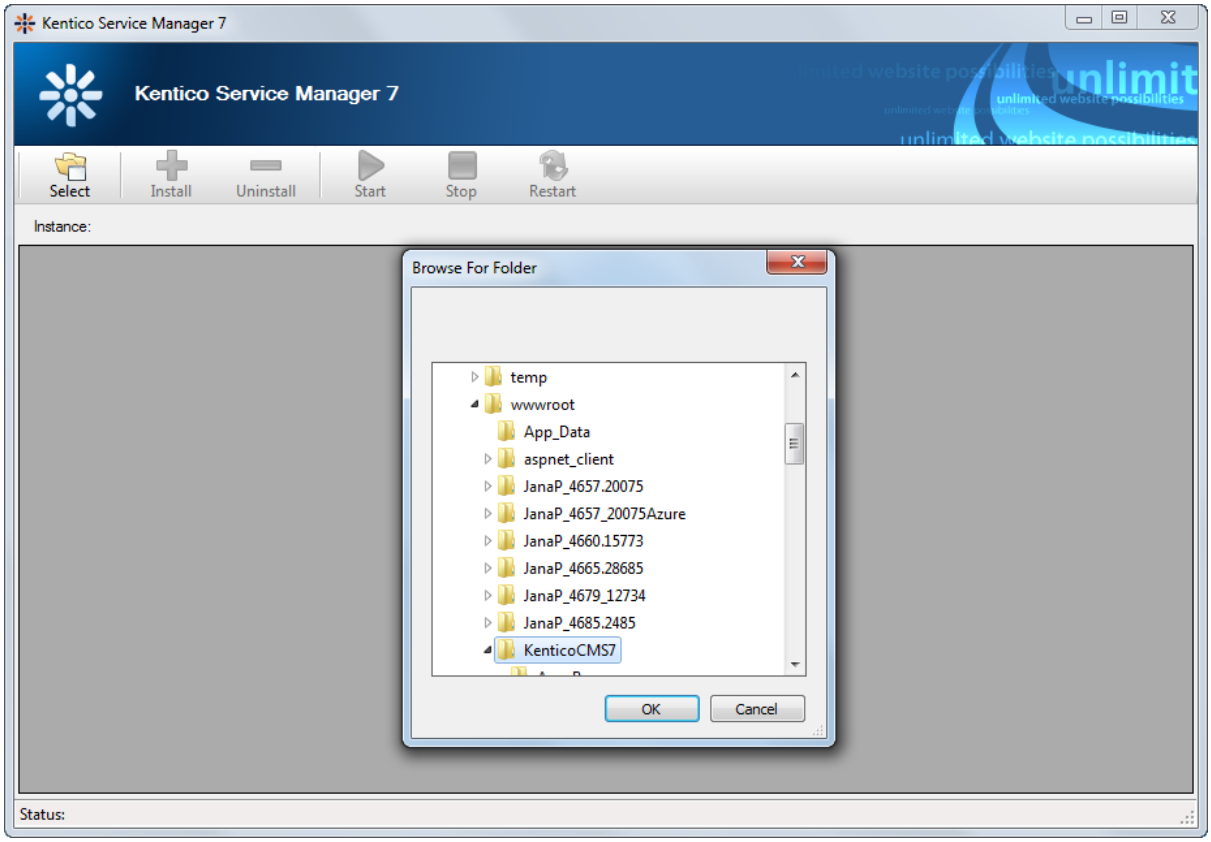
B) Second way:

1. Launch the **Kentico Service Manager** from Windows Start menu.



Running KSM 7.0 in Windows

2. Select the root of the Kentico CMS instance, whose services you want to manage, in the **Browse For Folder** dialog box.



Selecting a root folder of an insance

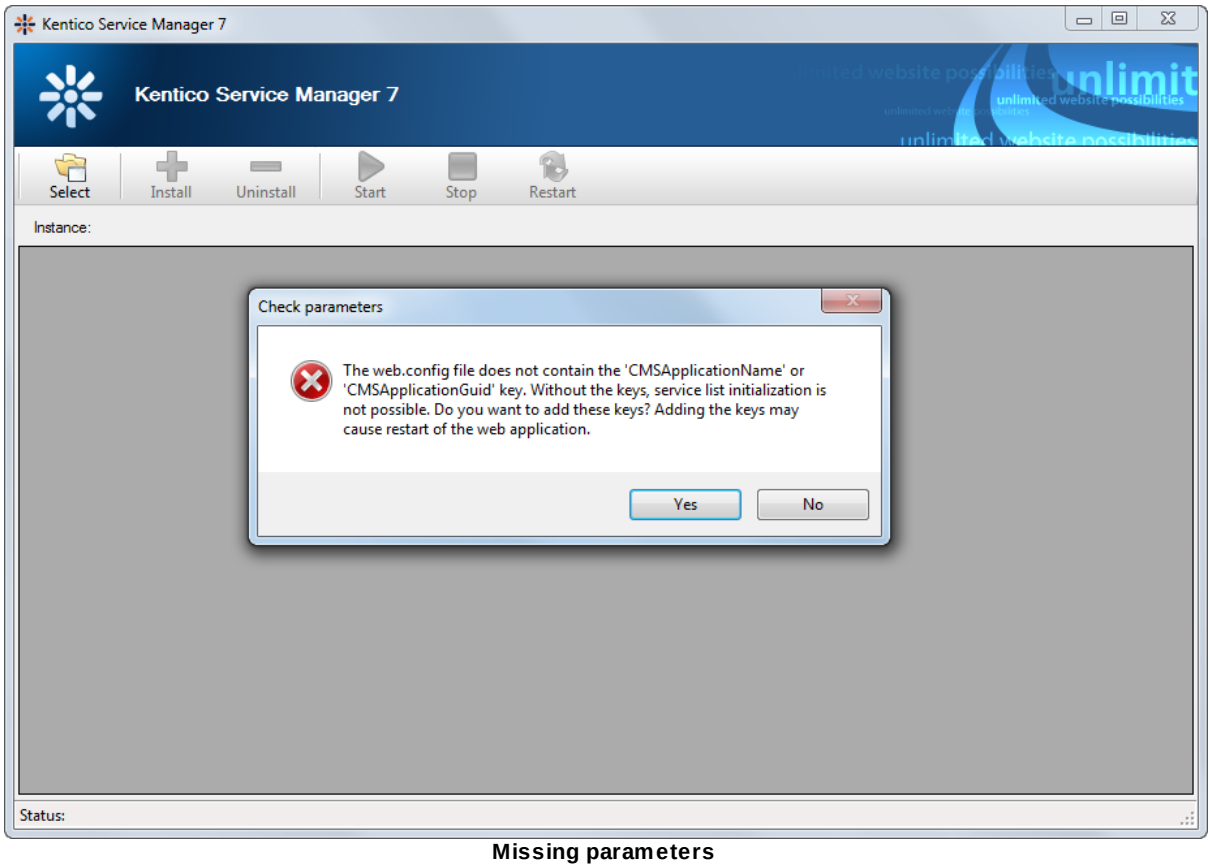
3. Click **OK**.

The Kentico Service Manager launches. For the description of its functions, continue to the Using the utility topic.

Problems with required web.config keys

Kentico CMS Windows services require the **CMSApplicationName** or **CMSApplicationGuid** keys to be added in the *appSettings* section of the *web.config* file. Values of these keys are used in names of the Windows services and for identification of the Kentico CMS instance by the services. These keys are included in the *web.config* by default. However, when neither of the two keys is in the *web.config* of the selected instance (e.g., after being removed manually), you will encounter the dialog depicted in the screenshot below.

By clicking **Yes**, you tell the Service Manager to add these keys to the *web.config* so that services can work with the selected Kentico CMS instance. By clicking **No**, the keys will not be added, resulting in no services to be manageable for the selected instance. In the second case, another instance can be selected after clicking the **Select** ( ) button in the top toolbar.



Missing parameters

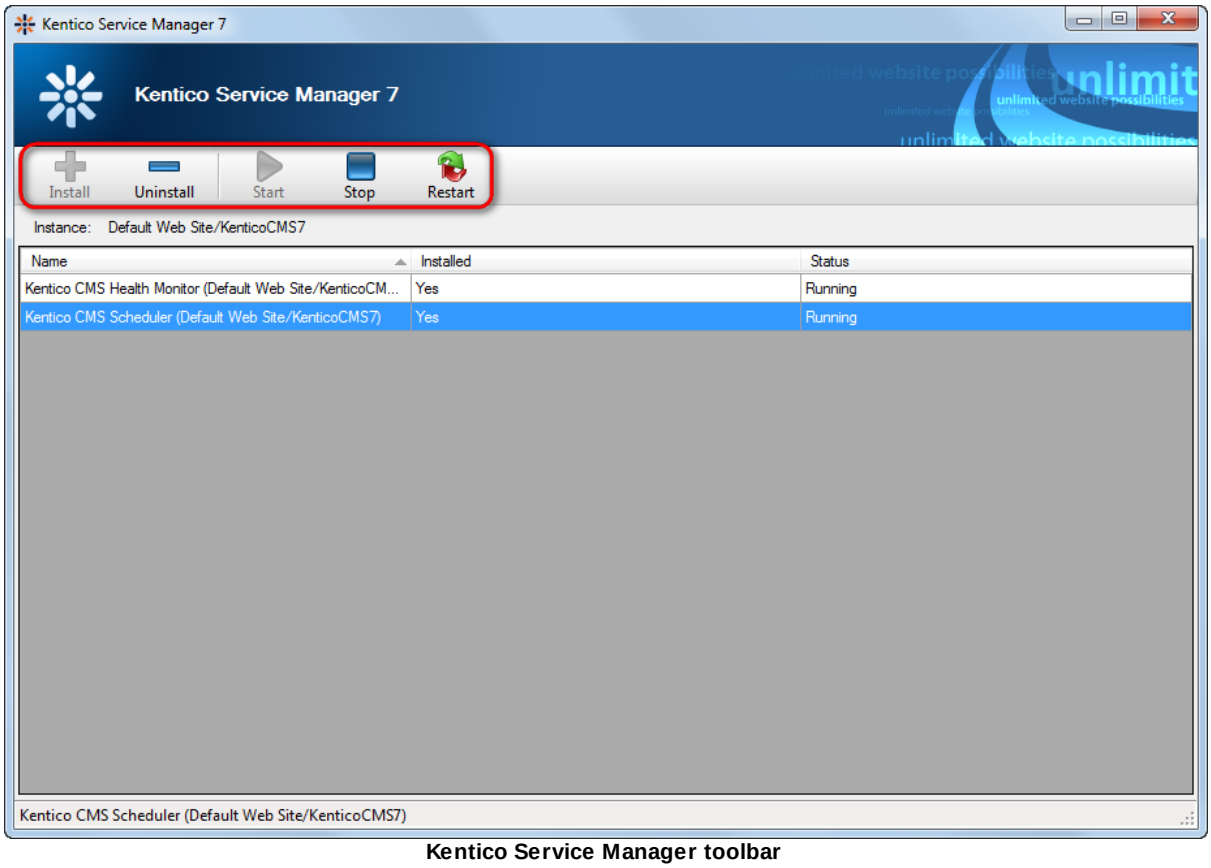
9.5.2 Using the utility

After launching the utility (and successful selection of the managed Kentico CMS instance when launched from Start menu), available services are listed in the main area. The following information is listed for each services:

- **Name** - name of the Windows service in format **<service name> (<CMSApplicationName web.config key value>)**.
- **Installed** - shows if the service is installed or not.
- **Status** - shows if the service is running or stopped, or intermediate statuses when the service is being started, stopped or restarted.

If you select a service in the listing, the following management actions are available in the top toolbar. The same actions are also available in a context menu accessible by right-clicking a service.

- **Install** - installs the selected service. Only available if the service is not installed.
- **Uninstall** - uninstalls the selected service. Only available if the service is installed.
- **Start** - starts the selected service. Only available if the service is installed and not running.
- **Stop** - stops the selected service. Only available if the service is installed and running.
- **Restart** - restarts the selected service. Only available if the service is installed and running.



Kentico Service Manager toolbar

9.5.3 Services definition XML

Windows services that can be managed by Kentico Service Manager are defined in *~/App_Data/CMSModules/WinServices/services.xml*. All health monitoring services that should be manageable by Kentico Service Manager and generally all Windows services that should work with the respective Kentico CMS instance (e.g. your custom ones) need to be defined in this file.

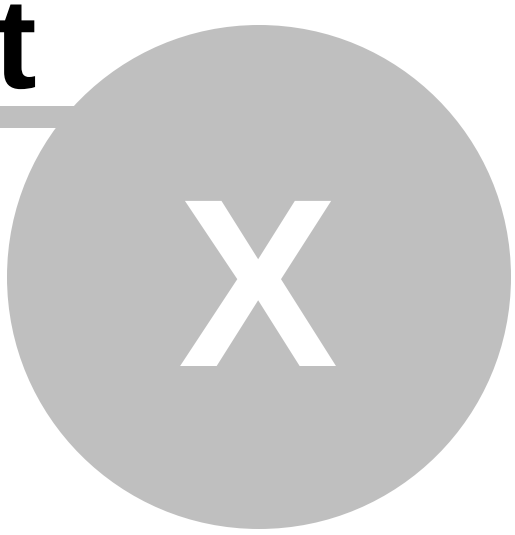
Below, you can see the default content of the *services.xml* file with the Health monitoring and Scheduler services defined. As you can see, the XML has a root **<Services>** element containing any number of **<Service>** elements, each of which represents a single Windows service. Each **<Service>** element contains the following sub-elements:

- **<basename>** - identifying name of the service, the same for all instances of Kentico CMS.
- **<name>** - the same as *<basename>*, with the {0} part appended. The {0} part gets replaced with the value of the *CMSApplicationName* or alternatively *CMSApplicationGuid web.config* keys to differentiate services of particular Kentico CMS instances.
- **<displayname>** - name of the Windows service in friendly format, displayed in user interfaces of Windows Services manager (*services.msc*) and Kentico CMS Service Manager.
- **<description>** - text describing the functionality provided by the service.
- **<assemblyname>** - name of the service's executable file stored inside the *bin* folder in the application root.

```
<?xml version="1.0"?>
```

```
<Services>
  <Service>
    <basename>KenticoCMSHealthMonitor</basename>
    <name>KenticoCMSHealthMonitor{0}</name>
    <displayname>Kentico CMS Health Monitor ({0})</displayname>
    <description>Registers categories with counters where information will be
subsequently stored. Then it gets data from the database (e.g. the number of e-
mails in the e-mail queue, the number of tasks in the queue, etc.) and stores it
in the respective counters.</description>
    <assemblyname>HealthMonitoringService.exe</assemblyname>
  </Service>
  <Service>
    <basename>KenticoCMSScheduler</basename>
    <name>KenticoCMSScheduler{0}</name>
    <displayname>Kentico CMS Scheduler ({0})</displayname>
    <description>Processes Kentico CMS scheduled tasks that have the 'Allow run in
external service' option enabled. Processing is performed in regular intervals and
only tasks with the mentioned option enabled are processed by the service.</
description>
    <assemblyname>SchedulerService.exe</assemblyname>
  </Service>
</Services>
```

Part



API programming and Kentico CMS internals

10 API programming and Kentico CMS internals

10.1 Overview

Kentico CMS allows you to script any action that you would normally make through the user interface, such as:

- content management (select/create/update/delete documents)
- workflow management
- security (create users, roles, set permissions, etc.)
- and many others

This allows you to create customized features and add them to the user interface or write procedures for integration with external systems.

10.2 CMSContext class

CMS.CMSHelper.CMSContext class provides useful methods that allow you to access information about the current page, user, etc. All methods are static, which means you can access them directly like `CMS.CMSHelper.CMSContext.CurrentAliasPath`, without instantiating the CMSContext class.

CurrentAliasPath	Alias path of the currently required document.
CurrentDocument	Currently processed document. This object contains all document data including the product data. Please note that retrieving this property is time consuming because it may need to access the database to get the data. You can use the property CurrentPageInfo to get basic document data without additional operations required. It is cached when Content caching is enabled.
CurrentPageInfo	Currently processed page info. This property provides only basic document fields, such as *NodeID*, *NodeName*, *AliasPath*, *ClassName*, etc. It is cached if Page info caching is enabled.
CurrentSite	Provides information on the current site.
CurrentSiteName	Provides code name of the current site.
CurrentUser	Provides information on the current user and their preferences. Beside the standard UserInfo properties, it also provides the following ones: **PreferredCultureCode** – gets/sets user's preferred content culture. You can use this property to change the culture of the displayed content. **PreferredUICultureCode** – gets/sets user's preferred UI culture. It's used mainly by Kentico CMS webparts, controls and in the administration interface. **IsAuthorizedPerResource** – returns true if the current user is authorized with given permission for given module (resource). **IsAuthorizedPerClassName** – returns true if the current user is authorized with given permission for the given document type (class name). It checks the global document permissions.

	IsAuthorizedPerTreeNode – returns true if the current user is authorized with given permission for given document (node). **IsPublic** – returns true if the current user is the Public user account used for anonymous visitors.
ViewMode	The view mode in which the documents are displayed (edit, design, form, live site, preview).
ResolveCurrentPath	Returns current path or its part according to the provided formatting string. You can use it to get only the first N levels of the path – e.g.: */{0}/{1}* applied on *"/products/computers/ibm"* results in: *"/products/computers"*

10.3 TreeHelper class

CMS.CMSHelper.TreeHelper class provides useful methods that allow you to access documents. All methods are static, which means you can access them directly like `CMS.CMSHelper.TreeHelper.SelectNodes`.

SelectSingleNode	Returns single node (document) specified by given NodeID or path.
SelectSingleDocument	Return single node (document) specified by given DocumentID (culture-specific document version).
SelectNodes	Returns a DataSet object containing nodes (documents) of given path.

10.4 Registering custom classes in App_Code

By preparing custom classes that inherit from an appropriate base class, you can extend the functionality of Kentico CMS. This approach allows you to implement the following types of objects:

- integration connectors
- marketing automation actions
- notification gateways
- scheduled tasks
- smart search custom indexes and analyzers
- translation services
- workflow actions

If you create the custom classes in your project's **App_Code** folder (or **Old_App_Code** on *web application* installations), you do not need to integrate a new assembly into the web project. Code in the App_Code folder is compiled dynamically and automatically referenced in all other parts of the application.

Registering custom classes in the App_Code folder

To ensure that the system can load custom classes placed in App_Code, you need to write code that

registers each class.

1. Extend the **CMSModuleLoader** partial class. You can either extend an existing definition of the class, or create a new source file with the following code.

[C#]

```
using System;

using CMS.SettingsProvider;

public partial class CMSModuleLoader
{

}
```

2. Create a new class inside the **CMSModuleLoader** that inherits from *CMSLoaderAttribute*.

[C#]

```
private class ClassLoader : CMSLoaderAttribute
{

}
```

3. Add the attribute defined by the internal class before the definition of the **CMSModuleLoader** partial class.

[C#]

```
[ClassLoader]
public partial class CMSModuleLoader
{
...
```

4. Override the **Init** method inside the attribute class, and assign a handler to the *ClassHelper.OnGetCustomClass* event.

[C#]

```
public override void Init()
{
    ClassHelper.OnGetCustomClass += GetCustomClassEventHandler;
}
```

5. Define the handler method for the *OnGetCustomClass* event as follows:

[C#]

```
private static void GetCustomClassEventHandler(object sender, ClassEventArgs e)
{
    if (e.Object == null)
    {
        switch (e.ClassName)
        {
            case "CustomAction":
                e.Object = new CustomAction();
                break;
        }
    }
}
```

To load your App_Code class into the system, create a new instance and assign it to the **Object** property of the event handler's *ClassEventArgs* parameter (**e**). The *switch* structure allows you to use the handler to load any number of custom classes based on the specified class name.

Overall code example:

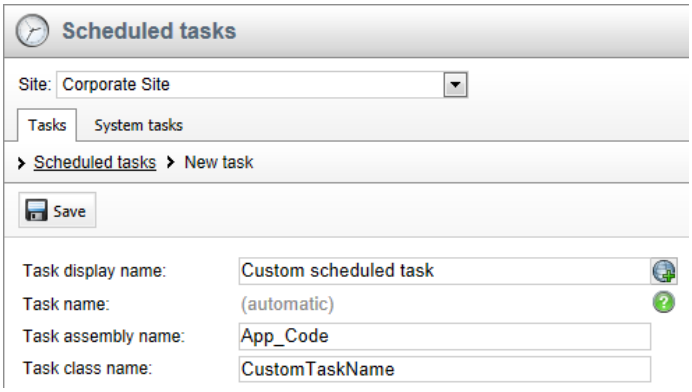
[C#]

```
using System;

using CMS.SettingsProvider;

[ClassLoader]
public partial class CMSModuleLoader
{
    /// <summary>
    /// Attribute class that ensures the loading of custom classes.
    /// </summary>
    private class ClassLoader : CMSLoaderAttribute
    {
        /// <summary>
        /// Called automatically when the application starts.
        /// </summary>
        public override void Init()
        {
            // Assigns a handler for the OnGetCustomClass event
            // This event is triggered when the system requests a class with
App_Code as the assembly name
            ClassHelper.OnGetCustomClass += GetCustomClassEventHandler;
        }

        /// <summary>
        /// Gets the custom class object based on the specified class name.
        /// </summary>
        private static void GetCustomClassEventHandler(object sender,
ClassEventArgs e)
        {
```

```
            // Checks whether an object is already assigned to the requested class
            if (e.Object == null)
            {
                // Checks the name of the requested class
                switch (e.ClassName)
                {
                    // Loads an instance of the 'CustomTask' class for the
'CustomTaskName' name
                    case "CustomTaskName":
                        e.Object = new CustomTask();
                        break;

                    // Loads an instance of the 'CustomIndex' class for the
'CustomIndexName' name
                    case "CustomIndexName":
                        e.Object = new CustomIndex();
                        break;
                }
            }
        }
    }
}
```

Once you have registered your custom classes, you can use them as the source for objects in the system. When assigning App_Code classes to objects in the editing interface, fill in the following values:

- **Assembly name**: App_Code
- **Class name**: must match the name that identifies the corresponding custom class in the switch statement of your *OnGetCustomClass* handler method. When the *OnGetCustomClass* event is triggered, the value of the *Class name* field is passed on through the *ClassName* property of the *ClassEventArgs* parameter.



Assigning a custom App_Code class to a new scheduled task

10.5 Site management, import and export

10.5.1 Creating a new website

The following example shows how you can create a new site based on the Blank website template:

[C#]

```
using CMS.SettingsProvider;
using CMS.CMSHelper;
using CMS.GlobalHelper;
using CMS.SiteProvider;
using CMS.CMSImportExport;

...

    // Site name
    string siteName = "testAPIsite";

    try
    {
        // Create site import settings
        SiteImportSettings settings = new SiteImportSettings();

        //Initialize the settings
        settings.SiteName = siteName;
        settings.SiteDisplayName = "Test API site";
        settings.SiteDescription = "Site for testing the API examples";
        settings.SiteDomain = "127.0.0.254";

        // Get 'Blank site' web template
        WebTemplateInfo template = WebTemplateInfoProvider.GetWebTemplateInfo
("BlankSite");

        // Template exists
        if (template != null)
        {
            // Set source file path
            string templatePath = template.WebTemplateFileName;
            settings.SourceFilePath = Server.MapPath(templatePath.TrimEnd('\\') +
"\\");

            // Load default selection to preselect the objects
            settings.LoadDefaultSelection();

            // Create new site using 'Blank' template
            ImportProvider.ImportObjectsData(settings);

            // Run site
            SiteInfoProvider.RunSite(siteName);

            this.lblInfo.Text = string.Format("New site with code name '{0}' has
been created.", siteName);
            return;
        }
        else
        {
            this.lblInfo.Text = string.Format("Failed to create new site '{0}'.<br
/>Web template 'Blank site' doesn't exist.", siteName);
            this.lblInfo.CssClass = "ErrorLabel";
        }
    }
    catch (RunningSiteException)
    {
```

```
            this.lblInfo.Text = string.Format("Failed to run site '{0}'.", siteName);
            this.lblInfo.CssClass = "ErrorLabel";
        }
        catch (Exception ex)
        {
            this.lblInfo.Text = string.Format("Failed to create new site '{0}'.<br />
>Original exception: " + ex.Message, siteName);
            this.lblInfo.CssClass = "ErrorLabel";
        }
```

10.5.2 Import and export of a website

You can export and import a website using Kentico CMS API, using the **CMS.CMSImportExport.ImportProvider/ExportProvider** classes. You can use the following methods:

- `public static void ImportSite(string siteName, string siteDisplayName, string siteDomain, string fullSourcePath, string websitePath, UserInfo currentUser)`
  - **siteName** - code name of the newly created website.
  - **siteDisplayName** - display name of the newly created website.
  - **siteDomain** - domain name of the newly created website.
  - **fullSourcePath** - physical disk path of the package containing the site that should be imported.
  - **websitePath** - physical disk path of the website root.
  - **currentUser** - the UserInfo object of the user under which the import should be performed.

- `public static void ExportSite(string siteName, string fullExportFilePath, string websitePath, bool template, UserInfo userInfo)`
  - **siteName** - code name of the site to be exported.
  - **fullExportFilePath** - physical disk path of the export package.
  - **websitePath** - physical disk path of the website root.
  - **template** - if false, a .zip file containing the site will be created under *fullExportFilePath* (should be the full path including the .zip file name). Otherwise, the whole site will be exported as a website template to the same location (in this case, the full path should be used without any file name specified).
  - **userInfo** - the UserInfo object of the user under which the export should be performed.

10.5.3 Updating website properties

The following example shows you how to modify website properties using the API:

[C#]

```
using CMS.CMSHelper;
using CMS.SiteProvider;
using CMS.WorkflowEngine;

...

    // Get current site name (you can use code name of the required site instead)
    string siteName = CMSContext.CurrentSiteName;

    // Check if the site name is available
```

```
    if (siteName != null)
    {
        try
        {
            // Stop the site because we will change the domain of the site
            SiteInfoProvider.StopSite(siteName);

            // Get site info
            SiteInfo si = SiteInfoProvider.GetSiteInfo(siteName);

            if (si != null)
            {
                si.DisplayName = "New display name";
                si.Description = "New description";

                // Change domain of the site
                si.DomainName = "mynewdomain.com";

                // Save the changes
                SiteInfoProvider.SetSiteInfo(si);
            }

            // Run the site
            SiteInfoProvider.RunSite(siteName);

            lblInfo.Text = string.Format("Site '{0}' has been edited.", siteName);
        }
        catch (RunningSiteException ex)
        {
            lblError.Text = "Site cannot be started.<br />Original exception: " +
ex.Message;
        }
        catch (Exception ex)
        {
            lblError.Text = "Error when modifying site properties.<br />Original
exception: " + ex.Message;
        }
    }
```

The following sample code shows how you can add a new content culture to an existing website:

[C#]

```
using CMS.SiteProvider;

...

        // Get site and culture objects
        SiteInfo si = SiteInfoProvider.GetSiteInfo("CorporateSite");
        CultureInfo ci = CultureInfoProvider.GetCultureInfo("AR-Sa");

        // If both exists
        if ((si != null) && (ci != null))
        {
            // Add culture to site
            CultureSiteInfoProvider.AddCultureToSite(ci.CultureID, si.SiteID);
```

```
        }
```

10.6 Custom providers

10.6.1 Writing custom providers

Providers in Kentico CMS are classes that allow the system to manipulate objects and perform various actions. The following list presents the providers that the system uses and their purpose:

- **Info providers** contain methods for working with specific types of objects stored in the database, such as users, contacts, custom table records, etc.
- **Data provider** handles low-level database operations. The Info providers are built on top of the Data provider and use it to store and fetch data.
- **E-mail provider** manages the e-mail queue and sends e-mails.
- **Search provider** ensures the functionality of the SQL search.
- **Storage providers** allow you to access various file systems. They extend the CMS.IO namespace. See File system access API for information how to write a custom file system provider.

By developing custom providers and using them instead of the standard ones, you can modify the behavior of the application (or a specific module) according to your exact requirements.

Adding your custom code files

When developing custom functionality, you can place your code files in:

- The **App_Code** folder (named **Old_App_Code** if you use a web application project)
- A separate project (assembly) integrated into the main web project

Placing your customizations into the App_Code folder ensures that the code is compiled dynamically when required and automatically referenced in all other parts of the application.

Writing the custom code

Every class used to customize the application must **inherit from the original class**. This allows you to implement your modifications or additions by overriding the members of the given class. When creating overrides for existing methods, it is recommended to call the original base method within your custom code.

You can find information about the available provider, helper and manager classes in the Kentico CMS API reference, including their inheritance hierarchy and lists of methods.

Registering custom providers

After you write the code, you must register your custom classes to ensure that the system uses them instead of the default providers. You can choose between two options of registering custom provider classes:

- In App_Code using the CMSModuleLoader partial class - the most straightforward way to register providers.

- Using the web.config file - allows you to switch between different providers (custom or default) without having to edit the application code.

Other customization options

In addition to providers, you can also customize the following helper classes:

Helper Class Name	Namespace	Description
AutomationHelper	CMS.DataCom.Automation CMS.SalesForce.Automation	These two helper classes provide marketing automation support for common actions related to Data.com and SalesForce integration.
CacheHelper	CMS.GlobalHelper	Handles operations with cache items.
ClassHelper	CMS.SettingsProvider	Takes care of dynamically loaded classes and assemblies.
CookieHelper	CMS.GlobalHelper	Contains methods for managing cookies.
DirectoryHelper	CMS.IO	Manages directories in the file system.
LeadReplicationHelper	CMS.SalesForce	Replicates contacts into SalesForce leads.
MediaHelper	CMS.GlobalHelper	Provides methods for rendering media content.
OutputHelper	CMS.OutputFilter	Manages the HTML output and the output cache.
SyncHelper	CMS.Synchronization	Synchronizes document and object data to other instances of the application when using Content staging.
TransformatonHelper	CMS.Controls	Provides methods that can be used in ASCX transformations.

You can also use global event handlers to customize the behavior of the system. Handlers allow you to execute custom code whenever a specific event occurs in the system, such as document or object changes, various parts of the user authentication process, etc.

Examples

- Custom Info provider - demonstrates how to customize standard Info providers.
- Custom E-mail provider - shows a sample customization of the e-mail provider. Uses the web.config extensibility section to register the custom provider.
- Custom Data provider - describes the specific steps that you need to take to customize the Data provider.
- Custom SQL search provider - demonstrates how you can modify the SQL search provider.

10.6.2 Registering providers in App_Code

The most direct way to register custom providers or helpers is using the **CMSModuleLoader** partial class, which is included in the App_Code folder of your web project by default (*Old_App_Code* folder on

web application installations).

1. Extend the **CMSModuleLoader** partial class inside the App_Code folder. You can either create a new code file for this purpose, or extend an existing definition of the class.

[C#]

```
using CMS.SettingsProvider;

public partial class CMSModuleLoader
{

}
```

2. If you have defined your custom providers in external assemblies, add using statements referring to these assemblies.

3. Create a new class inside the **CMSModuleLoader** that inherits from *CMSLoaderAttribute*.

[C#]

```
private class CustomProviderLoader : CMSLoaderAttribute
{

}
```

4. Add the attribute defined by the internal class before the definition of the **CMSModuleLoader** partial class.

[C#]

```
[CustomProviderLoader]
public partial class CMSModuleLoader
{
...
```

5. Override the **Init** method inside the attribute class.

[C#]

```
public override void Init()
{

}
```

6. In the *Init* method, assign a new instance of your custom provider class to the **ProviderObject** property of the original provider (*UserInfoProvider* in this example).

[C#]

```
public override void Init()
{
    UserInfoProvider.ProviderObject = new CustomUserInfoProvider();
}
```

You can assign classes from both App_Code and other assemblies.

Note: When customizing helpers, assign a new instance of your custom class into the **HelperObject** property instead.

Overall code example:

[C#]

```
using CMS.SettingsProvider;
using CMS.SiteProvider;
using CMS.GlobalHelper;

[CustomProviderLoader]
public partial class CMSModuleLoader
{
    /// <summary>
    /// Attribute class that ensures the loading of custom providers.
    /// </summary>
    private class CustomProviderLoader : CMSLoaderAttribute
    {
        /// <summary>
        /// Called automatically when the application starts.
        /// </summary>
        public override void Init()
        {
            // Registers the 'CustomUserInfoProvider' class as the
UserInfoProvider
            UserInfoProvider.ProviderObject = new CustomUserInfoProvider();

            // Registers the 'CustomCacheHelper' class as the CacheHelper
            CacheHelper.HelperObject = new CustomCacheHelper();
        }
    }
}
```

The application now uses your customized provider(s) instead of the default functionality.

10.6.3 Registering providers via the web.config

You can register custom providers and helpers through your application's **web.config** file. This is done by mapping provider objects to the appropriate custom classes. Registering providers in the web.config allows you to switch between different providers (custom or default) without having to edit the application

code

Note: If you do not need your provider customizations to be adjustable through the web.config file, it is more efficient to register custom providers directly through the API in the App_Code folder (as described in Registering providers in App_Code).

1. Edit your **web.config** file and add the following section into the <*configSections*> element:

```
<?xml version="1.0"?>
<configuration xmlns="http://schemas.microsoft.com/.NetConfiguration/v2.0">
  <configSections>

  ...

    <!-- Extensibility BEGIN -->
      <section name="cms.extensibility"
    type="CMS.SettingsProvider.CMSExtensibilitySection" />
    <!-- Extensibility END -->

  </configSections>
...
```

This defines the *cms.extensibility* web.config section where you can register custom providers and helpers.

2. Create the actual **<cms.extensibility>** section directly under the root level of the web.config.

3. Assign your custom classes to providers or helpers via <*add*> elements with the following attributes:

- **name** - must match the name of the provider/helper class that you wish to customize (without the extension).
- **assembly** - specifies the assembly where your custom class is located. Set the value to *App_Code* for classes placed in the App_Code folder (even on web application installations, where the actual name of the folder is *Old_App_Code*).
- **type** - specifies the name of your custom provider class (including namespaces).
  - If the class is in the App_Code folder, the system triggers the *OnGetCustomClass* event when requesting the provider class. The *type* value is passed on through the *ClassName* property of the event's *ClassEventArgs* parameter. When writing the event's handler, use the value to identify and load an instance of the appropriate custom provider class.

You need to categorize the <*add*> elements under <*providers*> or <*helpers*> sub-sections based on the type of the customized class.

```
<configuration>
...

  <!-- Extensibility BEGIN -->
  <cms.extensibility>
    <providers>
      <add name="EmailProvider" assembly="App_Code" type="CustomEmailProvider" />
```

```
    <add name="SiteInfoProvider" assembly="App_Code"
type="CustomSiteInfoProvider" />
    <add name="ForumPostInfoProvider" assembly="CMS.CustomProviders"
    type="CMS.CustomProviders.CustomForumPostInfoProvider" />
    ...
  </providers>
  <helpers>
    <add name="CacheHelper" assembly="App_Code" type="CustomCacheHelper" />
    ...
  </helpers>
</cms.extensibility>
<!-- Extensibility END -->

...

</configuration>
```

In the case of custom providers placed in a separate assembly, the registration is now complete.

Loading App_Code classes

When using the web.config to register custom providers defined in the App_Code folder, you also need to ensure that the system can load the classes.

Example:

[C#]

```
using System;

using CMS.SettingsProvider;

[ClassLoader]
public partial class CMSModuleLoader
{
    /// <summary>
    /// Attribute class that ensures the loading of custom classes.
    /// </summary>
    private class ClassLoader : CMSLoaderAttribute
    {
        /// <summary>
        /// Called automatically when the application starts.
        /// </summary>
        public override void Init()
        {
            // Assigns a handler for the OnGetCustomClass event
            // This event is triggered when the system requests a class with
App_Code as the assembly name
            ClassHelper.OnGetCustomClass += GetCustomClassEventHandler;
        }

        /// <summary>
```

```
        /// Gets the custom class object based on the specified class name (i.e.
type value).
        /// </summary>
        private static void GetCustomClassEventHandler(object sender,
ClassEventArgs e)
        {
            // Checks whether an object is already assigned to the requested
class.
            if (e.Object == null)
            {
                switch (e.ClassName)
                {
                    // Loads an instance of the 'CustomEmailProvider' class
                    case "CustomEmailProvider":
                        e.Object = new CustomEmailProvider();
                        break;

                    // Loads an instance of the 'CustomSiteInfoProvider' class
                    case "CustomSiteInfoProvider":
                        e.Object = new CustomSiteInfoProvider();
                        break;

                    // Loads an instance of the 'CustomCacheHelper' class
                    case "CustomCacheHelper":
                        e.Object = new CustomCacheHelper();
                        break;
                }
            }
        }
    }
}
```

When the system requests an App_Code provider class registered through the *cms.extensibility* section of the web.config, the *OnGetCustomClass* event is triggered and the handler method returns an instance of the appropriate class.

10.6.4 Custom Info provider

The system uses *Info providers* to work with individual types of objects. A provider class contains methods for creating, updating, getting, and deleting the data of the corresponding objects. By customizing the appropriate Info provider, you can change or extend the functionality of nearly any object in the system.

Example

The following example customizes the *ShippingOptionInfoProvider*, which the E-commerce module uses to manage shipping options for product orders. The sample customization modifies how the provider calculates shipping costs.

You can use the same general approach to customize any standard provider or helper class in Kentico CMS (i.e. those that inherit from the *AbstractProvider* or *AbstractHelper* class respectively).

Writing the custom provider

1. Open your web project in Visual Studio and create a new class in the **App_Code** folder (or **Old_App_Code** if the project is installed as a web application). For example, name the class **CustomShippingOptionInfoProvider.cs**.

2. Edit the class and add the following references:

[C#]

```
using CMS.Ecommerce;
using CMS.SettingsProvider;
using CMS.SiteProvider;
```

3. Modify the class's declaration so that it inherits from the **CMS.Ecommerce.ShippingOptionInfoProvider** class:

[C#]

```
/// <summary>
/// Customized ShippingOptionInfo provider.
/// </summary>
public class CustomShippingOptionInfoProvider : ShippingOptionInfoProvider
{

}
```

Custom providers must always inherit from the original provider class. This ensures that the custom provider supports all of the default functionality and allows you to add your own customizations by overriding the existing members of the class.

4. Add the following override of the **CalculateShippingInternal** method into the class:

[C#]

```
/// <summary>
/// Calculates the shipping price for the specified shopping cart (shipping taxes
are not included).
/// Returns the shipping price in the site's main currency.
/// </summary>
/// <param name="cart">
/// Object representing the related shopping cart. Provides the data used to
calculate the shipping price.
/// </param>
protected override double CalculateShippingInternal(ShoppingCartInfo cart)
{
    // Checks that the shopping cart specified in the parameter exists
    if (cart != null)
    {
        // Ensures free shipping for customers who belong to the 'Gold partner'
role
```

```
        if ((cart.User != null) && cart.User.IsInRole("GoldPartners",
cart.SiteName))
        {
            return 0;
        }
        else
        {
            // Gets the customer's shipping address details from the shopping cart
            AddressInfo address = AddressInfoProvider.GetAddressInfo
(cart.ShoppingCartShippingAddressID);

            if (address != null)
            {
                // Gets the country from the shipping address
                CountryInfo country = CountryInfoProvider.GetCountryInfo
(address.AddressCountryID);

                // Sets an extra shipping charge for addresses outside of the USA
                if ((country != null) && (country.CountryName.ToLowerCSafe() !=
"usa"))
                {
                    // Replace this flat charge with your own shipping
calculations
                    // You can also create a custom setting and use it to
dynamically load the value
                    double extraCharge = 10;

                    // Returns the standard shipping price with the extra charge
added
                    return base.CalculateShippingInternal(cart) + extraCharge;
                }
            }
        }
    }

    // Calculates the shipping price using the default method for regular
customers from the USA
    return base.CalculateShippingInternal(cart);
}
```

This override of the *CalculateShippingInternal* method ensures that:

- Users who belong to the *Gold partner* role always have free shipping
- An extra flat charge is added for customers with shipping addresses outside of the USA

Note: The override gets the standard shipping price by calling the base method from the parent class. This allows you to calculate the default price based on the shipping option settings, and then modify it as needed.

Registering the provider class

Use the following steps to register your custom provider directly in App_Code:

1. Extend the **CMSModuleLoader** partial class. You can either create a new App_Code class file for this purpose or add the partial class declaration directly into the file containing your custom provider (*CustomShippingOptionInfoProvider.cs* in this case).

[C#]

```
public partial class CMSModuleLoader
{
}
```

2. Create a new attribute class inside the **CMSModuleLoader** that inherits from *CMSLoaderAttribute*. Add the attribute to the **CMSModuleLoader** partial class:

[C#]

```
using CMS.SettingsProvider;

[CustomProviderLoader]
public partial class CMSModuleLoader
{
    /// <summary>
    /// Attribute class that ensures the loading of custom providers.
    /// </summary>
    private class CustomProviderLoader : CMSLoaderAttribute
    {
    }
}
```

3. Override the **Init** method inside the attribute class and assign a new instance of the *CustomShippingOptionInfoProvider* class into the **ProviderObject** property of the original *ShippingOptionInfoProvider*.

[C#]

```
using CMS.Ecommerce;
using CMS.SettingsProvider;

[CustomProviderLoader]
public partial class CMSModuleLoader
{
    /// <summary>
    /// Attribute class that ensures the loading of custom providers.
    /// </summary>
    private class CustomProviderLoader : CMSLoaderAttribute
    {
        /// <summary>
        /// Called automatically when the application starts.
        /// </summary>
        public override void Init()
        {
            // Registers the 'CustomShippingOptionInfoProvider' class as the
ShippingOptionInfoProvider
            ShippingOptionInfoProvider.ProviderObject = new
CustomShippingOptionInfoProvider();
        }
    }
```

```
}
```

Result

You can try out how the customization affects the e-commerce module on the sample Corporate site.

1. Open the live website and log on as the Sample Gold Partner user (user name *gold*).
2. Navigate to the **Products** section and add any product to the shopping cart.
3. Go through the checkout process for the order.

In the **Order preview** (step 5 of the checkout process), you can see that the shipping is always free, no matter what shipping address you entered.

If you repeat the same process as a regular user (e.g. *Andy*), the system calculates the shipping costs normally and adds the extra charge if the shipping address is in a different country than the USA.

10.6.5 Custom E-mail provider

Customizing the e-mail provider allows you to:

- Execute custom actions when sending e-mails (for example logging the sent e-mails for auditing purposes)
- Use third-party components for sending e-mails

Once you create and register your custom e-mail provider, it is used to process all e-mails sent out by Kentico CMS and its modules.

Example

The following example demonstrates how to define a custom e-mail provider. This sample provider supports all of the default e-mail functionality, but additionally creates an entry in the Event log whenever the system sends an e-mail.

Writing the custom e-mail provider

1. Open your web project in Visual Studio and create a new class in the **App_Code** folder (or **Old_App_Code** if the project is installed as a web application). For example, name the class **CustomEmailProvider.cs**.

2. Edit the class and add the following references:

[C#]

```
using System.ComponentModel;
using System.Net.Mail;

using CMS.EmailEngine;
using CMS.EventLog;
using CMS.SettingsProvider;
```

3. Modify the class's declaration to make it inherit from the **CMS.EmailEngine.EmailProvider** class:

[C#]

```
/// <summary>
/// Customized e-mail provider.
/// </summary>
public class CustomEmailProvider : EmailProvider
{

}
```

Custom e-mail providers must always inherit from the default **EmailProvider** class. This allows you to call the base methods before adding your own custom functionality.

4. Add the following method overrides into the class:

[C#]

```
/// <summary>
/// Customized e-mail provider.
/// </summary>
public class CustomEmailProvider : EmailProvider
{
    /// <summary>
    /// Synchronously sends an e-mail through the SMTP server.
    /// </summary>
    /// <param name="siteName">Site name</param>
    /// <param name="message">E-mail message</param>
    /// <param name="smtpServer">SMTP server</param>
    protected override void SendEmailInternal(string siteName, MailMessage
message, SMTPServerInfo smtpServer)
    {
        base.SendEmailInternal(siteName, message, smtpServer);

        string detail = string.Format("E-mail from {0} was sent via {1}
(synchronously)", message.From.Address, smtpServer.ServerName);

        EventLogProvider.LogInformation("CMSCustom", "EMAIL SENDOUT", detail);
    }


    /// <summary>
    /// Asynchronously sends an e-mail through the SMTP server.
    /// </summary>
    /// <param name="siteName">Site name</param>
    /// <param name="message">E-mail message</param>
    /// <param name="smtpServer">SMTP server</param>
    /// <param name="emailToken">E-mail token that represents the message being
sent</param>
    protected override void SendEmailAsyncInternal(string siteName, MailMessage
message, SMTPServerInfo smtpServer, EmailToken emailToken)
    {
        base.SendEmailAsyncInternal(siteName, message, smtpServer, emailToken);
```

```
        string detail = string.Format("E-mail from {0} was dispatched via {1}
(asynchronously)", message.From.Address, smtpServer.ServerName);

        EventLogProvider.LogInformation("CMSCustom", "EMAIL SENDOUT", detail);
    }


    /// <summary>
    /// Raises the SendCompleted event after the send is completed.
    /// </summary>
    /// <param name="e">Provides data for the async SendCompleted event</param>
    protected override void OnSendCompleted(AsyncCompletedEventArgs e)
    {
        base.OnSendCompleted(e);

        string detail = "Received callback from asynchronous dispatch";

        EventLogProvider.LogInformation("CMSCustom", "SEND COMPLETE", detail);
    }
}
```

The provider uses these methods to send out e-mails. Each method calls the base from the parent provider class and then logs an information event into the system's event log. You can use the same approach to add any type of custom functionality.

Registering the provider class (via the web.config)

This example uses the web.config approach for registering the custom e-mail provider:

1. Open your application's **web.config** file and add the following section into the *configSections* element (if it is not already present):

```
<?xml version="1.0"?>
<configuration xmlns="http://schemas.microsoft.com/.NetConfiguration/v2.0">
  <configSections>

  ...

    <!-- Extensibility BEGIN -->
      <section name="cms.extensibility"
    type="CMS.SettingsProvider.CMSExtensibilitySection" />
    <!-- Extensibility END -->

  </configSections>
...
```

This defines the *cms.extensibility* web.config section where you can register custom providers, helpers and managers.

2. Create the actual **cms.extensibility** section directly under the root level of the web.config and add the e-mail provider:

```
...

  <!-- Extensibility BEGIN -->
  <cms.extensibility>
    <providers>
      <add name="EmailProvider" assembly="App_Code"
  type="EmailProviderWithEventLog" />
    </providers>
  </cms.extensibility>
  <!-- Extensibility END -->

...

</configuration>
```

Because the custom provider is defined in App_Code, you also need to ensure that the system can load the class.

3. Extend the **CMSModuleLoader** partial class. You can either create a new class file for this purpose or add the partial class declaration directly into the file containing your custom provider (*CustomEmailProvider.cs* in this case).

[C#]

```
public partial class CMSModuleLoader
{
}
```

4. Create a new attribute class inside the **CMSModuleLoader** that inherits from *CMSLoaderAttribute*. Add the attribute to the **CMSModuleLoader** partial class:

[C#]

```
using CMS.SettingsProvider;

[ClassLoader]
public partial class CMSModuleLoader
{
    /// <summary>
    /// Attribute class that ensures the loading of custom classes.
    /// </summary>
    private class ClassLoader : CMSLoaderAttribute
    {
    }
}
```

5. Add the following code into the attribute class:

[C#]

```
/// <summary>
/// Called automatically when the application starts.
/// </summary>
public override void Init()
{
    // Assigns a handler for the OnGetCustomClass event
    ClassHelper.OnGetCustomClass += GetCustomClassEventHandler;
}

/// <summary>
/// Gets the custom class object based on the specified class name.
/// </summary>
private static void GetCustomClassEventHandler(object sender, ClassEventArgs e)
{
    if (e.Object == null)
    {
        // Checks the name of the requested class
        switch (e.ClassName)
        {
            // Gets an instance of the 'CustomEmailProvider' class
            case "EmailProviderWithEventLog":
                e.Object = new CustomEmailProvider();
                break;
        }
    }
}
```

Note: The string used to identify the custom e-mail provider class must match the value of the **type** attribute set for the corresponding *add* element in the web.config extensibility section (*EmailProviderWithEventLog* in this example).

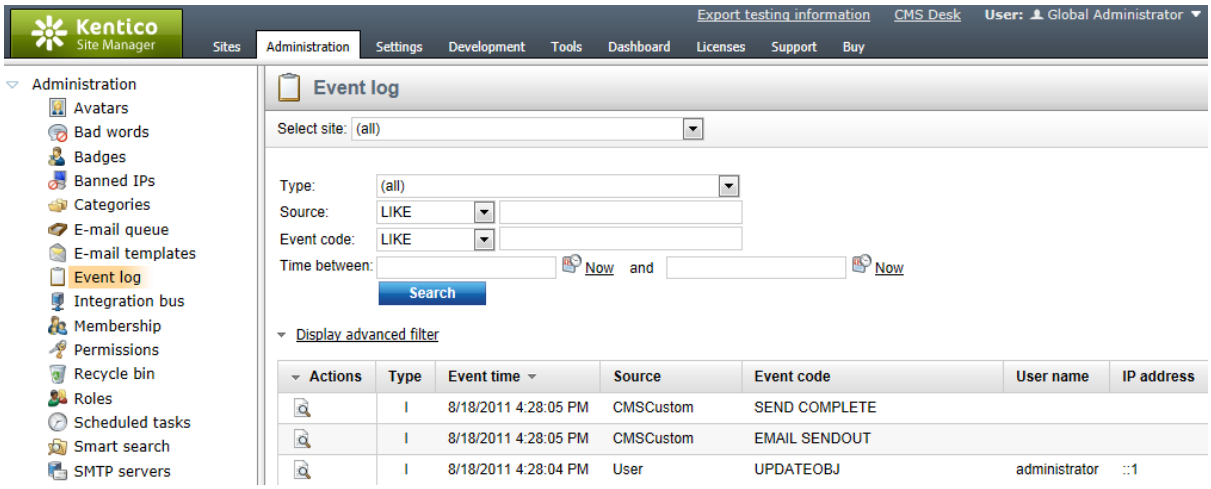
Result

The system now uses the custom e-mail provider (defined in the *CustomEmailProvider* class) when sending e-mails.

To try out the functionality:

1. Send some e-mails using Kentico CMS.
   - For example, you can use the **Site Manager -> Administration -> System -> E-mail** interface to send a test e-mail (synchronously).

2. Check the log in **Site Manager -> Administration -> Event log**.



Every successfully sent e-mail should have its own entry identified by the *CMSCustom* **Source** value (two entries for e-mails sent asynchronously).

10.6.6 Custom Data provider

Customization of the data provider allows you to implement your own database connector.

You cannot modify the data provider in the **App_Code** folder. The customizations must be added as part of a new assembly, which you then need to register via the **CMSDataProviderAssembly** web.config key.

Note

Custom data providers are NOT intended for accessing non-Microsoft SQL Server database engines. They only allow you to modify how queries are executed against the Microsoft SQL Server.

Adding the custom data provider assembly

1. Copy the **CustomDataProvider** project from your Kentico CMS installation directory (typically *C:\Program Files\KenticoCMS\<version>\CodeSamples\CustomDataProvider*) to a development folder.

2. Open your CMS web project in Visual Studio through the **WebProject.sln** file (or **WebApp.sln** if you are using a web application installation).

3. Click **File -> Add -> Existing Project** and select **CustomDataProvider.csproj** in the folder where you copied the CustomDataProvider project.

4. Unfold the **References** section of the **CustomDataProvider** project and delete all invalid references.

5. Right-click the **CustomDataProvider** project and select **Add Reference**.

6. Open the **Browse** tab of the **Add Reference** dialog and navigate to the **bin** folder of your CMS web

project on the disk.

7. Add references to the following libraries:

- CMS.DataEngine.dll
- CMS.SettingsProvider.dll

8. Right-click the CMS website object (or the **CMSApp** project if your installation is a web application) and select **Add Reference**.

9. Open the **Projects** tab and add a reference to the **CustomDataProvider** project.

The *CustomDataProvider* project is now integrated into your application. By default, the classes in the custom data provider are identical to the ones used by default, but you can modify them according to your own requirements. Remember to **Build** the *CustomDataProvider* project after making changes.

Registering the custom data provider

1. Edit your application's **web.config** file.

2. Add the following key to the **configuration/appSettings** section:

```
<add key="CMSDataProviderAssembly" value="CMS.CustomDataProvider"/>
```

The system loads the *DataProvider* class from the namespace specified as the value of the *CMSDataProviderAssembly* key.

Your application now uses the custom data provider for handling database operations.

10.6.7 Custom SQL search provider

Kentico CMS allows you to write your own SQL search provider, which you can use to integrate a third-party search engine. You can also create a custom search provider if you need to modify or filter search results returned by the standard search engine.

You cannot customize the search provider in the **App_Code** folder. The modifications must be added as part of a new assembly, which you then need to register via the **CMSSearchProviderAssembly** web.config key.

Note

This provider only affects the SQL Search engine. The Smart Search engine remains unchanged.

To modify the smart search, create and use a custom search index as described in Modules -> Smart Search -> Managing indexes -> Defining custom index content.

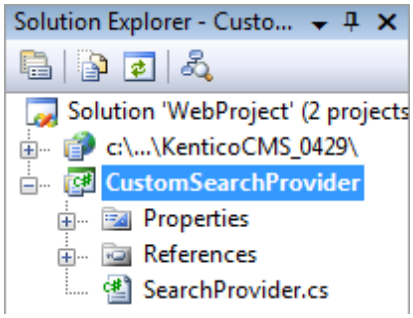
The following example demonstrates how to customize the SQL search provider so that it only searches a limited section of the website.

Adding the custom search provider assembly

1. Copy the **CustomSearchProvider** project from your Kentico CMS installation directory (typically *C:\Program Files\KenticoCMS\<version>\CodeSamples\CustomSearchProvider*) to a development folder.

2. Open your CMS web project in Visual Studio through the **WebProject.sln** file (or **WebApp.sln** if you are using a web application installation).

3. Click **File -> Add -> Existing Project** and select **CustomSearchProvider.csproj** in the folder where you copied the CustomSearchProvider project.



4. Unfold the **References** section of the **CustomSearchProvider** project and delete all invalid references.

5. Right-click the **CustomSearchProvider** project and select **Add Reference**.

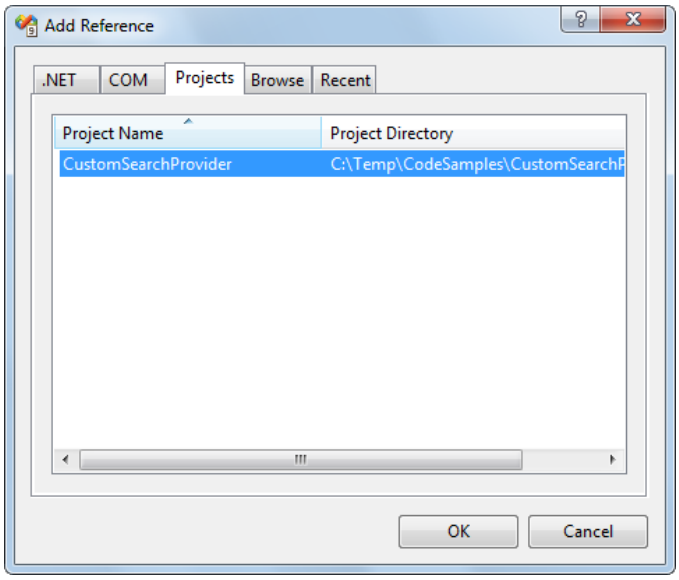
6. Open the **Browse** tab of the **Add Reference** dialog and navigate to the **bin** folder of your CMS web project on the disk.

7. Add references to the following libraries:

- CMS.ISearchEngine.dll
- CMS.SearchProviderSQL.dll

8. Right-click the CMS website object (or the **CMSApp** project if your installation is a web application) and select **Add Reference**.

9. Open the **Projects** tab and add a reference to the **CustomSearchProvider** project.



The *CustomSearchProvider* project is now integrated into your application.

Modifying the search provider class

1. Edit the **SearchProvider.cs** class in the *CustomSearchProvider* project.

2. Modify the code of the **Search** method according to the following:

[C#]

```
public DataSet Search(string siteName, string searchNodePath, string cultureCode,
string searchExpression, SearchModeEnum searchMode, bool searchChildNodes, string
classNames, bool filterResultsByReadPermission, bool searchOnlyPublished, string
whereCondition, string orderBy, bool combineWithDefaultCulture)
{
    // Limits the search path to only the /Products section of the website
    searchNodePath = "/Products/%";

    // Gets a new instance of the SQL SearchProvider
    CMS.SearchProviderSQL.SearchProvider limitedSearchProvider = new
CMS.SearchProviderSQL.SearchProvider();

    // Gets the search result dataset based on the parameters of the Search method
and the limited search path
    DataSet searchResults = limitedSearchProvider.Search(siteName, searchNodePath,
cultureCode, searchExpression, searchMode, searchChildNodes, classNames,
filterResultsByReadPermission, searchOnlyPublished, whereCondition, orderBy,
combineWithDefaultCulture);

    return searchResults;
}
```

In real customization scenarios, you can modify the **Search** method to behave according to any requirements. The method must always return a **DataSet** containing the search results.

3. Save the file and **Build** the *CustomSearchProvider* project.

Registering the custom search provider

1. Edit your application's **web.config** file.

2. Add the following key to the **configuration/appSettings** section:

```
<add key="CMSSearchProviderAssembly" value="CMS.CustomSearchProvider" />
```

The system loads the *SearchProvider* class from the namespace specified as the value of the *CMSSearchProviderAssembly* key.

Result

To try out the customized search engine, open the live website and search for text using the SQL Search engine. For example, you can use the **Examples -> Web Parts -> Full-Text Search -> SQL Search -> SQL Search dialog with results** page on the sample *Corporate site* website.

The search now only returns results from the **/Products** section of the website.

Tip

This customization example is purely demonstrational. If you need to change the search path, directly set the **Path** property of individual SQL search web parts or controls.

10.7 Data layer

10.7.1 Overview

Kentico CMS has its own data layer components that ensure unified access to the database. Each entity, such as “user", “site", “document", has its own “data class". The data class represents the data structure of the entity (using the XML schema) and SQL queries for INSERT, UPDATE, SELECT and DELETE operations. The data layer currently only works with Microsoft SQL Server, but it’s designed for use with other data engines in the future.

How it Works

When you need to create a new user record, you create a new instance of **SimpleDataClass** type and specify its class name as “cms.user", which is a code name of this entity. The system automatically creates a new DataRow based on the user entity XML Schema. Then you set the values of its attributes, such as UserName or FullName and call the Insert method. The system uses a pre-defined INSERT query to insert the values into the appropriate table that is also stored in the definition of the cms.user entity.

Related Namespaces

CMS.DataEngine
CMS.DataProviderSQL
CMS.IDataConnectionLibrary
CMS.SettingsProvider

Related Tables

CMS_Class
CMS_Query

10.7.2 Code examples

The following examples show how you can use the CMS.DataEngine and CMS.SettingsProvider libraries for low-level data manipulation.

Only for illustration

The code is used only for illustration. It's recommended that you use the **CMS.SiteProvider.UserInfoProvider** class for manipulation of the user data.

Creating a new user

[C#]

```
using CMS.DataEngine;
...

// create a new object based on the cms.user data class
IDataClass userObj = DataClassFactory.NewDataClass("cms.user");
// Set some values. The first parameter of the SetValue method accepts database
column names that belong to the particular object.
userObj.SetValue("UserName", "johns");
userObj.SetValue("FullName", "John Smith");
// insert the new user into the database, in this case, the CMS_User table
userObj.Insert();
```

Selecting and updating an existing user

[C#]

```
using CMS.DataEngine;
...

// prepare the user's ID
```

```
int userId = 10;

// get the object from database according to the data class name and ID
IDataClass userObj = DataClassFactory.NewDataClass("cms.user", userId);
// acquire the user's name
string userName = userObj.GetValue("UserName").ToString();
// change the user's full name
userObj.SetValue("FullName", "John Smith");
// update the database record
userObj.Update();
```

Deleting a user

[C#]

```
using CMS.DataEngine;
...
// prepare the user's ID
int userId = 10;

// get the object from the database
IDataClass userObj = DataClassFactory.NewDataClass("cms.user", userId);
// if the object exists, delete it
if (userObj != null)
{
    userObj.Delete();
}
```

Running a custom query

You can run a custom query if you first create it either manually in the **CMS_Query** table or through the administration interface (if this is supported for the given entity).

[C#]

```
using CMS.SettingsProvider;
...
QueryDataParameters parameters = new QueryDataParameters();
parameters.Add("@UserName", "administrator");

DataSet ds = SqlHelperClass.ExecuteQuery("cms.user.selectbyusername", parameters,
null, null);
```

10.7.3 Pre- and post-processing queries

You can pre-process database queries and post-process query results using the **OnBeforeExecuteQuery** and **OnAfterExecuteQuery** events in the **SqlHelperClass**.

Pre-processing queries

The **OnBeforeExecuteQuery** event is executed before any query is executed. Using it, you can influence the behavior of the query and its code on the fly.

The code below is the delegate definition for the event:

```
/// <summary>
/// Query execution event handler
/// </summary>
/// <param name="query">Executed query</param>
/// <param name="conn">Connection</param>
public delegate void BeforeExecuteQueryEventHandler(QueryParameters query,
IDataConnection conn);
```

And this code example shows how you can use the event. You need to create a new module class in *App_Code* (or *Old_App_Code* if you installed the project as a web application), e.g. *~\App_Code\Custom\PreProcessingQueriesModule.cs*, which will extend the *CMSModuleLoader* partial class. The class should contain the method to be executed on the event and its registration into the event in the *Init()* method, which is executed on each application start.

The method in this particular example replaces *CMS_User* with *View_CMS_User* in case that the processed query is *cms.user.selectall*.

```
using CMS.SettingsProvider;

[PrePocessingQueriesLoaderModule]
public partial class CMSModuleLoader
{
    /// <summary>
    /// Module registration
    /// </summary>
    private class PrePocessingQueriesLoaderModuleAttribute : CMSLoaderAttribute
    {
        /// <summary>
        /// Initializes the module
        /// </summary>
        public override void Init()
        {
            SqlHelperClass.OnBeforeExecuteQuery += new
SqlHelperClass.BeforeExecuteQueryEventHandler(BeforeExecuteQuery);
        }

        static void BeforeExecuteQuery(CMS.SettingsProvider.QueryParameters query,
CMS.SettingsProvider.IDataConnection conn)
        {
            if (query.Name != null)
            {
                switch (query.Name.ToLower())
                {
                    case "cms.user.selectall":
                        query.Text = query.Text.Replace("CMS_User",
"View_CMS_User");
                        break;
                }
            }
```

```
        }
    }
}
```

Post-processing queries

The **OnAfterExecuteQuery** event is raised after any query is executed and you can use it to modify the result of the query. It can handle only a *DataSet*, thereby only calls using the *ExecuteQuery* method can be post-processed this way.

The code below is the delegate definition for this event:

```
/// <summary>
/// Query execution event handler
/// </summary>
/// <param name="query">Executed query</param>
/// <param name="conn">Connection</param>
public delegate void BeforeExecuteQueryEventHandler(QueryParameters query,
IDataConnection conn);
```

And this code example shows how you can use the event. You need to create a new module class in **App_Code** (or **Old_App_Code** if you installed the project as a web application), e.g. *~\App_Code\Custom\PreProcessingQueriesModule.cs*, which will extend the **CMSModuleLoader** partial class. The class should contain the method to be executed on the event and its registration into the event in the *Init()* method, which is executed on each application start.

The method in this particular example basically gives dynamically generated full name instead of the one that is set in the user settings. This will not be reflected in the UI as you are only processing the result of the query, so please take this just as an example of how you can use the event.

```
using System.Data;

using CMS.SettingsProvider;

[PostPocessingQueriesModuleLoader]
public partial class CMSModuleLoader
{
    /// <summary>
    /// Module registration
    /// </summary>
    private class PostPocessingQueriesModuleLoaderAttribute : CMSLoaderAttribute
    {
        /// <summary>
        /// Initializes the module
        /// </summary>
        public override void Init()
        {
            SqlHelperClass.OnAfterExecuteQuery += new
SqlHelperClass.AfterExecuteQueryEventHandler(AfterExecuteQuery);
        }
```

```
        static void AfterExecuteQuery(QueryParameters query, IDataConnection conn,
ref DataSet result)
        {
            if (query.Name != null)
            {
                switch (query.Name.ToLower())
                {
                    case "cms.user.selectall":
                        if (result != null)
                        {
                            DataTable dt = result.Tables[0];
                            foreach (DataRow dr in dt.Rows)
                            {
                                dr["FullName"] = dr["FirstName"] + " " + dr
["MiddleName"] + " " + dr["LastName"];
                            }
                        }
                        break;
                }
            }
        }
    }
}
```

10.8 File system access API

10.8.1 Overview

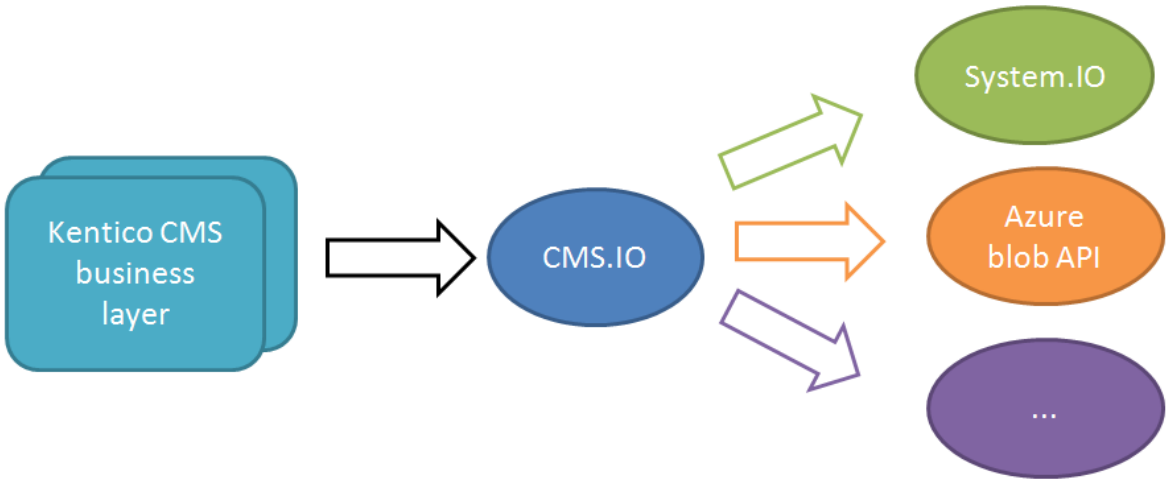
CMS.IO is a namespace that serves as an interlayer between the business layer of Kentico CMS and storages the application uses for physical files. It is used throughout the system instead of the default System.IO library provided by .NET Framework in order to be compatible with various types of file storages.

CMS.IO is an abstract layer, which accesses a file storage by means of a provider built on top of it. CMS.IO defines the classes and their methods and properties, which the provider overrides in order to manipulate files in a storage.

Depending on the storage type that you use for your files, the application utilizes one of the following providers:

- **CMS.CMSStorage** is used in the default installation which saves its files to the Windows file system. The provider only wraps around the System.IO library.
- **CMS.AzureStorage** is a custom-written provider that is employed when storing files in the Windows Azure blob storage (see the Windows Azure Deployment Guide for more information).
- **CMS.AmazonStorage** is used when you want to store files in the Amazon S3 storage service.

The following figure demonstrates how CMS.IO enables you to easily add the support of different file storages by building providers using the storages' APIs:



To learn how to use the CMS.IO namespace to access files in your custom code, refer to the Using CMS.IO topic.
To learn how to write a custom storage provider, see the Writing a custom storage provider topic.

Please note

This chapter assumes that you are familiar with the System.IO library and know how to use it to manipulate files and directories in the Windows file system. If you're new to System.IO, we recommend reading its documentation first.

10.8.2 Using CMS.IO

Working with CMS.IO is in most aspects the same as working with System.IO. We recommend using CMS.IO instead of System.IO in your code so that your custom functionality is not dependent on a single file system.

This topic describes the similarities and differences between the two libraries and presents examples of CMS.IO usage.

Classes

The following classes defined by CMS.IO are identical in function to their respective counterparts in the System.IO library:

- Directory
- DirectoryInfo
- File
- FileInfo
- FileStream
- MemoryStream
- Path
- StreamReader

- StreamWriter
- StringReader
- StringWriter

Enumerations

The following enumerations are also identical:

- FileAccess
- FileAttributes
- FileMode
- FileShare
- SearchOption
- SeekOrigin

Differences between CMS.IO and System.IO

The most significant difference is how new instances of objects are created. Instead of a constructor, each class contains a **New()** method, which accepts the same parameters as the class' constructor.

The following example shows how to write text into a file. On the highlighted line you can see that the instance of the *FileInfo* class is created using the *New()* method. Please note that the *StreamWriter* class used in the example is also a member of CMS.IO, not System.IO.

```
using CMS.IO;
...
FileInfo fi = FileInfo.New("MyFile.txt");

using (StreamWriter sw = fi.CreateText())
{
    sw.WriteLine("Hello world!");
}
```

There are a number of types which can be found in System.IO but are not implemented in CMS.IO. These include classes, class members and methods that are used rather seldom. Additionally, CMS.IO doesn't contain any definitions of **exceptions**. You will need to use System.IO exceptions or implement your own.

Working with streams

CMS.IO comes with its own set of streams and the associated readers and writers. These streams are not interchangeable with streams provided by System.IO. This can pose problems e.g. when a method accepts an object of type *CMS.IO.Stream* as a parameter, while you only have a *System.IO.Stream* available, and vice versa. However, CMS.IO provides classes that help work around the incompatibilities.

The following example shows how to store a System.IO.Stream object in a property that expects a CMS.IO.Stream.

```
using CMS.IO;
using CMS.SiteProvider;
...
MetaFileInfo mfi = new MetaFileInfo();
mfi.InputStream = StreamWrapper.New
(FileUploadControl.PostedFile.InputStream);
```

First, it creates a *MetaFileInfo* object, which represents a generic system file used throughout the CMS. The object has the *InputStream* property, which is of the *CMS.IO.Stream* type. To store the *InputStream* of a file uploaded via the standard ASP.NET upload control in that property, the **StreamWrapper.New()** method has to be called to encapsulate the *System.IO.Stream* in the *CMS.IO.Stream*.

The next example demonstrates the opposite approach - passing a *CMS.IO.Stream* to a method that accepts only objects of the *System.IO.Stream* type.

```
using System.Xml;
using CMS.IO;
...
Stream myStream = null;
XmlWriter xml = null;

myStream = FileStream.New("MyFile.xml", FileMode.Append);
xml = XmlWriter.Create(GeneralStream.New(myStream));
```

The example creates a *XmlWriter* object from a *CMS.IO.Stream*. Since *XmlWriter* is a system object, it expects that its *Create* method will be supplied with a *System.IO.Stream*. An instance of the *GereralStream* class, which inherits from *System.IO.Stream*, has to be passed to the method, as shown on the highlighted line.

Helper methods

CMS.IO contains a number of additional methods and properties, which simplify operations with files and directories. The following list describes the most useful methods:

DirectoryHelper

- **void DeleteDirectoryStructure(string path)** - deletes the directory specified by the **path** parameter and all underlying directories.
- **void EnsureDiskPath(string path, string startingPath)** - checks whether all directories between **startingPath** and **path** exist and creates them if necessary.
- **void EnsurePathBackslash(string path)** - adds a backslash to the end of the **path** specified if the backslash is missing.

Directory

- **void PrepareFilesForImport(string path)** - converts all physical media files' names to lower case to ensure compatibility with case-sensitive file systems.

StorageHelper

- **IsExternalStorage** - this property indicates if the application is using a storage other than the default Windows file system. Returns **true** if the **CMSExternalStorageName** web.config key is set.
- **ExternalStorageName** - this property returns the name of the storage provider set by **CMSExternalStorageName** web.config key. See the Configuration section of the next chapter for more information.

The complete list of CMS.IO members can be found in the Kentico CMS API Reference.

10.8.3 Writing a custom storage provider

The purpose of the CMS.IO library is to provide an easy way of customizing Kentico CMS to support a file system of your choice. As described in the Overview topic, you can achieve this by developing a custom provider based on the classes contained within CMS.IO.

Preparation

A generic storage provider project is available in the CodeSamples directory in your Kentico CMS installation (e.g. C:\Program Files (x86)\Kentico CMS\7.0). The project contains definitions of all required classes and their members for easy implementation.

1. Copy the generic project folder (**CustomFileSystemProvider**) into your web project folder.

2. Choose to **add an existing project** to your web project solution and select the custom storage provider.

3. Add **CMS.IO** reference to the new project.

4. Add the project reference to your web project.

Implementation

If you choose to take advantage of the prepared provider, go through all files contained in the project and replace the *NotImplementedExceptions* inside methods with your implementation and implement property and method overrides. You can consult the following steps for reference or if you wish to create a provider from scratch.

1. Create classes that will inherit from the following abstract classes:

- CMS.IO.AbstractDirectory
- CMS.IO.AbstractFile

2. Implement all methods defined in those abstract classes.

3. Create classes that will inherit from the following classes:

- CMS.IO.DirectoryInfo
- CMS.IO.FileInfo
- CMS.IO.FileStream

4. Override all methods and properties from those classes.

5. Create constructors for the classes listed previously according to the following table:

Inherits from	Constructors
CMS.IO.DirectoryInfo	• public DirectoryInfo(string path)
CMS.IO.FileInfo	• public FileInfo(string filename)
CMS.IO.FileStream	• public FileStream(string path, CMS.IO.FileMode mode) • public FileStream(string path, CMS.IO.FileMode mode, CMS.IO.FileAccess access) • public FileStream(string path, CMS.IO.FileMode mode, CMS.IO.FileAccess access, CMS.IO.FileShare share) • public FileStream(string path, CMS.IO.FileMode mode, CMS.IO.FileAccess access, CMS.IO.FileShare share, int bSize)

Configuration

To start using your custom provider, you need to make configuration changes in your web.config file.

1. Add the **CMSStorageProviderAssembly** key to the *appSettings* section and set its value to the assembly name of your custom provider.

2. Add the **CMSExternalStorageName** key to the *appSettings* section. You can choose any string of characters as a value. The value will be used in code to determine whether the application uses an external storage. See Using CMS.IO for more information about the code that processes the storage name.

For example, if you named the project "**CustomFileSystemProvider"** and the provider name is "**custom**", the code inserted into the *web.config* file will look like the following:

```
<appSettings>
...
  <add key="CMSStorageProviderAssembly" value="CustomFileSystemProvider" />
  <add key="CMSExternalStorageName" value="custom" />
...
</appSettings>
```

After you save the web.config, the system starts using your custom provider.

10.8.4 Configuring storage providers

Using different storage providers for specific parts of the application's file repository

CMS.IO allows you to access different sections of the application's file repository using different storage providers. For example, you can store media files in the Azure blob storage, while all other files stay in the default Windows file system.

1. Open the **CMSHttpApplication.cs** file from the *App_Code\Application* folder.

2. Add a *using* statement for **CMS.IO**.

3. In the **Application_Start** method, create a new instance of the **StorageProvider** class.

```
AbstractStorageProvider mediaProvider = new StorageProvider("Azure",
"CMS.AzureStorage");
```

The value in the **first parameter** must match the value of the **CMSExternalStorageName** settings key value of the provider that you want to use. The value in the **second parameter** must match the **assembly name** of the provider. Use the constructor without parameters to create an instance of CMSStorage - the provider for Windows file system. See the table in Storage provider names and assemblies for values belonging to built-in storage providers.

4. Use the following code to map a directory to the provider you've instantiated in step 3.

```
StorageHelper.MapStoragePath("~/MySite/Media/", mediaProvider);
```

5. Save the file.

6. If you're using web application, rebuild the solution.

The application now uses the storage provider specified in step 3 to access the project folder specified in step 4.

Changing paths to content files

CMS.IO allows you to change default file paths. For example, when using the Windows file system, you can store media files on a different disk drive.

You utilize a similar procedure to store files in a different container in Azure storage or to store files in a different bucket on Amazon S3.

1. Open the **CMSHttpApplication.cs** file from the *App_Code\Application* folder.

2. Add a *using* statement for **CMS.IO**.

3. In the **Application_Start** method, create a new instance of the **StorageProvider** class.

```
AbstractStorageProvider mediaProvider = new StorageProvider();
```

Use the constructor without parameters to create an instance of CMSStorageProvider - the provider for Windows file system. Otherwise, the method expects two string parameters. The value in the **first parameter** must match the value of the **CMSExternalStorageName** settings key value of the provider that you want to use. The value in the **second parameter** must match the **assembly name** of the provider. See the table in Storage provider names and assemblies for values belonging to built-in storage

providers.

4. Specify the target directory using the **CustomRootPath** property of the provider. This is the directory that you want to use instead of the default one.

```
mediaProvider.CustomRootPath = "D:\MySiteMedia";
```

5. (Optional) If the custom path is accessible via a URL (i.e. it is located on a file server), you can make the system access files via that URL. Insert the following code.

```
mediaProvider.CustomRootUrl = "http://www.example.com/MySiteMedia";
```

6. Use the following code to map a directory to the provider you've instantiated in step 3. This is the directory that you want to change.

```
StorageHelper.MapStoragePath("~/MySite/Media/", mediaProvider);
```

7. Save the file.

8. If you're using web application, rebuild the solution.

This code ensures that every time the application requests a file from the *~/MySite/Media* directory, the file will be served from the location specified in the *CustomRootPath* property. If you specified the *CustomRootUrl* property, links to the file will point to that URL.

Storage provider names and assemblies

The following table lists storage providers built into Kentico CMS, the CMSExternalStorageName setting values associated with the providers, and the assemblies where the providers reside.

Storage provider	CMSExternalStorageName value	Assembly
Amazon S3	Amazon	CMS.AmazonStorage
Azure blob storage	Azure	CMS.AzureStorage
Windows file system (default provider)	-	CMS.CMSStorage

10.8.4.1 Azure storage

Configuring Kentico CMS to use Azure storage

1. Add the following key into the **appSettings** section of your web.config. The key indicates that CMS.IO should use the Azure storage provider to access files.

```
<add key="CMSExternalStorageName" value="azure" />
```

2. Specify the storage account name by inserting the following key into the web.config.

```
<add key="CMSAzureAccountName" value="AccountName" />
```

3. Specify the primary access key by adding the following key into the web.config.

```
<add key="CMSAzureSharedKey" value="PrimaryAccessKey" />
```

Optionally, you can adjust the following properties:

Key	Description	Sample Value
CMSAzureTempPath	The folder specified by this key will be used to store temporary files on a local disk, e.g. when transferring large files to or from the storage account.	`<add key="CMSAzureTempPath" value="C:\AzureTemp" />`
CMSAzureCachePath	Specifies a folder on a local disk where files requested from the storage account will be cached. This helps minimize the amount of blob storage operations, which saves time and resources.	`<add key="CMSAzureCachePath" value="C:\AzureCache" />`
CMSAzureBlobEndPoint	Sets the endpoint used for the connection to the blob service of the specified storage account. If you wish to use the default endpoint, remove the setting completely from the appropriate files.	`<add key="CMSAzureBlobEndPoint" value="http://127.0.0.1:10000/devstoreaccount1" />`
CMSAzurePublicContainer	Indicates if the blob container used to store the application's file system should be public. If set to *true*, it will be possible to access files directly through the URL of the appropriate blob service, for example: *http://<StorageAccountName>.blob.core.windows.net/cmsroot/corporatesite/media/imagelibrary/logo.png*	`<add key="CMSAzurePublicContainer" value="true" />`

CMSAzureCDNEndpoint	URL of the HTTP endpoint of a Azure Blob storage CDN.	`<add key="CMSAzureCDNEndpoint" value ="kentico123.vo.msecnd.net" />`

Storing files in different containers

By default, the system stores files in a single container. However, the built-in Azure storage provider allows you to specify sections of the file system, which will be stored in a different container.

1. Open the **CMSHttpApplication.cs** file from the *App_Code\Application* folder.

2. Add a *using* statement for **CMS.IO**.

3. In the **Application_Start** method, create a new instance of the Azure storage provider.

```
AbstractStorageProvider mediaProvider = new StorageProvider("Azure",
"CMS.AzureStorage");
```

4. Specify the target container using the **CustomRootPath** property of the provider.

```
mediaProvider.CustomRootPath = "mymediacontainer";
```

Optionally, you can specify whether you want the container to be publicly accessible using the **PublicExternalFolderObject** property of the provider. *True* means the container is publicly accessible.

```
mediaProvider.PublicExternalFolderObject = true;
```

5. Use the following code to map a directory to the provider you've instantiated in step 3. This is the directory that you want to store in the container.

```
StorageHelper.MapStoragePath("~/MySite/Media/", mediaProvider);
```

6. Save the file.

7. If you're using web application, rebuild the solution.

10.8.4.2 Amazon S3

Configuring Kentico CMS to use Amazon S3

1. Make sure you that have your Amazon S3 account set up and and that you have created at least one bucket.

2. Add the following key into the **appSettings** section of your web.config. This key specifies that CMS.IO will use the Amazon S3 provider..

```
<add key="CMSExternalStorageName" value="amazon" />
```

3. Add the following key into the **appSettings** section of your web.config. This key specifies the bucket which you want to use to store files. Replace the value with the name of the bucket.

```
<add key="CMSAmazonBucketName"  value="YourBucketName" />
```

4. Specify the ID of your Amazon access key by inserting the following key into the **appSettings** section of your web.config.

```
<add key="CMSAmazonAccessKeyID" value="YourKey" />
```

5. Specify the Amazon access key by adding the following key into the **appSettings** section of your web.config.

```
<add key="CMSAmazonAccessKey" value="YourSecret" />
```

After you save your web.config, Kentico CMS starts storing its files in the specified Amazon S3 bucket.

You can configure the following optional settings:

Key	Description	Sample Value
CMSAmazonTempPath	Path to a local directory that will be used for storing temporary files. If you don't specify a value, the provider will use the default operating system temporary directory.	`<add key ="CMSAmazonTempPath" value="C:\Windows\Temp" />`
CMSAmazonCachePath	Path to a local directory where the provider will store cached files. If you don't use this setting, the system will create a directory in the default operating system temporary directory.	`<add key="CMSAmazonCachePath" value="C:\Cache" />`
CMSAmazonEndPoint	Allows to change the default endpoint, for example when you want to use CloudFront CDN. The default endpoint address is *http://<yourbucketname>.s3.amazonaws.com*	`<add key="CMSAmazonEndPoint" value="http://someendpoint.s3.amazonaws.com" />`
CMSAmazonPublicAccess	Specifies whether files uploaded to Amazon S3 through Kentico CMS will be accessible for public users. The default value is true if you specify an endpoint. If no endpoint is specified, the	`<add key ="CMSAmazonPublicAccess" value="true"/>`

	default value is false.	

Storing files in different buckets

By default, the system stores files in a single bucket. However, the built-in Amazon S3 storage provider allows you to specify sections of the file system, which will be stored in a different bucket.

1. Open the **CMSHttpApplication.cs** file from the *App_Code\Application* folder.

2. Add a *using* statement for **CMS.IO**.

3. In the **Application_Start** method, create a new instance of the Amazon S3 storage provider.

```
AbstractStorageProvider mediaProvider = new StorageProvider("Amazon",
"CMS.AmazonStorage");
```

4. Specify the target bucket using the **CustomRootPath** property of the provider.

```
mediaProvider.CustomRootPath = "mymediabucket";
```

Optionally, you can specify whether you want the bucket to be publicly accessible using the **PublicExternalFolderObject** property of the provider. *True* means the bucket is publicly accessible.

```
mediaProvider.PublicExternalFolderObject = true;
```

5. Use the following code to map a directory to the provider you've instantiated in step 3. This is the directory that you want to store in the bucket.

```
StorageHelper.MapStoragePath("~/MySite/Media/", mediaProvider);
```

7. Save the file.

8. If you're using web application, rebuild the solution.

10.9 Global events and their handling

10.9.1 Global events

Global events allow you to execute custom actions when some CMS event occurs. For example, if a document is created in the CMS system, you can handle the corresponding event, get information about the new document and send its content by e-mail or use a third-party component to generate a PDF version of the document.

Please note

If you are accustomed to using custom handler libraries, please note that these global events provide the same functionality in a much simpler way. You can refer to the Global events (obsolete) topic to see the difference between the two approaches.

Handler classes

Modules with handlers contain classes with "Events" in their name, which provide the available handlers. They may also contain classes that store the handler arguments or handler results. The following are samples of classes that are available:

- **<name>EventArgs** - e.g., *DocumentEventArgs*, is an object that contains only properties and is used to store event data. It inherits from the *System.EventArgs* class.
- **<name>Events** - e.g., *DocumentEvents*, is a static class containing definitions of handlers for various events as static properties.

Registering event handlers

Registration of handlers for events can be performed by creating a custom class in the *App_Code* folder, which eliminates the need to add a dedicated handler library and update it when performing upgrades to new versions. The class needs to extend the *CMSModuleLoader* partial class and a new attribute must be added to it. In the *Init()* method of a private class representing the attribute, handlers can be assigned to the required events in the following format:

<event class>.<action>.<before or after> += <handler name>

The definitions of the handlers can then be added directly into the attribute class as shown in the example below:

[C#]

```
using CMS.DocumentEngine;
using CMS.SettingsProvider;

[CustomDocumentEvents]
public partial class CMSModuleLoader
{
    /// <summary>
    /// Attribute class that ensures the loading of custom handlers
    /// </summary>
    private class CustomDocumentEventsAttribute : CMSLoaderAttribute
    {
        /// <summary>
        /// Called automatically when the application starts
        /// </summary>
        public override void Init()
        {
            // Assigns custom handlers to the appropriate events
```

```
            DocumentEvents.Insert.After += Document_Insert_After;
            DocumentEvents.InsertLink.Before += Document_InsertLink_Before;
        }

        private void Document_Insert_After(object sender, DocumentEventArgs e)
        {
            // Add custom actions here
        }

        private void Document_InsertLink_Before(object sender, DocumentEventArgs
e)
        {
            // Add custom actions here
        }
    }
}
```

Events that you can handle

You can use the following event classes and their respective events. These events cover data management for both objects and documents.

- **ObjectEvents** - contains events fired upon any object change. Replaces the old *CustomDataHandler*. These events are fired for any object type in the system.
  - Insert - fired upon the insertion of a new object.
  - Update - fired upon the update of the existing object.
  - Delete - fired upon the object deletion.
  - GetContent - provides the additional object content to the search indexer.
  - LogChange - allows you to alter how the system logs object changes for the purposes of staging and integration.

- **<name>Info.TYPEINFO.Events** - provides the same set of events as *ObjectEvents*, but specific for a particular object type, for example *UserInfo.TYPEINFO.Events*.

- **ColumnTranslationEvents** - contains handlers that you can use to store objects' IDs to ensure the IDs' translation during import/export and staging.
  - RegisterRecords - using this handler you can provide additional information for your custom field translation.
  - TranslateColumns - using this handler you can translate the field value using the provided info.

- **DocumentEvents** - contains events fired when changes are made to documents. These events relate to the published versions of documents, refer to *WorkflowEvents* for actions within editing of the documents under workflow. Replaces the old *CustomTreeNodeHandler*.
  - Insert - fired upon insertion of a new document.
  - InsertNewCulture - fired upon insertion of a new document culture version.
  - InsertLink - fired upon insertion of a new document link.
  - Update - fired upon update of any document.
  - Delete - fired upon deletion of any document.
  - Move - fired upon moving of any document.
  - Copy - fired upon copying of any document.
  - GetContent - provides additional document content to the search indexer.
  - LogChange - allows you to alter the way document changes are being logged.

- **NewsletterEvents** ***[Only available after applying hotfix 7.0.52 or newer]***
  - ResolveMacro - fired when all newsletter text macros are being resolved.

- **SystemEvents** - contains general system events, replaces the old *CustomExceptionHandler*.
  - Exception - fired when unhandled exception occurs.

- **SecurityEvents** - contains security and authentication events, replaces the old *CustomSecurityHandler*.
  - Authenticate - fired upon user authentication.
  - AuthorizeResource - fired upon security check for particular module permission.
  - AuthorizeClass - fired upon security check for particular object type or document type permission.
  - AuthorizeUIElement – fired upon permission check for particular UI element.

- **ActivityEvents** ***[Only available after applying hotfix 7.0.46 or newer]***
  - ActivtyLogged - fired immediately when on-line marketing activities of any type occur in the system. The event does not trigger for custom activities logged using the *ActivityInfoProvider.SetActivityInfo* method.

- **WorkflowEvents** – contains events related to specific workflow and versioning actions, replaces the old *CustomWorkflowHandler*.
  - Approve – fired when a document is approved to the next step.
  - Reject – fired when a document is rejected to a previous step
  - Publish – fired when a document is being published.
  - Archive – fired when a document is being archived.
  - Action – fired when a workflow action step is being executed.

The following events are specific to content locking within versioning.
- CheckOut – fired when the document is checked-out.
- CheckIn – fired when the document is checked-in.
- UndoCheckOut – fired when undoing the check-out is performed on the document.

The following events are specific to updating a document version.
- SaveVersion – fired when the edited version of the document is updated.
- SaveAttachmentVersion – fired when the attachment of the edited version of the document is updated or inserted.
- RemoveAttachmentVersion – fired when the attachment of the edited version of the document is removed.

The following events are related to document attachments:
- SaveAttachment – fired upon insertion or update of document attachment.
- DeleteAttachment – fired upon deletion of document attachment.

The following events are related to customizating document security:
- AuthorizeDocument – fired when security check for particular document is requested.
- FilterDataSetByPermissions – fired when the DataSet of documents is filtered according the document permissions.

Application events

There are several sets of application events that you can leverage to customize the global behavior of the application, these are following:

- **CMSSessionEvents** – contains events related to session management of the application.
  - Start – fired when the session starts
  - End – fired when the sessions ends.

- **CMSApplicationEvents** – contains events related to the whole application run.
  - Start – fired when the application starts. It is recommended to bind only *After* handlers to make sure your handler has all necessary data initialized.
  - End – fired when the application ends.
  - Error – fired when application error occurs. Similar to SystemEvents.Exception, except for that this one covers wider range of handling code.

- **CMSRequestEvents** – contains events related to each request done to the application.
  - Begin – fired when the begin request method executes.
  - End - fired when the end request method executes.
  - Authenticate - fired when the authenticate request method executes.
  - Authorize - fired when the authorize request method executes.
  - MapRequestHandler- fired when the map request handler method executes.
  - AcquireRequestState - fired when the acquire request state method executes.

Control and page events

The last set of events is related to the life cycle of pages and controls. These events have their respective handlers directly in the base classes of pages and controls. You can leverage them to programatically customize the UI without the need to modify the original code of the solution. Please note that these events are direct delegates and do not provide *Before* / *After* sub-items.

- **AbstractUserControl** - contains events that are fired during processing of user controls.
  - OnBeforeUserControlInit
  - OnAfterUserControlInit
  - OnBeforeUserControlLoad
  - OnAfterUserControlLoad
  - OnBeforeUserControlPreRender
  - OnAfterUserControlPreRender
  - OnBeforeUserControlRender
  - OnAfterUserControlRender

- **CMSPage** - contains events that are fired within processing of the system pages.
  - OnBeforePagePreInit
  - OnAfterPagePreInit
  - OnBeforePageInit
  - OnAfterPageInit
  - OnBeforePageLoad
  - OnAfterPageLoad
  - OnBeforePagePreRender
  - OnAfterPagePreRender
  - OnBeforePageRender
  - OnAfterPageRender

10.9.2 LogChange handler

By default, Kentico logs changes that you make to objects and documents. You can use the *LogChange* handler to affect logging of these changes for tasks such as staging or integration.

Excluding documents

In the following example, you can see the use of the *LogChange* handler to exclude documents contained under the "/Home" path from staging.

Fore more information on registering handlers, visit the Global events topic.

[C#]

```
using System;

using CMS.DocumentEngine;
using CMS.SettingsProvider;

/// <summary>
/// Exclude documents under '/Home' path from staging.
/// </summary>
[CustomDocumentEvents]
public partial class CMSModuleLoader
{
    /// <summary>
    /// Ensure the loading of custom handlers
    /// </summary>
    private class CustomDocumentEventsAttribute : CMSLoaderAttribute
    {
        /// <summary>
        /// Called automatically when the application starts
        /// </summary>
        public override void Init()
        {
            // Assign custom handlers to the appropriate events
            DocumentEvents.LogChange.Before += new
EventHandler<LogDocumentChangeEventArgs>(LogDocumentChange_Before);
        }

        void LogDocumentChange_Before(object sender, LogDocumentChangeEventArgs e)
        {
            var document = e.Settings.Node;
            if (document.NodeAliasPath.StartsWith("/Home"))
            {
                e.Settings.LogStaging = false;
            }
        }
    }
}
```

Excluding objects

In the following example, you can see the use of the *LogChange* handler to exclude objects of the object type "role" whose code name starts with "CMS" from staging.

Fore more information on registering handlers, visit the Global events topic.

[C#]

```
using System;

using CMS.SettingsProvider;

/// <summary>
/// Exlude objects of the object type 'role' whose code name starts with 'CMS'
from staging.
/// </summary>
[CustomObjectEvents]
public partial class CMSModuleLoader
{
    /// <summary>
    /// Ensure the loading of custom handlers
    /// </summary>
    private class CustomObjectEventsAttribute : CMSLoaderAttribute
    {
        /// <summary>
        /// Called automatically when the application starts
        /// </summary>
        public override void Init()
        {
            // Assign custom handlers to the appropriate events
            ObjectEvents.LogChange.Before += new
EventHandler<LogObjectChangeEventArgs>(LogObjectChange_Before);
        }

        void LogObjectChange_Before(object sender, LogObjectChangeEventArgs e)
        {
            var obj = e.Settings.InfoObj;
            if ((obj.ObjectType == PredefinedObjectType.ROLE) &&
(obj.ObjectCodeName.StartsWith("CMS")))
            {
                e.Settings.LogStaging = false;
            }
        }
    }
}
```

10.9.3 Translation handlers

When transferring data to a different instance of Kentico — via import/export or staging — IDs of the same objects can differ after the transfer. Kentico handles the situation for system objects, but not for custom objects. For example, when you stage a document type or a custom table that has a field containing object IDs, then the IDs are not guaranteed to stay the same once staged to the target server.

You can use the following two handlers to make sure the IDs are translated correctly:

RegisterRecords - use this handler to provide additional information for your custom field translation.
TranslateColumns - use this handler to translate the field value using the provided information.

Example

You have a custom table that contains a UserID field. The field contains the ID of a user to whom the item belongs. Because the UserID field is a custom field, you want to ensure its correct translation during import/export and staging.

1. See how you can register the handlers.

2. Use the following code to ensure the correct translation of the UserID field.

[C#]

```
using System;

using CMS.SettingsProvider;
using CMS.GlobalHelper;
using CMS.SiteProvider;

/// <summary>
/// Ensure translation of UserID column in a custom table item
/// </summary>
[CMSModuleLoaderAttribute]
public partial class CMSModuleLoader
{
    private class CMSModuleLoaderAttribute : CMSLoaderAttribute
    {
        public override void Init()
        {
            // Register the events
            ColumnsTranslationEvents.RegisterRecords.Execute += new
EventHandler<ColumnsTranslationEventArgs>(RegisterRecords_Execute);
            ColumnsTranslationEvents.TranslateColumns.Execute += new
EventHandler<ColumnsTranslationEventArgs>(TranslateColumns_Execute);
        }

        void RegisterRecords_Execute(object sender, ColumnsTranslationEventArgs e)
        {
            if (e.ObjectType == CustomTableItemProvider.GetObjectType
("customtable.sampletable"))
            {
                // Register the object type, user ID, and username for custom
field translation
                int userId = ValidationHelper.GetInteger(e.Data.GetValue
("UserID"), 0);
                e.TranslationHelper.RegisterRecord(PredefinedObjectType.USER,
userId, UserInfoProvider.GetUserNameById(userId), 0);
            }
        }

        void TranslateColumns_Execute(object sender, ColumnsTranslationEventArgs
e)
        {
            if (e.ObjectType == CustomTableItemProvider.GetObjectType
("customtable.sampletable"))
            {
```

```
                // Translate the field using the column name, User ID, and object
type of user
                e.TranslationHelper.TranslateColumn(e.Data, "UserID",
PredefinedObjectType.USER, 0, true, true);
            }
        }
    }
}
```

10.9.4 Event handler libraries (obsolete)

10.9.4.1 Overview

Obsolete!

This way of event handling is now obsolete. It is still functional due to backward compatibility reasons, but the recommended way is to use the global event handlers.

Events that can be handled

- Data updates - insert/update/delete actions for all data items.
- Exceptions
- Security events - authentication and authorization.
- Document events - insert/update/delete/logchange operations.
- Object events - insert/update/logchange operations.
- Workflow events - document approval/rejection/publishing/archiving operations.

Enabling custom event handling

If you want to use custom event handlers, make sure the <*appSettings*> node of your web project's *web.config* file contains the following key:

```
<add key="CMSUseCustomHandlers" value="true" />
```

Default event handler library

The default custom event handler library is **CMS.CustomEventHandler**. Its name can be changed by means of a parameter in the <*appSettings*> node of your project's *web.config* file.

If the following line is inserted in the <*appSettings*> node, the library name will be changed to *CMS.CustomEventHandlerVB*:

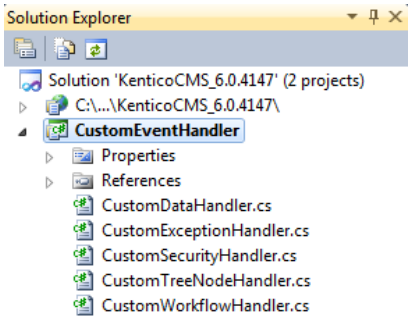
```
<add key="CMSCustomHandlersAssembly" value="CMS.CustomEventHandlerVB" />
```

If you wanted to use this library in the following series of examples, it would require you to write the code in Visual Basic instead of C#.

Writing an event handler

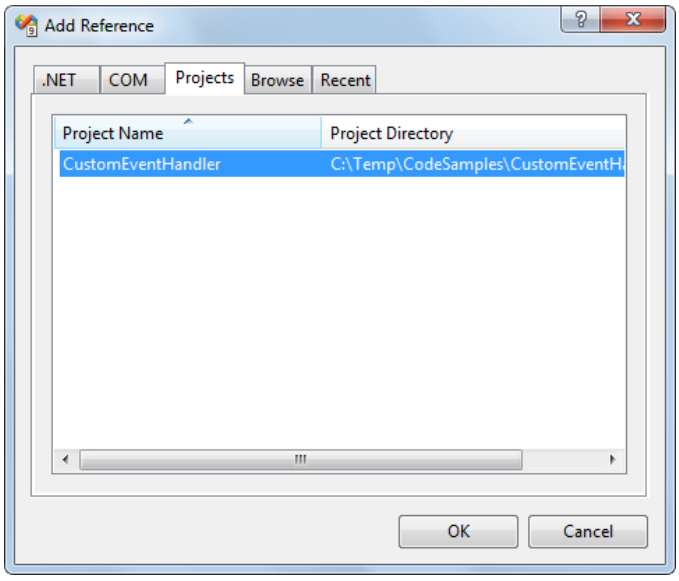
1. Copy the **CustomEventHandler** project from the Kentico CMS installation directory (typically *C:\ProgramFiles\KenticoCMS\<version_number>\CodeSamples\CustomEventHandler*) to any custom folder that is not under the root of your web project (e.g. *C:\customhandler*).

2. Open the **CMS** web project using the **WebProject.sln** file. Click **File -> Add -> Existing Project** and select the **CustomEventHandler.csproj** file located in the folder where you copied the **CustomEventHandler** project. Your Solution Explorer window should look like this:



3. Unfold the **References** section under the **CustomEventHandler** project and remove all invalid references (marked with the yellow exclamation icon).

4. Now you need to update the references. Right-click the **bin** folder and choose **Add Reference**. Choose the **Projects** tab and click **OK** to add a reference to the **CustomEventHandler** project:



5. Now right-click the **CustomEventHandler** project and choose **Add Reference**. Choose the **Browse** tab and locate the **bin** folder of your CMS web project on the disk. Choose to add a reference to the following libraries:

- CMS.GlobalEventHelper.dll
- CMS.DocumentEngine.dll
- CMS.GlobalHelper.dll
- CMS.SiteProvider.dll
- CMS.CMSHelper.dll
- CMS.SettingsProvider.dll
- CMS.Synchronization.dll
- CMS.DataEngine.dll
- CMS.EmailEngine.dll
- CMS.WorkflowEngine.dll

You can also navigate to the **bin** folder on the **Browse** tab and copy&paste the following string into the **Add Reference** dialog to add all the libraries.

```
"CMS.GlobalEventHelper.dll" "CMS.DocumentEngine.dll" "CMS.GlobalHelper.dll"
"CMS.SiteProvider.dll" "CMS.CMSHelper.dll" "CMS.SettingsProvider.dll"
"CMS.Synchronization.dll" "CMS.DataEngine.dll" "CMS.EmailEngine.dll"
"CMS.WorkflowEngine.dll"
```

If you need to use another library in your custom handlers, please add a reference to it the same way as described in this step.

6. Click **Build -> Rebuild solution**.

Now you can modify the libraries under the **CustomEventHandler** project and add your own code to handle the appropriate events. The places where you should place the code are marked with upper-case comments.

You can find more details on particular events in the following chapters.

10.9.4.2 Data handler (CustomDataHandler class)

This handler allows you to add custom actions to the following events:

- OnBeforeUpdate
- OnAfterUpdate
- OnBeforeInsert
- OnAfterInsert
- OnBeforeDelete
- OnAfterDelete
- OnGetContent

The events are applied to all data items that are stored to the database. It includes documents, user information or any other settings. They receive the data object of type **DataClass** that can be modified. In case of delete handlers, the class name and ID is provided.

> **Handling document events**
>
> For handling document events, please use the CustomTreeNodeHandler class instead

of the CustomDataHandler class. Every document uses up to 3 tables, which leads to three separate events in the CustomDataHandler class.

Example

The following example shows how to handle the OnAfterUpdate event and send the password to the user whenever their profile is updated:

1. Open the **CustomDataHandler** class and put the following code at the beginning of the file. It adds the reference to the namespaces we will use to handle the event:

[C#]

```
using CMS.DataEngine;
using CMS.GlobalHelper;
using CMS.EmailEngine;
```

2. Put the following code inside the **OnAfterUpdate** method.

[C#]

```
// type the data object as DataClass
IDataClass dataItem = (IDataClass)dataObj;

// we want to handle only updates of user objects
if (dataItem.ClassName.ToLower() == "cms.user")
{
// we will use the CMS.EmailProvider to send e-mails
EmailMessage email = new EmailMessage();
email.From = "admin@domain.com";
// get the user's e-mail address
email.Recipients = ValidationHelper.GetString(dataItem.GetValue("Email"),
"admin@domain.com");
email.Subject = "Your password";
// get the user's password
email.Body = "Your password is:" + ValidationHelper.GetString(dataItem.GetValue
("UserPassword"), "");
EmailSender.SendEmail(email);
}
```

Please note: getting user's password in the example above is only possible when passwords are stored in Plain text format; the setting is located in **Site Manager -> Settings -> Security & Membership**.

3. Set the From and Recipients e-mail addresses to your e-mail address.

4. If you don't have reference to CMS.EmailEngine in your CustomEventHandler, add it there.

5. Compile and run the project. Edit some user profile that uses your e-mail address. You should receive the e-mail message with your password.

10.9.4.3 Exception handler (CustomExceptionHandler class)

This class processes all **exceptions** that occur in the web application. You can use it to handle all exceptions and run custom actions, such as sending e-mail to administrator or logging the exception to your helpdesk or monitoring system.

The class has only one method: **OnException(Exception e)**

Example

The following example shows how to handle the OnException event and send the password to the administrator whenever an error occurs:

1. Open the **CustomExceptionHandler** class and put the following code at the beginning of the file. It adds the reference to the namespaces we will use to handle the event:

[C#]

```
using CMS.EmailEngine;
```

2. Put the following code inside the **OnException** method.

[C#]

```
// we will use the CMS.EmailProvider to send e-mails
EmailMessage email = new EmailMessage();
email.From = "admin@domain.com";
email.Recipients = "admin@domain.com";
email.Subject = "Exception";
email.Body = e.Message + "<br />" + e.StackTrace + "<br />" + e.Source;
EmailSender.SendEmail(email);
```

3. Set the From and Recipients e-mail addresses to your e-mail address.

4. Compile and run the project. Now, when some exception occurs or is raised by your code, the e-mail with exception details will be sent.

10.9.4.4 Security handler (CustomSecurityHandler class)

The security handler allows you to integrate external user databases and modify the authentication and authorization process.

Any code added to the handlers is executed after the standard authentication or authorization checks performed by the system.

The class contains handlers for the following events:

- **OnAuthentication** - triggered when a user attempts to sign in with a name and password.
- **OnResourceAuthorization** - triggered when the system checks if a user is authorized to access a module.
- **OnUIElementAuthorization** - triggered when the system checks if a UI element should be

displayed to a user.
- **OnClassNameAuthorization** - triggered when the system checks if a user is authorized to access a particular document type.
- **OnTreeNodeAuthorization** - triggered when the system checks if a user is authorized to access a document in the content tree.
- **OnFilterDataSetByPermissions** - triggered when a DataSet is filtered according to the permissions or custom personalization rules of the current user.

Example

In the following example, you will learn how to integrate external user authentication using the custom security handler.

The handler of the **OnAuthentication** event will be used for this purpose. It has the following parameters:

- **object userInfo** - an object representing the user attempting to log in. This object is returned as the result of the standard authentication check performed by the system. It is *null* if the default authentication failed.
- **string username** - a string containing the username entered during the login attempt.
- **string password** - a string containing the password entered during the login attempt.

The handler must return an object representing the user if external authentication using the entered credentials is successful, or *null* to indicate that authentication failed.

Now modify the code of the **OnAuthentication** handler according to the following:

[C#]

```
public override object OnAuthentication(object userInfo, string username, string
password)
{
    // Check if the user was authenticated by the system
    if (userInfo != null)
    {
        return userInfo;
    }

    // Sample external user credentials
    UserInfo usr = null;

    // Authenticate against the external source
    if ((username.ToLower() == "externaluser") && (password == "pass"))
    {
        // Create base user record if external authentication is successful
        usr = new UserInfo();
        usr.IsExternal = true;
        usr.UserName = username;
        usr.FullName = "externaluser fullname";
        usr.Enabled = true;
    }

    // Return the user info object
    return usr;
```

```
}
```

For simplicity, the example does not use any particular database. Instead, it only checks if the current user name and password are equal to some constants. In a real-world scenario, you would need to replace this condition with code that checks if the user name with the given password is authenticated against your external database. Also, instead of simply assigning the user to a role named *external role*, you would have to implement code that checks the external database for any roles that the authenticated user is a member of and assigns them dynamically.

Once this is done, save the changes and **Build** the **CustomEventHandler** project. The system will now be able to perform authentication according to user data from an external source.

The roles created during this external authentication will not have any permissions assigned by default, so they will not authorize the user to perform any actions. You can programmatically check if a user belongs to a role using the **CMS.CMSHelper.CurrentUserInfo.IsInRole(string roleName, string siteName)** method and implement your own security logic in the other event handlers under the **CustomSecurityHandler** class.

However, we recommend importing all external roles into the **CMS_Role** table of the website's Kentico database. Then you can configure the appropriate permissions for these roles. This way, you will be able to fully use the built-in security model of Kentico CMS together with external users.

10.9.4.5 TreeNode handler (CustomTreeNodeHandler class)

The CustomTreeNodeHandler class allows you to execute custom actions when a document (TreeNode) is created, updated or deleted. It's useful if you need to synchronize changes to external systems, generate off-line version of the document in PDF, etc.

It handles the following events:
- OnBeforeInsert
- OnAfterInsert
- OnBeforeUpdate
- OnAfterUpdate
- OnBeforeDelete
- OnAfterDelete
- OnBeforeMove
- OnAfterMove
- OnBeforeCopy
- OnAfterCopy
- OnBeforeInsertNewCultureVersion
- OnAfterInsertNewCultureVersion

Example

The following example shows how to handle the OnAfterInsert event and send the newly added news item by e-mail:

1. Open the **CustomTreeNodeHandler** class and put the following code at the beginning of the file. It adds the reference to the namespaces we will use to handle the event:

[C#]

```
using CMS.DocumentEngine;
using CMS.GlobalHelper;
using CMS.EmailEngine;
```

2. Put the following code inside the **OnAfterInsert** method.

[C#]

```
// type the document as TreeNode
TreeNode newsDoc = (TreeNode)treeNodeObj;

// handle the event only for news items
if (newsDoc.NodeClassName.ToLower() == "cms.news")
{
// get content of the inserted news item and send it by e-mail
EmailMessage email = new EmailMessage();
email.From = "admin@domain.com";
email.Recipients = "admin@domain.com";
email.Subject = ValidationHelper.GetString(newsDoc.GetValue("NewsTitle"), "");
email.Body = ValidationHelper.GetString(newsDoc.GetValue("NewsSummary"), "");
EmailSender.SendEmail(email);
}
```

3. Set the From and Recipients e-mail addresses to you e-mail address.

4. Compile and run the project. Create a new document of type News. You should receive the e-mail message with it text.

How to avoid neverending loops

If you need to call TreeNode.Update in the event handler (e.g. in the OnAfterUpdate event), you need to set **TreeProvider.UseCustomHandlers** property to false before calling the Update method.

10.9.4.6 Workflow handler

The workflow handler allows you to handle the following events:

- OnBeforeCheckOut - before document is checked out
- OnAfterCheckOut - after document is checked out
- OnBeforeCheckIn - before document is checked in
- OnAfterCheckIn - after document is checked in
- OnBeforeApprove - before document is approved
- OnAfterApprove - after document is approved
- OnBeforeReject - before document is rejected
- OnAfterReject - after document is rejected
- OnBeforePublish - before document is published
- OnAfterPublish - after document is published

When the document is published, the order of the events is following:

1. OnBeforeApprove
2. OnBeforePublish
3. OnAfterPublish
4. OnAfterApprove

Example

The following example shows how to send an e-mail with news document content when it's published:

1. Open the **CustomWorkflowHandler** class and put the following code at the beginning of the file. It adds the reference to the namespaces we will use to handle the event:

[C#]

```
using CMS.DocumentEngine;
using CMS.GlobalHelper;
using CMS.EmailEngine;
using CMS.WorkflowEngine;
```

2. Put the following code inside the **OnAfterPublish** method:

[C#]

```
// type the document as TreeNode
TreeNode newsDoc = (TreeNode)treeNodeObj;

// handle the event only for news items
if (newsDoc.NodeClassName.ToLower() == "cms.news")
{
  // get content of the inserted news item and send it by e-mail
  EmailMessage email = new EmailMessage();
  email.From = "admin@domain.com";
  email.Recipients = "admin@domain.com";
  email.Subject = ValidationHelper.GetString(newsDoc.GetValue("NewsTitle"), "");
  email.Body = ValidationHelper.GetString(newsDoc.GetValue("NewsSummary"), "");
  EmailSender.SendEmail(email);
}
```

3. Set the From and Recipients e-mail addresses to you e-mail address.

4. Compile and run the project.

5. Configure your project so that it uses workflow for news items and create and publish a news item. You should receive the content of the news item by e-mail.

Getting the workflow step name

If you need to get the name of the workflow step (for example in the OnAfterApprove event), you need to

use code like this:

[C#]

```
CMS.WorkflowEngine.WorkflowStepInfo previousStep =
(CMS.WorkflowEngine.WorkflowStepInfo) previousStepObj;
string stepCodeName = previousStep.StepName;
```

10.10 API examples

10.10.1 Overview

The **Kentico CMS API Examples** feature provides API examples for creating, getting, updating and deleting objects and for other useful actions that relate to most of Kentico CMS modules. If you have the feature installed and have become familiar with its user interface, security and the data it uses, you can choose a particular module and run its API examples.

As this chapter deals with API examples on the general level, documentation concerning actual API examples of the installed modules, together with description of these modules' database structure and closely related classes, can be found in the respective Internals and API sections; e.g. for the Media libraries module in the Developer's Guide -> Modules -> Media libraries -> Media libraries internals and API section.

Advanced API examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all methods from the module classes, please refer to the Kentico CMS API Reference.

10.10.2 Installation

1. To install API examples, please choose **Full installation** in **Step 4** of Kentico Web Installer. However, if you choose **Custom installation** in this step, API examples will not be installed.

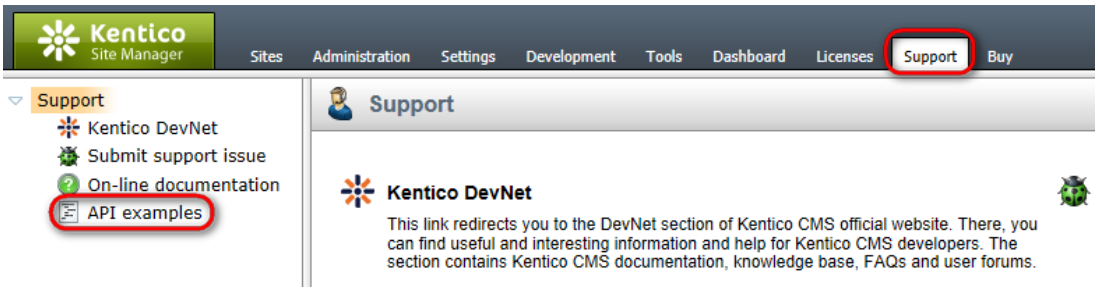
2. If you perform web project installation with the settings described above, all available API examples will be installed. API examples source code will be located in *<web project folder>/CMSAPIExamples* and the interface enabling users to view and run these examples will be located in **Site Manager -> Support -> API examples**.

> **Please note**
>
> You will need to have at least one site installed in order to be able to run API examples.

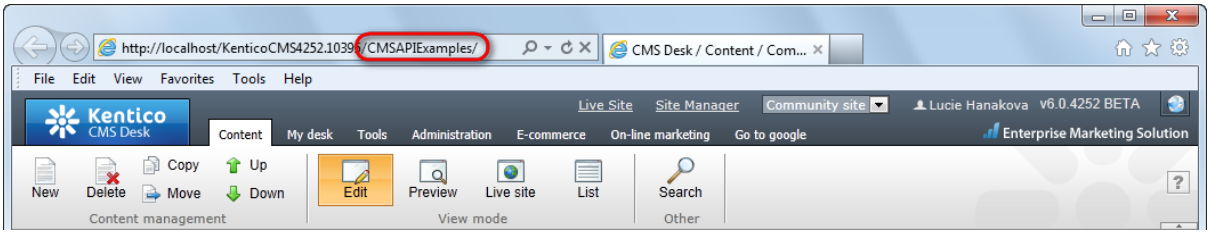
10.10.3 API examples user interface

If you have Kentico CMS API examples installed, you can access them by:

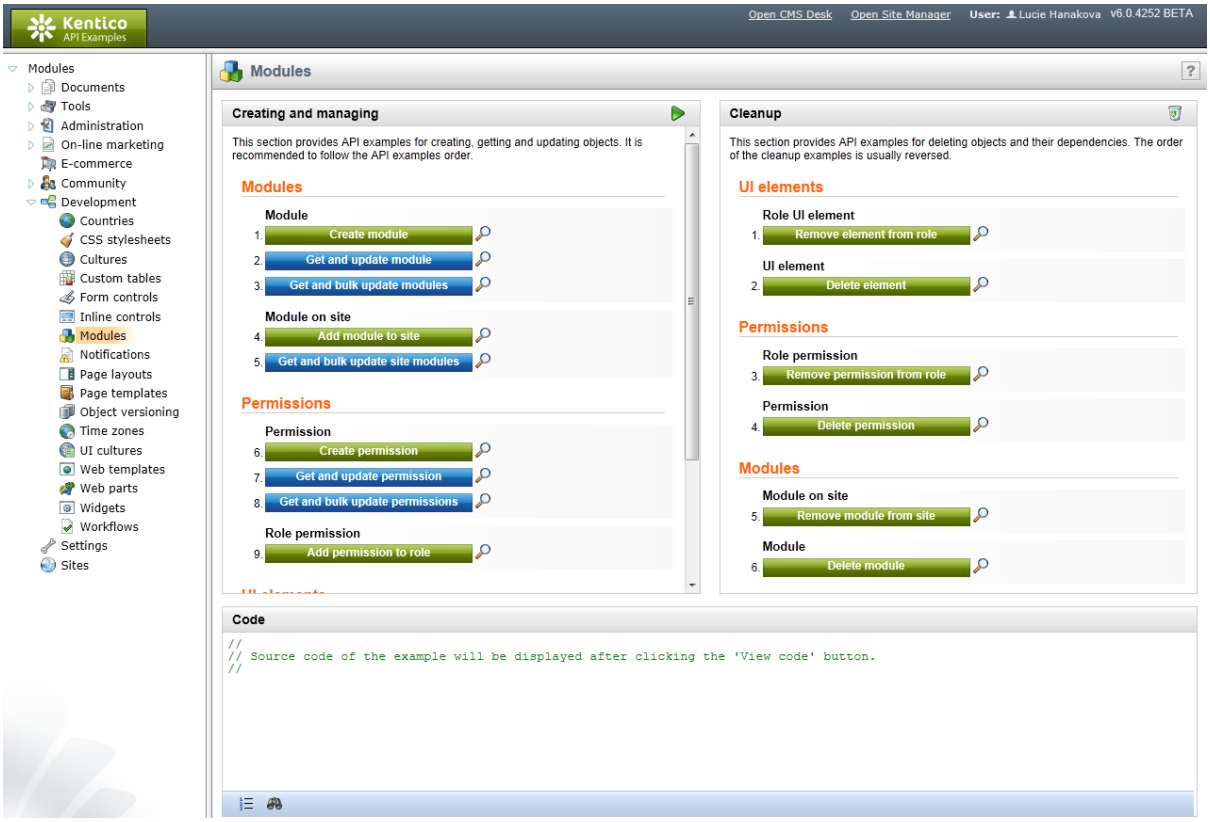
- Going to **Site Manager -> Support** and choosing **API examples** from the **Support** tree.



- Switching to the API examples directly by adding ***CMSAPIExamples*** to the application URL.



In either case, you will be redirected to the **Kentico CMS API Examples** user interface. Now choose from the **Modules** tree the module whose API examples you would like to see. This will display the interface of API examples of the particular module.



The API examples user interface consists of:

- **Creating and managing section** for creating, getting and updating objects,
- **Cleanup section** for deleting objects and their dependencies,
- **Code area** for displaying the examples code.

In this interface, you can run API examples and you can also display the code. This can be done by clicking the **View code** () icon next to the corresponding example. The code of the example will be displayed in the **Code** area. Using the following icons, you can show/hide line numbers (), toggle to the fit-to-window mode () and search through the code (). Besides, the interface enables you to copy the code to clipboard. However, editing the code directly in the **Code** area is not allowed.

Source codes of API examples can be found in ***<web project folder>/CMSAPIExamples/Code***.

Please note

The Kentico CMS API Examples user interface does not have multilingual support.

10.10.4 Data used by the API examples

All API examples work with their own objects in the *MyNew<object>* format. It means your module data in Kentico CMS should not be influenced by running API examples. However, all API examples actions directly affect the database.

Typically, the **Create** action creates an object in the *MyNew<object>* format, e.g. *MyNewUser.* The **Get and Update** action gets and updates the *MyNew<object>* object. And the **Get and bulk update** action gets and bulk updates objects which start with *MyNew<object>*. Please ensure that your own objects starting *MyNew* will not be overwritten. This can be done best by running API examples on a dedicated testing Kentico CMS installation.

If an API example needs to use one of the general objects (site, role, user, document etc.), the following ones are used:

- **For site** - current site.
- **For role** - role CMS Desk Administrator.
- **For user** - current user.
- **For document** - root document from the current site.
- **For country** - USA.
- **For state** - Alabama.
- **For currency** - USD.

If other general objects are used in API examples, their list can be found in the **Internals and API** section of the respective module.

10.10.5 Running API examples

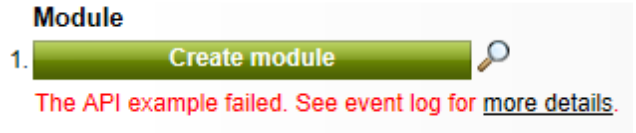
All API examples can be run by clicking the respective buttons in the module API examples user interface. In this interface you can display also the examples code. To prevent any failures, it is recommended to follow the API examples order in the **Creating and managing** section and the reversed order in the **Cleanup** section. Please note that API examples marked blue in the user interface can be skipped.

Running an API example may result in:

- **Success** - the operation ran as expected.

- **Failure** - due to **non-existence of the object**. The given object was not found in the database.

- **Failure** - due to **other reasons**. You can refer to the event log for more details.



API examples in the **Creating and managing** section and those in the **Cleanup** section can also be run all at once by clicking the **Run all** ( ) and **Cleanup all** ( ) icons, respectively.

Please note that the result of running an API example can be verified in the UI listing or directly in the corresponding table in the database.

10.10.6 Security

Only a global administrator can access and run API examples. No additional permission check for reading API examples and for creating, modifying and deleting the objects is needed.

10.11 Using transactions and connections

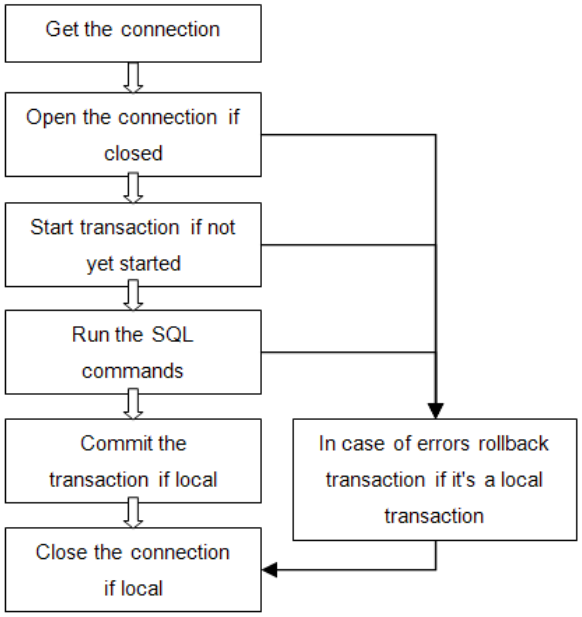
This chapter explains how to work with transactions in Kentico CMS API. Transactions are usually used to ensure database consistency.

Transactions

Kentico CMS uses SQL transactions to ensure the system database consistency when performing complex operations consisting of multiple steps. In such cases the whole operation forms a single block of SQL commands that can be rolled back when an error occurs or committed if the complete operation was successful. Transaction handling is provided by **CMS.DataEngine** library, in the **GeneralConnection** class.

Please note that you should always use a single connection for every operation within the open transaction to avoid deadlocks.

The transactions flow in Kentico CMS looks like this:



Please note: Always pass the existing connection as a parameter if you call a method that accesses the database within the transaction.

Example

Please use the following example as a template in all the methods that work with transactions:

[C#]

```
using System.Data;

using CMS.DataEngine;
using CMS.DocumentEngine;
using CMS.FileManager;
using CMS.WorkflowEngine;

/// <summary>
/// Sample method
/// </summary>
/// <param name="conn">Data connection to use</param>

public void DoSomethingInTransaction(GeneralConnection conn)
{

    // Process within transaction
    using (CMSTransactionScope tr = new CMSTransactionScope())
    {
        // --- HERE YOU CAN USE YOUR DATABASE ACCESS CODE LIKE: ----
        DataSet ds = conn.ExecuteQuery("cms.user.selectall", null, null, null);
```

```
        // --- IF YOU NEED TO USE TREEPROVIDER, ALWAYS INSTANTIATE IT WITH
EXISTING CONNECTION
        TreeProvider tree = new TreeProvider(null, conn);

        // --- ALWAYS USE METHOD OVERRIDES THAT USE CONNECTION (TREEPROVIDER)
PARAMETER ---
        AttachmentInfo ai = DocumentHelper.GetAttachment(Guid.NewGuid(), 1, tree);

        // Commit transaction, you can also use the Complete() method to do the
same
        tr.Commit();
    }

}
```

You may also work with the **TreeProvider** class instead of connection objects. In that case, use its **Connection** property the same way like the **GeneralConnection** in this example and use the same connection object in all operations inside the transaction.

Connections

You can also ensure that all operations performed by a piece of code use the same connection. This can be done similarly as with the transactions above. Please use the code extract below as a template for such code:

```
using CMS.DataEngine;

private void DoSomethingWithinTheSameConnection()
{
    using (CMSConnectionScope cs = new CMSConnectionScope())
    {
       // perform your actions
    }
}
```

The connection of the scope is put to the context, making all nested connections use the same backend connection. If you do not specify a connection using the constructor parameter, one will be created automatically. The overridden constructor gets the existing connection and also a parameter which says if the connection should be open within this scope. In case it is, it is also closed automatically:

```
new CMSConnectionScope(IDataConnection conn, bool openConnection)
```

Connections within threads

When a new thread is created, the connection is not passed to the thread to maintain thread safety. So you should create a dedicated within the thread *CMSConnectionScope* for its operations.

Nesting connection and transaction scopes

It is possible to nest connection and transaction scopes, because basically there is always a *CMSConnectionScope* around *CMSTransactionScope* so the outermost code handles the connection. So you may do following:

- Use *CMSConnectionScope* inside another *CMSConnectionScope* - this does exactly the same as not having the inner one. The outer one says "use this connection for everything within", so the inner one doesn't make sense in this context and is skipped. There is no need to do this at all (but this happens if you use some of our methods within your scope, yours (the outer one) will always take priority).
- Use *CMSTransactionScope* inside *CMSConnectionScope* - the transaction just uses the existing connection from the scope. This can be used if you do some reading and then use the same connection for writing within the transaction. A single connection will always be used.
- Use *CMSTransactionScope* inside another *CMSTransactionScope* - it acts the same way as for the connections. The outer one still needs to commit the change, so the inner one is not used (as if it didn't exist).
- Use *CMSConnectionScope* inside *CMSTransactionScope* - this is a situation where you use our reading code in your transaction code. A *CMSTransactionScope* does always have a *CMSConnectionScope* around it (either an existing one or a new one), so it is basically the same scenario as the first one, the inner connection is not used.

10.12 Customizing system objects with custom data or objects

You can extend system objects in Kentico CMS by adding your own fields (i.e. properties).

To do this, create a custom data class containing the required fields and connect it to the appropriate system object. All standard system objects implement the **IRelatedData** interface, which allows you to connect any type of object:

- Through the **RelatedData** property
- Dynamically by handling the **OnLoadRelatedData** event

If the connected object implements the **IDataContainer** interface, the data stored in the custom fields can then be accessed as part of the system object via the API (**GetValue** and **SetValue** methods) or macro expressions.

Example

The following sections demonstrate how to extend the **SiteInfo** object, which represents sites in Kentico CMS. The example adds two custom properties for sites:

- **SiteOwner** (string)
- **SiteValidUntil** (DateTime)

Defining the custom data class

1. Open your web project in Visual Studio.

2. Create a new class named **SiteRegistrationData.cs**.
   - Place the class file into the **App_Code** folder if your Kentico CMS installation is a *Web site* project.

3. Write the code of the class.
   - The class must implement the **IDataContainer** interface.
   - Define your custom properties and all required *IDataContainer* members (as shown in the code below).

[C#]

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

using CMS.SettingsProvider;

public class SiteRegistrationData : IDataContainer
{
    #region "Variables"

    private string mSiteOwner = null;
    private DateTime mSiteValidUntil = DateTime.MinValue;

    #endregion


    #region "Properties"

    /// <summary>
    /// Gets or sets the name of the site owner.
    /// </summary>
    public string SiteOwner
    {
        get
        {
            return mSiteOwner;
        }
        set
        {
            mSiteOwner = value;
        }
    }

    /// <summary>
    /// Gets or sets the date until which the site is valid.
    /// </summary>
    public DateTime SiteValidUntil
    {
        get
        {
            return mSiteValidUntil;
        }
        set
```

```
        {
            mSiteValidUntil = value;
        }
    }

    #endregion


    #region "IDataContainer members"

    /// <summary>
    /// Gets a list of column names.
    /// </summary>
    public List<string> ColumnNames
    {
        get
        {
            return new List<string>() { "SiteOwner", "SiteValidUntil" };
        }
    }

    /// <summary>
    /// Returns true if the class contains the specified column.
    /// </summary>
    /// <param name="columnName"></param>
    public bool ContainsColumn(string columnName)
    {
        switch (columnName.ToLower())
        {
            case "siteowner":
            case "sitevaliduntil":
                return true;

            default:
                return false;
        }
    }

    /// <summary>
    /// Gets the value of the specified column.
    /// </summary>
    /// <param name="columnName">Column name</param>
    public object GetValue(string columnName)
    {
        switch (columnName.ToLower())
        {
            case "siteowner":
                return mSiteOwner;

            case "sitevaliduntil":
                return mSiteValidUntil;

            default:
                return null;
        }
    }

    /// <summary>
```

```
    /// Sets the value of the specified column.
    /// </summary>
    /// <param name="columnName">Column name</param>
    /// <param name="value">New value</param>
    public bool SetValue(string columnName, object value)
    {
        switch (columnName.ToLower())
        {
            case "siteowner":
                mSiteOwner = (string)value;
                return true;

            case "sitevaliduntil":
                mSiteValidUntil = (DateTime)value;
                return true;

            default:
                return false;
        }
    }

    /// <summary>
    /// Returns a boolean value indicating whether the class contains the
specified column.
    /// Passes on the specified column's value through the second parameter.
    /// </summary>
    /// <param name="columnName">Column name</param>
    /// <param name="value">Return value</param>
    public bool TryGetValue(string columnName, out object value)
    {
        switch (columnName.ToLower())
        {
            case "siteowner":
                value = mSiteOwner;
                return true;

            case "sitevaliduntil":
                value = mSiteValidUntil;
                return true;

            default:
                value = null;
                return false;
        }
    }

    /// <summary>
    /// Gets or sets the value of the column.
    /// </summary>
    /// <param name="columnName">Column name</param>
    public object this[string columnName]
    {
        get
        {
            return GetValue(columnName);
        }
        set
        {
```

```
            SetValue(columnName, value);
        }
    }

    #endregion

}
```

Connecting the custom data class to the system object

You need to bind the custom *SiteRegistrationData* class to the *SiteInfo* object. The example uses the **OnLoadRelatedData** event, which you can handle for specific object types, including *SiteInfo* objects. This type of binding is dynamic, which means that the system loads the data only when it is requested.

1. Create a new class file in the **App_Code** folder (or **Old_App_Code** if your project was installed as a web application).

2. Delete the default class declaration and instead extend the **CMSModuleLoader** partial class.

3. Create a new class inside the CMSModuleLoader that inherits from **CMSLoaderAttribute**.

4. Add the attribute defined by the internal class before the declaration of the **CMSModuleLoader** partial class.

5. Override the **Init** method inside the attribute class, and assign a handler to the **SiteInfo.TYPEINFO.OnLoadRelatedData** event.

6. Define the handler method for the **OnLoadRelatedData** event:
   - The handler must return an instance of your custom data class (with appropriate values assigned to the class's fields).

[C#]

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

using CMS.SettingsProvider;
using CMS.SiteProvider;

[SystemObjectCustomizationLoader]
public partial class CMSModuleLoader
{
    /// <summary>
    /// Custom attribute class for the CMSModuleLoader.
    /// </summary>
    private class SystemObjectCustomizationLoaderAttribute : CMSLoaderAttribute
    {
        /// <summary>
        /// Called automatically when the application starts.
        /// </summary>
```

```
        public override void Init()
        {
            // Assigns a handler to the OnLoadRelatedData event of the SiteInfo
object type
            SiteInfo.TYPEINFO.OnLoadRelatedData += new
TypeInfo.ObjectLoadRelatedDataEventHandler(SiteInfo_OnLoadRelatedData);
        }

        /// <summary>
        /// Handler method for the OnLoadRelatedData event of the SiteInfo class.
        /// Gets the related data from the custom storage class (must implement
the IDataContainer interface).
        /// </summary>
        static object SiteInfo_OnLoadRelatedData(BaseInfo infoObj)
        {
            SiteRegistrationData siteData = new SiteRegistrationData();

            siteData.SiteOwner = "John Smith";
            siteData.SiteValidUntil = DateTime.Now.AddDays(1);

            return siteData;
        }
    }
}
```

Result

You can now work with the custom fields of site objects using the API.

For example, open your website in CMS Desk and add the following code into the ASCX layout of one of your website's pages:

```
<asp:Label runat="server" id="lblSiteInfo" />
<script runat="server">
  protected void Page_PreRender(object sender, EventArgs e)
  {
    CMS.SiteProvider.SiteInfo currentSite = CMSContext.CurrentSite;

    this.lblSiteInfo.Text = String.Format("Site '{0}' is valid until {1} and owned
by {2}.", currentSite.DisplayName, currentSite.GetValue("SiteValidUntil"),
currentSite.GetValue("SiteOwner"));
  }
</script>
```

The **GetValue** method allows you to retrieve the data stored in the custom properties, just like with native fields of the *SiteInfo* object.

The label control displays information on the page in the following format:

Site 'Corporate Site' is valid until 1/15/2013 1:00:00 PM and owned by John Smith.

Accessing properties through macros:

You can also load the values of custom properties using macro expressions. For example:

1. Add the **Static text** web part to the page (via the **Design** tab).
2. Copy the following text into the web part's **Text** property.

```
<ul>
  <li>Site: {% CurrentSite.SiteDisplayName %}</li>
  <li>Valid until: {% CurrentSite.SiteValidUntil %}</li>
  <li>Owned by: {% CurrentSite.SiteOwner %}</li>
</ul>
```

The web part displays a list of information about the current website, including the values of the custom fields.

10.13 Using API and CMS Controls outside the CMS project

This topic describes how to configure external ASP.NET applications so that they support the Kentico CMS API and Controls. Using Kentico CMS functionality outside of the CMS web project allows you to create scenarios such as:

- Separate applications integrated with your main Kentico CMS website
- Custom websites or applications that use Kentico CMS as a content repository

Start Visual Studio and open your external application's project.

Connecting to the Kentico CMS database

To be able to access the Kentico CMS database, you need to specify the appropriate connection string in your application's **web config** file.

Add the connection string into the *configuration/connectionStrings* section using an <*add*> element named **CMSConnectionString**.

```
<configuration>
  <connectionStrings>

    ...

    <add name="CMSConnectionString" connectionString="Persist Security Info=False;
database=KenticoCMS;server=myserver;user id=username;password=mypassword;Current
Language=English;Connection Timeout=120;" />

  </connectionStrings>
```

Note: It is recommended to copy the exact connection string from the web.config file of the Kentico CMS web project.

Adding references to Kentico CMS API libraries

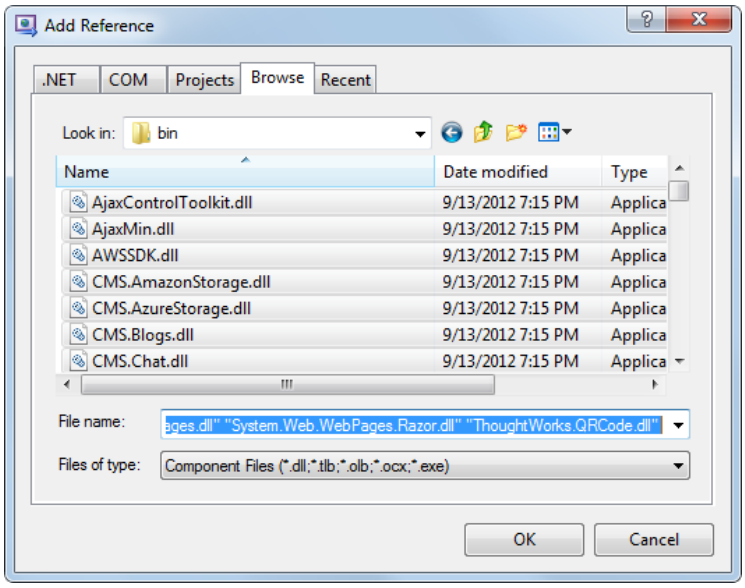
Before you can use Kentico CMS functionality in your application, you need to add references to the

libraries containing the required code:

1. Right-click your application's web project root in the **Solution Explorer** and select **Add Reference**.

2. Open the **Browse** tab and navigate to the **bin** folder of the Kentico CMS web project.

3. Select all dll files in the folder and click **OK**.



Initializing Kentico CMS

You need to initialize Kentico CMS before making calls to its API from an external project.

Execute the **CMSContext.Init** method in the code of your application:

[C#]

```
CMS.CMSHelper.CMSContext.Init();
```

You can call this method at any point in your application's life cycle, but it must occur before you use any other Kentico CMS API.

Enabling ASCX transformations in external applications

Registering the Virtual path provider:

Kentico CMS uses a virtual path provider to retrieve the code of ASCX transformations. If you wish to use ASCX transformations in your external project, you need to register the virtual path provider when the application starts.

1. If necessary, create the **Global.asax** file for your application.

2. Execute the following line from the **Application_Start** method in the *Global.asax* file.

[C#]

```
void Application_Start(object sender, EventArgs e)
{
    CMS.VirtualPathHelper.VirtualPathHelper.RegisterVirtualPathProvider();
}
```

Running without the virtual path provider

If you cannot use the virtual path provider in your environment (e.g. when using medium trust and .NET 3.5), you need to:

1. Save your virtual objects to the local disk using the **Site Manager -> Administration -> System -> Virtual objects** interface.
2. Copy the **~/CMSVirtualFiles** folder from the Kentico CMS project the to the root of your own project.

See also: Deployment and source control

Adding the CMSTransformation class:

The *CMSTransformation* partial class allows you to write your own methods for use in ASCX transformations. The system checks the CMSTransformation class for method definitions whenever listing controls load transformations and an error occurs if the class is not present in the given project.

If you plan to use ASCX transformations, you must add the class to your project:

1. Create a new class file in your project named **CMSTransformation.cs**.
   - In *web site* projects, you need to add the file into the **App_Code** folder.
   - In *web application* projects, you can create the class in any location.

2. Add the following code into the class:

[C#]

```
namespace CMS.Controls
{
    /// <summary>
    /// Base class for transformation methods
    /// </summary>
    public partial class CMSTransformation : CMSAbstractTransformation
    {
    }
}
```

You can leave the class empty if you do not need to register any custom methods.

Example: Displaying content from the Kentico CMS database

Once you have completed the configuration described in the sections above, you can use the Kentico CMS API and Controls in your application. A typical example is loading and displaying data from Kentico CMS.

Using the Kentico CMS API

The Kentico CMS API allows you to perform any action available in the standard administration interface. You can also use the API to load or modify records in the database.

The following example shows how to retrieve document content from the Kentico CMS database and display it using a standard ASP.NET Repeater control.

1. Create a new page (web form) in your external web project using Visual Studio.

2. Add the standard ASP.NET **Repeater** control onto the page.

3. Insert the following item template markup into the *<asp:Repeater>* control element:

```
<asp:Repeater ID="Repeater1" runat="server">

    <ItemTemplate>
        <strong>
        <%# Eval("NewsTitle") %>
        </strong>
        (<%# ((DateTime) Eval("NewsReleaseDate")).ToString("d") %>)
        <br />
        <i><%# Eval("NewsSummary") %></i>
        <br />
    </ItemTemplate>

</asp:Repeater>
```

4. Switch to the page's code behind and add the following references to the beginning of the code:

[C#]

```
using CMS.DocumentEngine;
using System.Data;
```

5. Add the following code into the page's **Page_Load** method:

[C#]

```
protected void Page_Load(object sender, EventArgs e)
{
```

```
    // Creates a data set containing all released news documents from the
Corporate Site
    TreeProvider tp = new TreeProvider();
    DataSet ds = tp.SelectNodes("CorporateSite", "/news/%", "en-us", true,
"cms.news", " NewsReleaseDate <= GetDate() ", " NewsReleaseDate DESC ", -1, true);

    // Binds the news data to the Repeater control
    Repeater1.DataSource = ds;
    Repeater1.DataBind();
}
```

Note

When performing other types of document operations (editing, deleting etc.), you need to initialize the tree provider within the context of a specific user:

[C#]

```
using CMS.SiteProvider;

...

// Gets an Info object representing the administrator user
UserInfo user = UserInfoProvider.GetUserInfo("administrator");

// Creates a tree provider instance using administrator context
TreeProvider tp = new TreeProvider(user);
```

If you run the website and open the new page, the Repeater control displays a list of news article summaries.

Community Website Section (6/28/2011)
As a result of our continuous effort to improve our services, we have recently introduced the Community section of our website. It is a place where you can both receive interesting information about the company from various communication channels and express your opinions and thoughts yourselves.
Company Growth Exceeds Expectations (6/17/2011)
Our company growth has reached astonishing 256% in the last financial year. It is not only thanks to the excellent and devoted work of our employees, but mainly thanks to you, our faithful customers. Therefore, we would like to thank you for your loyalty and state a promise that we will keep to the high standard of products and services we currently provide.
Apple iPad 2 In Stock (6/9/2011)
Today, we have good news for all fans of the awesome Apple iPad. We are glad to announce that its new version, Apple iPad 2, is available in our web shop. Furthermore, we keep our reasonable pricing policy, providing the lowest price currently available on the Web.
New Consulting Services (6/5/2011)
We are proud to announce that the range of services we provide was extended by web development consulting. The most experienced and skilled employees from our web development department have been promoted to consultants and are here to help you with your web development projects.

ASP.NET Repeater control displaying news data loaded by the Kentico CMS API

Using Kentico CMS controls

This example demonstrates how to display Kentico CMS documents via the built-in CMSRepeater control.

Note: For quick access to Kentico CMS controls, add the controls to your Visual Studio Toolbox.

1. Create a new page (web form) in your application's project using Visual Studio.

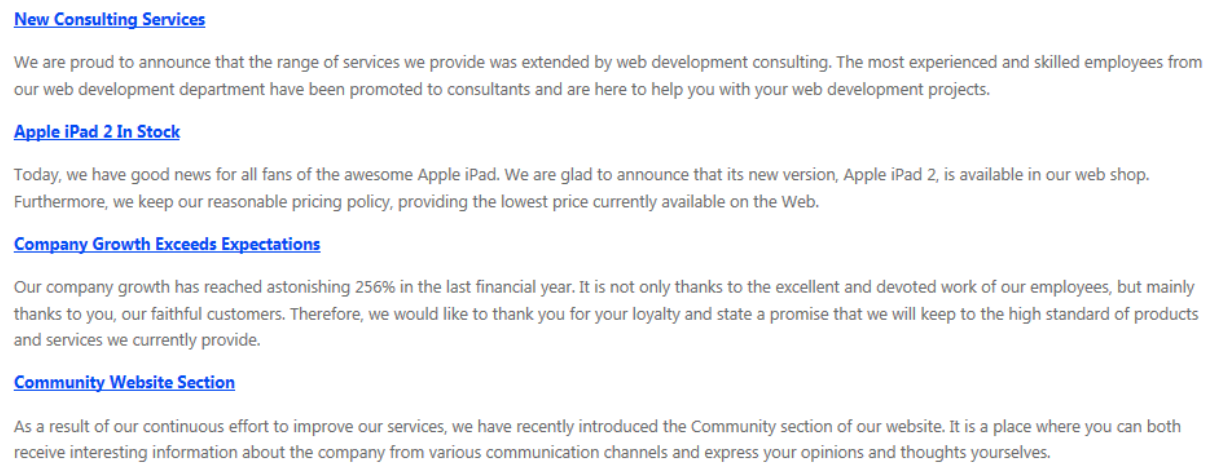
2. Add the **CMSRepeater** control onto the page.
   - You can either drag the control from the toolbox or manually register the *CMS.Controls* assembly on the page and then use the Visual Studio IntelliSense.

3. Set the following properties for the *CMSRepeater* control:

- **ClassNames**: cms.news
- **SiteName**: CorporateSite (or any other site code name)
- **TransformationName**: cms.news.preview
- **SelectedItemTransformationName**: cms.news.default

```
<%@ Register Assembly="CMS.Controls" Namespace="CMS.Controls" TagPrefix="cms" %>

...

<cms:CMSRepeater ID="CMSRepeater1" runat="server" ClassNames="cms.news"
SiteName="CorporateSite" TransformationName="cms.news.preview"
SelectedItemTransformationName="cms.news.default" />
```

If you run your project and navigate to the new page, you can see the list of news documents.



News documents displayed by the CMSRepeater

Handling document links in external applications:

The default URLs of Kentico CMS documents are based on the structure of the content tree and the settings of individual documents. These URLs will not work correctly outside of the Kentico CMS application. For example, the links in the news item headings in the example above lead to pages that most likely do not exist in your custom web project.

The following steps extend the previous example so that the news detail links work in external applications. The CMSRepeater control renders the links using the *cms.news.preview* transformation, which you can modify.

1. Open Kentico CMS and go to **Site Manager -> Development -> Document types**.

2. Edit () the **News** document type.

3. Select the **Transformations** tab and edit the **preview** transformation.

4. Change the transformation code that defines the link URL from:

```
<a href="<%# GetDocumentUrl() %>">
```

to:

```
<a href="?aliasPath=<%# Eval("NodeAliasPath") %>">
```

5. Click  **Save**.

The original *GetDocumentUrl()* transformation method generates document URLs in the default Kentico CMS format. The modified link URL leads to the same page containing the listing control, but with an **aliasPath** parameter in the query string of the URL. The parameter contains the alias path of the corresponding news document.

Kentico CMS listing controls automatically process the *aliasPath* URL parameter and insert its value into the **Path** property. In this case, the link URLs ensure that the CMSRepeater control only loads one specific news document. When the source data only contains one document, the control uses the transformation specified by the **SelectedItemTransformationName** property (*cms.news.default* in this case) to display the details of the given document.

Note: Controls cache transformations, so you need to restart your application to apply the changes.

Implementing custom document selection for listing controls:

You can alternatively use your own custom logic to dynamically set the path of listing controls according to the URL or other variables.

For example, the following steps demonstrate how to ensure document selection based on a custom URL parameter:

1. Edit the **cms.news. preview** transformation again and change the name of the URL parameter in the link code to *customParameter*.

```
<a href="?customParameter=<%# Eval("NodeAliasPath") %>">
```

2. Open the markup of the page in Visual Studio and set the **DataBindByDefault** property of the CMSRepeater control to *false*.

3. In the page's code behind, add the following code into the **Page_Load** method:

[C#]

```
protected void Page_Load(object sender, EventArgs e)
{
    // Checks whether the URL contains the 'customParameter' query string
parameter
    if (Request.QueryString["customParameter"] != null)
    {
       // Limits the path of the CMSRepeater to the document specified by the URL
parameter
       CMSRepeater1.Path = Request.QueryString["customParameter"];
    }
    else
    {
        // Loads documents from the entire website if the URL doesn't contain the
'customParameter' parameter
        CMSRepeater1.Path = "/%";
    }

    // Loads and binds the data of the CMSRepeater control (with regard to the
dynamically set Path)
    CMSRepeater1.ReloadData(true);
}
```

Setting CMS control properties dynamically

By default, Kentico CMS listing controls load content during the *Init* stage of the control life cycle. If you need to programmatically assign control properties that affect the content, use one of the following approaches:

1. Set the **DelayedLoading** property to *true* in the control's markup.
   - This moves the automatic data binding to the control's *Load* event.
2. Assign the required properties during the page's *Load* event (in the **Page_Load** handler) or sooner.

OR

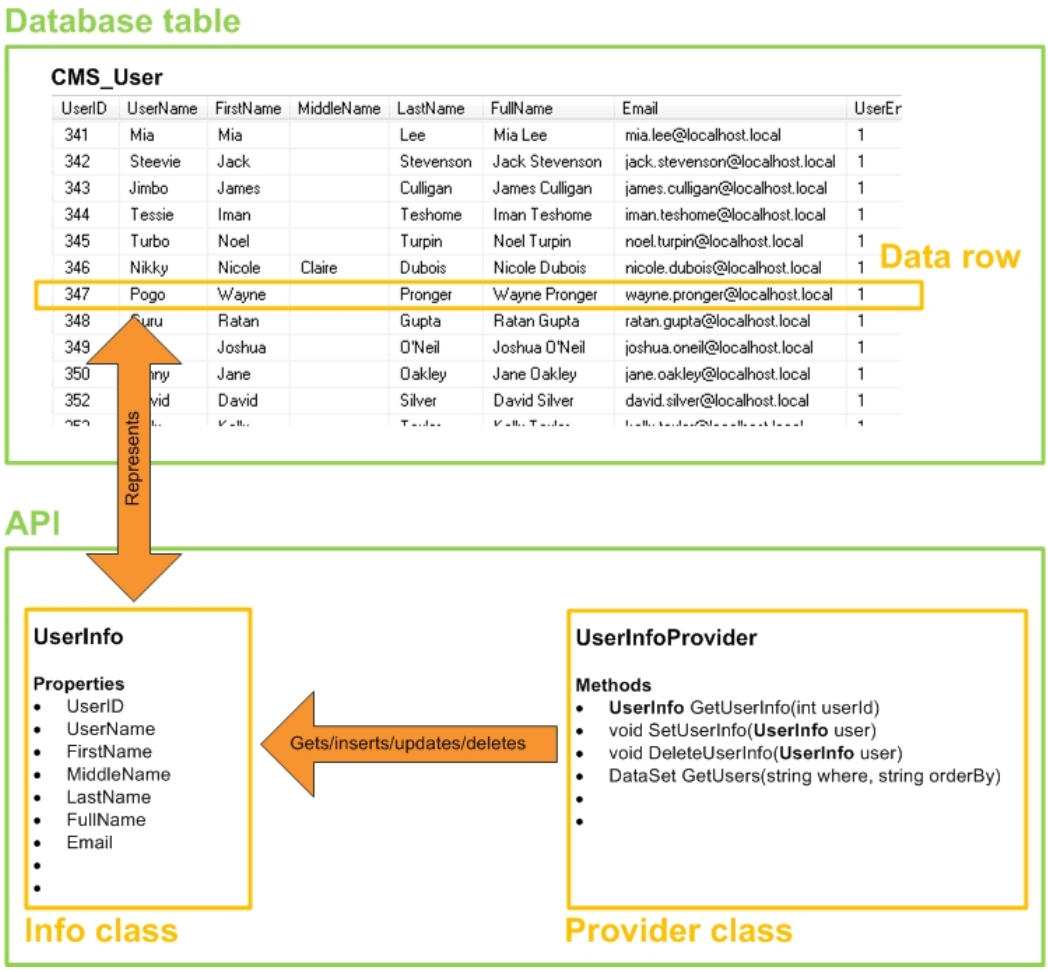
1. Set the **DataBindByDefault** property to *false* in the control's markup.
   - This completely disables automatic data binding for the control.
2. Assign the required properties at any appropriate point in the page life cycle.
3. Explicitly call the control's **ReloadData(true)** method.

The control then loads the content based on the dynamically assigned properties.

The page now uses the *customParameter* query string parameter to select specific news documents. The custom parameter works the same way as the default *aliasPath* parameter.

10.14 Database table API

Kentico CMS uses database tables to store data. There are two API classes for each database table to manage its data - Info class and Provider class. See the example for the *CMS_User* table below.



Info class

Each info class is related to a database table. An info class reflects one particular entry (line) in the table. This means an info class is a data container holding data of one table entry. The properties of an info class correspond to the columns of the related table.

Provider class

A provider class is used to manage data from the related table. It usually uses the related info object when manipulating with the data. A typical provider class contains methods used for getting, inserting, updating and deleting table data, together with various other methods used for further table data manipulation.

Part XI

Appendix A - Path expressions

11 Appendix A - Path expressions

Path expressions allow you to select a set of documents from the content tree of Kentico CMS websites. The expressions are based on the alias path values of documents. You can use either the exact paths of individual documents or expressions containing special characters, which specify multiple documents or relative paths.

For example, path expressions are commonly used in the **Path** property of web parts and controls that provide navigation elements or display document data.

Using wildcard characters % and _

You can use **% as a wildcard character for any number of characters**, which allows you to select all documents under the specified site section.

Examples:

/ - only the root document
/% - all documents

/Products - only the *Products* document
/Products/% - all child documents under the *Products* document

You can also use **_ as a wildcard representing single character**.

Examples:

/Product_ - selects documents */ProductA*, */Product1*, etc.

Leaving the Path value empty

In many cases, you can leave the Path value empty.

For **navigation** controls/web parts (such as *CMSMenu*/*Drop-down menu*), an empty value sets the path to all documents: /%

Listing controls/web parts with document data sources (such as *CMSRepeater*/*Repeater*) evaluate empty path values dynamically based on the document type of the current document:

- If the type matches the value of the web part or control's **ClassNames** (**Document types**) property, the empty path automatically selects the current document.
- If the document type does not match, the path is set to all child documents: <*current alias path*>/%

Getting parts of the path

You can use special expressions to **extract parts of the current document's path**.

- {0} - the alias of the document on the first level of the current path
- {1} - the document alias of the current path's second level
- ...

For example, if the document path is: ***/Company/Careers/USA-Branch/Development/QA-Engineer***

- **{0}** = Company
- **{1}** = Careers
- **{3}** = Development

Path examples:

/{0}/{1}/% - all documents under the second level of the current path
/{0}/{1}/Details - document *Details* under the second level of the current path

Note

If you attempt to extract a level that does not exist in the current document's path:

- The expression returns an empty value.
- If there is a slash (/) after the expression, the system removes it when resolving the overall path.

Using relative paths

You can use relative paths expressions to specify **sub-documents or parent documents**:

Examples:

. - current path
.. - parent document of the current path
./Product - document *Product* under the current path
../Product - document *Product* under the parent document of the current path
./% - all documents under the current path
../% - all documents under the parent document of the current path

Part XII

Appendix B - Web.config parameters

12 Appendix B - Web.config parameters

The system settings include appSettings keys and other settings, such as a connection string placed in appropriate sections of the web.config file. AppSettings keys are stored in the **/configuration/appSettings** section.

The following setting categories are available:

General settings

Key	Description	Sample Value

CMSProgrammingLanguage	Indicates the programming language used in transformations and in custom code added to web parts.
	
	The default value is C#.
		`<add key = "CMSProgrammingLanguage" value="C#" />`
		
		or
		
		`<add key = "CMSProgrammingLanguage" value="VB" />`

CMSTrialKey	Contains a temporary trial license key. You can remove this value after	

	installation. The default value is "" (empty string).	
CMSHashStringSalt	Sets the salt value that the system uses in hash functions, for example when creating macro signatures. The installer automatically adds the key for new instances of Kentico, with a random GUID as the value. **Note**: Changing the hash salt on a website that already has defined content may prevent macros from resolving correctly, or break dialog links and images on your website. If you encounter such problems, you need to re-save the given content or globally re-sign macros (the system then creates the hashes using the new salt).	`<add key="CMSHashStringSalt" value="eb2d6fac-8b9e-427c-b98b-3c562dffbb35" />`
CMSUseCustomHandlers	Indicates if custom handlers should be executed to process system events. See the Global events and their handling chapter for more details. The default value is *false*.	`<add key ="CMSUseCustomHandlers" value="true" />`
CMSUseVirtualPathProvider	Indicates if the virtual path provider should be used (true by default). Before you switch off the virtual path provider, please read the Pre-compilation (Publish function) chapter. The default value is *true*.	`<add key = "CMSUseVirtualPathProvider" value="false" />`
CMSShowWebPartCodeTab	Indicates if the Code tab should be displayed in the web part properties dialog in CMS Desk. This parameter can be used for the purposes of backwards compatibility. Otherwise, using the Code tab is now obsolete. The default value is *false*.	`<add key ="CMSShowWebPartCodeTab" value="true" />`
CMSShowWebPartBindingTab	Indicates if the Binding tab should be displayed in the web part properties dialog in CMS Desk. This parameter can be used for the purposes of backwards compatibility. Otherwise, using the Binding tab is now obsolete. The default value is *false*.	`<add key = "CMSShowWebPartBindingTab" value="true"/>`
CMSRenderGeneratorName	Indicates if the 'generator' meta tag stating that the page was generated by Kentico CMS is generated in the header	`<add key =`

	of each page. The default value is *false*.	"CMSRenderGeneratorName" value="true"/>
CMSClearFieldEditor	Determines field editor behavior when creating new fields. If true, new fields will have empty values of attributes. If false, new fields will have pre-defined values, the same as the previously selected field. The default value is *true*.	<add key ="CMSClearFieldEditor" value="true"/>
CMSShowTemplateASPXTab	Indicates if ASPX code tab is displayed when editing a page template. Using this tab, ASPX code of a page template created using the portal engine can be exported. The default value is *false*.	<add key = "CMSShowTemplateASPXTab" value="true"/>
CMSDatabaseCulture	Specifies the default culture of the system's database. The default value is *en-us*.	<add key="CMSDatabaseCulture" value="en-us"/>
CMSDisposeConnectionAfterClose	If true, database connection is automatically disposed (allocated resources released) when a database connection is closed. The default value is *false*.	<add key = "CMSDisposeConnectionAfterClose" value="true"/>
CMSUseSessionCookies	Indicates if session cookies are used or not. The default value is *true*.	<add key ="CMSUseSessionCookies" value="true"/>
CMSImportWindowsRoles	When Windows authentication is used and this key set to true, roles available in the Active Directory will be imported into the system. The default value is *true*.	<add key ="CMSImportWindowsRoles" value="true"/>
CMSFileScriptTimeout	The maximum number of seconds a script can run before the server terminates it. The default value is *7200*.	<add key ="CMSFileScriptTimeout" value="7200"/>
CMSUseExtensionOnPostback	When using friendly URL extensions, postback doesn't work in some cases. If you enable this setting, *.aspx* extension is attached to the URL in the form tag, which prevents the postback problems. The default value is *true*.	<add key = "CMSUseExtensionOnPostback" value="true"/>

CMSUseSQLResourceManager	If true, SQL Resource manager is used to retrieve strings used in the user interface. The default value is *true*.	`<add key = "CMSUseSQLResourceManager" value="true"/>`
CMSAllowCheckIOPermissions	If true, write permissions on the site folder are checked when necessary and produce an error message when they are insufficient. The default value is *true*.	`<add key = "CMSAllowCheckIOPermissions" value="true"/>`
CMSWorkflowSendEmailToModerator	If true, workflow notification e-mails will be sent to the user who is performing the current workflow step along with other users involved in the workflow. The default value is *false*.	`<add key = "CMSWorkflowSendEmailToModerator" value="true"/>`
CMSControlElement	If present in the web.config, the tag entered in the value will be used instead of the SPAN tag when generating pages. The default value is *span*.	`<add key="CMSControlElement" value="div"/>`
CMSUseParsedSelfClose	Indicates if parsed self closing tags operations (faster) are used instead of standard self close filter. The default value is *true*.	`<add key ="CMSUseParsedSelfClose" value="true"/>`
CMSGetFileEndRequest	If true, *ApplicationInstance.CompleteRequest()* is used instead of *Response.End()* in the *CompleteRequest* method. The default value is *true*.	`<add key ="CMSGetFileEndRequest" value="true"/>`
CMSDefaultUserID	Specifies default user ID. The default value is *0*.	`<add key="CMSDefaultUserID" value="53"/>`
ImportFilesDiskPath	Specifies path to attachments that should be attached to documents imported via the SQL Import windows application. The default value is *""* (empty string).	`<add key ="ImportFilesDiskPath" value="C:\Temp"/>`
CMSDeleteTemporaryAttachmentsOlderThan	Specifies how old should be attachments deleted by the 'Delete old temporary attachments' scheduled task. The value is entered in hours. Attachments older than the entered value will be deleted when the scheduled task is executed. The default value is *24*.	`<add key = "CMSDeleteTemporaryAttachmentsOlderThan" value="12"/>`

CMSAllowGZip	Enables Gzip compression of output HTML code. The default value is *false*.	`<add key="CMSAllowGZip" value="true" />`
CMSDisableAdministrationInterface	Disables the administration interface. The 'Access denied' screen is displayed on each attempt to access CMS Desk or Site Manager. The default value is *false*.	`<add key = "CMSDisableAdministrationInterface" value="true"/>`
CMSMaxNodeAliasLength	Determines the maximum allowed length of document aliases. The default value is 50 characters and larger names are trimmed. You can increase the allowed length if you need to enter very long aliases (URLs). If you increase the key's value, you also need to: 1. Edit the **~\CMSModules\Content\CMSDesk\Properties\Alias_List.aspx** file and increase the **MaxLength** property of the **txtAlias** control. 2. Delete the following views from your database: • View_CMS_Tree_Joined_Linked • View_CMS_Tree_Joined_Regular 3. Increase the **nvarchar** size for the **NodeAlias** column in the **CMS_Tree** database table. 4. Increase the **nvarchar** size for the **NodeAlias** column in the code of the following stored procedures: • Proc_CMS_Tree_InitNodeOrders • Proc_CMS_Tree_MoveNodeAlphabetical • Proc_CMS_Tree_OrderAlphaAsc • Proc_CMS_Tree_OrderAlphaDesc 5. Recreate the deleted views. Run the following scripts found in the **~\App_Data\Install\SQL\Objects** folder: • View_CMS_Tree_Joined_Linked.sql • View_CMS_Tree_Joined_Regular.sql **Note**: The maximum allowed length for the **Node alias path** is 450 characters, so the system may trim the alias path for	`<add key ="CMSMaxNodeAliasLength" value="120" />`

	documents deep in the content tree regardless of the allowed length.	
CMSMaxNodeNameLength	Sets the maximum possible length for the names of documents in the content tree. The default value is *100* characters and larger names are trimmed. This key can be used to increase the allowed length, which can be useful if you wish to enter very long document names. If you use the key, you also need to set the same size for the following columns in the database: • **NodeName** (*CMS_Tree* table) • **DocumentName** (*CMS_Document* table) • **PageTemplateDisplayName** (*CMS_PageTemplate* table) • **VersionDocumentName** (*CMS_VersionHistory* table)	`<add key ="CMSMaxNodeNameLength" value="150" />`
CMSWebAnalyticsSlidingIPExpiration	This key is used when the *Site Manager -> Settings -> On-line marketing -> Web Analytics -> Remember visitors by IP (minutes)* key has a value higher than 0. If *enabled* (by default), users who are active on the site but have disabled cookies are not logged as new visitors after the set time. If *disabled*, even an active user with disabled cookies is logged as a new site visitor after the set time.	`<add key = "CMSWebAnalyticsSlidingIPExpiration" value="false" />`
CMSAuthenticationType	This key overrides the values returned by the *IsFormsAuthentication* and *IsWindowsAuhentication* methods in *CMS.GlobalHelper.RequestHelper*. You will typically use this if you are using a custom authentication provider whose authentication type is a non-standard one (e.g. Federated authentication) to make Kentico CMS handle it as if it was windows or forms authentication. The following values are available: • **default**: standard behavior • **forms**: *IsFormsAuthentication* always returns *true*, *IsWindowsAuthentication* always returns *false* • **windows**: *IsFormsAuthentication* always returns *false*,	`<add key ="CMSAuthenticationType" value="windows" />`

	IsWindowsAuthentication always returns *true* • **both**: *IsFormsAuthentication* always returns *true*, *IsWindowsAuthentication* always returns *true* The default value is *default*.	
CMSTemporaryFilesFolderPath	Overrides the default *~/AppData/CMSTemp* location where various temporary files are stored by the system. As a value, you can use: • **physical disk path** - e.g. *c:\myfiles\mysite* • **virtual path** - e.g. *~/UploadedFiles* • **UNC path** - e.g. *\\server\folder*	`<add key = "CMSTemporaryFilesFolderPath" value="\\server\MyCustomFolder" />`
CMSEnableInlineControls	Indicates if Inline control insertion should be possible using the **Insert inline control** button in the WYSIWYG editor toolbar. This key does not affect the resolving of inline control macro expressions. The default value is *false*.	`<add key = "CMSEnableInlineControls" value="true" />`
CMSDefaultTheme	Sets name of the folder within *~\App_Themes* containing the theme to be used as the default theme by the CMS. The default value is *Default*.	`<add key="CMSDefaultTheme" value="MyTheme" />`
CMSImagesDirectory	Sets a custom path to the *Images* folder of the theme to be used by the CMS. It can be located in a different location than within the default theme folder. The default value is *~/App_Themes/<default theme folder>/Images*.	`<add key="CMSImagesDirectory" value="~/App_Themes/MyTheme/Images" />`
CMSApplicationName	Used by Kentico CMS Windows services to identify the Kentico CMS instance. In case of IIS installation, path to the instance in IIS is used as its value. In case of Visual Studio web server installation, name of the target web project root folder is used. The value must be less than 60 characters long. It is also used to register Health monitoring performance counters and to identify respective counter categories when writing monitored values to them.	`<add key="CMSApplicationName" value="Default Web Site/CMS" />`
CMSApplicationGuid	Unique identifier of the Kentico CMS	`<add`

	instance. Used by Kentico CMS Windows services to identify the Kentico CMS instance.	`key="CMSApplicationGuid" value="fe5493b6-4ea9-42e6-92fa-0db1156b9163" />`
CMSEnableAutomaticCampaignCreate	If set to *true*, a campaign object will automatically be added to the current website if it is viewed by a user coming from an undefined campaign (typically specified through the campaign tracking URL parameter). If the key is set to *false* (this is the default state), then undefined campaigns will be ignored. Please see Modules -> Web analytics -> Campaigns for further information.	`<add key = "CMSEnableAutomaticCampaignCreate" value="true" />`
CMSMediaFileMaxVersioningSize	Sets the maximal file size of versioned media files in kiloBytes. Files in media libraries that are larger than the entered value will not have versions created on their update. The default value is *2147483647 (int.MaxValue)*.	`<add key = "CMSMediaFileMaxVersioningSize" value="1024" />`
CMSShowForgottenPassLink	Indicates if a link that allows users to recover a forgotten password should be displayed on the logon page for the administration interface (CMS Desk and Site Manager). The default value is *true*.	`<add key = "CMSShowForgottenPassLink" value="false" />`
CMSDisableMacroParameters	Disables resolving of macro parameters. If set to *true*, macro parameters will be ignored and the macro expressions will return results as if they contained no parameters. The default value is *false*.	`<add key = "CMSDisableMacroParameters" value="true" />`
CMSSelectorMaxDisplayedTotalItems	Determines the maximum number of items that can be displayed in drop-down list selectors in the administration interface. If there are more selectable objects, the list will be shortened and the *(more..)* option will be added. The value of this key can be overridden for individual UniSelector controls.	`<add key = "CMSSelectorMaxDisplayedTotalItems" value="30" />`
CMSAlwaysCacheResources	Enables/disables client caching of (minifiable) resources.	`<add key="CMSAlwaysCacheResources" value="false"/>`

	The default value is *true*. Please note that caching on the client side is not done at all if the **Client cache (minutes)** setting is disabled (set to *0*) in **Site Manager -> Settings -> System -> Performance**.	
CMSStorageProviderAssembly	Indicates the assembly name of storage provider. The default value is *CMS.CMSStorage*.	`<add key="CMSStorageProviderAssembly" value="CMS.CMSStorage"/>`
CMSProcessTrailingSlashForFile	Allows to disable processing of the trailing slash for attachment URLs. The default value is *true*.	`<add key="CMSProcessTrailingSlashForFile" value="false"/>`
CMSWebAnalyticsShowFullData	Indicates whether the amount of data displayed in the analytics reports should be reduced before being rendered. The default value is *false* (for performance reasons).	`<add key="CMSWebAnalyticsShowFullData" value="true"/>`
CMSPhysicalFilesCacheMinutes	Determines expiration time in minutes that should be set for the physical files in the client cache. The default value is *10080*.	`<add key="CMSPhysicalFilesCacheMinutes" value="10080"/>`
CMSHTMLEncodeEval	If true, the EvalXXX() methods in the CMSAbstractTransformation class encode string values. The default value is *false*.	`<add key="CMSHTMLEncodeEval" value="true"/>`
CMSInstancePath	Contains the instance path.	`<add key="CMSInstancePath" value="c:\inetpub\wwwroot\KenticoCMS\"/>`
CMSEmailValidationRegex	Allows to customize a regular expression string for email validation.	`<add key="CMSEmailValidationRegex" value="^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}$"/>`
CMSEmailTransferEncoding ***[Only available after applying hotfix 7.0.44 or newer]***	Specifies the type of transfer encoding used for e-mails sent from the CMS. Supported values are: *base64*, *quotedprintable*, *sevenbit*. The default value is *base64*.	`<add key = "CMSEmailTransferEncoding" value ="QuotedPrintable" />`
CMS55Compatibility	Ensures compatibility with CMS version	`<add`

	5.5 (includes the CMS.Compatibility namespace in transformations). The default value is *false*.	`key="CMS55Compatibility" value="true"/>`
CMSMacrosCaseSensitiveComparison	If true, string comparisons are case sensitive (if not overriden by	*(casesensitive)* macor parameter). The default value is *false*.	`<add key="CMSMacrosCaseSensitiveComparison" value="true"/>`
CategoryIDLength	Category IDs in the CategoryIDPath column/field (the CMS_Category table) will be padded with leading zeros to the width specified in the key, e.g. 00001232. The default value is *8*.	`<add key="CategoryIDLength" value="9"/>`
UIelementIDLength	UI element IDs in the ElementIDPath column/field (the CMS_UIElement table) will be padded with leading zeros to the width specified in the key, e.g. 00001232. The default value is *8*.	`<add key="UIelementIDLength" value="9"/>`
SettingsCategoryIDLength	Settings category IDs in the CategoryIDPath column/field (the CMS_SettingsCategory table) will be padded with leading zeros to the width specified in the key, e.g. 00001232. The default value is *8*.	`<add key="SettingsCategoryIDLength" value="9"/>`
CMSDeleteTemporaryUploadFilesOlderThan	Specifies how old (in hours) unfinished upload files should be deleted by the 'Delete old temporary upload files' scheduled task. The default value is *24*.	`<add key="CMSDeleteTemporaryUploadFilesOlderThan" value="12"/>`
CMSImageEditorMaxVersionsCount	Indicates how many versions of edited image should be saved in the image editor temporary folder. The default value is *999*.	`<add key="CMSImageEditorMaxVersionsCount" value="500"/>`
CMSImageExtensions	Specifies a list of file extensions which should be allowed as image files. The default value is "".	`<add key="CMSImageExtensions" value=""/>`
CMSAudioExtensions	Specifies a list of file extensions which should be allowed as audio files. The default value is "".	`<add key="CMSAudioExtensions" value=""/>`
CMSVideoExtensions	Specifies a list of file extensions which should be allowed as video files.	`<add key="CMSVideoExtensions"`

	The default value is "".	value=""/>
CMSUrlPort	Specifies the port number that will be used in certain types of URLs in the administration interface (e.g. those of stylesheet images or modal dialogs). Standard navigation URLs are not affected. If not set, the URL port is automatically taken from the current request (80 in most cases). Add this key if you wish to generate URLs with a custom port number, regardless of the request port. It may also be useful in scenarios where requests are redirected to a non-standard port number, but you wish to use the usual HTTP port in URLs.	<add key="CMSURLPort" value="80" />
CMSUseServerSideLiveIDAuthentication	Determines which authentication mode should be used during Windows Live ID authentication. Each mode generates a different authentication token for the same Live ID user. Setting the value to *false* is necessary to ensure backward compatibility for users created via Live ID authentication by versions of Kentico CMS older than 6.0 (or those users created while this key is set to *false*). The default value is *true*.	<add key = "CMSUseServerSideLiveIDAuthentication" value="false" />
CMSDefaultCookieLevel	Can be used to set the default cookie level system-wide. Available cookie levels are: *None*, *System*, *Essential*, *Editor*, *Visitor*, *All*. The default value is *All*.	<add name="CMSDefaultCookieLevel" value="System" />
CMSUseSilverlightUploader	Indicates whether the system should use the multi-file uploader written in Microsoft Silverlight.	<add key = "CMSUseSilverlightUploader" value="false" />
CMSDefaultPageClassName	Use the key to change the default value of the page menu item classname. The default value is *CMS.MenuItem*.	<add key = "CMSDefaultPageClassName" value="CMS.MenuItem" />

CMSXUACompatibleValue	Defines a value of X-UA-Compatible header meta tag. This meta tag allows you to choose the version of Internet Explorer, in which the page should be rendered. Available options are: • IE=5 • IE=EmulateIE7 • IE=7 • IE=EmulateIE8 • IE=8 • IE=EmulateIE9 • IE=9 • IE=edge Default value is *IE=9*. The meta tag is added if IE browser is used.	`<add key ="CMSXUACompatibleValue" value="IE=edge" />`
CMSUseXUACompatible	Defines if X-UA-Compatible header meta tag can be added. The default value is *true*.	`<add key ="CMSUseXUACompatible" value="false" />`

Special settings for assemblies

You can use the following keys to specify which assemblies are used:

Key	Description	Sample Value

CMSDirectoryProviderAssembly	Name of the assembly that should be used for operations in the file system. You can choose from the following options:
	
	• **CMS.DirectoryProviderDotNet** - this is a managed code library that uses System.IO methods for disk operations. It is useful for environment that allows only managed code.
	
	• **CMS.DirectoryProviderWin32** - this is an unmanaged code library that uses Win32 API for disk operations. It is useful for environment that requires Win32 API calls.
	
	The default assembly is *CMS.DirectoryProviderDotNet*.
		`<add key = "CMSDirectoryProviderAssembly" value = "CMS.DirectoryProviderDotNet" />`
		
		or
		
		`<add key = "CMSDirectoryProviderAssembly" value = "CMS.DirectoryProviderWin32" />`

CMSSearchProviderAssembly	Name of the assembly that is used for full-text search based on the SQL engine. The default assembly is *CMS.SearchProviderSQL*.	`<add key = "CMSSearchProviderAssembly" value = "CMS.SearchProviderSQL" />`
CMSDataProviderAssembly	Specifies custom data provider assembly used as the database connector for the CMS. See this topic for more details. The default assembly is *CMS.DataProviderSQL*.	`<add key = "CMSDataProviderAssembly" value = "CMS.CustomDataProvider"/>`
CMSCustomHandlersAssembly	Specifies custom event handler assembly. Read this topic for more information. The default assembly is *CMS.CustomEventHandler*.	`<add key = "CMSCustomHandlersAssembly" value = "CMS.CustomEventHandler"/>`
CMSVirtualPathProviderAssembly	Specifies the Custom virtual path provider assembly. The default assembly is *CMS.VirtualPathProvider*.	`<add key = "CMSVirtualPathProviderAssembly" value = "CMS.CustomVirtualPathProvider"/>`
CMSCustomEcommerceProviderAssembly	Specifies the custom e-commerce provider assembly. The default assembly is *CMS.EcommerceProvider*.	`<add key = "CMSCustomEcommerceProviderAssembly" value = "CMS.CustomECommerceProvider"/>`

Special settings for forbidden characters in user and role names

You can use the following keys to configure forbidden characters in user and role names:

Key	Description	Sample Value
CMSEnsureSafeUserNames	Indicates if forbidden characters in user	`<add`

	names imported from AD should be replaced. If turned off by setting the value to *false*, you may experience problems when editing users imported with forbidden characters. Therefore, it is NOT RECOMMENDED to turn it off. The default value is *true*.	`key = "CMSEnsureSafeUserNames" value="false" />`
CMSEnsureSafeRoleNames	Indicates if forbidden characters in role names imported from AD should be replaced. If turned off by setting the value to *false*, you may experience problems when editing roles imported with forbidden characters. Therefore, it is NOT RECOMMENDED to turn it off. The default value is *true*.	`<add key = "CMSEnsureSafeRoleNames" value="false" />`
CMSForbiddenUserNameCharactersReplacement	Sets the character by which forbidden characters in user names imported from AD should be replaced. If not set, value from *Site Manager -> Settings -> URLs and SEO -> Forbidden characters replacement* is used.	`<add key = "CMSForbiddenUserNameCharactersReplacement" value="-" />`
CMSForbiddenRoleNameCharactersReplacement	Sets the character by which forbidden characters in role names imported from AD should be replaced. If not set, value from *Site Manager -> Settings -> URLs and SEO -> Forbidden characters replacement* is used.	`<add key = "CMSForbiddenRoleNameCharactersReplacement" value="-" />`
CMSUserValidationRegEx	Sets custom regular expression used for user name validation (used when new accounts are created or when existing usernames are modified). The default value is "^[a-zA-Z0-9_\\-\\.@]+$". If the *CMSEnsureSafeUserNames* key is set to *false*, the following regular expression is used by default: "^[a-zA-Z0-9_\\-\\.\\\\@]+$".	`<add key = "CMSUserValidationRegEx" value="([A-Za-z0-9-]+)" />`

Special settings for forbidden character replacement in URLs

You can use the following keys to configure how forbidden URL characters should be replaced:

Key	Description	Sample Value
CMSForbiddenURLValues	The characters entered as the value of this key are forbidden in URLs (document	`<add key`

	aliases and URL paths) and will be replaced automatically by the character specified in the **Forbidden characters replacement** setting in **Site Manager -> Settings -> URLs and SEO**. By default, the following characters are forbidden: `\/:*?"<>|&%.'#[]+=„“` and the space character. If you add this key to the web.config, its value will override the default forbidden character set. This way, you can either allow some of the forbidden characters or add new ones. Please note that it is recommended to keep the default characters forbidden, since they may prevent certain types of URLs from working correctly if entered into URL paths. Also keep in mind that some characters need to be escaped in the web.config according to standard XML rules, e.g. enter *<lt;* instead of the < character.	`="CMSForbiddenURLValues" value="\/:$?&quot;&lt;&gt;|&amp;%.&apos;#[]=" />`
CMSLimitUrlReplacements	While enabled, consecutive forbidden characters in URLs will be replaced by only a single replacement character and forbidden URL characters located at the beginning or end of the path will be removed completely instead of being replaced. If set to *false*, each forbidden character is replaced individually. The default value is *true*.	`<add key="CMSLimitUrlReplacements" value="false" />`
CMSUseLimitReplacementsForUrlPath	Indicates if the functionality enabled by the **CMSLimitUrlReplacements** key should be applied to the **Document URL Path** property. The default value is *true*.	`<add key="CMSUseLimitReplacementsForUrlPath" value="false" />`

Special settings for content staging

You can use the following keys to achieve specific behavior of the Staging module:

Key	Description	Sample Value
CMSStagingAcceptAllCertificates	Causes all certificates to be accepted when performing content staging tasks	`<add key`

	through SSL (X.509). If false, only certificates generated by a certification authority will be accepted. The default value is *false*.	= "CMSStagingAcceptAllCertificates" value="true" /> >
CMSStagingUseTreeCustomHandlers	Ensures that events will be raised in *CustomTreeNodeHandler* when performing content staging synchronization. The *CMSUseCustomHandlers* key needs to be enabled too for this to work. The default value is *false*.	<add key = "CMSStagingUseTreeCustomHandlers" value="true" / >
CMSMediaFileMaxStagingSize	Sets the maximal file size of synchronized media files in kiloBytes. Files in media libraries that are larger than the entered value will not be synchronized using the Staging module. The default value is *2147483647 (int.MaxValue)*.	<add key = "CMSMediaFileMaxStagingSize" value="1024" />
CMSStagingTreatServerNamesAsInstances	Global metadata changes such as changes to document types, custom tables and system tables produce staging tasks for all staging servers of all sites. In such case, it is recommended to synchronize such changes to all servers of all sites at the same time to prevent overwriting of such metadata and losing the data by synchronizing the older tasks later. You can use this key to make sure that once the global task is synchronized, it is deleted from all other servers with the same name to prevent such possibility. The default value is *false* since staging can use multiple target instances targeted with the same names.	<add key = "CMSStagingTreatServerNamesAsInstances" value="true" />
CMSStagingServerName	Name of the staging server used for advanced bi-directional content staging. The value needs to be used as **Server code name** when defining this server as a target server on the remaining servers. The default value is "" (empty string).	<add key ="CMSStagingServerName" value="server2" />
CMSStagingServiceTimeout	Sets timeout interval for the staging service. The value should be entered in seconds. The default value is *180*.	<add key = "CMSStagingServiceTimeout" value="240" />

CMSStagingLogSynchronization	Indicates if performed staging synchronization tasks should be logged as new staging tasks (that can be subsequently transferred to other staging servers). The primary use of this key is to disable logging of these tasks globally for all sites in the system. The default value is *true*.	`<add key = "CMSStagingLogSynchronization" value="false" />`
CMSSynchronizeSharedTemplatesWithDocuments	Indicates if changes made to page templates used by multiple documents should be synchronized together with document update synchronization tasks of all documents using the template. If disabled, document update tasks will not include page template synchronization tasks. The default value is *true*.	`<add key = "CMSSynchronizeSharedTemplatesWithDocuments" value="false" />`
CMSStagingLogGlobalObjectsOnlyForAssignedSites	If true, tasks for global objects are logged only for the sites to which the respective objects are assigned. The default value is *false*.	`<add key = "CMSStagingLogGlobalObjectsOnlyForAssignedSites" value="true" />`

Special settings for the WYSIWYG editor

You can use the following keys to configure the WYSIWYG editor:

Key	Description	Sample Value
CMSWYSIWYGFixXHTML	Indicates if the WYSIWYG editor should automatically try to fix XHTML incompatibilities in the code it generates. Supported values are *true* and *false*. The default value is *true*.	`<add key="CMSWYSIWYGFixXHTML" value="true" />`
CKeditor:BasePath	Specifies the location of the WYSIWYG editor (CKEditor). By default, it is located in *~/CMSAdminControls/CKeditor*.	`<add key="CKeditor:BasePath" value="/CKeditor/"/>`
CKEditor: PersonalizeToolbarOnLiveSite	Indicates if the CK toolbar can be personalized on the live site. See this topic for more details. Supported values are *true* and *false*. The default value is *false*.	`<add key="CKEditor:PersonalizeToolbarOnLiveSite" value="true" />`

Special settings for the code editor

You can use the following keys to configure the Code editor, which is used in the interface to ensure syntax highlighting and work with code fields:

Key	Description	Sample Value
CMSEnableSyntaxHighlighting	Globally enables or disables the advanced editor and syntax highlighting support for all code fields. This can be used to turn off the editor if it is causing performance issues or other problems. The default value is *true*.	`<add key = "CMSEnableSyntaxHighlighting" value="false" />`
CMSEnableSyntaxHighlighting.<Language>	Can be used to disable the advanced editor and syntax highlighting support for fields that display code in a specific language. Replace the *<Language>* string with the name of the language that you wish to disable. The following language options are available: **Text**, **HTML**, **CSS**, **JavaScript**, **XML**, **CSharp**, **SQL**, **HTMLMixed**, **ASPNET**, **CMSSharp**. All languages are enabled by default.	`<add key = "CMSEnableSyntaxHighlighting.CSS" value="false" />`
CMSShowLineNumbers	If set to true, the panel that displays the number of every line in the editor will be displayed by default when the page is loaded. Please note that some fields in the user interface are not affected by this setting. The default value is *false*.	`<add key="CMSShowLineNumbers" value="true" />`

Special settings for E-commerce

You can use the following keys when you need to achieve specific behavior of the E-commerce module:

Key	Description	Sample Value
CMSShoppingCartExpirationPeriod	Number of days after which E-commerce shopping cart content is deleted from the database. It's used for deleting unused shopping carts of anonymous users that are stored in the database with ID stored in the browser cookie. The default value is *30*.	`<add key = "CMSShoppingCartExpirationPeriod" value="60" />`
CMSUseCurrentSKUData	Indicates if the E-commerce module should use the price from the existing order items or from the current SKU data when re-calculating the order.	`<add key ="CMSUseCurrentSKUData" value="true"/>`

	The default value is *false*.	
CMSEnableOrderItemEditing	If true, order item parameters, such as order item name and order item unit price, can be modified when editing an existing order. The default value is *false*.	`<add key = "CMSEnableOrderItemEditing" value="true"/>`
CMSUseMetaFileForProductImage	Indicates if meta files should be used for product images in E-commerce. The default value is *true*.	`<add key = "CMSUseMetaFileForProductImage" value="true"/>`
CMSUseCustomEcommerceProviders	Specifies whether to use custom E-commerce provider handlers. See this topic for more details. The default value is *false*.	`<add key = "CMSUseCustomEcommerceProviders" value="true"/>`

Special settings for Event log

You can use the following keys when you need to achieve specific behavior of the Event log:

Key	Description	Sample Value
CMSLogEvents	Indicates if logging of events in the Event log is enabled. The default value is *true*.	`<add key="CMSLogEvents" value="true"/>`
CMSLogKeepPercent	Coefficient for Event log deletion. Keeps the specified percentage of extra log items in the log with regards to the **Site Manager -> Settings -> System -> Event log size** setting. The specified percentage of the oldest events is deleted by batch when the percentage is exceeded. If 0, the exact number of records is kept in the log. The default value is *10*.	`<add key="CMSLogKeepPercent" value="10"/>`
CMSLogEventsToFile	If true, events are also logged into the *~\App_Data\logevents.log* file. The default value is *false*.	`<add key="CMSLogEventsToFile" value="true"/>`
CMSLogFieldChanges	If true and the **Site Manager -> Settings -> System -> Log metadata changes** option is enabled, details about particular object changes are included in the respective log records. The default value is *false*.	`<add key="CMSLogFieldChanges" value="true"/>`
CMSLogDocumentFieldChanges	If true and the **Site Manager -> Settings -> System -> Log metadata changes**	`<add key`

	option is enabled, details about changes of values in document fields are included in the respective log records. The default value is *false*.	= "CMSLogDocumentFieldChanges" value="true"/>
CMSLogNewsletterIssueEvents ***[Only available after applying hotfix 7.0.18 or newer]***	Indicates if the system logs newsletter issue changes into the event log (including logging of all sent issues). You can disable logging of these events if you encounter problems with performance or a cluttered event log when mailing issues to a very large number of subscribers. The default value is *true*.	<add key = "CMSLogNewsletterIssueEvents" value="false" />
CMSLogMATransitions ***[Only available after applying hotfix 7.0.18 or newer]***	Indicates if the system logs all step transitions in Marketing automation processes into the event log. You can disable logging of these events if you encounter performance issues or a cluttered event log when running automation processes for a very large number of contacts. The default value is *true*.	<add key ="CMSLogMATransitions" value="false" />
CMSDataExportTemplateFolder	Indicates the starting path for template lookup. The default value is *~/App_Data/CMSModules/DataExport*.	<add key="CMSDataExportTemplateFolder" value="~/ App_Data/CMSModules/ DataExport"/>

Special settings for export/import

You can use the following keys when you want to configure file exporting:

Key	Description	Sample Value
CMSExportExcludedFolders	Specifies which folders will be filtered from being included in Files folder of the export package. **.svn** folders are excluded by default, even without this key added. More info here.	<add key = "CMSExportExcludedFolders" value="test*;cms*" />
CMSExportExcludedFiles	Specifies which files will be filtered from being included in the Files folder of the export package. **.scc** files are excluded by default, even without this key added. More info here.	<add key = "CMSExportExcludedFiles" value="test*;cms*" />

CMSSiteUtilsFolderPath	Enables to override the default *~/CMSSiteUtils/* location where export and import packages are stored. As a value, you can use: • **physical disk path** - e.g. *c:\myfiles\mysite* • **virtual path** - e.g. *~/UploadedFiles* • **UNC path** - e.g. *\\server\folder*	`<add key = "CMSSiteUtilsFolderPath" value="\\server\MyCustomFolder" />`

Special settings for item listing

You can use the following keys when you want to configure item listing:

Key	Description	Sample Value
CMSDefaultListingFilterLimit	Determines the minimum number of items that must be included in a listing in order for a filter to be shown. If the number of listed items is lower than this value, the filter is not displayed. If it is larger, the filter is displayed. This applies to all listings (UniGrid controls) across the entire UI The default value is *25*. The value of this key can be overridden for individual UniGrid controls.	`<add key = "CMSDefaultListingFilterLimit" value="40" />`
CMSDefaultListingPageSize	Initial page size (the *Items per page* setting) of listings across the whole UI. The default value is *25*.	`<add key = "CMSDefaultListingPageSize" value="50" />`
CMSListingShowFirstLastButtons	If enabled, the first and last page link buttons will be included in the pagers of listings in the UI with a large enough number of items. If disabled, the buttons will always be hidden. If both this and the *ShowDirectPageControl* keys are disabled, only *TopN* items are loaded, while *TopN = PageSize * (currentPageIndex + CurrentPagesGroupSize)*. The default value is *true*.	`<add key = "CMSListingShowFirstLastButtons" value="false" />`
CMSListingShowDirectPageControl	If enabled, a textbox that allows the current page to be changed by directly entering a number will be included in pagers of listings in the UI with a large enough number of items. If disabled, the	`<add key = "CMSListingShowDirectPageControl"`

	control will always be hidden. If both this and the *ShowFirstLastButtons* keys are disabled, only *TopN* items are loaded, while *TopN = PageSize * (currentPageIndex + CurrentPagesGroupSize)*. The default value is *true*.	`value="false" />`

Special settings for culture URLs

You can use the following keys to set up URL behavior for the culture (language) versions of documents:

Key	Description	Sample Value
CMSUseCultureForBestPageInfoResult	Indicates whether the currently selected culture should have the highest priority when deciding which language version of a document should be displayed. The default value is *false*, which means that accessing a document through the custom URL set for one of its specific culture versions will override and change the preferred culture accordingly. If set to *true*, the currently selected culture will be reflected even when a culture-specific document URL path is used.	`<add key = "CMSUseCultureForBestPageInfoResult" value="true" />`
CMSRedirectLangToPrefix	This key indicates whether URLs with a language defined through a query string parameter should automatically be redirected to a corresponding URL with a language prefix. Supported values are *true* and *false*. The default value is *true*.	`<add key = "CMSRedirectLangToPrefix" value="false" />`

Special settings for query string parameter names

You can use the following keys to change certain query string parameter names:

Key	Description	Sample Value
CMSLanguageParameterName	Changes the name of the query string parameter used to set the culture. For example, this allows you to get *Home?sprache=de-de* instead of the default *Home?lang=de-de*.	`<add key = "CMSLanguageParameterName" value="sprache" />`

	The default value is *lang*.	
CMSAliasPathParameterName	Changes the name of the *aliaspath* query string parameter so that you get *products.aspx?ap=/products/myproduct* instead of the default *products.aspx?aliaspath=/products/myproduct*. The default value is *aliaspath*.	`<add key = "CMSAliasPathParameterName" value="ap" />`

Special settings for transaction isolation

You will use the following settings only in special cases when you encounter problems with deadlocks when updating a document:

Key	Description	Sample Value
CMSTransactionIsolationLevel	Isolation level for explicit transactions. The default value is *ReadUncommitted*.	`<add key = "CMSTransactionIsolationLevel" value="ReadCommitted" />`
CMSDefaultIsolationLevel	Isolation level for all queries, even without transactions. The default value is *ReadUncommitted*.	`<add key = "CMSDefaultIsolationLevel" value ="ReadUncommitted" />`
CMSUseDefaultIsolationLevelOnlyWithOpenTransactions	If true, the default isolation level is used only when some transaction is already running. The default value is *true*.	`<add key = "CMSUseDefaultIsolationLevelOnlyWithOpenTransactions" value="true" />`
CMSAllowSimultaneousTransactions	If false, there can be only one transaction running at the same time. The default value is *true*.	`<add key = "CMSAllowSimultaneousTransactions" value="true" />`
CMSMaxTransactionWaitTimeout	Indicates how many seconds a transaction should wait for completion of another running transaction if simultaneous transactions are not allowed. The default value is *1*.	`<add key = "CMSMaxTransactionWaitTimeout" value="1" />`

Special settings for the scheduler

By adding the following keys to your web.config, you can configure the Scheduler:

Key	Description	Sample Value
CMSUseAutomaticScheduler	Indicates if automatic scheduling should be used. When enabled, the scheduler periodically requests the **cmspages/scheduler.aspx** page which ensures that scheduled tasks are processed regularly, even if there is no website activity. If turned off (*false* - by default), tasks are processed at the end of standard page requests.	`<add key = "CMSUseAutomaticScheduler" value="true" />`
CMSRunSchedulerWithinRequest	If *true* (the default value), the scheduler is executed within the standard **EndRequest** event of a page. If *false*, the scheduler is executed via the *~/cmspages/scheduler.aspx* page.	`<add key = "CMSRunSchedulerWithinRequest" value="false" />`
CMSSchedulerURL	URL of the physical location of the scheduler.aspx page. The default value is *~/cmspages/scheduler.aspx*	`<add key="CMSSchedulerURL" value="https://domain/cmspages/scheduler.aspx" />`
CMSSchedulerAcceptAllCertificates	If true, all security certificates (including not valid ones) will be accepted when accessing the *scheduler.aspx* page via a secured protocol. The default value is *false*.	`<add key = "CMSSchedulerAcceptAllCertificates" value="true" />`
CMSSchedulerUserName	Sets the user name under which the *scheduler.aspx* page should be accessed (e.g. when using windows authentication). The default value is "" (blank username).	`<add key ="CMSSchedulerUserName" value="office\myname" />`
CMSSchedulerPassword	Sets the password for the user name under which the *scheduler.aspx* page should be accessed. The default value is "" (blank password).	`<add key ="CMSSchedulerPassword" value="mypassword123" />`

Special Contact management settings

By adding the following keys to your web.config, you can configure the behavior of the Contact management module.

Key	Description	Sample Value
CMSLogActivityImmediatelyToDB	If *false*, activities are logged to temporary files on the server disk, which are then	`<add key="CMSLogActivityImmed`

	regularly processed by a scheduled task. Otherwise, activities are logged immediately to the database. The default value is *false*. Please note that logging activities directly to the database may cause performance issues on high-traffic websites.	`iatelyToDB" value="true"/>`
CMSEnableContactBots	Determines whether web bots (e.g. search engine crawlers) should be logged and tracked as contacts when they access the website. The default value is *false*.	`<add key ="CMSEnableContactBots" value="true" />`

Special settings for security

By adding the following keys to your web.config, you can configure certain Security options:

Key	Description	Sample Value
CMSCheckParameters	Indicates if parameter checking is allowed. Read this topic for more details. The default value is *false*.	`<add key="CMSCheckParameters" value="true"/>`
CMSReportCheckParameters	If true, an exception reporting the parameters is thrown when the parameters do not match. Read this topic for more details. The default value is *false*.	`<add key = "CMSReportCheckParameters" value="true"/>`
CMSAcceptAllCertificates	Indicates whether application requests should accept all certificates (including invalid certificates). This key ensures the same as when both *CMSSchedulerAcceptAllCertificates* and *CMSStagingAcceptAllCertificates* were enabled at the same time. The default value is *false*.	`<add key = "CMSAcceptAllCertificates" value="true"/>`
CMSPasswordSalt	The password salt is a string that is appended to user passwords before they are hashed (to improve security). By default, it always contains the randomly generated GUID of the given user. The content of this key is added after the GUID, so it can be used to further increase the length of the salt. This is only applied if you store passwords using the *SHA2 with salt*	`<add key="CMSPasswordSalt" value="SaltText" />`

	format, which can be configured in **Site Manager -> Settings -> Membership & Security -> Passwords -> Password format**.	
CMSAllowOnlySimpleMacros	If true, only simple macros (i.e. those which do not need a security check) are allowed. All others will not be resolved. If true, CMSTextBox does not add security. The default value is *false*.	`<add key="CMSAllowOnlySimpleMacros" value="true"/>`

Special settings for the Smart search module

The following keys can be used to configure the Smart search module:

Key	Description	Sample Value
CMSSearchIndexPath	Sets the path of a custom directory where smart search index files should be physically stored.	`<add key="CMSSearchIndexPath" value = "App_Data\MyCustomIndexes" />`
CMSSearchOnlyWhenContentPresent	Determines if smart searches should be performed only when there is at least one standard content keyword specified in the search expression. The default value is *true*. Setting this key to *false* allows search expressions containing only special Lucene query syntax, i.e. direct field searches such as: *DocumentNodeID:(int)17* This key is now obsolete. The same result can be ensured by leaving the **Block field-only search** property disabled for individual Smart search results or Smart search dialog with results web parts.	`<add key = "CMSSearchOnlyWhenContentPresent" value="false" />`
CMSCreateTemplateSearchTasks	If enabled, any changes made to a page template will automatically trigger an update of all documents that are based on the given template in the appropriate smart search indexes. The default value is *true*. You can set this key to ***false*** if you wish to	`<add key = "CMSCreateTemplateSearchTasks" value="false" />`

	improve performance or save resources in scenarios where you have a very large number of documents on your website that share the same page template. This way, the system will no longer perform bulk updates of documents in the search index whenever their template is modified. Disabling this key will mean that your document index will not reflect changes to the template (e.g. if you add static text to the template through a web part) until the given documents are updated for some other reason, or the entire index is rebuilt. The key only affects changes to page templates. Editing the content of editable regions on a specific document will always cause it to be updated in the index.	
CMSProcessSearchTasksByScheduler	By default, the smart search creates and runs indexing tasks immediately whenever content covered by a search index is created or modified. If you set this key to *true*, the system does NOT run indexing tasks upon creation — they need to be processed periodically, for example using the **Execute search tasks** scheduled task. The default value is *false*.	`<add key = "CMSProcessSearchTasksByScheduler" value="true" />`
CMSSmartSearchLockPollInterval	Specifies how often (time in milliseconds) the smart search indexer tries to acquire a lock on an index file. The default value of this key is 500 milliseconds. You should lower the value if the following exception gets logged into your Event log. *Lock obtain timed out: CMS.SiteProvider.SearchLock*	`<add key = "CMSSmartSearchLockPollInterval" value="500" />`
CMSSmartSearchLockTimeout	Defines the timeout period (in milliseconds) during which the smart search indexer should try to acquire a lock on the index file.	`<add key = "CMSSmartSearchLockTimeout" value="1000" />`

	The default value of this key is 1000 milliseconds. You should increase the value if the following exception gets logged into your Event log. *Lock obtain timed out: CMS.SiteProvider.SearchLock*	

You can also use the keys below to configure how **Documents crawler** search indexes access sites:

Key	Description	Sample Value
CMSCrawlerSearchDomain	Specifies the name of the domain that the crawler should use when indexing sites. This is most useful for web farm servers that do not have access to the main domain.	`<add key = "CMSCrawlerSearchDomain" value="domain_name" />`
CMSCrawlerFormsUserName	Sets the user name under which the crawler indexes site documents. Documents for which the specified user does not have *Read* permissions will not be indexed. If not defined, the default *administrator* user account is used.	`<add key = "CMSCrawlerFormsUserName" value="user_name" />`
CMSCrawlerDomainUserName	Used instead of the **CMSCrawlerFormsUserName** key if Windows authentication is enabled.	`<add key = "CMSCrawlerDomainUserName" value ="windows_user_name" />`
CMSCrawlerDomainPassword	Contains the password for the account used by the crawler if Windows authentication is enabled.	`<add key = "CMSCrawlerDomainPassword" value="password" />`

Special settings for the Translation services module

You can use the following key to configure the Translation services module:

Key	Description	Sample Value
CMSTranslationServicesUseCDATAForTransUnit	Use if you want to export XLIFF files without the CDATA notation.	`<add key = "CMSTranslationServicesUseCDATAForTransUnit"`

		value="false" />

Special settings for user interface cultures

By adding the following keys to your web.config, you can configure UI cultures:

Key	Description	Sample Value
CMSDefaultSpellCheckerCulture	Specifies the default culture of the built-in spell-checker. This culture is used when the dictionary for the currently selected content culture is not found. The default value is *en-us*.	<add key = "CMSDefaultSpellCheckerCulture" value="en-US" />
CMSShowLogonCultureSelector	Indicates if the user interface logon page should display a drop-down list with available user interface languages. The default value is *true*.	<add key = "CMSShowLogonCultureSelector" value="false" />
CMSDefaultUICulture	Specifies the default UI culture. If you use this key, you also need to rename the *~\CMSResources\CMS.resx* file to *CMS.en-us.resx* and the *CMS.en-nz.resx* file to *CMS.resx*. This is needed because the *CMS.resx* file is used when the *(default)* option is selected in users' *Preferred user interface culture*. The default value is *en-us*.	<add key ="CMSDefaultUICulture" value="en-nz"/>
CMSUseSQLResourceManagerAsPrimary	Changes the priority of used localization resource strings to: 1. custom.resx 2. cms.resx 3. database (Site Manager -> Development -> UI Cultures) The default value is *true*.	<add key = "CMSUseSQLResourceManagerAsPrimary" value="false" />
UICulture	Specifies the default user interface culture. The default value is *en-us*.	<add key="UICulture" value="en-us"/>

Special settings for synchronization on web farms

By adding the following keys to your web.config, you can enable or disable web farm synchronization of certain kind of files stored in the file system:

Key	Description	Sample Value
CMSWebFarmEnabled	Indicates if Web farms are enabled (true) or not (false).	<add key="CMSWebFarmEnabled"

	The default value is *false*.	value="true"/>
CMSWebFarmServerName	Code name of the web farm server. This value is used for native web farm synchronization support. The default value is "" (empty string).	<add key ="CMSWebFarmServerName" value="server1" />
CMSWebFarmSynchronizeFiles	General key determining if files should be synchronized (true) or not (false). This key enables synchronization of: • Attachments • Meta files • Media files • Form files • Avatars • Forum attachments The default value is *true*.	<add key = "CMSWebFarmSynchronizeFiles" value="true" />
CMSWebFarmSynchronizeAttachments	Enables/disables synchronization of attachments. The default value is *true*.	<add key = "CMSWebFarmSynchronizeAttachments" value="true" />
CMSWebFarmSynchronizeMetaFiles	Enables/disables synchronization of meta files. The default value is *true*.	<add key = "CMSWebFarmSynchronizeMetaFiles" value="true" />
CMSWebFarmSynchronizeMediaFiles	Enables/disables synchronization of media files. The default value is *true*.	<add key = "CMSWebFarmSynchronizeMediaFiles" value="true" />
CMSWebFarmSynchronizeBizFormFiles	Enables/disables synchronization of form files. The default value is *true*.	<add key = "CMSWebFarmSynchronizeBizFormFiles" value="true" />
CMSWebFarmSynchronizeAvatars	Enables/disables synchronization of Avatars. The default value is *true*.	<add key = "CMSWebFarmSynchronizeAvatars" value="true" />
CMSWebFarmSynchronizeForumAttachments	Enables/disables synchronization of forum attachments. The default value is *true*.	<add key = "CMSWebFarmSynchronizeFo

		rumAttachments" value="true" />
CMSWebFarmSynchronizeDeleteFiles	Enables/disables synchronization of deleted files. The default value is *true*.	<add key = "CMSWebFarmSynchronizeDeleteFiles" value="true" />
CMSWebFarmMaxFileSize	If the **CMSWebFarmSynchronizeFiles** key is enabled, you can limit the maximal size of synchronized files using this key. The value is entered in kiloBytes and files larger than this value will not be synchronized. The default value is *2147483647 (int.MaxValue)*.	<add key ="CMSWebFarmMaxFileSize" value="1024" />
CMSWebFarmUpdateWithinRequest	If true, web farm servers are updated with changes made on the other servers once per request. If false, web farm servers can be updated based on the interval settings of the Synchronize web farm changes scheduled task - recommended for high-traffic sites. The default value is *true*.	<add key = "CMSWebFarmUpdateWithinRequest" value="false" />
CMSWebFarmUseDBUpdater	Indicates whether the database updater should be used.	<add key = "CMSWebFarmUseDBUpdater" value="true" />
CMSWebFarmGenerateServers	If true, web farm servers will be generated automatically. You don't need to create them manually in the user interface.	<add key = "CMSWebFarmGenerateServers" value="true" />
CMSWebFarmApplicationPhysicalPath	Path to the application on the disk used for synchronizing physical files.	<add key = "CMSWebFarmApplicationPhysicalPath" value="C:\inetpub\wwwroot\KenticoCMS" />
CMSTeamDevelopmentEnabled	If true, turns on CMSWebFarmGenerateServers and CMSWebFarmUseDBUpdater settings	<add key = "CMSTeamDevelopmentEnabled" value="true" />

Special settings for deployment to Windows Azure

By adding the following keys to your web.config, you can set up the deployment of your website to Windows Azure and configure its behaviour. For more information, please refer to the Windows Azure Deployment Guide.

Key	Description	Sample Value
CMSAzureProject	Must be set to *true* if you wish to run the application on Windows Azure. *True* by default if the application is installed as a Windows Azure project, *false* in standard installations.	`<add key="CMSAzureProject" value="true"/>`
CMSAzureAccountName	Specifies the name of the storage account that the application will use for its file system. The account must belong to a valid Windows Azure subscription. If you wish to run the application on the local emulator, enter *devstoreaccount1* as the value.	`<add key ="CMSAzureAccountName" value ="devstoreaccount1"/>`
CMSAzureSharedKey	Must contain the primary access key of the storage account specified in the **CMSAzureAccountName** setting. You can find the appropriate value for your storage account on the Windows Azure Management Portal.	`<add key="CMSAzureSharedKey" value = "Eby8vdM02xNOcqFlqUwJPLlmEtlCDXJ1OUzFT50uSRZ6IFsuFq2UVErCz4I6tq/K1SZFPTOtr/KBHBeksoGMGw==" />`
CMSAzureBlobEndPoint	Sets the endpoint used for the connection to the blob service of the specified storage account. If you wish to use the default endpoint, remove the setting completely from the appropriate files.	`<add key ="CMSAzureBlobEndPoint" value = "http://127.0.0.1:10000/devstoreaccount1"/>`
CMSAzureQueueEndPoint	Sets the endpoint used for the connection to the queue service of the specified storage account. If you wish to use the default endpoint, remove the setting completely from the appropriate files.	`<add key ="CMSAzureQueueEndPoint" value = "http://127.0.0.1:10001/devstoreaccount1"/>`
CMSAzureTableEndPoint	Sets the endpoint used for the connection to the table service of the specified storage account. If you wish to use the default endpoint, remove the setting completely from the appropriate files.	`<add key ="CMSAzureTableEndPoint" value = "http://127.0.0.1:10002/devstoreaccount1"/>`

CMSAzureRootContainer	Specifies the name of the blob container that will serve as the root of the application's file system on the Windows Azure storage account. This can be useful in scenarios where multiple applications use the same storage account. The default value is *cmsroot*.	`<add key ="CMSAzureRootContainer" value="CustomRoot"/>`
CMSAzurePublicContainer	Indicates if the blob container used to store the applications file system should be public. If set to *true*, it will be possible to access files directly through the URL of the appropriate blob service, for example: *<StorageAccountName>.blob.core.windows.net/cmsroot/corporatesite/media/imagelibrary/logo.png*	`<add key = "CMSAzurePublicContainer" value="true"/>`

If you wish to host the website itself on-premise, but use a file system based on the Blob service of a Windows Azure storage account, you can specify the following settings:

Key	Description	Sample Value
CMSExternalStorageName	You can add this key to specify that the application should use a Windows Azure storage account for its file system. This can be done by setting the value to *Azure*. It is not necessary to add this key if **CMSAzureProject** is set to *true*. Only use this option if you are setting up a hybrid scenario with the application itself hosted on-premise outside of the Windows Azure cloud.	`<add key = "CMSExternalStorageName" value="Azure" />`
CMSAzureTempPath	The folder specified by this key will be used to store temporary files on a local disk, e.g. when transferring large files to or from the storage account. Do not use this key if the entire application is deployed as a Windows Azure hosted service.	`<add key="CMSAzureTempPath" value="C:\AzureTemp" />`
CMSAzureCachePath	Specifies a folder on a local disk where files requested from the storage account will be cached. This helps minimize the amount of blob storage operations, which saves time and resources. Do not use this key if the entire application is deployed as a Windows Azure hosted service.	`<add key="CMSAzureCachePath" value="C:\AzureCache" />`

CMSDownloadBlobTimeout	Specifies the timeout interval in minutes for importing files from Windows Azure Blob storage into Kentico. The default value is *1.5* minutes. Increase the interval if you encounter problems when importing large (about 2GB) files.	`<add key = "CMSDownloadBlobTimeout" value="50" />`

Special settings for debugging

While debugging can be turned and configured on by means of settings in **Site Manager -> Settings -> System -> Debug**, it is also possible to perform this configuration by adding certain keys into your project's *web.config* file. These keys are listed and described together with the respective settings in the Development -> Debugging and system information chapter of this guide.

CMSDebugScheduler	If false, all the scheduler operations are excluded from debugs. The default value is *true*.	`<add key="CMSDebugScheduler" value="false"/>`
CMSDebugFiles	Enables IO operation debugging and the **IO** tab in **Site Manager -> Administration -> System -> Debug**. The default value is *false*.	`<add key="CMSDebugFiles" value="true"/>`
CMSDebugFilesLive	If enabled, IO operation debug information is also displayed at the bottom of each live site page. This option requires IO operation debugging to be enabled. The default value is *false*.	`<add key="CMSDebugFilesLive" value="true"/>`
CMSDebugAllFiles	If enabled, IO operations called by pages of the administration interface (CMS Desk and Site Manager) will also be included in the IO operation debug. This option requires IO operation debugging to be enabled. The default value is *false*.	`<add key="CMSDebugAllFiles" value="true"/>`
CMSDebugFilesLogLength	Sets the maximum length of the IO operation debug log on the **Debug -> IO** tab, i.e. the number of requests for which debug information is preserved and displayed on the tab. If empty, value of the **Default log length** setting (or the **CMSDebugEverythingLogLength** key) is used. The default value is *10*.	`<add key="CMSDebugAllFiles" value="15"/>`

CMSLogFiles	If enabled, the IO operation debug log is saved into the *logfiles.log* file in the *~\App_Data* folder. This option does not require IO operation debugging to be enabled. The default value is *false*.	`<add key="CMSLogFiles" value="true"/>`

Connection string

The database used by Kentico CMS engine is specified by the connection string CMSConnectionString in the **/configuration/connectionStrings** section. Here's an example of such connection string:

```
<add name="CMSConnectionString" connectionString="Persist Security Info=False;
database=CMS;server=myserver;user id=sa;password=mypassword123;Current
Language=English;Connection Timeout=120;" />
```

Index

- 3 -

- A -

- B -

- C -

- D -

- E -

- F -

- G -

- H -

- I -

- J -

- K -

- L -

- M -

- N -

- O -

- P -

- Q -

- R -

- S -

- T -

- U -

- V -

- W -

- X -

- Y -