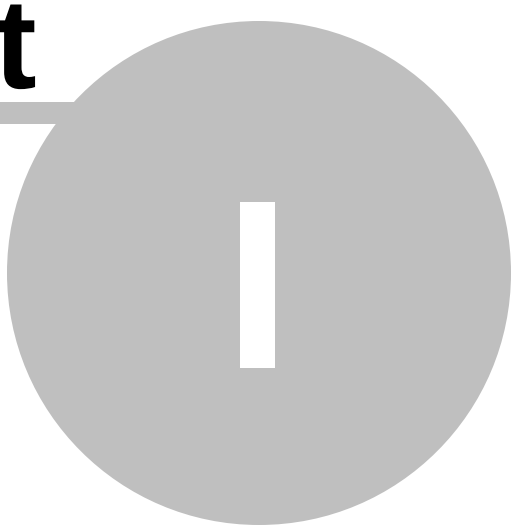
Kentico CMS 6.0 Developer's Guide

Kentico
unlimited website possibilities

www.kentico.com

Table of Contents

Part



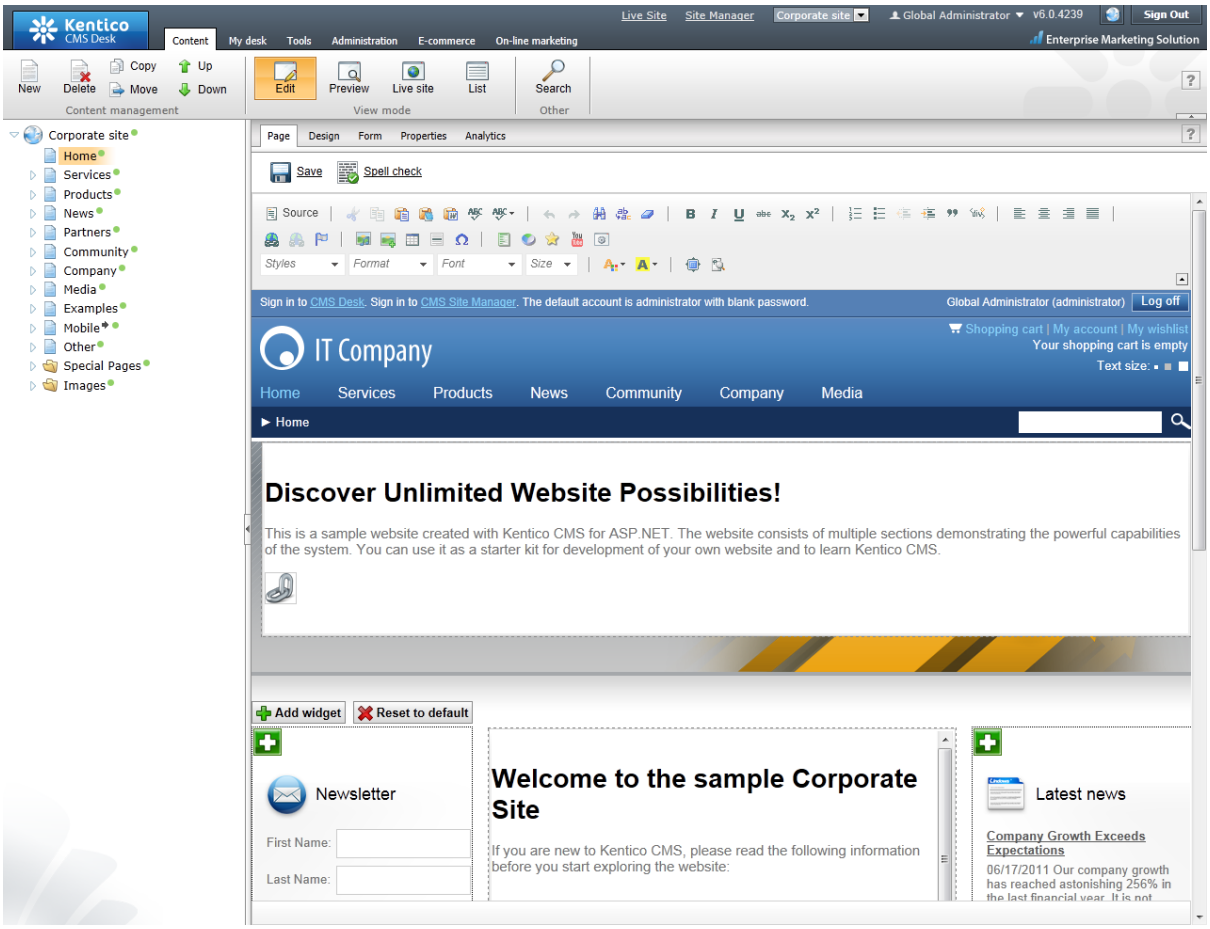
Introduction

1 Introduction

1.1 Introduction

Thank you for using Kentico CMS, the content management system for ASP.NET developers. Kentico CMS 6.0 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 and 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 - reference on in-box modules in Kentico CMS
- API programming and Kentico CMS internals - examples of the usage of Kentico CMS API and explanation of the internals of the system

Valuable reference 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

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.

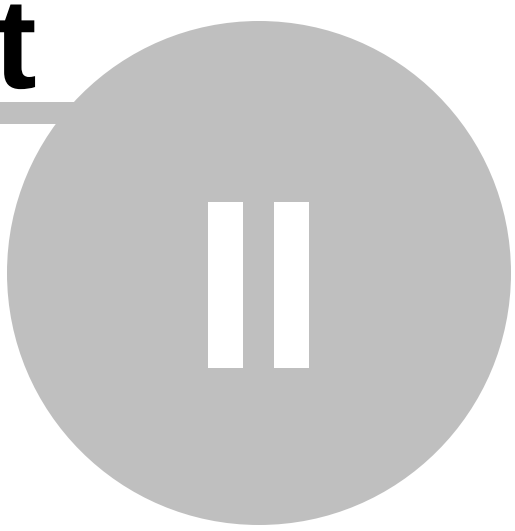
Paid license owners can also contact our highly-responsive support team, please visit www.kentico.com/support.aspx to learn more details.

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/Download/Version_History.aspx.

Part



Kentico CMS overview

2 Kentico CMS overview

2.1 What are the benefits of Kentico CMS?

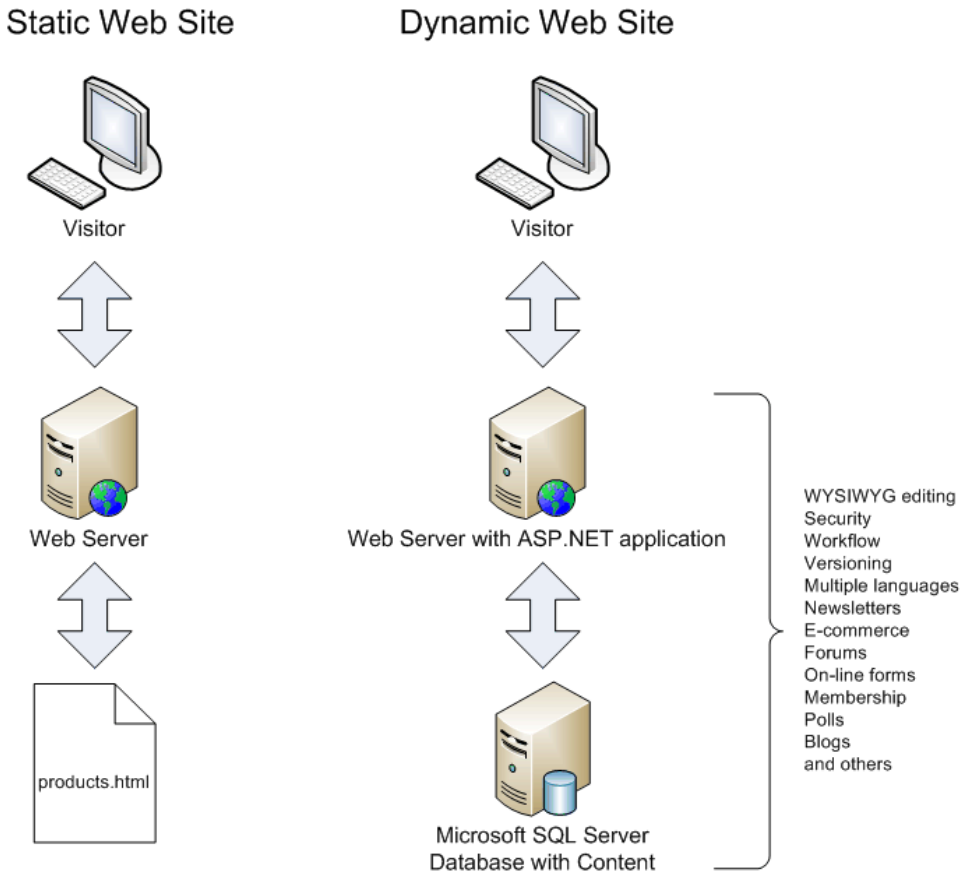
Before you dive into the details, it may be useful to understand the big picture. This chapter explains how the system works, describes its architecture and answers the most common questions you may have.

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 uses static HTML files stored on the 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 dynamic website with 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 at 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 hundreds of websites worldwide and it was designed to fit various needs of web developers and their clients. Beside, 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**.

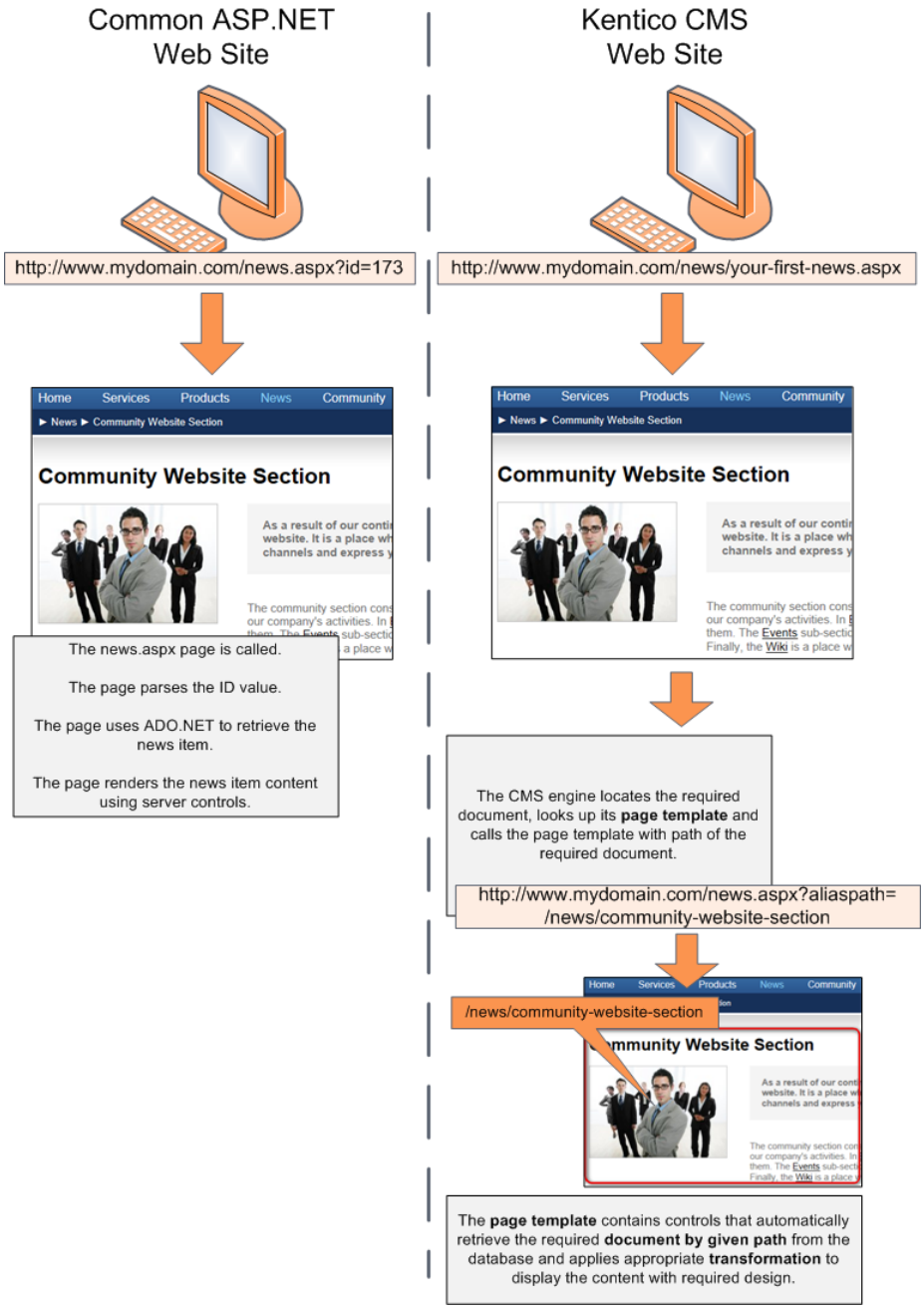
Don't take our word for it

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. It's 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. Then, you write .NET code only if you need to add additional business logic or functionality that is not supported out-of-the-box.

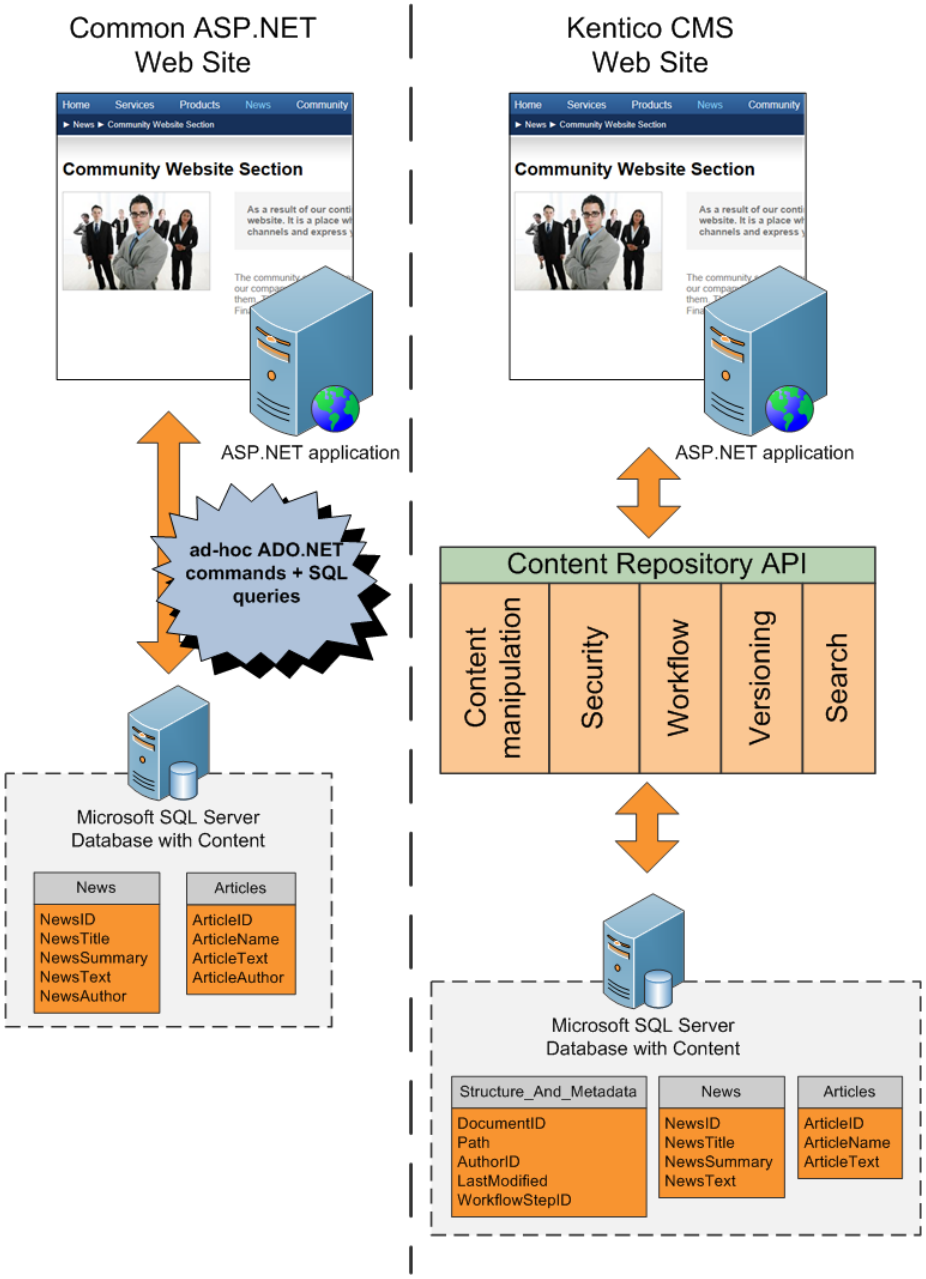
What else does the CMS do for me?

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**
- **same rules and mechanisms can be applied to any content type**, without writing additional code for every new type

An important part of 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 current workflow step
- content expiration
- permissions for the given document

In the example above, you can see that with 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

All metafiles and attachments are stored under a folder specified in **Site Manager -> Settings -> System -> Files -> Files folder**. 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**.

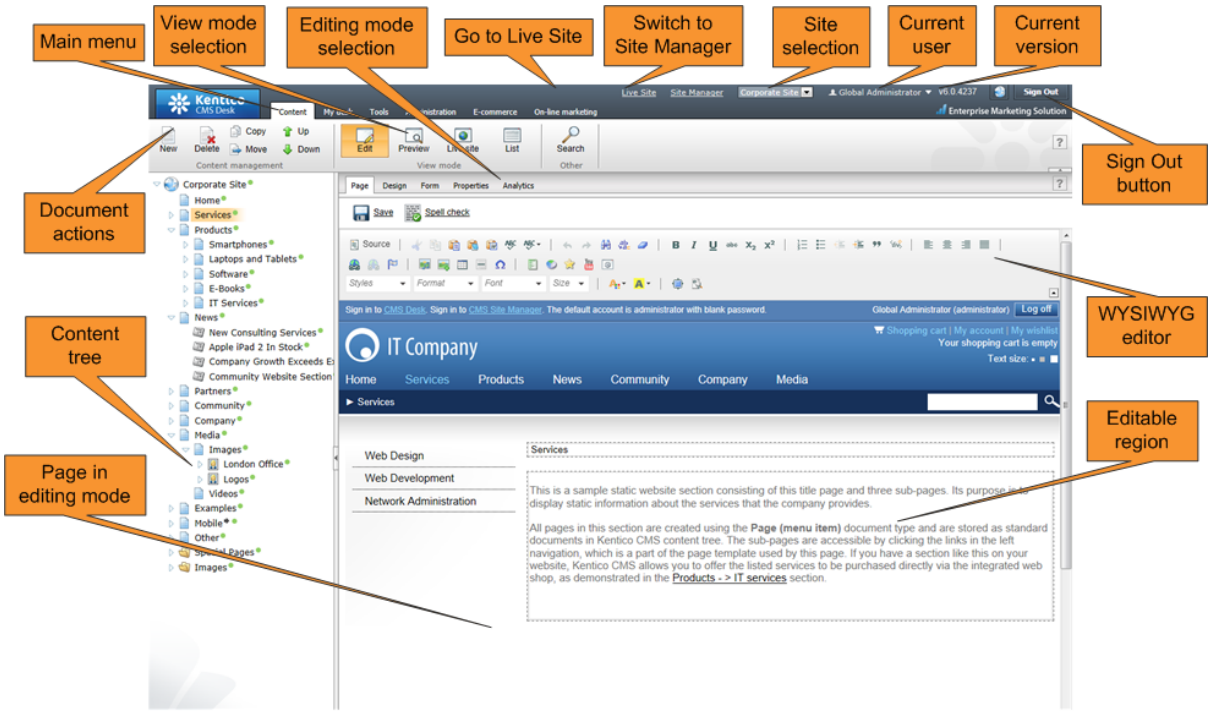
2.4 How do I edit content?

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

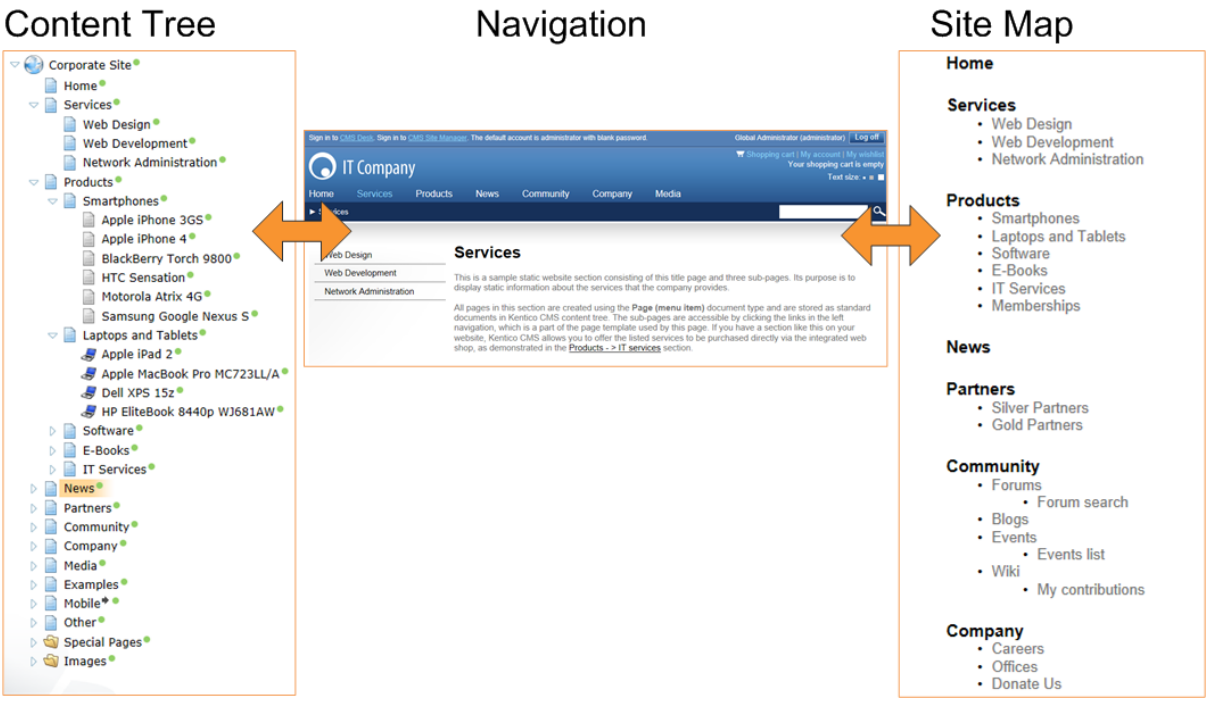
Kentico CMS comes with browser-based administration interface. It's divided into two parts:

- **CMS Desk** (typically http://www.example.com/cmsdesk) - the user interface for content editors.
- **CMS Site Manager** (typically http://www.example.com/cmssitemanager) - the user interface for administrators and web developers.

The following figure shows the CMS Desk user interface.



The left **content tree** 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 order documents.

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

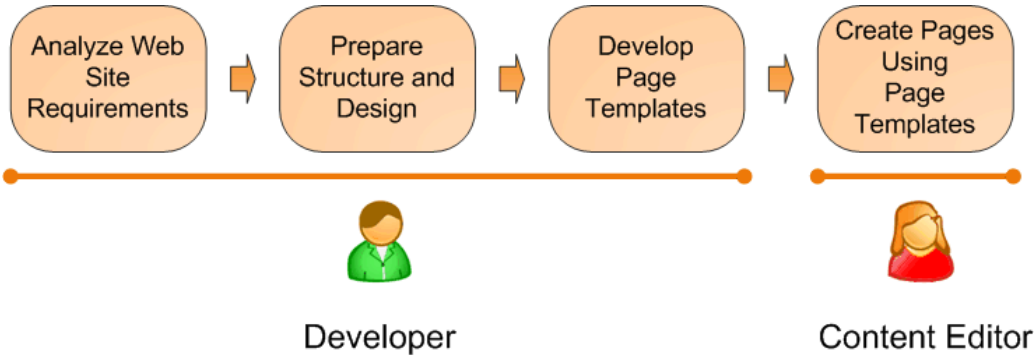
- **Edit** - editing mode
- **Preview** - preview mode. It displays the current version of the page before it's published (if you're using workflow). It also displays the content without using caching, which allows you to preview the content even if the live website displays the cached version.
- **Live site** - live site view
- **List** - displays a list of child documents under the currently selected document. It can be used for delete of multiple documents at the same time and it's useful if there are too many documents in the 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 you 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. Besides, any document can have also structured content that can be edited in the editing form on the Form tab.
- **Design** - here you can modify the page layout and web parts (this applies to portal engine development that will be described later in this guide). This tab is only available for global administrators (developers).
- **Form** - here you can modify the structured data, such as news title, news summary, release date, etc.
- **Product** - here you can modify the product specification if the given document represents a product that can be added to the shopping cart (if you're using the e-commerce module).
- **Properties** - here you can modify various settings, permissions, metadata and design settings of the document.

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 have take a quick overview of the development process:



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

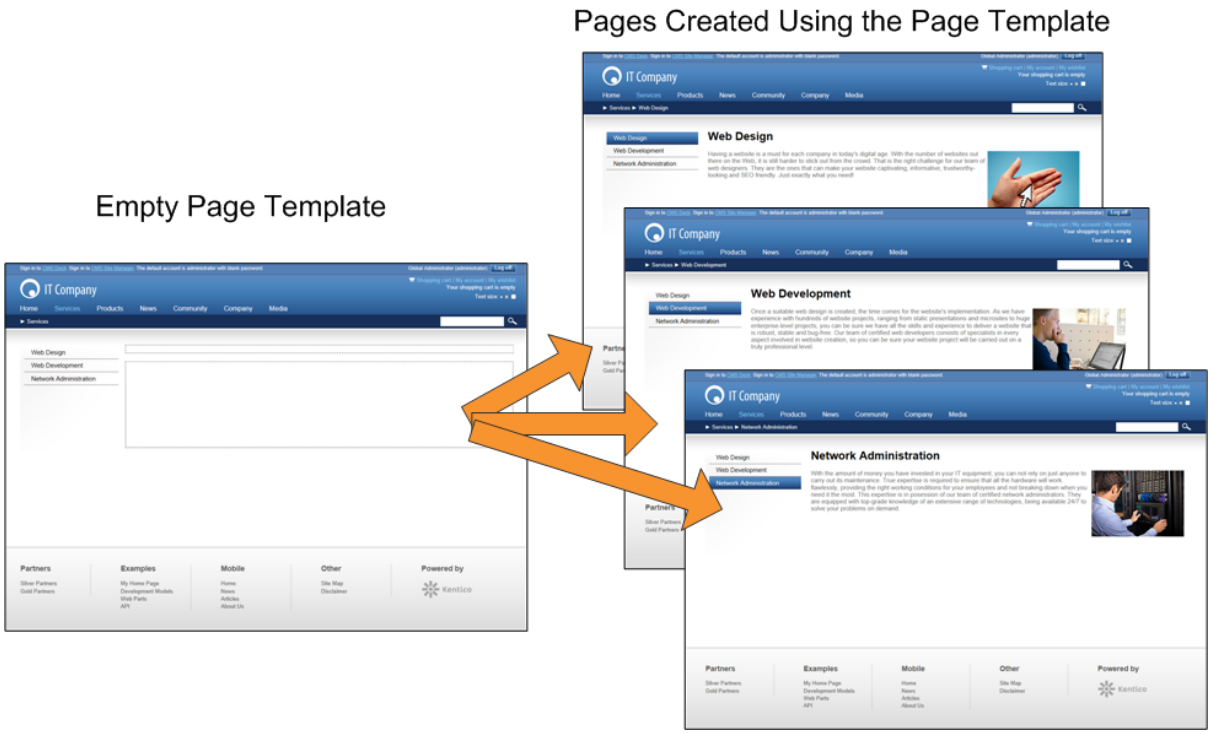
1. The developer analyzes the client requirements.
2. The developer prepares the site structure (site map) and web design.
3. The developer creates **page templates** for every type of the page (home page, solutions, products, news, etc.)
4. The content editor creates new pages - they enter text and images into the page templates defined by

the developer.

What is a page template?

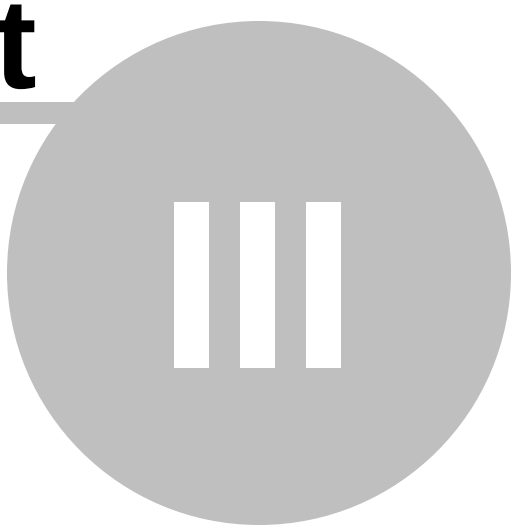
The page template is a predefined look of the page that allows content editors to enter the content. A single page template can be re-used for multiple pages with the same structure and design, but with different content.

The 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 will be described later.

Part



Installation and deployment

3 Installation and deployment

3.1 Overview

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

Installation on the local development machine

- Setup (kenticocms.exe) - installs only the web project 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.
- 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.
- Database setup - runs in the web user interface. It creates a new database on your SQL Server with system tables and basic data.
- 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 the 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:

- After you install Kentico CMS on your development machine using Setup (KenticoCMS.exe) you need to run Web Installer on your development machine and choose to install Kentico CMS on a remote server. Choose the temporary folder on your local disk.
- 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.
- Open 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.

If you encounter any problems during the installation, please see the Troubleshooting installation issues chapter.

Tip: There's no magic behind!

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 do 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 common backup/restore procedure. The connection string is stored in the *web.config* file, in the *ConnectionStrings/CMSConnectionString* element.

3.2 System requirements

The following configurations are supported by Kentico CMS. Other configurations may work too, but the system was not tested on them.

Server-side Requirements

- Windows XP, 2003, 2008, 2008 R2, Windows Vista Home Premium/Business/Enterprise/Ultimate or Windows 7 (both 32bit and 64bit)
- Microsoft .NET Framework 3.5 SP1 or higher
- Microsoft Internet Information Services (IIS) or Visual Studio/Visual Web Developer 2005/2008/2010 built-in web server
- Microsoft SQL Server 2005, 2008 (including free SQL Server Express Edition)

Hosting Requirements

- ASP.NET 3.5 SP1 (or higher) and Microsoft SQL Server 2005/2008 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 125 MB or more memory and 100+ MB database.

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

Development Tools

If you want to create custom web parts or integrate custom code, you need **Visual Studio 2005/2008/2010** or **Visual Web Developer 2005/2008/2010 Express Edition**.

Supported Client Browsers for Content Editors

- Internet Explorer 7, 8, 9
- Firefox 3.6, 4.0, 5.0
- Chrome 12
- Safari 4.0, 5.0, 5.1

Supported Client Browsers for Site Visitors

- Internet Explorer 6.0+
- Firefox 3.6+
- Chrome 12+
- Safari 4.0+
- Opera 10.50+

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

3.3 Installation procedure

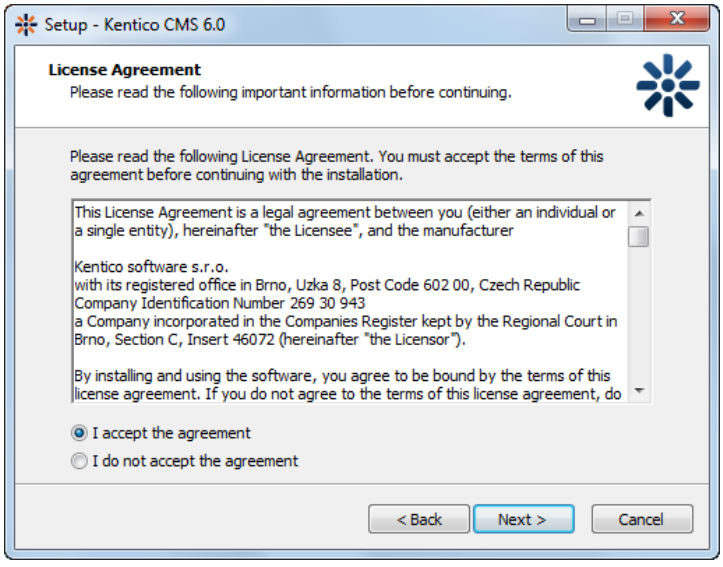
3.3.1 Setup (KenticoCMS.exe)

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.

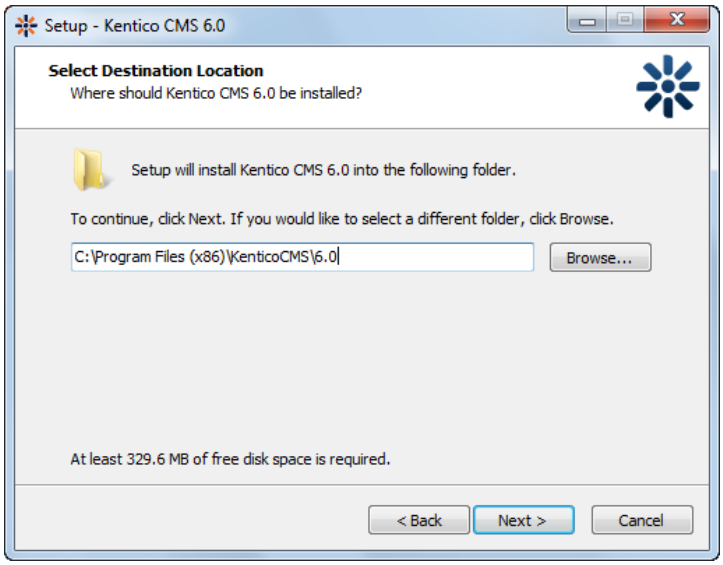
1. Run *KenticoCMS_<version>.exe*. You will see the welcome screen. Click **Next**.

2. Read and accept the license agreement if you want to continue. Click **Next**.



3. Choose the location where the Kentico Web Installer and documentation will be deployed. Click **Next** and then **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.



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

3.3.2 Web installer

3.3.2.1 Overview

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

Step 1 - Select .NET Framework version

First, you need to choose whether you use:

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

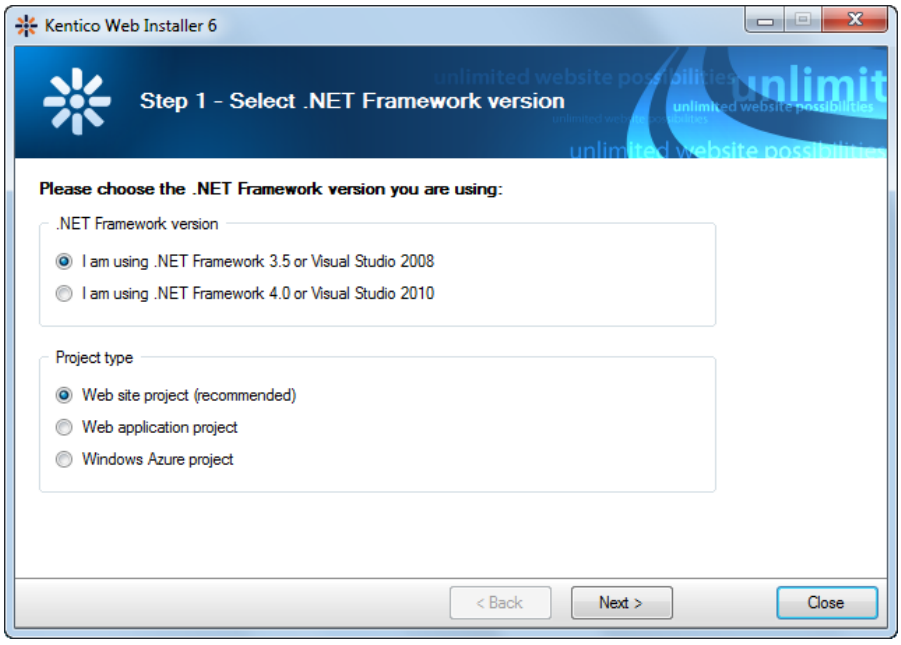
Depending on your choice, the installer will use appropriate *web.config* and *webproject.sln* files. The binaries and code are the same for both options; they are compiled for .NET 3.5 SP1 and can be used with .NET 4.0 as well.

Then you need to select from the following:

- **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.



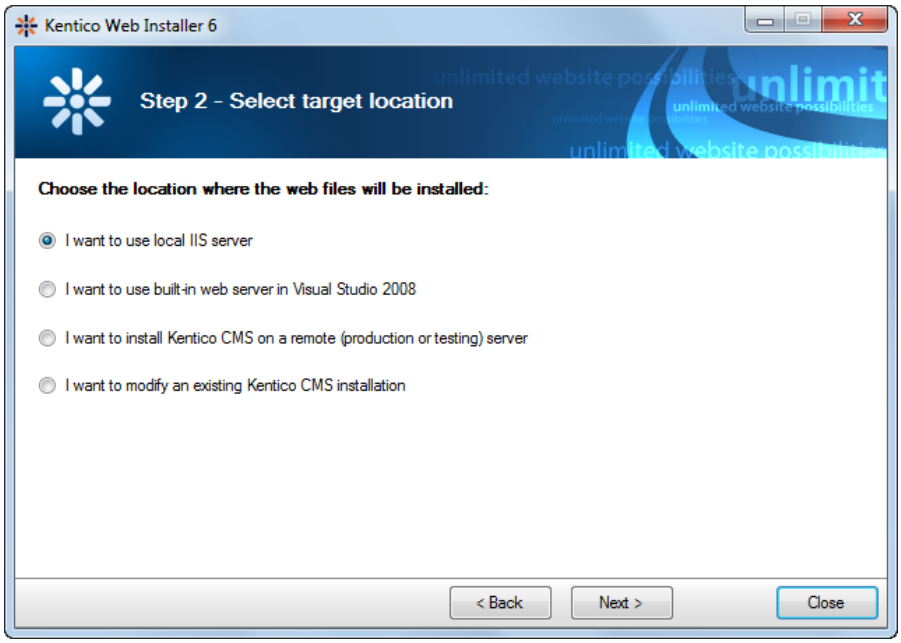
Step 2 - Choosing the target location

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 the local machine



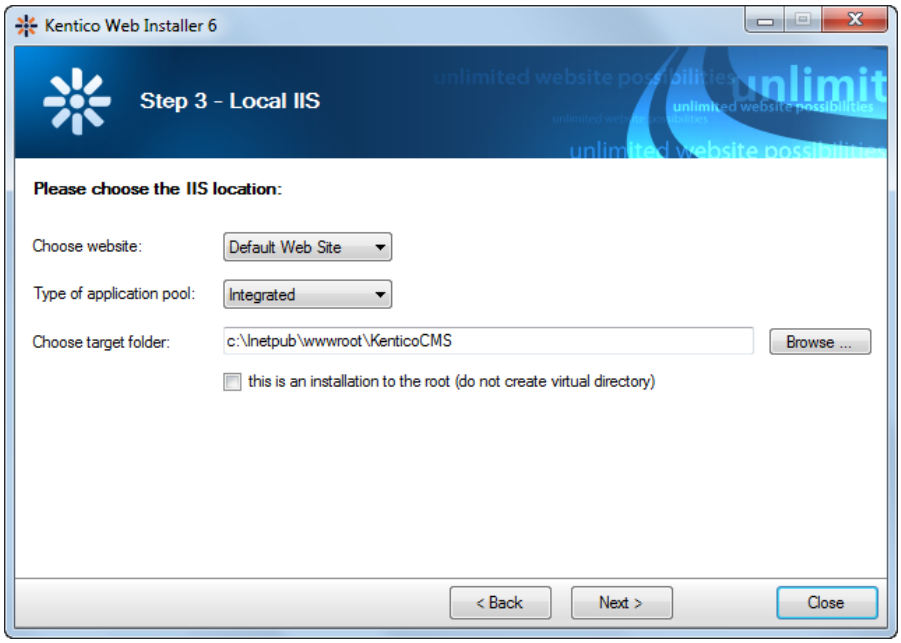
3.3.2.2 Local IIS server

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

1. In Step 3, you need to 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 here for more information)
- **Choose target folder**: choose the disk folder where web project files will be placed

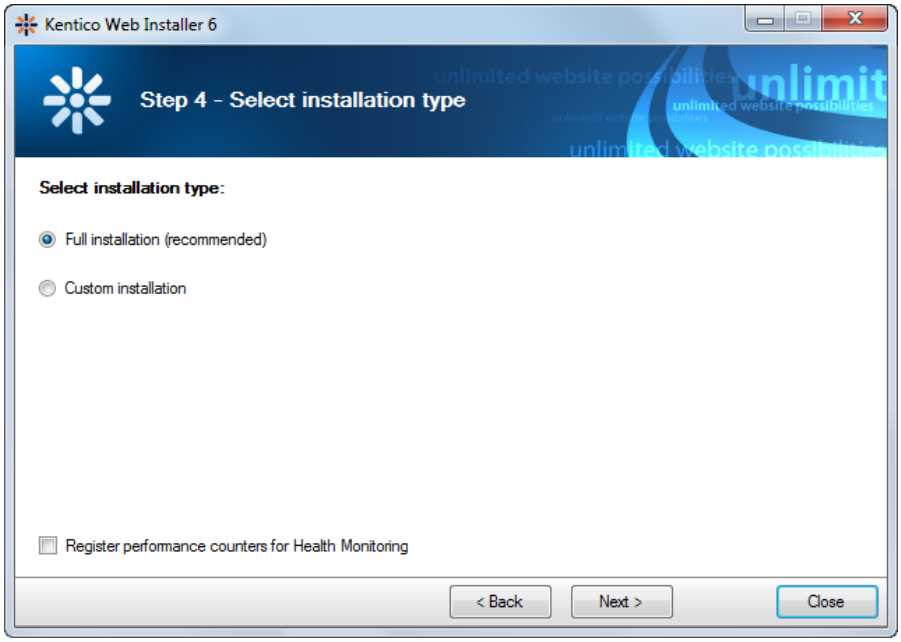
Please note: if you're installing Kentico CMS into the root of your website (such as *http://www.domain.com*) and do not wish to create a virtual directory (such as *http://www.domain.com/cms*), please check the **This is an installation to the root** check-box.



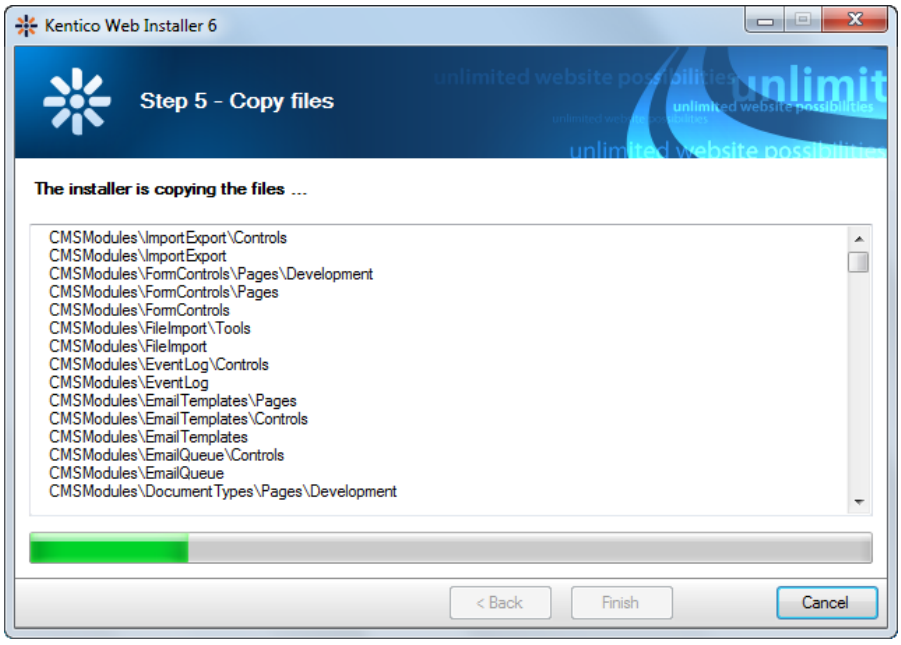
2. Now you need to decide between the following options:

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

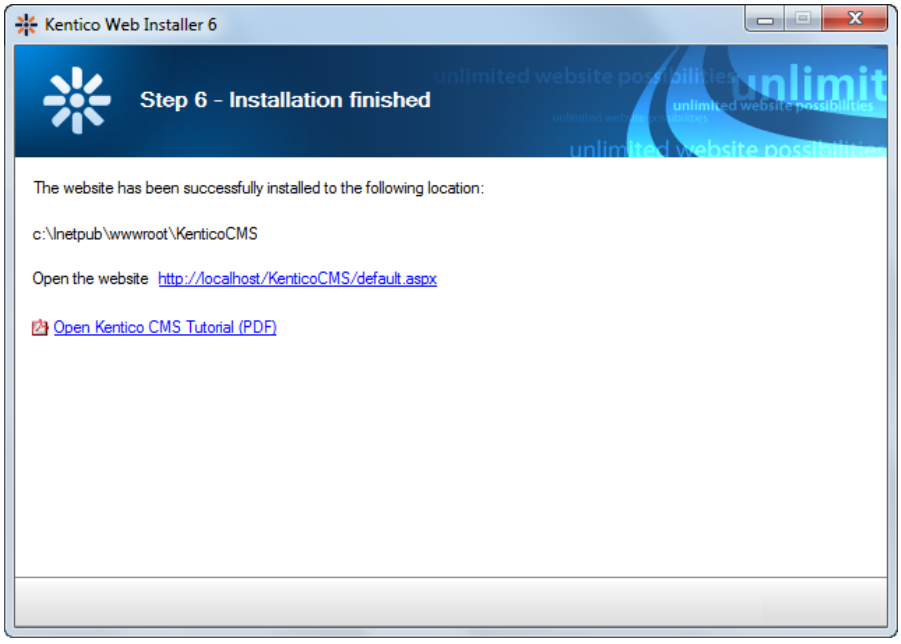
Enabling the **Register performance counters for Health Monitoring** option registers performance counters for the currently installed instance of Kentico CMS in Windows. The counters can be used for monitoring of system load and performance in an external application. Please refer to Modules -> Health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



3. In Step 5, the installer copies the project files to the specified folder:



4. At the end, the URL of the new Kentico CMS installation is displayed. Please put down this URL for future use.

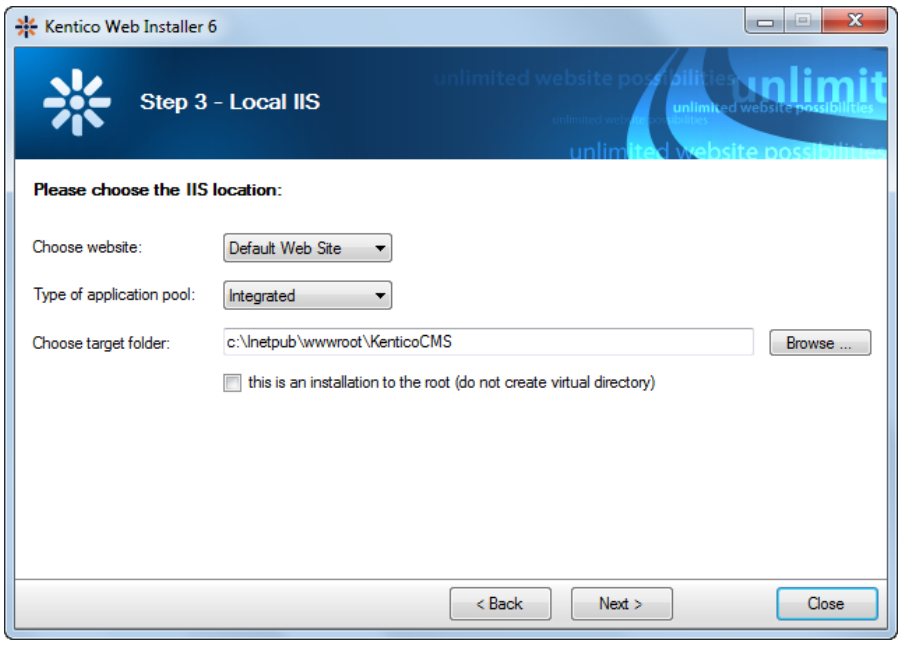


5. Now you can continue to the Database setup wizard.

3.3.2.3 Built-in web server in Visual Studio

If you want to use the build-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**.

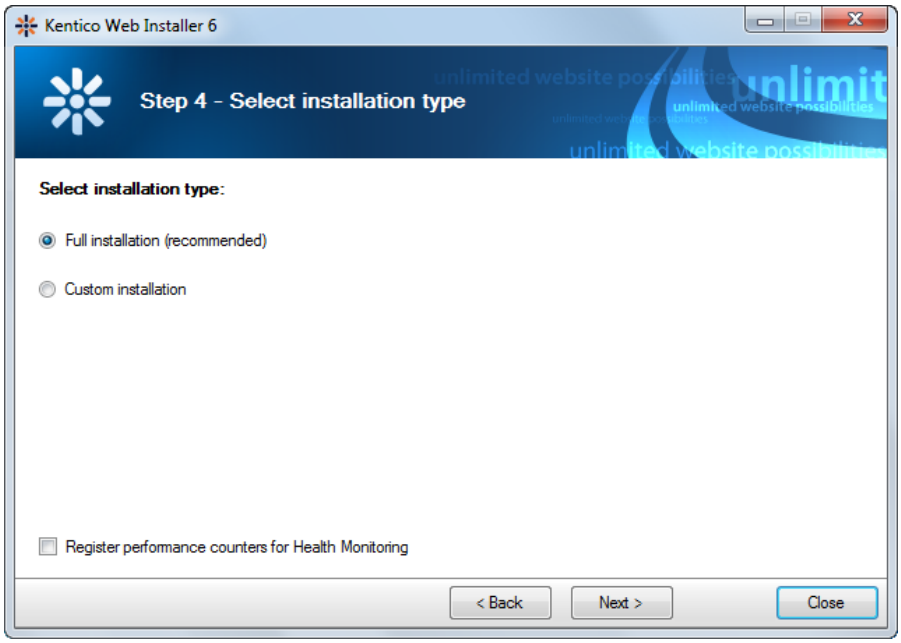
1. You will need to enter the target location of the files on your disk in Step 3 and proceed by clicking **Next**.



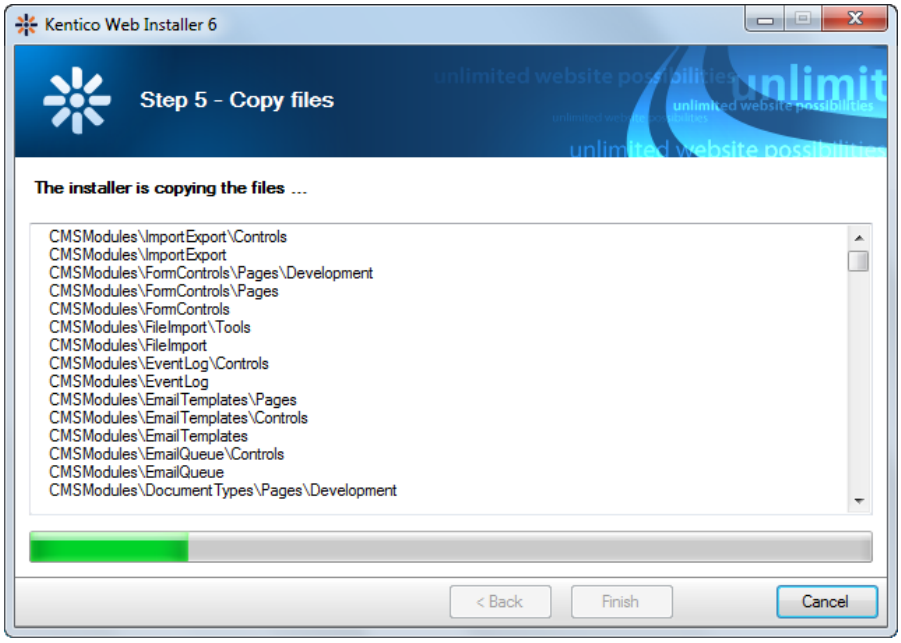
2. Now you need to decide between the following options:

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

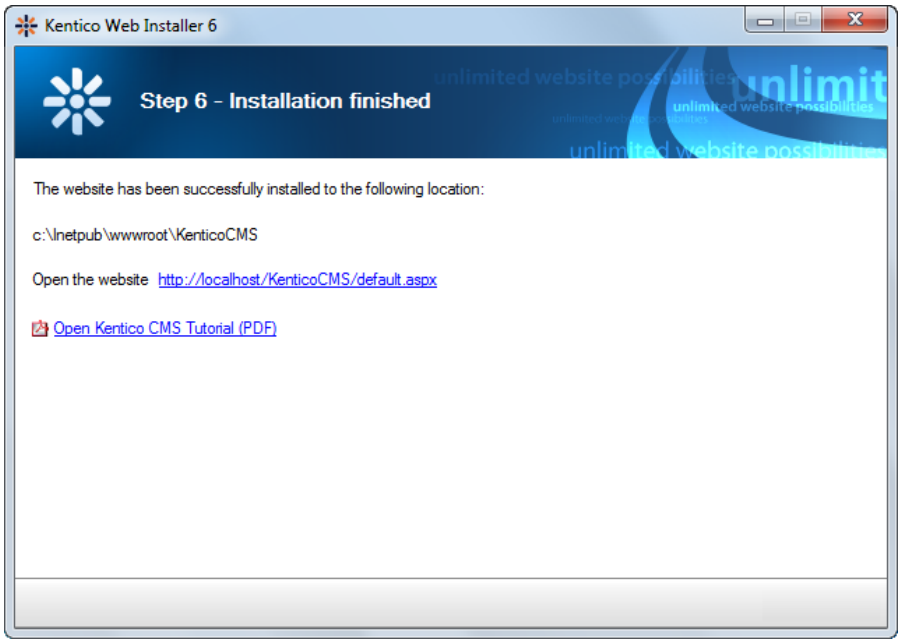
Enabling the **Register performance counters for Health Monitoring** option registers performance counters for the currently installed instance of Kentico CMS in Windows. The counters can be used for monitoring of system load and performance in an external application. Please refer to Modules -> Health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



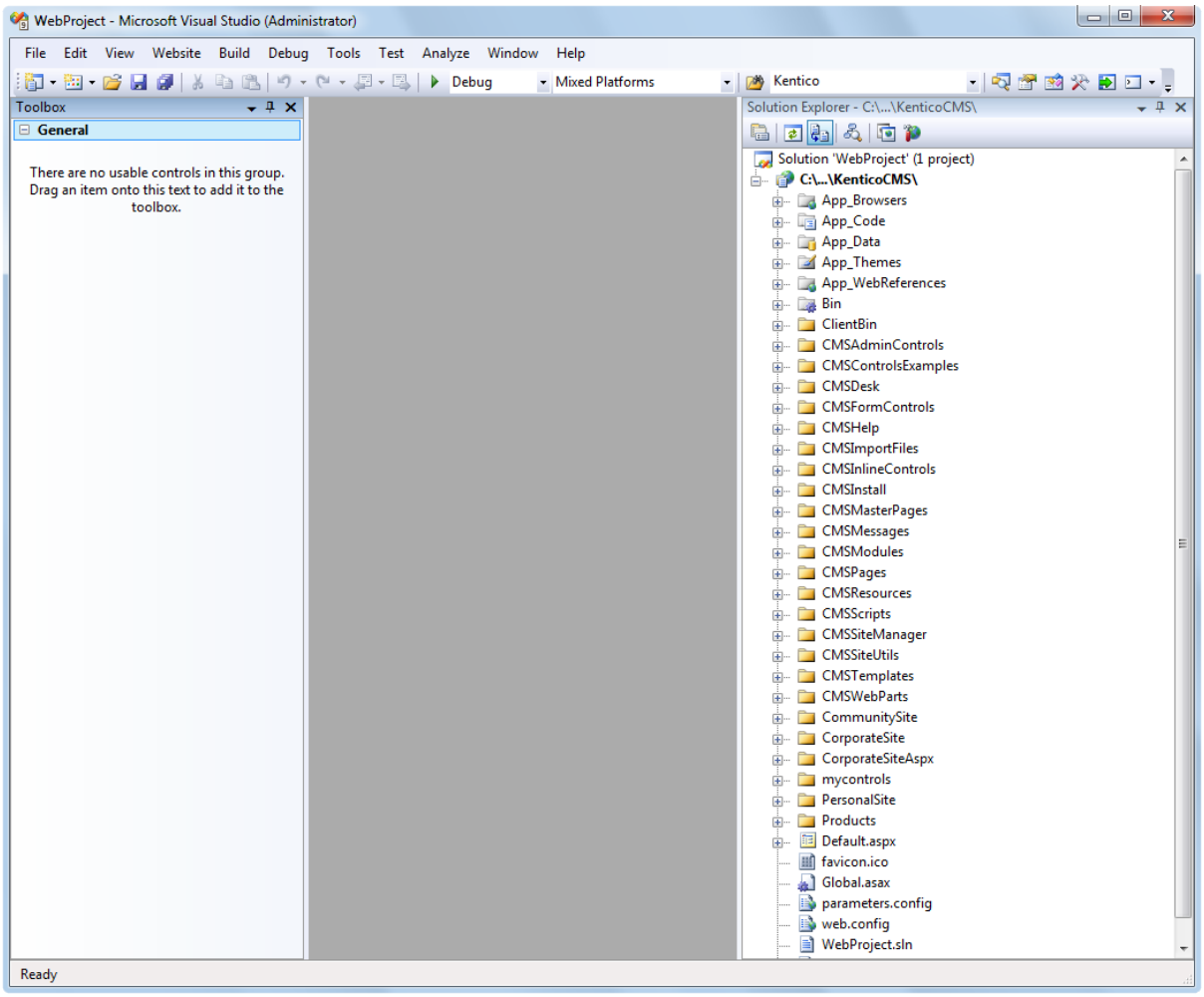
3. In Step 5, the installer copies the project files to the specified folder:



4. After the files are copied to the folder, you can open the solution in Visual Studio by clicking the link:



5. The project will open in Visual Studio:



6. 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.

7. Now you can continue to Database setup wizard.

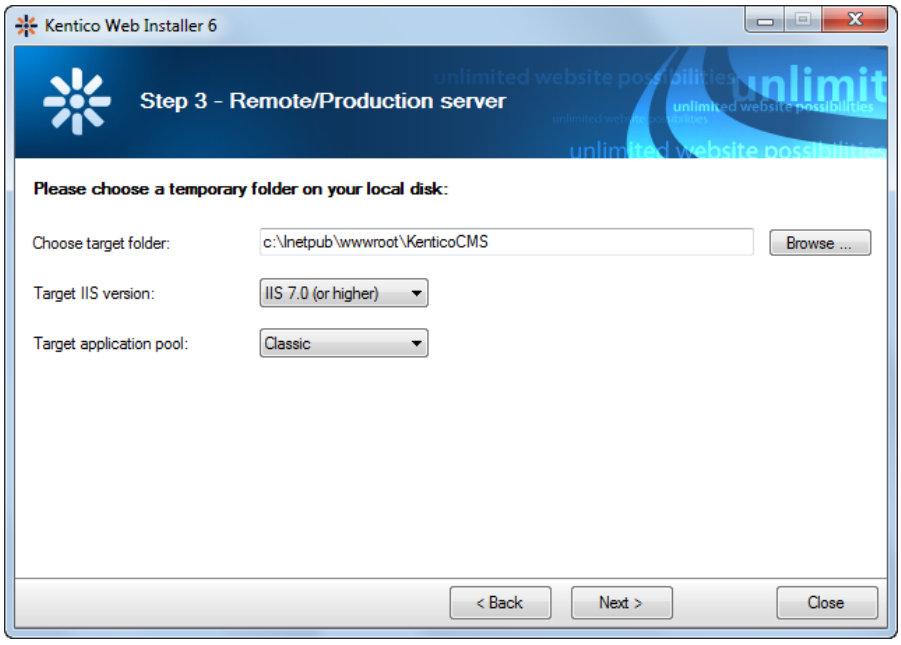
3.3.2.4 Remote server

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.

1. In Step 3, you will need to specify:

- **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.

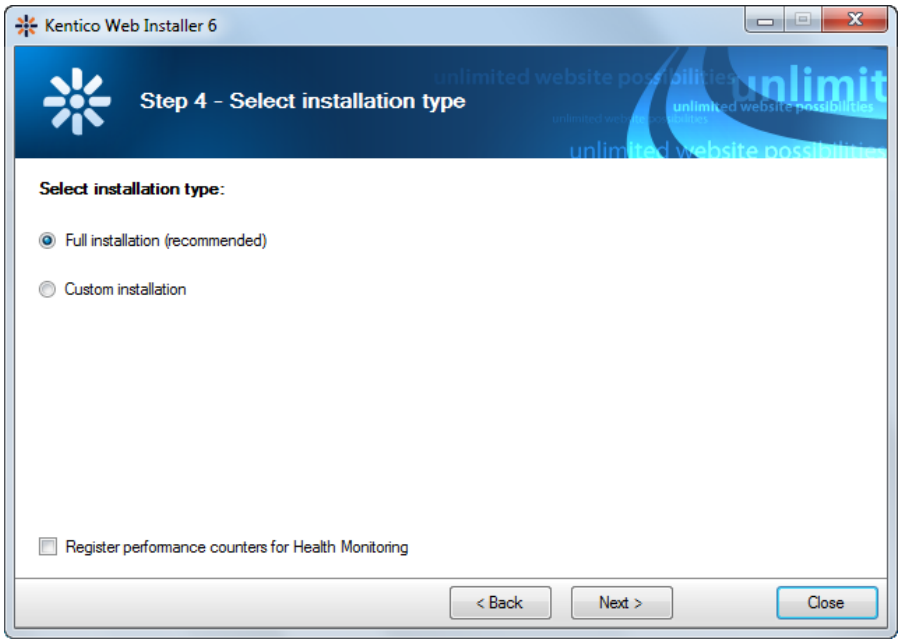
Click **Next** to proceed through the rest of the wizard.



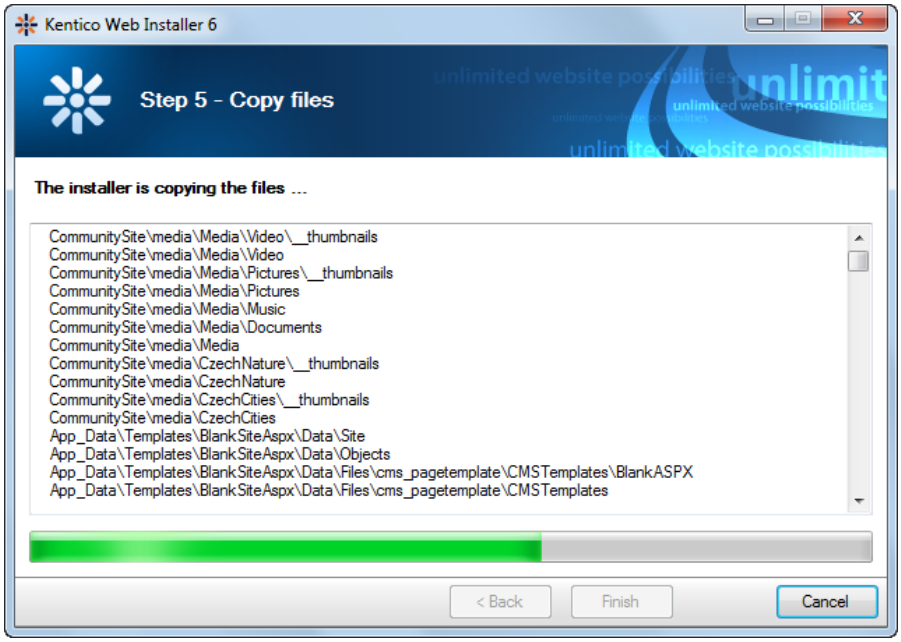
2. Now you need to decide between the following options:

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

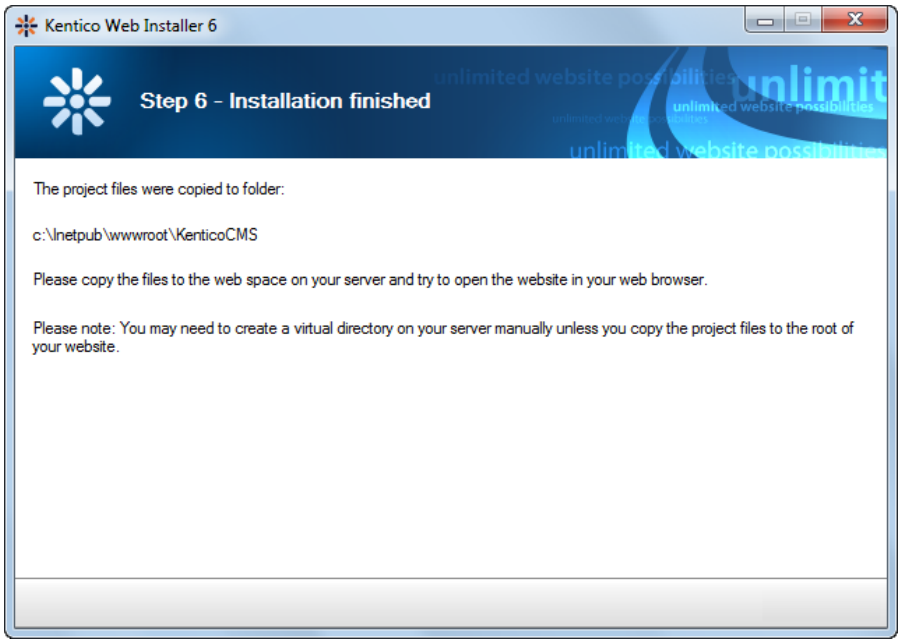
Enabling the **Register performance counters for Health Monitoring** option registers performance counters for the currently installed instance of Kentico CMS in Windows. The counters can be used for monitoring of system load and performance in an external application. Please note that by enabling this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



3. A log will be displayed in Step 5, showing you the progress of copying website files to the temporary location.



4. When you reach Step 6, the web project is created on your disk and a confirmation message is displayed:



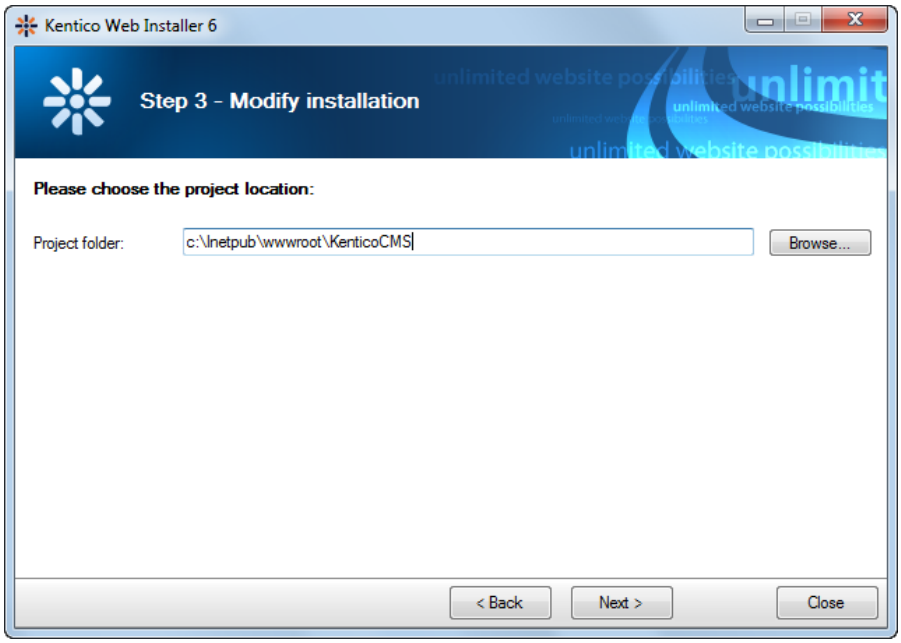
5. You need to copy the website to your server (e.g. over FTP). If your web project isn't placed in the root of the remote website, you may need to create a virtual directory as described in Creating a virtual directory.

6. Now you can continue to Database setup wizard.

3.3.2.5 Existing 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.

1. In Step 3, you need to specify the root folder of the project that you want to modify. Enter the path to the project folder and click **Next**.

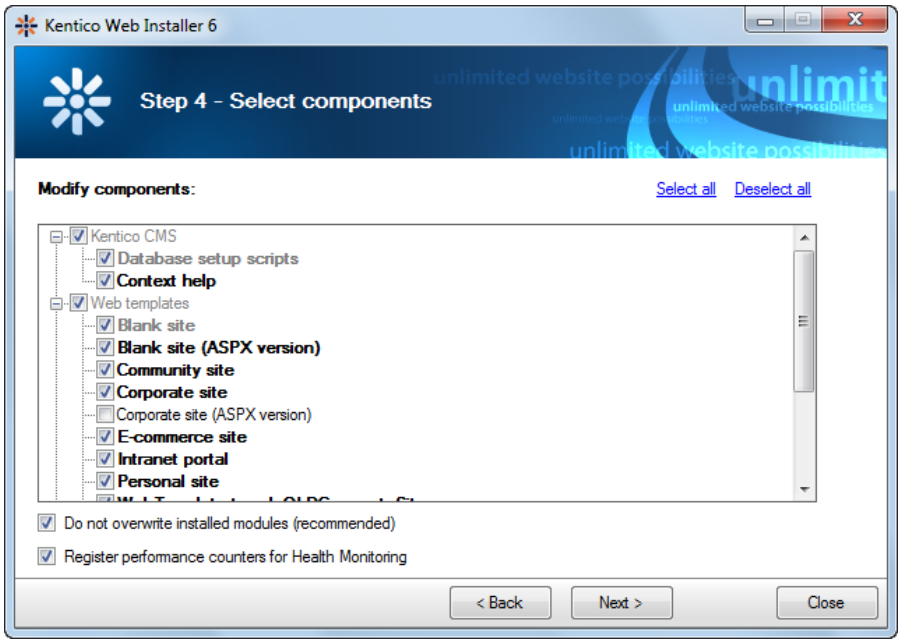


2. The following step displays the component tree. Gray components are a native part of the installation and cannot be removed. Black components are optional. The check-boxes indicate which components are currently installed and you can modify the installation by selecting or unselecting them.

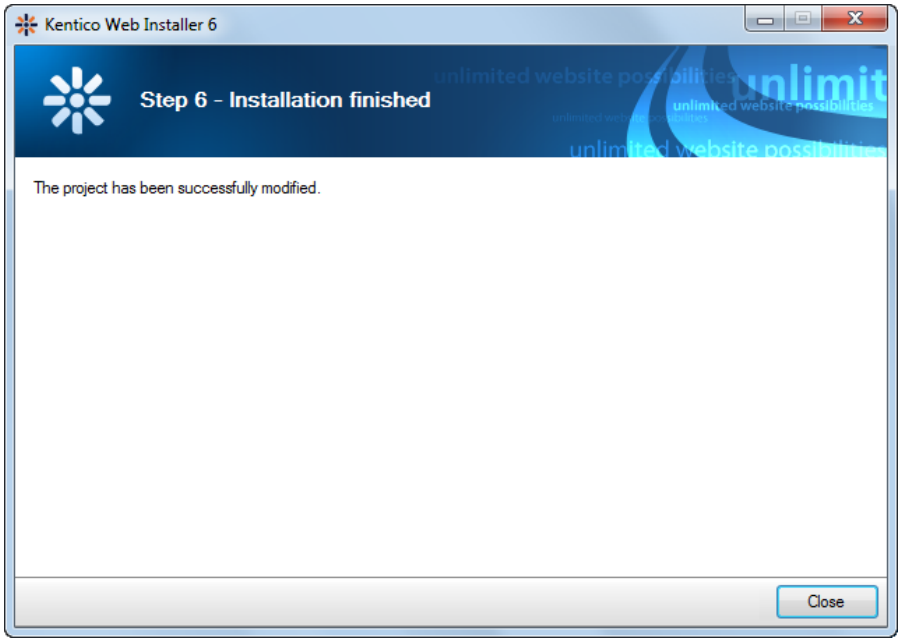
If you enable the **Do not overwrite installed modules (recommended)**, only unselected components will be removed and additionally selected components will be added. If the option is disabled, all enabled components (including *web.config*) will be overwritten by the installation, which means that your 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. Please refer to Modules -> Health monitoring for more details on this feature.

Click **Next**.



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



3.3.3 Database setup

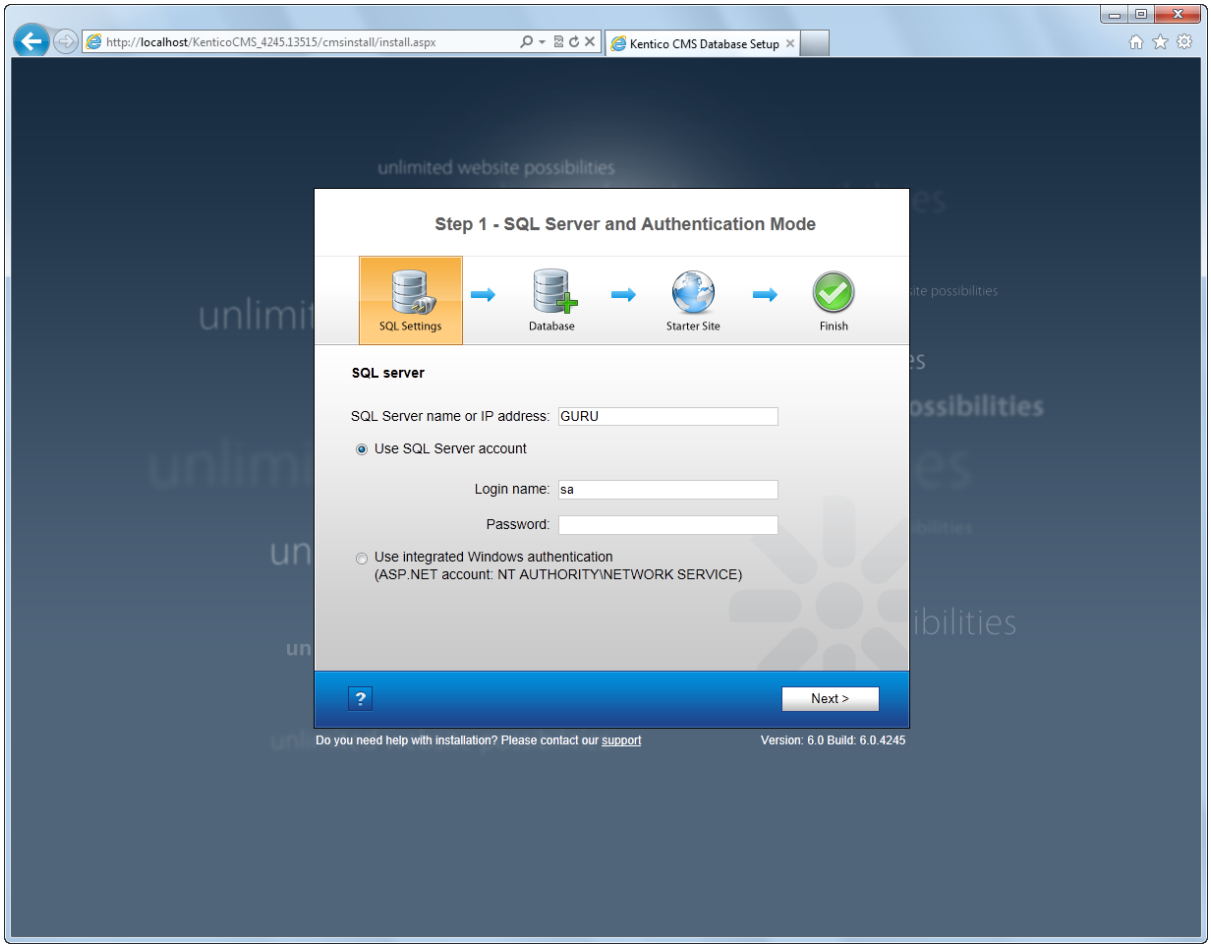
After you finish the Web Installer successfully, you open a web browser with the Database Setup. It will create database tables for Kentico CMS in a specified SQL Server database.

1. In Step 1, you need to 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).

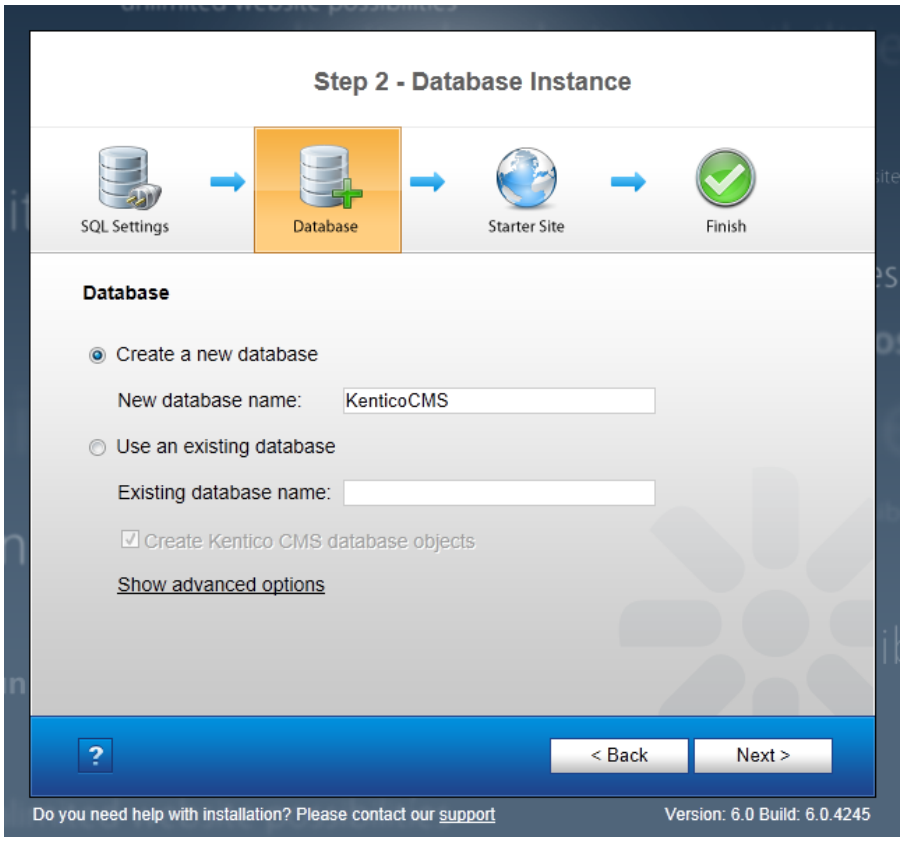
Click the **Next** button.



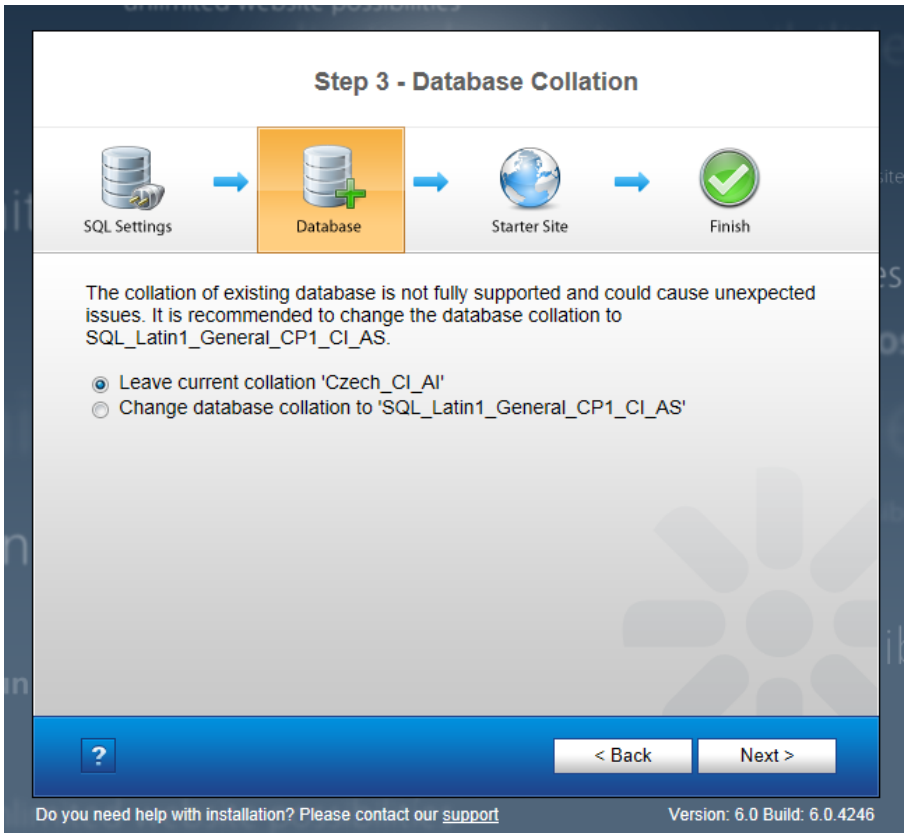
2. Now you can decide if you want to use an **existing database** or **create a new database**. In both cases, you need to enter the name of the database into the appropriate field.

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), you need to uncheck the box. If the database does not contain these objects (typically when you are installing into an empty database), you need to leave the option enabled.

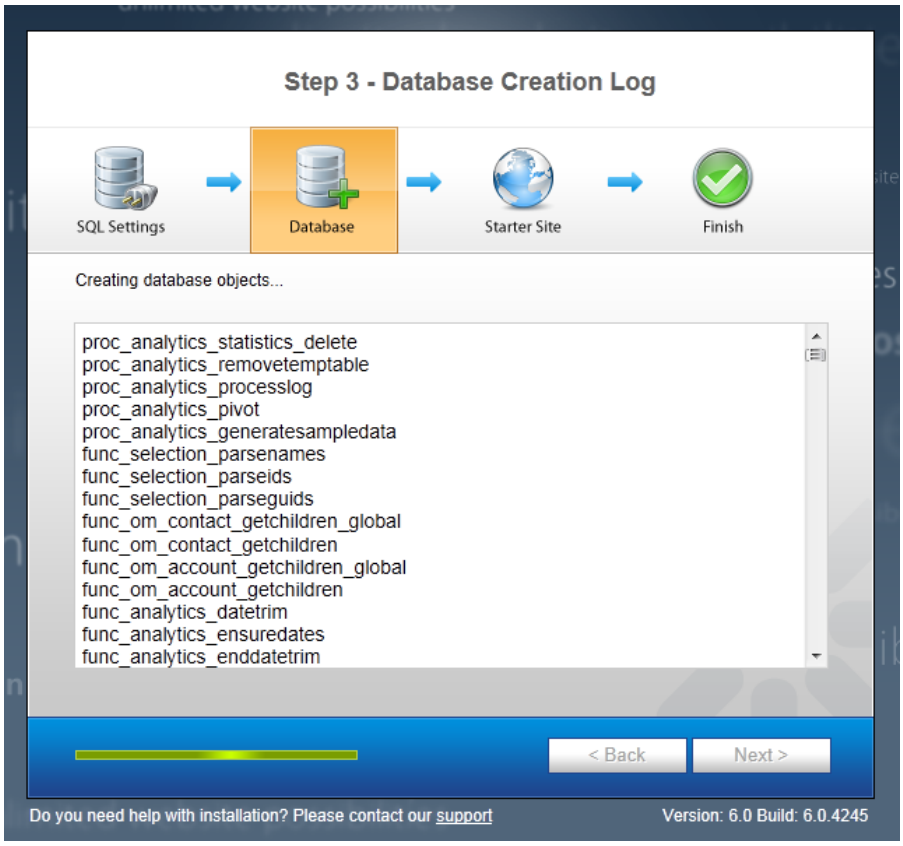
Click **Next.**



3. When using an existing database, you may also come across the following dialog. It is displayed in case that the database collation is different than *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.

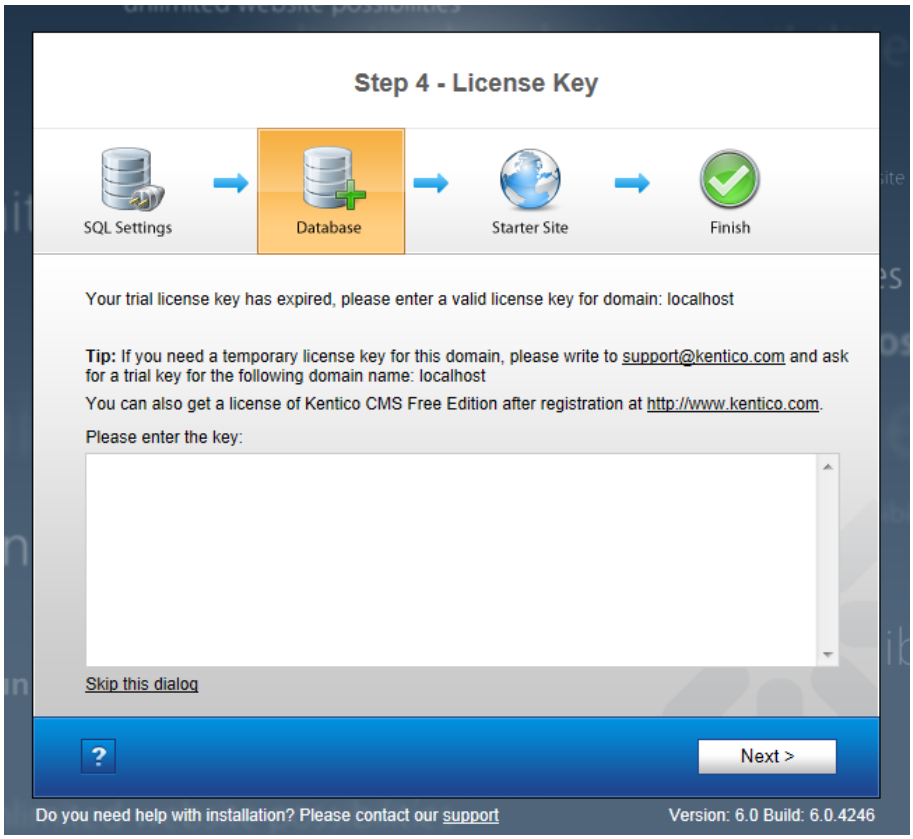


4. A log will be displayed, showing the progress of database creation. After it finishes, a message *Database has been successfully created*. appears at the top of the log and you are moved forward to the next step.



5. If you run Kentico CMS on a domain other than *localhost* or *127.0.0.1*, you will be asked for inserting a license key, since the trial version works only with *http://localhost* and *http://127.0.0.1*. 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 and go to **Site Manager -> Sites -> New site wizard** by clicking **Skip this dialog** at the bottom left part of the dialog.

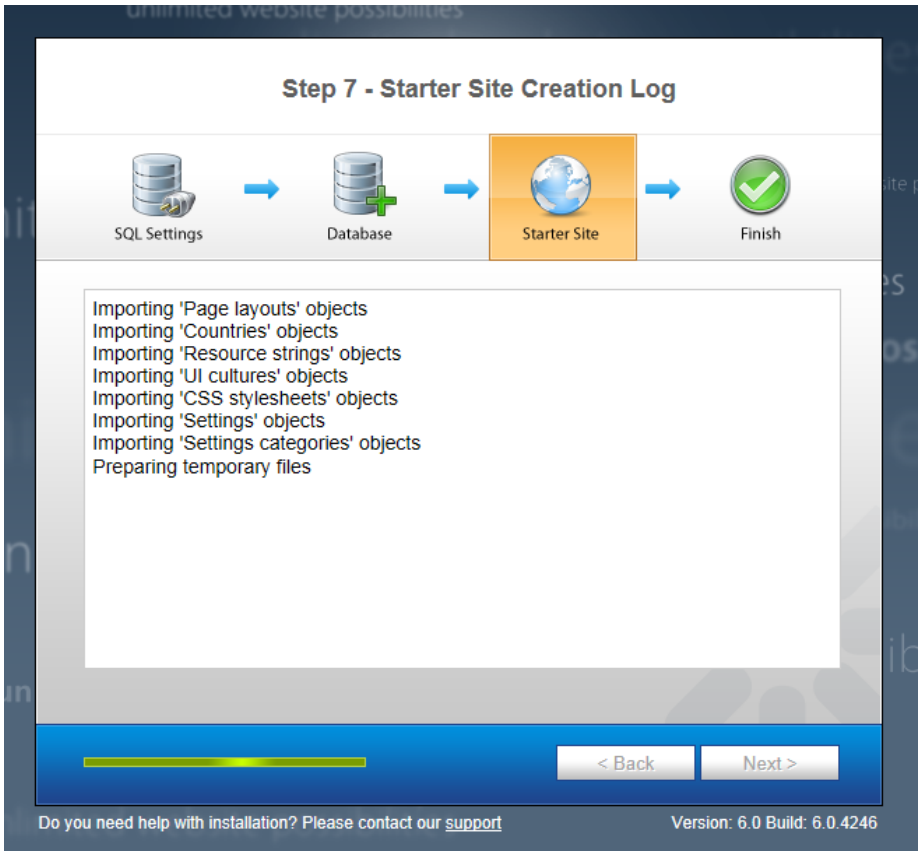


6. On the next screen, you will be offered with the following options:

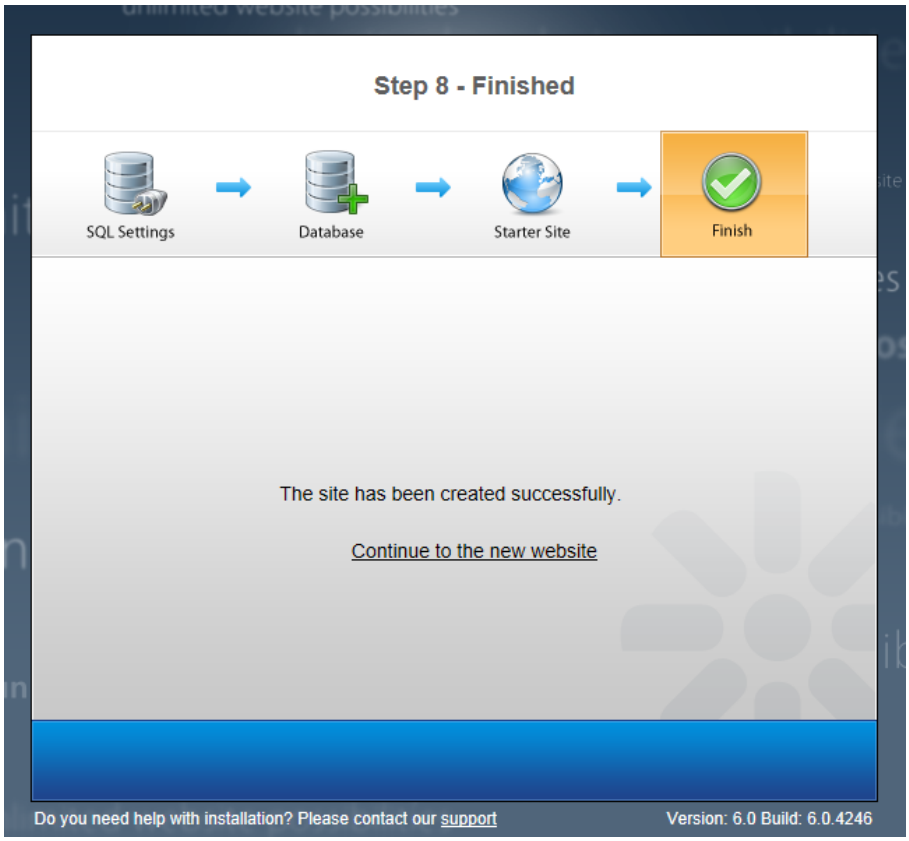
- **Choose a starter site such as:**
  - **Corporate Site** - this option installs the sample corporate site - it is recommended for most users, especially for evaluators
  - **Blank site** - this is a blank site without any content; you will use it to create a new site from scratch
  - **Blank site 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 an existing Kentico CMS website** - use this option if you have already created a website with Kentico CMS and 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 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.

7. A log will be displayed in the next step, showing you progress of website creation.



8. Once the website is created, a confirmation will be displayed and you can follow the link to access the live website.



Default user name and password

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

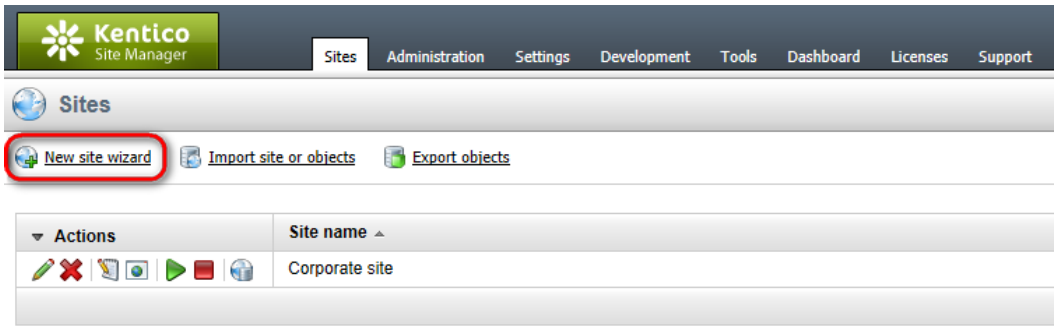
It's highly recommended that you change the password after you finish the installation.

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

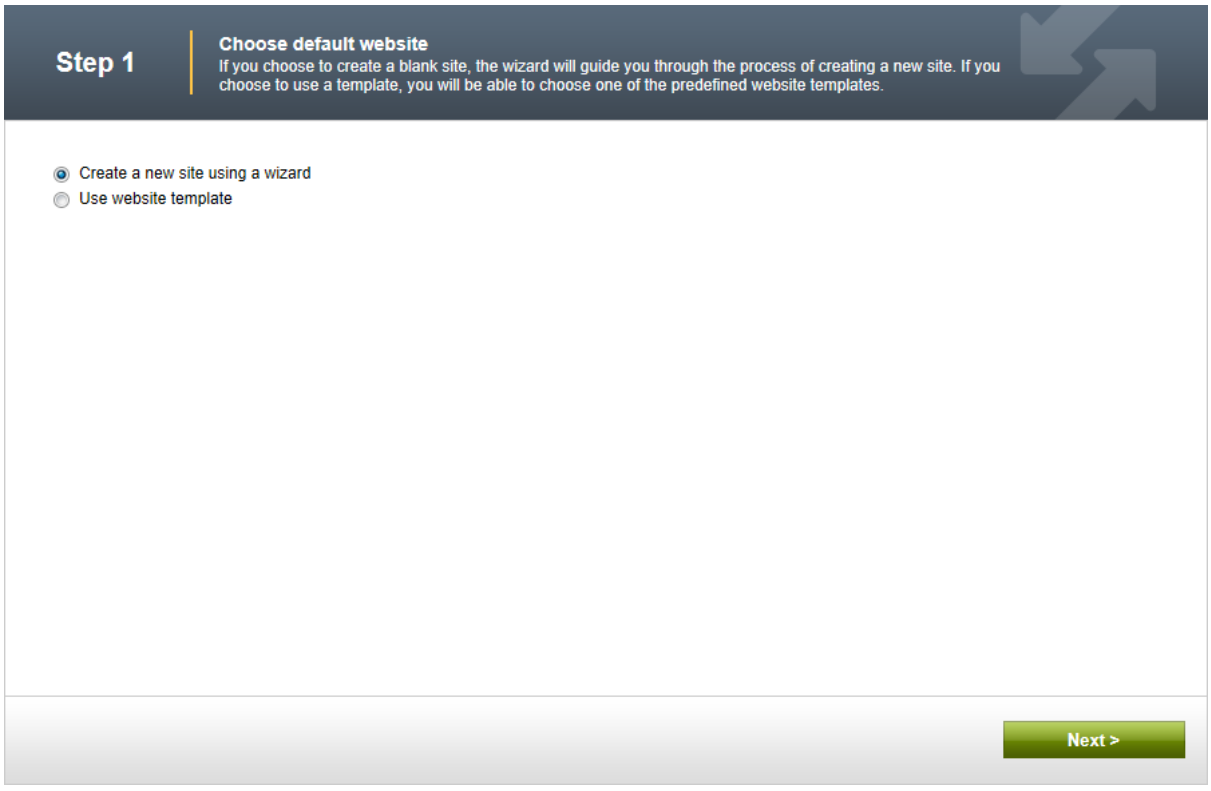
The New site wizard guides you through the process of adding a new website to the system. It is accessible from the *Starter Site* step of the Database setup. You can also launch the wizard from **Site Manager -> Sites**.



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.



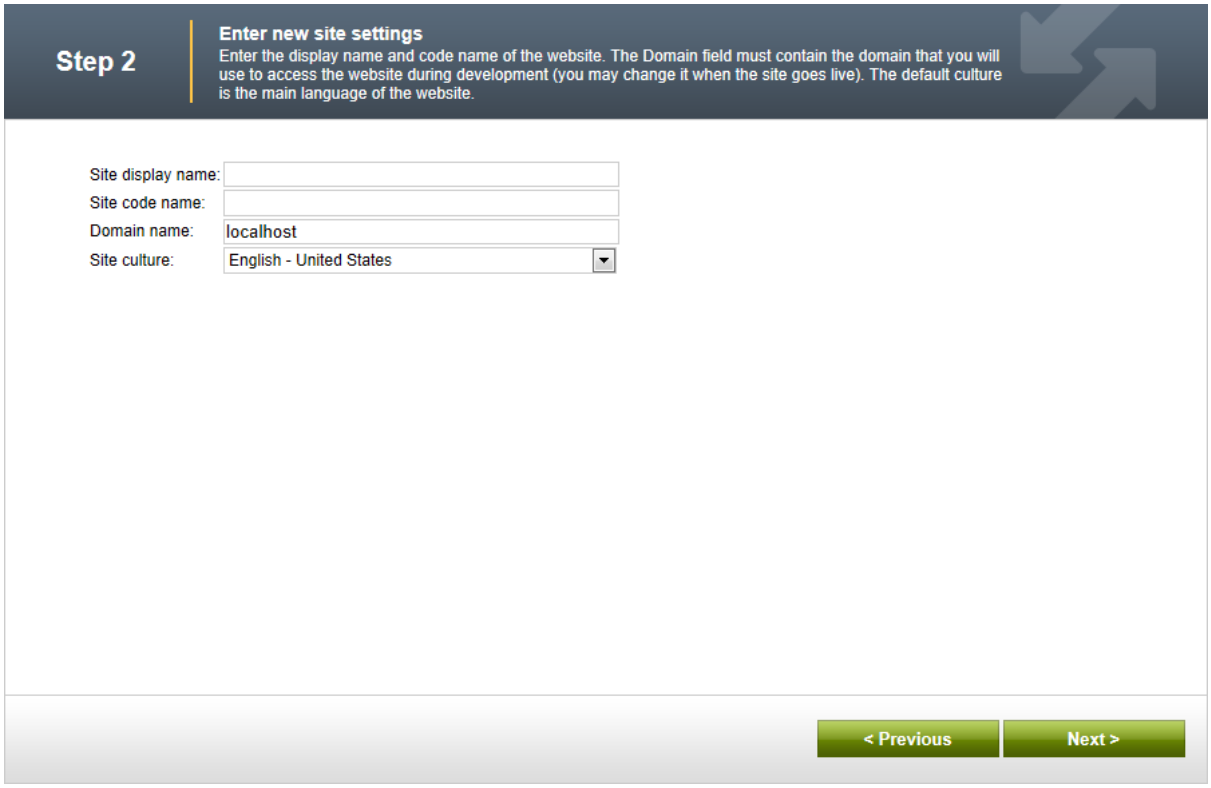
3.3.4.2 New site

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.

1. In the second step, you need to 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** - default culture of the site.

Click **Next** to continue.



2. In Step 3, 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, you will be offered with the following options:

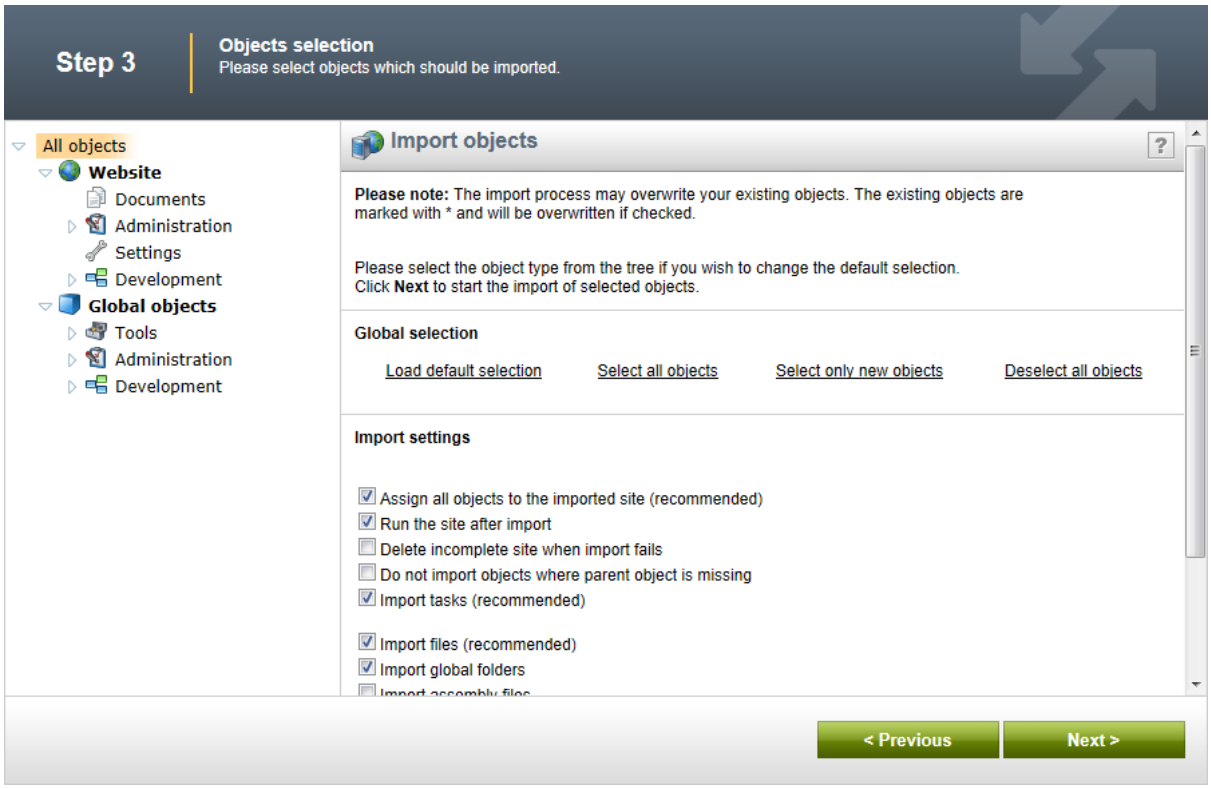
Global selection

- **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.

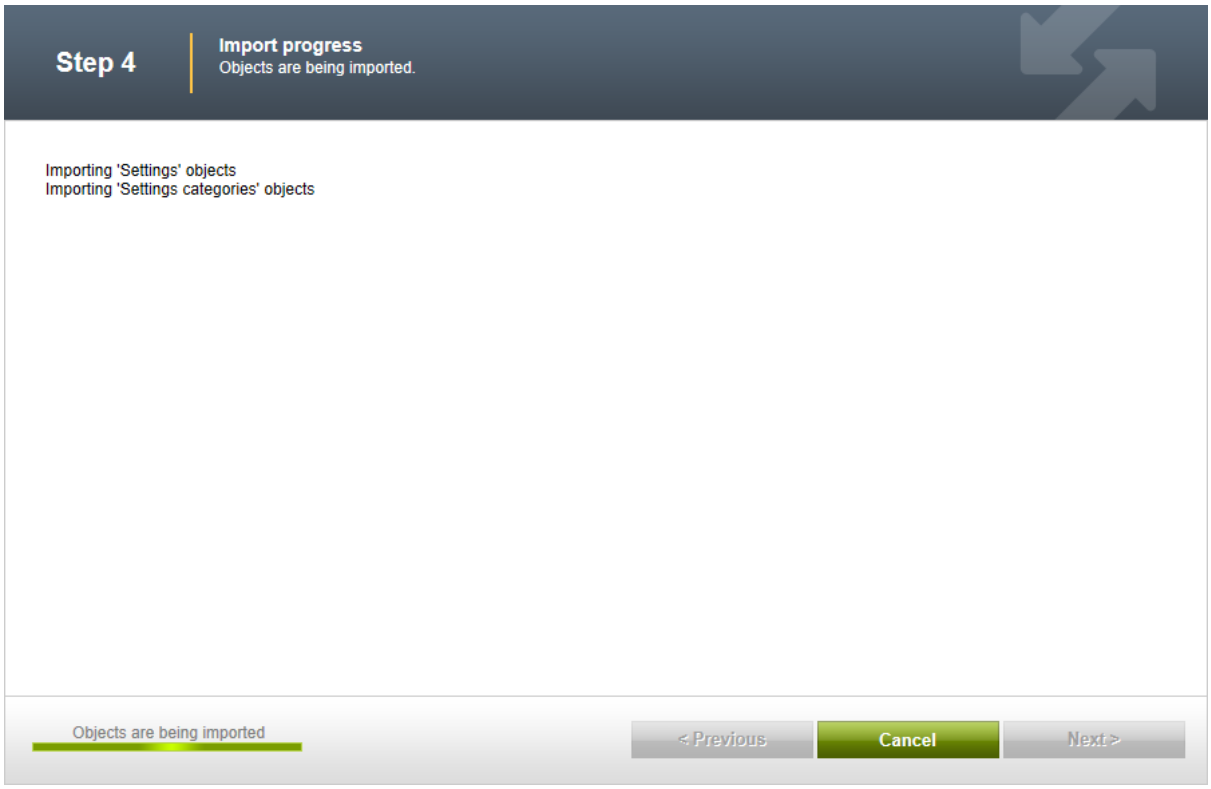
Import settings

- **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

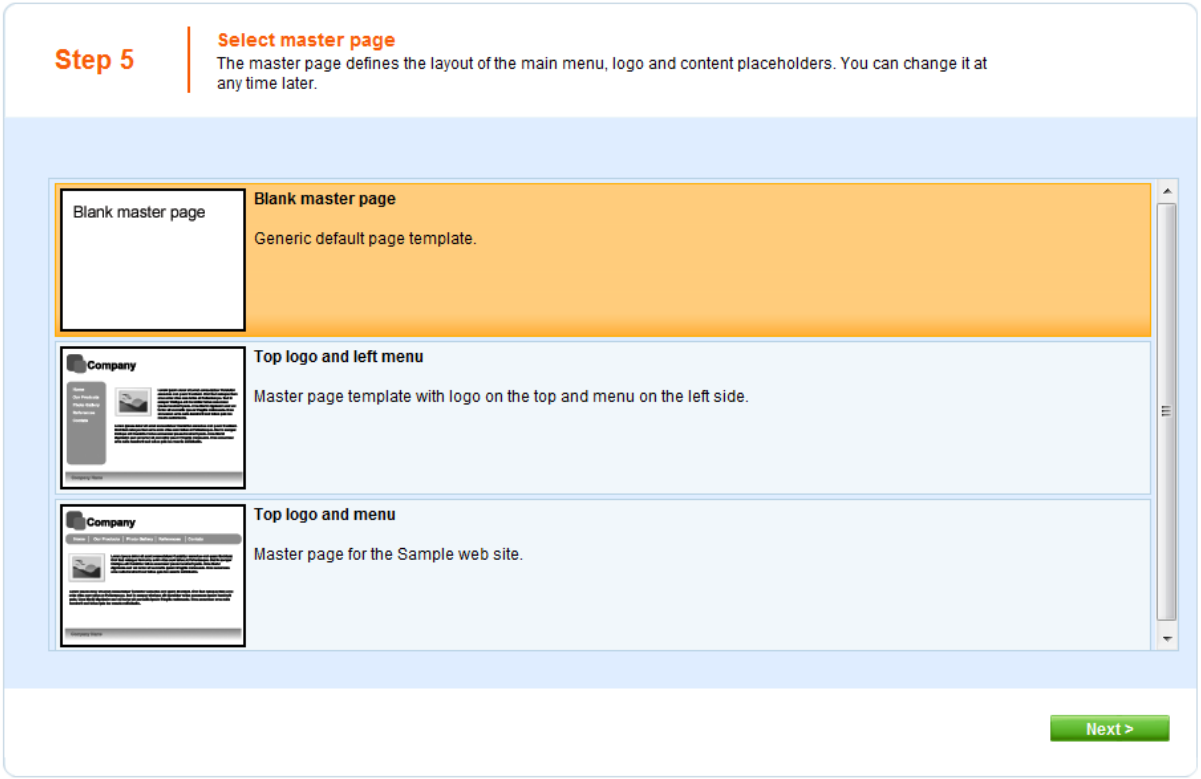
Click **Next** to continue.



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

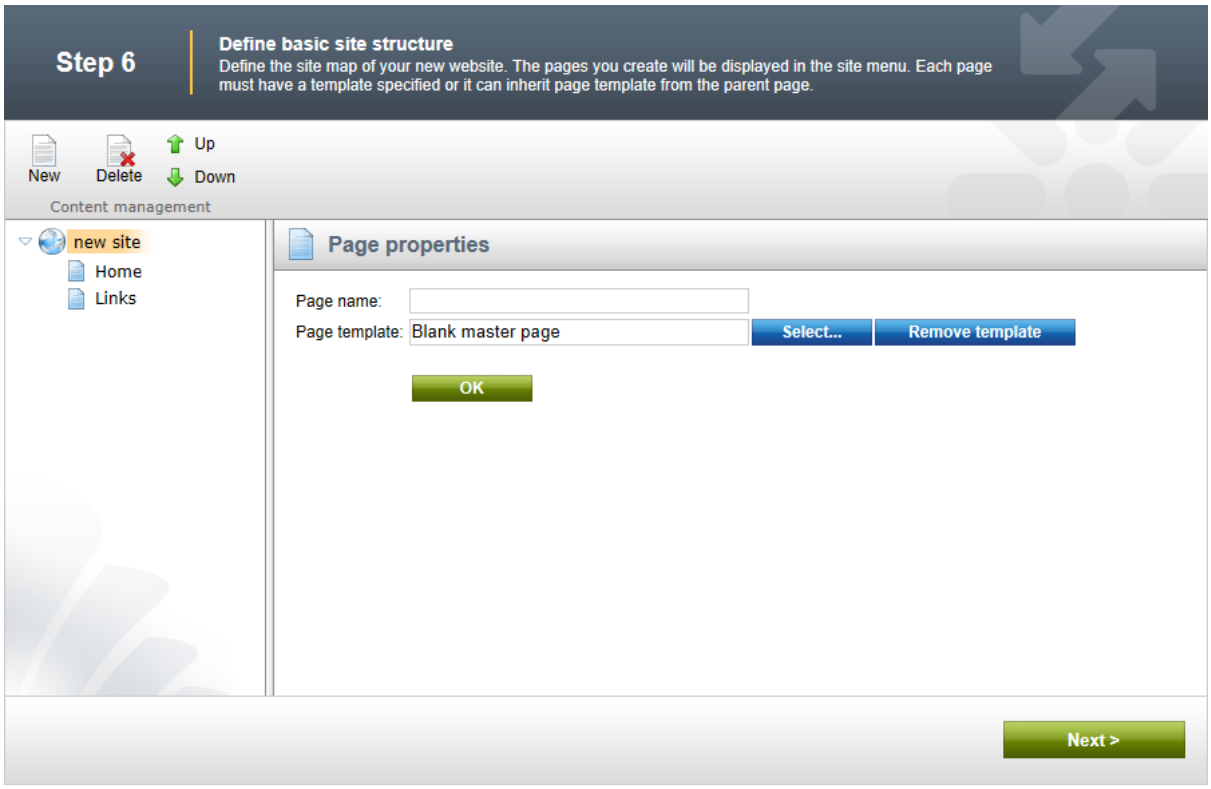


4. In Step 5, you can select the master page layout. This defines the basic visual structure of the website. These settings can be altered any time later, no matter which layout you have selected. Select one of the layouts and click **Next**.

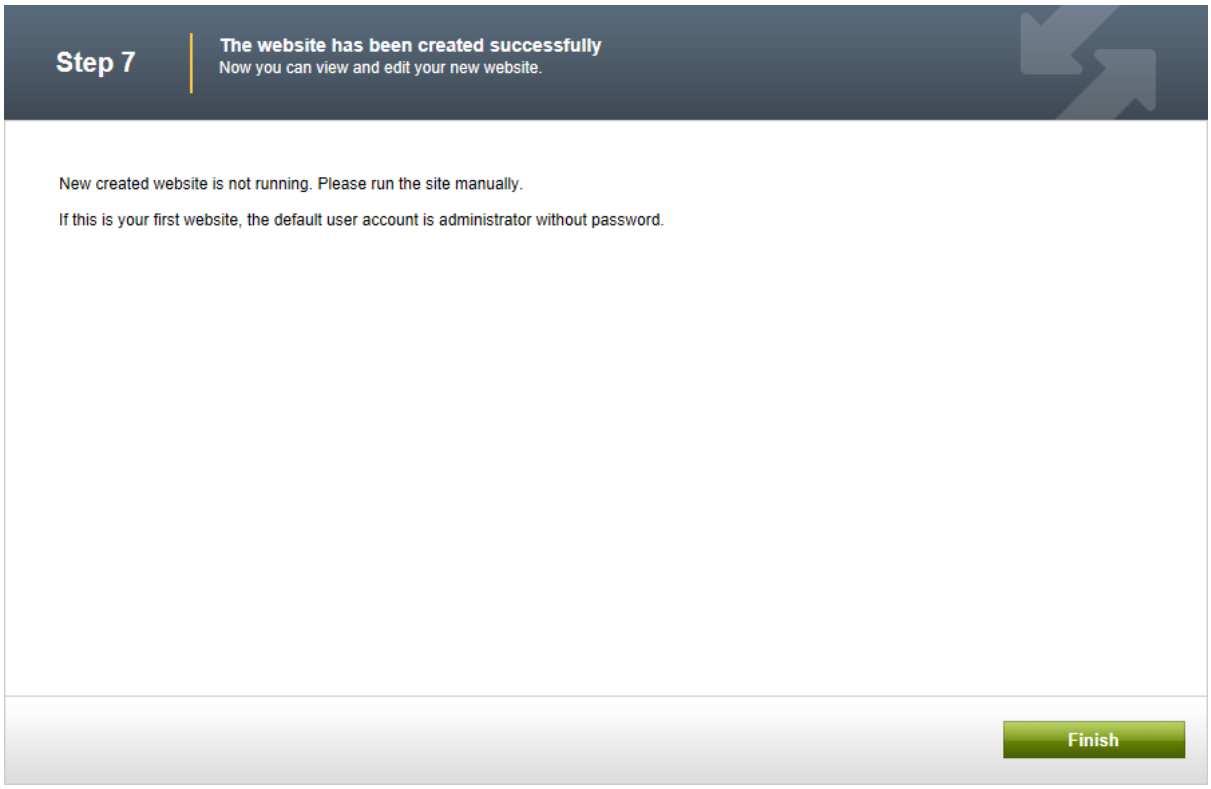


5. Site map of you new website can be defined in Step 6. Select the node of the tree under which you want to place a new page and click **New**. Enter the name of the new page and select a page template which will be used on the page. Alternatively, you can inherit the page template from the parent page 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.



6. If you have reached Step 7, 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**.



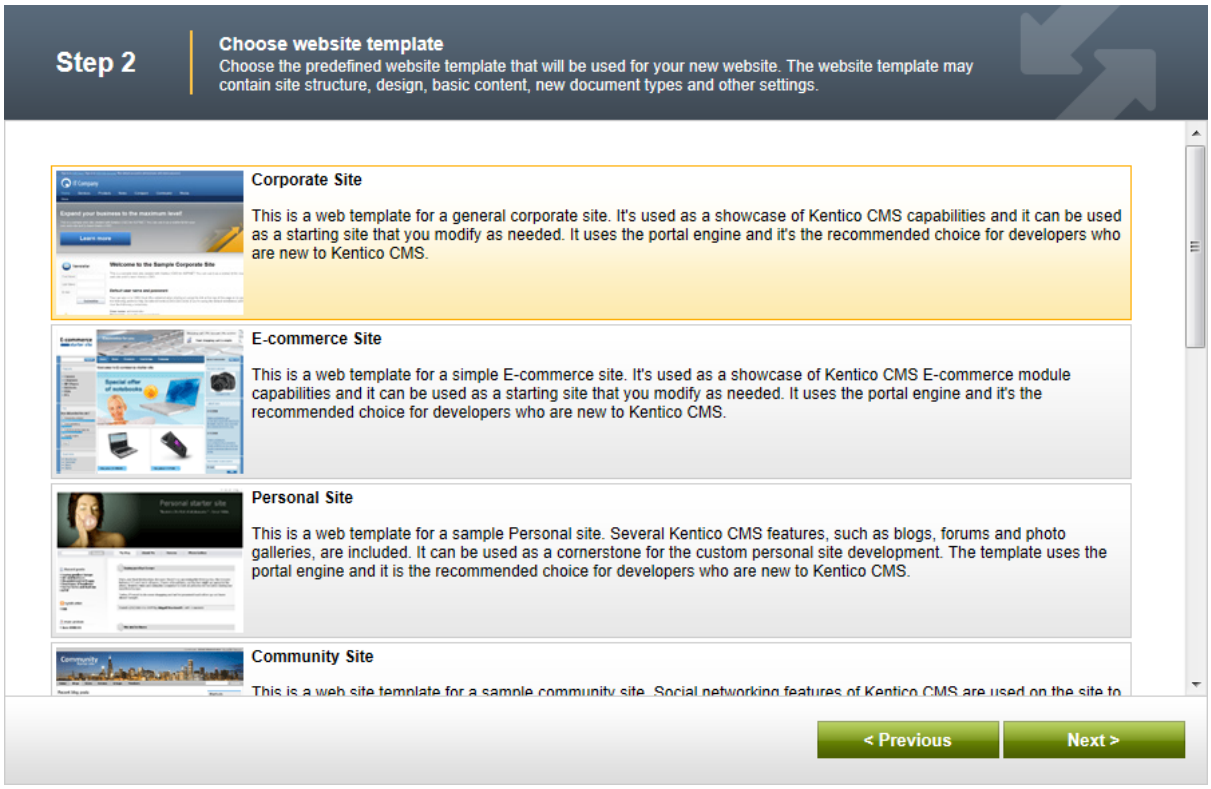
3.3.4.3 Website template

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.

1. In Step 2, 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.

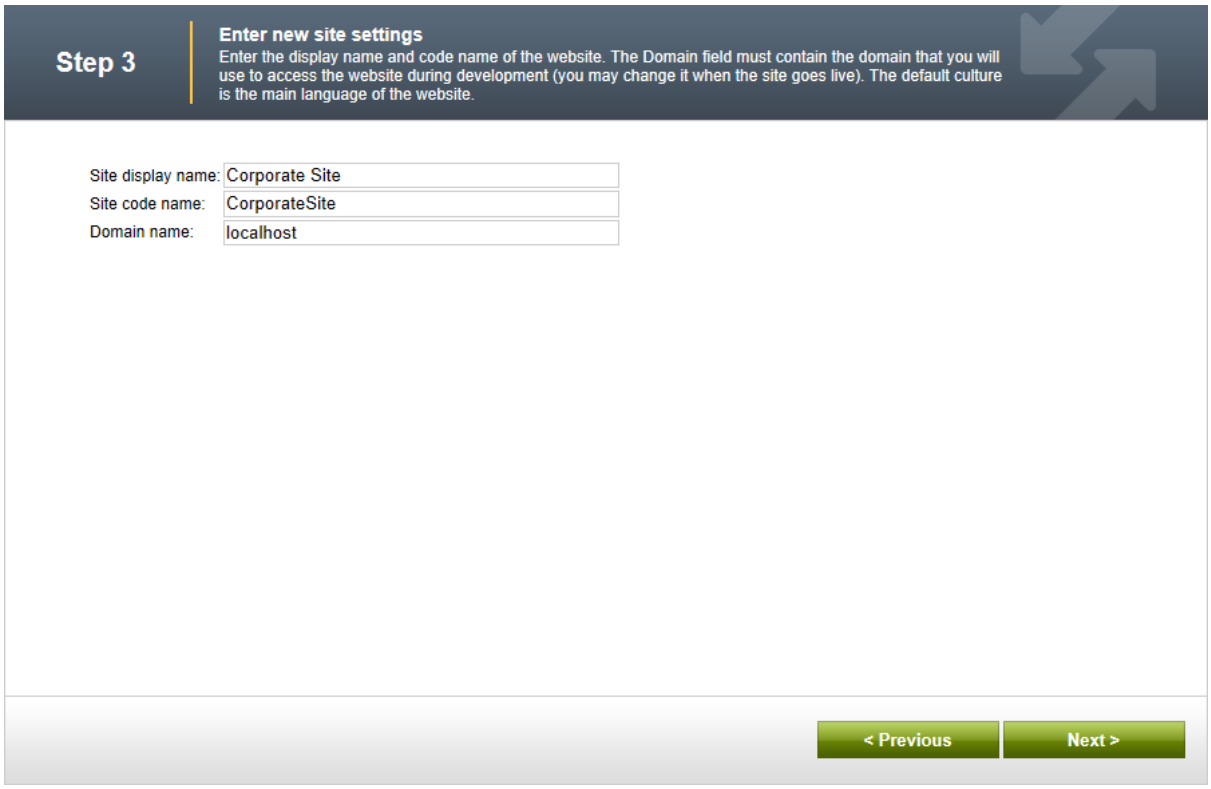
Some of the templates are available in two versions, one using the portal engine and the other using ASPX page templates. Choose one of the listed website templates and click the **Next** button.



2. In this step, 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.



3. In Step 4, 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 to be imported.

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

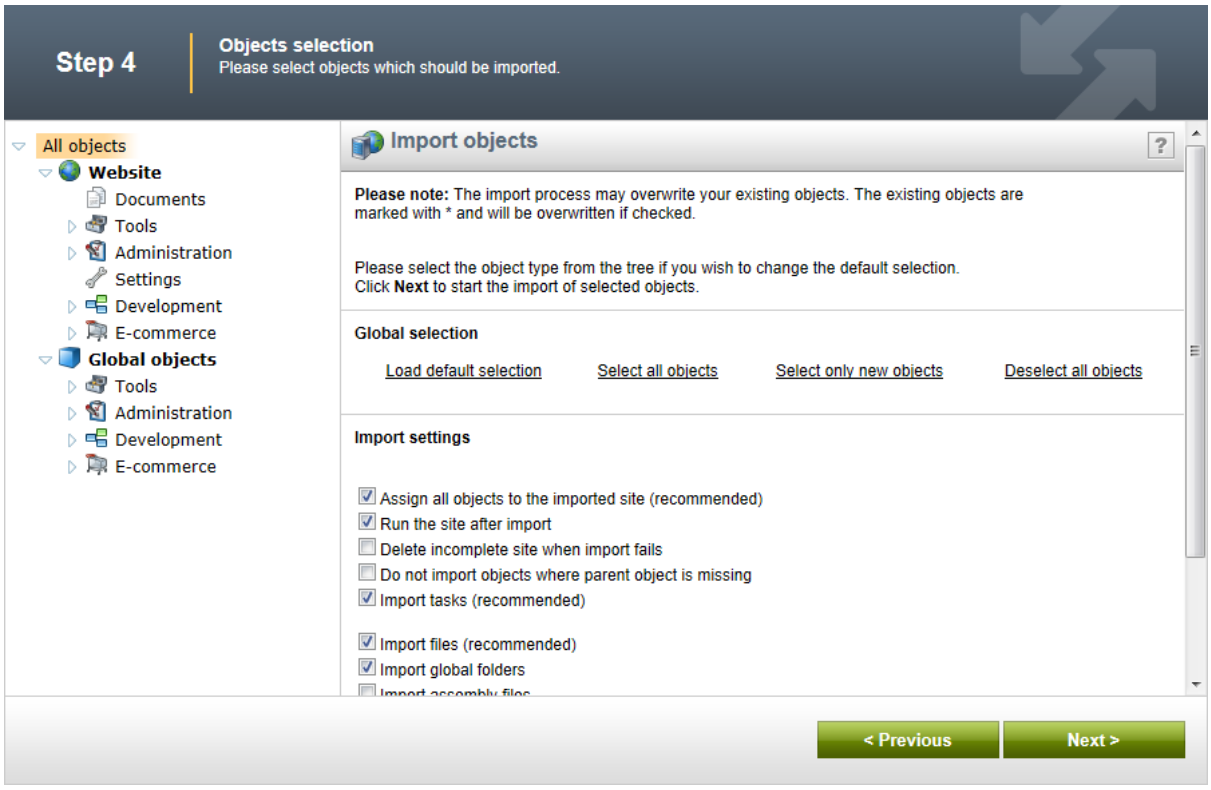
Global selection

- **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.

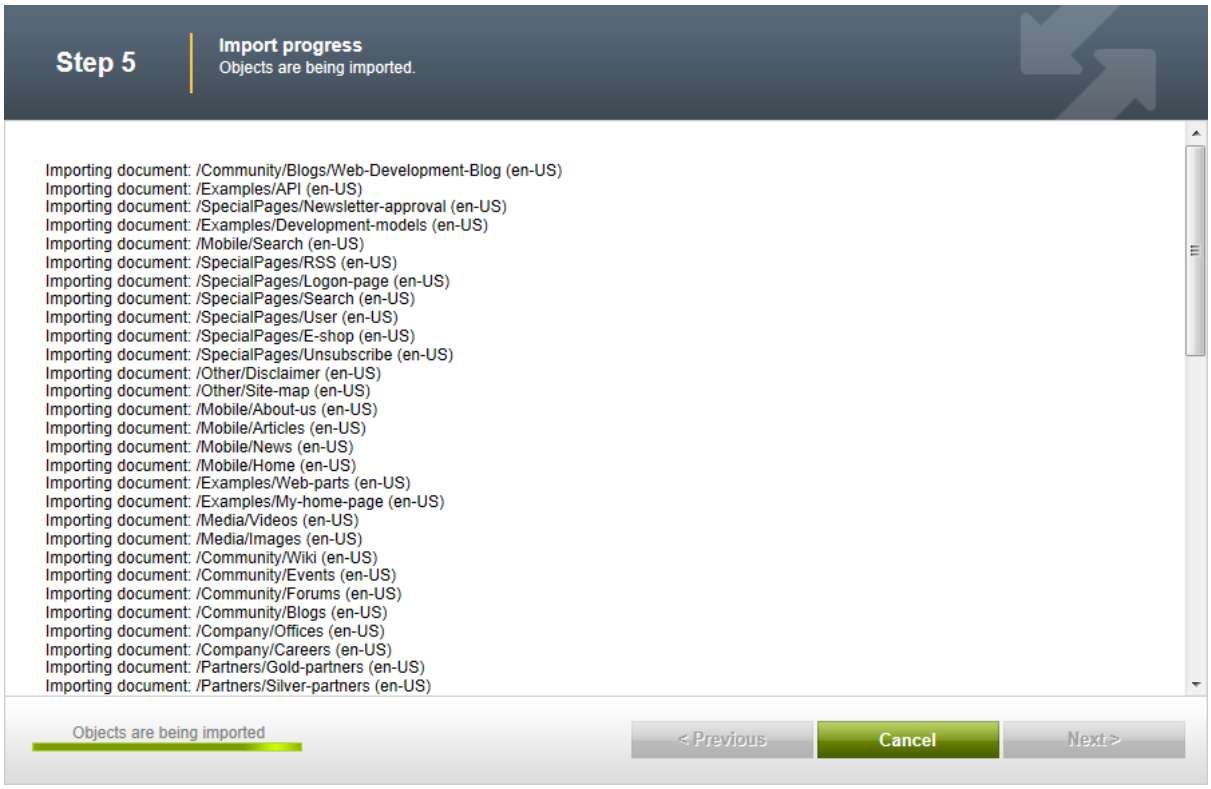
Import settings

- **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

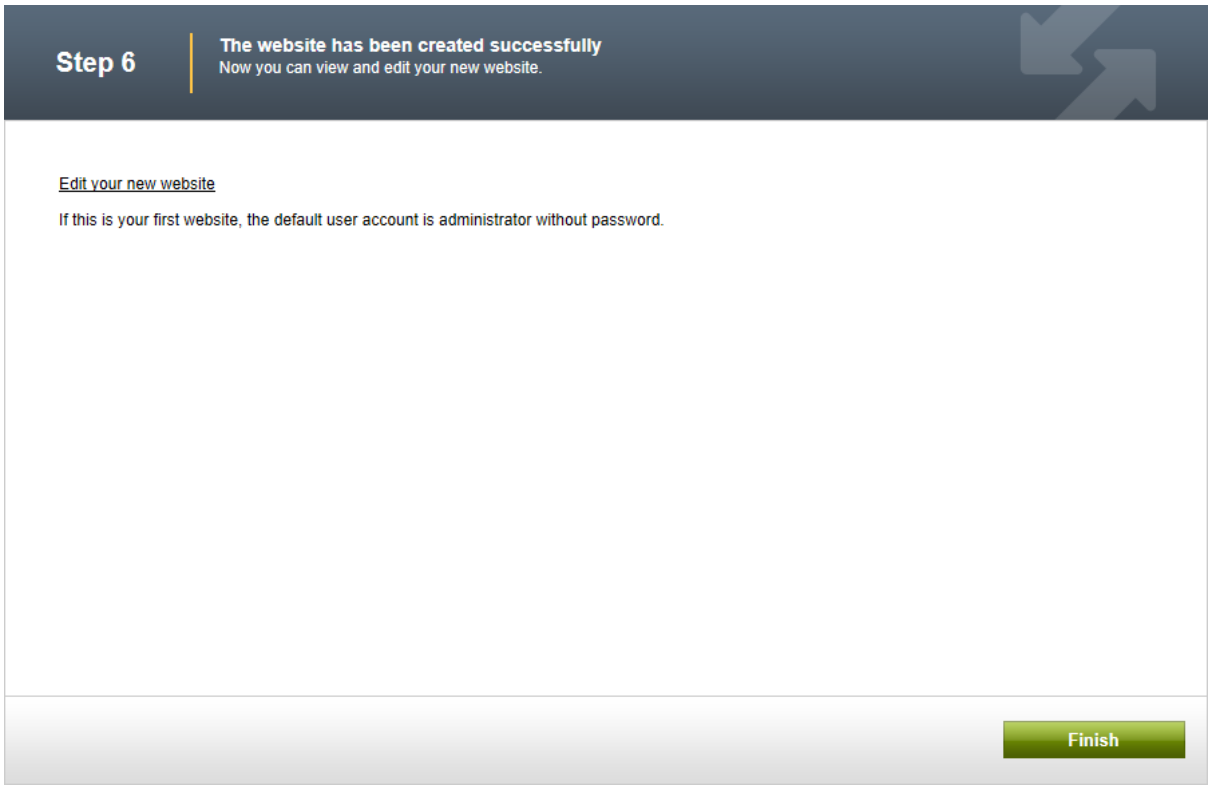
Click **Next** to continue.



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



5. If you have reached Step 6, 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**.



3.3.5 Deployment to the live server

With Kentico CMS, you can easily develop the 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, you need to:

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 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. You will be displayed with Database setup wizard. 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.

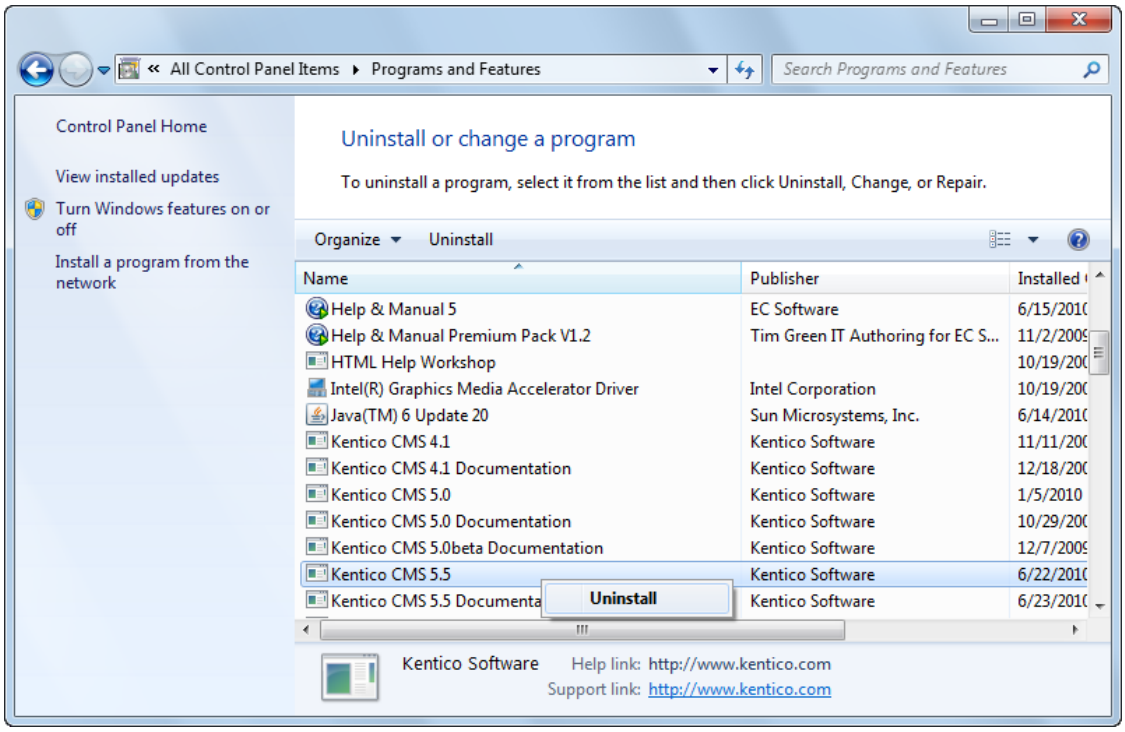
Shared Hosting Environment and Medium Trust Level

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)

3.3.6 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 all 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.

3.3.7 Silent Install

Silent Install is a tool which enables you to install Kentico CMS directly from command line, without any user interface and user interaction involved. All that is needed is to prepare an XML file with configuration of the installation and execute a command from the command line. The main purpose of the tool is the possibility of automated installation of Kentico CMS.

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

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

A ZIP package containing the tool can be downloaded from http://www.kentico.com/downloads/SilentInstall.zip. Detailed documentation of the silent installation procedure together with reference on required format of the configuration XML file can be found in the *Docs* folder within the package.

3.4 Additional configuration tasks

3.4.1 Overview

The following chapters contain description 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.

3.4.2 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.

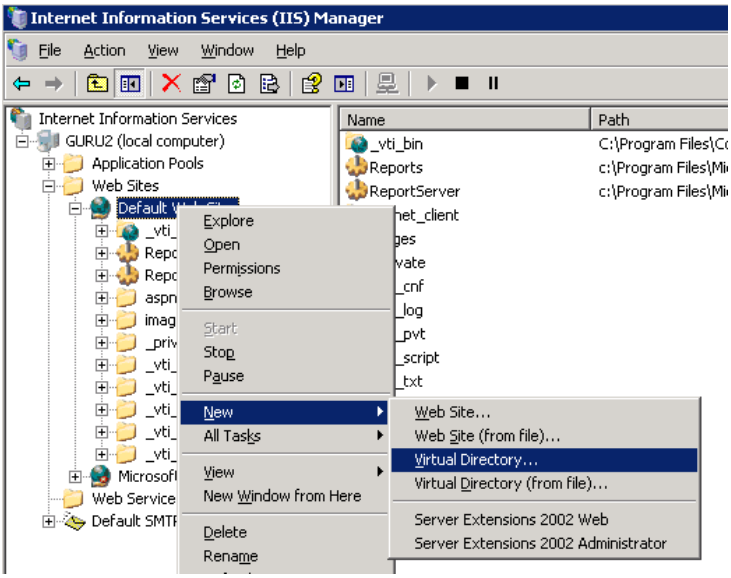
Application root

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**.

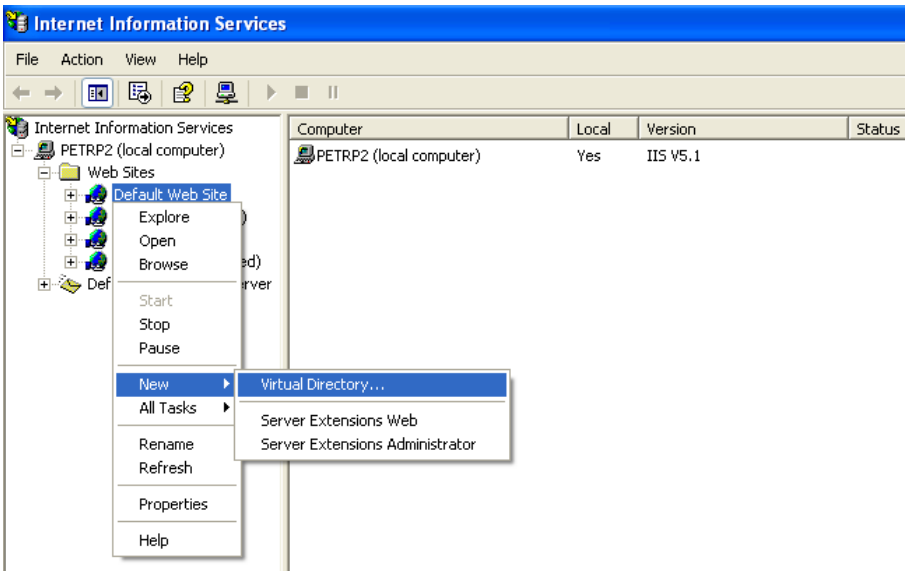
Follow these steps to create and configure a virtual directory on your web server:

Configuring a Virtual Directory

On Windows Server 2003, open **Start -> Administrative Tools -> Internet Information Services (IIS) Manager**. Right-click **Computer -> websites -> chosen website** and choose **New -> Virtual Directory**.



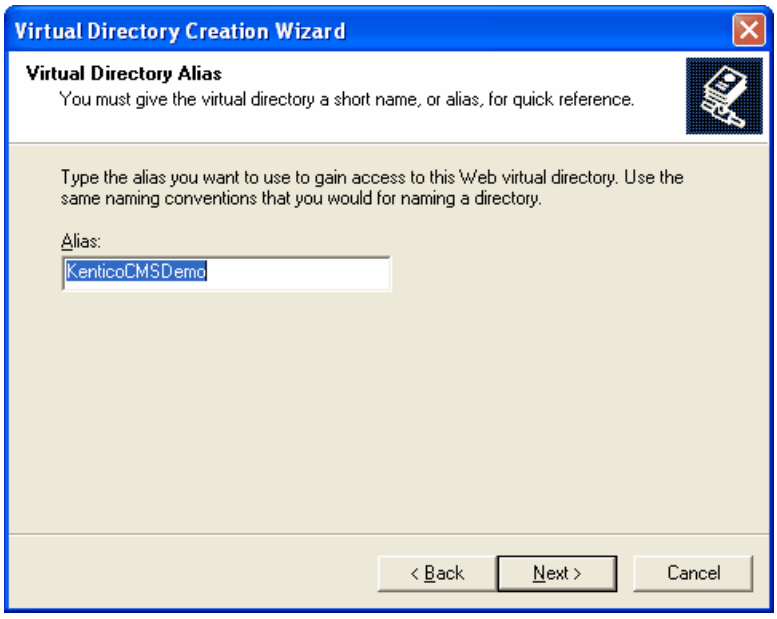
On Windows XP, **open Start -> Control Panel -> Administrative Tools -> Internet Information Services**.Right-click the **Computer -> websites -> Default website** item (or other website if you're running multiple websites on the same computer) and choose **New -> Virtual Directory...**.



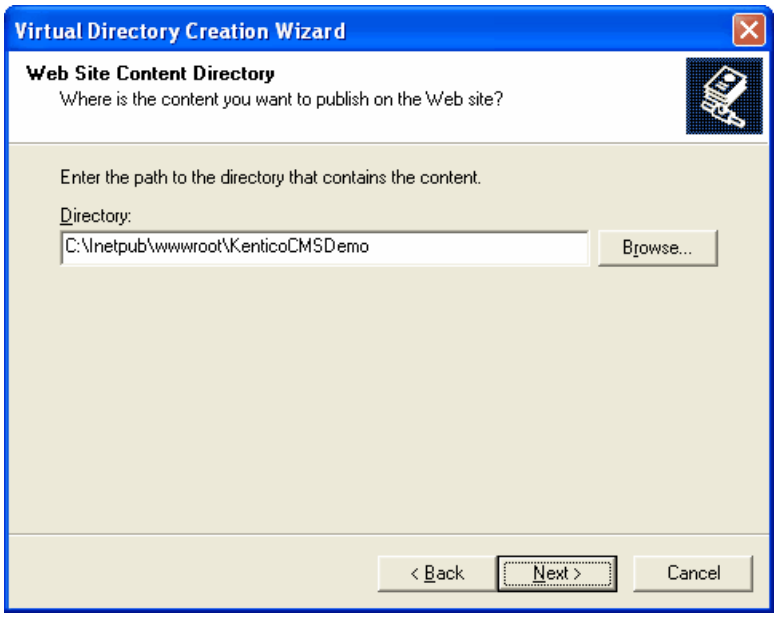
The rest is same for all operating systems:

1. The **Virtual Directory Creation Wizard** appears. Click **Next** on the first screen.

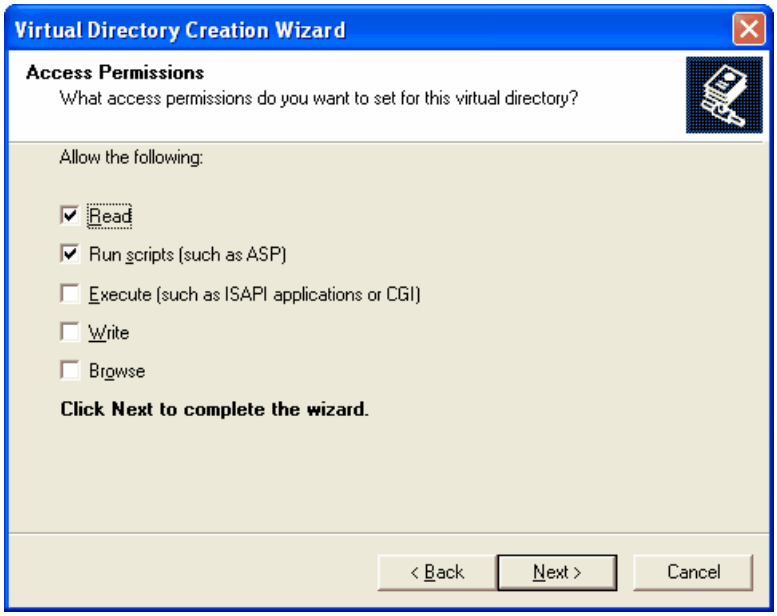
2. Enter the **Virtual Directory Alias**. If you want the website to run as http://localhost/KenticoCMSDemo, you need to enter alias **KenticoCMSDemo**. Click **Next**.



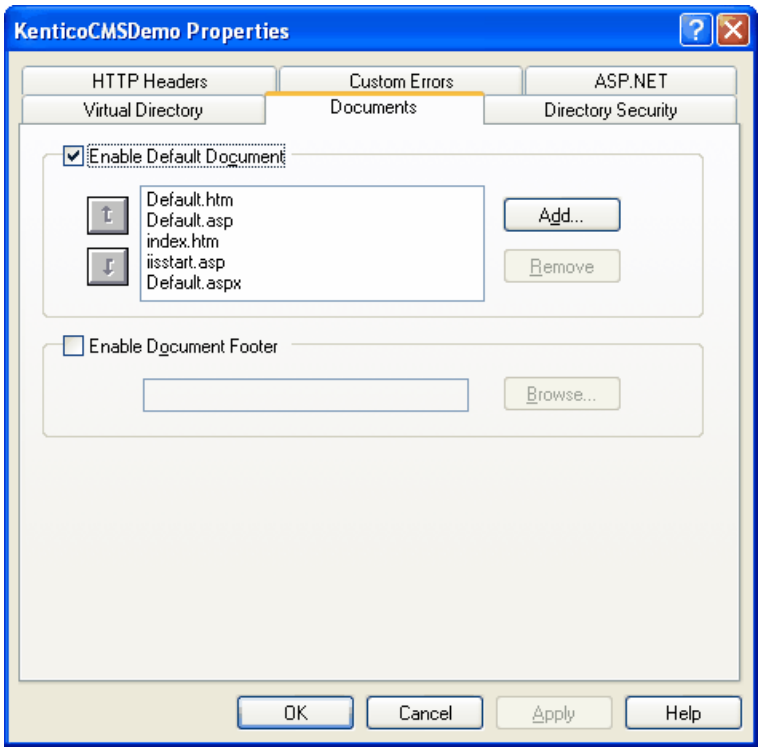
3. Now choose the disk folder on your machine where the Kentico CMS web project files are placed. Please note that you cannot use a remote (shared) disk. The web project files are NOT the files in the C:\Program Files folder - you need to create the web project using the Kentico Web Installer first - see chapter Web Installer for details. Enter the directory path and click **Next**.



4. Now you need to specify the access permissions for the new virtual directory. You need to choose at least **Read** and **Run scripts** and it's recommended that you do NOT allow any other options. Click **Next** and click **Finish**.

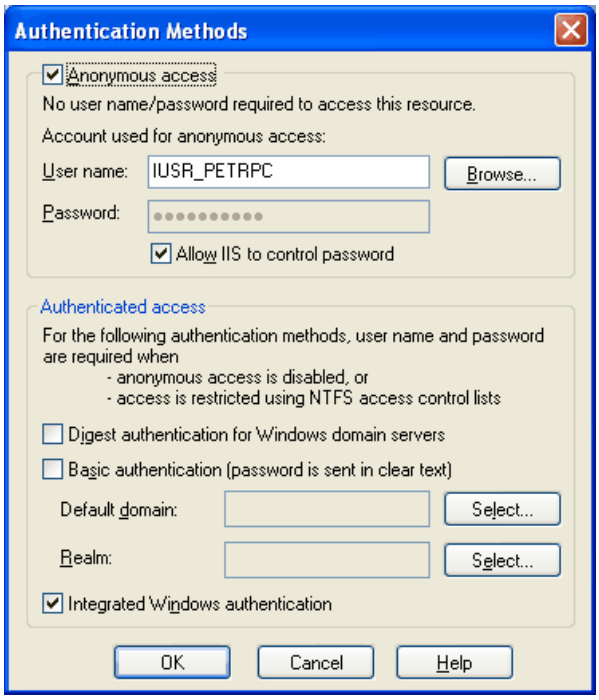


5. Right-click the newly created virtual directory and choose **Properties**. Choose the **Documents** tab and make sure the **Enable Default Document** box is checked and the list contains the **Default.aspx** file.



6. Choose the **ASP.NET** tab and make sure the **ASP.NET version is 2.0.50727** (the last number may be different). 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.

7. 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.



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

3.4.3 Configuration for Medium Trust environment

This chapter describes the procedure of running Kentico CMS in a Medium Trust Level environment and the steps required to configure the system for it. It also describes the Precompilation/Deployment procedure and requirements.

Medium trust level

The .NET Framework comes with a batch of predefined code access security policies, categorized into several trust levels, which define 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 provided with the Microsoft .NET Framework. 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** - provides virtual objects (layouts, transformations) from the database.
- 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.

Virtual path provider

This library provides the interface for virtual objects stored in the database that can be compiled, such as document transformations and page layouts. The system references the files with a virtual path, and the VirtualPathProvider provides the control code to the compiler.

Since the virtual provider cannot run in a medium trust environment (requires AspNetHostingPermission with "high" trust level), you need to store the physical files to the file system. You can save all the virtual objects to the file system in **CMS Site Manager -> Administration -> System -> Deployment** interface by clicking the **"Save all virtual objects to disk"** button. This will make copies of the virtual objects in the following folders:

- **~/CMSTransformations** - contains transformation files for documents and custom tables
- **~/CMSLayouts** - contains shared page layouts
- **~/CMSTemplateLayouts** - contains custom page template layouts
- **~/CMSAdhocTemplateLayouts** - contains custom ad-hoc page template layouts

Please note that these files are just copies of the actual virtual object and will be used by the system only if the VirtualPathProvider cannot start. Also that the changes to the objects through the administration interface will not affect these files until you save all the objects to the disk again.

Limitations

In the medium trust environment, the VirtualPathProvider is stopped automatically. When VirtualPathProvider is stopped, **you cannot edit transformations and layouts through the user interface** without saving them on the disk again.

In the portal development model, you cannot use **custom web part code** (**Web part properties –> Code tab**). If you need to add custom code on the Code tab and run the website in a medium trust environment, you need to create user controls, place web parts into the user controls and add your custom code to the web parts. Then, you can place the user controls on the page using the General/User control web part.

Please note that **you shouldn't run the system in medium trust while developing the website**. You should use this trust level only for the live website.

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.

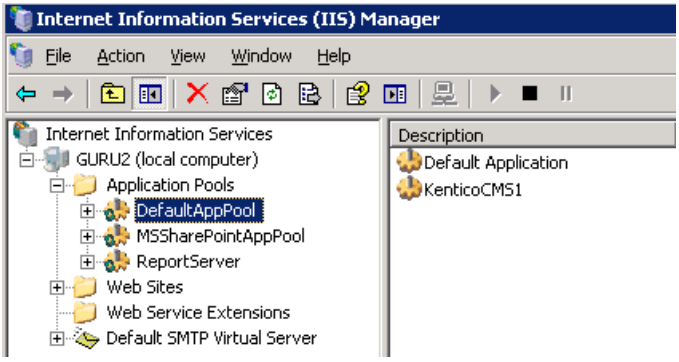
Running the website

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.

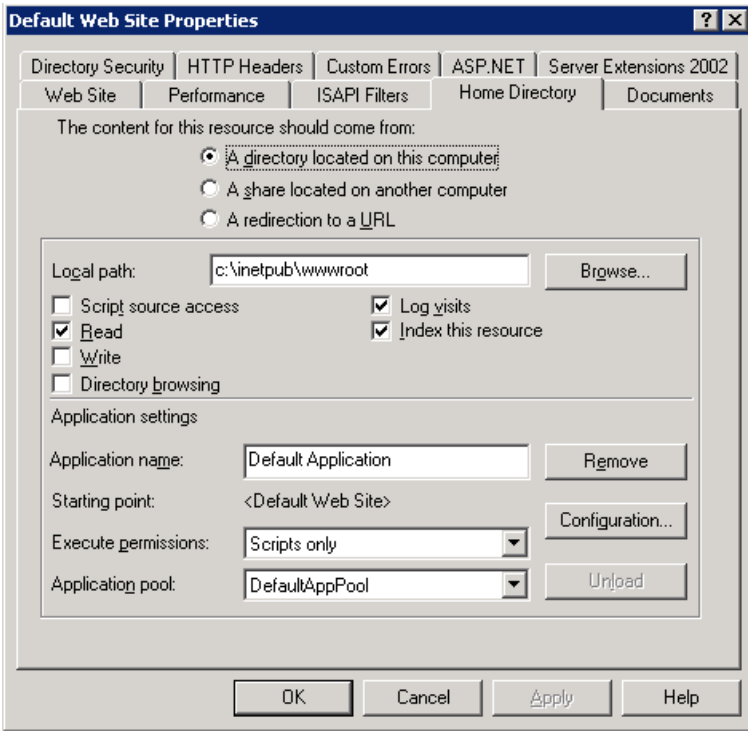
3.4.4 Configuring Application Pools

Application Pools provide you with additional level of website management. They are supported only on Windows Server 2003/2008/Vista/7. You can configure them in **Start -> Control Panel -> Administrative Tools -> Internet Information Services (IIS) Manager -> Computer -> Application Pools.**

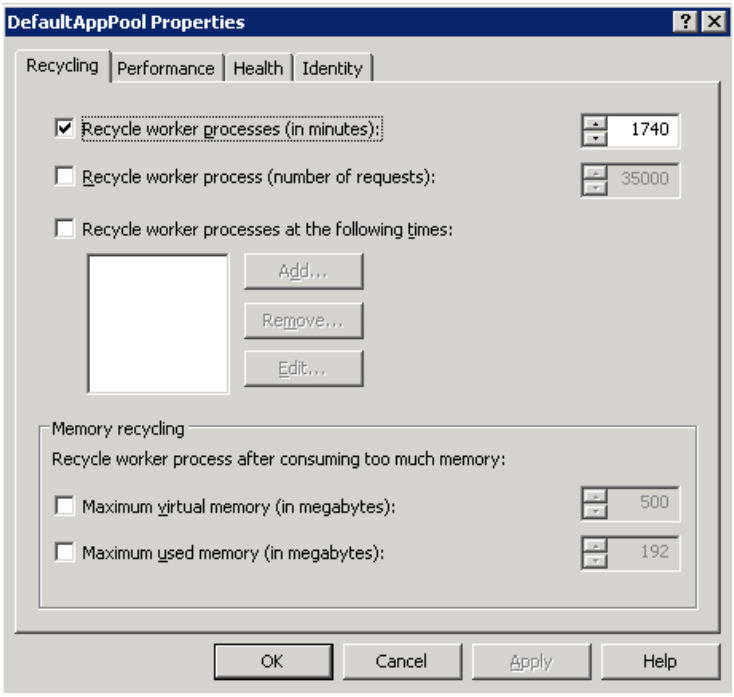


The application pools allow you to group applications (websites) into pools that share server resources using the same rules. This chapter contains recommendations on how to configure the website application pool for Kentico CMS.

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

1. 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.

2. 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.

3. 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.

4. 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).

5. 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**.

6. 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.

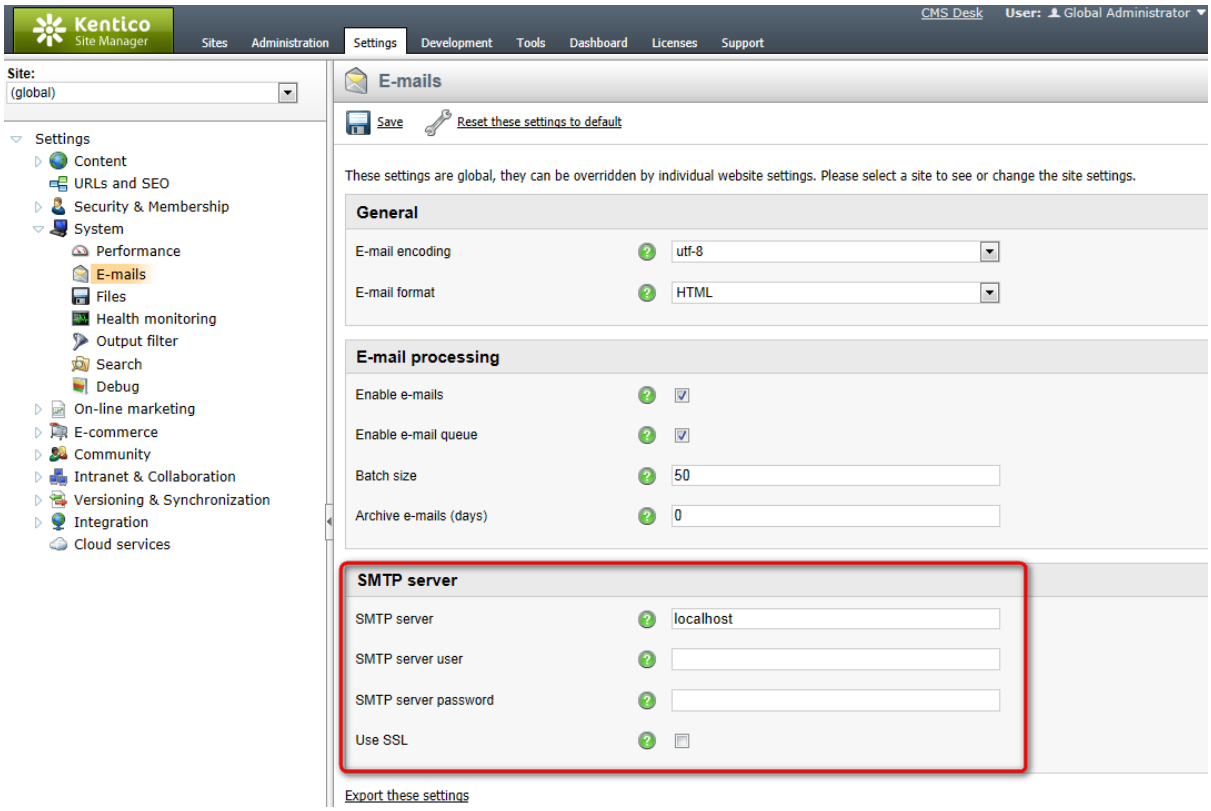
7. Kentico CMS **does not support Web garden**. Therefore, the **Maximum number of worker processes** has to be set to **1**.

3.4.5 SMTP server configuration

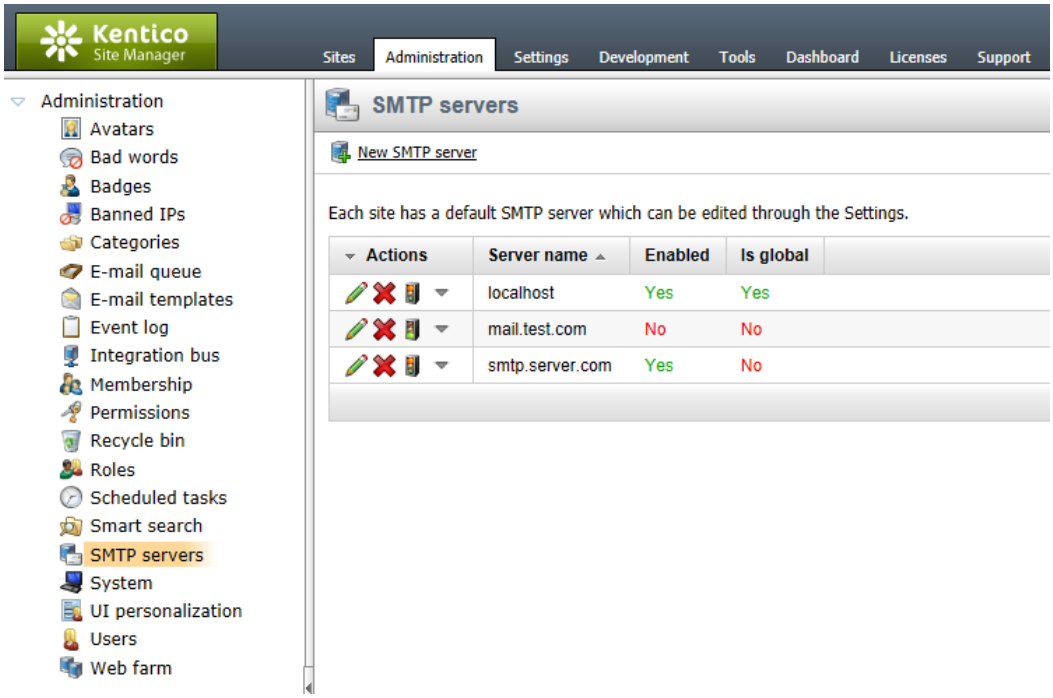
Kentico CMS includes many features and modules that utilize e-mail messages as part of their functionality, including automatic notifications, various types of confirmation messages, newsletters and more. To allow the application to send out e-mails, you need to register and configure SMTP servers for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

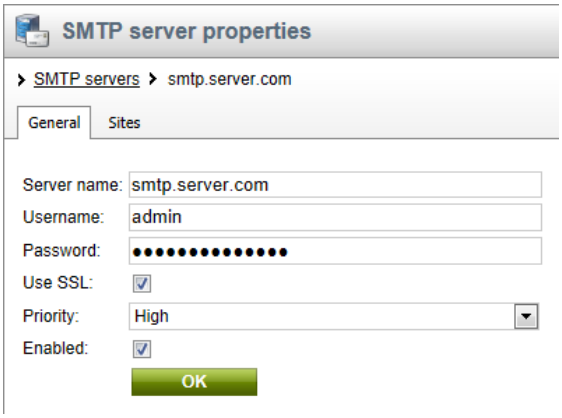


In addition to the default server, you may add any number of servers in **Site Manager -> Administration -> SMTP servers**. Having multiple SMTP servers available for a website allows it to send out a greater volume of e-mail traffic, since the extra servers will be used whenever the default one is busy.



Servers can be defined by clicking the **New SMTP server** link. 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.

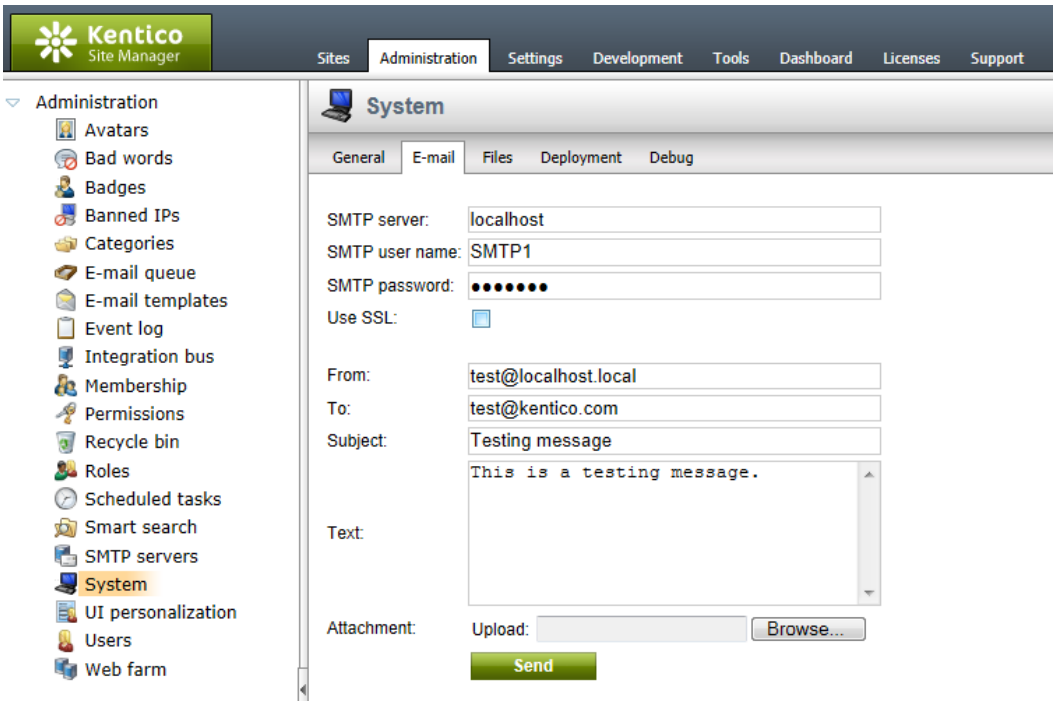
This queue is processed periodically (every minute by default) and the stored e-mails are asynchronously distributed to the appropriate SMTP servers. First, a predefined amount of e-mails is loaded from the queue as a batch. The system then goes through the available SMTP servers in the order of their 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.



3.4.6 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 Kentico Web Installer (you can find it in Start -> All Programs -> Kentico CMS).
3. Choose 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, you will need to create a new virtual directory as described in Creating a virtual directory.
5. **Open web browser** and navigate to your website. You will be displayed with **Database Setup Wizard**.
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 either to 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 topics: Deployment to the live server, Configuration for Medium Trust environment

3.4.7 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 **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 Kentico Web Installer (you can find it in Start -> All Programs -> Kentico CMS).
3. Choose 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, you will need to 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 web browser** and navigate to your website. You will be displayed with the first step of the Database Setup.
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 either to create a **new website or import your existing** Kentico CMS website.
9. After you create a new website, you will need to save all virtual objects to the disk. Go to **Site Manager -> Administration -> System -> Deployment** and click **Save all virtual objects to disk**.

If your web application is not allowed to write on the disk, you will need to install Kentico CMS on your local machine, save all virtual objects to disk and then copy the web project over FTP to your server.

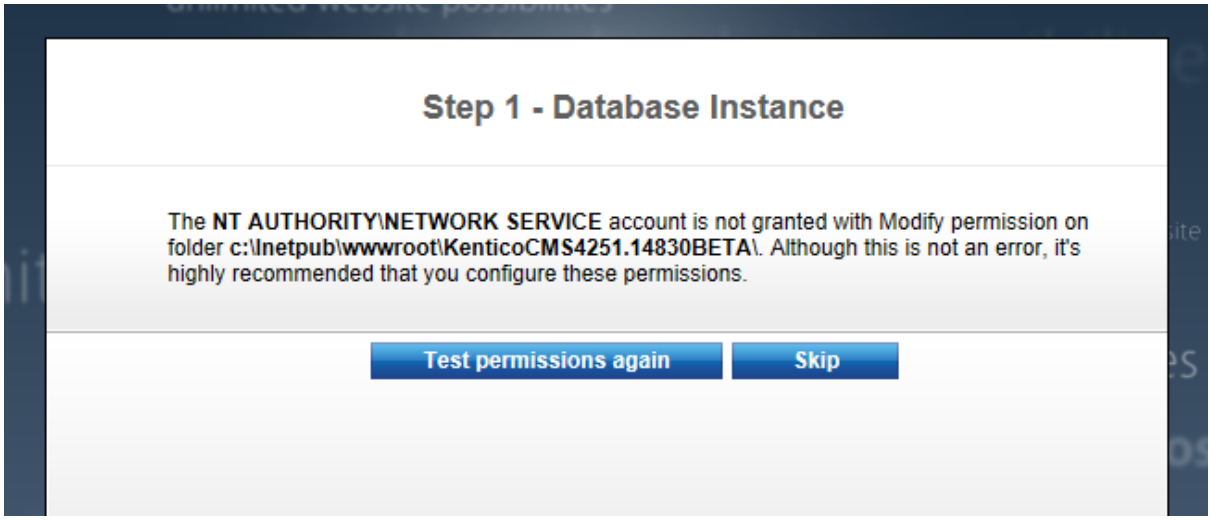
Medium Trust Environment Specifics

For more information on medium trust level please read chapter Configuration for Medium Trust environment.

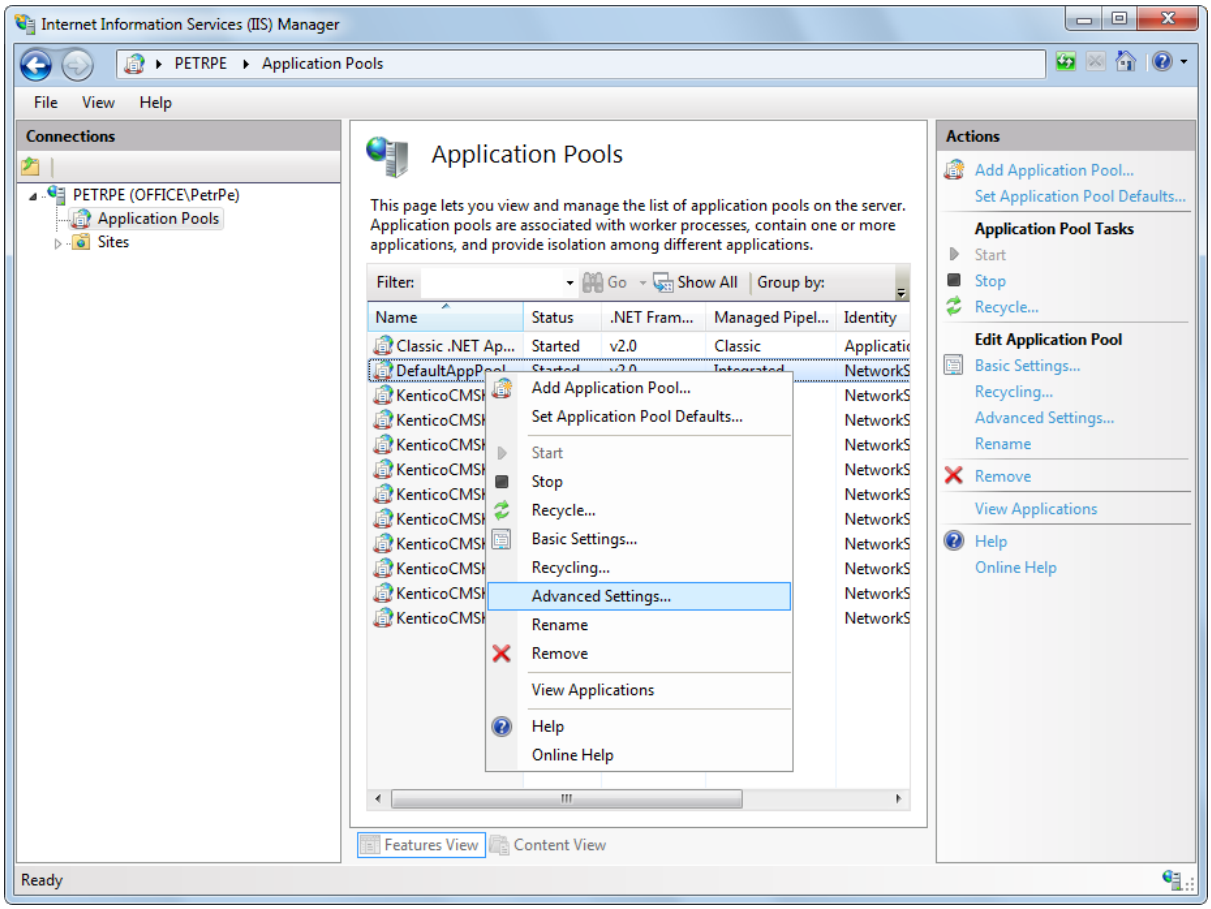
3.4.8 AppPool permissions on Windows 7 or Windows Server 2008 R2

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.

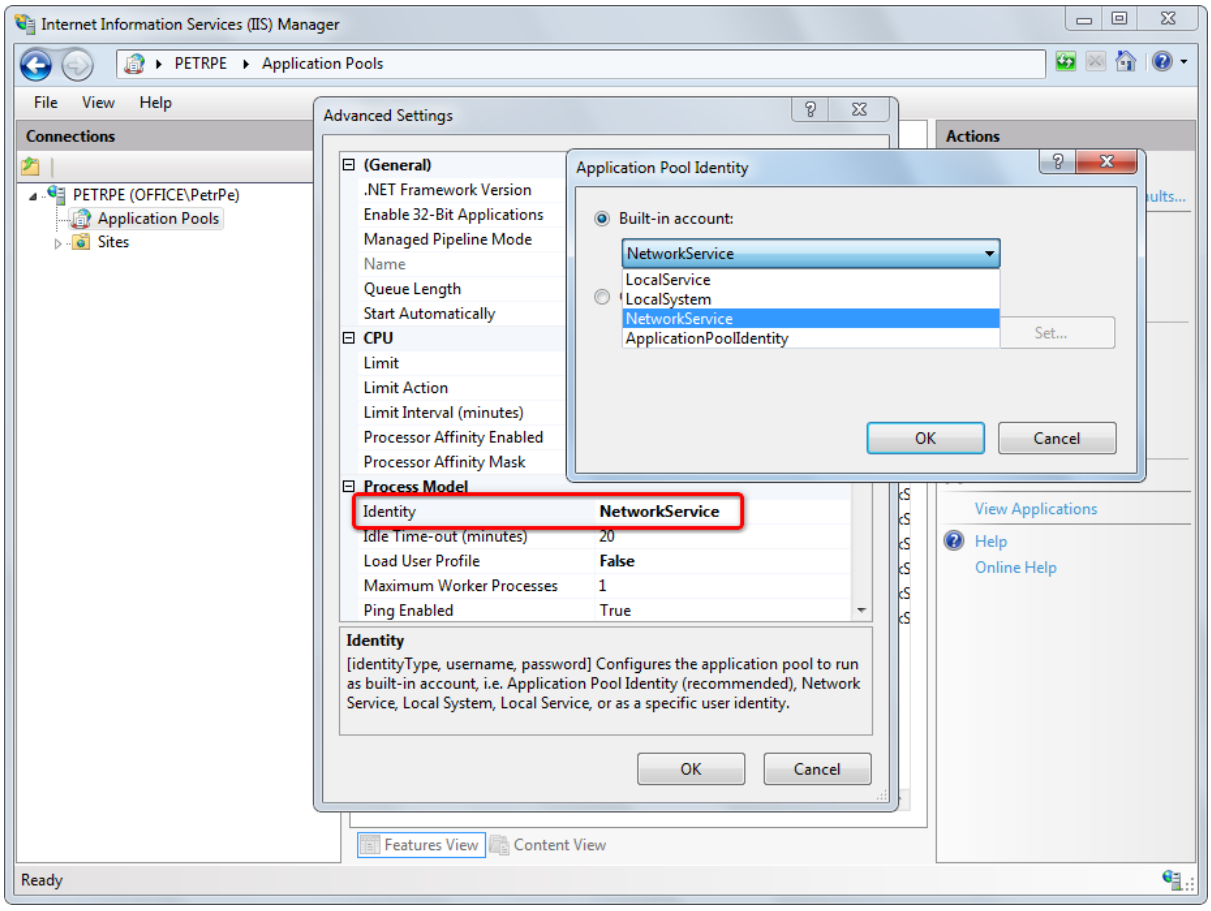
If you can still see the following screen after the end of the Database setup in either of the systems, it's necessary to make some further settings in your IIS.



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.

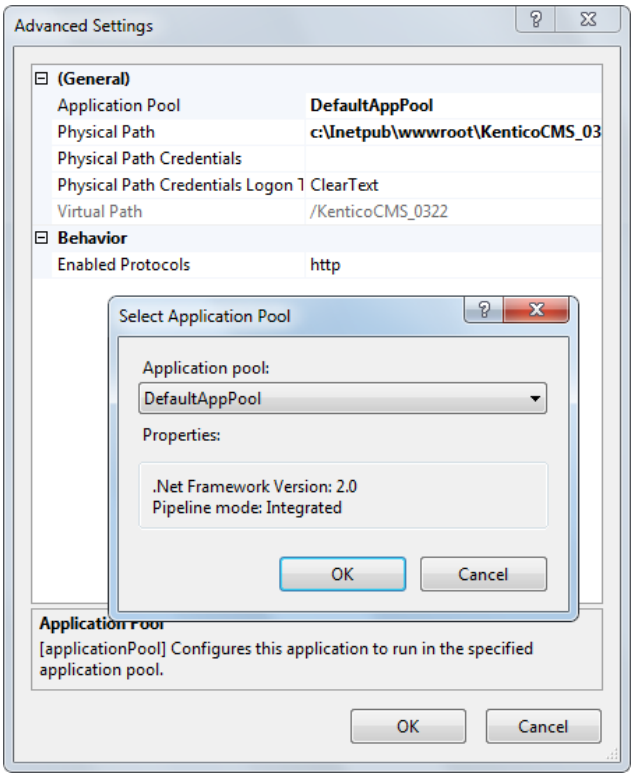


2. In the following dialog, switch the **Identity** value in the **Process Model** category to **Network Service**.



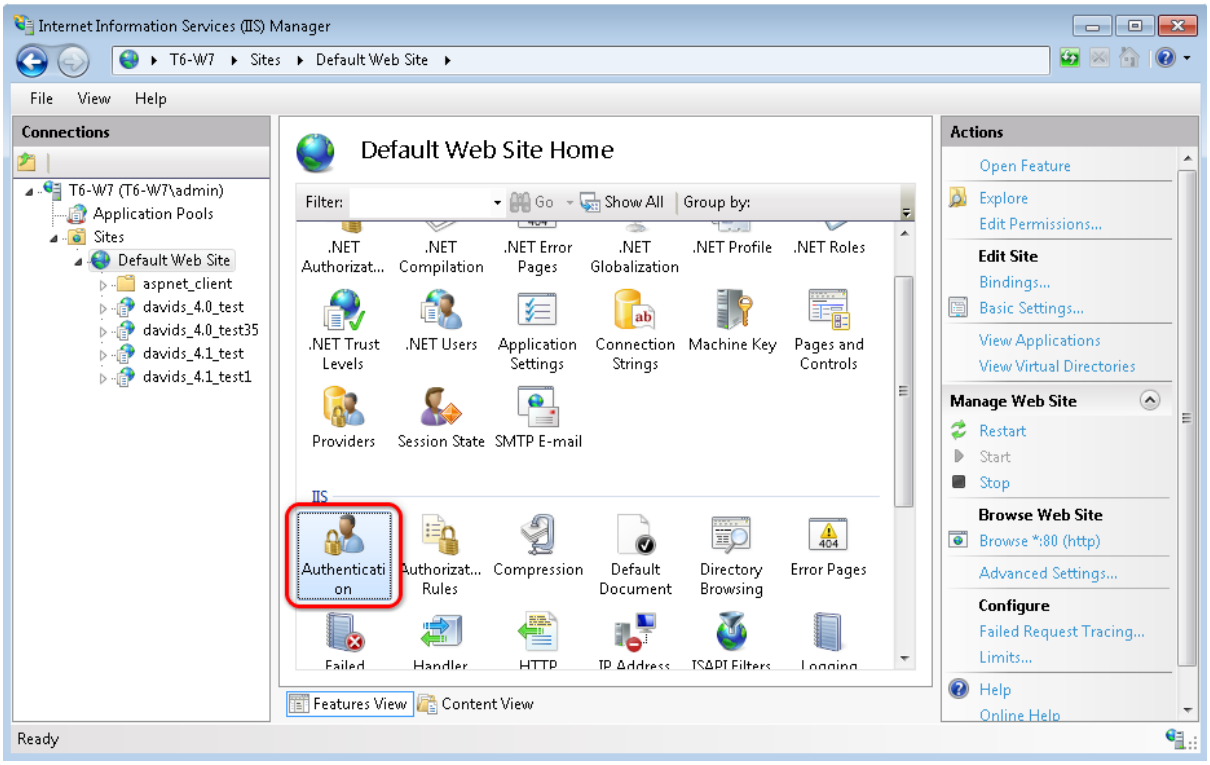
3. Now go to your website's **Advanced settings** ...

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

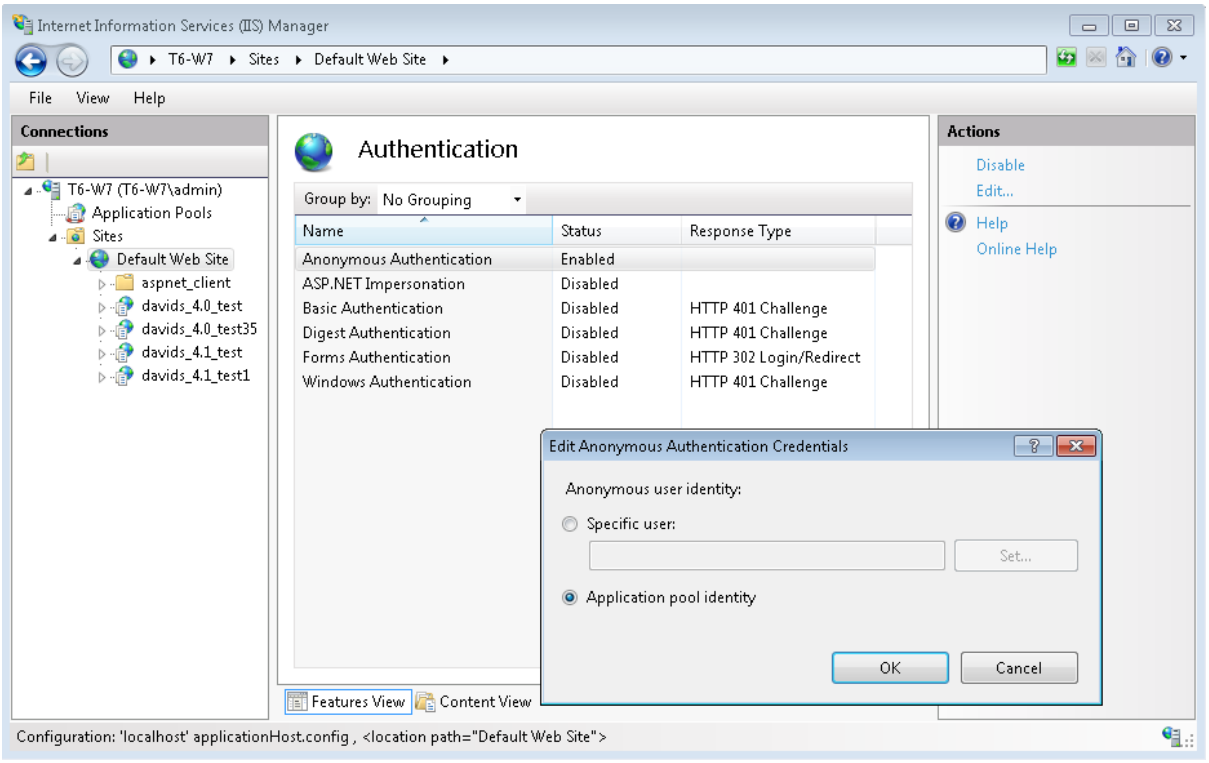


On **Windows 7**, you also need to change the Anonymous authentication settings as described below:

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



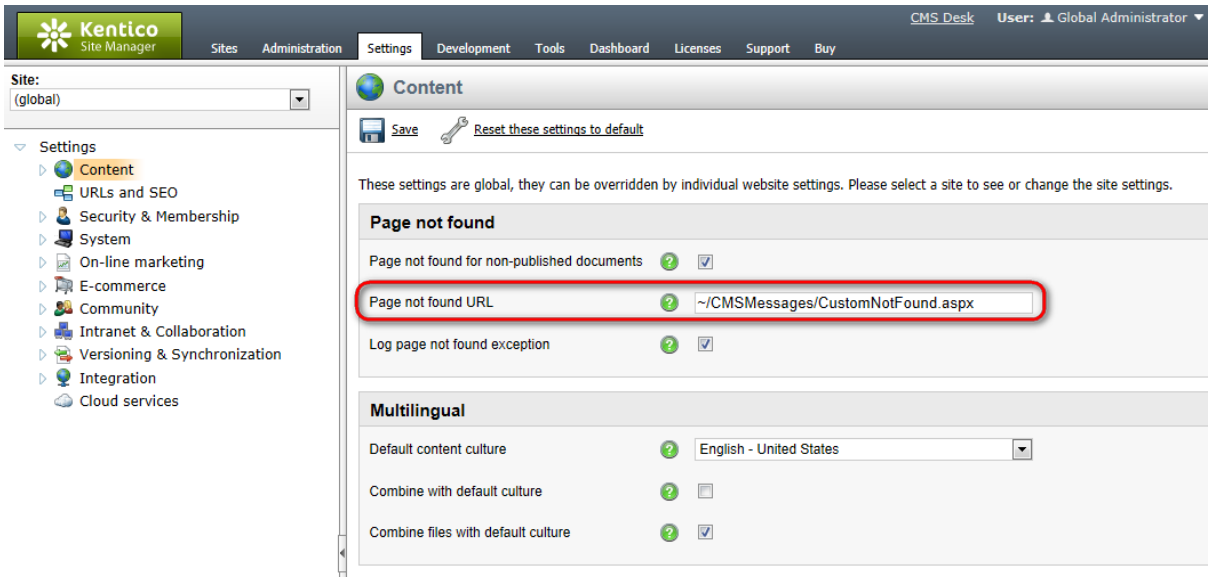
2. Select **Anonymous authentication** from the list and click **Edit**. In the pop-up dialog, choose **Application pool identity** and click **OK**.



3.4.9 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 encounter 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.

One of the most common problems is the *Page not found* error (404 HTTP status code). A custom error page that will be shown as a response may be specified directly through the site settings. Simply go to **Site Manager-> Settings -> Content** and enter the URL into the **Page not found URL** field, for example: *~/CMSMessages/CustomNotFound.aspx*



Adding the error page

When creating a custom error page, it is recommended to add it directly into your web project as an *.aspx* file, e.g. under the **CMSMessages** folder. 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 (*404* in this case).

Since Kentico CMS is a standard ASP.NET application, the general handling of errors and exceptions can be configured by editing the **<customErrors>** section of your **web.config** file.

```
<system.web>
...
    <customErrors defaultRedirect="~/CMSMessages/error.aspx" mode="Off">
        <error statusCode="404" redirect="~/CMSMessages/PageNotFound.aspx" />
    </customErrors>
...
</system.web>
```

The **mode** attribute sets the basic error page behaviour. You may use the following possible values:

- **Off** - in this mode, an error page containing details of the encountered error is displayed to all users (including the stack trace etc.). This is the default setting for Kentico CMS websites.
- **On** - in this case, all users will see a simplified version of the error page with no technical details about the problem.
- **RemoteOnly** - this option ensures that full error details are shown to the local host, but simplified error pages are displayed to everyone else, i.e. end users who access the site remotely.

If you wish to use a completely custom error page, set the **mode** to *On* or *RemoteOnly* and specify the URL through the **defaultRedirect** attribute. You may also add any number of child **<error>** elements to assign a different error page to individual types of HTTP errors. 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 **redirect** value.

3.4.10 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, the HTML content of the on-line help can be opened by any users (including public) if they enter 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 will be able to view it. You can follow the steps below to perform the required configuration:

1. First, find the <*system.webServer*> section of your application in the **web.config** file. One option is to add the **runAllManagedModulesForAllRequests** attribute to the <*modules*> element as shown in the example below:

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

Setting this attribute to *true* ensures that all requests will be processed by the application and will require authentication if needed.

Alternatively, if you do not want the application to process all additional request types, only **.html** and **.htm**, you can 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 environment.

2. Next, define the authorization rules that should be applied to the content of the **CMSHelp** folder. You can do so 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 will only allow authenticated users to access the on-line help files, and public users will not be able to reach them 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>
```

Now only users who belong to the given roles (specified by their code names) will have access.

3.4.11 Database replication

3.4.11.1 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.

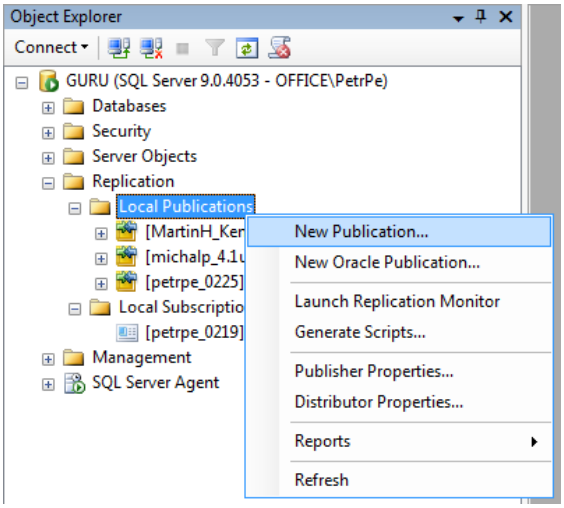
In case that you needed to use any of the limited features listed above, you would need to disestablish your existing replication, make the required changes and set up new database replication from scratch.

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

3.4.11.2 Creating a publication

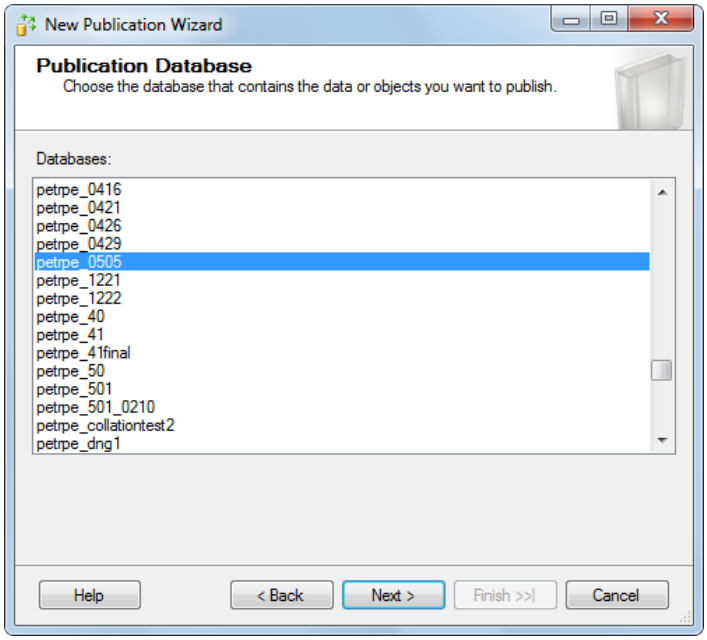
The first logical step in setting up database replication is to create the publication.

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.

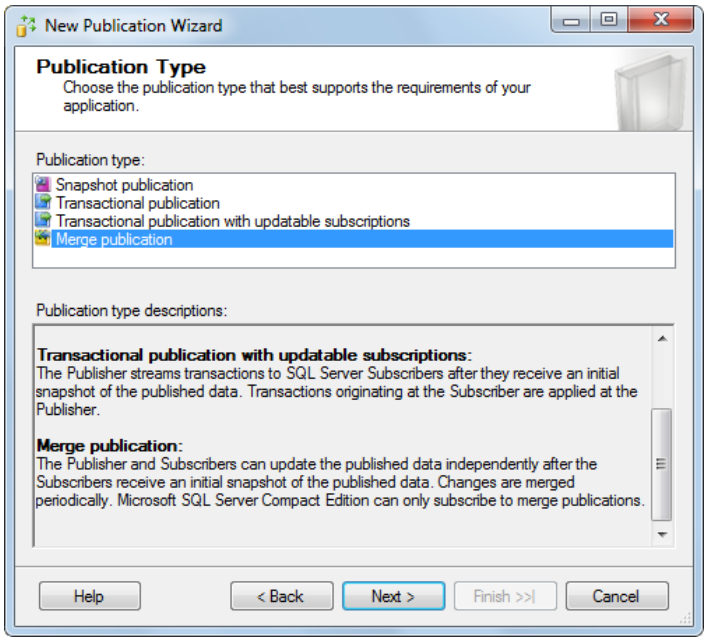


2. The **New Publication Wizard** starts. Click **Next** in the first step.

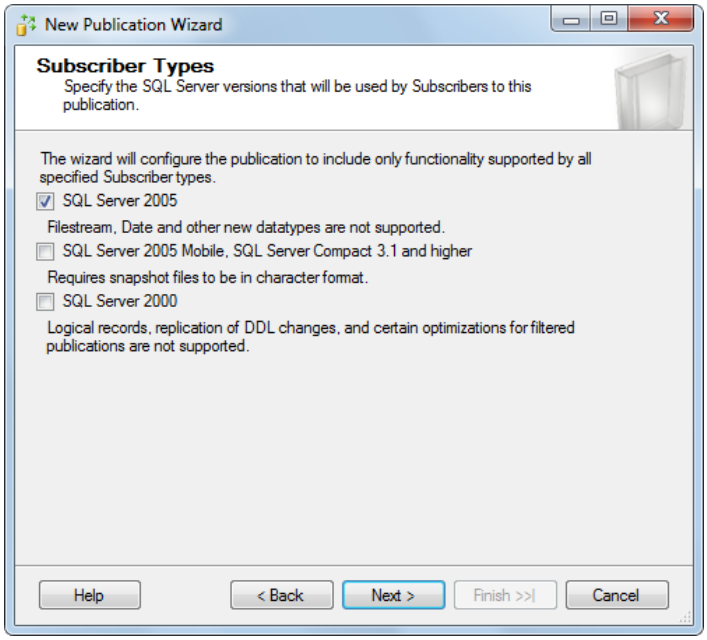
3. Now you need to select the publication database. All databases on the server will be listed, so choose the one which you want to be the publisher and click **Next**.



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**.

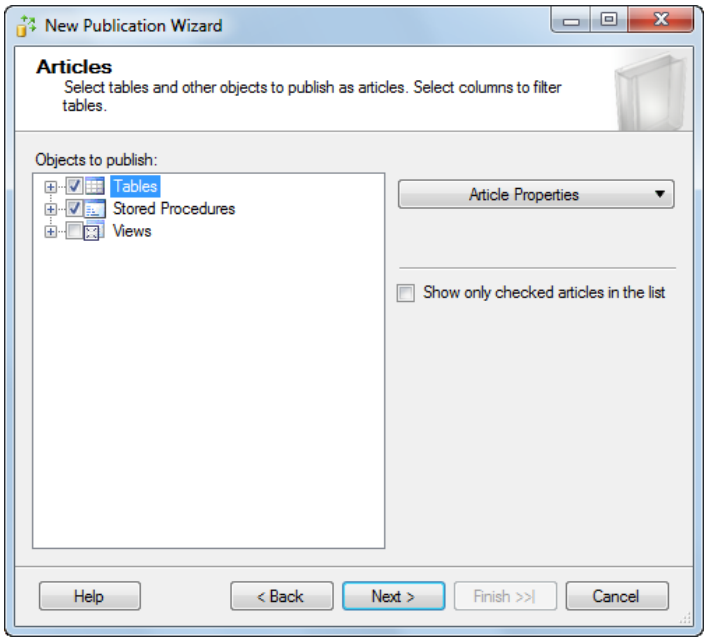


5. This step lets you specify which versions of SQL Server are supported for the subscribers. Choose the versions that your subscribers are running on and click **Next**.



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).

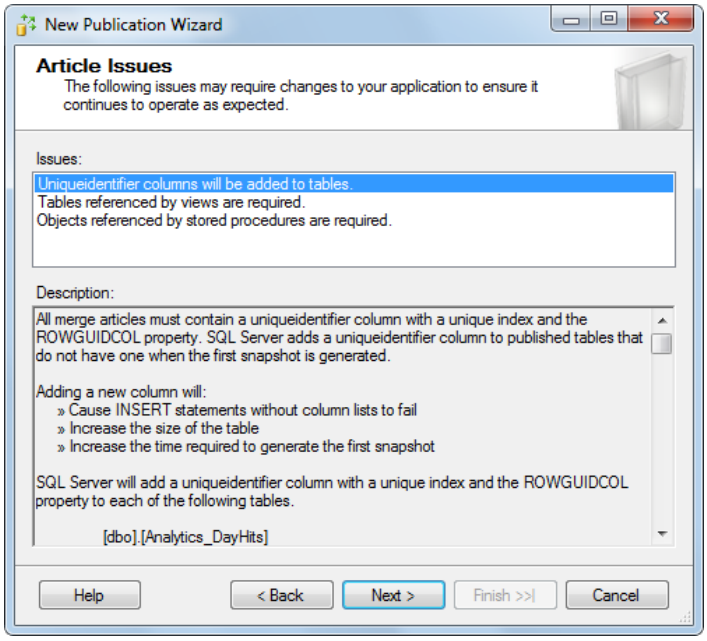
Select the tables and stored procedures that you wish to synchronize (unless you have some specific needs, it makes the most sense to select all) and click **Next**.



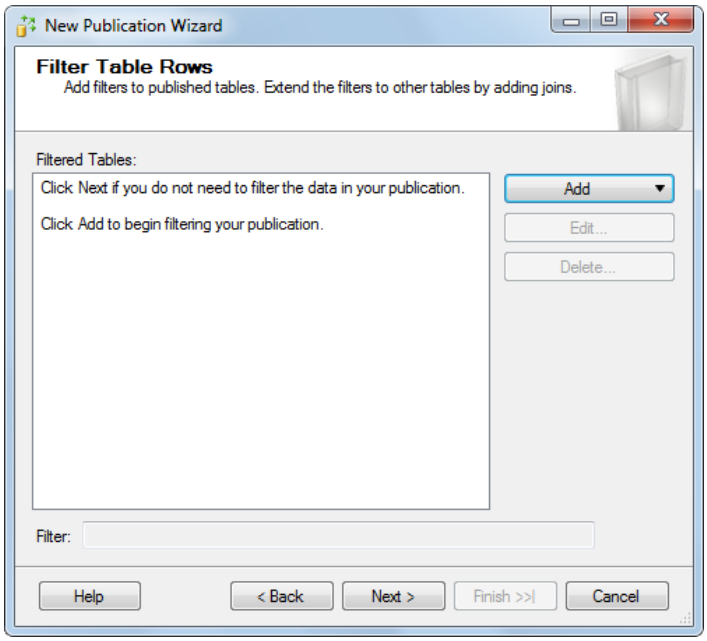
7. The **Arcile Issues** step is only informational - it notifies you about modifications to your database which are required for the application to work as expected with the replication.

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**.



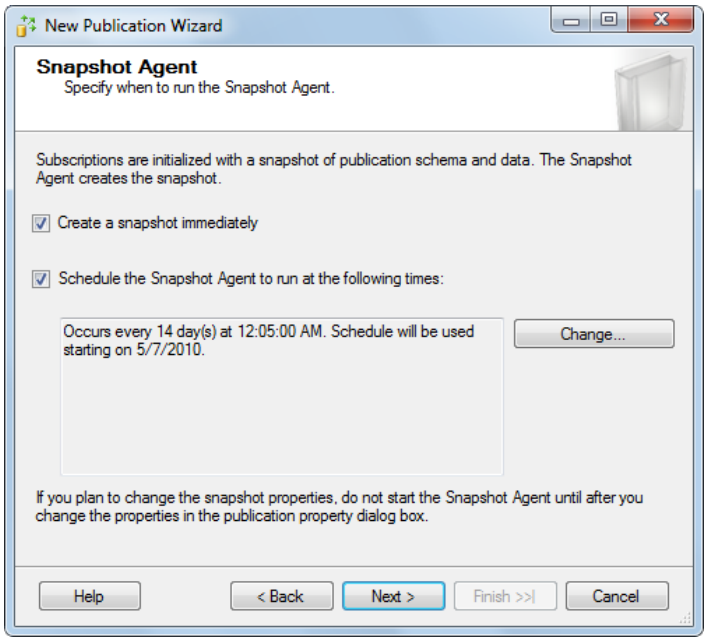
8. In the **Filter Table Rows** step, you can add filters the published tables. This depends on your specific needs, so add filters as needed (or do not add any) and click **Next**.



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:

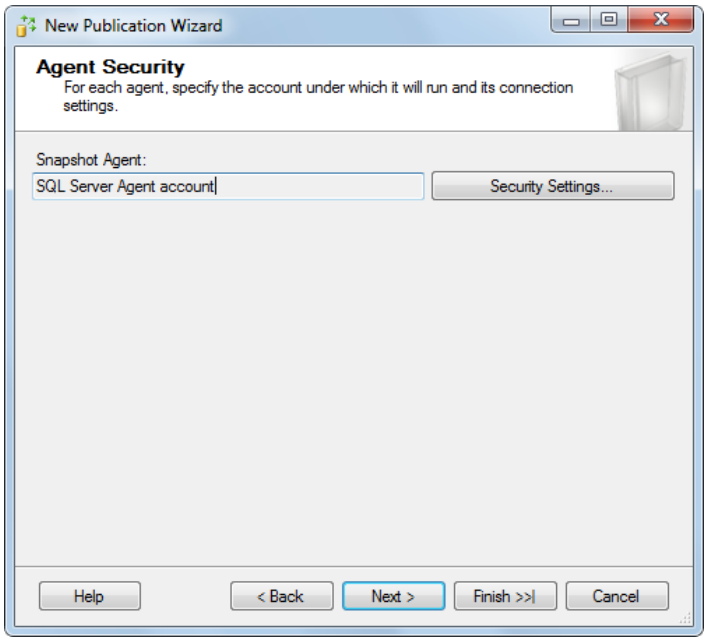
- **Create a snapshot immediately** - if enabled, the Snapshot Agent will take the snapshot immediately after this wizard is completed
- **Schedule the Snapshot Agent to run at the following times** - if enabled, the Snapshot Agent will take the snapshot on a regular basis, which can be specified if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

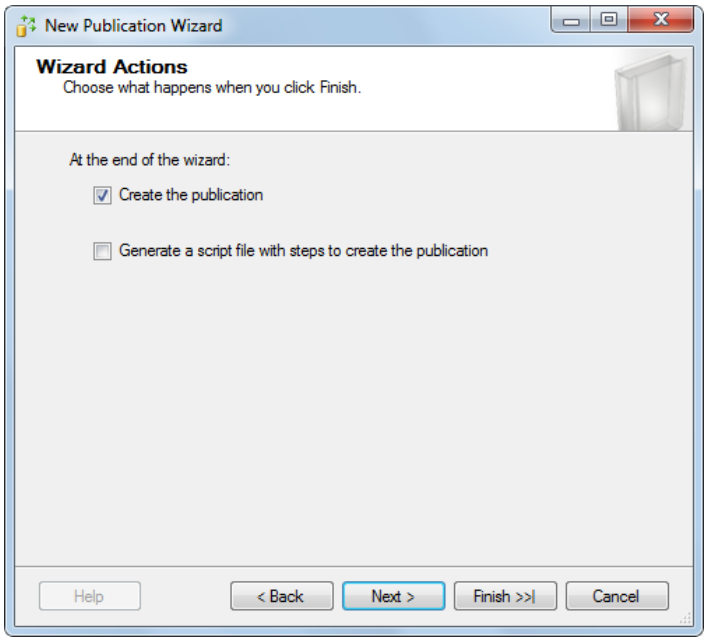
Make a configuration suitable for your environment and click **Next**.



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

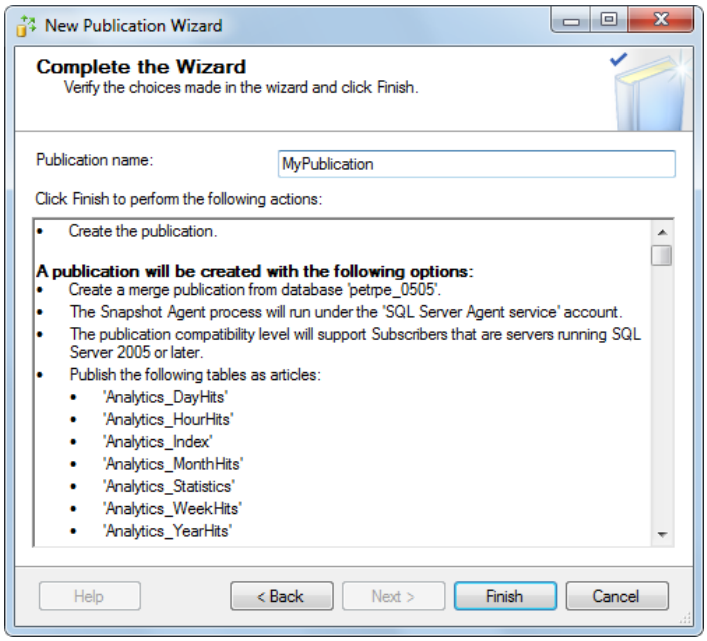
- **Create the publication** - if enabled, the publication will be created as defined throughout the wizard
- **Generate a script file with steps to create the publication** - if enabled, the wizard generates a script which, when executed, creates the publication as defined throughout the wizard

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.

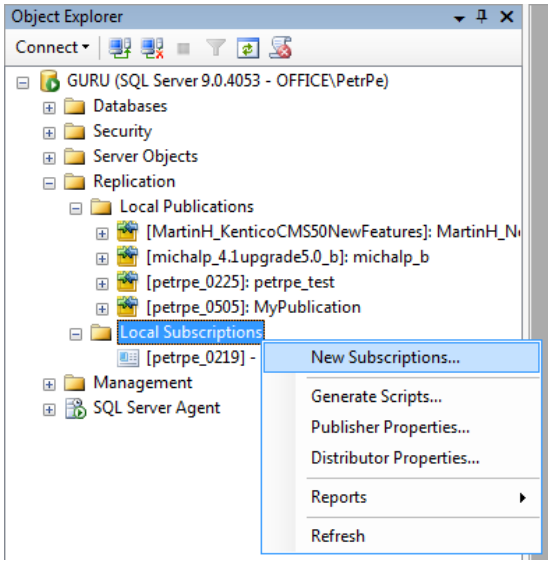
Click **Finish** to create the publication based on the listed options.



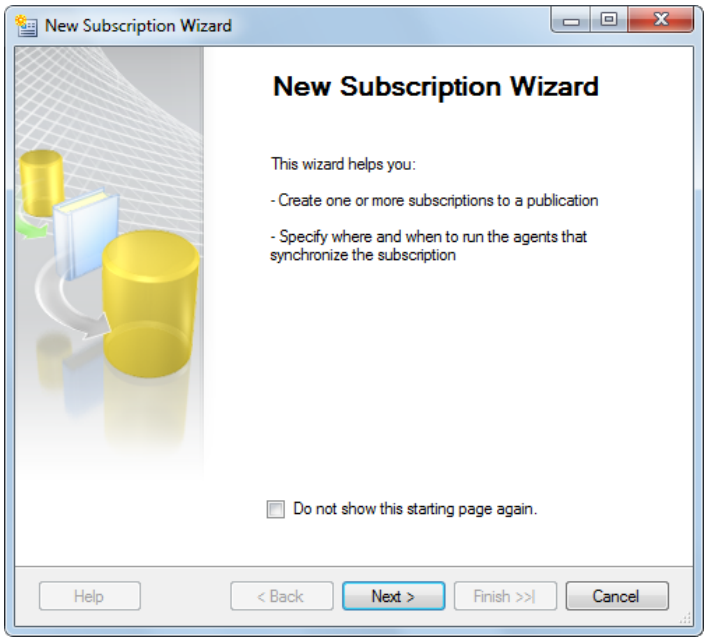
3.4.11.3 Creating a subscription

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.

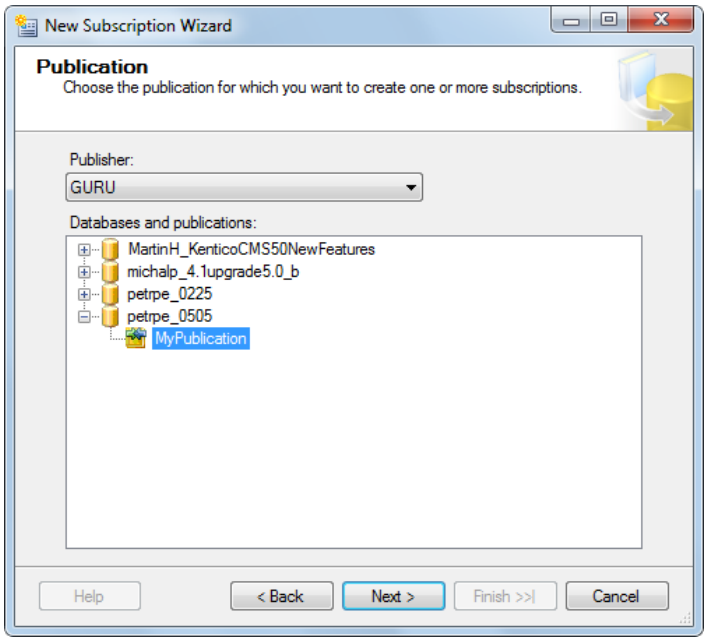


2. The **New Subscription Wizard** starts. Click **Next** in the first step.



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.

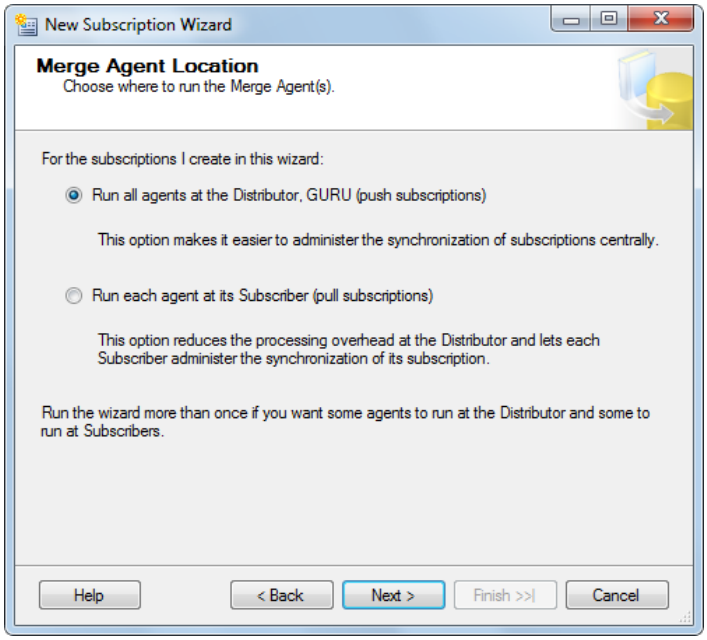
Select the required publication and click **Next**.



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)

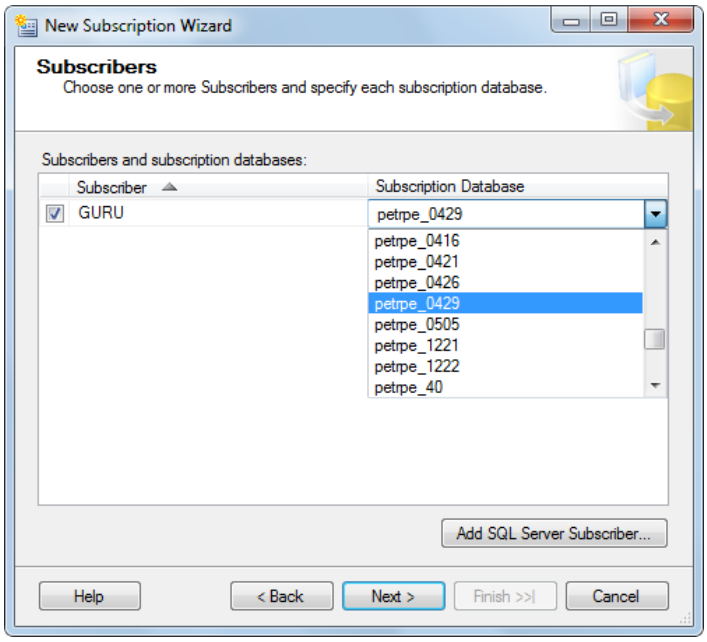
Make you choice (running the agent at the Distributor is recommended for most cases) and click **Next**.



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

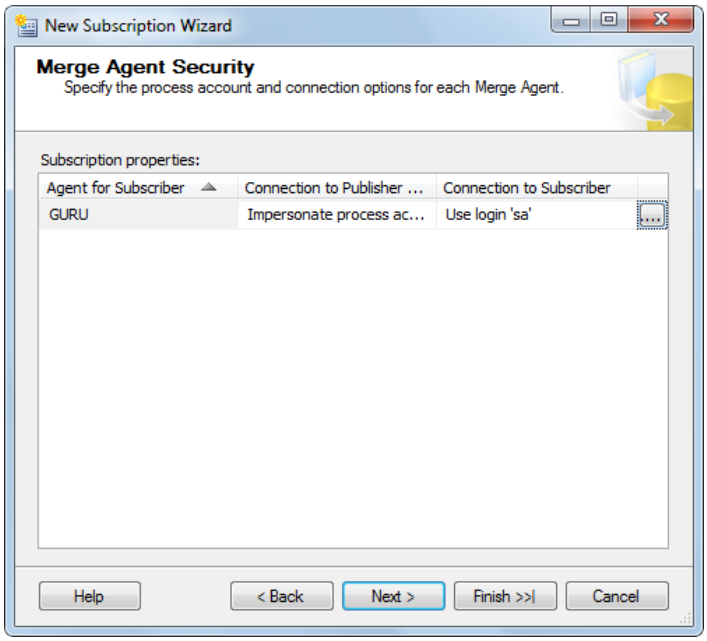
The currently managed database server will be offered in the **Subscriber** column. Using the **Add SQL Server Subscriber** button, you can add other servers to the list, which enables you to create subscriptions on a different server than the currently managed one. Next to each server, you can find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

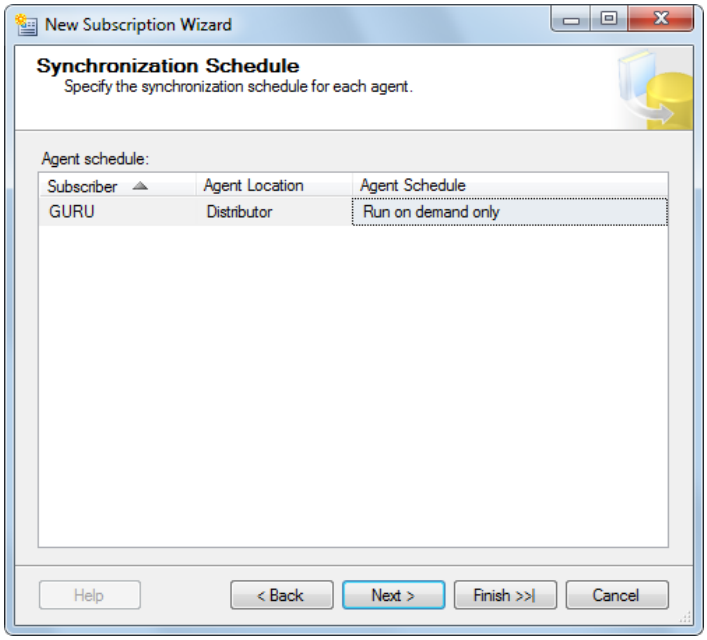
Specify the required information according to your environment and click **Next**.



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:

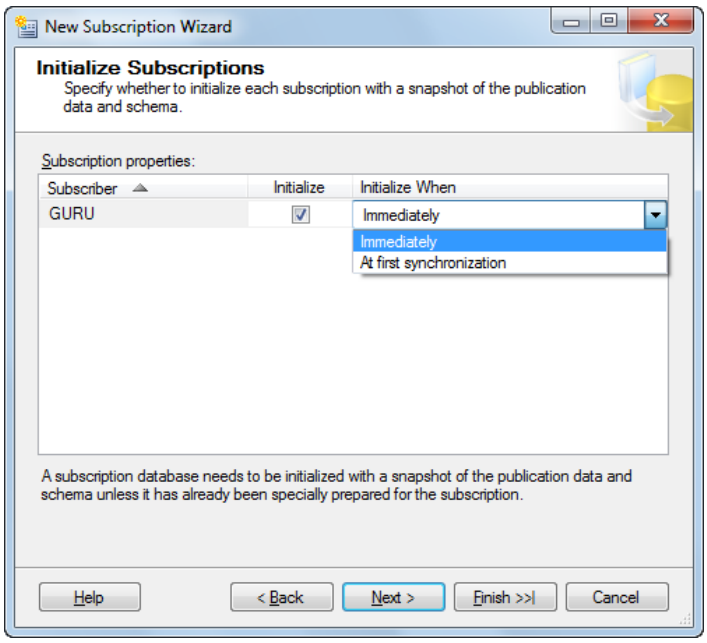
- **Run continuously** - synchronizes immediately whenever a change is made
- **Run on demand** - synchronization is performed only when executed manually from **Replication Monitor** in **SQL Server Management Studio**
- **Run on schedule** - synchronization is performed on a regular basis after a set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

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

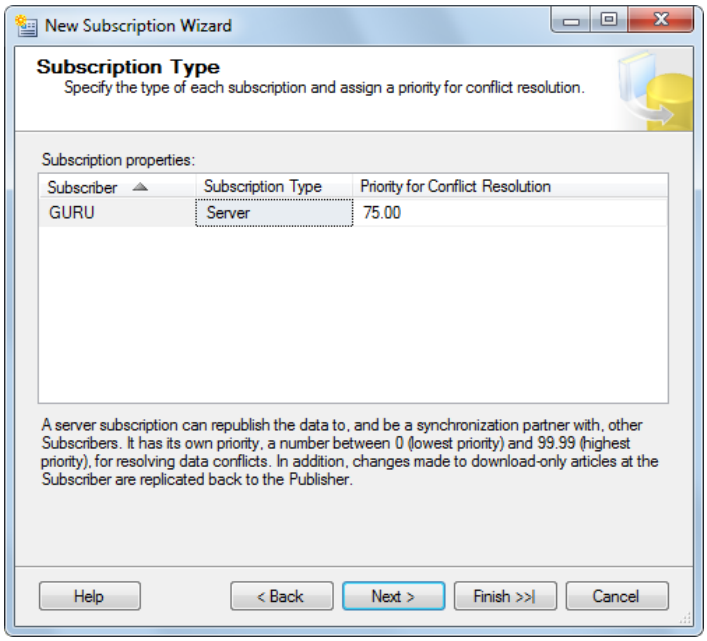


8. This step lets you initialize the subscription, i.e. fill the subscription database with data from the

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**.



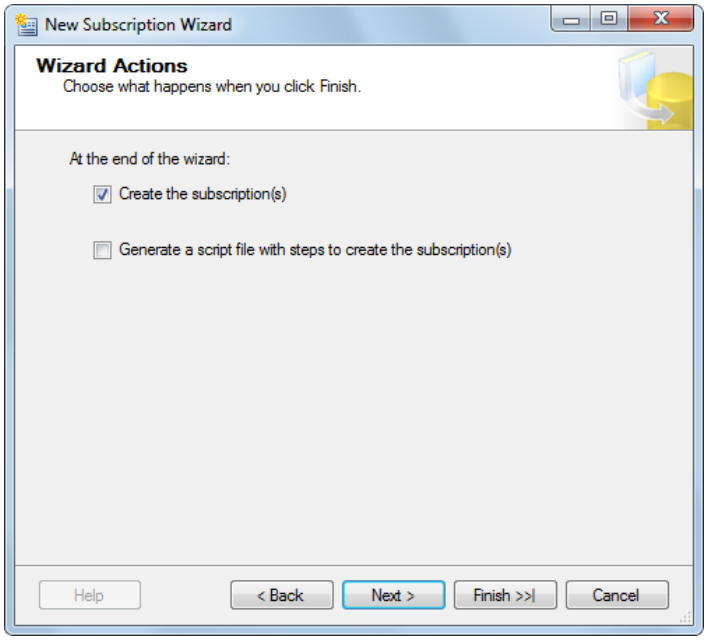
10. The **Wizard Actions** step lets you decide what happens when you finish the wizard. You have the following two options:

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

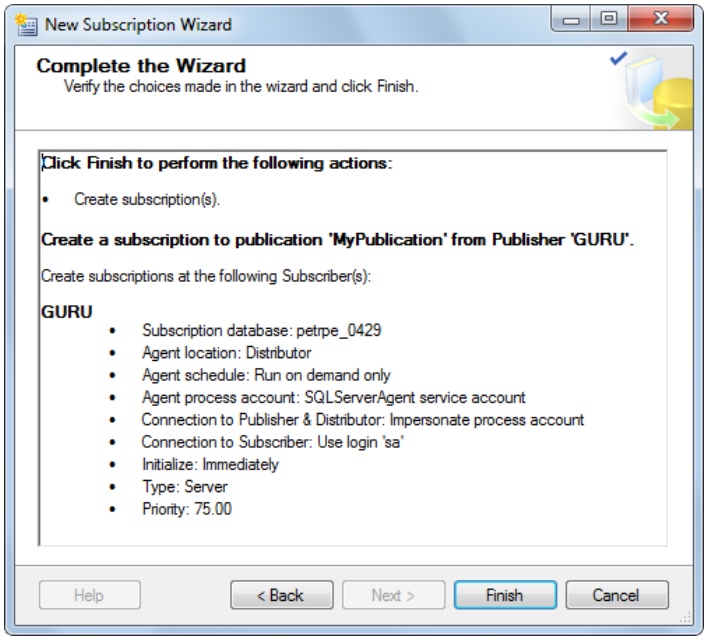
wizard

- **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.



3.4.11.4 Modifying structure of a replicated DB

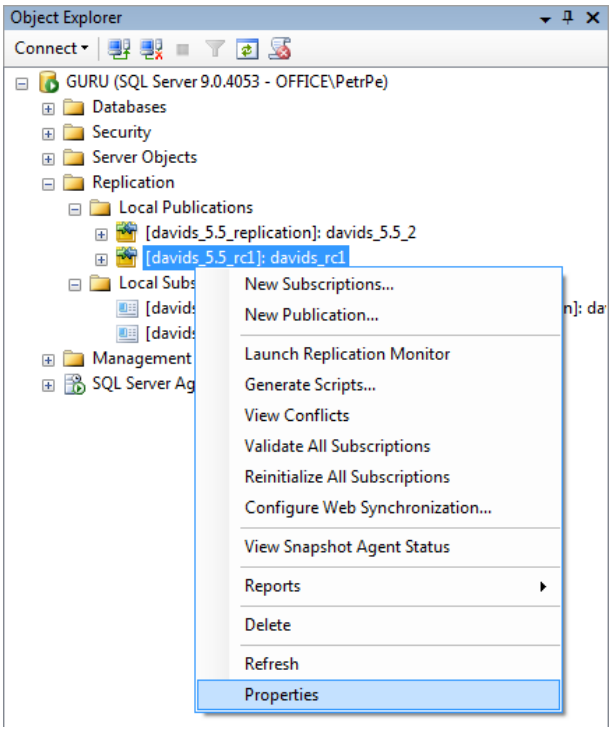
It is possible to make some changes to the DB structure even without disestablishing the existing replication.

Adding forms, custom tables or document types

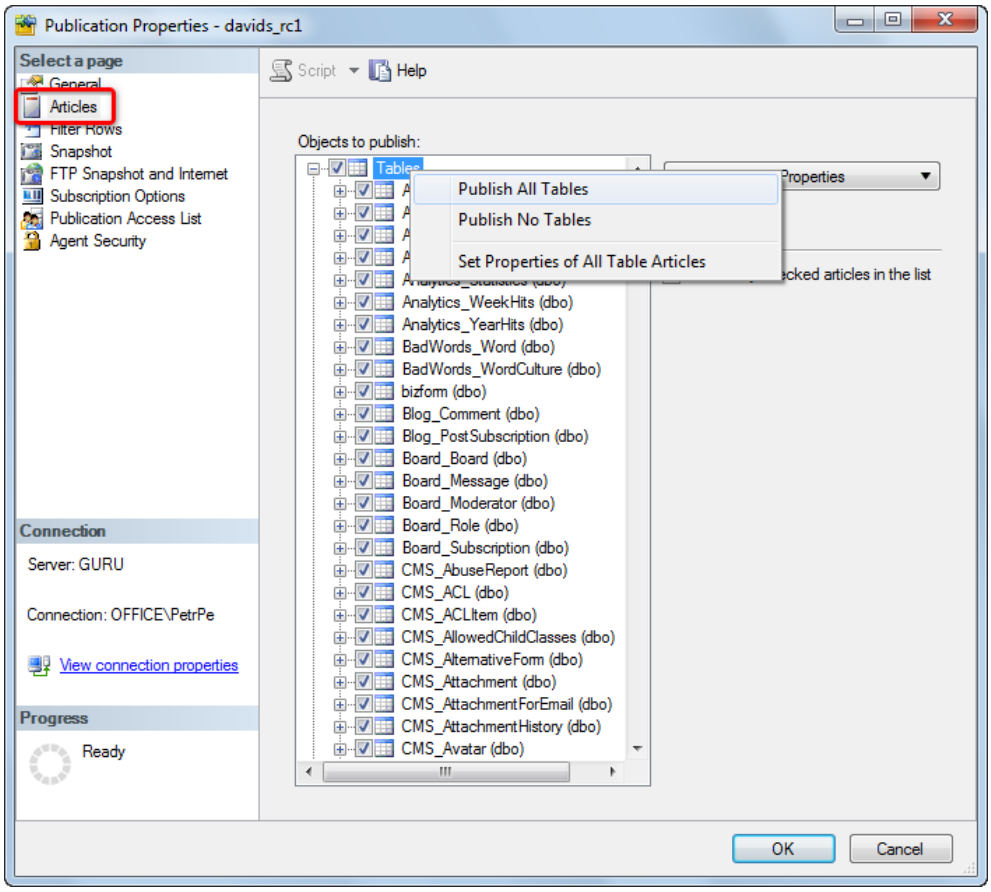
Forms, custom tables and document types can be created even when database replication is established. You only need to follow the steps below to

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

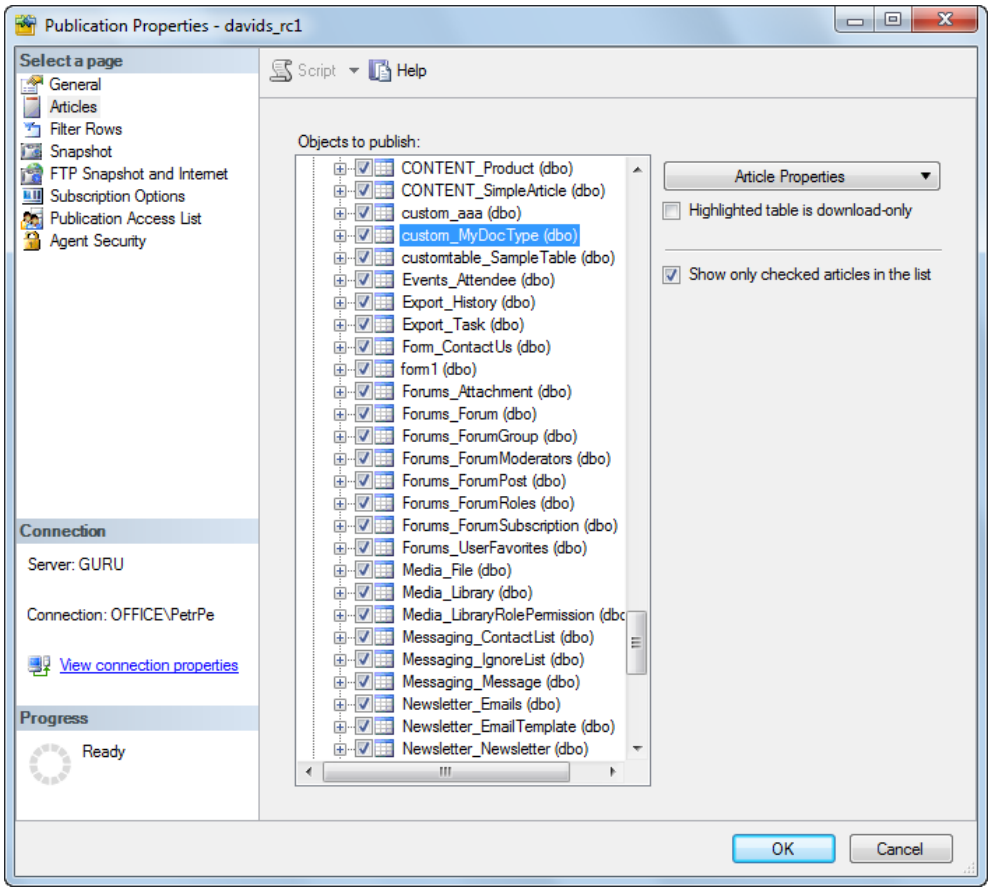
2. Open **SQL Server Management Studio**. In **Object Explorer**, expand the **Replication -> Local publications** node, right-click your publication and choose **Properties** from the context menu.



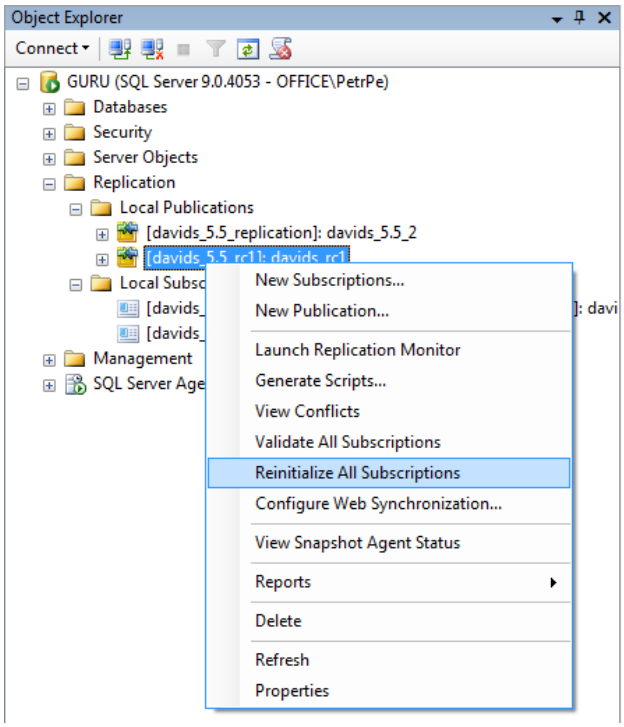
3. In **Publication Properties**, select **Articles** from the right menu. You will see a tree of database tables which are synchronized to the subscribers. You can verify that the new table created in step 1 is missing, i.e. it is not synchronized. Right-click **Tables** (the root of the tree) and click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



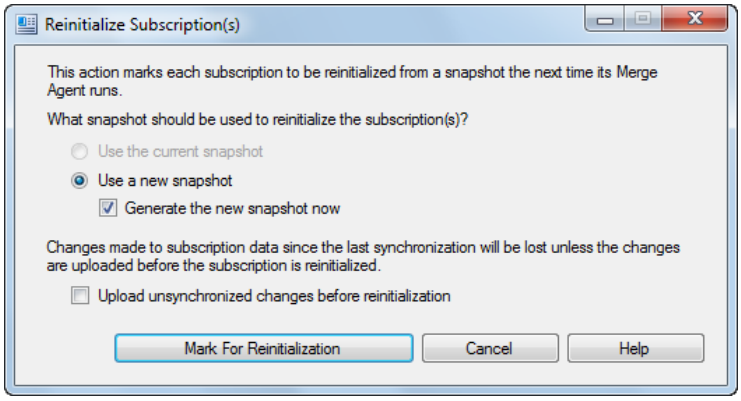
5. Now back in **Object Explorer**, right-click the publication again and choose **Reinitialize All Subscriptions** from the context menu.



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

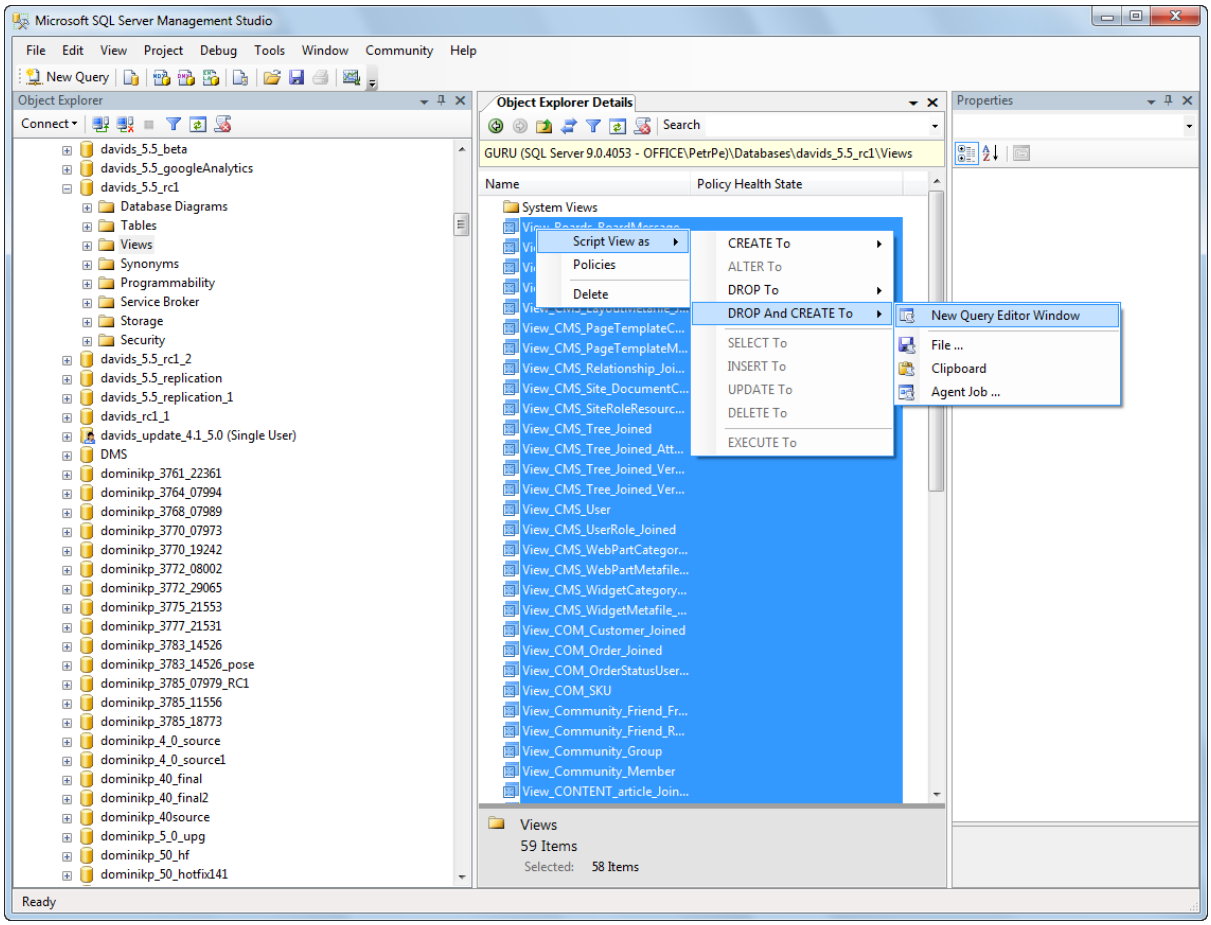
modified publisher database.

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.

3.4.12 Configuration of full-text search in files

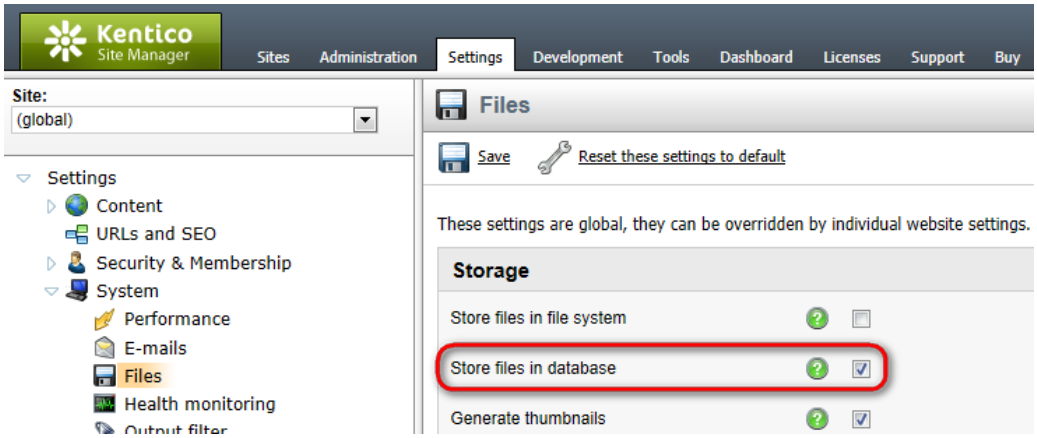
3.4.12.1 Overview

Kentico CMS allows you to full-text search files uploaded into the database. It uses standard Microsoft SQL Server Search Engine that allows you to index a wide variety of files. This chapter explains how you can configure your installation of Kentico CMS for search in files.

Prerequisites - Important!

If you're using Microsoft SQL Server 2005, you need to ensure that full-text search support is installed on the server. The full-text search is available for all editions of Microsoft SQL Server 2005, including Express Edition with Advanced Services.

Generally, full-text search support must be installed on your SQL Server and your website must be configured for storing files in the database (**Site Manager -> Settings -> System -> Files -> Store files in database** must be enabled).



Now follow one of the following guides:

- Configuration of full-text search on Microsoft SQL Server 2005 and 2008
- Configuration of full-text search on Microsoft SQL Server 2005 Express Edition

Supported file types

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

You can install a free driver from Adobe for searching in PDF files as described in the Searching PDF files topic.

To search Microsoft Office 2007 and 2010 document files such as DOCX or XLSX, it is necessary to install a free IFilter pack as described in the Searching Office 2007/2010 documents topic.

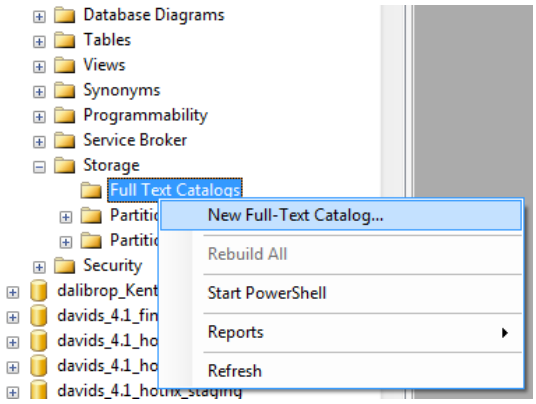
If you want to search other types of documents, you need an appropriate IFilter library.

> IFilter libraries can be purchased from third-party vendors (you can find them using Google).

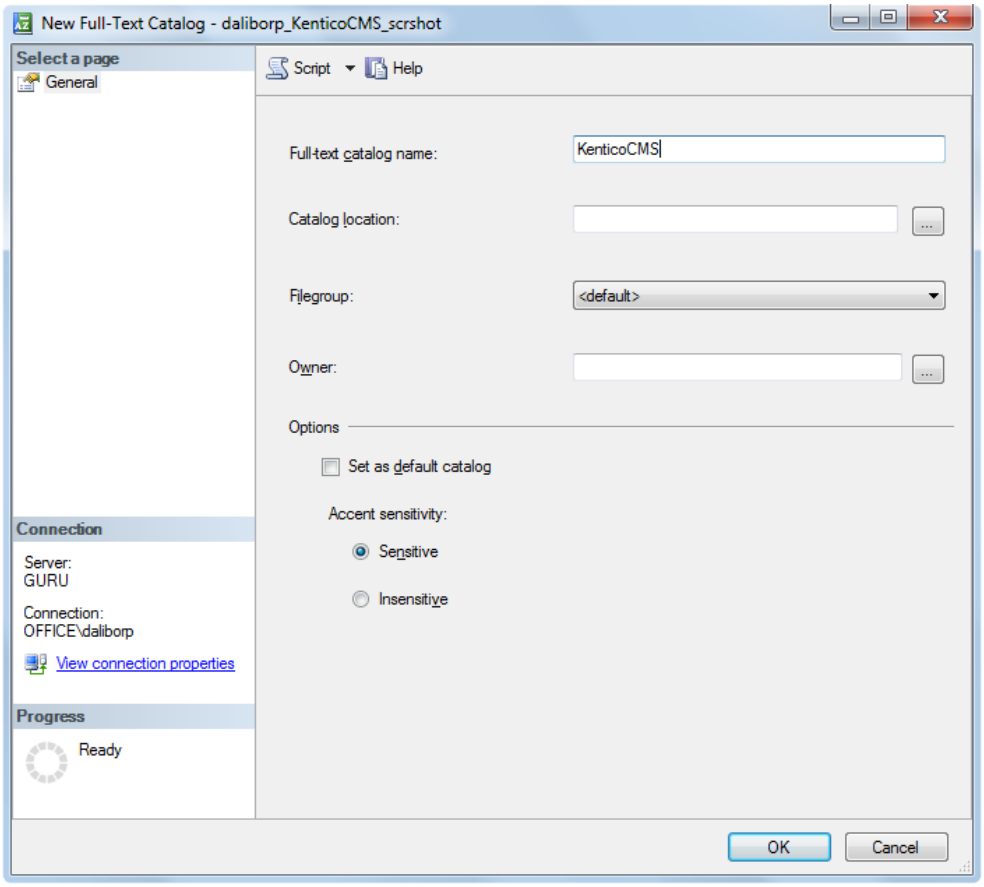
3.4.12.2 Configuration on MSSQL 2005 and 2008

1. Start **Microsoft SQL Server Management Studio**.

2. Locate your database with Kentico CMS objects, unfold **Storage**, right-click **Full Text Catalogs** and choose **New Full-Text Catalog**.

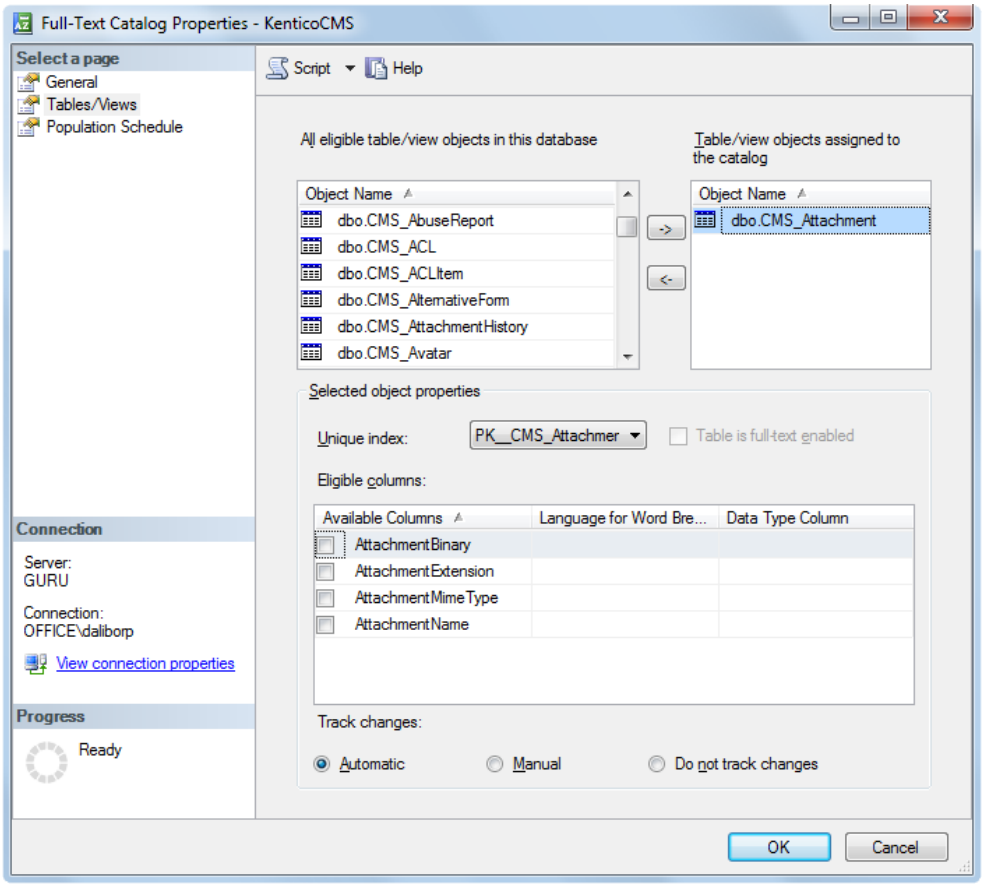


3. On the **New Full-Text Catalog** dialog, enter Full-text catalog name *KenticoCMS* and click **OK**.



4. Right-click the newly created KenticoCMS full-text catalog and choose **Properties**. On the Full-Text Catalog Properties dialog, choose the **Tables/Views** tab.

5. Assign the *CMS_Attachment* table to the catalog, check the box next to the *AttachmentBinary* column, set the **Language for Word Breaker** to *English* or other value and set the **Data Type Column** to *AttachmentExtension* as shown on the following figure and click **OK**.



6. Sign in as administrator and go to **Site Manager -> Development -> Document types -> Root -> Queries** and edit the **searchattachments** query. Uncomment the following query:

```
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 NodeAliasPath LIKE @NodeAliasPath
AND
([AttachmentName] Like N'%'+ @Expression + N'%' OR FREETEXT(attachmentbinary,
@expression))
AND @SiteName = SiteName AND (@DocumentCulture = DocumentCulture OR
@DefaultCulture = DocumentCulture )
```

Note: It is not necessary to uncomment the query if a **Smart search** web part is used for search. In that case the **Search in attachments** option in **Smart search dialog with results** has to be checked.

3.4.12.3 Configuration on MSSQL 2005 Express Edition

If you cannot use the management console for full-text search configuration, you can use the following SQL script:

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

-- Create Full-Text catalog
exec sp_fulltext_catalog 'MyKenticoCMSCatalog','create'

-- Add Full-Text catalog to table
exec sp_fulltext_table 'CMS_Attachment','create','MyKenticoCMSCatalog',
'PK_CMS_Attachment'

-- Add Index to table column
exec sp_fulltext_column 'CMS_Attachment','AttachmentBinary','add',NULL,
'AttachmentExtension'

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

Please note: Similarly to Configuration on MSSLQ 2005 and 2008, you need to go to **Site Manager -> Development -> Document types -> Root -> Queries**, edit the **searchattachments** query and uncomment the following query to be able to run full-text search.

```
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 NodeAliasPath LIKE @NodeAliasPath
AND
([AttachmentName] Like N'%'+ @Expression + N'%' OR FREETEXT(attachmentbinary,
@expression))
AND @SiteName = SiteName AND (@DocumentCulture = DocumentCulture OR
@DefaultCulture = DocumentCulture )
```

Note: It is not necessary to uncomment the query if a **Smart search** web part is used for search. In that case the **Search in attachments** option in **Smart search dialog with results** has to be checked.

3.4.12.4 Searching PDF files

Searching in PDF files is not supported in Kentico CMS by default. To enable it, you need to install the free **Adobe PDF IFilter** on your machine. Please note that Adobe PDF IFilter is not a product of Kentico Software and we cannot guarantee its functionality.

The installation differs for 32-bit and 64-bit systems. Both are described in the text below.

Configuration on 32-bit platforms

The following procedure needs to be followed 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. After you finish the installation, rebuild the full-text search catalog that you created for Kentico CMS. Now you should be able to search PDF files.

4. If you're using MS SQL Server 2005, you may need to run the following SQL commands so that the server can load PDF IFilter libraries:

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

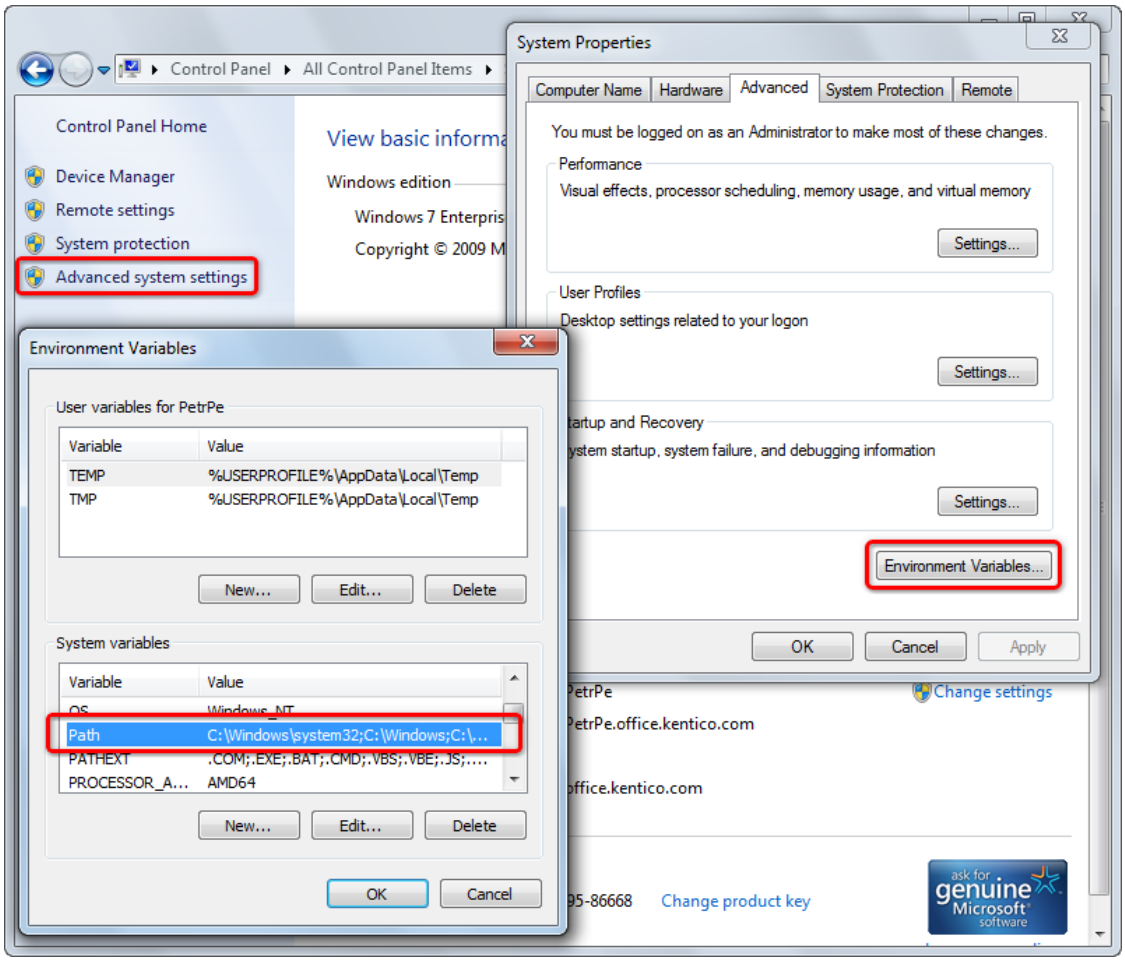
Configuration on 64-bit platforms

The following procedure needs to be followed 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, run the installer and follow the instructions.

3. After installing the PDF IFilter, it is recommended to set your system **Path** environment variable to the **bin** folder of the IFilter installation. Go to **Control Panel -> System -> Advanced system settings -> Environment Variables** and append *C:\Program Files\Adobe\Adobe PDF IFilter 9 for 64-bit platforms\bin* to the value of the *Path* variable.



4. Restart the computer. When restarted, check whether Adobe IFilter is registered with the server - in a new query window, type and execute the following:

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

You should see a list of installed filters in the output window. Verify if you can see an entry for **.pdf** with a correct path set.

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

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

6. Restart the SQL Server. Run the query from step 4 above again and verify that you can see the entry for **.pdf**.

3.4.12.5 Searching Office 2007/2010 documents

Searching in Microsoft Office 2007/2010 documents is not supported in Kentico CMS by default. To enable it, you need to install and configure **Microsoft Filter Pack** IFilters on your machine. Please note that Microsoft Filter Pack is not a product of Kentico Software and we cannot guarantee its functionality.

The following procedure needs to be completed to enable searching in the desired file types:

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, select the instance that contains 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 installed filters on the **Results** tab of the output window. Verify that the list contains entries for all desired file extensions with a correct path set.

5. Restart the SQL Server service. Once this is done, **Rebuild** the full-text search catalog that you created for Kentico CMS. Now you should be able to search Office 2007/2010 files.

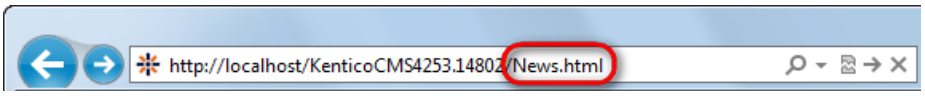
3.4.13 Custom URL extensions and extensionless URLs

3.4.13.1 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.

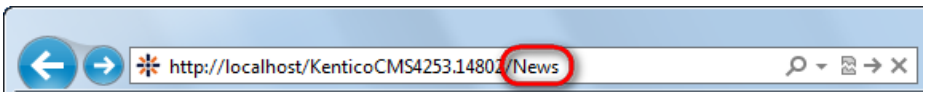
So for example, instead of the default *http://www.example.com/news**.aspx***, you can have URLs ending with **.html**:

*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.

3.4.13.2 IIS 7 and higher

If your instance of Kentico CMS is running on a server with IIS 7 or higher installed, the procedure is quite simple - it consists only of adding and modifying a few keys in your project's **web.config** file. Please be aware that this procedure only works for IIS 7 sites that use an **Application Pool** with **Managed Pipeline Mode** set to *Integrated*. This should be the default setting.

Required configuration

First, you need to modify the *system.webServer* section of your **web.config** file by changing the opening <*modules*> tag as shown in the code example below.

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

With this modification in your web.config, you can proceed to configuring the extensions in the administration interface of Kentico CMS. You can also perform some additional configuration as described below.

Optional configuration (recommended)

3. 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 specifying the URL of the appropriate page as the value of the **Page not found URL** setting in **Site Manager -> Settings -> Content**, for example: *~/CMSMessages/PageNotFound.aspx*.

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.

4. 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" />
```

5. 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" />
```

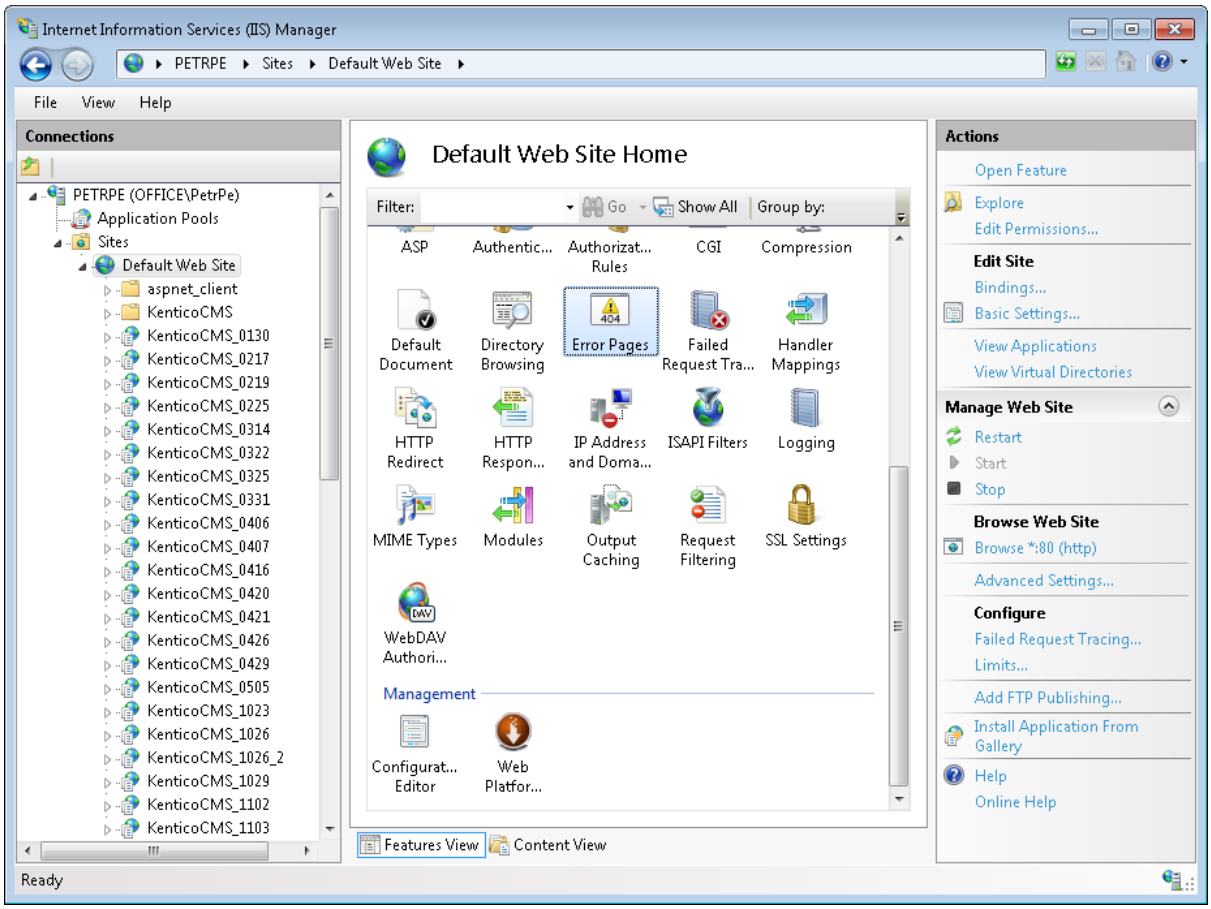
Using the cmspages/handler404.aspx special page (obsolete)

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.

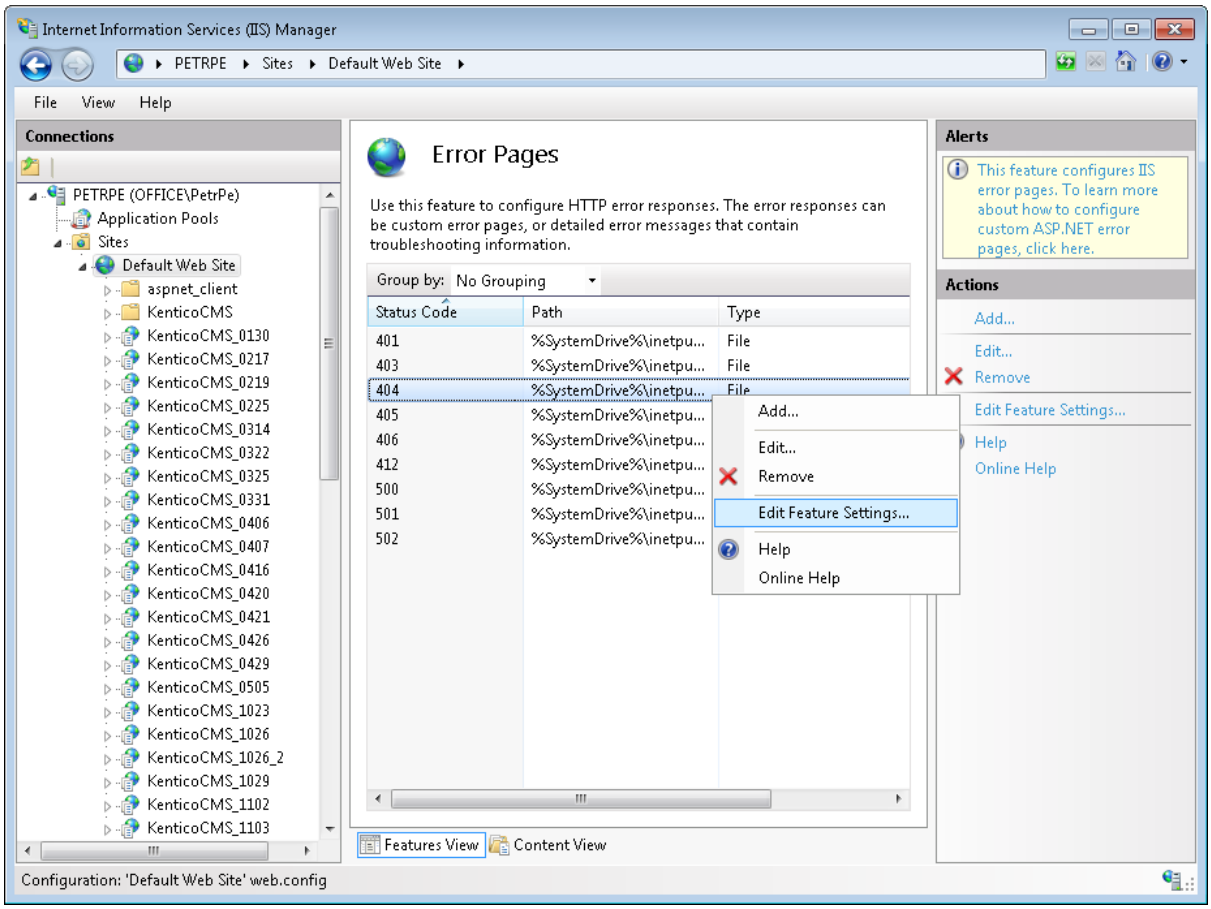
The setup procedure on IIS 7 is the following:

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.



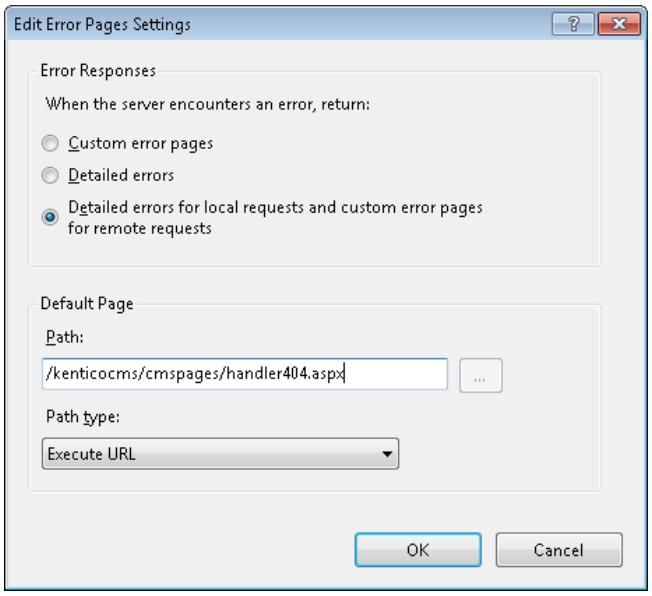
3. Right-click the **404 error** and choose **Edit Feature Settings**.



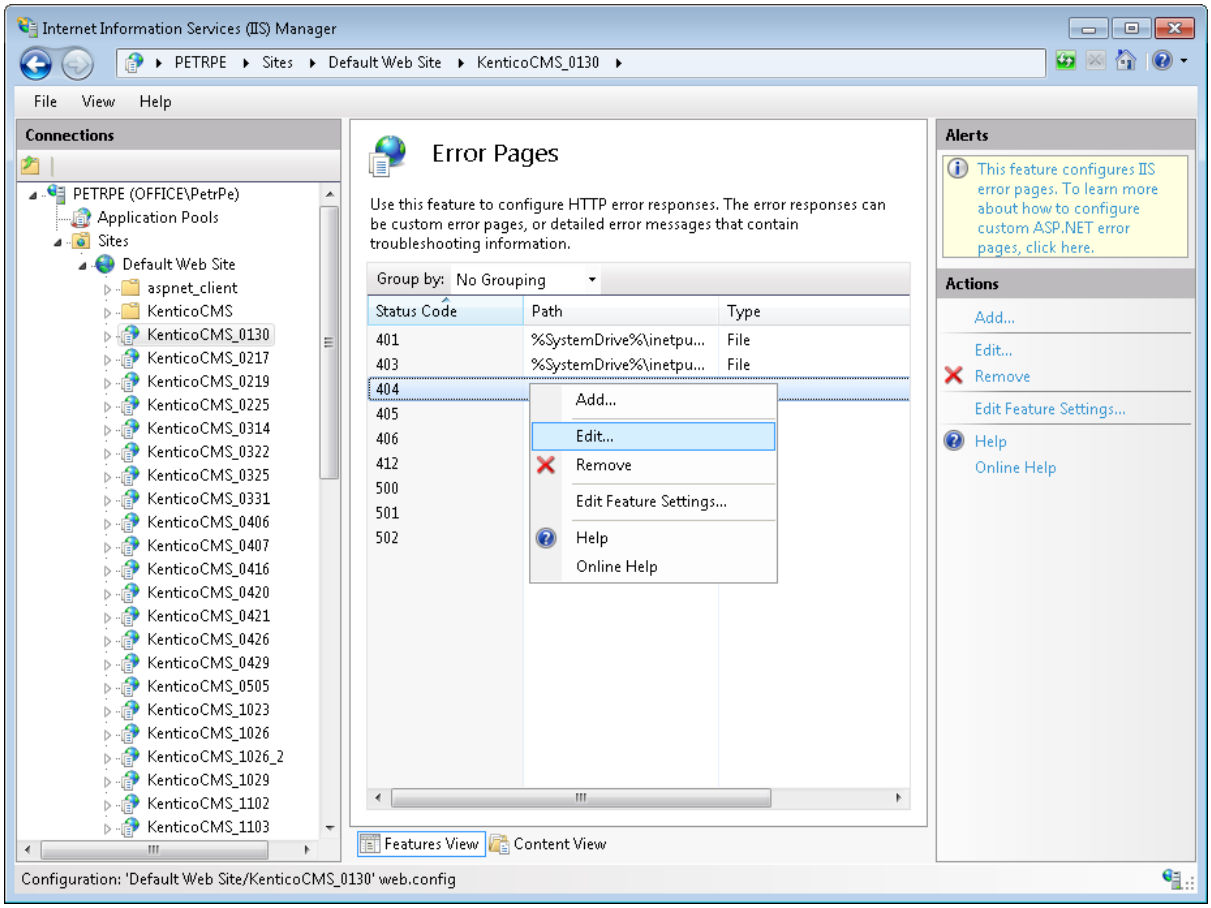
4. Enter the following values:

- **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

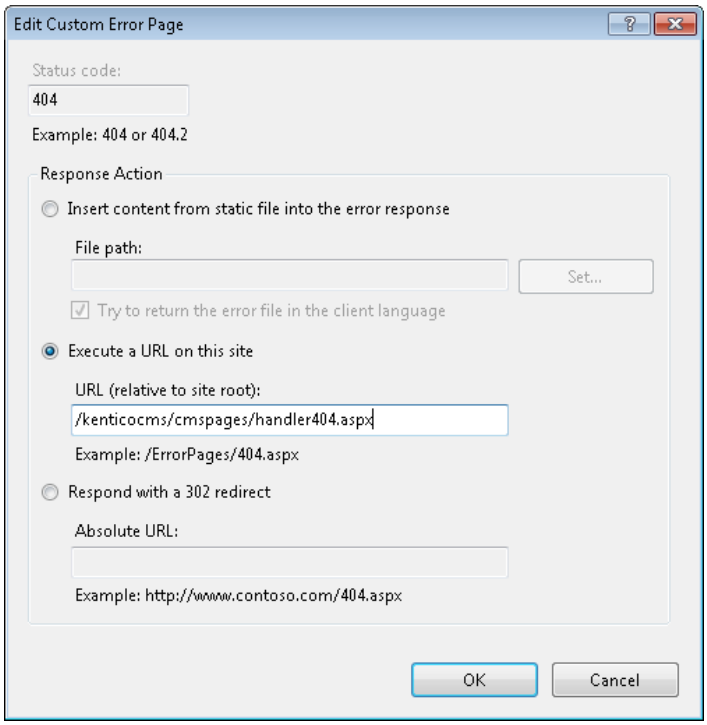
Click **OK**.



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.

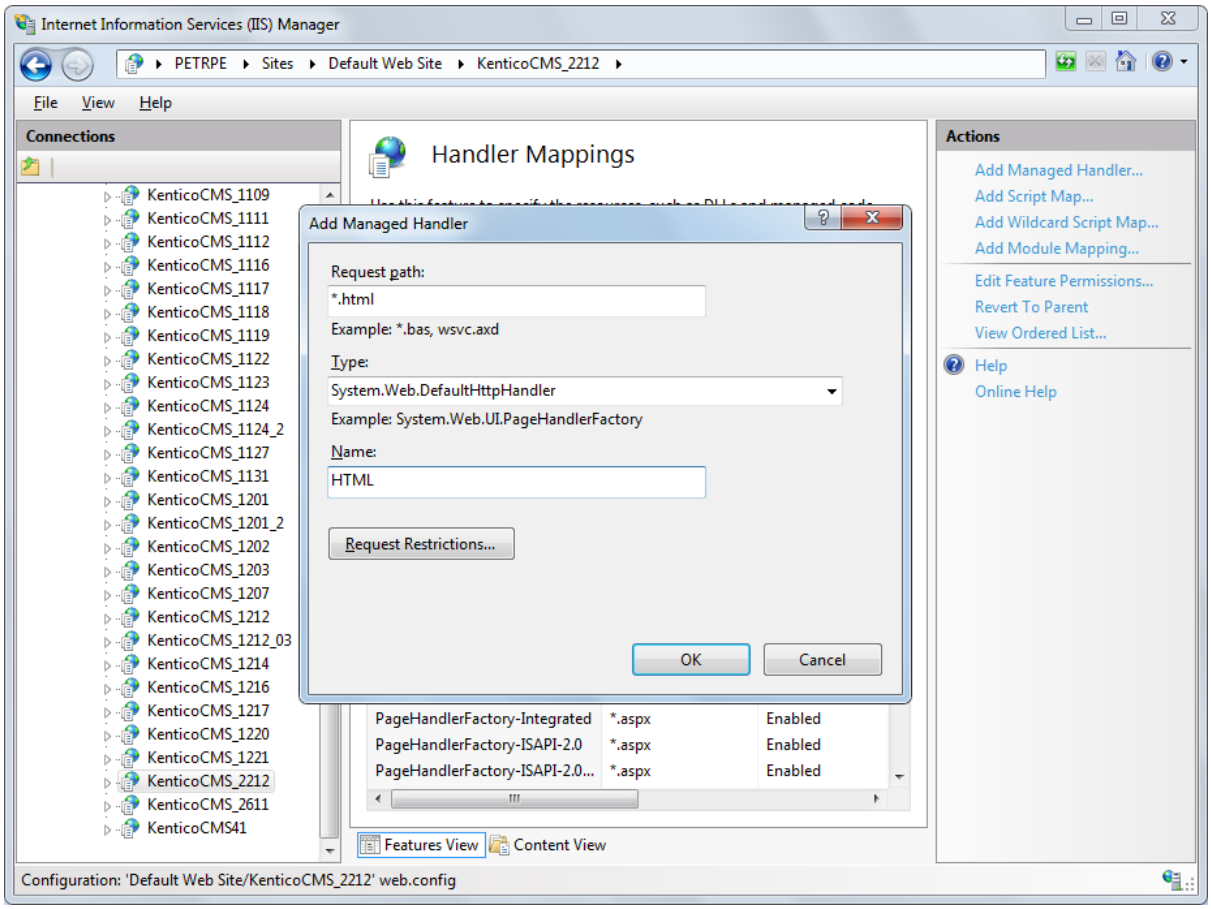
3. Open **Handler mappings**.

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

5. Enter the following values:

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

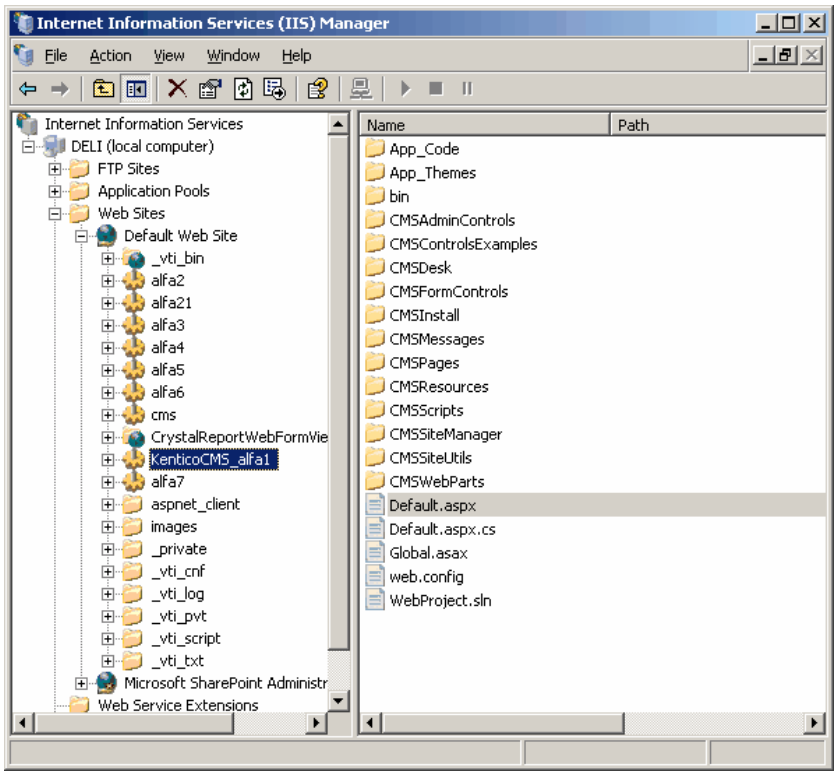
Click **OK**.



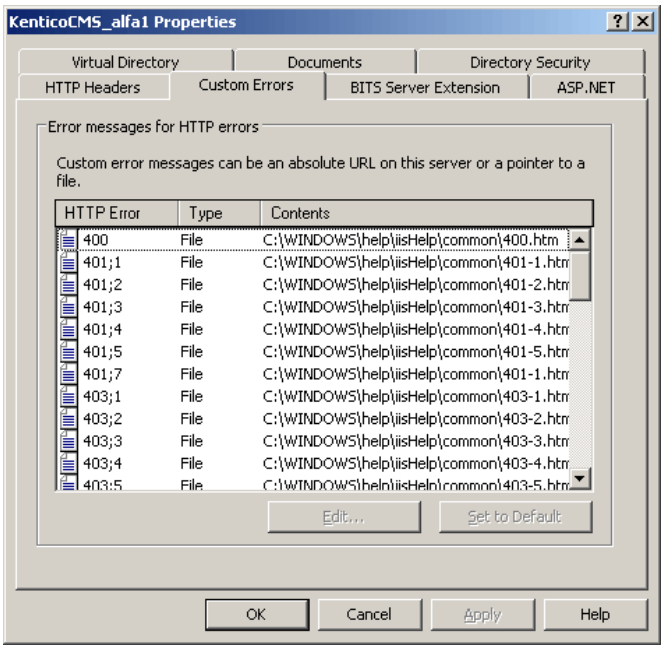
3.4.13.3 IIS 6

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).



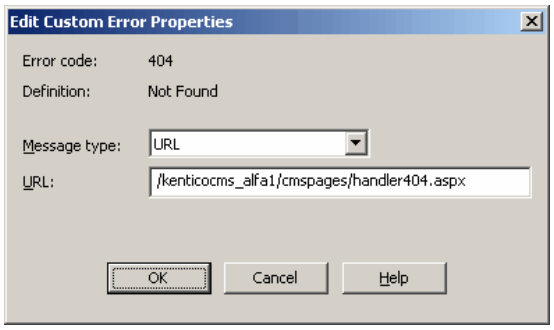
2. Right-click the directory (or website) and choose **Properties** and then click the **Custom Errors** tab:



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`



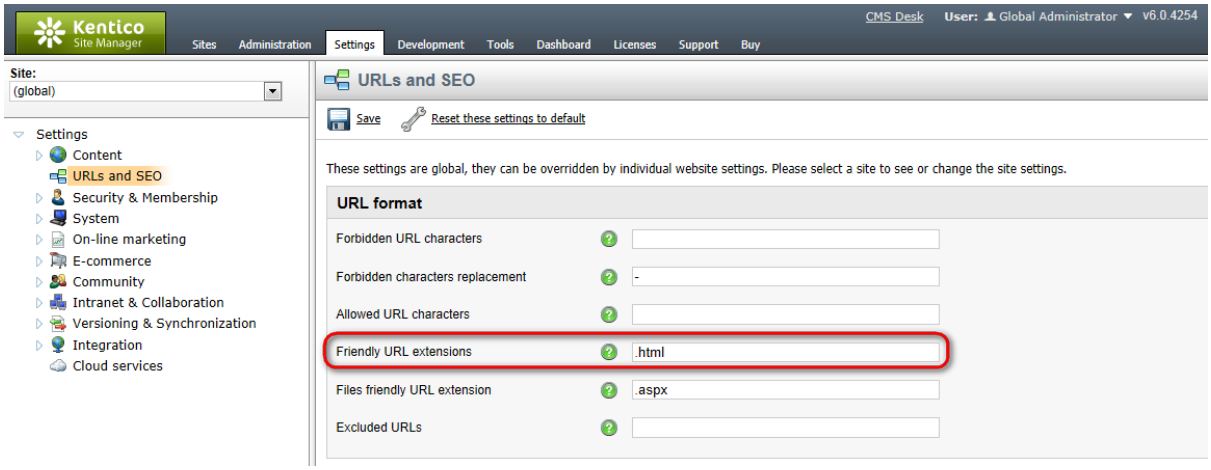
4. Now repeat the same for the 405 error, using the same custom URL: `/kenticocms/cmspages/handler404.aspx`

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

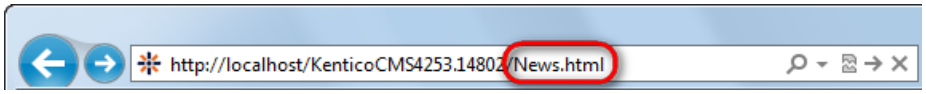
3.4.13.4 Configuration of custom URL extensions (.html or other)

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 use a semicolons without any extension in front of it, just as in the middle of the sample entry above, extension-less URLs can be achieved.

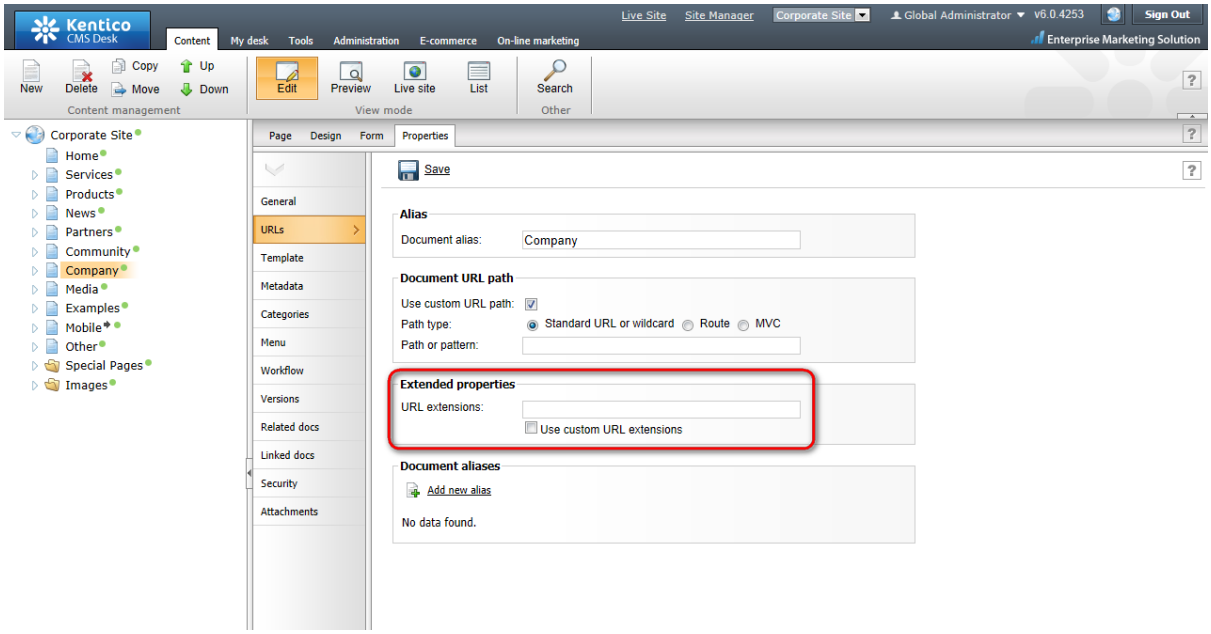
The **Redirect documents to main extension** setting, which can also be found at **Site Manager -> Settings -> URLs and SEO**, causes URLs with non-default extensions to be redirected to corresponding URLs with the current default extension. This can be useful for SEO purposes. For example, if you want 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.

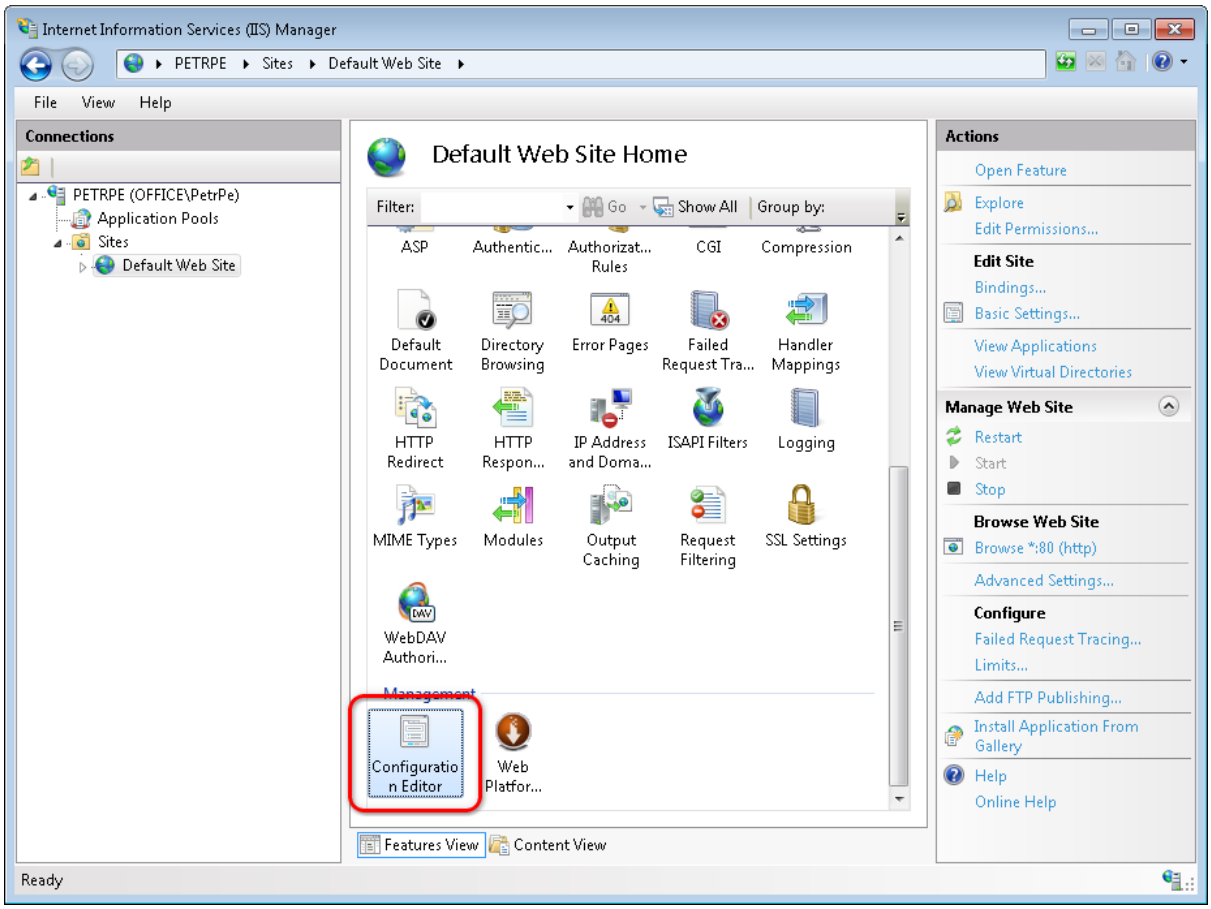
Please note: Even if the **Use custom URL extensions** option is disabled, files (**cms.file** documents) can be accessed under their physical extensions.



3.4.13.5 Lock violation on IIS7

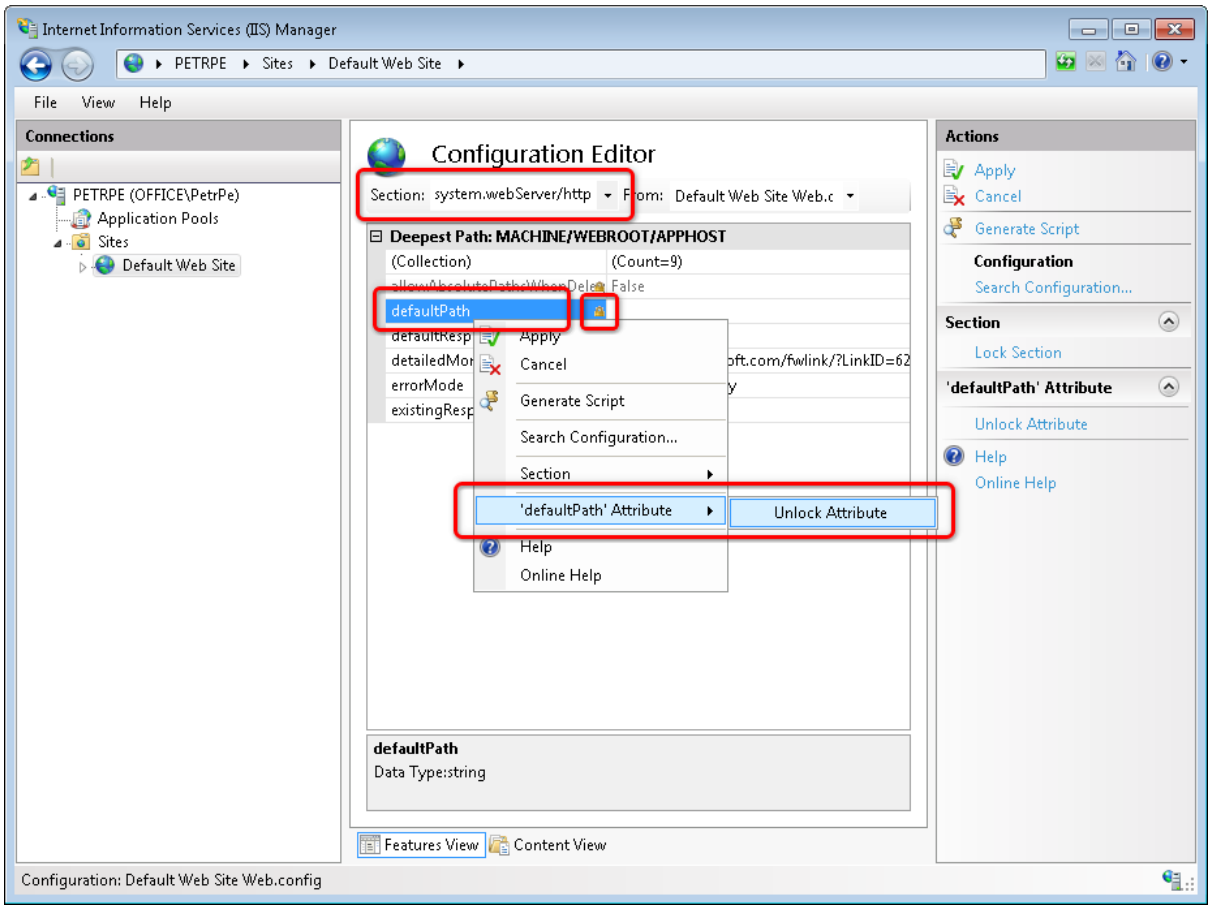
When configuring your IIS 7 (or later) to allow custom extensions or extension-less URLs, especially if you're running Kentico CMS in a virtual directory, you may receive the **Lock violation** error message. This typically doesn't allow you to specify the path settings in step 4 of the instructions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

3.5 Windows Azure deployment

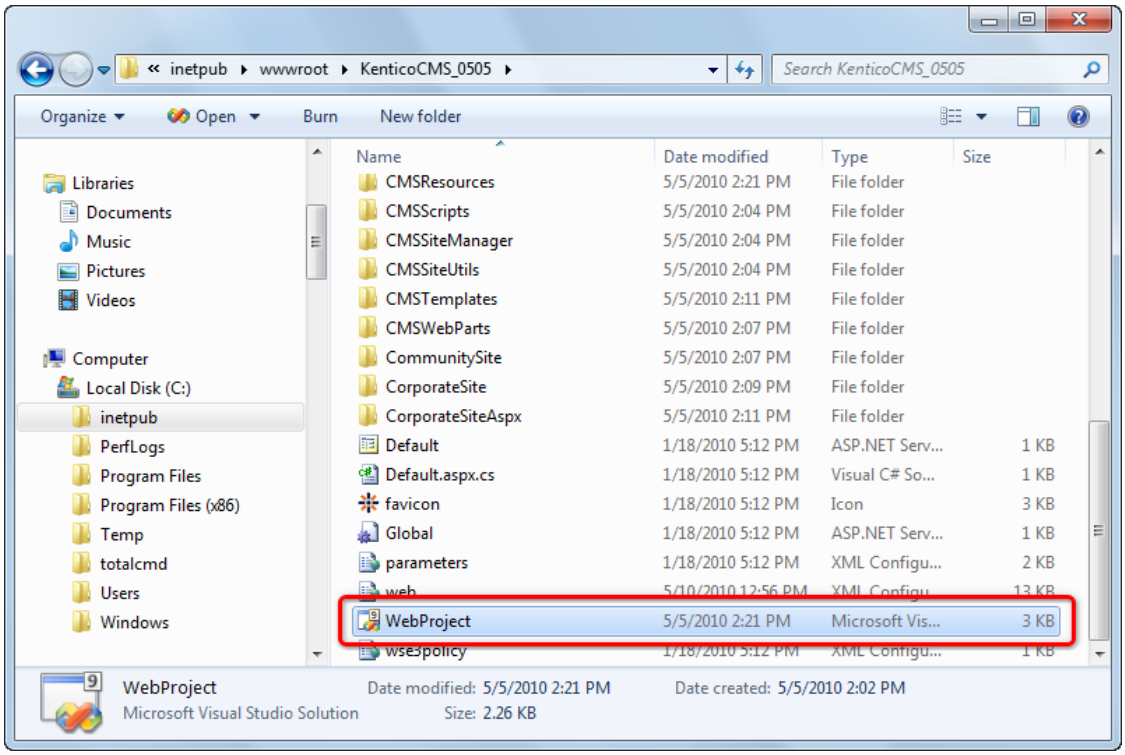
3.5.1 Deployment to Windows Azure

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.

3.6 Visual Studio integration

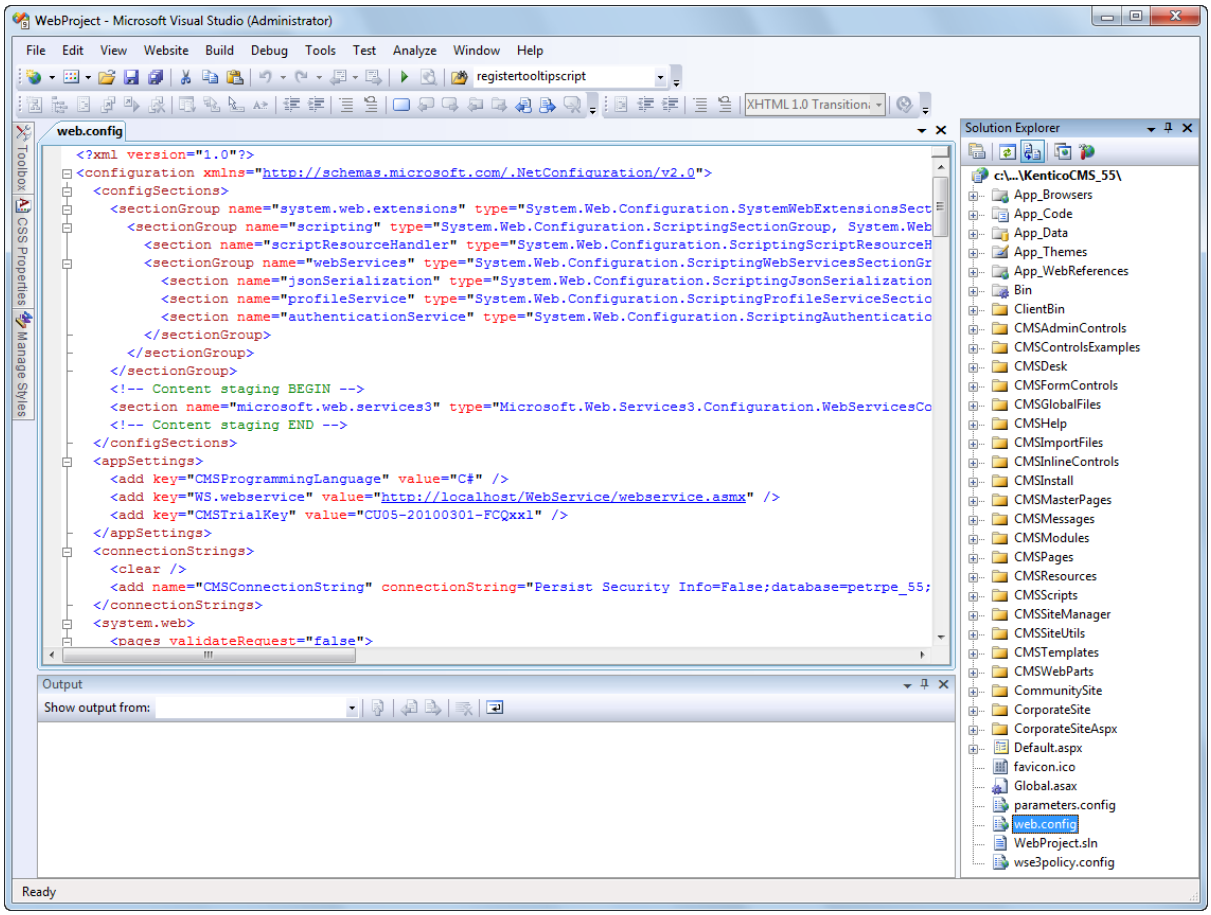
3.6.1 Opening the project

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.

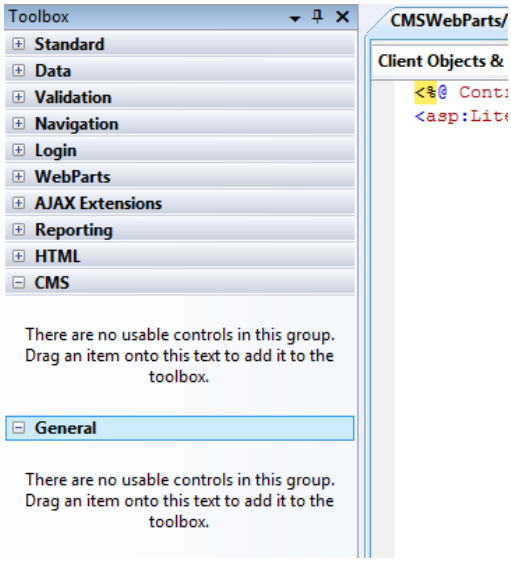
3.6.2 Adding Kentico CMS Controls to the Toolbox

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

1. Open the website project in Visual Studio and open some ASPX page.

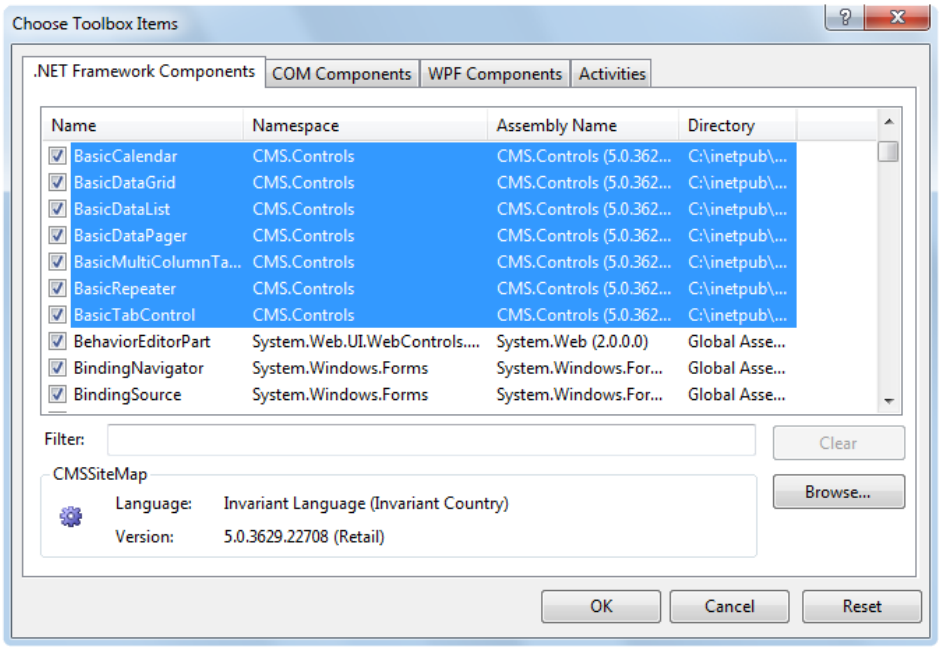
2. Right-click the **Toolbox** and choose **Add tab** from the context menu.

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

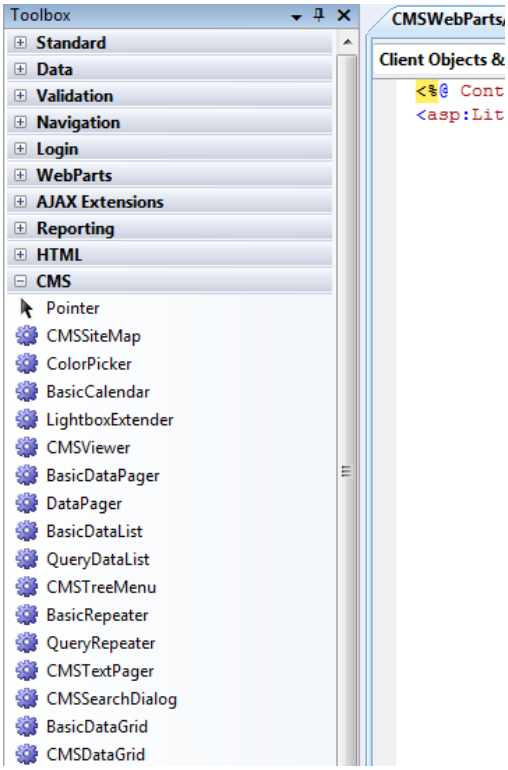


4. Right-click the new tab and choose **Choose items...** from the context menu.

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**.

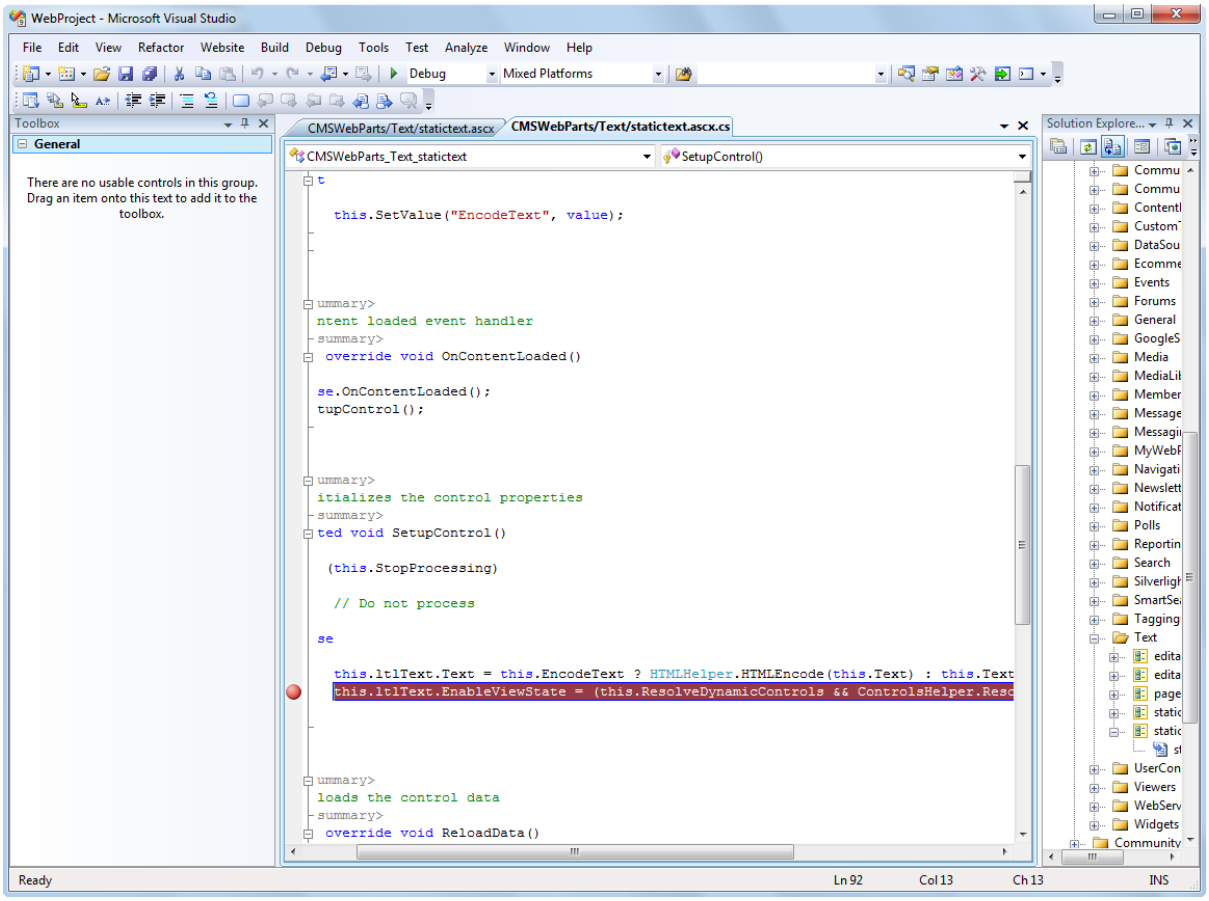


6. The controls are now added to the Toolbox, as you can see in the following screenshot. Now you can easily drag and drop the controls on your Web form.

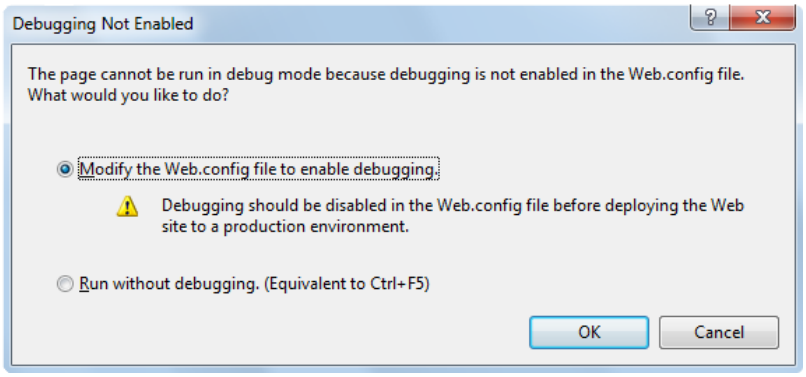


3.6.3 Debugging

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

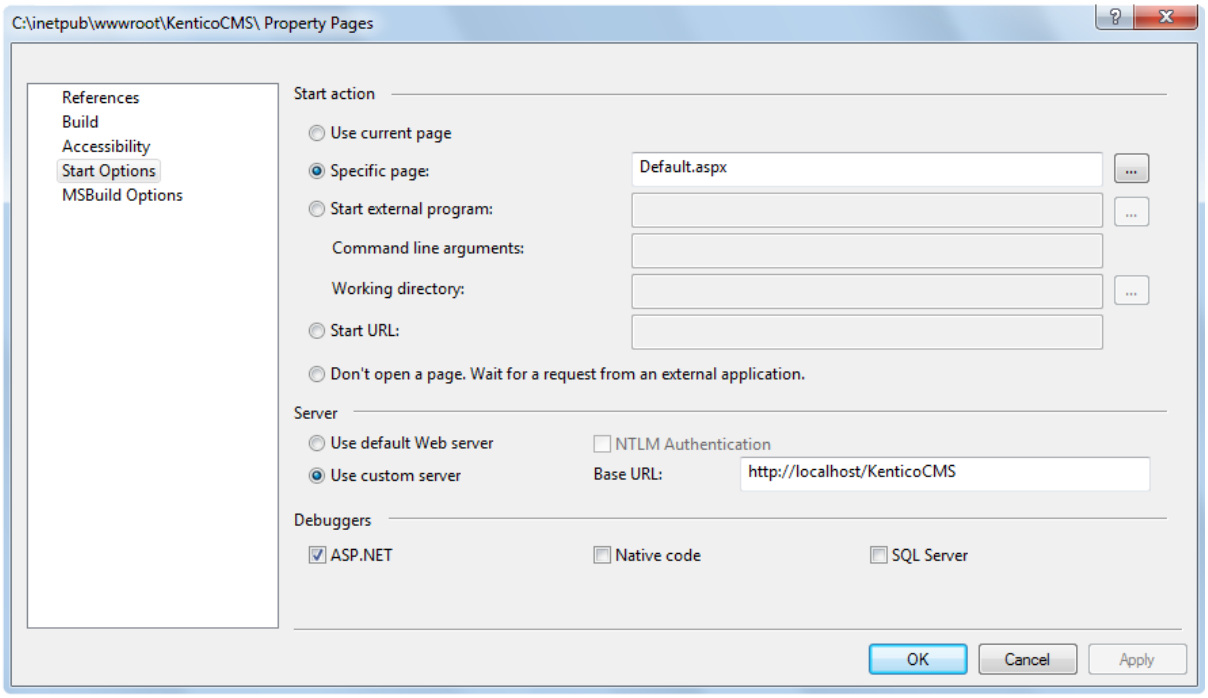
setting the value

```
<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 Pre-compilation (Publish function)

If you want to pre-compile the website before placing it on the live server, you can use Visual Studio's Publish function. It allows you to compile the source code into assemblies. This provides several advantages:

- faster application start (no compilation is required)

- intellectual property protection - if someone gets your website code, they will not be able to read the source code easily
- better security - the code cannot be easily modified by potential hacker

However, there's also a disadvantage: if you compile the website, you will not be able to use some of the Kentico CMS features:

- import of the website
- New site wizard.
- You cannot add code in the Code tab of web parts (the Code tab is an obsolete feature as of Kentico CMS 3.0, it is mentioned here due to backward compatibility reasons).

If you run a pre-compiled website, you will get the errors like:

The file '/CMSTransformations/cms/event/preview.ascx' does not exist.

This means that the system requires a virtual object, but the **VirtualPathProvider** is probably not running and cannot provide the system with that object.

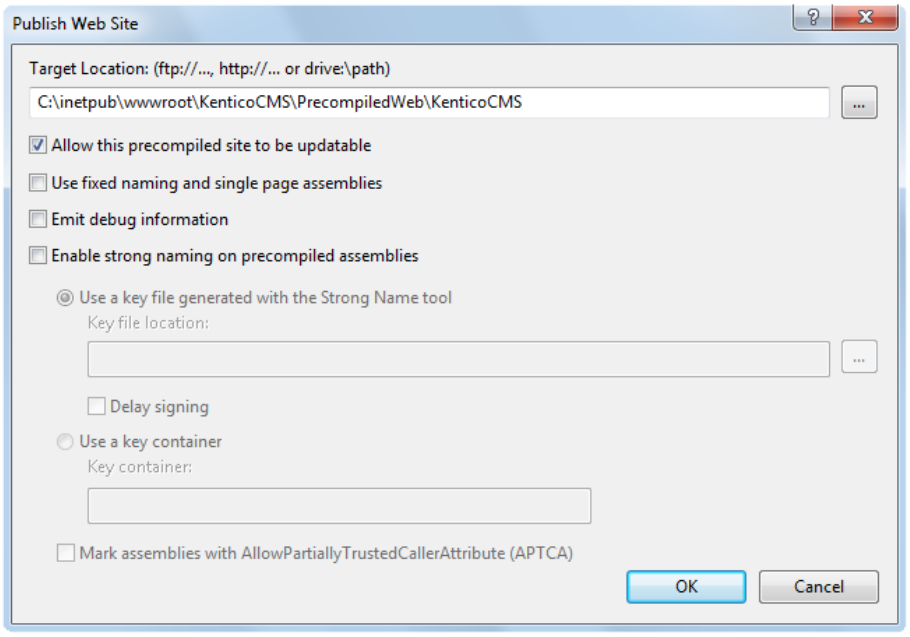
Pre-compiling the website

Please note that for building the pre-compiled website, you have to rename **Microsoft.Web.Services3.dll.rename** file to **Microsoft.Web.Services3.dll** in the ~*Bin* directory.

Since the virtual provider cannot run in a pre-compiled website, you need to store the physical files to the system before compilation. You can save all the virtual objects to the file system in **CMS Site Manager -> Administration -> System -> Deployment interface by clicking the button "Save all virtual objects to disk"**.

Please note that these files are just copies of the actual virtual object and will be used by the system only if the VirtualPathProvider cannot start. Also, the changes to the objects through the administration interface will not affect these files until you save all the objects to the disk again.

Then, you can use **Build -> Publish** item in **Visual Studio** main menu and pre-compile the website. You will see a dialog like this:



You need to enter the path outside your current web project where the compiled version will be placed. You can choose between two compilation modes using the **Allow this precompiled site to be updatable** checkbox:

- **Allow the site to be updatable (checked)** - the code behind and any classes in the *App_code* folder are compiled, but ASPX files are stored in their original source code form.
- **Do not allow the site to be updatable (unchecked)** - the code behind and *App_code* classes, as well as ASPX files are compiled and the website contains only empty "stub" ASPX files without any code.

We generally recommend that you do not allow the site to be updatable, since it provides the best performance.

Limitations

In the precompiled website, the VirtualPathProvider is stopped automatically. When VirtualPathProvider is stopped, **you cannot edit transformations and layouts through the user interface** without saving them on the disk again.

You may need to copy the database to the server using the standard **backup/ restore** operation since the compiled website cannot be used for SQL Server database installation. Other options are:

- to 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.
- to install Kentico CMS website locally and run the database setup against the remote SQL Server on the live server.

In the portal development model, you cannot use **custom web part code** (**Web part**

properties –> Code tab). If you need to add custom code on the Code tab and run the website in the compiled version, you need to create user controls, place web parts to the user controls and add your custom code to the web parts. Then, you can place the user controls to the page using the General/User control web part.

Testing the website before compilation

If you wish to check if your website runs without using **VirtualPathProvider** (to simulate the 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" />
```

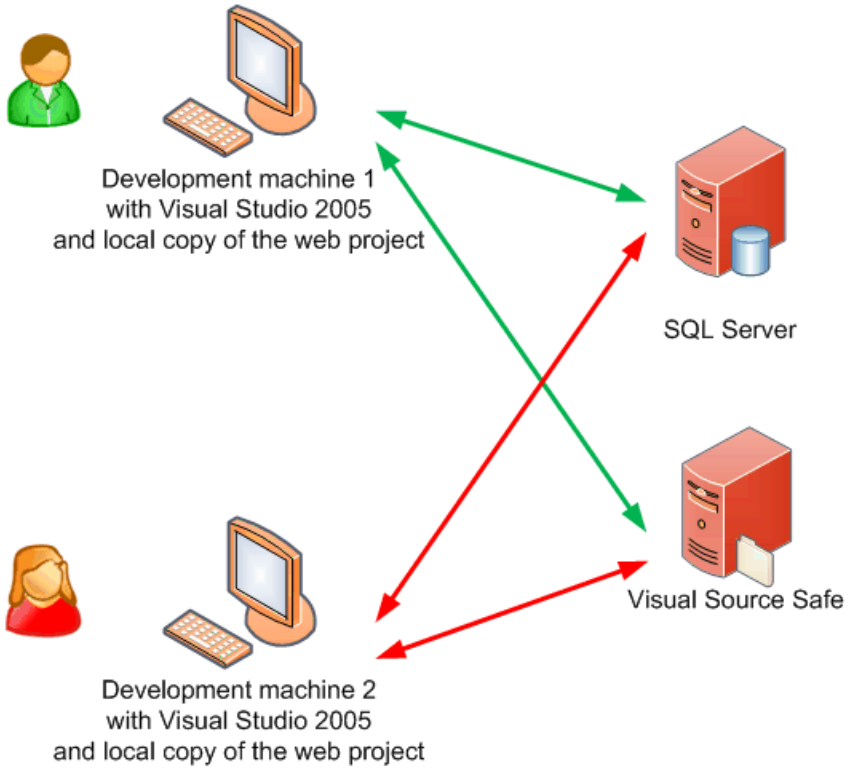
If your system runs with VirtualPathProvider disabled, you should be able to run it after the precompilation or deployment process.

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. That's why 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.

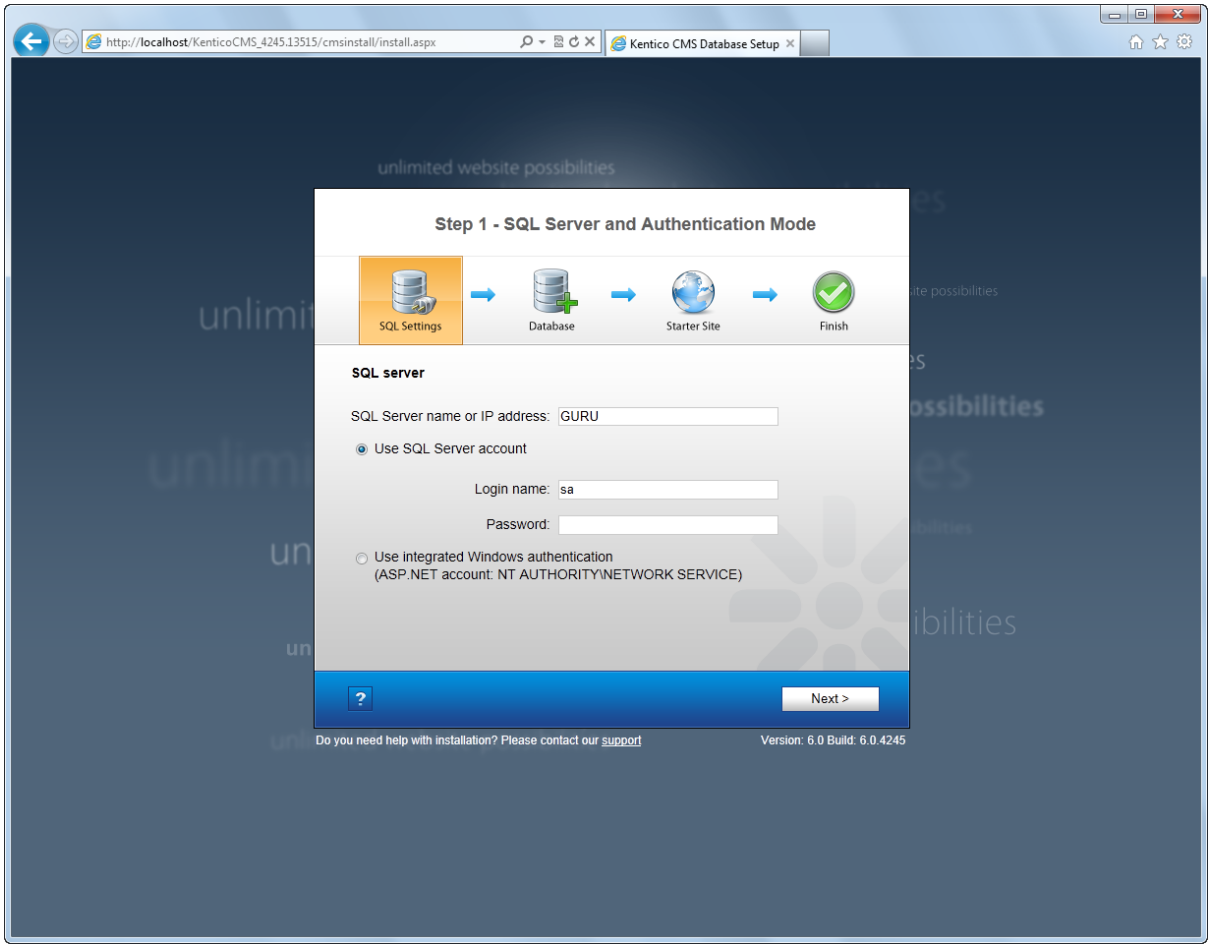
You may encounter problems in following areas:

- 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)

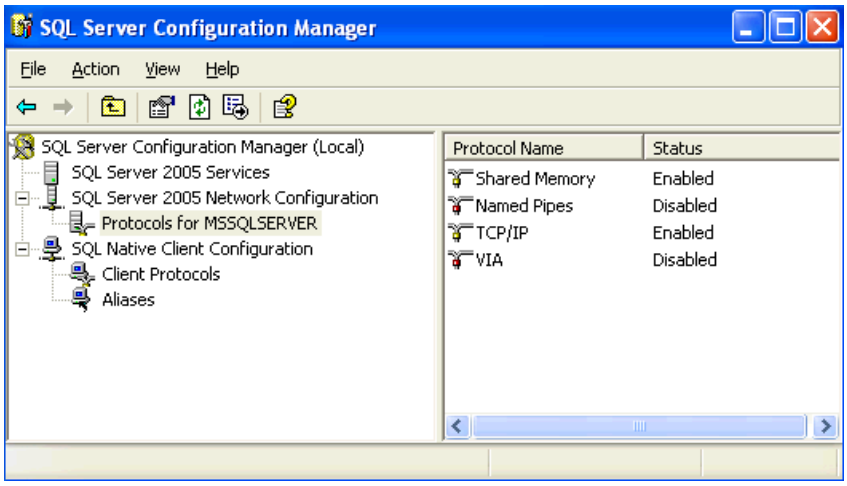
2. Make sure the server has Microsoft SQL Server 2005 or 2008 installed and running.

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 ****).

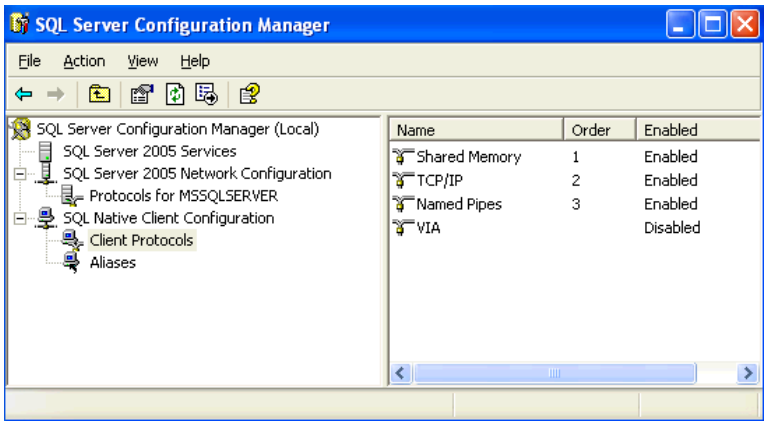
4. If you're using Microsoft SQL Server Express 2005 with default installation settings, the correct server name is **.\sqlexpress** or **computername\sqlexpress** .

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:



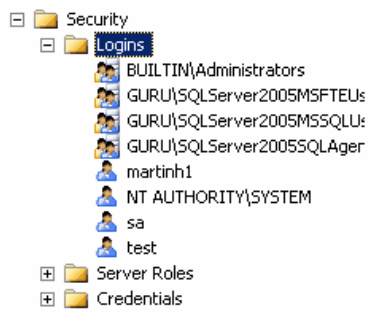
Error 2: Login failed for user 'xy'

Error message:

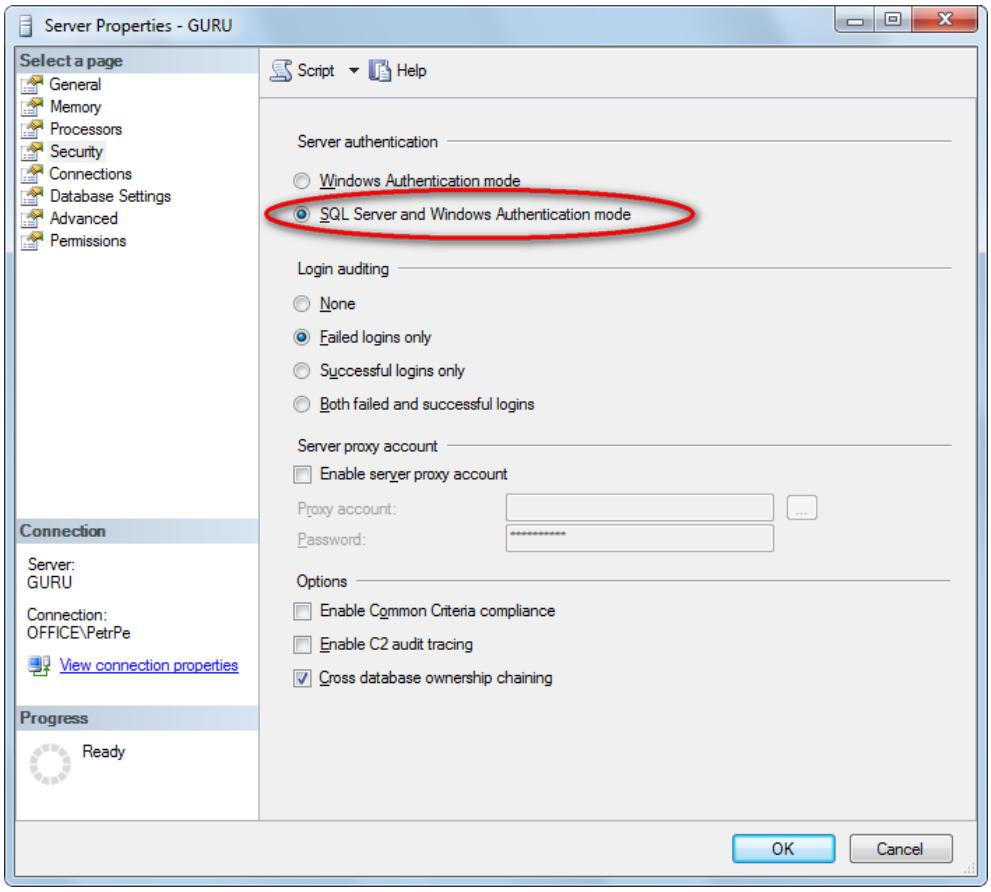
```
Login failed for user 'xy'
```

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:

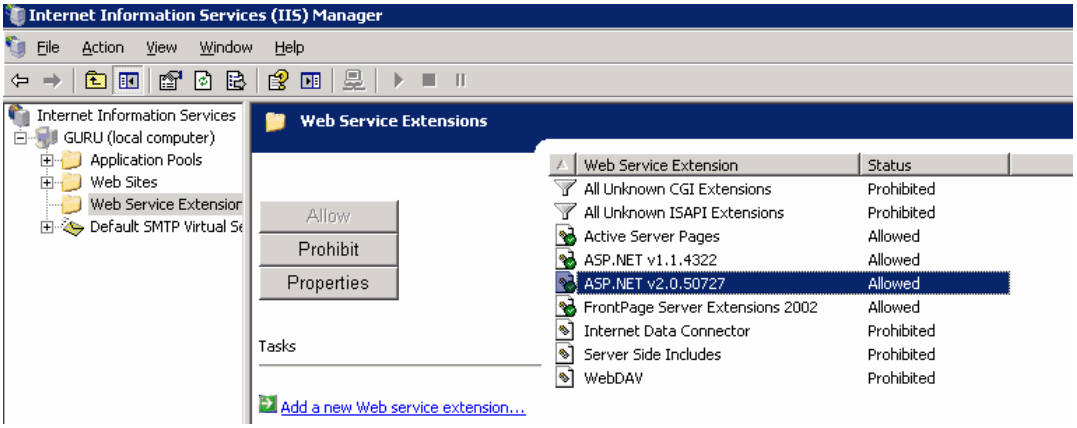
This account must have its own login with Windows authentication in the SQL Server. You can create the login in **Enterprise Manager\SQL Server Management Studio -> Security -> Logins** and grant appropriate permissions on the server to it. If you SQL Server is located on a different machine than your 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:

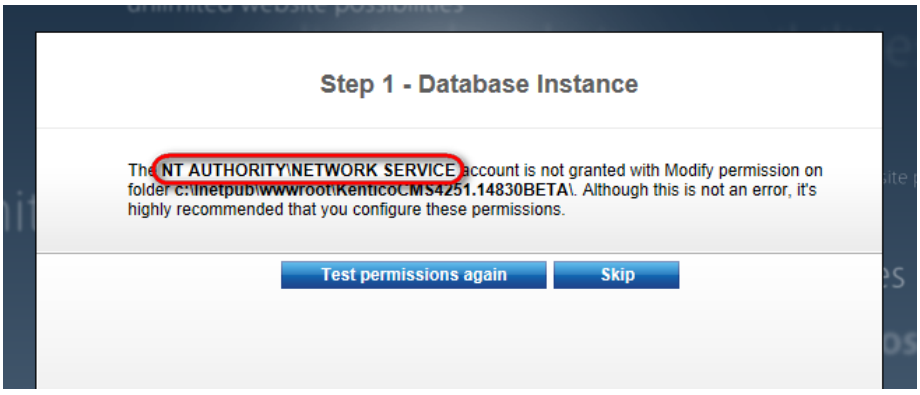


3.7.4 Disk permissions problems

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.

1. On Windows 7, the user account is the local **IIS_IUSRS** account by default.
2. On Windows Vista or Server 2008, the user account is the local **NETWORK SERVICE** account by default.
3. On Windows XP, the user account is the local **ASPNET** account (aspnet_wp) by default.
4. On Windows 2003, the user account is the local account **NT Authority\Network Service** by default.
5. 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.

Choosing the component for directory operations

If you're running Kentico CMS under restricted trust level, you may need to use the managed component for directory operations (create/delete/rename directory). You can configure it by setting the following web.config parameter:

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

If you're running Kentico CMS on a shared hosting server, some providers require that you use the non-managed methods for directory operations:

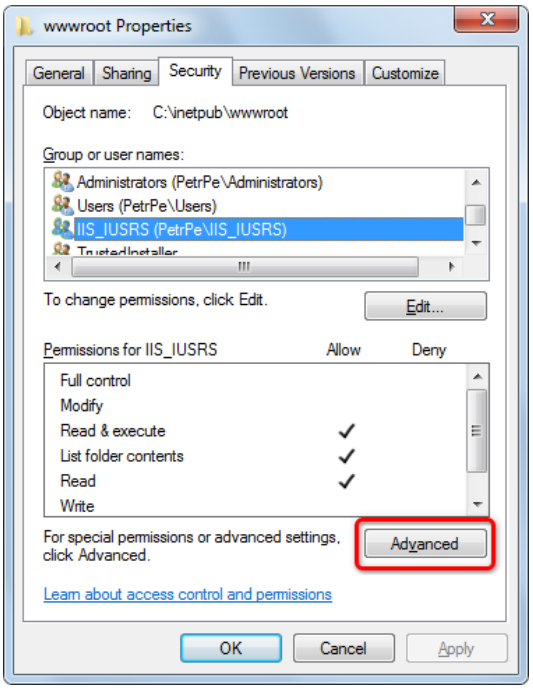
```
<add key="CMSDirectoryProviderAssembly" value="CMS.DirectoryProviderWin32" />
```

3.7.4.2 Solution on Windows 7

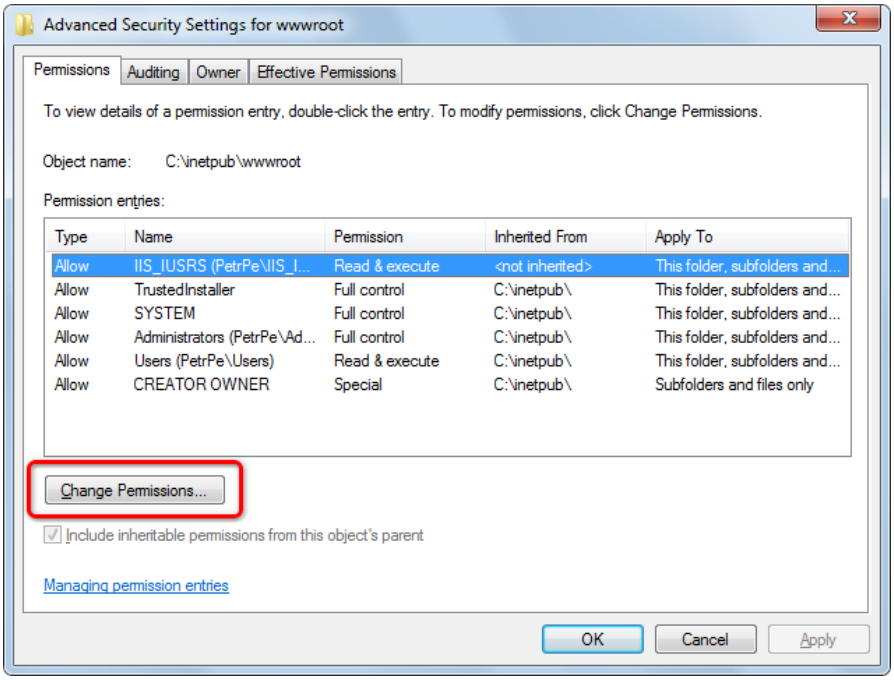
1. Open Windows Explorer, locate the folder with your website, right-click the folder and display its **Properties**. Choose the **Security** tab.

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.

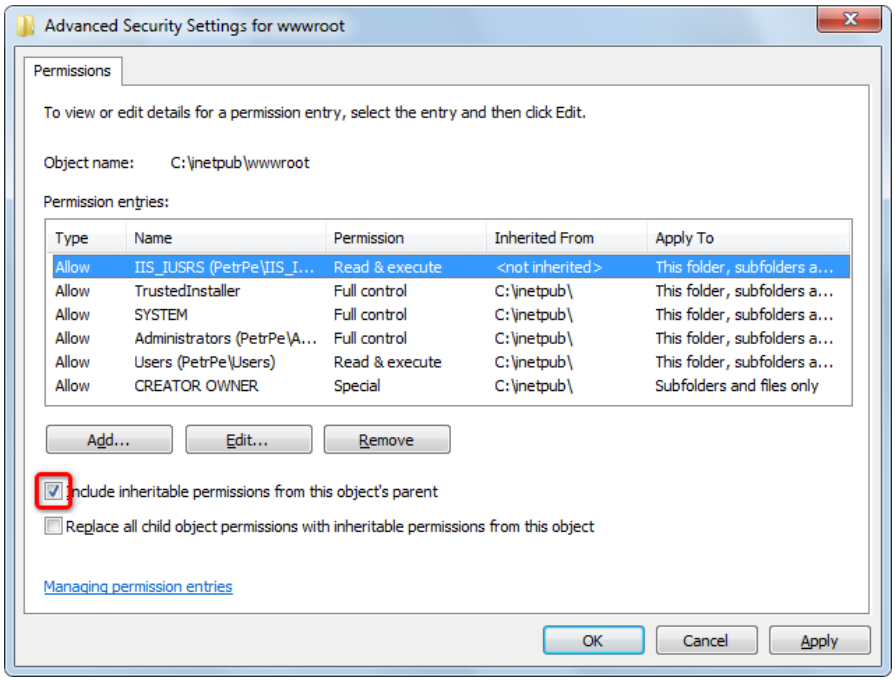
3. Back on the **Security** tab of the folder properties, select the appropriate account and click **Advanced**
.



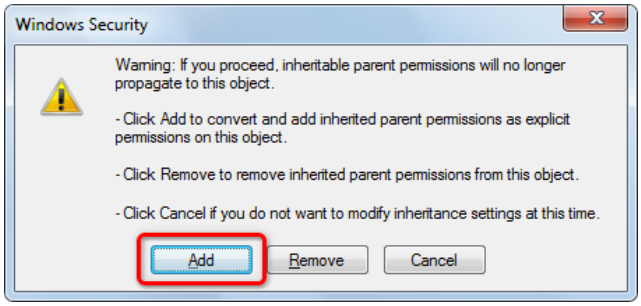
4. In the pop-up dialog, select the account again and click **Change Permissions**.



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

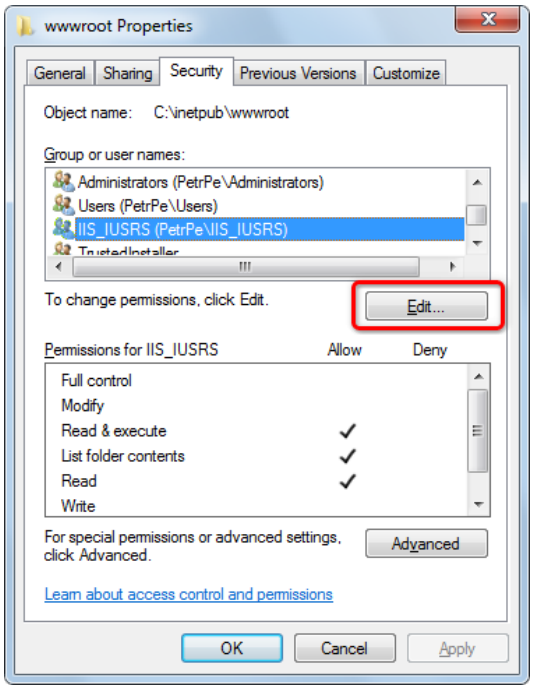


6. In the pop-up dialog, click **Add**.

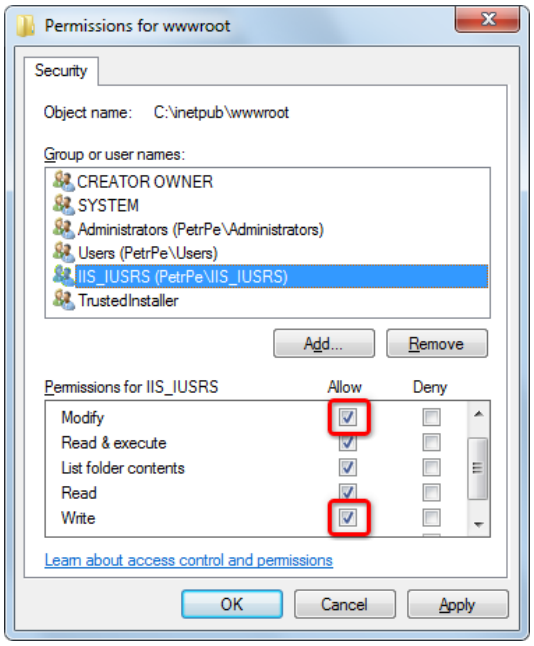


7. When the dialog closes, click **OK** to close the dialog under it. Click **OK** again to close the dialog under the previous one.

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**.



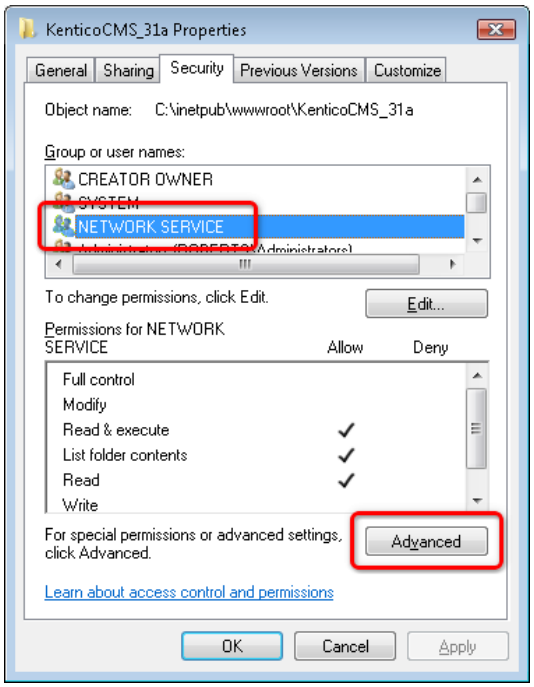
9. You have assigned the account with the required permissions. Kentico CMS should now be able to perform all disk write operations and therefore work correctly.

3.7.4.3 Solution on Windows Vista or Server 2008

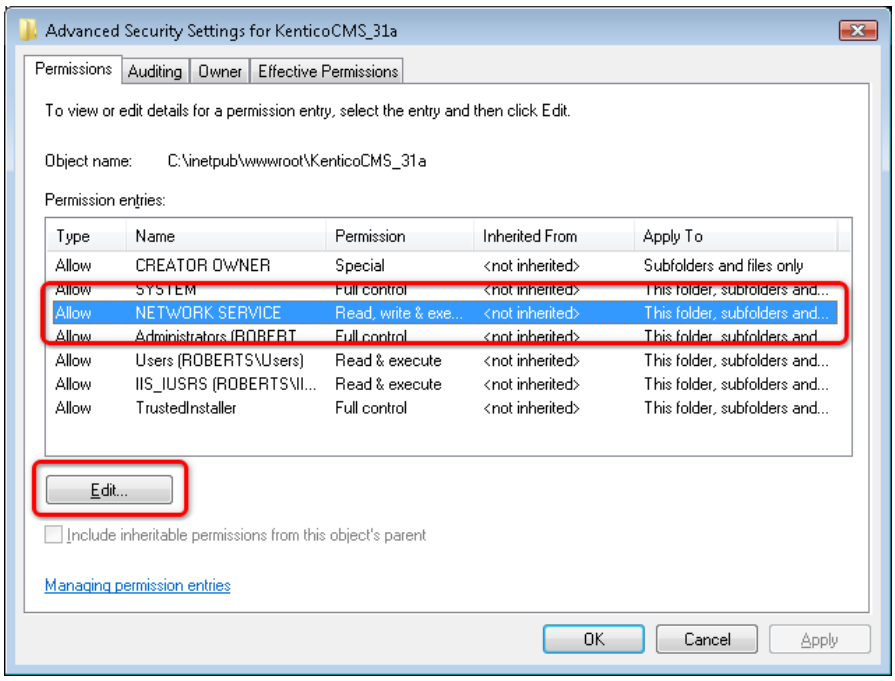
1. Open Windows Explorer, locate the folder with your website, right-click the folder and display its **Properties**. Choose the **Security** tab.

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.

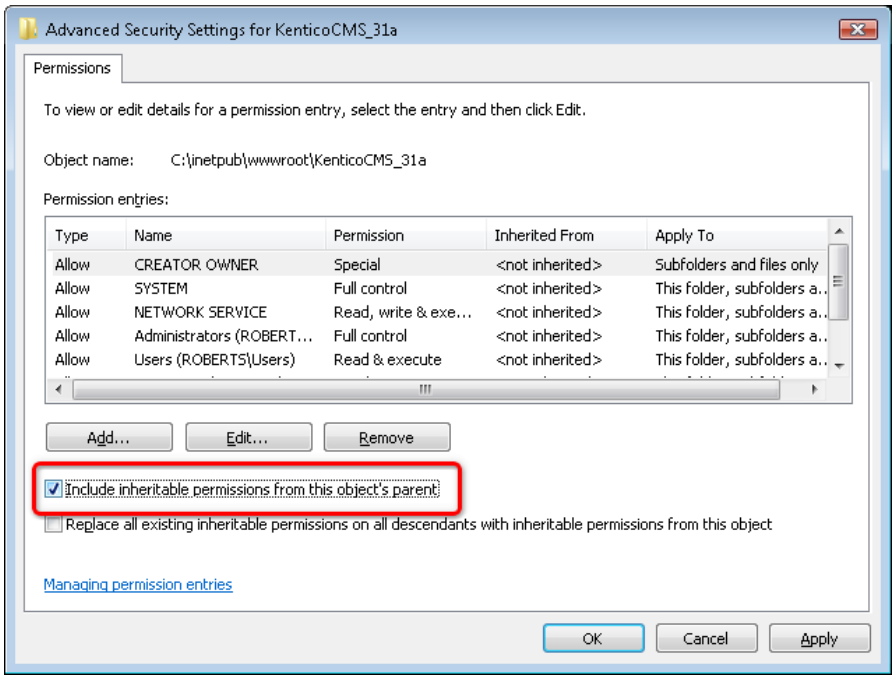
3. Select the required account and click **Advanced**.



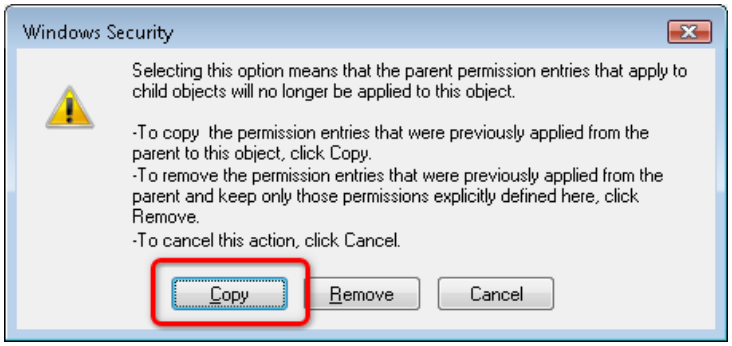
4. In the pop-up dialog, select the account again and click **Edit**.



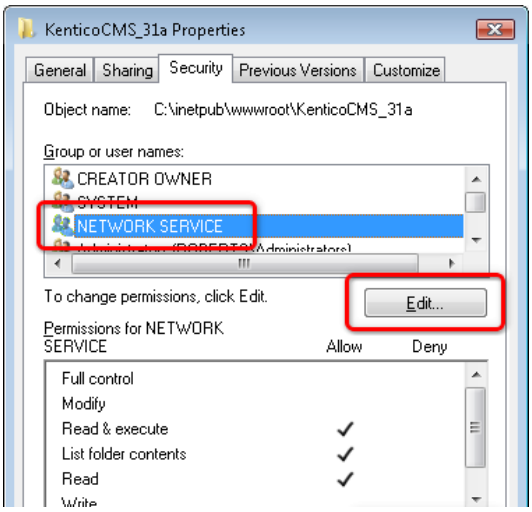
5. Disable the **Include inheritable permissions from this object's parent** option.



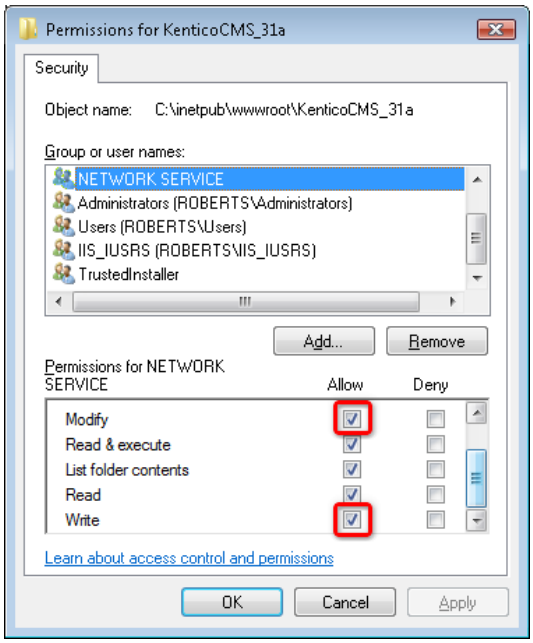
6. On the pop-up dialog, click **Copy**.



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**.



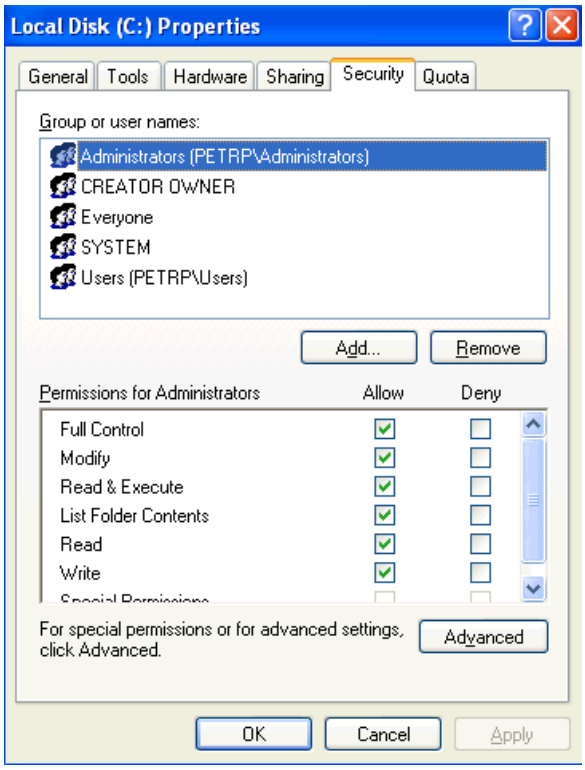
8. Check the **Allow** check-box for the **Write** and **Modify** permission and click **OK**.



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.

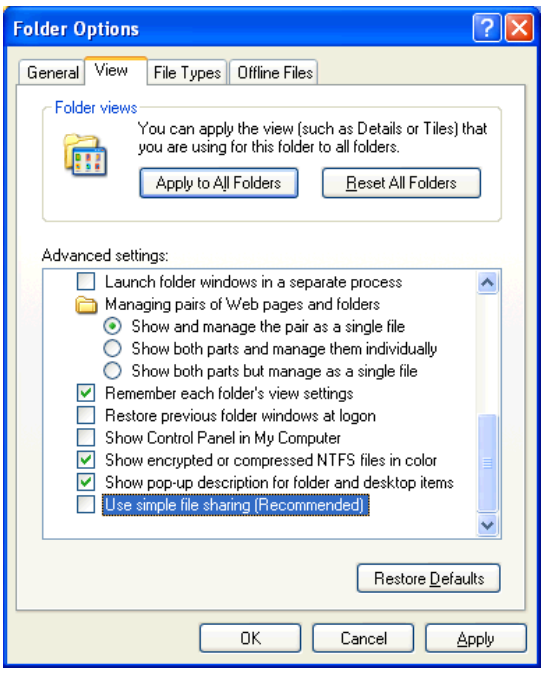
3.7.4.4 Solution on Windows XP

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

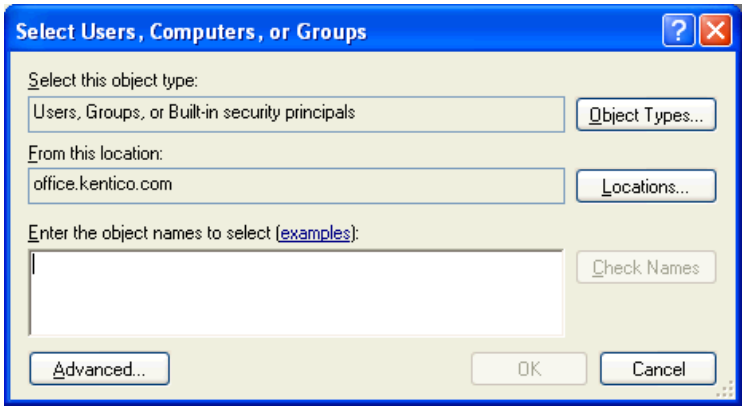


Missing Security tab in folder properties dialog

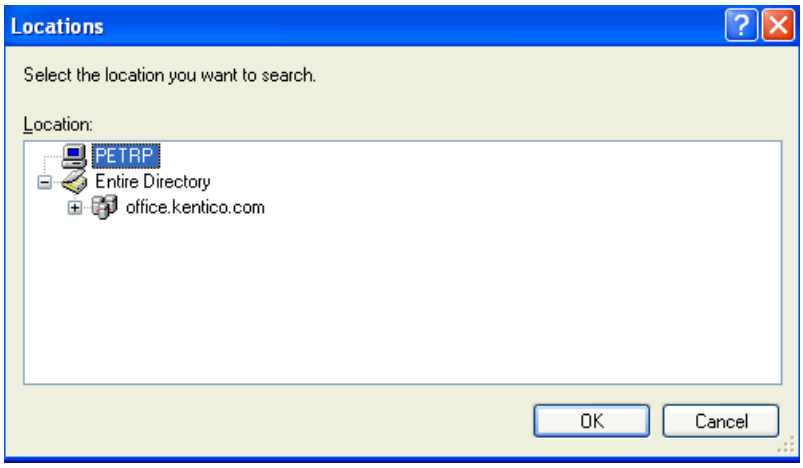
If you cannot see the Security tab, click **Tools -> Folder options** in the Windows Explorer main menu, choose the **View** tab and uncheck the **Use simple file sharing** box. Click OK. Now you should find the Security tab in the folder properties dialog.



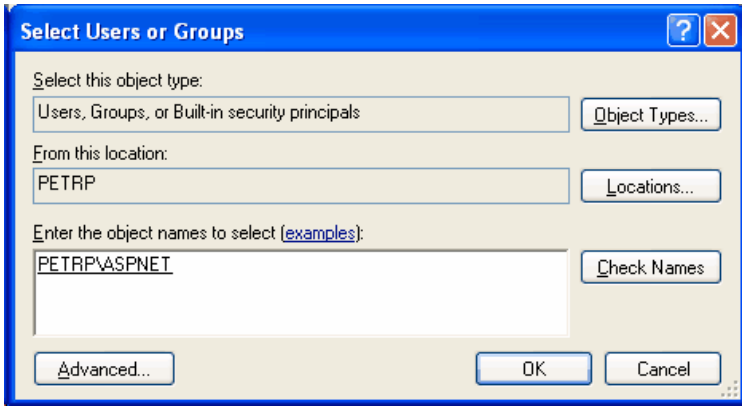
2. The **Select Users, Computers and Groups** dialog appears. Click **Locations...**.



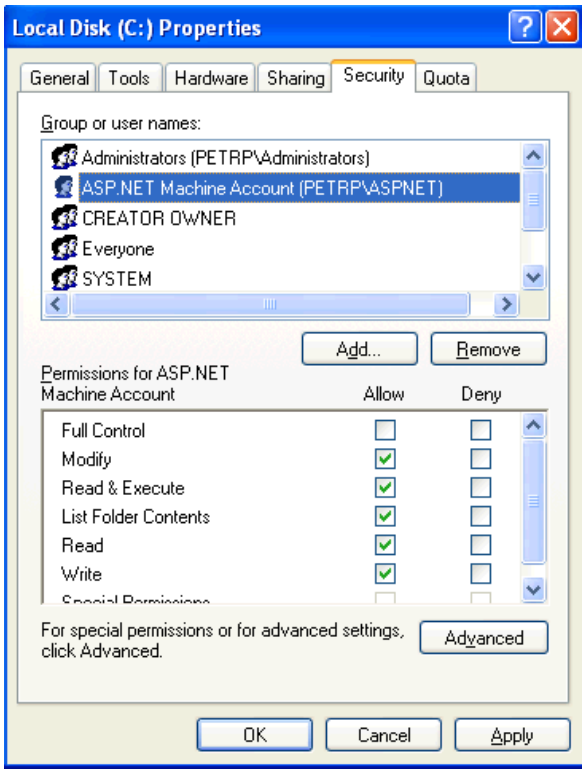
3. Choose your local computer in the pop-up window and click **OK**.



4. Enter **aspnet** into the box and click **Check Names**. The name should be resolved to **<your computer name>\ASPNET**. Click **OK**.

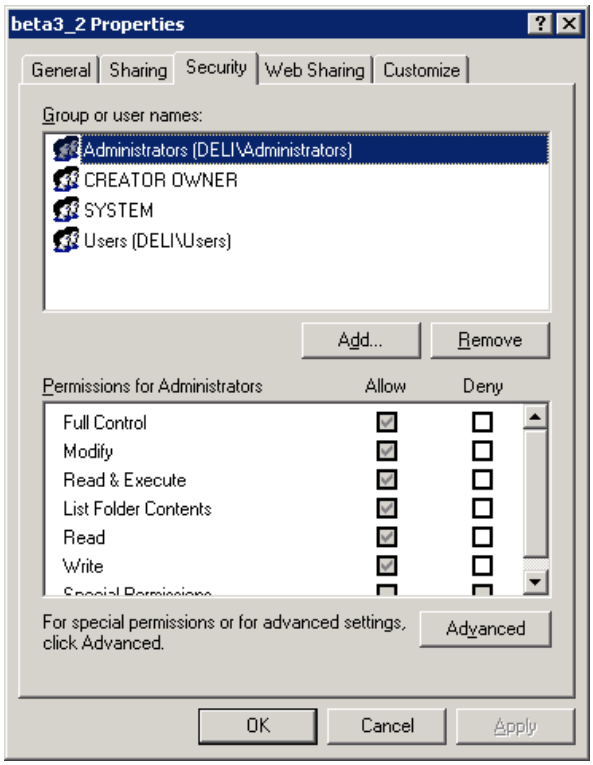


5. The account is added to the list of accounts. Grant the **Modify** permission to the account and click **OK**.

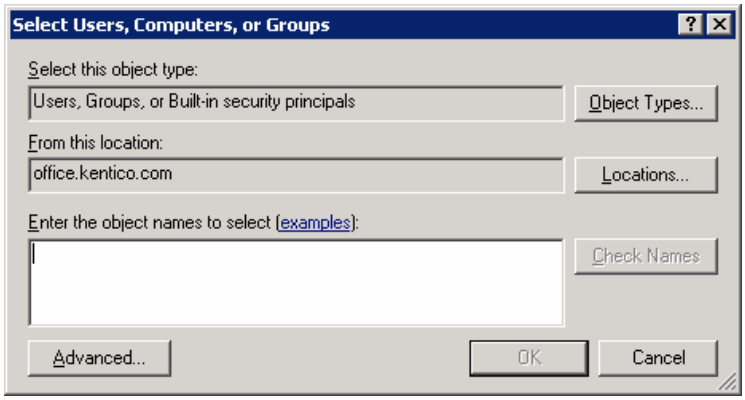


3.7.4.5 Solution on Windows 2003

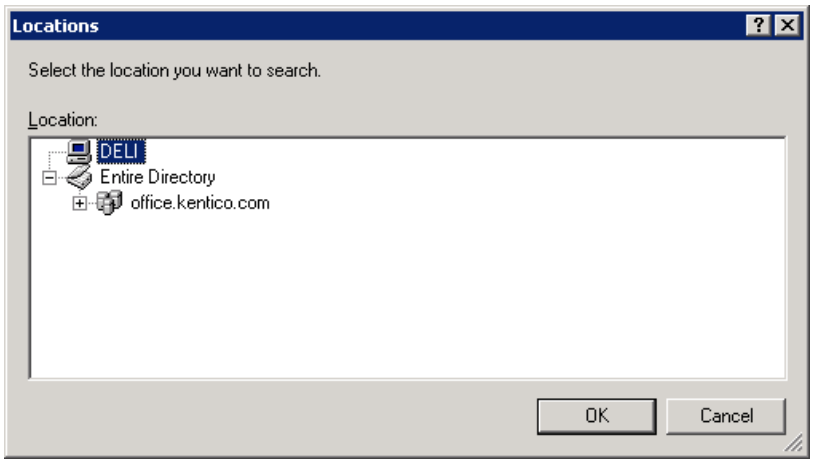
1. Open Windows Explorer, locate the folder with your website, right-click the folder and display its **Properties**. Choose the **Security** tab and click **Add...**.



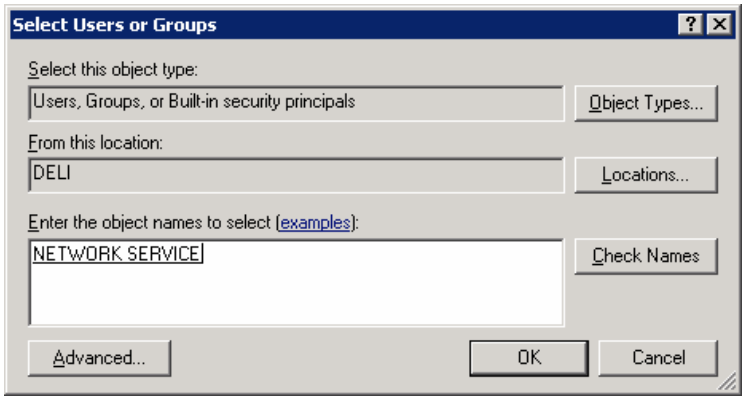
2. The **Select Users, Computers and Groups** dialog appears. Click **Locations...**.



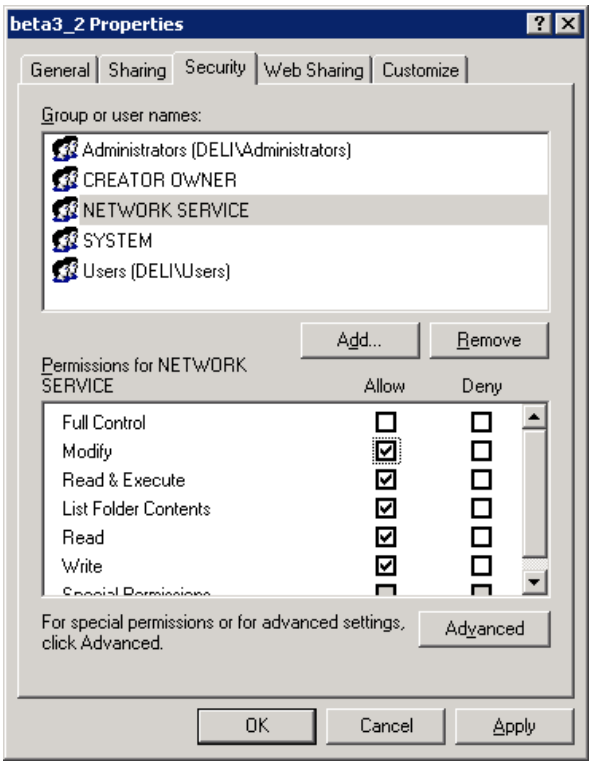
3. Choose your local computer and click **OK**.



4. Enter *network service* into the box and click **Check Names**. The name should be resolved to **NETWORK SERVICE**. Click **OK**.



5. The account is added to the list of accounts. Grant the **Modify** permission to the account and click **OK**.



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.

Part

Content management

4 Content management

4.1 Overview

The content can be managed through a browser-based, WYSIWYG user interface. You need to sign in to Kentico CMS Desk using your user name and password and you need to have appropriate permissions.

Links for access to the user interface

Kentico CMS user interface, i.e. **CMS Desk** and **Site Manager**, can be accessed via the following links, where the first part of the link is dependent on the installation and domain name. If the default URL of your website is *http://localhost/KenticoCMS*, then:

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

Please note: The beginning of the addresses may be different depending on your installation and domain name, but the highlighted end of the URL is always in this format.

Default user name and password

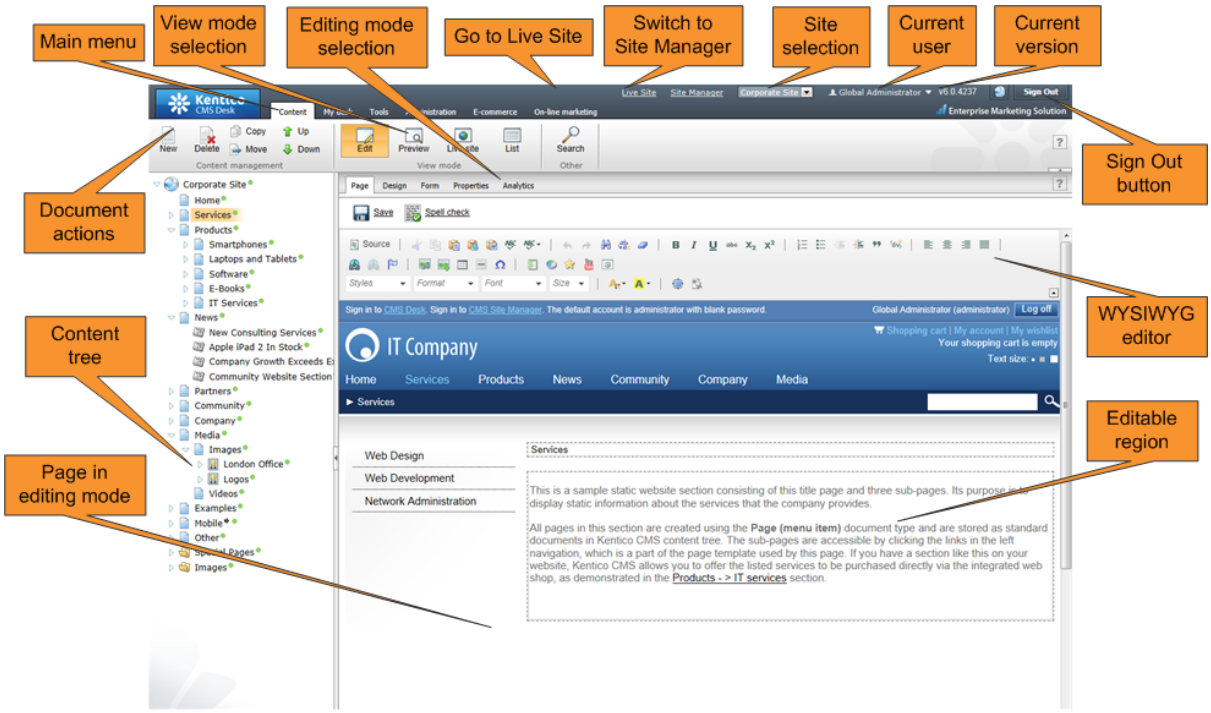
To log on to a Kentico CMS website for the first time, use the default logon credentials:

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

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

User interface overview

The following figure shows the CMS Desk user interface.



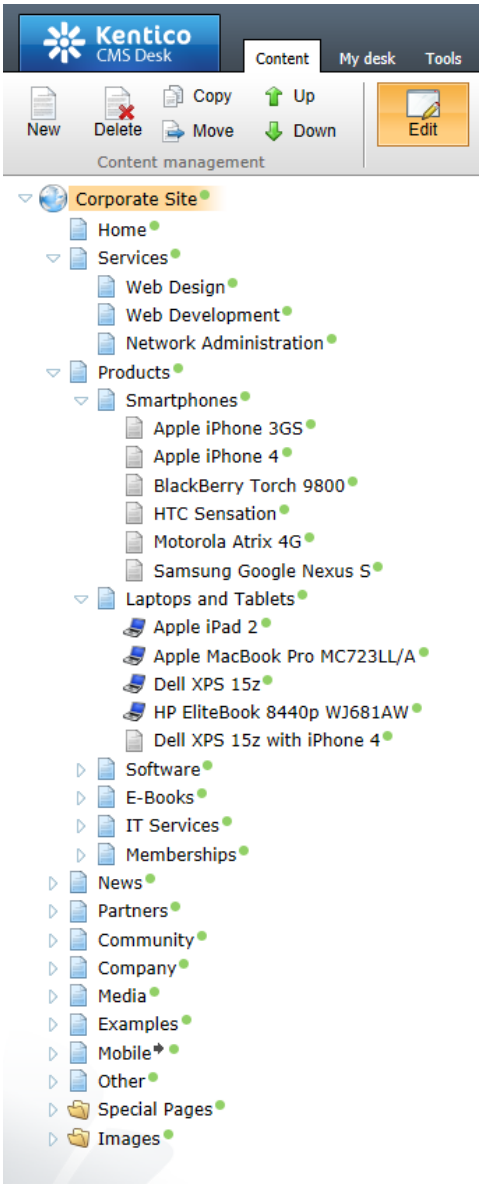
The user interface consists of the following main sections and features:

- **Main menu** with Content, My Desk, Tools and Administration tabs.
- **Content tree** that represents the site map of the website and allows you to organize the structure of documents and choose document that appears on the right side of the screen.
- **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 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 the user interface. 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 user interface. This option is only available to 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 drop-down list.
- **Current user** - user name of the current user.
- **Current version** - version of Kentico CMS.
- **Sign Out button** - clicking this button log you off 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 stored in **Editable regions**, change text formatting or insert graphics into the text.
- **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**:

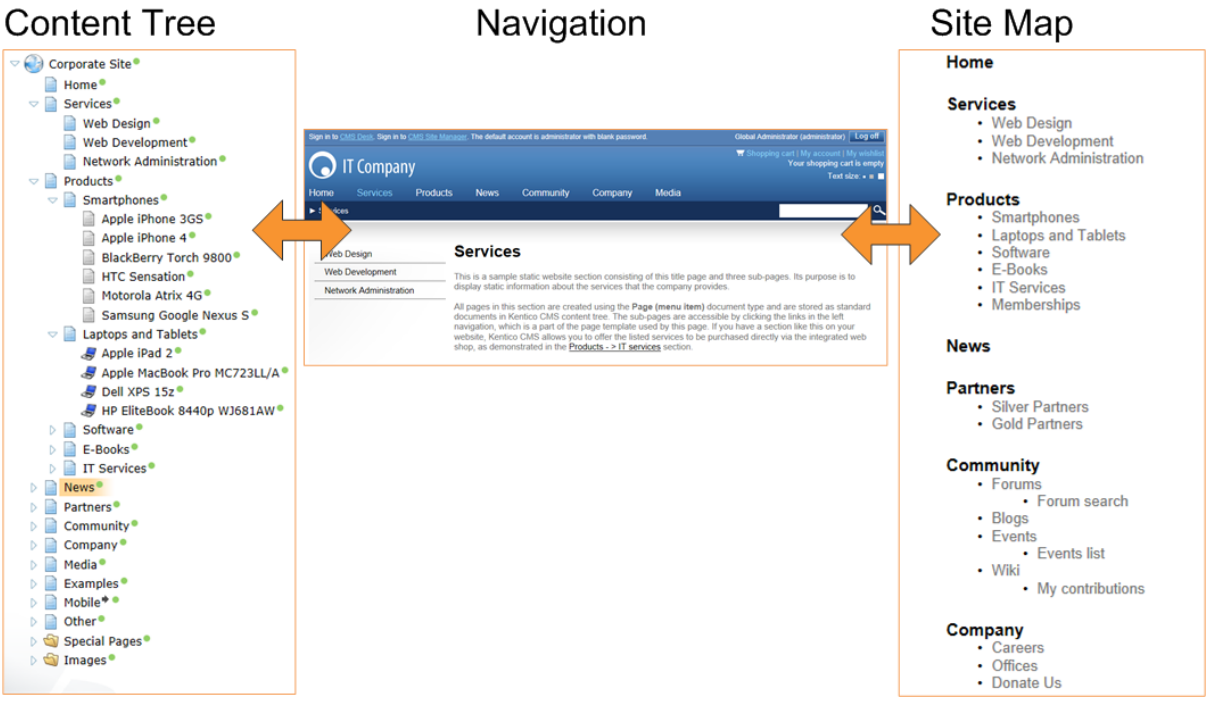


The tree hierarchy provides many advantages:

- 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 that consists of the document path in the tree hierarchy,
  such as `/products/lcd-displays/nec-52vm.aspx`.
- The structure allows you to define permissions for a particular site section and inherit them to

underlying items.

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 (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 basically documents. However, there's a special type of documents called **pages**. The pages (such as /Home, /Products/PDAs) display the content and they are displayed as menu items by default (this can be also customized).

Unlike pages, structured documents (such as news item /News/Your first news in 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 modifies 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?

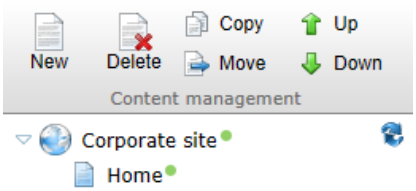
4.3 Editing content

4.3.1 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.

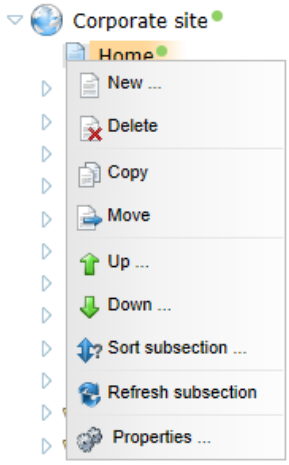
The ribbon offers the following actions:

- **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)



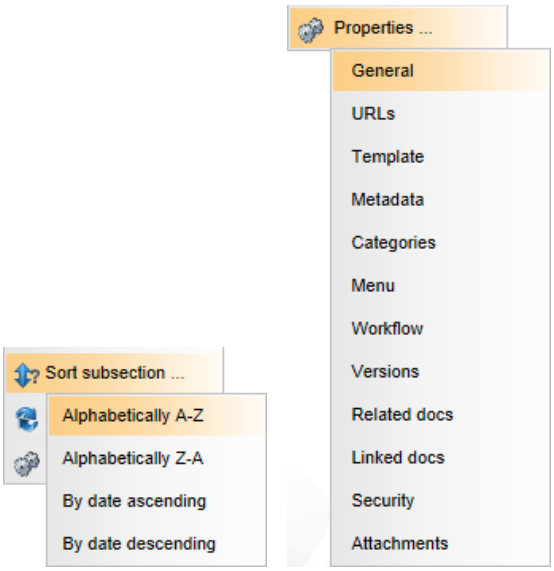
If you right-click a document in the content tree, a **context menu** appears. The menu offers the same actions as listed above, while the following extra options are available on top of the ones from the ribbon:

- **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:

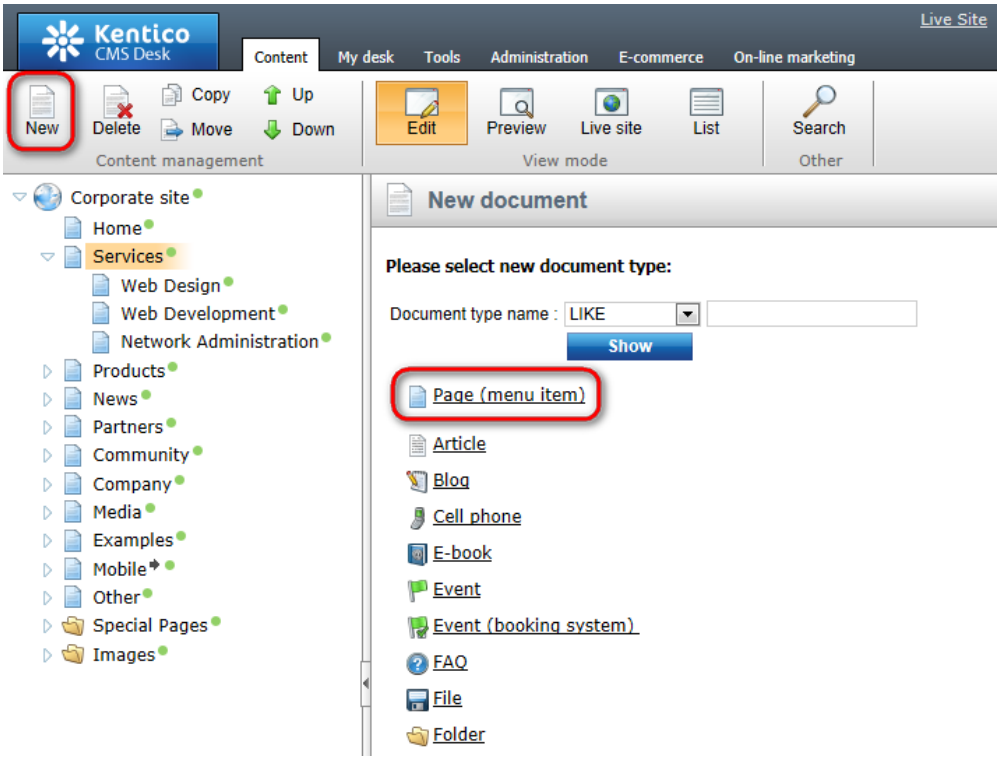
- **New** - the sub-menu offers you the document type of the new document, while only allowed child document types are offered
- **Up** - the sub-menu offers only the **Top** option, which moves the document to the first position in the current tree level
- **Down** - the sub-menu offers only the **Bottom** 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



4.3.2 Creating a new page

When you're creating 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.

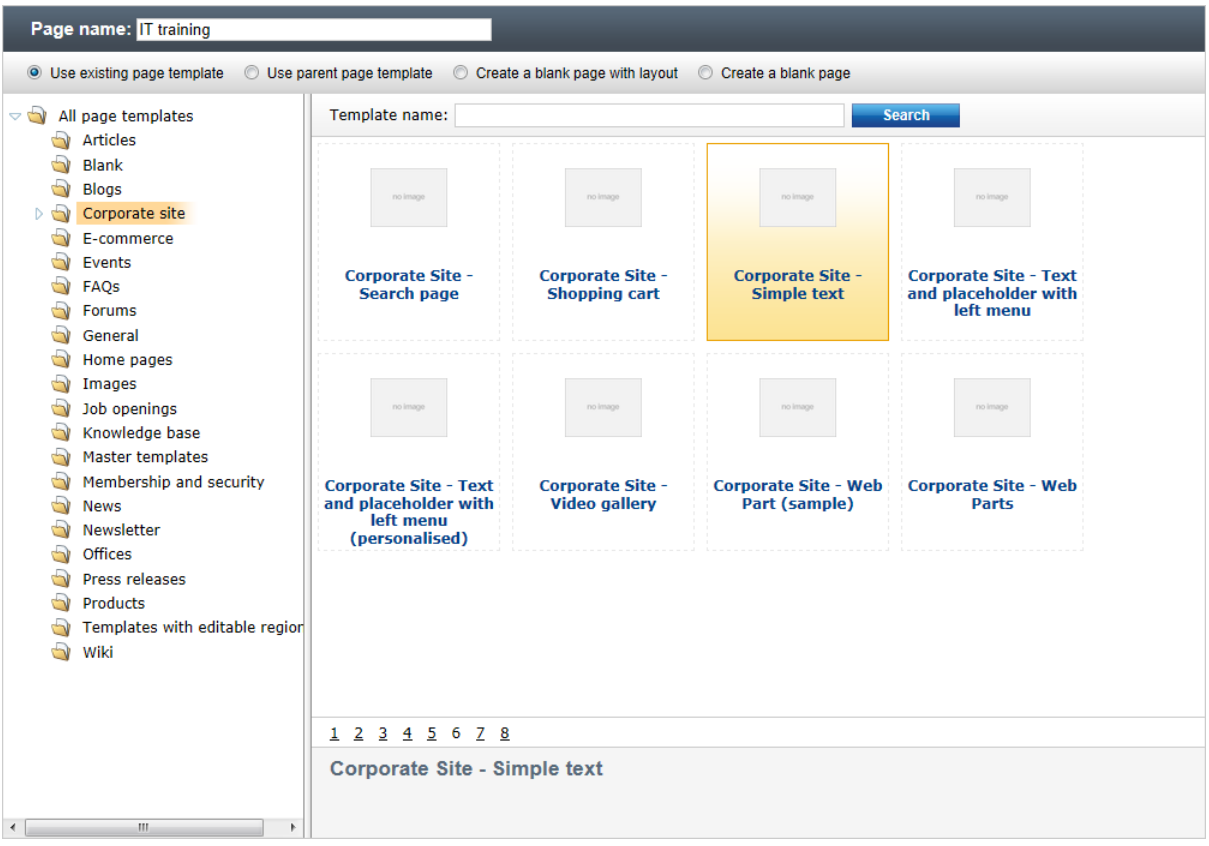


2. Enter the new page's name into the **Page name** field and select the page template that will be used by the page.

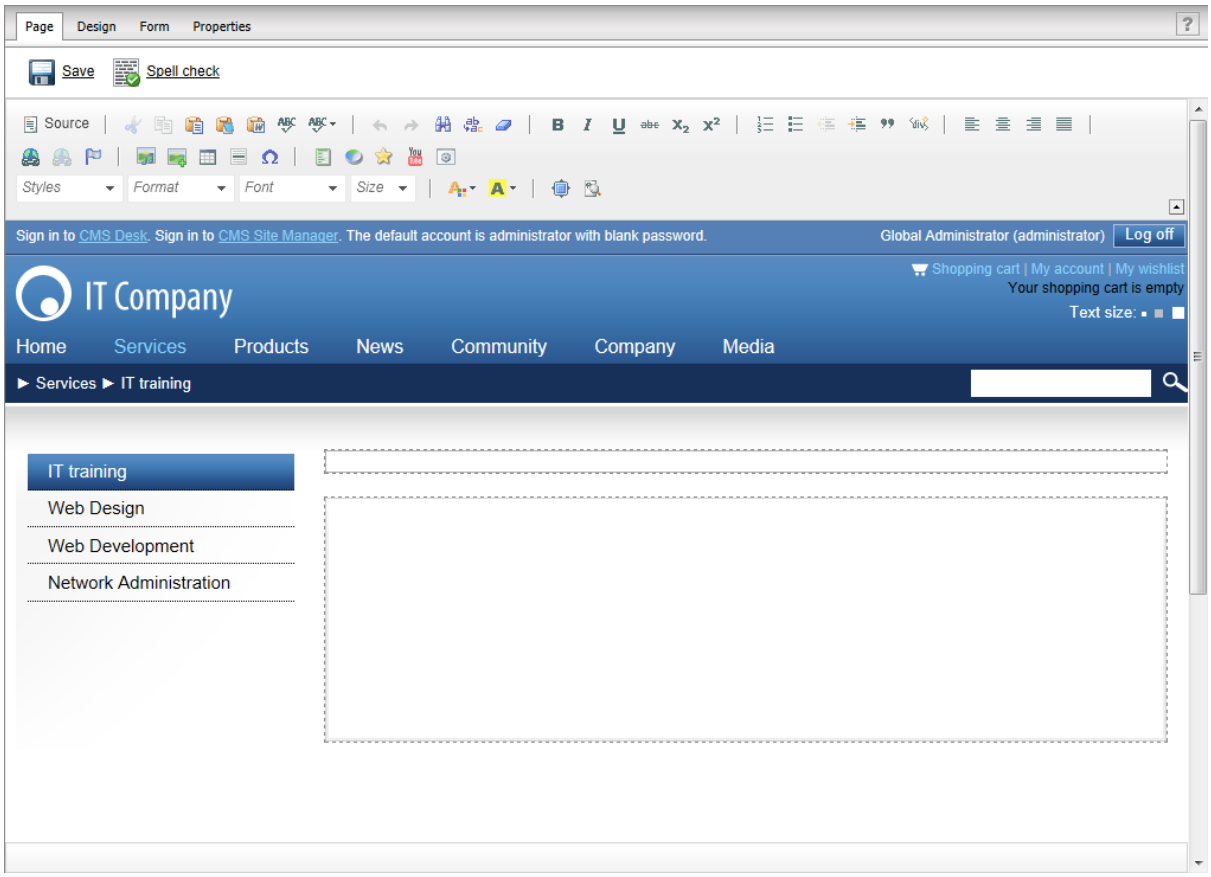
The page template defines the layout and design of the new page. When creating a new page, you can choose from the following options using the check-boxes below the **Page name** field:

- **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**.



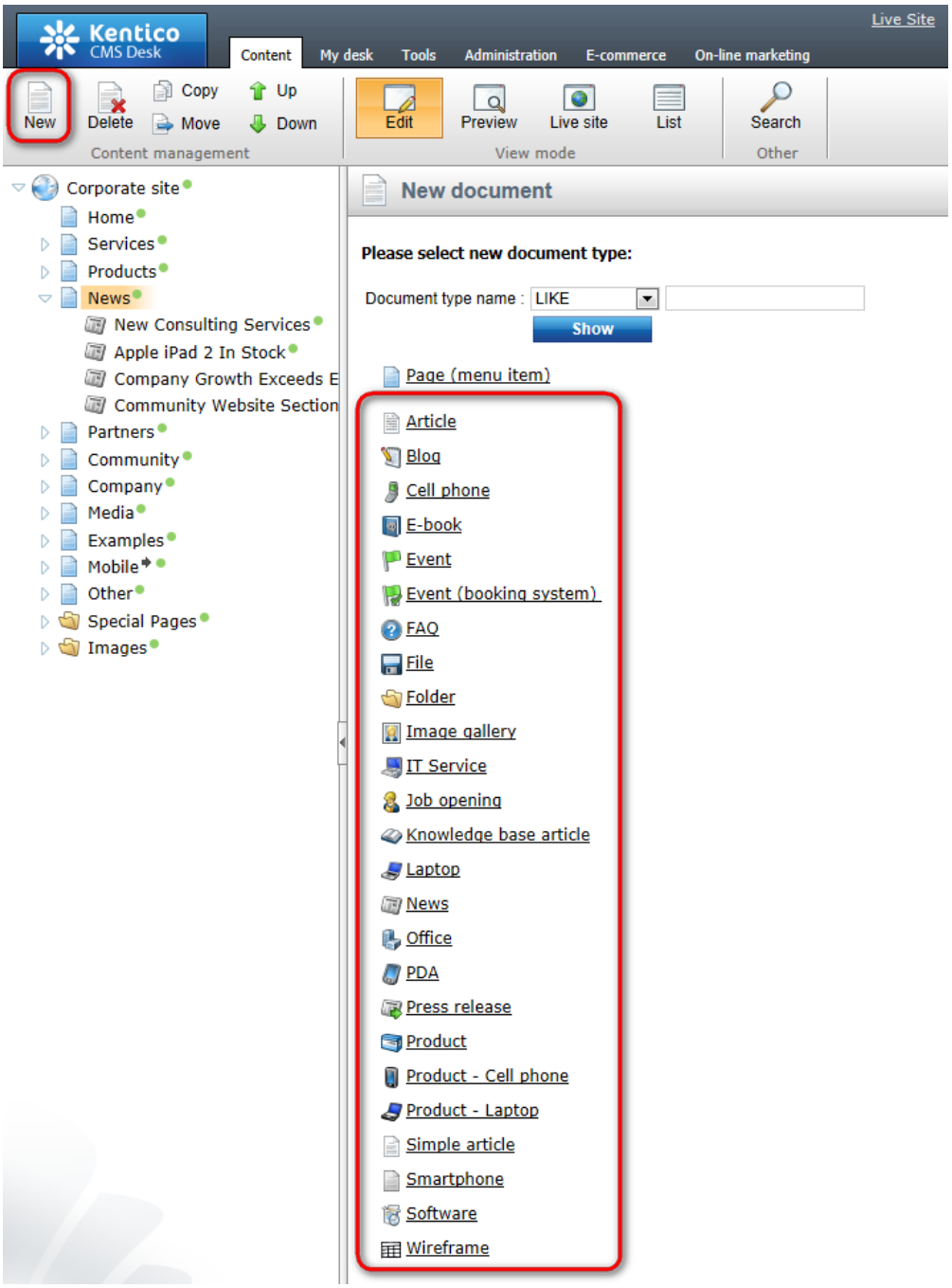
3. You're redirected to the **Page** tab and you can enter some content. Click **Save** to save your content changes.



4.3.3 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.



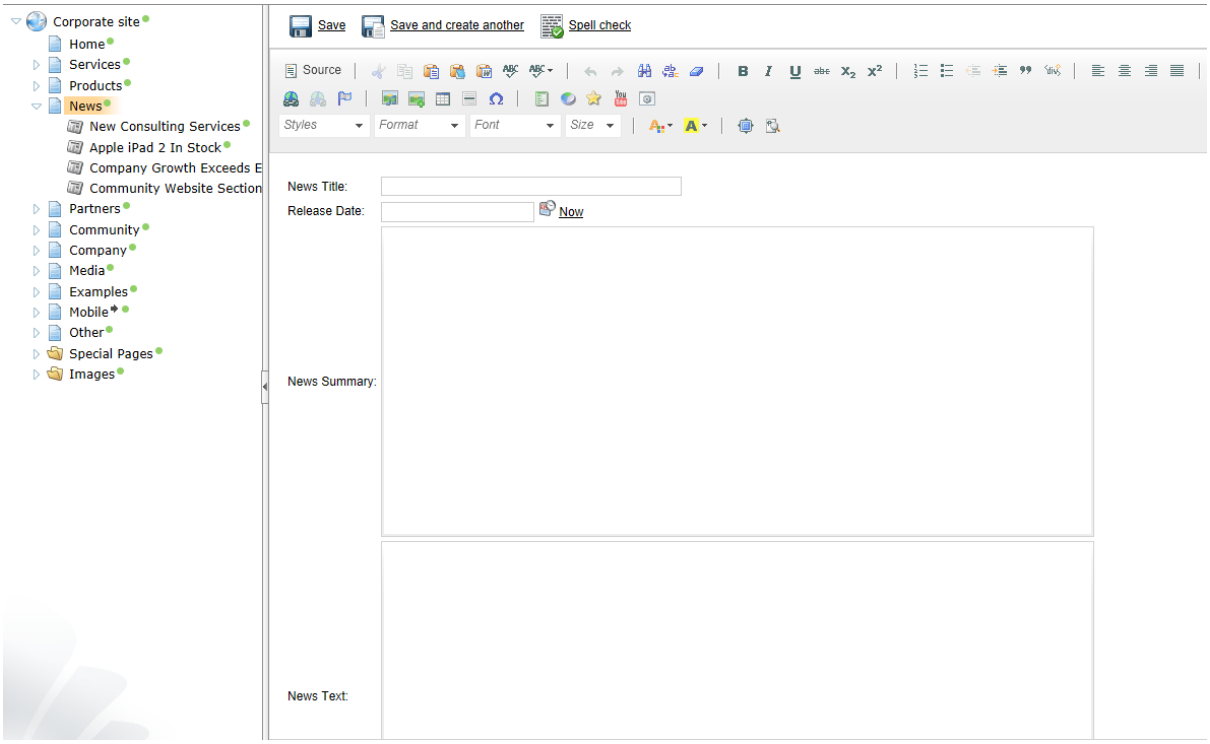
Available document types

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:** 8/15/2007 (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 create another news document in the same location. The later option is useful if you need to create several documents of the same type in the same location at once.

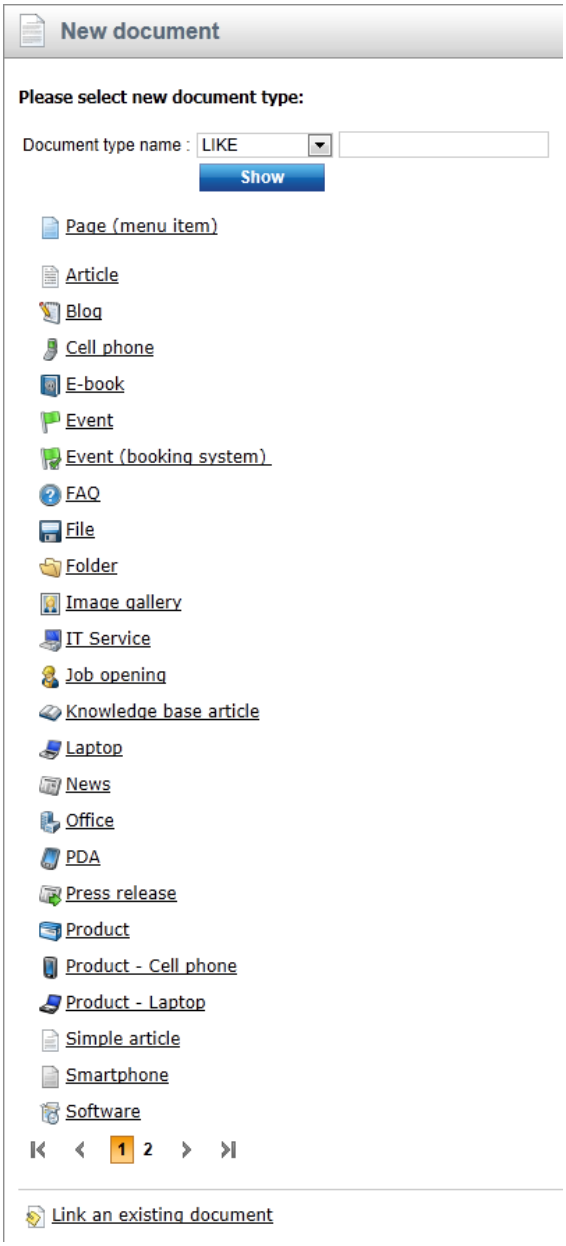


4.3.4 Creating a linked document

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.

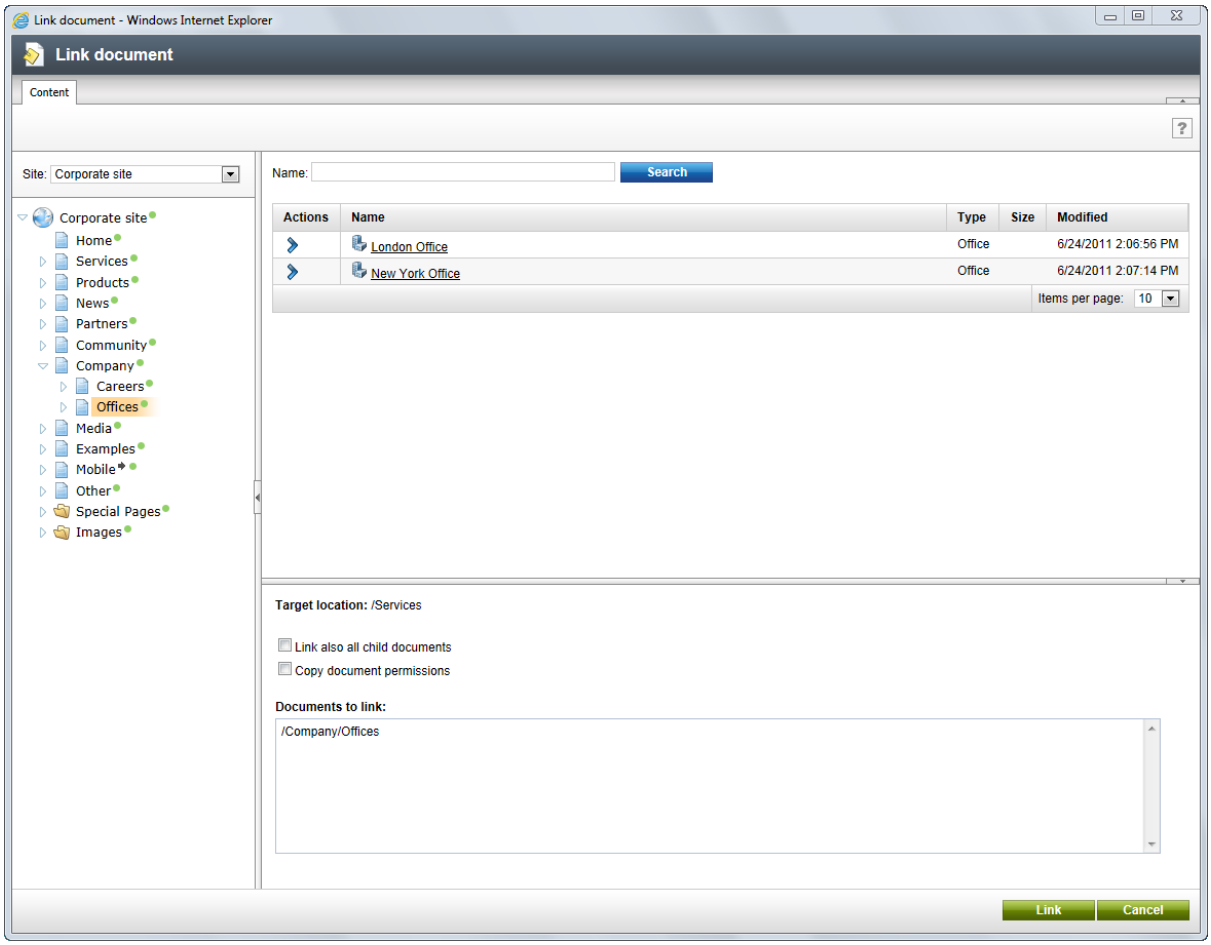
In order to create a linked document, select the document under which you want the linked one to be placed, click **New** and choose **Link an existing document** in the bottom of the screen:



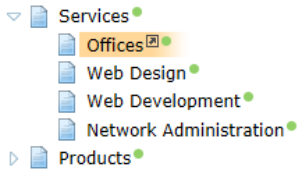
Then, choose the document that should be linked. You have two additional options to configure:

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



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

4.3.5 Drag-and-drop operations with documents

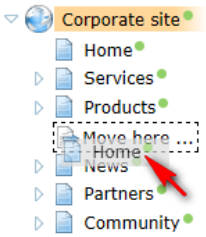
You can perform the following operations with documents quickly by dragging and dropping in the content tree.

- **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

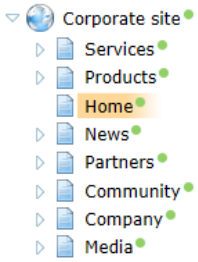
The following steps describe the process of dragging and dropping a document:

1. Select the document that you want to move.

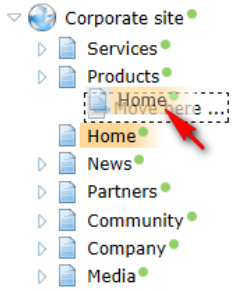
2. Move the mouse to the target location while still holding down the mouse button.



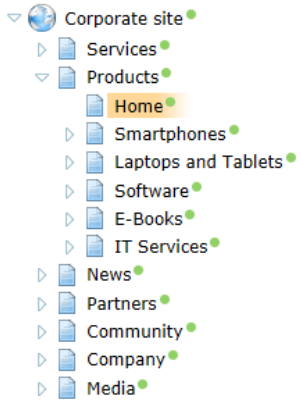
3. Release the mouse button. The document will be moved to the location where you dropped it.



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

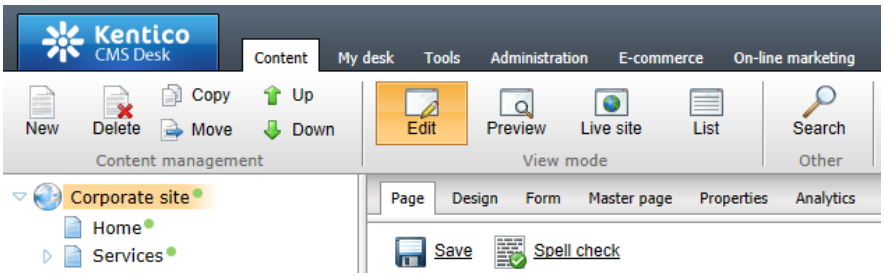


5. ... it will be placed under the document above it.



4.3.6 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's currently displayed on the live site to website visitors.
- **Preview** - this mode shows you the page version as it will be published on the site before it's actually published.
- **List** - this mode shows the list of all documents under the currently selected document. It's useful if you have many documents under a single parent document and need to browse them effectively.

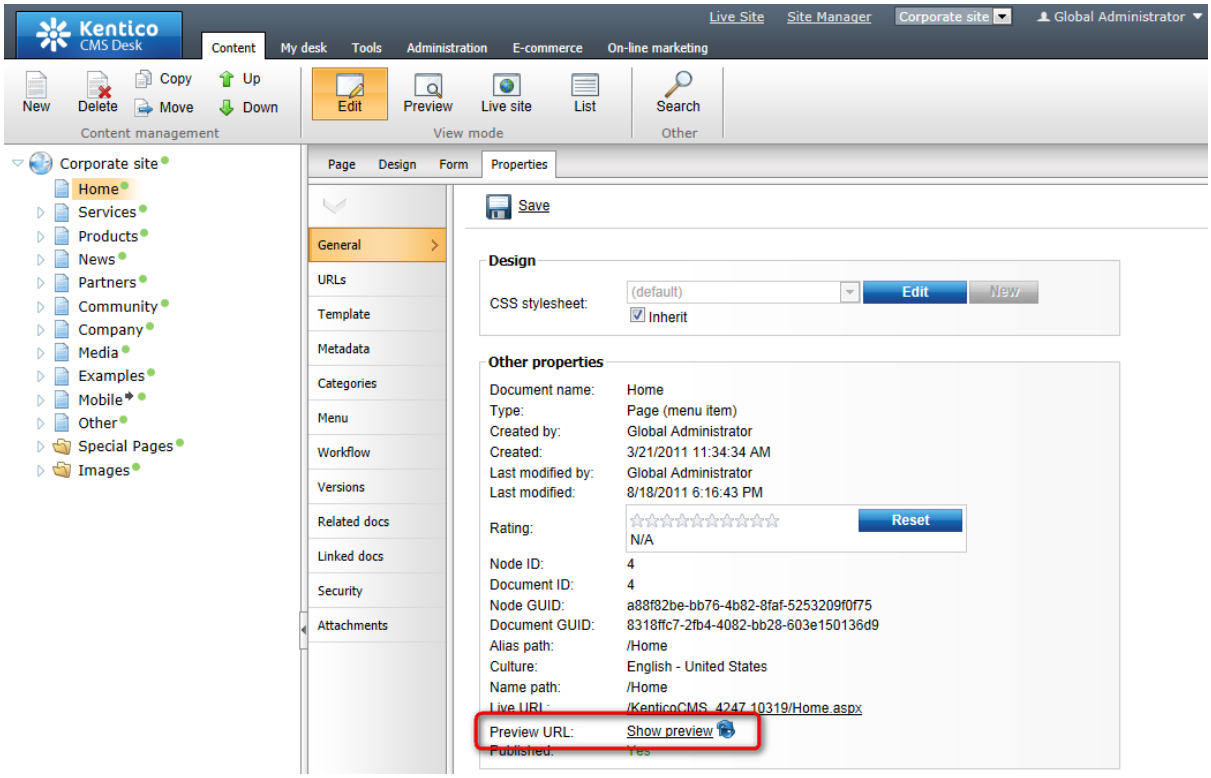
The following points summarize the difference between the **Live site** and **Preview** view modes:

- 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.

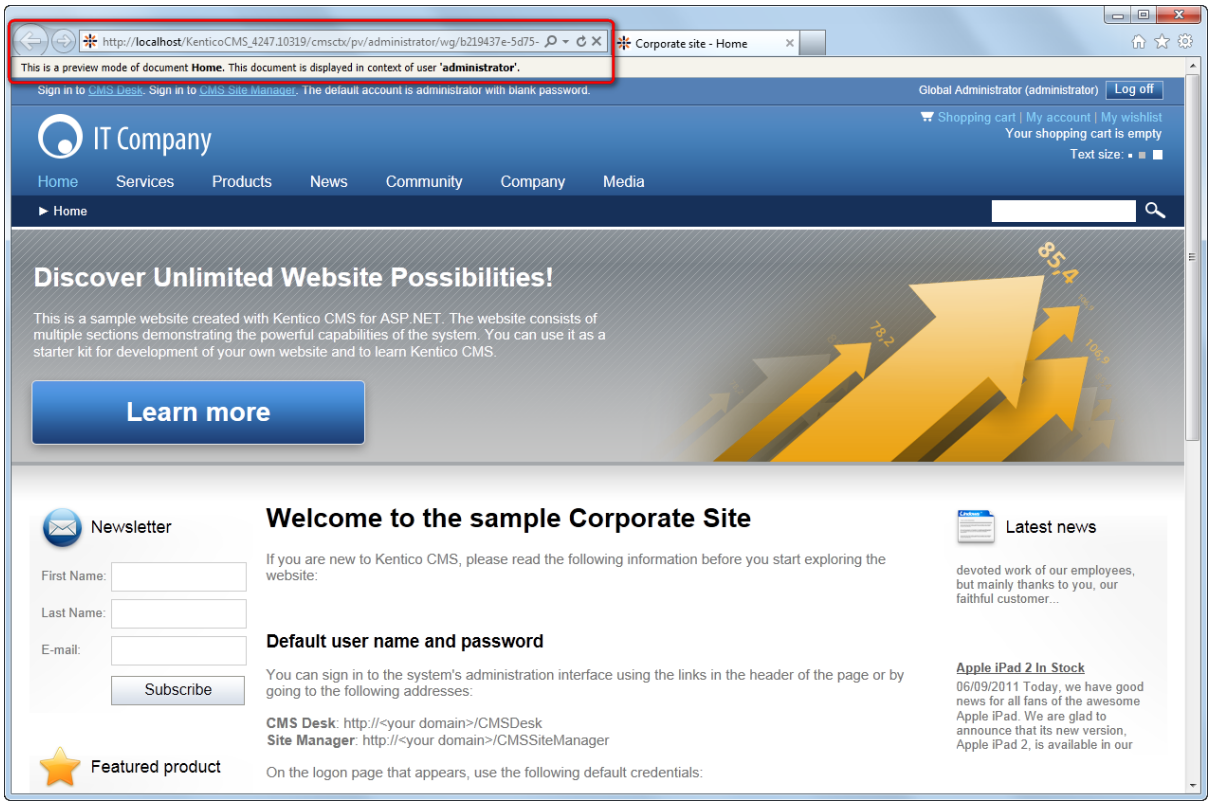
- The **Preview** mode does not use caching, so it may display published changes that are not visible in **Live site** mode yet due to caching.
- When a document does not use workflow and its **Publish from** property is set to a future date and time, the **Live site** mode does not display any content, while the **Preview** mode displays the content that will be published after the specified date and time.
- When the document uses workflow and didn't get to the **Published** step yet, the **Live site** mode doesn't display any content, while the **Preview** mode displays the content created in the current workflow step.
- When the document uses workflow, it already got to 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

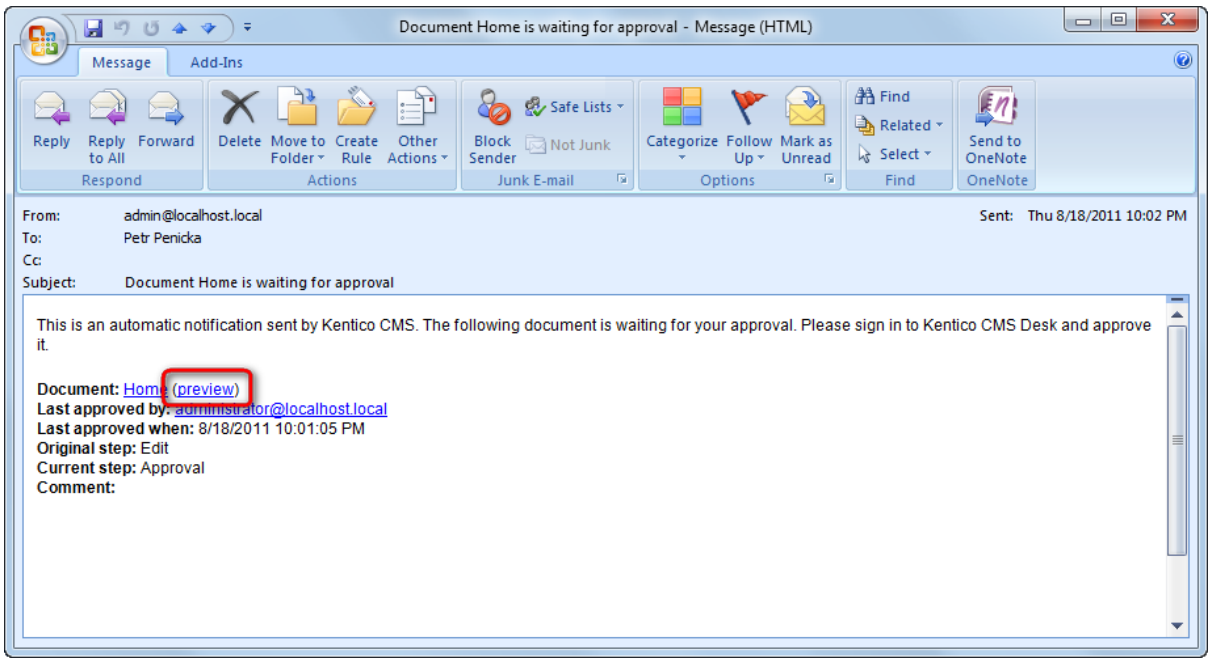
On the **Properties -> General** tab of an edited document, you can find the **Preview URL** property. If you click the **Show preview** link next to it, a new window will be opened, displaying the current document in **Preview** mode outside Kentico CMS user interface.



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.

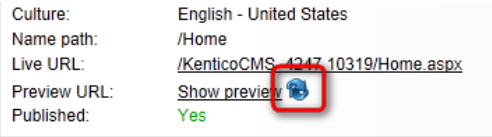


Preview links are also included in workflow notification e-mail templates. In this case, the preview is displayed in context of the user who performed the workflow action about which the e-mail is notifying.



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 the **Generate preview link** ( ) button 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:

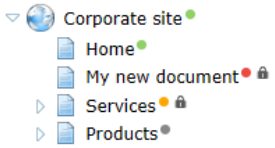
1. You need to install the URL Rewrite Module into the IIS. It can be downloaded from the following location: http://www.iis.net/download/urlrewrite.

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

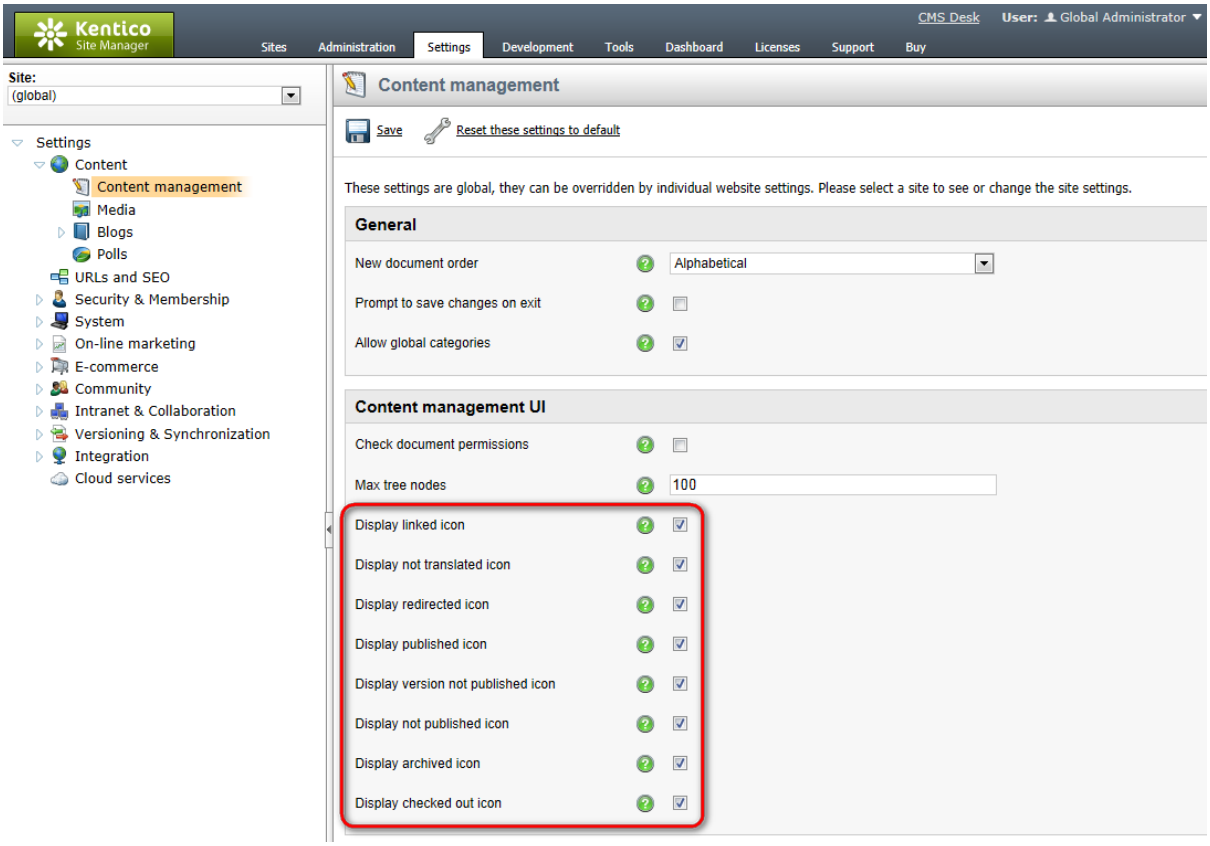
```
<system.webServer>
  <rewrite>
    <rules>
      <rule name="Remove virtual context prefix" stopProcessing="false">
        <match url="^cmsctx/(.+)/-/(.+)$" />
        <action type="Rewrite" url="cmsctx/-/{R:2}" />
      </rule>
    </rules>
  </rewrite>
  ...
</system.webServer>
```

4.3.7 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 settings in **Site Manager -> Settings -> Content -> Content management** need 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 translated in the currently edited culture, i.e. documents that are only present in the default culture, but have no version in the currently edited culture.	Display not translated icon
	This icon appears next to documents that have a redirection configured in	Display redirected

	Properties -> Menu -> URL redirection.	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 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

4.4 WYSIWYG editor

4.4.1 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. However, it is possible to integrate your custom WYSIWYG editor instead of the built-in one.

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.

4.4.2 Insert image or media

4.4.2.1 Overview

The **Insert image or media** dialog can be used to insert the following types of files:

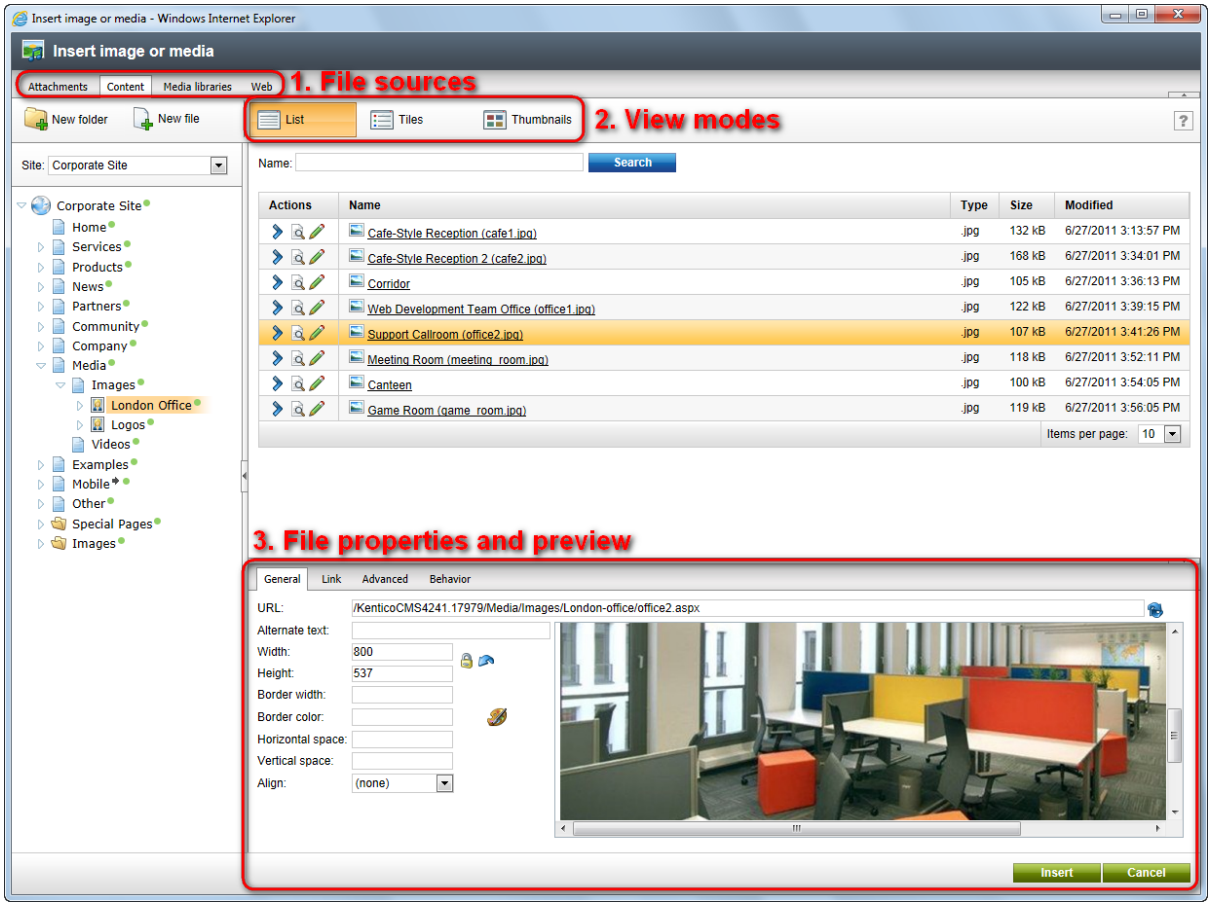
- **Images** - bmp, gif, ico, png, wmf, jpg, jpeg, tiff, tif.

- **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.

4.4.2.2 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

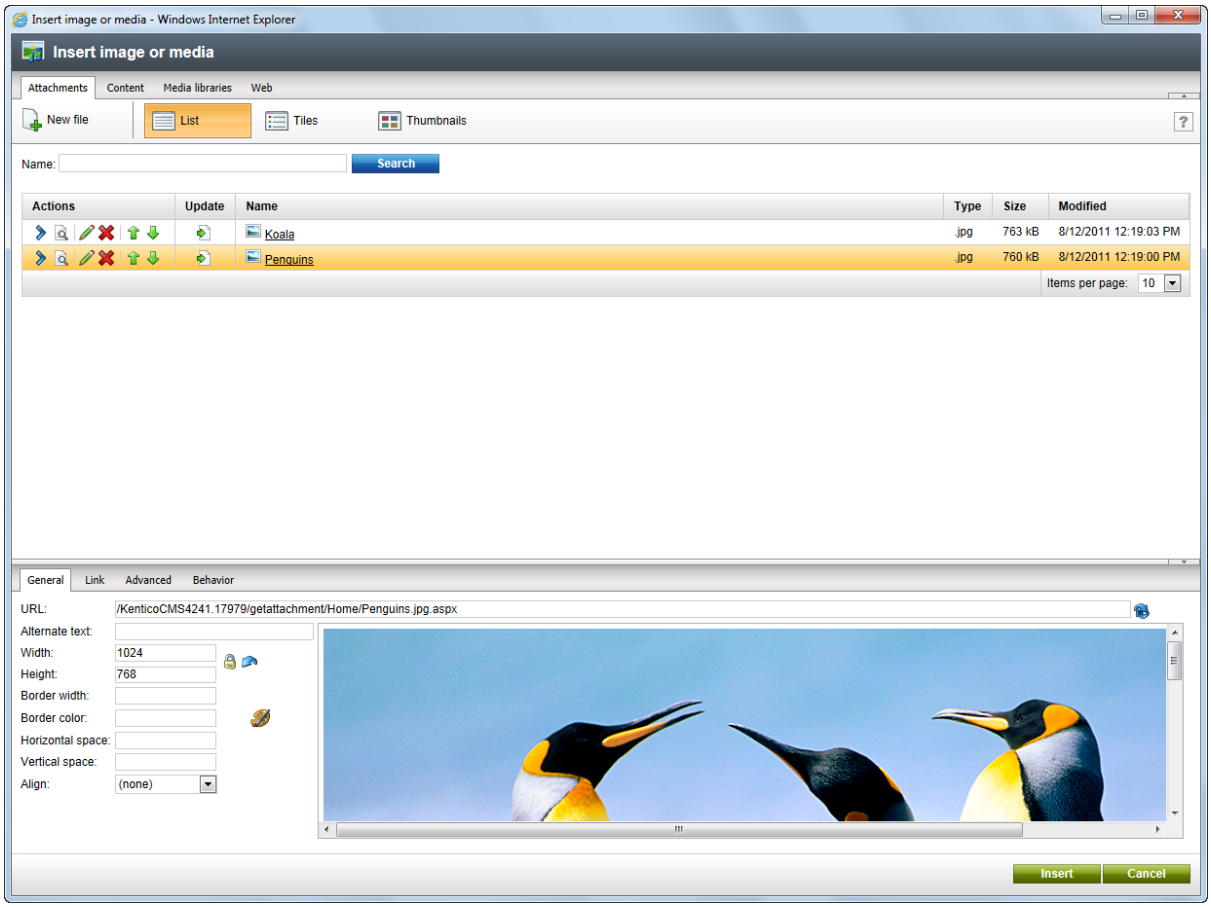
This tab is available only if the WYSIWYG editor is placed on a certain document (e.g. **Form** tab, user contributions etc.) so that attachments of the current document can be offered. In other parts of the CMS, the tab is not displayed.

From this tab, you can insert unsorted attachments of the document into the text. For more information about document attachments, please refer to the Document attachments chapter of this guide.

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 to the document via this dialog 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 re-order the 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 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

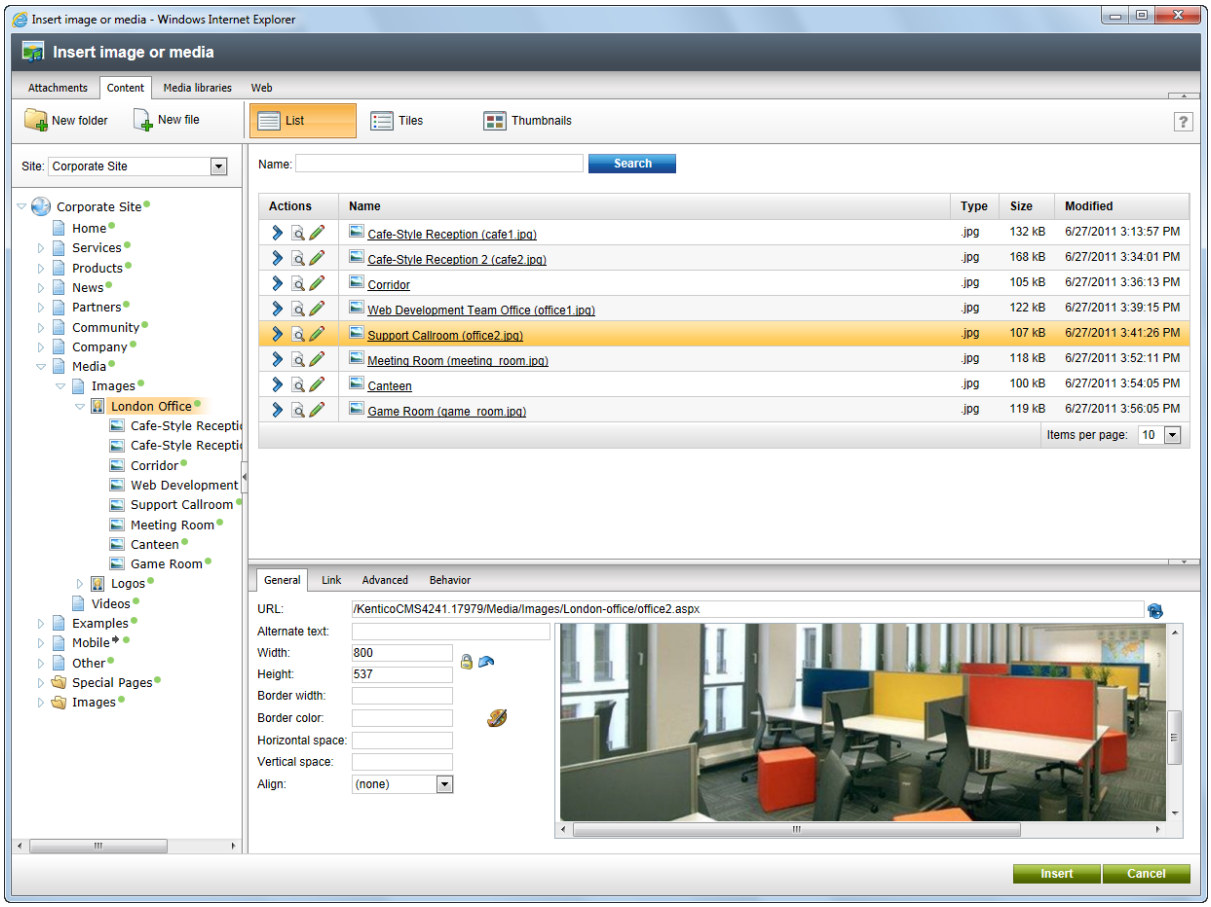
From this tab, you can insert files stored in the content tree of any of the sites running in the system. The site to insert from can be selected using the **Site** drop-down list, while its content tree will be displayed below. You can define which sites will be available. You can also define the starting alias path of the displayed content tree when defining a field in the field editor, as described in the Dialogs 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.

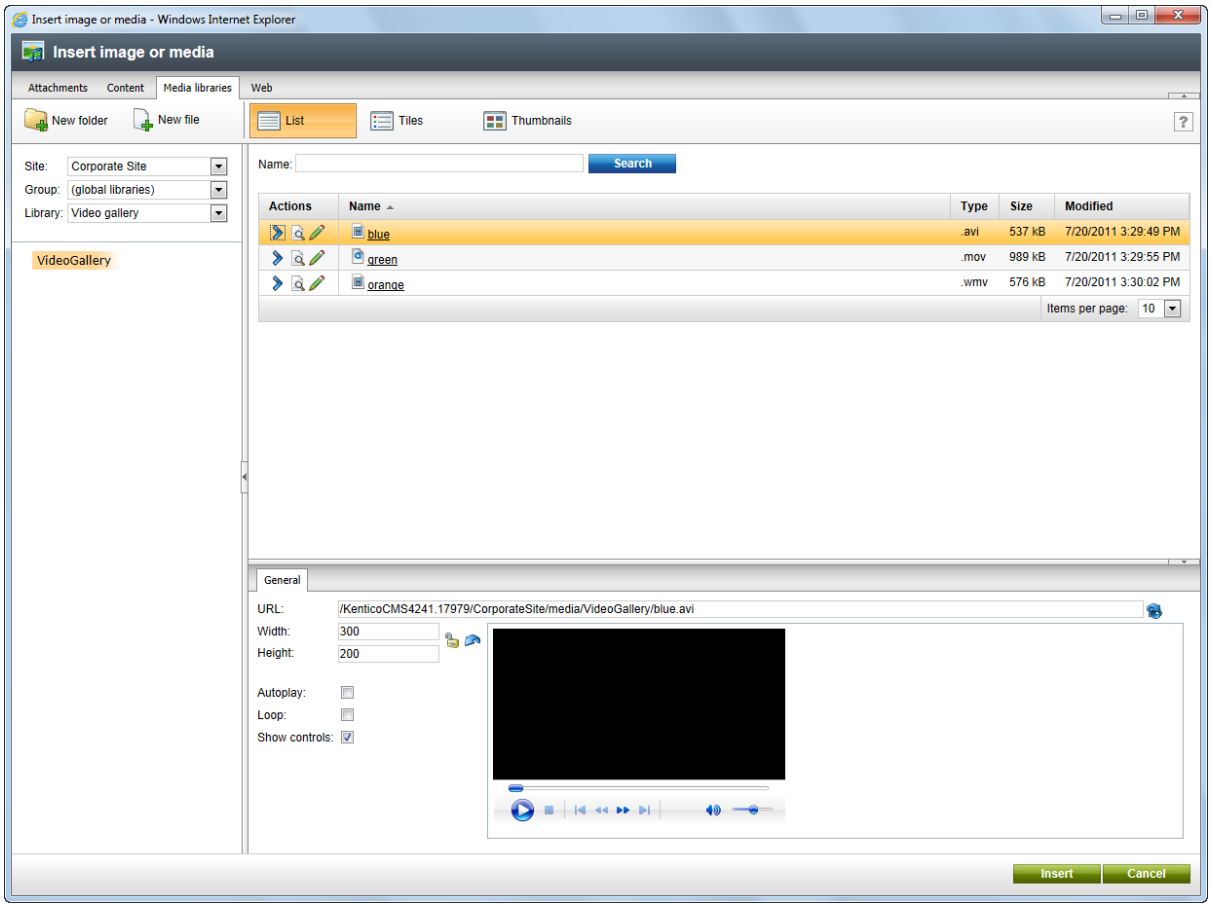
More information about Kentico CMS Media libraries can be found in the Modules -> Media libraries chapter.

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).

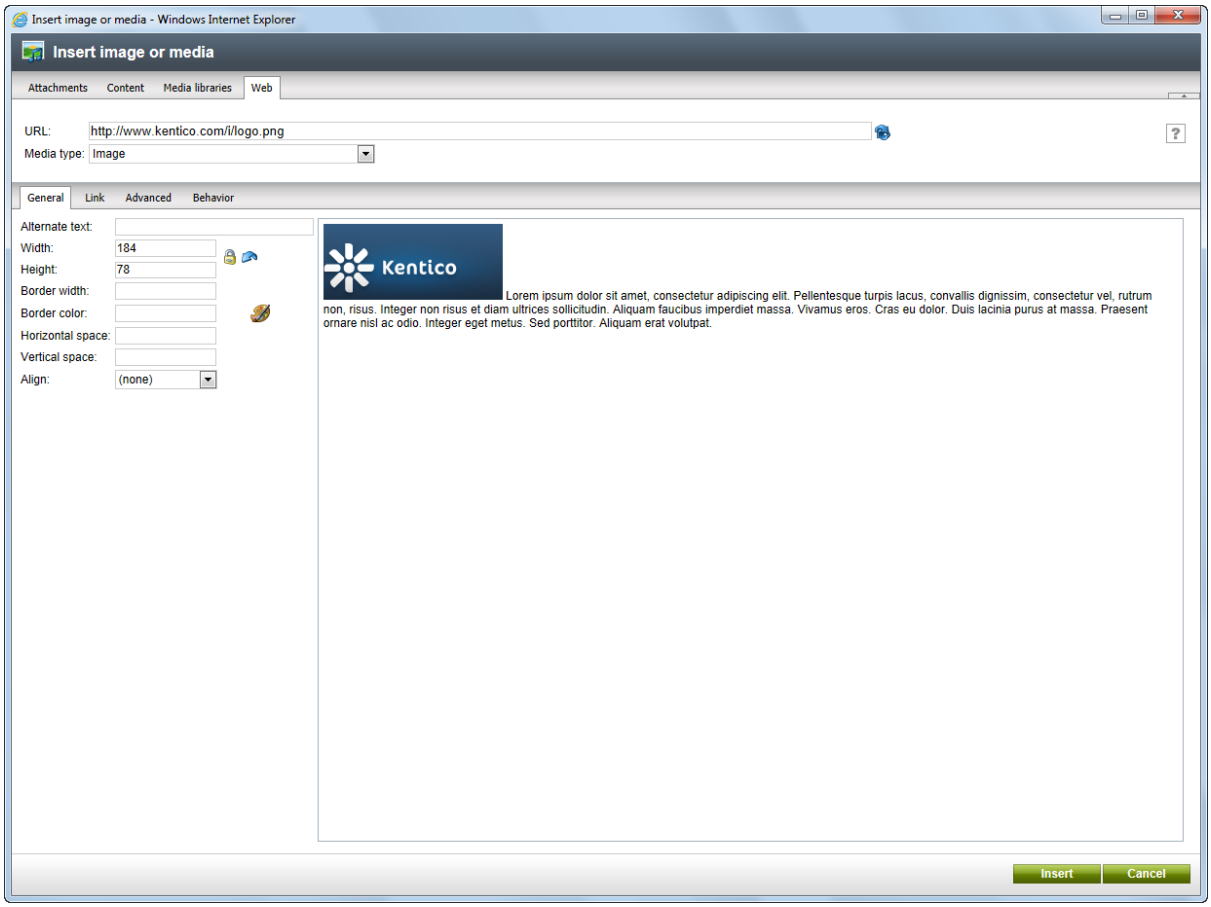
You can also perform the following actions with the listed files:

- 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

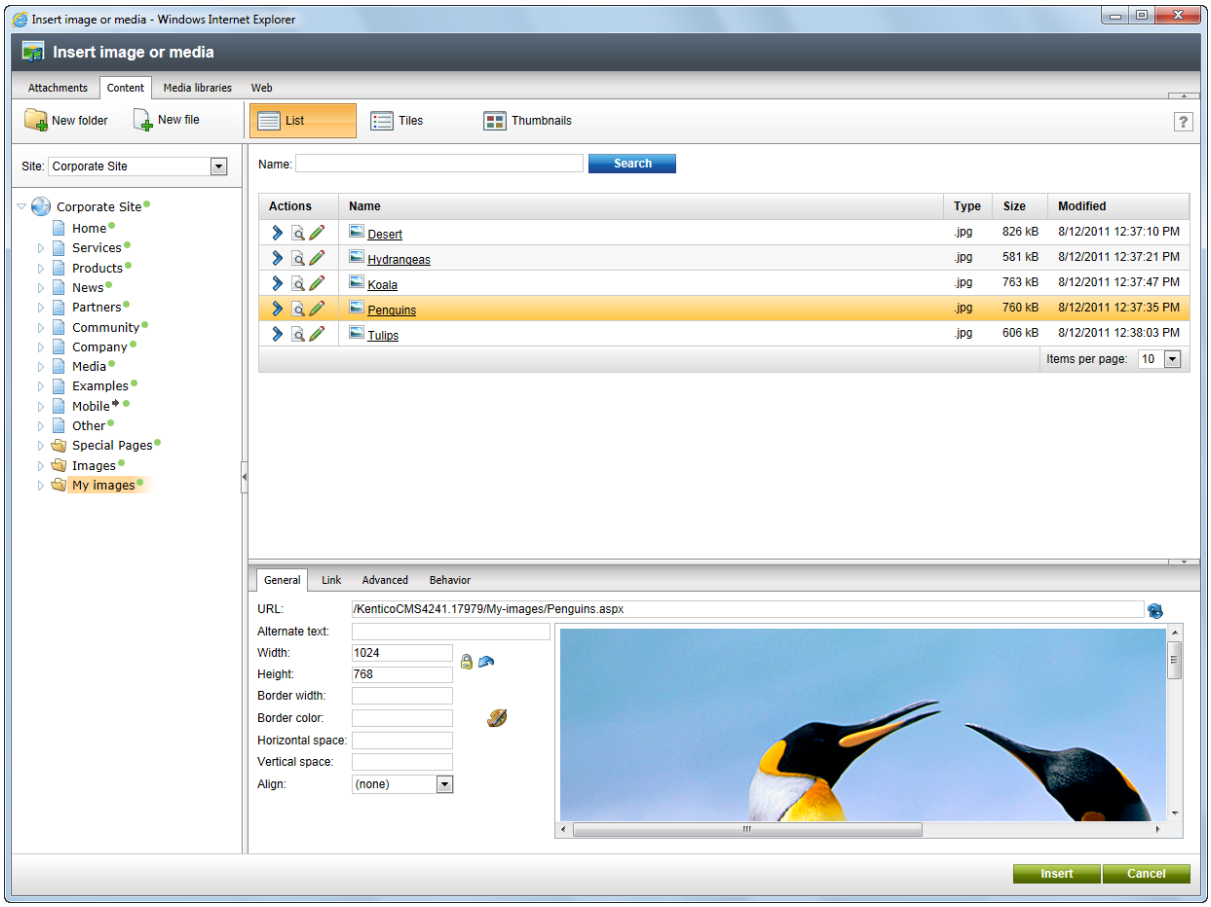
From this tab, you can enter an image, audio, video or flash from the web by entering its URL. More information on how to use this tab can be found in the Inserting images or media from the Web topic.



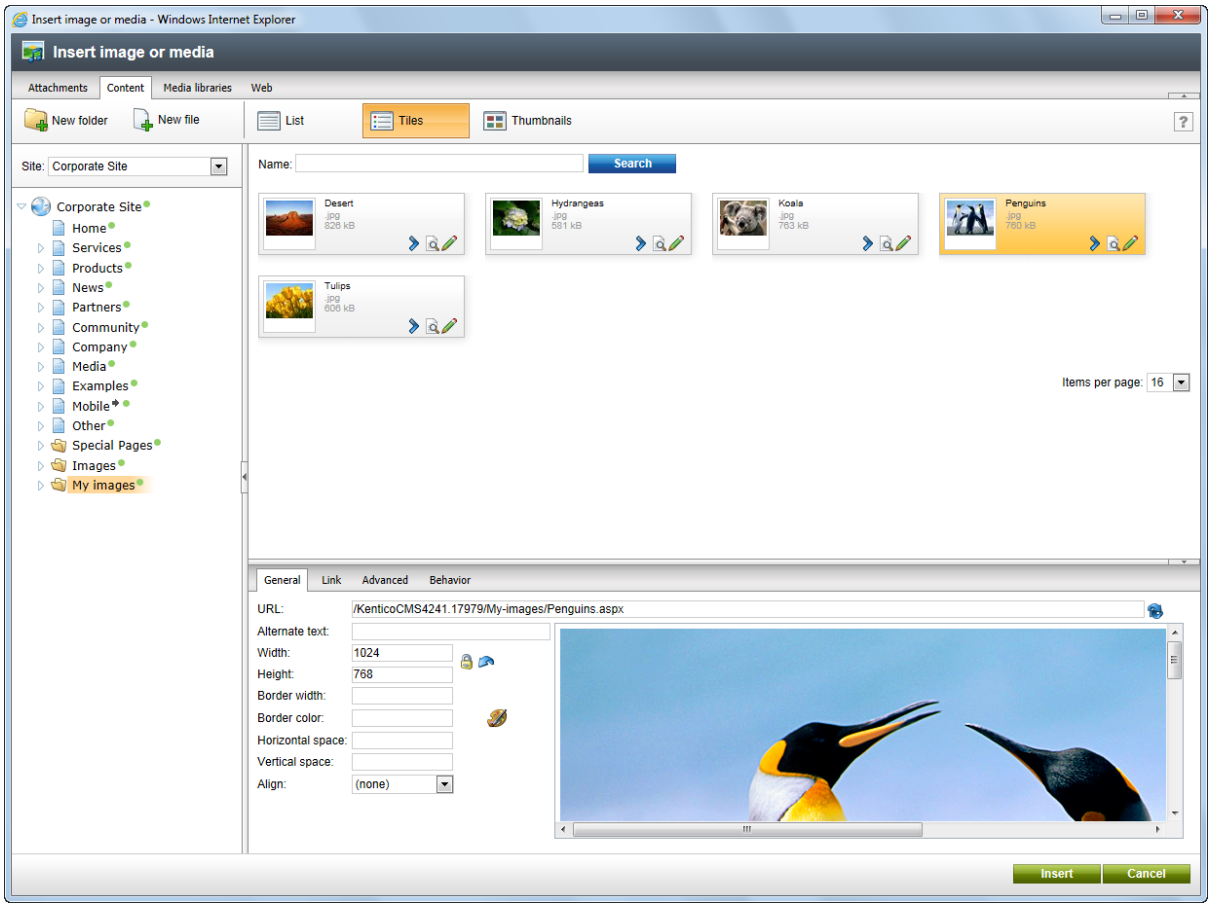
4.4.2.3 View modes

The following three view modes are available on the **Attachments**, **Content** and **Media libraries** tabs:

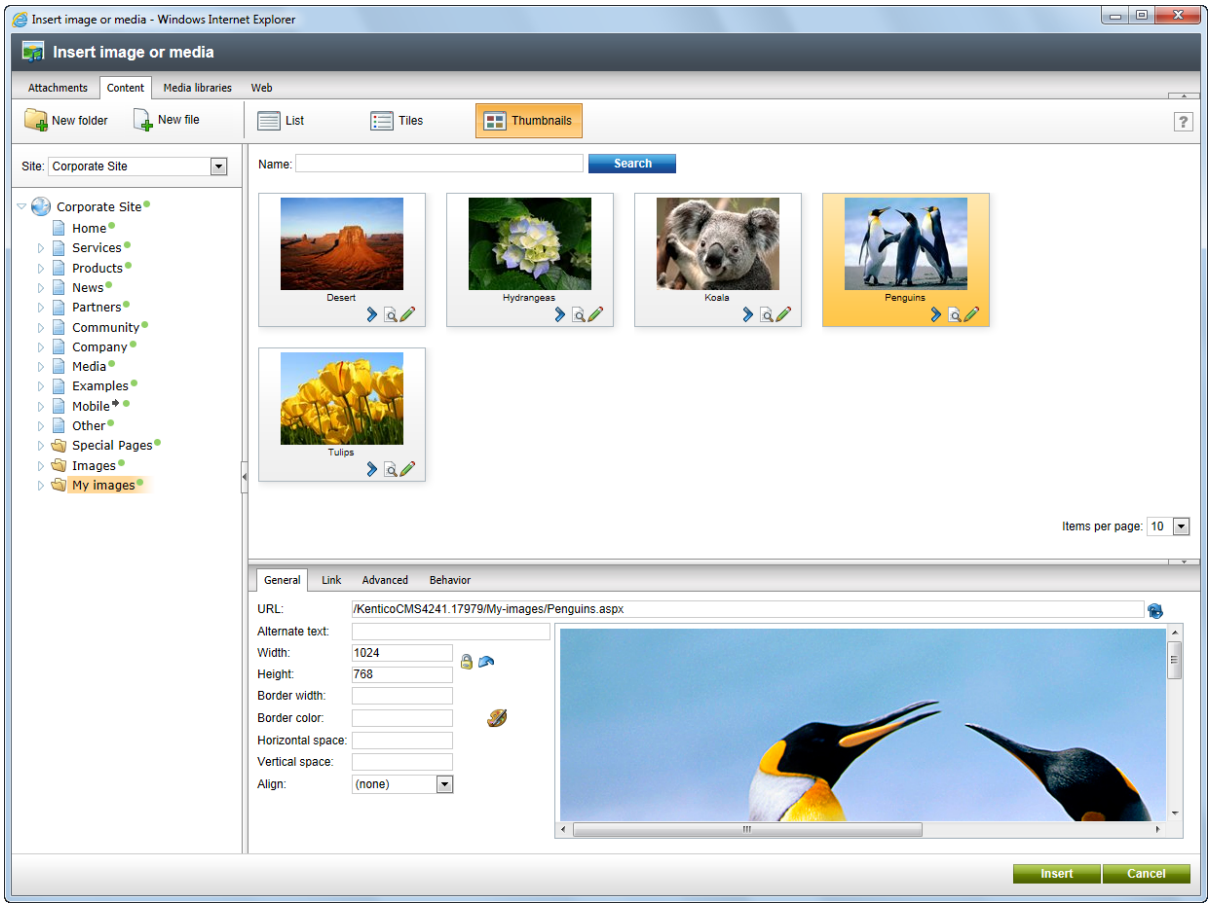
List


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
Actions
Name
Type
Size
Modified
Desert .jpg 826 kB 8/12/2011 12:37:10 PM
Hydrangeas .jpg 581 kB 8/12/2011 12:37:21 PM
Koala .jpg 763 kB 8/12/2011 12:37:47 PM
Penguins .jpg 760 kB 8/12/2011 12:37:35 PM
Tulips .jpg 606 kB 8/12/2011 12:38:03 PM
Items per page: 10
Corporate Site
Home
Services
Products
News
Partners
Community
Company
Media
Examples
Mobile
Other
Special Pages
Images
My images
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

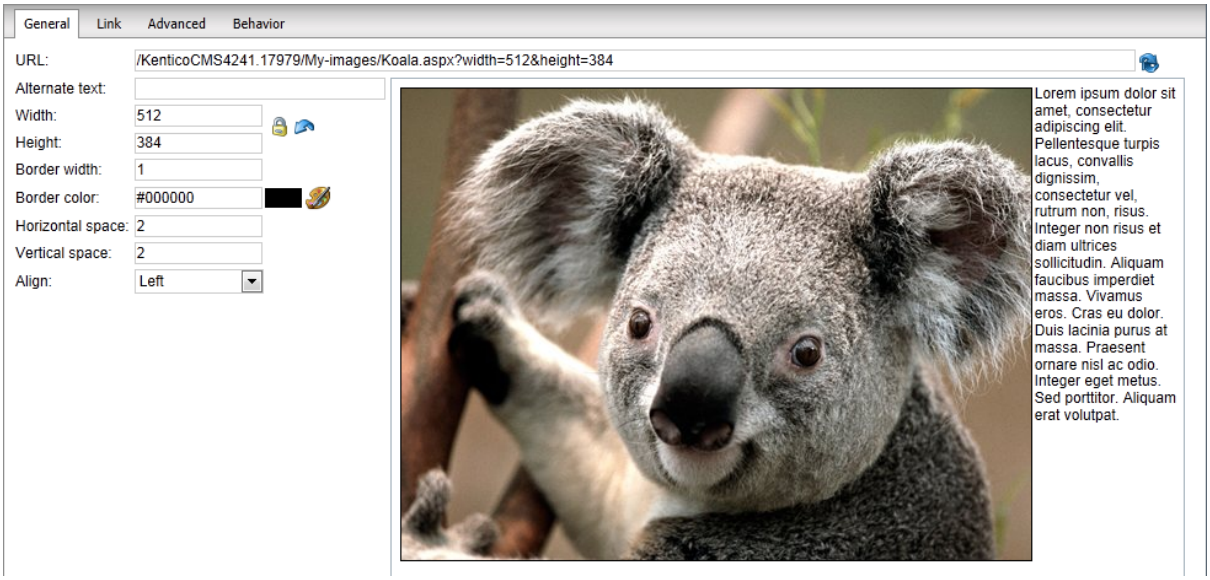
Tiles



Thumbnails



4.4.2.4 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 is not loaded correctly.
- **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.

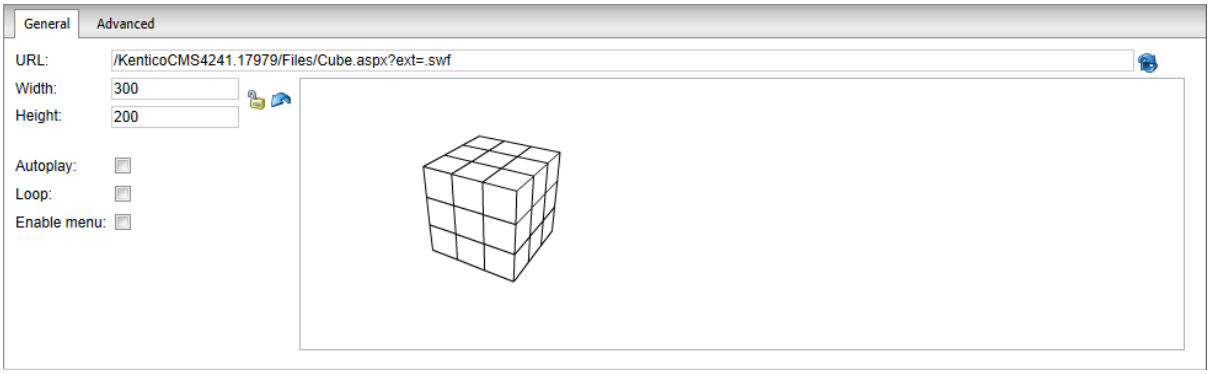
When the image is mouse-overed, it is displayed in a new layer in size defined on the **Behavior** tab as shown in the screenshot above. The image is clickable and when clicked, the user is redirected to the specified URL in the same browser window.

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>
```

4.4.2.5 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

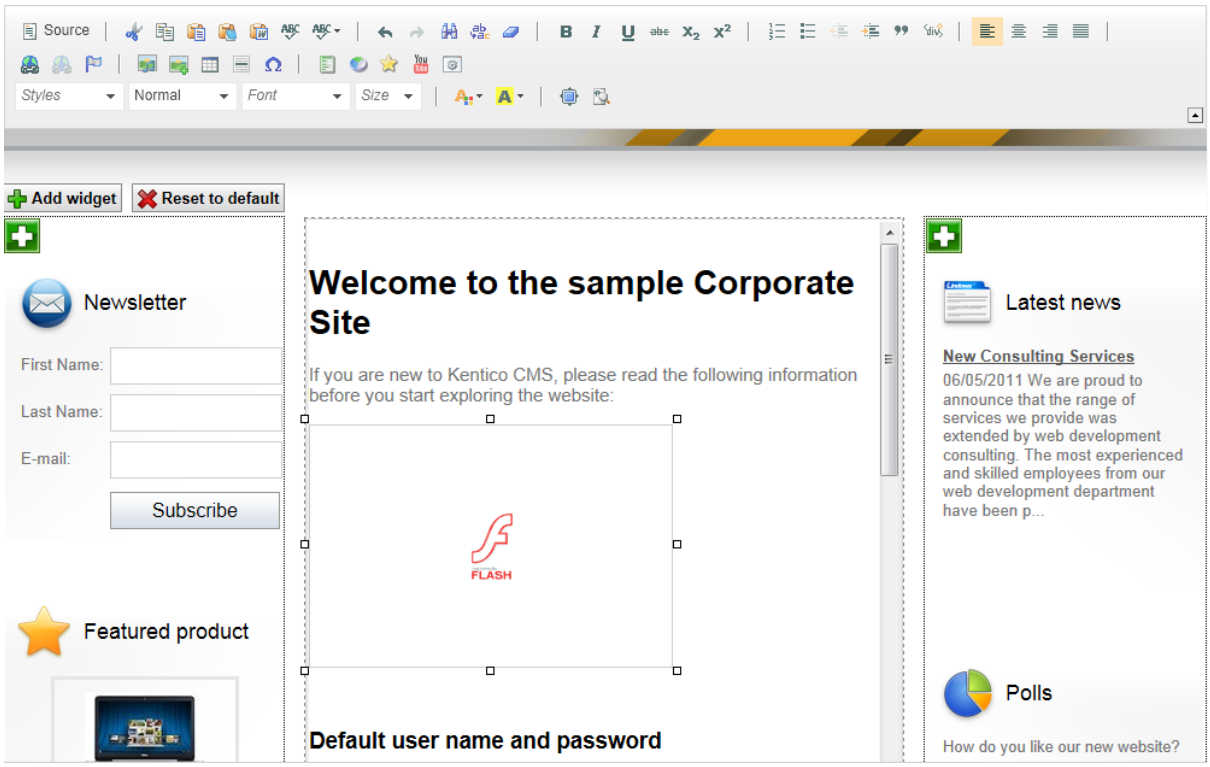
- **Scale** - defines how the flash player stretches, shrinks and resizes when the browser window is resized.
- **ID** - the identifier of the flash HTML object.
- **Advisory title** - the text displayed when mouse cursor is placed over the flash player.
- **Class** - the flash element CSS class.
- **Style** - 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:

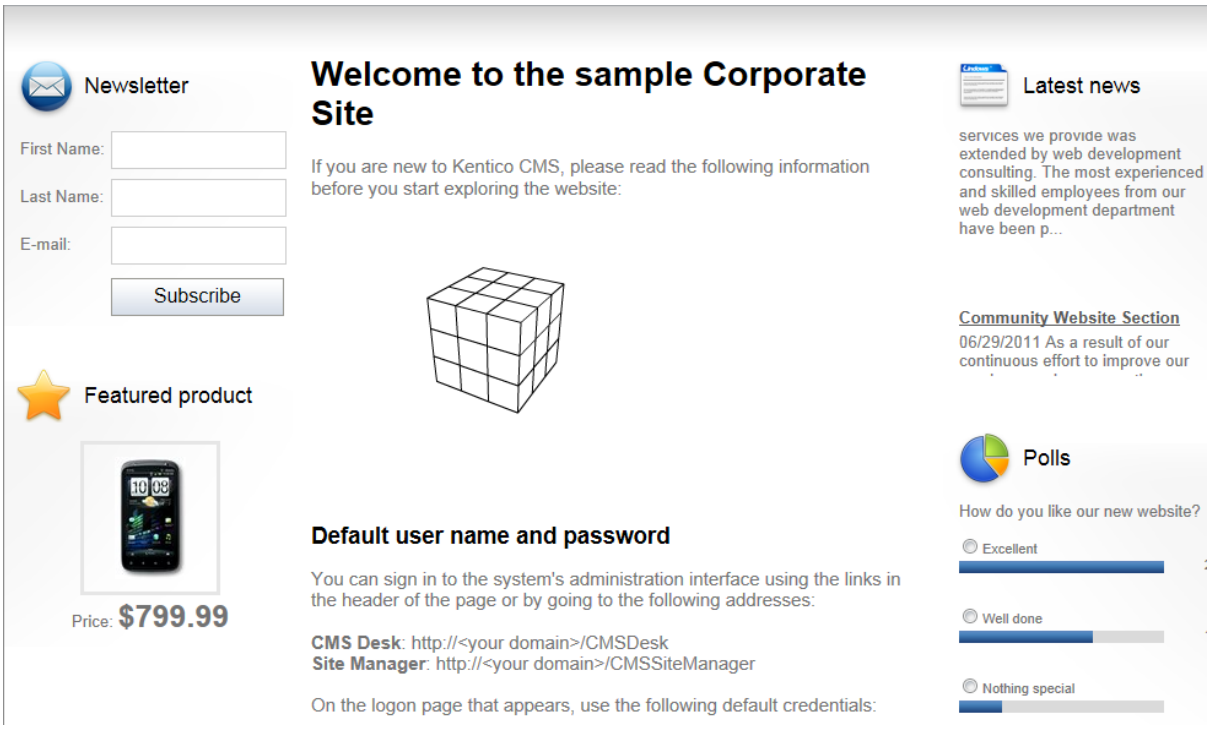
```
<object codetype="CMSInlineControl" height="200" type="Media" width="300">
```

```
<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>
```

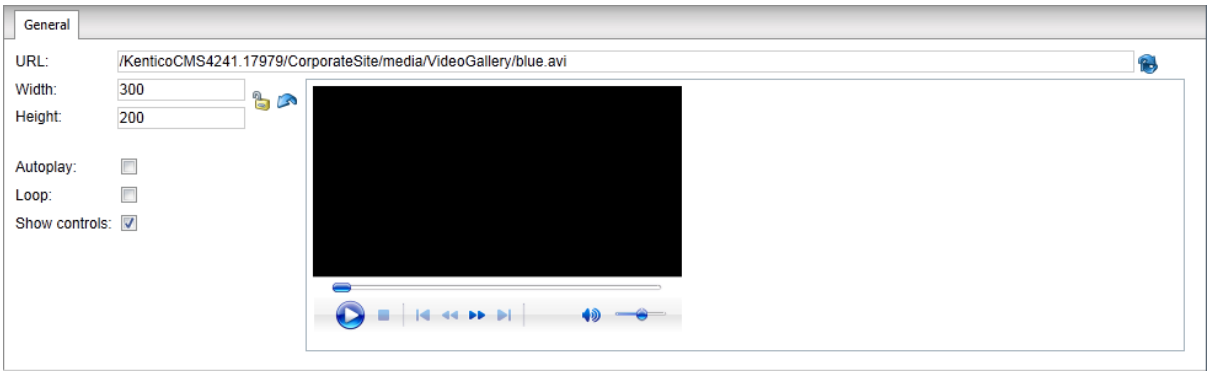
In the WYSIWYG editor, the flash is displayed only in the form of a box with the Flash logo, giving information about the size of the player:



And this is the result on the live site:



4.4.2.6 Inserting audio/video



When inserting both audio and video, the following properties can be set:

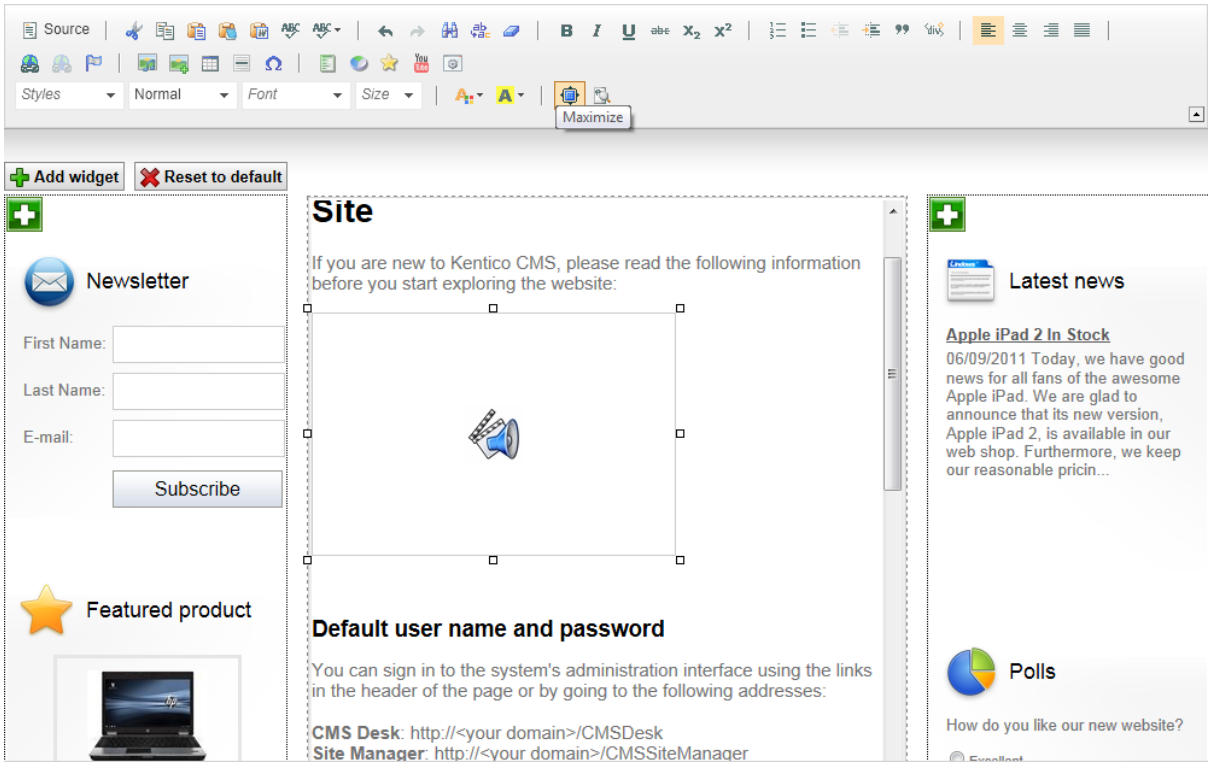
- **URL** - URL of the attached audio/video file.
- **Width**, **Height** - the width and height of the media player; 300x200px is used by default.
  - Aspect ratio can be locked (), which makes the second dimension recalculated automatically when you change one dimension, while the ratio between the 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

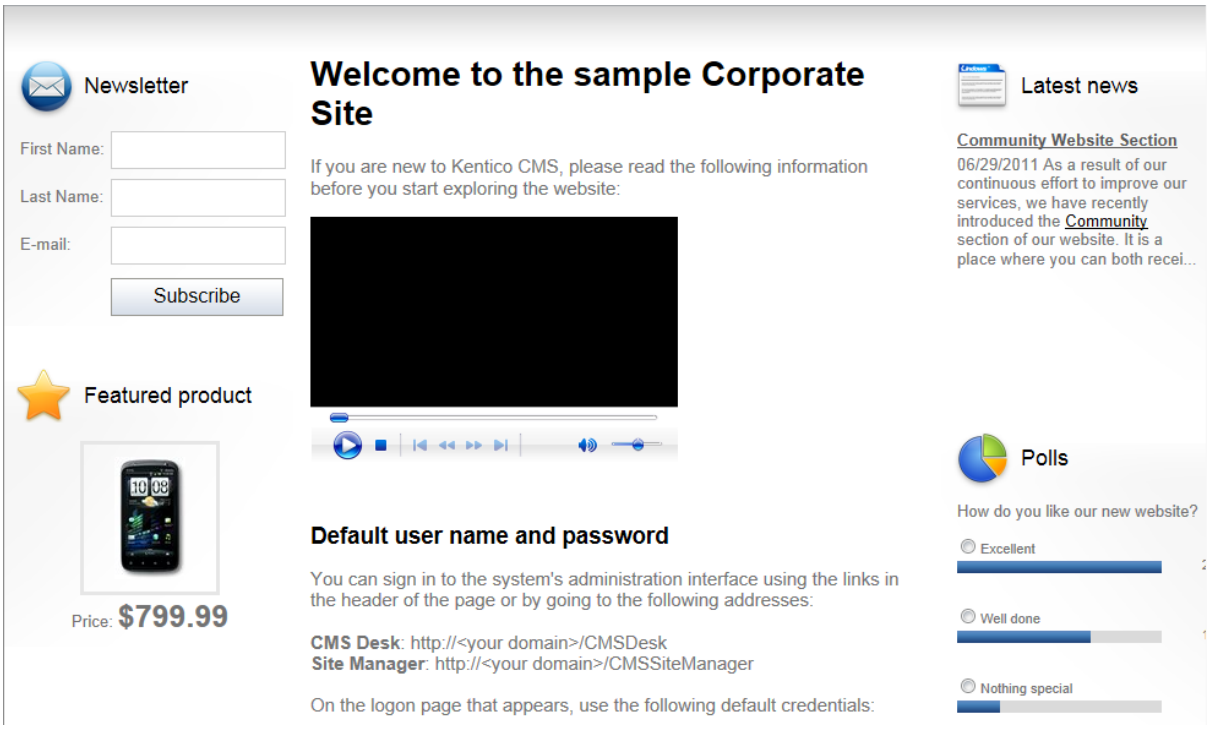
the following code sample:

```
<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:



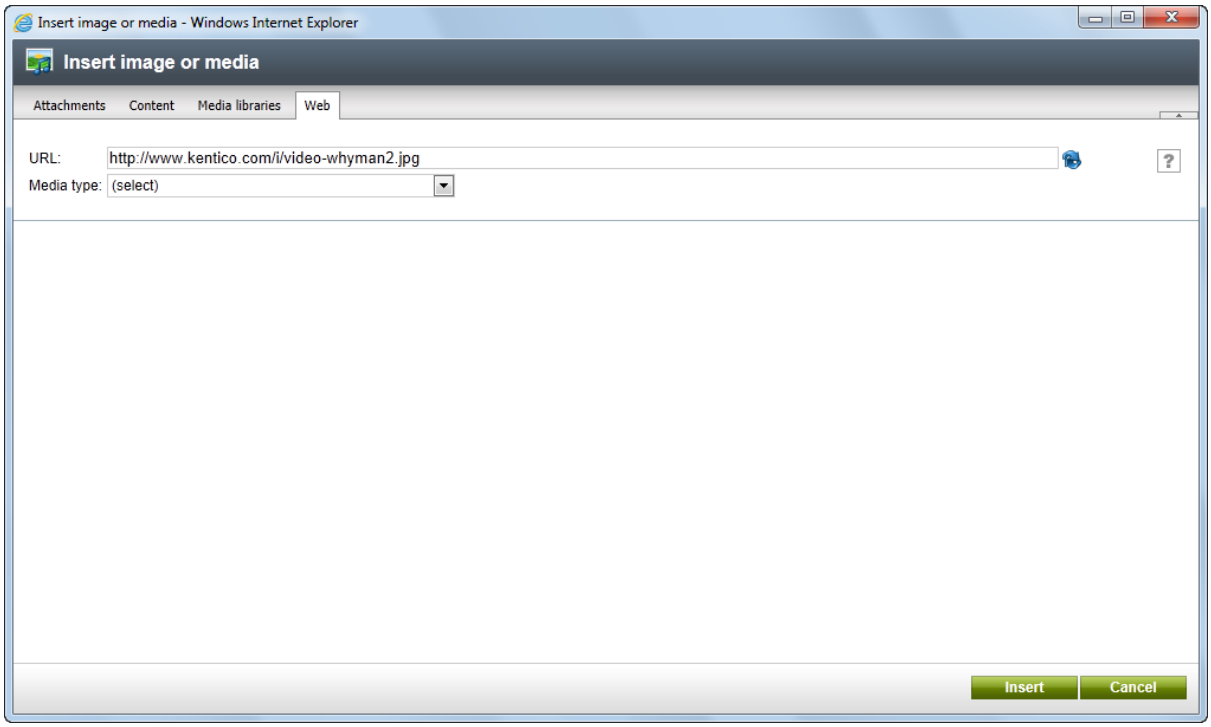
And this is how it looks on the live site:



4.4.2.7 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:



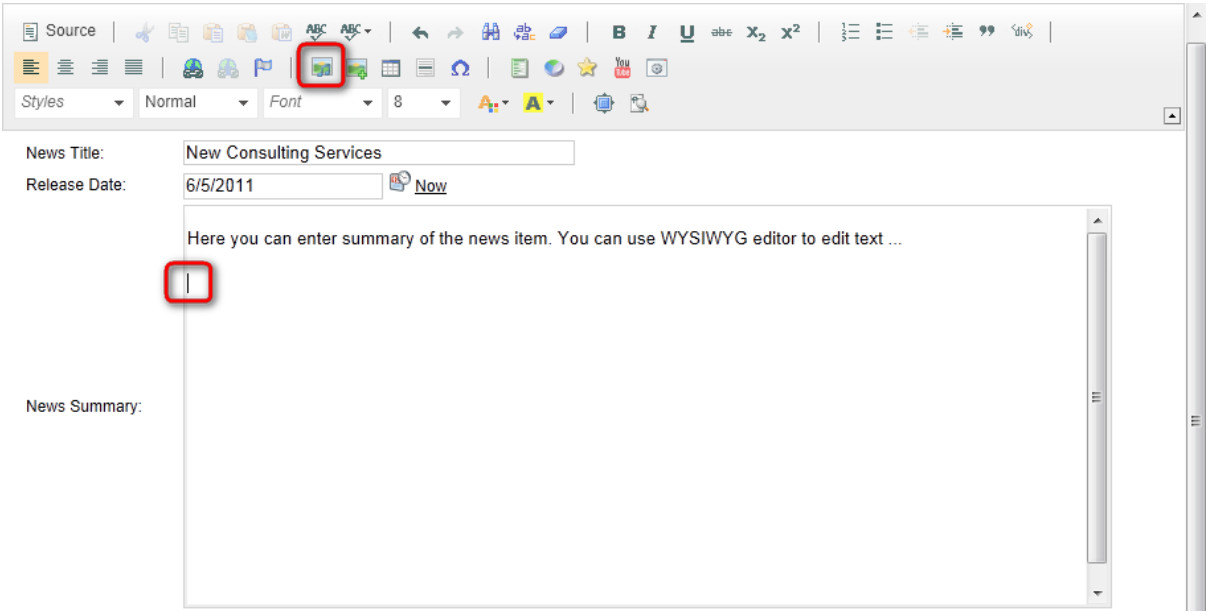
The general process of adding images or media from the web is as follows:

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.

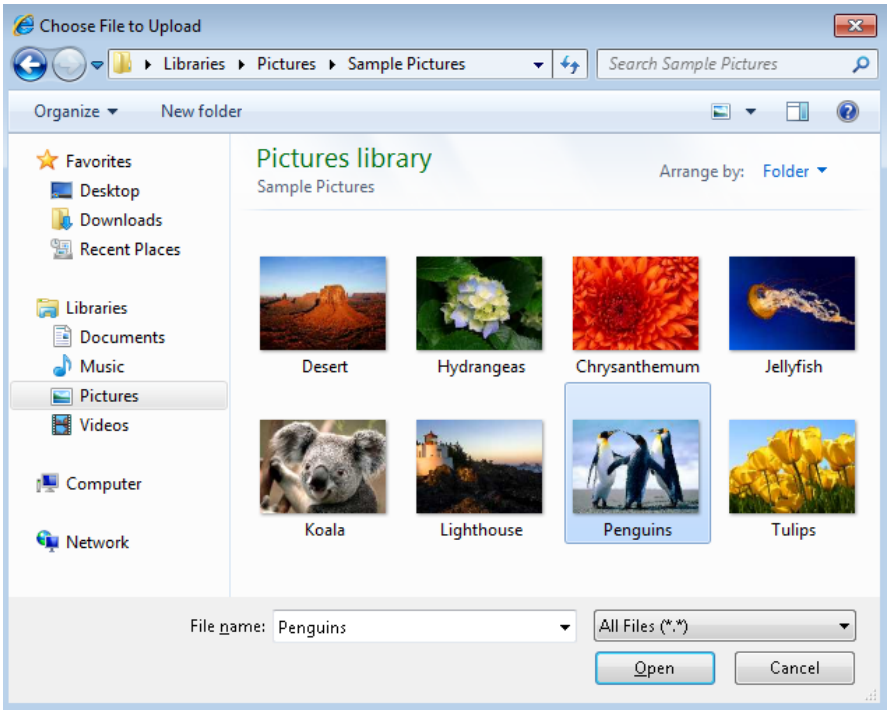
4.4.3 Quickly insert image or media

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**.

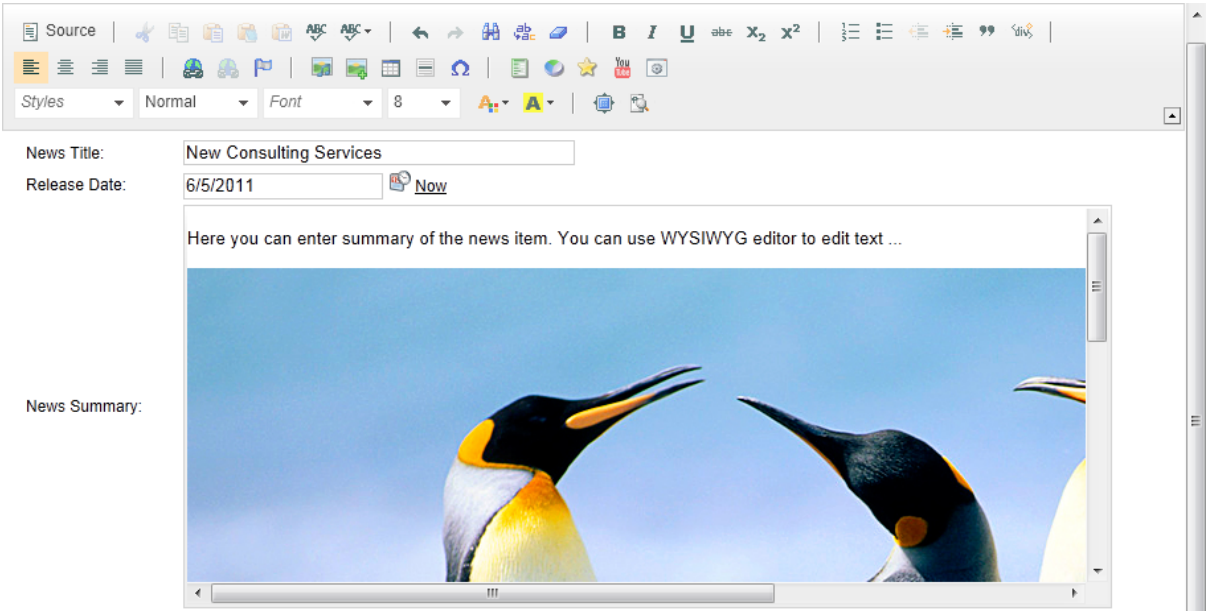


3. Result

Image

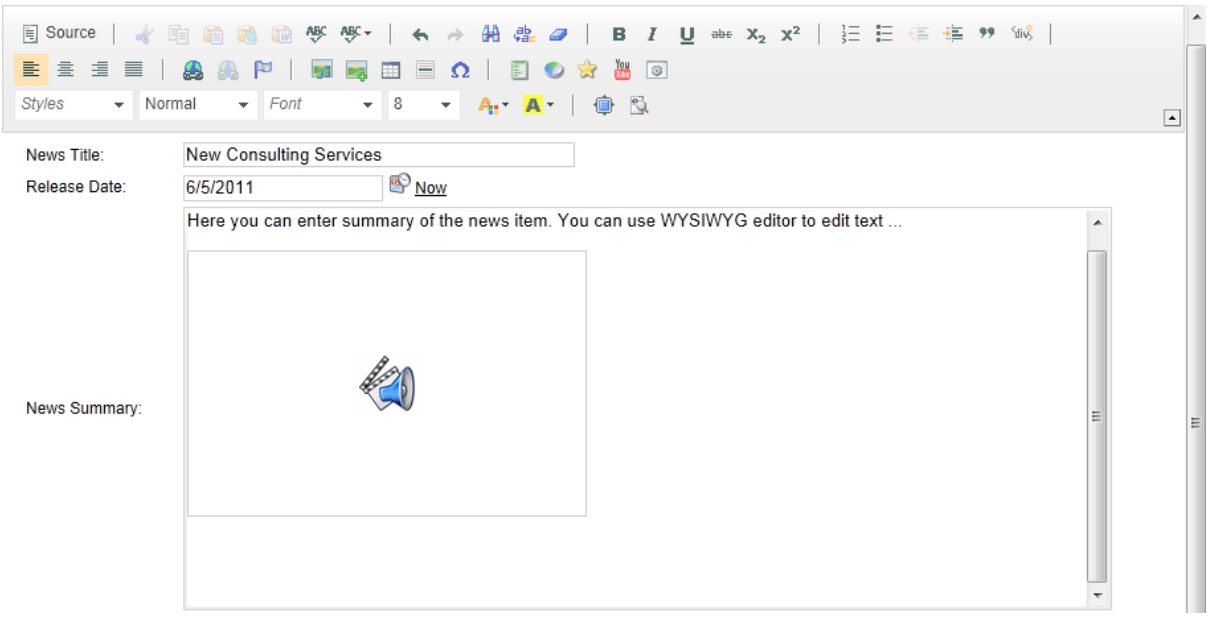
An image is inserted into the text in its original size; to learn how to change this setting, please refer here. At the same time, it is uploaded to the document as its attachment. However, you can edit the image just as images inserted via the **Insert image or media** dialog. For more information, 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.

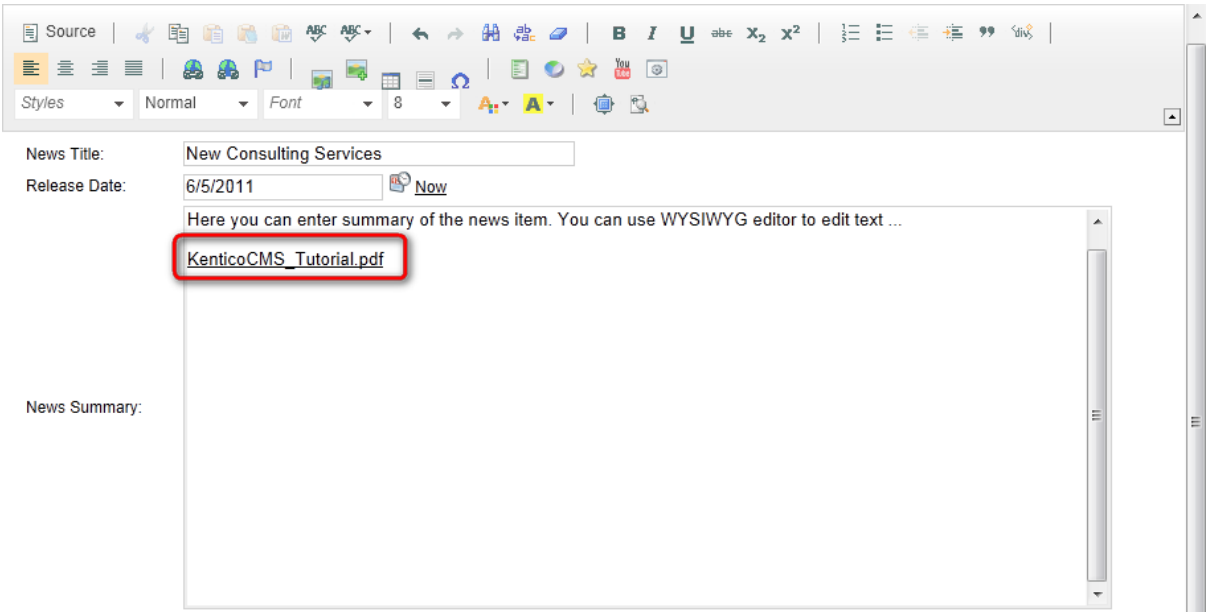


Please note

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.



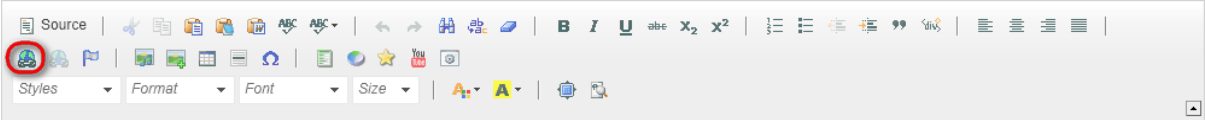
Please note

The link to a document file has the following format: ***<attachment name.file extension>***.

4.4.4 Insert link

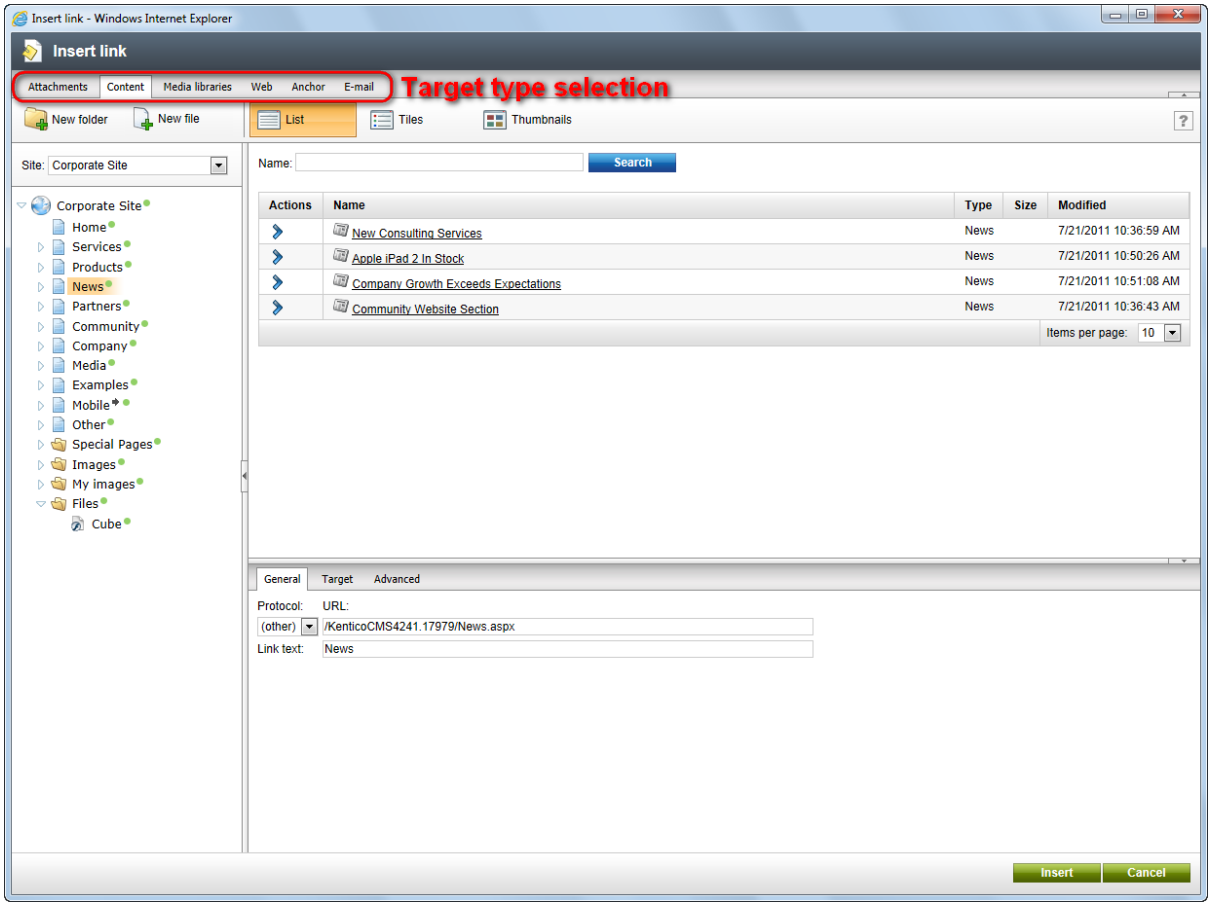
4.4.4.1 Overview

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.

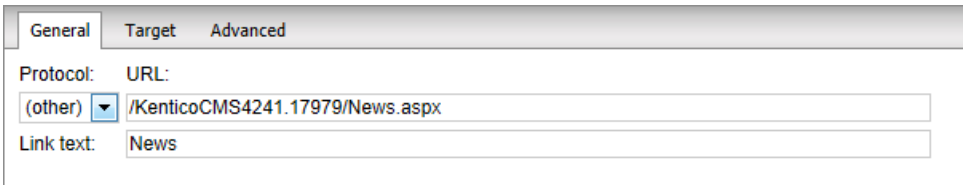


General process of inserting a link

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.

4.4.4.2 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.

4.4.4.3 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.

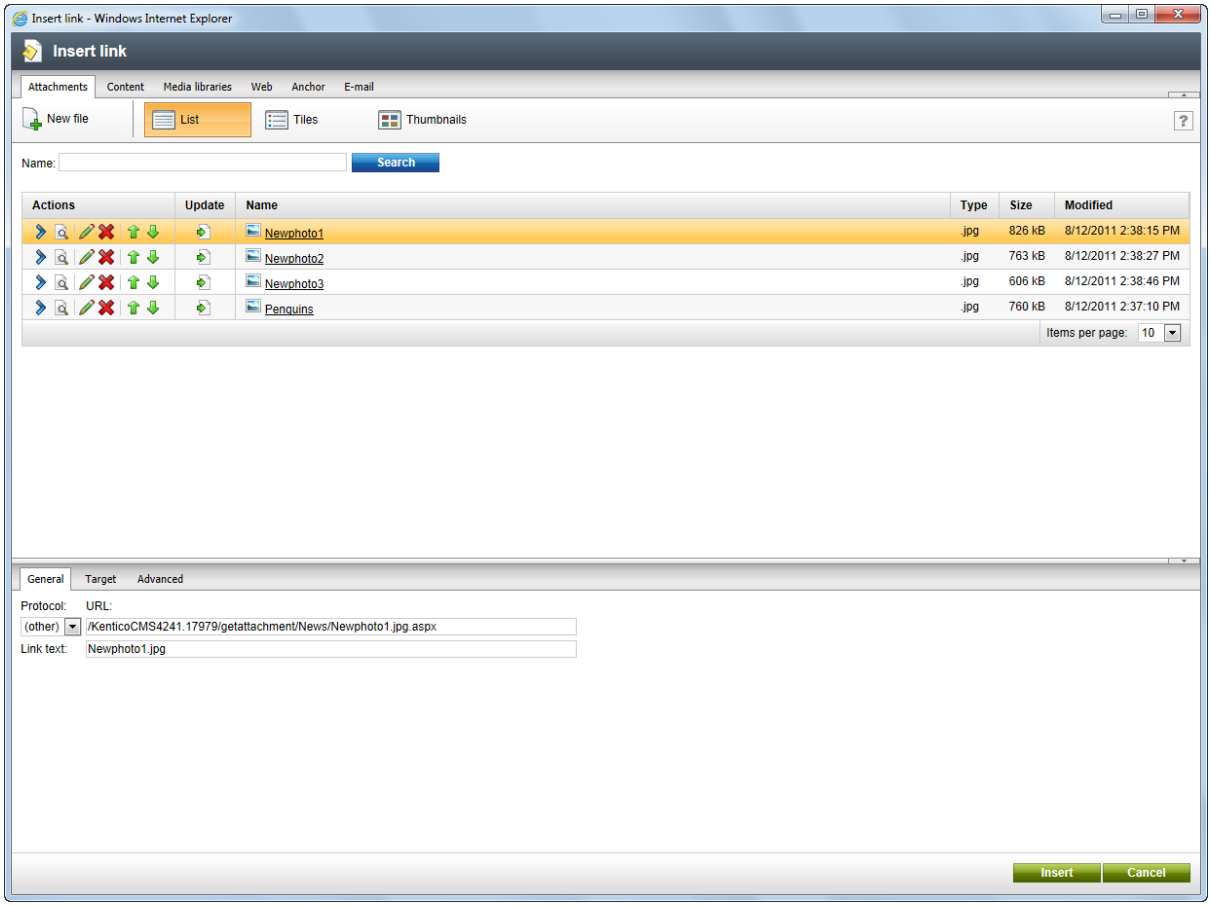
Attachments

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>
```

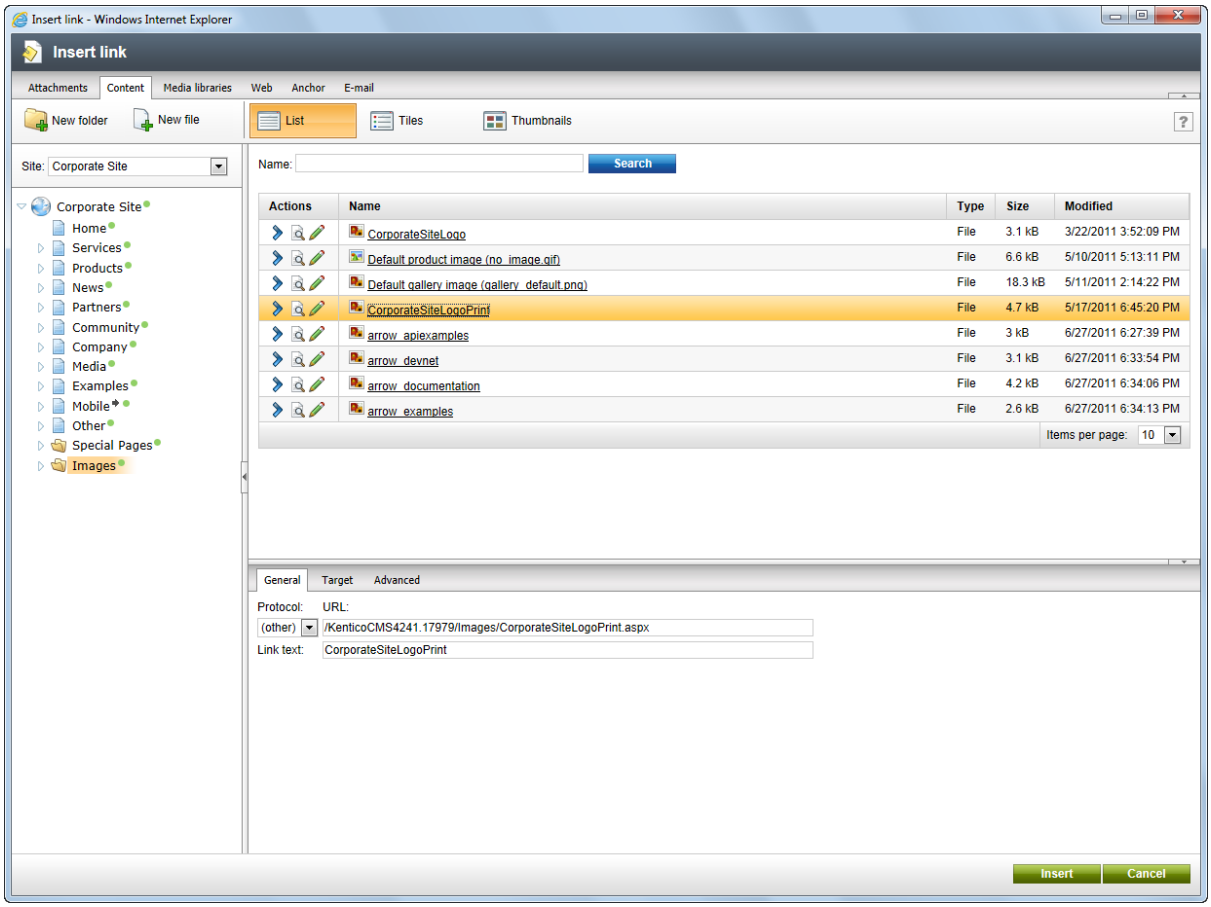


Content

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 Dialogs 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>
```

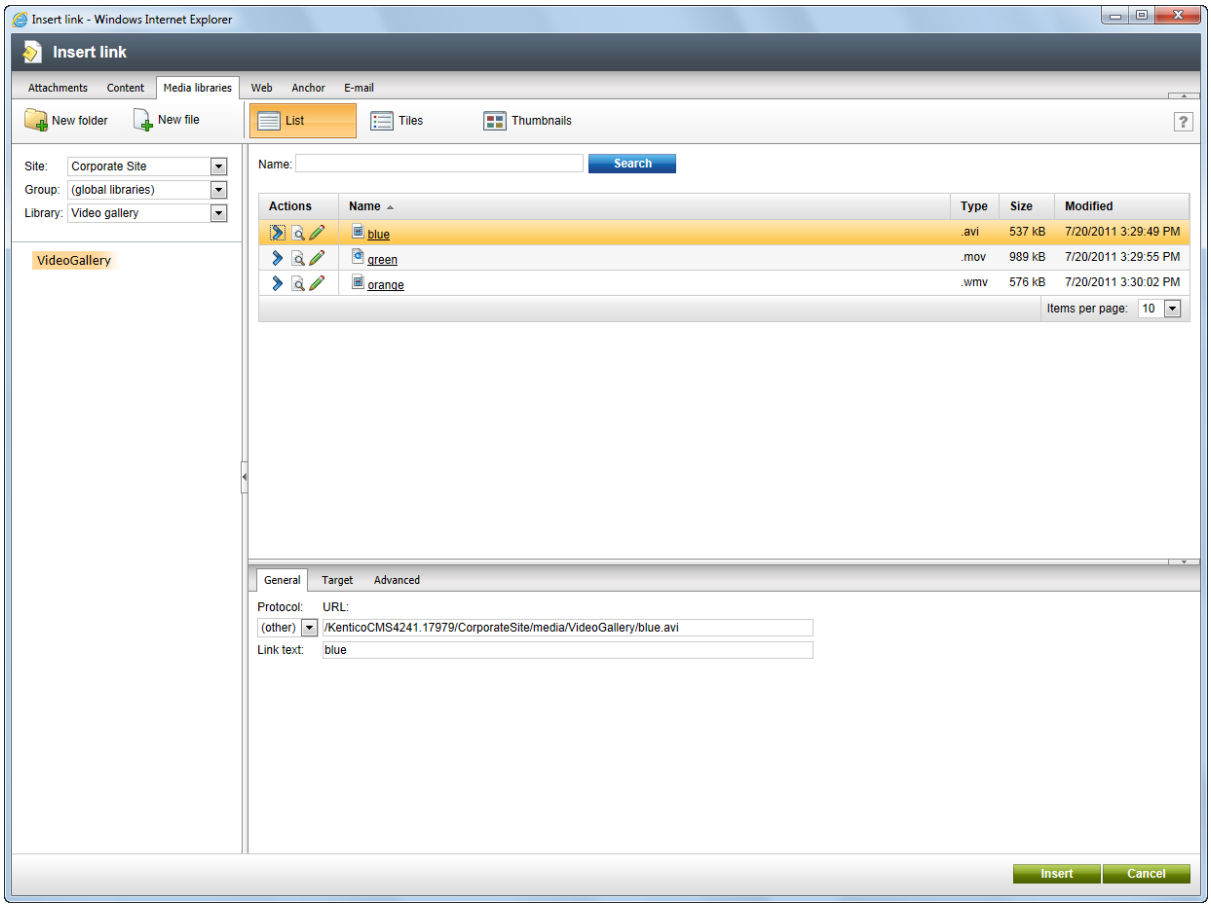


Media libraries

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>
```

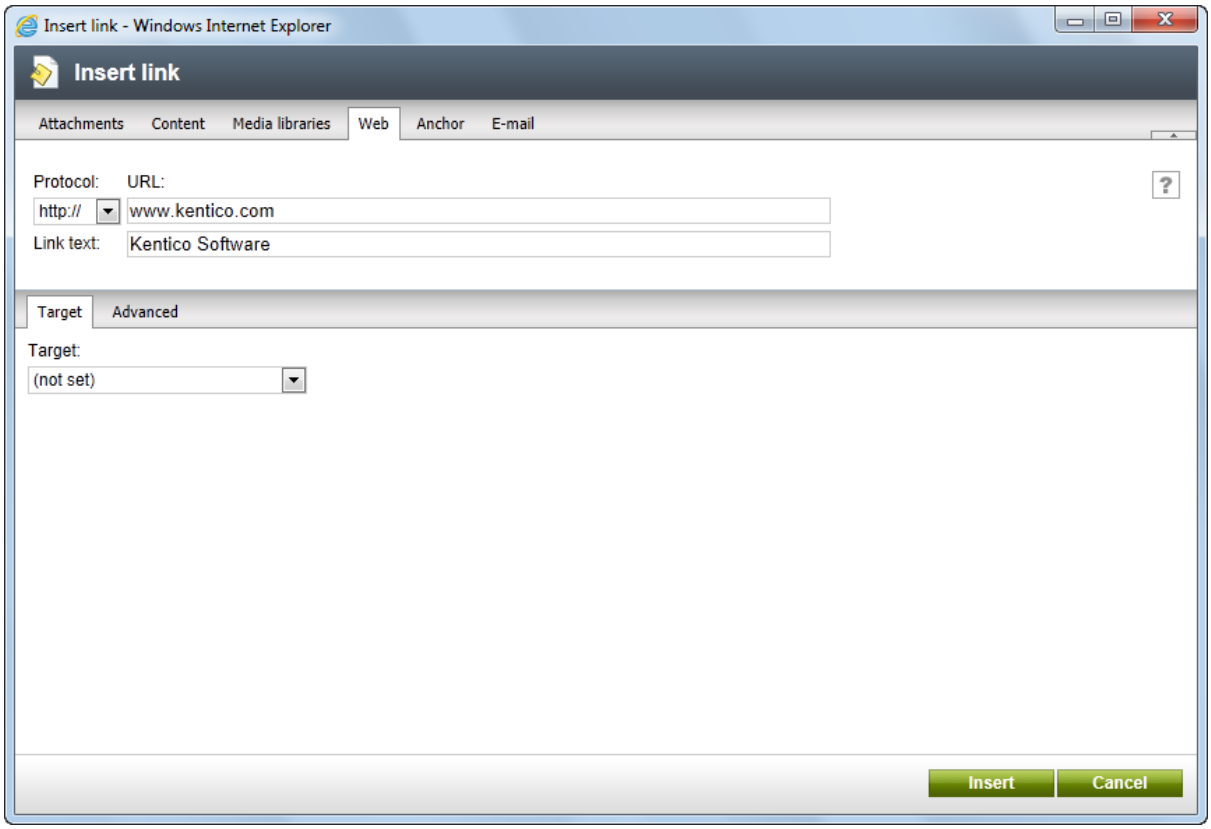


4.4.4.4 Links to the Web

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.



4.4.4.5 Links to anchors

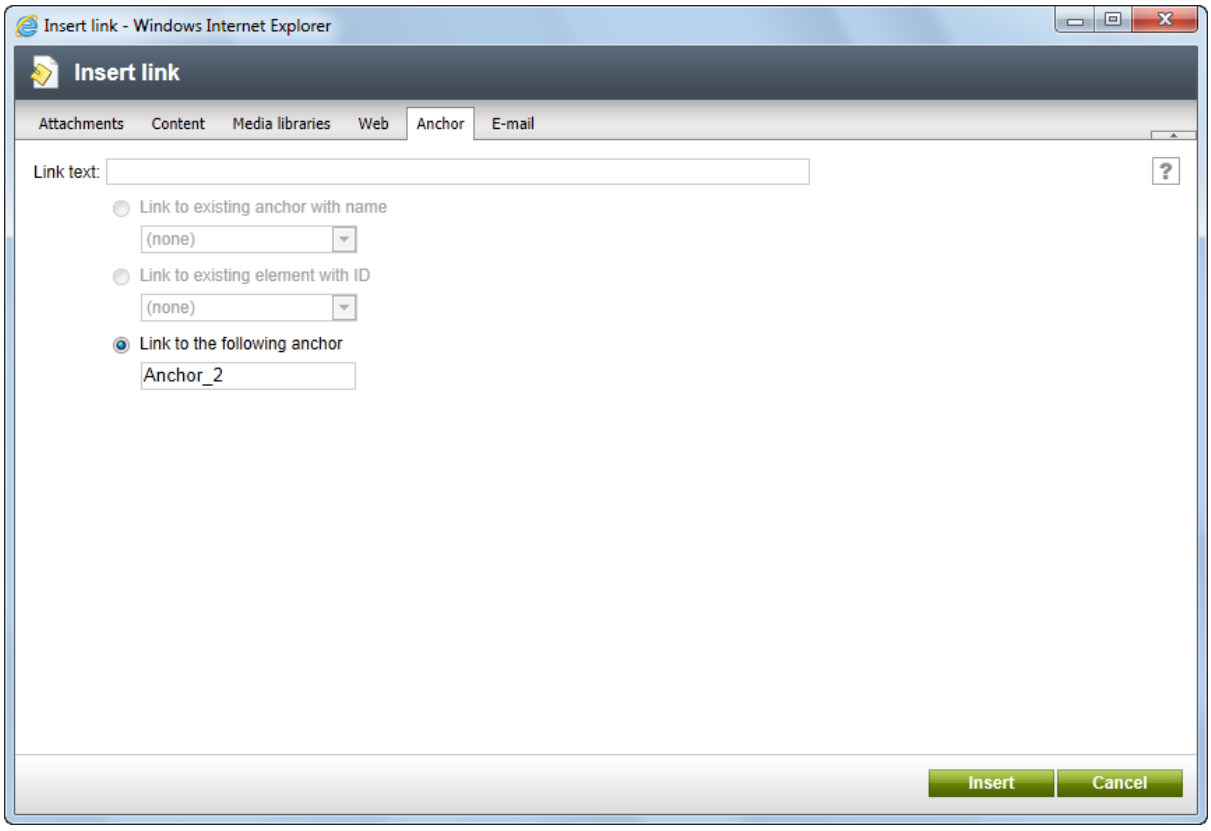
Via the **Anchor** tab, users can insert links to any anchor or any HTML element with a specified ID attribute on the current page. Anchors are <a> html elements with the **Name** attribute specified. They can be inserted using the **Anchor** ( ) icon on the WYSIWYG editor toolbar. If you link to an 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>
```



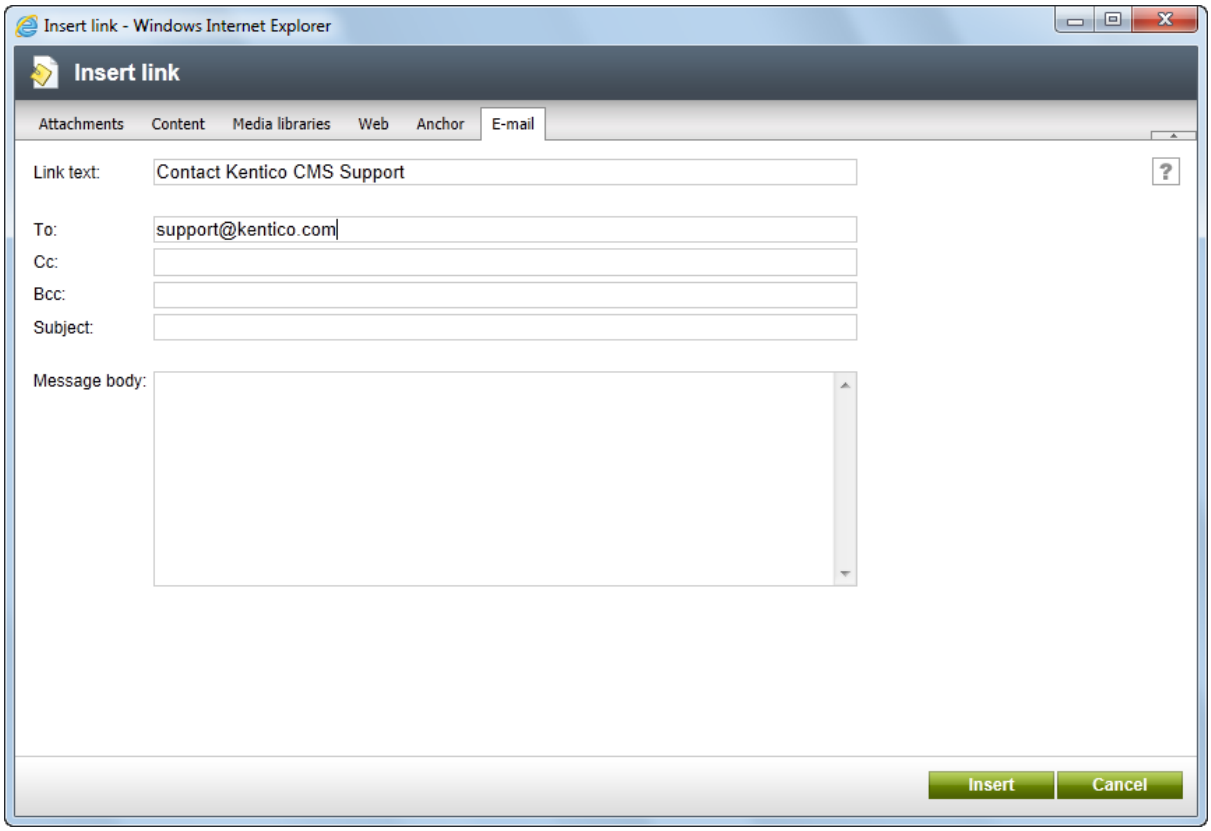
4.4.4.6 Mailto links

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>
```



4.4.5 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 the **Go to YouTube** ( ) link 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:

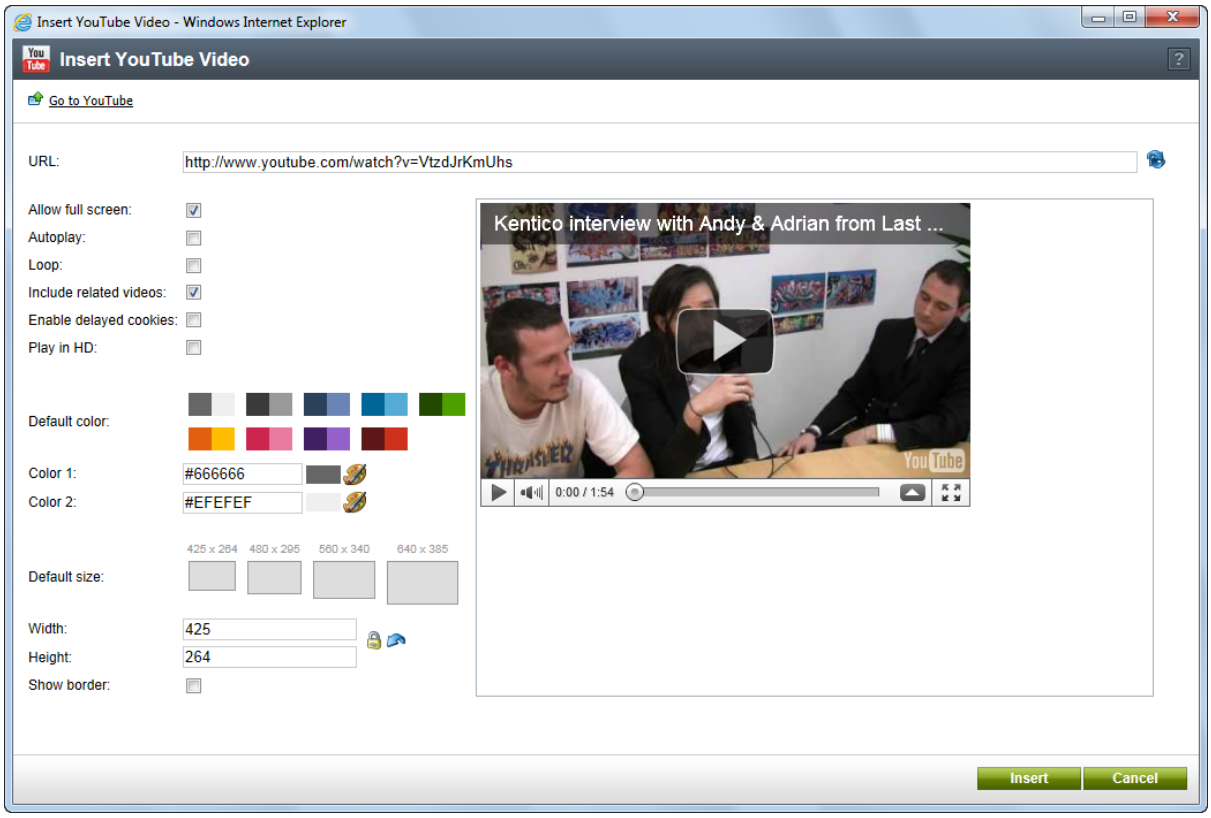
- **URL** - URL of the YouTube video. You can copy&paste it from the address line of your browser or from the **URL** field on the video page.

- **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.

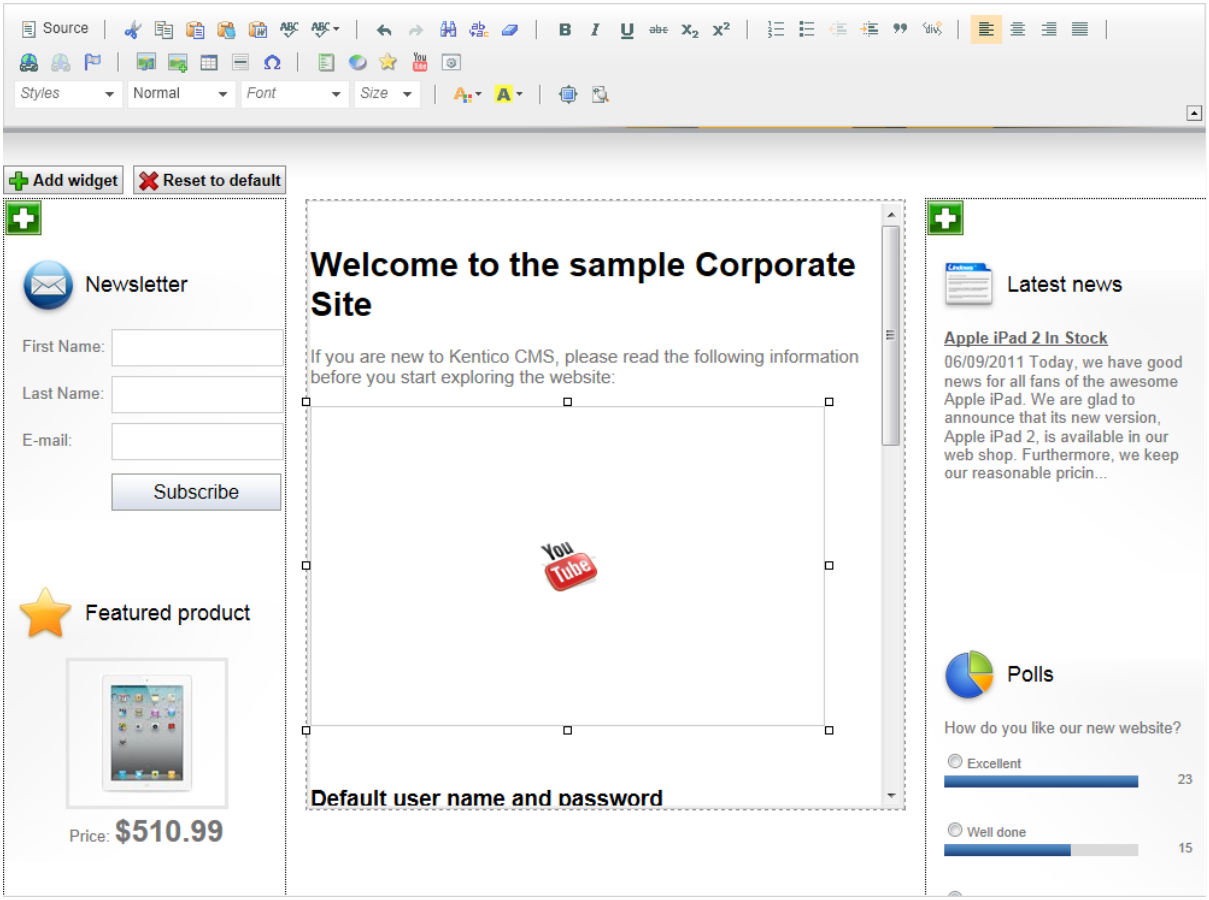
- **Default color** - you can choose one of the default color combinations, which will be used in the **Color 1** and **Color 2** properties.
- **Color 1** - the color of the border around the player, relevant only if the **Show border** property is enabled.
- **Color 2** - the color of the player toolbar.

- **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:



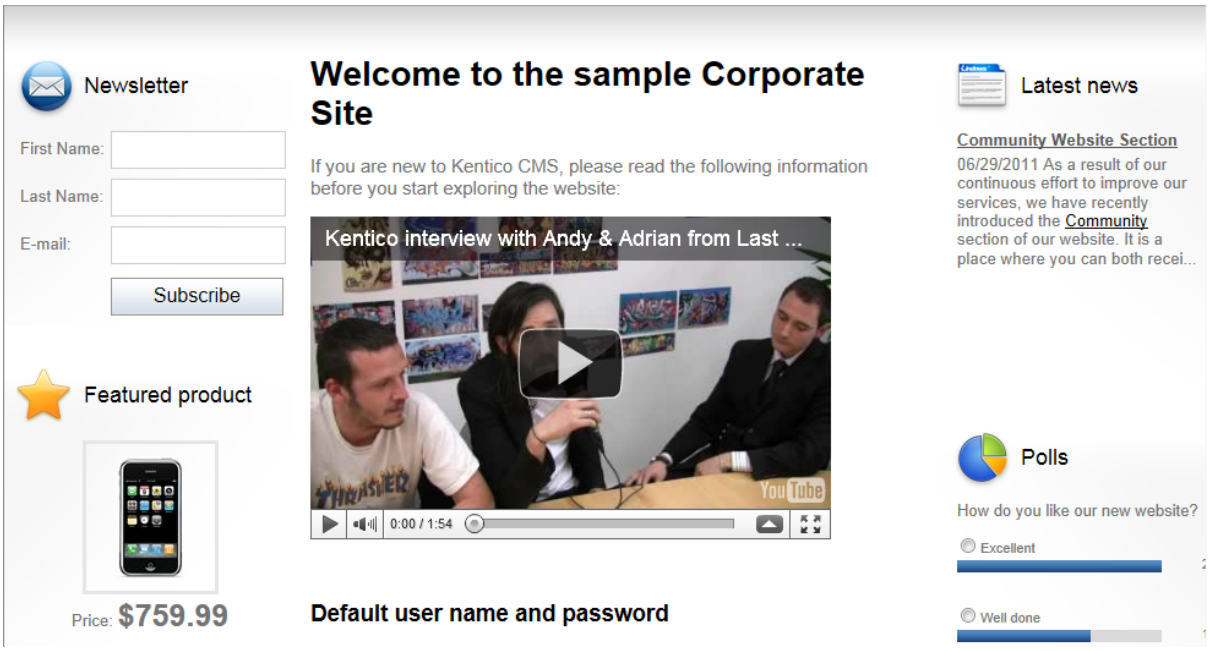
In the WYSIWYG editor, the actual video is not displayed. You can only see a box with the YouTube logo in the middle to give you information about the player size:



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^}
```

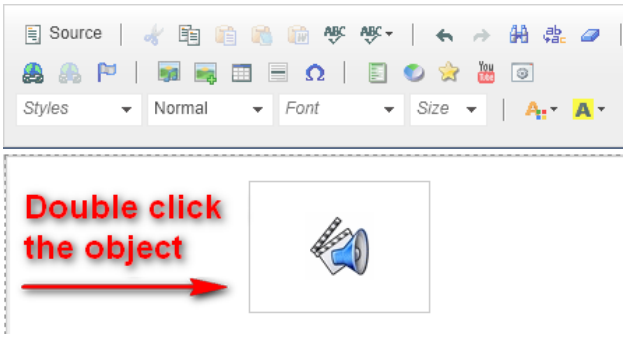
On the live site, the video is displayed in the player:



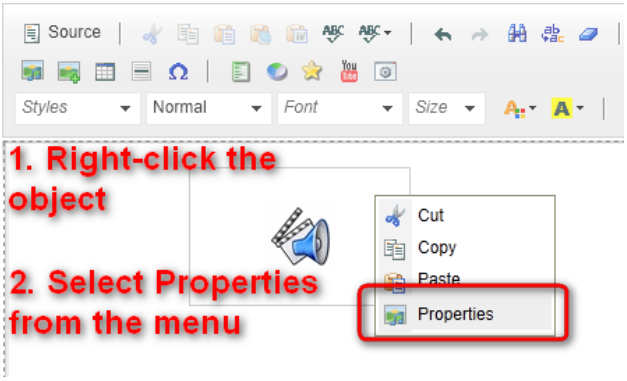
4.4.6 Editing inserted items

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:

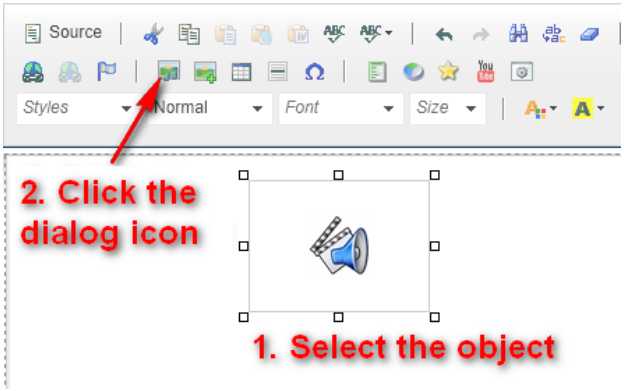
1. Double-click the item.



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.



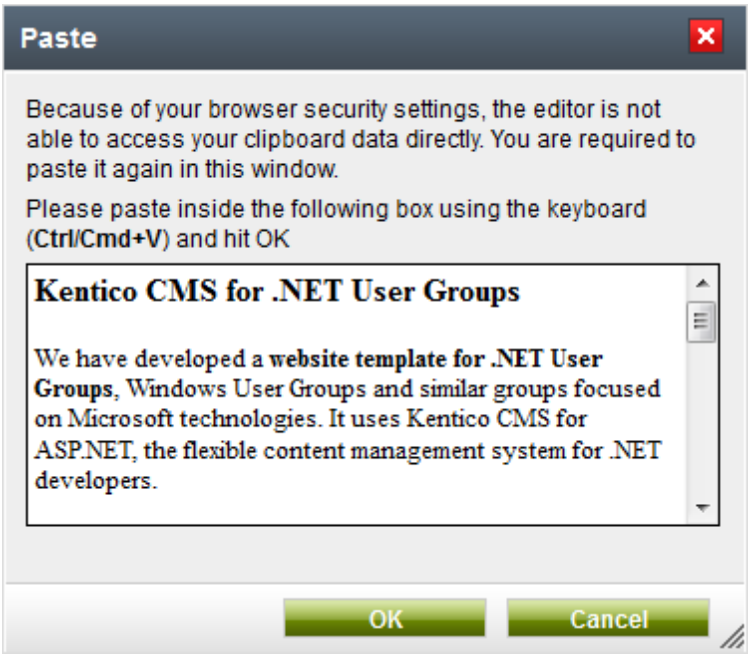
4.4.7 Copy & Paste from Microsoft Word

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.

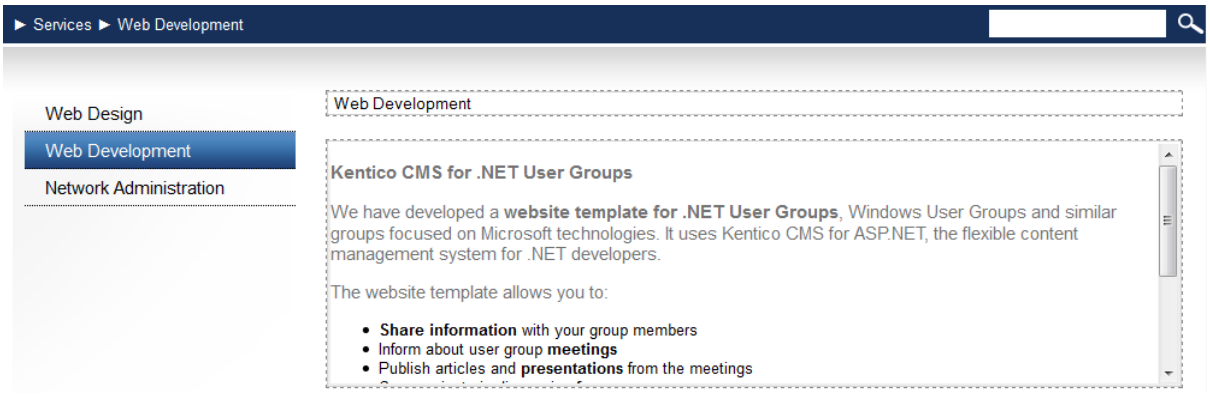
1. Select the text in a Microsoft Word document and copy it to clipboard (**CTRL+C**):

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.

4.4.8 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.

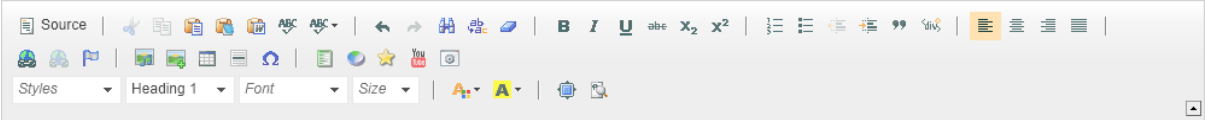
1. Defining toolbar sets

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.

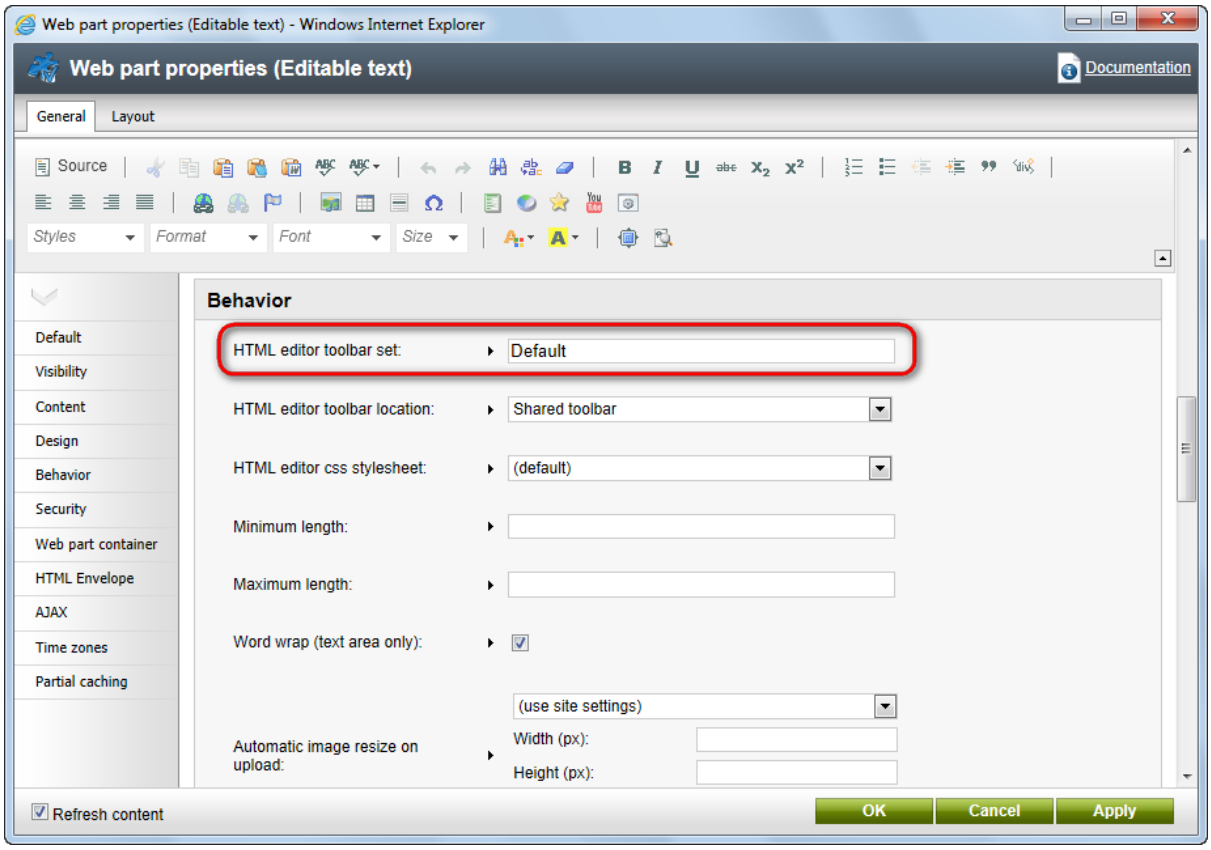
If you want to define your own toolbar set, add a command in the ***config.toolbar_ToolbarName*** format to the **config.js** file. When you have done this, save the changes you made to the file.

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.

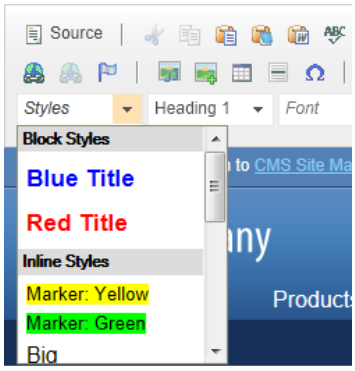
Standard toolbar sets

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.

4.4.9 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>
```

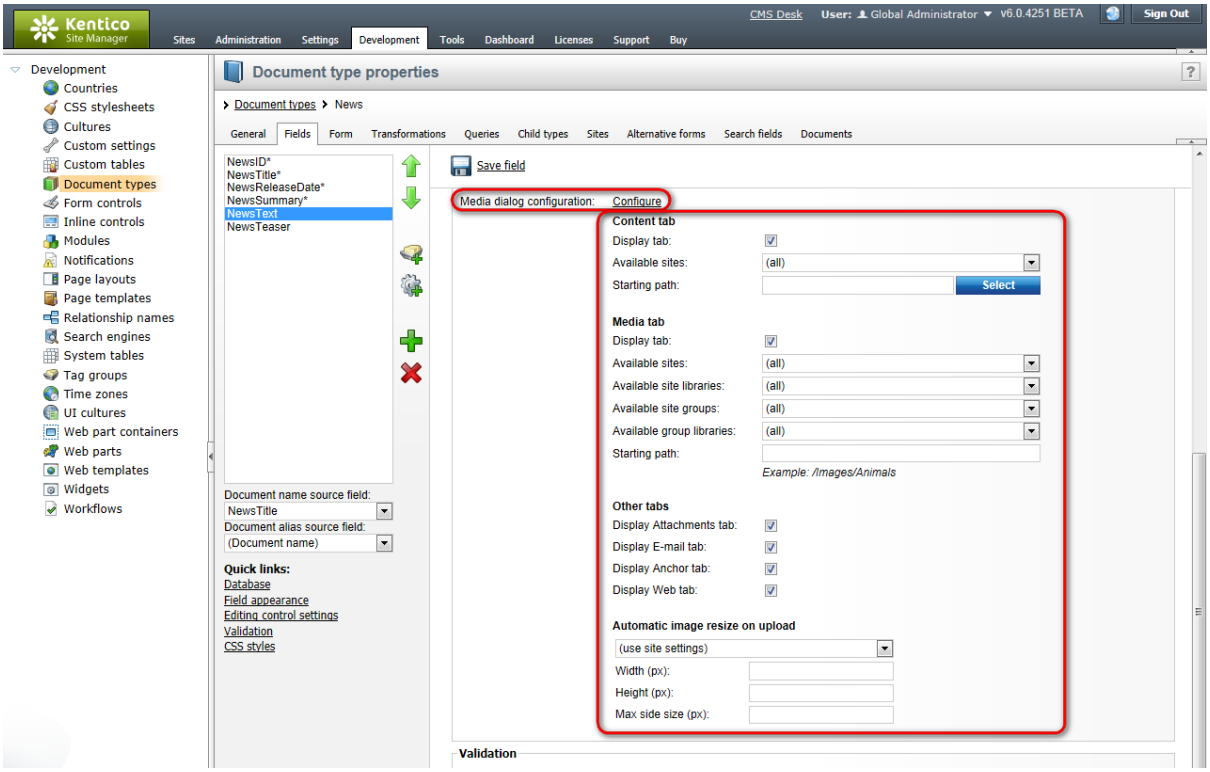
In your CSS stylesheet, you need to define the *GreenText* class name like this:

```
.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.

4.4.10 Dialogs configuration

When defining a new **HTML area (Formatted Text)** form control field, you can display advanced dialog settings by clicking the **Configure** link in the **Media dialog configuration** section.



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.

4.4.11 Dialogs 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 permission	Groups module permissions	Media library permissions

	s											
						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 in dialogs	X											X
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.

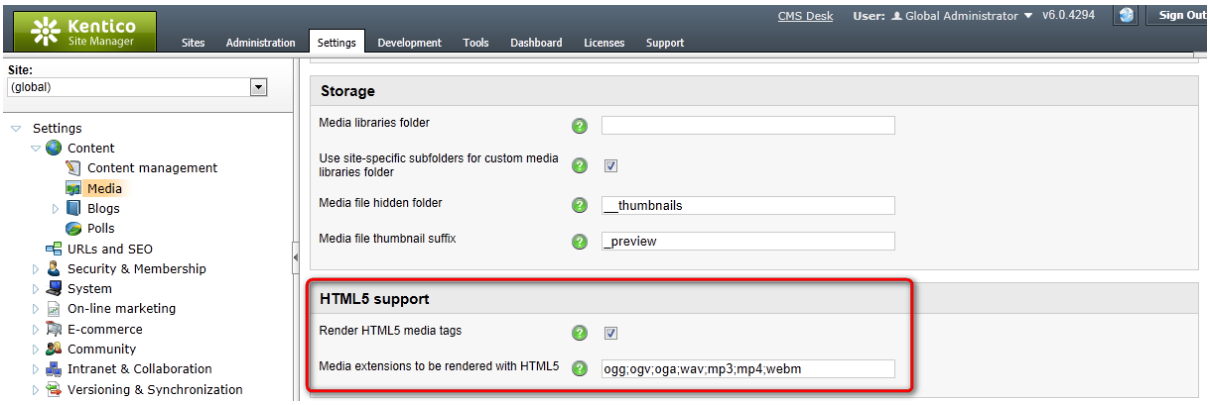
The last two columns indicate if the ***cms.file*** and ***cms.folder*** document types are **allowed child types** of the document under which they should be created. This can be set separately for each document type in **Site Manager -> Development -> Document types -> Edit ( ) the particular document type** on the **Child types** tab.

Action/Permission	Document persmissions					
	Browse tree	Create Manage	Modify	Delete	CMS. File is allowed	CMS. Folder is allowed
Creating new cms.file		X			X	
Creating new folder		X				X
See child documents of the specified document	X					

4.4.12 HTML5 media tags

In **Site Manager -> Settings -> Content -> Media**, you can find the following two settings related to rendering of HTML 5 media tags:

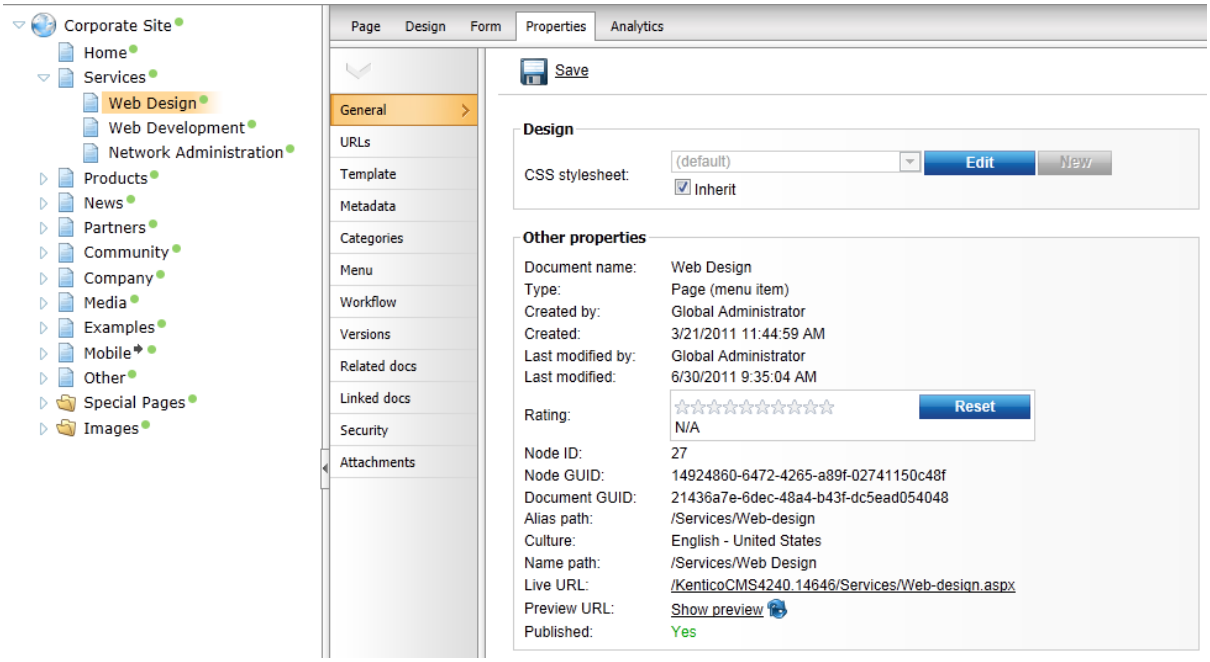
- **Render HTML5 media tags** - if enabled, the system renders HTML5 <*audio*> and <*video*> tags for media content inserted using WYSIWYG editor dialogs.
- **Media extensions to be rendered with HTML5** - specifies a list of media extensions to be rendered with HTML5 media tags. Enter a list of extensions separated by a semicolon, e.g. *.mp4;. ogg*.



4.5 Document properties

4.5.1 Overview

You can edit document properties in **CMS Desk -> Content**, after you click the document and click the **Properties** tab in the **Edit** mode:



4.5.2 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	
Output cache	This option allows you to specify if the page content should be cached in memory (full-page caching). You can choose to inherit the settings from the parent document or to set a different value or to disable full-page caching. The page can be removed from the output cache using the **Clear output cache** button.
Cache minutes	Determines how long the content of this page should be cached if caching is enabled.
Search	
Exclude this document from search	If enabled, the current document and its content (including attachments) will not be included in any searches. This affects both the Smart search module and the SQL search engine. Please note: Objects or documents displayed by web parts will not be excluded by enabling this property for the page that contains them.
On-line marketing	
Log on-line marketing activity	If disabled, page-related on-line marketing activities (e.g. the Landing page, Page view or Content rating activity types) are not logged for the page.
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.

4.5.3 URLs

URLs tab fields:

Document alias	The unique name of the document in the given section of the website. This name doesn't change when you modify the document name and it's used: 1. To define the **AliasPath**, which is the unique path to the document. The developers typically use the AliasPath in the Path property of the 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 with a safe character (by default, it's dash -). You can specify the forbidden characters and the replacement character in **Site Manager -> Settings -> URLs and SEO**, in values "Forbidden characters replacement" and "Forbidden URL characters".
Document URL path	If you want to use a specific URL for a given document that doesn't depend on the document AliasPath, you need to check the box **Use custom URL path** and enter the value, such as: **/cms-for-asp-net** The URL path must always start with /. Then, the document is accessible through two URLs: www.kentico.com**/products/cms**.aspx www.kentico.com**/cms-for-asp-net**.aspx The URL path is useful if you need to define a short URL of some page or if you need to optimize the URL for search engines. The URL path is, unlike the alias path, culture-specific. It means if you use URL paths: /product (for English version) /produkt (for German version) then when the user comes to /produkt.aspx, the language of the website is automatically switched to German.
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.
Track campaign	A web analytics campaign with the same name as entered here will be created on the first access to the page through the entered URL.
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 create a new alias using the '**Add new alias**' link.

4.5.4 Template

In the Template dialog, you can specify the page template that will be used for the current document.

General properties

The page can be based on some specified page template or inherit the page template from the parent document.

Template	Current template name. Different template can be chosen after clicking the **Select** button.
Save as new template...	Saves the current template appearance as a new re-usable page template.
Inherit template	Indicates if page template should be inherited from the parent page.
Clone as ad-hoc template	Creates a new ad-hoc template based on the current page template. In many cases, you may want to re-use the ad-hoc page template for other pages. In this case, you need to save the ad-hoc template as a new re-usable template**.**
Edit template properties	Displays a new window with the template properties and settings.

Inherit content

This option allows you (developer) specify if and how the content should be inherited from the parent page(s). It overrides the content inheritance settings of the page template.

Inherit all	Document inherits all the content from all parent pages.
Do not inherit any content	Document does not inherit any content from the parent page.
Inherit only master page	Document inherits content 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.
Select inherited levels	Using this option, you can specify from which particular levels should the content be inherited.

4.5.5 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 title	Title of the browser window (<title> element inner text).
Page description	Page description used for meta tags of the page and for searching content of the website.
Page keywords	Keywords used for meta tags of the page and for searching content of the website.
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 has also an insinuation function implemented. This functions offers you tags from the selected tag group while you are writing.

Global Settings

You can configure prefix of the page title, description and keywords for all documents on the website using the **Site Manager -> Settings -> Content** dialog. Here you can configure 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 into the title or other metadata fields. See Development -> Macro expressions for more details on macro expressions.

4.5.6 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 the **New category** link, 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.

4.5.7 Menu

This dialog allows you to specify how the current document is displayed in the navigation.

General properties

Menu caption	The name of the document as it's displayed in navigation. It may be different to the document name. If no value is entered, the document name is used.
Show in navigation	Indicates if the document should be displayed in the navigation (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 user must be allowed to read the given document so that it appears in the navigation controls.
Show in site map	Indicates if the document should be displayed by the Site map web part (in the dynamic site map). **Please note:** the document is displayed in the navigation if all of the following conditions are met: 1. The Show in site map box is checked. 2. The document is published. 3. The type of the document matches the document types configured in the Site map 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 user must be allowed to read the given document so that it appears in the navigation controls.

Menu action

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.	The menu item click doesn't raise any action - the item is disabled. This option is useful if you need to create a menu item for the main section in the drop-down menu that cannot be clicked, but the sub-items can be clicked.
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 the menu item of this document is clicked. Example: *http://www.domain.com* or *~/products.aspx*

Menu item design properties

The menu item design properties are available in three alternatives:

- standard design
- mouse-over design - when you mouse-over the menu item
- highlighted design - style of the selected document

These values override the settings of the menu control (web part) and the CSS styles defined in the CSS stylesheet.

Please note: some of the following properties may not be applied to the menu control depending on the menu control you are using.

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.

4.5.8 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.

4.5.9 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.

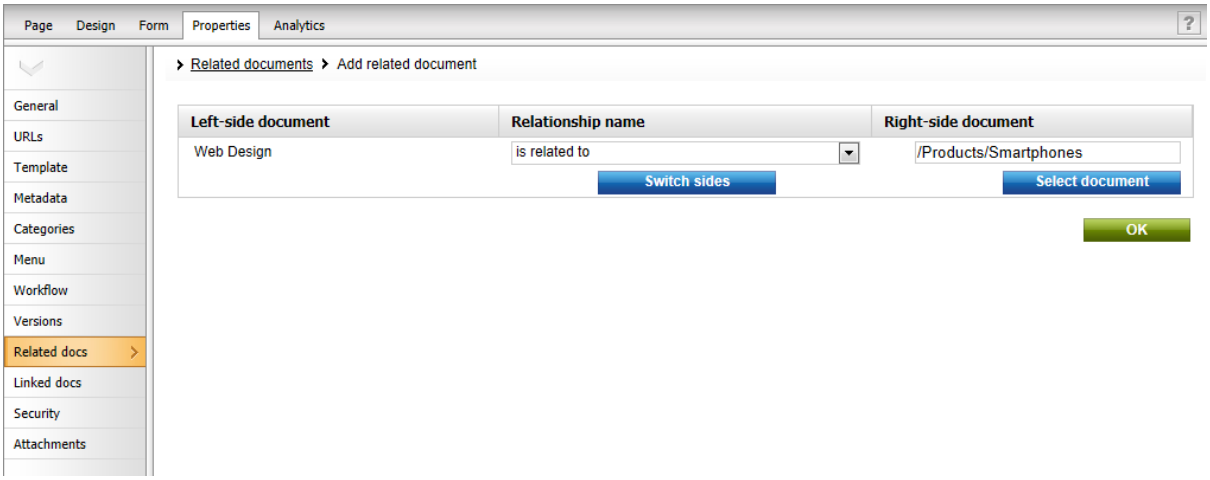
See chapter Workflow and versioning for more details.

4.5.10 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



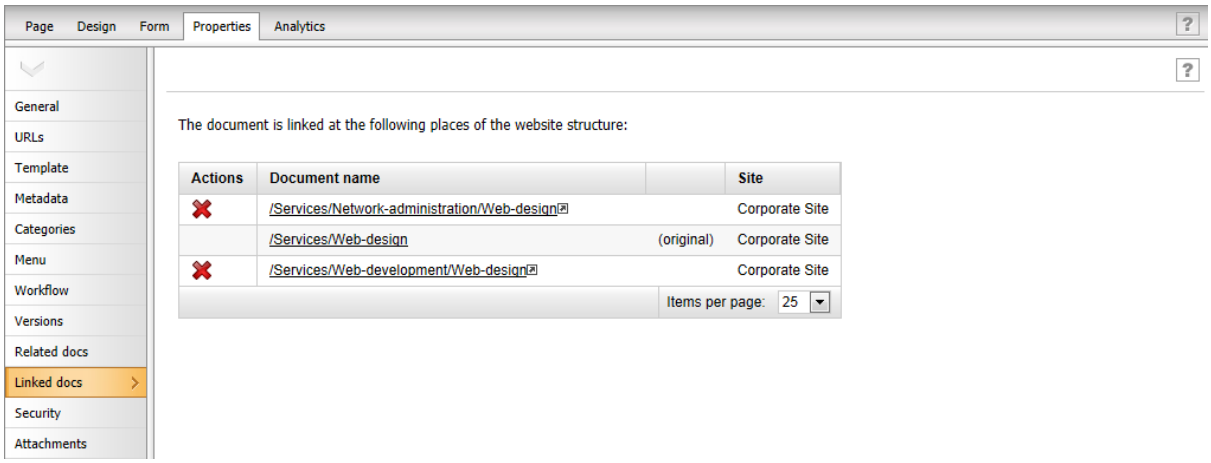
The related documents need to be displayed by a control (web part) that is configured for displaying related documents. If you're a developer, please see the **Kentico CMS Web parts and Controls reference -> Displaying related documents** for more details.

The relationship names need to be defined in **Site Manager -> Development -> Relationship names**. Only relationship names enabled for the current website are available in the drop-down list.

4.5.11 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:



4.5.12 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

4.5.13 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 using the **New attachment** ( ) link. 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.

4.5.14 Languages

This tab is displayed only only for multilingual sites with Translation management enabled. See the Multilingual and international support -> Translation management chapter for more details.

On this tab, you can view the language versions of the selected document. The language versions can have the following statuses:

The colors indicate the following translation statuses:

- **Translated** - the document is translated and up-to-date
- **Outdated** - the document is translated but outdated, which means that the default language version has been modified (or published when using workflow) more recently than the translated version
- **Not available** - the document's version in the language does not exist

By clicking the **Edit culture version** ( ) icon, you can be redirected to the **Edit -> Page** mode of the selected culture version. By clicking the **Add new culture version** ( ) icon of a not-available version, you can proceed to creating the new culture version of the document.

4.6 File management

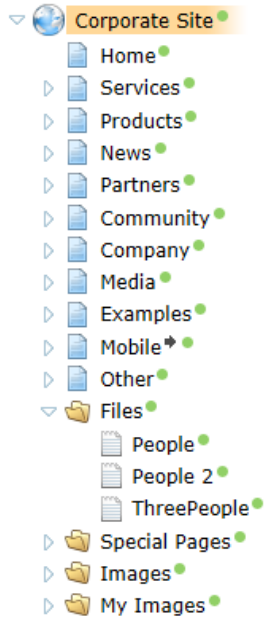
4.6.1 File management overview

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.

4.6.2 Document attachments

4.6.2.1 Overview

Document Attachments are a concept of attaching files to documents in Kentico CMS. 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.

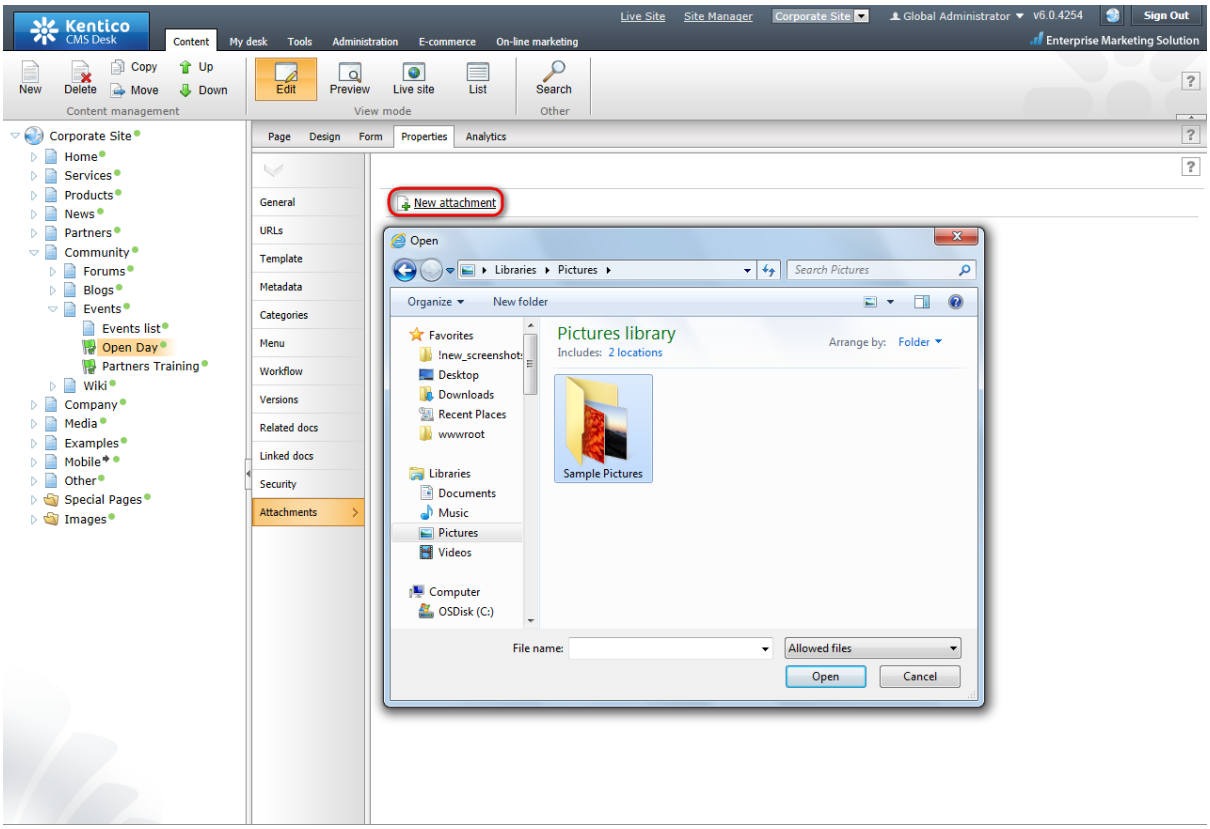
4.6.2.2 Example: Unsorted attachments

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** ( ) link. 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** ( ) link. 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:

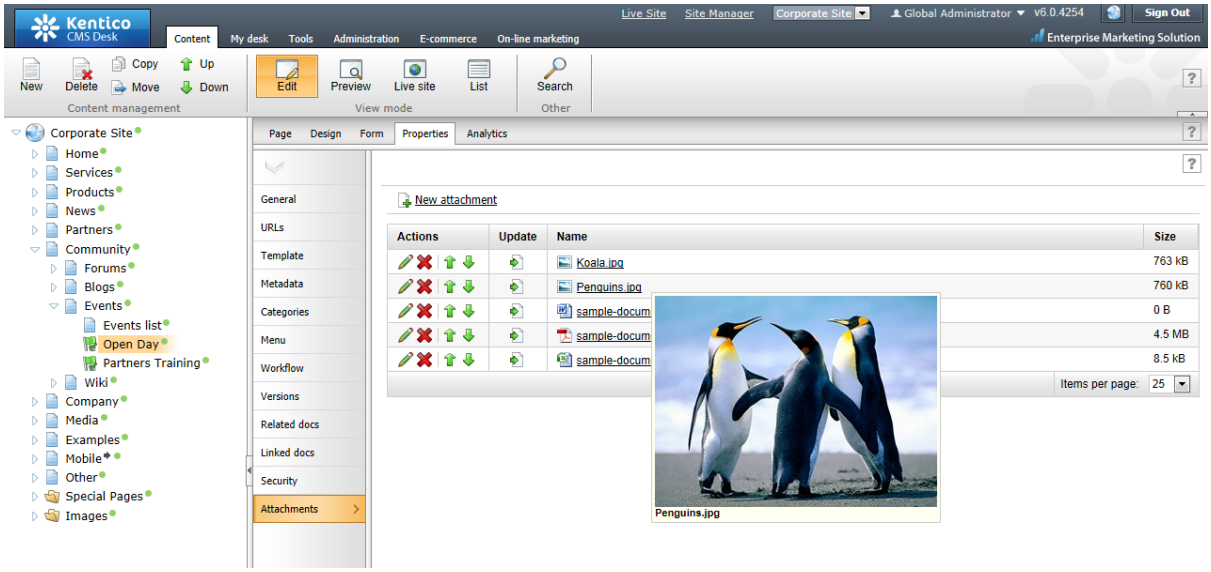
- **Delete** (✖) - removes the attachment from the document.
- **Move up** (⬆) and **Move down** (⬇) - re-orders the attachments. The order is stored in the **AttachmentOrder** property of each attachment. You can enter *AttachmentOrder* into the **ORDER BY expression** property of a displaying web part 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.

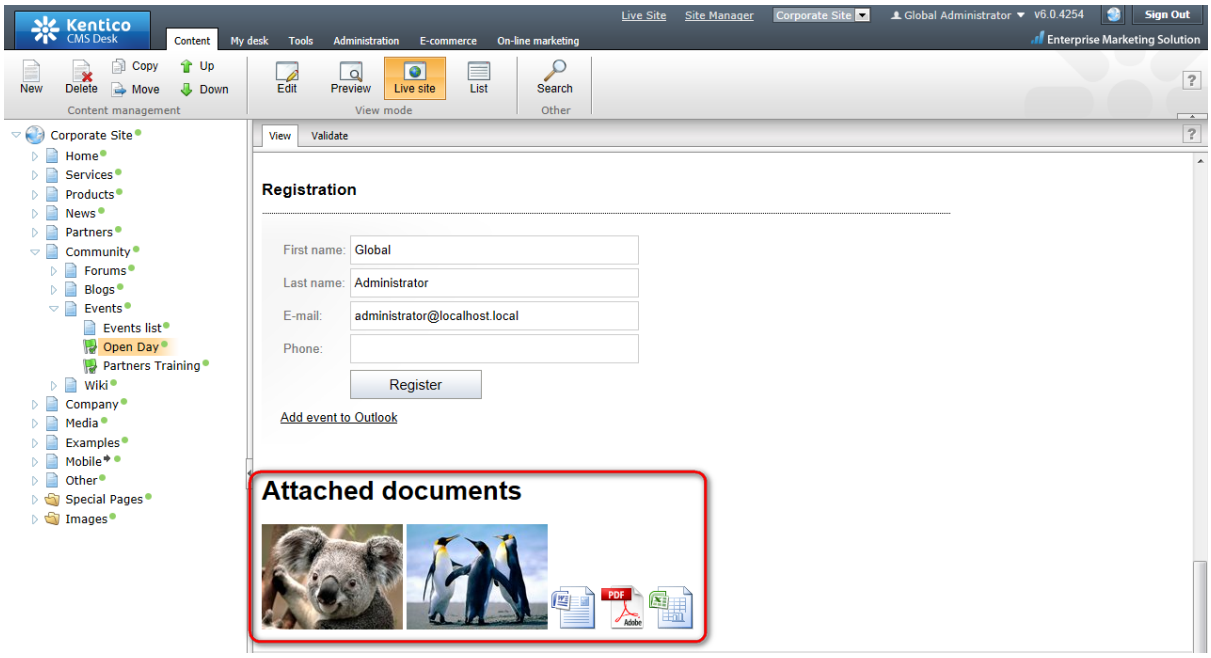


4. Now that we have the attachments uploaded, we can display them on the live site. The events are displayed by a repeater on the **Events** page, so we will have to add the **Document attachments** web part to this page.

Select the **Events** page from the content tree and switch to the **Design** tab. Click the **Add web part** ( ) icon of **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.

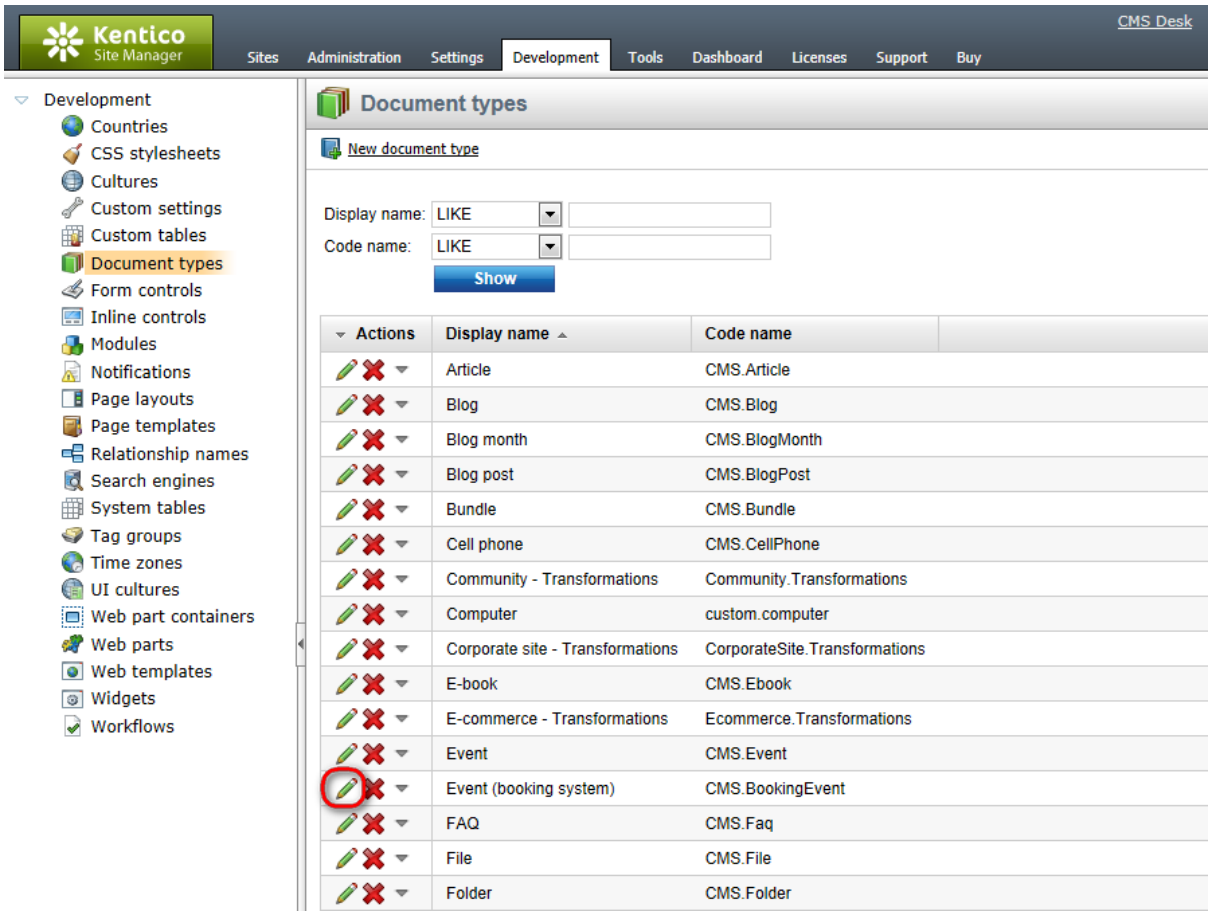


4.6.2.3 Example: Grouped attachments

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 **zoneLeft** 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 **CMS.BookingEvent** document type.

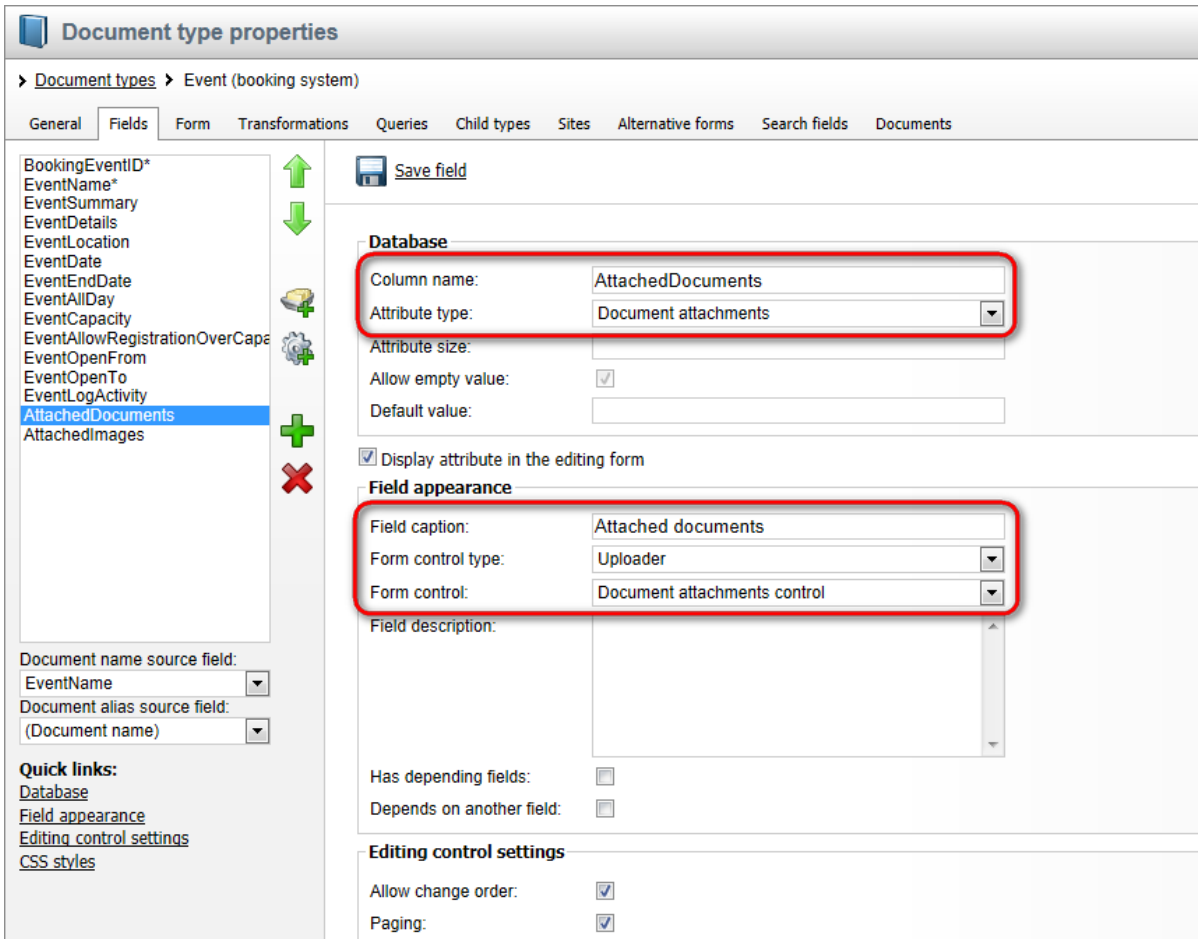


2. Switch to the **Fields** tab. Use the **New attribute** () icon to add the following two attributes. For each attribute, set only the following properties and leave the default values for the rest.

- **Column name**: AttachedImages
- **Attribute type**: Document attachments
- **Field caption**: Attached images
- **Form control type**: Uploader
- **Form control**: Document attachments control

- **Column name**: AttachedDocuments
- **Attribute type**: Document attachments
- **Field caption**: Attached documents
- **Form control type**: Uploader
- **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.



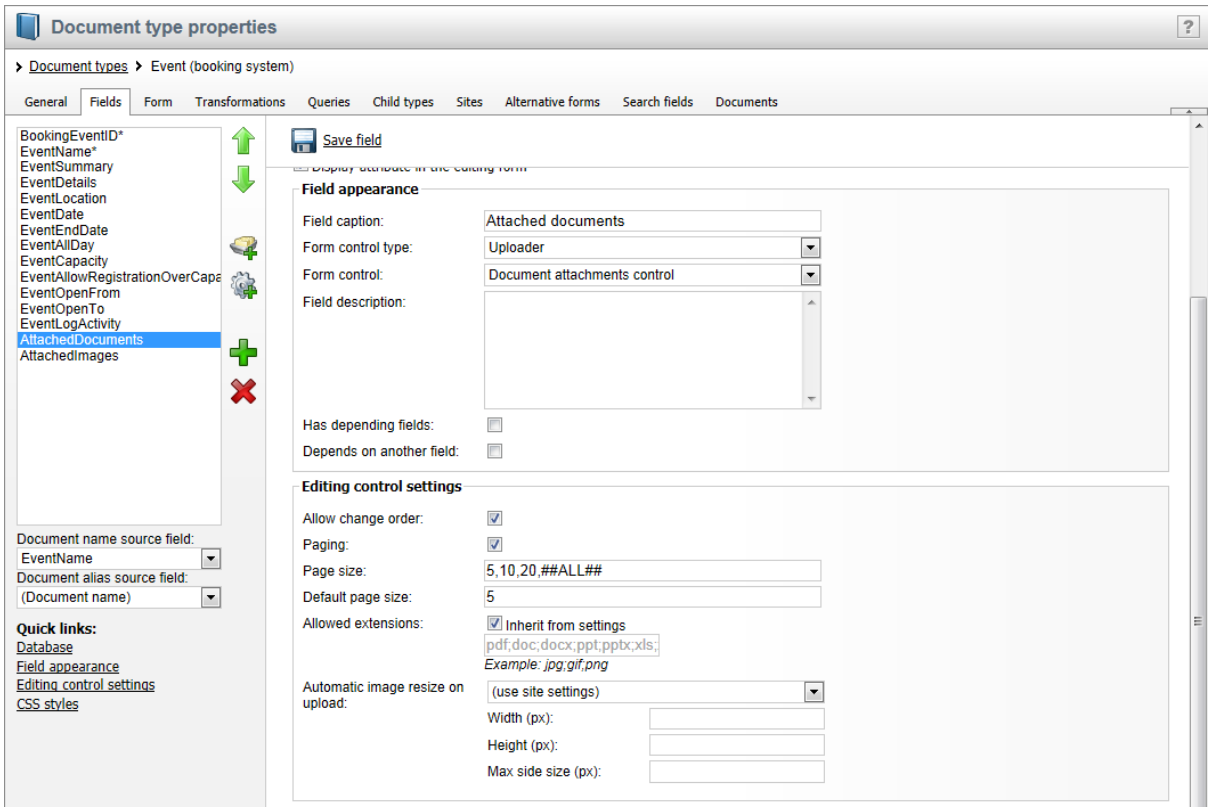
3. Optionally, you can set the following additional parameters for the fields in their **Editing control settings** section:

- **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*

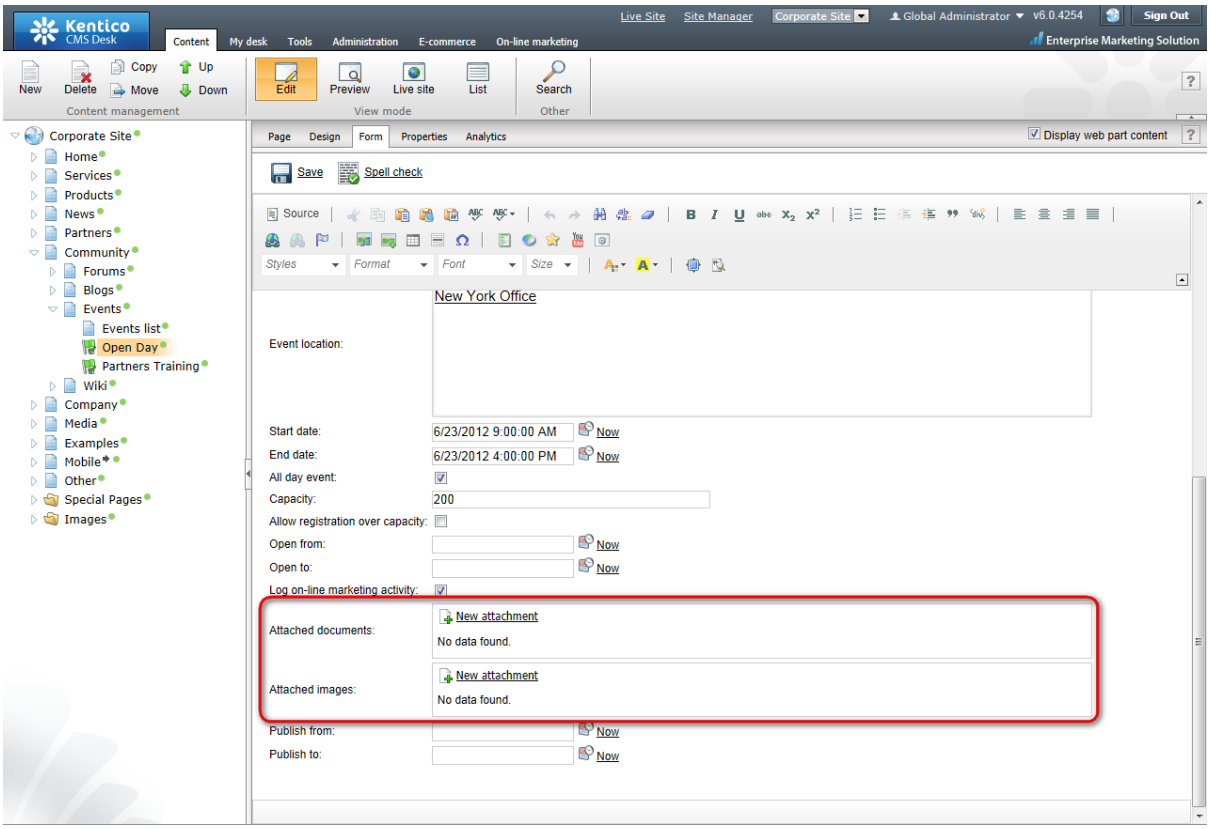
- **Automatic image resize on upload**:
  - **(do not resize)** - uploaded images will not be resized.
  - **(use site settings)** - uploaded images will be resized according to site settings in *Site Manager -> Settings -> System -> Files -> Automatic image resize on upload*.
  - **(use custom settings)** - 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 the **New attachment** ( ) link 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:

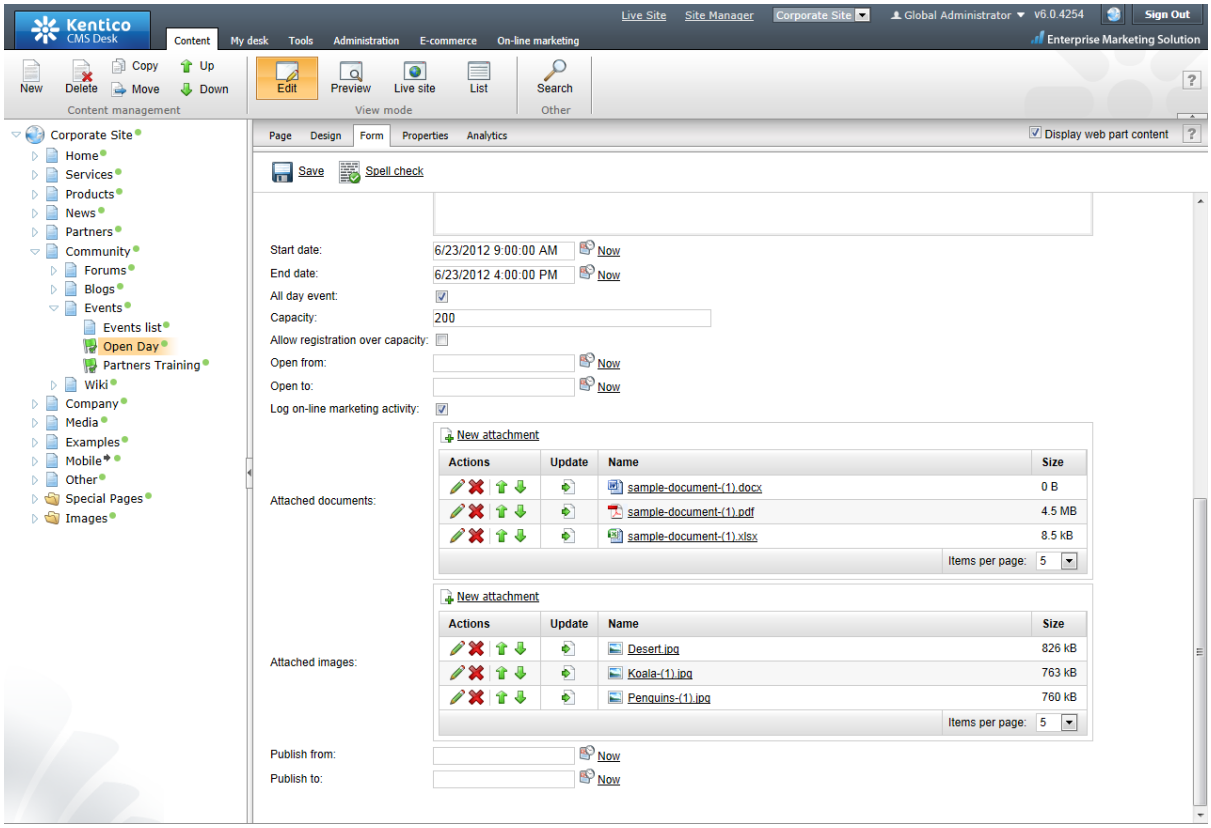
- **Delete** ( ) - removes the attachment from the document.
- **Move up** ( ) and **Move down** ( ) - re-orders the attachments. The order is stored in the **AttachmentOrder** property of each attachment. You can enter *AttachmentOrder* into the **ORDER BY expression** property of a displaying web part 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 **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**.

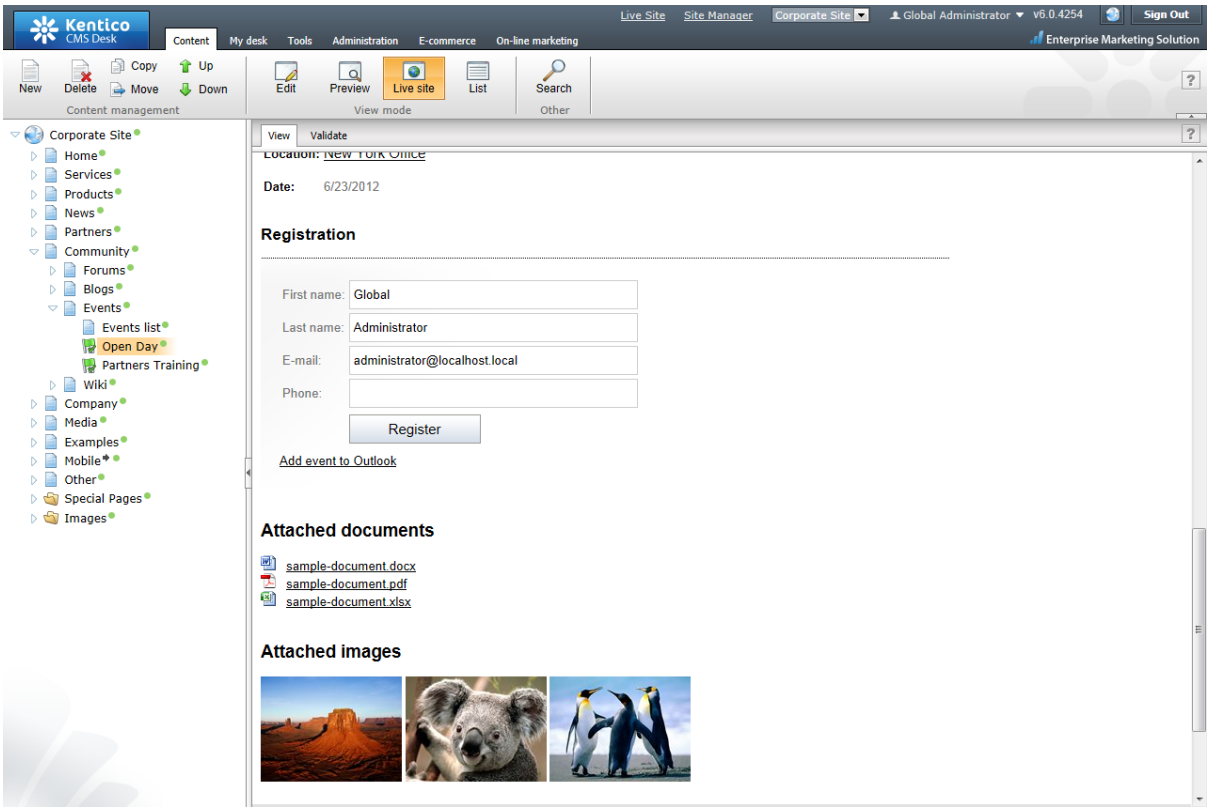
7. Below the documents, we will want the attached images to be displayed. For the images, it will be better to use the **Attachment image gallery** web part. Click the **Add web part** ( ) icon of **zoneLeft** web part zone and choose the **Attachments -> Attachment image gallery** web part. Set the 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.



4.6.2.4 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:

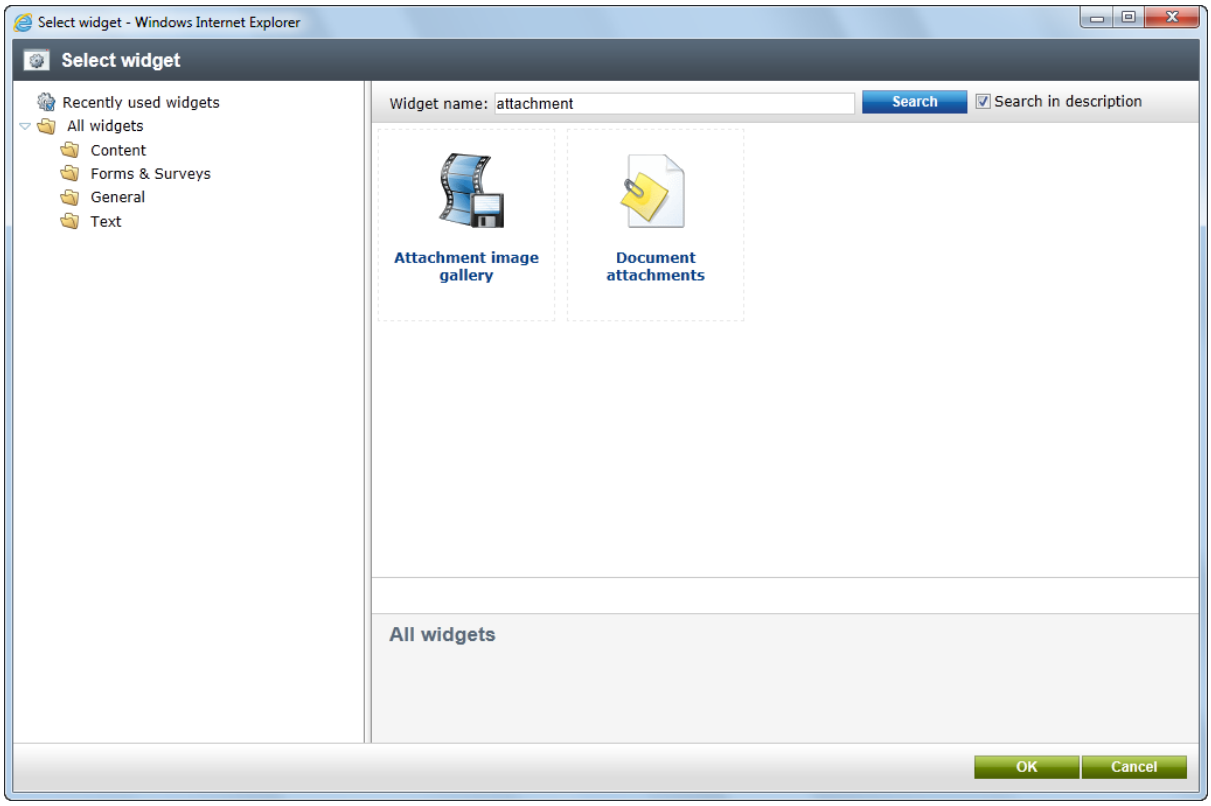
- **Path**: alias path of the document whose attachments should be displayed; if blank, the currently displayed document's attachments will be displayed
- **Attachment group**: specifies the field from which the attachments should be displayed; if blank, ungrouped attachments from the Properties -> Attachments 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.

4.6.2.5 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:



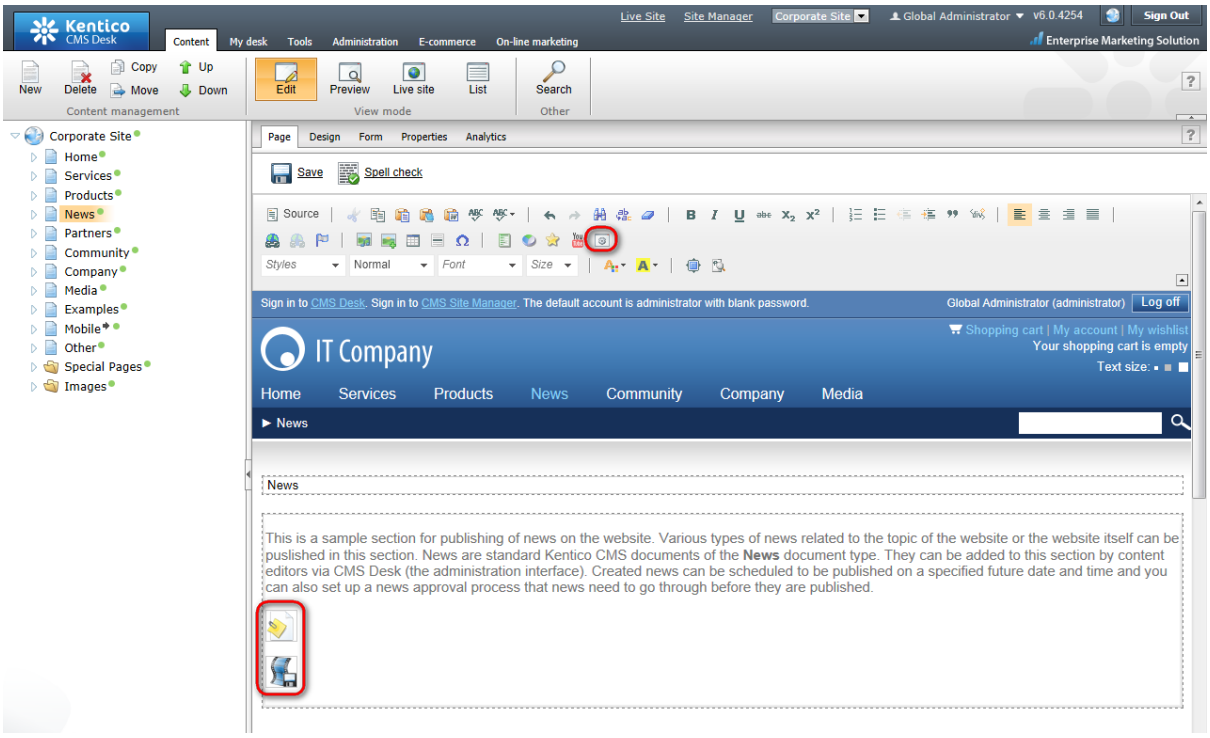
Attachment image gallery

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.



4.6.2.6 Handling attachments in transformations

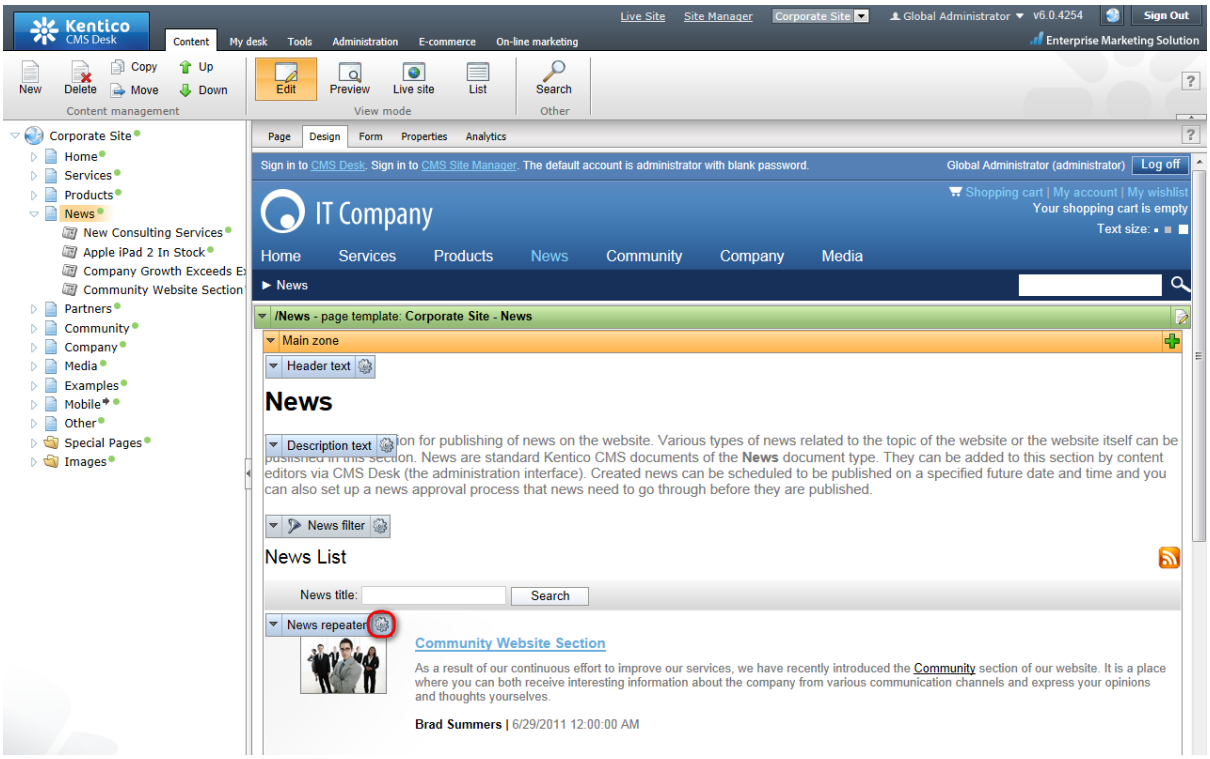
You can display document's unsorted attachments by adding one of the following two controls directly into the transformations:

- **~/CMSAdminControls/Attachments/DocumentAttachments.ascx**
- **~/CMSAdminControls/Attachments/AttachmentLightboxGallery.ascx**

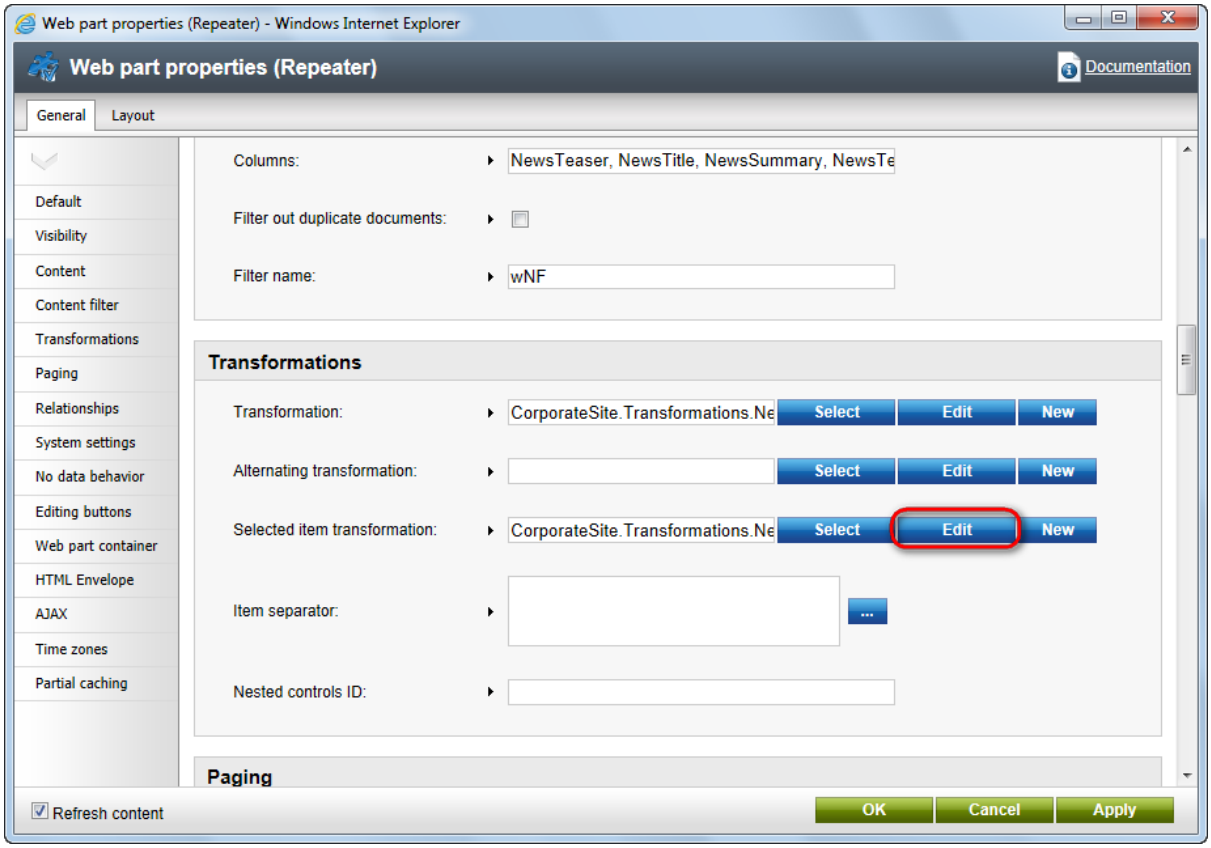
In the following example, you will learn how to use these controls in transformations. The news in the **News** section of the sample **Corporate Site** have some images attached by default. In this example, you have learned how to display these attachments using a web part. Another way of displaying 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 **NewsRepeater** web part.



2. In the web part properties window, choose to **Edit** the **Selected item transformation**.

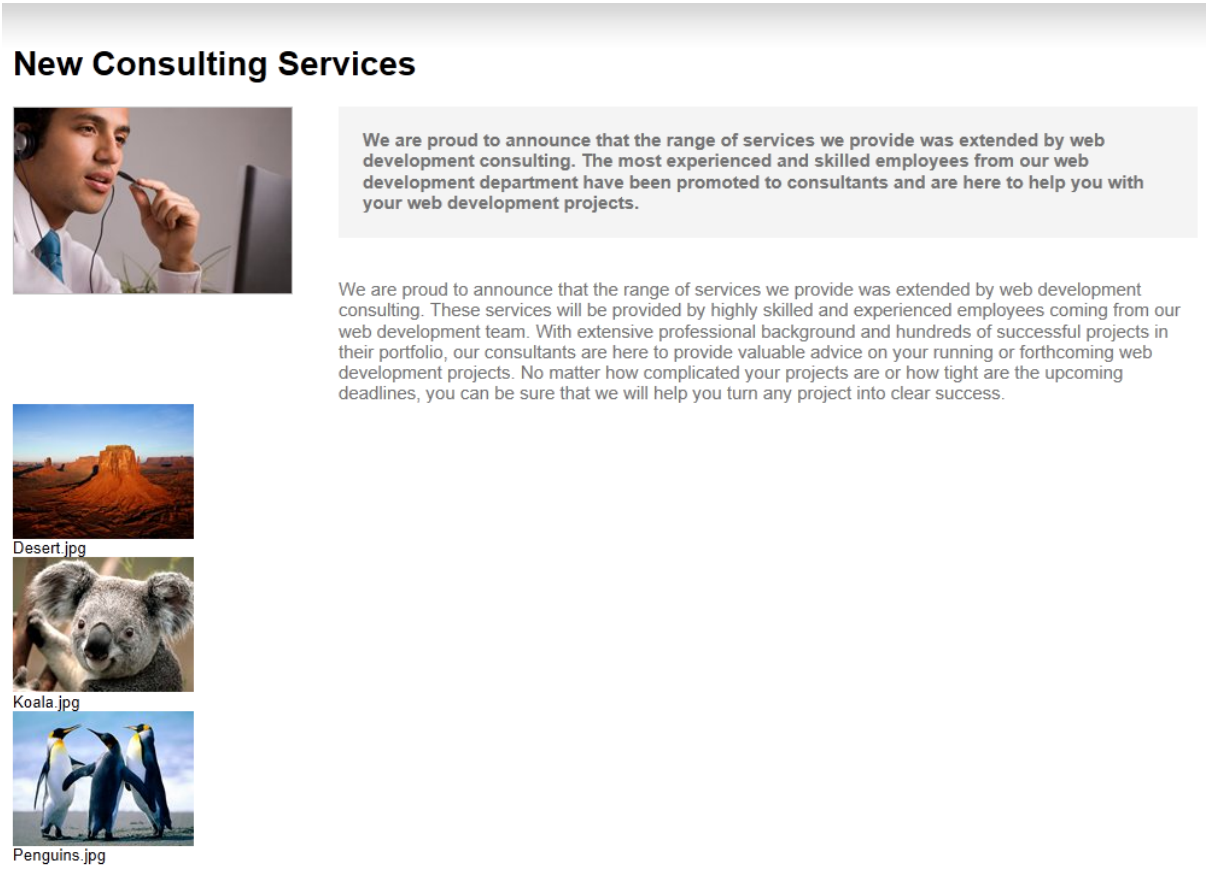


3. Replace the original transformation with the following code, which is the original transformation with the highlighted parts added.

```
<%@ 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.



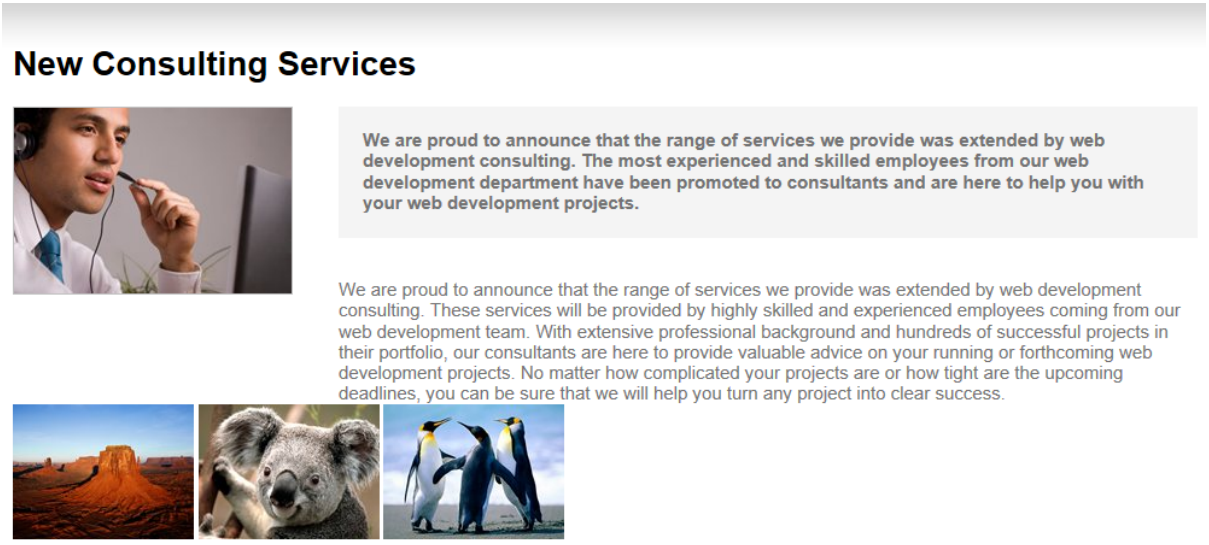
5. Let's try the other control now. Choose to **Configure** ( ) the **NewsRepeater** again and choose to **Edit** its **Selected item transformation**. Replace the transformation with the code below. Again, it is the original transformation with the highlighted parts added.

```
<%@ 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.

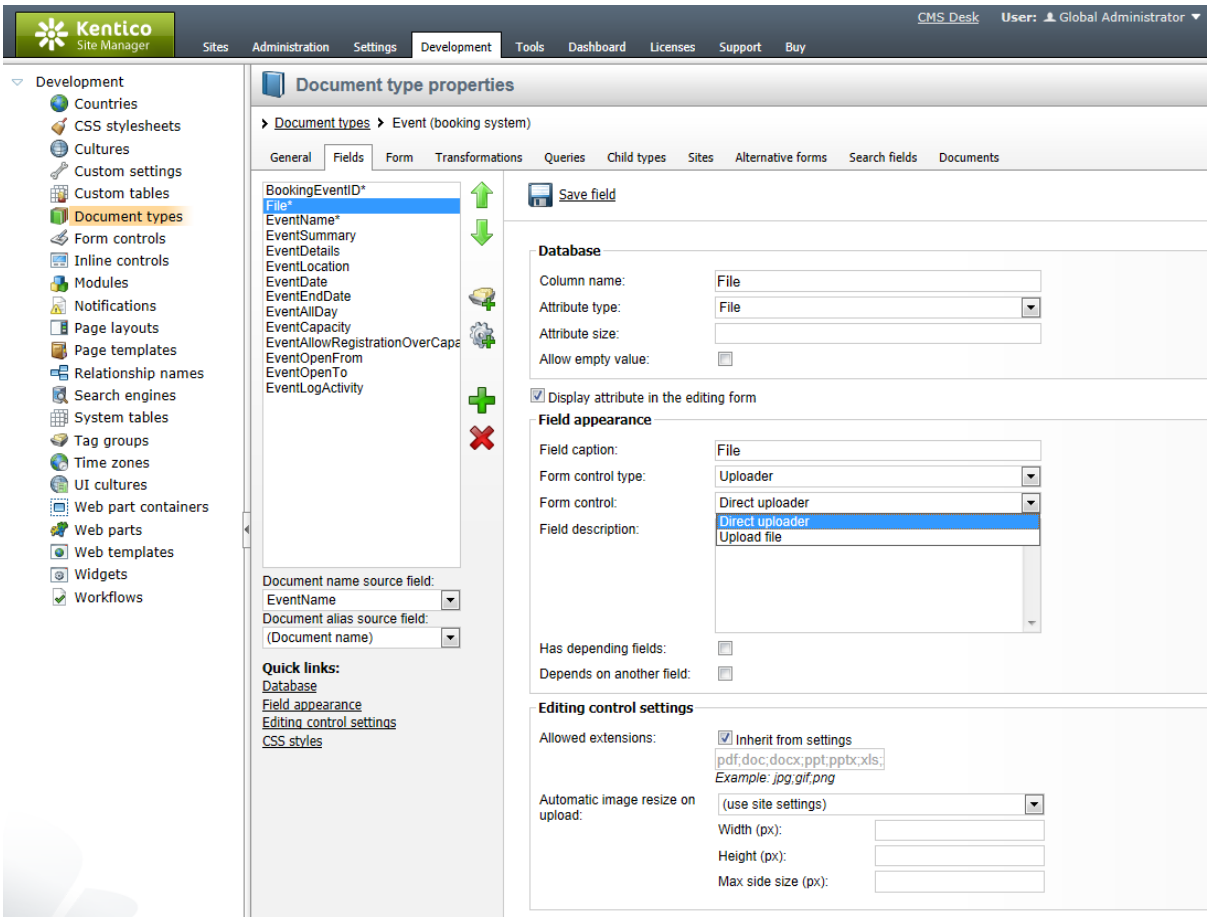


Please note: the appearance shown in this example is based on the controls' default transformations. You can fully customize the appearance by modifying the default transformations or creating your own transformations and specifying them in the **TransformationName** and **SelectedItemTransformationName** properties of the controls.

4.6.2.7 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:

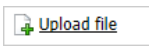
- **Upload file**

  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:



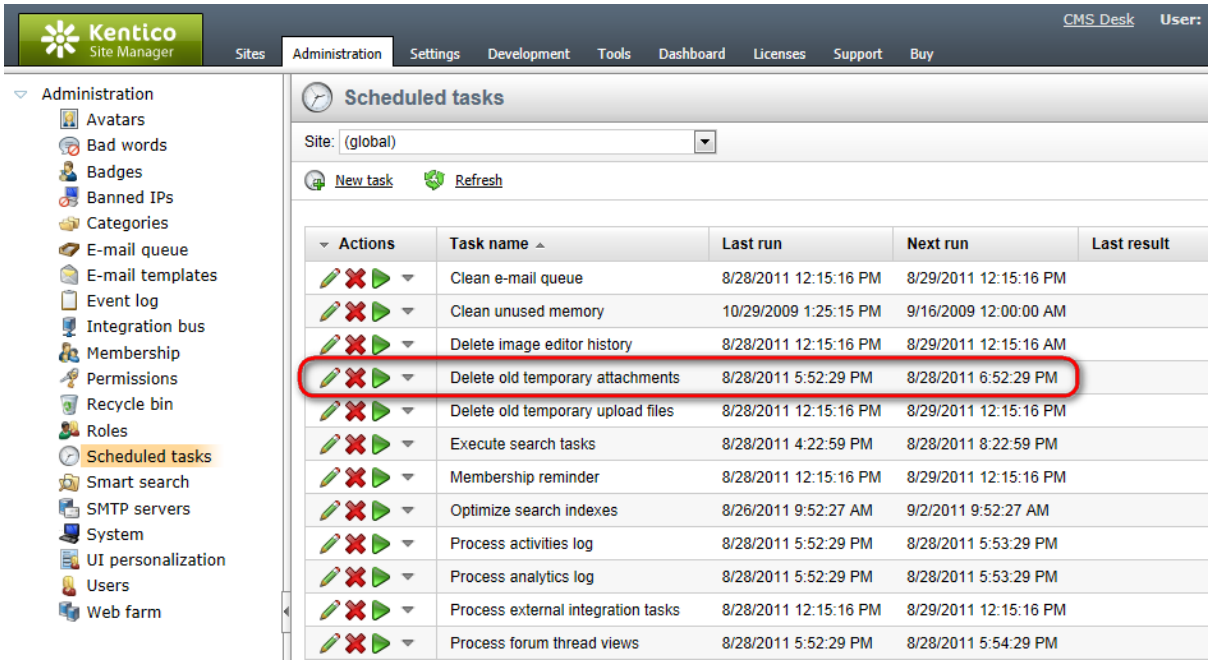
  And here is the control after file upload:

4.6.2.8 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.



4.6.2.9 Attachment names

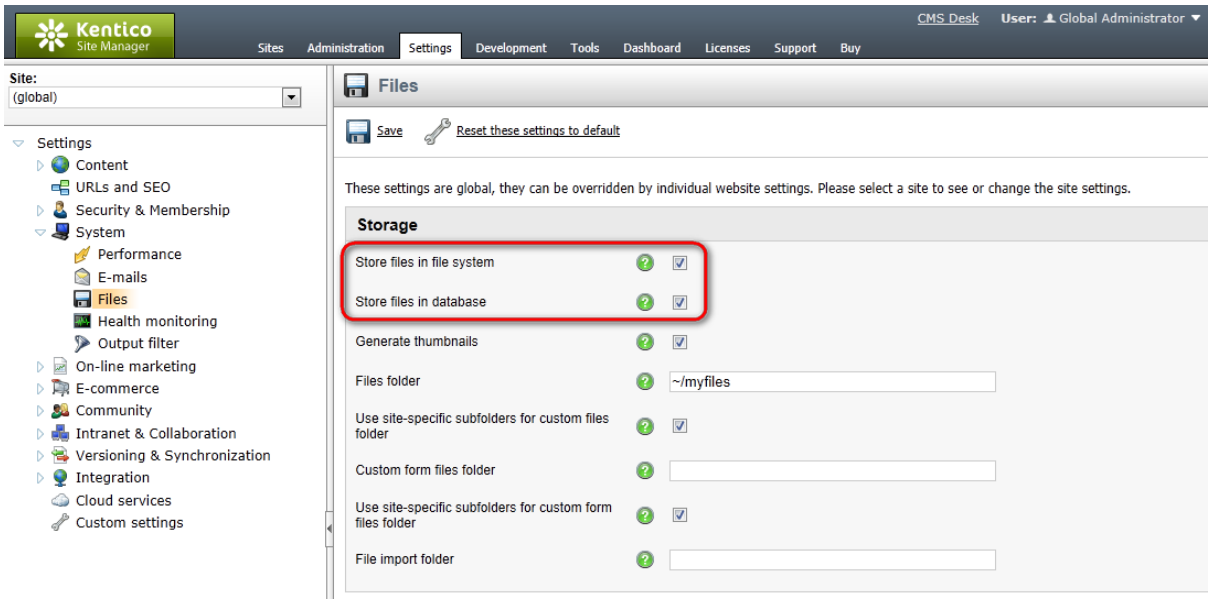
Attachment names are unique for a document (or its version). If you try to upload an attachment that has the same name as some already existing attachment of the particular document, the new attachment gets a number in the **-(1)** format attached to its name.

This happens for the attachments to be accessible via friendly URLs in the following format:

- **~/getattachment/<node_alias_path>/<safe_filename>.<extension>**

4.6.3 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 Files-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 ->**

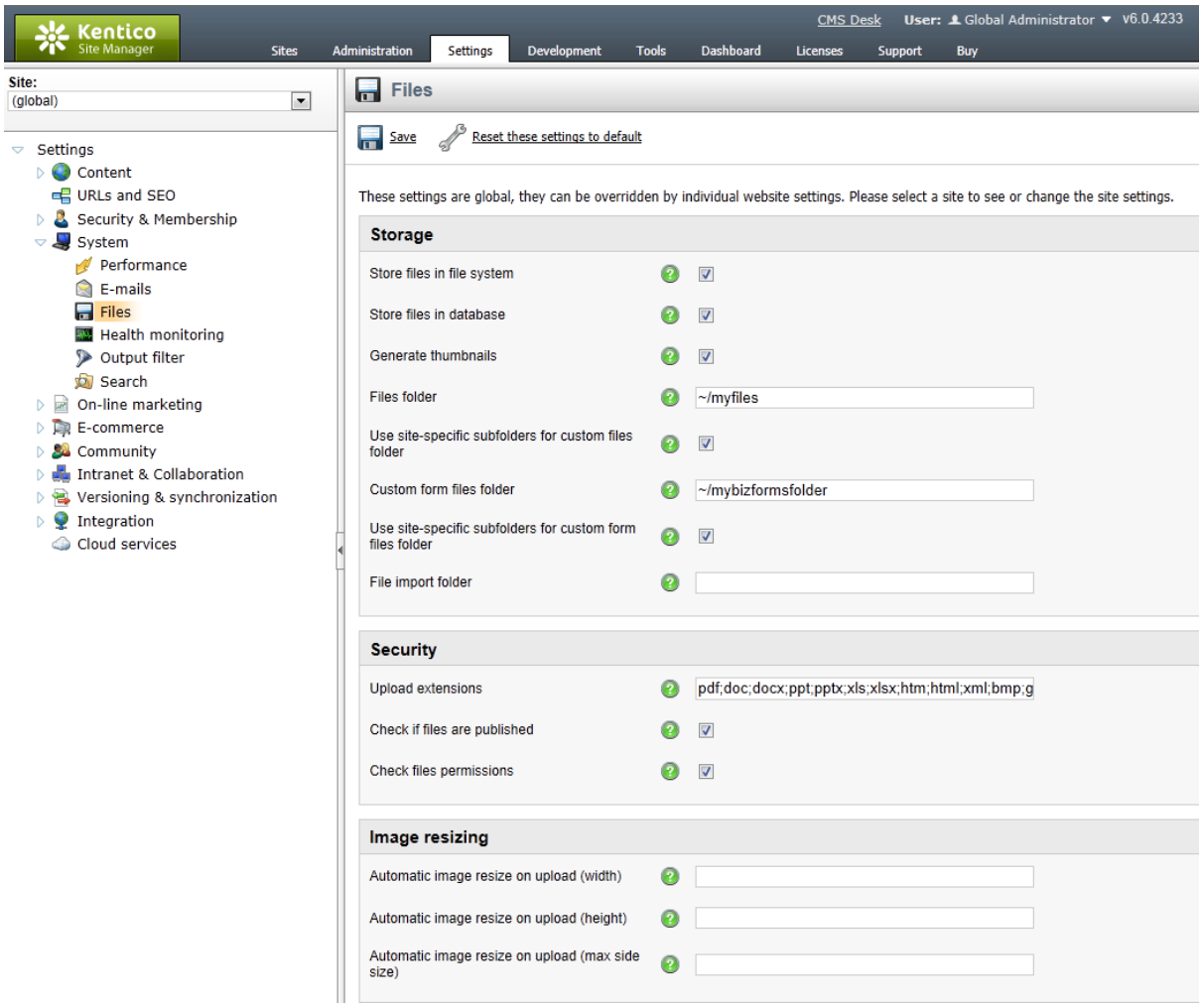
System -> Files -> Form files folder.

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.

4.6.4 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.

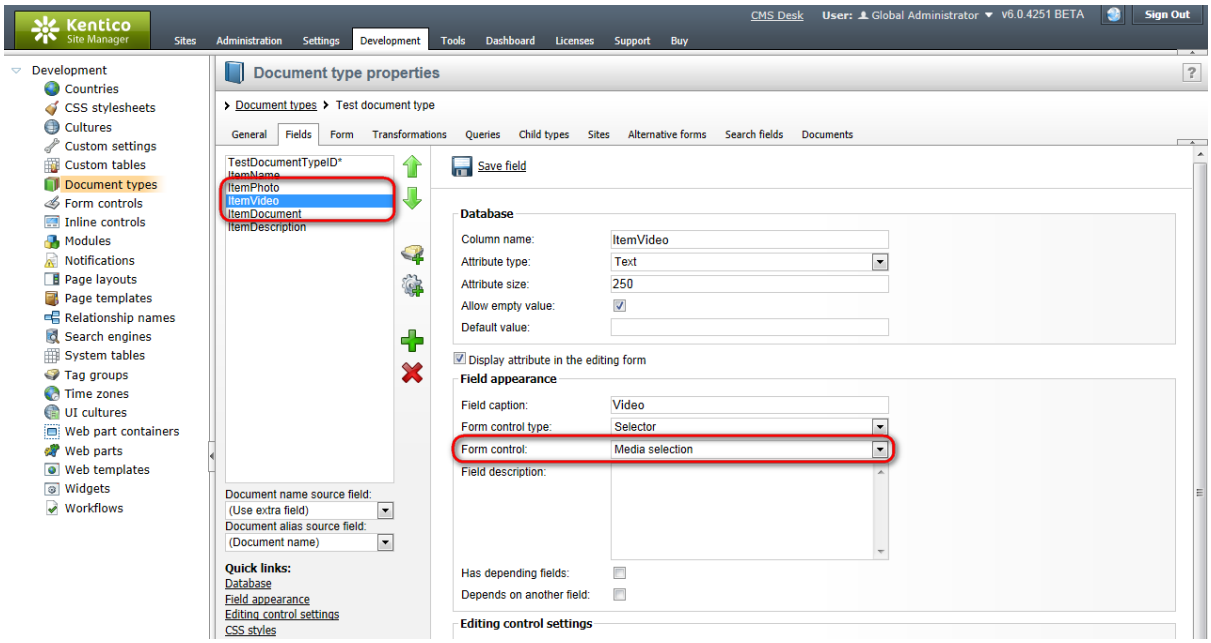
Store files in database	Indicates if files should be stored in the database.
Generate thumbnails	Indicates if the CMS should generate image thumbnails on the disk when a **resized version** of the image is displayed. This option only applies if files are stored in the file system. It **improves site 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 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.

4.6.5 Using the Media selection control

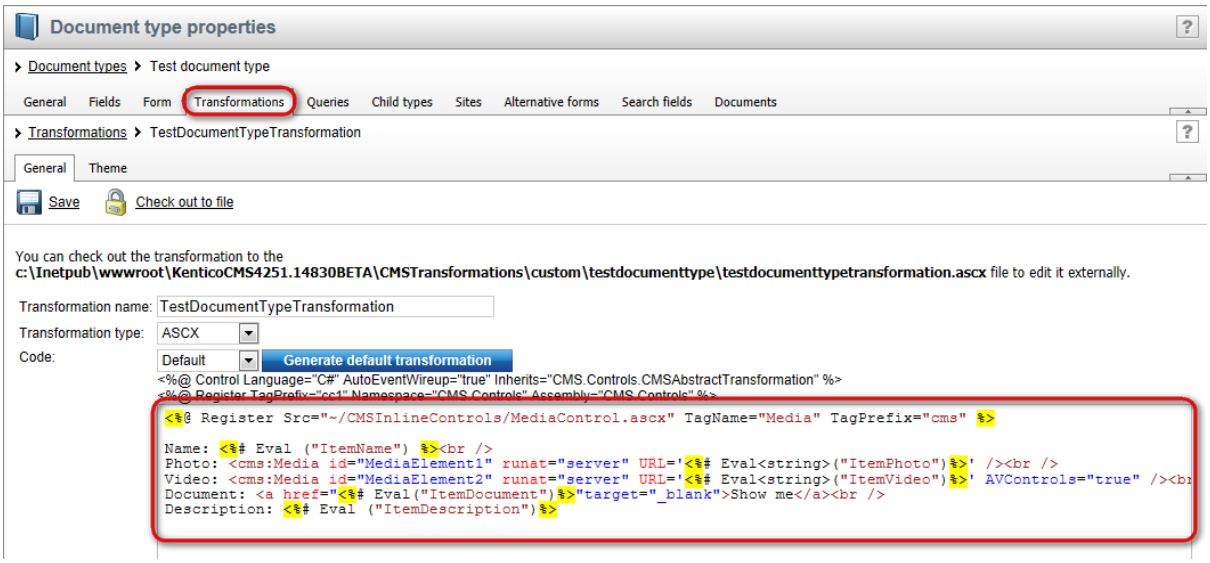
The **Media selection** form control can be used to enable users to select files on the **Form** tab of a document. It allows to use any file from any source (from Document attachments, Content and Media libraries) in your documents. The following example demonstrates how you can achieve this:

1. Navigate to **Site Manager -> Development -> Document types** and create your 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** (the *Selector* **Form control type** must first be chosen):

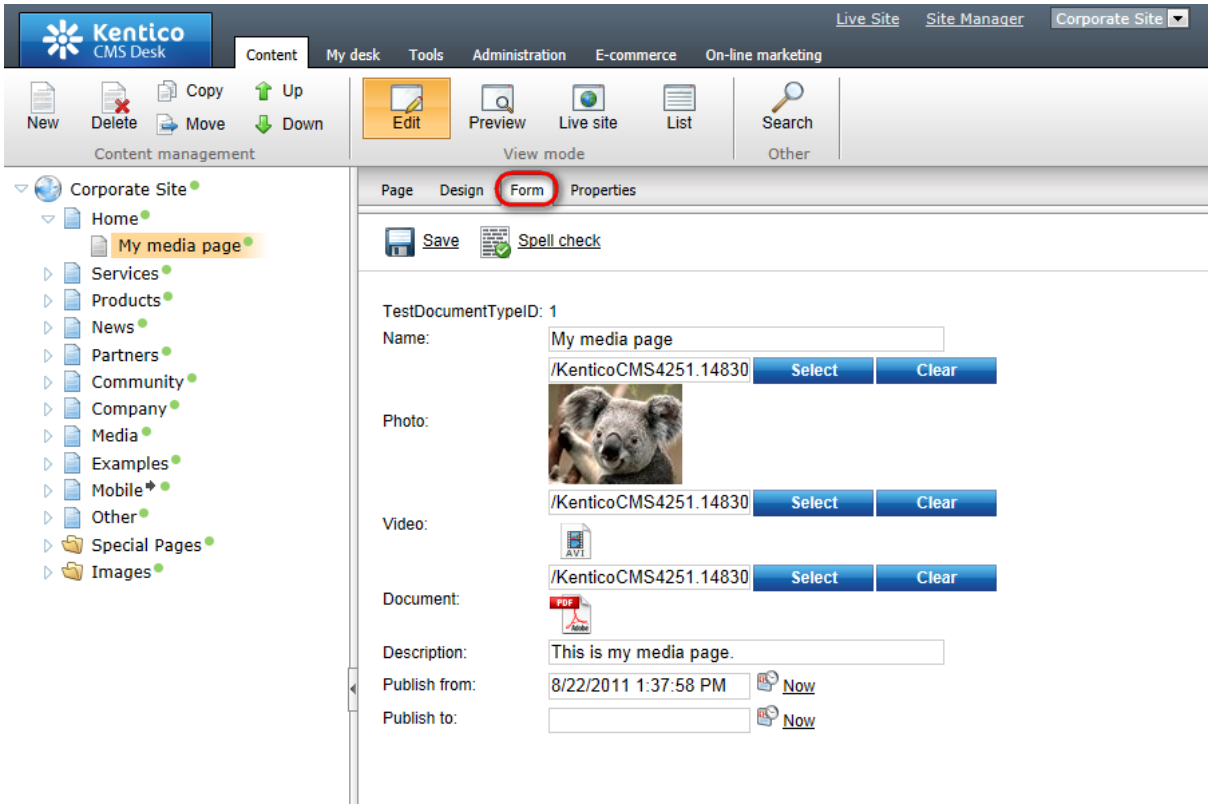


2. In the **Document type properties** user interface, switch to the **Transformations** tab and specify the new transformation as follows:

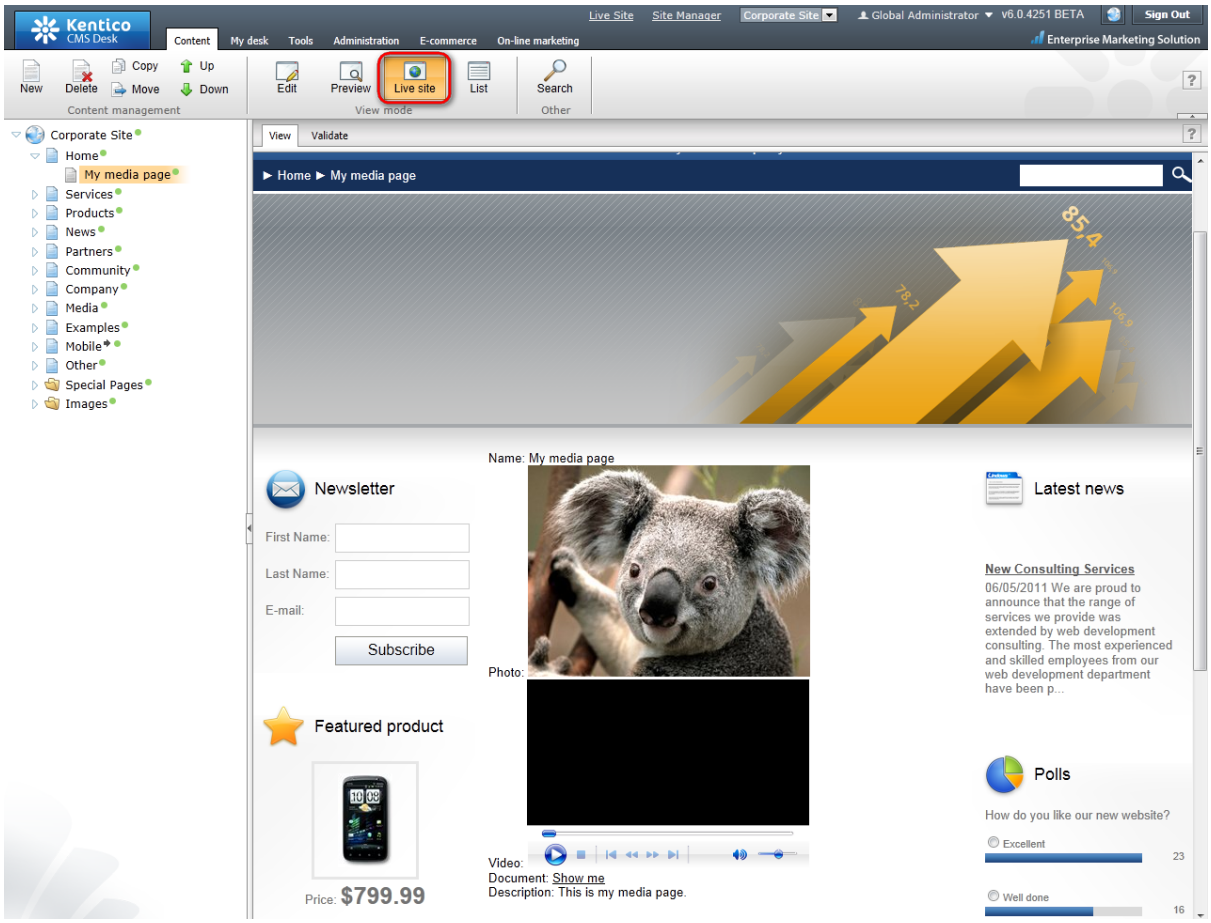
```
<%@ Register Src="~/CMSInlineControls/MediaControl.ascx" TagName="Media"
TagPrefix="cms" %>

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:



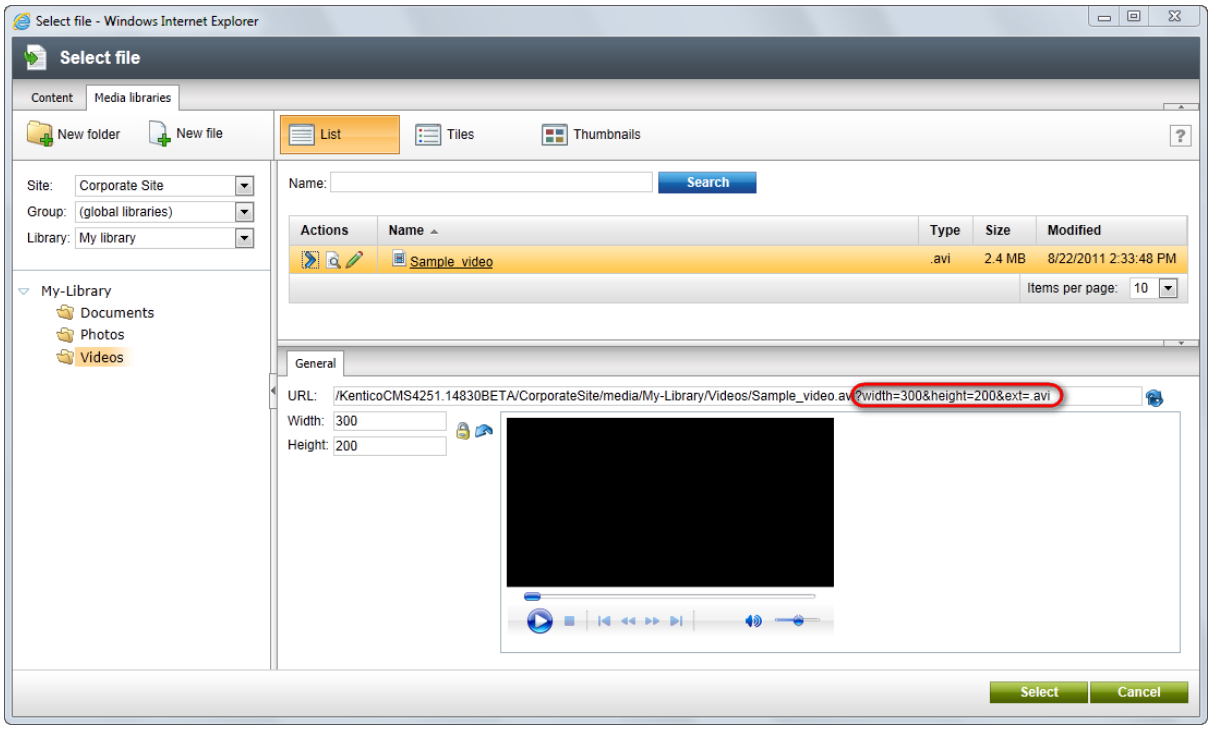
4) Finally, switch to the **Live site** mode to see the results. You will need to use some of the viewers (e. g. the **Repeater** web part) to display data of your new document using your defined transformation:



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:



4.6.6 Image editor

4.6.6.1 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.

4.6.6.2 Image editing

Here you will learn how to edit the selected image. Besides, you will learn how to view the image properties:

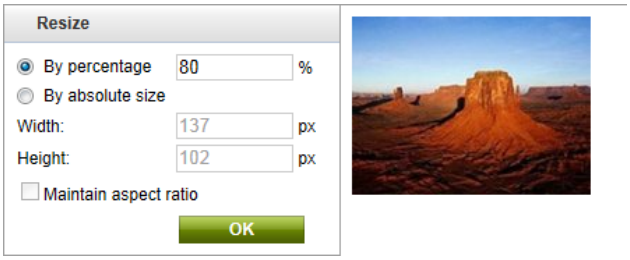
- Resize
- Rotation
- Convert
- Crop
- Color
- Properties

Each action is performed after clicking the **OK** button. Please refer to the Saving changes topic for more

details.

Resize

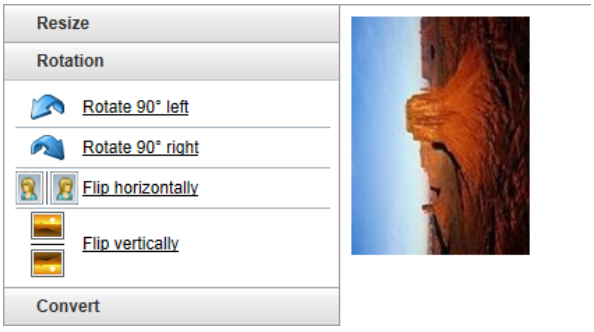
The **Resize** action lets you change the size of the image. The image is either enlarged or reduced to the size you specify. You may specify either a relative percentage to resize the image by, or an absolute pixel size. If the **Maintain aspect ratio** option is checked, then typing a new **Width** 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.

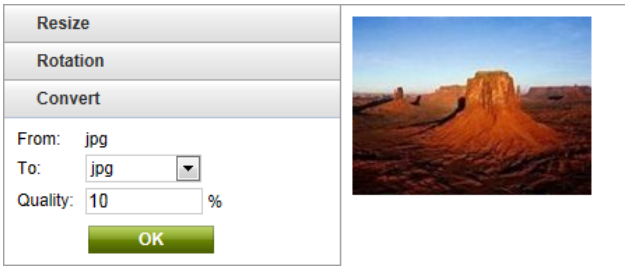


- **Rotate 90° left** - rotates the image to the left by 90 degrees.
- **Rotate 90° right** - rotates the image to the right by 90 degrees.
- **Flip horizontally** - flips the image horizontally.
- **Flip vertically** - flips the image vertically.

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

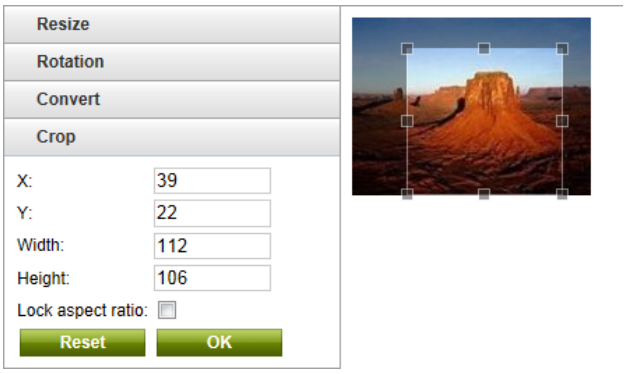
image quality percentage into the **Quality** field.



- **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.

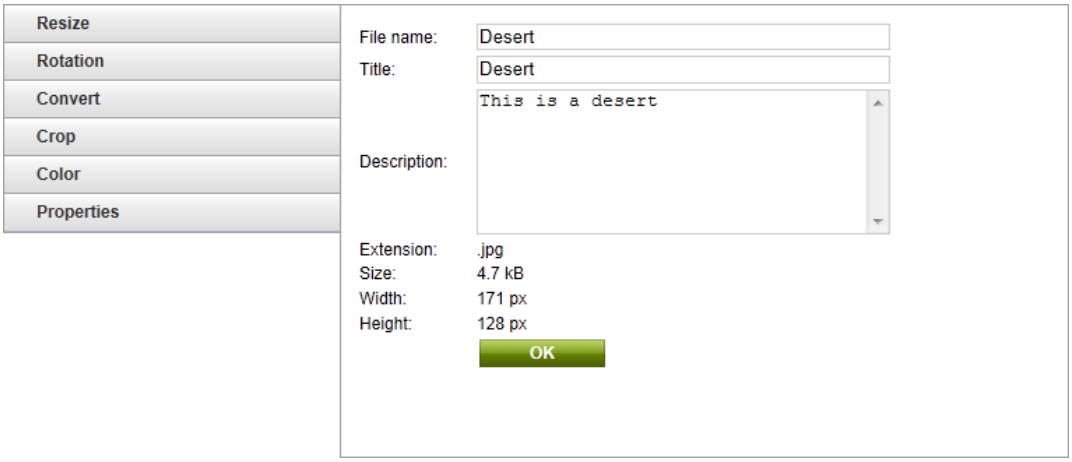
Color

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.

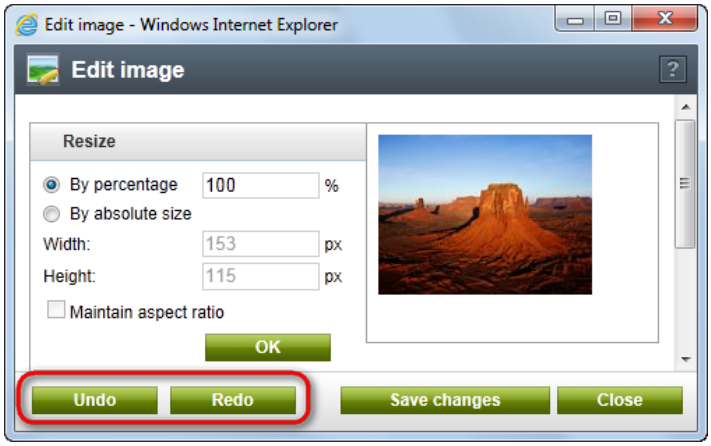
Please note

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.

4.6.6.3 Saving changes

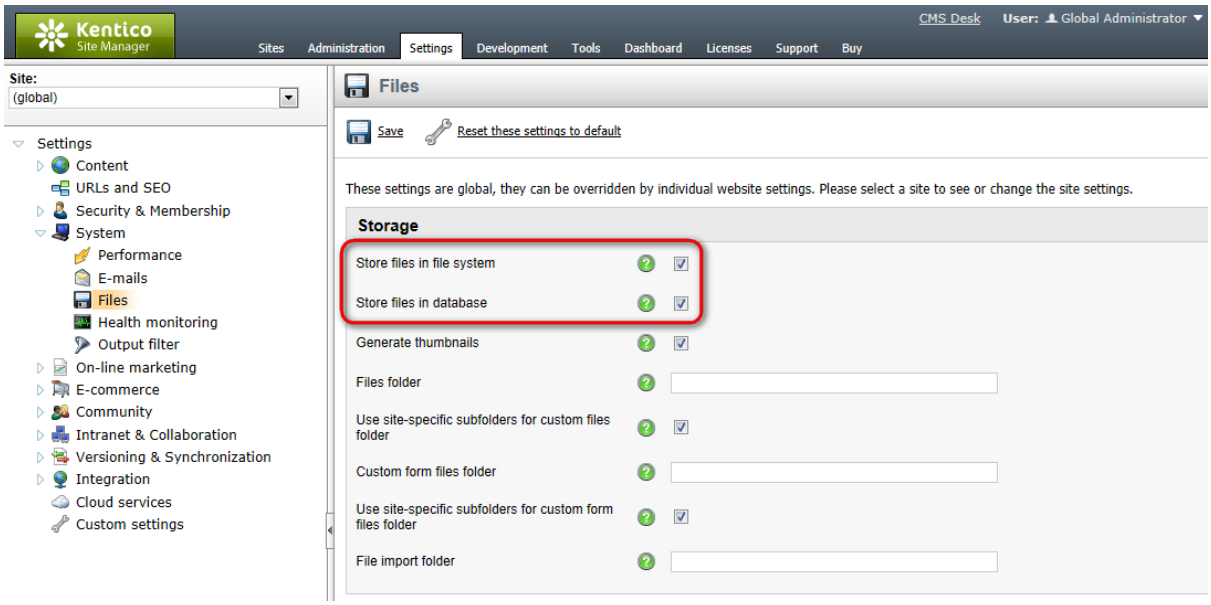
Undo and Redo

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.

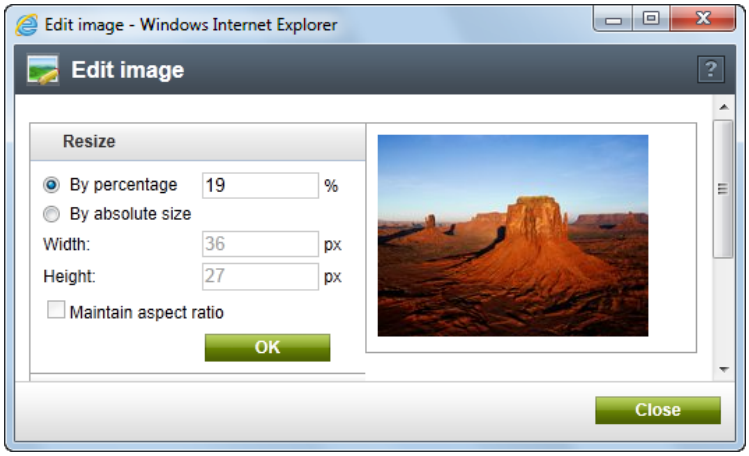


Storing image versions

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**.

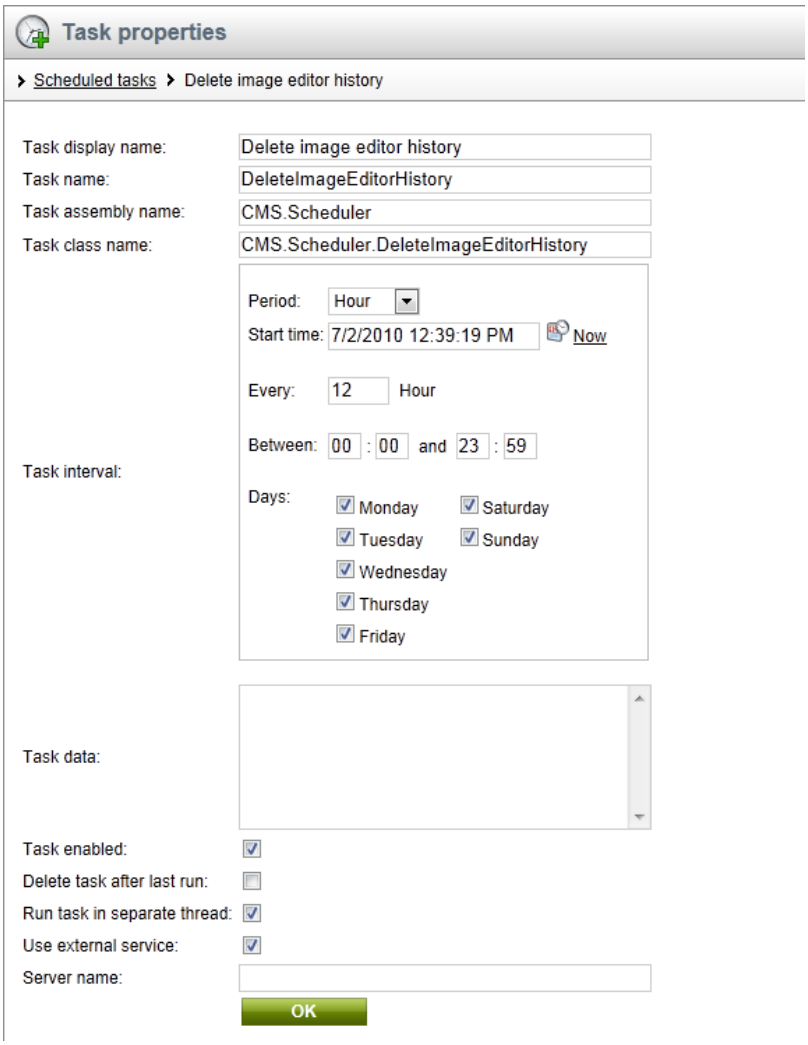


Please note that if only the storing of files in the file system is enabled and permission to write to the local disk is missing, the **Undo**, **Redo** and **Save changes** buttons are hidden and the changes are applied immediately to the original image.



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.

4.6.7 Metadata editor

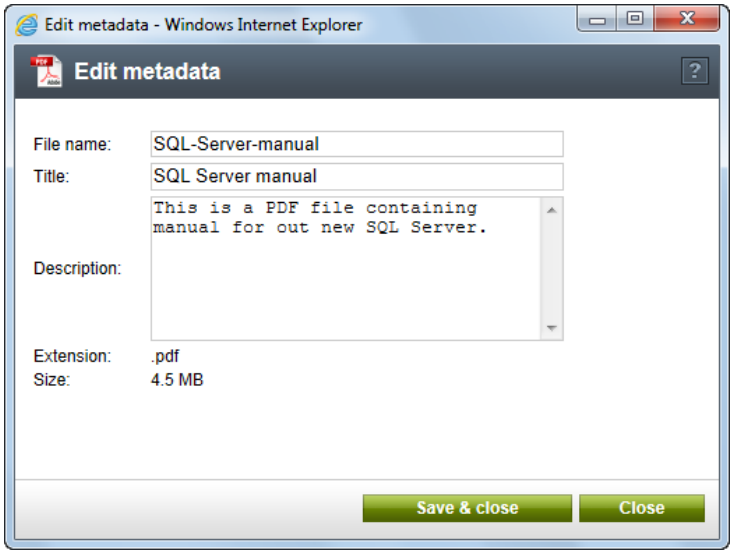
The Metadata editor can be used to edit metadata (i.e. descriptions) of non-image files stored in Media libraries, as document attachments or as object metafiles. The dialog is always accessible by clicking the **Edit** ( ) icon next to the file listed in the respective part of the UI.

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.

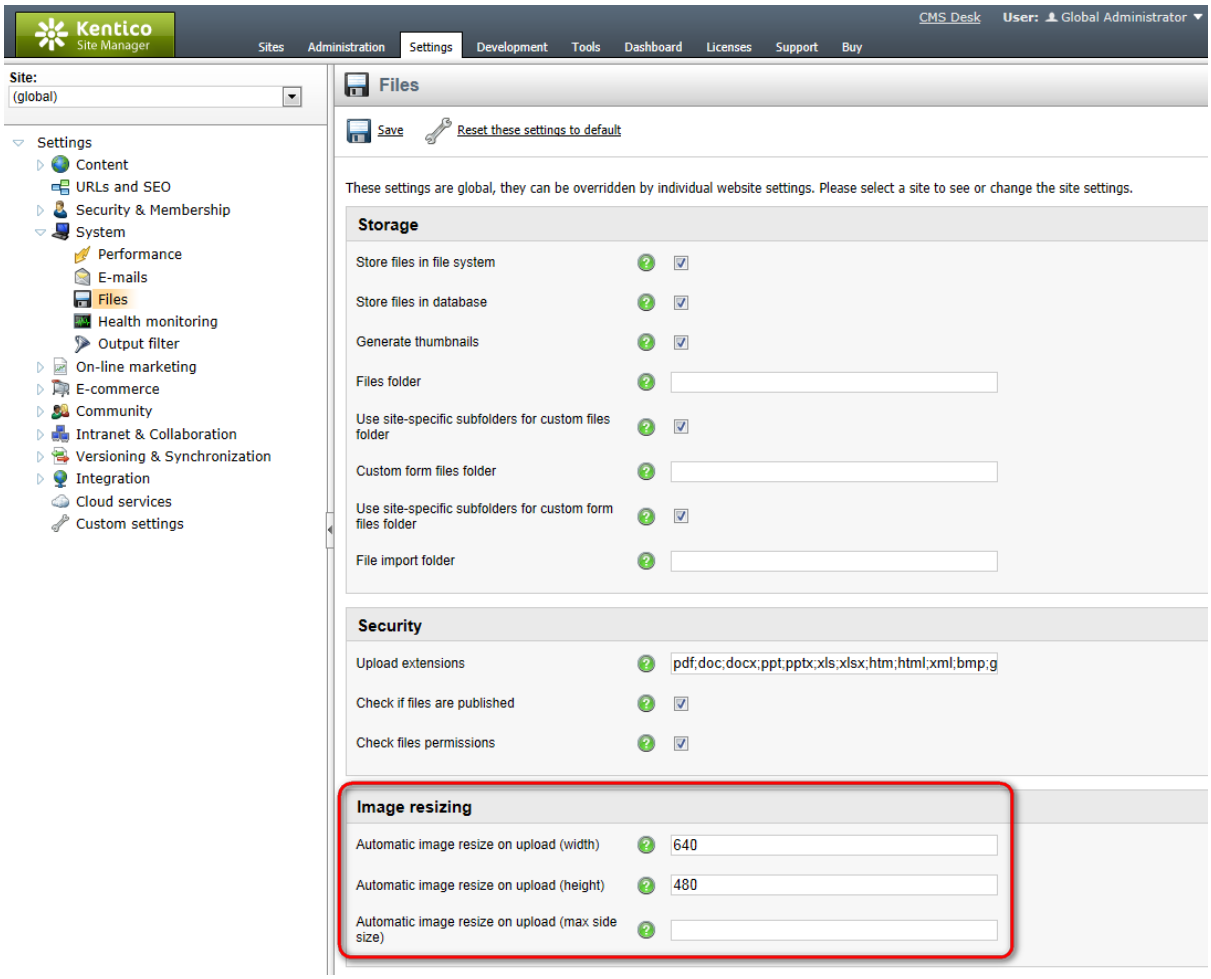
- **Title** - title of the file (may be different than the actual file name).
- **Description** - text describing the content of the file.



4.6.8 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, e.g. in forms, Document type field editor, etc.
  - **File** attribute type -> **Upload file** and **Direct uploader** field types
  - **Text** attribute type -> **HTML Area (Formatted Fext)**, **BBcode editor**, **Image selection**, **Media selection** and **File selection** field types

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.).

4.6.9 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:

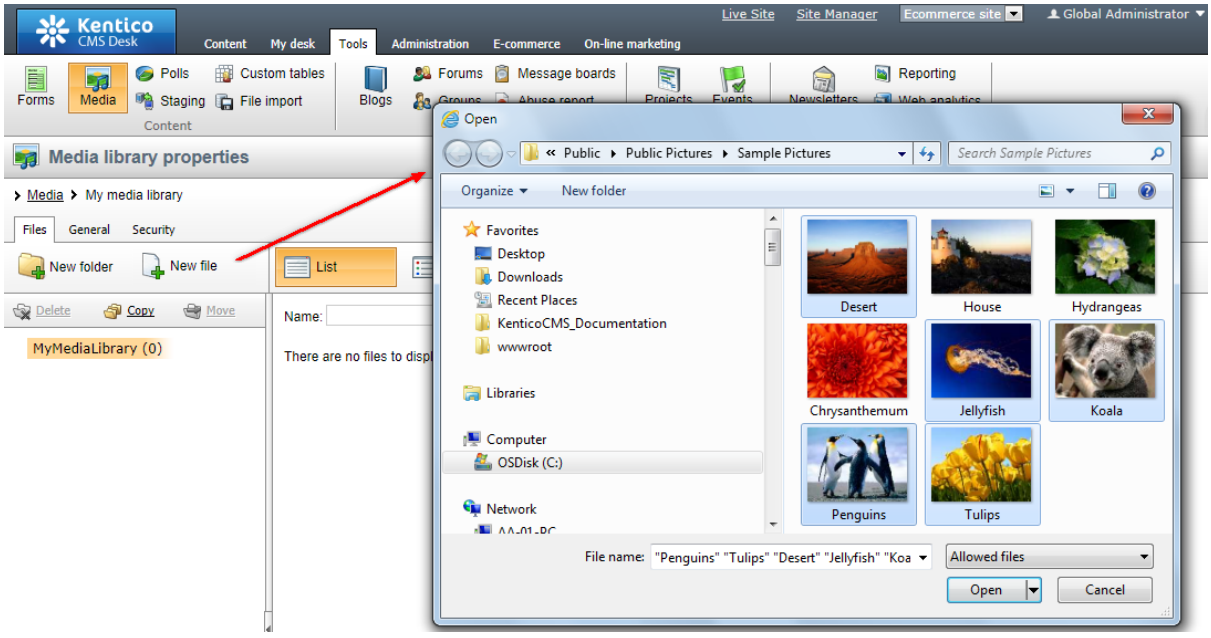
1. The value of the ***CMSUseSilverlightUploader*** key is set to TRUE or the key is not added in the *appSettings* section of the *web.config* file (because the upload of multiple files is the default option).

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.

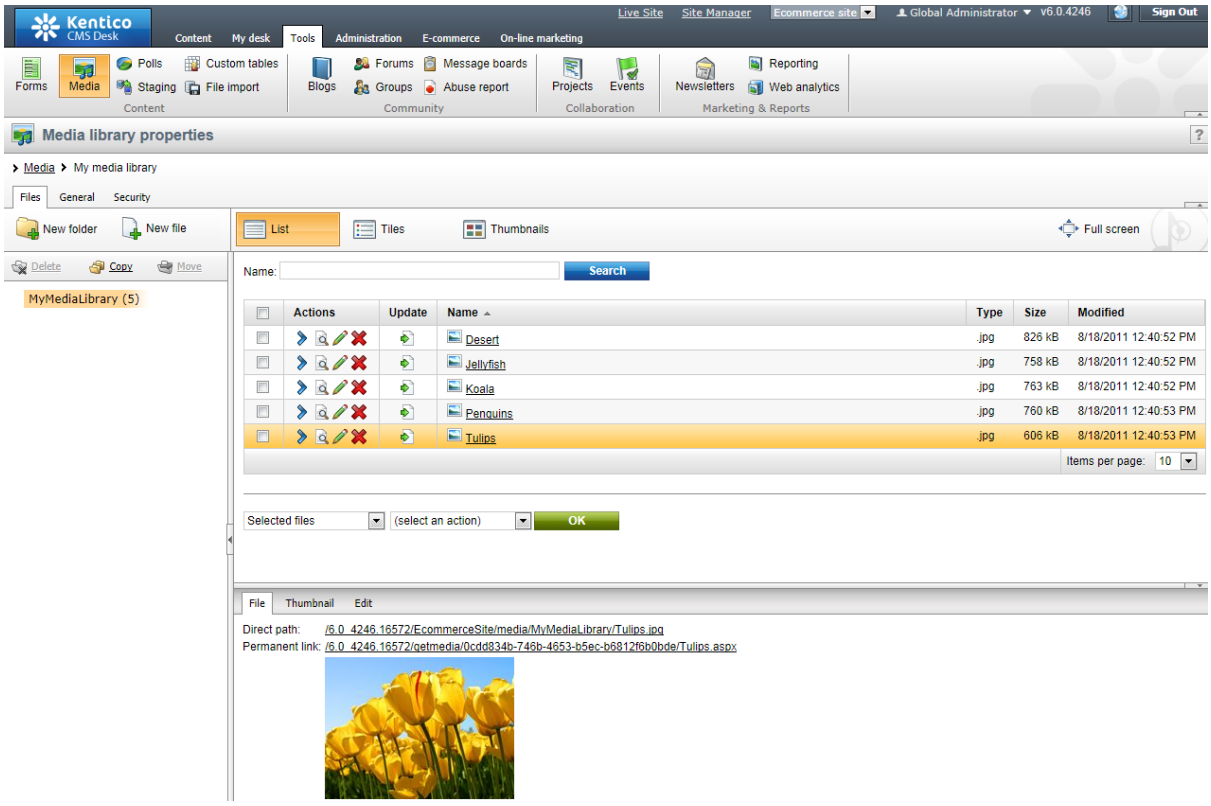
How it works

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).

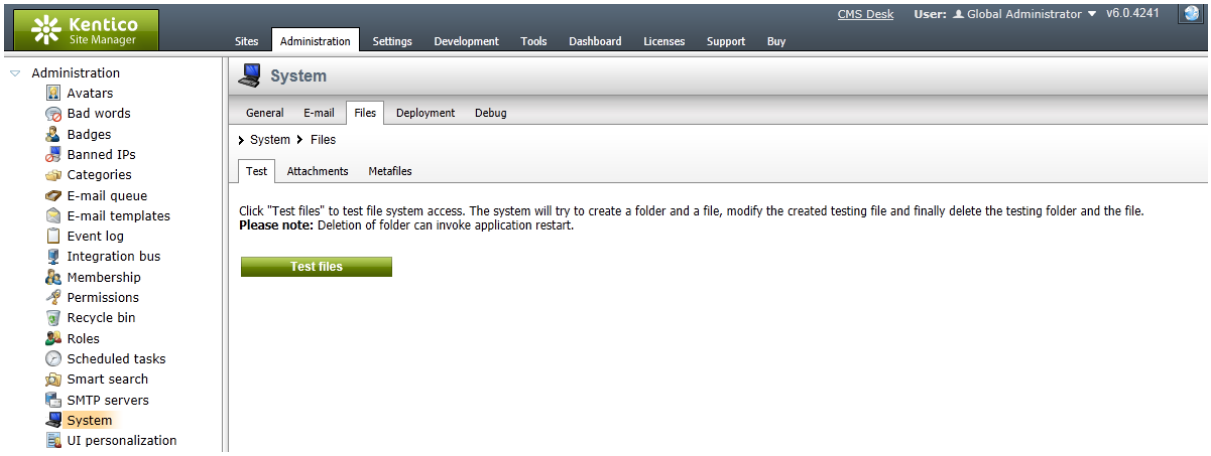
4.6.10 Files administration UI

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:

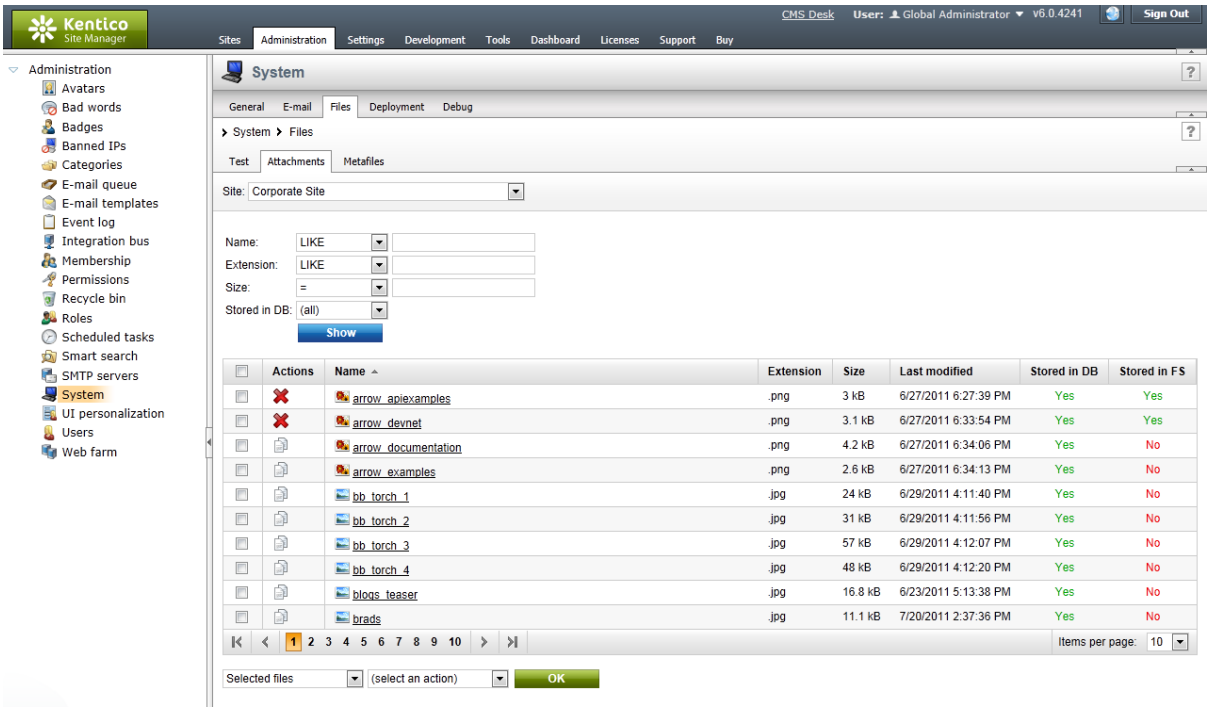
- **Copy to database** - copies the attachment to the database. This action is only displayed if the attachment is stored only in the file system while settings are configured for files to be stored both in the file system and in the database.
- **Delete from database** - deletes the attachment from 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:

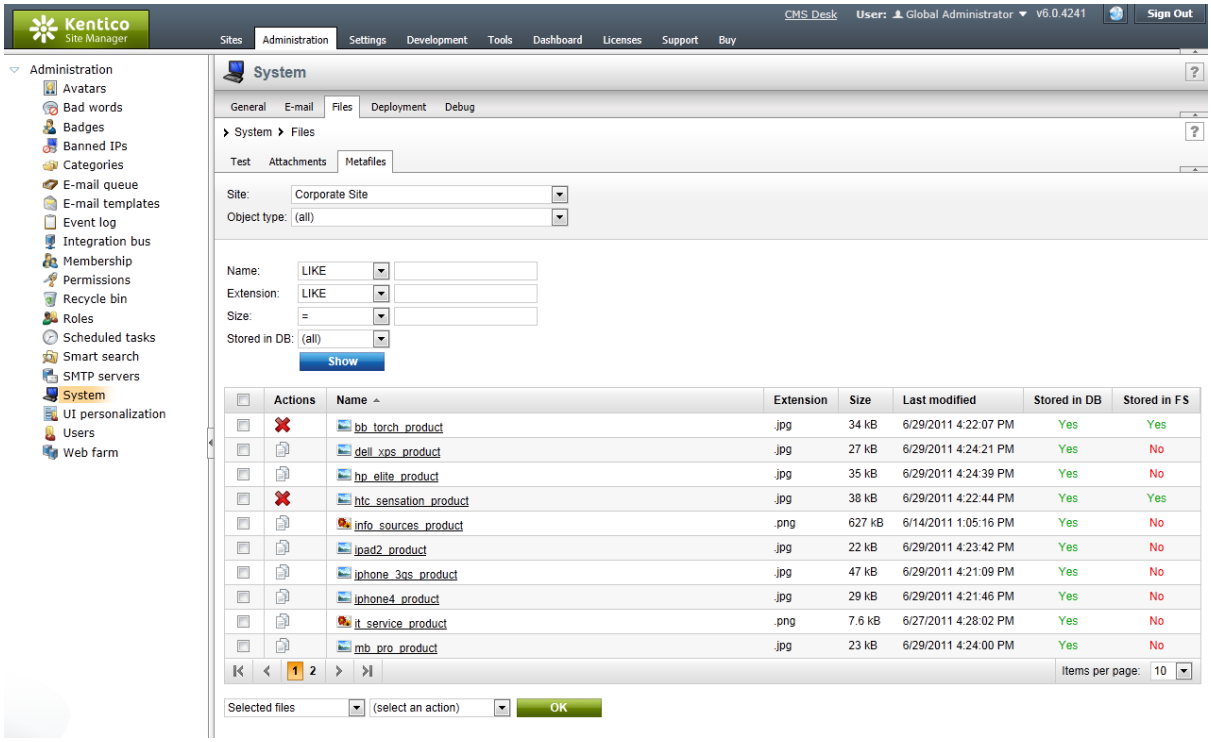
- **Copy to database** - copies the metafile to the database. This action is only displayed if the metafile is stored only in the file system while settings are configured for files to be stored both in the file system and in the database.
- **Delete from database** - deletes the metafile from the database. 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.



4.7 Multilingual content

4.7.1 Overview

Kentico CMS websites can have their content translated into multiple languages. In a multilingual website, there are separate versions of documents for each language. Individual cultural versions of the website may be displayed to users automatically based on various settings, while users can also switch between individual cultural versions of a website manually using a dedicated web part.

The list of all content cultures available in Kentico CMS can be found in **Site Manager -> Development -> Cultures**. All major cultures are provided in the list by default. These cultures can be assigned to websites, whose content can then be translated to the respective cultures. More information on configuring a website to use multiple content cultures can be found in the Configuring multilingual content topic.

In the Displayed language selection topic, you can learn about configuration that determines which one of the available cultures will be displayed to registered users and website visitors when they access the website. To learn about URLs under which individual document language versions are accessible and with which they are rendered, please refer to the Languages and URLs topic. In the Translation management sub-chapter, you can find descriptions of various features related to website content localization.

The user interface of Kentico CMS can also be multilingual. To learn about localization of Kentico CMS UI, please refer to the Development -> UI cultures and localization chapter of this guide.

4.7.2 Configuring multilingual content

This chapter explains how to configure a website to display content in multiple languages.

Default content culture

When you create a new website, its default culture is the culture you specify in Step 2 of the New site wizard. If you create a website based on a web template, it uses the culture of the 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**

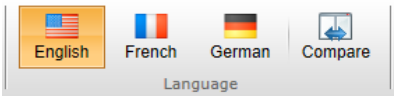
The two settings are interlinked, i.e. values configured at one section are reflected in the other one.

Configuring multilingual content

In the following example, you can learn how to add extra language versions to the sample Corporate Site:

1. Go to **Site Manager -> Sites** and click the **Edit site** (✎) icon next to the **Corporate Site**. Go to the **Cultures** tab. Click the **Add cultures** button and add the following cultures: **French - France** and **German - Germany**.

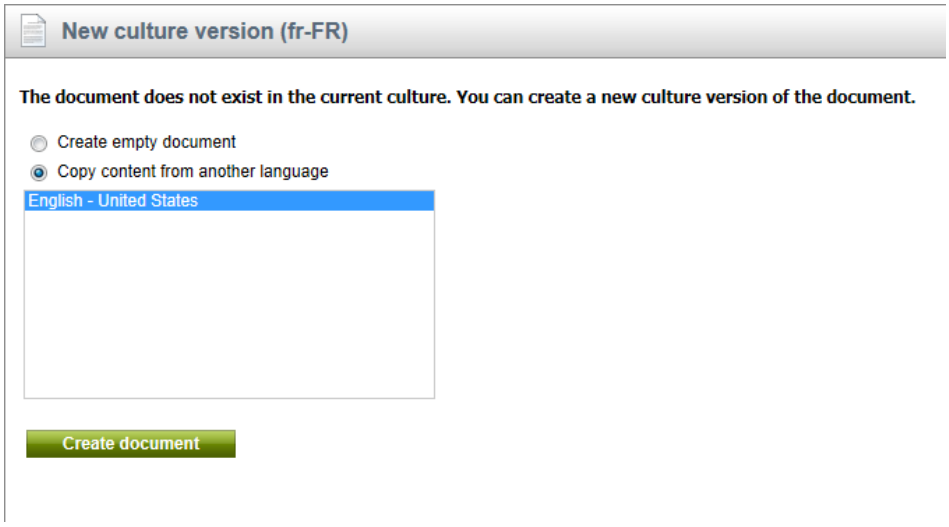
2. Switch to CMS Desk. You can see there's a new section with language selection in the main toolbar. Using this toolbar, you can switch between particular language versions of the currently selected document. You can also click the **Compare** ( ) button to switch to Language versions comparison mode and edit language versions side-by-side.



3. Click the root document and choose the **French** culture. Since the French version of the document doesn't exist yet, you're offered with two options:

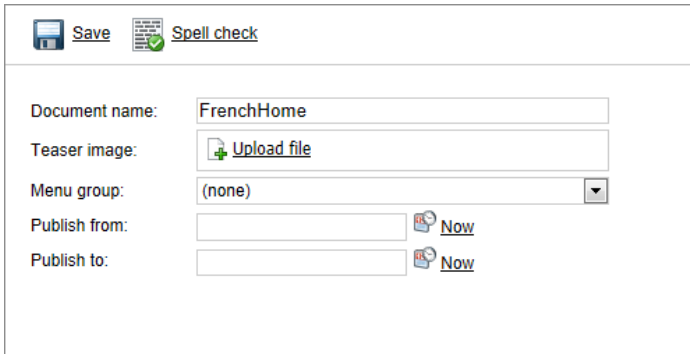
- **Create empty document** - creates a new document of the same type without any content.
- **Copy content from another language** - creates a new document of the same type and copies its content from another language version.

Choose to copy content from the English version and click **Create document**:

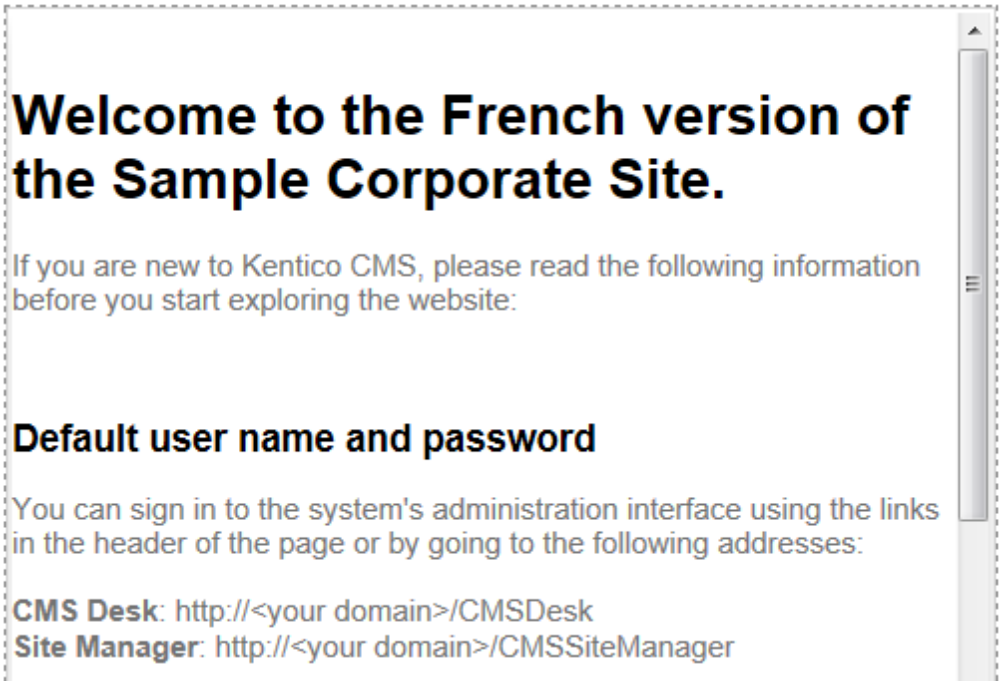


4. You are redirected to the editing form of the new version. The root cannot have any name, so the box is now disabled. Click **Save**. The new document culture version is created.

5. Now repeat the same for the **Home** page. On the editing form, change the document name to **French Home**. Click **Save**.



6. Change the heading text to "Welcome to the French version of the Sample Corporate Site". Click **Save**.



7. Sign out. Now, you can see the live site in French version:

8. 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** web part.

9. Now sign in as administrator to **Site Manager** and go to **Settings -> Content**. Choose the Corporate Site in the drop-down list and set the value **Combine with default culture** to **true**. Click **Save**. Sign out and see the French version of the live website now. As you can see, the documents that are not translated to French are now displayed in the default culture (English).

Creating content for another language version

Every time you create content for another language version, please be sure you create language versions for at least the **root** document and the **home page**. Otherwise, the website will not be displayed correctly.

The documents that are not translated yet are displayed with a cross (✖) icon on the right so that you know which of them need to be translated.

4.7.3 Displayed language selection

This topic explains configuration that affects which language version is displayed to users when they access a website.

Configuration for registered users

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).

Configuration for anonymous visitors

It is also possible to configure the 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 ( ) -> General -> Default visitor culture** - used for visitors accessing the website through a URL using the main domain.
- **Site Manager -> Sites -> edit ( ) -> Domain Aliases -> edit ( ) -> Default visitor culture** - used for visitors accessing the website through a URL using 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** will be used (in Internet Explorer, you can set the default language in Tools -> Internet Options -> Languages).

Domain-based language selection for anonymous visitors

Using the settings for anonymous visitors explained above, you can configure the system to use different default languages 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 comes first to e.g. *example.fr*, website content is automatically displayed in French.

4.7.4 Languages and URLs

When you're using multilingual content, all language versions of the given document use the same URL (based on the alias path) by default. E.g. the home page always has this URL: */Home.aspx*

If you want to see the same page in French, you need to go to: */Home.aspx?lang=fr-fr*

Once the language is changed, the selected language is stored in the visitor's browser using a cookie and when the user comes back to URL */Home.aspx* (without any query parameters), the French version will still be displayed to them.

Using language prefixes for URLs

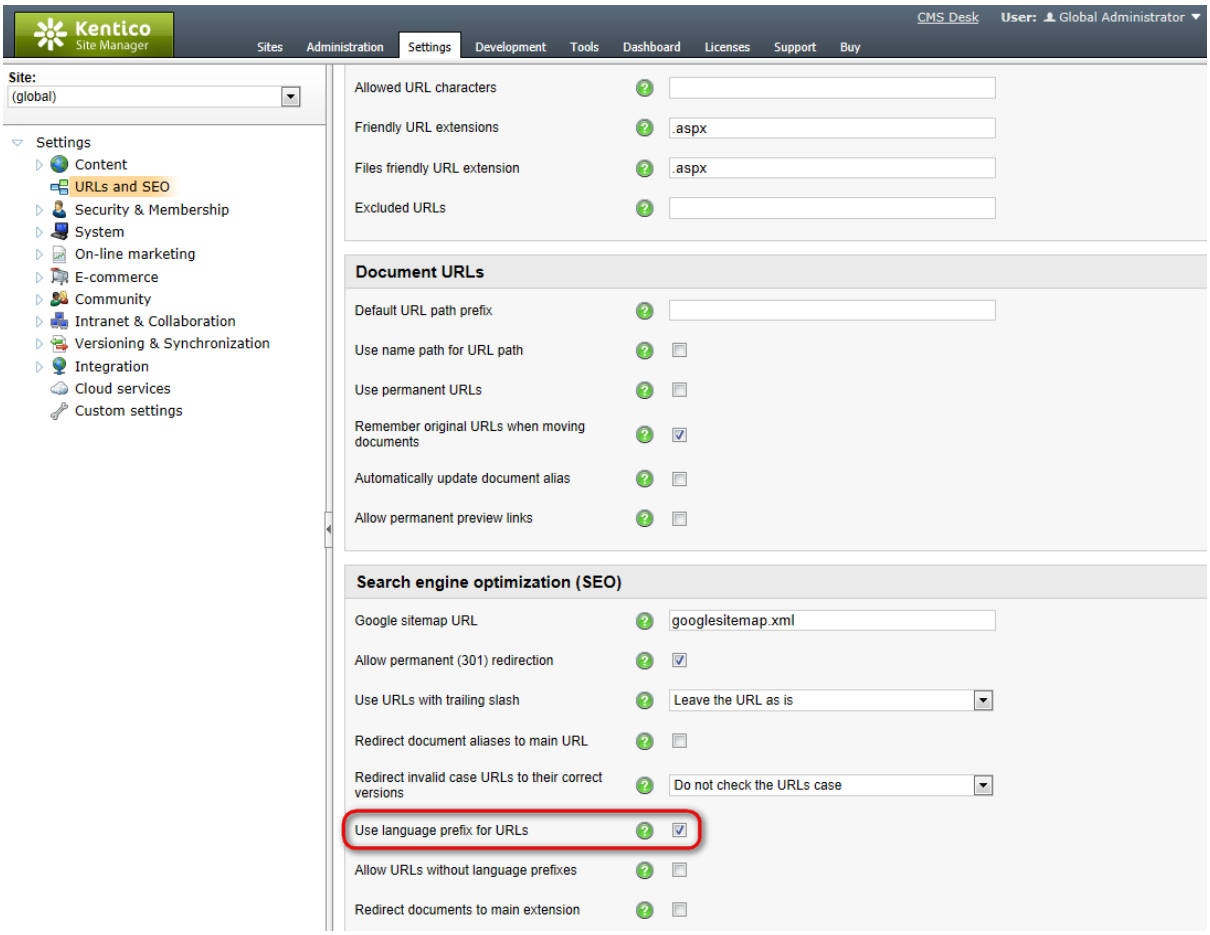
If you want to avoid having to use query string parameters such as *<domain>/Home.aspx?lang=fr-fr* to select languages in URLs, you can use language prefixes instead. The format of language prefixes is <domain>/<languageprefix>/<remaining url>, such as for example:
<domain>/fr-FR/Home.aspx

The language prefix matches the culture code of the requested language. If a culture alias is set for the language, it takes precedence and is used instead of the culture code. If you want to change the language prefix for a certain culture, it is best to set a culture alias.

For example, if you wish to change the language prefix for the French language to *France*, go to **Site**

Manager -> Development -> Cultures, enter *fr-fr* into the **Culture code** field of the filter and press **Show**. This should display only the French - France culture. Choose to **Edit** () it, set its **Culture alias** property to *France* and click the **OK** button to save the changes. Now the language prefix for URLs of French culture versions of documents will be displayed as for example: *<domain>/France/Home.aspx*

To enable language prefixes, go to **Site Manager -> Settings -> URLs and SEO**, and enable **Use language prefix for URLs**.



Unless you check the **Allow URLs without language prefixes** setting, all URLs without a language prefix will automatically be redirected to a corresponding URL that includes a language prefix (according to the current content culture).

You can set individual Language selection web parts to generate URLs with either a standard query string parameter or a language prefix through their **Use URLs with language prefix** property. Although please keep in mind that all other settings still apply, so the resulting URLs may be redirected.

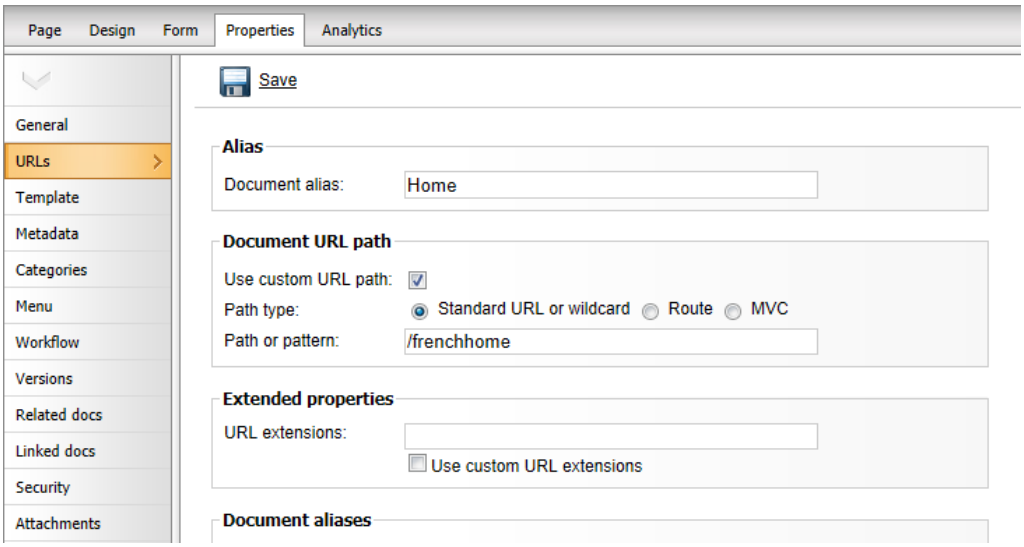
Using custom URL path for different culture versions

Using language prefixes for URLs will ensure that you have a different URL per every document culture version. The following approach can however be used if language prefixes for URLs are disabled or when you want different culture versions of documents to have different names in the URL path:

If you want the French home page to have a different URL than the English version, you need to go to

CMS Desk -> Content -> choose the French language, select */Home* in the content tree and click **Properties -> URLs**.

Check the **Use custom URL path** box, leave the **Standard URL or wildcard** option selected as the **Path type** and and set the **Path or pattern** value to:
/frenchhome



Now sign out, switch to the English language version and go to URL: *<web project>/frenchhome.aspx page.* The website culture will automatically be switched to French and the French version of the page will be displayed.

Please note that setting a custom URL path for a document's culture version may override the functionality of *Language selection* web parts on the given page. This occurs because the custom URL path has a greater priority than the selected culture by default. If necessary, you can change this behavior by adding the **CMSUseCultureForBestPageInfoResult** key into the **/configuration/appSettings** section of your *web.config* file, as shown below:

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

Please keep in mind that while this key is enabled, culture-specific document URLs will not change the current culture if a different one is selected.

It is also possible to set a Document alias for a document dedicated to a specific culture. When the document is accessed through this alias, it will always be opened in the corresponding culture.

Please 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.

4.7.5 Translation management

4.7.5.1 Overview

The translation management features brings various improvements aimed to facilitate website localization.

- Culture-dependent workflow scopes - each language version can have its own workflow defined.
- Translation status overview on the List tab - you can see which language versions of documents under the selected node are translated, missing or outdated.
- New Properties -> Languages tab - displays information about all language versions of the current document.
- Language-bound editors - you can specify users who can edit particular language versions of documents.
- Language versions comparison - enables content editors to edit two language versions of a document side-by-side.

4.7.5.2 Culture-dependent workflow scopes

Workflows can now be defined separately for each language version of your documents by using culture-dependent workflow scopes. A workflow scope defines which documents should a particular workflow be applied to. If you are not familiar with the workflow functionality of Kentico CMS, please refer to the **Workflow and versioning** chapter of this guide and especially this topic, where the process of defining a workflow is described step-by-step.

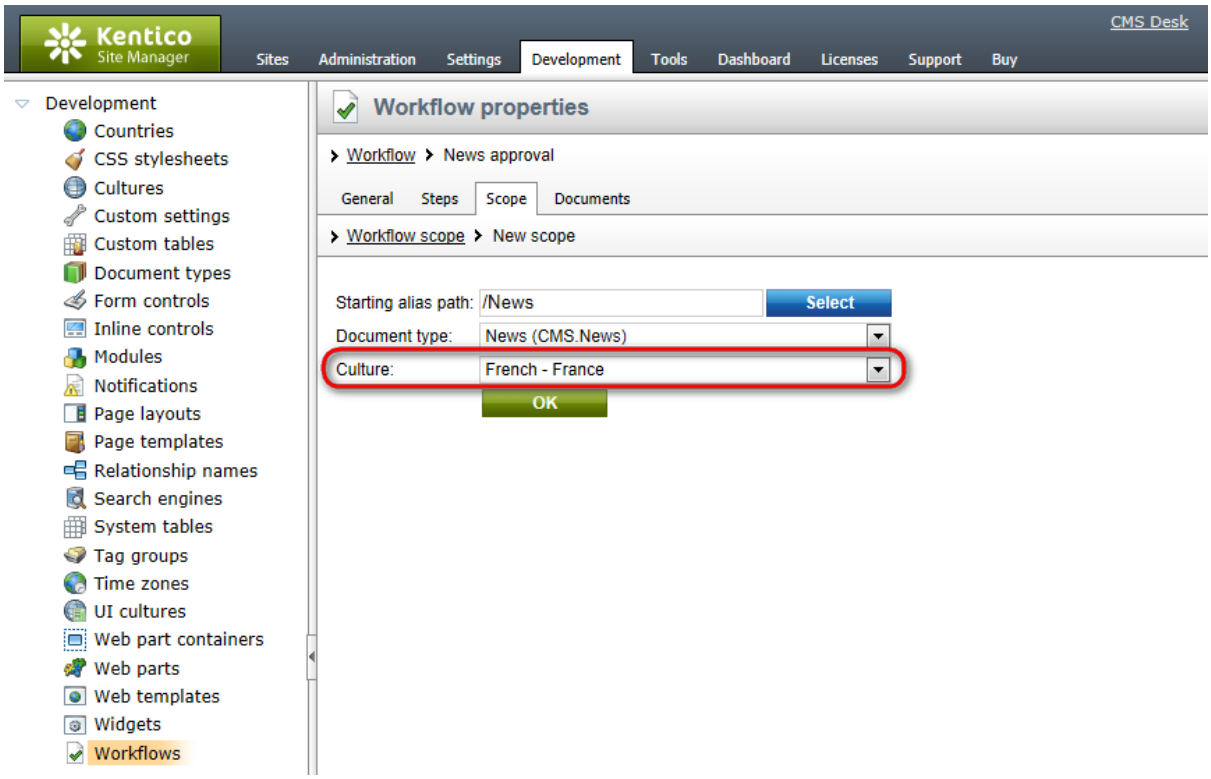
When defining a workflow scope, you can now specify the **Culture** property. Using this property, you can specify that the workflow will be applied only to the specific language versions of the documents.

The three workflow scope properties work as the following:

- **Starting alias path** - the workflow will be applied only to documents stored under the entered alias path
- **Document type** - the workflow will be applied only to documents of the selected document type
- **Culture** - the workflow will be applied only to the selected language versions of documents specified by the two properties above; choose (all) for the workflow to be applied to all cultural versions

Workflow 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



4.7.5.3 Translation status overview

CMS Desk's **List** tab provides a useful overview of language versions of the documents. By selecting a node from the content tree and switching to this tab, you can see a list of documents placed under the selected node.

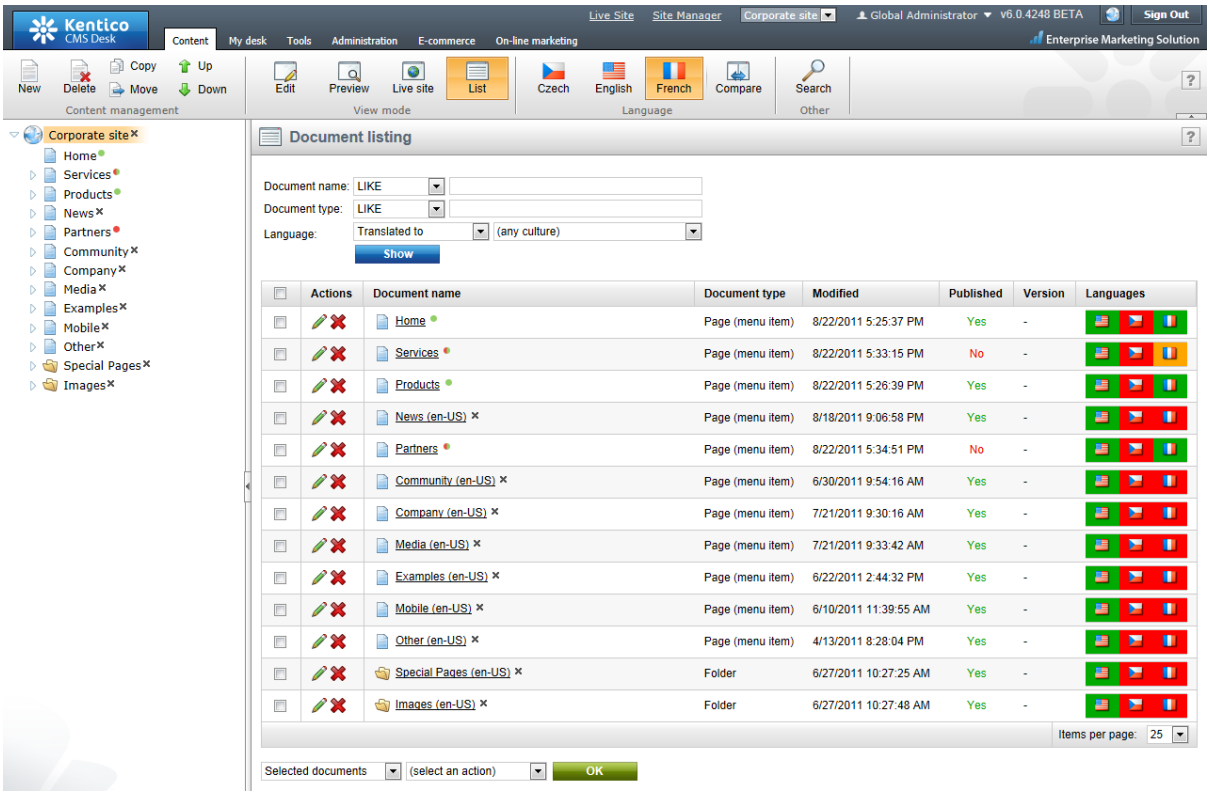
As you can see in the screenshot below, there is the **Languages** column with flags on different colored backgrounds. This column is displayed only on multi-language sites. The flags represent the particular language versions of each document.

The colors indicate the following translation statuses with respect to the default site culture:

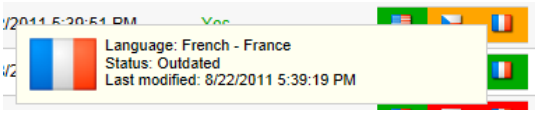
- **Green - Translated** - the document is translated and up-to-date.
- **Orange - Outdated** - the document is translated but outdated, which means that the default language version has been modified (or published when using workflow) more recently than the translated version.
- **Red - Not available** - the document's version in the language does not exist. You can also see the ✖ icon next to documents in this status, both in the listing and in the content tree.

If you click a **Translated** or **Outdated** flag, you are redirected to the **Edit -> Page** tab of the appropriate language version of the document. If you click a **Not available** flag, you are redirected to the language version's creation dialog.

Another thing you may notice in the screenshot below is the **(en-US)** string attached to some document's names. The **Name** column displays names of the documents in the currently edited culture (which is French in the screenshot). If the document in this language version **does not exist**, the column displays the name from the **default culture** with the default culture code appended in brackets.



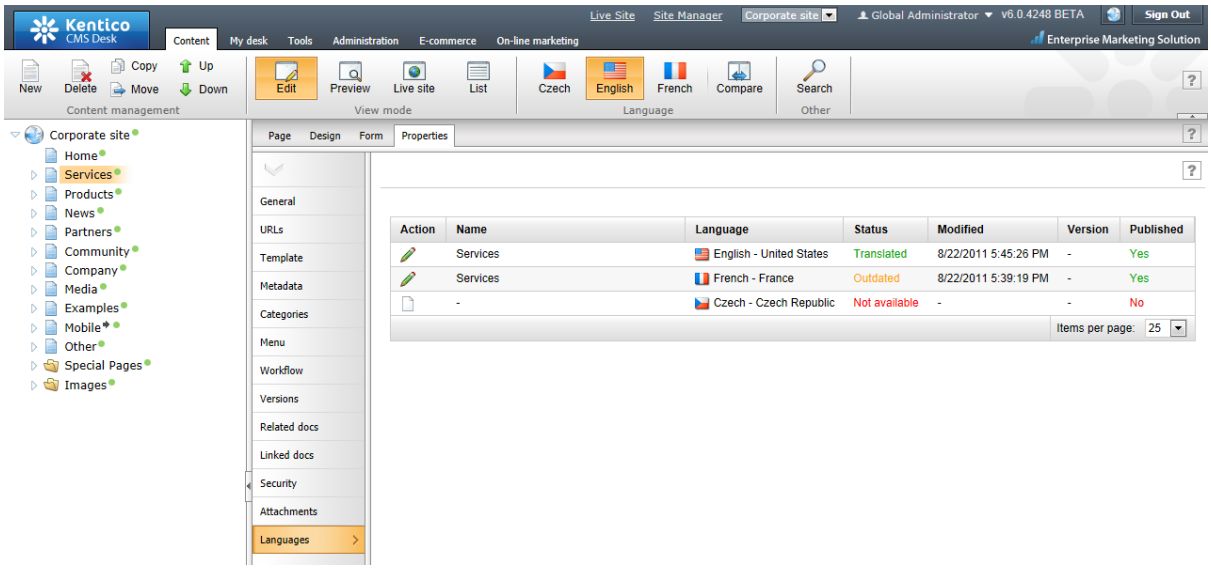
If you hover a flag with the mouse cursor, an info box will be displayed, giving additional information about the language version.



The Languages tab

The same statuses can be shown separately for each document. You can view them by selecting the document from the content tree and switching to the **Properties -> Languages** tab.

By clicking the **Edit culture version** ( ) icon, you can be redirected to the **Edit -> Page** tab of the selected culture version. By clicking the **Add new culture version** ( ) icon of a not-available version, you can proceed to creating the new culture version of the document.



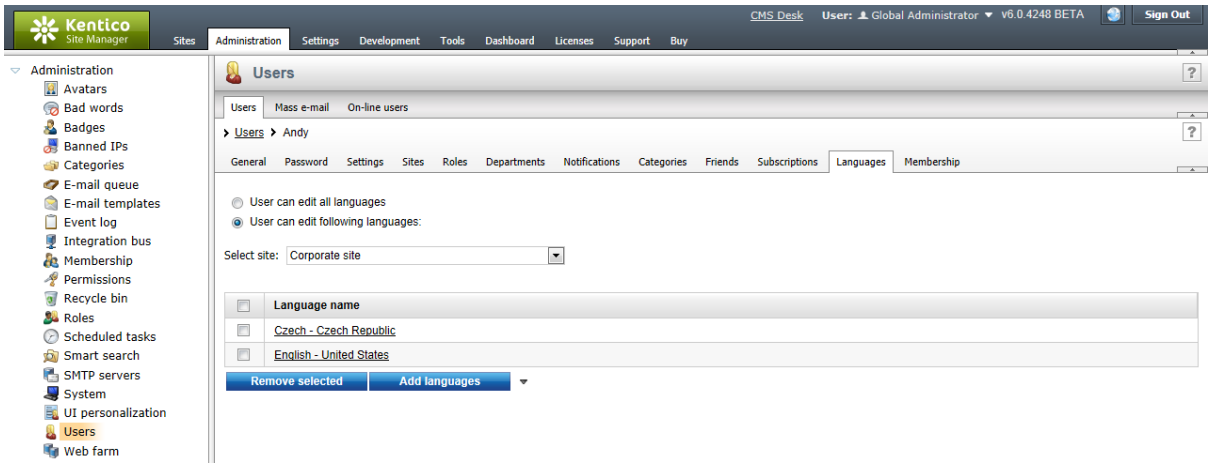
Hiding the Languages tab

The **Languages** tab can be displayed or hidden to members of particular roles. This can be set up using UI personalization as described here.

4.7.5.4 Language-bound editors

You can specify who will be able to edit which language versions of documents. This setting can be done separately for each user. If you go to **Site Manager -> Administration -> Users**, choose to **Edit** ( ) a particular user and switch to the **Languages** tab, you have the following two options:

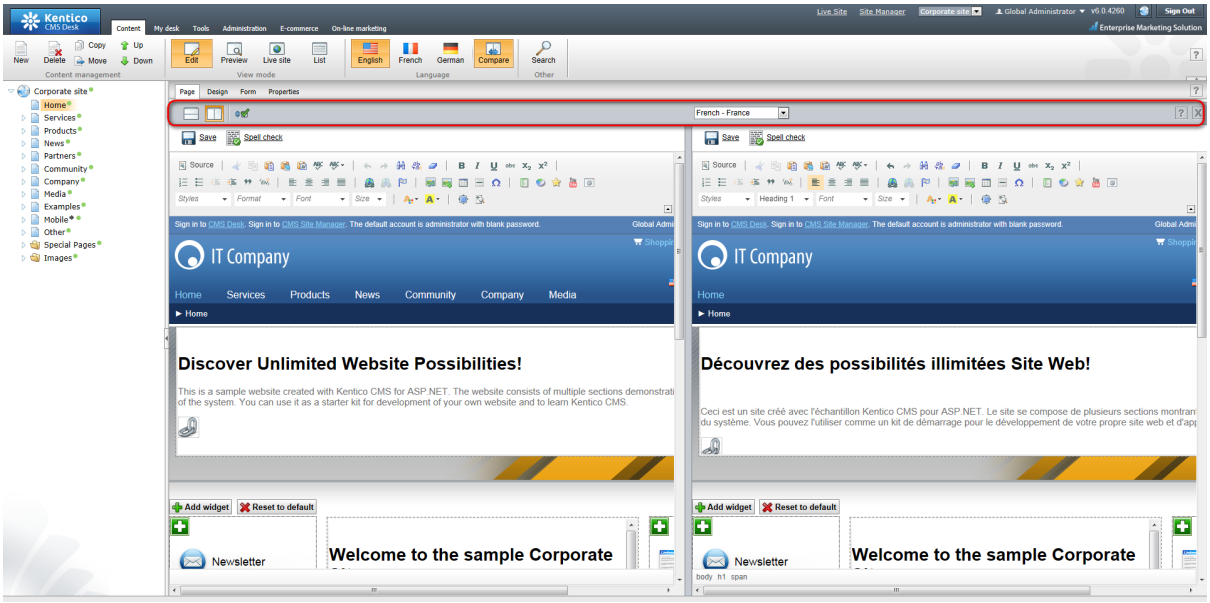
- **User can edit all languages** - if selected, the currently edited user can edit documents in all language versions of all sites in the system
- **User can edit following languages** - if selected, you can specify which language versions can be edited by the user by checking the check-boxes in the list of language versions; this can be set separately for each site in the system using the **Select site** drop-down list



4.7.5.5 Language versions comparison

Language versions 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.

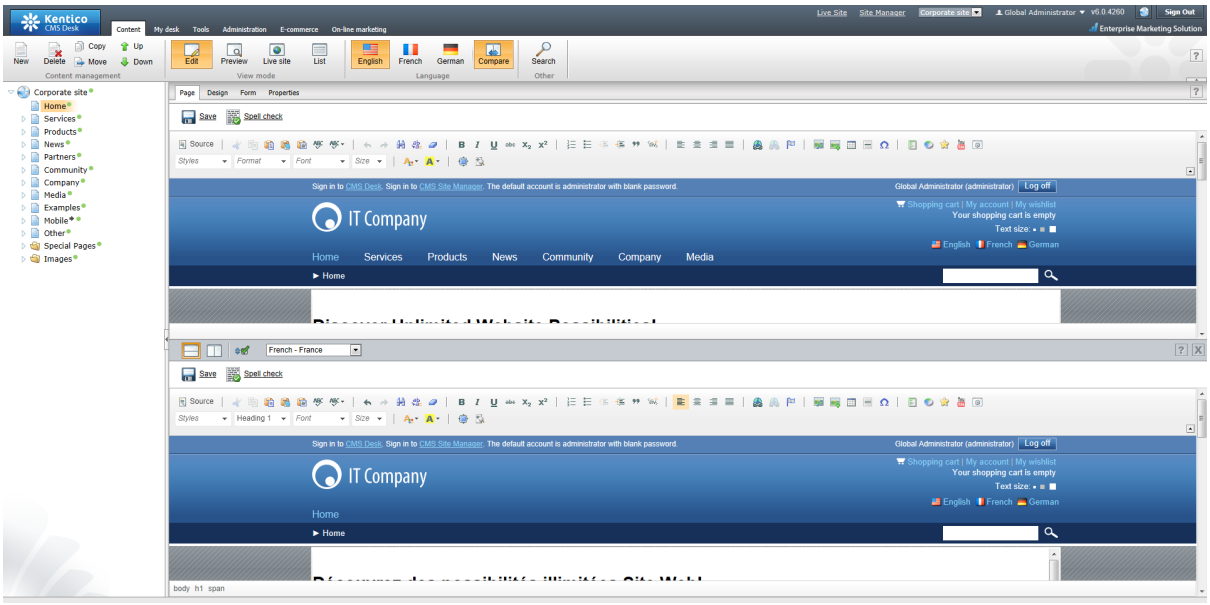
To switch to the language versions comparison mode, select a document in the content tree and click the **Compare** ( ) button in the **Language** toolbar. It results in the language versions comparison toolbar being displayed (as highlighted in the following screenshot), while the document editing area 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 versions comparison toolbar. Both displayed language versions can be edited independently at the same time.



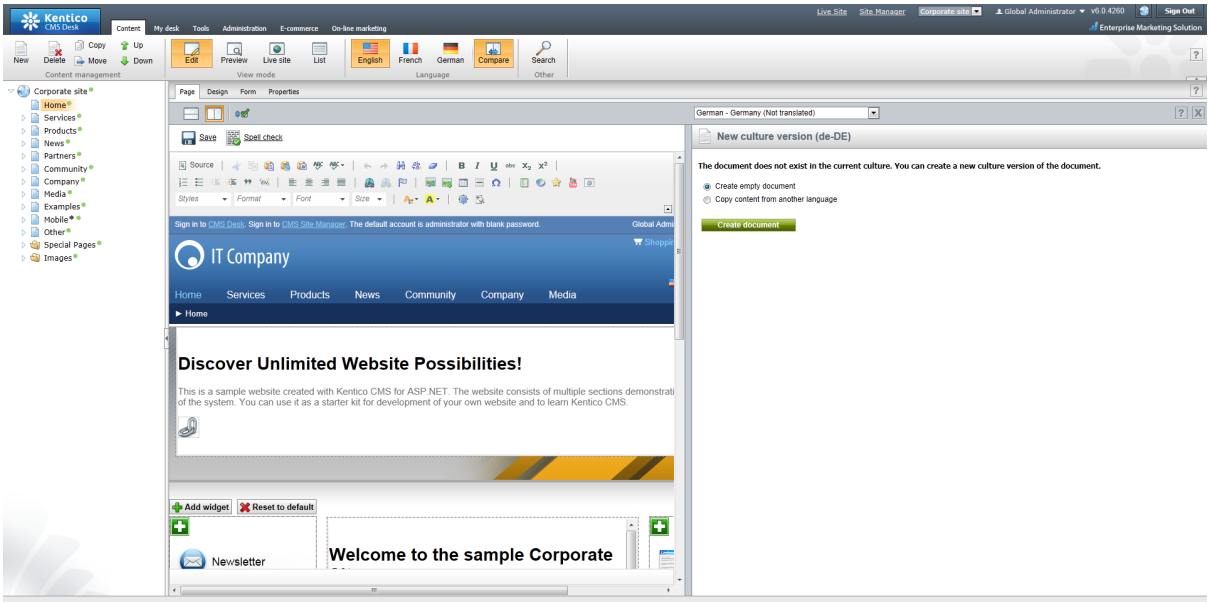
On the language versions comparison toolbar, you can find the following controls:

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 versions 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 versions 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.

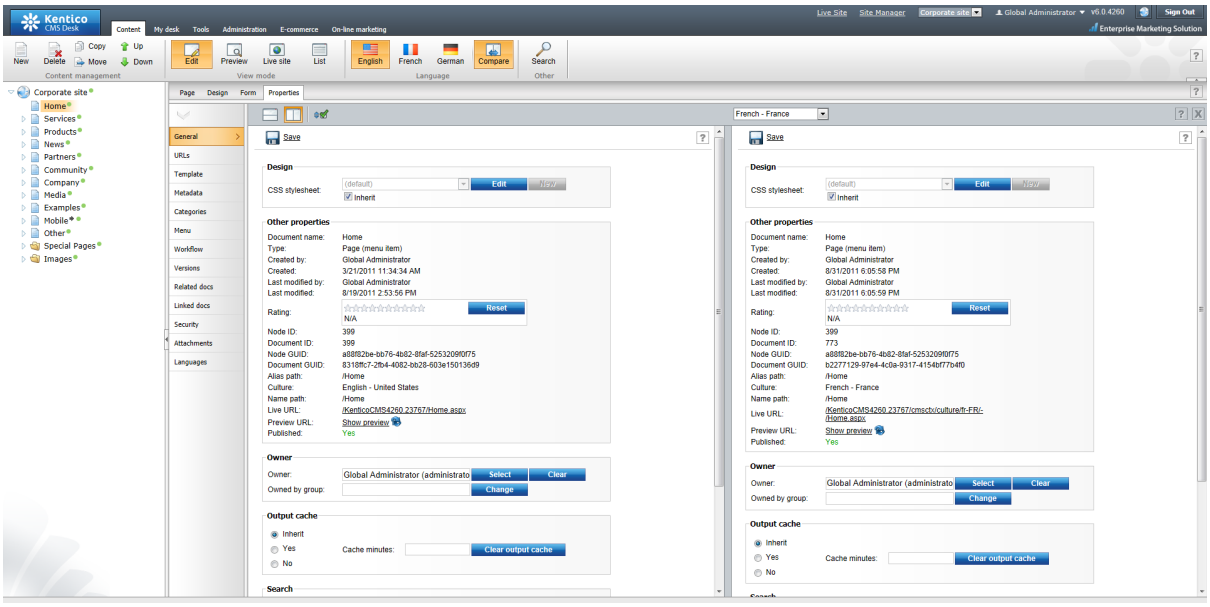
Close comparison mode	Closes the comparison mode and returns to the standard view of a single language version. The same can be achieved by clicking the **Compare** ( ) button in the **Language** toolbar again.



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 versions 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** (displayed only when editing the root document) 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.



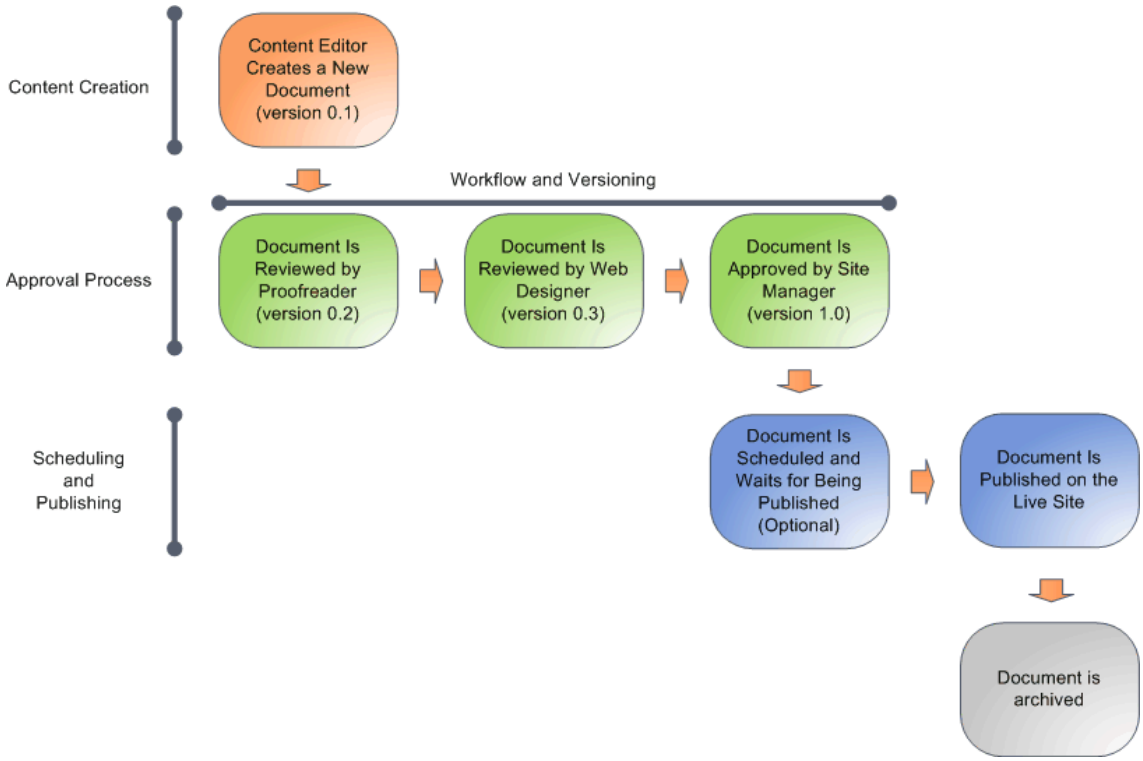
4.8 Workflow and versioning

4.8.1 Overview

Kentico CMS allows you to use workflow for all documents, including uploaded files. It includes the following features:

- The **workflow** support allows you to organize the process of content creation, updates and publishing on your website.
- The **versioning** support is tightly bound with workflow and it allows you to store (audit), view and roll-back previous versions of the content.
- It is also possible to use **versioning without workflow**, which 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, sub-sequent steps. Only members of authorized roles are allowed to modify and approve/reject a document in a particular step. The document can be switched to the following step by any authorized role member. 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.

As you can see, the workflow process can be used to ensure quality of content and design. In the following topics, you will learn how to define and apply workflow processes for your documents.

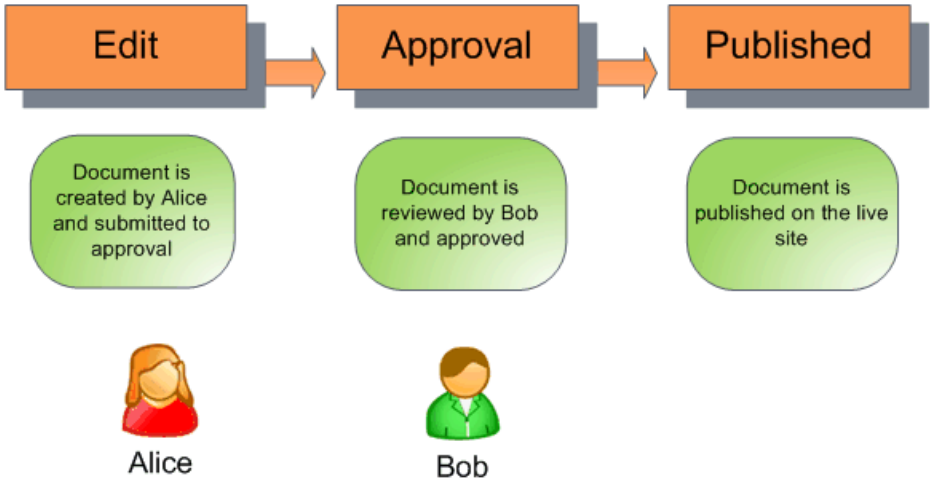
4.8.2 Defining a workflow

Workflow settings can be managed in **Site Manager -> Development -> Workflows**.

In this topic, we will create a sample workflow which will be used in the following topics. The workflow can be described as follows:

- Any editor will be able to create a news item in the **/News** section of the website.
- Then, the news item must be approved by one of the members of the **PR Managers** role. Users authorized to approve the document in the given step are also authorized to modify the document in the step.
- Approved news item is published on the website.

This a simple example of a typical real-world workflow containing custom steps. It is also possible to configure a workflow so that it contains no custom steps and is used just for versioning purposes. Configuration of such workflow is described in the Versioning without workflow topic.



Creating testing users and roles

Before we configure the workflow, we will create testing users and testing roles whose members the users will be.

1. Go to **Site Manager -> Administration -> Users** and click **New user** ( ). Create a new user with following details:

- **User name**: Alice
- **Full name**: Alice Murphy
- **E-mail**: <your e-mail>
- **Enabled**: yes
- **Is editor**: yes

Click **OK**. Go to the **Sites** tab and assign the user to your current website. Go to the **Roles** tab and assign the user to the **CMS Editors** role of the given website.

2. Create another user with the following details:

- **User name**: Bob
- **Full name**: Bob Johnson
- **E-mail**: <your e-mail - you may want to use a different e-mail for testing purposes>
- **Enabled**: yes
- **Is editor**: yes

Click **OK**. Go to the **Sites** tab and assign the user to your current website. Go to the **Roles** tab and assign the user to the **CMS Editors** role of the given website.

3. Go to **Site Manager -> Administration -> Roles**, choose the appropriate site from the **Site** drop-down-list and click **New role** ( ). Enter the following details:

- **Role display name**: PR Managers
- **Role code name**: PRManagers

Switch to the **Users** tab and add user *Bob Johnson* to this role. As you can see, Alice is the news editor (because of the *CMS Editors* role) and Bob is the PR Manager who will approve the news (*PR Managers*

role) and he can also edit the news (*CMS Editors* role).

Please note

The *PR Managers* role has no permissions configured for the purpose of this example. This means that if a user was a member of just this single role, they would not be able to perform a full range of tasks such as creating, editing or deleting documents, etc. In your real website, you would typically need to go to **Site Manager -> Administration -> Permissions** and grant appropriate permissions to the role, especially those in the **Modules -> CMS Content** permission matrix.

Please see the Development -> Membership, permissions and security -> Permissions chapter for detailed information on how to do this.

Defining a new workflow

With the testing users and roles defined, we may start creating the actual workflow.

4. Go to **Site Manager -> Development -> Workflows**, click **New workflow** ( ) and enter the following values:

- **Display name**: News approval
- **Code name**: NewsApproval
- **Automatically publish changes**: disabled
- **Use check-in/check-out**: No

Click **OK** to create the workflow.

Defining custom workflow steps

If you switch to the **Steps** tab, you should see that each workflow has **three default steps**:

- **Edit** - the document was created or modified after it was published.
- **Published** - the document is published on the live site. When you edit such a document, it's automatically moved to the **Edit** step.
- **Archived** - the document is not published on the live site, but it remains in the content repository.

We will add one extra custom step to the workflow.

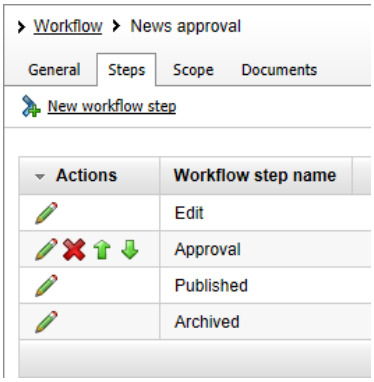
5. Switch to the **Steps** tab and click **Add workflow step** ( ). Enter the following details in the displayed dialog:

- **Display name**: Approval
- **Code name**: Approval

Click **OK**. In the workflow step editing interface, switch to the **Roles** tab. Choose the appropriate website in the drop-down list and add the **PR Managers** role to the list. Now the *PR Managers* are authorized to edit, approve or reject documents in this step.

When you view the list of workflow steps now, it looks as in the screenshot below. As you can see, the

custom steps are placed between the **Edit** and **Published** steps. If there is more than one step, you can change their order using the up (⬆) and down (⬇) arrows.



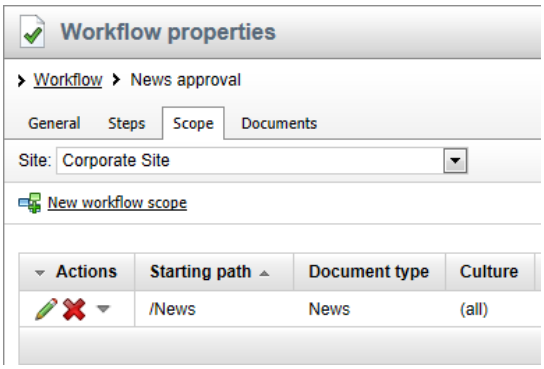
Defining a workflow scope

Workflow scopes define which documents should the workflow be applied to.

6. Switch to the **Scopes** tab, choose the appropriate website from the drop-down list and click **New workflow scope** (). Enter the following details:

- **Starting alias path**: */news* (this is the section of the website for which the workflow will be applied)
- **Document type**: *News*
- **Culture**: *(all)*

Click **OK**.



You have configured the workflow process for the news documents. In the Using workflow topic, you will learn how the workflow works in our sample scenario.

> **Using workflow for the root document**
>
> It is possible to set up a workflow so that it is applied to the root document of your content tree. This may be useful in some cases, but you always have to be careful if you choose this setup.

> The reason for this is that the root document provides not only the master page, but also base settings for all documents that inherit them. If you setup workflow on the root, it may eventually happen that the document gets unpublished for some time if an editor is not careful. This can break the entire site for all public visitors for that time.

Workflow's documents overview

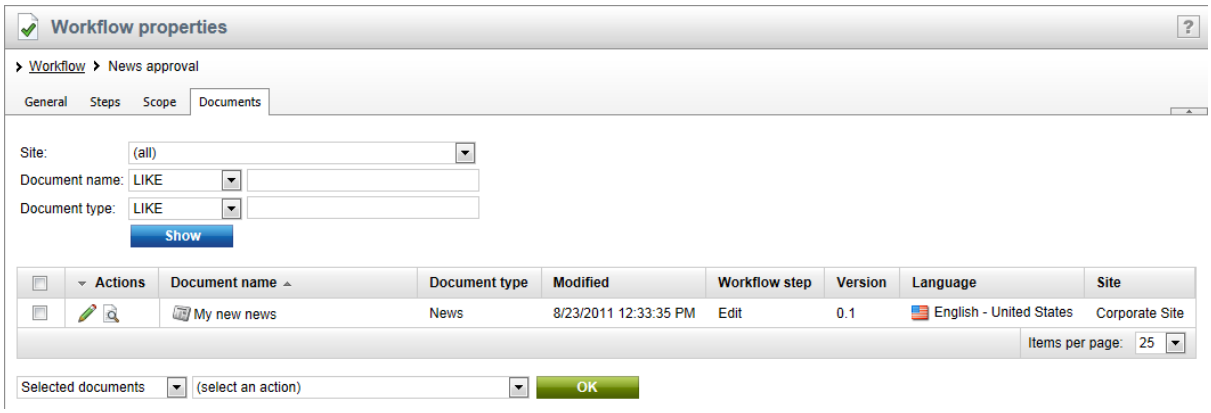
On the **Documents** tab of a workflow's editing interface, you can see a list of documents that are using the workflow. Clicking a document in the list opens it for editing in **CMS Desk -> Edit**.

The following actions can be performed with documents in the list. These actions are also useful if you want to **finish the documents' workflow**:

- **Publish and finish workflow** - the new version of the document will be approved and published
- **Remove workflow and keep currently published data** - the original version of the document remains published and all the changes will be discarded

Based on the selection in the first drop-down list, these actions can be performed for:

- **All documents** - the action will be performed with all documents
- **Selected documents** - the action will be performed with documents selected by the check-boxes ( )



4.8.3 Using a workflow

This example explains how the workflow process works from the perspective of content editors. It requires the system to be configured as described in Defining a workflow.

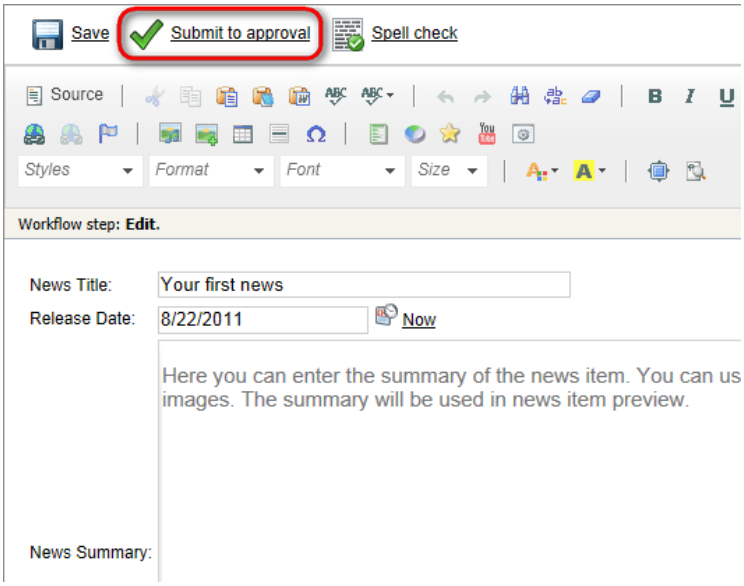
Creating a new document

1. Sign in to **CMS Desk** as **Alice** and create a news item in the **/News** section:

- **News title**: My first news item
- **Release date**: <click **Now** to get the current date>
- **News summary**: Some summary
- **News text**: Some text

- **Publish from-to**: leave the fields blank

Click **Save**. After the document is saved, you can see that there is the **Submit to approval** action displayed in the document menu:



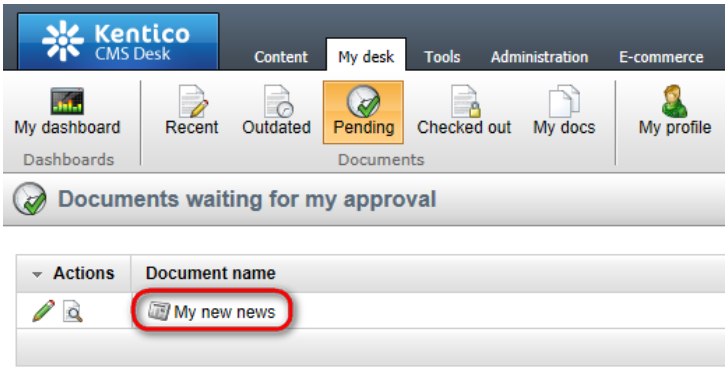
2. Before we submit the document to approval, we will preview it for a final check. Click **Preview** in the main toolbar and you will see exactly what the document will look like on the live site. After doing so, go back to the editing form and click the **Submit to approval** button.

The document has been moved to the **Approval** step and it is now waiting for approval by some member of the *PR Managers* role and Alice can no longer edit it. Also, a notification e-mail is sent to all members of the PR Managers role.

Approving the document

With the document submitted to approval, we can switch to the PR Manager's perspective and approve it.

3. Sign out of **CMS Desk** and sign in again as **Bob**. Go to **My desk -> Pending**. You should see the new document listed as in the screenshot below.



4. Click the document, which will open it for editing in **CMS Desk -> Content -> Edit**. You can now preview and optionally edit it.

If you were not satisfied with it and wanted Alice to re-work it, you could click **Reject** to return it back to the **Edit** step. For the purpose of this example, we will approve it by clicking the **Publish** button. Once you approve the document, it gets published on the live site. A notification e-mail is sent to Alice so that she knows the document has been published if configured as described in E-mail notification in workflow process.

Tip 1: Approval/rejection comments

When you're approving or rejecting the document, you can go to **Properties -> Workflow**, add additional comments and click **Approve**.

Tip 2: "Documents waiting for my approval" and "Recent documents"

Each user can find a list of documents waiting for their approval in **My desk -> Pending**.

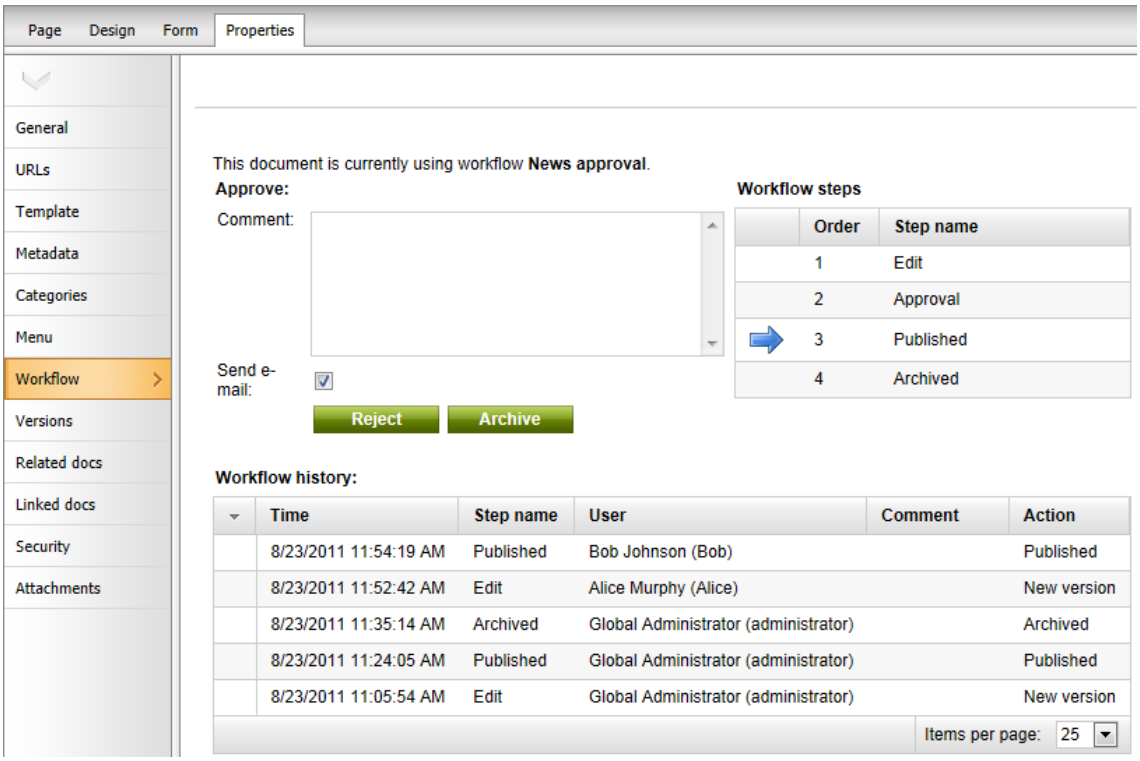
Similarly, the user who submitted the document can see the document status in **My desk -> Recent**.

Archiving the document

If you want to archive the document so that it's no longer displayed on the live site, but it remains in the content tree and in Kentico CMS database, you can go to the **Properties -> Workflow** dialog of the document and click the **Archive** button. This can be performed in any workflow step, unless the document is already archived.

Workflow history

In the **Properties -> Workflow** dialog, you can also see the workflow history, i.e. a list of all workflow status changes made throughout the document's lifecycle. With each status change, you can see its exact date and time, final workflow step and the user who performed the action and the actual type of action that was performed.



Workflow and permissions

Once you enable workflow, the user needs to be authorized to edit the document and to manage the document in the give workflow step.

The users with permission "Manage workflow" can approve/reject any document in any workflow step, even if they are not authorized for the given workflow step. This permission can be set in the **Site Manager -> Administration -> Permissions** section, in the permission matrix called **Modules: Content**.

4.8.4 E-mail notification in workflow process

The workflow process automatically notifies the members of the authorized roles that there's a document waiting for their approval. It also sends a notification to the document author (the user who last edited the document in the Edit step and sent it to the next workflow step) when the document status changes. Lastly, a notification is also sent to all users who have the **Manage workflow** permission for all content .

Please note: notification e-mails are not sent among users who have the same e-mail address, i.e. when user1 and user2 have the same address and user1 sends a document for user2's approval, no notification e-mail is sent.

You can turn off e-mail notifications in the workflow process in **Site Manager -> Settings -> Content -> Content Management**. Here you can set the following values:

- **Send workflow e-mails** - indicates if workflow e-mails should be sent.

- **Send workflow e-mails from** - the e-mail address from which the e-mails will be sent.

Workflow notification e-mail templates

You can customize the workflow notification e-mails in **Site Manager -> Administration -> E-mail templates**. The workflow uses the following e-mail templates:

- **Workflow - Document approved**
- **Workflow - Document archived**
- **Workflow - Document published**
- **Workflow - Document ready for approval**
- **Workflow - Document rejected**

You can use the following specific context macros in the workflow notification templates:

- **{% applicationurl %}** - URL of the current Kentico CMS application, including protocol and domain name.
- **{% approvedby %}** - e-mail address of the user who approved the document.
- **{% approvedwhen %}** - date and time of approval/rejection/archiving.
- **{% originalstepname %}** - original workflow step in which the document was before the approval/rejection/archiving.
- **{% currentstepname %}** - workflow step into which the document was switched by the approval/rejection/archiving.
- **{% comment %}** - comment added by the user who changed the workflow status of the document.
- **{% firstname %}** - first name of the user who changed the workflow status of the document.
- **{% lastname %}** - last name of the user who changed the workflow status of the document.
- **{% username %}** - username of the user who changed the workflow status of the document.
- **{% email %}** - e-mail address of the user who changed the workflow status of the document.
- **{% fullname %}** - full name of the user who changed the workflow status of the document.
- **{% documentpreviewurl %}** - preview link to the unpublished version of the document.
- **{% documentediturl %}** - URL leading to editing interface of the document in CMS Desk.

You can also access the following objects and their properties (e.g. *{% User.UserName %}*):

- **{% Document %}** - *TreeNode* object of the document whose workflow status was changed.
- **{% User %}** - *UserInfo* object of the user who changed the workflow status of the document.
- **{% OriginalStep %}** - *WorkflowStepInfo* object of the original workflow step in which the document was before the workflow status change.
- **{% CurrentStep %}** - *WorkflowStepInfo* object of the workflow step into which the document was switched.

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.

4.8.5 Versioning and rollback

Versioning allows you to store all versions of a document which uses workflow. You can view the versions of a document in **CMS Desk -> Content -> Properties -> Versions**.

You can perform the following actions with 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 use the **Destroy history** button to clear the history and restart version numbering. The **Destroy** permission for this document must be granted to the user.

The history also shows when the particular version has been published and when it was replaced with a new version.

The length of the version history can be configured in **Site Manager -> Settings -> Content -> Content management -> Version history length**.

Automatic version numbering

Kentico CMS supports automatic version numbering:

- If you use workflow without content locking, automatic version numbering is used by default.
- If you use content locking, this option is optional and can be configured in **Site Manager -> Settings -> Content -> Content management -> Use automatic version numbering**.

The automatic version numbering works like in the following example:

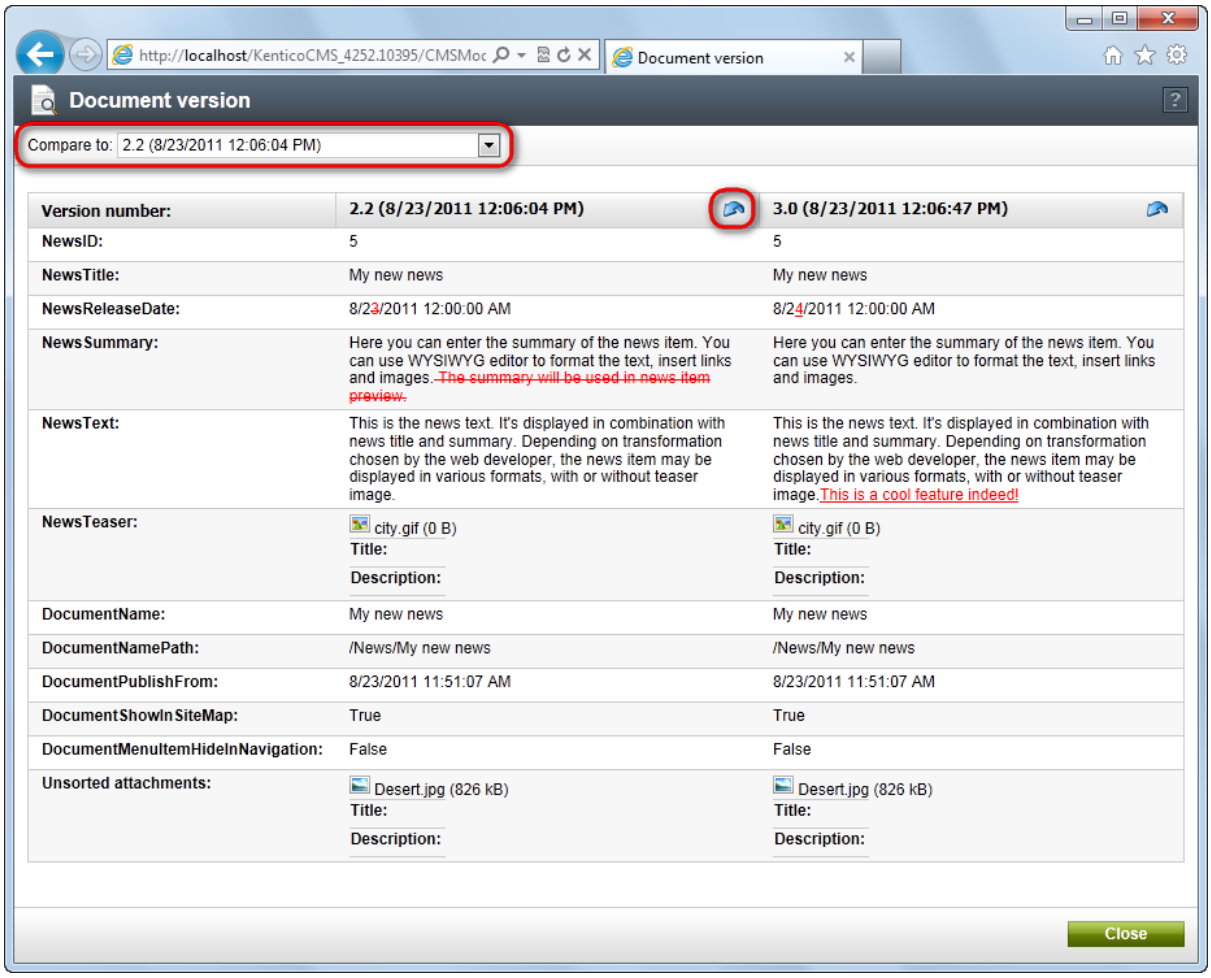
- 0.1 - first version of the document when it's created
- 0.2 - second modification of the document
- 1.0 - first published version of the document
- 1.1 - first modification of the published document
- 2.0 - second published version of the document

Side-by-side comparison of versions

If you view click the **View** ( ) icon next to a version, the pop-up window allows you to view a side-by-side comparison with other document versions.

You only need to select the appropriate version in the **Compare to** drop-down list. Content of the two versions will be displayed side-by-side with the changes highlighted in red font. You can roll back the document to a particular displayed version by clicking the **Rollback** ( ) icon in the top-right column of

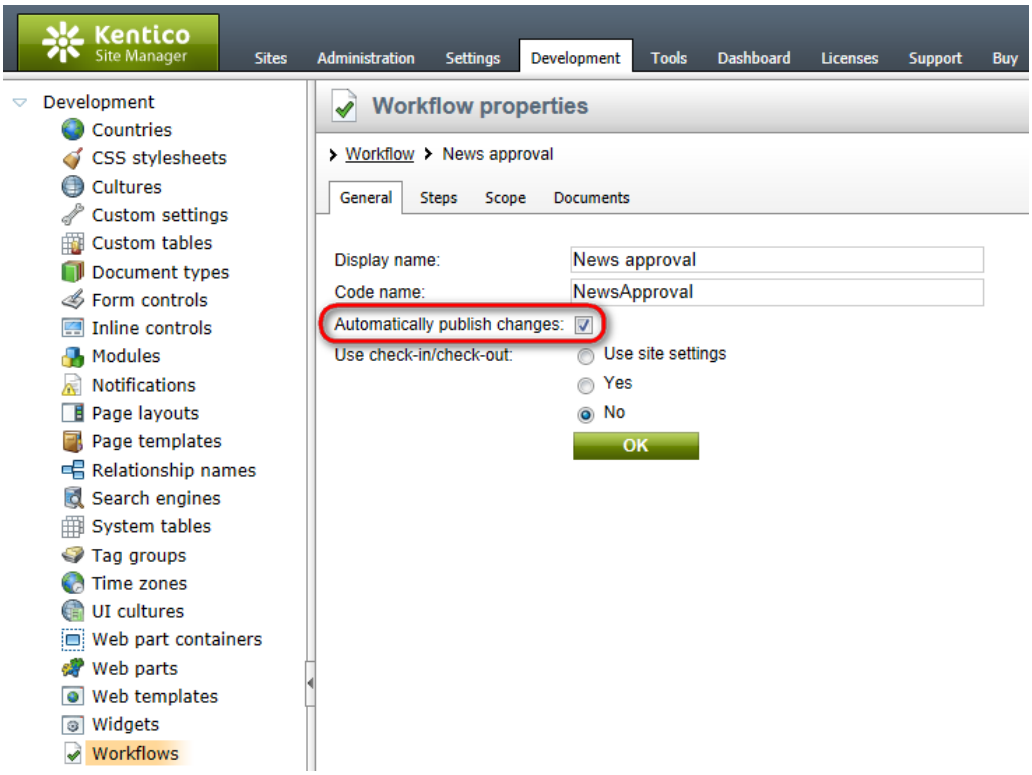
the version's column.



4.8.6 Versioning without workflow

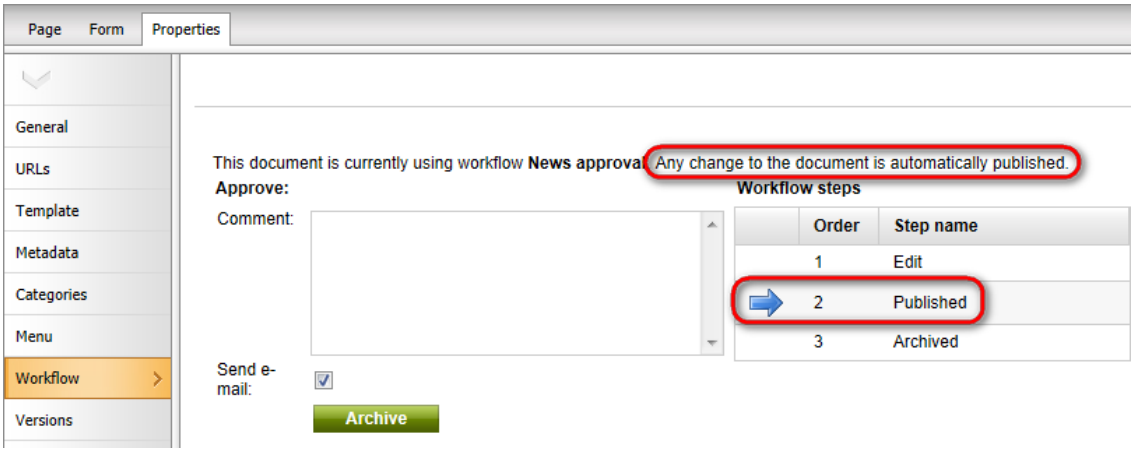
It is possible to use versioning without actually using any workflow steps. In other words, it is possible to have a new document version created whenever a document is modified and saved.

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.



4.8.7 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 configuration

You can enable content locking for versioned documents at two levels:

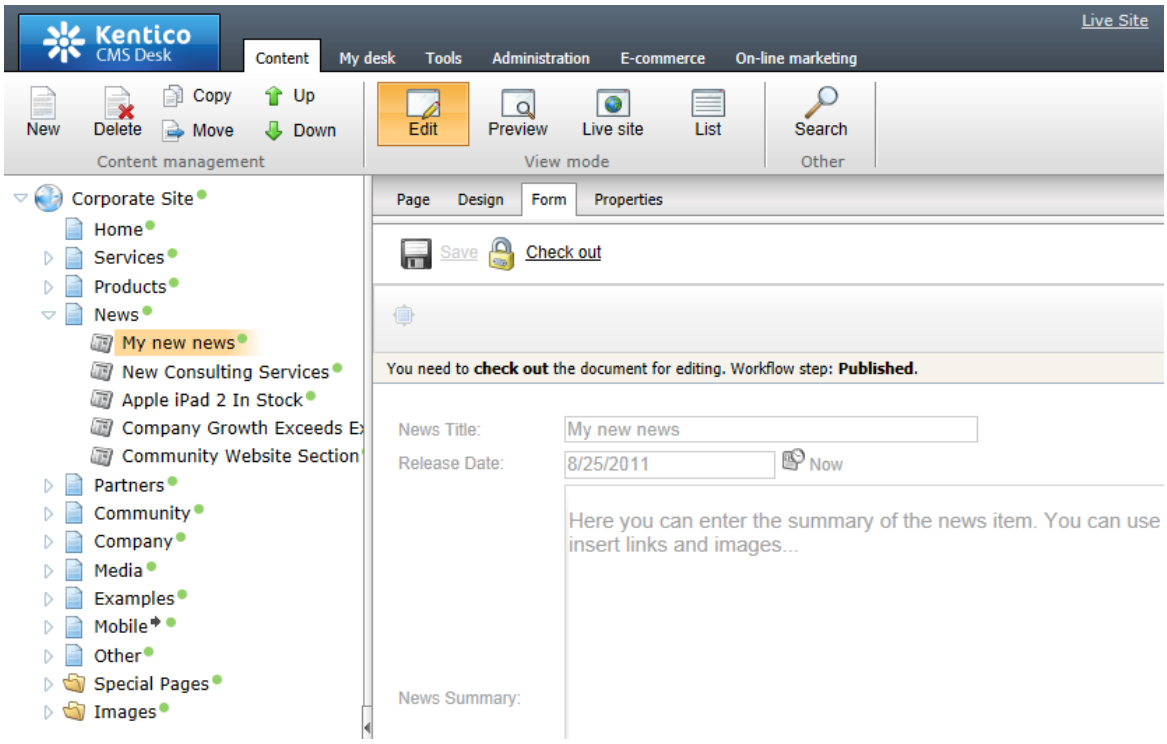
- Globally in **Site Manager -> Settings -> Content -> Content management -> Use check-in/check-out**.
- Separately for each particular workflow, in **Site Manager -> Development -> Workflows -> edit ( ) a workflow -> General**; here, you can configure the **Use check-in/check-out** option to either inherit from the global settings, or set explicitly if content locking should be used with this workflow.

Example

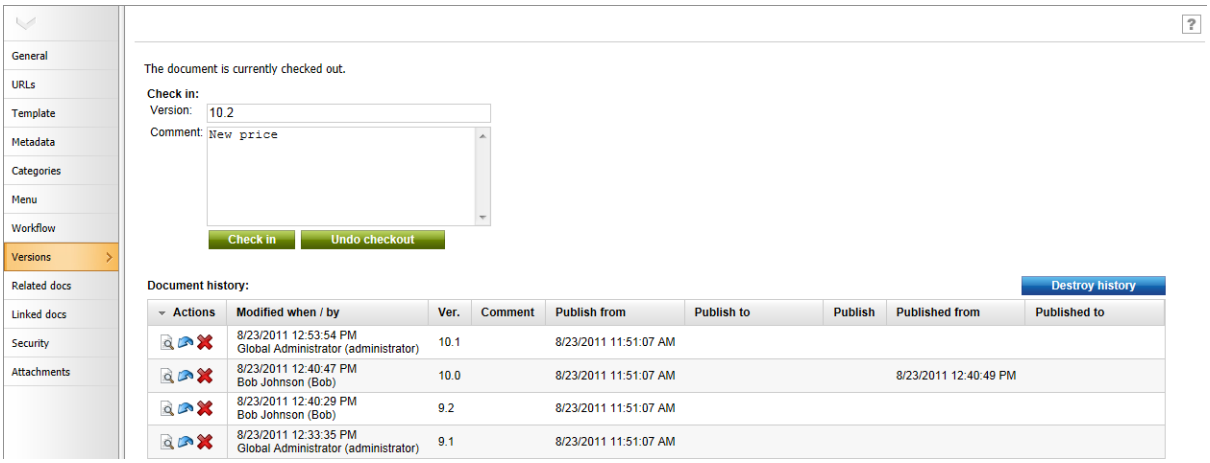
Here's an example of operations in the document life cycle when content locking is used:

1. User Alice creates a news item. The news item is automatically checked out (locked) for her. The other users cannot edit it.
2. User Alice finishes the changes and checks in the document.
3. Now the other users can modify the document as well. As the document is checked in, Alice can also submit it to approval.

When you try to edit a checked in document, you will get a disabled editing form with **Check out** button:



You can also use the **Check in/Check out** buttons in **Properties -> Versions dialog**. This dialog allows 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 the **Site Manager -> Administration -> Permissions** section, in the permission matrix called **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 **Undo check-out** buttons.

4.8.8 Workflows internals and API

4.8.8.1 Overview

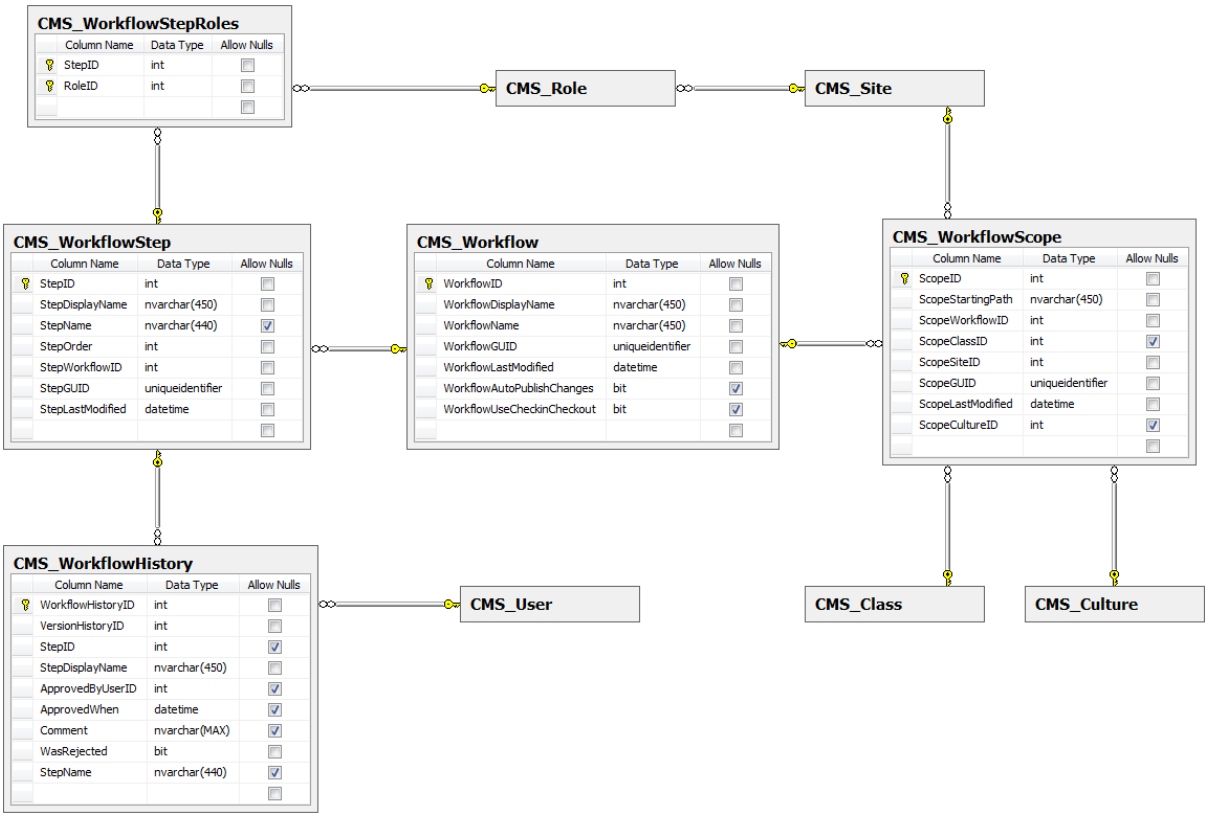
In this chapter, you will learn which database tables and API classes are used by workflow and versioning. You will also see the most common API examples.

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.

4.8.8.2 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.



4.8.8.3 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.

4.8.8.4 API examples

4.8.8.4.1 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;
```

4.8.8.4.2 Managing w orkflow s

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);
}
```

4.8.8.4.3 Managing w orkflow 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;
}
```

4.8.8.4.4 Managing w orkflow 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;
}
```

4.9 Validators

4.9.1 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.

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** and **Link checker** 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.

4.9.2 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.

The following actions are available in the top row:

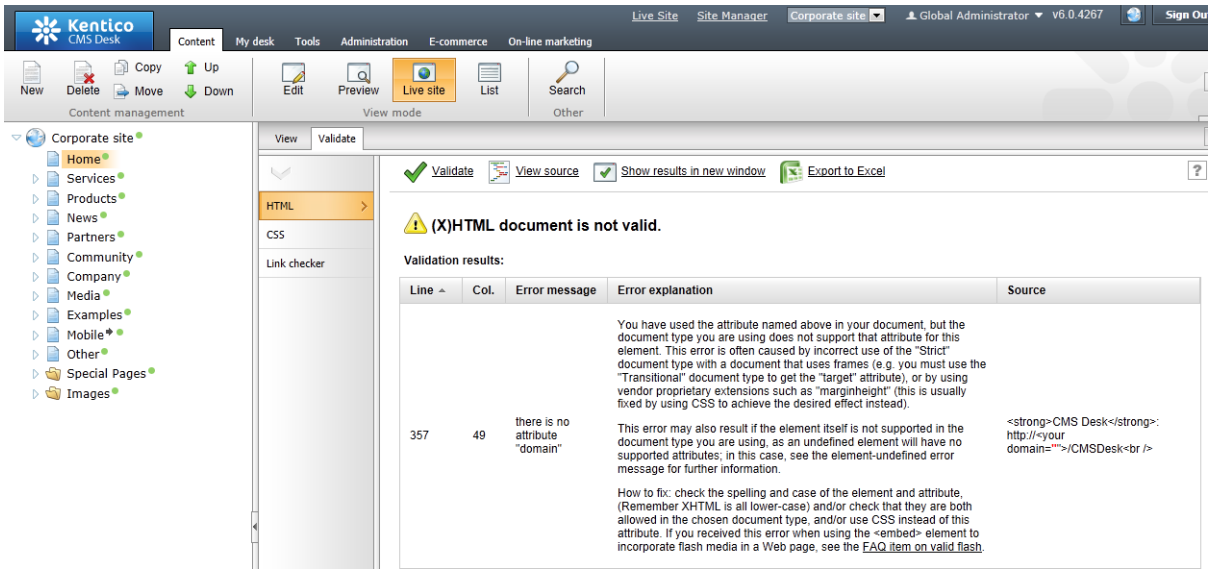
- **Validate** - performs HTML validation of the page currently selected in the content tree and displays validation results below.
- **View source** - opens a new pop-up windows where source code of the page selected in the content tree will be displayed.

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 Excel** - exports validation results to an Excel spreadsheet (in XLSX format).

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.
- **Col.** - column of the HTML code (i.e. number of character 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 code where the validity issue appears.



4.9.3 CSS validator

On the **Validate -> CSS** tab, you can perform validation of CSS style definitions used on 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.

Only styles taken from linked stylesheets and inline styles declared in the head of the page are validated — inline styles declared as attributes of individual HTML elements are not validated. Validation is always performed against the CSS 2.1 standard.

The following actions are available in the top row:

- **Validate** - performs CSS validation of the page currently selected in the content tree and displays validation results below.
- **View source** - opens a new pop-up windows where source code of the page selected in the content tree will be displayed.

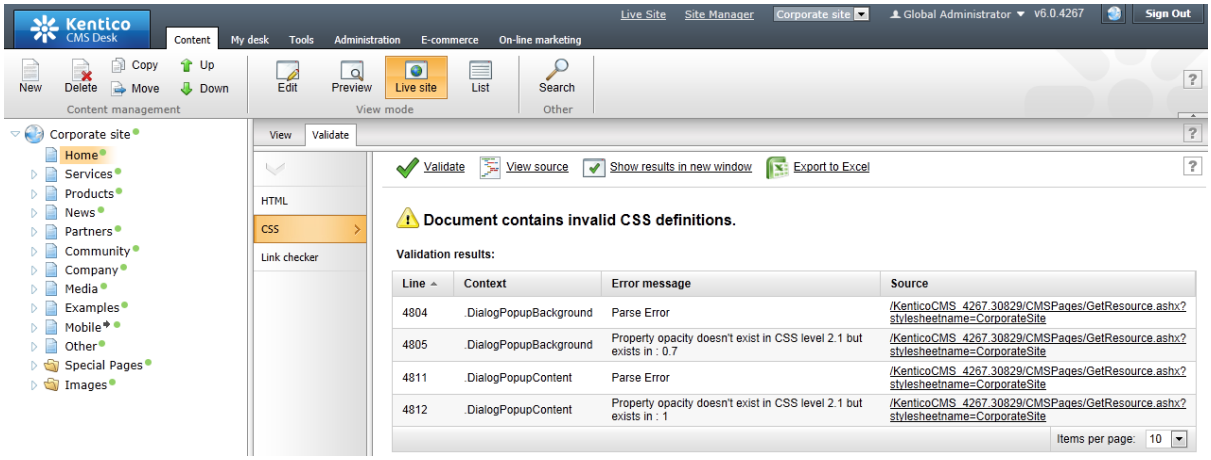
The following actions are only available after performing validation which showed that the document is not valid:

- **Show results in new window** - displays validation results in a new window.
- **Export to Excel** - exports validation results to an Excel spreadsheet (in XLSX format).

When invalid style definitions are found, 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 CSS stylesheet (or approximate line of the HTML source code) where the issue was found.
- **Context** - name of the CSS class which contains the issue.
- **Error message** - message explaining the issue.
- **Source** - if the issue was found in a linked stylesheet, a link to this stylesheet is displayed in this column. In case that the current user has permissions to edit the stylesheet, clicking the link opens

a pop-up window with stylesheet editor and the stylesheet can be edited directly. If the user does not have these permissions, the stylesheet can only be viewed by clicking the link. If the issue was found in inline styles, a message is displayed in the column informing about this fact.



4.9.4 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:

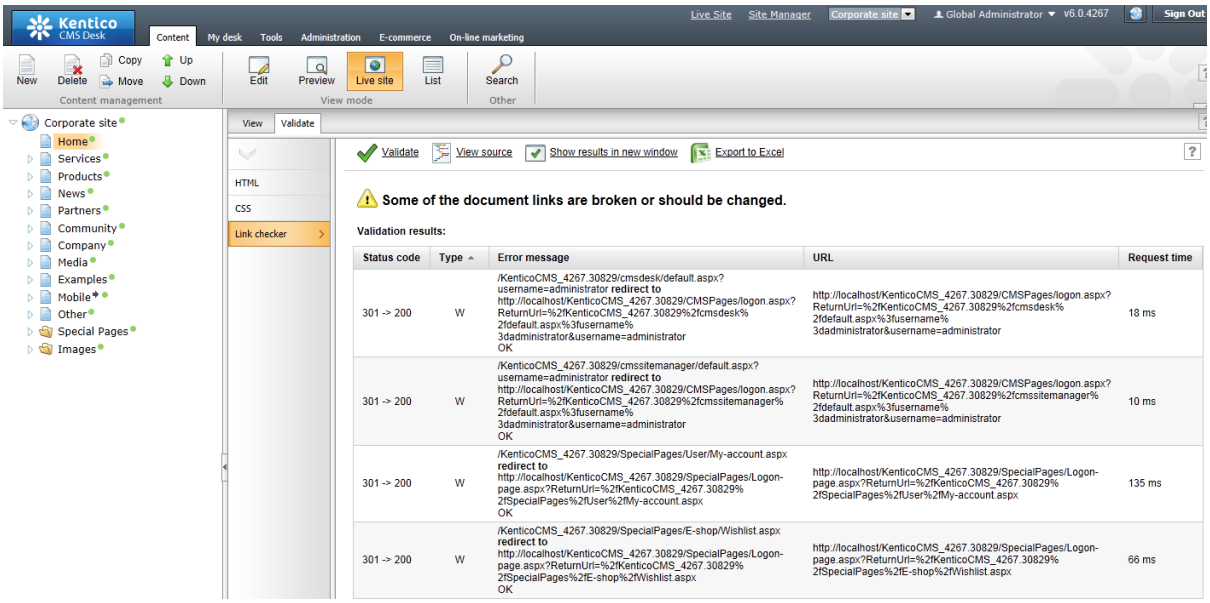
- **Validate** - performs validation of links on the page currently selected in the content tree and displays validation results below.
- **View source** - opens a new pop-up windows where source code of the page selected in the content tree will be displayed.

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 Excel** - exports validation results to an Excel spreadsheet (in XLSX format).

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.

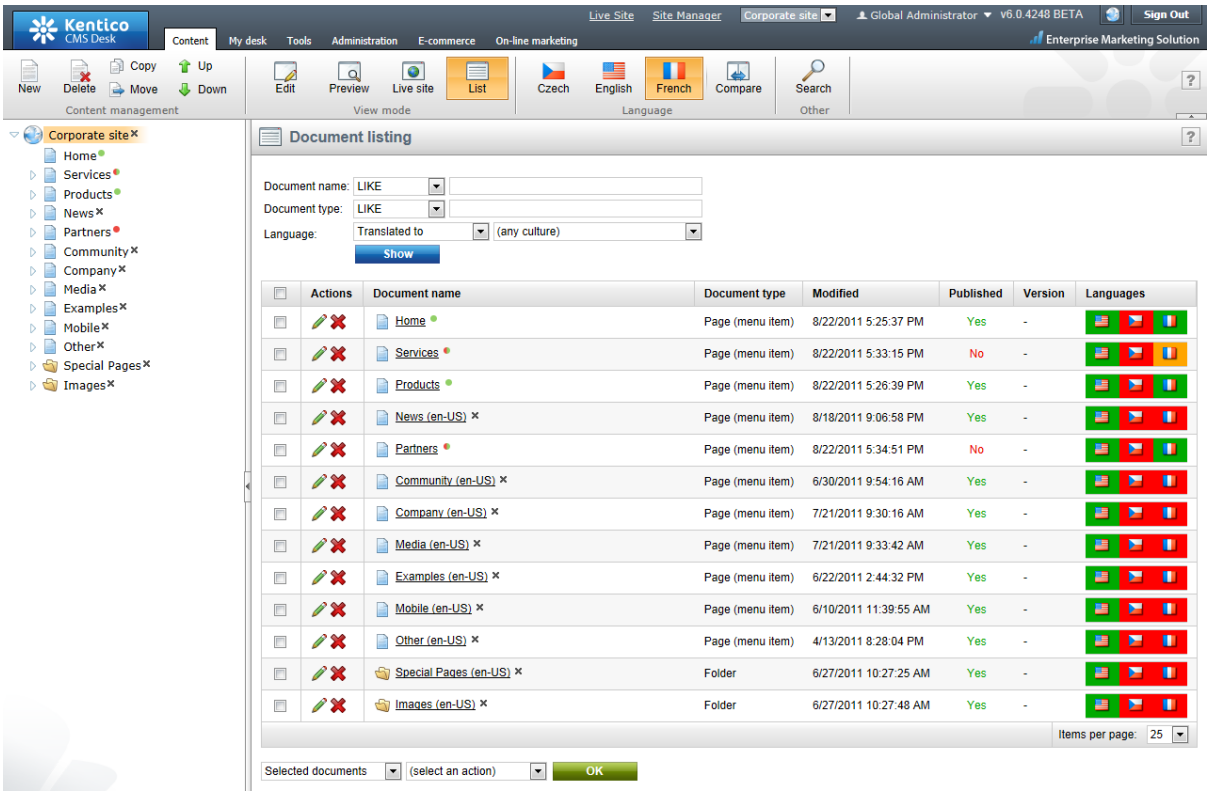


4.10 Document listing

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 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 translation statuses of the documents' particular language versions. More information about translation statuses can be found in Development -> Multilingual and international support -> Translation management -> Translation status overview.



Single document actions on the List tab

The following operations can be performed with particular 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.
- By clicking a document's name, the list tab will display the documents found under the clicked document.

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 document

permanently, i.e. without storing it in the recycle bin.
  - **Delete all culture version of the specified documents** - only for multilingual sites. If enabled, all culture versions of the documents will be deleted. If disabled, only the versions in the currently edited culture will be deleted.
- **Publish/Archive** - this option is available only for global administrators and publishes/archives the selected documents. Before performed, a dialog gets 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.

4.11 Content search

The interface for searching the documents in the content tree can be accessed by clicking the **Search** (🔎) icon in **CMS Desk -> Content**.

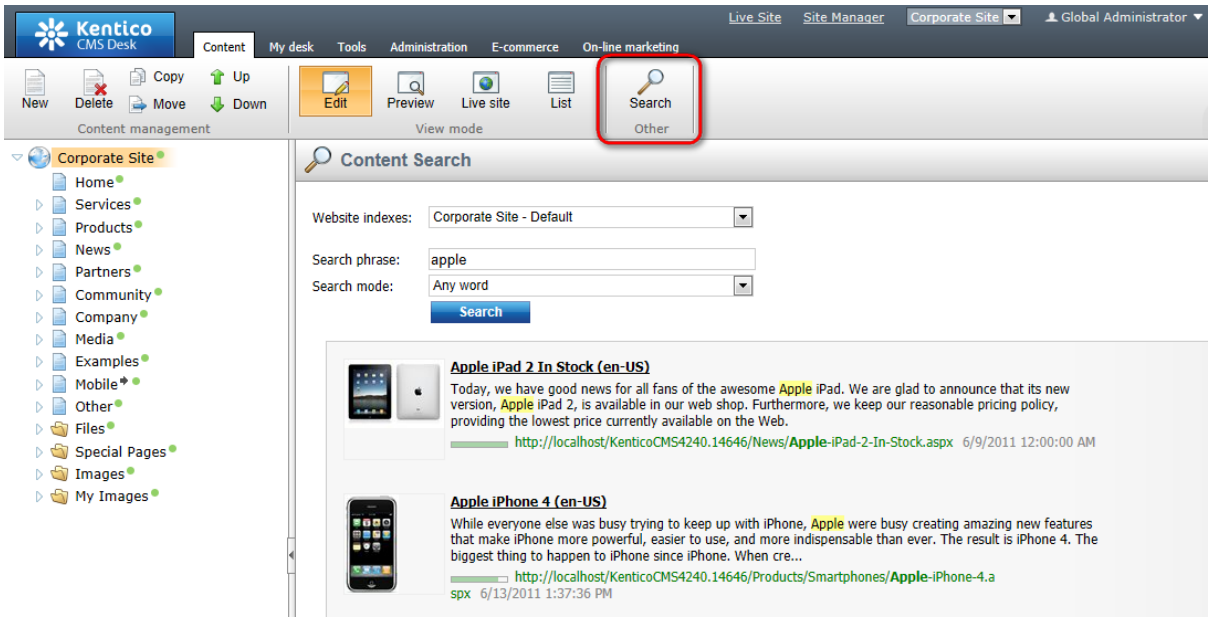
1. First, you need to decide if you want to perform the search using Smart search or SQL search. This depends on the selection made in the **website indexes** drop-down:

- **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:

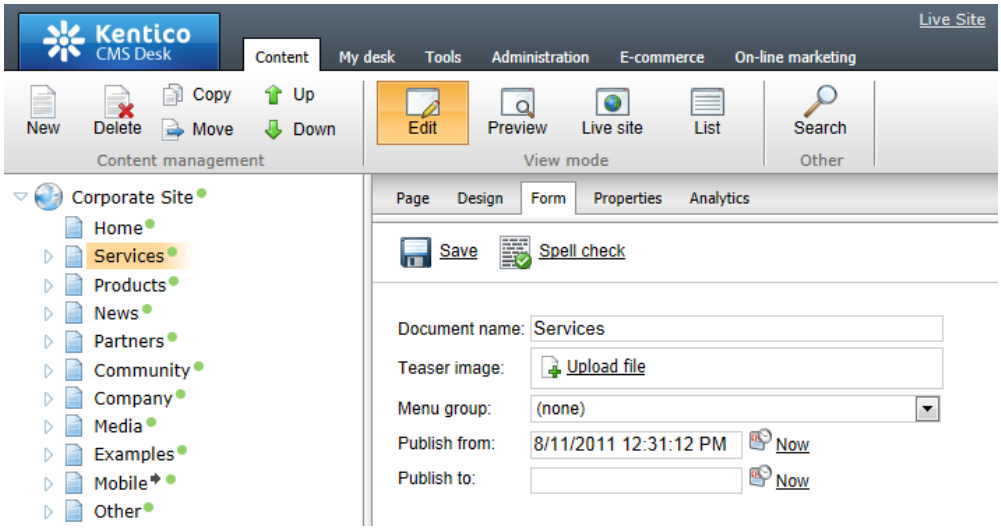
- **Search phrase**: the text that you are looking for; if you are searching using a Smart search index, you can use the syntax described here; if you are searching using SQL search, standard SQL syntax can be used
- **Search mode**: specifies how the search phrase will be used:
  - *Exact phrase* - 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.



4.12 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.

4.13 Code editor

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.

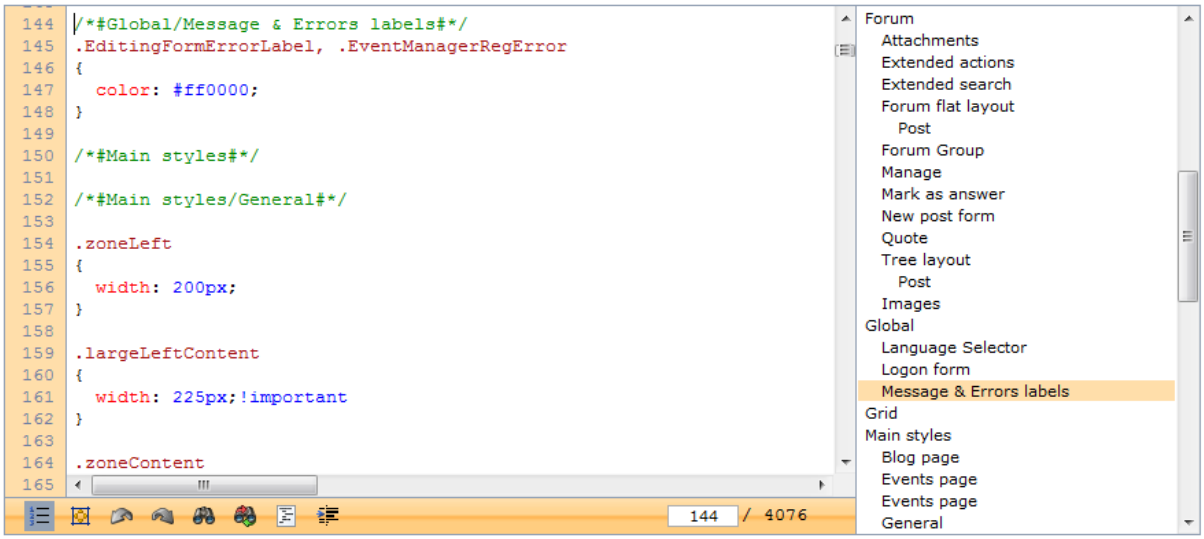
Highlighting ensures that text elements are displayed in different colors depending on the syntax of the given language. This greatly improves the readability of the code and makes it easier to spot and avoid mistakes. All languages commonly used for web development are supported, including HTML, ASP. NET, 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.
- **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.
- **Indent all** - increases 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. This panel can have a negative effect on the performance of the editor in some cases, so it may be disabled if the code contains a large number of lines. Please see the settings section below for more information.

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*.
- **CMSLineCountersThreshold** - if the displayed code contains more lines than the value specified for this key, the line counter will be disabled. This can be used to optimize editing performance, since the line counter may slow down the response time of the editor if there is a very large amount of displayed lines. The default value is *500*.

Sample key values:

```
<add key="CMSEnableSyntaxHighlighting" value="false" />
<add key="CMSEnableSyntaxHighlighting.CSS" value="false" />
<add key="CMSShowLineNumbers" value="true" />
```

```
<add key="CMSLineCountersThreshold" value="100" />
```

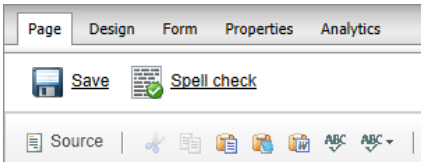
Customization of individual editors

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.

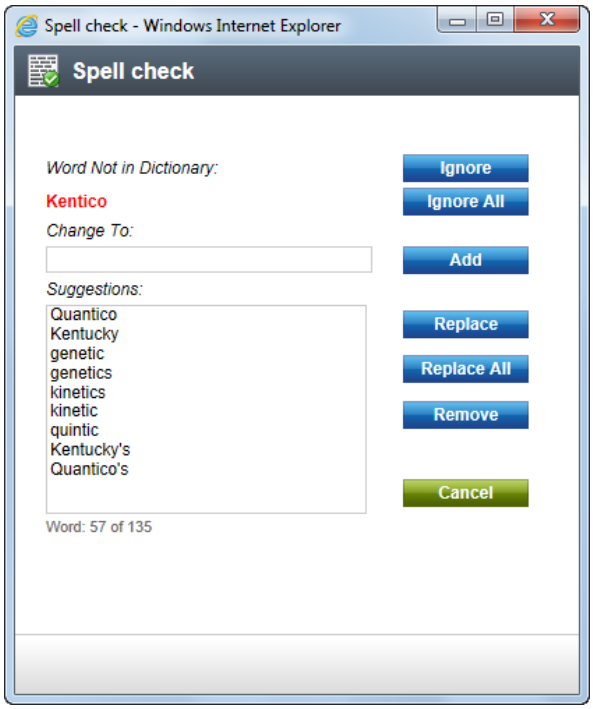
The editor is also integrated into the **Large text area** (**Input** type) Form control, which can be used if you wish to create a custom field somewhere in the UI, which will be used to enter a certain type of code.

4.14 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.

4.15 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.

4.16 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 the 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 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.

4.17 Content management internals and API

4.17.1 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

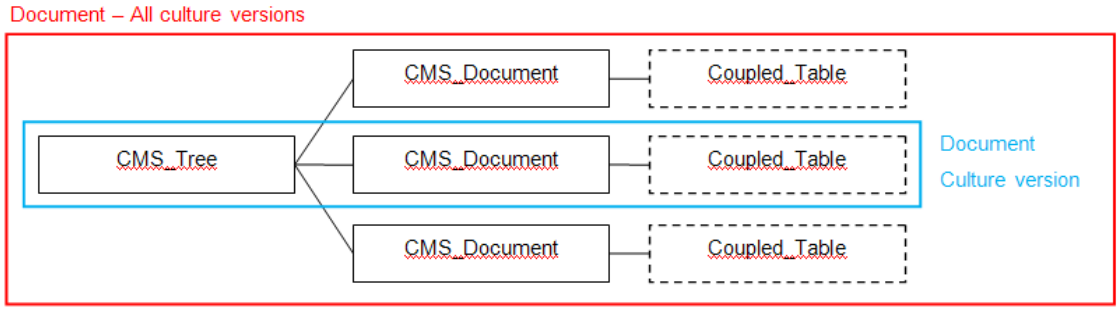
Advanced API examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all methods from the module classes, please refer to Kentico CMS API Reference.

4.17.2 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:

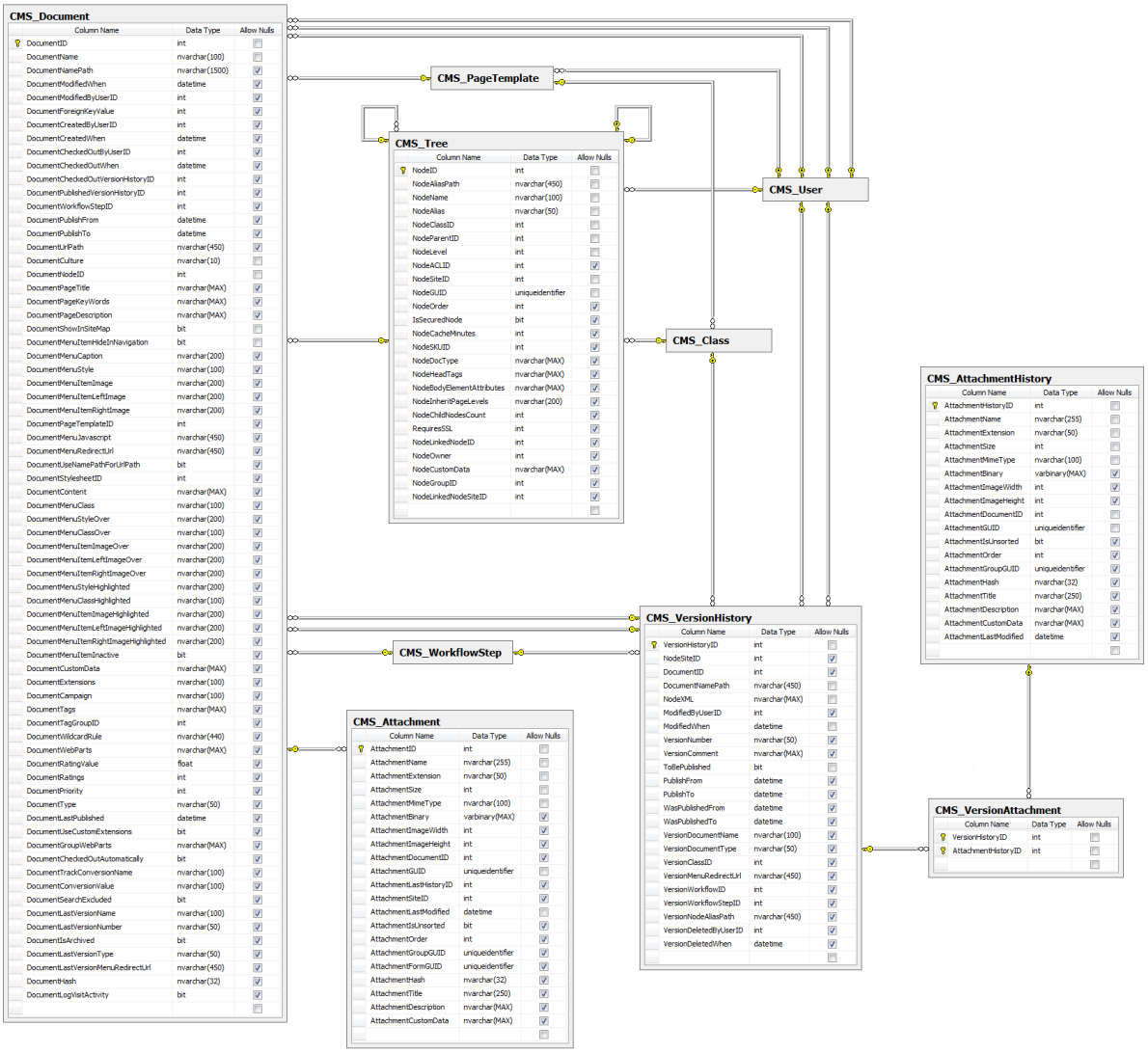
- **CMS_Tree** - a table with basic document data shared between different language versions of same document. This table determines the tree structure of the website document content. Table contains **one record for all culture versions** of the document. The table does not contain any versioned data.
- **CMS_Document** - a table with document data of specified language version of the document. Table 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:



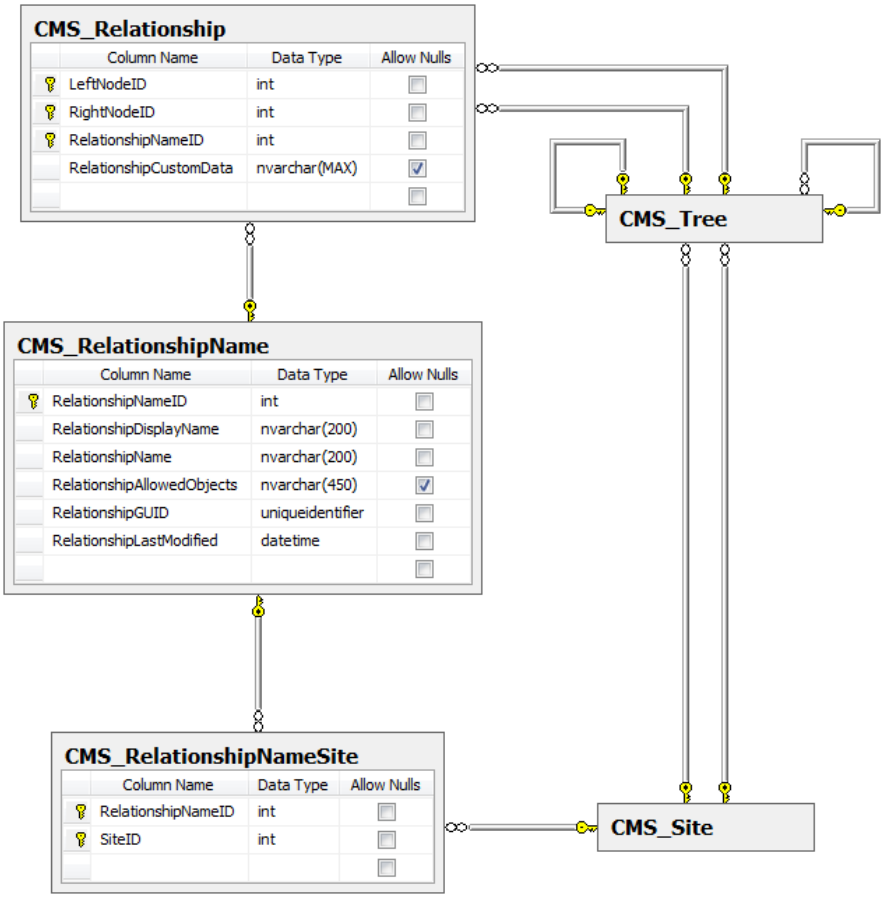
- **CMS_Attachment** - contains records representing document attachments.
- **CMS_VersionHistory** - contains records representing document versions. When a new document version is being created, it is only stored in this table. When it is published, respective records in the *CMS_Tree* and *CMS_Document* 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.



4.17.3 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.

CMS_Tree table API:

- **CMS.TreeEngine.TreeNode** - encapsulates data from the CMS_Tree and CMS_Document tables and the respective coupled tables.
- **CMS.TreeEngine.TreeProvider** - provides functionality for manipulation with tree nodes.

CMS_Relationship table API:

- **CMS.TreeEngine.RelationshipInfo** - represents a relationship between two documents.
- **CMS.TreeEngine.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.

4.17.4 API examples

4.17.4.1 Basics

4.17.4.1.1 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.TreeEngine;
using CMS.UIControls;
```

4.17.4.1.2 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");

        node.DocumentCulture = "de-de";

        // Insert into database
        node.InsertAsNewCultureVersion();

        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;
}
```

4.17.4.1.3 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;
}
```

4.17.4.1.4 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;
}
```

4.17.4.2 Advanced

4.17.4.2.1 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.TreeEngine;
using CMS.UIControls;
using CMS.WorkflowEngine;
```

4.17.4.2.2 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;
}
```

4.17.4.2.3 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;
}
```

4.17.4.2.4 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 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;
}
```

4.17.4.3 Document attachments

4.17.4.3.1 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.TreeEngine;
using CMS.FileManager;
using CMS.WorkflowEngine;
```

4.17.4.3.2 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 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;
}
```

4.17.4.3.3 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
            AttachmentManager manager = new AttachmentManager();
            manager.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;
}
```

4.17.4.4 Document relationships

4.17.4.4.1 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.TreeEngine;
```

4.17.4.4.2 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);
}
```

4.17.4.4.3 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;
}
```

4.17.4.5 Workflow basics

4.17.4.5.1 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.TreeEngine;
using CMS.UIControls;
using CMS.WorkflowEngine;
```

4.17.4.5.2 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 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;
}
```

4.17.4.5.3 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 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 document in the English culture
    TreeNode node = DocumentHelper.GetDocument(siteName, aliasPath, culture,
combineWithDefaultCulture, classNames, where, orderBy, maxRelativeLevel,
selectOnlyPublished, columns, tree);

    if (node != null)
    {
        // Translate its name
        node.DocumentName = "My new translation";
        node.SetValue("MenuItemName", "My new translation");

        node.DocumentCulture = "de-de";

        // Insert into database
        DocumentHelper.InsertNewCultureVersion(node, tree);
```

```
        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;
}
```

4.17.4.5.4 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 = new VersionManager(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 docu

```
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;
}
```

4.17.4.6 Workflow advanced

4.17.4.6.1 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.TreeEngine;
using CMS.UIControls;
using CMS.WorkflowEngine;
```

4.17.4.6.2 Sample documents and objects_2

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(step.StepWorkflowID);

            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;
}
```

4.17.4.6.3 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 = new WorkflowManager(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 = new VersionManager(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 = new WorkflowManager(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 = new VersionManager(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;
}
```

The following example reverts changes made to the document after the last check-out and checks it back in.

```
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 = new WorkflowManager(tree);

        // Make sure the document uses workflow
        WorkflowInfo workflow = workflowmanager.GetNodeWorkflow(node);

        if (workflow != null)
        {
            if (node.IsCheckedOut)
            {
                VersionManager versionmanager = new VersionManager(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;
}
```

4.17.4.6.4 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 = new WorkflowManager(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.CanUserApprove(node))
                {
                    // 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 = new WorkflowManager(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.CanUserApprove(node))
                {
                    // 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 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 = new WorkflowManager(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;
}
```

4.17.4.6.5 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 = new VersionManager(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 = new VersionManager(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 = new VersionManager(tree);

        // Destroy the version history
        versionManager.DestroyDocumentHistory(node.DocumentID);

        return true;
    }

    return false;
}
```

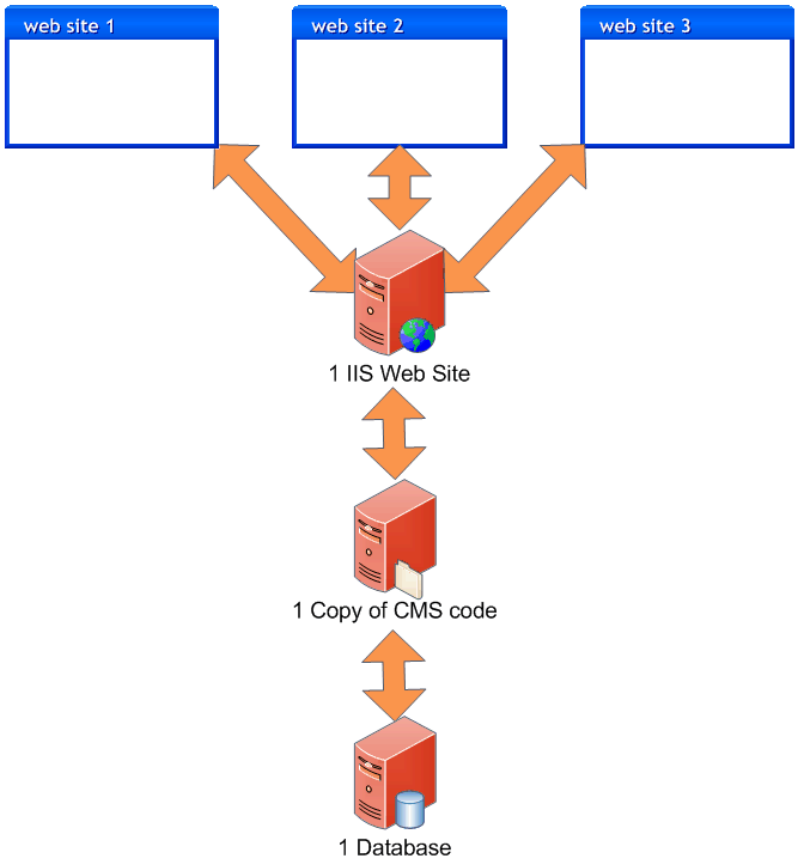
Part V

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 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 administration interface or 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 by administrator's mistake.

5.2 Managing sites

You can manage the websites in **Site Manager -> Sites** dialog.

Please see New site wizard for more details on creating a new website.

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 content. It can be changed using the **Change** button.
Default 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.
Site CSS stylesheet	Default CSS style sheet used for all pages unless they override the value with their own CSS stylesheet.
Editor CSS stylesheet	CSS style sheet 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.

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.
Default 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.
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. **Example**: Imagine that the main domain of your website is *www.domain.com* and you have *domain.com* defined as a Domain alias with the **Redirect URL** field set to: *{%protocol%}://www.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.com/example.aspx*, they will be redirected to *http://www.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.

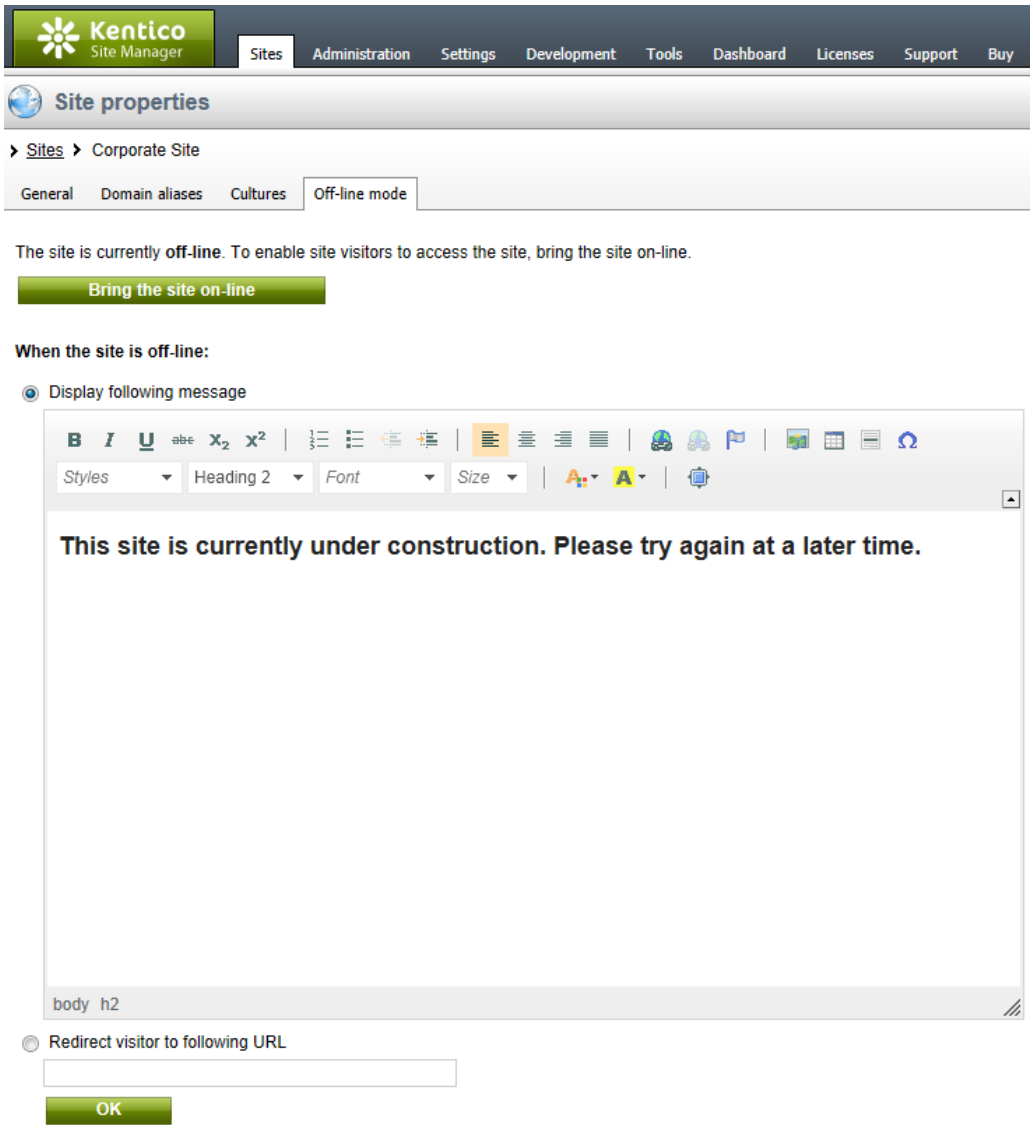
Off-line mode tab

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.

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.

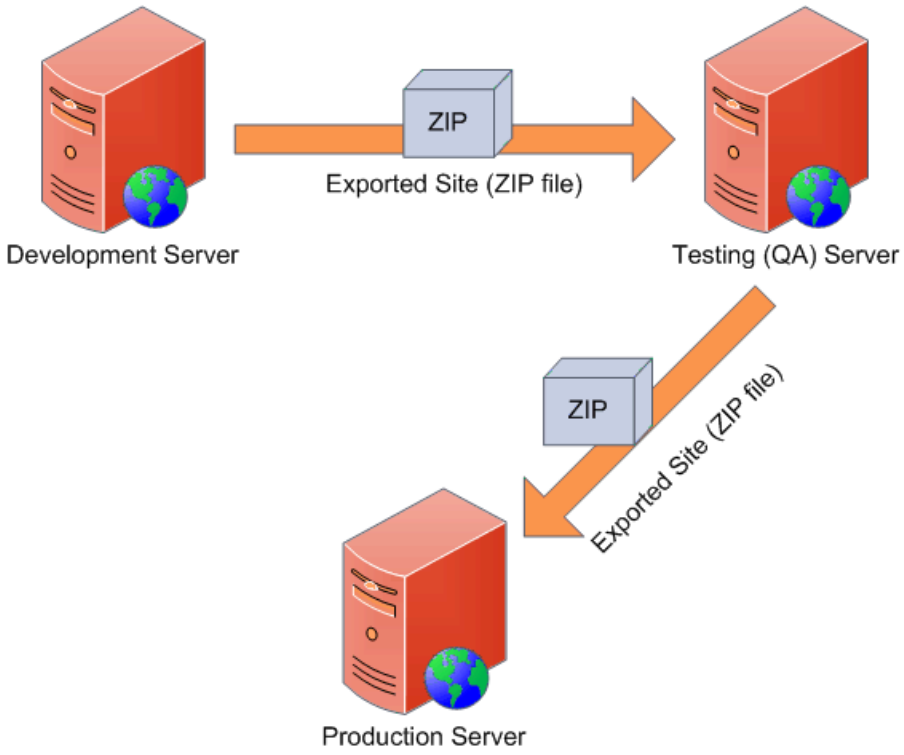
5.4 Creating a new site

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)

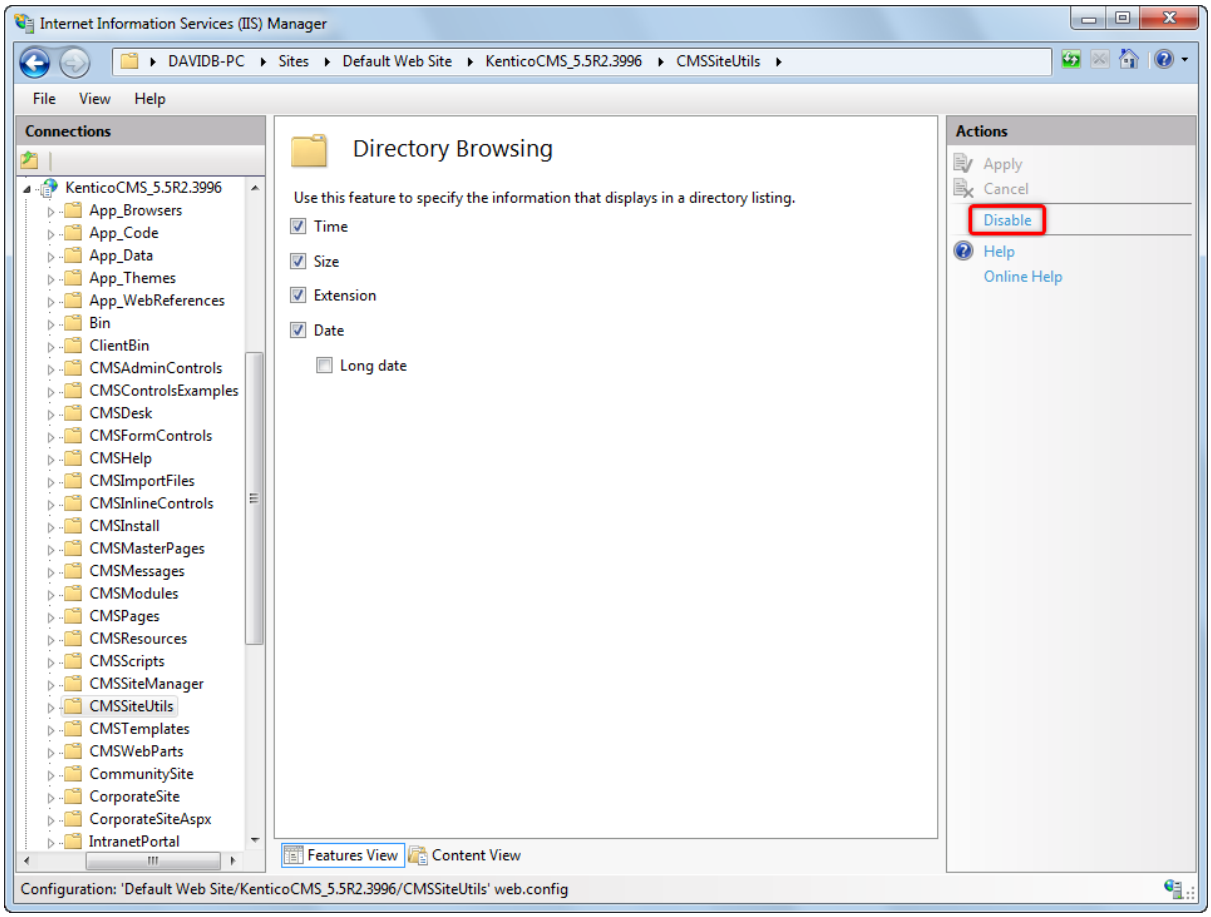
- **CMSGlobalFiles** (exports with any website, needs to be created manually; folder is exported if the 'Export global folders' option is checked in Step 2 of the export process)
- **CMSImportFiles**
- **CMSInlineControls** (all form controls selected in Step 2 of the export process are exported with 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**
- **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 Troubleshooting on W2008/IIS7

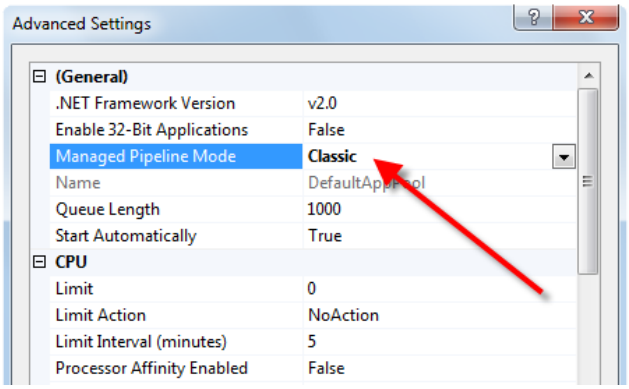
Kentico CMS fully supports **Microsoft Windows Server 2008** and **Internet Information Server 7**. Should you experience difficulties with import/export on IIS7, please go through the following steps:

1. Open **IIS manager**.

2. Open **Application pools** section.

3. Select application pool of your site and select **Advanced settings...** .

4. For property **Managed Pipeline Mode** select **Classic** instead of **Integrated**, or, you can select **Classic .NET AppPool** for your website.



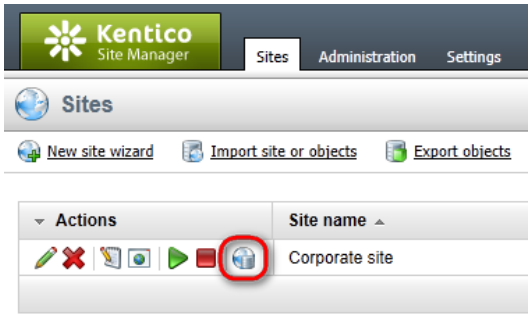
5.5.5 Export

5.5.5.1 Exporting a site

Kentico CMS allows you to export whole website including its settings and related objects (such as document types, workflows, web parts, page templates, etc.) into a single file 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 will start the export wizard.

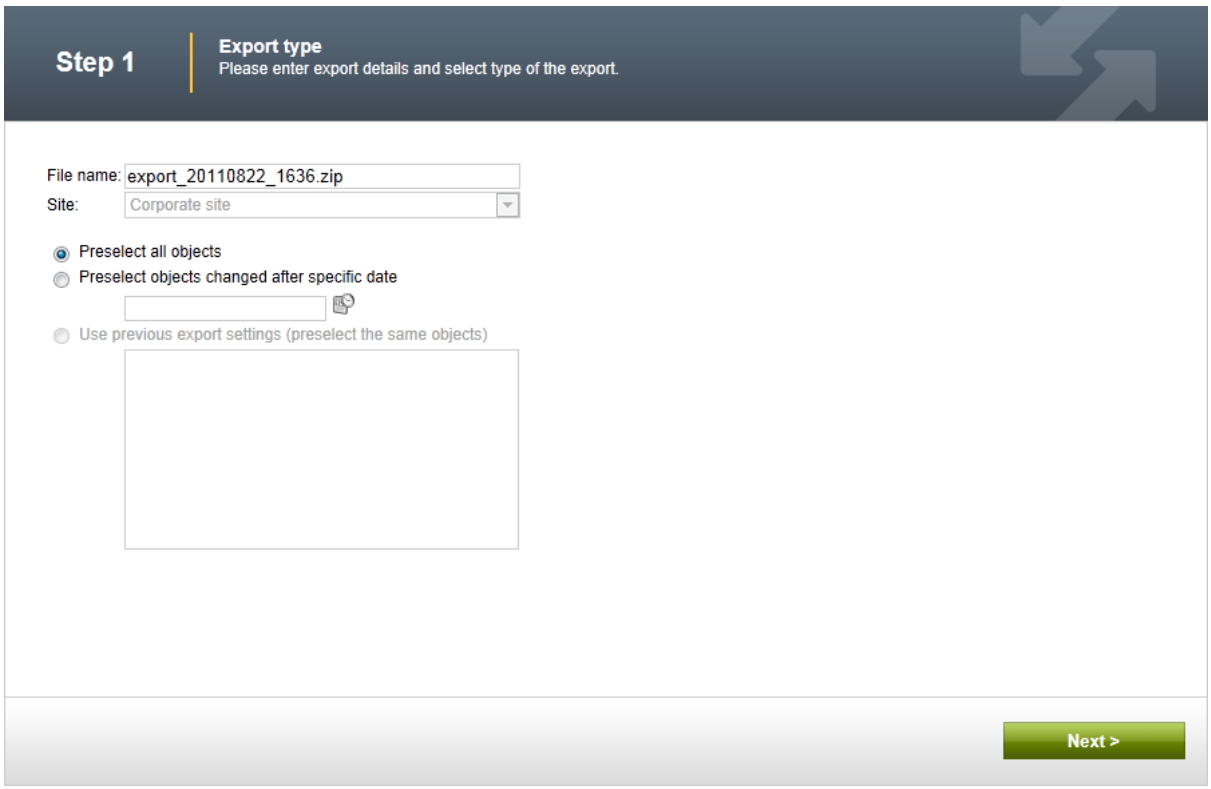


2. In the first step of the wizard, you have to fill in the name of the export package and choose type of

objects pre-selection:

File name	Name of the export package. A default name will be pre-filled and the package will be stored in *<web project>\CMSSiteUtils\Export*.
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 the **Next** button.

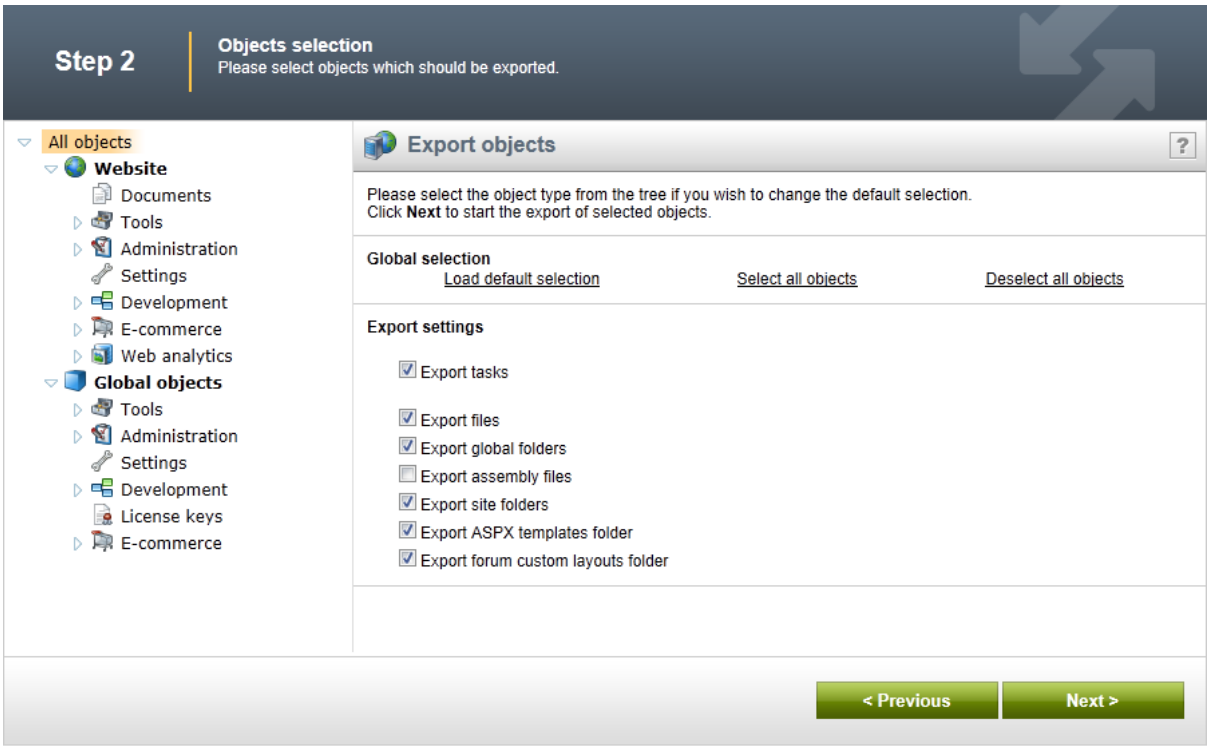


3. In **Step 2**, you can select which objects will be exported. The tree on the left represents object categories. These reflect the sections of the administration interface under that the objects can be found. The **Site** category contains objects related to the selected website. The **Global objects** category contains global objects that can be used by all sites.

By selecting a category, a set of check boxes appears in the right part of the screen, letting you select which objects will be exported. If you select the root of the tree (*All objects*), you will be offered with the following options:

Global selection	
Load default selection	If clicked, object preselection will be done based on your choice in Step

	1.
Select all objects	If clicked, all objects will be preselected.
Deselect all objects	If clicked, all objects will be deselected.
Export settings	
Export tasks	If enabled, delete tasks (incremental deployment) will be included in the package.
Export files	Some objects in the database are linked with physical files in the file system. If you check this check-box, these files will be exported along with the database objects.
Export global folders	If enabled, global files under the folders listed below will be exported: - <*web project*>*App_Code**Global* - <*web project*>*CMSGlobalFiles*
Export assembly files	If enabled, bound assembly files will be exported together with notification gateways, payment options, integration connectors, scheduled tasks and smart search indexes.
Export site folders	If enabled, files under the folders listed below will be exported: - <*web project*>*App_Code*\<*site code name*> - <*web project*>\<*site code name*>
Export ASPX templates folder	If enabled, the folder with ASPX page templates will be exported: - <*web project*>*CMSTemplates*
Export forum custom layouts folder	If enabled, folder with custom forum layouts will be exported: - <*webproject*>*CMSModules**Forums**Controls**Layouts**Custom*

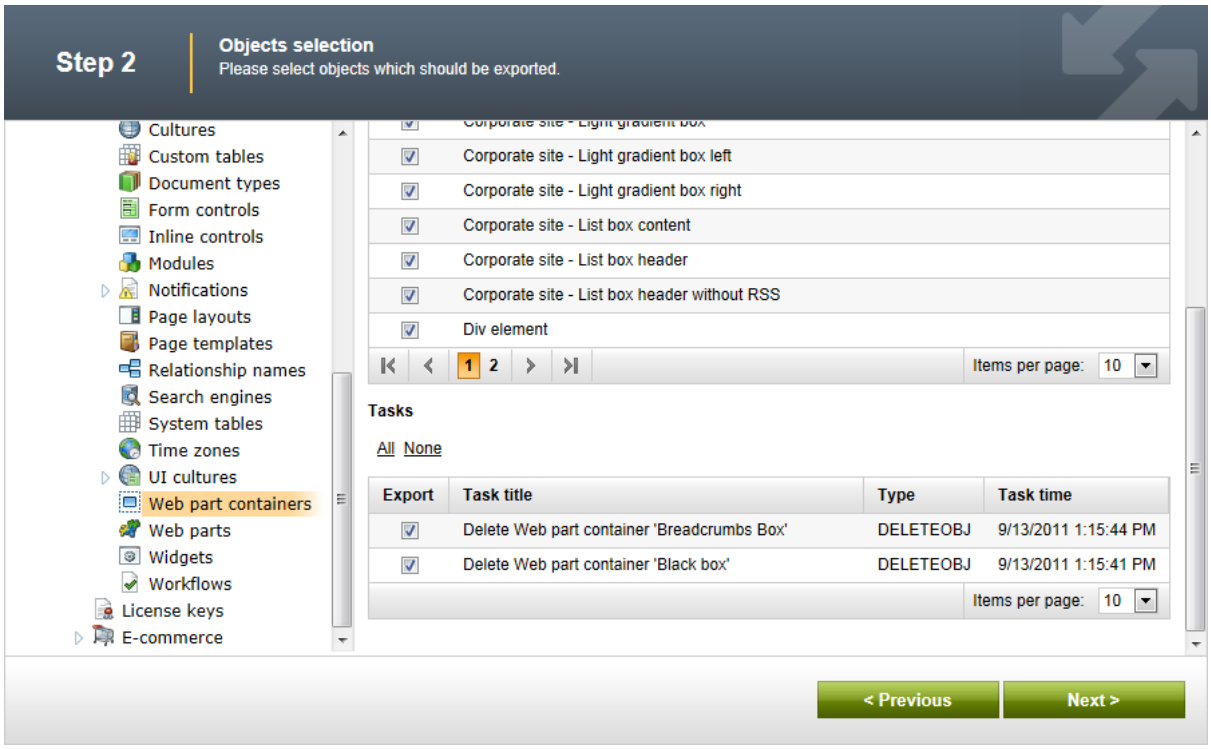


4. The following categories contain extra options to be set:

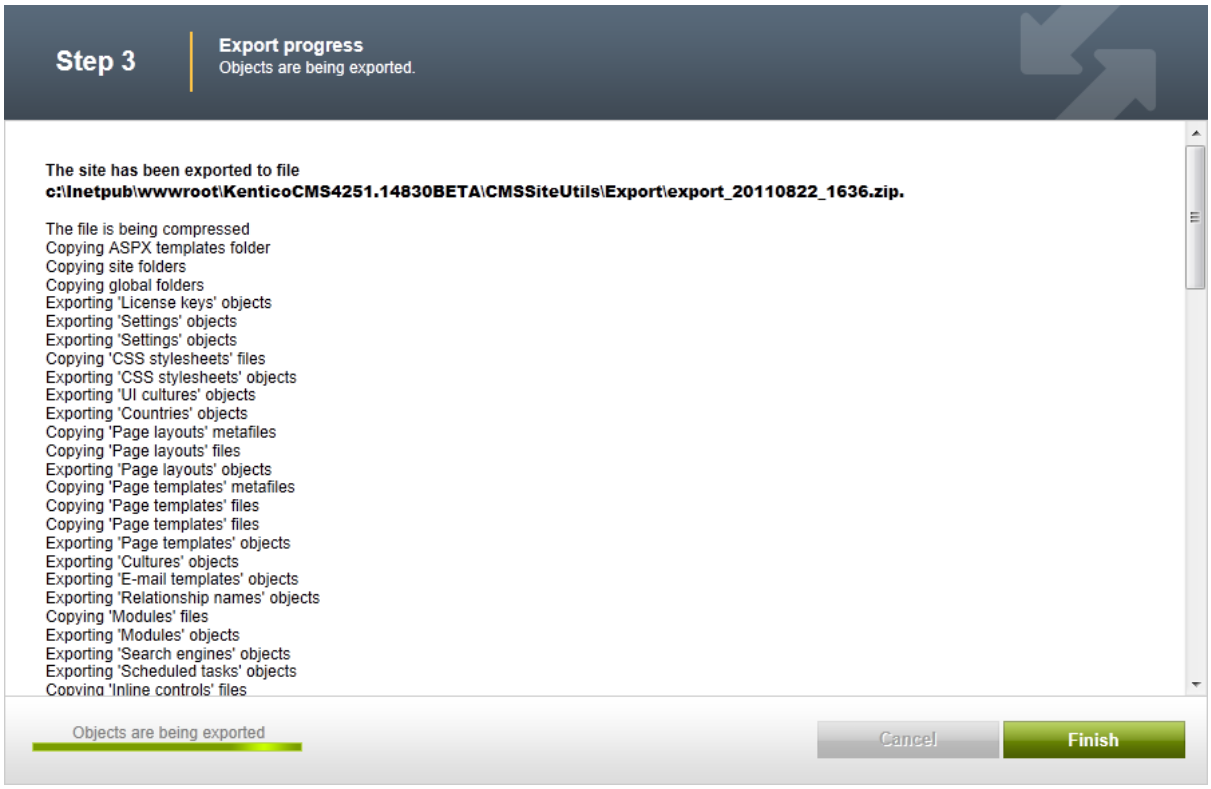
Custom tables	
Export custom table data	If checked, custom table records (the actual data stored in the tables) will be exported along with the selected custom tables.
Documents	
Export documents	If checked, documents will be exported
Export document histories	If checked, histories of all exported documents 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 the **Cancel** button 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 the **Finish** button.



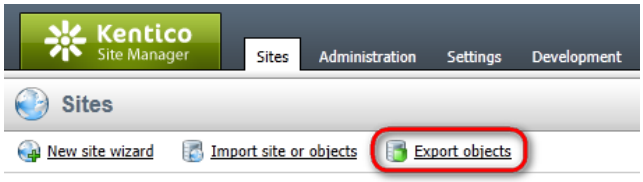
7. You will be redirected back to **Site manager** -> **Sites**. Now you can copy the exported package to the target installation of Kentico CMS, into the **<web project>\CMSSiteUtils\Import** folder and use the **Import site or objects** wizard described in the Importing a site or objects topic.

5.5.5.2 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.

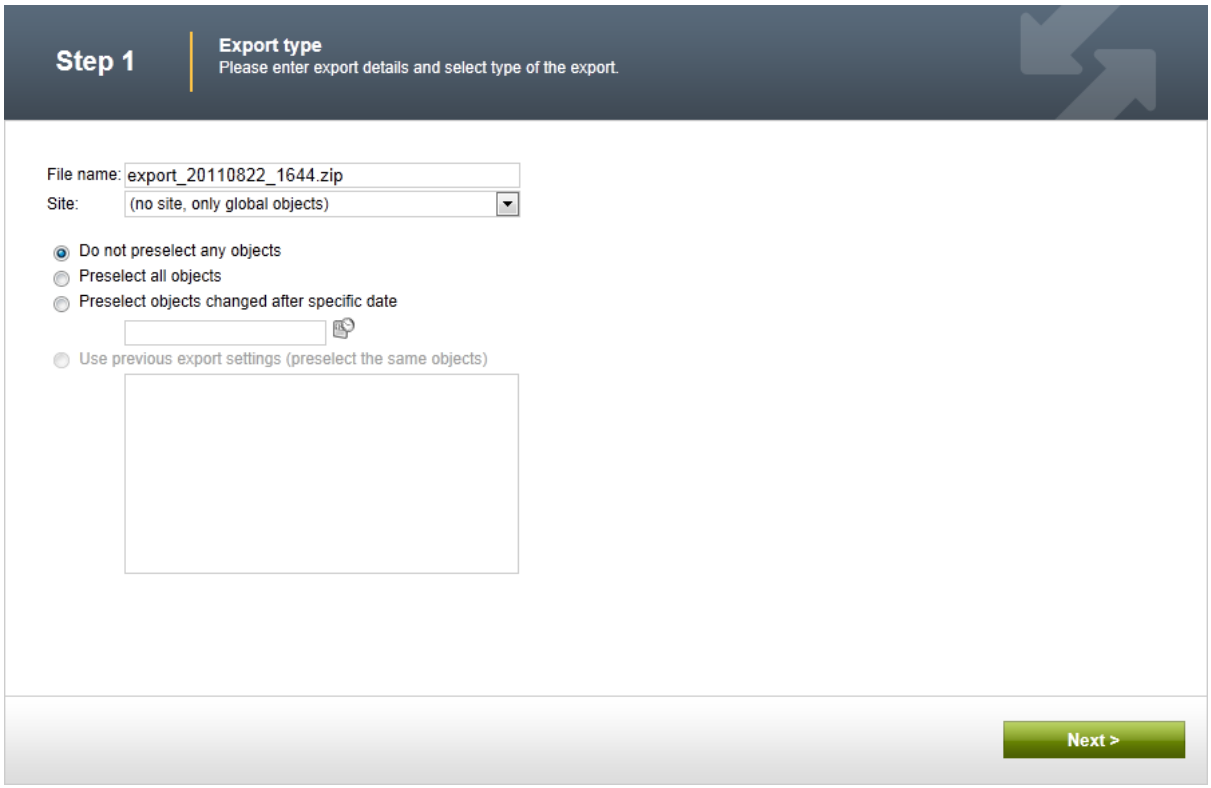


2. In the first step of the wizard, you have to fill in the name of the export package and choose type of objects pre-selection:

File name	Name of the export package. A default name will be pre-filled and the package will be stored in *<web project>\CMSSiteUtils\Export*.

Do not preselect any objects	If chosen, no objects will be preselected in the next step.
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.

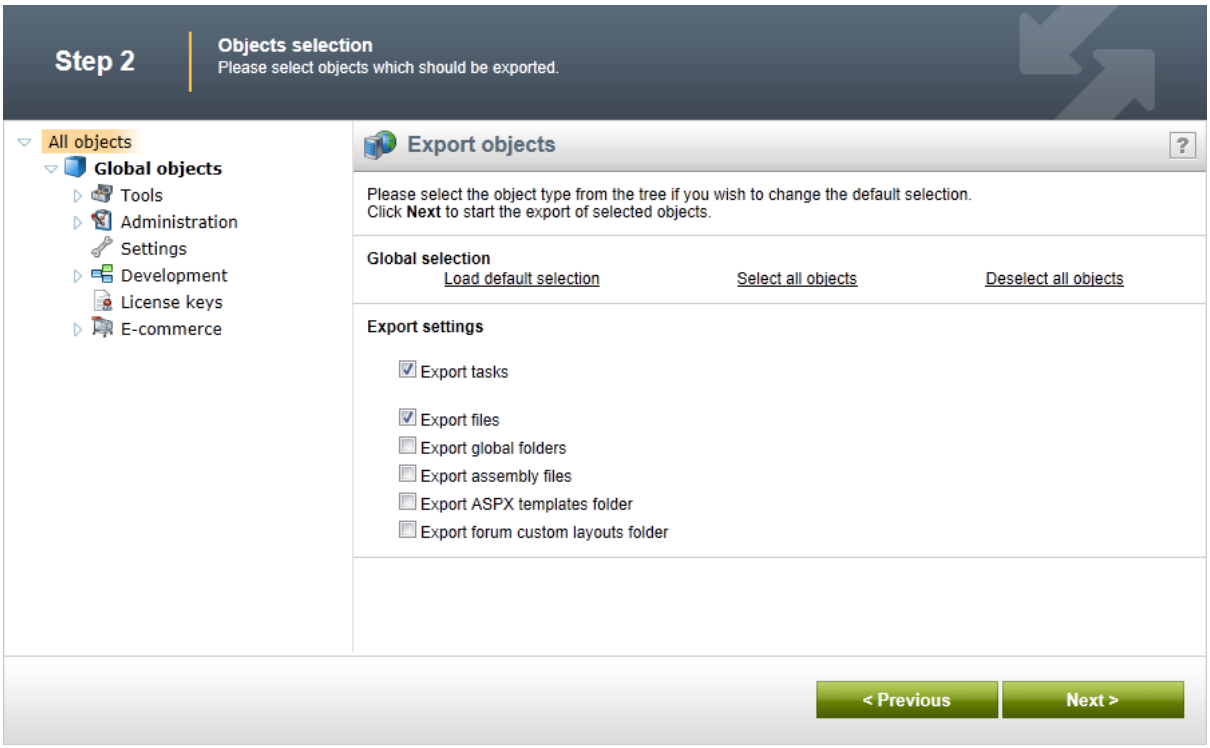
Select the option that suits your purposes and click the **Next** button.



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 that the objects can be found. By selecting a category, a set of check boxes appears in the right part of the screen, letting you select which objects will be exported. If you select the root of the tree (All objects), you will be offered with the following options:

Global selection	
Load default selection	If clicked, object preselection will be done based on your choice in Step 1.
Select all objects	If clicked, all objects will be preselected.
Deselect all objects	If clicked, all objects will be deselected.
Export settings	

Export tasks	If enabled, delete tasks (incremental deployment) will be included in the package.
Export files	Some objects in the database are linked with physical files in the file system. If you check this check-box, these files will be exported along with the database objects.
Export global folders	If enabled, global files under the folders listed below will be exported: - *<web project>\App_Code\Global* - *<web project>\CMSGlobalFiles*
Export assembly files	If enabled, bound assembly files will be exported together with notification gateways, payment options, integration connectors, scheduled tasks and smart search indexes.
Export ASPX templates folder	If enabled, the folder with ASPX page templates will be exported: - *<web project>\CMSTemplates*
Export forum custom layouts folder	If enabled, folder with custom forum layouts will be exported: - *<webproject>\CMSModules\Forums\Controls\Layouts\Custom*

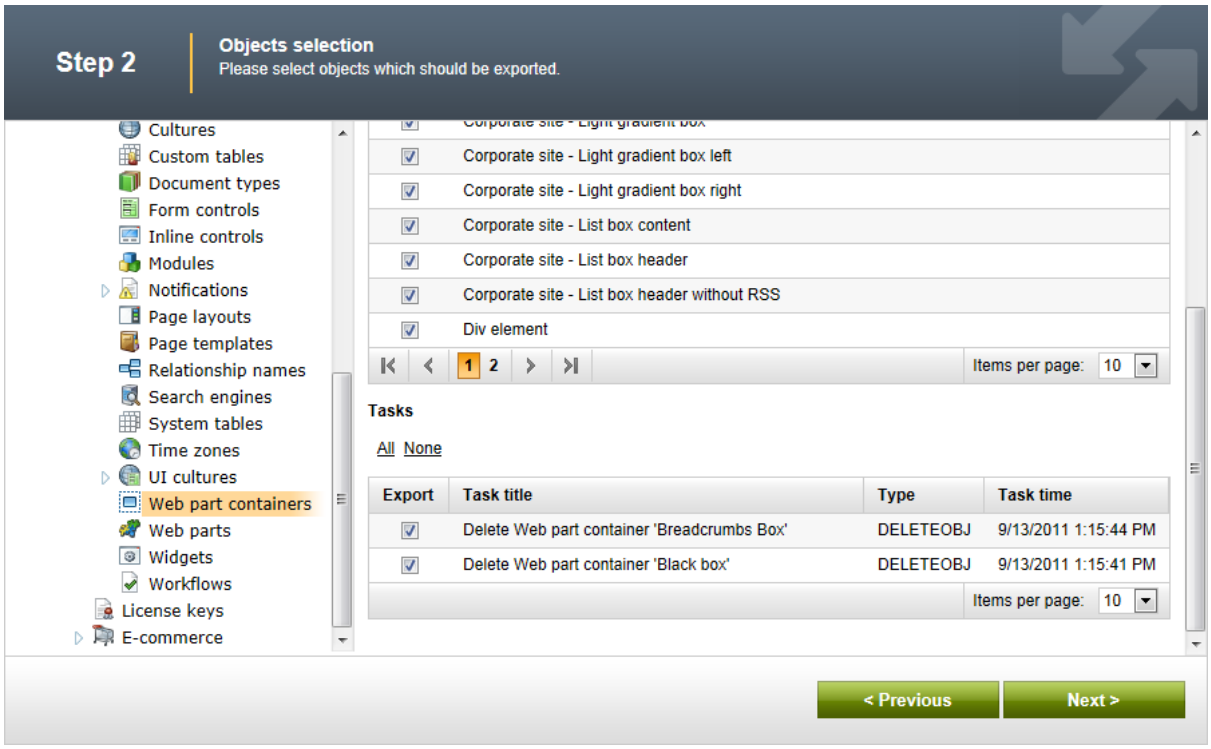


4. The following categories contain extra options to be set:

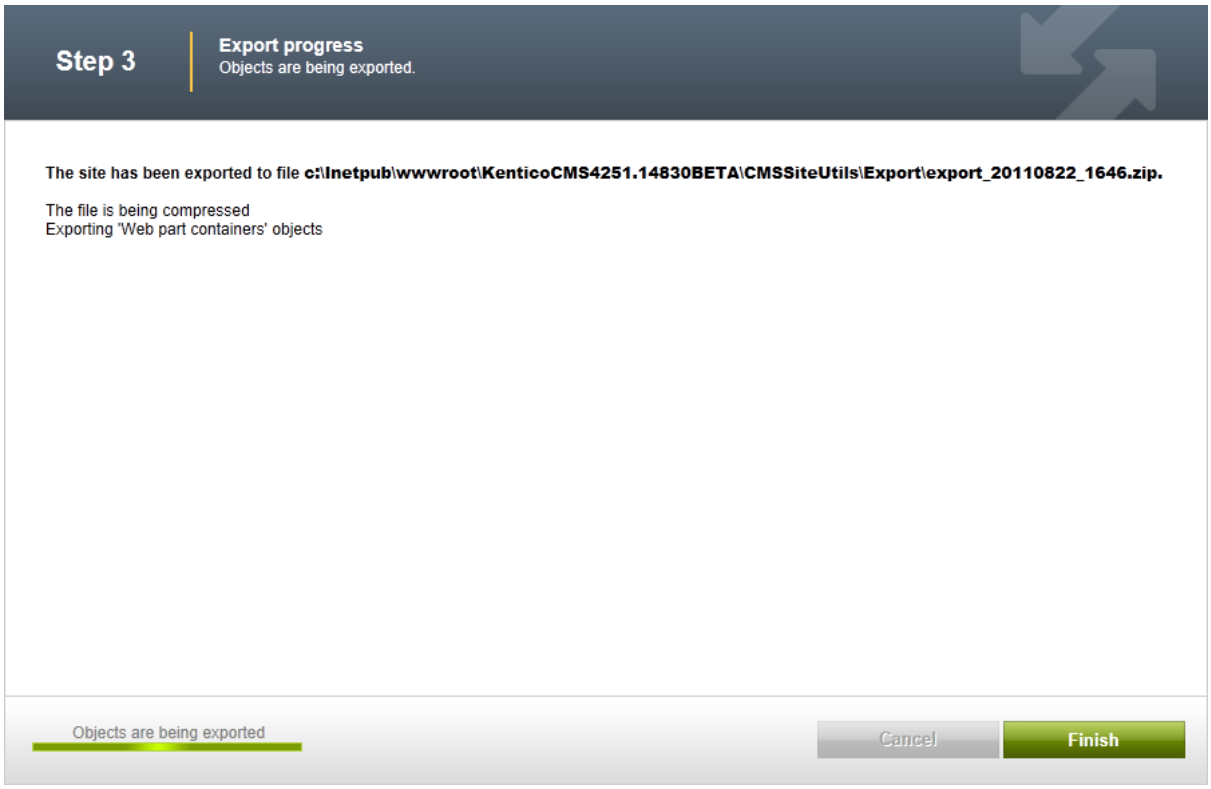
Custom tables	
Export custom table data	If checked, custom table records (the actual data stored in the tables) will be exported along with the selected custom tables.
Documents	

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 the **Cancel** button 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 the **Finish** button. You will be redirected back to **Site manager** -> **Sites**.



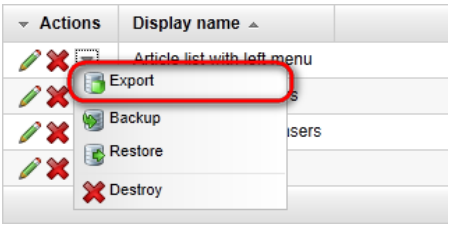
7. Now you can copy the exported file to the target installation of Kentico CMS, into the **<web project>\CMSSiteUtils\Import** folder and use the **Import site or objects** wizard described in the Importing a site or objects topic.

5.5.5.3 Exporting single objects

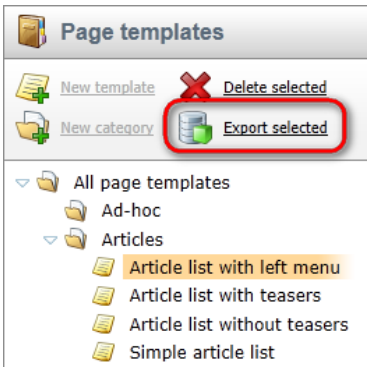
Some objects can also be exported separately as single object packages. This is useful when you want to quickly export only one or few objects apart from the rest of all other objects.

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.

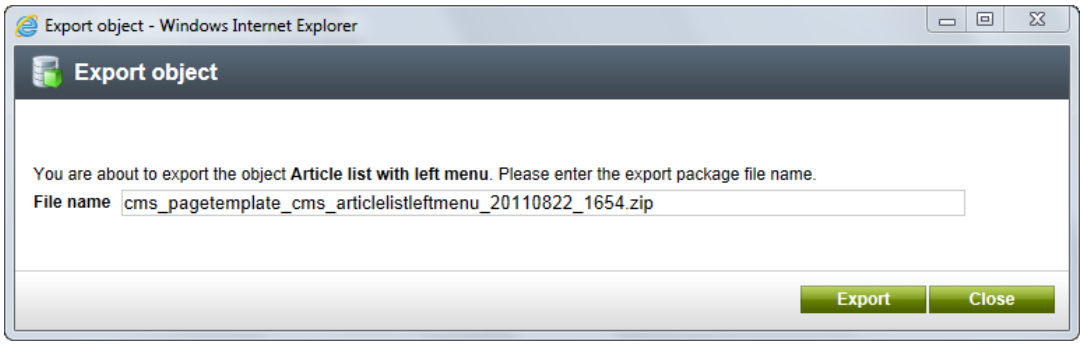
These objects can be found in the corresponding sections in **Site manager -> Development** or **Administration**. You can export an object by clicking the **Export object** ( ) icon.



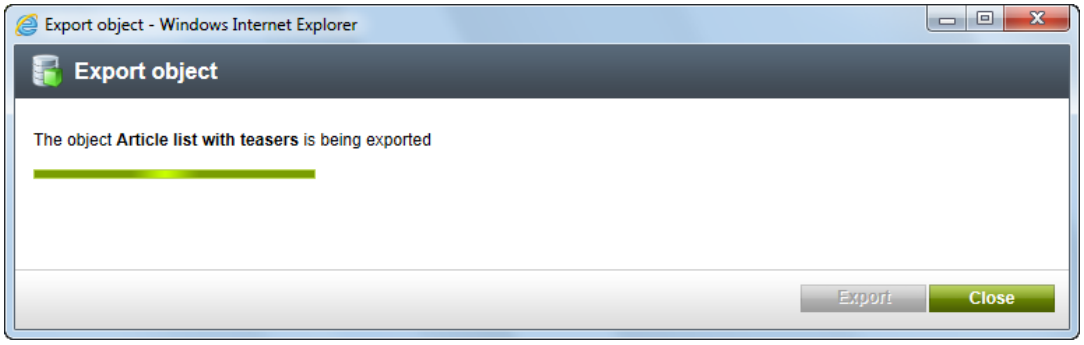
Or the **Export selected** link in case of Page templates and Web parts.



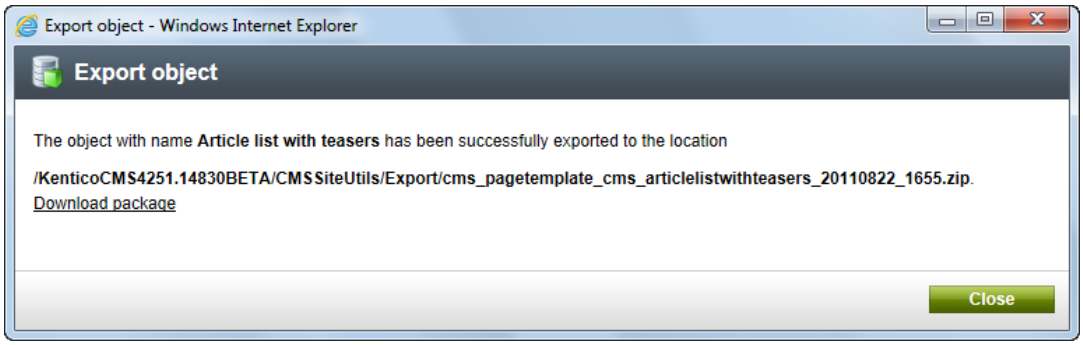
The following window will appear. Enter the name of the export file (default name will be pre-entered) and click **OK**.



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.



Finally, close the window by clicking the **Close** button.

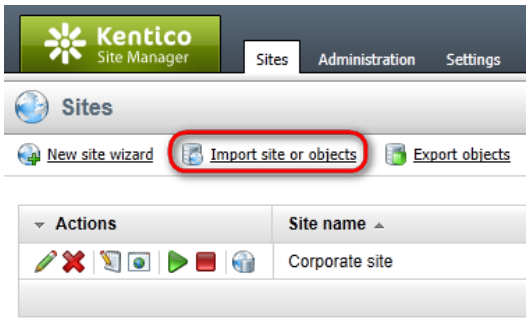
5.5.6 Import

5.5.6.1 Importing a site or objects

After you export some site or objects using the **Export site wizard** or **Export objects wizard**, you can import it using the **Import site or objects wizard**. Before you start the wizard, you need to copy your exported packages into the **<web project>\CMSSiteUtils\Import** folder.

Importing a site

1. Go to **Site Manager -> Sites** and click **Import site or objects**.



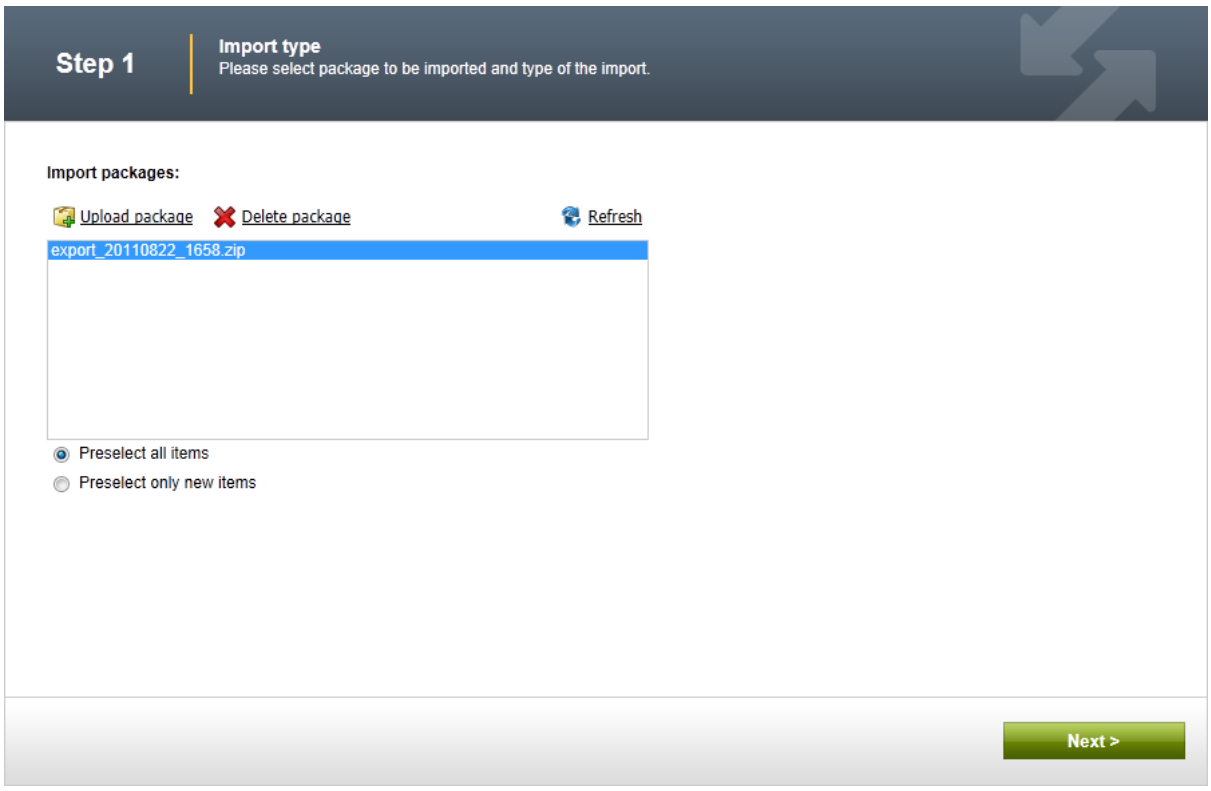
2. In the first step of the wizard, you need to choose the import package that you want to import. Packages that are already stored in the in *<web project>\CMSSiteUtils\Import* folder are listed in the **Import packages** listbox.

You can upload additional packages to the folder (and hence add them to the list) using the **Upload package** ( ) link. You can also add the package to the folder manually and click the **Refresh** ( ) link for it to get displayed in the list. Packages can be deleted (both from the list and from the file system) using the **Delete package** ( ) button.

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 the **Next** button.

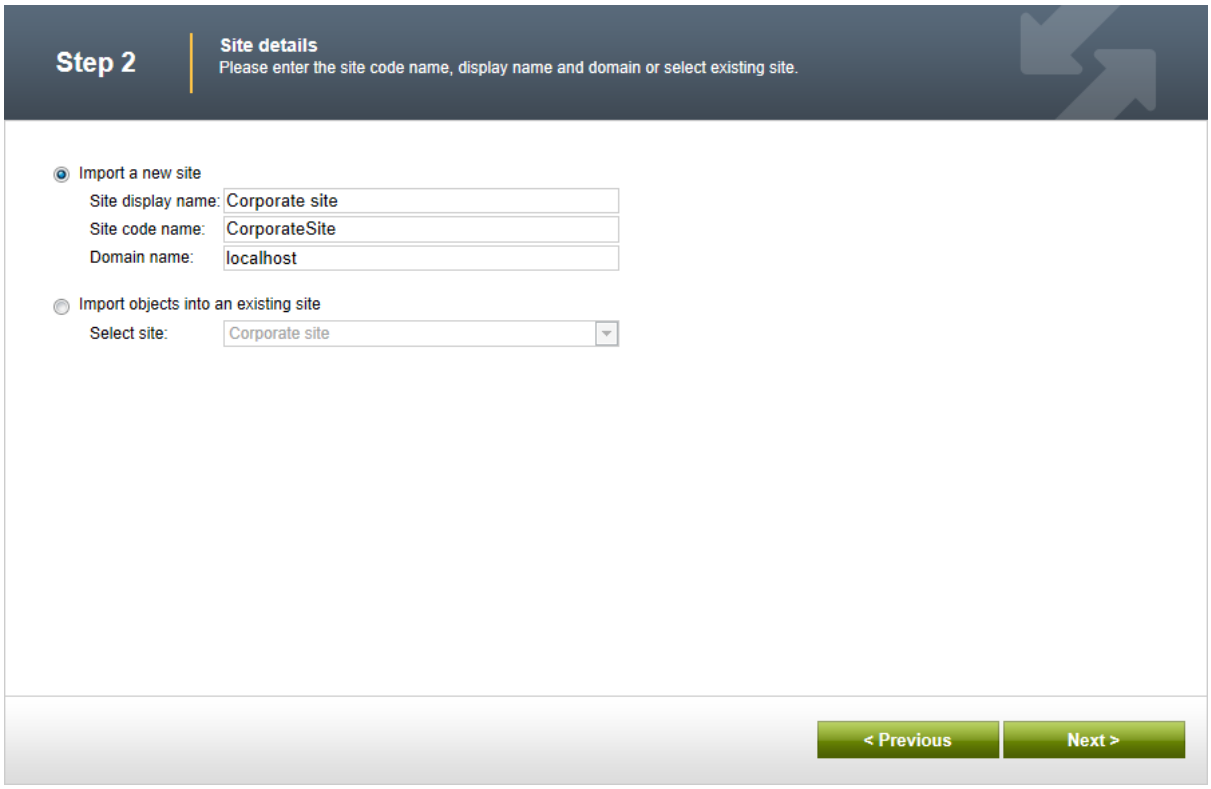


3. **Step 2** will be displayed only when you are importing a site package. In case that you are importing a package containing only global objects, Step 3 will be displayed instead.

You have the following two options in Step 2:

Import a new site	When chosen, a new site will be created based on the contents of the package. You have to enter the site's display name, code name and domain name.
Import objects into an existing site	When chosen, contents of the package will be imported into the site chosen by the check-box below.

Click the **Next** button to proceed to the next step.



4. In **Step 3**, you can 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 that the objects can be found. 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. Objects that already exist on the target server are marked with *. If you leave the check-box of such object checked, this existing object will be overwritten with the newly imported one.

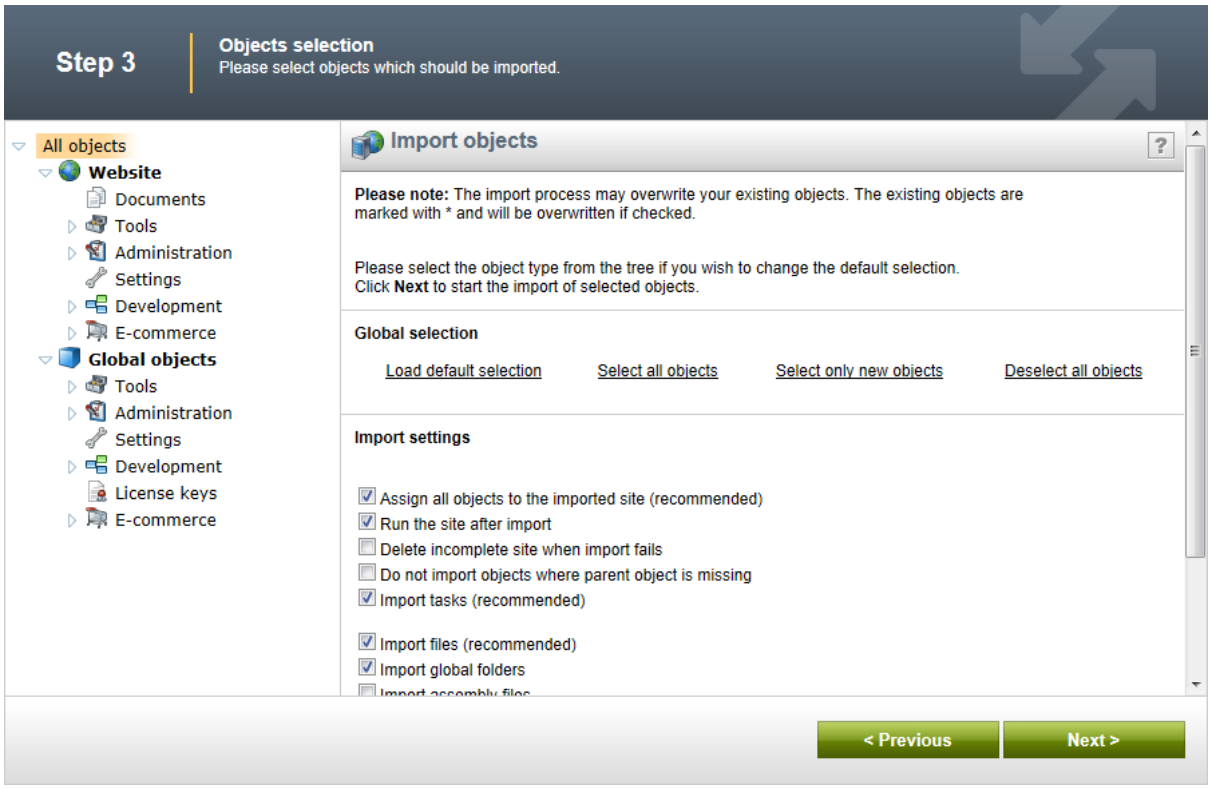
If you select the root of the tree (**Import objects**), a number of global choices will be offered to you:

Global selection	
Load default selection	If clicked, object preselection will be done based on your 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	
Update site definition	Displayed only when importing to an existing site. If enabled, all settings stored as a part of the site object will be updated with those contained in the package. These settings are contained in the *Site\cms_site.xml* file inside the export package.
Assign all objects to the imported site	If enabled, all imported site-related objects will be assigned to the imported site.

(recommended)	
Run the site after import	If enabled, the imported site will be run after the import process finishes.
Delete incomplete site when import fails	If enabled, incompletely imported site will be deleted when the import process fails.
Do not import objects where parent object is missing	If enabled, child objects whose parent objects are not found will be skipped and the import process will continue.
Import tasks (recommend)	If enabled, delete tasks (incremental deployment) included in the package will be performed.
Import files (recommended)	Some objects in the database are linked with physical files in the file system. If you check this check-box, such files contained in the package will be imported too.
Import global folders	If checked, global files originally found under the folders listed below will be imported: - *<web project>\App_Code\Global* - *<web project>\CMSGlobalFiles*
Import assembly files	If enabled, bound assembly files will be imported together with notification gateways, payment options, integration connectors, scheduled tasks and smart search indexes if they are included in the package.
Import site folders	If checked, files originally found under the folders listed below will be imported: - *<web project>\App_Code\<site code name>* - *<web project>\<site code name>*
Log staging synchronization tasks	Indicates if staging tasks reflecting changes made by the import should be logged on the target server (typically when you want to synchronize the imported data to other staging servers).
Log integration tasks	Indicates if outgoing integration tasks should be logged for changes made by the import (typically when you want to transfer the imported data to a system connected via System integration bus).
Overwrite system queries	If checked, system queries will be overwritten by those contained in the package. See the info box below for more details.

Overwrite system queries

This option is only displayed when importing from version older than the current one. If checked, all queries from the package will be imported and will overwrite the current ones; if the **package contains your own custom queries** that you added to the system, it is necessary to have this option enabled.



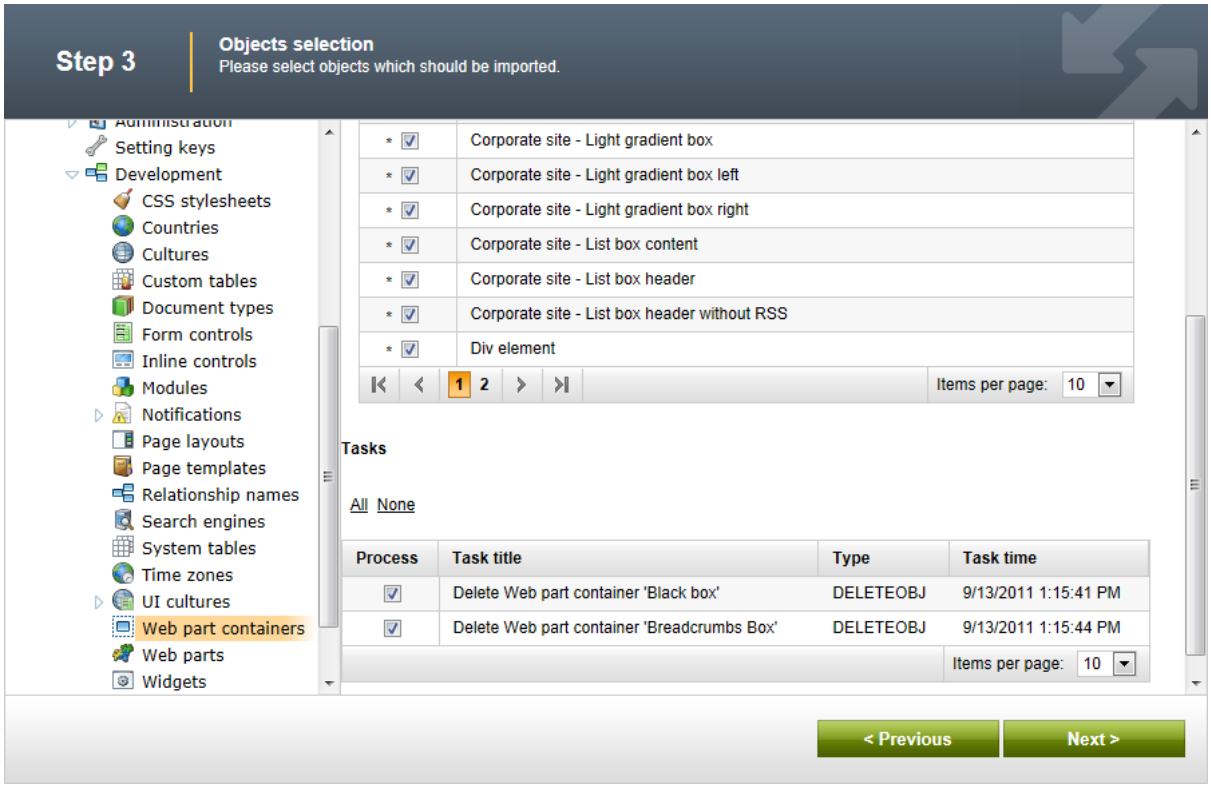
5. The following categories contain extra options to be set:

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

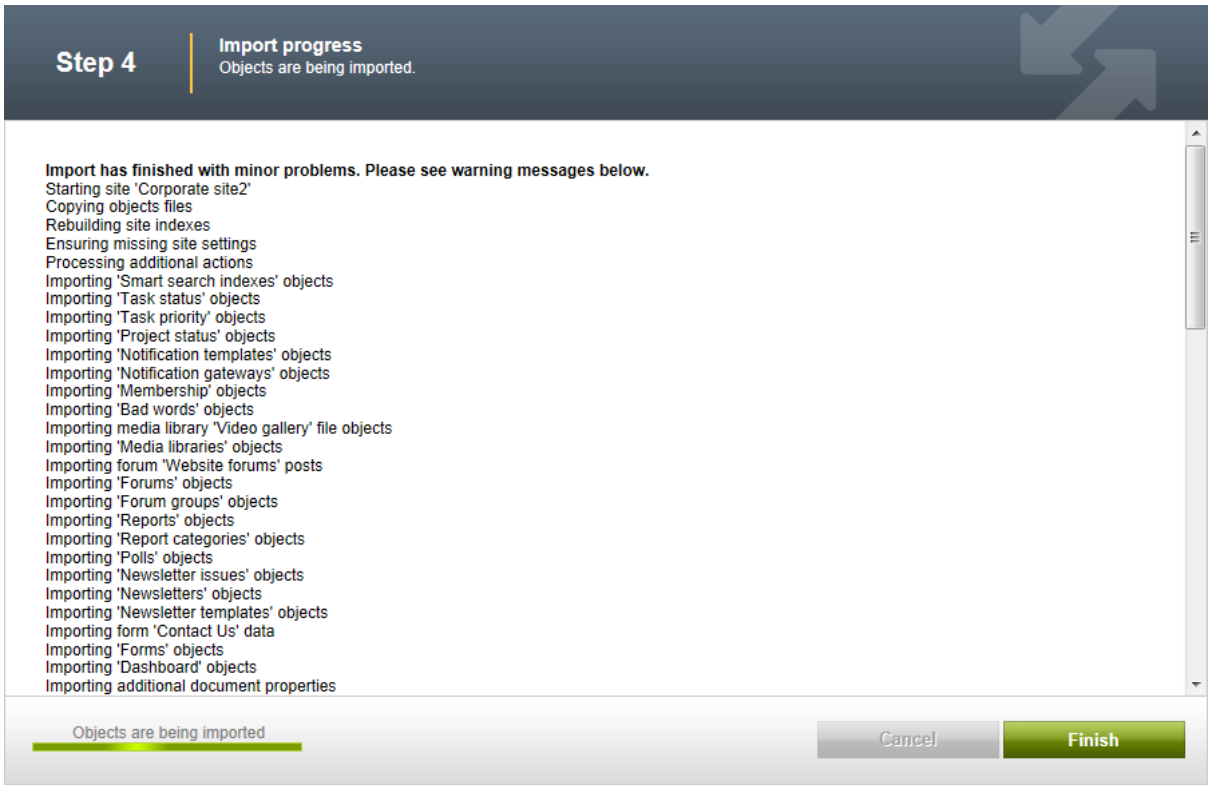
	imported events.
Import all user personalization	If checked, user personalization of all users will be imported, even for users who are not selected to be imported. If you are importing to an existing site, this option is de-selected by default.
Forms	
Import form data	If enabled, forms data included 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 a 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 de-selected 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 (just as the two Web part containers in the screenshot below). If you leave the check-boxes checked, these objects will be deleted on the target server.



7. Finally, click **Next** to execute the import process. An import log will be displayed, showing the progress of importing (you can abort importing by clicking the **Cancel** button any time). Click the **Finish** button. You will be redirected back to **Site manager** -> **Sites**, where the newly imported site will be running.



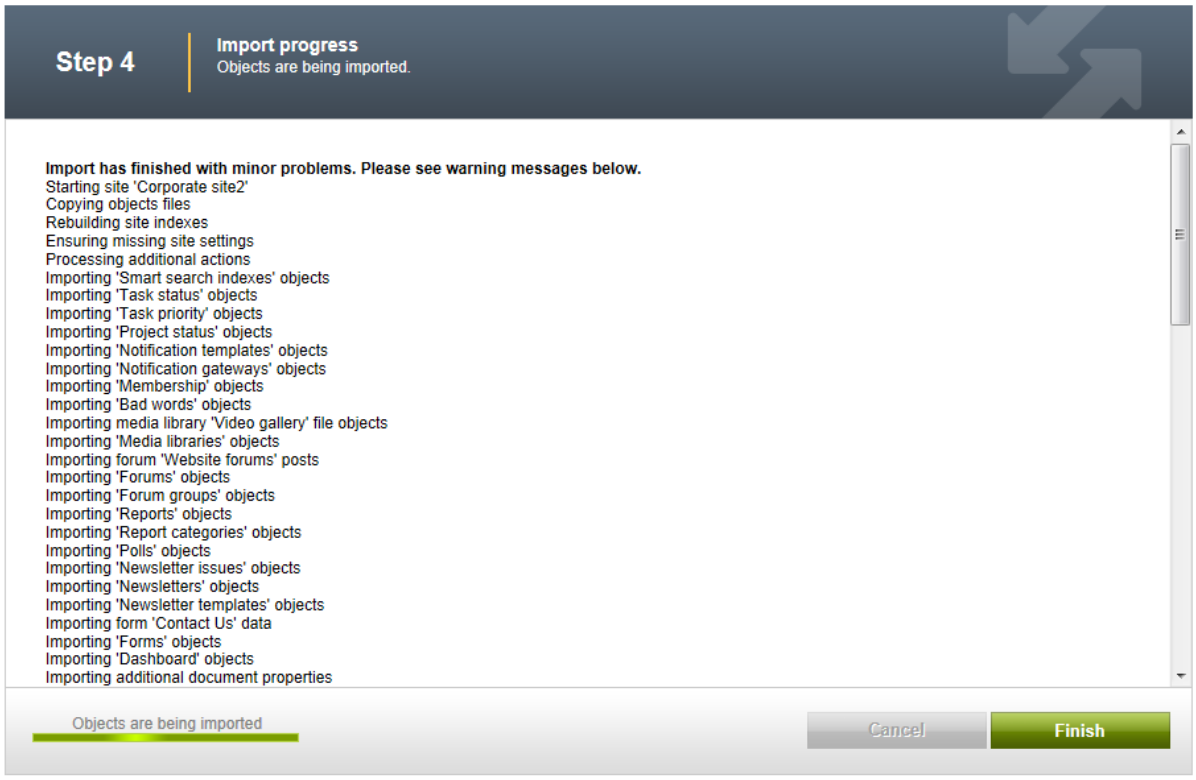
Please note

Packages from different versions of Kentico CMS have different structure. When importing packages from an older version of Kentico CMS to a newer one, structure of the package is always converted to the newer format automatically.

Please pay special attention when importing **Form user control**, **Inline control** and **Web part** objects from older packages. If possible, avoid overwriting your current objects of these types with objects from older packages, 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 that already run on your server, you may get an error message at the end of the import:



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 case, you need to go to the **Sites** section, change the domain name or domain alias and start the new website manually using the **Start site** button.

Application restart

At the end of the import process, you may get the following error message:

"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."

If so, you will have to finish the import process manually:

1. Open the import package and extract all contents of the *<package>\Data\Files* folder.

2. Remove the *.export* extension from 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 DB). 5. Copy all contents of each folder named as an object type (e.g. *cms_avatar*, *cms_documenttype*, *forums_forum*, etc.) to the root of the web project.

5.5.6.2 Importing from web site to web application projects

Due to differences in web site and web application project structures, import from one to another cannot be completed automatically. You need to take additional manual steps after the standard import procedure has completed in order for the imported site to run properly in the web application environment.

1. All physical files that belong to the imported package have been placed into the *~/App_Data/CMSTemp/ImportExport/Files* folder. Copy the contents of the folder into the root of your web application project.

Important!

It is highly recommended to make a backup of your project before overwriting any files!

2. Open the web project solution file (*WebApp.sln*) in Visual Studio from the web project directory.

3. If you added files that didn't exist in the project before the import, they need to be included in the project first. Click **Project** in the main menu and make sure the **Show all files** option is turned **on**. Locate the newly added files in the Solution Explorer. Select the files you wish to include one-by-one while holding the **Ctrl** key, then click **Project > Include In Project** in the main menu.

4. In the Solution Explorer, right-click the project node (CMSApp) and select **Convert To Web Application**.

5. Rebuild the solution.

5.5.7 Export and import internals and API

5.5.7.1 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.

5.5.7.2 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.

5.5.7.3 API examples

5.5.7.3.1 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.TreeEngine;
using CMS.WorkflowEngine;
```

5.5.7.3.2 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;
}
```

5.5.7.3.3 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;
}
```

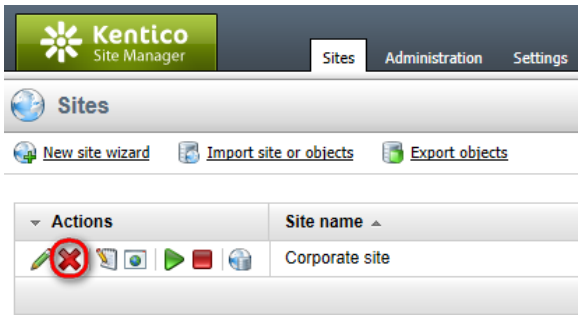
The following example exports the website imported by the first example on the previous page into a new export package.

```
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**.

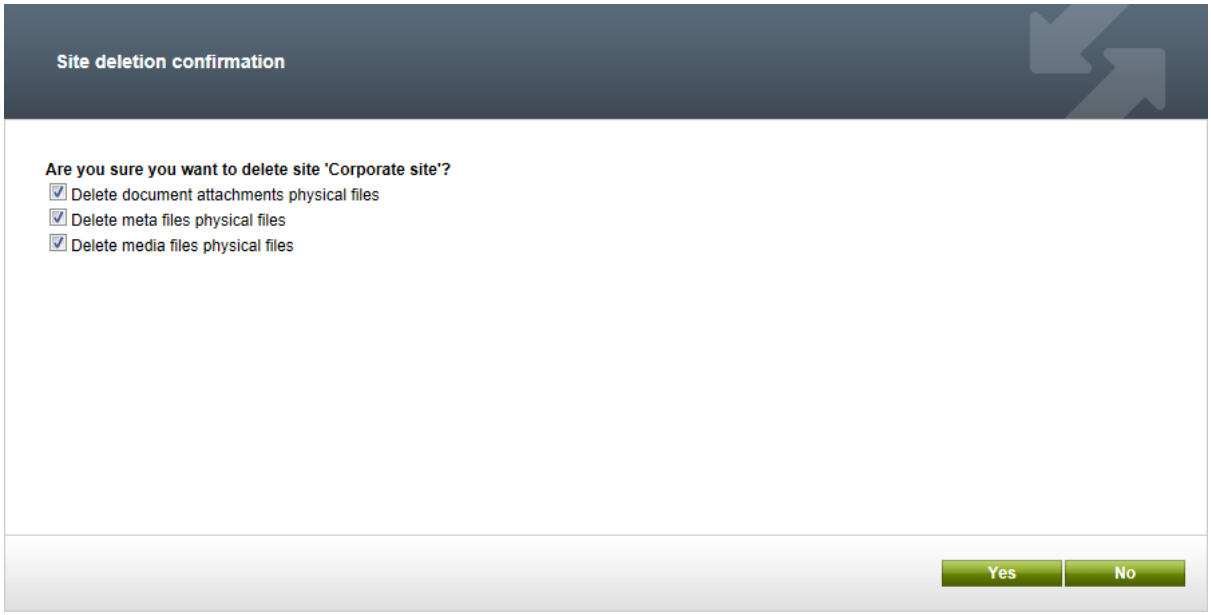
1. Clicking the **Delete** (✖) icon of the site that you want to delete.



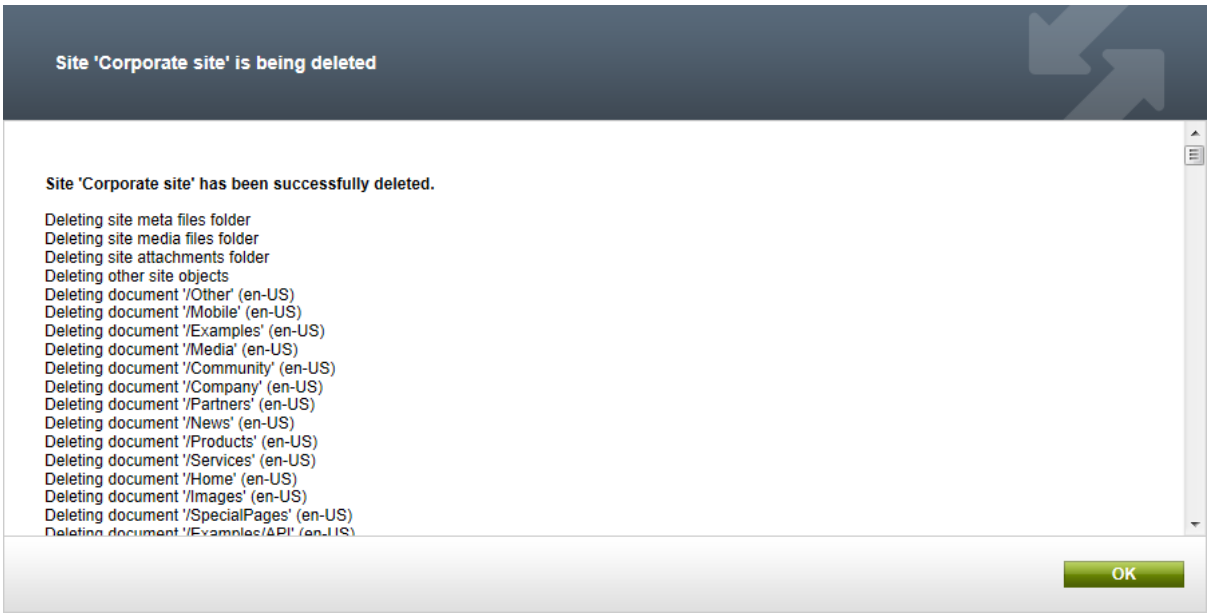
2. In the **Site deletion confirmation** dialog, you can select the following options:

- **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.

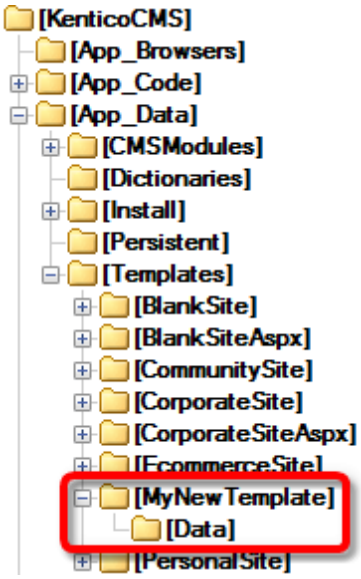


5.7 Creating web templates

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:

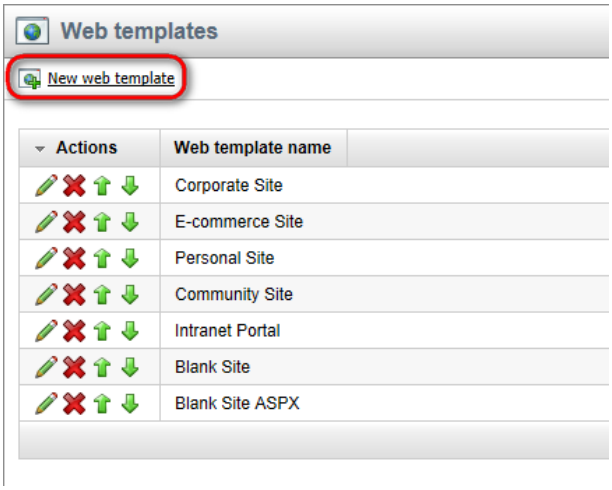
1. Export your site. For a step-by-step tutorial on how to export a site, please refer to the Exporting a site chapter of this guide.

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 the **New web template** link.



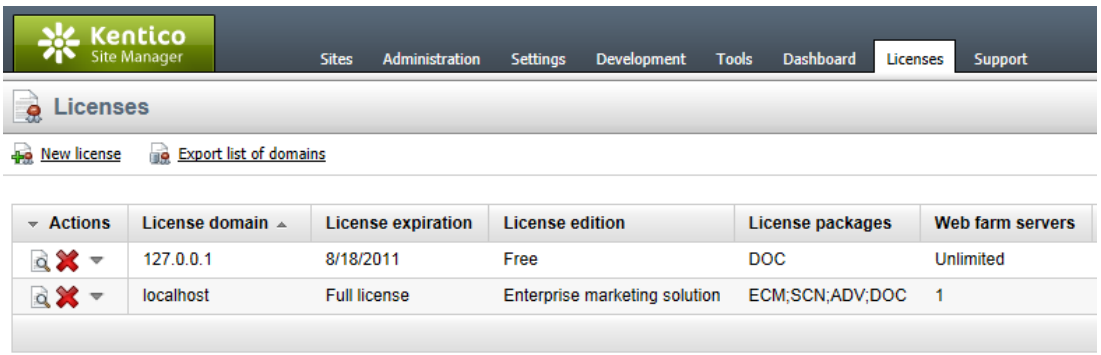
5. Enter the following details:

- **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 the **Export list of domains** ( ) link to export your domains. The export package will be saved in ~*CMSSiteUtils**Export* and can be used to import the licenses on another instance of Kentico CMS.

How licensing works

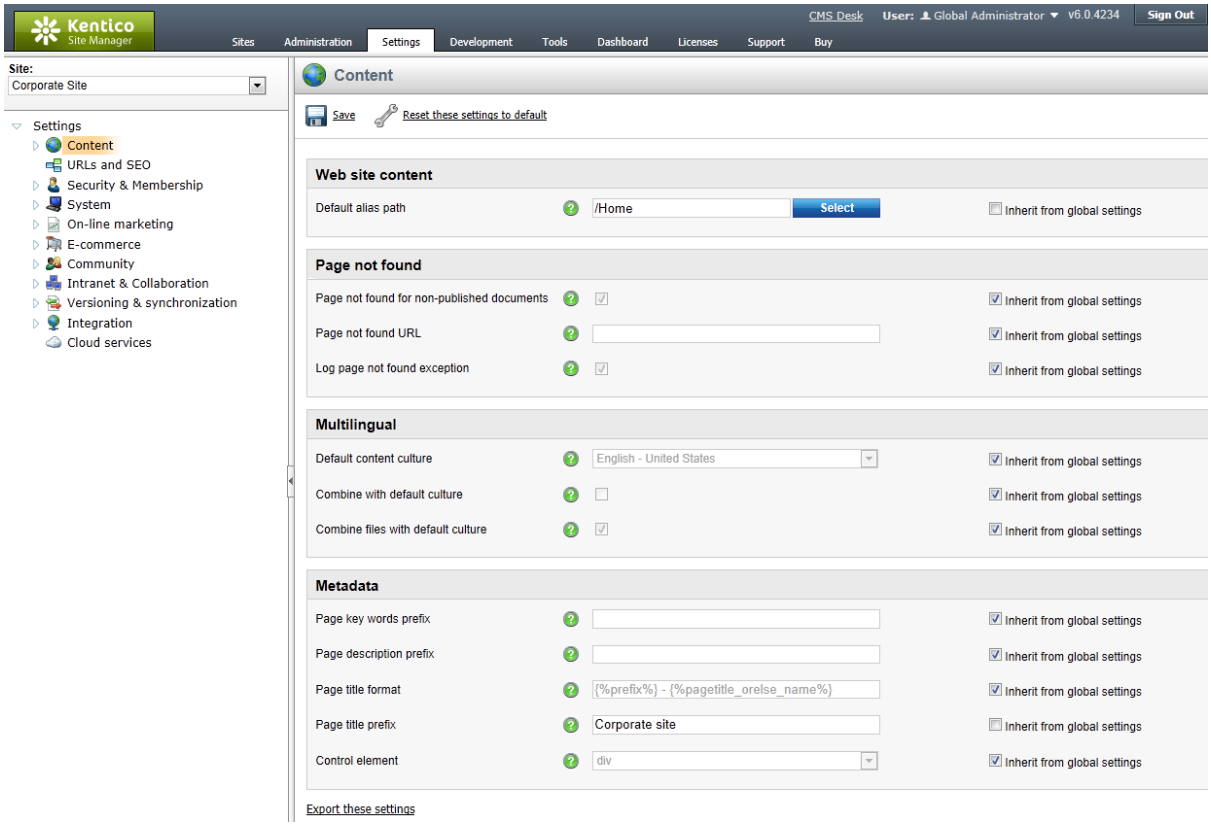
If you're running website on domain example.com, you need a single license key that will also work for:

- http://example.com
- https://example.com
- http://www.example.com
- https://www.example.com
- http://localhost
- http://127.0.0.1

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**

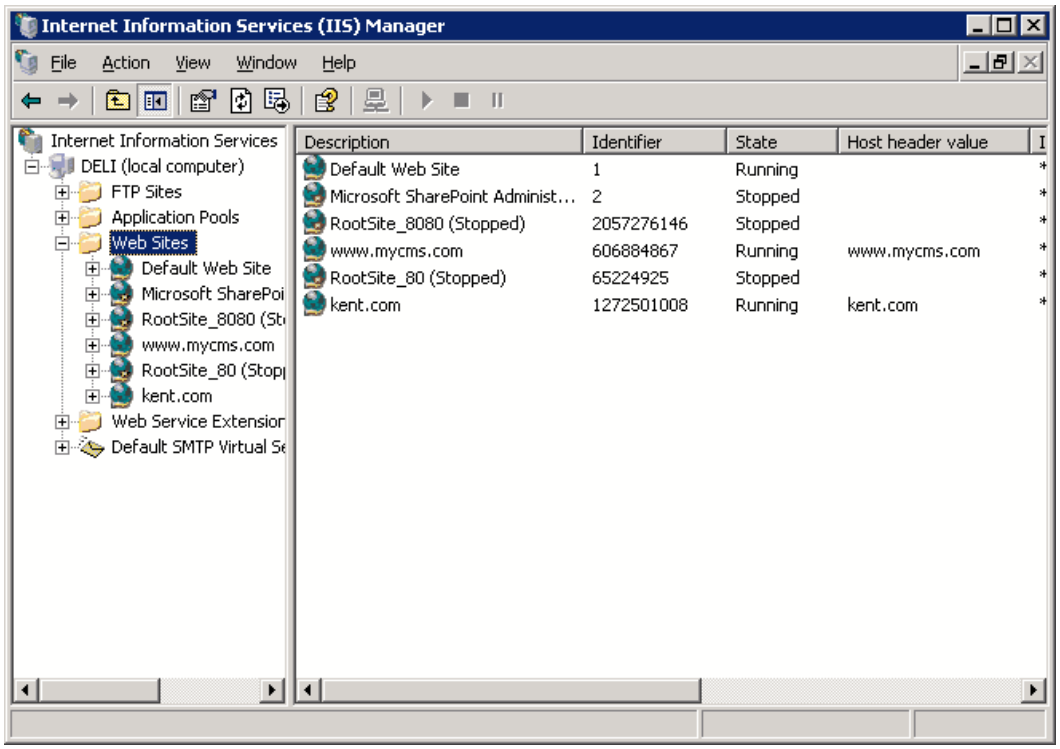
Configuring multiple sites in Kentico CMS (common for all operating systems)

This part is common for all operating systems.

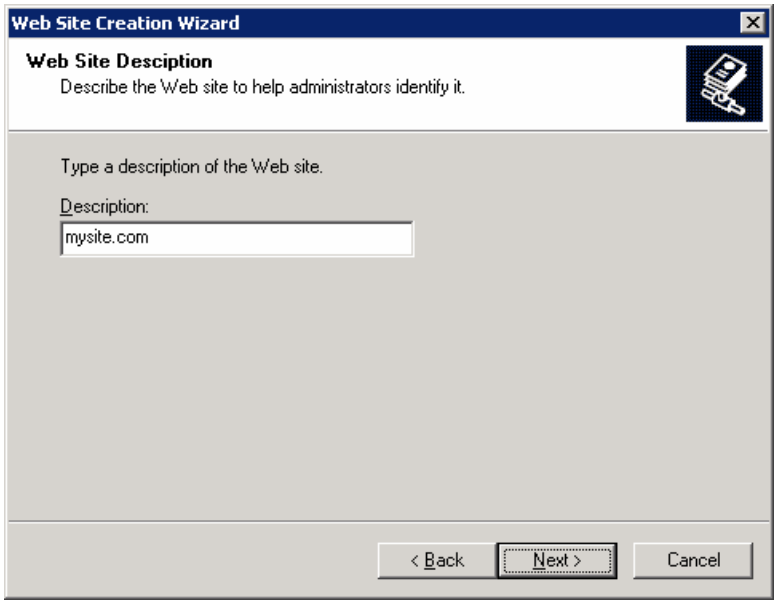
1. Create two websites in **Site Manager -> Sites** - you can either import your existing websites or you can create new websites using the New site wizard.
2. Edit the properties of each website in **Site Manager -> Sites** and set the **Site domain name** value of the each website to the appropriate 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

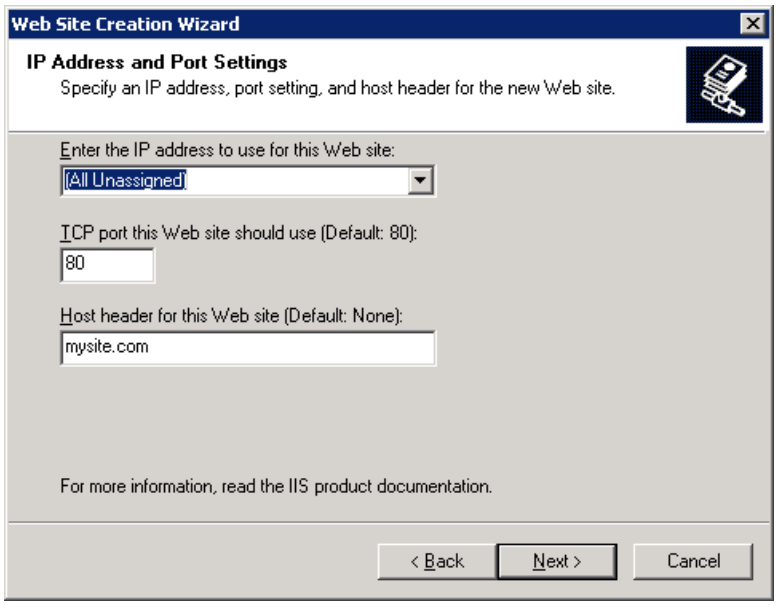
Open **Start -> Administrative tools -> Microsoft Internet Information Services (IIS) Manager**, go to websites section and if the website does not exist yet, create it.



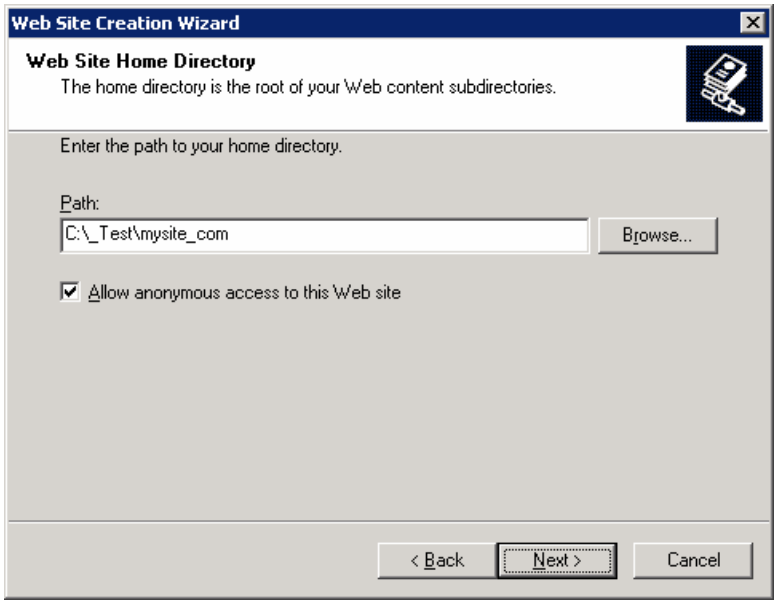
Right-click **websites** and choose **New -> website...** The website Creation Wizard starts. Enter the site descriptive name, such as **mysite.com**:



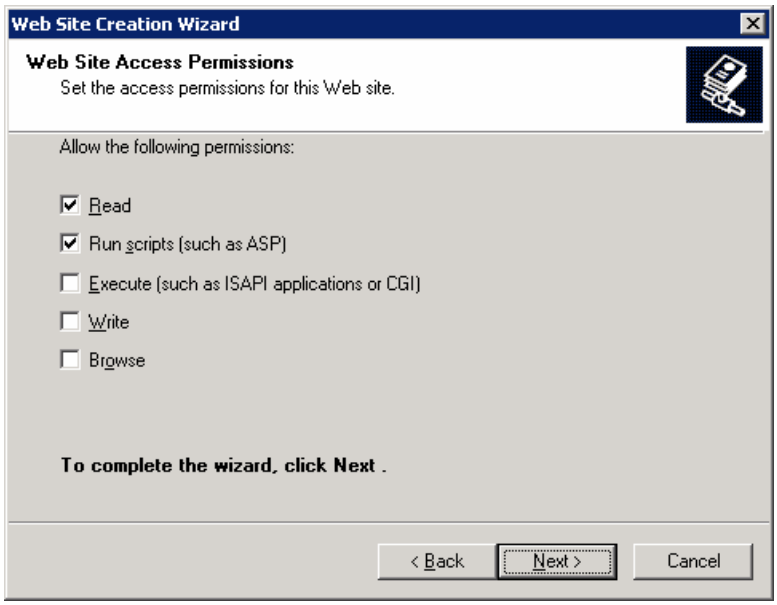
Click **Next**. Enter the **Host header for this website** as **mysite.com**:



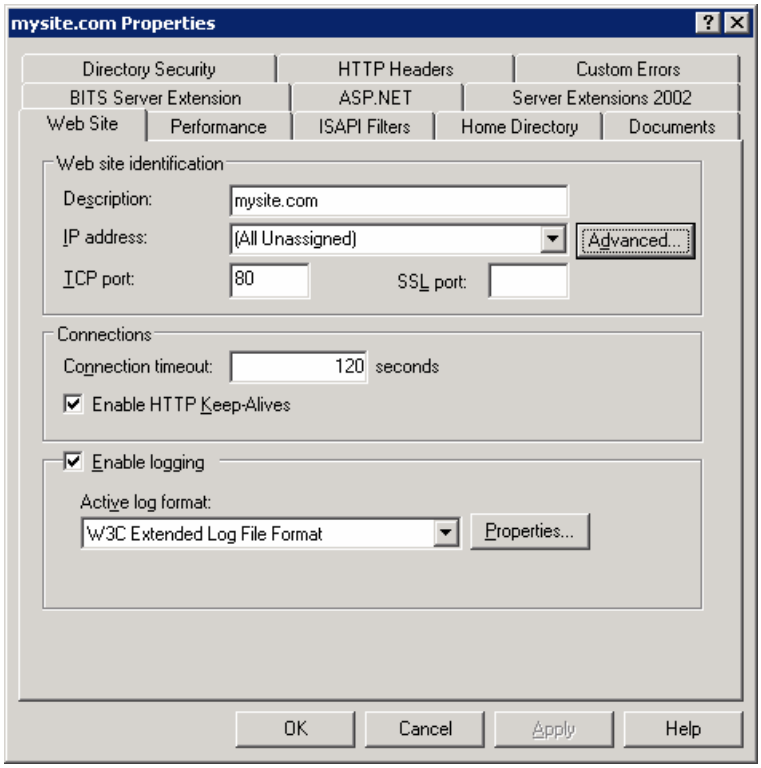
Click **Next**. Select the disk path where the root of your website is placed. This must be the folder where web.config file is placed.



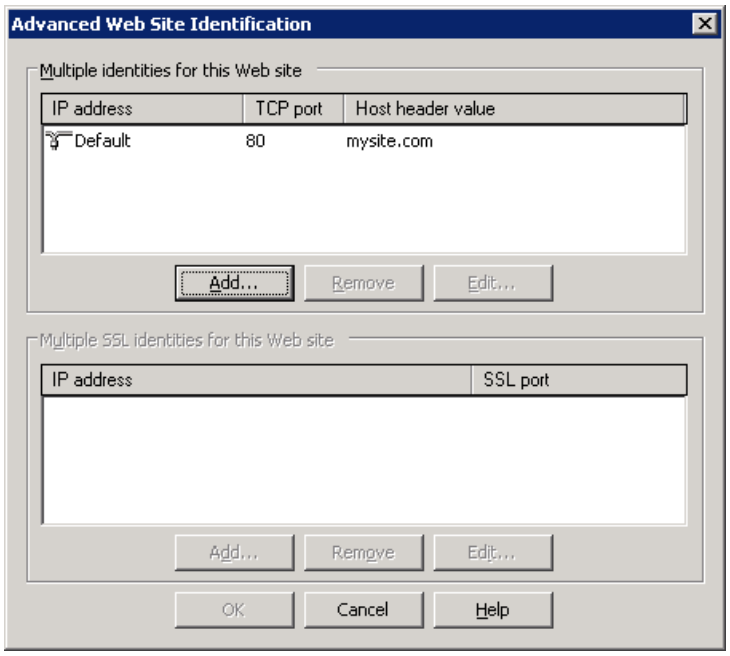
Click **Next**. In the next step, enable the **Read** and **Run scripts** boxes.



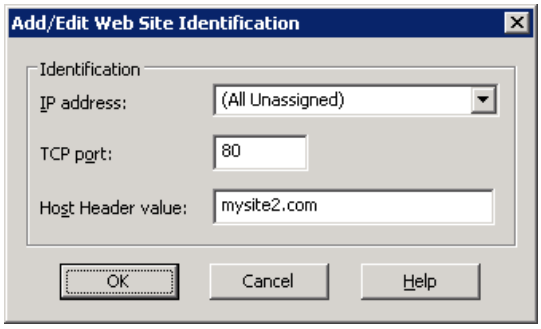
Click **Next**. The wizard is finished. Click **Finish**. Now we need to add the other domain name to the list of URLs hosted by this website. Right-click the newly created website and display website properties. On the **website** tab, click **Advanced...**



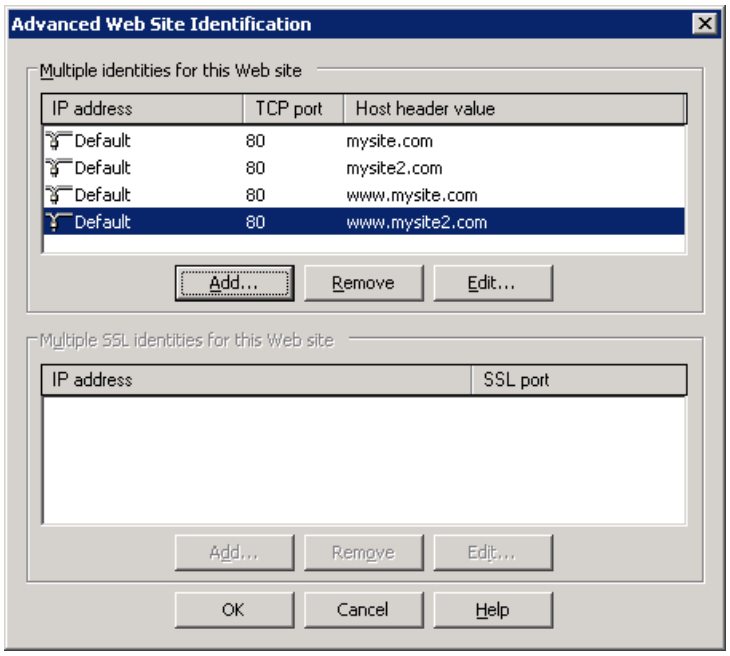
Now you need to add another host header value for your second domain:



Click the top **Add...** button and enter the appropriate values. The standard HTTP port is 80:

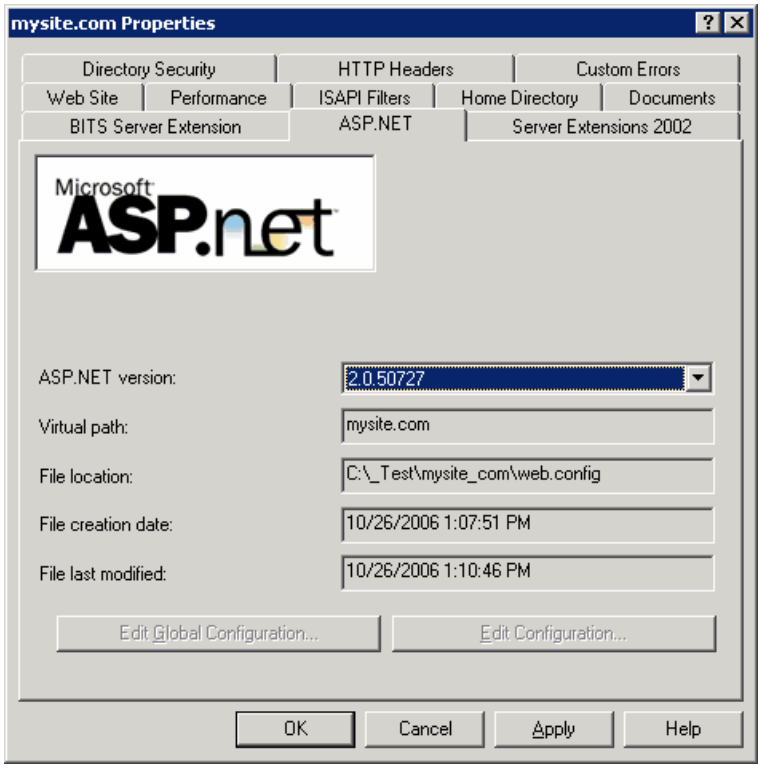


You need to repeat this also for www.* alternatives of your website:



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:

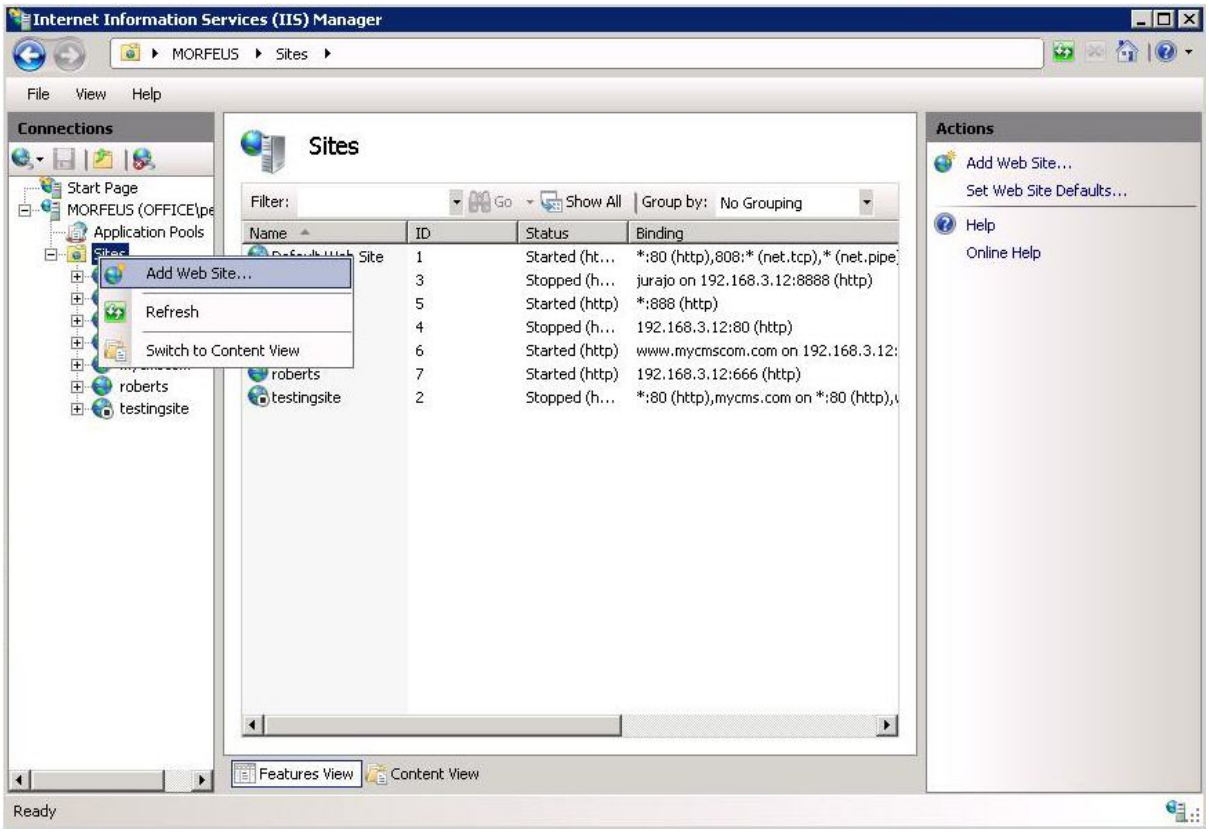
```
127.0.0.1       mysite.com
127.0.0.1       www.mysite.com
127.0.0.1       mysite2.com
127.0.0.1       www.mysite2.com
```

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

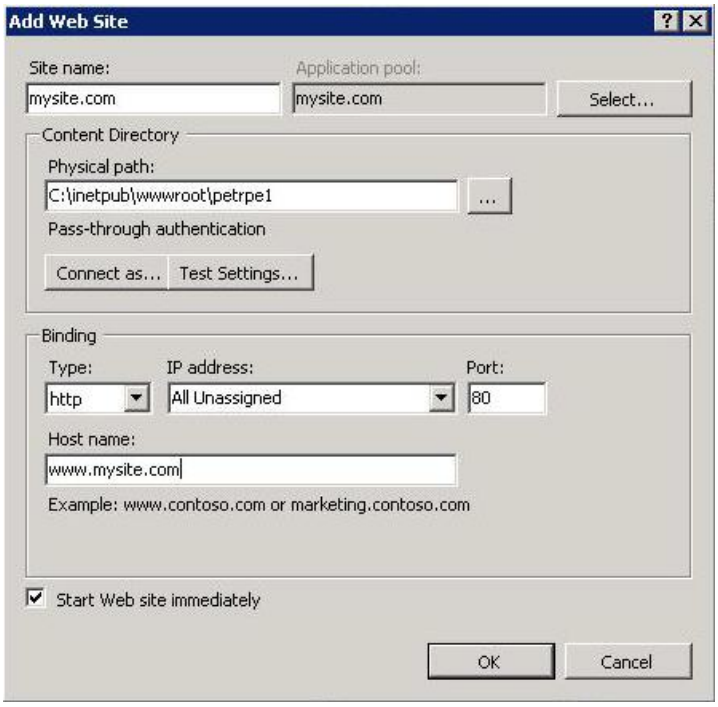
Go to **Start -> Control Panel -> Administrative tools -> Microsoft Internet Information Services (IIS) Manager.** In the tree view, right click **Sites** and choose **Add website...**.



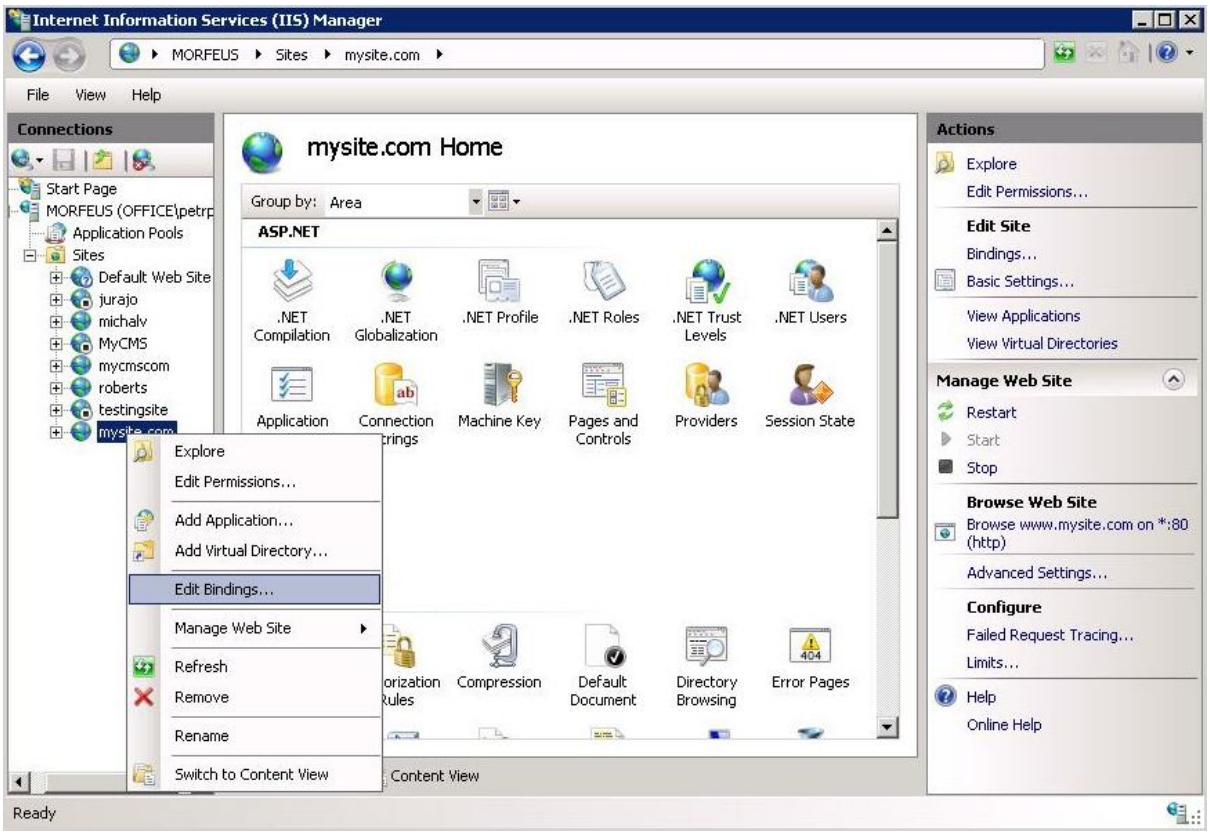
Enter the following details:

- **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

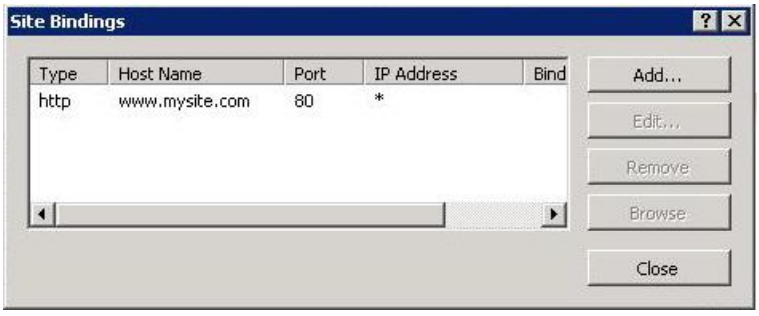
Click **OK**.



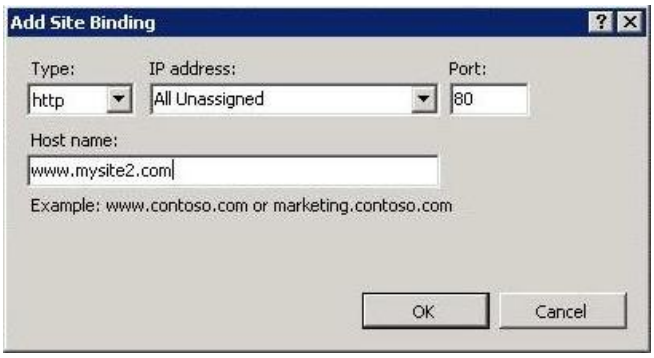
The site should now appear in the tree, under the **Sites** node. Right click the site and choose **Edit bindings** (or **Bindings** on Vista).



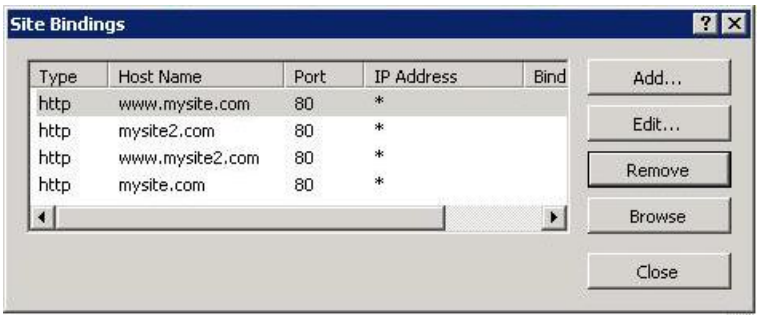
The site bindings dialog appears. Click **Add...**.



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:

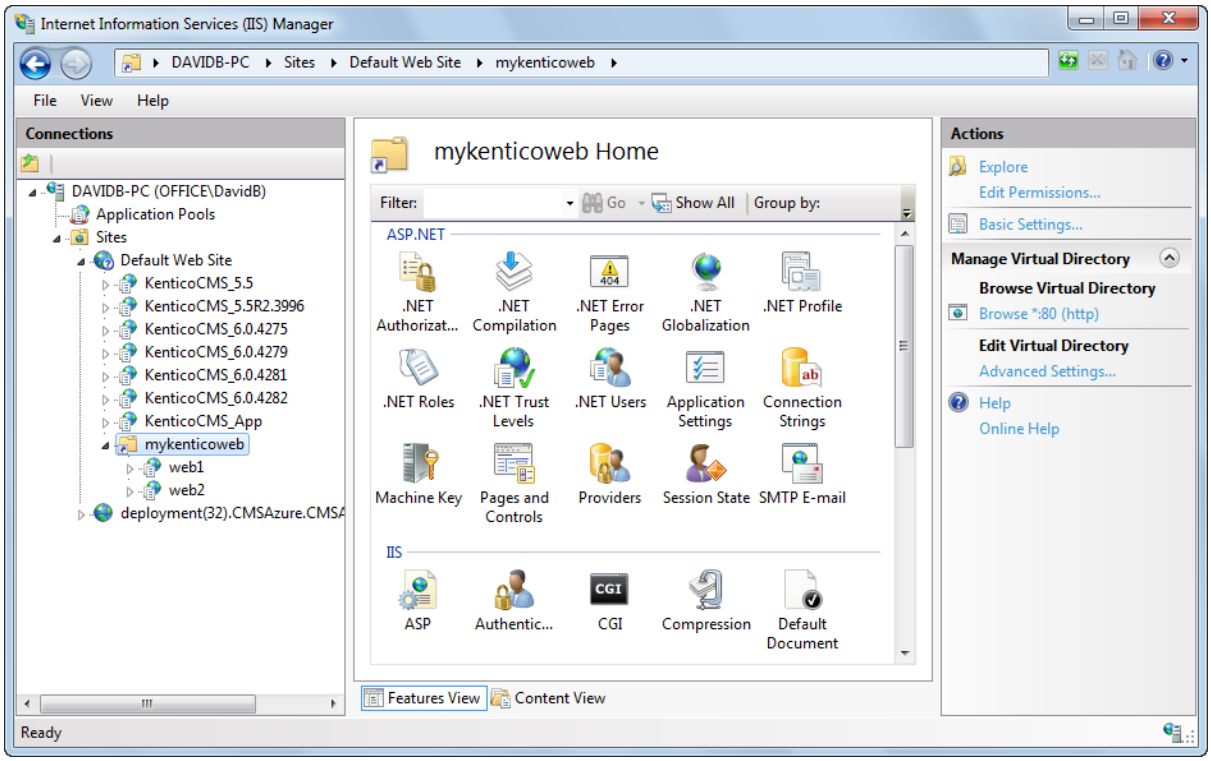
Example

1. Install Kentico CMS to the following folder: **C:\inetpub\wwwroot\mykenticofolder**. Make sure the name of the folder is different from the name in the URL 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.

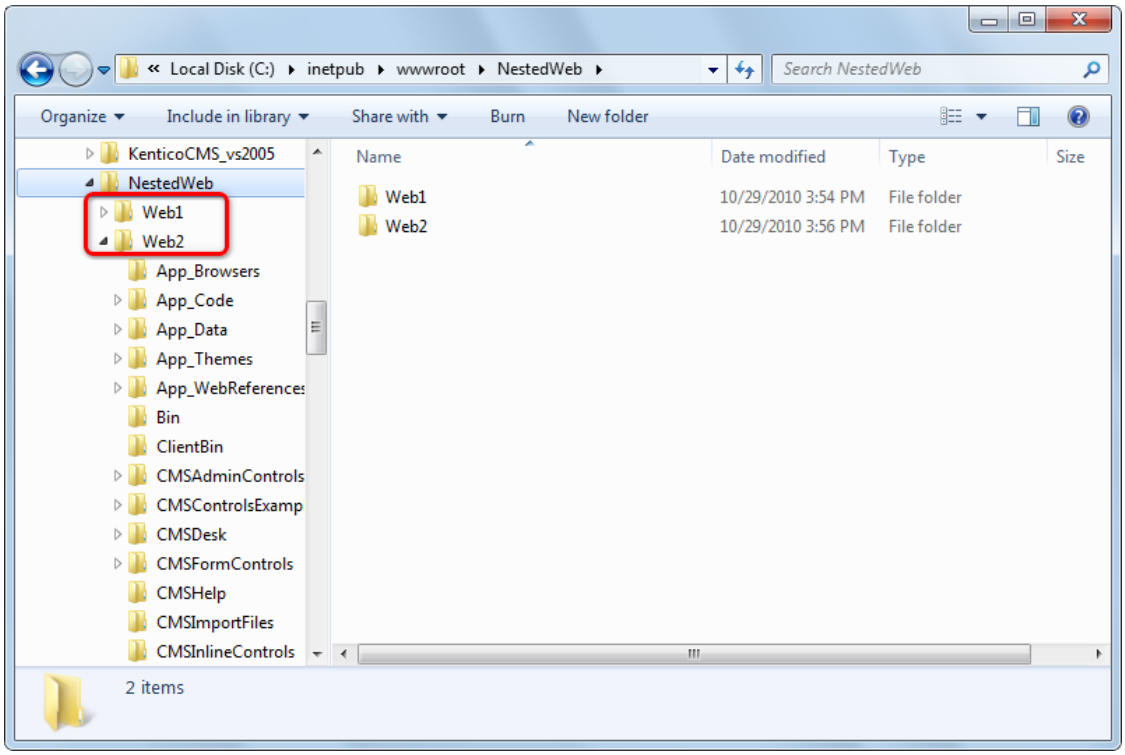
5.12 Configuring nested web sites

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)

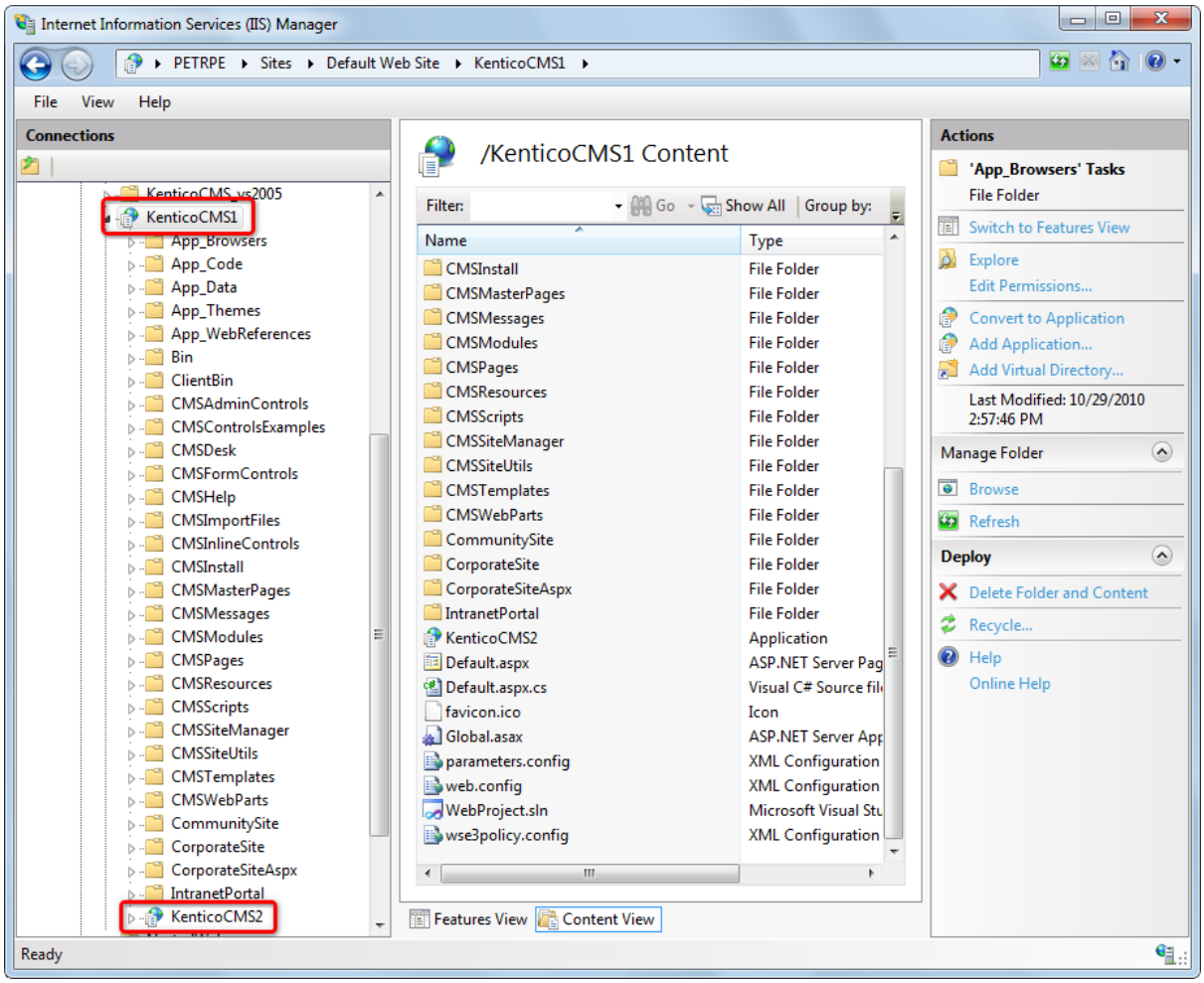
This means that you have two completely independent projects, as you can see in the screenshot below:



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.

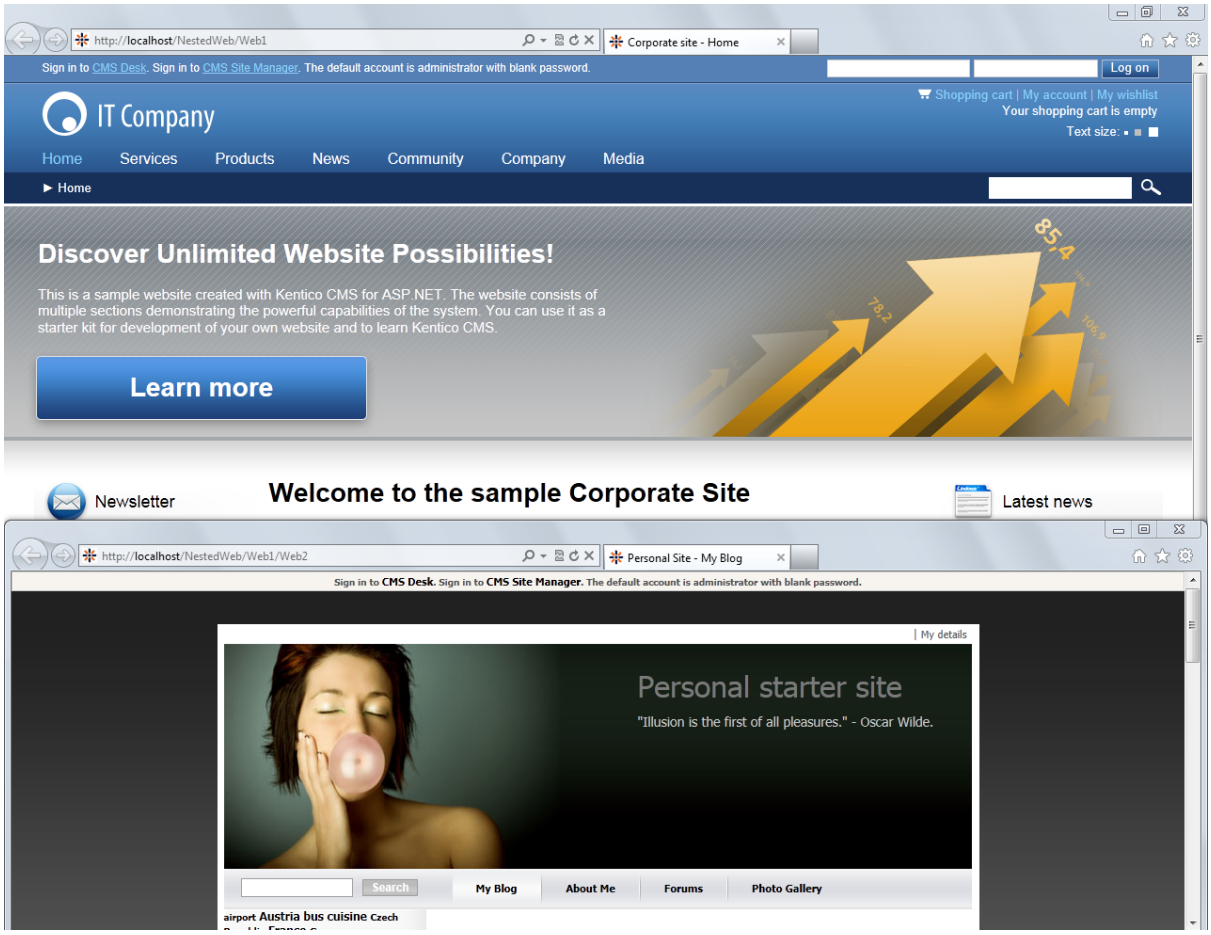


3. The last step is to avoid duplicate module and handler definition keys in the two projects' *web.config* files. To achieve this, open the nested application's (*Web2*) *web.config* file and remove all module and handler definitions from the two sections highlighted in the following code extract.

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>
```

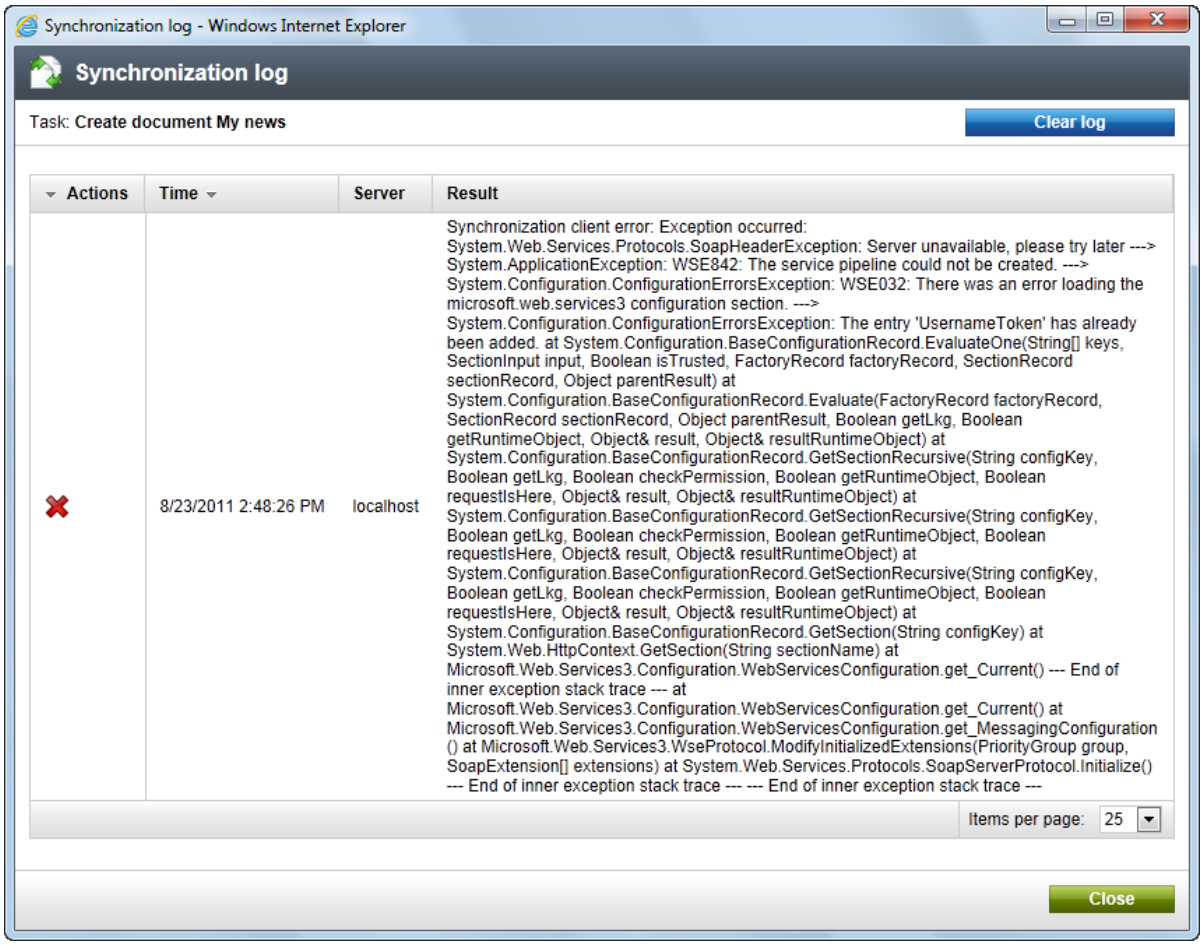
4. The websites are now accessible via nested URLs as you can see in the screenshot below. The websites can be configured independently without experiencing any issues.



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 Sites internals and API

5.13.1 Overview

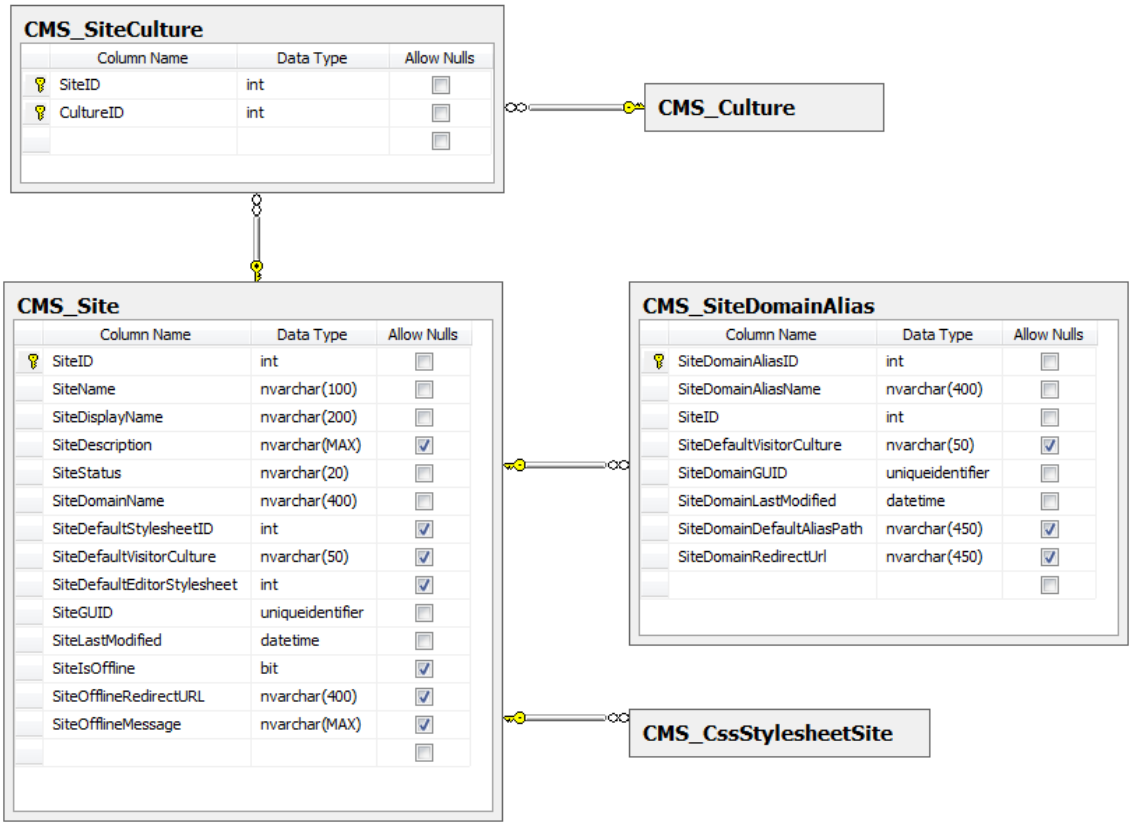
In this chapter, you will learn which database tables and API classes are used for websites. You will also see the most common API examples.

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.

5.13.2 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.



5.13.3 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.

5.13.4 API examples

5.13.4.1 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.TreeEngine;
```

5.13.4.2 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;
}
```

5.13.4.3 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;
}
```

5.13.4.4 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 domain alias
    SiteDomainAliasInfo deleteAlias = SiteDomainAliasInfoProvider.
GetSiteDomainAliasInfo("127.0.0.1", CMSContext.CurrentSiteID);

    // Delete the site domain alias
    SiteDomainAliasInfoProvider.DeleteSiteDomainAliasInfo(deleteAlias);

    return (deleteAlias != null);
}
```

5.13.4.5 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;
    }
```

5.14 Web templates internals and API

5.14.1 Overview

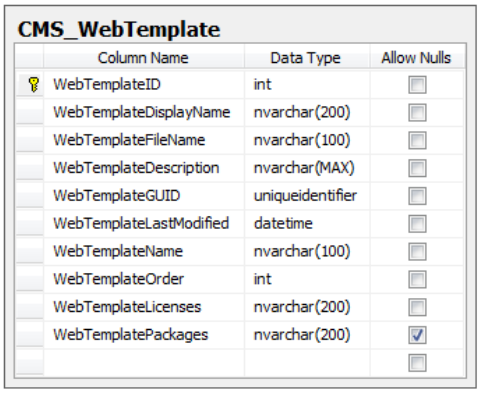
In this chapter, you will learn which database tables and API classes are used for web templates. You will also see the most common API examples.

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.

5.14.2 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)	☑

5.14.3 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 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.

5.14.4 API examples

5.14.4.1 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;
```

5.14.4.2 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

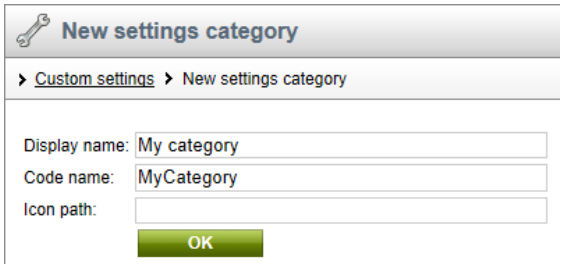
These settings are used for low-level configuration of the CMS system. More details can be found in Appendix B - Web.config parameters.

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.

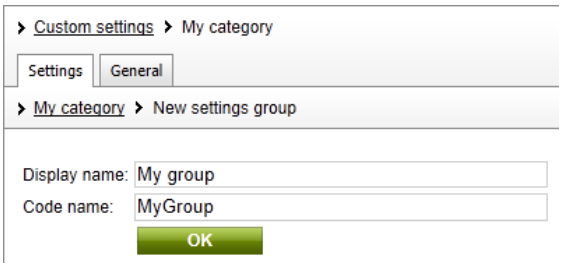
Adding custom settings category

If you need to create a new category, go to **Site Manager -> Development -> Custom settings**, click the **New category** link, enter category properties and click **OK**.



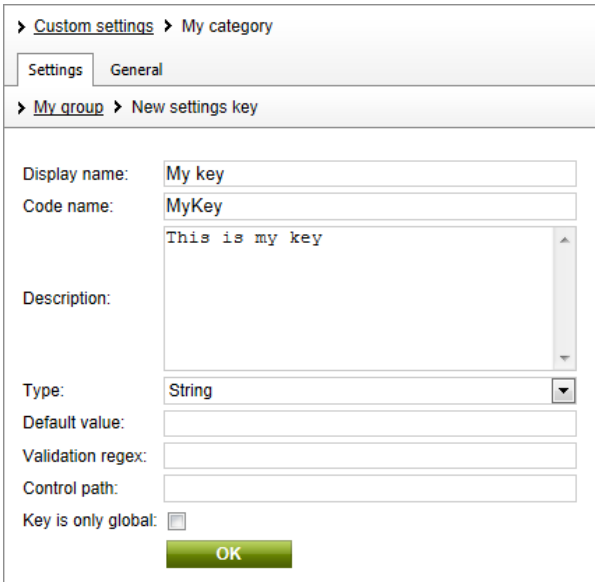
Adding custom settings group

If you need to create a new group, in the **Custom settings** categories tree, select a category and on its **Settings** tab click the **New settings group** link. Enter group properties and click **OK**.



Adding custom settings key

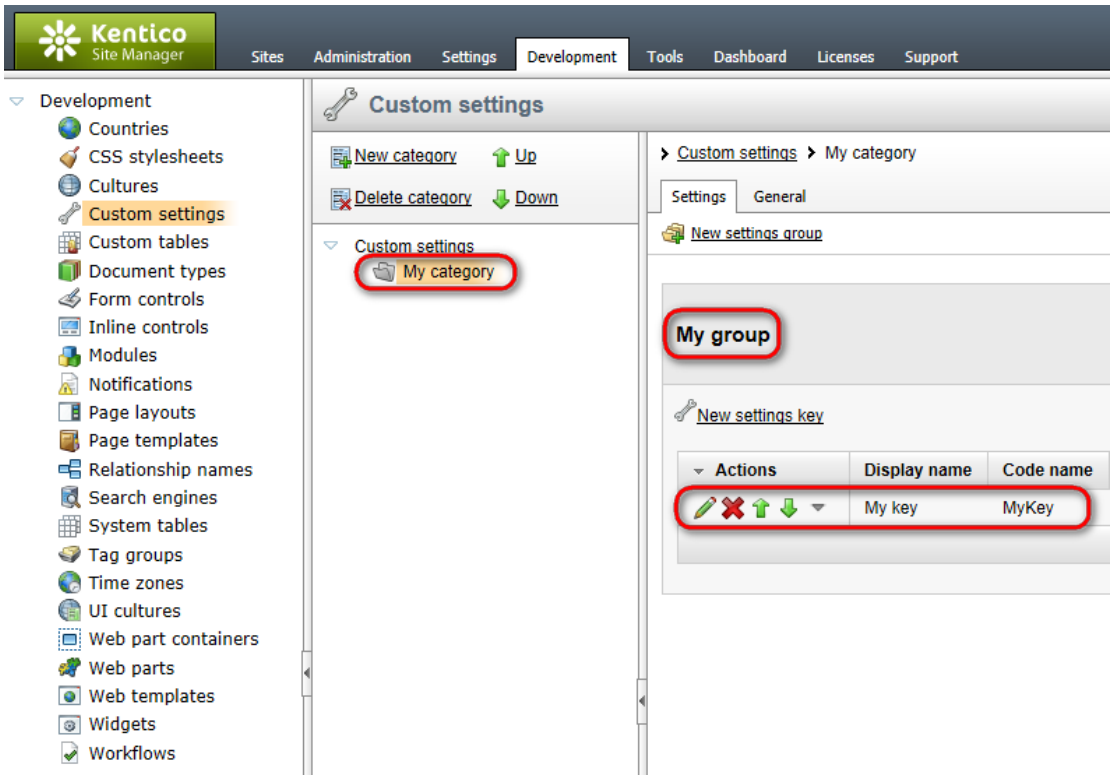
On the **Settings** tab of the selected category click the **New settings key** link beside the group where you would like to add the particular key. Enter key properties and click **OK**. For more information on these properties, please refer to the Managing custom settings topic.



Result

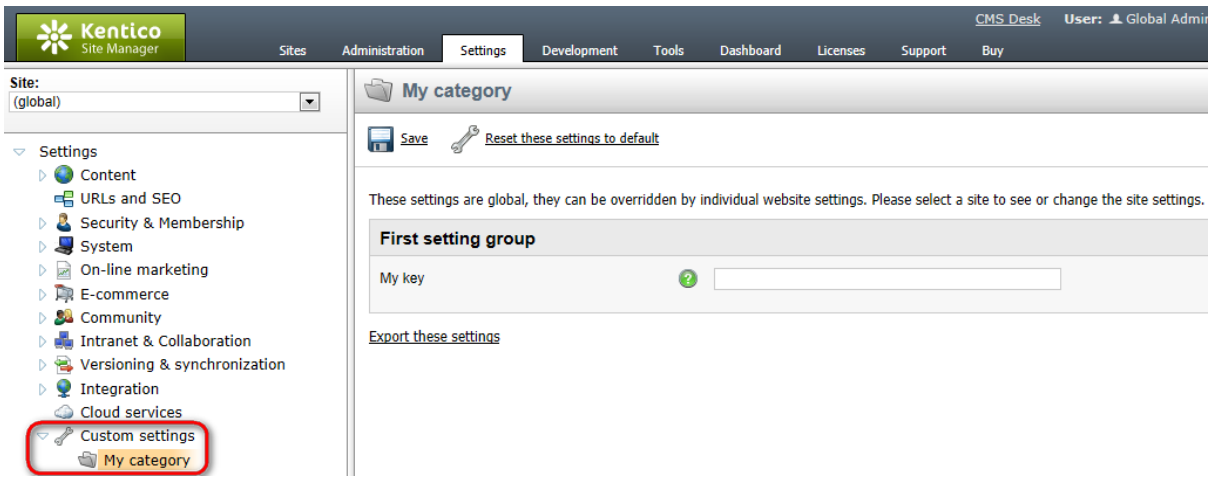
The following screenshots show the result of adding a custom category, group and key:

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

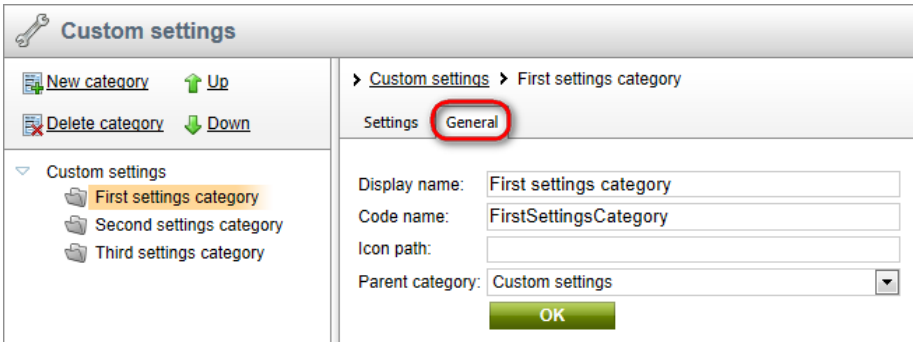
All custom settings added according to the description given in this topic can be configured along with the built-in settings in **Site Manager -> Settings**; if you would like to learn how to do it, please refer to the Configuring settings topic.
By default, there are no custom settings available after 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

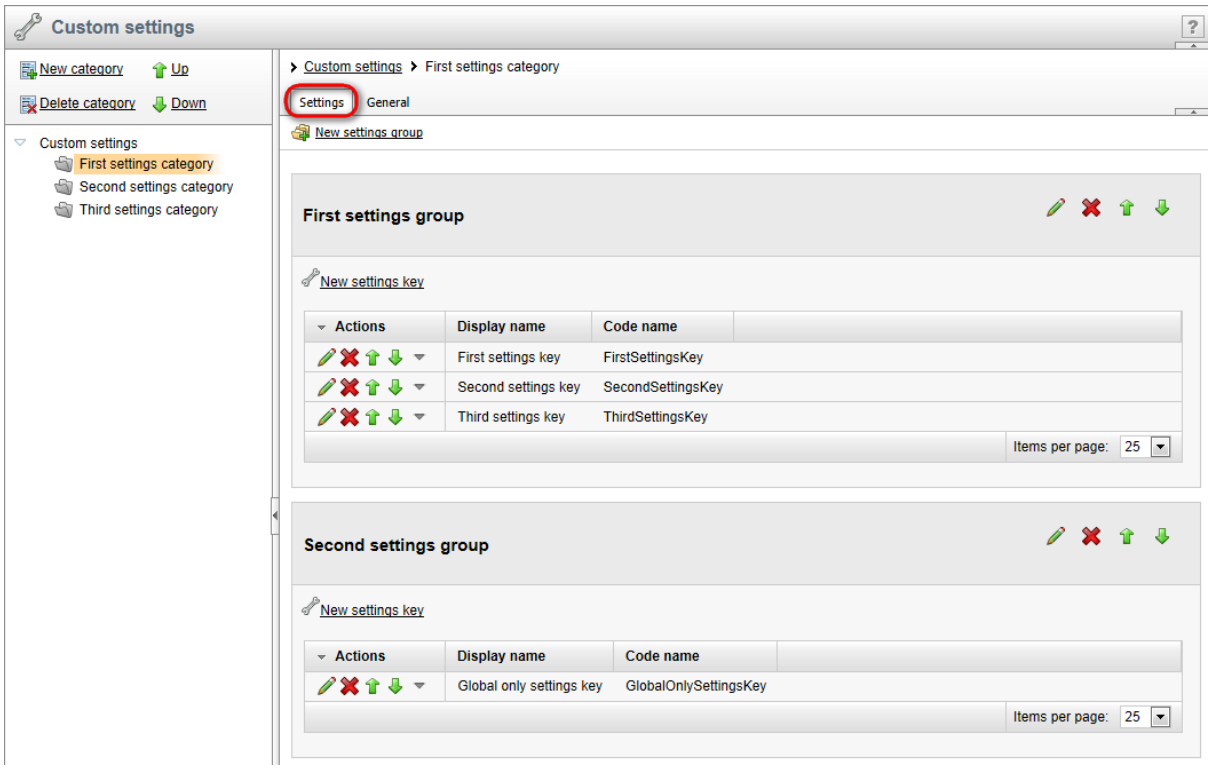
If you need to edit properties of a category, click the category in the categories tree. On the **General** tab, you can modify its general properties:



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.

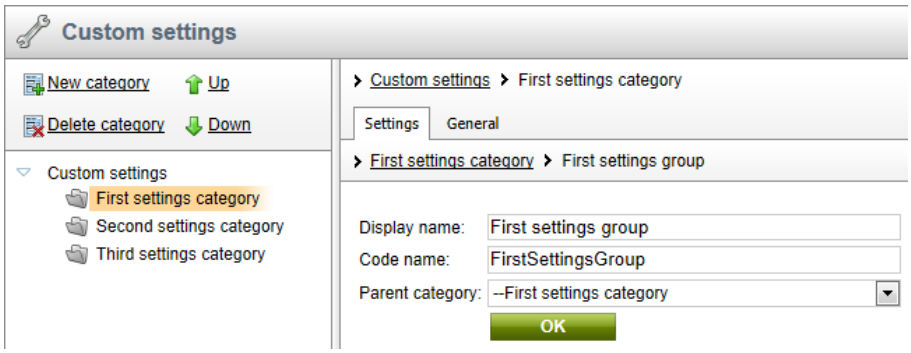
The **Settings** tab allows you to manage category groups and their keys:



Custom categories can also be deleted using the **Delete category** link 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:



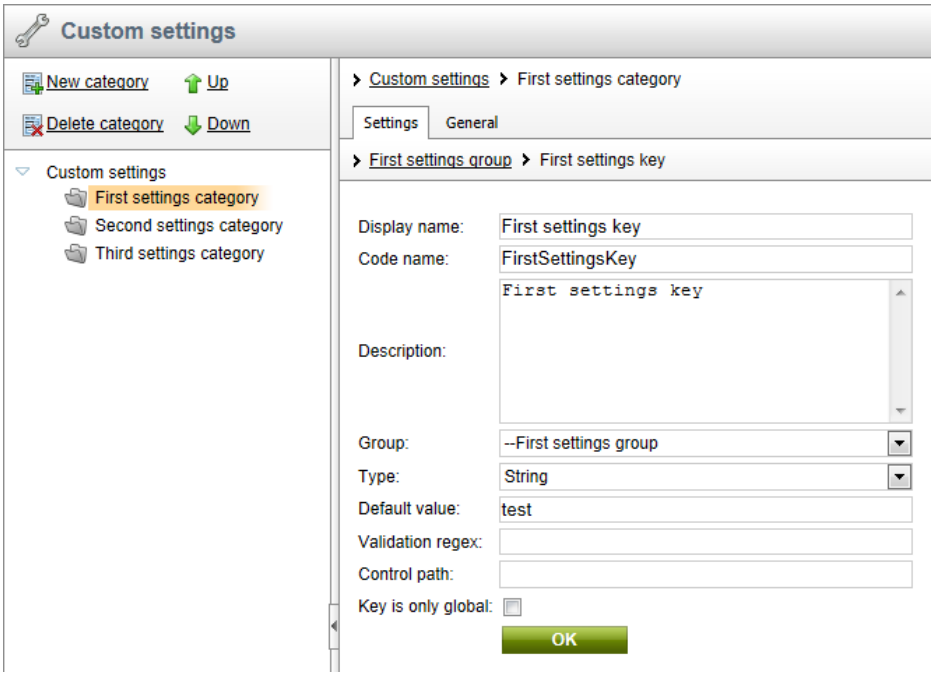
Group properties:

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:

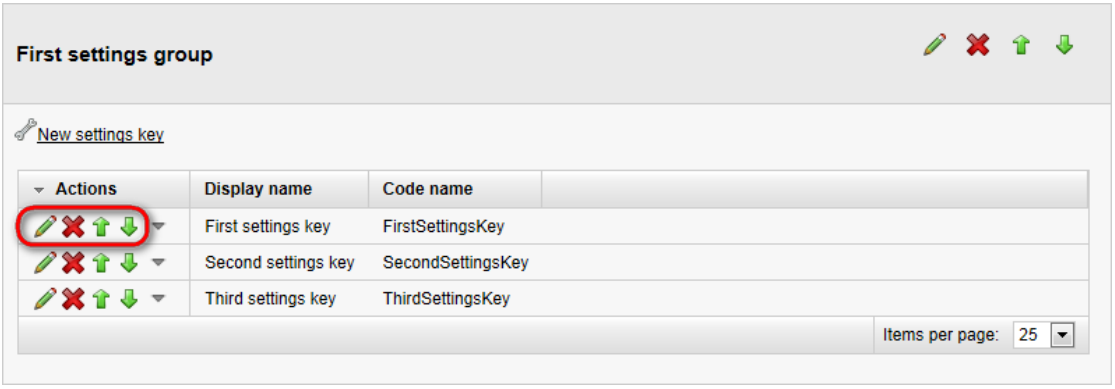


Key properties:

Display name	The name of the key displayed to website administrators.

Code name	The name of the key used in the code.
Description	The text describing the key.
Group	The group where this key belongs; you can select another group to move the key to a different group.
Type	The type of the key 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	The regular expression used to validate the entered value.
Control path	The path to the control that will be used to edit the value, e.g. *~/CMSFormControls/SimpleCountrySelector.ascx*. You can create your custom form control and set its path.
Key is only global	If checked, all available sites will share the same custom key settings.

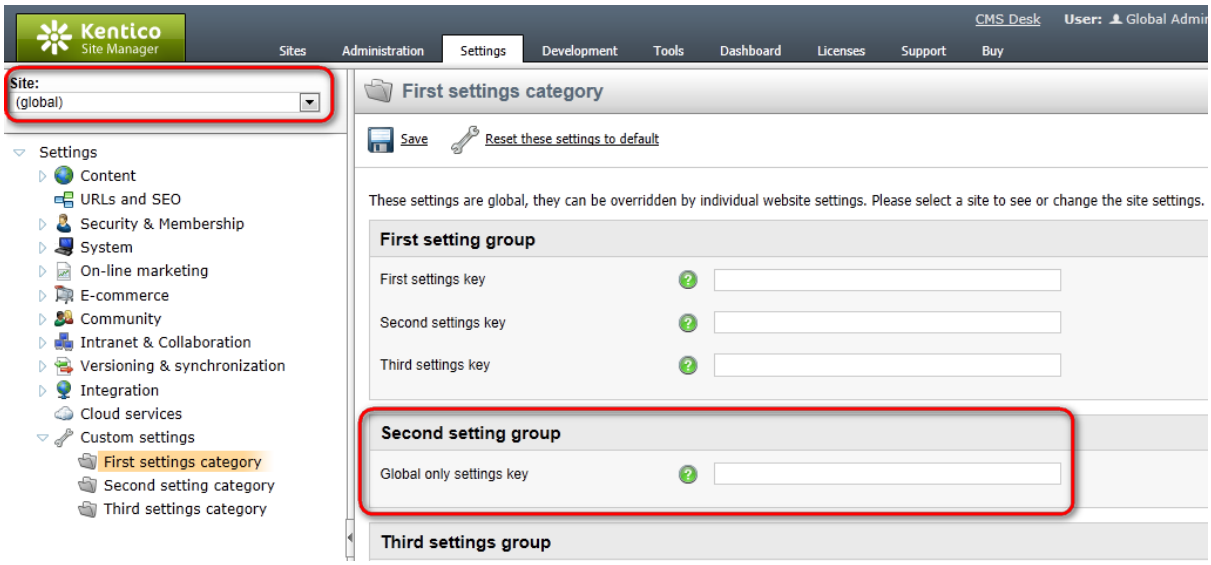
The key can also be **Deleted** (), **Moved Up** () and **Moved Down** ().



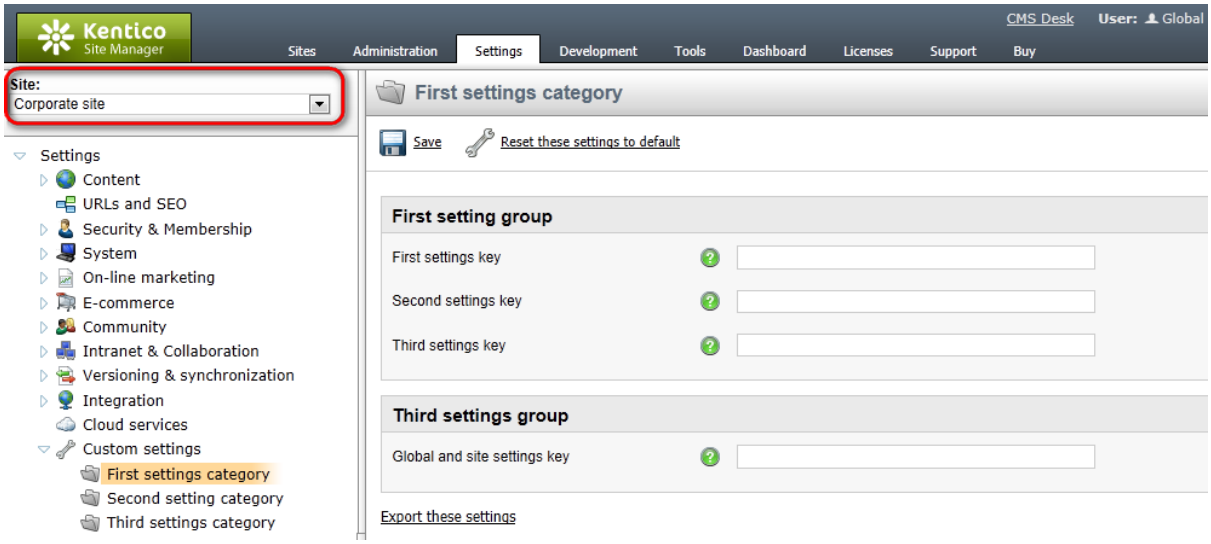
Please note

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.

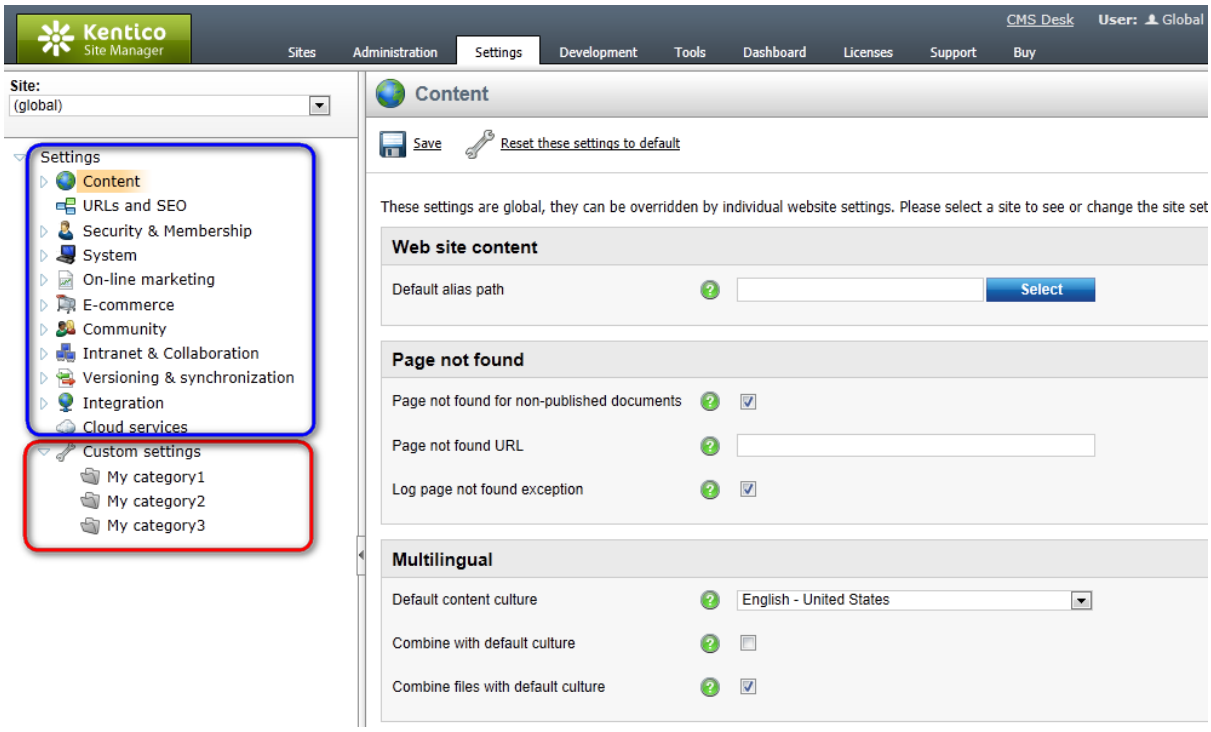


b) A particular site is selected -> the *Global only settings key* is hidden.



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 clicking 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

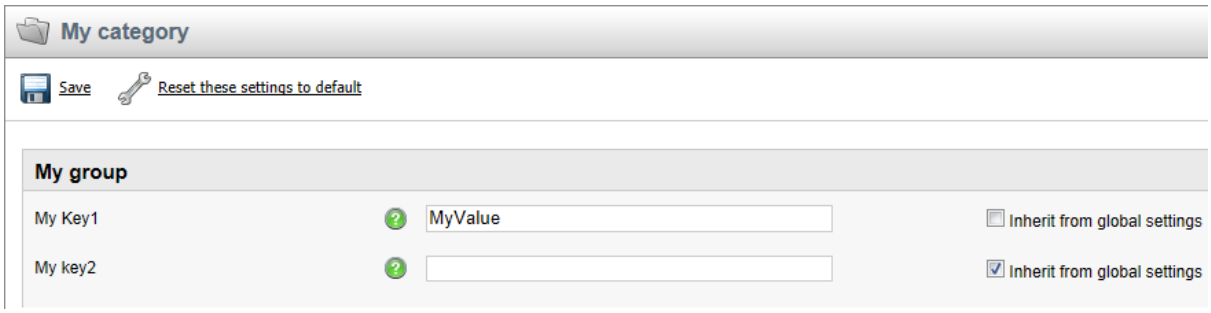
From the **Site** drop-down list, you can choose either a particular site and thus define settings for this particular site - see Site-specific settings keys or you can choose ***(global)*** and define global settings - see Global settings keys.

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

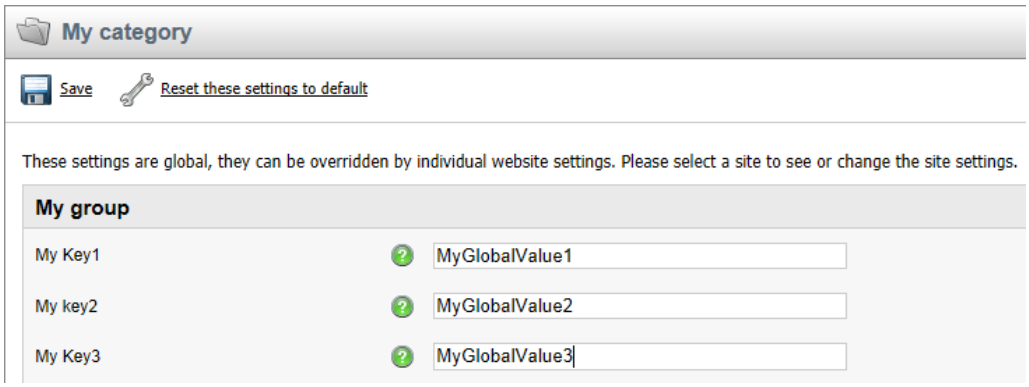
If one of the available sites is selected, all keys whose **Key is only global** property is set to true are hidden; for more information on how to set a key as global only, please refer here.



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**.



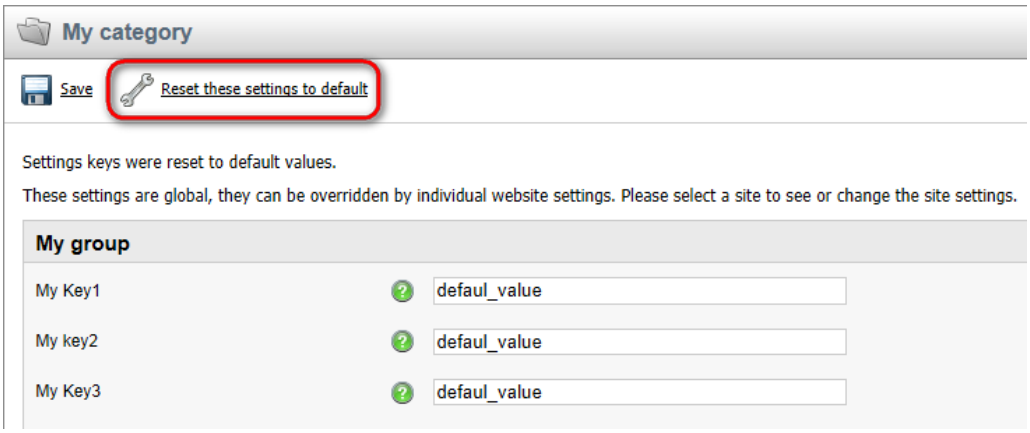
All the three keys are visible and you can edit their values.

How to reset settings keys

Here you will learn how to reset values of settings keys to their default values.

1. Click the **Reset these settings to default** link 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 the **Export these settings** link 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.

6.5 Website settings internals and API

6.5.1 Overview

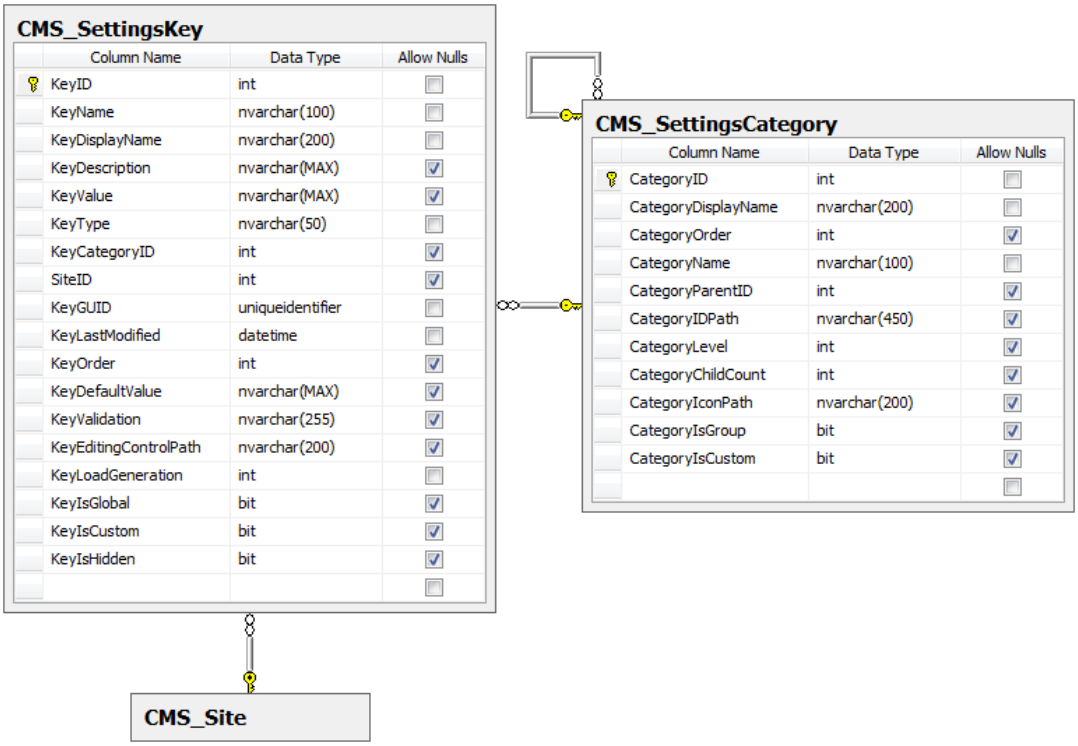
In this chapter, you will learn which database tables and API classes are used in Website settings. You will also see the most common API examples.

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 VII

Development

7 Development

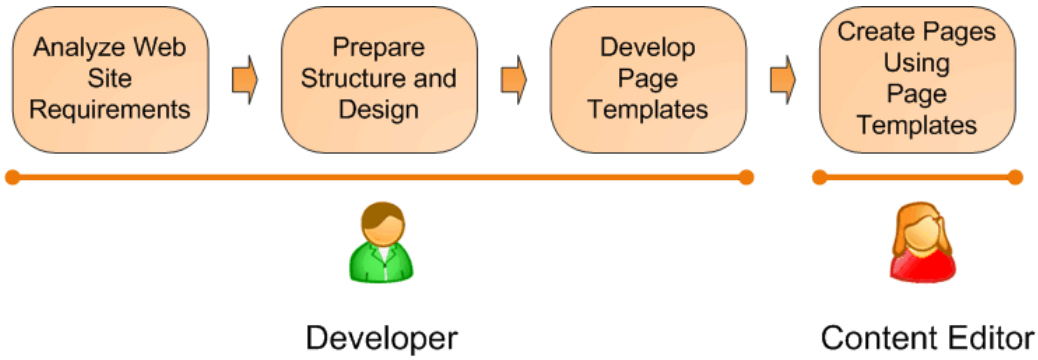
7.1 Overview

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:



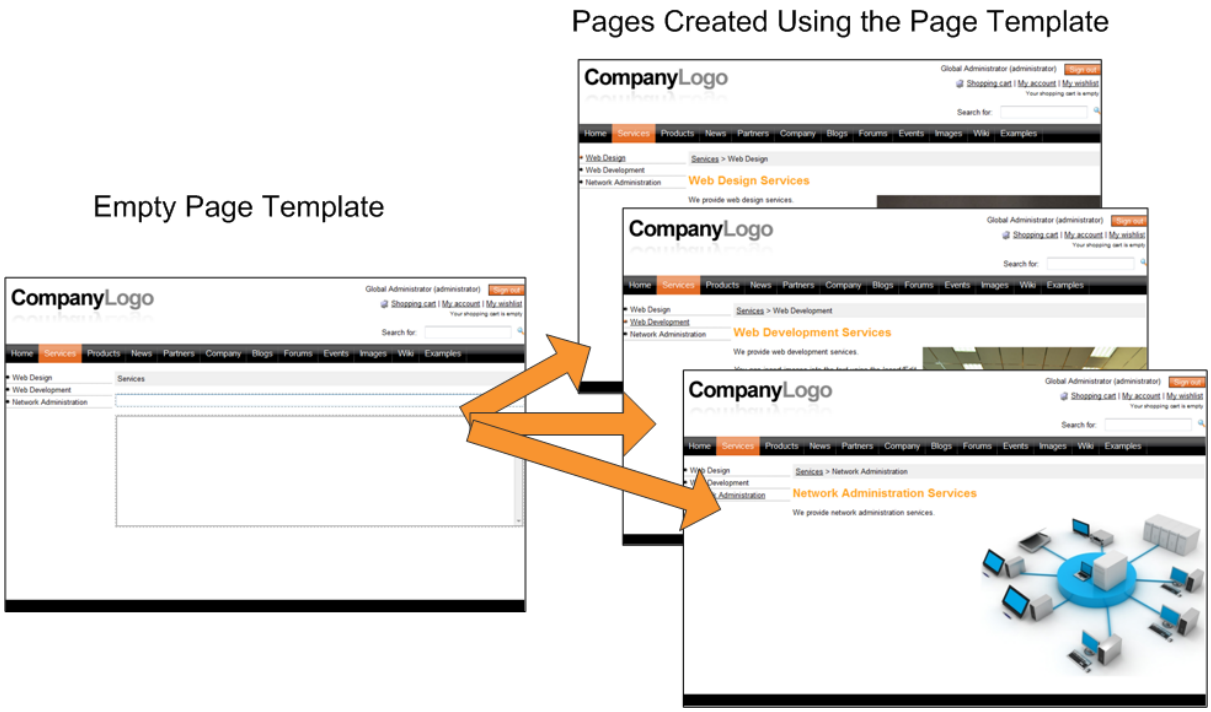
This figure shows how you develop a website and how roles are split between developers and editors. A typical development process consists of the following steps:

1. The developer analyzes client requirements.
2. The developer prepares site structure (site map) and web design.
3. The developer creates **page templates** for every type of page required on the site (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.

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 page template can be re-used for multiple pages with the same structure and design, but with different content.

The 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 following topic describes the two development models for creating such page templates with Kentico CMS.

7.2.3 Portal templates versus ASPX templates

Kentico CMS provides two development models and you can choose which one suits you better:

- **Portal Engine** - this model allows you to build websites using a portal engine. It's the recommended way 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 development and have at least basic programming knowledge of C# or VB.NET.

Both approaches are fully supported and they provide **the same level of flexibility and extensibility**. We recommend that you use the portal engine model, but if you're an advanced .NET developer and do not trust portal engines, you may want to use ASPX templates.

If required, both models can be combined on a single website. For example, you can integrate ASPX templates into a portal engine website, or even custom ASPX pages implementing your own applications. On the other hand, special areas can be defined on ASPX templates where editing through the portal engine is possible.

The following table compares the portal engine and ASPX templates:

	Portal Engine	ASPX Template

How you work	You build the website using the browser-based interface. No programming knowledge is required for common tasks.	You build ASPX pages in Visual Studio. At least basic programming knowledge of ASP.NET and either C# or VB.NET is required.
How you assemble pages	You use built-in or custom web parts that you place into customizable page layouts (HTML code with placeholders for web parts).	You use built-in or custom ASP.NET server controls that are placed on the ASPX pages. You can also use the code behind files - these are standard ASPX pages and they are part of the website project that you can open in Visual Studio. You can also place web parts (which are actually ASCX user controls) on the page templates if the appropriate server control is not available.
Master pages and visual inheritance	Sub-pages inherit the content from the parent pages by default (so called "visual inheritance"). The inheritance can optionally be broken if you want to create a page without parent content.	All page templates may use a master page, which is a standard ASP.NET master page. The pages do not inherit content from their parents, they only inherit content from the master page (if it's used).
Custom code integration and extensibility	You can create your own user controls (ASCX) and web parts (ASCX with specific interface) in Visual Studio if you need to integrate some custom functionality. You can add any custom controls and code to the web parts or user controls that you use on your website.	You build standard ASPX pages with code behind files, which means you can put any custom controls and code onto the page in Visual Studio as you normally would.
Advantages	• Easier and faster to build a website. • ASP.NET programming knowledge is not required for common tasks. • You can build the whole website very quickly, using only the web browser.	• Standard ASP.NET architecture. • You can use your favorite development tools, such as Visual Studio for all changes.
Disadvantages	• Proprietary architecture and development.	• Requires ASP.NET programming knowledge.

Is Kentico CMS just another portal engine?

Now you may wonder what the difference between Kentico CMS and DotNetNuke or Rainbow Portal is.

Well, the main difference is the **flexibility**. Kentico CMS gives you full control over:

- site structure
- site navigation
- page layout
- design
- content structure

Also, it's important to explain that Kentico CMS is a **content management system**, not only a portal engine. This means it provides features of advanced CMS systems, such as:

- content repository using a logical tree hierarchy for all types of content - see Where is the content stored? for details
- content/design separation
- custom document types with custom fields
- workflow and versioning
- content locking (check-out, check-in)
- multilingual content
- content preview and content staging
- document-level permissions with permission inheritance
- full-text search in all content
- document management features for uploaded files

Moreover, Kentico CMS comes with many **professional and flexible built-in modules out-of-the-box** , including:

- Newsletters
- On-line forms
- Forums
- E-commerce
- Staging
- Image gallery
- Blogs
- Polls
- Web analytics
- Reporting

It means you do not need to obtain or purchase third-party modules with inconsistent user and programming interfaces, but you get everything from a single source, with complete documentation and support.

7.2.4 Portal engine development model

7.2.4.1 Portal engine overview

Each page in a Kentico CMS website uses some page template. A **page template** consists of a layout with particular web parts and their configuration. The portal engine page templates are physically stored in the database, so you will not find them on the disk.

The design of a page template is determined by its **page layout**, which can either be defined through full ASCX markup or standard HTML code. The layout's code can be used to set up a two-column, three-column, or virtually any custom layout you can think of.

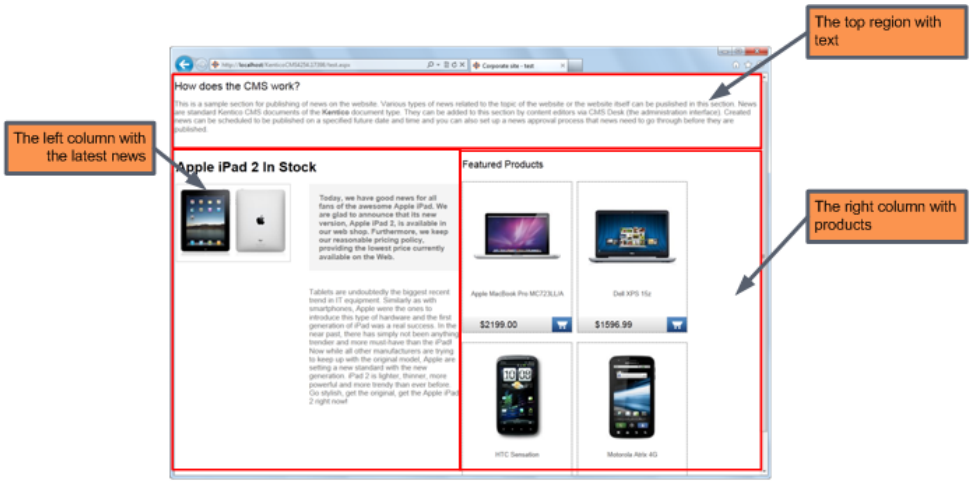
The page layout contains special markup tags that specify places where developers can place web parts - so called **web part zones**. A web part zone can contain any number of web parts.

A **web part** (also called "servlet", "portlet" or "module" in other solutions) is a piece of code that displays some content or provides some functionality. Technically, web parts are standard ASCX user controls with some predefined programming interface (inherited from *CMSAbstractWebPart*). More 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:

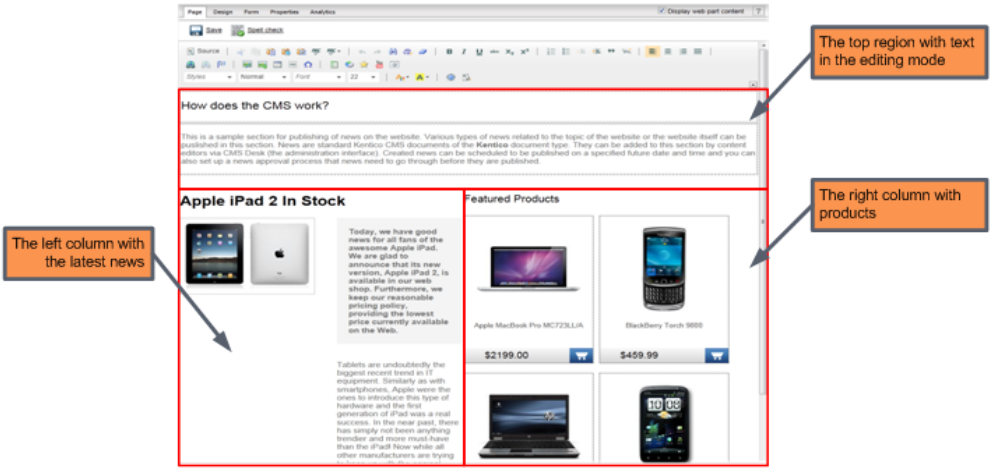
1) The sample page on the live site

This image shows our sample page on the live site. It displays text on the top, news in the left column and products on the right.



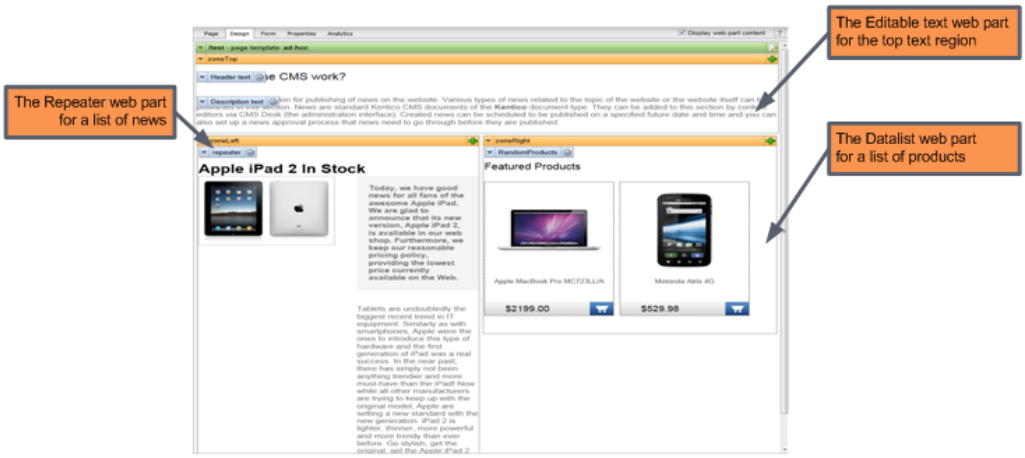
2) The sample page in the editing mode

Here you can see the top region with editable text and the lists of news and products.



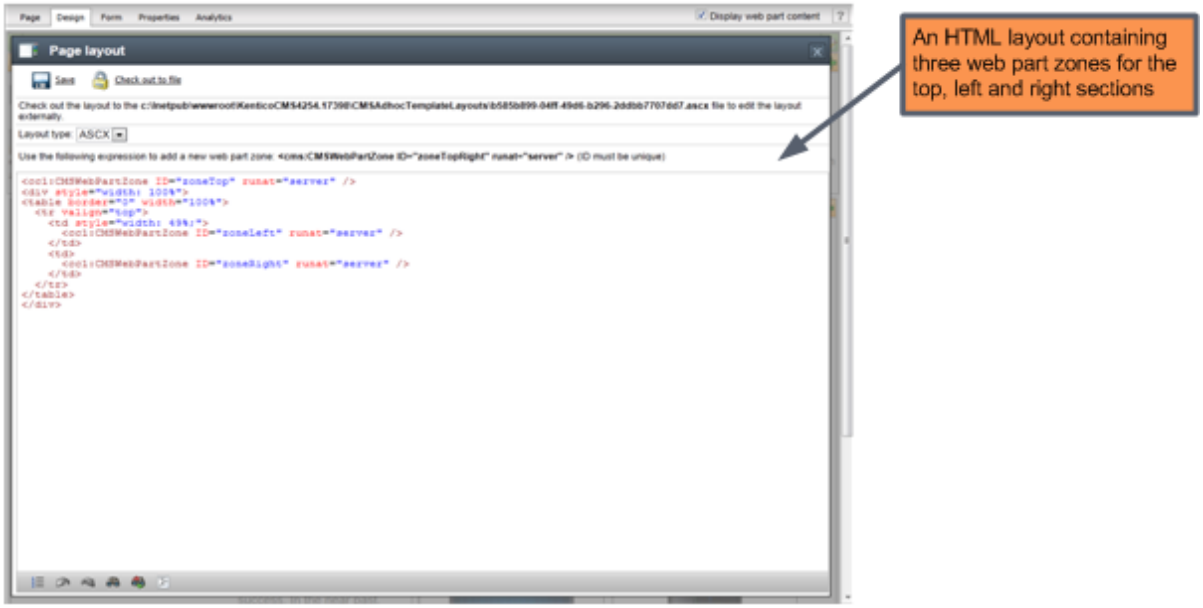
3) The page template of the sample page on the Design tab

This is the **Design** mode view of the page template used for our sample page. As you can see, 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 of the page template with tags representing web part zones on the top and in the table columns. We are using tables for the layout here, but you can easily use a CSS-based layout - 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:

1. Creating a blank page using an ad-hoc template.
2. Cloning an existing page template.

We will start with the first option.

What is an ad-hoc template?

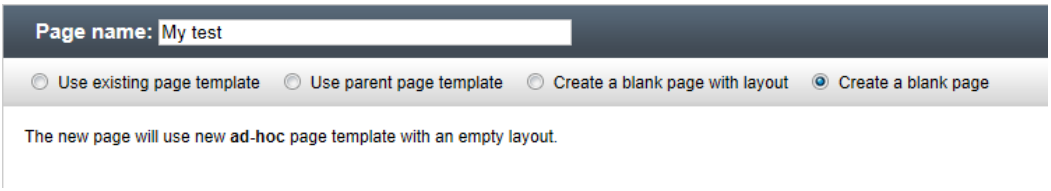
An ad-hoc template is a page template that is used by only one page. An ad-hoc template doesn't have any name. If you want to re-use the template for multiple pages, you need to save it as a named template first.

Creating a blank page using an ad-hoc template

Please note: It's recommended that you use the sample **Corporate Site** for this example.

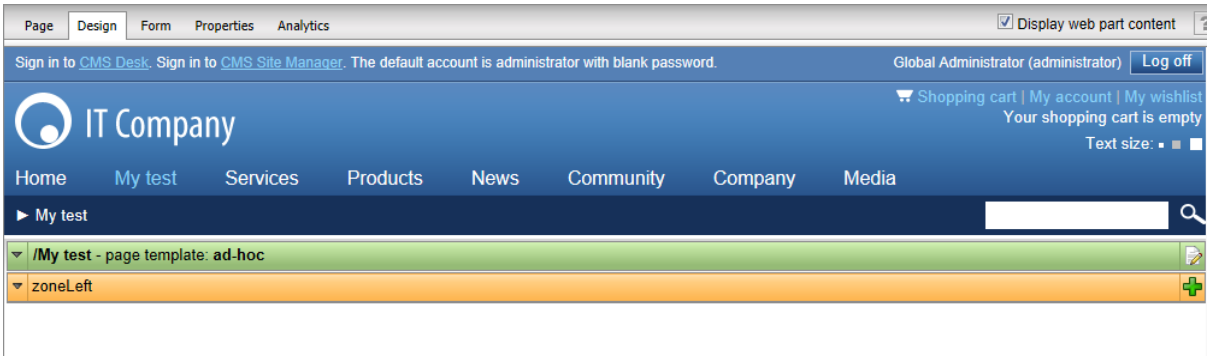
1. Go to **CMS Desk -> Content**, select the root of the content tree, click **New** and choose the **Page (menu item)** document type.

2. Enter *My test* into the **Page name** field, choose the **Create a blank page** option and click **Save**.

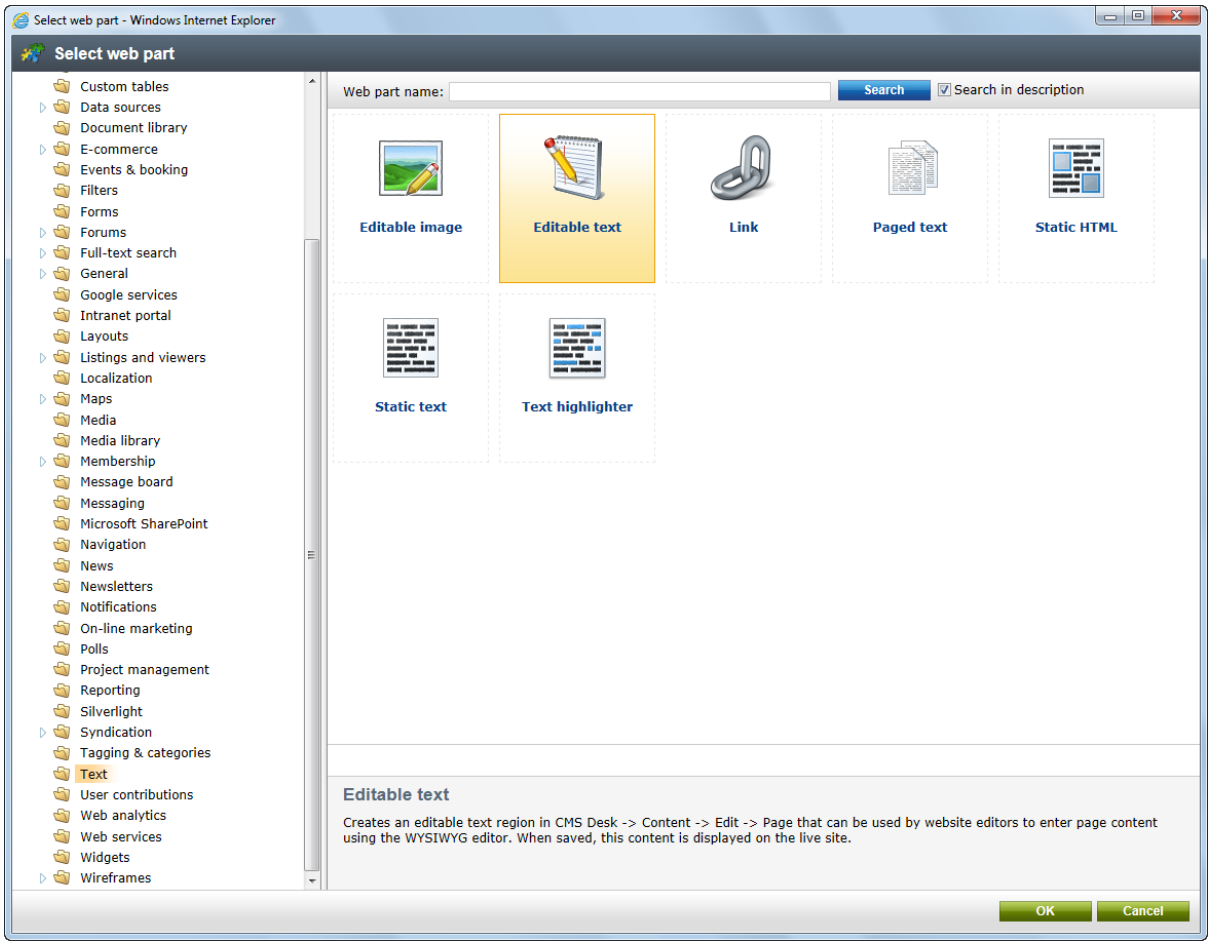


3. The page is now displayed in the tree view and selected. Click the **Design** tab. You will see a page as in the screenshot below.

The green box with **/My test** title displays the current page. As you can see, we are now using an ad-hoc template that is specific for this page. The yellow box with the **zoneLeft** title displays the web part zone, where web parts can be placed.

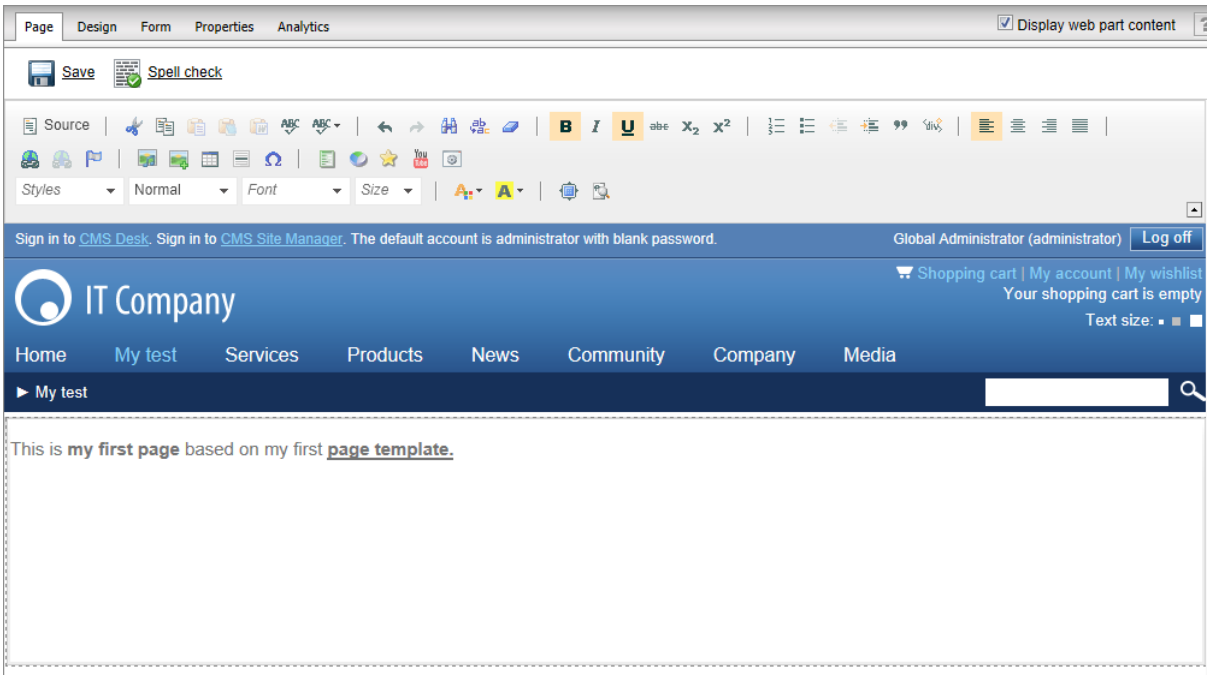


4. Click **Add web part** (). The **Select web part** dialog opens. Select the **Text/Editable text** web part and click **OK**. This web part allows you to create editable regions where content editors can insert content.

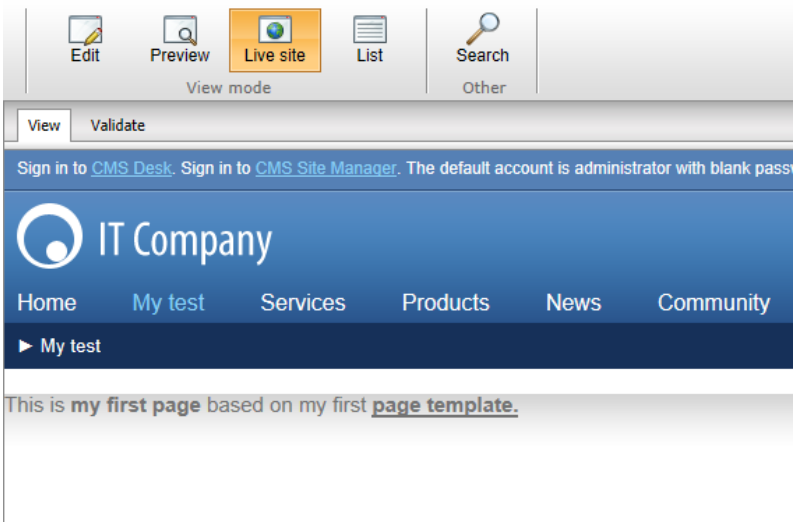


5. The **Web part properties** dialog opens. Enter *Main text* into the **Editable region title** property and click **OK**. The web part is now added to the web part zone:

6. Now click the **Page** tab. You will see the page in editing mode. The page now contains your web part which displays the editable region. Enter some text into the region and click **Save**.



7. Click **Live site** in the main toolbar. You will see the live version of the page that is displayed to the visitors:



What you did

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 into the page.

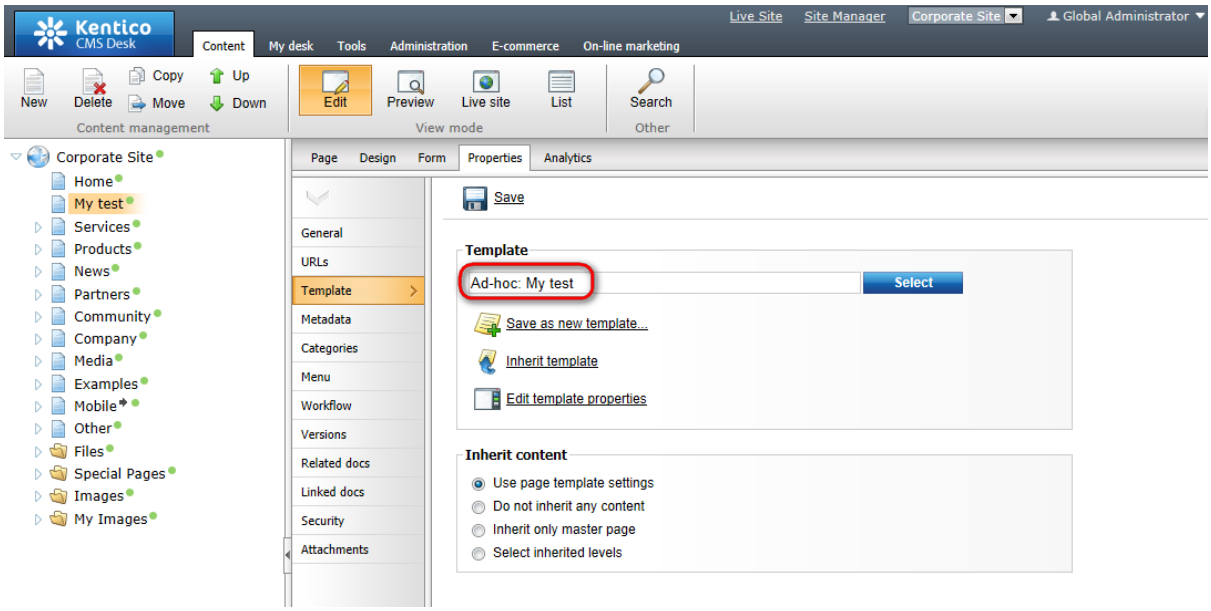
Important!

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 is deleted as well.

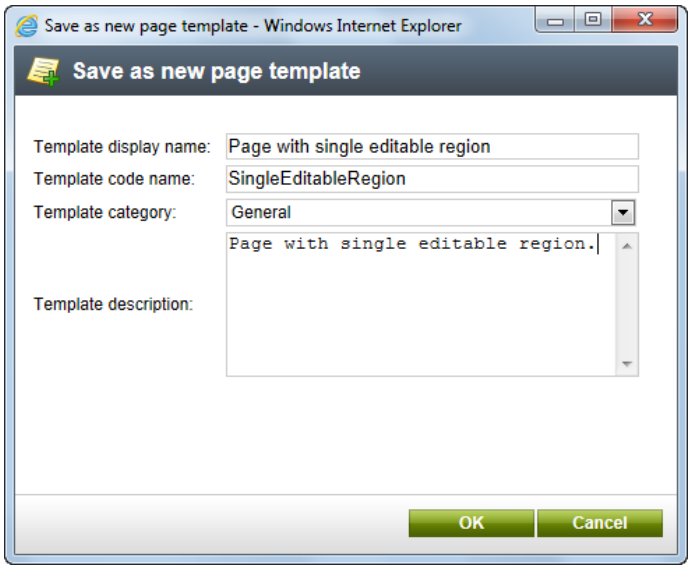
7.2.4.3 Re-using an ad-hoc page template

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**.

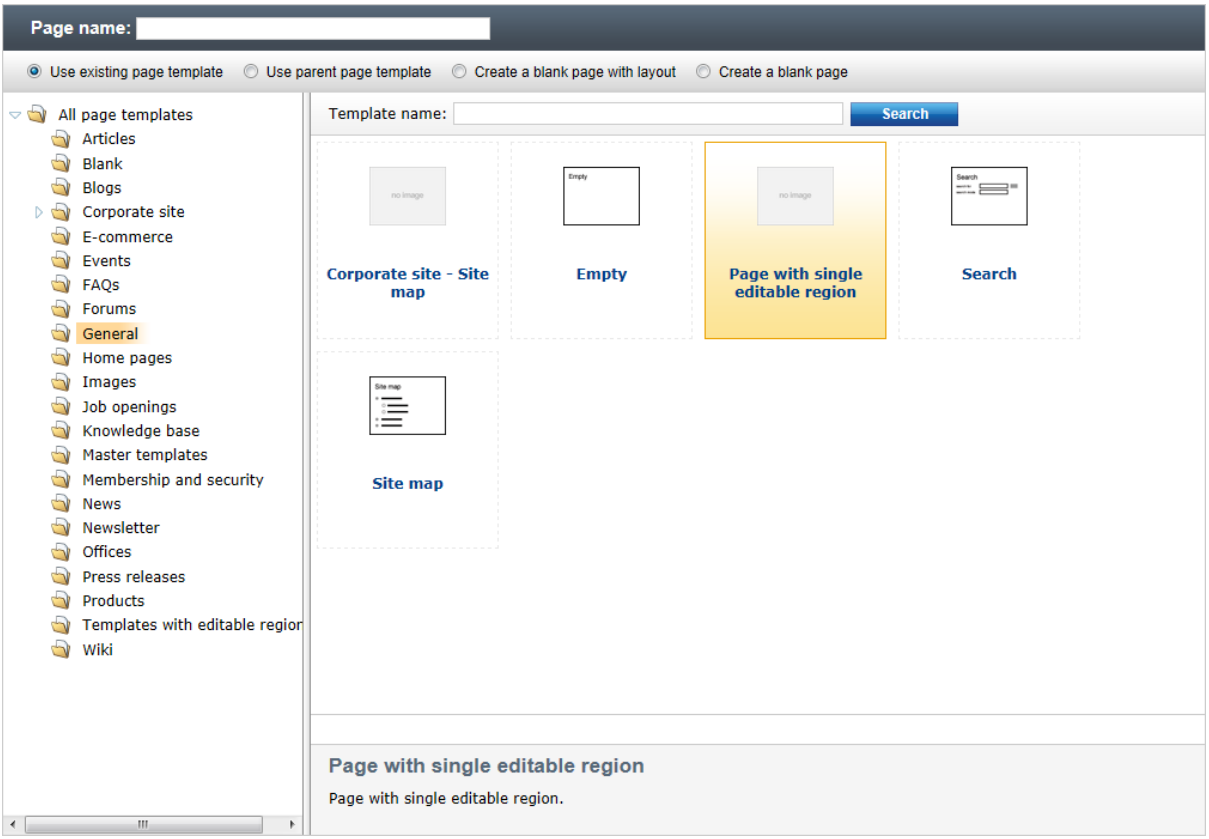
1. Go to **Content ->** switch to **Edit** mode on the main toolbar **->** select your new page created in the previous example **-> Properties -> Template**. You will see a page as in the following screenshot. As you can see, the current page is based on an ad-hoc template.



2. If you want to re-use this page template for some other page, click **Save as new template**. The **Save as new page template** dialog opens. Fill in values as shown in the picture below and click **OK**, then click **Save** on the **Template** dialog.



3. Now when you try to create a new page, you will be able to create the page based on this new page template:



Additionally, the template is now added to the content tree at **Site Manager -> Development -> Page templates**, where it can be configured.

What you did

You have converted the previously created ad-hoc page template that was specific only for a single page into a re-usable page template that can be used for any number of pages.

7.2.4.4 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. But 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.

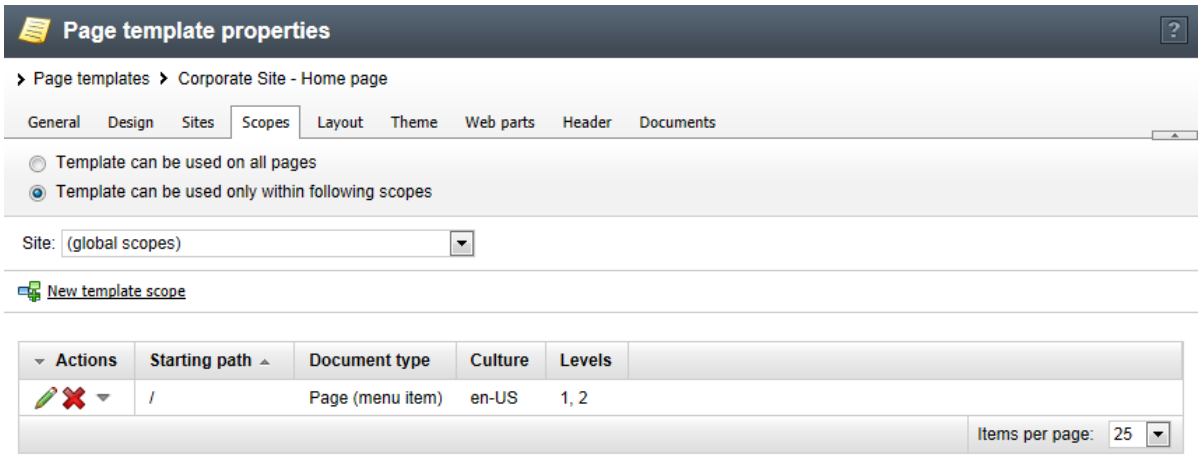
Template scopes can be defined on the **Scopes** tab when selecting a page template from the content tree at **Site Manager -> Development -> Page templates**. You have the following two basic options:

- **Template can be used on all pages**: the use of the page template is not limited and it can be used anywhere
- **Template can be used only within following scopes**: if selected, the page can be used based on the defined template scopes

If you choose the second of the options mentioned above, UI controls for managing the scopes will be displayed below the line.

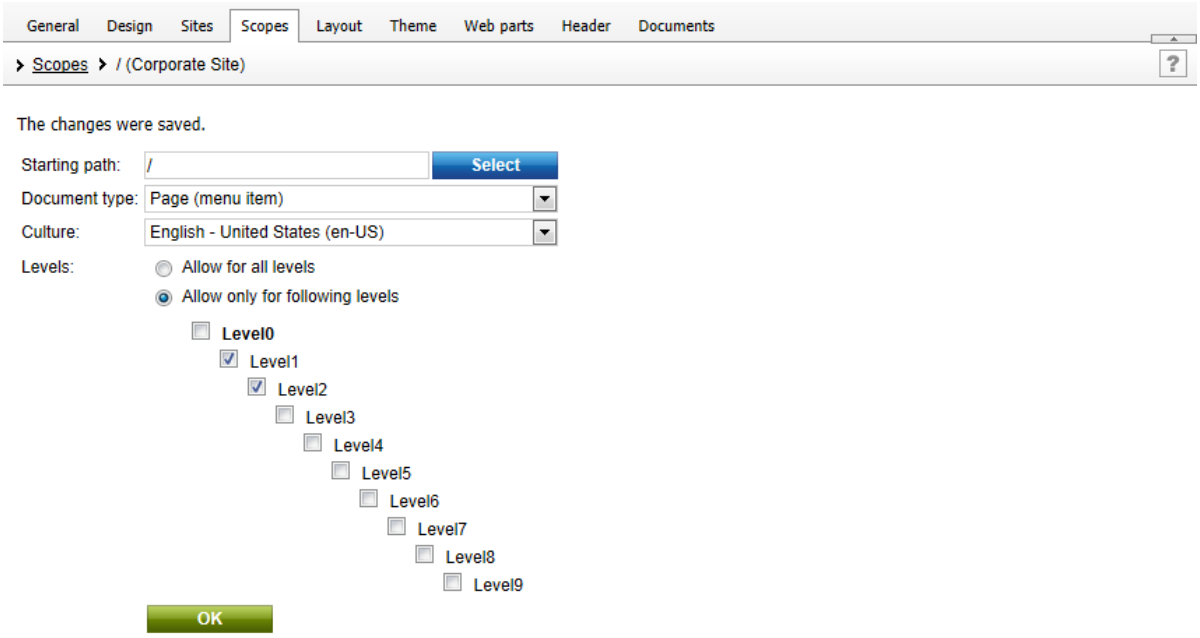
Based on the site selected in the **Site** drop-down list, you can display either site-related scopes or global scopes. Site-related scopes are applied only on the selected site, while the **(global scopes)** option indicates that the scopes are valid on all sites in the system.

Using the **New template scope** link, 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:

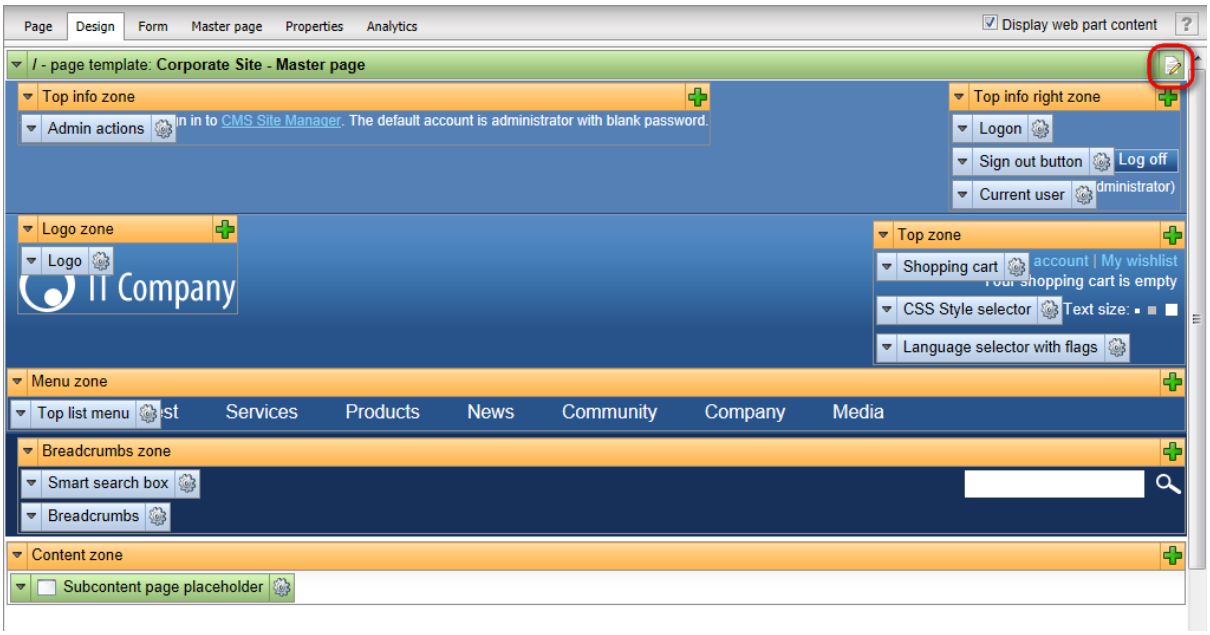
- **Starting path**: path in the content tree where the page template can be used; the page template cannot be used outside this path
- **Document type**: the page template can be used only for documents of the specified type
- **Culture**: the page template can be used only in the specified culture
- **Levels**: the page can either be used on all levels or only on the selected levels in the content tree

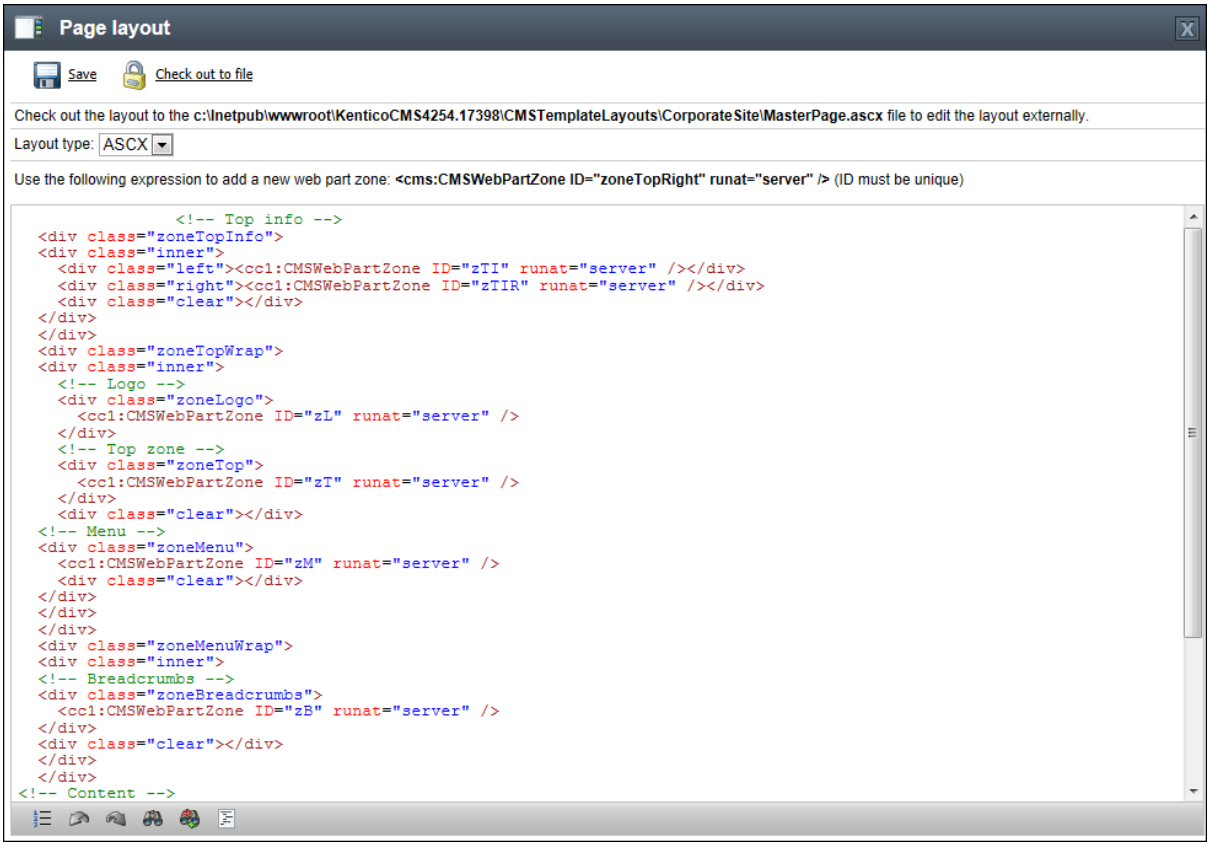


7.2.4.5 Page layouts

A **page layout** consists of the **HTML layout** of the page template and **web part zones** that specify regions where web parts can be placed. You will use page layouts to define the **layout and design of your site**.

The easiest way to edit a page layout is to switch to the **Design tab** of a page and click the **Edit layout** ( ) button at the top-right:





There are two possible ways that the layout's code can be processed, depending on the selected **Layout type**:

- **ASCX** - with this option, the code of the layout will support 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.
- **HTML** - if this option is selected, the layout code will be processed as basic HTML. This means that ASCX markup, such as controls or inline code, will not work when the layout is rendered.

Please note that for security reasons, the code of *ASCX* type layouts 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 HTML layouts will only process basic HTML syntax, they cannot be used to compromise the security of the website. If you wish to insert dynamic values into HTML layouts, you can do so by entering Kentico CMS Macro expressions or methods. An advantage of HTML layouts is that they do not need to be compiled, which means they can be used and modified even if the Virtual path provider is not available, such as in a precompiled or medium trust environment.

Web part zones can be inserted into the layout through the following tags/expressions:

ASCX

```
<cms:CMSWebPartZone ID="zoneTop" runat="server" />
```

HTML

```
{^WebPartZone|(id)zoneTop^}
```

The ID value must be unique for each web part zone within the given layout.

As you can see, the layout format is composed of standard HTML elements, which means **you have full control over how the page will be rendered**. You can choose between a table or CSS-based layout.

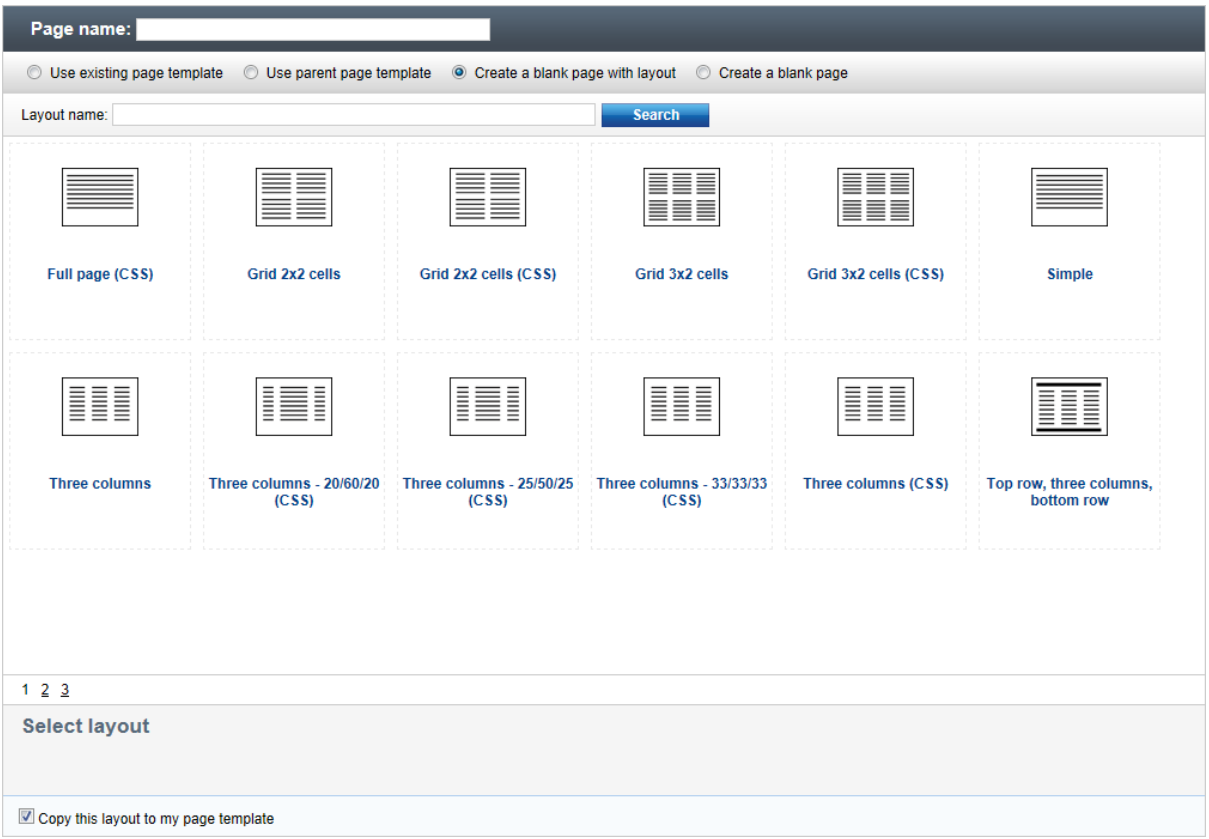
Editing the layout in your favorite editor

In some cases, editing complex code in a web editor may not be very comfortable. You can use the **Check out to file** button to save the layout code to the disk and open it in your favorite editor.

Please note: The file is saved on the disk where the web application is running.

Managing pre-defined page layouts

When you are creating a new blank page and select the **Create a blank page with layout** option, you can choose from a number of pre-defined page layouts:



The **Copy this layout to my page template** option allows you to decide whether you will be using the layout as shared or if you want to create an ad-hoc copy of the layout, specific for your page template only. If you leave the option disabled and make changes to the layout code, the changes will affect all pages that use the shared page layout. Therefore, it is better to leave the option enabled in most cases.

You can manage the pre-defined page layouts in **Site Manager -> Development -> Page layouts**. You can configure the following properties:

Layout display name	Name of the layout which will be displayed in the page layout list.
Layout code name	A unique name that will be used to identify the page layout (e.g. in code).
Layout description	Can be used to enter an optional text description for the page layout.
Page layout thumbnail	Upload field for the layout preview image (available only when editing an existing layout). This image is used in the page layout selection dialog shown when creating a new blank page as seen in the image above.
Layout type	Allows you to choose from two possible types of layout code: • **ASCX** • **HTML** Please see the text above to learn about the differences between these two layout types.

Layout code	This is where you can enter the actual code that will be used for the custom page layout. The format used when the code is processed depends on the selected **Layout type**.
CSS styles	This field becomes available if you click the Add CSS styles link below the layout's code editor. Here you can define any CSS classes used within the code of the layout. Once the page layout is assigned to a page template, the specified styles will be loaded on all pages that use the given template. Please note that this requires the **Allow CSS from components** setting to be enabled in **Site Manager -> Settings -> System -> Performance**. If the entered styles require any additional files (such as images), you can add them on the **Theme** tab of the layout's editing interface. For more information about CSS styles of page components, please see the Development -> CSS stylesheets and design -> CSS for page components topic in the Developer's Guide.

Sample layout code

The following sample page layout code uses a table to define a two-column layout:

```
<table>
  <tr>
    <td>
      <cc1:CMSWebPartZone ID="zoneLeft" runat="server" />
    </td>
    <td>
      <cc1:CMSWebPartZone ID="zoneRight" 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;">
    <cc1:CMSWebPartZone ID="zoneLeft" runat="server" />
  </div>
  <div style="width: 50%; float: right;">
    <cc1:CMSWebPartZone ID="zoneRight" runat="server" />
  </div>
</div>
```

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, please read the Development -> Web parts -> Layout web parts chapter of this guide.

7.2.4.6 The master page concept

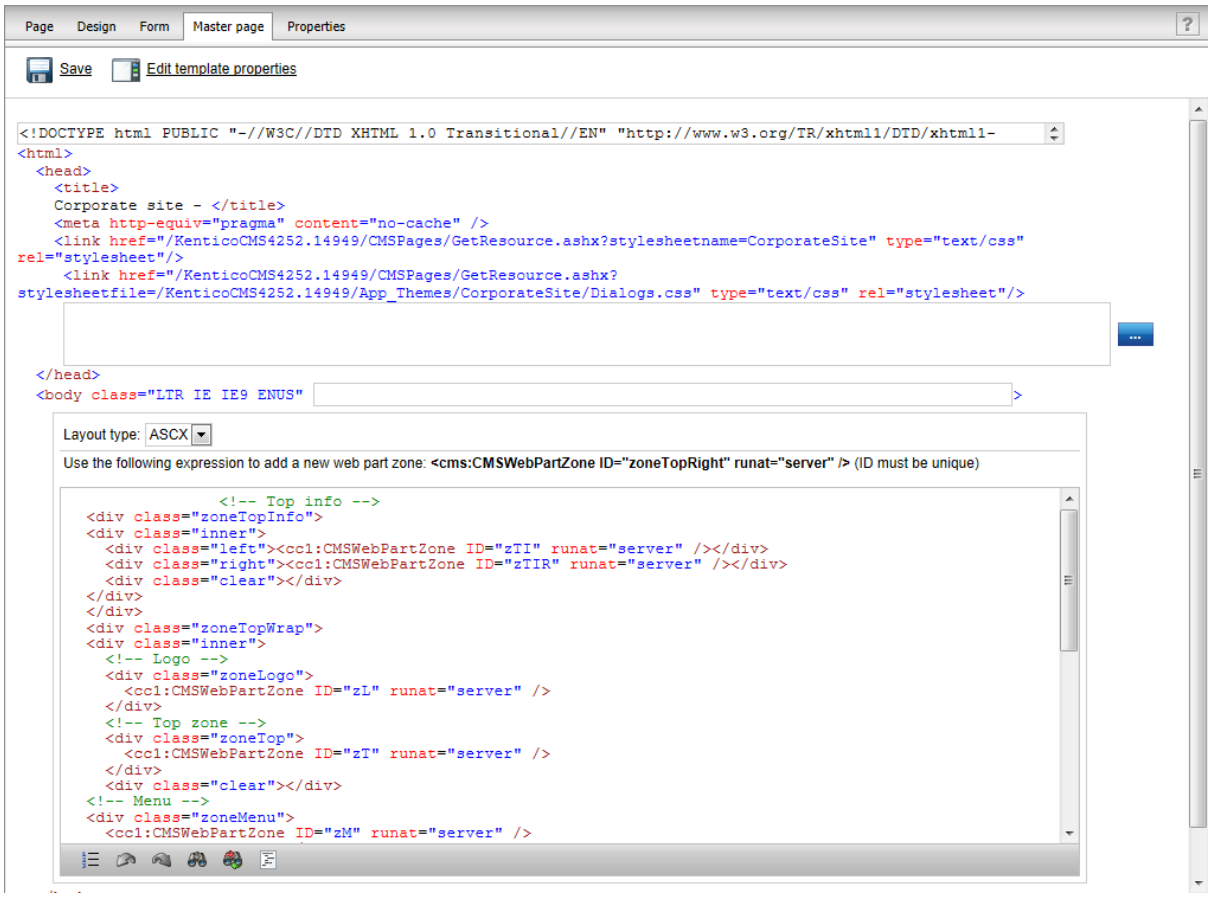
Master pages represent the powerful concept of sharing the same header and footer for all pages on the website. It allows you to manage repeated items, such as the site logo, main menu and footer content in a single place.

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:

7.2.4.7 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 at **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. In addition, there's a special **Master page** tab available only for master pages:



As you can see, you can define the following sections of the HTML code of the master page. This code is used for all pages that inherit content from the master page:

- **DOCTYPE** - here you can enter any code that needs to be placed at the beginning of the file, typically the DOCTYPE definition
- **HEAD** - here you can put any HTML code that needs to be placed inside the **<head>** element of the page.
- **BODY** - here you can add custom attributes to the **<body>** element.
- **Master page code** - this is actually the HTML layout of the master page template. This is the same code that you can edit on the **Design** tab, by using the **Edit layout** ( ) button.

Please be aware that the code between the editable sections is of informative character only and may not be identical to the actual rendered code.

> **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.

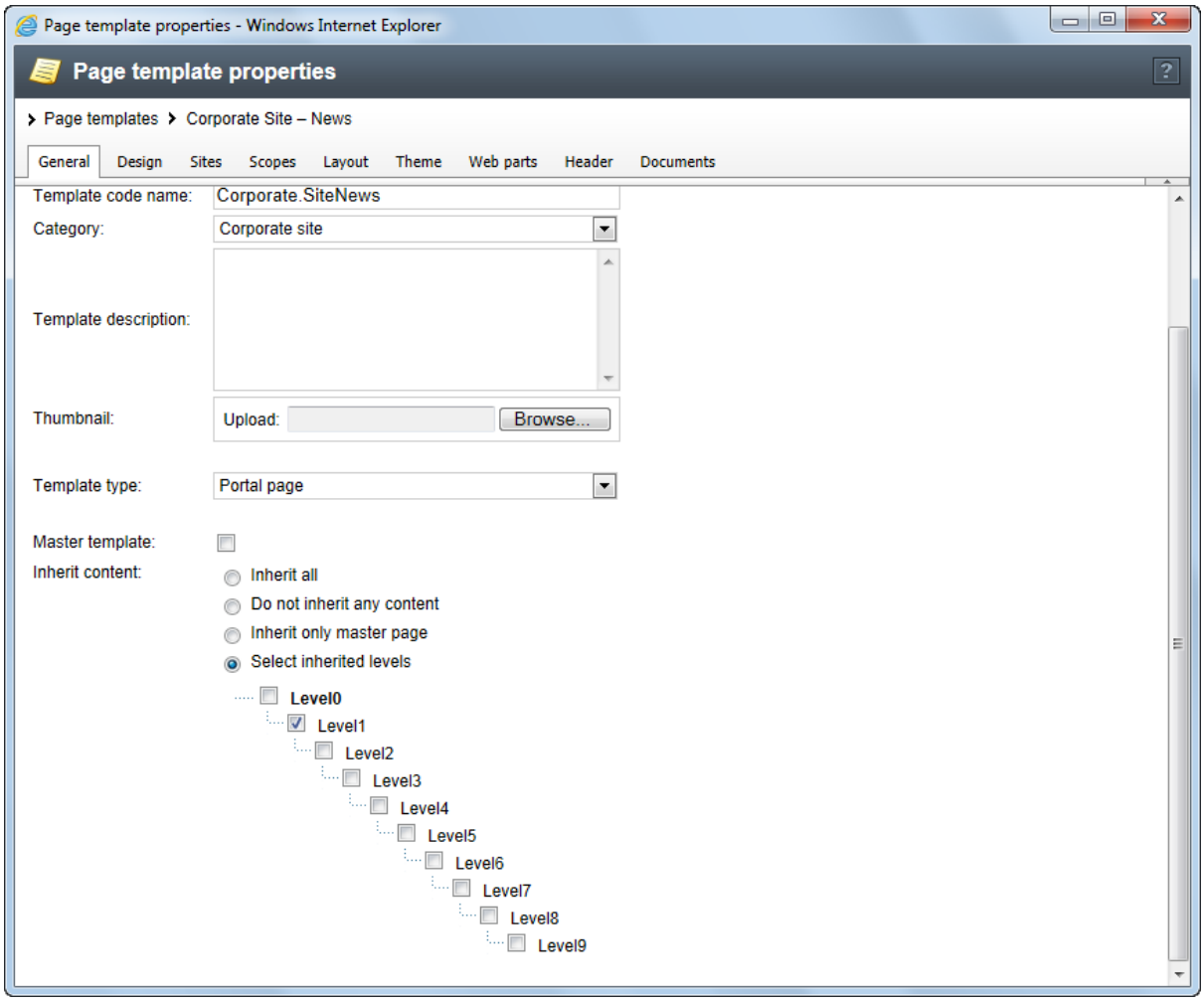
7.2.4.8 Visual inheritance

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.

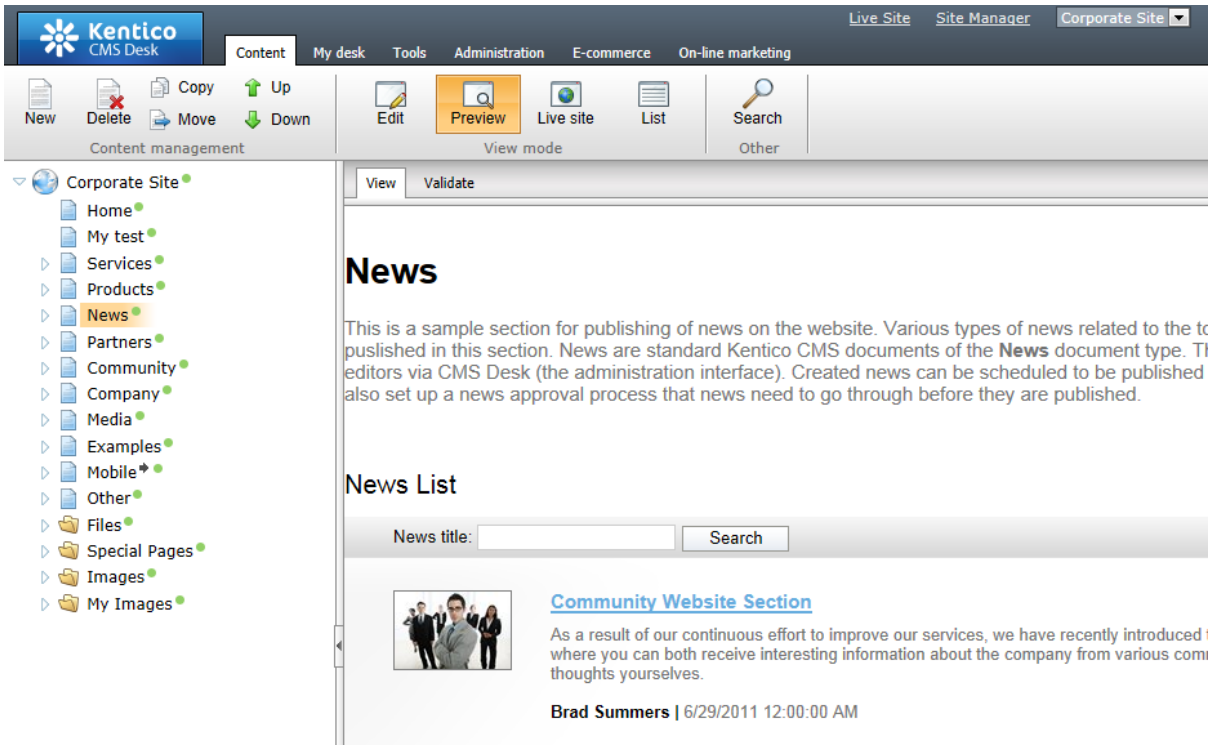
In some cases, you may want to hide some parts of the parent page. There are several ways to achieve that:

Using the "Inherit content" property of the page template

Click the **/News** page and click **Properties -> Template -> Edit template properties**. Now you can set the **Inherit content** value to **Selected inherited levels** and check only the **Level1** box. It 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 save the changes.

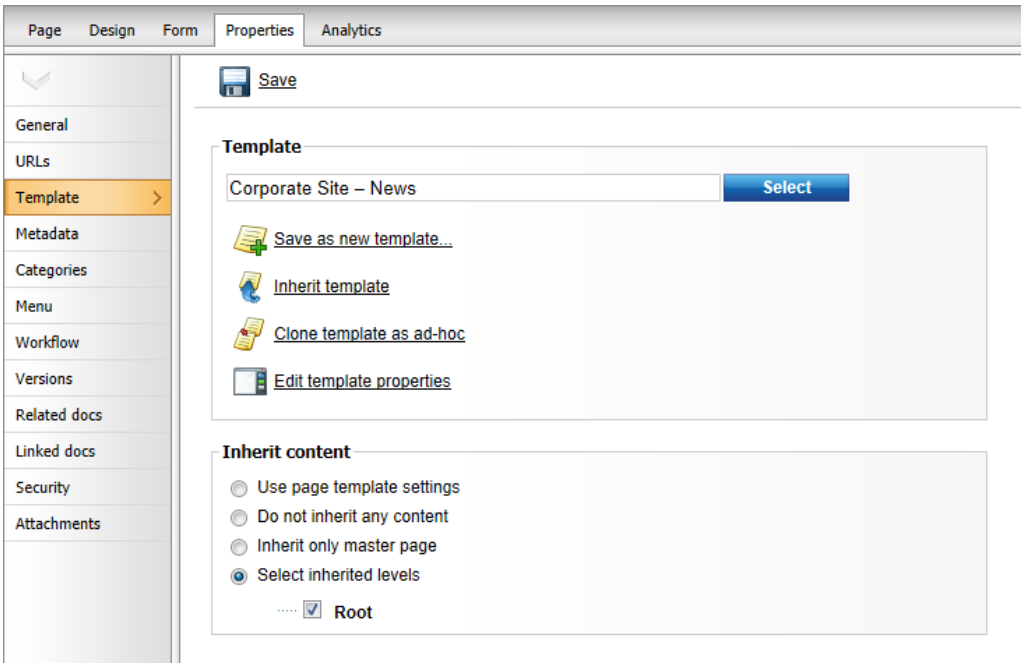


The page will now look like this:



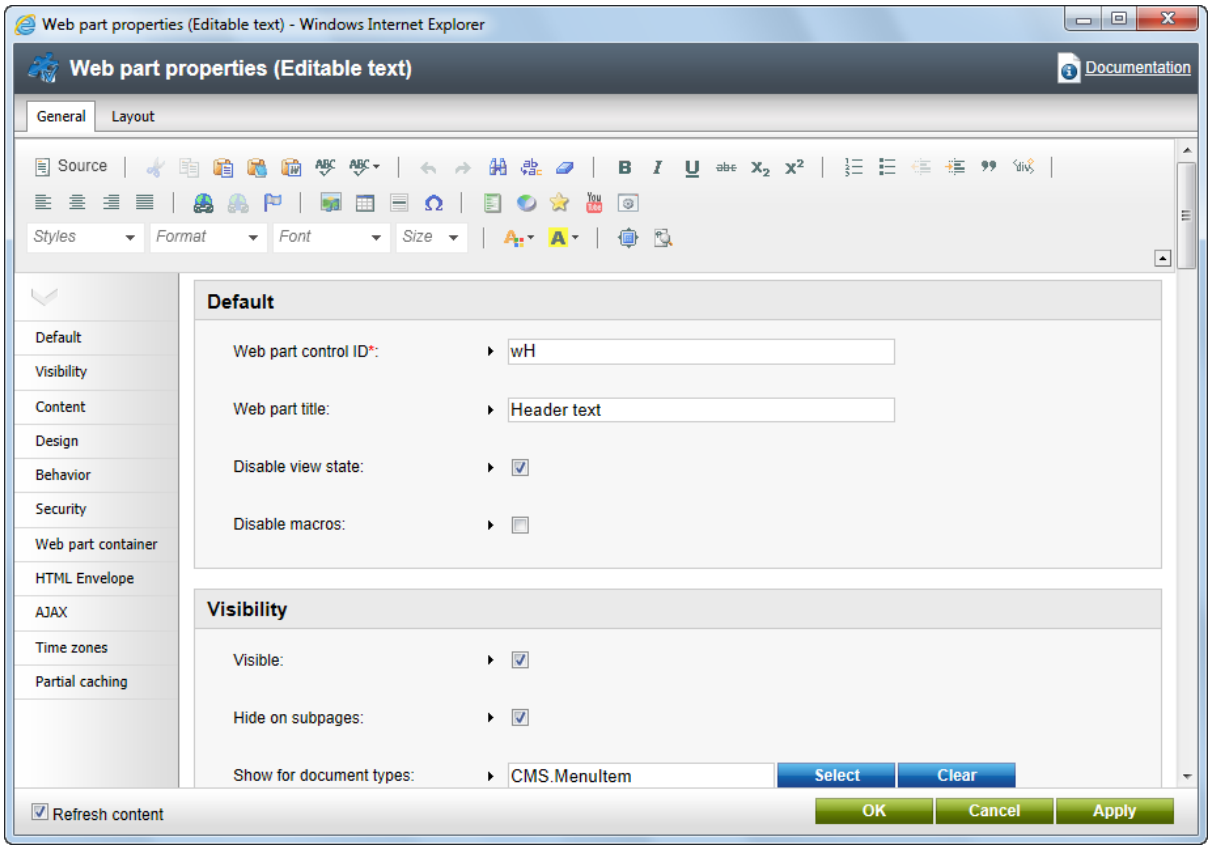
Set the **Inherit content** value back to **Level0** for now.

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 the page override the page template settings:

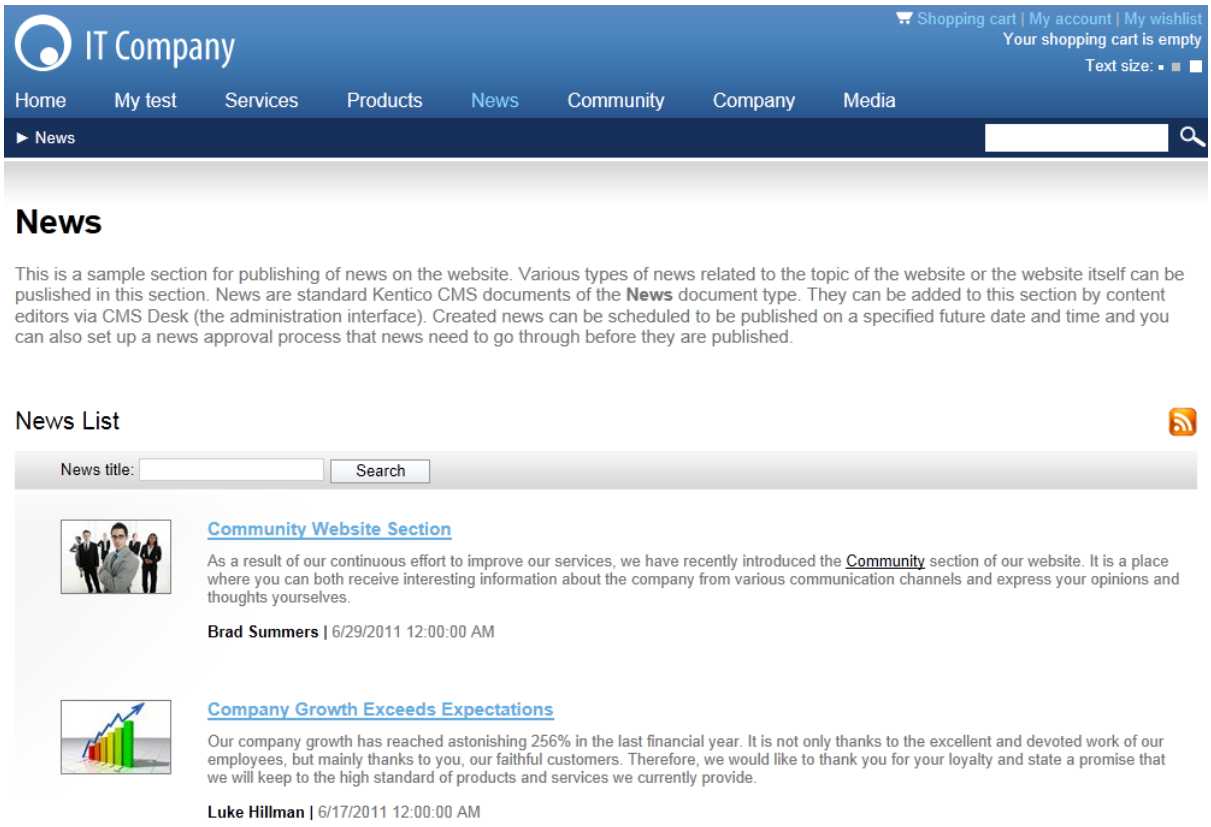


Using the "Hide on subpages" web part property

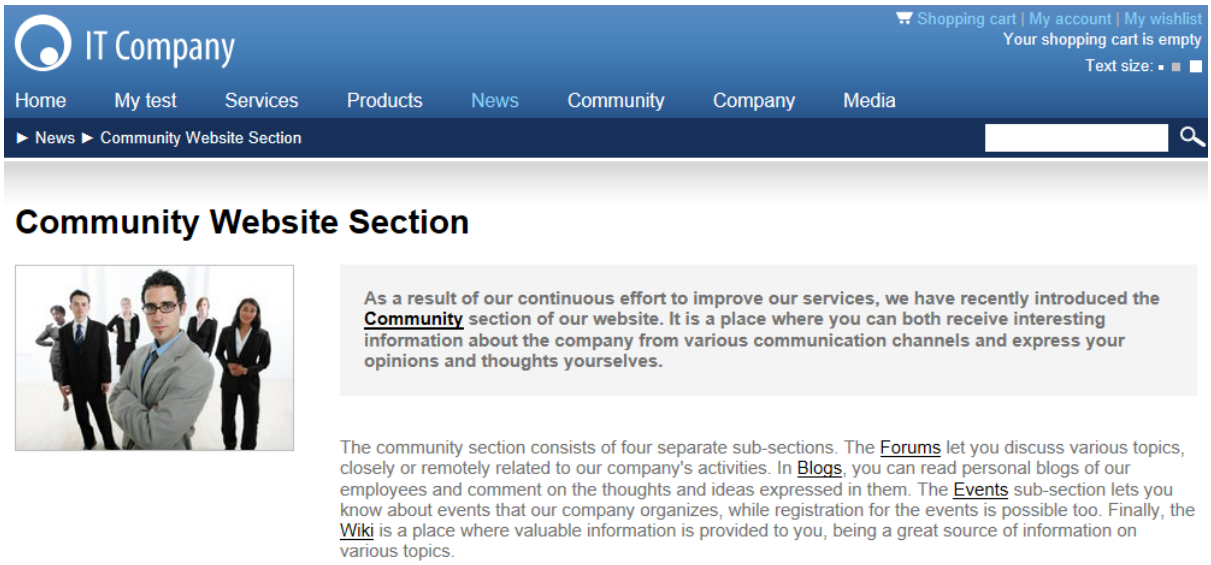
Click **/News**, switch to the **Design** tab and click the **Configure** ( ) button of the **HeaderText** web part. The web part has the property **Hide on subpages** set to true:



Click **Cancel** and click **Live site**. Please note that when you display the list of news, the title **News** is displayed:



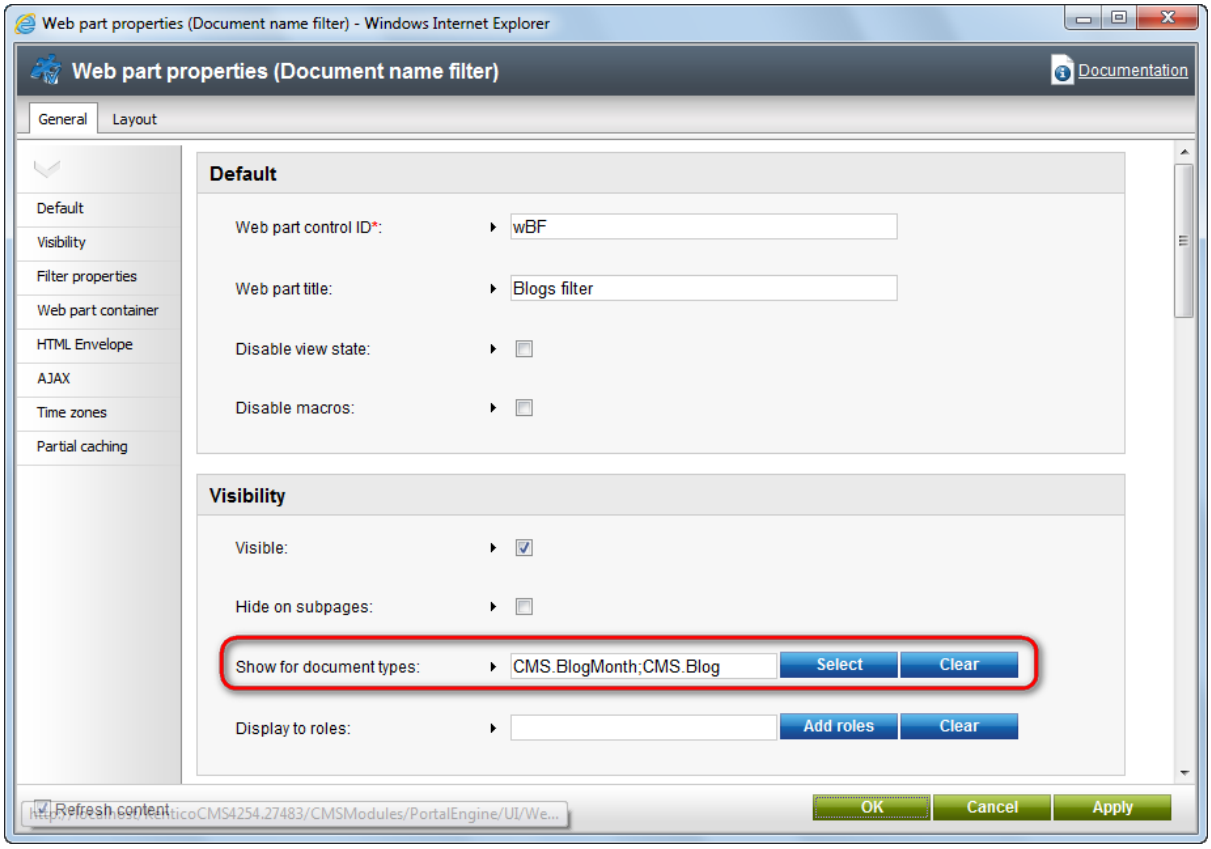
If you go to some particular news item, the title is hidden:



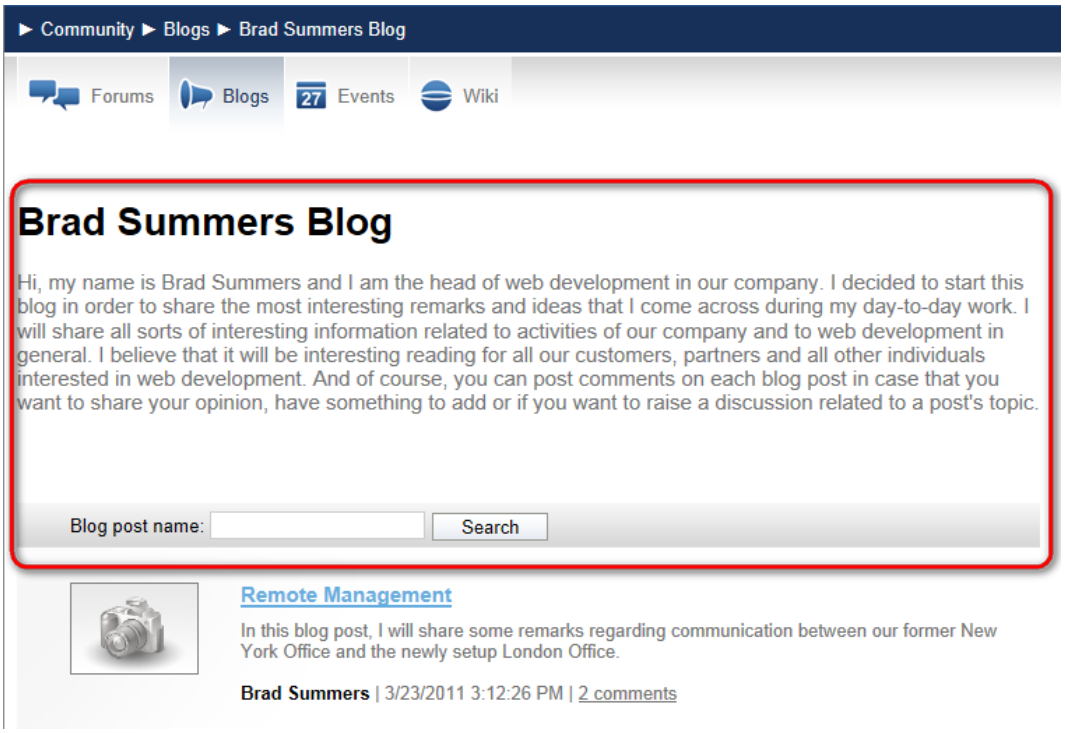
This is ensured by the **Hide on subpages** property that hides the web part displaying the **News** title.

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.



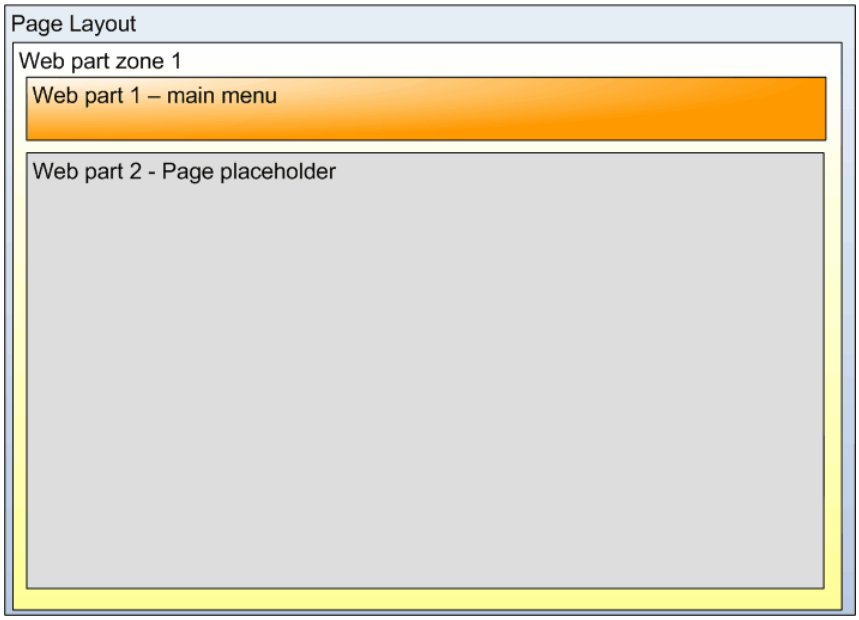
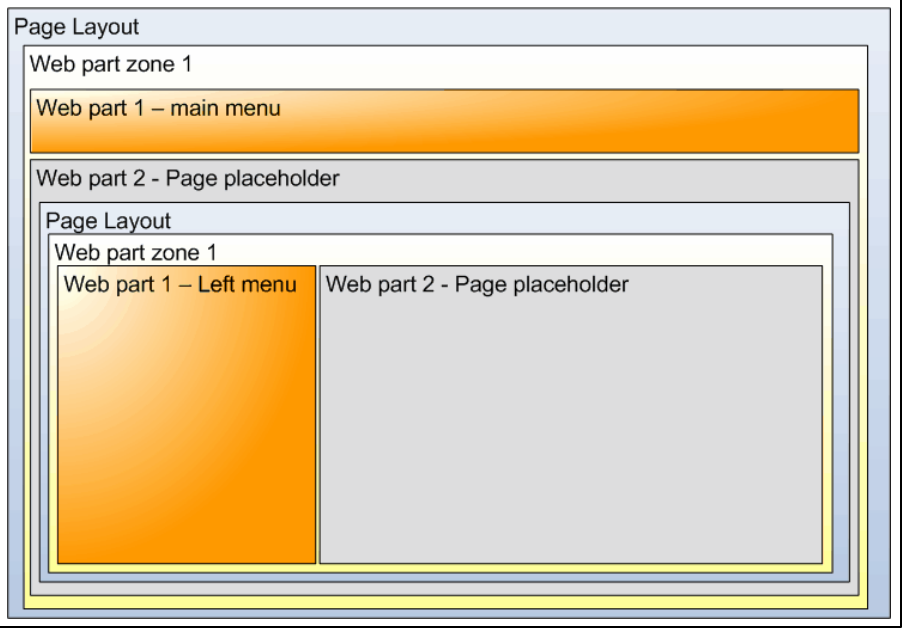
When you display some particular blog post, the web parts are not displayed because *CMS.BlogPost* is not among the enumerated document types.

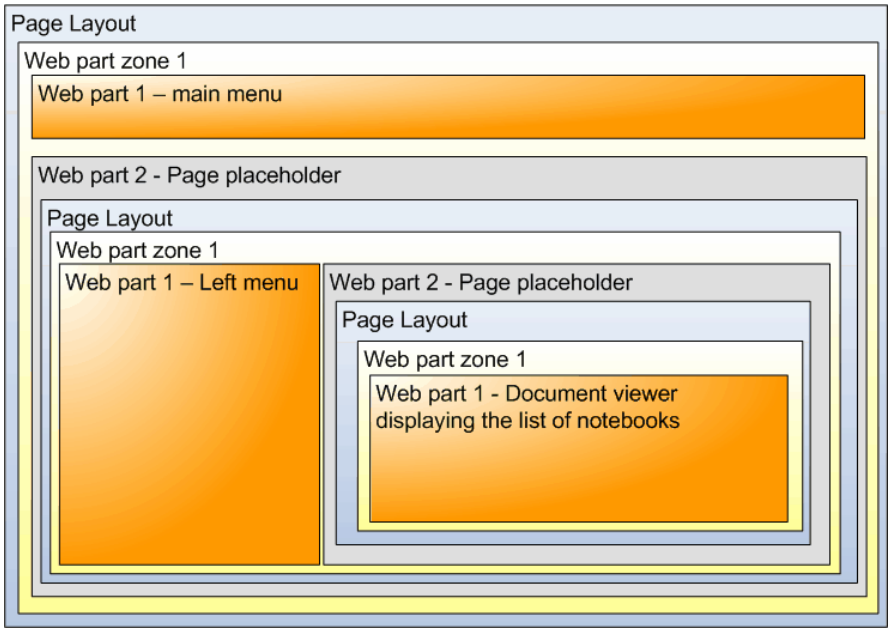
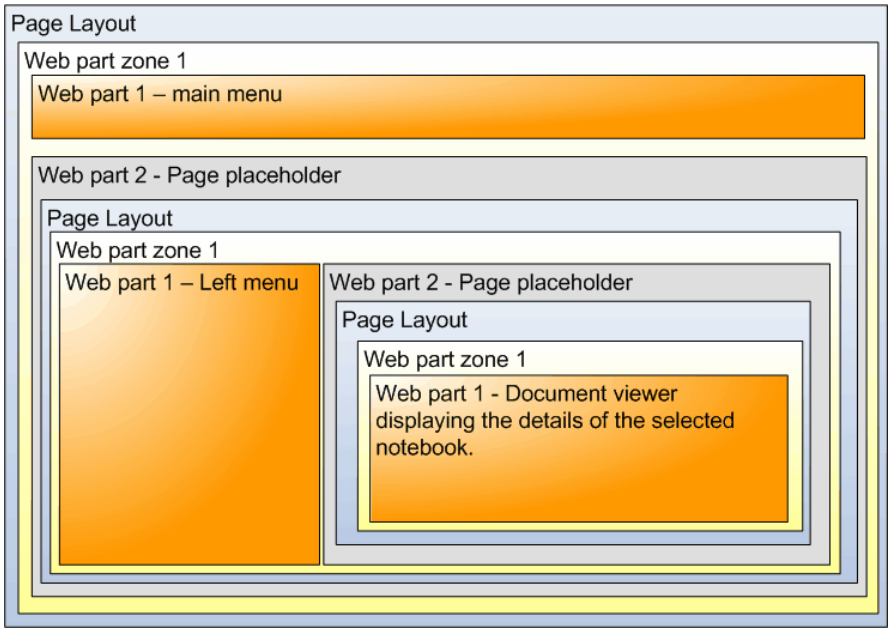
You have learned how to break inheritance of content, hide content on subpages and how to display content based on the current document type.

7.2.4.9 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.*	

7.2.4.10 Managing page template catalog

You can manage page templates in **Site Manager -> Development -> Page templates**. 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 page in the content tree of **CMS Desk**, and using **Edit -> Properties -> Template -> Edit template properties**.

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.
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.
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*

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 specify custom HTML code that will be added to the **<head>** element (it will be placed between the `<head></head>` tags) of all pages that use this page template. This is useful if you need to link some additional CSS or JavaScript files.

Documents tab

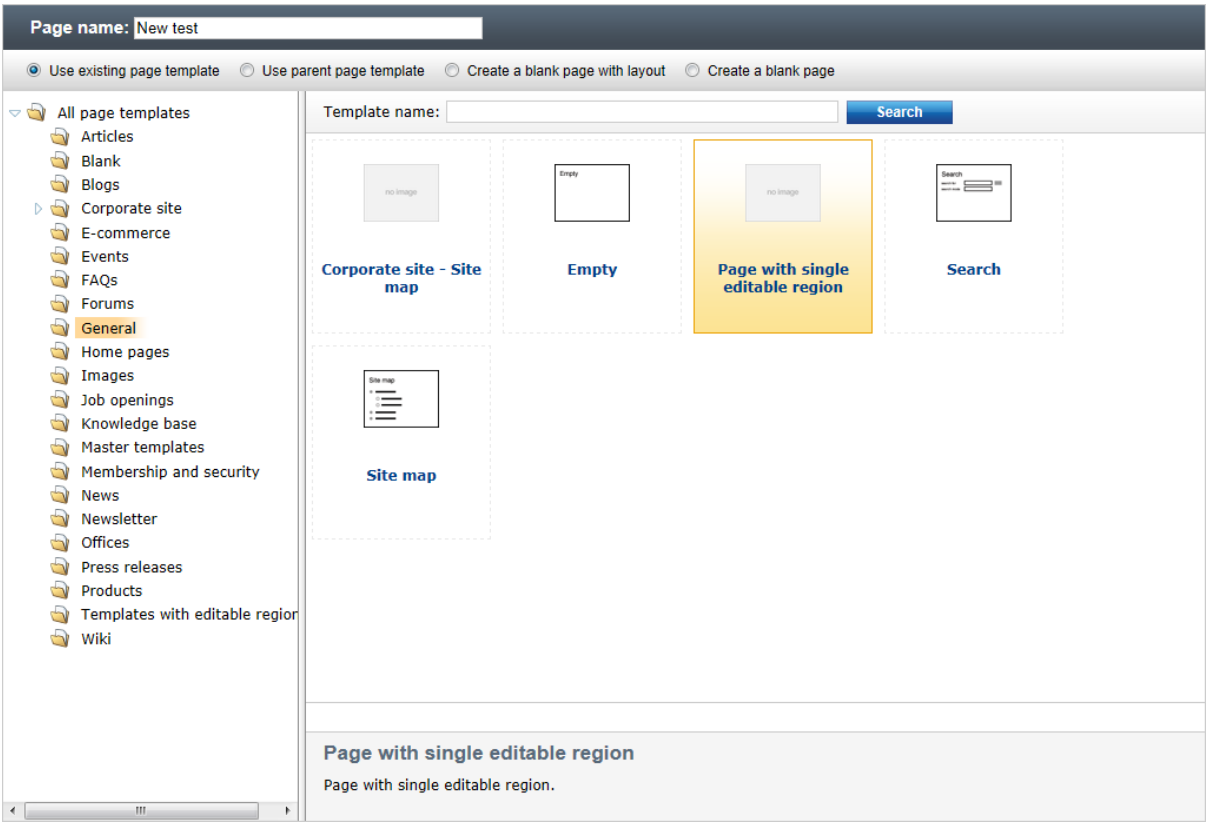
On this tab, you can see a list of documents (pages) that are using the page template. By clicking a

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**.

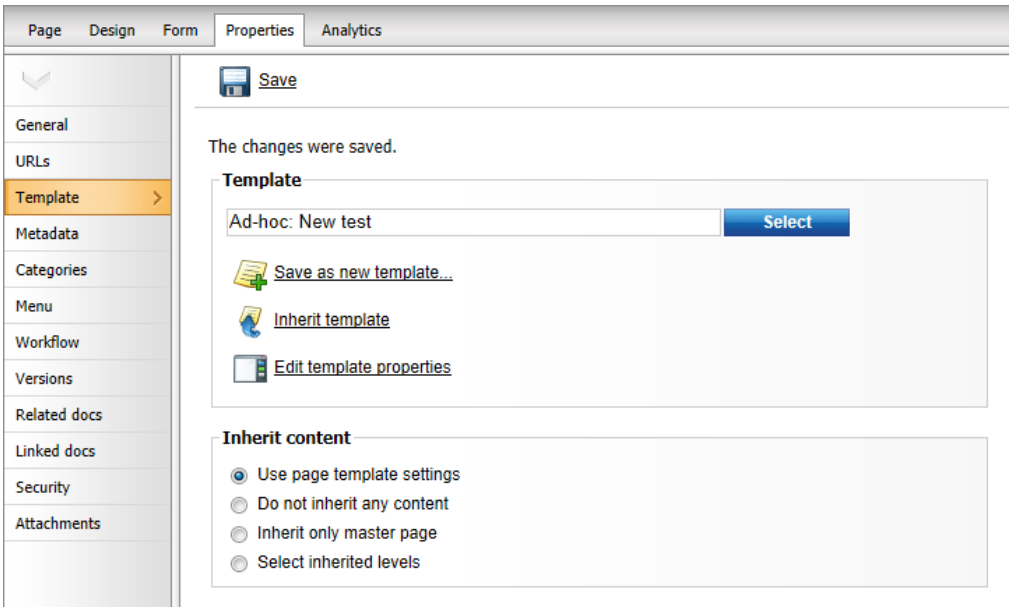
7.2.4.11 Cloning and modifying a page template

You may need to create a new page that will be similar to existing pages, but with some minor modification of the page template. Imagine you want to display the editable region from the previously created page template in a container box.

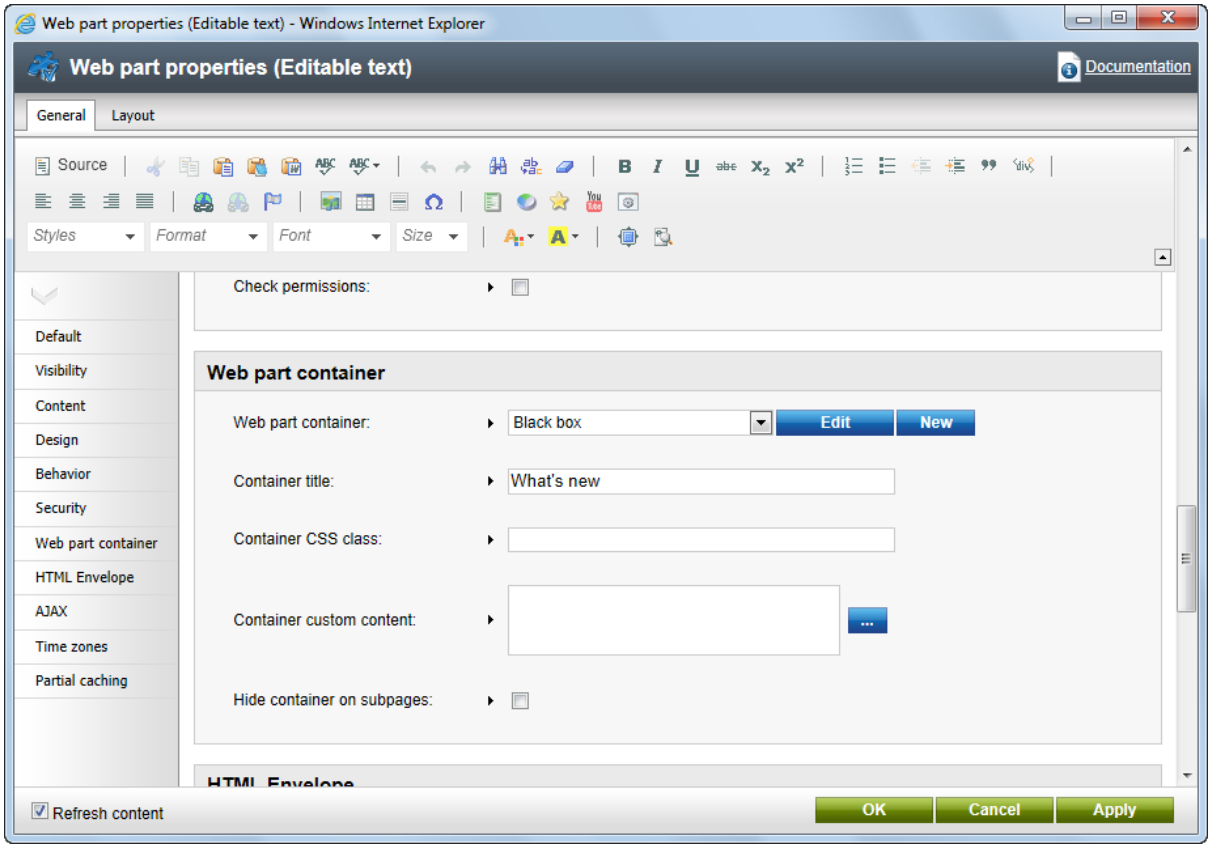
Instead of creating the page template from scratch, simply create a new page based on your existing page template.



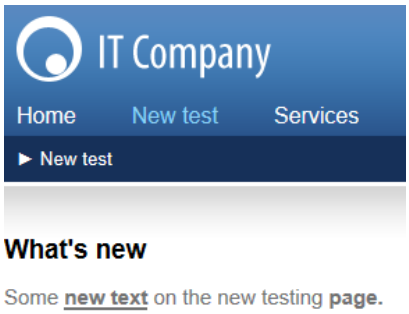
Then choose the page, click **Properties -> Template**, click **Clone as ad-hoc template** and click **Save**. A new ad-hoc template is now created and you can edit it without modifying the existing pages based on the original template:



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 **Web part container title** field:



Click **OK**. Switch to the **Page** tab and enter some text into the editable region. Click **Save** and click **Live site** in 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 it uses the original design and it wasn't affected by the change we made to our new copy of the page template.

Re-using the adjusted template again

If you want to re-use the new, adjusted page template, you can click the **Save as new template** button in the **Properties -> Template** dialog as you did previously.

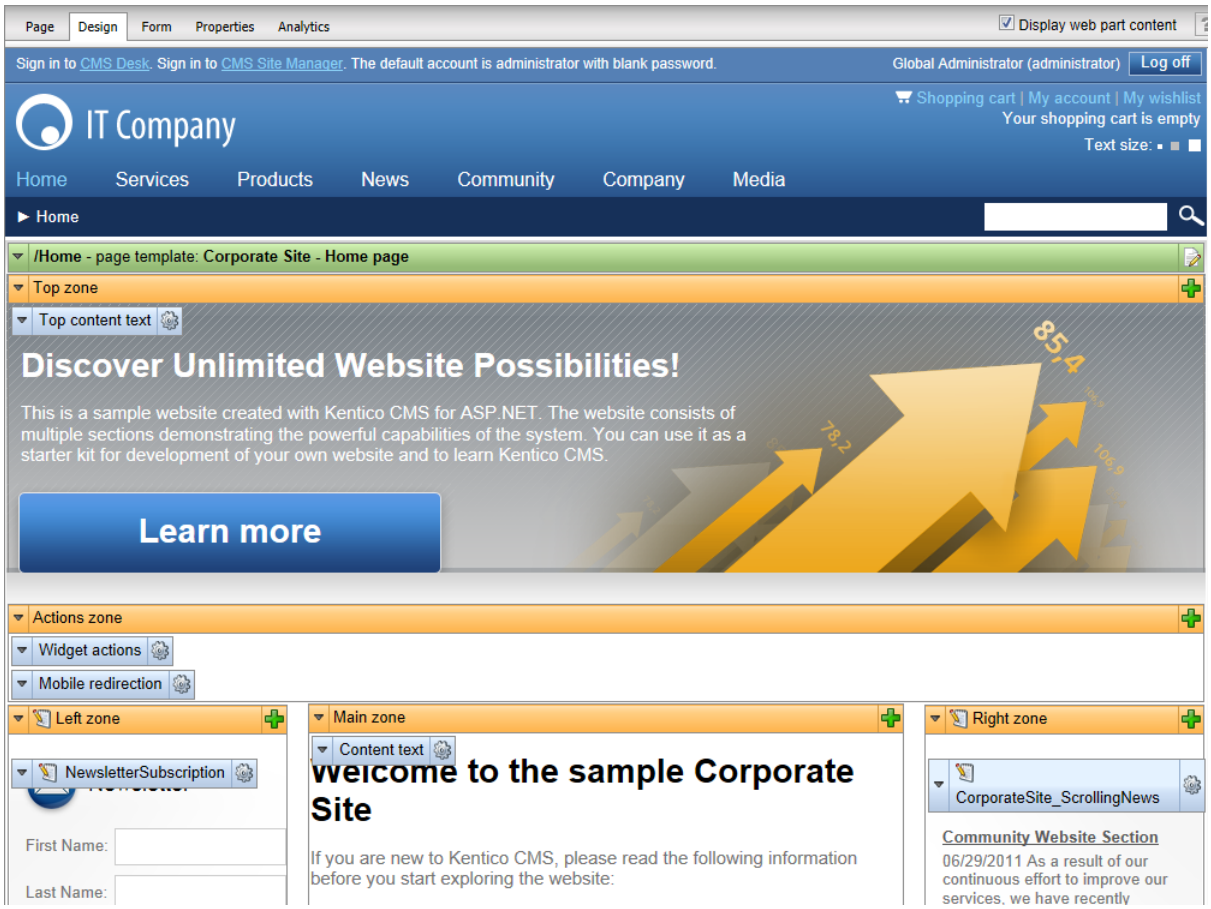
Important!

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 is deleted as well and **cannot be restored**.

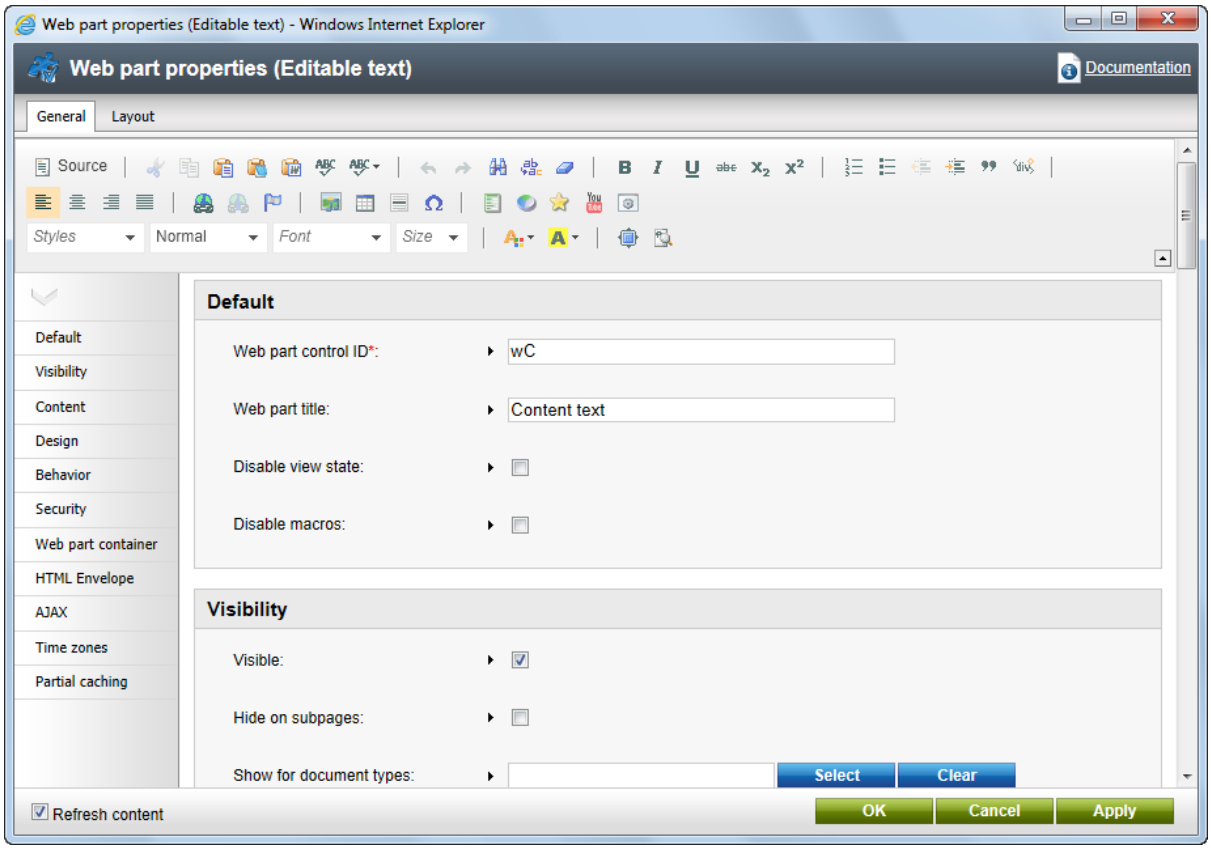
7.2.4.12 Using and configuring web parts

Web parts can be used on both portal page templates and ASPX page templates. However, with ASPX page templates, you lose the friendly user interface and need to set the properties in Visual Studio.

Go to **CMS Desk -> Content**, click some page and choose **Design** in the toolbar. You will see the page template structure like this:



Any web part can be moved up/down or to another web part zone and its properties can be edited using the **Configure** ( ) button. This opens the **Web part properties** dialog as shown below:



When you click **OK**, any changes are saved and the window closes. If you click **Apply**, the changes are saved, but you can continue modifying the properties. All changes are applied immediately. Page templates are not connected with workflow, but it is possible to use object versioning to keep track of changes made to templates, including their web part content (and roll back to previous versions if necessary). Please read the Development -> Object versioning chapter to learn more.

Impacts of changing page templates

When you change a re-usable page template that is used by several pages, the changes will appear on all page templates.

If you need to modify only a single page structure, you need to clone the page template as an ad-hoc template or as a new page template. See also Cloning and modifying a page template.

7.2.4.13 Web part binding (obsolete)

Obsolete feature

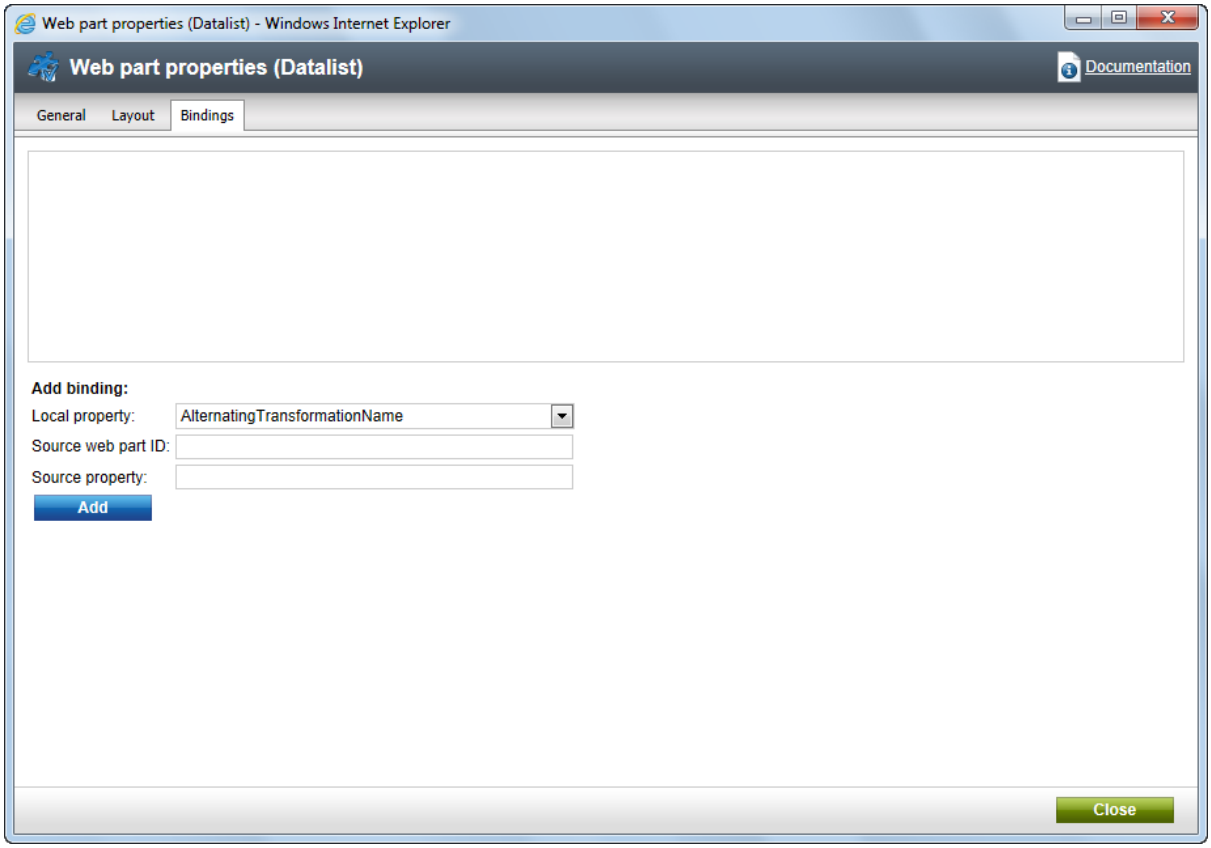
This feature is now obsolete. If you need to bind the behavior of some web part to another web part value, please use URL parameters or create a user control that

contains both web parts and write custom code that will ensure the communication between the web parts.

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:



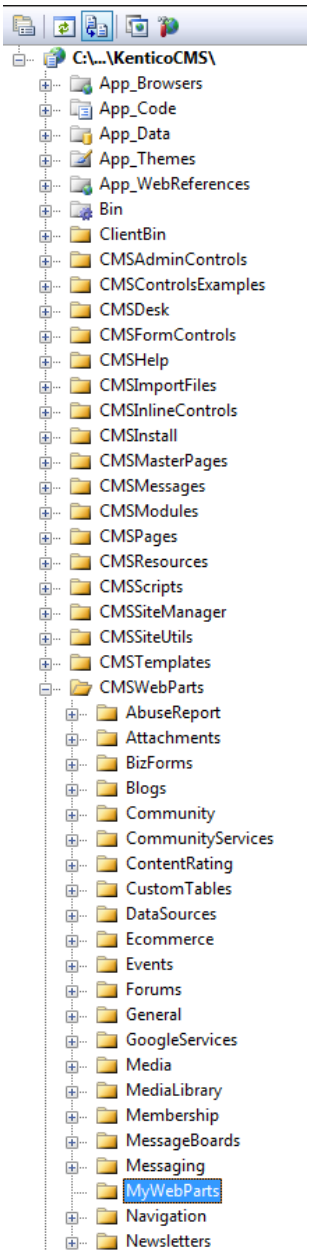
Alternative option

If binding doesn't work as expected, you can alternatively create your own user control, place the required web parts on it and manage their interactions using your own code. You can find more details on creating your own user control in the Adding custom code to the page topic.

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.TreeEngine;
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.TreeEngine;
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:

- **Web part display name**: Product selector
- **Web part code name**: ProductSelector
- **Web part file name**: MyWebParts/ProductSelector.ascx

5. Create another new web part:

- **Web part display name**: Product viewer
- **Web part code name**: ProductViewer
- **Web part file name**: 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.

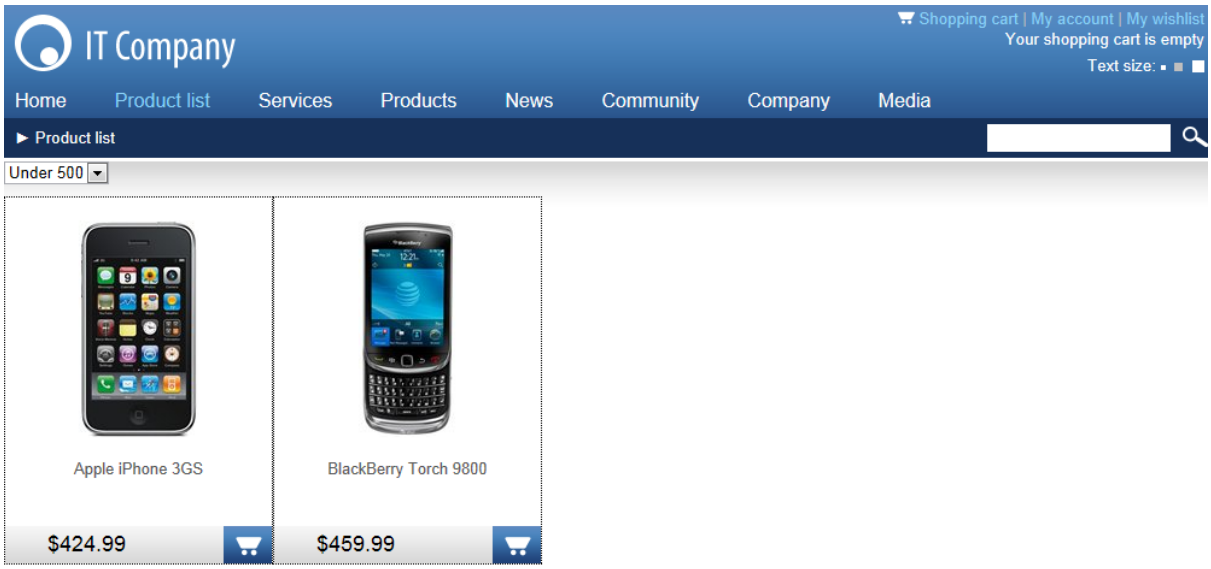
8. Next, add the **Product viewer** web part, set the following properties on the **General** tab:

- **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.

7.2.4.14 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.

You can find more details in the Modifying web part behavior topic.

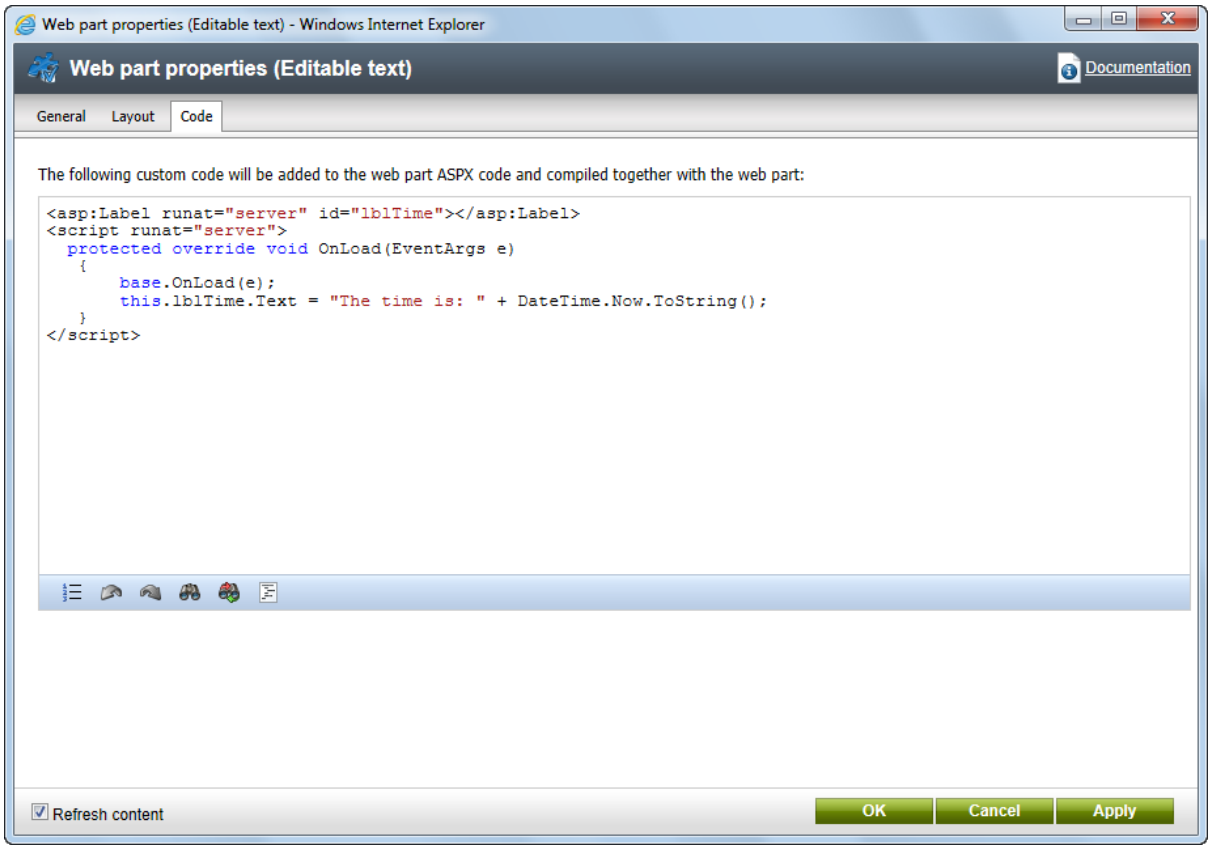
If you still need to use this feature (for backward compatibility), you need to enable it by adding the following parameter to your web.config file, to the **appSettings** section:

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

Limitations when you use 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 medium trust environment. If this is an issue for you, you can use an alternative solution described in the Modifying web part behavior topic.

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.

7.2.4.15 Common web part properties

Many web parts use the same or similar properties. The following table summarizes the most important properties found in most 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

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	cms.news;cms.event

	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.	cmseditors;customrole

Web part container

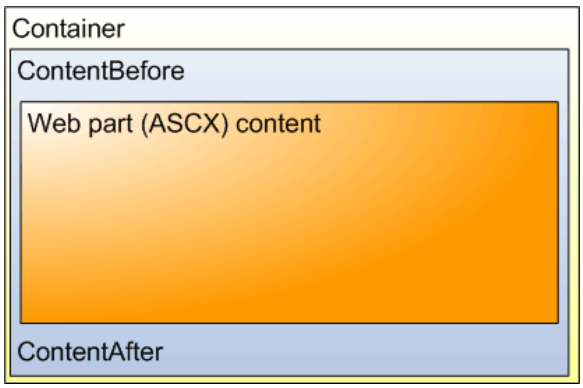
Property Name	Description	Sample Value
Web part container	Specifies the name of the container (box) to be displayed around the web part. Only the containers defined at **Site Manager -> Development -> Web part containers** can be selected. The selected container can be edited directly by using the **Edit** 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.	

HTML Envelope

Property Name	Description	Sample Value
Content before	HTML content placed before the web part.	<table style="background-

	Can be used to display a header or add some encapsulating code, such as * * or *<table>* elements to achieve the required layout.	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 documents to be displayed. See Appendix A - Path expressions for details.	/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	Name of the query to be used for retrieving data from the Kentico CMS Database. The queries can be defined in **Site Manager -> Document types -> edit some document type -> Queries**.	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	cms.article;cms.news;cms.menuitem

	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 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	If the displayed data contains multiple links	

documents	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.	

7.2.4.16 Path and macro expressions in web part properties

Many web parts use the **Path** property to specify what content from the content tree should be displayed. You can find all path format options in Appendix A - Path expressions.

Also, all web part properties support macro expressions that allow you to insert a dynamic value instead of a constant in the property. Such a dynamic value is evaluated at run-time and allows you to specify context-dependent values. See Development -> Macro expressions for details.

7.2.4.17 Adding custom code to a portal page template

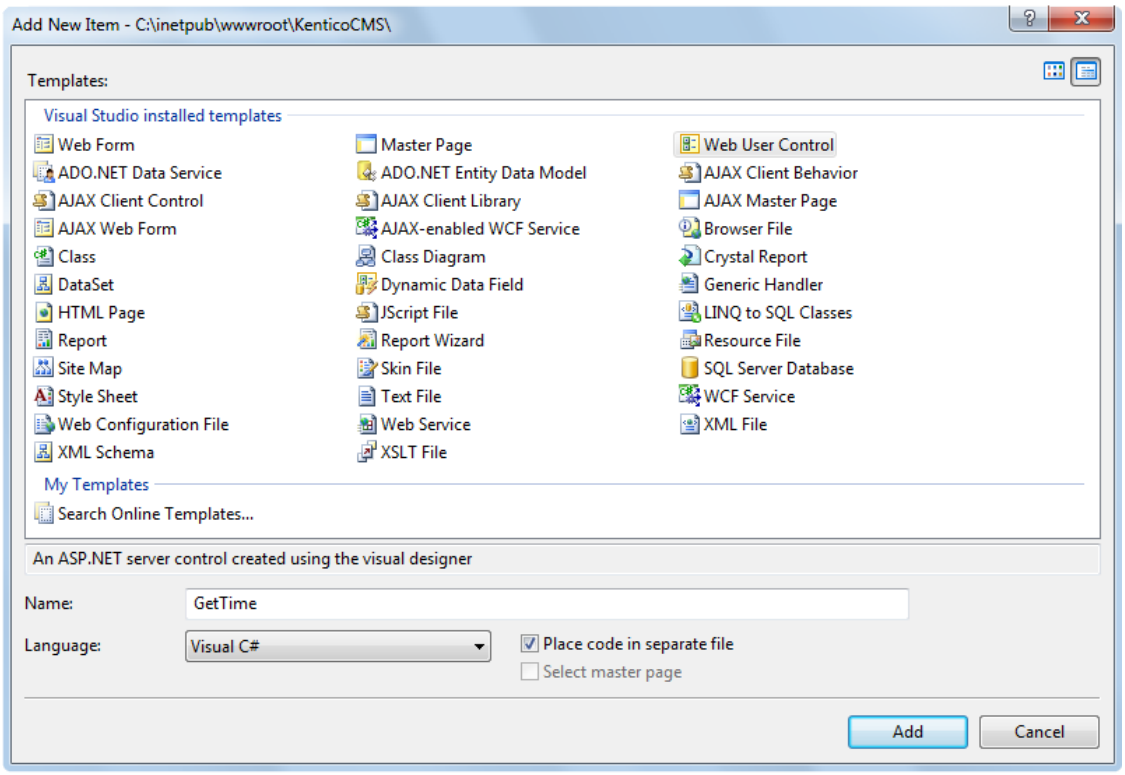
The easiest way to insert custom code into a portal engine-based website is using standard ASCX user controls. This chapter will show you how to do this. If you're not familiar with Visual Studio development, you can skip this chapter.

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.



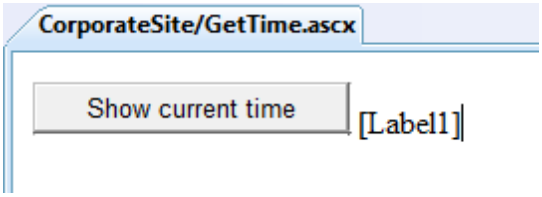
Click **Add**. Switch to the **Design** tab and drag and drop the following controls and set their properties:

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. It's not necessary to compile the project - the user controls are compiled at run time.

Save all changes.

Adding the user control on the page

Sign in to **Kentico CMS Desk**, click the **Home** page and click **Design**. Remove the web part in the **zoneCenter** zone and click the **Add web part** () button in this zone. Choose the **General/User control** web part. Enter the following value in 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**. Click the **Live site** mode and now you can see the user control in 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 case you need to create re-usable, parameterized user controls. Web part development is described in the Development -> Web parts -> Developing web parts topic.

7.2.4.18 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#]

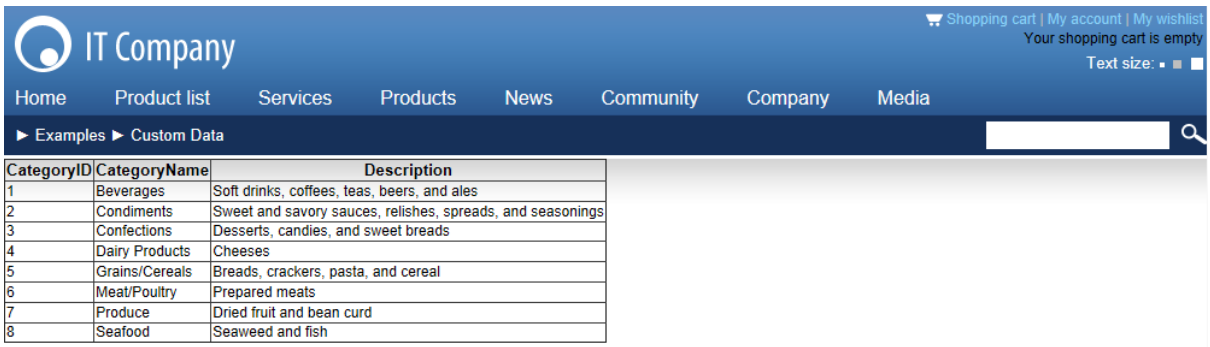
```
// create sql connection - you could use Oracle or OLEDB provider as well

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.

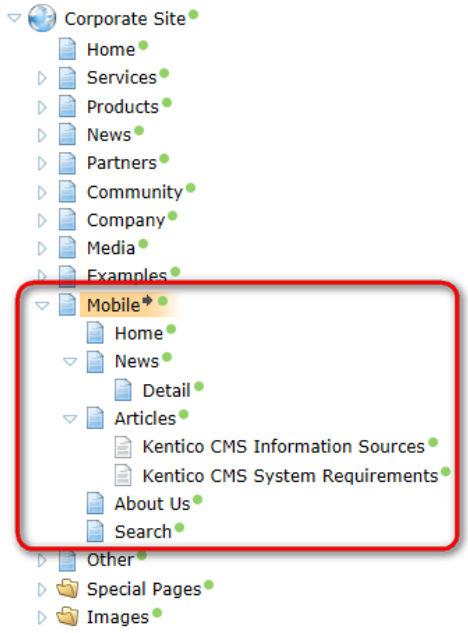
7.2.4.19 Developing sites for mobile devices

Kentico CMS allows you to create a dedicated website section where users accessing the site from mobile devices will be redirected.

Example on the Corporate Site

An example of such a website section can be found on the sample **Corporate Site**, under the **/Mobile** node. The **Mobile** page works as a master page for the mobile section. It has no template inheritance configured so that it doesn't inherit from the site's main master page. Under it, you can find the following sections:

- **Home** – this is the page where the mobile user gets redirected from the site's main *Home* page. In the top part of the page, there are two Editable text web parts with some explanatory text. Below them, content is loaded dynamically from the Editable text web part on the main *Home* page by the Repeater web part with title *Home page repeater*.
- **News** – contains a Repeater web part displaying the *News* documents stored under */News*. Its transformation is written so that when you click a news item's title, you get redirected to the */Mobile/News/Detail* page with the *ID* of the news item in the wildcard part of the URL (*/Mobile/News/{id}*).
- **Articles** – the Repeater web part on this page displays the articles stored under it. The content of this section is separate, i.e. it is not shared with the rest of the site and is used only in the mobile section.
- **About us** – contains only two Editable text web parts and its content is separate and used only in the mobile section.
- **Search** - this page is not accessible through navigation. Users are redirected here when they search within the mobile section. Search results are displayed here to them by the Smart search results web part.



The figures below show what the mobile section looks like when viewed on different mobile devices:

CSS stylesheet for the mobile section

There is a dedicated CSS stylesheet - **Corporate Site – Mobile** - which is used only by the pages in the mobile section. While the site's main stylesheet is **Corporate Site**, the mobile section's master page has the **Corporate Site – Mobile** stylesheet assigned in **Properties -> General -> CSS**

stylesheet. All pages under it inherit this configuration and use the same stylesheet as the master page.

Mobile device redirection web part

Recognition of mobile devices is ensured by the Mobile device redirection web part placed on the main **Home** page of the website. Site visitors accessing the site's main **Home** page from a mobile device get recognized by the web part and get redirected to one of the two URLs specified by its **Small device redirection URL** and **Large device redirection URL** properties. In this case, both large and small devices get redirected to *~/Mobile/Home.aspx*.

Choice of one of the two URLs is performed based on detection of the device's user agent and can be configured by means of web part properties:

Small device redirection URL	URL to which mobile devices recognized as small should be redirected. The properties below can be used to configure small/large device recognition.
Large device redirection URL	URL to which mobile devices recognized as large should be redirected. The properties below can be used to configure small/large device recognition.
Redirect Android	Determines if Android mobile devices should be recognized as small or large devices. If set to *Automatic*, they will be recognized as small. If set to *Never*, they will never be redirected.
Redirect BlackBerry	Determines if BlackBerry mobile devices should be recognized as small or large devices. If set to *Automatic*, they will be recognized as small. If set to *Never*, they will never be redirected.
Redirect iPad	Determines if iPad devices should be recognized as small or large devices. If set to *Automatic*, they will be recognized as large. If set to *Never*, they will never be redirected.
Redirect iPhone	Determines if iPhone devices should be recognized as small or large devices. If set to *Automatic*, they will be recognized as small. If set to *Never*, they will never be redirected.
Redirect Nokia	Determines if Nokia mobile devices should be recognized as small or large devices. If set to *Automatic*, they will be recognized as small. If set to *Never*, they will never be redirected.
Always redirect	On first access from a mobile device, a redirection cookie is stored in the device's browser. If this option is disabled, the device is not redirected if the cookie is present in its browser, i.e. it is only redirected for the first time. If enabled, devices are redirected on each access.
Other small devices (User agent)	List of additional user agents which should be recognized as small mobile devices. Multiple user agents can be entered, each on a separate line.
Other large devices (User agent)	List of additional user agents which should be recognized as large mobile devices. Multiple user agents can be entered, each on a separate line.

7.2.4.20 Page template internals and API

7.2.4.20.1 Overview

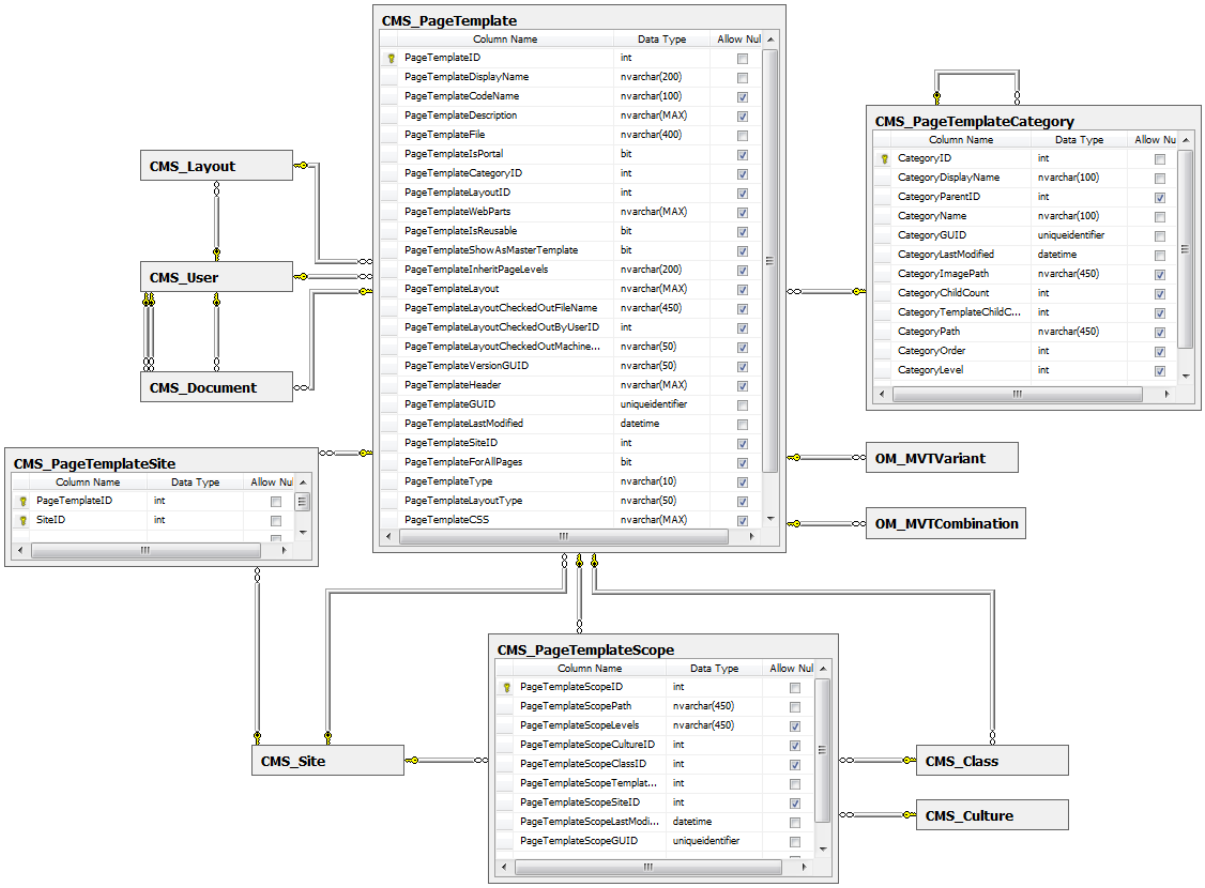
In this chapter, you will learn which database tables and API classes are used by page templates. You will also see the most common API examples.

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.

7.2.4.20.2 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.



7.2.4.20.3 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.

7.2.4.20.4 API examples

7.2.4.20.4.1 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;
```

7.2.4.20.4.2 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);
}
```

7.2.4.20.4.3 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);
}
```

7.2.4.20.4.4 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;
}
```

7.2.4.20.4.5 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;
}
```

7.2.4.21 Page layout internals and API

7.2.4.21.1 Overview

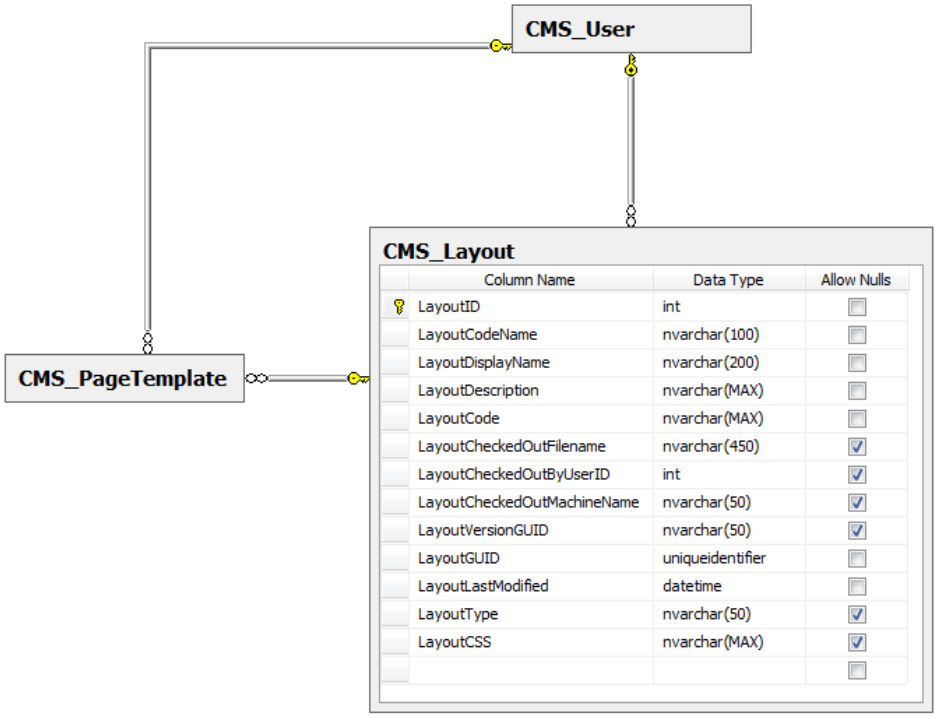
In this chapter, you will learn which database tables and API classes are used by page layouts. You will also see the most common API examples.

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.

7.2.4.21.2 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.

7.2.4.21.3 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.SiteProvider** namespace.

CMS_Layout table API:

- **LayoutInfo** - represents one page layout object.
- **LayoutInfoProvider** - provides management functionality for page layouts.

7.2.4.21.4 API examples

7.2.4.21.4.1 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.SiteProvider;
```

7.2.4.21.4.2 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 ID=\"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);
}
```

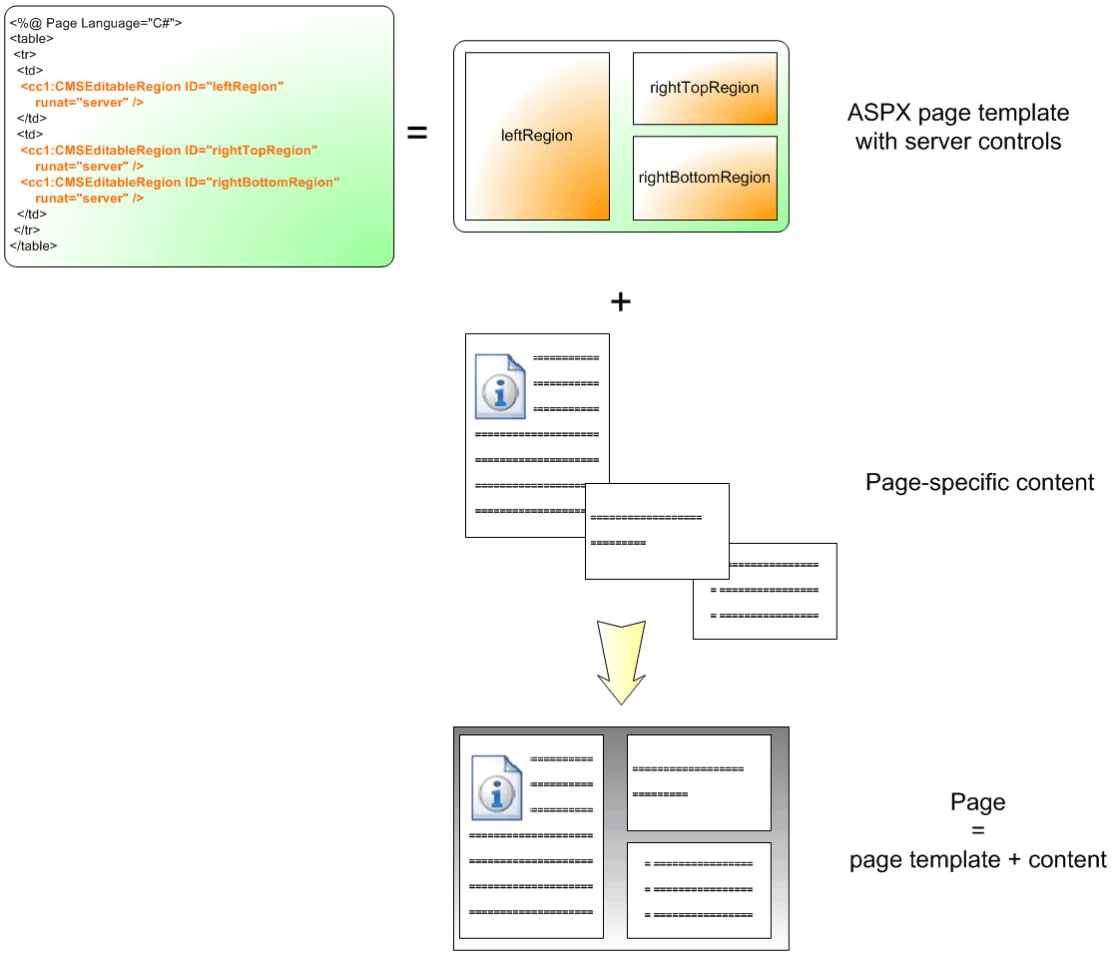
7.2.5 ASPX page template development model

7.2.5.1 Overview

If you're 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 **aliasPath** as a URL parameter that tells the page template which page should be displayed.

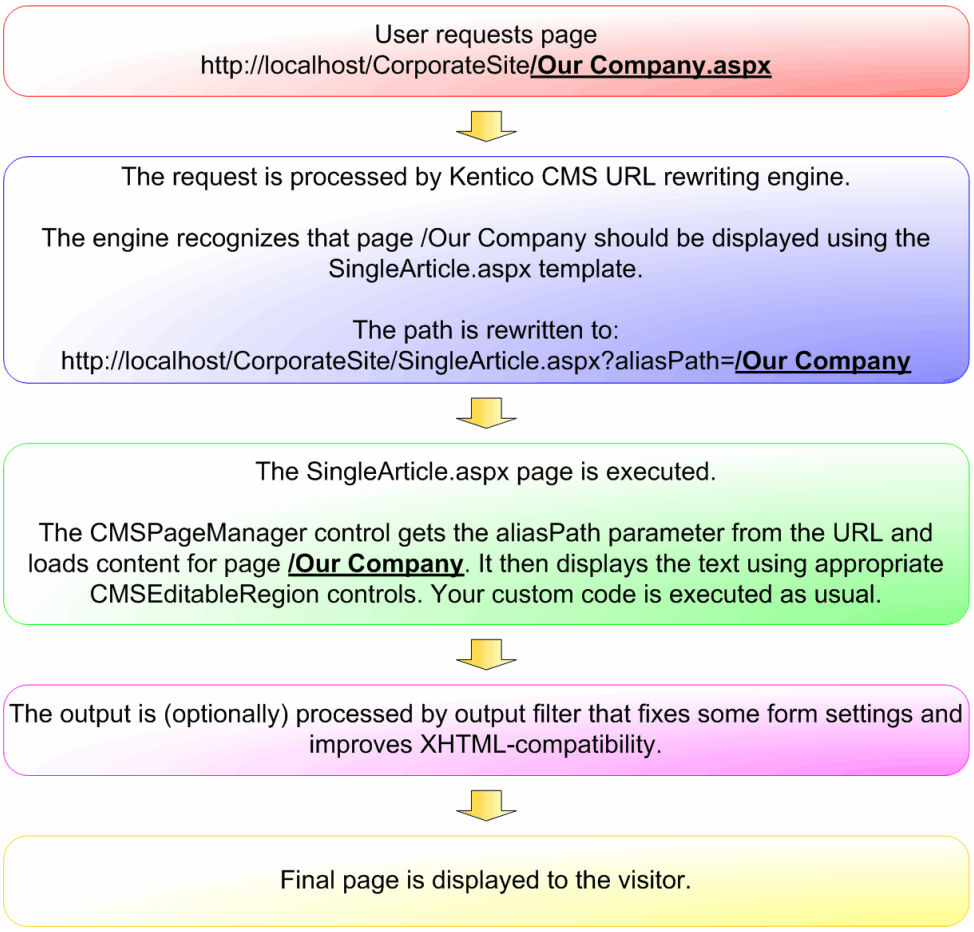
What does the ASPX page template consist of?

The 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 is the ASPX page template processed?

When a user requests some 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

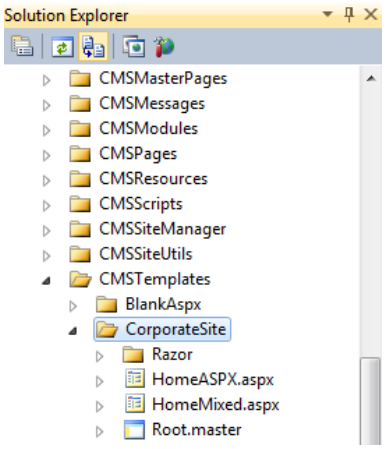
Now you will learn how to create a new ASPX page template. We will create a new page with two columns that will contain editable regions.

> **Adding Kentico CMS Controls to your Visual Studio Toolbox**
>
> Before you start development of ASPX page templates, it's recommended that you add Kentico CMS Controls to your Visual Studio Toolbox so that you can simply drag-and-drop the controls onto the ASPX pages.
>
> You can find step-by-step instructions in the Adding Kentico CMS Controls to the

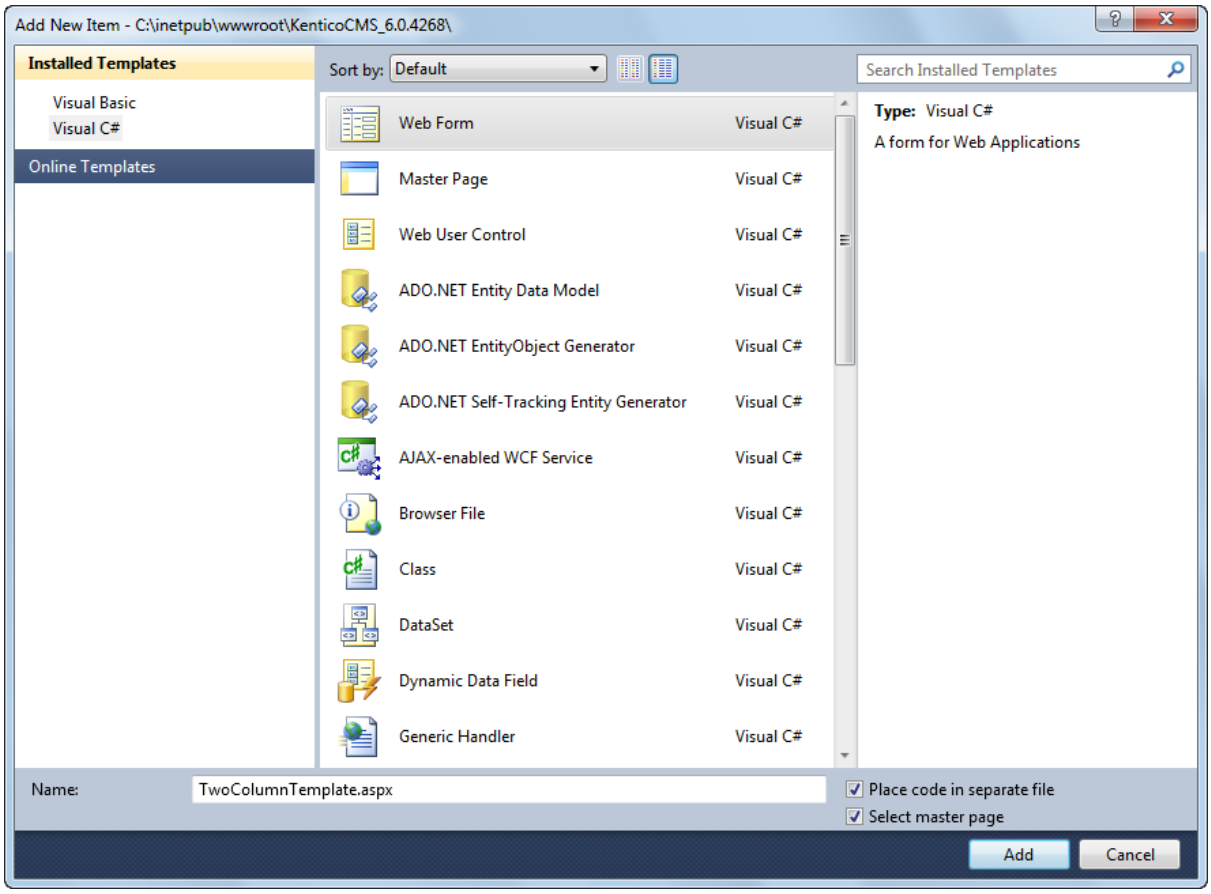
Toolbox topic.

1. Open the web project in **Visual Studio**. You can open it either using the **WebProject.sln** file or using **File -> Open -> Web Site...** in the menu.

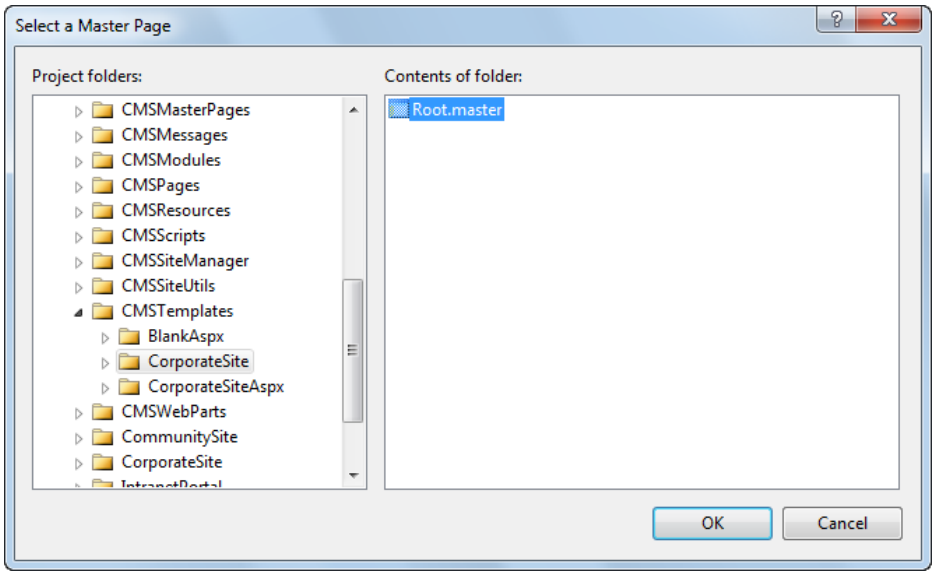
2. Now right-click the **CMSTemplates -> CorporateSite** folder in the Solution Explorer and select **Add New Item**:



3. Choose to create a new **Web form** and call it **TwoColumnTemplate.aspx** and check the **Select master page** box. Click **Add**.



4. The **Select a Master Page** dialog appears. Choose the folder **CMSTemplates/CorporateSite**, select the **Root.master** file and click **OK**.



Writing the ASPX code

5. Switch to 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 specifies that this content will be loaded into the master page (as defined in the Root.master file). As you can see, you can use the standard ASP.NET concept of master pages.

The **CMSEditableRegion** control defines an editable region that will be displayed as an HTML editor on the **Content -> Edit -> Page** tab of **CMS Desk**. On the live site, it ensures that the entered content is displayed on the page.

Please note: this example uses a table layout. If you prefer a CSS layout, you can simply replace the surrounding HTML code with <DIV> elements. As you can see, you have full control over the HTML code.

6. Switch to the code behind. You need to add a reference to the **CMS.UIControls** namespace:

[C#]

```
using CMS.UIControls;
```

7. The last step is to modify the class from which our page is inherited. Change the following code:

[C#]

```
public partial class CMSTemplates_CorporateSite_TwoColumnTemplate : System.Web.UI.
Page
```

to this:

[C#]

```
public partial class CMSTemplates_CorporateSite_TwoColumnTemplate : TemplatePage
```

so that the page can be used as a page template in Kentico CMS.

Please 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.

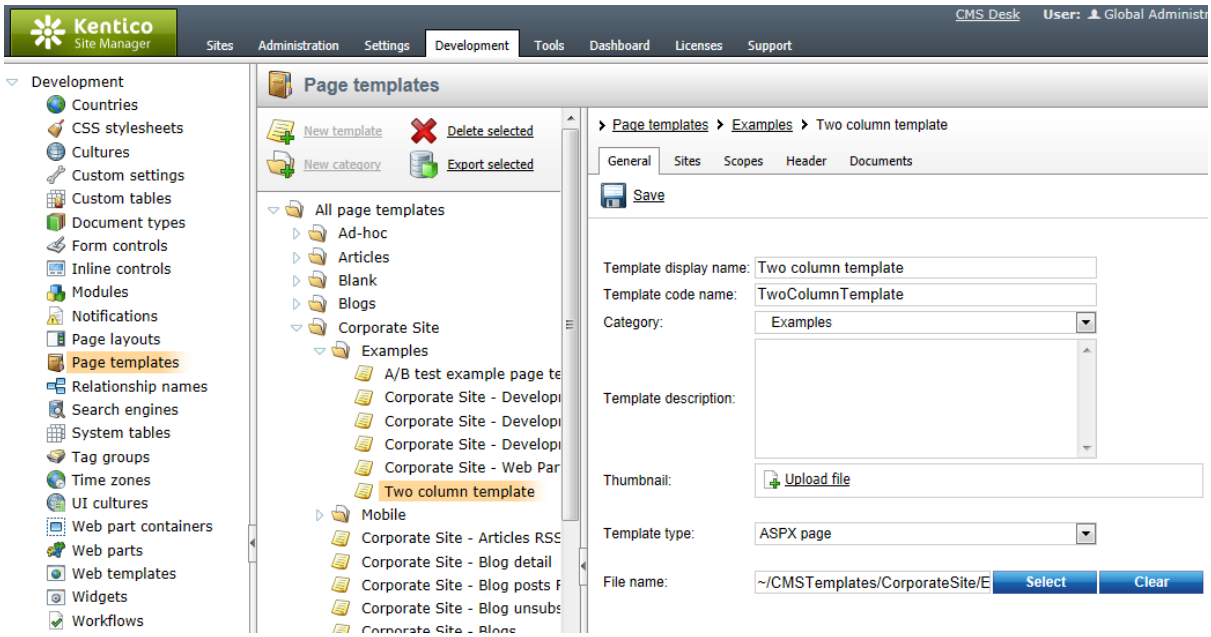
8. Sign in to **Site Manager** and go to **Development -> Page templates**. Select an appropriate folder, for example **Corporate Site/Examples** and click **New template**. Enter the following values:

- **Template display name**: Two column template
- **Template code name**: TwoColumnTemplate

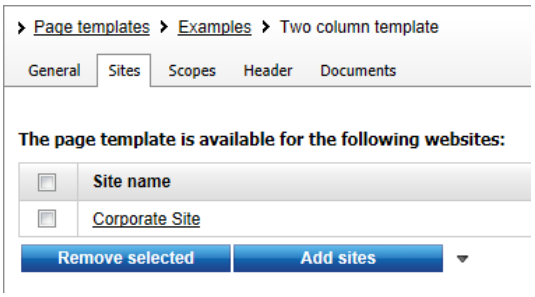
Click **OK**. The template will be created and its **General** tab will be displayed. Since the template was defined in Visual Studio as a standard ASPX page, the **Template type** must be set to *ASPX* (this is the case by default). Now enter the following value into the **File name** field:

~/CMSTemplates/CorporateSite/TwoColumnTemplate.aspx

It is the virtual path of our ASPX page. Alternatively, the **Select** button can be used to manually select the file. Click **Save**.

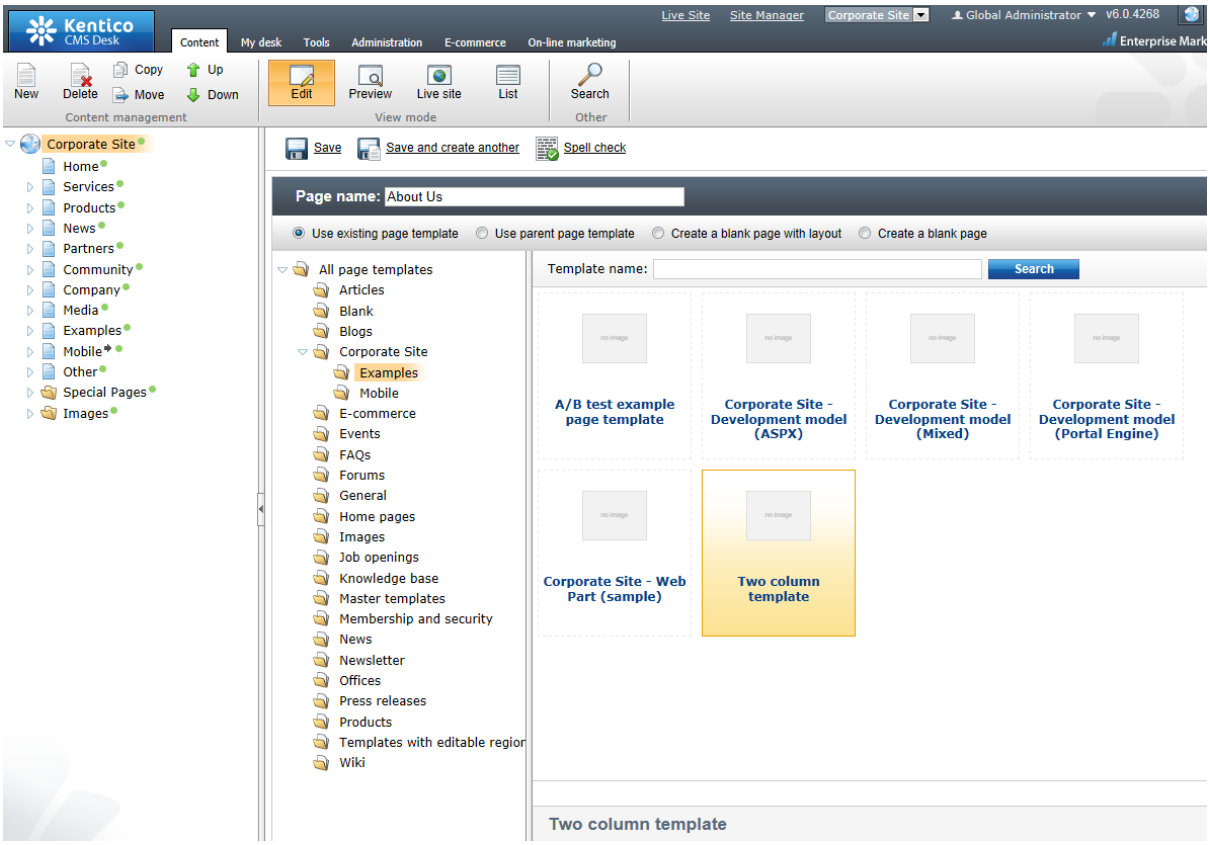


9. Now switch to the **Sites** tab, assign the page template to the current website using the **Add sites** button and click **OK**.



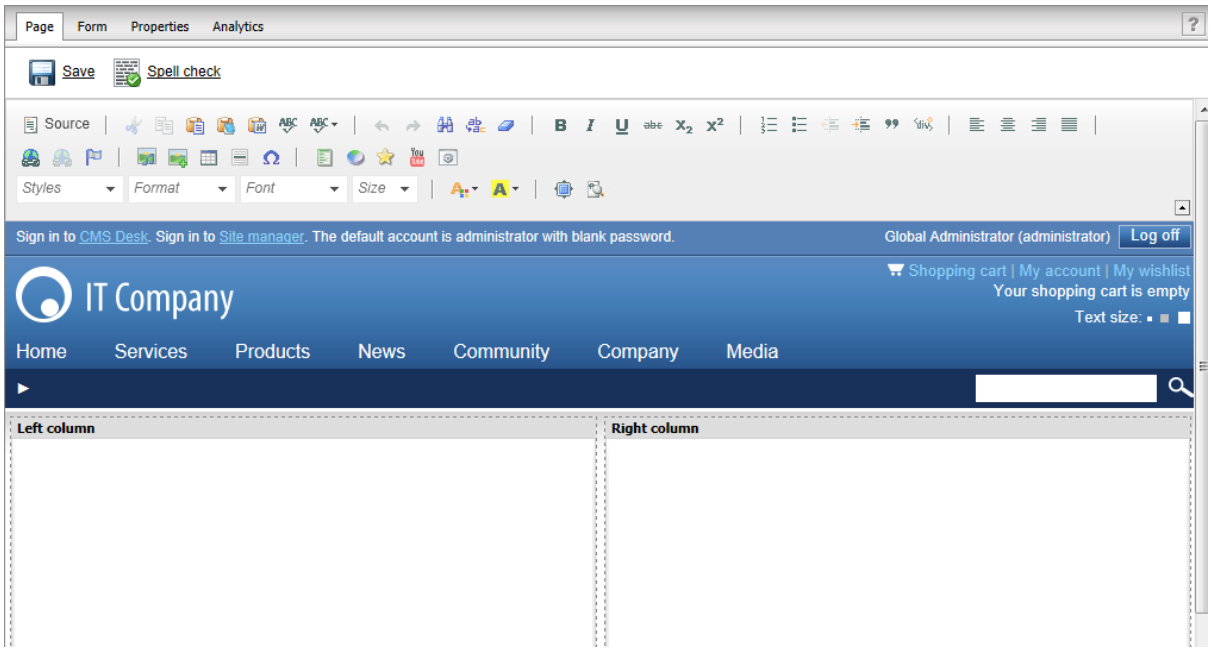
Creating an About Us page based on the new page template

10. Go to Kentico **CMS Desk -> Content**. Select **Corporate Site** (the root of the content tree) and click **New** in the main menu of the **Content** section. Choose to create a new **Page (menu item)**. Enter *About Us* into the page name field and choose to create a page using the **Corporate Site/Examples/ Two column template** page template.



Click **Save** to create the new page.

11. Switch to the **Page** tab and you will see a page with editable regions like this:



Congratulations, you have just created your first ASPX page template. Now you can enter some text and click **Save** to store the content.

7.2.5.3 Creating ASPX master pages

Kentico CMS allows you to use standard ASP.NET master pages together with ASPX page templates. This is a very powerful concept which allows you to share the same site header and footer with a logo, main menu, search box, etc. across all pages without having to create these sections on each page template again and again.

Master pages are defined in files with the **.master** extension. You can assign a single master page to each ASPX page. The master page must always contain the **ContentPlaceHolder** control as shown here:

```
<asp:ContentPlaceHolder ID="plcMain" runat="server"></asp:ContentPlaceHolder>
```

The **ContentPlaceHolder** control specifies where the content of page templates that use this master page should be loaded. So the master page typically contains the main logo and navigation elements and the content is displayed by ASPX pages loaded into the master page.

The following code sample defines a very simple master page:

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. Also, the attribute's value must be set to the name of the master page's code behind file and the *Inherits* attribute must be set according to the location and name of the master page.

```
<%@ 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">
        <cms:CMSPortalManager ID="CMSPortalManager1" runat="server"
EnableViewState="false" />
        <ajaxToolkit:ToolkitScriptManager ID="manScript" runat="server"
EnableViewState="false"
        ScriptMode="Release" />
        </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>
```

The **CMSPortalManager** control ensures the loading and saving of content between the database and editable regions. It also provides the management necessary for web part or widget zones if any are defined on child ASPX pages (as described in Adding portal engine functionality to ASPX templates).

CMSPageManager control (obsolete)

Older versions of Kentico CMS used the **CMSPageManager** control instead of the CMSPortalManager. It is still available for the purposes of backward compatibility, but it does not support web/part widget zones and we recommend replacing it.

The **CMSMenu** control generates a drop-down menu used for navigation. The **ContentPlaceHolder** control defines the area where the content of sub-pages should be loaded.

If you are planning to use AJAX components on the pages of your site, you need to add the **ToolkitScriptManager** control in addition to the CMSPortalManager control.

```
<ajaxToolkit:ToolkitScriptManager ID="manScript" runat="server" EnableViewState
="false" />
```

In the code behind file, you need to add a reference to the CMS.UIControls namespace:

[C#]

```
using CMS.UIControls;
```

[VB.NET]

```
Imports CMS.UIControls
```

The master page must be inherited from the **TemplateMasterPage** class, so the class definition must look like this (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
```

And you also need do add the following code to the master page code behind class:

[C#]

```
protected override void OnPreRender(EventArgs e)
{
    base.OnPreRender(e);

    this.ltlTags.Text = this.HeaderTags;
}
```

[VB.NET]

```
Protected Overloads Overrides Sub OnPreRender(ByVal e As EventArgs)
    MyBase.OnPreRender(e)

    Me.ltlTags.Text = Me.HeaderTags
End Sub
```

It is recommended to store master pages in the **CMSTemplates** folder together with page templates, so that they are exported with your website.

7.2.5.4 Adding portal engine functionality to ASPX templates

When developing or maintaining a website built of 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 will behave in a way similar to a portal engine page template. Multiple controls of this type may be placed onto a single page if necessary.

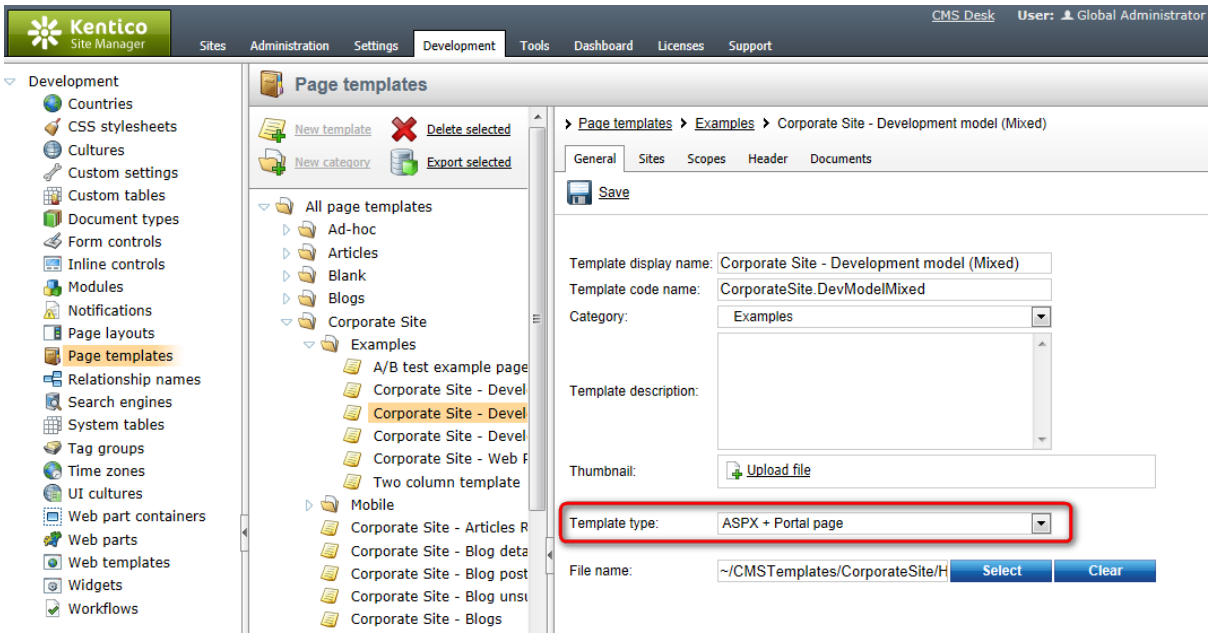
The **<LayoutTemplate>** element is used to define the graphical layout of the area. This is typically done by specifying a table structure or a CSS-based layout applied through HTML elements (e.g. , , etc.). It may contain multiple **CMSWebPartZone** controls, which represent fully functional portal engine zones. Each zone can be configured to serve as either a standard web part zone or any type of widget zone. These zones may be managed when editing a page based on the page template on the **Edit -> Design** tab of **CMS Desk**. When some web part/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 an ASPX template, the **CMSPortalManager** control must be present. Usually it is located on the master page set for the template.

Some master pages may still be using the older **CMSPageManager** control (now obsolete), which does not support web part/widget zones. If this is the case, the control must be replaced 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**, where their **Template type** must be set 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 can be designed 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:

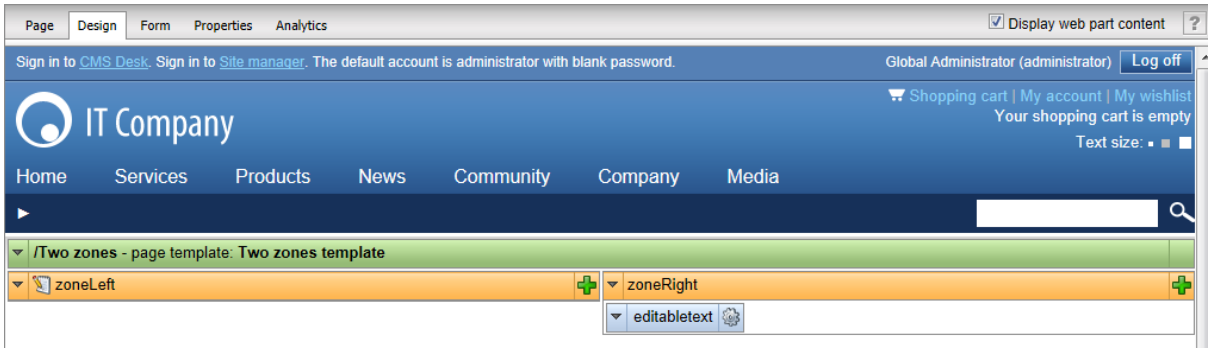
```
<cms:CMSPagePlaceholder ID="plcZones" runat="server">
    <LayoutTemplate>
        <table width="100%" cellspacing="0" cellpadding="0">
            <tr valign="top">
              <td width="50%">
                <cms:CMSWebPartZone ID="zoneLeft" runat="server" />
              </td>
              <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.

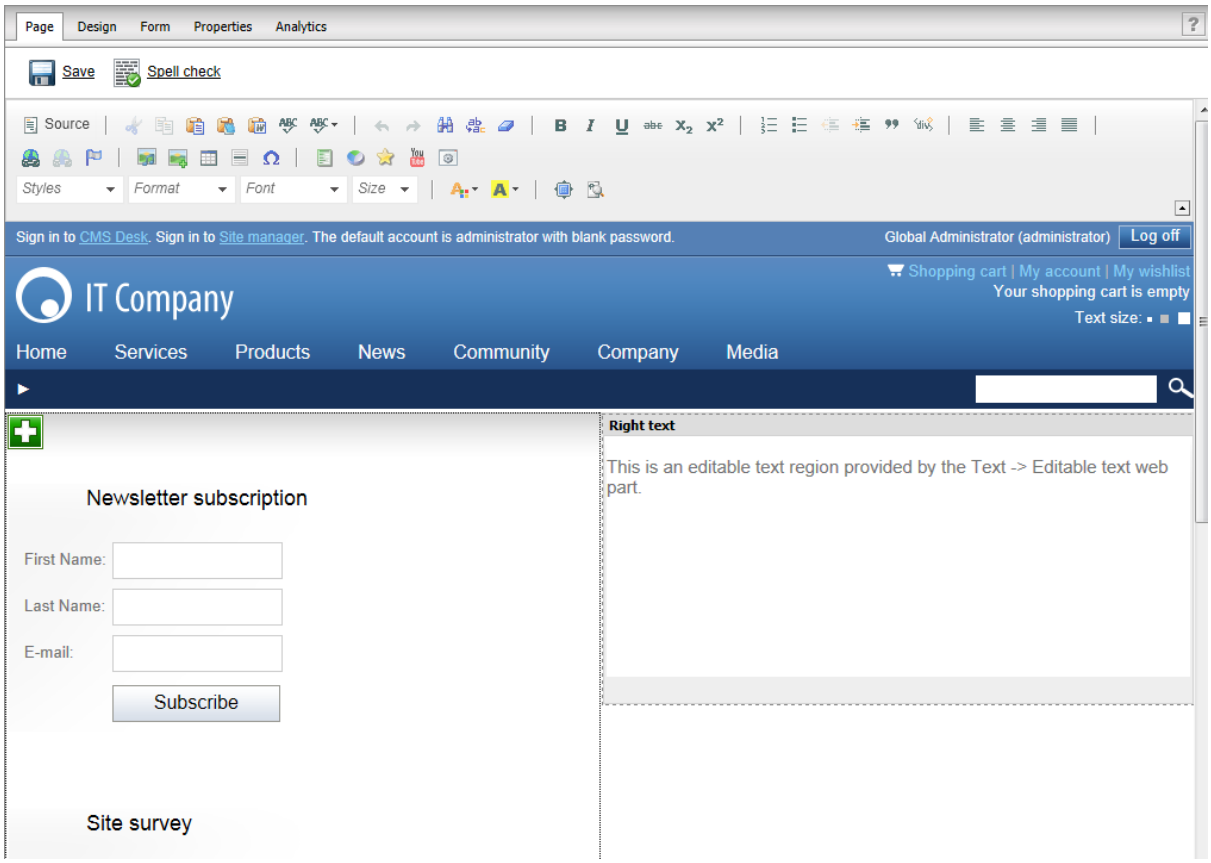
2. When registering the page template in **Site Manager -> Development -> Page templates**, be sure to select the *ASPX + Portal page* option for the **Template type** property.

3. Once the template is created and registered, go to **CMS Desk** and add a Page (Menu item) document to the content tree using the new page template. Switch to the **Design** tab of the new page and you will see two web part zones. Expand the menu (▾) of the **zoneLeft** zone, select **Properties** and change

its **Widget zone type** property from *None* to *Customization by page editor*. This zone will now serve as a widget zone for page editors. Add (+) some web part to the **ZoneRight** zone, for example **Text -> Editable text**.



4. Now open the **Page** tab and you will be able manage the editor widget zone on the left or 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.



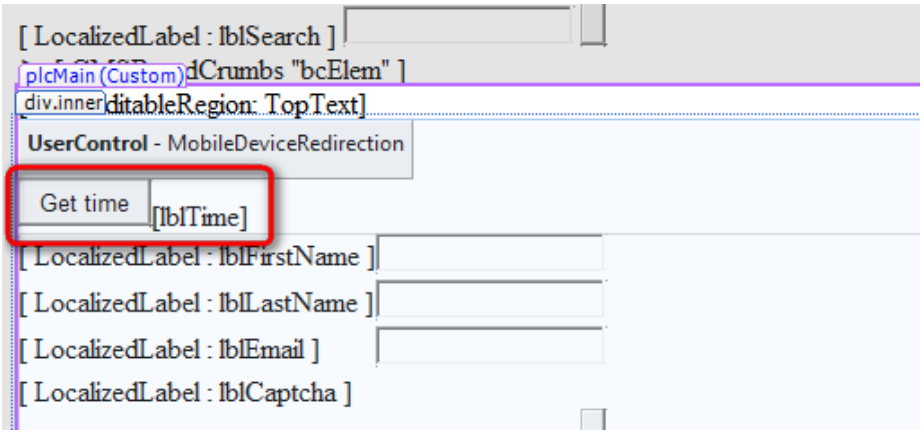
As you can see, it is possible to use web parts or widgets to build the design of pages based on ASPX page templates. This approach combines the standard architecture and development process of ASPX templates with the flexibility and user-friendliness of the portal engine.

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.

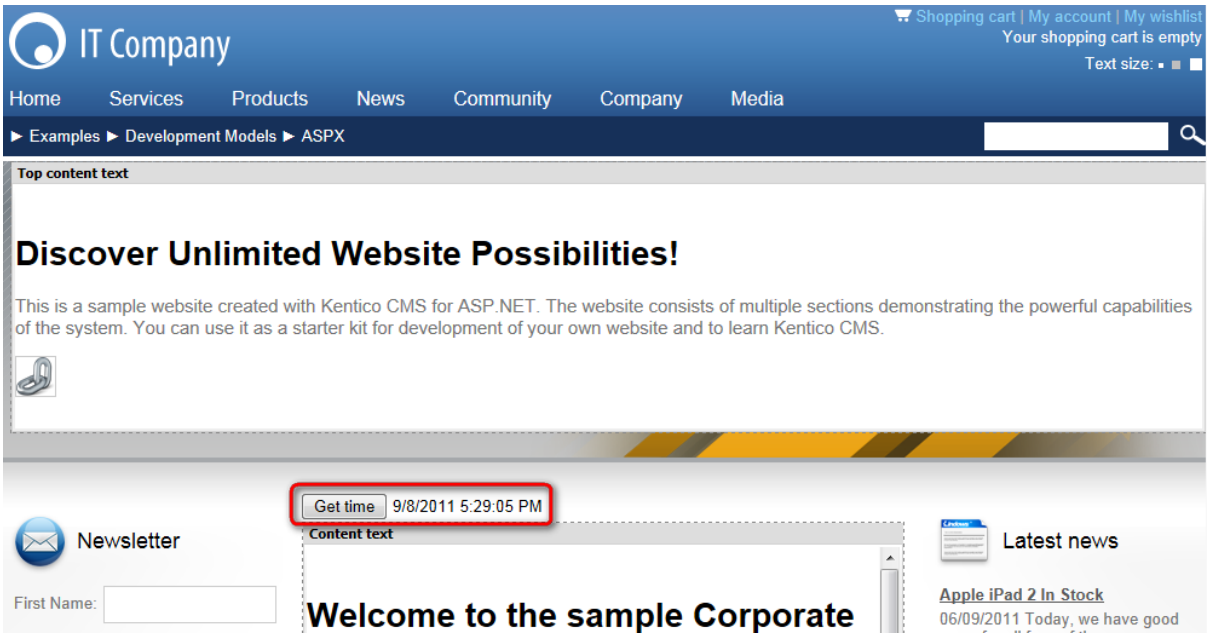


3. Double-click the button and add the following code inside the click event handler:

[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

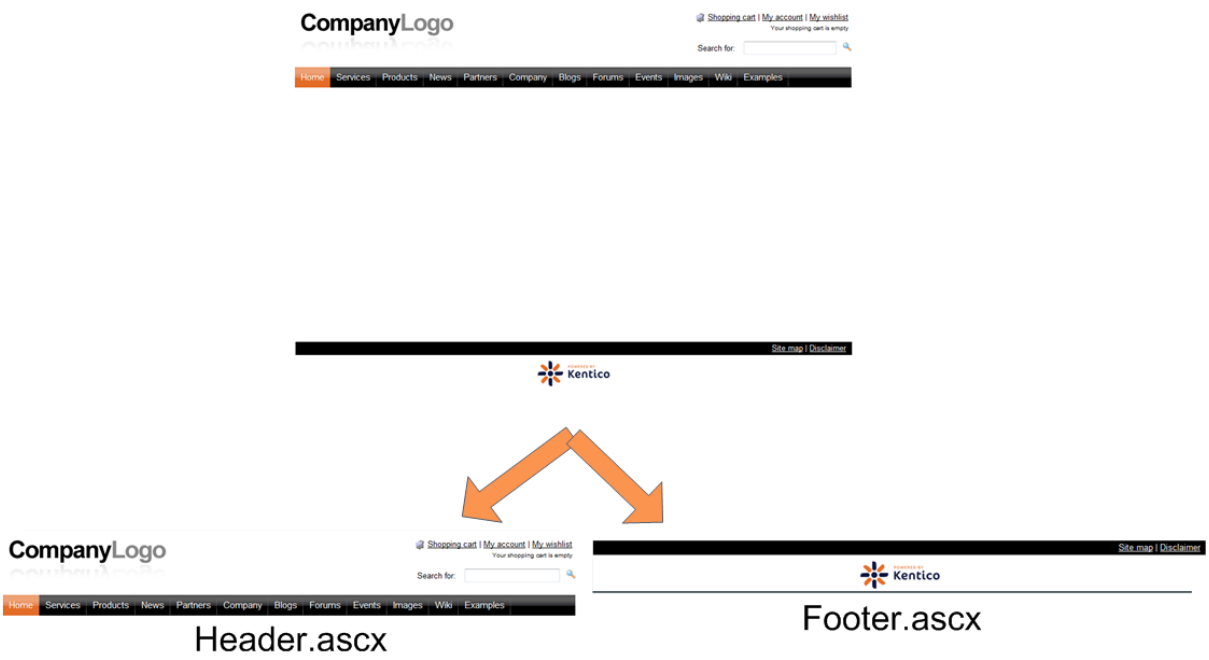
In some cases, you may need to combine ASPX page templates or external ASPX pages and portal engine page templates on a single website. The following text describes how you can accomplish this.

Imagine that you have created a website using the portal engine model. This means that the master page and the other pages use portal engine page templates. The problem here is "inserting" ASPX pages under the same master page. Unfortunately, it's not possible to do this directly. However, you can use the following workaround:

Use two master pages: the portal engine master page defined in the root document of the content tree and an ASPX master page (.master file) used for all ASPX pages within the website.

The drawback is that you will need to manage the master page in two places. But this issue also has a simple solution:

Take the web parts and HTML code you have "above" the **Page placeholder** web part in your original master page and place them into a user control called **Header.ascx**. Then, take the web parts and HTML code "below" the **Page placeholder** and place them into a user control called **Footer.ascx** as shown in the following figure:



Now delete the content of the original **portal engine master page** and place only the following web parts onto the master page template:

- **General/User control** web part using the **Header.ascx** user control
- **General/Layout/Page placeholder** web part
- **General/User control** web part using the **Footer.ascx** user control

In the **ASPX master page** (.master file), you simply put:

- the **Header.ascx** user control
- the **ContentPlaceHolder** control
- the **Footer.ascx** user control

In this way, you ensure that you can 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

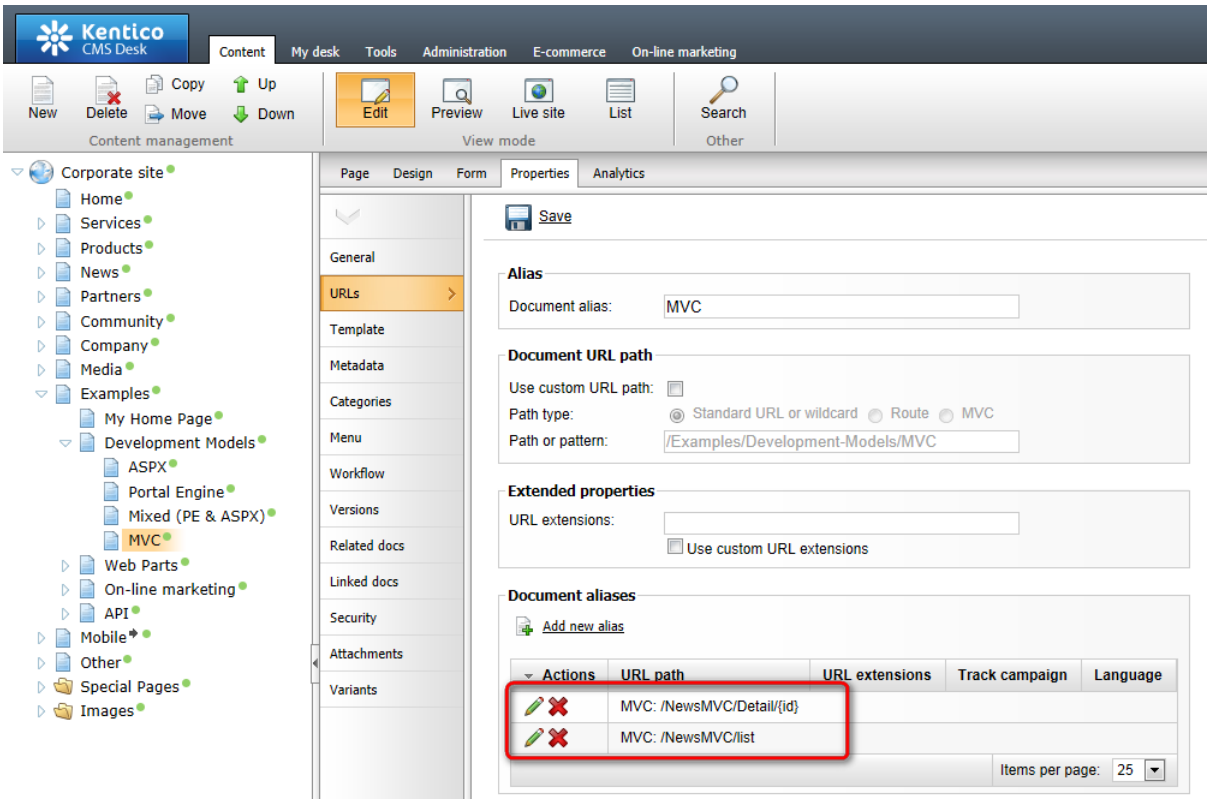
Kentico CMS supports website development using ASP.NET MVC. You can find an example of a page developed using the MVC development model on the ~/**Examples/Development-models/MVC.aspx**

page of the sample **Corporate Site**.

Document aliases configuration

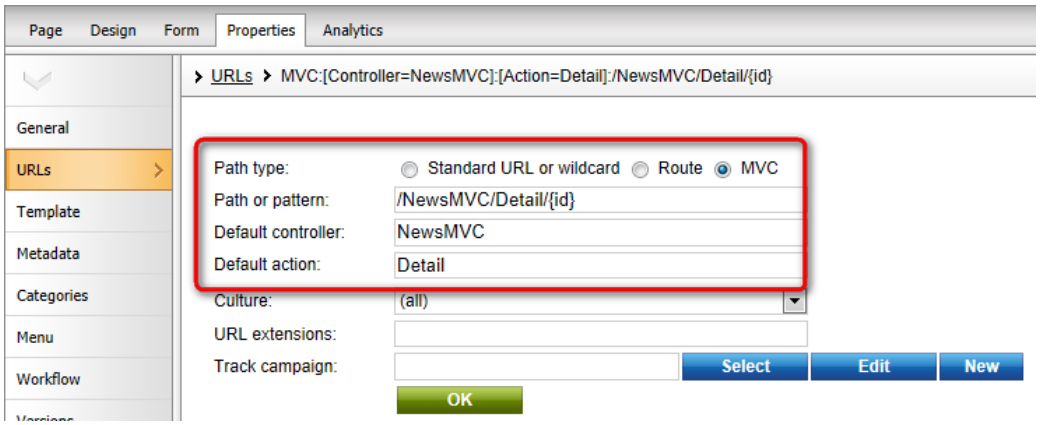
The document has two MVC Routes registered as document aliases on the **Properties -> URLs** tab:

- **/NewsMVC/List** - the final route is */NewsMVC/List* with default controller set to *NewsMVC* and default action set to *List*.
- **/NewsMVC/Detail/{id}** - the final route is */NewsMVC/Detail/{id}* with default controller set to *NewsMVC* and default action set to *Detail*.



If you edit the aliases, you can see the following configuration:

- **Path type** - when MVC is selected, requests to the alias URL are handled as requests for MVC pages.
- **Default controller** - name of the controller containing the action which should be performed when the alias URL is accessed, without the *Controller* part at the end. E.g. if the controller class is called *NewsMVCController*, enter *NewsMVC*. The specified controller is first searched in the *CMS.Controllers.<current site code name>* namespace. If it is not found there, it is searched in the *CMS.Controllers.Global* namespace.
- **Default action** - action defined within the controller which should be performed when the alias URL is accessed.



Other options are also available with MVC patterns:

- **{controller}, {action}** - enable to provide controllers and actions dynamically from the URL, e.g. */{controller}/{action}/{id}* provides the same output at */NewsMVC/Detail/My-First-News*, but with more options to direct the user to particular items (the URL parts may vary).
- **{name;value}** - provides a default value for generating URLs from URL path. E.g. */{controller;NewsMVC}/{action;Detail}/{id;My-First-News}* provides the same functionality and pattern as the previous option, but the system is able to generate the default URL of the */NewsMVC/Detail/My-First-News* document automatically.
- **{*name;value*}** - hidden value, which is registered as a value in the processing, but isn't incorporated into the URL itself. E.g. */NewsMVC/List{*TopN;10*}* generates the */NewsMVC/List* pattern and provides the value of *TopN* 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 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.

MVC code files

The **NewsMVC** controller specified in the **Default controller** field above is physically located in *~\App_Code\Controllers\Global\NewsMVCController.cs* (or *~\Old_App_Code\...* if you installed Kentico CMS as a web application). The **Detail** action specified in the **Default action** field is implemented in its code, along with the **List** action configured for the second alias. The following code is the full code of the controller class:

```
using System.Web.Mvc;
using System.Data;

using CMS.CMSHelper;
using CMS.TreeEngine;
using CMS.GlobalHelper;
using CMS.URLRewritingEngine;

namespace CMS.Controllers.Global
{
    /// <summary>
    /// Sample controller for the news.
    /// </summary>
    public class NewsMVCController : Controller
```

```
    {
        /// <summary>
        /// Process the detail action.
        /// </summary>
        public ActionResult Detail()
        {
            // Prepare the data for 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
            {
                // Document not found
                URLRewriter.PageNotFound();
                return null;
            }

            return View();
        }

        /// <summary>
        /// Process the list action.
        /// </summary>
        public ActionResult List()
        {
            // Prepare the data for 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_CorporateSite_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.
        </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>
```

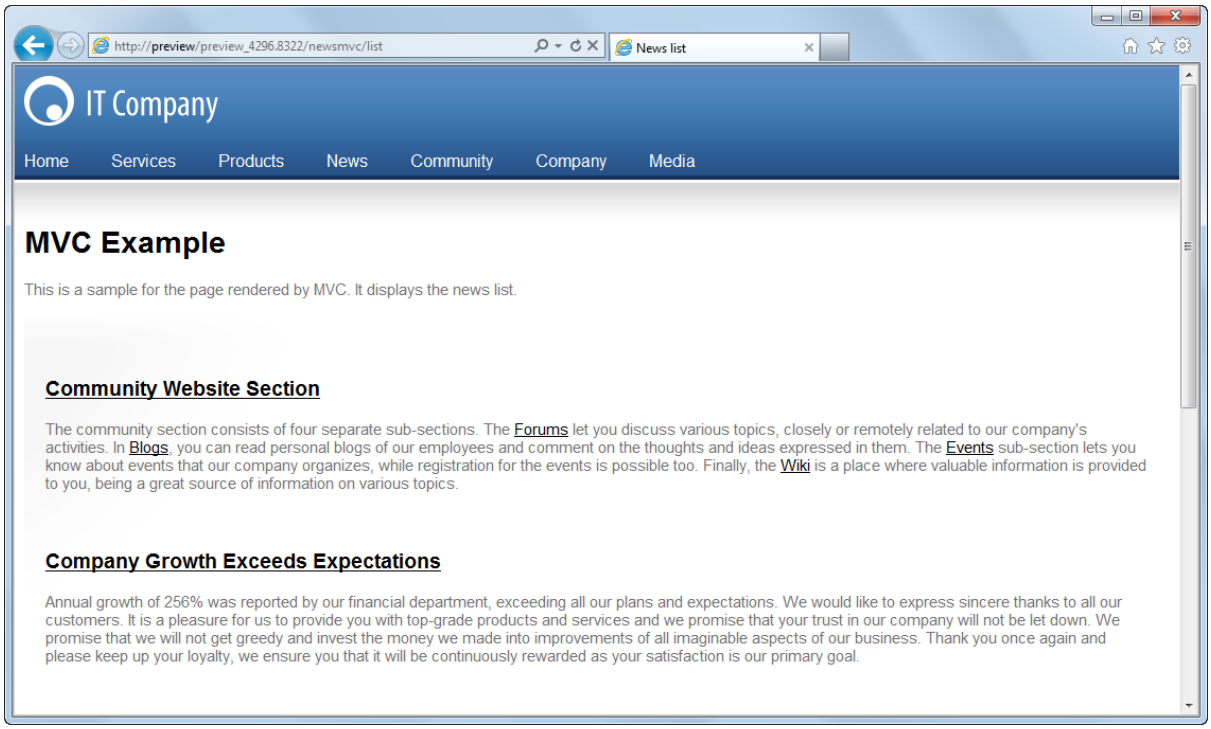
And the following code is the full code of its *Detail.aspx.cs* code behind file:

```
using System;
using System.Web.Mvc;
using System.Data;

using CMS.TreeEngine;

 /// <summary>
 /// Sample view for the news.
 /// </summary>
public partial class Views_CorporateSite_NewsMVC_Detail : ViewPage
{
    /// <summary>
    /// Returns the displayed document
    /// </summary>
    public TreeNode Document
    {
        get
        {
            return (TreeNode)ViewData["Document"];
        }
    }
}
```

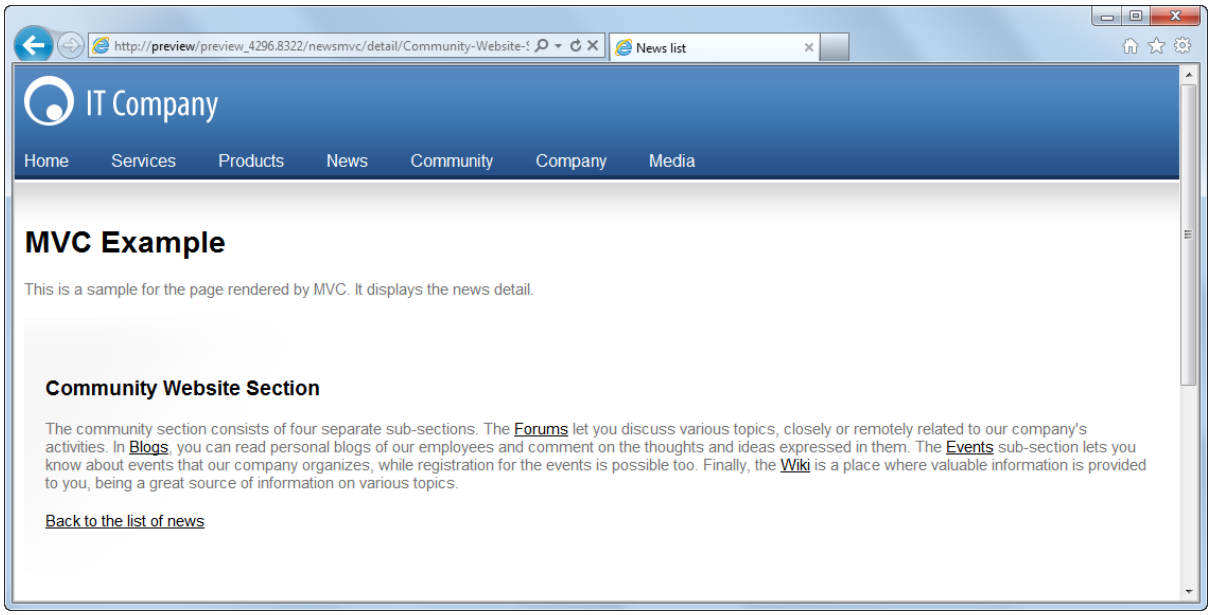
How it works on the live site

When you navigate to the page on the live site, you can see the standard portal engine page content created on the **Page** tab in **CMS Desk -> Content -> Edit**. MVC is not involved in what you see at this point. The page contains the **Click here to see the example output** link highlighted in the screenshot below.

The link is pointing to */NewsMVC/List* alias, so when you access it, 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.



Titles of the news items contain links to the */NewsMVC/Detail/{id}* URL, where the *{id}* wildcard part is substituted with a particular news item title. If you click it, the **Detail** action from the **NewsMVC** controller is executed. It displays a detail of the particular news item, as visible in the following screenshot.



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

Global caching settings

Go to **CMS Site Manager -> Settings -> System -> Performance** and either choose your website from the drop-down list or choose (global) to configure global settings inheritable by all websites. The following values are related to caching:

Server content caching	
Cache page info (minutes)	Number of minutes the page information should be cached for. This option is used for caching of page content and metadata. Since Kentico CMS often retrieves page information many times during the processing of a single page, it's actually a must to **always set this value to at least 10 minutes!** Kentico CMS automatically removes the cached page when it's modified, so it doesn't cause

	outdated content to be displayed.
Cache content (minutes)	Number of minutes content should be cached for. This option specifies that all web parts/controls should cache the content that they retrieve from Kentico CMS. You can override this value by setting the **Cache minutes** property of web parts to 0, which disables caching for the given control, or generally to some different number of minutes. It's recommended that you cache all possible content that is not modified too often. The drawback of this option is that when you modify some content, the changes appear on the live site only after the old version expires in the cache.
Server file caching	
Cache images (minutes)	This option is used only for caching of images and sets the number of minutes that the images should be cached for. It's recommended that you always use it. Kentico CMS automatically removes a cached image when it's modified, so it doesn't cause displaying of outdated content.
Maximum file size to cache	Specifies the maximum size of a file 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.
Client caching	
Client cache (minutes)	Number of minutes for which content can be cached in the client browser. If the value is set to 0, client caching is disabled.
Client cache must revalidate	If enabled, content that is cached in the client browser must be revalidated by calling the server. If disabled, the browser will not have to perform server requests if the requested content is already cached.
Output caching	
Enable output caching	If checked, the output caching is enabled. If unchecked, no output caching is allowed on the whole site. The document settings still apply, both this settings and document settings must be enabled to use output cache.
Cache output in file system (minutes)	Specifies the amount of time for which the output cache should be stored in the file system to provide persistent cache storage on application restart. If not set, standard caching mechanism in memory is used. If set, the system checks both caches.
Enable partial caching	If checked, the partial caching of web parts is enabled. If unchecked, no partial caching is allowed on the whole site. The web part settings still apply, both this settings and web part settings must be enabled to use partial cache.

Full-page caching

Full-page caching represents the most powerful option. It caches the whole page, so it's not necessary

to contact the SQL Server and run the page code again when the page is requested a second time. To use it, you first need to adjust the **Output caching** settings mentioned above. Then you can configure full-page caching in **CMS Desk -> Properties -> General**. The configuration is automatically **inherited by child pages** unless you disable caching on them.

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 full-page caching and use content caching instead.

The page stored in the cache is automatically removed when you modify the given page. However, if the page displays other documents (such as a news list) and you modify these documents, the page will not be updated. Another way of removing the page from the cache is clicking the **Clear output cache** button on the **Properties -> General** tab.

It is also possible to define custom dependencies for the output cache of a page using the Output cache dependencies web part.

Web part/control-level caching settings

Some web parts/controls used for displaying content have three properties related to caching:

- **Cache item name/CacheItemName** - this property specifies the key name under which the content will be stored in the cache. If not specified, the system generates the key name automatically based on the site name, page path, web part ID and current user name.
- **Cache minutes/CacheMinutes** - this property specifies for how long the web part should cache the content it retrieves from Kentico CMS. The default value is defined in the **Cache content (minutes)** site settings value described above. You can override the global value by setting this property to a different number.
- **Cache dependencies/CacheDependencies** - using this property, you can specify which object changes cause the web part's cache to get cleared. Below, you can find a table showing which dummy cache keys get touched when some object gets changed, including some examples. By entering the appropriate dummy keys, one per line, you can specify that when the object gets changed, the cache gets cleared.

If you check the **Use default cache dependencies** check-box, default settings will be used. The default settings are configured for each web part and include all possible object changes that the content of the web part could depend on.

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	<classname>|all	cms.user|all

(except documents)	<classname>|byid|<id> <classname>|byname|<codename> <classname>|byguid|<guid>	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: let's presume that you have a web part displaying some information about users. Therefore, whenever some user gets its 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: now let's presume that your web part is displaying information about one particular user - the administrator. Her user name is *administrator*, her ID is *53* and her 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 specify the user by the previously named properties:

- **cms.user|byid|53**
- **cms.user|byname|administrator**
- **cms.user|byguid|1ced44f3-f2fc-...**

Partial caching

Partial caching is, simply put, full-page caching for web parts. Like with full-page caching, partial caching also stores the output HTML code. But in this case, it doesn't store the whole page, but only the output HTML code of the particular web part. The web parts have the **Partial cache minutes** and **Partial cache dependencies** properties, which can be used the same way as the **Cache minutes** and **Cache dependencies** properties described above.

Please note: default partial cache dependencies are not specified for the web parts, so if you want to use this feature, you need to specify them manually by entering the dummy keys as described above.

Caching and personalization

If your website contains sections for site members, the caching will be personalized, which means each signed in user will have their own cache. This may lead to large memory consumption, so it's recommended that you set caching to lower values for membership sites.

Previewing modifications made to pages with caching

Caching may confuse the content editors since they will not see the changes they made on the live site immediately. In such case, they can preview their changes in the **Preview** mode in Kentico CMS Desk

since this mode doesn't use caching. The **Edit** mode doesn't use caching either.

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.

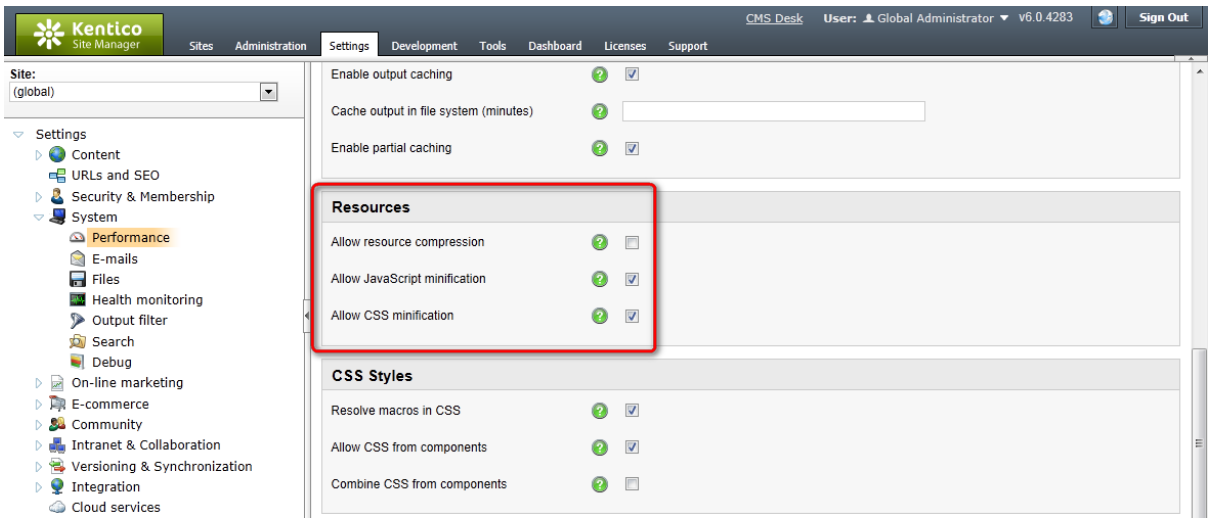
Kentico CMS has two types of built-in functionality that can be used to optimize the performance of requests for the above mentioned resources:

- **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, 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.
  URL example: *http://<domain>/CMSPages/GetResource.ashx?stylesheetname=corporatesite*

- **transformations, layouts, templates, webparts, webpartlayouts, containers** - used to request the internal stylesheets of the corresponding type of page components. The code names of the given components must be entered as the value of the parameter, separated by semicolons (if multiple component stylesheets are requested). Please see CSS stylesheets and design -> CSS for page components for additional information.
  URL examples:
  http://<domain>/CMSPages/GetResource.ashx?containers=blackbox;orangebox
  http://<domain>/CMSPages/GetResource.ashx?transformations=CMS.News.Preview

- **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.
  URL example: *http://<domain>/CMSPages/GetResource.ashx?stylesheetfile=/app_themes/default/cmsdesk.css*

- **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.
  URL example: *http://<domain>/CMSPages/GetResource.ashx?scriptfile=/cmsscripts/cmsedit.js*

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

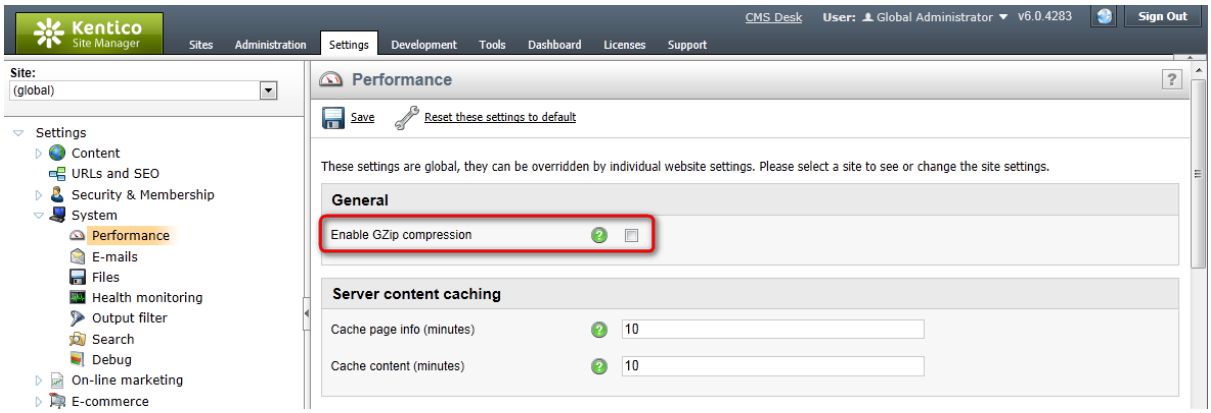
minification or compression of resources.

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 CSS stylesheets and design

7.4.1 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.

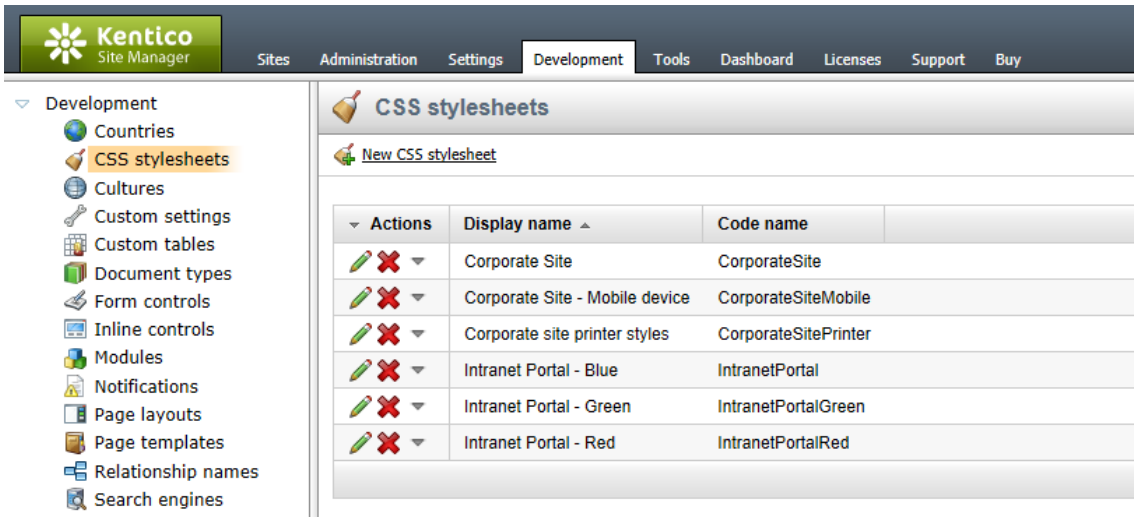
Individual pages can either use the default website stylesheet or override it with their own stylesheet. The stylesheet used by a specific page can be configured in **CMS Desk -> Content -> Properties -> General**, through the **CSS stylesheet** selector. The stylesheet assigned to a page is automatically 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.

Individual pages can be checked to detect if styles that they use are valid against the CSS 2.1 standard. For more information, see Content management -> Validators -> CSS validator.

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:

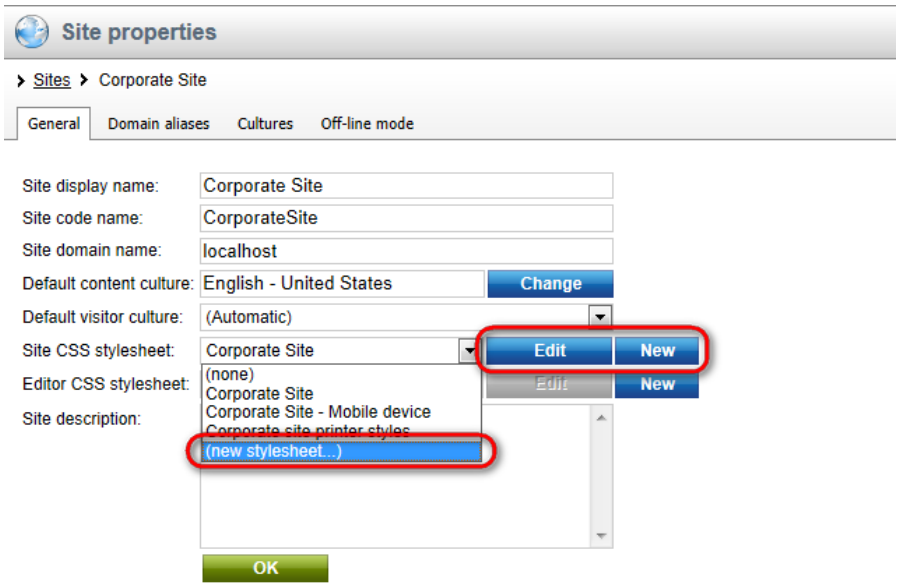
- **Stylesheet display name** - name of the stylesheet displayed in the administration interface.
- **Stylesheet code name** - a unique name used to identify the stylesheet (e.g. in code).
- **Stylesheet text** - here you can enter the definitions of the classes contained in the stylesheet (using standard CSS code). This field uses the code editor to make working with the code more user-friendly.

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

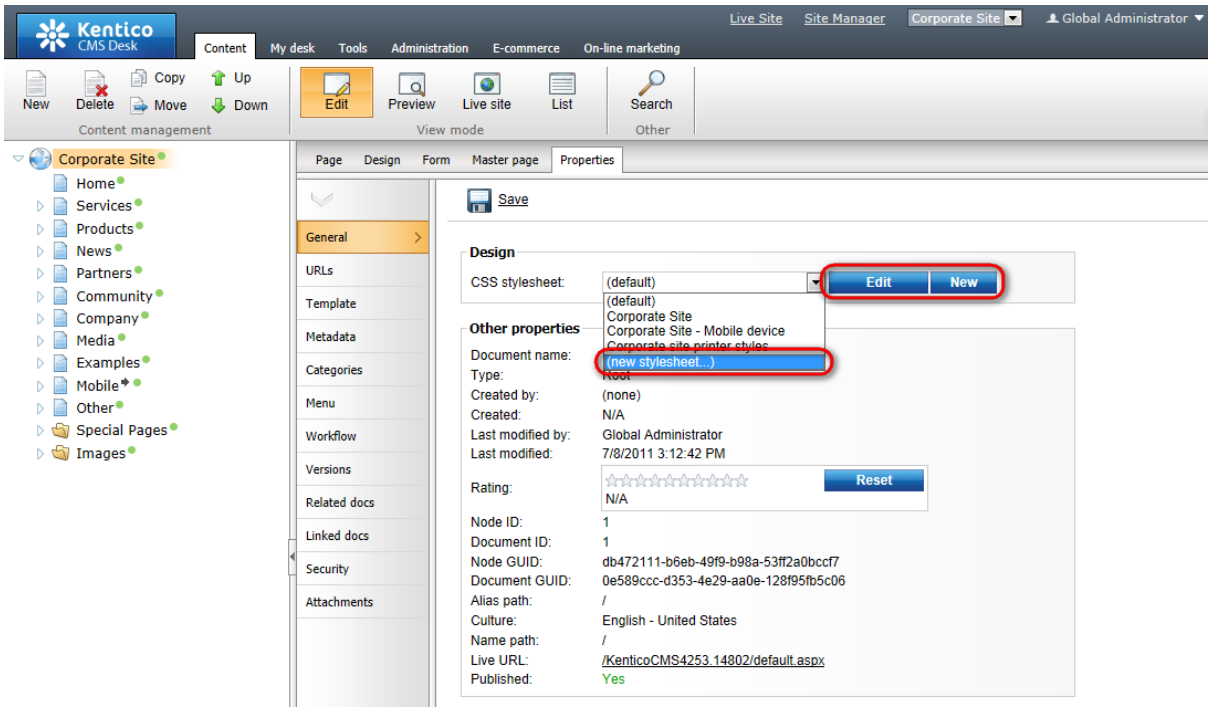
themes topic.

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# */*.

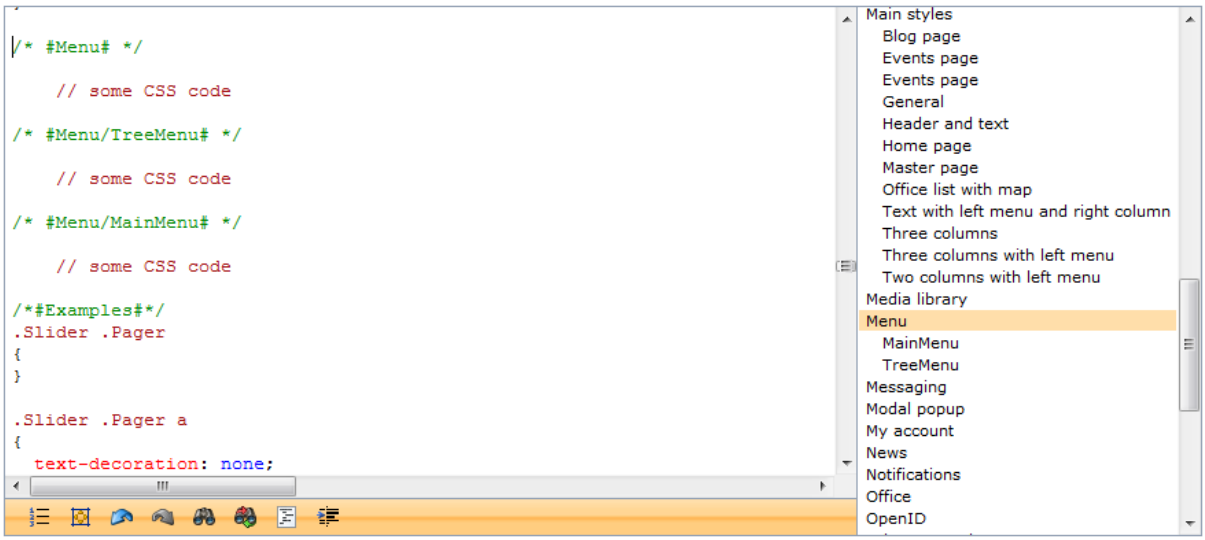
Example:

[CSS]

```
/* #Menu# */

// some CSS code

/* #Menu/TreeMenu# */
```

```
// some CSS code

/* #Menu/MainMenu# */

// some CSS code
```

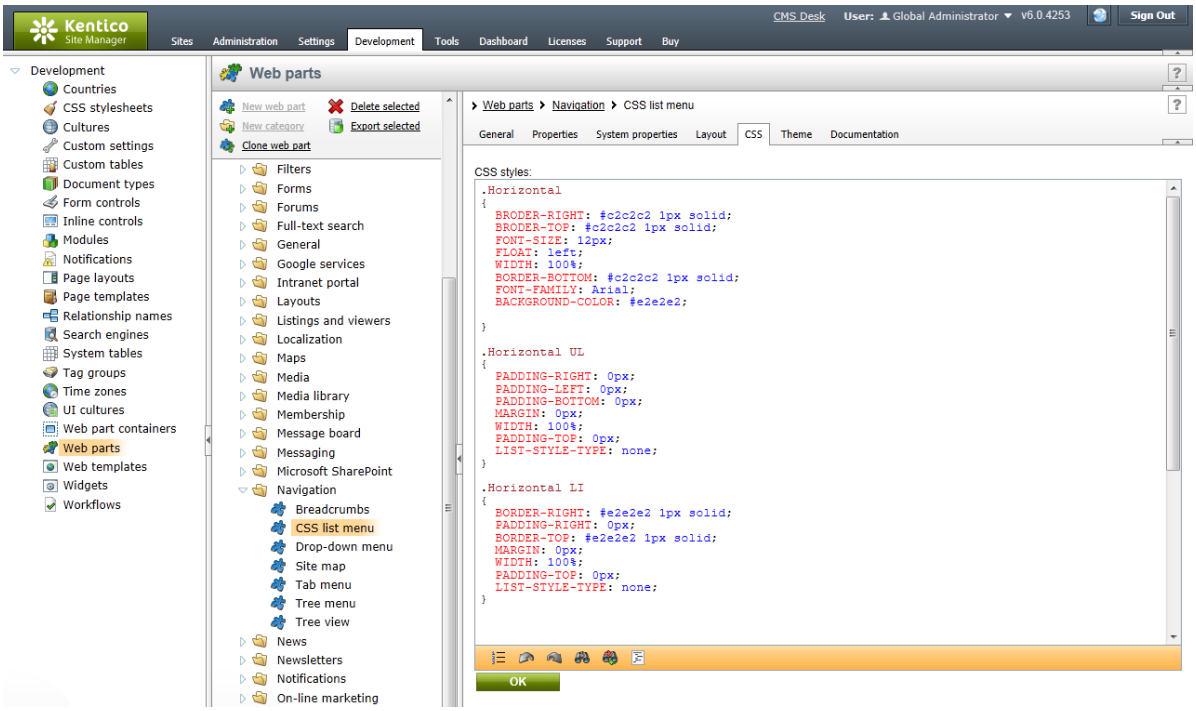
The outlined structure will look like this:



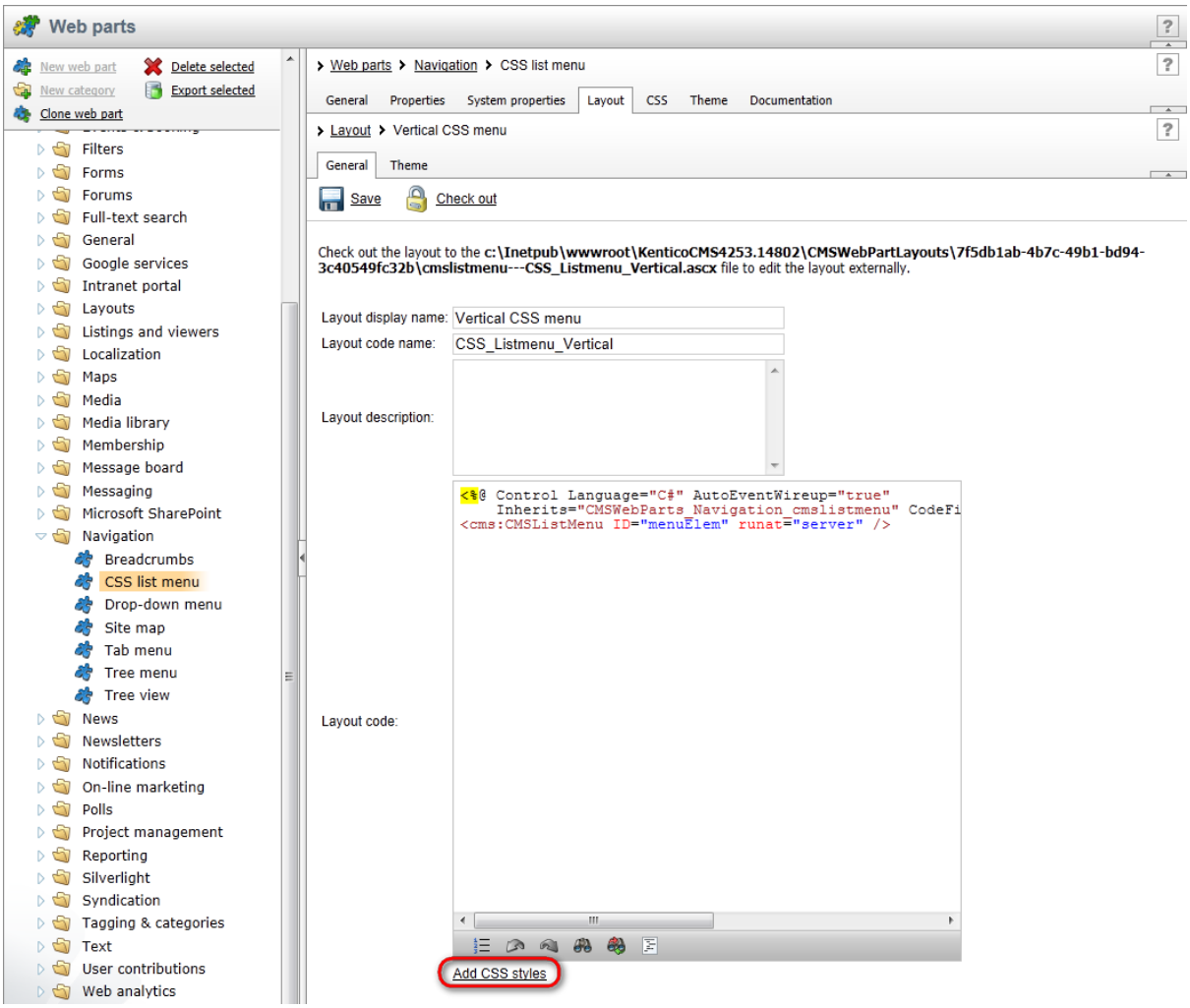
7.4.2 CSS for page components

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 the Add CSS styles link 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.

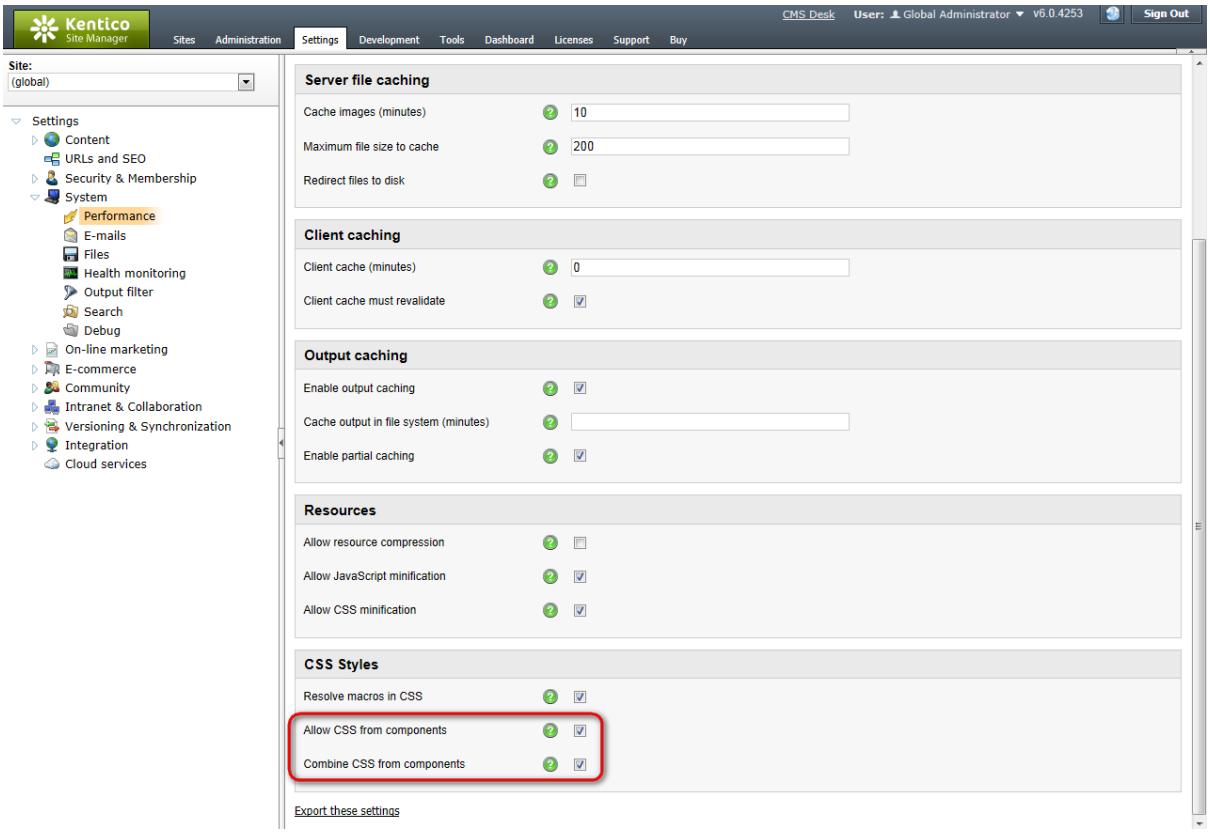
- **Custom page layouts for specific page templates** - in many cases, page templates use a custom page layout that is unique for the given template. To add CSS styles to this type of template, edit it in *Site Manager -> Development -> Page templates,* switch to its *Layout* tab and click the Add

CSS styles link 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.

There are two global settings that affect the behaviour of component CSS on all sites in the system. These can be configured 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.

7.4.3 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.

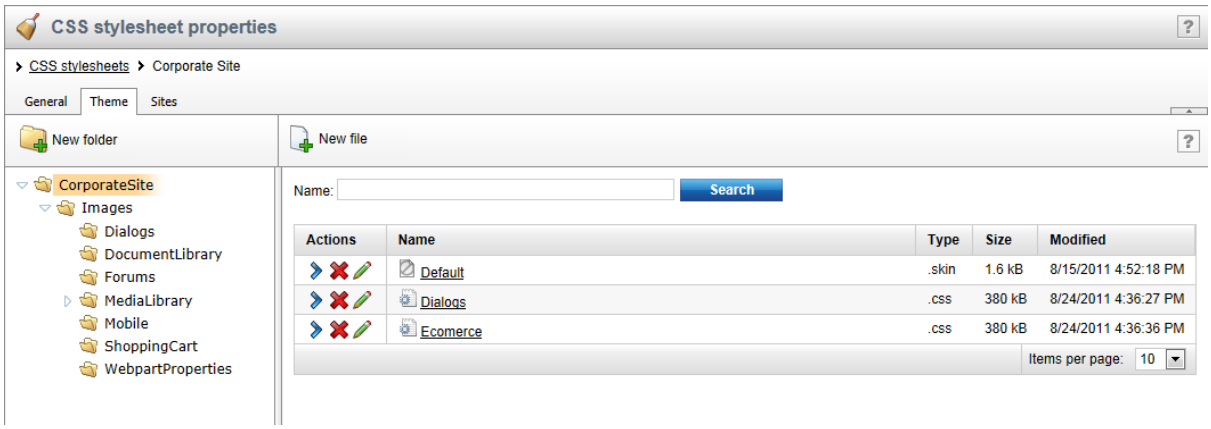
Skins for your controls need to be added to the **Default.skin** file under this folder. Here's an example of a skin for the **CMSCalendar** control / **Calendar** web part:

```
<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.

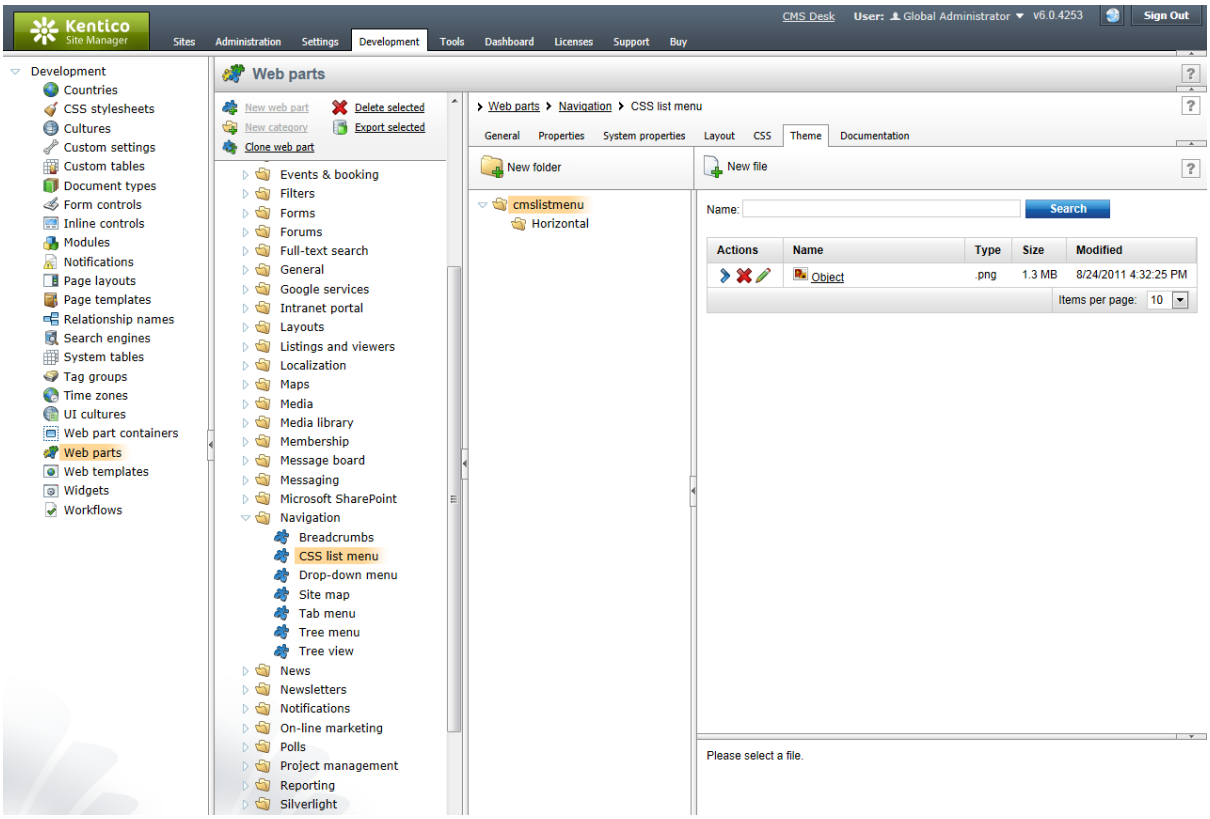
Themes folders for component styles

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**.



7.4.4 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 the all the elements that should not be visible in the print version. You can do this by adding **display:none;** to the given style.

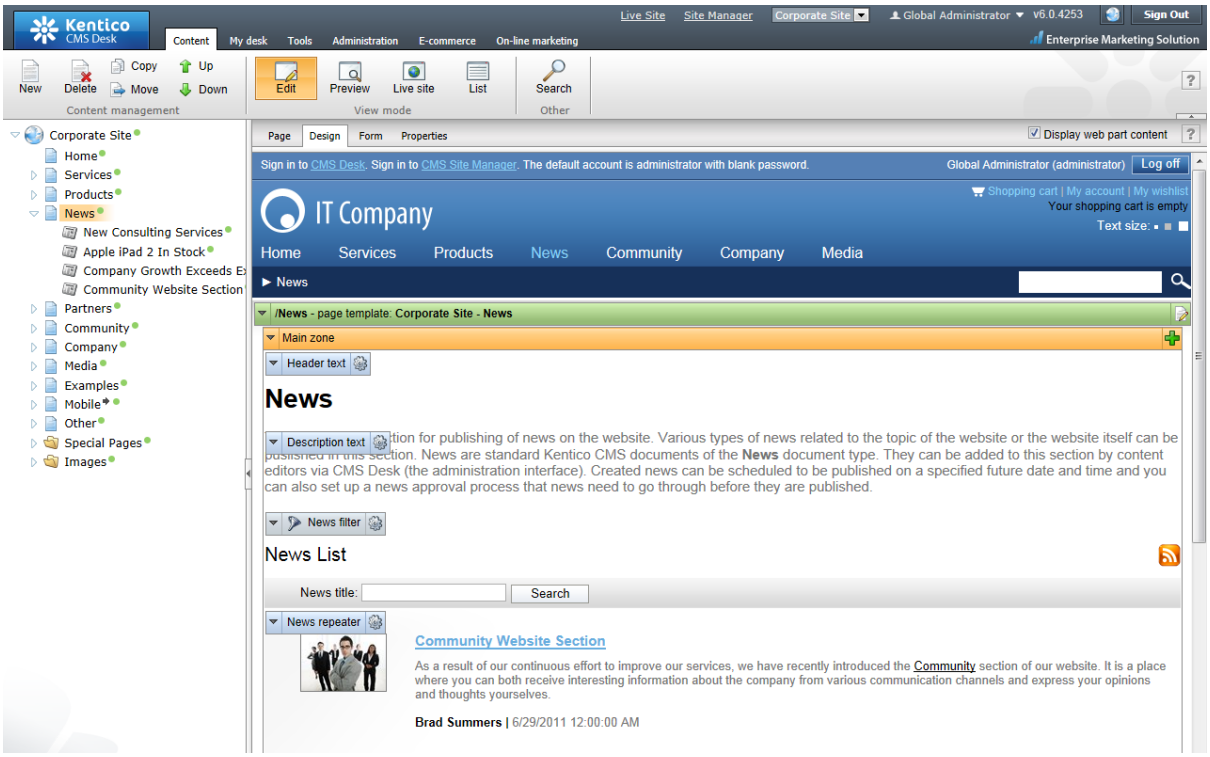
2. Add a link to this CSS stylesheet into the header tag of your master page. This can be done by selecting the root document of your website in **CMS Desk -> Content -> Edit** and switching to the **Master page** tab. For example:

```
<link href="CMSPages/GetResource.ashx?stylesheetname=Printer_styles" type="text/
css" rel="stylesheet" media="print" />
```

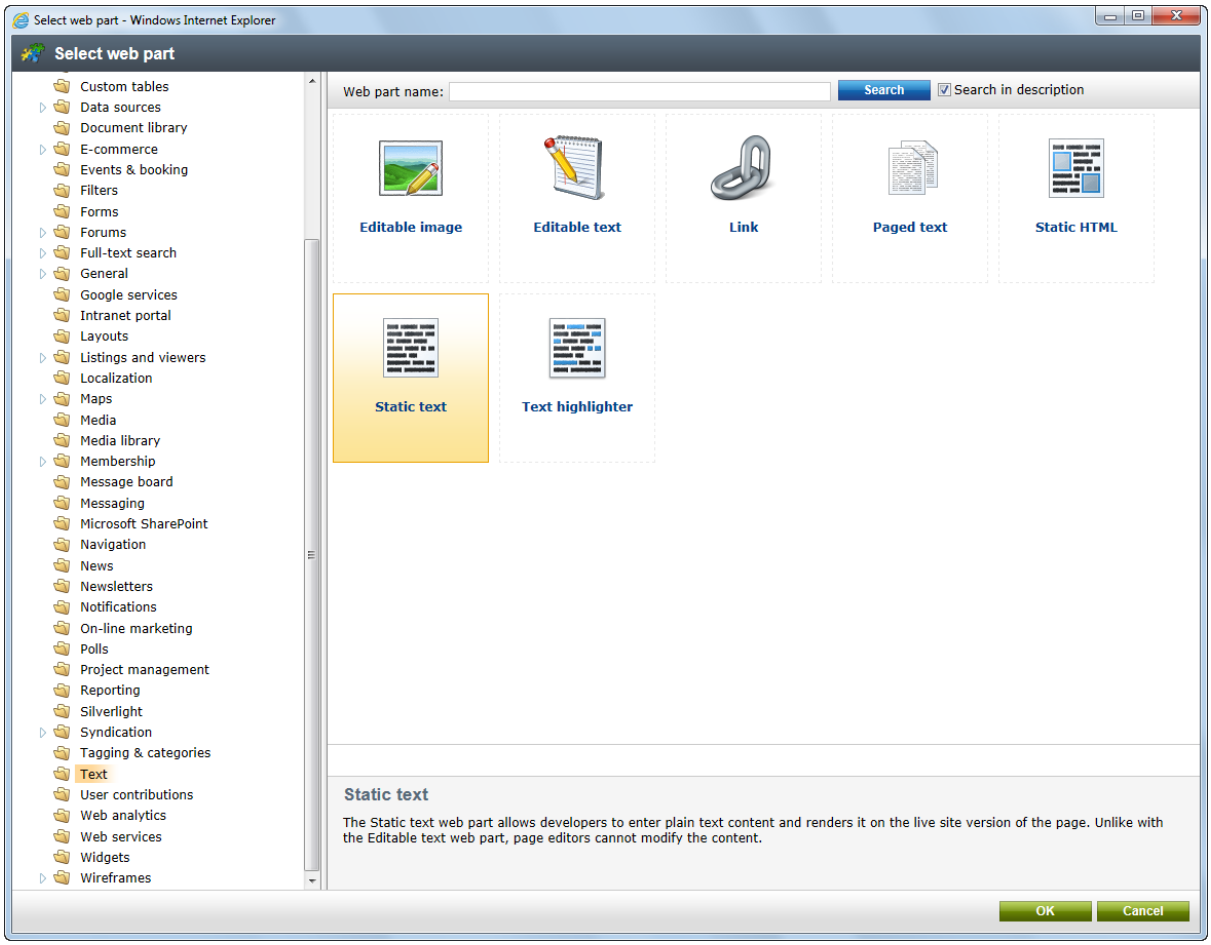
7.4.5 Print page

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** -> **Design** and click the **Add web part** () button at the **zoneLeft**.

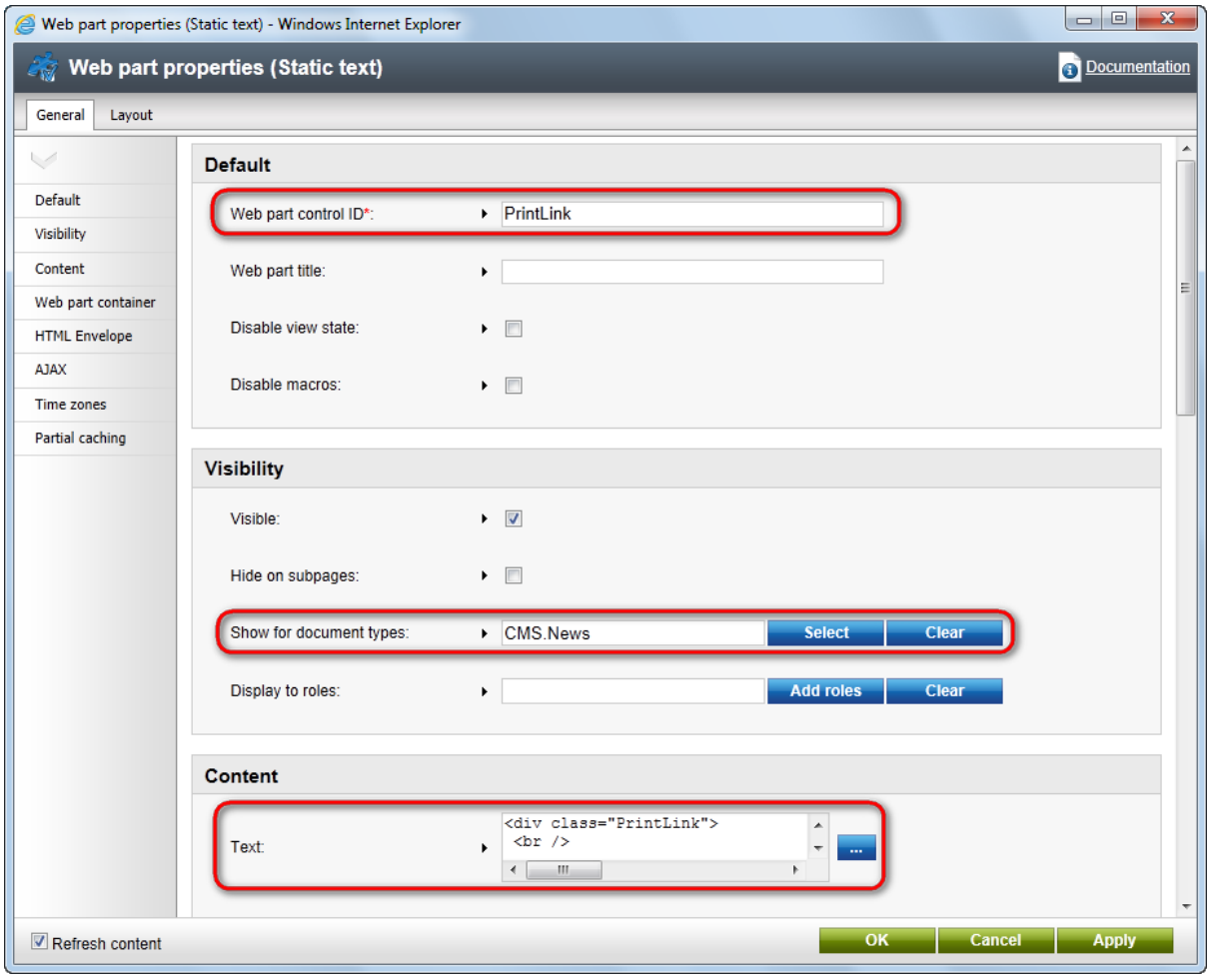


2. Select **Text/Static Text**.

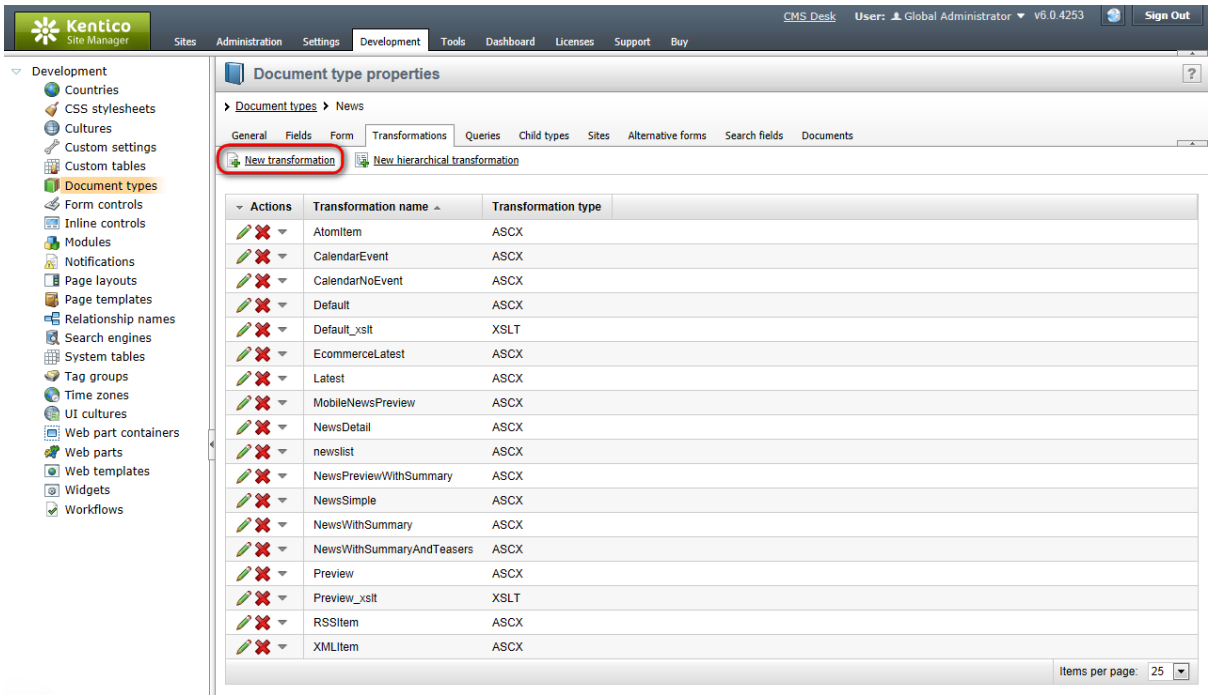


3. In the web part properties, enter *PrintLink* as the **ID** and choose **CMS.News** in the **Show for document types** text box. Enter the following code into the **Text** field. Then click **OK**.

```
<div class="PrintLink">
       <br />
       <a href="~/SpecialPages/Print.aspx?printpath={%NodeAliasPath%}&classname={%
ClassName%}" target="_blank" >
              <img class="PrintImage" src="~/App_Themes/CorporateSite/Images/
Print.gif" alt="Print" />
              Print
       </a>
</div>
```

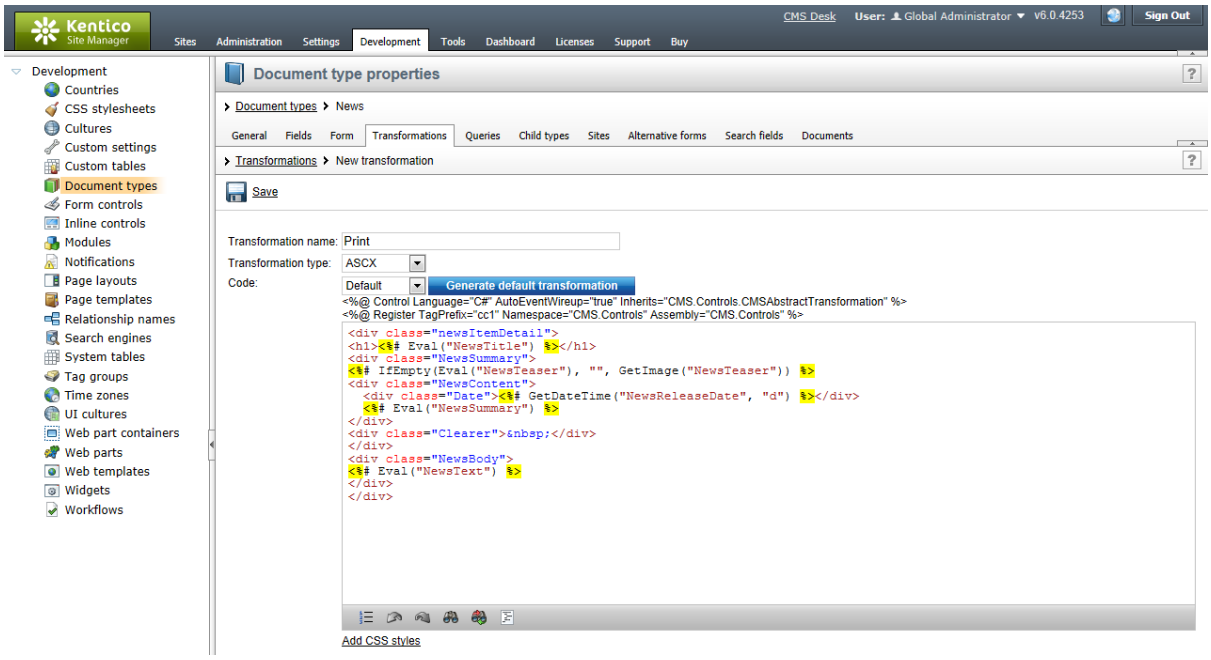


4. Now you have to specify the Print transformation for the new document type. Go to **CMS Site Manager** -> **Development** -> **Document Types** and click the **Edit** (🖉) button next to the **News** document type. Switch to the **Transformation** tab and click **New Transformation**.

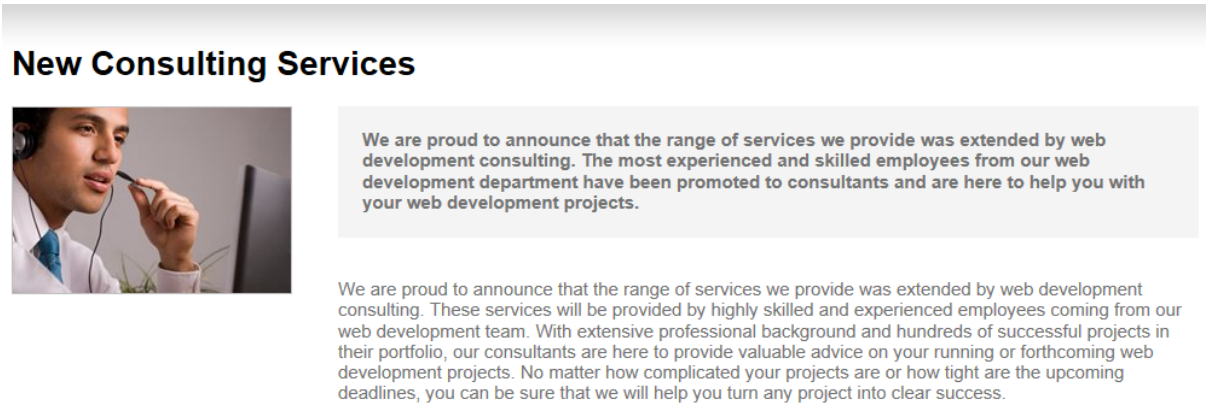


5. Enter *Print* as the **Transformation name** and enter the following code into the **Code** text box. Then click **Save**.

```
<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>
    <%# Eval("NewsSummary") %>
  </div>
  <div class="Clearer"> </div>
</div>
<div class="NewsBody">
  <%# Eval("NewsText") %>
</div>
</div>
```



6. Now, go back to **CMS Desk** -> **Content** -> **News** -> **Your second news** and click the newly created **Print** button. You will be redirected to the **Print** page that displays the print version of the given news item.

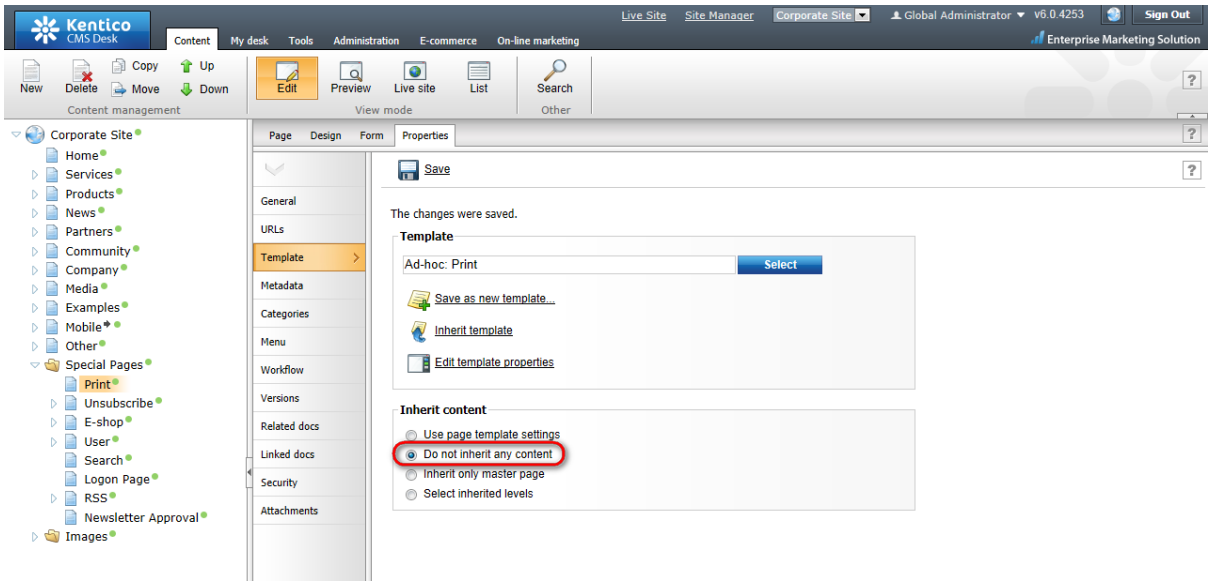


Creating the Print page on your site

The **Print** page is already created on 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 blank page to the **Special pages** folder and name it **Print**.

2. Firstly, you have to disable content inheritance at **Properties** -> **Template**.



3. Every **Print** page contains the repeater web part that renders the **Print** transformation for the given document type. The transformation can be specified at **Site Manager** -> **Development** -> **Document Types** -> **<edit (🖉) document type>** -> **Transformations**.

Therefore, add the **Repeater** web part and set its properties to the following values:

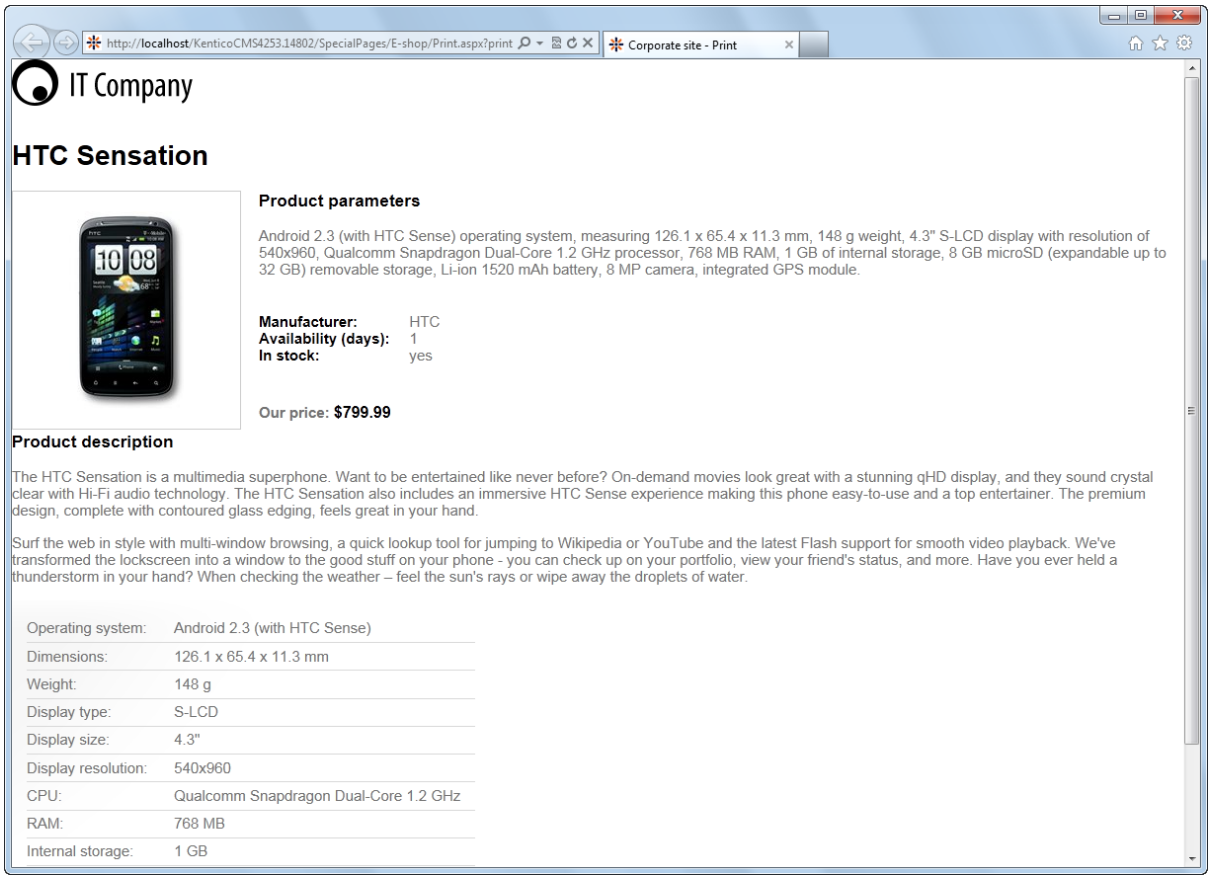
- **Path:** {?printpath|/%?}
- **Document types:** {?classname|cms.root?}
- **Transformation:** {?classname|cms.root?}.print

The **Path** text box specifies the path to the document whose print version you want to make. The value of the **printpath** macro expression is supplied in URL. If no value is supplied in URL (for instance you go directly on the **Print** page from the content tree and not through the print link button), the default **Print** transformation for **cms.root** is displayed.

The **Document types** text box specifies the document types that should be displayed.

The **Transformation** text box specifies the name of the transformation that should be used to render the print version. If the **Print** transformation is not defined for the given document type, the **Print** transformation for the **cms.root** is used.

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.



7.4.6 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):

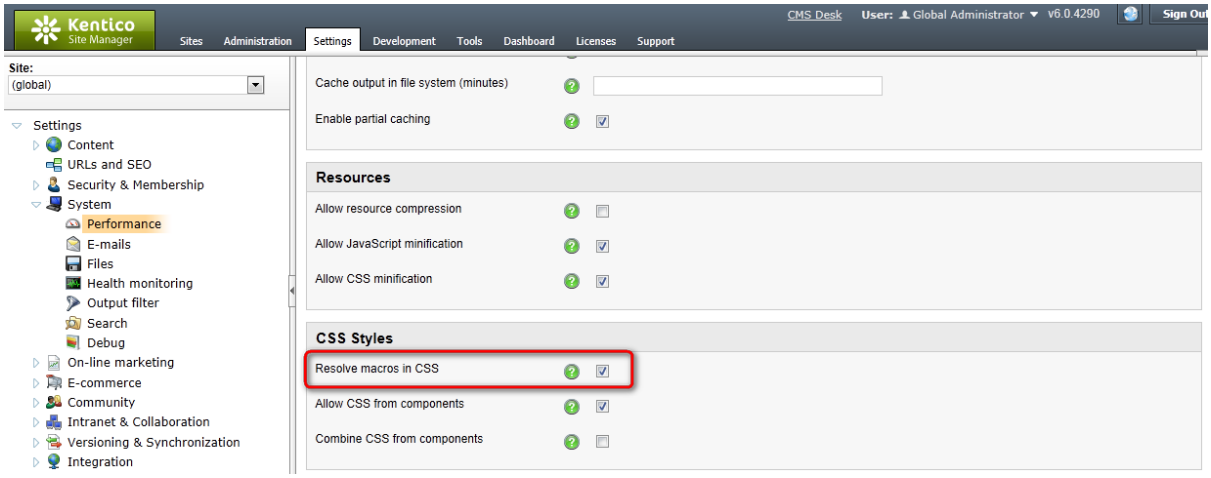
- *{% CSS.Containers.<web part container code name> %}*, e.g. *{% CSS.Containers.BlackBox %}*
- *{% CSS.Templates.<page template code name > %}*, e.g. *{% CSS.Templates.MyTemplate %}*
- *{% CSS.WebParts.<web part code name> %}*, e.g. *{% CSS.WebParts.AbuseReport %}*

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);
```

7.4.7 CSS stylesheet internals and API

7.4.7.1 Overview

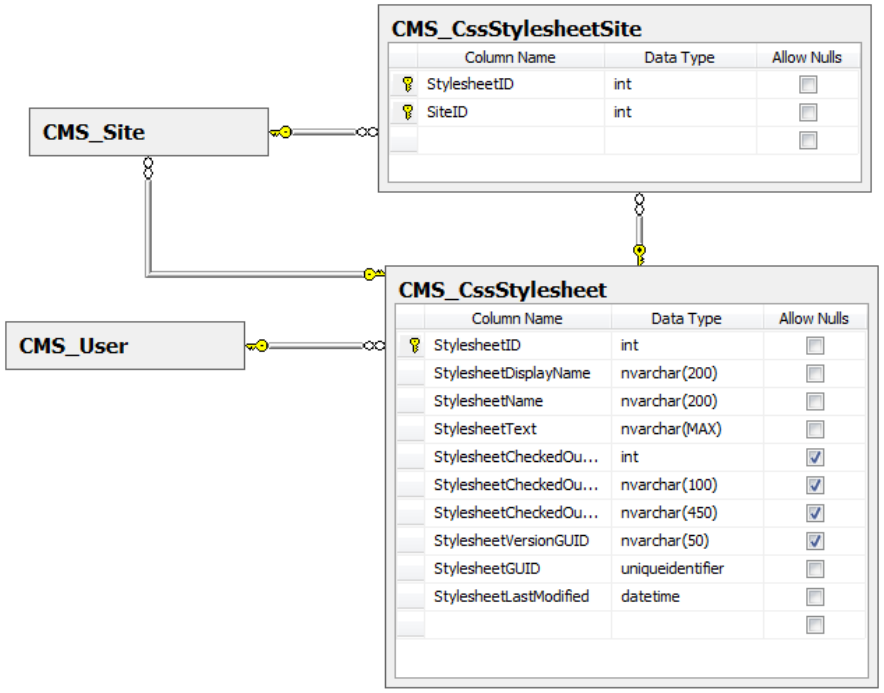
In this chapter, you will learn which database tables and API classes are used by CSS stylesheets. You will also see the most common API examples.

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.

7.4.7.2 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.



7.4.7.3 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.

7.4.7.4 API examples

7.4.7.4.1 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;
```

7.4.7.4.2 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);
}
```

7.4.7.4.3 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;
}
```

7.5 Debugging and system information

7.5.1 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.

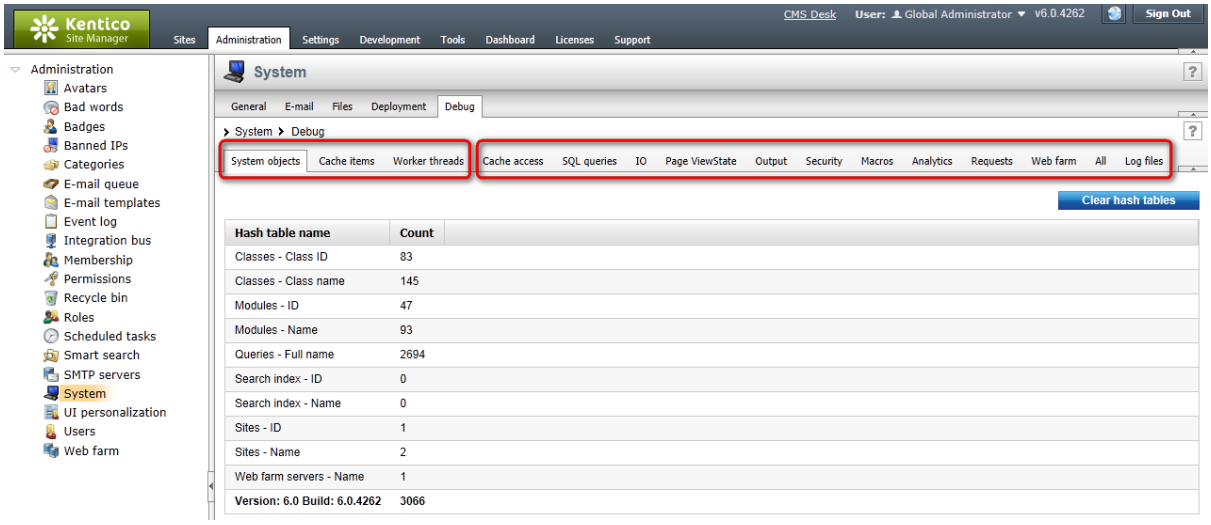
The debugging interface is divided into several sub-tabs. Only the following two tabs are displayed by default. Click the name of the tab to learn more.

- System objects
- Cache items
- Worker threads

The other sub-tabs are displayed only after enabling certain settings in **Site Manager -> Settings -> System -> Debug** 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.

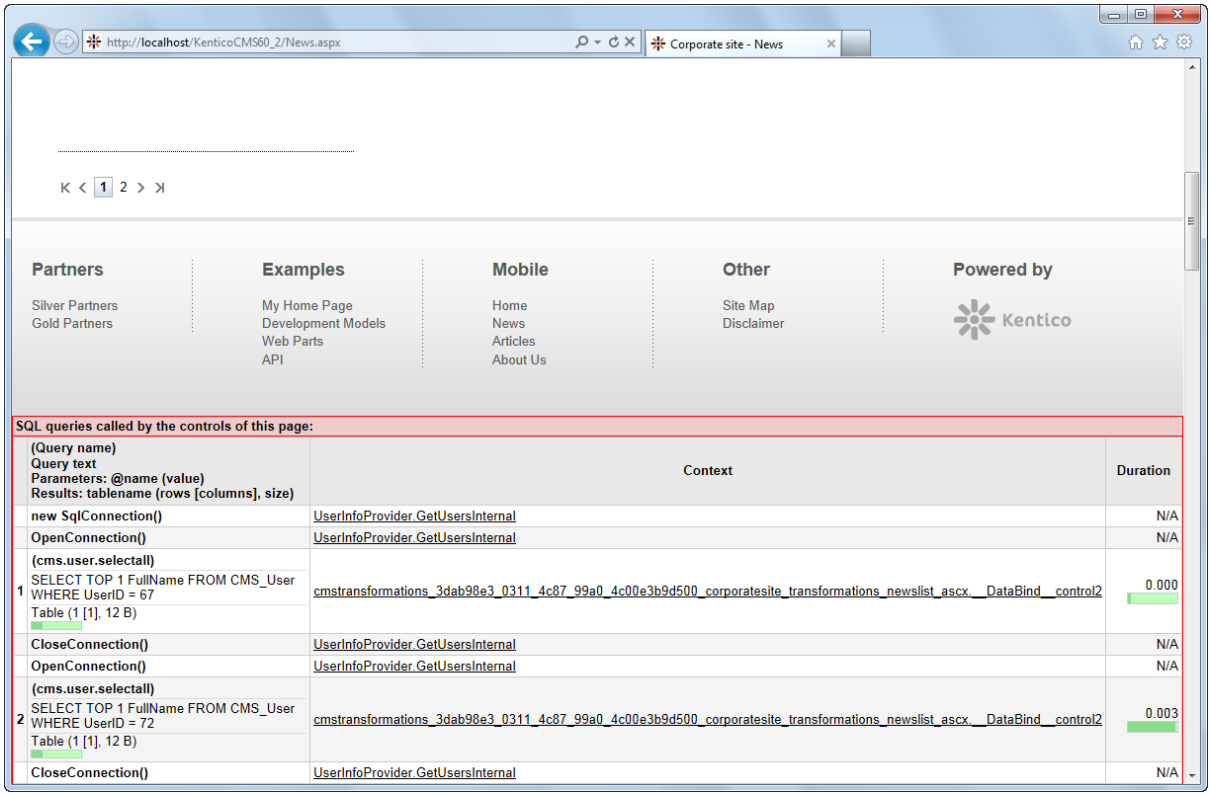
- Cache access
- SQL queries
- IO
- Page ViewState
- Output
- Security
- Macros
- Analytics
- Requests
- Web farm
- All
- Log files

You can also enable all these debugs at once using the general settings and keys.



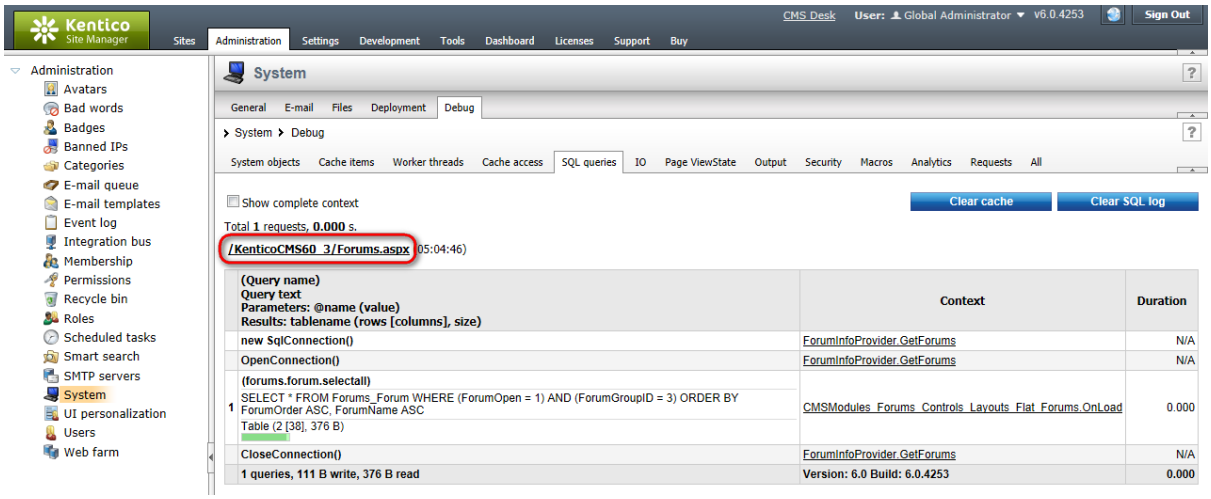
Debugging on the live site

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.

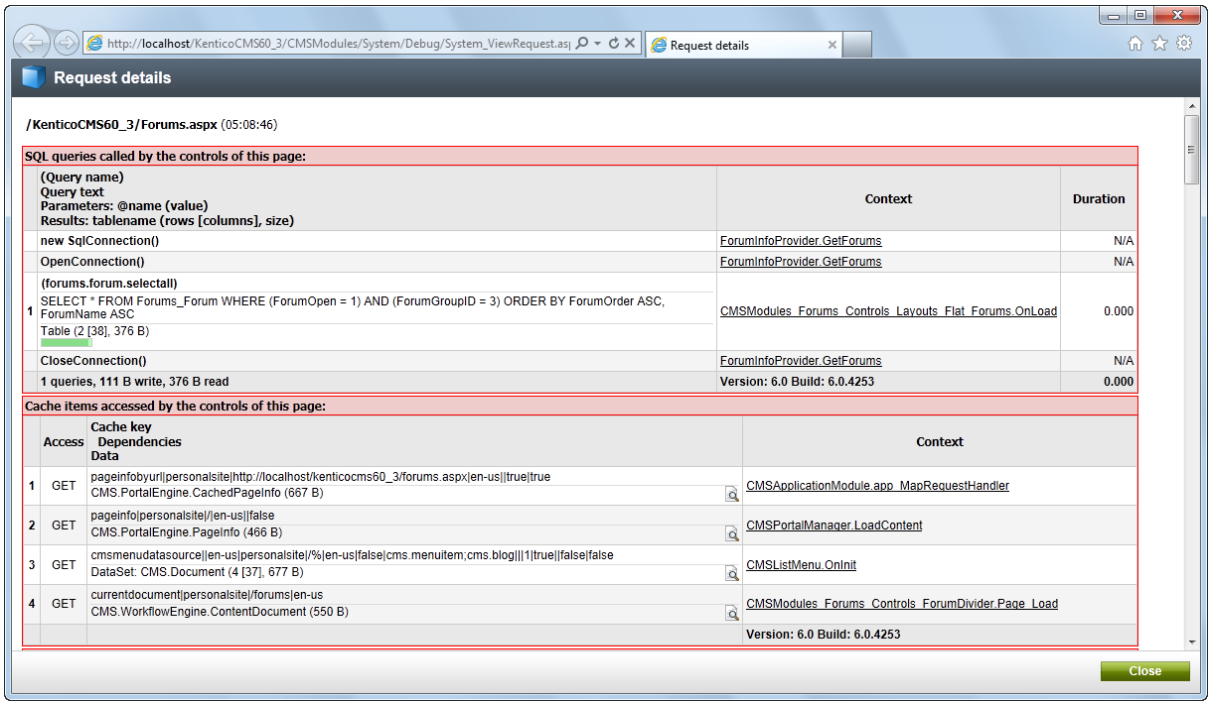


Debug request details

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 gets opened, listing the request's debug log for all enabled debugs.



7.5.2 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 system database (the actual database data (the *.mdf* file) + database log (the *.ldf* file)).

Using the **Refresh interval (seconds)** drop-down list, you can set the interval after which values in the fields 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.

- **Clear counters** - clears values currently stored in all performance counters registered for the current Kentico CMS instance. See Modules -> Health monitoring for more details.

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.

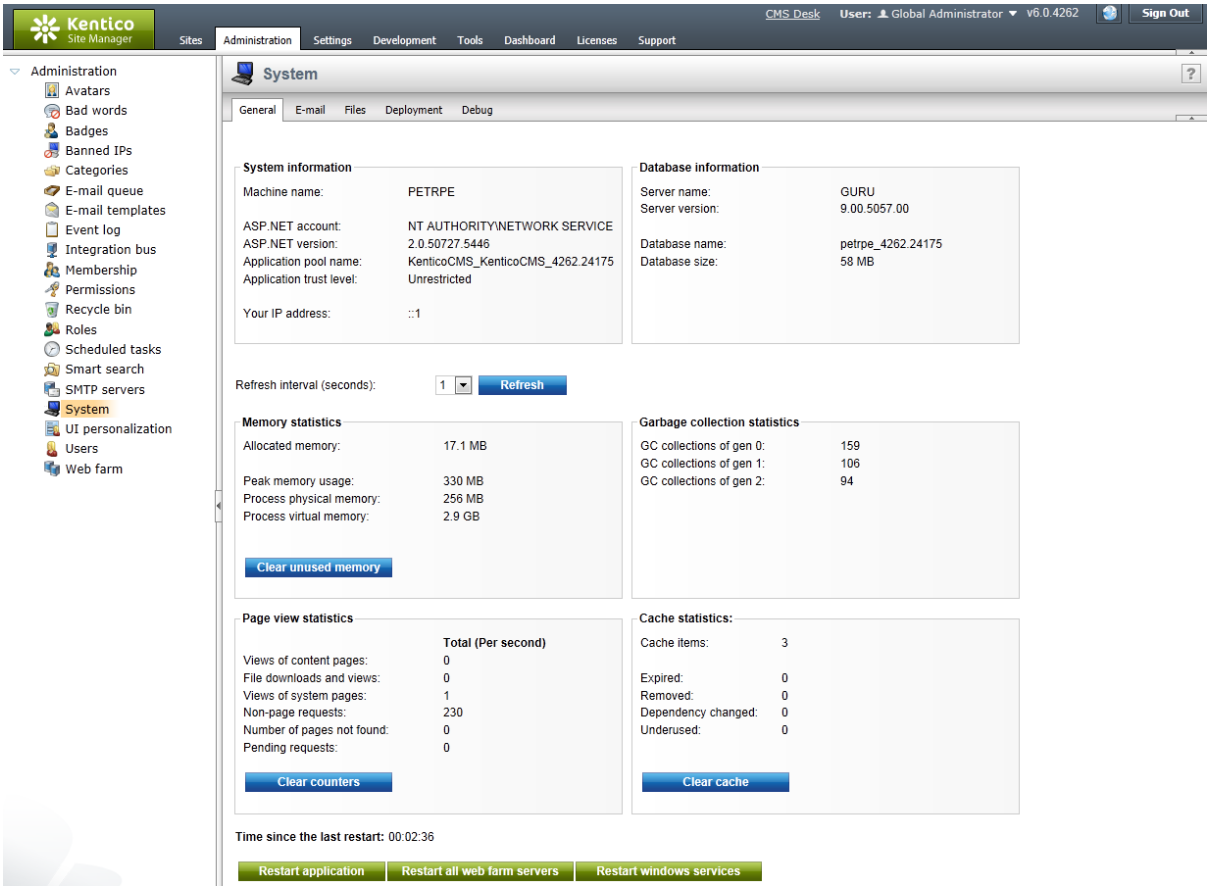
- **Time since last restart** - time since the system was last restarted (or since it was launched if no restart was performed).

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.



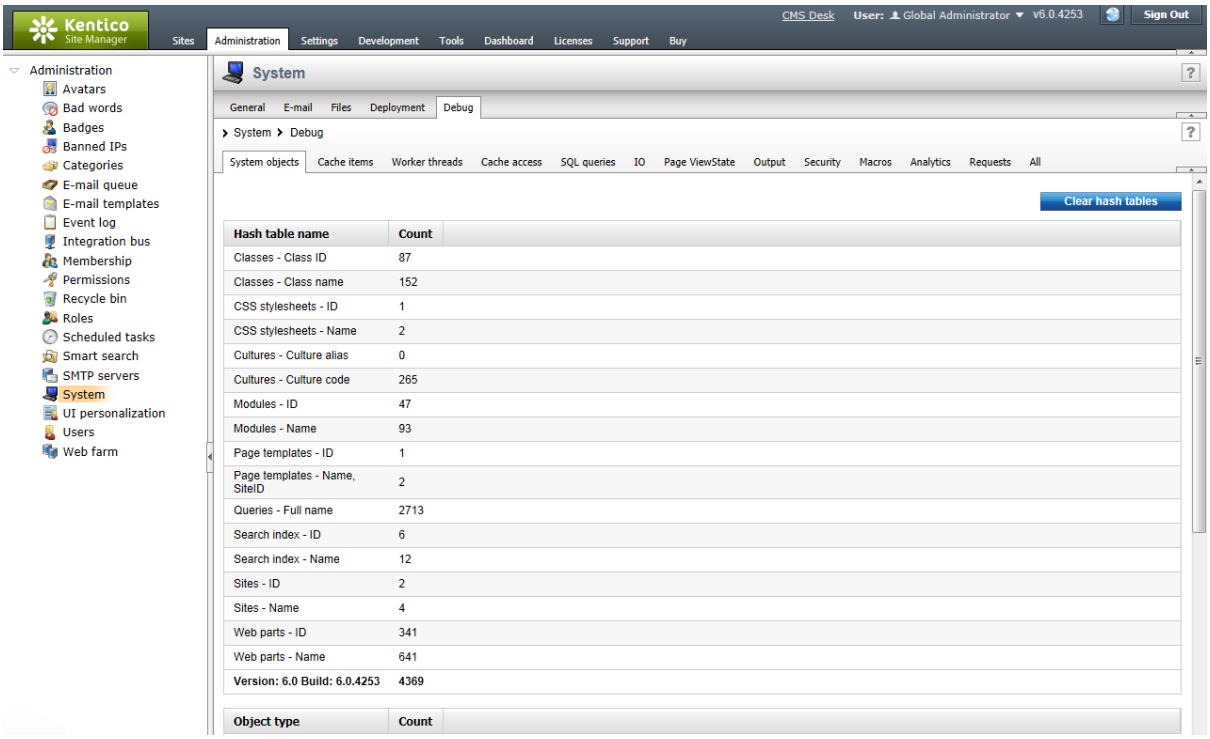
7.5.3 Particular debug tabs

7.5.3.1 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.

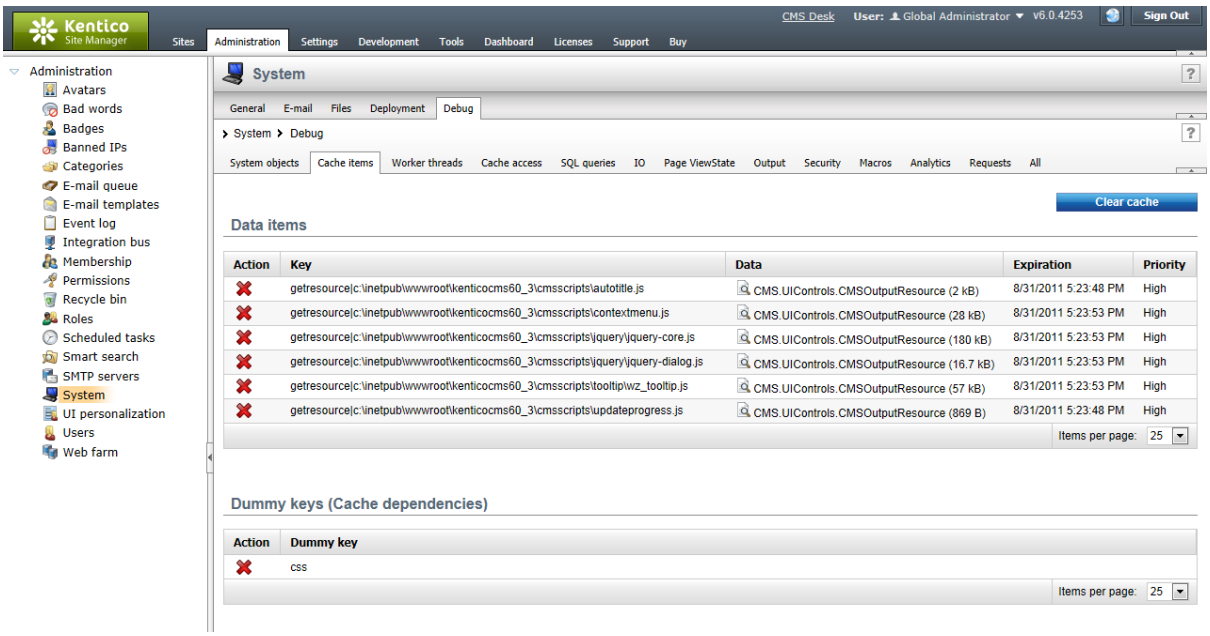
All hash tables can be cleared using the **Clear hash tables** button.



7.5.3.2 Cache items

On the **Debug -> Cache items** tab, you can see which **Keys** and **Dummy keys** are currently stored in the system cache.

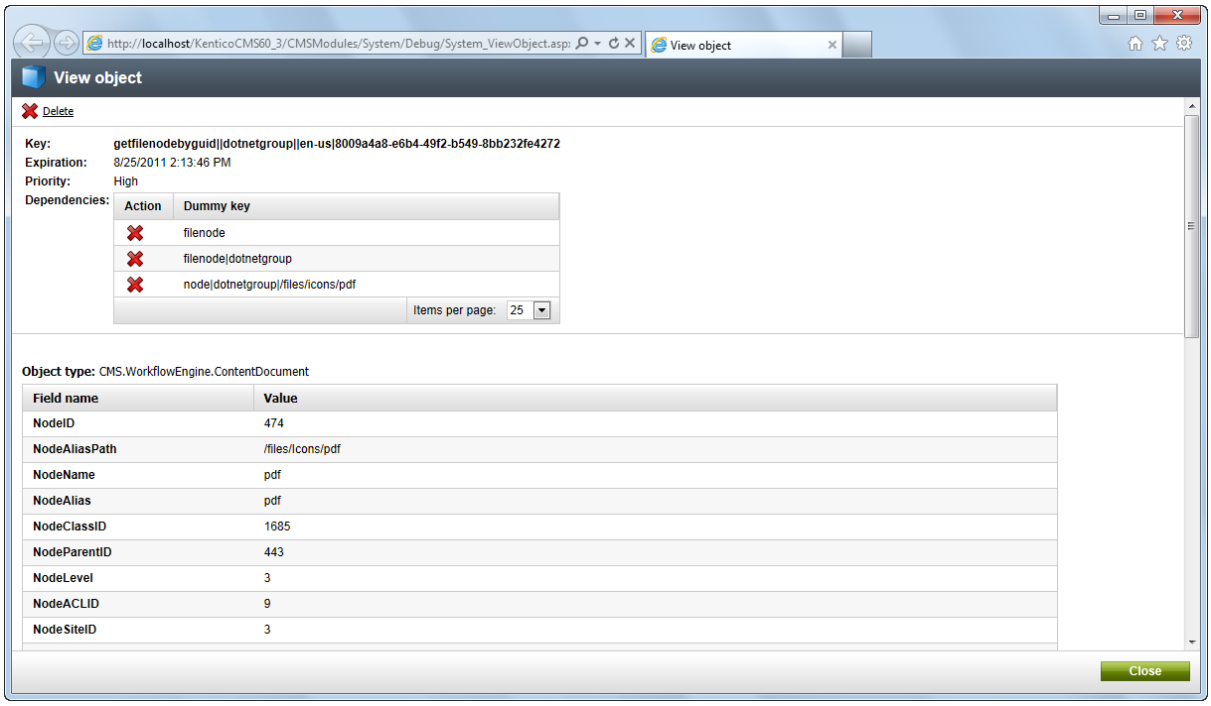
Both real and dummy keys can be deleted from within this part of the user interface using the **Delete** (✖) icon next to them. All dummy keys can be cleared from the cache using the **Clear cache** button.



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.

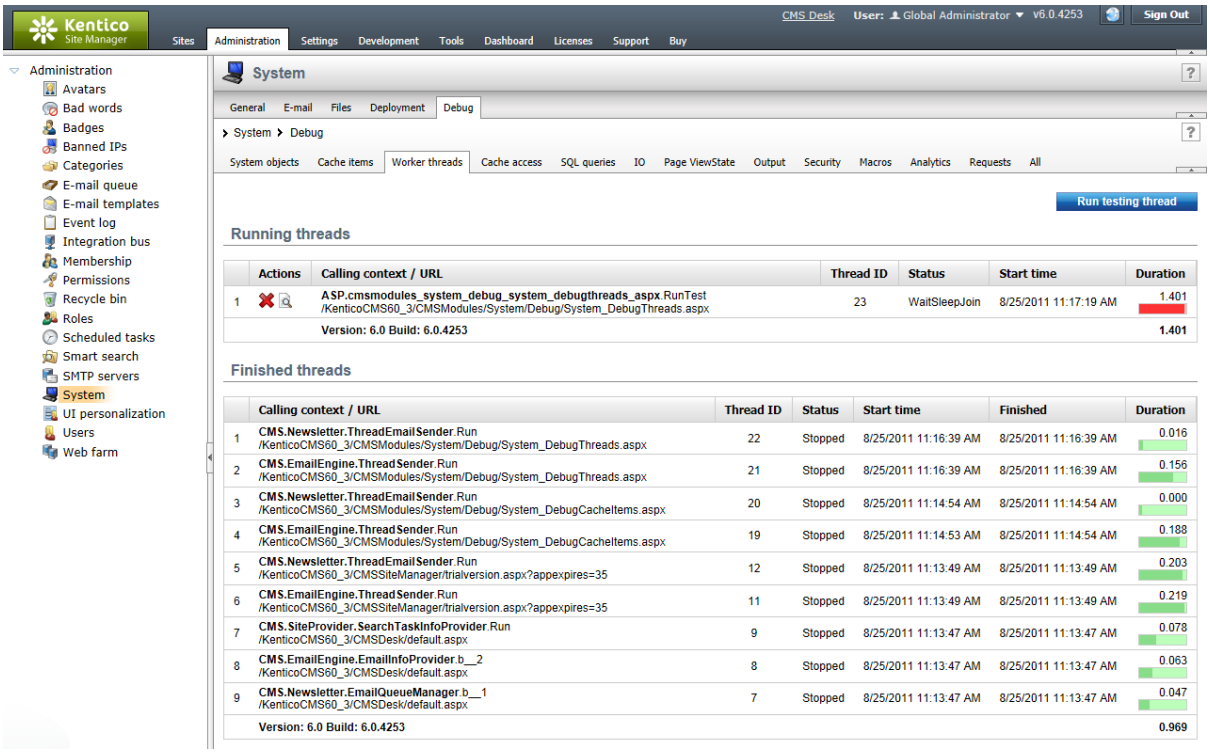
Finally, the table under the **Object type** value represents the actual data of the cached object. Displayed fields vary based on the currently displayed object type. The cache item can be cleared from cache using the **Delete** button in the top part of the window.



For more information about caching in Kentico CMS, please refer to Development -> Caching and performance -> Caching options.

7.5.3.3 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.



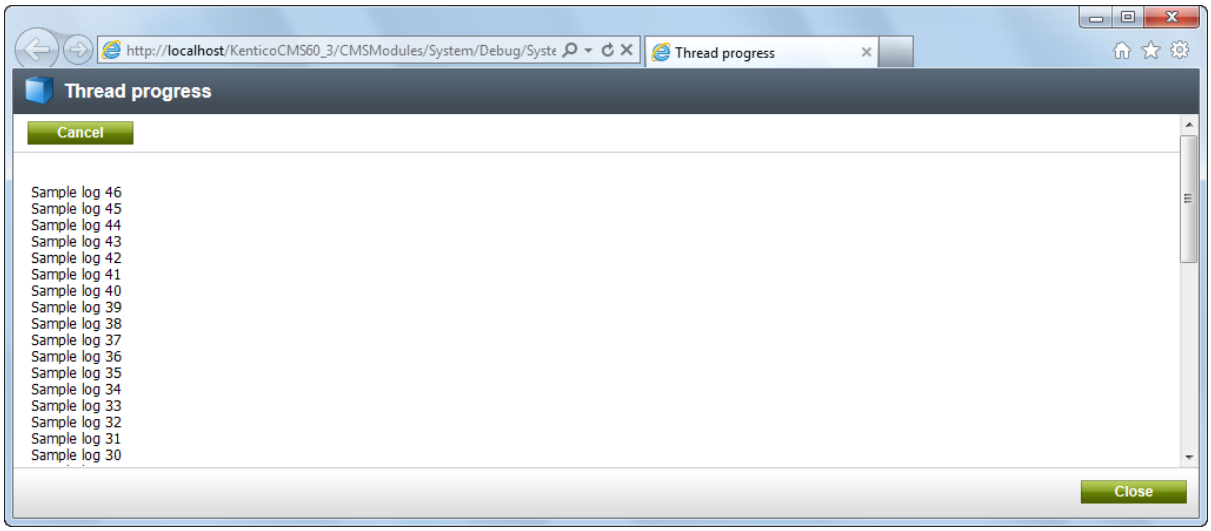
The UI is split into two sections:

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.

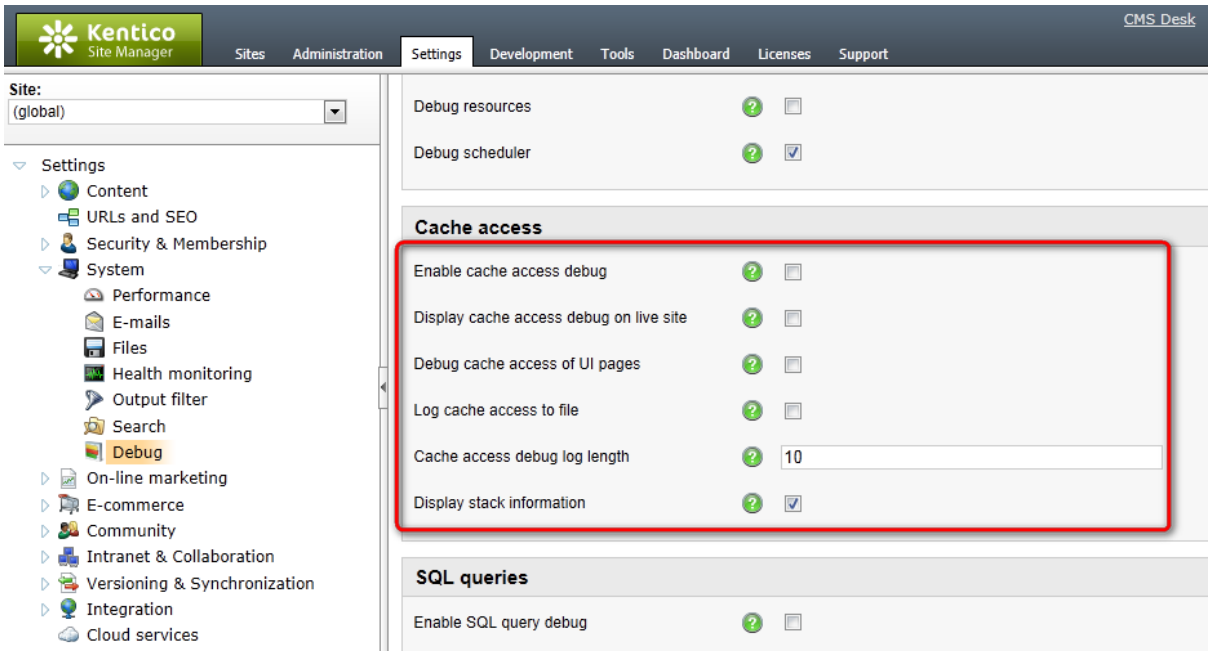
```
<add key="CMSMaxFinishedThreadsLogged" value="10" />
```

7.5.3.4 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" />
```

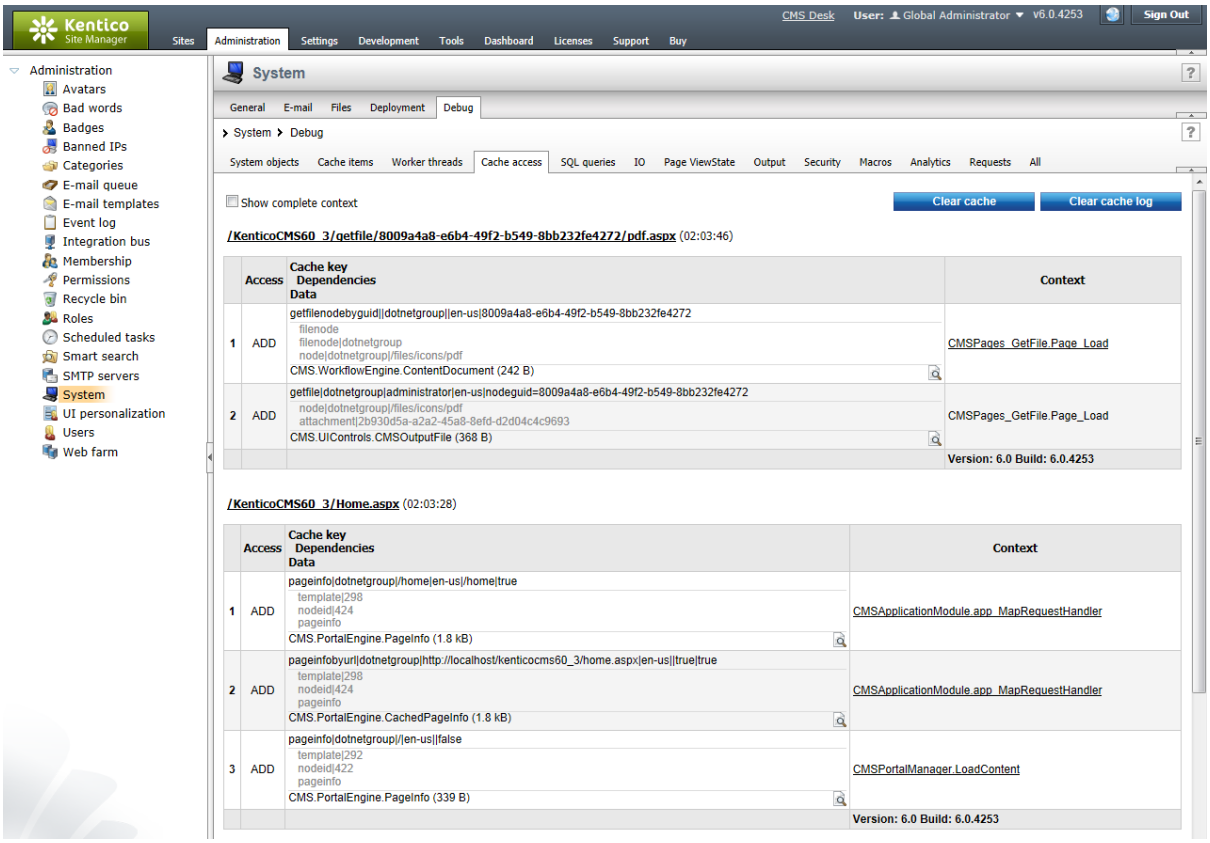
Cache access debugging can also be enabled using the general settings and keys.

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.

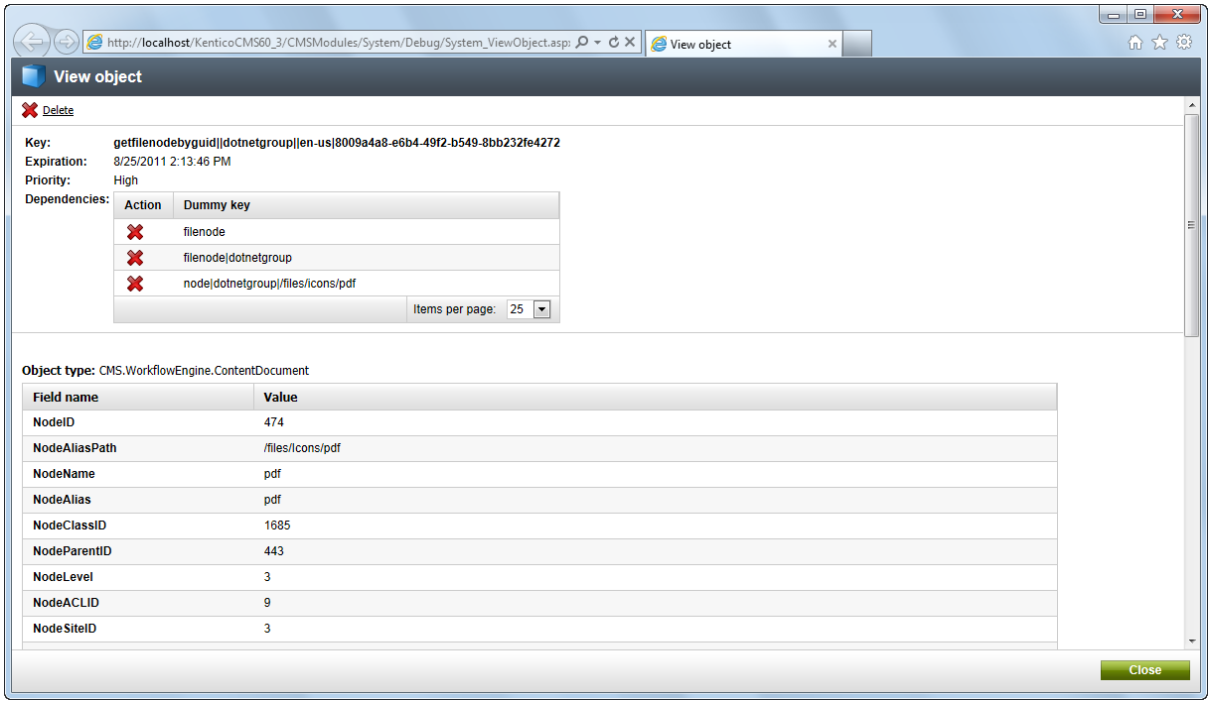
Enabling the **Show complete context** check-box displays complete context (not only the topmost item) in the **Context** column. All dummy keys can be cleared from the cache using the **Clear cache** button. Clicking the **Clear cache log** button clears all records in the log.



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.



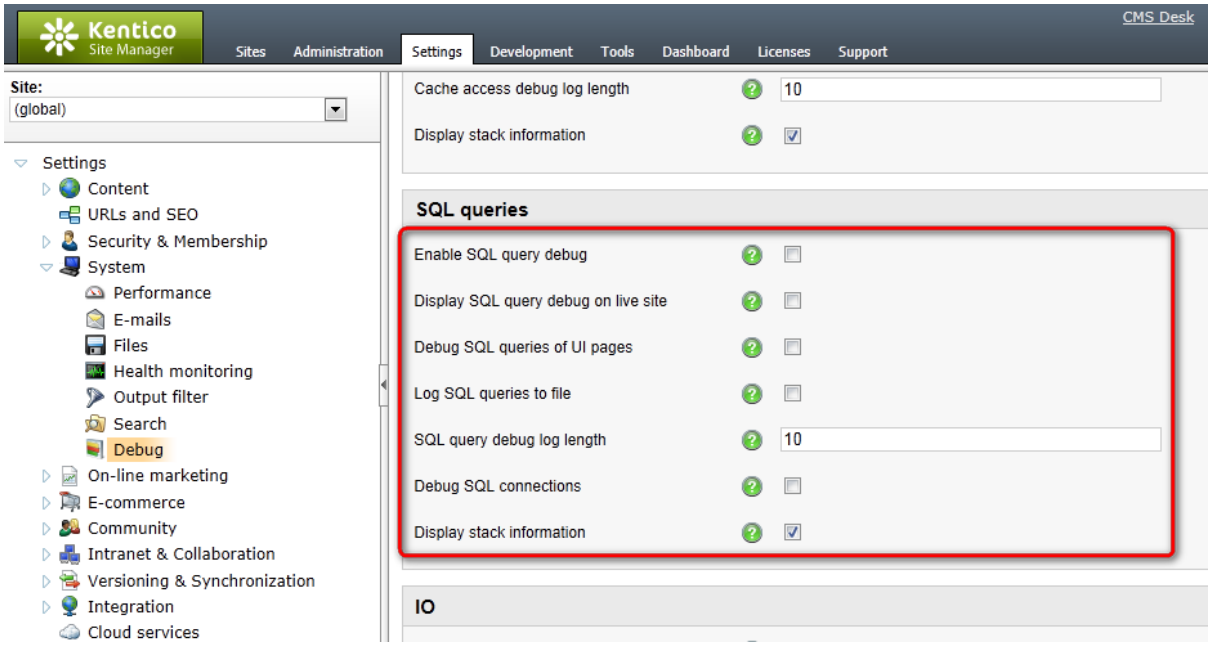
For more information about caching in Kentico CMS, please refer to Development -> Caching and performance -> Caching options.

7.5.3.5 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" />
```

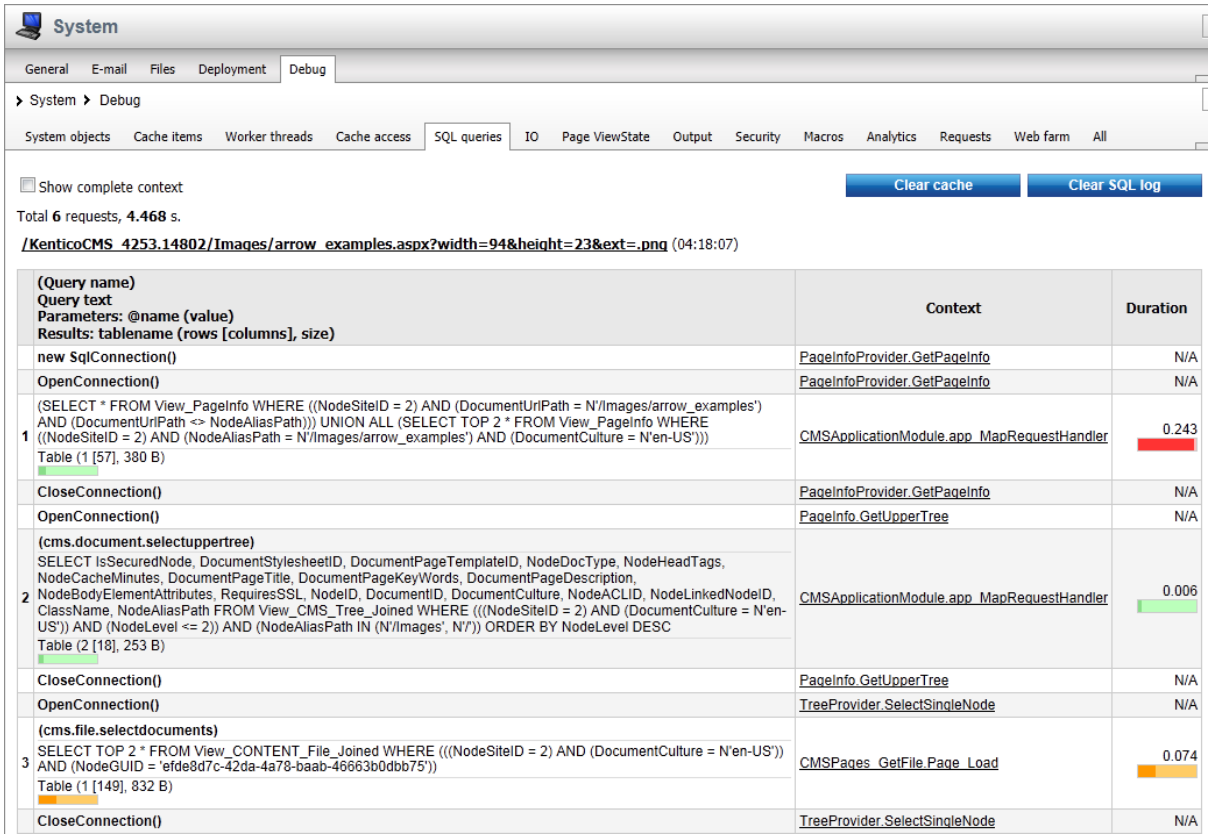
SQL query debugging can also be enabled using the general settings and keys.

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.

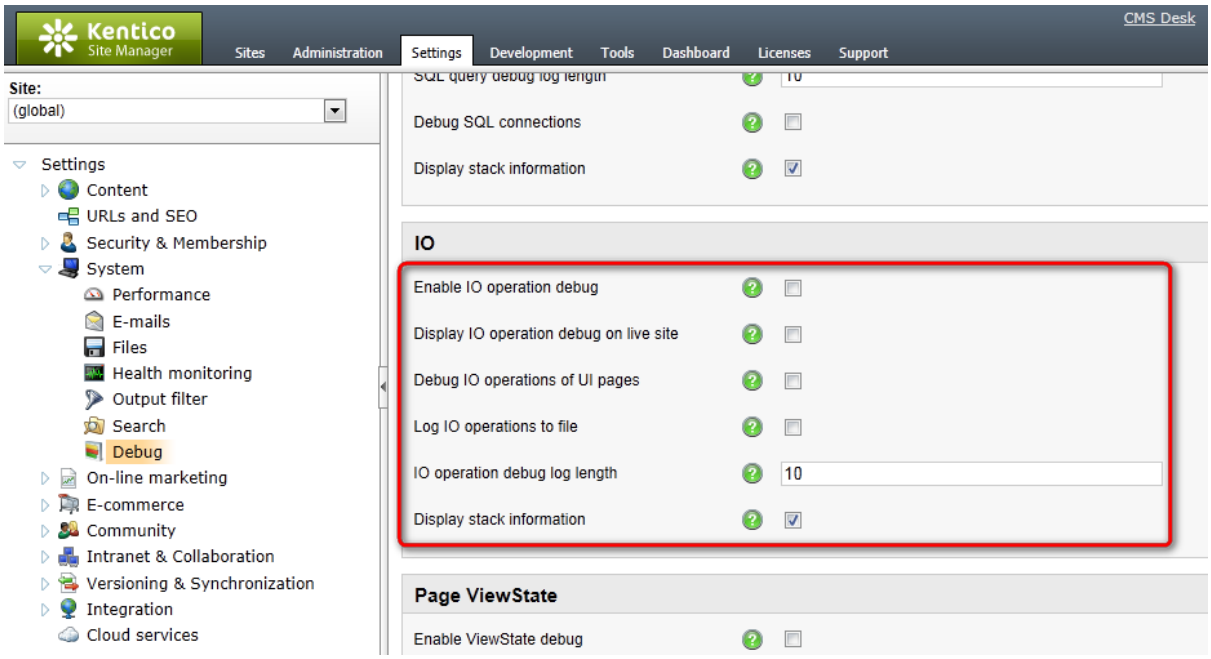
Enabling the **Show complete context** check-box displays complete context (not only the topmost item) in the **Context** column. Clicking the **Clear SQL log** button clears all records in this debug log.



7.5.3.6 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*:

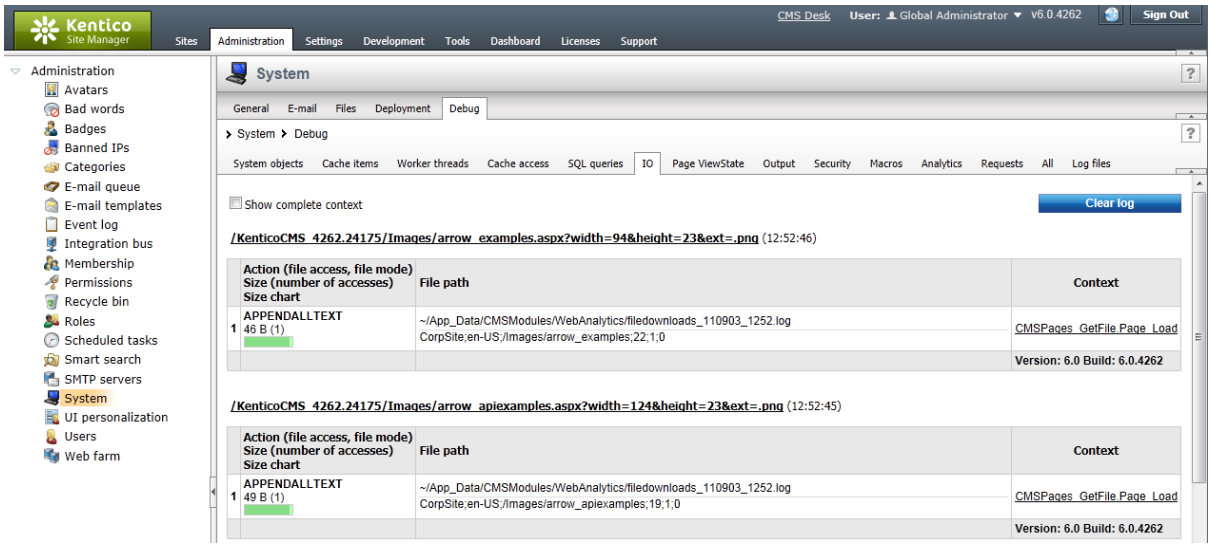
```
<add key="CMSDebugFiles" value="true" />
<add key="CMSDebugFilesLive" value="true" />
<add key="CMSDebugAllFiles" value="true" />
<add key="CMSLogFiles" value="true" />
<add key="CMSDebugFilesLogLength" value="10" />
<add key="CMSDebugFilesStack" value="true" />
```

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.

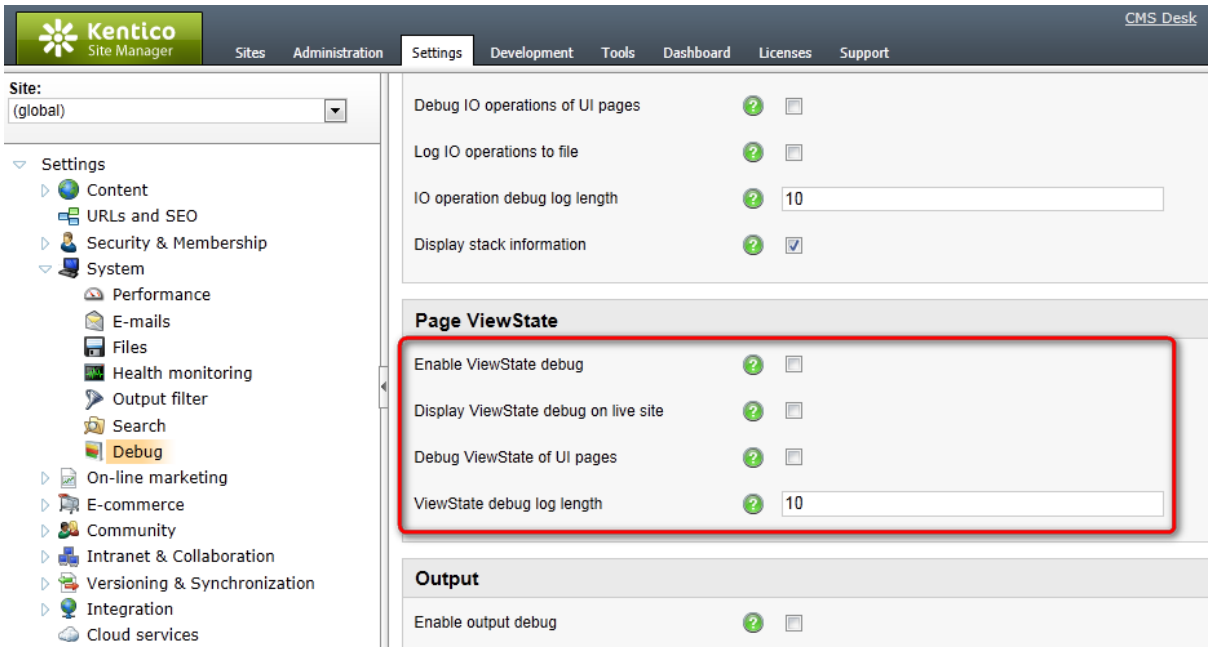
Enabling the **Show complete context** check-box displays complete context (not only the topmost item) in the **Context** column. The log can be cleared using the **Clear log** button.



7.5.3.7 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	CMSDebug ViewState	Enables ViewState debugging and the **Page ViewState** tab in **Site Manager -> Administration -> System -> Debug**.
Display ViewState debug on live site	CMSDebug ViewStateLive	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	CMSDebugAll ViewStates	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	CMSDebug ViewStateLogLength	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" />
```

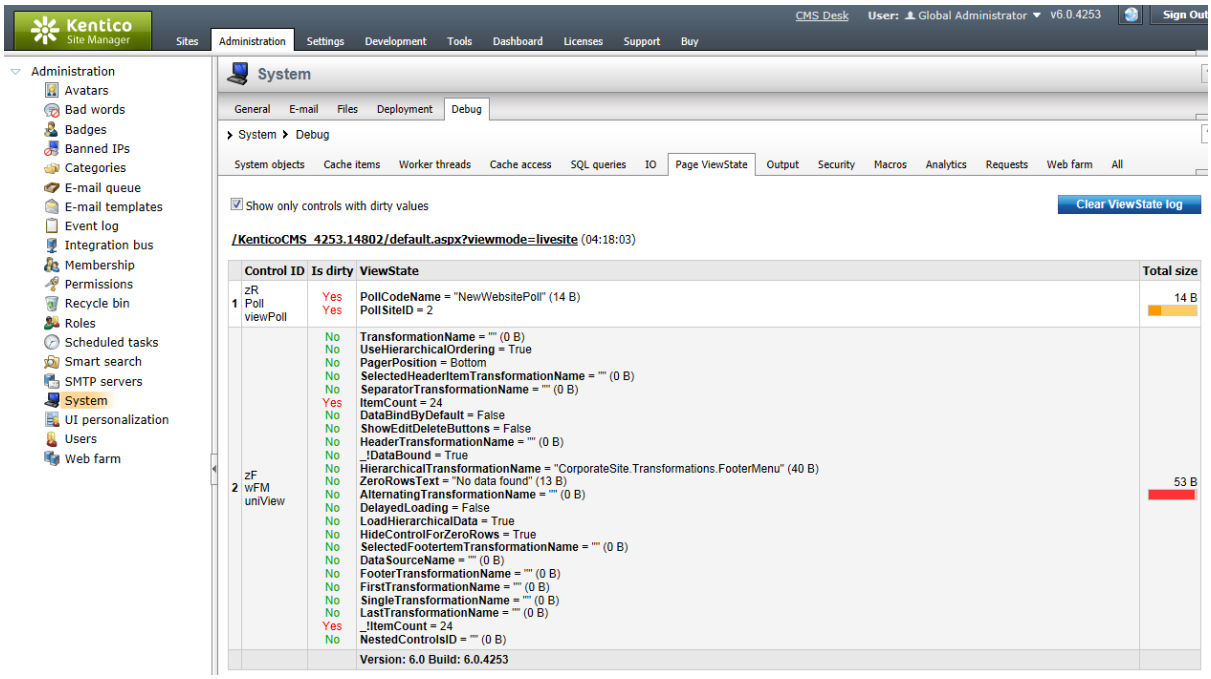
Page ViewState debugging can also be enabled using the general settings and keys.

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.

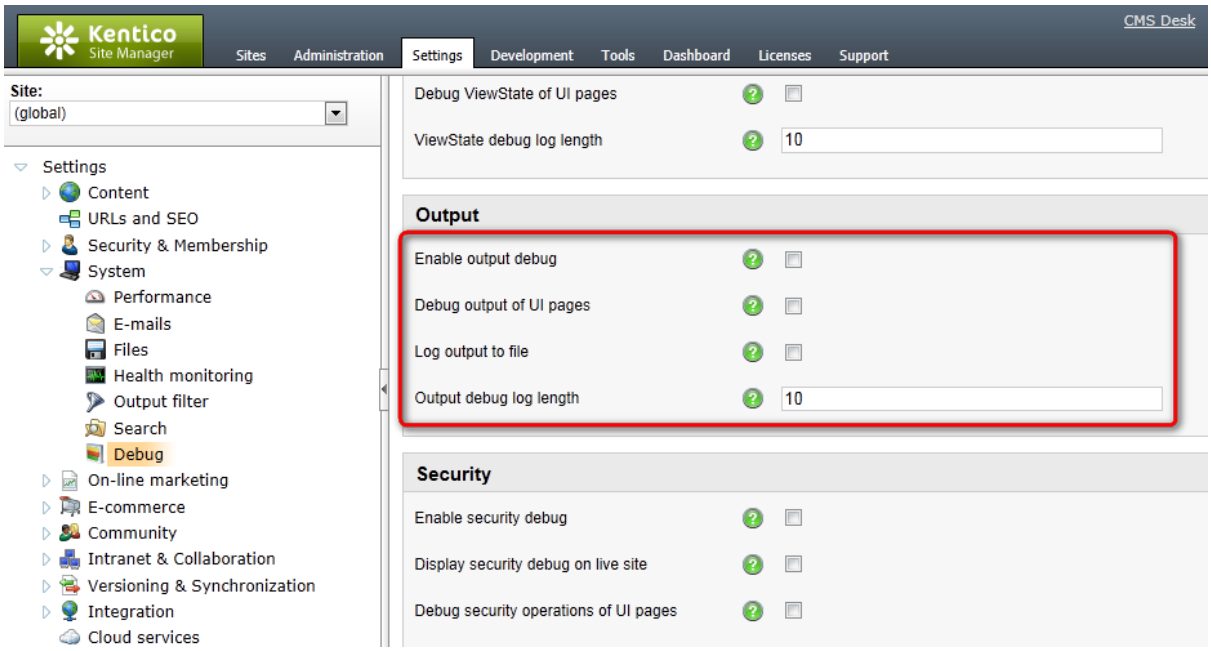
Please note: ViewState of the controls is retrieved using reflection. Therefore this debug may not work properly in a medium trust environment or other specific circumstances.



7.5.3.8 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*:

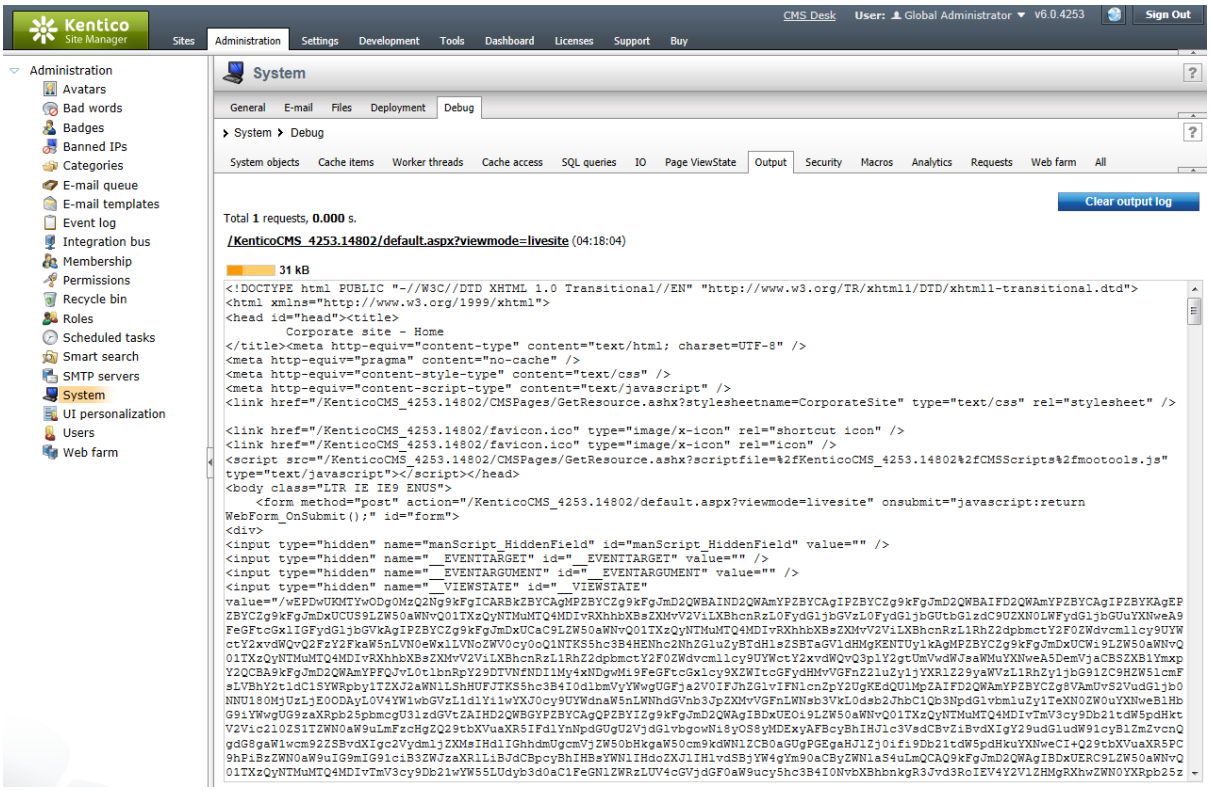
```
<add key="CMSDebugOutput" value="true" />
<add key="CMSDebugAllOutputs" value="true" />
<add key="CMSLogOutputToFile" value="true" />
<add key="CMSDebugOutputLogLength" value="10" />
```

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.

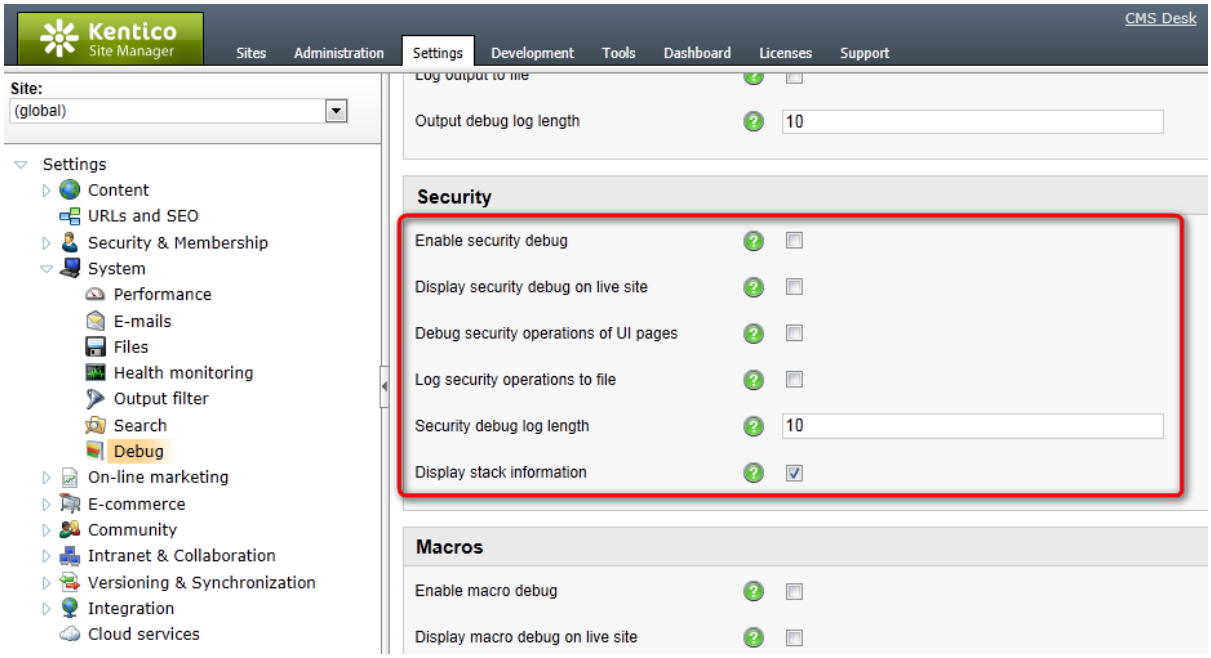


7.5.3.9 Security

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*:

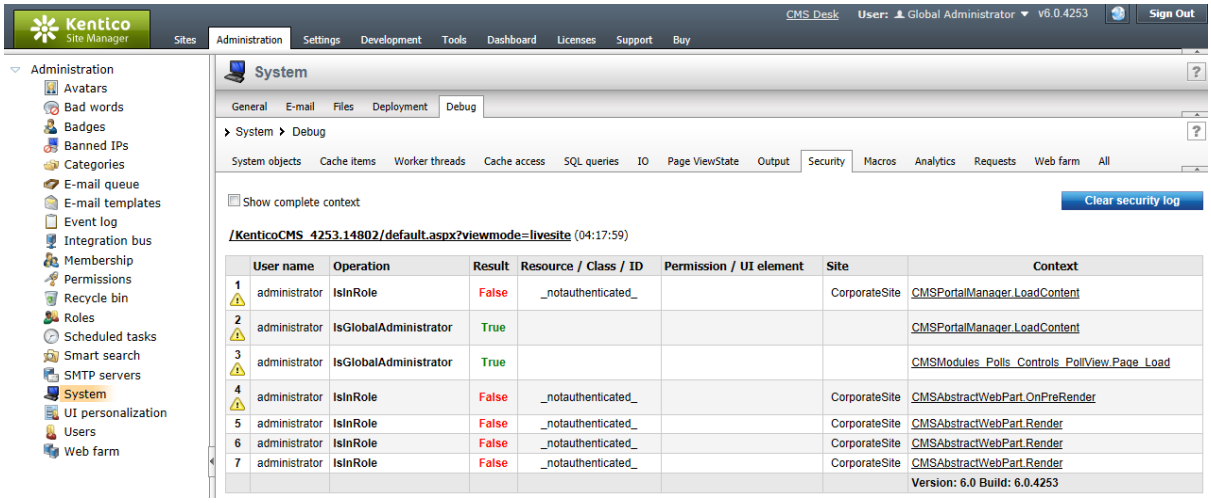
```
<add key="CMSDebugSecurity" value="true" />
<add key="CMSDebugSecurityLive" value="true" />
<add key="CMSDebugAllSecurity" value="true" />
<add key="CMSLogSecurity" value="true" />
<add key="CMSDebugSecurityLogLength" value="10" />
<add key="CMSDebugSecurityStack" value="true" />
```

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.

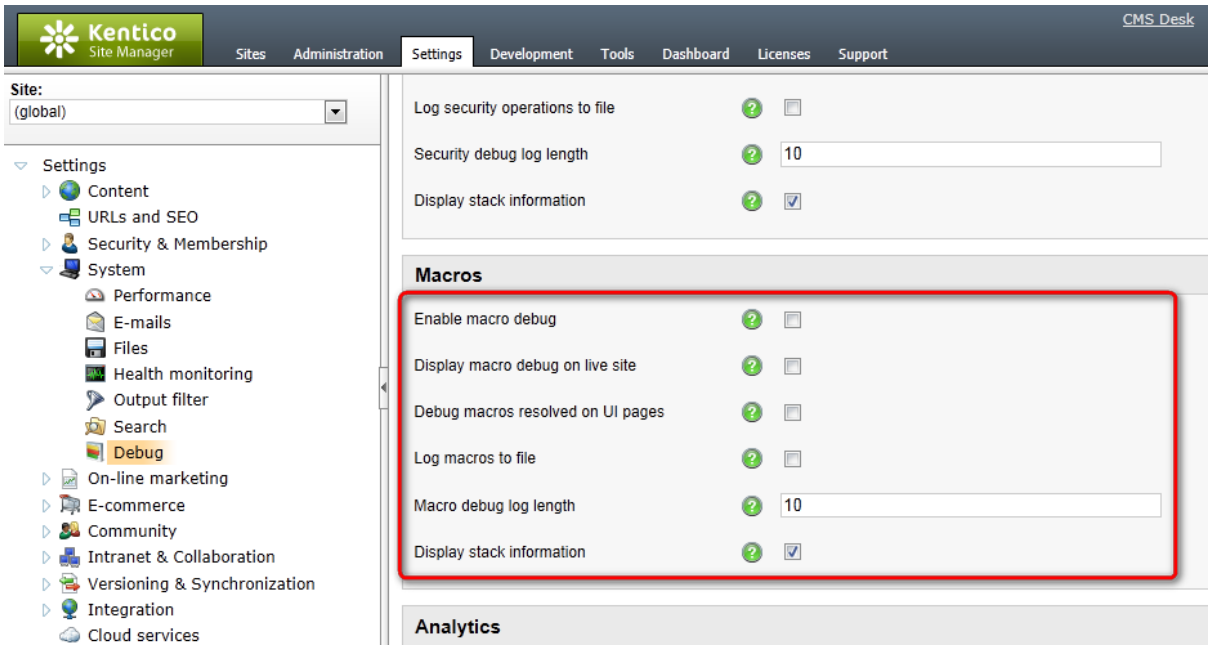


7.5.3.10 Macros

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**.
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" />
```

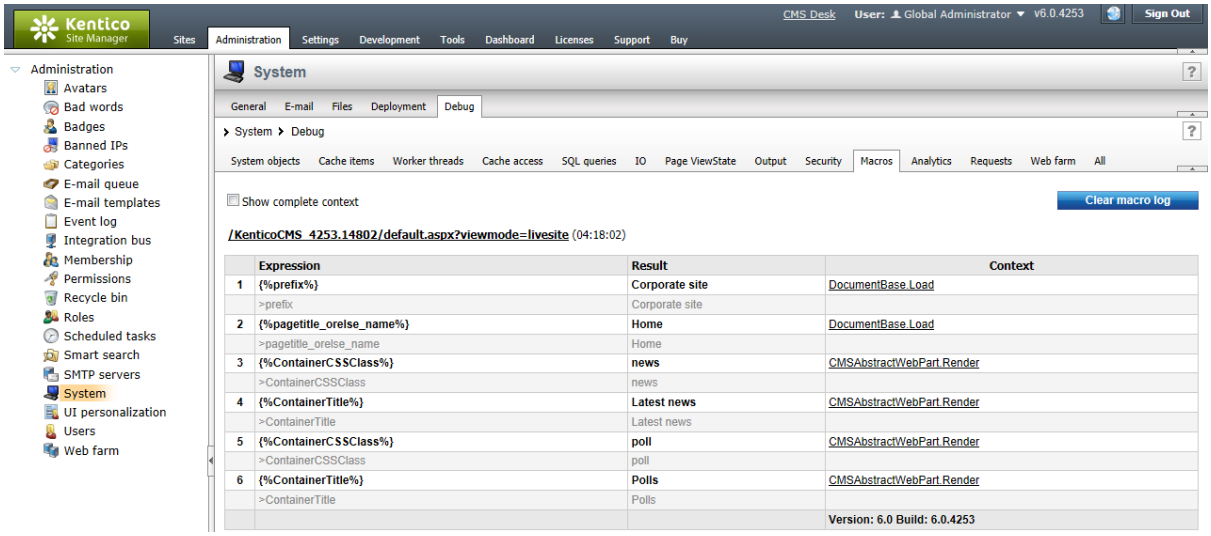
Macro debugging can also be enabled using the general settings and keys.

User interface

On the **Debug -> Macros** tab, you can see which macros were recently resolved. For each macro, its

exact expression, result (the value into which it was resolved) and the context from which it was called is logged. This is useful if you want to analyze how macros were 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 macro log** button.

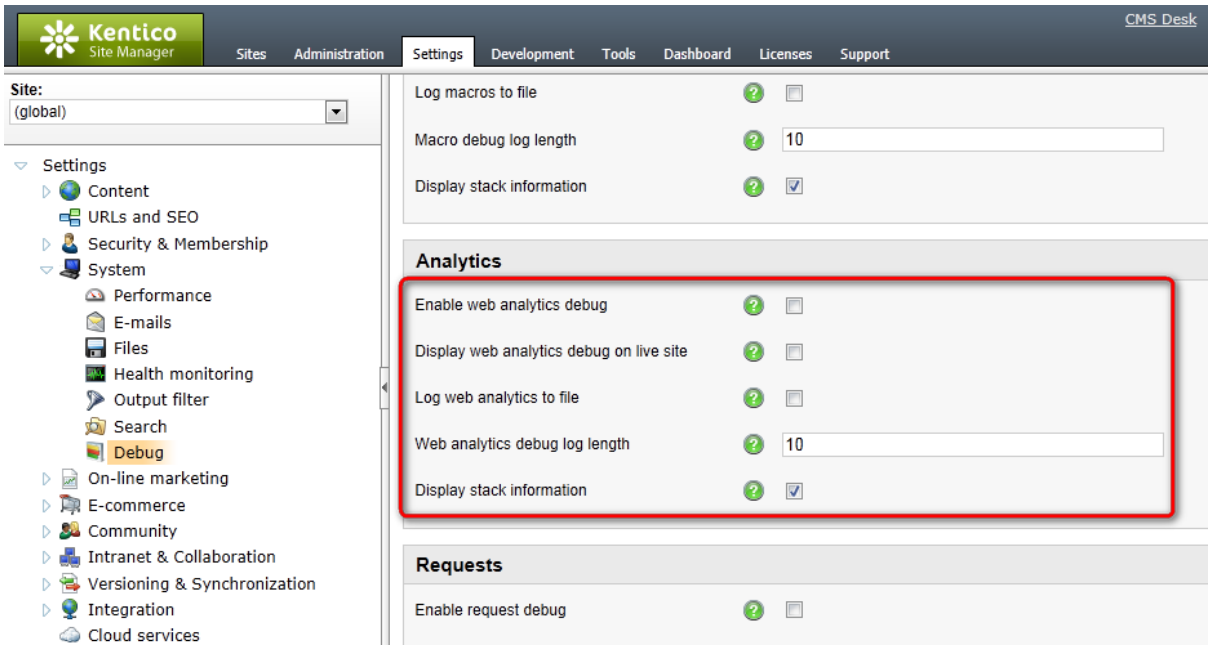


7.5.3.11 Analytics

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	CMSDebugAnalytics	If enabled, stack is tracked when debugging web

information	Stack	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" />
```

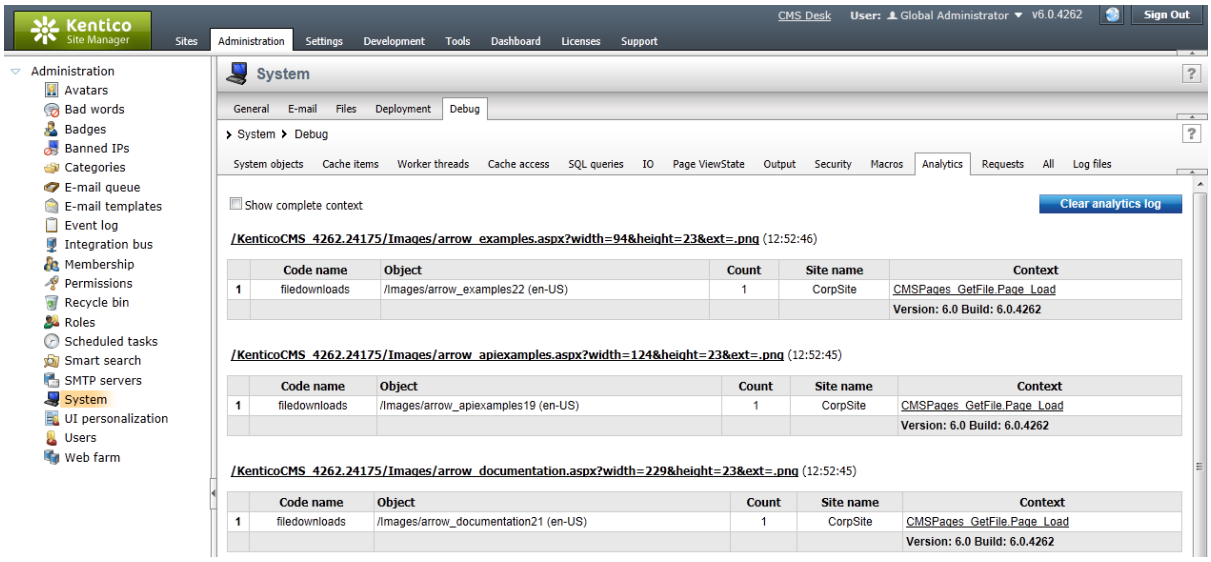
IO operation debugging can also be enabled using the general settings and keys.

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.

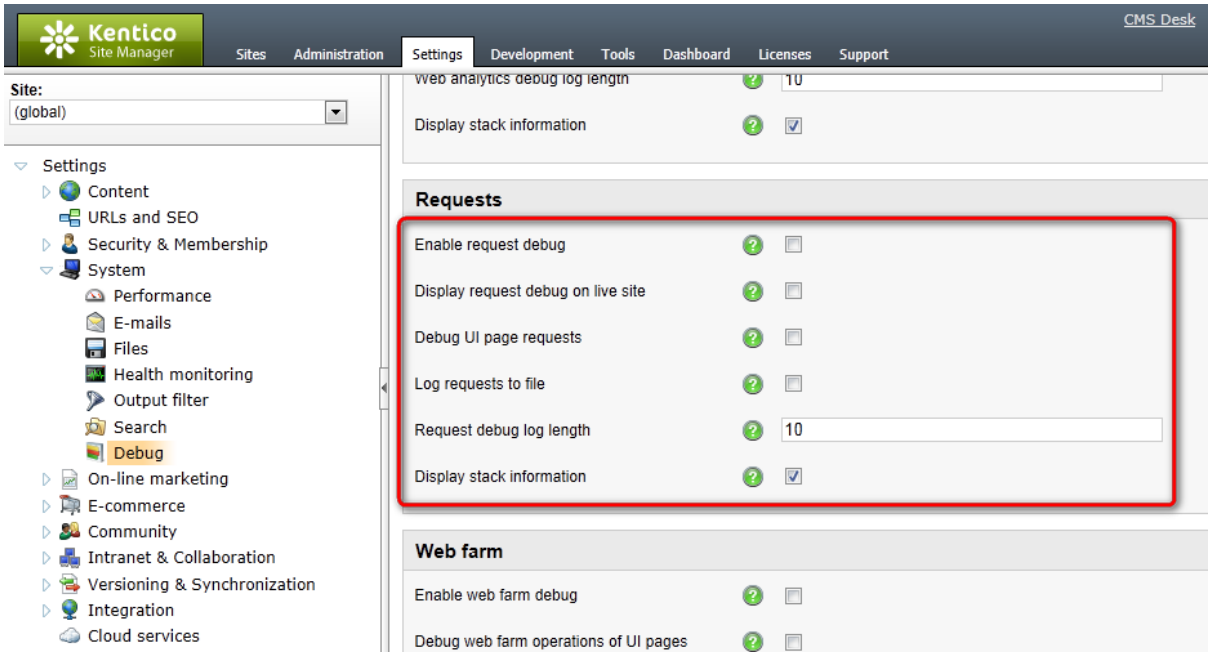


7.5.3.12 Requests

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" />
```

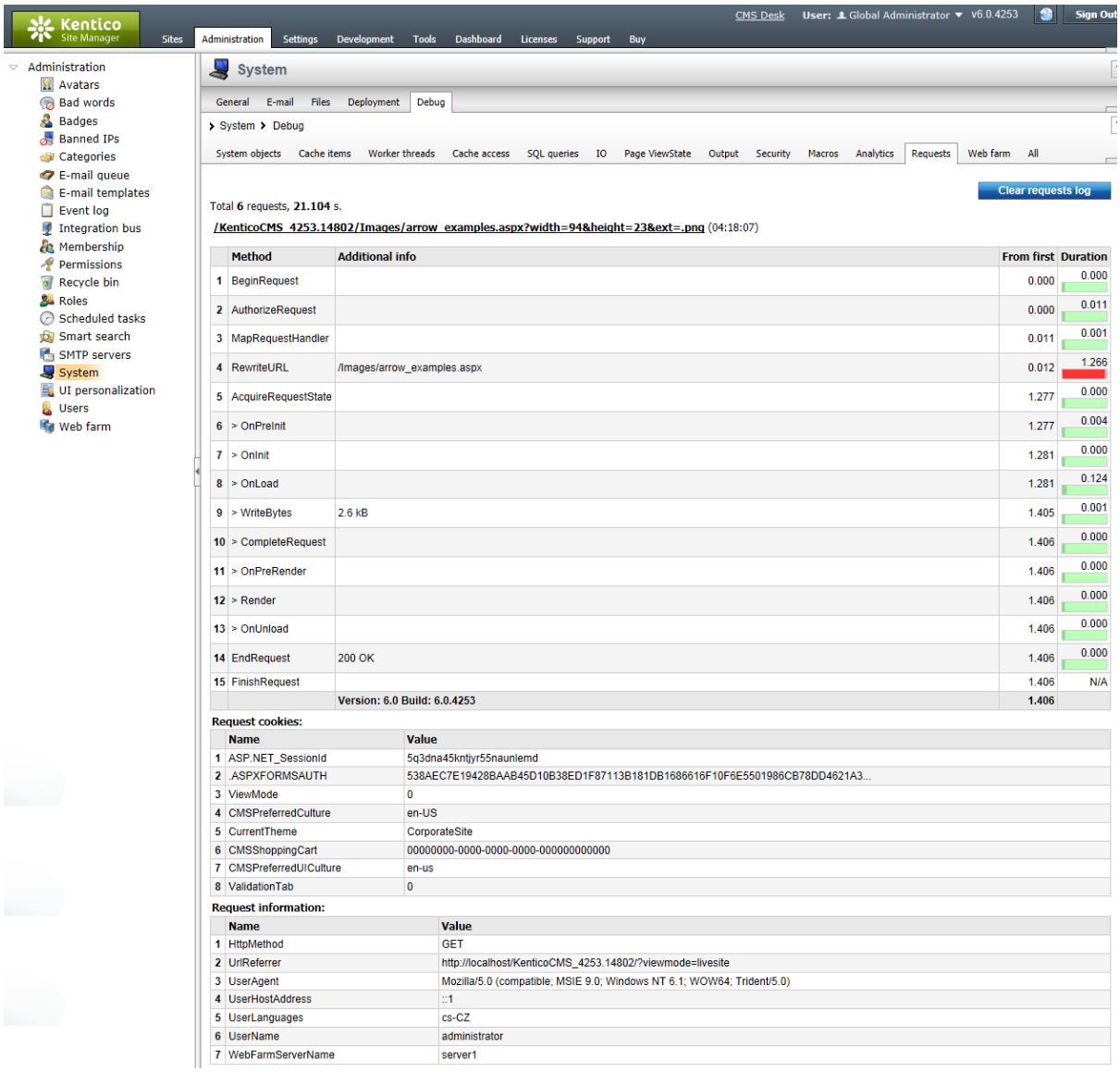
Request debugging can also be enabled using the general settings and keys.

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.

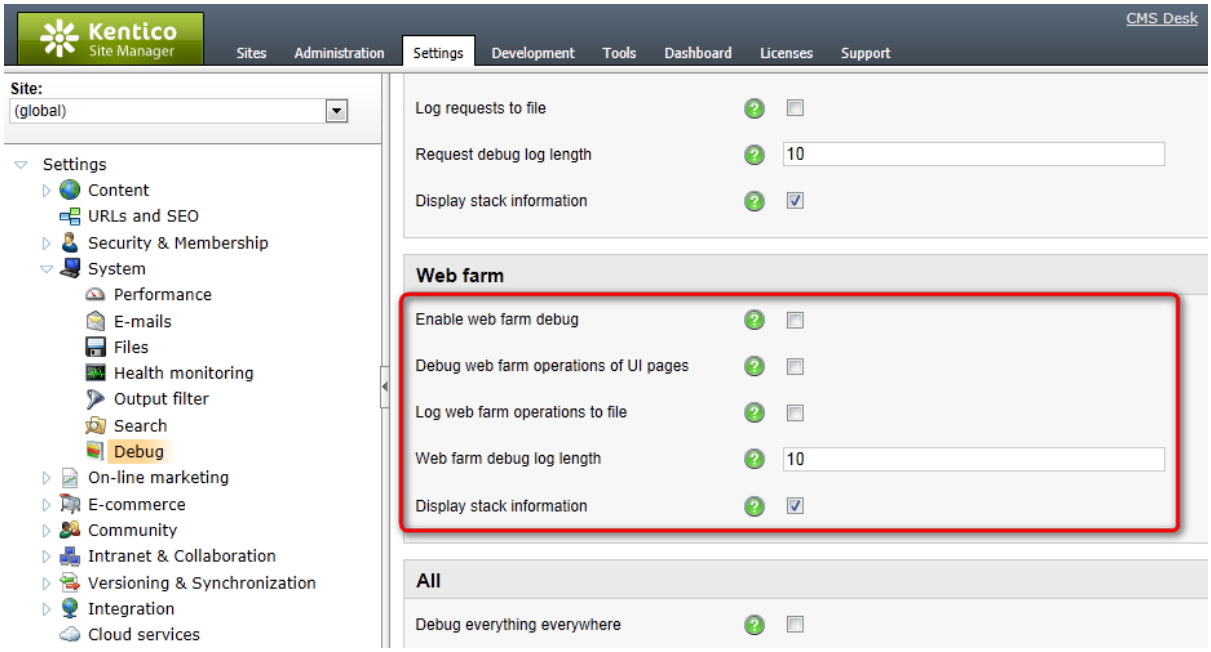
Enabling the **Show complete context** check-box displays complete context (not only the topmost item) in the **Context** column. The log can be cleared using the **Clear request log** button.



7.5.3.13 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	CMSDebugWebFarmLogLength	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	CMSDebugWebFarmStack	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*:

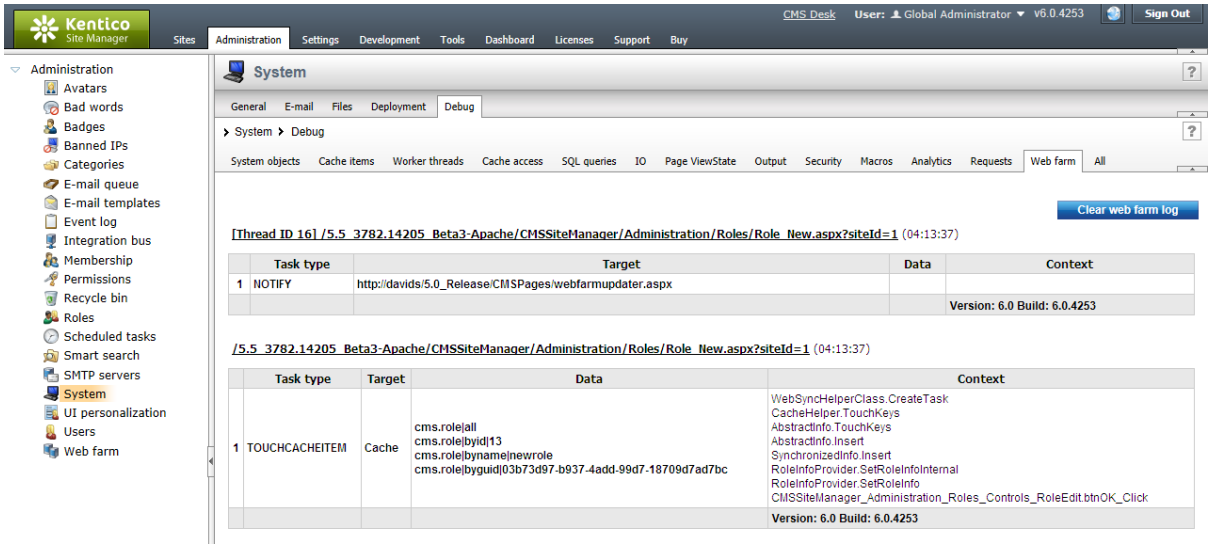
```
<add key="CMSDebugWebFarm" value="true" />
<add key="CMSDebugAllWebFarm" value="true" />
<add key="CMSLogWebFarm" value="true" />
<add key="CMSDebugWebFarmLogLength" value="20" />
<add key="CMSDebugWebFarmStack" value="true" />
```

Web farm debugging can also be enabled using the general settings and keys.

User interface

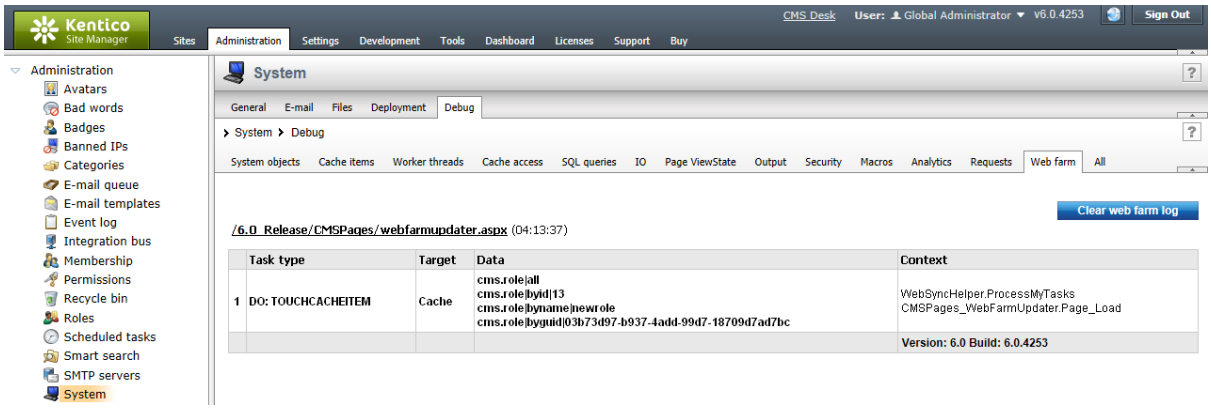
On the **Debug -> Web farm** tab, you can see which web farm synchronization tasks were logged and which servers were notified about the changes. This is particularly useful if you want to find out if the web farm works effectively and synchronizes data correctly, or generally if you want to troubleshoot 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.

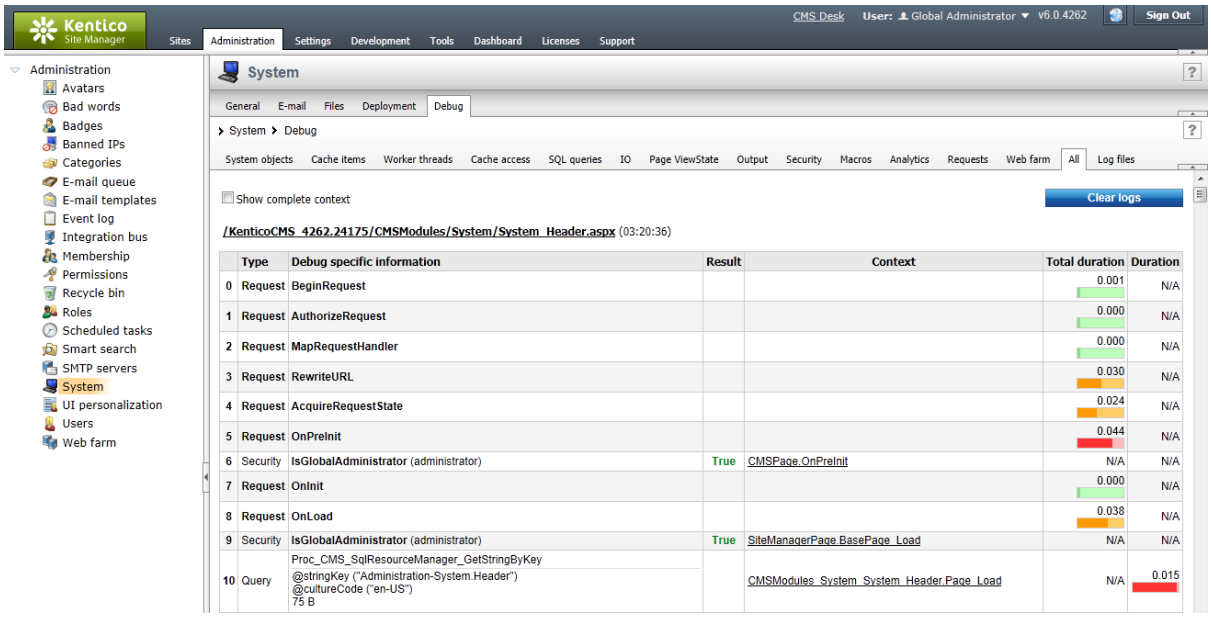


7.5.3.14 All

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.

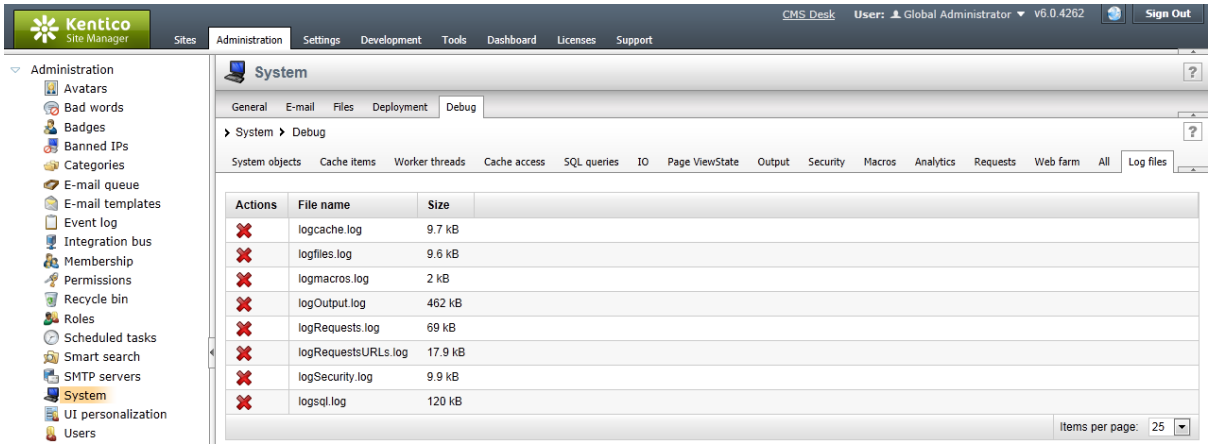


7.5.3.15 Log files

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.

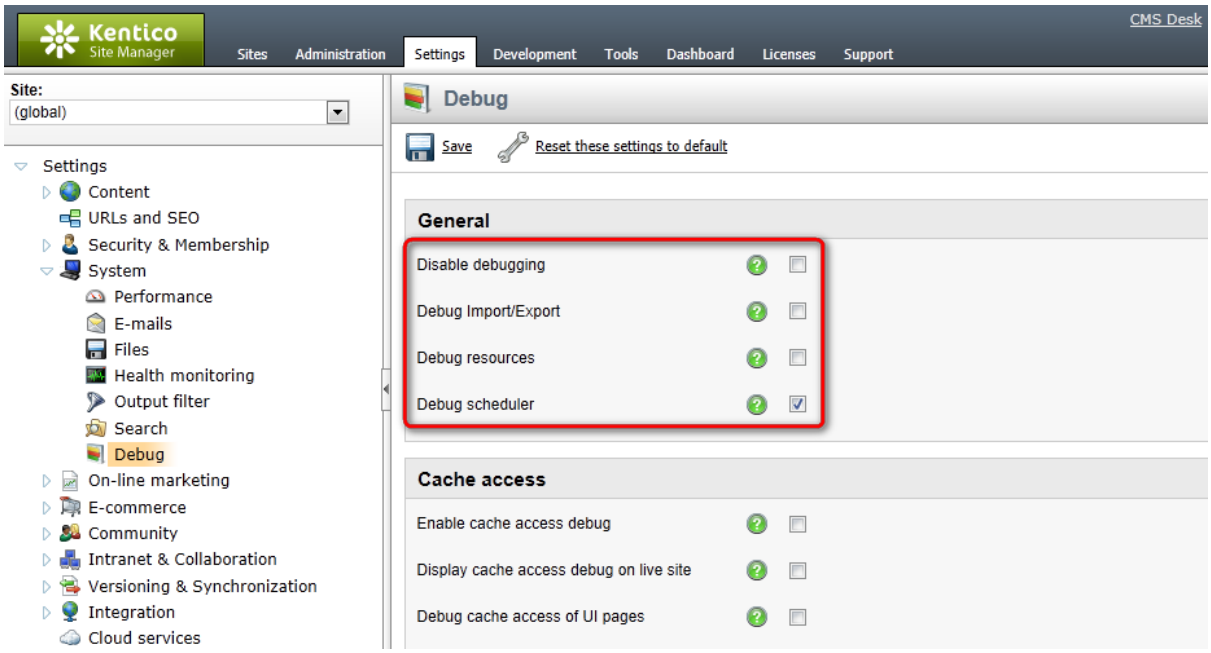


7.5.4 General settings and keys for all debugs

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:

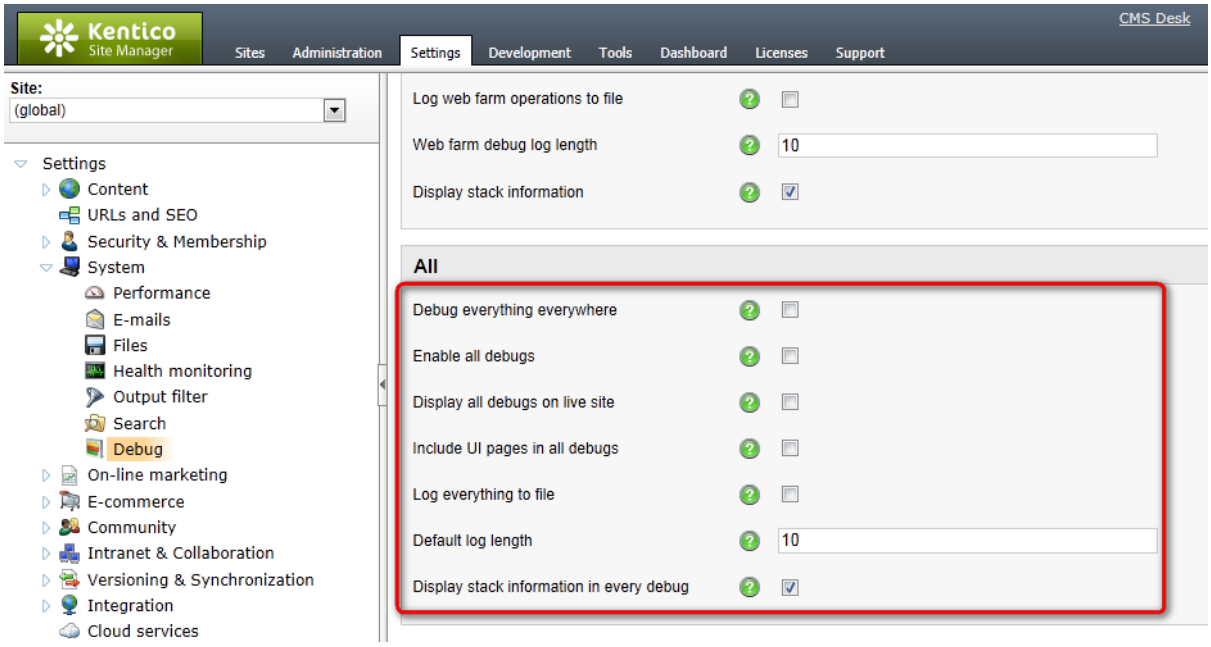
Setting	Web.config key	Description
Disable debugging	CMSDisableDebug	Globally disables all debugs, regardless of individual debug settings.
Debug Import/Export	CMSDebugImportExport	If disabled, debug information is not logged for the Import/Export user interface. To optimize system performance, it is 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**.



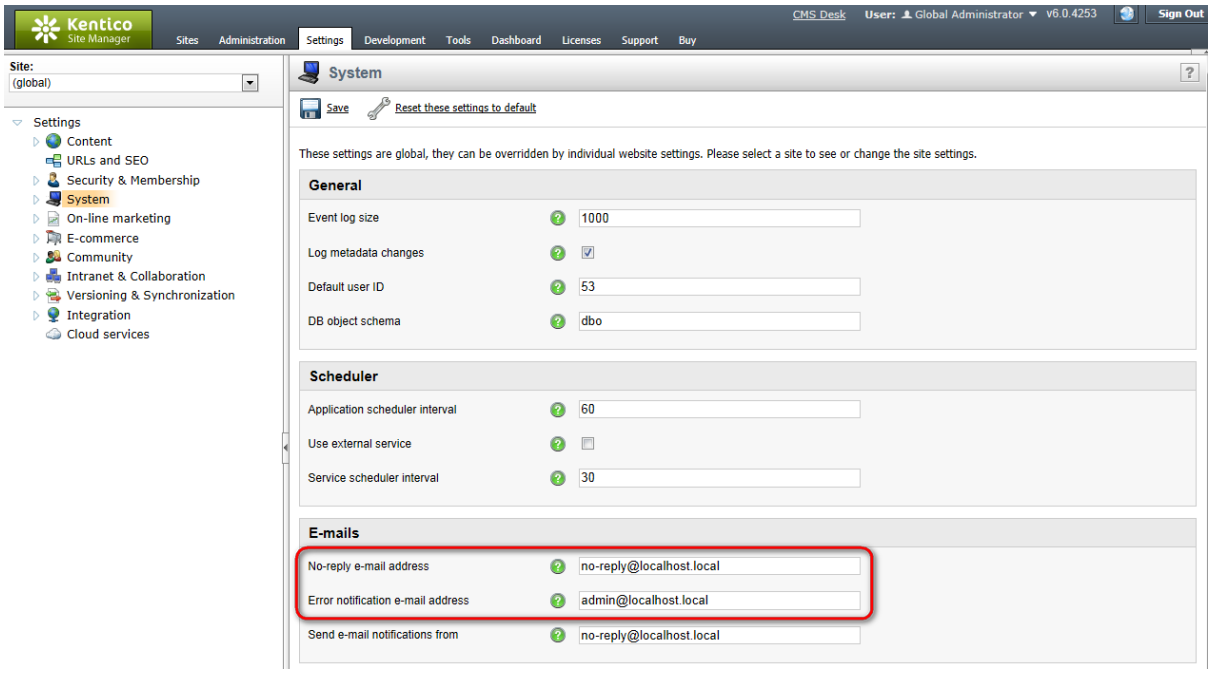
Here is a list of the keys for easy copy&paste into your *web.config*:

```
<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"/>
```

7.5.5 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.



7.6 Document types and transformations

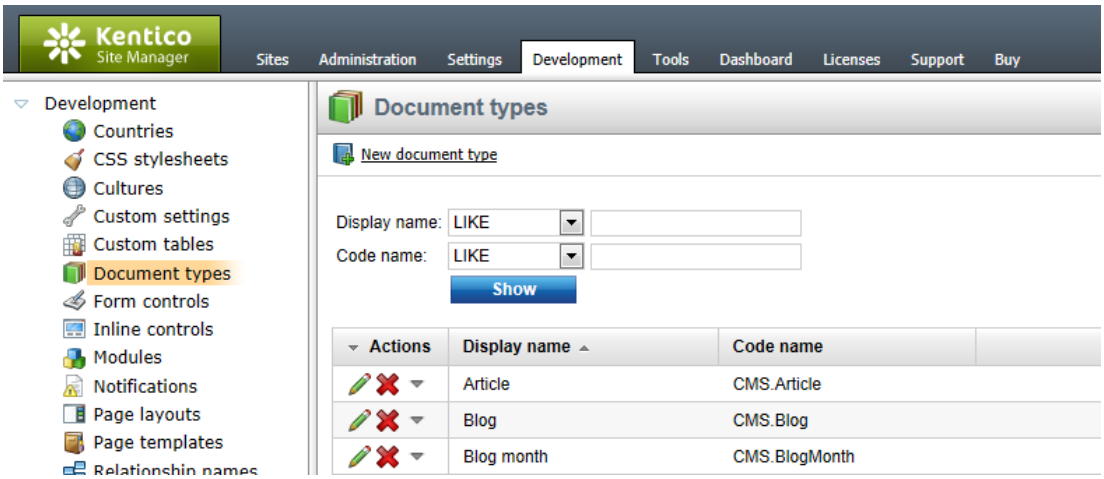
7.6.1 Overview

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.

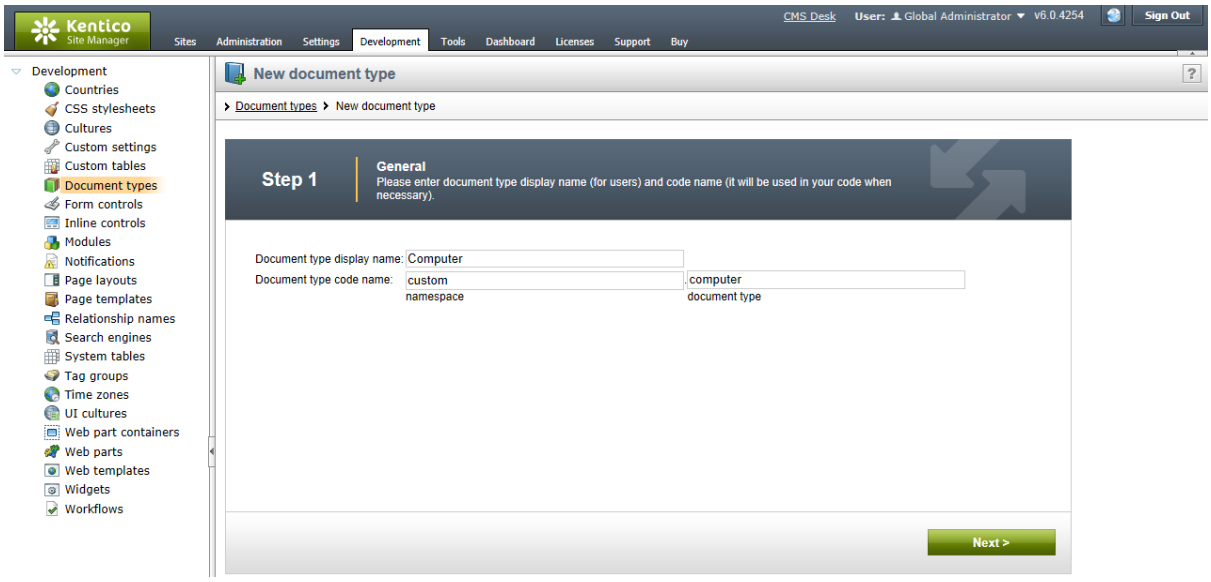
7.6.2 Defining a new document type

In this example, you will learn how to define a new custom document type.

1. Go to **Site Manager -> Development -> Document types** and click the **New document type** link. 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 the users)
- **Document type code name:** custom.computer (*custom* is your namespace to distinguish your 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.

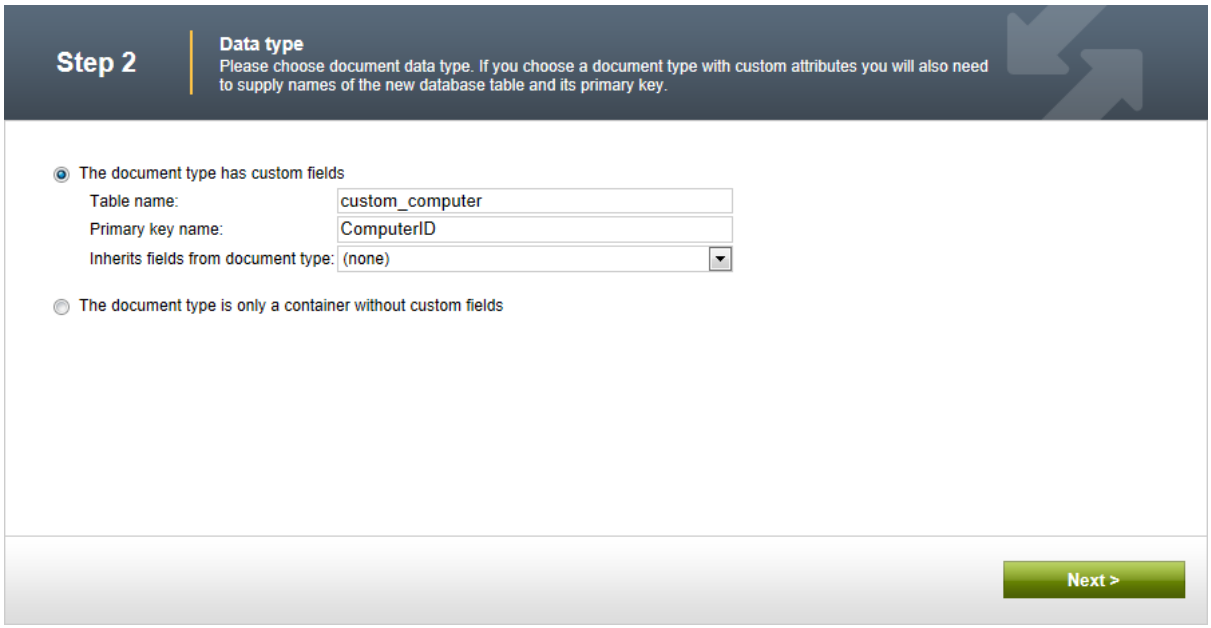
Click **Next**.



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 field** and repeat the procedure until you have all the listed fields defined.

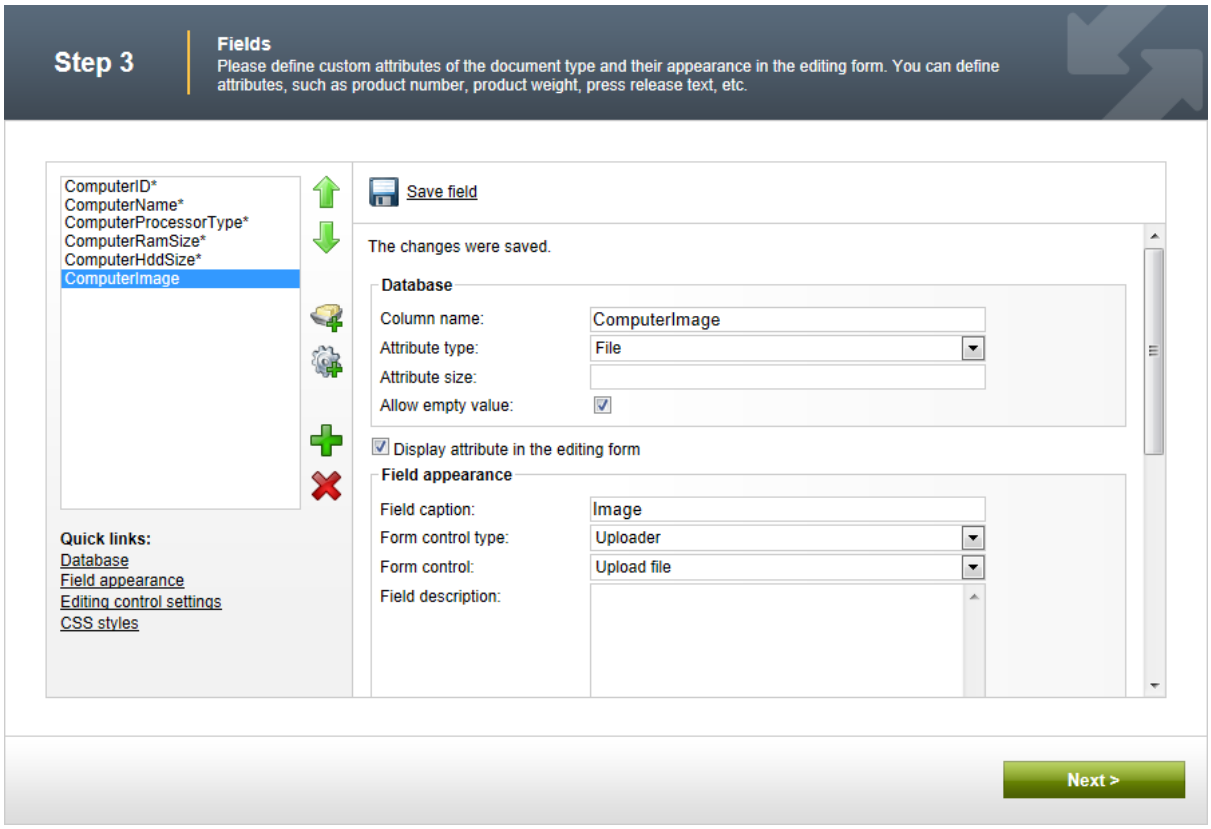
- **Column name:** ComputerName
- **Attribute type:** Text
- **Attribute size:** 200
- **Field caption:** Computer name
- **Form control type**: Input
- **Form control:** Text box

- **Column name:** ComputerProcessorType
- **Attribute type:** Text
- **Attribute size:** 200
- **Field caption:** Processor type
- **Form control type**: Input
- **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 type**: Input
- **Form control:** Text box

- **Column name:** ComputerHddSize
- **Attribute type:** Integer number
- **Field caption:** HDD (GB)
- **Form control type**: Input
- **Form control:** Text box

- **Column name:** ComputerImage
- **Attribute type:** File
- **Allow empty value:** check the box
- **Field caption:** Image
- **Form control type**: Uploader
- **Form control:** Upload file

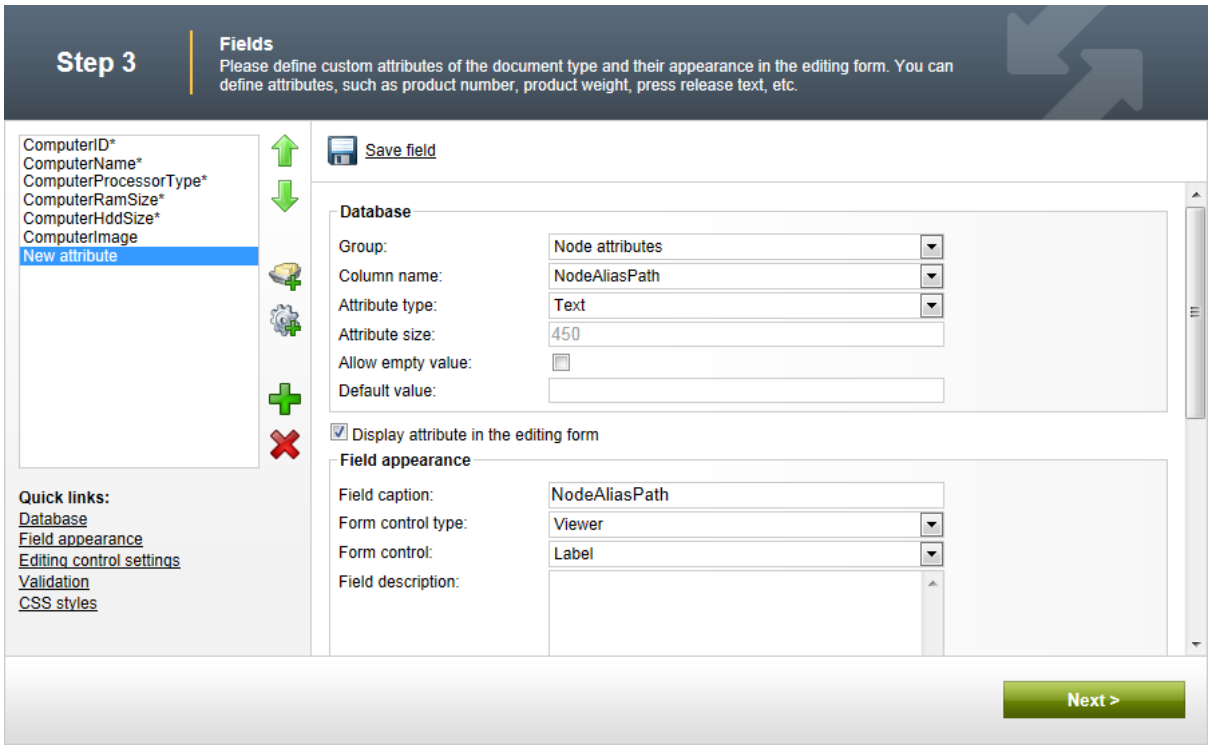


Please note that you can also define system fields that will be displayed when editing documents of this type on the **Form** tab. This can be done by clicking the **New system attribute** ( ) button. Using the **Group** drop-down list, you can then choose from the following two groups of system fields:

- **Document attributes** - offers the system fields of documents.
- **Node attributes** - offers the system fields of content tree nodes.

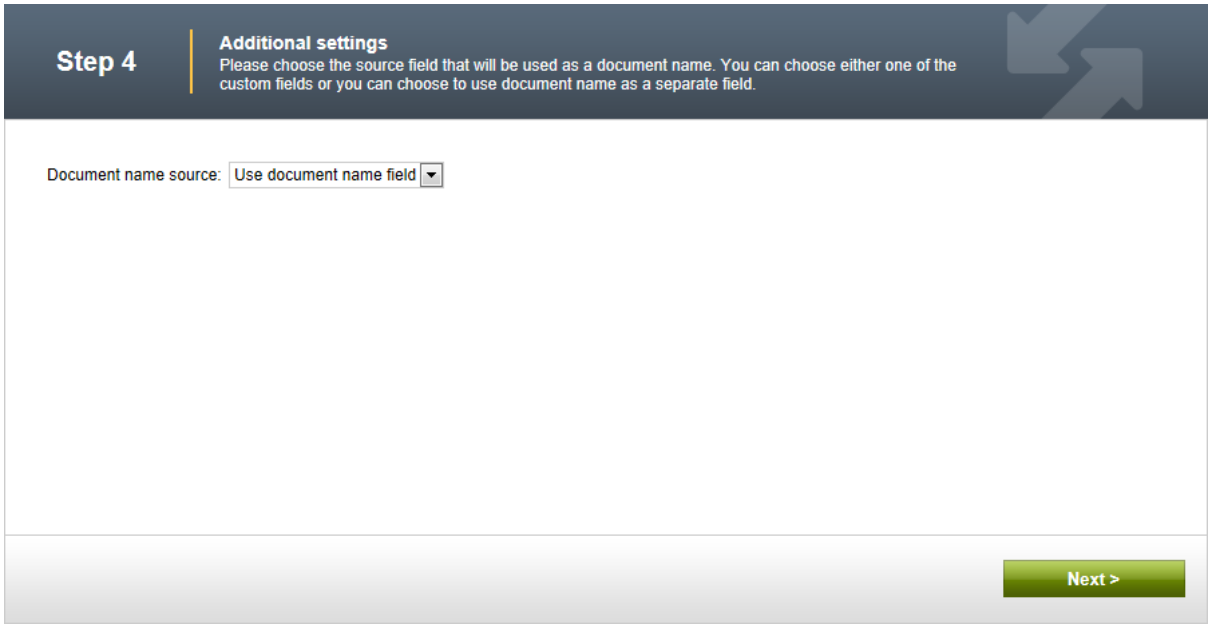
Document or node system fields will then be offered in the **Attribute name** drop-down list. If you leave the **Display attribute in the editing form** check-box checked, the field will be visible on the document **Form** tab.

Click **Next**.



4. Now you need to choose the field that will be used as document name. Choose the **Use document name field** option from the drop down list. It means that when you create a new computer document, its name will be automatically taken from the *ComputerName* value and this value will appear in site navigation and in the CMS Desk content tree.

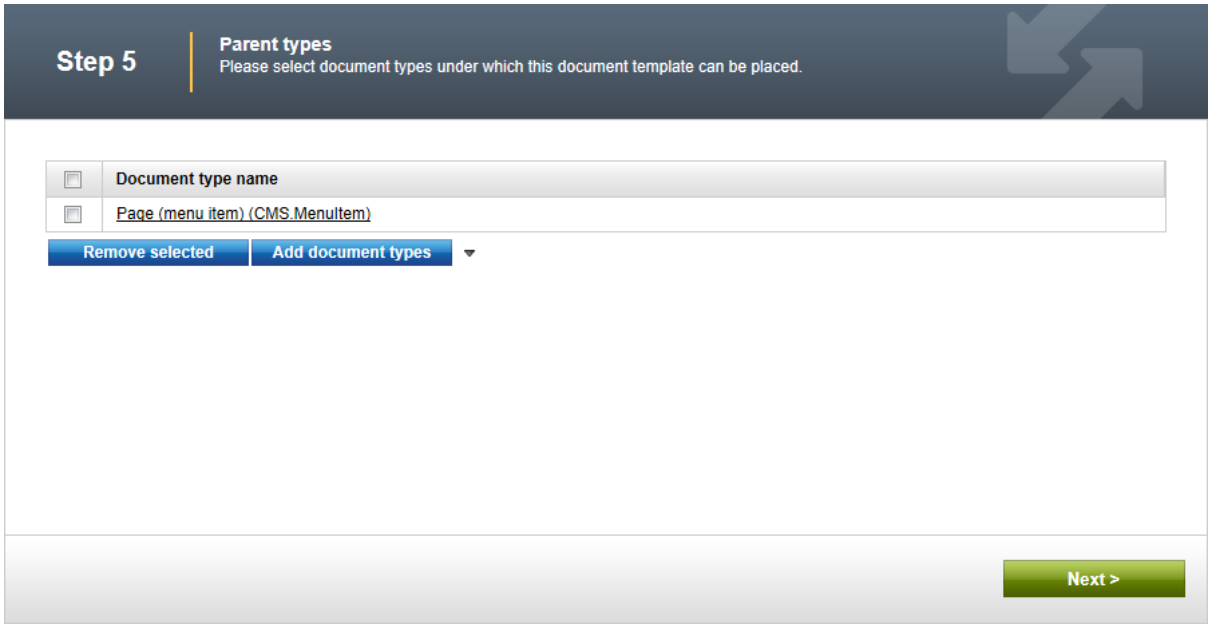
Click **Next**.



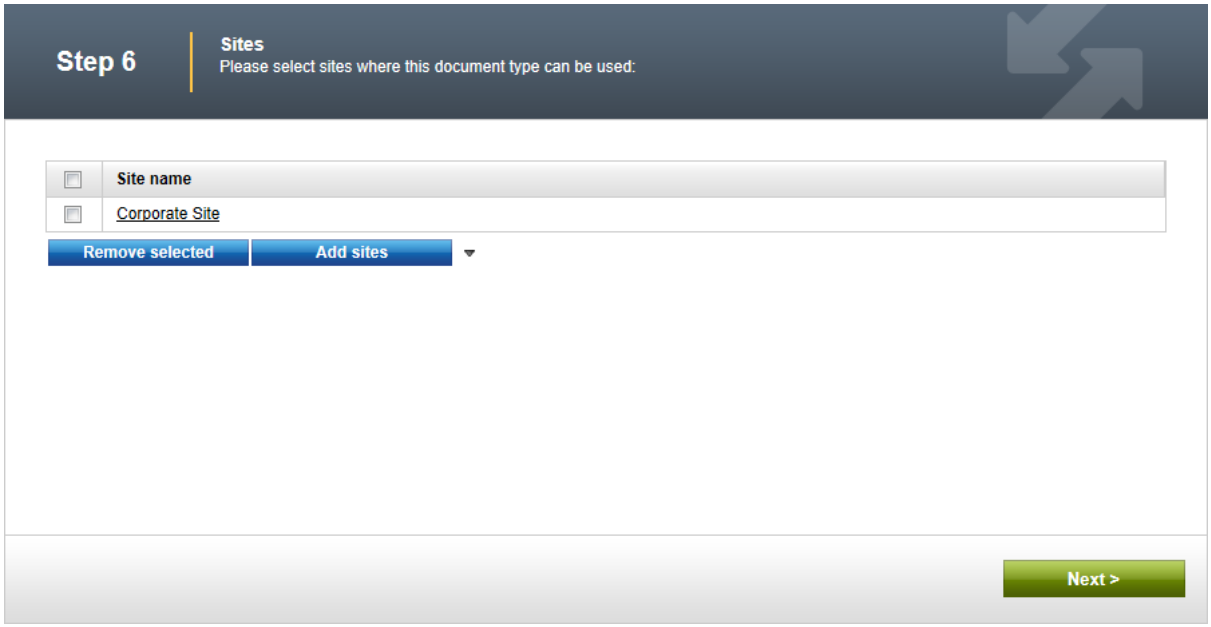
5. In Step 5, you need to select the document types under which computer documents will be displayed.

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.

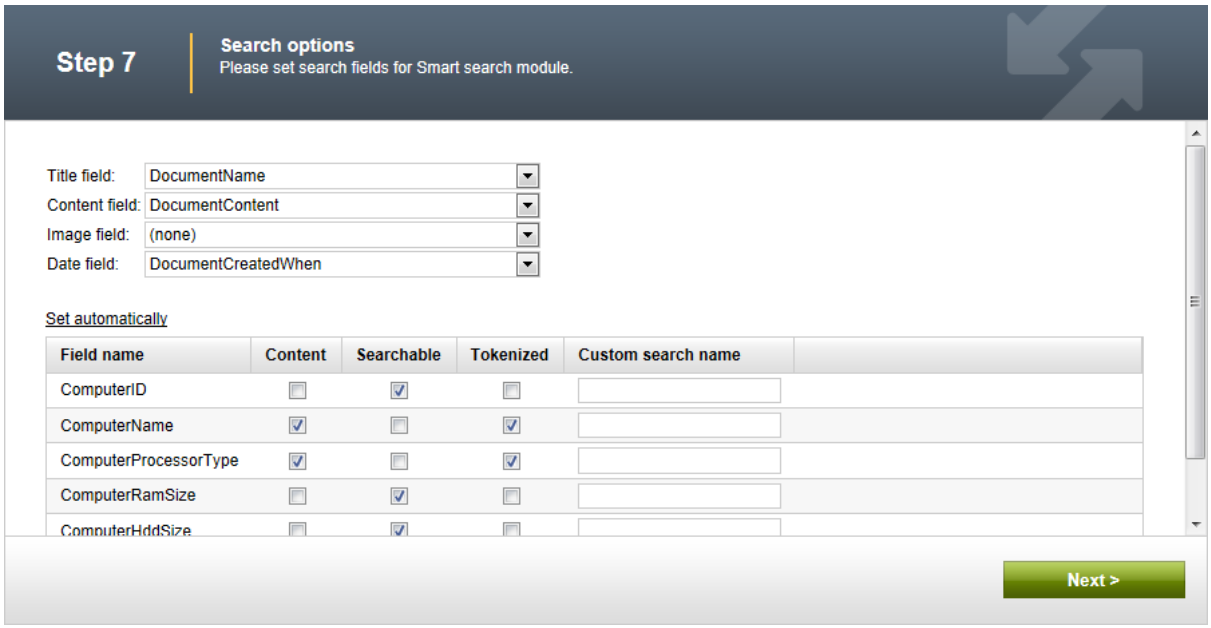
Click **Next**.



6. In Step 6, you need to choose which websites will use this document type. Check the appropriate website and click **Next**.

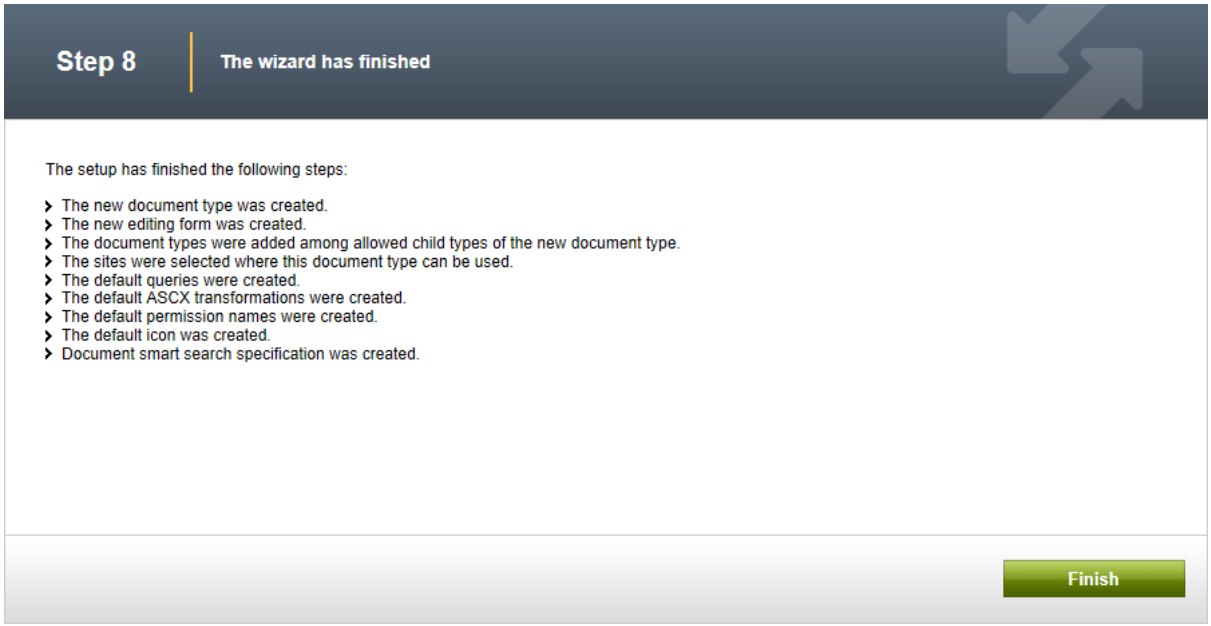


7. In Step 7, you are asked to specify how documents of this type will be indexed and displayed in the search results. For more information on these settings, please refer to this topic. Make your choice and click **Next**.



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 a default transformation.

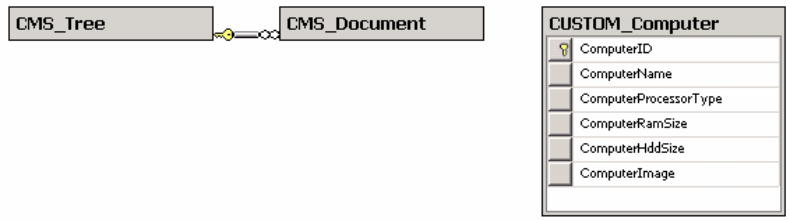
Click **Finish**. Congratulations, you have learned how to define a new document type.



How the content is stored

As you already know, the new document type *Computer* has its own database table. Each document is stored in three tables: *CMS_TREE* (tree structure), *CMS_Document*

(document properties and metadata) 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).

Changing the document type icon

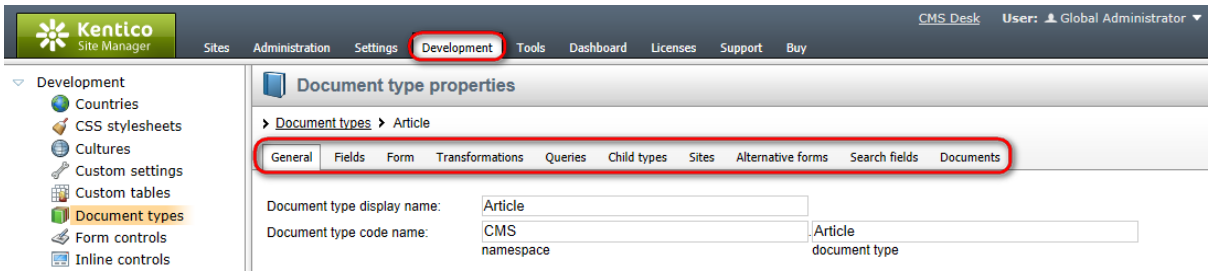
If you create a new document type, documents of this type will appear with the default document icon ( ) in the administration interface. To change the document type icon:

1. go to <*web project folder*>*App_Themes**Default**Images**DocumentTypeIcons*\
2. find your document type's icon named <*namespace*>_<*document type*>.*png*
3. open the file in your image editor and modify it, or create a new image with the same name and format and replace the original one with it

After doing so, all documents of this type should appear with the new icon. You can also change any other document types' icons this way, as all document type icons are stored in the location mentioned above.

7.6.3 Document type properties

Document type properties can be accessed in **Site Manager -> Development -> Document types -> Edit ( ) Document type**.

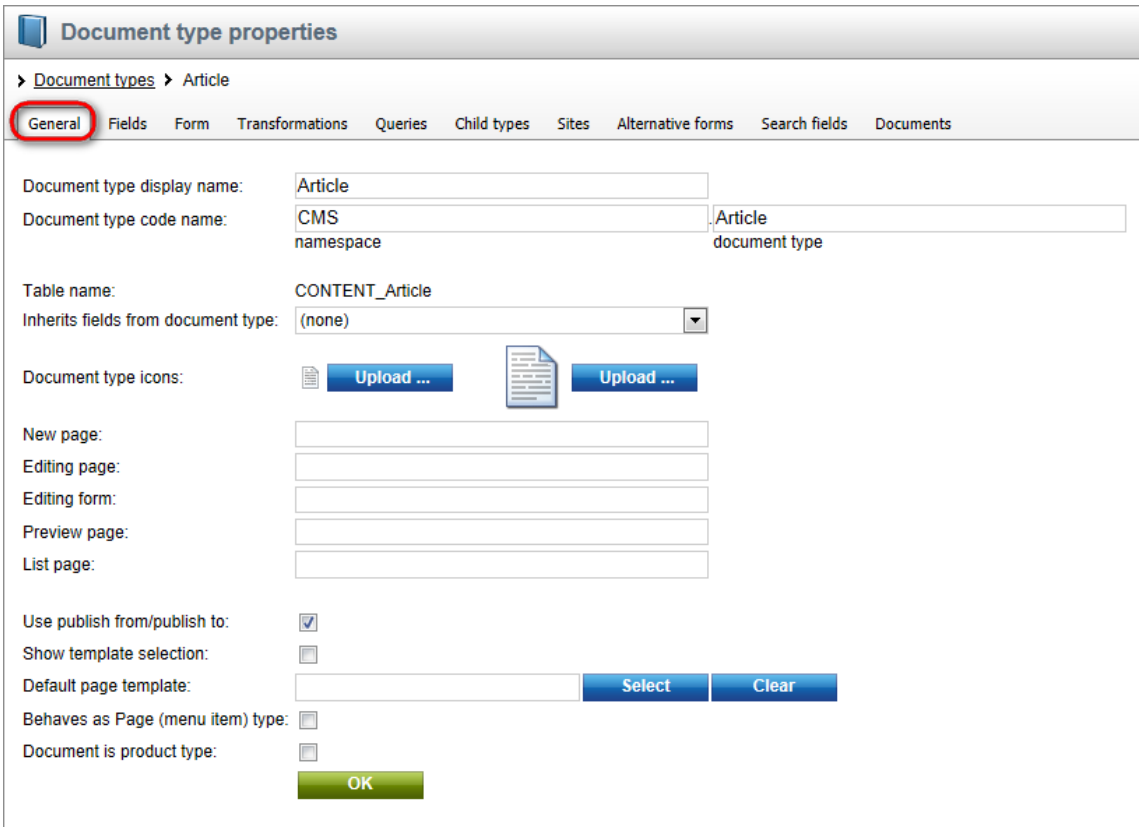


The Document types properties user interface consists of the following tabs: General, Fields, Form, Transformations, Queries, Child types, Sites, Alternative forms, Search fields and Documents.

General tab

General properties of a document type define its basic characteristics as shown below:

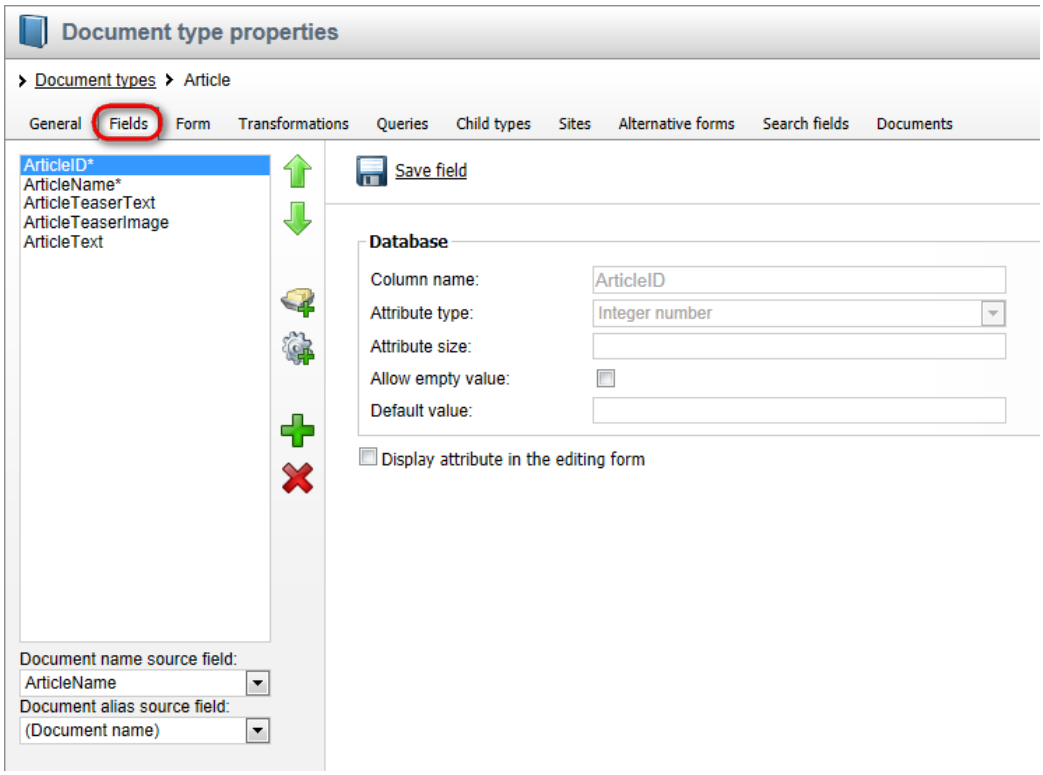
Document type display name	The name of the document type displayed to the users.
Document type code name	The name of the document type used in the code.
Table name	The name of the table in the database.
Inherits fields from document type	Indicates if fields from the selected document type should be inherited.
Document type icons	Enables uploading your own icons and associating them with the document type.
New page	URL of the page that will be used for creating new documents of this type.
Editing page	URL of the editing page that will be used when the document is displayed in the editing mode, using the Page tab.
Editing form	URL of the editing page that will be used when the document is displayed in the editing mode, using the Form tab.
Preview page	URL of the editing page that will be used when the document is displayed in the preview mode.
List page	URL of the editing page that will be used when the document is displayed in the list mode.
Use publish from/publish to	Indicates if publish from/to fields should be displayed for this document type.
Show template selection	Indicates if users must select some page template first when they create a new document of this type.
Default page template	The page template used by default when the document is created. If no page template is specified, the document inherits the parent page template.
Behaves as Page (menu item) type	Indicates if the document type has similar behavior as 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 the child documents if the path is not configured and the document does not inherit its parent template by default.
Document is product type	Indicates if documents of this type are products of the E-commerce module.



Fields tab

On the **Fields** tab, you can find the field editor that can be used to manage fields (columns) of the table. 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 the document name source field and document alias source field using the respective drop-down lists.

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.



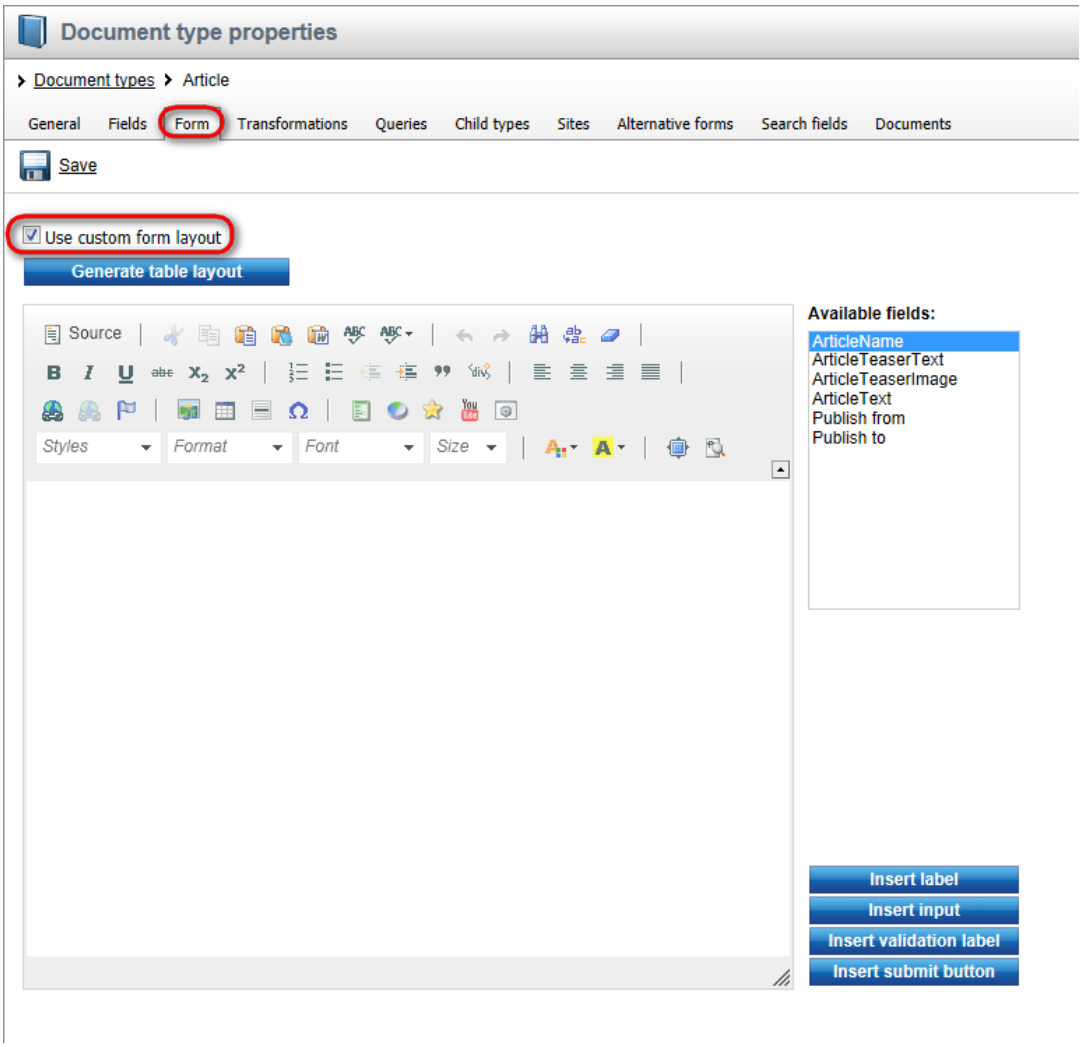
Please note

If a document type is inherited from another document type, the inherited fields are grayed 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.

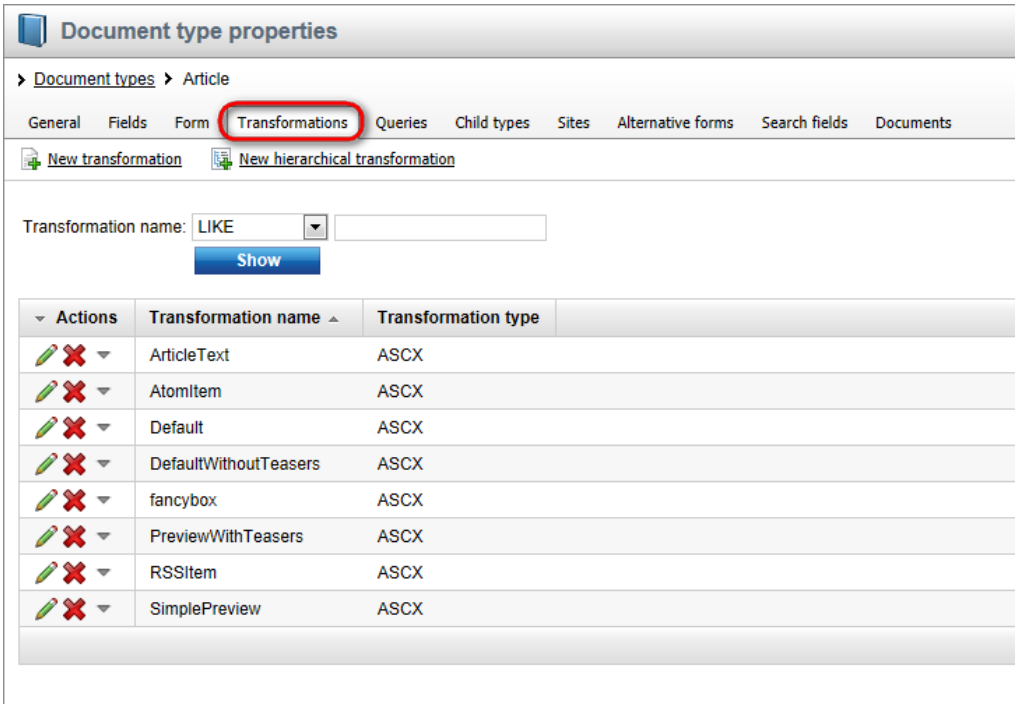
More details about the use of the **Form** tab can be found in Forms -> Defining custom form layout in the **Modules** section or in Custom fields visibility -> Use in custom form layout in the **Development -> Membership, permissions and security** section of this guide.



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 the **New transformation** or **New hierarchical transformation** links. The transformations can also be **Edited ()** or **Deleted ()** here.

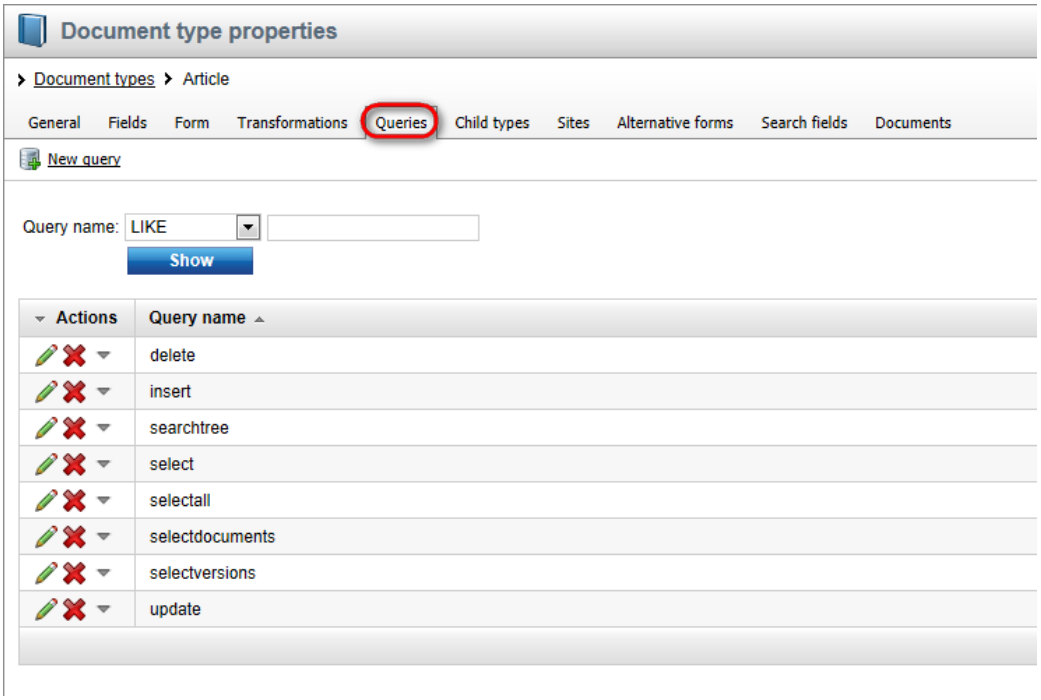
More information about transformations can be found in this chapter in the Transformations, Adding custom functions to transformations, Writing transformations and Hierarchical transformations topics.



Queries tab

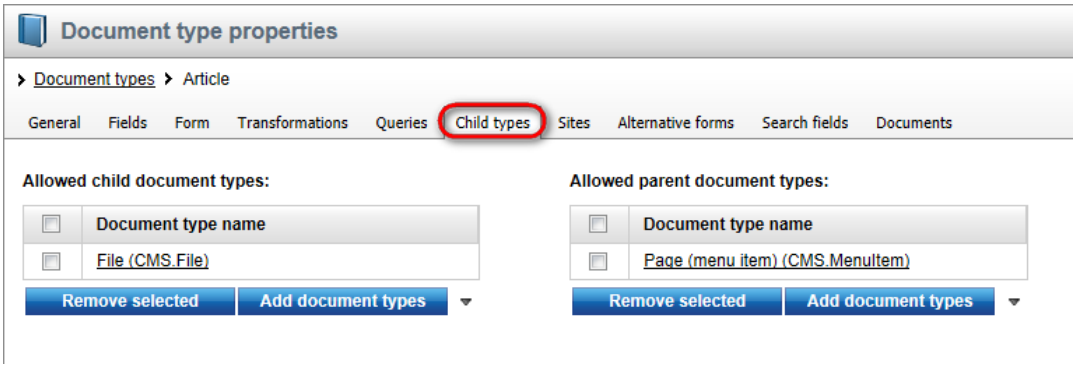
The **Queries** tab displays a list of all available SQL queries added to the particular document type. You can **Edit (**  **)** or **Delete (**  **)** items from the list. New queries can be added using the  **New query** link.

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.



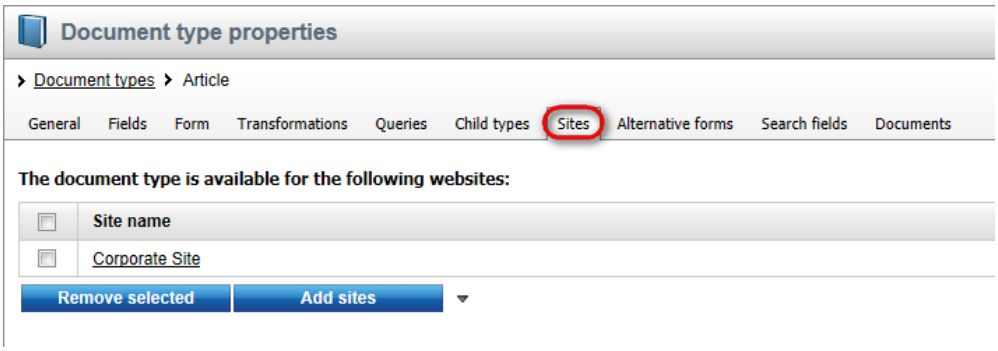
Child types tab

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

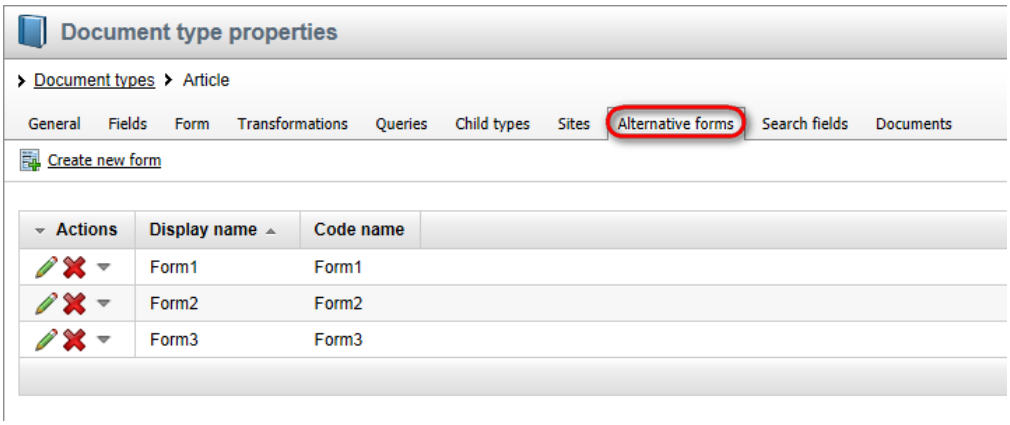
The **Sites** tab enables you to select sites where the given document type can be used. If you need to add a site, click the **Add sites** button and select it from the list of available sites. To remove a site, check the checkbox next to the site you would like to remove and click the **Remove selected** button.



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** link. You can also **Edit** (✎) or **Delete** (✖) the existing alternative forms listed in the table.

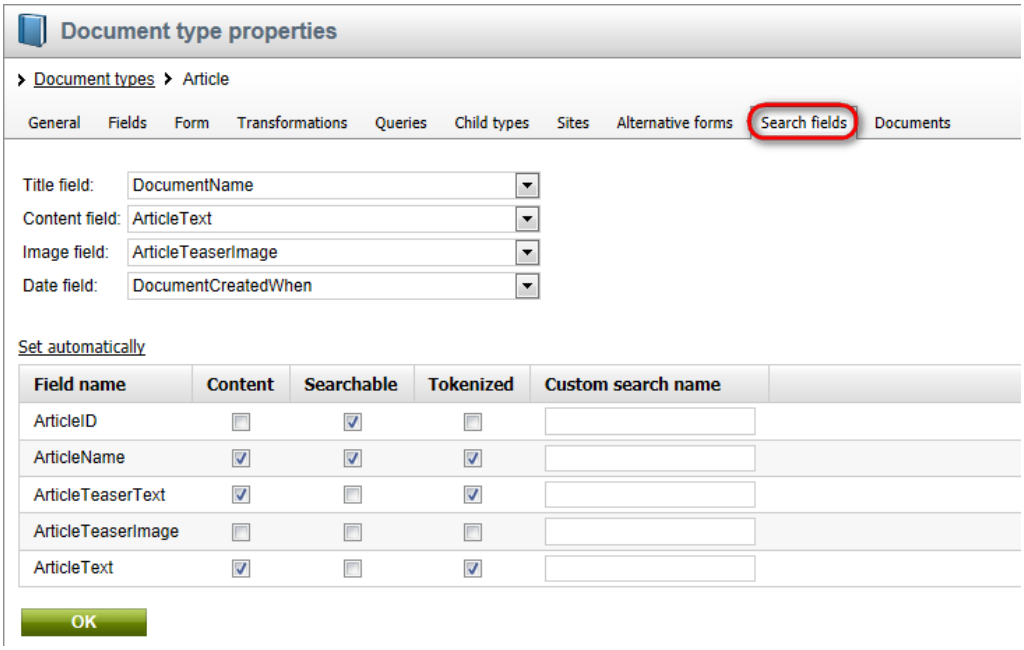
More information about alternative forms can be found in the Alternative forms chapter in the **Modules** section of this guide.



Search fields

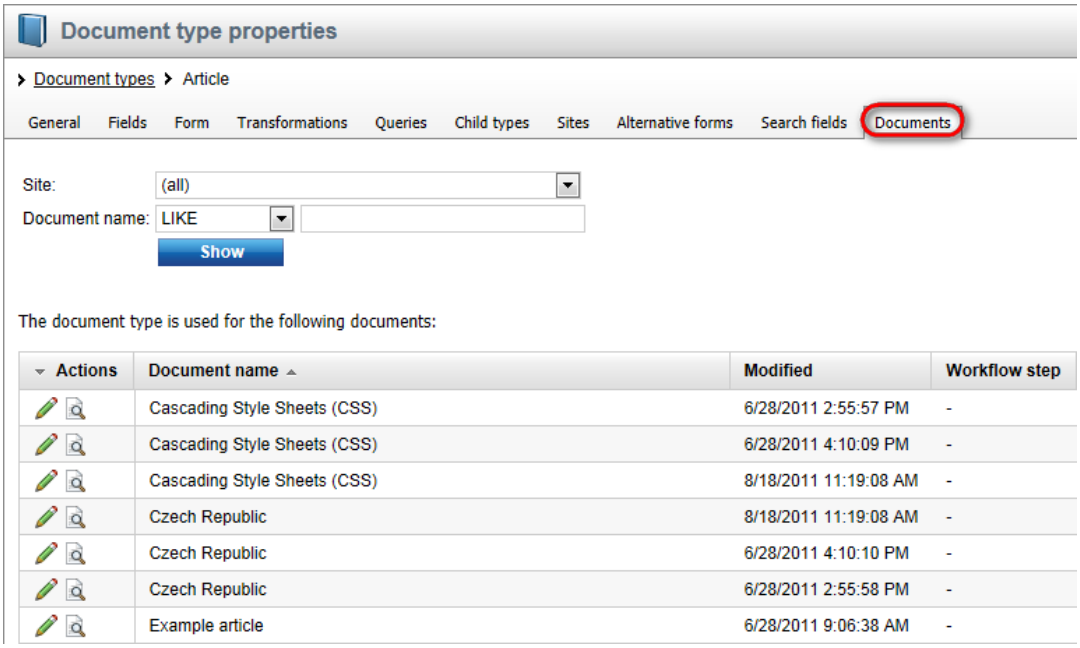
On the **Search fields** tab, you can define how data stored in the document type will be indexed by the **Smart search** module. In the top part, you can specify how documents of the given type will be displayed in search results. Lines of the table in the bottom part of the tab represent document fields 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.



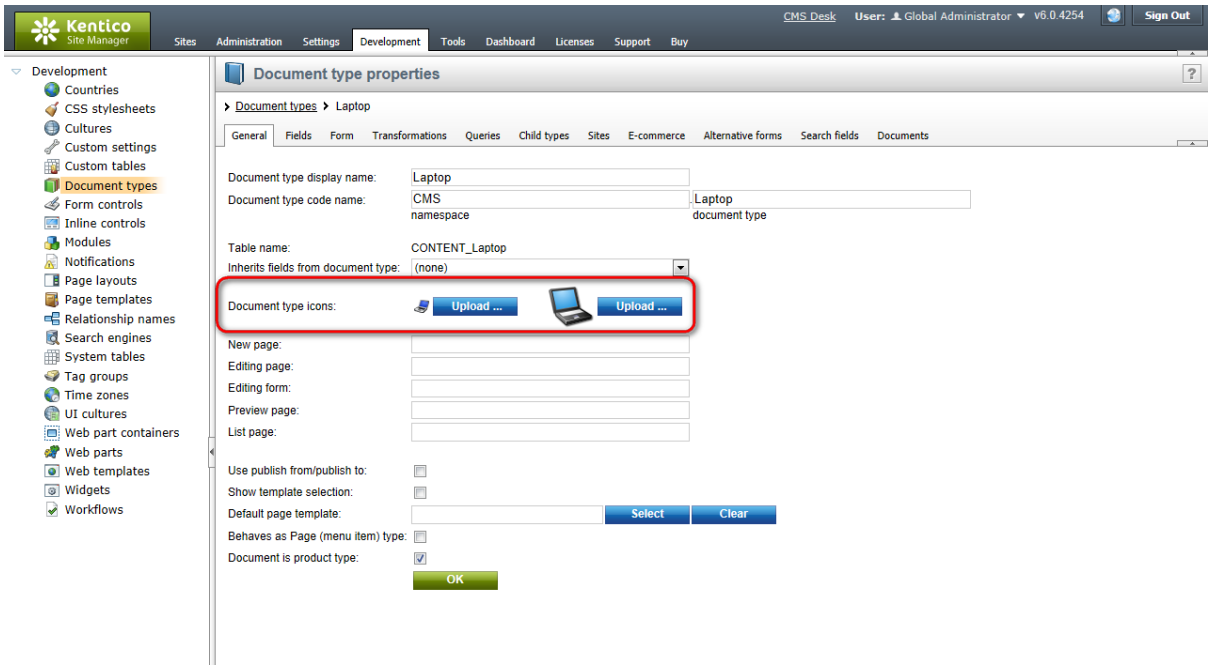
7.6.4 Uploading document type icons

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

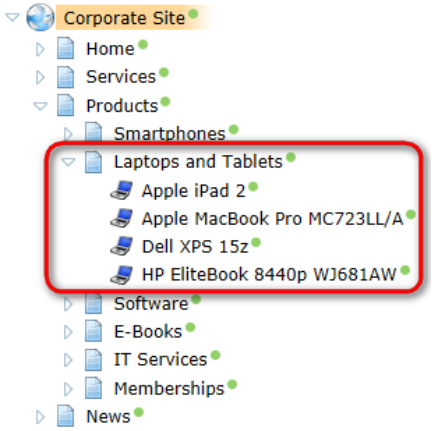
When importing and upgrading your custom document types using an export package, the associated icons are also included in this package.

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:

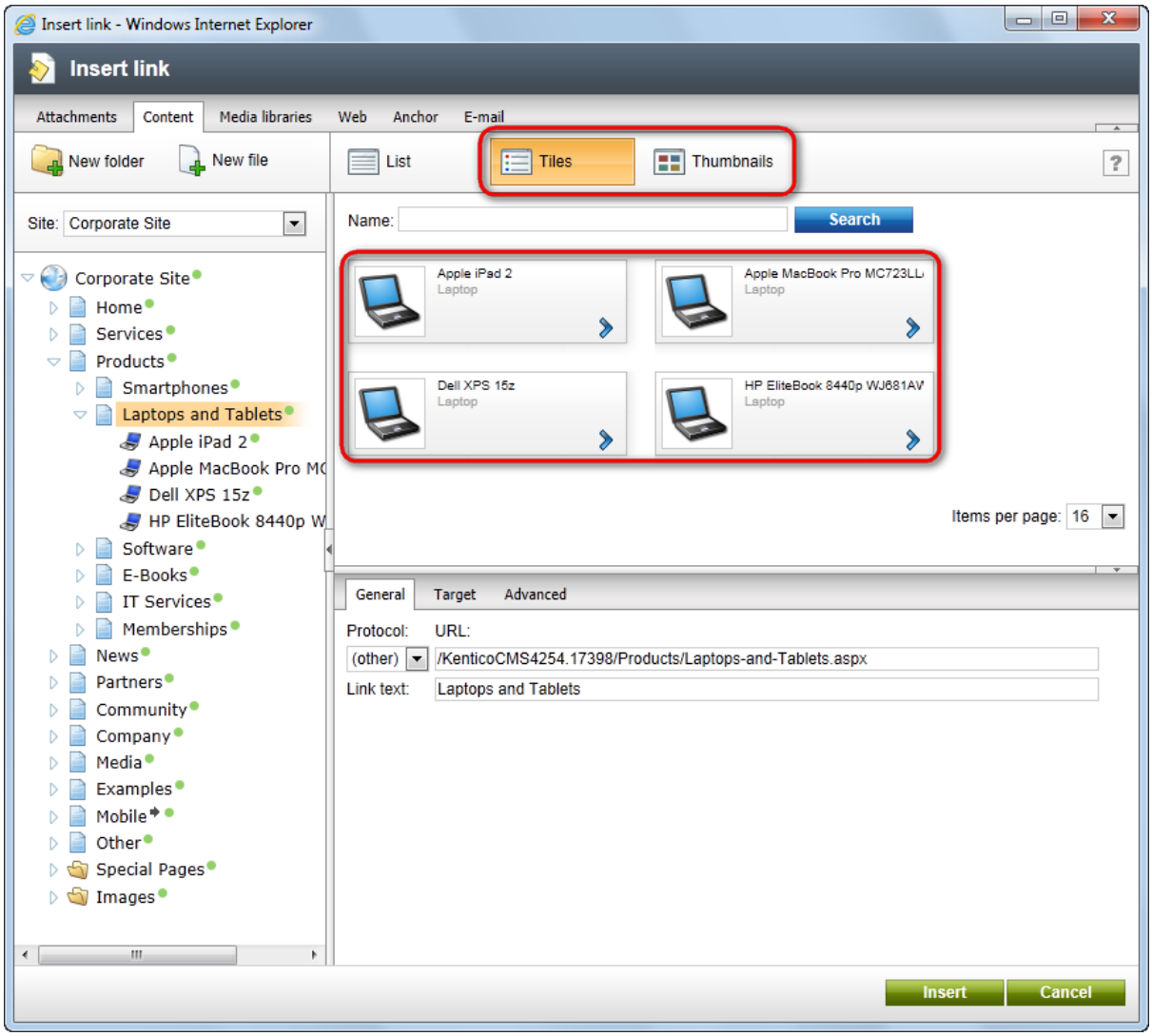


2. Click one of the **Upload** buttons and select a **.png image** from your local or network file system. Its size will be changed automatically. Click **OK**. From now on, this image will be associated with the given document type.

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 bigger 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.

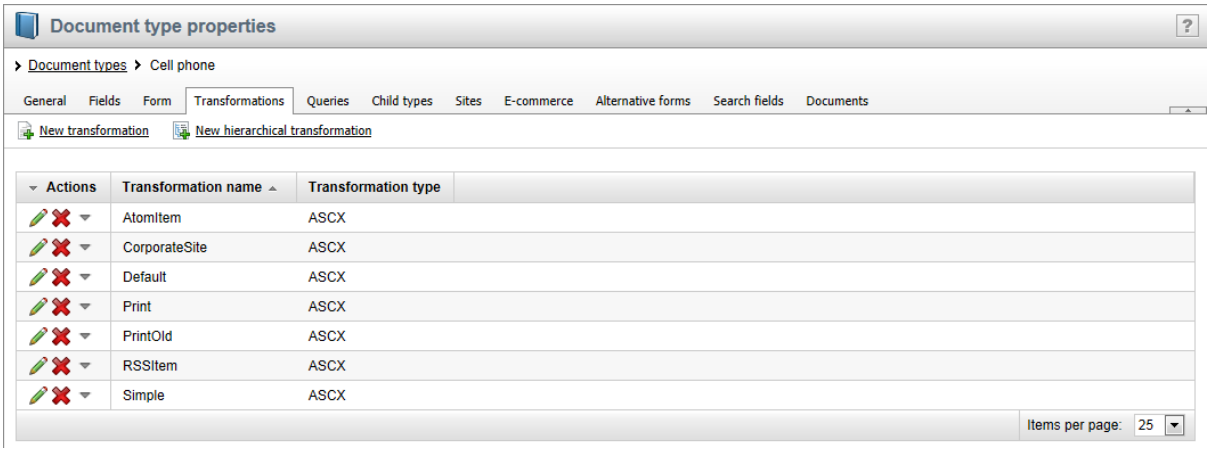
7.6.5 Transformations

7.6.5.1 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.

7.6.5.2 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 can be used to write transformations. You can choose how a transformation's code should be rendered by setting the appropriate type. The following are available:

- **ASCX** - with this option, the code of the transformation will support 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. Document fields can be accessed using expressions in format: *<%# Eval ("ColumnName") %>*

- **Text/XML** - the code will only be processed as basic HTML. This means that any ASCX markup, such as controls or inline code, will not be 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** - works the same way as the **Text/XML** option, while these transformations are used when resolving jQuery templates.

Please note that 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"><%# EcommerceFunctions.GetFormatedPrice(Eval
("SKUPrice"), Eval("SKUDepartmentID"), null, 0, true, ValidationHelper.GetInteger
(Eval("SKUSiteID"), 0) == 0) %></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>
```

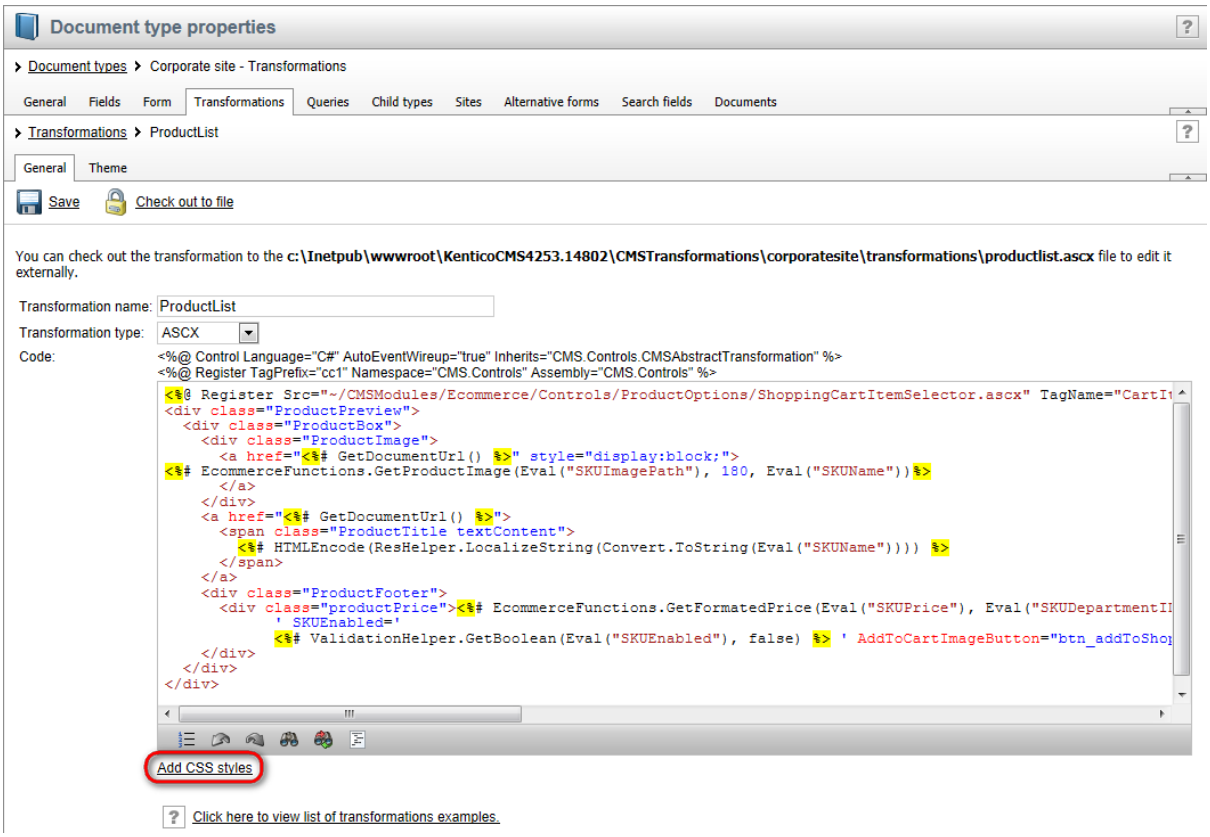
When this transformation is assigned to a listing web part or control that has products (SKUs) in its data source, the output code of individual products will contain the values returned by the methods and data binding expressions, like the following example (please note that a part of the code is omitted due to space reasons):

```
<div class="ProductPreview">
  <div class="ProductBox">
    <div class="ProductImage">
      <a href="/preview_4253.23678/Products/Laptops-and-Tablets/Apple-iPad-2.aspx"
 style="display:block;">
<img alt="Apple iPad 2" src="/preview_4253.23678/getmetafile/f1e09daf-f8c5-4252-
ba7b-222d33e01ff8/ipad2_product.aspx?maxsidesize=180" border="0" />
      </a>
    </div>
    <a href="/preview_4253.23678/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 will then look like this:

Transformation CSS styles

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, edit the transformation on its **General** tab and click the Add CSS styles link below the code editor. The **CSS styles** field will be displayed, where you can add any required CSS classes.



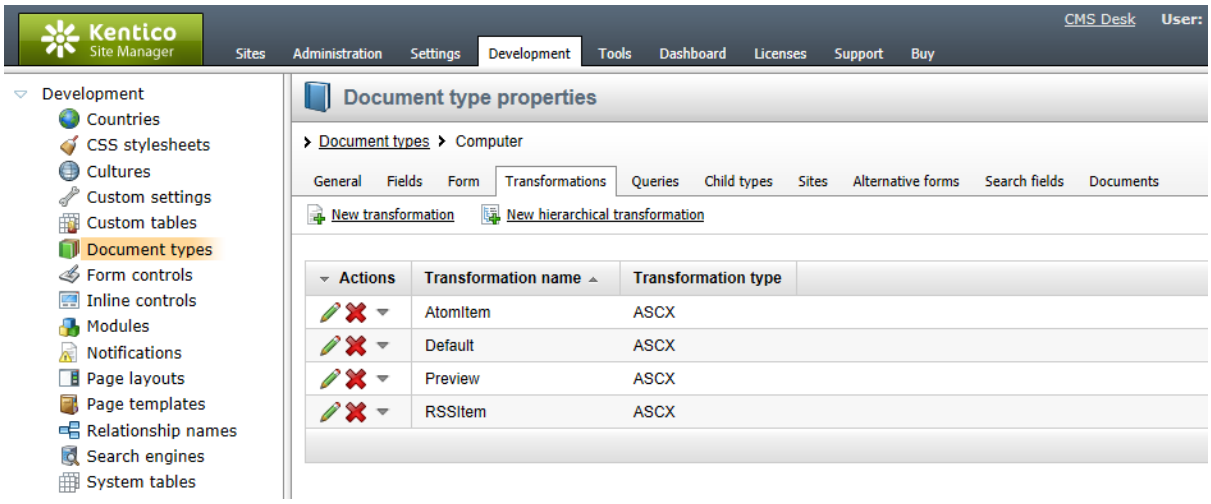
If the styles require any additional files (such as images), you can add them on the **Theme** tab.

For more information about page component CSS styles, please see the Development -> CSS stylesheets and design -> CSS for page components topic.

7.6.5.3 Writing transformations

In this topic, you will learn how to create transformations for the Computer document type created in the Defining a new document type topic.

In the **Computer** document type properties dialog, click the **Transformations** tab:



As you can see, the wizard has created some default transformation. You will use it for the detail view. Click **Edit** (🖉) and enter the following code:

```
<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>
```

Click **OK**. As you can see, the transformation code is the standard **ItemTemplate** template that you may already know from the ASP.NET **Repeater** and **DataList** controls. It combines HTML code with ASP.NET commands and data binding expressions. You can use several built-in functions such as **GetImage** that simplify some tasks. You can find the list of all functions immediately under the transformation code.

Now you will create a transformation for the list of computers. Go back to the transformations list and click the **New transformation** link. Enter the following values:

- **Transformation name:** preview
- **Transformation type:** ASCX (it is also possible to use a text-based transformation or XSLT, but we will not do so in this example)

Enter the following transformation code:

```
<div style="text-align:center;border: 1px solid #CCCCCC">
<h2><a href="<%# GetDocumentUrl() %>"><%# Eval("ComputerName") %></a><h2>
<img src="<%# GetFileUrl("ComputerImage") %>?maxsidesize=120" />
</div>
```

Click **OK**.

Please note how the link to the document is created:

```
<a href="<%# GetDocumentUrl() %>"><%# Eval("ComputerName") %></a>
```

It consists of standard HTML tags for links and it inserts the URL and link text dynamically.

Similarly, you can create an image tag with parameter that ensures automatic resize of the longest side to 120 pixels on the server side:

```
<img src="<%# GetFileUrl("ComputerImage") %>?maxsidesize=120" />
```

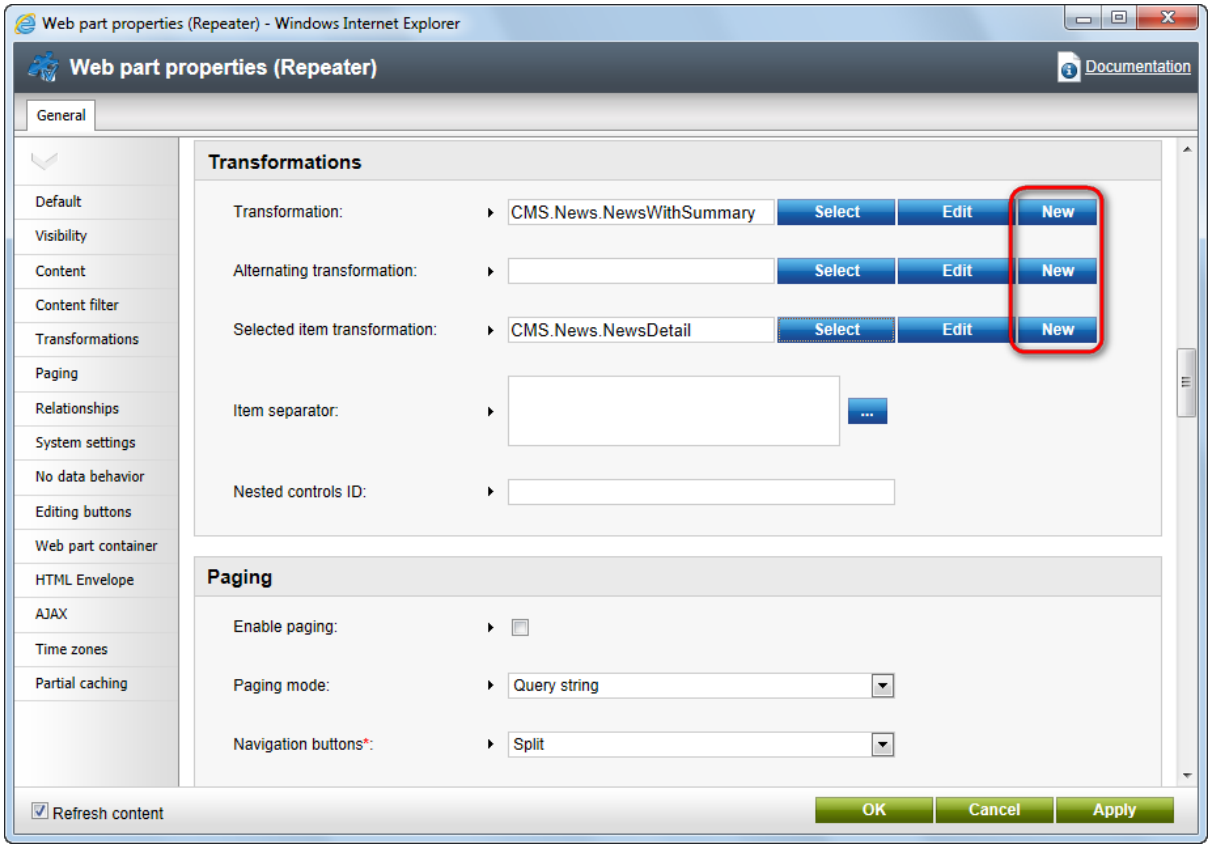
You have learned how to write transformations for displaying the content of structured documents.

Please note: Should you want to use the XSLT transformation, it can be used for the **XSLT viewer (CMSViewer)** web part. Otherwise it won't work.

Creating transformations directly in web part properties

Transformations can also be created directly in CMS Desk, in the web part properties dialog. Wherever

there is a web part property for selecting a transformation, the **New** button is present next to it. By clicking the button, you open a new dialog where you can create a new transformation. The dialog lets you create transformations not only for the document type currently selected in the web part properties, but also for any other document type present in the system.



Transformations for multilingual websites

In some cases, you may need to display different texts in transformations, based on the currently selected language. If you are using the built-in multilingual support, you can achieve this by creating another transformation with its name ending with culture code.

Example:
English (default language) transformation code name: *cms.news.detail*
French transformation code name: *cms.news.detail_fr-fr*

When you switch the content language to French, the French transformation will automatically be used in this case.

7.6.5.4 Hierarchical transformations

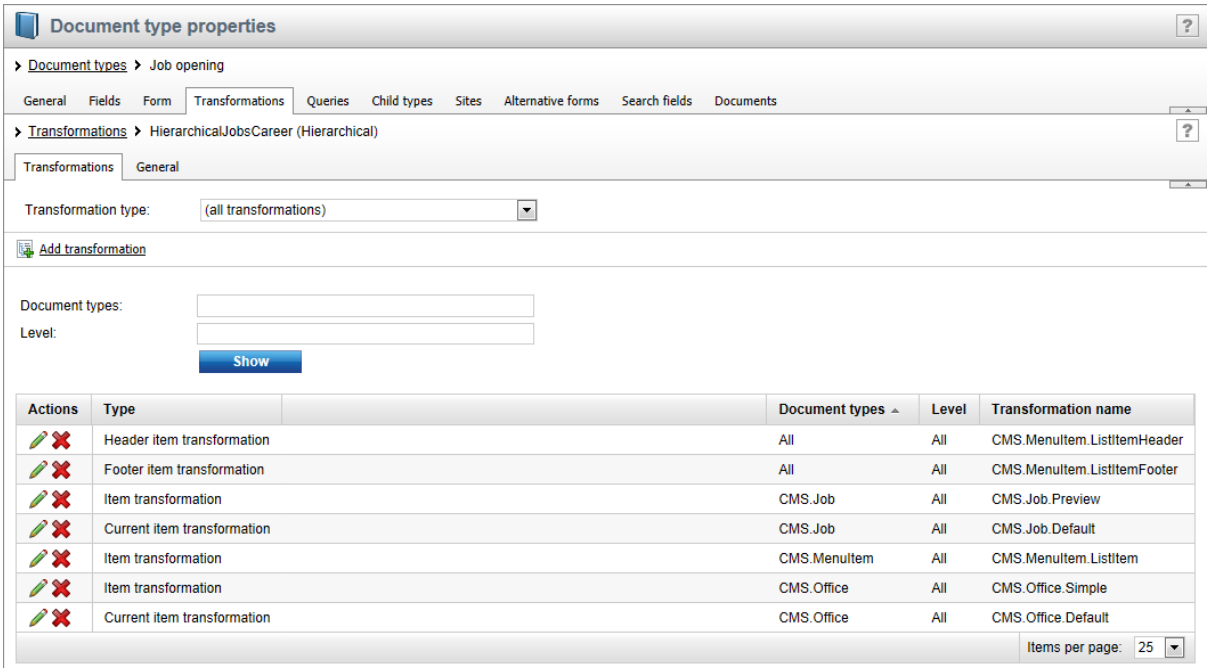
Hierarchical transformations are special types of transformation that can be used to display documents (and other types of appropriately organized data) in a hierarchical structure. These transformations can affect multiple document types and may be customized differently for every level of the hierarchy (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

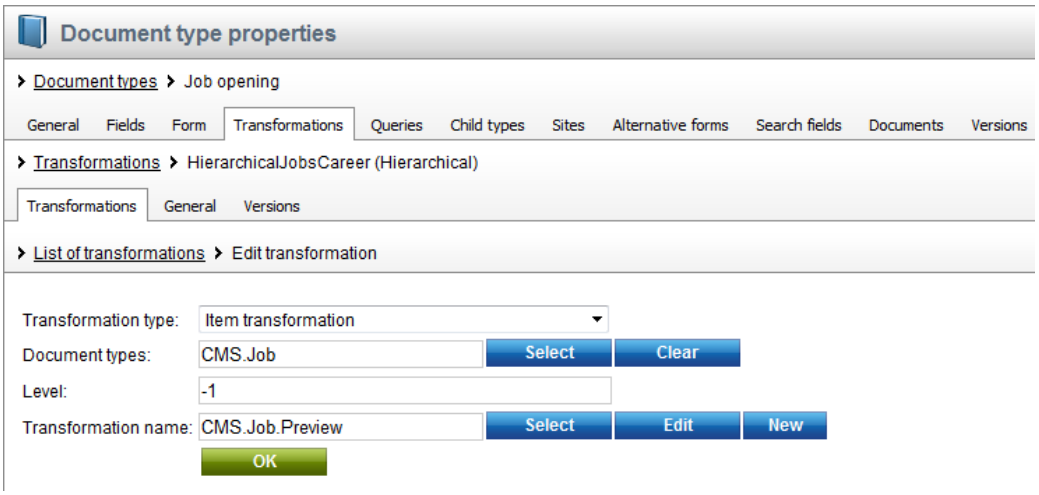
As well as the following Kentico CMS controls:

- 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.

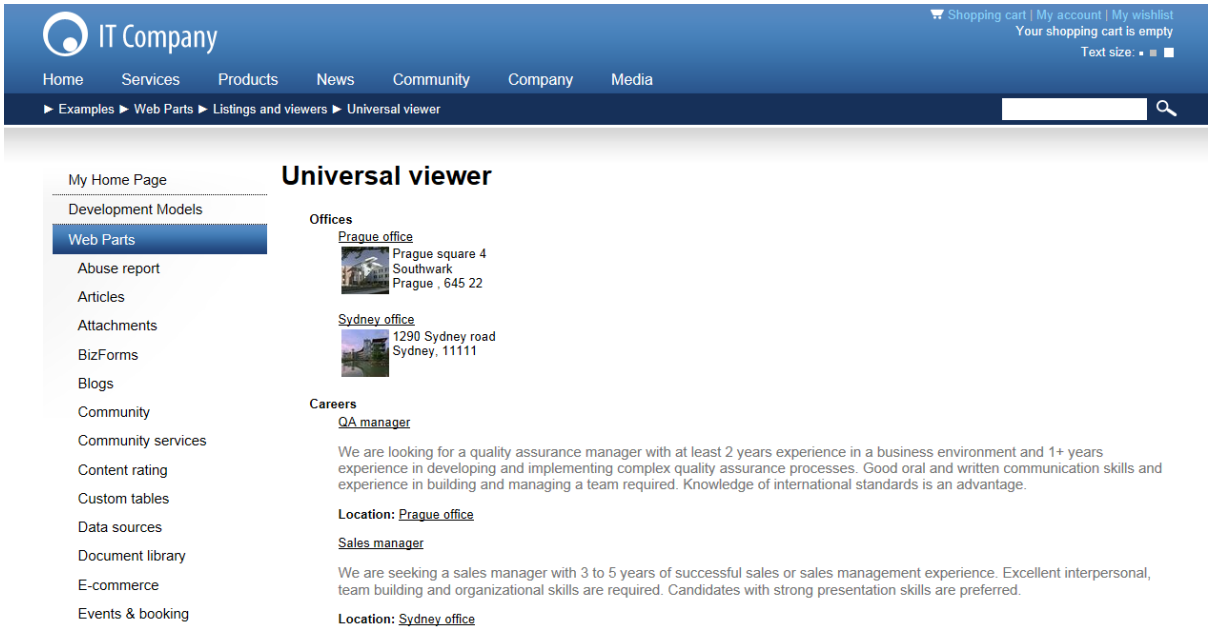
Breaking level inheritance

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

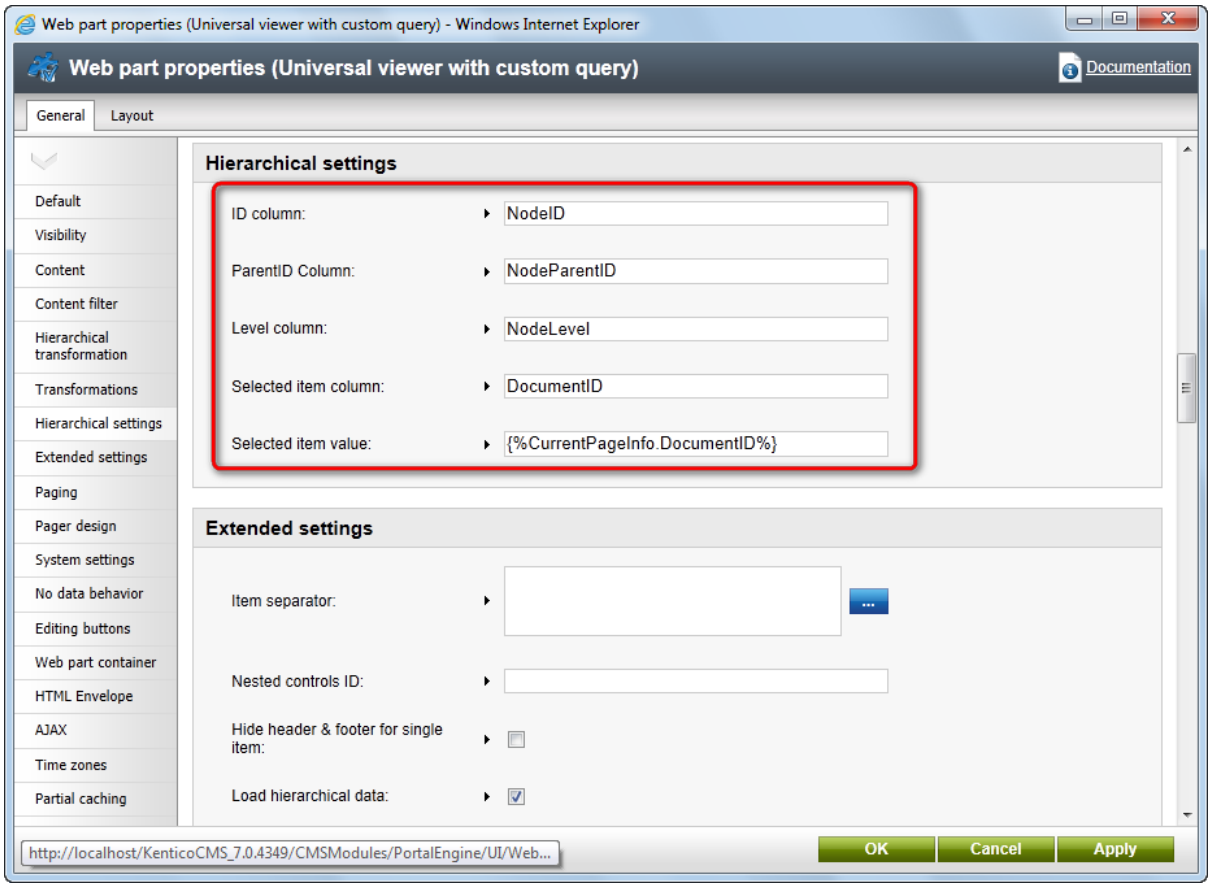
available for all supported web parts and controls. Please note that paging is not possible in this scenario, because the separation of data into pages would break the hierarchy.

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.

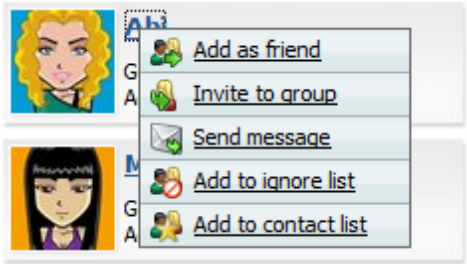
If you also wish to set up special behavior for the displayed items that reflects some type of selection made by the current user, you can enter the name of a column into the **Selected item column** property (the values of this column must be unique for every record to ensure correct functionality). 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.



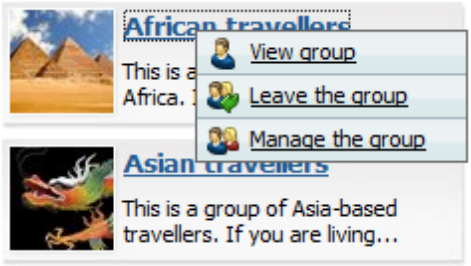
7.6.5.5 Context menus in transformations

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.

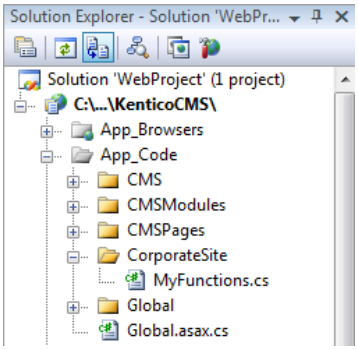
You can also develop your custom controls for this purpose. In this case, you need to include the **MenuControlPath** parameter in the *cms:usermenucontainer* or *cms:groupmenucontainer* controls in the transformation and set its value to the path to your control:

```
<cms:groupmenucontainer runat="server" ID="groupMenuElem" MenuID
="groupContextMenu" Parameter='<%# Eval("GroupID").ToString() %>'
ContextMenuCssClass="UserContextMenu" MenuControlPath
="~\CMSAdminControls\ContextMenus\MyGroupContextMenu.ascx" >

... transformation code ...

</cms:groupmenucontainer>
```

7.6.5.6 Adding custom functions to transformations

In many cases, you may need to process values and display them in a different format or add custom conditions. The following example shows you how to create a custom function that will return the first N characters of a text and how to use it in a transformation.

Open the 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 call it as your site code name. Right-click the folder and choose **Add New Item**. Choose to add a new Class and call it **MyFunctions.cs** (custom transformation functions can be developed only in C# at this moment).



Paste the following code to the *MyFunctions.cs* file:

[C#]

```
using System;
using System.Data;
using System.Configuration;
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;

/// <summary>
/// Summary description for MyFunctions
/// </summary>
public static class MyFunctions
{
        public static string TrimText(object txtValue, int leftChars)
        {
            if (txtValue == null | txtValue == DBNull.Value)
            {
                return "";
            }
            else
            {
                string txt = (string) txtValue;
                if (txt.Length <= leftChars)
                {
                    return txt;
                }
                else
                {
                    return txt.Substring(0, leftChars) + "...";
                }
            }
        }
}
```

Please note: The function must be defined as **static** so that you can call it from the transformation.

Then, go to **Site Manager -> 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><%# MyFunctions.TrimText(Eval("NewsSummary"), 20) %><br /><br />

..
```

Click **Save** to confirm the modification. Go to the live site and view the **News** page. As you can see, the news summary text is truncated to the first 20 characters.

In this topic, you have learned how to write your own transformation 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.

7.6.5.7 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 -> 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**:

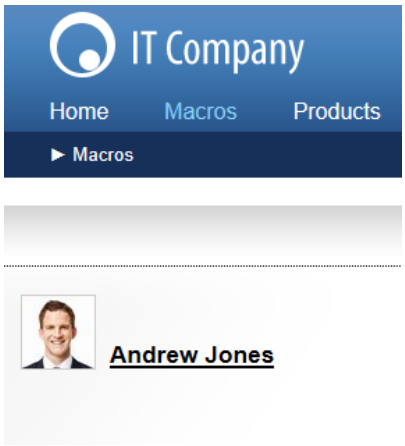
```
<div class="member">
  <a href="{% GetMemberProfileUrl(UserName) %}">
    {% GetUserAvatarImage(UserAvatarID, UserID, FullName, 52, 0, 0) %}
  </a>
<div class="memberInfo">
<p>
  <h3>
    <a href="{% GetMemberProfileUrl(UserName) %}">
      {% FullName %}
    </a>
  </h3>
</p>
</div>
</div>
```

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:

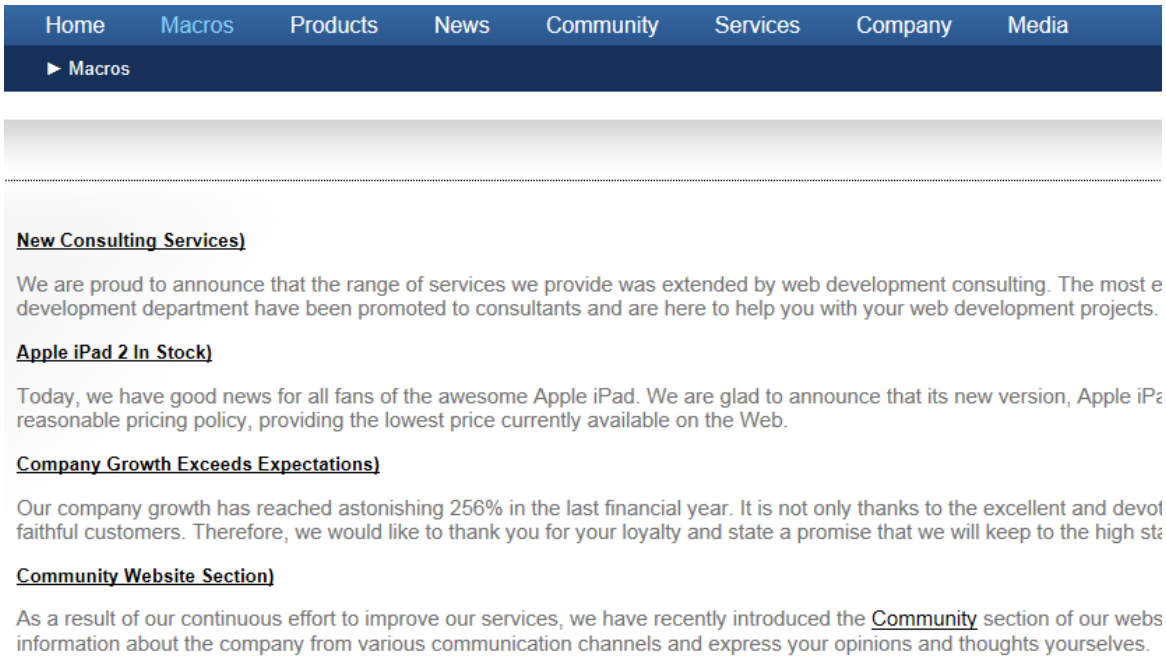
```
<div class="description">
  <a class="header bold" href="{% GetDocumentUrl() %}">
    {% NewsTitle %})
  </a>
  <p>
    {% NewsSummary %}
    <br />
  </p>
</div>
```

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**

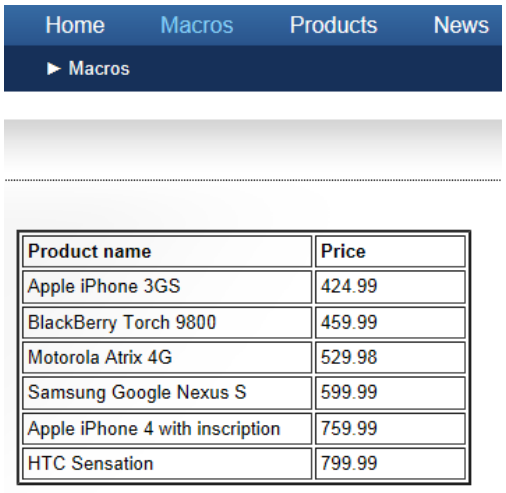
```
</table>
```

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:



7.7 E-mail templates

7.7.1 E-mail templates

Kentico CMS sends automatic system e-mails for various purposes, such as for example workflow notification. 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 at **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.
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	HTML code of the e-mail template.
Plain text version	Plain text of the e-mail template.

Example of the HTML version of a template:

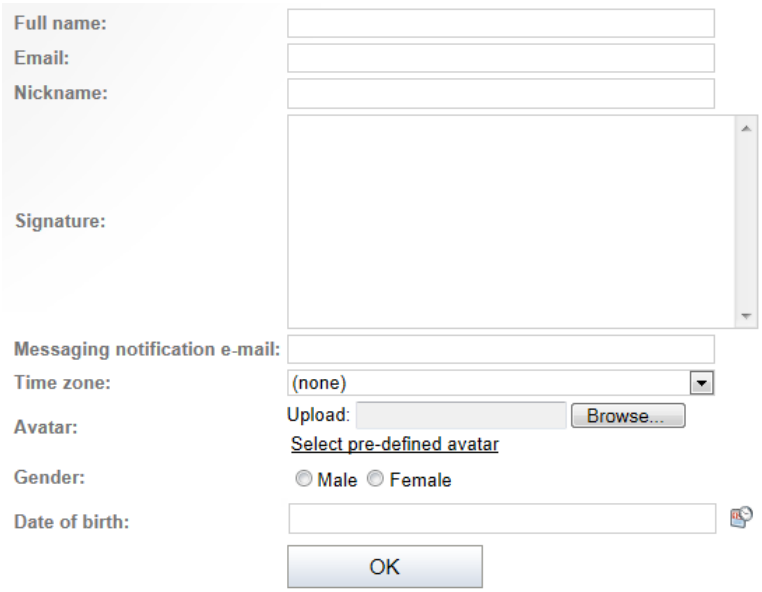
```
<html>
       <head>
       </head>
       <body style="FONT-SIZE: 12px; FONT-FAMILY: arial">
              <p>
              This an automatic notification sent by Kentico CMS. The following
document is waiting for your approval. Please sign in to Kentico CMS Desk and
approve it.
              </p>
              <p>
              <b>Document:</b> <a href="{%applicationurl%}/cmsdesk/default.aspx?
expandidpath={%idpath%}&expandmode=publish">                {%documentname%}</a>
              <br>
              <b>Last approved by:</b> {%approvedby%}
              <br>
              <b>Last approved when:</b> {%approvedwhen%}
              <br>
              <b>Original step:</b> {%originalstep%}
              <br>
              <b>Current step:</b> {%currentstep%}
              <br>
              <b>Comment:</b>
              <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.

7.8 Form controls

7.8.1 Overview

Form controls provide the interface for the 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.



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
- Forms
- Alternative forms for the objects mentioned above
- Report parameters

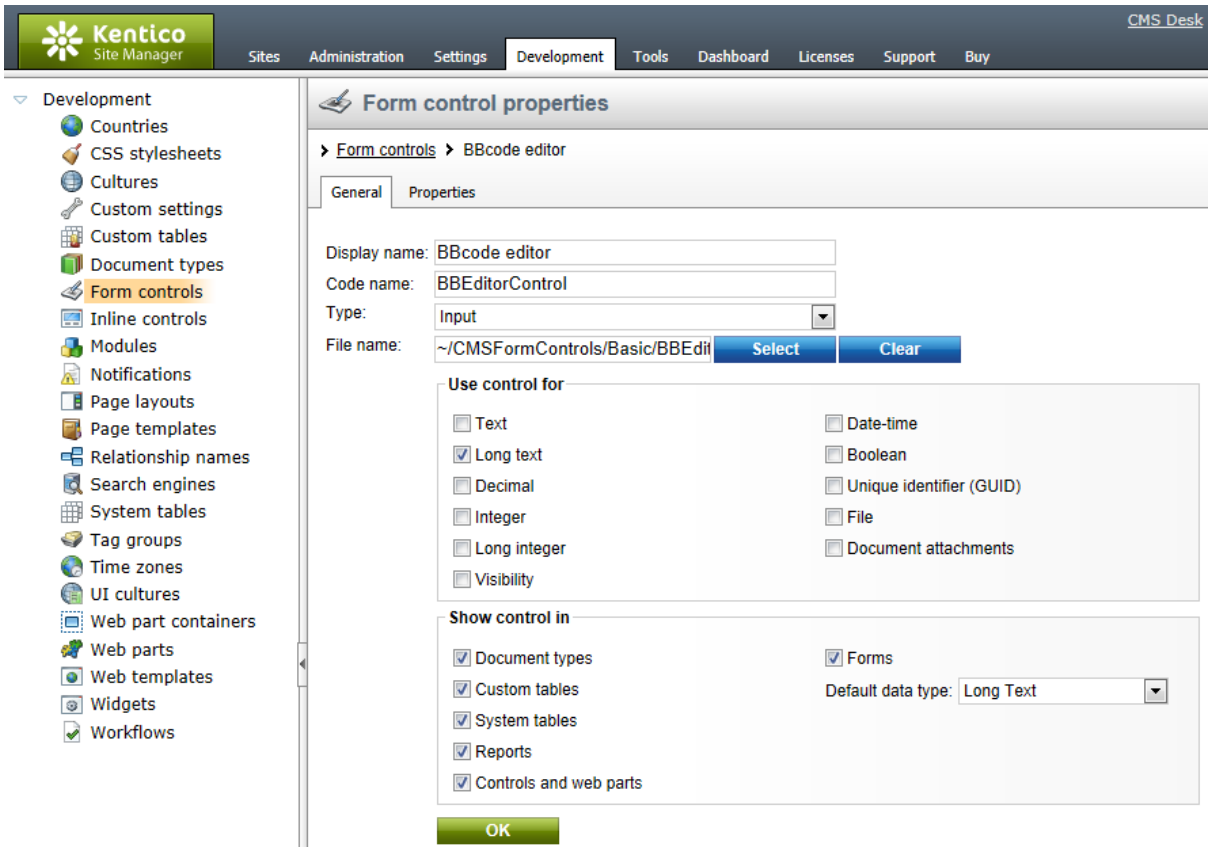
From a website development point of view, form controls are implemented by standard user controls (*.ascx* files) that inherit from the **CMS.FormControls.FormEngineUserControl** class.

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.

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 used in the interface, e.g. when selecting a control in a field editor.
- **Code name** - sets the name of the control that is used as its unique identifier in code.
- **Type** - allows the selection of the category under which the form control will be placed. The type can be used to filter form controls in the field editor.
- **File name** - contains the relative path to the *.ascx* file that implements the form control, for example: *~/CMSFormControls/Basic/TextBoxControl.ascx*



The properties under the **Use control for** section are used to set 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:

- **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 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 fields of forms, which can be edited in *CMS Desk -> Tools -> Forms*. 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:

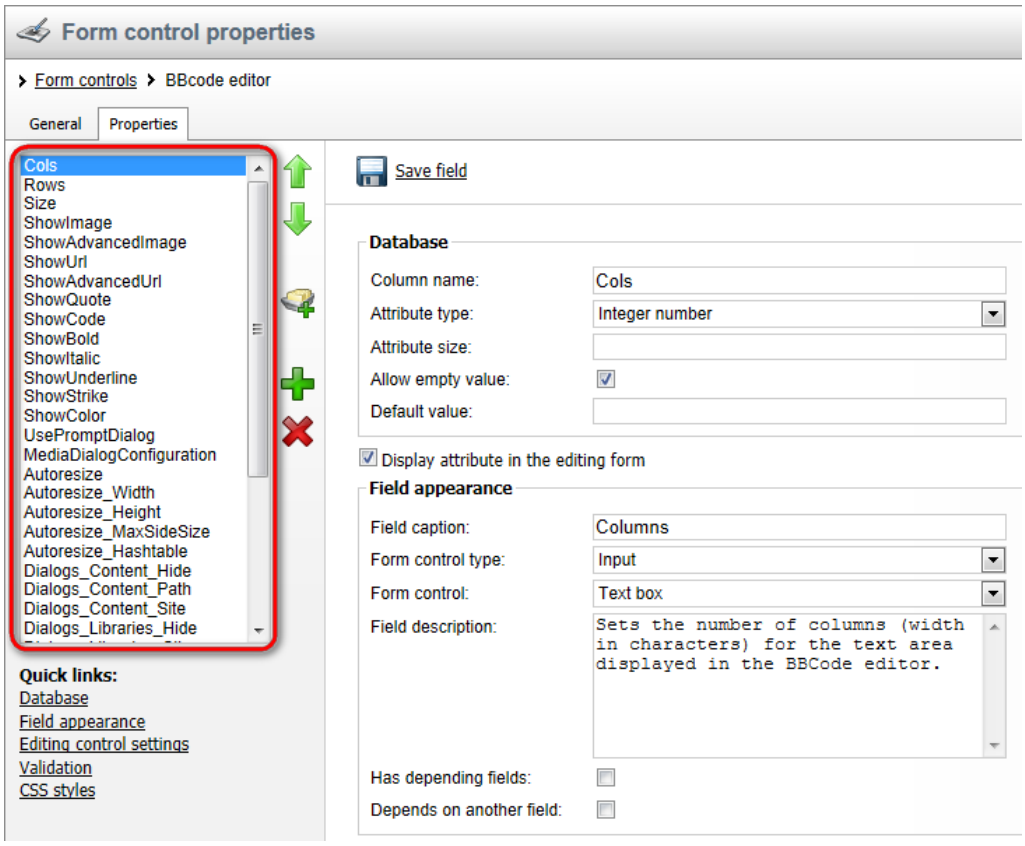
  - **Default data type** - the data type of the field that will be used by default when a user chooses to create a new field of this type.
  - **Column size** - only applies to the *Text* data type. This sets the maximum size of the database column used to store the field, which also limits amount of 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.

7.8.2 Form control parameters

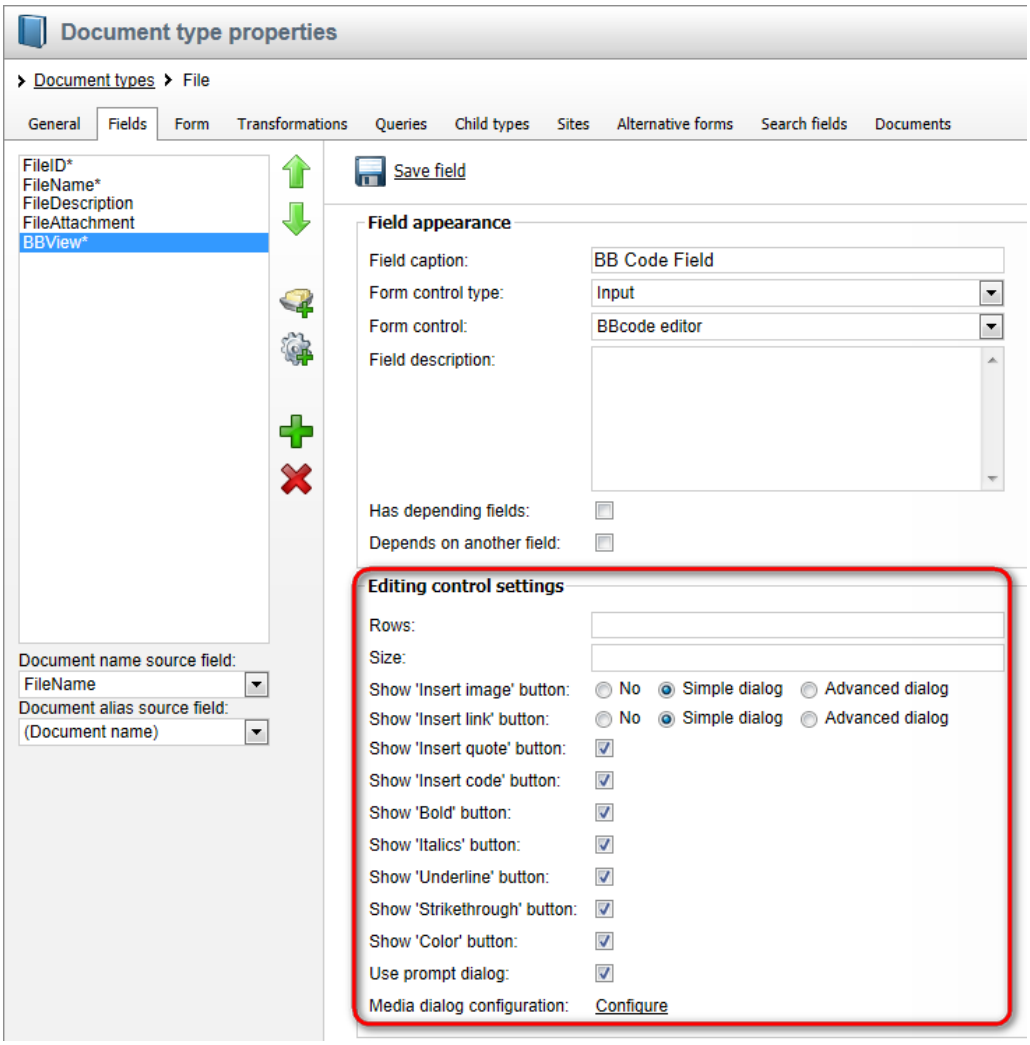
It is possible to define parameters for form controls that can be used to further specify the behaviour or appearance of the generated form fields. Parameters can greatly increase the flexibility of a form control, since they provide additional configuration options for individual fields that use the 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 using 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.



Any 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 new form controls when creating custom fields.

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.

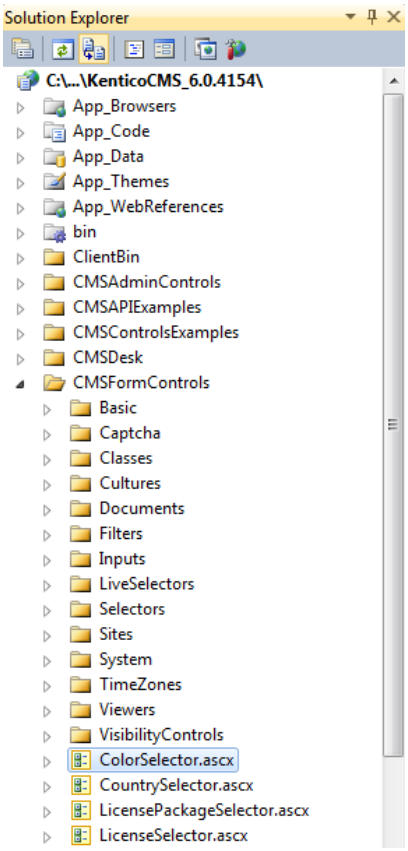
For more details, please see the example in the Developing form controls topic, which demonstrates how a parameter can be added to a form control and handled in the code.

7.8.3 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.

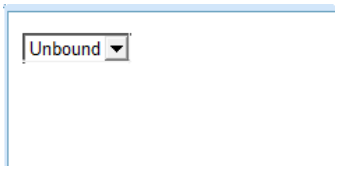
1. Open the web project in Visual Studio (or Visual Web Developer) using the **WebProject.sln** file or via **File -> Open -> Web site** in Visual Studio.

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 **DropDownList** control onto the form:



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:

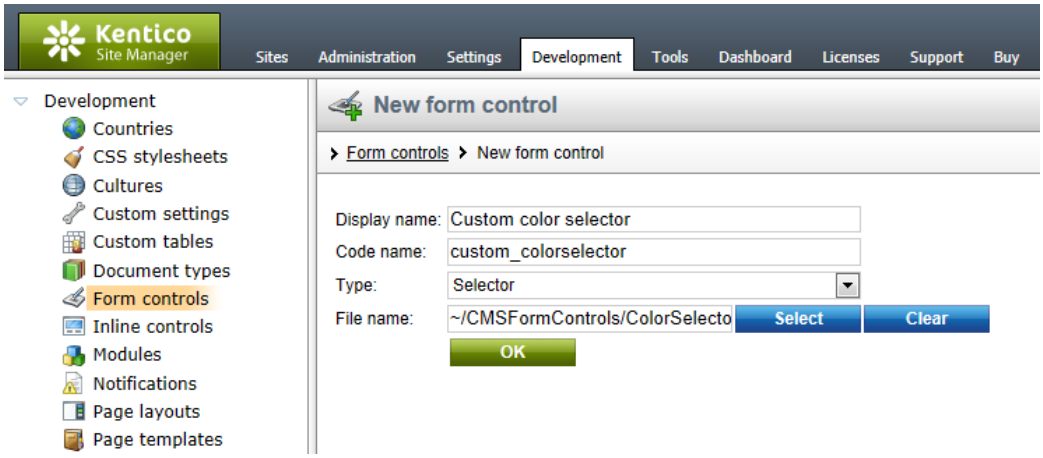
- **Value** - it is necessary to override this property for every form control. It is used to get and set the value of the field provided by the control.
- **GetOtherValues()** - this method is used to set values for other fields of the object in which the form control is used. It must return a two dimensional 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.

8. Go to **Site Manager -> Development -> Form controls** where you can register the new form control in the system. Click the **New form control** link. Enter the following values:

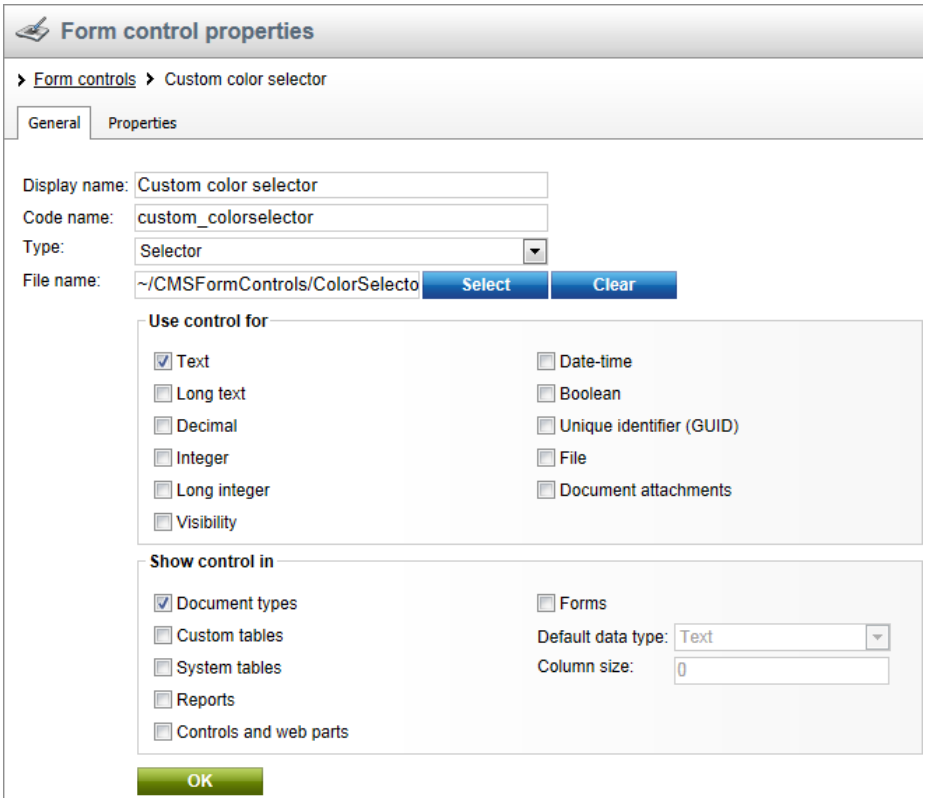
- **Display name**: Custom color selector
- **Code name**: custom_colorselector
- **Type**: Selector
- **File name**: ~/CMSFormControls/ColorSelector.ascx (you can use the **Select** button to choose the file)



Click **OK** and the form control object 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 **OK**.

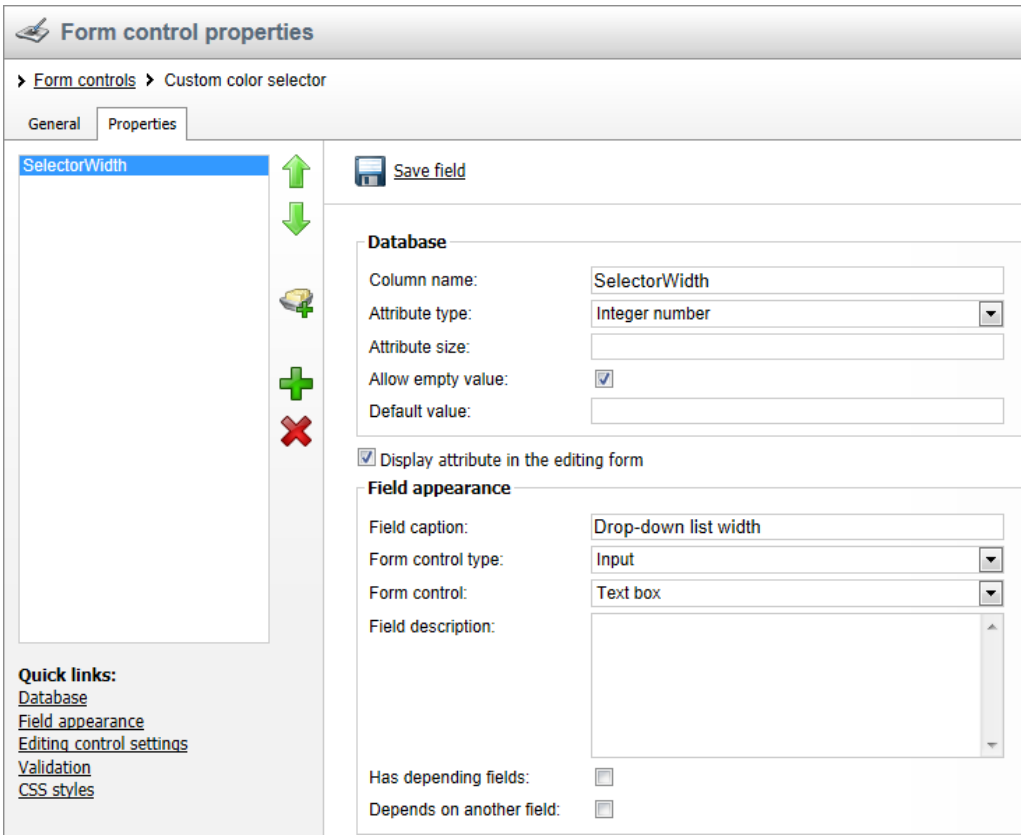


10. Switch to the **Properties** tab where parameters can be defined for the form control. Use the **New attribute** () action and set the following properties:

- **Column name**: SelectorWidth
- **Attribute type**: Integer number
- **Allow empty value**: true
- **Display attribute in editing form**: true

- **Field caption**: Drop-down list width
- **Form control type**: Input
- **Form control**: Text box

When finished, click  **Save field**.



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 field** and create the next field:

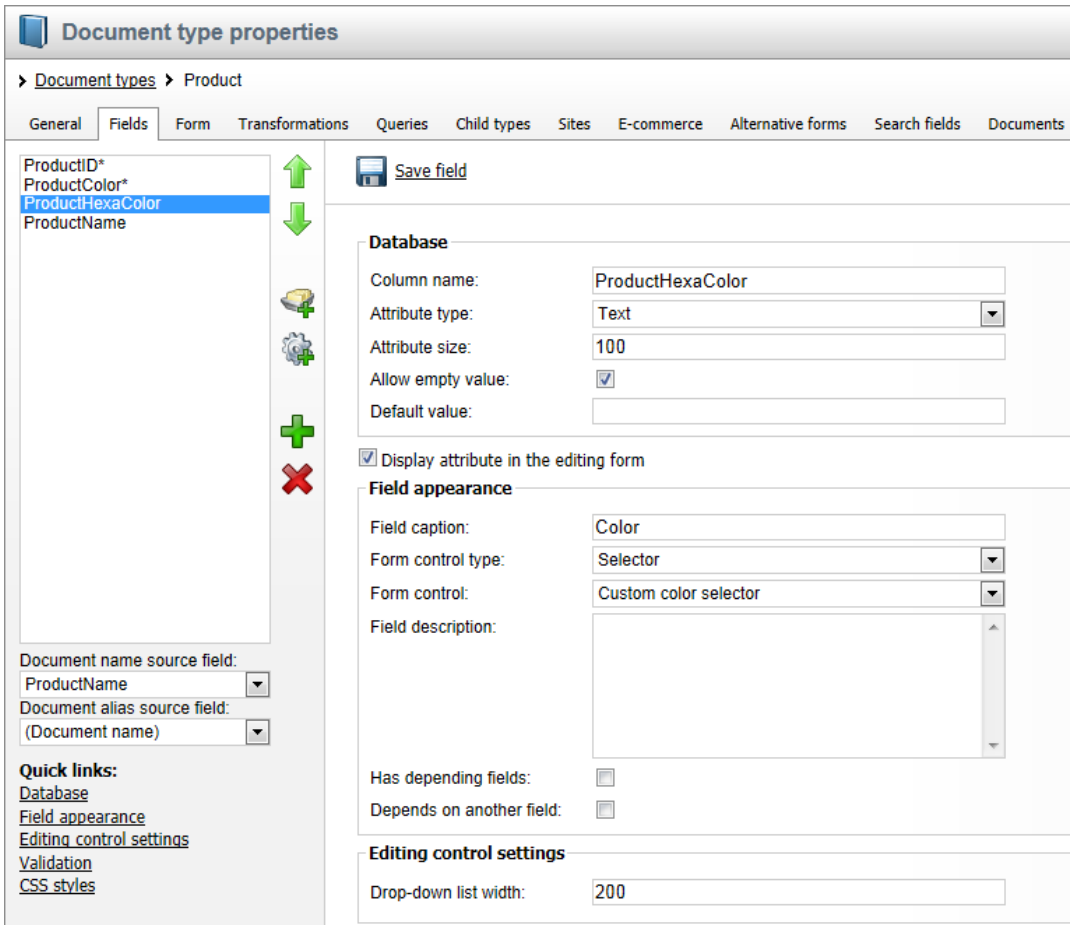
- **Column name**: ProductHexaColor
- **Attribute type**: Text
- **Attribute size**: 100
- **Allow empty value**: true
- **Display attribute in editing form**: true

- **Field caption**: Color
- **Form control type**: Selector
- **Form control**: Custom color selector

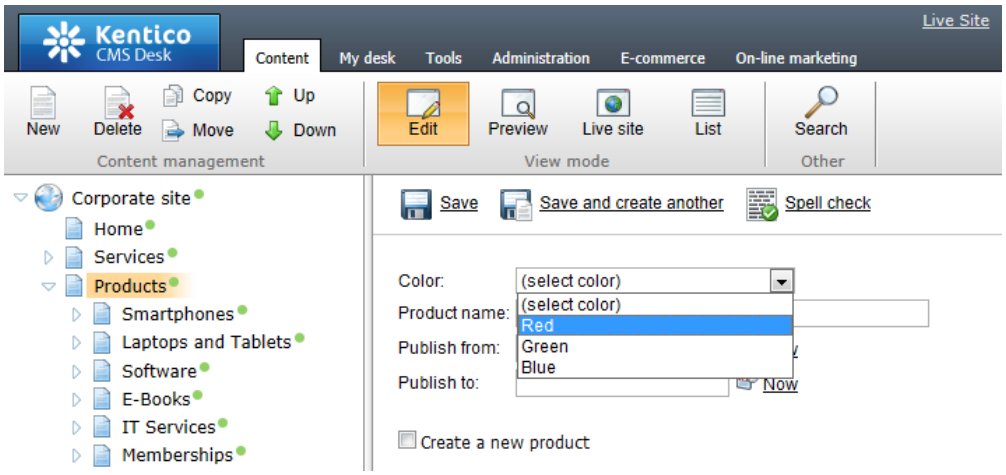
This field will store the hexadecimal code of the selected color. In the code of the form control, this value is handled through the **Value** property. The field will be displayed in the document's editing form according to the design of the custom form control.

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 field** again.



12. Go to **CMS Desk -> Content** and create a new document of the **Product** document type under the **Products** 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 using the **Form.Data** property of the form control inherited from the **FormEngineUserControl** class. You can get or set the values of other fields that are part of the form using the following syntax:

- **Form.Data.GetValue(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.Data.GetValue("ProductName"), "New product");
```

- **Form.Data.SetValue(string columnName, object value)** - sets a value for the specified field.

For example, the following code sets the value of the **ProductName** field to *Colored product*:

[C#]

```
Form.Data.SetValue("ProductName", "Colored product");
```

If you wish to change the value of a field **before the form is loaded**, you need to place the code inside the **Page_Load** method of your form control.

If you need 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.

7.9 Inline controls

7.9.1 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:

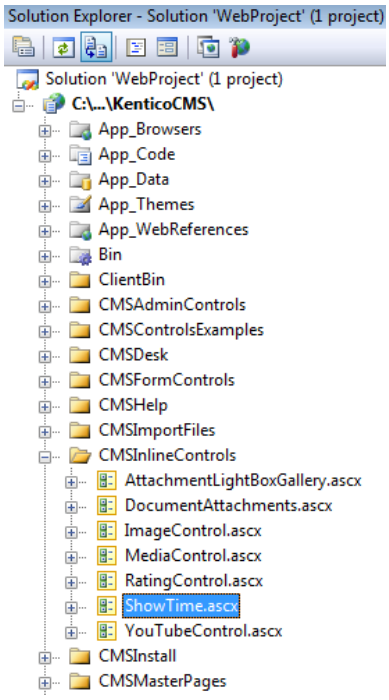
```
<add key="CMSEnableInlineControls" value="true" />
```

7.9.2 How to develop inline controls

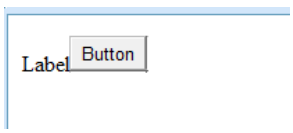
This topic will show you an example of inline control development. We will create a simple control that will display the current time when a button is clicked.

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
```

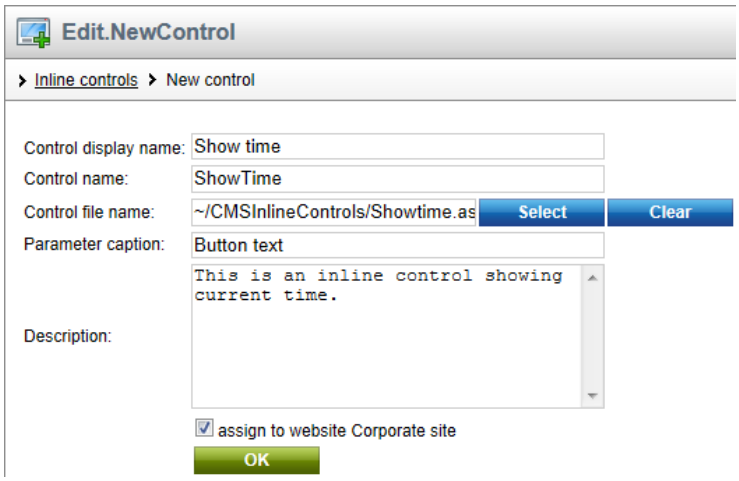
What you did

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:

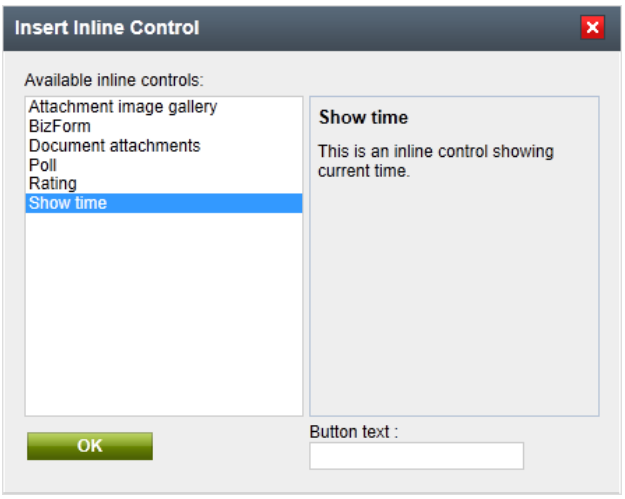
- **Control display name:** Show time
- **Control name:** ShowTime (the name of the user control without the extension)
- **Control file name:** ~/CMSInlineControls/ShowTime.ascx
- **Parameter caption:** Button text



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.

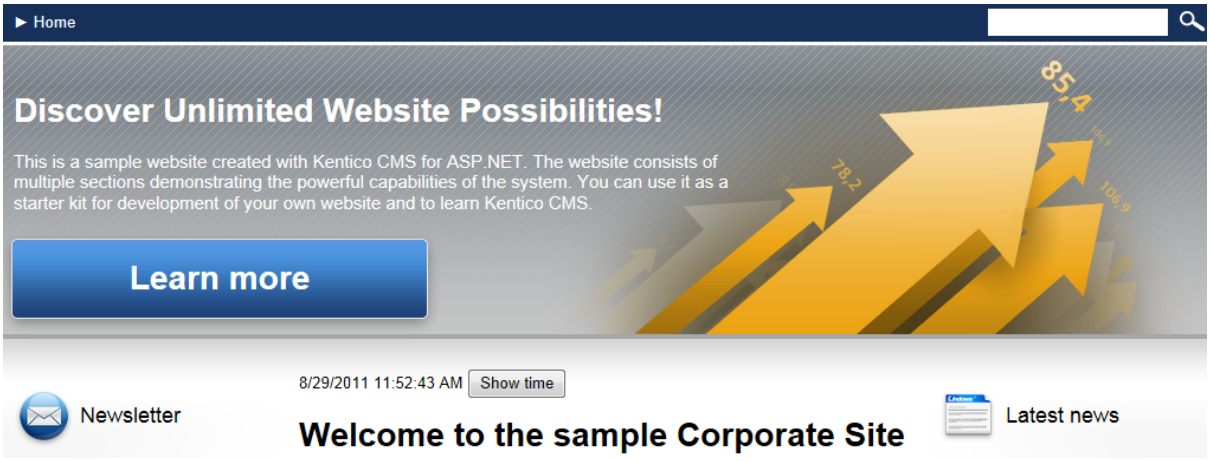


Please note

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:



7.10 Macro expressions

7.10.1 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 versions of Kentico CMS prior to 6.0 are still functional and available, even though the same results can now be achieved using the 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. 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. At first, it is possible to define custom macro methods and use them in macro expressions along with the default ones. A step-by-step process of creation of such methods is described in the Registering custom macro methods topic. It is also possible to cache values of resolved macros in order for certain macros not to be resolved repeatedly. More information on this can be found in the Macro result caching topic. Finally, macro expressions can be resolved in your custom code, while it is even possible to define your custom macro resolvers and resolve macros using them. This is described in the Resolving macros using API topic.

7.10.2 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:

- % - Context (data) macro
- $ - Localization macro
- ? - QueryString macro
- @ - Cookie macro
- # - Custom macro
- ^ - Control macro
- & - Path 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 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 locatization macros:

Basic

Basic localization macros are entered in the *{$string.key$}* format. The system uses the *ResHelper.GetString("string.key")* method and replaces the macro with the appropriate resource string.

Example: "*The weather is {$General.OK$}"* will be resolved as "*The weather is OK"*.

In-place locatization

The string and its localized equivalents are included within the entered macro. The macro is entered in the *{$=Default string|cs-cz=Czech string|de-de=German string$}*. It uses a localized string in case it is available or the default (first) string if it is not.

Example: "*The weather is {$=OK|cs-cz=dobre|de-de=gut$}*" will be resolved into "*The weather is dobre*" in Czech culture, "*The weather is gut*" in 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 cookie values. These macros can be used for example to parametrize the web parts with client-based persistent values like styles or user options.

The macro is entered in the *{@cookie_name@}* format. Alternatively, you can also enter them as data macros like the following: *{%Cookies.cookie_name%}*. The macro is replaced by the cookie value.

Example: *"Current style: {@StyleCookie@}"* will be resolved as "*Current style: Red*" if *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. 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 will be resolved as parent document path and will result 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

These macros can be used to define your own macro. The macro is in the *{#Expression#}* format. Alternatively, you can also enter them as data macros like the following: *{%Custom.Expression%}*.

When such a macro needs to be resolved, methods registered in the **MacroResolver.OnResolveCustomMacro** handler are called. These methods ensure resolving of individual expressions into their result equivalents. An example of registration of such a method can be found in the *App_Code/Samples/Modules/SampleMacroModule.cs* class. The class extends the **CMSModuleLoader** partial class.

In the class, the following method is implemented. This method ensures that the *{#someexpression#}* macro gets resolved into the *"Resolved expression"* string.

```
/// <summary>
/// Resolves the custom macro
/// </summary>
/// <param name="sender">Sender</param>
```

```
/// <param name="e">Event arguments</param>
private void MacroResolver_OnResolveCustomMacro(object sender, MacroEventArgs e)
{
    if (!e.Match)
    {
        // Add your custom macro evaluation
        switch (e.Expression.ToLower())
        {
            case "someexpression":
                e.Match = true;
                e.Result = "Resolved expression";
                break;
        }
    }
}
```

In the **Init()** method, which is executed on each application start, you can find the following code which ensures registration of the method in the **MacroResolver.OnResolveCustomMacro** handler (the code is commented by default).

```
/// <summary>
/// Registers module methods.
/// </summary>
public override void Init()
{
    // -- Custom macro methods
    //CustomMacroMethods.RegisterMethods();

    // -- Custom macro resolving
    MacroResolver.OnResolveCustomMacro += MacroResolver_OnResolveCustomMacro;

    // -- Custom output substitution resolving
    //OutputFilter.OnResolveSubstitution += OutputFilter_OnResolveSubstitution;
}
```

Your custom macros can be defined the same way, either by adding other cases into the switch statement within the **MacroResolver_OnResolveCustomMacro** method in this class, or by implementing your custom classes extending the **CMSModuleLoader** partial class which would contain your custom macro resolving methods and their registration in the **MacroResolver. OnResolveCustomMacro** handler.

Substitution macros

Substitution macros are entered in the {~expression~} format. They are similar to the Custom macros described above. The difference is that these macros are **resolved in page output code**, i.e. the page's final code before it is sent to the browser for rendering. The main advantage of this type of macro is that these macros are resolved on each request, even on pages that use full-page caching.

When such a macro needs to be resolved, methods registered in the **OutputFilter. OnResolveSubstitution** handler are called. These methods should ensure resolving of individual

expressions into their result equivalents. An example of registration of such a method can be found in the *App_Code/Samples/Modules/SampleMacroModule.cs* class. The class extends the **CMSModuleLoader** partial class.

In the class, the following method is implemented. This method ensures that the *{~somesubstitution~}* macro gets resolved into the *"Resolved substitution"* string.

```
/// <summary>
/// Resolves the output substitution
/// </summary>
/// <param name="sender">Sender</param>
/// <param name="e">Event arguments</param>
private void OutputFilter_OnResolveSubstitution(object sender, CMS.OutputFilter.
SubstitutionEventArgs e)
{
    if (!e.Match)
    {
        // Add your custom macro evaluation
        switch (e.Expression.ToLower())
        {
            case "somesubstitution":
                e.Match = true;
                e.Result = "Resolved substitution";
                break;
        }
    }
}
```

In the **Init()** method, which is executed on each application start, you can find the following code which ensures registration of the method in the **OutputFilter.OnResolveSubstitution** handler (the code is commented by default).

```
/// <summary>
/// Registers module methods.
/// </summary>
public override void Init()
{
    // -- Custom macro methods
    //CustomMacroMethods.RegisterMethods();

    // -- Custom macro resolving
    //MacroResolver.OnResolveCustomMacro += MacroResolver_OnResolveCustomMacro;

     // -- Custom output substitution resolving
     OutputFilter.OnResolveSubstitution += OutputFilter_OnResolveSubstitution;

}
```

Your custom substitutions can be defined the same way, either by adding other cases into the switch statement within the **MacroResolver_OnResolveSubstitution** method in this class, or by implementing your custom classes extending the **CMSModuleLoader** partial class which would contain

your custom substitution resolving methods and their registration in the **MacroResolver.OnResolveSubstitution** handler.

7.10.3 K# syntax

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 **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." %}
```

- K# supports 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"} %}
```

- K# supports 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"} %}
```

- K# supports the **for** flow control command written in format: *for (<init_expression>; <condition>; <incr_expression>) {<expressions>}*. The flow command itself has a return value. It returns a concatenated list of return values of <*expressions*> from each iteration of the cycle.

```
// 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 } %}
```

- K# supports the **foreach** flow command written in format: *foreach (<variable> in <enumerable>) {<expressions>}*. The flow command itself has a return value. It returns a concatenated list of return values of <*expressions*> from each iteration of the cycle.

```
// returns "HELLO"
{% z = ""; foreach (x in "hello") {z += x.toupper()}; z %}

// returns "HHEHELHELLHELLO"
{% z = ""; foreach (x in "hello") {z += x.toupper()}%}
```

- K# supports the **while** flow control command written in format: *while (<condition>) {<expressions>}*. The flow command itself has a return value. It returns a concatenated list of return values of <*expressions*> from each iteration of the cycle.

```
// returns "10"
{% z = 1; while (z<10) {++z}; z %}

// returns "2345678910"
{% z = 1; while (z<10) {++z} %}
```

- K# supports the **break** command in cycles. This command terminates the nearest enclosing loop in which it appears.

```
// returns "12345"
{% z = 0; while (z < 10) {if (z > 4) {break}; ++z} %}
```

- K# supports the **continue** command in cycles. 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} %}
```

K# supports the **return** command in cycles. 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;}} %}
```

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 cycle 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

Please note that the operators support DateTime and TimeSpan operands only if hotfix 6.0.22 or later is applied.

- The + operator can be used 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 is 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 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.

7.10.4 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. If hotfix 6.0.22 or later is applied, 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. If hotfix 6.0.22 or later is applied, 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. If hotfix 6.0.22 or later is applied, 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. If hotfix 6.0.22 or later is applied, 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. If hotfix 6.0.22 or later is applied, 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. Please note that this method is only available if hotfix 6.0.22 or higher is applied.
Object value: Object to convert.

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 item stored on the specified index from the collection.
IEnumerable collection: Collection.
Int32 index: Index of the item.

AbstractObjectCollection OrderBy (AbstractObjectCollection collection, String orderBy)
Order by on collection.
AbstractObjectCollection collection: Collection to order.
String orderBy: ORDER BY column(s).

AbstractObjectCollection Where (AbstractObjectCollection collection, String where)
Where on collection.
AbstractObjectCollection collection: Collection to filter.
String where: WHERE condition.

AbstractObjectCollection TopN (AbstractObjectCollection collection, Int32 parameters)
TopN on collection.
AbstractObjectCollection collection: Collection to filter.
Int32 parameters: TOP N parameter.

AbstractObjectCollection Columns (AbstractObjectCollection collection, String columns)
Columns on 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.

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

Please note that these methods are only available if hotfix 6.0.22 or higher is applied.

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 will be 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 0.01 multiple of first argument.
Double percent: Number of percent (number to be multiplied by 0.01).

Int32 Modulo (Int32 left, Int32 right)
Returns modulo of two values.
Int32 left: Left operand.
Int32 right: Right operand.

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.

7.10.5 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 using 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 URL returned by the macro expression.
- **|(unresolveurl)** - unresolves 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.
- **|(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.

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*.

7.10.6 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
- Edit value dialog in web part properties
- Macro condition editor

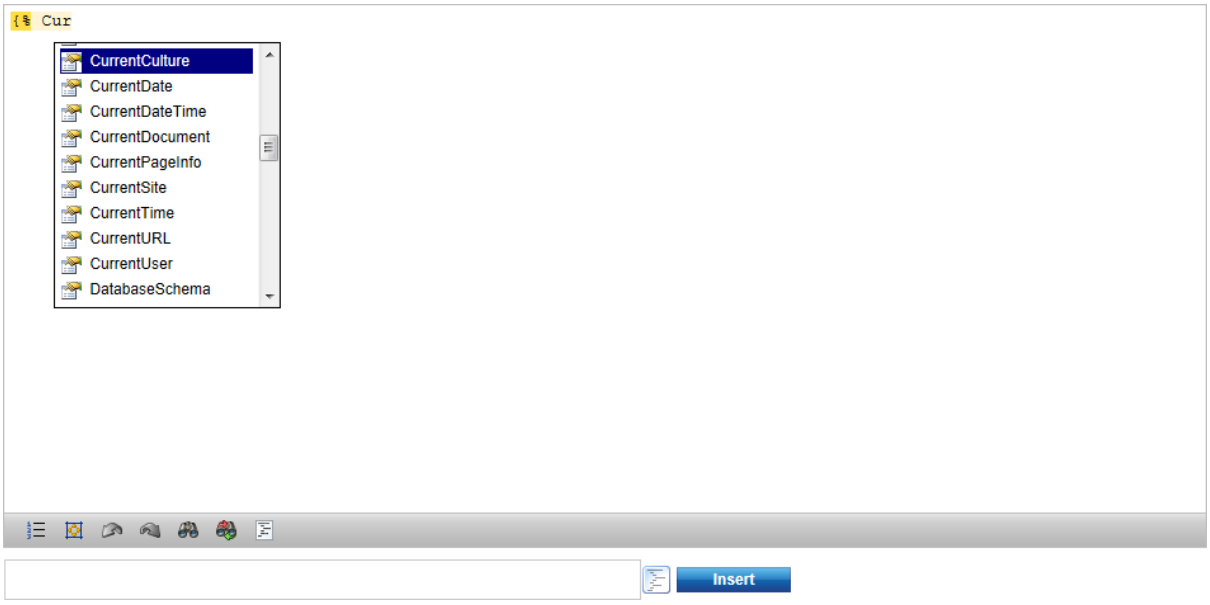
Click the feature name above to get redirected to the section of this page where it is described.

Macro autocompletion

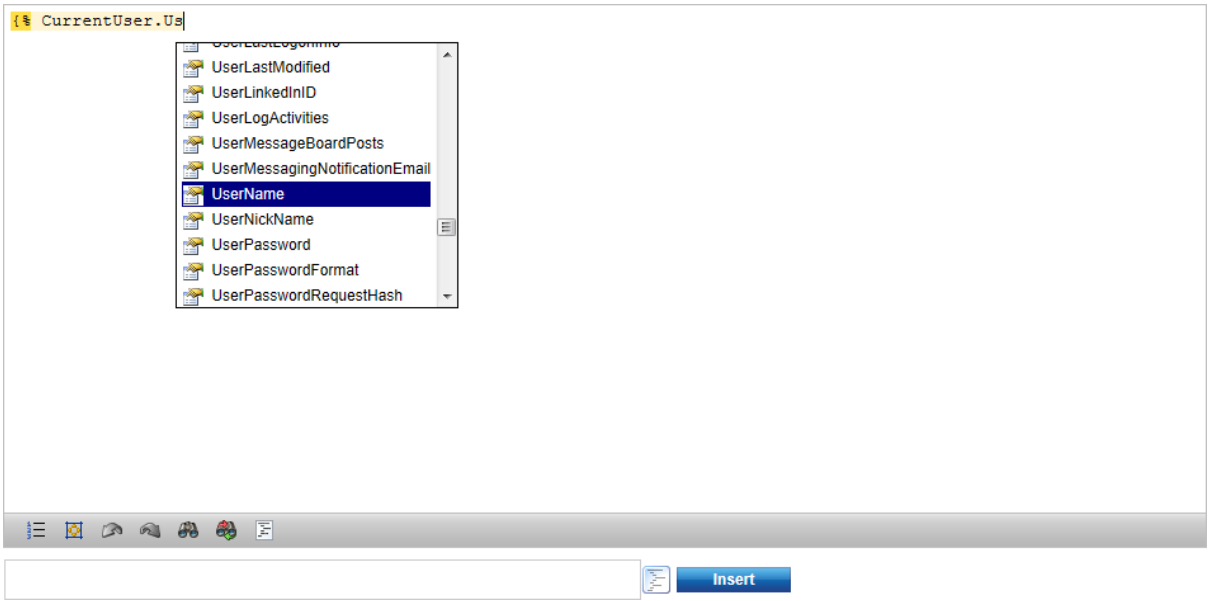
Automatic completion of macro expressions is available when writing macro expressions in:

- **E-mail templates**
- **Transformations**
- **Web part properties**

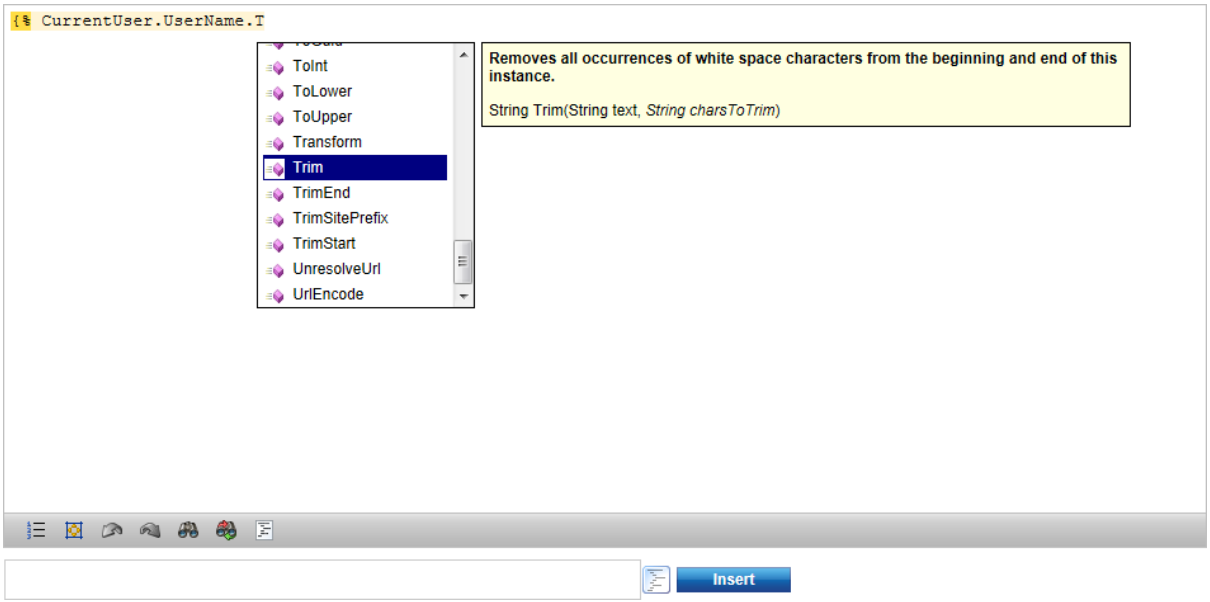
This feature is similar to IntelliSense in Visual Studio — as you type, a box with methods or properties that begin with the letters you wrote is displayed below the cursor. Only those methods and properties that are relevant in the current context are displayed in the box. The methods and properties are listed in alphabetical order and you can navigate through them using the up and down arrows. Once you select the appropriate one, press Enter or space to insert it into the text.



The box with available methods and properties is also displayed when creating further parts of expressions using the dot-suffix notation.



When a method is selected from the drop-down list, its description and signature (containing 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 italic letters (as *String charsToTrim* in the screenshot below).



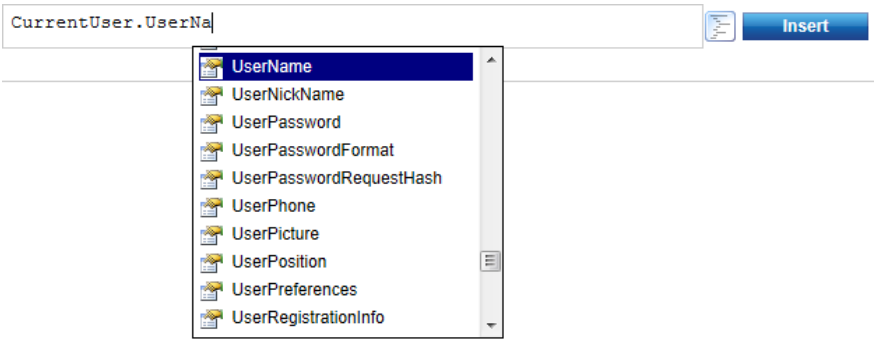
Macro selection control

Another feature that makes entering of macro expressions easier is the macro selection control.

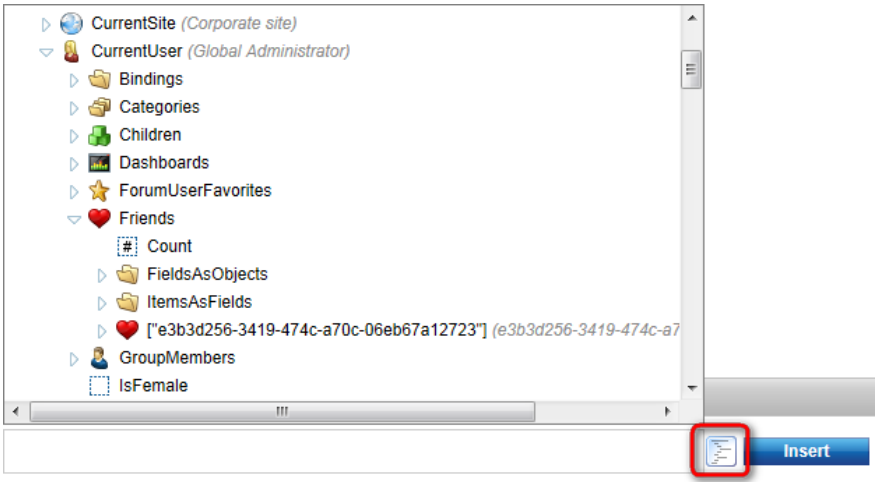
It is present in the following locations:

- **Newsletter issues**
- **Newsletter templates**
- **E-commerce invoice templates**
- **E-mail templates**
- **Web part properties**

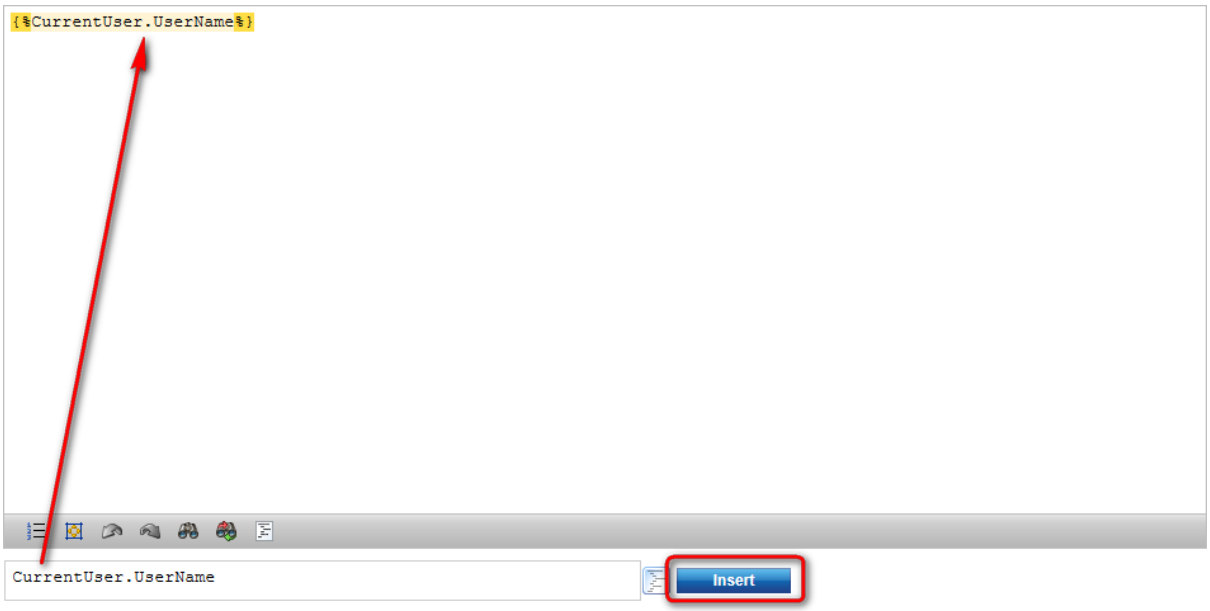
The control can be used two different ways. The first one is simply typing the macro text into the text box, while macro autocompletion is available here as well.



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 select objects or their properties from the current context. By clicking an object or its property, the respective expression is entered into the text box automatically.

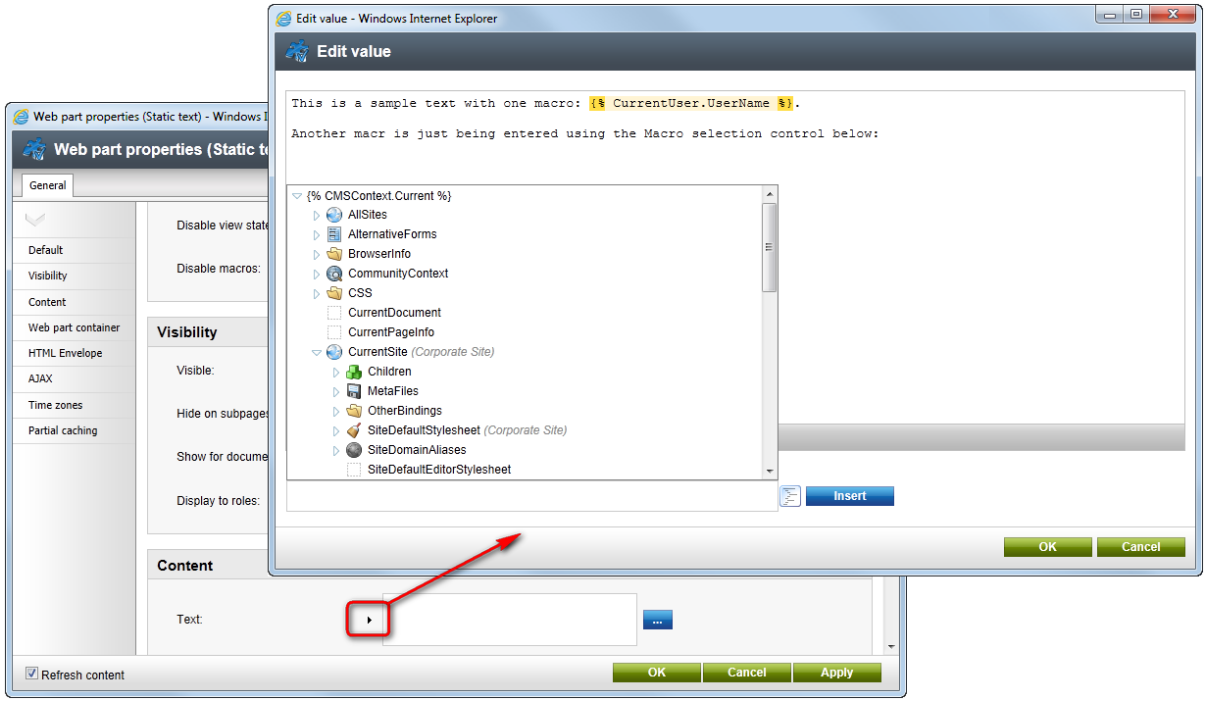


When you finally have the required expression in the text box, click the **Insert** button to paste it into the current position in the edited text. The expression will be pasted, enclosed within the {%%} data macro parentheses.



Edit value dialog in web part properties

Macro expressions can be used in values of web part properties. Next to all properties, you can find the ▸ button highlighted in the screenshot below. If you click it, the **Edit value** dialog pops up and lets you enter the property value. In the dialog, you can use both macro autocompletion and the macro selection control described above.



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.

To disable single quote escaping for all properties of a specific 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";
```

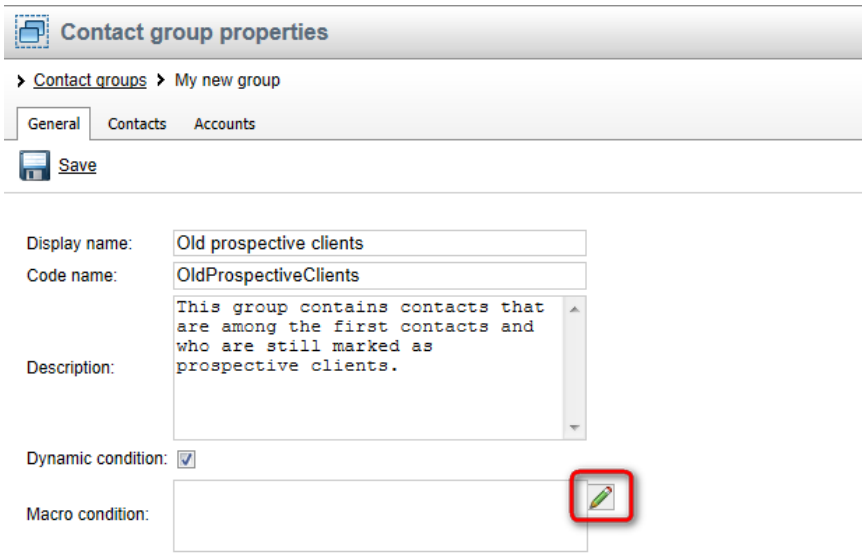
Please note that disabling SQL protection may create a **security vulnerability** if the macro resolves its value according to data that can be modified by the website's users, such as in the case of QueryString macros.

Macro condition editor

The **Macro condition editor** facilitates entering of dynamic conditions which depend on current values of specified macros. The control can be currently used when editing definitions of:

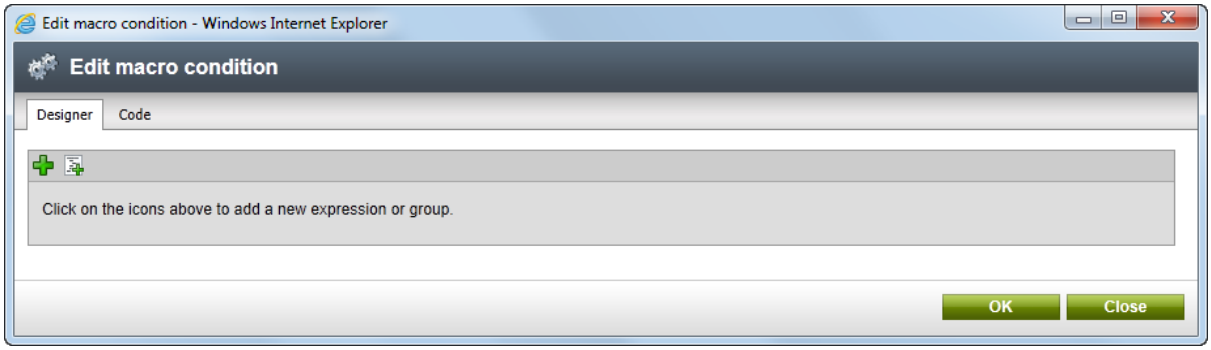
- **Contact groups**
- **Web analytics campaigns**

It is accessible by clicking the icon next to the respective condition field.



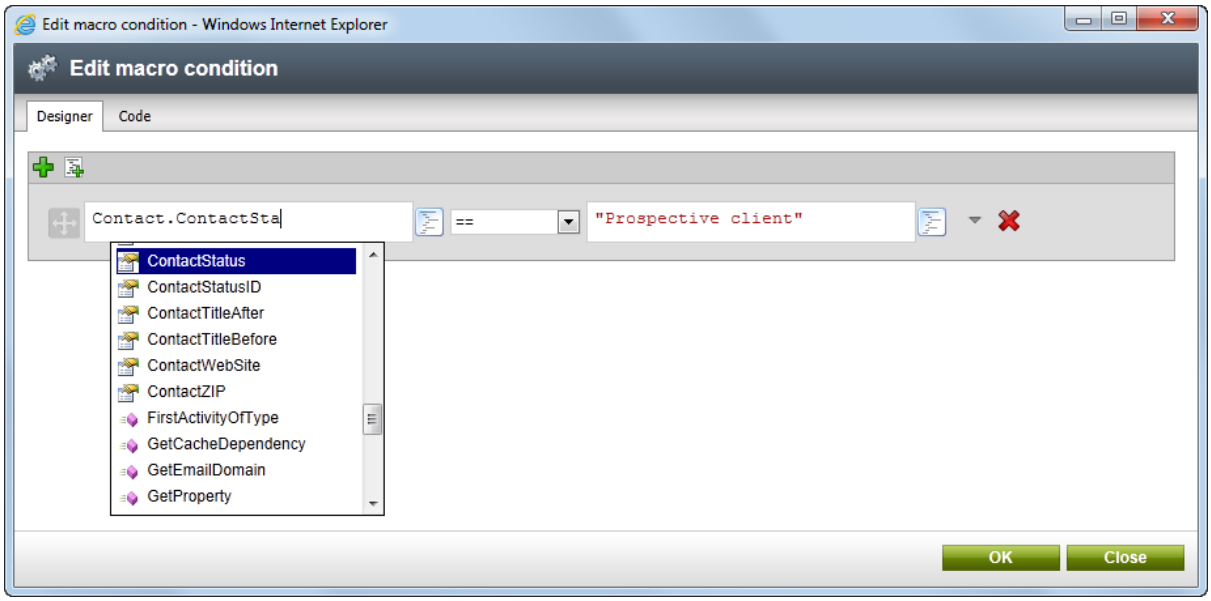
After clicking the icon, the **Edit macro condition** pop-up window appears. Initially, it looks as in the following screenshot — the editor contains one group where macro expressions or other sub-groups can be added, while the following actions are available in its header:

- **Add group** - adds a new sub-group into the group.
- **Add expression** - adds a new macro expression into the group.

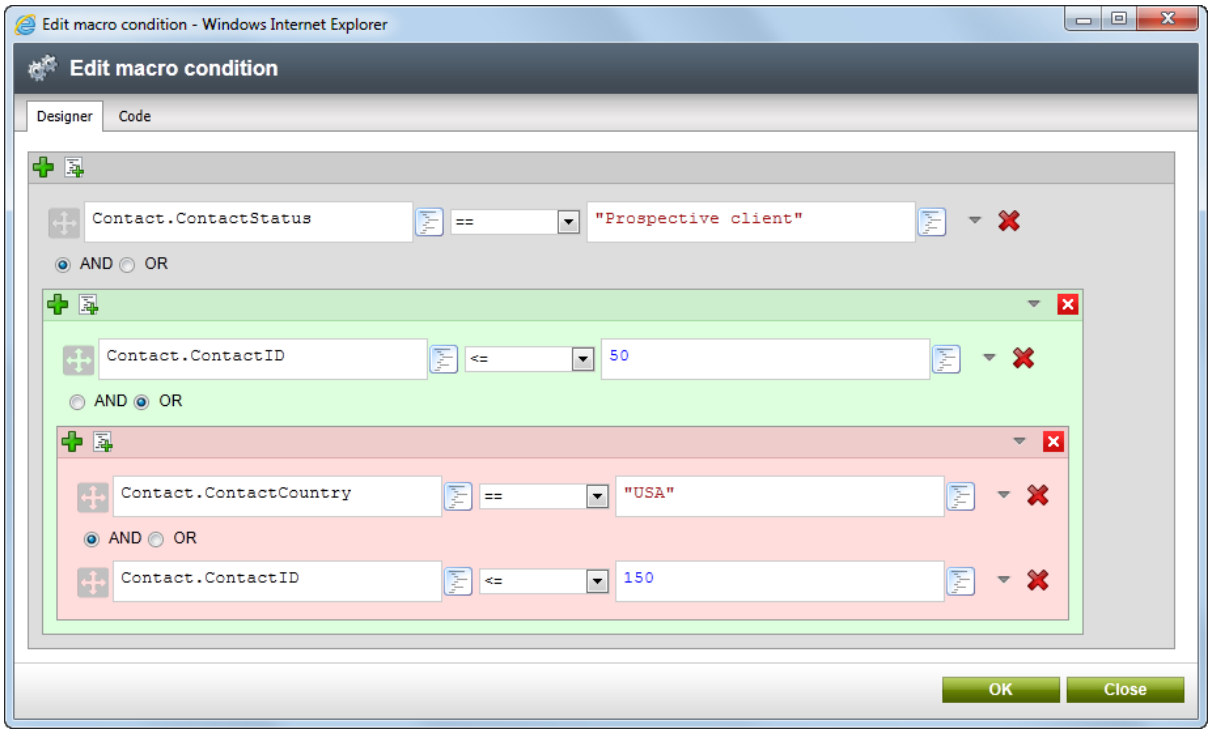


When you insert an expression, a set of two macro selection controls with a relation drop-down list in between them is displayed. Using the macro selection controls, you can either enter a macro expression, or just a simple value. Macro autocompletion and the **Show/hide macro object tree** ( ) button are available for easier macro specification. Using the relation drop-down list, you can specify the relation between the values on the left and on the right:

- **==** - the values must be equal.
- **!=** - the values must not be equal.
- **>** - the left value must be greater than the right value.
- **<** - the left value must be lesser than the right value.
- **>=** - the left value must be greater than or equal to the right value.
- **<=** - the left value must be lesser than or equal to the right value.



By adding groups, you can specify multiple expressions and combine them using logical conjunction (**AND**) or logical disjunction (**OR**). The same can be done with multiple expressions within one group. As a result, you can specify advanced conditions, just as the one displayed in the screenshot below.



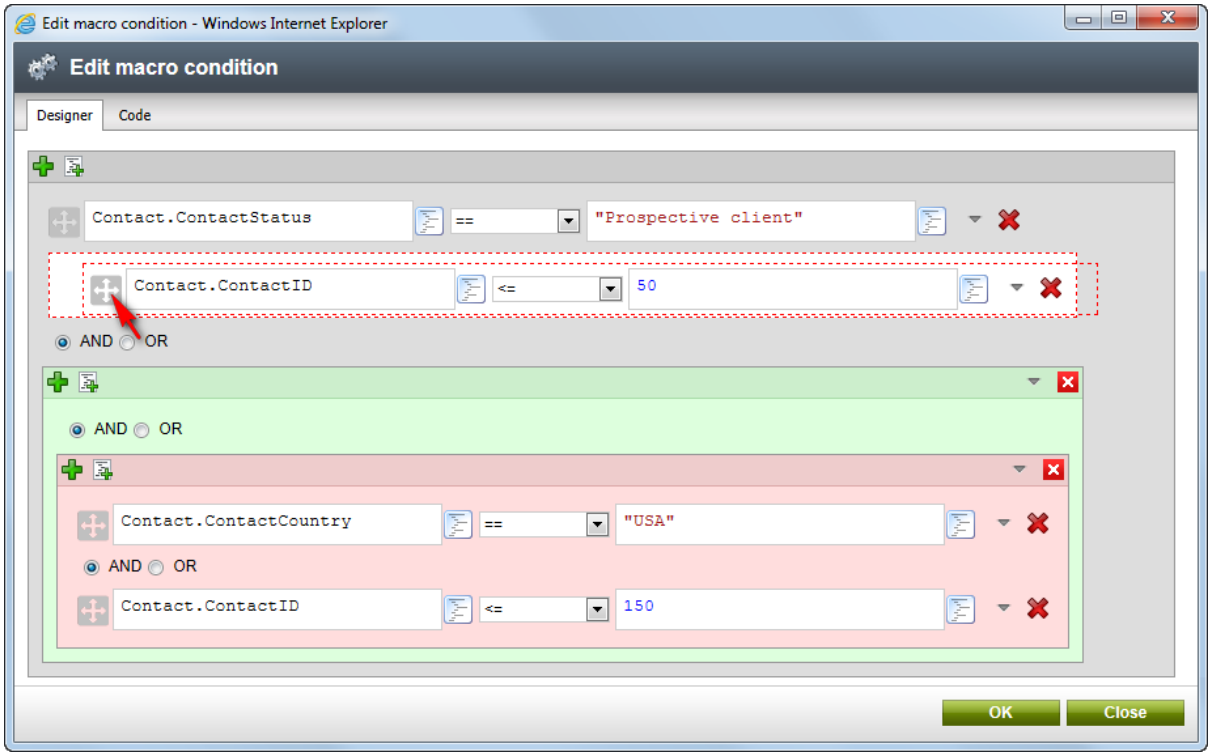
Already defined expressions and groups can be removed using the following action icons:

- **Remove group** - removes the whole group, including all expressions and sub-groups that it contains.
- **Remove expression** - removes the respective macro expression.

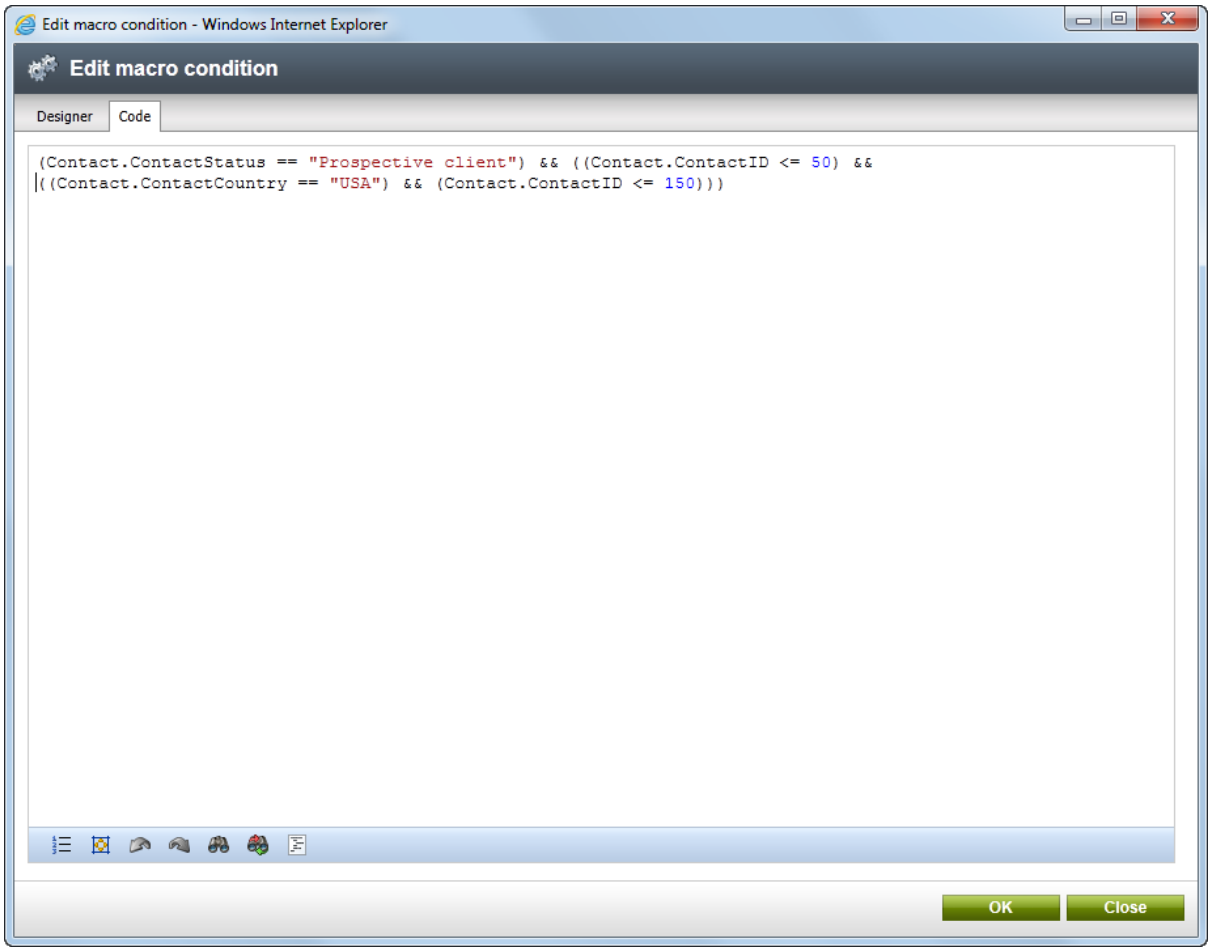
Individual expressions and groups can also be re-ordered using actions in the context menus accessible by clicking the respective ▼ icons:

- **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.

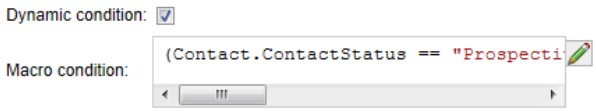
Re-ordering can alternatively be performed by dragging and dropping an expression or group into the desired location. Groups need to be dragged by their header row, expressions need to be dragged by the cross icon on the left.



After switching to the **Code** tab, you can view the code of the defined macro condition and edit it manually when needed.



Finally, when you finish specifying the condition on either of the two tabs, click **OK** to insert the condition into the field.



7.10.7 Macro security

When resolving certain macros, permissions to read the resolved data are checked. The check is performed for **permissions granted to the user who entered the macro expression**, not the one who is viewing its result.

The permissions are checked when resolving macros that:

- access an *InfoObject* through another *InfoObject* (i.e. access an encapsulated object): *{% InfoObject1.InnerInfoObject %}*
- access an *InfoObjectCollection*: *{% CurrentUser.Children["cms_category"][0].CategoryName %}*

Security is not only checked for the result, but for each access to any collection.

These macros get signed when the text containing them is saved, which is recognizable by the # character prepended before the closing **%}** character sequence. The security check of such macros is always performed at the time of resolving, not at the time of signing.

```
{% CurrentUser.Children["cms_category"][0].CategoryName #%}
```

It is also possible to prepend the @ character before the closing **%}** character sequence. Such macros don't get signed automatically and are evaluated with public user permissions. The advantage of these macros is that they never exceed the visible string length and can therefore be safely used in fields with value length limit. Please note that this feature is only available in Kentico CMS 6.0 with applied hotfix 6.0.16 or later.

```
{% CurrentDocument.DocumentName @%}
```

Please note

If you experience problems with resolving of such macros, it may be caused by unsuccessful security checks. In these cases, it is recommended to check the event log or the macro debug log, where information about performed macro security checks and their results should be visible.

7.10.8 Registering custom macro methods

Besides the default methods that can be used in data macros, it is also possible to define custom methods that could be used in macro expressions. An example of such a custom method is located in the *~\App_Code\Samples\Classes\CustomMacroMethods.cs* class. The following points summarize what has to be done to implement such a custom method and demonstrates it on the code used in the above mentioned class.

1. Implement all overloads of the custom method.

```
public static string MyMethod(string param1)
{
    return MyMethod(param1, "default");
}

public static string MyMethod(string param1, string param2)
{
    return param1 + " " + param2;
}
```

2. Create a wrapper for the method that is suitable for *MacroResolver*. When creating a wrapper for any custom method, its signature and return type must be exactly the same as shown below.

```
public static object MyMethod(params object[] parameters)
{
    switch (parameters.Length)
    {
        case 1:
            return EcommerceFunctions.MyMethod(ValidationHelper.GetString
(parameters[0], ""));

        case 2:
            return EcommerceFunctions.MyMethod(ValidationHelper.GetString
(parameters[0], ""), ValidationHelper.GetString(parameters[1], ""));

        default:
            throw new NotSupportedException();
    }
}
```

3. Add a method for registration of the new method using *MacroMethods.RegisterMethod*. Its parameters are the following:

- 1. parameter: Method name.
- 2. parameter: Method delegate (wrapper method).
- 3. parameter: Return type of the method.
- 4. parameter: Comment for the method used in macro autocompletion.
- 5. parameter: Formatting string for "human readable" translation of the method call (optional, you do not have to specify).
- 6. parameter: Minimal number of parameters needed to call the method (mimimal overload).
- 7. parameter: Parameter definition in format *{{name, type, comment}, {name, type, comment}, ...}*.
- 8. parameter: A list of special parameters needed to be supplied by resolver (these parameters are automatically passed by MacroResolver as the first parameters to the wrapper method).
- 9. parameter: List of types for which the method is applicable (set to *null* for all types to be allowed). When not specified, the method will be applicable to the type of its first parameter.
- 10. parameter: Code snippet which is used in macro autocompletion when TAB is pressed (to determine cursor position, use the pipe character |).

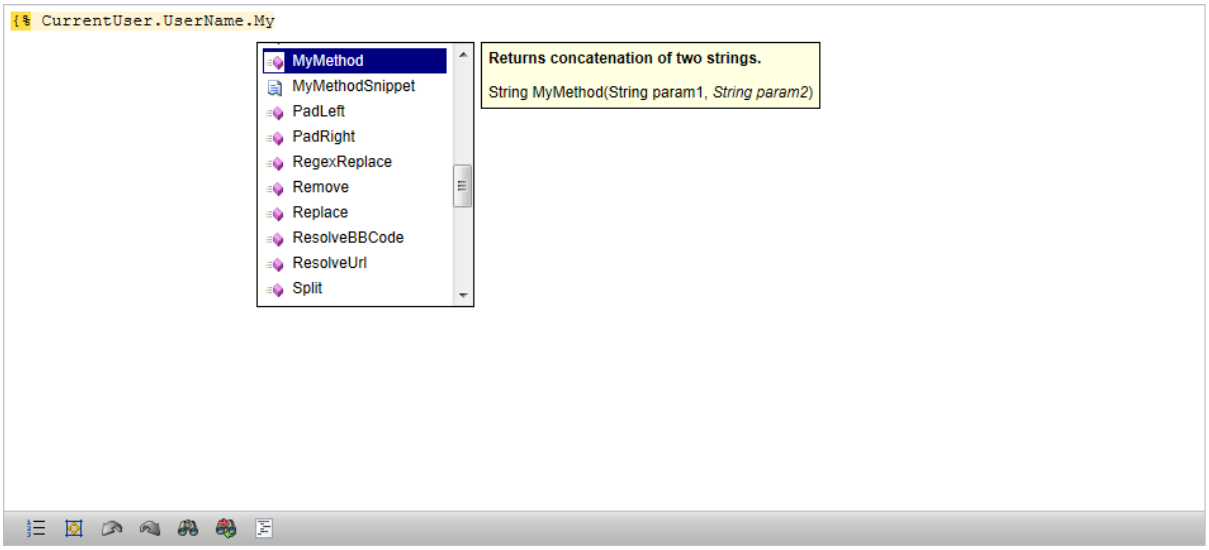
As you can see in the code below, the method is registered twice. At first, it is registered as a standard method with a comment and parameter descriptions for use in macro autocompletion. The second registration registers the method with the *MyMethod(ToLower(CurrentDocument.DocumentName), |);* code snippet. The | character indicates that the cursor will be at the position of the second parameter when the code snippet is inserted.

```
public static void RegisterMethods()
{
    MacroMethods.RegisterMethod("MyMethod", MyMethod, typeof(string), "Returns
concatenation of two strings.", null, 1, new object[,] { { "param1", typeof(string
), "First string to concatenate." }, { "param2", typeof(string), "Second string to
concatenate." } }, null);

    MacroMethods.RegisterMethod("MyMethodSnippet", MyMethod, typeof(string),
"Calls MyMethod on lower case current user name.", null, 0, null, null, new List<
Type>() { typeof(string) }, "MyMethod(ToLower(CurrentDocument.DocumentName), |);",
false);
}
```

4. Finally, ensure execution of the *RegisterMethods* method implemented in the previous step in the *Init ()* method in the *CMSModuleLoader* partial class. In case of the mentioned example, there is a commented line ensuring this in the *Init()* method in *~\App_Code\Samples\Modules\SampleMacroModule.cs*.

```
public override void Init()
{
    // -- Custom macro methods
    CustomMacroMethods.RegisterMethods();
}
```

5. The method is now registered and ready to be used. To verify this, try e.g. creating a new transformation and using a macro in its text. The new method should be offered in the autocompletion box if you try to apply it to a string parameter. As you can see in the screenshot below, the second parameter in the method signature — *String param2* — is displayed in italic letters. This means that the method is overloaded and this parameter is optional and only present in the overload. You should also see the *MyMethodSnippet* method below the one mentioned above. Selecting this one inserts the defined code snippet.



7.10.9 Macro result caching

It is possible to cache macro results so that certain expressions don't have to be evaluated repeatedly. This is possible using 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, while only the first parameter is mandatory — the rest of the parameters is optional. The parameters determine the following:

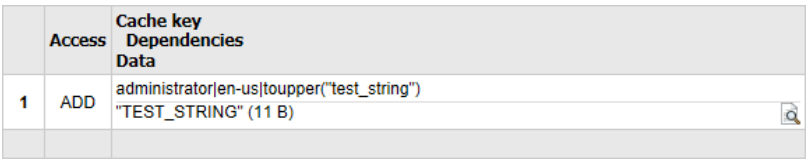
- string macroExpression: Macro expression (without encapsulating character sequences, e.g. *{% %}*) to be evaluated and cached.
- *Int32 cacheMinutes*: Number of minutes for which the value will be cached.

- *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.

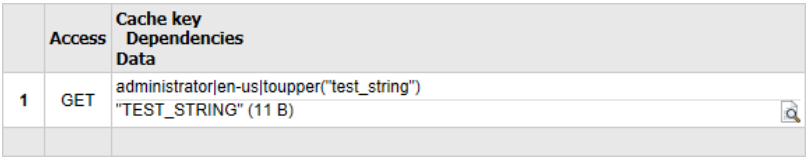
```
{% Cache("test_string".ToUpper()) %}
```

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.

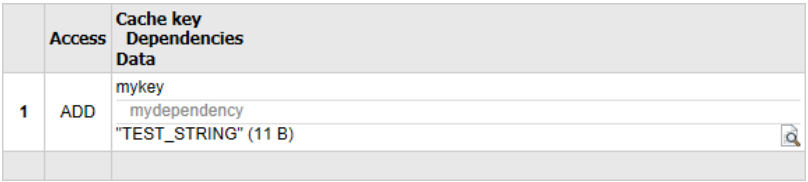


	Access	Cache key Dependencies Data
1	GET	administrator|en-us|toupper("test_string") "TEST_STRING" (11 B)

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)

7.10.10 Resolving macros using API

There is an easy way to resolve macros in .NET code. To resolve all the macros (recommended), use the following static method:

```
string CMS.CMSHelper.CMSContext.CurrentResolver.ResolveMacros(string inputText)
```

To resolve just the localization macros, use another static method:

```
string CMS.GlobalHelper.ResHelper.LocalizeString(string inputText)
```

Creating a custom resolver

In the following example, you can see how a custom macro resolver can be created, loaded with data and used to resolve a text containing macros.

```
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 the text
    resolvedText = resolver.ResolveMacros("Name of my user is (in upper case): {%
MyUser.UserName.ToUpper() %}. First eight characters of simple val#1: {%
MySimpleValue.Substring(0, 8) %}");
}
```

7.11 Membership, permissions and security

7.11.1 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:

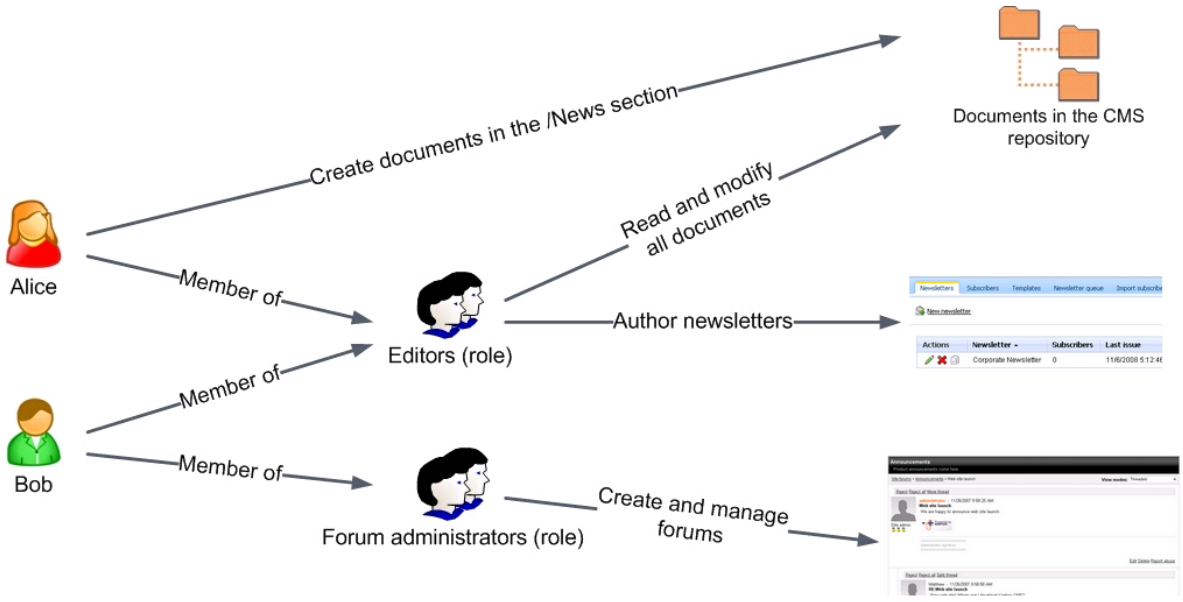
- users (shared among websites)
- roles (can be defined for particular websites or globally for all sites in the system)
- memberships (collections of roles that can be assigned to users)
- module permissions
- document permissions
- UI personalization

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.

7.11.2 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 the user account:

- **Is editor** - the user can access the **CMS Desk** interface. This attribute doesn't grant any particular permissions — it only differentiates between site editors and "registered users" who can only access the live website and its secured areas and thus provides an extra security layer. The user can access CMS Desk for all sites to which they are assigned on the **Sites** tab
- **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

Global administrators

Global administrators are the only users who can 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** – 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 the **New user** link 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 authorized to sign in to CMS Desk. It's used to differentiate users who are only allowed to visit member areas of the website from content editors who can use the CMS Desk user interface. This provides an extra security layer.
- **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 Authentication -> Password settings 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 authorized to sign in to CMS Desk. It is used to differentiate users who are only allowed to visit member areas of the website from content editors who can use the CMS Desk user interface. This provides an extra security layer.
- **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 agent of the user's last

logon.

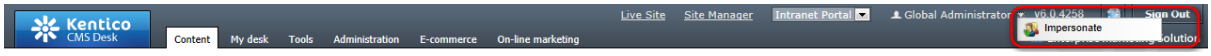
- **Starting alias path** - the starting alias path of the content tree in CMS Desk -> Content; if you specify this value, the user is not allowed to browse other sections of the website in the content tree; please note that this feature is only intended for better usability and it doesn't ensure security control - if you need to establish access rights for a given user, grant appropriate document permissions (Properties -> Security) to them

Impersonation

Global administrators can also see the **Log in as this user** link at the top of this tab. By clicking this link, the administrator gets logged in as the currently edited user and gets redirected depending on the type of user:

- **Global administrator** - if you log in as some other global administrator, you will stay on the **General** tab
- **Editor** - if you log in as some editor (a user with the **Is editor** option enabled), you will be redirected to CMS Desk
- **Standard user** - if you log in as a standard user, you will be redirected to the title page of the live site

User impersonation may also be performed by global administrators 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 is displayed where the specific user to be impersonated can be selected from a list of all users in the system.

In **Site Manager -> Administration -> Event log**, any actions carried out while impersonating a user will be logged under the user name in format *<user name> (<original user name>)* where the original user is the administrator using the impersonate function.

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

- **Waiting for approval** - if checked, the user 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

- **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** link, 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 page, you can manage the currently edited user's friends.

Subscriptions

On this tab, you can manage the user's subscriptions to newsletters and notifications about new blog posts and message board messages.

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

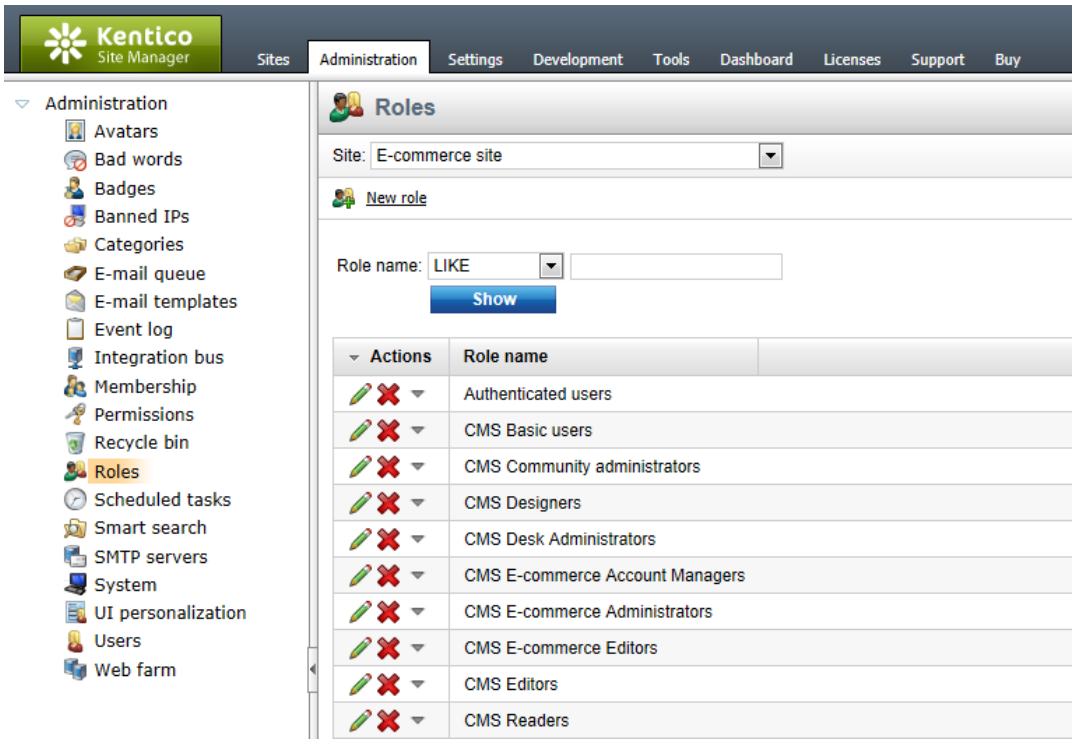
Memberships

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.

7.11.3 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 the **New role** link. 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

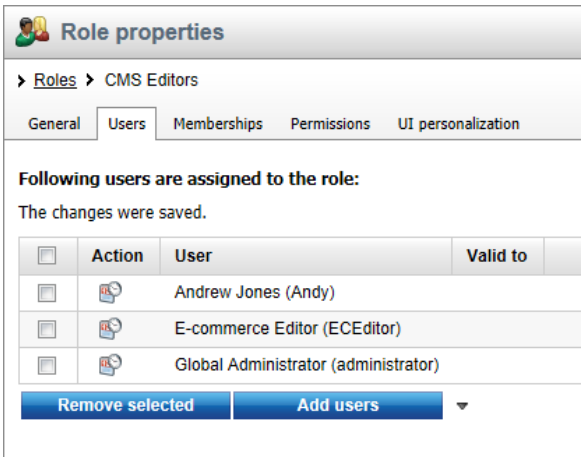
There are four tabs available when editing () a role:

General

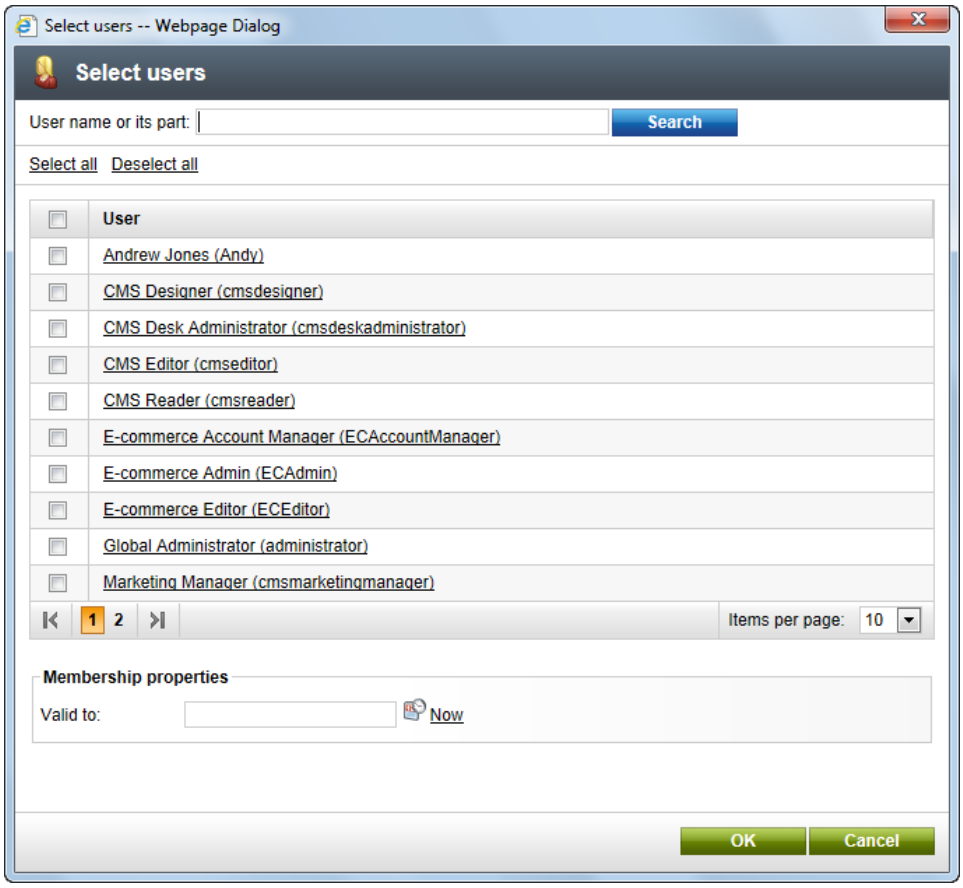
On this tab you can edit the same properties that were specified when the role was created.

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.

7.11.4 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;
    }
}
```

7.11.5 Permissions

7.11.5.1 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

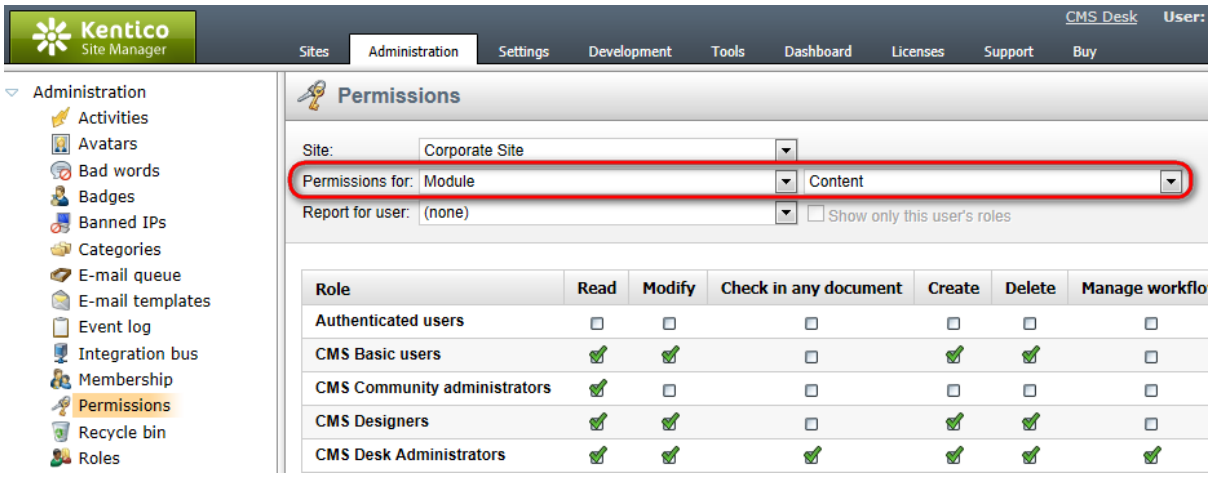
currently edited website, while in **Site Manager**, you can configure permissions for all sites in the system or for global roles by selecting a site from the **Site** drop-down.

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:

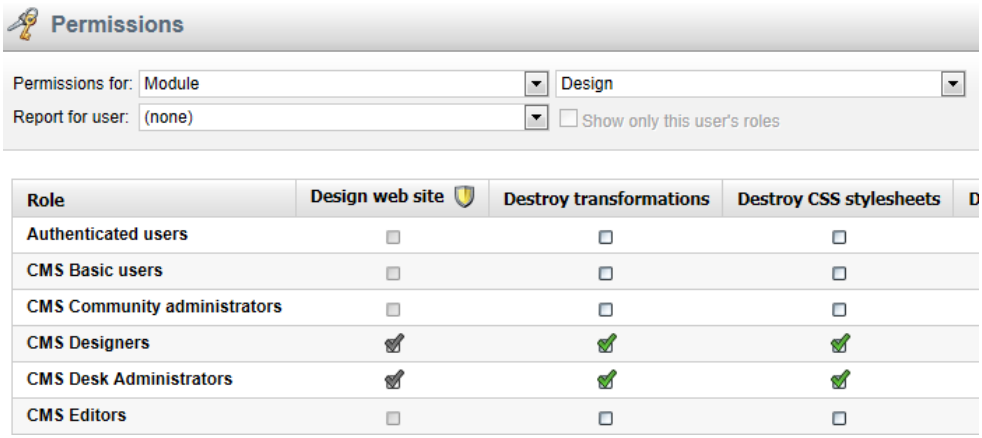
- ☑ - the permission is granted to the role.
- ☐ - the permission is not granted to the role.



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:

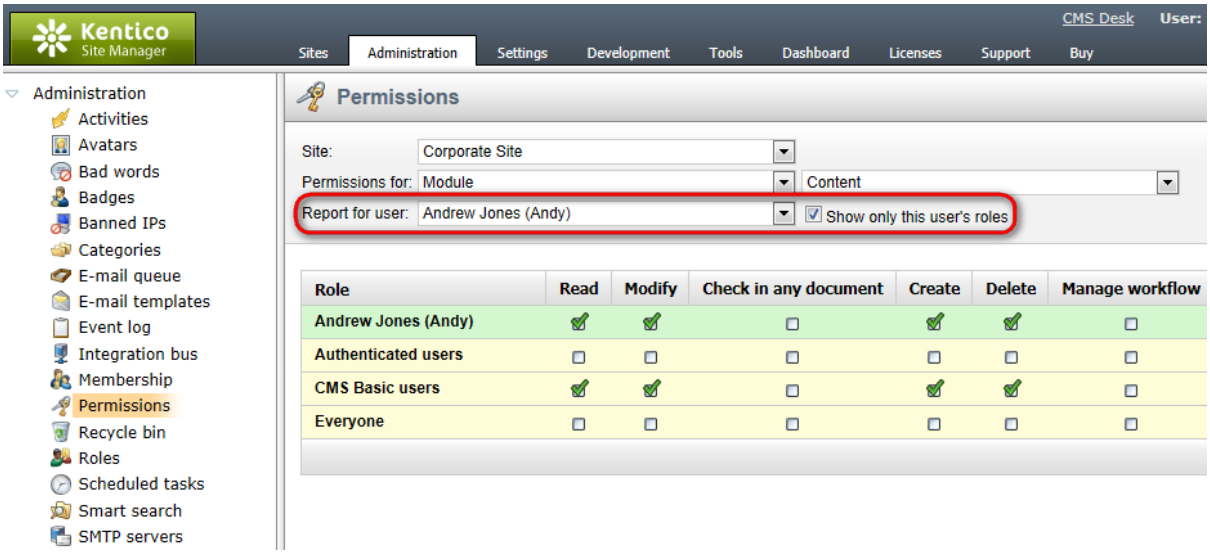
- ☑ - the permission is granted to the role, and only a global administrator can change it.
- ☐ - the permission is not granted to the role, and only a global administrator can change it.

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

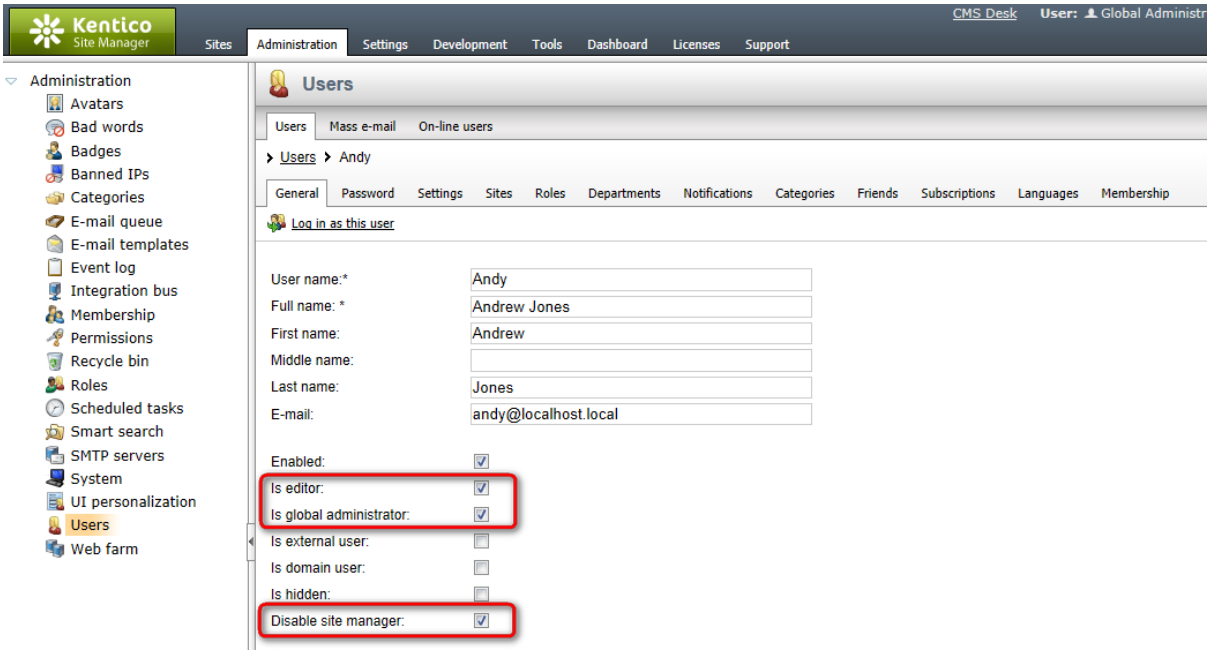
When editing (✎) a user in **Administration -> Users**, you can enable the following two options. These options have impact on permission checking and provide an extra security layer:

- **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 the CMS Desk interface. This attribute doesn't implicate any

particular permissions - it only differentiates site editors from “registered users” who only access the live website and its secured areas. The user can access CMS Desk for 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.



7.11.5.2 Document permissions

7.11.5.2.1 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.

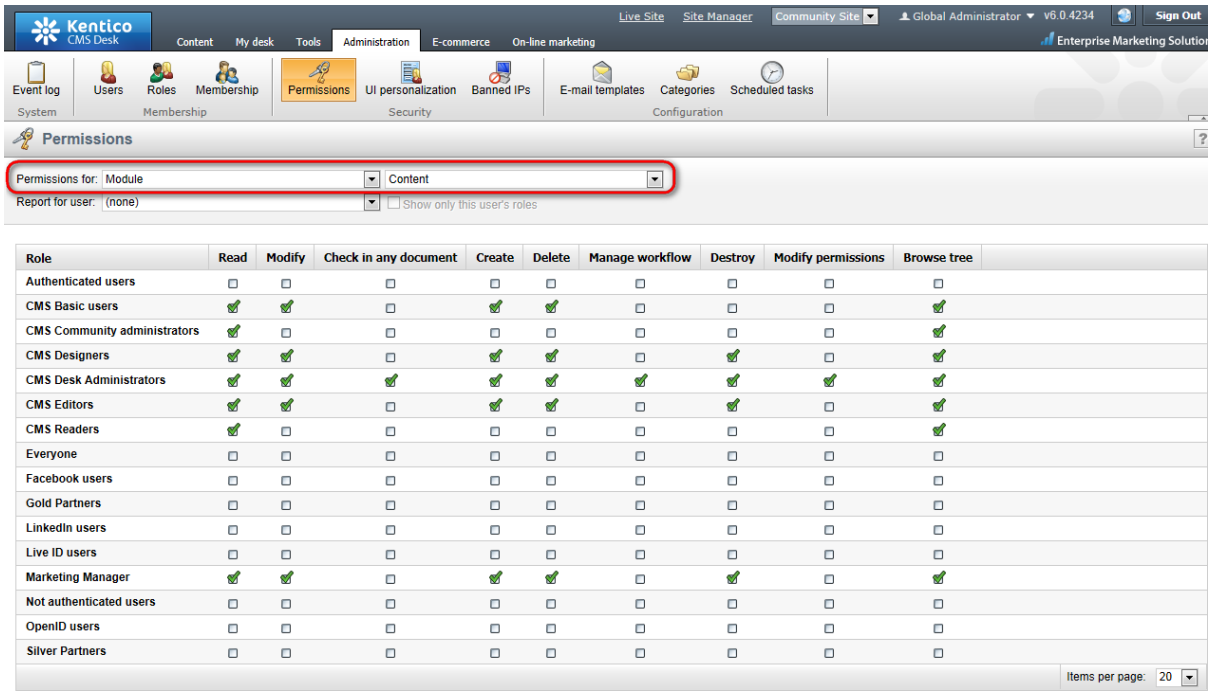
There is also one special setting that allows hiding of documents in the content tree depending on document permissions granted to the current user. See the Hiding documents in the content tree based on permissions topic for more information on this possibility.

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.

7.11.5.2.2 Permissions for all content

In **Site Manager -> Administration -> Permissions**, there is 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:

Read	Allows members of the role to view any document in the content tree.
Modify	Allows members of the role to modify any document in the content tree.
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.
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**.
Browse tree	Allows members of the role to see all documents in the content tree (not their actual content — that is ensured by the *Read* permission). If not assigned, the **Content** tab may not be displayed (unless the role has the **Read** permission for the *CMS.Root* document type or for the *Root* document on document-level).
Design website	Allows members of the role to access and use the **Design** tab. Please note: although users can make changes only to the current website, changes to page templates may affect other websites if a page template shared among multiple websites is modified.

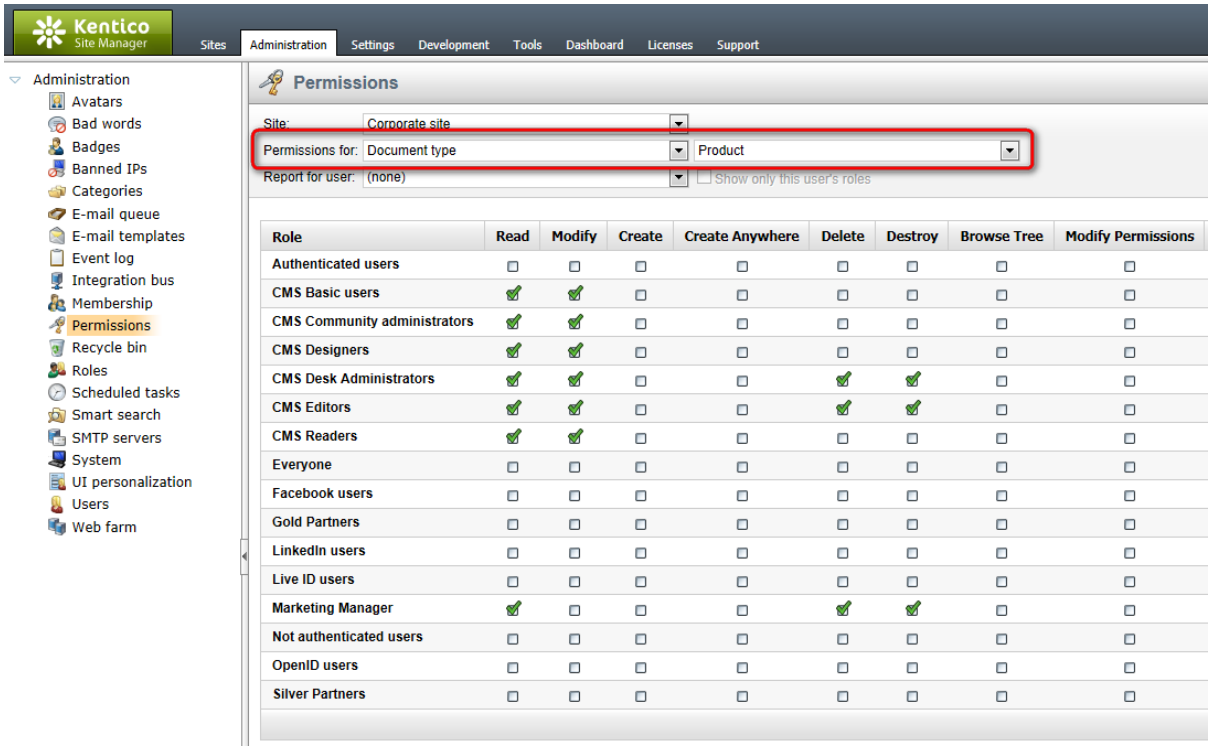


7.11.5.2.3 Document type permissions

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**.



7.11.5.2.4 Document-level permissions

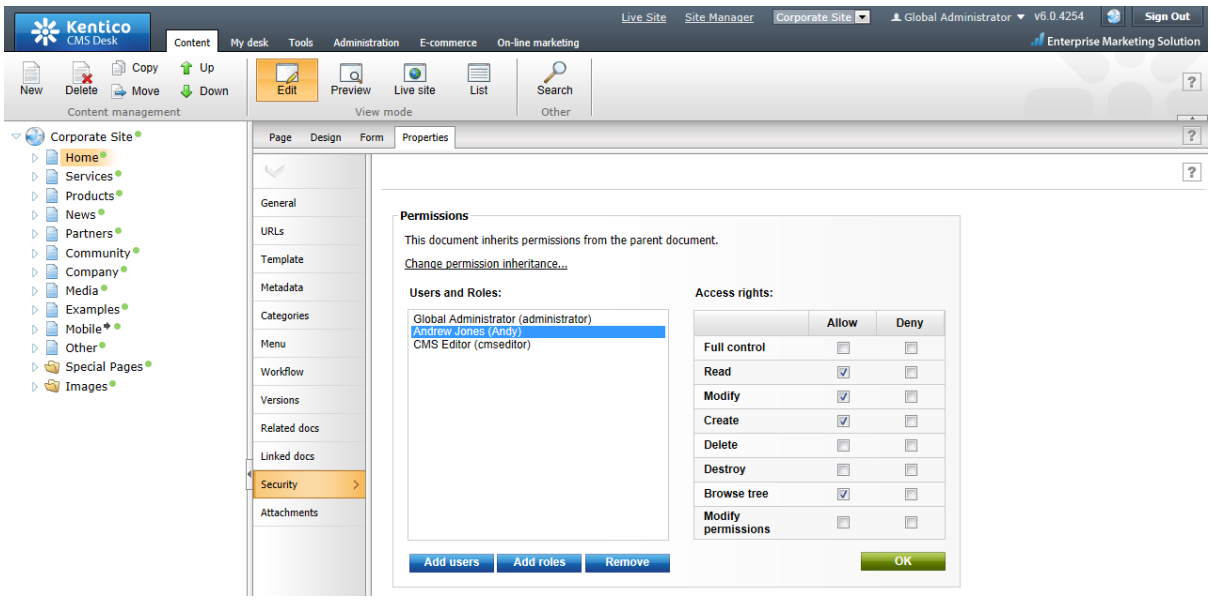
You can manage document-level permissions (i.e. permissions for a particular document or a particular website section) in **CMS Desk -> Content -> Edit -> Properties -> Security**. Select the appropriate user or role in the left box. If the user or role is not available in the box, you may need to add 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:

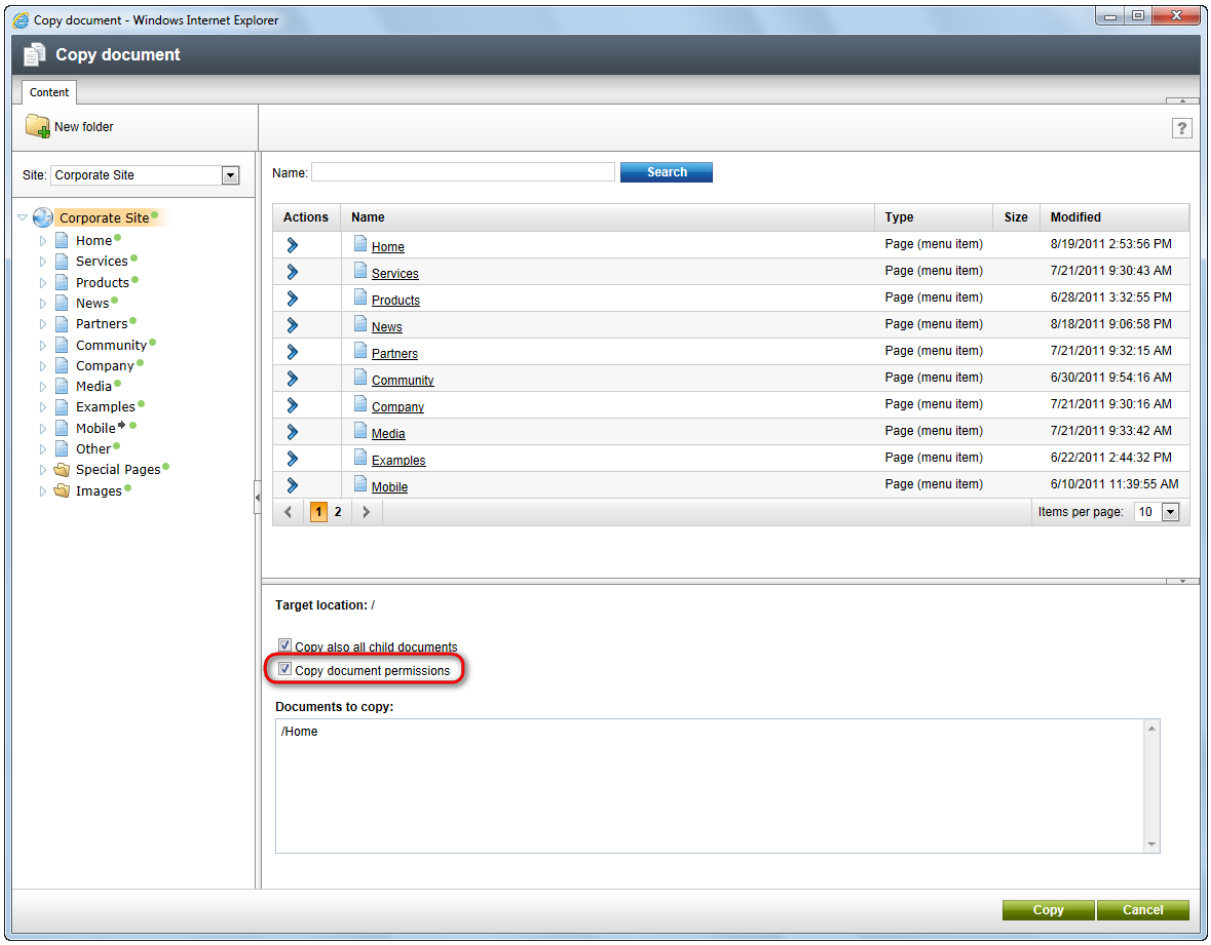
JohnS	**Marketing manager** John can manage all content.	Grant the **Full control** permission on the root to the user or grant permissions for the **CMS Content** module to some of this user's roles.

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.

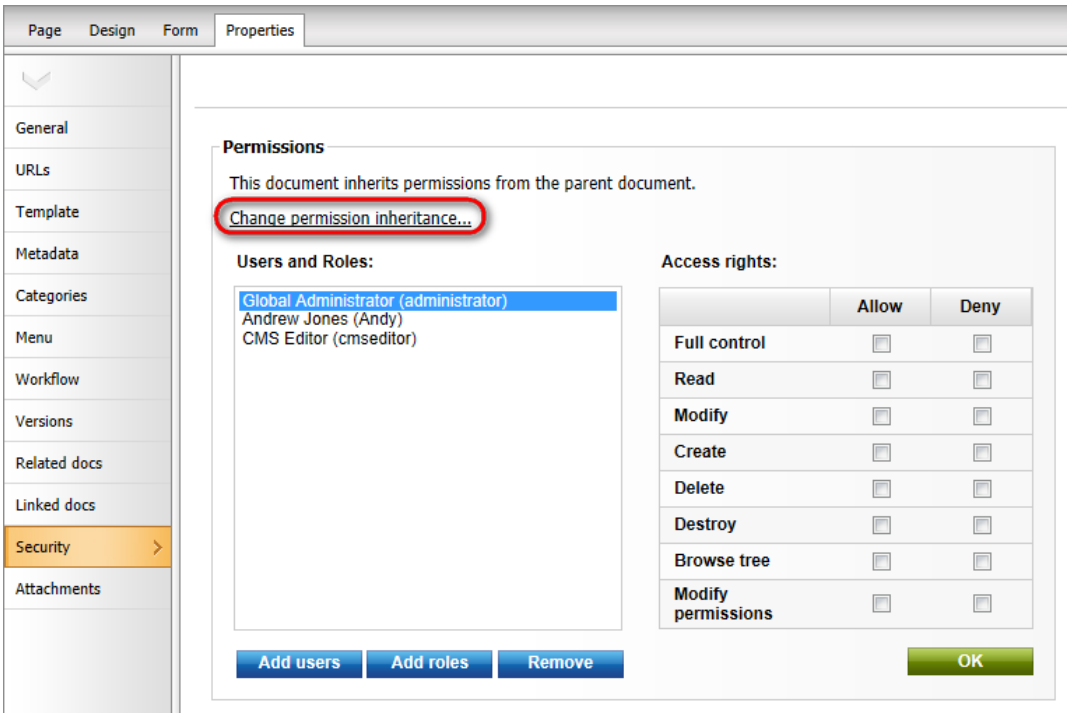
Copying permissions along with documents

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.



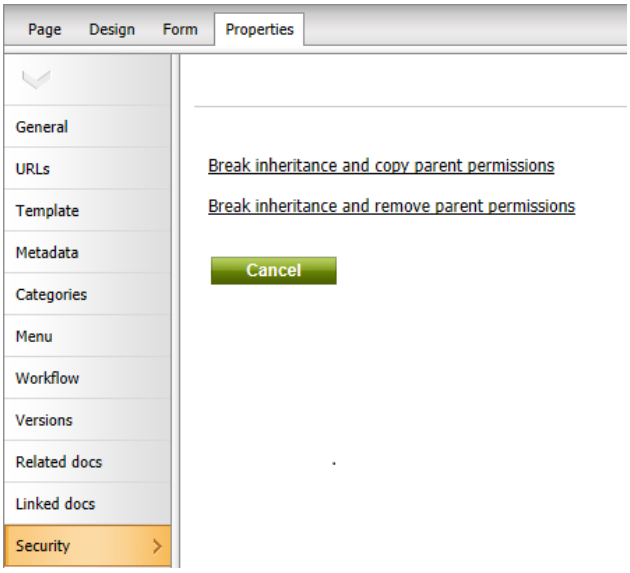
Changing permission inheritance

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.



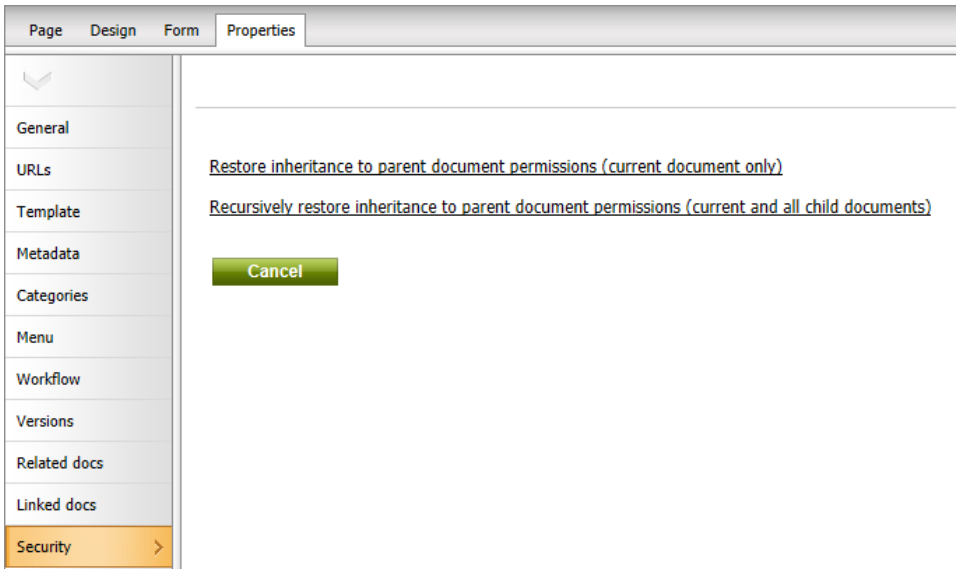
If permissions are inherited by the current document, the following two options will be offered:

- **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:

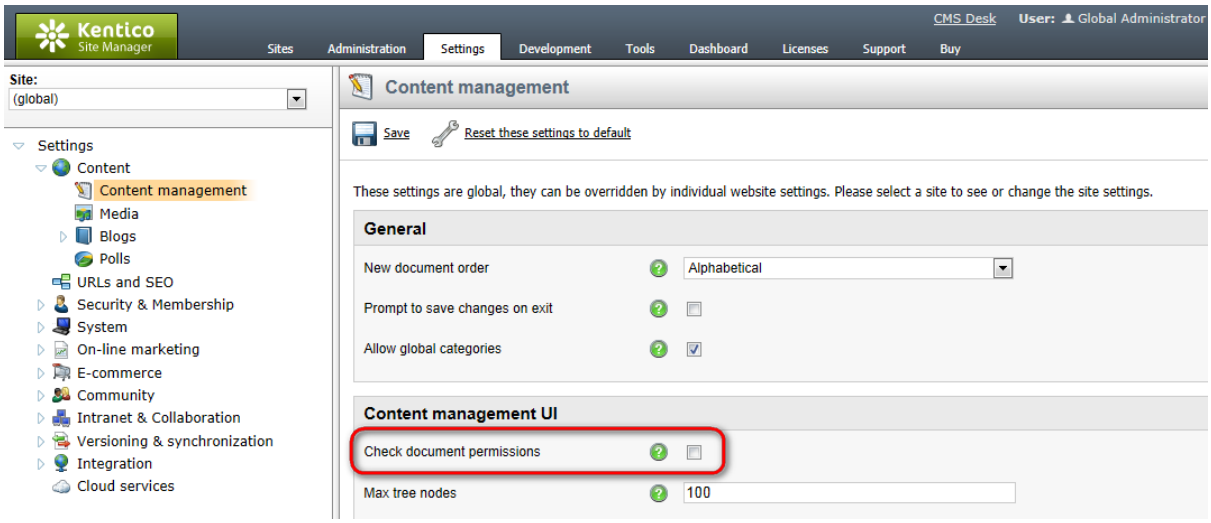
- **Restore inheritance to parent document permissions (current document only)** - makes the current document inherit permissions of the parent document.
- **Recursively restore inheritance to parent document permissions (current and all child documents)** - makes the current document and all its child documents inherit permissions of the parent document, while only documents which do not inherit parent permissions are affected by this action.



7.11.5.2.5 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.



7.11.5.2.6 Document permisssions internals and API

7.11.5.2.6.1 Overview

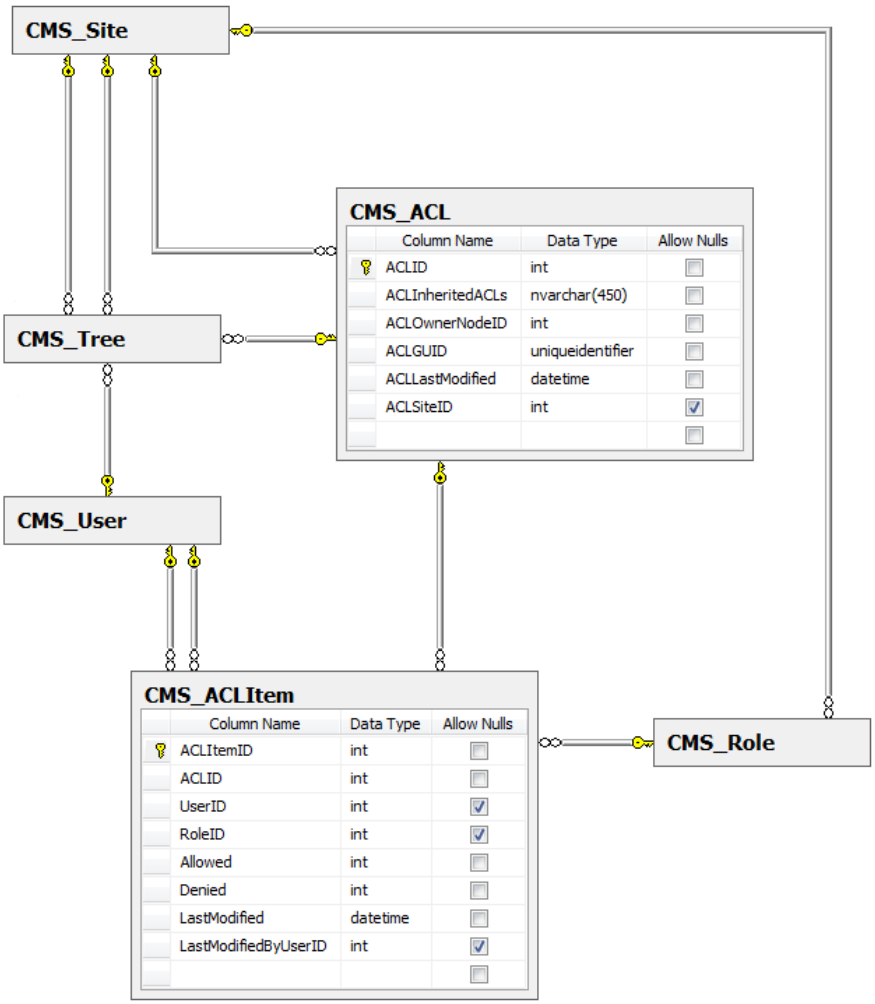
In this chapter, you will learn which database tables and API classes are used for managing document-level permissions. You will also see the most common API examples.

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.

7.11.5.2.6.2 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.



7.11.5.2.6.3 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.TreeEngine** namespace.

CMS_ACL table API:

- **AclInfo** - represents one document level permission matrix.
- **AclProvider** - provides management functionality for document level permission.

7.11.5.2.6.4 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.TreeEngine;
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;
}
```

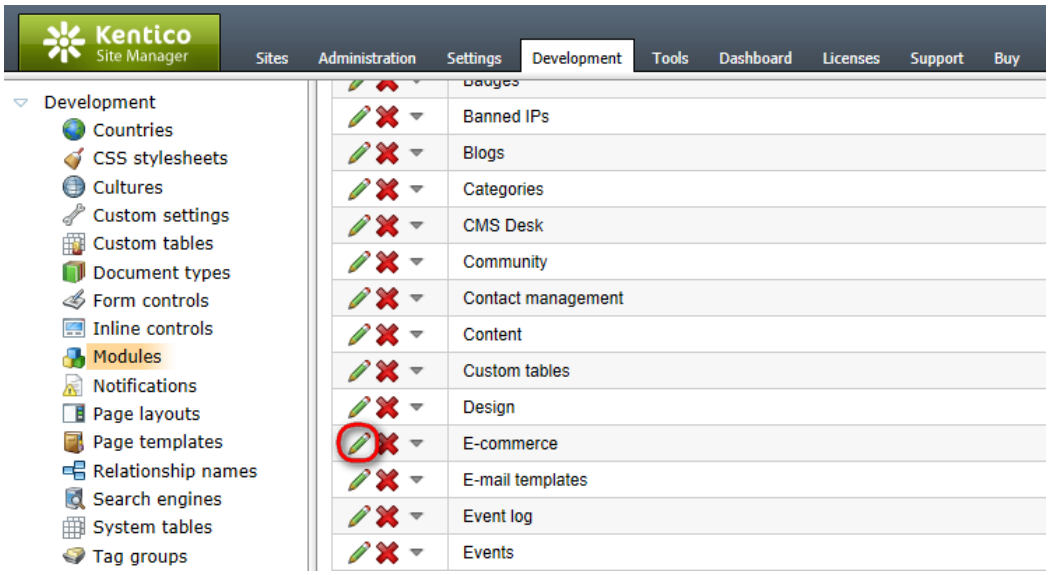
The following example gets multiple document into a DataSet, filters the documents depending on if the **Modify permissions** permission was granted to the **CMS Editor** user on their document level, and breaks permission inheritance of the filtered documents.

```
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;
}
```

7.11.5.3 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**.



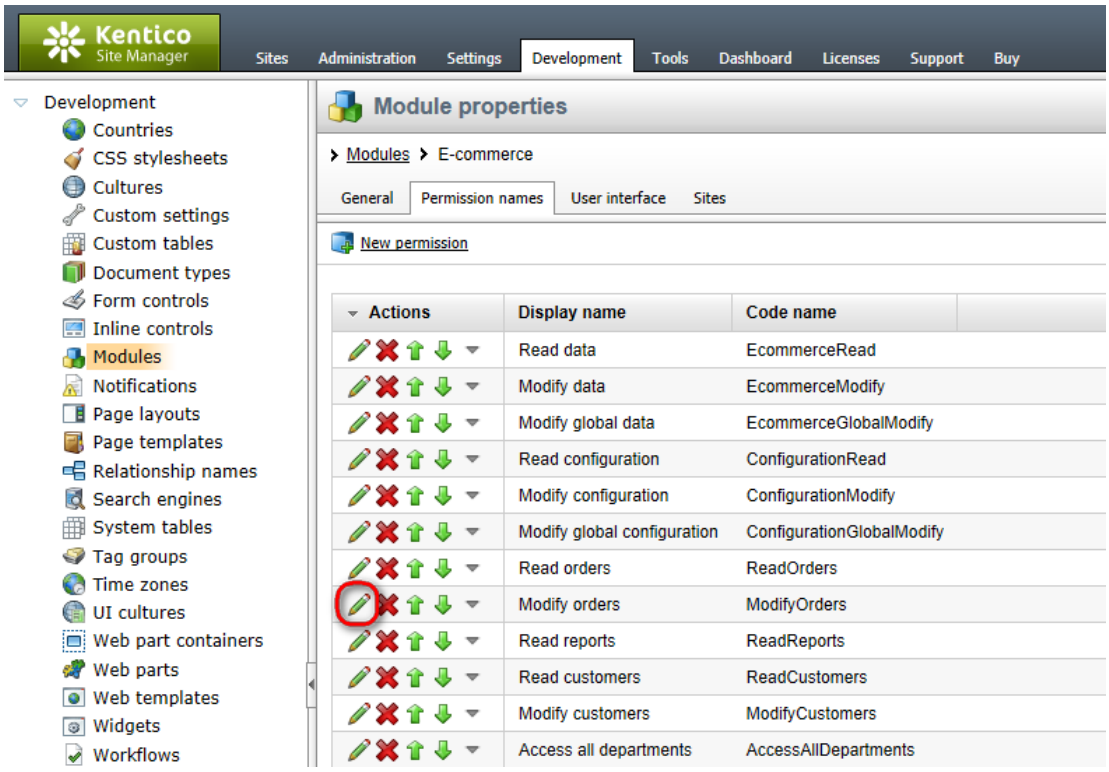
In the module's administration UI, switch to the **Permission names** tab. Here, you can see a list of all permissions defined for the module. They are listed in the same order that will be used in the permission matrix (top-down order here equals to left-right order in the matrix). The **Move up** ( ) and **Move down** ( ) icons can be used to re-order permissions according to your needs.

Please note

New permissions can be created by clicking the **New permission** link 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.

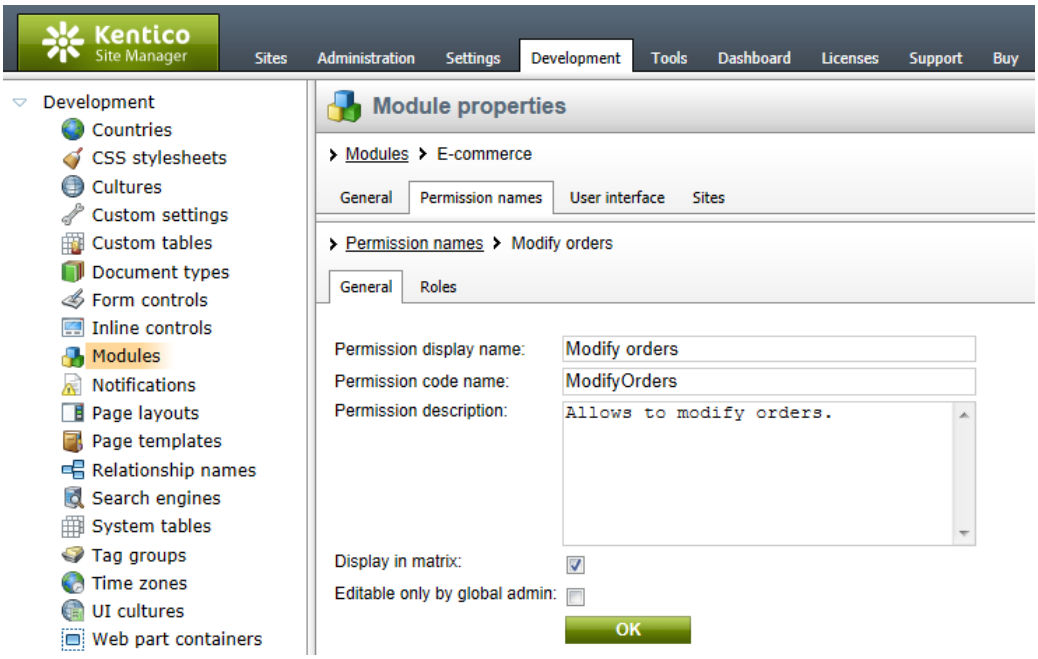
An example of handling permissions in custom module code can be found in the Modules -> Developing custom modules topic.

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.

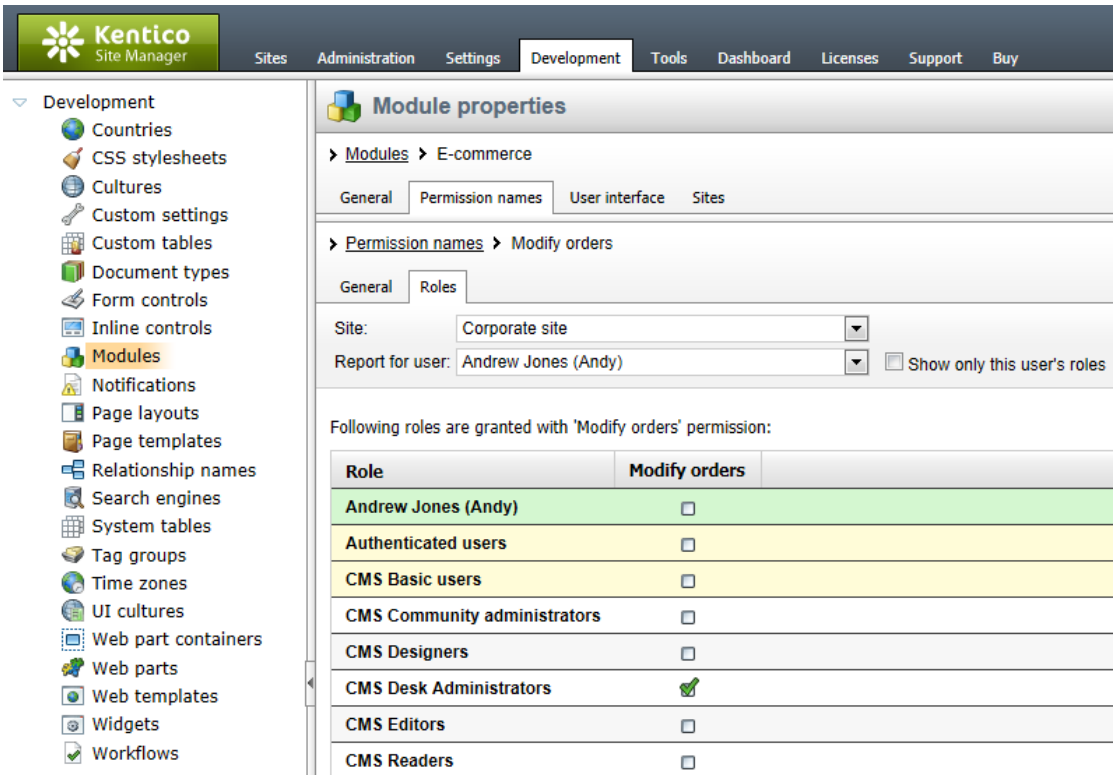
- **Permission display name** - name of the permission displayed in Kentico CMS UI.
- **Permission code name** - name of the permission used in website code.
- **Permission description** - text providing more details about the permission (typically where and when it is checked).
- **Display in matrix** - 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.



7.11.6 UI personalization

7.11.6.1 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.

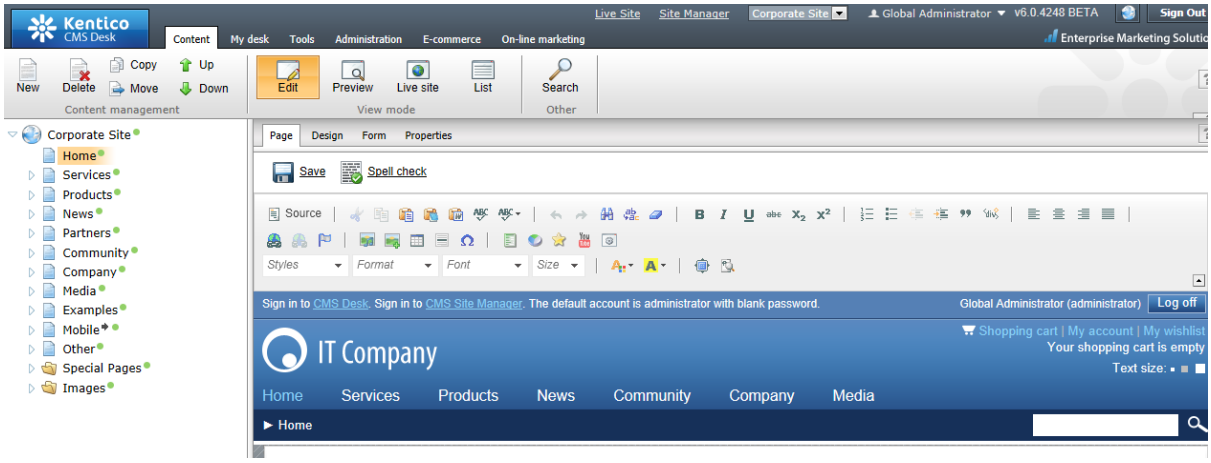
3. With some basic information from the Quick example, you can get deeper knowledge of the terminology, concept and internals in the How it works chapter.

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.

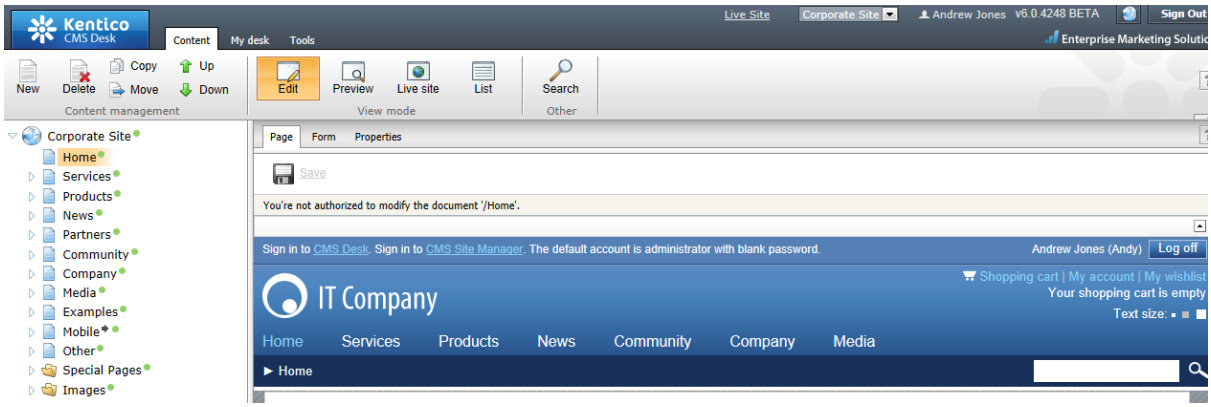
7.11.6.2 Quick example

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.

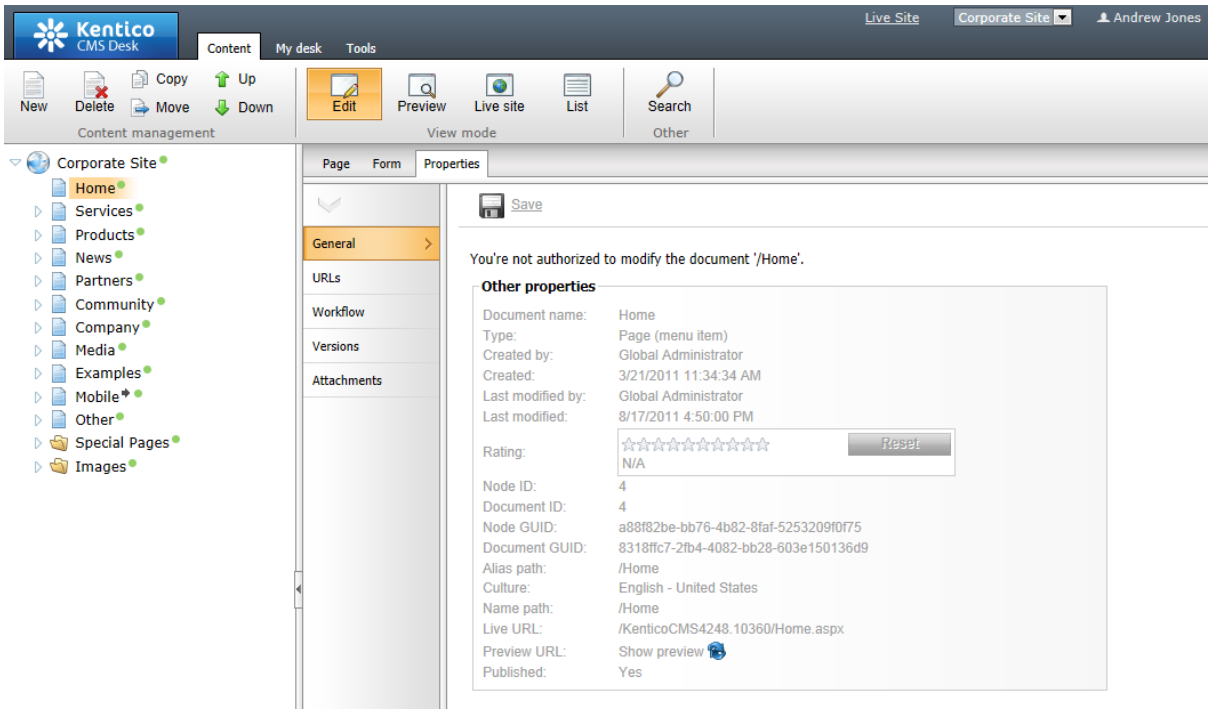


Now log out and try logging in as **Andrew Jones** (login *andy* with blank password). Andy is a member of the **CMS Basic users** role only. This is a role which was added to Kentico CMS to demonstrate the capabilities of UI personalization and its members see a simplified user interface.

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.

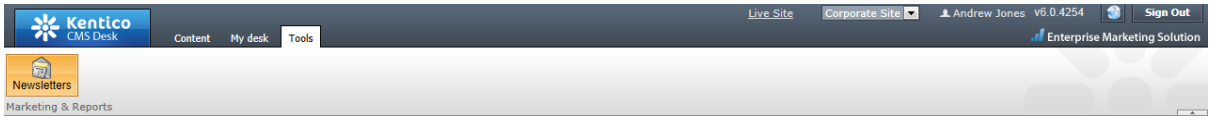


If you switch to the **Properties** tab, you can notice the limited number of items in the left menu. Particular sections accessible via the menu also do not contain all the options available for administrators.



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.

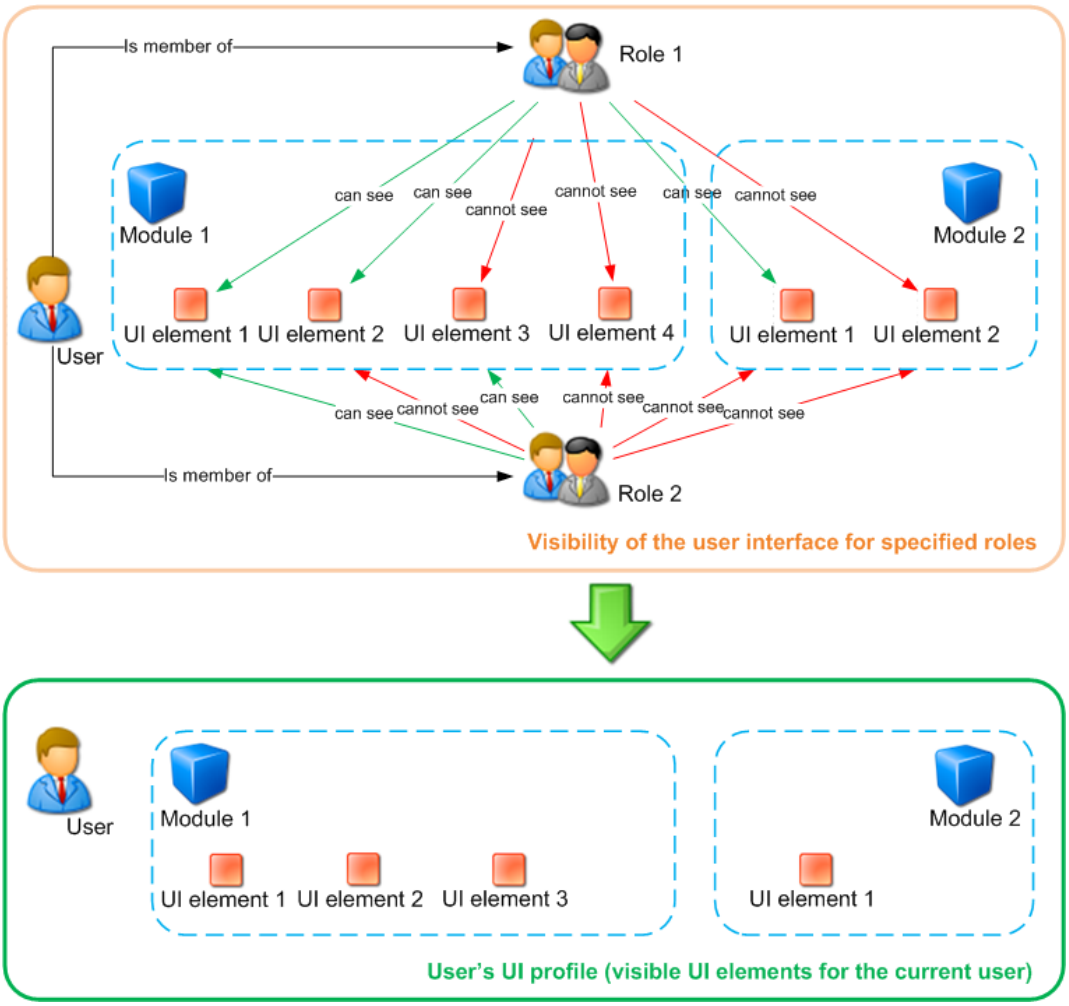
7.11.6.3 How it works

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.



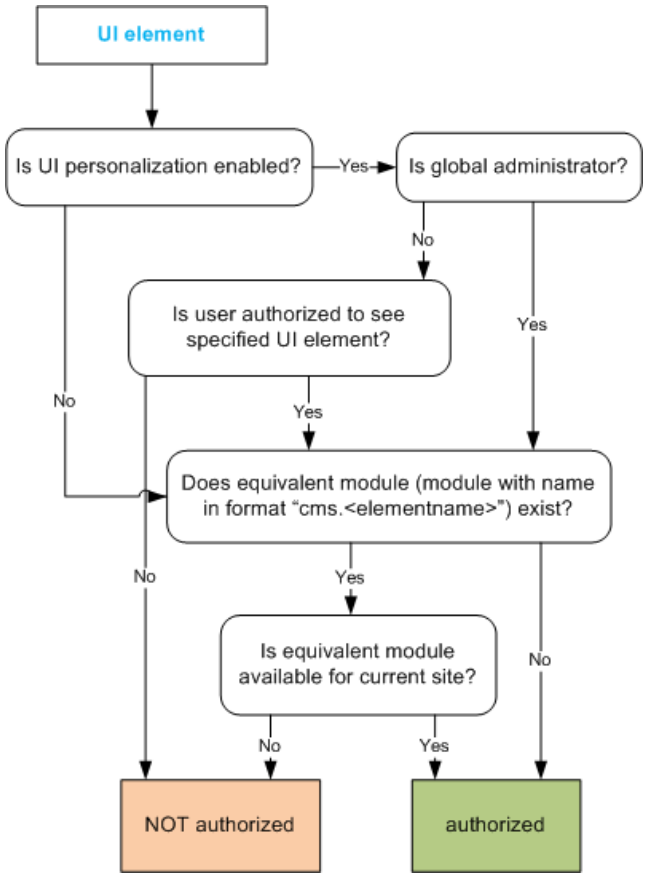
Please note: The roles assigned to a user determine which UI profile will be displayed. A user who is assigned to site-specific roles will have a **site-related** UI profile. This means that a user can see a certain personalized UI when editing one site and a completely different UI when editing another 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

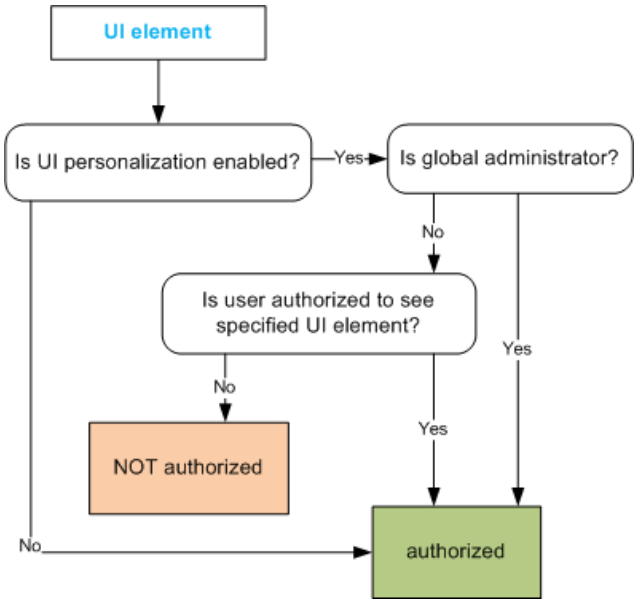
The diagram below shows the process of authorization when deciding if a single UI element should be

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, ...)
- 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.

7.11.6.4 Personalizable parts of CMS Desk

7.11.6.4.1 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 rather than a simplified UI.

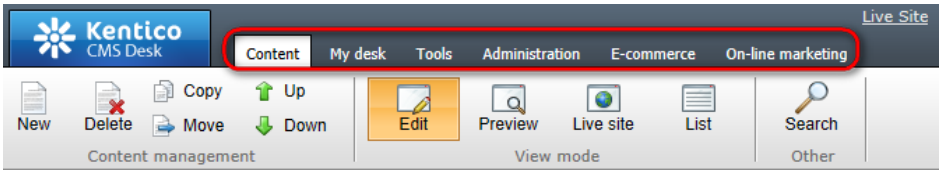
As for CMS Desk, you can personalize the UI elements in the following parts of CMS Desk. Module names in parentheses show which module represents these parts of CMS Desk:

- 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)

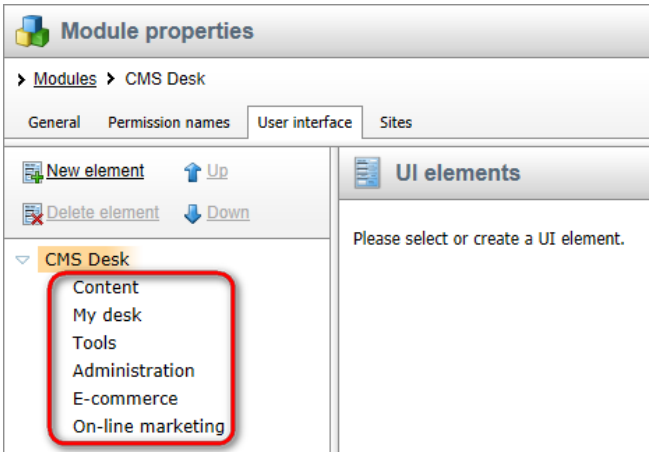
Click particular sections to learn more.

7.11.6.4.2 CMS Desk main tabs

You can hide each of the four main tabs - **Content**, **My desk**, **Tools**, **Administration**, **E-commerce** and **On-line marketing** - as highlighted in this screenshot:



As you can see, the **CMS Desk** module has UI elements with exactly the same names as the tabs above:

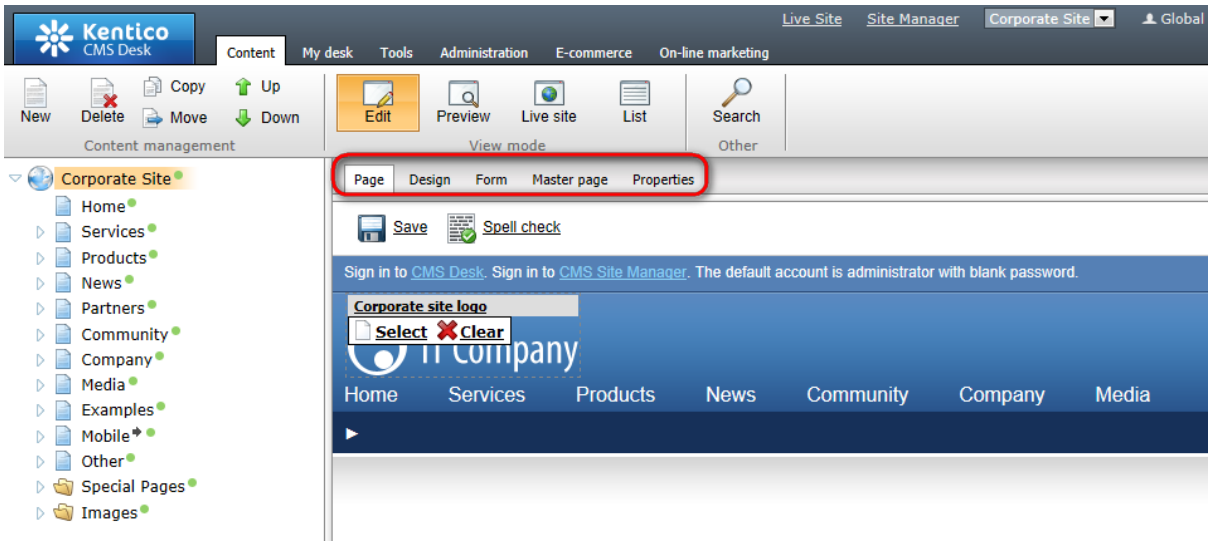


7.11.6.4.3 CMS Desk -> Content tab

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.

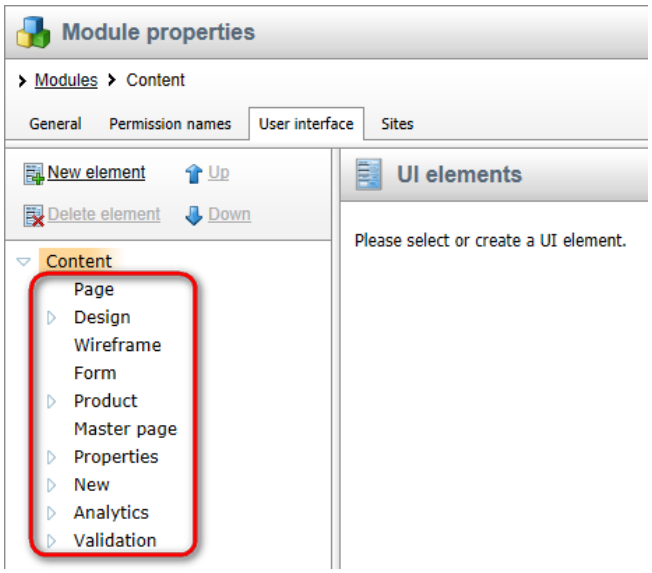
Edit mode tabs

You can hide any of the main tabs displayed in **Edit** mode as highlighted in the screenshot below.



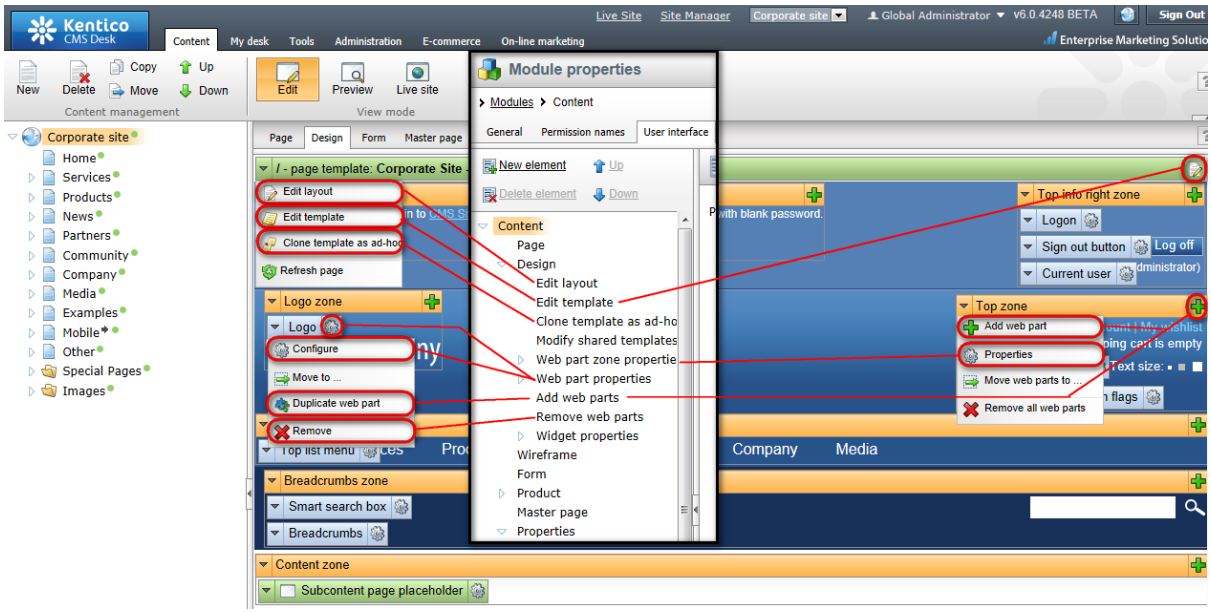
These main tabs are represented by the first-level UI elements of the **Content** module as highlighted below.

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.



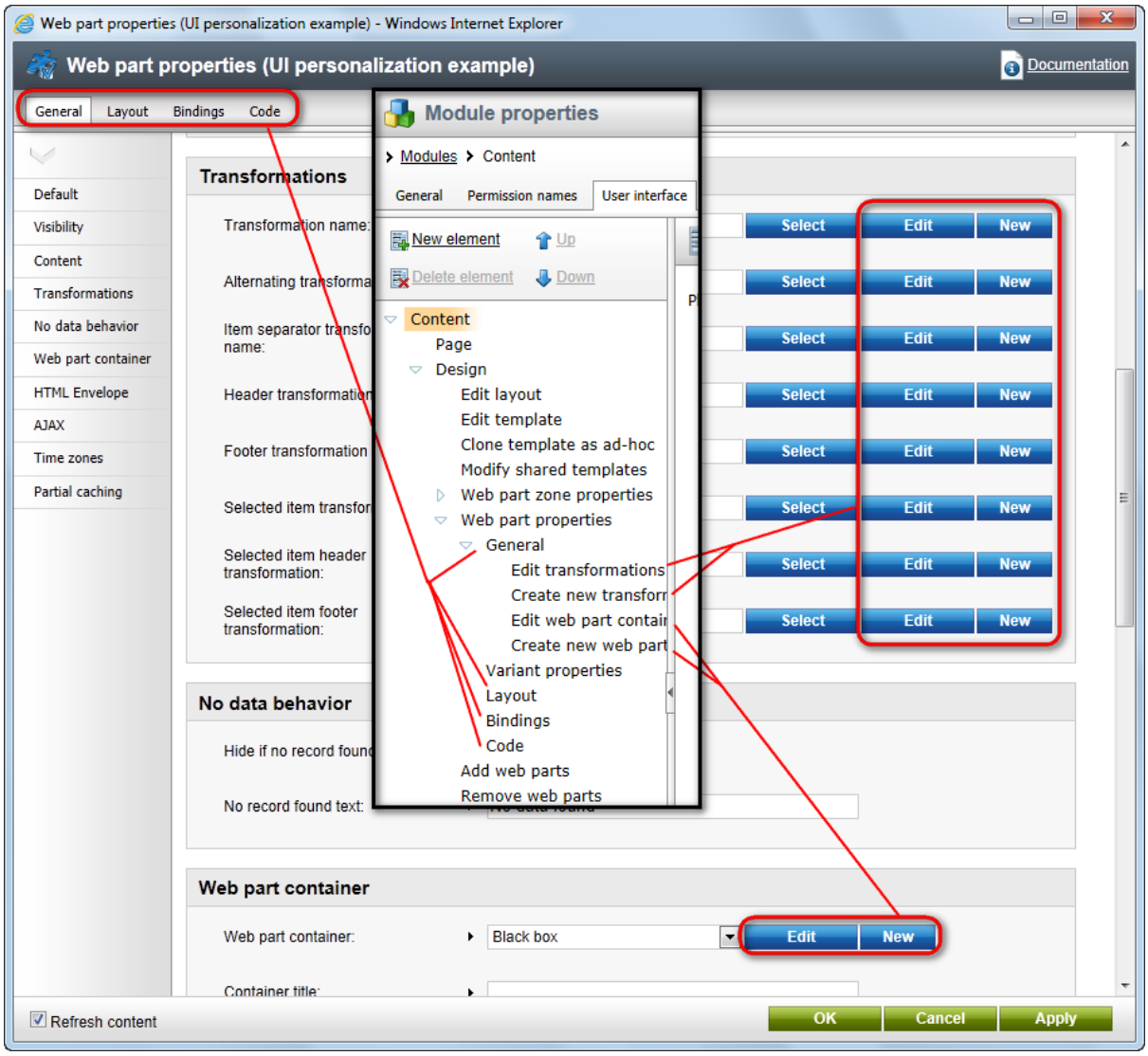
Design tab

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.



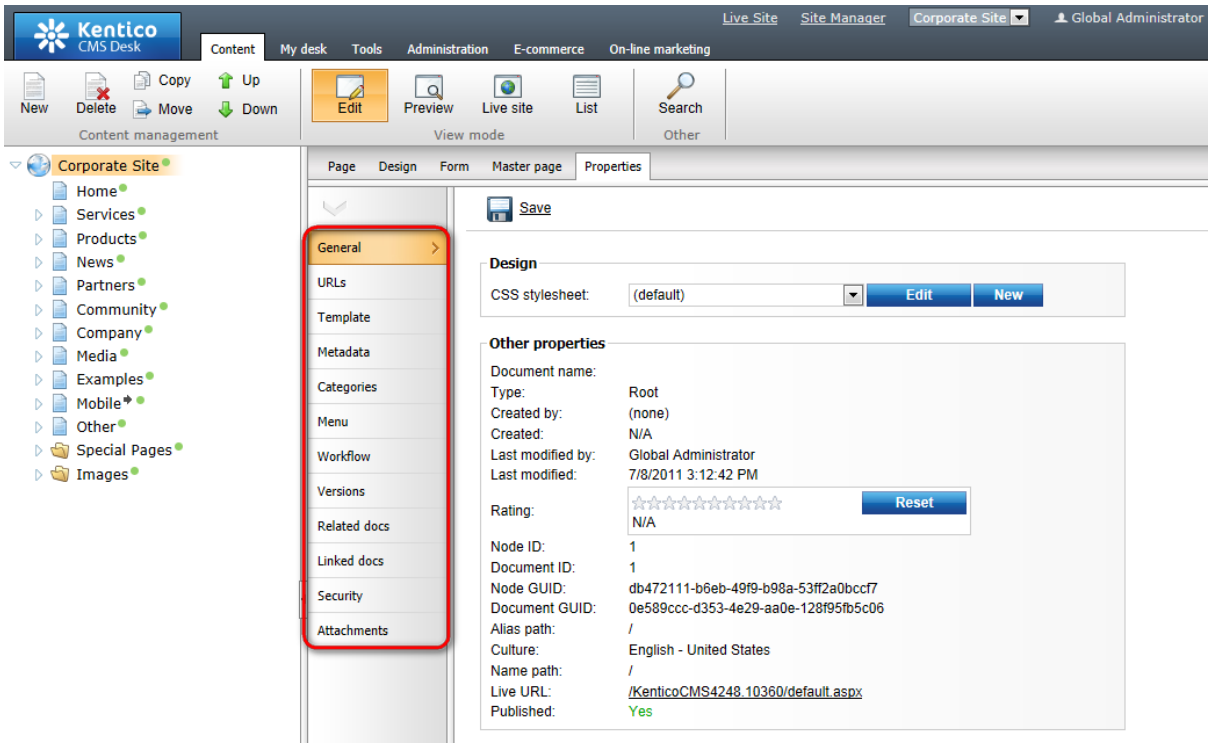
Properties tab

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**, **Menu** and **Security** pages. More information can be found on this page.

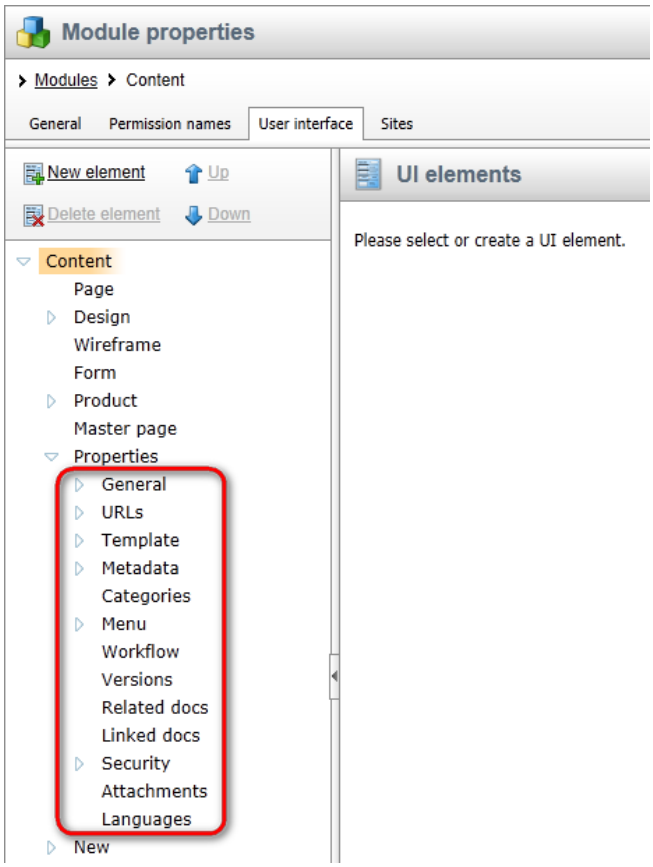
7.11.6.4.4 CMS Desk -> Content -> Edit -> Properites tab

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.



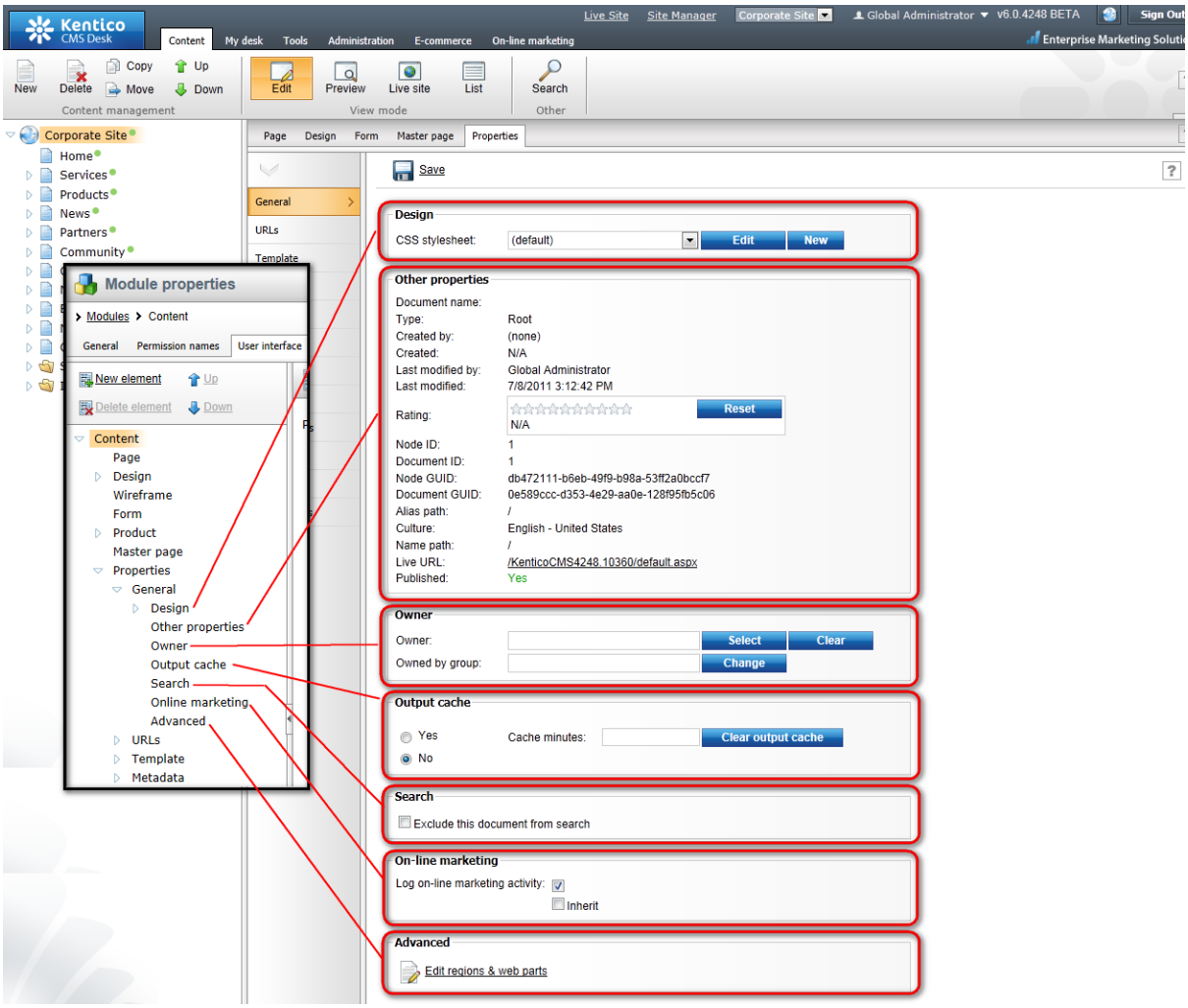
In the **Content** module, you can find the relevant UI elements under the **Properties** node. These elements have the same names as the names of particular menu items.



You can also expand the **General**, **URLs**, **Template**, **Metadata**, **Menu** 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:

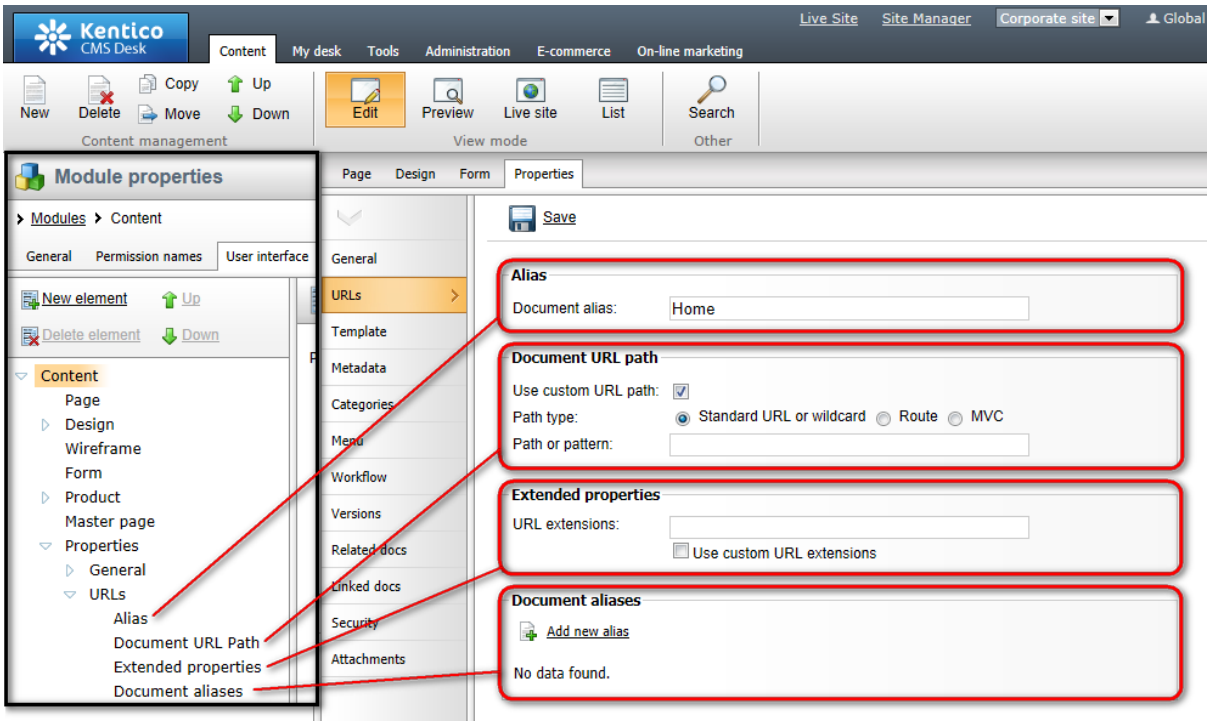
General tab

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:



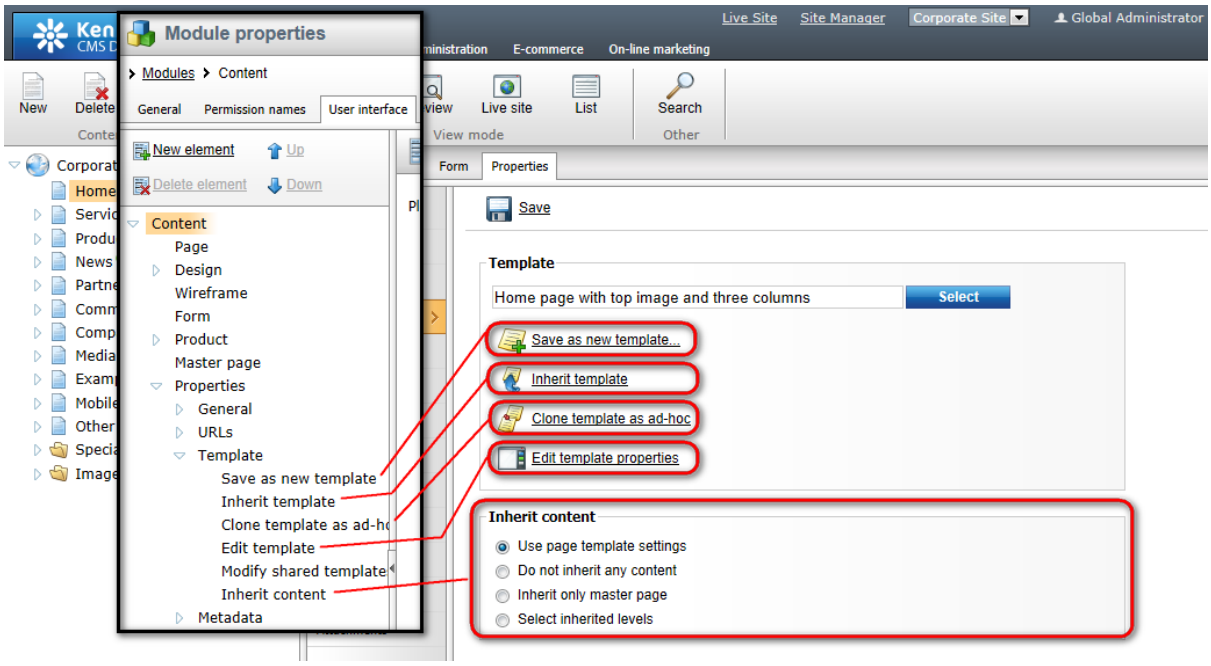
Template tab

On the **Template** tab, you can hide the **Save as new template**, **Inherit 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.

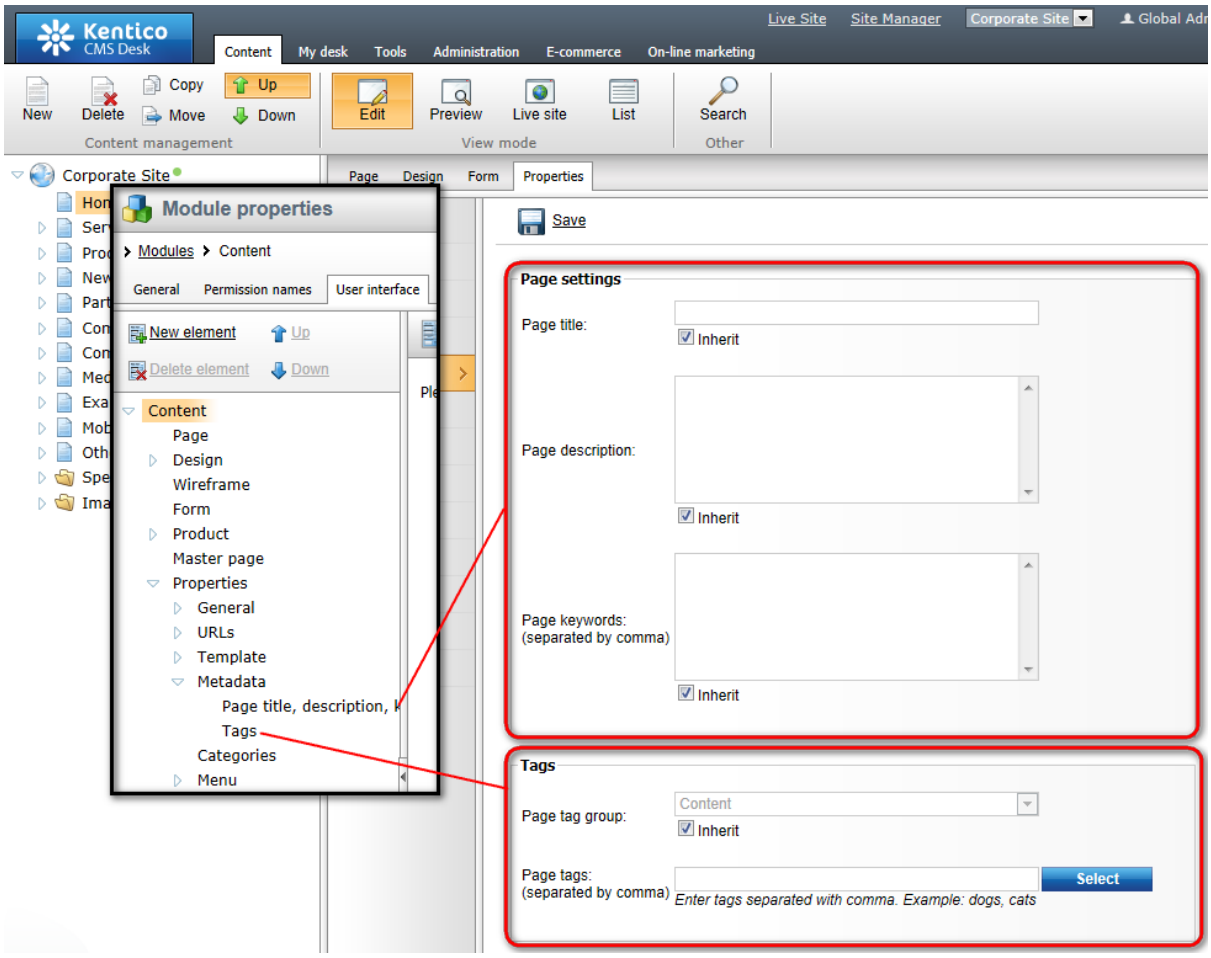
The **Modify shared templates** UI element works as a permission: it determines if the users can modify shared page templates both from here and on the **Design** tab.

Please note that the **Save as new template** action is not displayed in the screenshot because it is available only for ad-hoc or reusable templates. If displayed, the action is positioned instead of the **Inherit template** action. This is why the UI element is connected by a dotted line with the action button.



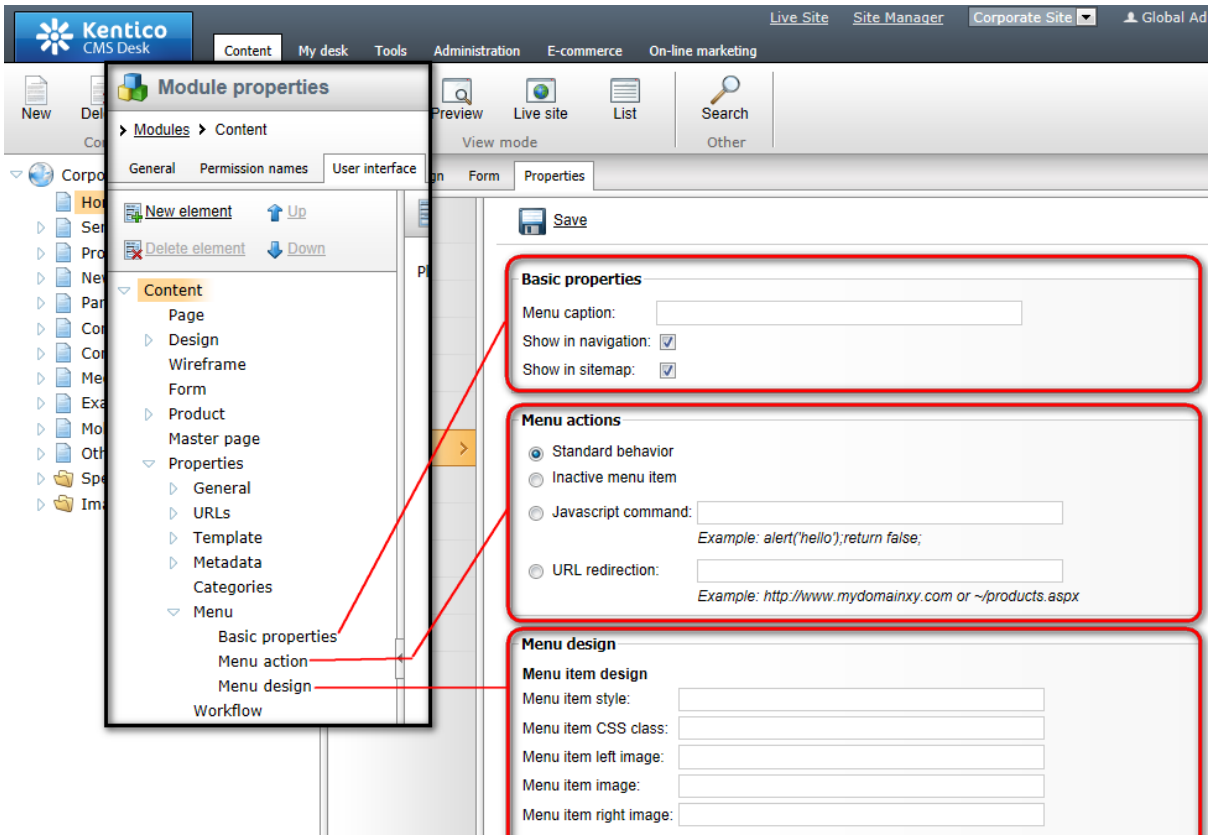
Metadata tab

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.



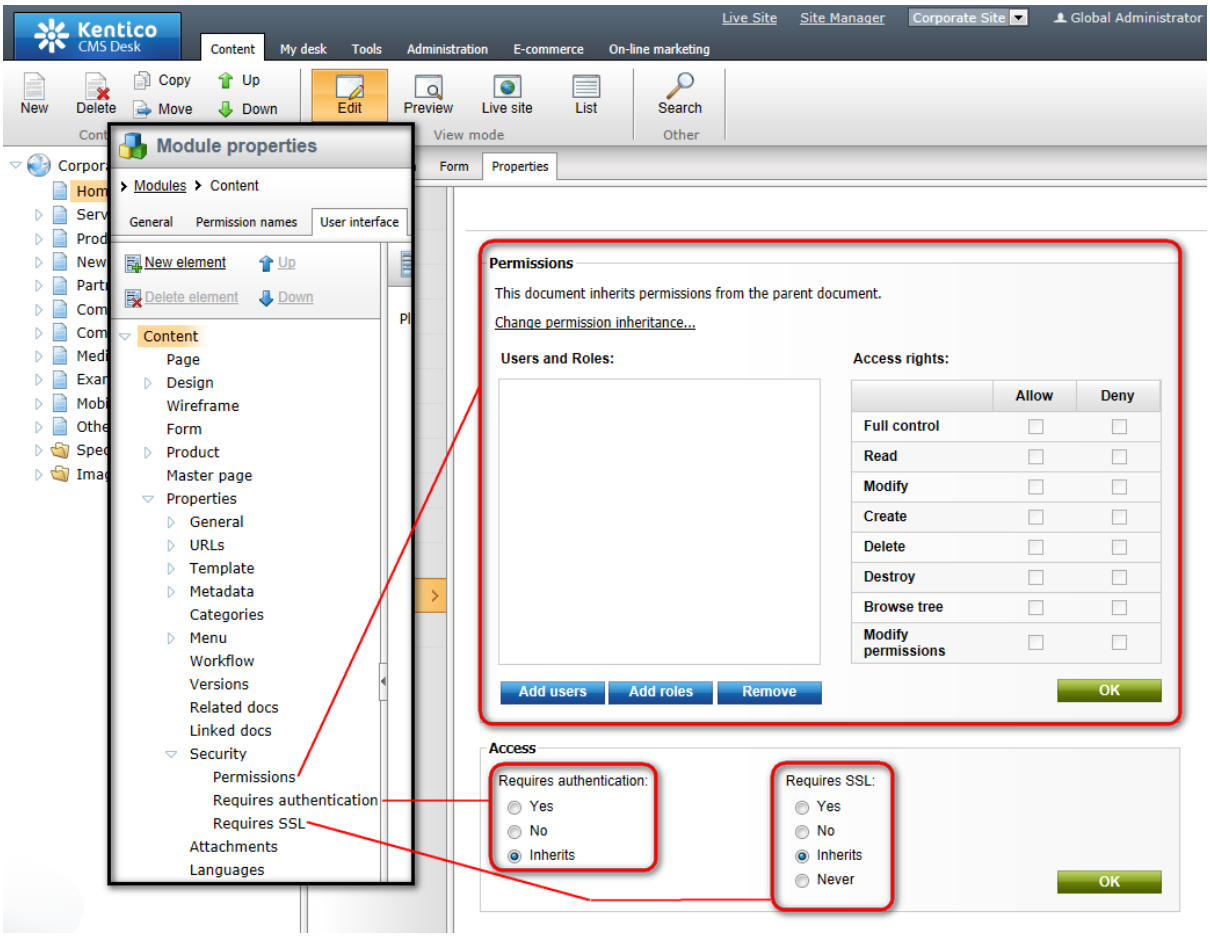
Menu tab

The **Menu** tab consists of three sections - **Basic properties**, **Menu actions**, **Menu design** - while these sections are represented by UI elements with the same names as shown in the following screenshot:



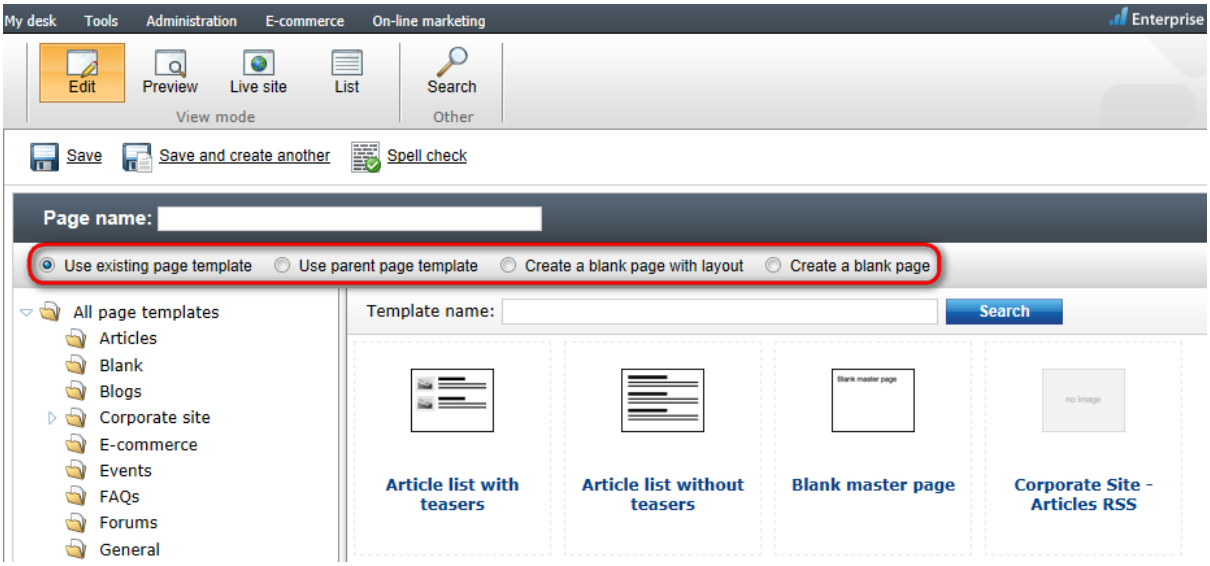
Security tab

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**:



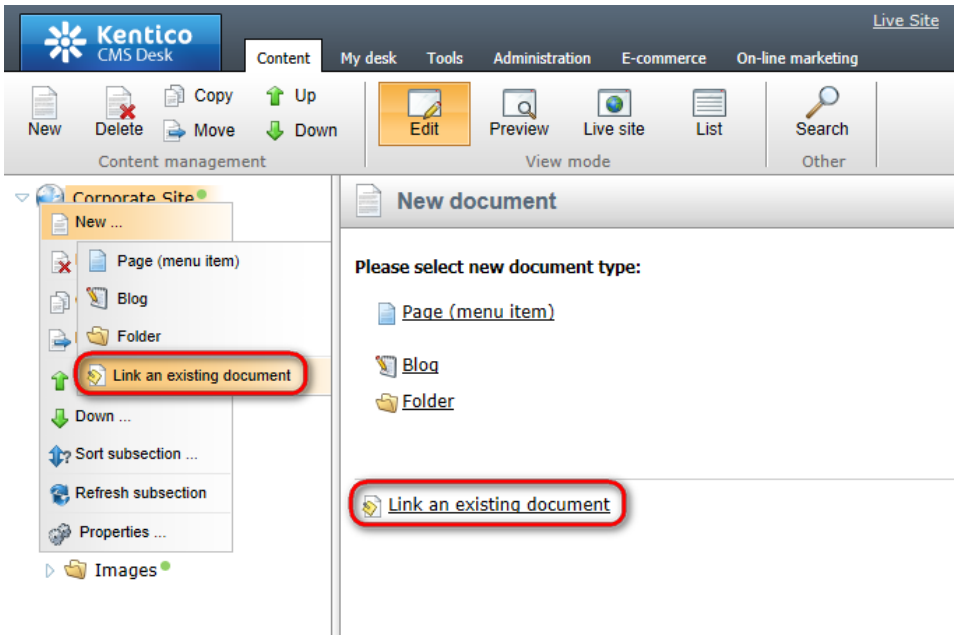
7.11.6.4.5 CMS Desk -> Content -> New

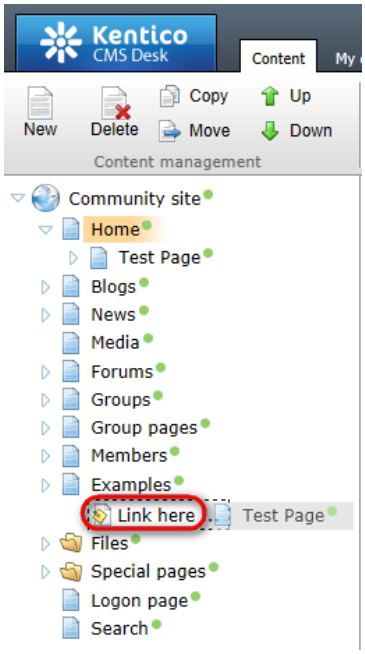
You can hide individual UI elements which are used when choosing a page template while creating a document.



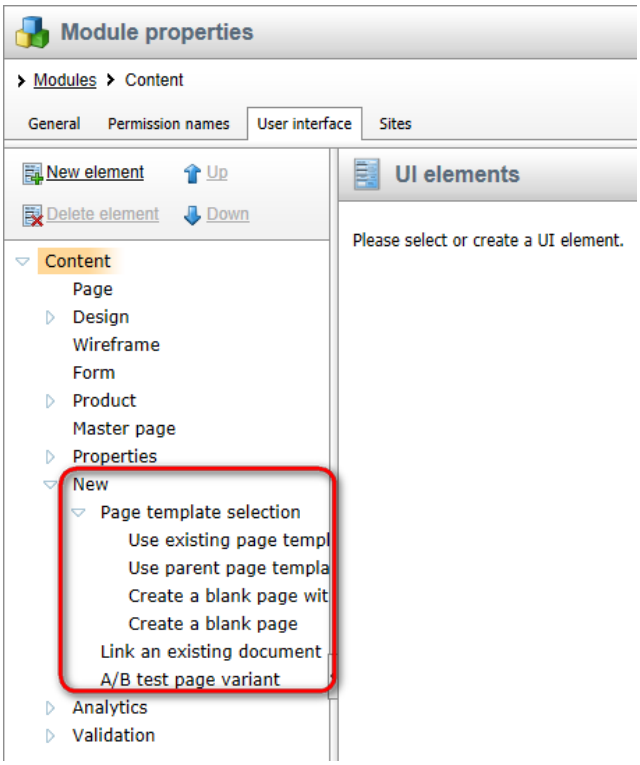
You can also hide the **Link an existing document** link from:

- Context menu
- Document type selection page
- Drag & Drop operation



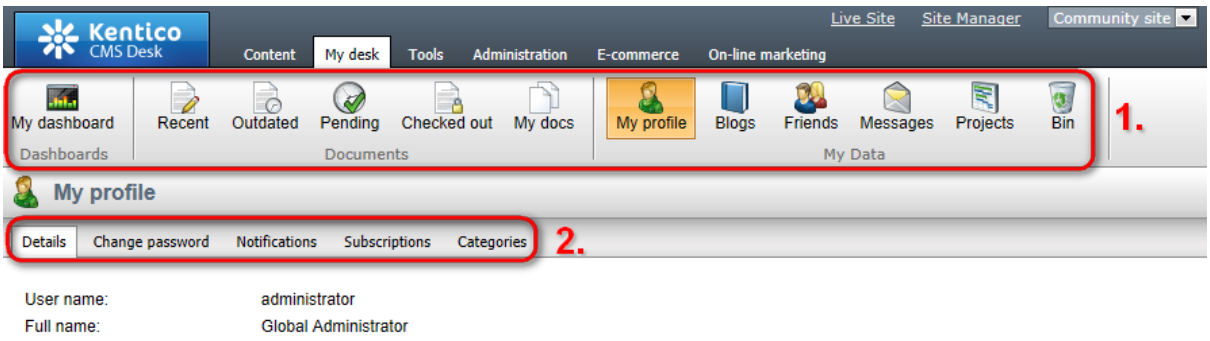


The list of these elements can be found on the **User interface** tab in **Site Manager -> Development -> Modules -> Edit** (🖉) ***Content***.

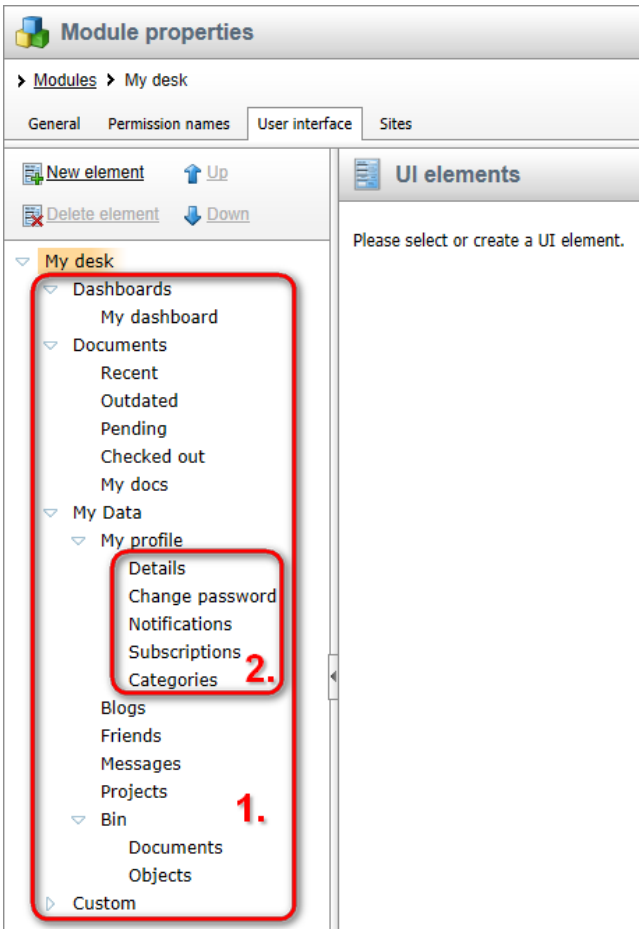


7.11.6.4.6 CMS Desk -> My desk tab

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.).

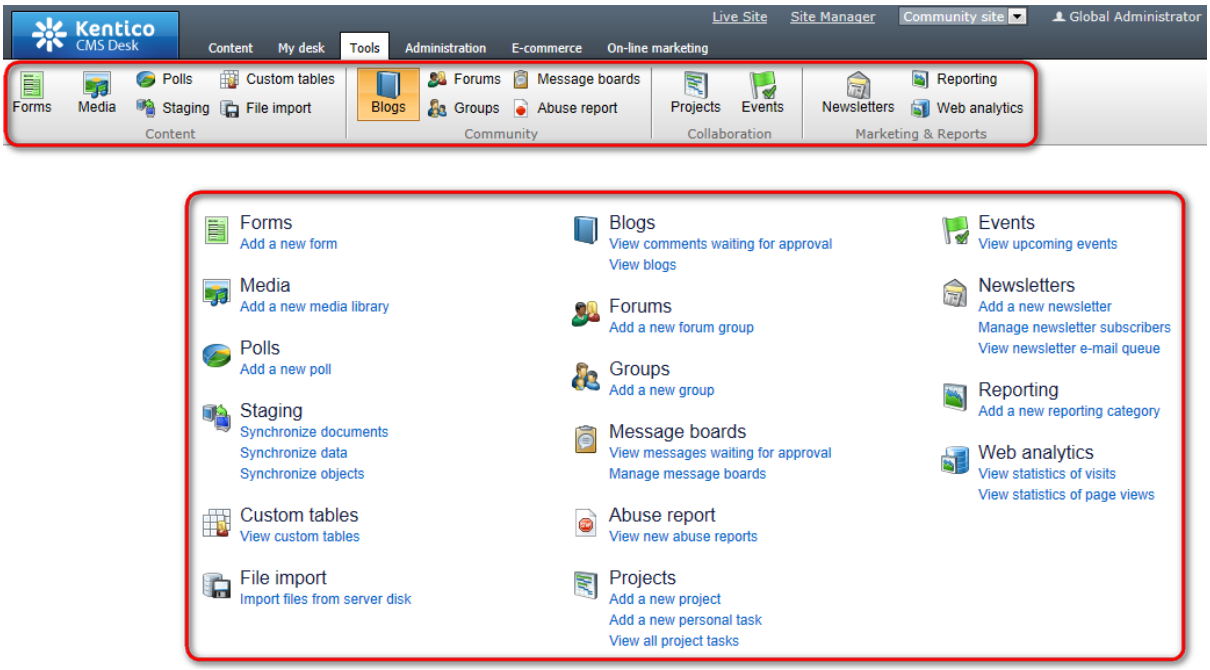


In the screenshot below, you can see the UI elements of the **My desk** module. Their names match the names of the menu items and tabs on the screenshot above.

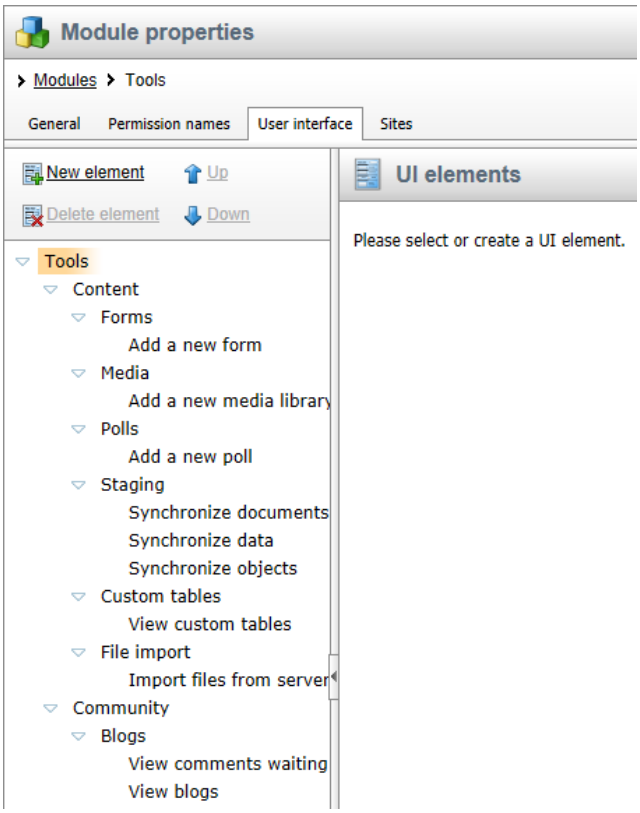


7.11.6.4.7 CMS Desk -> Tools tab

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:

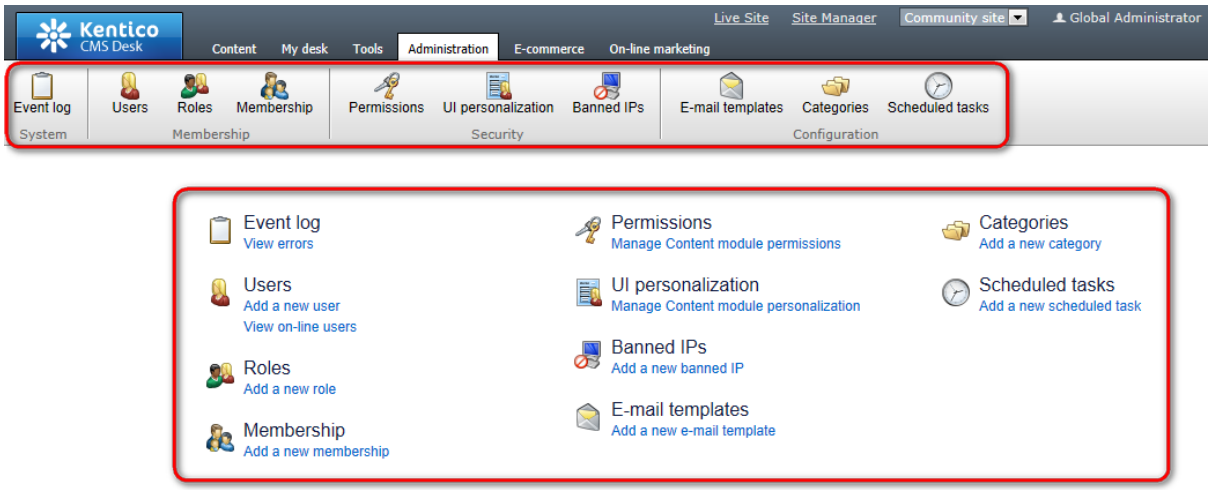


The Tools module has UI elements with the same names as the ribbon and panel menu items in the screenshot above.

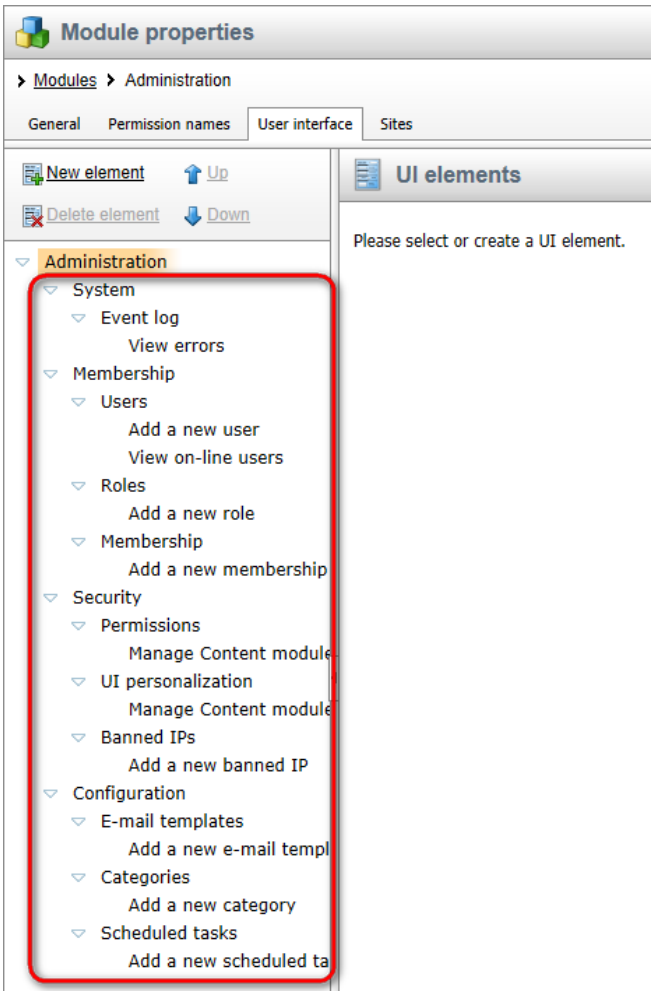


7.11.6.4.8 CMS Desk -> Administration tab

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.



7.11.6.4.9 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.

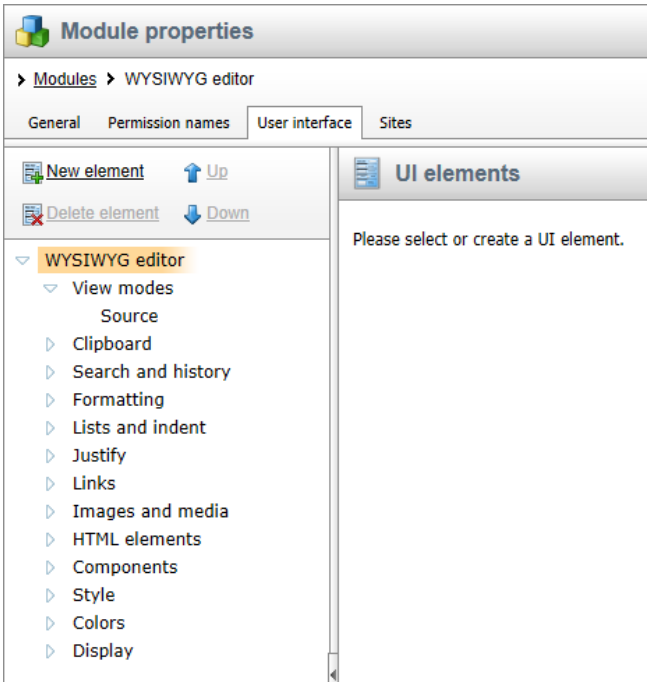
7.11.6.4.10 WYSIWYG editor

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" />
```

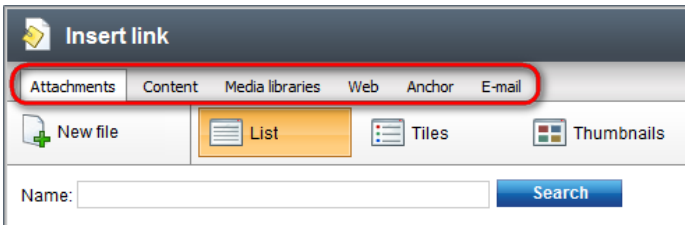
Once this is done, all UI personalization settings will be applied to the WYSIWYG editor on the live site.

More web.config settings can be found in Appendix B - Web.config parameters -> WYSIWYG editor settings.

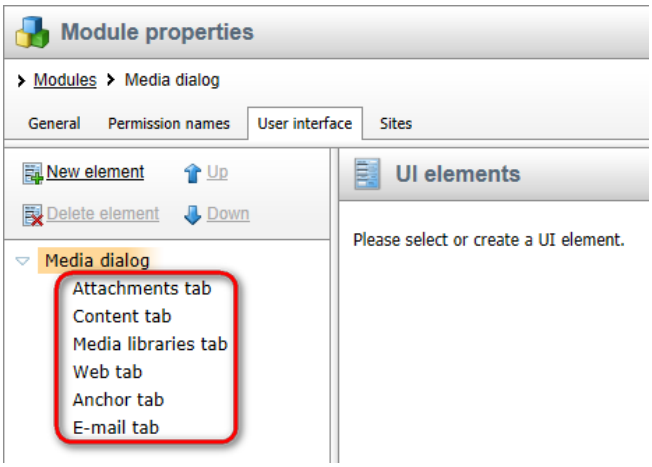
7.11.6.4.11 Media dialog

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**.



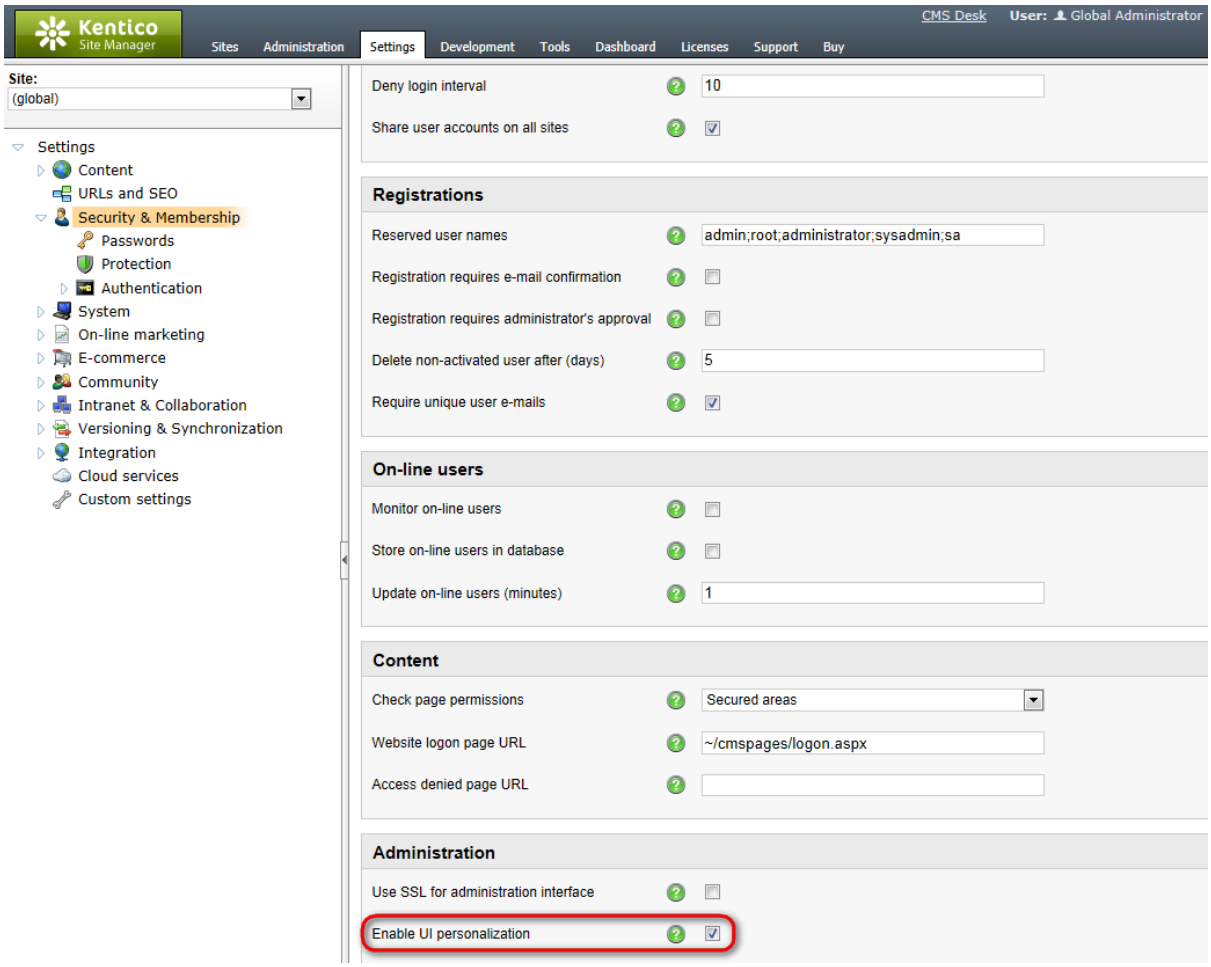
UI elements are defined centrally in the **Media dialog** module. They are applied to all occurrences and versions of the dialog, even though some of these tabs are not displayed in all cases (e.g. the E-mail tab is not displayed in Insert image or media dialog).



7.11.6.5 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.



7.11.6.6 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.

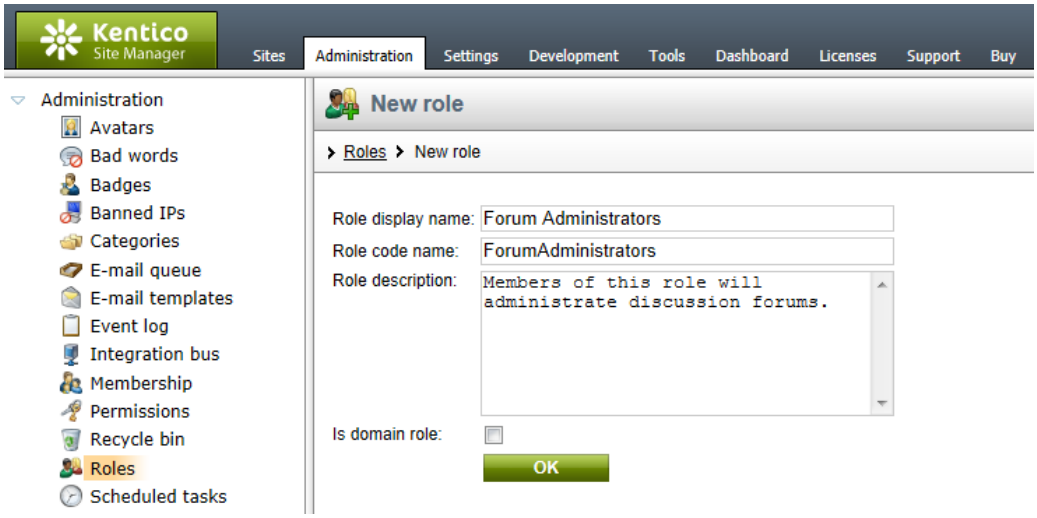
How to achieve it? We will create a new role and a new user who will be a member of this role only. Then we will define UI profile settings for this role and try logging in as the new user to see the simplified UI.

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 the **New role** ( ) icon. In the following dialog, enter these details:

- **Role display name**: Forum Administrators
- **Role code name**: ForumAdministrators
- **Role description**: some text describing the role
- **Is domain role**: leave the field blank

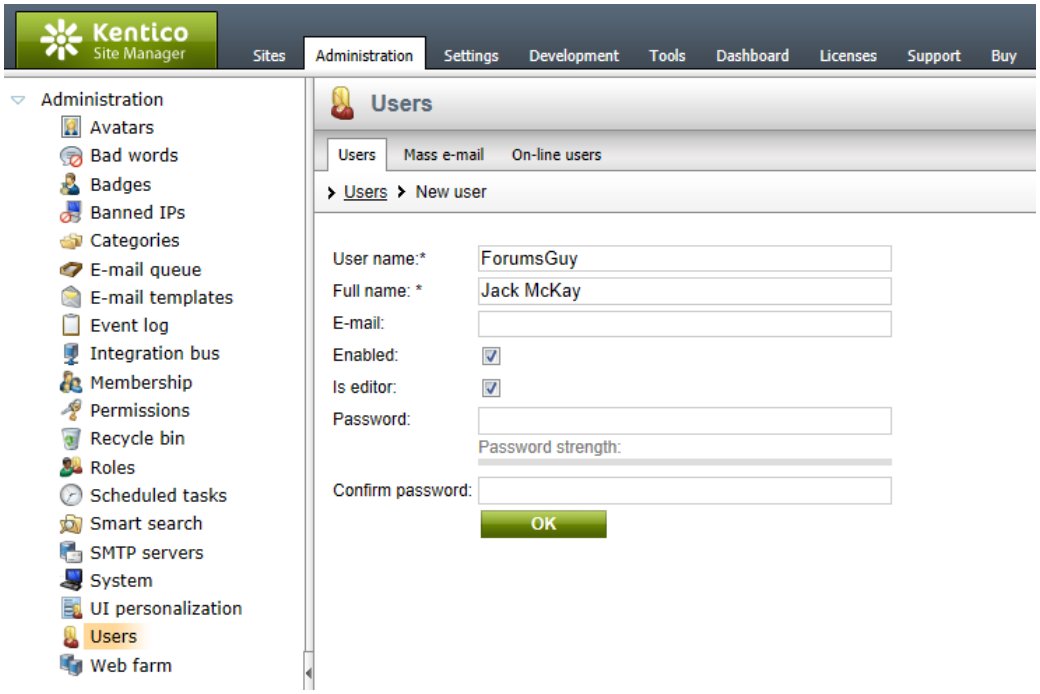
Click **OK**.



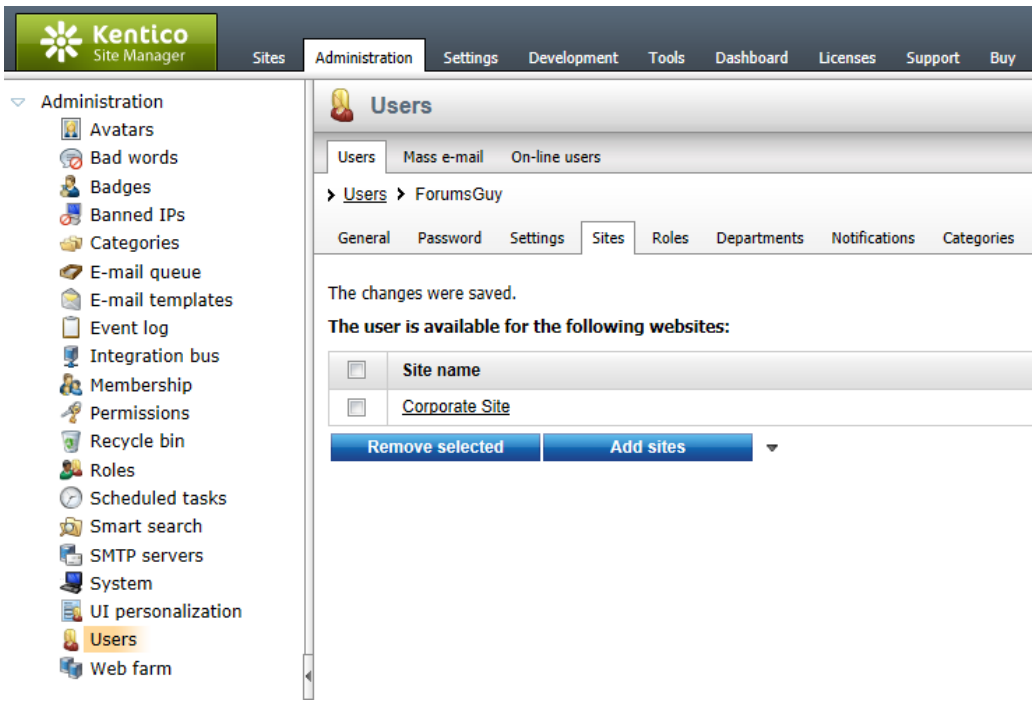
3. Now let's create the user. Go to **Site Manager -> Administration -> Users** and click the **New user** link. Enter the following details in the New user dialog:

- **User name**: ForumsGuy
- **Full name**: Jack McKay
- **Enabled**: enabled
- **Is editor**: enabled

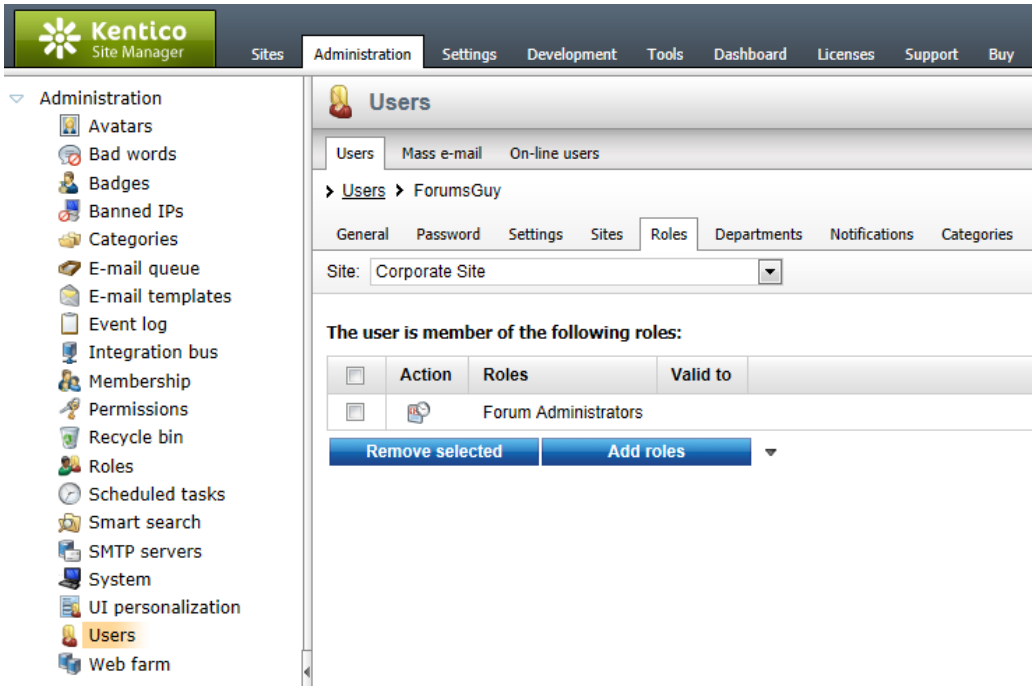
Click **OK**.



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:

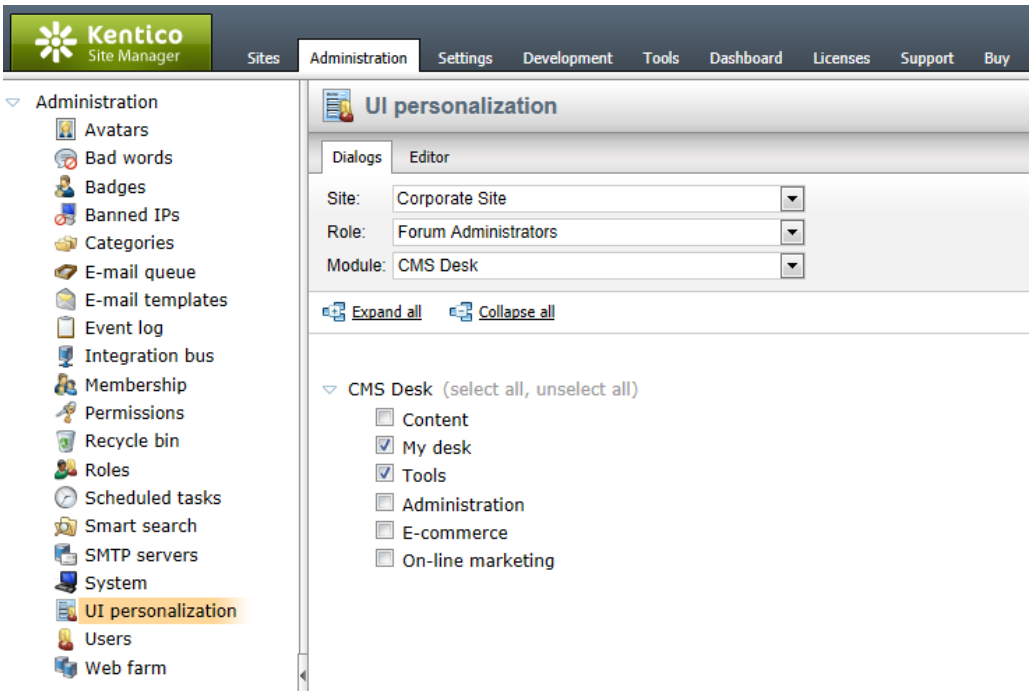
- **Site**: your site
- **Role**: Forum Administrators

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:

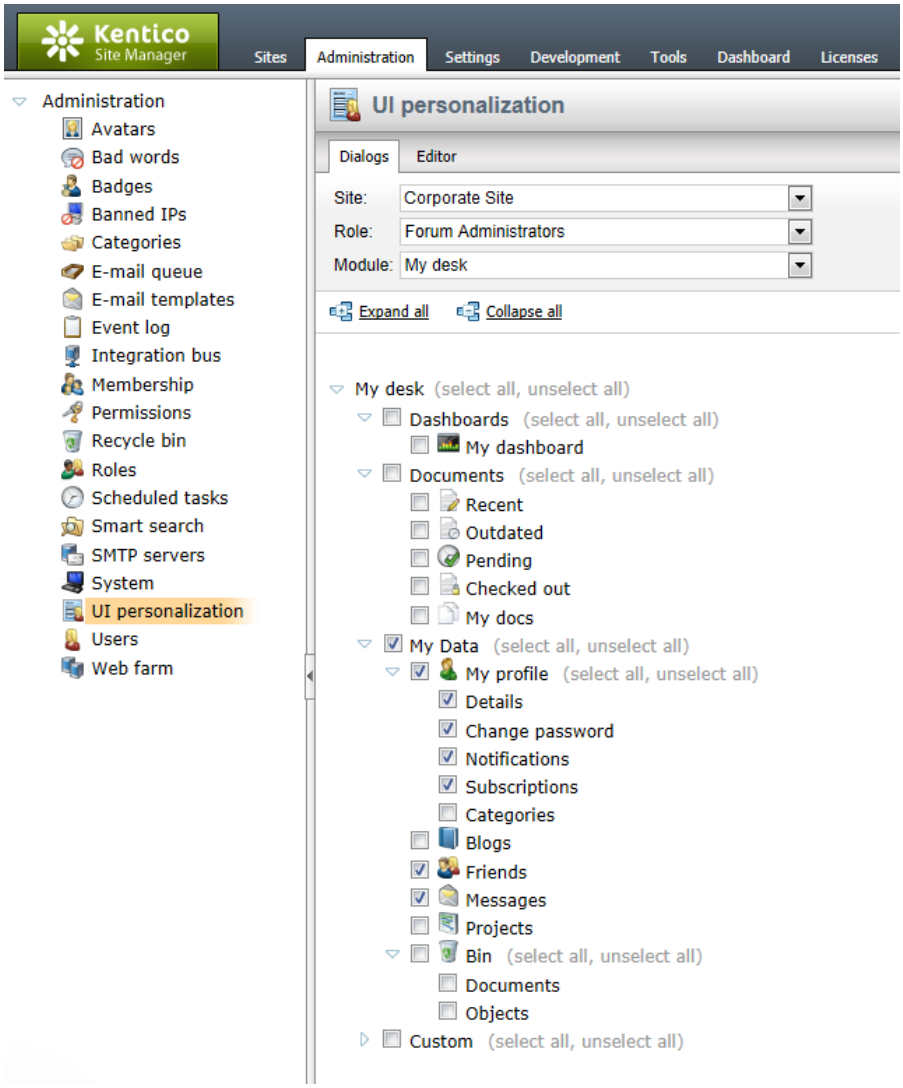
- **Content**: disabled
- **My desk**: enabled
- **Tools**: enabled
- **Administration**: disabled



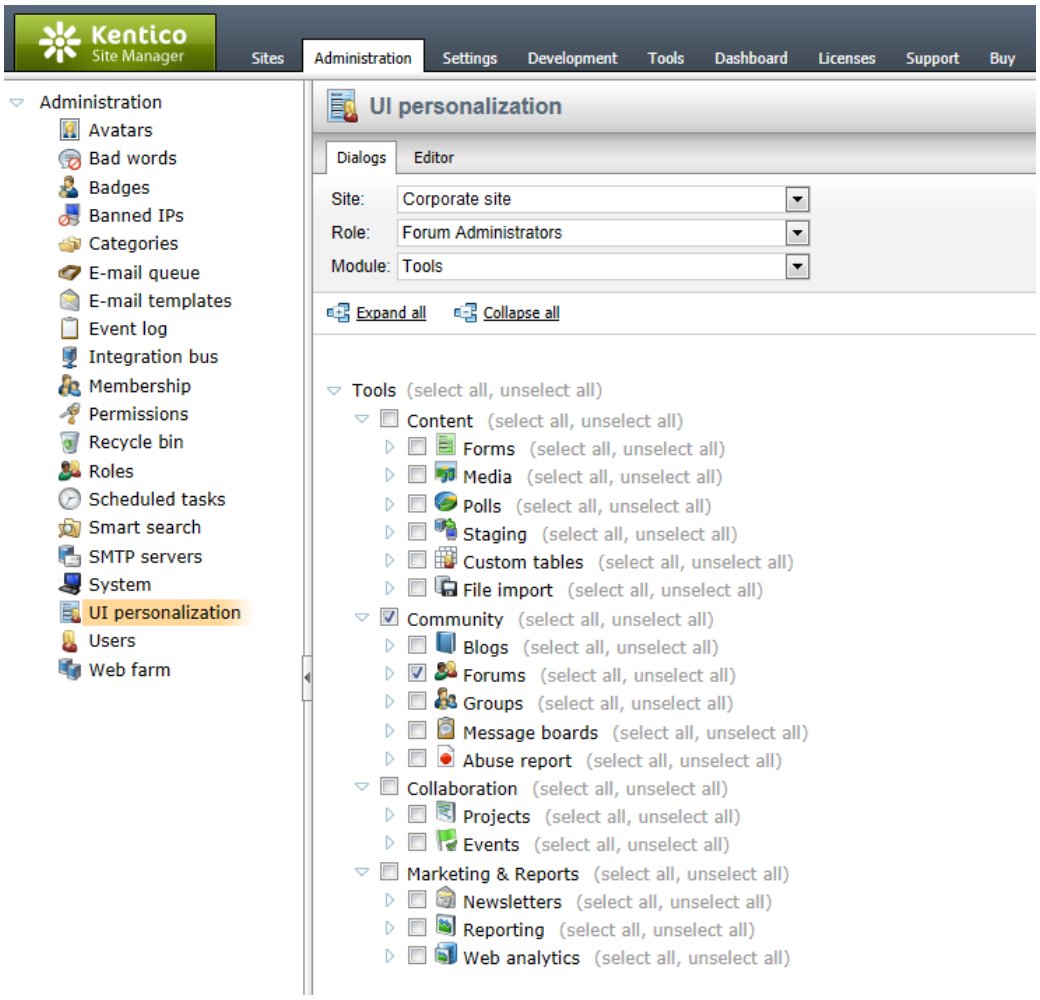
8. On the My desk tab, we will need only the Account, Messages and Friends sections, all others are not needed for forum administrators. Choose **Module**: My Desk and enable only the UI elements listed below, do not enable the rest.

- **My data**: enabled
  - **Account**: enabled
    - **Details**: enabled
    - **Change password**: enabled
    - **Notifications**: enabled

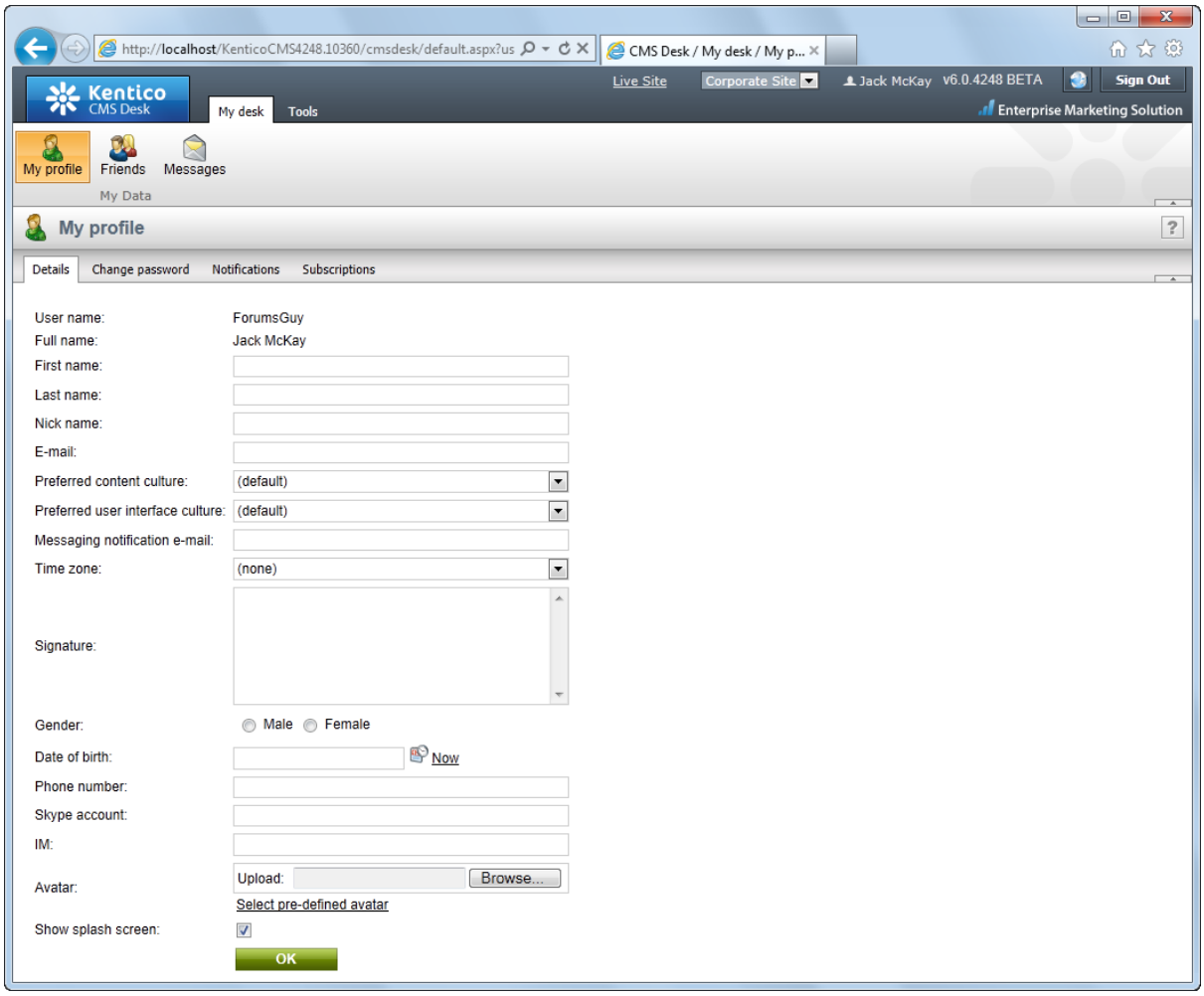
- **Subscriptions**: enabled
- **Friends**: enabled
- **Messages**: enabled



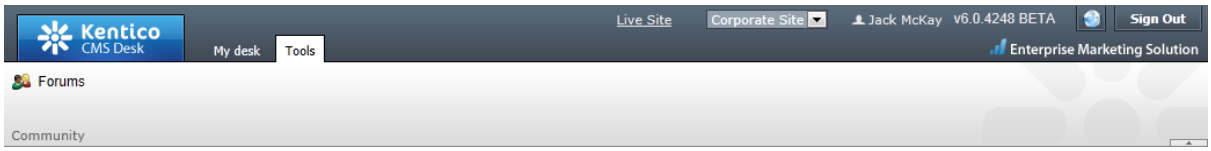
9. Finally, we need only the **Forums** option in the **Tools** menu. Choose **Module**: Tools and enable only the **Community** and **Tools** UI elements as you can see in the screenshot below. Leave all the remaining UI elements disabled.



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.



7.11.6.7 UI elements management

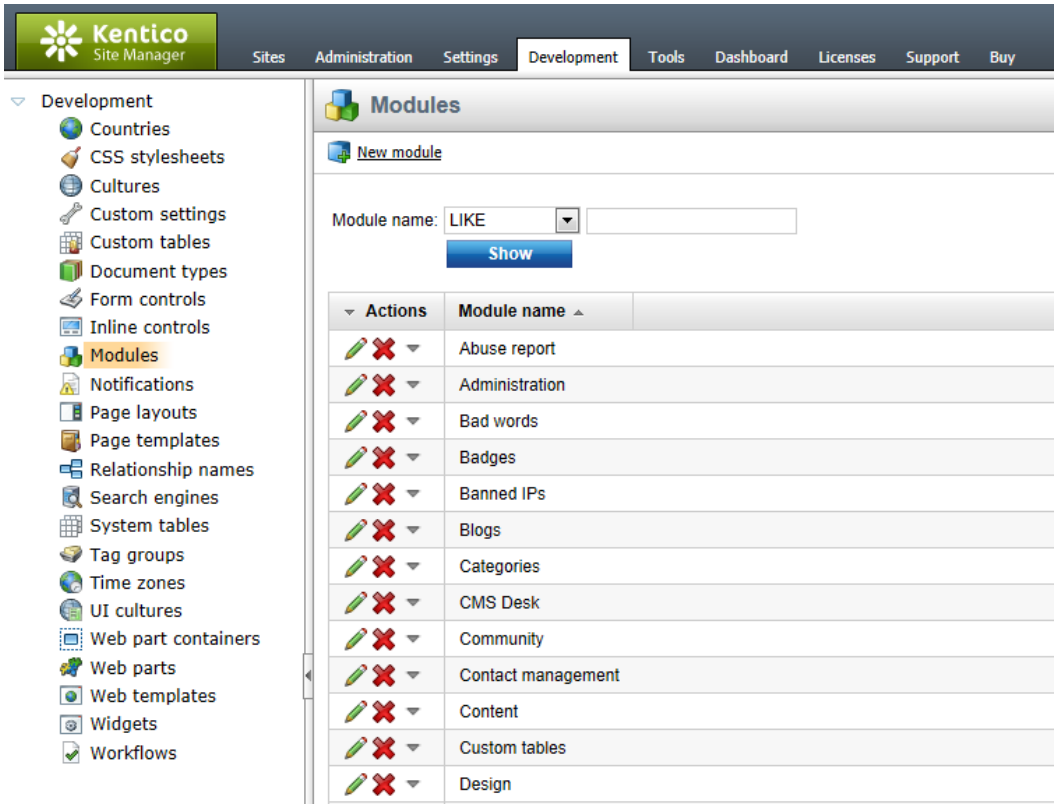
7.11.6.7.1 UI elements management

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**.

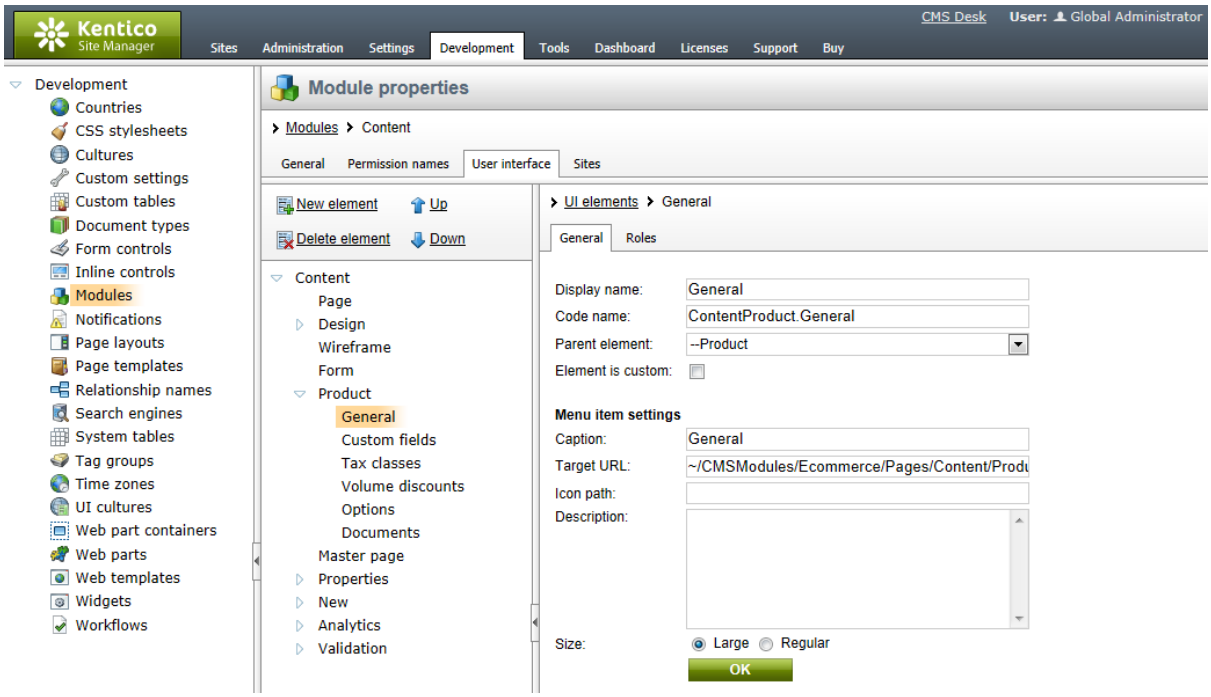
On the **General** tab, you can set the following properties of the UI element.

- **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:

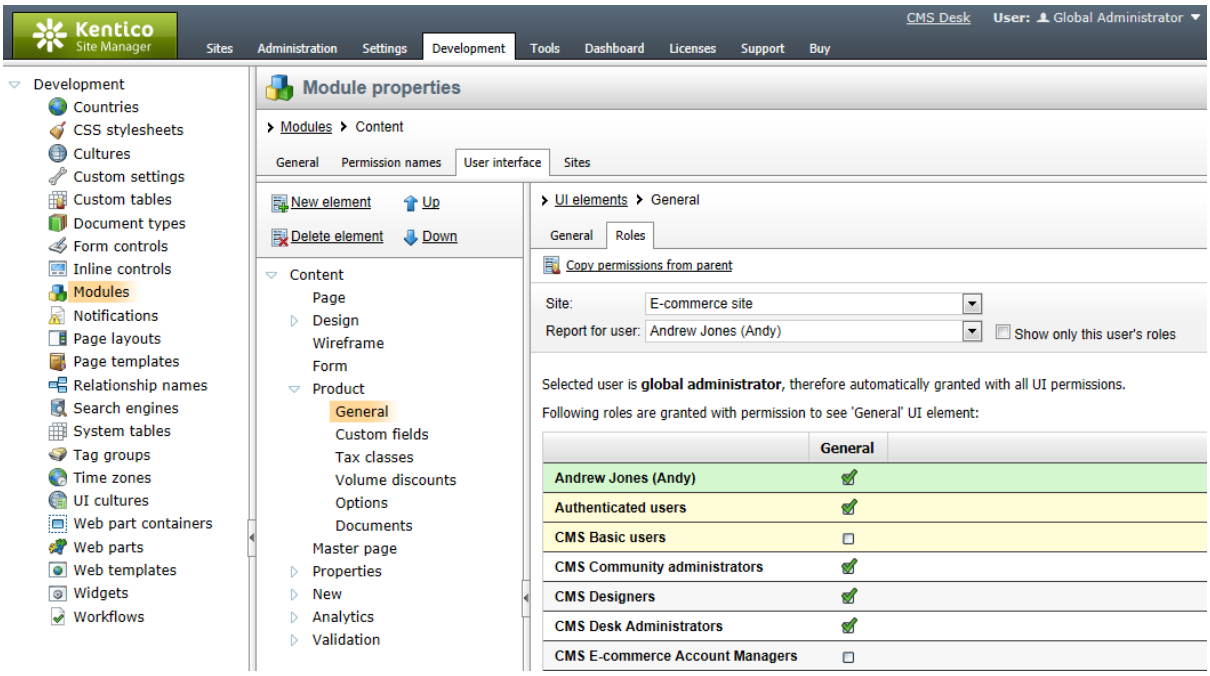
- If you enable () a check-box, the UI element will be visible to members of the role.
- If you disable () a check-box, members of the role will not see the UI element (unless they are

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, please click the **Copy permissions from parent** link. 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.

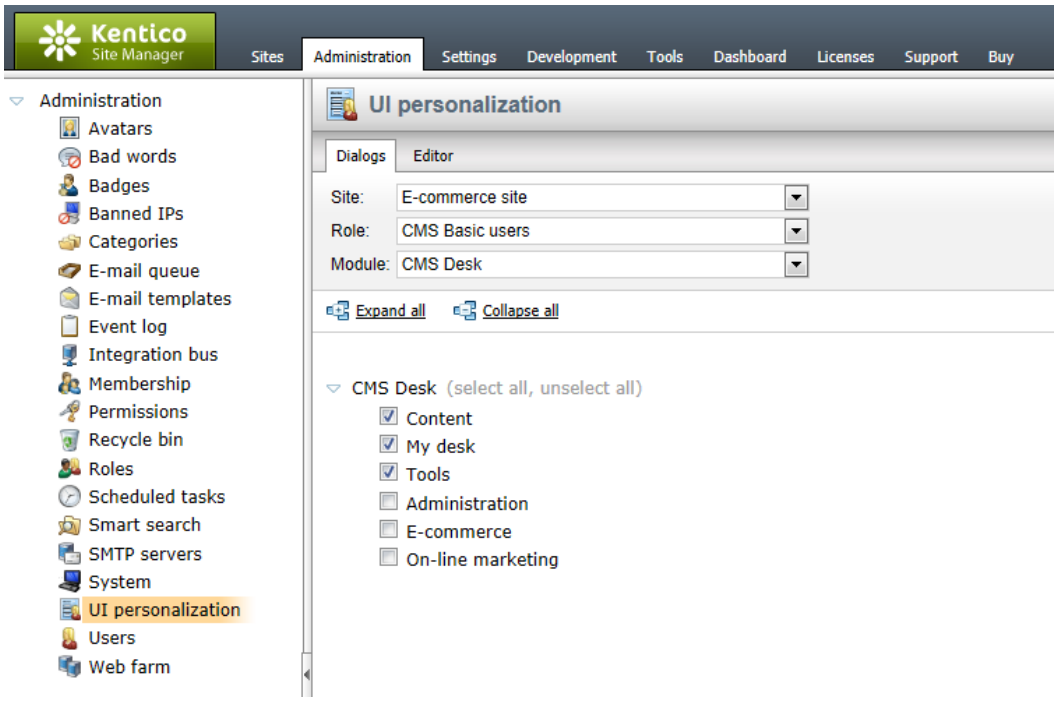
To check if a particular element is visible to a user, you can select the user from the **Report for user** drop-down list. After doing so, a sum of all permissions granted to the user's roles is displayed in the first line, highlighted in green color. Roles where the selected user is a member will be 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 the **Copy permissions from parent** link 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 link is disabled for all first-level UI elements.

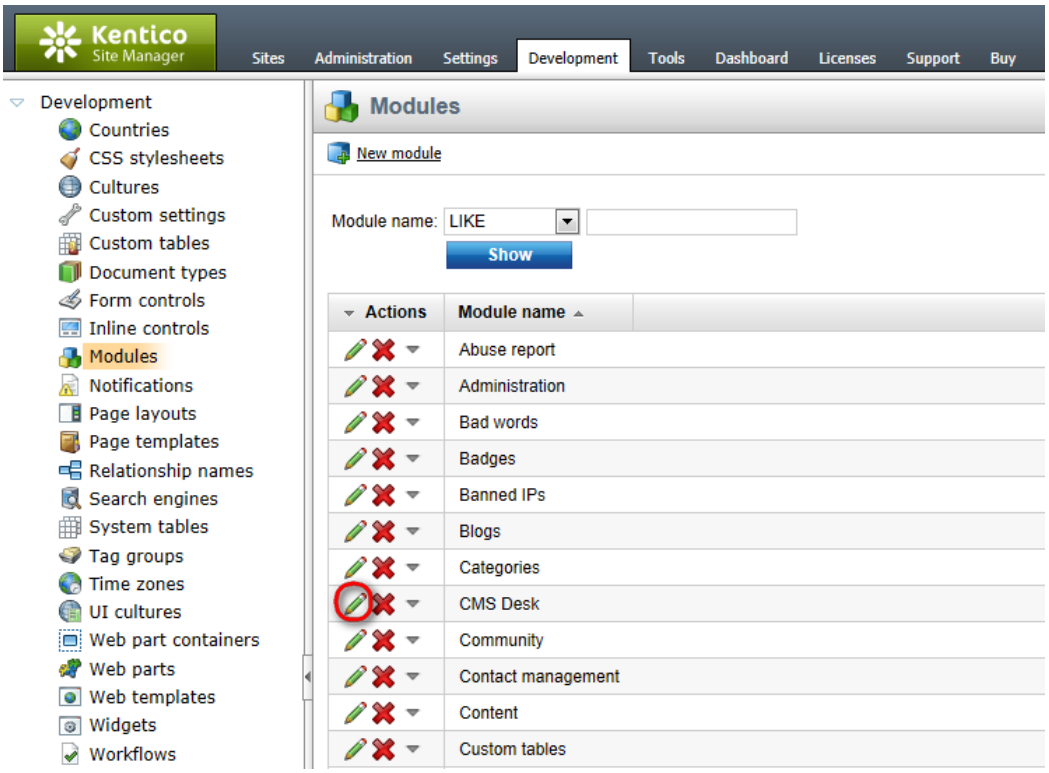
This is just another way to make the same configuration as described in the UI profile configuration topic, i.e. settings made here are reflected in **Site Manager -> Administration -> UI personalization** (see the screenshot below) and vice-versa.



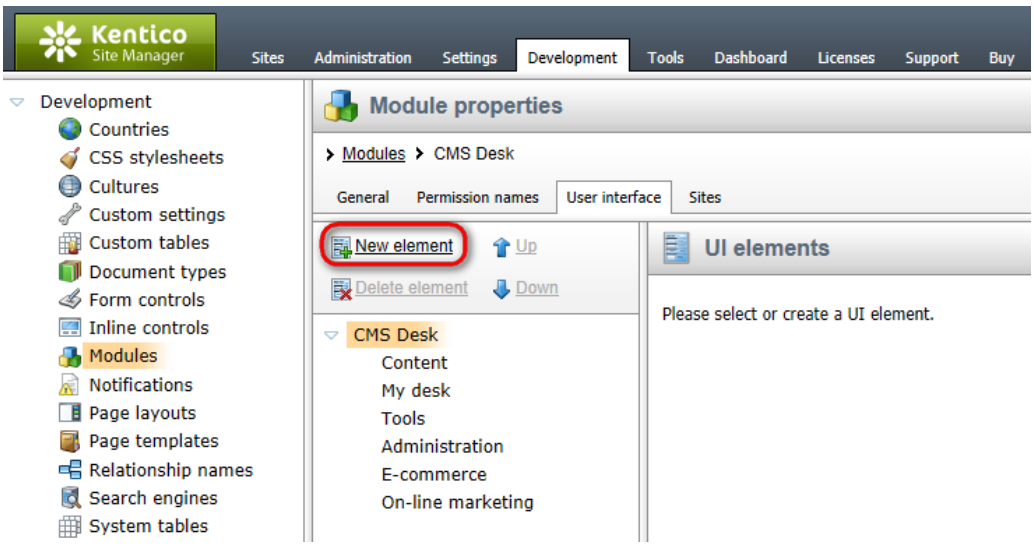
7.11.6.7.2 Example: Adding a new main tab to CMS Desk

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.

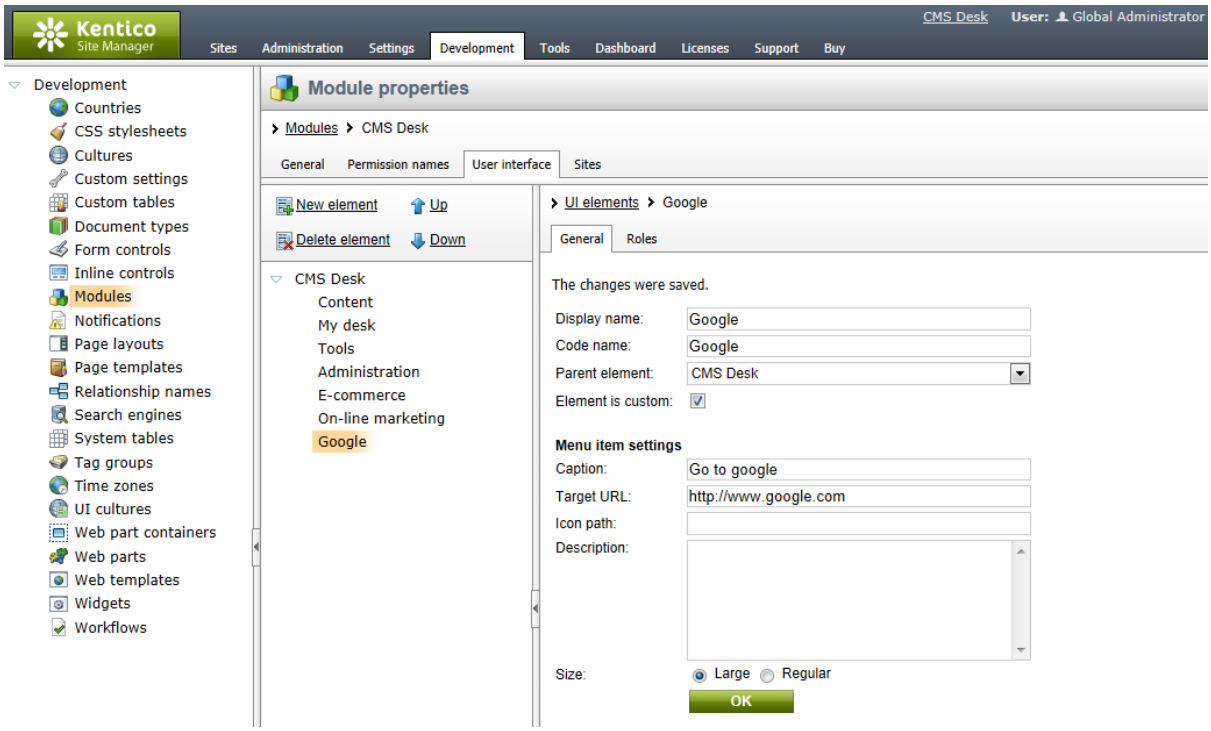


3. In the **New element** dialog, enter the following details:

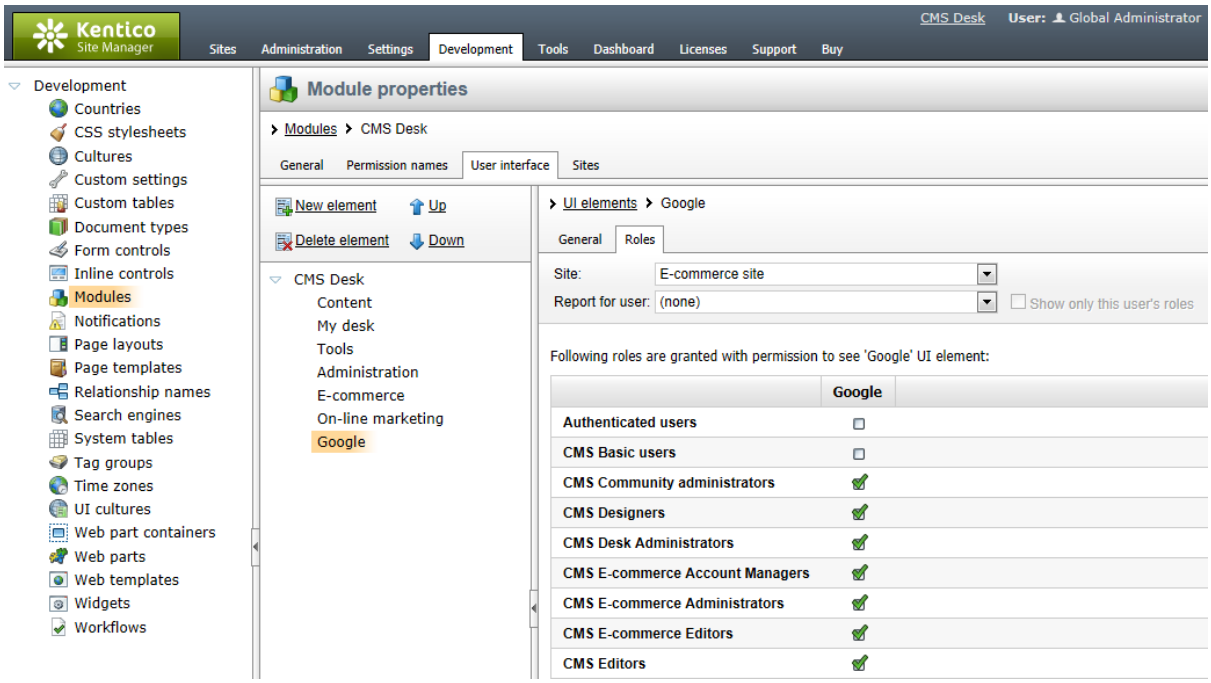
- **Display name**: *Google*.
- **Code name**: *Google*.
- **Element is custom**: *TRUE*.
- **Caption**: *Go to Google*.

- **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).

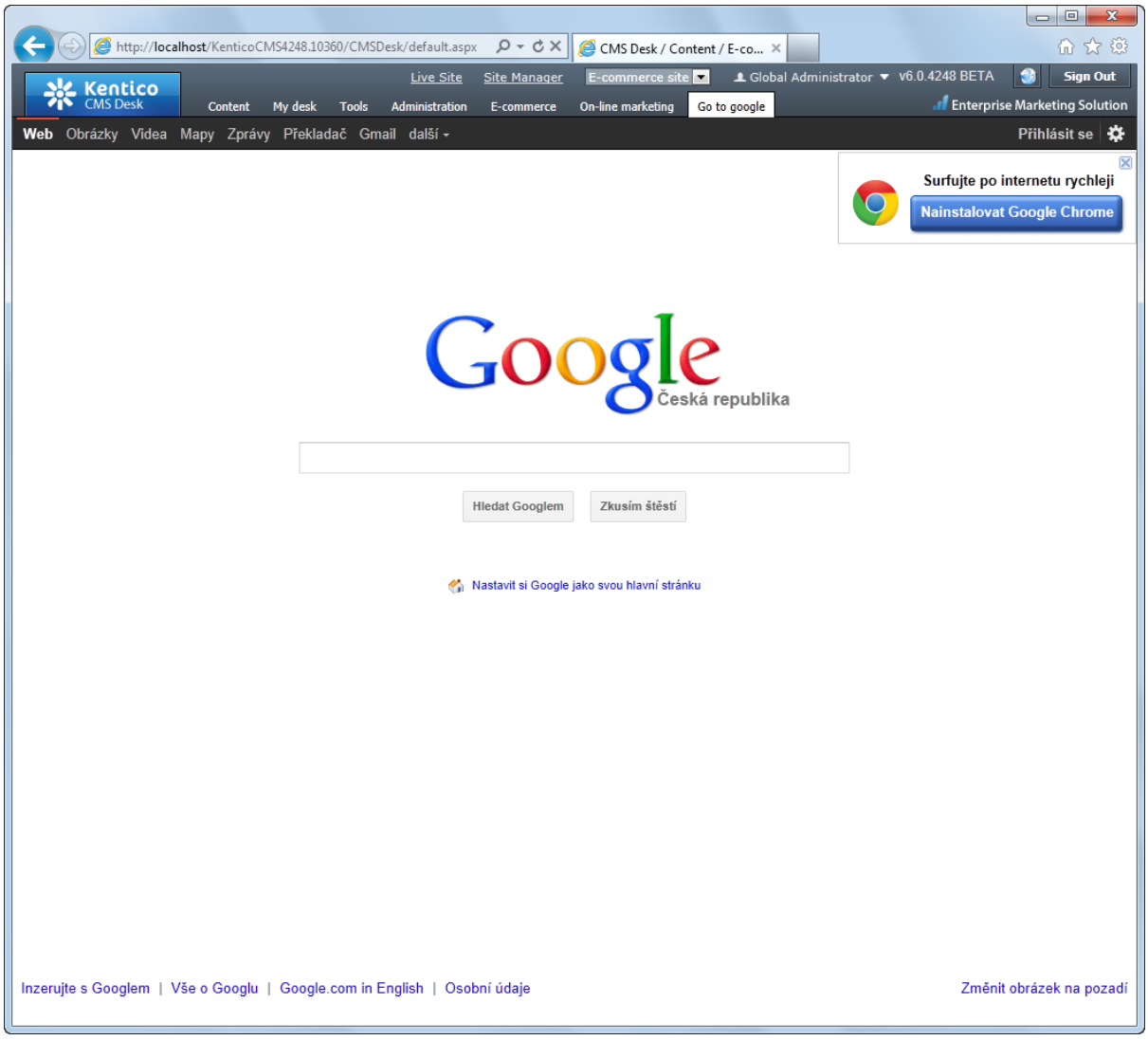
and click **OK**. The new UI element will be added to the tree as you can see in the screenshot below.



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.

7.11.7 Memberships

7.11.7.1 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.

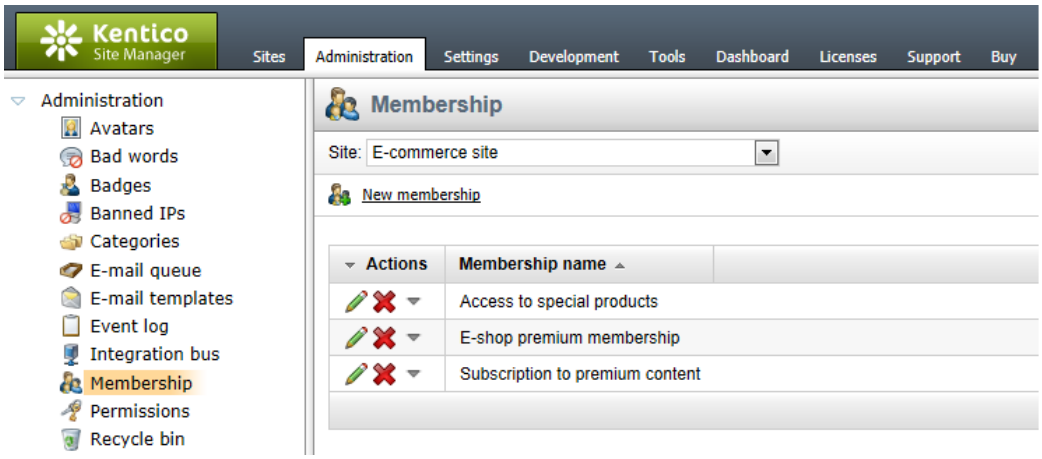
Memberships versus roles

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.

7.11.7.2 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 the **New membership** link. 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.

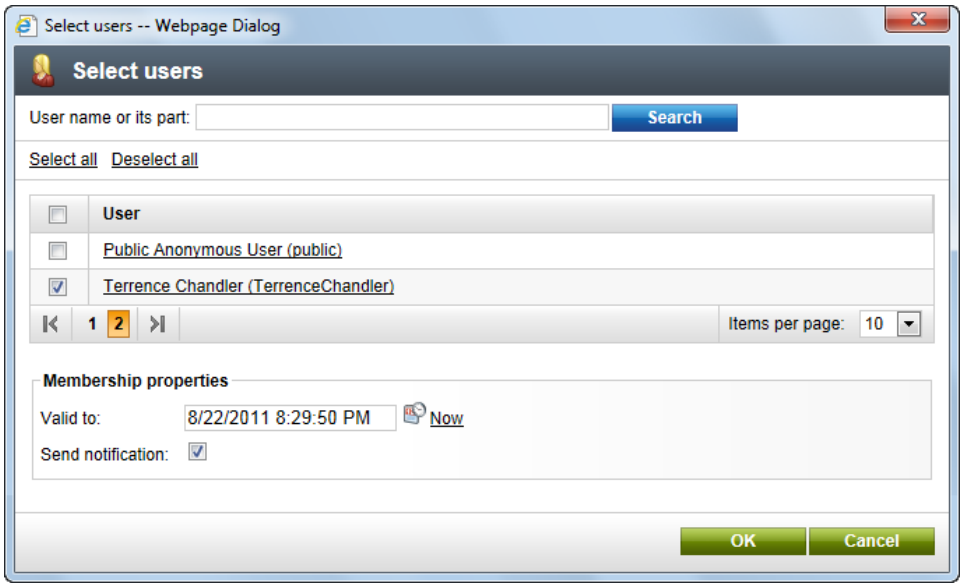
If you need to add a role under multiple memberships, you can save time by navigating to **Administration -> Roles** and editing () the given role on its **Memberships** tab. Here you may easily assign it to any number of memberships using a single action.

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.

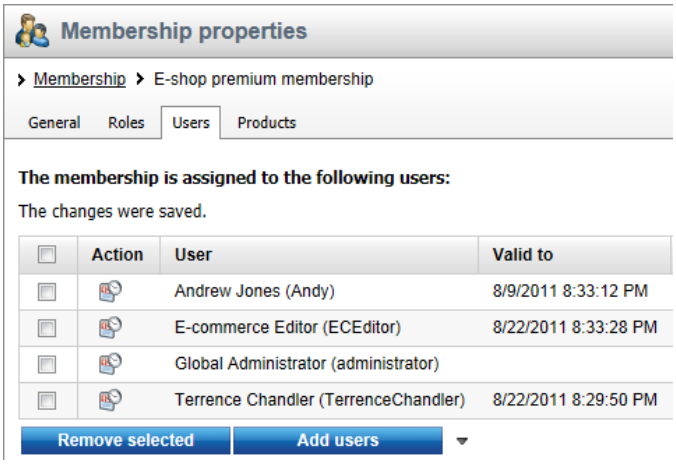
Typically users will be added automatically when they purchase the membership as a product, so it is not necessary to manually assign them one by one. The purpose of this tab is to allow administrators to monitor the membership and override its settings if necessary.

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 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.

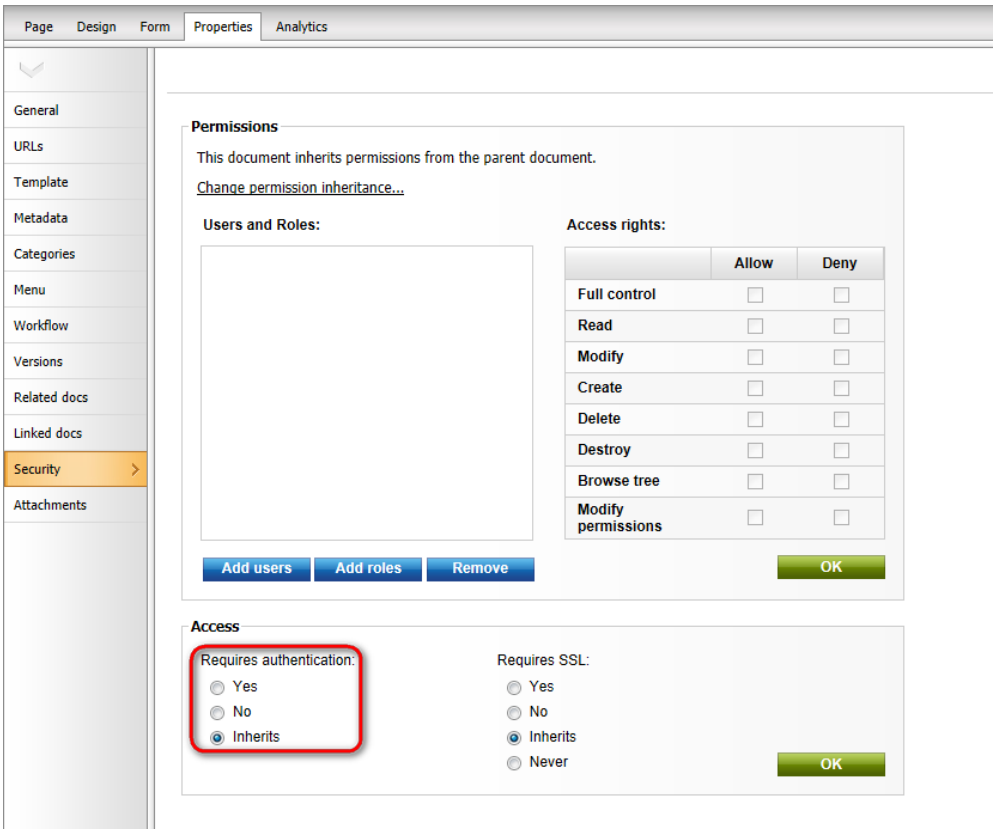
7.11.8 Security

7.11.8.1 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

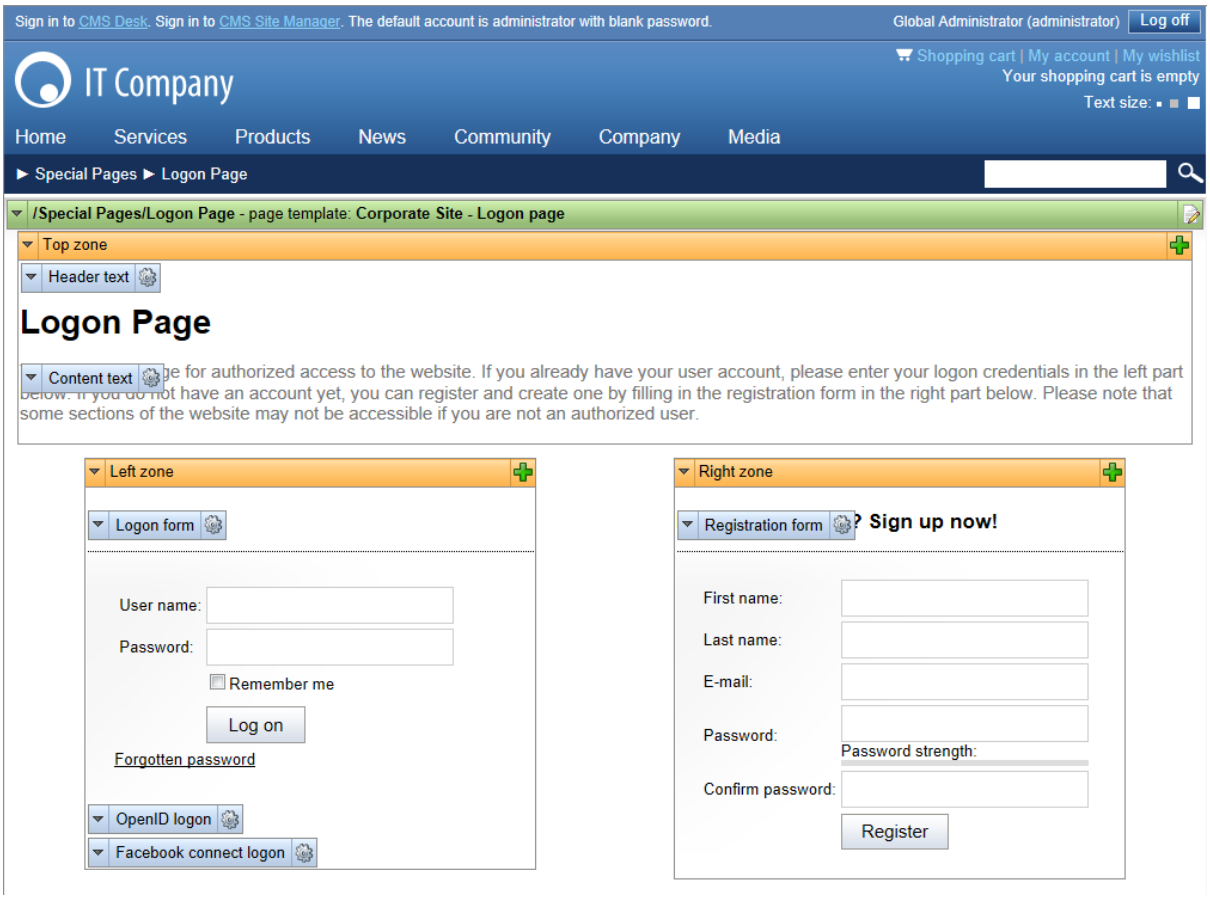
This example explains how to secure the **Products** section of the sample Corporate Site.

1. Sign in as administrator to **CMS Desk**. Go to the **Content** section and click the **Products** document in the content tree.

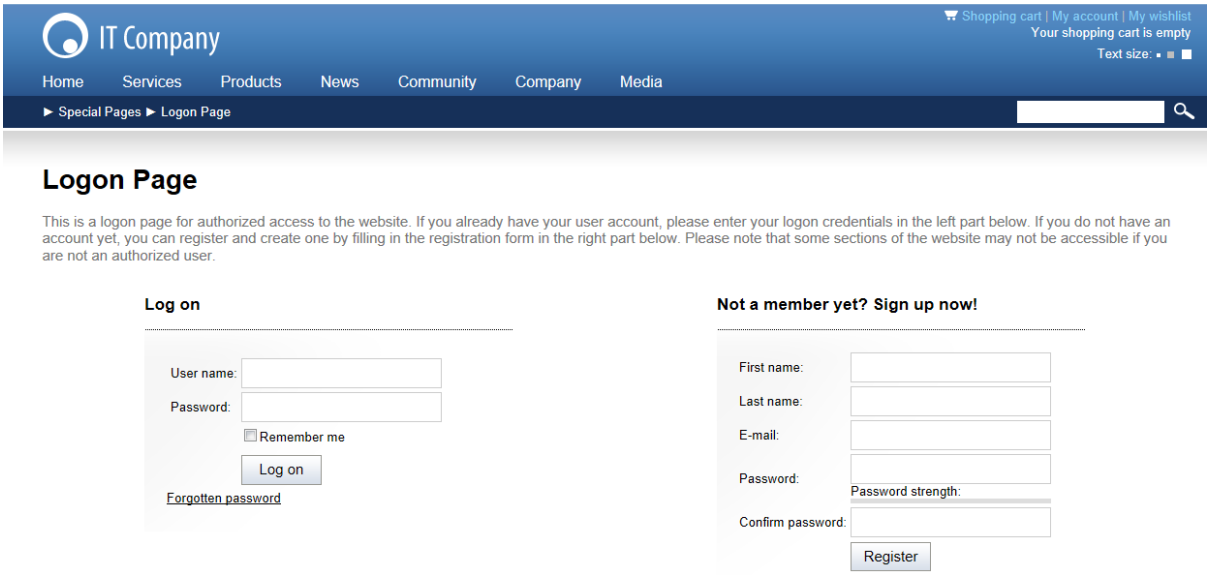
2. Click **Properties -> Security**. Set the value of the **Requires authentication** attribute to **Yes** and click **OK**.

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.

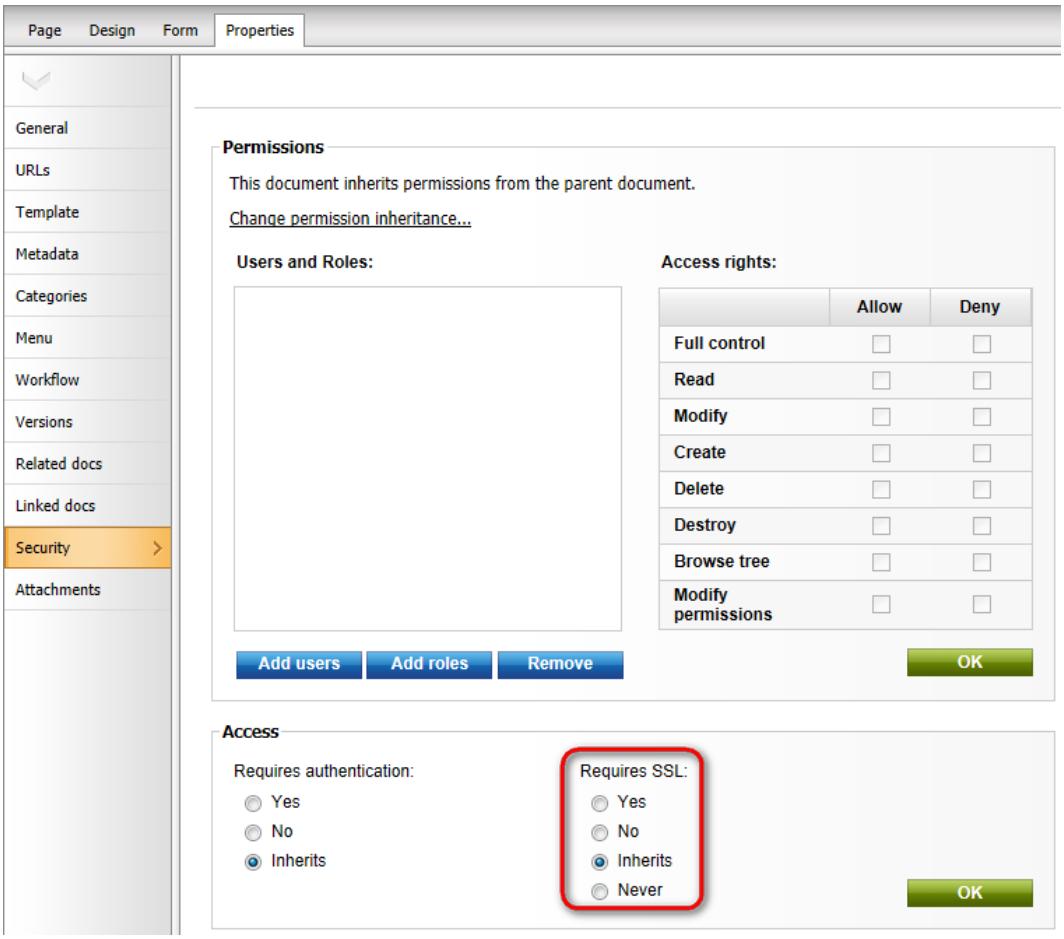
7.11.8.2 SSL (HTTPS) support

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

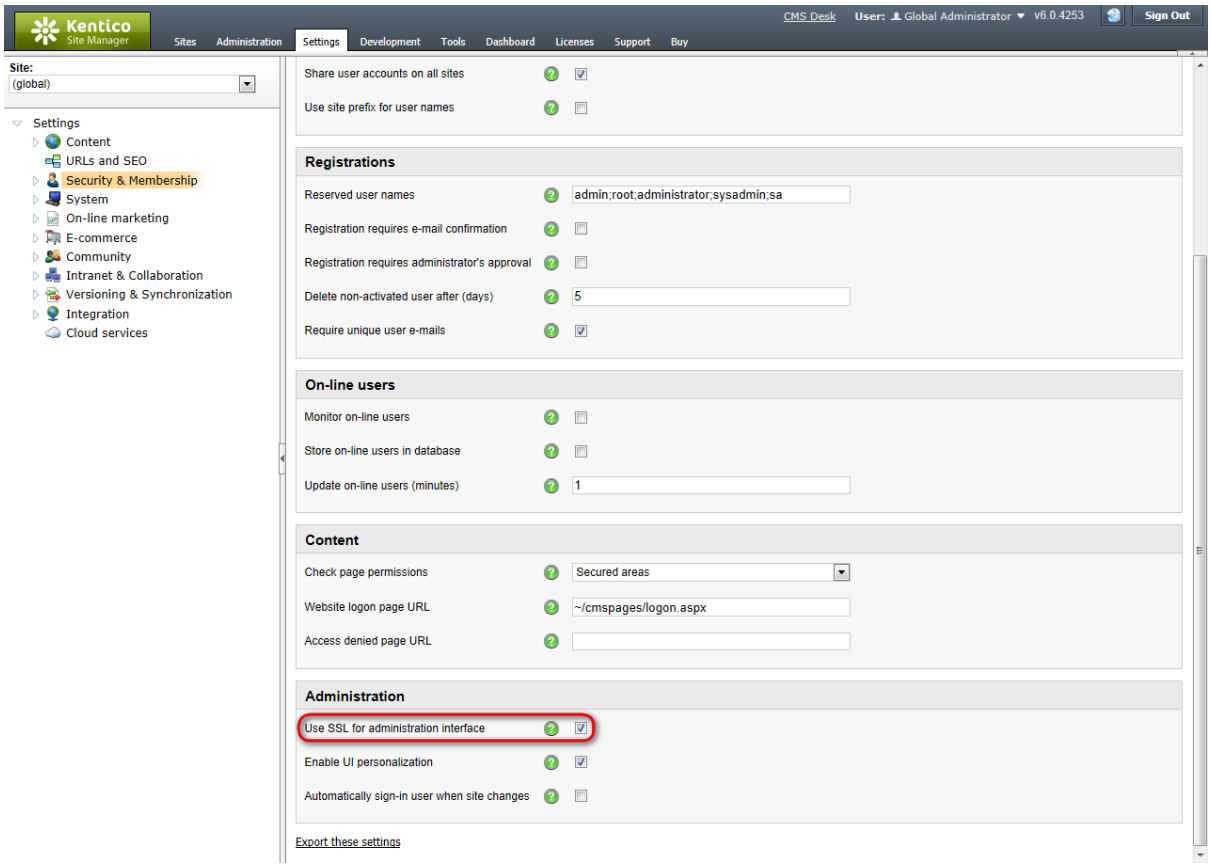
In order to specify that a page should only be available through the SSL protocol, you need to go to **CMS Desk -> Content** and select the corresponding document in the content tree. Then you can choose one of the following options in **Properties -> Security -> Requires SSL**:

- **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 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.

This issue can be solved 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 into it called **SSLRequestLoader.cs**.

2. Edit the class and add the following references:

[C#]

```
using CMS.SettingsProvider;
using System.Collections.Specialized;
using CMS.GlobalHelper;
```

3. Next, 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.

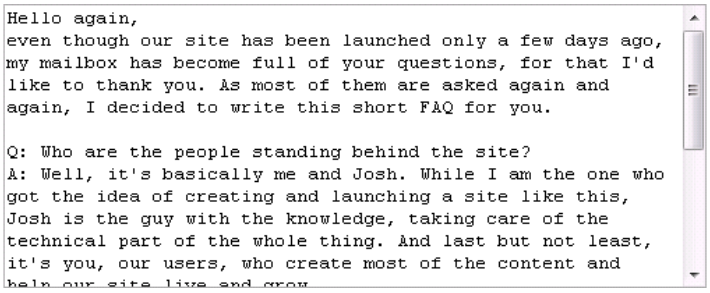
This example uses the **X-Forwarded-Ssl** header to check if the original request was submitted via HTTPS before it was forwarded to the application by the SSL offload device or load balancer. If this is the case, the **IsSSL** property is set to *true* and the system will process the request as if it 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.

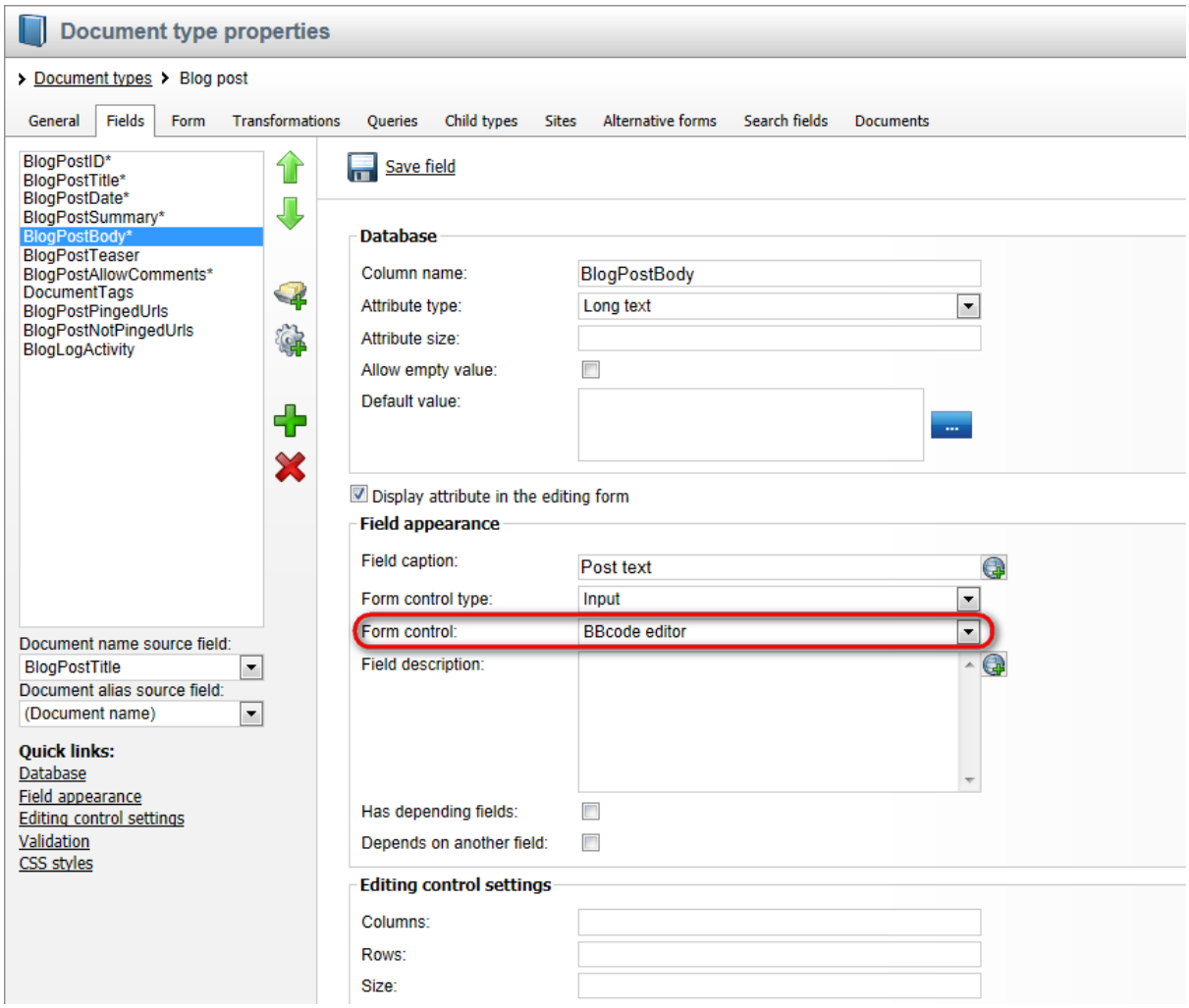
7.11.8.3 Cross site scripting (XSS)

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**.



This editor is displayed by selecting *BBcode editor* as the value of the **Form control** property when defining fields for document types (or other objects):



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) %>
```

7.11.8.4 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.

7.11.9 User registration

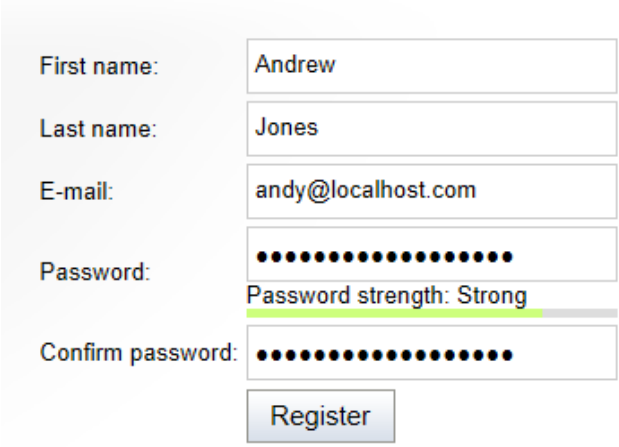
7.11.9.1 Available registration web parts

There are three different ways how you can let site visitors register on your site:

- Using the **Registration form** web part. Read more here.
- Using the **Custom registration form** web part. Read more here.
- Via **Third-party authentication**. Learn more here.

7.11.9.2 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.



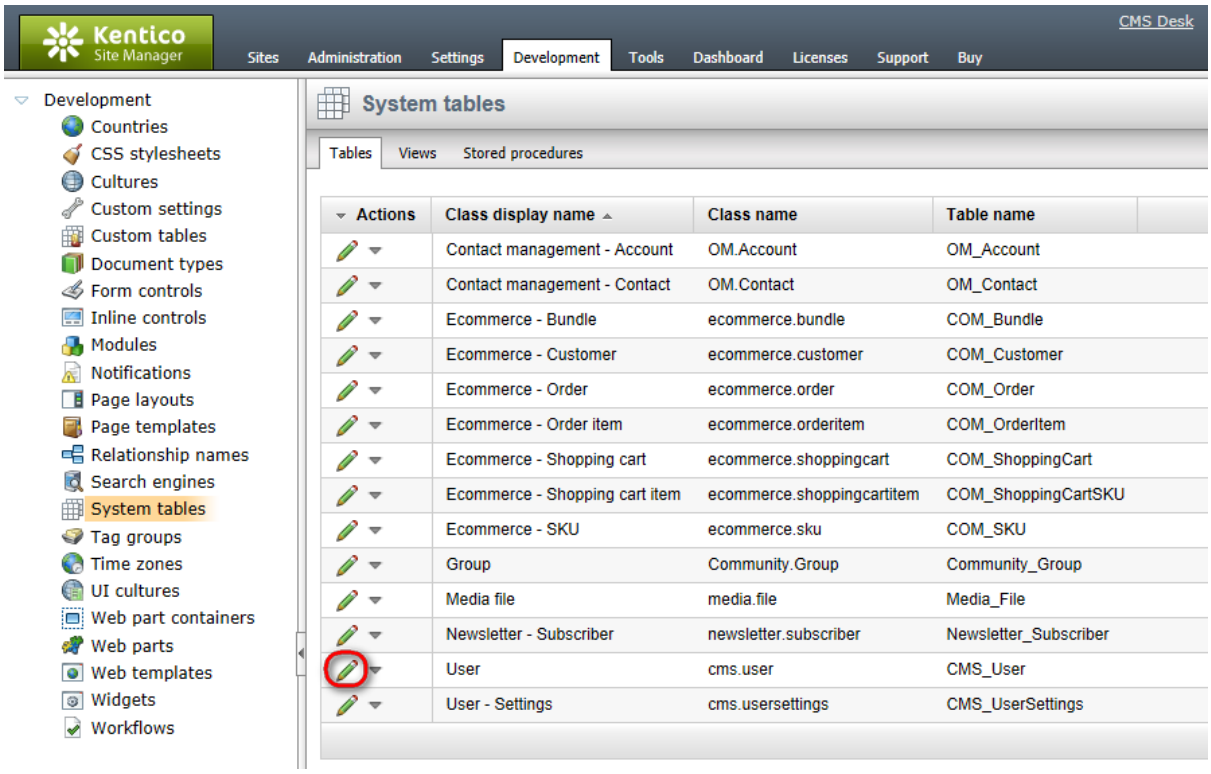
The password field in the form supports validation according to a password policy specified for the website and also calculates and displays the relative strength of the currently entered password. To learn more about how these security options can be configured, please see the Authentication -> Password settings topic.

7.11.9.3 Creating a custom registration form

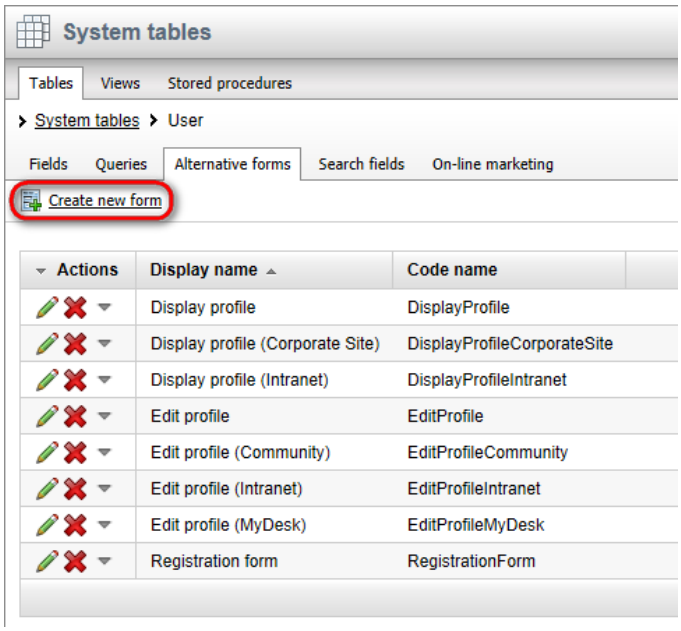
The **Custom registration form** web part can be used in situations when you want to use a different registration form than the default one provided by the **Registration form** web part. This is typically when you want users to provide different details or when you want to customize the form's layout.

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** (✎) the **User** system table.



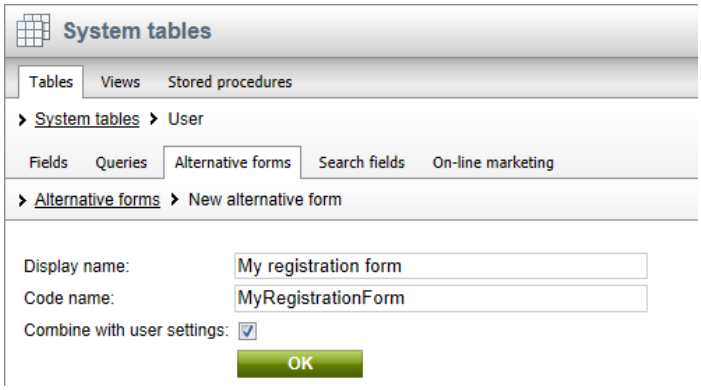
2. Switch to the **Alternative forms** tab and click the **Create new form** link above the list.



3. Fill in the following details:

- **Display name**: My registration form
- **Code name**: MyRegistrationForm
- **Combine with user settings**: checked; this ensures that all default user fields will be available.

and click **OK**.

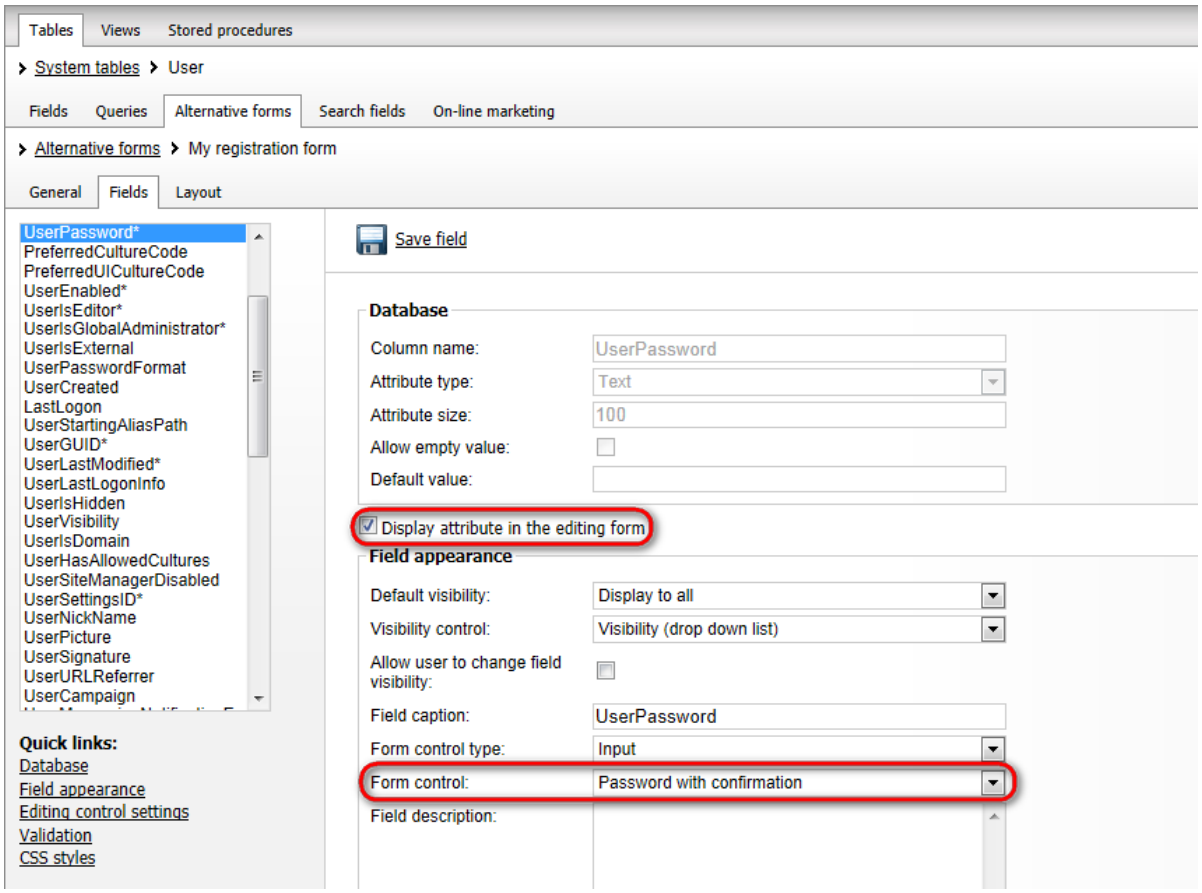


4. Your new form is now created and you are redirected to the editing interface of it. Switch to the **Fields** tab.

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:

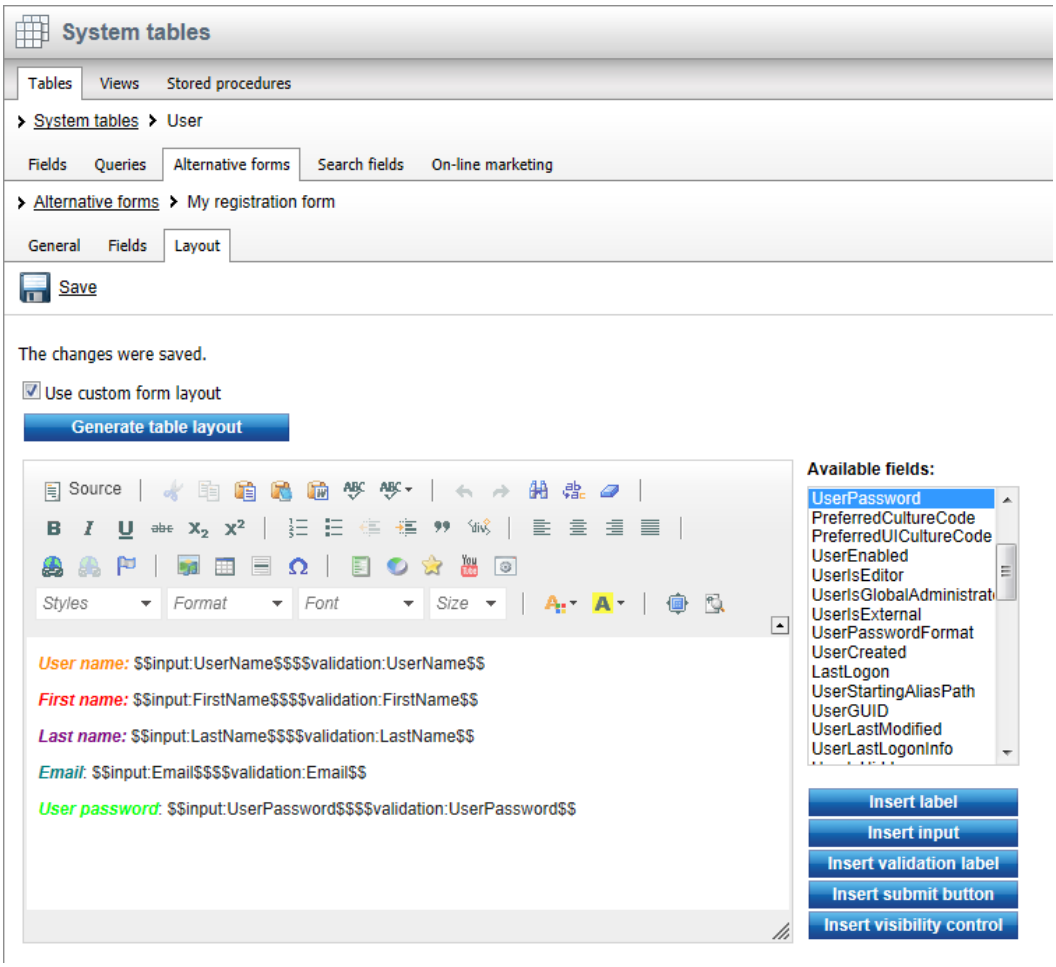
- **UserName**
- **FirstName**
- **Email**
- **UserPassword**

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 field** to confirm the changes to every field. Uncheck the check-box for the remaining fields. For the **UserPassword** field, also change the value of the **Form control** drop-down list to *Password with confirmation*.



5. You can also switch to the **Layout** tab and define the registration form's layout using the built-in WYSIWYG editor.

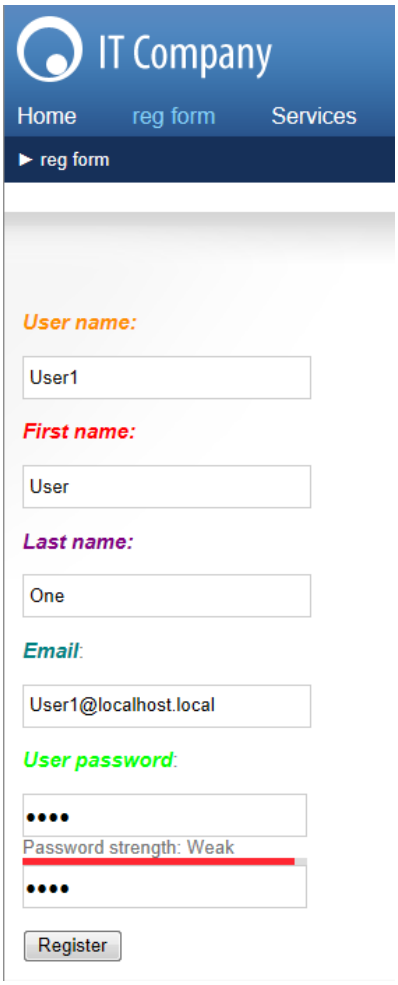
To do it, check the **Use custom layout** check box and click the **Generate table layout** button. A default layout will appear in the editor. Try playing around a bit with the layout or just create something similar to what you see in the screenshot below. Click **Save** when you are finished.



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:

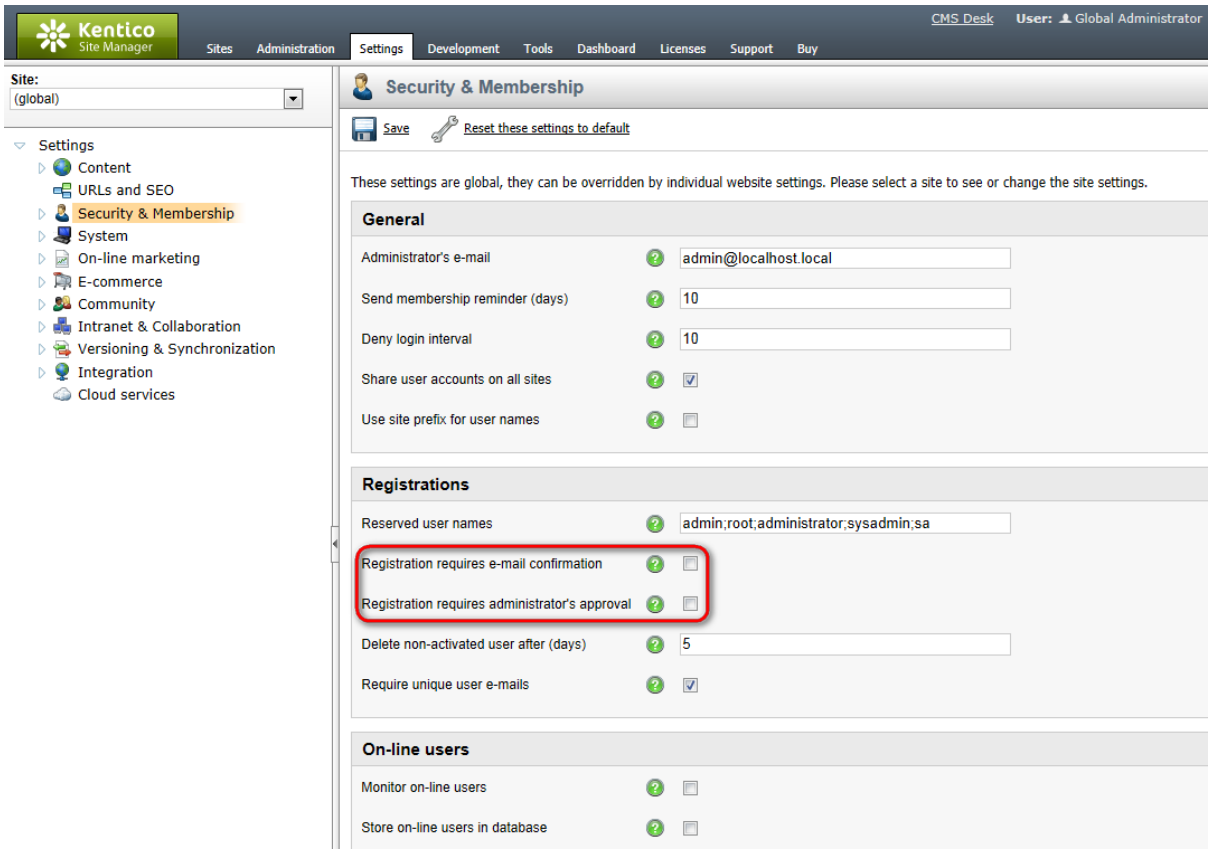
- **Alternative form** - cms.user.MyRegistrationForm

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**.



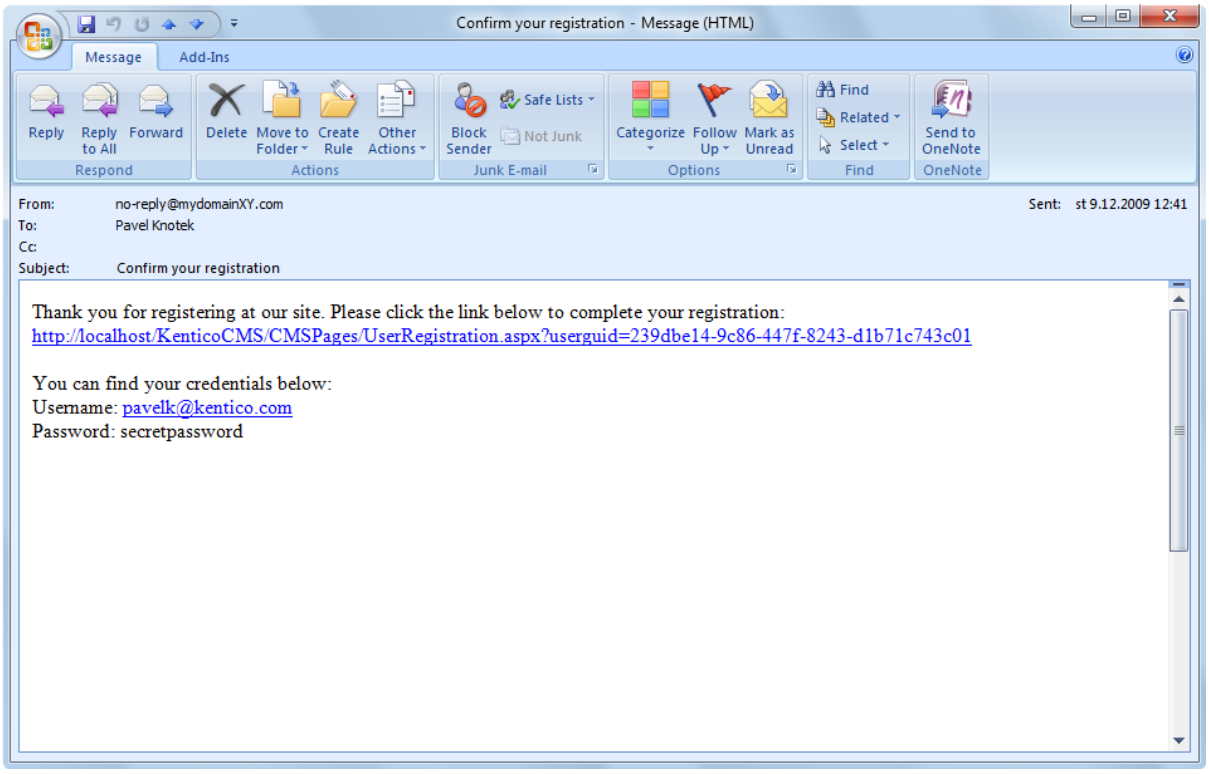
7.11.9.4 Registration approval and double opt-in

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.



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.

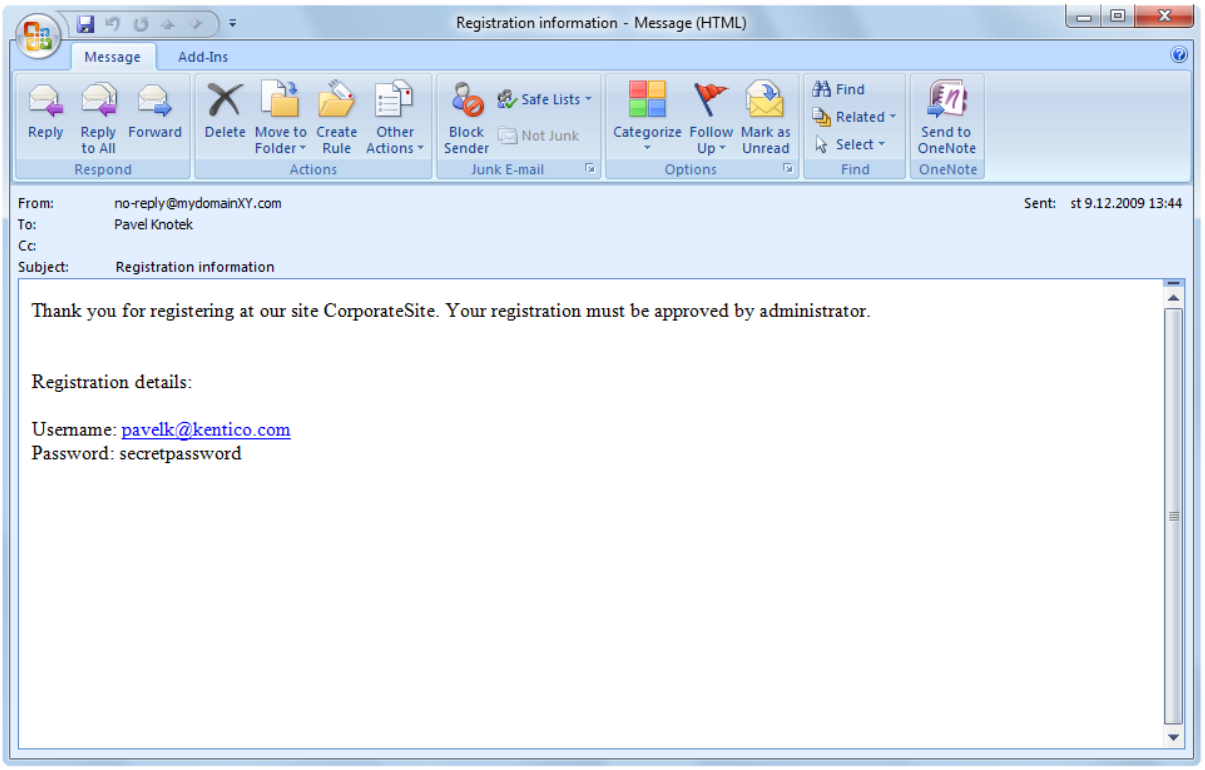
By default, the *~/CMSPages/Dialogs/UserRegistration.aspx* page is used, which displays the following message:

> Your user account is now active. You can sign in using your user name and password. Click here to continue.

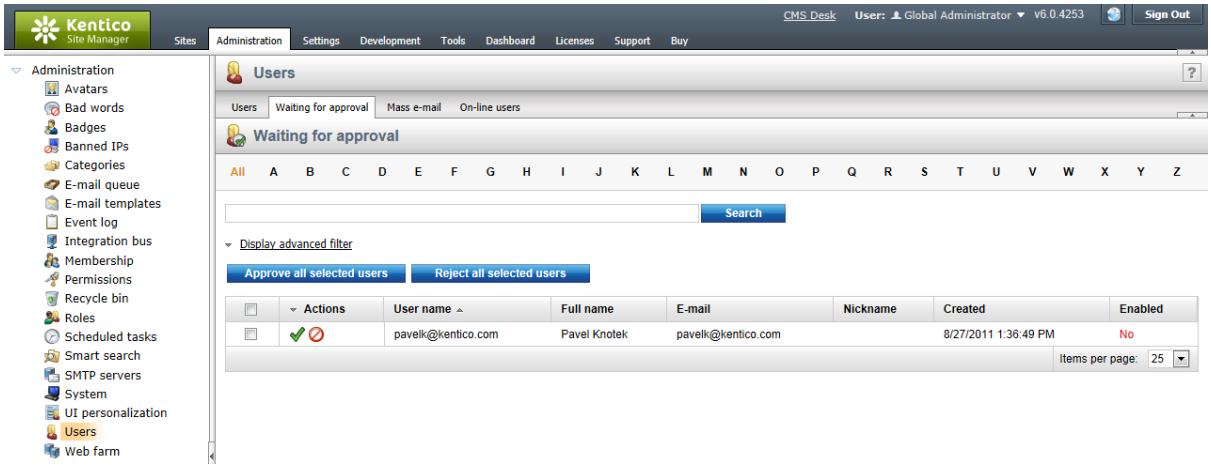
The link at the end of the message will redirect the user to the title page of the website. The user can then log in using the registration details received in the e-mail.

Registration requires administrator's approval

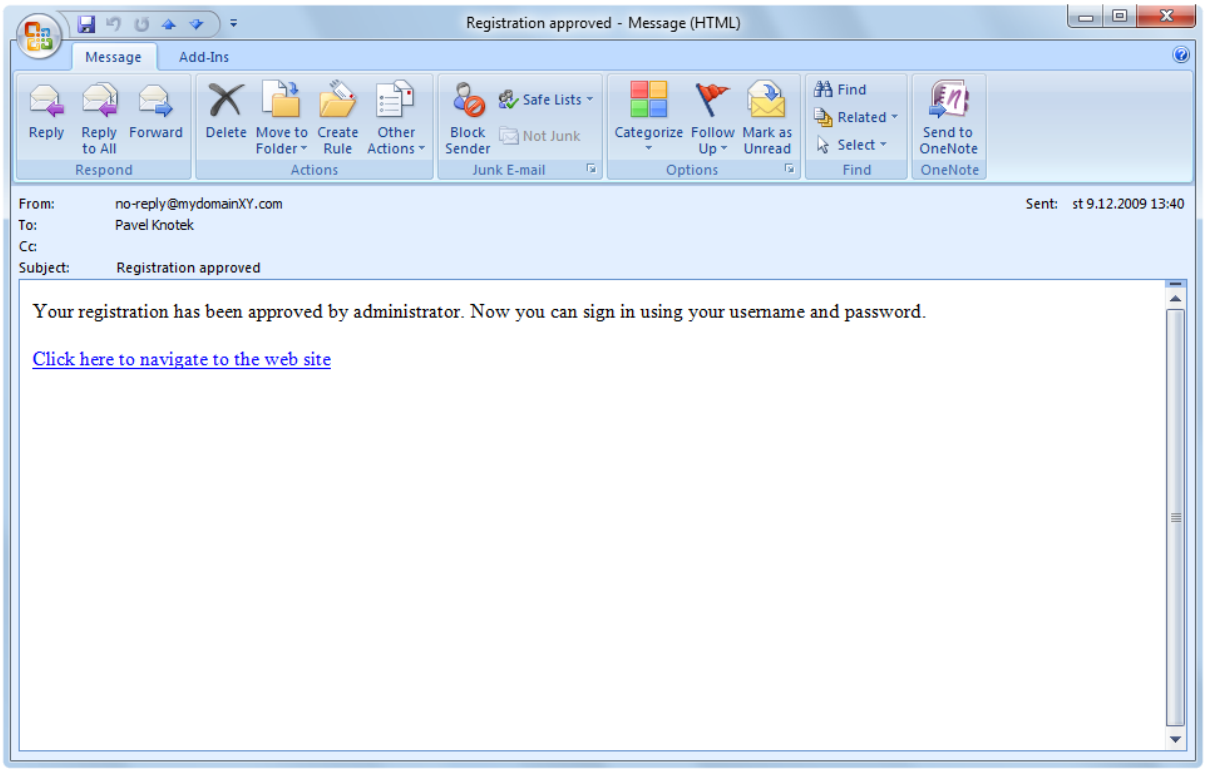
If this option is enabled, users will not be able to sign-in immediately after registration. Their registration will have to be approved by the site administrator. At this point, users will receive an e-mail based on the **Membership - Registration waiting for approval** e-mail template. You can see the default version of the e-mail in the screenshot below.



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.



Enabling both options

In case that you enable both of the options mentioned above, the e-mail with the confirmation link will be 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 is not 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.

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.

7.11.9.5 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.

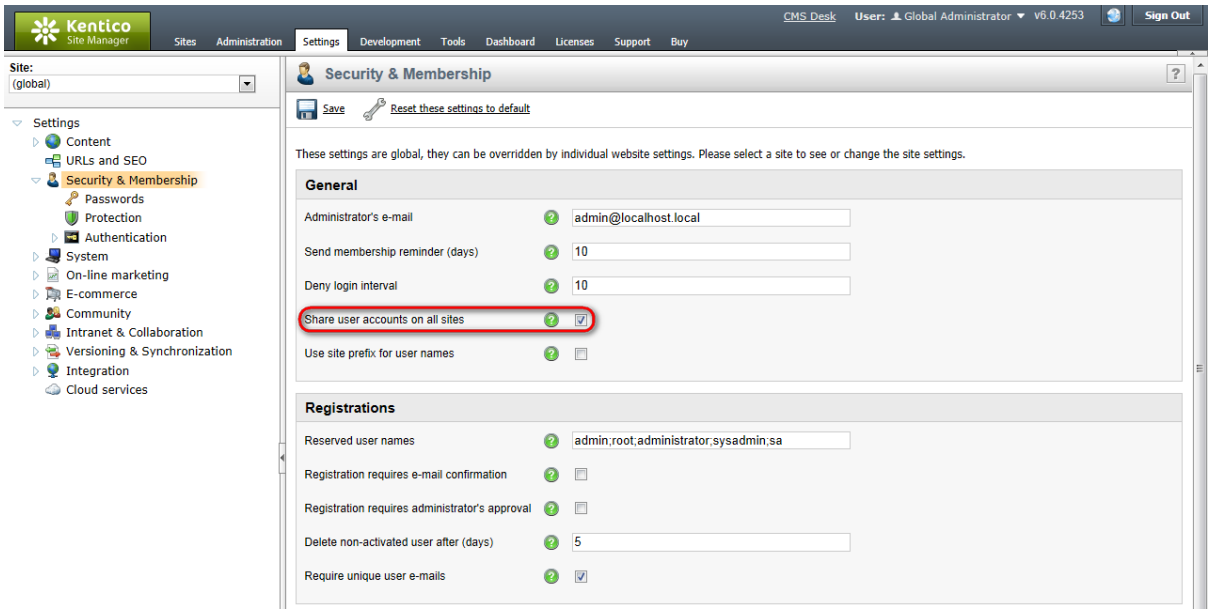
In addition to the special ones listed above, you can also use all other standard macro expressions in the templates. See the Development -> Macro expressions chapter of this guide for more information about macro expressions in Kentico CMS.

7.11.9.6 Shared user accounts

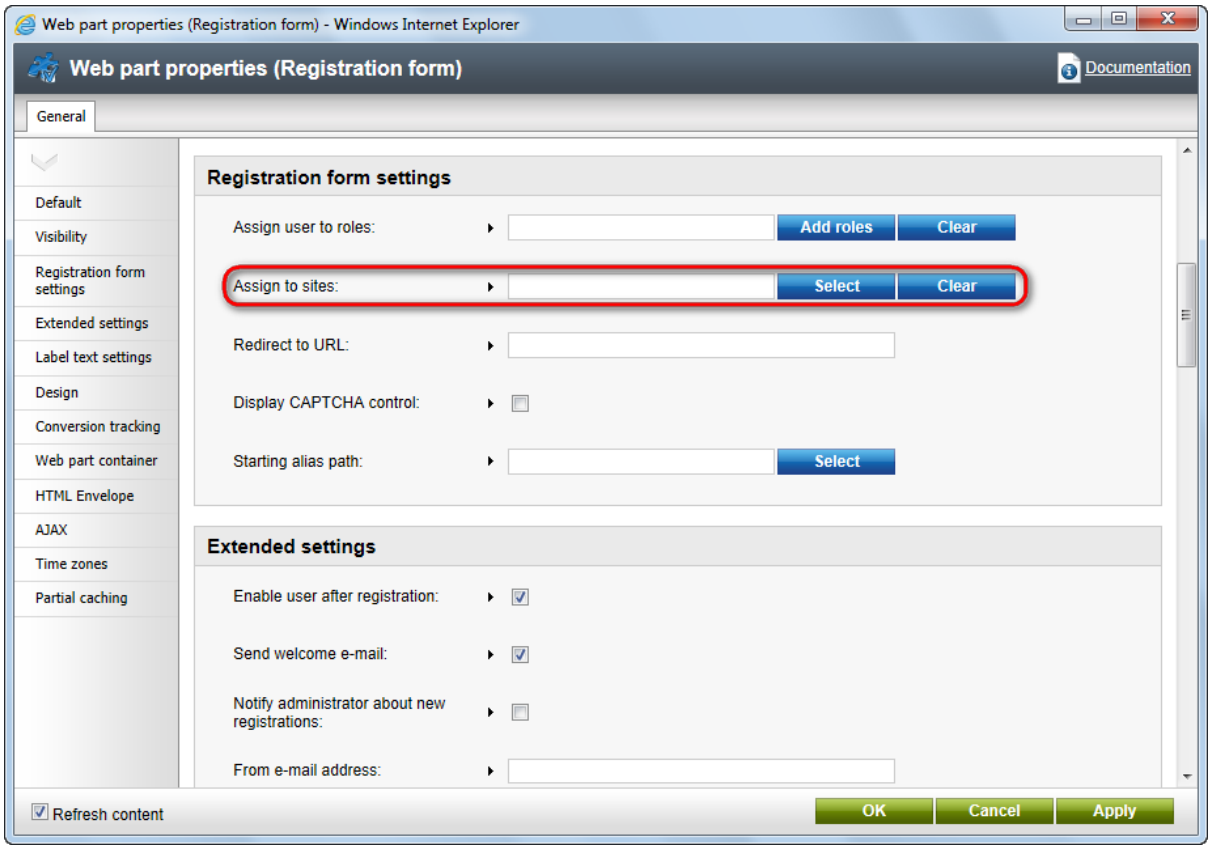
User accounts can be shared among all sites running on one Kentico CMS installation. This means that if a user creates an account on one site, they can automatically log-on to the other sites running on the same installation using the same credentials.

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, which may lead to problems in scenarios where accounts are shared across multiple sites.

7.11.10 Badges

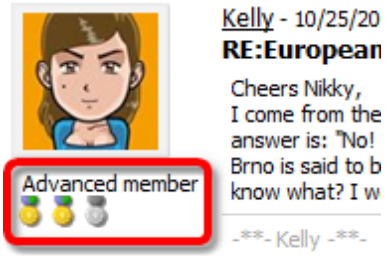
7.11.10.1 Badges

Users can be labeled with badges. These are images with a short text, expressing a user's activity level, importance or role in the context of the website. They are displayed in forum posts, on user public profiles or in your own custom controls.

There are two types of badges:

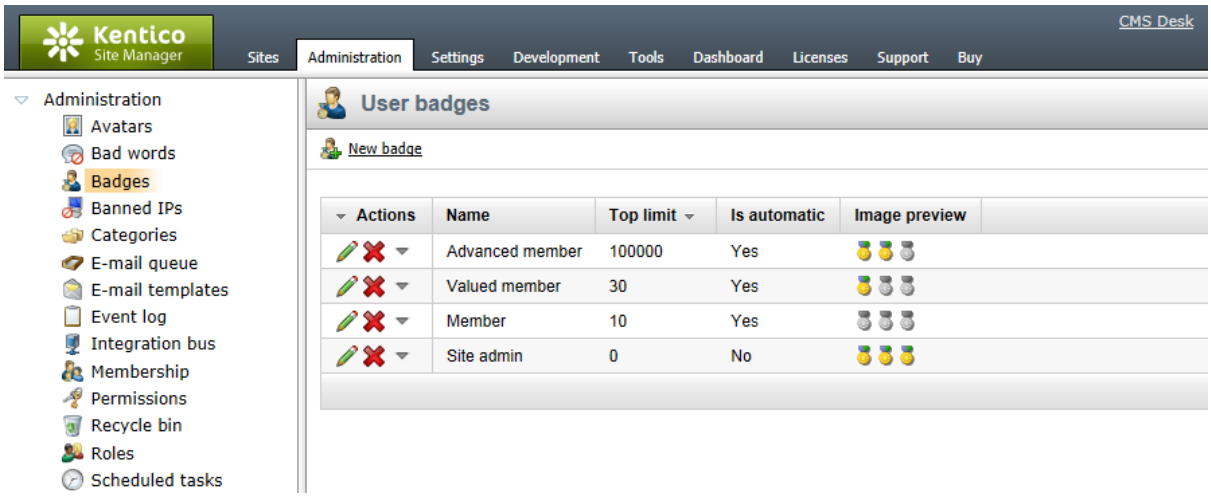
- **Automatic badges** - these are assigned to users based on the number of gained activity points
- **Non-automatic badges** - these are assigned to users manually by site administrators and are assigned permanently, regardless of the number of gained activity points

In the screenshot below, you can see one of the pre-defined badges in a forum post.



7.11.10.2 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** link at the top part of the page.



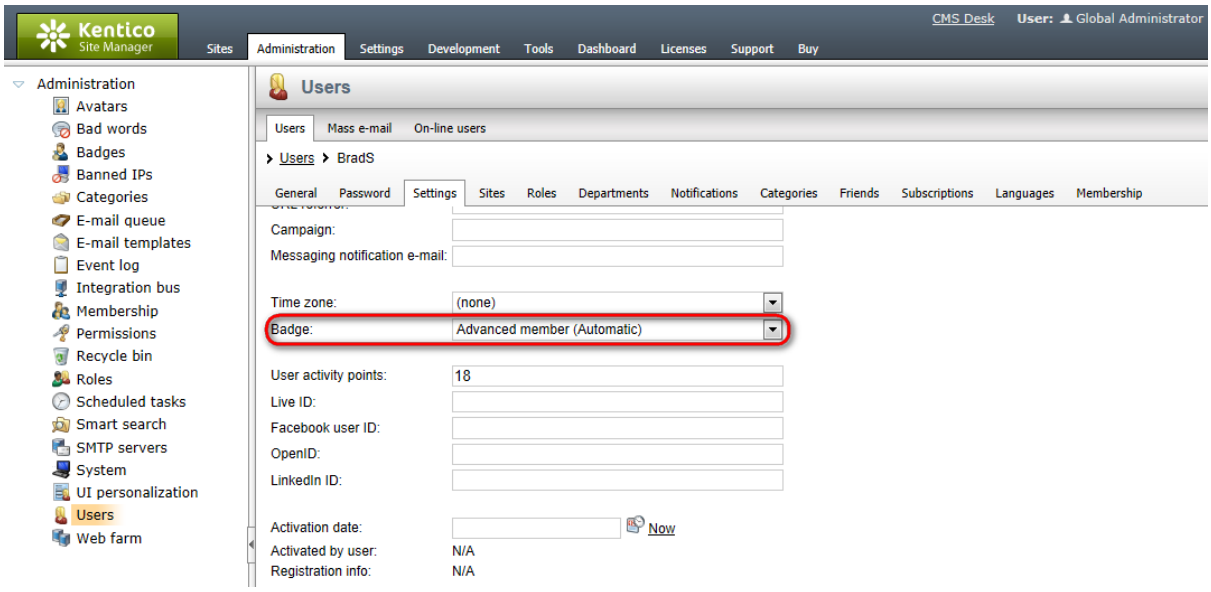
When creating a new badge, the following properties can be defined:

- **Display name** - name of the badge displayed in the administration interface and on the site
- **Code name -** name of the badge used in code
- **Image URL** - URL of the badge's image
- **Is automatic** - if checked, the badge will be assigned to users automatically based on the number of gained activity points; if unchecked, the badge can be assigned to users by site administrator and will remain assigned permanently, regardless of the number of gained activity points
- **Top limit** - number of activity points required for the user to get the badge; applies only for automatic badges

7.11.10.3 Assigning badges to users

Site administrators can assign badges to users in **Site Manager -> Administration -> Users -> Edit (✎) user -> Settings**. It can be done by the **Badge** drop-down list.

This is typically used to assign users with non-automatic badges. However, automatic badges can be assigned to users this way too.

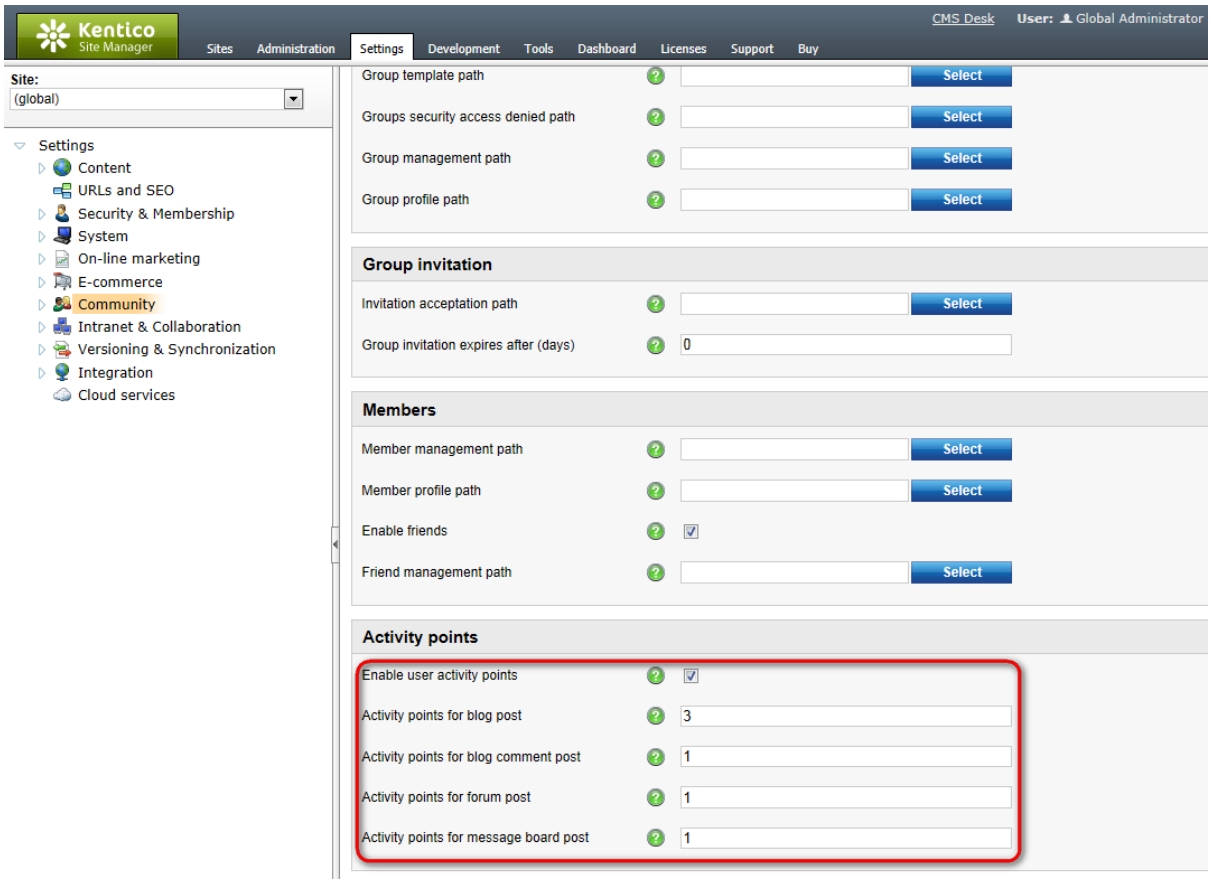


7.11.10.4 Activity points

Users can gain activity points for their activity on the site. Based on the number of gained activity points, user can be assigned with automatic badges. For this feature to be functional, you have make the following settings in **Site Manager -> Settings -> Community**:

- **Enable user activity points** - enables the activity points feature

- **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



7.11.10.5 Available form controls

The following form controls can be used in your custom controls to display users' badges:

- **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.

7.11.10.6 Badges internals and API

7.11.10.6.1 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.

7.11.10.6.2 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	☐
		☐

7.11.10.6.3 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.

7.11.10.6.4 API examples

7.11.10.6.4.1 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;
```

7.11.10.6.4.2 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);
}
```

7.11.10.6.4.3 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;
    }
```

7.11.11 Custom field visibility

7.11.11.1 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**.

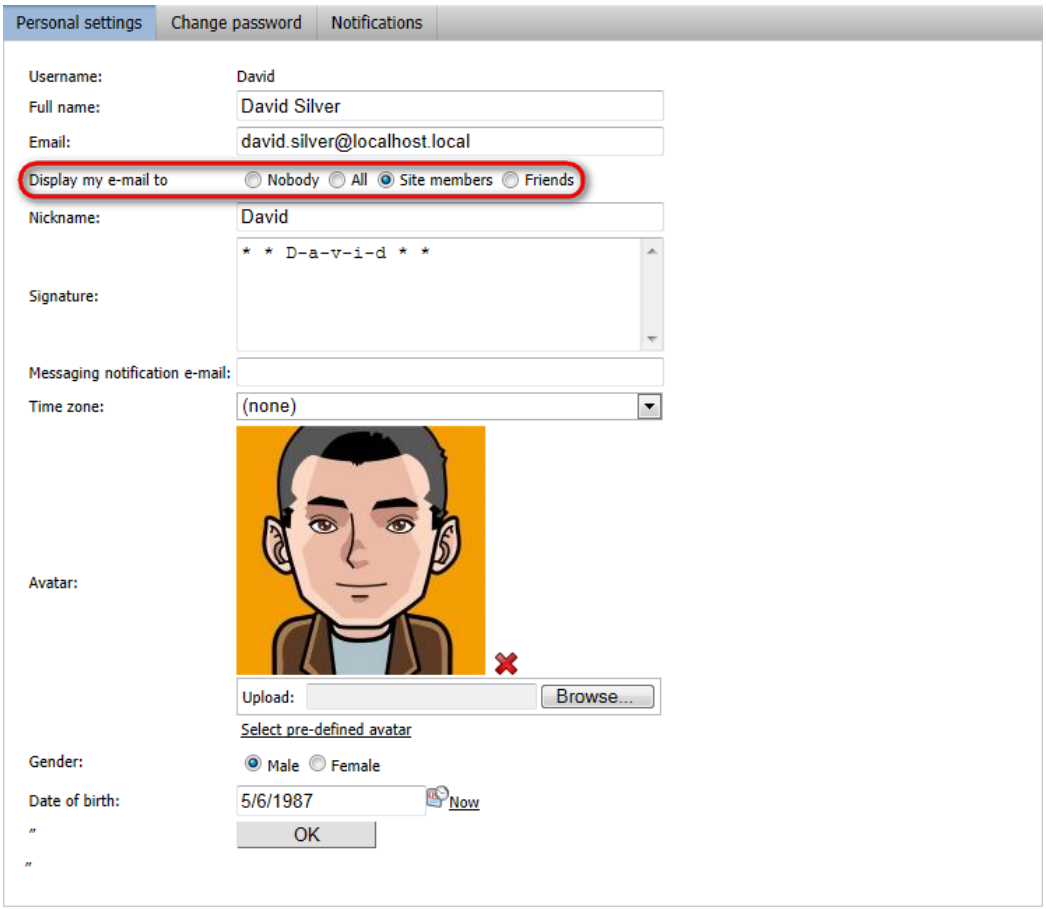
1. Run the **Community site** and sign out of the administration interface. Log on to the site using the **Sign in** form on the right. Enter *David* with blank password and click **Log on**.

2. Once logged in, the **Shortcuts** menu will be displayed where the Sign in form previously was. Click the **Edit my profile** link, you will be redirected to the user's profile editing page.

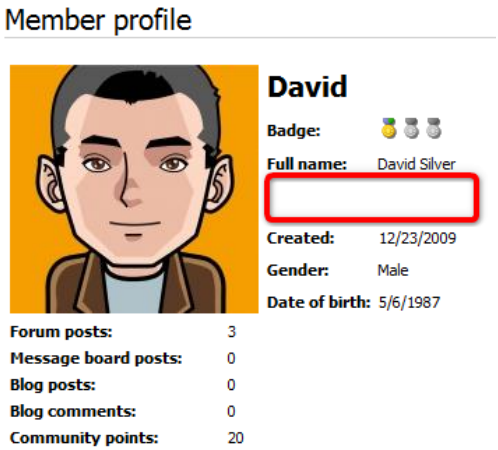
3. You can see a drop down list next to the **E-mail** field, as in the screenshot below. Using this control, users can define to whom will the e-mail address be displayed. The following four options are available:

- **Display to none** - the field will not be displayed to anyone
- **Display to all** - the field will be displayed to everyone
- **Display to authenticated** - the field will be displayed only to authenticated users
- **Display to friends** - the field will be displayed only to the user's friends

Choose **Display to authenticated** and click **OK**.



4. You have just configured the user's profile so that only authenticated users can see their 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 will be visible.

7.11.11.2 Enabling visibility controls

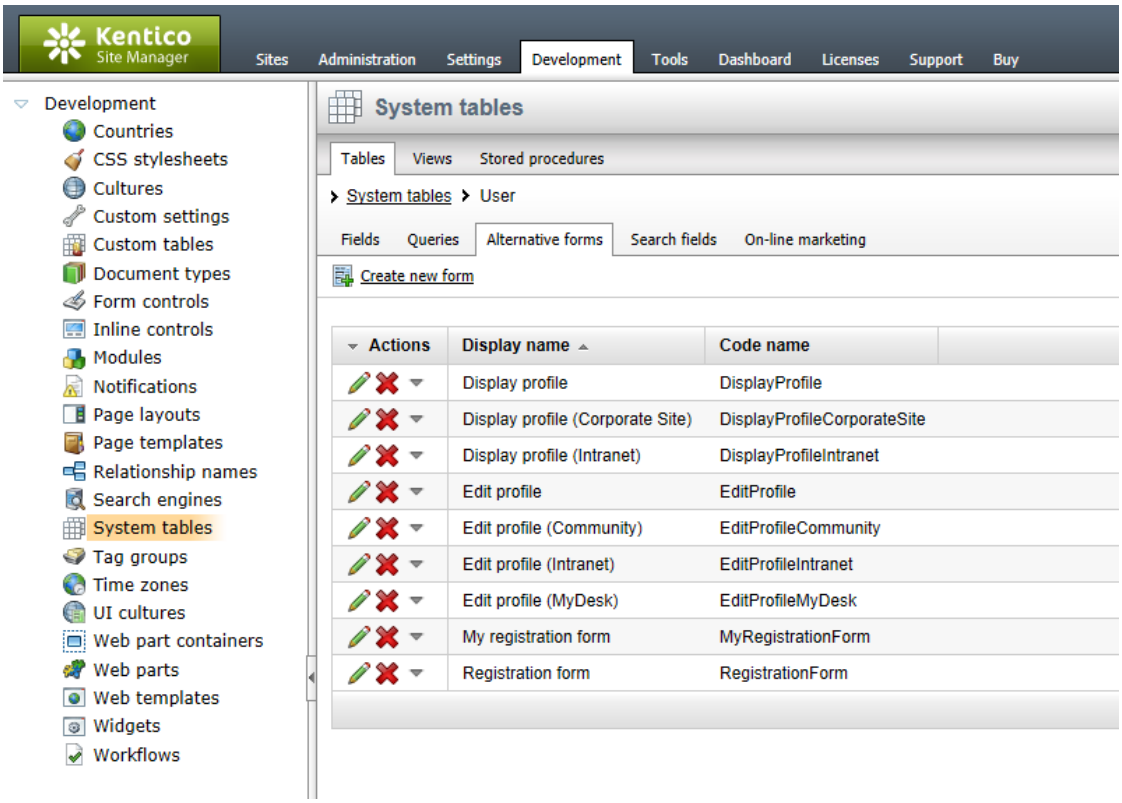
The visibility selection drop-down list can be added to any field of the user's profile, not only the e-mail field as shown in the example in the How it works topic. This can be changed in **Site Manager -> Development -> System tables**. Choose to **Edit** (✎) the **User (CMS_User)** system table and 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 where information about registered users is stored. Each of the four alternative forms that you see in the list is used in a specific situation when the system accesses the table:

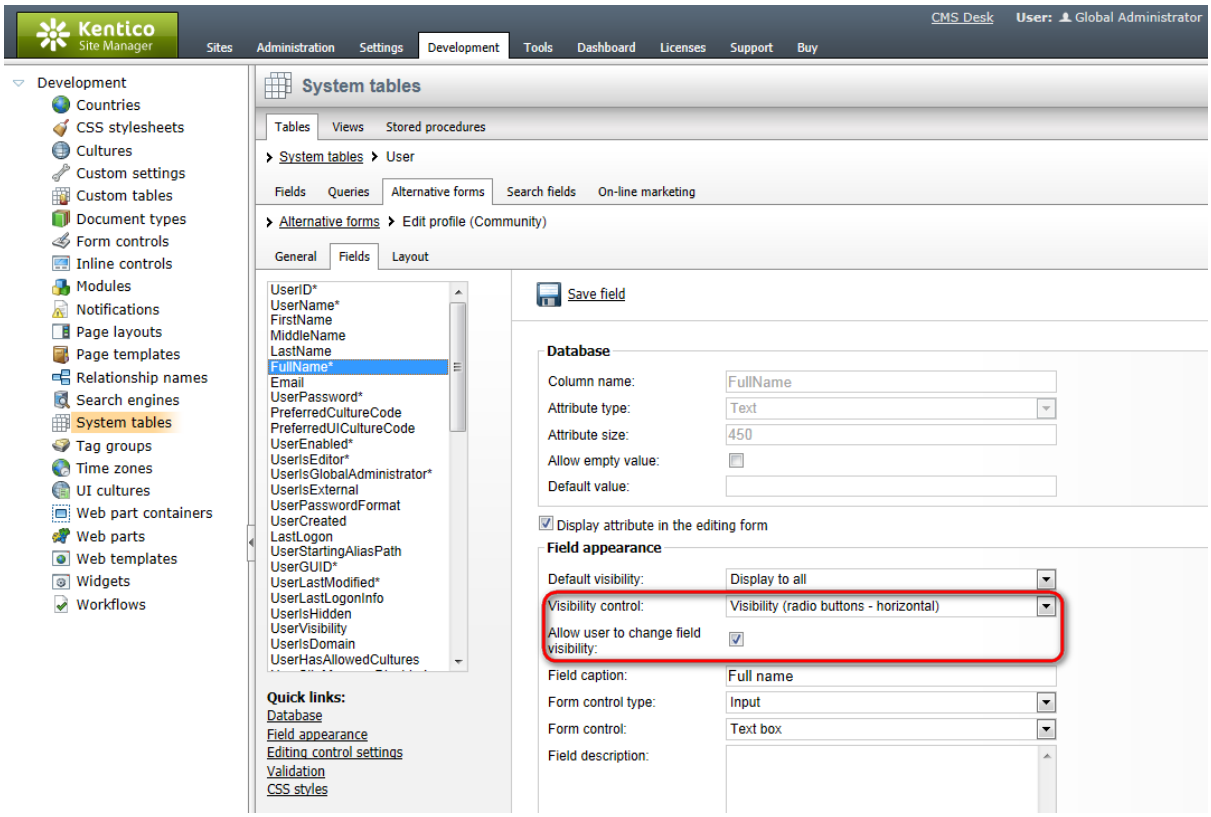
- **Registration form** - when registering a new user using the **Custom registration form** web part.
- **Display profile** - when displaying a user's public profile using the **User public profile** web part.
- **Edit profile** - when editing a user's profile using the **My account** web part.
- **Edit 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. Choose to **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** check-box and click **Save field**.

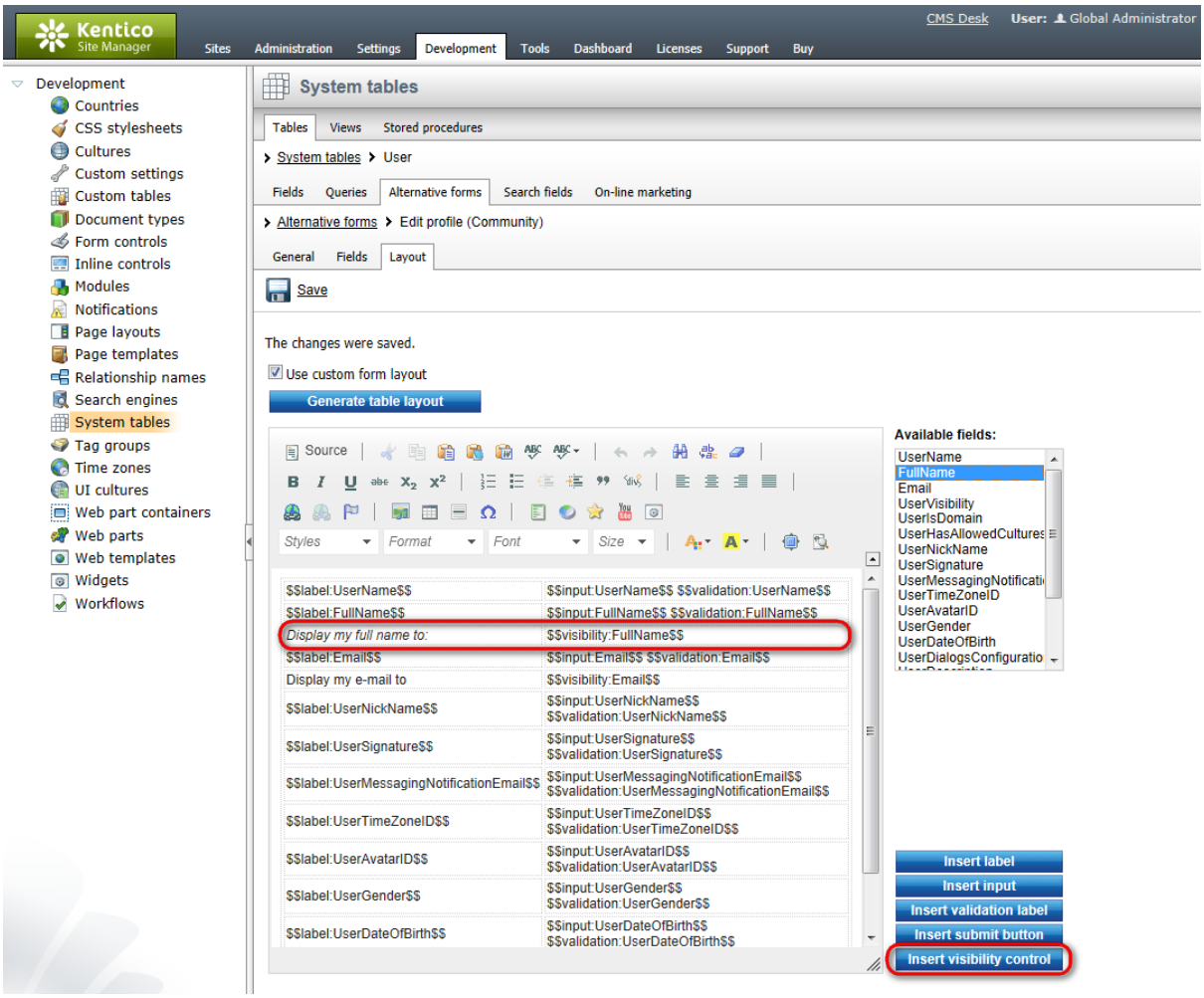


2. Switch to the **Layout** tab. We will need to create another line in the table and add the visibility control into it.

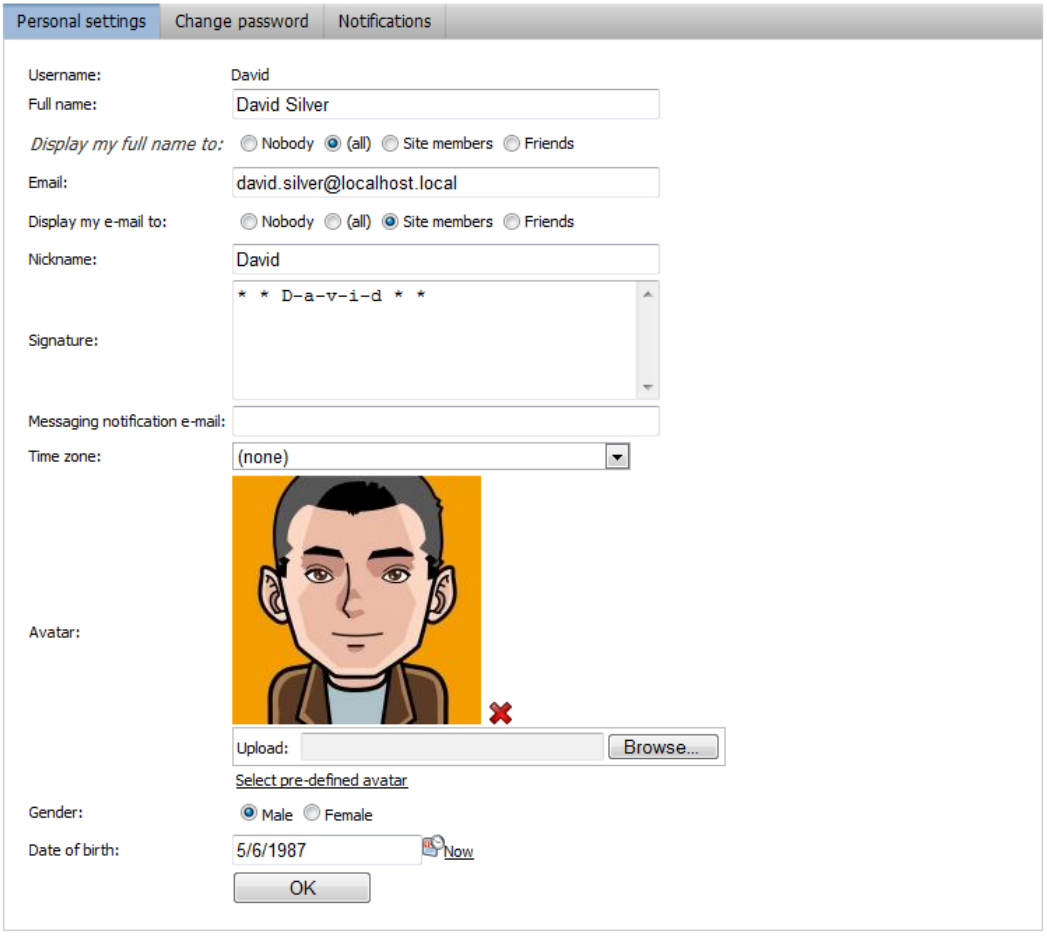
Place the cursor into the FullName line (where the $$label:FullName$$ value is). Right-click and choose **Row -> Insert Row After** from the context menu. Into the first column of the new row, enter: *Display my full name to:*

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. Now sign out of the administration interface and sign in to the website as *David* with blank password. Click the **Edit** my profile link in the right **Shortcuts** menu. You should see a drop-down list next to the full name field, so that now the users can determine the visibility of the Full name field.



7.11.11.3 Use in custom form layouts

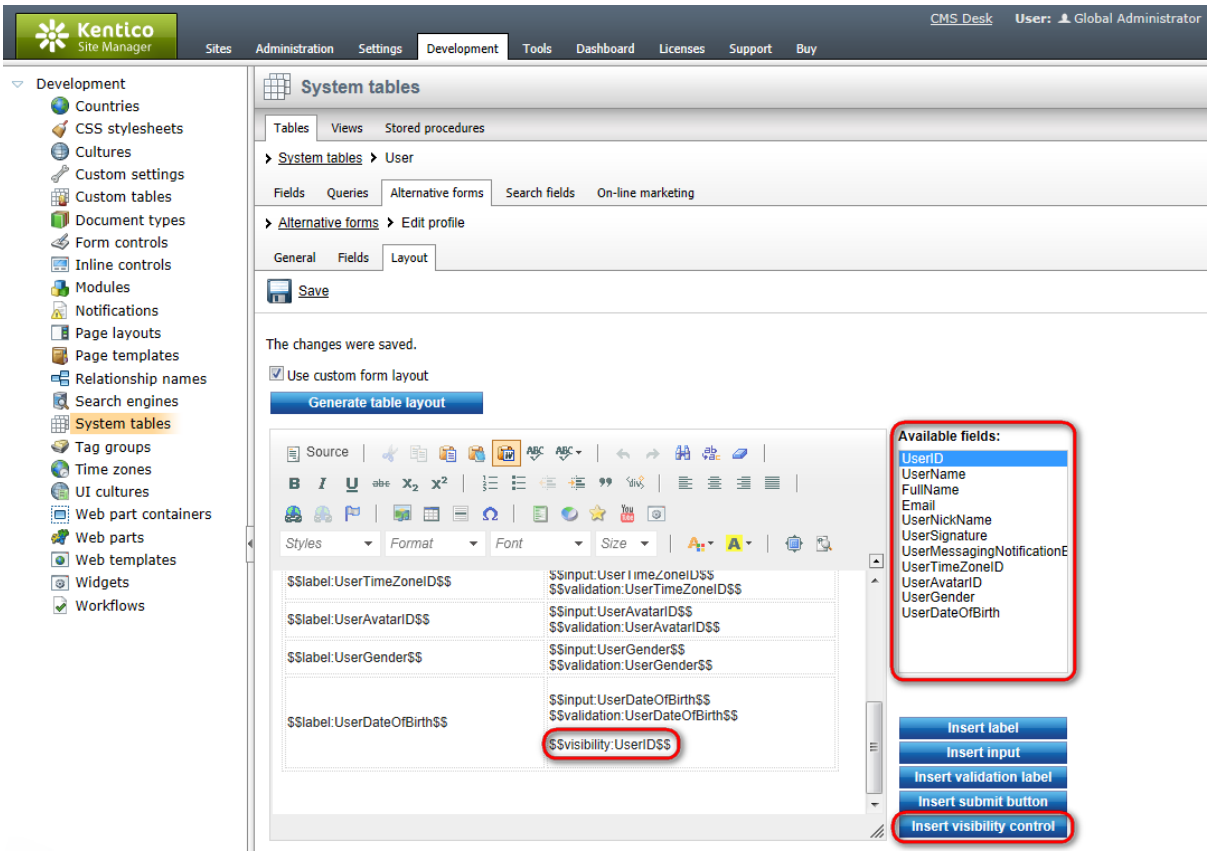
If you want to define a custom form layout and use the visibility drop-down list for some field, you have to do the following two things:

- 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.



7.11.11.4 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.

7.11.12 Authentication

7.11.12.1 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>
```

Additional configuration options related to user passwords may also be defined in the system, as described in the Password settings topic.

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

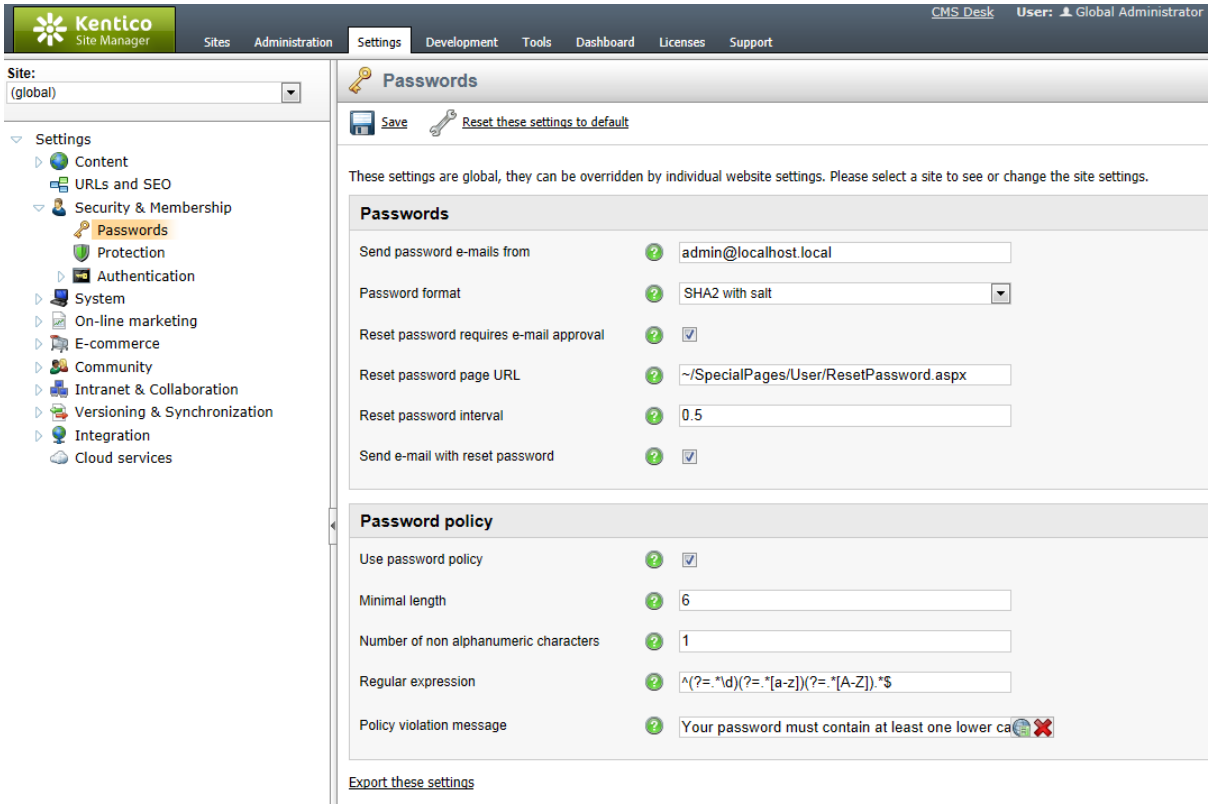
Please see the Windows authentication (Active Directory) sub-chapter to learn more.

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.

7.11.12.2 Password settings

Passwords are a critical part of any authentication process, so Kentico CMS provides various password-related configuration options that you can adjust according to the level of security required by your website. These settings can be found in **Site Manager -> Settings -> Security & Membership -> Passwords**.



Password format

There are multiple different formats that can be used to store passwords in the database. They may either be saved 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*.

This option is recommended, since it provides the best security. A salt is a string that is appended to passwords before they are hashed, which helps protect against dictionary or other types of brute force attacks, and also ensures that every user has a different password hash (even if multiple users have the same password). The randomly generated 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 **/configuration/appSettings** section of your web.config file:

```
<add key="CMSPasswordSalt" value="SaltText" />
```

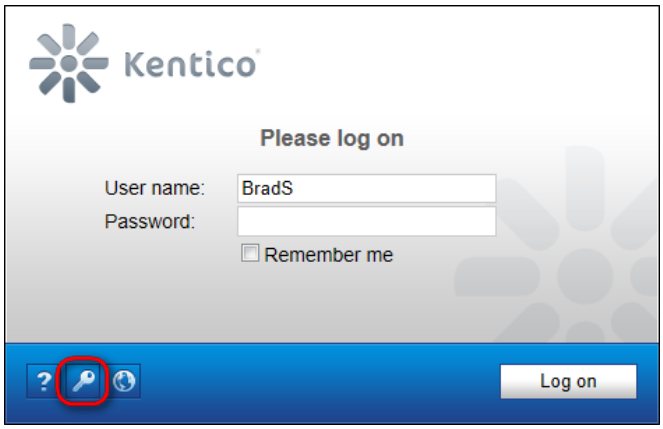
The value of this key will be added to the salt after the user GUID.

If you change the password format, please keep in mind that this only affects how future passwords are stored and existing passwords will remain unchanged. It is necessary to set all passwords again so that they are stored in the new format. For this reason, it is recommended to set the appropriate format directly after the installation, before you create user accounts or allow users to start registering.

Forgotten password recovery

If a user forgets their password, they may retrieve or reset it as long as they have access to the e-mail address specified for the given account. This can be done by submitting a request through one of the website's logon forms.

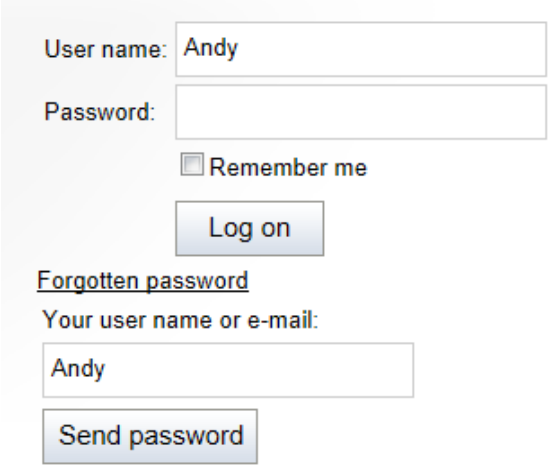
By default, a forgotten password button is included on the logon page of the CMS Desk and Site Manager administration interface.



If necessary, the button can be hidden by adding the following key to the **/configuration/appSettings** section of your web.config file:

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

On the live site, password recovery may be performed 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:

If it is *disabled*, then users who request their password will receive an e-mail containing the password directly. The current password will be sent if it is stored in plain text, or a new one will be generated for the user account if a hashed password format is used.

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 and the reset link is only valid temporarily (also, it can only be used to reset the password once).

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 into a form. The URL of the link contains a token in its query string that automatically identifies the user whose password should be changed. For security reasons, this link will only remain valid for a limited amount of time after the initial request (12 hours by default). Once the link expires, it will no longer be possible to change the password through it. You can set the exact length of the expiration interval in the **Reset password interval** setting.

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. The URL of the custom page must then be entered 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 recover by editing the database directly:

- If you're using plain text passwords, you can read or change the password in the **CMS_User** table.
- If you're using hashed passwords, you need to set the password in the **CMS_User** table to an empty string. Then, you can sign in to **Site Manager** with an empty password and change the password as necessary.

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 should have the prefix trimmed out through 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.

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.

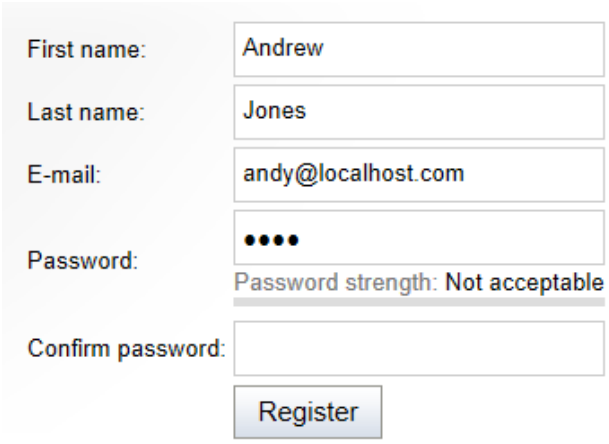
To enforce a password policy on your website, enable the **Use password policy** setting. The specific rules of the policy can be configured through the remaining settings in the category:

- **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 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.

7.11.12.3 Windows authentication (Active Directory)

7.11.12.3.1 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 2007, 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="-" />*

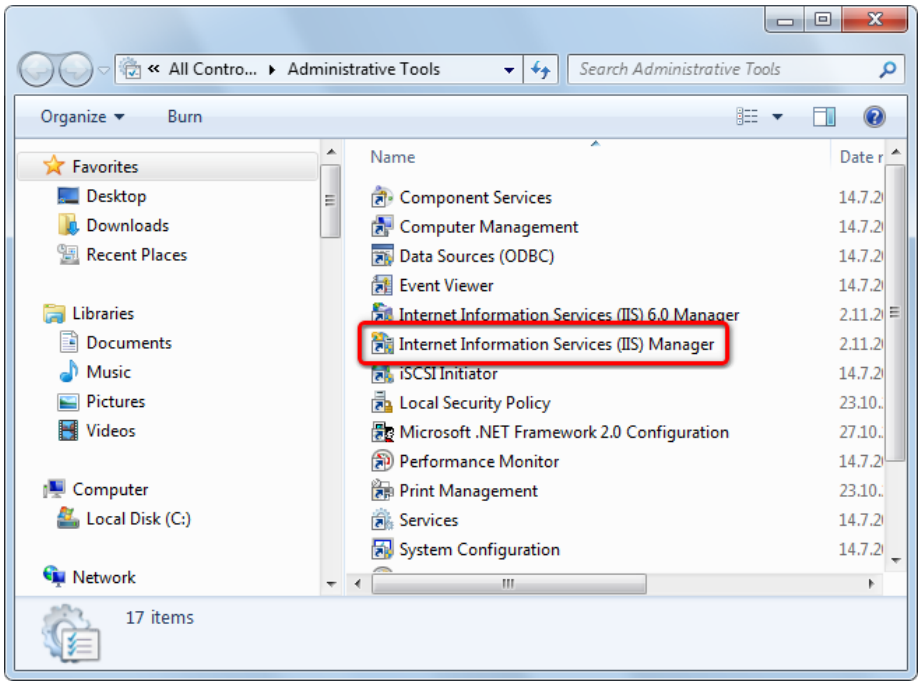
If you want to achieve the same functionality as in **older versions** of Kentico CMS (**office\username**), forbidden characters replacement can be turned off completely using the following two keys. This may cause problems when using wildcard URLs with user names in the wildcard part and is therefore not recommended.

- *<add key="CMSEnsureSafeUserNames" value="false" />*
- *<add key="CMSEnsureSafeRoleNames" value="false" />*

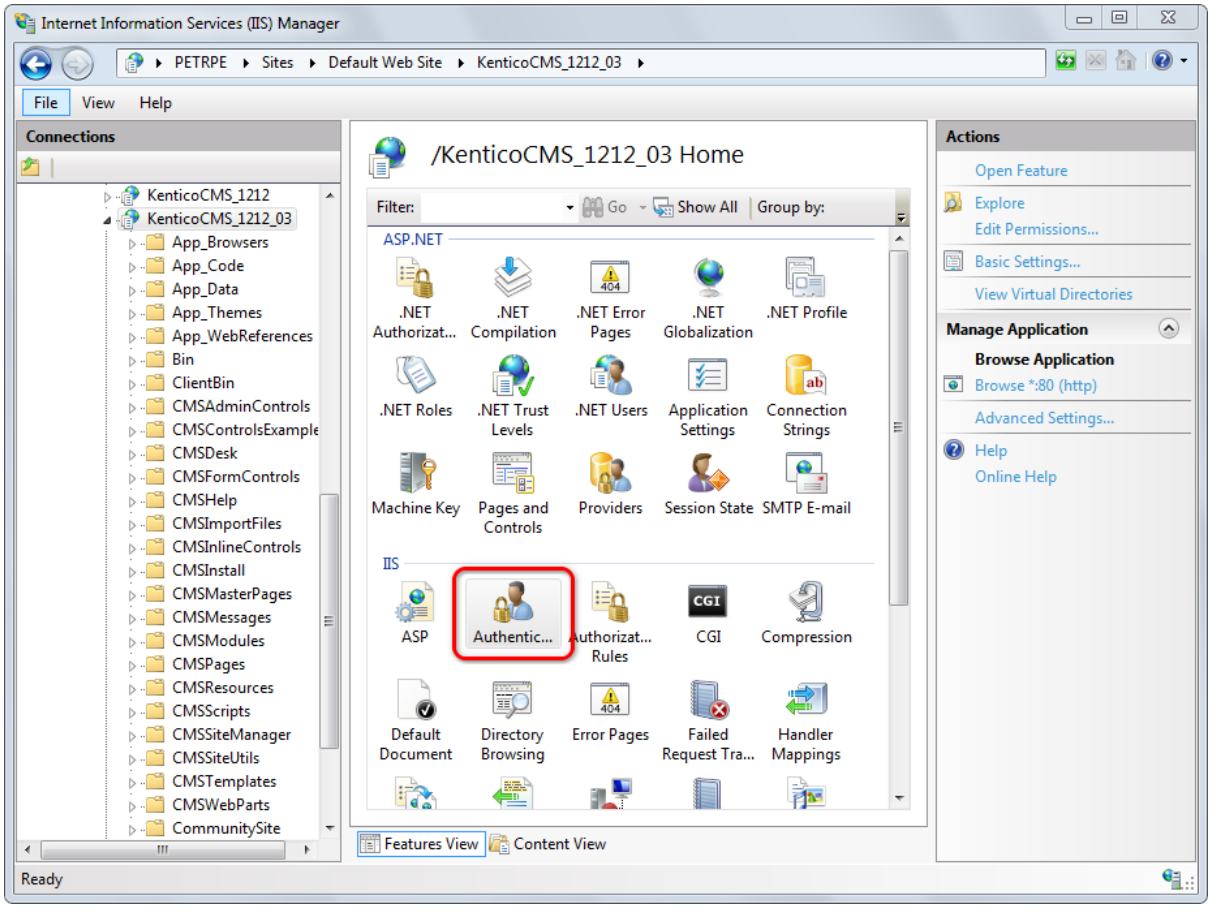
7.11.12.3.2 Windows authentification on Windows 7/2008 R2/Vista (IIS7 or higher)

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:

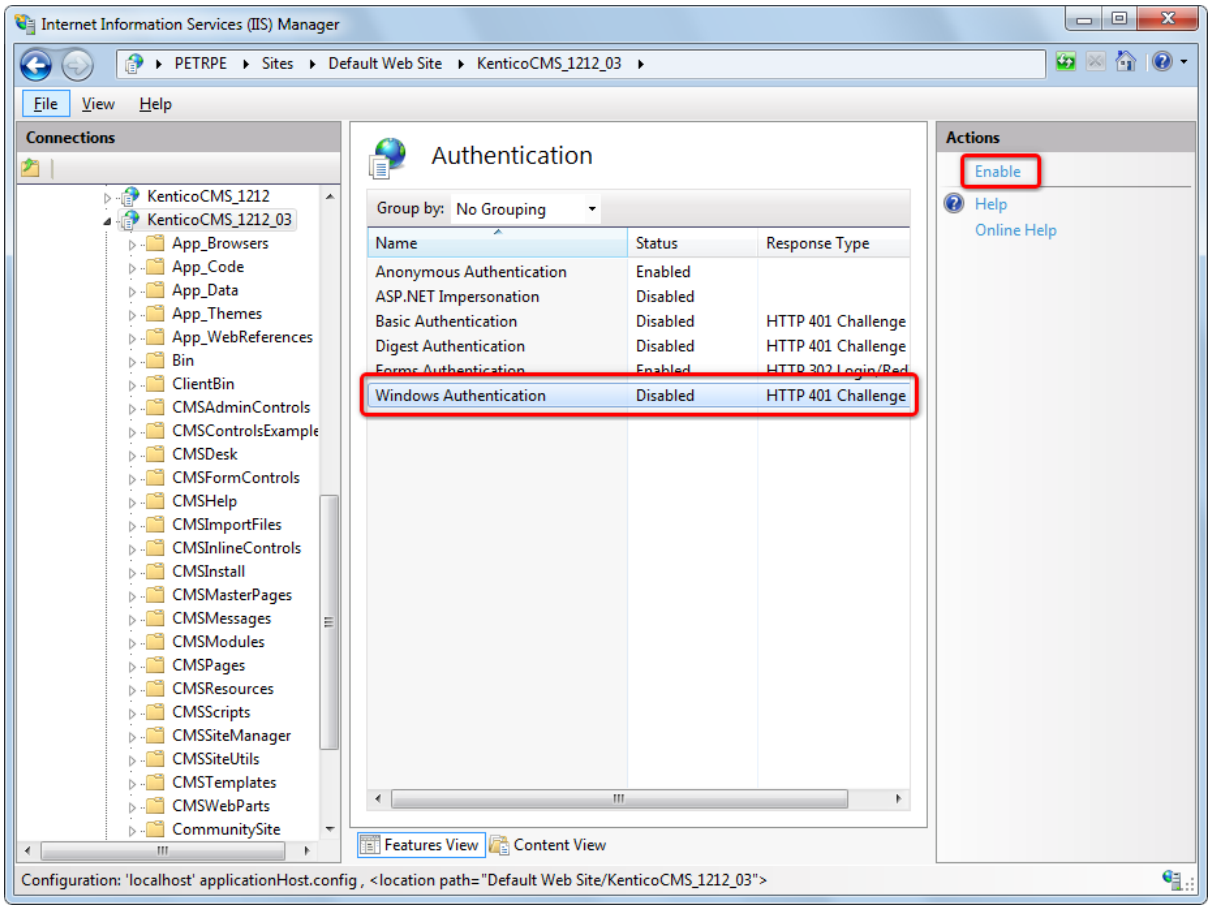
1. Go to **Start** -> **Control Panel** -> **Administrative Tools** and start the **Internet Information Services (IIS) Manager**.



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.



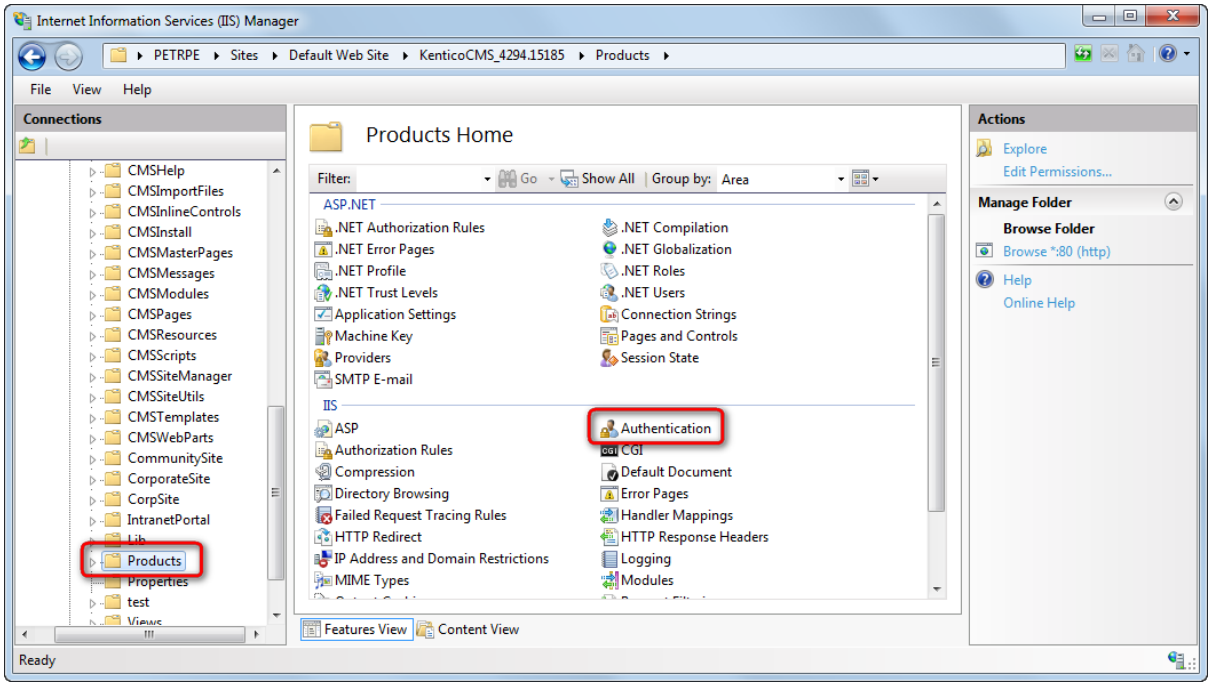
7.11.12.3.3 Securing a web site section using Windows authentication

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:

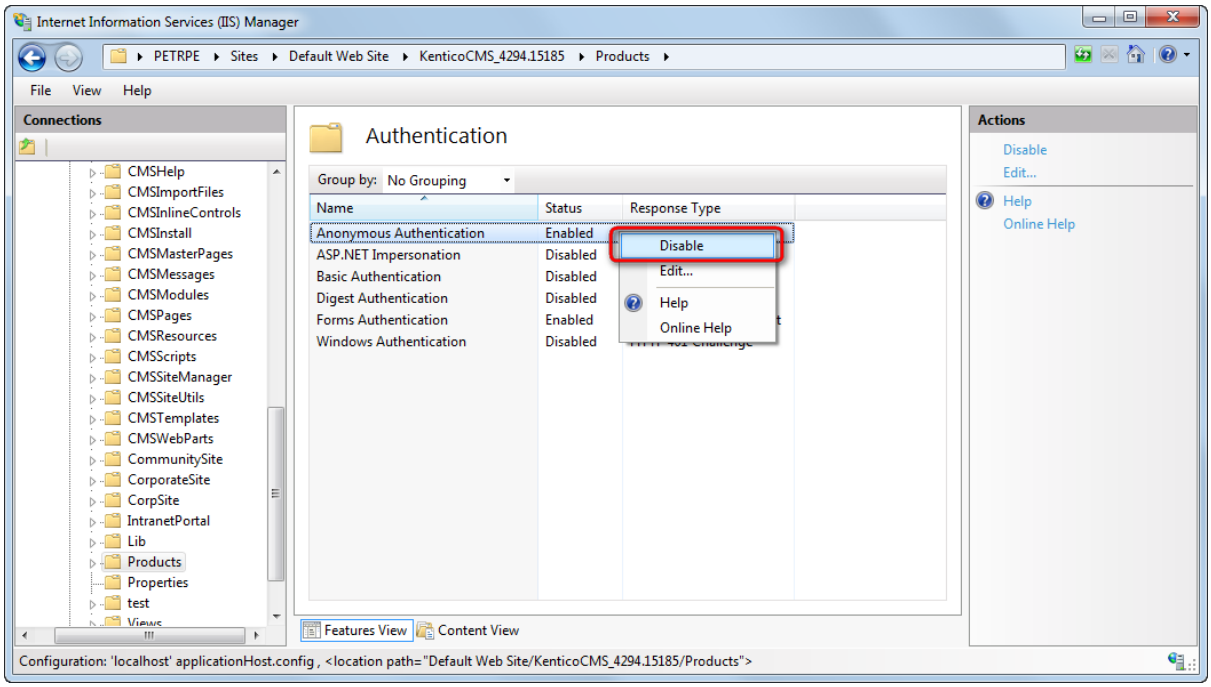
1. Locate your web project on the disk (typically *c:\Inetpub\wwwroot\<web project>*). Create a new directory in your web project's folder and give it the same name as the filename in the document's URL.

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.



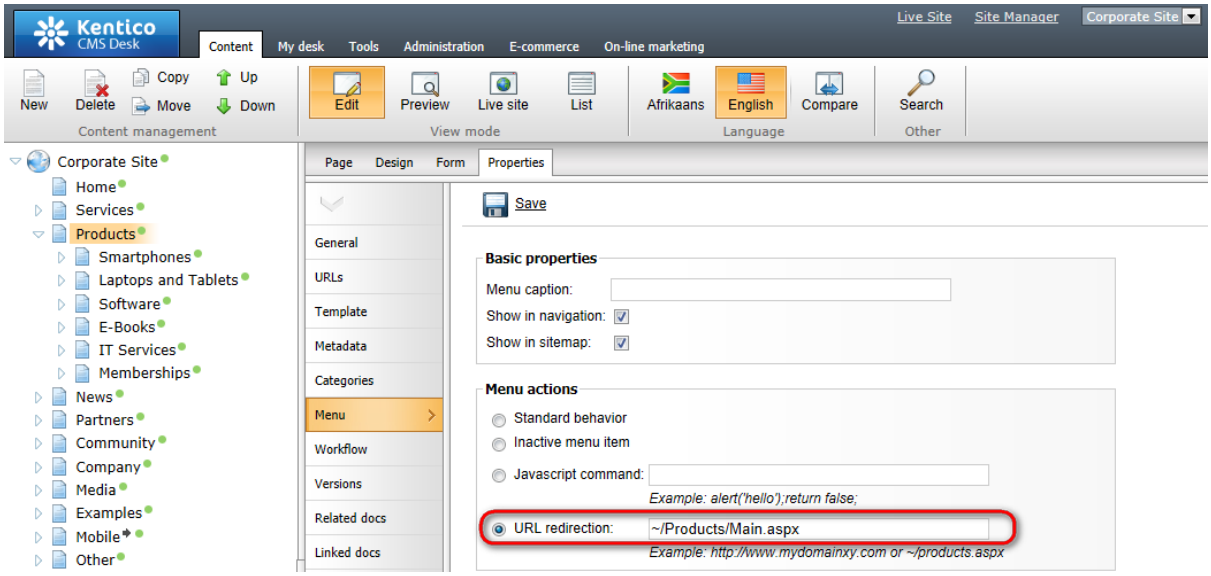
4. Open the web.config file of your web project and change value of the **mode** attribute of the

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.



7.11.12.4 Configuring mixed mode authentication

Mixed mode authentication enables users to sign in to your website using both Windows authentication and standard forms authentication at the same time.

To enable this authentication mode, you have to modify your *web.config* file so that the **connectionStrings**, **membership** and **roleManager** sections are the same as the code sample below:

```
<connectionStrings>
  <add name="CMSADConnectionString" connectionString="<LDAP connection string>" />
</connectionStrings>

<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>
```

The <LDAP connection string> part highlighted in green in the code example above should be replaced with the actual connection string. It should be entered in the format shown below. The first part is the full domain. In the second part, the same domain is divided using the DC parts:

```
LDAP://mydomain.example.com/DC=mydomain,DC=example,DC=com
```

User name and password need to be entered only on Windows XP, in the following format:

- **Username**: <domain name>\user
- **Password**: relevant password

When you have entered this code to your *web.config*, users can log in using their Active Directory user name (without domain) and password, or using their standard Kentico CMS user name and password.

You can also enable 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" />
```

7.11.12.5 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 learn more about custom authentication in the Security handler (CustomSecurityHandler class) chapter.

7.11.12.6 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" ...="" />
```

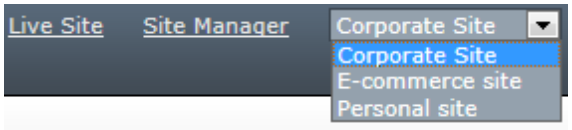
Windows Authentication

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 = UserInfoProvider.GetUserAuthenticationUrl(userInfo, "/default.aspx");
// Redirect user to target URL and authenticate him
URLHelper.Redirect(url);
```

7.11.12.7 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

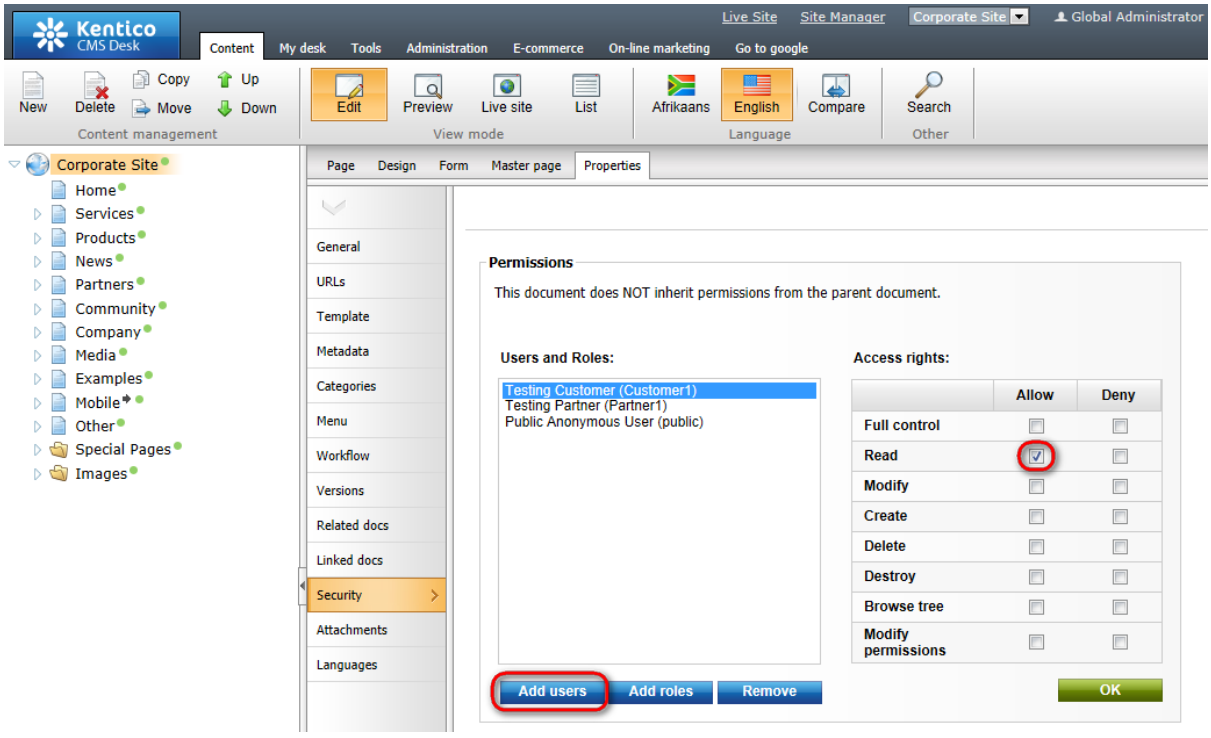
Click **OK**. Go to the **Sites** tab and assign the user to the *Corporate Site*. Go to the **Roles** tab and add the user to the **Partners** role.

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.

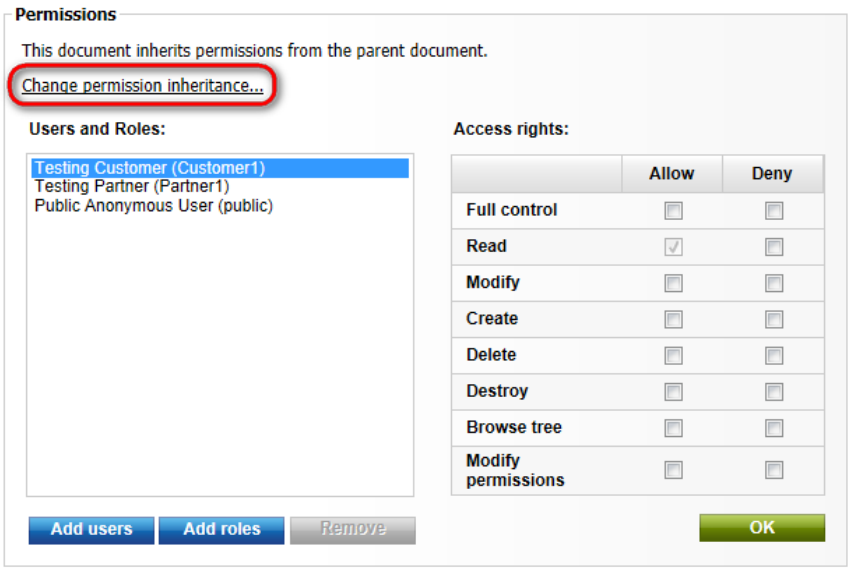
6. Switch to **CMS Desk -> Content** and click the root of the content tree. Switch to **Properties -> Security**.

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.

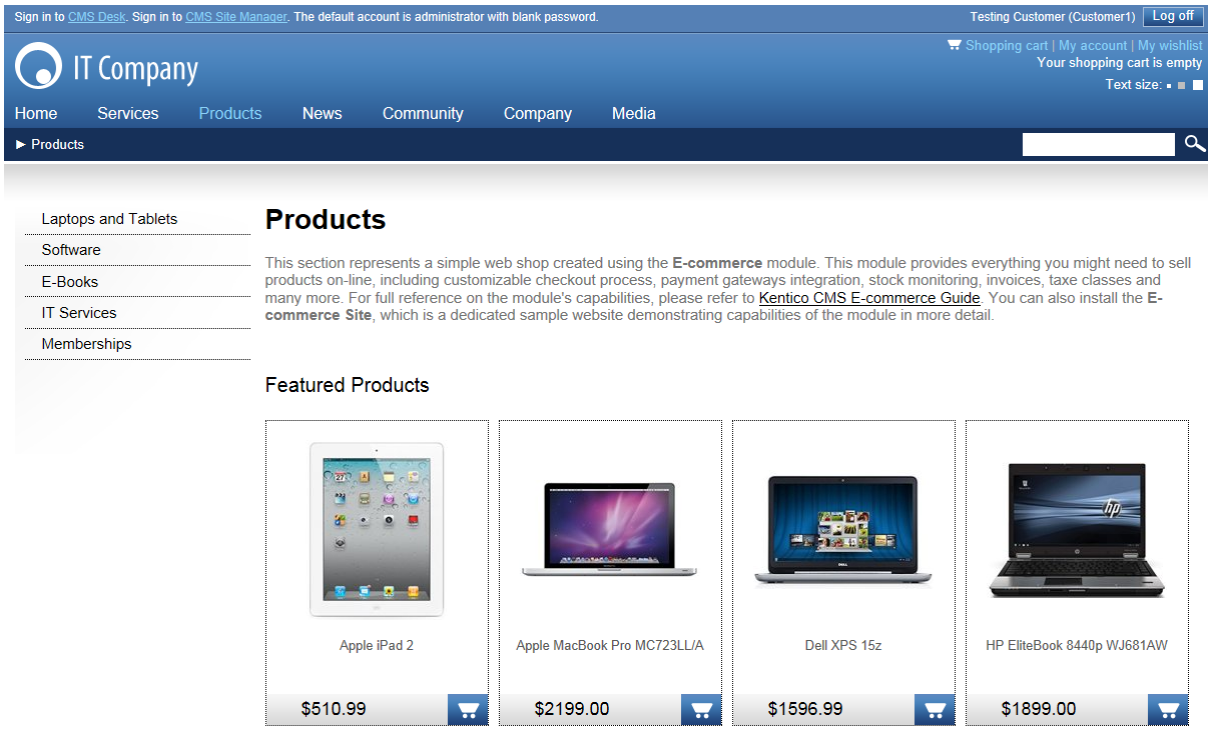
9. Repeat the last two steps for the **Laptops and Tablets** page. Also deny the **Smartphones** category for the **Customers** role.

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.

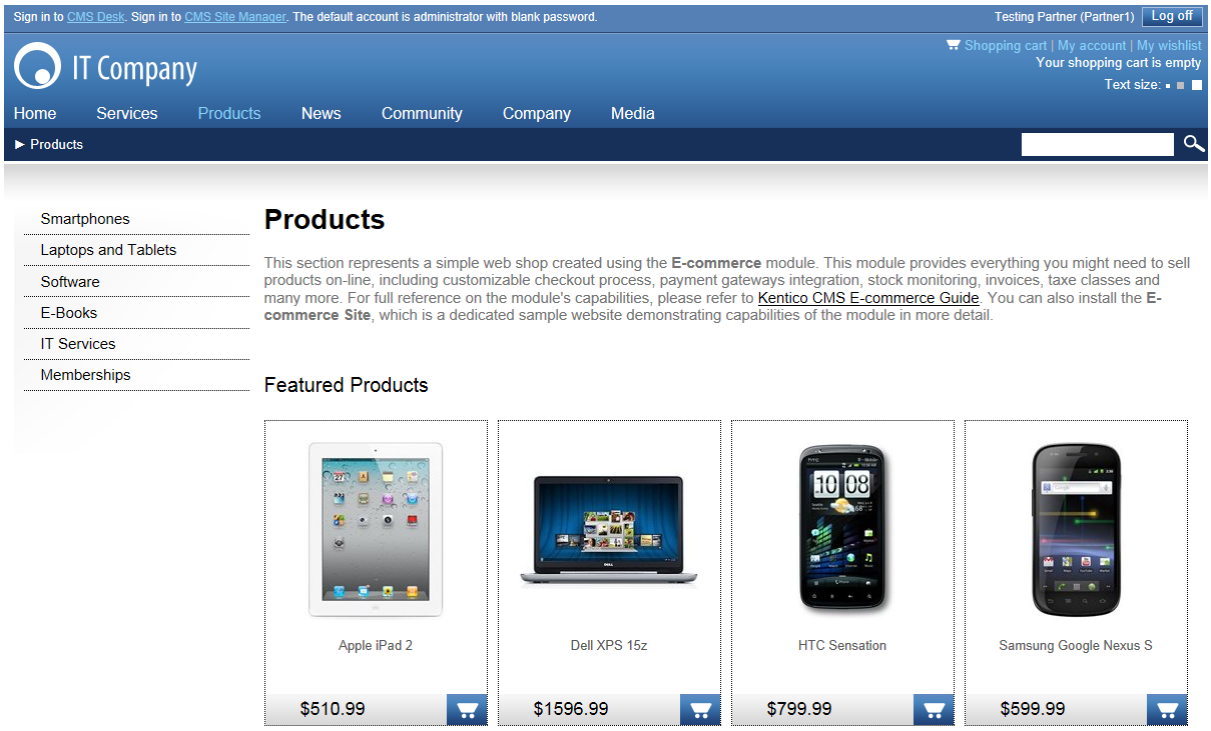
Choose the root in the content tree and click the **Design** tab. Configure ( ) the **Top list menu** (CSS list menu) web part and set its **Check permissions** property in the **System settings** category to true ( ). Click **OK**. Repeat the same for the **CSS list menu** and **Featured products** 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.

7.11.13 Third-party authentication services

7.11.13.1 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 some 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
- LinkedIn
- OpenID
- Facebook Connect

Click the links above to learn more about how to set up authentication using the particular provider.

7.11.13.2 Windows Live ID

7.11.13.2.1 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.

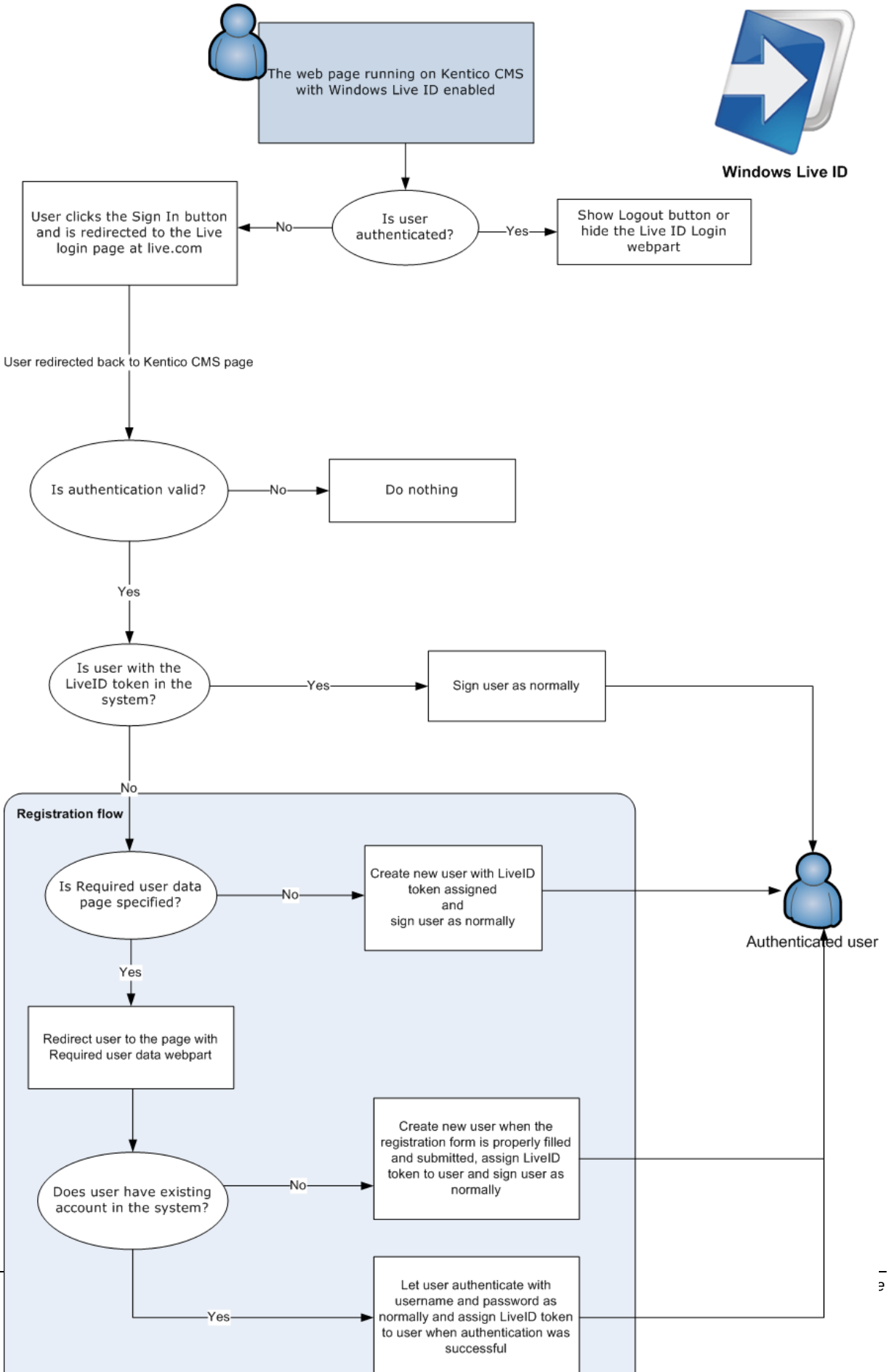
How to start using it

For this to work on your site, you have to do the following things:

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.

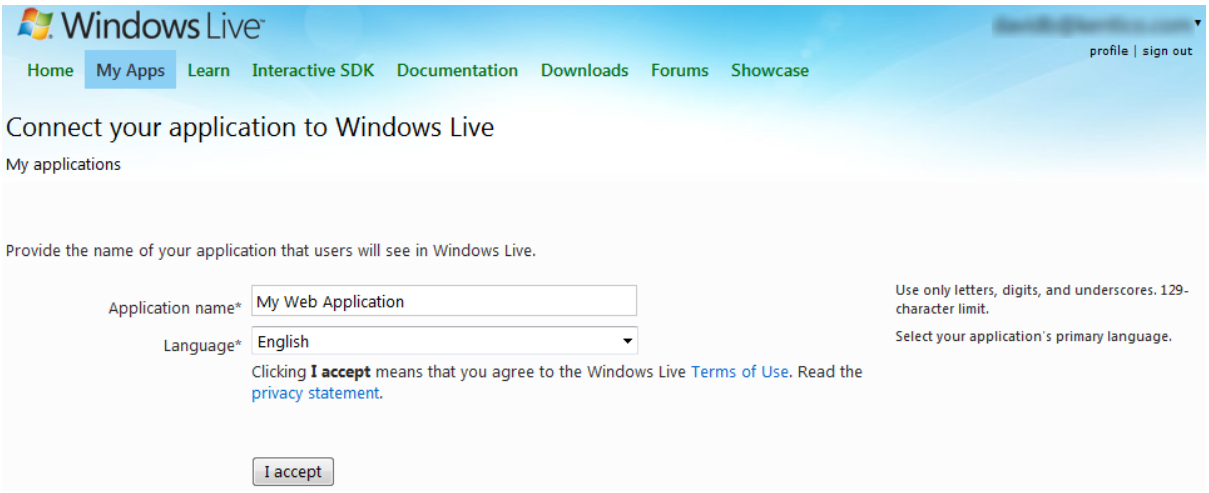
This issue can be avoided by creating a **Required user data page** (described later in this chapter) where users must enter an e-mail address for their account. When this is done, users will receive a notification e-mail about the status of their account.

7.11.13.2.2 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/.

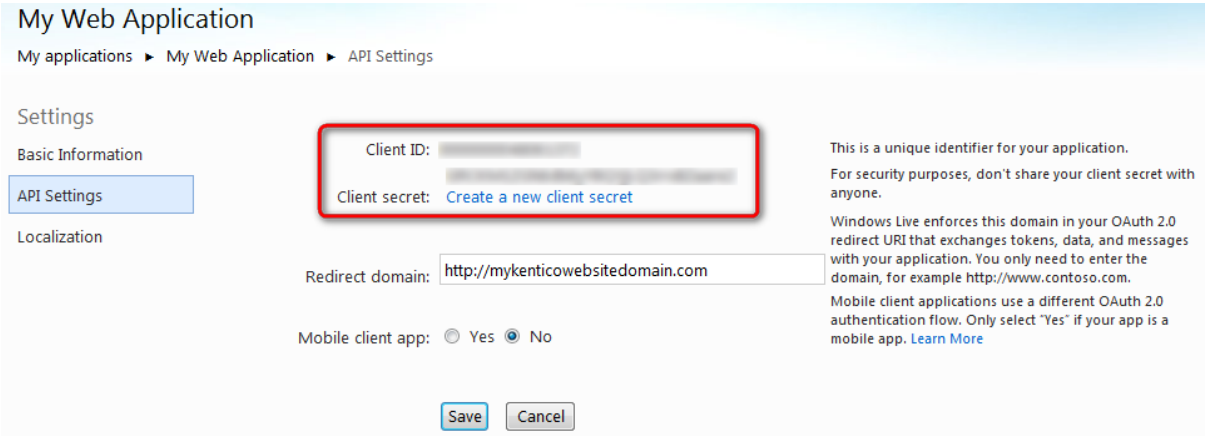
1. Once on the page, sign in with your Windows Live ID (you may need to create an account if you don't already have one) and you will be redirected to the application registration page.

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.



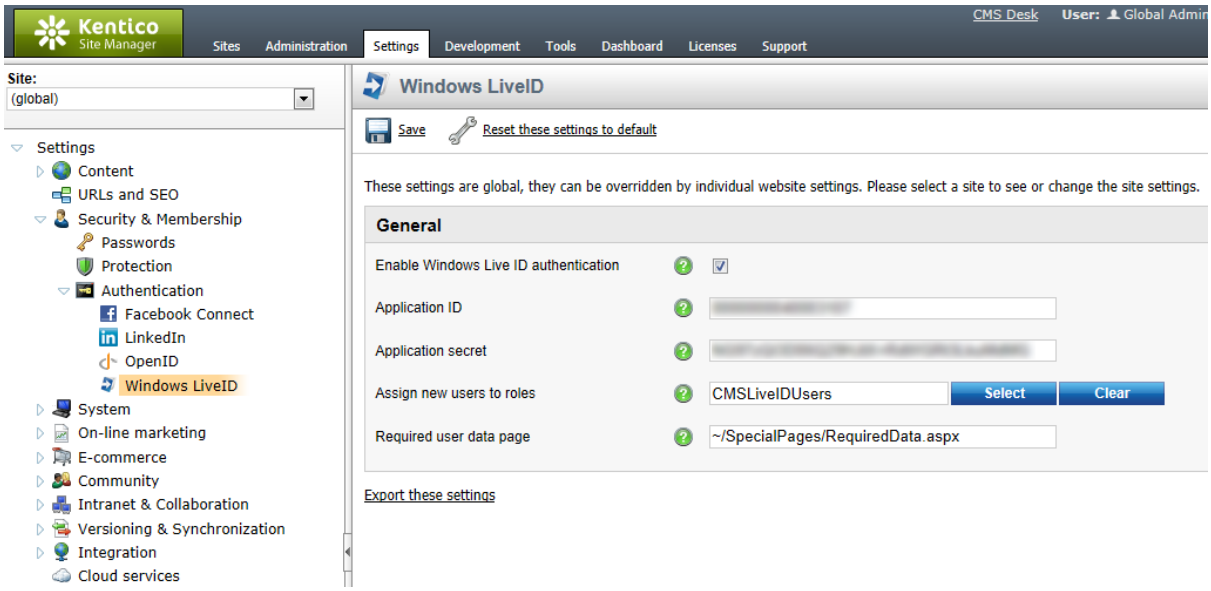
Click **Save** when you are done. Note the **Client ID** and **Client secret** values, since they will be required later when filling in the Live ID authentication settings in Kentico CMS.

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.

7.11.13.2.3 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

Hotfix required

The functionality described below is only available if **hotfix 6.0.8** or newer is applied to your installation of Kentico CMS.

You may download the appropriate hotfix package from http://devnet.kentico.com/Bugtracker/Hotfixes.aspx.

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.

7.11.13.2.4 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.

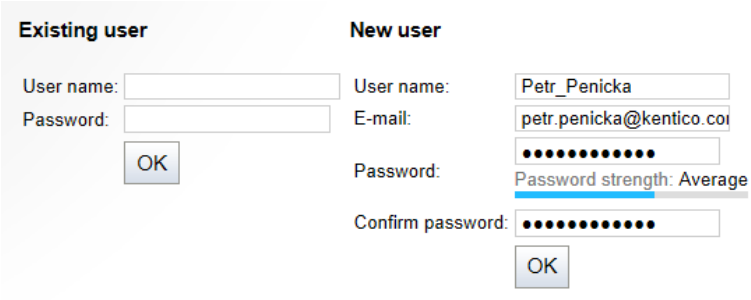
This can be achieved using the *Live ID required data* web part. The web part must be placed on the page specified by the **Required user data page** in **Site Manager -> Settings -> Security & Membership -> Authentication -> Windows LiveID**.

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 **Documentation** ( ) 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.

7.11.13.3 LinkedIn

7.11.13.3.1 Overview

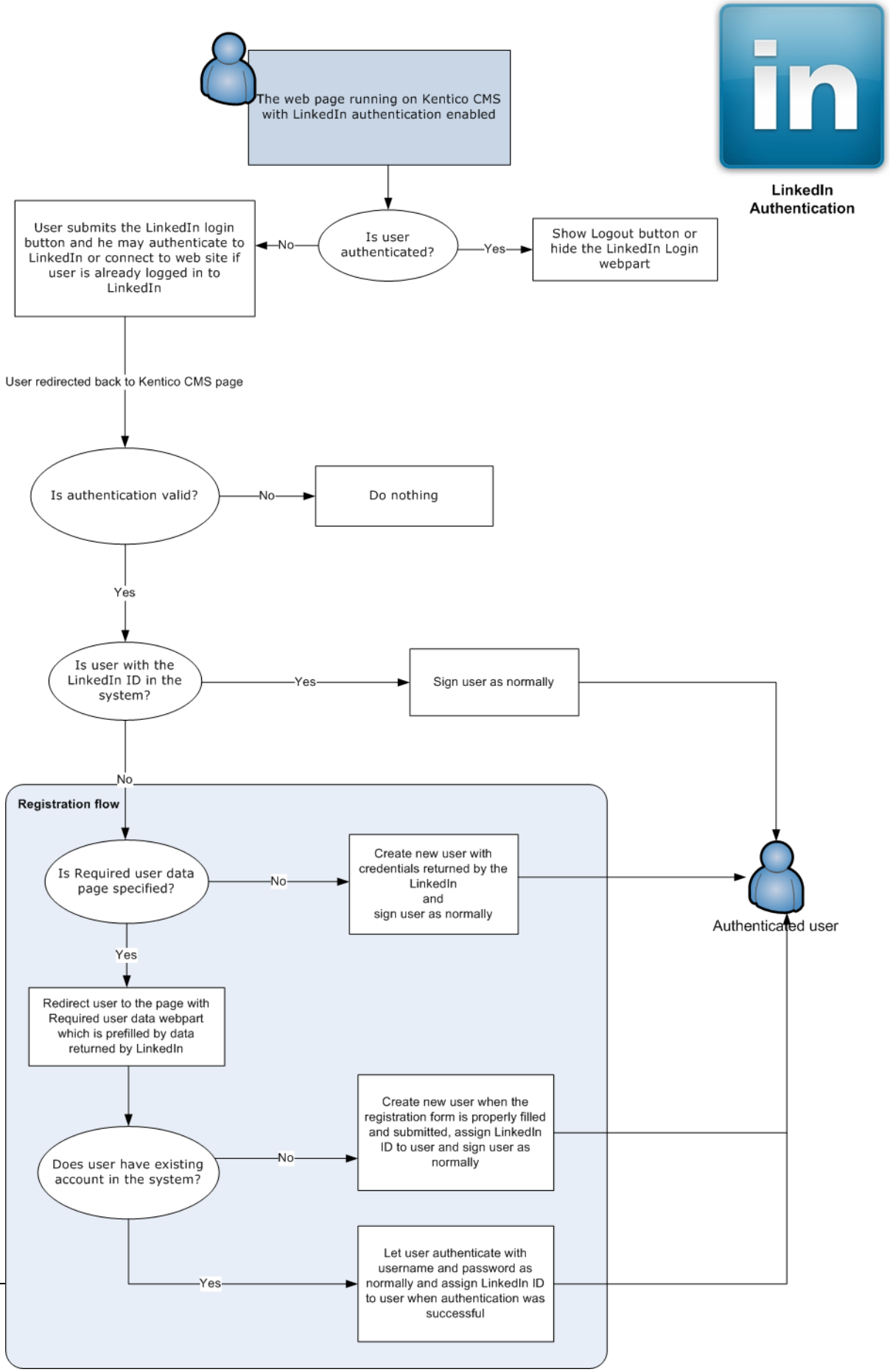
LinkedIn is a business-related social networking website. By integrating LinkedIn authentication into your website, you can let users log on to your website using their LinkedIn user name and password.

How to start using it

1. Register your application at https://www.linkedin.com/secure/developer - learn here.
2. Set up Kentico CMS for LinkedIn authentication - learn here.
3. Use one of the LinkedIn web parts on any page of your site - learn here.

How it works

The following diagram shows how the process of LinkeIn authentication works:

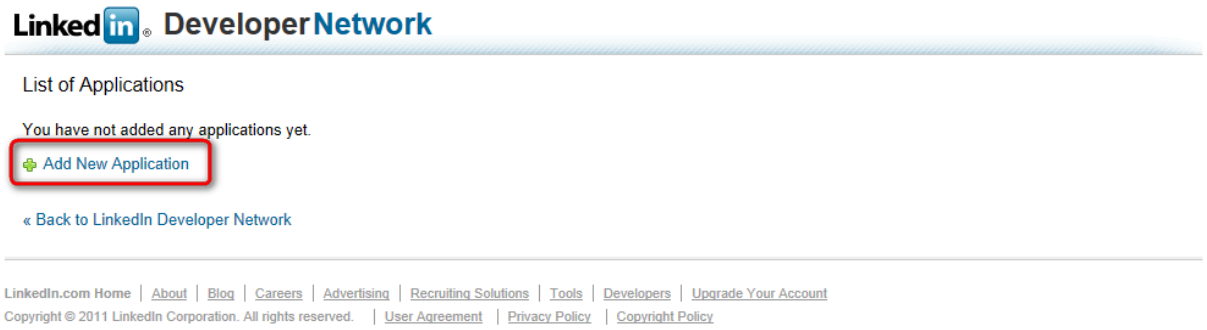

LinkedIn Authentication
The web page running on Kentico CMS with LinkedIn authentication enabled
Is user authenticated?
No
Yes
User submits the LinkedIn login button and he may authenticate to LinkedIn or connect to web site if user is already logged in to LinkedIn
Show Logout button or hide the LinkedIn 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

7.11.13.3.2 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 at 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.

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:

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.

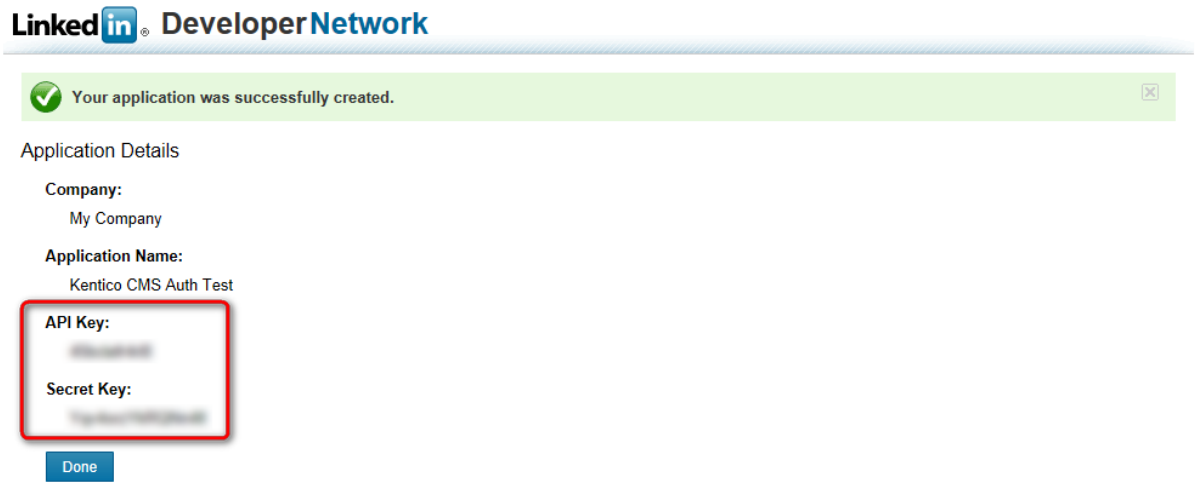
* Agreement Language: English

Select the display language of the user agreement screen. Browser Locale Setting is recommended.

Term of Service

*Agree: ☑

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.

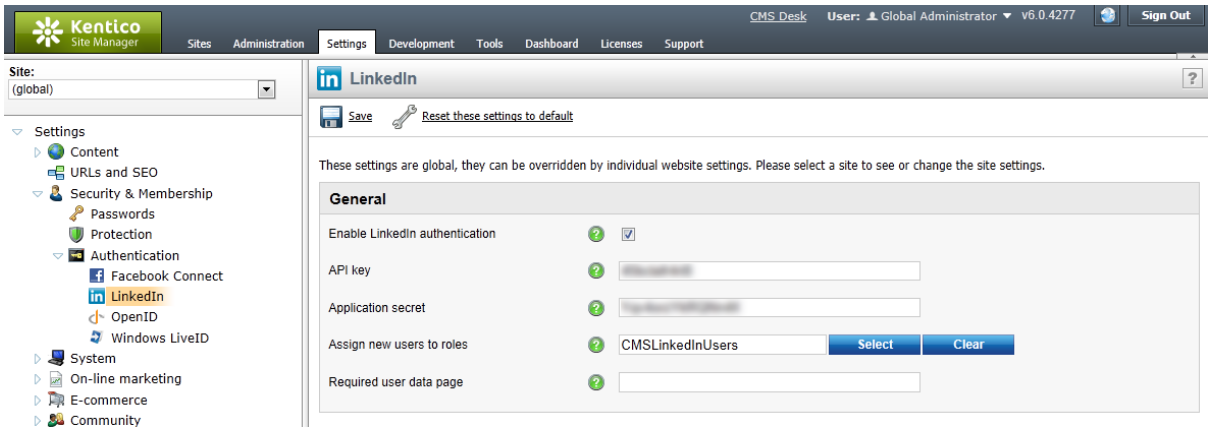


7.11.13.3.3 Settings

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 -> LinkedIn**. 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 LinkedIn authentication** - indicates if LinkedIn authentication is enabled.
- **API key** - key obtained during registration of your application at LinkedIn.
- **Application secret -** key obtained during registration of your application at LinkedIn.
- **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.

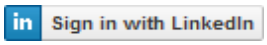


7.11.13.3.4 Available web parts

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

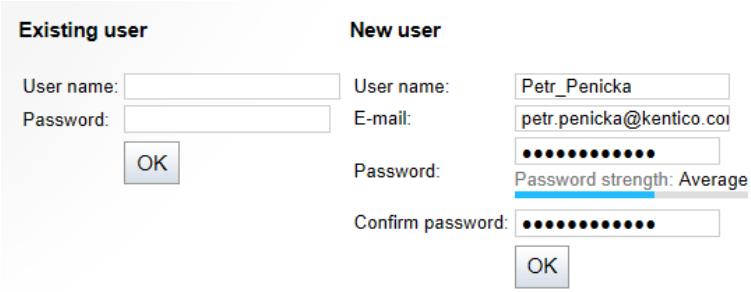
In some cases, you may want new users to provide some extra details before their account is created. For example, if your site is configured to require registration approval or double opt-in, all users need a valid e-mail address to help activate their account.

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 -> Security & Membership -> Authentication -> 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 acounts.
- **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 **Documentation** ( ) 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.

7.11.13.4 OpenID

7.11.13.4.1 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 it

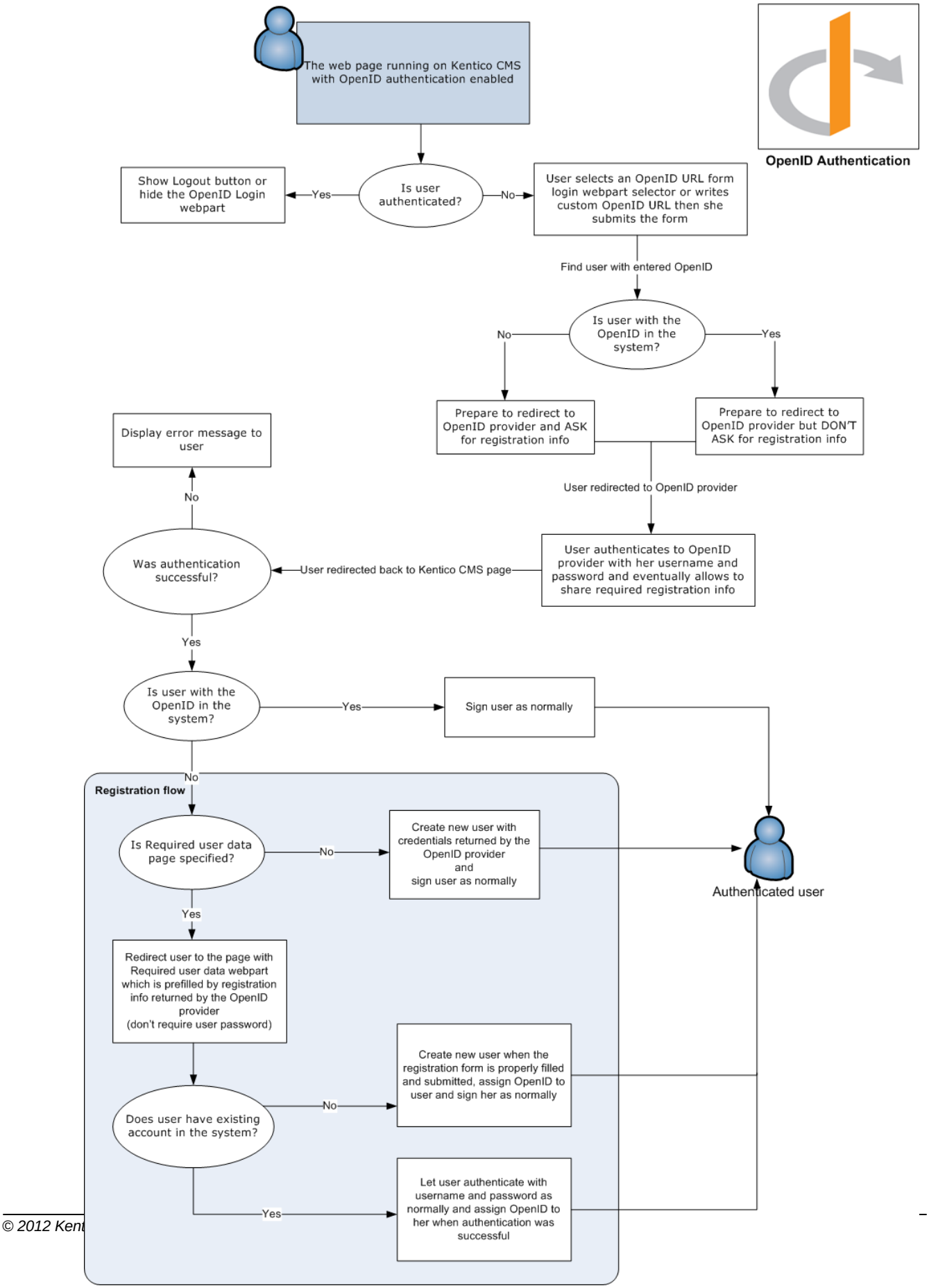
OpenID is the easiest to use of the three supported third-party authentication services. For it to work on your site, you don't need to register your site, all you need to do is take the following two steps:

1. Set up Kentico CMS for OpenID authentication - learn here

2. Use one of the OpenID web parts on any page of your site - info here

How it works

The following diagram shows how the process of OpenID login works:

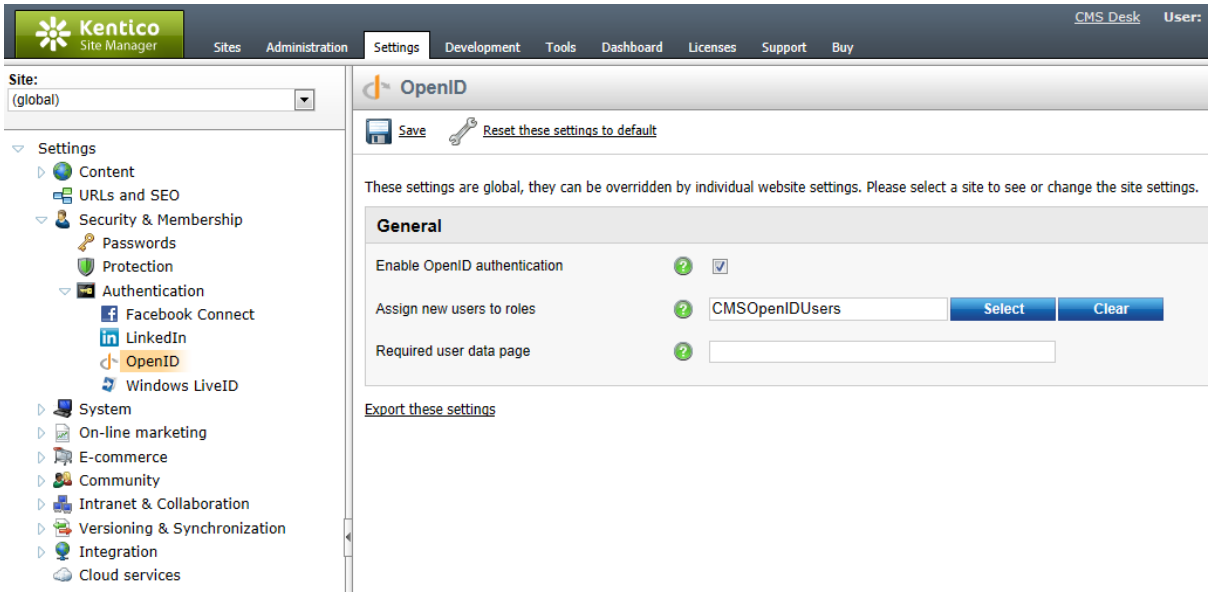

The web page running on Kentico CMS with OpenID authentication enabled
OpenID Authentication
Is user authenticated?
Yes
Show Logout button or hide the OpenID Login webpart
No
User selects an OpenID URL form login webpart selector or writes custom OpenID URL then she submits the form
Find user with entered 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

7.11.13.4.2 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



7.11.13.4.3 Available web parts

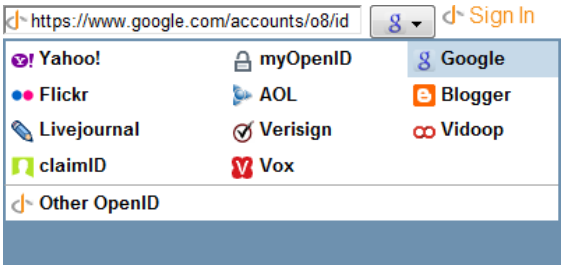
After configuring the settings, you can use the following two web parts on your website to allow OpenID authentication. The web parts are located under the **Membership** category in the web part catalog.

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

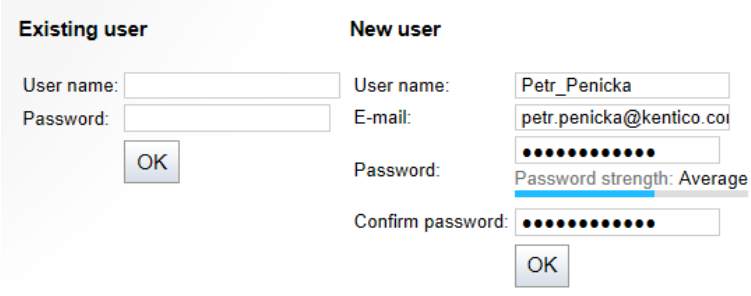
In some cases, you may want new users to provide some extra details before their account is created. For example, if your site is configured to require registration approval or double opt-in, all users need a valid e-mail address to help activate their account.

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 **Documentation** ( ) 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.

7.11.13.5 Facebook Connect

7.11.13.5.1 Overview

Facebook is one of the most popular social networking sites in the world. With its continuously growing number of registered users, it is quite likely that visitors who come to your site will already have a Facebook account. In this case, it is much more convenient for them to use their Facebook logon details than to go through yet another registration on your site. That's why you should consider using Facebook Connect authentication.

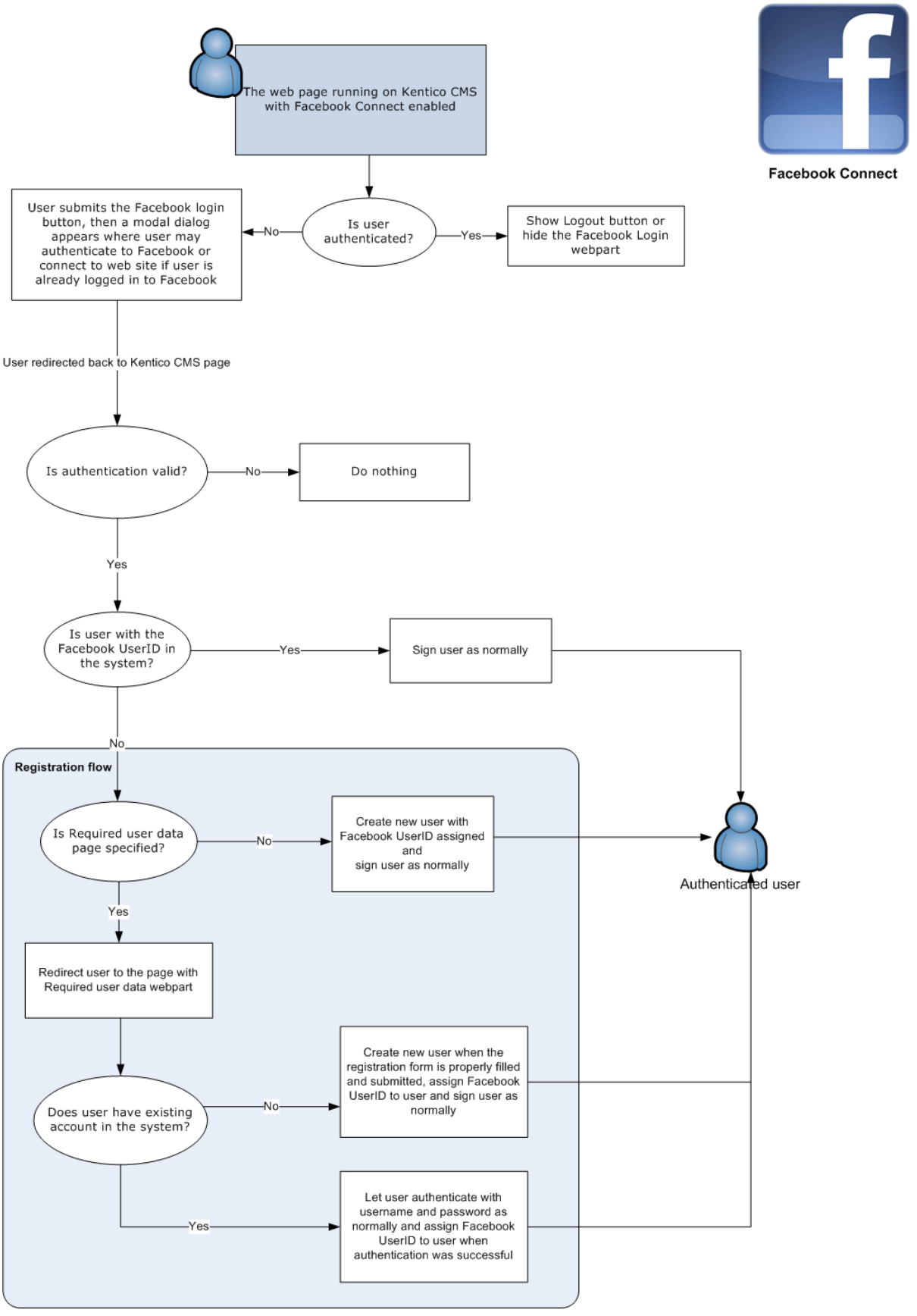
How to start using it

For Facebook Connect to work on your site, you need to register the site, all you need to do is take the following two steps:

1. Register your website at http://www.facebook.com/developers/createapp.php - for more info ready here
2. Set up Kentico CMS for Facebook Connect authentication - learn here
3. Use one of the Facebook Connect web parts on any page of your site - info here

How it works

The following diagram shows how the process of Facebook Connect login works:



7.11.13.5.2 Registering your application

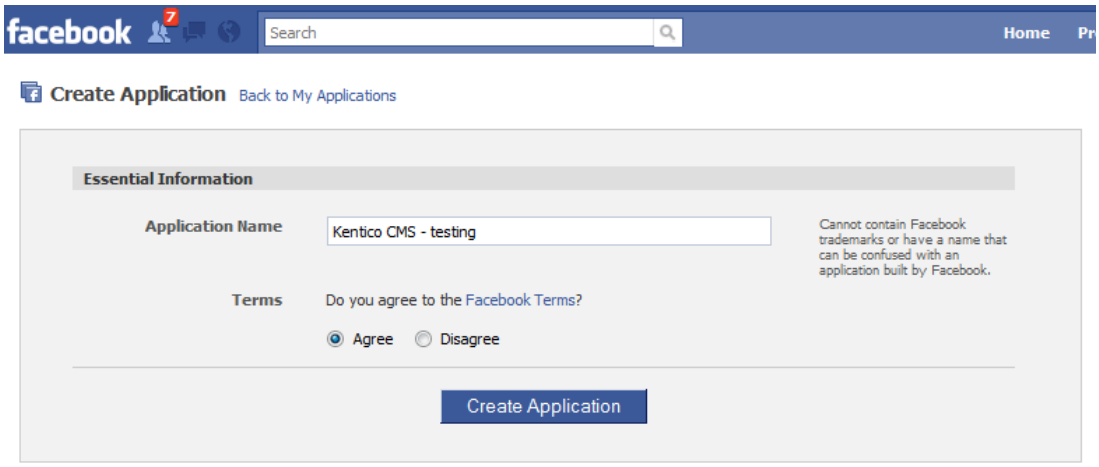
To use Facebook Conntect authentication on your website, you need to register the site at http://www.facebook.com/developers/createapp.php. You need to have a Facebook account to get to the page, so if you don't have one yet, please create it here (you would be prompted to do so otherwise).

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.

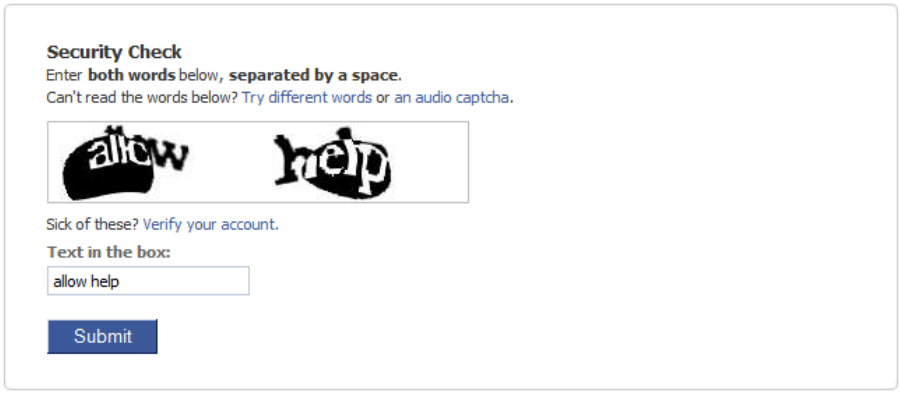
1. With a facebook account created and logged in, visit http://www.facebook.com/developers/createapp.php. Enter the following details into the form:

- **Application Name**: enter the name of your website, e.g. Kentico CMS - testing

Read and **Agree** to the *Facebook Terms* and click **Create Application**.



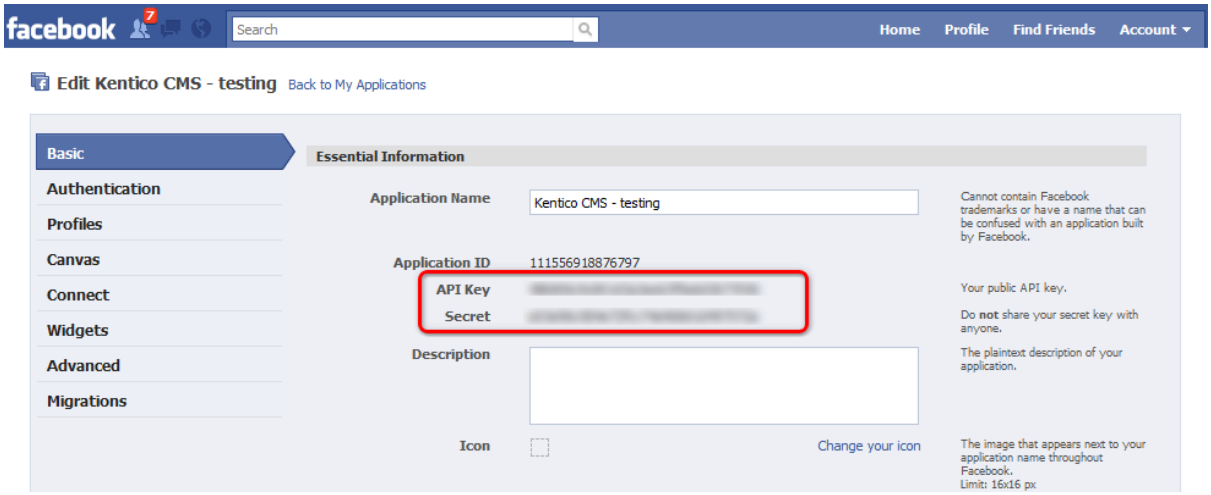
2. Retype the CAPTCHA on and click **Submit**.



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, you will need to enter them in **Site Manager -> Settings -> Security & Membership -> Authentication -> Facebook Connect**:

- **API Key** - this is what you will enter into the **API Key** field
- **Secret** - this is what you will enter into the **Application secret** field

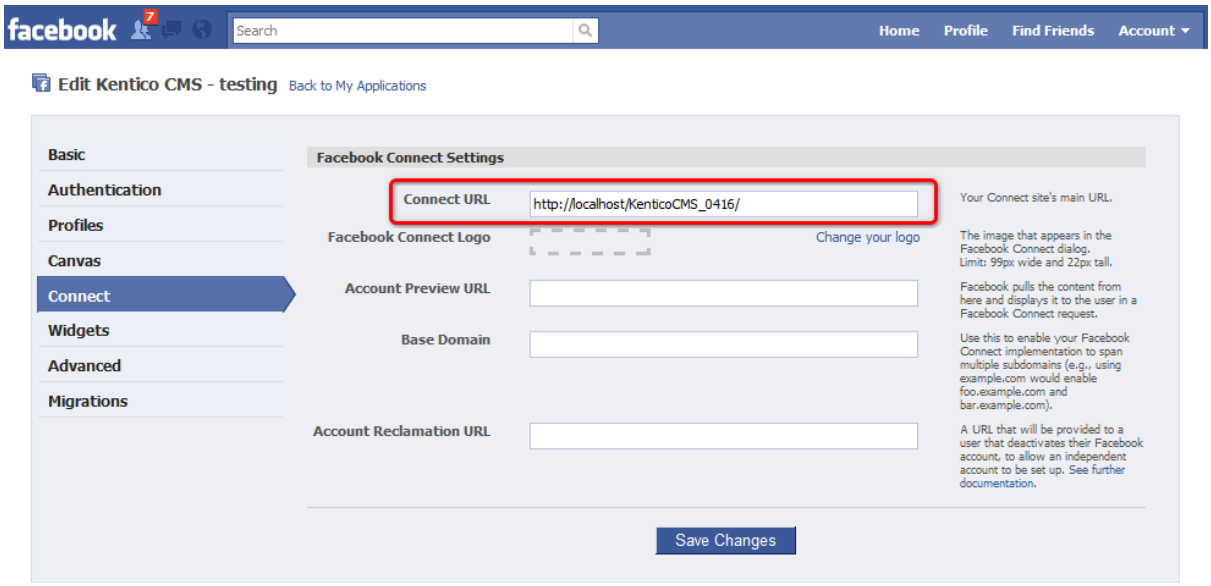
Note them down and set them accordingly in your Kentico CMS instance as described in the Settings chapter.



4. Now switch to the **Connect** tab. Here, you need to enter the URL of your website into the **Connect URL** field. Click **Save Changes**.

Please note!

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 really running.

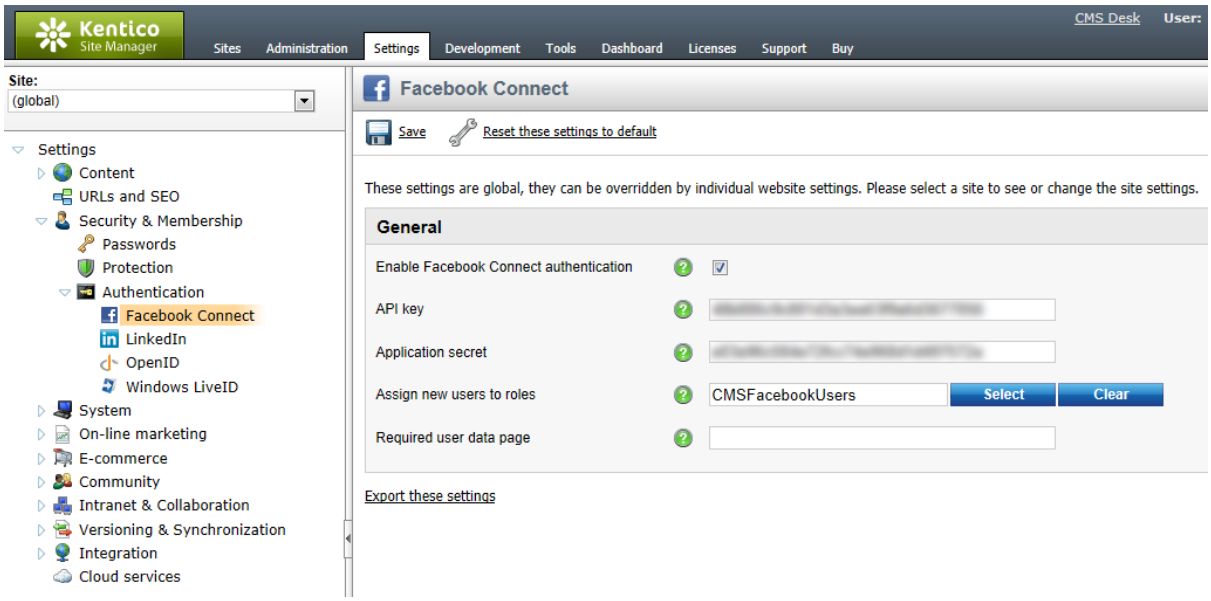


Your web application is now registered at Facebook. With proper settings and web parts, users should now be able to authenticate to your site using their Facebook logon details.

7.11.13.5.3 Settings

Facebook Connect authentication settings are located in **Site Manager -> Settings -> Security & Membership -> Authentication -> Facebook Connect**. 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 Facebook Connect** - indicates if Facebook Connect authentication is enabled
- **API key** - key obtained during registration of your application on Facebook (as described here)
- **Application secret** - key obtained during registration of your application on Facebook
- **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



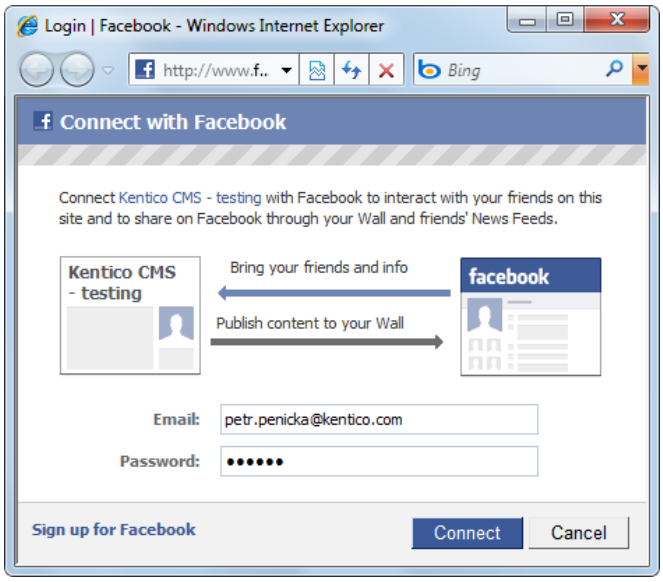
7.11.13.5.4 Available web parts

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.

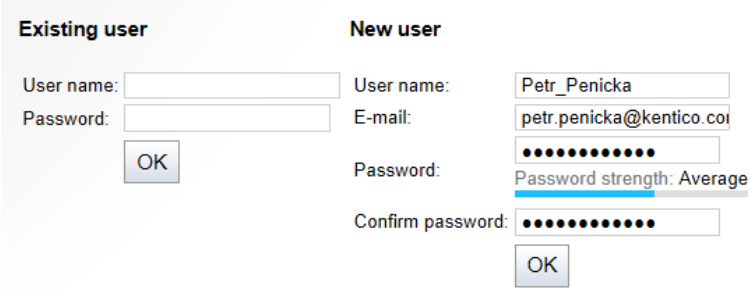
This can be achieved using the *Facebook Connect required data* web part. The web part must be placed on the page specified by the **Required user data page** value in **Site Manager -> Settings -> Security & Membership -> Authentication -> Facebook Connect**.

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 **Documentation** ( ) 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.

7.11.13.6 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

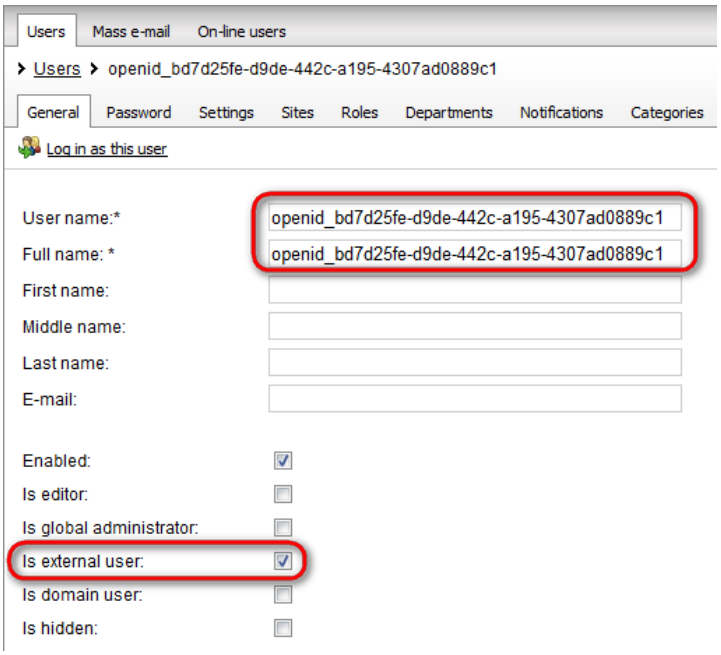
- **Live ID**: *liveid_<liveidtoken>*
- **OpenID**: *openid_<openid>*
- **Facebook Connect**: *facebookid_<facebookuserid>*

Full name

- **Live ID**: *LiveID - <liveidtoken>*
- **OpenID**: *OpenID - <openid>*
- **Facebook Connect**: *Facebook ID - <facebookuserid>*

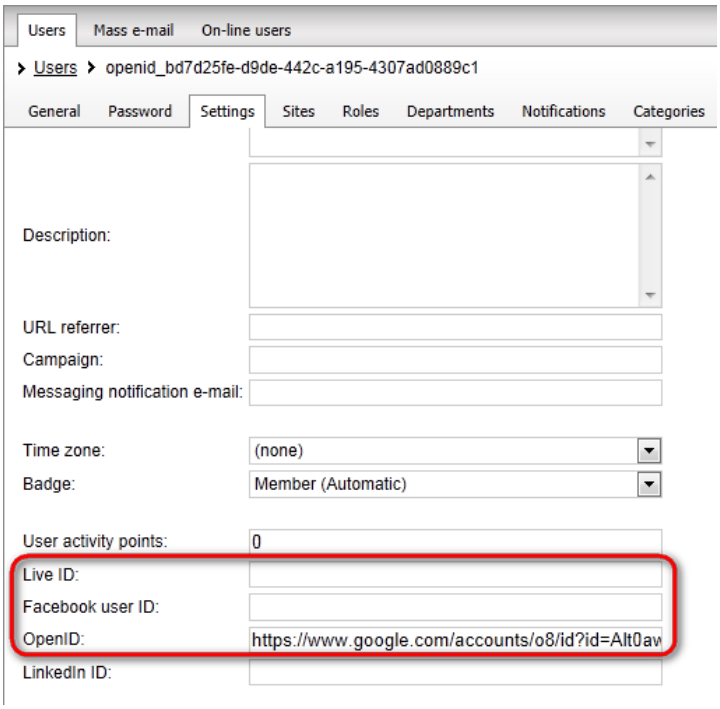
These values can be changed manually without any effects on the functionality.

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.



7.11.14 Membership internals and API

7.11.14.1 Overview

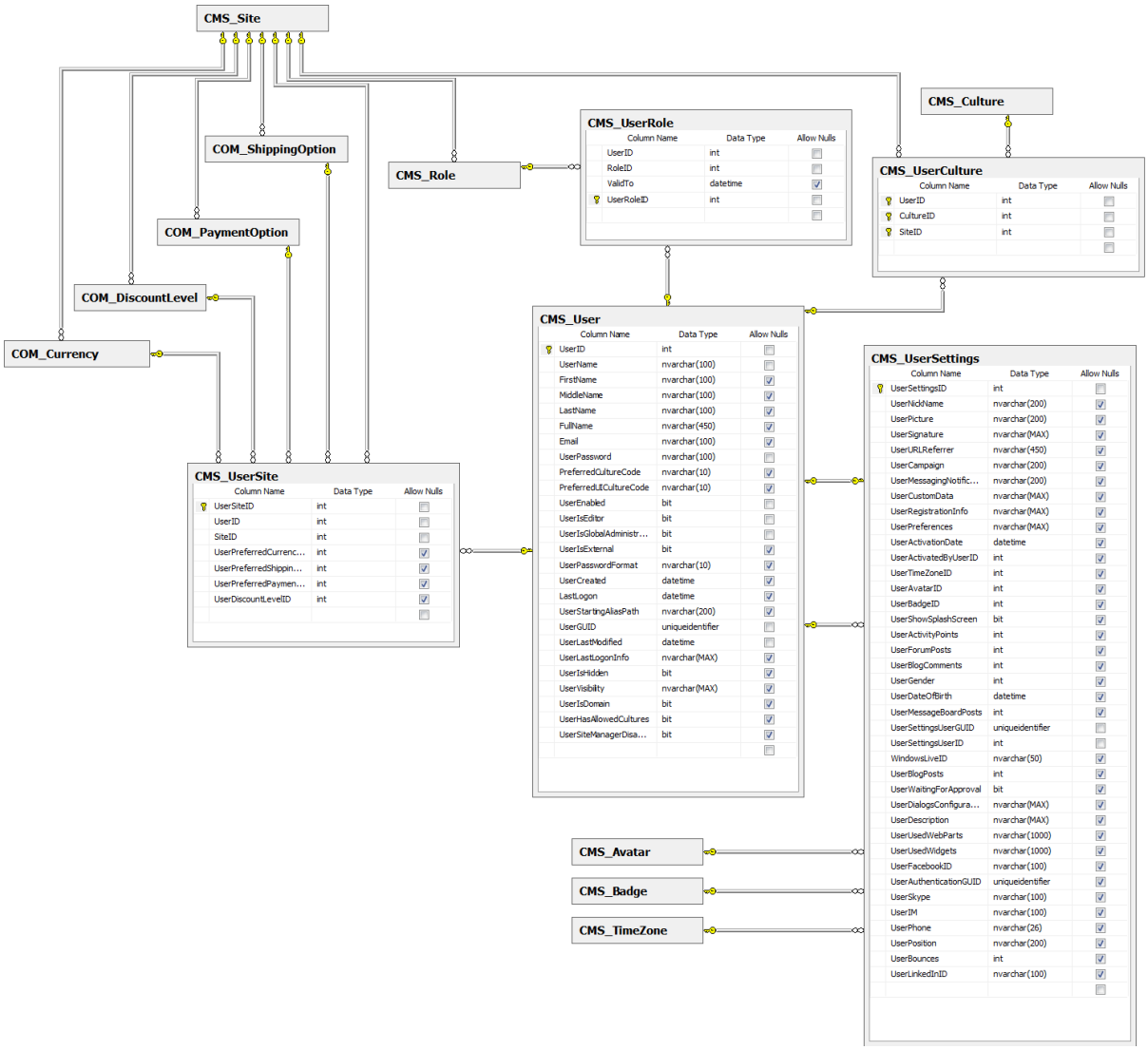
In this chapter, you will learn which database tables and API classes are used for the management of website membership (users and roles). You will also see the most common API examples.

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.

7.11.14.2 Database tables

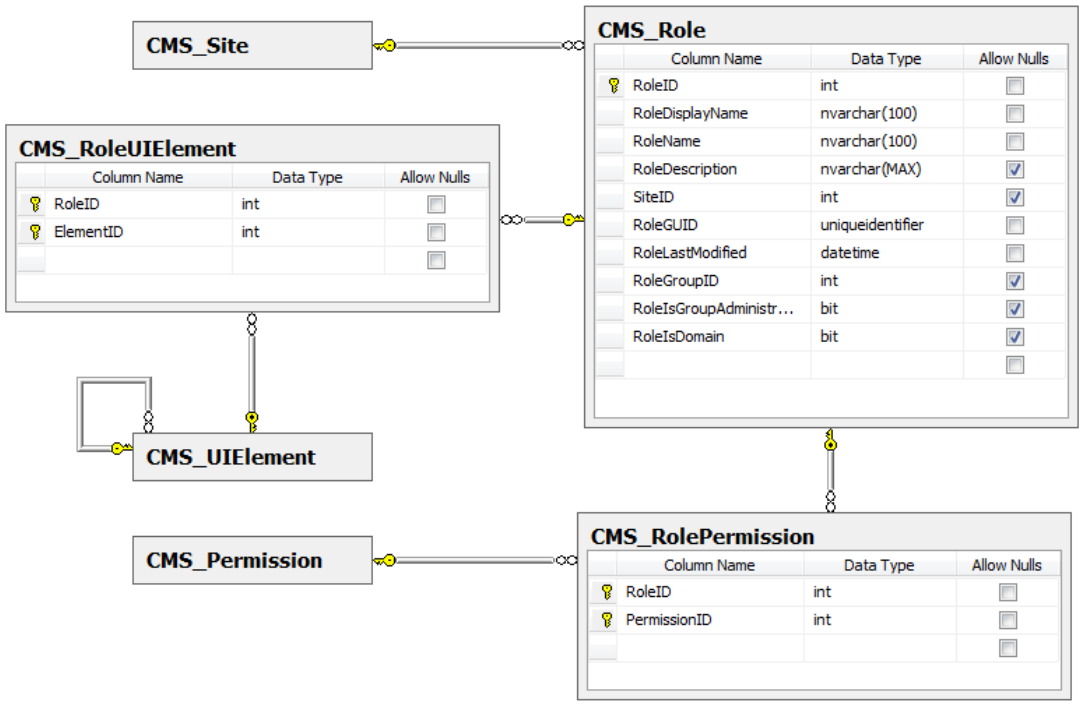
The following database tables are used to store information about users:

- **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.



7.11.14.3 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 usd 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.

7.11.14.4 API examples

7.11.14.4.1 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;
```

7.11.14.4.2 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 (UserInfoProvider.AuthenticateUser("MyNewUser", "", CMSContext.
```

```
CurrentSiteName) != null)
        {
            return true;
        }
    }

    return false;
}
```

7.11.14.4.3 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;
}
```

7.11.14.4.4 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);
}
```

7.11.14.4.5 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;
}
```

7.12 Microsoft Silverlight integration

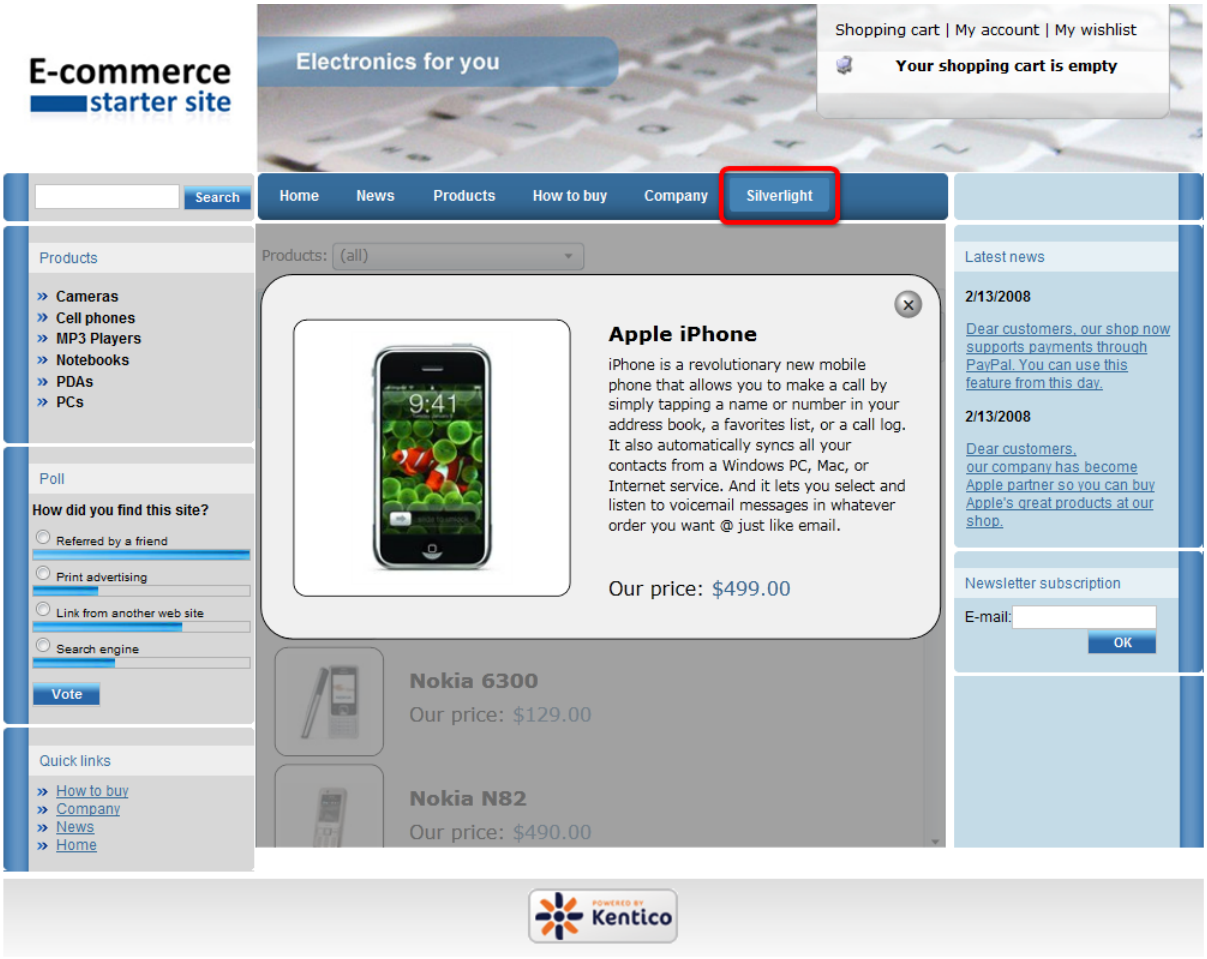
7.12.1 Overview

Kentico CMS comes with native support of **Microsoft Silverlight**. Microsoft Silverlight is a new cross-browser, cross-platform technology for building and delivering the next generation of 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, an install banner leading to the download link will be offered automatically.

You can find an example of a Silverlight application on the **E-commerce Site** sample website, in the **Silverlight** section.



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 there valuable tutorials 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.

7.12.2 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 **E-commerce Site** sample website, in the **Silverlight** section.

The following steps need to be taken to add a Silverlight application 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 and click the **Add web part** () icon at the top right-hand corner of the web part zone where you want to place the application.

3. Choose the **Silverlight\Silverlight application** web part and click **OK**.

4. In the web part properties window which pops-up, you can set the following web-part-specific 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

Click **OK**, switch to the **Live site** and enjoy your Silverlight application running on your Kentico CMS website.

7.12.3 IIS configuration

Silverlight introduces two new file extensions:

- **.xaml** for XAML files
- **.xap** for the zip-based binary packaging format

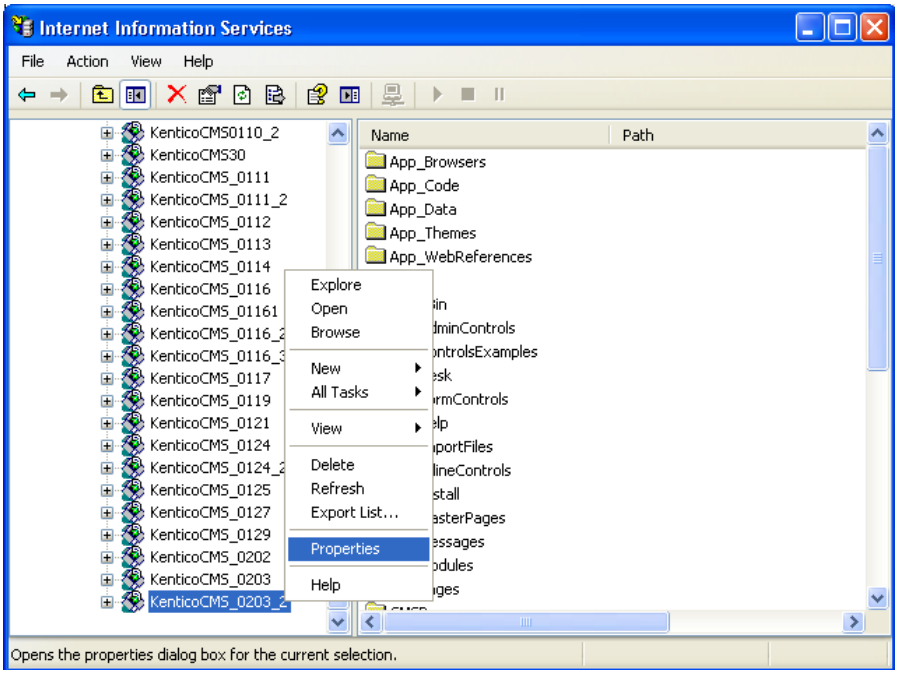
For your application to work correctly, you need to add the MIME types for these file extensions to your web server so that it recognizes Silverlight content appropriately:

- **Extension**: .xaml
- **MIME type**: application/xaml+xml

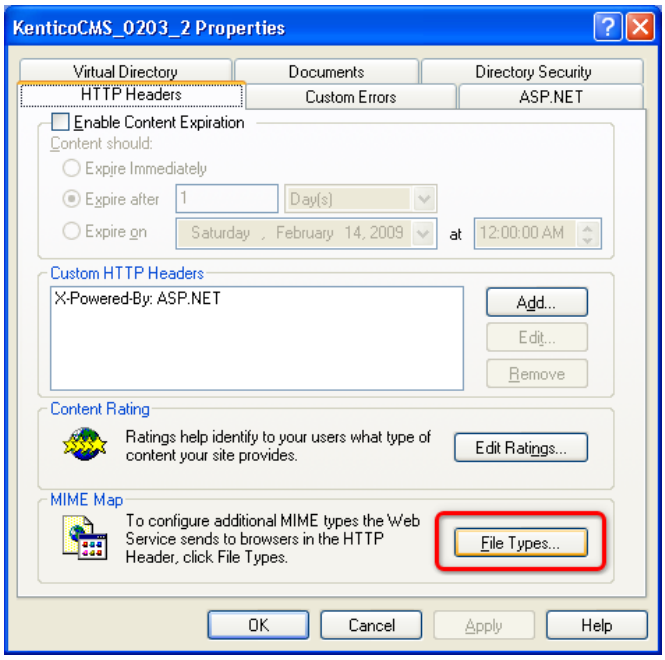
- **Extension**: .xap
- **MIME type**: application/x-silverlight-app

Here is a step-by-step guide on how to do it on IIS 5.1:

1. Open IIS, right-click your website and choose **Properties**.



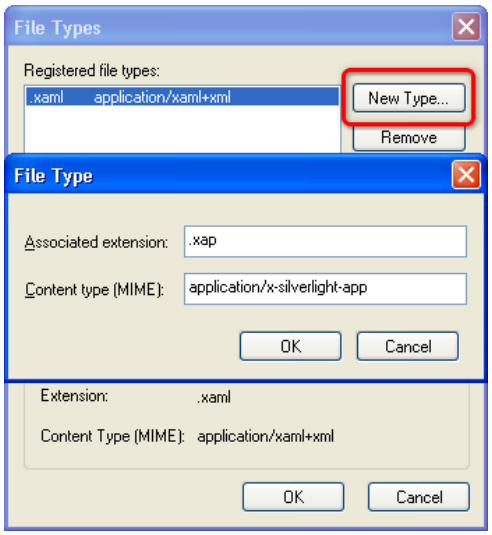
2. Switch to the **HTTP Headers** tab and choose **File Types** in the **MIME Map** section.



3. Using the **New Type** button, add the following MIME types:

- **Associated extension**: .xaml
- **Content type (MIME)**: application/xaml+xml

- **Associated extension**: .xap
- **Content type (MIME)**: application/x-silverlight-app



7.13 Object versioning

7.13.1 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.

7.13.2 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.)

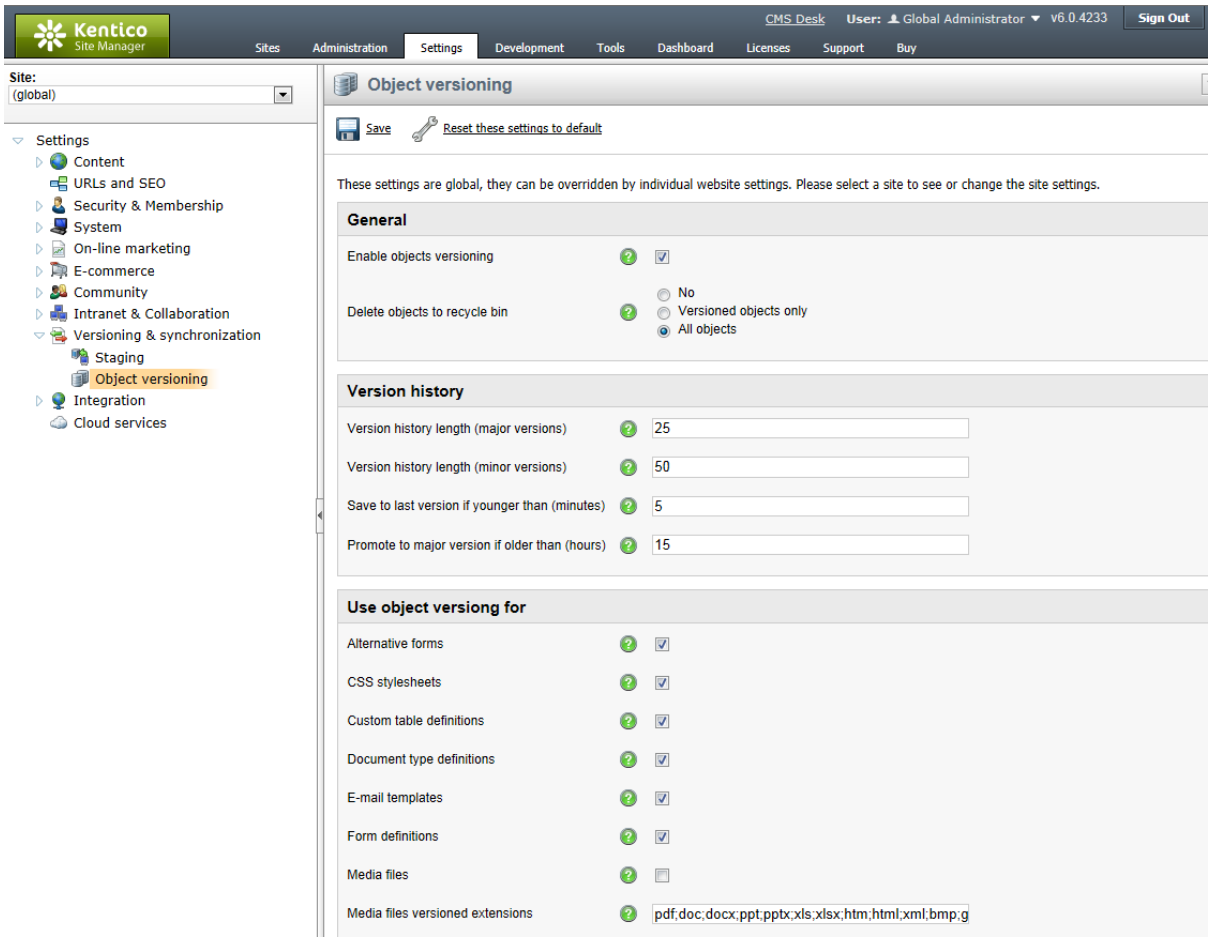
Custom table definitions	Site Manager -> Development -> Custom tables -> edit ()
Document type definitions	Site Manager -> Development -> Document types -> edit ()
E-mail templates	Site Manager -> Administration -> E-mail templates -> edit ()
Form definitions	CMS Desk -> Tools -> Forms -> edit ()
Media 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 not versioned. ** Only custom queries and web part layouts are versioned — system queries and default web part layouts are not versioned.	

7.13.3 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.
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.



7.13.4 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:

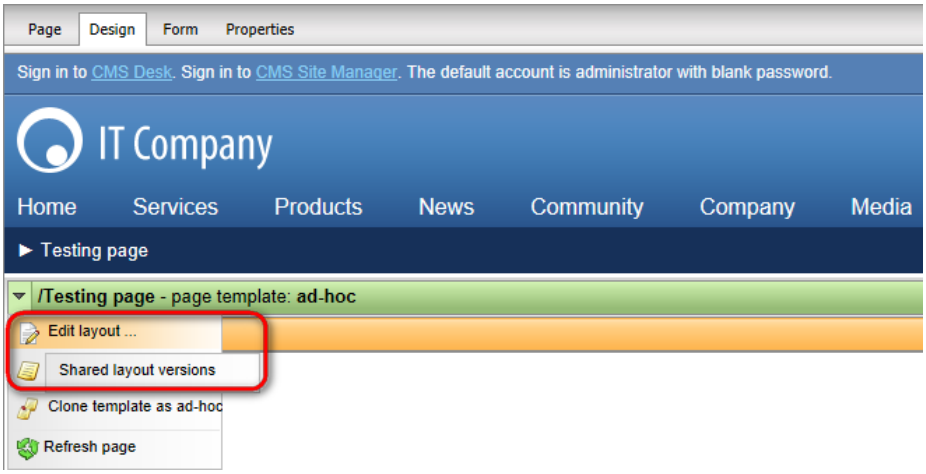
- **View version** - opens a new window where field data of the version can be inspected and compared side-by-side with data of another version. See the Version data view and comparison section below for more details.
- **Rollback version** - performs rollback of the object to the particular version. 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:

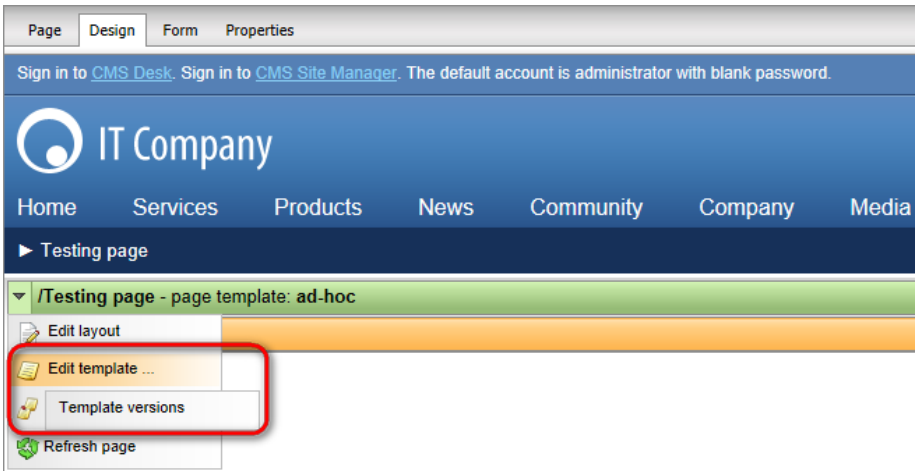
- **Rollback with children** - if the object has some child objects (e.g. alternative forms, transformations, queries, ...), it performs rollback of the object to the particular version, while child object data are rolled back as well. If it does not have any child objects, only the object itself is rolled back.

Management of versions in CMS Desk -> Content -> Edit -> Design

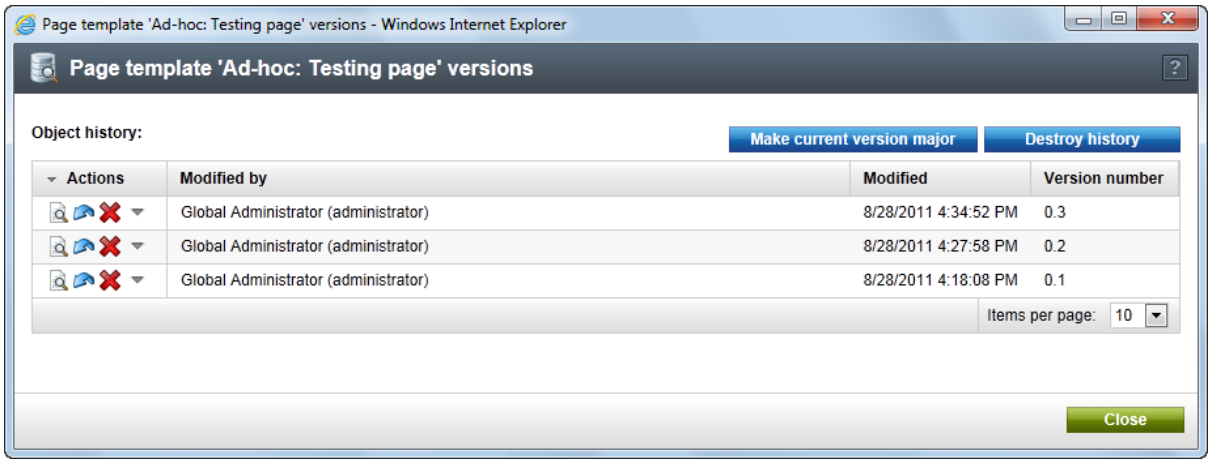
If you expand a page template's drop-down menu and hold your mouse pointer over the **Edit layout** menu item, an additional menu item **Shared version layouts** appears.



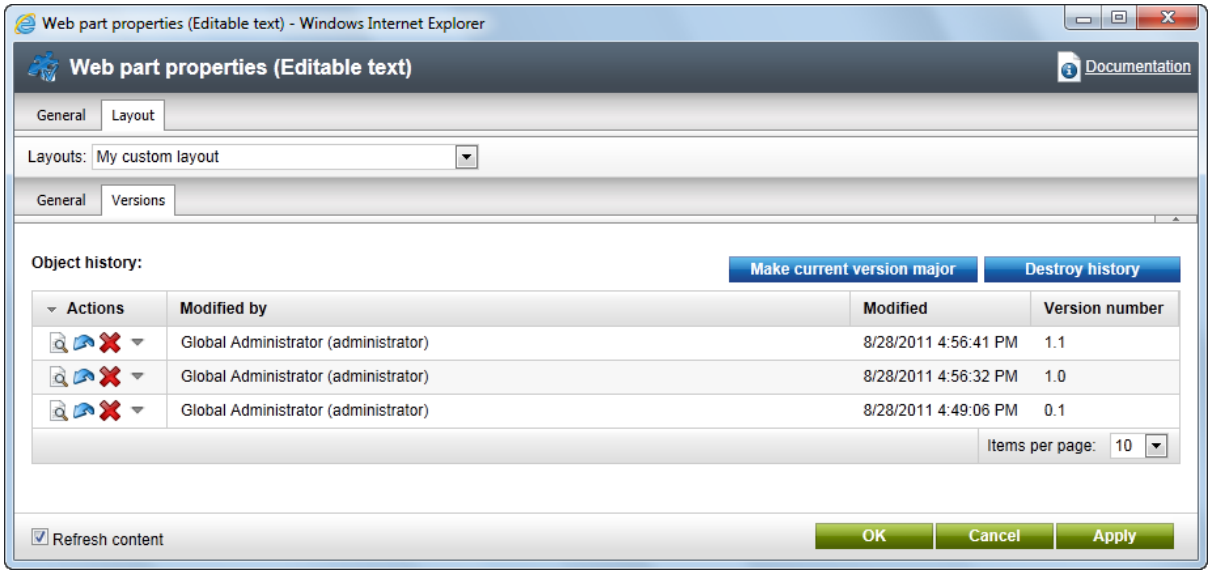
Similarly, if you hold the mouse pointer over the **Edit template** menu item, an additional menu item **Template versions** appears.



After clicking both of the menu items mentioned above, an pop-up window 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:

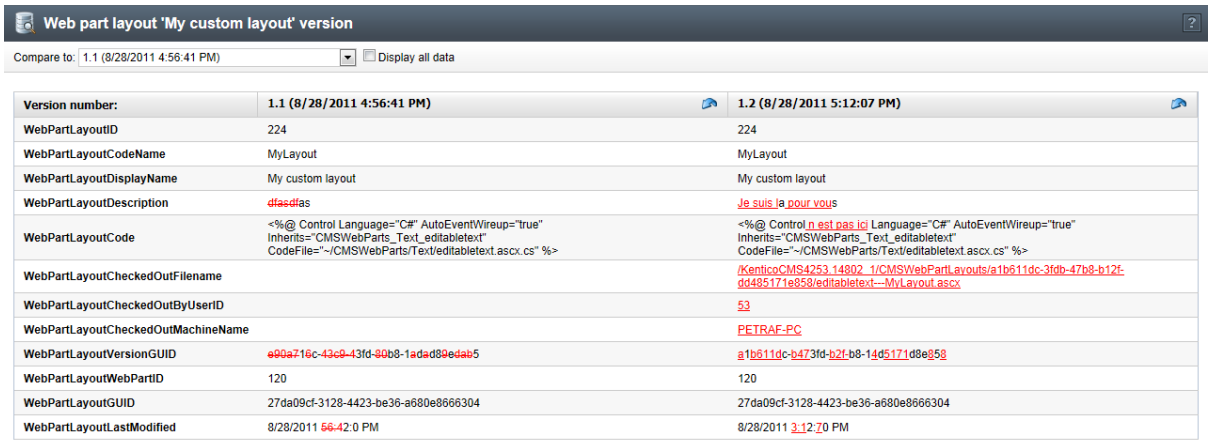
- **Compare to** - by selecting a version in this drop-down list, field data of the selected version get compared side-by-side with field data of the originally viewed version in the table below. Differences between the two versions are highlighted in red font.
- **Display all data** - by enabling this 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	e90a716c-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

7.13.5 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).

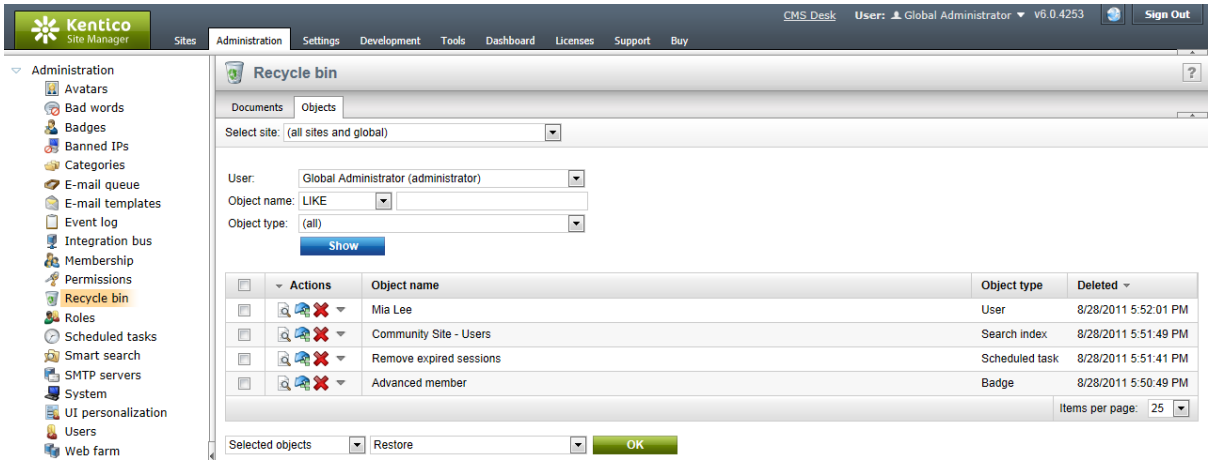
- Site objects (objects whose definition contains the *SiteID* column):

  - **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).

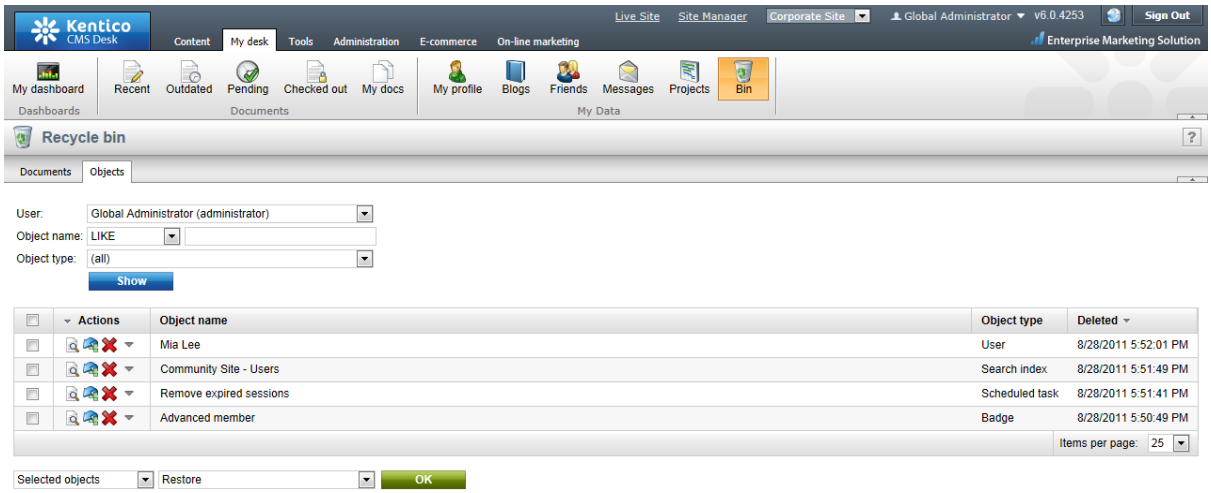
Using the two drop-down lists below the grid, you can perform the **Restore**, **Delete**, **Restore without site bindings** and **Restore to current site** actions to more objects in a single click. Using the first drop-down list, you can choose:

- **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.



7.13.6 Object versioning internals and API

7.13.6.1 Overview

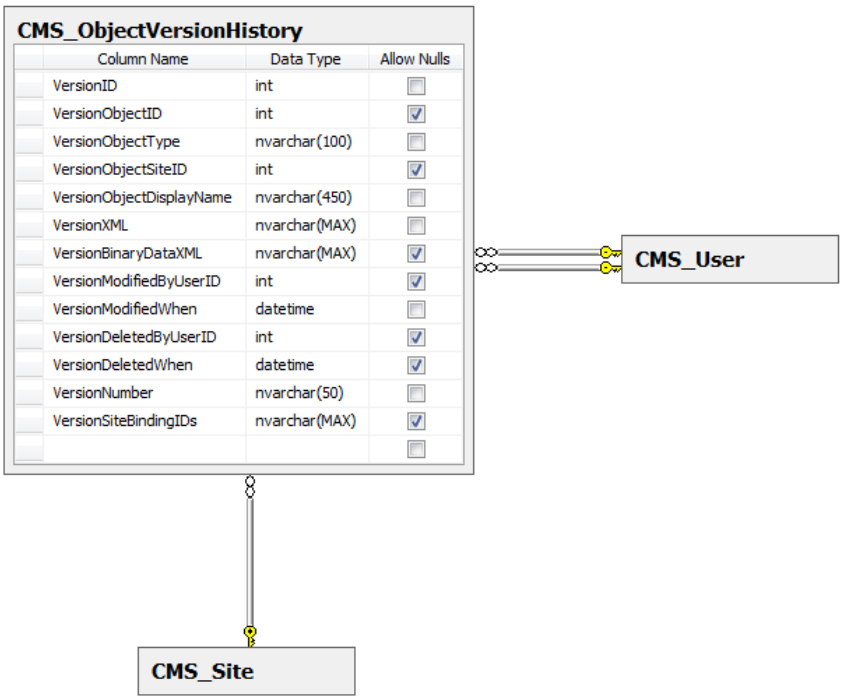
In this chapter, you will learn which database tables and API classes are used to provide the object versioning functionality. You will also see the most common API examples.

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.

7.13.6.2 Database tables

The following database tables are used by object versioning:

- **CMS_ObjectVersionHistory** - contains records representing particular versions of versioned objects.



7.13.6.3 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.

7.13.6.4 API examples

7.13.6.4.1 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;
```

7.13.6.4.2 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, CMSContext.
CurrentSiteName))
    {
        // 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,
CMSContext.CurrentSiteName))
        {
            // 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)
    {
        // Check if object versioning of stylesheet objects is allowed on current
site
        if (ObjectVersionManager.AllowObjectVersioning(stylesheet, CMSContext.
CurrentSiteName))
        {
            // 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, CMSContext.
CurrentSiteName))
        {
            // Ensure version
            ObjectVersionManager.EnsureVersion(stylesheet, false);

            return true;
        }
    }

    return false;
}
```

7.13.6.4.3 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, CMSContext.
CurrentSiteName))
        {
            // 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.

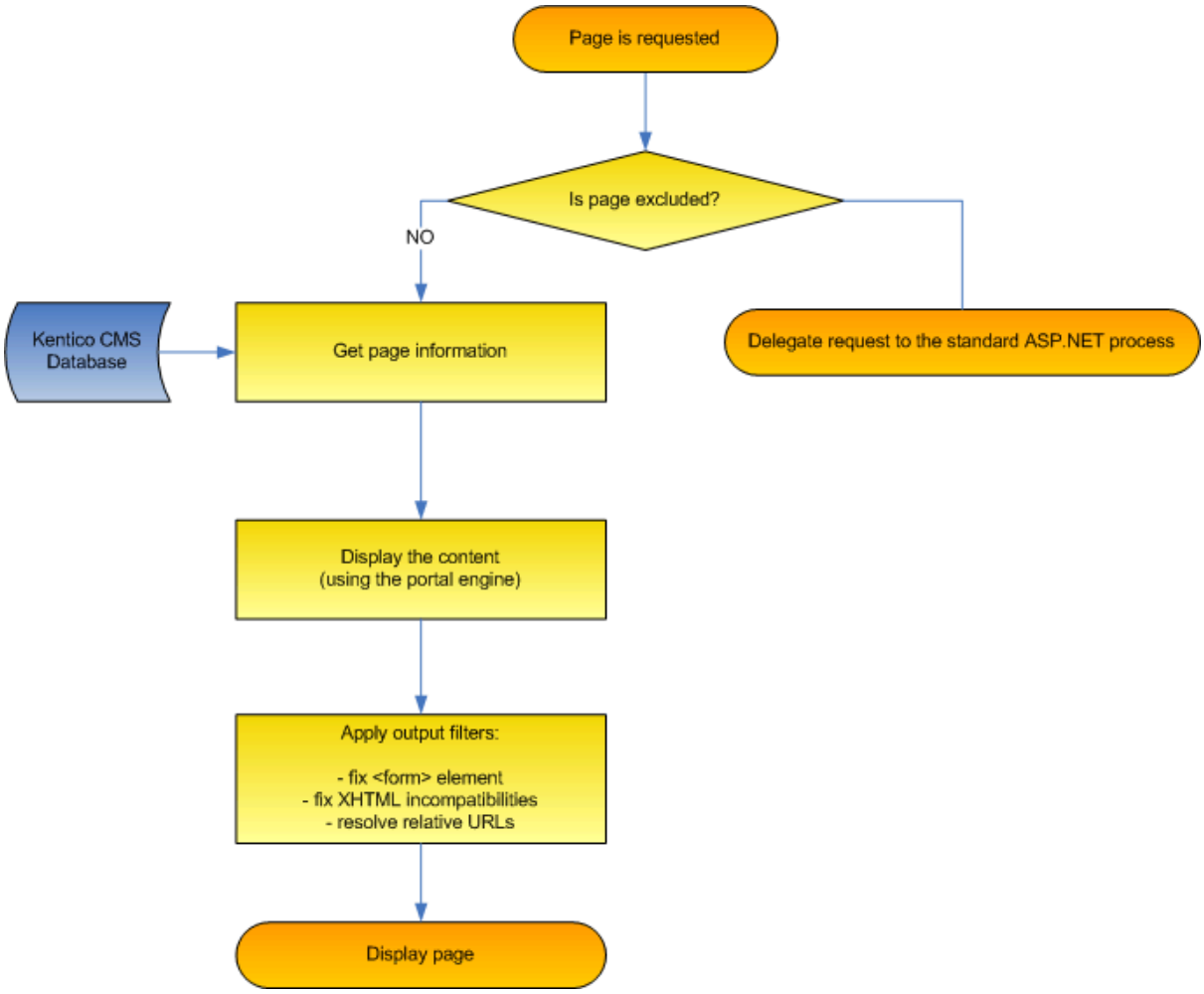
```
private bool DestroyObject()
{
    // Get the css stylesheet
    CssStylesheetInfo destroyStylesheet = CssStylesheetInfoProvider.
GetCssStylesheetInfo("MyNewVersionedStylesheet");

    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;
}
```

7.14 Page processing and URLs

7.14.1 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 URL rewriting engine, it is run through the output filters that ensure additional changes in the rendered HTML code.

The following figure shows the page processing step-by-step:



7.14.2 URL rewriting

Kentico CMS uses a system of friendly URL addresses. It allows you to use URLs like:

`http://www.example.com/products/kentico-cms.aspx`

instead of

`http://www.example.com/products.aspx?id=527`

Every document has its own URL. When multiple languages are used, the document is recognized either using its **URL path** (if specified) or using a combination of **alias path and preferred culture** that is stored in a cookie.

URL processing

1. The system gets the incoming request **`http://www.example.com/products/kentico-cms.aspx`** and looks up the website based on the domain name (either on the main domain name or domain aliases). If it doesn't find any running website, it displays the page **`/cmsmessages/invalidwebsite.aspx`**. If the domain name with the required port number is not found, it also tries to find a site with the same domain

name without any port number.

2. Then it looks up a document with alias path equal to **/products/kentico-cms** and in the current culture. It can optionally also search for the default culture version of the same document in case the required culture version is not available - this can be set in **Site Manager -> Settings -> Content -> Combine with default culture**.

3. If such a URL path is not found, the system tries to find a document with URL path **/products/kentico-cms**.

4. If no document is found for the requested URL, the system doesn't process the request and the web server displays the standard 404 – Page not found error.

5. If the requested document is found, the URL rewriting engine looks up which page template should be used to display it and calls the appropriate page template like this: **/products.aspx?aliaspath=/prodcuts/kentico-cms**. The page template is responsible for the displaying of the document. If the document doesn't have any page template specified, the URL rewriting engine tries to find and use the page template of the parent document. If the page is managed by the portal engine, the page called is **/cmspages/portaltemplate.aspx**, which is a page that renders the page from web parts.

The system of "friendly" or "smart" URLs provides several benefits:

- They are easy to remember and easy to write into the browser address bar.
- They are friendly towards search engines (SEO friendly).
- They show users where they are located on the website.
- You can easily send the URL of the document to your friends and they will see the same page with the particular document.

7.14.3 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:

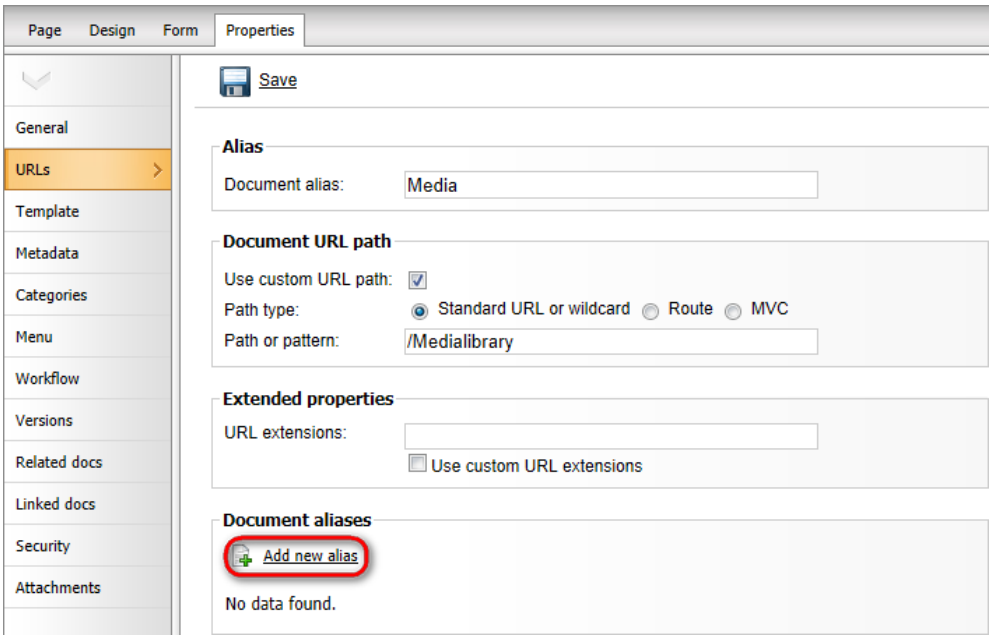
- **Document alias** - this is the unique name of the document in the given section of the website. If *Media* is the value, then *<domain>/Media.aspx* is the URL under which the document can be accessed by default.

- **Document URL path** - the fields in this section can be used to specify an alternate URL for the document. If you enter */Medialibrary*, the page will also become accessible through *<domain>/Medialibrary.aspx*

Further document aliases can be added in the following way:

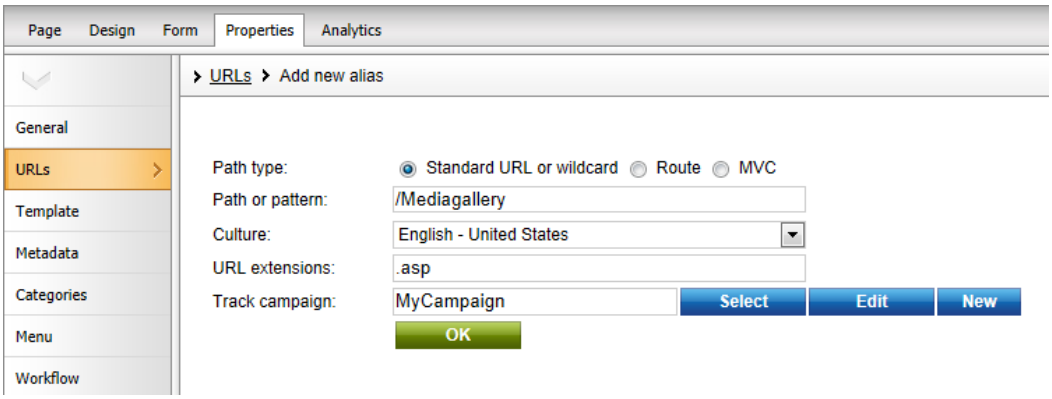
1. Sign in to CMS Desk, select some document from the content tree and switch to its **Properties -> URLs** tab.

2. Click the **Add new alias** link in the **Document aliases** section.



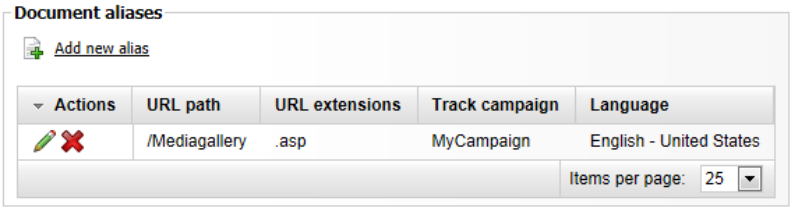
3. Fill in the following details:

- **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.
- **Path or pattern -** enter a URL path for the document alias according to the selected *Path type*:
  - **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 will be handled as requests for an MVC page. Please see the MVC development model chapter for more information.
- **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. E.g. if the controller class is called *NewsMVCController*, enter *NewsMVC*. The specified controller is first searched in the *CMS.Controllers.<current site code name>* namespace. If it is not found there, it is searched in the *CMS.Controllers.Global* namespace.
- **Default action (MVC only)** - specifies the action defined within the controller that should be performed when the alias URL is accessed.
- **Culture** - sets which culture version of the document will 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.



7.14.4 URL format and configuration

Defining the extension of URLs

The URLs can use various extensions. By default, all URLs end with *.aspx*, such as: *http://www.example.com/products/kentico-cms.aspx*

You can also use **custom extensions**, such as *.htm*, *.html* or any custom extension. Alternatively, you can even use **URLs without extensions**, such as *http://www.example.com/products/kentico-cms*. However, in this case, you need to configure the system as described in Configuration of custom URL extensions.

Excluding URLs from the CMS engine

If you need to add your own pages to the website, you may need to exclude them from CMS engine processing. You can do that by adding the page URL (without extension) to the **Site Manager -> Settings -> URLs and SEO -> Excluded URLs** value (you can enter several URLs separated by a semicolon (;).

Forbidden URL characters

URLs (document aliases and URL paths) cannot contain certain special characters. 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" />
```

Using URL Prefixes

If you need to add a prefix to all URLs (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 look like this: *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 contains some of these settings and provides descriptions for them:

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 will be used 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 will be forbidden. If empty, only the characters specified by the **Forbidden URL characters** setting will be 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 would only allow alphanumeric characters and the caret symbol (^) to be used in URLs.
Friendly URL extension	Specifies the extension of friendly URLs. The extension should be preceded by a dot when entered, such as: *.aspx* or *.html*. When you omit this value, the friendly URLs will be the same as the alias path (e. g. */products/nokia*). **Please note**: The system of "friendly" or "smart" URLs provides several benefits. They are easy to remember and easy to write into the browser address bar. They are search engine friendly (SEO friendly). They show users where they are located on the website. You can easily send the URL of the document to your friend and she will see the same page with the particular document.
Files friendly URL extension	Specifies the extension for files that will be used in friendly URLs, such as ".aspx" or ".html". **Example**: *getfile/<node alias>/myimage.aspx* When you omit this value, the friendly URLs of files will end with no extension: *getfile/<node alias>/myimage*
Excluded URLs	List of URLs (without domain) that should be excluded from the CMS engine. You can enter several paths separated by a semicolon (;).
Document URLs	
Default URL path prefix	Defines a default URL path prefix that will be used for all URLs of the content pages. This prefix is rewritten to *urlpathprefix* query string parameter.

Use name path for URL path	If checked, this key indicates that 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 at Linking pages and files.
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.

SEO related settings can be found in the Search engine optimization chapter.

7.14.5 Search engine optimization

Search engine optimization (SEO) is a process which attempts to place a website earlier (higher) in search engine results and thus attract more visitors. Kentico CMS has several settings which can be useful when optimizing websites.

SEO related settings can be configured at **Site Manager -> Settings -> URLs and SEO**. The following table contains these settings and provides descriptions:

Google sitemap URL	URL that will be automatically rewritten to the physical location of the Google Sitemap. The physical location is *~/CMSPages/GoogleSiteMap.aspx*. More information can be found at Google Sitemap .
Allow permanent (301) redirection	If enabled, the system uses permanent (301) redirection instead of standard (302) redirection for SEO purposes.
Use URLs with trailing slash	Specifies how URLs with trailing slash should be handled. Possible options: Leave the URL as is, Always use URLs with trailing slash, Always use URLs without trailing slash.
Redirect document aliases to main URL	If enabled, every document always has only one valid URL and other aliases are redirected to this main URL, for SEO purposes.
Redirect invalid case URLs to their correct versions	Checks the URL letter case, and if configured, redirects to their correct version. Options: Do not check the URLs case, Use the exact URL of the document, Redirect all requests to lower case URLs, Redirect all requests to upper case URLs.
Use language prefix for URLs	If enabled, all document URLs will be generated with language prefixes. The language prefix is identical to to the culture code or culture alias (if set) of the currently selected content culture. **Example**: *<domain>/en-US/Home.aspx*
Allow URLs without language prefixes	If enabled, URLs without language prefixes are allowed. Otherwise they are redirected to a corresponding URL with a language prefix. This setting is usable only if **Use language prefix for URLs** is enabled.

Redirect documents to main extension	If enabled, URLs will be redirected to a corresponding URL with the current main extension if their extension is different. The main extension is the first one specified in the **Friendly URL extension** field.
Move ViewState to the end of the page	If enabled, the hidden ViewState field is relocated to the end of the page, allowing more content to be processed by search engines.

To configure the metadata of individual documents, select one from the content tree of CMS Desk and go to **Properties -> Metadata**. Here you can set SEO related properties such as the page title, description and keywords. More details can be found at Content management -> Document properties -> Metadata.

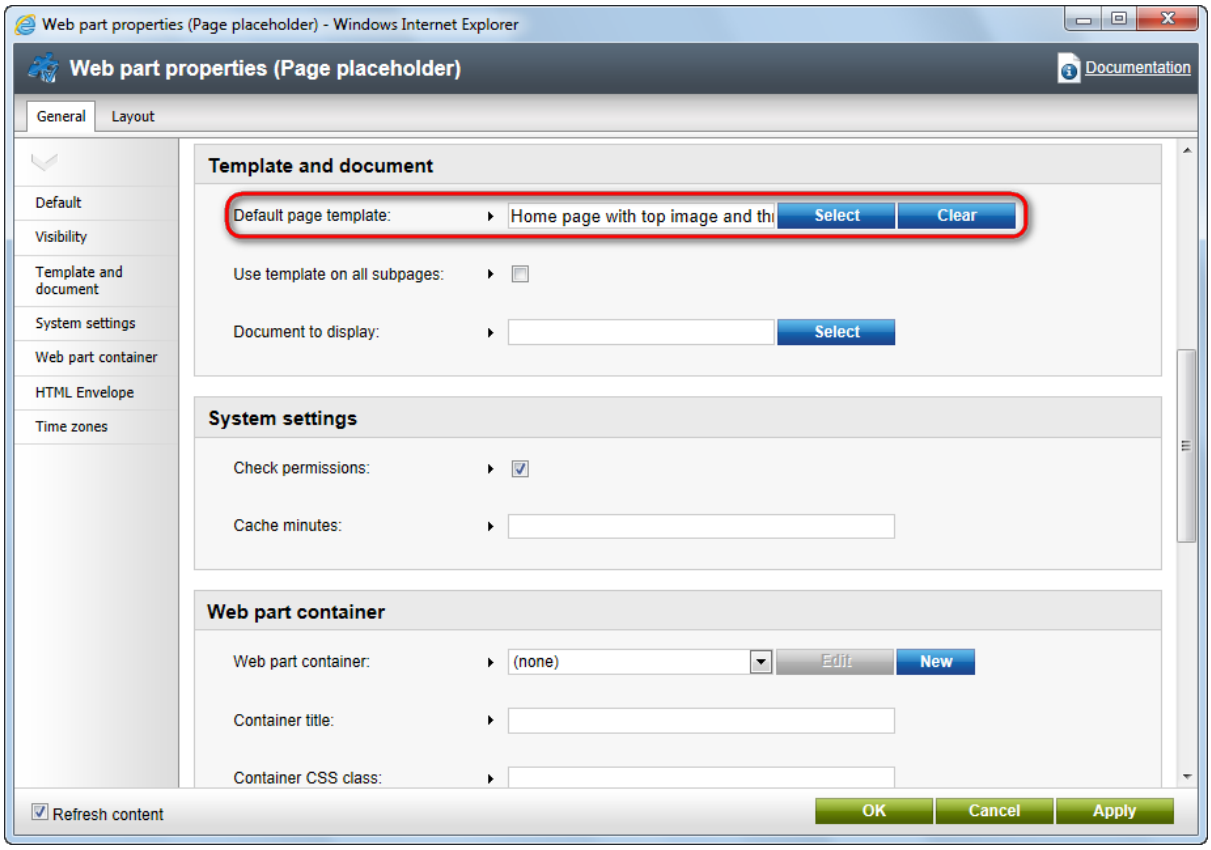
How to set your domain URL as your home page

For SEO purposes, it can be useful to have all URLs that access the Home page of your website redirect to one single URL. The following steps describe how to configure Kentico CMS so that all these URLs are redirected specifically to the base URL of your domain:

1. Go to **Site Manager -> Settings -> Content**, select the site you wish to configure from the drop-down menu, delete the contents of the **Default alias path** field and click **Save**.

2. Switch to CMS Desk, select your Home page and go to **Properties -> Menu**, select the **URL redirection** radio button, enter *http://<domain>/* into its field and click **Save**. This step will ensure that all attempts to access your Home page are redirected to the root of your website.

3. Select the root of your website, switch to the **Design** tab and **Configure** () the **Page placeholder** web part. Find the **Default page template** field, press the **Select** button and choose the same page template that your Home page uses (this can be found by selecting your Home page and going to **Properties -> Template**). Click **OK**.



This will cause the root of your website to display your Home page, but be aware that this does not transfer any property settings your Home page document may have had or the contents of Editable text and Editable image web parts. These will have to be entered again for the root document. Also make sure that the **Web part control ID** properties of all web parts on your Home page are different than those of web parts on the root of your website.

4. Add the following key to the **/configuration/appSettings** section of your web.config file:

```
<add key="CMSDefaultPageURL" value="~/" />
```

This will change the default URL of your website root to <domain>/.

5. If your instance of Kentico CMS is running on a server with IIS 7 or higher installed, add the following key to the **/configuration/appSettings** section of your web.config file:

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

Now find the opening *<modules>* tag in the **/configuration/System.webServer** section of your web.config file and change it to the following:

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

Then go to **Site Manager -> Settings -> URLs and SEO** and enable the **Redirect document aliases to main URL** setting. This step will ensure that the <domain>/default.aspx URL is always redirected to <domain>/.

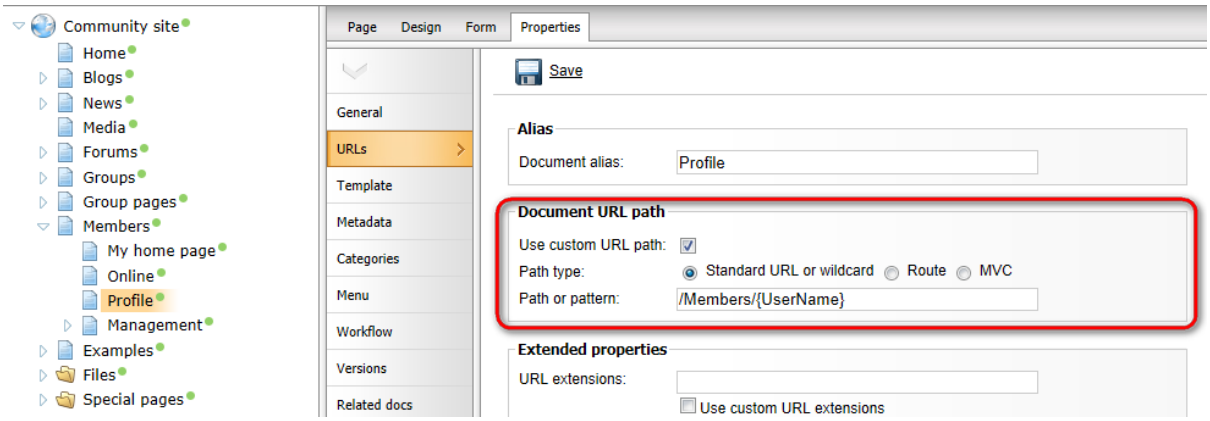
Important!

This step is not possible when running on IIS versions lower than 7, as they do not differentiate between <domain>/ and <domain>/default.aspx and adding the key to your web.config file may cause an infinite redirection loop. If you are running on IIS versions lower than 7, follow only steps 1, 2, 3 and 4.

7.14.6 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 will be displayed. The system translates 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

Wildcards in URLs have the option of setting default values. This can be done by using the following format: **{<wildcard name>;<Default value>}**. This is useful when a document has a wildcard in its URL path, but you 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, 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.

You may also choose to take the current value of a wildcard and use it 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**.

This functionality can be enabled by adding the **CMSUseCurrentWildcardValueAsDefaultValue** key into the **/configuration/appSettings** section of your *web.config* file, as shown below:

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

Please note that even if this key is set to *true*, the current value will not be transferred to wildcards that already have a default value specified in the URL path.

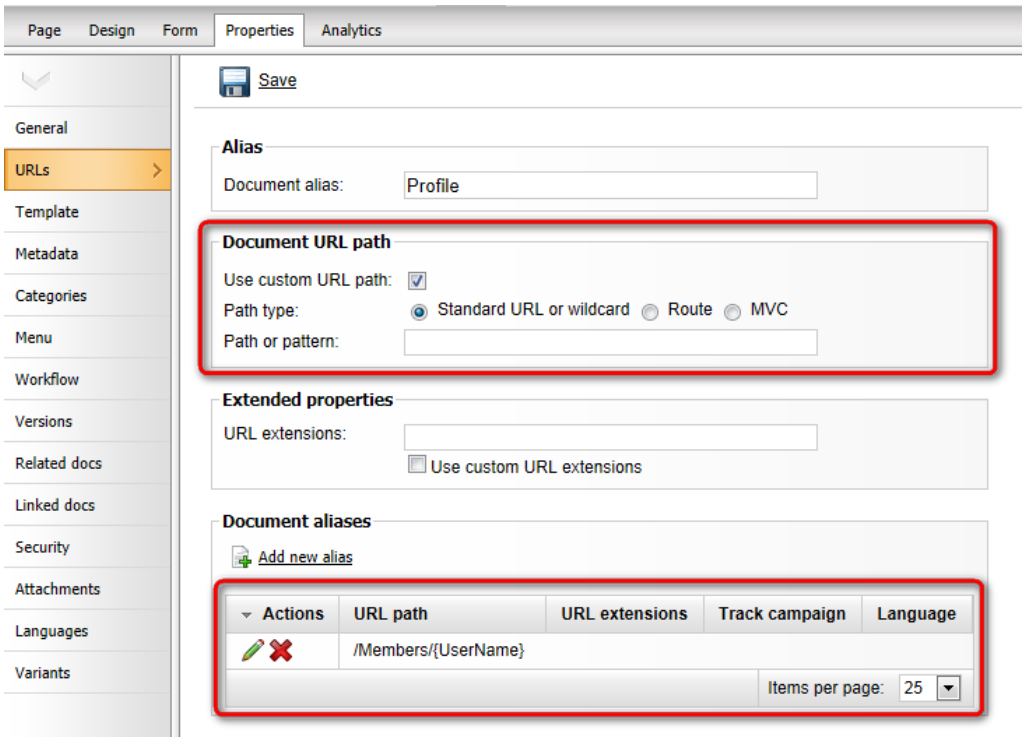
Using wildcard URLs on multi-language sites

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 by a validation of the **UserName** field in the **Registration form** alternative form of the **User** system table.

If you needed to allow dots in user names and use wildcard URLs with user names at the same time, you can achieve this 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.

7.14.7 Linking pages and files

Linking documents (pages)

If you need to create a permanent link to a document, you need to use a URL in the following format:

```
http://www.example.com/getdoc/016fad52-0d69-46d5-80dc-daec9173c0c7/Products.aspx
```

It's an equivalent of:

```
http://www.example.com/company/products.aspx
```

However, in the first case, the link keeps working even if you move the document to some other place.

The URL consists of the following parts:

<domain>/getdoc/<document GUID>/<document name><extension>

The <*document GUID*> value is a unique identifier of the document. You can find this value at **CMS Desk -> Content -> Properties -> General**, in the **Node GUID** field.

The <*document name*> value may contain any value - it's not used by the system and it's only used for search engine optimization. By default, the system uses the document name for this value.

Linking a specific language version of the document

If you need to link to a specific language version of the document, you need to use a URL in the following format:

```
http://www.example.com/getdoc/8FG7-84E394-FABD-5678/our-services/fr-fr.aspx
```

It displays the given document in French (if the document is translated). It's an equivalent of

```
http://www.example.com/company/our-services.aspx?lang=fr-fr
```

The URL consists of the following parts:

<domain>/getdoc/<document GUID>/<document name>/<culture code><extension>

Linking attachments

If you need to create a permanent link to a file uploaded as a document attachment, you need to use a URL in the following format:

```
http://www.example.com/getattachment/763c8921-be94-4610-99b4-25e8d3be5b08/logo.aspx
```

The URL consists of the following parts:

<domain>/getattachment/<file GUID>/<filename><extension>

The <*file GUID*> value is not the same as the document GUID. It's a GUID of the file in the *CMS_Attachment* table. You can find this GUID if you display the attachment in **CMS Desk** and view its URL.

The <*file name*> value can contain any text.

You can find more details on available parameters in the following topic: GetFile.aspx parameters.

7.14.8 GetFile.aspx parameters

The GetFile.aspx script is used in many cases to retrieve an uploaded file from the database. It is called whenever you use /getdoc, /getattachment or a direct URL based on the alias path of a cms.file document.

The GetFile.aspx script accepts the following URL 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 either **inline** (opens the file in the browser window if possible) or **attachment** disposition (opens the "Save or Open" dialog).	inline or attachment

7.14.9 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.

Form filter

The form filter fixes the issue with non-working postbacks on pages that use URL rewriting. It ensures that forms, dialogs and buttons will work correctly on Kentico CMS-managed pages.

XHTML filter

The XHTML filter fixes certain types of XHTML incompatibilities. It closes unclosed tags, invalid <script> tags, etc.

XHTML errors may also be fixed in the WYSIWYG editor when they are saved. This can also be configured globally through the **CMSWYSIWYGFixXHTML** web.config key (supported values are "true" and "false").

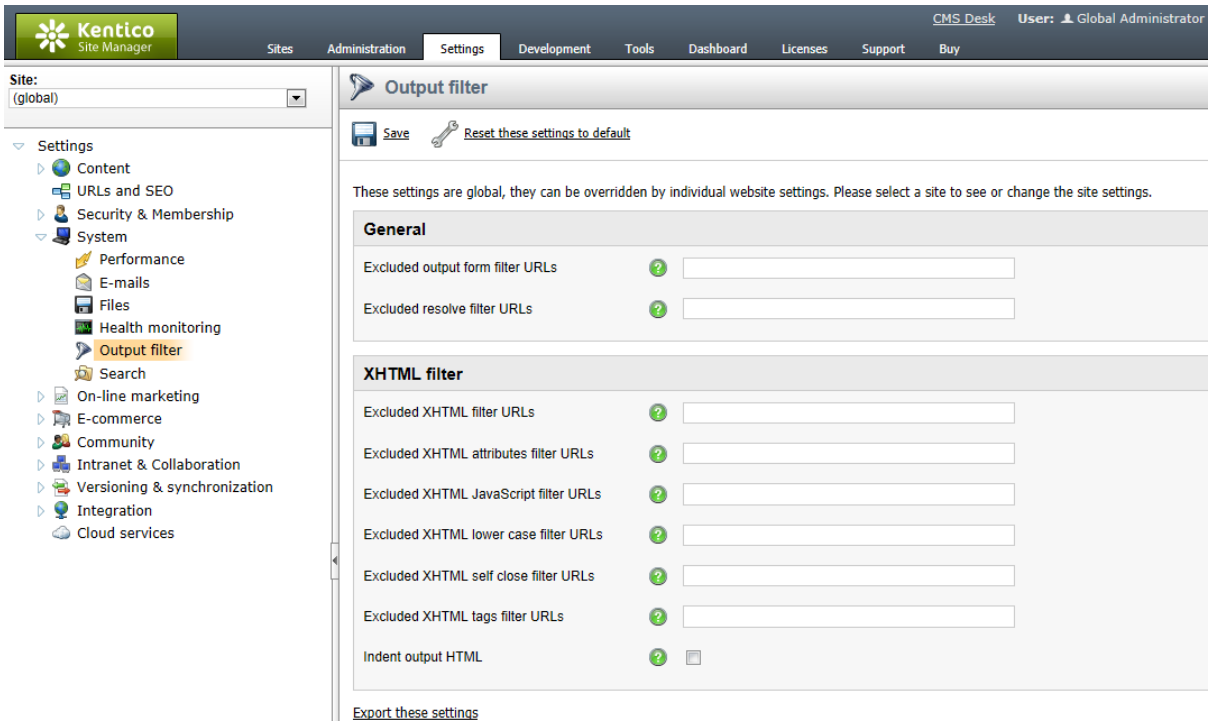
Resolve filter

The resolve filter changes relative URLs in format `~/mypage1/mypage2.aspx` to `/application/mypage1/mypage2.aspx` (application running in a sub-folder) or `/mypage1/mypage2.aspx` (if the application is running in the root) to their full version. Only URLs inside **src** and **href** attributes are changed.

Output filter settings

The following settings can be adjusted in **Site Manager -> Settings -> System -> Output filter**:

General	
Excluded output form filter URLs	List of URLs (without domain) that should be excluded from output form filter. You can enter several paths separated by a semicolon (;).
Excluded resolve filter URLs	List of URLs (without domain) that should be excluded from the 'ResolveURL' output filter that ensures replacing of '~' character in URLs with website root URL. You can enter several paths separated by a semicolon (;).
XHTML filter	
Excluded XHTML filter URLs	List of URLs (without domain) that should be excluded from the XHTML output filter. You can enter several paths separated by a semicolon (;).
Excluded XHTML attributes filter URLs	List of URLs (without domain) that should be excluded from Tag parameter output filter. This output filter ensures that the HTML tag attributes are in a valid form.You can enter several paths separated by a semicolon (;).
Excluded XHTML JavaScript filter URLs	List of URLs (without domain) that should be excluded from Javascript output filter that ensures presence of type and language attributes.You can enter several paths separated by a semicolon (;).
Excluded XHTML lower case filter URLs	List of URLs (without domain) that should be excluded from Lowercase output filter that ensures that the HTML tags and attributes are lowercase. You can enter several paths separated by a semicolon (;).
Excluded XHTML self close filter URLs	List of URLs (without domain) that should be excluded from Selfclose output filter that ensures that the HTML tags without end tags are properly closed. You can enter several paths separated by a semicolon (;).
Excluded XHTML tags filter URLs	List of URLs (without domain) that should be excluded from Tag output filter that ensures that HTML tags use their appropriate versions (instead of <b> and instead of <i>). You can enter several paths separated by a semicolon (;).
Indent output HTML	Indicates if the HTML output of 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.

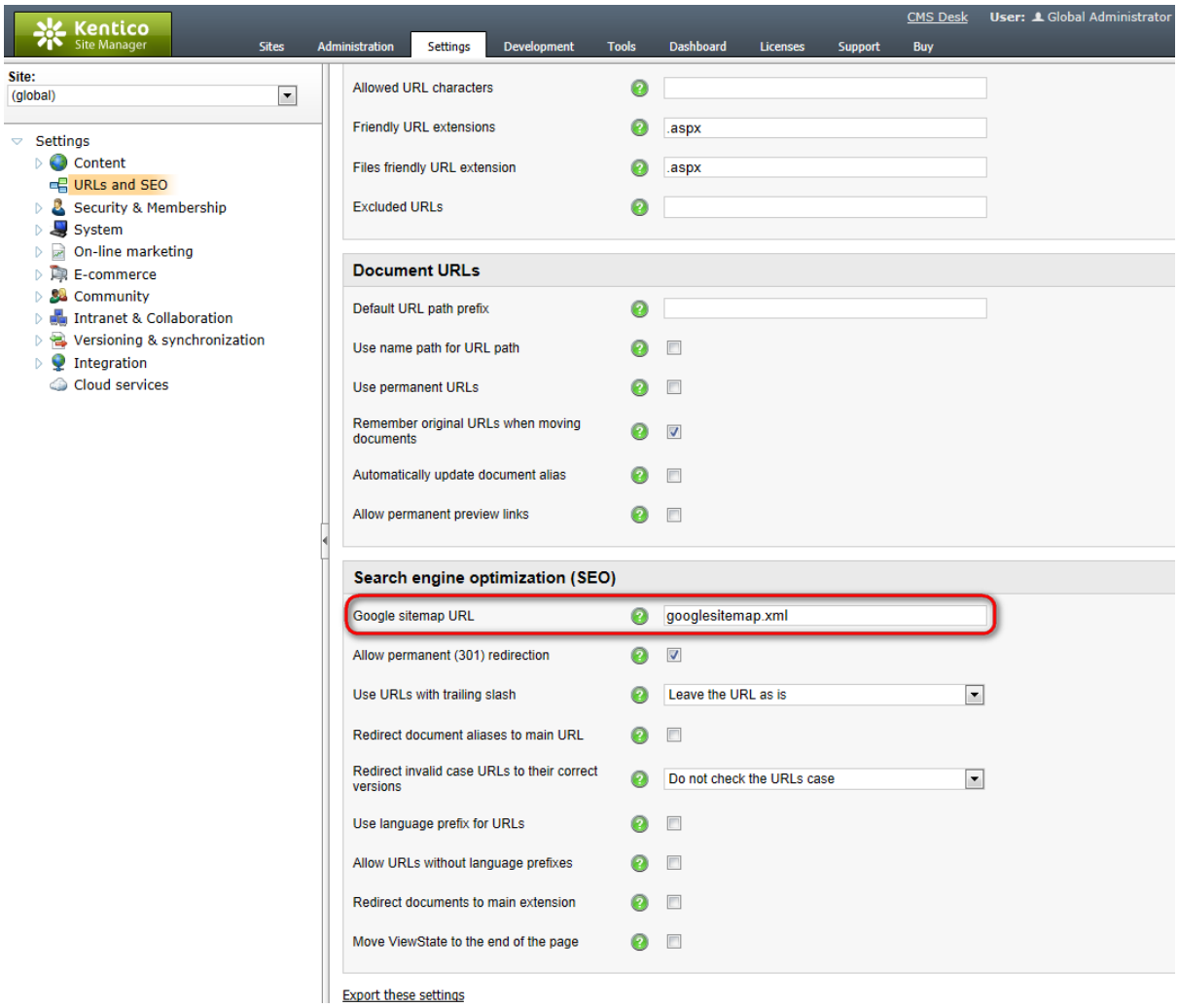


7.14.10 Google Sitemap

Kentico CMS comes with automatic support of the Google SiteMap Protocol. This is a protocol designed to help search engines with the indexing of your site and can improve your site's position in search engine results. Simply put, a Sitemap is an XML file where URLs of your site are stored. Detailed information about the SiteMap Protocol can be found at http://en.wikipedia.org/wiki/Site_map.

Kentico CMS creates your site's sitemap automatically. Physically, it is located at *<domain>/CMSPages/googlesitemap.aspx*. It can also be accessed through the URL specified in **Site Manager -> Settings -> URLs and SEO -> Google sitemap URL**. This value is set to *googlesitemap.xml* by default, which means that the site's sitemap can also be accessed through *<domain>/googlesitemap.xml*. This is the URL that you enter into google when registering a new sitemap.

Please note that if you want the sitemap to be accessible via a URL with the **.xml** extension, you need to configure your IIS to handle the **404 error** through Kentico's **handler404**, as described in the Installation and deployment -> Additional configuration tasks -> Custom URL extensions and extensionless URLs chapter. In this case, it is also necessary **not to set** the .xml extension to be processed by .NET.



7.14.11 Document aliases internals and API

7.14.11.1 Overview

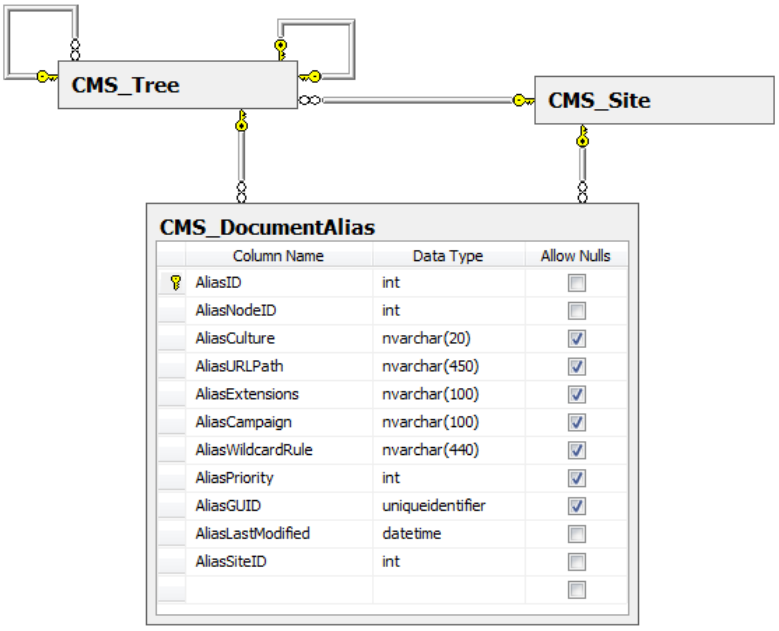
In this chapter, you will learn which database tables and API classes are used to manage document aliases. You will also see the most common API examples.

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.

7.14.11.2 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.



7.14.11.3 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.TreeEngine** namespace.

CMS_DocumentAlias table API:

- **DocumentAliasInfo** - represents one document alias.
- **DocumentAliasInfoProvider** - provides management functionality for document aliases.

7.14.11.4 API examples

7.14.11.4.1 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.TreeEngine;
```

7.14.11.4.2 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.

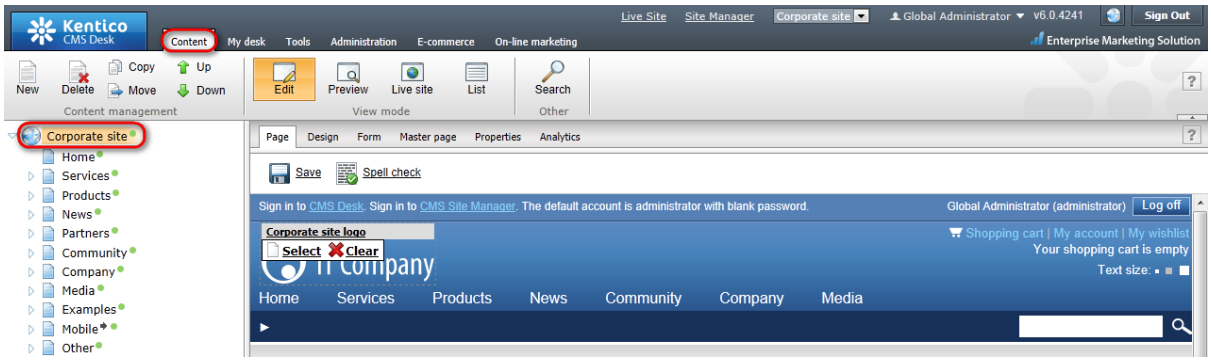
```
private bool DeleteDocumentAlias()
{
    // Prepare the parameters
    string orderBy = "";
    string where = "AliasURLPath = N'/MyNewAlias'";

    // 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;
}
```

7.15 Rebranding

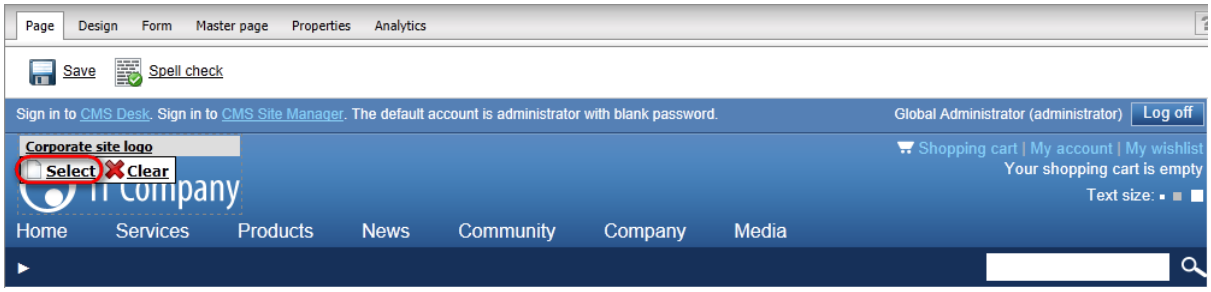
7.15.1 Changing the logo in the header

In this topic, you will learn how to change the header logo on the sample Corporate Site.

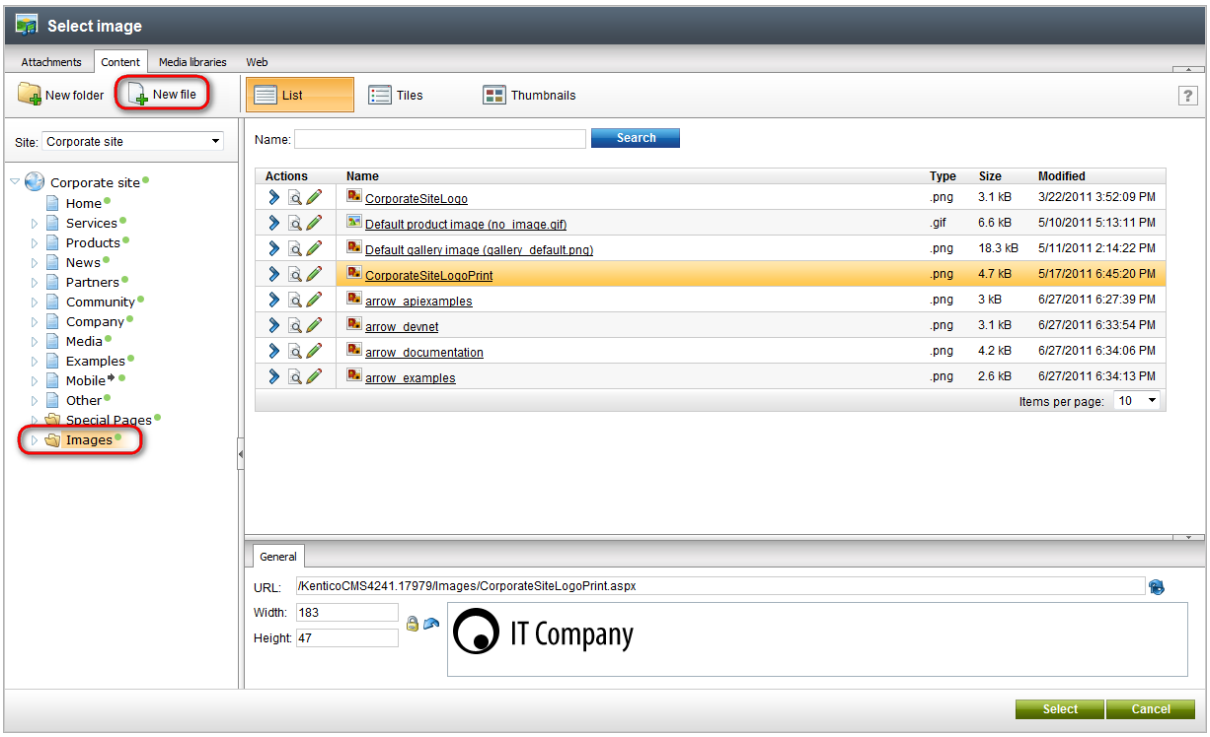
1. Go to **CMS Desk** -> **Content** and select the root of the content tree (**Corporate Site**).



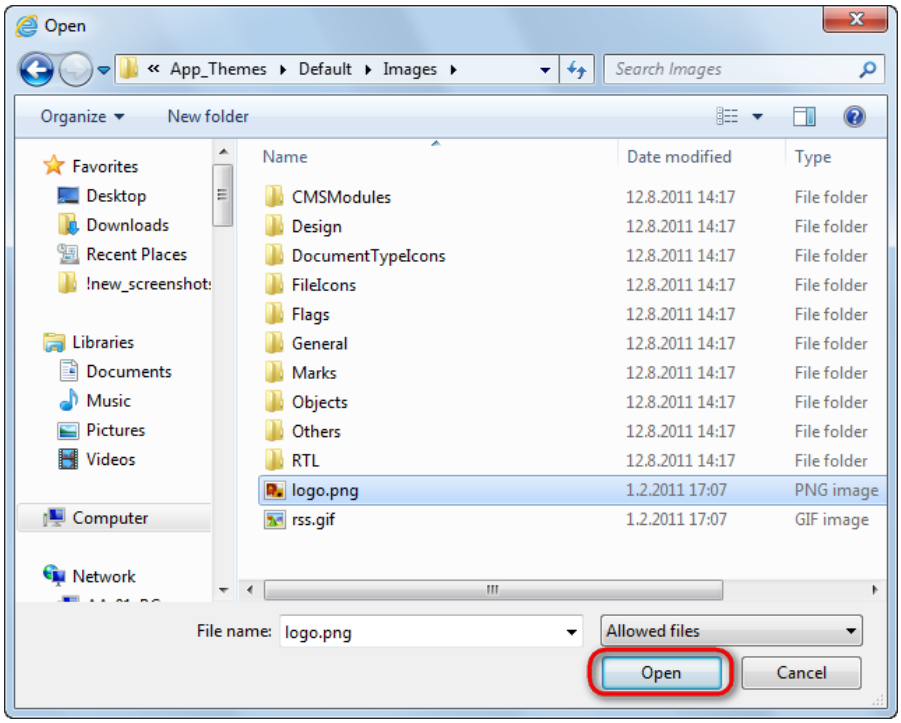
2. On the **Page** tab, click the **Select** button displayed over the logo.



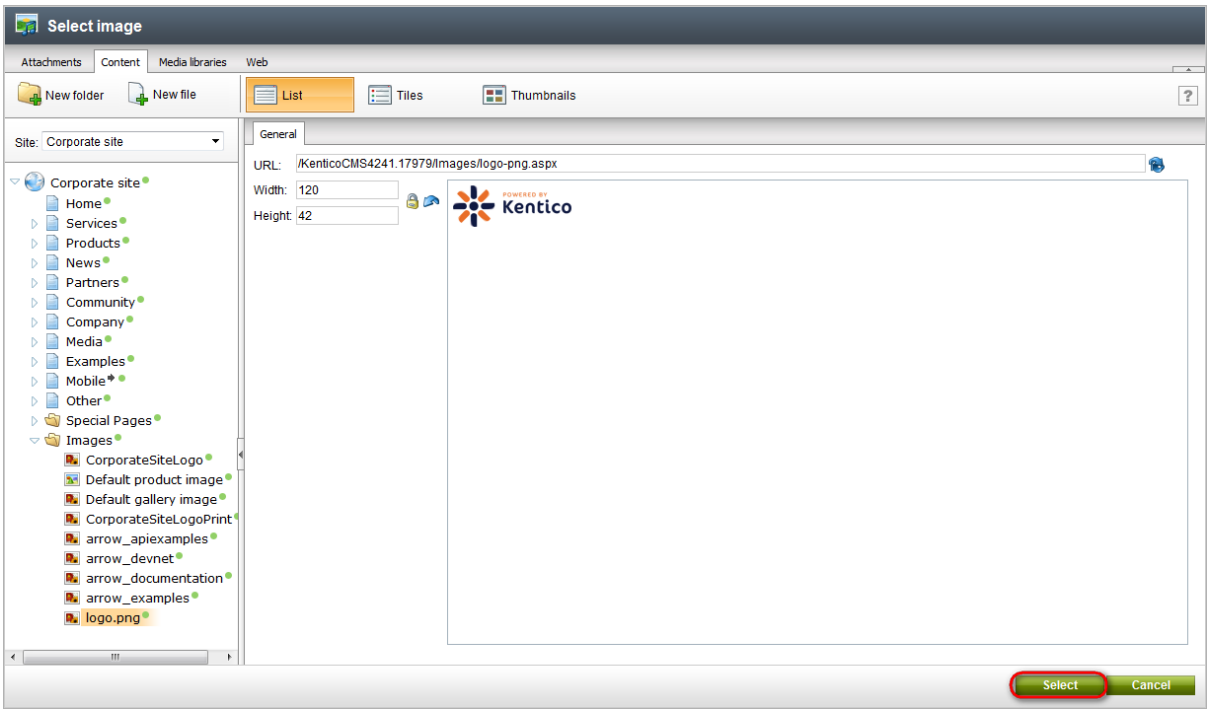
3. In the **Select image** dialog that pops up, select **Images** in the content tree and click the **New file** button above the content tree.



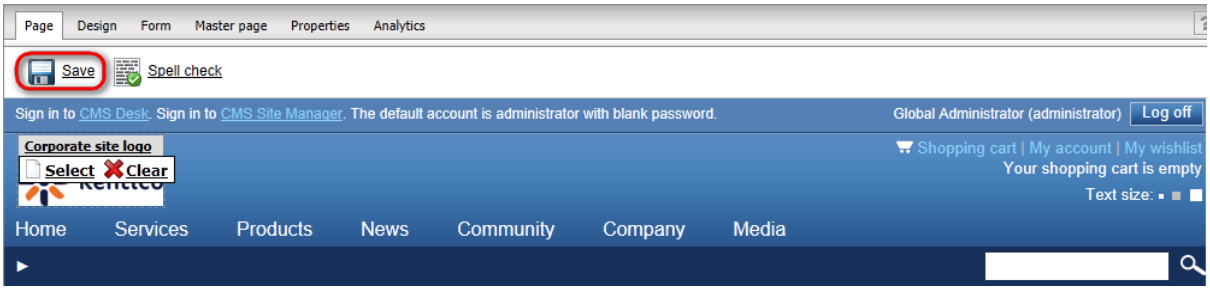
4. Find an image you want to upload and click **Open**.



7. The logo should be added to the Images folder. With the logo selected in the content tree, click **Select**.



8. Now click **Save**.

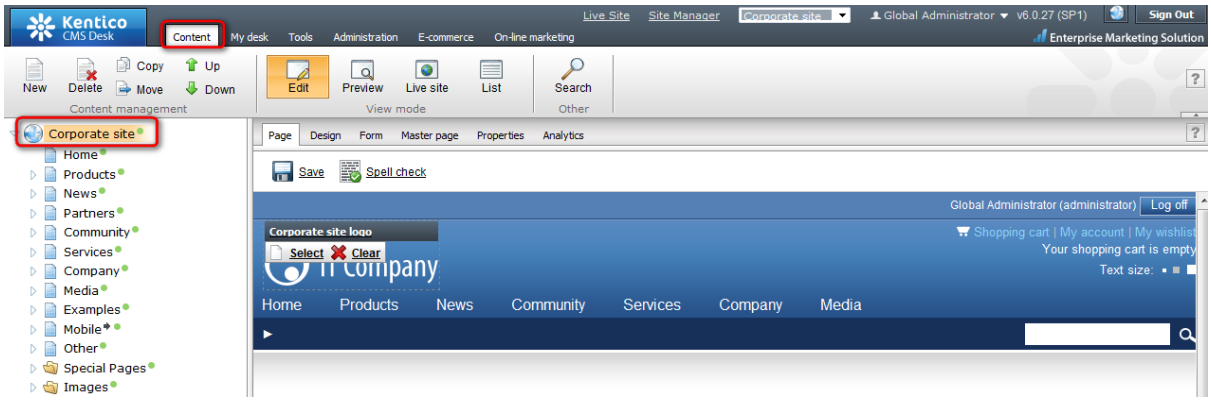


You've just publish a new logo on your website.

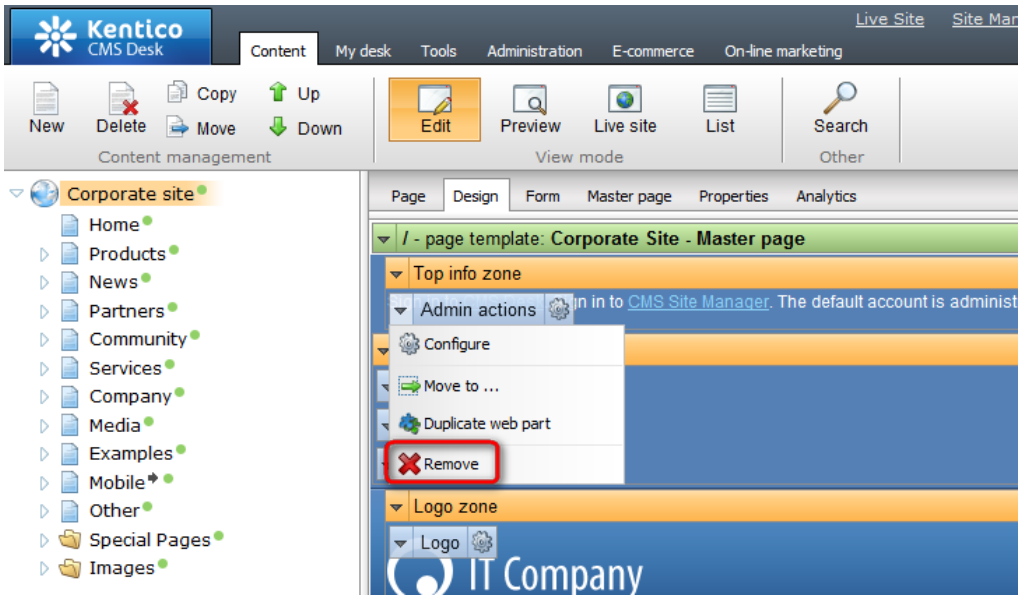
7.15.2 Removing the log-on bar

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:

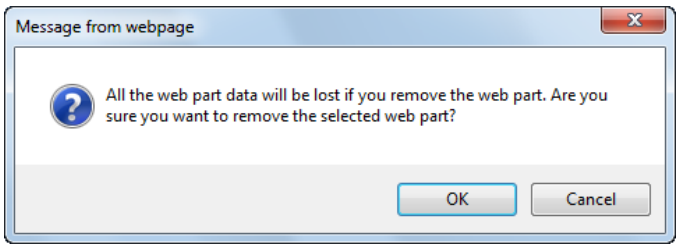
1. Go to **CMS Desk -> Content** and select the root of the content tree (**Corporate Site**).



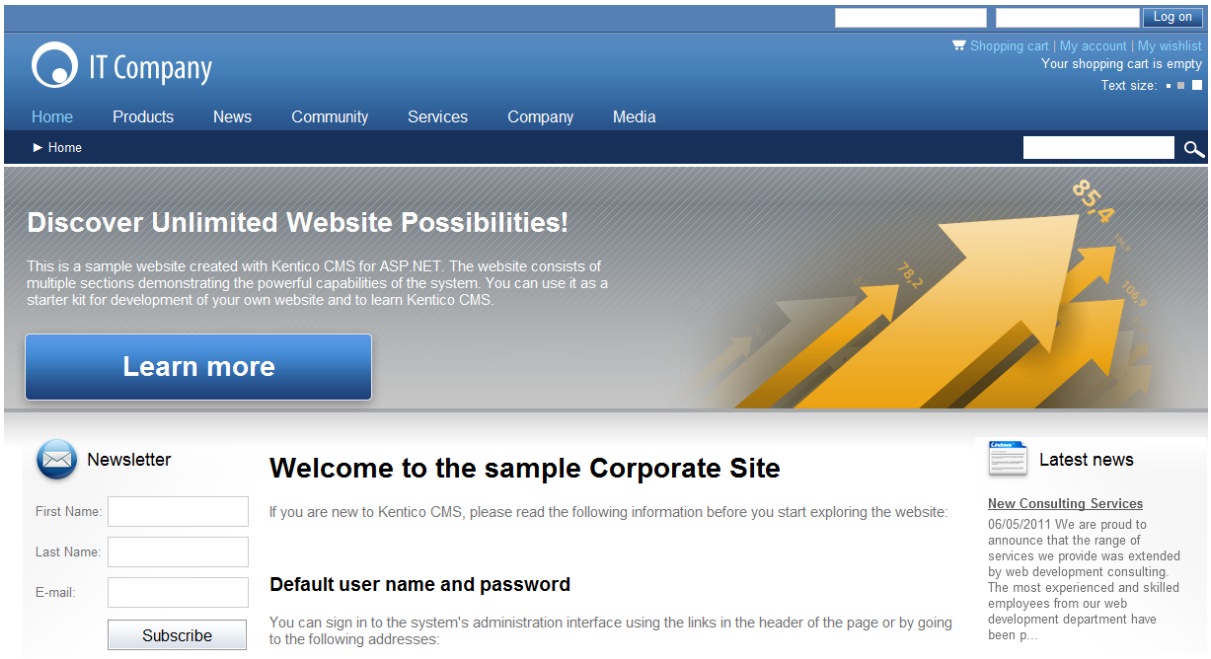
2. Switch to the **Design** tab, right click the **Admin actions** web part in the **Top info zone** and choose the **Remove** option.



3. Click **OK** to remove the links from your website.

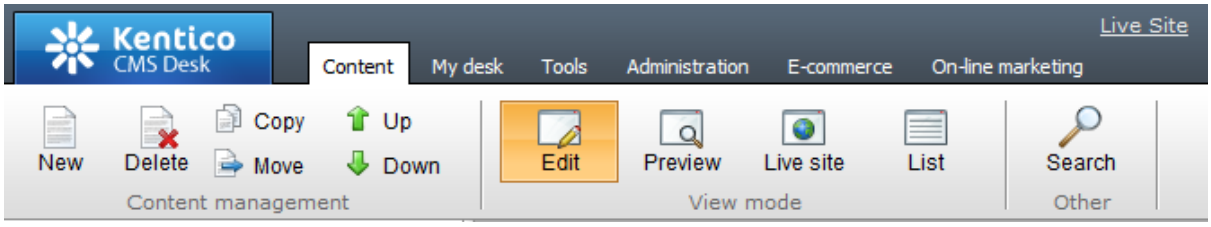


If you view the live site, the page will be displayed without the CMS Desk and Site Manager login links.



7.15.3 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\Design\Branding** folder in your Kentico CMS project's directory. Then simply replace the image files with your own logos.

The logon page also contains several images that you may wish to replace. These can be found in the **App_Themes\Default\Images\Others\LogonForm** directory.

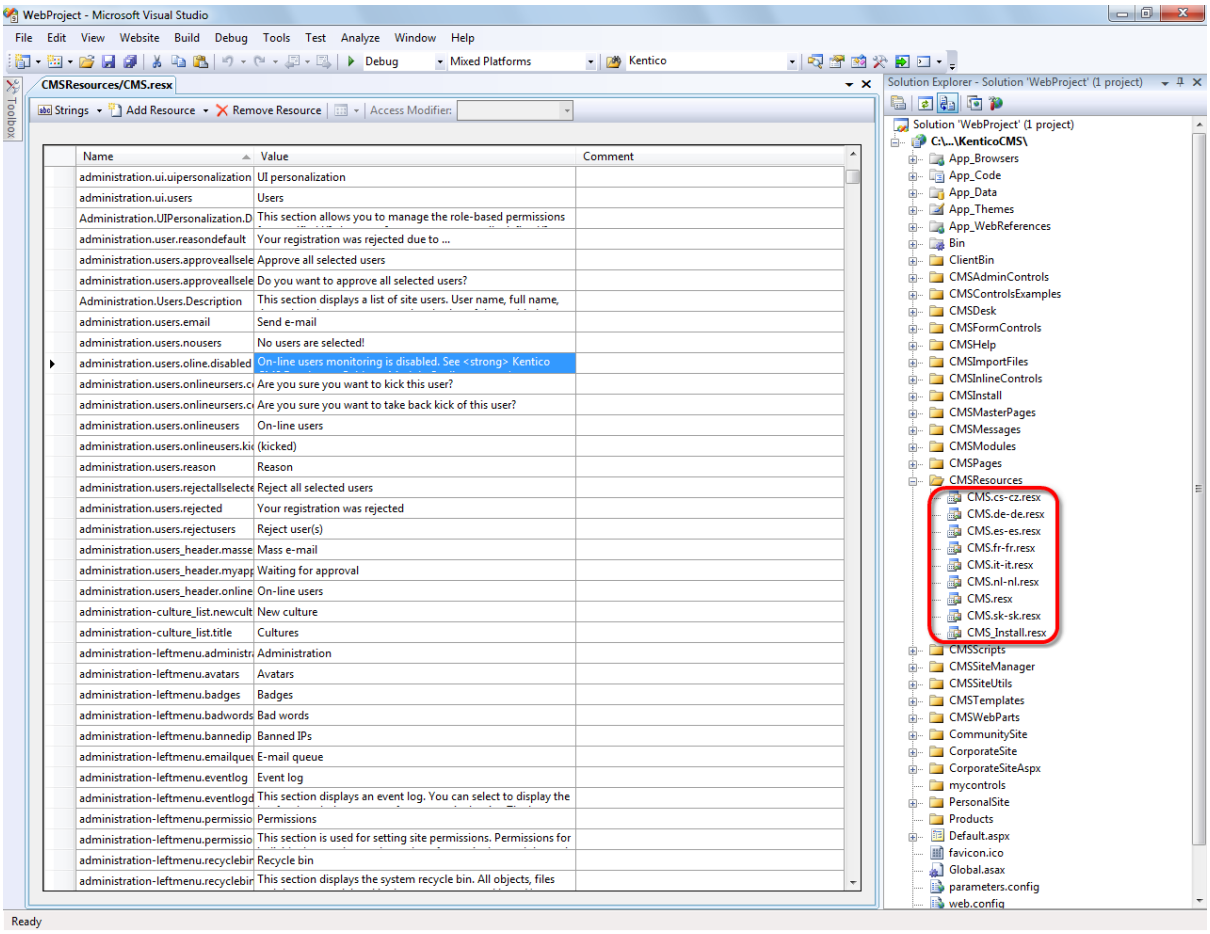
Please note

The names of the image files must remain unchanged, otherwise it will be necessary to 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.

7.15.4 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.



7.16 REST service

7.16.1 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 can be performed by requesting simple and intuitive URLs. 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 pre-requisites need to be met on the machine where the Kentico CMS instance is installed. To learn more, please proceed to the Pre-requisites topic. Once you have the pre-requisites met, you can proceed to the Configuration for REST topic, where required configuration of a particular Kentico CMS instance is described.

Once all the configuration is performed, the REST service should be functional. At this point, you can proceed to the Object methods and Document methods to learn how you can preform object and document data retrieval and manipulation requests. 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.

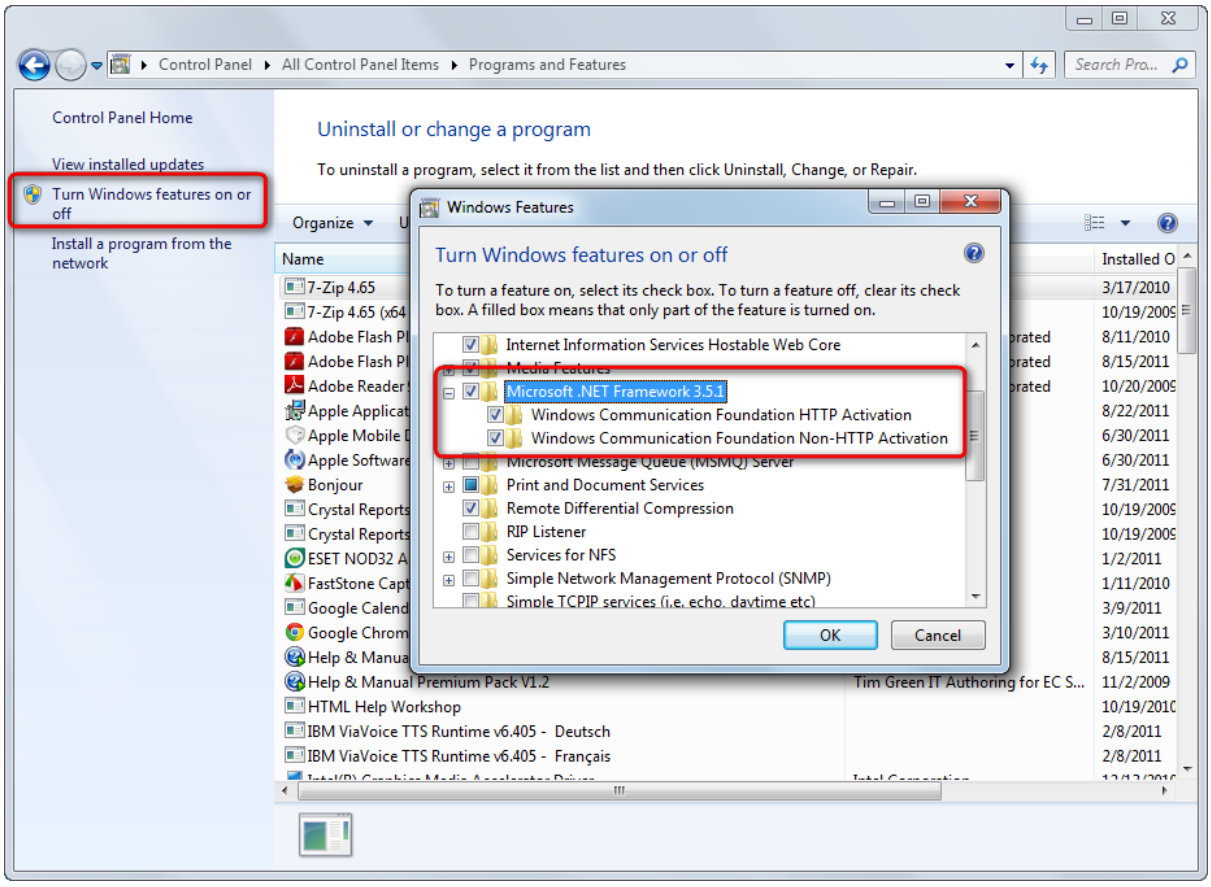
The REST service also supports ODATA browsing by providing service documents with information about data that can be obtained by the service. More information on this can be found in the ODATA service documents topic.

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.

7.16.2 Pre-requisites

In order for the Kentico CMS REST service to be functional, you must ensure the following:

1. In Windows, go to **Control Panel -> Programs and Features** and choose the **Turn Windows features on or off** option from the left menu. In the pop-up dialog, expand the **Microsoft .NET Framework <version>** node and make sure that both **Windows Communication Foundation HTTP Activation** and **Windows Communication Foundation Non-HTTP Activation** are installed.

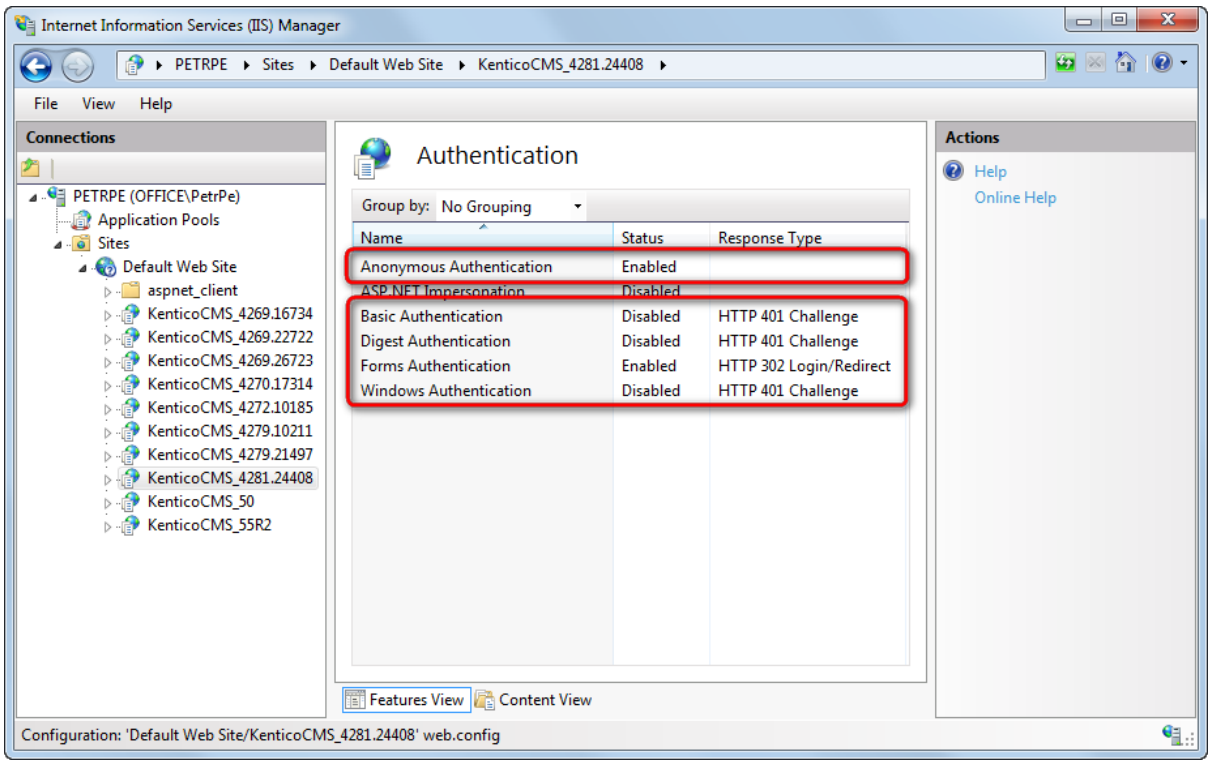


2. In **IIS Manager**, select the website for which you want REST to be enabled and go to its **Authentication** configuration. Here, ensure that:

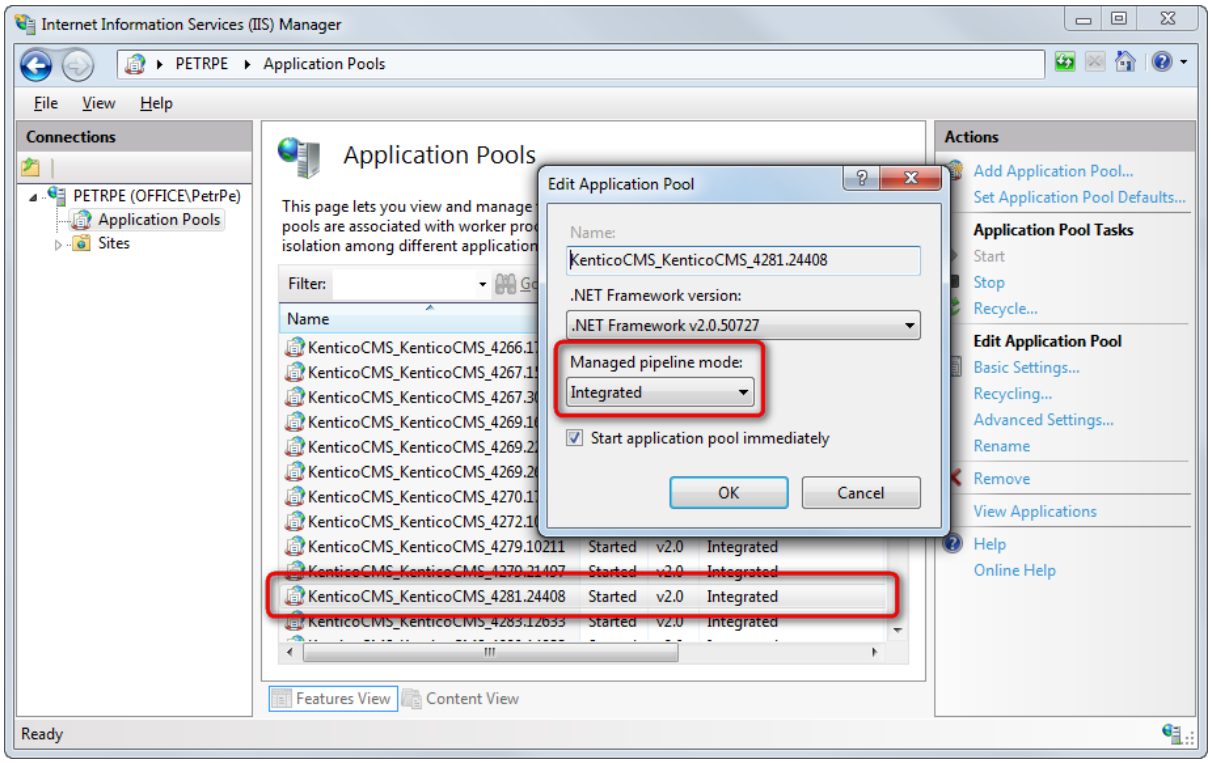
- **Anonymous** authentication is enabled.
- either **Basic** or **Forms** authentication is enabled, while both can't be enabled at the same time. The enabled one is the actual authentication type used by the REST service. No other authentication

types than **Basic** and **Forms** are supported.

- **Digest** and **Forms** authentication is disabled.



3. Still in IIS, choose **Application Pools** in the tree on the left. From the list of application pools, choose the pool used by your website and make sure that it uses the **Integrated** managed pipeline mode.



Once you have these pre-requisites met, you can proceed to the Configuration for REST topic, where required configuration of a particular Kentico CMS instance is described.

7.16.3 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. First, you need to add an attribute to the **<modules>** element in the **system.webServer** section of your **web.config** file:

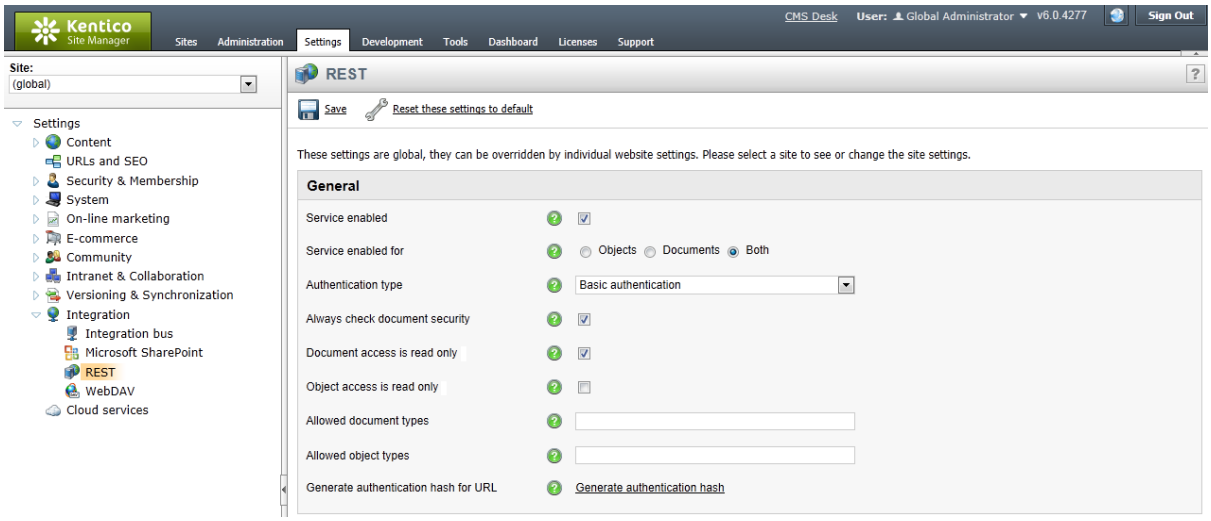
```
<modules runAllManagedModulesForAllRequests="true">
```

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	Allows to choose if the REST service should be used for access to objects, documents or both.
Authentication type	Determines which type of authentication will be used by the REST service. Supported types are **Basic** and **Forms** authentication.
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, only GET document requests can be processed by the service and documents can't be modified by the service.
Object access is read only	If enabled, only GET object requests can be processed by the service and objects can't be modified by the service.
Allowed document types	List of document types which will be accessible by the service. If the field is empty, all document types will be accessible.
Allowed object types	List of object types which will be accessible by the service. If the field is empty, all object types will be accessible.
Generate authentication hash for URL	Click the link to open a dialog where you can generate an authentication hash for a specified URL. This hash can be used instead of authentication afterwards to access specified data.



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 required to specify the required data size limit in the instance's *web.config* file. This can be done by adding the following piece of code directly under the root element of the *web.config* file.

Please note that the code sample below sets all limits to 10 MB, while you may need to specify different values according to your specific needs. The *baseAddress* in the code below has also only a sample value and needs to be replaced with the actual root address of the REST service.

```
<system.serviceModel>
  <bindings>
    <!-- Customizations for REST service -->
    <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>
  </bindings>
  <services>
    <service name="CMS.RESTService.RESTService">
      <host>
        <baseAddresses>
          <add baseAddress="http://<your website domain>/rest" />
        </baseAddresses>
      </host>
      <endpoint address="" bindingConfiguration="RESTQuotaBinding" binding="
webHttpBinding" contract="CMS.RESTService.IRESTService" />
    </service>
  </services>
</system.serviceModel>
```

7.16.4 Object methods

In this topic, you can find examples of REST methods that can be used to retrieve or manipulate object data from a Kentico CMS instance using the REST service.

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. The URLs 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*. You can also achieve a more detailed specification of retrieved data by appending querystring parameters to the URLs. More information can be found in the URL parameters topic.

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*

- 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, UTF8 is used by default. Please note that this feature is only available in Kentico CMS 6.0 with applied hotfix 6.0.13 or later.

For example:

```
POST 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
```

Object data manipulation methods

Besides the GET requests listed above, the REST service also supports PUT, DELETE and POST Http methods. You can update, delete and insert objects (respectively) using these Http methods. Both XML and JSON formats are supported when performing these requests.

The following URLs can be requested by the respective type of request in order to manipulate Kentico CMS objects:

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*

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.

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 a null value in an object 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.

Please note that this feature is only available in Kentico CMS 6.0 with applied hotfix 6.0.14 or later.

The following example updates the sample Corporate Site and ensures that it will not have any default CSS stylesheet assigned.

```
PUT http://localhost/CMS/rest/cms.site HTTP/1.1
User-Agent: Fiddler
Authorization: Basic <enter Base64-encoded <username>:<password> here>
Host: localhost
Content-Type: text\xml
Content-Length: 271
```

```
<CMS_Site><SiteDefaultStylesheetID>##null##</SiteDefaultStylesheetID></CMS_Site>
```

7.16.5 Document methods

In this topic, you can find examples of REST methods that can be used to retrieve or manipulate document data from a Kentico CMS instance using the REST service.

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. The URLs 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*. You can also achieve a more detailed specification of retrieved data by appending querystring parameters to the URLs. More information can be found in the URL parameters topic.

- 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.

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, UTF8 is used by default. Please note that this feature is only available in Kentico CMS 6.0 with applied hotfix 6.0.13 or later.

For example:

```
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>
Accept-Charset: utf-8
Host: localhost
Content-Type: text\xml
```

Document manipulation methods

Besides the GET requests listed above, the REST service also supports the PUT, DELETE and POST Http methods. You can update, delete and insert documents (respectively) using these Http methods. Both XML and JSON formats are supported when performing these requests.

The following URLs can be requested by the respective type of request in order to manipulate Kentico CMS documents:

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.

Validation of inserted or updated data

Please note that when inserting or updating document 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.

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>
```

Null value macro

In case that you need to specify an empty value in a document 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.

Please note that this feature is only available in Kentico CMS 6.0 with applied hotfix 6.0.14 or later.

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>
```

7.16.6 URL parameters

This topic contains a list of querystring parameters supported by the REST service. These parameters can be appended to data retrieval request URLs in order to further specify the data to be retrieved. In the brackets, you can find the expected data type of the parameter along with its default value (in **bold** font).

General parameters

This parameter can be used with all REST data retrieval requests in Kentico CMS.

- **Format** (json/atom10/rss20/**xml**)
  Format of the request, if this is specified for PUT/POST requests, this parameter has higher priority than the *Content-Type* parameter in the request header. If you want to use e.g. the JSON format, append *?format=json* to the URL.

Object retrieval parameters

These parameters can be used with object retrieval requests:

- **ObjectData** (**true**/false)
  Indicates whether object data should also be retrieved (set it to false if you want only metadata of the object). If you want the data to be retrieved, append *?objectdata=true* to the URL.

- **MetaData** (true/**false**)
  Indicates whether metadata of the object (type, list of properties / columns) should be attached to the retrieved result. If you want the metadata to be attached, append *?metadata=true* to the URL.

- **Binary** (true/**false**)
  Indicates whether to include binary data in the result. Binary data is retrieved in *Base64* format. If you want the binary data to be included, append *?binary=true* to the URL.

- **Children** (true/**false**)
  Indicates whether to include child objects in the result. If you want child objects to be included, append *?children=true* to the URL.

- **MaxRelativeLevel** (int, **-1 = all levels**)
  If the *Children* parameter is true, this parameters indicates the deepness of the exported object tree structure. If you want to get children from two levels, append *?children=true&maxrelativelevel=2* to the URL.

- **Bindings** (true/**false**)
  Indicates whether to include bound objects in the result. If you want the bound objects to be included, append *?bindings=true* to the URL.

- **OtherBindings** (true/**false**)
  Indicates whether to include other bound objects in the result. If you want other bound objects to be retrieved, append *?otherbindings=true* to the URL.

- **Metafiles** (true/**false**)
  Indicates whether to include metafiles attached to exported object in the result. If you want the metafiles to be included, append *?metafiles=true* to the URL.

- **Relationships** (true/**false**)
  Indicates whether to include object relationships in the result. If you want the relationships to be included, append *?relationships=true* to the URL.

- **Categories** (true/**false**)
  Indicates whether to include the object's category structure in the result if the object is categorized. If you want the structure to be included, append *?categories=true* to the URL.

- **Translations** (true/**false**)
  Indicates whether to include a translation table of foreign keys in the result. If you want it to be included, append *?translations=true* to the URL.

- **Hierarchy** (true/**false**)
  If true, the result is exported in a hierarchical structure (if false, the *children – bindings – parent* structure is flat). If you want data to be exported in the hierarchical structure, append *?hierarchy=true*

to the URL.

Multiple object retrieval parameters

These parameters can be applied when a whole data set of objects should be retrieved. The dataset can be filtered and ordered using standard parameters in Kentico CMS.

- **Where** (string, **empty by default**)
  WHERE condition to filter the dataset. If you want to filter retrieved data using a WHERE condition, append ?where=<text of the condition> to the URL.

- **OrderBy** (string, **empty by default**)
  ORDER BY clause to order the data. You can use ?orderby=##default## for ordering according to object display name.

- **Columns** (string, **all columns by default**)
  Columns of the object to retrieve. If you want e.g. to have the UserName and UserID columns retrieved when retrieving users, append *?columns=UserName,UserID* to the URL. If this parameter is used, the *Binary* parameter is not reflected (you can choose whether to include binary columns in the result by enumerating the respective columns).

- **TopN** (int, **all records by default**)
  TOP N clause to filter the data. If you want top 10 records to be retrieved, append *?topn=10* to the URL.

- **Offset** (int, **first record by default**)
  Parameter used for paging of the data. Enter the number of the record which should be the first. If you want data to be retrieved from the third result, append *?offset=2* to the URL.

- **MaxRecords** (int, **all record by default**)
  Maximal number of records to retrieve (used for paging in combination with the *Offset* parameter). If you want e.g. 10 records to be retrieved, append *?maxrecords=10* to the URL.

Document retrieval parameters

Here is the list of supported parameters for all document retrieval methods (except for these you can also use standard filtering and ordering parameters from previous section).

- **ClassNames** (string, **all classnames**)
  List of classNames to select separated by semicolon. If you want to retrieve only *cms.article* documents, append *?classnames=cms.article* to the URL.

- **CombineWithDefaultCulture** (**true**/false)
  Specifies if return the default culture document when specified culture not found. If you want documents to be retrieved from the default culture when not found in the specified one, append *?combinewithdefaultculture=true* to the URL.

- **SelectOnlyPublished** (**true**/false)
  Select only published nodes. If you want unpublished nodes to be retrieved as well, append *?selectonlypublished=false* to the URL.

- **Version** (**published**/last)

Determines which version of the document should be retrieved. Published (version you can see on live site) or last version (version which is being edited). If you want the last document version to be retrieved, append *?version=last* to the URL.

- **CoupledData** (**true**/false)
  If false, coupled data is not included in the result. If you don't want coupled data to be included in the result, append *?coupleddata=false* to the URL.

Document deletion parameters

- **DeleteAllCultures** (true/**false**)
  Indicates if all cultural versions of the specified document should also be deleted by the request. If you want all cultural versions to be deleted, append *?deleteallcultures=true* to the URL.

- **DestroyHistory** (true/**false**)
  Indicates if document version history should also be deleted by the request. If you want version history to be deleted together with a document, append *?destroyhistory=true* to the URL.

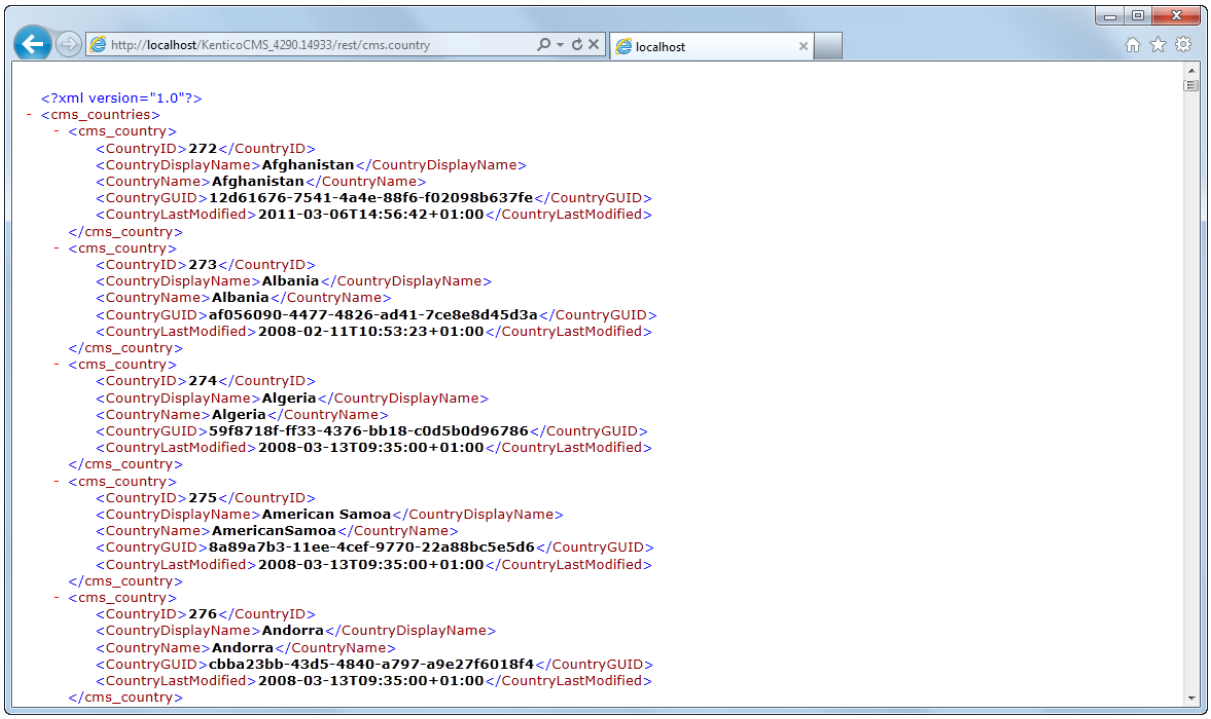
- **DeleteProduct** (true/**false**)
  Indicates if bound E-commerce product should be deleted together with the specified product document. If you want E-commerce product to be deleted together with a document, append *?deleteproduct=true* to the URL.

7.16.7 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

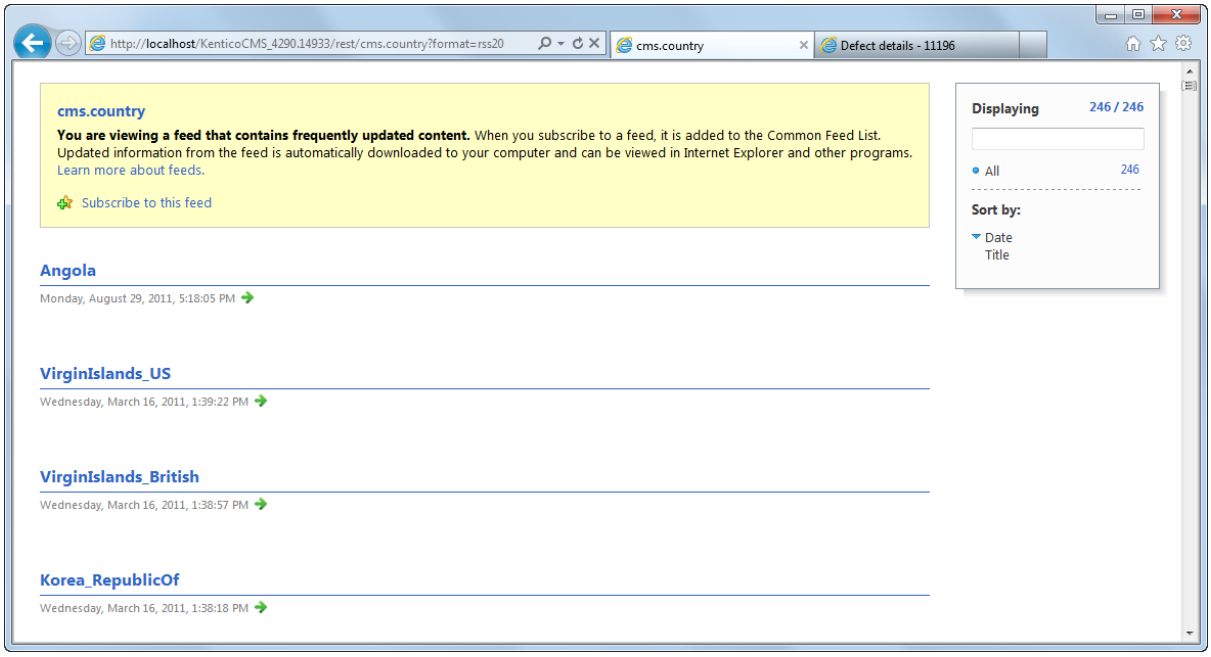
The code below is an extract from data of the *cms.country* objects retrieved in JSON format. This result was obtained using the following URL: **~/rest/cms.country?format=json**

```
{"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"}
]
}
]
}
```

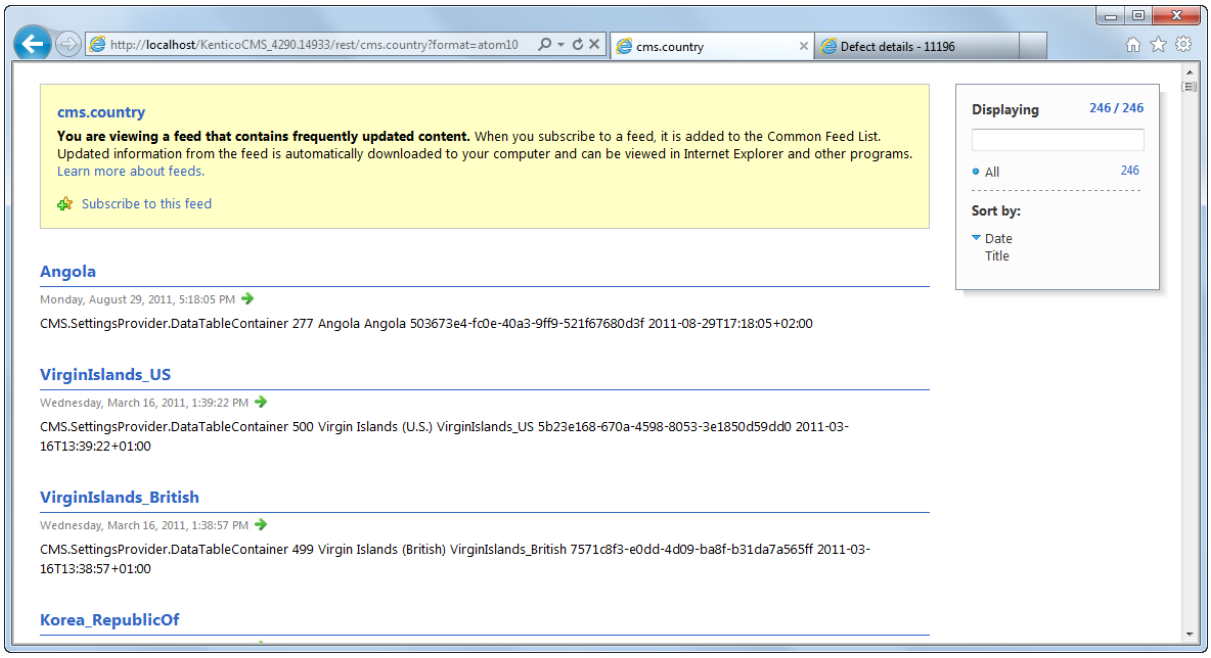
RSS 2.0

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**

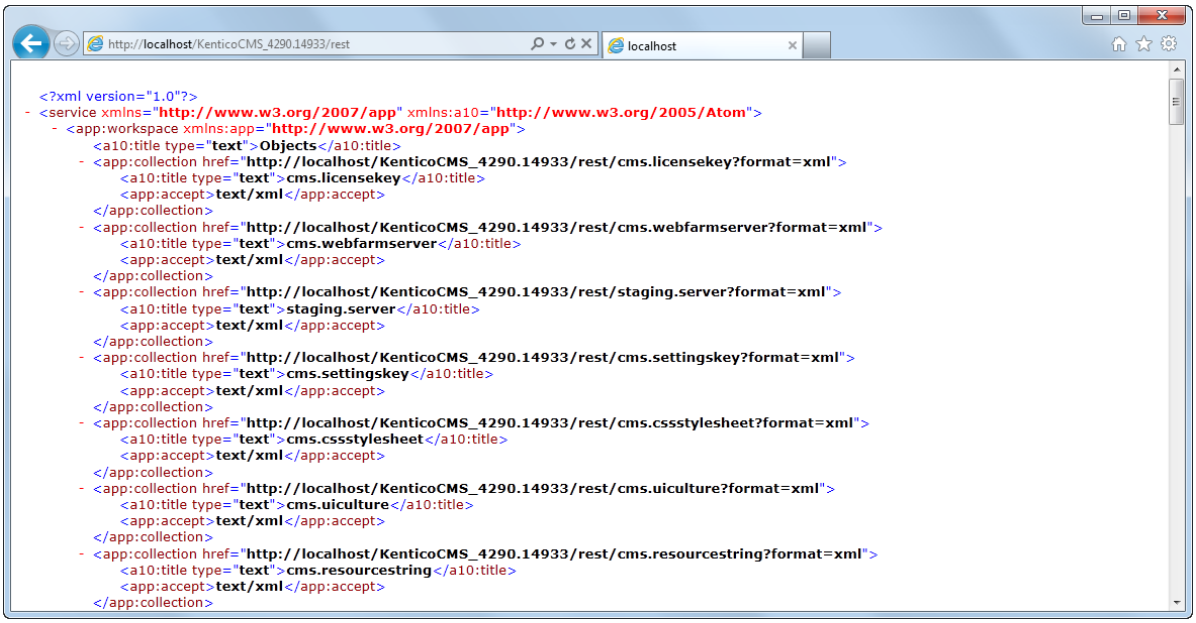


7.16.8 ODATA service documents

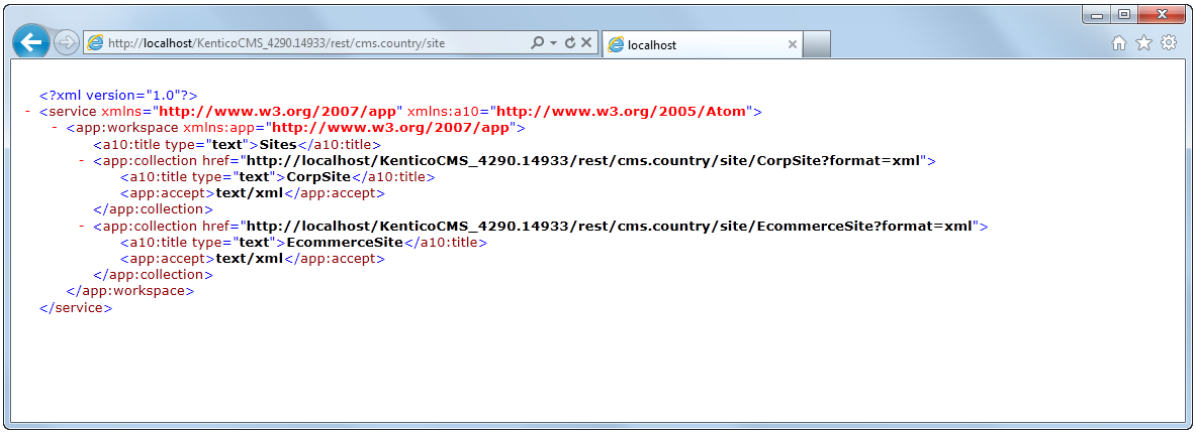
Kentico CMS REST service supports ODATA by providing service documents in a standard format for ODATA services. The service documents provide information about the types of data available via the REST service and about the URLs under which the data can be accessed.

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.



7.17 Scheduler

7.17.1 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.

7.17.2 Configuring task execution

This topic explains how to configure execution of scheduled tasks.

Configuring execution by 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 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 tasks 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 want to execute them 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. The values are entered in seconds and the checks and execution can be completely disabled by setting the value to 0. The minimum possible value is 30 seconds.

By default, the application performs the check at the end of standard page requests. This means that tasks are executed only when user activity on your website is generating requests. Because of this, the **Application scheduler interval** setting actually sets the minimum interval between these checks.

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** setting sets the precise interval between these checks. This can be done 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 is ensured and less application load generated. The values are entered in seconds and the checks and execution can be completely disabled by setting the value to 0.

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.

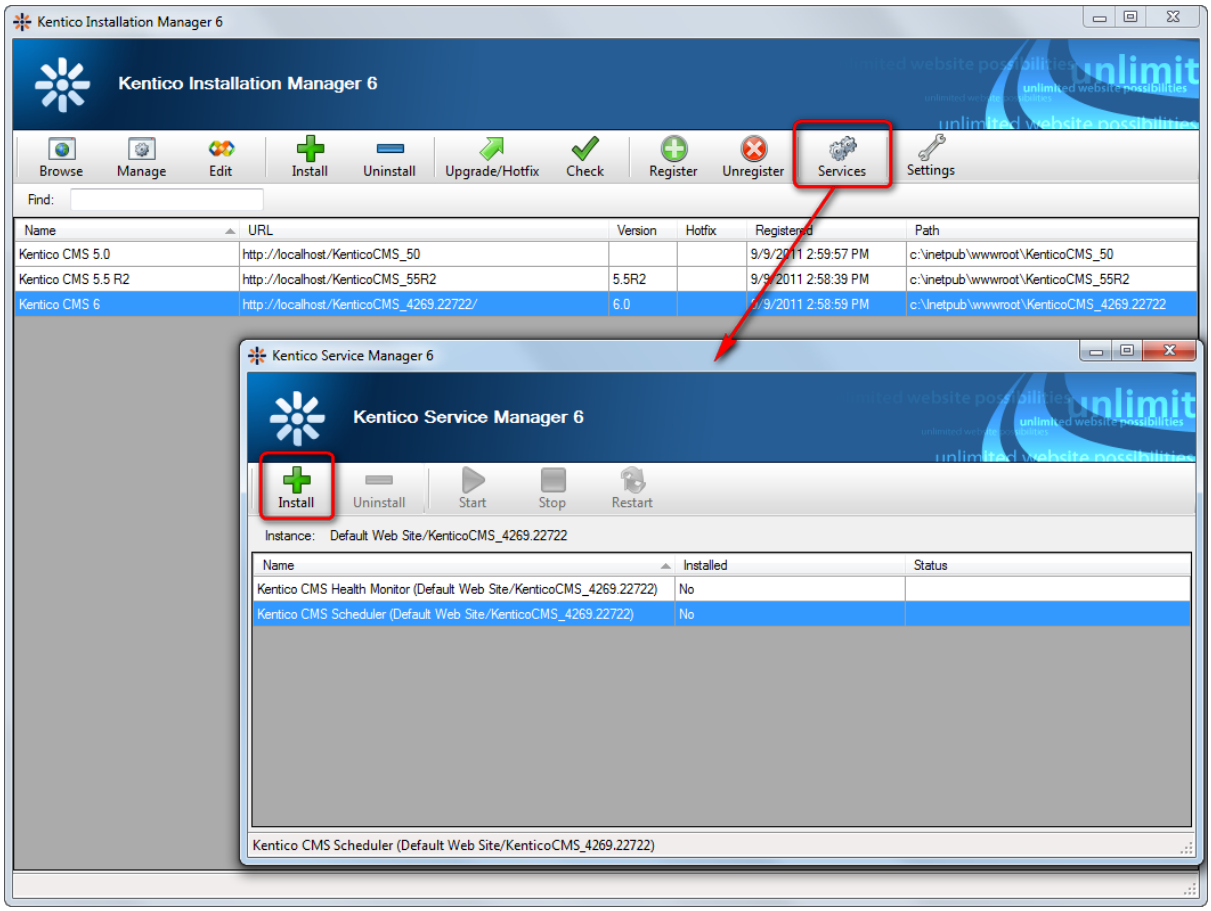
7.17.3 Installing the Scheduler Windows service

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.

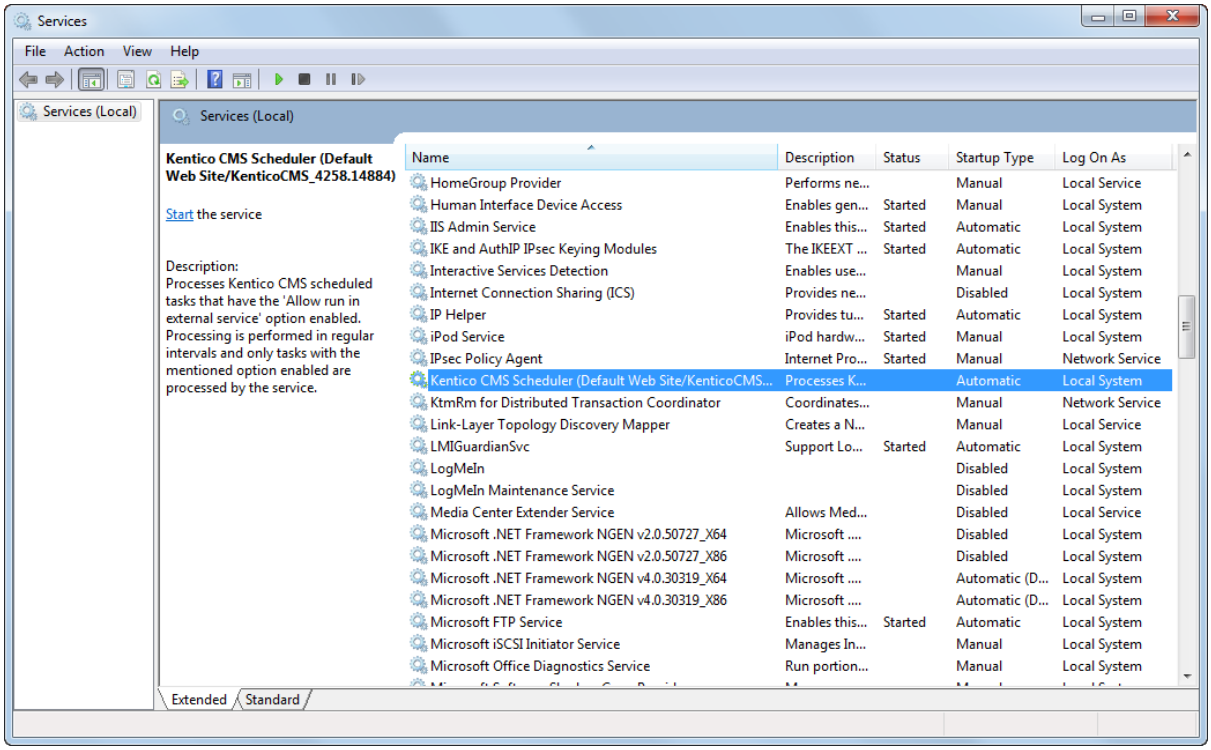
1. Open Windows command line (type *cmd* in the Start menu search box) and navigate to the .NET folder containing *InstallUtil.exe* (e.g. *c:\Windows\Microsoft.NET\Framework64\v4.0.30319*).

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

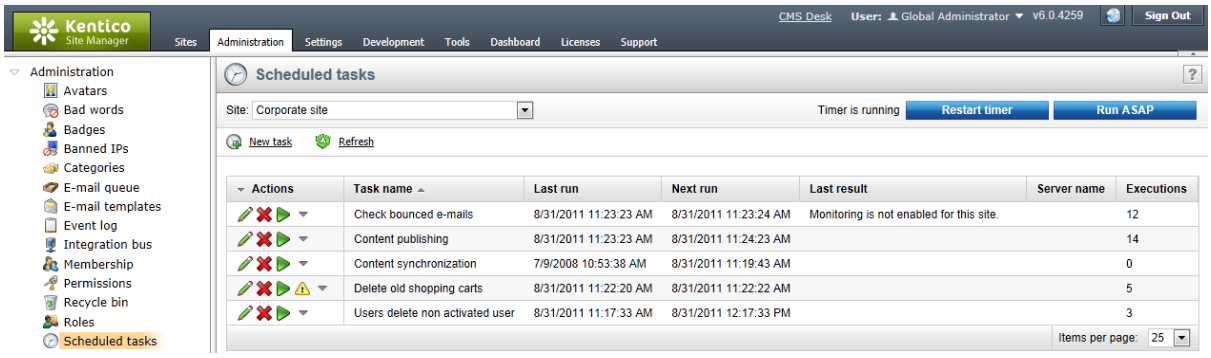
Uninstalling the Scheduler Windows service can be performed exactly the same way as described above, while you only need to use the **/u** parameter instead of **/i** at the end of the command:

```
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.

7.17.4 Scheduled tasks administration

You can configure scheduled tasks in **Site Manager -> Administration -> Scheduled tasks**.



The **Site** drop-down list is used for selecting a site. After a site is selected, a list of tasks scheduled for the given site will be displayed in the table below. Tasks that are not site-specific can only be viewed when **(Global)** is selected. 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:

- **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.

The **New task** link allows you to schedule new custom tasks. The process of creating a scheduled task is covered in more detail in the Scheduling a custom code topic. The **Refresh** action updates the information displayed below in the list of tasks.

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.

The icon is displayed in the **Actions column** 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 for these task to be executed.

7.17.5 Scheduling custom code

The process of scheduling a custom task includes two steps: Writing the code that performs the required actions and registering it in the CMS as a scheduled task.

Writing task code

Each scheduled task must be defined by a class that implements the **CMS.Scheduler.ITask** interface.

To integrate this type of class into the application, you can add a new assembly (Class library) to your project and include it there. In this case, it is necessary to add the appropriate references to both the assembly and the main CMS project.

Alternatively, you can define scheduled tasks in *App_Code* without the need to compile an assembly. This approach is described in the example below:

1. Open your web project, expand the **App_Code** folder (or **Old_App_Code** if you installed Kentico CMS as a web application), navigate to the **Samples\Classes** folder and edit the **MyTask.cs** file. This file already contains a sample class that follows the basic structure required for a scheduled task. Modify the code according to the following:

[C#]

```
namespace Custom
{
    /// <summary>
    /// Sample task class.
    /// </summary>
    public class MyTask : ITask
    {
        /// <summary>
        /// Executes the task.
        /// </summary>
        /// <param name="ti">Task info</param>
        public string Execute(TaskInfo ti)
        {
            string detail = "Executed from '~/App_Code/Samples/Classes/MyTask.cs'.
Task data:"
                            + ti.TaskData;

            // Logs the execution of the task in the event log.
            EventLogProvider.LogInformation("MyTask", "Execute", detail);

            return null;
        }
    }
}
```

The **Execute** method must always be included when writing a scheduled task. It is called whenever the given task is executed, so it must 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 that it is being executed. The **TaskInfo** parameter of the method can be used to access the data fields of the corresponding scheduled task object. As you can see in the code above, the content of the **TaskData** field is added into the details of the event log entry.

The string returned by the method will be displayed in the administration interface when the task is executed and can be used to describe the result. You can leave it as *null* in this case.

Save the changes made to the file.

2. Next, it is necessary to ensure that the **MyTask** class is loaded when the given scheduled task is executed. Expand the **Samples\Modules** folder and open the **SampleClassLoaderModule.cs** file, which demonstrates how this can be done. You do not have to make any modifications for the purposes of this example, since the sample class loader handles the *Custom.MyTask* class by default.

An object of the appropriate class is assigned by the **ClassHelper_OnGetCustomClass** event handler:

[C#]

```
/// <summary>
/// Gets a custom class object based on the given parameters.
/// </summary>
private void ClassHelper_OnGetCustomClass(object sender, ClassEventArgs e)
{
    if (e.Object == null)
    {
        // Passes on an object of the appropriate custom class.
        switch (e.ClassName)
        {
            // Gets an object of the MyTask class implementing the ITask
interface.
            case "Custom.MyTask":
                e.Object = new Custom.MyTask();
                break;

            ...

        }
    }
}
```

In the case of scheduled tasks, the value of the **ClassName** property of the handler's *ClassEventArgs* parameter will match the **Task class name** specified for the given task (*Custom.MyTask* in this example). The value is checked to identify which specific task was executed, so that an instance of the correct class can be created and passed on.

Build the project if you installed it as a web application.

Creating a new scheduled task

Go to **Site Manager -> Administration -> Scheduled tasks** and select the **Site** for which the task should be scheduled (or *(global)* if it should be performed for all sites or affect global objects). Next, click the **New task** link and fill in the properties of the task according to the following:

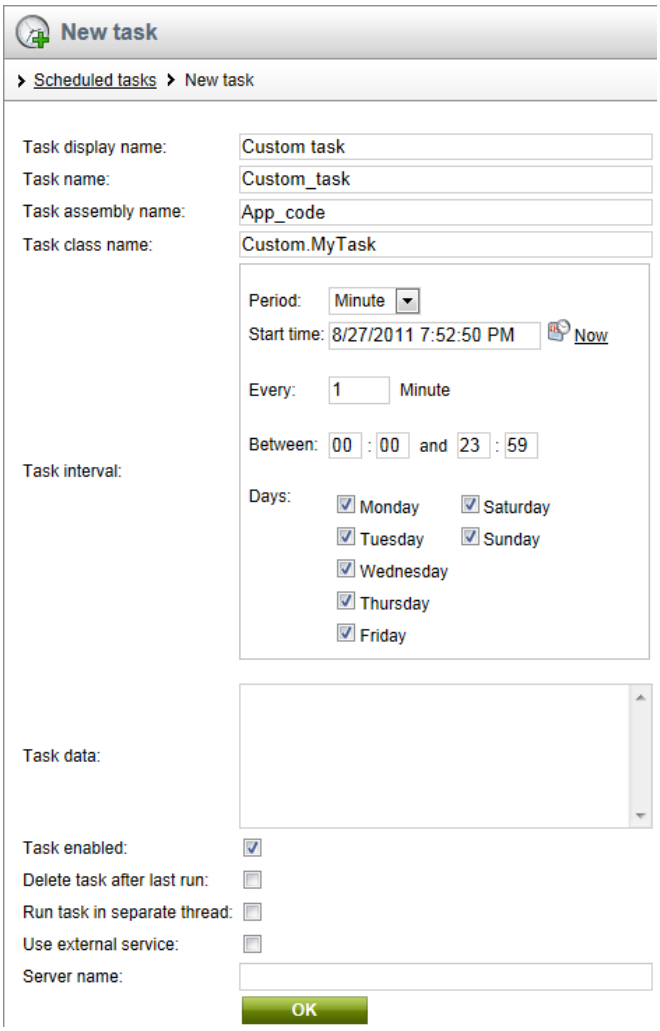
- **Task display name** - *Custom task*; sets a name for the task that will be shown in the administration interface.
- **Task name** - *Custom_task*; sets a name that will serve as an identifier for the scheduled task.
- **Task assembly name**: *App_Code*; this field must contain the name of the assembly where the task is implemented.
- **Task class name**: *Custom.MyTask*; specifies the exact class (including any namespaces) that defines the functionality of the scheduled task.
- **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 to be executed. The precise execution time depends on the general settings of the scheduler.
- **Task data** - data which should be provided to the assembly. This field can be accessed from the code of the task, so it can be used as a parameter to easily modify the task without having to edit its implementation. In this example, any entered content will be included in the details of the event log entry created by the task.
- **Task enabled**: *yes (checked)*; this field indicates if the task should be scheduled.
- **Delete task after last run** - indicates if the task should be deleted after its final run (applicable if the task is set to run only once).

- **Run task in separate thread** - indicates if the task should be executed in a separate thread in order to improve application performance.
- **Use external service** - indicates if the task should be processed by the Scheduler Windows service instead of the web 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. The option is available only for some scheduled tasks. The ones that do not have it available in their editing interfaces can only be processed by the application itself.

Important!

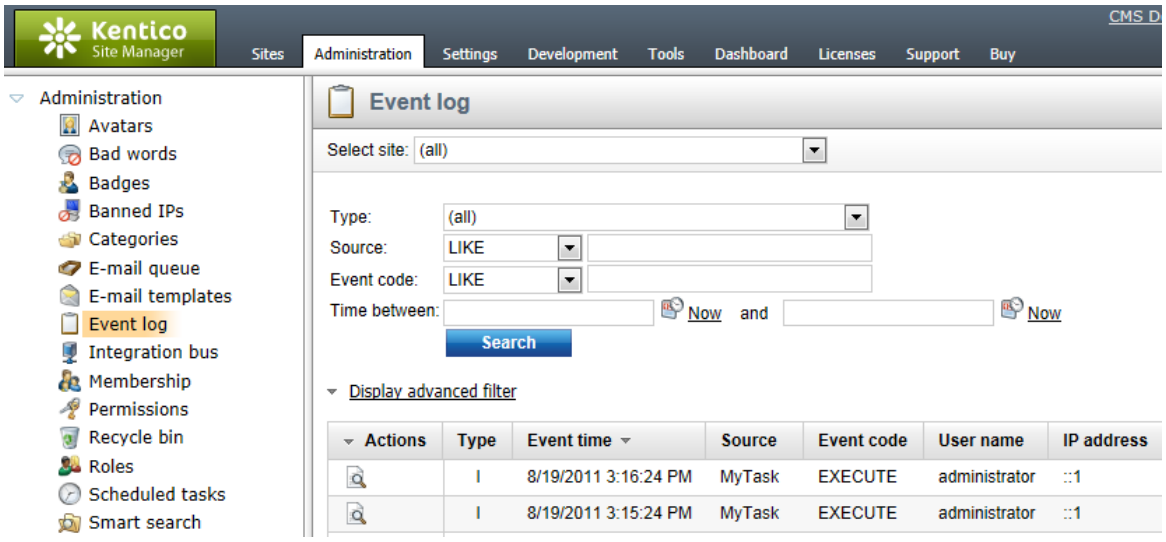
Custom tasks implemented in the *App_Code* folder cannot be executed by the external Windows service. If you wish to run a custom task this way, it is necessary to add a new assembly to your project and then define the task class there.

- **Server name** - name of the web farm server where the task should be executed. This field is applicable only if your application runs in a web farm environment.
- **Create tasks for all web farm servers** - if checked, tasks will be created for all web farm servers and the **Server name** field will be grayed out. This field is displayed only if your application runs in a web farm environment.



Click **OK**. The task will now be executed regularly according to the specified interval.

To check the results once the task is enabled, go to **Site Manager -> Administration -> Event log** and look for entries with *MyTask* as their **Source**.



7.17.6 Scheduler internals and API

7.17.6.1 Overview

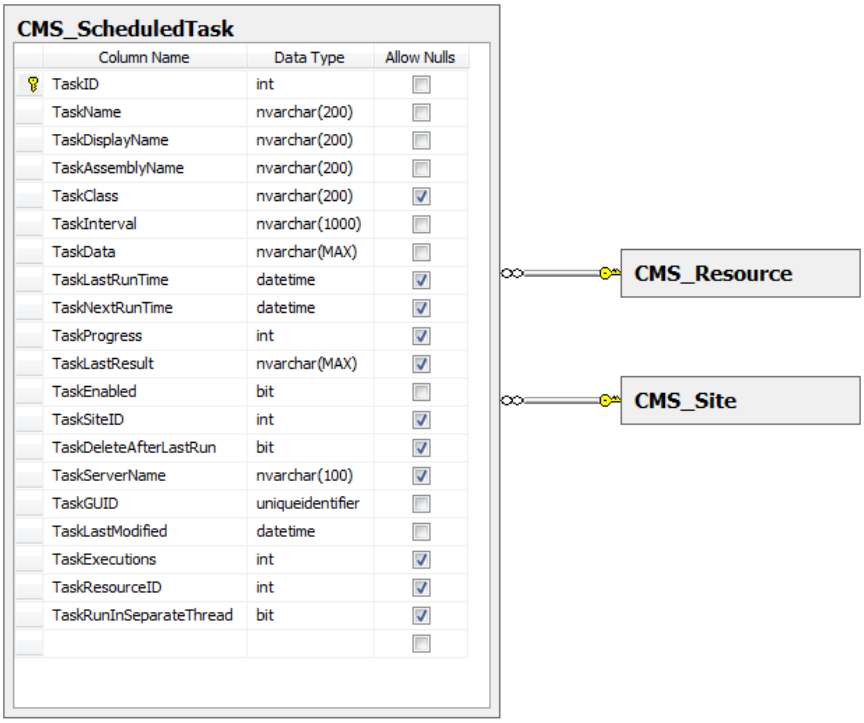
In this chapter, you will learn which database tables and API classes are used by the Scheduler. You will also see the most common API examples.

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.

7.17.6.2 Database tables

The following database tables are used to store scheduler-related information:

- **CMS_ScheduledTask** - contains records representing scheduled tasks and their settings.



7.17.6.3 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 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.

7.17.6.4 API examples

7.17.6.4.1 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;
```

7.17.6.4.2 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.

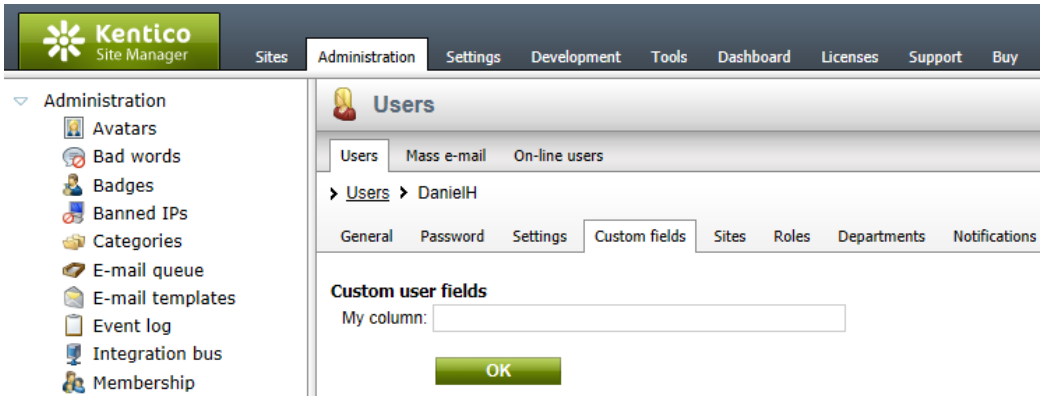
```
private bool RunTask()
{
    // Get the scheduled task
    TaskInfo runTask = TaskInfoProvider.GetTaskInfo("MyNewTask", CMSContext.
CurrentSiteID);

    if (runTask != null)
    {
        // Run task
        SchedulingExecutor.ExecuteTask(runTask);

        return true;
    }

    return false;
}
```

7.18 System tables and custom fields

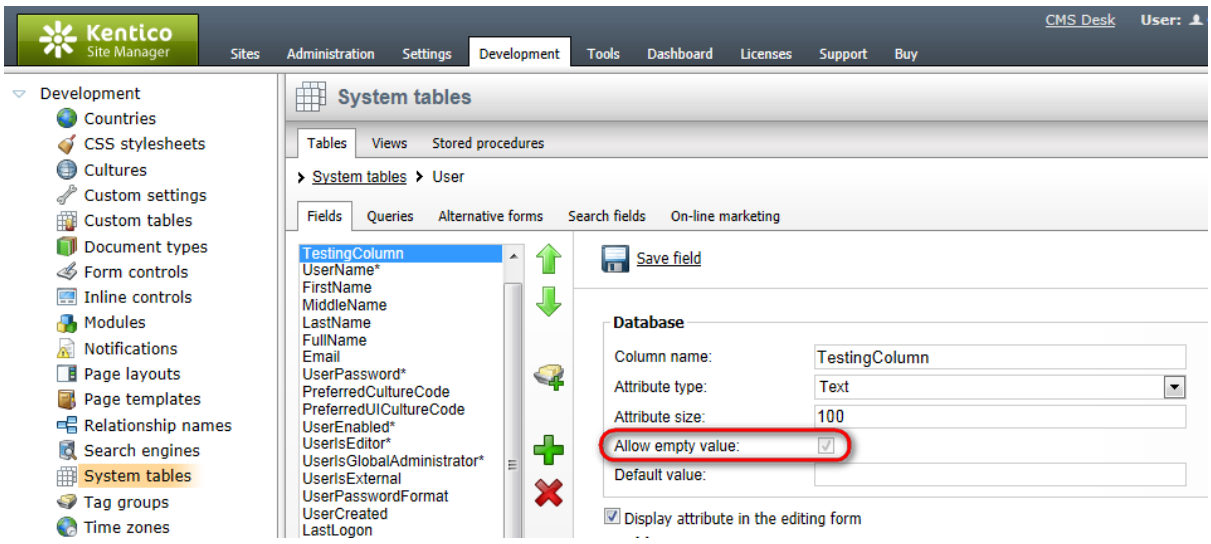
7.18.1 Overview

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. However, you can change it to *FALSE* 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.

7.18.2 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.

7.19 UI cultures and localization

7.19.1 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

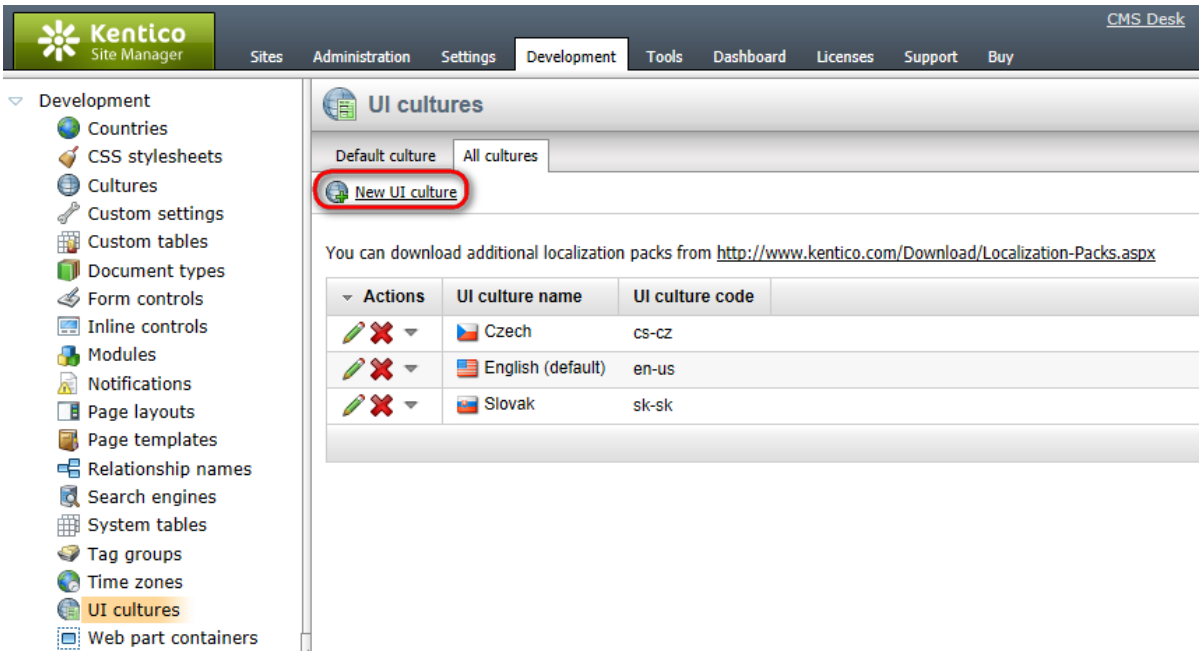
Content of Kentico CMS websites (i.e. documents in the content tree) can also be multilingual. To learn about localization of website content, please refer to the Content management -> Multilingual content chapter of this guide.

7.19.2 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

If you want the administration interface to be displayed in a different language or at least with different culture settings (e.g. calendar and numeric format), go to **Site Manager -> Development -> UI cultures -> All cultures** tab and add a new UI culture using the **New UI Culture** link.

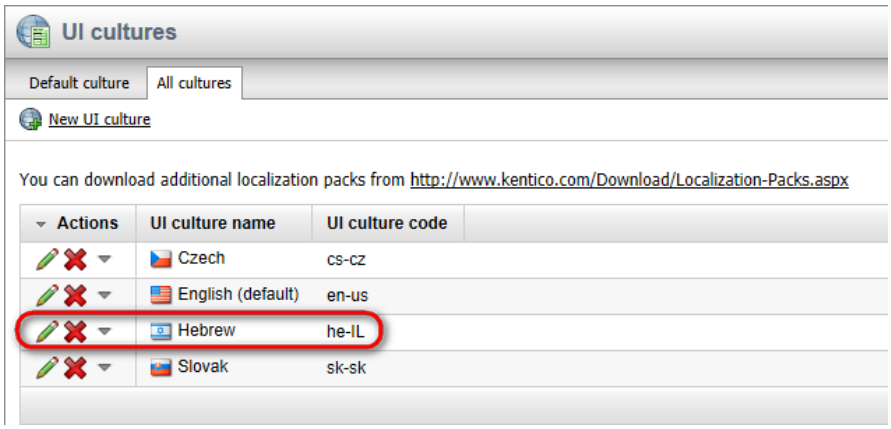


The following properties will have to be entered:

- **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 culture you have just created will be shown in the list.



Then 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). Now 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: when you make changes to a *.resx* file, you need to restart the web application using the **Site Manager -> Administration -> System -> Restart application** button so that the changes are updated in the user interface.

Modifying standard 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.

How to add 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, please take the following steps:

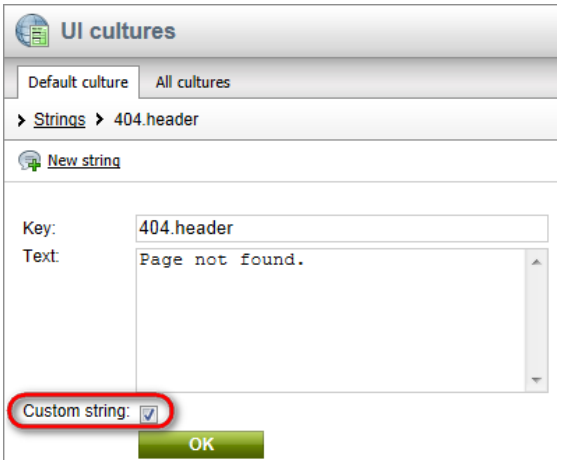
1. Create a custom default culture string in **Site Manager -> Development -> UI cultures -> Default culture** tab in the default culture language using the **New string** link.

The following properties will have to be entered:

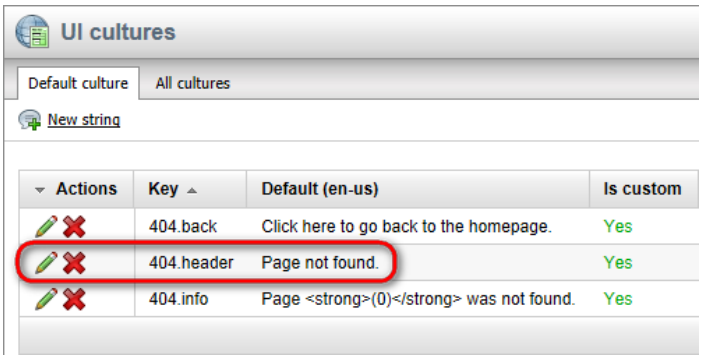
- **Key**: 404.header (example - you can use any other key name)

- **Text**: Page not found. (example - you can use any other text)

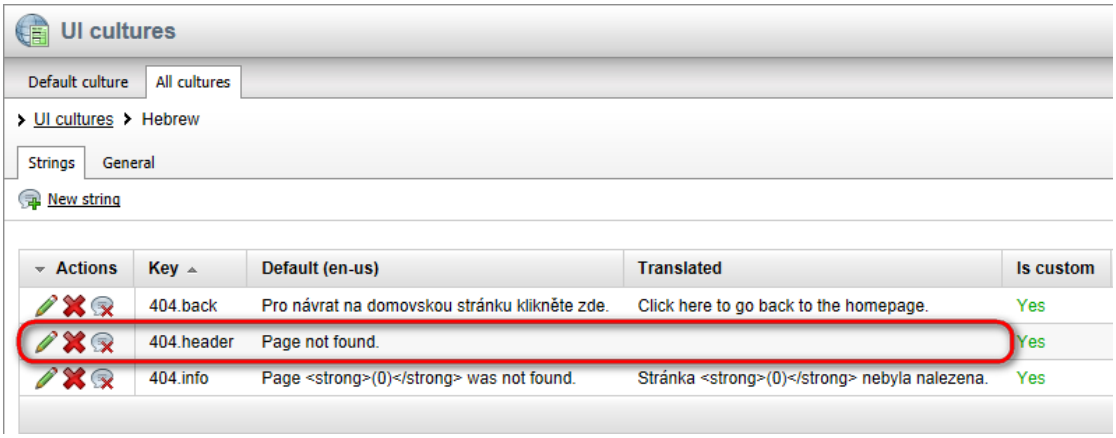
Please be sure to check the **Custom string** check box in this case, so that the string will automatically be exported with your website.



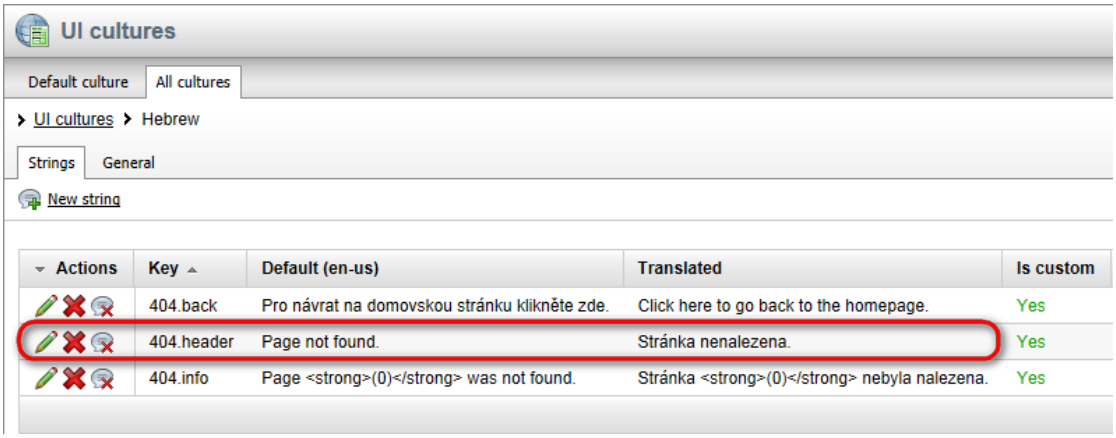
Click **OK**. A new default culture string will now be displayed in the list.



2. 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.



Choose to **Edit** ( ) the custom string created in step 1. and translate it into the desired language using the **Text** field of the string. Please do not forget to check the **Custom string** check box and click **OK**. When you to edit this particular UI culture again, the string will now be 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 the **New string** link on this tab.

3. The resource string and its translation are now created and will be 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.

Priority of the resource strings

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:

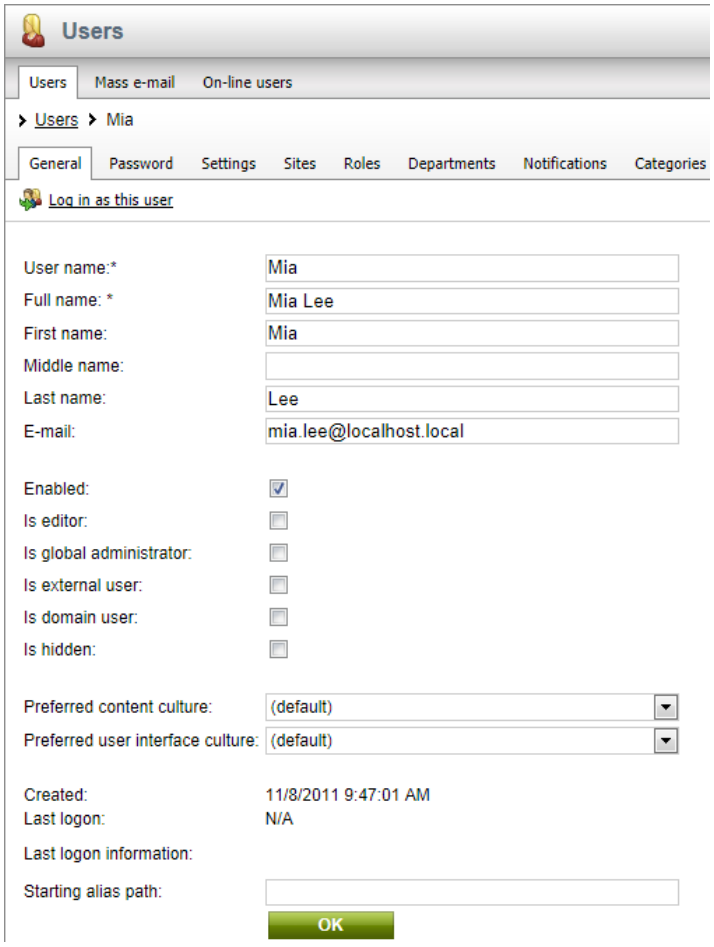
```
<add key="CMSUseSQLResourceManagerAsPrimary" value="false" />
```

When this key is added, the priorities are as follows:

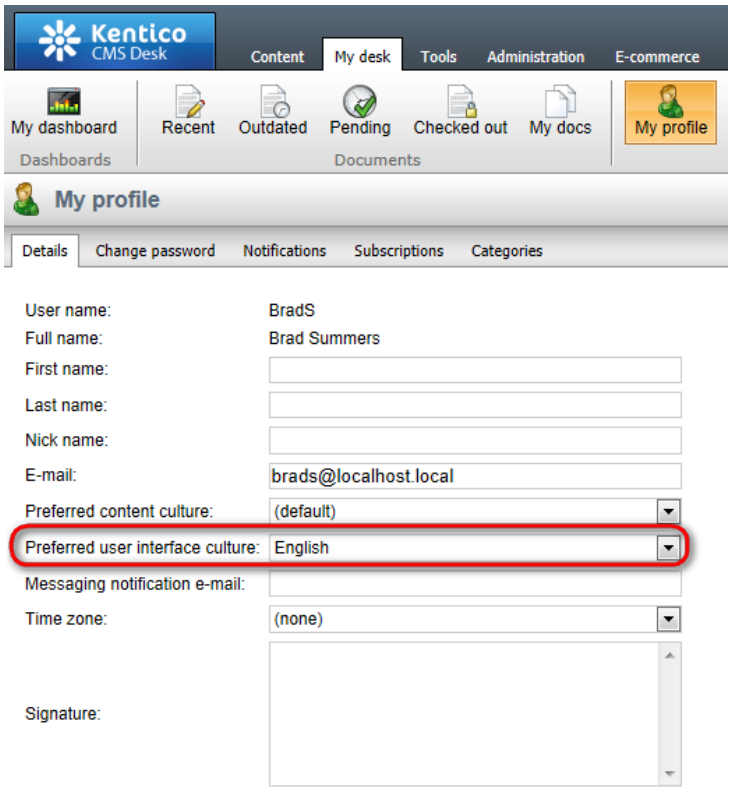
1. custom.resx
2. cms.resx
3. database (Site Manager -> Development -> UI Cultures)

Applying a culture to the user interface

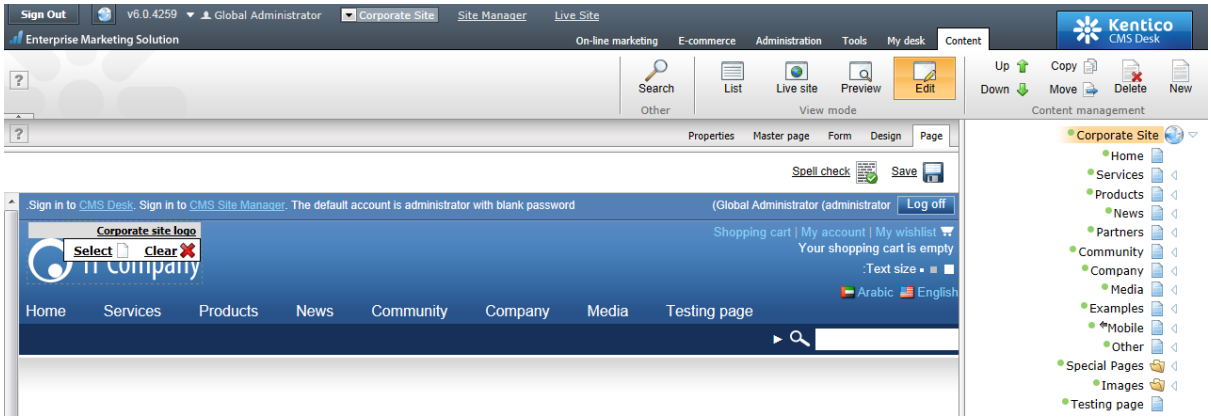
The user interface can be set to a specific culture for each user in the system. To do this, go to **CMS Desk / Site Manager -> Administration -> Users**, edit (✎) the given user and select the required value from the **Preferred user interface culture** drop-down list on the **General** tab.



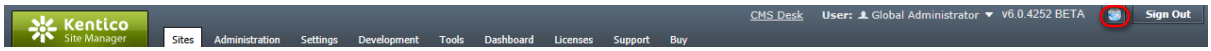
Users may also select their own user interface culture by going to **My Desk -> Account -> Details** and setting the **Preferred user interface culture**.



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.

7.19.3 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.

7.19.4 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.

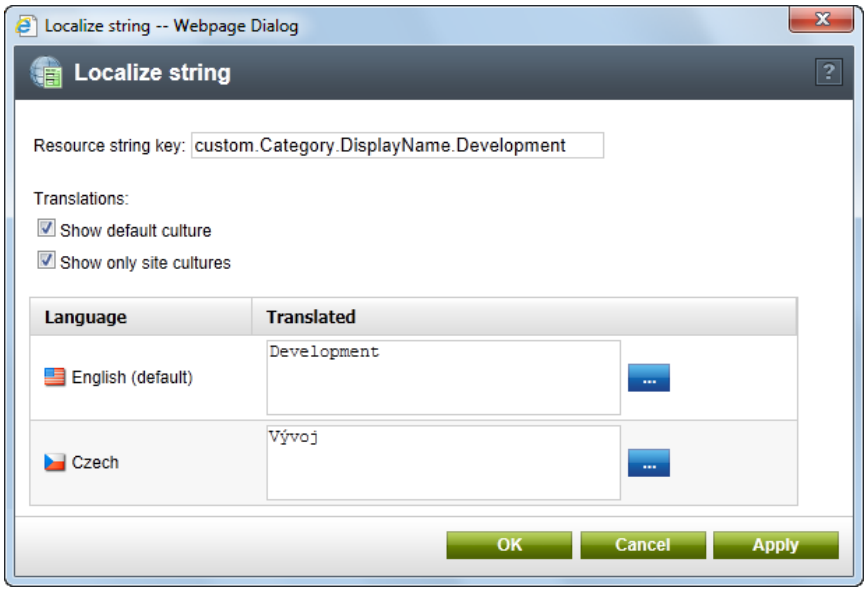
Please note that these actions are only available for websites that are multilingual, i.e. those that have more than one culture assigned in **Site Manager -> Sites -> edit (**) **site -> Cultures**. Because this feature provides a way to edit any resource string, it is only allowed for users designated as *global administrators*.

To add a resource string to a field where this functionality is supported, click the **Localize** () button.

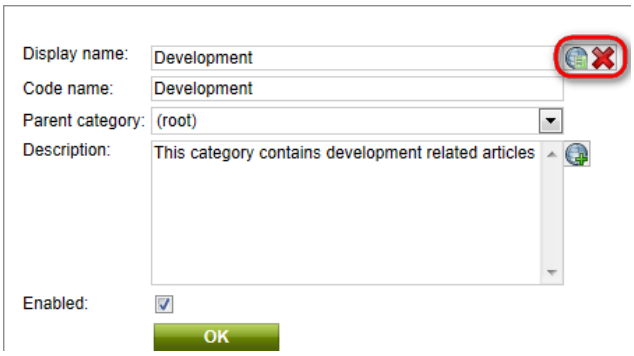
This opens a dialog where a new string can be created, or an existing one can be selected for the field.

Next, the text of the resource string may be edited for specific cultures. 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 that are assigned to the currently active site will be 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 will also affect all other occurrences of the same string.



When finished, click **OK** to confirm any changes to the resource string.



It is necessary to save the form using the **OK** button to confirm that the string should be inserted into the field (some forms may use a **Save** button at the top of the page instead). The field will now be localized and its value will be displayed according to the current culture settings.

The text of the resource string can be edited at any time by using 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.

7.20 Web parts

7.20.1 Overview

Web parts represent a block of content or combination of content and functionality. They are the basic building blocks of portal engine page templates. 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 by using the **CMS Desk -> Content -> Edit -> Design** interface. All web parts must be placed into web part zones.

From a developer's point of view, a web part is a user control (ASCX) that inherits from the **CMSPortalControls.CMSAbstractWebPart** class. You can easily create your own web parts as described in the Developing web parts topic.

The Kentico CMS installation contains many built-in web parts, but sometimes it may be necessary to modify the behavior, design or functionality of one of them. The Modifying web parts sub-chapter covers several ways how this can be done.

All web parts have the option of using an AJAX UpdatePanel. More information can be found in the AJAX support topic.

Also of interest may be Widgets, which provide functionality similar to web parts, but allow further personalization by users.

Web part management

Web parts can be managed in **Site Manager -> Development -> Web parts**. Here you can see all web parts organized into categories in a tree structure. Each web part has the following properties on the **General** tab:

- **Display name** - name of the web part displayed in the administration interface.
- **Code name** - name of the web part used in website code.
- **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 behaviour 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 a connected web part.
  - **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.

- **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 the web parts on the disk in the same way as in the categories. Example: *Search/cmscompletesearchdialog.ascx*
- **Description** - a text description of the web part that will be displayed in the web part catalog.
- **Thumbnail** - image used in the web part catalog.

On the **Properties** tab, you can define the web part properties and how they appear in the **Web part properties** configuration dialog.

Each web part has 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 attributes (e.g. for it not to be displayed in the web part configuration dialog) you can do so by adding and configuring it on the **Properties** tab.

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. If empty, the value of the *Web part control ID* property is used for this purpose.
- **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.

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 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. 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.

AJAX

- **Use update panel** - indicates if an AJAX UpdatePanel container should be used for the web part.

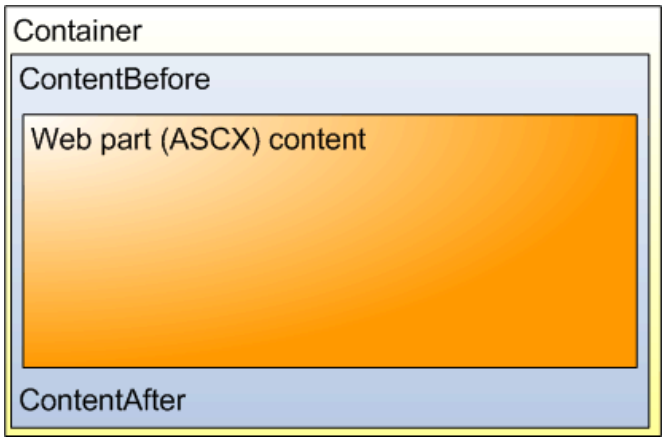
Time zones

- **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.

Partial caching

- **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.

The structure of a web part, its content before/after sections and a Web part container is the following:



The containers, unlike the *ContentBefore* and *ContentAfter* sections, are re-usable and they can contain dynamically inserted values of web part properties.

Storing files related to web parts

If your web part consists of several files (such as 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 exported/imported correctly when you move your website or when you distribute the web part to other developers.

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 particular properties, you need to fill in the **Field description** on the **Properties** tab.

For complete documentation of all web parts, please see the Kentico CMS Web Parts reference.

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.

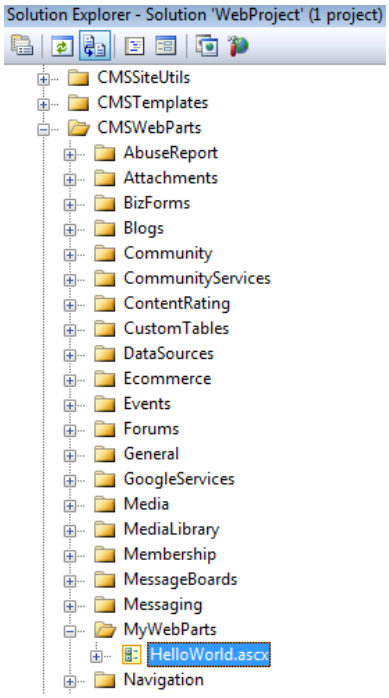
7.20.2 Developing web parts

This topic 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, it updates the current time displayed by the label.

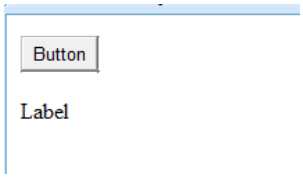
1. Open the web project in Visual Studio (or Visual Web Developer) using the **WebProject.sln** (or **WebApp.sln**) file or using **File -> Open -> Web site...** in Visual Studio.

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**. Choose to create a new **Web User Control** and call it **HelloWorld.ascx**.



4. Display the **HelloWorld** control on the Design tab. Drag and drop a new **Button** control and a new **Label** control onto the form:



5. 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()
```

6. Add the following line to the beginning of the code:

[C#]

```
using CMS.PortalControls;
```

[VB.NET]

```
Imports CMS.PortalControls
```

7. 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.

8. 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
```

It sets the button text to the value configured in Kentico CMS Desk.

9. Save all changes. If your Kentico CMS project was installed as a web application, you must **Build** the project.

10. Open **Site Manager -> Development -> Web parts**, click the root and click **New category**. Enter *My web parts* into the **Category display name** field, *MyWebParts* into the **Category name** field and click **OK**.

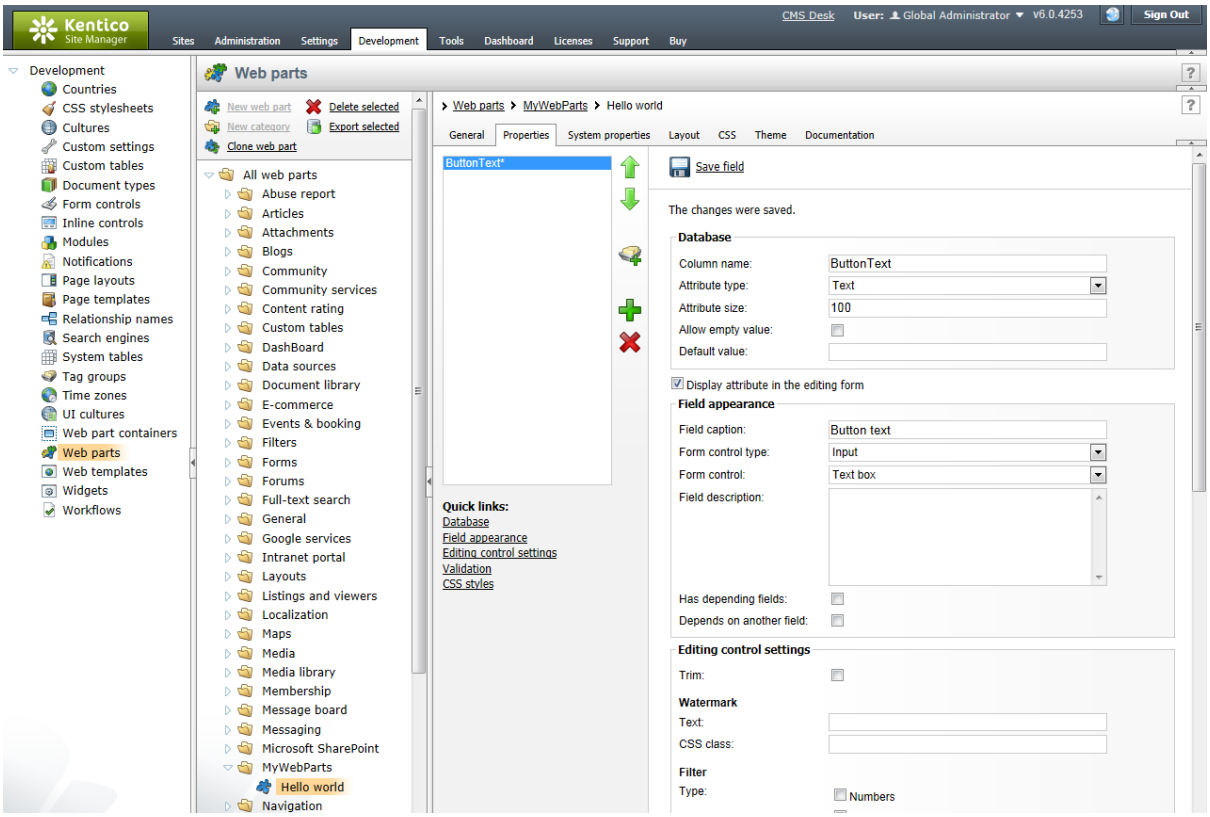
11. Select the new category and click **New web part**. Choose to **create a new web part** and enter the following values:

- **Display name**: Hello world
- **Code name**: HelloWorld
- **File name**: ~/CMSWebParts/MyWebParts/HelloWorld.ascx

Click **OK**.

12. 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 type**: Input
- **Form control**: Text box

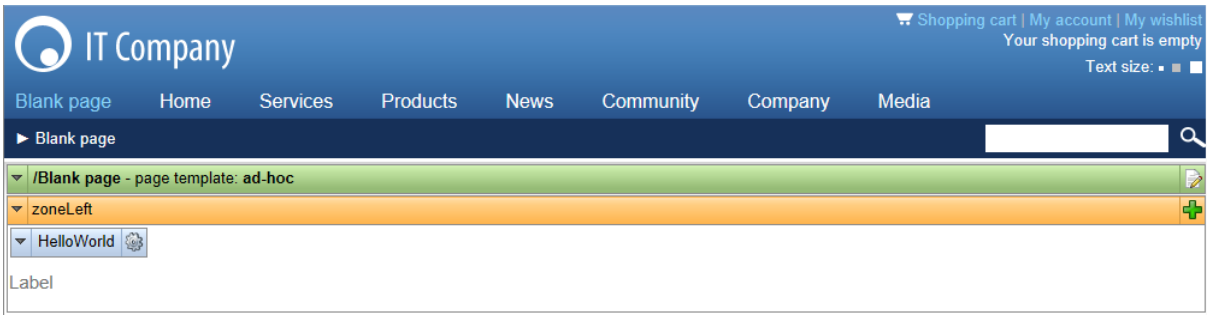


Click **Save field**.

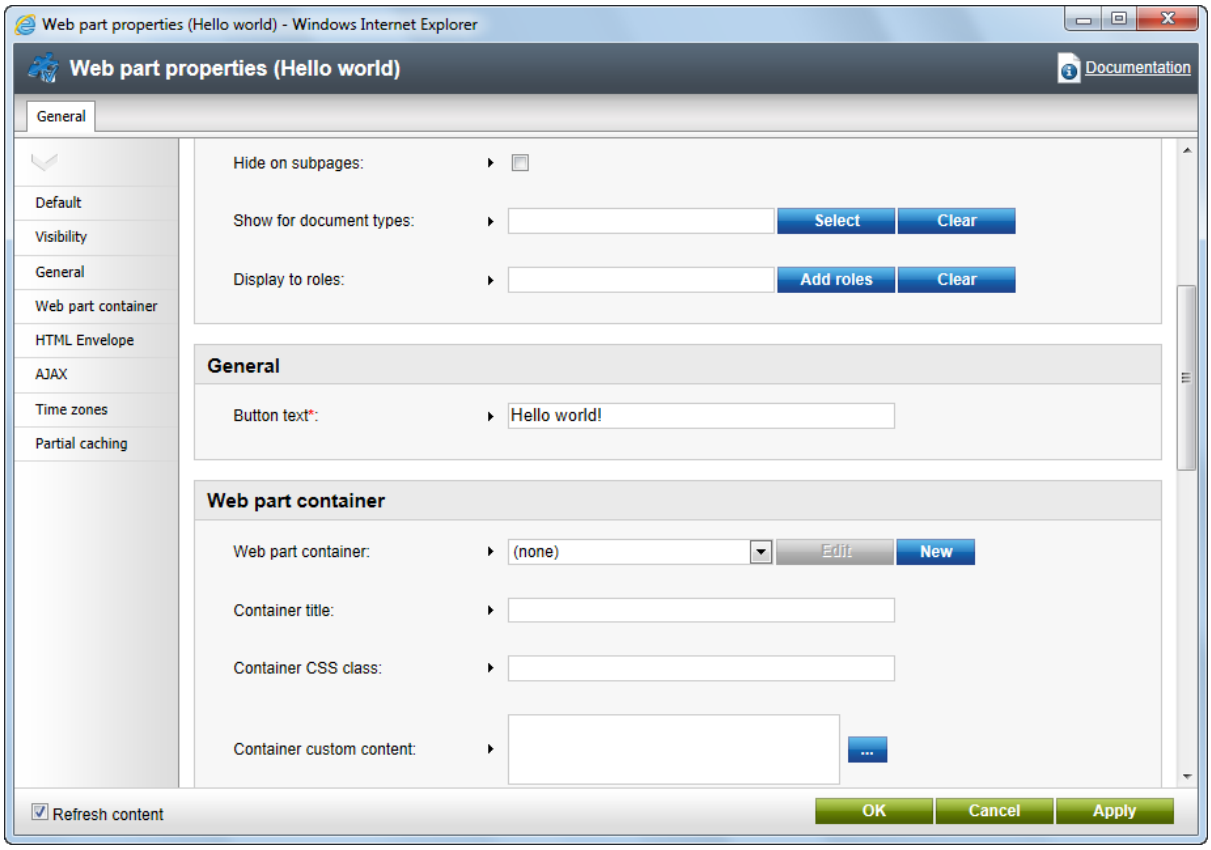
13. Switch to **CMS Desk**.

14. Create a new **blank page** using the **Simple** layout (or any other layout) under the root and switch to the **Design** tab.

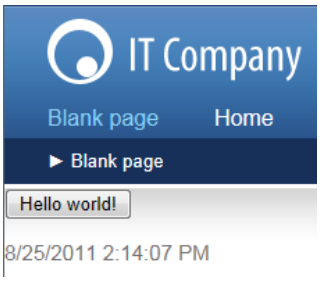
15. Click **Add web part** (+) in the upper right corner of the web part zone and choose to add the **Hello world** web part:



16. The Web part properties dialog of the **HelloWorld** web part will be displayed. Set the value of the **Button text** field to *Hello world!*



17. 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.

> **Tip: Displaying content in the web part**
>
> You can also use Kentico CMS Controls in the web part (in the ASCX control) to display content from Kentico CMS in a customized form.

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;
    }
}
```

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 the controls' 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 classic .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 making a control from the web part and loading it dynamically (e.g. using the **General -> User control** web part).

7.20.3 Modifying web parts

7.20.3.1 Modifying web parts

If you need to modify the behavior of a standard web part, you have the following options according to your requirements:

1. **You only need to set the web part 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 or design 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.

7.20.3.2 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 **zoneCenter** 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.

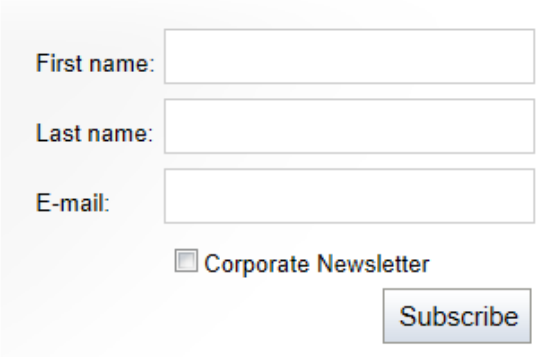
7.20.3.3 Customizing web part layout

Kentico CMS comes with many built-in web parts and their appearance and design can easily be modified to fit your specific requirements. The concept of web part layouts allows you to customize the look of a web part by defining a custom HTML layout. So the web part layout is basically a custom skin for the web part.

The layout of a specific web part can be selected by configuring () it on the **Design** tab of **CMS Desk** and switching to the **Layout** tab. Here you can select from a list of layouts created for the current web part, or write a new layout. This list of layouts can also be managed in **CMS Site Manager -> Development -> Web parts -> .. select the given web part from the tree ... -> Layout**.

Example: Customizing 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. Enter the following values:

- **Display name:** Narrow layout
- **Code name:** NarrowLayout

Enter the following HTML code:

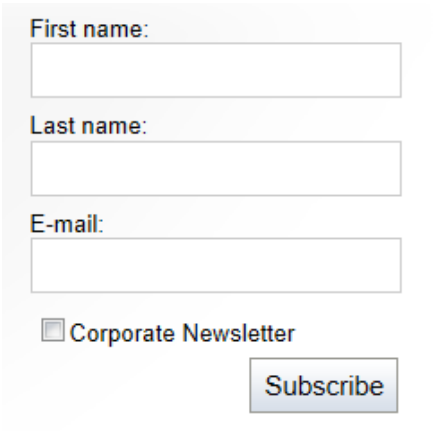
Please note: If you installed the Kentico CMS project as a web application, you need to rename the **CodeFile** attribute on the first line to *CodeBehind* for the code example to be functional.

```
<%@ 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):



Do not remove any controls from the layout

It's important to **keep all the controls** in the layout. If you need to hide some of them, you can add the *Visible="False"* attribute to the control, but the control must stay in the layout to allow the web part to keep working.

This issue **may also cause problems when upgrading to a new Kentico CMS version** - if some of the built-in web parts use a new control and you use your web part layout created in the previous version, the web part may stop working. Please be sure to test your website after an upgrade carefully if you're using web part layouts.

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.

7.20.3.4 Modifying 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:

- **Display name**: Form with custom e-mail
- **Code name**: FormWithEmail
- **Category**: Forms
- **File name**: BizForms/formwithemail.ascx
- **Clone web part files**: yes (checked)

Click **OK**. The system will create a copy of the existing web part using the new name and also copy the code files (ASCX and CS).

2. Now we will make the modifications to the web part. Open the web project using the **WebProject.sln** (or **WebApp.sln**) file in Visual Studio and edit the **~/CMSWebParts/BizForms/formwithemail.ascx** file.

3. If you installed Kentico CMS as a web application, the file will not be visible right away, since it is not included in the project. In this case, click the **Show all files** button at the top of the Solution Explorer, right click the **formwithemail.ascx** file in the BizForms folder and select **Include in Project**. It is also necessary to manually convert the control to the web application format. You can do this by right clicking the file again and choosing the **Convert to Web Application** option. This will generate the necessary designer file for the control. You may skip this step if you are using a web site project.

4. 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.

5. Next, 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.

6. 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.

7. 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.

7.20.3.5 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**, click **Listings and viewers** and click **New web part**.

2. Click **Inherit from an existing web part** and enter the following values:

- **Display name:** Custom news list

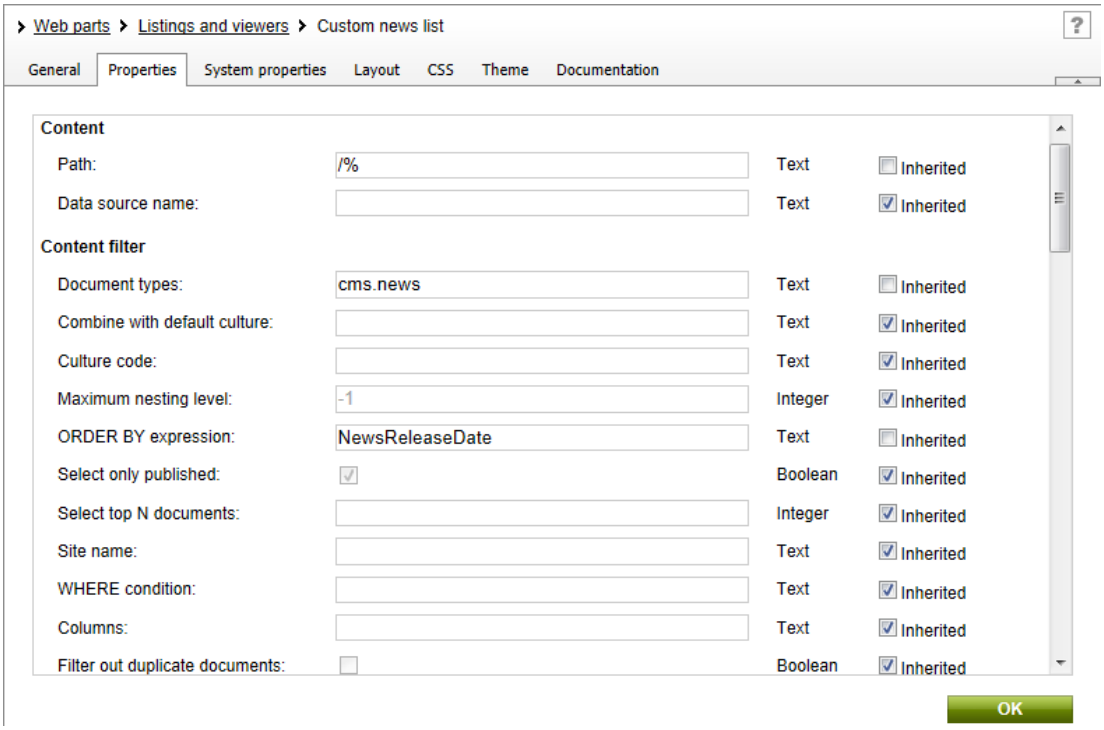
- **Code name:** CustomNewsList
- **Inherit from:** Listings and viewers/Documents/Repeater

Click **OK**.

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

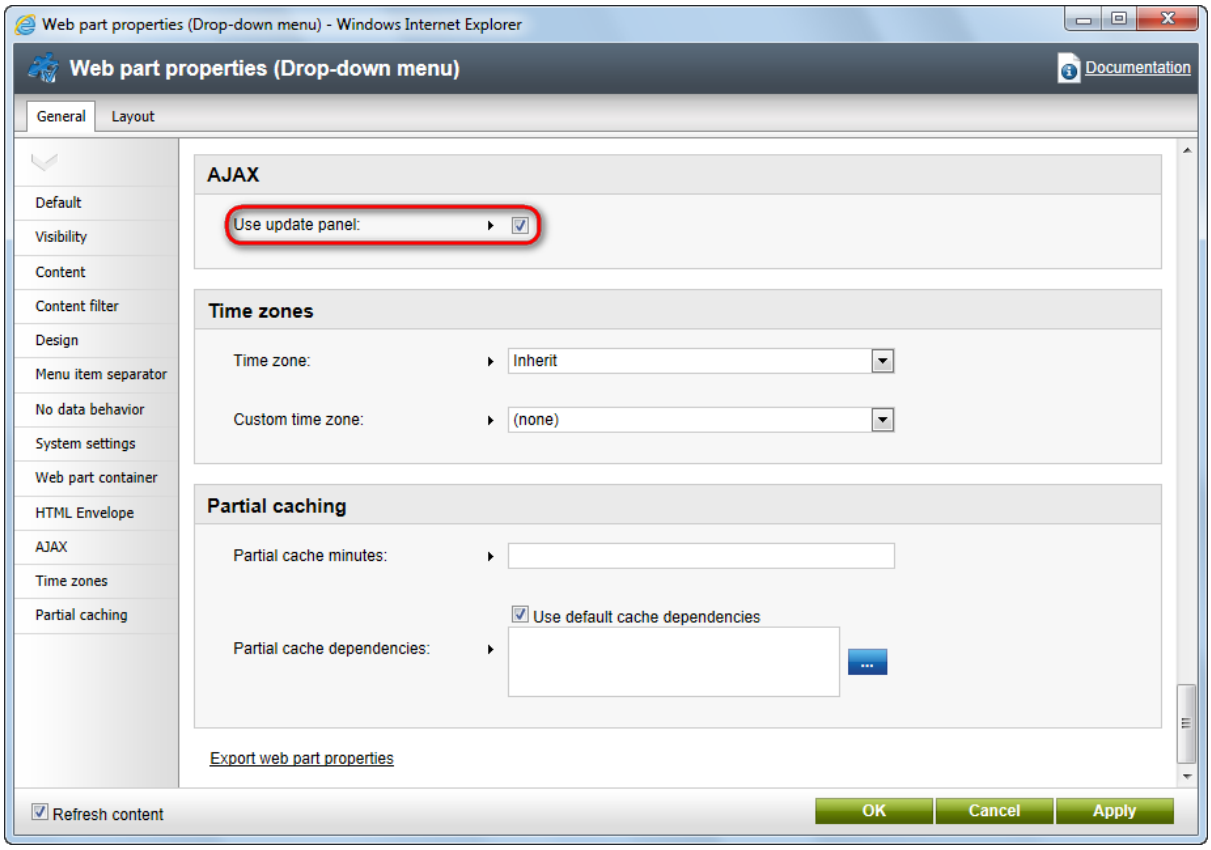
Click **OK**.



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.

7.20.4 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:

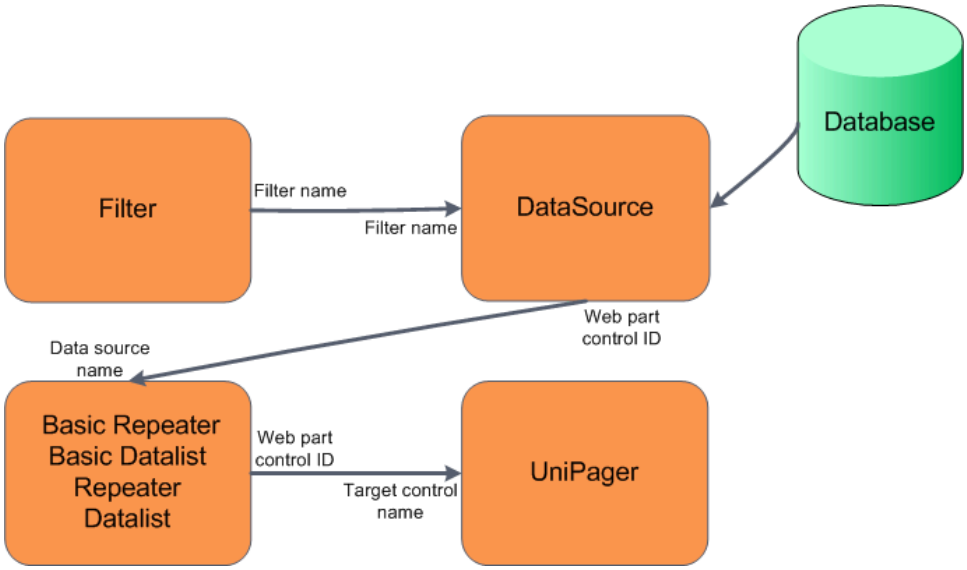
```
CMS.ExtendedControls.ControlsHelper.GetPostBackEventReference(this, "");
```

7.20.5 Data source web parts

7.20.5.1 Using Data source web parts

Data source web parts are designed for retrieving data from the database and sending it to other web parts that display it. This allows you to have separate web parts for retrieving data from the database, displaying data, filtering the displayed data and paging. These web parts can be placed anywhere on the page without any change in functionality and their design can be set separately, which results in higher design flexibility compared with using the original Repeater, Datalist, etc.

The following diagram shows how a Data source web part can be connected with other web parts to form a functional group of interconnected web parts. Captions of connecting lines show which properties of the web parts have to be set identically for the group to work properly.



Please note: Repeater and Datalist web parts can only be used for documents (provided by the **Documents data source** web part), not for e.g. custom tables etc. You will need to use the Basic repeater or Basic datalist for this purpose.

In the following example, we will create a group of interconnected web parts just as in the diagram above in order to see how the Data source concept works in practice.

1. Sign in to CMS Desk as the administrator (login *administrator* with blank password).

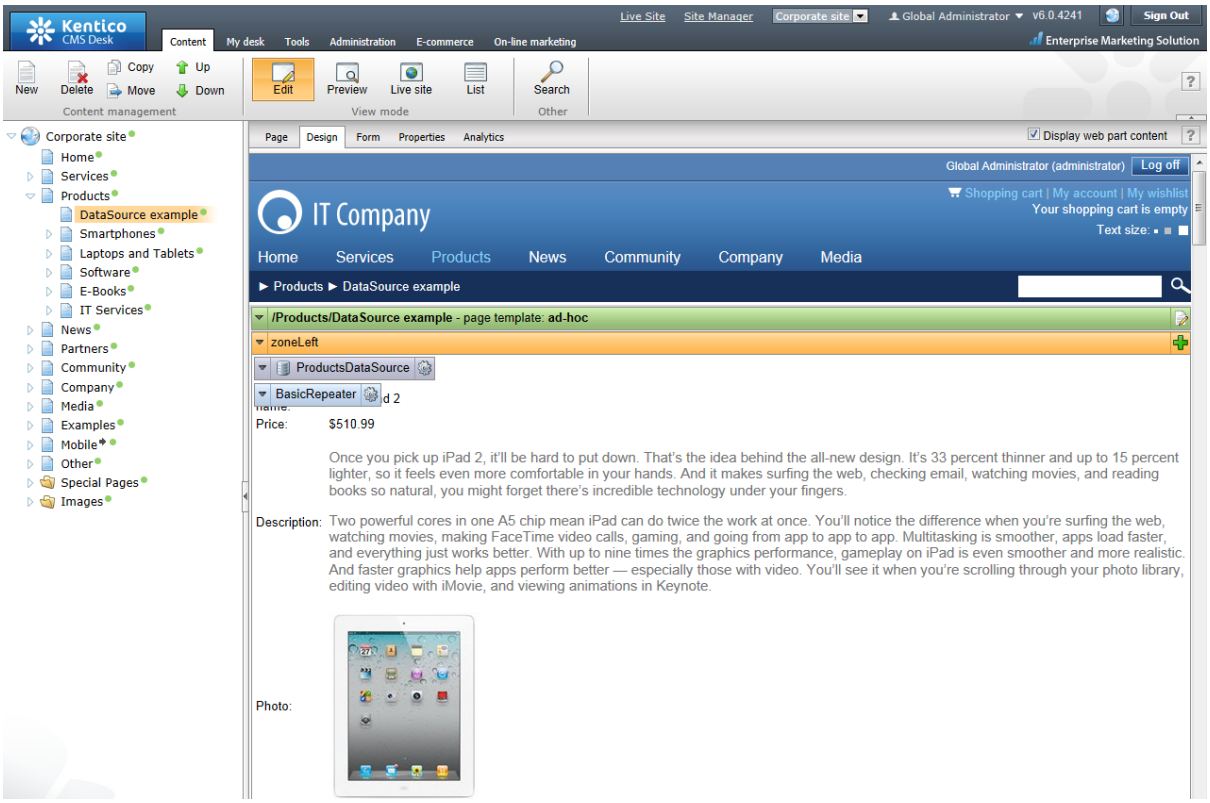
2. Create a new page under the **Products** section. Name it **DataSource example** and choose the **Create a blank page** option. Click **Save**.

3. Add the **Data Sources -> Products data source** web part to the only web part zone on the page. Leave default values for all its properties and click **OK**.

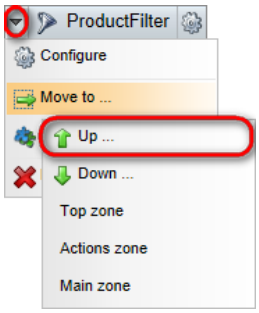
4. Add the **Listings & Viewers -> Basic Repeater** web part to the same web part zone. Set the following properties:

- **Data source name**: ProductsDataSource
- **Transformation name**: CMS.Product.Default

Click **OK**. You should see the repeater displaying products as in the following screenshot.



5. Now we will add the filter for the users to be able to filter displayed records. Add the **E-commerce -> Product filter** web part to the same web part zone. The web part will be added to the bottom of the page, so you may not be able to see it. You might want to scroll down and move the web part above the repeater. Open the web part's menu and choose to move the web part **Up**.



Set the following property of the filter web part:

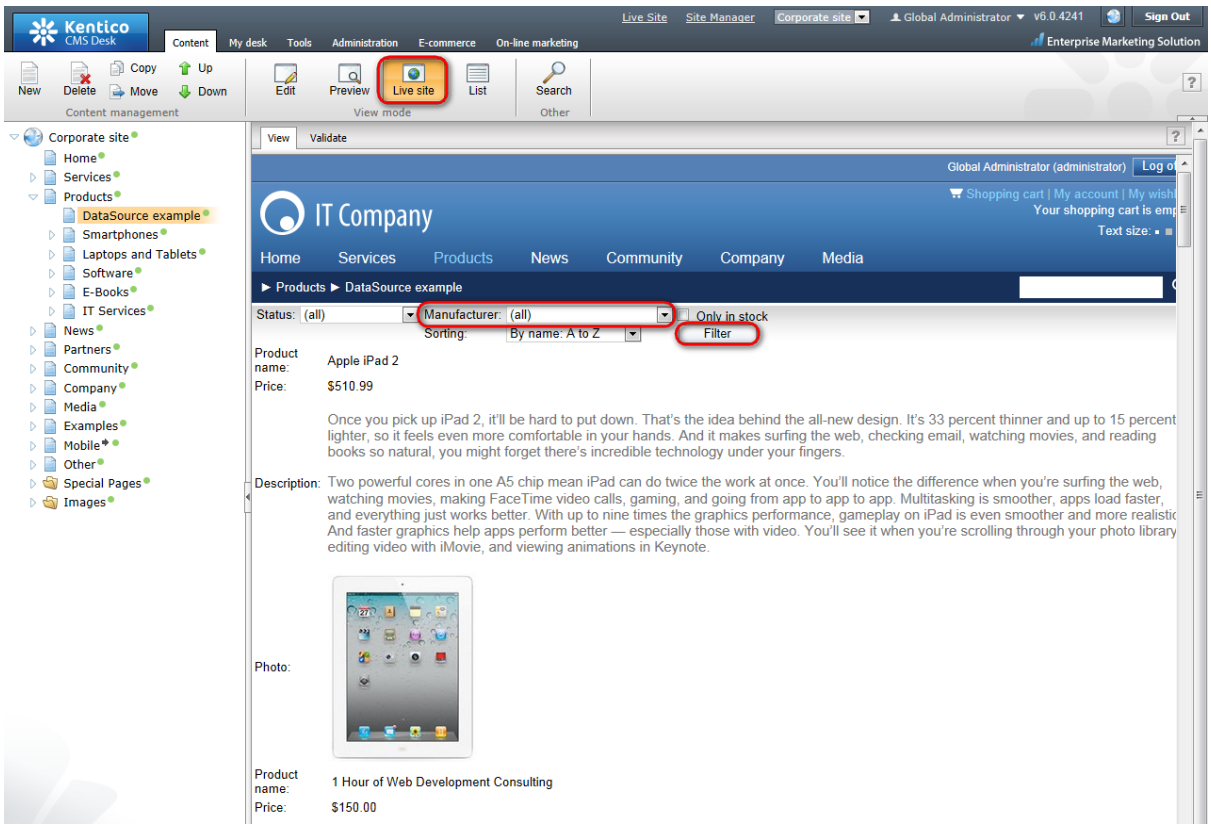
- **Filter name**: ProductFilter

Also open the properties of the Products data source web part and enter the same filter name into its **Filter name** property:

- **Filter name**: ProductFilter

You have just connected the data source with the filter. Data sent from the Data source to the repeater can now be filtered using the Product filter.

6. To verify the functionality, switch to **Live site** mode. You can for example choose to display only products manufactured by Asus using the **Manufacturer** drop-down list. After selecting, click **Filter** for the changes to take effect. After doing so, you should see only Asus products in the repeater, as you can see in the screenshot below.



7. As the amount of data displayed by the repeater might grow very large in some cases, the next logical step is to add a pager. Add the **Listings & Viewers -> Universal pager** web part to the web part zone and move it **Up** above the repeater the same way that you moved the filter in step 5.

Set the following properties:

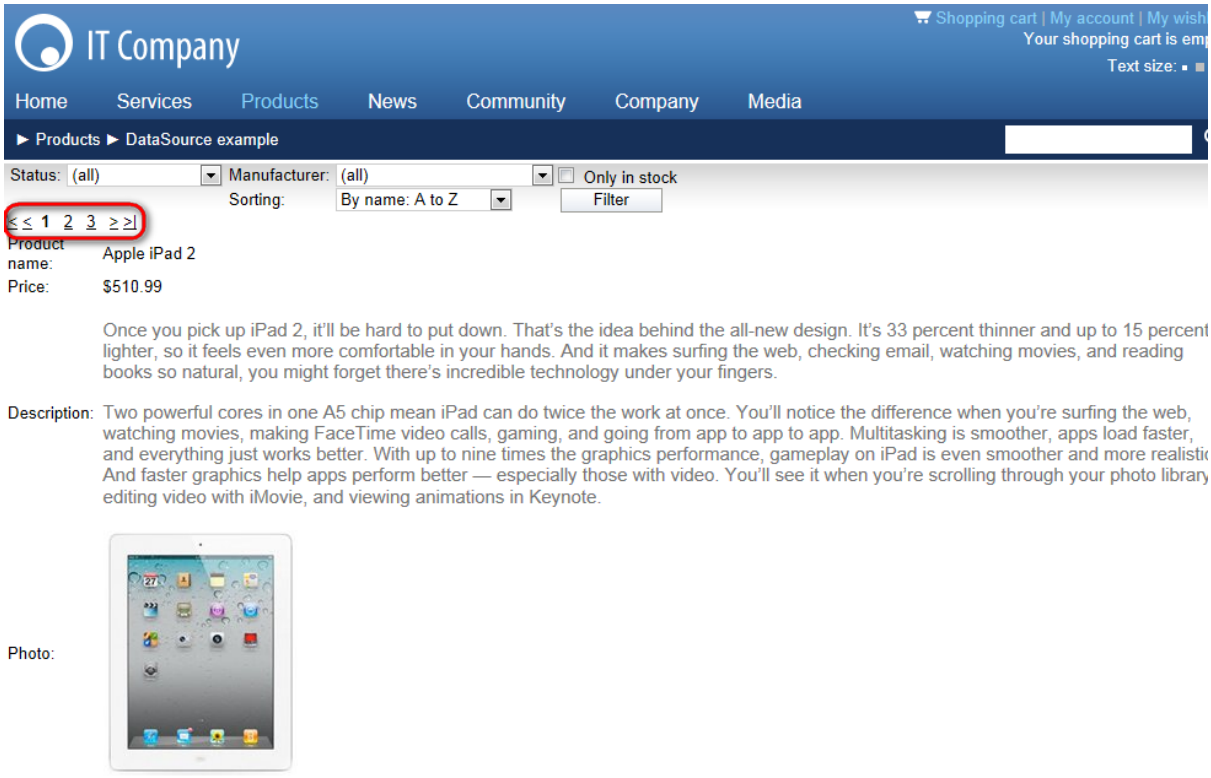
- **Target control name**: BasicRepeater
- **Page size**: 5
- **Group size**: 5

and make sure the following transformations are set:

- **Pages**: CMS.PagerTransformations.General-Pages
- **Current page**: CMS.PagerTransformations.General-CurrentPage
- **Previous page**: CMS.PagerTransformations.General-PreviousPage
- **Next page**: CMS.PagerTransformations.General-NextPage
- **Layout**: CMS.PagerTransformations.General-PagerLayout

Click **OK** to save the changes.

8. Now if you switch to **Live site** mode, the size of the page should be reduced to 5 products and you should be able to switch between pages using the pager.



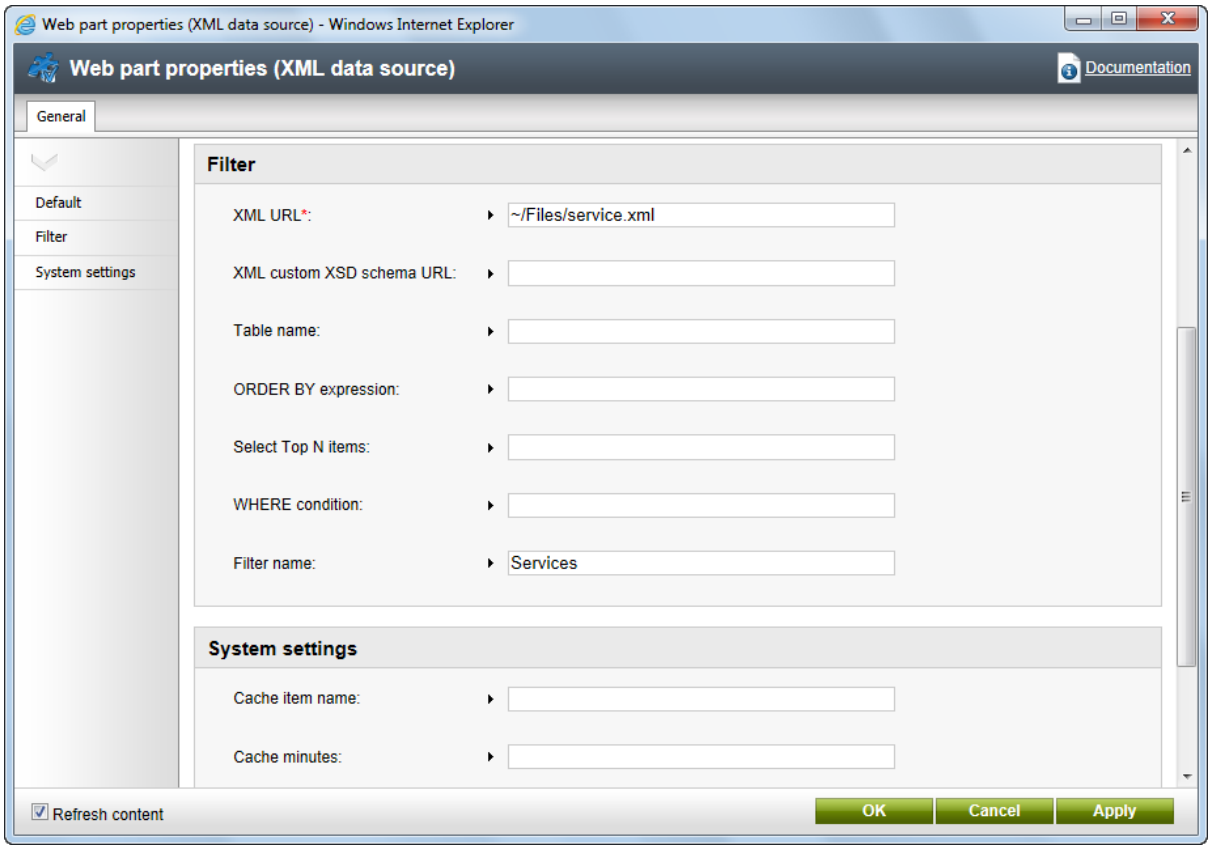
7.20.5.2 Problems with XML data sources

The XML data source web part can be used to provide data from an XML file specified by its **XML URL** property. It uses the Dataset.ReadXml() method to read the XML files. In some cases, this method may separate data from the source XML file into more than one dataset table. In this case, you need to specify which dataset table should be used through the **Table name** property.

Unfortunately, it cannot be explicitly determined into which table the data that you require is loaded. Therefore, we recommend finding this out using the **Debug** function in **Visual Studio**.

If the **Table name** property is not specified appropriately in this case, no data is provided to the connected Repeater. It is therefore advisable to disable the Repeater's **Hide if no record found** property in order to prevent the repeater from being hidden on the page, which may be confusing in some cases.

In some special cases, it may also happen that the required data may be distributed into more than one dataset table. In such a case, the **Table name** property cannot be used to solve the issue and the only workaround is to modify the source XML (if possible).



7.20.5.3 Developing Data source web parts

The code example below shows the creation of a sample custom DataSource web part for providing data from the CMS. The web part consists of two controls, so you need to take the following two steps when developing your custom DataSource web part:

1. Create a user control that inherits from **CMSBaseDataSource**. This will be the control that gets the

data.

- The **OnInit** and **DataFilter_OnFilterChanged** methods are necessary for handling a connected filter. If you are not planning to use a filter with the data source web part, you do not need to implement these methods.
- The **GetDataSourceFromDB** method ensures the loading of the actual data.

DataSourceControl.ascx

Please keep in mind that the following code is only an example, the actual values of the **CodeFile** and **Inherits** attributes will be different according to the name and location of the user control.

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="DataSourceControl.ascx.
cs" Inherits="CMSTestingSite_APIExamples_Controls_DataSourceControl" %>
```

DataSourceControl.ascx.cs

Please be aware that the name of the class will be different according to the name and location of the newly created user control.

[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 CMSTestingSite_APIExamples_Controls_DataSource :
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 filter change 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>
    /// Gets the datasource data.
    /// </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
}
```

2. Create another user control and insert the first control into it. This control will become the actual web part.

- The control must inherit from **CMSAbstractWebPart**.
- The line of the **SetupControl** method beginning with *this.srcUsers.FilterName* ensures that the inner datasource's name is set the same as the *Web part control ID* property of this web part. Don't get perplexed by the property being called *FilterName*. This is by design, because datasources use the same objects as filters.

DataSourceWebPart.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="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 CMSTestingSite_APIExamples_Controls_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)
        {
            // Do nothing.
        }
        else
        {
            this.srcUsers.WhereCondition = this.WhereCondition;
            this.srcUsers.OrderBy = this.OrderBy;

            // Sets the current web part name as the datasource name.
            this.srcUsers.FilterName = ValidationHelper.GetString(this.GetValue(
"WebPartControlID"), this.ClientID);
            // Sets the name of 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.

You can now register this control as a web part in Kentico CMS via the **Site Manager -> Development -> Web parts** interface (as described in the Developing web parts topic). Remember to set the **Type** of the web part to *Data source* on the General tab.

7.20.5.4 Developing custom filters

Even though Kentico CMS comes with a built-in set of filter web parts for various different types of data, it may still in many cases be necessary to create a custom filter according to your specific requirements.

This can be achieved by implementing the filter as a *user control* that inherits 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.

These base classes can be found in the **CMS.Controls** namespace. The appropriate base class must be selected depending on the type of data to be filtered.

When complete, the control can be placed onto your website through the **Filters -> Filter** web part. All that needs to be done is to specify the path to the *.ascx* file in the **Filter control path** property. The Filter web part can then be attached to data source web parts of the matching type.

Of course, a custom filter can also be used anywhere in your code as a standard user control, for example on ASPX templates or in other web parts. All necessary properties, like the **FilterName** used to connect with the data source are inherited from the used base class.

Example

The example below demonstrates how a custom filter can be developed. This sample filter will work with

product documents. It will support filtering according to product departments and allow the selection of several options that determine the order in which products are displayed. Filters for all types of documents or other objects can be created using the same approach.

1. Open the web project in Visual Studio and start by creating 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. Next, add a new **Web User Control** called *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. Now 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
{

    ...

}
```

The methods described in the following steps of this example will be added as members of this class.

4. Add the following three methods:

[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))
    {
        // Bind deparments to the drop-down list
        this.drpDepartment.DataSource = departments;
        this.drpDepartment.DataTextField = "DepartmentDisplayName";
        this.drpDepartment.DataValueField = "DepartmentID";

        this.drpDepartment.DataBind();

        // Add default '(all)' value
        this.drpDepartment.Items.Insert(0, new ListItem("(all)", "##ALL##"));
    }
}
```

These ensure that the correct filtering options are loaded into the child drop-down lists.

5. The most important part of the filter is performed by the **SetFilter()** method. Define it 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. Finally, add two methods that override the handlers of the **Init** and **PreRender** events as shown below:

[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);
}
```

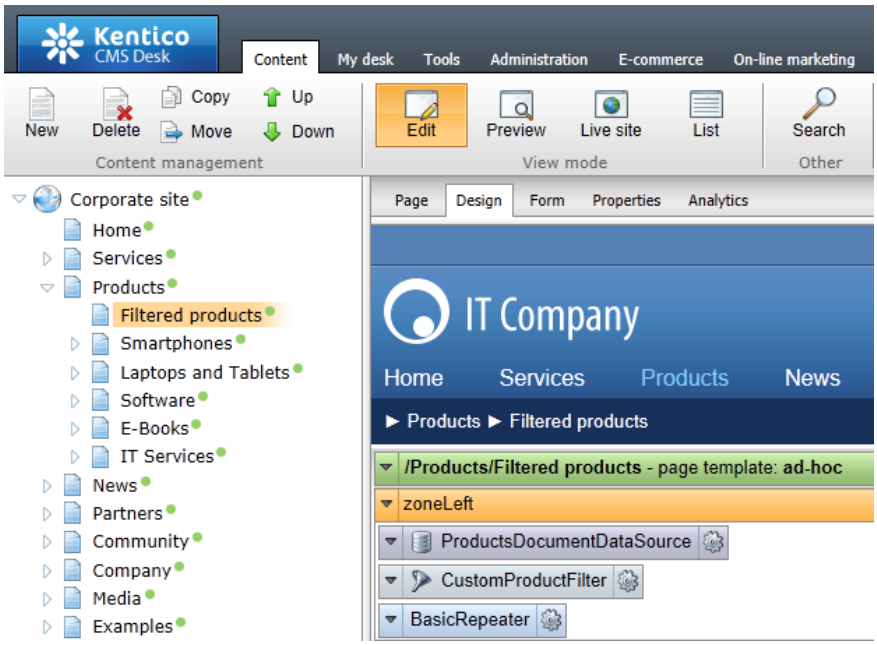
The handlers ensure that the appropriate private methods (defined previously in the example) are called during the correct stages of the page life cycle.

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.

Save the changes to both files.

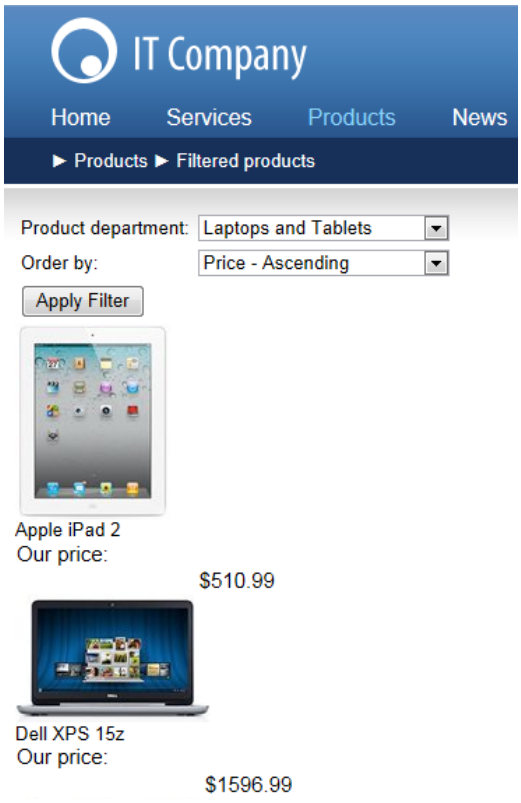
7. Now that the filter control is finished, you can try out its functionality. It is recommended to test the filter on the **Corporate** or **E-commerce** sample sites, since they already contain examples of product documents. Go to **CMS Desk**, create a new page and place the following web parts on it:

- **Documents data source** - configure it to load product documents and fill in its **Filter name** property (e.g. *CustomProductFilter)*. Alternatively, you can use the *Products data source* web part for this purpose.
- **Filter** - configure a matching **Filter name** (*CustomProductFilter*) and enter the path of the .ascx file implementing your custom filter into the **Filter control path** property (*~/CMSGlobalFiles/CustomProductFilter.ascx*).
- **Basic repeater** - enter the ID of the Documents data source web part into the **Data source name** property and set an appropriate **Transformation name**, for example *Ecommerce.Transformations.Product_SimplePreview*.



Detailed information about how this kind of web part combination works can be found in the Using Data source web parts topic.

8. If you open the page on the live site, you will see a list of products with the custom filter displayed above. The products can be filtered by their department and the order in which they are displayed can be changed.



7.20.6 Layout web parts

7.20.6.1 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.

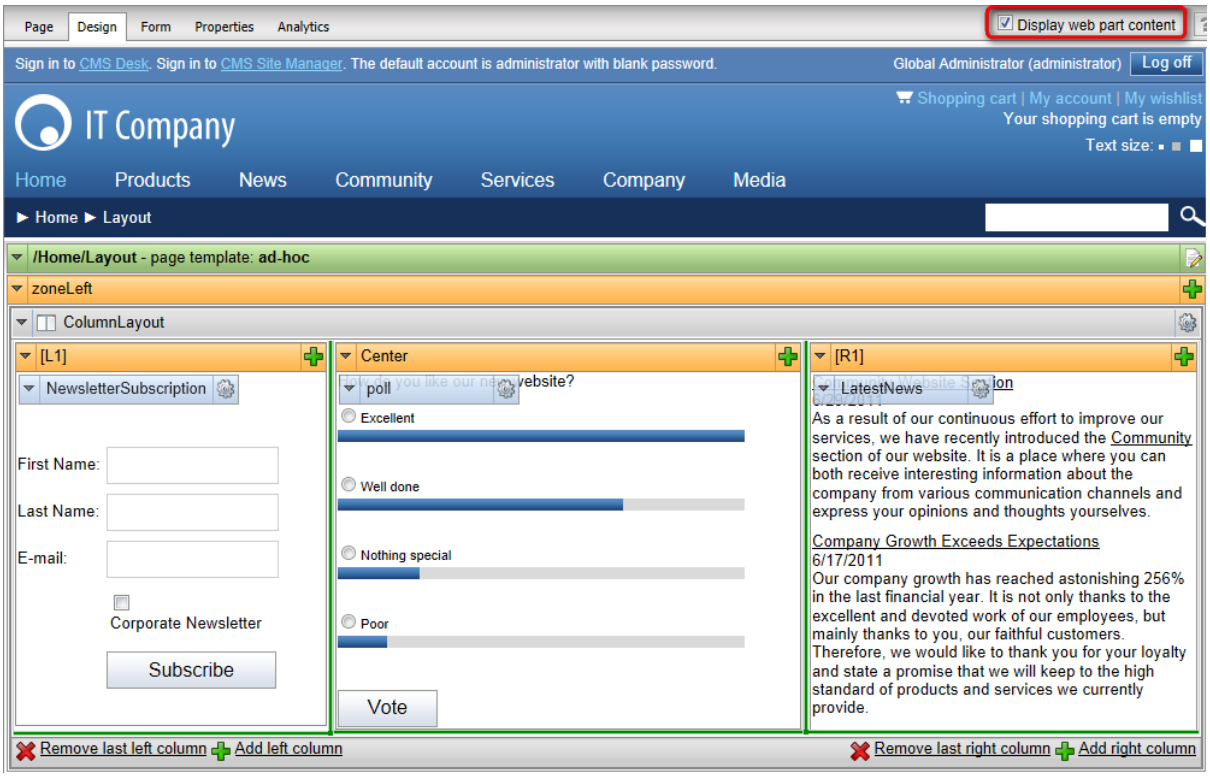
For additional details about individual web parts, please refer to the Kentico CMS Web Parts reference.

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. To ensure that all of the layout's zones are displayed correctly, it is necessary to enable the **Display web part content** option using the checkbox on the right side of the **Edit** mode header. This way 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.

7.20.6.2 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 name**: ~/CMSWebParts/Layouts/CustomZone.ascx

Click **OK** 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 **OK** again. All layout web parts must use this type in order to function correctly.

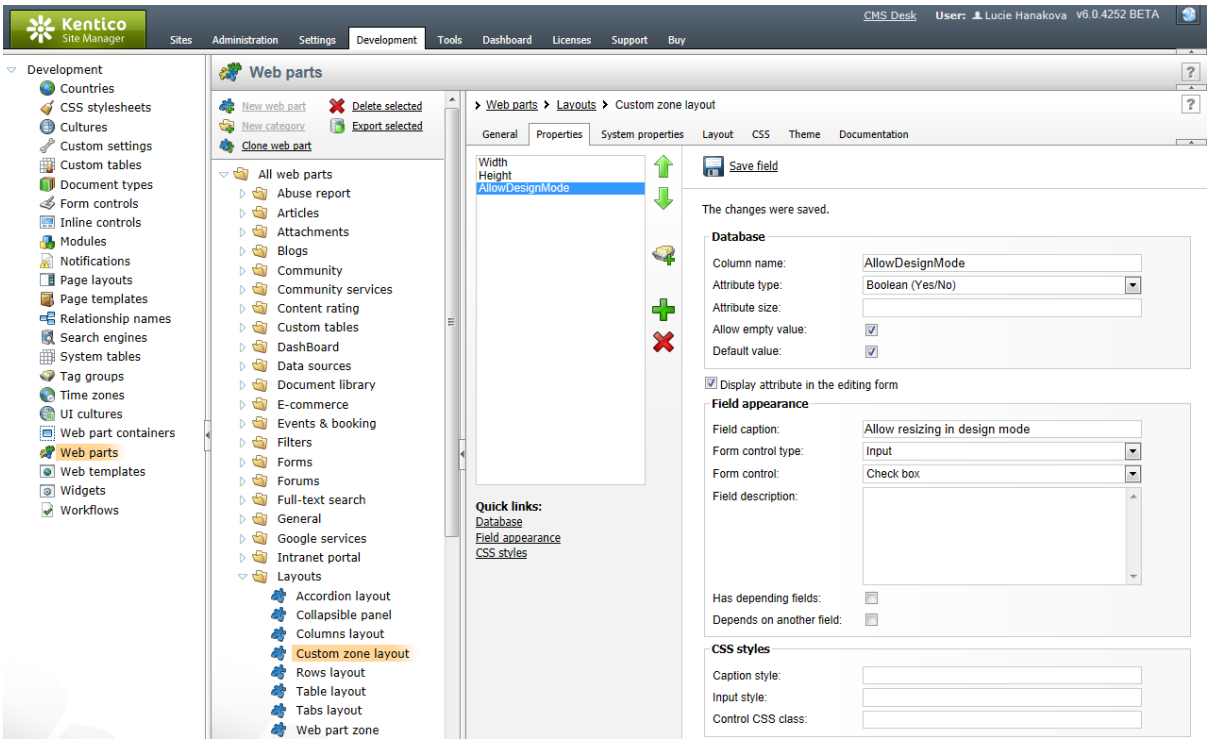
6. Switch to the **Properties** tab and add (+) three properties according to the information below. Click **Save field** for each property before moving on to the next one.

- **Column name**: Width
- **Attribute type**: Text
- **Attribute size**: 100
- **Allow empty value**: yes (checked)
- **Field caption**: Zone width
- **Form control type**: Input
- **Form control**: Text box

- **Column name**: Height
- **Attribute type**: Text
- **Attribute size**: 100
- **Allow empty value**: yes (checked)
- **Field caption**: Zone height
- **Form control type**: Input
- **Form control**: Text box

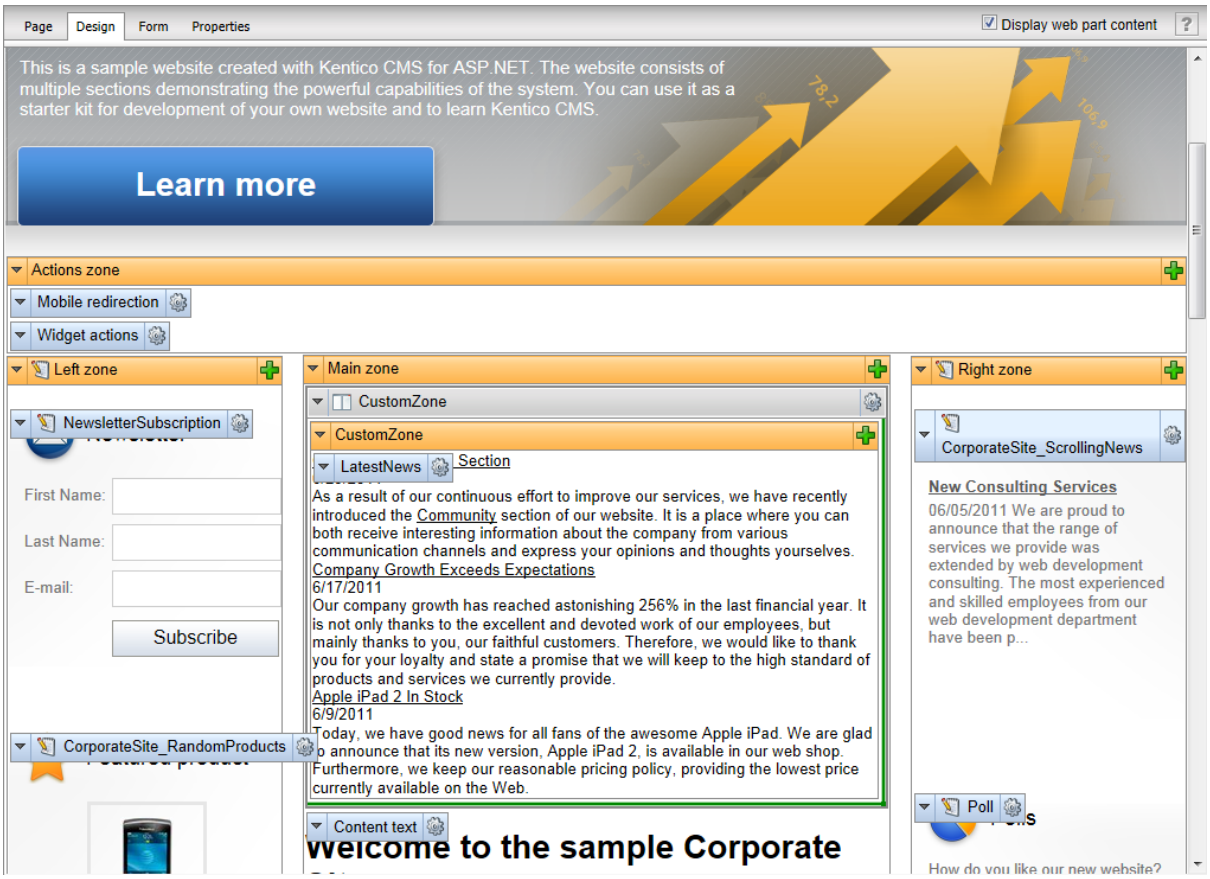
- **Column name**: AllowDesignMode
- **Attribute type**: Boolean (Yes/No)
- **Allow empty value**: yes (checked)
- **Default value**: yes (checked)
- **Field caption**: Allow resizing in design mode
- **Form control type**: Input

- **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.

7.20.7 Web part containers

7.20.7.1 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 using the **ContentBefore** and **ContentAfter** properties:

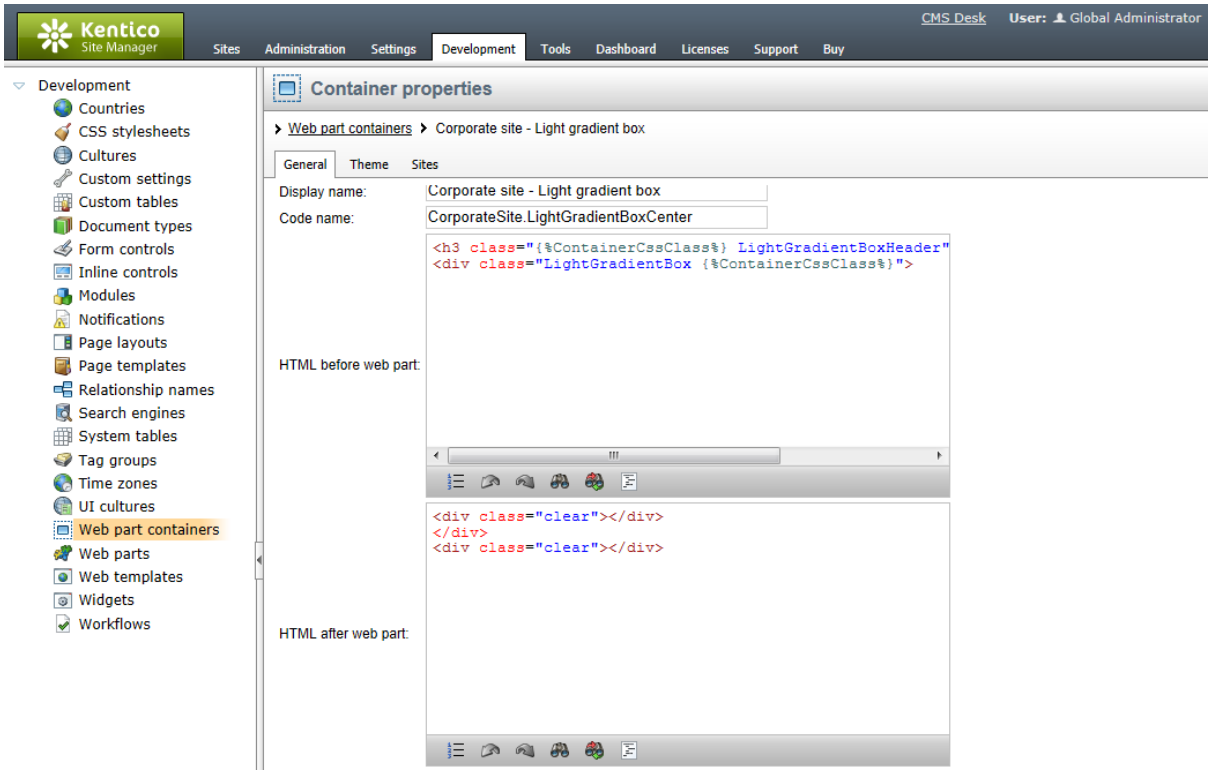
1. They are re-usable for many web parts.
2. They can contain a title and dynamically inserted values of the contained web part's properties.
3. They are objects in the system, which means they can be Exported and Imported.

Containers can be assigned not only to web parts, but also widgets and entire web part zones.

The containers can be managed at **Site Manager -> Development -> Web part containers**. The following properties can be defined when editing (✎) or creating a web part container:

Display name	The name of the container displayed to the users.
Code name	The name of the container used in code.

HTML before web part	HTML content displayed before the web part - the beginning of the envelope.
HTML after web part	HTML content displayed after the web part - the closing part of the envelope.
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.



The **HTML before/after web part** fields can contain dynamically inserted values of the contained web part's properties. You can insert them using the following expression:

{%propertyname%}

You will most often use the following expression to insert the container title:

{%ContainerTitle%}

These macro expressions are resolved even when macro resolving for the particular web part (**Disable macros** property) is disabled.

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_6.0/CMSPages/GetCSS.aspx?containers=BlackBox;OrangeBox" type="text/css" rel="stylesheet"/>

The value of the **containers** URL parameter is dependant on the containers used on the page, the example above is for a page that contains the *Black box* and *Orange box* containers.

If CSS minification is enabled, the request will use the following format instead:

<link href="/KenticoCMS_6.0./CMSPages/GetResource.ashx?containers=BlackBox;OrangeBox" 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.

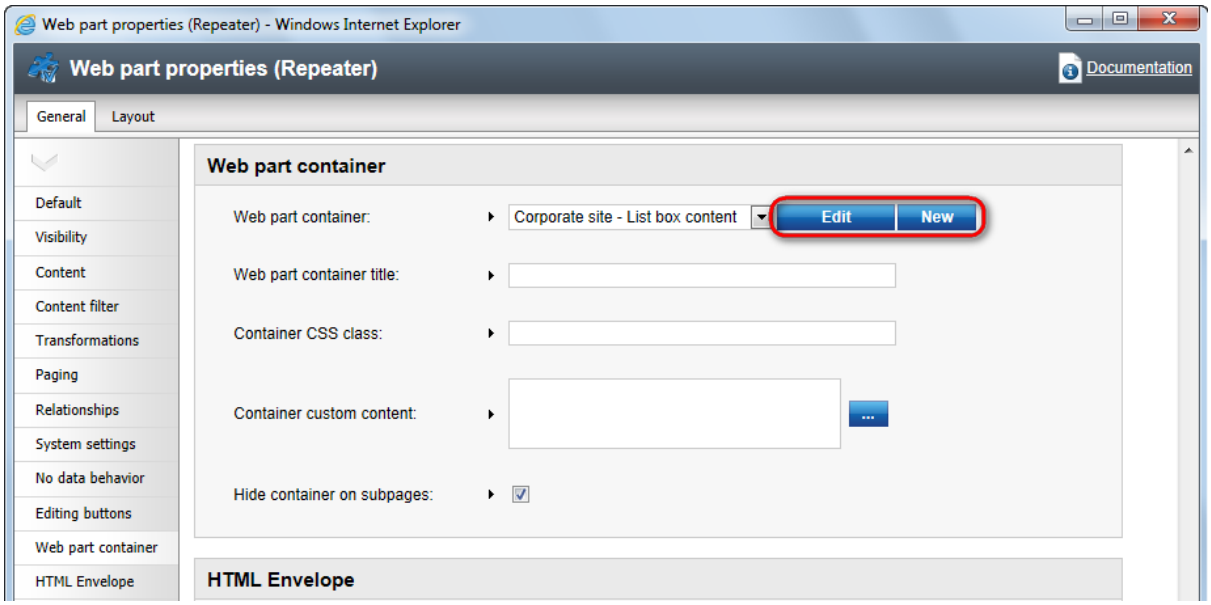
The content of this folder can be managed directly from the administration interface at **Site Manager -> Development -> Web part containers** by editing () the given container and switching the **Theme** tab.

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.

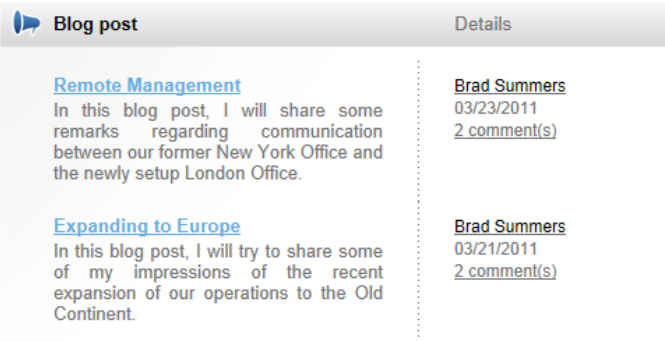
Using web part containers

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 is created.

7.20.7.2 Creating web part containers

Here's an example of defining a new web part container:

1. Go to **Site Manager -> Development -> Web part containers**.

2. Click **New container**.

3. Enter the following values:

- **Display name**: Blue box
- **Code name**: BlueBox
- **HTML before web part**:

```
<table width="100%" class="BlueTable" cellpadding="5" cellspacing="0">
<tr valign="top">
<td class="BlueTitle">
{%ContainerTitle%}
</td>
</tr>
<tr valign="top">
<td>
```

- **HTML after web part**:

```
</td>
</tr>
</table>
```

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 **OK** 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 display some page on the **Design** tab. **Configure** () some web part and set its properties like this:

- **Use container**: Blue box
- **Container title**: My web part with 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.

7.20.8 Web parts internals and API

7.20.8.1 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.

7.20.8.2 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.

7.20.8.3 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 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.

7.20.8.4 API examples

7.20.8.4.1 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;
```

7.20.8.4.2 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);
}
```

7.20.8.4.3 Managing web 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);
}
```

7.20.8.4.4 Managing w eb 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);
}
```

7.20.8.4.5 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);
}
```

7.20.8.4.6 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;
}
```

7.21 Widgets

7.21.1 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. As all widgets are based on existing web parts, you can alternatively create 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.

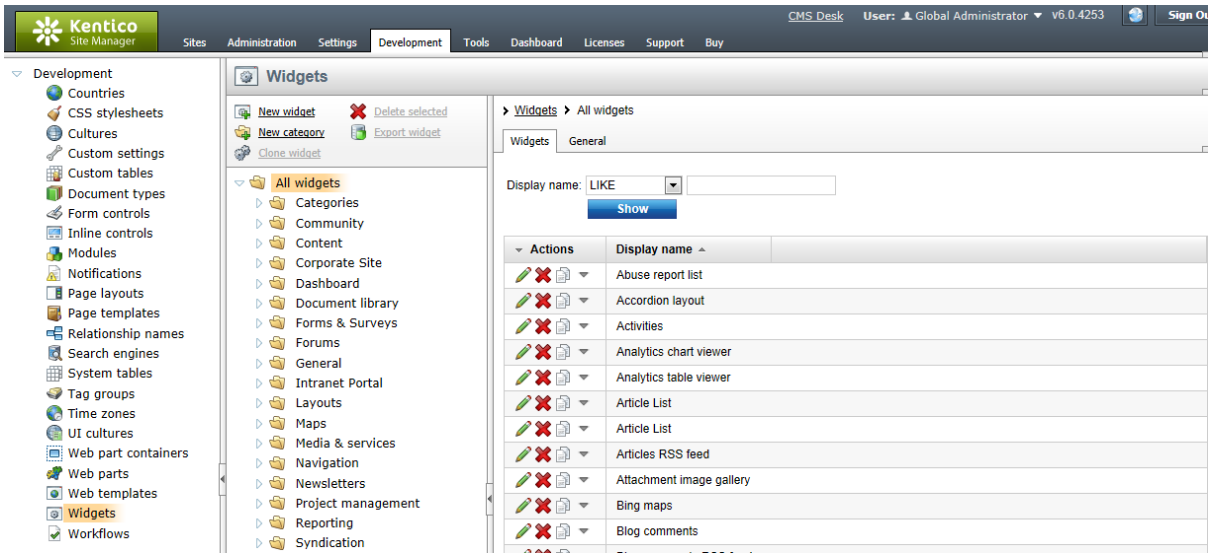
Security issues, such as the definition of where individual widgets can be used and permissions for different types of roles are discussed in the Security topic.

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.

7.21.2 Developing widgets

This topic will lead you step-by-step through the process of creating a widget. As a site developer, you can simply 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 will first have to develop the appropriate web part.

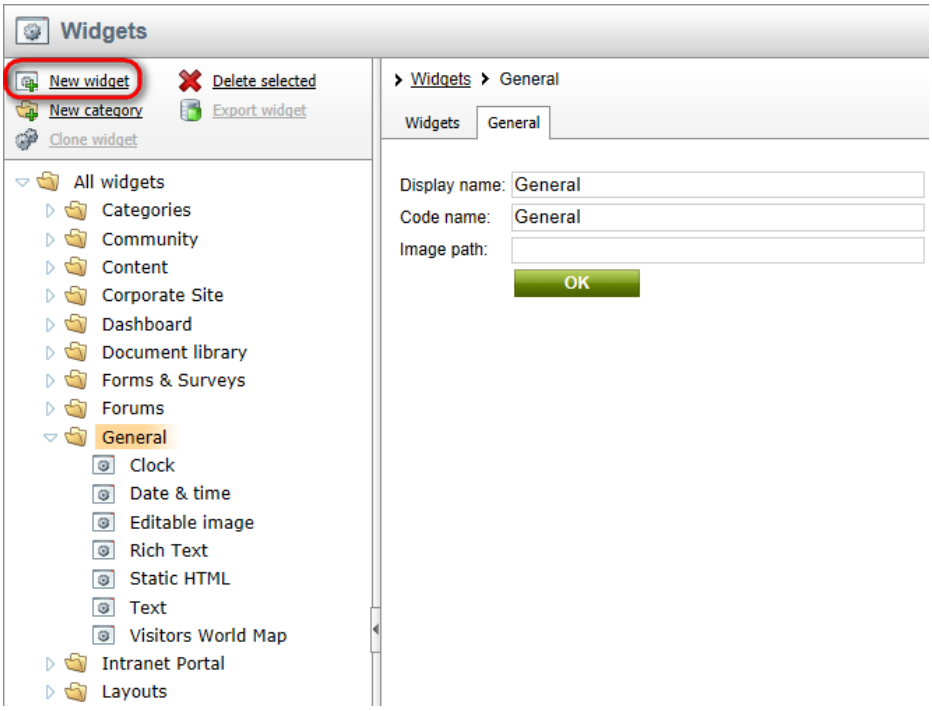
1. As mentioned above, it is not possible to create a widget without a reference to a web part. To start, switch to **Site Manager** -> **Development** -> **Widgets**, for the administration interface where widgets can be created. You should see a screen similar to the following one.



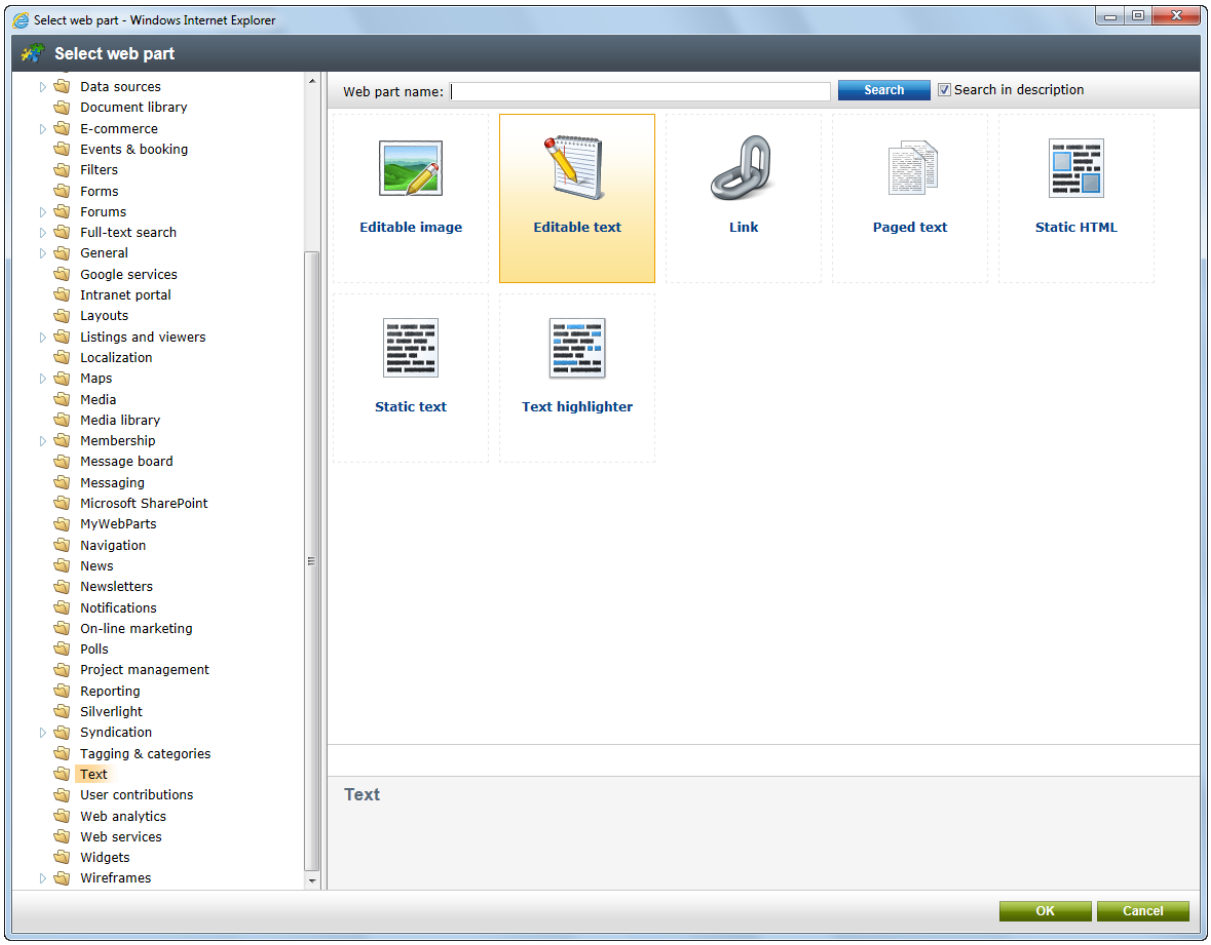
2. Widgets are grouped into categories. It is important to note that categories are used for the sake of organization only. The category under which a given widget is stored has no influence on who can use or modify the widget.

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 which can be used as the category icon in the **Add widget** dialog (the recommended size is 16x16px).

Now select any category where the new widget will be stored 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.



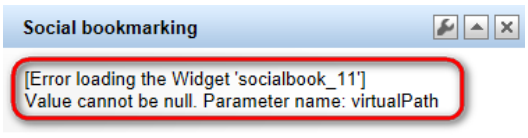
3. A dialog opens, where you can select the web part that will be used as a template for the new widget. In the example below the **Editable text** web part from the **Text** category is selected. Click the **OK** button to confirm the selection.



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 the **Clone widget** link. If you are about to create a new category, use the **New category** link.

To delete a selected widget or an entire category (except the *All widgets* category), click the **Delete selected** link. 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.

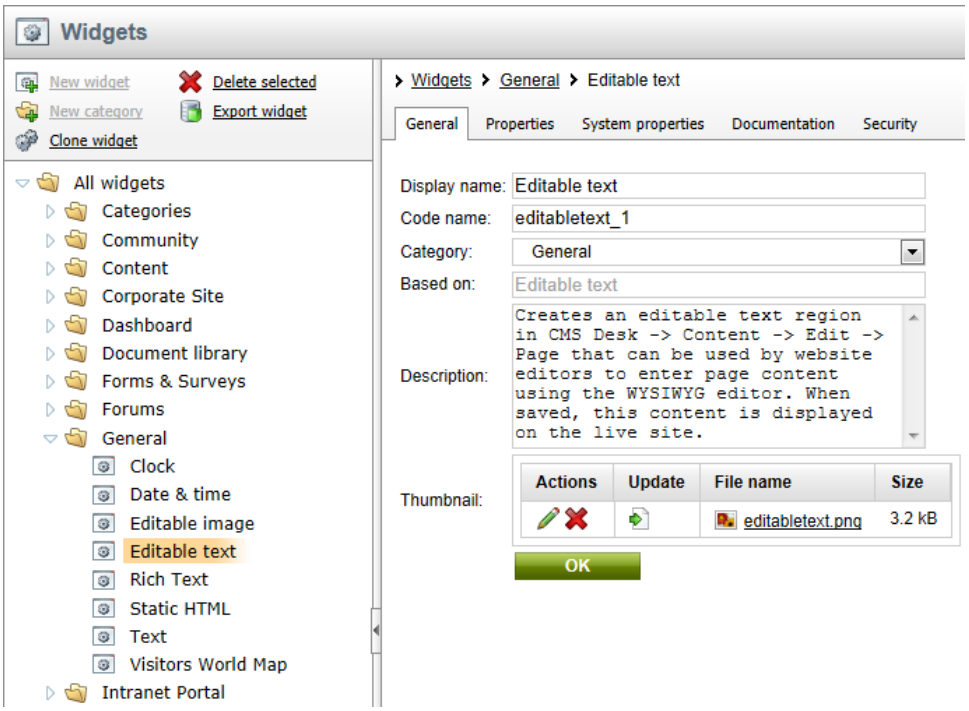


A selected widget can also be exported by clicking on the **Export widget** link. For details regarding export see Exporting single objects.

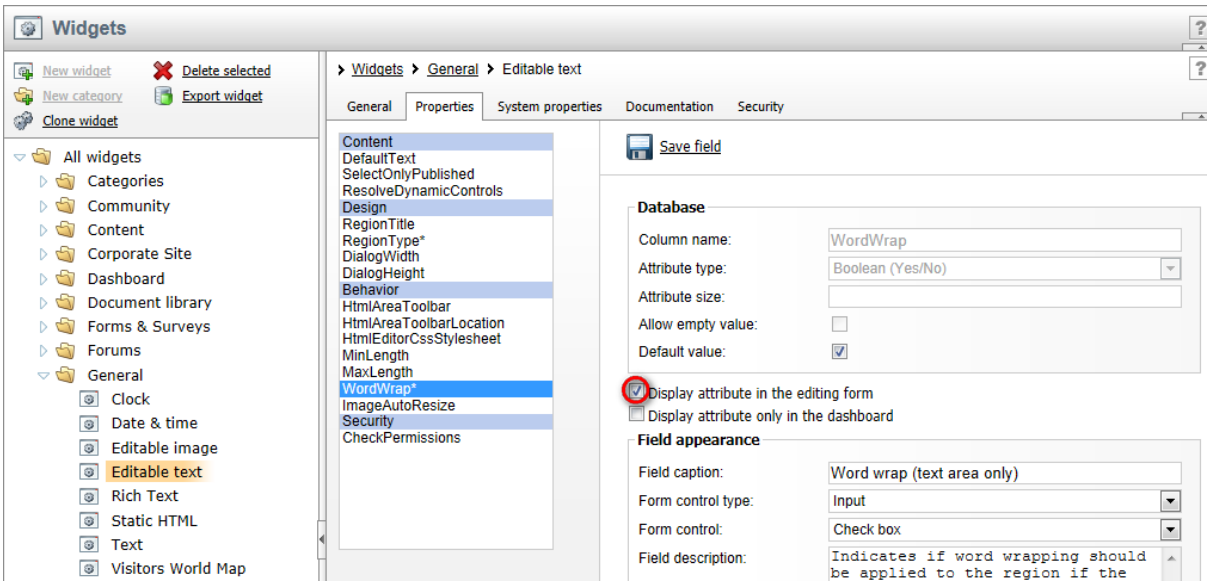
5. For each selected widget there are five tabs available. On the **General** tab there are all the basic settings:

- **Display name** - a descriptive name for the widget used in the administration interface and widget selection dialog.
- **Code name** - the name of the widget used in website code.
- **Category** - defines the category to which the widget belongs. You can change it here by selecting a new category from the drop-down list.
- **Based on** - the name of the web part that the widget is derived from.
- **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.

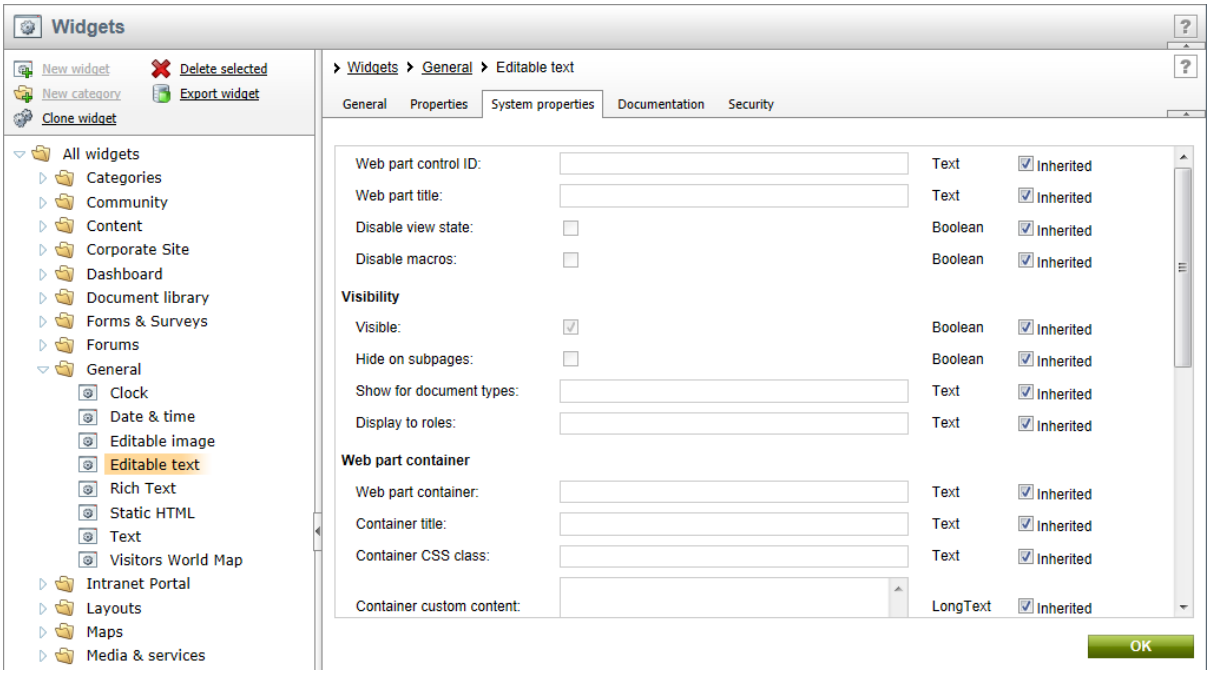
Perform any changes required and click the **OK** button.



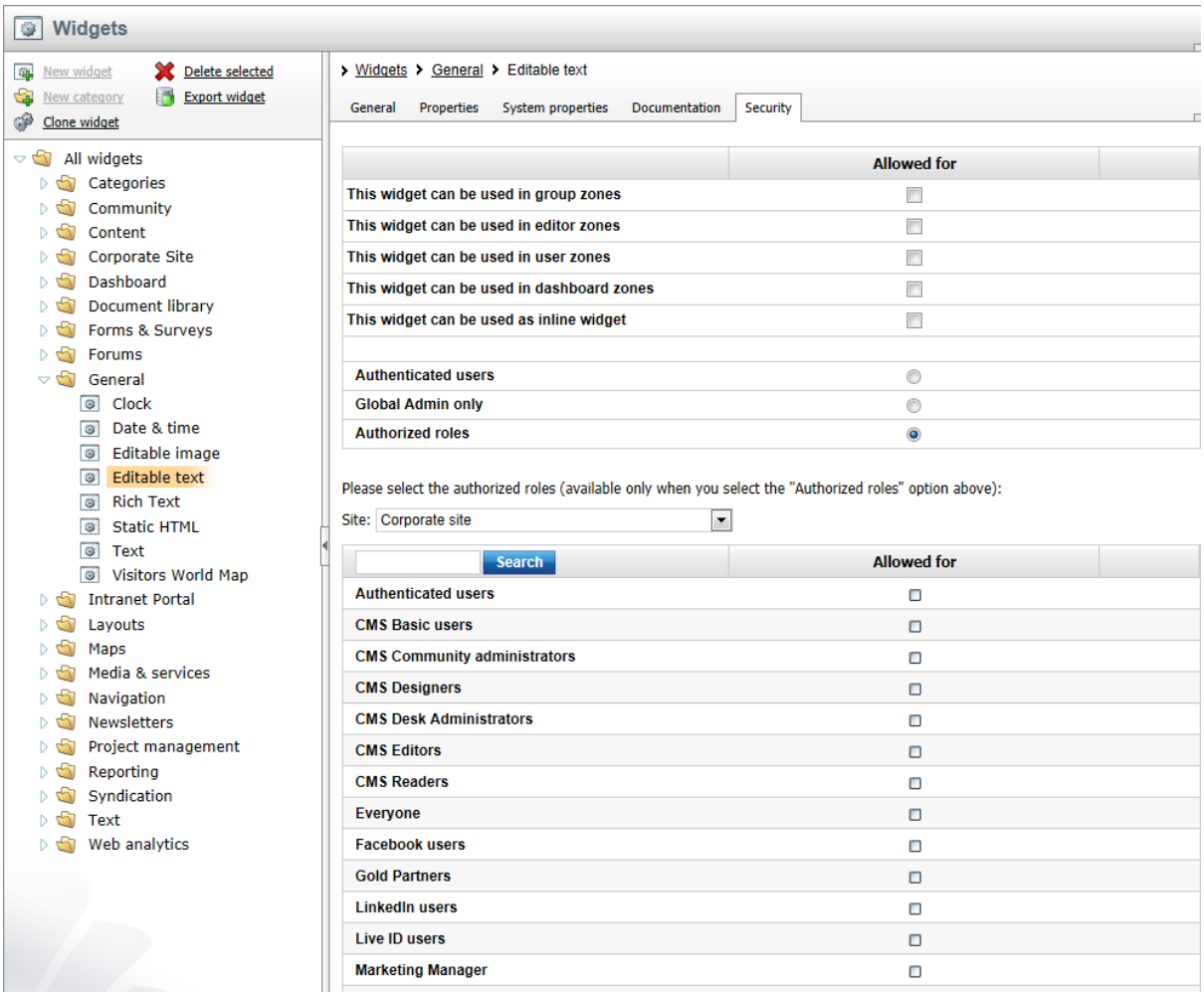
6. The **Properties** tab includes all the properties taken from the parent web part. Using the **Display attribute in the editing form** checkbox, you can choose if a given property should be available to users in the widget configuration dialog. You can notice that all properties are hidden by default. If the **Display attribute only in the dashboard** box is also checked, the property will only be accessible when configuring the widget on one of the dashboard sections located in the CMS administration interface. Please bear in mind that any changes must be confirmed by clicking the **Save field** button at the top.



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 the **OK** button.



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. When all settings are done the widget is ready for use in a page template.

7.21.3 Using widgets

There are different sets of widget related actions that users can perform depending on their role as shown below:

- Site developers/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).

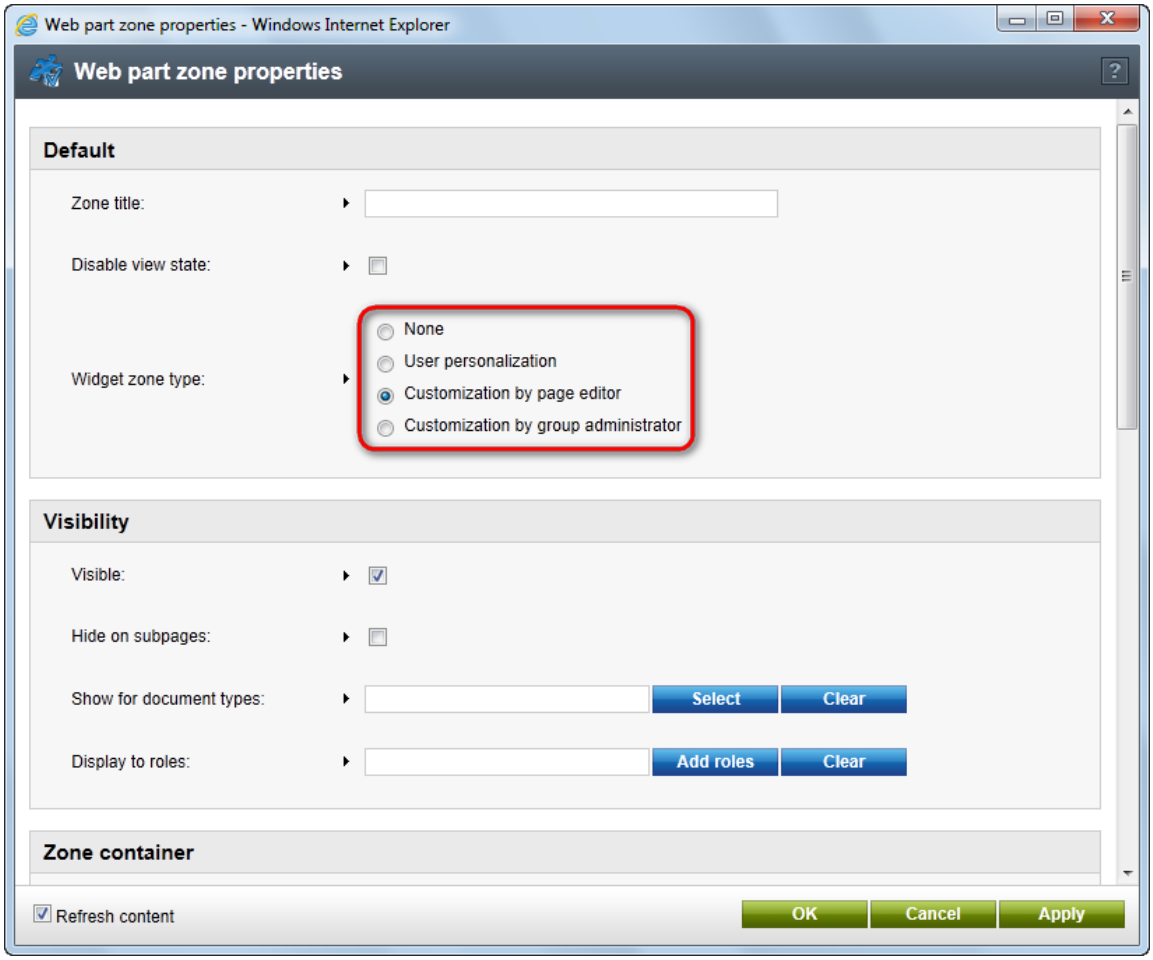
To change a web part zone into a widget zone, go to **CMS Desk -> Content -> Design**, expand the menu of the selected web part zone (▾) and click on the **Properties** item.

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

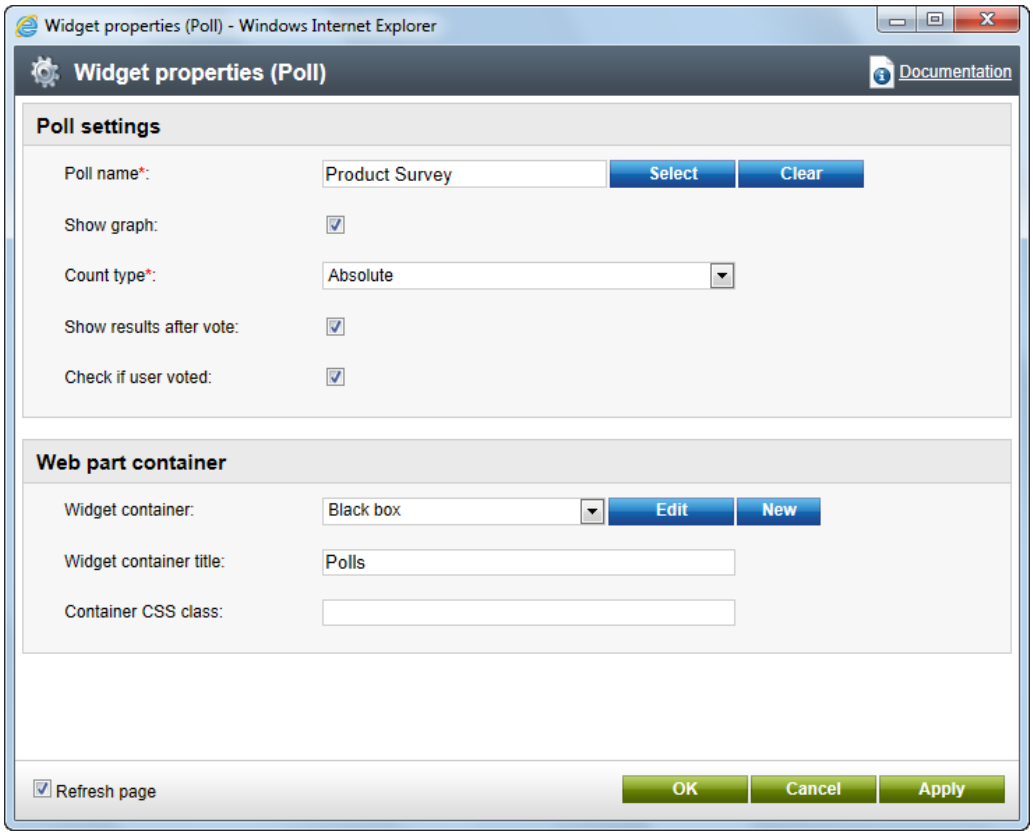
it on the live site. The zone will be marked with the **User zone** ( ) icon.
- **Customization by page editor** - the zone will be a widget zone and website editors will be able to customize it. The zone will be marked with the **Editor zone** ( ) icon.
- **Customization by group administrator** - the 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.

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, where you can change any properties available for customization. To save any changes, just click on the **OK** button.



This way, developers can define the default content of widget zones. It is up to the users to personalize these zones further as is 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 any widget placed in an editor zone by clicking on the **Configure widget** ( ) button. You can also delete widgets using the **Delete widget** ( ) button. Widgets can be moved between different widget zones by clicking on the **Drag widget** ( ) button and dragging 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.

Please note

Group ownership of a document can be configured at **CMS Desk -> Edit -> Properties -> General -> Owned by group**.

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

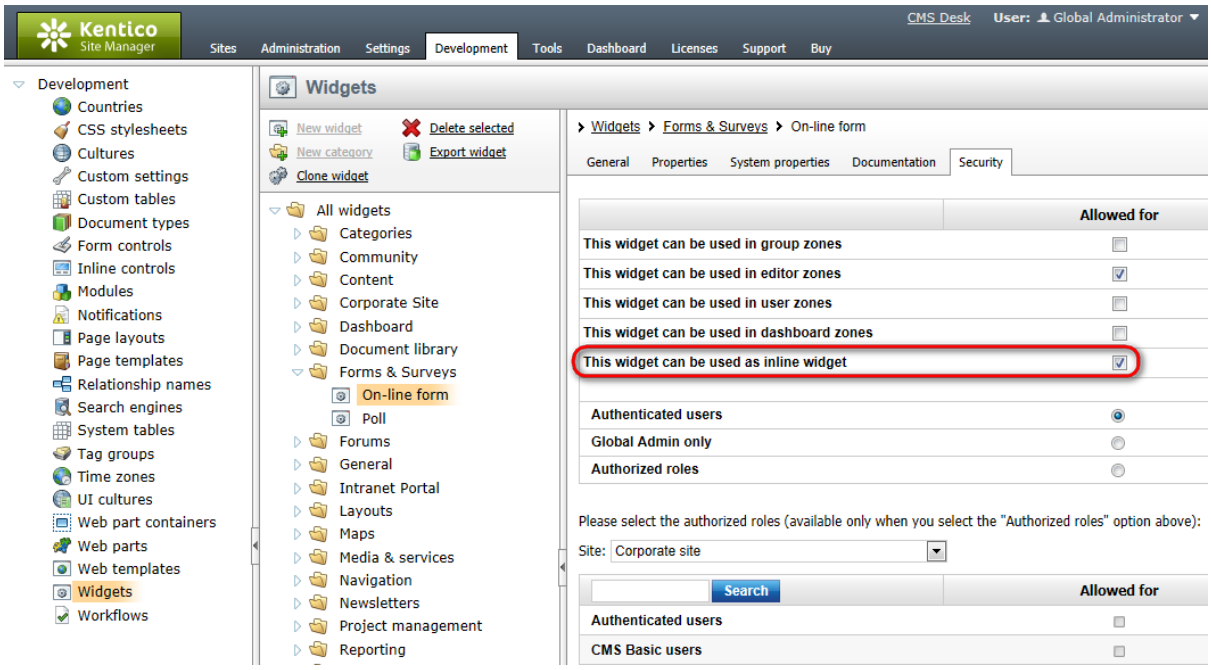
If there is a **Widget actions web part** on the page, users can add widgets or reset all widget zones on the page to their default setting by clicking on the appropriate links. The web part can only work with the type of widget zones specified by its **Widget zone type** property. When adding a widget using the **Add widget** link, the widget is placed into the zone specified by its **Widget zone** property. Users can drag the widget into another zone if the default placement is not suitable.

7.21.4 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 by using widgets.

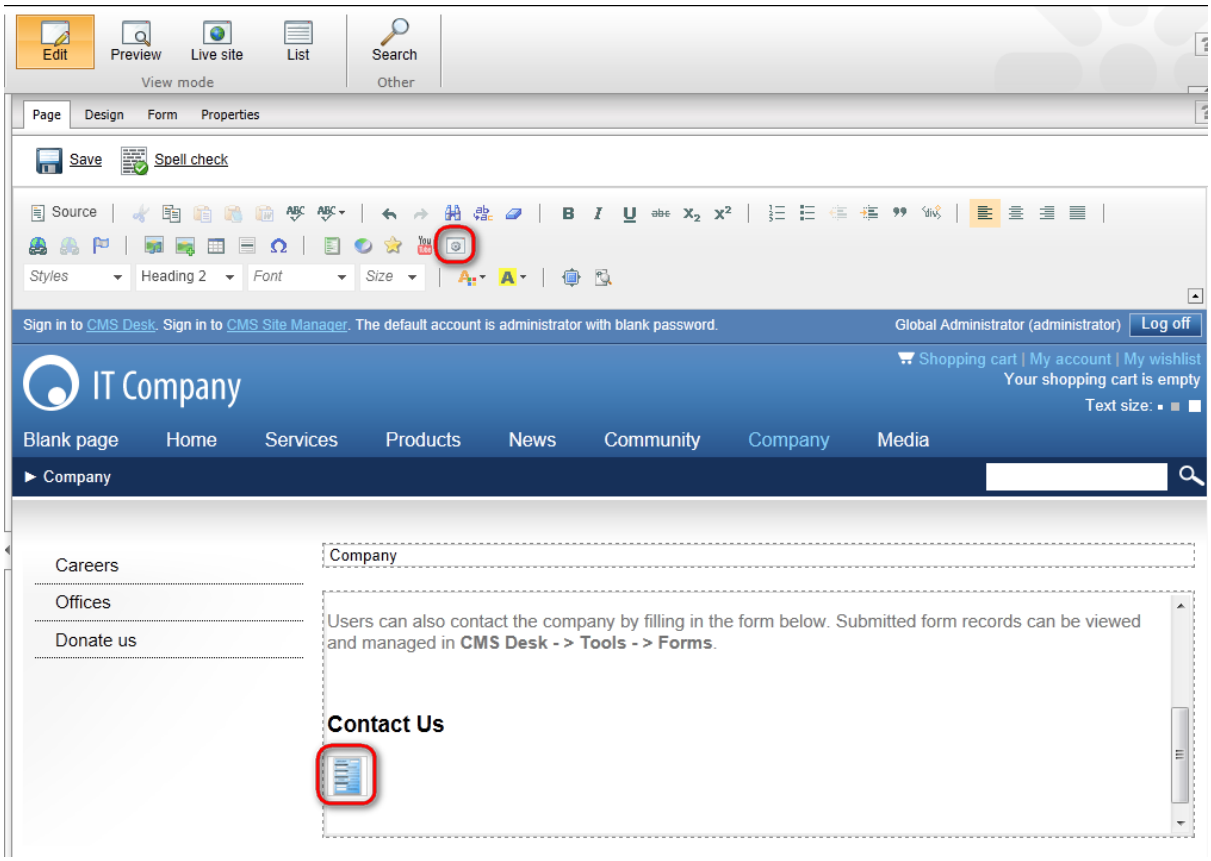
How to insert inline widgets into text

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**.

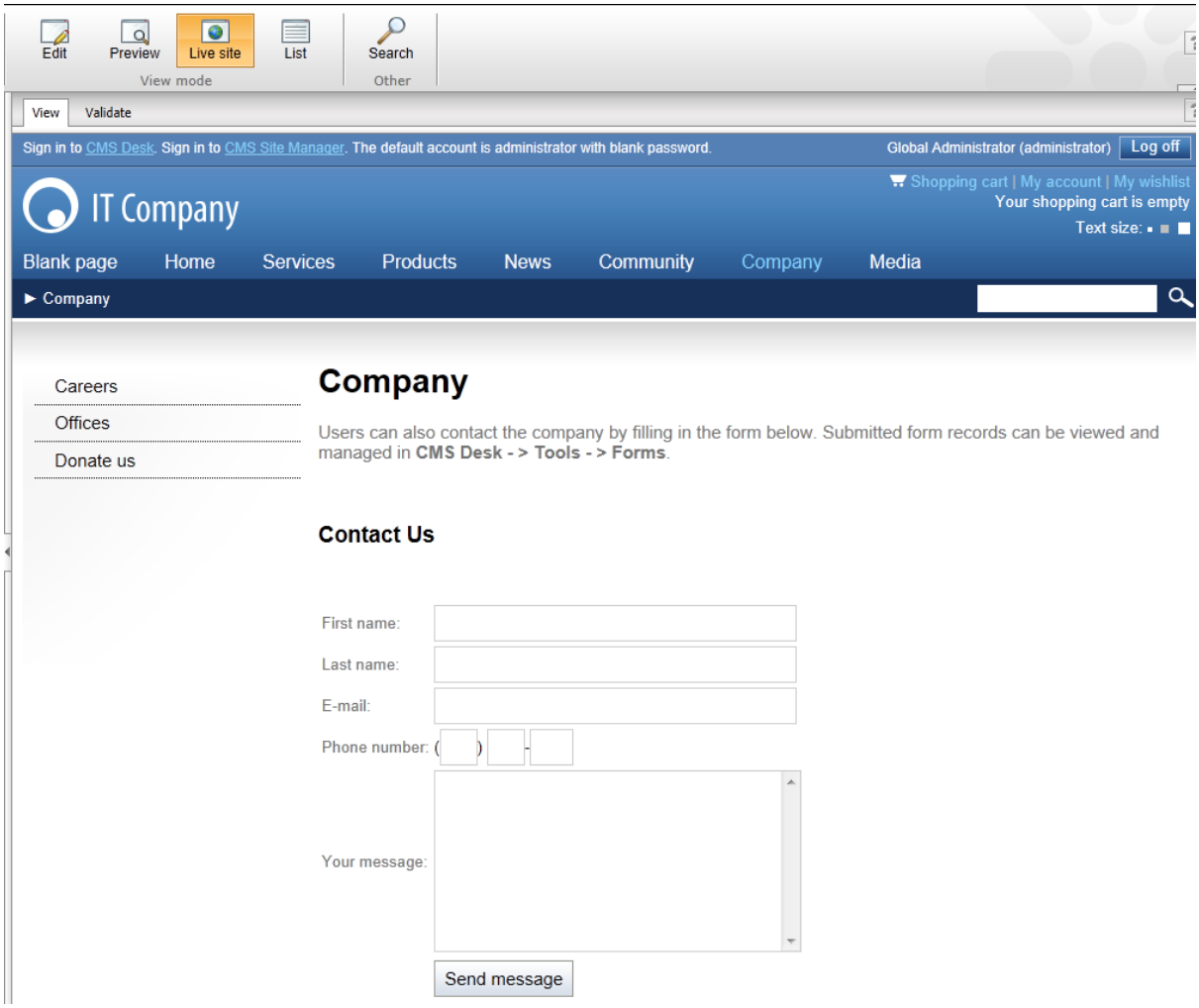


After you select the desired widget, the **Widget properties** dialog opens, where you can configure any of the widget's available properties. Once inserted into the text, an inline widget is represented by a placeholder image on the **Edit** tab of **CMS Desk** and is displayed fully on the **Preview** 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**:



How to configure inline widgets

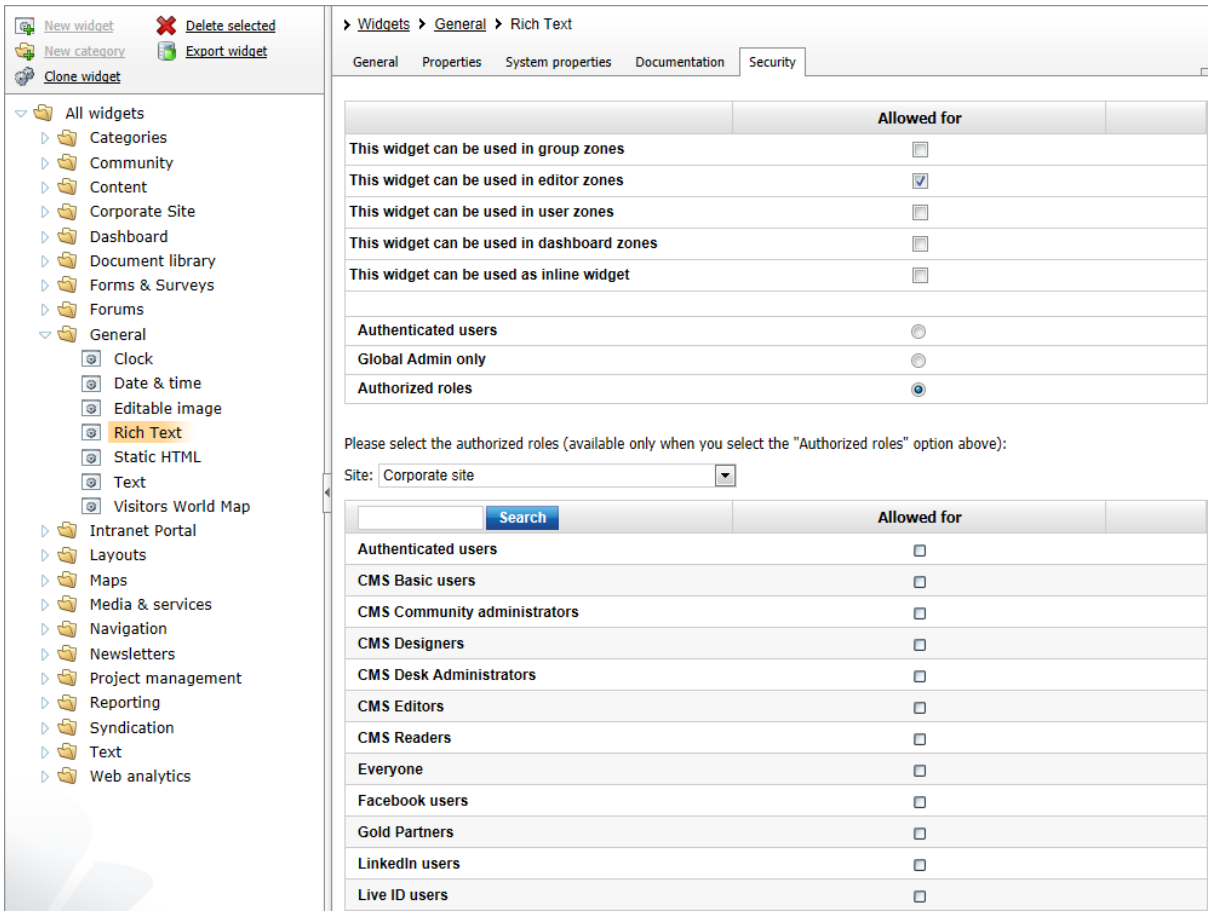
There are three different ways to open an inline widget's properties dialog:

- 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

7.21.5 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.

7.21.6 Widgets internals and API

7.21.6.1 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.

7.21.6.2 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).

7.21.6.3 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.

7.21.6.4 API examples

7.21.6.4.1 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;
```

7.21.6.4.2 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);
}
```

7.21.6.4.3 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);
}
```

7.21.6.4.4 Managing w idget 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;
}
```

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.

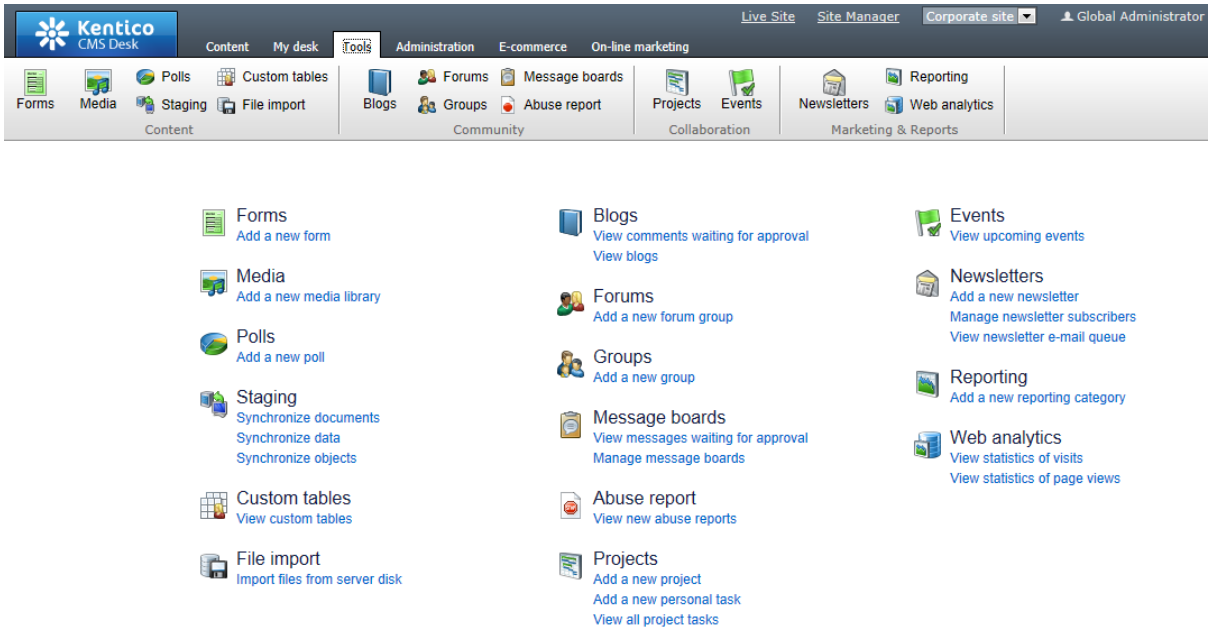
This topic describes where you can easily access Kentico CMS modules and how to customize the user interface.
In this example, you can learn how to develop a simple module and integrate it into the Kentico CMS user interface.

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

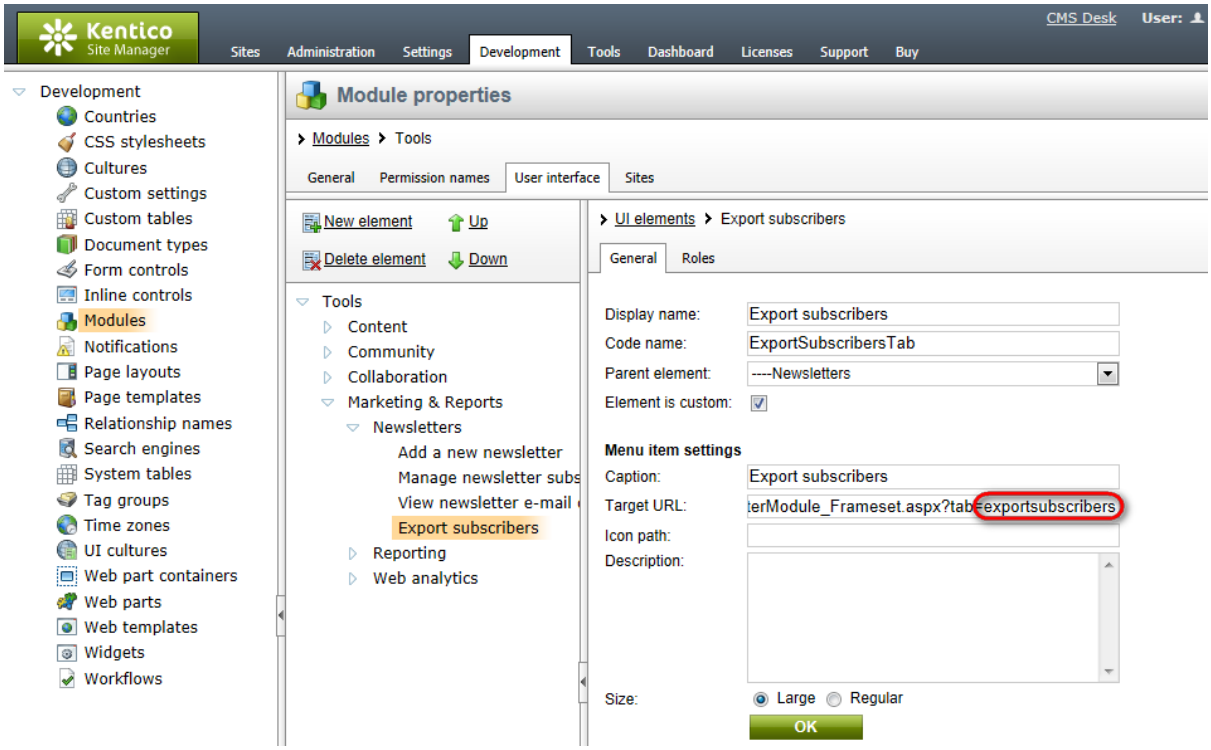
How to do it in general

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 the **New element** (📄) link and proceed as described in the UI personalization chapter in the Membership, permissions and security section of this guide.

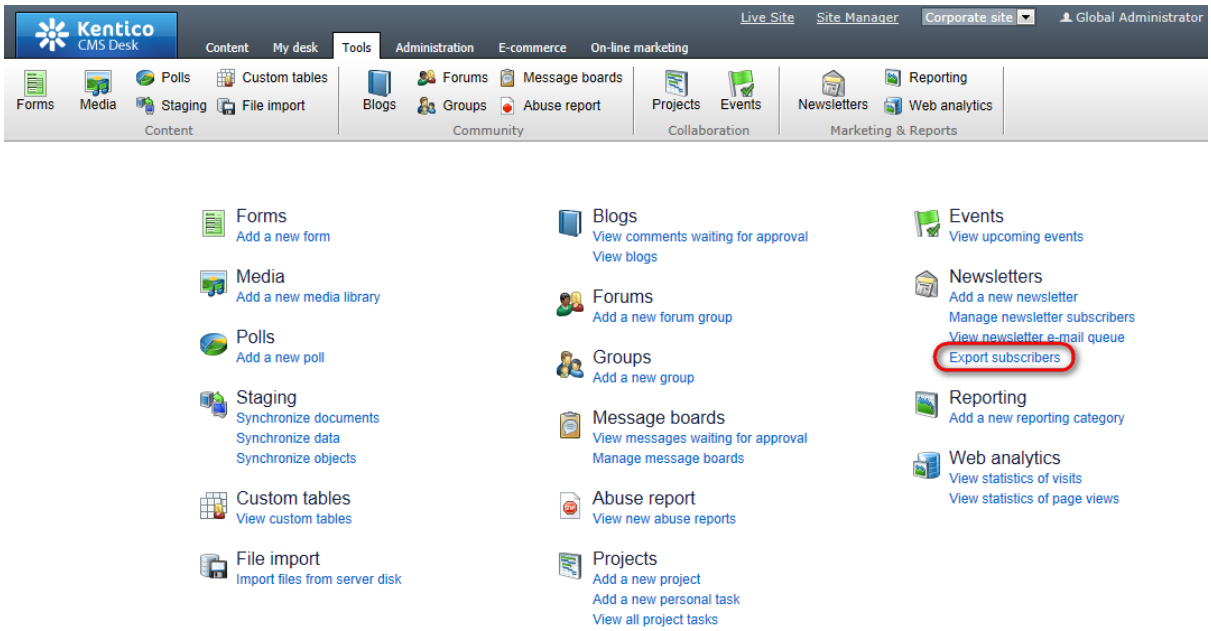


Please note

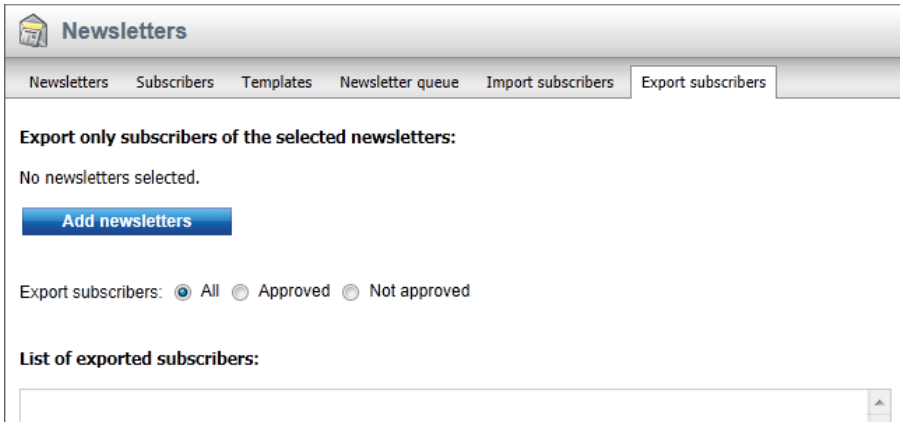
The value of the **Target URL** property must contain a *tab* query parameter whose value is the code name of the tab to which you want to link.
The code name of the tab is the same as the code name of the particular UI element.

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.

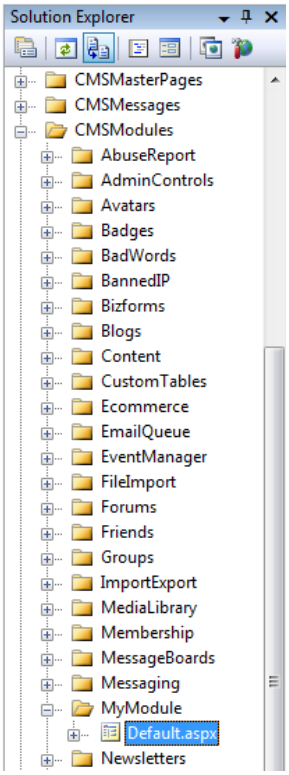
8.3 Developing custom modules

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");
}
```

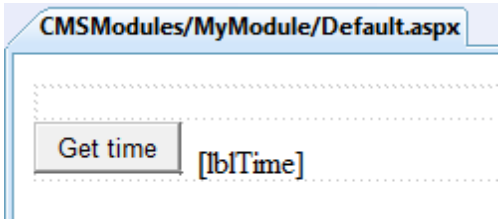
It checks if the current user has the **Read** permission for the *myprojects.mymodule* module.

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

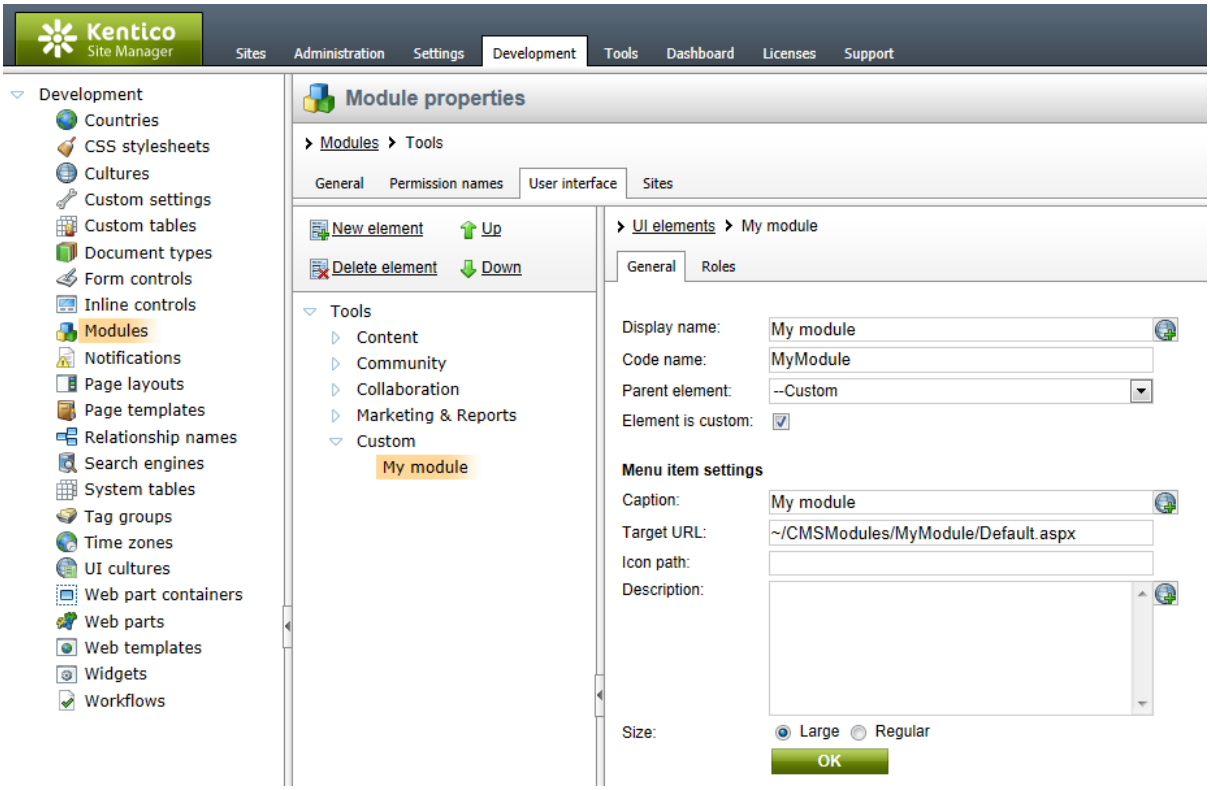
Click **OK**.

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 .

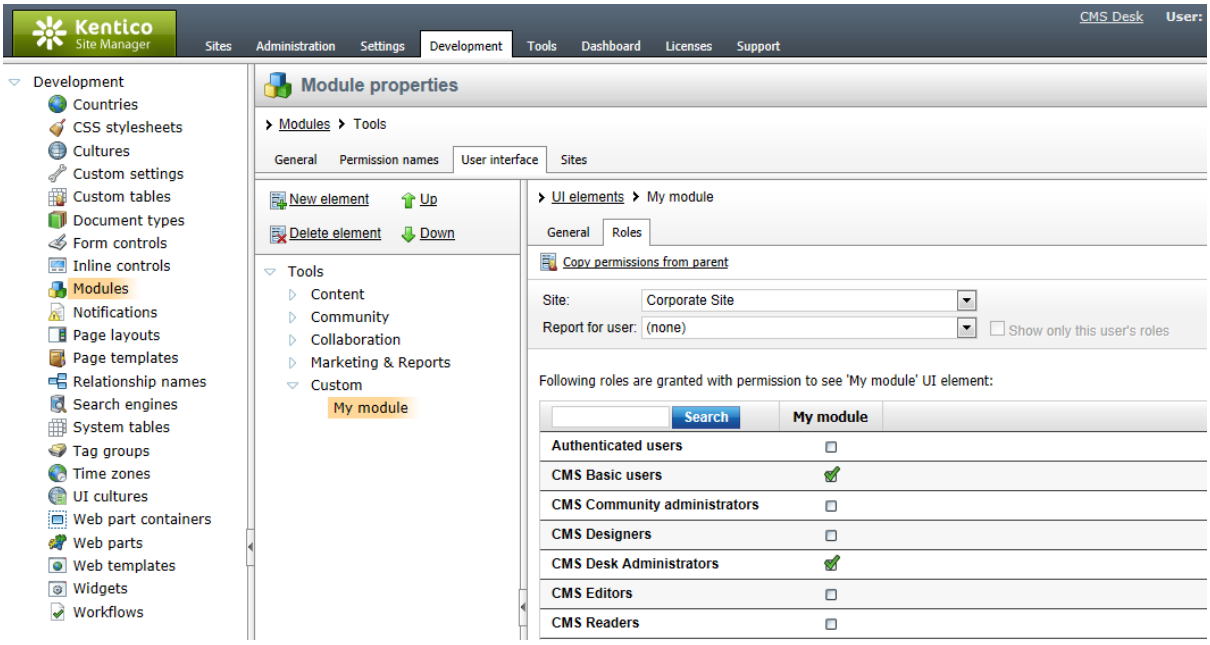
14. Go back to **Development -> Modules** and **Edit** () the **Tools** module. Switch to its **User interface** tab, select **Custom** in the Tools tree and click **New element**. Enter the following details:

- **Display name**: My module
- **Code name**: MyModule
- **Element is custom**: enabled
- **Caption**: My module
- **Target URL**: ~/CMSModules/MyModule/Default.aspx

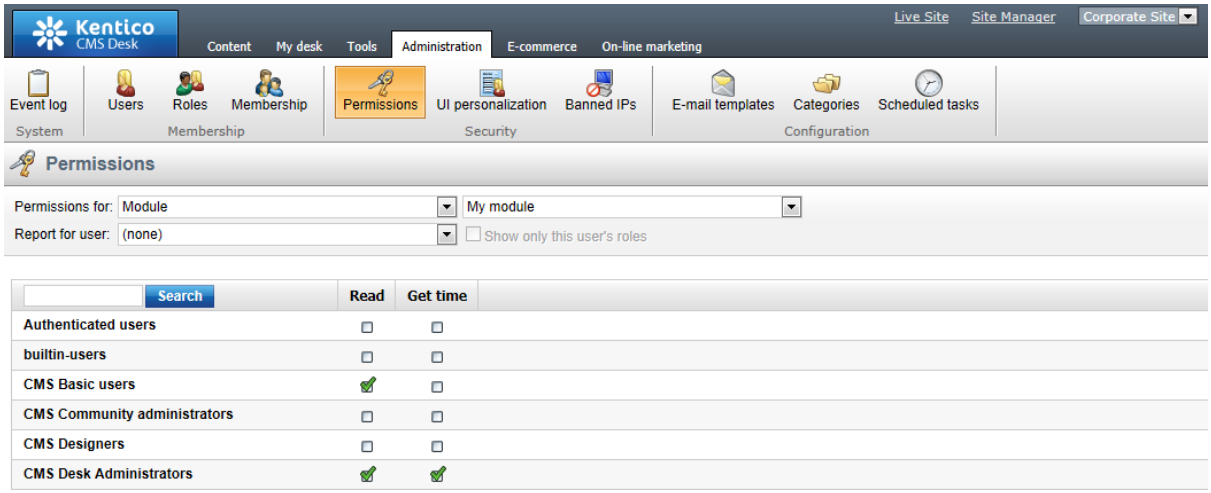
Click **OK**.



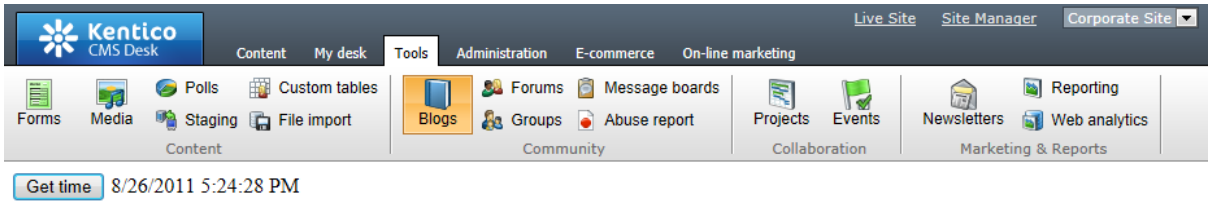
15. We will use UI personalization to enable access to this module only to members of certain roles. For this to work, UI personalization needs to be enabled. With the new UI element still selected, switch to the **Roles** tab and enable the element for members of the **CMS Basic users** and **CMS Desk Administrators** roles.



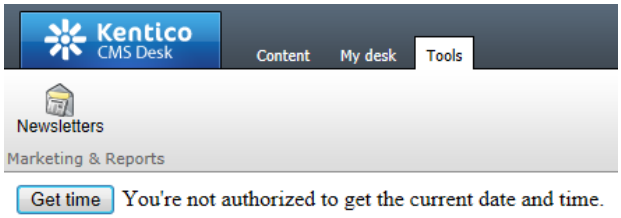
16. Go to **CMS Desk -> Administration -> Permissions** and choose **Permission for**: *Module* / *My module*. Grant the *Read* permission to the **CMS Basic users** role and both the *Read* and *Get time* permissions to the **CMS Desk Administrators** role:



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.

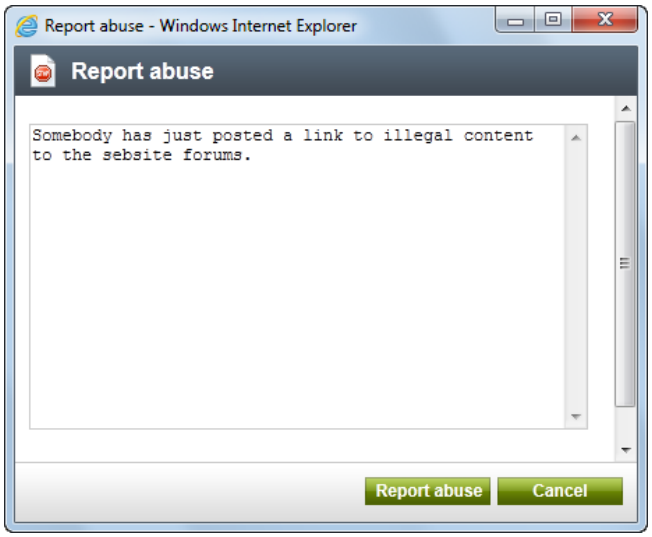
8.5 Abuse report

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.

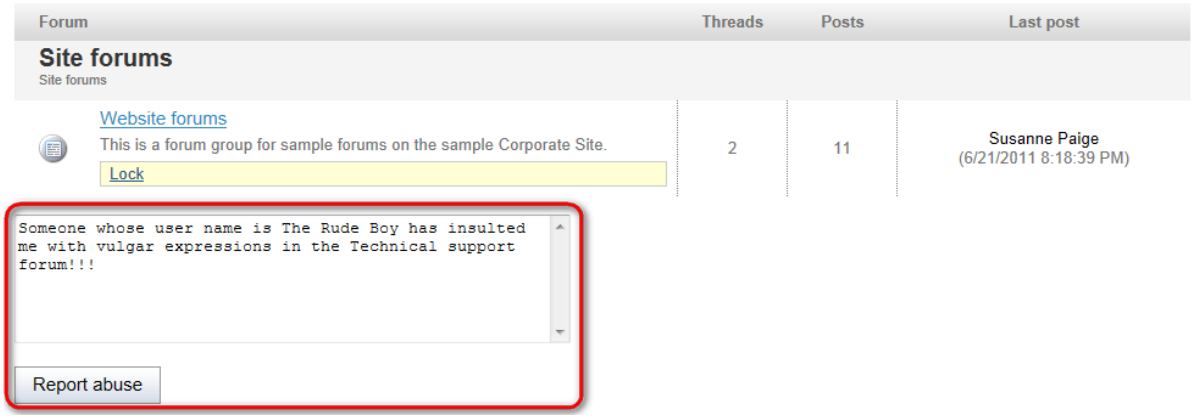
Abuse report

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.

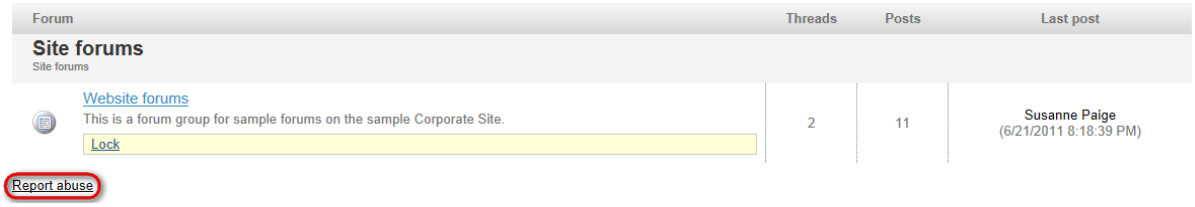
In the screenshot below, you can see the web part added below the forum group on the **/Forums** page of the sample Corporate Site. When a user types in some report text and clicks the **Report abuse** button, a report is logged in **CMS Desk -> Tools -> Abuse report**. Read the Abuse reports management topic for information on how reports can be managed in this section of CMS Desk.



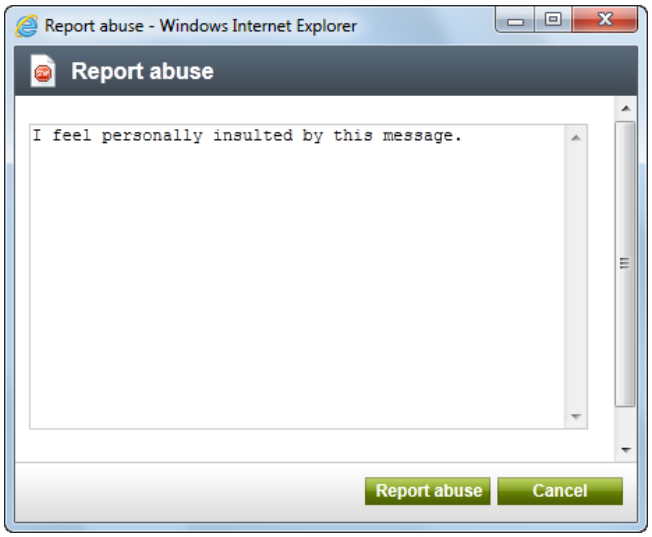
In-line abuse report

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.



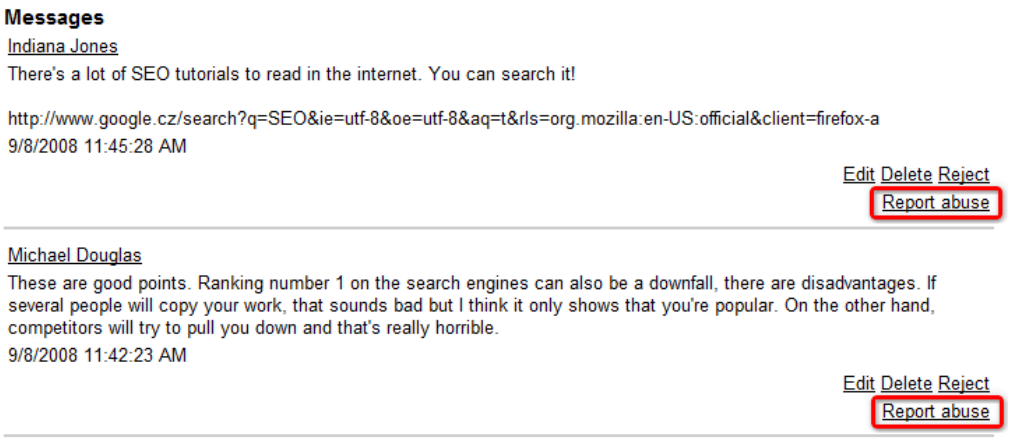
When a site visitor clicks the link, a dialog pops-up, letting the site visitors report abuse to the site administrators. Reports submitted via this web part are also logged in **CMS Desk -> Tools -> Abuse report** and can be managed there as described in the Abuse reports management topic.



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;"/>
```

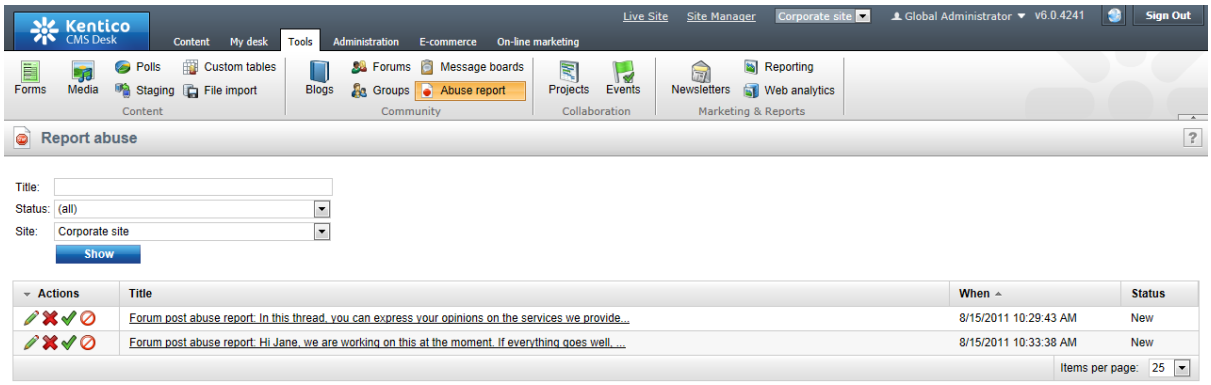
Let's take a closer look at the highlighted parts of the code. The highlighted line at the top is here to register the web part (actually control) so that it can be used in transformation code. As you can see, the web part code file is *~/CMSModules/AbuseReport/Controls/InlineAbuseReport.ascx*.

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

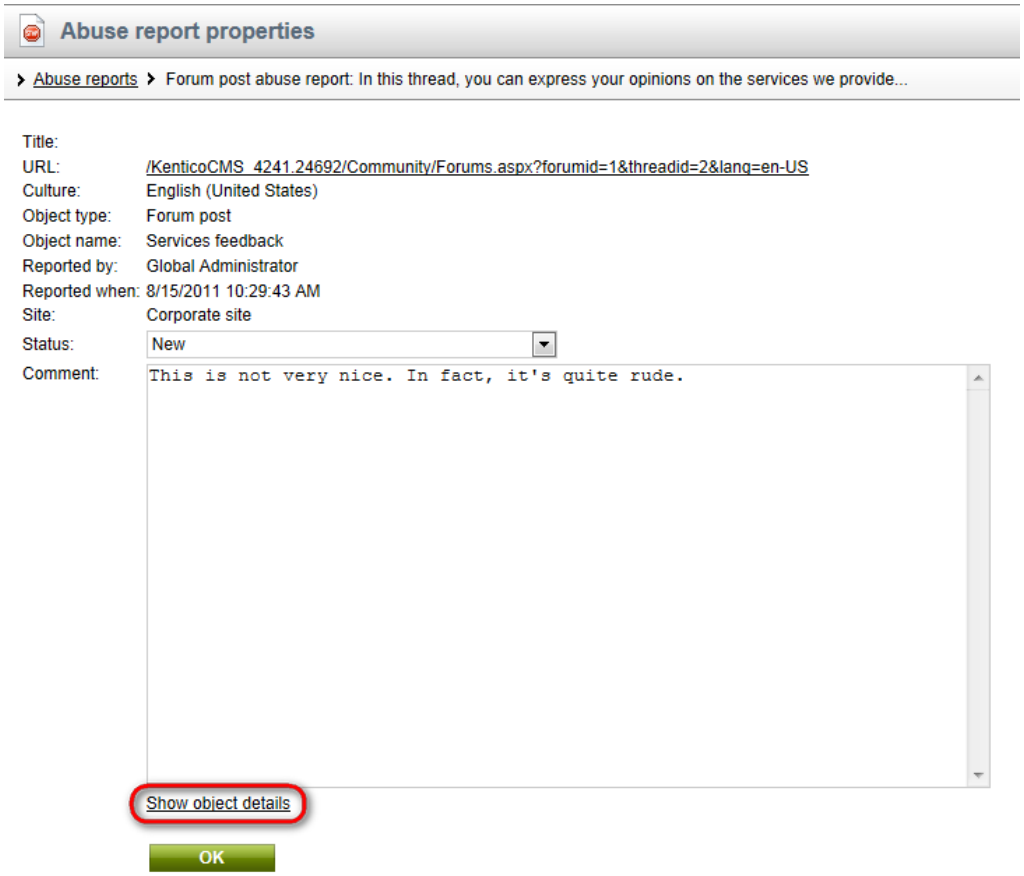
When editing ( ) a report, the following information is displayed:

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.

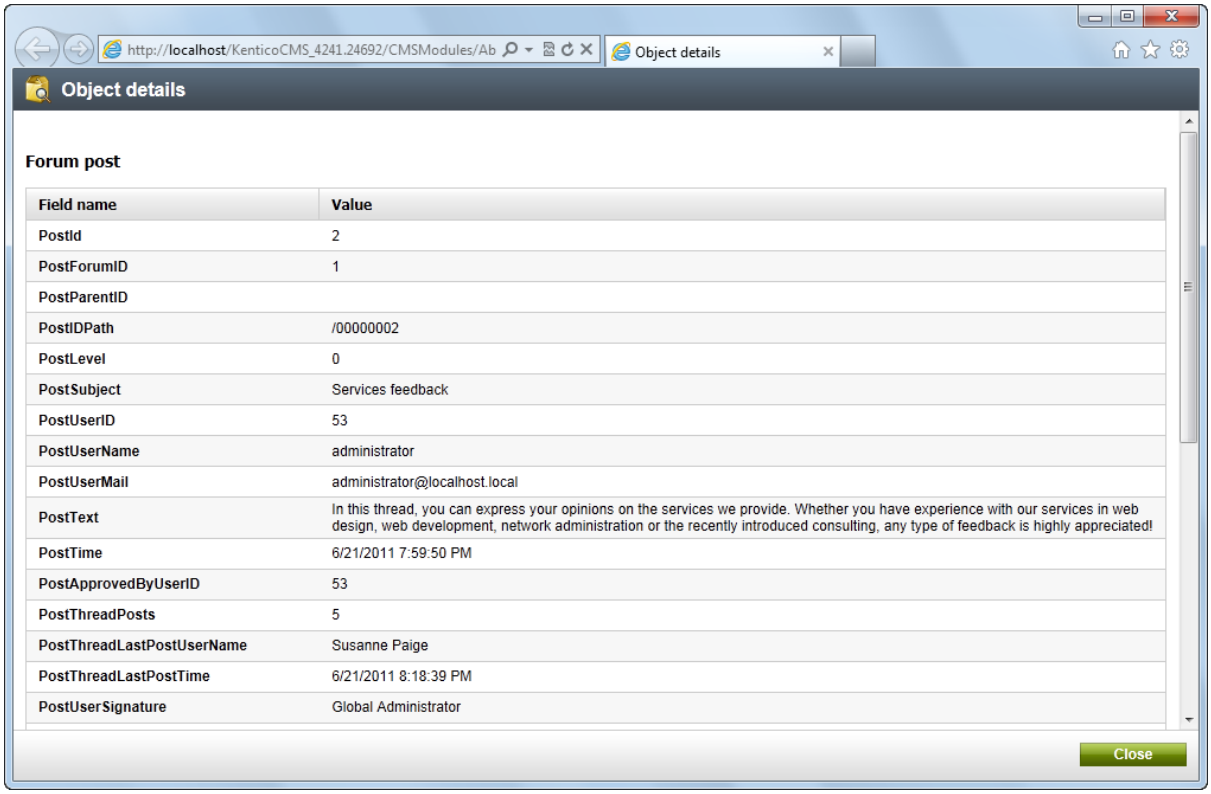
Object details

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, as highlighted in the screenshot above.



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:

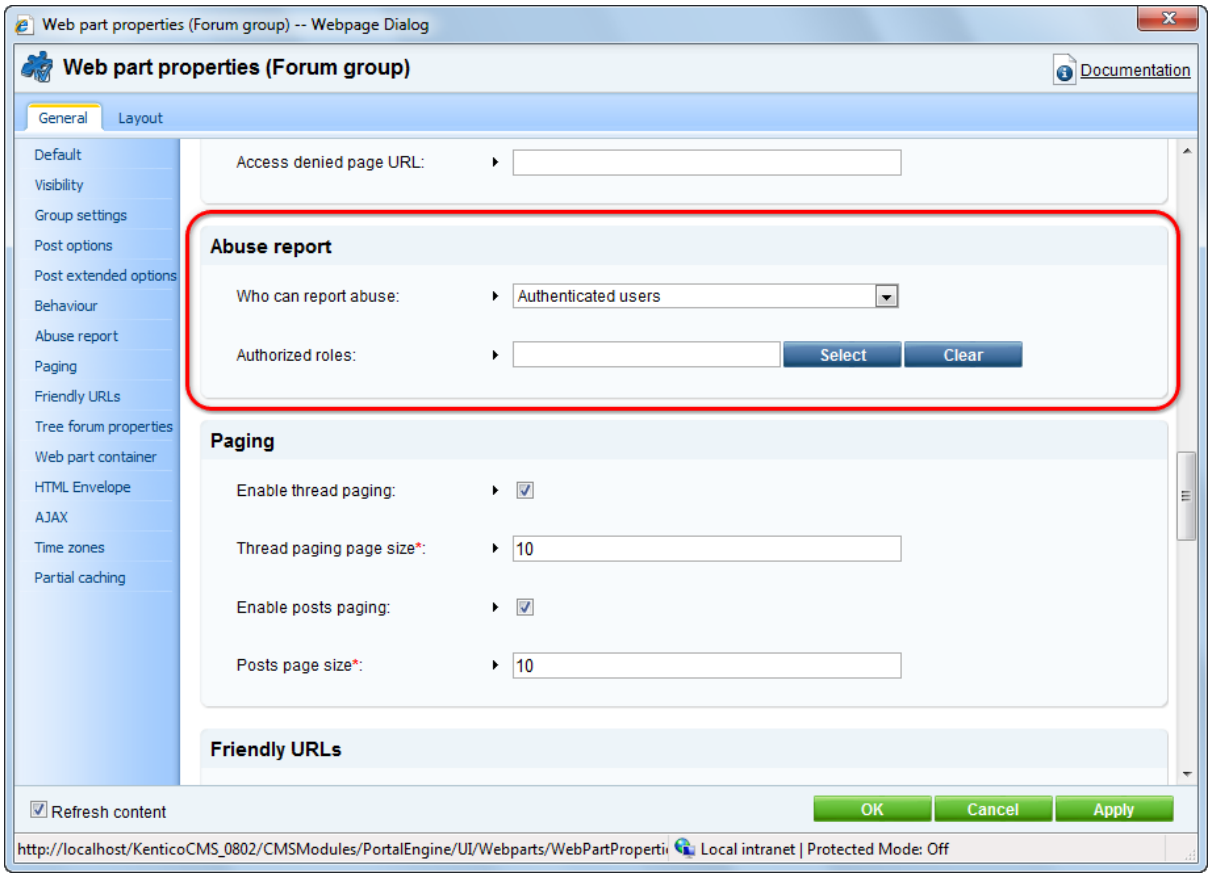
- **Blogs -> Comment view**
- **Community -> Group forum list**
- **Forums -> Forum group**
- **Forums -> Forum (Single forum - General)**
- **Forums -> Forum (Single forum - Tree layout)**
- **Forums -> Forum (Single forum - Flat layout)**

Each of these web parts has the **Abuse report** section in its properties, as depicted in the screenshot below. 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.

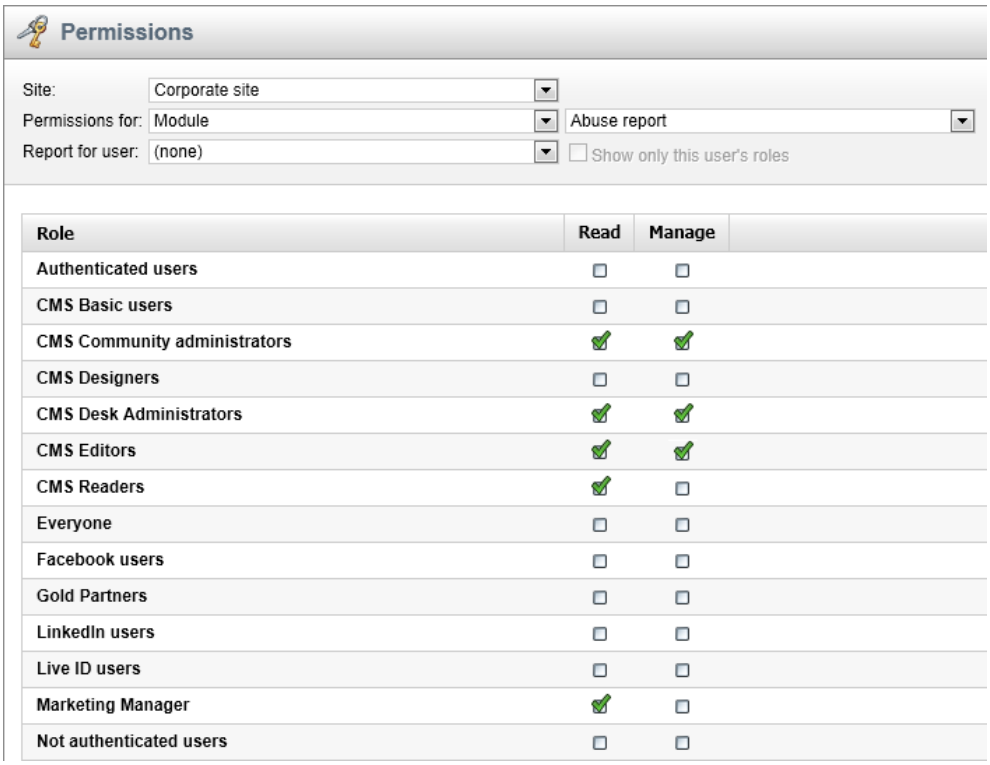


8.5.6 Security

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

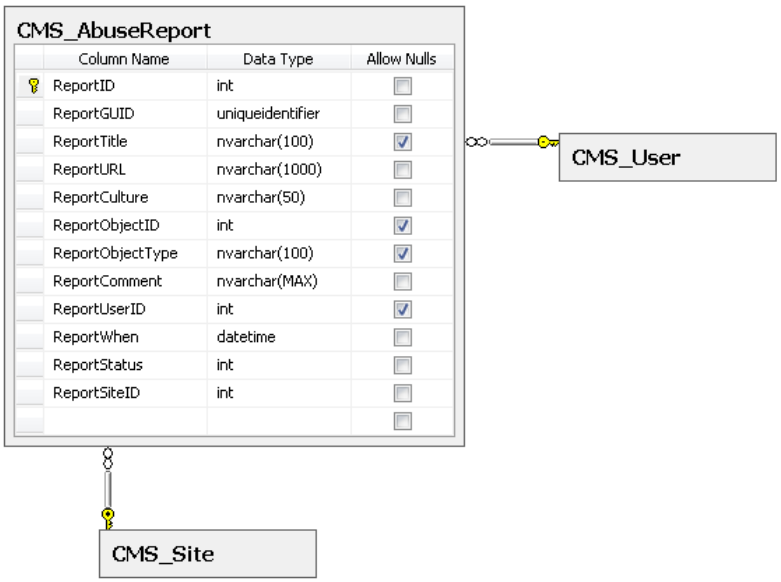
In this chapter, you will learn which database tables and API classes are used in the Abuse report module. You will also see the most common API examples.

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 respective sections of the UI:

- **Site manager -> Development -> Custom tables**
- **Site manager -> Development -> Document types**
- **Site manager -> Development -> System tables**

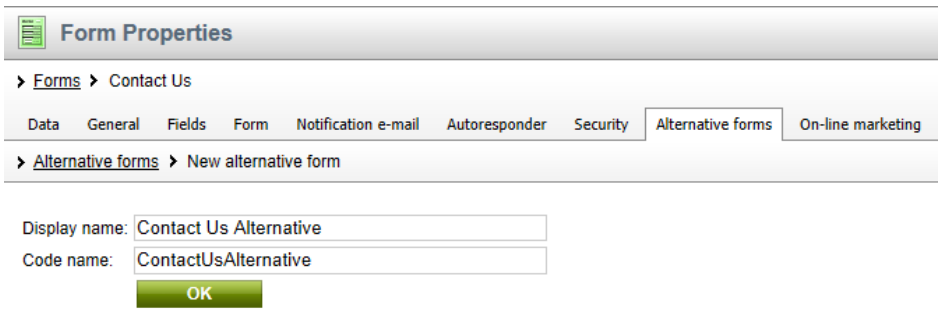
There, just choose to **Edit** () the particular item, switch to its **Alternative forms** tab and follow the instructions below, starting from the second step.

1. Go to **CMS Desk -> Tools -> Forms**. Choose to **Edit** () the **Contact us** form. The **Form Properties** screen will appear.

2. On **Form Properties**, switch to the **Alternative forms** tab and choose to **Create new form** ().

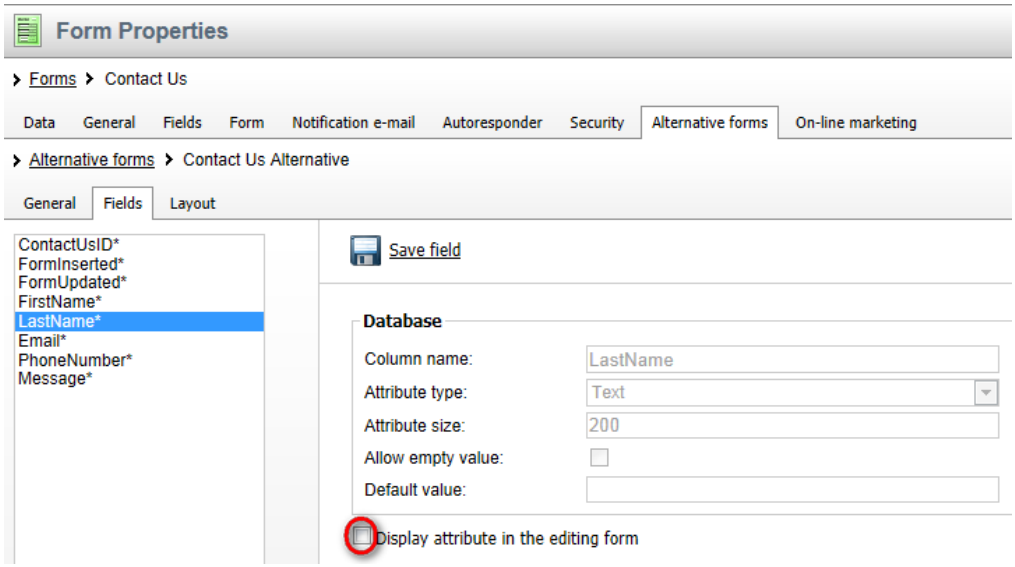
3. On the form that will be displayed, enter the following details and click **OK**.

- **Display name**: Contact Us Alternative
- **Code name**: ContactUsAlternative

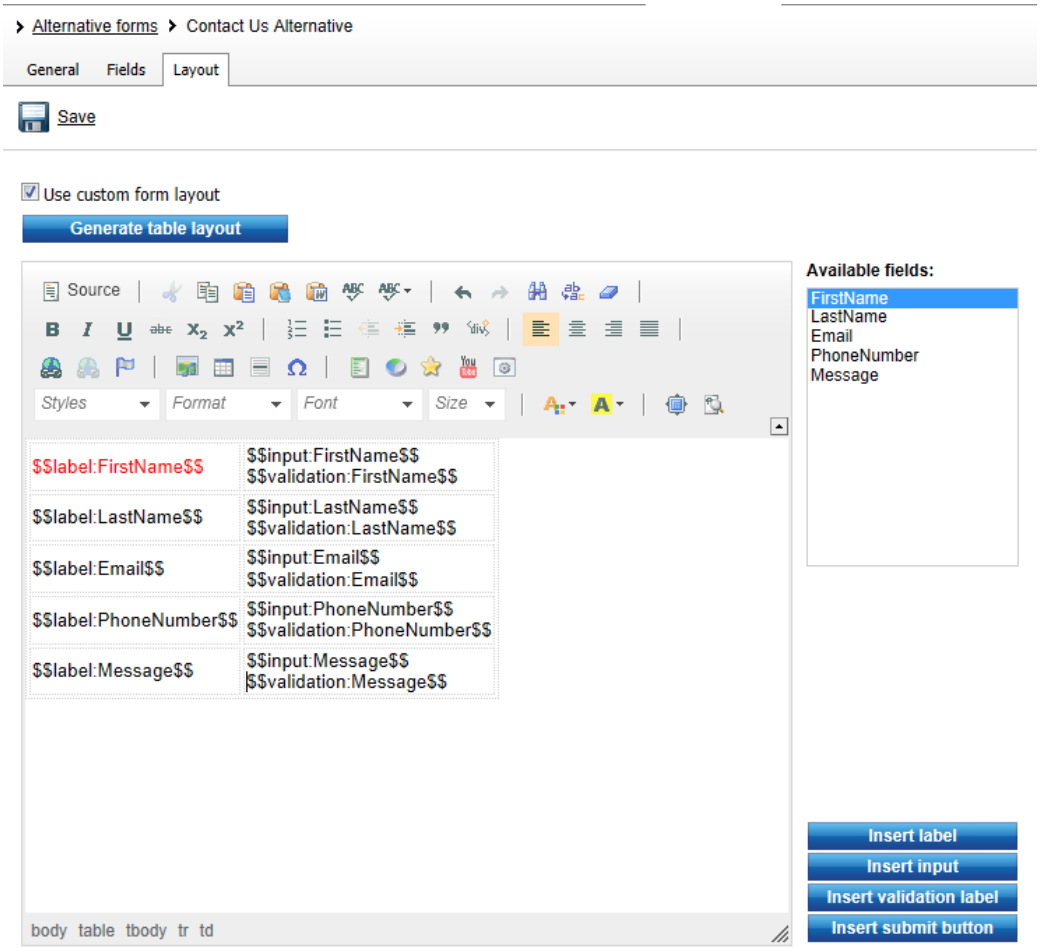


4. Switch to the **Fields** tab. As you can see, all fields present in the original form are present here as well and you can now make modifications to them. Let's disable the **LastName** field as an example. Select the field from the list on the left and on it's properties, uncheck the **Display attribute in the**

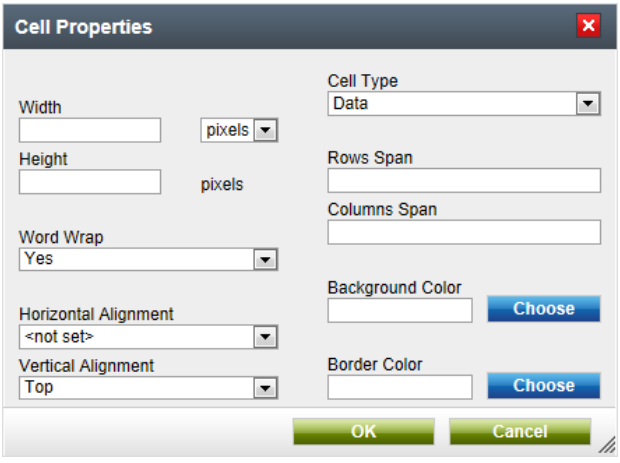
editing form check-box. Confirm by clicking **Save field**. Like this, you can modify any field in the form according to your needs.



5. Not only the fields, but also the layout of the form can be modified. Switch to the **Layout** tab and check the **Use custom layout** check-box. The layout editor will appear. 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. Table layout will be generated in the editing field below. Highlight the *$$label: FirstName$$* text in the first row and change its color to red.



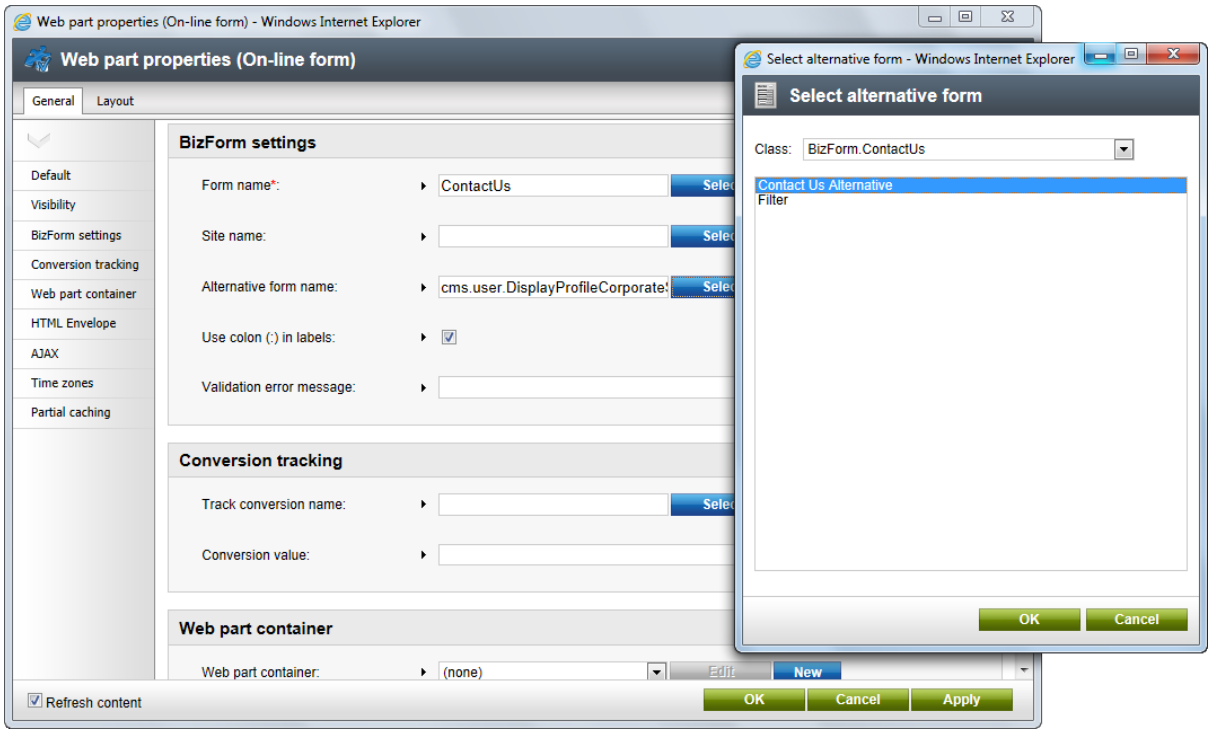
Also select the rows of the first table column, right click it and from the context menu, choose **Cell -> Cell properties**. In the displayed dialog, choose **Vertical alignment: Top** and click **OK**. This will make the labels to be aligned to their fields. Click **Save**.



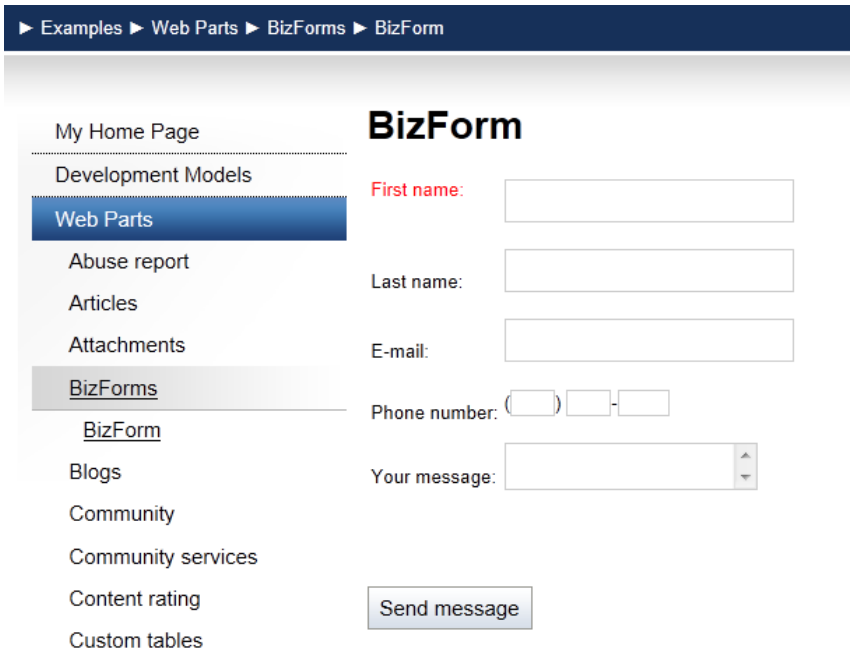
6. Let's take a look at what we've created. Switch to **CMS Desk -> Content**. From the content tree, select **Examples -> Web parts -> Forms -> On-line form**. As you can see, there is the original

version of the **Contact Us** form 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 choose to configure the web part properties. We will be concerned about the **Alternative form name** field. Click the **Select** button next to it and from the list, select our **Contact Us Alternative**. Click **OK**.



8. Now when you switch to the **Live site** mode, you should see the modified version as in the following screenshot:

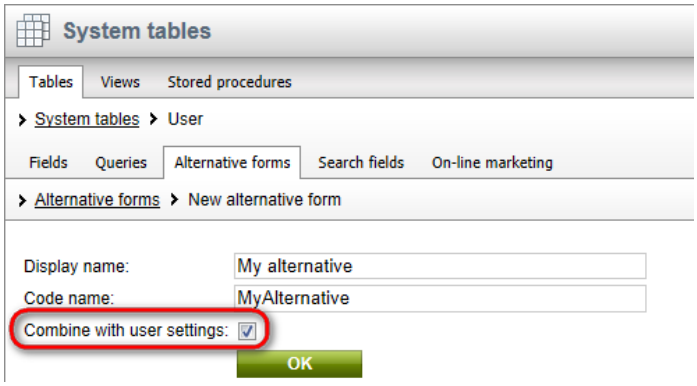


You have learned how to create an alternative form to an existing form and use it on your website.

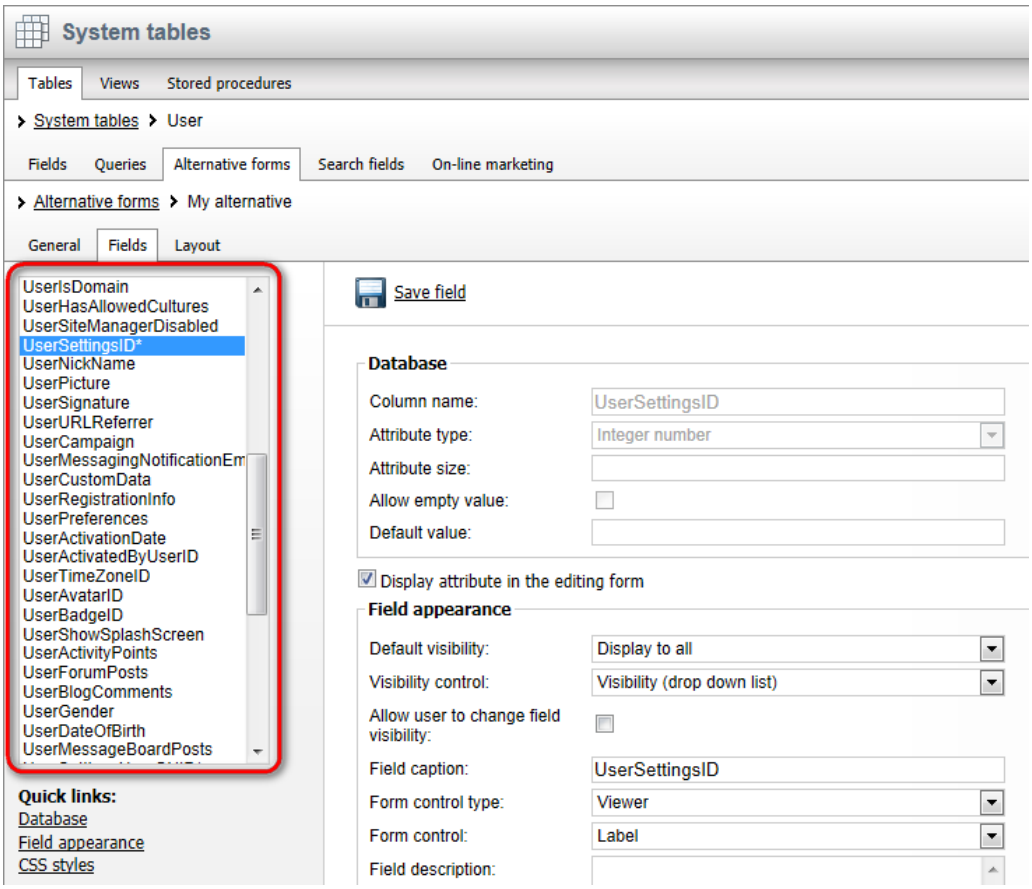
8.6.3 Joining two classes into one form

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 of the **User** system table, you have the option to check the **Combine with user settings** check-box.



2. Now if you switch to the **Fields** tab, you will see that besides the original fields contained in the **Users** table, fields from the **User - Settings** table are also available in the list of field, 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 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

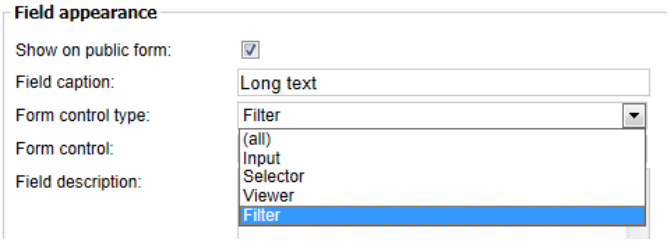
The number of records is 25 by default, while it can be modified by adding the following key into the *appSettings* section of your *web.config* file:

```
<add key="CMSDefaultListingFilterLimit" value="10" />
```

Filtering is possible based on all fields that store:

- Boolean (Yes/No) values
- Text
- Integer numbers
- Long integer numbers
- Decimal numbers

All that is required is to include the required fields in the alternative form and choose the **Form control type**: *Filter* for each field.

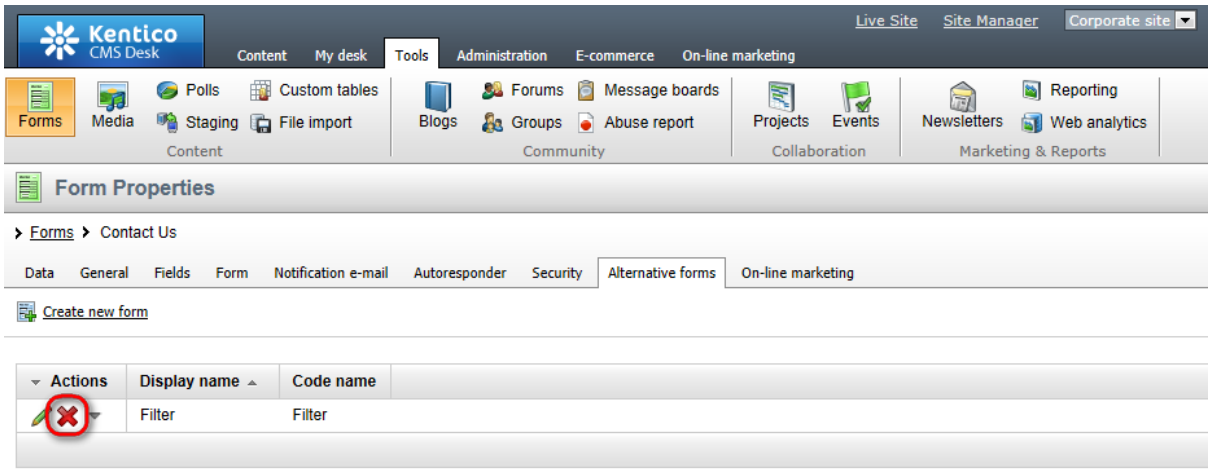


Example: Creating a form filter

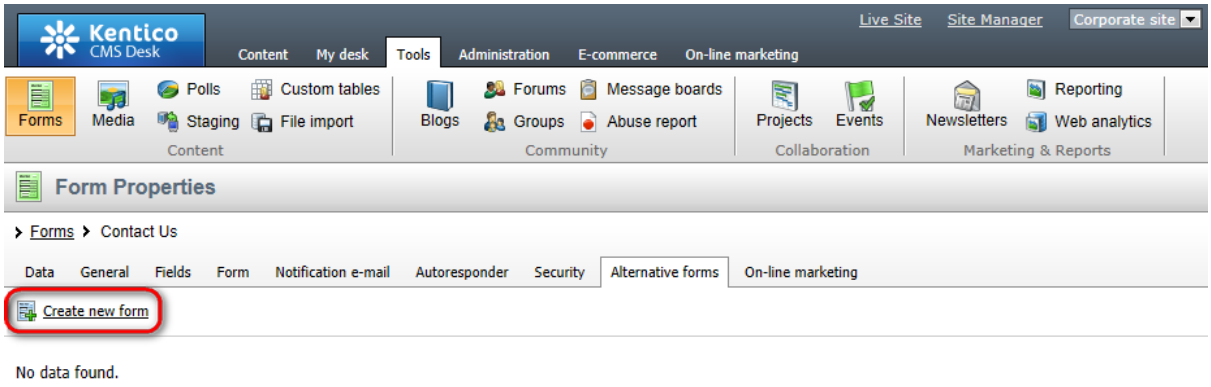
The following example will show you how to create a sample filter for the **Contact Us** form on the sample **Corporate Site**. The procedure demonstrated in the example can be used for any other form. Creation of custom table filters is also performed the same way.

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, so click **Delete** () next to it so that you can go through the rest of this example and create your own filter from scratch.



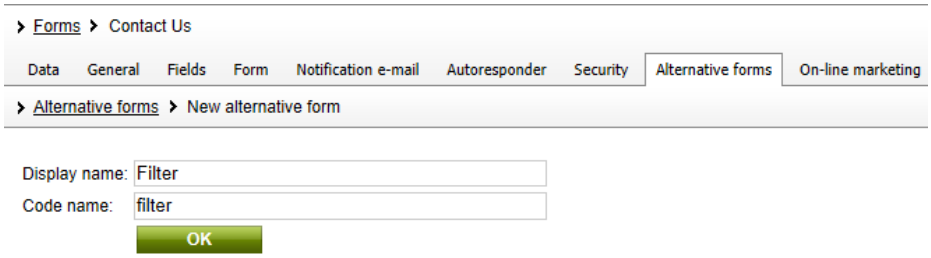
2. Once you have the form deleted, click **Create new form** to create a new one.



3. In the **New alternative form** dialog, enter the following details:

- **Display name**: Filter
- **Code name**: filter

and click **OK**.

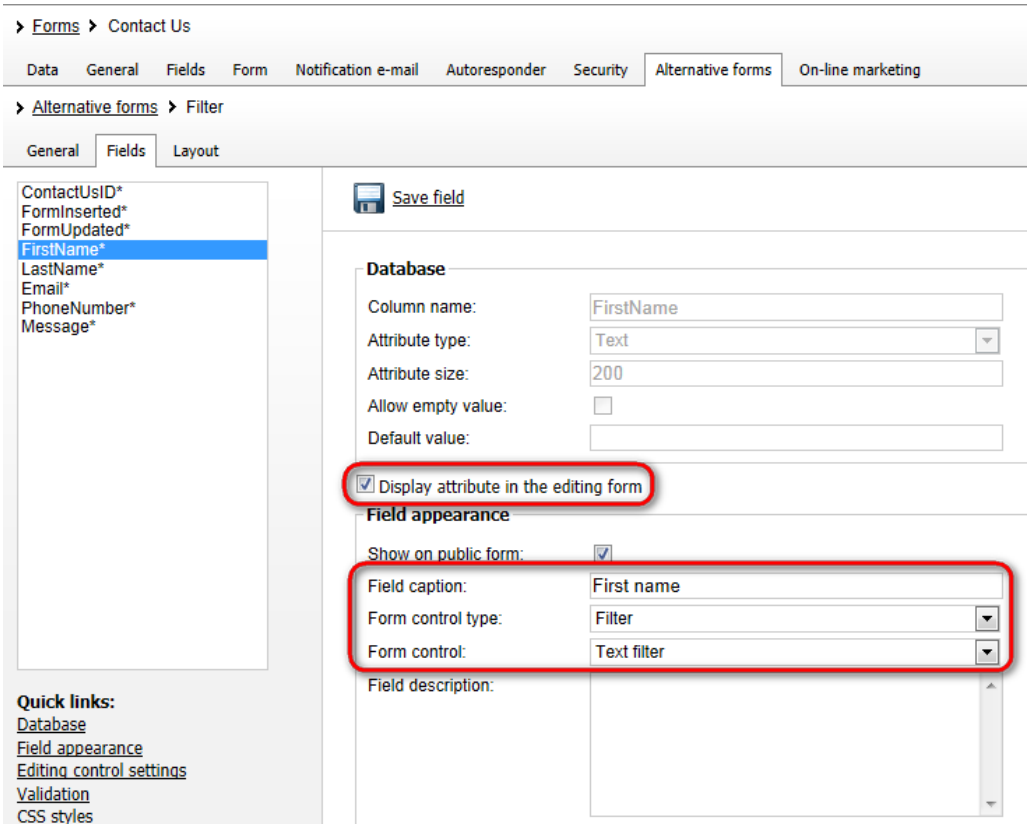


4. Now that the form is created, let's go to the most essential part of its configuration — let's switch to the **Fields** tab. In the listbox on the left, you should see all fields of the form. The first three of them are system fields that we do not want in the form, so select the fourth one - **FirstName**. 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 form.

- **Field caption**: First name; caption displayed next to the filter field.
- **Form control type**: *Filter*; ensures that the field will be a filter field.
- **Form control**: *Text filter*; the only filter type available for a text field.

Click **Save field** when you are finished.



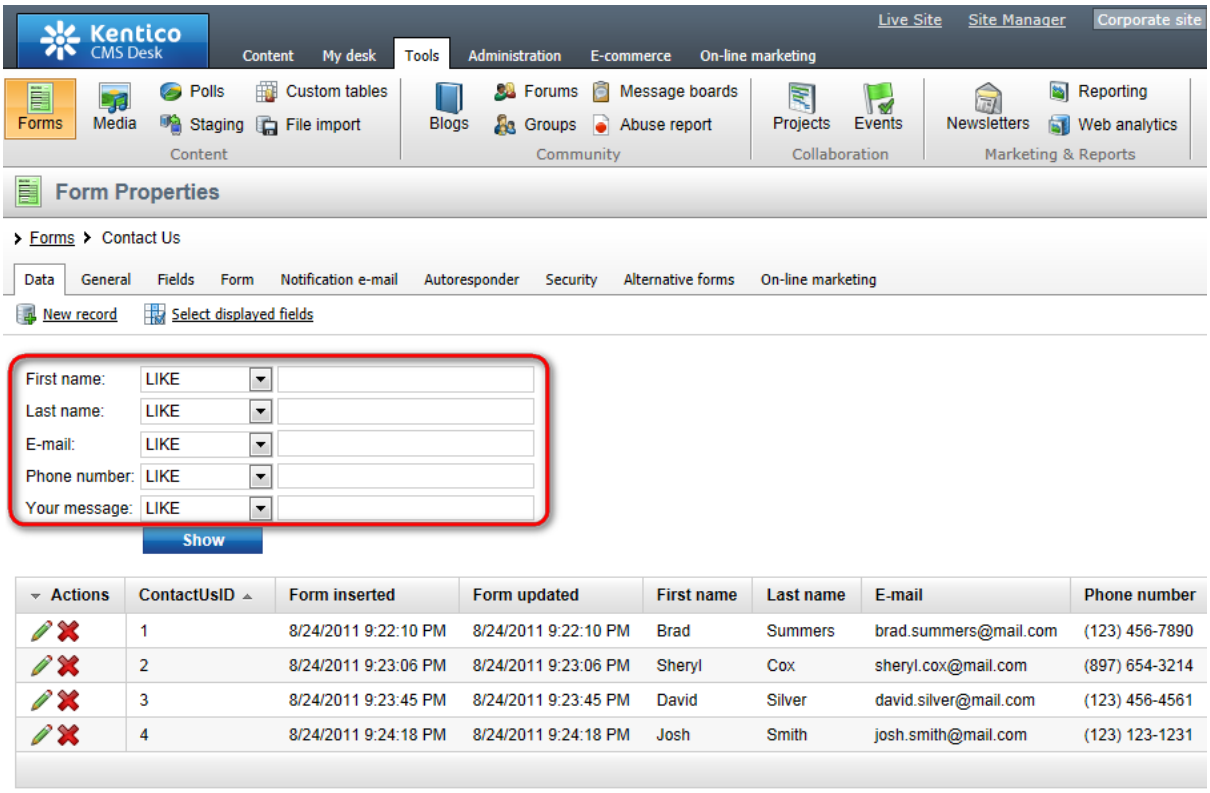
5. Repeat the configuration explained in step 4 for the **LastName**, **Email**, **PhoneNumber** and **Message** fields. This will ensure that all the fields will be included in the filter.

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 particular parameters to see that filtering really works.

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 create less of them and use the **CMSDefaultListingFilterLimit** *web.config* key to lower the filter display 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, in forum posts, 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 either choose an avatar from a gallery of pre-defined avatars if this option is enabled or create a custom avatar by uploading their own image from a file on their local disk. All common image formats are supported, including animations.

Unlike pre-defined avatars, custom avatars can't be selected by other users and they are deleted from the system if the user they were uploaded by changes her avatar.

Groups can 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.

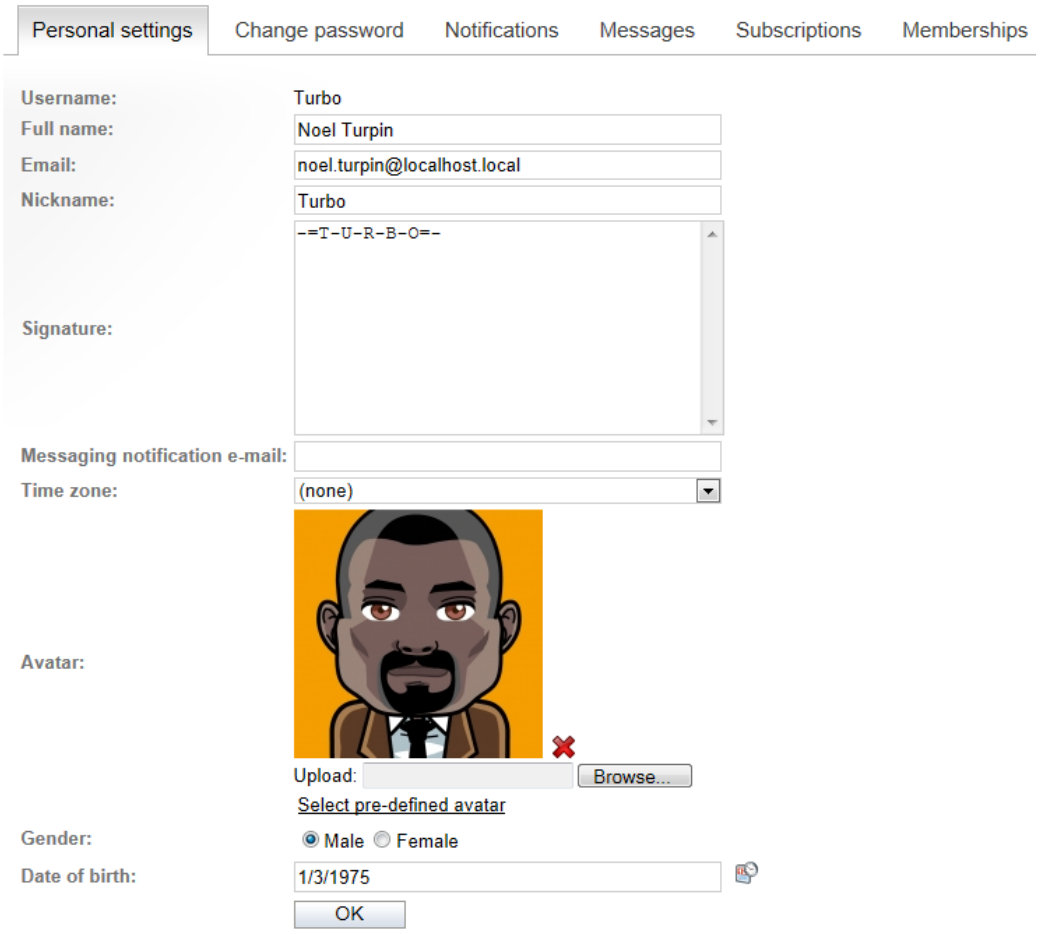
There are several ways for users or group administrators to add or change avatars. See the Changing user avatars or Changing group avatars topics for more details.

Site administrators can manage all avatars or change their settings. This is described in the Managing avatars and Settings topics.

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, the default avatar will be assigned to them. After that, they can change thier avatar using the **My account** web part. It can be done on its **Personal settings** tab, as you can see in the screenshot below.



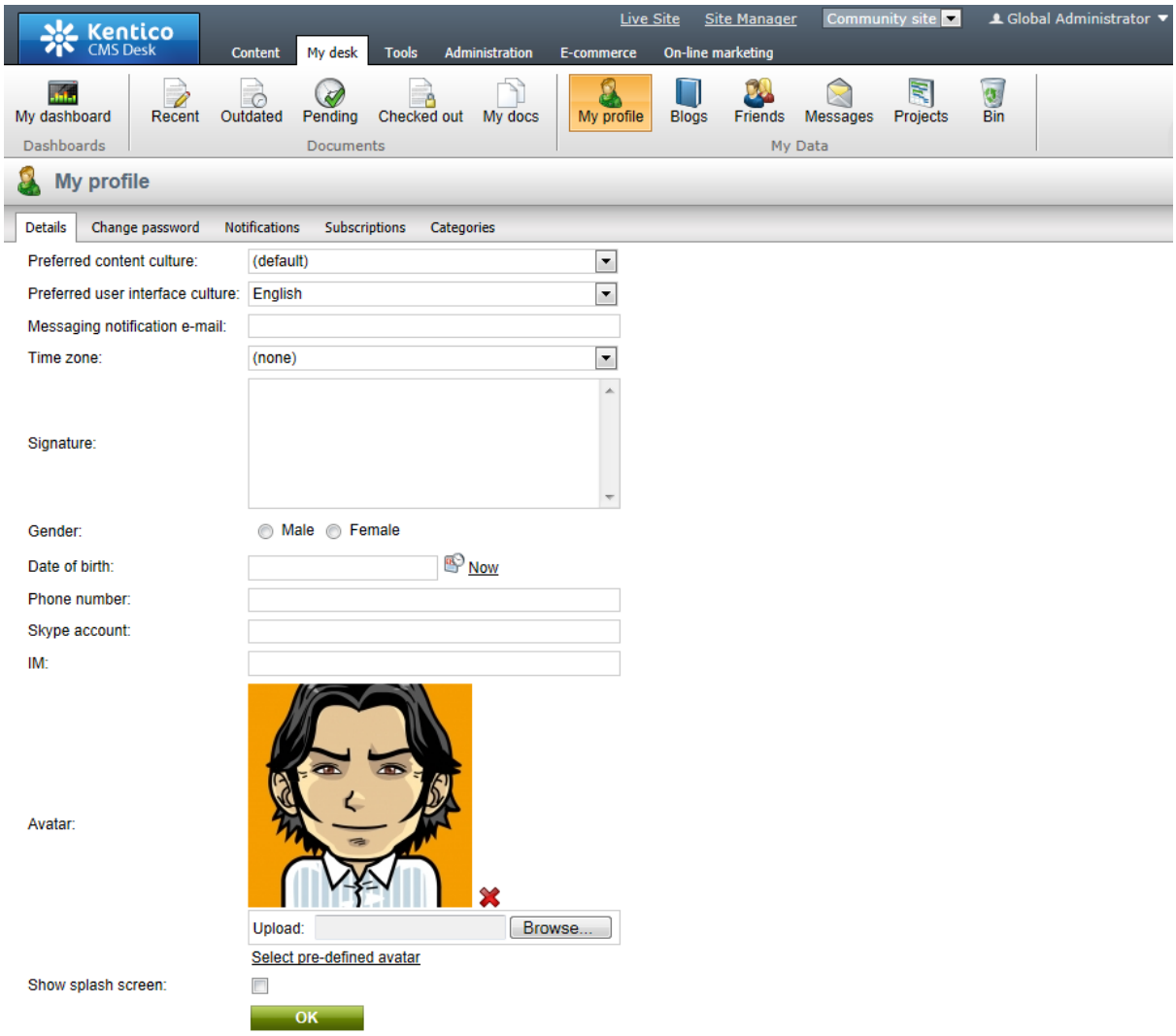
Users can **Delete** (✖) the avatar or **Upload** a custom one from a file. If pre-defined avatars are enabled in site settings, users can also click the **Select pre-defined avatar** link, which displays a gallery of pre-defined avatars.

You can find a live example of this on our **Community starter site**. Just log-in as some of the pre-defined users (e.g. *Turbo* with blank password) and click the **Edit my profile** link in the **Shortcuts** menu on the right.

Changing user avatar in CMS Desk

Users with access to **CMS Desk** can change their avatars in **My Desk -> Account -> Details**. It can be done the same way as described above. 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**.

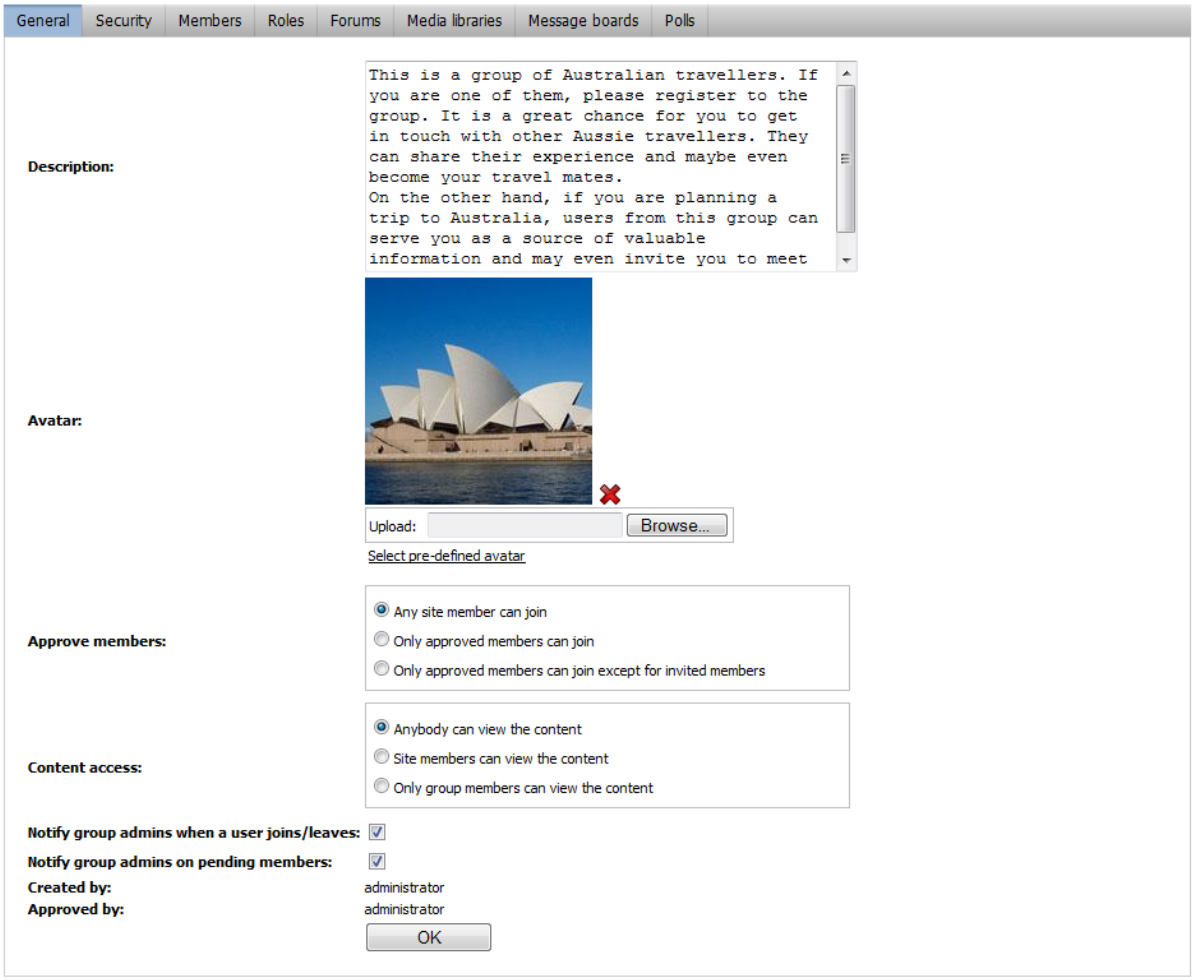


8.7.3 Changing group avatars

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 pre-defined avatars are enabled in site settings, users can also click the **Select pre-defined avatar** link, which displays a gallery of pre-defined avatars.

You can find a live example of this on the **Community starter site**. Sign-in as some 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 in the **My groups** section. Click it, and what you can see in the screenshot below will be displayed to you.

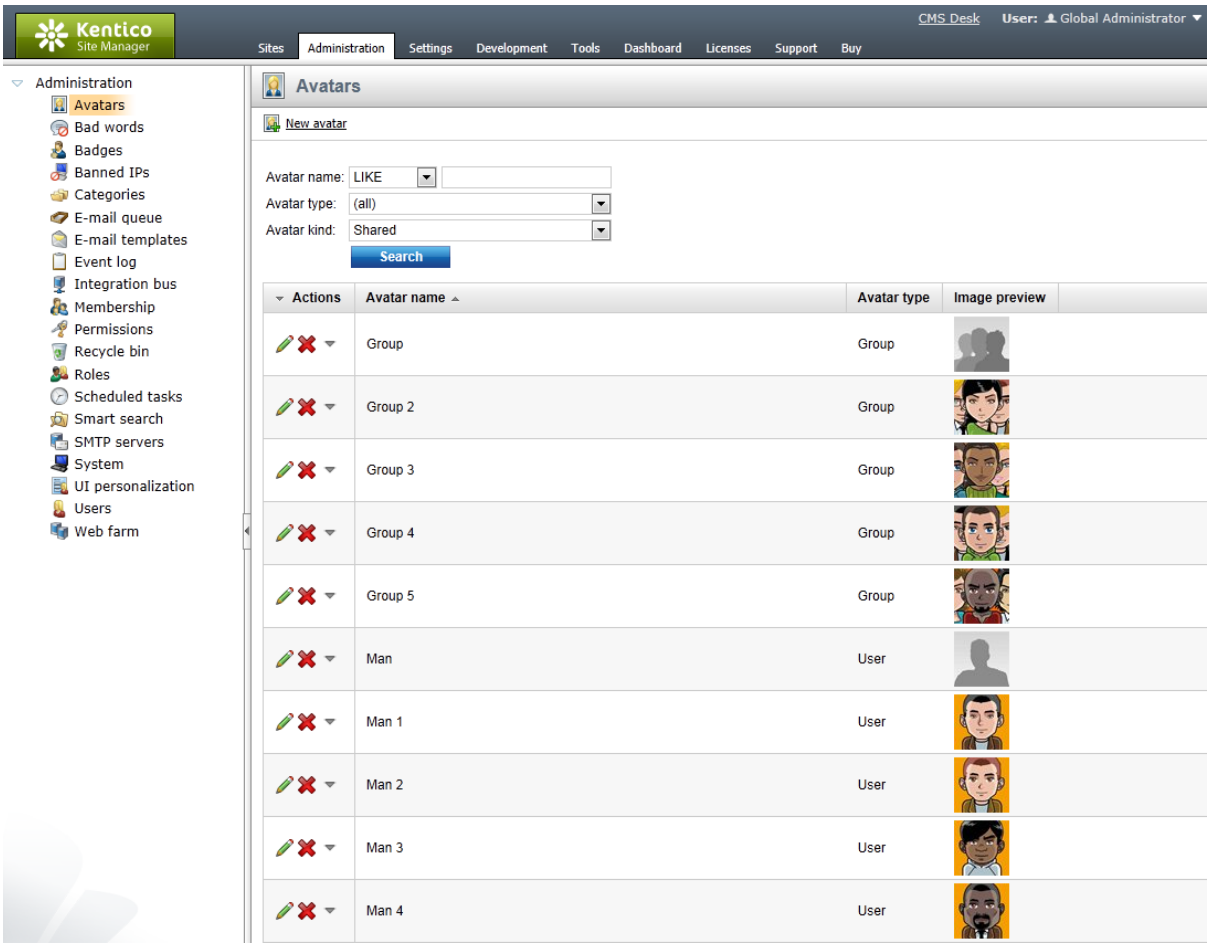


8.7.4 Managing avatars

The administration interface for avatar management is located in **Site Manager -> Administration -> Avatars**.

You can filter displayed avatars 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 pre-defined 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** ( ) listed avatars.



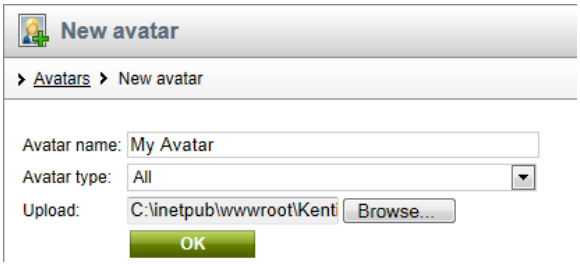
Creating pre-defined avatars

1. New **pre-defined avatars** can be created using the **New avatar** link at the top part of the page. Click it.

2. You will be asked to enter the following details:

- **Avatar name** - name of the avatar
- **Avatar type** - choose if the avatar can be used for users, groups or both
- **Upload** - enter the path to the avatar image on your local machine or click the **Browse** button to browse and locate the file

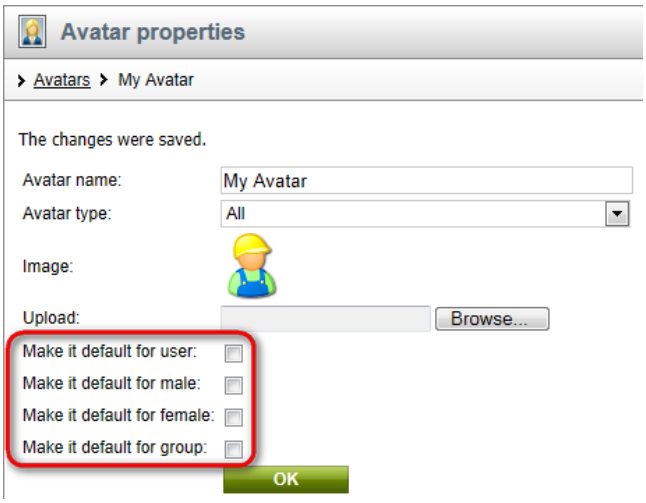
When entered, click **OK** to proceed.



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
- **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

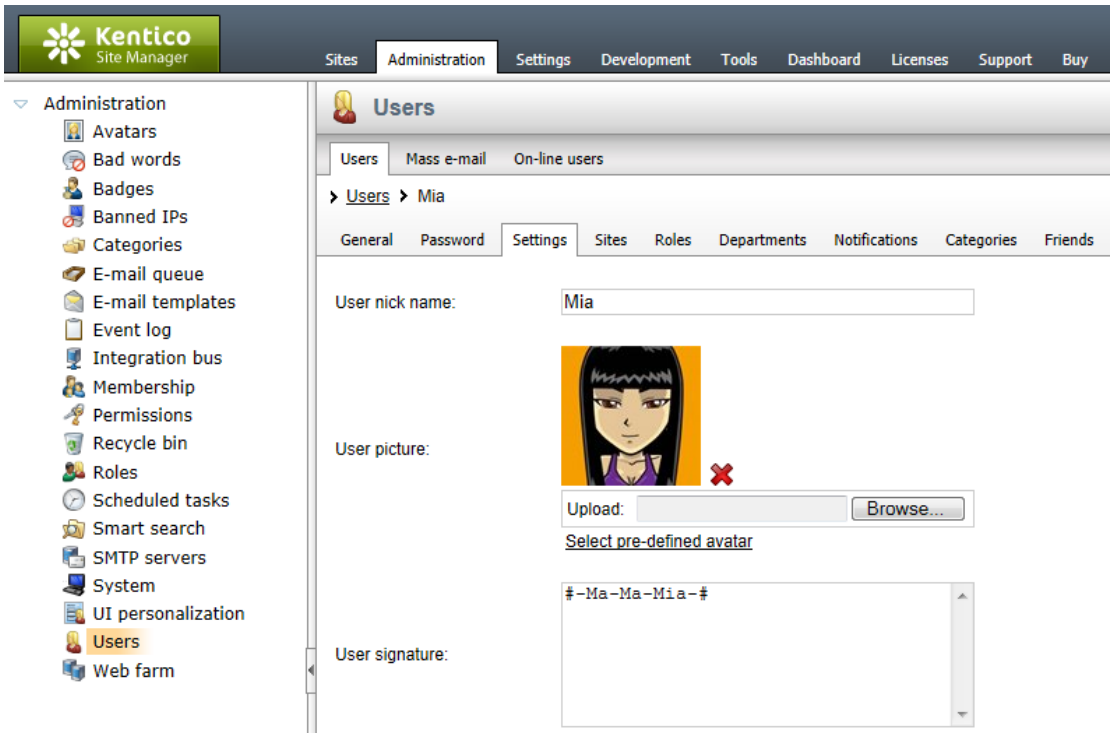
Default avatars will be assigned to newly created users or groups automatically when the user or group is created.



Administrating users' avatars

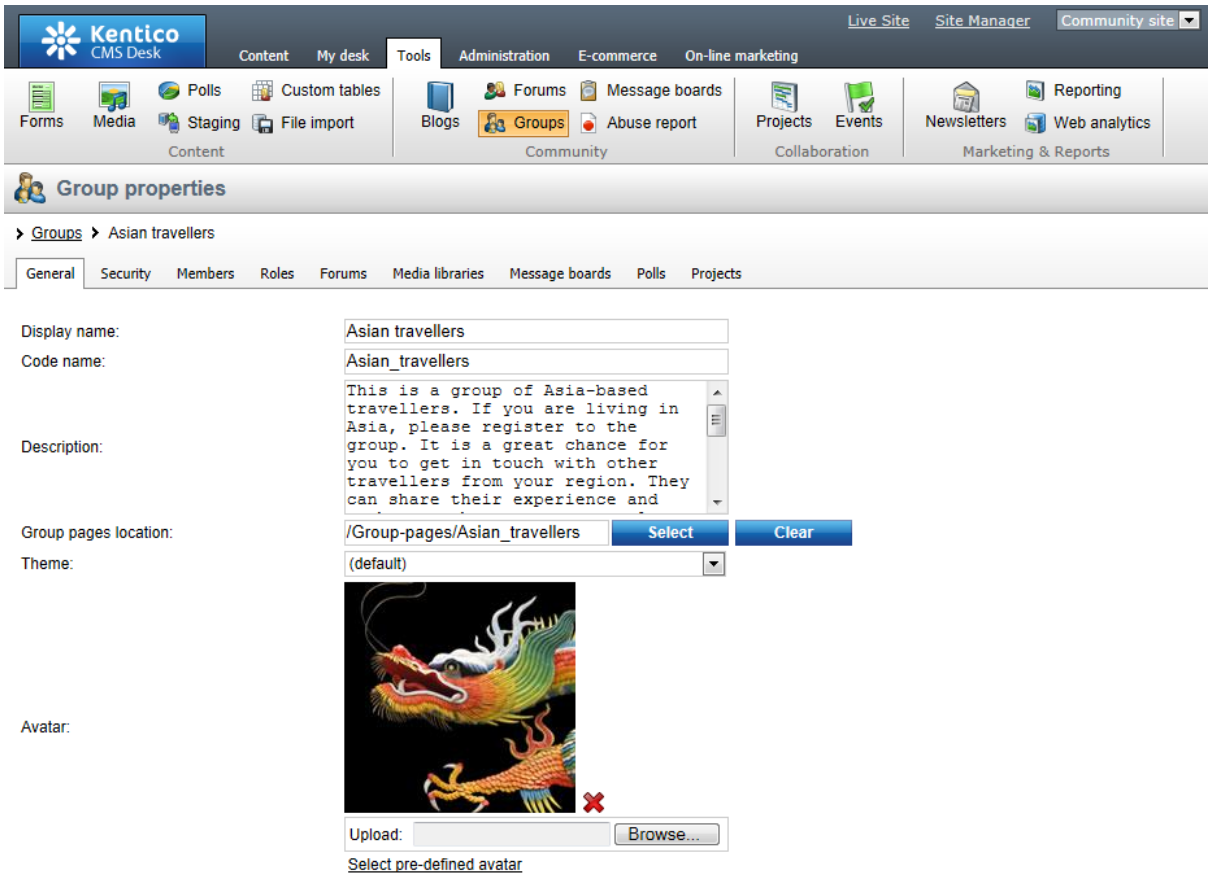
Site administrators can change the avatar of any user. If you go to **Site Manager -> Administration -> Users**, choose to **Edit** () some of the users in the list and switch to their **Settings** tab, you should see the user's avatar in the **User picture** field, as you can see in the screenshot below. 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 pre-defined avatars (if this is enabled in site settings).



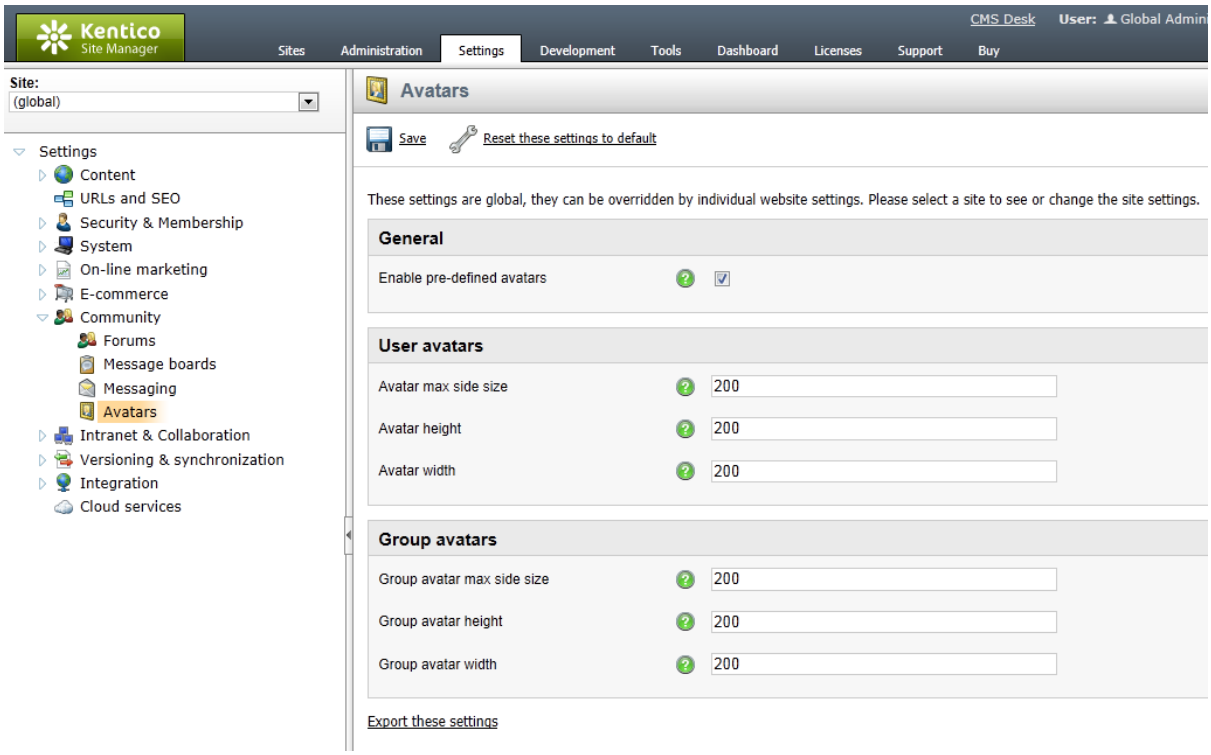
8.7.5 Settings

Settings of the Avatars module can be done in **Site Manager -> Settings -> Community -> Avatars**. The following settings are available:

- **Enable pre-defined avatars** - if checked, default avatars can be selected when selecting a user's or group's avatar; if unchecked, only custom uploaded avatars can be used

- **Avatar max side size** - maximal size of **user avatars**; if one or both sides of the image are longer, the image will be resized so that the longer side's size matches the entered value; if 0 is entered, *Avatar height* and *Avatar width* values will be used instead
- **Avatar height** - if *Avatar max side size* is set to 0, images will be resized to this height
- **Avatar width** - if *Avatar max side size* is set to 0, images will be resized to this width

- **Group avatar max side size** - maximal size of **group avatars**; if one or both sides of the image are longer, the image will be resized so that the longer side's size matches the entered value; if 0 is entered, *Group avatar* height and *Group avatar width* values will be used instead
- **Group avatar height** - if *Group avatar max side size* is set to 0, images will be resized to this height
- **Group avatar width** - if *Group avatar max side size* is set to 0, images will be resized to this width



8.7.6 Avatars internals and API

8.7.6.1 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.

The Displaying avatars in transformations topic contains examples of how avatars can be displayed in transformations.

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.

8.7.6.2 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.

CMS_Avatar

	Column Name	Data Type	Allow Nulls
🔑	AvatarID	int	☐
	AvatarName	nvarchar(200)	☑
	AvatarFileName	nvarchar(200)	☐
	AvatarFileExtension	nvarchar(10)	☐
	AvatarBinary	varbinary(MAX)	☑
	AvatarType	nvarchar(50)	☐
	AvatarIsCustom	bit	☐
	AvatarGUID	uniqueidentifier	☐
	AvatarLastModified	datetime	☐
	AvatarMimeType	nvarchar(100)	☐
	AvatarFileSize	int	☐
	AvatarImageHeight	int	☑
	AvatarImageWidth	int	☑
	DefaultMaleUserAvatar	bit	☑
	DefaultFemaleUserAv...	bit	☑
	DefaultGroupAvatar	bit	☑
	DefaultUserAvatar	bit	☑
			☐

8.7.6.3 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 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.

8.7.6.4 API examples

8.7.6.4.1 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;
```

8.7.6.4.2 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);
}
```

8.7.6.4.3 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;
}
```

8.7.6.5 Displaying avatars in transformations

User avatars

User avatars can also be displayed in transformations by using the *<%# GetUserAvatarImage(...) %>* method. Here are some examples of how it can be used:

The following method returns the image tag of 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:

```
<%# GetUserAvatarImage(50, HTMLEncode(GetNotEmpty("UserNickname;UserName"))) %>
```

The following method 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(Eval("UserID"), 200, 0, 0, Eval("UserName")) %>
```

The following method returns the image tag of 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(Eval("UserAvatarID"), Eval("UserID"), 50, Eval("UserName"))
%>
```

The following method returns the image tag of 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*:

```
<%# GetUserAvatarImage(Eval("UserAvatarID"), Eval("UserID"), 0, 40, 45, Eval(
"UserName")) %>
```

Group avatars

Group avatars can by displayed using the *<%# GetGroupAvatarImage(...) %>* method. Here are some examples:

The following method returns the image tag of the group avatar contained in 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(50, Eval("GroupDisplayName", true)) %>
```

The following method returns the image tag of the group avatar with ID *GroupAvatarID*, with a maximum *sidesize* of 50 px. and *Alt tag* equal to the group's *Display name*:

```
<%# GetGroupAvatarImage(Eval("GroupAvatarID"), 50, Eval("GroupDisplayName", true))
%>
```

The following method returns the image tag of 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*:

```
<%# GetGroupAvatarImage(Eval("GroupAvatarID"), 0, 40, 45, Eval("GroupDisplayName",
true)) %>
```

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 style="border-width: 0px;" alt="Andy" src="/CMSModules/Avatars/CMSPages/
GetAvatar.aspx?avatarguid=42f9d97e-b0c8-43a2-8a35-62dfec5ae64a&amp;maxsidesize=50"
/>
```

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 - filters forum posts
- Blogs - filters blog comments
- Messaging - filters messages sent via the module
- Message boards - filters messages posted on particular boards

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.

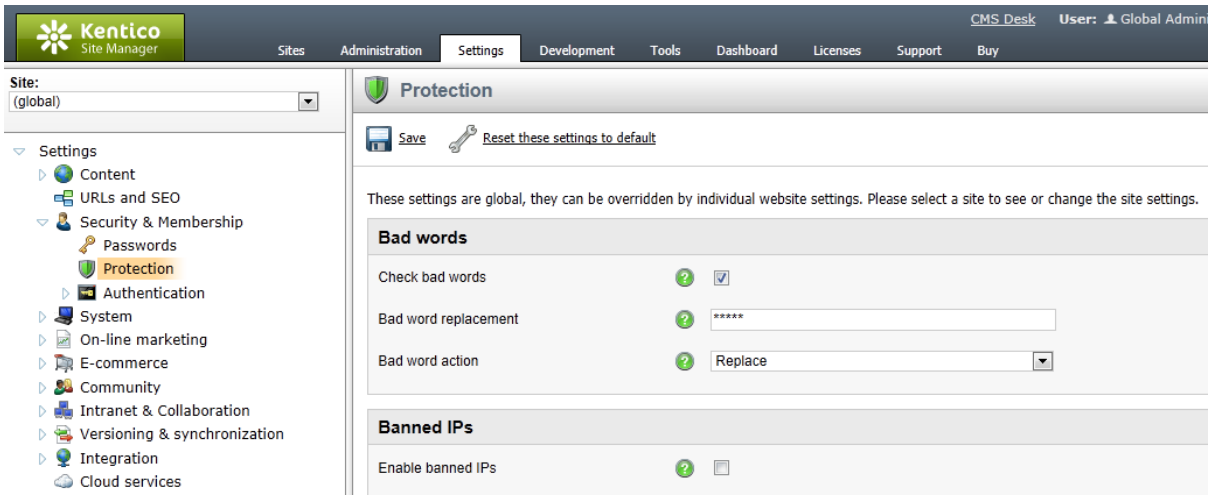
The Bad words module is just one of the modules which can help you deal with unwanted user input. Besides detecting unwanted input based on keywords, you may also let your site visitors report such content via the Abuse report module. The last resort in dealing with unwanted content may be blocking access 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.

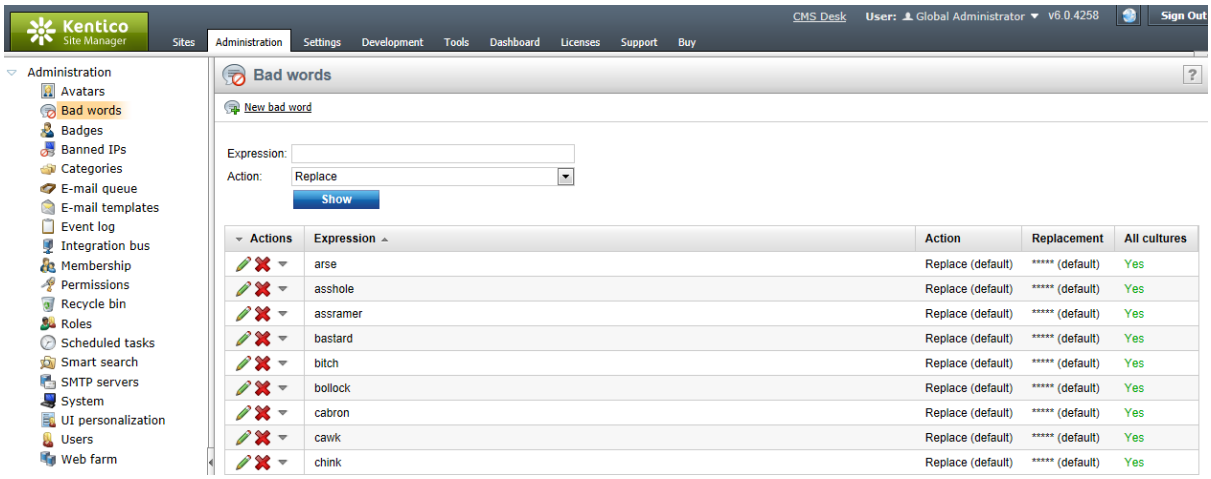
On this page, you can also define the default action that will be assigned to newly created bad words if the *Use default settings* option is enabled in the bad word creation dialog (see Defining a bad word for more details). The default action can be selected from the **Bad word action** drop-down list.

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.



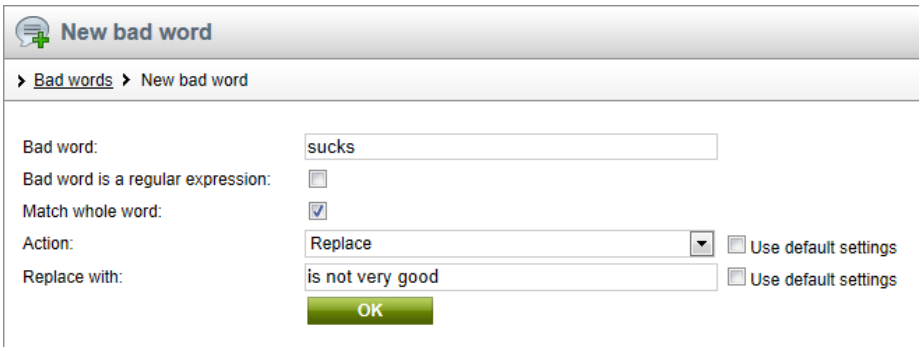
The top part of the page is a filter. Using it, you can display only those bad words that match the specified criteria. You can filter by the **Expression** and by the **Action** that should be taken on its detection. After specifying the filtering criteria and clicking the **Show** button, only those 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 the **New bad word** ( ) link at the top of the page. You will be redirected to the **New bad word** dialog. The following details can be entered:

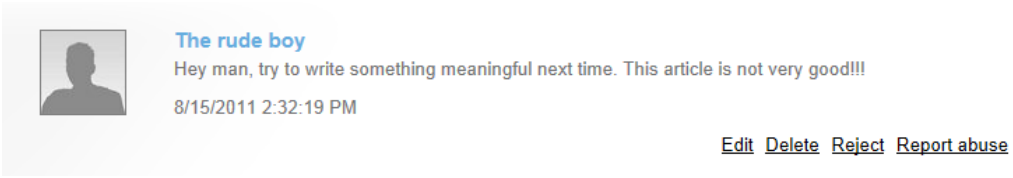
- **Bad word** - the string that should not appear in input text.
- **Bad word is a regular expression** - if checked, the string entered into the previous field will searched as a regular expression.
- **Match whole word** - if enabled, only whole word occurrences of the expression will be identified. 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**.

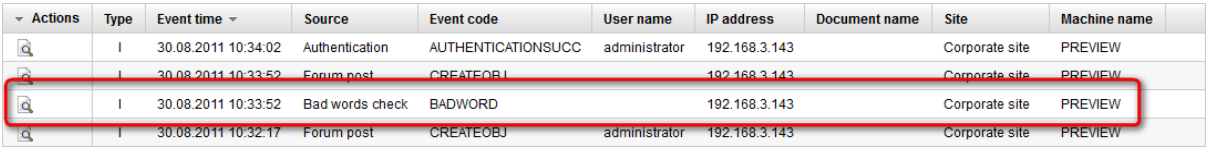


2. Let's try the functionality now. Go to the live Corporate Site, enter the Blogs section and open some of the blog posts. Enter a comment as in the following screenshot and click **Add**.

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

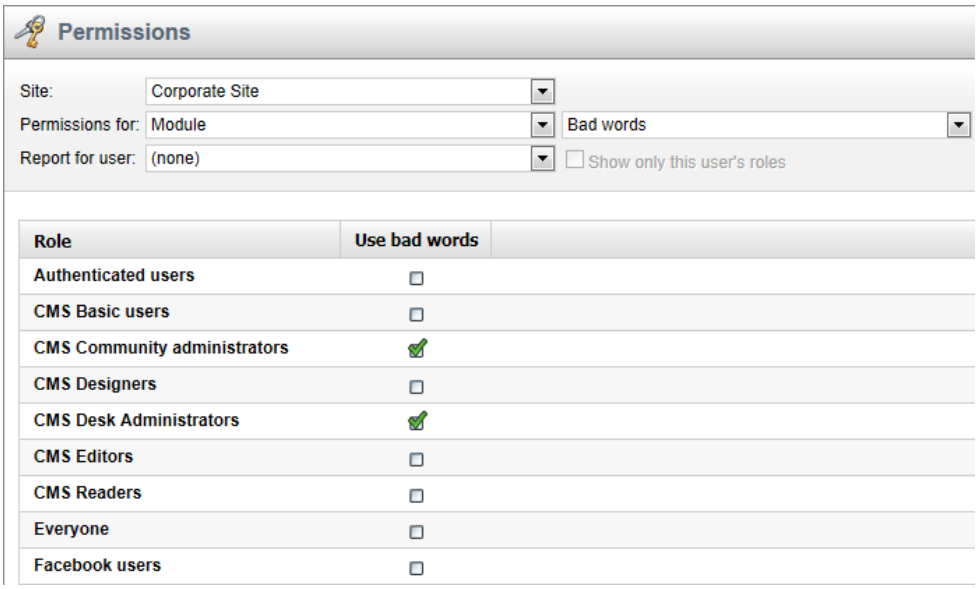
Using the **Add cultures** button, you can add cultures to the list. The **Remove selected button** removes all cultures selected by the check-boxes from the list.

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

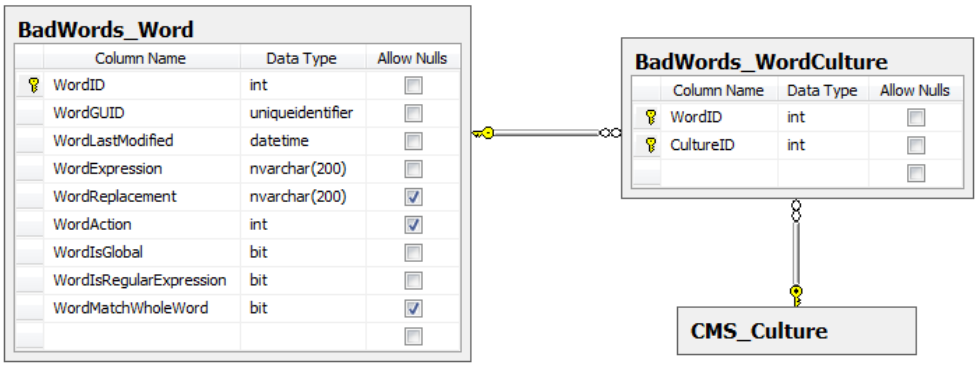
In this chapter, you will learn which database tables and API classes are used by the Bad words module. You will also see the most common API examples.

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.

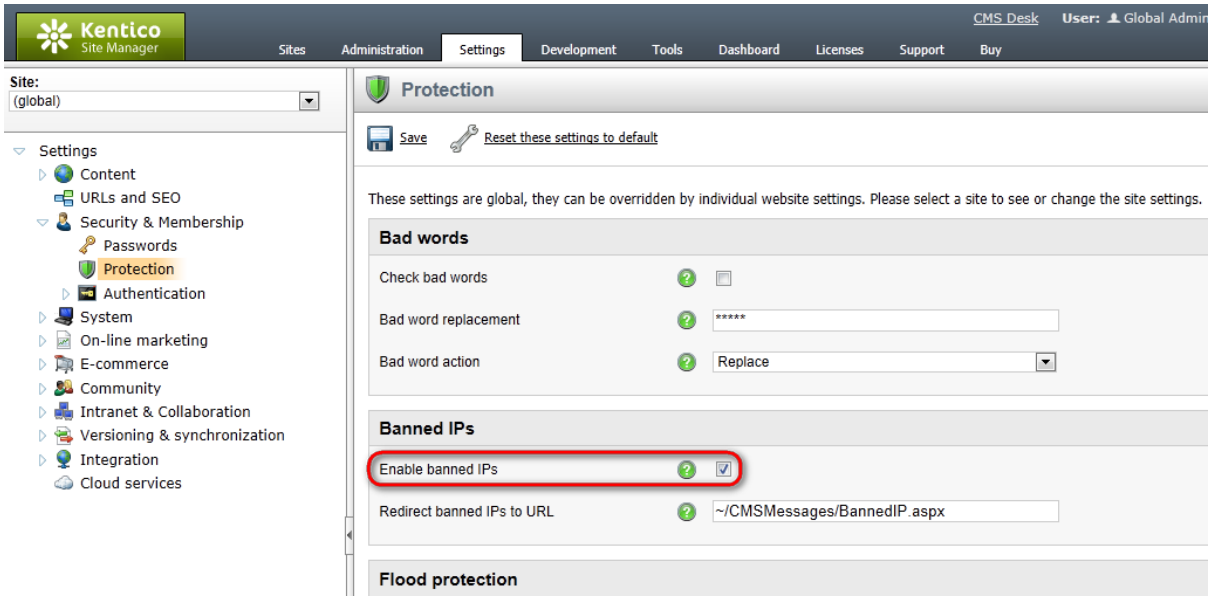
The Banned IPs internals and API sub-chapter provides information about the database tables and

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.



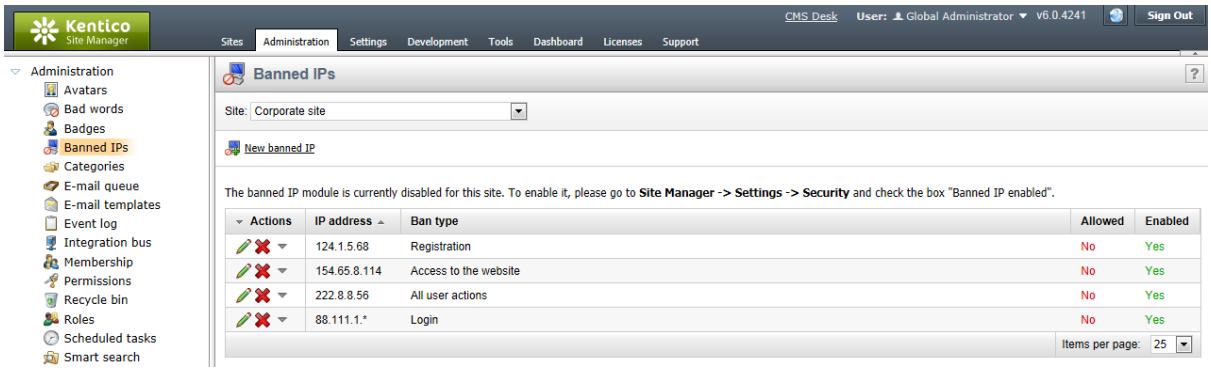
Users attempting to perform an action from an IP address that is banned should get a page with the following message displayed in their browsers.

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

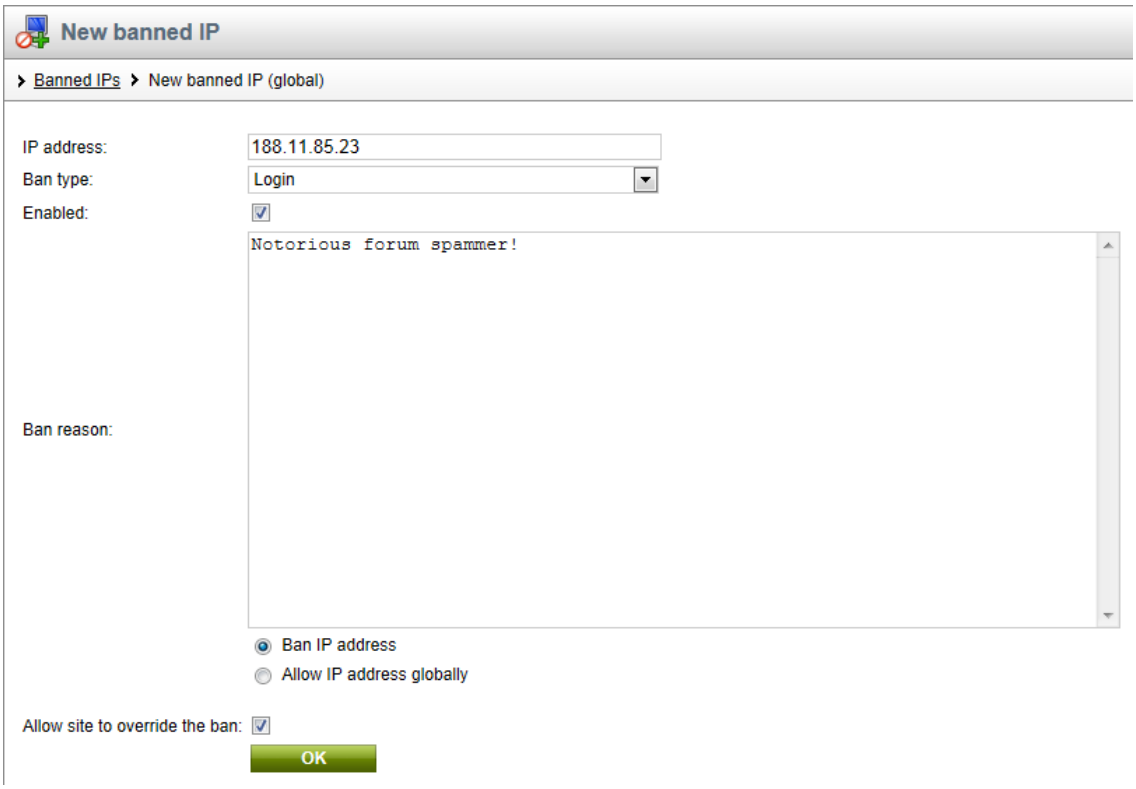
Banned IPs can be managed in **Site Manager -> Administration -> Banned IPs**. Alternatively, you can access the same page from **CMS Desk -> Administration**, but you can manage banned IPs only for the currently edited site.

From the **Site** drop-down list, choose the site you want to add the IP for and click the **New banned IP** link.



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 specific actions will not be allowed
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.

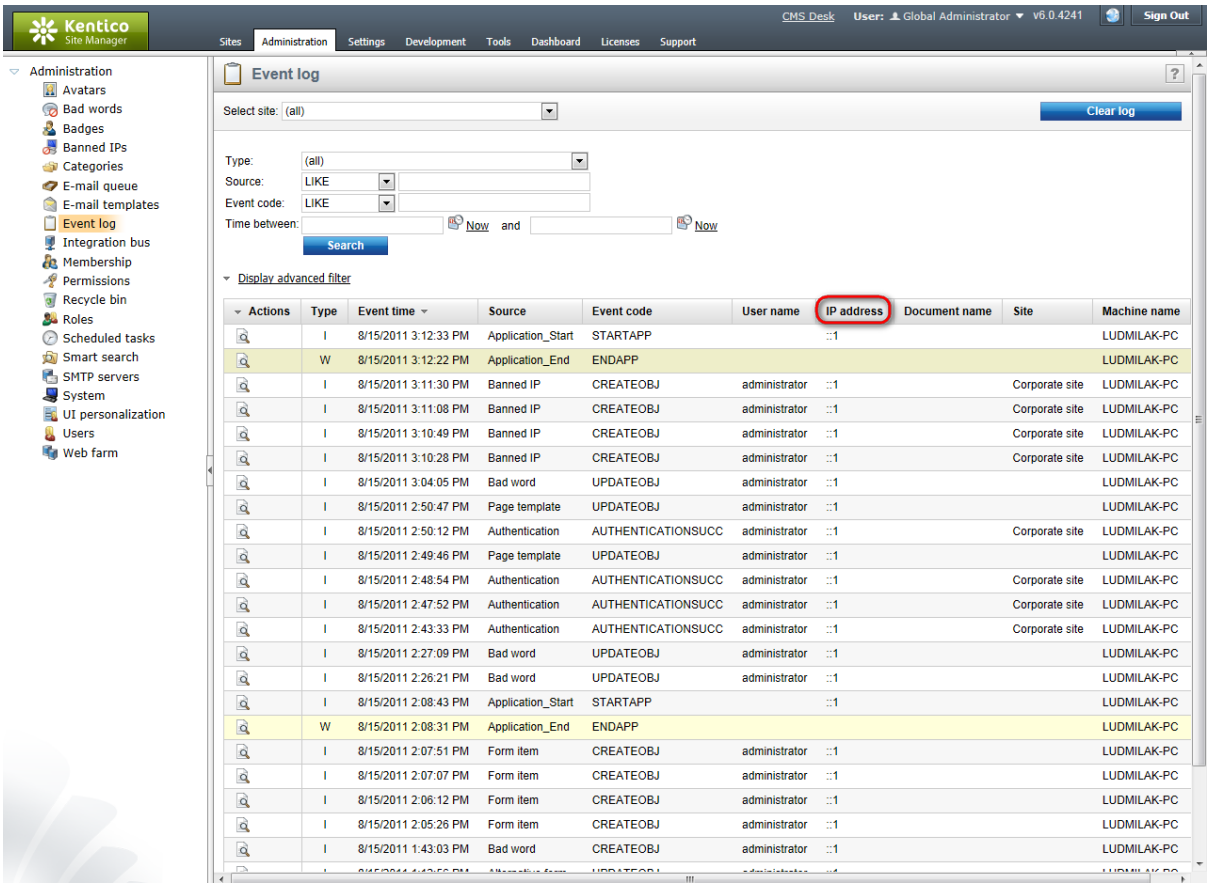
Editing an existing ban

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.

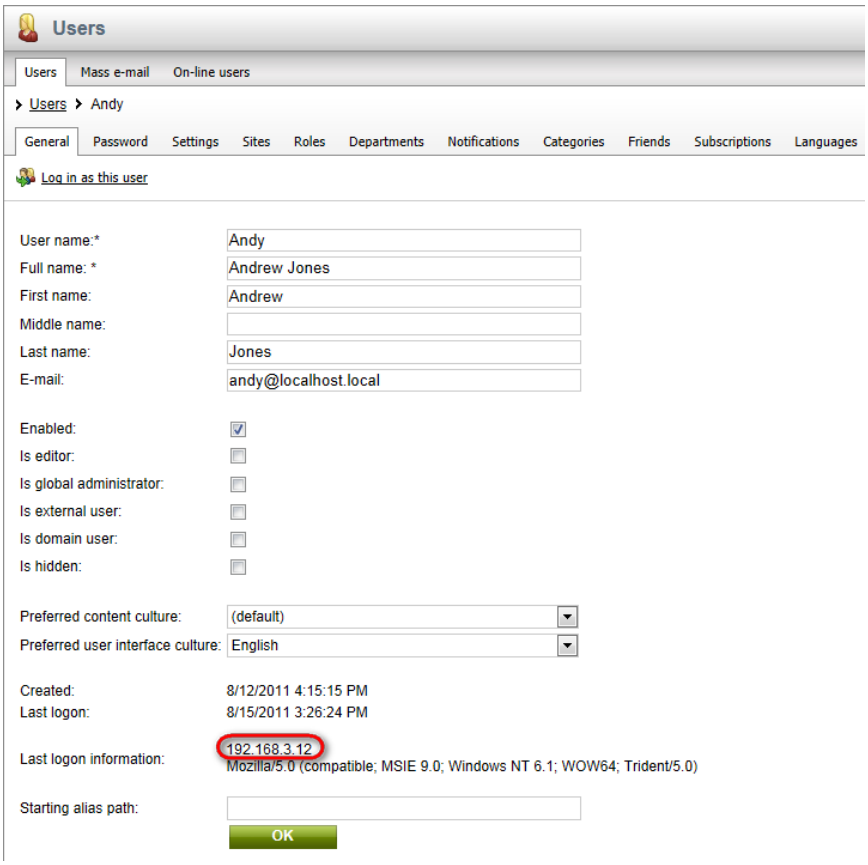
If you wish to find the IP address of a user responsible for a certain event, go to **Site Manager -> Administration -> Event log** and find the event you are interested in. The IP address of the user who caused the event has its own column in the displayed table and can also be viewed if you choose 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 **LIKE** is selected from the drop-down list. This will display 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.

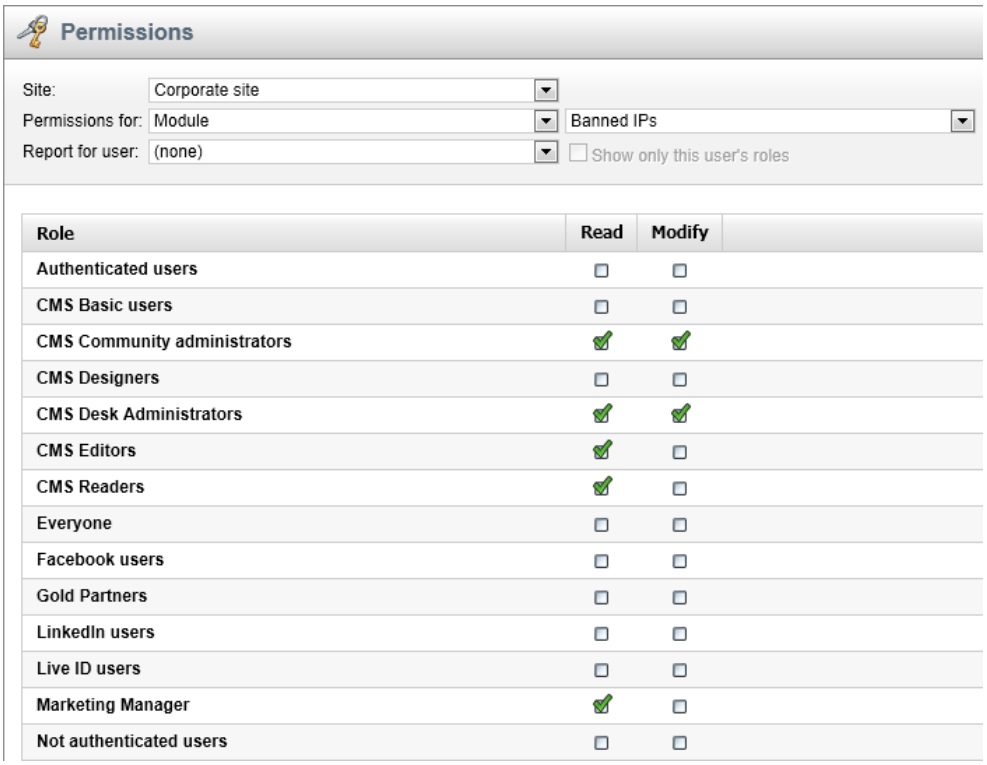


8.9.5 Security

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

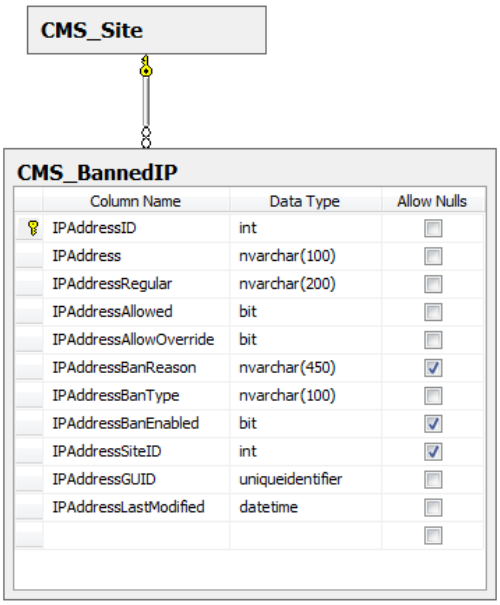
In this chapter, you will learn which database tables and API classes are used in the Banned IPs module. You will also see the most common API examples.

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;
}
```

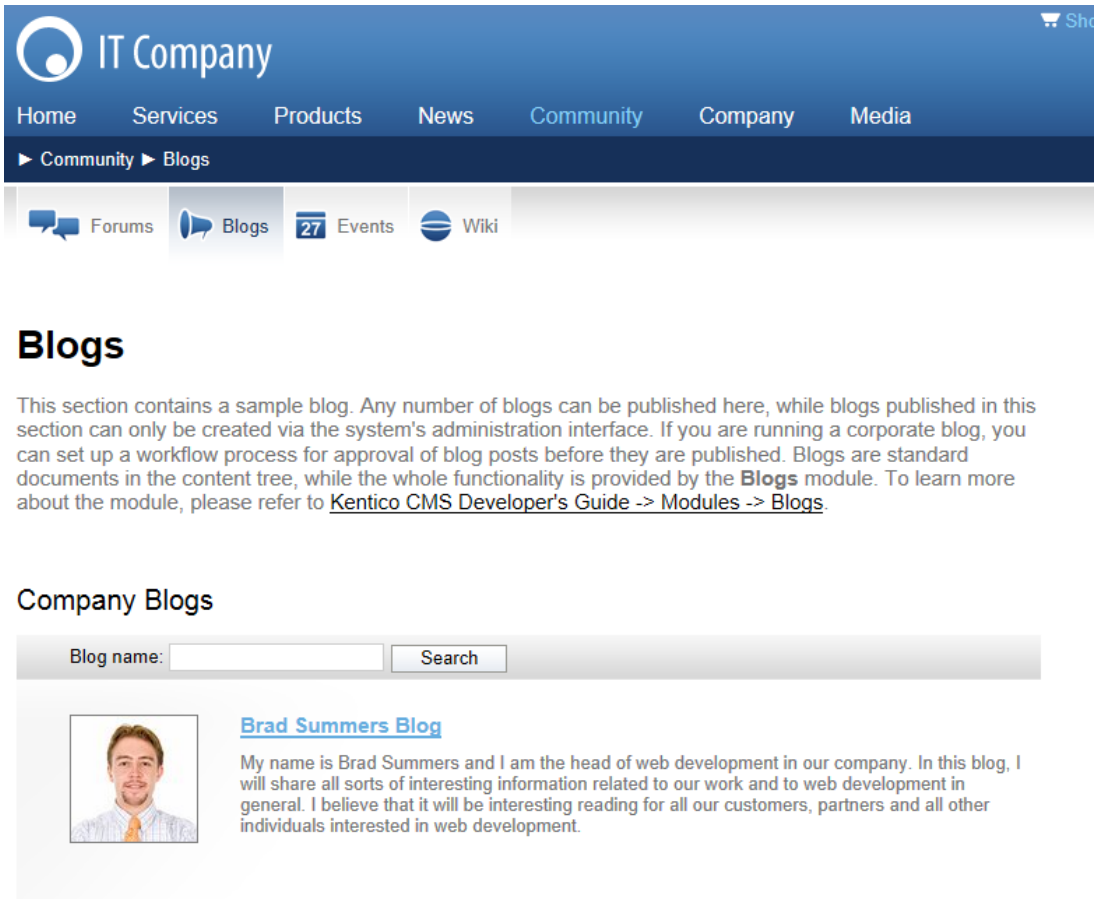
The method in the following example returns *true* if the specified IP address is banned for the current site, otherwise it returns *false*.

```
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 Blogs

8.10.1 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

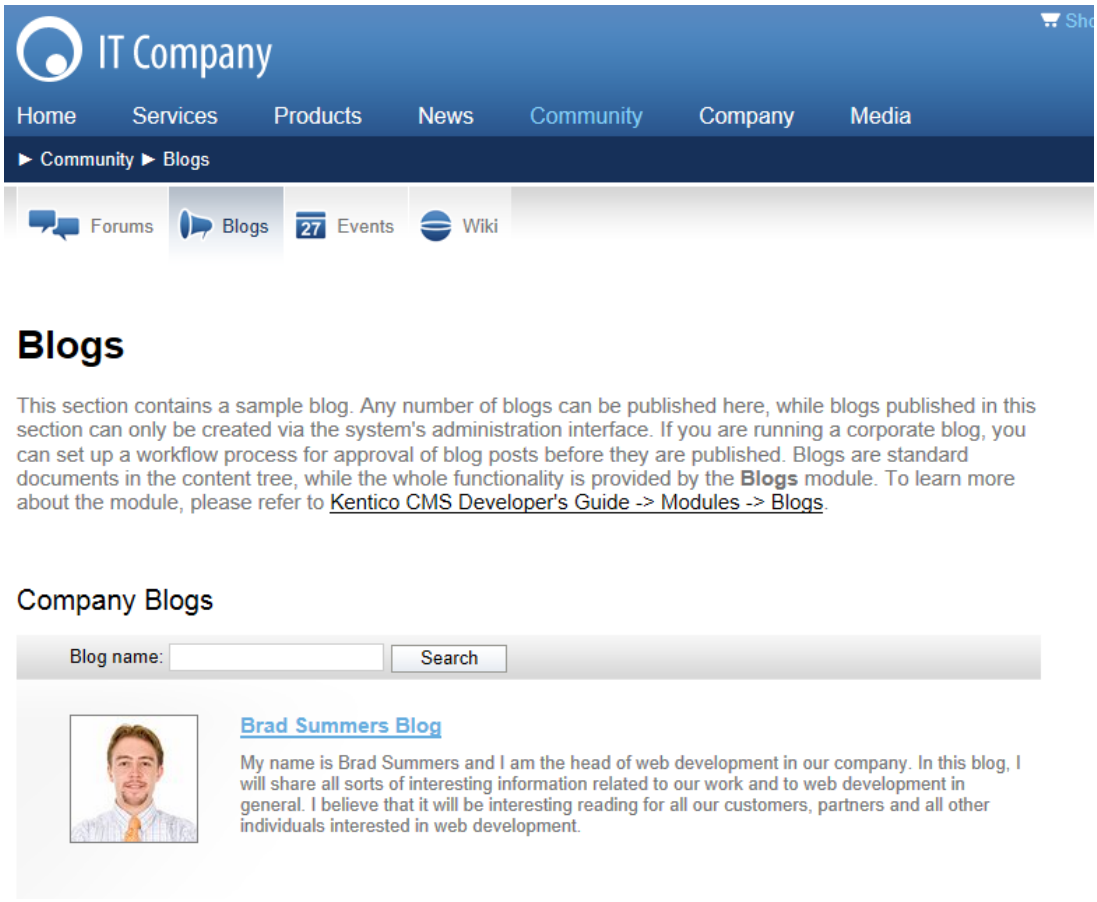
but also a tag cloud. The Tag cloud web part is used to display key words, called tags that are associated with a document, and is convenient for simple marking of documents according to various criteria, e.g. your interests.

Kentico CMS Community Site Guide contains some additional examples and tutorials related to blogs:

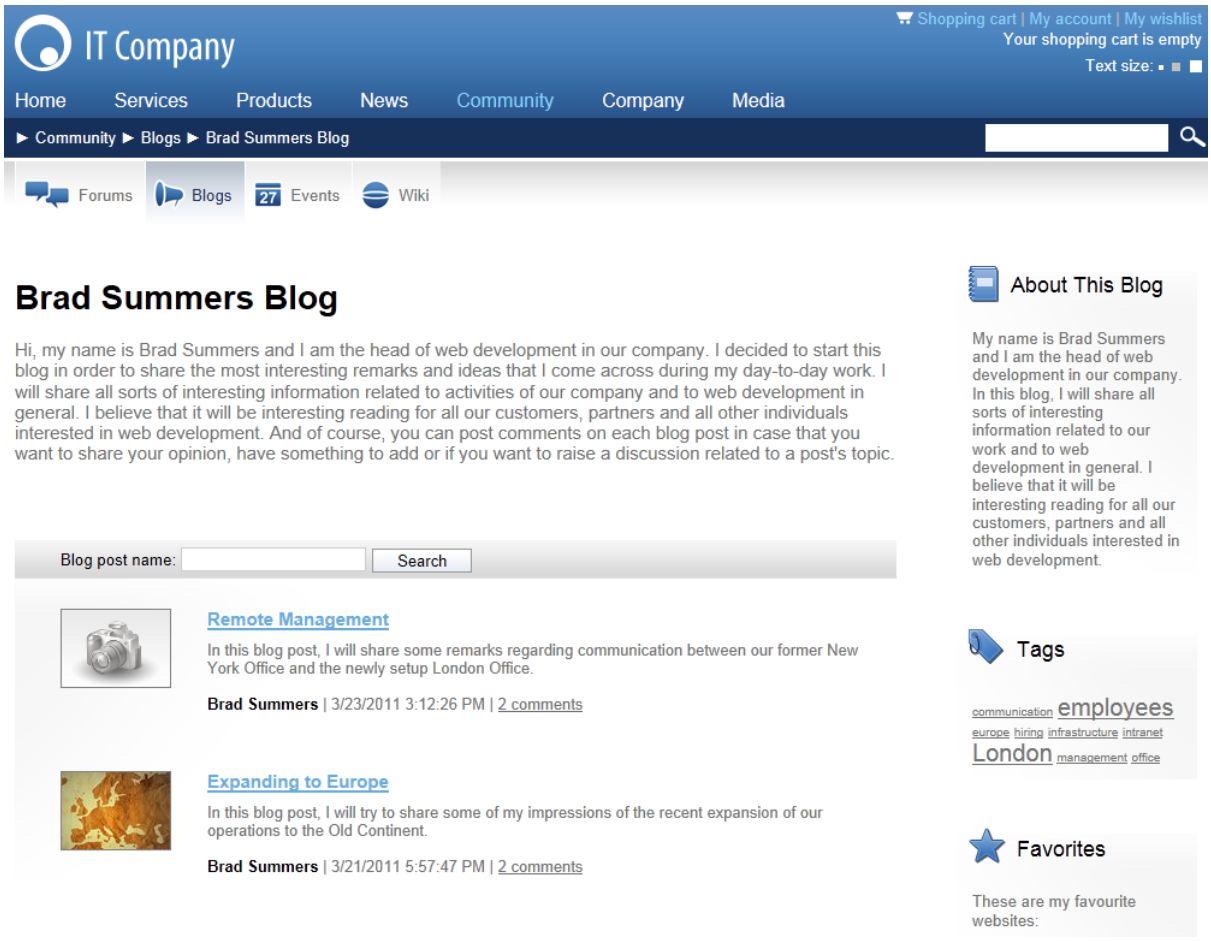
You will learn not only how to create a page on blogs (learn here how to do it) but also how to create a special page where users are redirected when they want to create a new blog (as explained here), a page that displays a list of all blogs on the site (as explained here) or how to create a page where only blog posts corresponding to the selected tag from the tag cloud are displayed (as explained here).

8.10.2 Sample blog

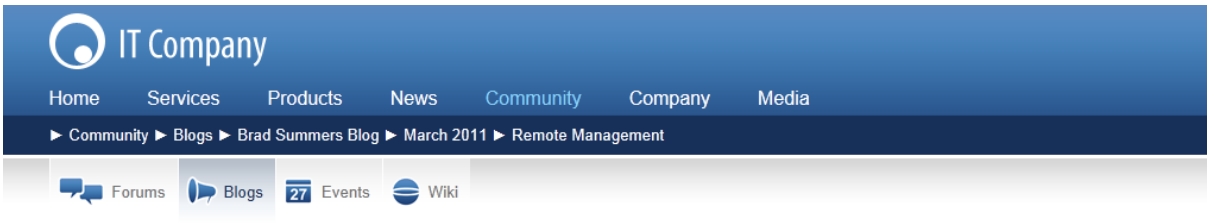
To see a sample blog, on the sample Corporate Site switch to the live site and from the **Main menu** choose **Blogs**. The Blogs page displays a list of blogs, each of which can be accessed by clicking on the particular blog link.



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.

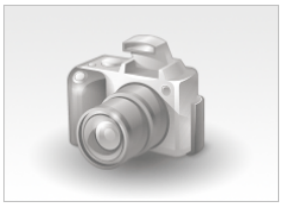


Finally, when you click on some particular blog post link, you can see the text plus the **Comments** section that allows you to read and add comments to the blog post. Additional information related to the current blog is also available.



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 | 1 comments
Filed under: employees, intranet, London, management, communication

Comments

Brad Summers

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

Leave comment Subscribe

Name:
Global Administrator

E-mail:
administrator@localhost.local

Your URL:

Comments:

Subscribe me to this blog post

Add

8.10.3 Adding a blog to your site

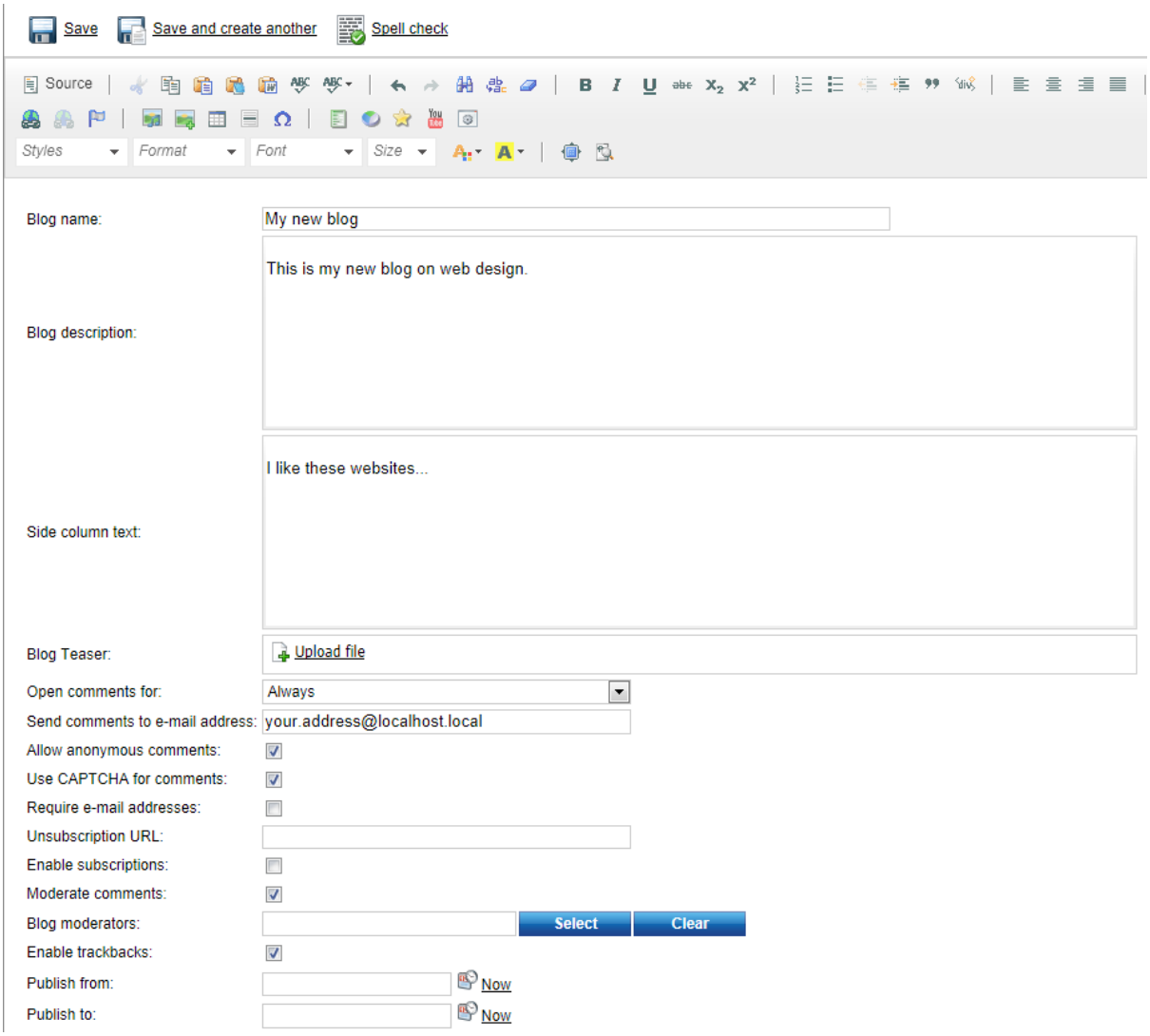
In this example, you will learn how to create a new blog in the sample Corporate Site, under the Blogs section.

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 on web design.
- **Side column text:** I like these websites...
- **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

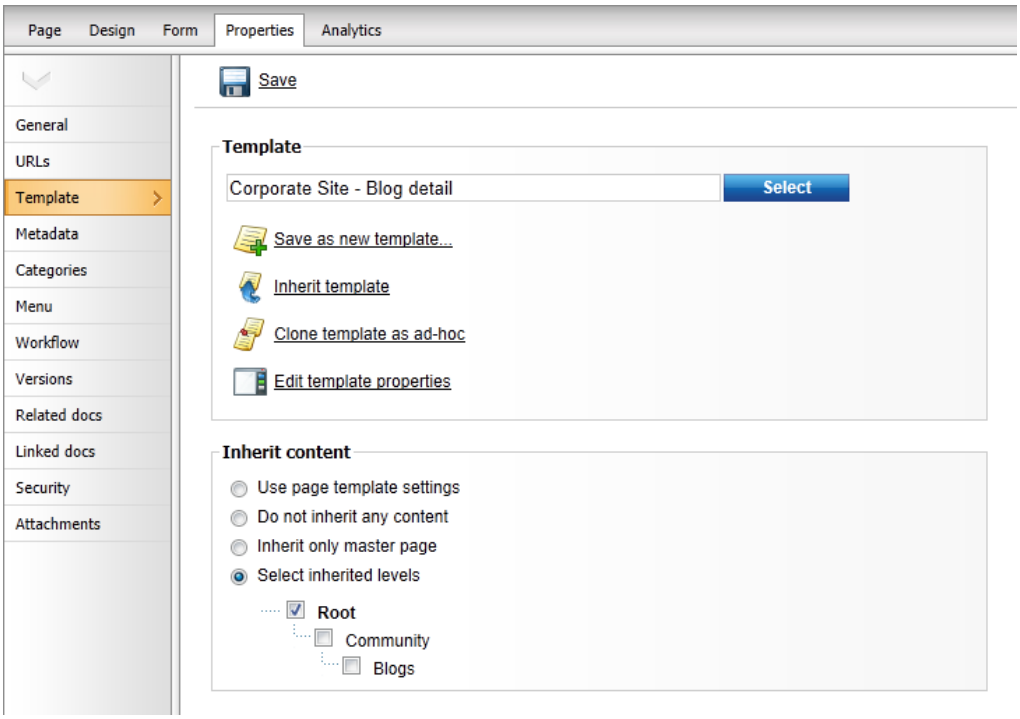
The following fields are also available, but you don't need to enter them for the purpose of this example:

- **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



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 **OK**.

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.

8.10.4 Adding posts to your blog

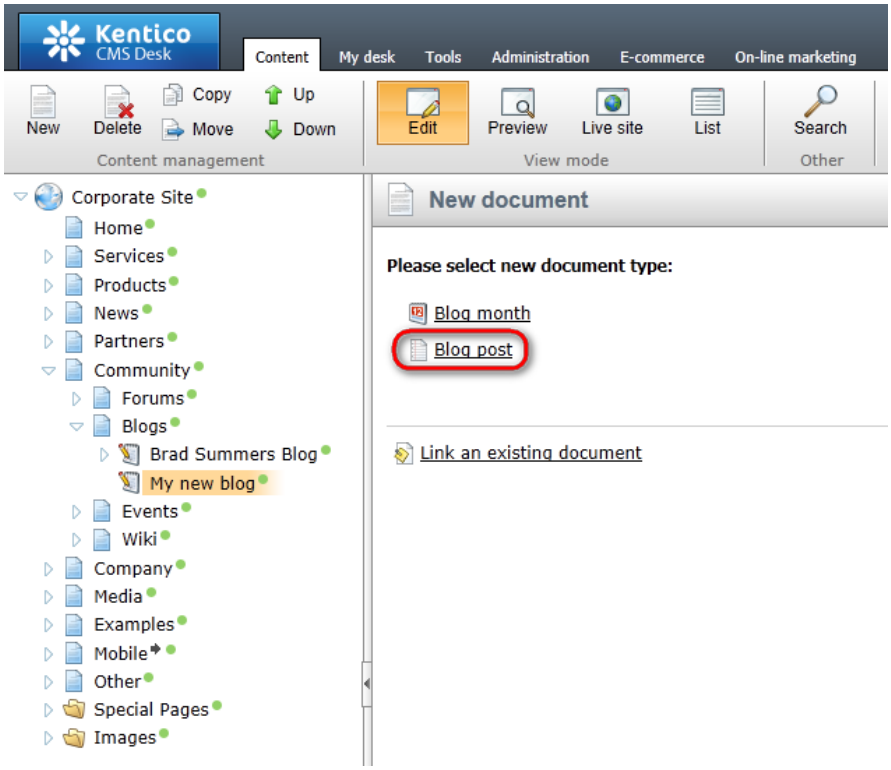
You can add new posts to your blog in two ways:

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

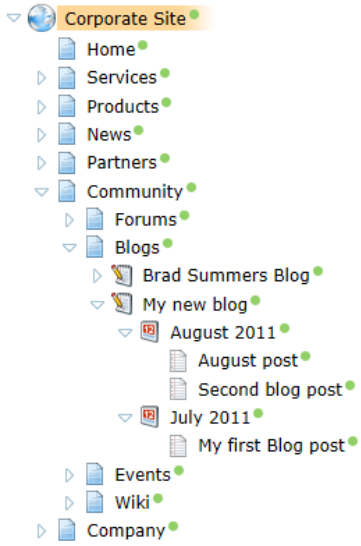
Choose the **Content** tab in **CMS Desk**, select the blog's main document and then click **New**. Choose to create a new **Blog post**:



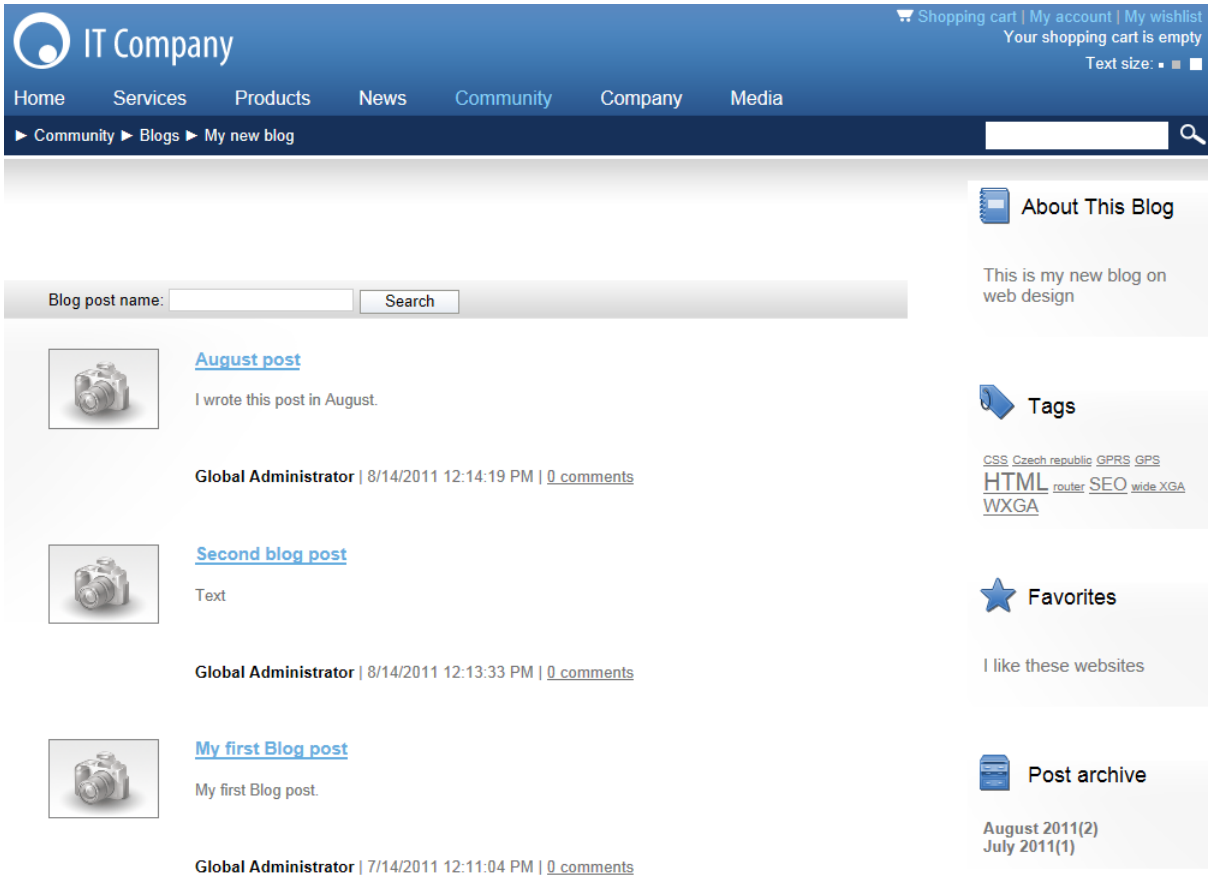
Enter the post content and click **Save**.

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:



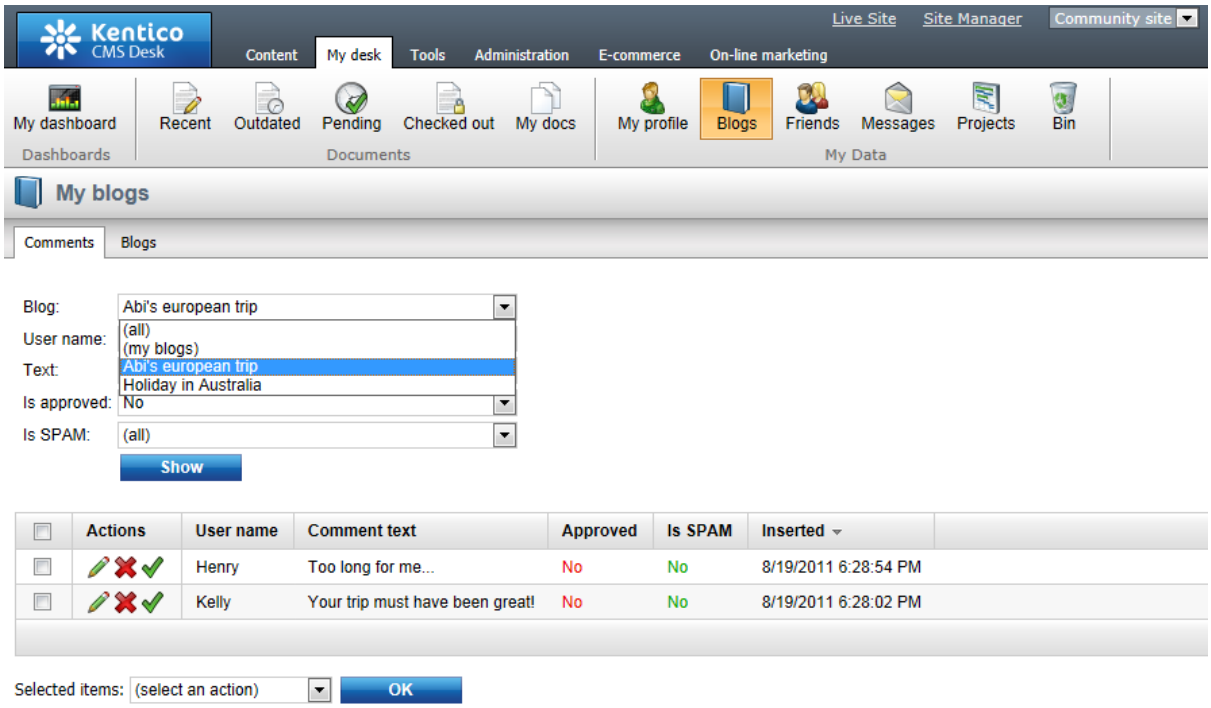
8.10.5 Moderating comments

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**.

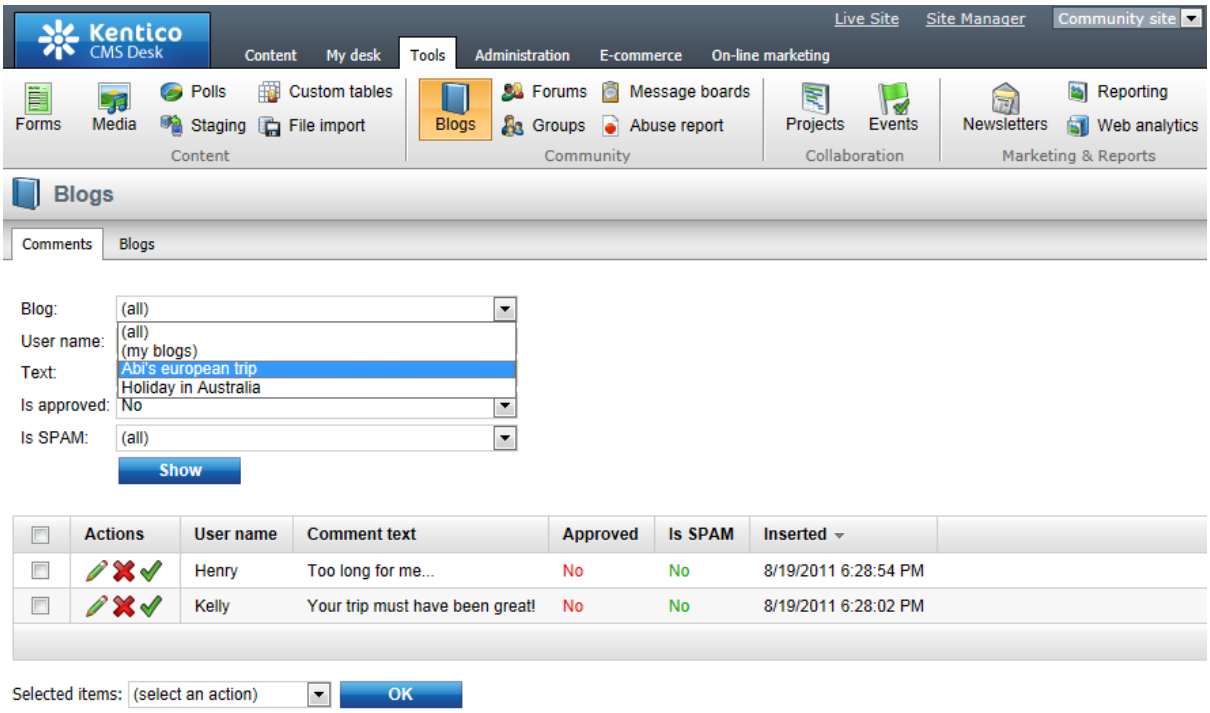


2. In the CMS Desk -> Tools -> Blogs section

In this section, management of blog comments on the site is possible based on the permissions:

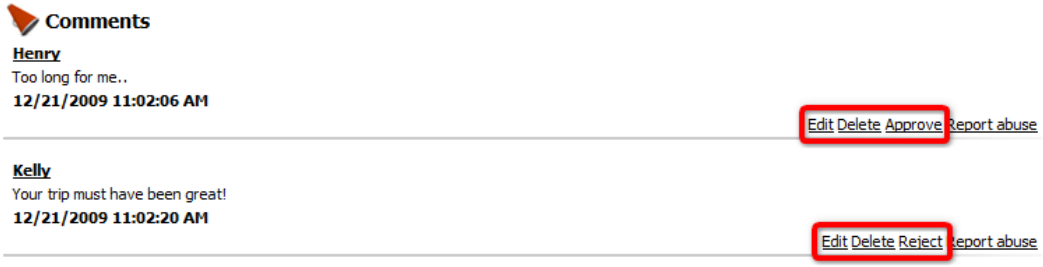
Global administrators and users with the **Manage comments** permission for the Blogs module can manage comments of all blogs on the current site. Blog **moderators** and blog **owners** can manage comments of their own blogs and blogs for that they are moderators. These permissions are reflected by 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**.



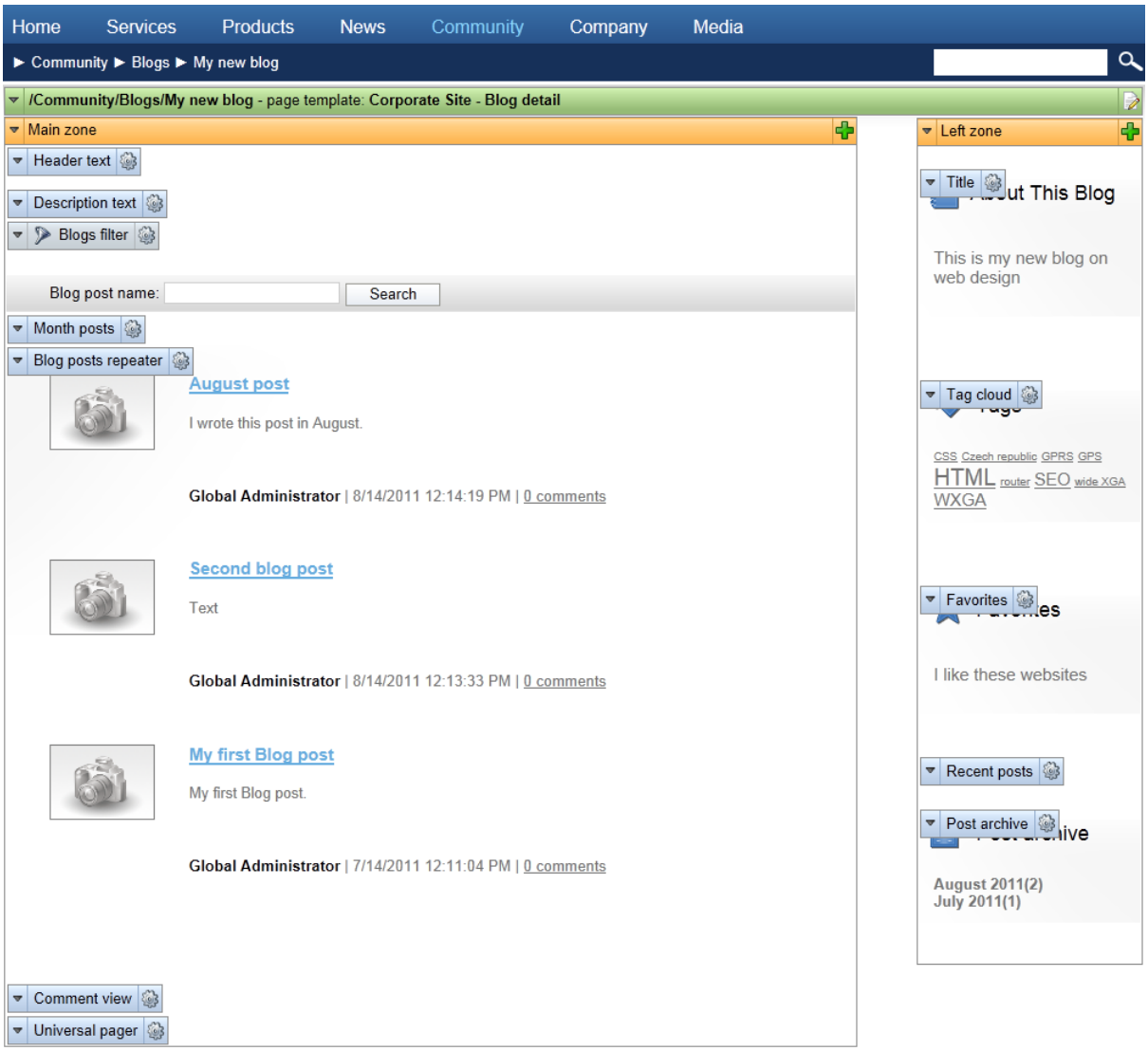
3. On-site management

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.



8.10.6 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

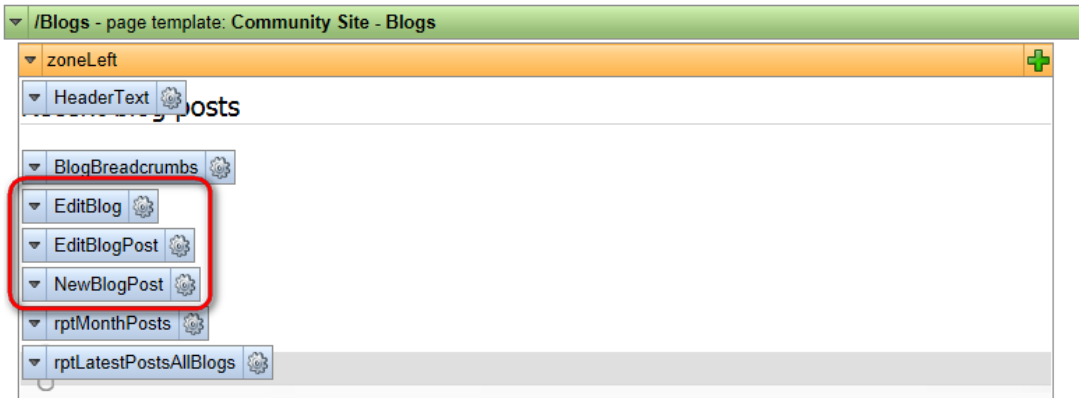
8.10.7 On-site management via User contributions

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.

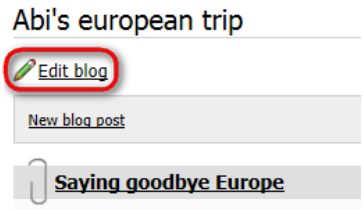
Edit blog properties

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.



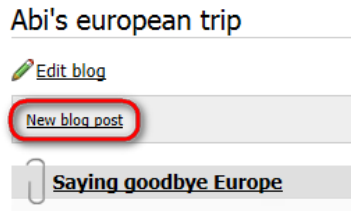
Add new blog posts

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.



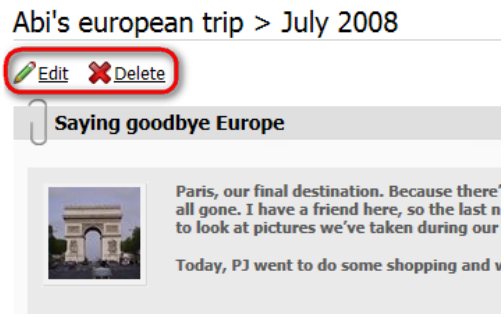
Edit or delete blog posts

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

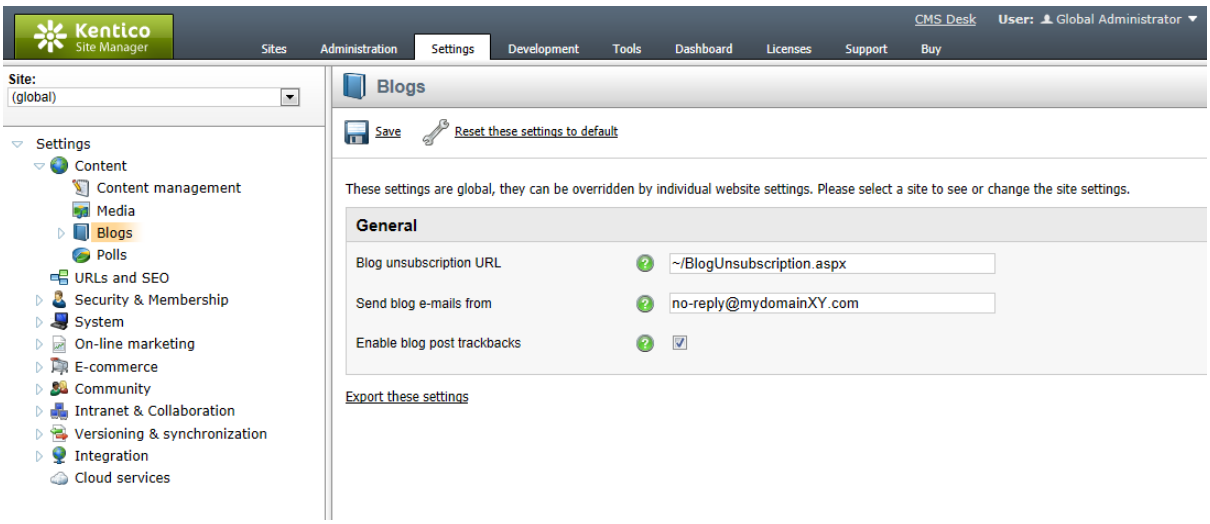
If you go to the live site and display some blog post placed under the blogs section title page, you should see the **Edit** and **Delete** links as highlighted in the screenshot below. After clicking the **Edit** link, the alternative form will be displayed, letting blog owners edit the blog post. Using the **Delete** link, they can delete the blog post.



8.10.8 Settings

Settings of the Blogs module can be done in **Site Manager -> Settings -> Content -> Blogs**. The following settings are available:

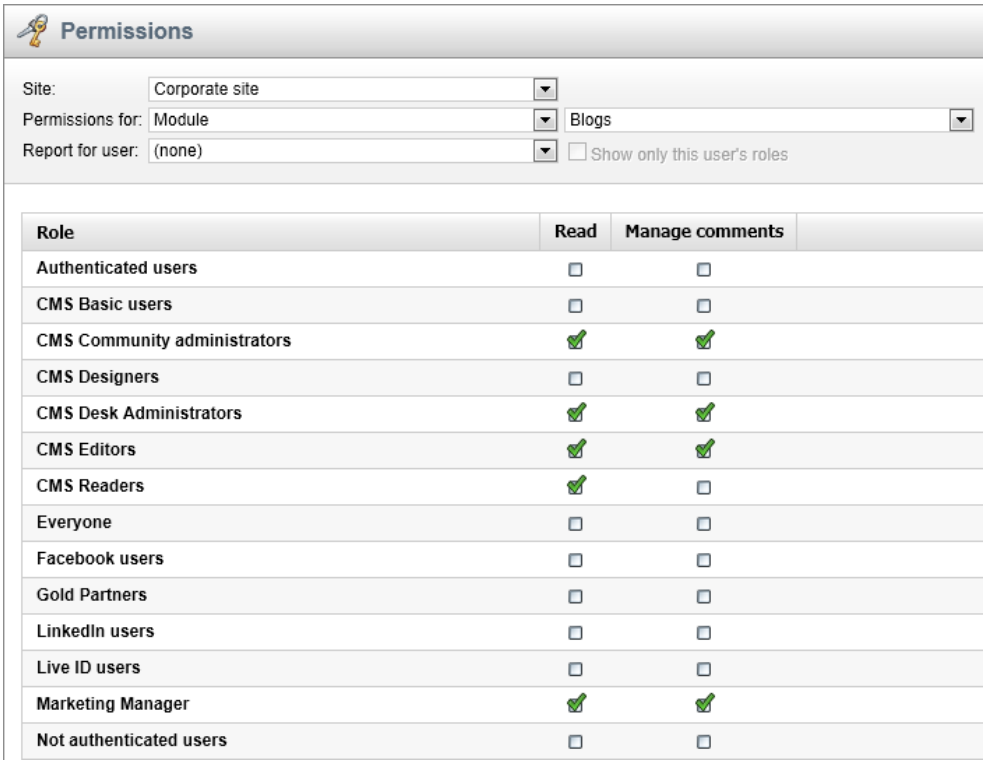
- **Blog unsubscription URL** - URL of a page on which the **Blog post unsubscription** web part is placed; this is a special web part used for handling unsubscriptions from receiving notifications about new blog posts
- **Send blog e-mails from** - e-mail address that will be used as the Sender ('From') 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.



8.10.9 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.

8.10.10 Trackbacks

8.10.10.1 Trackbacks overview

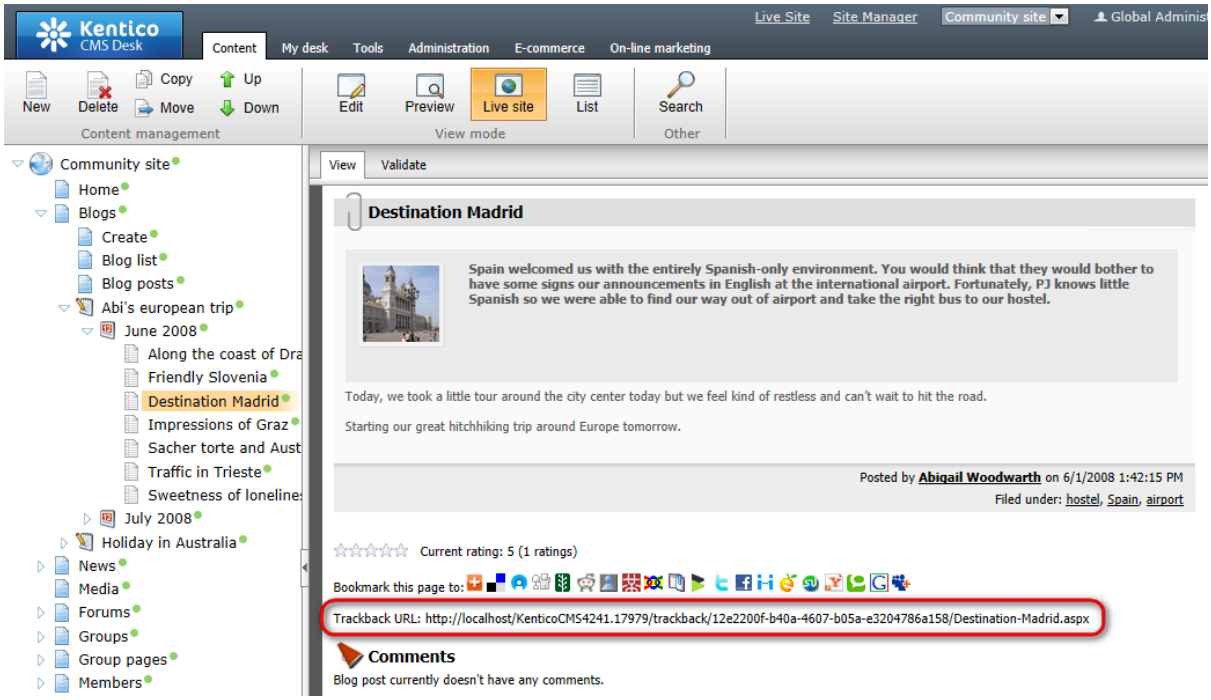
The **Trackbacks** feature allows sending of blog post links to other blogs when a new blog post is created. The link is typically, but not always, added as a blog comment in the target blog, as implemented in Kentico CMS. This can be performed not only within the same site, but also to completely different 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

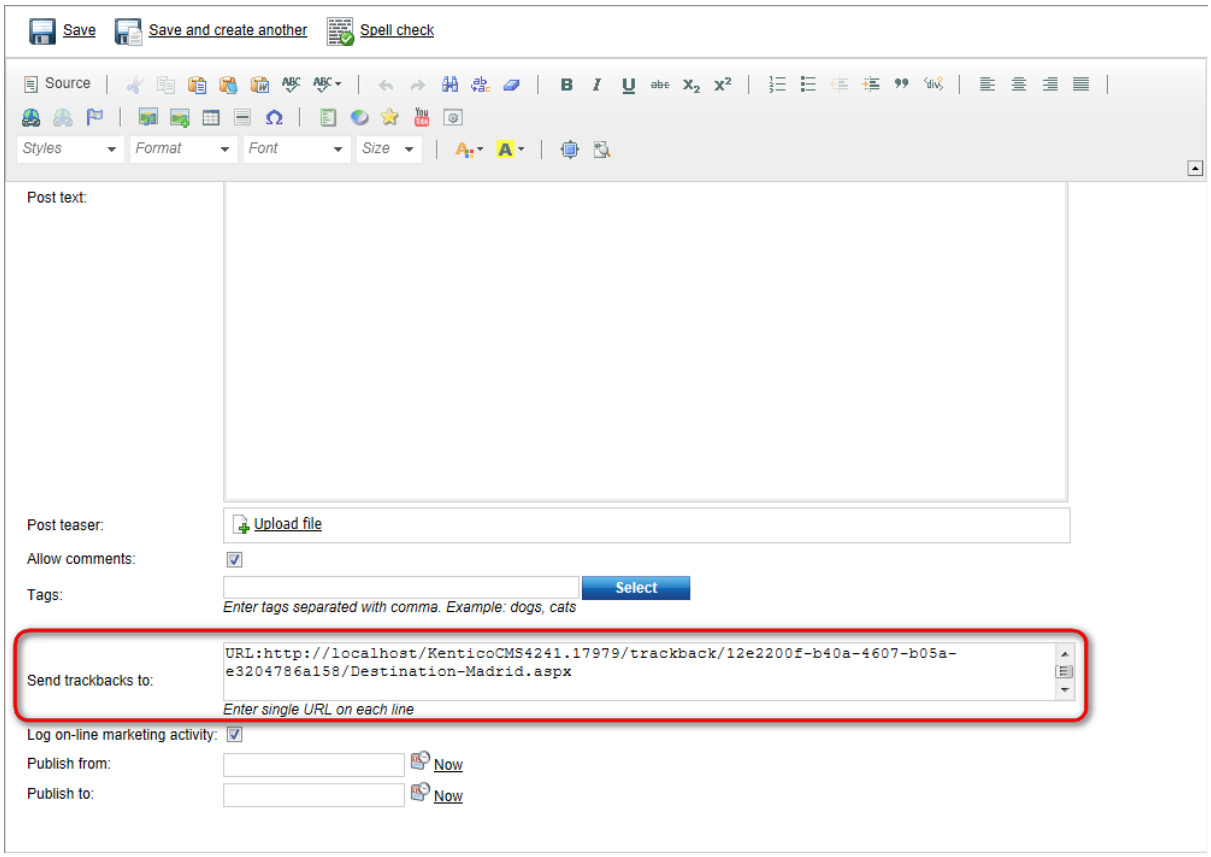
Read here to learn how to enable trackbacks on your site.

How it works

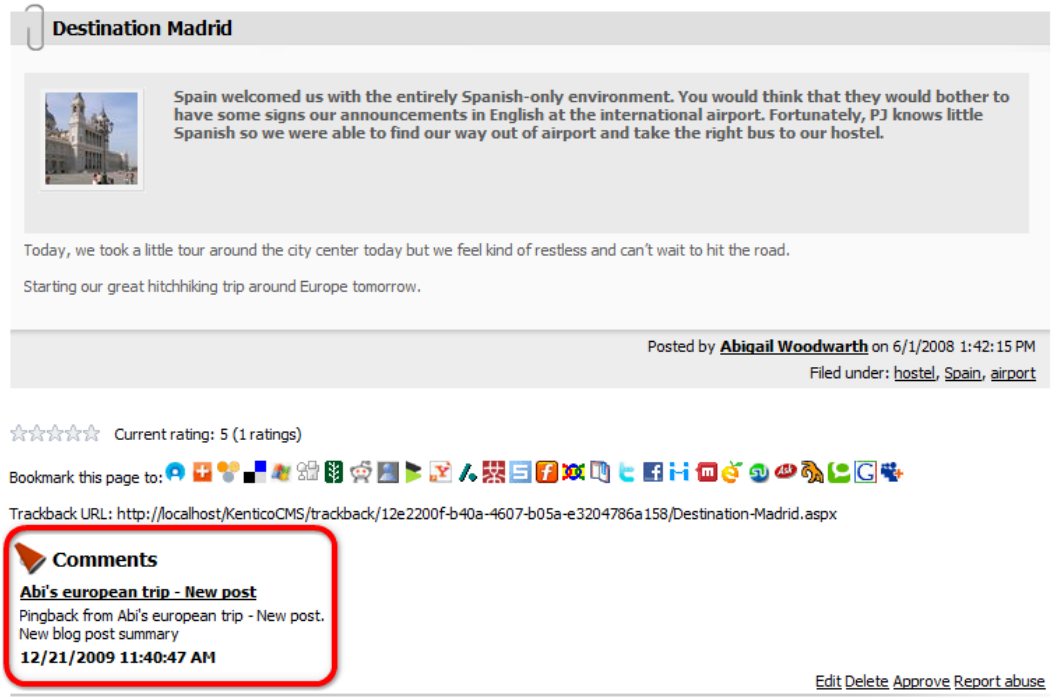
1. There is a Trackback URL published below the blog post. These URLs can be found on various blogs at sites developed not only with Kentico CMS.



2. When users are creating a new 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.



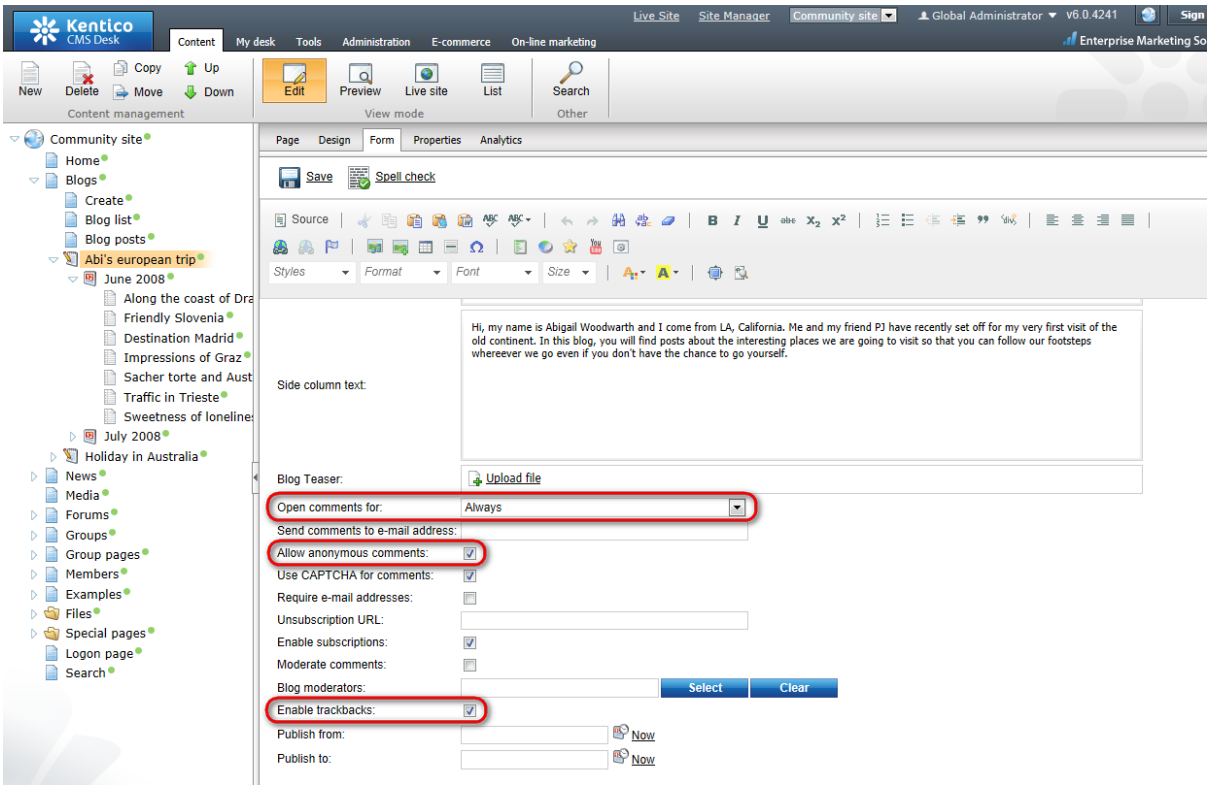
3. When the blog post is submitted, trackback ping is sent to the trackback URLs. If everything is configured correctly at the other blogs, a blog comment is added with the blog post link, title and summary, as you can see in the screenshot below.



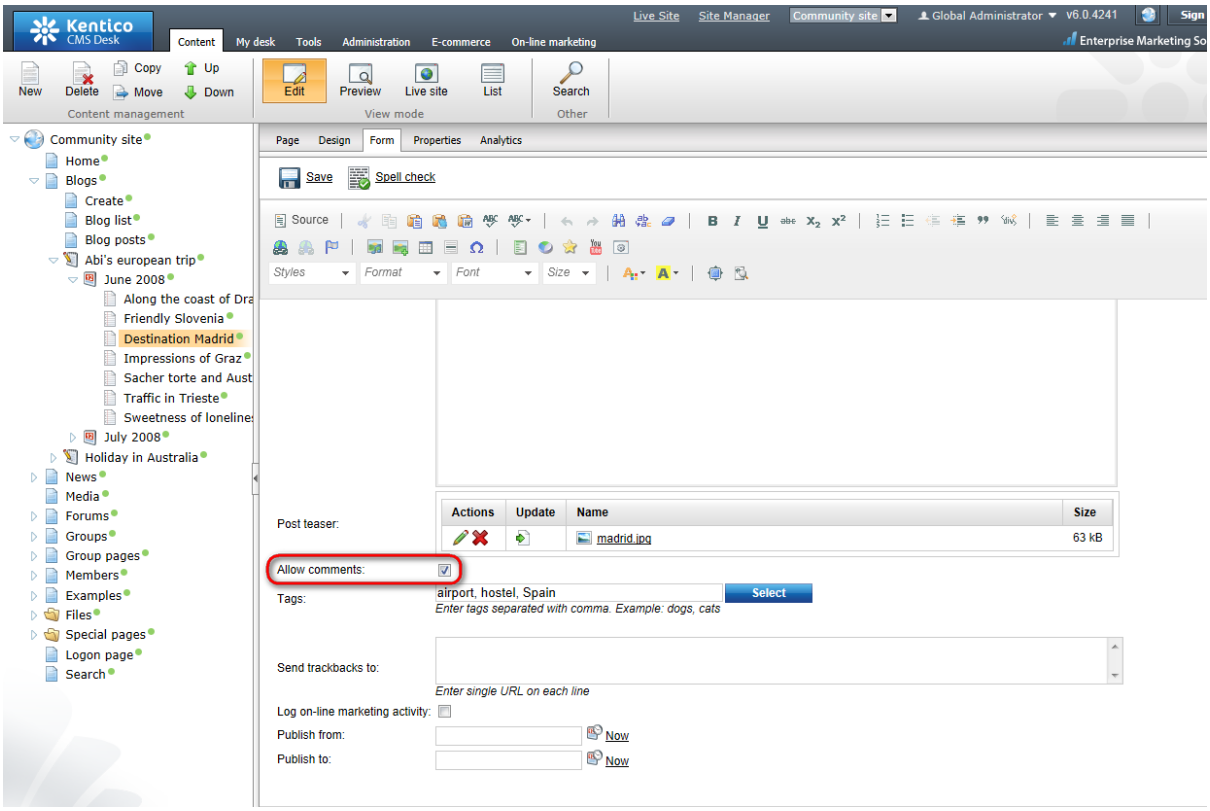
8.10.10.2 Enabling trackbacks

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.



8.10.11 Blog comments notifications

8.10.11.1 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

8.10.11.2 User subscriptions

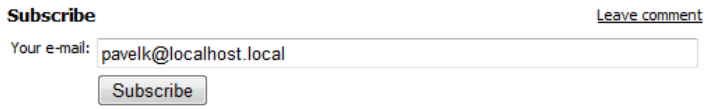
Users can subscribe to receiving notifications about new blog comments at some blog post. It can be done in two ways:

1. 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.

2. Users can also subscribe without leaving a comment, by clicking the **Subscribe** link at the top of the **Leave comment** form.

This displays a subscription form. By entering the e-mail address to the **Your e-mail** field and clicking

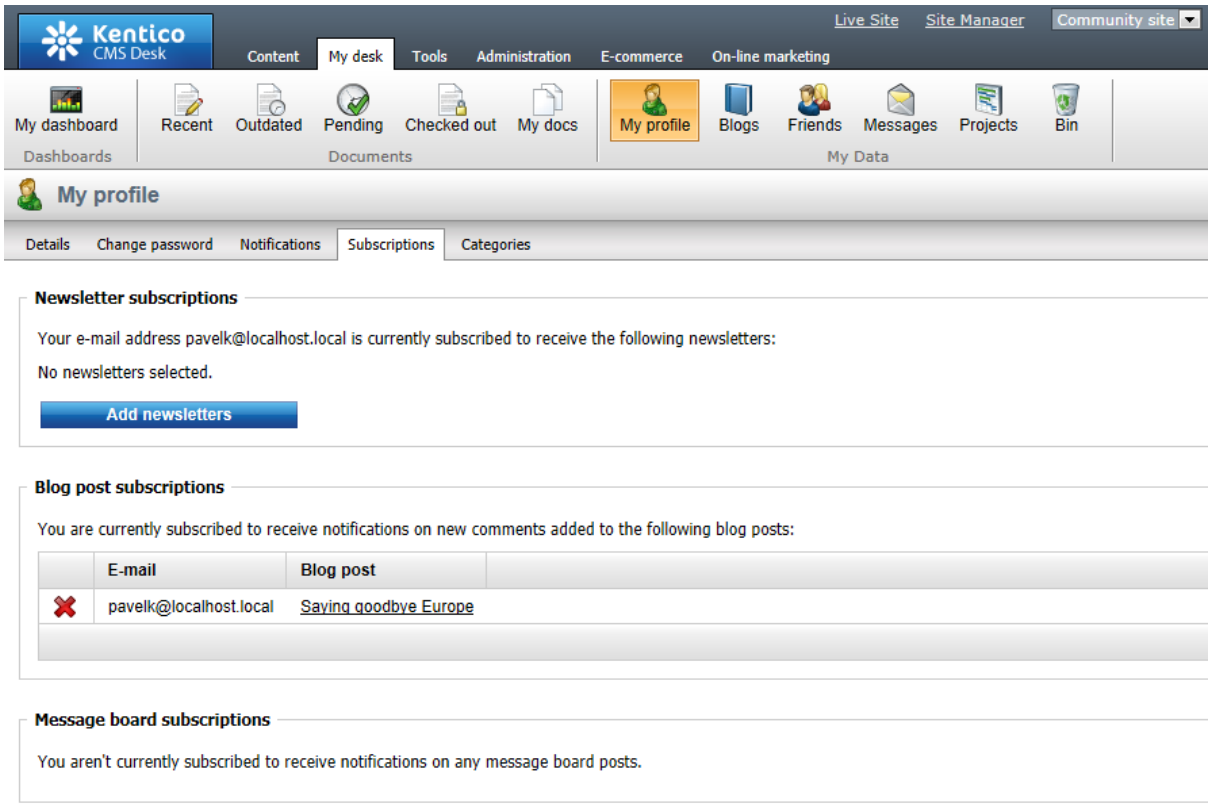
the **Subscribe** button, users can subscribe too.



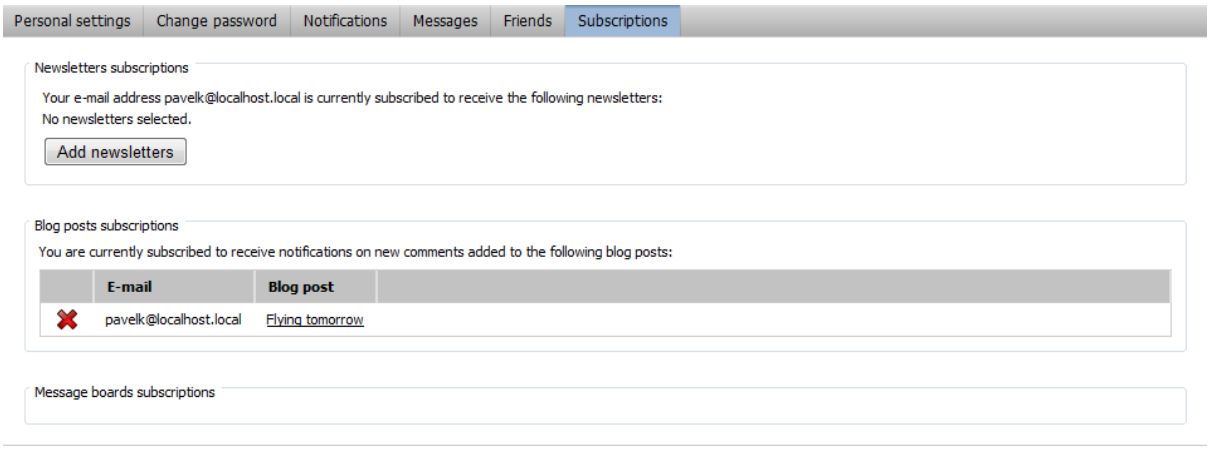
Subscriptions management

Users can view their subscriptions and eventually **unsubscribe** using the **Delete** (✖) icon at the following two places:

1. Users with access to **CMS Desk** can view their subscriptions on the **My Desk -> Account -> Subscriptions** tab.

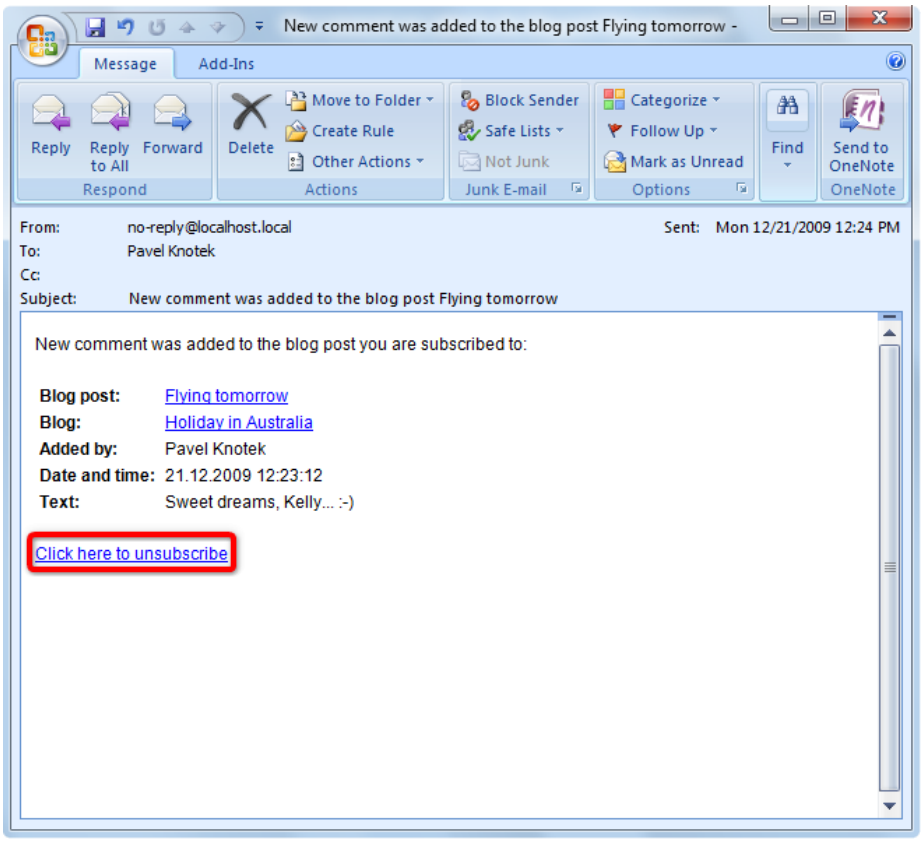


2. On live site, users can view their subscriptions in the **My account** web part. The **Display my subscriptions** property of the web part must be enabled for this to be possible.



Unsubscription possibilities

Subscribers can 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 to some 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**.

8.10.11.3 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:

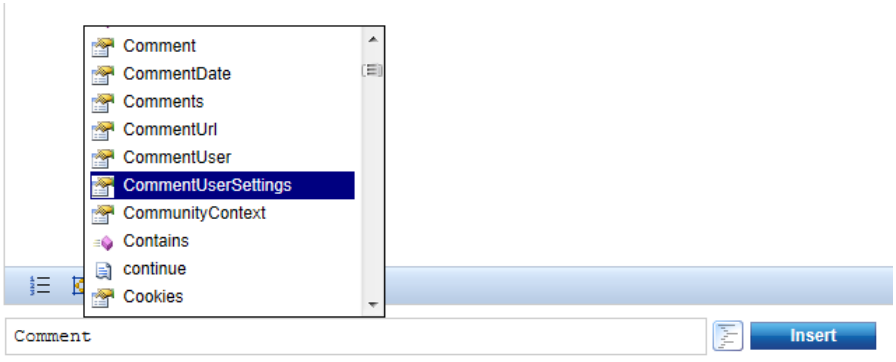
- **Blog owners** - e-mails are based on the **Blogs - Notification to blog owner** e-mail template
- **Blog moderators** - e-mails are based on the **Blogs - Notification to blog moderators** e-mail template
- **Subscribers** - e-mails are based on the **Blogs - Notification to blog post subscribers** 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

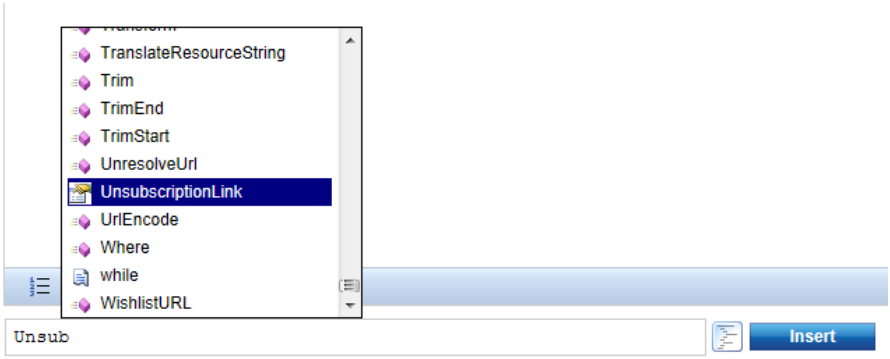
a) Links

- **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%}*



8.10.12 MetaWeblog API

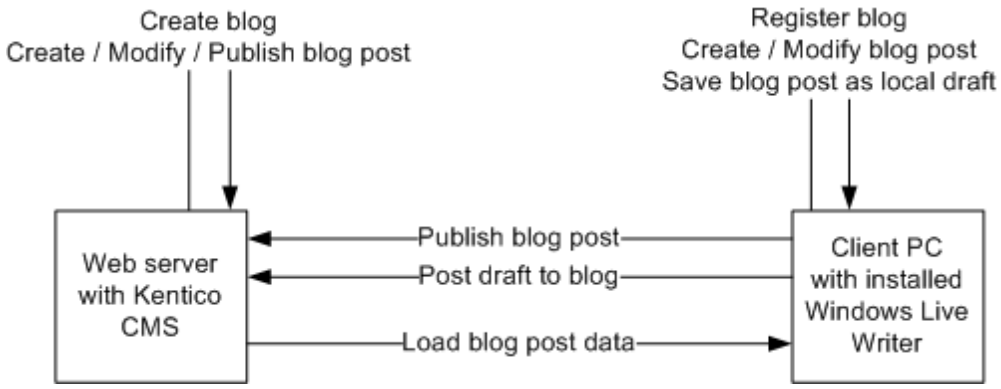
8.10.12.1 Overview

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

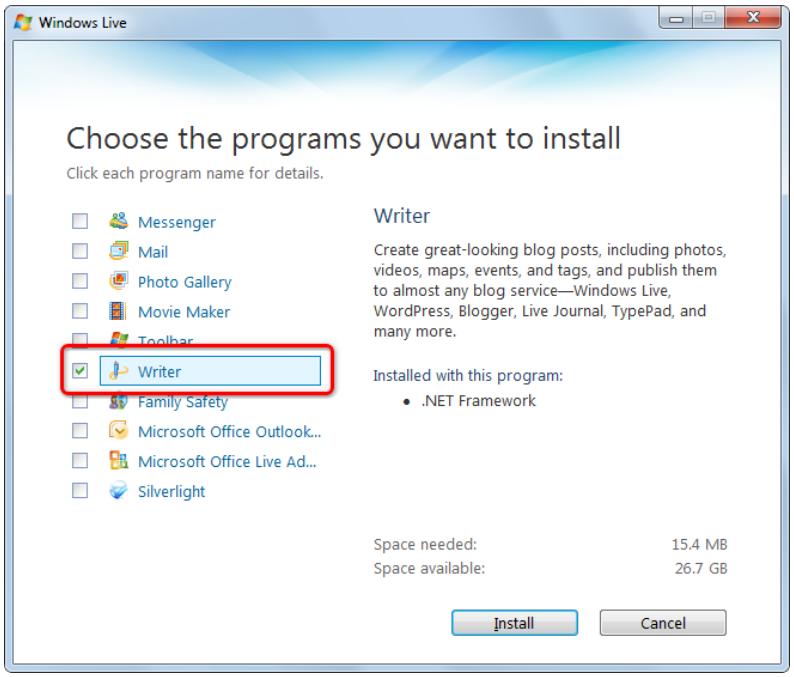
MetaWeblog API cannot be used by users authenticated by third-party authentication services (LiveID, OpenID etc.).

8.10.12.2 Windows Live Writer installation

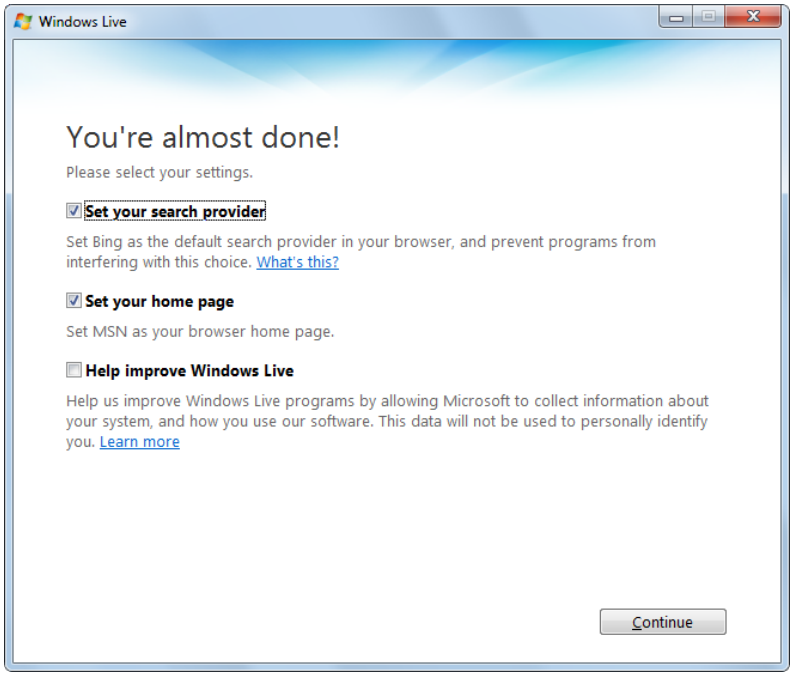
The following steps need to be taken in order to install Windows Live Writer on your machine:

1. Download Windows Live installer from http://download.live.com/writer.

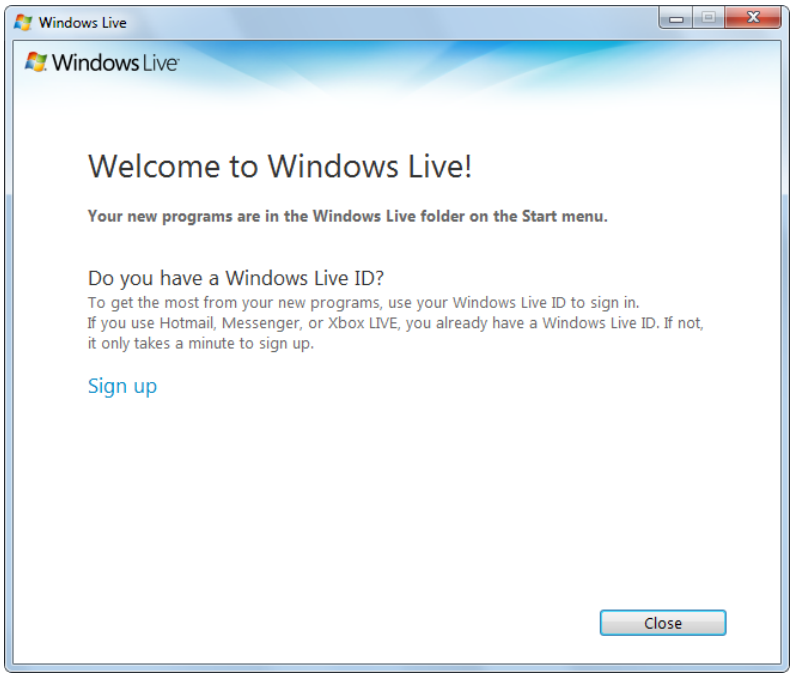
2. Execute the installer. Choose the **Writer** option from the list of available programs. Other options are not directly related to the Windows Live Writer. Choose as you wish and click **Install**.



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.

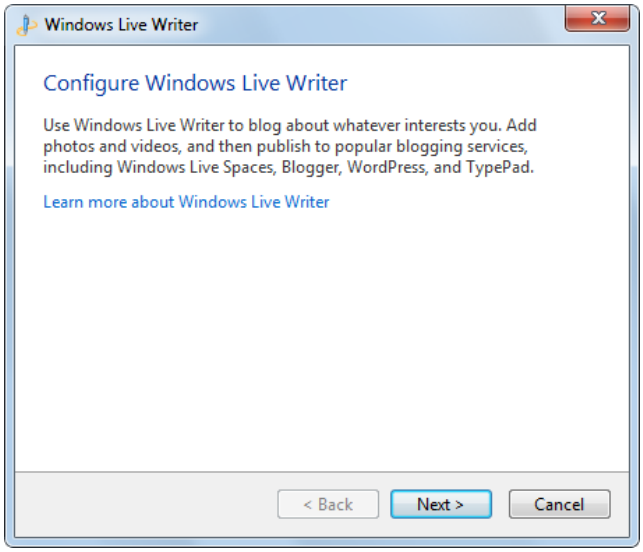


8.10.12.3 Registering blog account

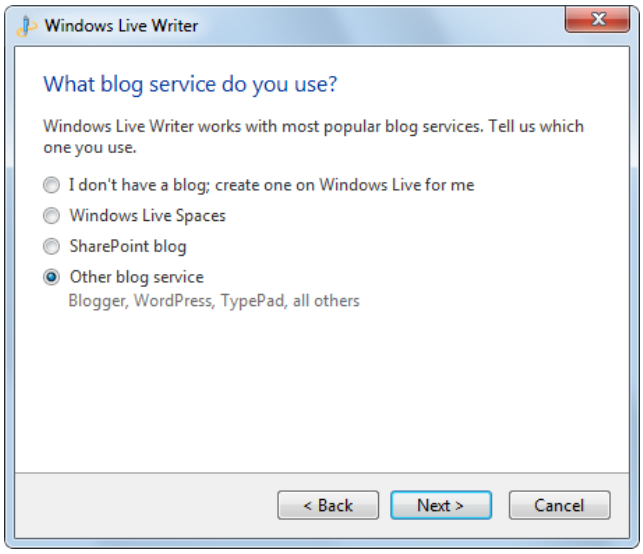
When you run Windows Live Writer for the first time, a wizard will appear, letting you configure connection to your on-line web service. After the initial setup, the wizard is still accessible under the **Tools -> Accounts -> Add** or **Blogs -> Add blog account** menu options.

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.



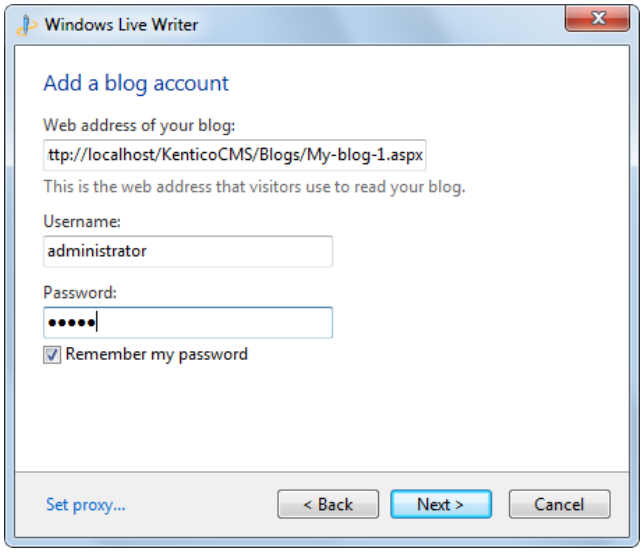
2. This step requires you to select the type of blog service. Select **Other blog service** and click **Next**.



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.



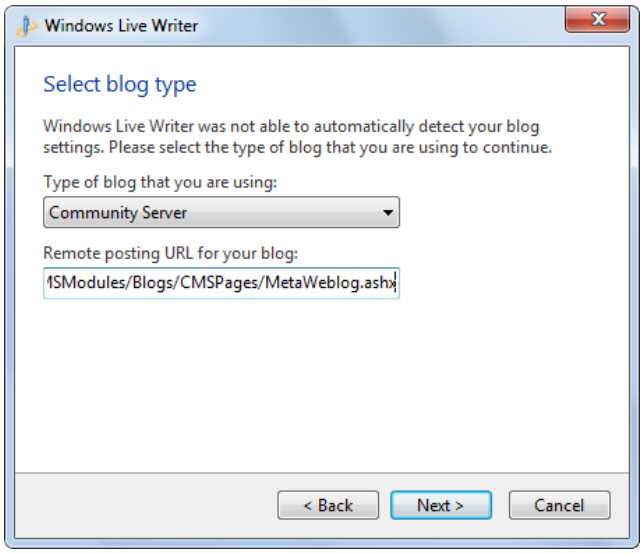
4. In the fourth step, you need to select the type of the blog and the posting URL of the blog. Enter the following details:

- **Type of blog that you are using**: *Community Server* or *MetaWeblog API* (the first one provides

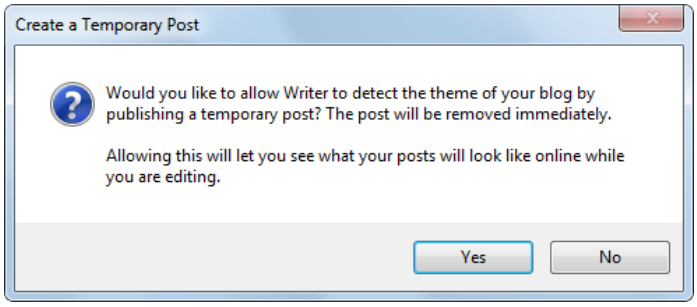
some extra functionality in WLW)

- **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*)

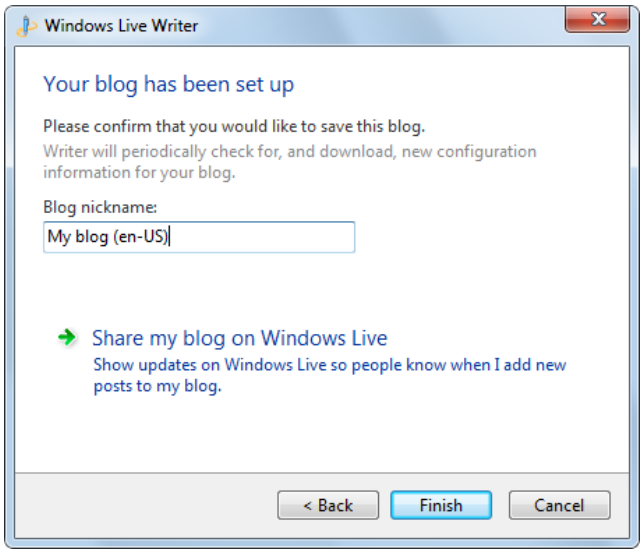
Click **Next**.



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.



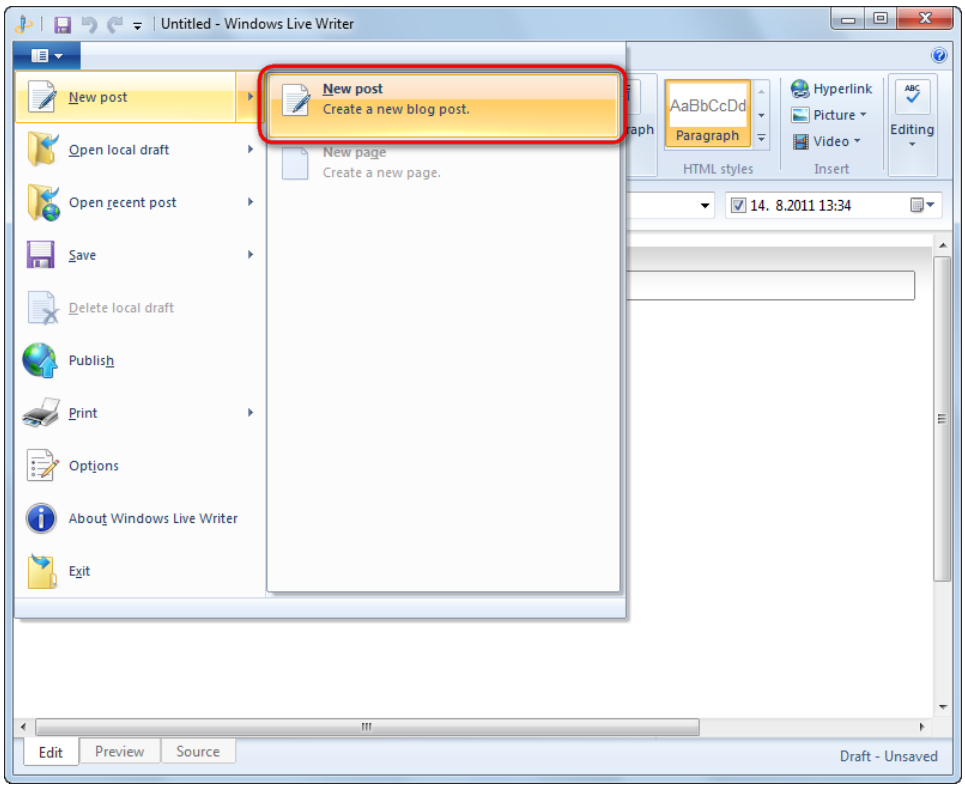
6. In the final step, you just need to enter the **Blog nickname**, which is just a name under which the blog will appear in WLW user interface. Enter whatever you wish and click **Finish**. You blog is now registered and you can start blogging from within Windows Live Writer.



8.10.12.4 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.

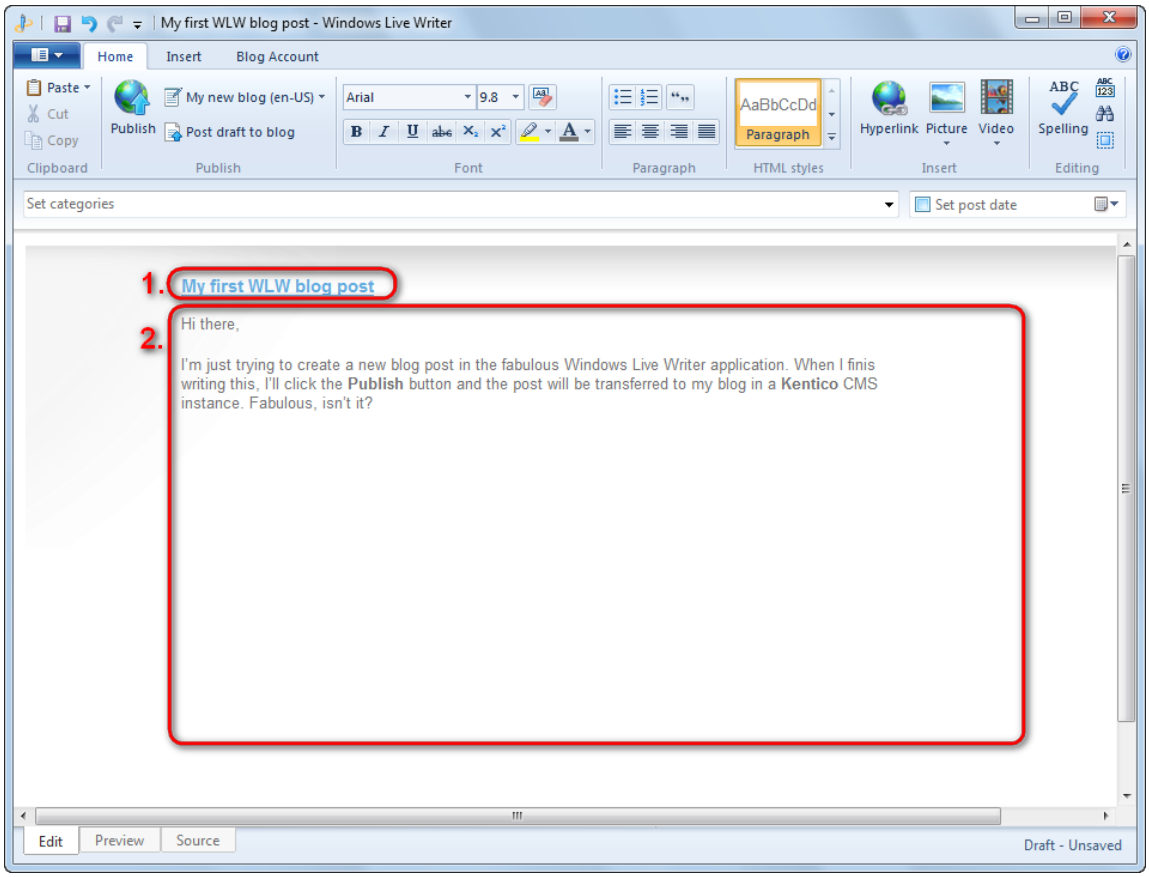
1. Open Windows Live Writer and select **New post** from the **File** menu (or click **Ctrl+N**).



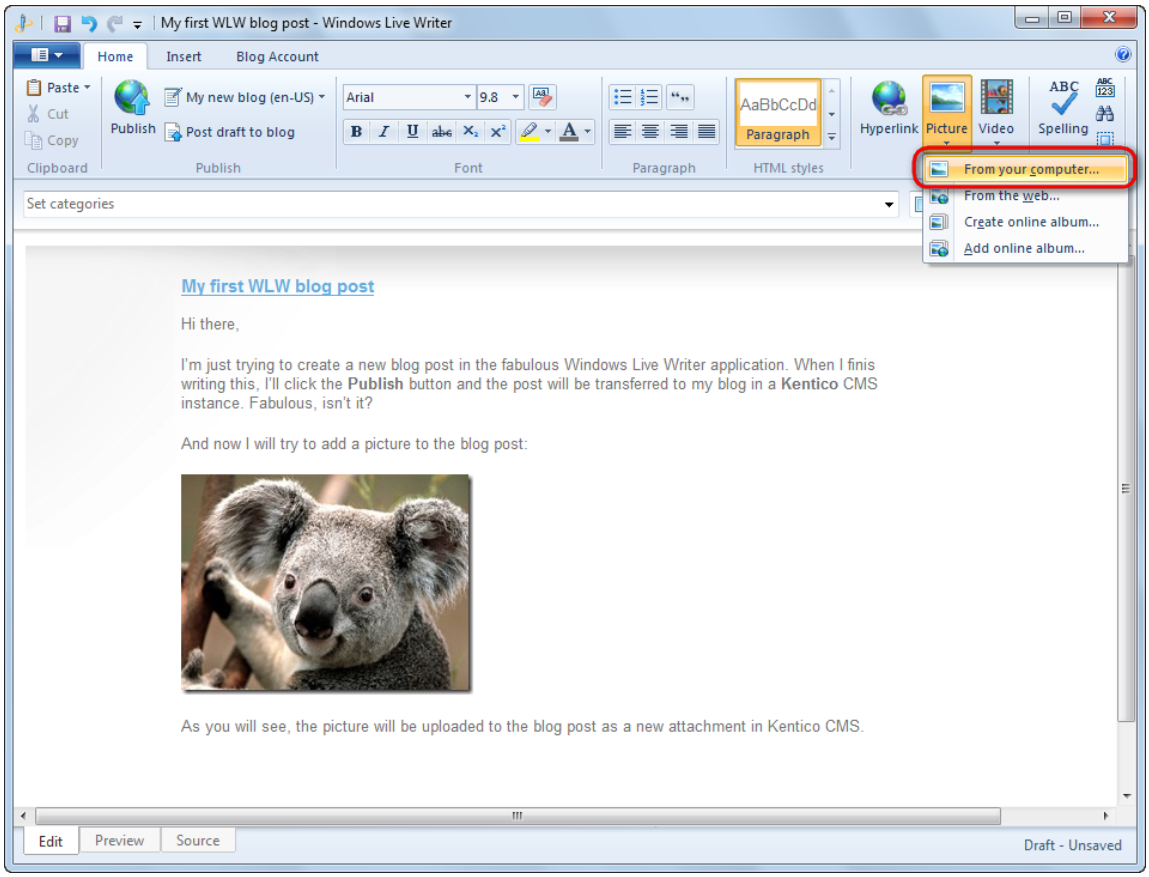
2. The gray rectangle at the top (1.) is where you enter the title of the blog post. What you enter here will be used as a value of the **Post title** field in your Kentico CMS blog post.

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.

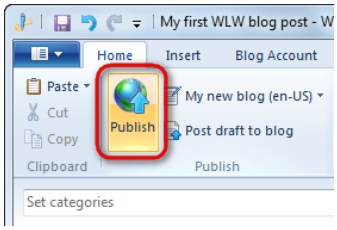
Enter some title and blog post text.



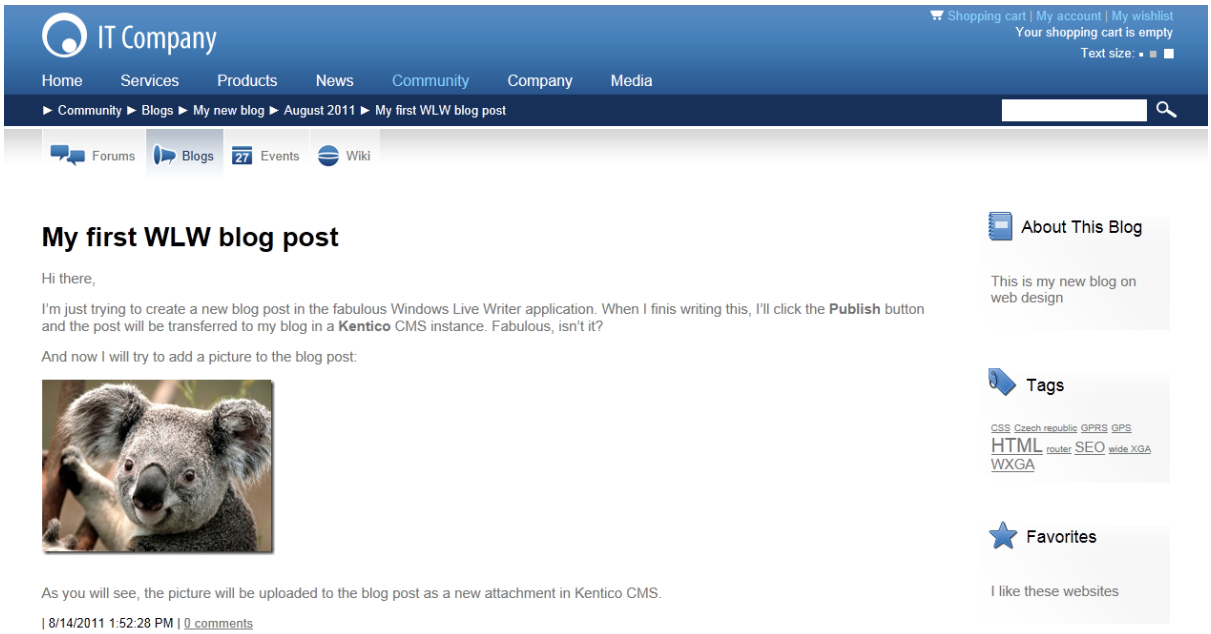
3. Now let's try adding an image. It is quite easy with Windows Live Writer, just click the **Picture...** option in the **Insert** menu on the right and select some image from your local disk. You should achieve that your blog post looks somewhat similar to the screenshot below.



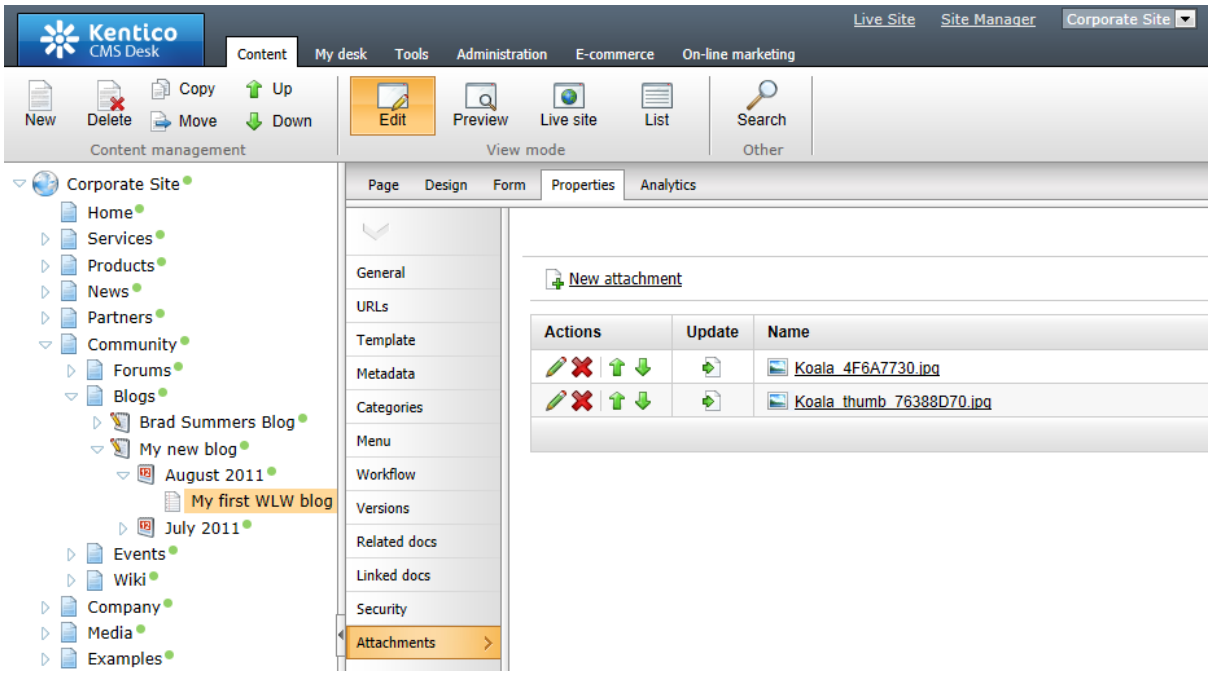
4. Now that your blog post is ready, we can publish it to the live site. Click the **Publish** button in the top toolbar.



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).

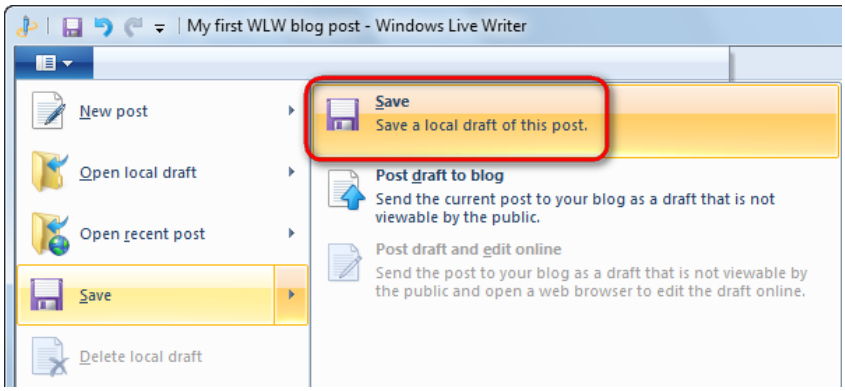


8.10.12.5 Managing blog posts

This chapter describes some additional options that you have when creating or editing a blog post.

Save local draft

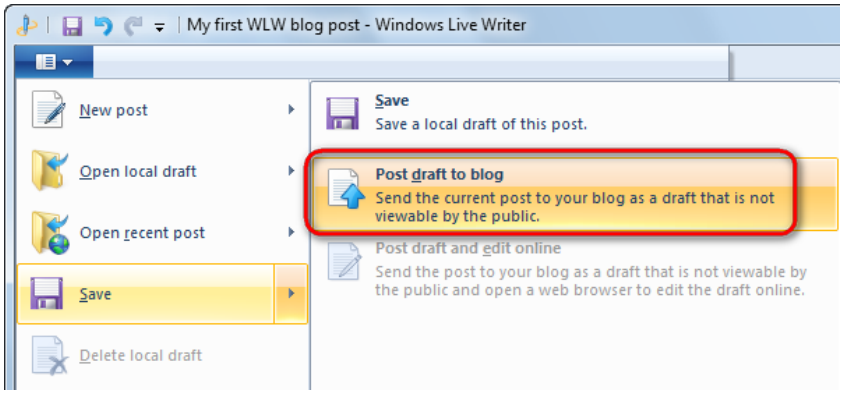
Current local version of the blog post is saved as blog post local draft; each blog post can have only one local draft. This action doesn't propagate any changes into the CMS.



Post draft to blog

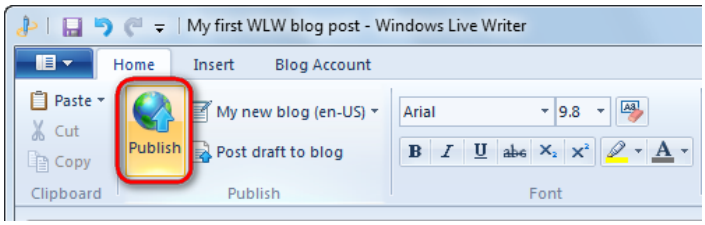
Current local version of the blog post is posted to blog, but its local draft is not removed. Depending on workflow settings, the following two situations can occur in the CMS:

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



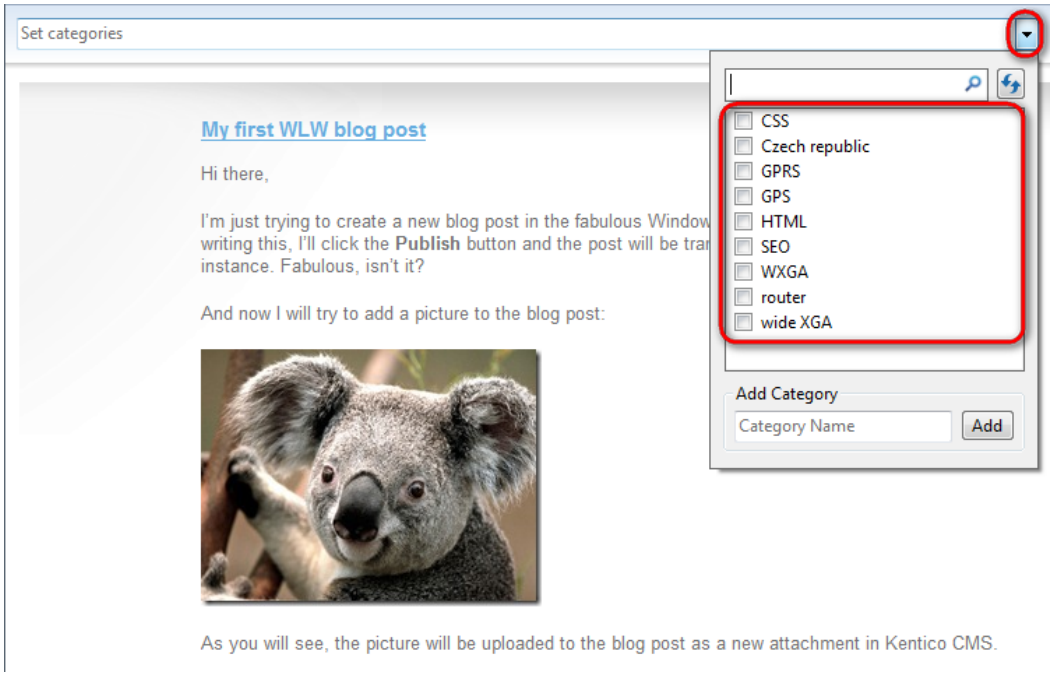
Publish

Current local version of the blog post is posted to blog and published immediately. Local draft of the published blog post is automatically removed.



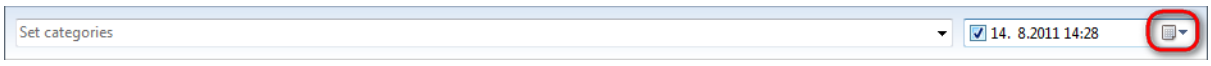
Tagging

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.



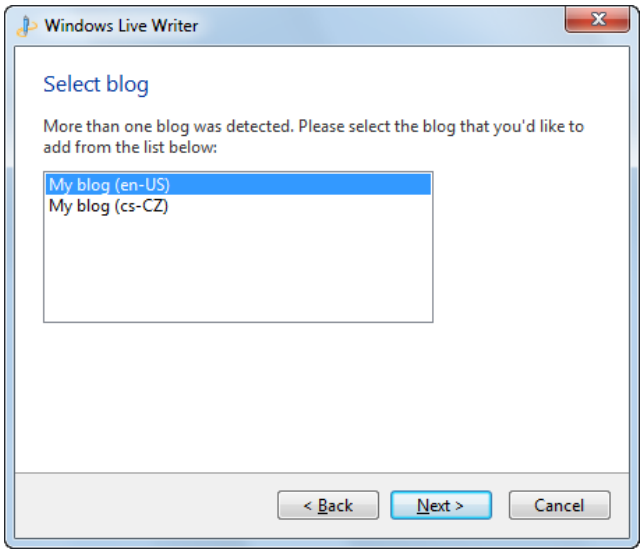
Publish date

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.



8.10.12.6 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.

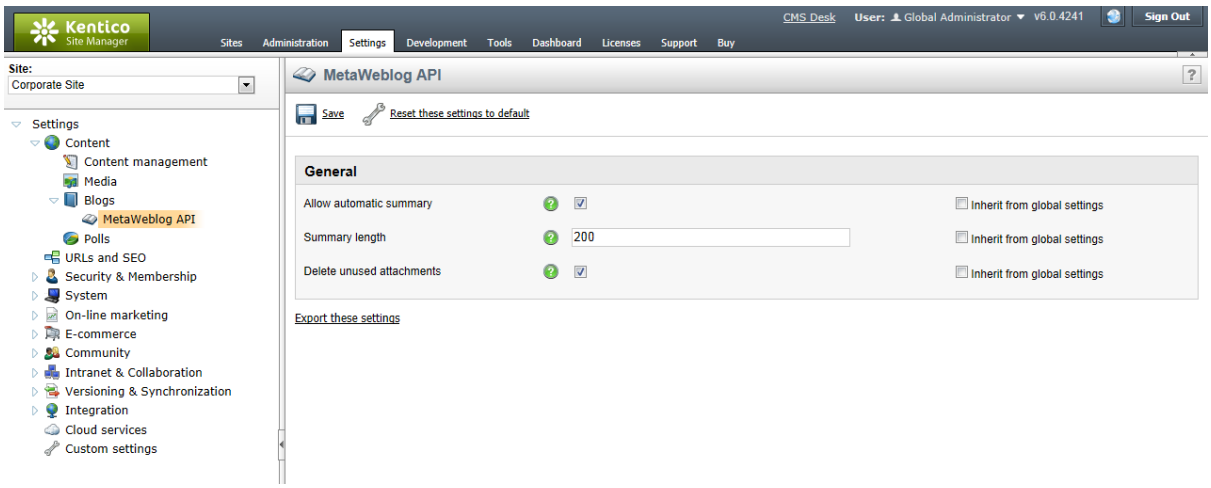


In the dialog, you need to choose which cultural version will the account be registered for. This means that you need to register **one dedicated account for each cultural version** of your blog.

8.10.12.7 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



8.10.12.8 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+, ...).

8.10.13 Blogs internals and API

8.10.13.1 Overview

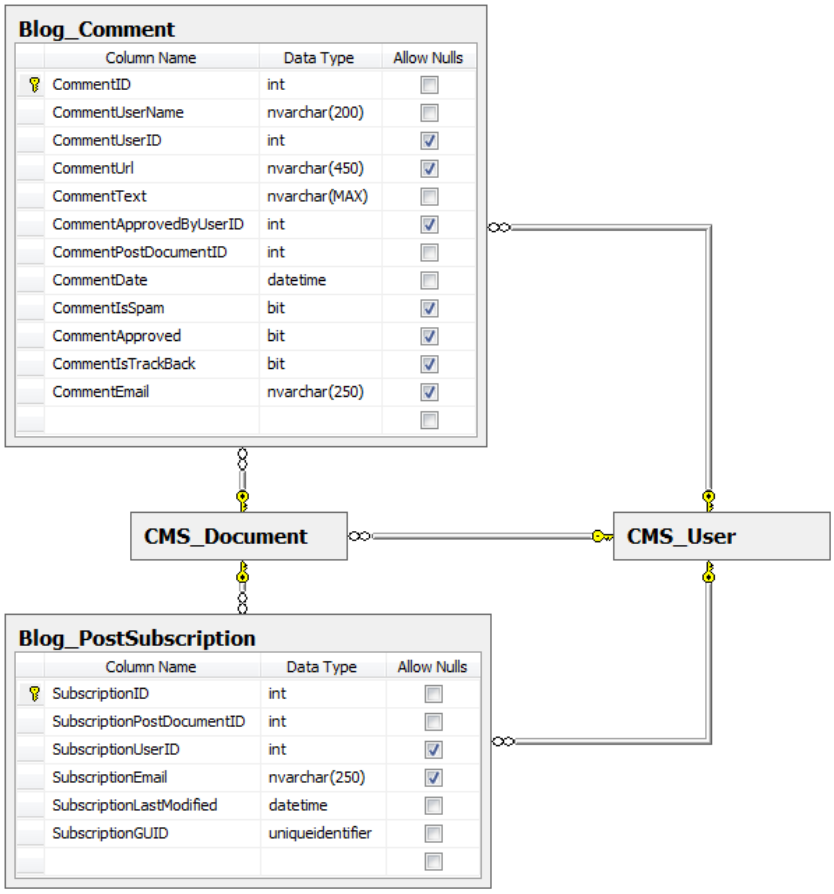
In this chapter, you will learn which database tables and API classes are used in the Blogs module. You will also see the most common API examples.

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.

8.10.13.2 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.



8.10.13.3 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.

8.10.13.4 API examples

8.10.13.4.1 Overview

Please note

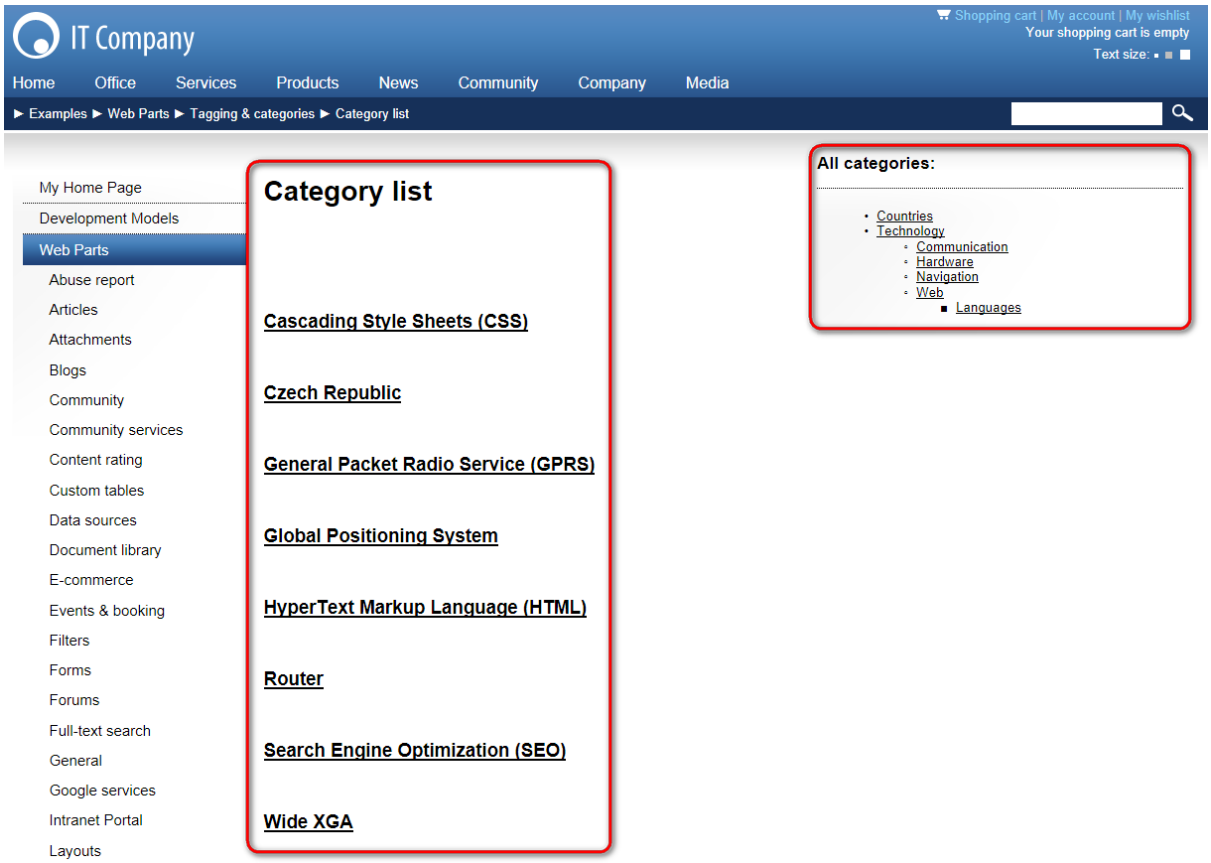
All API examples can be simulated in the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Community\Blogs\Default.aspx.cs***.

8.11 Categories

8.11.1 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.



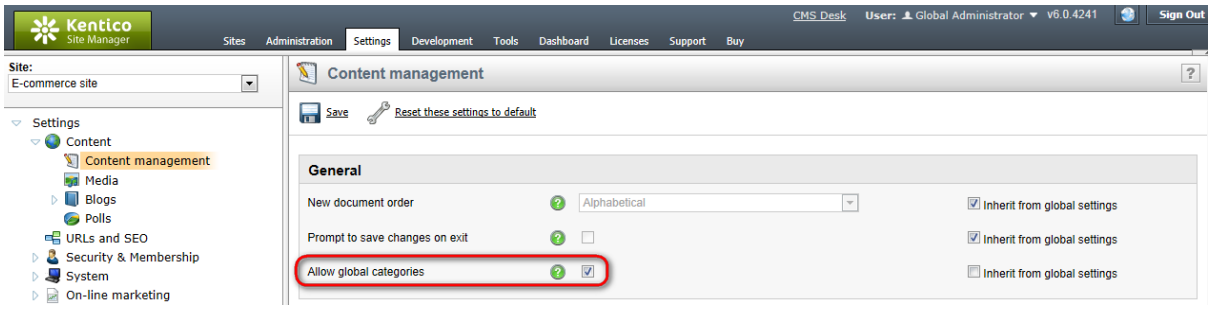
There are three types of categories from the point of view of the user:

- **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.

8.11.2 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.



8.11.3 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

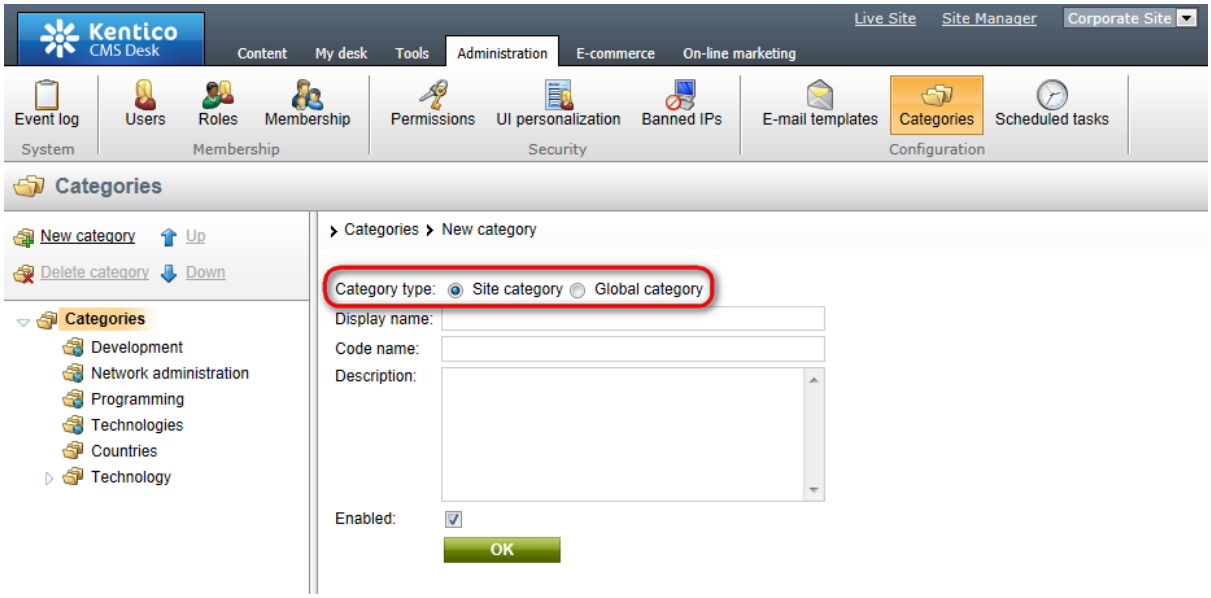
You can create a category using the **New category** link and you can delete one using the **Delete category** link. Categories can also be moved **Up** and moved **Down** in the Categories tree;

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.

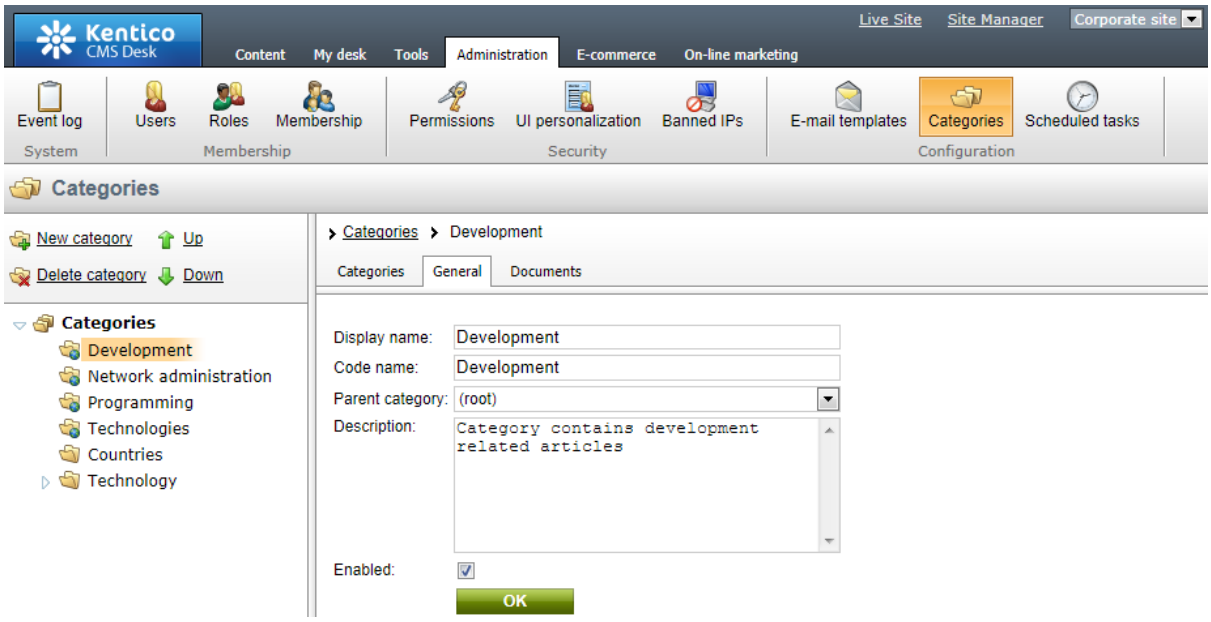
Besides, to view the **Category type** selector and thus be able to create global categories, at least one of the roles of which you are a member must have the **Modify global categories** permission assigned.



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.

8.11.4 Assigning categories to documents

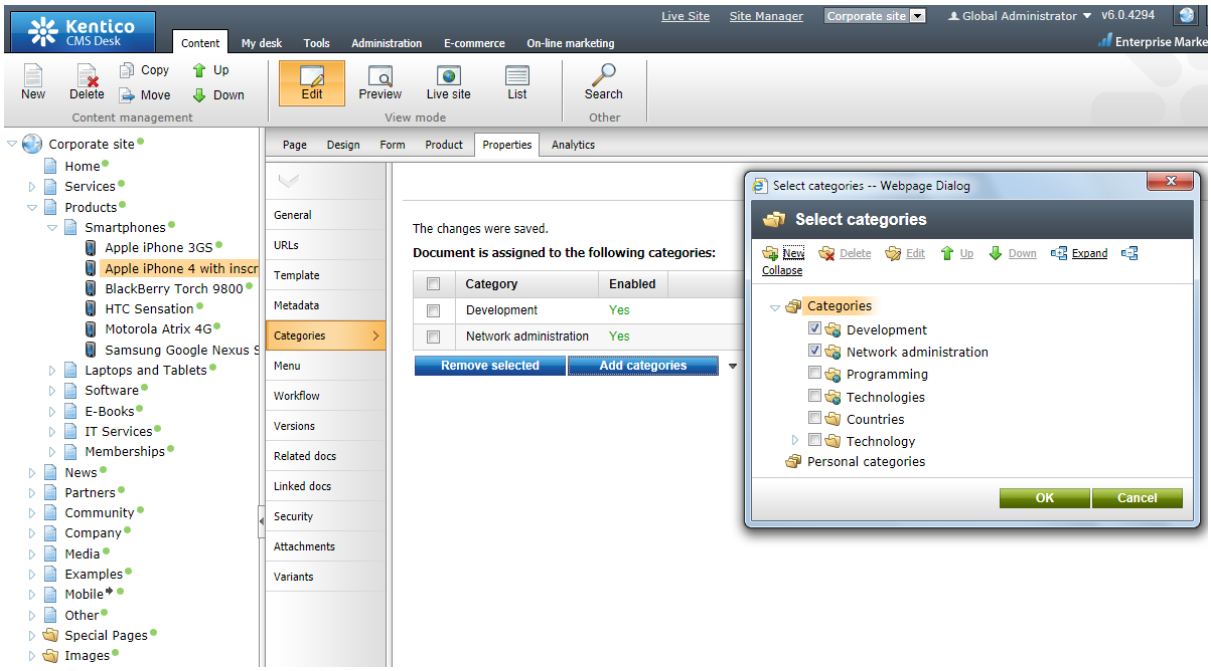
Here you will learn how to assign categories to a document.

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**.

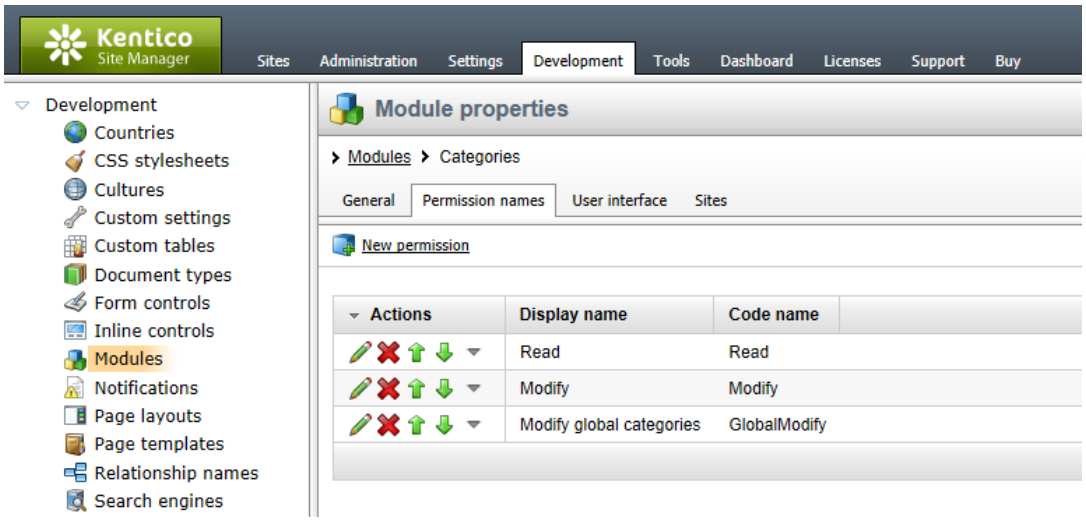
Please note

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.

8.11.5 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.

8.11.6 Categories internals and API

8.11.6.1 Overview

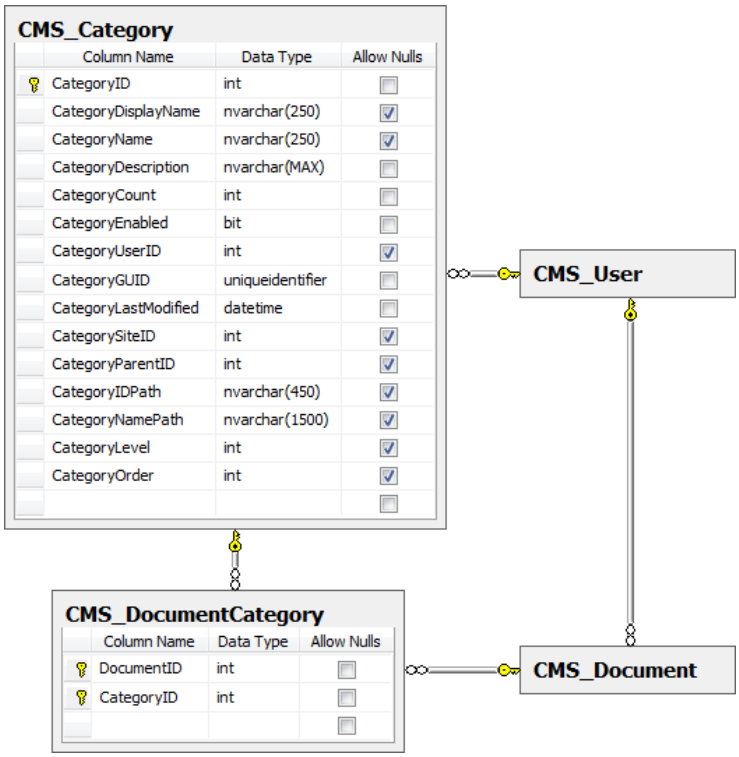
In this chapter, you will learn which database tables and API classes are used in the Categories module. You will also see the most common API examples.

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.

8.11.6.2 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.



8.11.6.3 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.

8.11.6.4 API examples

8.11.6.4.1 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\Development\Categories\Default.aspx.cs***.

8.12 Contact management

8.12.1 Overview

For more information about the **Contact management** module please refer to Kentico CMS Online Marketing Guide.

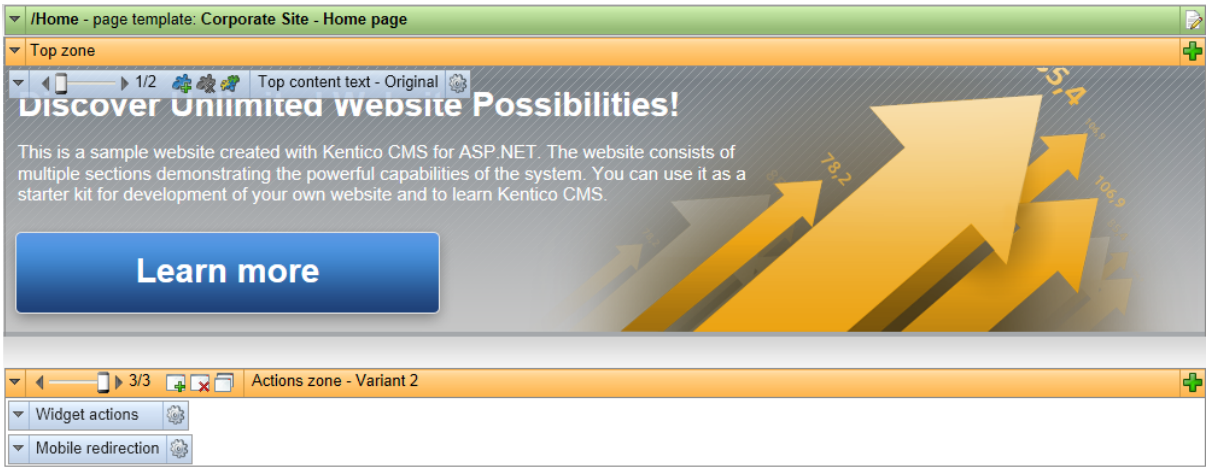
8.13 Content personalization

8.13.1 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

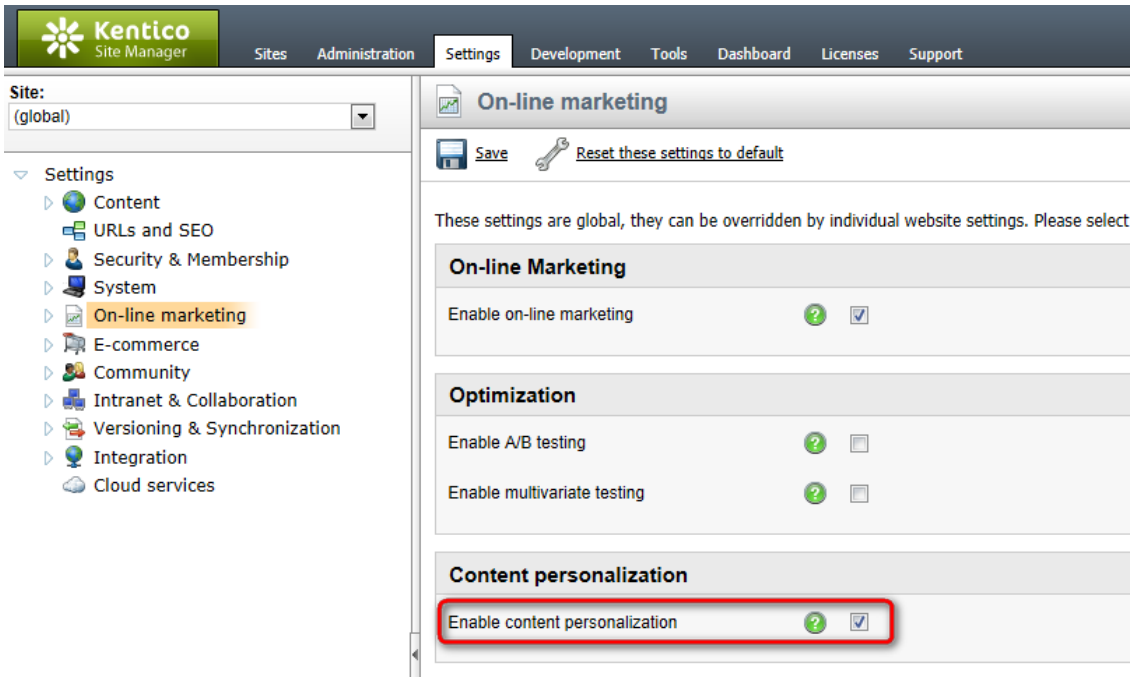
Web parts, entire web part zones and Widgets added into page editor zones. The first necessary step is to define different versions of these objects called *variants*. The details of how personalization variants can be created and configured for objects are described in Managing personalization variants.



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.



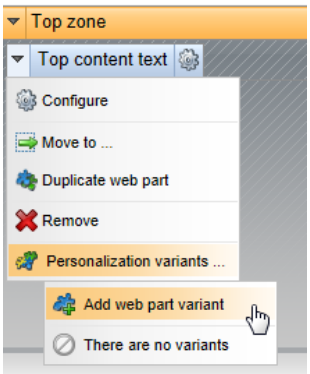
If you disable content personalization at a later time, existing variants on your website will not be deleted, but it will not be possible to manage them or define new ones, and the original objects will always be shown on the live site.

8.13.2 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:

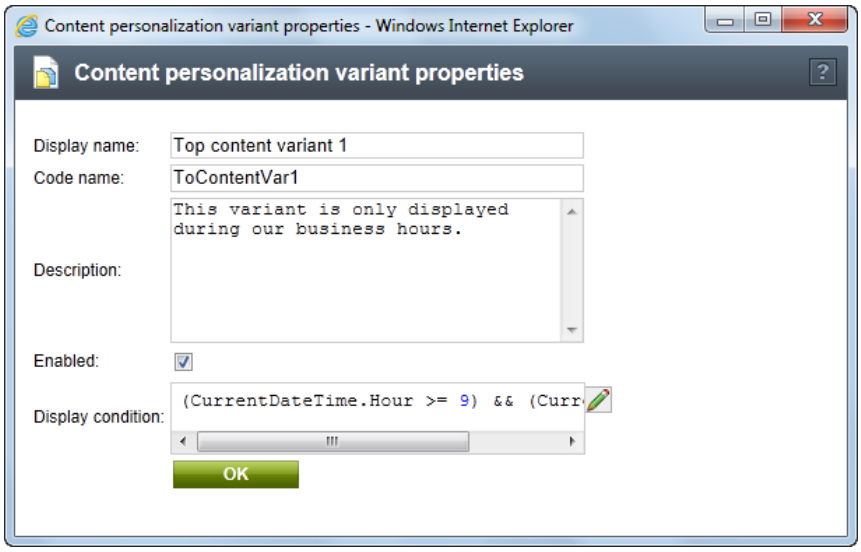
Web parts

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.



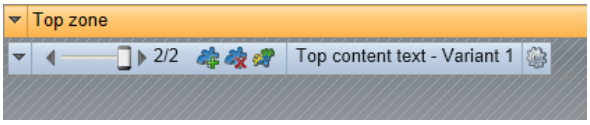
A dialog will be displayed where you can set the properties of the variant:

- **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:

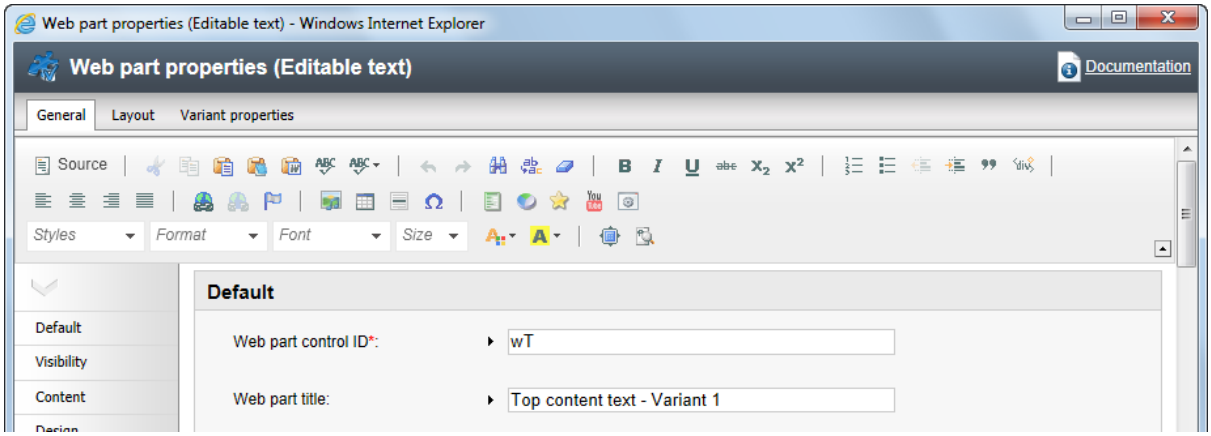


This slider can be used to switch between individual variants as needed (including the original). The content specified by the currently selected variant will be displayed on the **Design** and **Page** tabs, as well as in **Preview** mode. The conditions are only resolved on the actual live site. This 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

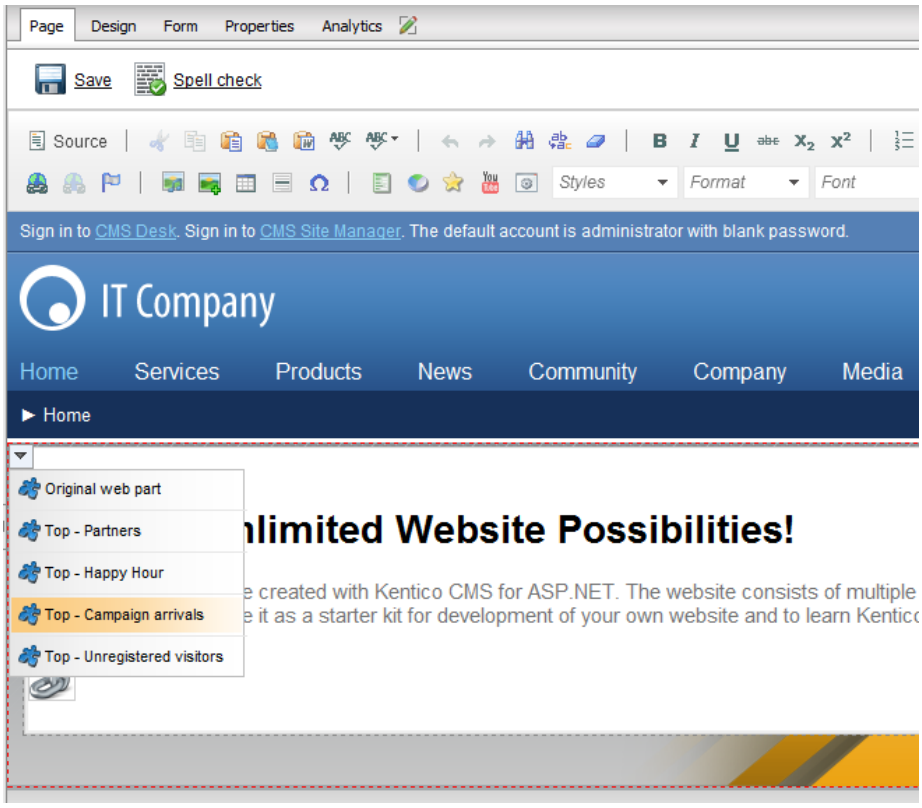
that you can identify which one you are working with straight from the header text.

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.

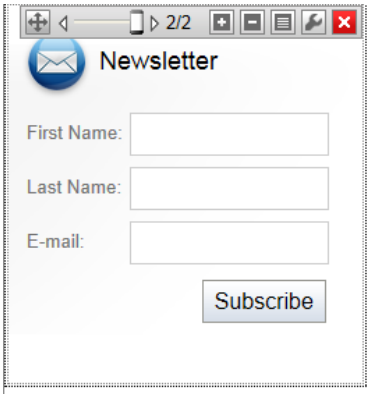
Variants of zones can be created and managed on the **Design** tab in the same way as with web parts (the buttons on the slider use different icons for zones). The only difference is that you need to work with zone objects rather than specific web parts. When a new zone variant is added, the content 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. Please note that using the **Reset to default** action provided by a Widget actions web part will also cause all widget variants to be deleted 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.

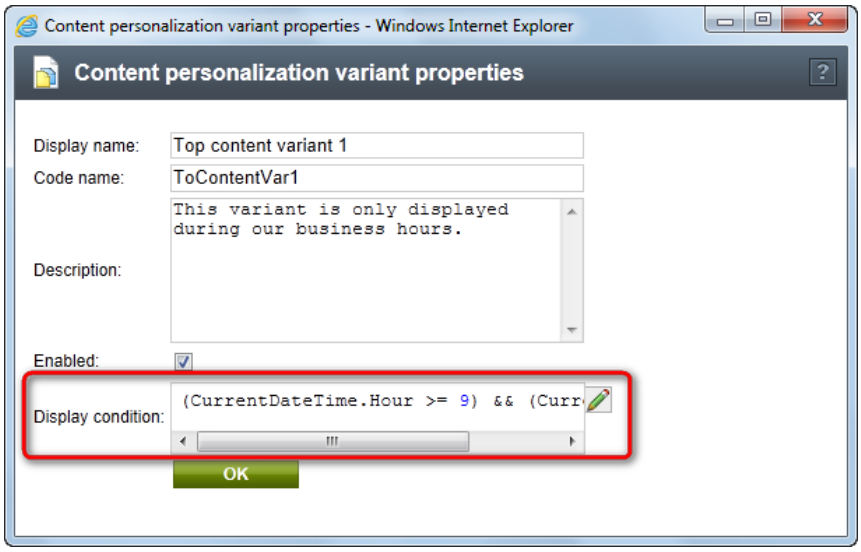
Multivariate testing and Content personalization

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.

8.13.3 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.

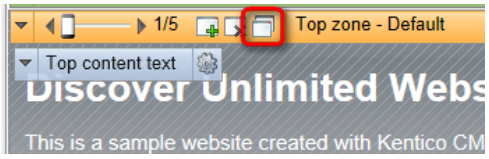
A variant's condition is specified in its **Display condition** property.



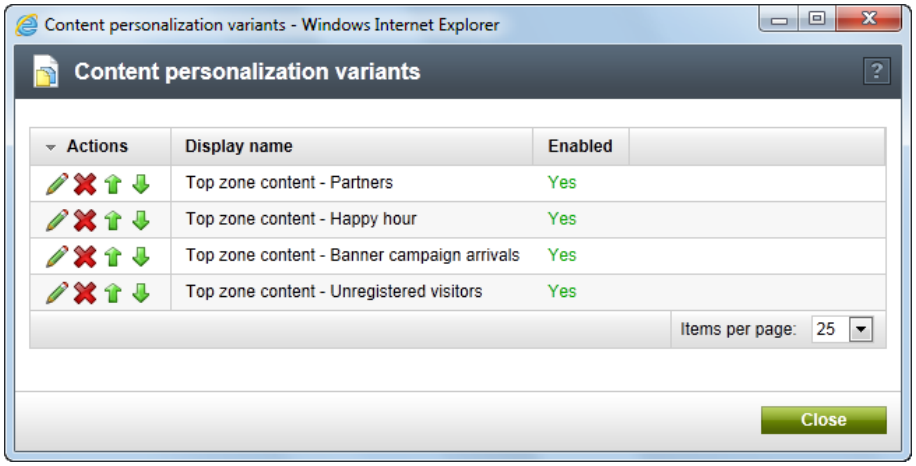
Through the edit icon (), you can use the Macro condition editor, which provides a graphical interface that makes it easier to build complex conditions.

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.

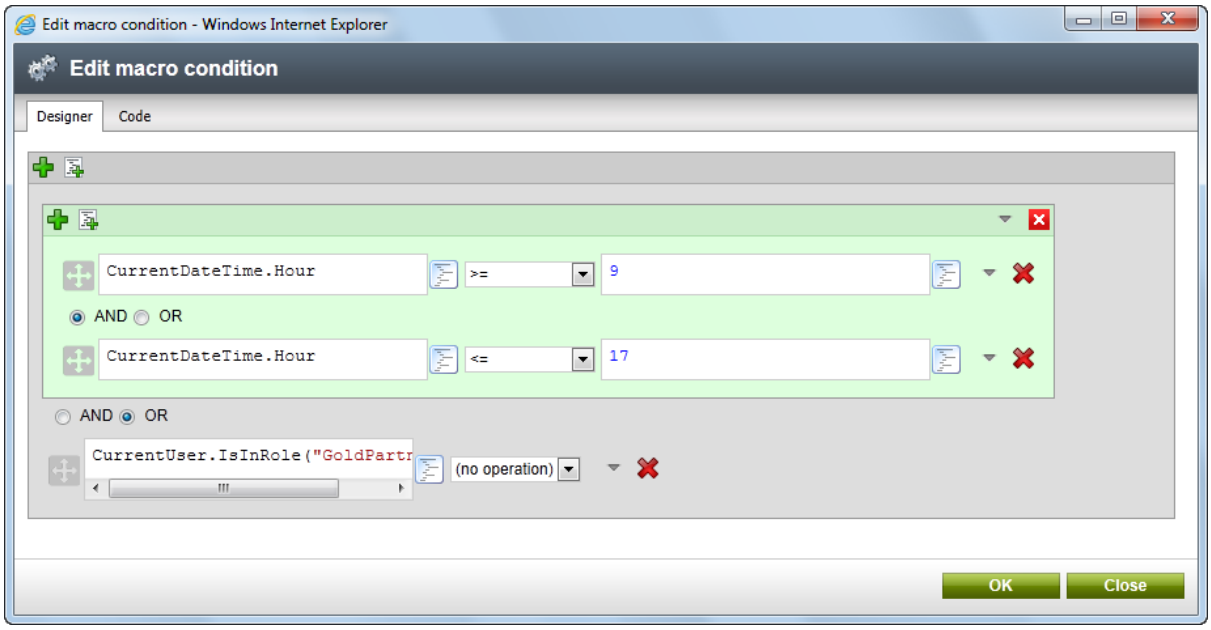
Create a content personalization variant for an object, open its properties dialog and click the edit icon () next to the **Display condition** field to open the macro condition editor. Stay on the **Designer** tab, remove () any existing expressions, then click the **Add Group** () action and use **Add expression** () to create two expression fields inside the new group. Fill in the expressions according to the following:

- Enter *CurrentDateTime.Hour* into the field on the left, select the **>=** operator (greater than or equal to) and write *9* on the right.

- Enter *CurrentDateTime.Hour* into the field on the left, select the **<=** operator (less than or equal to) and write *17* on the right.

Leave the **AND** option selected between the two expressions.

Then, 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 *(no operation).*



If you click **OK**, the final result will look like the following:

((CurrentDateTime.Hour >= 9) && (CurrentDateTime.Hour <= 17)) || CurrentUser.IsInRole ("GoldPartners")

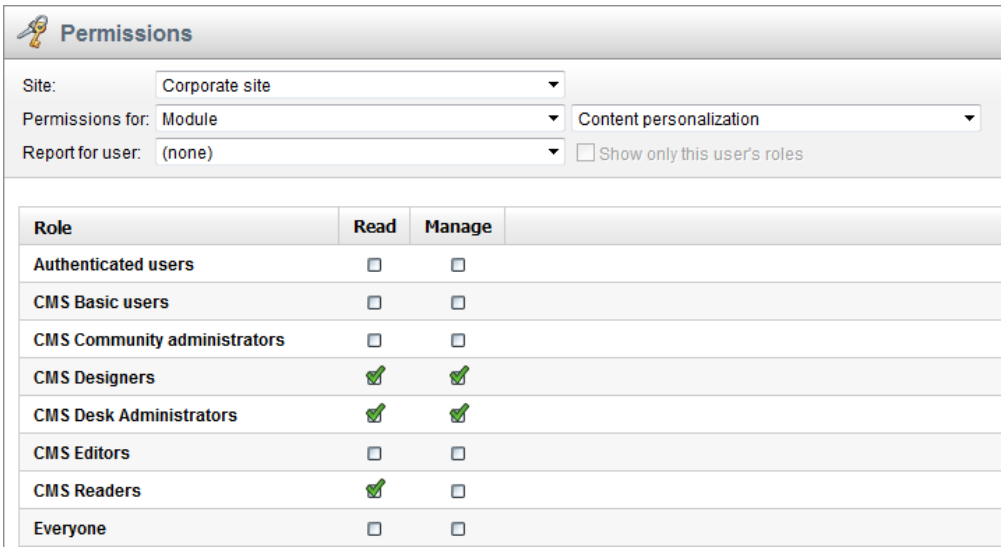
This condition would ensure that the variant is only displayed to ordinary users if the current time is between 9 AM and 5 PM. Users who belong to the *GoldPartners* role will be able to see the content specified by the variant at any time.

8.13.4 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.

8.14 Content rating

8.14.1 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.

8.14.2 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)	3 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 -> Your first news** from the content tree.

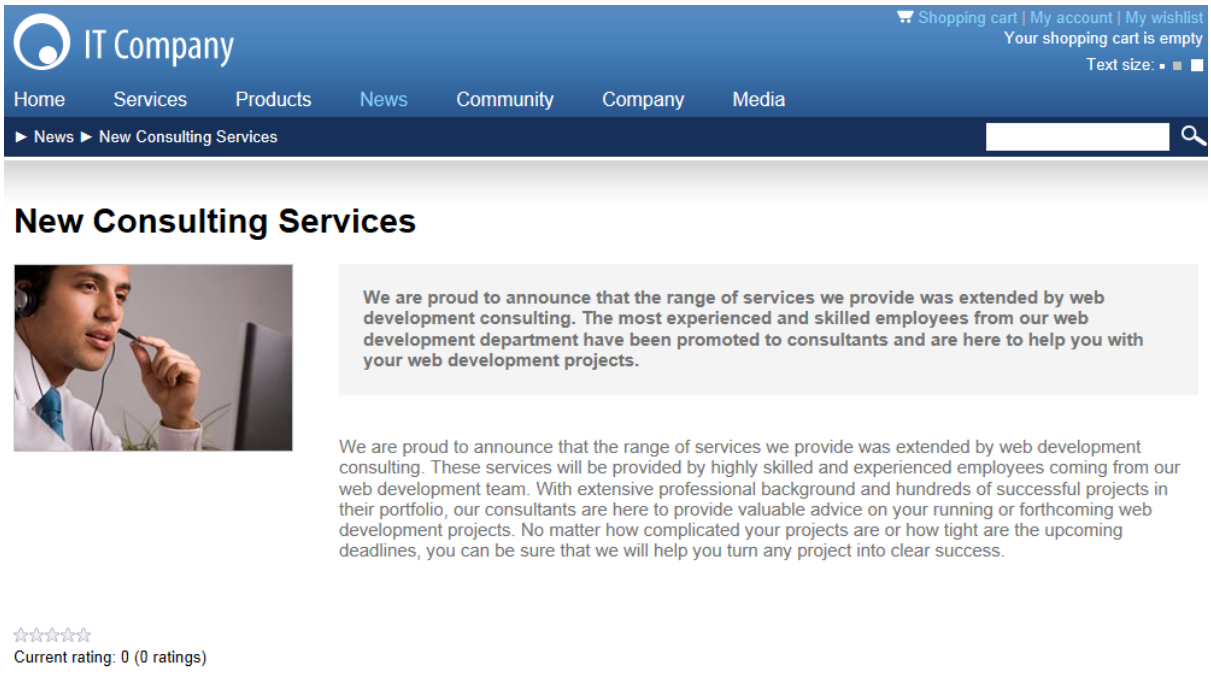
2. Click the **Add web part** (+) icon at the top right corner of the **zoneLeft** web part zone. Select the **Content rating\Content rating** web part and click **OK**.

3. In the **Web part properties** window, just configure the following property:

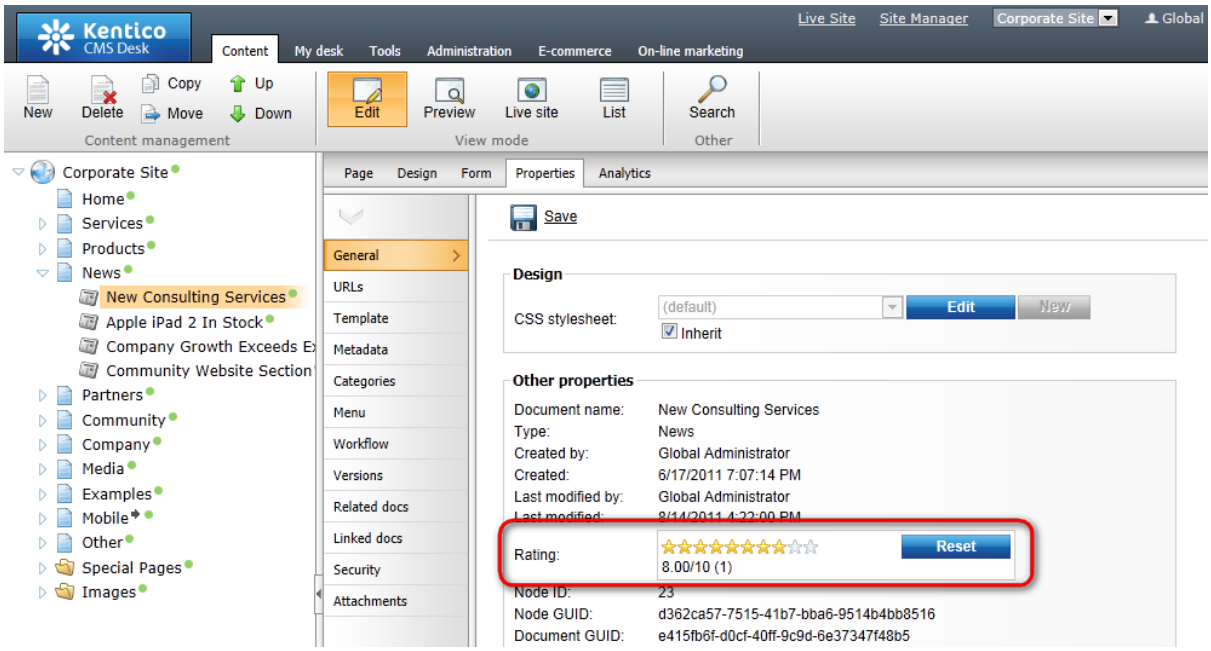
- **Show for document types**: *cms.news*; this ensures that the web part will be displayed only for the particular news items and not for the list of news on the title page of the *News* section.

Leave the rest of the properties at their default values and click **OK**.

4. Now go to the live site and view the **Your first news** 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.



8.14.3 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.

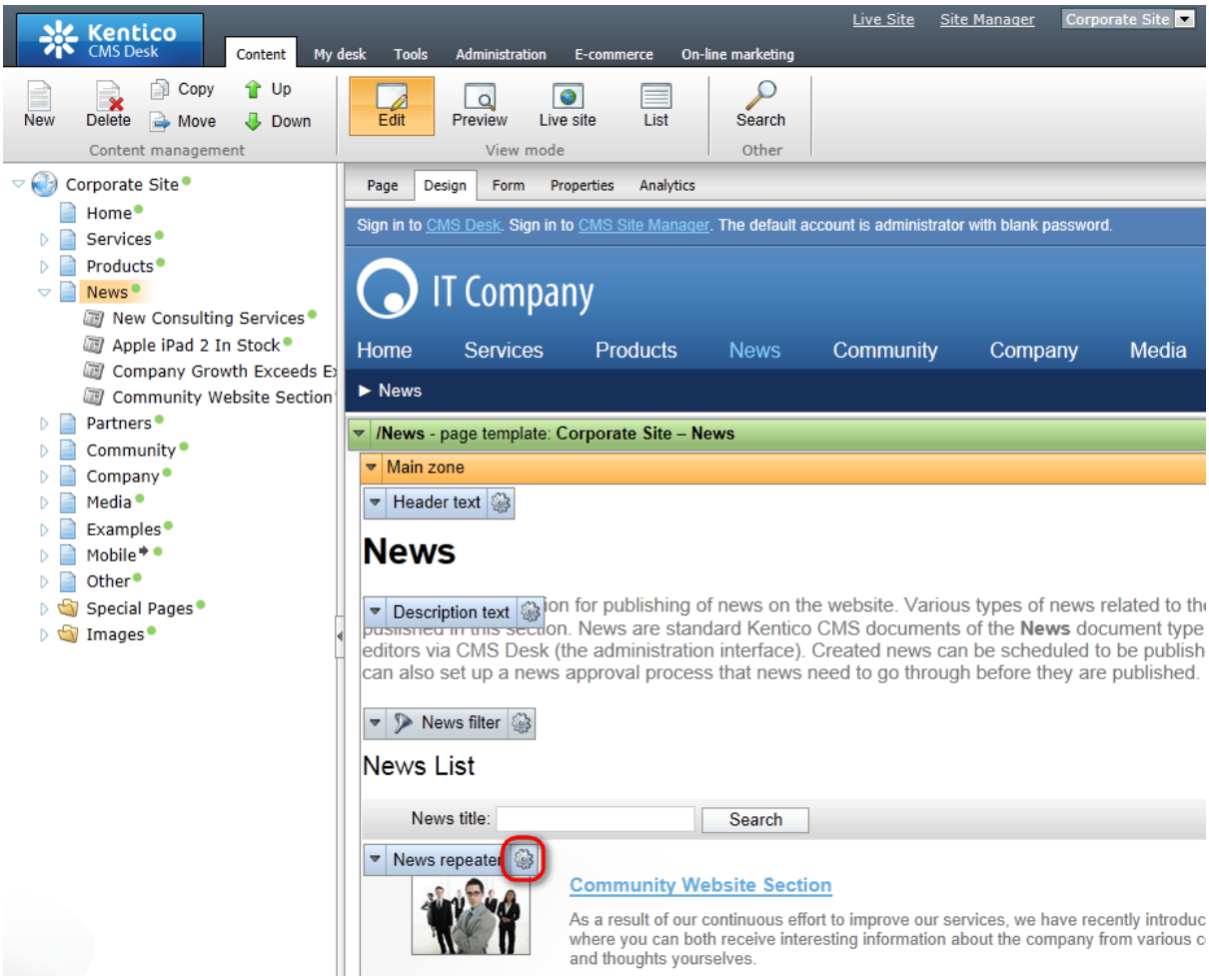
Please note

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.



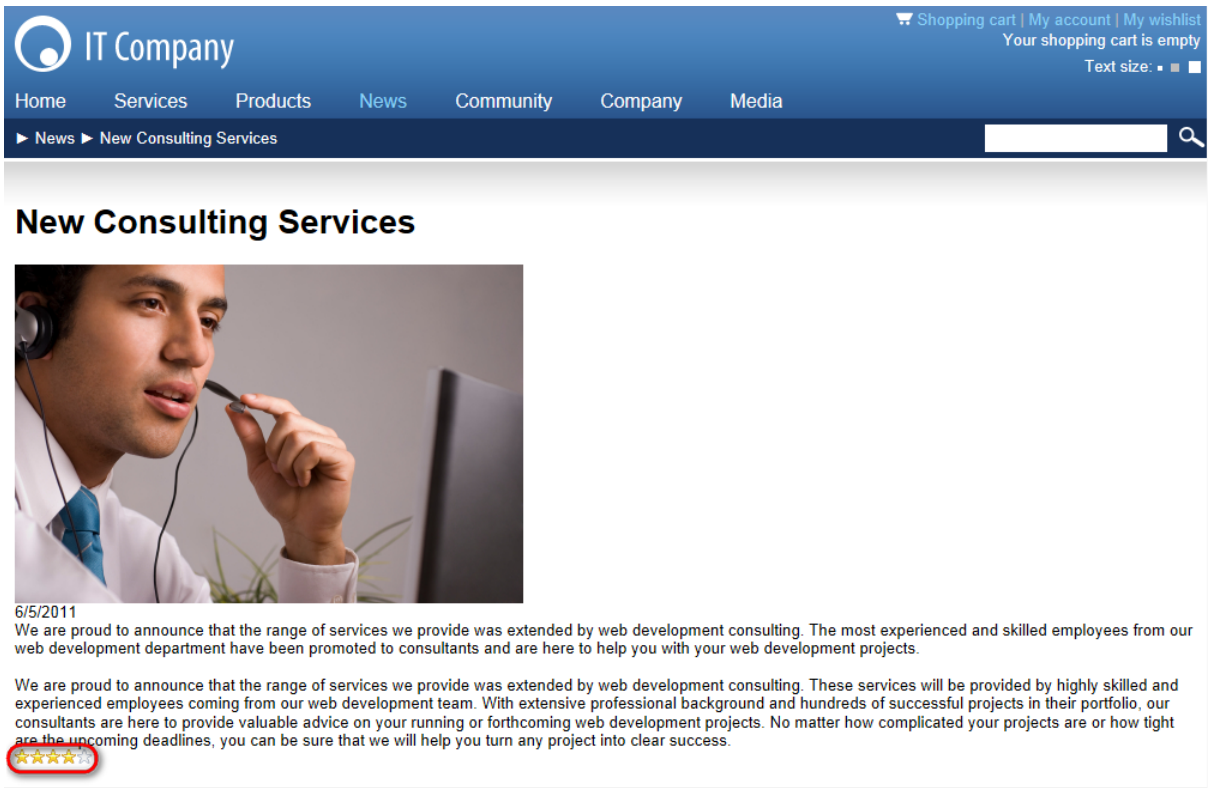
2. In the **Web part properties** dialog which pops up, click the **Edit** button next to the **Selected item transformation** property. Replace the original transformation with the following code, which is the original code with the highlighted parts added.

```
<%@ 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.

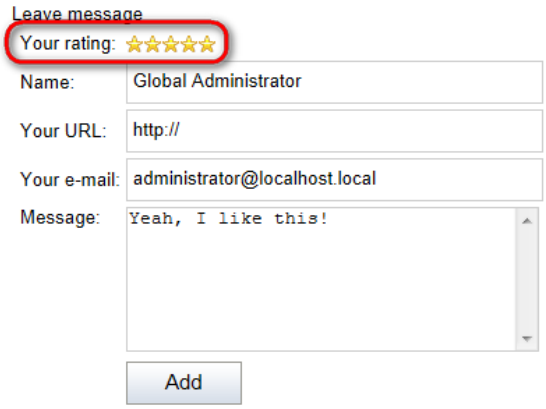


8.14.4 Integration with Message boards

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.



8.14.5 Content rating internals and API

8.14.5.1 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.TreeEngine** 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.

8.14.5.2 Adding document rating

The following code example explains how content rating can be added to a document using the API.

[C#]

```
using CMS.TreeEngine;
using CMS.CMSHelper;

int nodeId = 241;
double rating = 0.5f;

// Get the document
TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);
CMS.TreeEngine.TreeNode node = tree.SelectSingleNode(nodeId, CMSContext.
PreferredCultureCode);

// Add 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);
```

8.14.5.3 Getting document rating

The following code example demonstrates how you can get rating of a document using the API.

[C#]

```
using CMS.TreeEngine;
using CMS.CMSHelper;

int nodeId = 241;
double rating = 0.0f;

// Get the document
TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);
CMS.TreeEngine.TreeNode node = tree.SelectSingleNode(nodeId, CMSContext.
PreferredCultureCode);

// Get its rating
rating = node.DocumentRatingValue / node.DocumentRatings;
```

8.14.5.4 Modifying document rating

The following code example shows how rating of a document can be modified using the API.

[C#]

```
using CMS.TreeEngine;
```

```
using CMS.CMSHelper;

int nodeId = 241;
double rating = 1.0f;
int numOfRatings = 2;

// Get the document
TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);
CMS.TreeEngine.TreeNode node = tree.SelectSingleNode(nodeId, CMSContext.
PreferredCultureCode);

// Modify its rating
TreeProvider.SetRating(node, rating, numOfRatings);
```

8.14.5.5 Resetting document rating

The following code example shows how to reset content rating of a document using the API.

[C#]

```
using CMS.TreeEngine;
using CMS.CMSHelper;

int nodeId = 241;

// Get the document
TreeProvider tree = new TreeProvider(CMSContext.CurrentUser);
CMS.TreeEngine.TreeNode node = tree.SelectSingleNode(nodeId, CMSContext.
PreferredCultureCode);

// Reset its rating
TreeProvider.ResetRating(node);
```

8.15 Custom tables

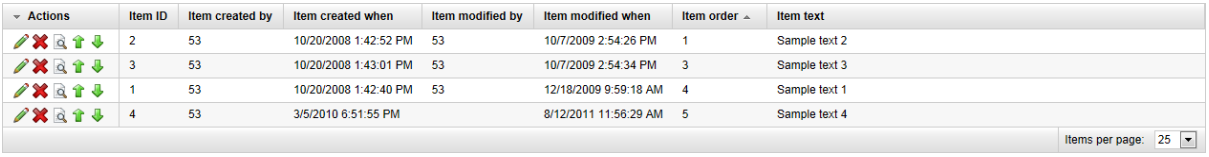
8.15.1 Overview

The Custom tables module allows users to create their own tables in the system database and manage data in them via Kentico CMS user interface, without the need to use Microsoft SQL Server Management Studio or any other database management tool. This may come in handy in many scenarios, typically when you need to store a large number of structured data items, for which standard Kentico CMS documents are not efficient. Another typical scenario may be development of your custom modules, as you can store module data in custom tables and access them conveniently from your code using the provided API.

User interface of the module can be found at two locations, while each of them is intended for different purposes. Custom tables themselves can be created and managed in **Site Manager -> Development -> Custom tables**. This section is intentionally located in Site Manager, so that only global administrators can create the tables. The Creating custom tables topic describes creation of custom tables by means of the **New custom table wizard**, which is accessible from this part of the user

interface. Once a custom table is created, its configuration (not the actual data stored in it) can be managed in the same section of the user interface, as described in the Managing custom tables topic.

The other user interface related to this module can be found in **CMS Desk -> Tools -> Custom tables**. This is where new data records can be inserted into the tables and where existing data records can be viewed, modified or deleted. It is located in CMS Desk, so that manipulation with data in custom tables is available to content editors with appropriate permissions, as described in the Security topic. Management of the actual data items is described in Managing data in custom tables.



Actions	Item ID	Item created by	Item created when	Item modified by	Item modified when	Item order	Item text
	2	53	10/20/2008 1:42:52 PM	53	10/7/2009 2:54:26 PM	1	Sample text 2
	3	53	10/20/2008 1:43:01 PM	53	10/7/2009 2:54:34 PM	3	Sample text 3
	1	53	10/20/2008 1:42:40 PM	53	12/18/2009 9:59:18 AM	4	Sample text 1
	4	53	3/5/2010 6:51:55 PM		8/12/2011 11:56:29 AM	5	Sample text 4

Items per page: 25

The module comes with three web parts that can display data from custom tables on the live site. They are described in the Available web parts topic. Use of transformations is essential with these web parts, so we have prepared the Transformations for custom tables topic. There, you can find information on how to modify the default transformations in order to change the way the data items are displayed.

Finally, the module comes with an API, which enables you to handle custom table data in your custom code. Code examples, along with a list of database tables and API classes related to the module, can be found in the Custom tables internals and API sub-chapter.

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.

8.15.2 Creating custom tables

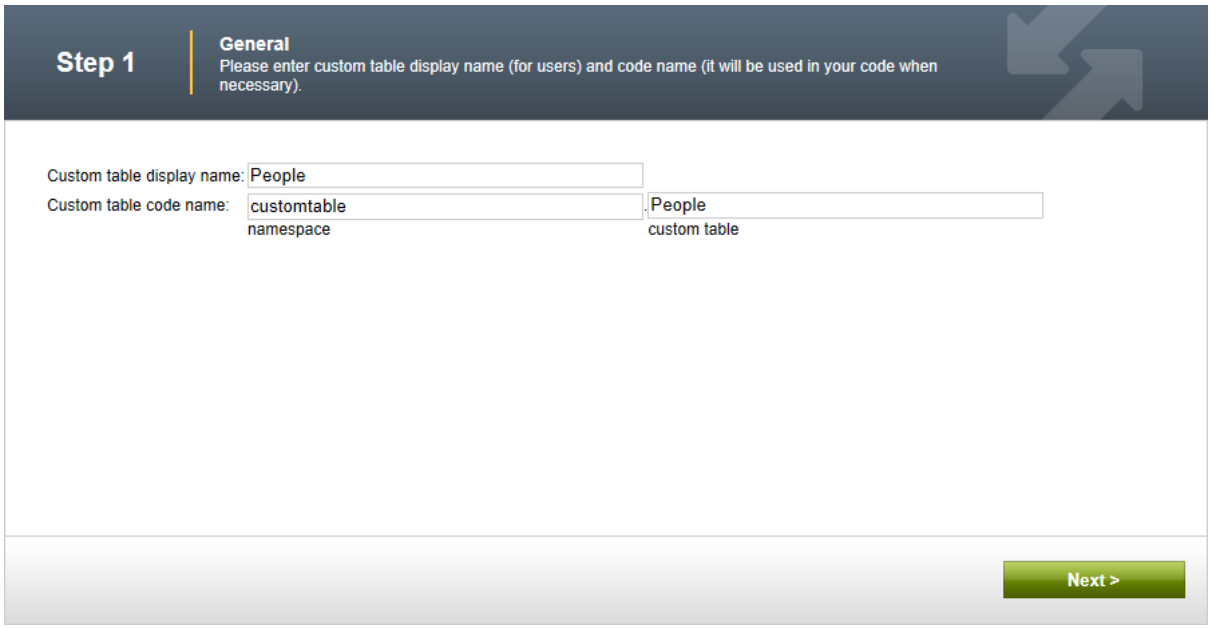
In the following example, we will create a new custom table using the **New custom table wizard**. It will be a simple table for storing information about people. Each data record in the table will contain a person's first name, last name and date of birth.

1. Go to **Site Manager -> Development -> Custom tables** and click the **New custom table** ( ) link.

2. The **New custom table wizard** starts. In the first step, you are asked to fill in the custom table's display name (used in Kentico CMS user interface) and code name (used in website code). The code name is always preceded by a namespace, which allows you to have different tables of the same name used in different contexts. Enter the following details:

- **Custom table display name:** People
- **Custom table code name:** customtable.People

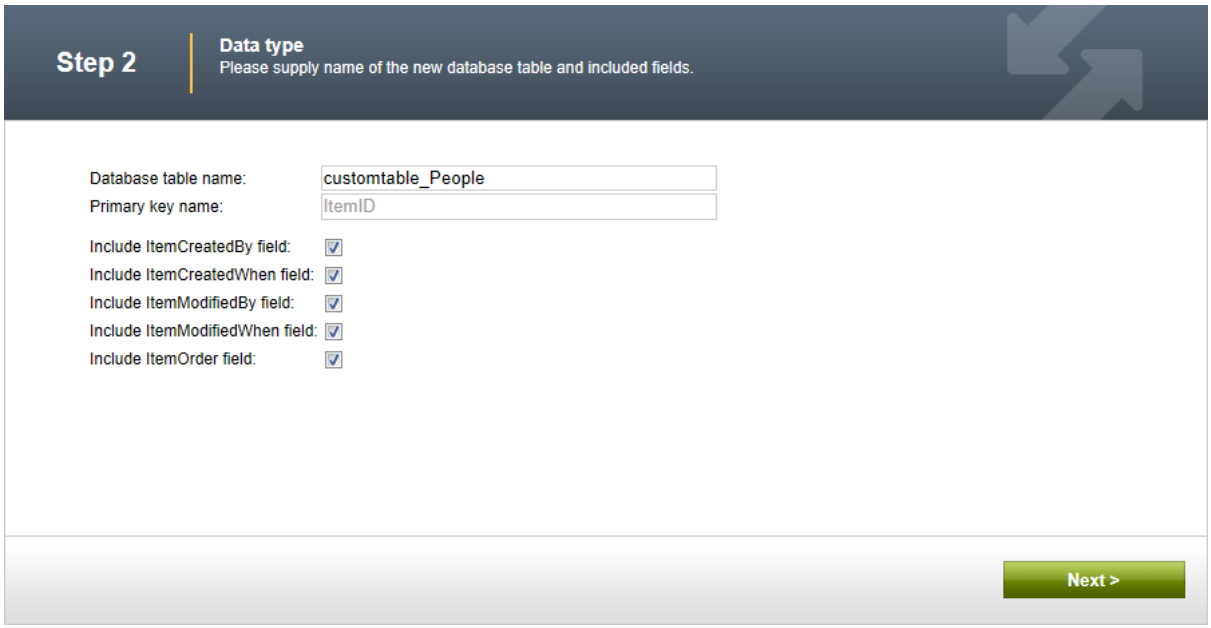
and click **Next** to continue.



3. In the second step, you are asked to fill in the database table name, i.e. the actual name of the table in the system database. A name in the <*namespace*>_<*code name*> format is pre-filled automatically. The primary key name cannot be changed by default. Using the check-boxes below, you can determine which of the default fields should be included in the table:

- **ItemCreatedBy** - user name of the user who created the item
- **ItemCreatedWhen** - date and time of when the item was created
- **ItemModifiedBy** - user name of the user who last modified the item
- **ItemModifiedWhen** - date and time of last modification
- **ItemOrder** - order of the item when table content list is displayed; the lower number, the earlier position in the list

For the purposes of this example, leave default values for all options and click **Next** to continue.



4. In the third step, the field editor is displayed. It lets you define which columns will be included in the database table. Create three fields using the **New attribute** ( ) button with the following properties:

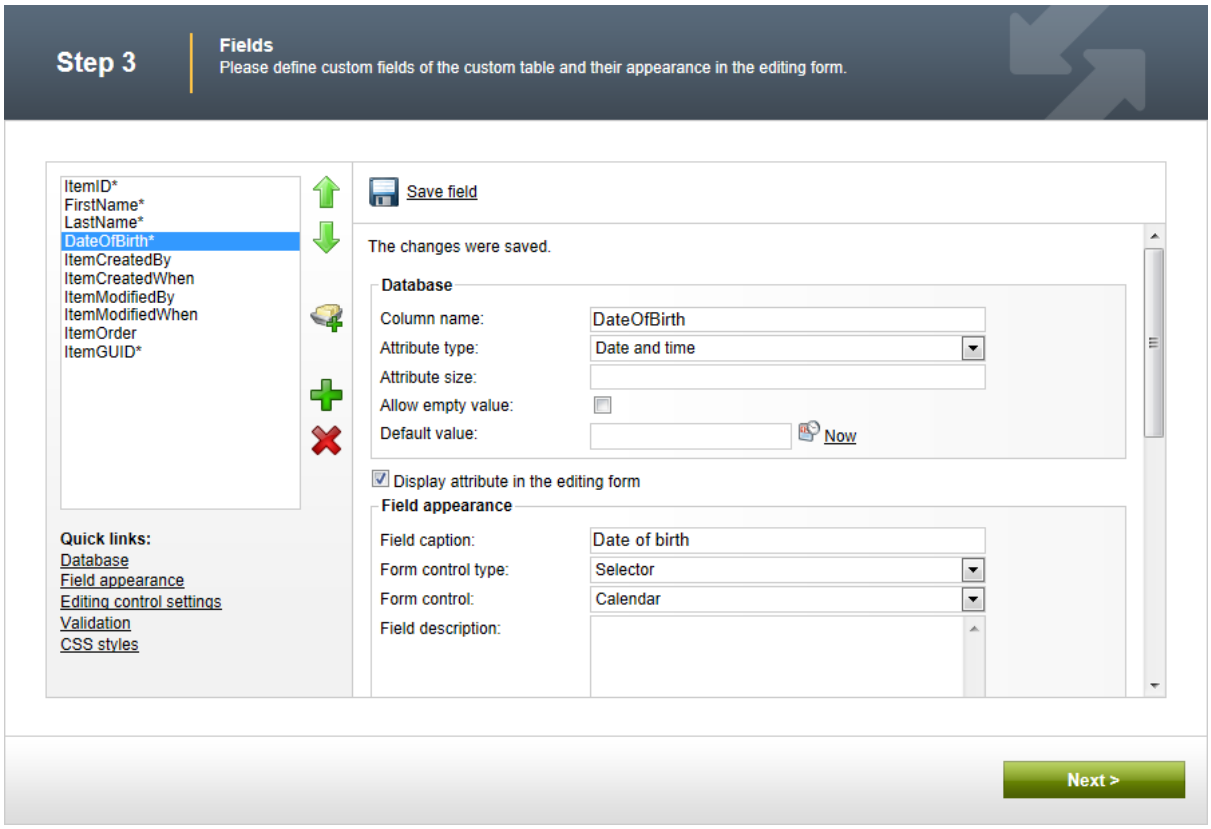
- **Column name:** FirstName
- **Attribute type:** Text
- **Attribute size:** 100
- **Field caption:** First name
- **Form control type:** Input
- **Form control:** Text box

- **Column name:** LastName
- **Attribute type:** Text
- **Attribute size:** 100
- **Field caption:** Last name
- **Form control type:** Input
- **Form control:** Text box

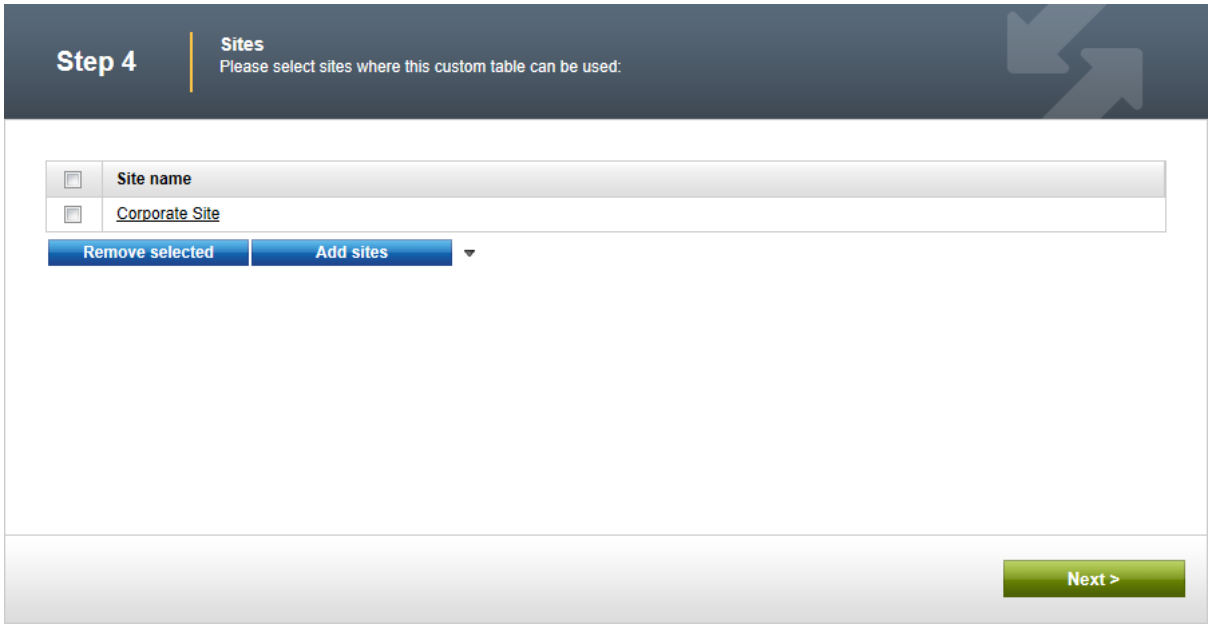
- **Column name:** DateOfBirth
- **Attribute type:** Date and time
- **Field caption:** Date of birth
- **Form control type:** Selector
- **Form control:** Calendar
- **Editing control settings -> Edit time:** unchecked

It is necessary to click the **Save field** button to create each individual field.

When you are finished, click **Next** to proceed to the following step.



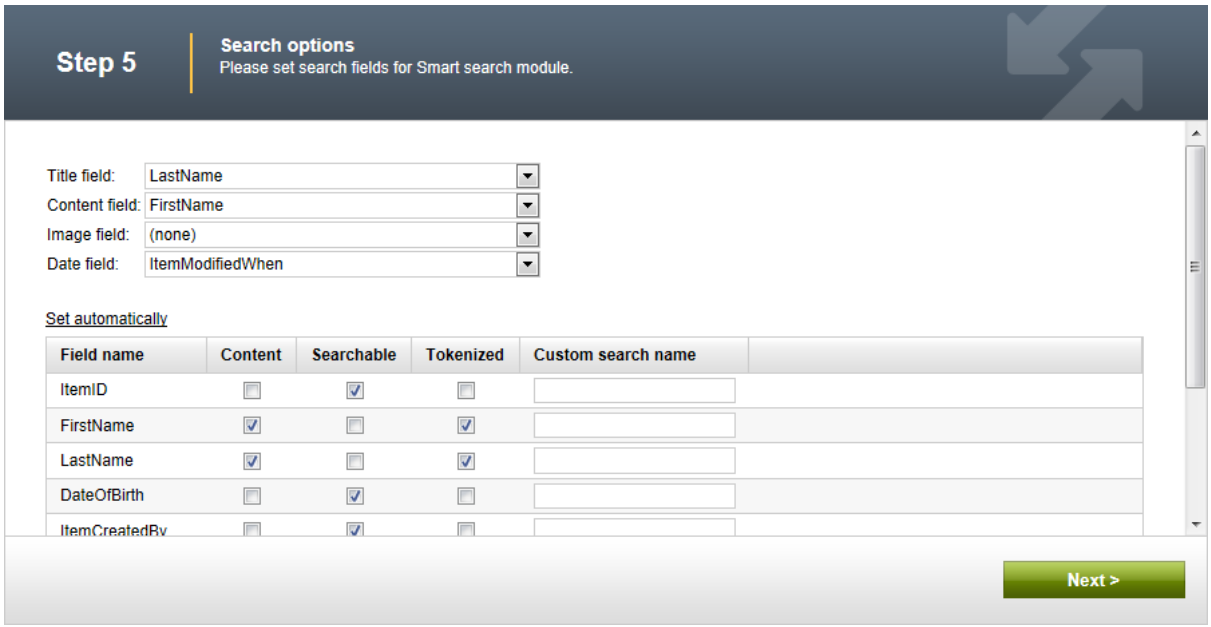
5. In Step 4, you only have to select in which sites will the custom table be available. Click **Add sites** and choose **Corporate Site** in the pop-up dialog. Finally click **Next** to proceed to the following step.



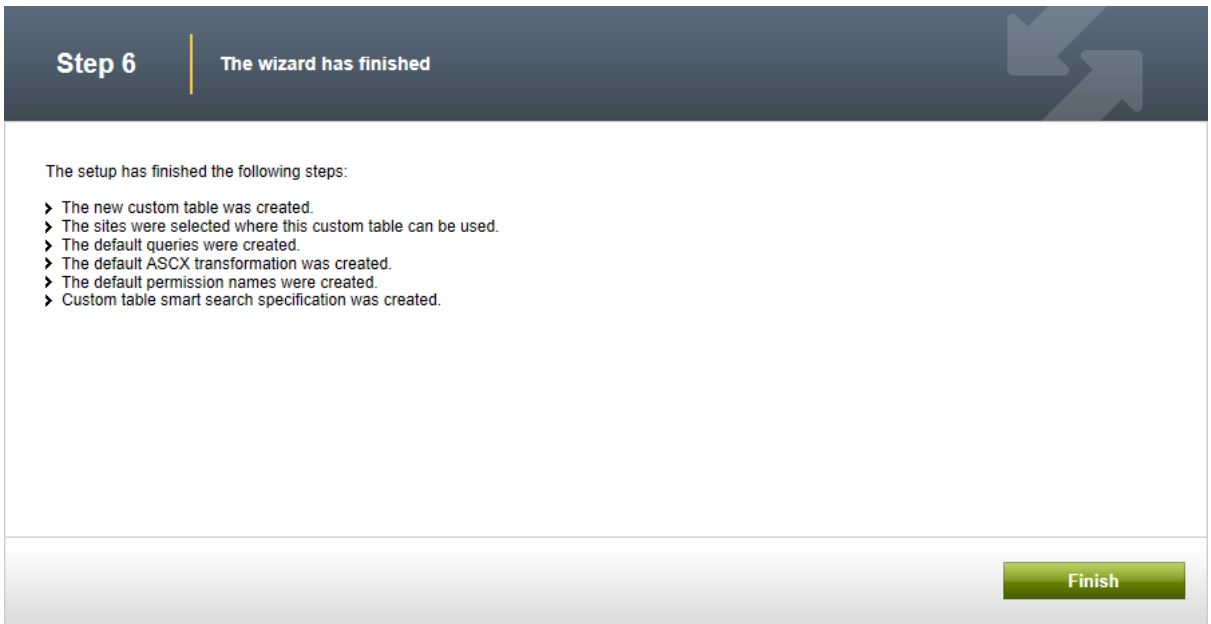
6. Step 5 offers you the option of configuring the indexing of data in the custom table by the Smart Search module. The four drop-downs at the top determine which fields of the table will be displayed in search results. Change the values to the following:

- **Title field**: LastName
- **Content field**: FirstName
- **Image field**: none
- **Date field**: ItemModifiedWhen

Leave the rest of the settings at default values and click **Next**. For more information about settings in this step, please refer to Modules -> Smart search -> Object settings.



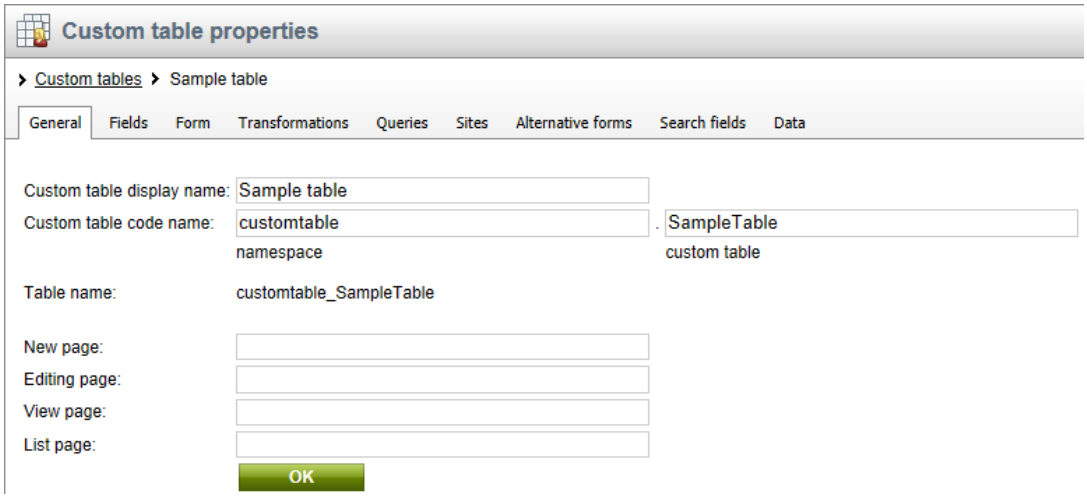
7. The fifth step gives you an overview of what has been done automatically by the wizard. To finish the wizard, click **Finish**. You will be redirected back to the list of custom tables in **Site Manager -> Development -> Custom tables**.



8.15.3 Managing custom tables

The administration interface for management of custom tables is located in **Site Manager -> Development -> Custom tables**.

In this section, you can create new custom tables as described in the Creating custom tables topic. You can also **Edit** (✎) or **Delete** (✖) existing tables in the list. If you choose to edit (✎) an existing custom table, you will be displayed with the user interface depicted in the screenshot below.



As you can see, the interface is divided into the following tabs:

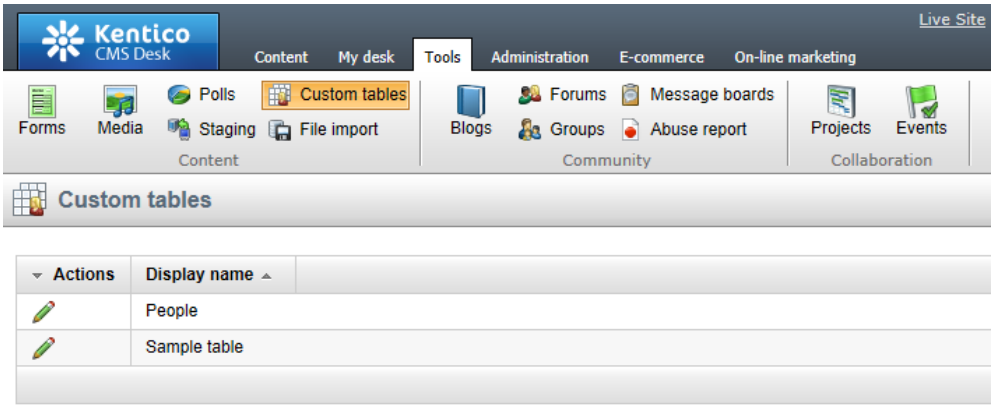
- **General** - this is where you can modify the display and code name of the custom table, as well as define URLs of custom pages that can be used instead of the default pages for adding, editing, viewing and listing of data items stored in the table
- **Fields** - on this tab, you can find the field 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

More information on the particular tabs can be found in **Kentico CMS Context Help**, which is accessible by clicking the ? icon in the top right corner of each tab.

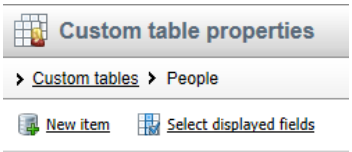
8.15.4 Managing data in custom tables

The interface for management of data items stored in custom tables is located in **CMS Desk -> Tools -> Custom tables**. This is where content editors can edit the data. The same interface is also available to administrators in **Site Manager -> Development -> Custom tables -> edit ( ) -> Data**, as described at the bottom of this page.

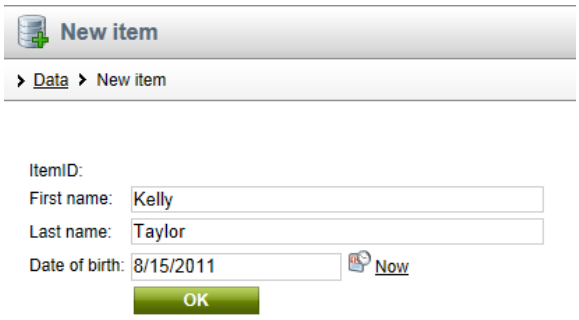


In the following example, you will learn how to add data into the sample **People** custom table we have created in the Creating custom tables chapter. The procedure is the same for any other custom table (except for the fields, which are always specific for a particular table):

1. Click the **Edit** ( ) icon next to the **People** table. A blank page with a header as in the following screenshot will be displayed. The page is blank because there is no data in the custom table yet.



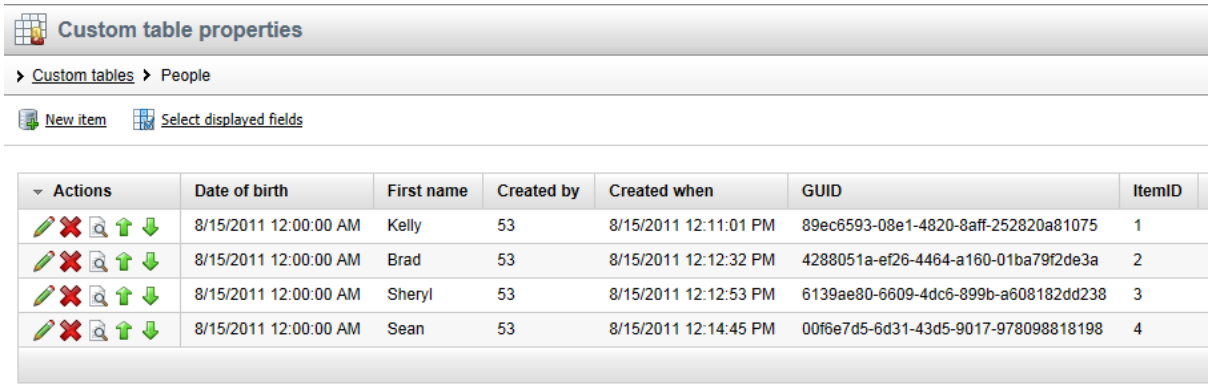
2. Click the **New item** link. The form that you can see in the following screenshot will be displayed. Enter some testing data and click **OK**.



3. The data you entered has been saved into the custom table. Now you can either edit it or use the **Create another** link to create another data record. Try creating at least two other records.

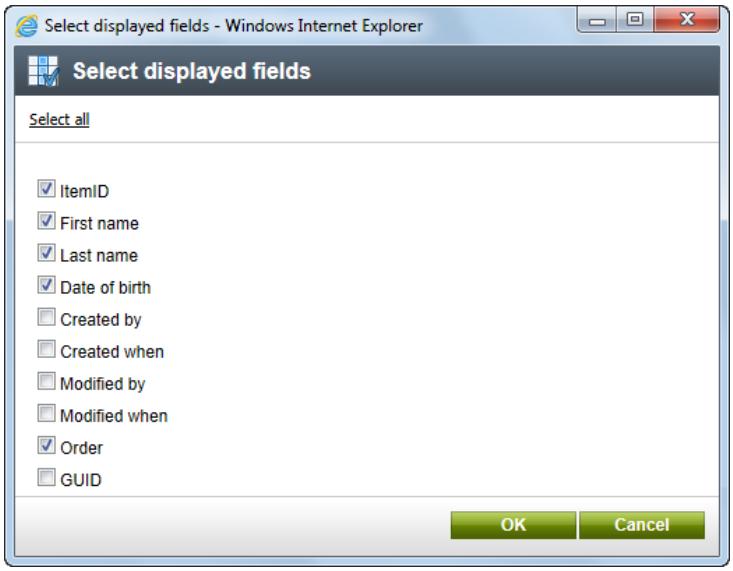
4. When you're finished, switch back to the **Custom tables** listing in **CMS Desk -> Tools** and choose to **Edit** () the **People** custom table. You should see a list of entries where the blank page was in step 1, before entering the records.

The data records can be **Edited** () or **Deleted** (). You can also re-order records in the listing using the **Up** () and **Down** (). The order is be stored in the **ItemOrder** property of each data record.

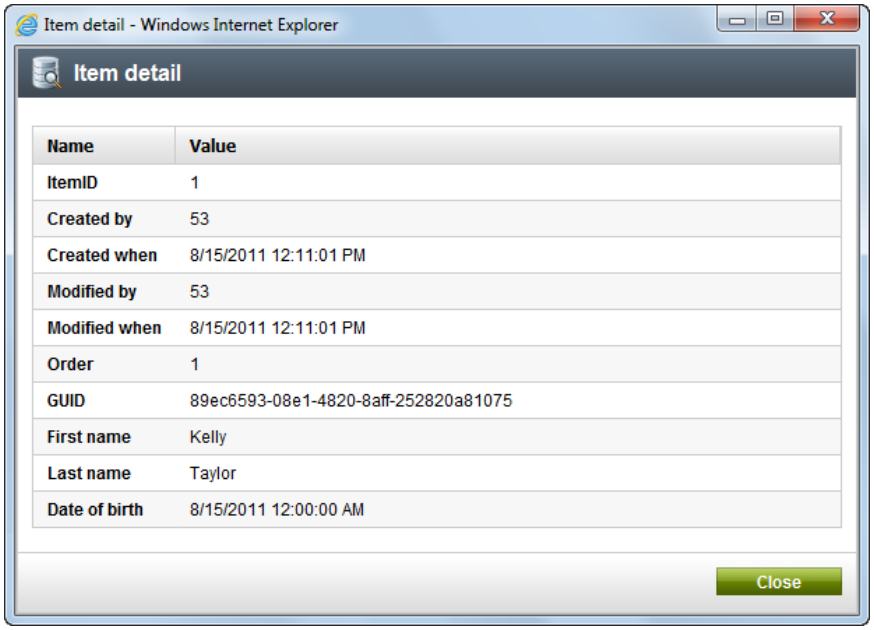


You can also create a filter for filtering displayed records in cases where large numbers of records are displayed (as can be seen in the screenshot above). To learn more about this possibility, please refer to Modules -> Alternative forms -> Creating filter forms.

5. You can choose which fields should be displayed in the listing by clicking the **Select displayed fields** link. The window depicted in the following screenshot will be displayed. Using the check-boxes, you can determine which fields are to be displayed in the listing. When you make the selection and click **OK**, you should see the result immediately.

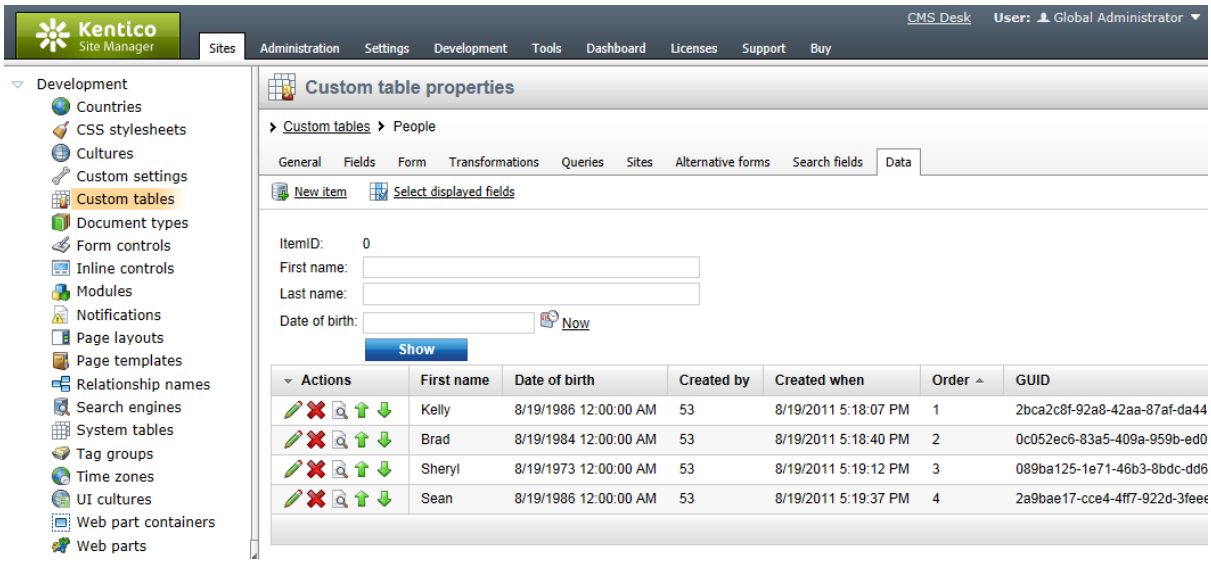


6. If you click the **View** ( ) icon next to any of the data records, detailed information about the record will be displayed in a pop-up window. All details will be displayed, no matter which fields are selected to be displayed in the listing.



Custom tables data management in Site Manager

The same user interface as described above is also available in **Site Manager -> Development -> Custom tables**. It is located on the **Data** tab of each custom table's interface displayed after clicking the **Edit** ( ) icon in the custom tables list. The functionality provided on this tab is identical to what is described above, providing administrators with access to **Site Manager** with custom tables data management possibilities without the need to switch to **CMS Desk**.



8.15.5 Available web parts

The Custom tables module comes with three web parts that can be used 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**.

Custom table datagrid

This web part displays a grid with selected columns of data records from a custom table. Design of the web part can be modified 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 screenshot below, you can see the basic appearance of the Custom table datalist web part. The appearance can be changed 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 screenshot below, you can see the basic appearance of the Custom table repeater web part. The appearance can be changed 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

8.15.6 Transformations for custom tables

Use of transformations is essential for web parts of the custom tables module. In the following example, you will learn how to modify the way data is displayed by modifying default transformation code.

We will use the **Examples -> Webparts -> Custom tables -> Custom table repeater** example on the sample Corporate Site. We will also use the **People** custom table that we have created in the Creating custom tables topic and populated with some data items in the Managing data in custom tables topic.

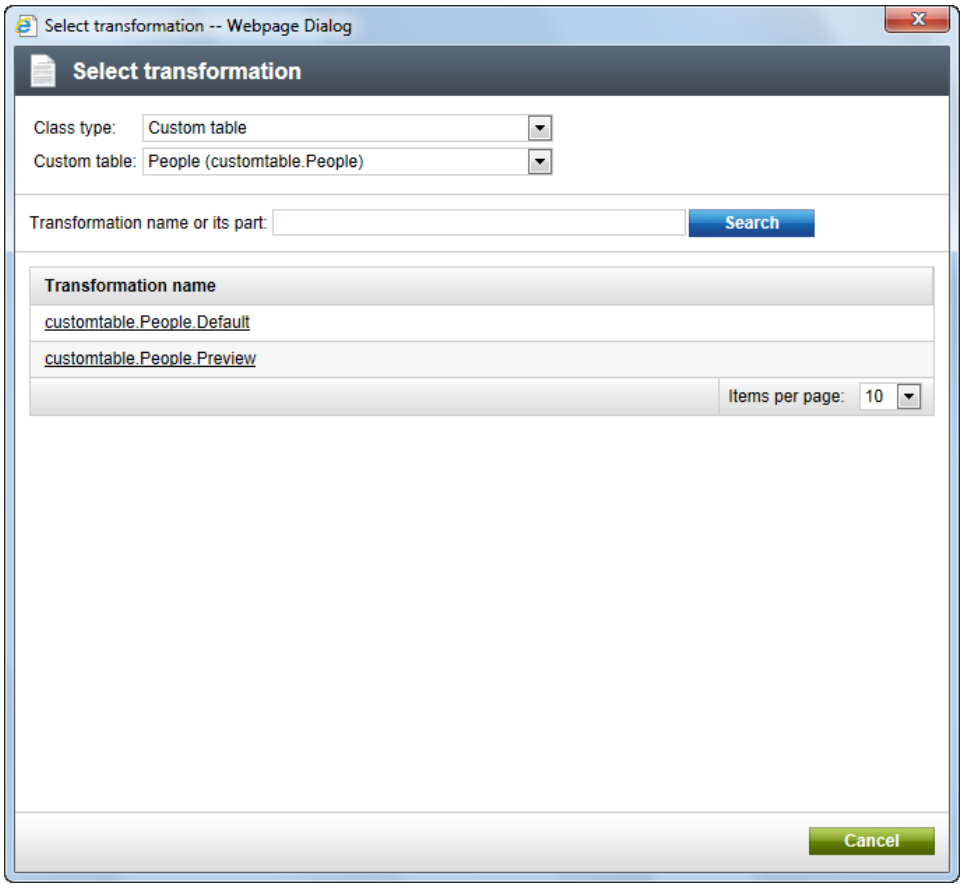
1. Go to **CMS Desk**, switch to **Edit** mode and select **Examples -> Webparts -> Custom tables -> Custom table repeater**. By default, the repeater is bound to the **Sample table**, which is a sample custom table contained in the Corporate site. It should look as in the following screenshot:

Examples > Webparts > Custom tables > Custom table repeater

Item text: Sample text 1
Item text: Sample text 2
Item text: Sample text 3

2. Switch to the **Design** tab and choose to **Configure** ( ) the repeater's properties. First, use the **Custom table** drop-down list to select the custom table from which the data will be displayed - it should be the **People** table in our case.

3. Now it's time to choose a transformation via the **Transformation name** property. If you click the **Select** button next to the mentioned field, you will get the pop-up window as depicted in the following screenshot. There are two default transformations generated by default for each custom table - **Default** and **Preview**. Choose the **Default** transformation and click **OK**. Click **OK** again to close the **Web part properties** window.



4. Now if you switch to the **Live site** mode, you should see that the repeater is displaying something like the following:

Examples > Webparts > Custom tables > Custom table repeater

First name: Kelly
Last name: Taylor
Date of birth: 2/23/1983 12:00:00 AM
ItemCreatedBy: 53
ItemCreatedWhen: 10/28/2008 4:47:02 PM
ItemModifiedBy:
ItemModifiedWhen:
ItemOrder: 2
First name: Jack
Last name: McDonald
Date of birth: 3/5/1980 12:00:00 AM
ItemCreatedBy: 53
ItemCreatedWhen: 10/28/2008 4:48:21 PM

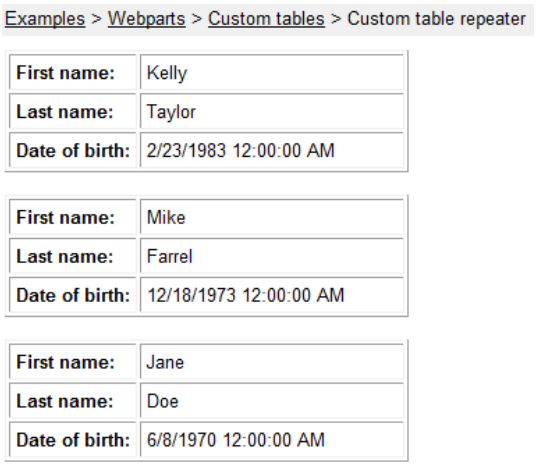
5. As you probably wouldn't want to have all the empty fields displayed and the table to be unformatted on your real website, you will need to edit the transformation code. To do it, switch to the **Edit** mode again, switch to the **Design** tab, choose to **Configure** ( ) the web part's properties and click the **Edit** button next to the **Transformation name** property. Now you can edit the transformation code according to your specific needs.

For the purpose of this tutorial, please replace the transformation code with the following code example.

You can notice that it is just a modification of the former **Default** transformation, with the unwanted fields deleted and the 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/>
```

6. Click **Save**. Now if you go to the live site, you should see that the data is displayed in a formatted way, as depicted below.

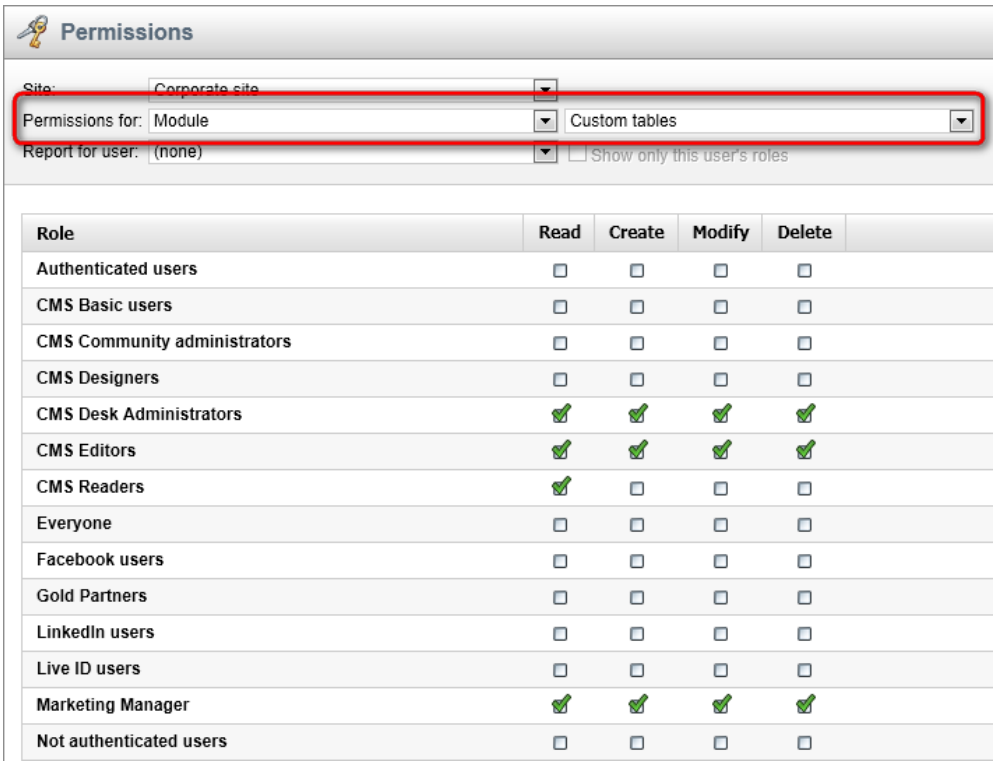


8.15.7 Security

Permissions of the Custom tables module can be configured on two levels - globally for all custom tables belonging to the current site and separately for each particular custom table. The first, global level is configuration via the **Modules -> Custom tables** permission matrix in **Site Manager -> Administration -> Permissions**.

The following permissions can be granted to particular roles on this level:

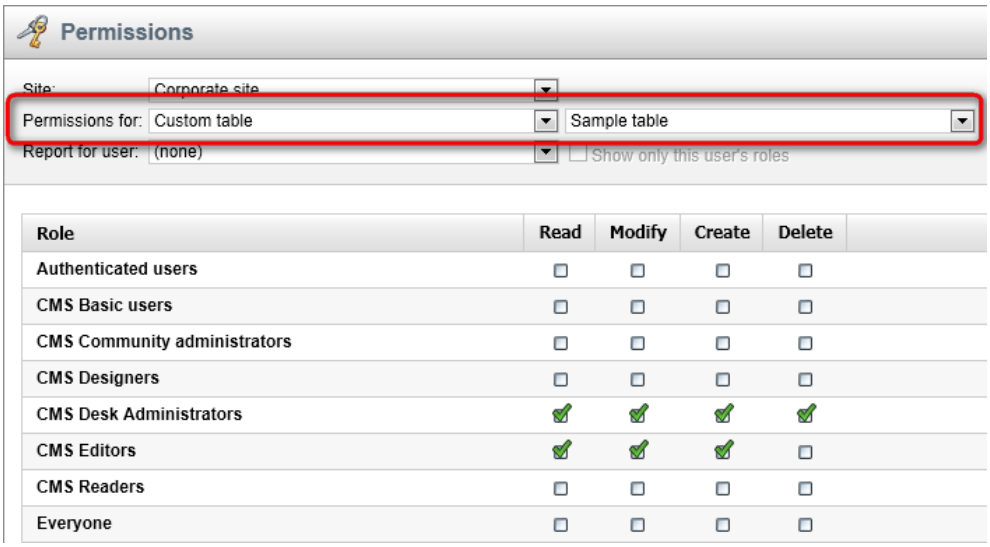
- **Create** - members of the role are allowed to create data in any custom table
- **Delete** - members of the role are allowed to delete data in any custom table
- **Modify** - members of the role are allowed to modify data in any custom table
- **Read** - members of the role are allowed to read any custom table data



You can also configure these permissions separately for particular custom tables. This can also be done in **Site Manager -> Administration -> Permissions**, but in this case, you have to select **Custom tables** from the **Permission type** drop-down list. The **Permission matrix** drop-down list will then offer you all custom tables assigned to the selected site.

The permissions have the same names and meanings as the global ones, but this time, they are related only to the one selected custom table:

- **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



8.15.8 Custom tables internals and API

8.15.8.1 Overview

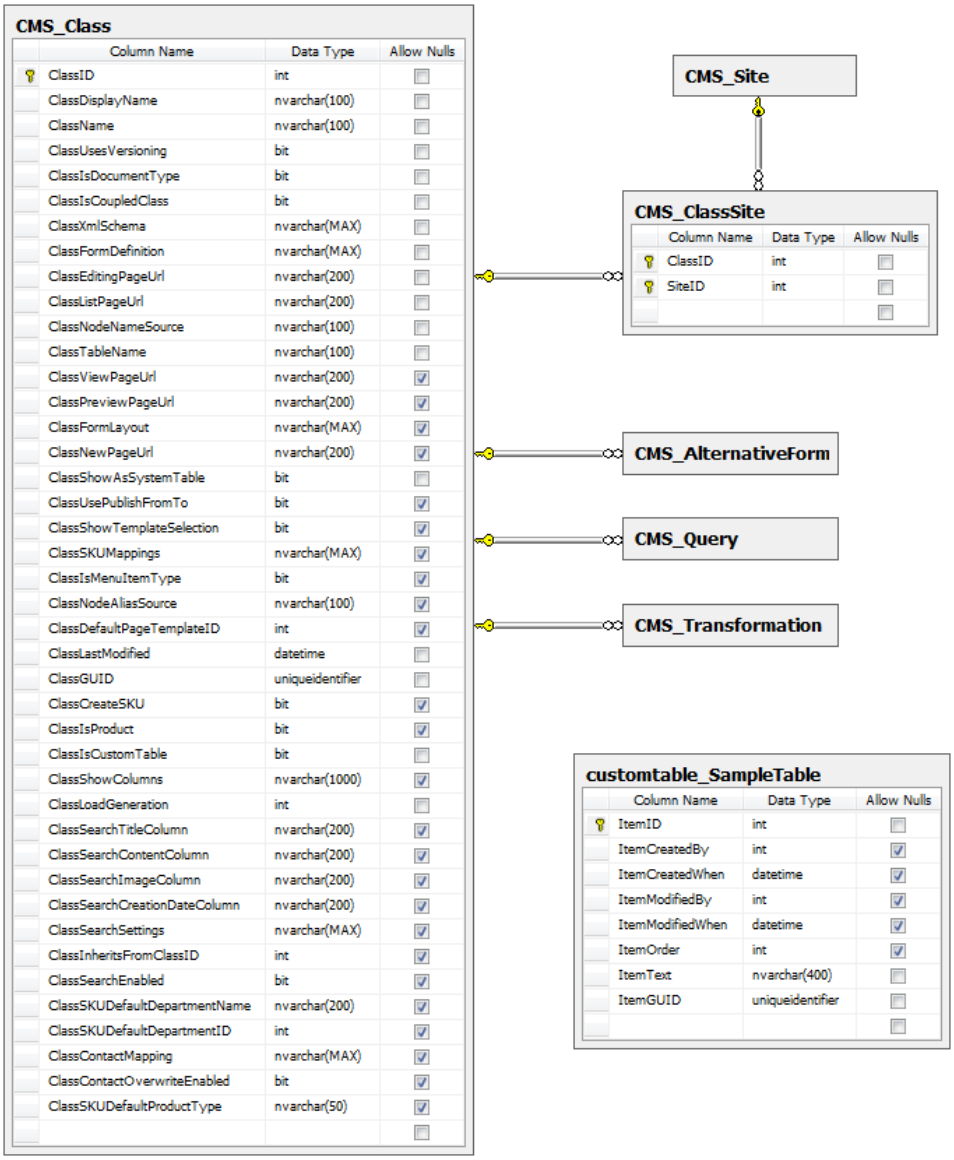
In this chapter, you will learn which database tables and API classes are used in the Custom tables module. You will also see the most common API examples.

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.

8.15.8.2 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.



8.15.8.3 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 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.

8.15.8.4 API examples

8.15.8.4.1 Overview

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;
```

8.15.8.4.2 Managing custom table data

The following example creates a custom table item.

```
private bool CreateCustomTableItem()
{
    // Create new Custom table item provider
    CustomTableItemProvider customTableProvider = new CustomTableItemProvider(
CMSContext.CurrentUser);

    // Prepare the parameters
    string customTableClassName = "customtable.sampletable";

    // Check if Custom table 'Sample table' exists
    DataClassInfo customTable = DataClassInfoProvider.GetDataClass
(customTableClassName);
    if (customTable != null)
    {
```

```
        // Create new custom table item
        CustomTableItem newCustomTableItem = new CustomTableItem
(customTableClassName, customTableProvider);

        // Set the ItemText field value
        newCustomTableItem.SetValue("ItemText", "New text");

        // Insert the custom table item into database
        newCustomTableItem.Insert();

        return true;
    }

    return false;
}
```

The following example gets and updates the custom table item created by the previous code example.

```
private bool GetAndUpdateCustomTableItem()
{
    // Create new Custom table item provider
    CustomTableItemProvider customTableProvider = new CustomTableItemProvider(
CMSContext.CurrentUser);

    string customTableClassName = "customtable.sampletable";

    // Check if Custom table 'Sample table' exists
    DataClassInfo customTable = DataClassInfoProvider.GetDataClass
(customTableClassName);
    if (customTable != null)
    {
        // Prepare the parameters
        string where = "ItemText LIKE N'New text'";
        int topN = 1;
        string columns = "ItemID";

        // Get the data set according to the parameters
        DataSet dataSet = customTableProvider.GetItems(customTableClassName,
where, null, topN, columns);

        if (!DataHelper.DataSourceIsEmpty(dataSet))
        {
            // Get the custom table item ID
            int itemID = ValidationHelper.GetInteger(dataSet.Tables[0].Rows[0][0],
0);

            // Get the custom table item
            CustomTableItem updateCustomTableItem = customTableProvider.GetItem
(itemID, customTableClassName);

            if (updateCustomTableItem != null)
            {
```

```
                string itemText = ValidationHelper.GetString
(updateCustomTableItem.GetValue("ItemText"), "");

                // Set new values
                updateCustomTableItem.SetValue("ItemText", itemText.ToLower());

                // Save the changes
                updateCustomTableItem.Update();

                return true;
            }
        }
    }

    return false;
}
```

The following example gets a custom table item and moves it down in the order of items in the table.

```
private bool GetAndMoveCustomTableItemDown()
{
    // Create new Custom table item provider
    CustomTableItemProvider customTableProvider = new CustomTableItemProvider(
CMSContext.CurrentUser);

    string customTableClassName = "customtable.sampletable";

    // Check if Custom table 'Sample table' exists
    DataClassInfo customTable = DataClassInfoProvider.GetDataClass
(customTableClassName);
    if (customTable != null)
    {
        // Prepare the parameters
        string where = "ItemText LIKE N'New text'";
        int topN = 1;
        string columns = "ItemID";

        // Get the data set according to the parameters
        DataSet dataSet = customTableProvider.GetItems(customTableClassName,
where, null, topN, columns);

        if (!DataHelper.DataSourceIsEmpty(dataSet))
        {
            // Get the custom table item ID
            int itemID = ValidationHelper.GetInteger(dataSet.Tables[0].Rows[0][0],
0);

            // Move the item down
            customTableProvider.MoveItemDown(itemID, customTableClassName);

            return true;
        }
```

```
    }

    return false;
}
```

The following example gets a custom table item and moves it down in the order of items in the custom table.

```
private bool GetAndMoveCustomTableItemUp()
{
    // Create new Custom table item provider
    CustomTableItemProvider customTableProvider = new CustomTableItemProvider(
CMSContext.CurrentUser);

    string customTableClassName = "customtable.sampletable";

    // Check if Custom table 'Sample table' exists
    DataClassInfo customTable = DataClassInfoProvider.GetDataClass
(customTableClassName);
    if (customTable != null)
    {
        // Prepare the parameters
        string where = "ItemText LIKE N'New text'";
        int topN = 1;
        string columns = "ItemID";

        // Get the data set according to the parameters
        DataSet dataSet = customTableProvider.GetItems(customTableClassName,
where, null, topN, columns);

        if (!DataHelper.DataSourceIsEmpty(dataSet))
        {
            // Get the custom table item ID
            int itemID = ValidationHelper.GetInteger(dataSet.Tables[0].Rows[0][0],
0);

            // Move the item up
            customTableProvider.MoveItemUp(itemID, customTableClassName);

            return true;
        }
    }

    return false;
}
```

The following example gets multiple custom table items based on a WHERE condition and updates values in their *Item*

```
private bool GetAndBulkUpdateCustomTableItems()
{
```

```
    // Create new Custom table item provider
    CustomTableItemProvider customTableProvider = new CustomTableItemProvider(
CMSContext.CurrentUser);

    string customTableClassName = "customtable.sampletable";

    // Check if Custom table 'Sample table' exists
    DataClassInfo customTable = DataClassInfoProvider.GetDataClass
(customTableClassName);
    if (customTable != null)
    {
        // Prepare the parameters

        string where = "ItemText LIKE N'New text%'";

        // Get the data
        DataSet customTableItems = customTableProvider.GetItems
(customTableClassName, where, null);
        if (!DataHelper.DataSourceIsEmpty(customTableItems))
        {
            // Loop through the individual items
            foreach (DataRow customTableItemDr in customTableItems.Tables[0].Rows)
            {
                // Create object from DataRow
                CustomTableItem modifyCustomTableItem = new CustomTableItem
(customTableItemDr, customTableClassName);

                string itemText = ValidationHelper.GetString
(modifyCustomTableItem.GetValue("ItemText"), "");

                // Set new values
                modifyCustomTableItem.SetValue("ItemText", itemText.ToUpper());

                // Save the changes
                modifyCustomTableItem.Update();
            }

            return true;
        }
    }

    return false;
}
```

The following example deletes the custom table item created by the first code example on this page.

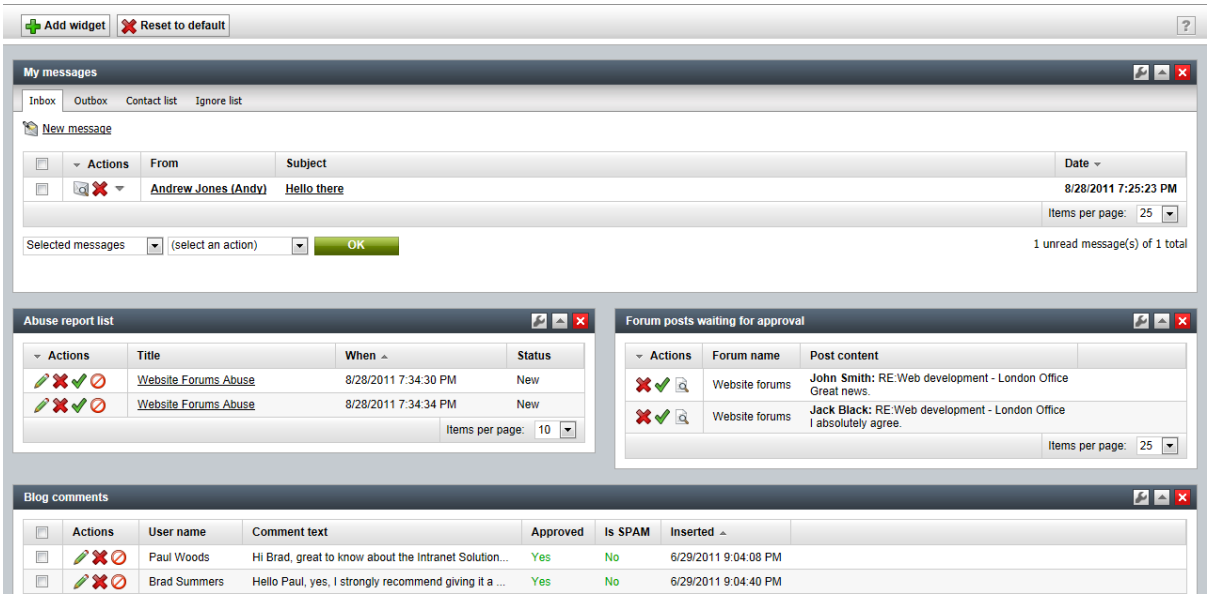
```
private bool DeleteCustomTableItem()
{
    // Create new Custom table item provider
    CustomTableItemProvider customTableProvider = new CustomTableItemProvider(
CMSContext.CurrentUser);
```

```
    string customTableClassName = "customtable.sampletable";

    // Check if Custom table 'Sample table' exists
    DataClassInfo customTable = DataClassInfoProvider.GetDataClass
(customTableClassName);
    if (customTable != null)
    {
        // Prepare the parameters
        string where = "ItemText LIKE N'New text%'";

        // Get the data
        DataSet customTableItems = customTableProvider.GetItems
(customTableClassName, where, null);
        if (!DataHelper.DataSourceIsEmpty(customTableItems))
        {
            // Loop through the individual items
            foreach (DataRow customTableItemDr in customTableItems.Tables[0].Rows)
            {
                // Create object from DataRow
                CustomTableItem deleteCustomTableItem = new CustomTableItem
(customTableItemDr, customTableClassName);

                // Delete custom table item from database
                deleteCustomTableItem.Delete();
            }

            return true;
        }
    }

    return false;
}
```

8.16 Dashboards

8.16.1 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**

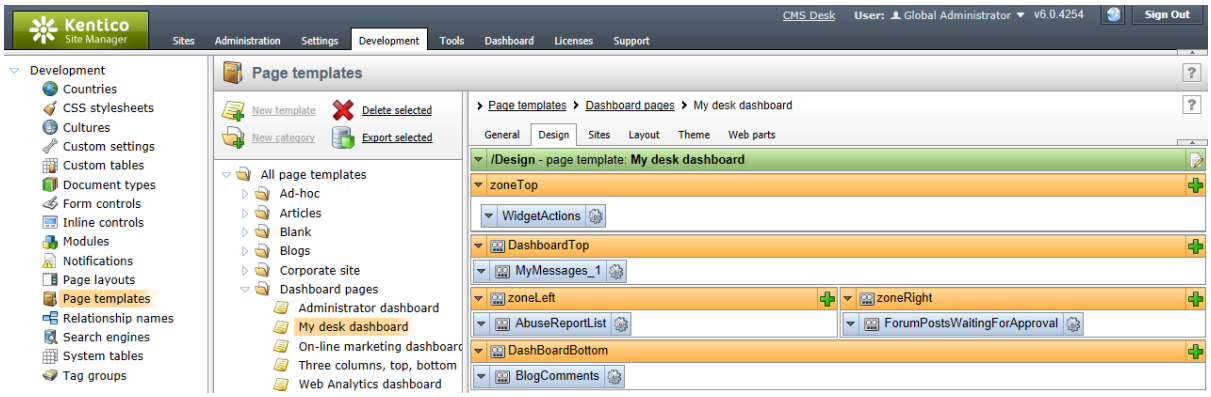
If required, a dashboard page may be created at a custom location according to the procedure described in the Adding a new dashboard to the interface topic.

8.16.2 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*.

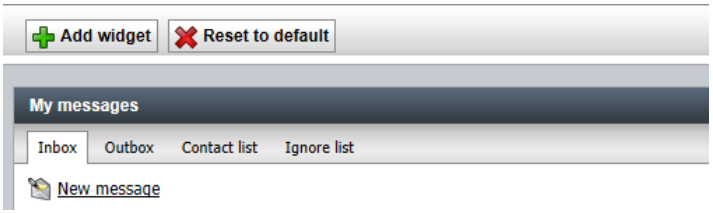
Every dashboard template should contain the **Widget actions** web part somewhere in its layout in order to allow users to add new widgets or reset the zones of the dashboard to their default widget content. The web part must also be configured ( ) to have its **Widget zone type** property set to *Dashboard*. 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** link 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 the **Reset to default** link.



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.

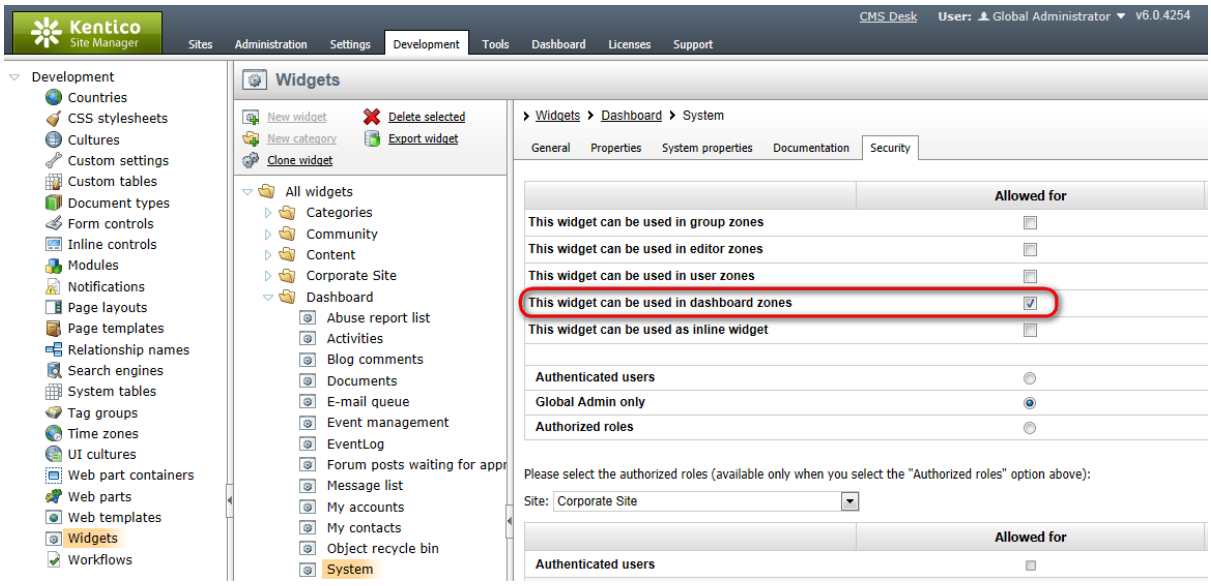
Clicking the **Configure widget** ( ) button opens the **Widget properties** dialog where the widget's properties can be modified. Every widget has its own set of properties depending on its function (detailed information can be found in the Widgets reference guide). The **Widget title** property is 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.

8.16.3 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.

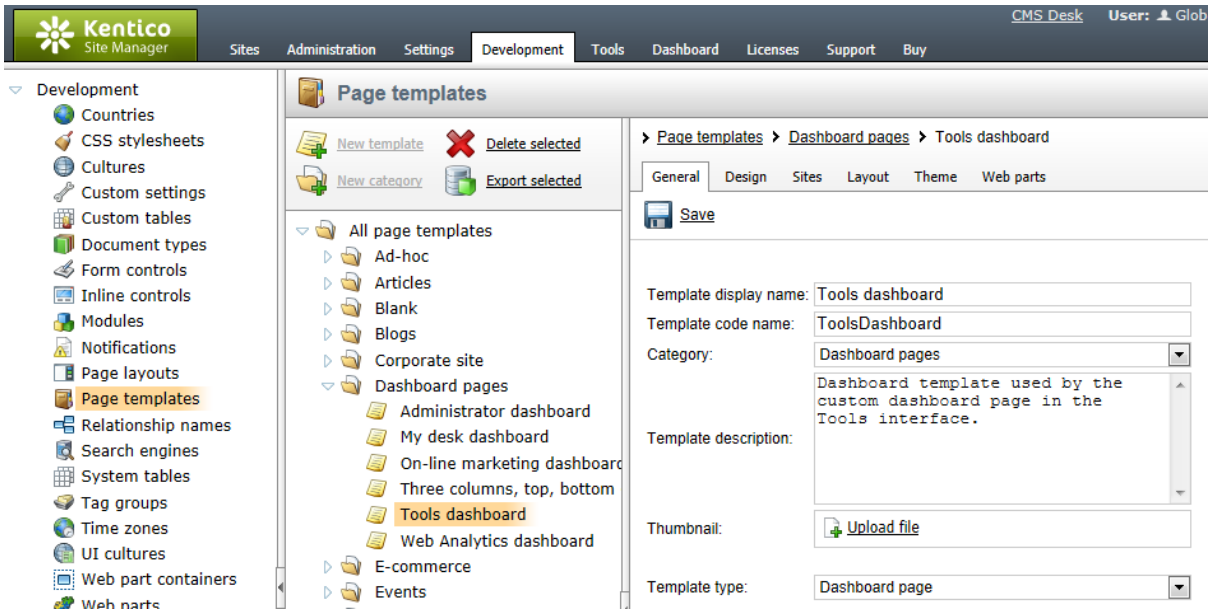
The following example demonstrates how a custom dashboard can be added to the **Tools** tab of the **CMS Desk** administration interface. The same principles can be applied when creating a dashboard in other locations.

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 the **New template** link. 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.

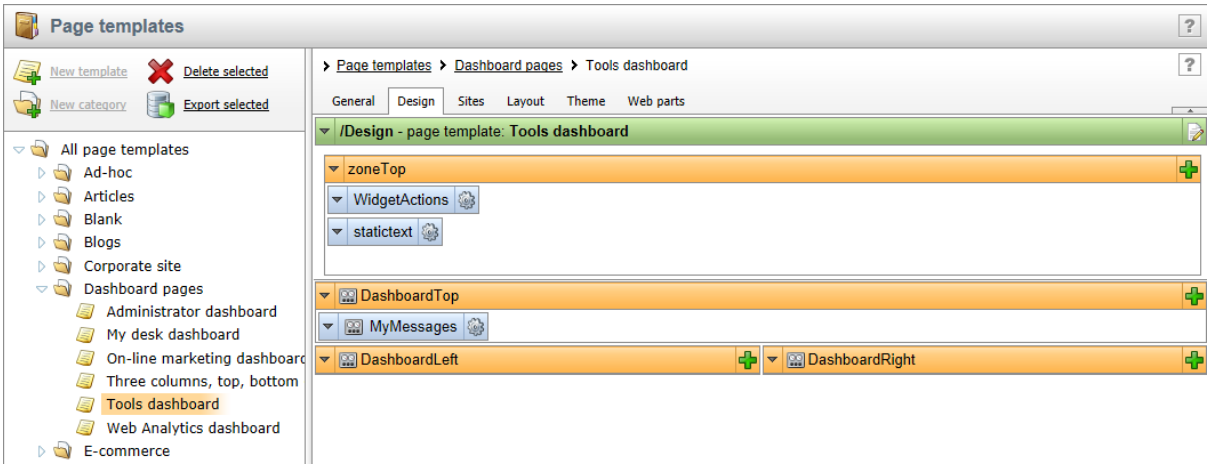
Leave the remaining properties in their default state and click **OK**.

5. Add another web part to the same zone. This time select **Text -> 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 and 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)

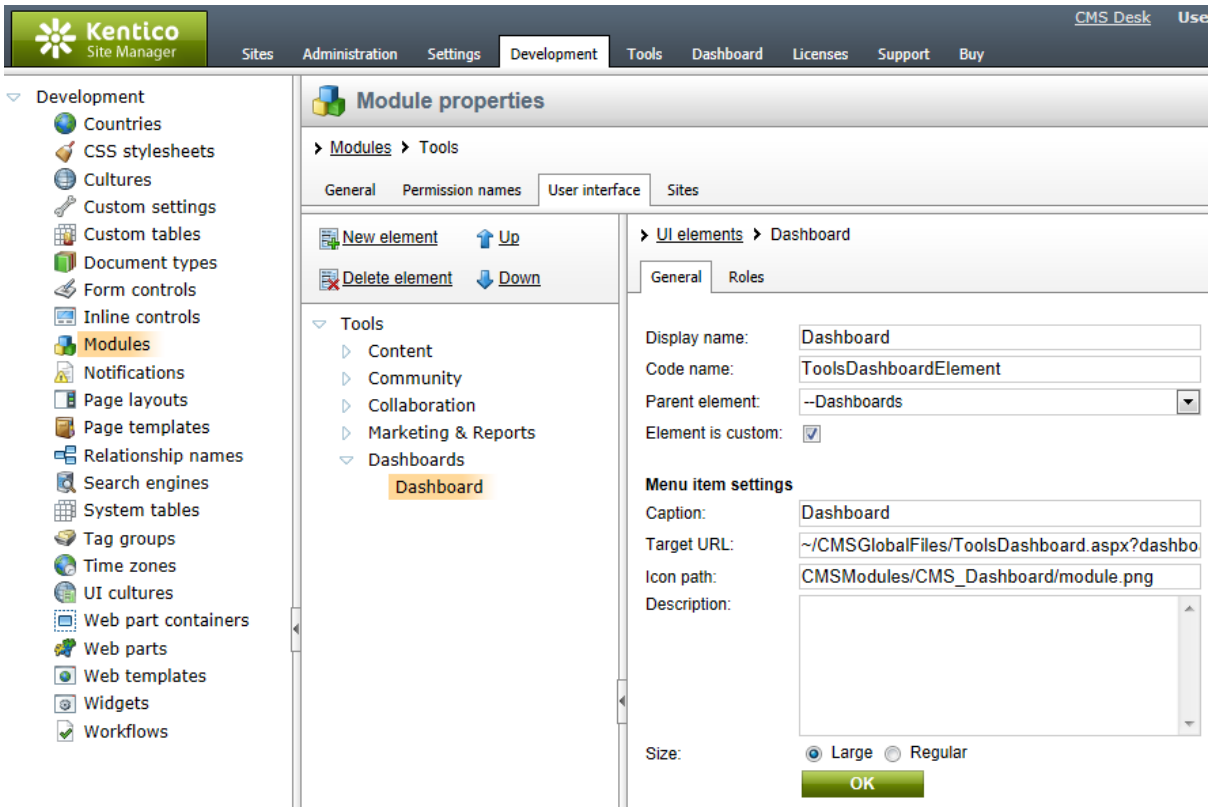
- **Caption**: Dashboards

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 using the appropriate macro expression in 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.

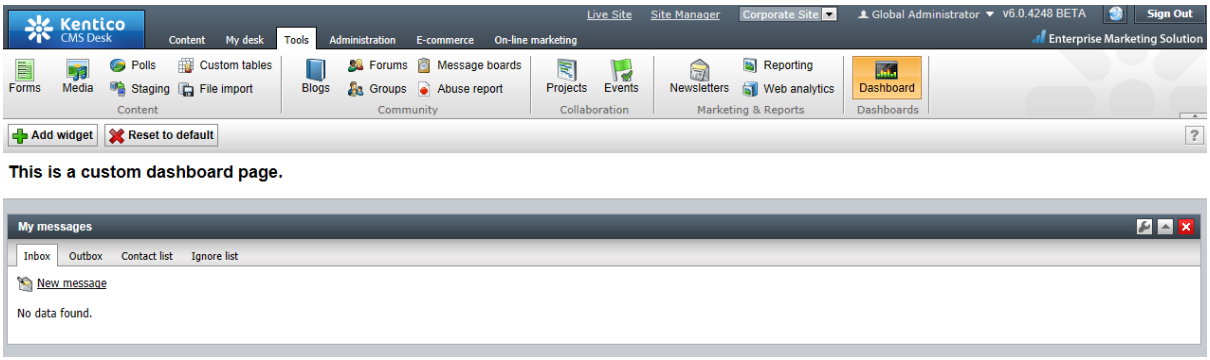
Creating dashboards in the Site Manager interface

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.



8.17 Document library

8.17.1 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.

8.17.2 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*, 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 (as can be seen in the screenshot below) and the **Translate** option will be 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

8.17.3 Document library widget

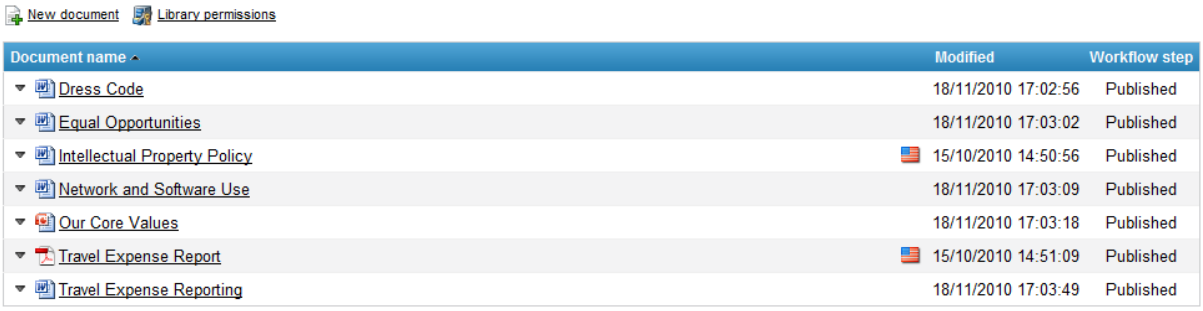
The module also comes with the **Document library** widget. The widget is stored under the **Document library** widget category and it is an editor widget, i.e. it can only be added to editor widget zones of page templates by website editors in CMS Desk.

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 (as can be seen in the screenshot below) and the **Translate** option will be 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.



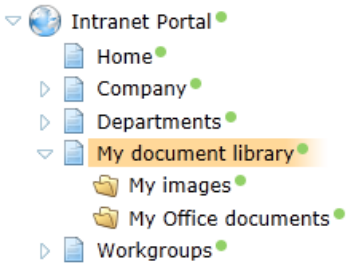
8.17.4 Creating a document library

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:

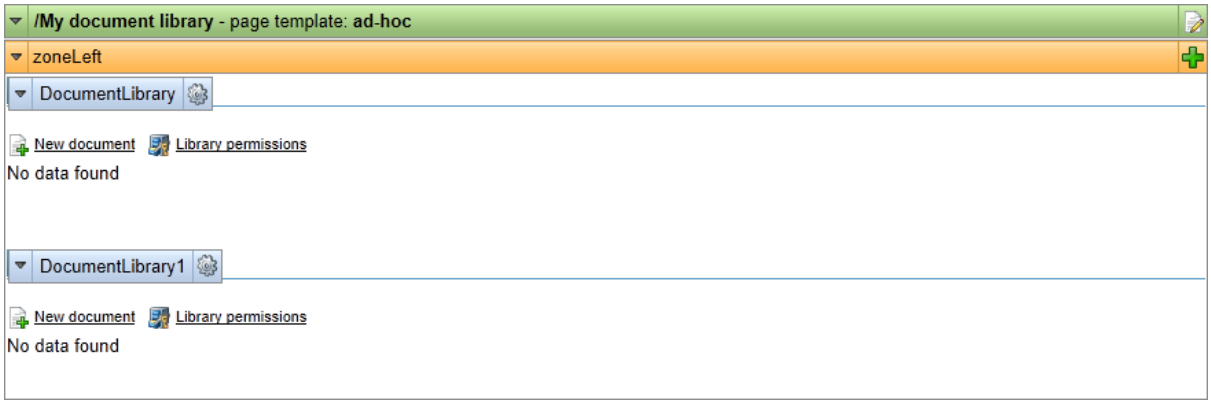
- **Library path**: /My-document-library/My-images
- **Page size**: 15
- **Document form**: leave the field blank
- **Web part container**: Intranet simple container (with header line)
- **Container title**: My images
- **Content after**:

Click **OK**.

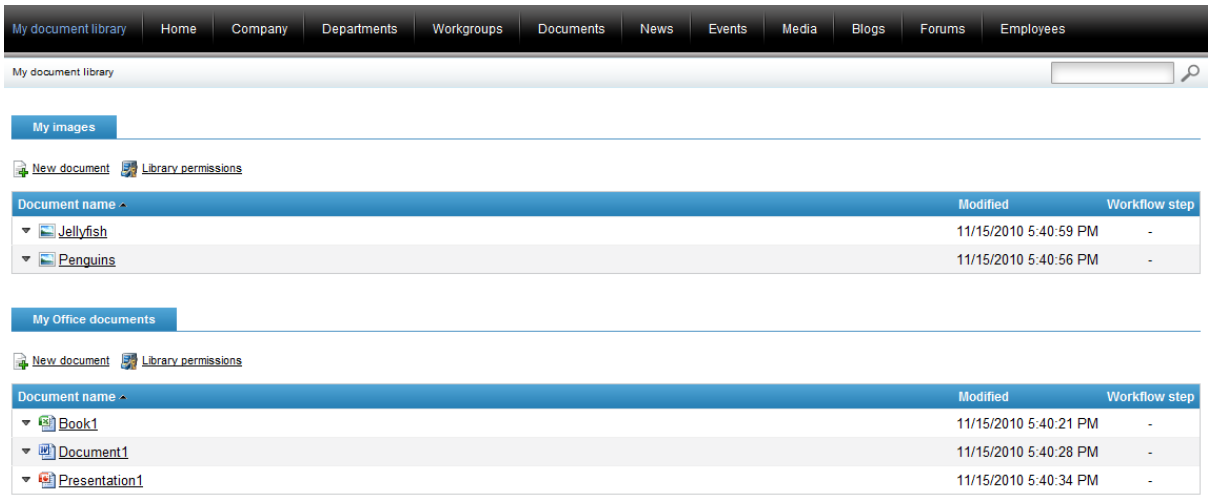
3. Repeat step 2 and add another **Document library** web part to the page. This time, use the following values of its properties:

- **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.

8.17.5 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.

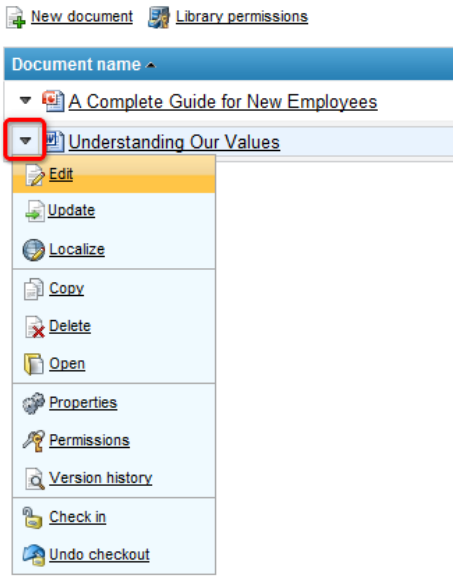
When a new document library is created, there are no files in it, so live site users can only see the web part with the following two links at the top.

- 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

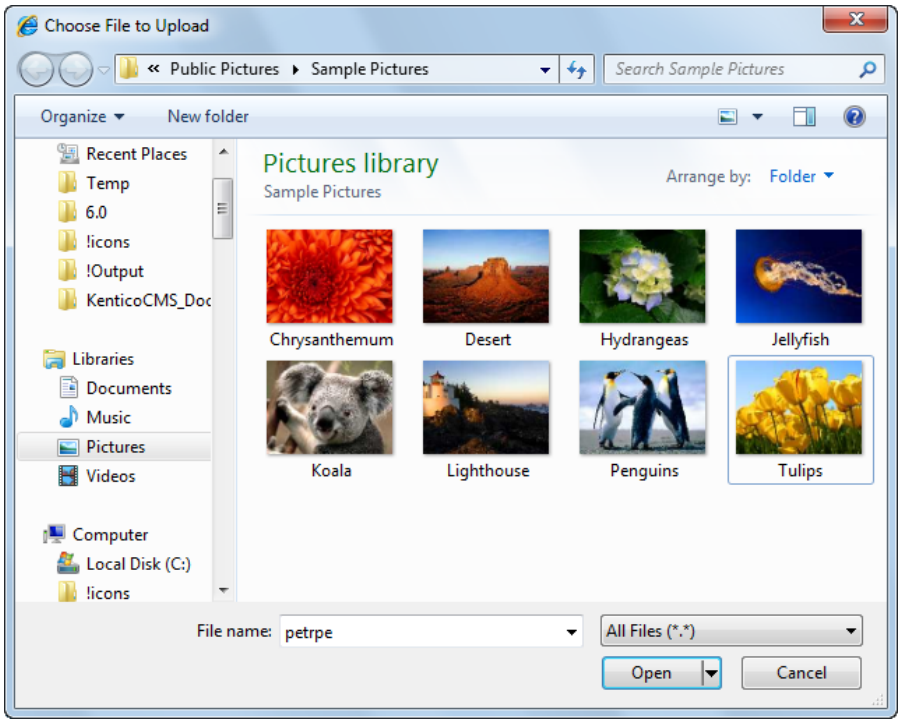
These actions can be executed from a context menu, which is accessible by right-clicking anywhere in a document's row, or by left-clicking the down-arrow to the left of a document's name (as highlighted in the screenshot below).



The following text explains particular actions that can be performed in a document library:

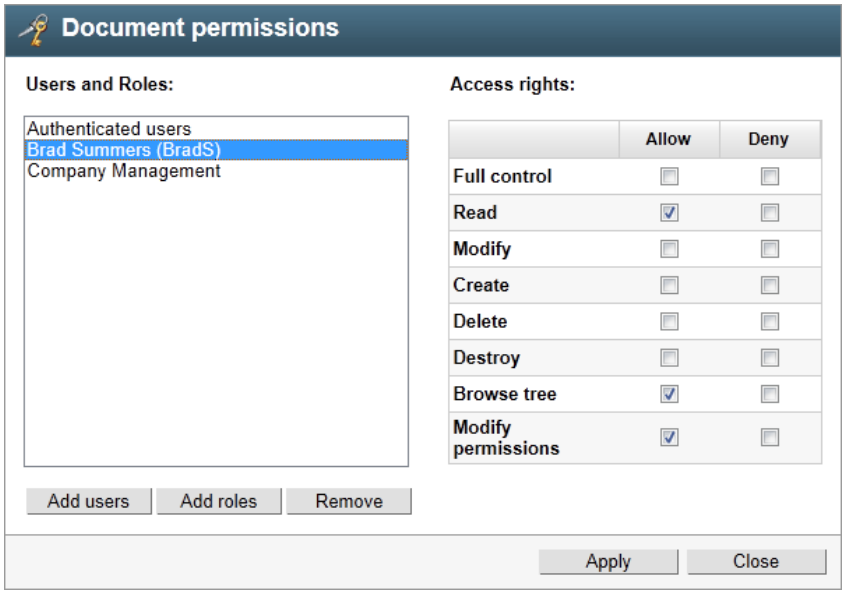
New document

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

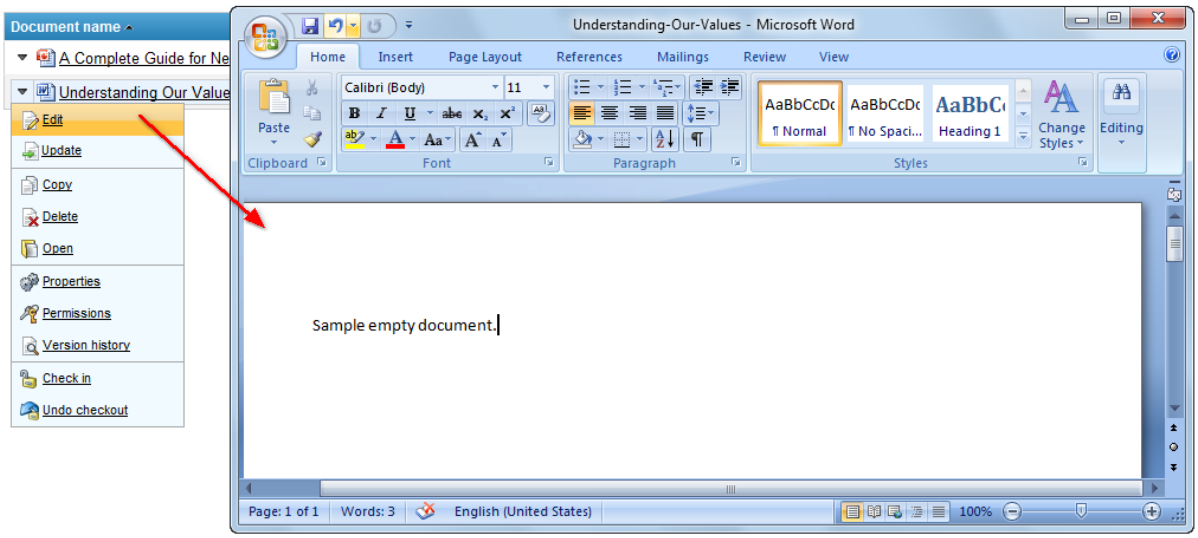
Clicking the **Library permissions** link opens a dialog for document library permissions configuration. Please refer to the **Configuring document-level permissions on the live site** section of the Security topic for more details.



Edit

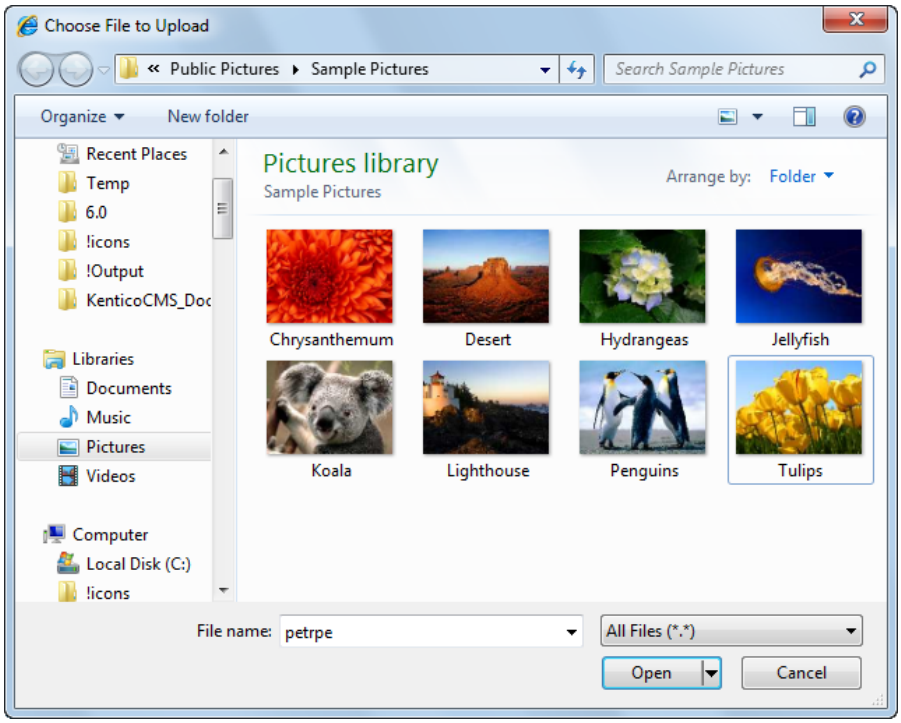
The **Edit** action is only available if WebDAV integration is enabled. It opens the file in a client

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.



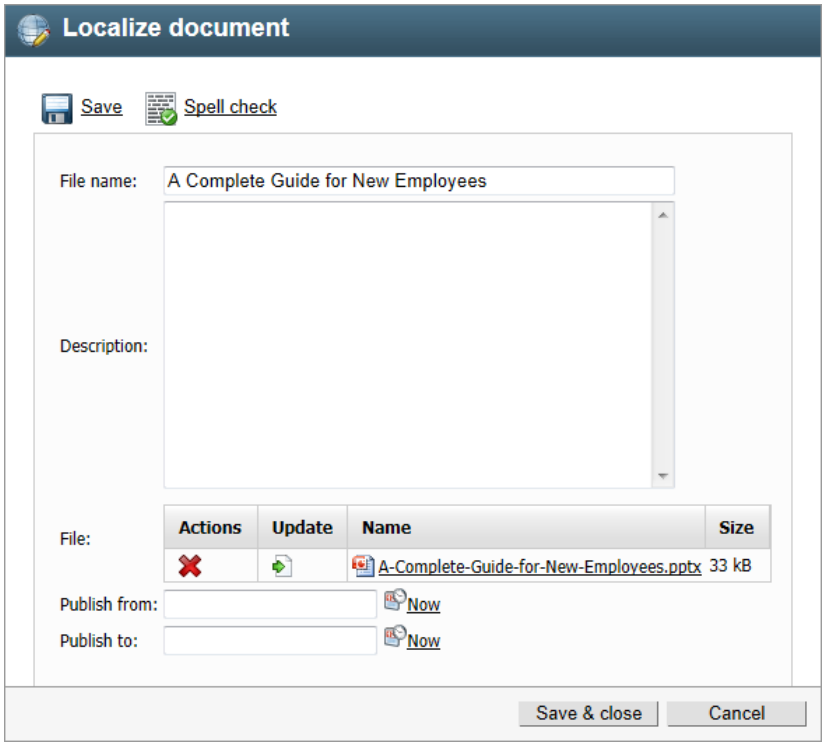
Update

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.



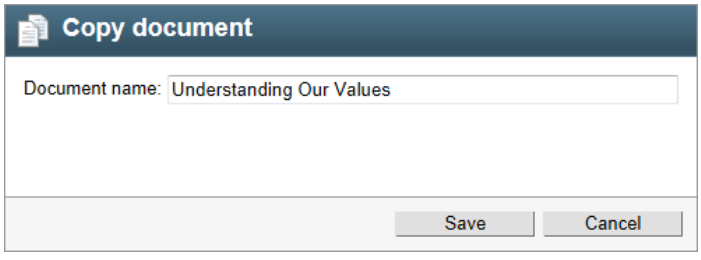
Localize

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

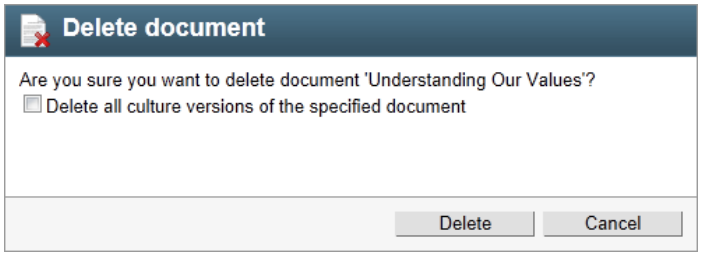
The **Copy** action creates a copy of the file in the same document library. As only flat structure of document libraries is currently possible, a different file name needs to be entered for the copy, while the " - Copy" suffix is offered to be added to the file name automatically. When copying a linked document (marked with the flag), the copy will be a standard document, not a linked one.



Delete

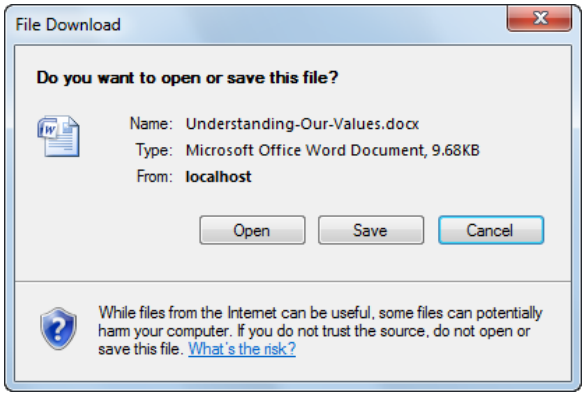
Clicking the **Delete** action opens the dialog depicted below. Using the dialog, you can remove the document from the document library. On multilingual websites where more that one language version of

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.



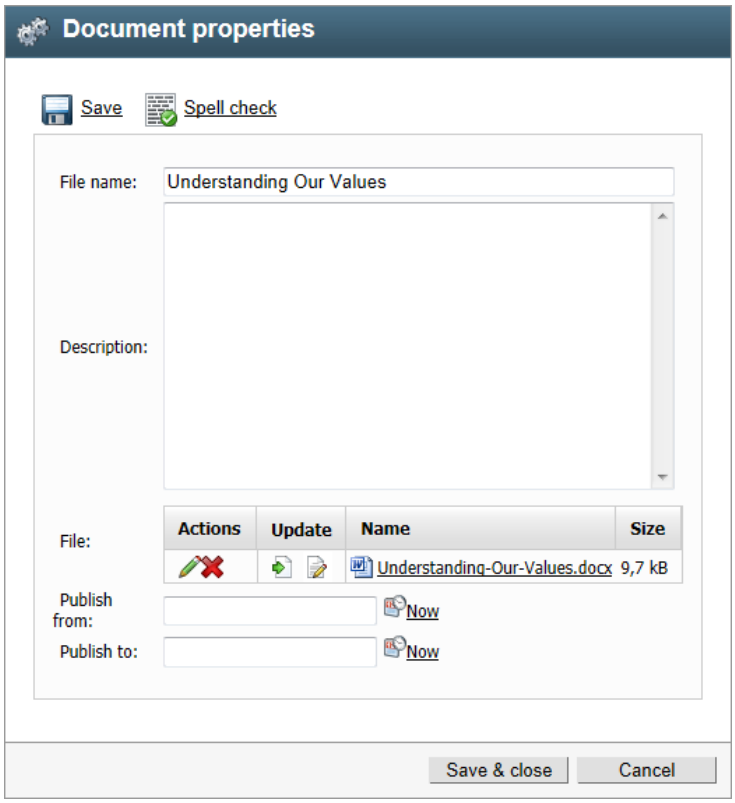
Open

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.

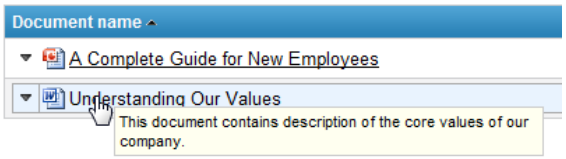


Properties

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.

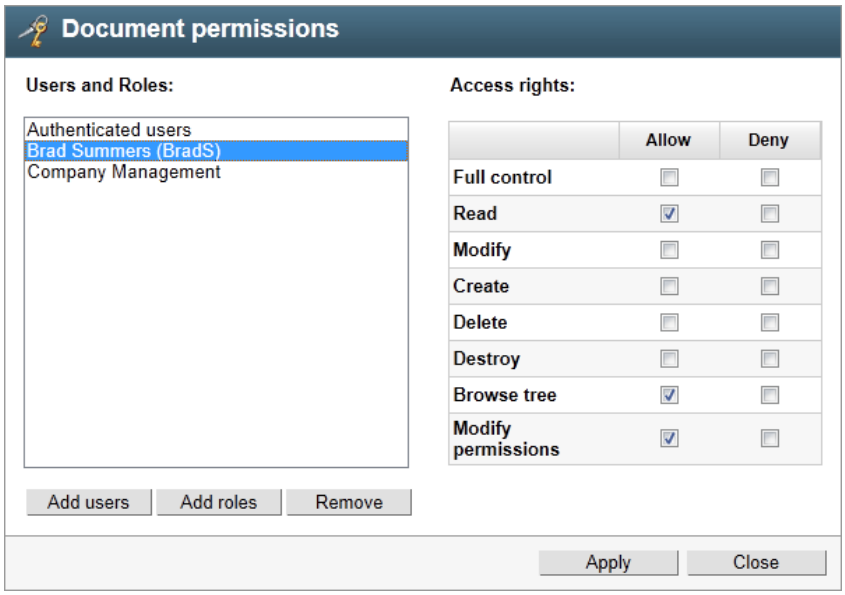


The text filled in the **Description** field is displayed in a tooltip when the mouse pointer is placed above the document.



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

Clicking the **Destroy history** button deletes all versions of the document except the current one.

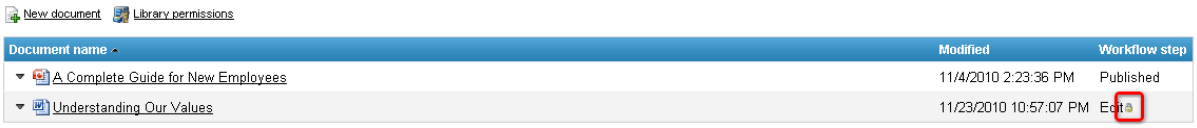
Workflow actions

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.



8.17.6 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]	•	•					

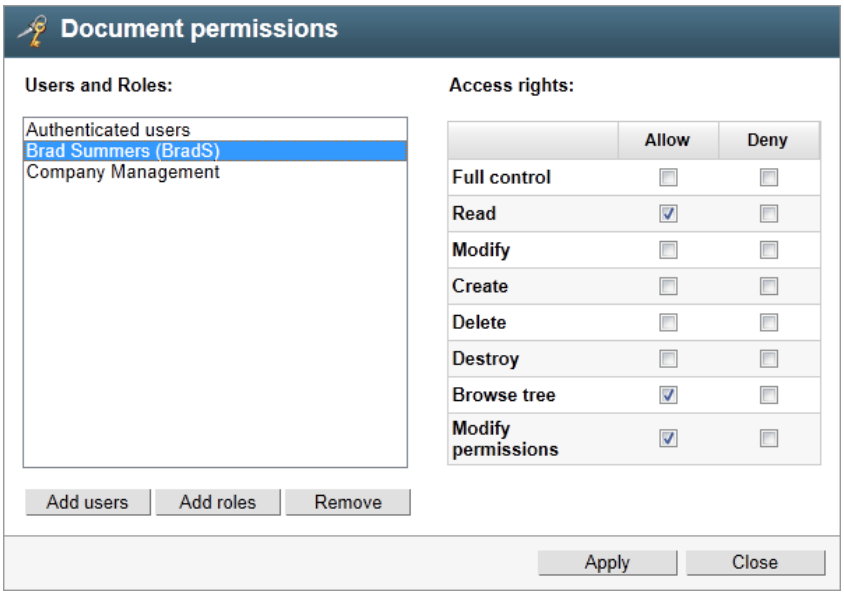
[1] The **Destroy** permission is required for the user to be able to delete particular versions or the whole version history.
[2] For these actions to be available, the user must also be in one of the roles that are allowed to approve/reject the document in the current workflow step or have the **Manage 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

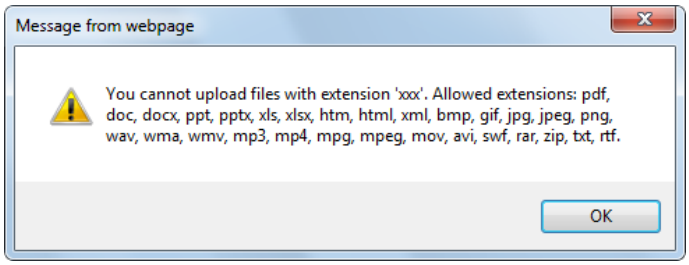
Document libraries reflect workflows applied to documents stored in them. Unless the current user has the **Modify** permission for a document, the currently published version of the document is always displayed to the user. If the document is currently archived or not published, the document is not 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.



8.18 E-commerce

8.18.1 Overview

For information about the **E-commerce** module please refer to Kentico CMS E-commerce Guide.

8.19 E-mail queue

8.19.1 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.

8.19.2 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:

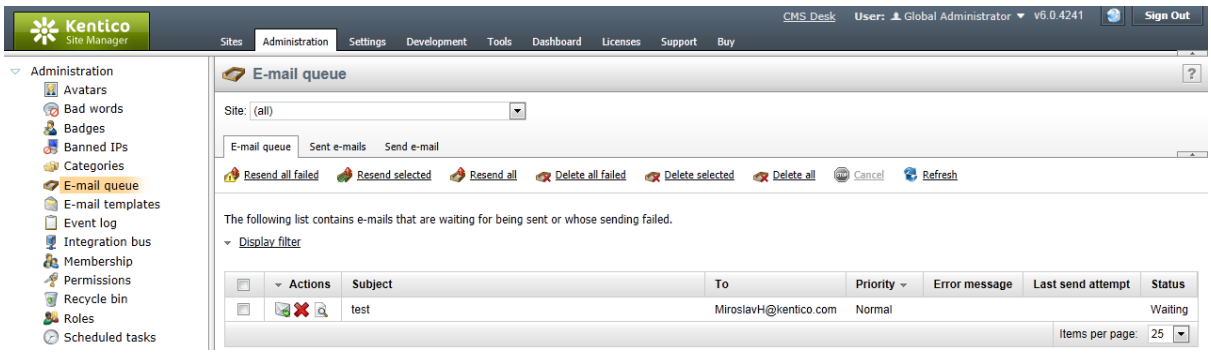
E-mail queue tab

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

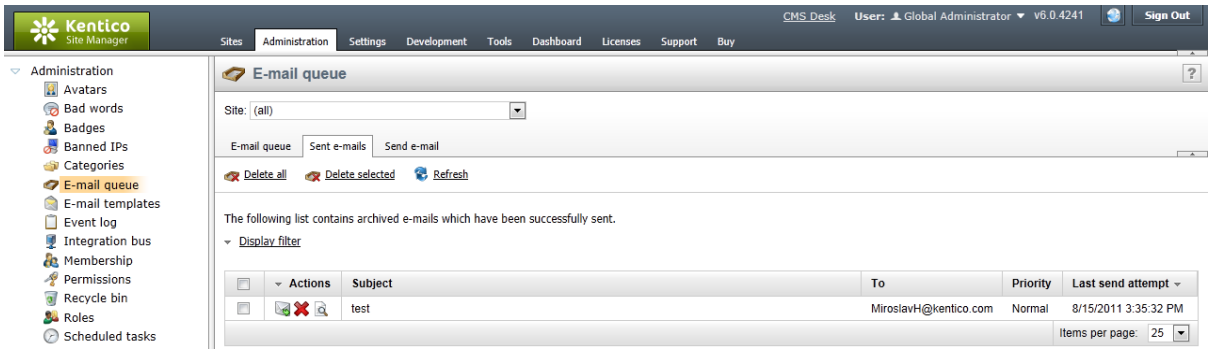


Sent e-mails tab

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.

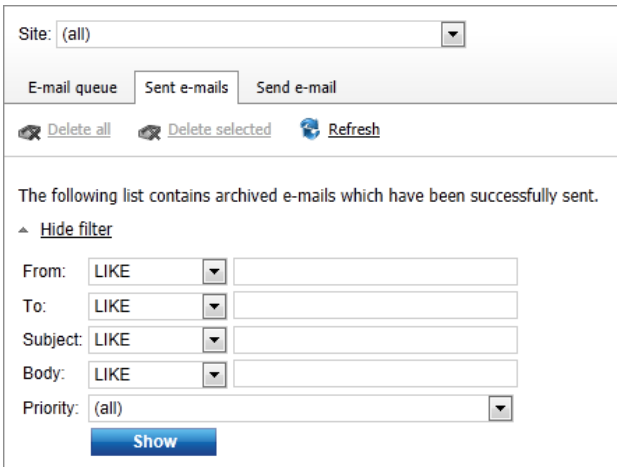
You can **Resend** ( ) the mail, **Delete** ( ) it or **View** ( ) its details. There are also three links at the top of the page:

- **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.

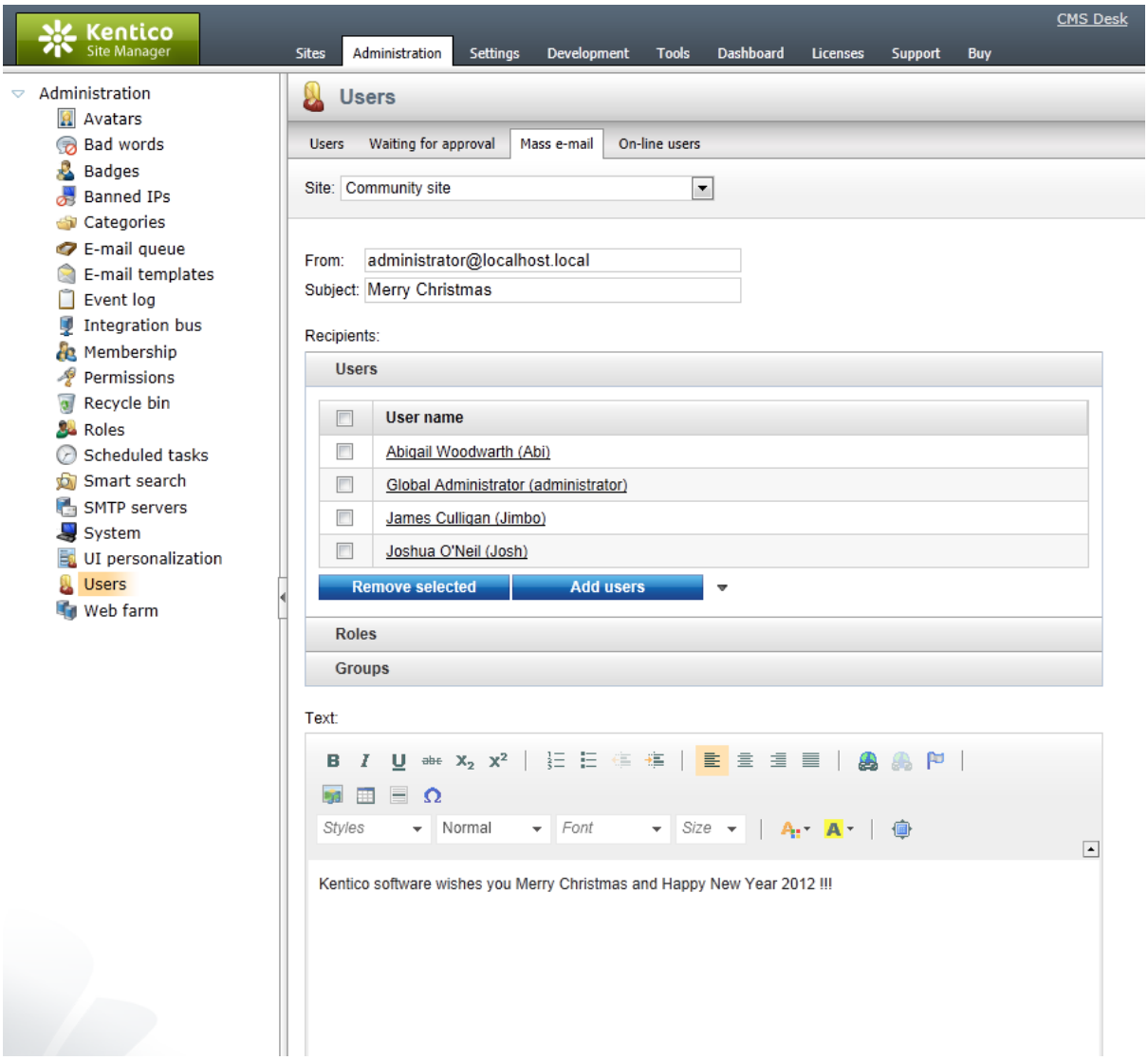
8.19.3 Sending mass e-mails

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.

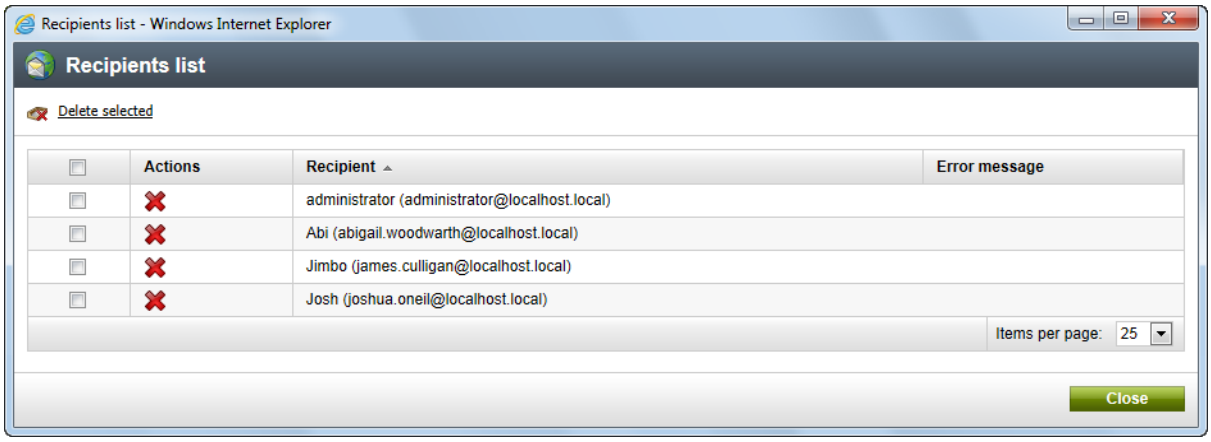
You can also add an attachment to the mail using the **Attach file** link.



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.

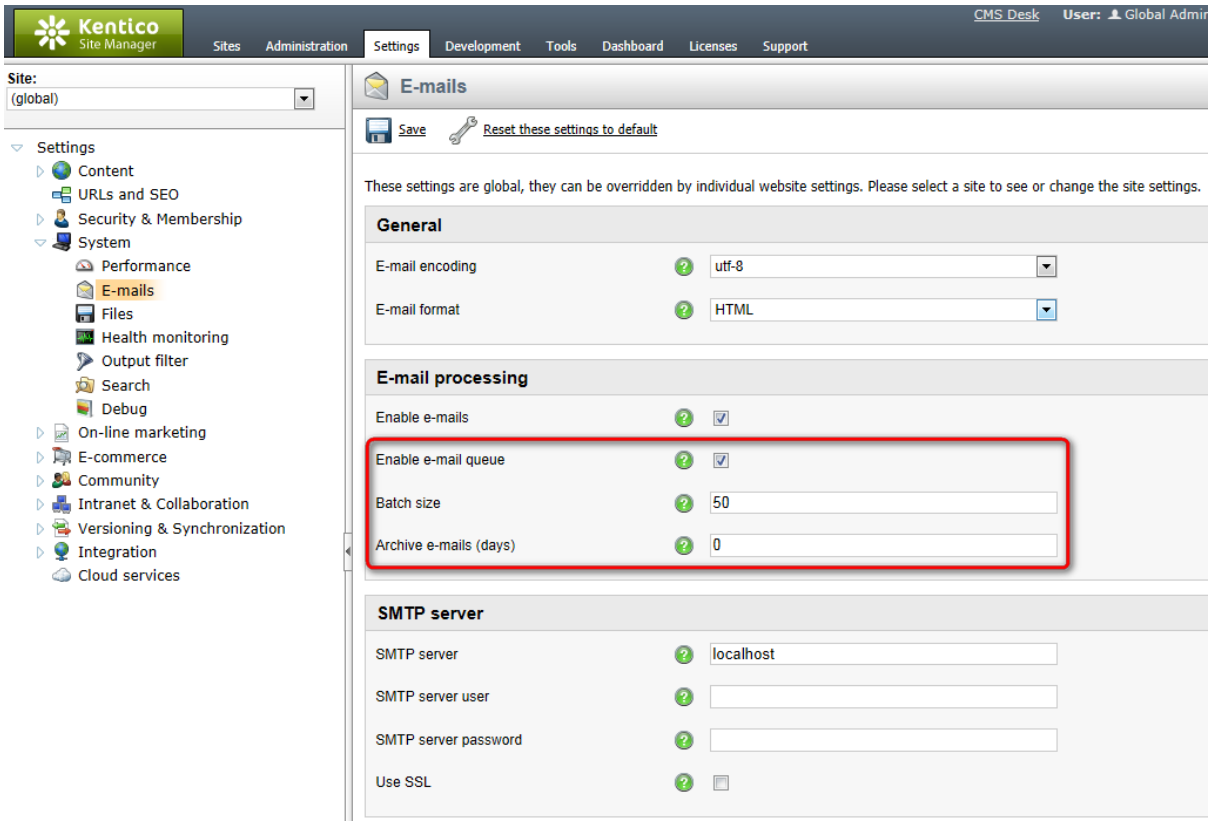
After clicking the link, a window will appear as in the screenshot below. You can delete recipients from the list, so that the mass e-mail will not be delivered to them. This can be done using the **Delete** (✖) icons or by selecting more recipients using the checkboxes and clicking the **Delete selected** link.



8.19.4 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.



8.20 Events

8.20.1 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.

8.20.2 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**.

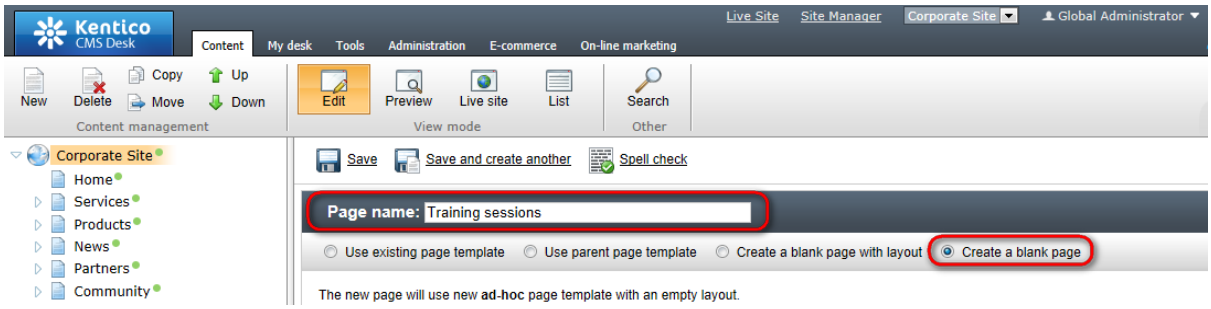
Example

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**: * *

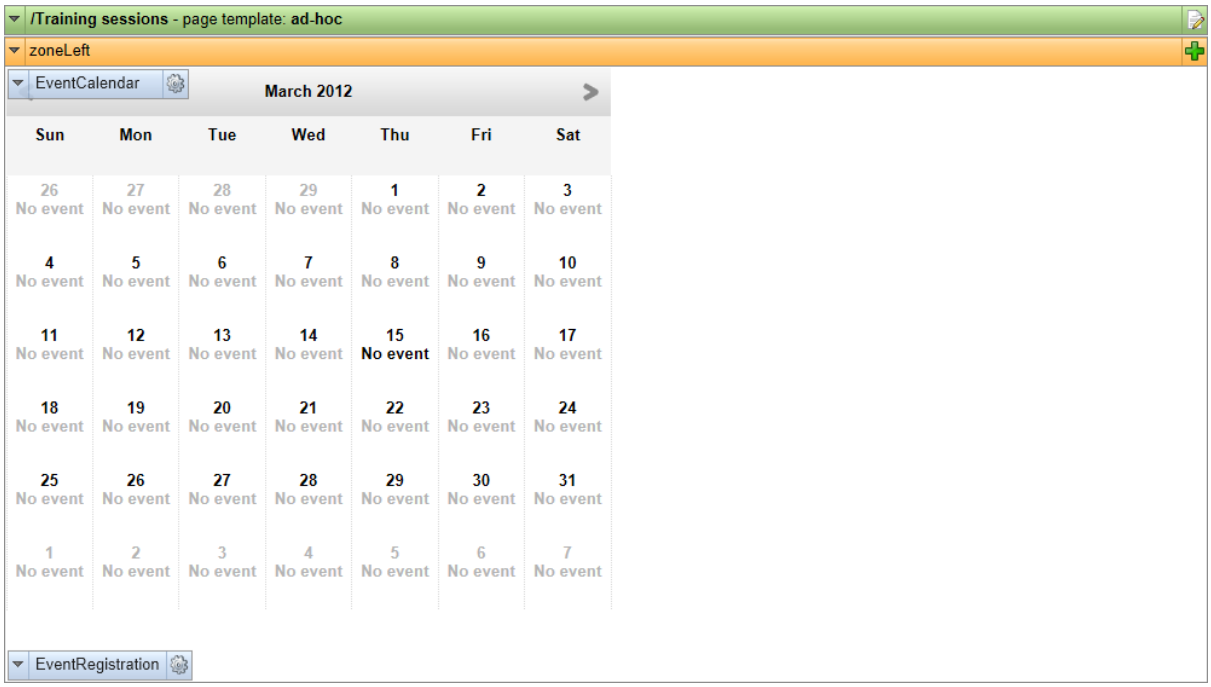
Leave the rest of the properties at their default values and click **OK**.

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.



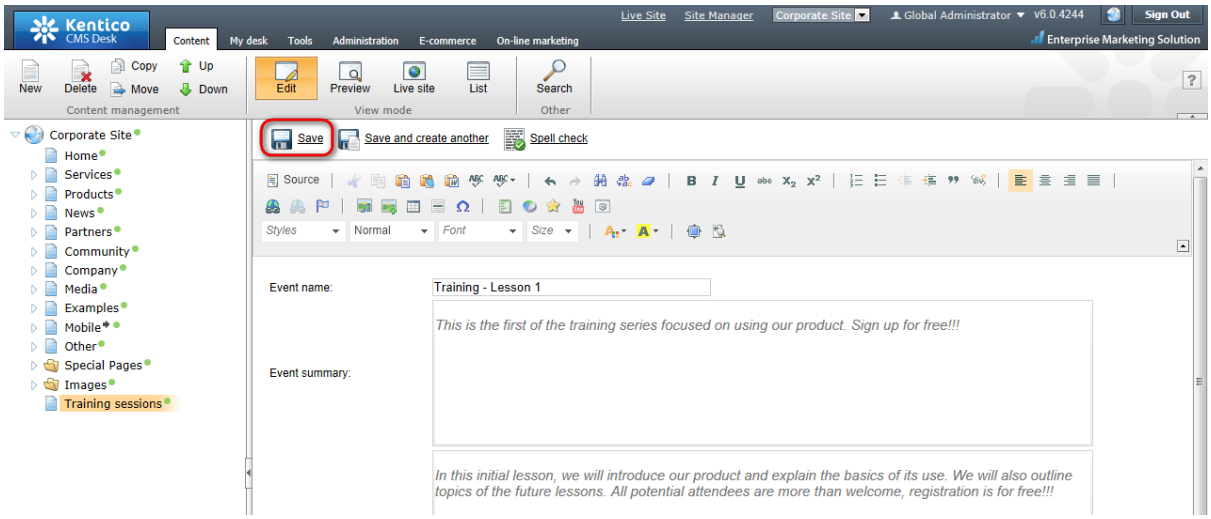
Part 2: Creating the events

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**.

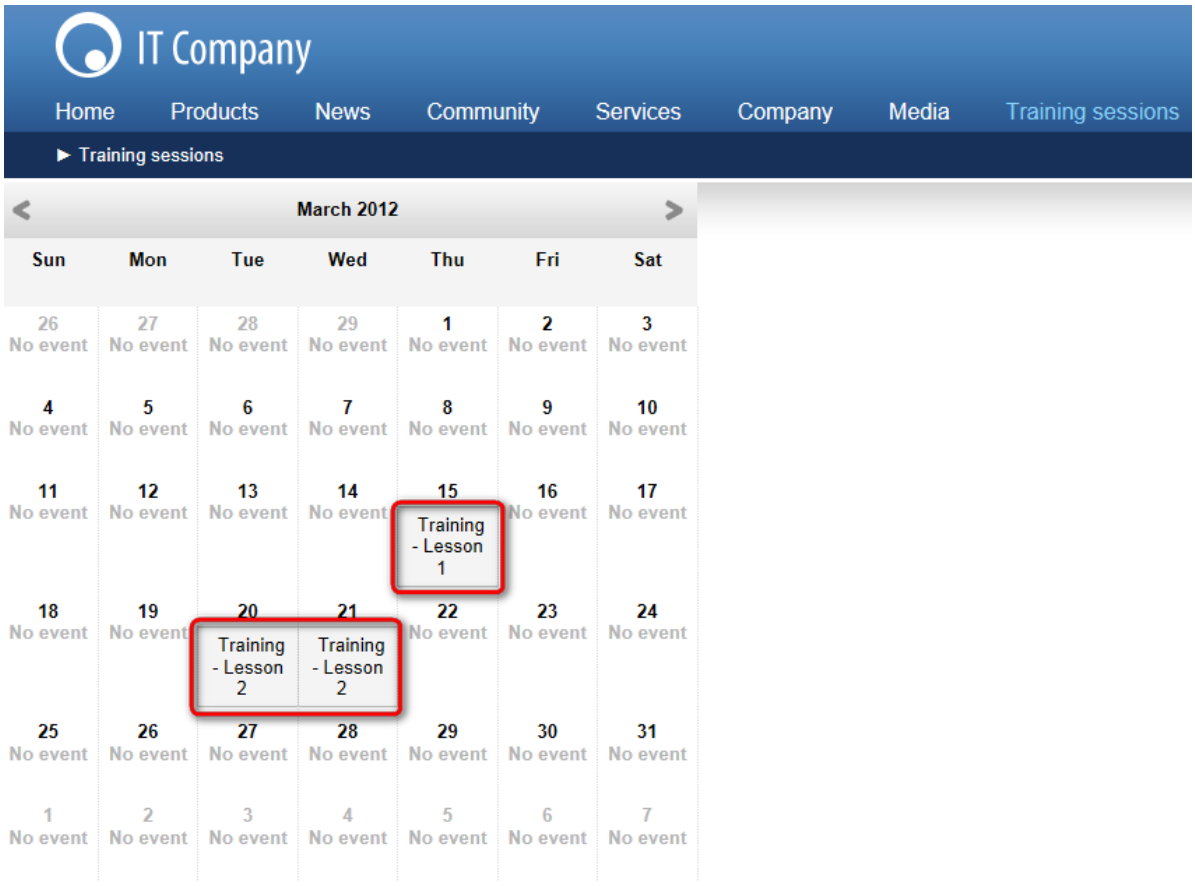


6. Now let's repeat the previous step and create a multi-day event. Select the **Training sessions** document again, click the **New** button and choose the **Event (booking system)** document type. This time, enter the following details:

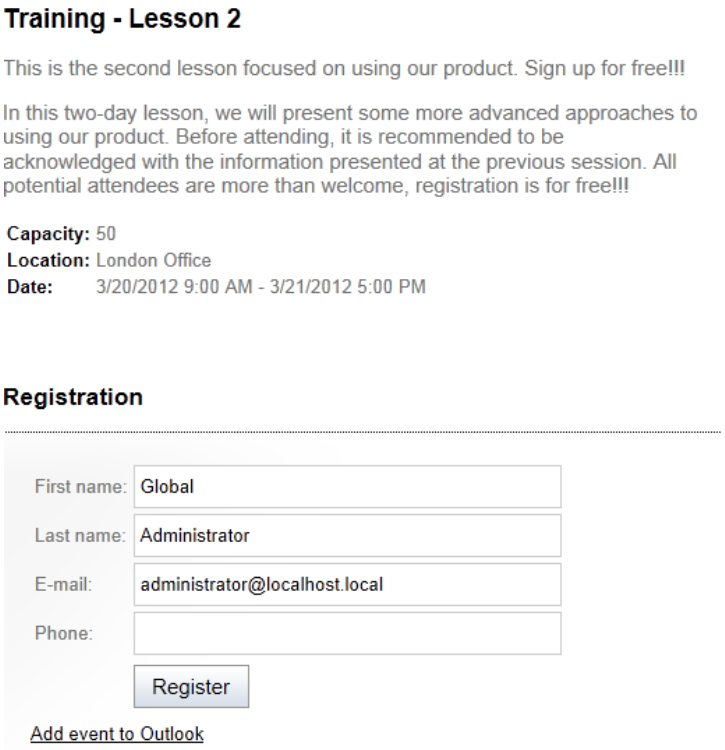
- **Event name**: *Training - Lesson 2*
- **Event summary**: *This is the second lesson focused on using our product. Sign up for free!!!*
- **Event details**: *In this two-day lesson, we will present some more advanced approaches to using our product. Before attending, it is recommended to be acknowledged 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.



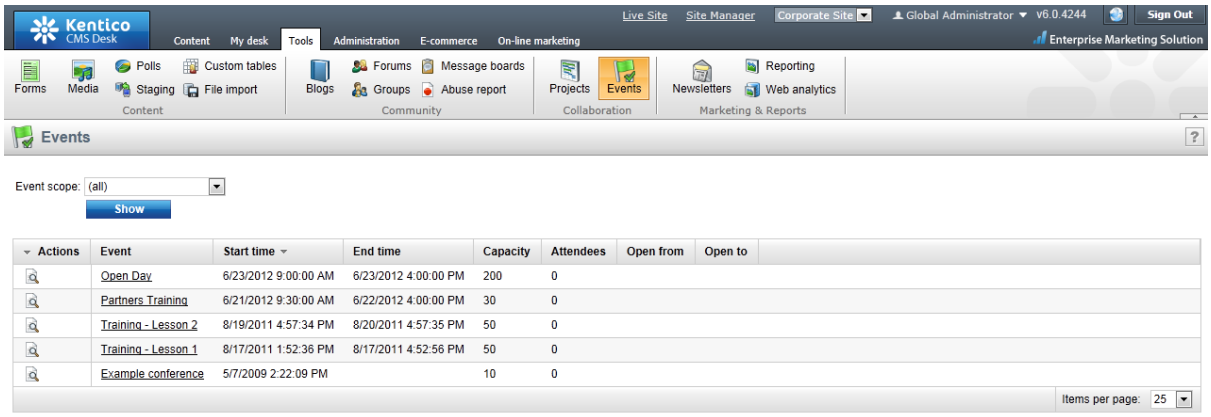
8. If you click the date field where an event is displayed, you should get its details displayed below the calendar. You should also see the registration form, enabling you to register for the event.



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.

8.20.3 Managing attendees

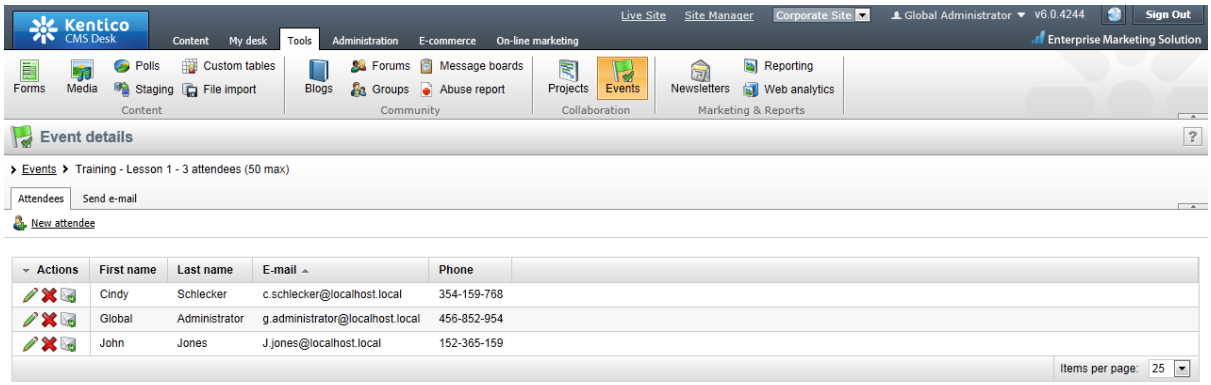
Events can be managed in **CMS Desk -> Tools -> Events**. Here, you can see a list of all events defined on the currently edited site, their capacity, current numbers of registered attendees, etc.



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.

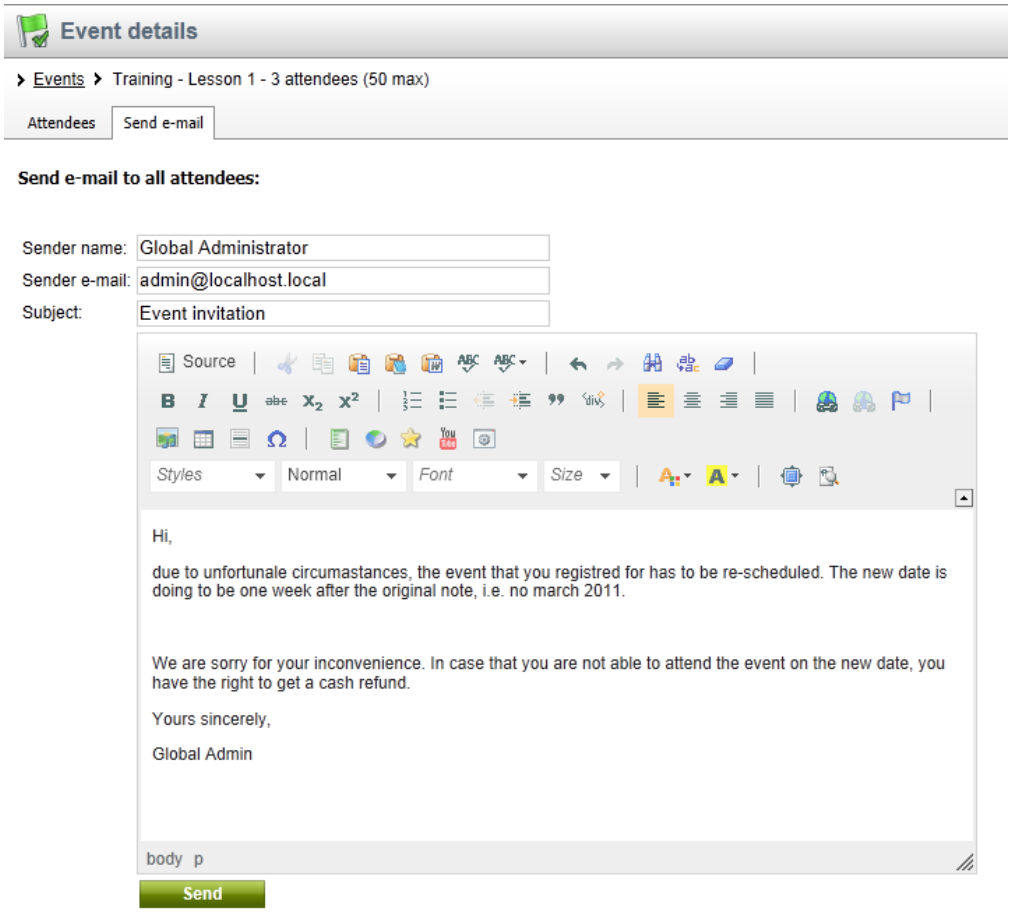
It is split into two tabs, while the **Attendees** tab will be displayed initially. Here, you can see a list of all attendees registered for the event. New attendees can be added to the event manually after clicking the **New attendee** link above the list. The following actions can be performed with 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.



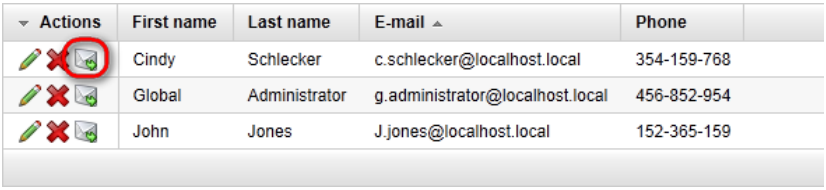
Sending e-mails to all attendees

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.

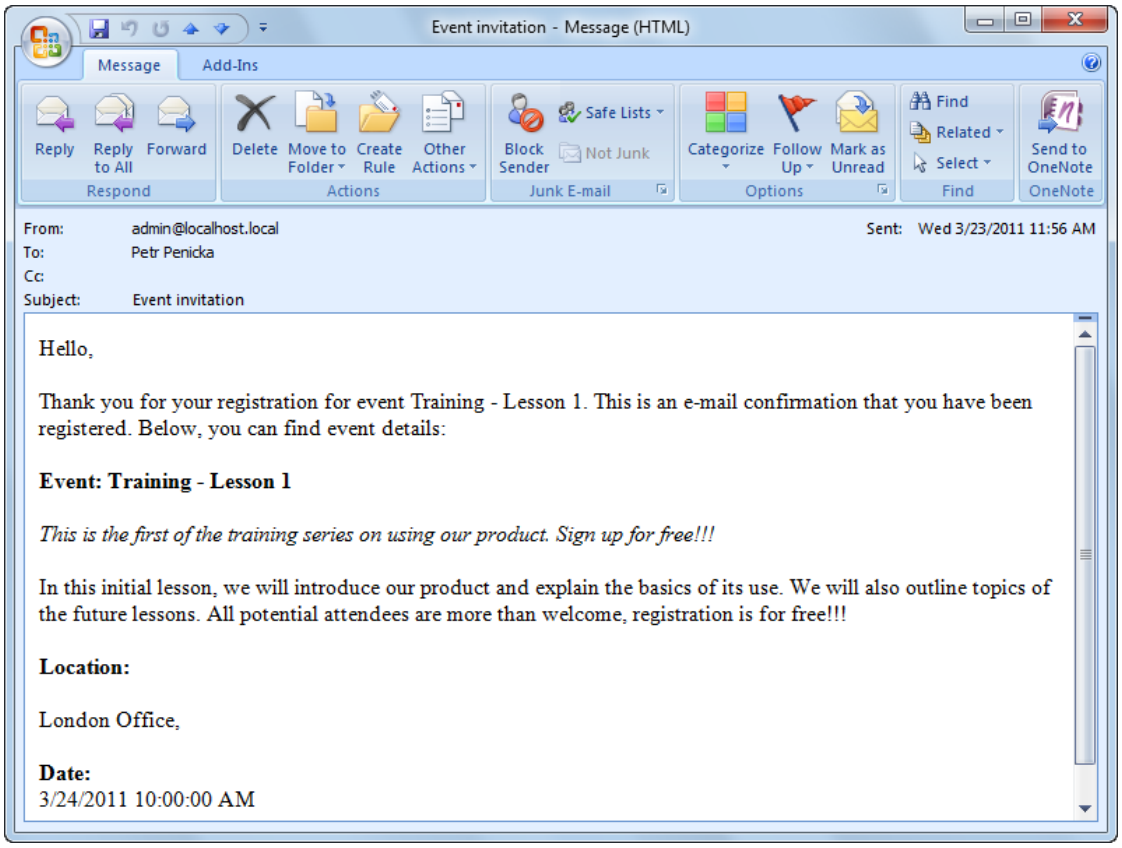


8.20.4 E-mail invitations

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.

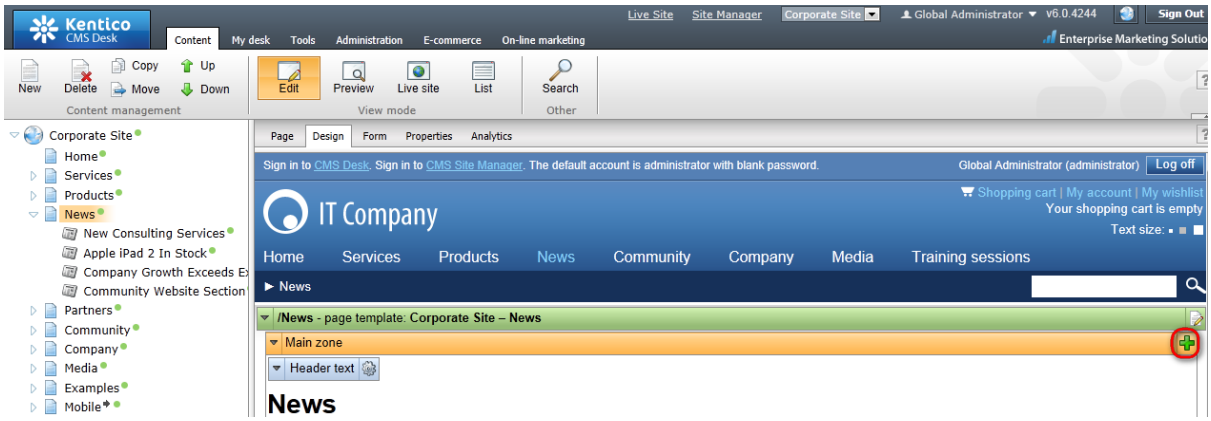


8.20.5 Using the Event calendar with other document types

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 **zoneLeft** 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

8.20.6 Security

Security settings of the Events module can be configured at two levels:

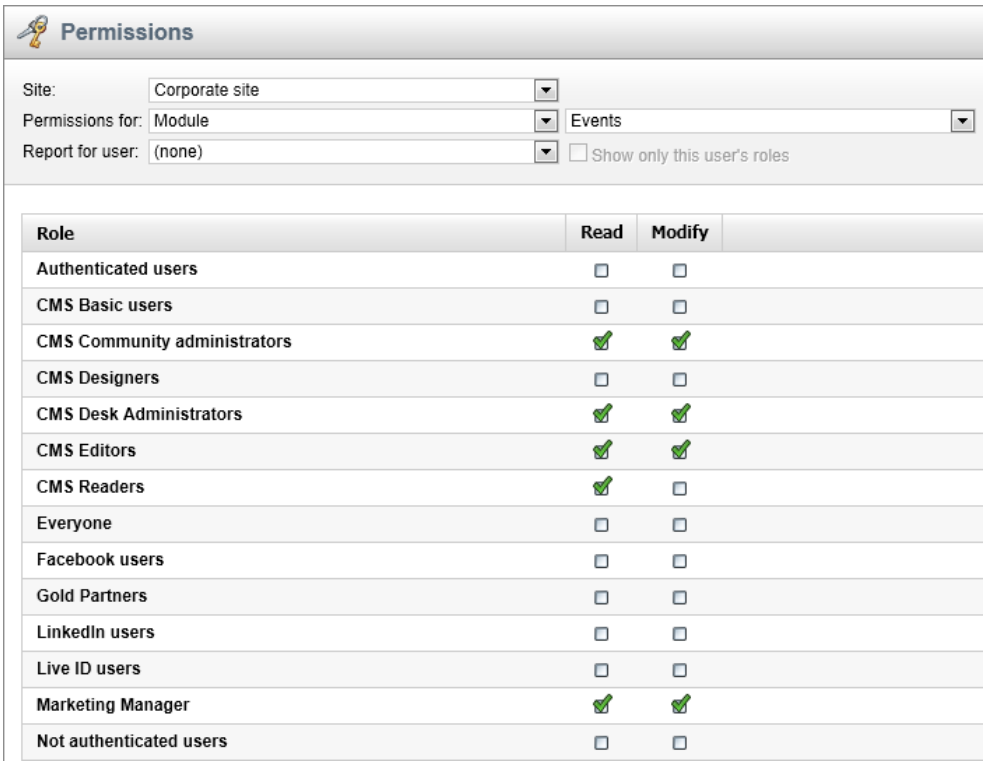
Management of events

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.



8.20.7 Events internals and API

8.20.7.1 Overview

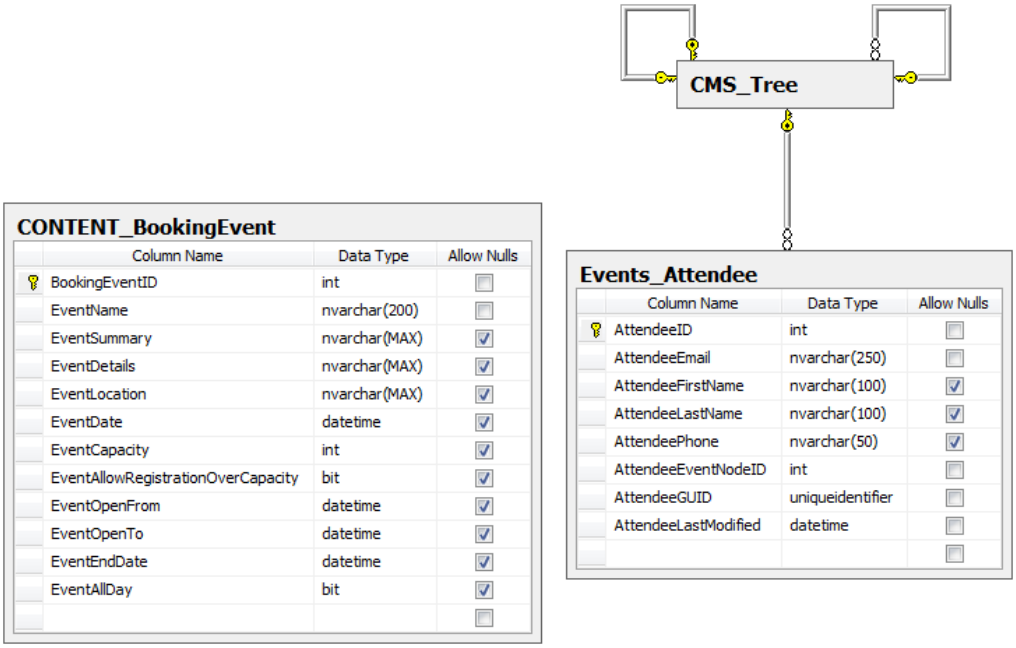
In this chapter, you will learn which database tables and API classes are used in the Events module. You will also see the most common API examples.

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.

8.20.7.2 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.



8.20.7.3 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.TreeEngine** 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.

8.20.7.4 API examples

8.20.7.4.1 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.TreeEngine;
using CMS.WorkflowEngine;
using CMS.PortalEngine;
```

8.20.7.4.2 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.DocumentPageTemplateID = 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.DocumentPageTemplateID = 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;
}
```

8.20.7.4.3 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;
}
```

8.21 Event log

8.21.1 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.

8.21.2 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:

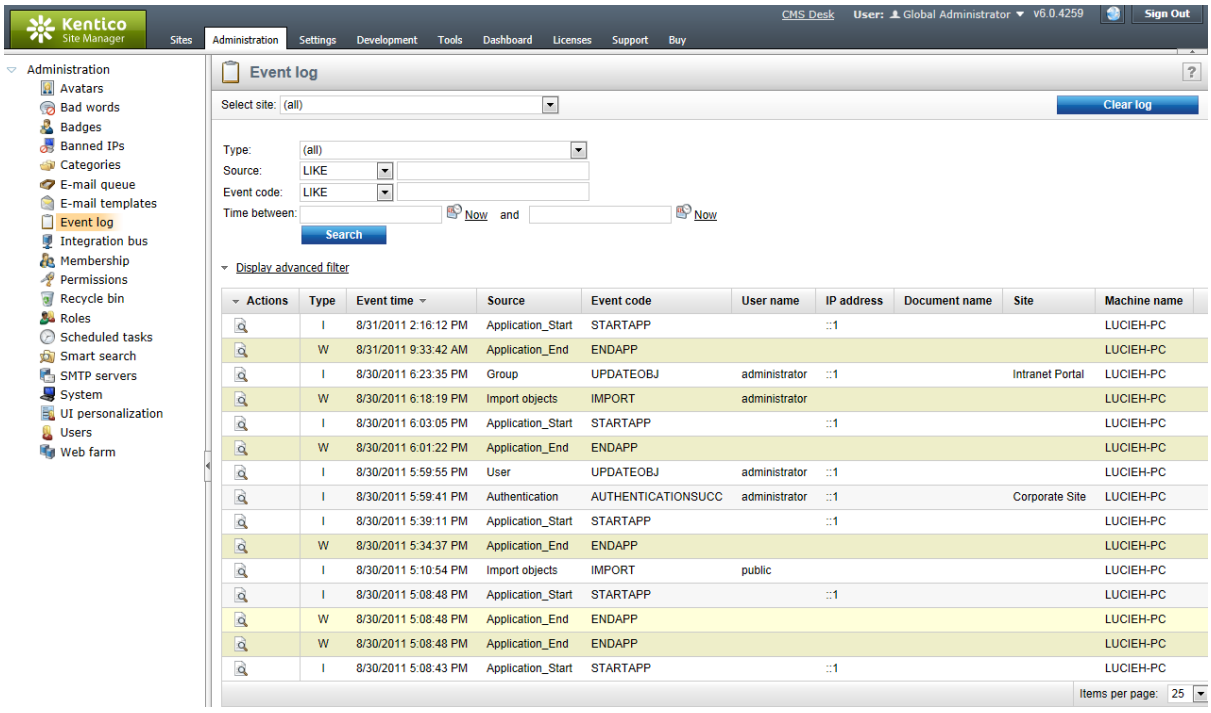
Select site	Using this drop-down list, you can select what events will be displayed. You can either select a particular website to display only event that occurred on the website, or you have the following extra options: • **(all)** - displays all events that occurred in the whole system. • **(only global 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.

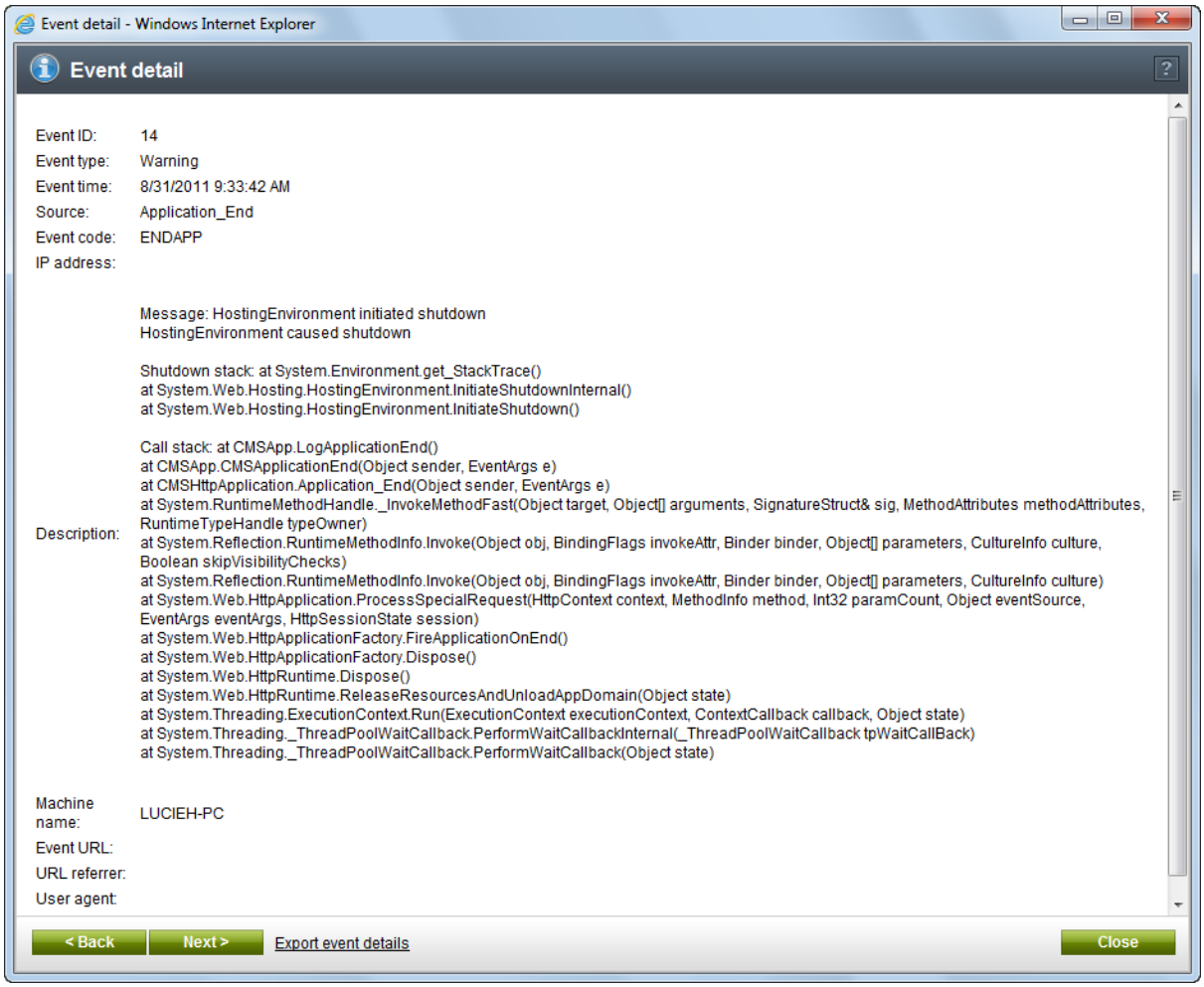


Event details

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.



8.21.3 Related settings

The following settings in **Site Manager -> Settings** are related to the Event log module:

Settings category	Description

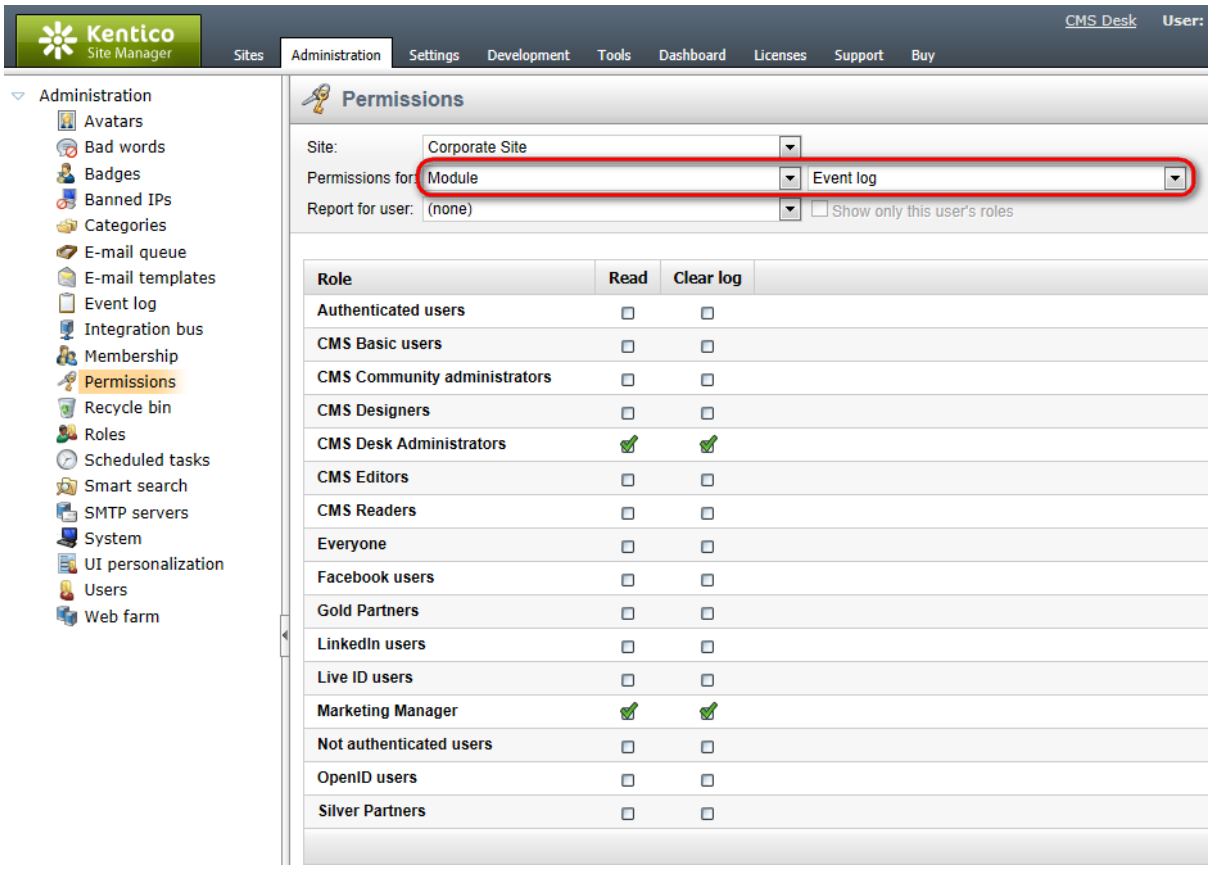
Content -> Log page not found exception	If enabled, page not found (404) exceptions will be logged as a warning in the event log.
System -> Event log size	Number of events stored in the event log. When exceeded by 10% (or a different percentage set by means of the *CMSLogKeepPercent* web.config key), 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.

8.21.4 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).



8.21.5 Event log internals and API

8.21.5.1 Overview

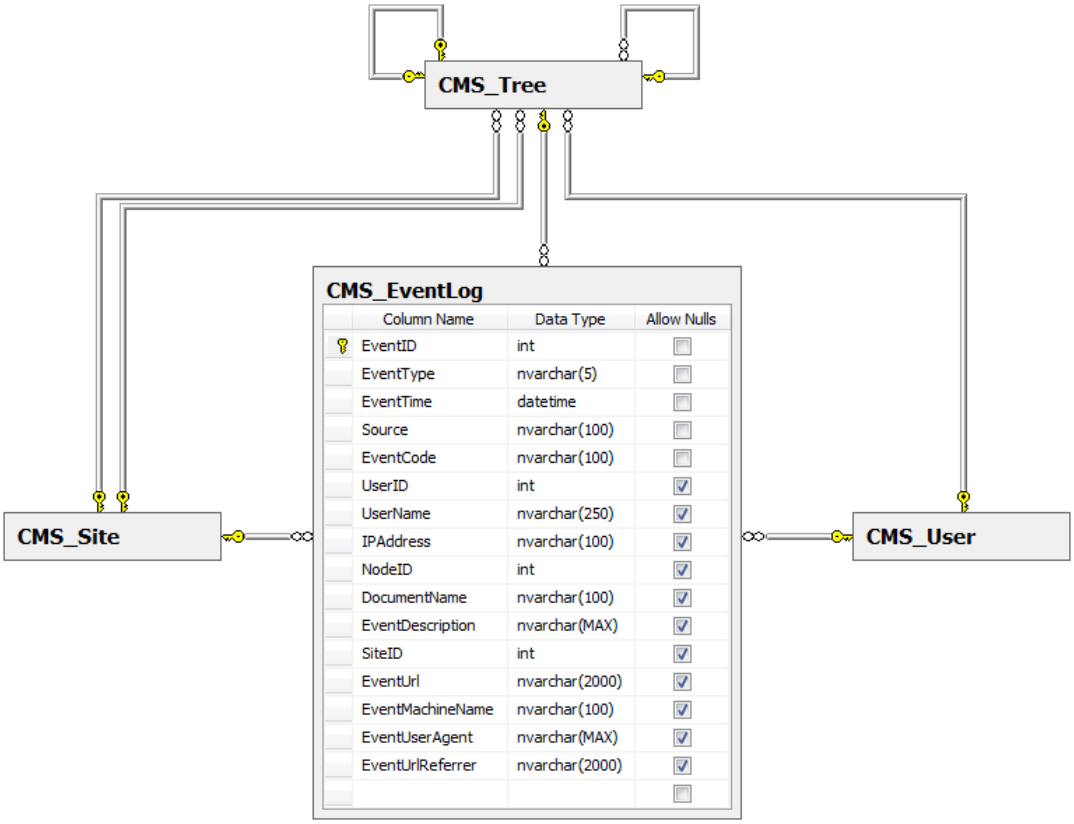
In this chapter, you will learn which database tables and API classes are used in the Events module. You will also see the most common API examples.

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.

8.21.5.2 Database tables

The Event log module uses the following database table:

- **CMS_EventLog** - contains records representing logged events.



8.21.5.3 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.

8.21.5.4 API examples

8.21.5.4.1 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;
```

8.21.5.4.2 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;
}
```

8.22 File import

8.22.1 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.

8.22.2 Importing files

The following is a step-by-step guide through the process of importing files:

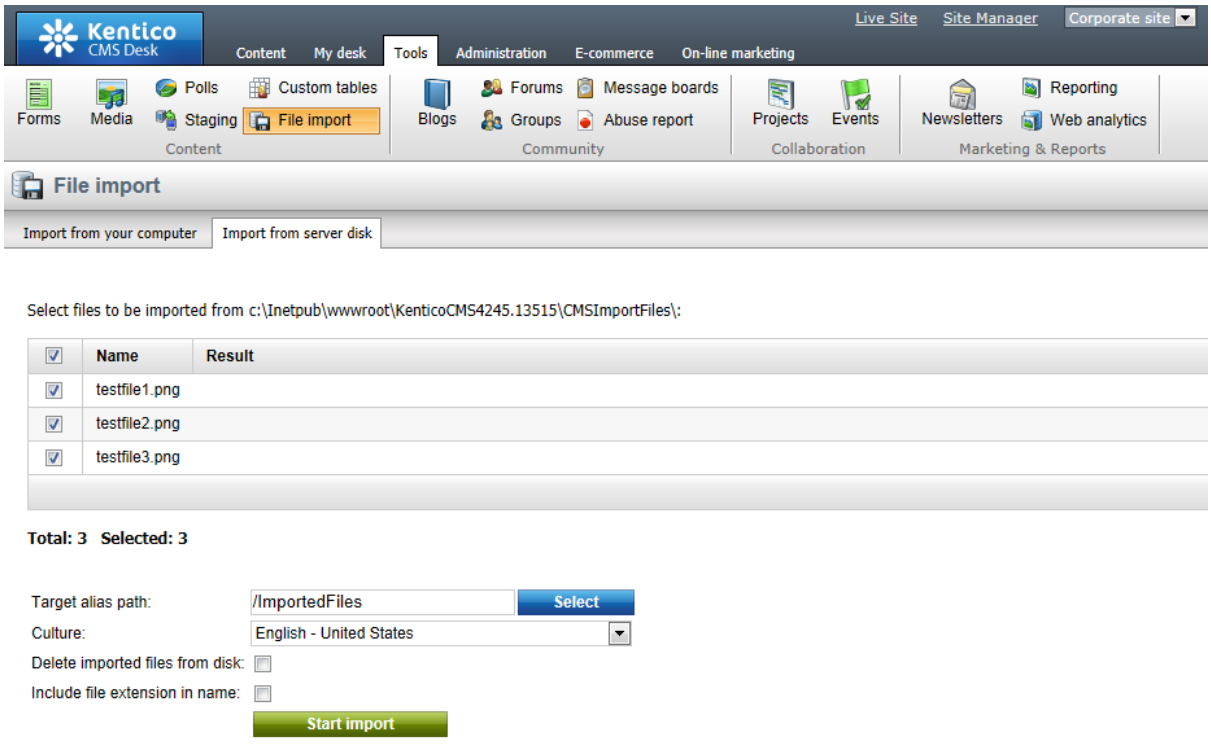
1. Copy your files into the specified import folder on your local disk.

2. Go to **CMS Desk -> Tools -> File import**. You should see a list of files that are currently in the import folder.

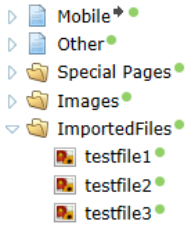
3. Choose which files should be imported using the checkbox next to each file and specify the following properties:

- **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**.



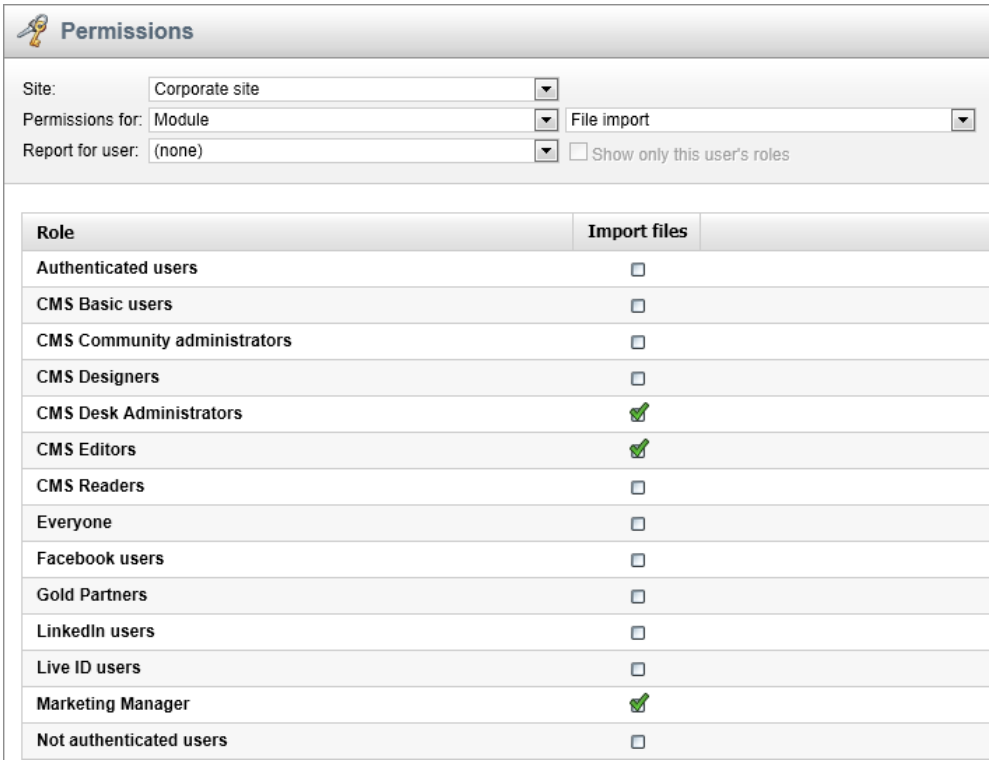
4. Now if you switch to the **Content** tab and locate the alias path that you specified in the previous step, you should see the files uploaded.



8.22.3 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:

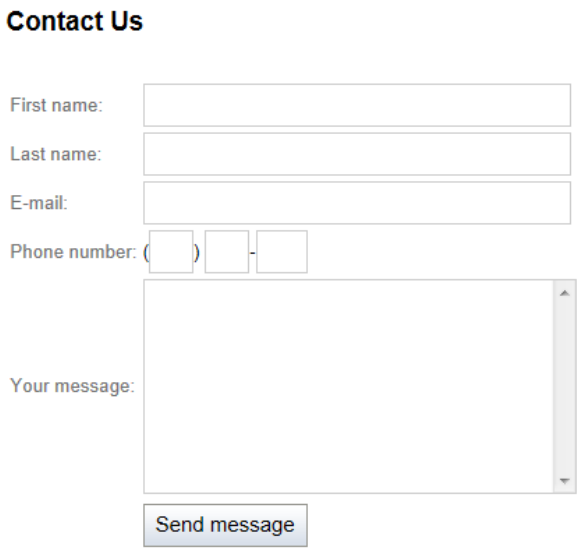
- **Import files** - members of the specified roles are allowed to import files using the File import module



8.23 Forms

8.23.1 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.



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.

8.23.2 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** link. Enter the following details:

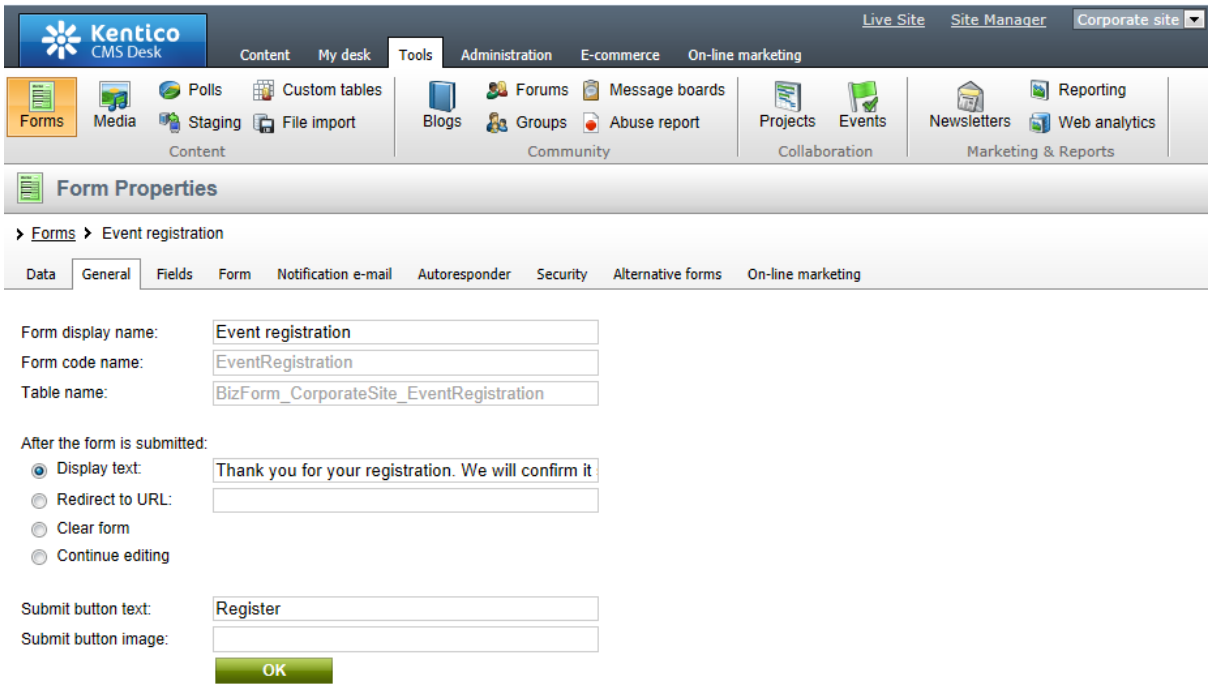
- **Form display name:** Event registration
- **Form code name:** EventRegistration
- **Table name:** Form_<current site code name>_EventRegistration

Click **OK**.

2. You will be redirected to the **General** tab of the new form's editing interface. Enter the following values:

- **Display text:** Thank you for your registration. We will confirm it shortly by e-mail.
- **Submit button text:** Register

Click **OK**.



3. Now we will define the form fields. Go to the **Fields** tab. Add the following fields using the **Add attribute** ( ) button. For each field, enter the values, click **Save field** 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

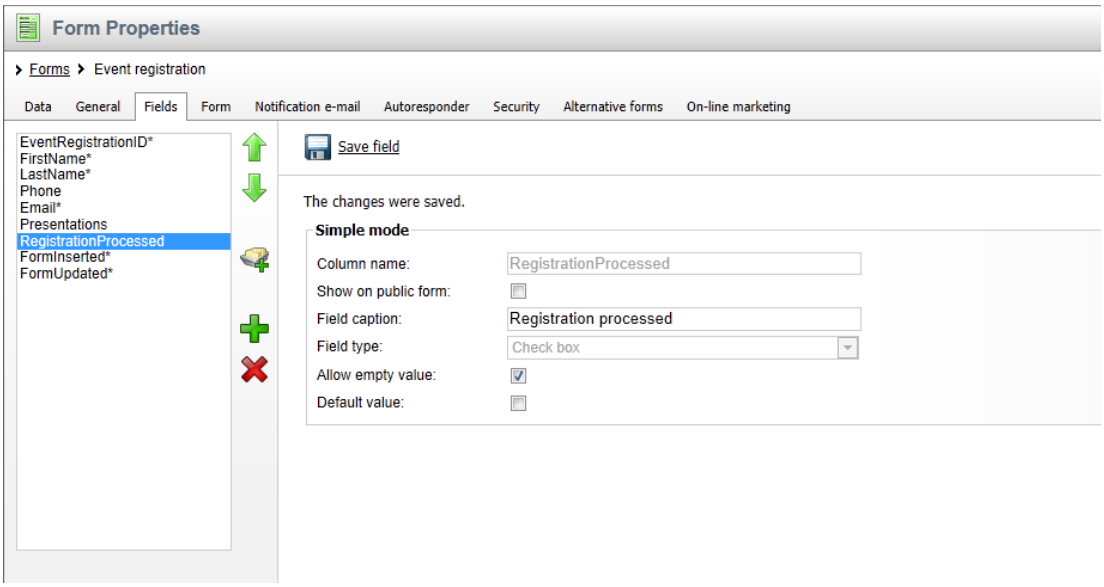
- **Column name**: Email
- **Show on public form**: enabled
- **Field caption**: E-mail
- **Field type**: E-mail
- **Maximum length**: 100
- **Allow empty value**: disabled

- **Column name**: Presentations
- **Show on public form**: enabled
- **Field caption**: Presentations you want to visit
- **Field type**: Multiple choice
- **Options**: 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*, depending on the selected **Field type**. This gives you almost unlimited form flexibility and customization options. Please see the Development -> Form controls chapter for deeper 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.

8.23.3 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

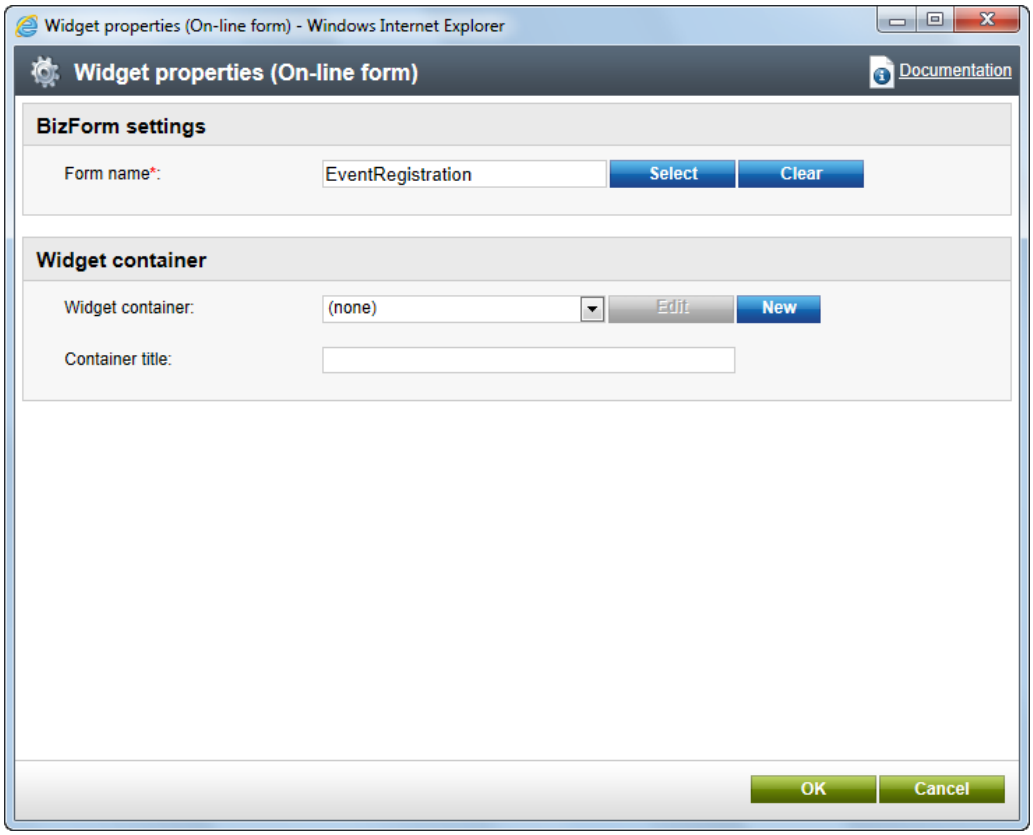
In this example, we will add the *Event registration* form, created in the Creating a new form topic, to a page using the **Insert Form** ( ) button on the WYSIWYG editor toolbar. Any other form can be added

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 **Insert Form** ( ) button on the WYSIWYG editor toolbar, which will create a form inline widget.

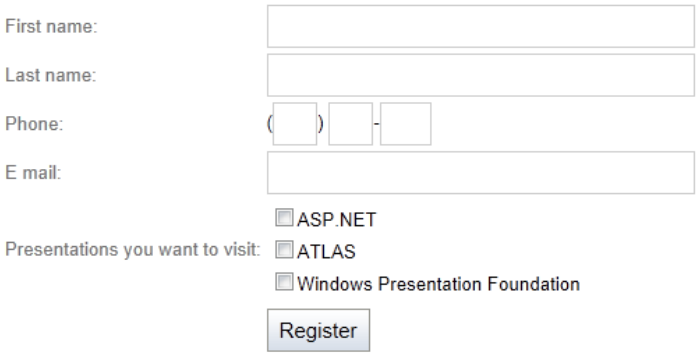
3. The configuration dialog of the widget will be opened. Click the **Select** button next to the **Form name** field and choose the **Event registration** form.



Click **OK**.

A placeholder image will be pasted into the text. The properties of the 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.

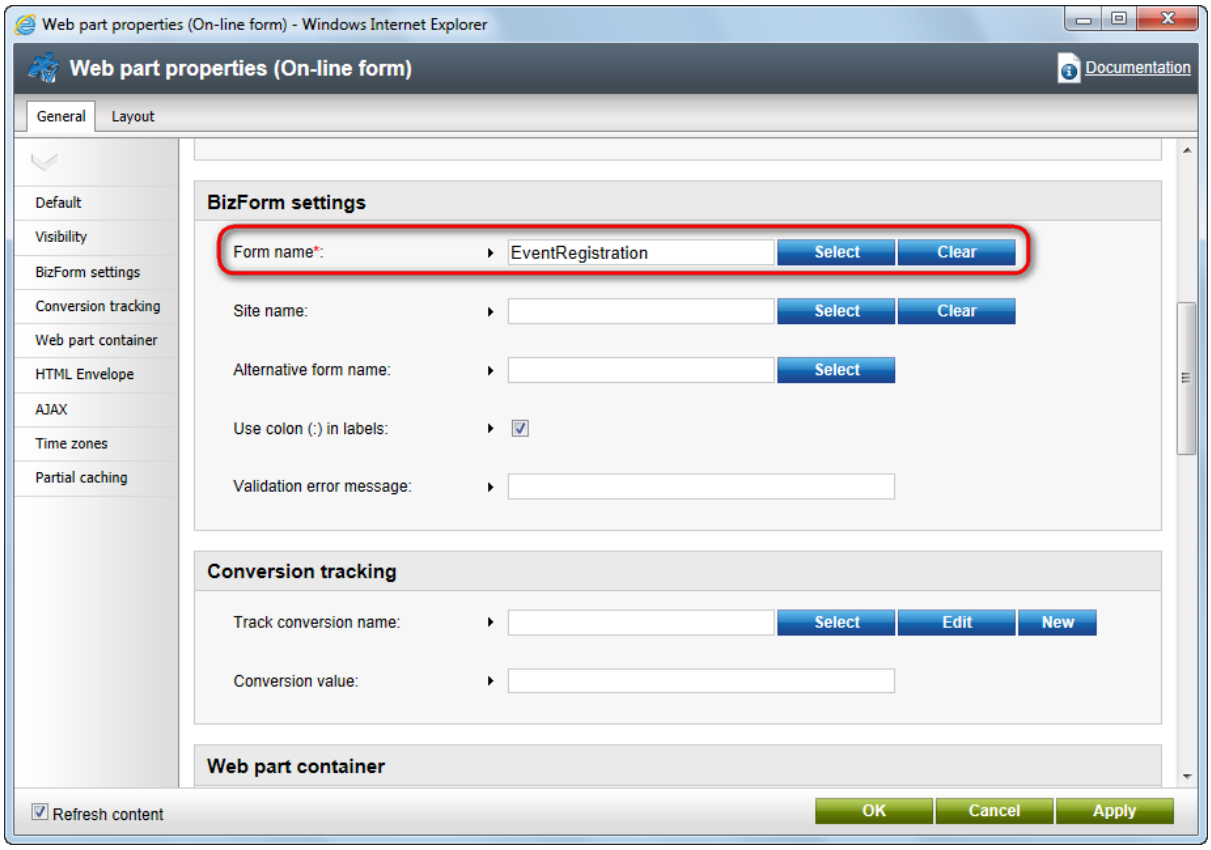
Example 2: Adding a form using the On-line form web part

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.

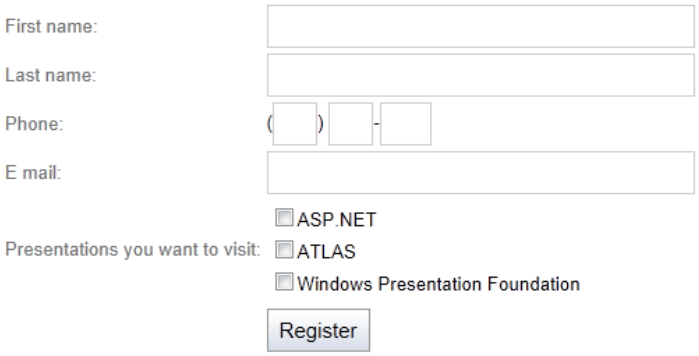
1. Go to **CMS Desk -> Content** and select some page with a web part zone. View the page in the **Edit -> Design** mode.

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.

8.23.4 Managing form data

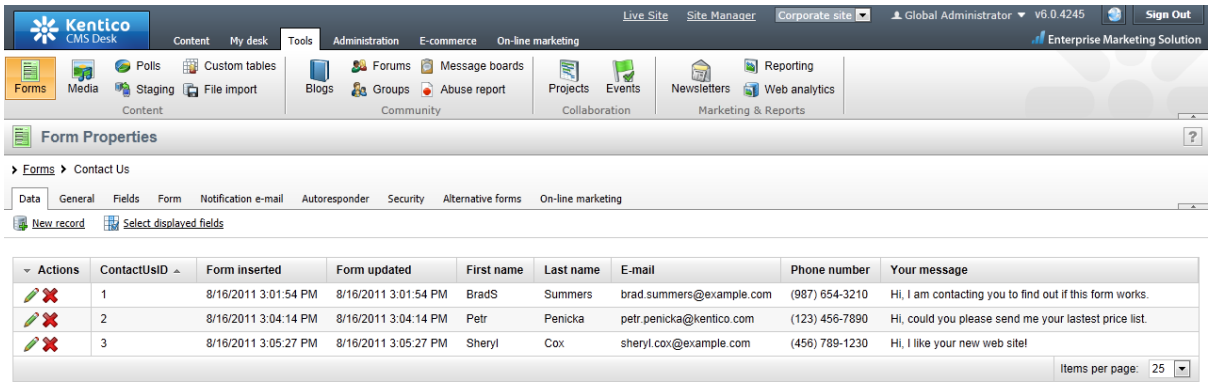
Form records (data submitted by website users) can be viewed and managed in **CMS Desk -> Tools -> Forms**. In the list of forms, you need to click the **Edit** (✎) icon next to a particular form. You will get redirected to the **Data** tab of the form's editing interface, where particular records are listed.

Each of the records listed in the table has the following options:

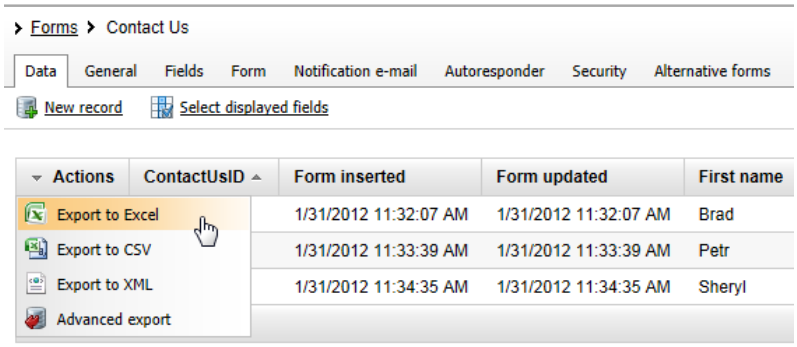
- **Edit record ( )** - allows you to alter the record.
- **Delete record ( )** - allows you to delete the record.

You can also perform the following actions using the links above the listing:

- **New record** - creates a new record in this form.
- **Select displayed fields** - enables you to set which columns should be displayed in the listing.



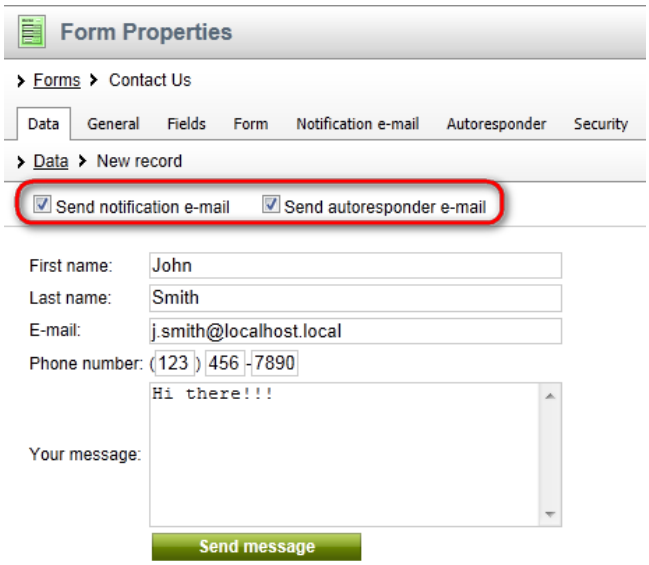
A form's records may be exported into an external file using either the XLSX (Excel), CSV or XML format. This can be done by clicking the icon in the header of the **Actions** column in the grid and then selecting the appropriate option from the context menu.



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. On top of it, you can decide if notification and autoresponder e-mails will be sent when you save the record:



8.23.5 Notification and autoresponder e-mails

8.23.5.1 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-mail notifying the person responsible for forms data management (content editor, administrator, ...) about the new 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.

8.23.5.2 Notification e-mails

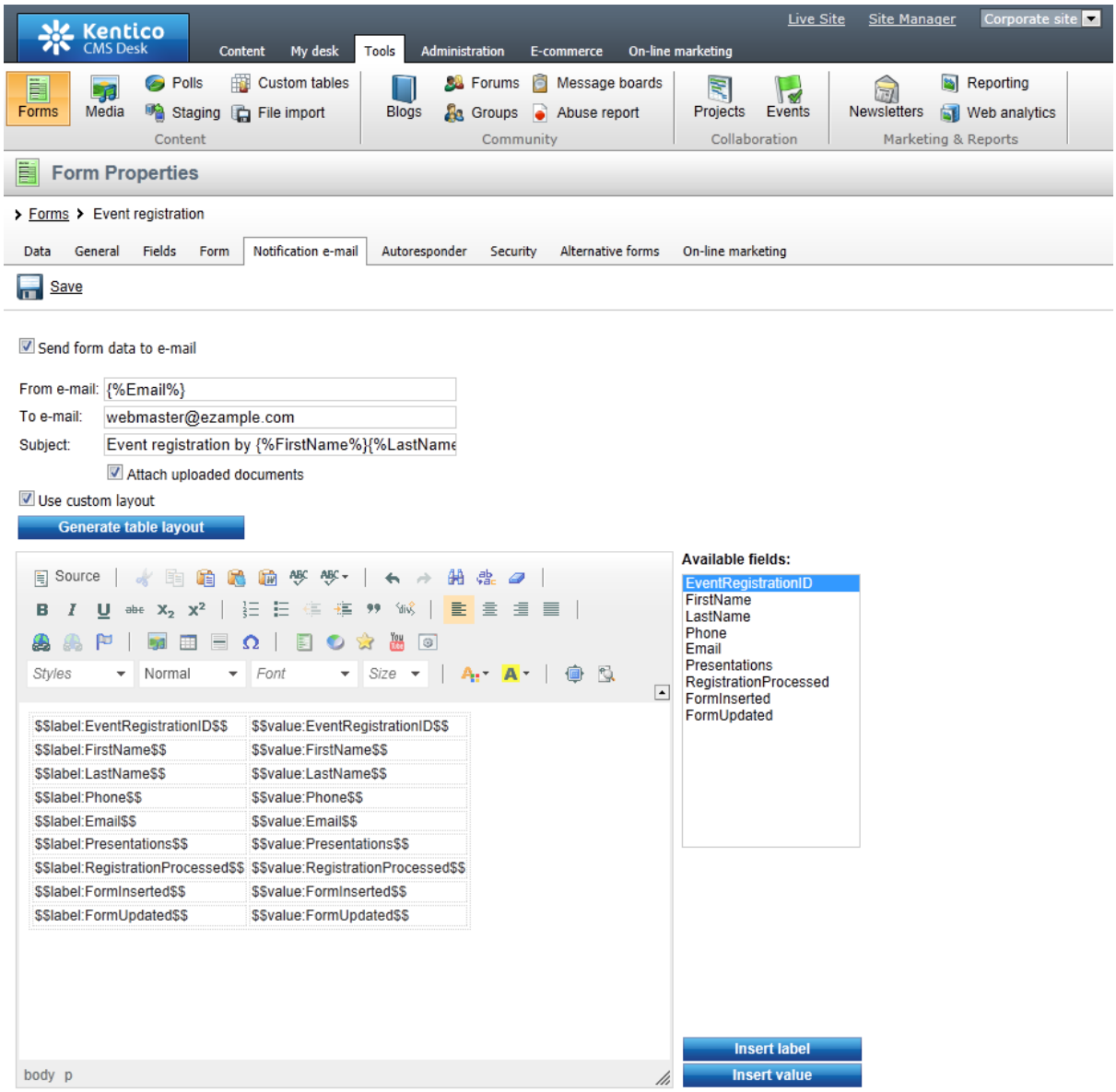
Notification e-mails can be configured on the **Notification e-mail** tab of a form's editing interface. To enable them, you first need to enable the **Send form data to e-mail** option. Then you need to configure the following options:

- **From e-mail -** e-mail address of the user who submitted the form; you will typically use a macro to get a value from a field of 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 address of the person responsible for management of form records
- **Subject** - subject of the notification e-mails
- **Attach uploaded documents** - enable this option if you want to attach documents submitted via the form (if there are some) to the e-mails
- **Use custom layout** - if disabled, notification e-mails' body will contain all field names with the entered values, each on a single line; if enabled, the text area is displayed, letting you define custom

layout of the e-mail body

- **Generate table layout** - this button generates a table with 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 values of these fields, as described in the Using macros with forms topic. With all the options configured, click **Save** to save your configuration.



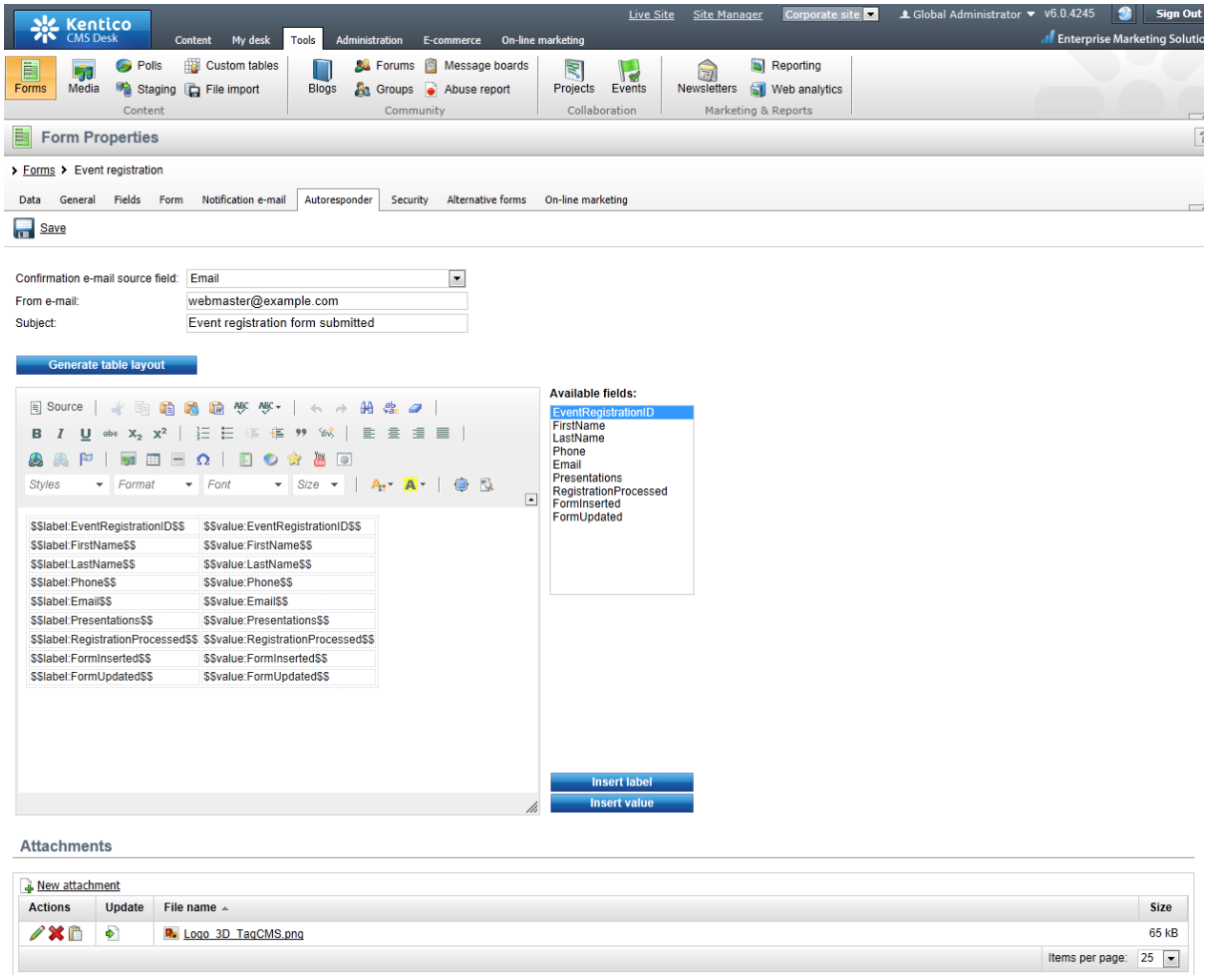
8.23.5.3 Autoresponder e-mails

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, just the **Confirmation e-mail source field** drop-down list will be visible. Expand it and choose the field which contains the e-mail address that the automatic response

should be sent to. This means that autoresponder e-mails can only be used with forms where such a field is present (however, this is the case of most real-world on-line forms). After choosing the field, the following extra options will be displayed:

- **From e-mail** - e-mail address of the sender of the autoresponder e-mails (the *From* field of the e-mail message)
- **Subject** - subject of the autoresponder e-mails
- **E-mail body** - the main text area allows you to define the body of the autoresponder e-mails
- **Attachments** - this option allows you to attach a file to each autoresponder e-mail, such as event agenda, white papers, etc.
- **Generate table layout** - this button generates a table with 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 values of these fields, as described in the Using macros with forms topic. With all the options configured, click **Save** to save your configuration.



8.23.6 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	QueryString	Cookie	Custom	Path
Fields tab **- Default value** attribute	•	•	•	•	•	•
Notification e-mail and **Autoresponder** tabs - all fields	•	•	•	•	•	
General tab - **Form display name, Display text , Submit button text; Fields** tab - **Field caption; Form** tab - **Table layout**	•					
General tab - Redirect to URL		•				

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.

8.23.7 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:

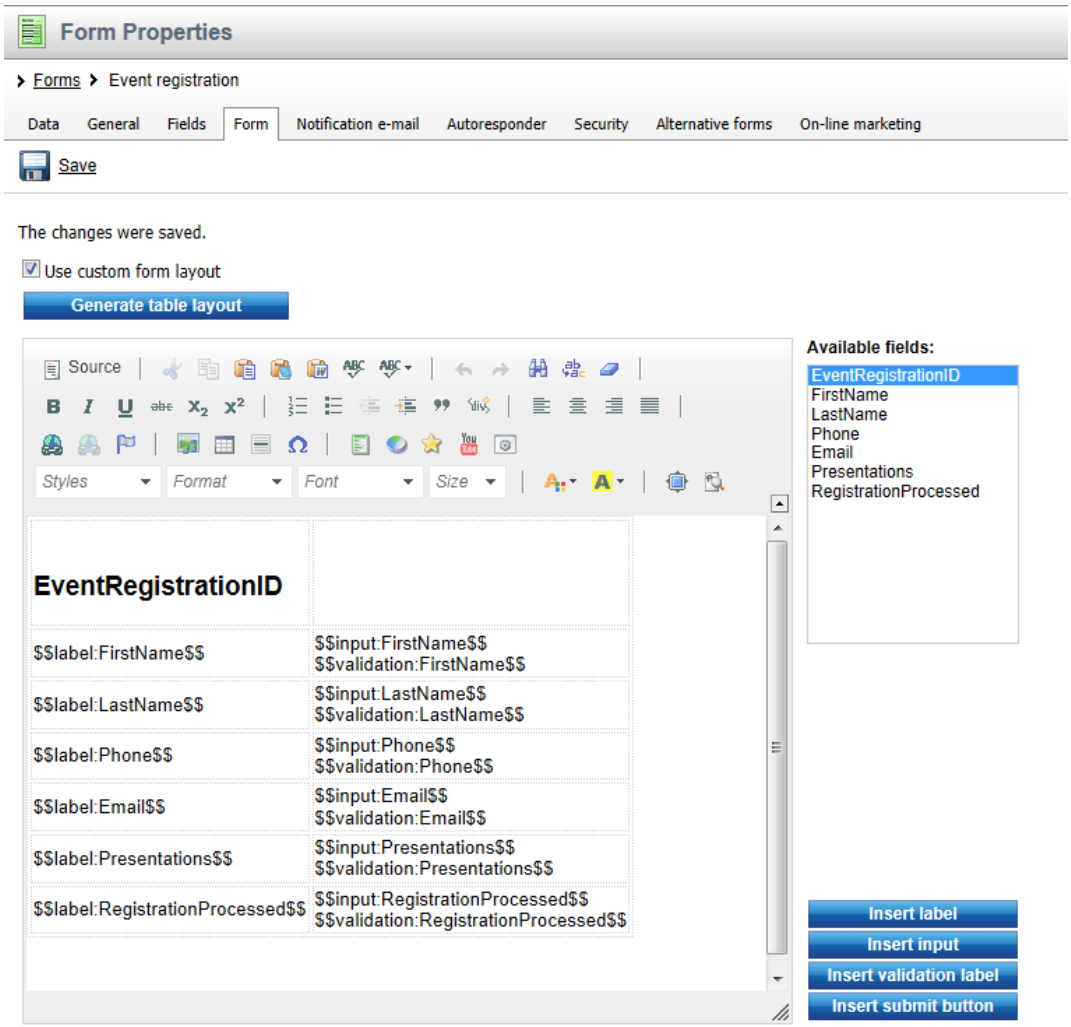
1. Go to **CMS Desk -> Tools -> Forms** and click the **Edit** () icon next to the form.

2. In its editing interface, go to the **Form** tab and check the **Use custom form layout** check-box.

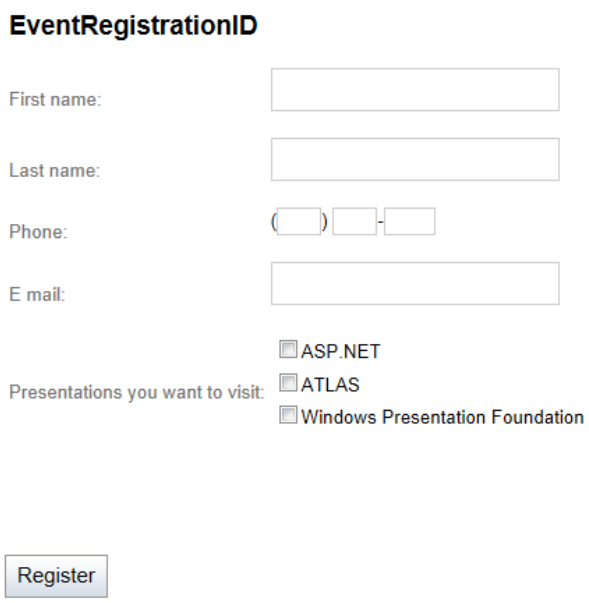
3. Click **Generate table layout** button. This will quickly generate a table containing all fields of the form, with appropriate labels, input controls and validation controls. Alternatively, you may use the **Insert label**, **Insert input** and **Insert validation label** buttons on the right to achieve 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 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 choose **Format: Heading 2** in the WYSIWYG editor toolbar. You may also delete the first line of the table (*EventRegistrationID*), which is not wanted to appear on the public form.

The result should look as in the screenshot below.



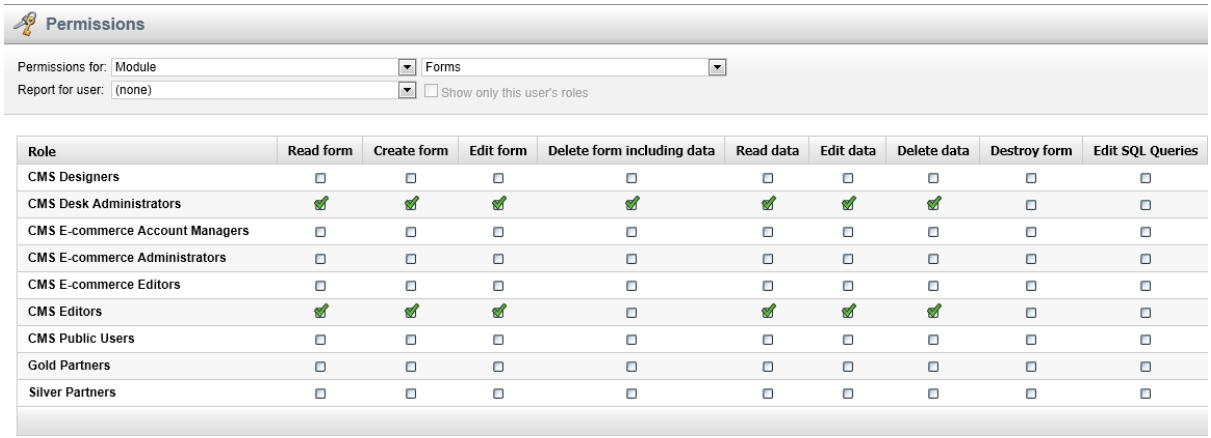
5. Click **Save** to save the new layout. Now if you add the form to the live site, it should look as in the following screenshot:



8.23.8 Security

Access to the Forms module can be managed in **CMS Desk -> Administration -> Permissions**, after you select the **Modules -> Forms** permission matrix. The Forms module has the following permissions:

- **Create form** - members of the role are allowed to create new forms
- **Delete data** - members of the role are allowed to delete existing form records
- **Delete form including data** - members of the role are allowed to delete forms including stored records
- **Edit data** - members of the role are allowed to create and edit form records
- **Edit form** - members of the role are allowed to edit form configuration, fields and layout (not the actual records)
- **Read data** - members of the role are allowed to view form records
- **Read form** - members of the role are allowed to view form configuration, fields and layout (not the actual records)

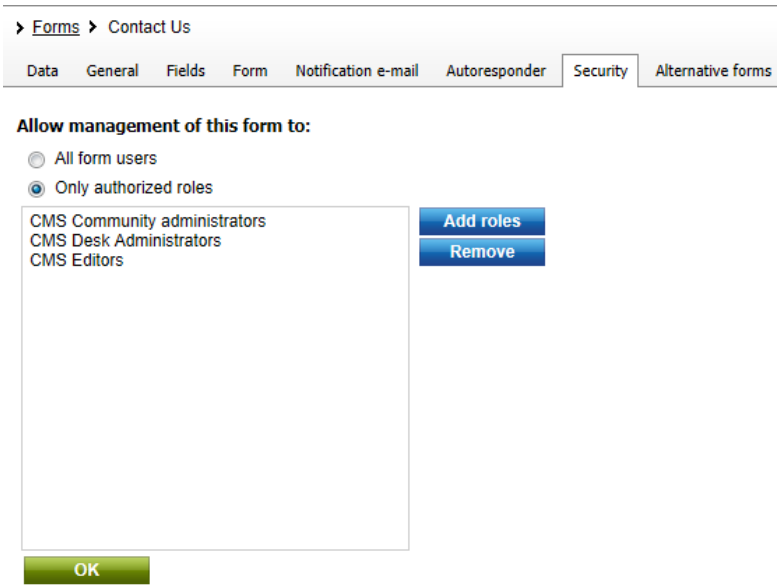


Security for particular forms

Roles which are authorized to read and modify a form and its data can also be specified on form level. To do this, edit ( ) a particular form and switch to its **Security** tab. The following two options are available there:

- **All form users** - all users with access to the **Tools -> Forms** section can edit the form
- **Only authenticated users** - only members of roles added to the box will be allowed to edit 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 the role is granted with access to a particular form is not sufficient - the form-level settings only define if the particular form will be listed in **Tools -> Forms**.



8.23.9 Form files settings

Files may be submitted as part of form records, typically when a form contains an uploader type field (*Upload file* 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	This setting is only applied when a **Custom form files folder** is

for custom form files folder	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.

8.23.10 Forms internals and API

8.23.10.1 Database tables and API classes

The Forms module uses the following database tables:

- **Form_<website code name>_<table name>** - each form has its own database table with name in this format, e.g. *BizForm_CorporateSite_EventRegistration*; this table contains all fields that you specified on the **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 this table; you can recognize them by the *BizForm.* prefix in the *ClassName* column.

The Forms API is provided by the following **CMS.SiteProvider** namespace classes:

- **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.

8.23.10.2 Creating a new record

The sample code below demonstrates how a new record can be added to a form using the API.

[C#]

```
using CMS.FormEngine;
using CMS.SettingsProvider;
using CMS.DataEngine;
using CMS.GlobalHelper;

...

        string bizFormName = "TestingSiteContactUs";
        string siteName = "CMSTestingSite";

        // 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 = new BizFormItem(dci.ClassName, bProvider);

            // Insert some data
            formRecord.SetValue("FirstName", "Alice");
            formRecord.SetValue("LastName", "Cooper");
            formRecord.SetValue("Email", "alice@email.com");
            formRecord.SetValue("Message", "Hallo world");
            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);
            }
        }
```

8.23.10.3 Updating a record

This code example shows how an existing form record can be modified.

[C#]

```
using CMS.FormEngine;
using CMS.SettingsProvider;
using CMS.DataEngine;
using CMS.GlobalHelper;

...

        string bizFormName = "TestingSiteContactUs";
        string siteName = "CMSTestingSite";

        // 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.";
                }
            }
        }
```

8.23.10.4 Deleting a record

This code example shows how a form record can be deleted.

[C#]

```
using CMS.FormEngine;
using CMS.SettingsProvider;
using CMS.DataEngine;
using CMS.GlobalHelper;

...

        string bizFormName = "TestingSiteContactUs";
        string siteName = "CMSTestingSite";

        // 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.";
            }
        }
    }
```

8.23.10.5 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 values of form fields using the **BasicForm.GetDataValue(string fieldName)** method, which may be called through the corresponding property of the BizForm control. For example: *BizForm.BasicForm.GetDataValue("FirstName")*

Similarly, the **BasicForm.SetDataValue(string fieldName, object value)** method allows you to assign field values programatically. If necessary, the data of the entire form may be accessed as a data container through the **BizForm.BasicForm.Data** property.

You can also stop further processing (saving) of the form by setting the **BizForm.StopProcessing** property to *true*. 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.

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

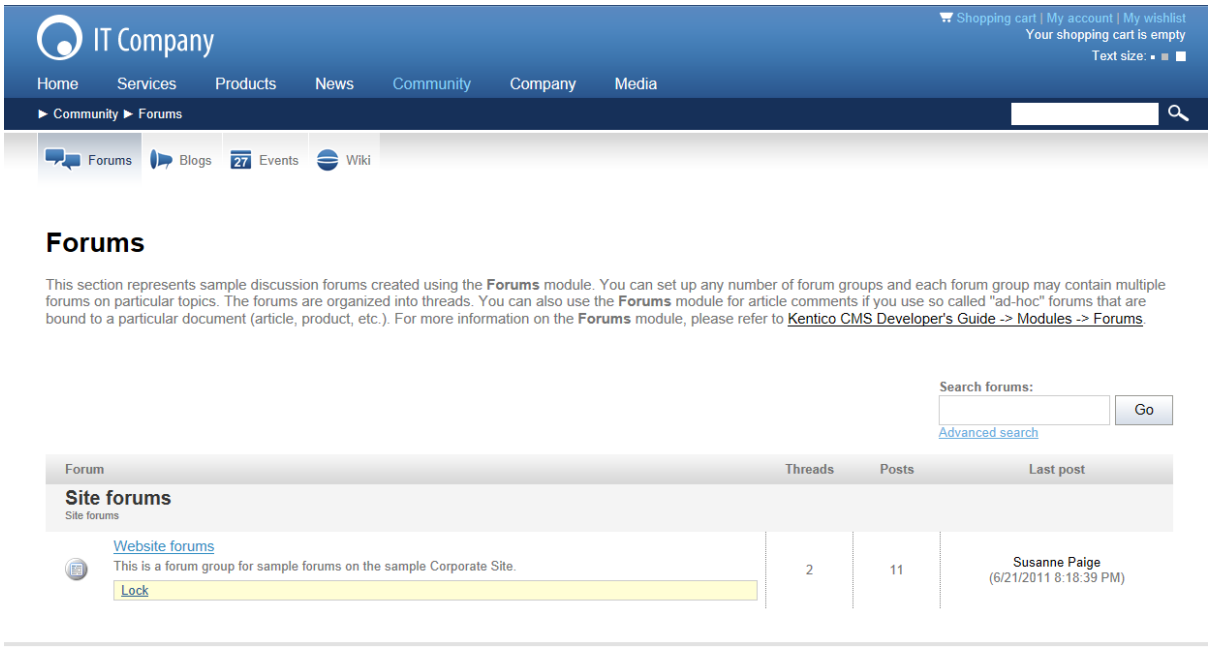
You can create your own field controls as described in chapter Form controls. If you want to make them available in the form field editor, you need to check the **Show control in Forms** box in the **Form control properties** dialog (in **Site Manager -> Development -> Form controls**) and choose a **Default data type**.

8.24 Forums

8.24.1 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.

8.24.2 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:

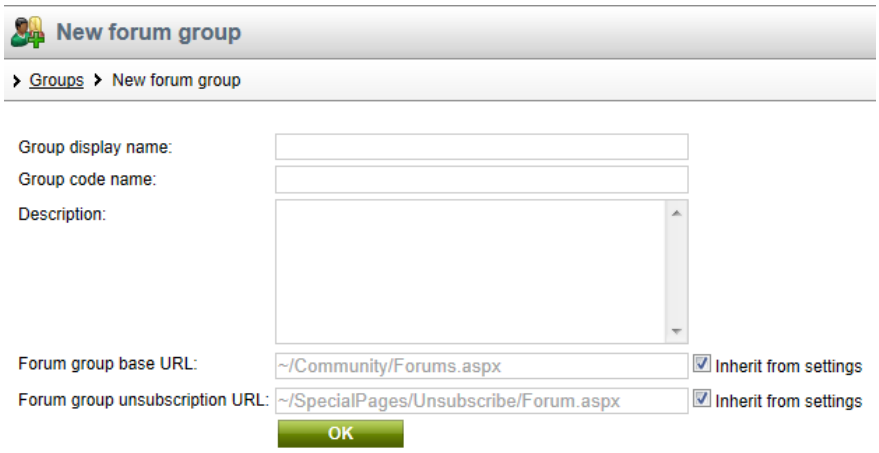
- Computers (forum group)
  - Announcements (forum)
  - Technical questions (forum)
  - FAQ's (forum)
- Web design (forum group)
  - CSS (forum)
  - XHTML (forum)

Creating a new forum group

Go to **CMS Desk -> Tools -> Forums** and click **New forum group**. Fill in the following fields:

- **Group display name:** the forum group name displayed on your website.
- **Group code name:** the forum group name used in your code.
- **Description:** description of the forum group displayed on your website.
- **Forum group base URL:** URL displayed when the user accesses the forum; *e.g. ~/MyForums.aspx*
- **Forum group unsubscription URL:** URL of a page where users can unsubscribe from the given forum group.

Click **OK** to save.



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..

8.24.3 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:

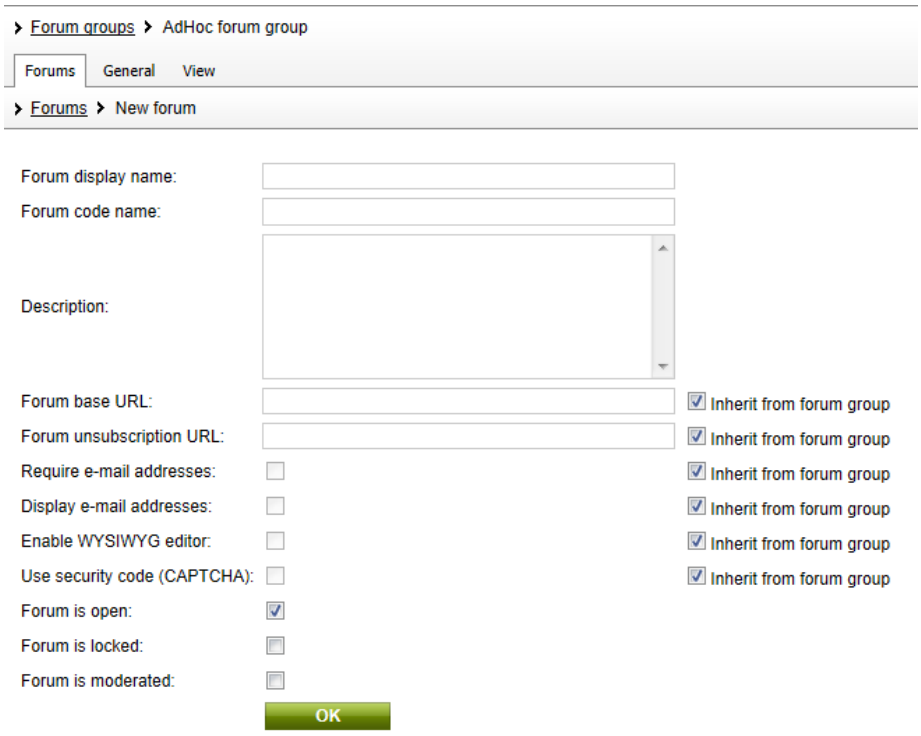
- **Forum display name** - the name of the forum that will be displayed on your website.
- **Forum code name** - the name of the forum used in your code.
- **Description** - the description displayed on your website.
- **Forum base URL** - URL displayed when the user accesses the forum; *e.g. ~/MyForums.aspx.*
- **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 displayed next to most 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.

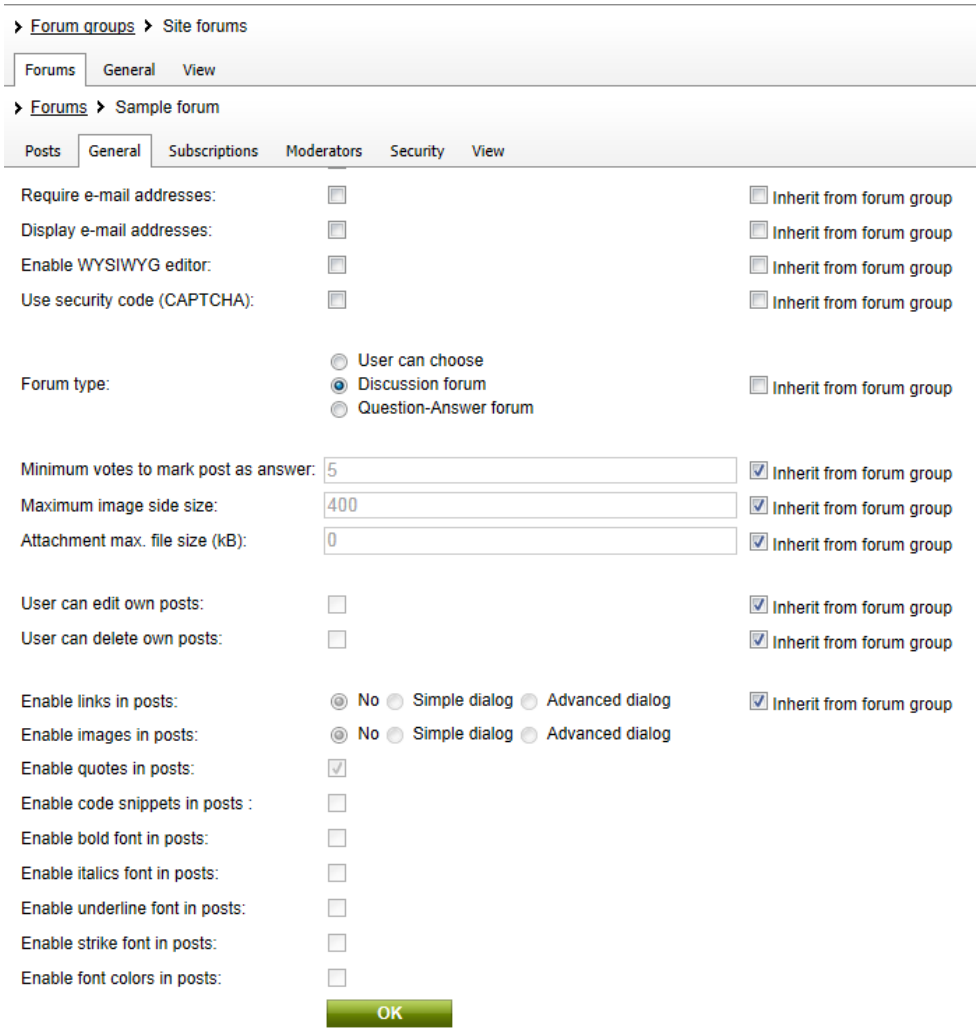
Click **OK** to add the forum.



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 at the bottom 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.

8.24.4 Publishing a pre-defined forum on the website

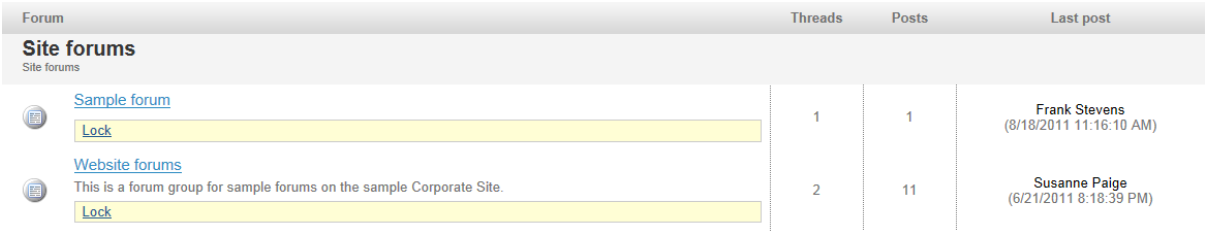
When you want to publish a forum on your website, you can use the built-in web parts on your page templates. You can find more details on each web part in the Kentico CMS Web Parts reference.

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.

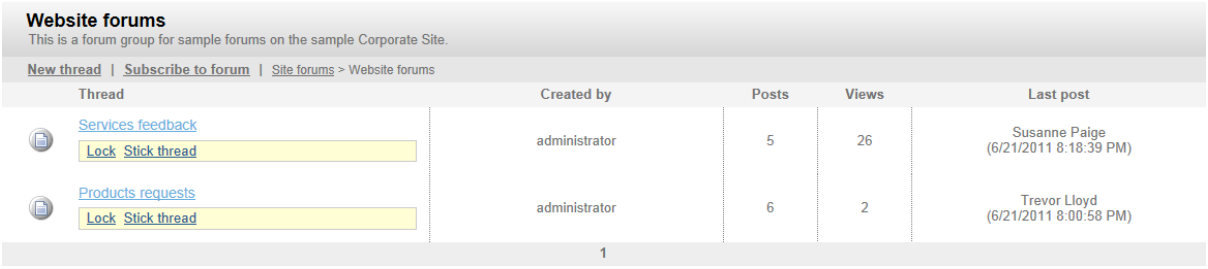
Publishing a forum group on the website

You can publish the whole 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:

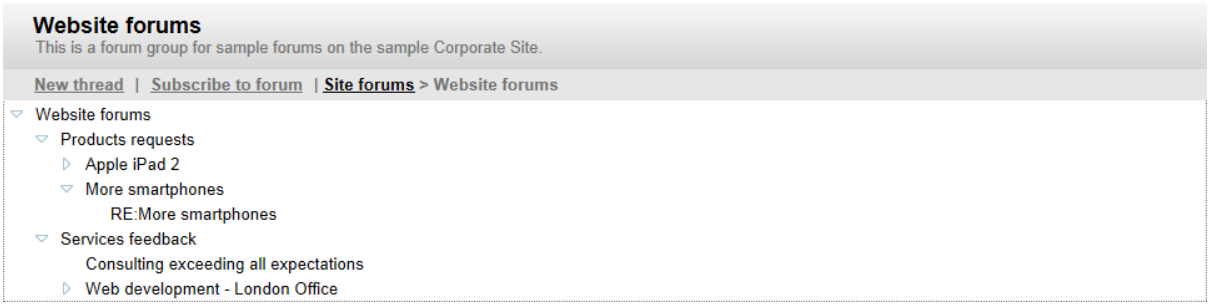


The forum threads in the selected forum can be displayed in two ways:

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**



8.24.5 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**.

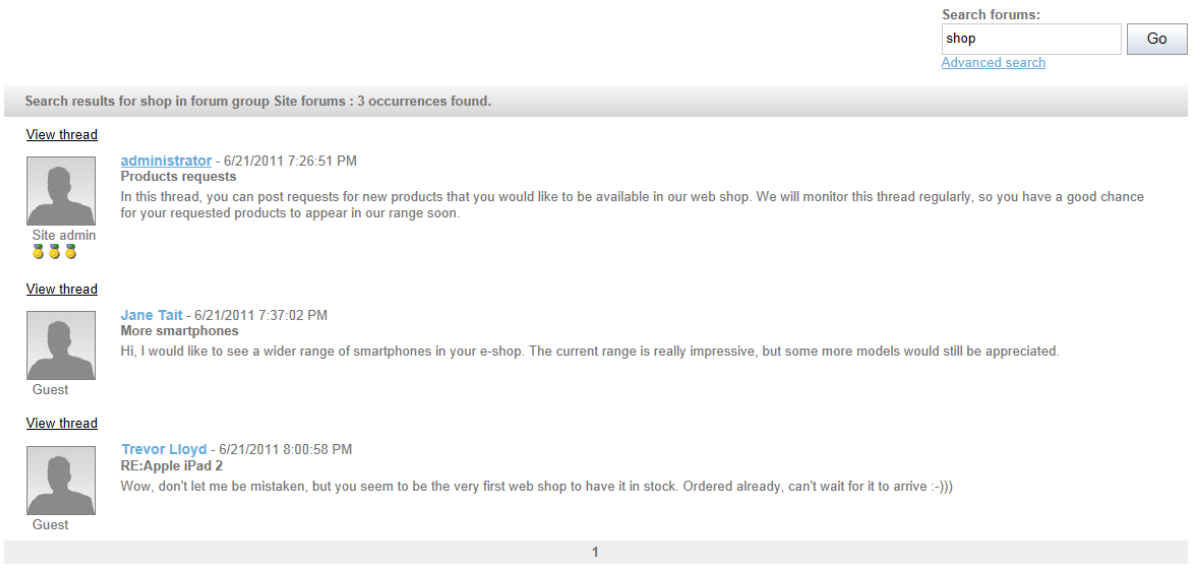
Ad-hoc forums are uniquely identified by the document they belong to.

The following two web parts are **deprecated**, and they are available only because of **backward compatibility** with versions prior to 4.0:

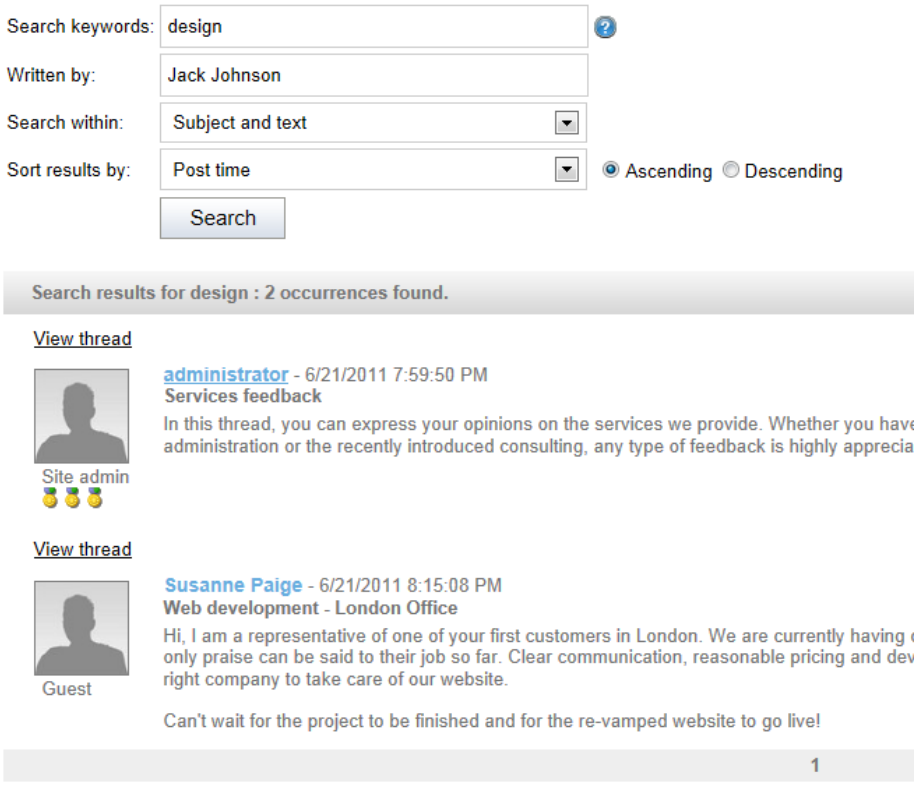
- **Forum (Single forum - Flat layout)** - this web part ensures backward compatibility with the former **Forum thread list** web part
- **Forum (Single forum - Tree layout)** - this web part ensures backward compatibility with the former **Forum tree** web part

8.24.6 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.

8.24.7 Managing forum posts

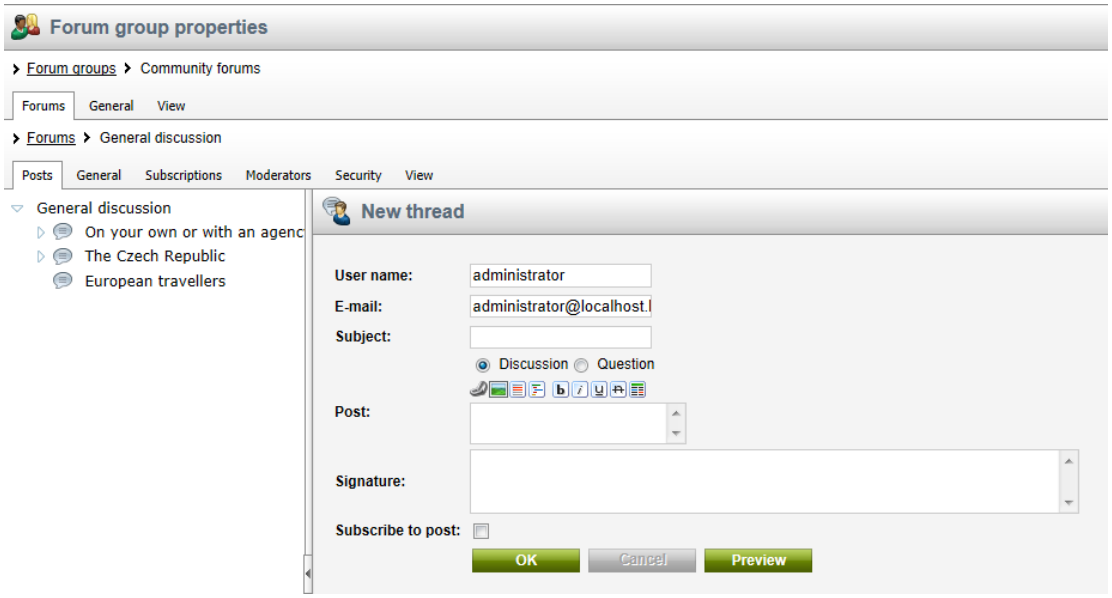
Threads and posts can be managed on the **Posts** tab when editing the forum that contains them.

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-

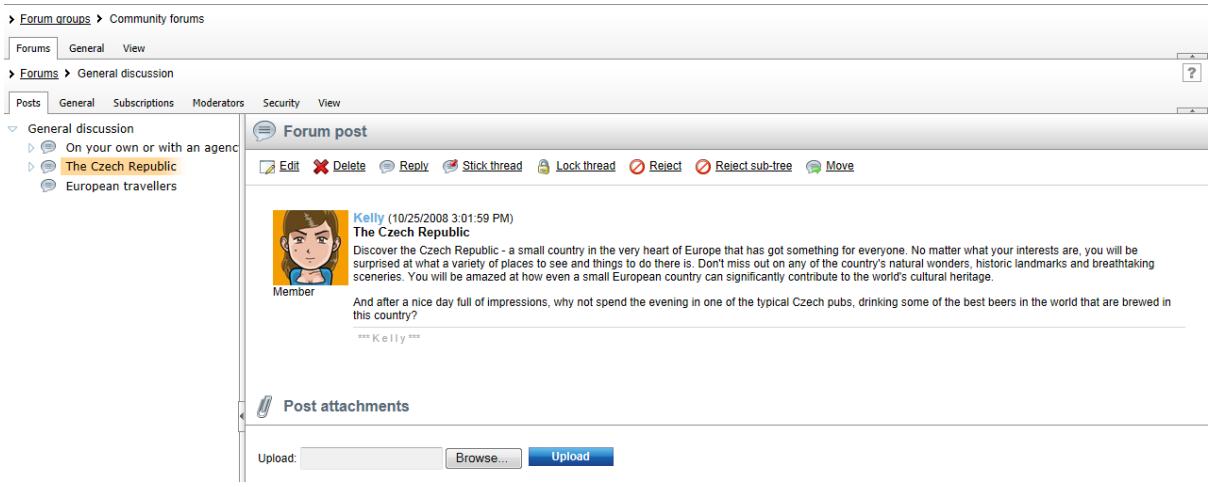
mail address specified in the *E-mail* field



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

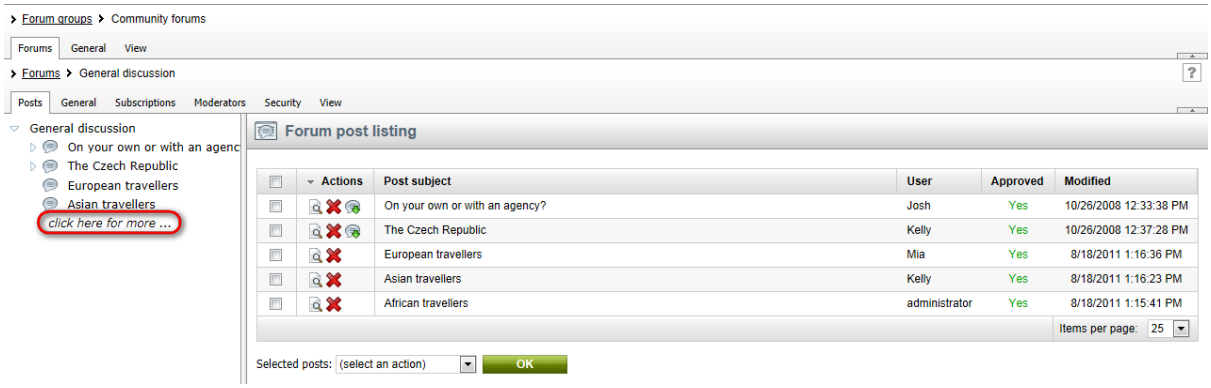


Forum post listing

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)

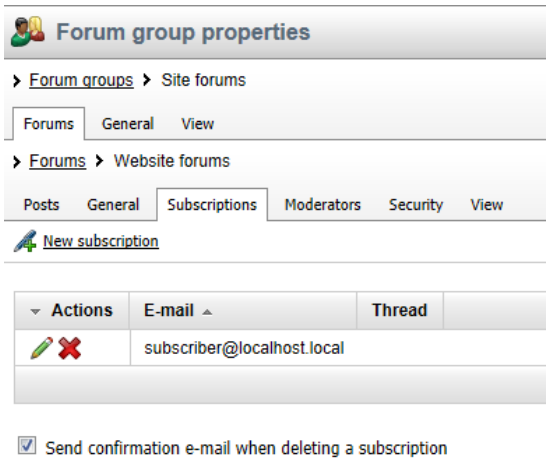


8.24.8 Subscriptions

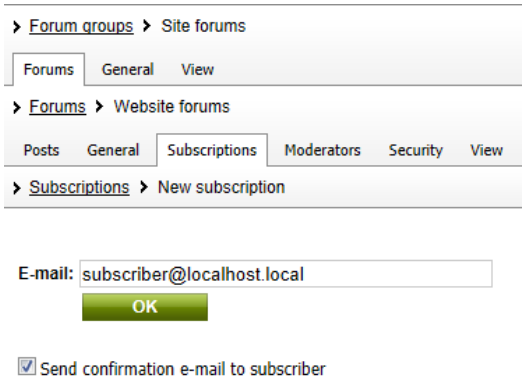
Subscriptions allow users to receive e-mail notifications when a new post is added to the selected forum.

There are several ways how a user can subscribe for receiving e-mail notifications:

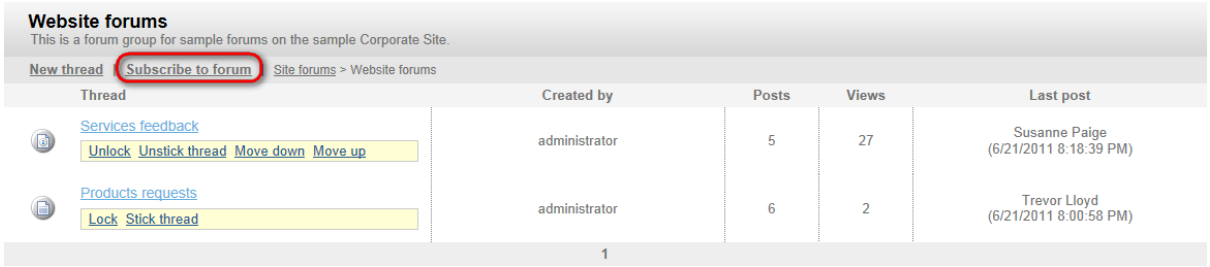
1. The forum administrator can subscribe users to the whole forum and manage all subscriptions at **Forum edit -> Subscriptions**.



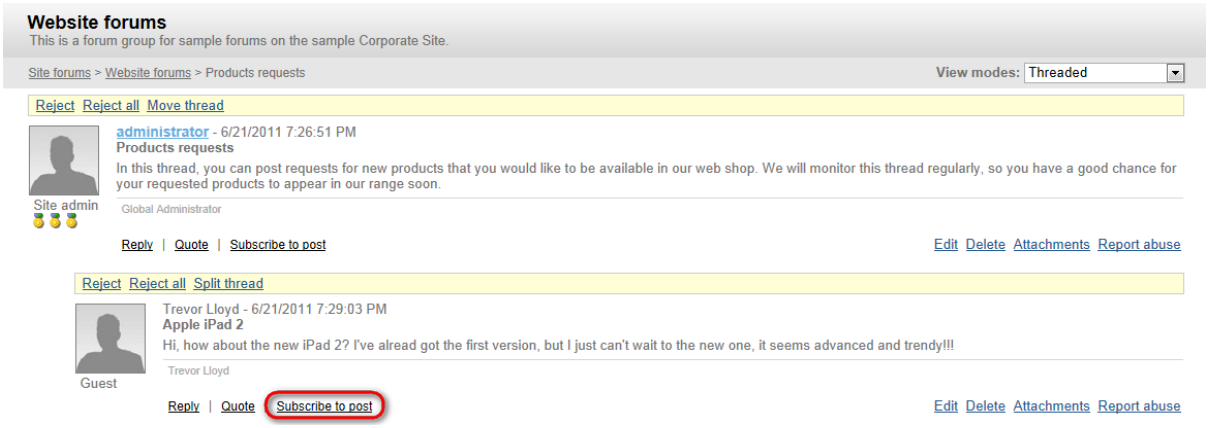
After clicking the **New subscription** link, the following dialog depicted in the screenshot below will be displayed. You only need to enter the user's e-mail address and click **OK**. If you do it with the **Send confirmation e-mail to subscriber** check-box checked, the user will receive a confirmation e-mail, telling them that they have been subscribed. These confirmation e-mails are sent automatically when the user subscribes on-site, as described in the following two points.



2. The user clicks the **subscribe to forum** link at the top of the forum:



3. The user subscribes to a particular post and sub-posts:



Users can unsubscribe at any time by using the link in the e-mail.

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 the **Site Manager -> Administration -> E-mail templates** dialog. The notifications informing users about their forum subscriptions or unsubscriptions are based on the **Forums - Subscription confirmation** and **Forums - Unsubscription confirmation** templates respectively.

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 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.

Forum base URL and forum e-mails

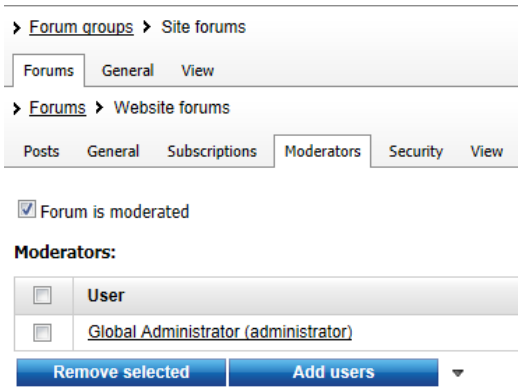
It's important to set a correct **Forum base URL value** in the forum properties so that the notification e-mails contain valid links to the forum. Use URL in format http://localhost/kenticocms/forums.aspx **- enter the full URL of the page with the forum, including the domain.**

8.24.9 Forum moderation

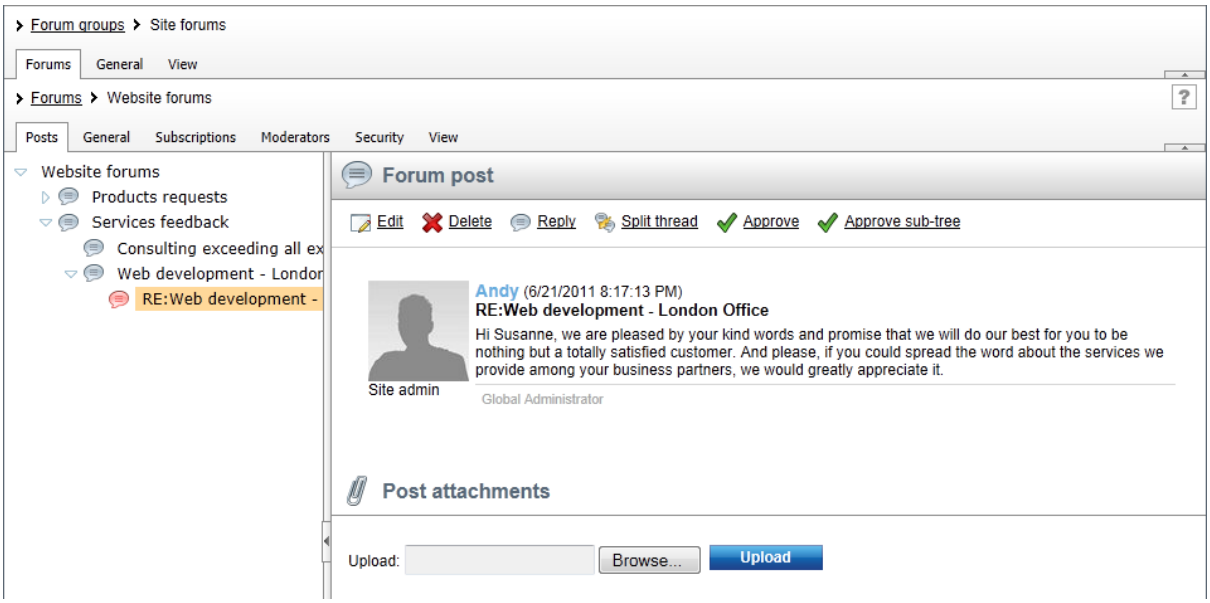
You can configure a forum so that it is moderated. It means that all posts must be approved by one of the forum moderators before they are published in the forum.

Configuring forum moderation

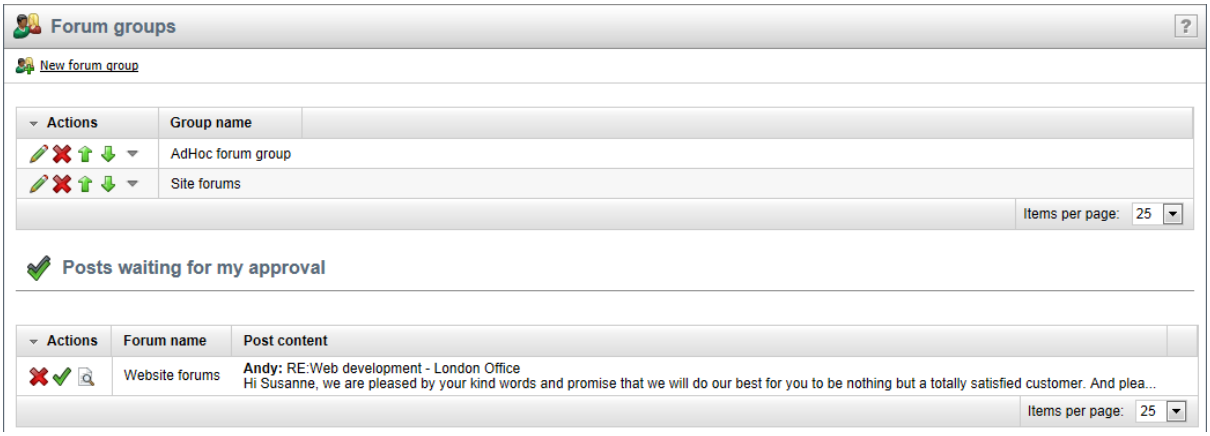
You can enable forum moderation and configure the list of moderators under the **Forum properties -> Moderators** tab. Even when this option is disabled, approval of posts might be required, e.g. when a Bad word is detected or when a post has been rejected.



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.

8.24.10 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:

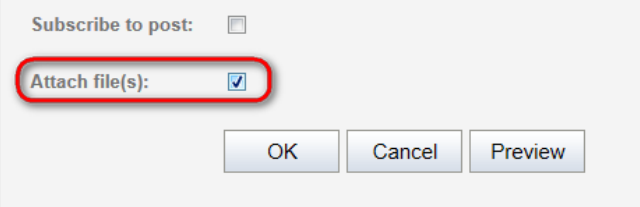
- **Display attachment image** - if checked, attached images will be displayed as images, not only as links.
- **Attachment image maximal side size** - if an attached image is larger than the entered value, it will be resized so that its larger side's size is equal to the entered value.

When editing a forum in **CMS Desk -> Tools -> Forums**, you can set the following property on the **General** tab:

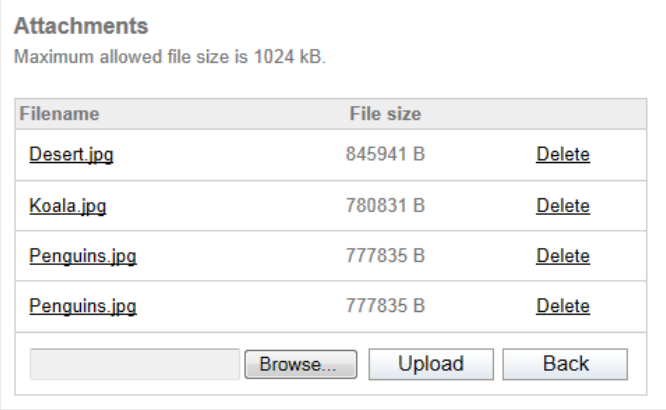
- **Attachment max. file size (kB)** - you can define the maximum size of an attached file in kB.

Adding attachments to a forum post

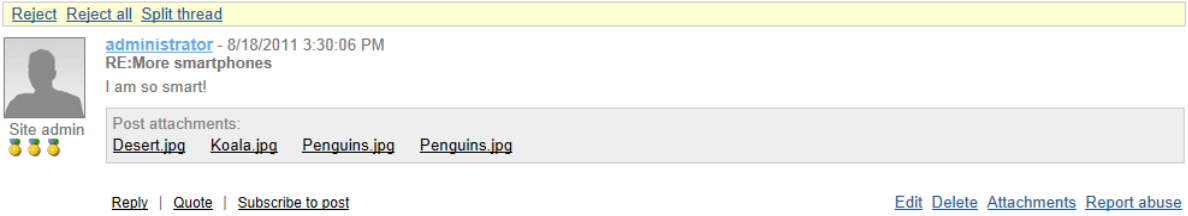
When adding a post to a forum, users can check the **Attach file(s)** check box at the bottom of the post adding dialog.



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.



8.24.11 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 text selected in the 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 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	Enable links in posts
	[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	Enable image macros in posts

		Advanced dialog - clicking the icon displays the Insert image dialog where the image can be selected in various ways	
	[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

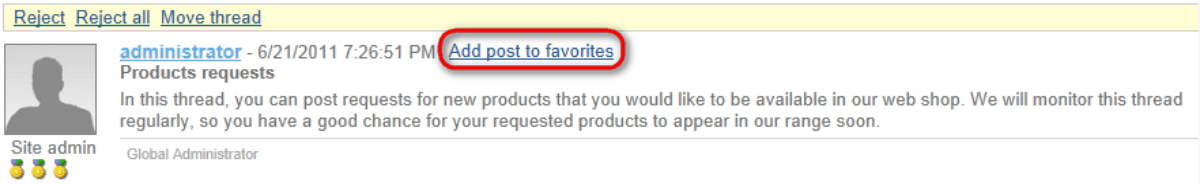
There is an easy way to resolve macros in ASPX or codebehind code. To resolve all the macros (recommended), use the following method:

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.

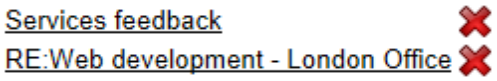
8.24.12 Forum favorites

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.



8.24.13 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}**

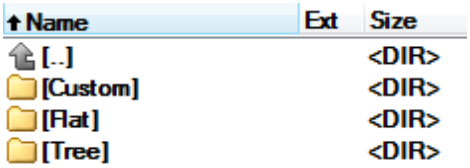
Single forum friendly URLs

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}**

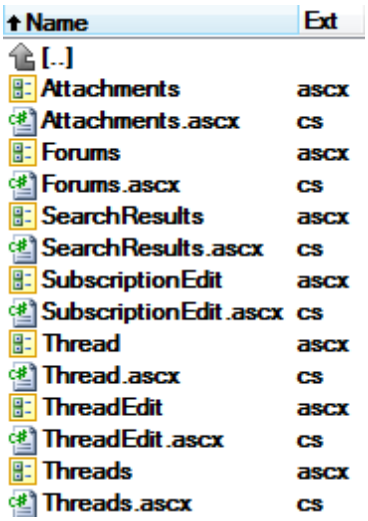
8.24.14 Customizing forum design

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**.



8.24.15 Security

The security model of the Forums module has two parts:

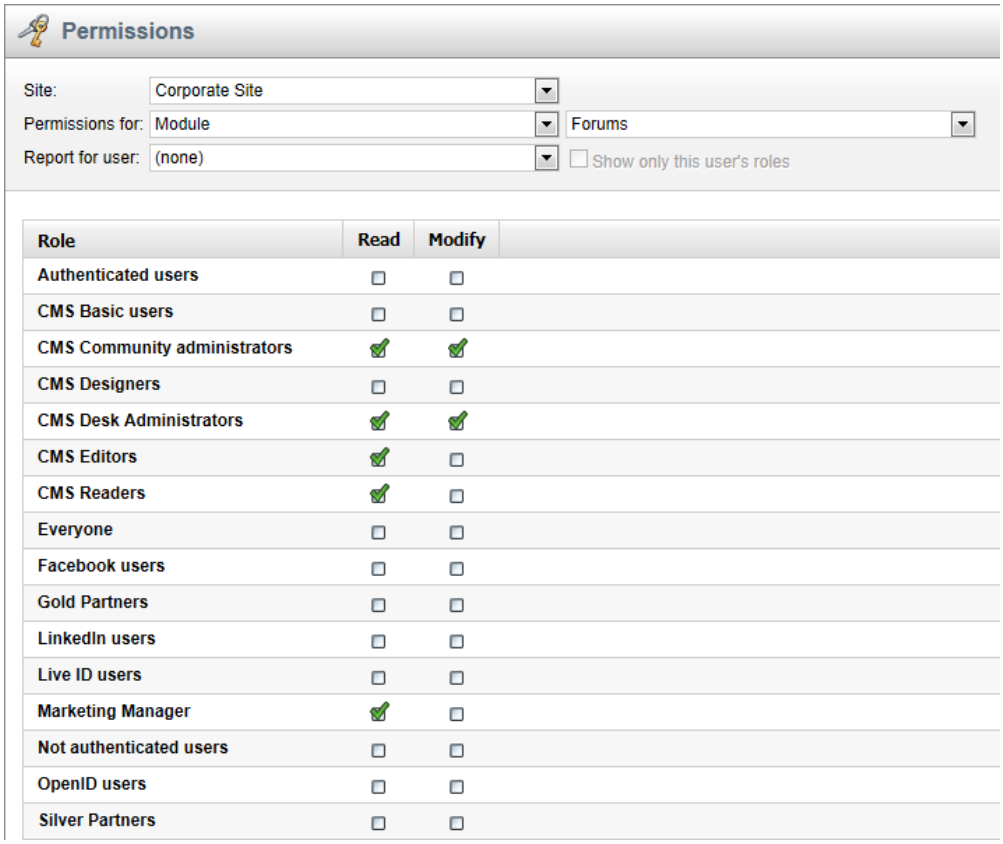
1. Security of the Forums module administration interface.
2. Security of the forums published on the website.

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:

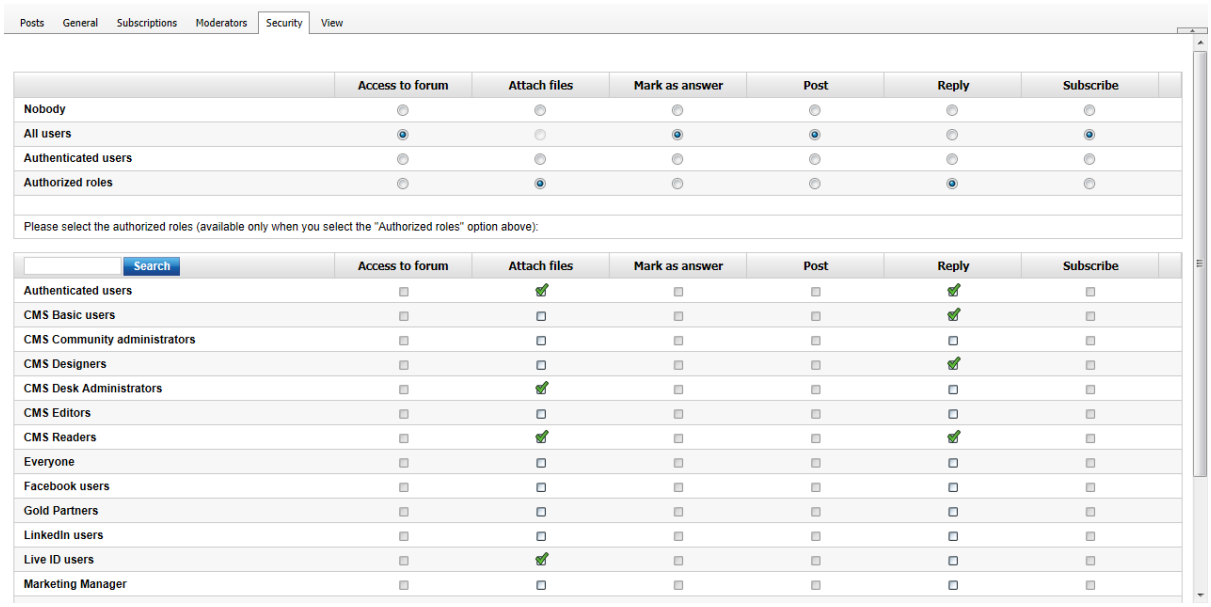
- **Access to forum** - defines who can enter the forum and view posts
- **Attach files** - defines who can attach files to forum posts
- **Mark as answer** - defines who can mark posts as answers in Question - Answer forums
- **Post** - defines who can add posts to the forum
- **Reply** - defines who 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



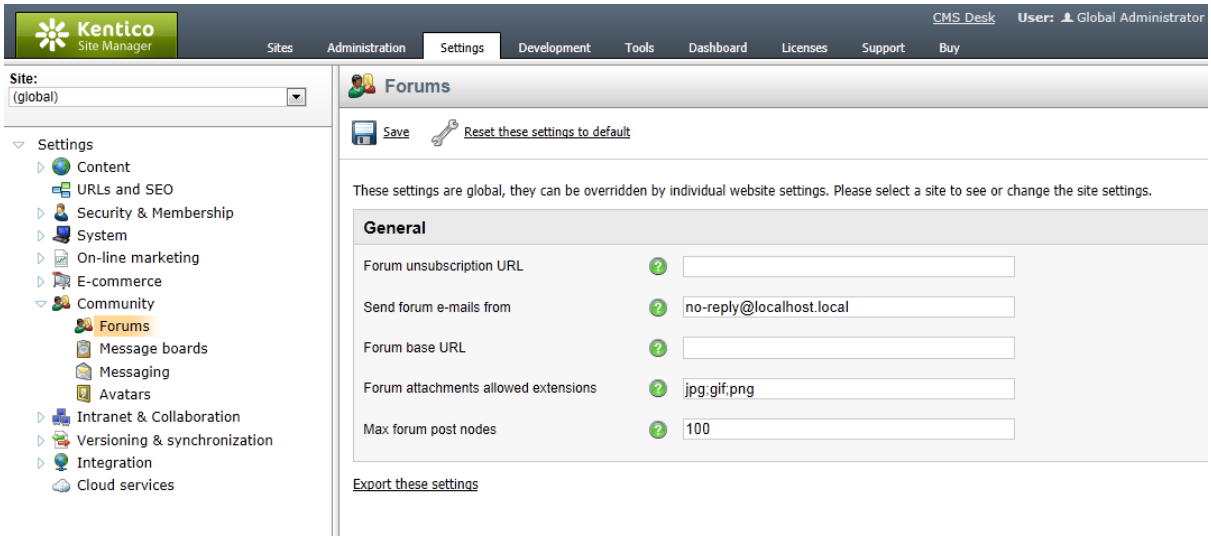
The following properties of the **Forum group** web part are also related to forum security:

- **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

8.24.16 Settings

Settings of the Forums module are located in **Site Manager -> Settings -> Community-> Forums**. The following settings are available:

- **Forum unsubscription URL** - URL that will be used for unsubscriptions from receiving notifications about new forum posts. Can be inherited by forum groups and further by particular forums. The **Forum unsubscription** web part must be placed on the specified page for the unsubscriptions to work.
- **Send forum e-mails from** - e-mail address that will be used as the sender address of forum notification e-mails.
- **Forum base URL** - URL that will be displayed when viewing a forum and in forum notification e-mails, e.g. *~/Forum.aspx*
- **Forum attachments allowed extensions** - you can specify extensions of files that can be attached to forum posts. Extensions should be entered without dots and separated by semicolons. If blank, all extensions are allowed.
- **Max forum post nodes** - maximum number of forum post nodes displayed in the forum post tree when editing a forum in CMS Desk -> Tools -> Forums. When more nodes are in the forum group than 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.



8.24.17 Forums internals and API

8.24.17.1 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.

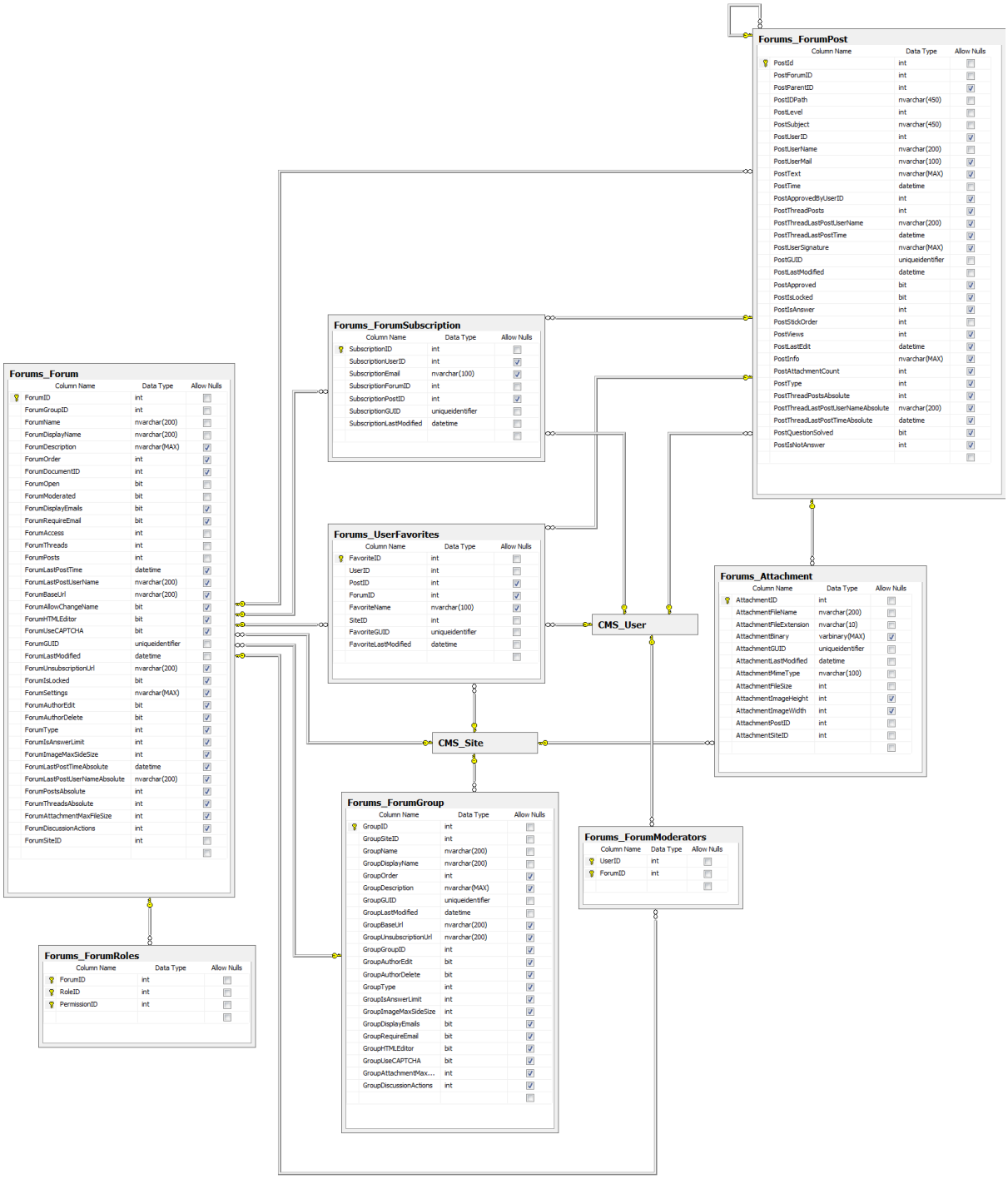
Further API-related information and examples can be found in the API programming and Kentico CMS internals section of this guide. For detailed API documentation, such as a list of all members of the module's classes, please refer to Kentico CMS API Reference.

8.24.17.2 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.



8.24.17.3 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.

8.24.17.4 API examples

8.24.17.4.1 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;
```

8.24.17.4.2 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);
}
```

8.24.17.4.3 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);
}
```

8.24.17.4.4 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;
}
```

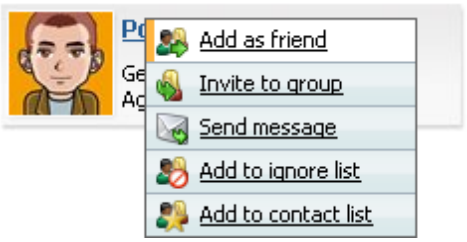
The following example deletes a forum post.

```
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;
}
```

8.25 Friends

8.25.1 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.

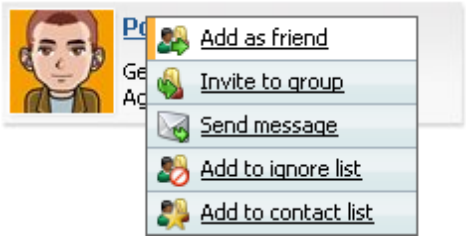
8.25.2 Requesting friendships

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.

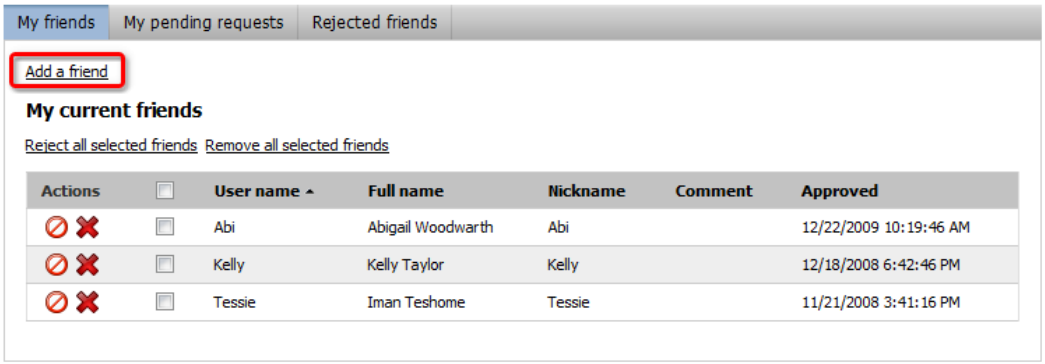


Request friendship and My friends web parts

The **Request friendship** web part is just a tiny link with customizable text. Clicking it opens the **Add a new friend** pop-up dialog.

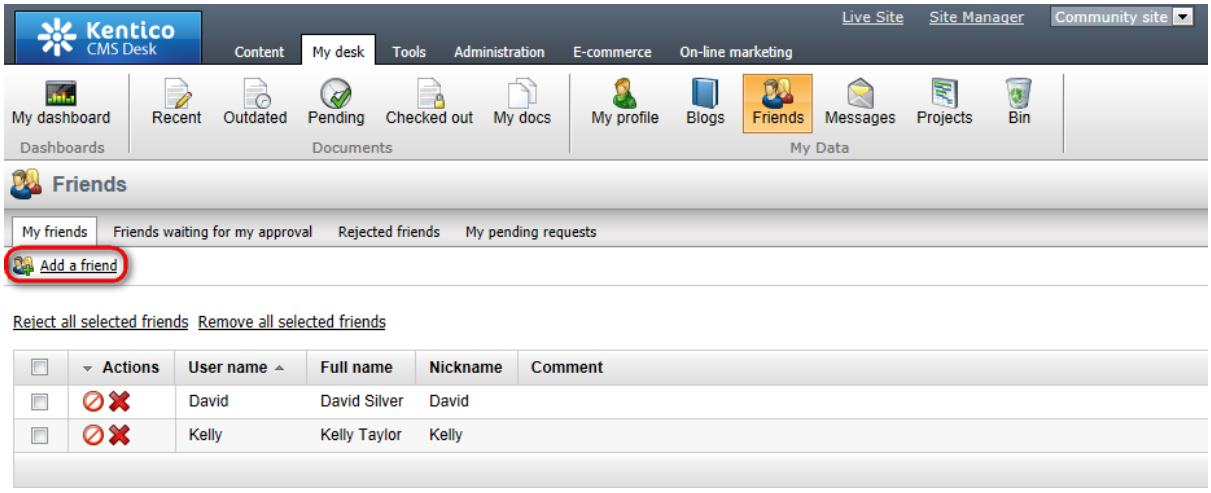
Request friendship

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** link, which opens the **Add a new friend** dialog.

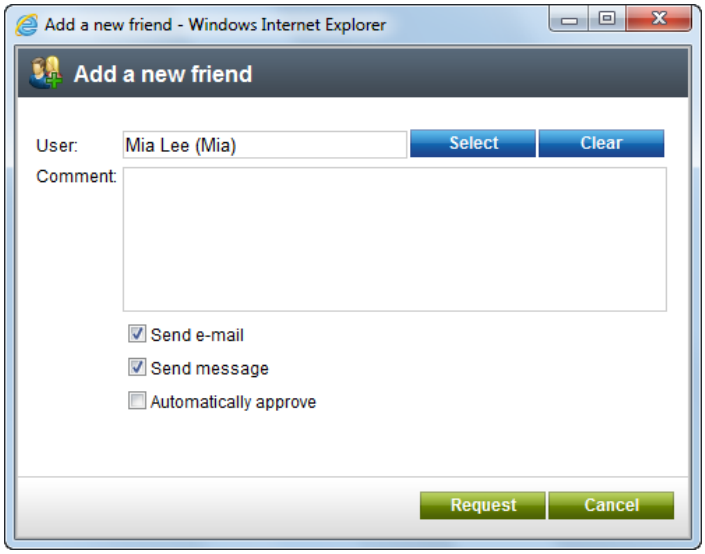


Add a new friend dialog

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.



8.25.3 Related e-mail templates

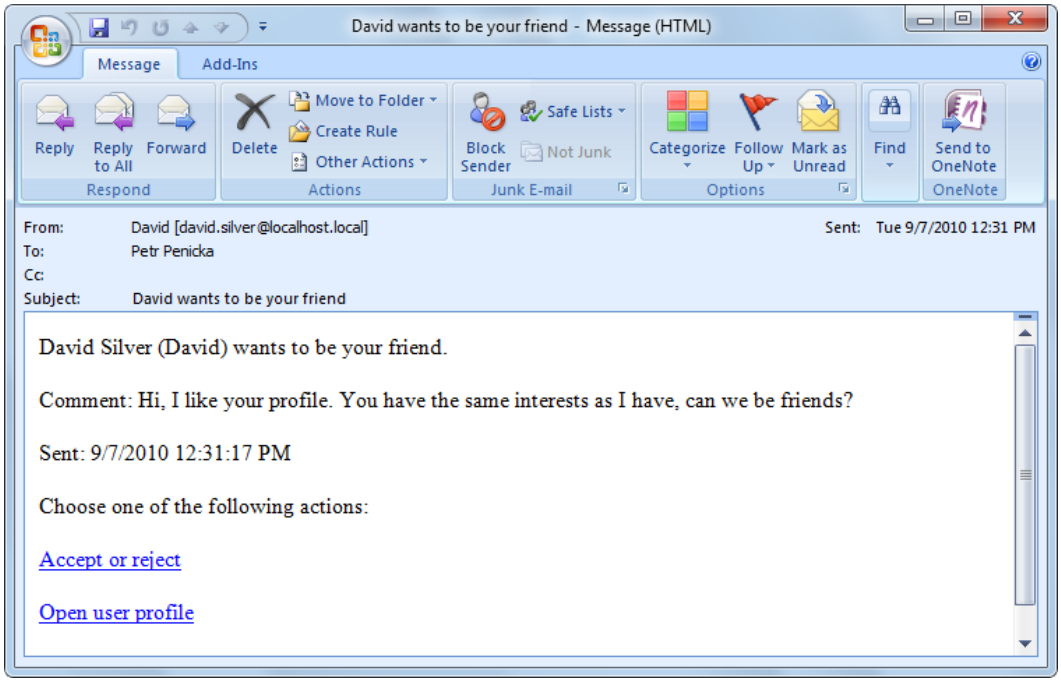
In **Site manager -> Administration -> E-mail templates**, you can find the following e-mail templates related to the Friends module:

- **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.

8.25.4 Friends management

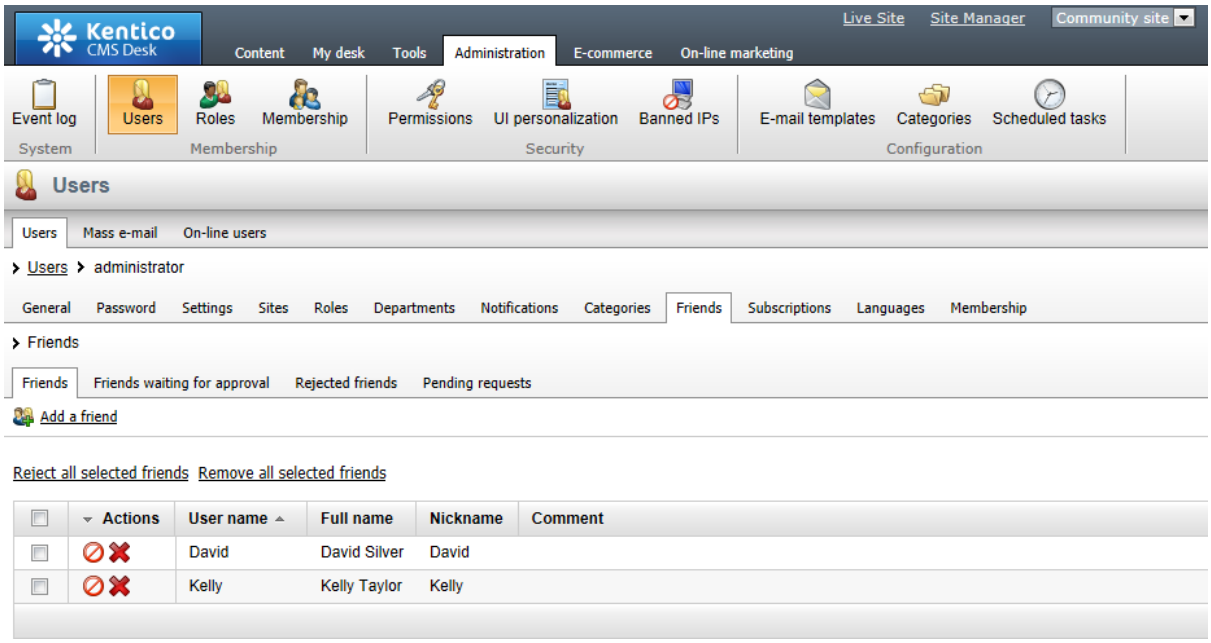
Users with access to **CMS Desk -> My desk -> Friends** can manage their own friendships from within this section of the administration interface.

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.

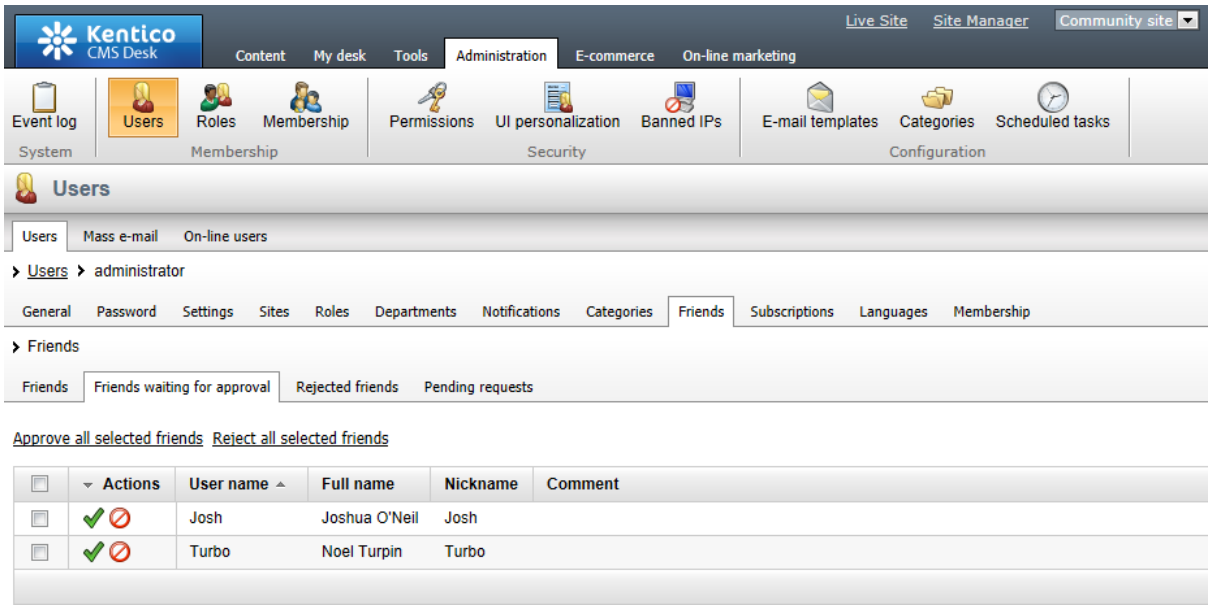
Friends (My friends)

On this tab, you can see a list of all your current friends. New friendships can be requested using the **Add a friend** link. 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.



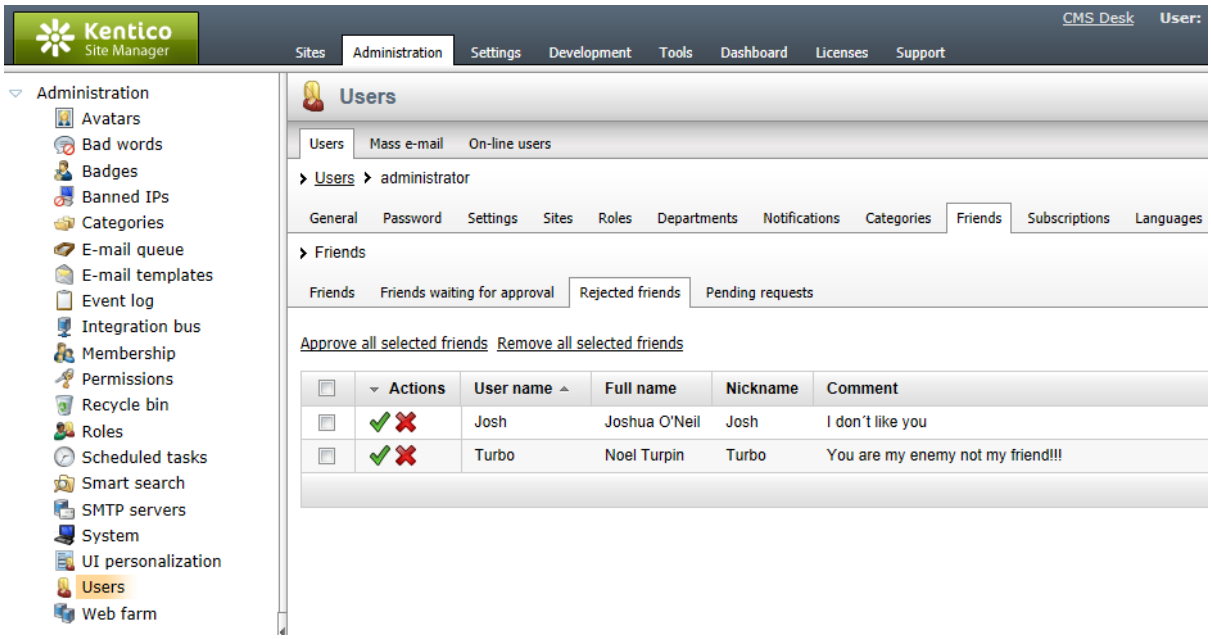
Friends waiting for approval (Friends waiting for my approval)

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

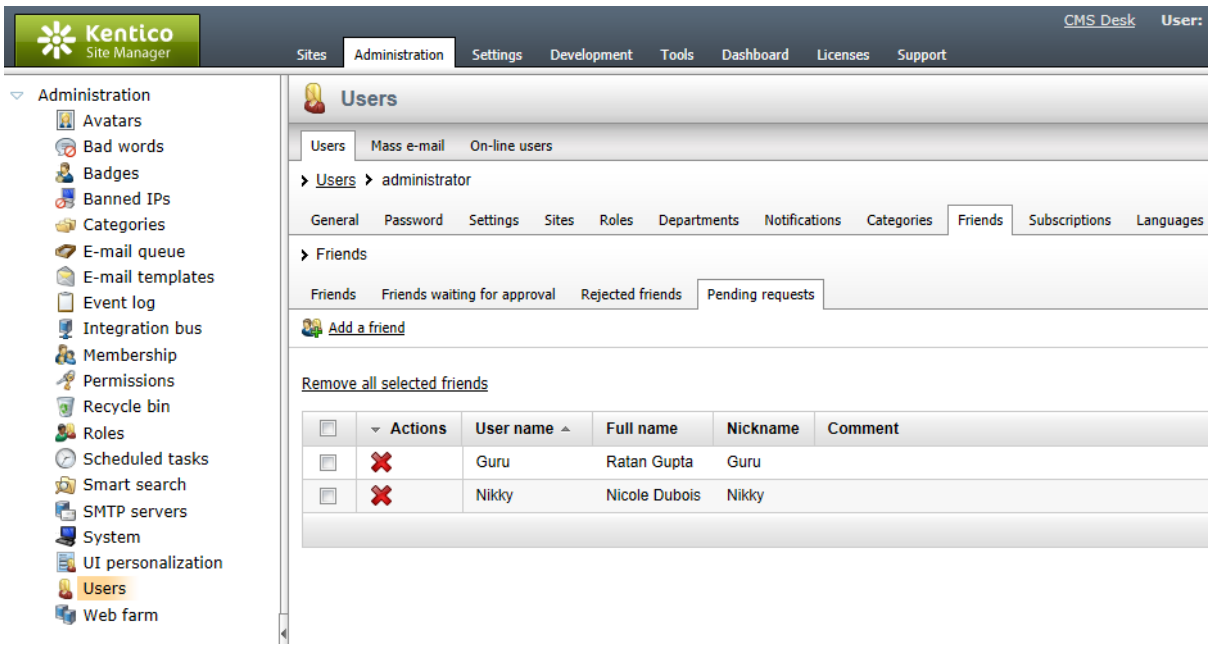
On this tab, you can see a list of all users whose friendship you rejected. Once in the rejected status, the user can't request your friendship anymore. You can either **Approve** (✔) some user's friendship, which makes them your friend, or **Remove** (✖) the user from the list of rejected, which enables 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 the **Add a friend** link.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.

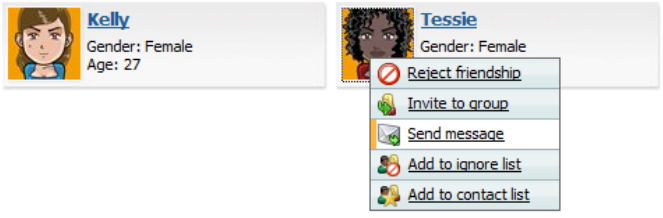


8.25.5 Available web parts

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

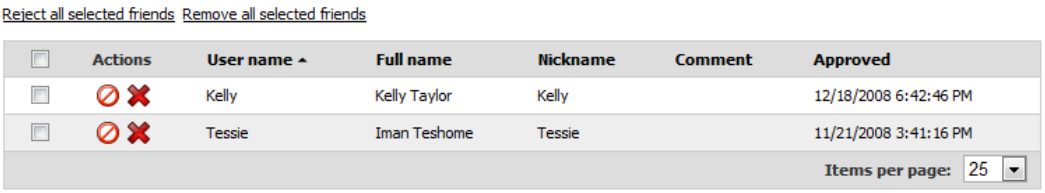
This web part can typically be used on user's profile page for viewing their friends. If you right-click the user's avatar image, you will be offered several tasks, as you can see in the screenshot below.



The following web parts are actually on-site equivalents of the friends management administration interface described in the Friends management topic:

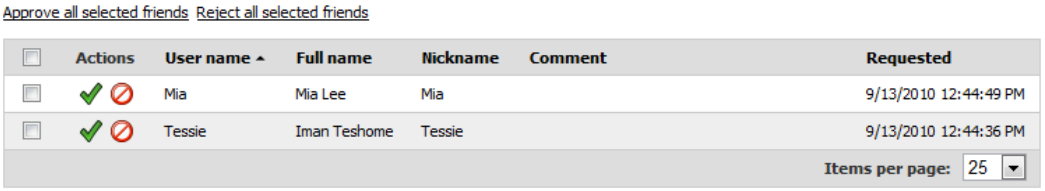
Friends list

This web part displays a list of all friends of the current user.



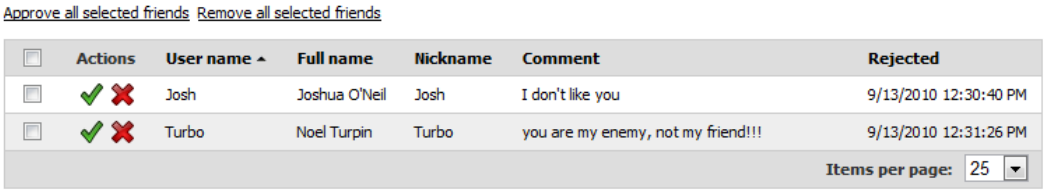
Friends waiting for approval

This web part displays a list of all users waiting for the current user's friendship approval.



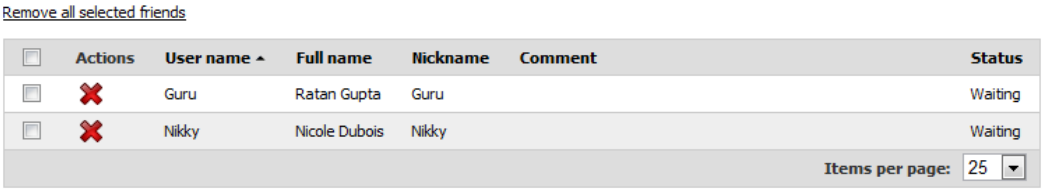
Rejected friends

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.

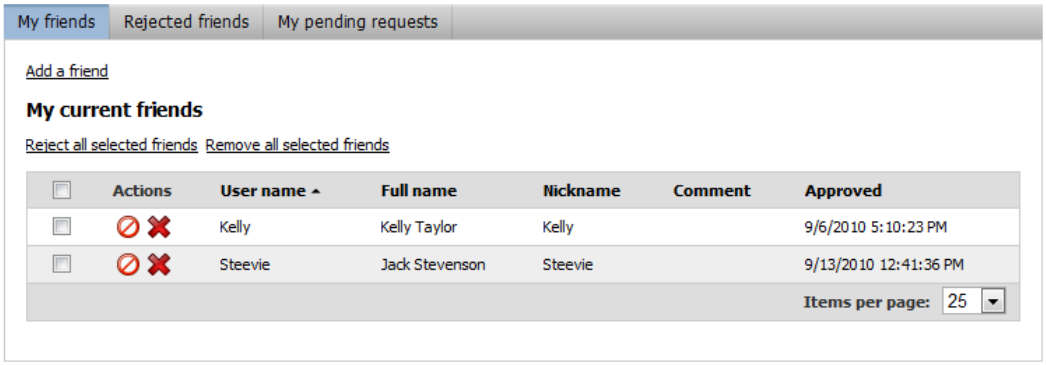


Request friendship

This web part displays only a link that, when clicked, opens the **Add a new friend** dialog.

My friends

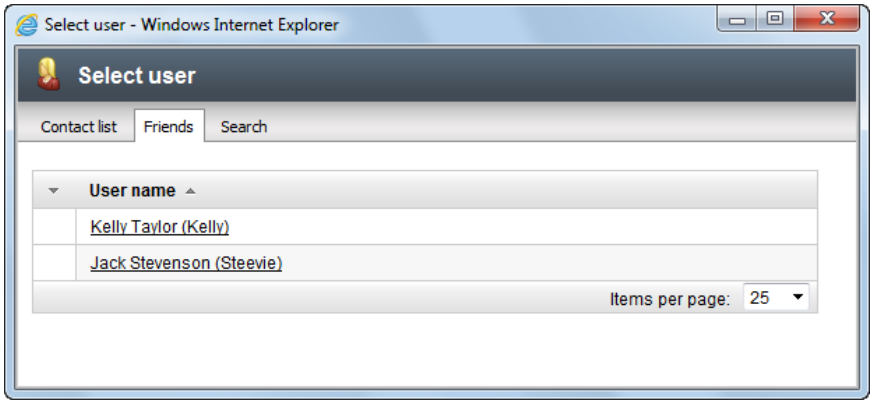
This web part combines the five previously mentioned web parts in one single web part. It's functionality is fully equal to the UI in **CMS Desk -> My Desk -> Friends**.



8.25.6 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

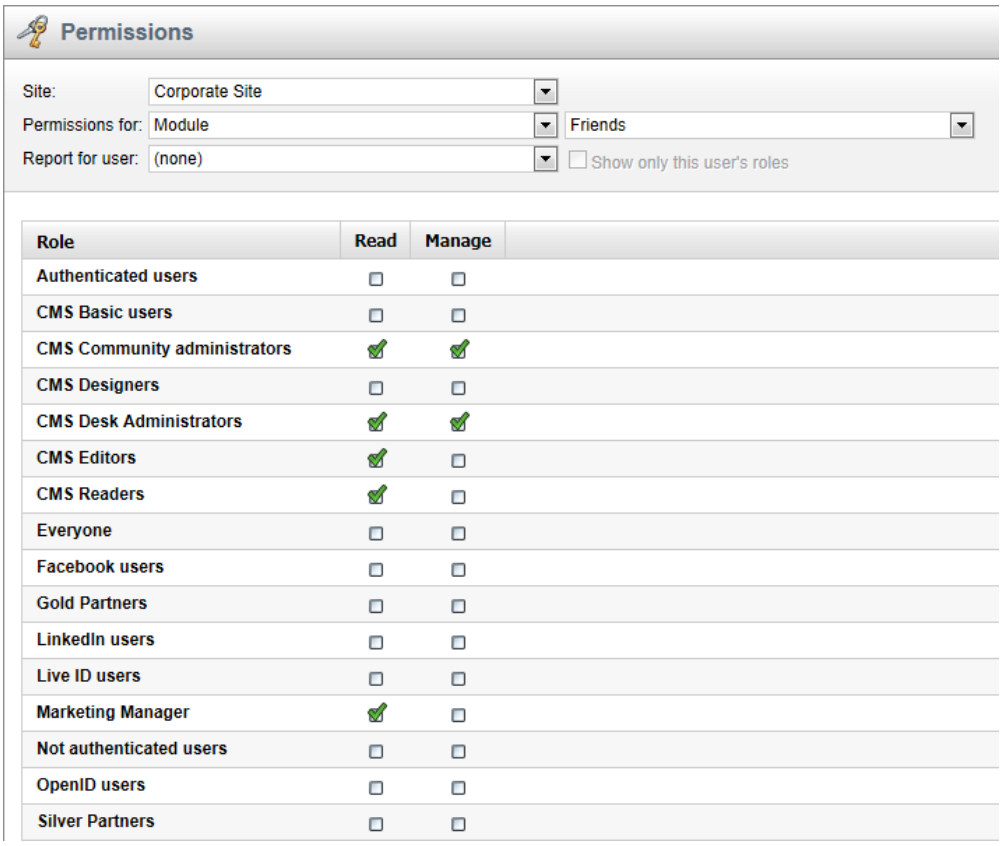
Users can set the visibility of some fields on their public profile to **Display to friends**, which makes the field visible only to their friends. For more information on this feature, please refer to Development -> Membership, security and permissions -> Custom fields visibility.

8.25.7 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:

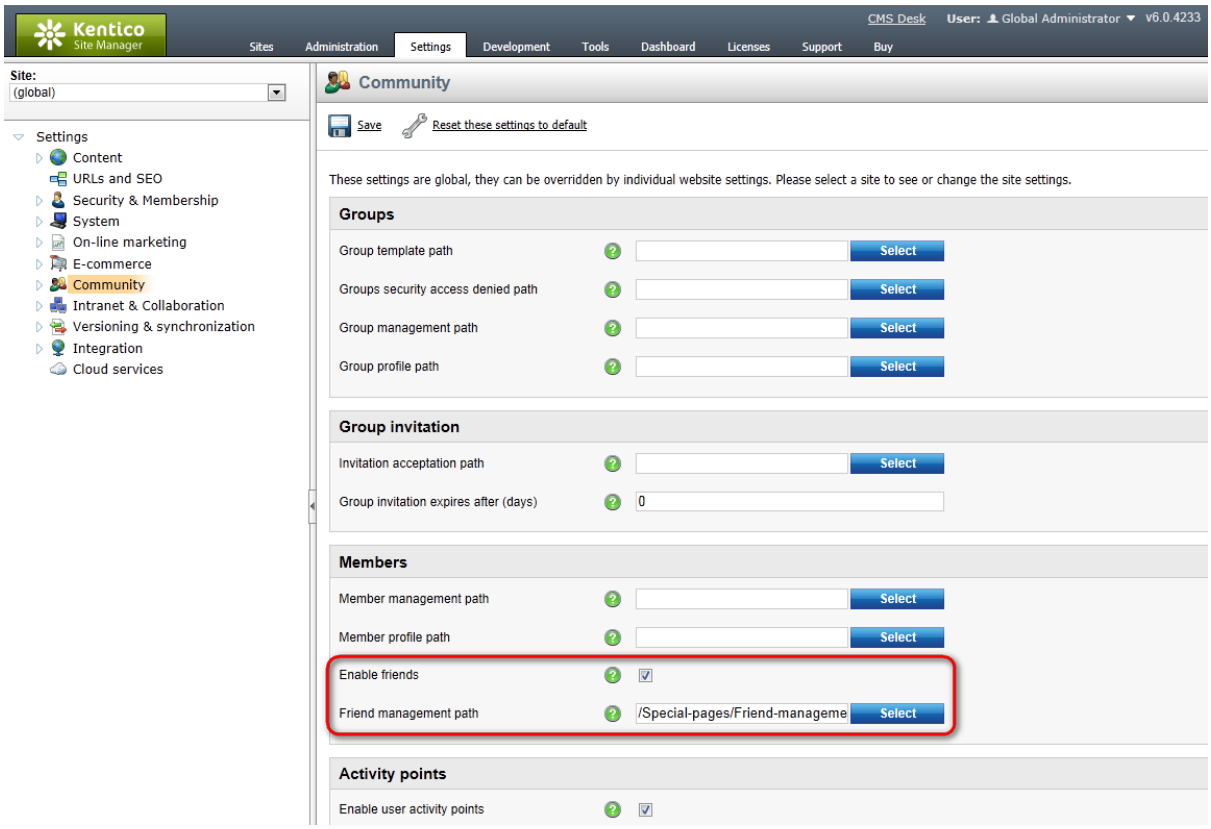
- **Manage** - members of the role can manage Friends of particular users in *CMS Desk -> Administration -> Users*
- **Read** - members of the role can view friends of particular users in *CMS Desk -> Administration -> Users*



8.25.8 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.



8.25.9 Friends internals and API

8.25.9.1 Overview

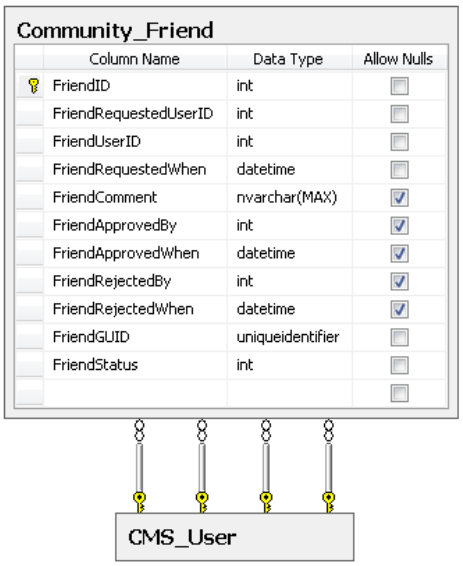
In this chapter, you will learn which database tables and API classes are used in the Abuse report module. You will also see the most common API examples.

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.

8.25.9.2 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.



8.25.9.3 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.

8.25.9.4 API examples

8.25.9.4.1 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;
```

8.25.9.4.2 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()
{
```

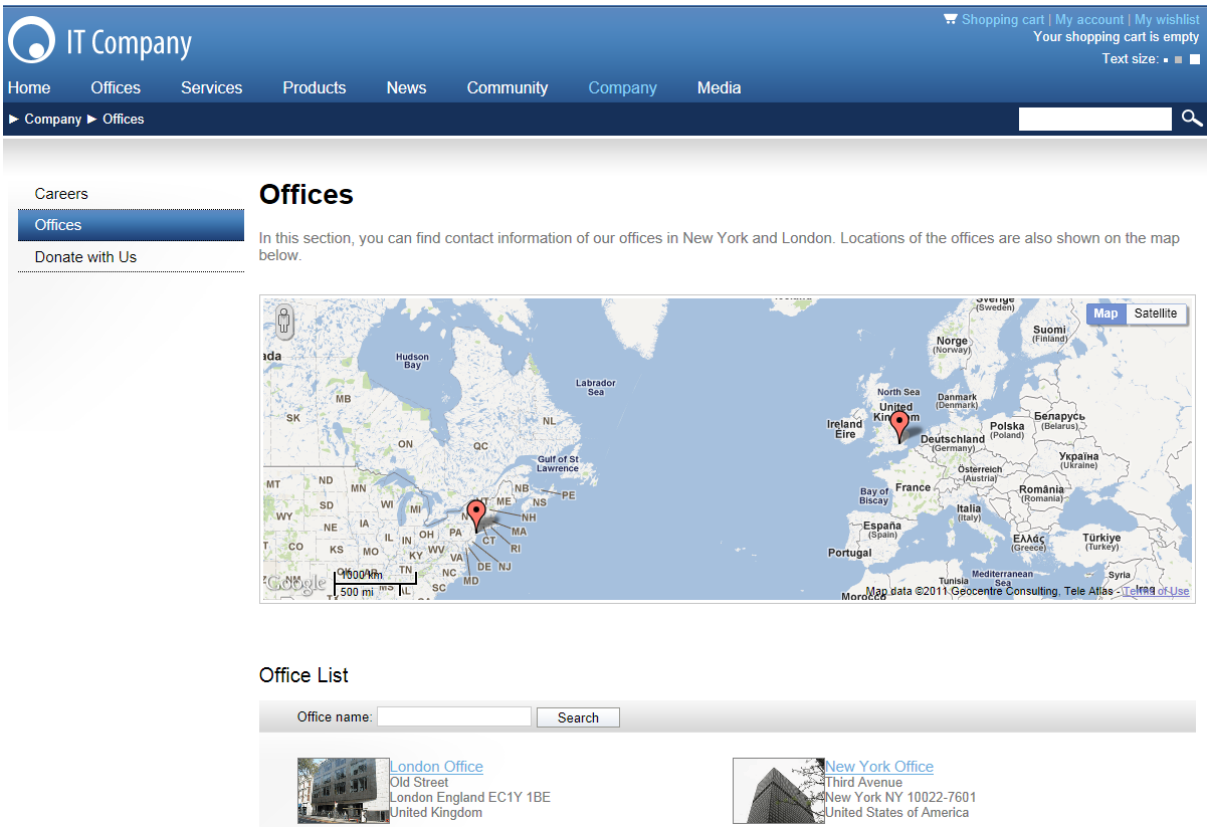
```
    // Get the user
    UserInfo friend = UserInfoProvider.GetUserInfo("MyNewFriend");

    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;
}
```

8.26 Geo mapping

8.26.1 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.

8.26.2 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/.

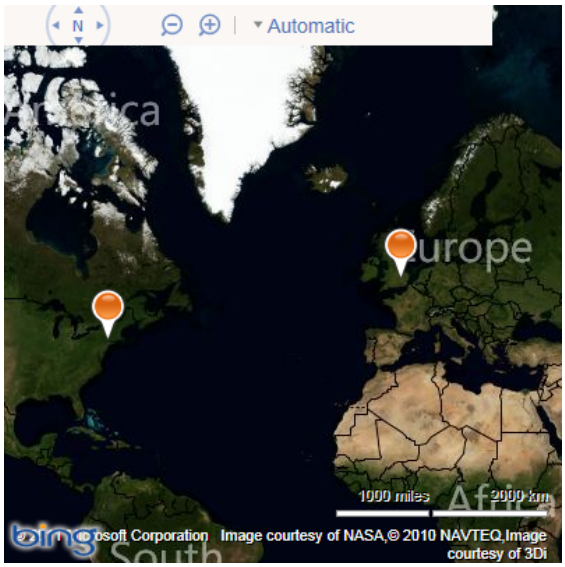
The maps show the location markers and are zoomed according to the settings made by the **Large view scale** property. For a document-related map and for a map with a data source, the center is on the coordinates specified by the **Default latitude** and **Default longitude** properties of the corresponding 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

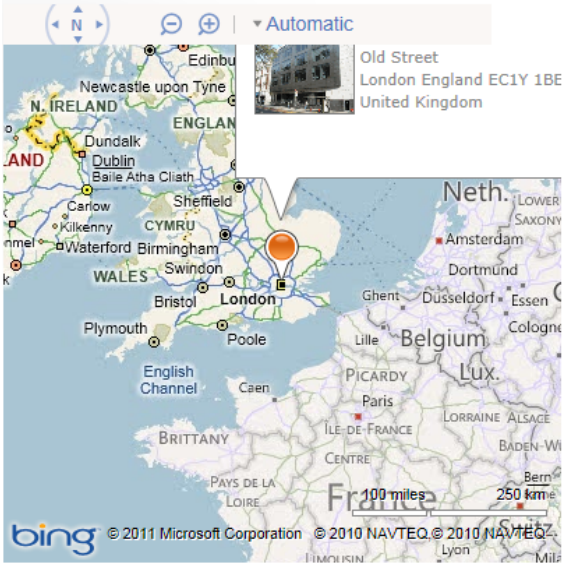
The **Bing maps** web part is used. Geographical location is displayed dynamically based on the properties of the document. More locations can be displayed at a time. Learn here how to display location markers on the map.
In the screenshot below, you can see how the web part looks on page load.



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.

8.26.3 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.

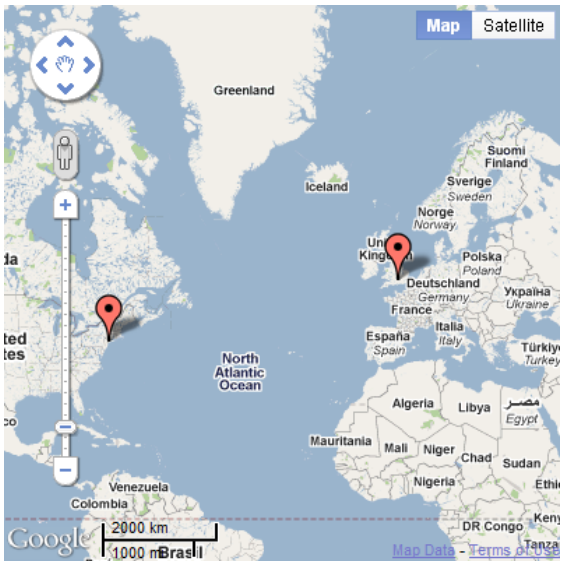
The maps show the location markers and are zoomed according to the settings made by the **Large view scale** property. For a document-related map and for a map with a data source, the center is on the coordinates specified by the **Default latitude** and **Default longitude** properties of the corresponding 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.

8.26.4 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

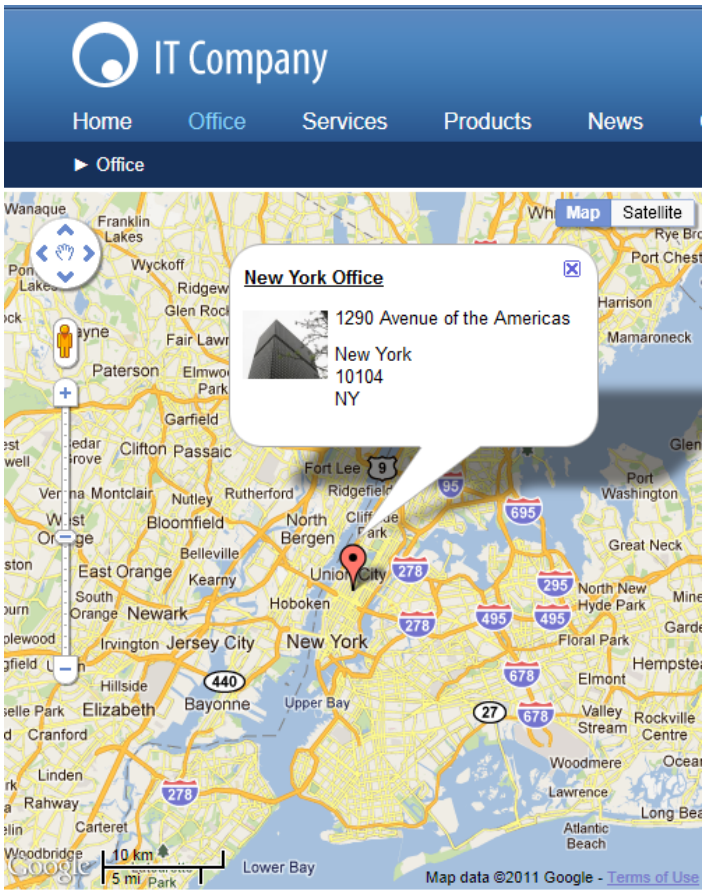
You can enable geolocation translations, i.e. addresses to coordinates, on the server side by checking the **Use server processing** checkbox. If you decide to do so, please check that the maps services query limits (IP-based) are sufficient for your needs.

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:



8.26.5 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.

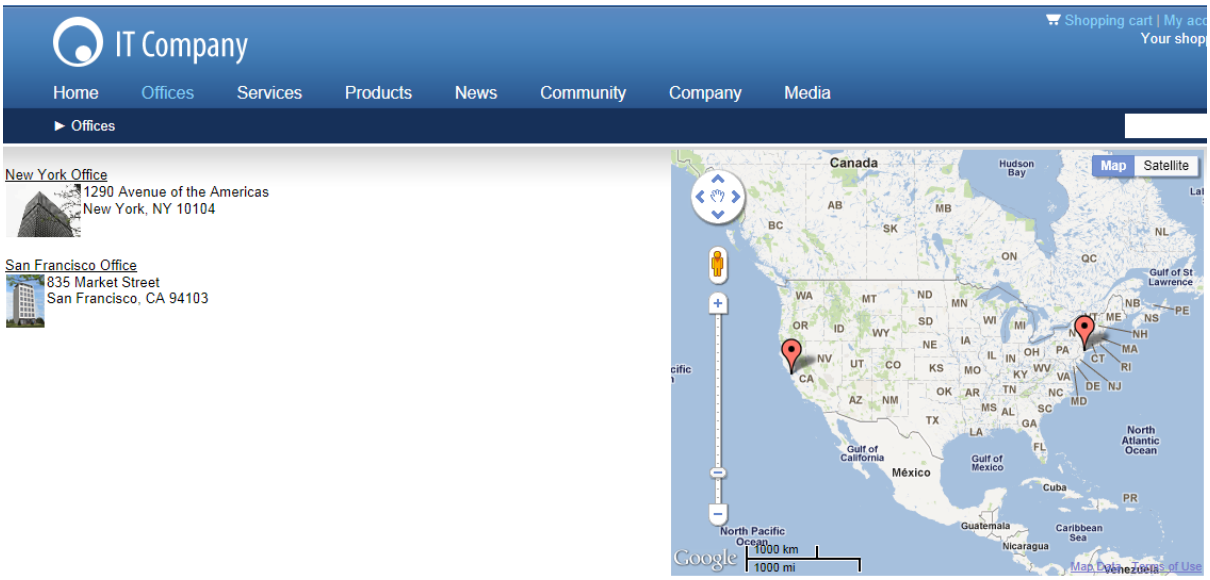
Please note

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.

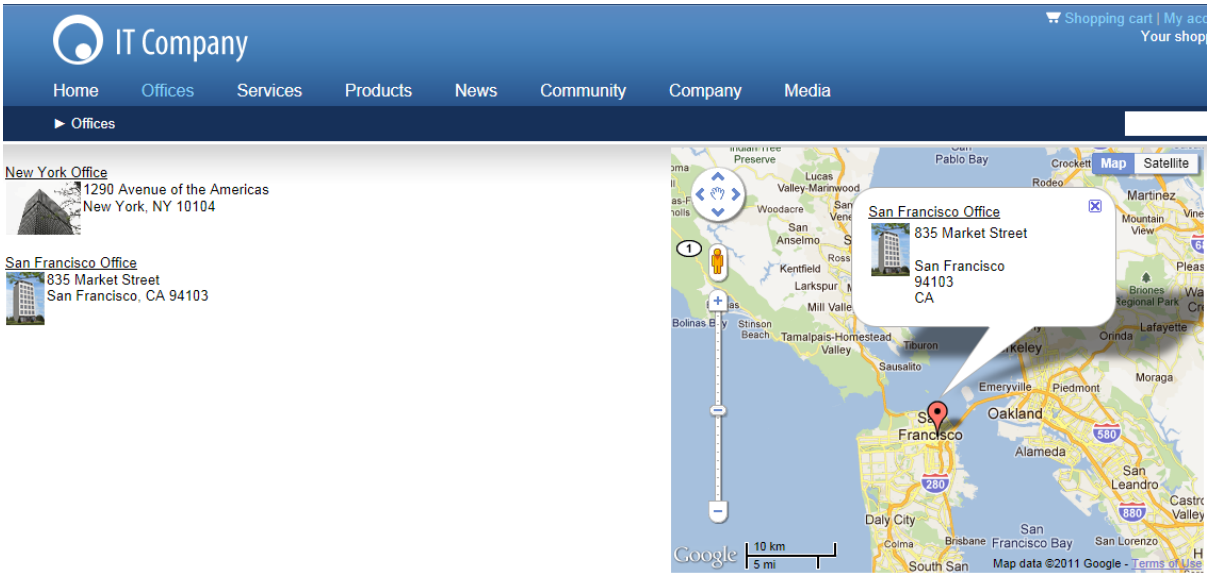
To ensure maximum efficiency, maps services results are cached if the **Use server processing** option is enabled. What is more, separate cache keys are used for the live site mode and for other modes.

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:



8.26.6 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.

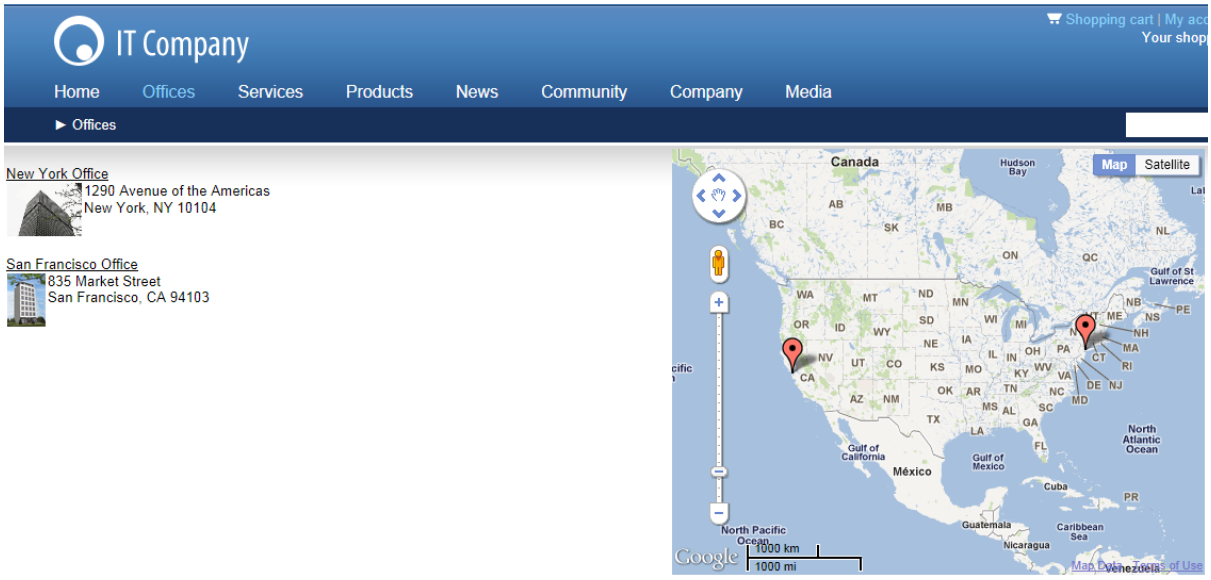
Please note

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.

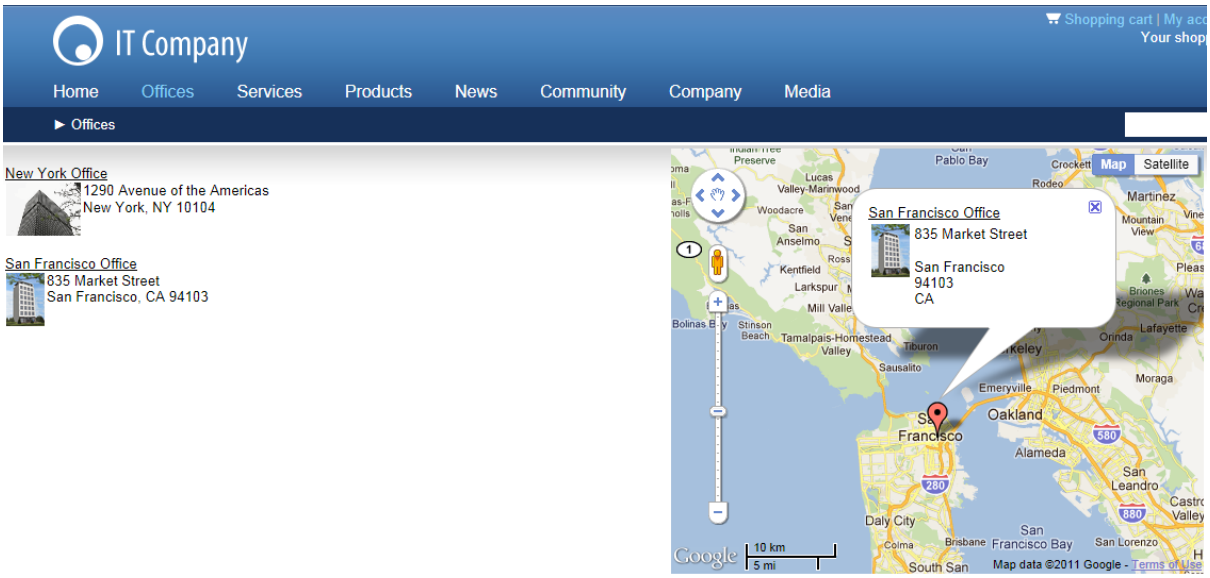
To ensure maximum efficiency, maps services results are cached if the **Use server processing** option is enabled. What is more, separate cache keys are used for the live site mode and for other modes.

Click **OK**.

8. Sign out and view the page. It will look like this:



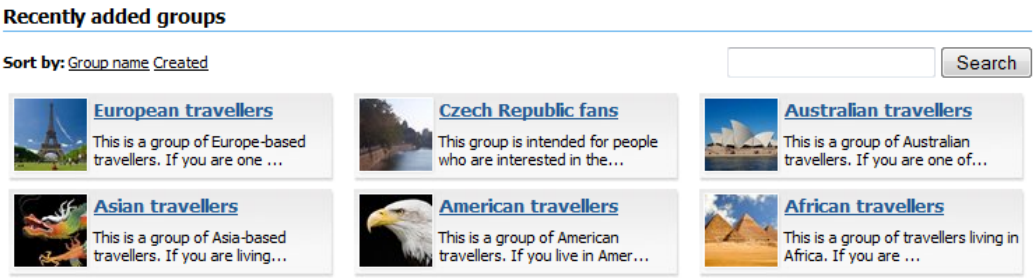
You can see two baloons on the map - these are the location markers. If you mouse-over a balloon, you will see the office name. If you click a balloon, you will see the office detailed view:



8.27 Groups

8.27.1 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.

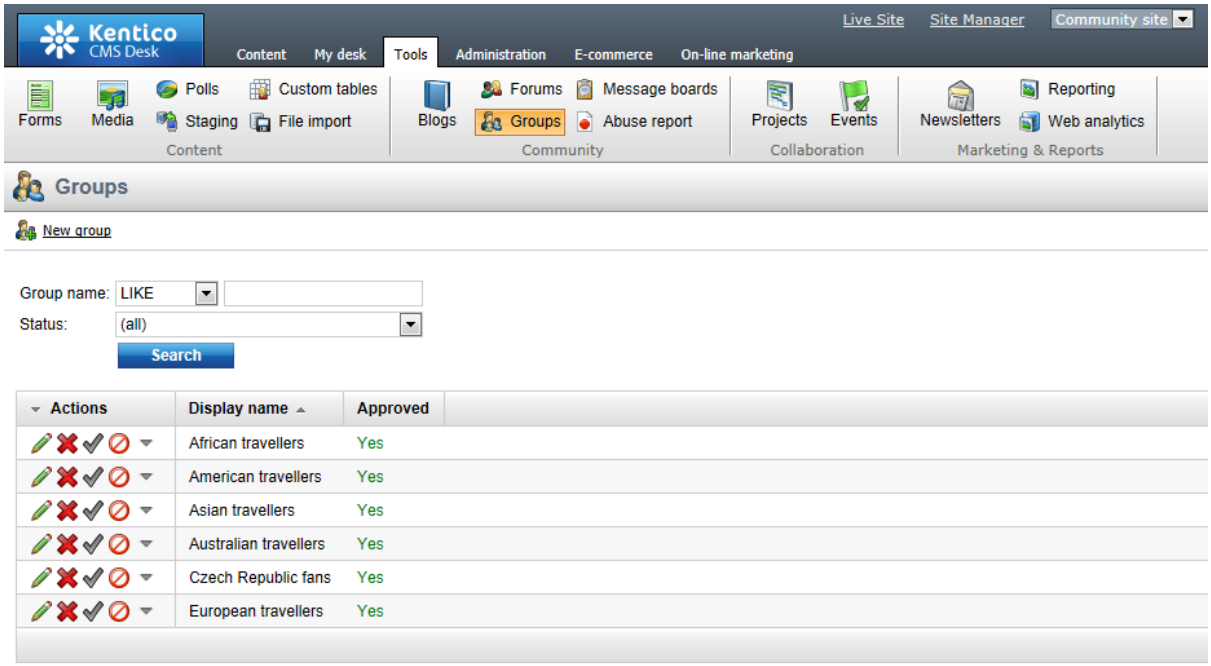
You will need to have the sample Community Site installed to follow Kentico CMS Community Site Guide.

8.27.2 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 the **New group** link 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.



8.27.3 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.

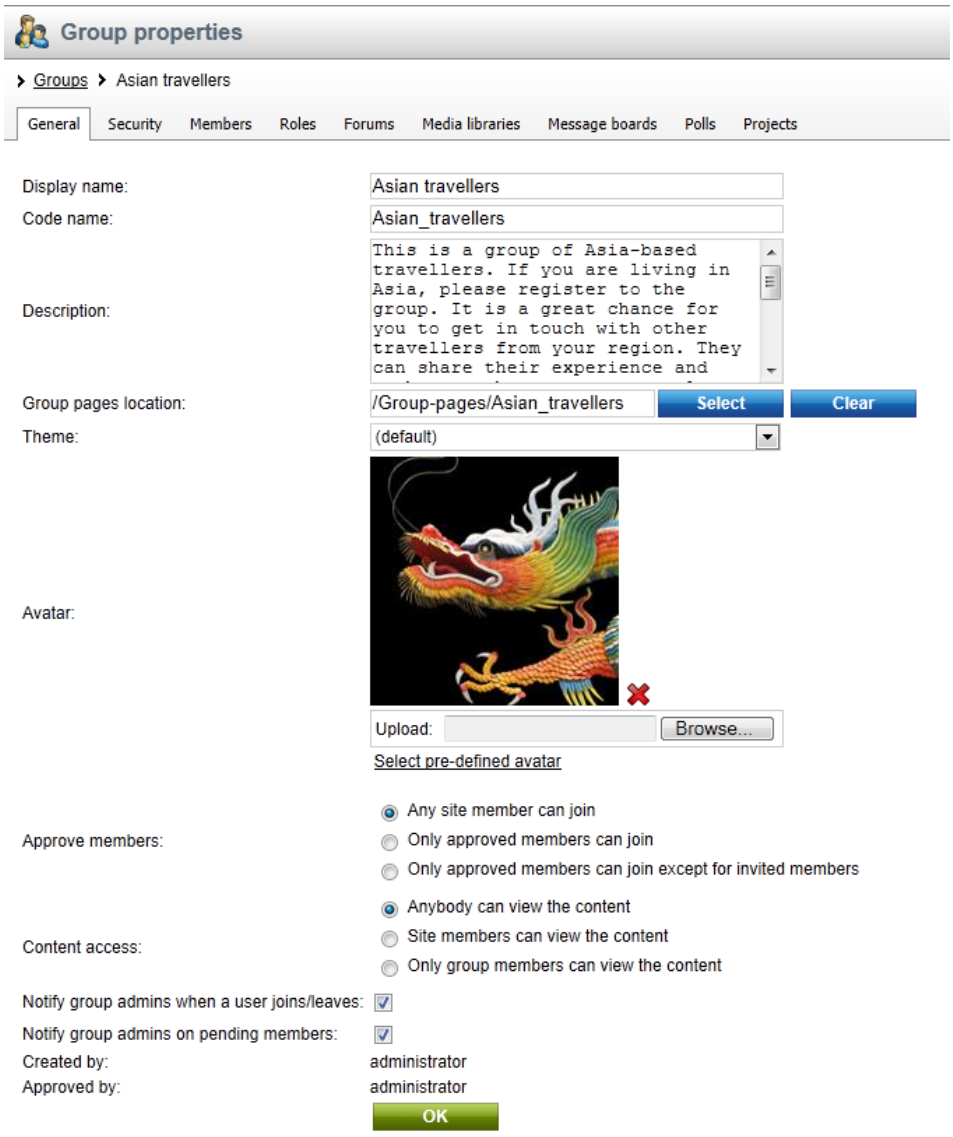
General tab

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

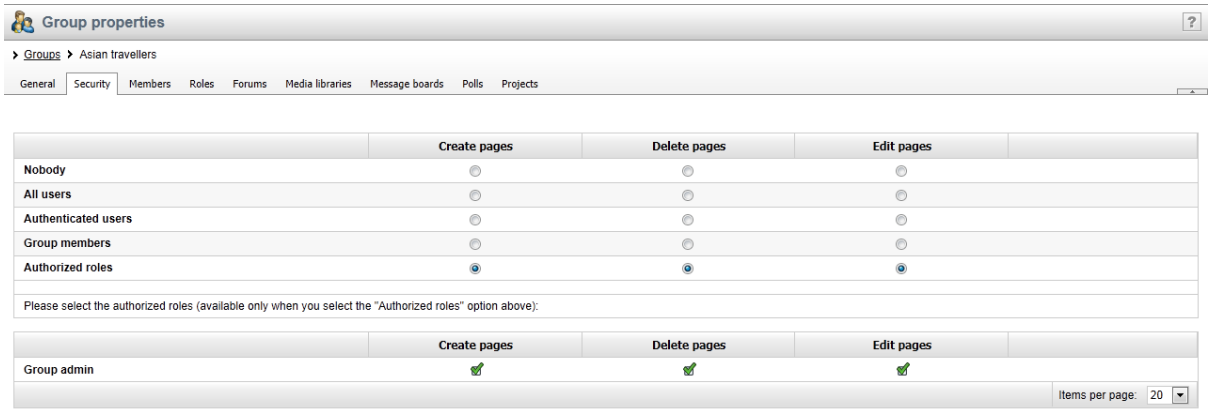
These permissions can be assigned to:

- **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

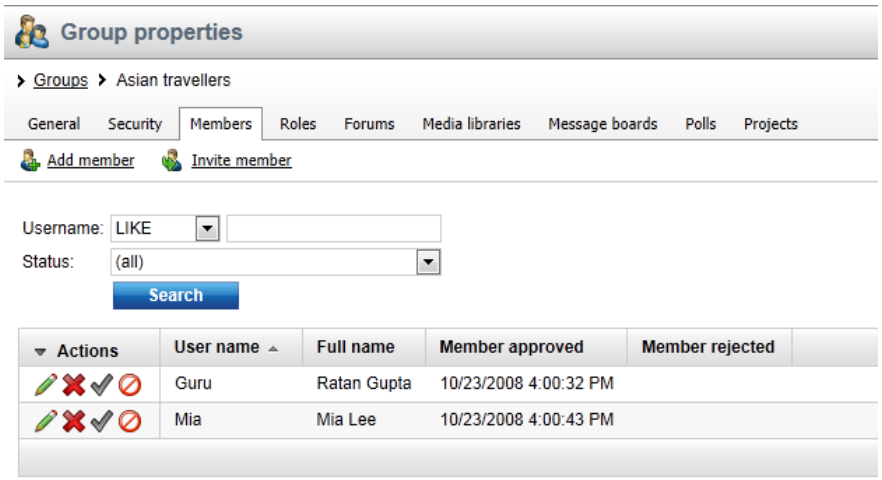
Group admin's permissions

Group administrators can perform any of these actions, even if they don't have the permissions assigned.



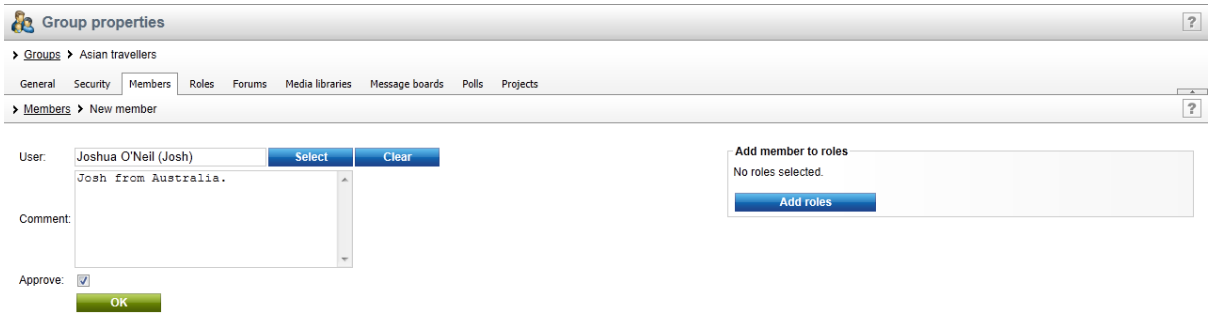
Members tab

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 the **Add member** link, 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 the **Invite member** link is displayed. When adding a user to a group, you have the following options:

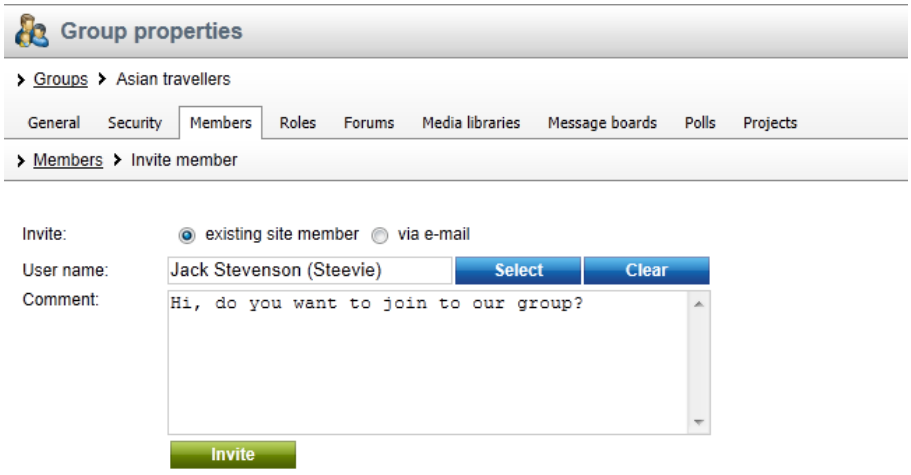
- **User** - select an existing site user who you want to add to the group
- **Comment** - text comment that you can add to the user; this comment is not sent to the user, it is only displayed in the administration interface
- **Approve** - if checked, the user will be automatically approved; if not, 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 the **Invite member** link, 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.

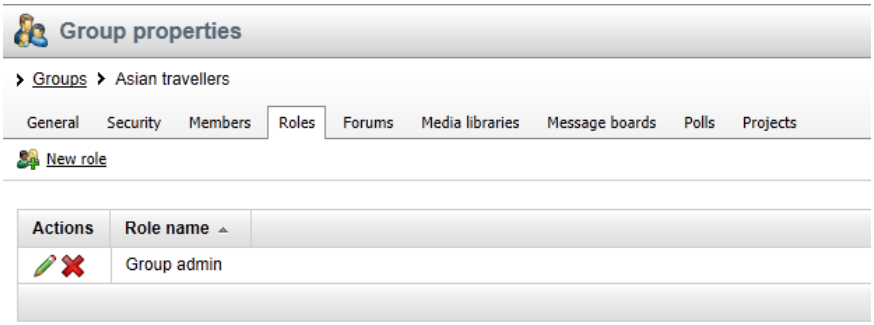
- **Invite via e-mail** - this way, you can send the invitation to any e-mail address that you enter into the **E-mail** field. In this case, user will be required to register to the site after clicking the join link in the e-mail. Text entered to the **Comment** field will be included in the e-mail.



Roles tab

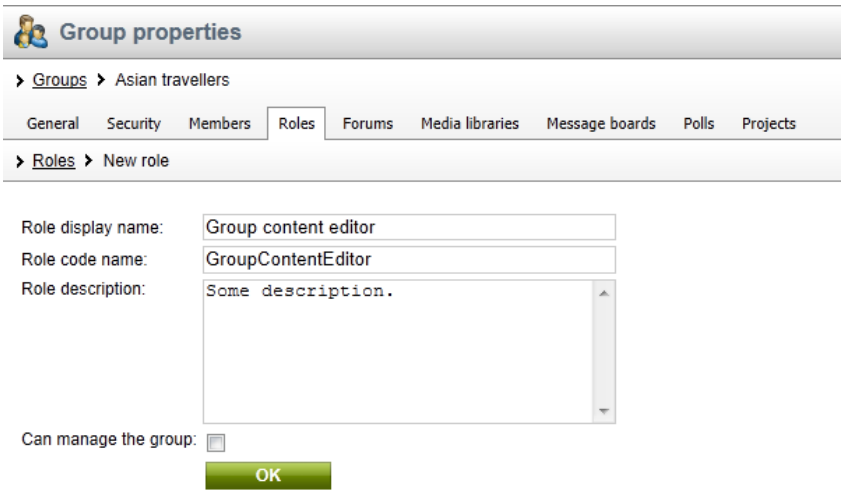
On this tab, you can see a list of roles defined for the group. These roles are applicable only in the context of the group. Don't confuse them with website roles, which can be set in **Site Manager -> Administration -> Roles**.

Roles in the list can be **Edited** () and **Deleted** ().

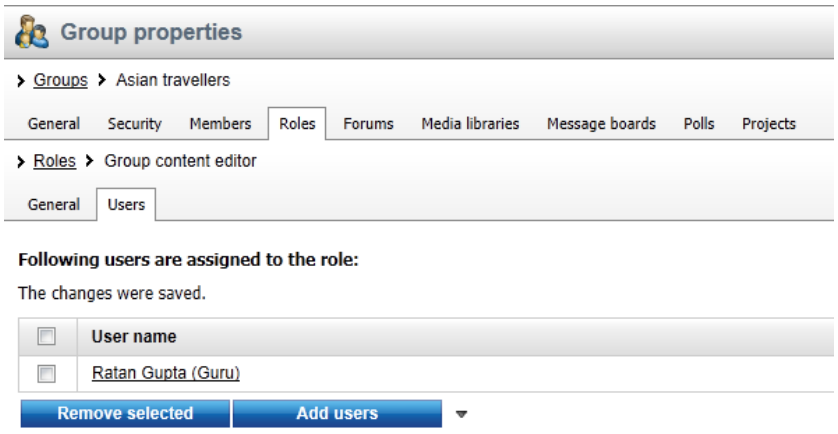


If you click the **New role** link above the list, you can define a new role for the group. The following properties can be entered:

- **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.

8.27.4 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

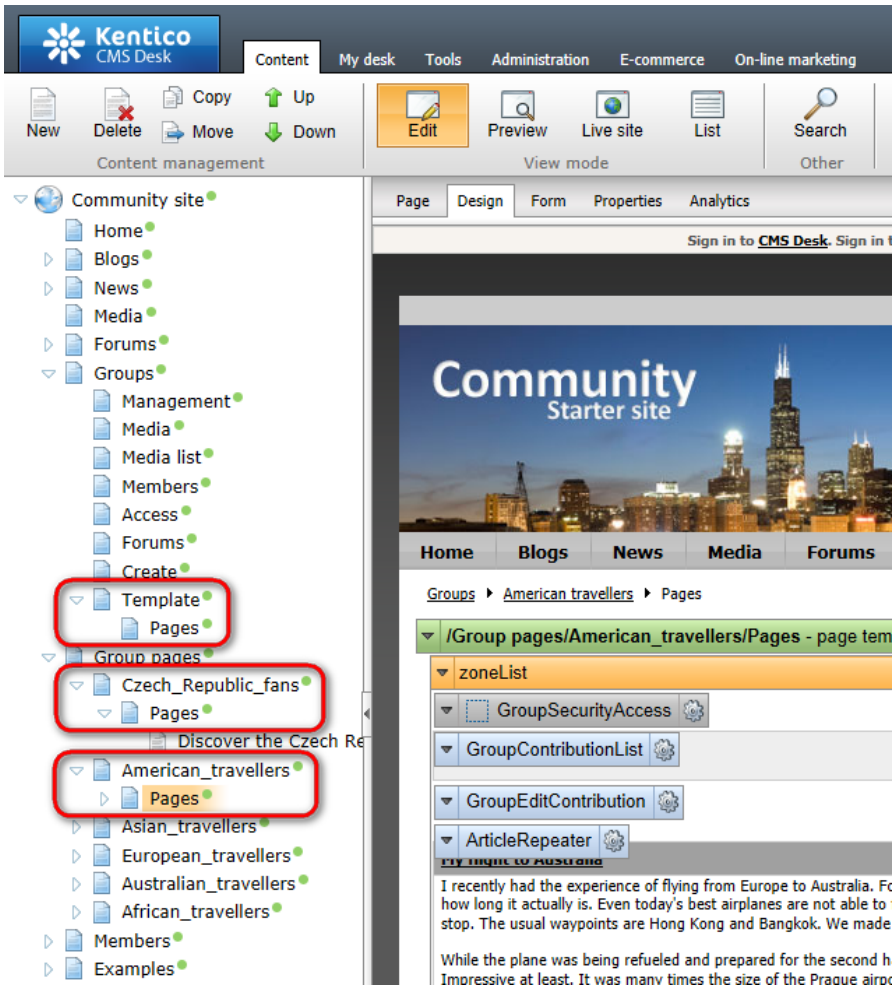
Each group has its own section on the website where its content is stored - so called group pages. When adding the **Group registration** web part to your site, you have to specify the **Template source alias path** and **Template target alias path** properties. These two properties are essential when 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:

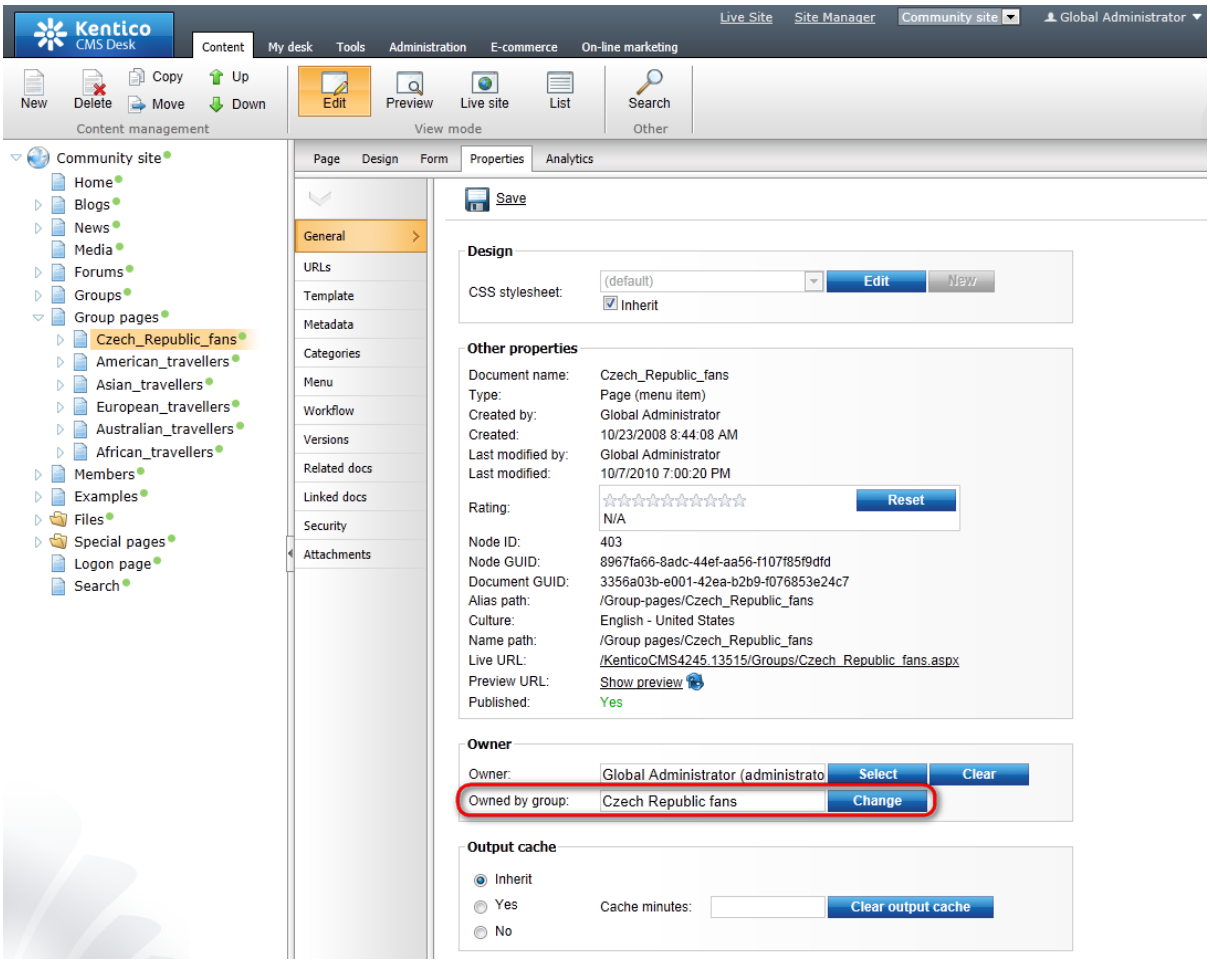
- **Template source alias path**: /Groups/Template
- **Template target alias path**: /Group-pages

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.



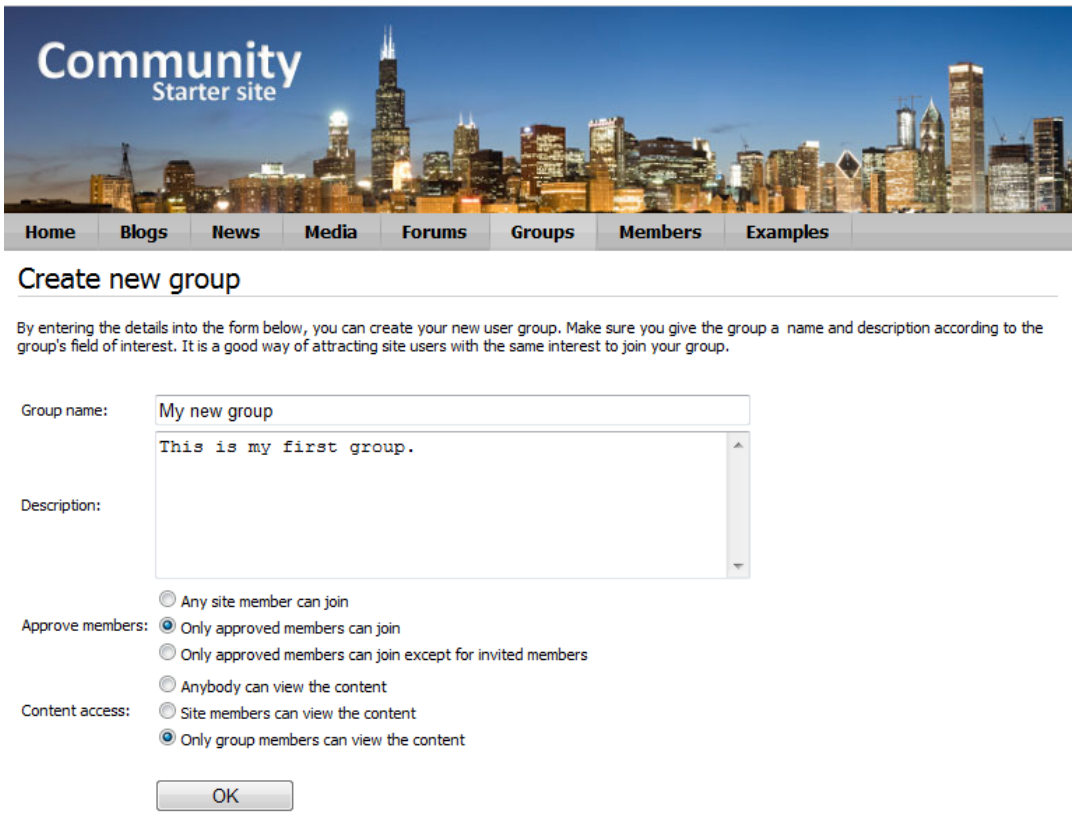
Please refer to Community Site Guide -> Part 2 -> Creating the Groups section for a step-by-step tutorial on how to create a sample group pages section of a website.

8.27.5 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.



8.27.6 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 %}*):

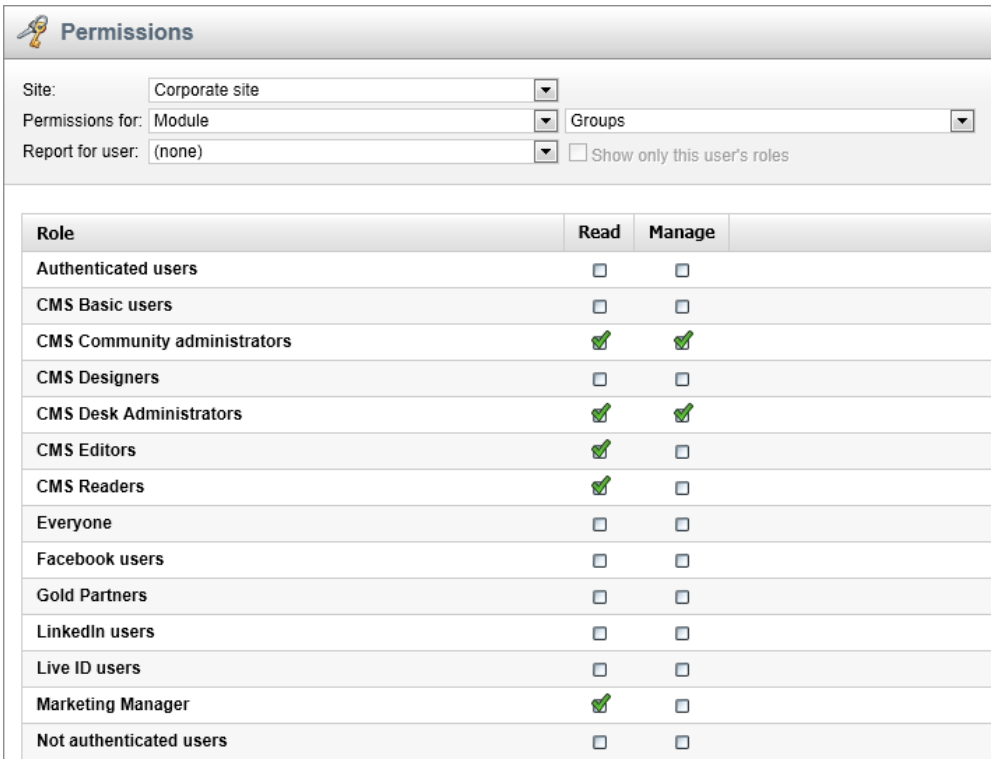
- **{% Group %}** - *GroupInfo* object of the group related to the notification.
- **{% MemberUser %}** - *UserInfo* object of the user who is attempting to join the group.
- **{% Invitation %}** - *InvitationInfo* object that can be used to access the properties of the group invitation.
- **{% Sender %}** - *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.

8.27.7 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

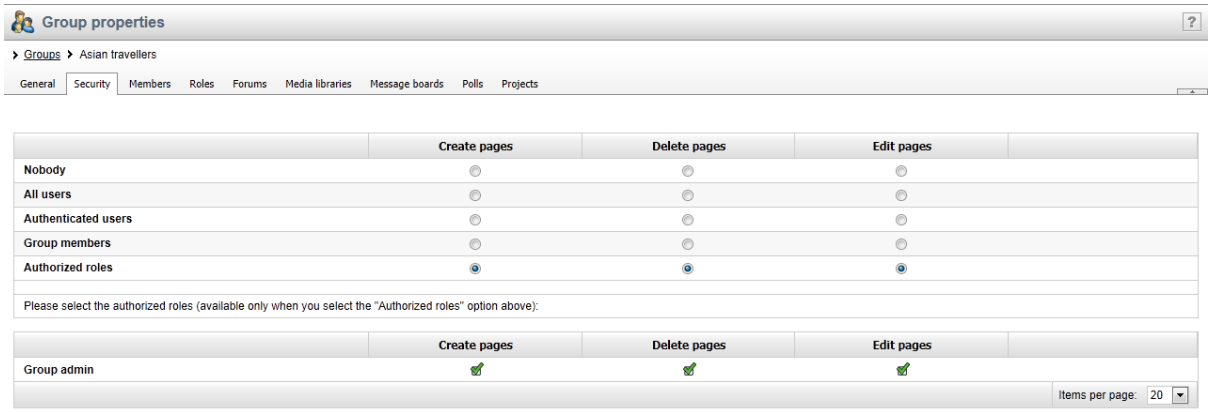
- **Create pages** - users can create group pages
- **Delete pages** - users can delete group pages
- **Edit pages** - users can edit group pages

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.



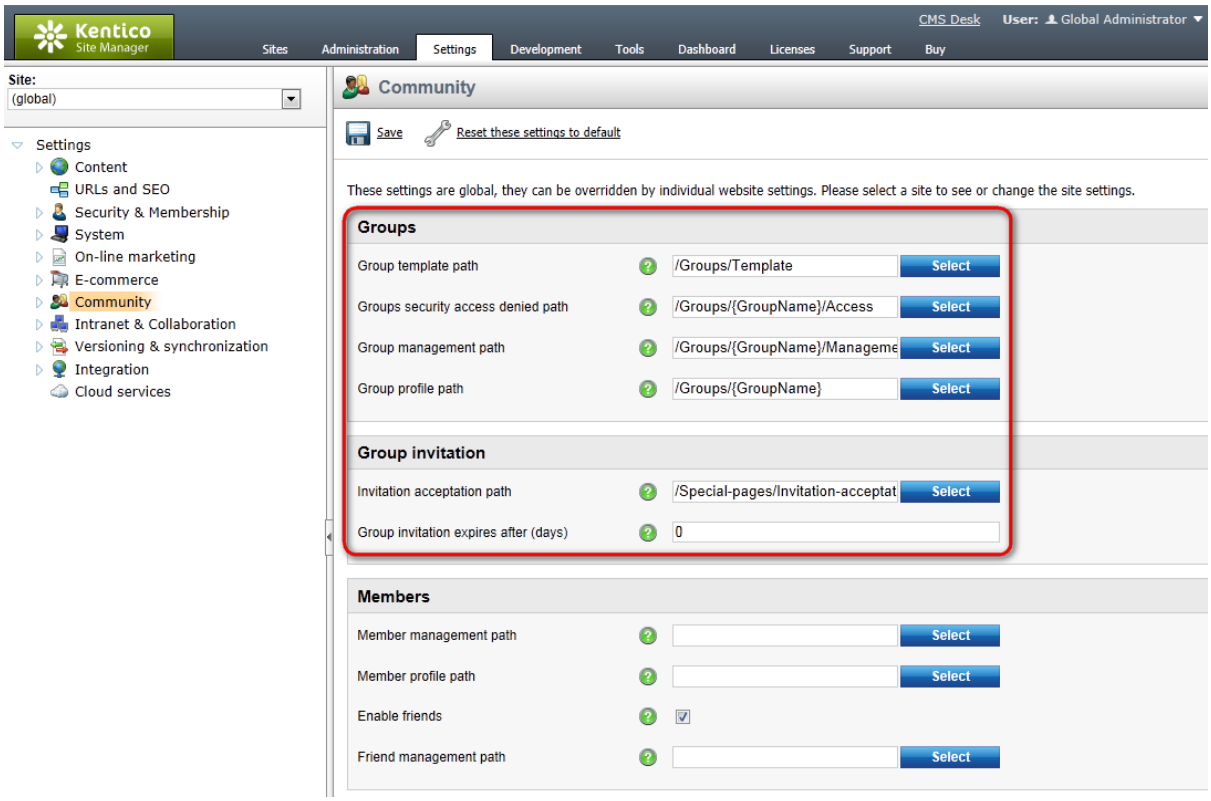
8.27.8 Settings

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.



8.27.9 Groups internals and API

8.27.9.1 Overview

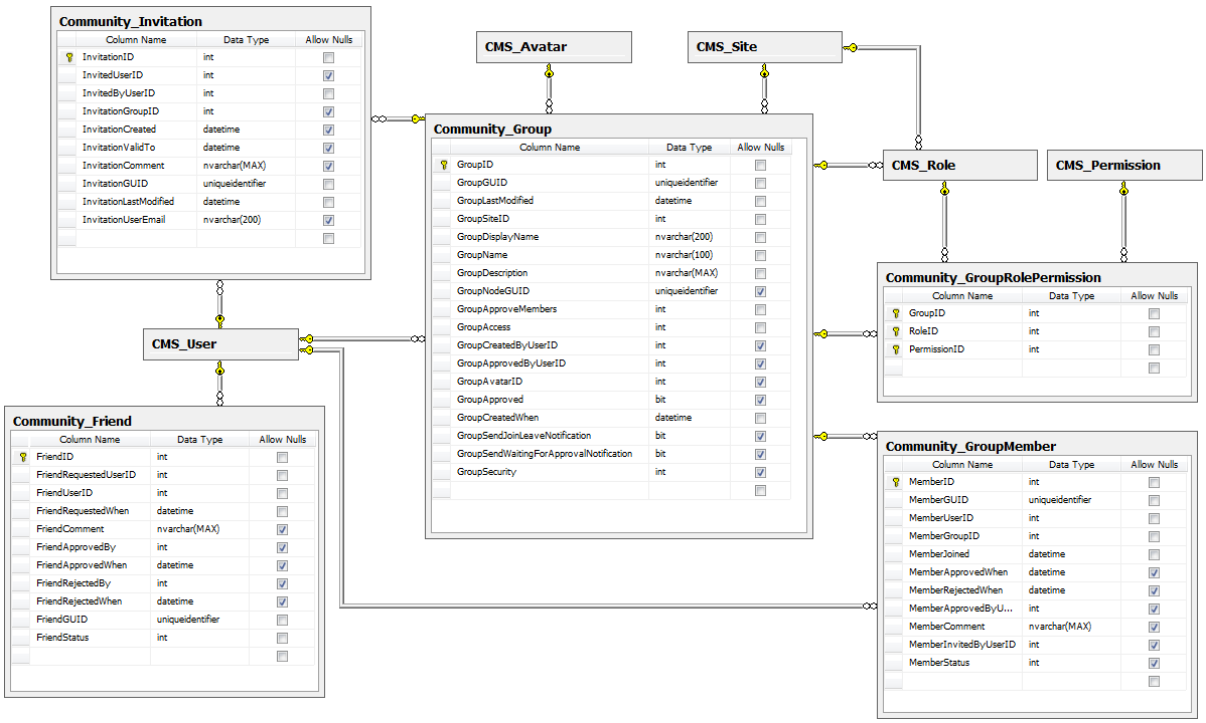
In this chapter, you will learn which database tables and API classes are used in the Groups module. You will also see the most common API examples.

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.

8.27.9.2 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*.

8.27.9.3 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 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.

8.27.9.4 API examples

8.27.9.4.1 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;
```

8.27.9.4.2 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);
}
```

8.27.9.4.3 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;
}
```

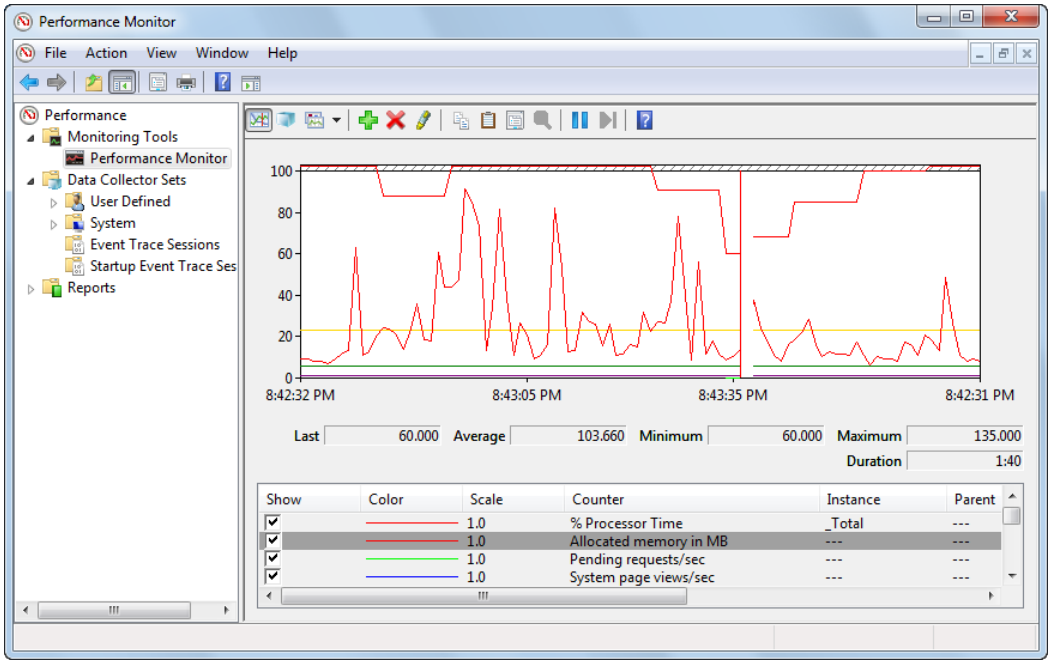
The following example creates an invitation to a group for a user.

```
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;
}
```

8.28 Health monitoring

8.28.1 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 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 how to Windows service can be installed, please refer to the Installing Health monitoring Windows service topic.

8.28.2 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

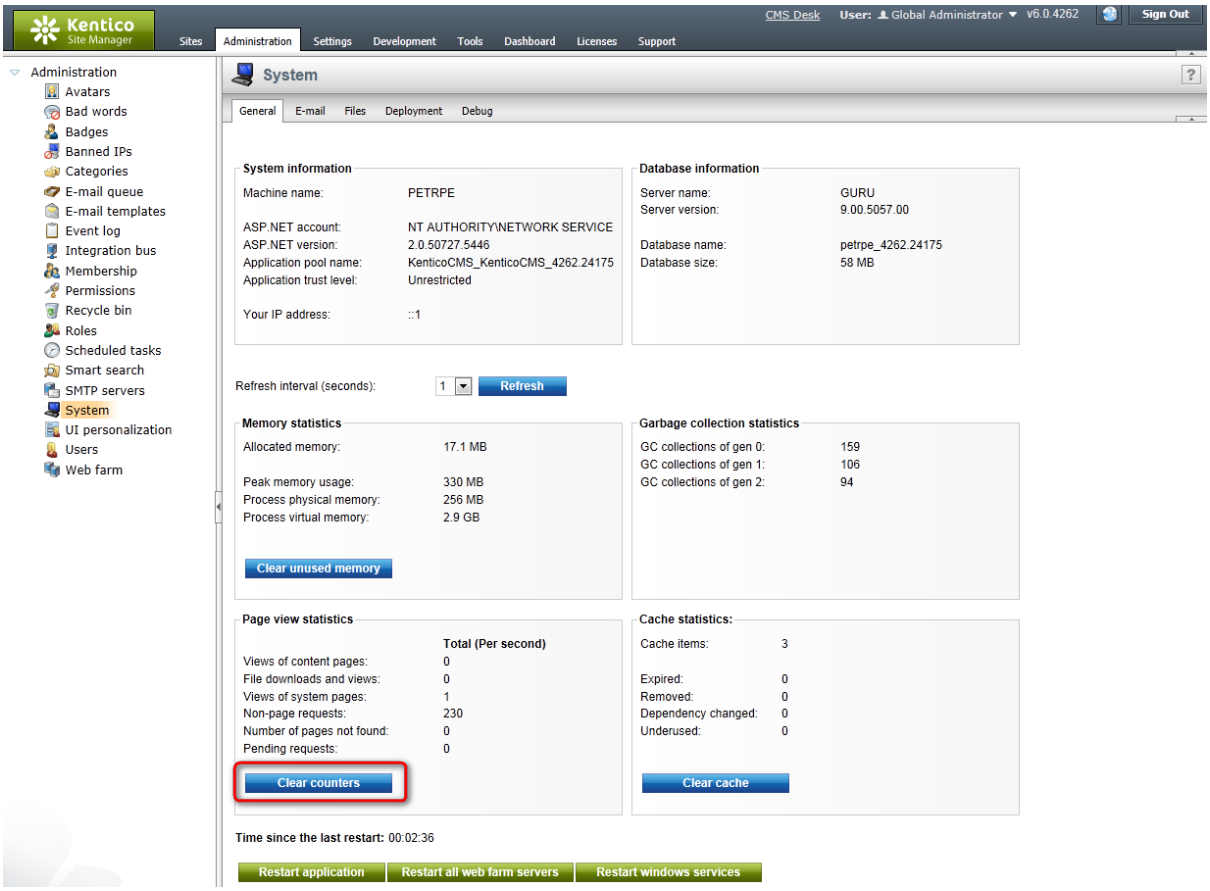
Cache underused items/sec	Global	The number of underused cache items per second.	Application
Content page views/ sec	Global and Sites	The number of content pages viewed per second.	Application
E-mails in queue	Global and Sites	The number of e-mails in E-mail queue.	Application or Windows service[1]
Error 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.

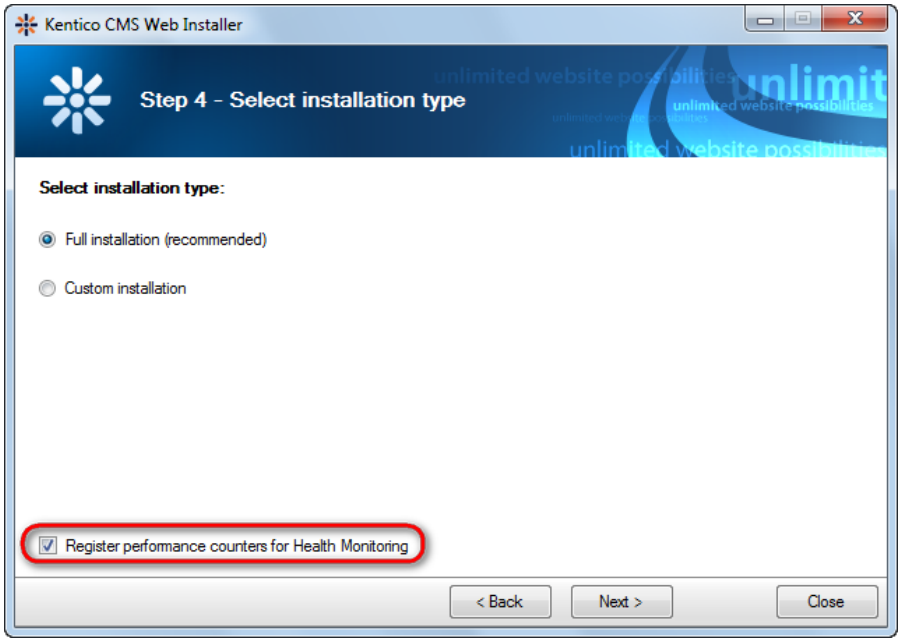


8.28.3 Registering performance counters

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.



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 inside the Kentico CMS installation folder (typically *C:\Program Files\Kentico CMS\<version>\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

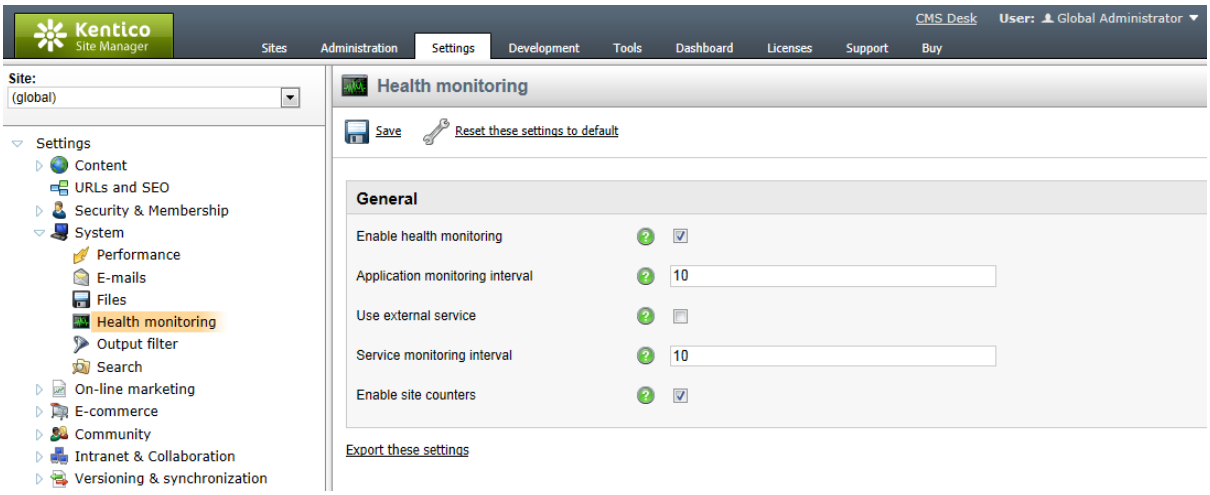
Registration of performance counters is technically performed by adding specific keys to Windows registry. In the registry, they are located in *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services*. The keys represent individual registered counter categories and have the same names as the respective registered counter categories. By deleting the registry keys, you perform removal of the respective counter categories so that they are no longer registered.

8.28.4 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.

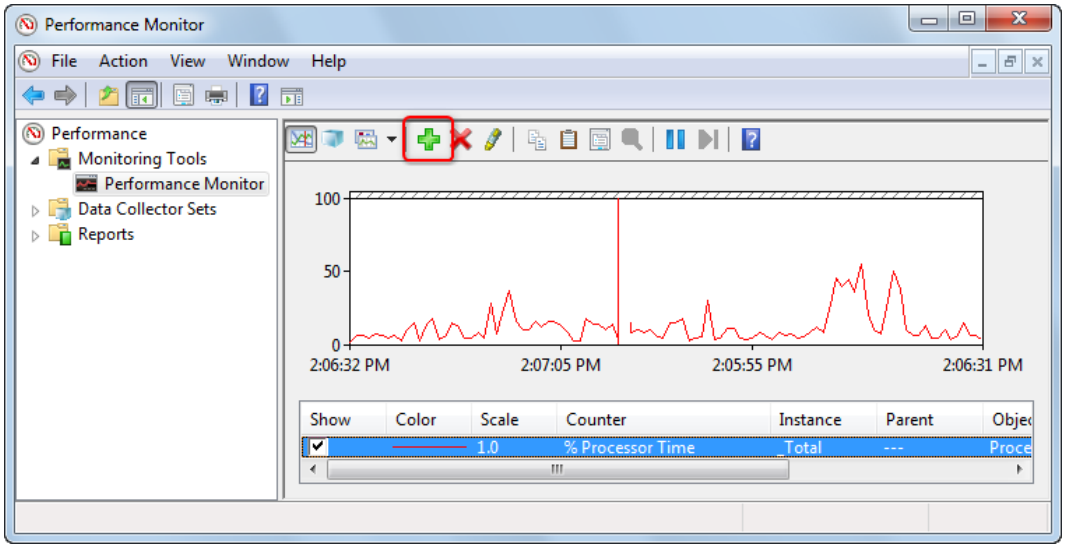


8.28.5 Monitoring using Performance monitor

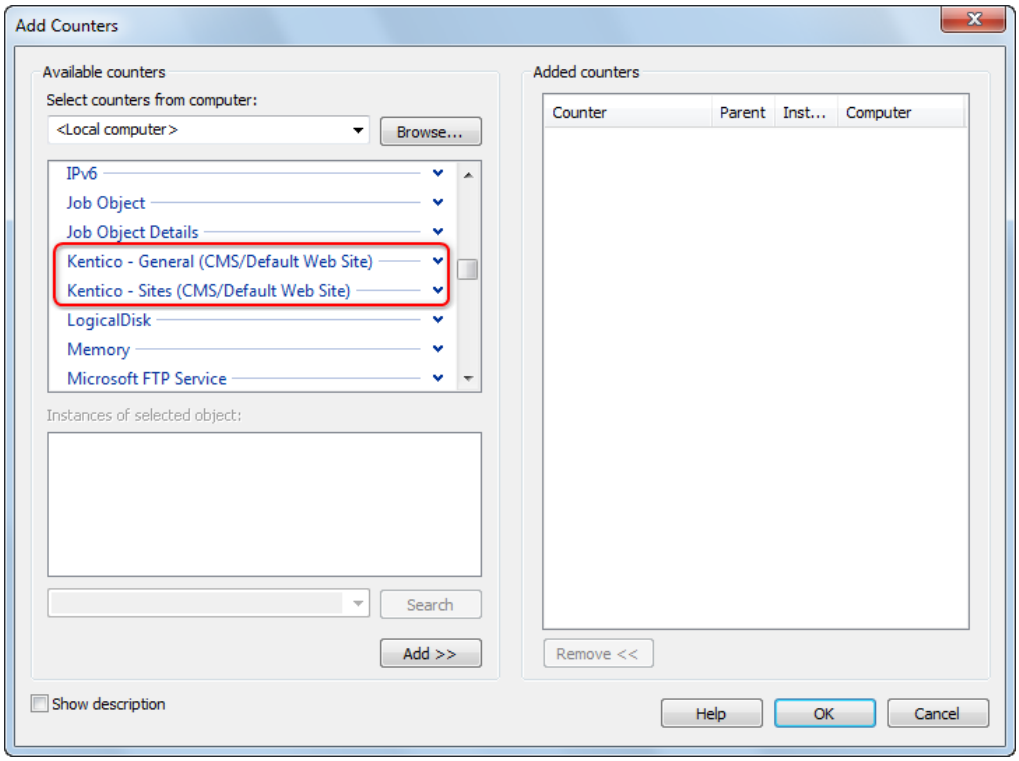
When you have performance counters for your instance of Kentico CMS registered in Windows and Health monitoring settings adjusted, you can start monitoring values written to the counters. There is a number of applications that can be used for this purpose. In this topic, we will use the built-in Performance 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.



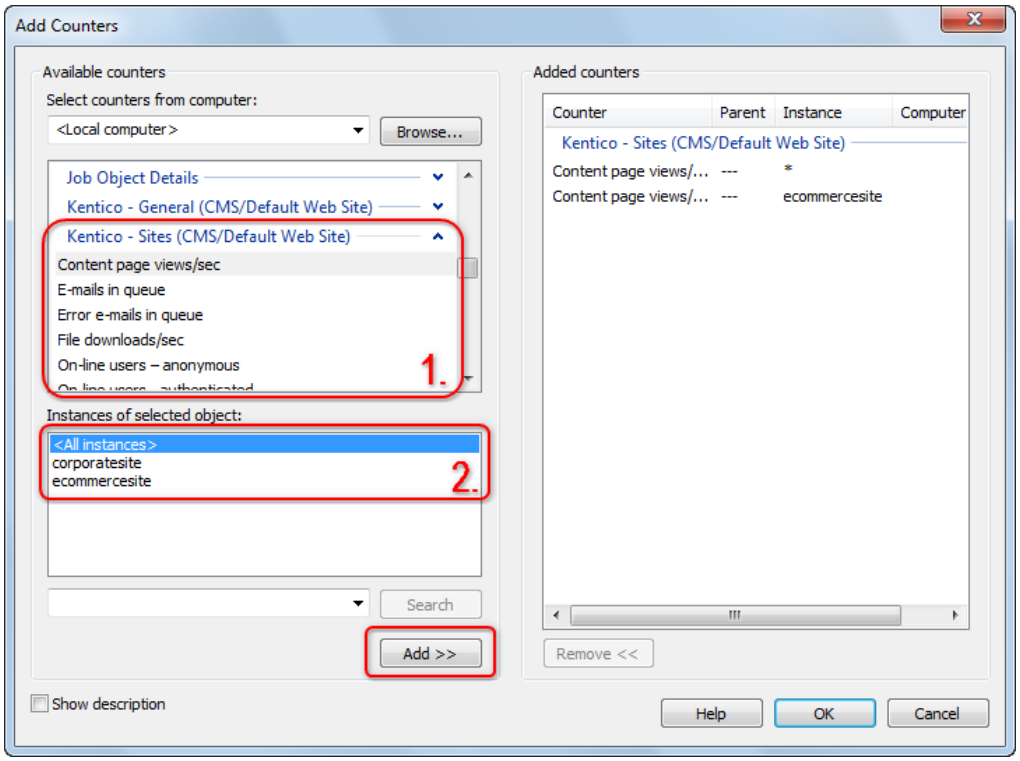
The **Add Counters** window pops up. From the **Select counters from computer** drop-down list, choose **<Local computer>**. In the section below, you will see a list of all counter categories currently registered in Windows. Among them, you will find two counter categories for each Kentico CMS instance with registered performance counters — **Kentico - General (<IIS path>/<IIS website>)** and **Kentico - Sites (<IIS path>/<IIS website>)**.



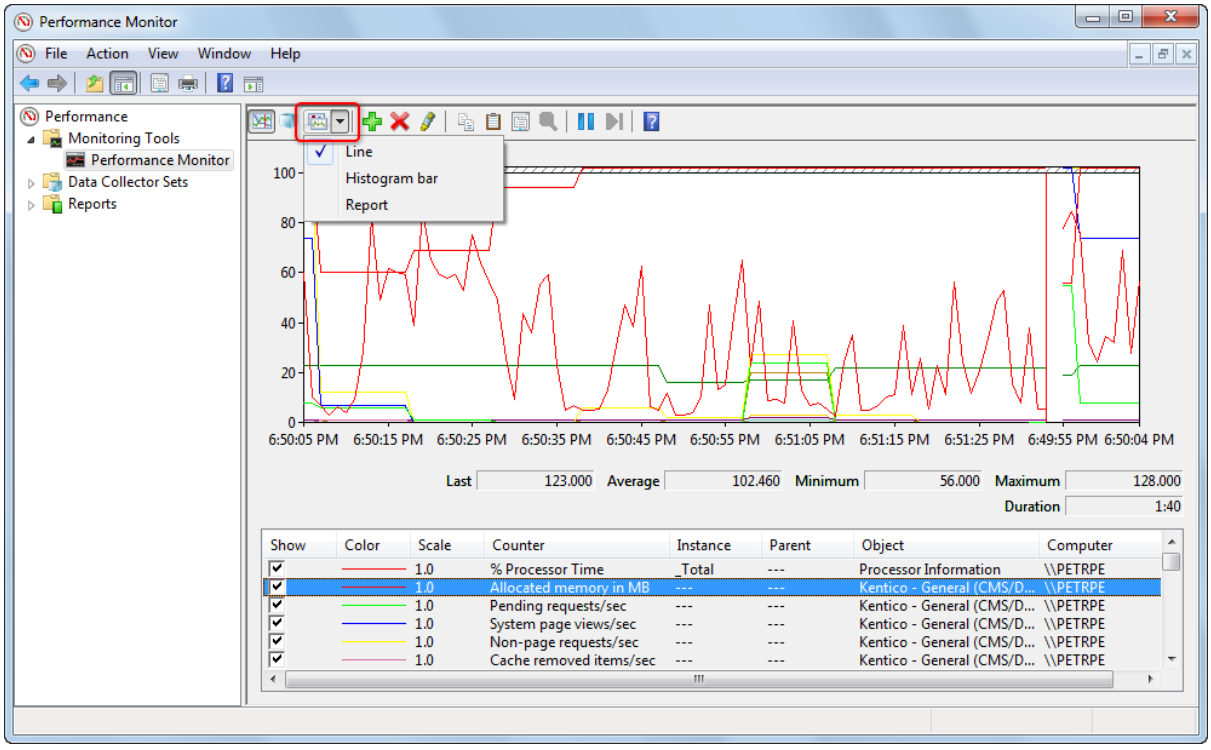
If you click the down-arrow next to a category name, you will expand all counters present in the category (1. in the screenshot below). You can either add all counters in a category by selecting the category and clicking **Add >>**, or just individual counters by selecting the counter and clicking the **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.



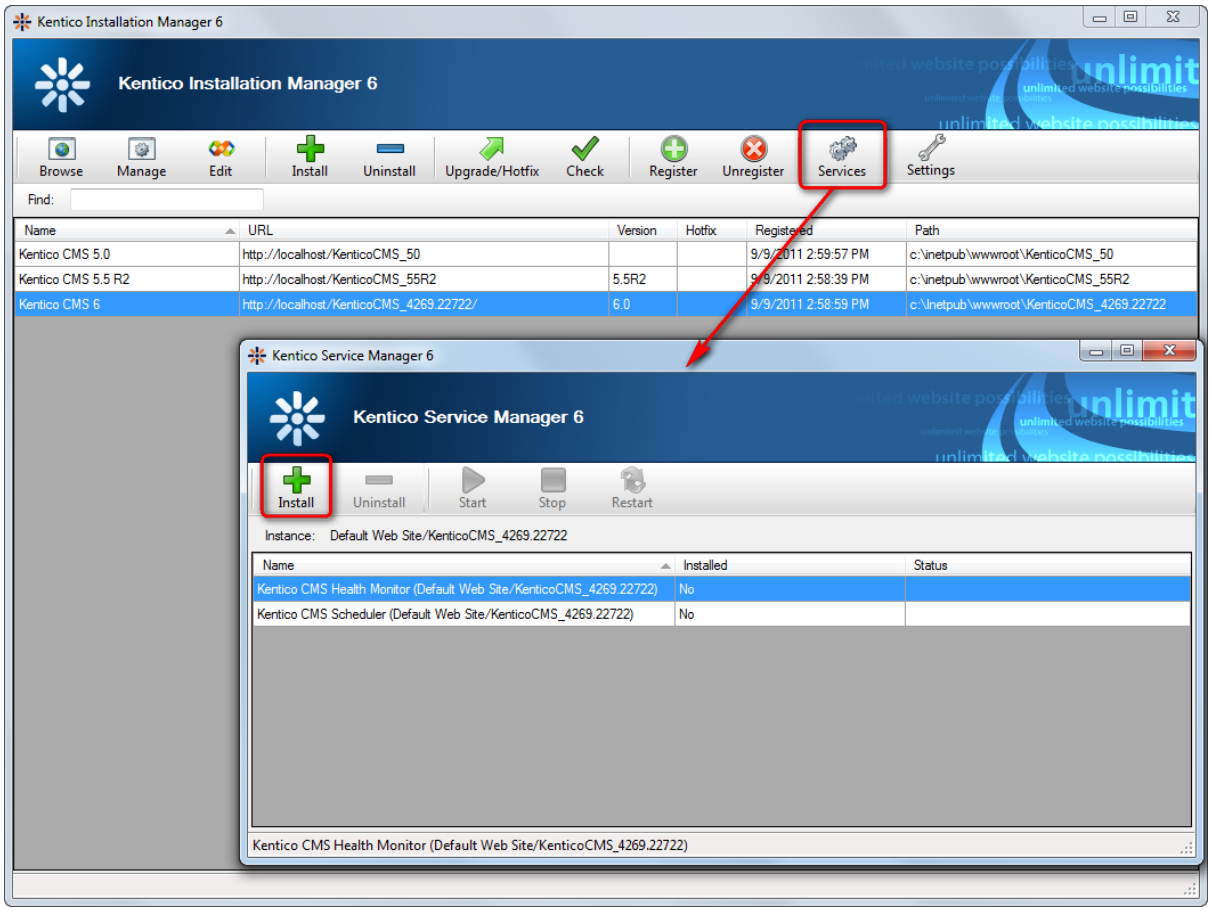
8.28.6 Installing Health monitoring Windows service

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.

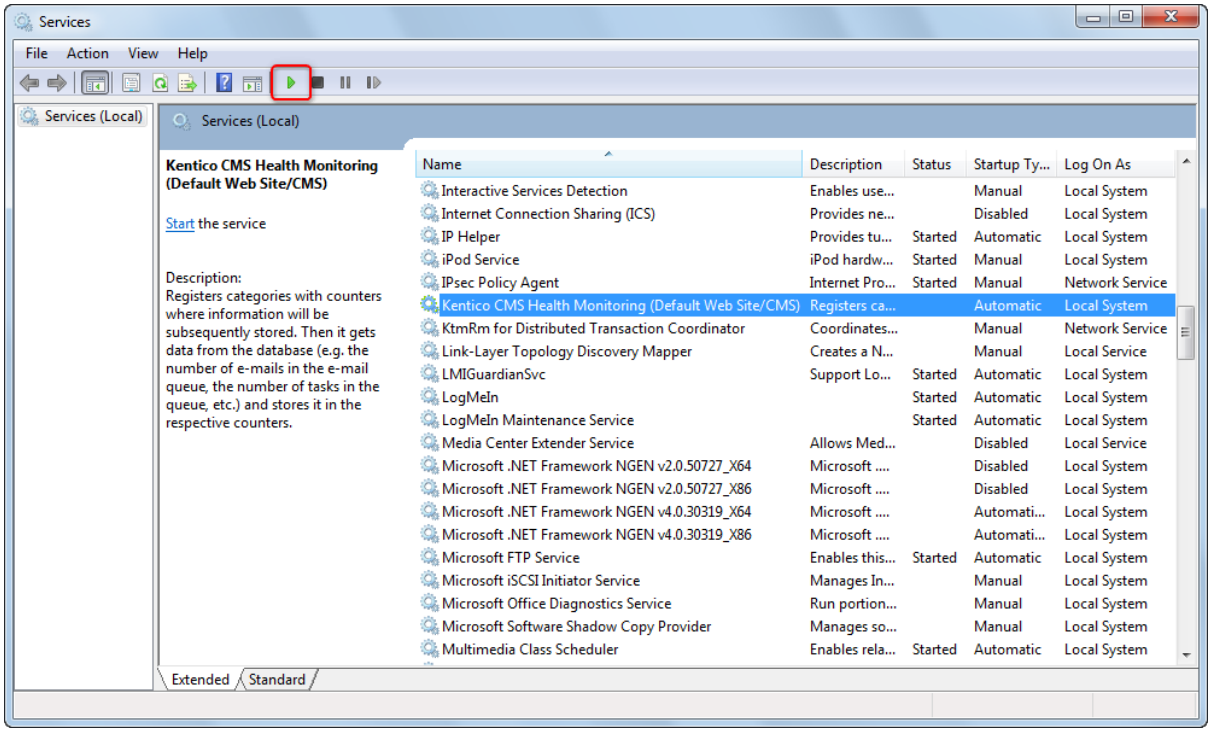
1. Open Windows command line (type *cmd* in the Start menu search box) and navigate to the .NET folder containing *InstallUtil.exe* (e.g. *c:\Windows\Microsoft.NET\Framework64\v4.0.30319*).

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.

8.28.7 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. First, 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. Next, 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:

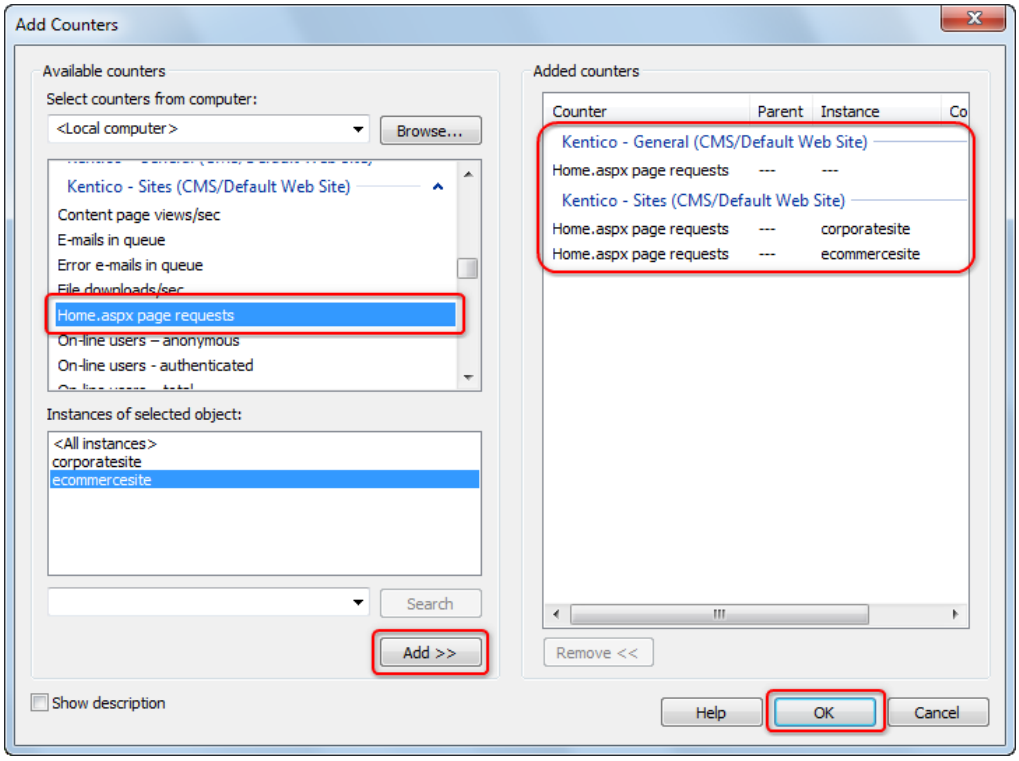
```
HealthMonitoringService.exe /webpath="<disk path to web project root>" /
createcounters
```

This action will reload all counters in both categories, including the newly added one.

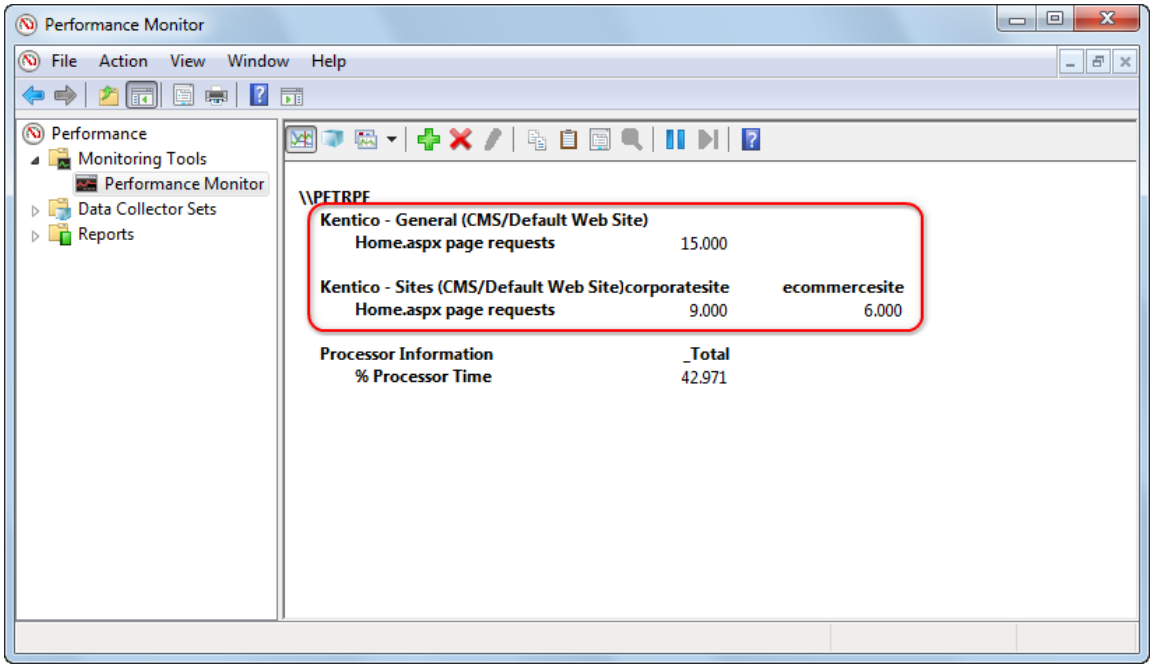
> **Custom counters and Health Monitoring Windows service**
>
> Values of all custom counters can be logged only by the application itself, i.e. it is not possible for values of custom counters to be logged by the Windows service. The only purpose of using the service in this step is to register the 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.



8.29 Image gallery

8.29.1 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).

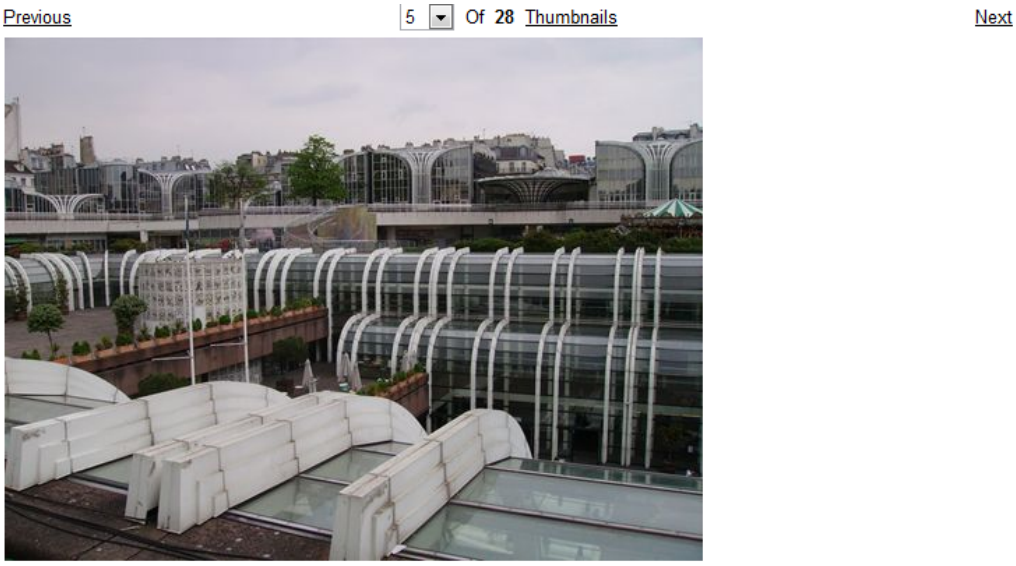
8.29.2 Available web parts

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:



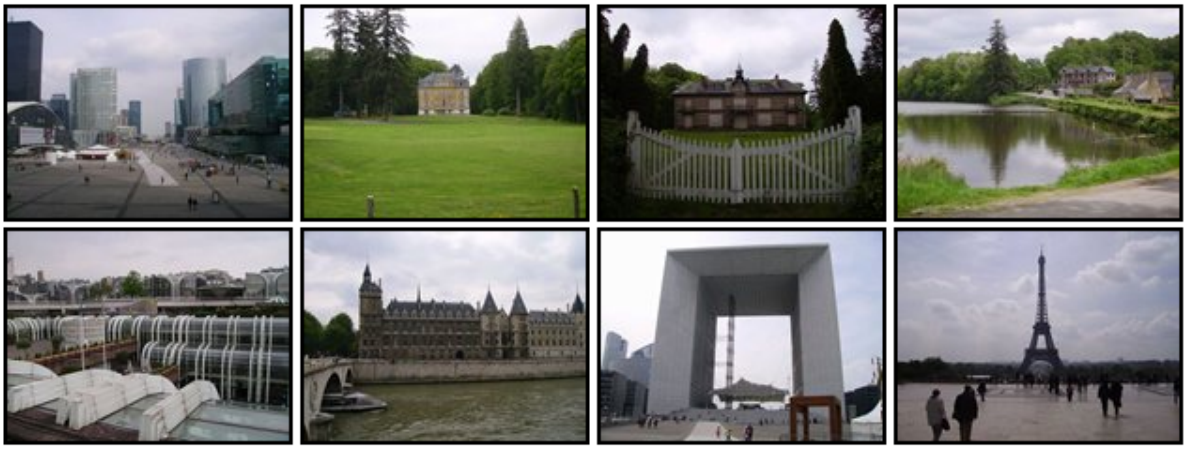
Besides the usual parameters common to all web parts, these properties can be set to customize the appearance of the gallery:

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

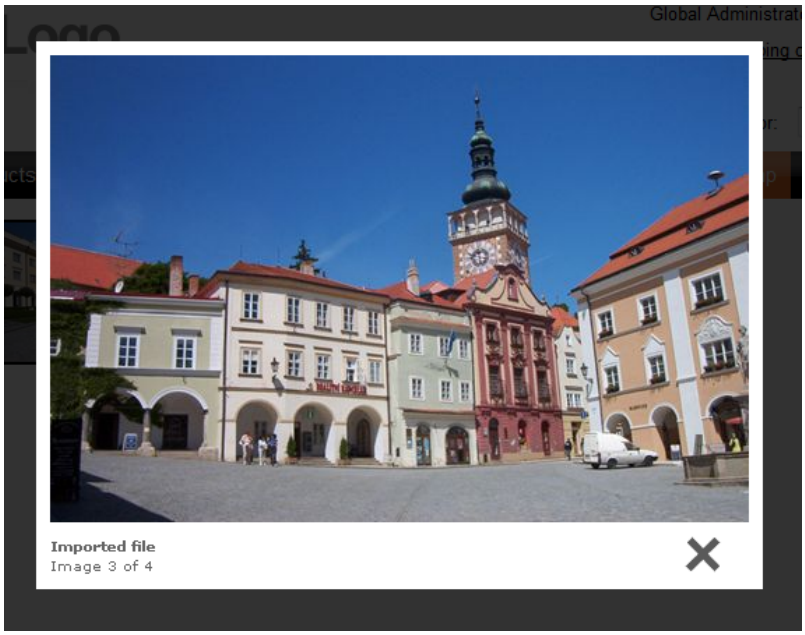
This web part's thumbnail view is similar to that of the **Image gallery** web part:



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:



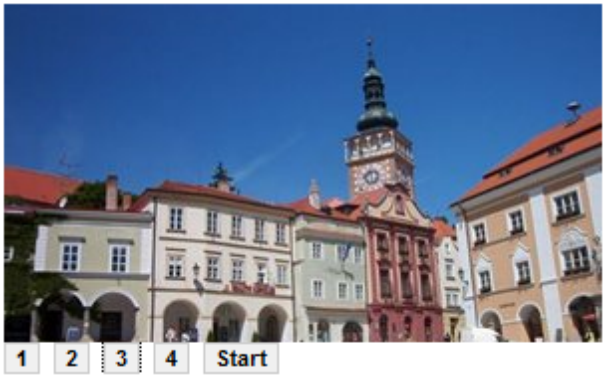
Here is a list of properties specific to the **Lightbox gallery** web part:

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.

8.29.3 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.

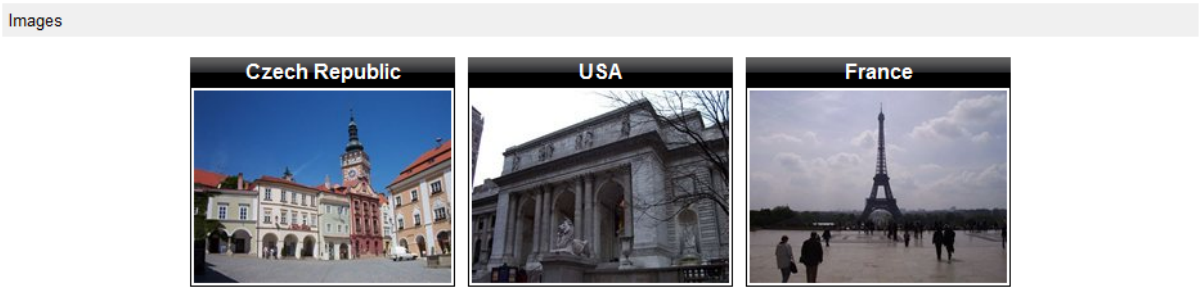


Image gallery

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.

8.29.4 Importing images

Images used in galleries are imported the same way as any other files, as described in the File import chapter.

8.29.5 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 properties window. For each transformation, 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

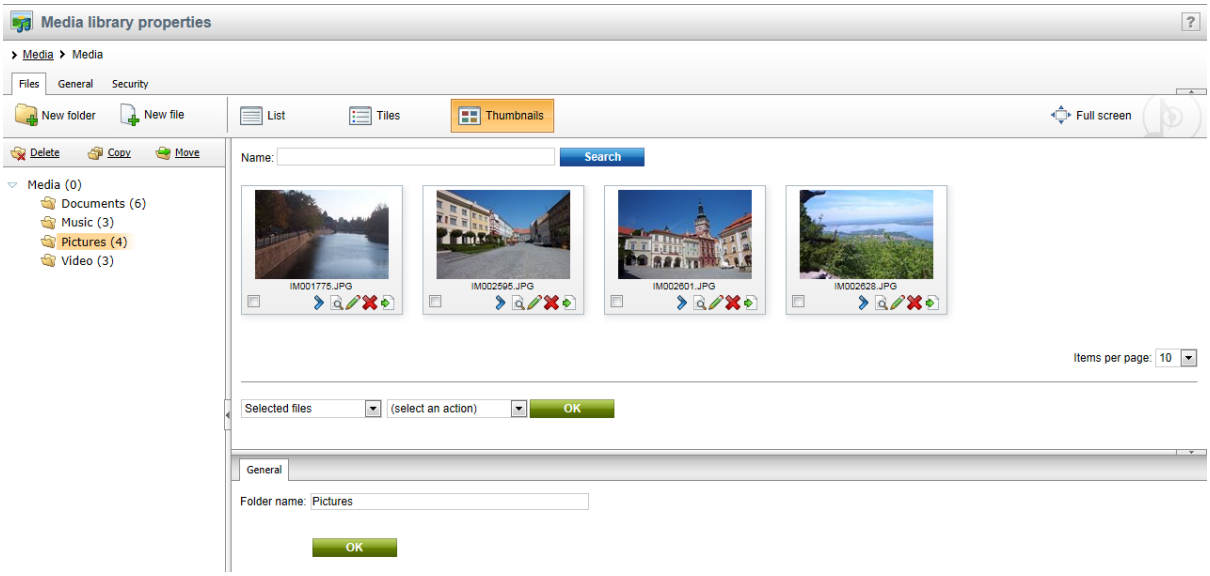
When writing a custom transformation for the **Lightbox gallery** web part, it is necessary to use the 'rel' and 'rev' parameters as highlighted in the transformation code below. The 'title' parameter is used to determine the description of the image displayed in the lightbox.

```
<a href="<%# GetDocumentUrl() %>" rel="lightbox[group]" rev="<%# Eval
("NodeAliasPath") %>"
title="<%# Eval("FileDescription") %>"><img src="<%# GetFileUrl
("FileAttachment") %>?maxsidesize=150"
alt="<%# Eval("FileName") %>" /></a>
```

8.30 Media

8.30.1 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.

8.30.2 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.

8.30.3 Creating media library

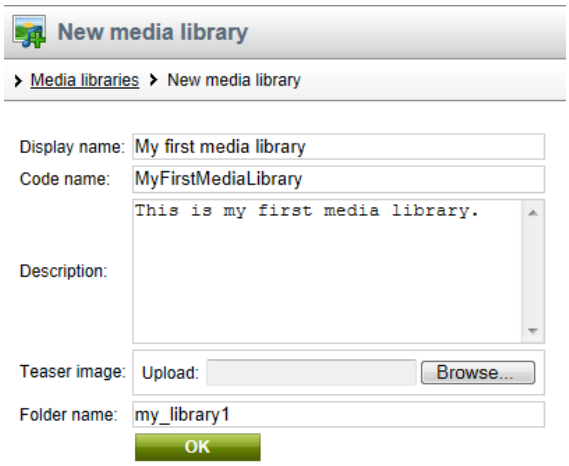
There are two ways how media libraries can be created:

- **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**.

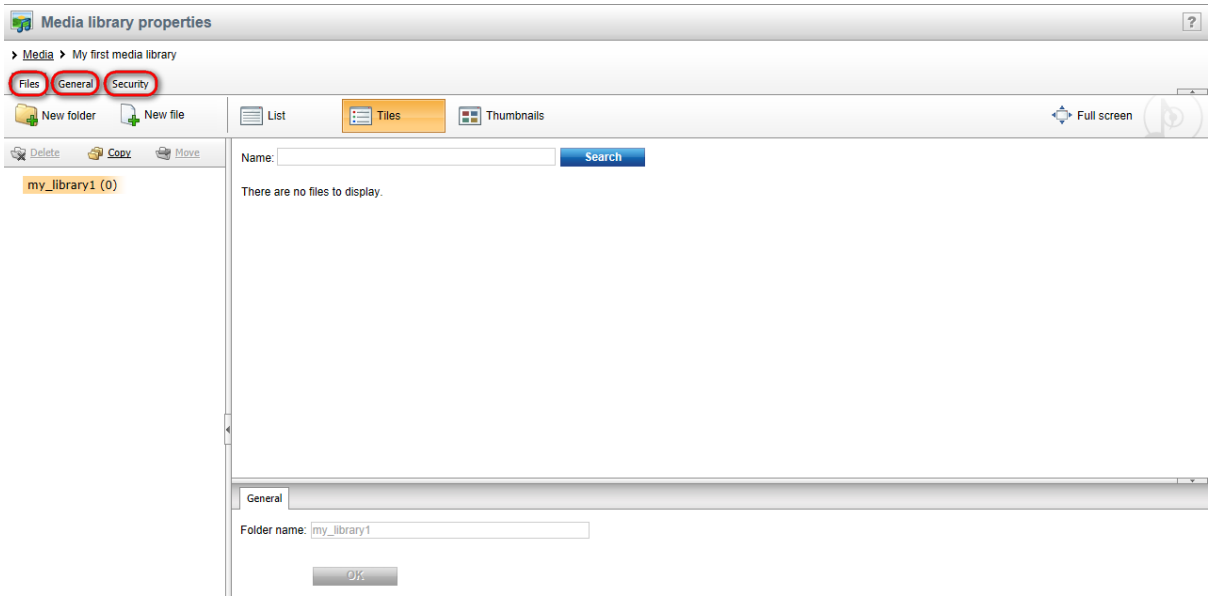
1. In all of these cases, media libraries can be created by clicking the **New media library** link above the list of libraries, as you can see in the screenshot below.

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.

8.30.4 Media library content

8.30.4.1 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.

8.30.4.2 Uploading files

There are three ways how files can be uploaded into a media library:

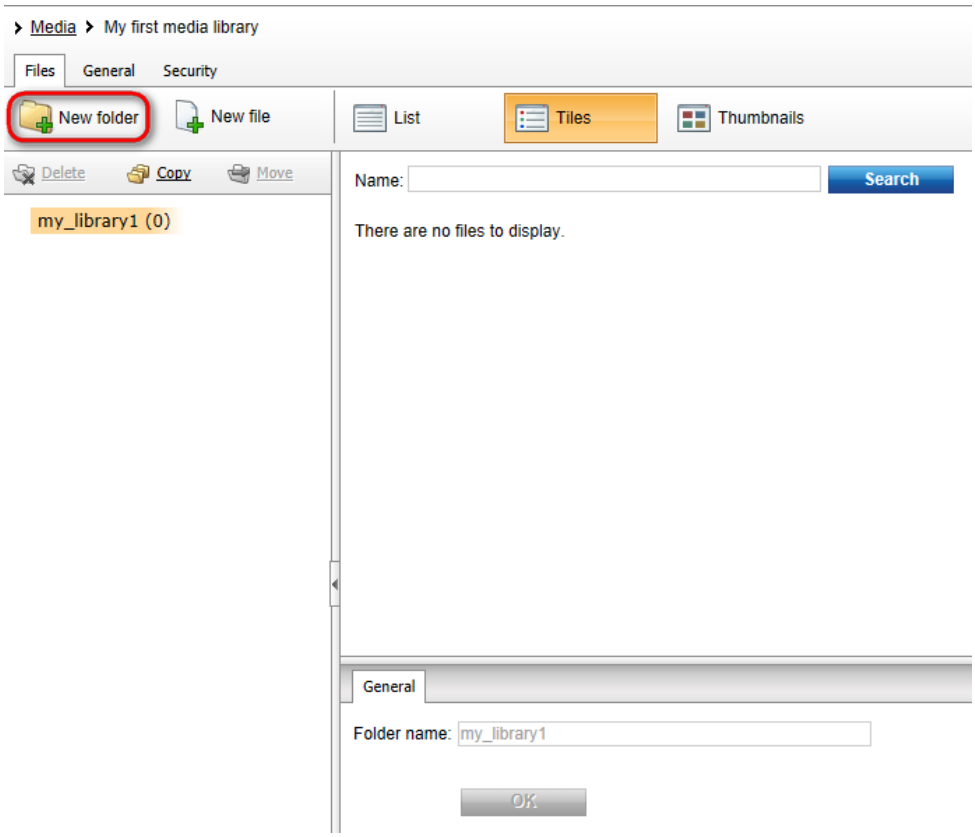
- **By means of the administration interface**
- **On the live site using the Media gallery or Media file uploader web parts**
- **Externally, directly into the file system - e.g. via FTP**

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.

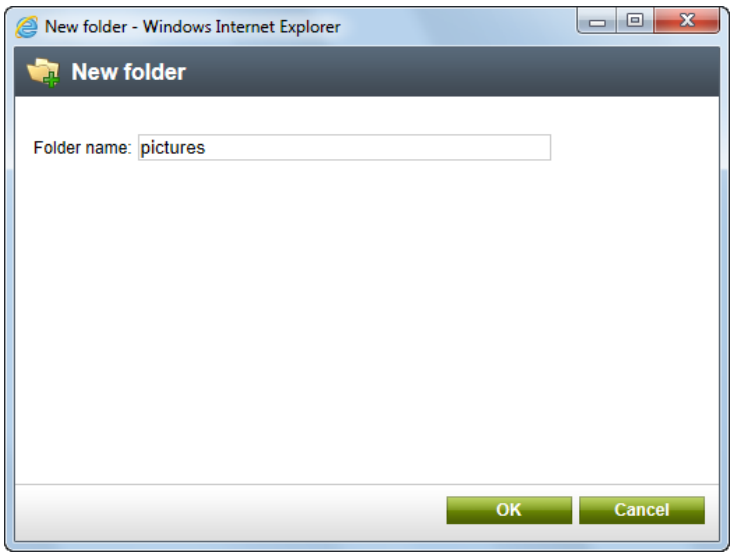
Uploading files via the administration interface

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.

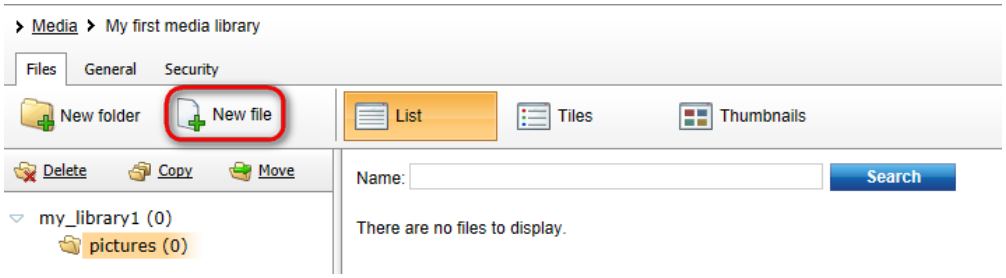
1. It is a good idea to keep your files organized in folders within the media library. To create a new folder, click the **New folder** button in the top left corner of the page.



2. Enter the name of the folder, e.g. *pictures*, and click **OK**.

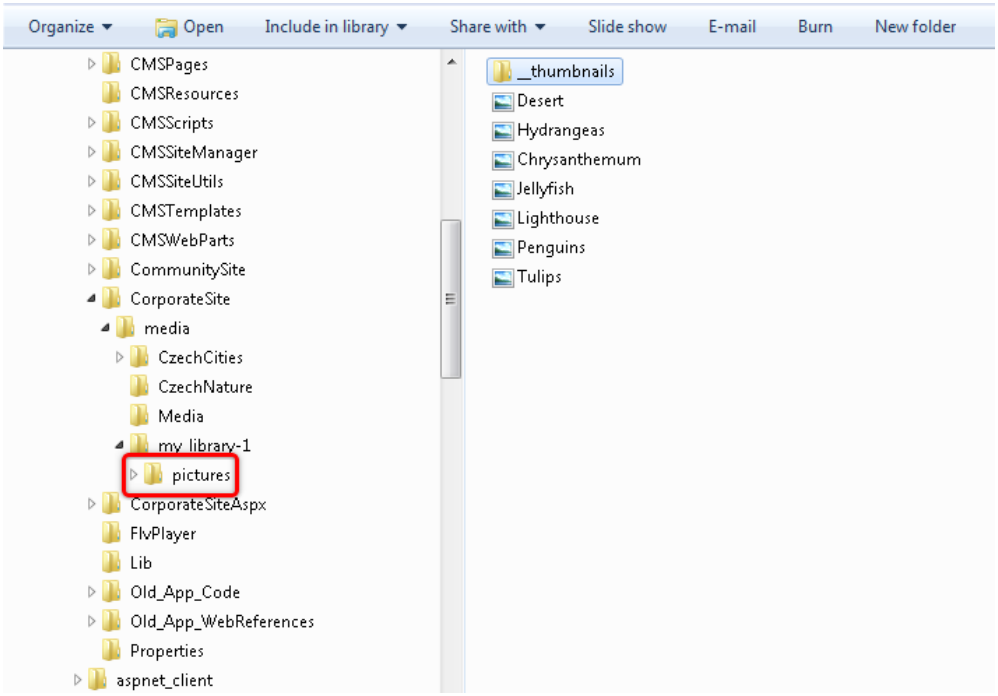


3. The folder is now created. You can upload files into the selected folder by clicking the **New file** button.



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.

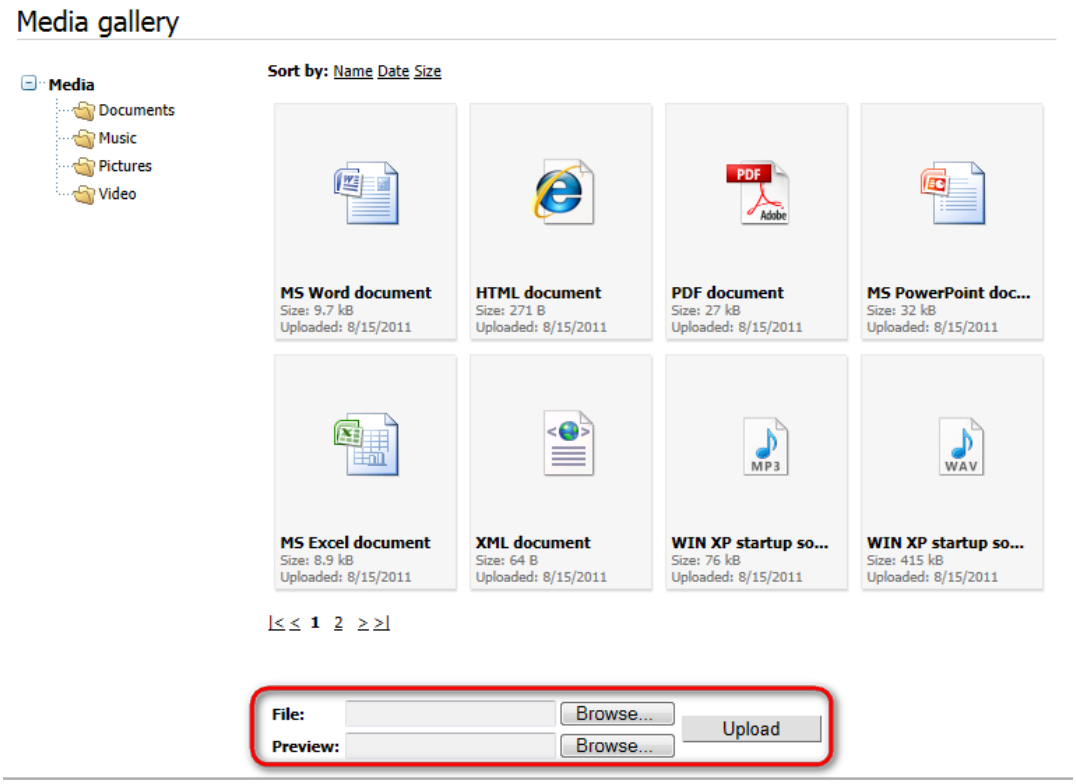


On-site upload via the Media gallery and Media file uploader web parts

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:

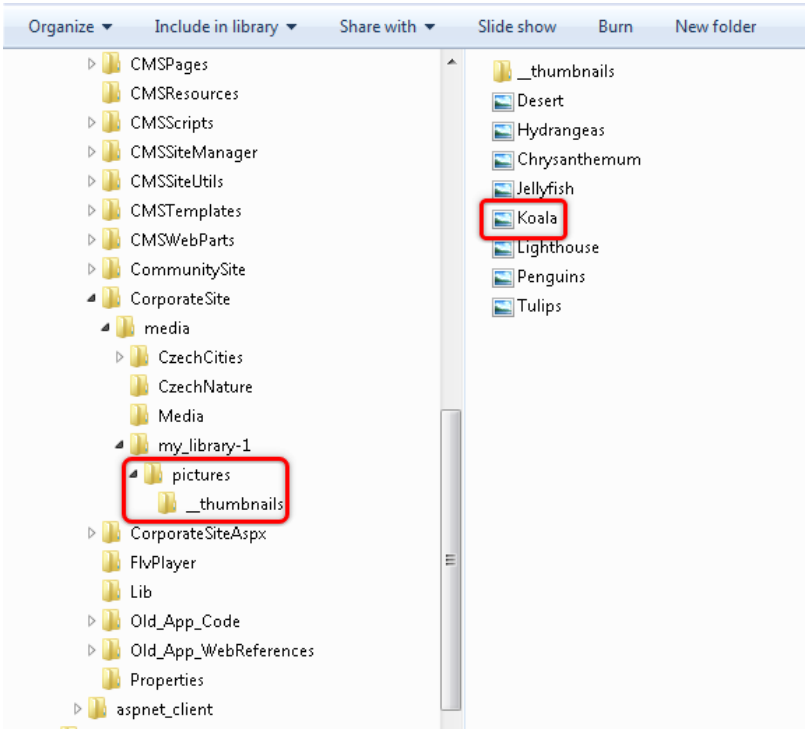
- **Enable upload thumbnail** - enables on-site upload of file thumbnails.

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:

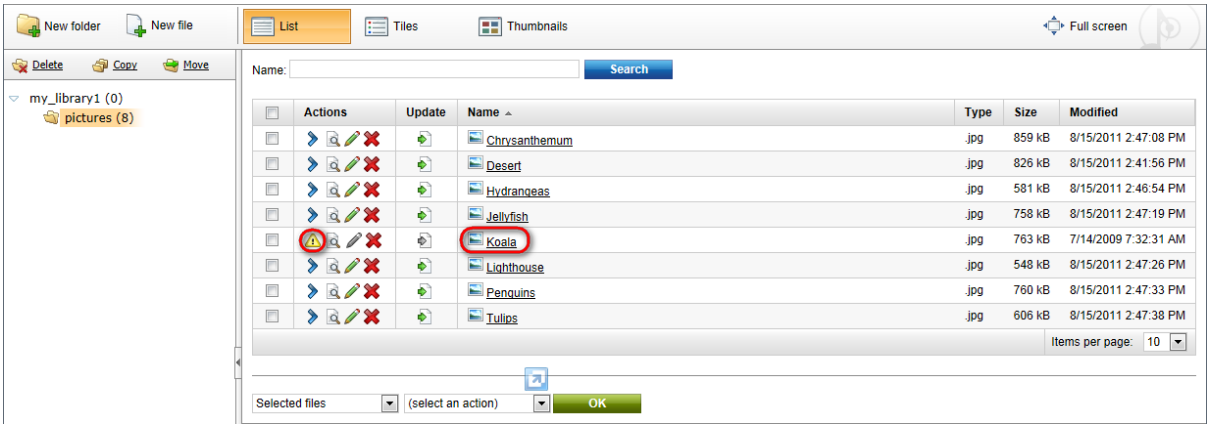
External upload

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.

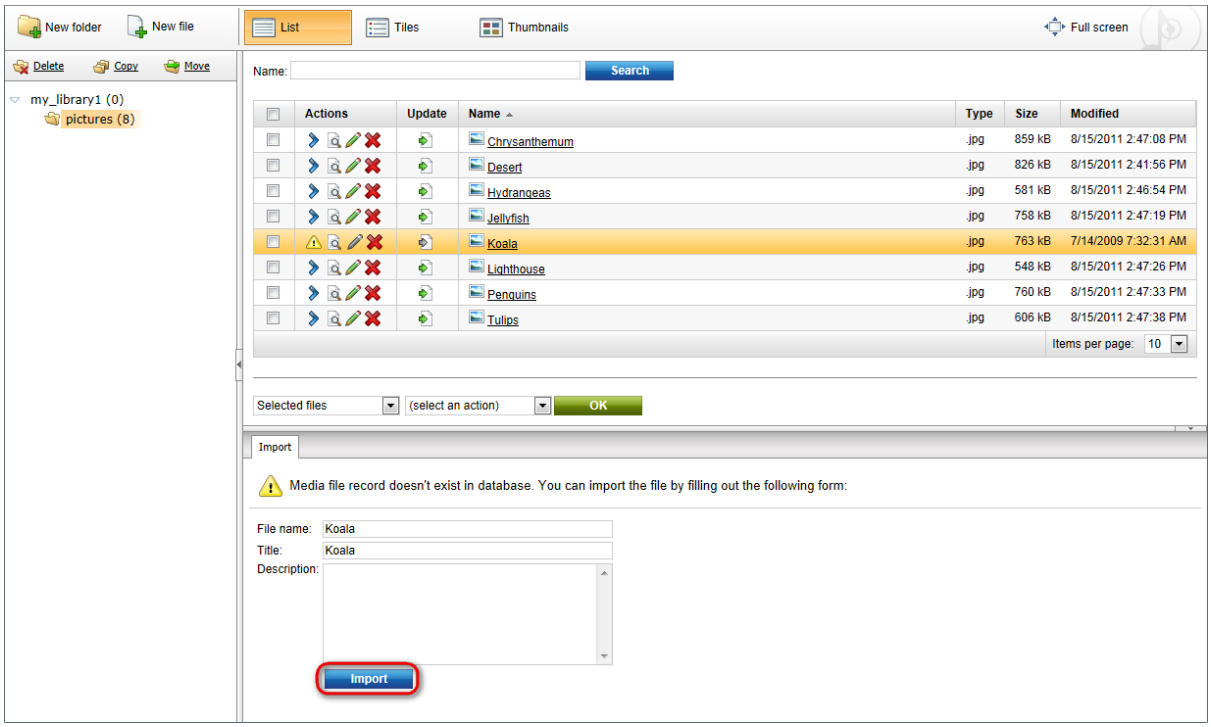
1. For the purposes of this example, try uploading another file into the **pictures** folder by copying it using Windows Explorer or any other file manager.



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.



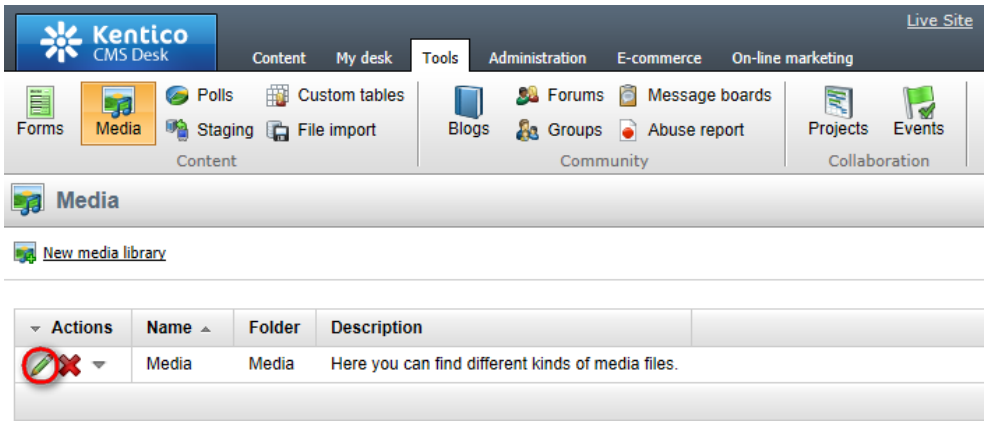
To register the file, you have to click the **Import** button, which creates the file's record in the database. Please note that the files are not physically copied into the database, they remain in the file system only and a record linking to the physical file is created in the database.



Multiple files can also be imported conveniently as a batch. See Batch operations in the **Files and folders management** topic for more details.

8.30.4.3 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:

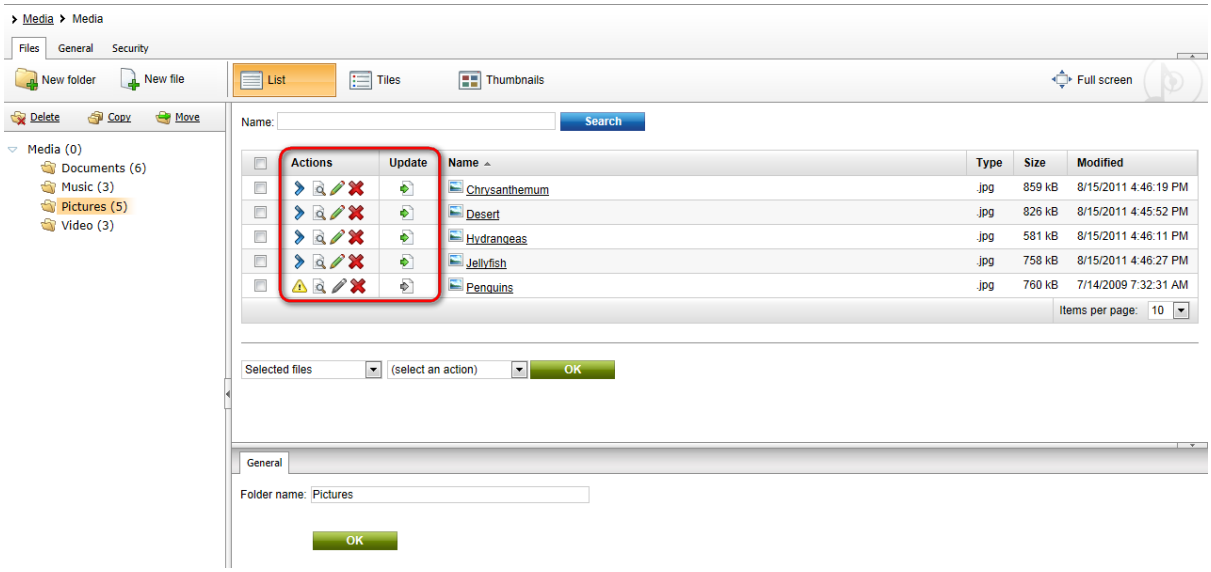
- **New folder** - creates a new folder under the currently selected folder.

- **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

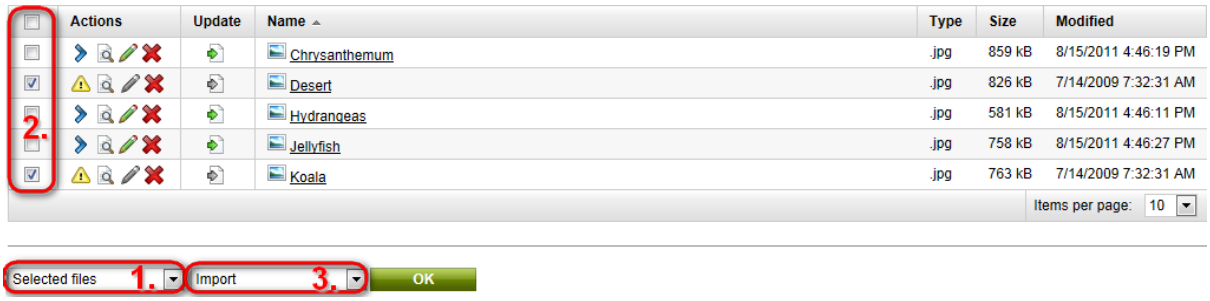
You can also perform the **Copy**, **Move**, **Delete** and **Import** operations for multiple files at once. To perform a batch operation, take the following steps:

1. Use the first drop-down list (**1.** in the screenshot) to select if you want to perform the operation for:

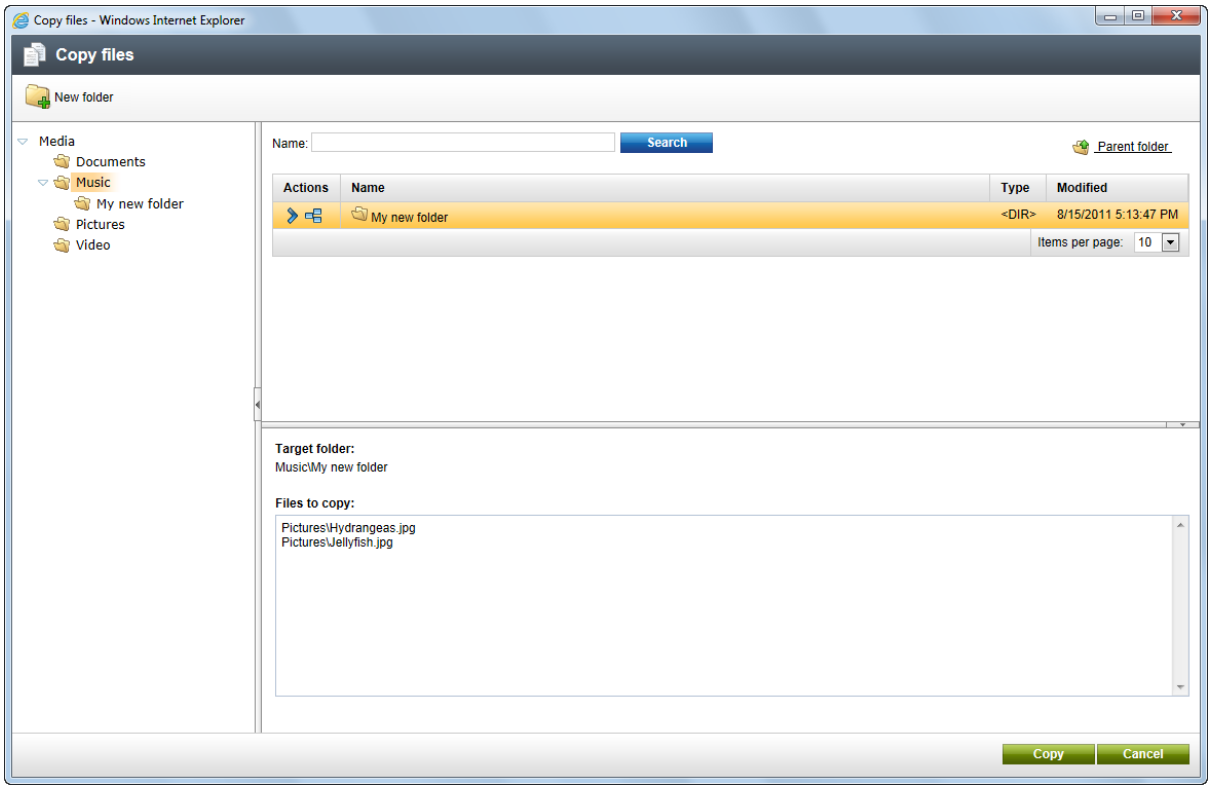
- **All files** - performs the operation for all files in the current folder.
- **Selected files** - performs the operation only for files selected by the check-boxes ( ).

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.

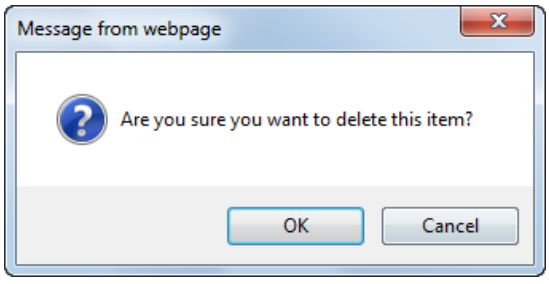
3. Choose the operation to perform using the second drop-down list (**3.** in the screenshot) and click **OK**.



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.

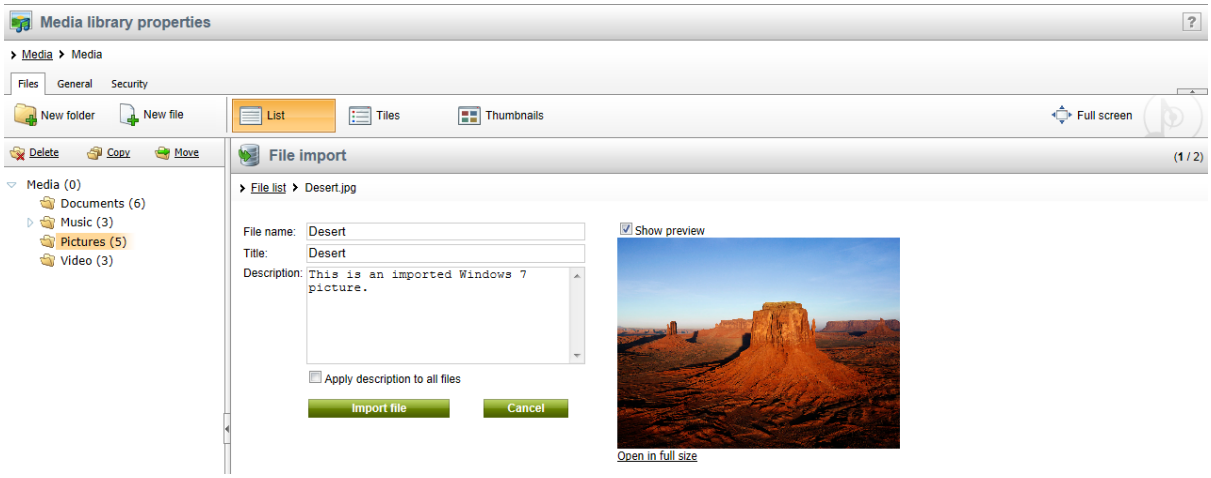


4.c If you are performing a batch **Import** operation, the dialog as in the following screenshot will appear for each imported file. Please specify the following details for each imported file:

- **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.

8.30.4.4 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.

8.30.4.5 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.

8.30.4.6 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*.

8.30.5 Displaying files from media library

8.30.5.1 Overview

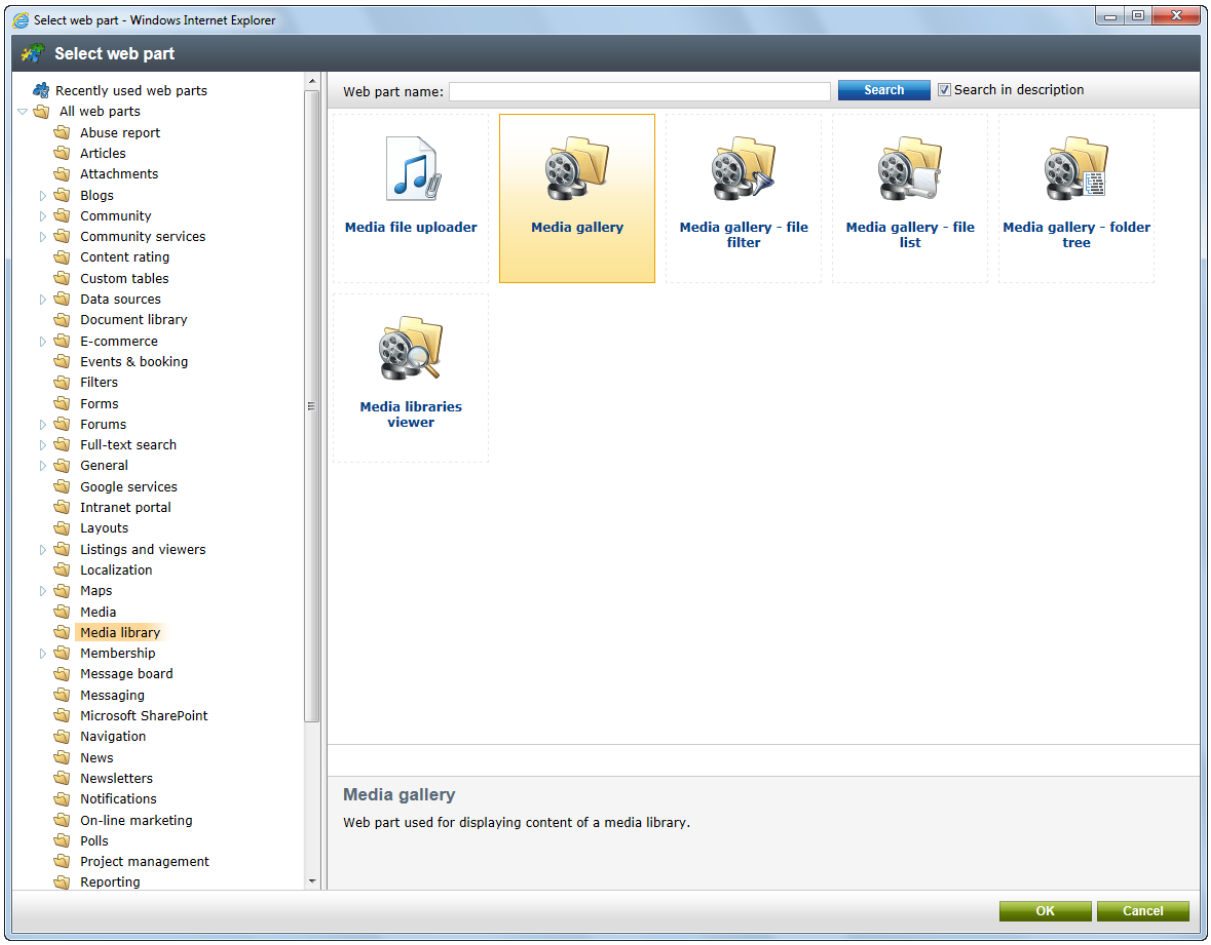
In this subchapter you will learn how to display files from a media library.

- 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.

8.30.5.2 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**.

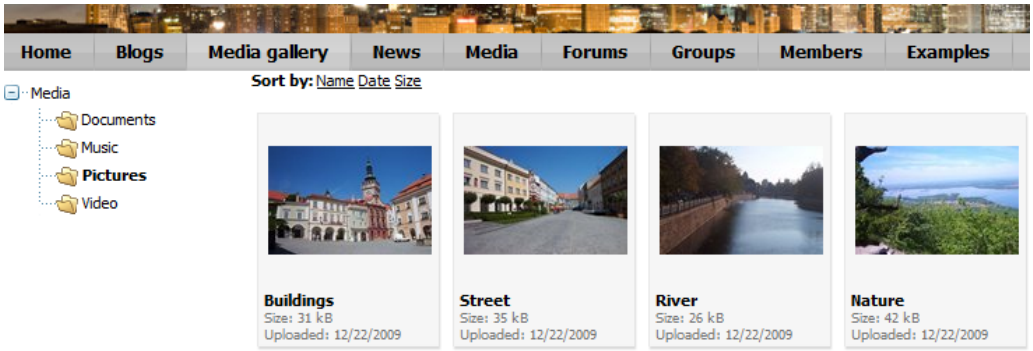


2. In the web part properties window, you only have to choose the media library that you want to display:

- **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.



8.30.5.3 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).

8.30.5.4 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.

8.30.6 Settings

8.30.6.1 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.

8.30.6.2 Media library settings

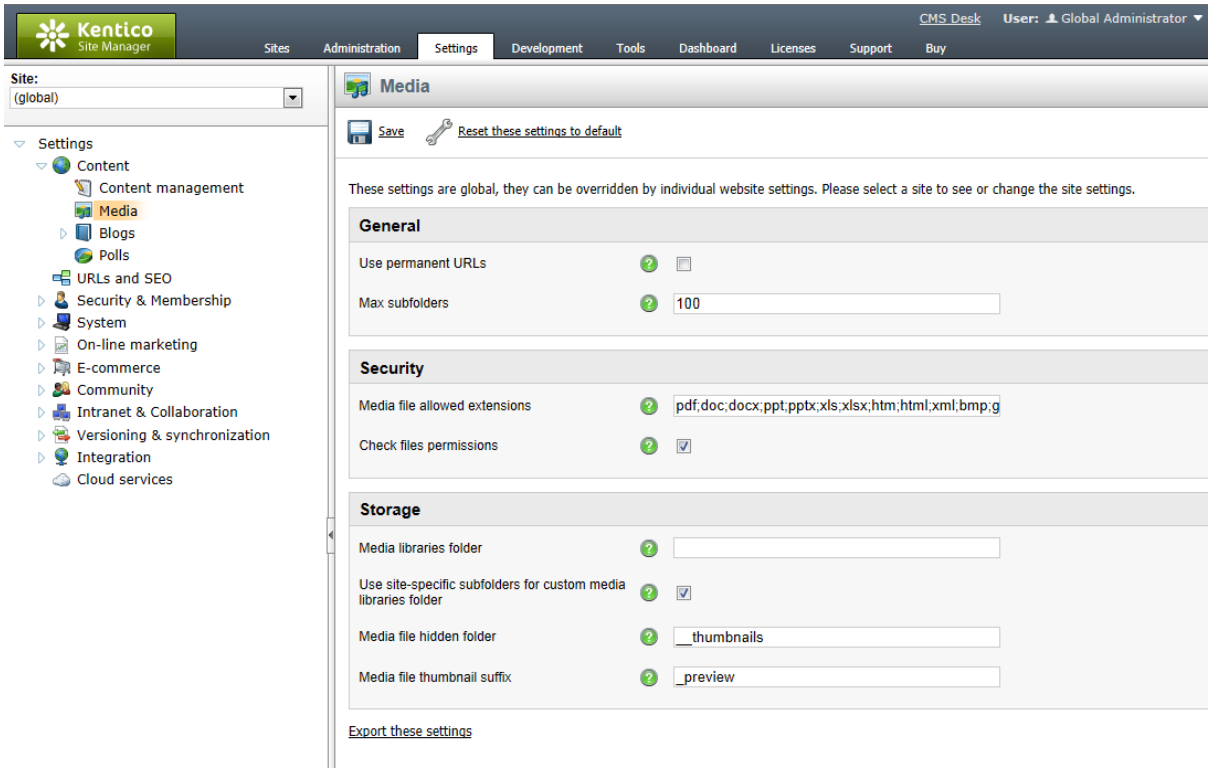
Settings related to the Media libraries module are located in **Site Manager -> Settings -> Content -> Media**. The following settings are related to the module:

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 maximal 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.



8.30.6.3 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"/>
```

8.30.6.4 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 libraries 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.

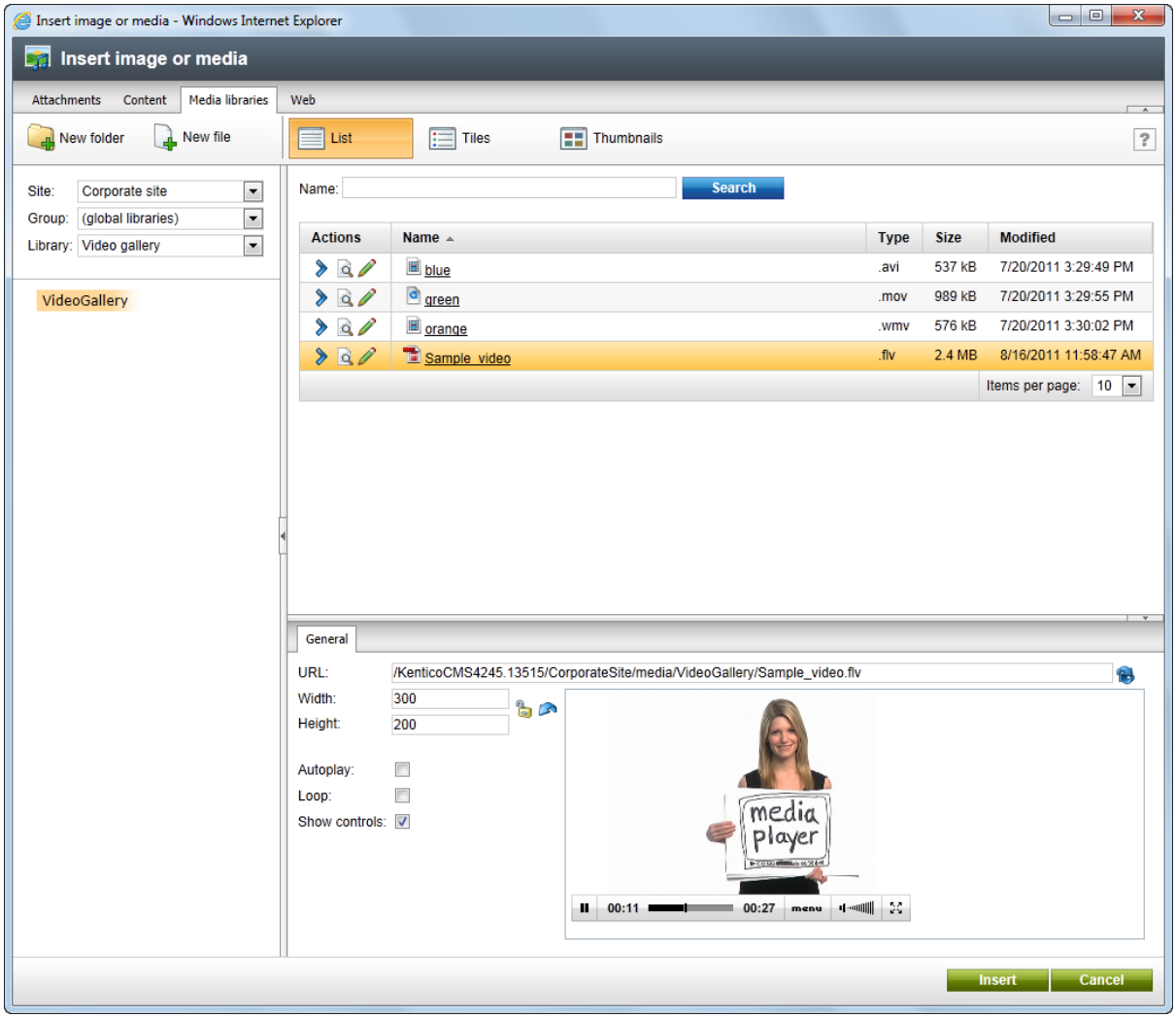
```
private string CreateFlvObject(string url)
{
    string playerID = Guid.NewGuid().ToString("N");
    string flvUrl = URLHelper.GetAbsoluteUrl(url)
                .Replace("?", "%3F")
                .Replace("=", "%3D")
                .Replace("&", "%26");

    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.



8.30.6.5 Configuring maximal uploaded file size

The default maximal 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. Besides, you also 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>
```

8.30.7 Security

8.30.7.1 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.

8.30.7.2 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:

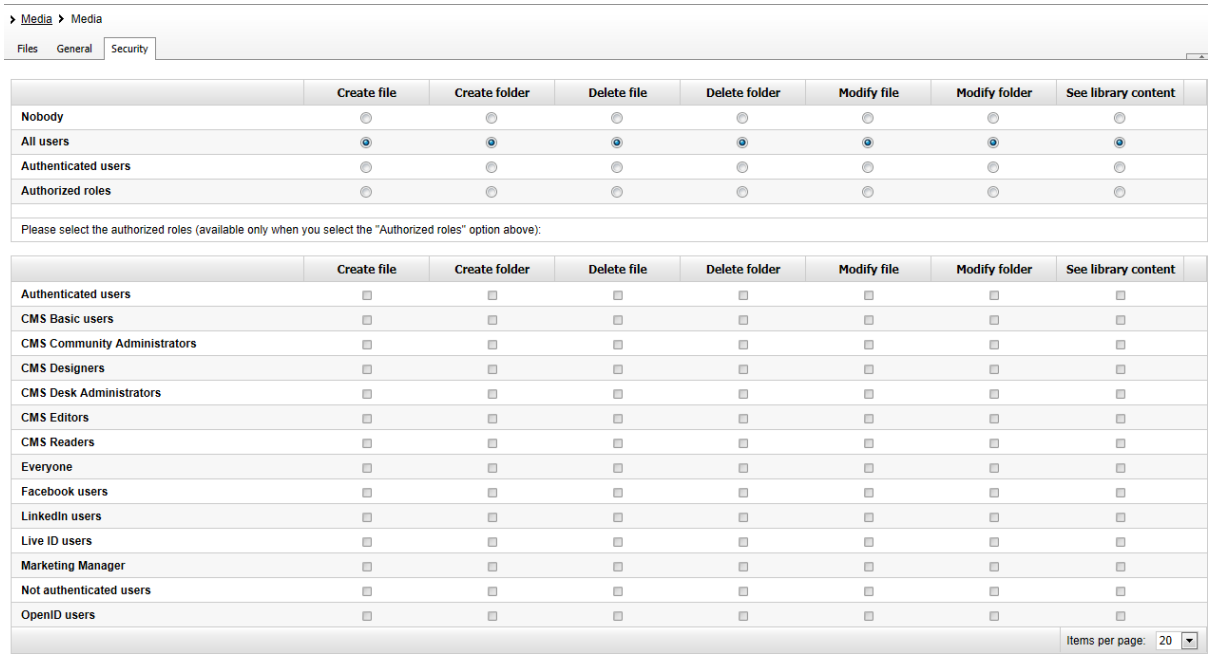
- **Manage** - allows to create, delete and edit media libraries and manage media library content via the administration interface.
- **Read** - allows seeing media library content and its properties when editing a media library, but does not allow to make any changes to it.
- **Destroy** - allows to destroy 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

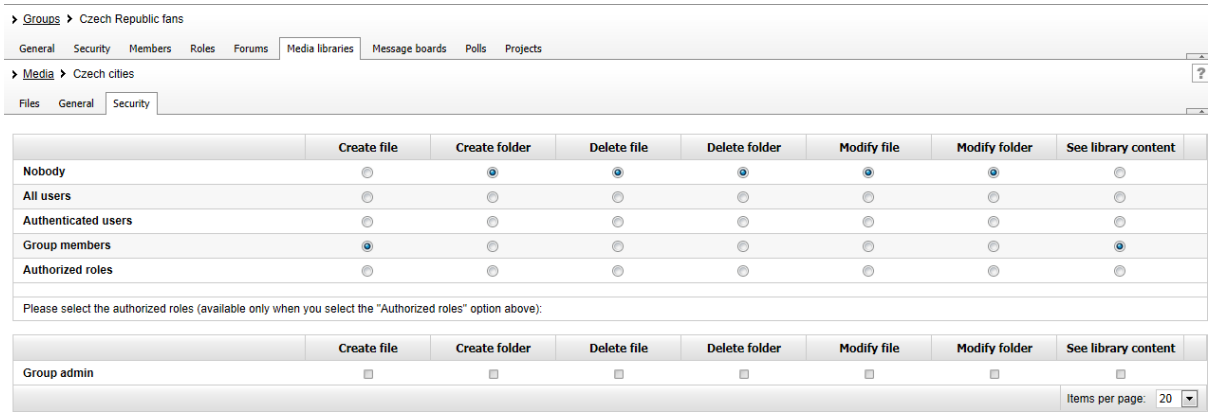
On the **Security** tab of each media library, you can assign permissions for particular actions. This can be useful if settings for the Media library module permissions are not sufficient for your needs and you want to restrict users from performing certain actions in certain media libraries. If this 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.

8.30.7.3 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

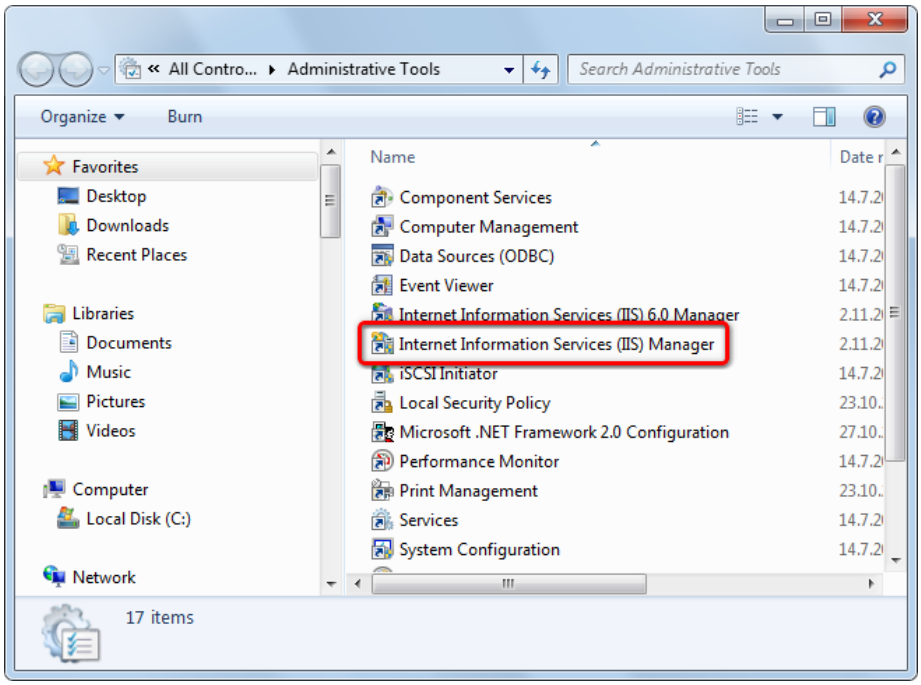
Secured media libraries allow viewing of their content only to members of authorized roles or only to authenticated users, based on the settings made on the library's **Security** tab. Secured libraries are also slower than the non-secured ones as permission checking involves some processing overhead.

To set up a media library to behave as a secured library, you have to take the following steps:

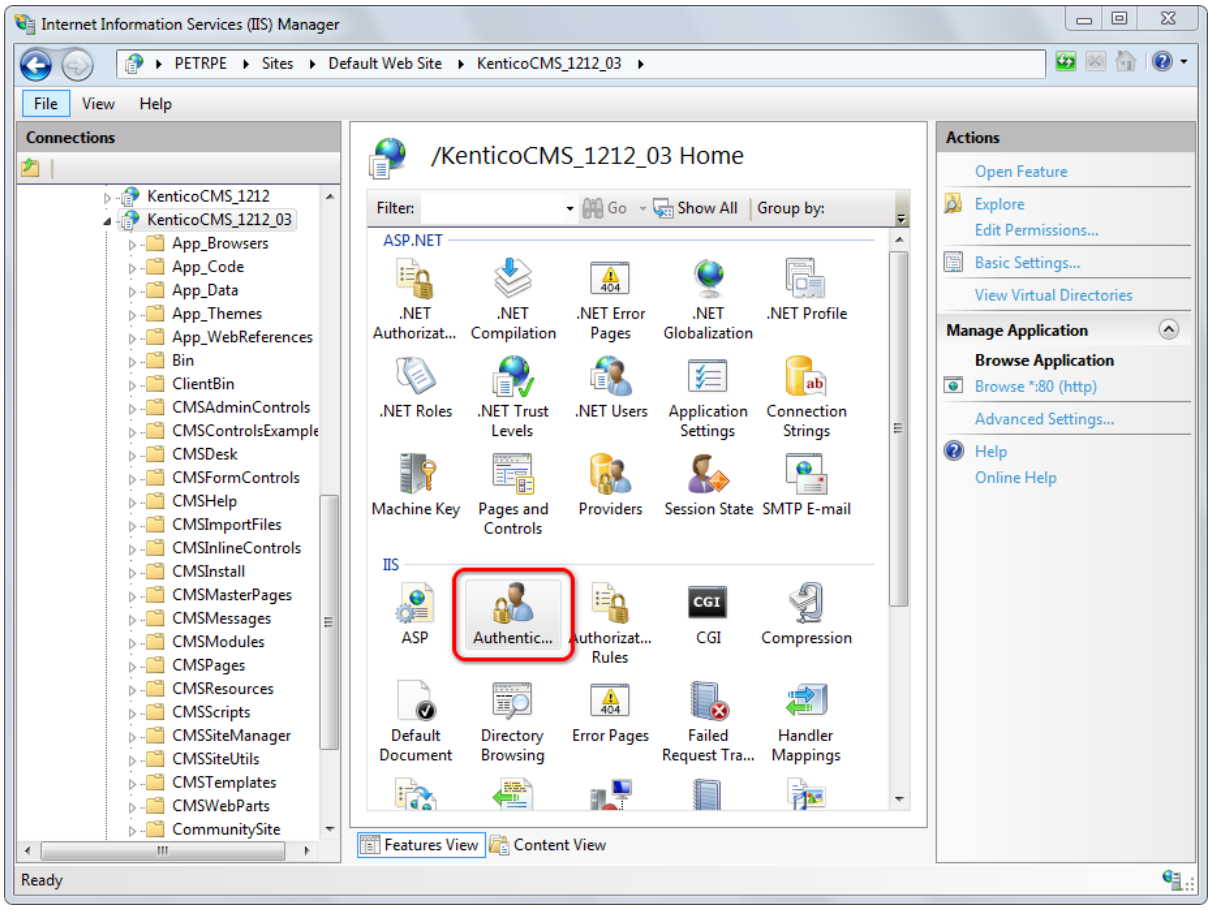
1. IIS setup

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.

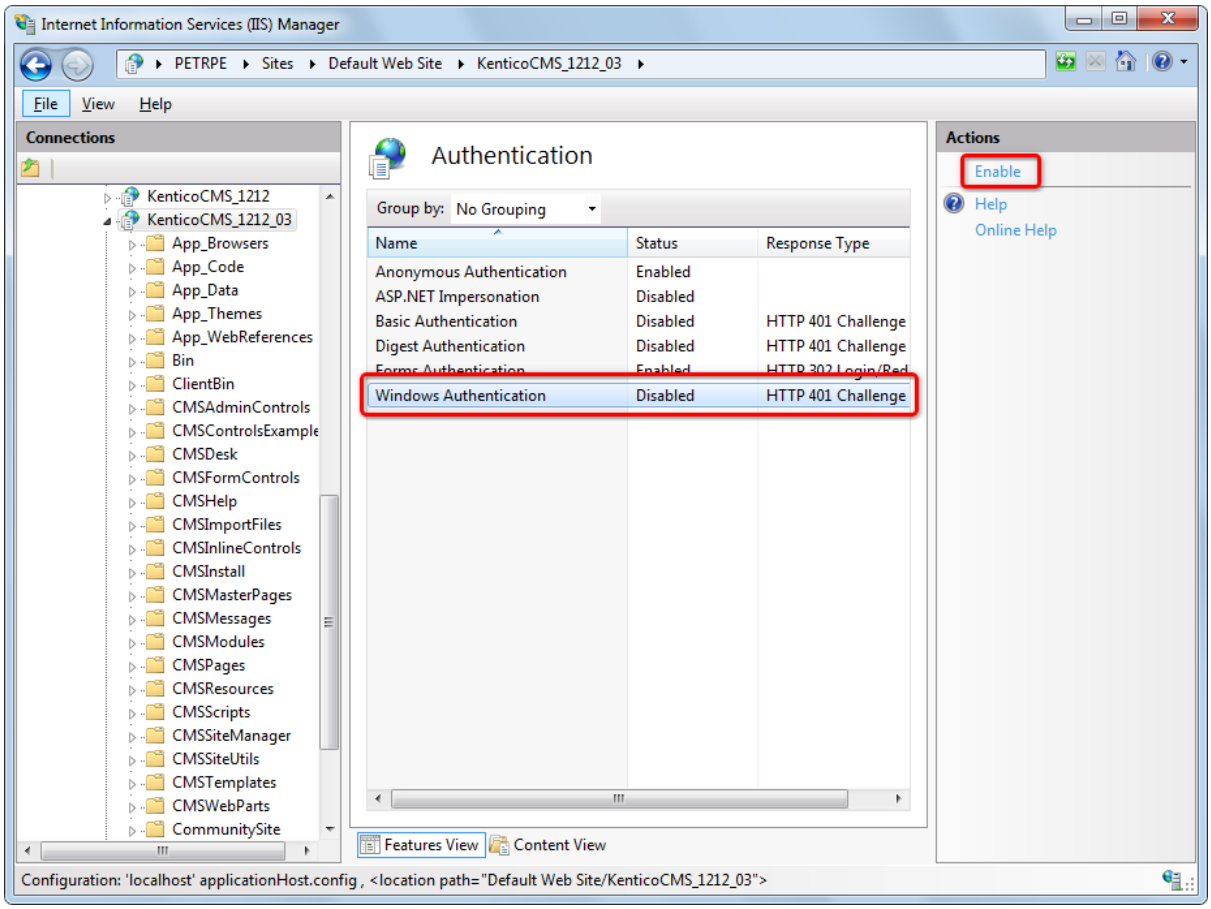
a) Go to **Start** -> **Control Panel** -> **Administrative Tools** and start the **Internet Information Services (IIS) Manager**.



b) Locate and select the media library folder in the IIS tree, then click on the **Authentication** icon.

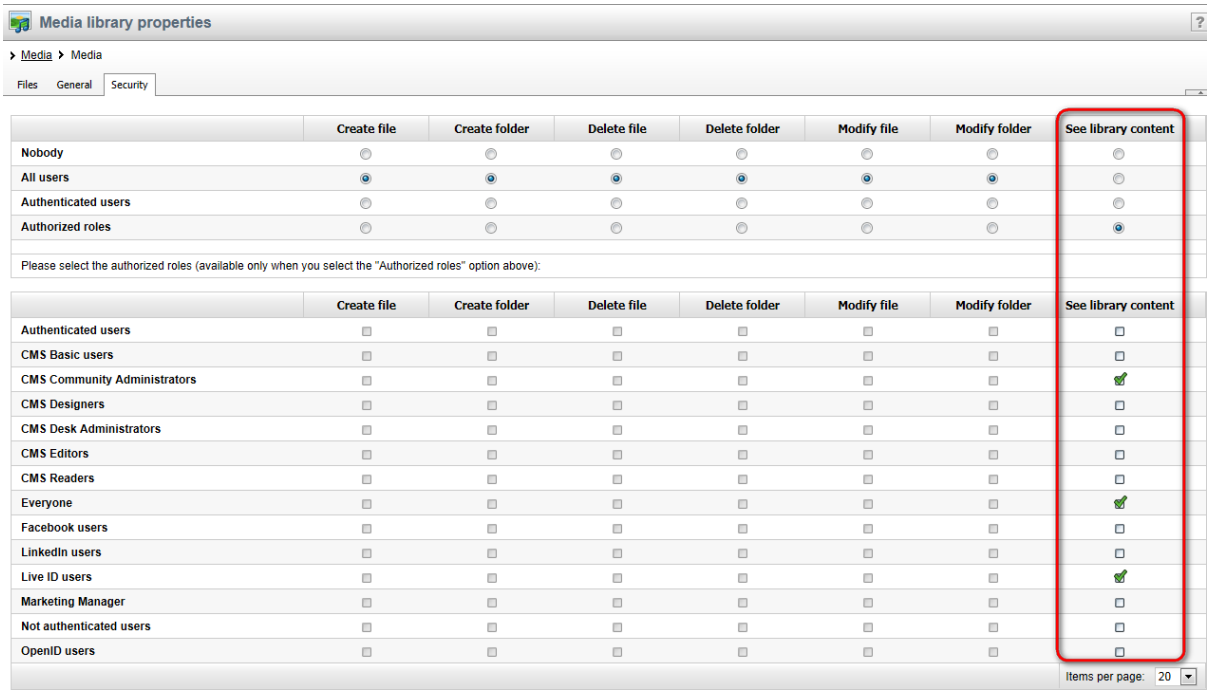


c) Disable the **Anonymous Authentication** by clicking the **Disable** link in the **Actions** list. If enabled, disable also the **Windows Authentication**.



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.

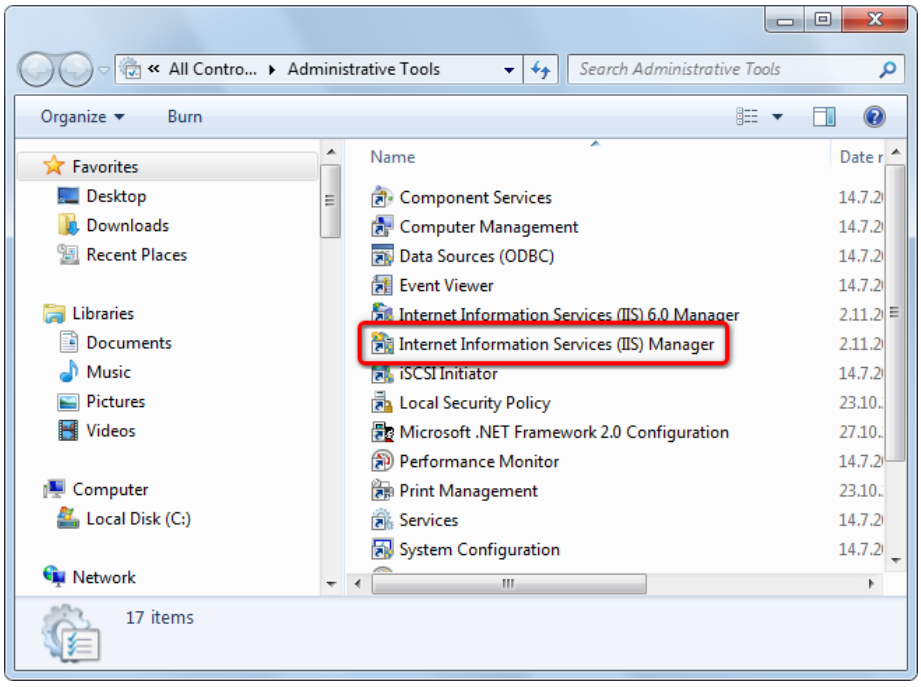
Non-secured libraries

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.

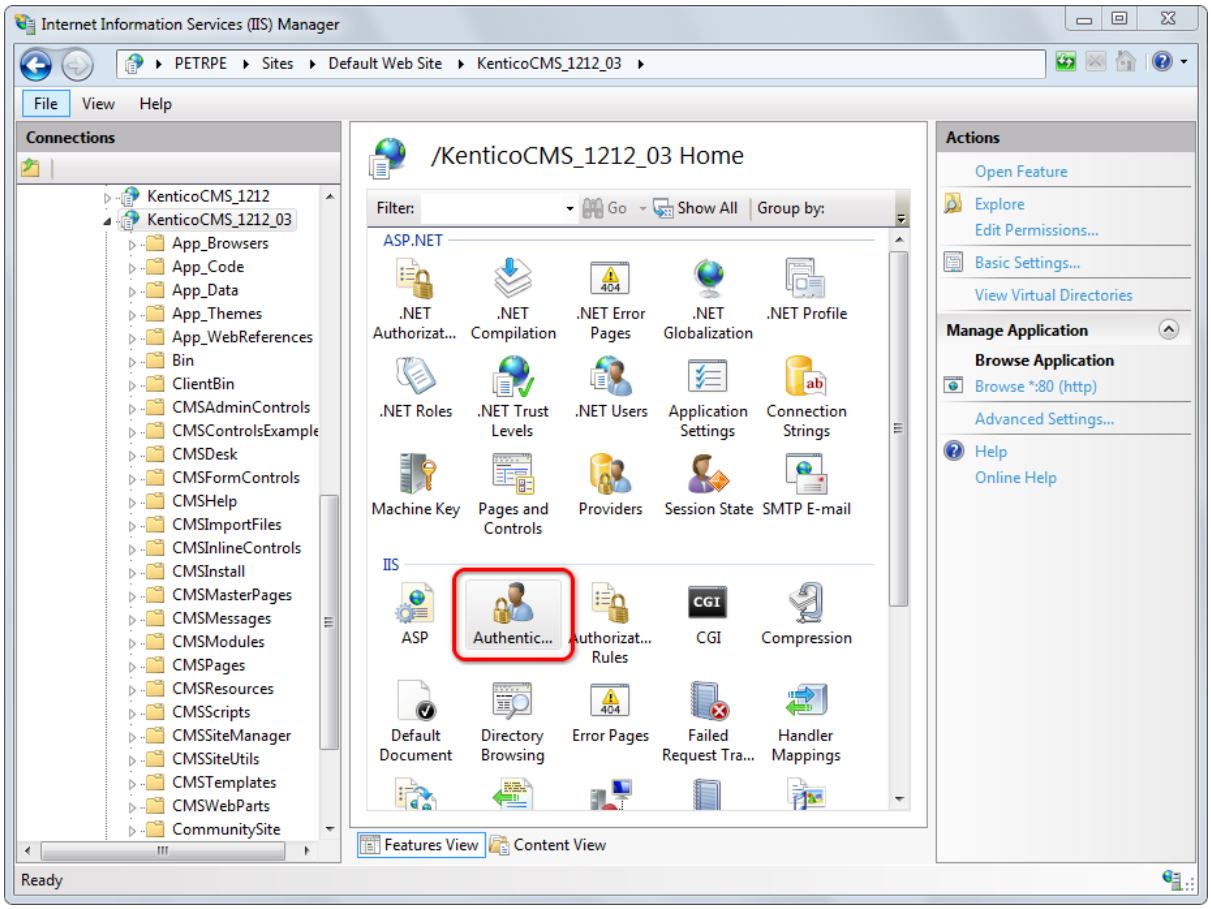
1. IIS setup

You have to **set up your IIS** so that files in the library can be accessed directly by anonymous users.

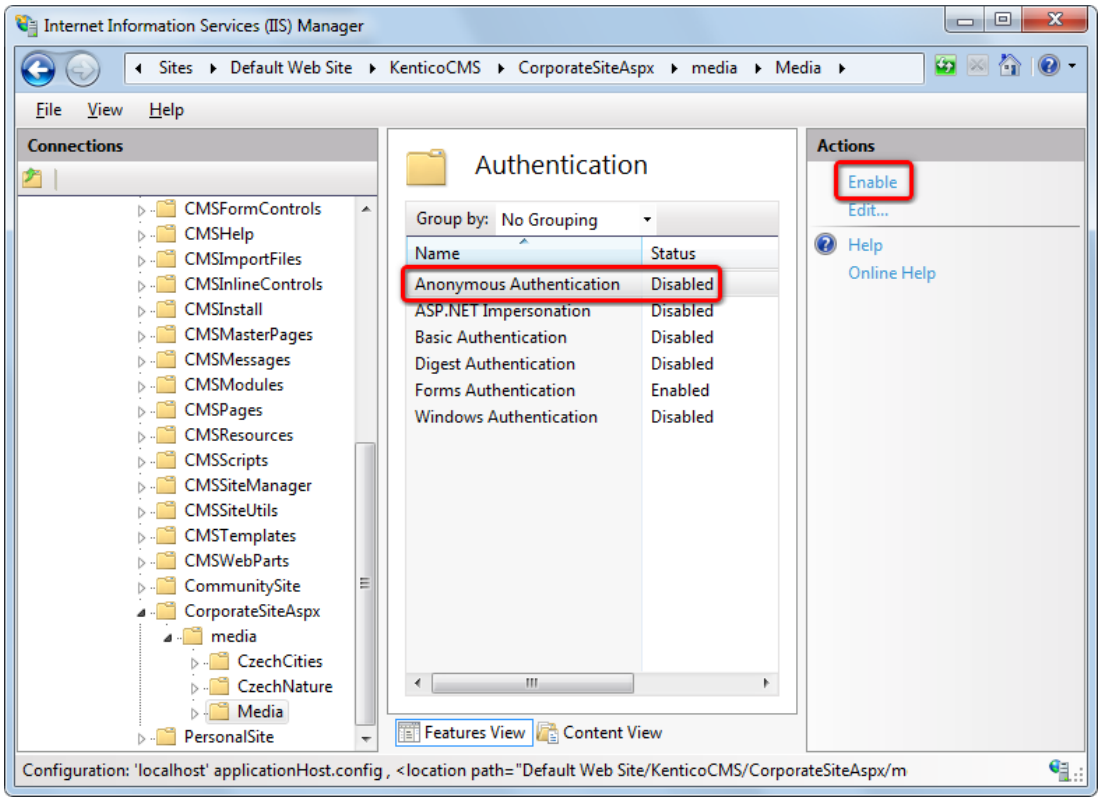
a) Go to **Start** -> **Control Panel** -> **Administrative Tools** and start the **Internet Information Services (IIS) Manager**.



b) Locate and select the media library folder in the IIS tree, then click on the **Authentication** icon.

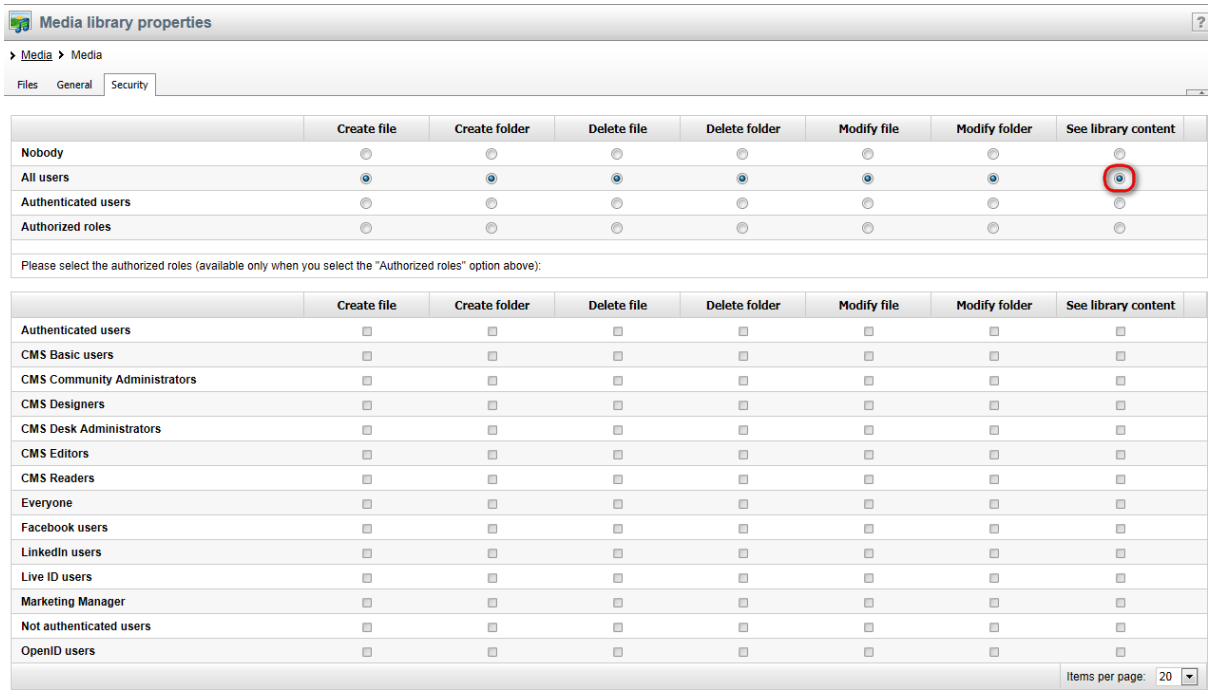


c) Enable the **Anonymous Authentication** by clicking the **Enable** link in the **Actions** list.

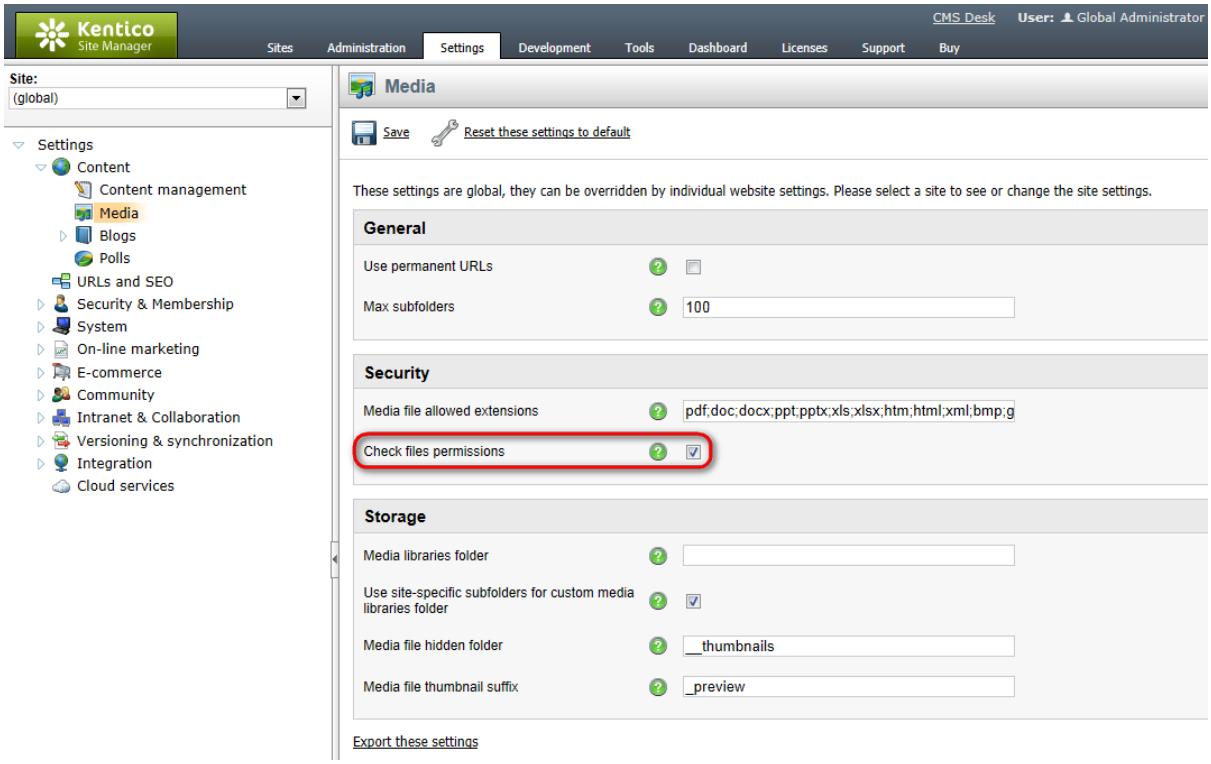


2. Media library security settings

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.

8.30.8 Media internals and API

8.30.8.1 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.

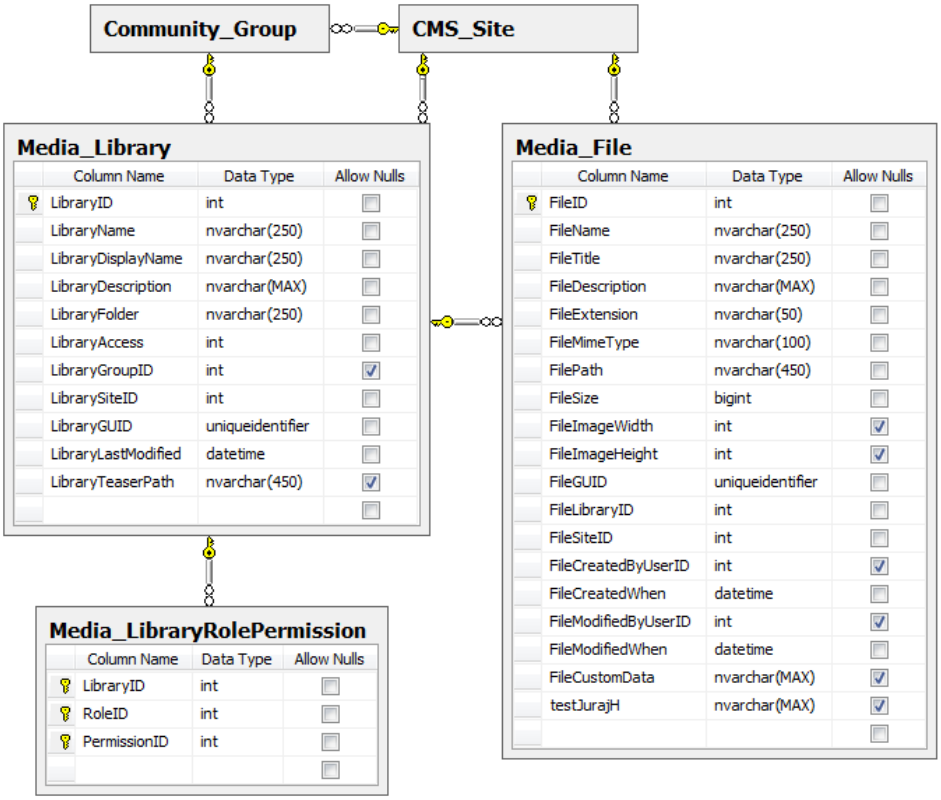
Advanced API examples can be found in the API programming and Kentico CMS internals section of this

guide. For detailed API documentation, such as a list of all methods from the module classes, please refer to Kentico CMS API Reference.

8.30.8.2 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.



8.30.8.3 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.

8.30.8.4 API examples

8.30.8.4.1 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.

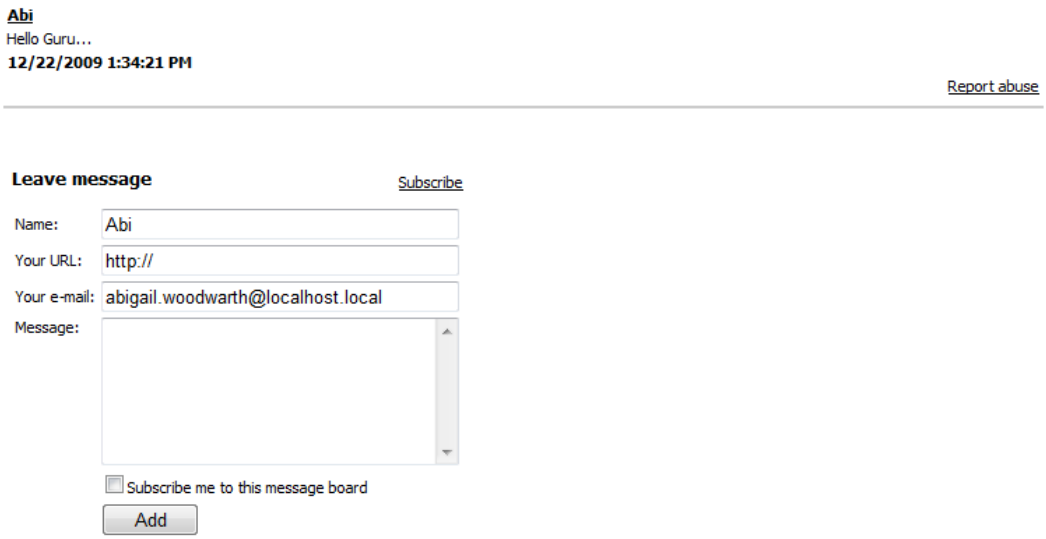
If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Tools\MediaLibrary\Default.aspx.cs***.

8.31 Message boards

8.31.1 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.

8.31.2 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* with blank password. From the content tree, select **News -> New consulting services**.

2. Click the **Add web part** ( ) icon at the top right corner of **zoneLeft** web part zone and in the web part selection window, 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

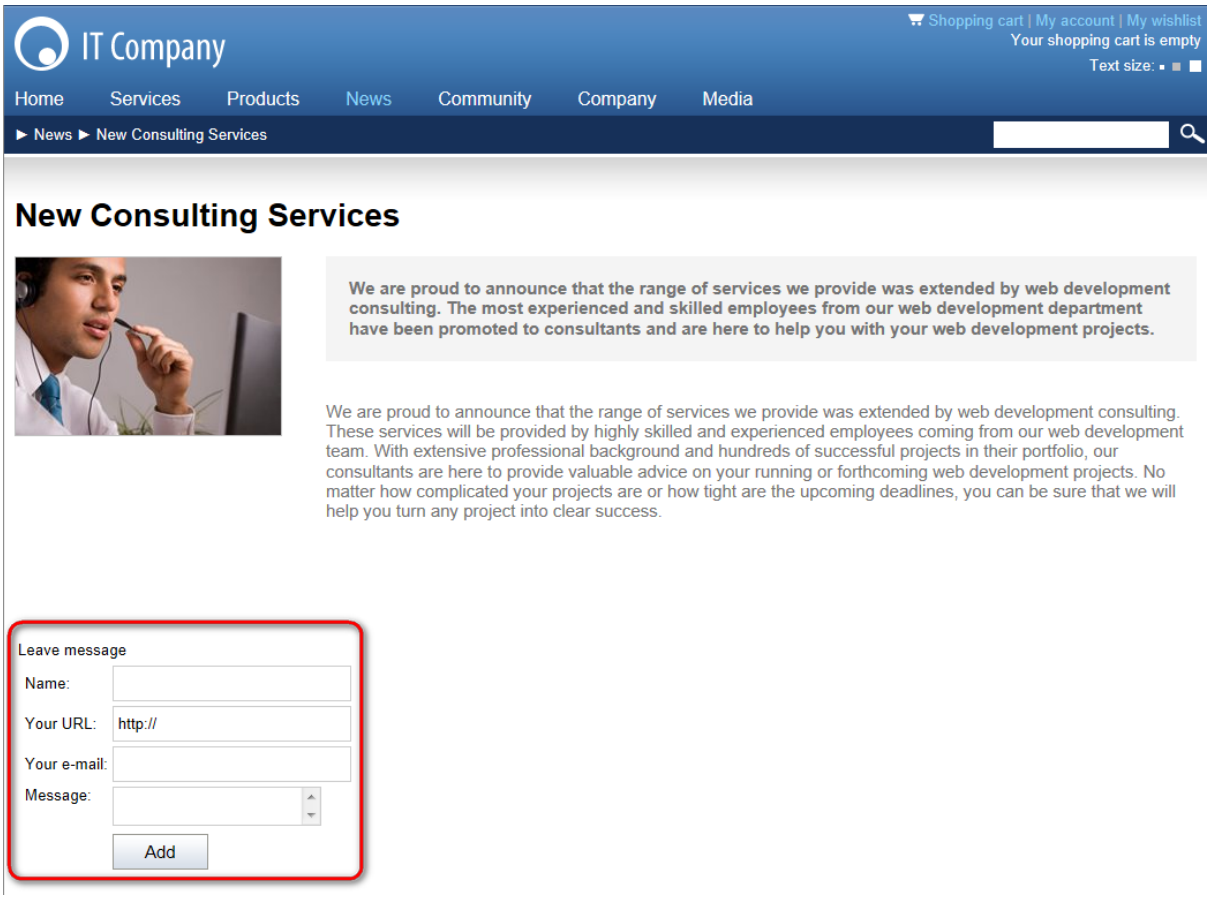
messages
- **No messages text** - text message that will be displayed to site visitors when there are no messages in the board
- **Display name** - display name of the message board, it will be used in the administration interface to identify the message board, so it is advisable to use some well-descriptive 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 !!

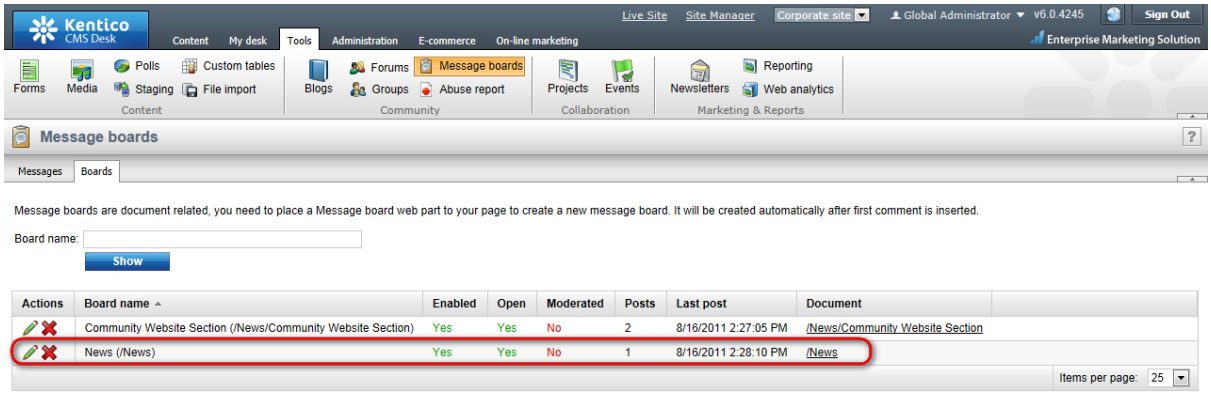
Properties in the **New board settings** section will be used when the message board is created (after first message or subscription). Thereafter, changes made to the web part properties will take no effect. To change some of these properties, you will have to edit the message board in **CMS Desk -> 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.



8.31.3 Administrating message boards

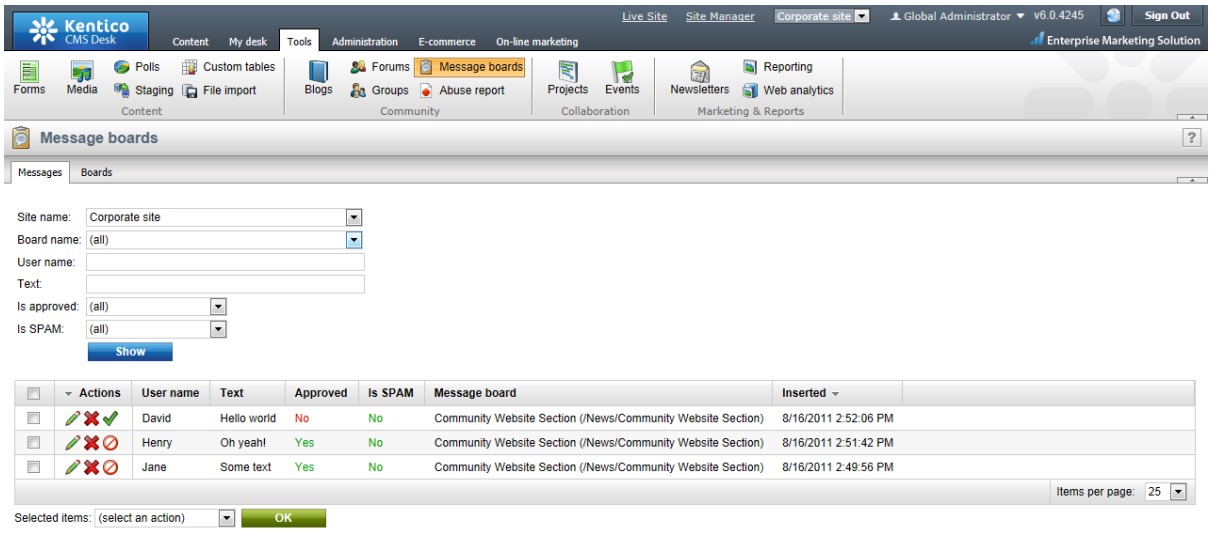
Message boards administration can be performed in **CMS Desk -> Tools -> Message boards**. The section is divided into two tabs.

Messages tab

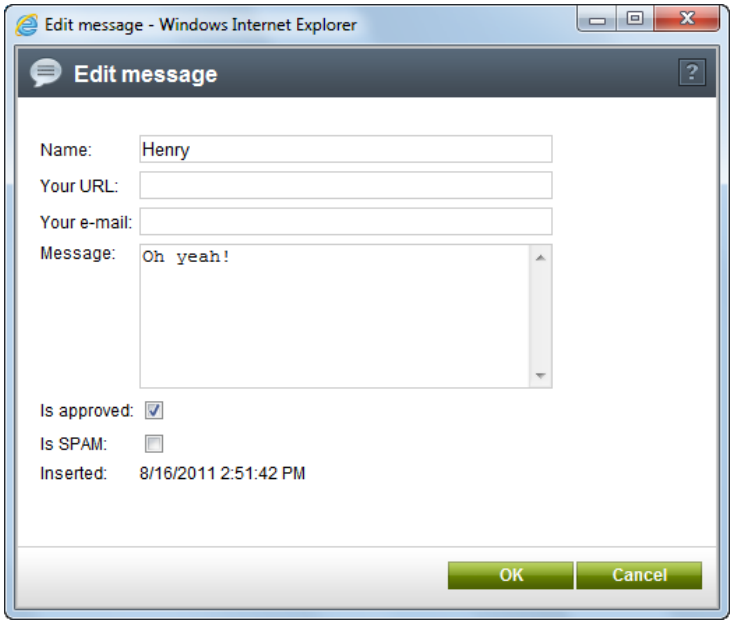
On this tab, moderators can manage board messages. By default, there are only messages requiring approval displayed when you access the page, so that the moderator sees only new messages that need to be approved or rejected. Until the messages are approved, they won't be displayed on the message board. 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



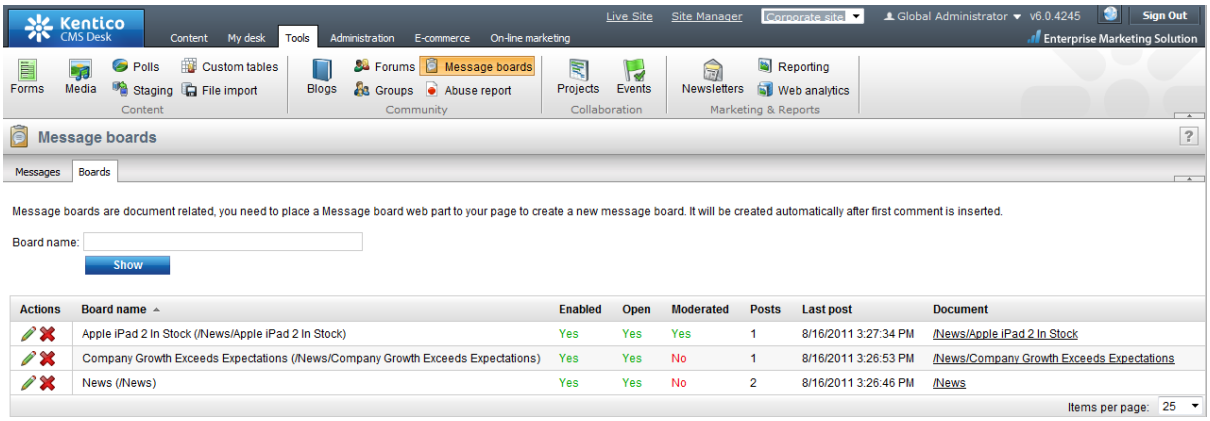
Messages in the list can be **Approved** (✔), **Rejected** (⊘), **Deleted** (✖) or **Edited** (✎). If you choose to edit a message, the following window pops up, letting you make changes to it. You can also select more messages using the check-boxes and perform one of these actions for all of them using 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.



8.31.4 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!

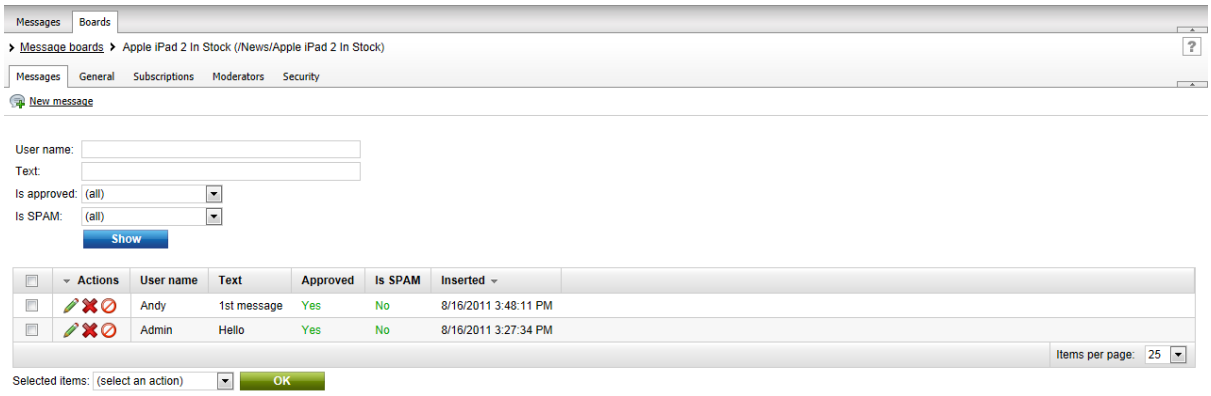
The administration interface for editing message boards is divided into five tabs:

Messages tab

This tab displays a list of all messages on the message board. Using the **New message** link, 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

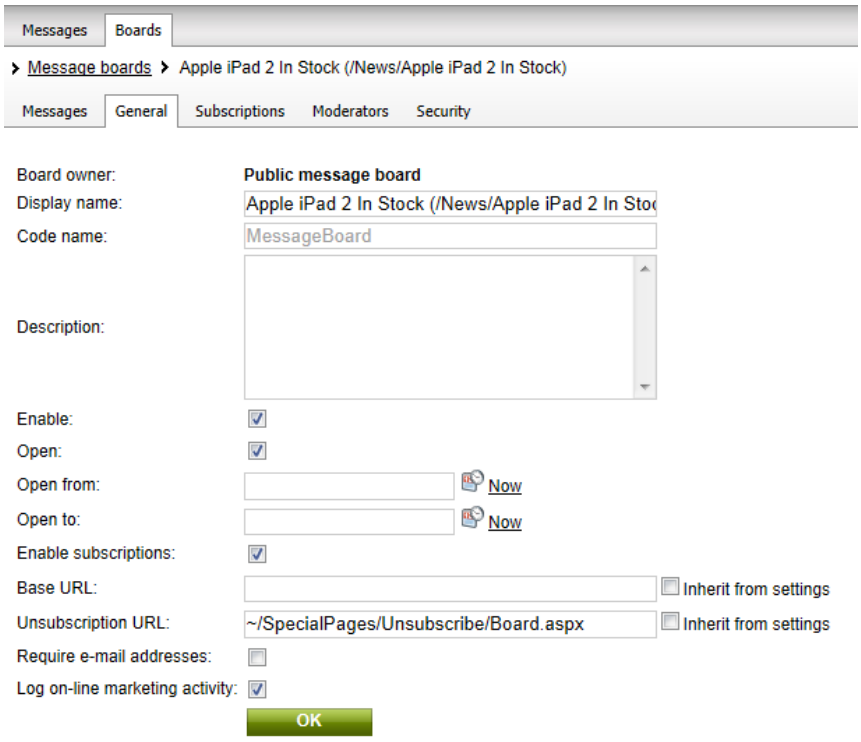
Messages can be **Edited** (🖉), **Deleted** (✖), **Approved** (✔) or **Rejected** (⊘). You can also select more messages using the check-boxes and perform one of these actions for all of them using the **Selected items** drop-down list and clicking the **OK** button.



General tab

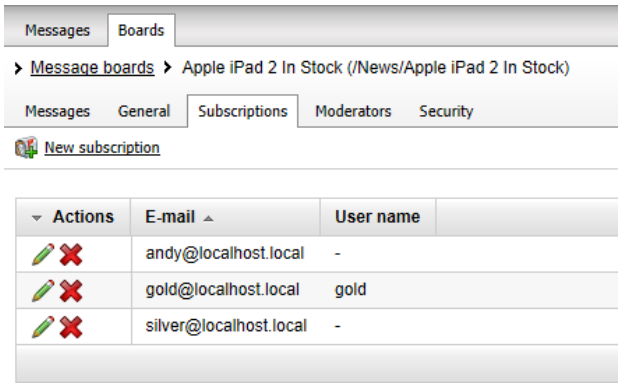
On the general tab, you can specify the following properties of the message board:

- **Display name** - display name of the message board
- **Code name** - code name of the message board
- **Description** - text describing the message board
- **Enable** - if unchecked, the message board will be hidden; if checked, the message board works normally
- **Open** - if checked, adding messages is enabled; if unchecked, messages are displayed but cannot be added
- **Open from / to** - using these fields, you can define the time interval during that new messages can be added to the board
- **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 property is empty too, 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 not set, 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



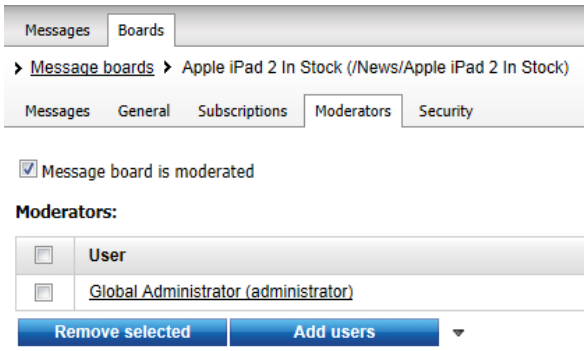
Subscriptions tab

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** link. 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.

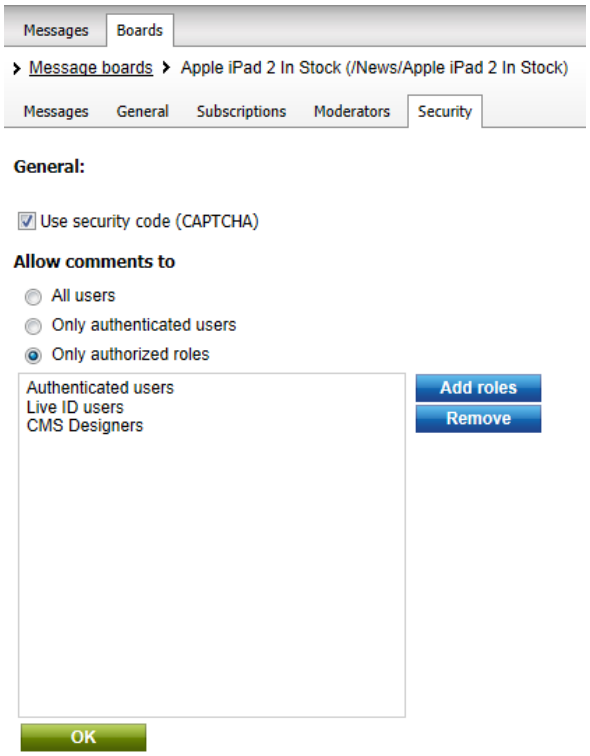
If the **Use security code (CAPTCHA)** check-box is checked, users will have to retype the CAPTCHA security code before adding a new message.

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



8.31.5 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:

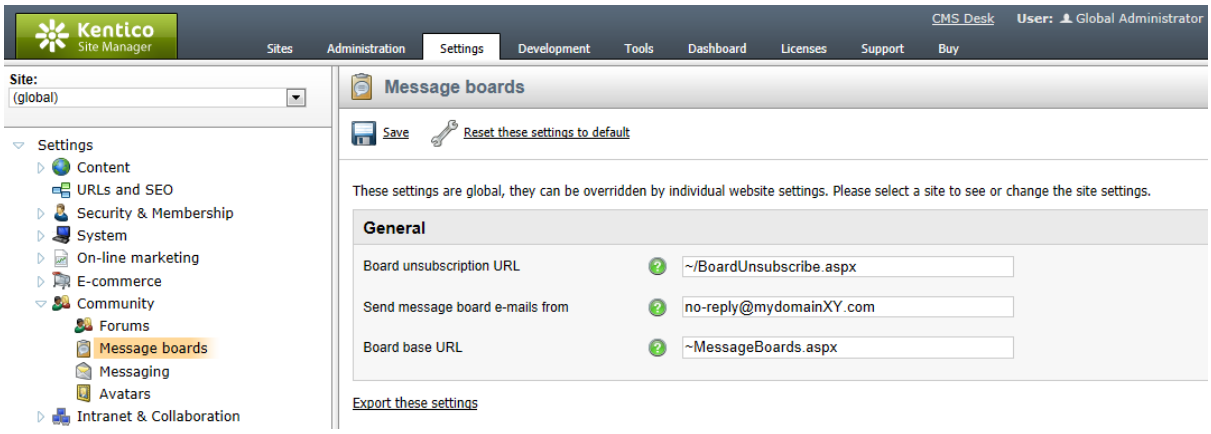
```
~/Groups/Profile.aspx
```

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.

8.31.6 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:

- **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
- **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



8.31.7 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 first message or subscription), 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:

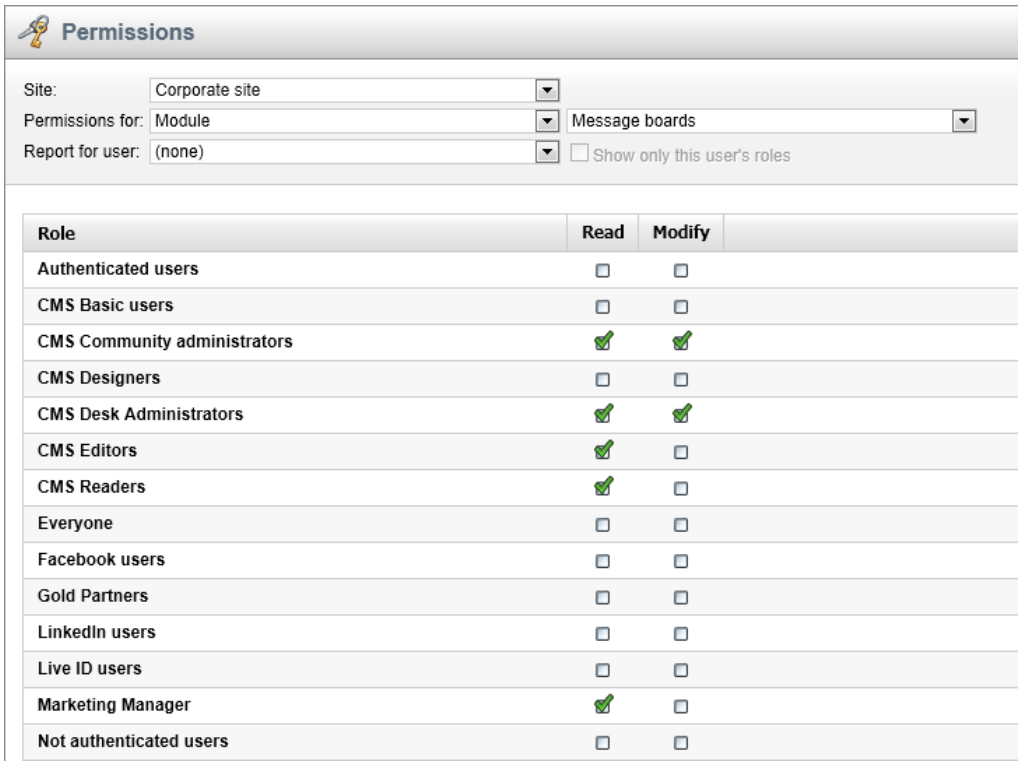
- the page must be accessed with *groupid* parameter in querystring
- the current user must be member of the group whose *groupid* is in the querystring
- the current user must not be hidden (configured by the *Is hidden* option when editing the user)

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



8.31.8 Message board notifications

8.31.8.1 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.

8.31.8.2 Who can be notified

When a new message is added or during message editing, notifications can be sent to:

- **Board moderators**
- **Subscribers**

The following text explains to whom notifications are sent under specific conditions:

New message

1. has been added by the 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. 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

Message editing

1. the message is switched from NOT APPROVED to APPROVED

- the e-mail is sent to subscribers

2. other message changes

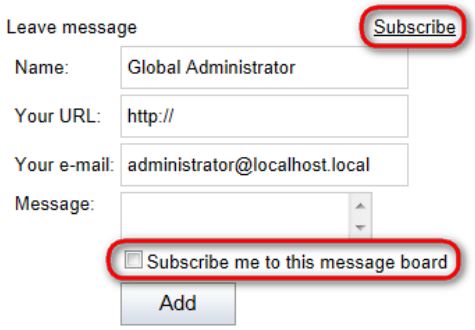
- no e-mail is sent

8.31.8.3 User subscriptions

Message board notifications

You can let users subscribe to receiving notifications about new messages added to your message board. You need to check the **Enable subscriptions** check-box in **Message board** web part properties. Alternatively, when the message board is already created, you can allow subscriptions in **CMS Desk -> Tools -> Message boards -> Boards**. If you choose to **Edit** () the message board and switch to the **General** tab, you can check the same **Enable subscriptions** check-box here, which enables the subscriptions.

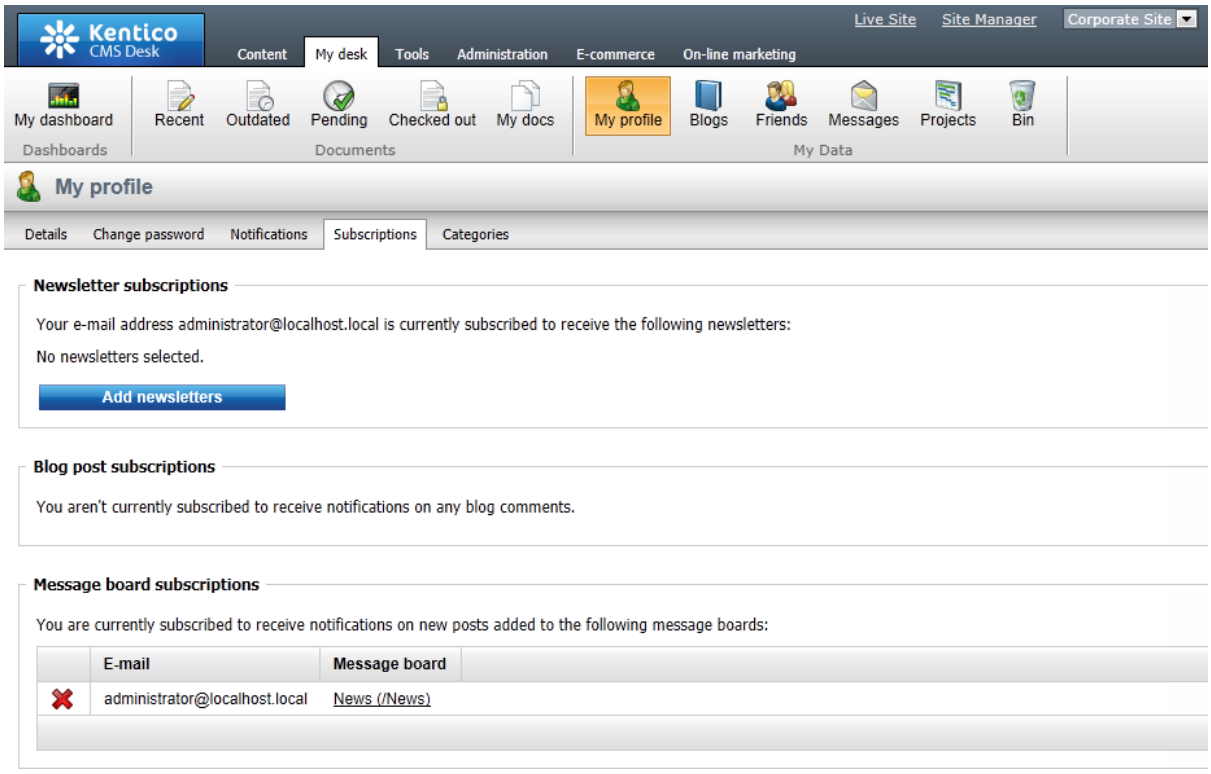
The subscription itself can be done two ways. Users can either check the **Subscribe to message board** check-box, which subscribes them along with adding the message. They can also click the **Subscribe to board** link, which displays only one **E-mail** field. After entering their address and clicking the **Subscribe** button, they can subscribe to notifications about this board without leaving any message.



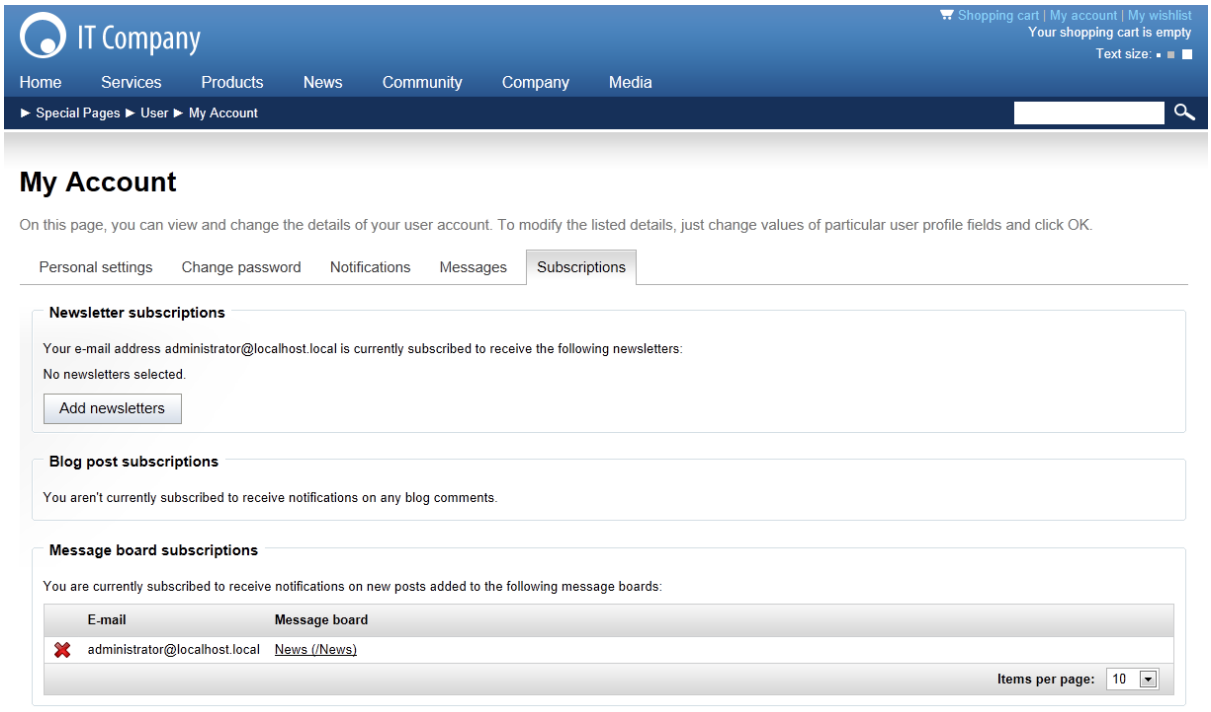
Subscriptions management

Users can view their subscriptions and eventually **unsubscribe** using the **Delete** () icon at the following two places:

1. Users with access to **CMS Desk** can view their subscriptions on the **My Desk -> Account -> Subscriptions** tab.

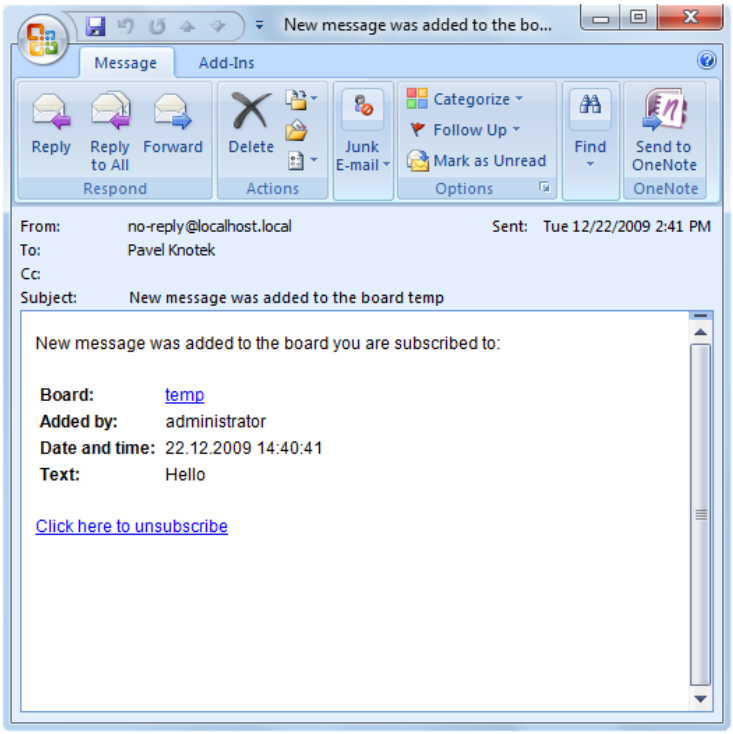


2. On live site, users can view their subscriptions in the **My account** web part. The **Display my subscriptions** property of the web part must be enabled for this to be possible.



Unsubscription links configuration

Each message board notification e-mail contains an unsubscription link. 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 to a page. It is recommended to create a special page for this purpose, as you can see at **Corporate site -> Special pages -> Board unsubscribe** 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 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.

8.31.8.4 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

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
- **UnsubscriptionLink** - unsubscription link

8.31.9 Message boards internals and API

8.31.9.1 Overview

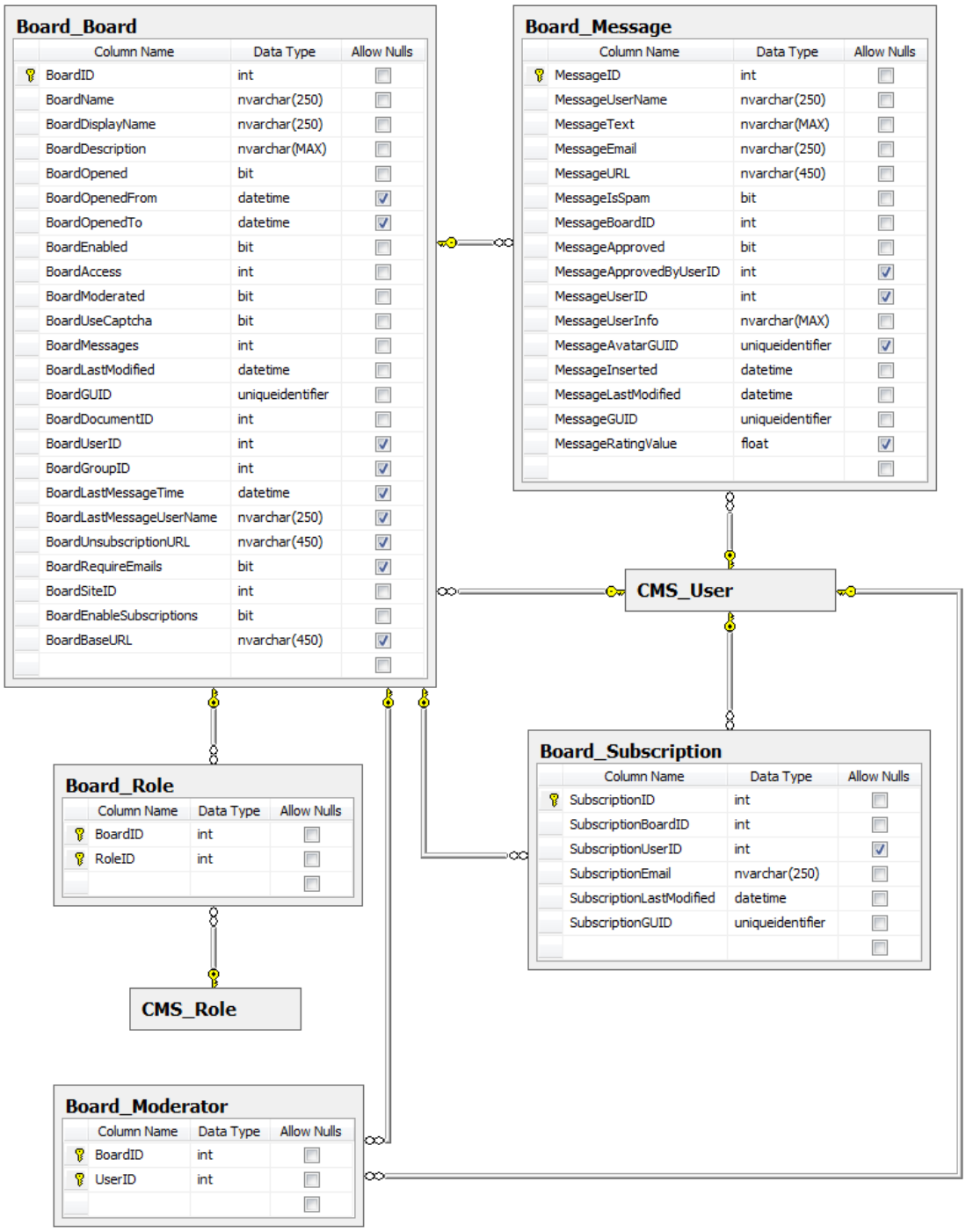
In this chapter, you will learn which database tables and API classes are used in the Message boards module. You will also see the most common API examples.

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.

8.31.9.2 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.



8.31.9.3 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.

8.31.9.4 API examples

8.31.9.4.1 Overview

Please note

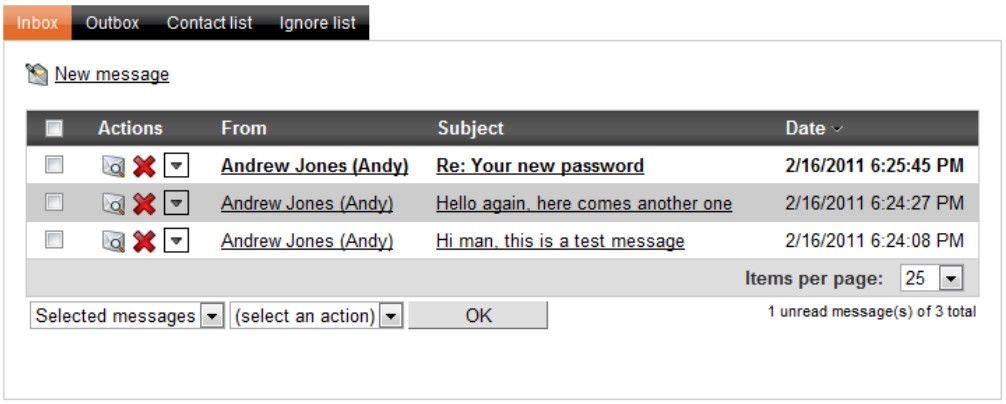
All API examples can be simulated in the API examples user interface; please refer to the API examples chapter for more details.

If you would like to check the code of the examples, please refer to ***<web project folder>\CMSAPIExamples\Code\Community\MessageBoards\Default.aspx.cs***.

8.32 Messaging

8.32.1 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.

8.32.2 My messages

The **CMS Desk -> My desk -> Messages** section of the administration interface is divided into the following four tabs:

Inbox

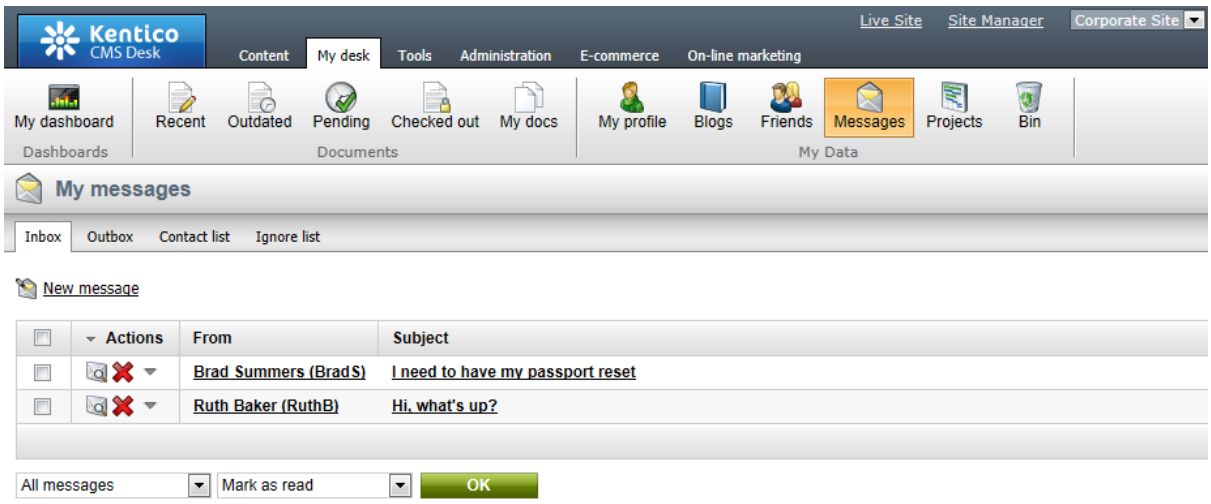
On this tab, the current user can see a list of all their received messages. The following actions are available for each message:

- **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.

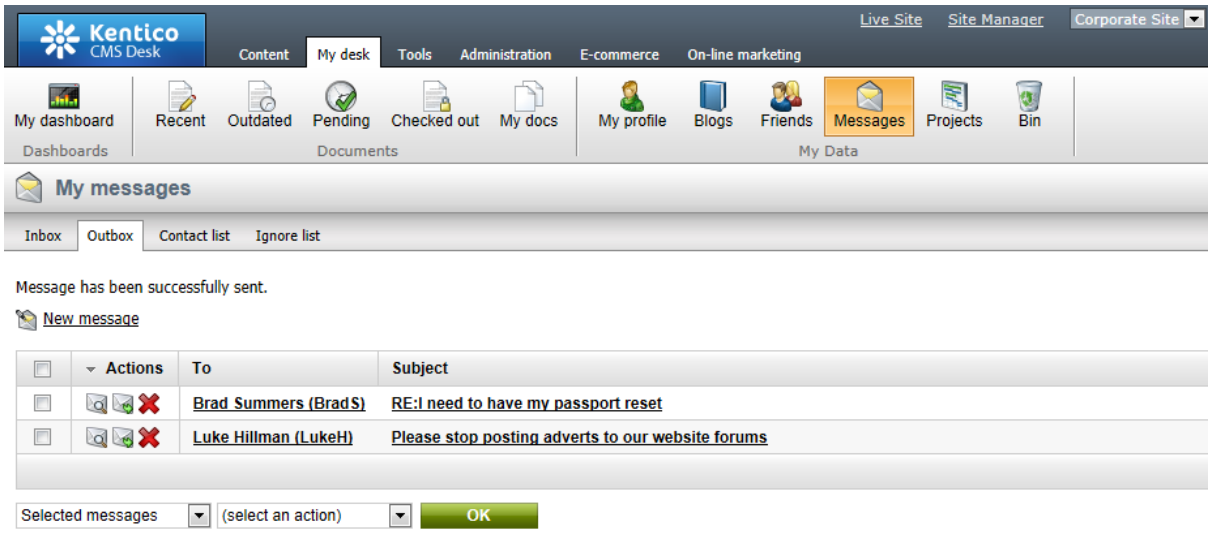


Outbox

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.

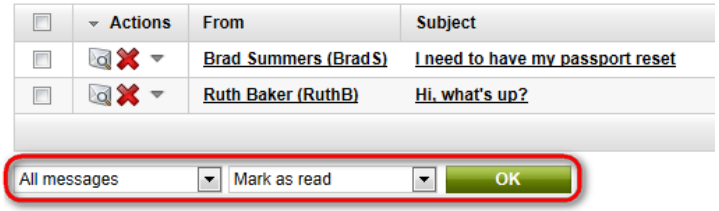


Bulk actions in Inbox and Outbox

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.

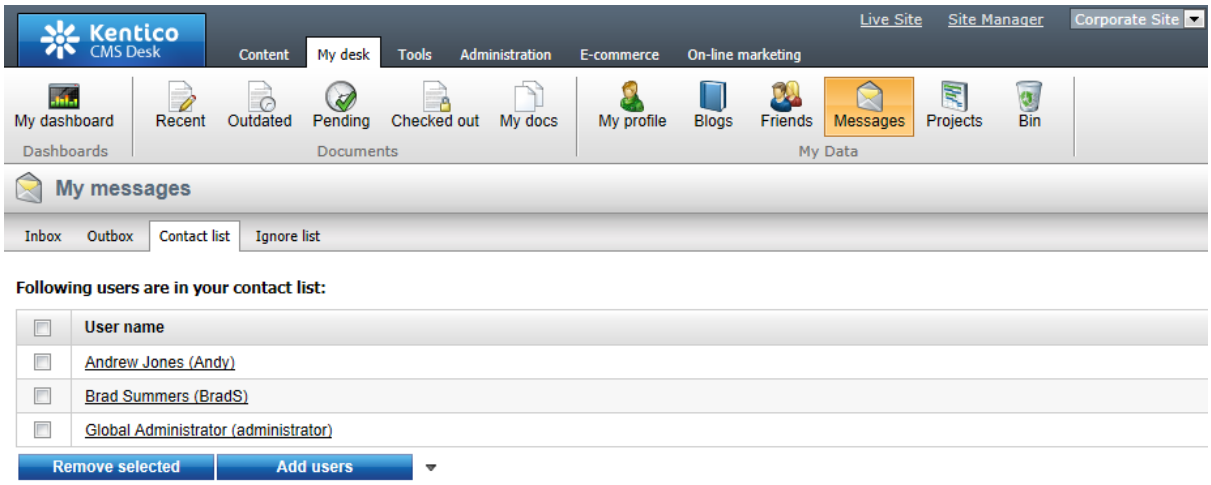
After choosing from the first drop-down list, you need to select the action to be taken from the second one and click **OK**. In **Inbox**, the **Delete**, **Mark as read** and **Mark as unread** actions are available. I **Outbox**, you can only use the **Delete** action for multiple files at once.



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.

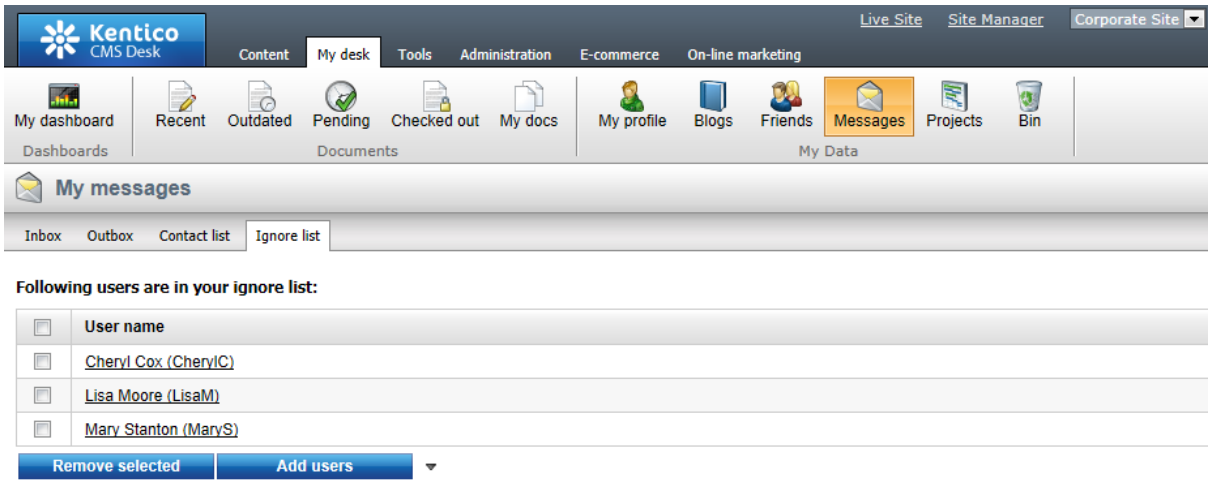
The **Add users** button opens a pop-up dialog where you can select users to be added to the list. The **Remove selected** button removes users selected by the check-boxes from the list.



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.

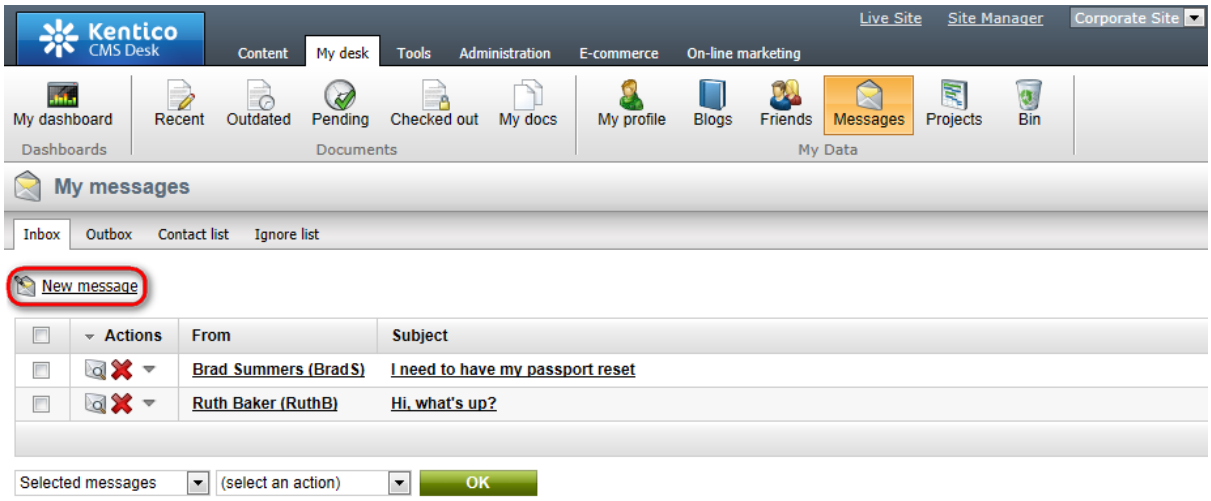
The **Add users** button opens a pop-up dialog where you can select users to be added to the list. The **Remove selected** button removes users selected by the check-boxes from the list.



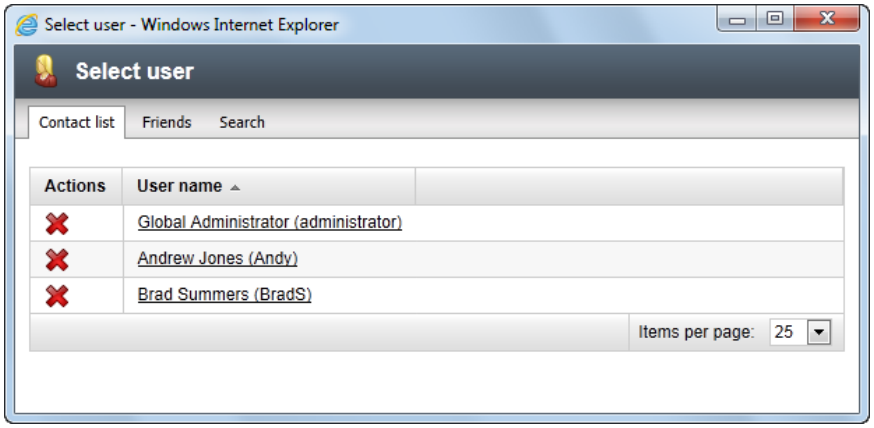
8.32.3 Sending a new message

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.

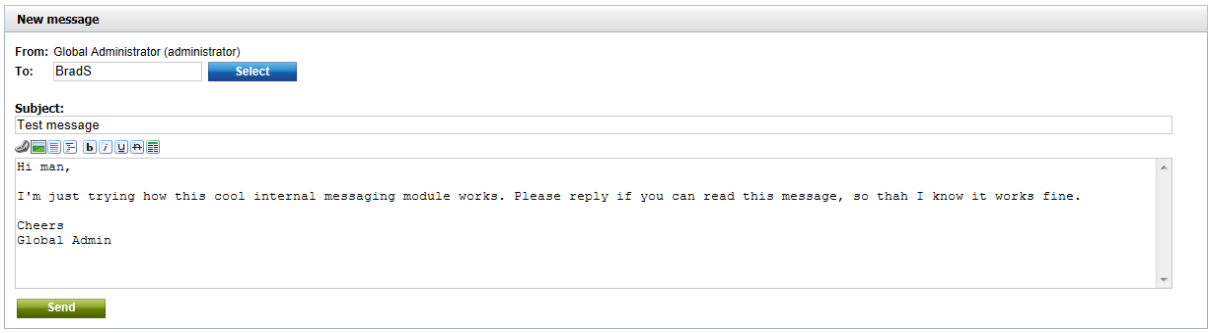
1. On each of these tabs, the first thing to do is to the **New message** link above the messages listing.



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.



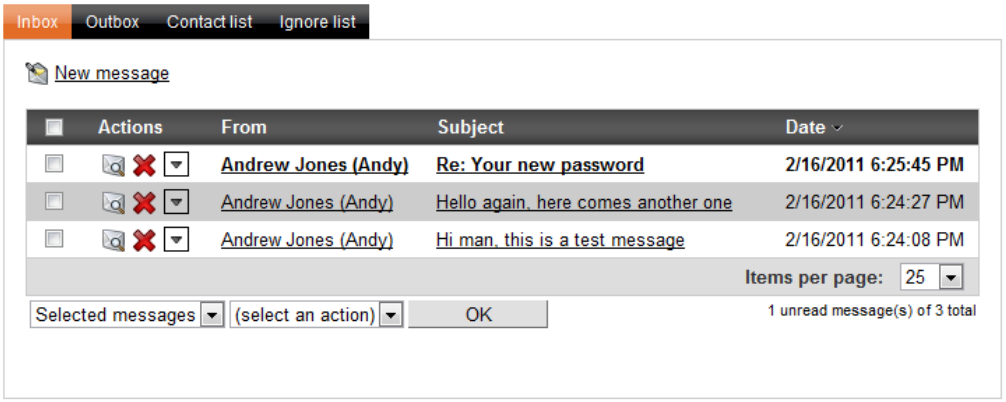
3. Then you need to fill in the **Subject** field and the actual text of the message into the large text area. Entered text can be formatted using BBCode. This is possible either by entering BBCode tags directly, or using the icons above the main text area. Supported BBCode tags are listed and explained in Modules -> Forums -> BBCode support.



4. Finally, the message can be sent out using the **Send** button.

8.32.4 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.

8.32.5 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 -> Account**.
- 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:

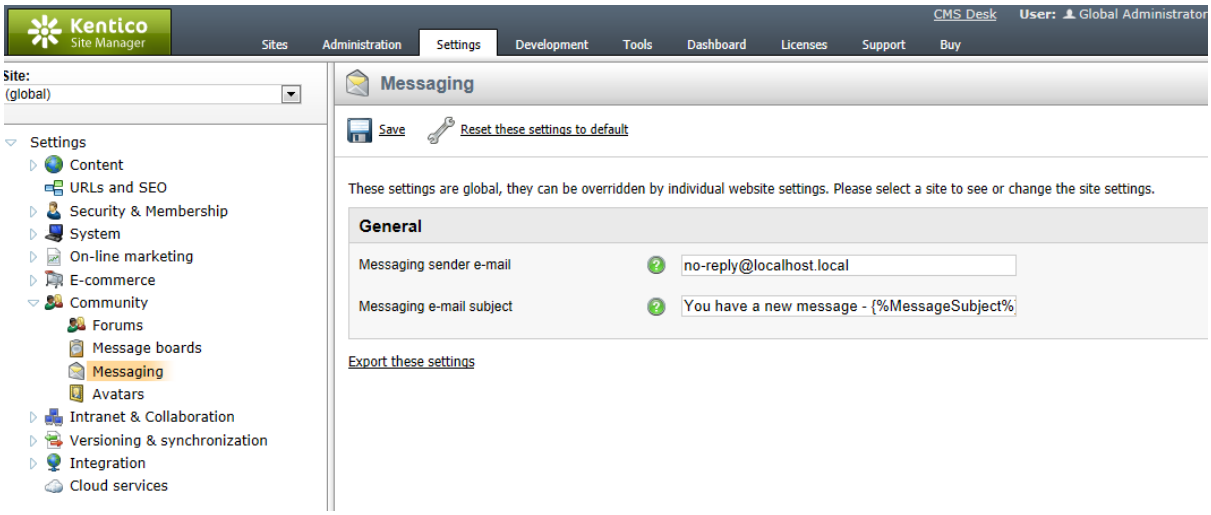
- **{% Sender %}** - *UserInfo* object of the message sender.
- **{% Recipient %}** - *UserInfo* object of the message recipient.
- **{% Message %}** - *MessageInfo* object of the message.

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.



8.32.6 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.

8.32.7 Messaging internals and API

8.32.7.1 Overview

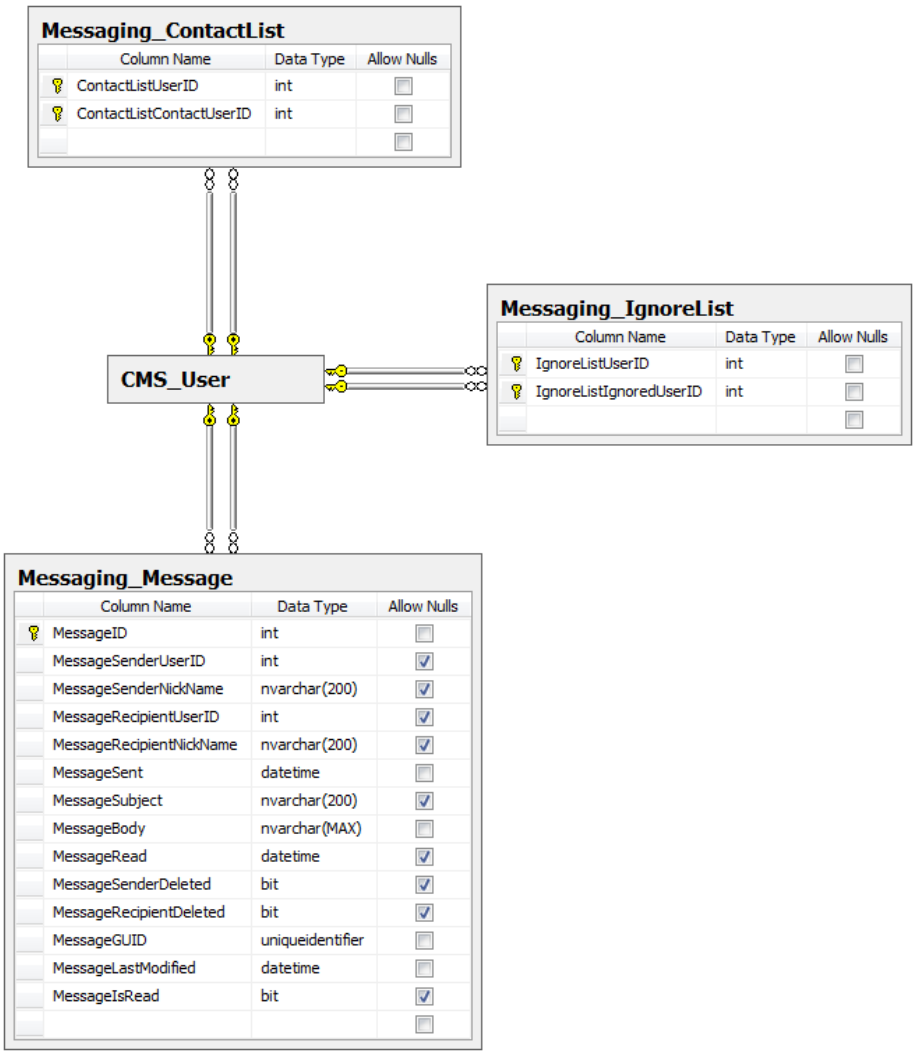
In this chapter, you will learn which database tables and API classes are used in the Messaging module. You will also see the most common API examples.

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.

8.32.7.2 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.



8.32.7.3 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.

8.32.7.4 API examples

8.32.7.4.1 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;
```

8.32.7.4.2 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;
}
```

8.32.7.4.3 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;
```

```
}
```

8.32.7.4.4 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;
}
```

8.33 Multivariate testing

8.33.1 Overview

For information about **Multivariate testing**, please see the Modules -> Website optimization -> Multivariate testing chapter of this guide.

8.34 Newsletters

8.34.1 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 resent. 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 use templates to give all issues and related notification e-mails a unified appearance by using 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 with the tracking and monitoring of e-mails, which can be very useful when carrying out e-mail marketing campaigns. More information about this is given in the On-line marketing topic.

The Troubleshooting topic contains a list of solutions that can be used to resolve common problems with newsletter e-mails.

8.34.2 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:

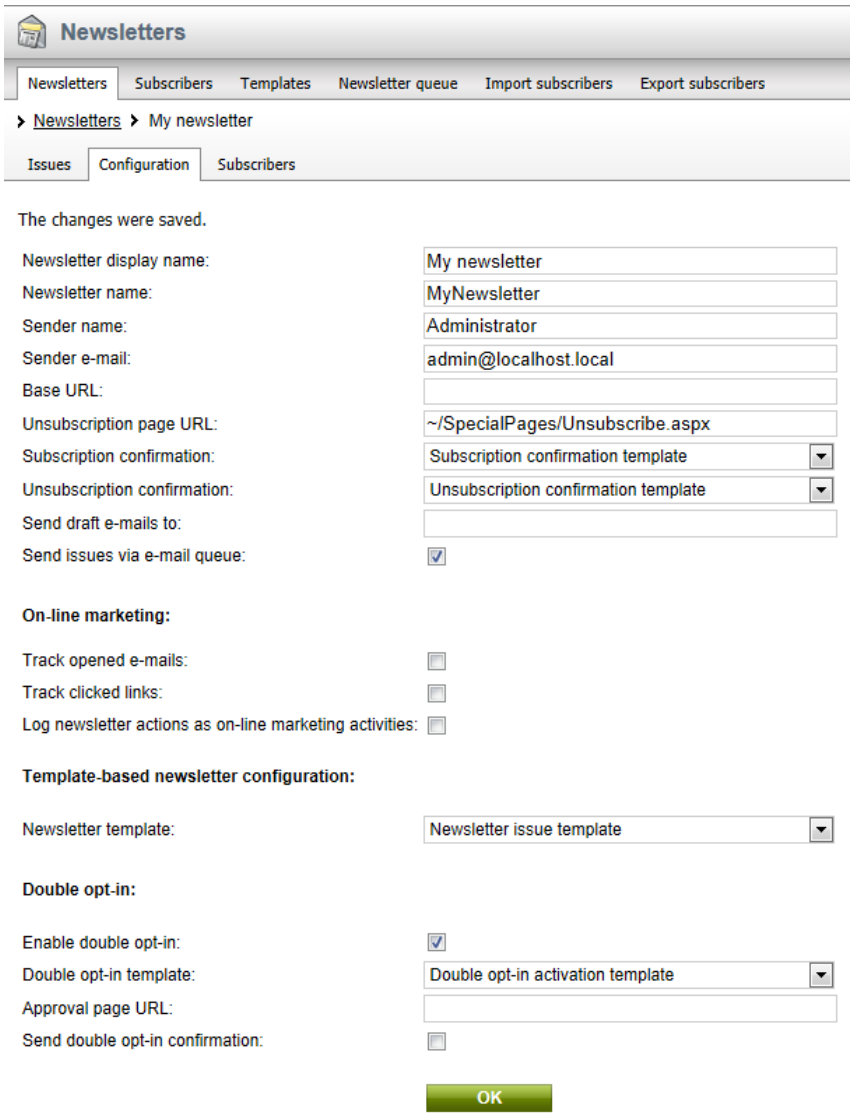
- **Newsletter display name**: My newsletter *(this is the name displayed to the users)*
- **Newsletter name**: MyNewsletter *(this is the code name used in web parts and code)*
- **Subscription confirmation**: Subscription confirmation template
- **Unsubscription confirmation**: Unsubscription confirmation template
- **Sender name**: Enter your full name
- **Sender e-mail**: Enter your e-mail address

Choose **Template based newsletter** and use the *Newsletter issue template***.** Click **OK**. 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, 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're using a different URL for your editing environment than for the live website. Example: https://www.example.com
- **Unsubscription page URL** - enter *~/SpecialPages/Unsubscribe.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.

- **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 topic for further details.

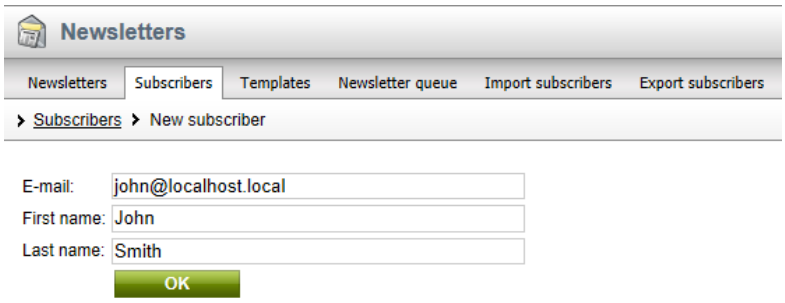
- **Double opt-in** - the properties in this section are related to double opt-in subscription. Please refer to the Double opt-in topic for more information.



Click **OK**.

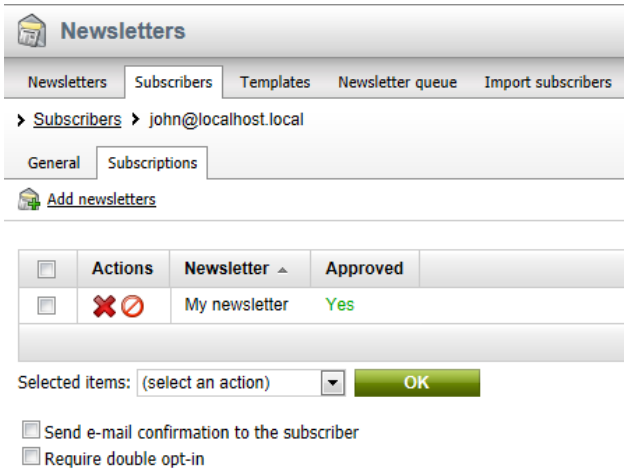
2. Now we will create a new subscriber. Go to the main **Subscribers** tab (in the top tab menu) and click **New subscriber**. Enter the following details:

- **E-mail** - subscriber's e-mail address.
- **First name** - subscriber's first name.
- **Last name** - subscriber's last name.



Click **OK**. The subscriber has been created. Now we will assign this subscriber to our previously created newsletter.

3. Go to the **Subscriptions** tab, click the **Add newsletters** link, check the box next to **My newsletter** 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 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 subscriptions.

4. Your newsletter is configured. You can now create new issues as is described in the Authoring static newsletter issues topic.

8.34.3 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 and edit (✎) **My newsletter**. Click **Create new issue** on the **Issues** tab. The wizard will guide you through the process of creating a new newsletter issue.

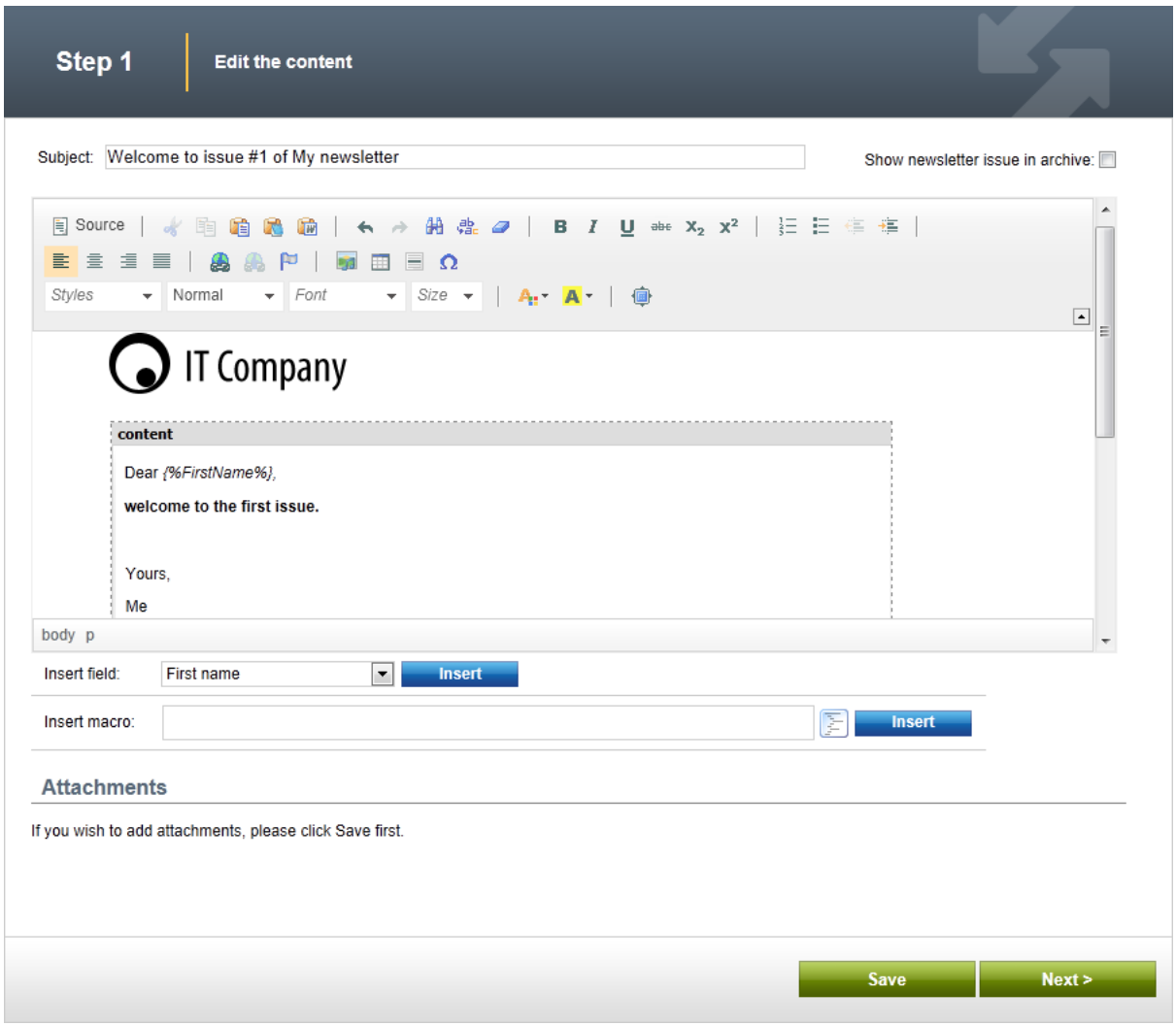
2. Enter the following **Subject**: Welcome to issue #1 of My newsletter

3. Now enter the following text into the **content** editable region:

```
Dear ,

welcome to the first issue.


Yours,

Me
```

Place the cursor after the word Dear, choose **First name** from the **Insert field** drop-down list below the region and click the **Insert** button. The macro expression *{%FirstName%}* will be placed into the text. This macro will automatically be replaced by the e-mail recipient's first name during the mail merge.

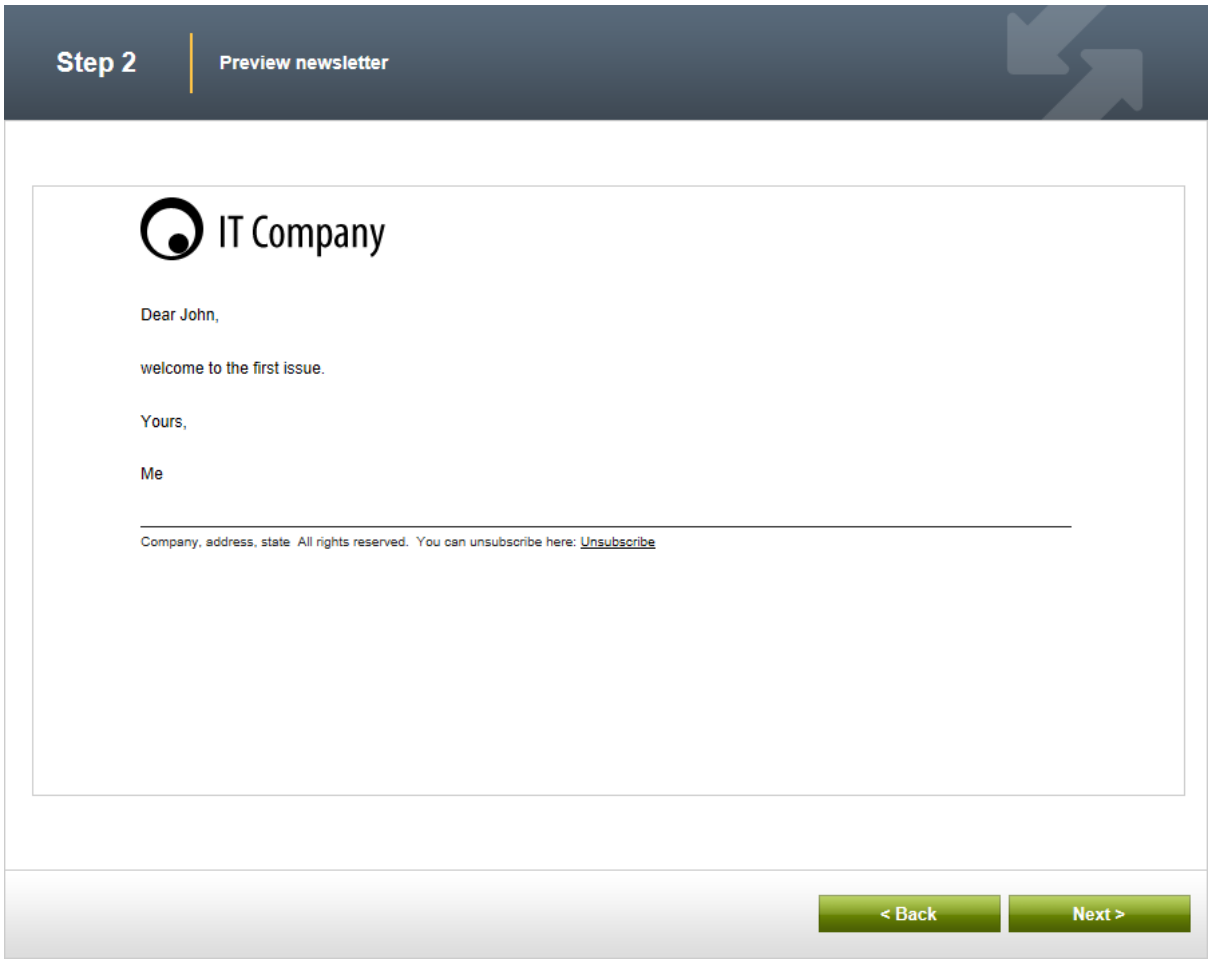
The issue now looks like this:



Newsletter templates

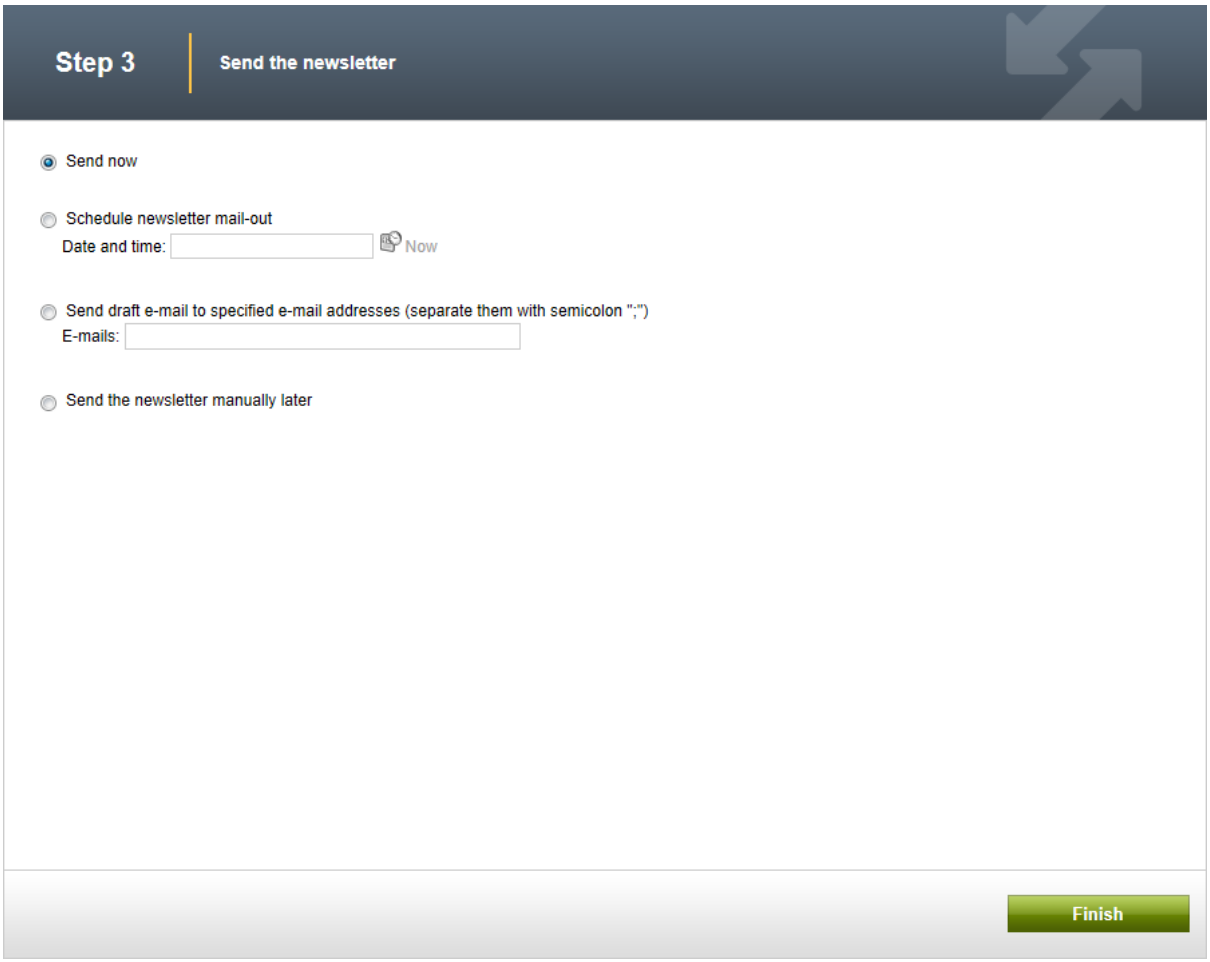
The structure of the newsletter text is defined by the newsletter issue template that can be edited on the **Templates** tab. Templates are described in the Newsletter templates topic.

4. Click **Next**. Now you can preview the content of the newsletter for each subscriber. You can click **Back** and modify the text if necessary.

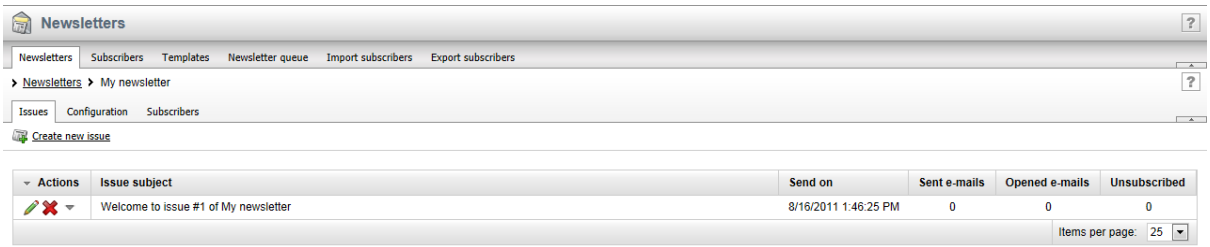


5. Click **Next** again. Now you can choose when the newsletter issue should be sent out:

- **Send now** - the newsletter issue is sent out immediately to all subscribers.
- **Schedule newsletter mail-out** - the issue is sent out on a specified date and time.
- **Send draft e-mail to specified e-mail addresses** - the issue is sent to the specified e-mail addresses (separated by semicolons). The e-mails sent this way are only for testing purposes, so any macros/fields are replaced by their default values, the unsubscription link is not functional and the e-mails are not included in tracking statistics (e-mail opening and link clicking).
- **Send the newsletter manually later** - the issue will not be sent and you can decide on the mail-out time later.



Choose **Send now** and click **Finish**. On the Issues tab, you can see how many e-mails have already been sent out and how many subscribers unsubscribed after receiving this issue:



6. If you experience problems with receiving the newsletter issue e-mail, please follow the instructions in the Troubleshooting topic.

8.34.4 Creating a dynamic newsletter

Dynamic newsletters contain the content of a specified page and they are sent out automatically on a regular basis, using the built-in scheduling system. When the mail-out 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 page and ensures that the issues are sent out according to the set time interval.

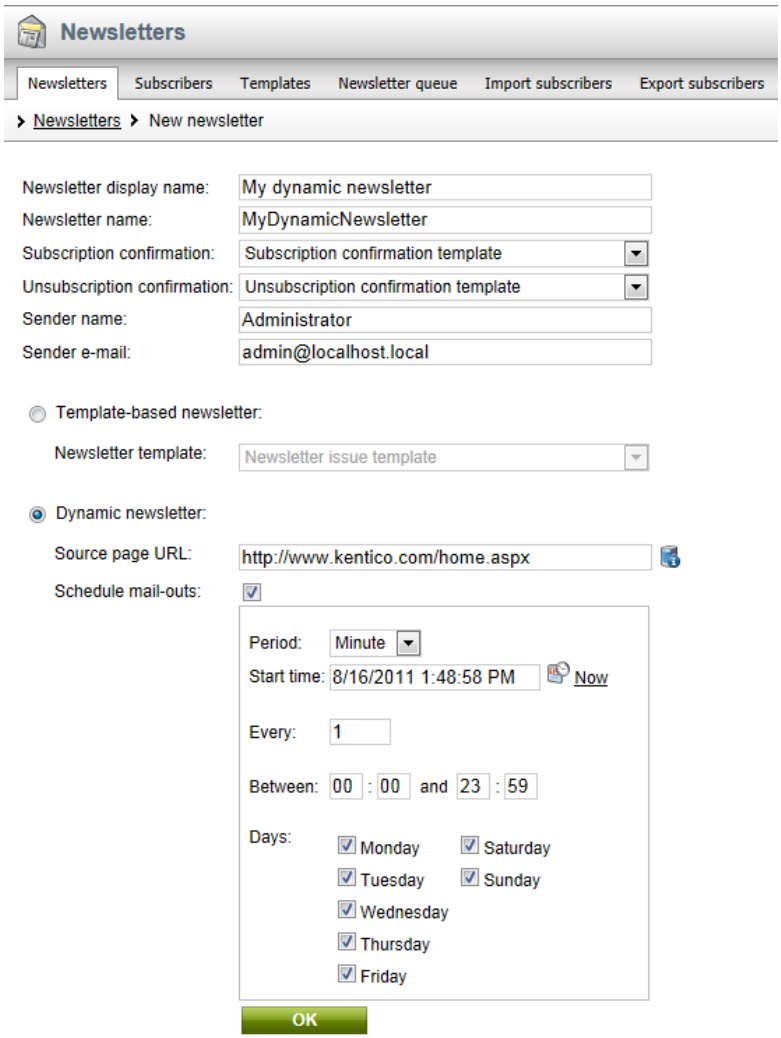
The following steps will guide you through the creation of a dynamic newsletter:

1. Go to **CMS Desk -> Tools -> Newsletters** and click **New newsletter**. Enter the following details:

- **Newsletter display name**: My dynamic newsletter
- **Newsletter name**: MyDynamicNewsletter
- **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/home.aspx* (this is the URL of the page the newsletter takes its content from)
- **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

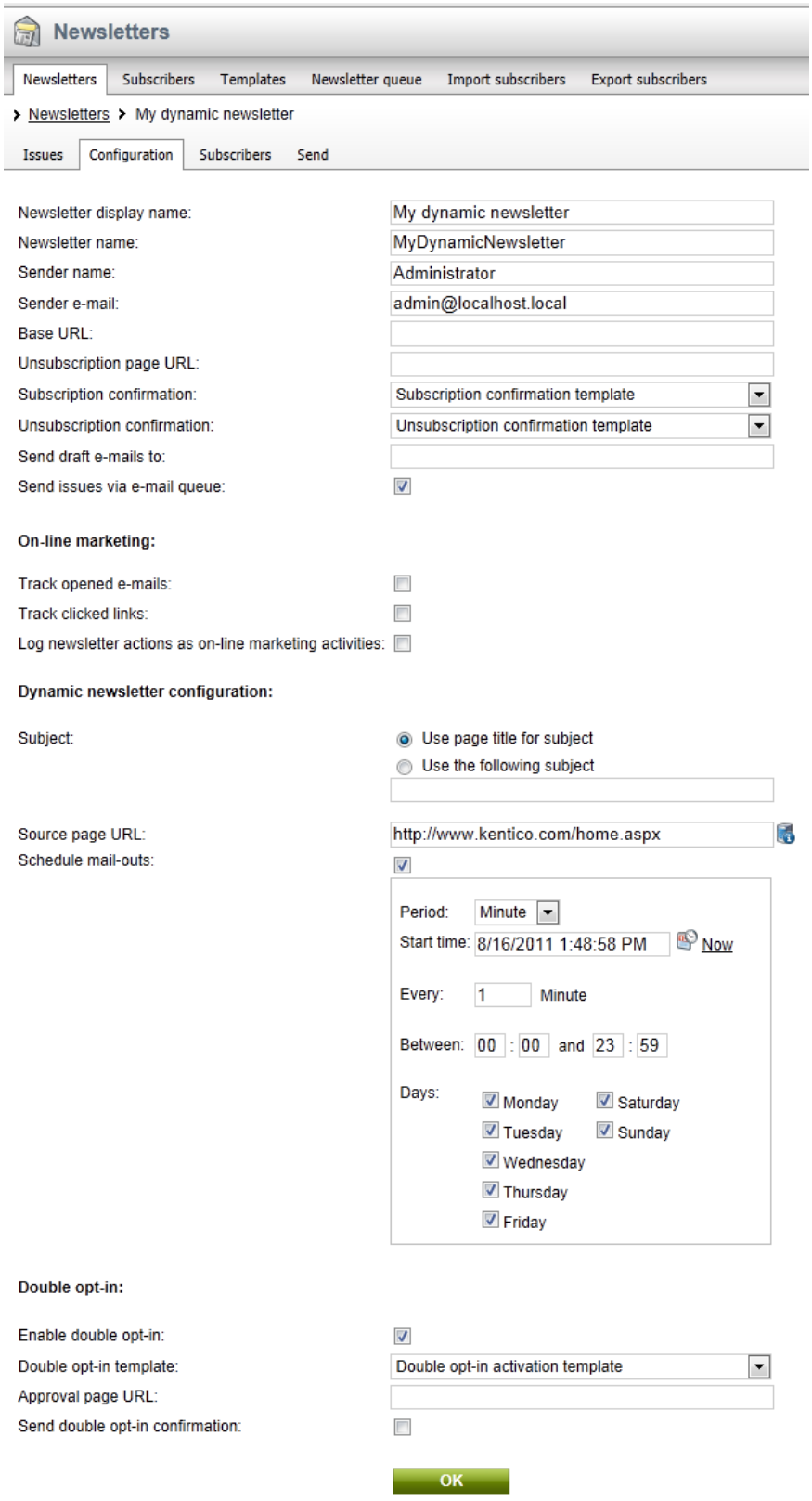


Click **OK**. The newsletter will be created and you will be redirected to its **Configuration tab**, where the following additional options can now be set:

- **Base URL** - here you can specify the base URL of your website, 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're using a different URL for your editing environment than for the live website. Example: https://www.example.com
- **Unsubscription page URL** - enter *~/SpecialPages/Unsubscribe.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.

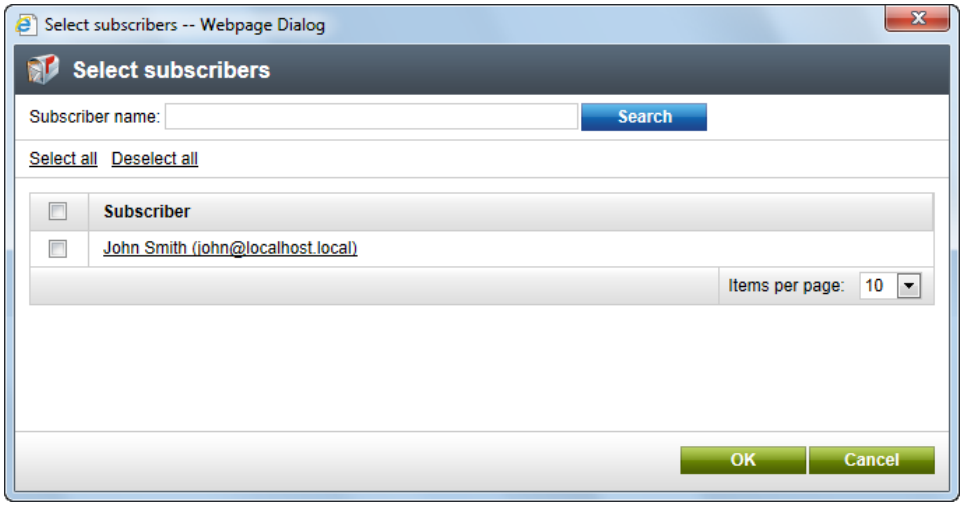
- **On-line marketing** - the properties in this section are related to tracking of the newsletter's e-mails and keeping marketing statistics. Please see the On-line marketing topic for further details.

- **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*.
- **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.

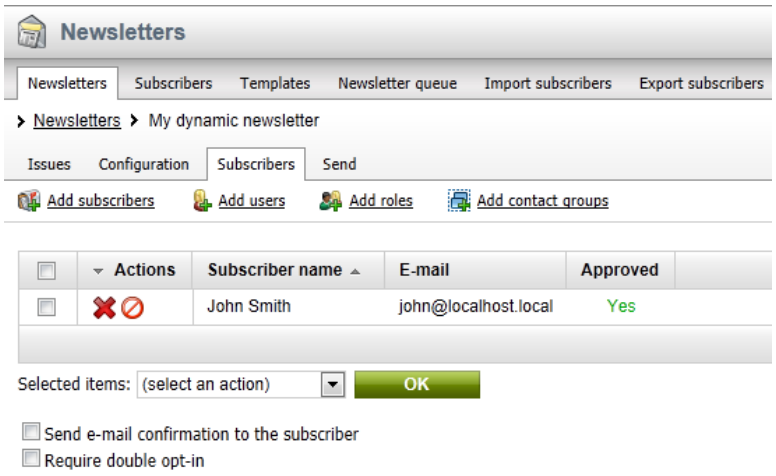


Click **OK**.

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 the **Add subscribers** link. The following window 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** links 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 the list of sent issues. You may need to wait up to 2 minutes until the first issue is sent out. You can refresh the page by clicking the **Issues** tab again.

4. Check your mail box, you should receive the content of the given page by e-mail:

In this way, you can send out any page from your website. You can create a new page only for newsletter purposes that will display e.g. new articles added to your website during the last month.

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 at **CMS Desk -> Content -> Edit -> ... select a page from the content tree ... -> Properties -> Metadata -> Page title**.

8.34.5 Double opt-in

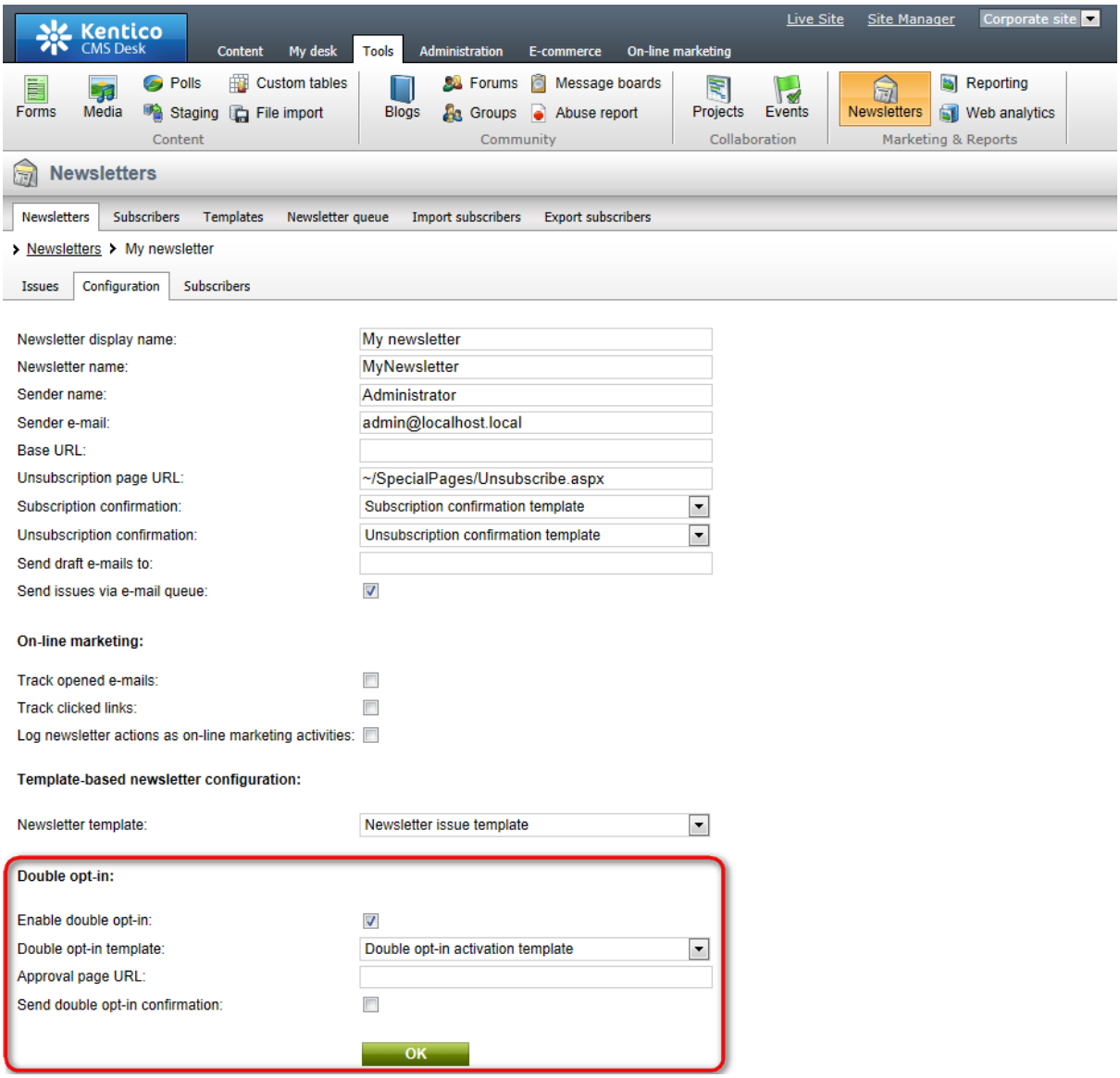
Double opt-in functionality, also referred to as confirmed opt-in, provides an additional security layer to newsletter subscription. It confirms that the entered e-mail address actually exists and prevents users from being unknowingly subscribed to a newsletter by someone else, either intentionally or 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.

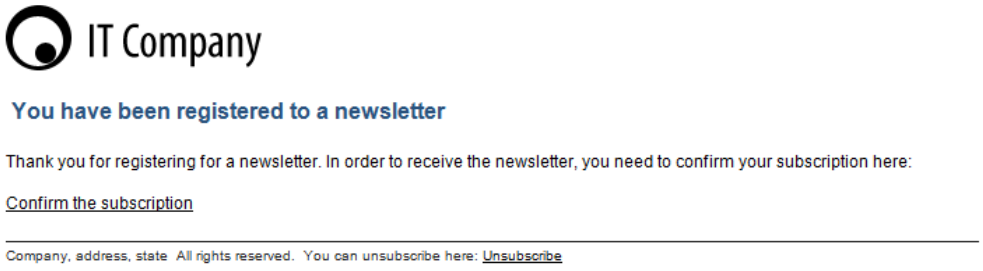
Enabling double opt-in

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** box in the bottom section of properties and set the remaining ones below:

- **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 **OK** 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):



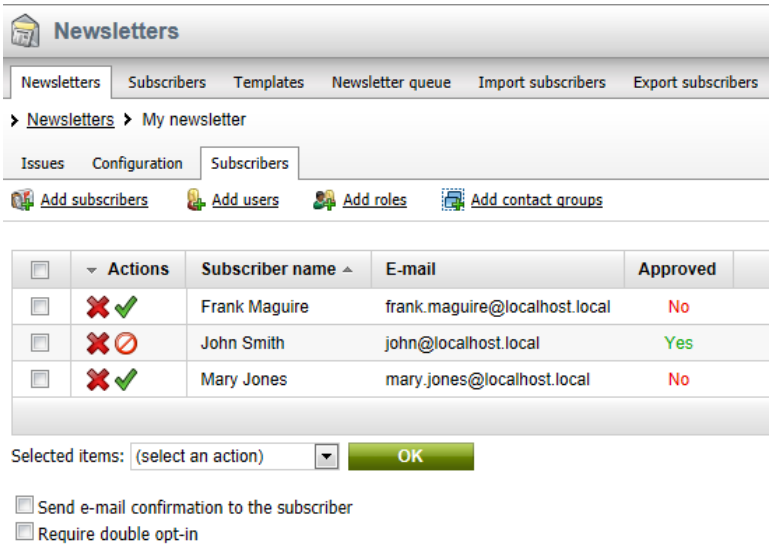
When a user clicks the **Confirm the subscription** link, they will be redirected to the default **~/CMSModules/Newsletters/CMSPages/SubscriptionApproval.aspx** page, where their subscription

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.

8.34.6 Integrating newsletters into the site

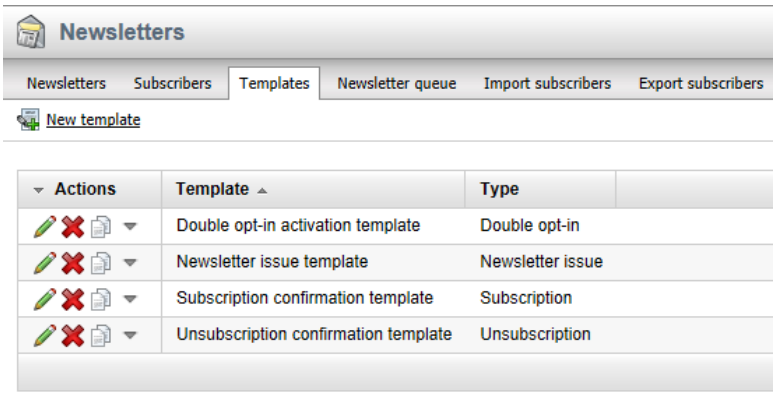
You can integrate newsletters into your website using the following web parts:

- Newsletter subscription - displays the newsletter subscription dialog.
- 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.

8.34.7 Newsletter templates

The e-mails sent by the Newsletter module are defined by templates. These templates can be managed at **CMS Desk -> Tools -> Newsletters -> Templates**.



There are four types of templates:

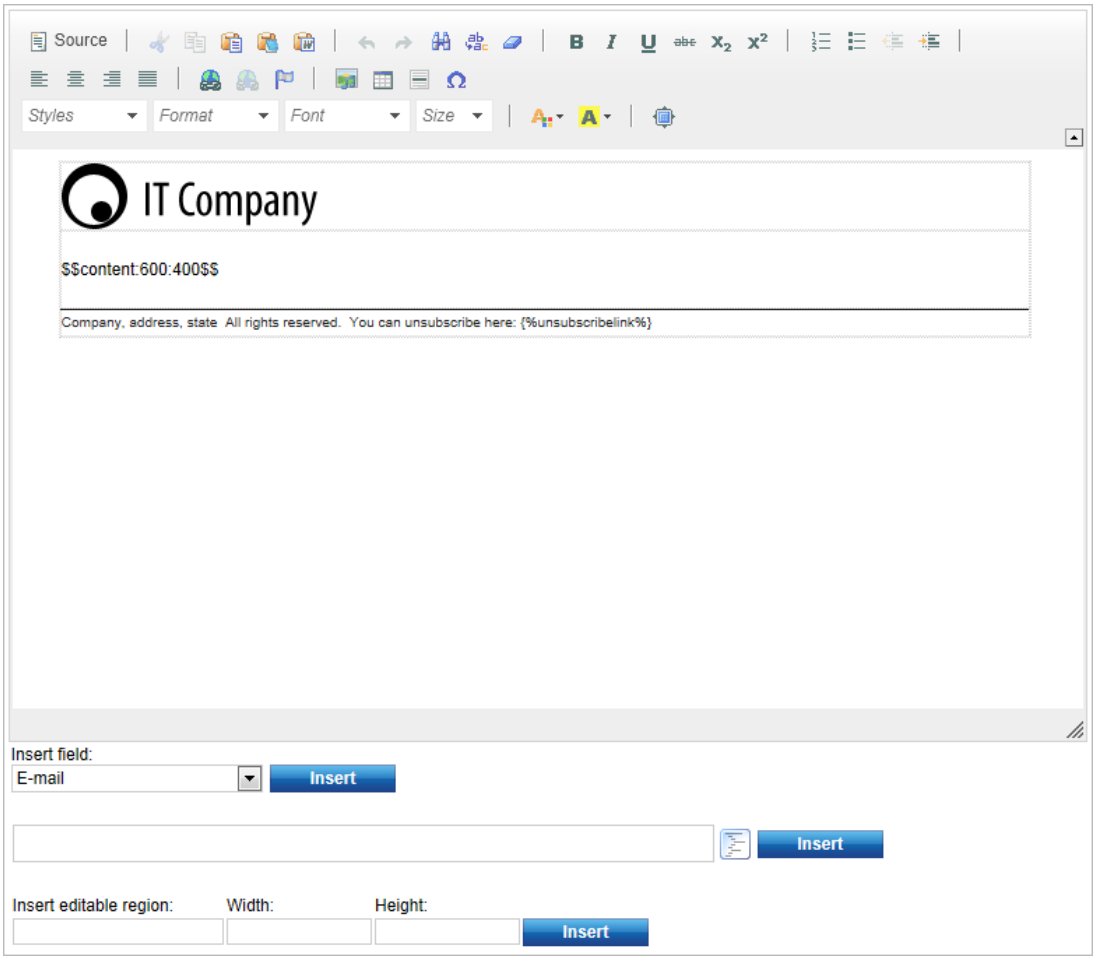
- **Double opt-in** - template for e-mail messages sent to users to confirm their subscription to a newsletter that has double opt-in enabled.
- **Newsletter issue** - this is a template that defines the layout and design of static newsletter issues. It contains editable regions where newsletter authors can enter the content.
- **Subscription** - template for e-mail message sent when a user subscribes to the newsletter.
- **Unsubscription** - template for e-mail message sent when a user unsubscribes.

Editing templates

When editing a template, the following configuration can be made:

- **Display name** - sets the name of the template that is displayed in the newsletter interface.
- **Code name** - sets the name of template that is used in code.
- **Subject** - can be used to set the subject of the e-mails that use 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 leading HTML code 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. Regions are inserted as macro

expressions in format: *$$regionName:width:height$$*



- **Footer (HTML)** - the closing HTML code.
- **CSS stylesheet** - the CSS styles applied to the e-mails that use this template. These styles are used by the newsletter issue editors and they are included in the e-mails.

The following newsletter fields may be inserted anywhere in the body of the template:

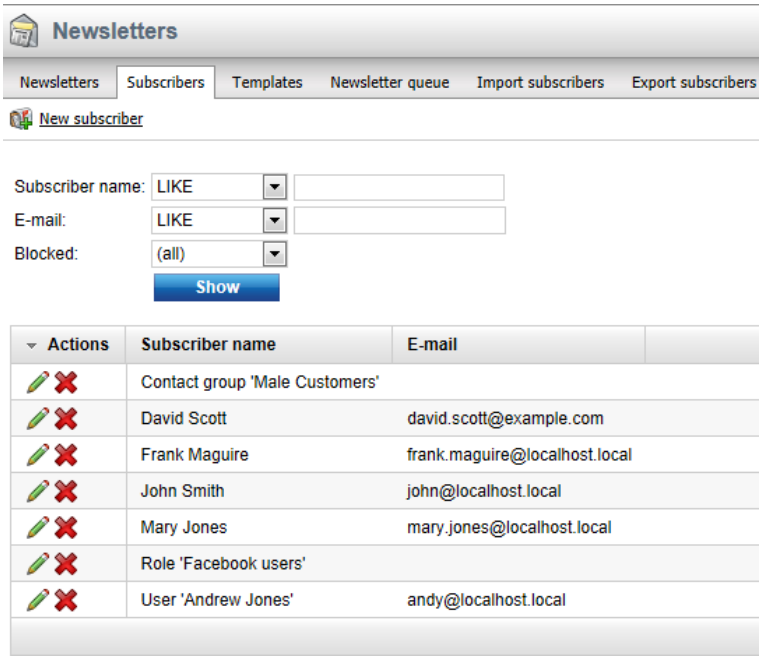
- **{%ActivationLink%}** - only available for *Double opt-in* templates. It is resolved into a link to the subscription approval page, as defined by the *Approval page URL* property of the given newsletter.
- **{%Email%}** - resolves into the e-mail address of the e-mail recipient.
- **{%FirstName%}** - resolves into the first name of the e-mail recipient.
- **{%LastName%}** - resolves into the last name of the e-mail recipient.
- **{%UnsubscribeLink%}** - resolves into a link to the unsubscription page, as defined by the *Unsubscription page URL* property of the given newsletter.

Additionally, other types of macro expressions as seen in Development -> Macro expressions can be entered into templates.

File attachments can also be added to the template at the bottom of the page.

8.34.8 Subscriber management

You can manage subscribers at **CMS Desk -> Tools -> Newsletters -> Subscribers**.

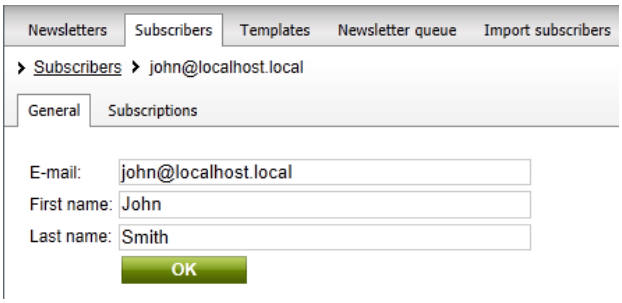


On this page, you can see a list of all subscribers of all of the website's newsletters. You can filter these according to their name or e-mail, by entering the desired value into the appropriate filter field and clicking the **Show** button.

By clicking the **Delete** ( ) action icon next to a subscriber record, you can remove the subscriber from the list and consequently from all newsletters they are subscribed to.

By clicking the **Edit** ( ) icon next to a subscriber record, you can change the subscriber's details and manage newsletter subscriptions.

On the **General** tab, you have the same options as when creating a new subscriber.

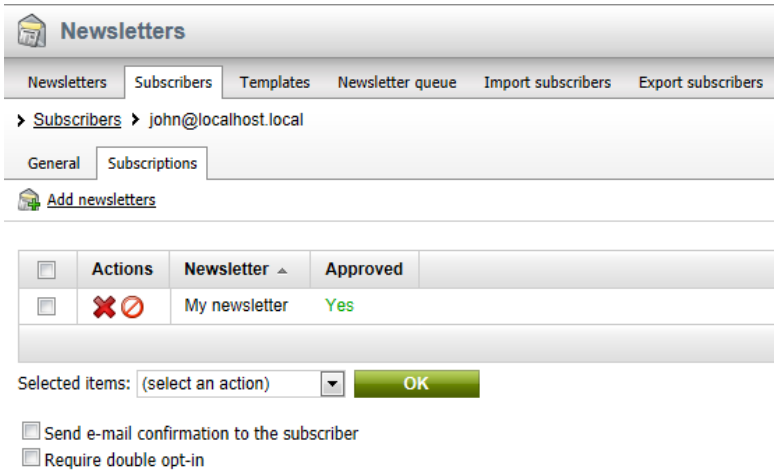


On the **Subscriptions** tab, you can see a list of newsletters that the edited subscriber is subscribed to. More can be added by using the **Add newsletters** button. Individual newsletter subscriptions can be managed via the following actions:

- **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

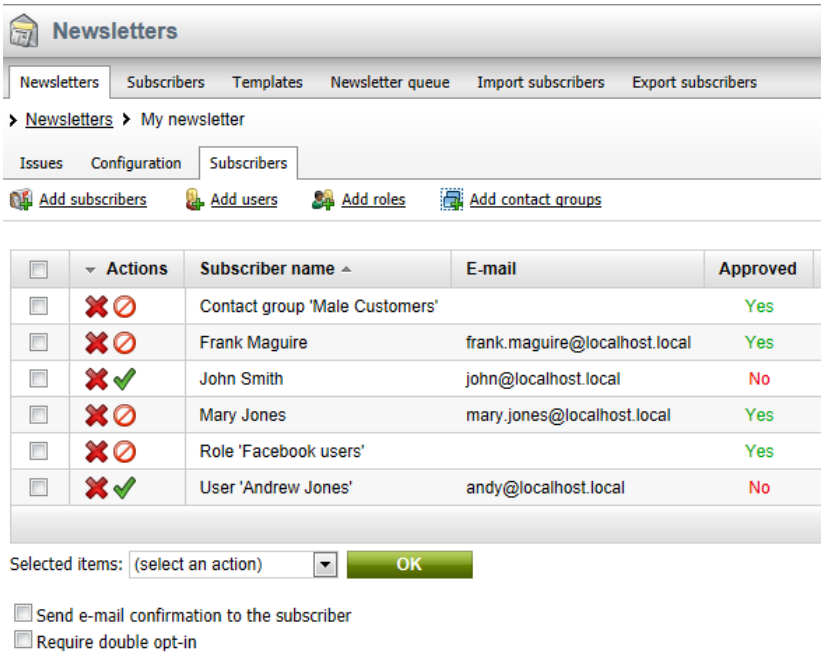
Alternatively, subscribers of individual newsletters can be managed at **CMS Desk -> Tools -> Newsletters -> Newsletters -> ... edit ( ) Newsletter ... -> Subscribers**. This tab displays a list similar to the one on the main Subscribers tab, but it only contains subscribers of the current newsletter. 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 links:

- **Add subscribers** - allows selection from a list of all subscribers in the system.
- **Add users** - allows selection from the 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 in 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.

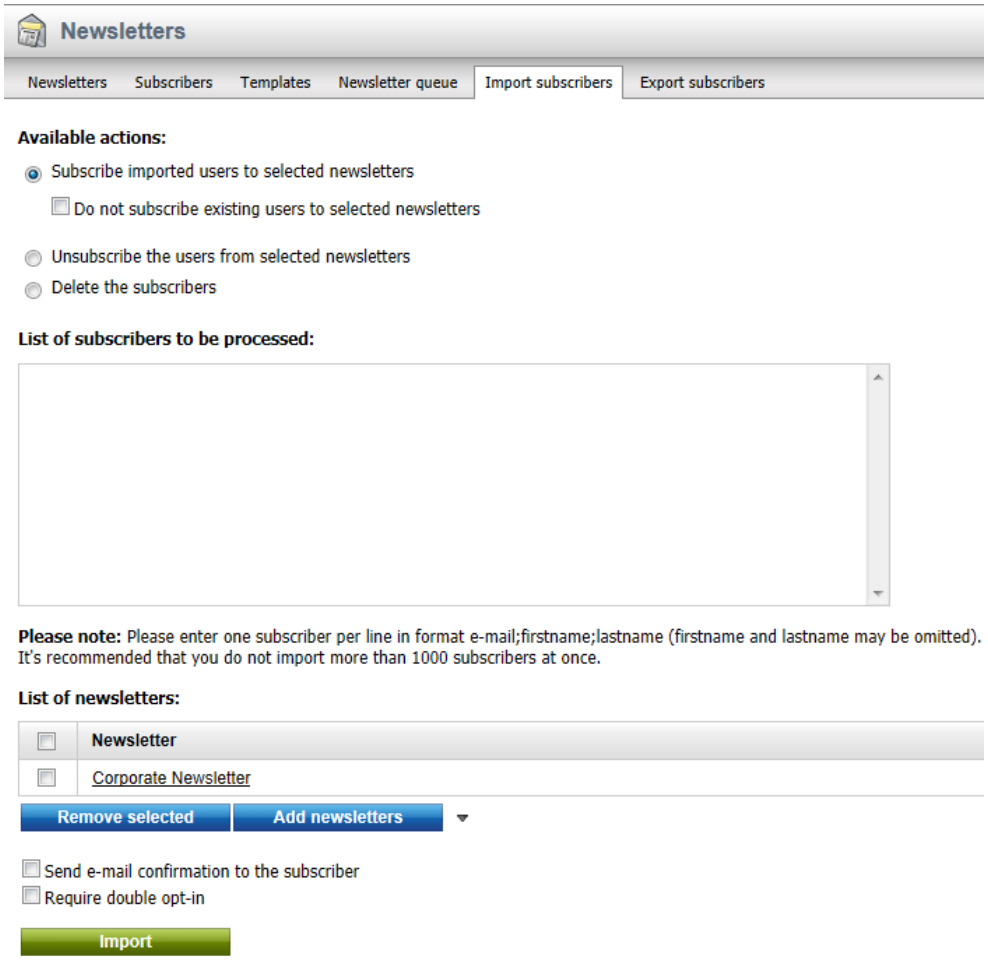
If you need to add or modify a large amount of subscribers, the **Import subscribers** tab described in the Subscriber import and export topic provides an easier way.

8.34.9 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

The importing or modification of subscribers can be carried out using the **CMS Desk -> Tools -> Newsletters -> Import subscribers** dialog.



A list of subscribers needs to be prepared in the following format:

- **email;firstname;lastname**

Copy it to the '**List of subscribers to be processed**' text area. Each line should contain one record in the mentioned format. 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
```

By selecting one of the three radio buttons above the text area and adding some newsletters using the **Add newsletters** button below, the following actions can be done:

- **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, users with already existing e-mail addresses will not be subscribed to the selected newsletters, only their names will be updated.
- **Unsubscribe the users from selected newsletters** - the entered users will be unsubscribed from

the newsletters selected below.
- **Delete the subscribers** - the entered users will be deleted from the list of subscribers.

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. Importing more than 1000 subscribers at once is not recommended.

If you check the **Send e-mail confirmation to the subscriber** box 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 the **Require double opt-in** box, subscriptions to newsletters that have double opt-in enabled will be inactive (rejected) until the users confirms them.

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.

8.34.10 On-line marketing

When using newsletters for e-mail marketing campaigns, it is important to determine the overall effectiveness by tracking e-mails and 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.

Please note that the features described below are only available if the **Enable on-line marketing** setting is enabled in **Site Manager -> Settings -> On-line marketing**.

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.

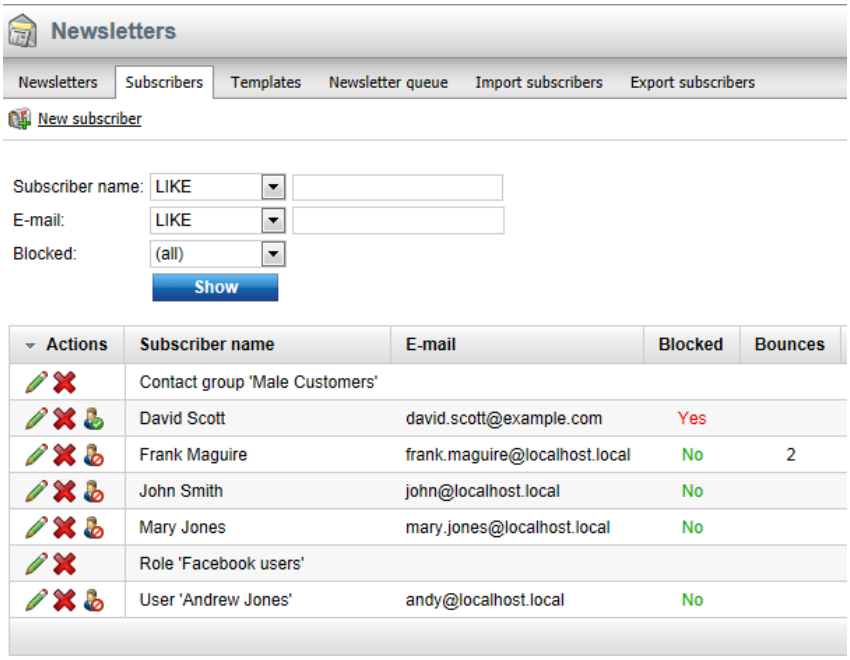
To enable this feature, go to **Site Manager -> Settings -> On-line marketing -> Newsletters** and check the **Monitor bounced e-mails** box. It is also necessary to properly fill in the remaining settings in the **Bounced e-mails** and **POP3 settings** categories (more information about individual 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.

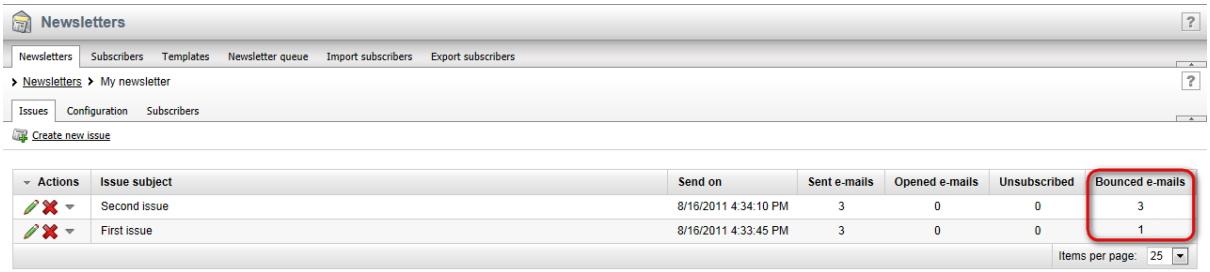
Bounce statistics can be viewed in various parts of the **CMS Desk -> Tools -> Newsletters** interface. On the **Subscribers** tab, the amount of bounces associated with individual subscribers is displayed.



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 at **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

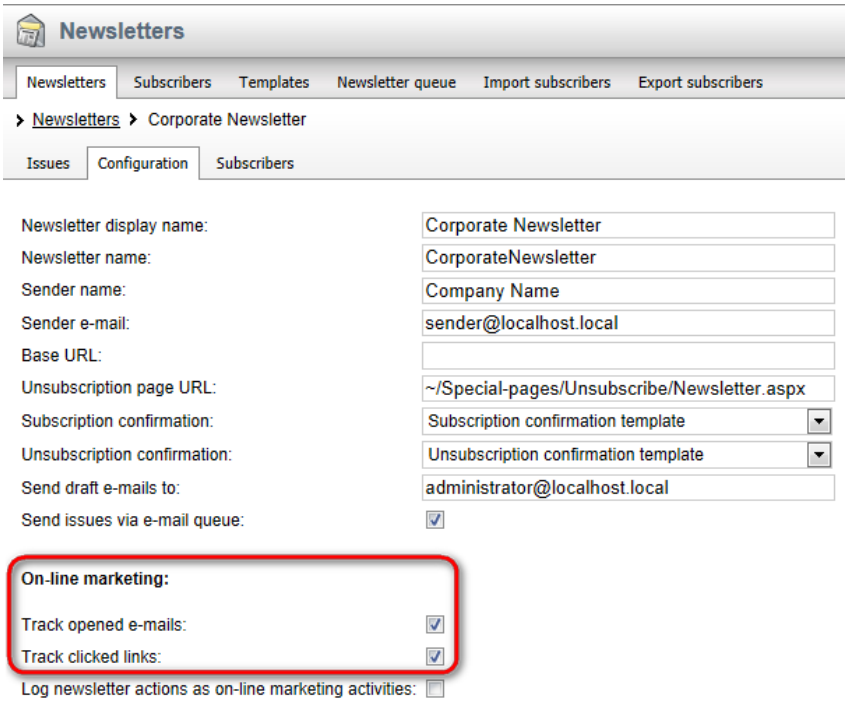
If your application is running in a medium trust environment, the bounced e-mails feature will not be functional by default.

For more details and a possible solution, please refer to Configuration for Medium Trust environment.

E-mail tracking

Monitoring of the e-mails that deliver individual issues to subscribers can be enabled for a newsletter by editing it 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.

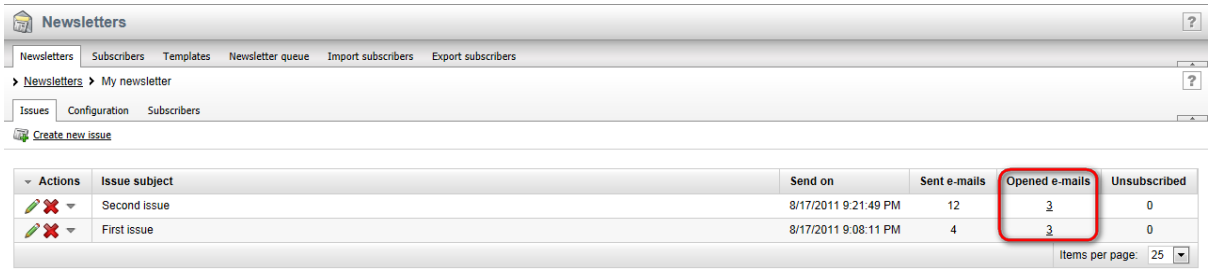
You can then keep track of the logged activities in the **CMS Desk -> On-line marketing -> Activities** interface.

Opened e-mails

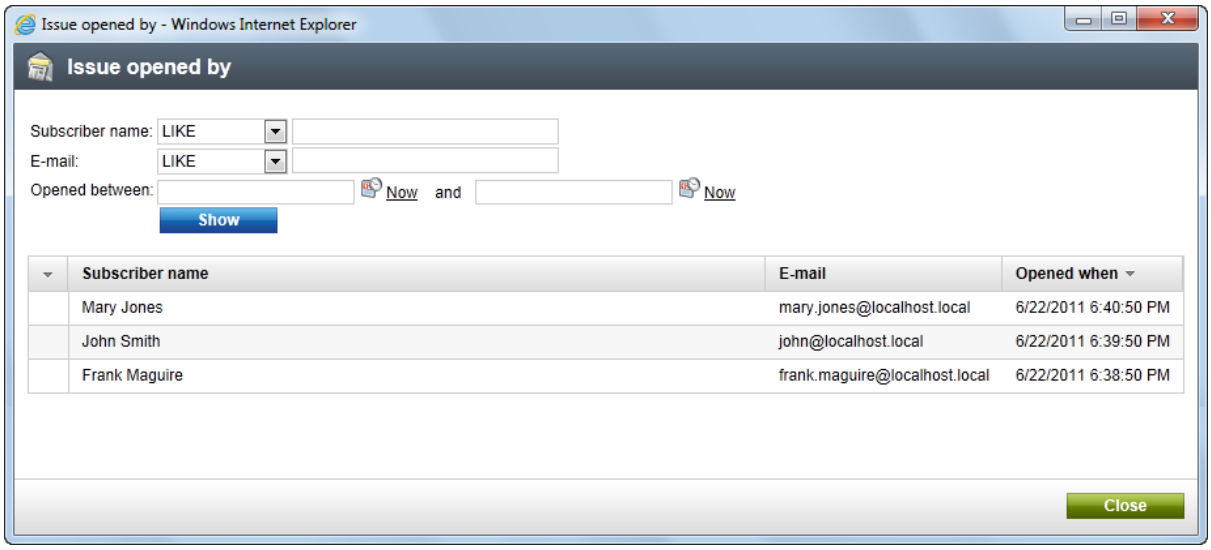
This feature allows you to monitor how many newsletter e-mails are actually opened by subscribers. Open rate is one of the key metrics for judging an e-mail campaign's success.

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.

When 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.

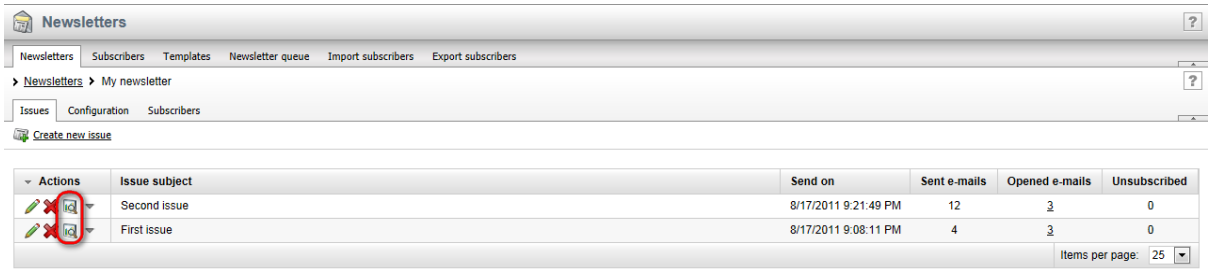


The subscribers can be filtered according to their name, e-mail address or the time interval during which they opened the e-mail.

Clicked links

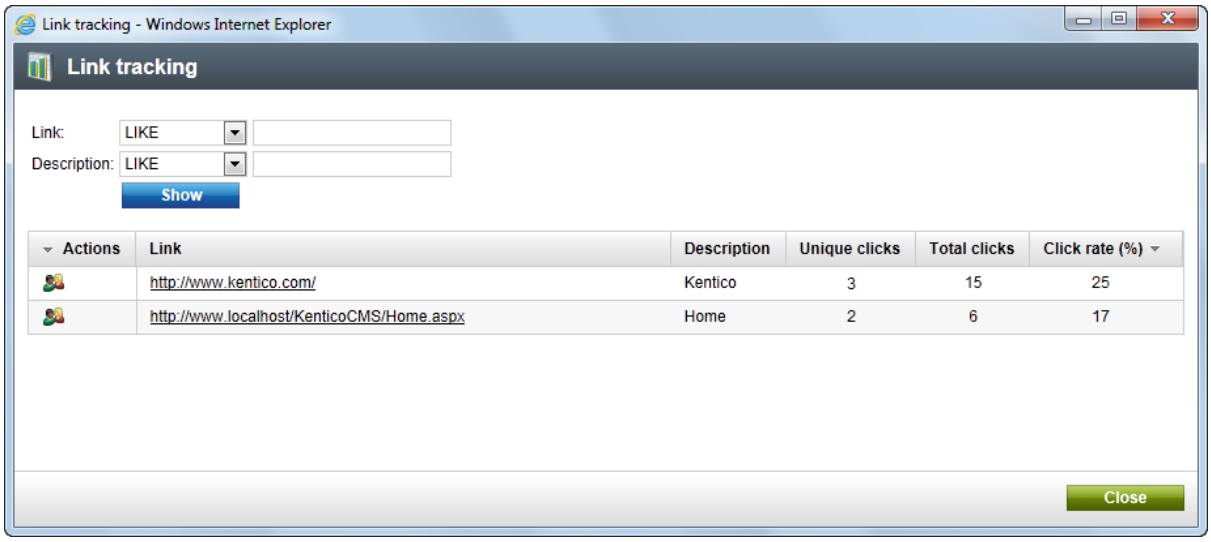
Another way to measure the effectiveness of an e-mail campaign is to track which links are clicked by the recipients and how many times, also known as the *clickthrough rate*.

When 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.

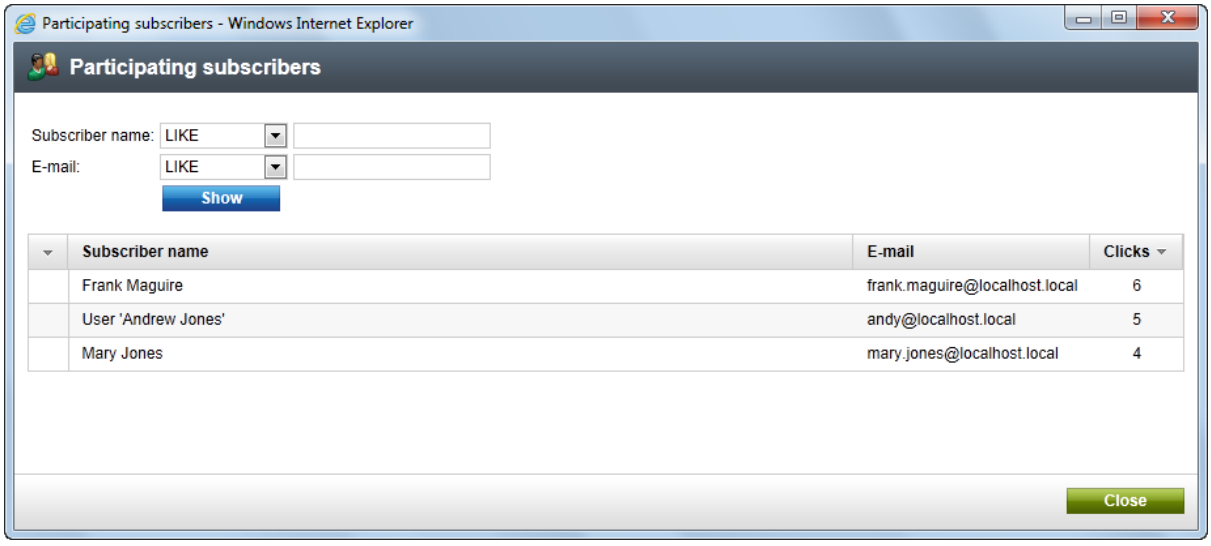


When clicked, this action opens a modal dialog that lists all links in the given issue and displays the following information about them:

- **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. The action opens another dialog containing the names and addresses of the subscribers that used the given link and their respective click count.



Links excluded 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:

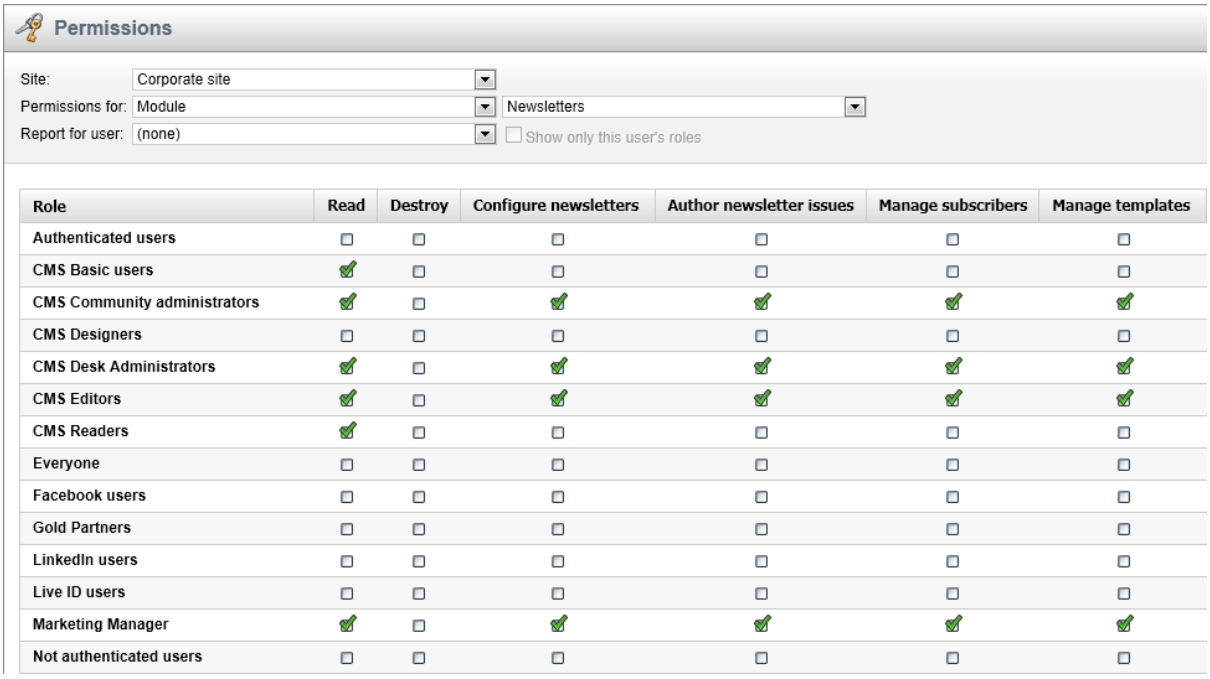
Kentico CMS

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.

8.34.11 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.
- **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.



Role	Read	Destroy	Configure newsletters	Author newsletter issues	Manage subscribers	Manage templates
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	☐	☐	☐	☐	☐	☐

8.34.12 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.

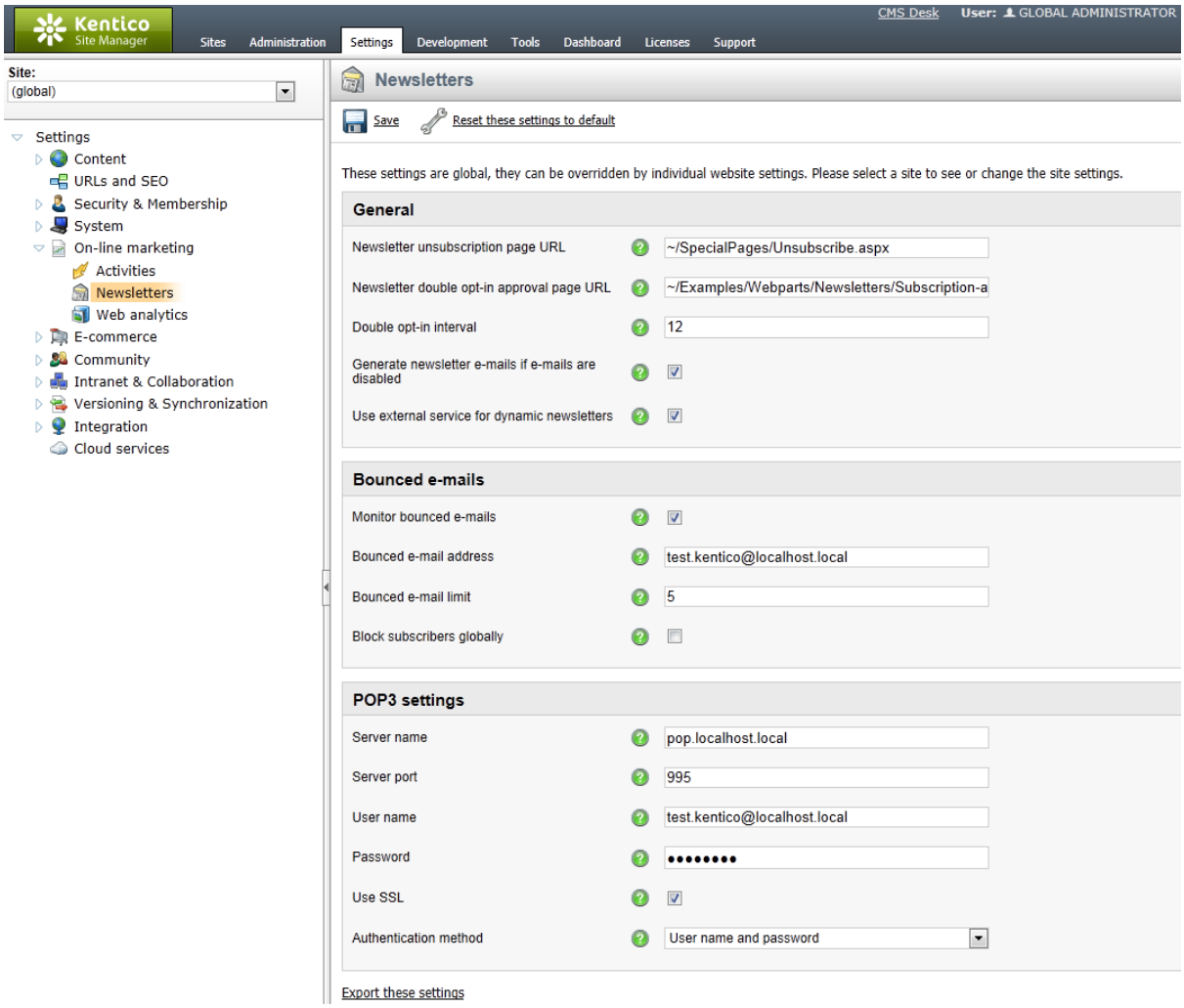
- **Newsletter double opt-in approval page URL** - sets the URL of the page where users can confirm their subscription to a newsletter with double opt-in enabled. The *Subscription approval* web part must be placed on the specified page to ensure the required functionality. The value of this setting 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.



8.34.13 Troubleshooting

Problems with 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 resent here.

4. Make sure you're 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 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 logging of all sent e-mails to *<web root>/AppData/logemails.log*. The second key ensures that all sent e-mails are logged, but not actually sent. This is helpful when you need to test the functionality but do not want the testing e-mails to actually be sent.

Problems with unsubscription links

If you encounter problems with unsubscription link resolving, you should check that the **Base URL** property of the newsletter is set. This property can be set at **CMS Desk -> Tools -> Newsletter**, choose to **Edit** () the newsletter and switch to its **Configuration** tab.

Problems with role 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, go to **CMS Desk -> Tools -> Newsletter**, choose to **Edit** () your newsletter, switch to its **Configuration** tab and enable the **Send issues via e-mail queue** property.

8.34.14 Newsletters internals and API

8.34.14.1 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.

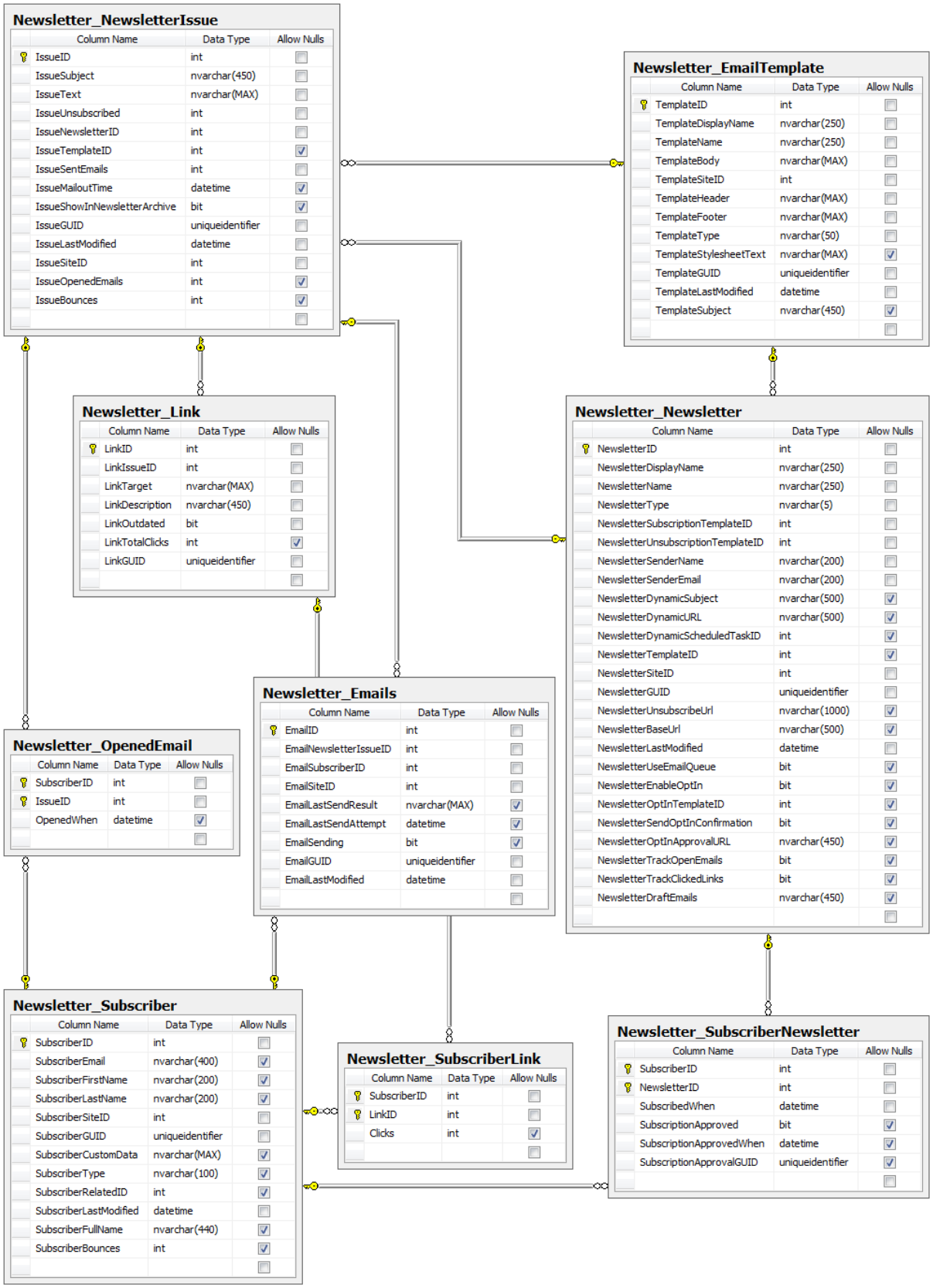
8.34.14.2 Database tables

The following database tables are used in 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_Emails** - this table stores newsletter e-mails could not be sent out successfully (i.e. the newsletter queue). This allows them to be resent 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.



8.34.14.3 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:

- **Newsletter** - represents on newsletter object.
- **NewsletterProvider** - provides management functionality for newsletters.

Newsletter_NewsletterIssue table API:

- **Issue** - represents one newsletter issue.
- **IssueProvider** - provides management functionality for newsletter issues.

Newsletter_Subscriber table API:

- **Subscriber** - represents one newsletter subscriber.
- **SubscriberProvider** - 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:

- **EmailTemplate** - represents one newsletter e-mail template.
- **EmailTemplateProvider** - provides management functionality for newsletter templates.

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.

8.34.14.4 API examples

8.34.14.4.1 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;
```

8.34.14.4.2 Managing email templates

The following example creates a subscription template.

```
private void CreateSubscriptionTemplate()
{
    // Create new subscription template object
```

```
    EmailTemplate newTemplate = new EmailTemplate();

    // Set 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;

    // Save the subscription template
    EmailTemplateProvider.SetEmailTemplate(newTemplate);
}
```

The following example creates an unsubscription template.

```
private void CreateUnsubscriptionTemplate()
{
    // Create new unsubscription template object
    EmailTemplate newTemplate = new EmailTemplate();

    // Set 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;

    // Save the unsubscription template
    EmailTemplateProvider.SetEmailTemplate(newTemplate);
}
```

The following example creates an issue template.

```
private void CreateIssueTemplate()
{
    // Create new issue template object
    EmailTemplate newTemplate = new EmailTemplate();

    // Set 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;

    // Save the issue template
    EmailTemplateProvider.SetEmailTemplate(newTemplate);
}
```

The following example gets and updates an issue template.

```
private bool GetAndUpdateIssueTemplate()
{
    // Get the issue template
    EmailTemplate updateTemplate = EmailTemplateProvider.GetEmailTemplate(
"MyNewIssueTemplate", CMSContext.CurrentSiteID);
    if (updateTemplate != null)
    {
        // Update the properties
        updateTemplate.TemplateDisplayName = updateTemplate.TemplateDisplayName.
ToLower();

        // Save the changes
        EmailTemplateProvider.SetEmailTemplate(updateTemplate);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates issue templates.

```
private bool GetAndBulkUpdateIssueTemplates()
{
    // Prepare the parameters
    string where = "TemplateName LIKE N'MyNewIssueTemplate%'";

    // Get the data
    DataSet templates = EmailTemplateProvider.GetEmailTemplates(where, null);
    if (!DataHelper.DataSourceIsEmpty(templates))
    {
        // Loop through the individual items
        foreach (DataRow templateDr in templates.Tables[0].Rows)
        {
            // Create object from DataRow
            EmailTemplate modifyTemplate = new EmailTemplate(templateDr);

            // Update the properties
```

```
            modifyTemplate.TemplateDisplayName = modifyTemplate.
TemplateDisplayName.ToUpper();

            // Save the changes
            EmailTemplateProvider.SetEmailTemplate(modifyTemplate);
        }

        return true;
    }

    return false;
}
```

The following example deletes a subscription template.

```
private bool DeleteSubscriptionTemplate()
{
    // Get the subscription template
    EmailTemplate deleteTemplate = EmailTemplateProvider.GetEmailTemplate(
"MyNewSubscriptionTemplate", CMSContext.CurrentSiteID);

    // Delete the subscription template
    EmailTemplateProvider.DeleteEmailTemplate(deleteTemplate);

    return (deleteTemplate != null);
}
```

The following example deletes an unsubscription template.

```
private bool DeleteUnsubscriptionTemplate()
{
    // Get the unsubscription template
    EmailTemplate deleteTemplate = EmailTemplateProvider.GetEmailTemplate(
"MyNewUnsubscriptionTemplate", CMSContext.CurrentSiteID);

    // Delete the unsubscription template
    EmailTemplateProvider.DeleteEmailTemplate(deleteTemplate);

    return (deleteTemplate != null);
}
```

The following example deletes an issue template.

```
private bool DeleteIssueTemplate()
{
    // Get the issue template
    EmailTemplate deleteTemplate = EmailTemplateProvider.GetEmailTemplate(
"MyNewIssueTemplate", CMSContext.CurrentSiteID);
```

```
    // Delete the issue template
    EmailTemplateProvider.DeleteEmailTemplate(deleteTemplate);

    return (deleteTemplate != null);
}
```

8.34.14.4.3 Managing new sletters

The following example creates a static newsletter.

```
private bool CreateStaticNewsletter()
{
    EmailTemplate subscriptionTemplate = EmailTemplateProvider.GetEmailTemplate(
"MyNewSubscriptionTemplate", CMSContext.CurrentSiteID);
    EmailTemplate unsubscriptionTemplate = EmailTemplateProvider.GetEmailTemplate(
"MyNewUnsubscriptionTemplate", CMSContext.CurrentSiteID);
    EmailTemplate myNewIssueTemplate = EmailTemplateProvider.GetEmailTemplate(
"MyNewIssueTemplate", CMSContext.CurrentSiteID);

    if ((subscriptionTemplate != null) && (unsubscriptionTemplate != null) &&
(myNewIssueTemplate != null))
    {
        // Create new static newsletter object
        Newsletter newNewsletter = new Newsletter();

        // Set 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;

        // Save the static newsletter
        NewsletterProvider.SetNewsletter(newNewsletter);

        return true;
    }

    return false;
}
```

The following example gets and updates a static newsletter.

```
private bool GetAndUpdateStaticNewsletter()
{
    // Get the static newsletter
    Newsletter updateNewsletter = NewsletterProvider.GetNewsletter(
"MyNewStaticNewsletter", CMSContext.CurrentSiteID);
    if (updateNewsletter != null)
    {
        // Update the properties
        updateNewsletter.NewsletterDisplayName = updateNewsletter.
NewsletterDisplayName.ToLower();

        // Save the changes
        NewsletterProvider.SetNewsletter(updateNewsletter);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates static newsletters.

```
private bool GetAndBulkUpdateStaticNewsletters()
{
    // Prepare the parameters
    string where = "NewsletterName LIKE N'MyNewStaticNewsletter%'";

    // Get the data
    DataSet newsletters = NewsletterProvider.GetNewsletters(where, null, 0, null);
    if (!DataHelper.DataSourceIsEmpty(newsletters))
    {
        // Loop through the individual items
        foreach (DataRow newsletterDr in newsletters.Tables[0].Rows)
        {
            // Create object from DataRow
            Newsletter modifyNewsletter = new Newsletter(newsletterDr);

            // Update the properties
            modifyNewsletter.NewsletterDisplayName = modifyNewsletter.
NewsletterDisplayName.ToUpper();

            // Save the changes
            NewsletterProvider.SetNewsletter(modifyNewsletter);
        }

        return true;
    }

    return false;
}
```

The following example creates a dynamic newsletter.

```
private bool CreateDynamicNewsletter()
{
    EmailTemplate subscriptionTemplate = EmailTemplateProvider.GetEmailTemplate(
"MyNewSubscriptionTemplate", CMSContext.CurrentSiteID);
    EmailTemplate unsubscriptionTemplate = EmailTemplateProvider.GetEmailTemplate(
"MyNewUnsubscriptionTemplate", CMSContext.CurrentSiteID);

    if ((subscriptionTemplate != null) && (unsubscriptionTemplate != null))
    {
        // Create new dynamic newsletter object
        Newsletter newNewsletter = new Newsletter();

        // Set 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;

        // Save the dynamic newsletter
        NewsletterProvider.SetNewsletter(newNewsletter);

        return true;
    }

    return false;
}
```

The following example gets and updates a dynamic newsletter.

```
private bool GetAndUpdateDynamicNewsletter()
{
    // Get the dynamic newsletter
    Newsletter updateNewsletter = NewsletterProvider.GetNewsletter(
"MyNewDynamicNewsletter", CMSContext.CurrentSiteID);
    if (updateNewsletter != null)
    {
        // Update the properties
        updateNewsletter.NewsletterDisplayName = updateNewsletter.
NewsletterDisplayName.ToLower();

        // Save the changes
        NewsletterProvider.SetNewsletter(updateNewsletter);
```

```
        return true;
    }

    return false;
}
```

The following example gets and bulk updates dynamic newsletters.

```
private bool GetAndBulkUpdateDynamicNewsletters()
{
    // Prepare the parameters
    string where = "NewsletterName LIKE N'MyNewDynamicNewsletter%'";

    // Get the data
    DataSet newsletters = NewsletterProvider.GetNewsletters(where, null, 0, null);
    if (!DataHelper.DataSourceIsEmpty(newsletters))
    {
        // Loop through the individual items
        foreach (DataRow newsletterDr in newsletters.Tables[0].Rows)
        {
            // Create object from DataRow
            Newsletter modifyNewsletter = new Newsletter(newsletterDr);

            // Update the properties
            modifyNewsletter.NewsletterDisplayName = modifyNewsletter.
NewsletterDisplayName.ToUpper();

            // Save the changes
            NewsletterProvider.SetNewsletter(modifyNewsletter);
        }

        return true;
    }

    return false;
}
```

The following example deletes a dynamic newsletter.

```
private bool DeleteDynamicNewsletter()
{
    // Get the dynamic newsletter
    Newsletter deleteNewsletter = NewsletterProvider.GetNewsletter(
"MyNewDynamicNewsletter", CMSContext.CurrentSiteID);

    // Delete the dynamic newsletter
    NewsletterProvider.DeleteNewsletter(deleteNewsletter);

    return (deleteNewsletter != null);
}
```

The following example deletes a static newsletter.

```
private bool DeleteStaticNewsletter()
{
    // Get the static newsletter
    Newsletter deleteNewsletter = NewsletterProvider.GetNewsletter(
"MyNewStaticNewsletter", CMSContext.CurrentSiteID);

    // Delete the static newsletter
    NewsletterProvider.DeleteNewsletter(deleteNewsletter);

    return (deleteNewsletter != null);
}
```

8.34.14.4.4 Managing subscribers

The following example creates a subscriber.

```
private bool CreateSubscriber()
{
    // Create new subscriber object
    Subscriber newSubscriber = new Subscriber();

    // Set the properties
    newSubscriber.SubscriberFirstName = "Name";
    newSubscriber.SubscriberLastName = "Surname";
    newSubscriber.SubscriberFullName = "Name Surname";
    newSubscriber.SubscriberEmail = "subscriber@localhost.local";
    newSubscriber.SubscriberSiteID = CMSContext.CurrentSiteID;

    // Save the subscriber
    SubscriberProvider.SetSubscriber(newSubscriber);

    return true;
}
```

The following example gets and updates a subscriber.

```
private bool GetAndUpdateSubscriber()
{
    // Get the subscriber
    Subscriber updateSubscriber = SubscriberProvider.GetSubscriber(
"subscriber@localhost.local", CMSContext.CurrentSiteID);
    if (updateSubscriber != null)
    {
        // Update the properties
        updateSubscriber.SubscriberFullName = updateSubscriber.SubscriberFullName.
ToLower();
```

```
        // Save the changes
        SubscriberProvider.SetSubscriber(updateSubscriber);

        return true;
    }

    return false;
}
```

The following example gets and bulk updates subscribers.

```
private bool GetAndBulkUpdateSubscribers()
{
    // Prepare the parameters
    string where = "SubscriberEmail LIKE N'subscriber@localhost.local%'";

    // Get the data
    DataSet subscribers = SubscriberProvider.GetSubscribers(where, null);
    if (!DataHelper.DataSourceIsEmpty(subscribers))
    {
        // Loop through the individual items
        foreach (DataRow subscriberDr in subscribers.Tables[0].Rows)
        {
            // Create object from DataRow
            Subscriber modifySubscriber = new Subscriber(subscriberDr);

            // Update the properties
            modifySubscriber.SubscriberFullName = modifySubscriber.
SubscriberFullName.ToUpper();

            // Save the changes
            SubscriberProvider.SetSubscriber(modifySubscriber);
        }

        return true;
    }

    return false;
}
```

The following example subscribes a subscriber to newsletter.

```
private bool SubscribeToNewsletter()
{
    // Get the subscriber and newsletter
    Subscriber subscriber = SubscriberProvider.GetSubscriber(
"subscriber@localhost.local", CMSContext.CurrentSiteID);
    Newsletter newsletter = NewsletterProvider.GetNewsletter(
"MyNewStaticNewsletter", CMSContext.CurrentSiteID);
```

```
    if ((subscriber != null) && (newsletter != null))
    {
        // Subscribe to 'My new static newsletter'
        SubscriberProvider.Subscribe(subscriber.SubscriberID, newsletter.
NewsletterID, DateTime.Now);

        return true;
    }

    return false;
}
```

The following example unsubscribes a subscriber from newsletter.

```
private bool UnsubscribeFromNewsletter()
{
    // Get the subscriber and newsletter
    Subscriber subscriber = SubscriberProvider.GetSubscriber(
"subscriber@localhost.local", CMSContext.CurrentSiteID);
    Newsletter newsletter = NewsletterProvider.GetNewsletter(
"MyNewStaticNewsletter", CMSContext.CurrentSiteID);

    if ((subscriber != null) && (newsletter != null))
    {
        // Unubscribe from 'My new static newsletter'
        SubscriberProvider.Unsubscribe(subscriber.SubscriberID, newsletter.
NewsletterID);

        return true;
    }

    return false;
}
```

The following example deletes a subscriber.

```
private bool DeleteSubscriber()
{
    // Get the subscriber
    Subscriber deleteSubscriber = SubscriberProvider.GetSubscriber(
"subscriber@localhost.local", CMSContext.CurrentSiteID);

    // Delete the subscriber
    SubscriberProvider.DeleteSubscriber(deleteSubscriber);

    return (deleteSubscriber != null);
}
```

8.34.14.4.5 Managing issues

The following example creates a static issue.

```
private bool CreateStaticIssue()
{
    // Get the newsletter
    Newsletter newsletter = NewsletterProvider.GetNewsletter(
"MyNewStaticNewsletter", CMSContext.CurrentSiteID);

    if (newsletter != null)
    {
        // Create new static issue object
        Issue newIssue = new Issue();

        // Set 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;

        // Save the static issue
        IssueProvider.SetIssue(newIssue);

        return true;
    }

    return false;
}
```

The following example gets and updates a static issue.

```
private bool GetAndUpdateStaticIssue()
{
    // Get the newsletter
    Newsletter newsletter = NewsletterProvider.GetNewsletter(
"MyNewStaticNewsletter", CMSContext.CurrentSiteID);

    if (newsletter != null)
    {
        // Prepare the parameters
        string where = "IssueNewsletterID = " + newsletter.NewsletterID;

        // Get the data
        DataSet issues = IssueProvider.GetIssues(where, null);

        if (!DataHelper.DataSourceIsEmpty(issues))
```

```
        {
            // Create object from DataRow
            Issue updateIssue = new Issue(issues.Tables[0].Rows[0]);

            if (updateIssue != null)
            {
                // Update the properties
                updateIssue.IssueSubject = updateIssue.IssueSubject.ToLower();

                // Save the changes
                IssueProvider.SetIssue(updateIssue);

                return true;
            }
        }
    }
    return false;
}
```

The following example gets and bulk updates static issues.

```
private bool GetAndBulkUpdateStaticIssues()
{
    // Get the newsletter
    Newsletter newsletter = NewsletterProvider.GetNewsletter(
"MyNewStaticNewsletter", CMSContext.CurrentSiteID);

    if (newsletter != null)
    {
        // Prepare the parameters
        string where = "IssueNewsletterID = " + newsletter.NewsletterID;

        // Get the data
        DataSet issues = IssueProvider.GetIssues(where, null);
        if (!DataHelper.DataSourceIsEmpty(issues))
        {
            // Loop through the individual items
            foreach (DataRow issueDr in issues.Tables[0].Rows)
            {
                // Create object from DataRow
                Issue modifyIssue = new Issue(issueDr);

                // Update the properties
                modifyIssue.IssueSubject = modifyIssue.IssueSubject.ToUpper();

                // Save the changes
                IssueProvider.SetIssue(modifyIssue);
            }

            return true;
        }
    }
```

```
    return false;
}
```

The following example creates a dynamic issue.

```
private bool CreateDynamicIssue()
{
    // Get the newsletter
    Newsletter newsletter = NewsletterProvider.GetNewsletter(
"MyNewDynamicNewsletter", CMSContext.CurrentSiteID);

    if (newsletter != null)
    {
        // Generate dynamic issue
        EmailQueueManager.GenerateDynamicIssue(newsletter.NewsletterID);

        return true;
    }

    return false;
}
```

The following example gets and updates a dynamic issue.

```
private bool GetAndUpdateDynamicIssue()
{
    // Get the newsletter
    Newsletter newsletter = NewsletterProvider.GetNewsletter(
"MyNewDynamicNewsletter", CMSContext.CurrentSiteID);

    if (newsletter != null)
    {
        // Prepare the parameters
        string where = "IssueNewsletterID = " + newsletter.NewsletterID;

        // Get the data
        DataSet issues = IssueProvider.GetIssues(where, null);

        if (!DataHelper.DataSourceIsEmpty(issues))
        {
            // Create object from DataRow
            Issue updateIssue = new Issue(issues.Tables[0].Rows[0]);

            if (updateIssue != null)
            {
                // Update the properties
                updateIssue.IssueSubject = updateIssue.IssueSubject.ToLower();

                // Save the changes
```

```
                IssueProvider.SetIssue(updateIssue);

                return true;
            }
        }
    }
    return false;
}
```

The following example gets and bulk updates dynamic issues.

```
private bool GetAndBulkUpdateDynamicIssues()
{
    // Get the newsletter
    Newsletter newsletter = NewsletterProvider.GetNewsletter(
"MyNewDynamicNewsletter", CMSContext.CurrentSiteID);

    if (newsletter != null)
    {
        // Prepare the parameters
        string where = "IssueNewsletterID = " + newsletter.NewsletterID;

        // Get the data
        DataSet issues = IssueProvider.GetIssues(where, null);
        if (!DataHelper.DataSourceIsEmpty(issues))
        {
            // Loop through the individual items
            foreach (DataRow issueDr in issues.Tables[0].Rows)
            {
                // Create object from DataRow
                Issue modifyIssue = new Issue(issueDr);

                // Update the properties
                modifyIssue.IssueSubject = modifyIssue.IssueSubject.ToUpper();

                // Save the changes
                IssueProvider.SetIssue(modifyIssue);
            }

            return true;
        }
    }
    return false;
}
```

The following example deletes a dynamic issue.

```
private bool DeleteDynamicIssue()
{
    // Get the newsletter
    Newsletter newsletter = NewsletterProvider.GetNewsletter(
```

```
"MyNewDynamicNewsletter", CMSContext.CurrentSiteID);

    if (newsletter != null)
    {
        // Prepare the parameters
        string where = "IssueNewsletterID = " + newsletter.NewsletterID;

        // Get the data
        DataSet issues = IssueProvider.GetIssues(where, null);

        if (!DataHelper.DataSourceIsEmpty(issues))
        {
            // Create object from DataRow
            Issue deleteIssue = new Issue(issues.Tables[0].Rows[0]);

            // Delete the dynamic issue
            IssueProvider.DeleteIssue(deleteIssue);

            return (deleteIssue != null);
        }
    }
    return false;
}
```

The following example deletes a static issue.

```
private bool DeleteStaticIssue()
{
    // Get the newsletter
    Newsletter newsletter = NewsletterProvider.GetNewsletter(
"MyNewStaticNewsletter", CMSContext.CurrentSiteID);

    if (newsletter != null)
    {
        // Prepare the parameters
        string where = "IssueNewsletterID = " + newsletter.NewsletterID;

        // Get the data
        DataSet issues = IssueProvider.GetIssues(where, null);

        if (!DataHelper.DataSourceIsEmpty(issues))
        {
            // Create object from DataRow
            Issue deleteIssue = new Issue(issues.Tables[0].Rows[0]);

            // Delete the static issue
            IssueProvider.DeleteIssue(deleteIssue);

            return (deleteIssue != null);
        }
    }
    return false;
```

```
}
```

8.35 Notifications

8.35.1 Overview

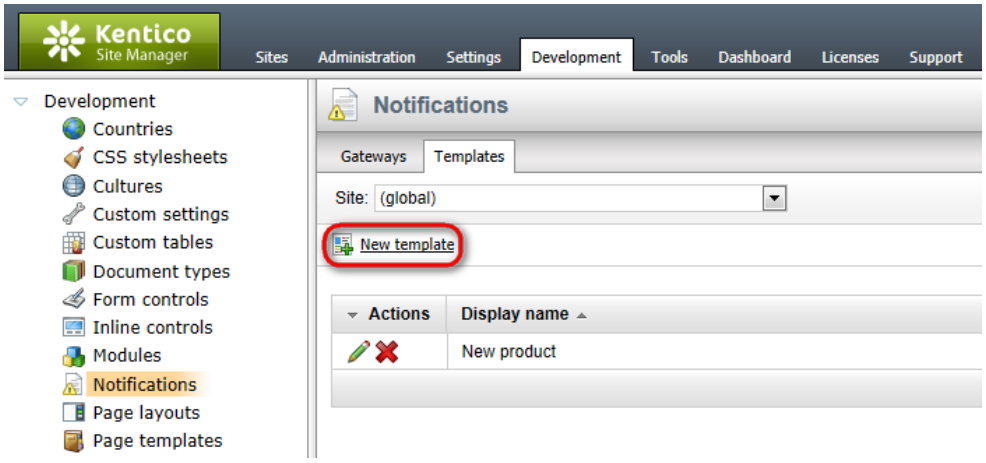
The Notifications module enables sending of notification messages using notification providers. Kentico CMS comes with a built-in e-mail notification gateway. However, you can create your custom gateways, such as an SMS or ICQ provider.

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.

8.35.2 Creating a notification message template

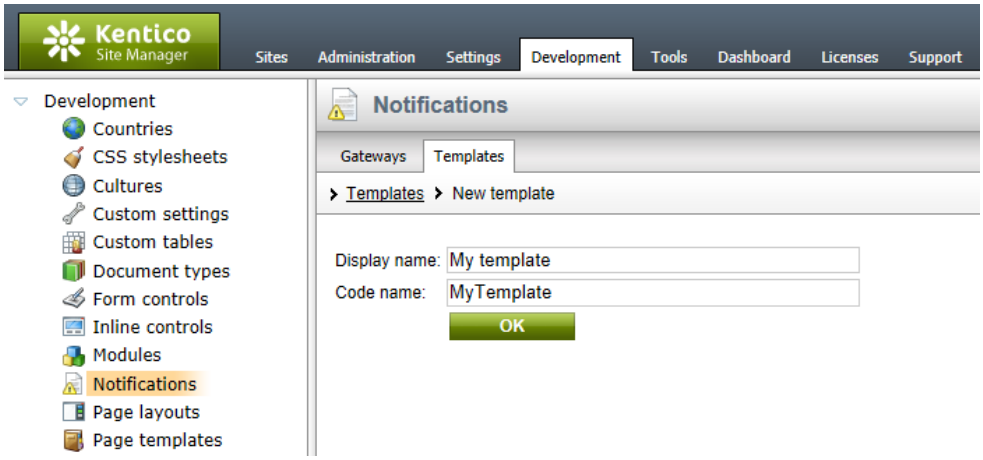
1. Go to **Site Manager -> Development -> Notifications** and switch to the **Templates** tab. Using the **Sites** drop-down list, you can select which site will be the template used by. If you choose **(global)**, the template will be available for all sites. When selected, click the **New template** link. 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.



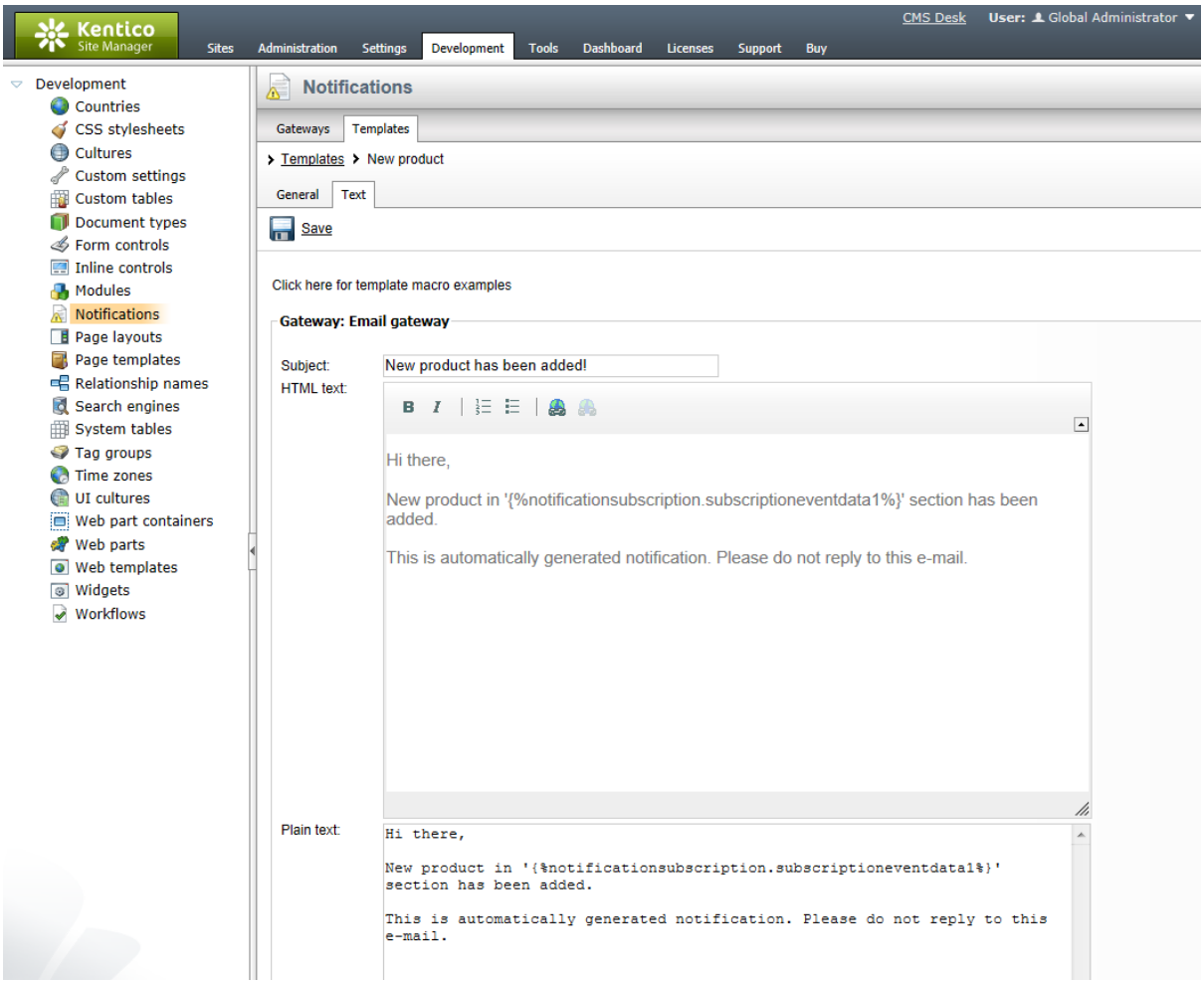
3. Now you should see the same form that you've just filled, but in the **General** tab, while another tab called **Text** is also available. Switch to the **Text** tab.

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. Columns from *View_CMS_Tree_Joined* and the document type's table can be used (e. g. *CONTENT_article* for *CMS.article* document type).
- **{%documentlink%}** - displays the link to the document (for content event notifications only)

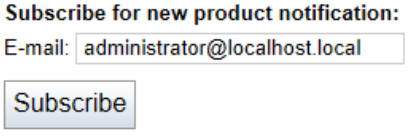


When you have entered appropriate text for each available gateway, click **Save**.

8.35.3 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 **zoneLeft** web part zone, below the **Left tree 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:

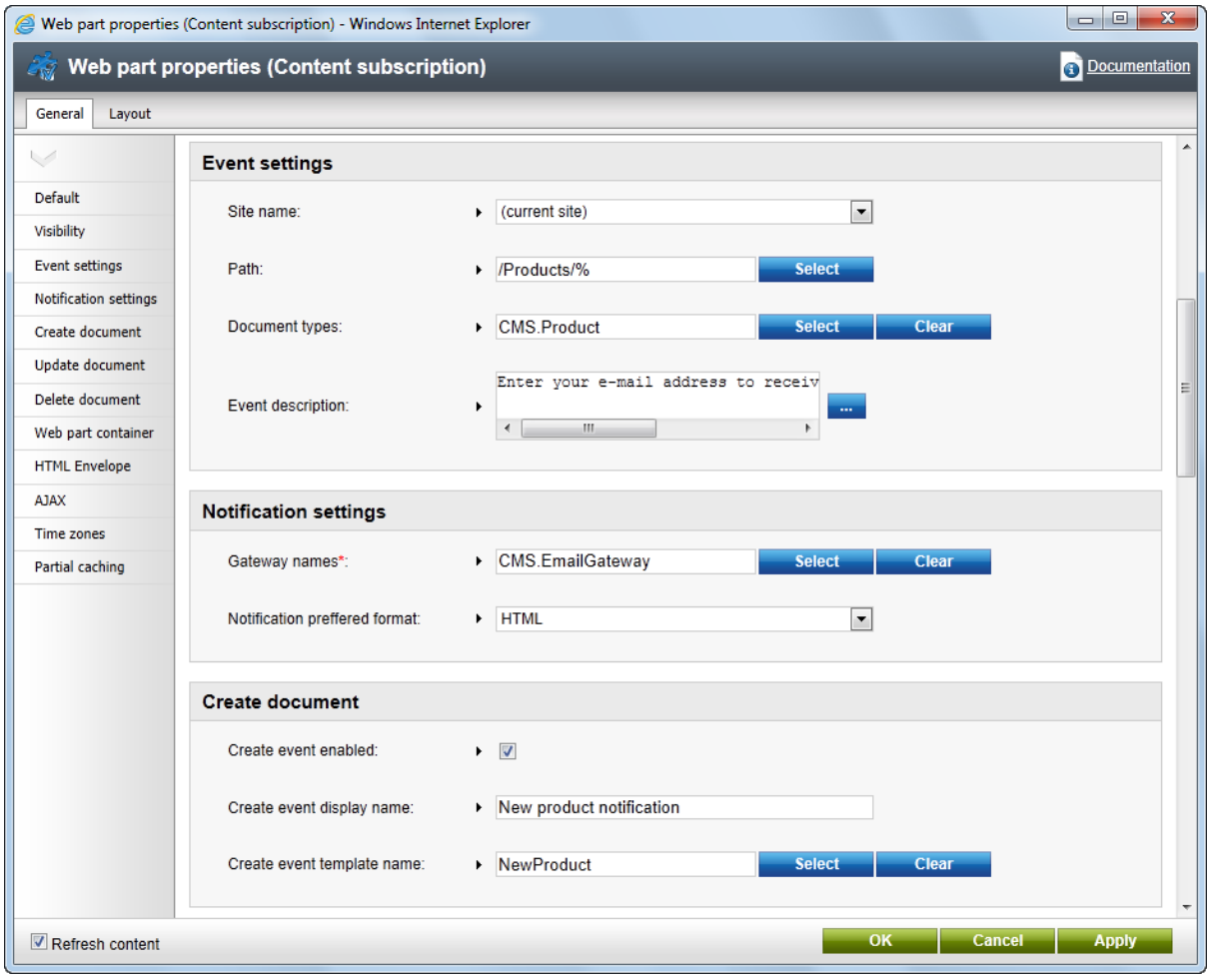
- **Gateway names** - CMS.EmailGateway
- **Notification preferred format** - HTML

- **Create event enabled** - true
- **Create event display name** - New product notification
- **Create event template name** - MyTemplate

and click **OK**.

Please note

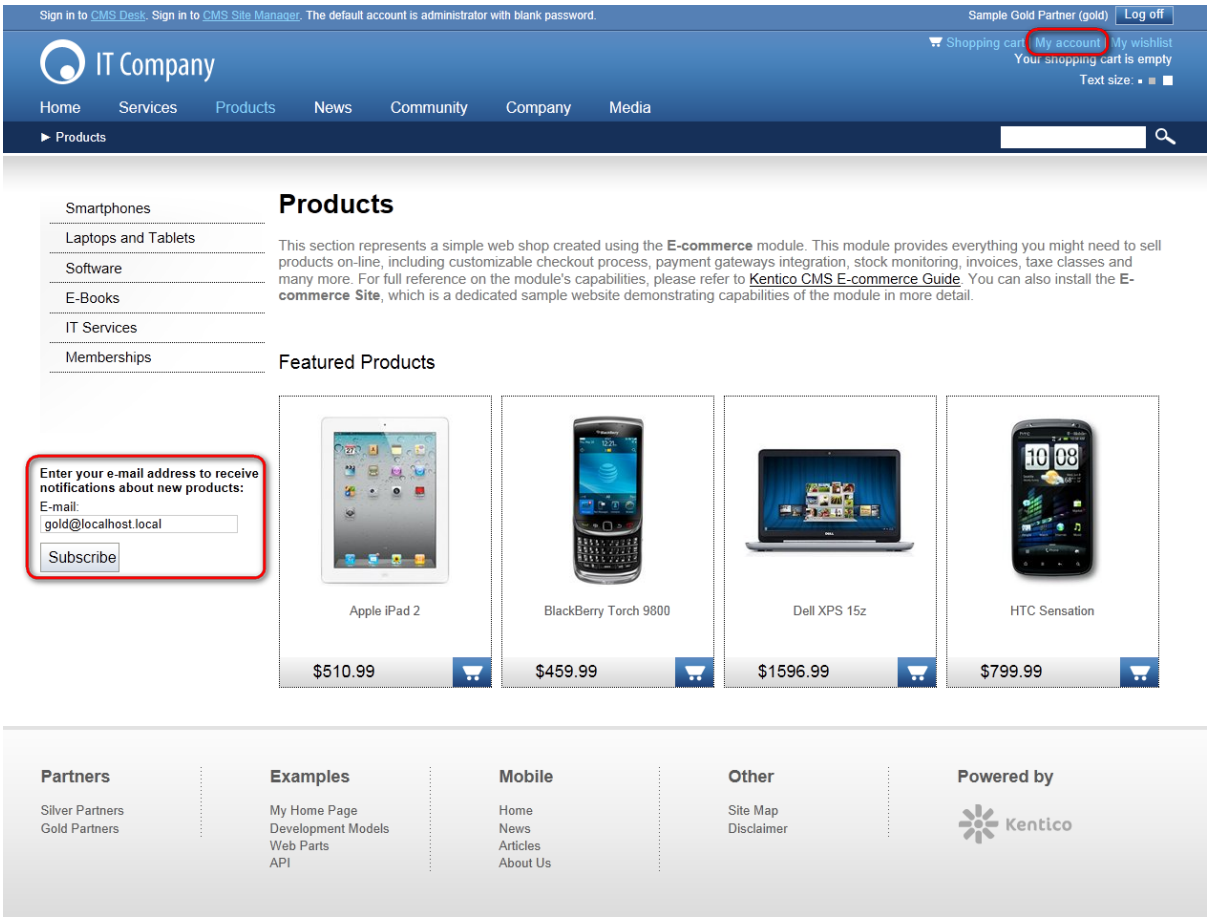
It is important that the template you have recently created according to the *Creating a notification message template* topic is global or assigned to the current site.



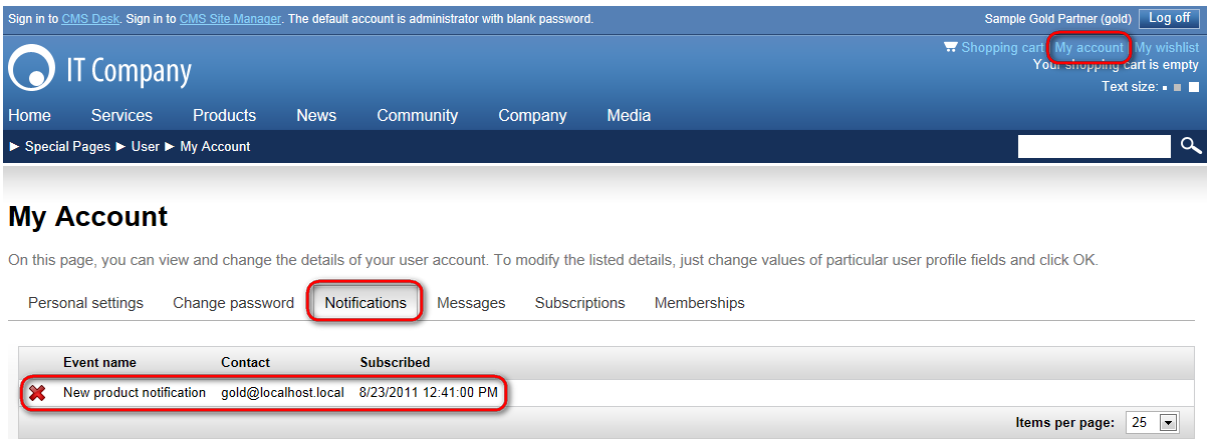
3. Let's switch to the **site visitor's** perspective now. Sign out of **CMS Desk**.

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.



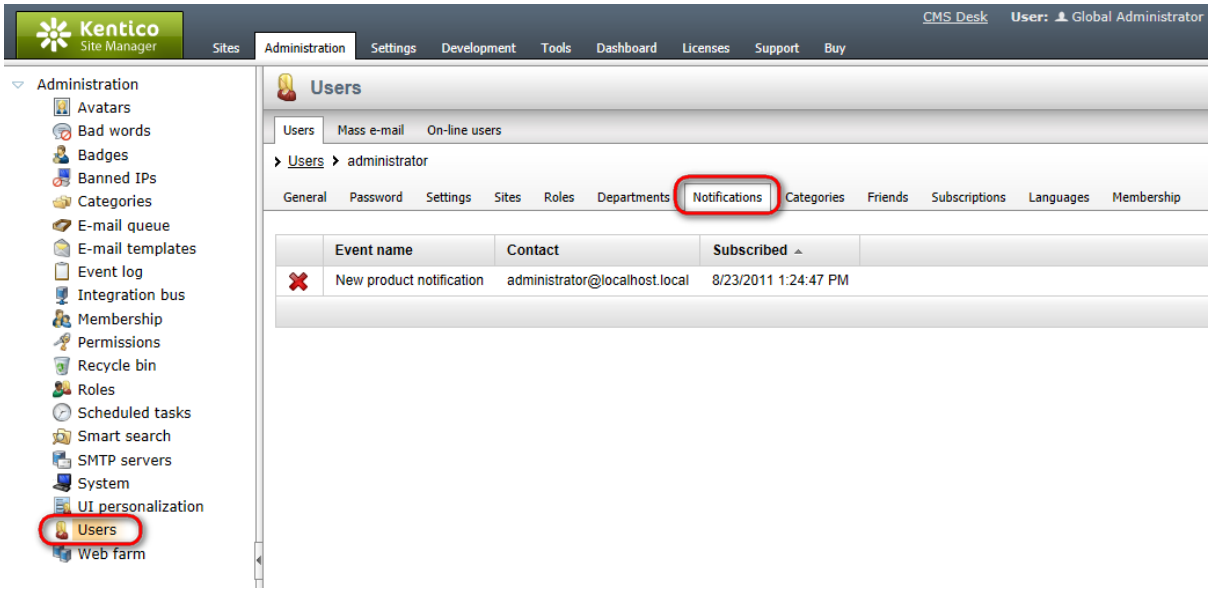
4. You can immediately verify your subscription by going to the **My account** section again. If you switch to the **Notifications** section of the **My account** web part, you should see the notification subscription present.



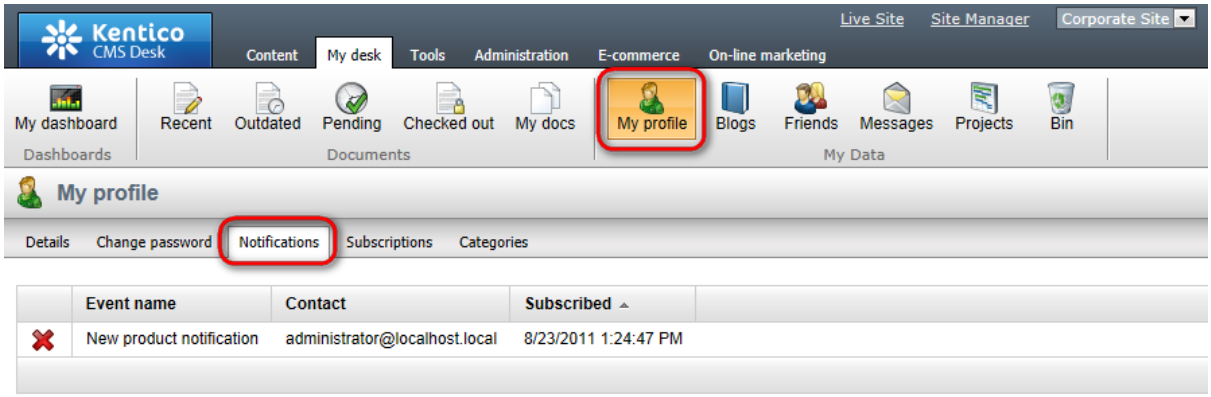
5. Now you can verify that the whole setup works. Log in to **CMS Desk** as the *administrator* again. From the content tree, select **Products** and click the **New** icon above the content tree. Choose to create some new product, e.g. a Cell phone, enter some sample data about the cell phone and click **Save**. Check your e-mail inbox in a few minutes. You should have received a new notification message about the newly added product.

8.35.4 Managing users' notifications

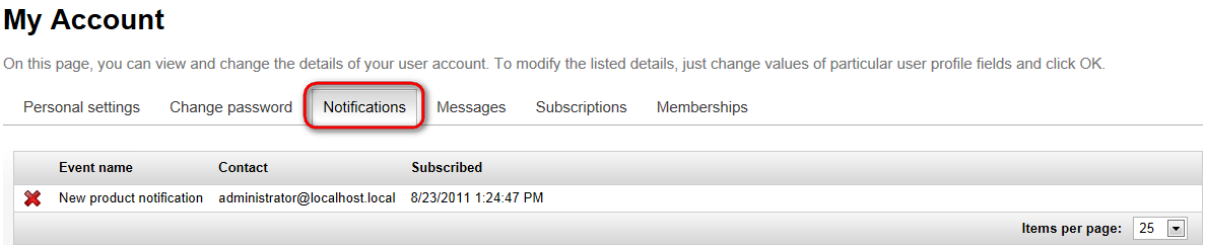
Site administrators can manage subscriptions of particular users in **Site Manager -> Administrator -> Users**. If you choose to **Edit** (✎) a user and switch to the **Notifications** tab, you will see all notifications that the current user is subscribed to. You can unsubscribe the user from a notification 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.



8.35.5 Custom notification gateway

8.35.5.1 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.

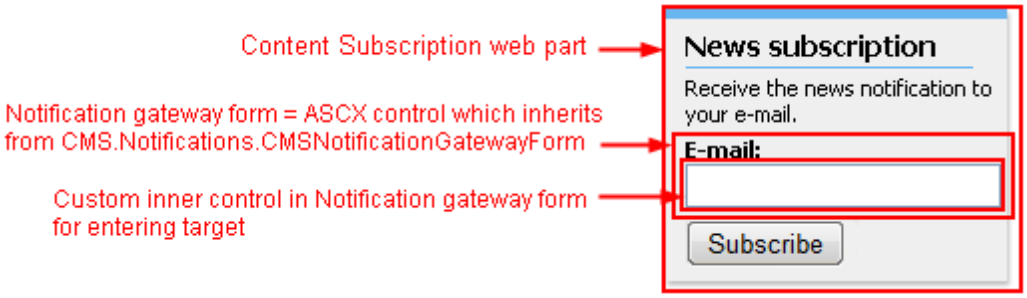
General steps

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.

8.35.5.2 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
}
```

8.35.5.3 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.
GatewaySupportsHTMLText) &&
                                (templateText.TemplateHTMLText != ""))
                            {
                                // HTML format, set Body property, resolve macros
if possible
```

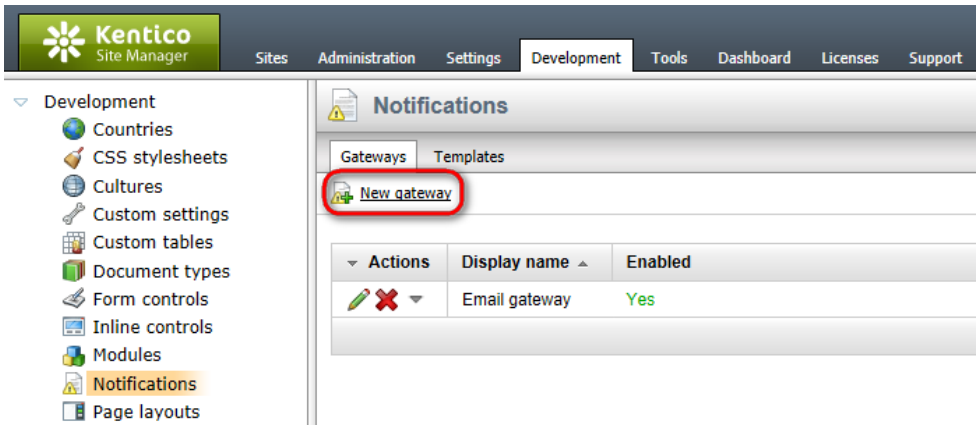
```
                                message.EmailFormat = EmailFormatEnum.Html;
                                if (this.Resolver != null)
                                {
                                    message.Body = this.Resolver.ResolveMacros
(templateText.TemplateHTMLText);
                                }
                                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
                }
            }
        }
    }
}
```

8.35.5.4 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:

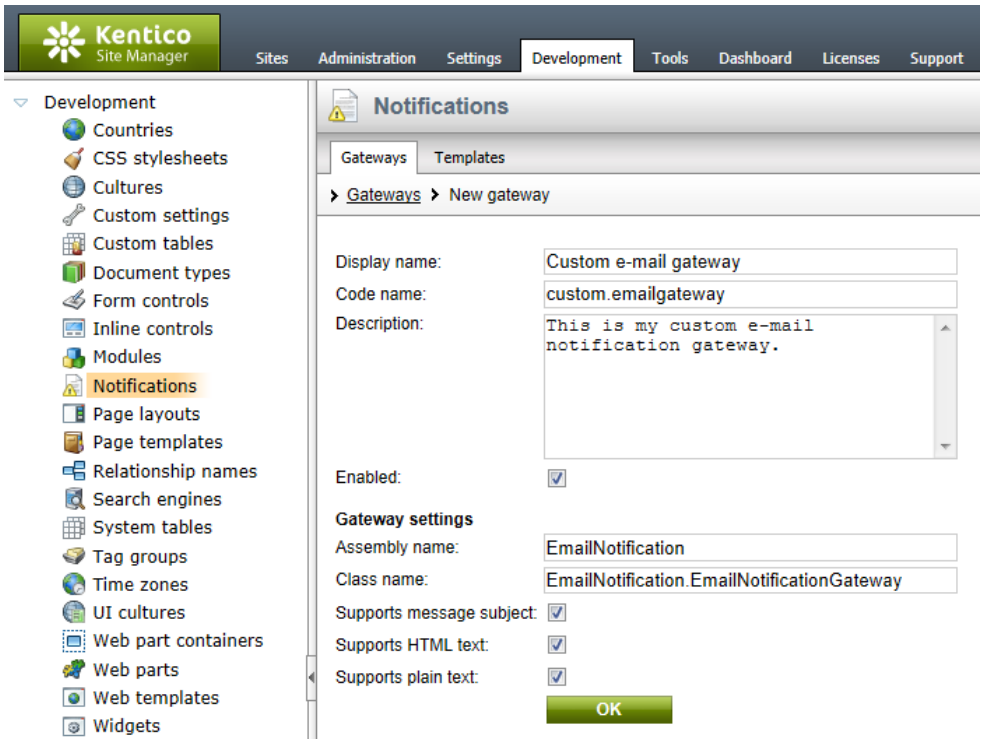
1. Go to **Site Manager -> Development -> Notifications**. On the **Gateways** tab, you can find the **New gateway** link at the top of the page. Click it, the **New gateway form** will be displayed.



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

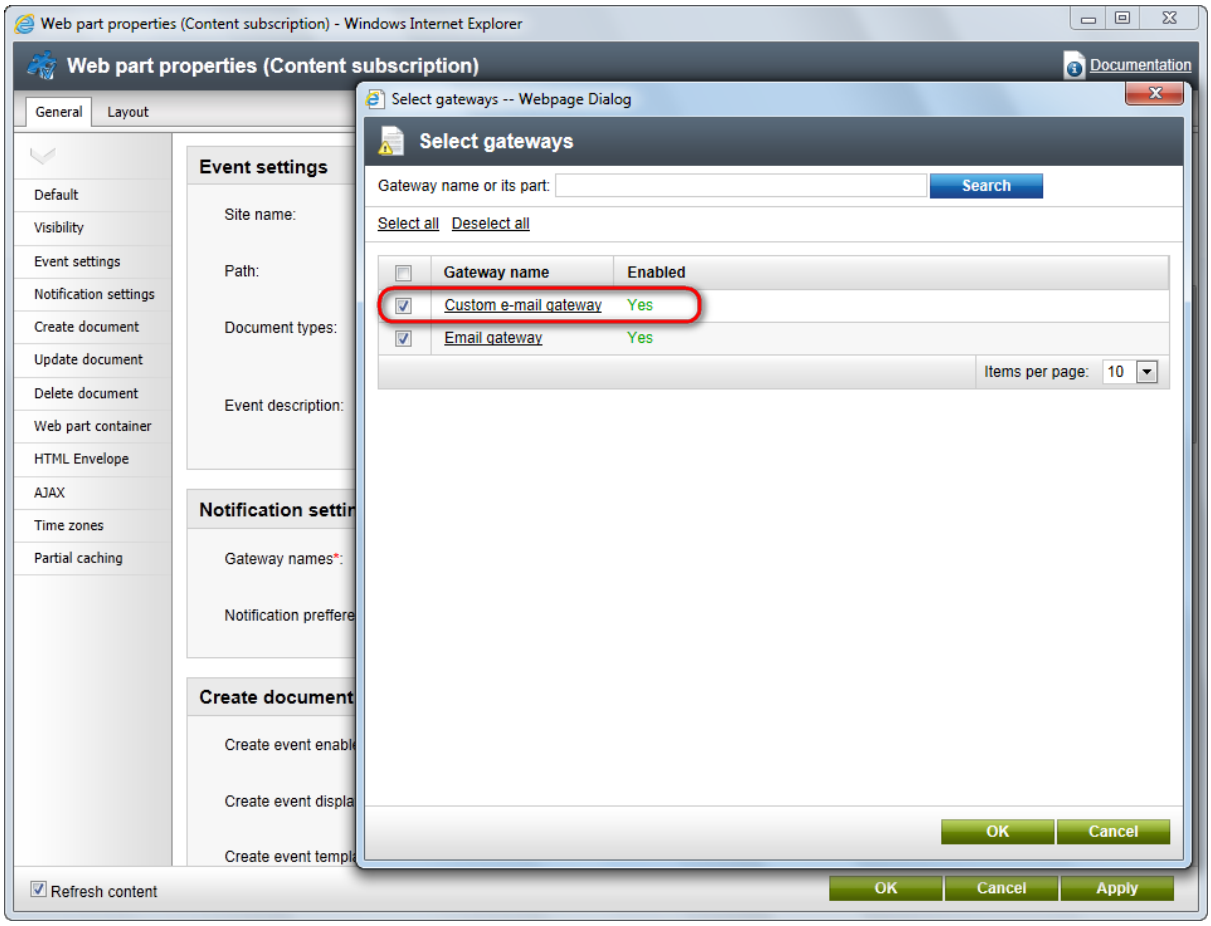
- **Assembly name** - name of the assembly in that the gateway code is stored
- **Class name** - name of the class containing the gateway code; must be entered including the assembly name, as you can see in the screenshot below
- **Supports message subject** - enable if the gateway's message format supports 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)



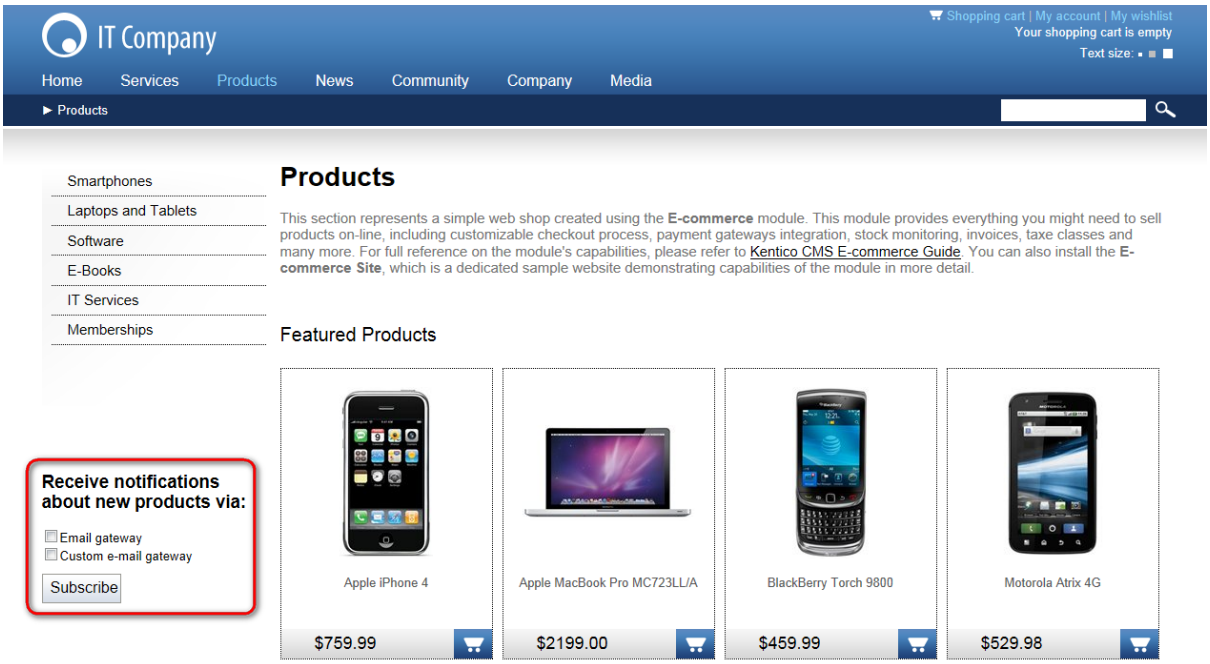
3. After entering all the details, click **OK** to confirm the changes you have made.

8.35.5.5 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.



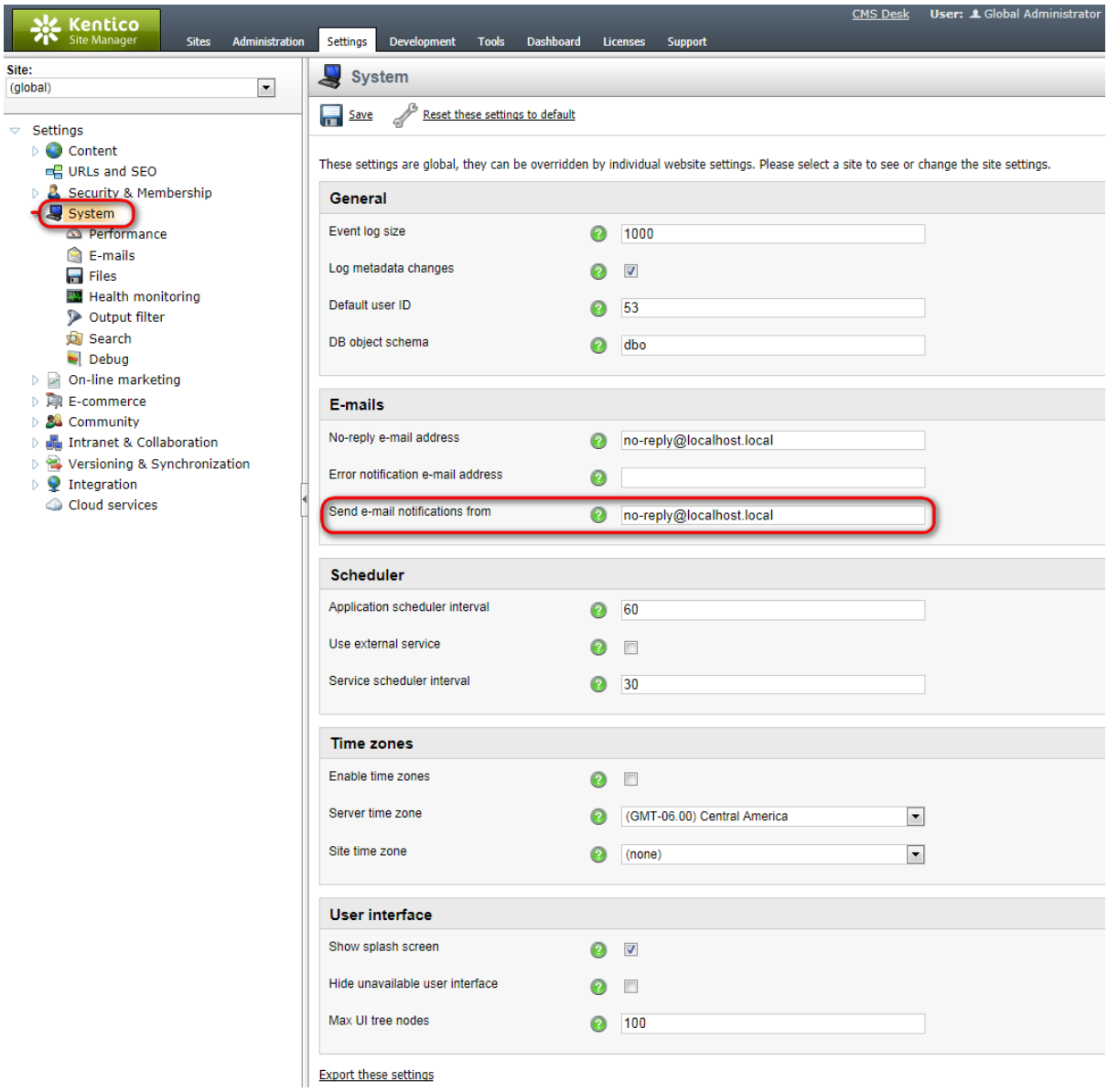
On the live site, the web part displays a check-box for each gateway when multiple gateways are selected. If you enable a particular check-box, the gateway's form is displayed, as you can see in the screenshot below.



8.35.6 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



8.35.7 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.

8.35.8 Notifications internals and API

8.35.8.1 Overview

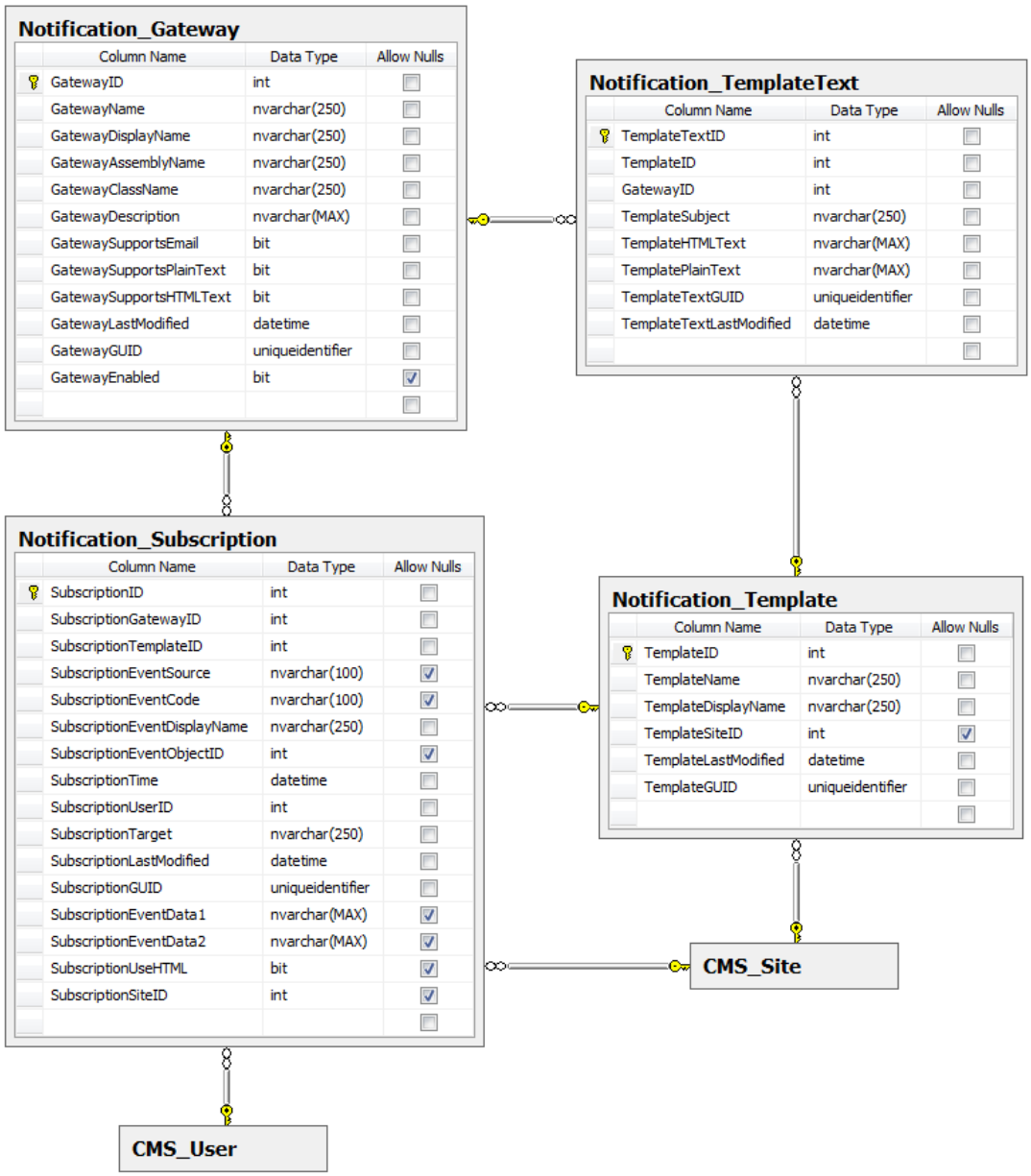
In this chapter, you will learn which database tables and API classes are used in the Notifications module. You will also see the most common API examples.

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.

8.35.8.2 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.



8.35.8.3 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.

8.35.8.4 API examples

8.35.8.4.1 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;
```

8.35.8.4.2 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;
}
```

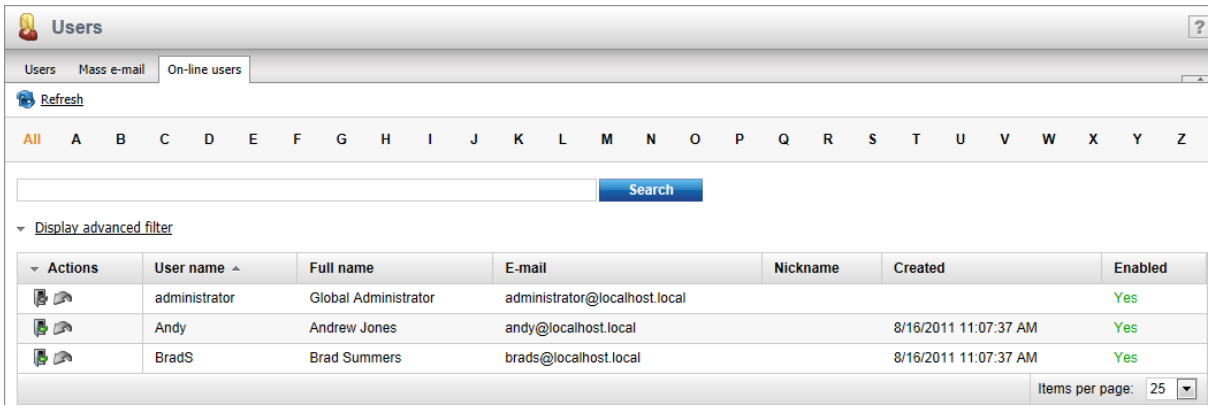
The following example deletes a notification template.

```
private bool DeleteNotificationTemplate()
{
    // Get the notification template
    NotificationTemplateInfo deleteTemplate = NotificationTemplateInfoProvider.
GetNotificationTemplateInfo("MyNewTemplate", CMSContext.CurrentSiteID);

    // Delete the notification template
    NotificationTemplateInfoProvider.DeleteNotificationTemplateInfo
(deleteTemplate);

    return (deleteTemplate != null);
}
```

8.36 On-line users

8.36.1 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.

8.36.2 Enabling the On-line users module

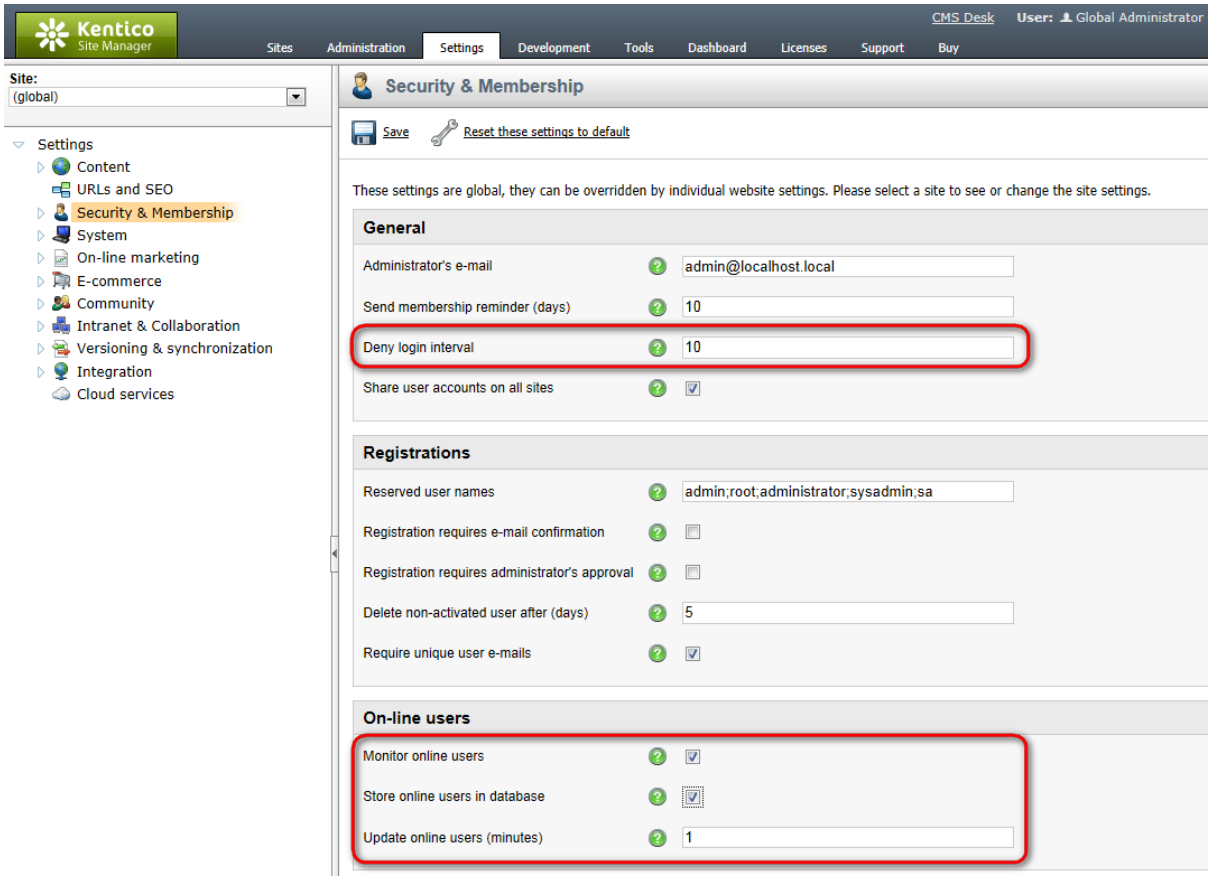
The On-line users module can be enabled in **Site Manager -> Settings -> Security & Membership** by checking the **Monitor on-line users** check-box.

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.

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**).



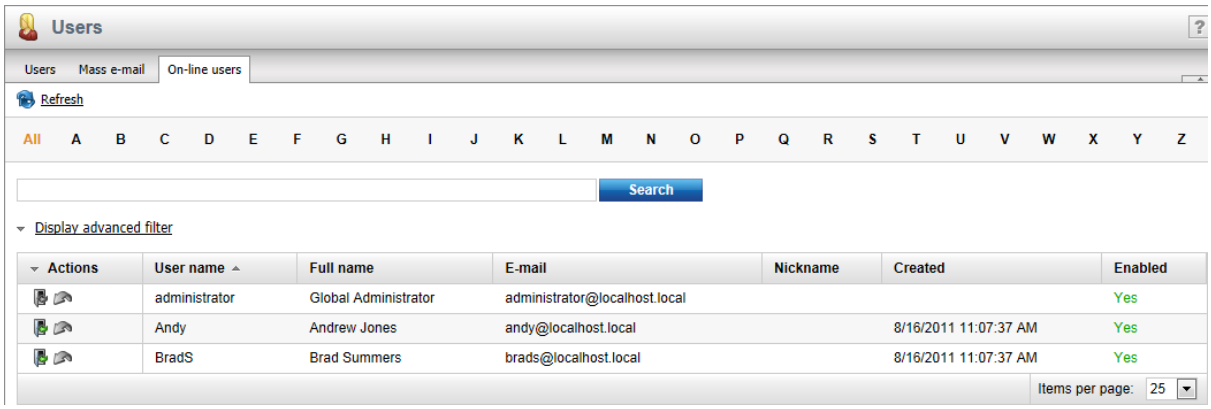
8.36.3 On-line users tab

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 some of the letters at the top of the page, you can view only those users whose user names begin with the clicked 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

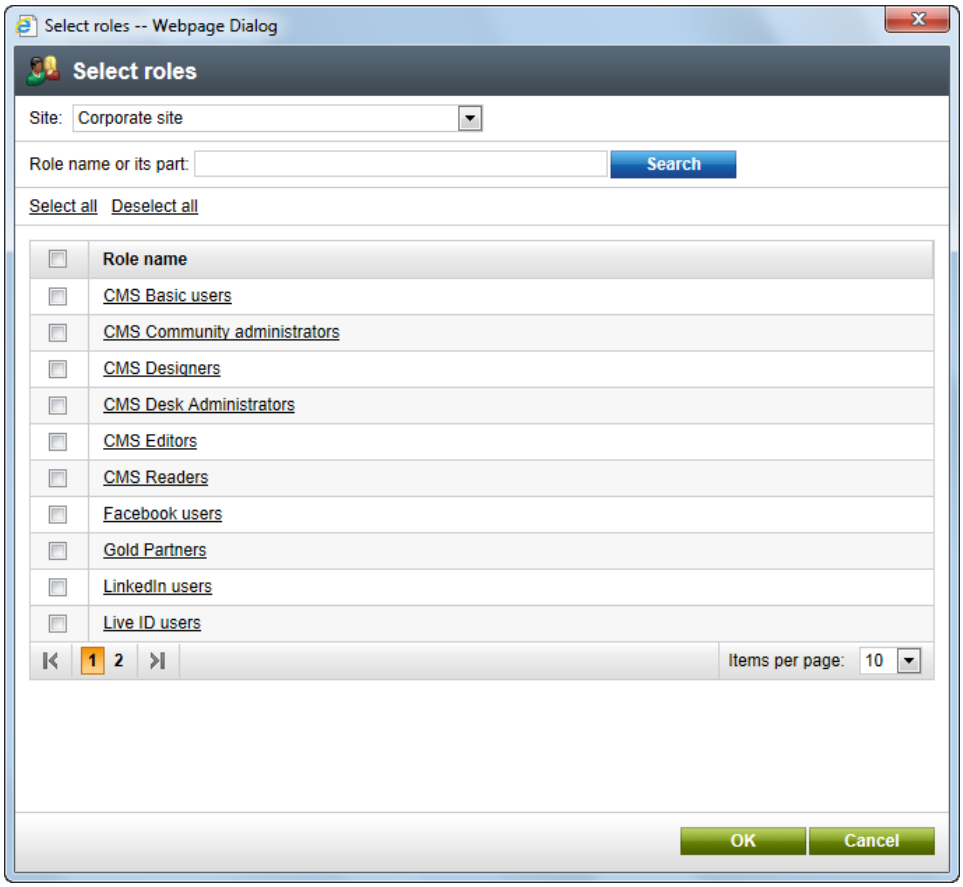
This option offers only one field. The expression entered into the field will be searched in users' **User Name**, **Full Name**, **E-mail Address** and **Nickname** properties.



Advanced filter

The advanced filter offers searching by **User Name**, **Full Name**, **E-mail Address** and **Nickname**.

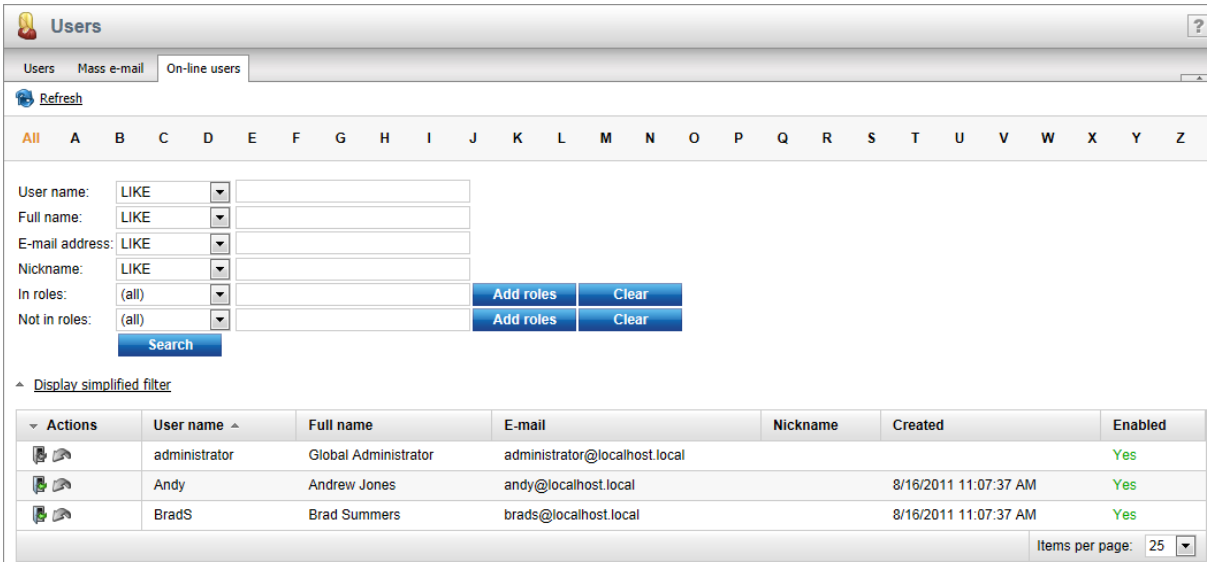
You can also filter on-line users by the roles they belong to. You can specify which roles the listed users are (**In roles**) or are not (**Not in roles**) members of by using the **Add roles** button to open the role selection dialog. Click **OK** to confirm the selection.



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.

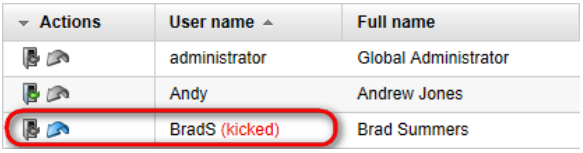
When you have entered all search criteria, click the **Search** button.

Only those users that match the specified criteria will be listed.



Kicking a user

If you click the **Kick** ( ) icon next to one of the listed users, you can kick the user off of the website, which means that the user will be logged from of the website and won't be able to log in for the duration specified in the **Site Manager -> Settings -> Security & Membership -> Deny login interval** setting. After doing so, there will be a (kicked) label written in red letters after the user's user name. If you want to take back the kick, you can do so easily by clicking the **Take back** ( ) icon.



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.

8.36.4 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.

8.36.5 On-line users internals and API

8.36.5.1 Overview

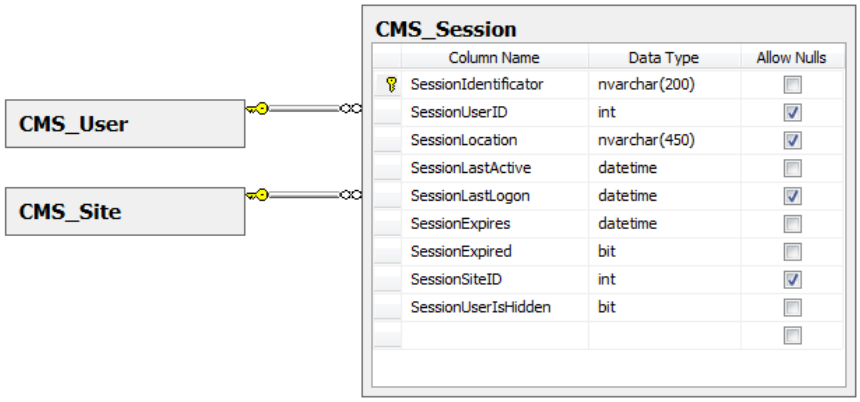
In this chapter, you will learn which database tables and API classes are used in the On-line users module. You will also see the most common API examples.

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.

8.36.5.2 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.



8.36.5.3 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.

8.36.5.4 API examples

8.36.5.4.1 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;
```

8.36.5.4.2 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(CMSContext.CurrentUser.UserID);
    SiteInfo site = SiteInfoProvider.GetSiteInfo(CMSContext.CurrentSiteName);

    if ((user != null) && (site != null))
    {
        // Check if user is on-line
        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;
}
```

8.37 Polls

8.37.1 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.

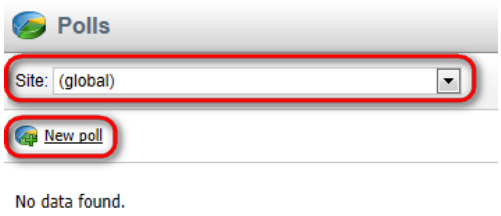
8.37.2 Managing polls

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 d oes 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.

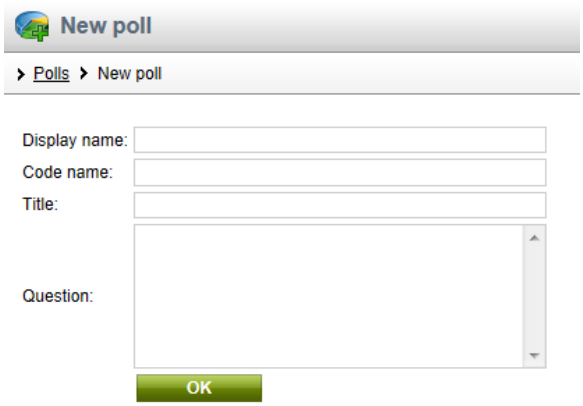
Please note

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 the **New poll** link 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.

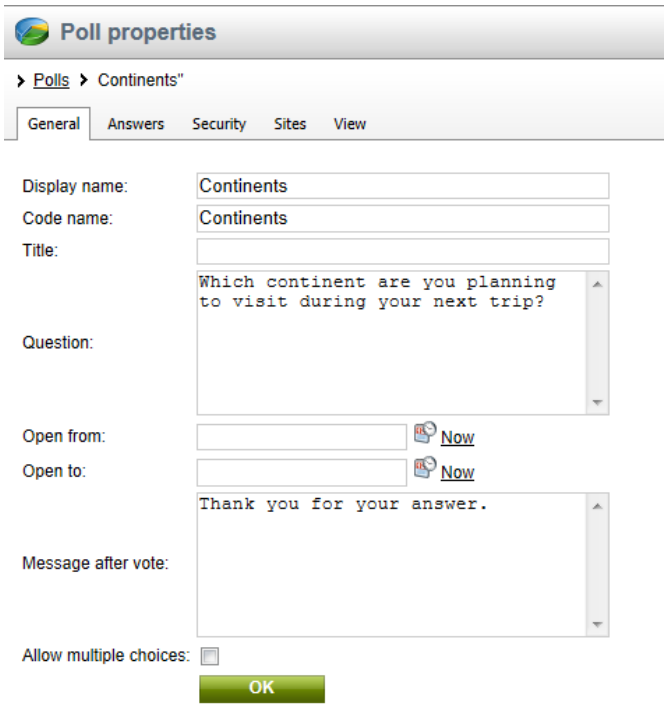


After entering the required values, click **OK** to save the changes you have made.

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.

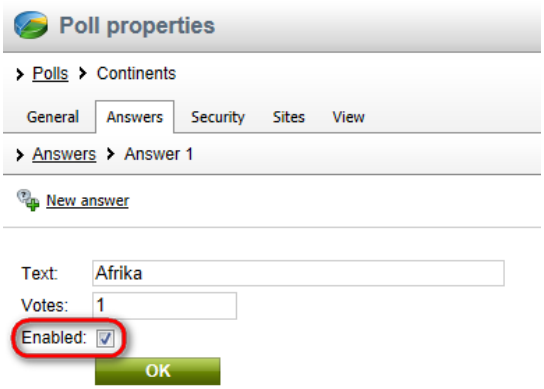


Click **OK** to save the changes you have made to the poll.

Defining answers

Then you can define a list of available answers on the **Answers** tab. To create a new answer, click the **New answer** link and enter the answer text. Your poll answers can be **Edited** (), **Deleted** (), **Moved Up** () and **Moved down** ().

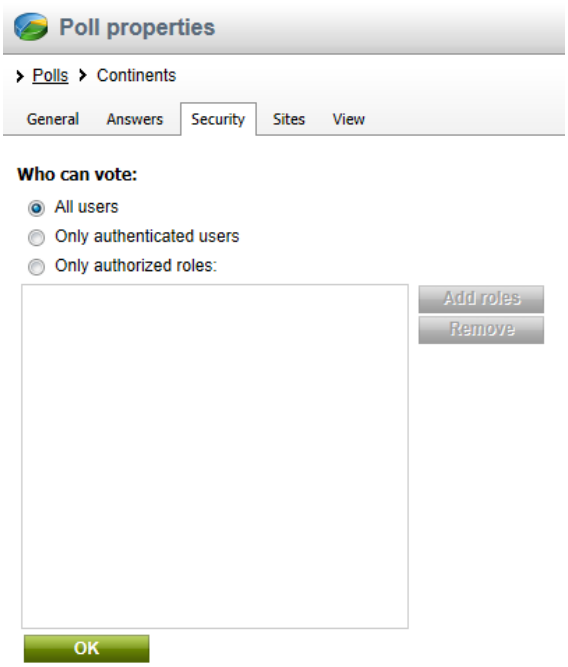
You can choose if the given answer should be enabled, i.e. displayed in the list of options. This is useful if you need to remove an answer from the poll while keeping the number of votes in the history. The disabled option is then not calculated into the displayed results.



You can also reset answers in your poll by clicking the **Reset answers** link.

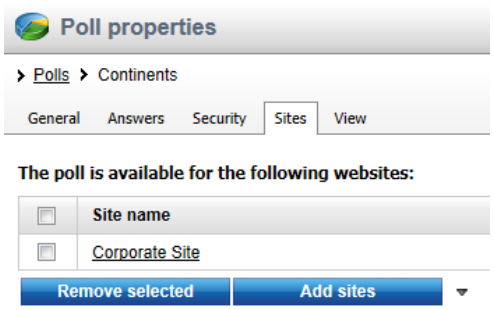
Defining who can vote

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).

8.37.3 Publishing polls

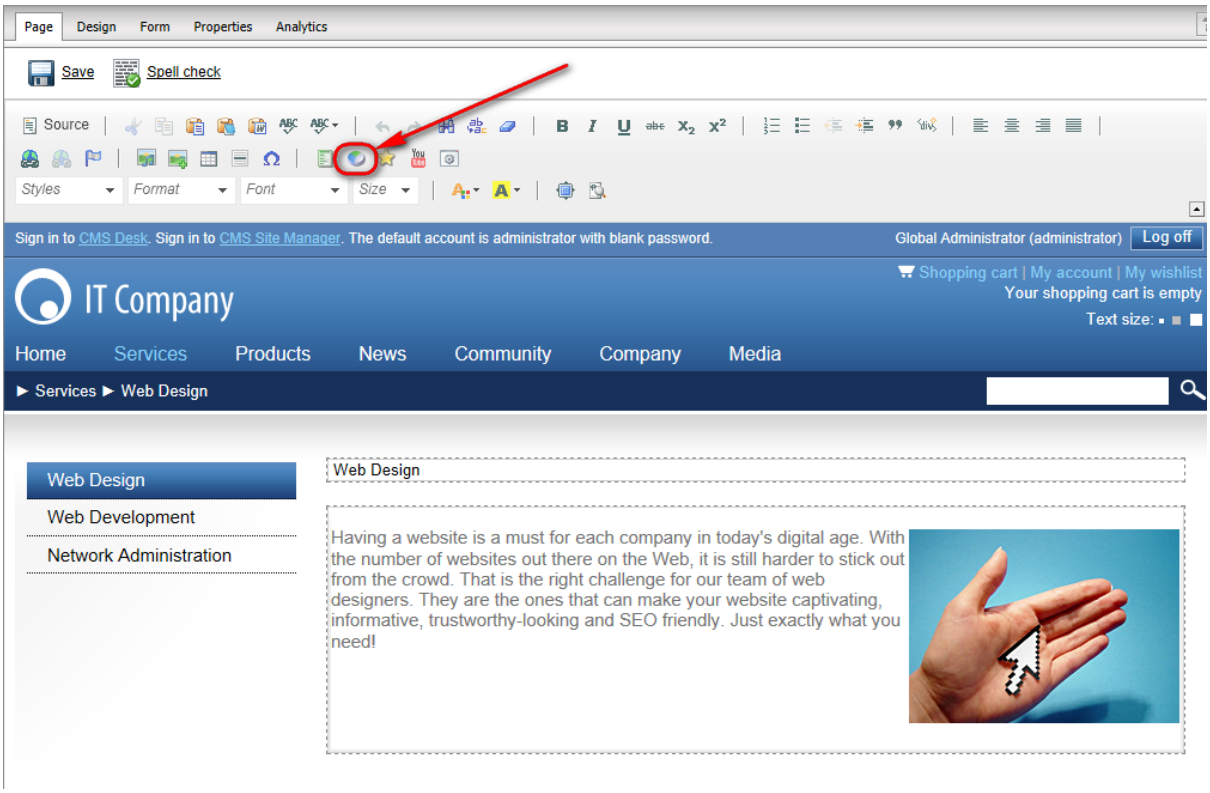
There are two ways how you can publish a poll on your website:

- 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.

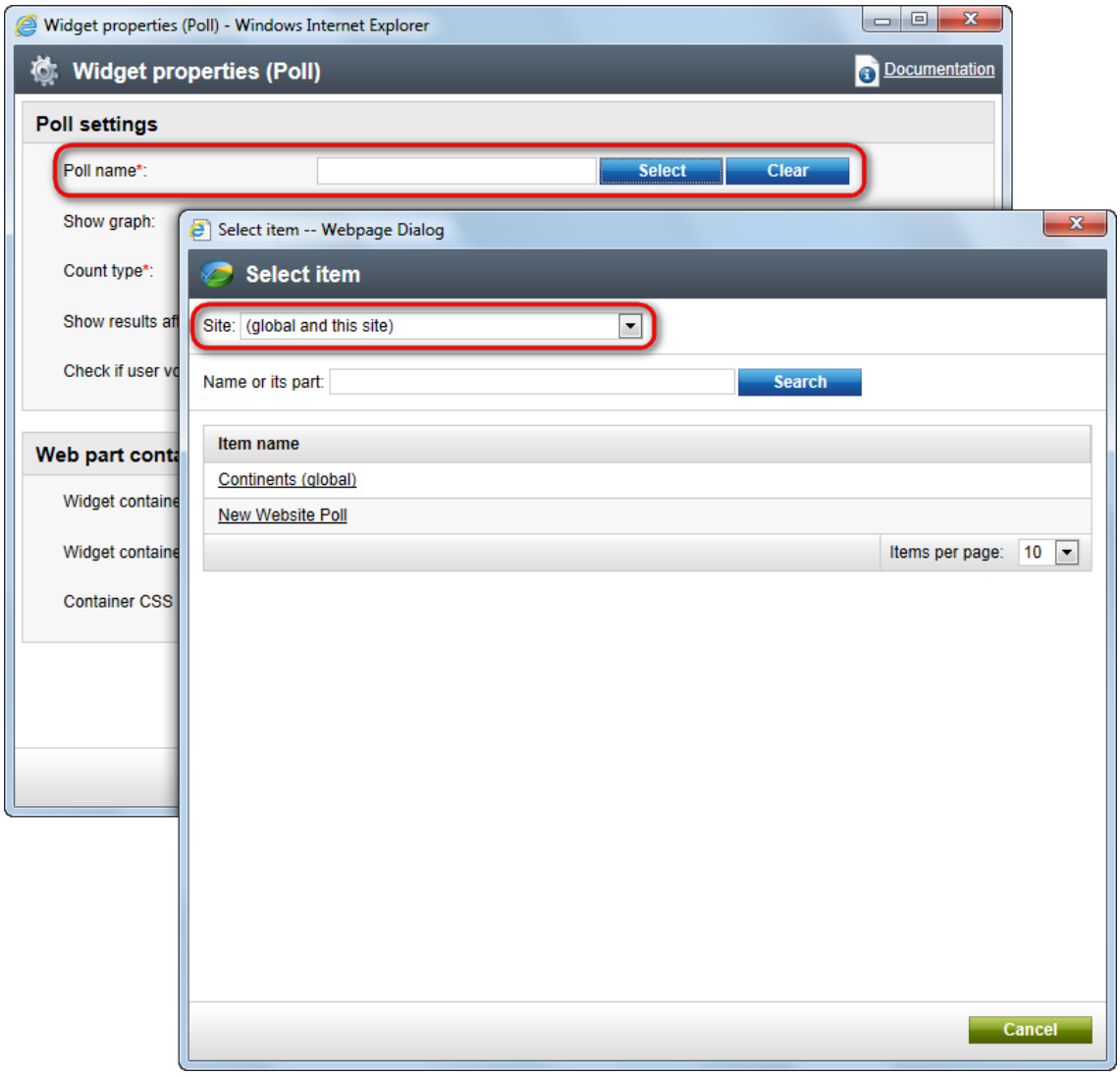
- Users with appropriate permissions can insert a poll into a widget zone of a page using the **Forms & Surveys -> Poll** widget.

Publishing polls for content editors

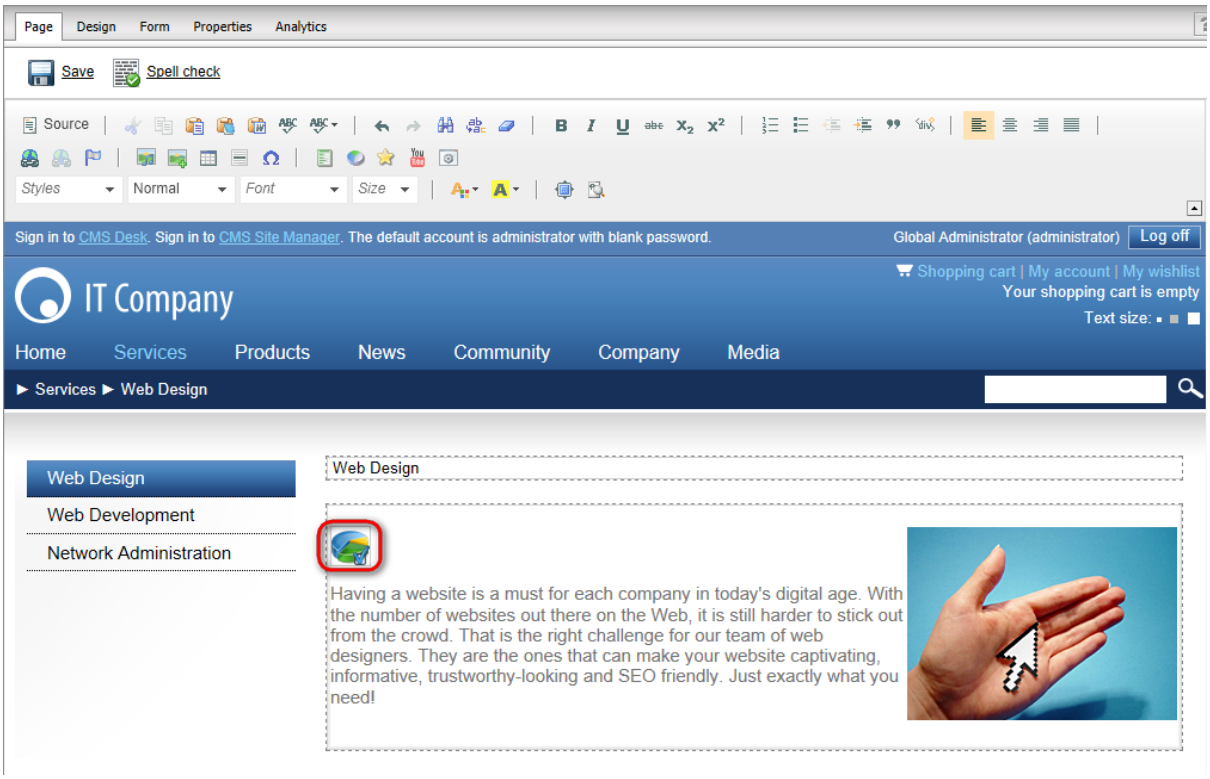
After defining your poll, you need to go to **CMS Desk -> Content** and navigate to the page where you want to have the poll displayed. Switch to the **Edit -> Page** tab and edit the page content using the built-in WYSIWYG editor. You will insert the poll by clicking the **Insert Poll** ( ) button on 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.

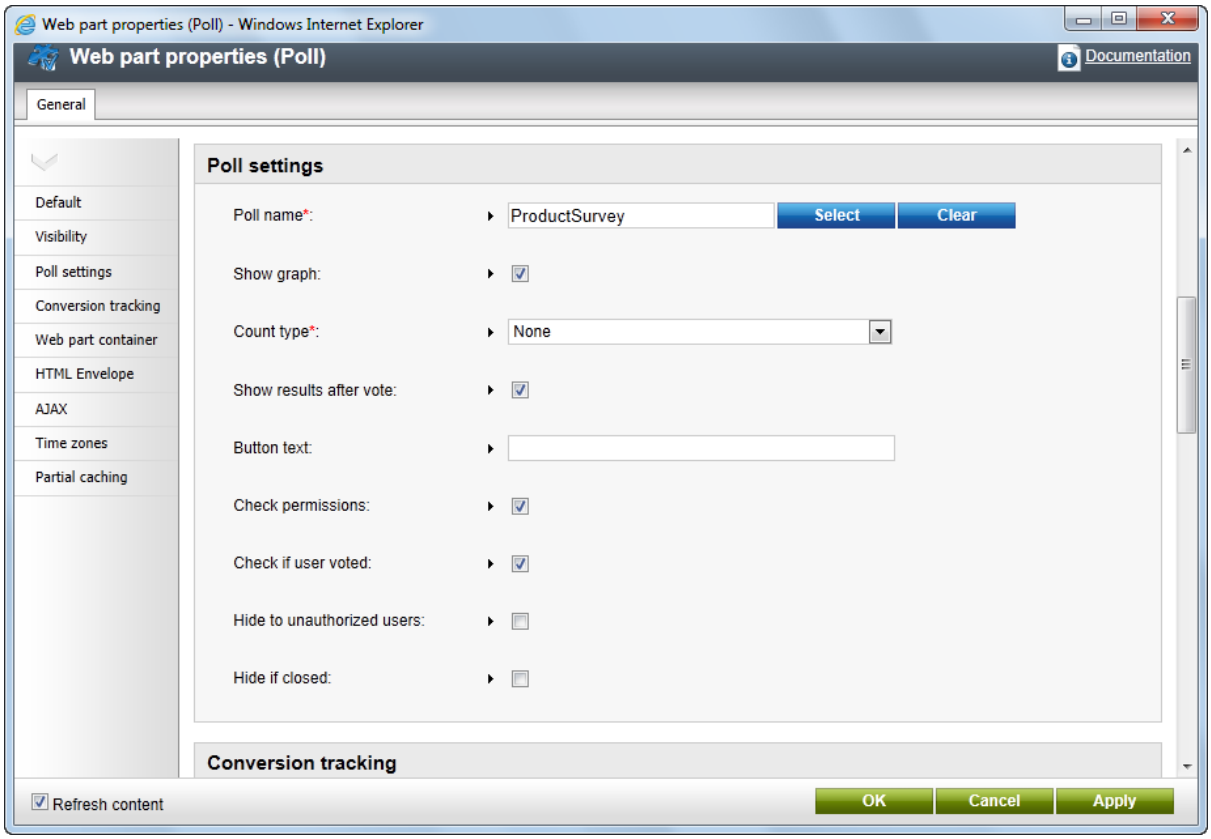


The placeholder image will be replaced by the actual poll on the live site.

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.

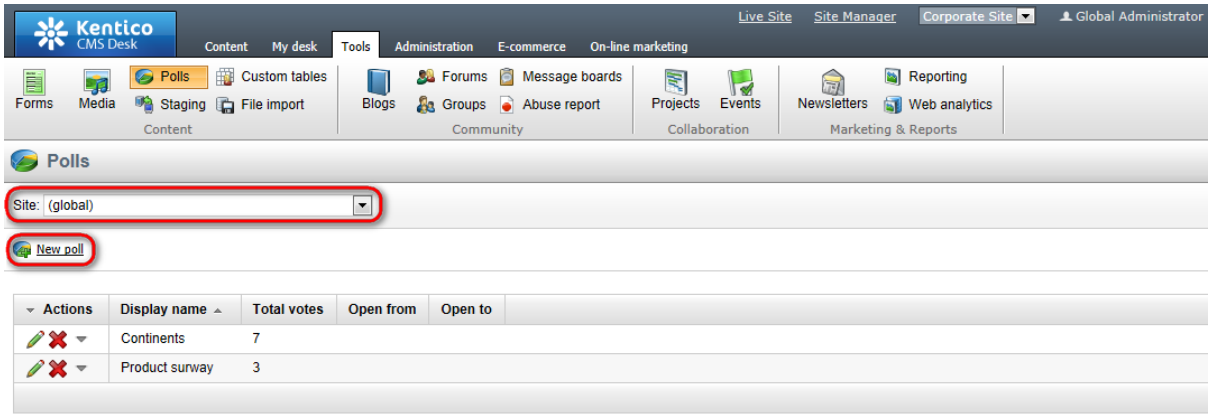
Tip: Global polls not showing up in the list

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.

8.37.4 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.

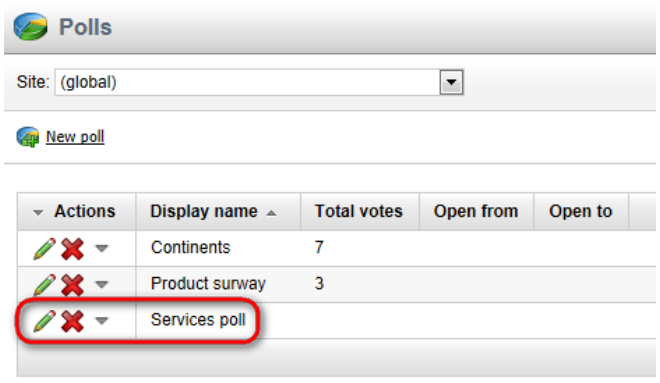
1. First go to **CMS Desk -> Tools -> Polls**, set the **Site** property to *(global)* and click the **New poll** link.



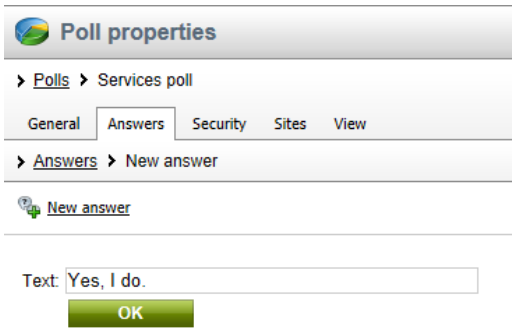
2. Enter the following information for the new poll.

- **Display name:** *Services poll*
- **Code name:** *ServicesPoll*
- **Title:** *Services poll*
- **Question:** *Do you like our services?*

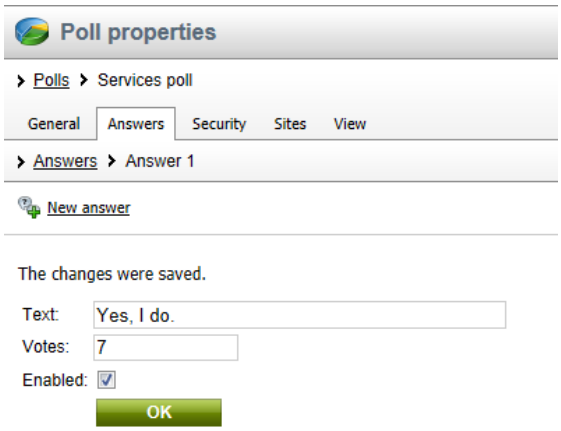
Click **OK** to save the changes. If you switch back to the **CMS Desk -> Tools -> Polls** tab, the poll you have just created will be displayed in the list.



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 the **New answer** link. Enter *Yes, I do.* into the **Text** text box and click **OK**.

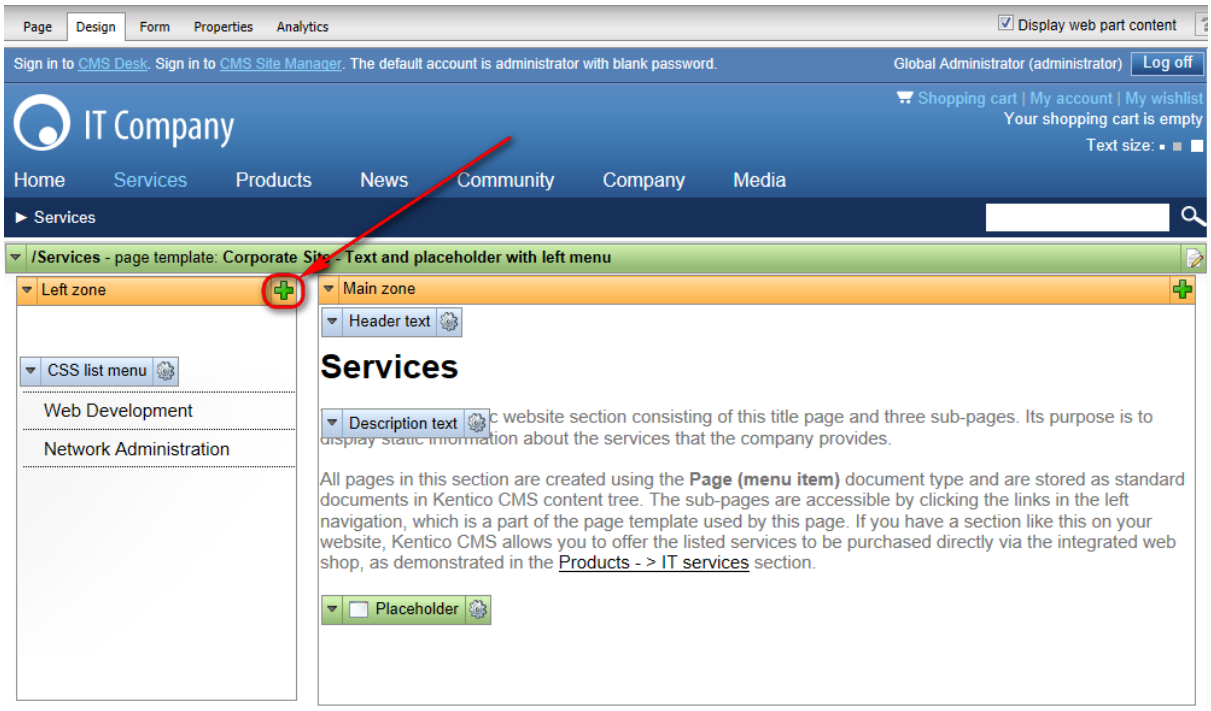


To make the answer available on the live site, check the **Enabled** check box. Besides, you can also enter the initial value for votes. For the purpose of this example, enter 7 and click **OK** to save the changes.

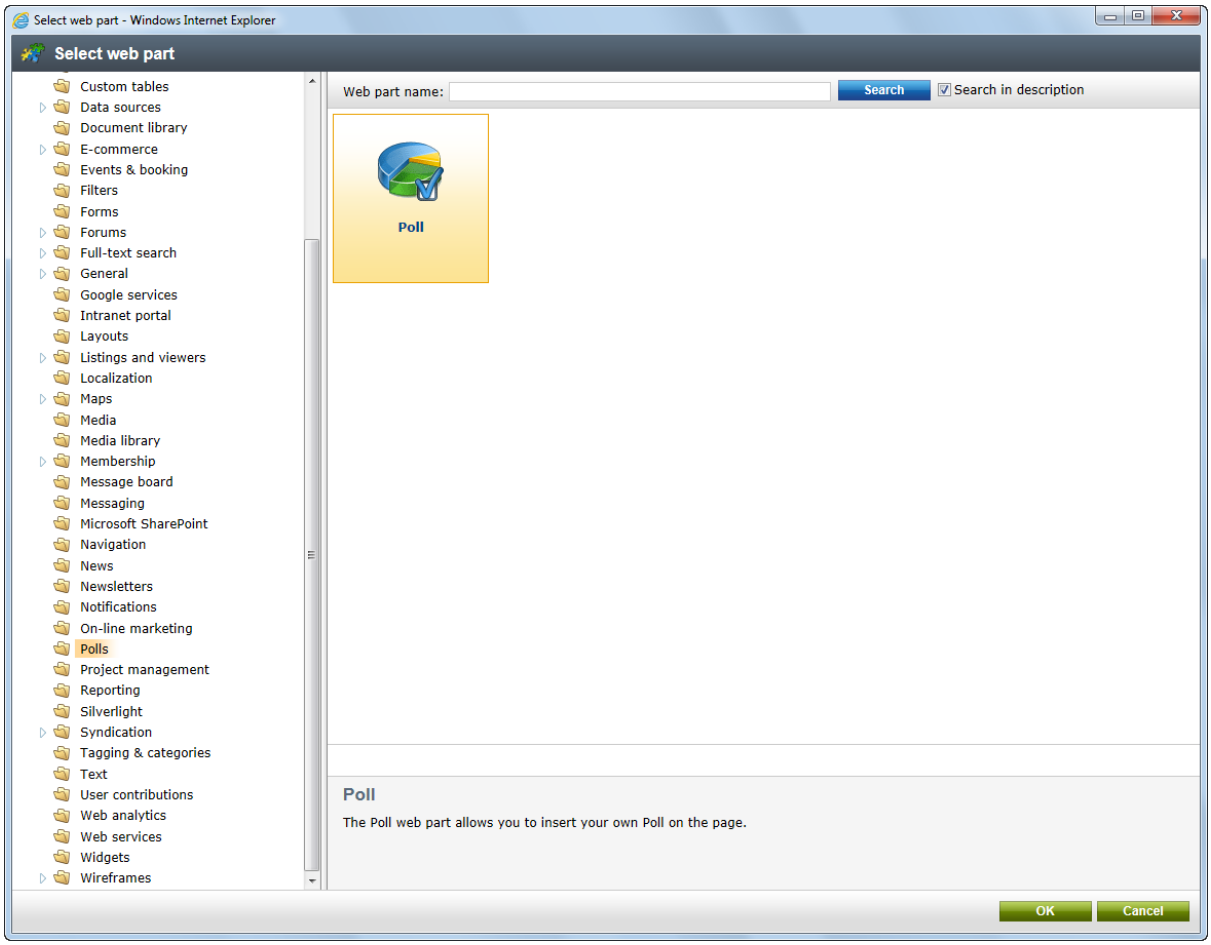


Repeat this step entering *No, I don't.* as an alternative answer and 3 as its initial value for votes. Optionally you can define more answers for your poll.

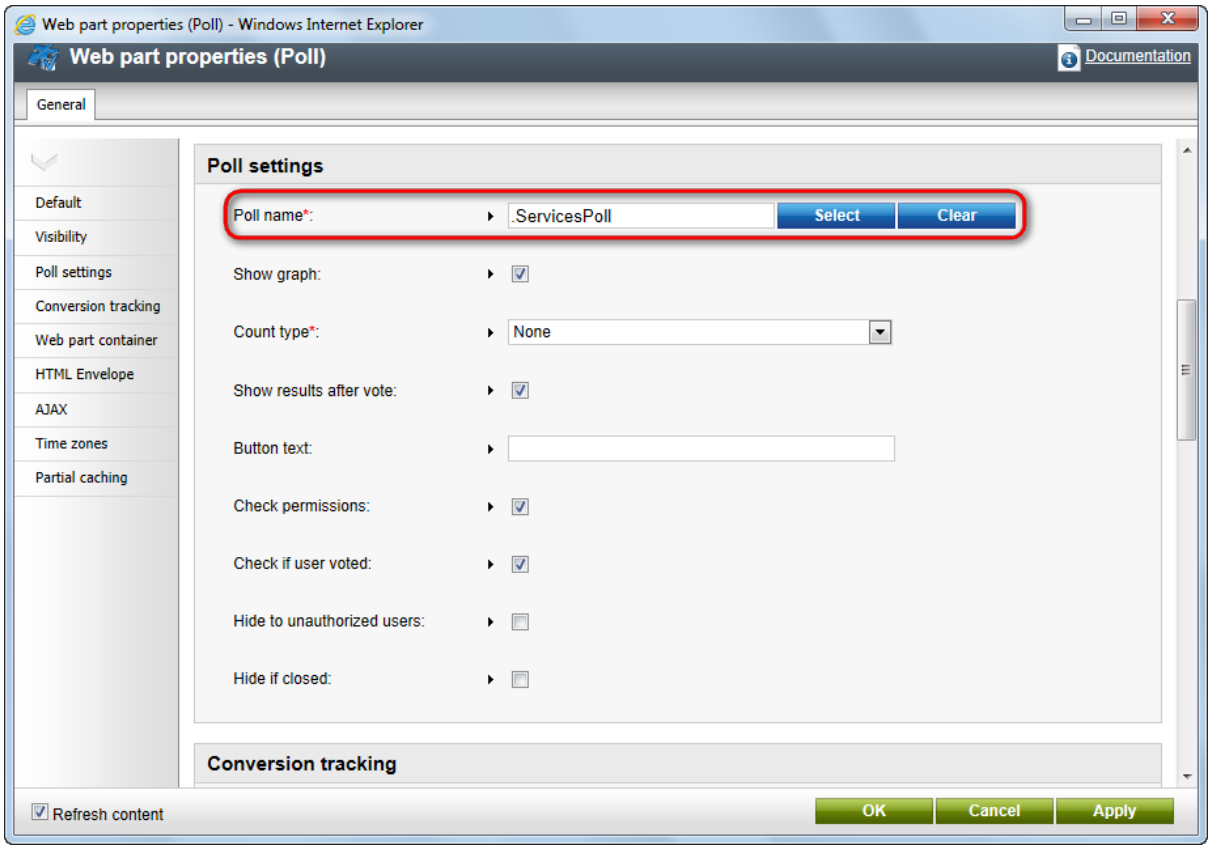
4. Now you are ready to publish your new poll on the website. Go to **CMS Desk -> Content**, navigate to the page where you want to add your poll and switch to the **Design** tab. Click **Add web part** () in one of the available web part zones



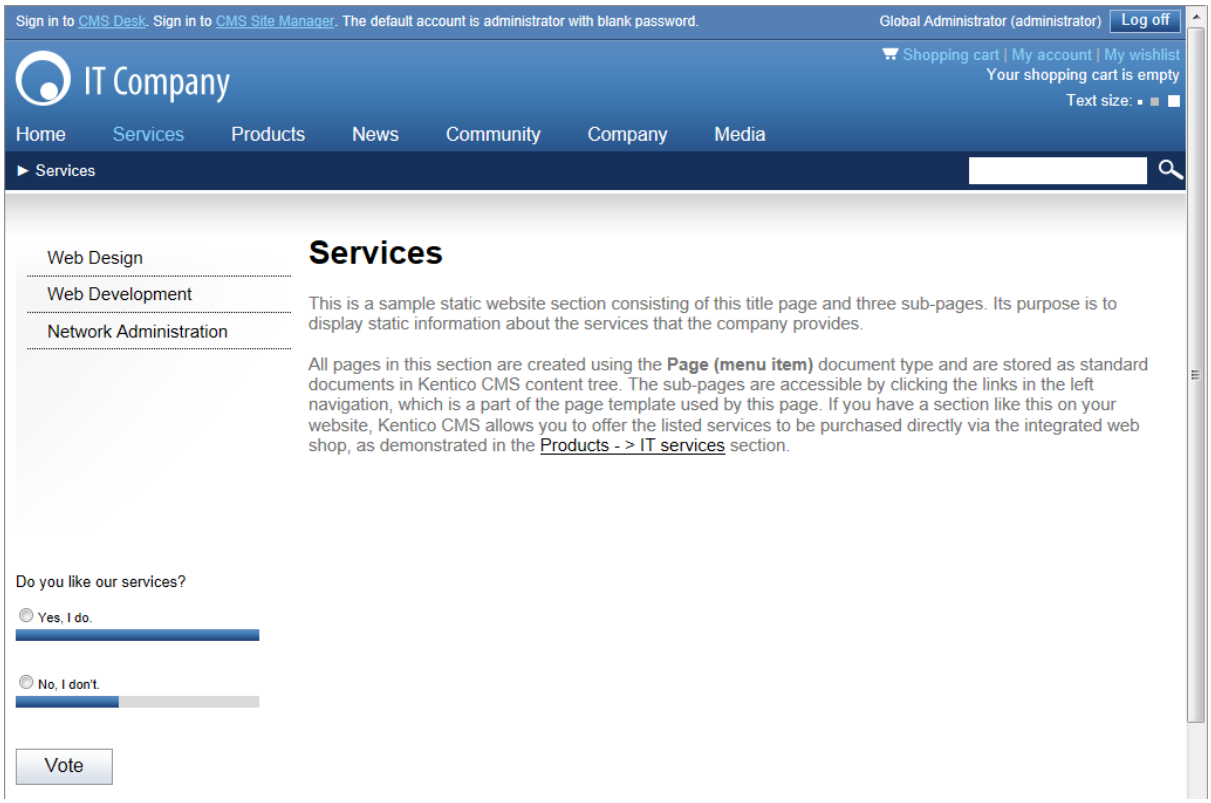
and in the **Select web part** dialog choose the **Polls -> Poll** web part.



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.



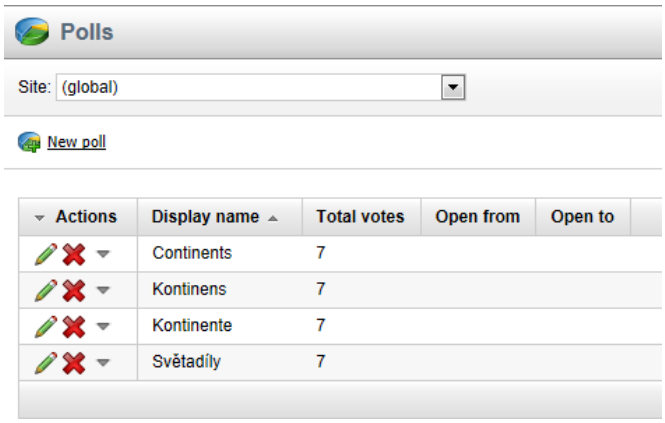
Please note

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.

8.37.5 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.



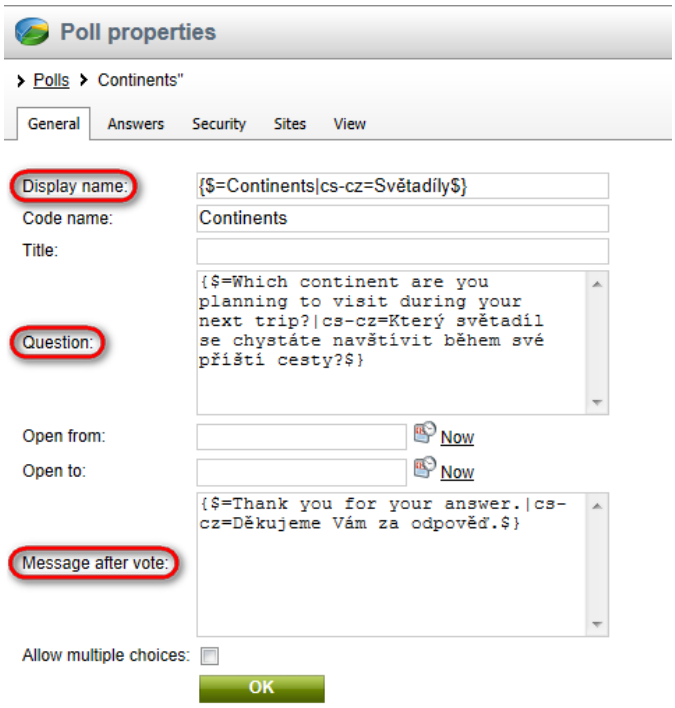
or

- 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.

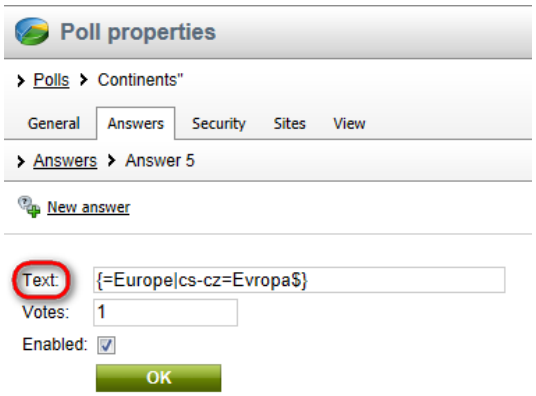
If you want to create a single localized poll (or adjust an existing one to create a single localized poll), you will need to take steps similar to those described in the Managing polls topic. The main difference consists in entering localization expressions into the corresponding text fields. For example, 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**



Click **OK** to 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 **OK** 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.

8.37.6 Security

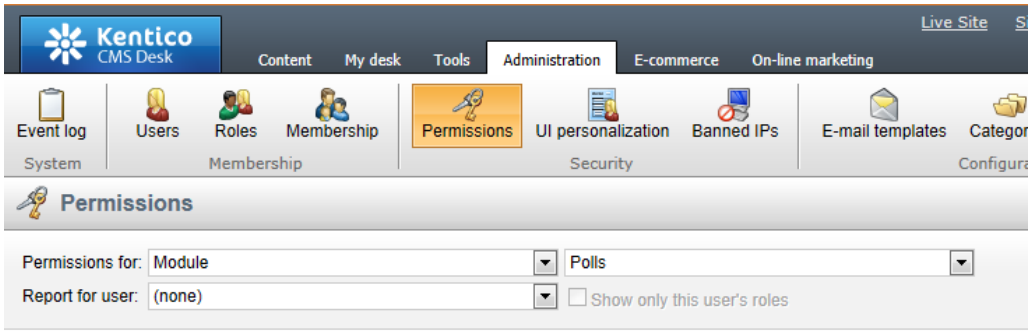
Here you will learn how to configure your CMS to enable users to access the Polls module, to allow global polls and you will also learn how to configure which users are authorized to vote in the poll.

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.



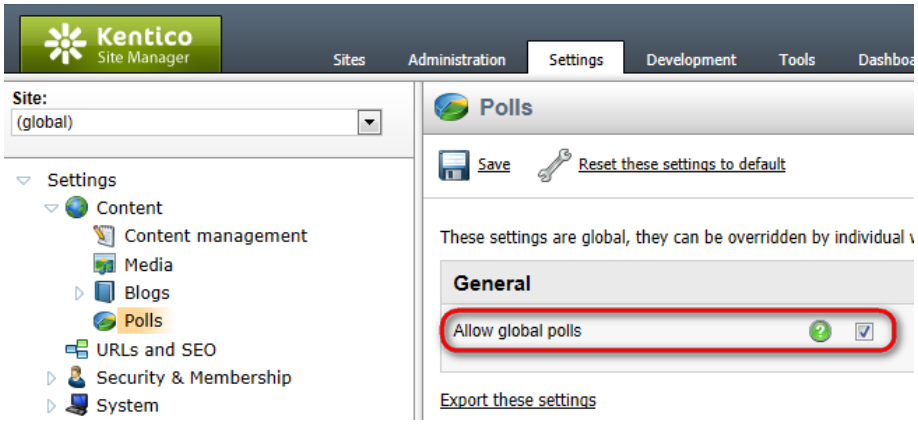
Role	Read global	Modify global	Read	Modify
Authenticated users	☐	☐	☐	☐
CMS Basic users	☐	☐	☐	☐
CMS Community administrators	☑	☑	☑	☑
CMS Designers	☐	☐	☐	☐
CMS Desk Administrators	☑	☑	☑	☑
CMS Editors	☑	☐	☑	☐
CMS Readers	☑	☐	☑	☐
Everyone	☐	☐	☐	☐
Facebook users	☐	☐	☐	☐
Gold Partners	☐	☐	☐	☐
LinkedIn users	☐	☐	☐	☐
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**.

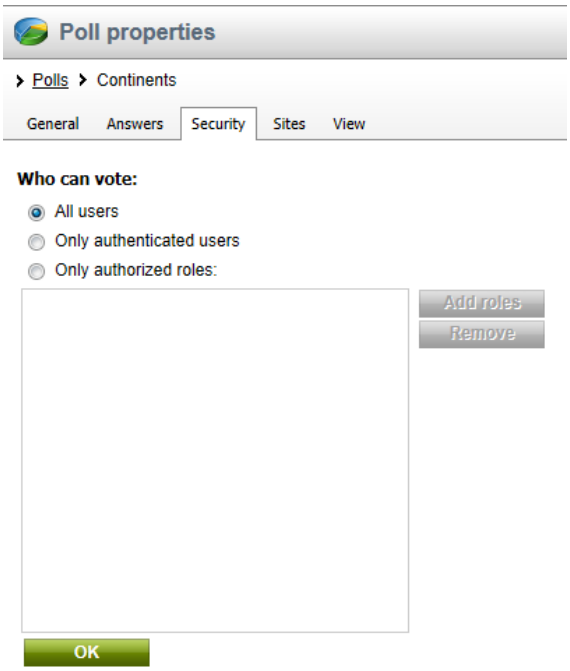


For detailed information on how to configure these settings, please refer to the Website settings topic.

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 a cookie.

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.

8.37.7 Polls internals and API

8.37.7.1 Overview

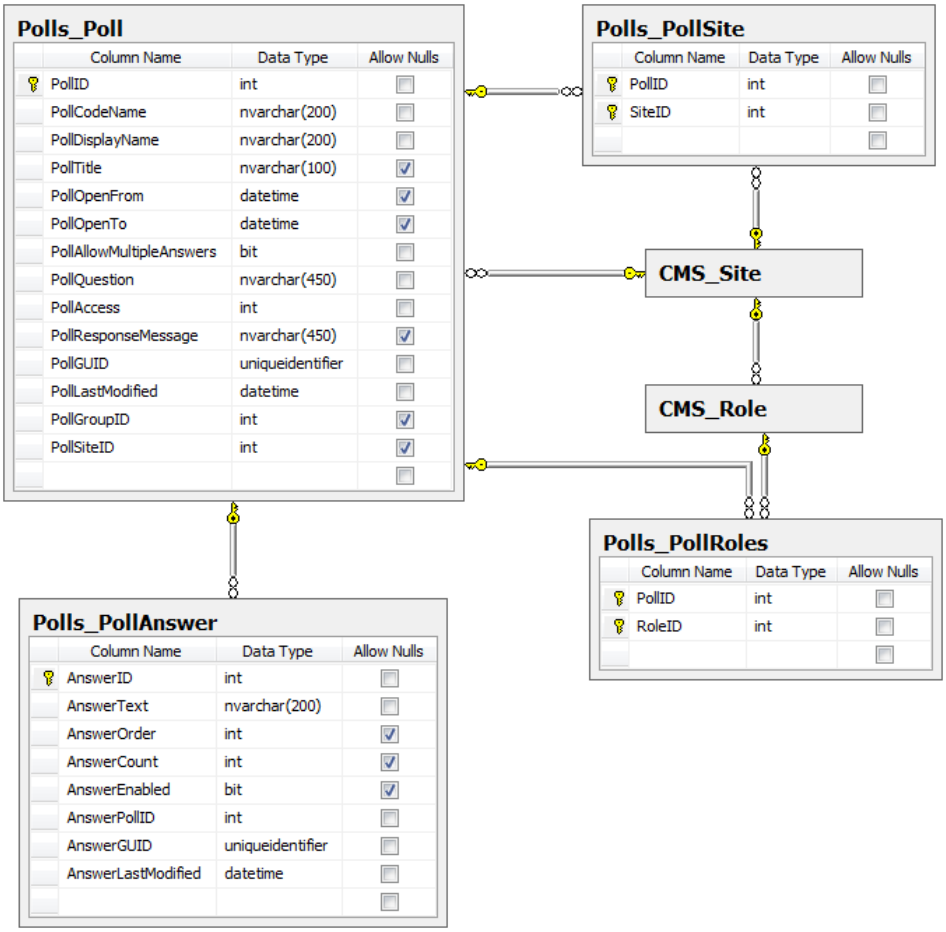
In this chapter, you will learn which database tables and API classes are used in the Polls module. You will also see the most common API examples.

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.

8.37.7.2 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).



8.37.7.3 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.

8.37.7.4 API examples

8.37.7.4.1 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;
```

8.37.7.4.2 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);
}
```

8.37.7.4.3 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;
}
```

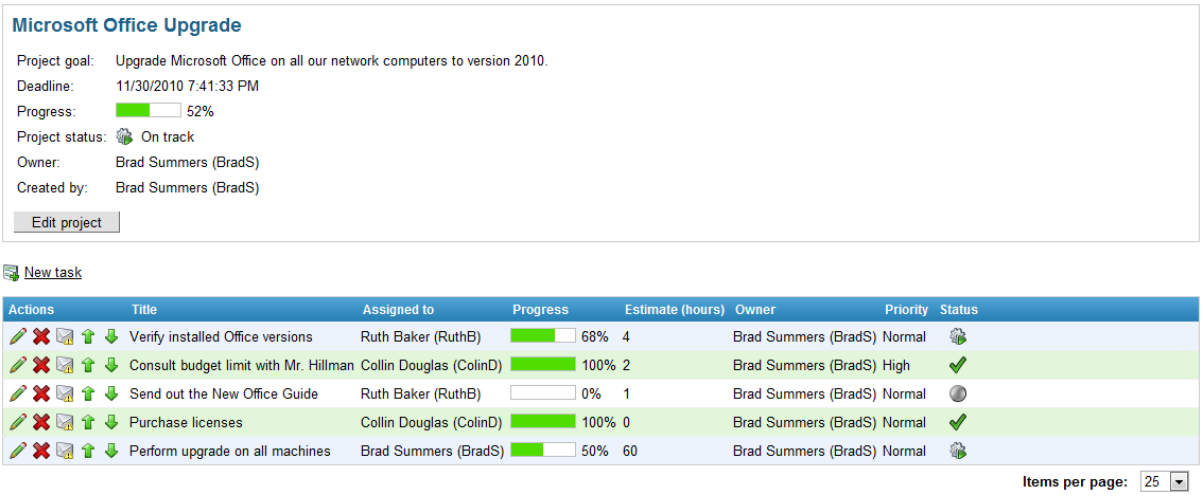
The following example deletes an answer.

```
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;
}
```

8.38 Project management

8.38.1 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.

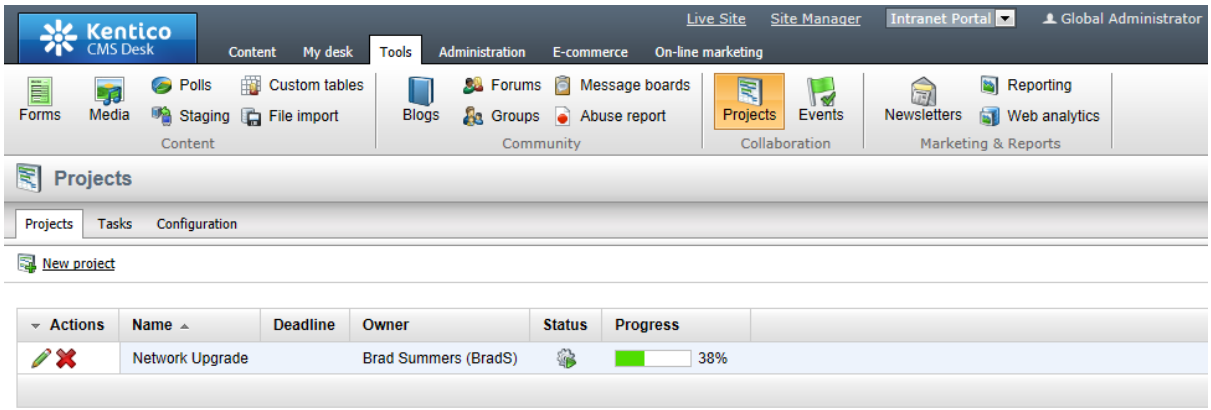
Permissions for project/task management are determined by many factors. Please refer to the Security topic for more information.

8.38.2 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**.

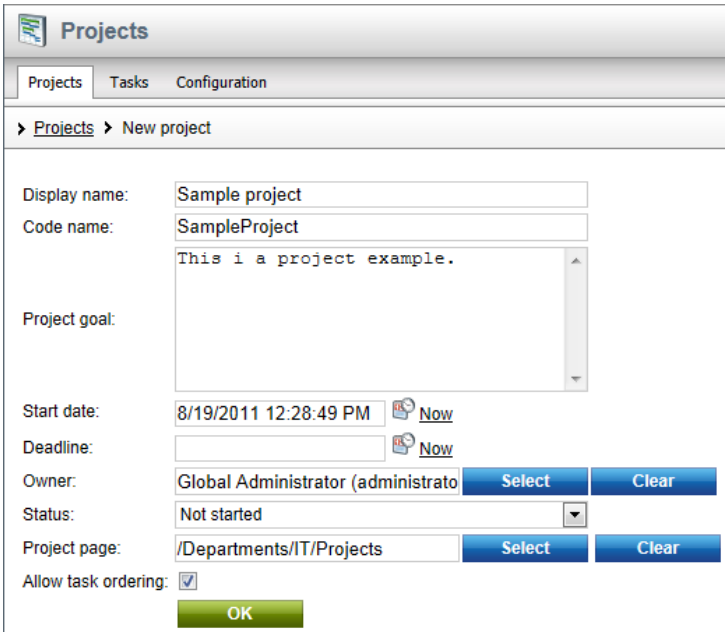


Here, existing projects are displayed in a list where they may be **Edited** ( ) or **Deleted** ( ).

Create a new project using the **New project** link 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

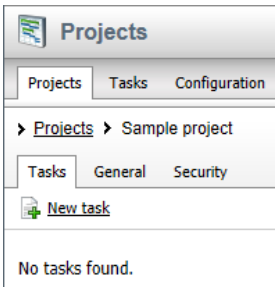
buttons.



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.

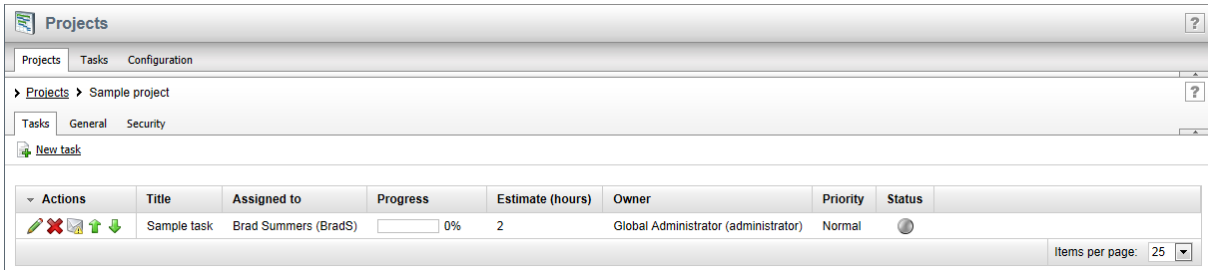


Click the **New task** link and enter the following properties:

- **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.

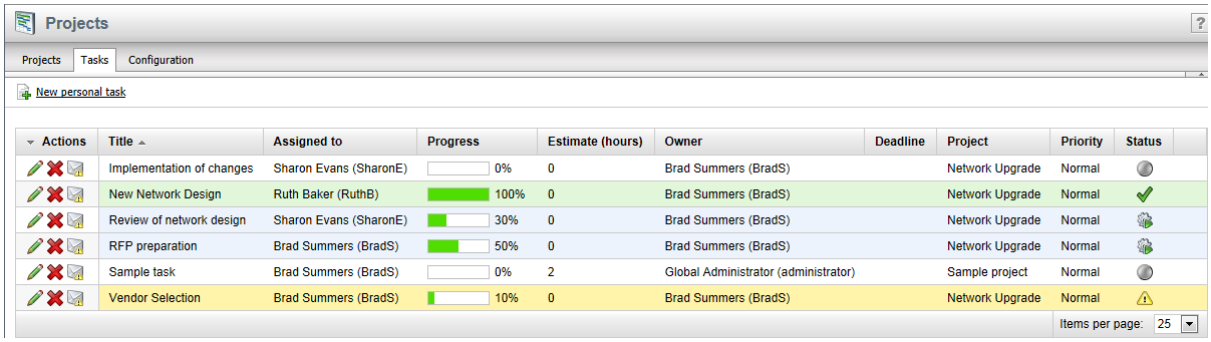
- **Priority** - allows one of the predefined task priorities to be selected.
- **Is private** - if enabled, the task will only be displayed to its owner, the user to whom it is assigned and to users who have permissions to manage the project that the task belongs under (if applicable).
- **Assigned 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**.

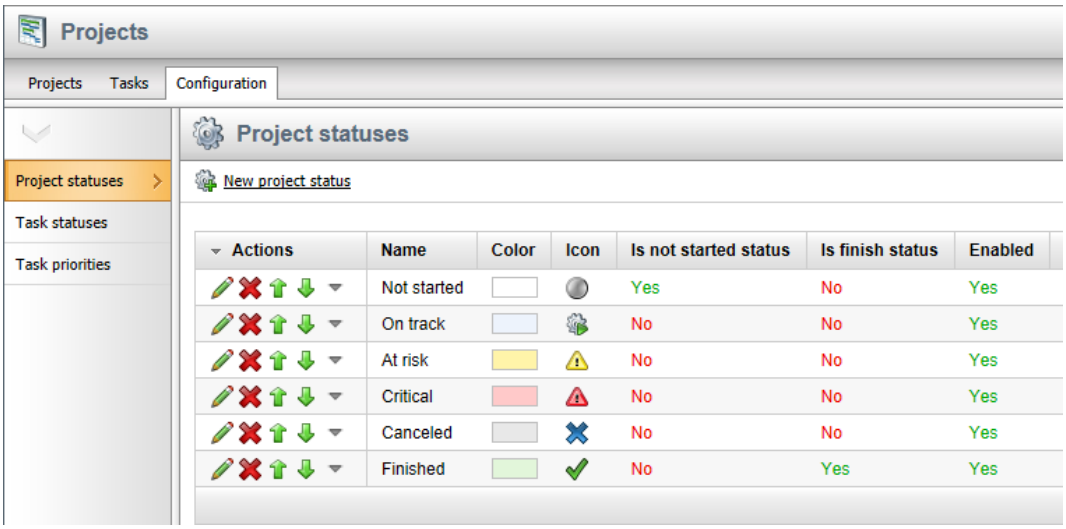


You can manage existing tasks using the appropriate action buttons on the left side: **Edit** () or **Delete** (). The **Send reminder** () button allows an e-mail to be sent to the user to whom the task is assigned, reminding them about the task. If the **Allow task ordering** property of the edited 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.

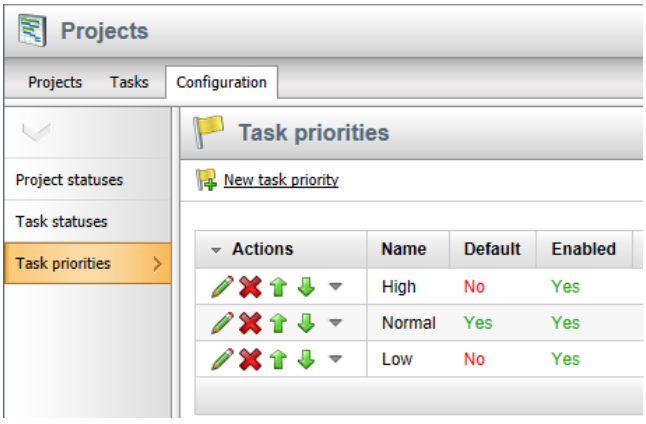


When creating or editing a **status**, the following properties can be set:

- **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.

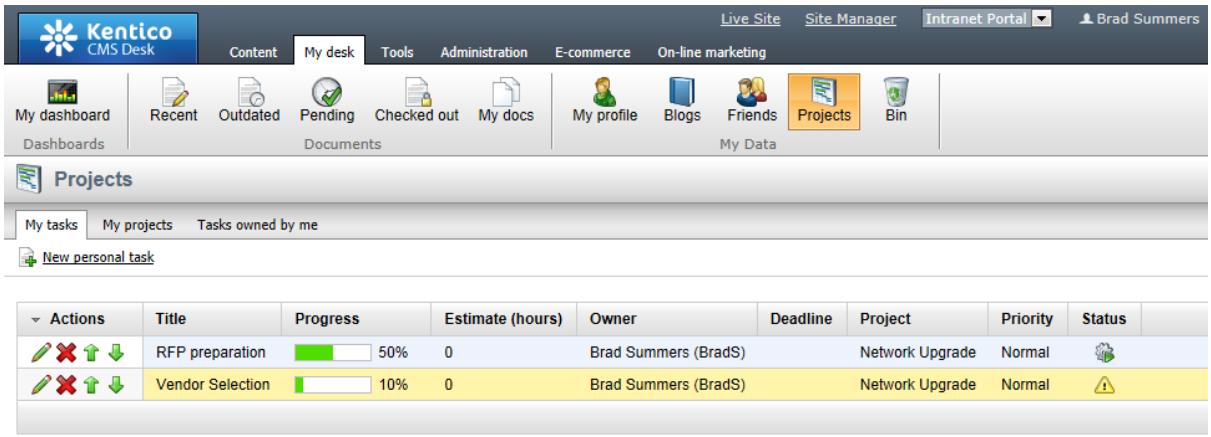
Using the Group management interface

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.

8.38.3 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.

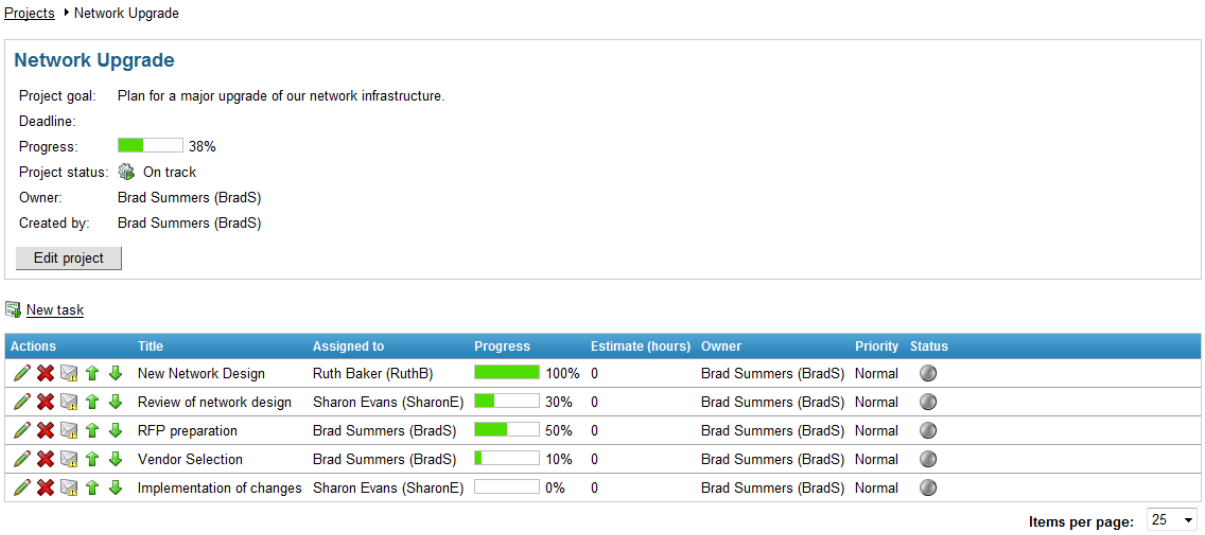
Project list

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.

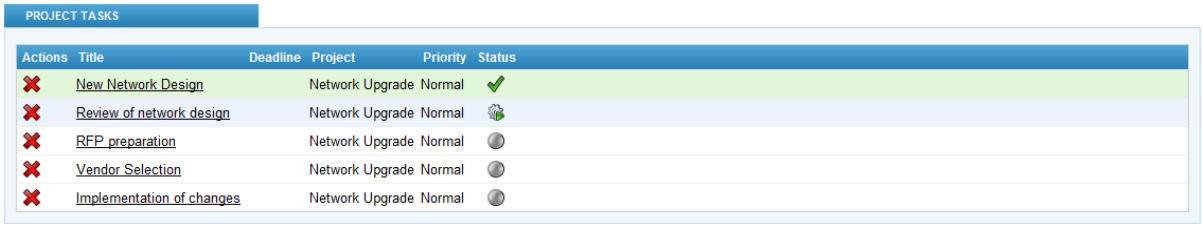
Project tasks

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.

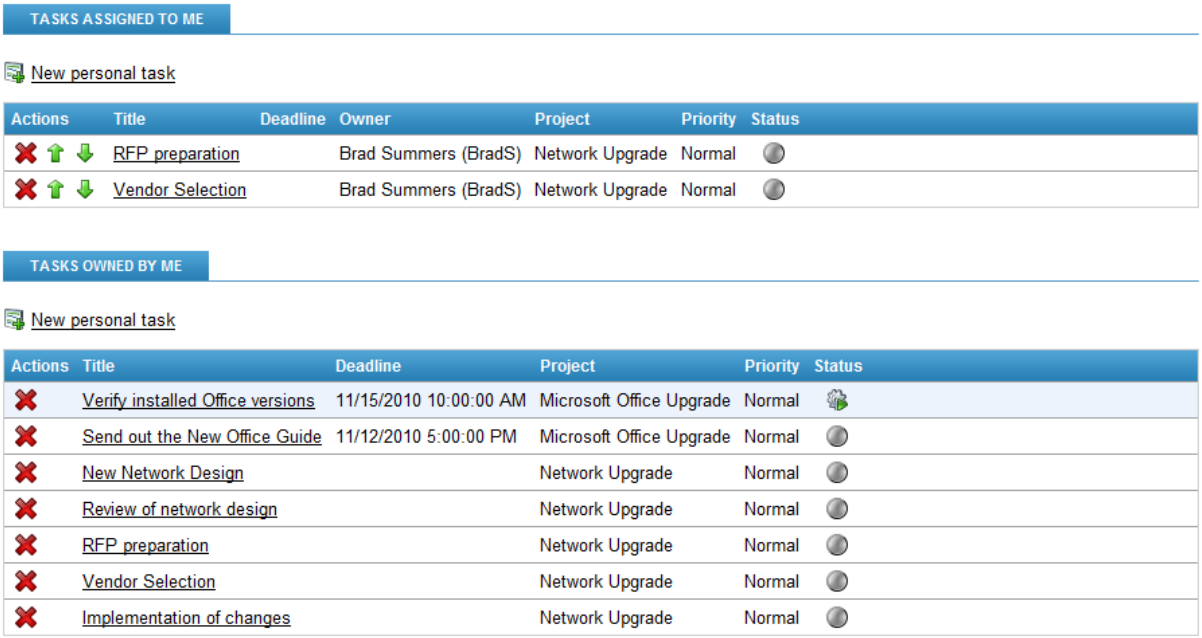


Tasks owned by me / Tasks assigned to me

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:

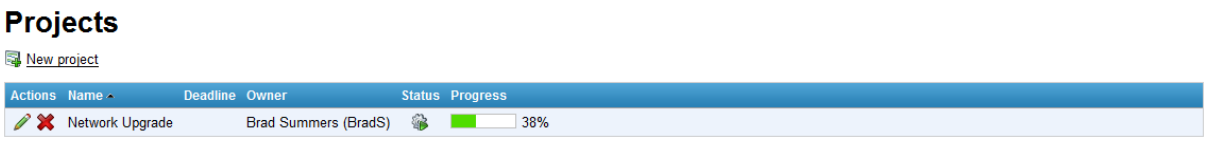
- **Task detail page URL** - contains the URL of the document that is linked when the displayed message is clicked. If left empty, the value is taken from the *Site Manager -> Settings -> Intranet & Collaboration -> Projects -> Task detail page* field. In the case that neither of these locations contain 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.

8.38.4 Using project management on the live site

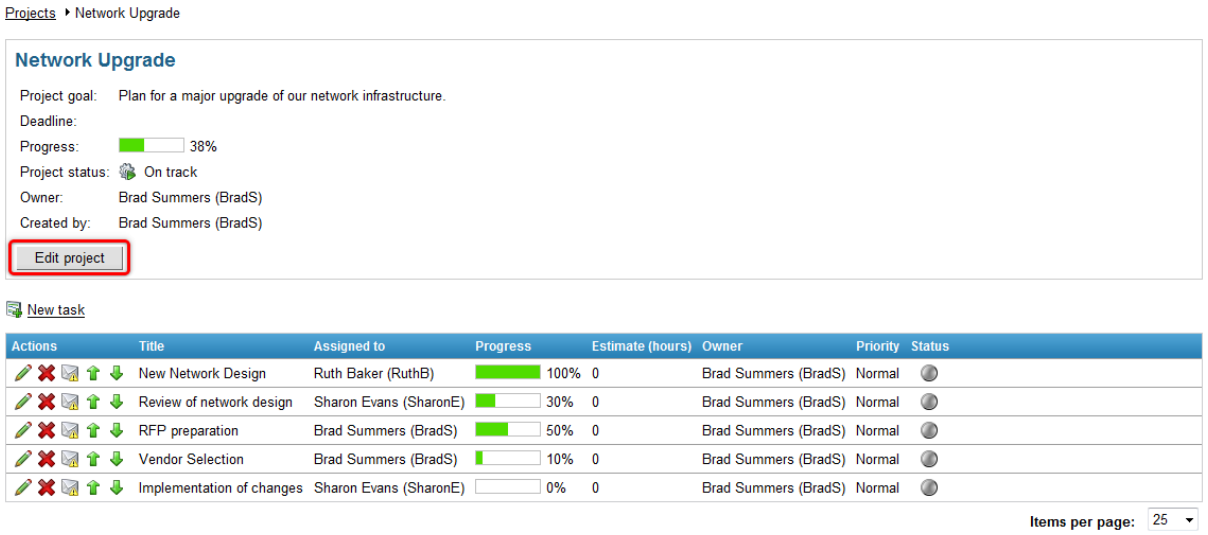
Users can work with projects and tasks on the live site through Project management web parts and widgets. The actions that are available to users depend on project security settings and the configuration of specific web parts or widgets.

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

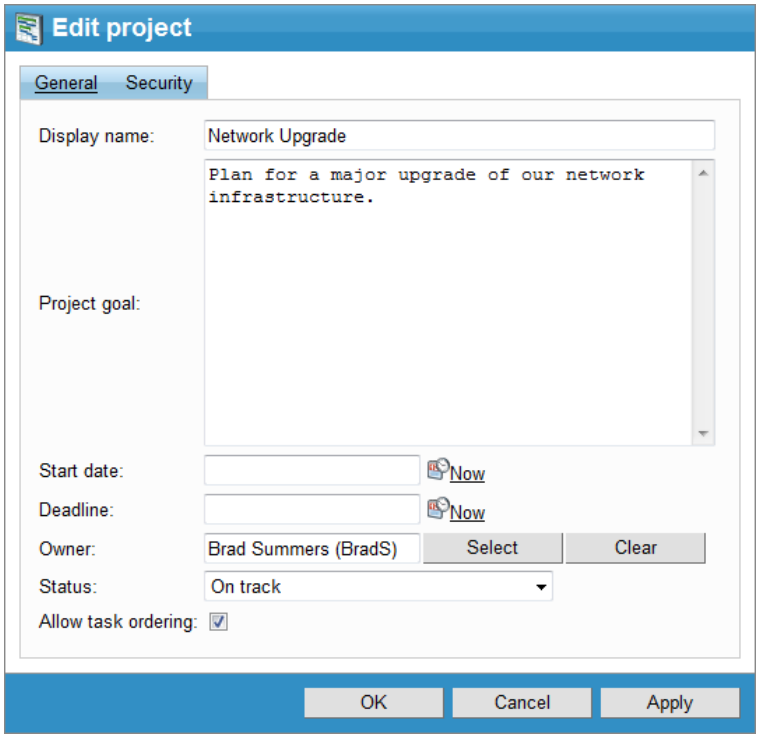
tasks topic.



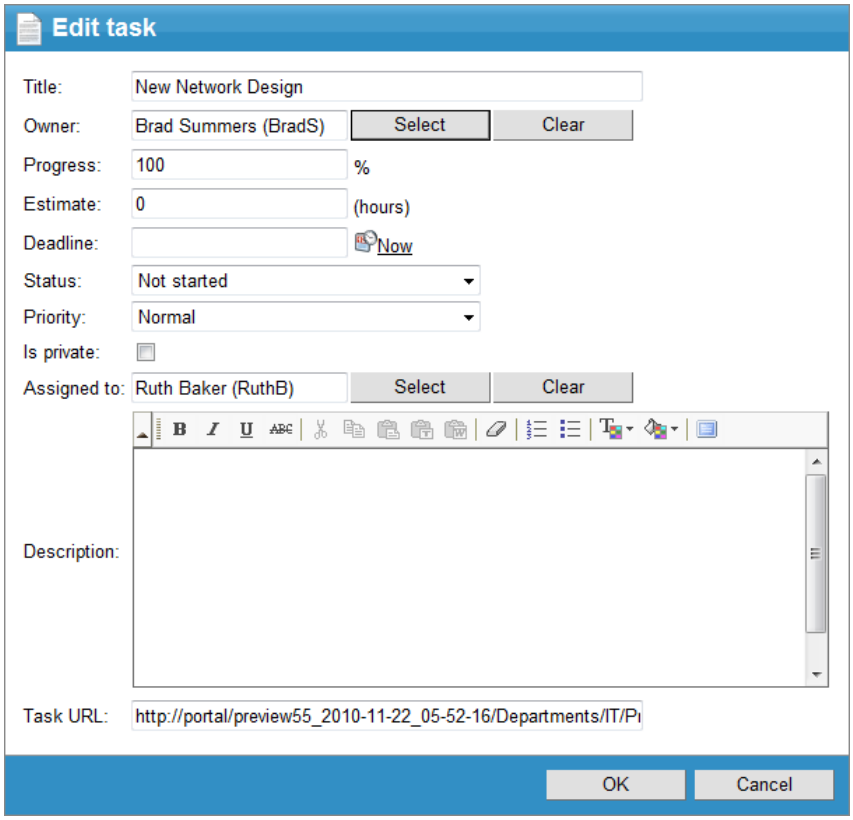
When a project is edited ( ), its details and the tasks it contains are displayed. Configuration of the project itself can be achieved by clicking the **Edit project** button.



This causes the following dialog to appear:



Editing ( ) a task on the live site is done through the dialog depicted below:

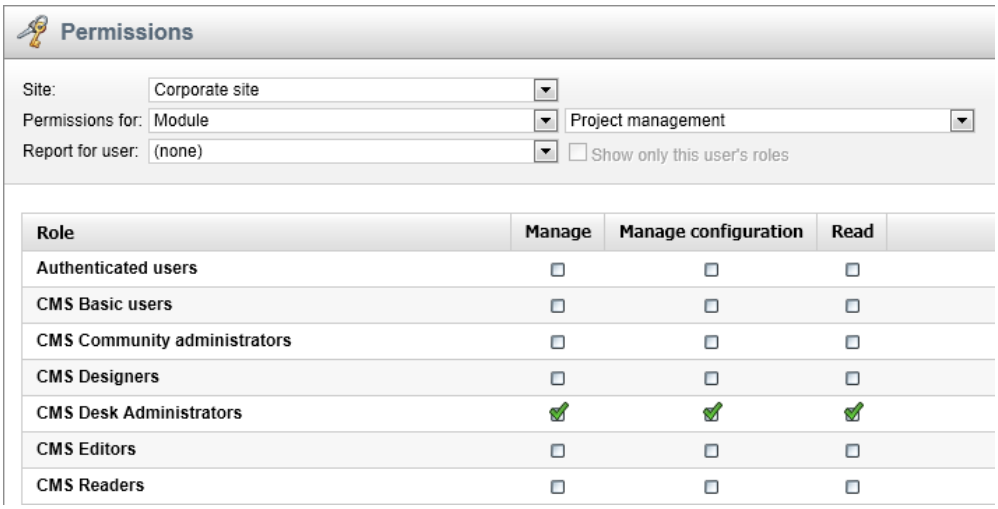


In addition to the task properties described in the Managing projects and tasks topic, the **Task URL** is shown, which can be used to link to the edited task.

Certain other web parts and widgets, such as **Tasks assigned to me**, may be used to edit displayed tasks in the same way.

8.38.5 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.



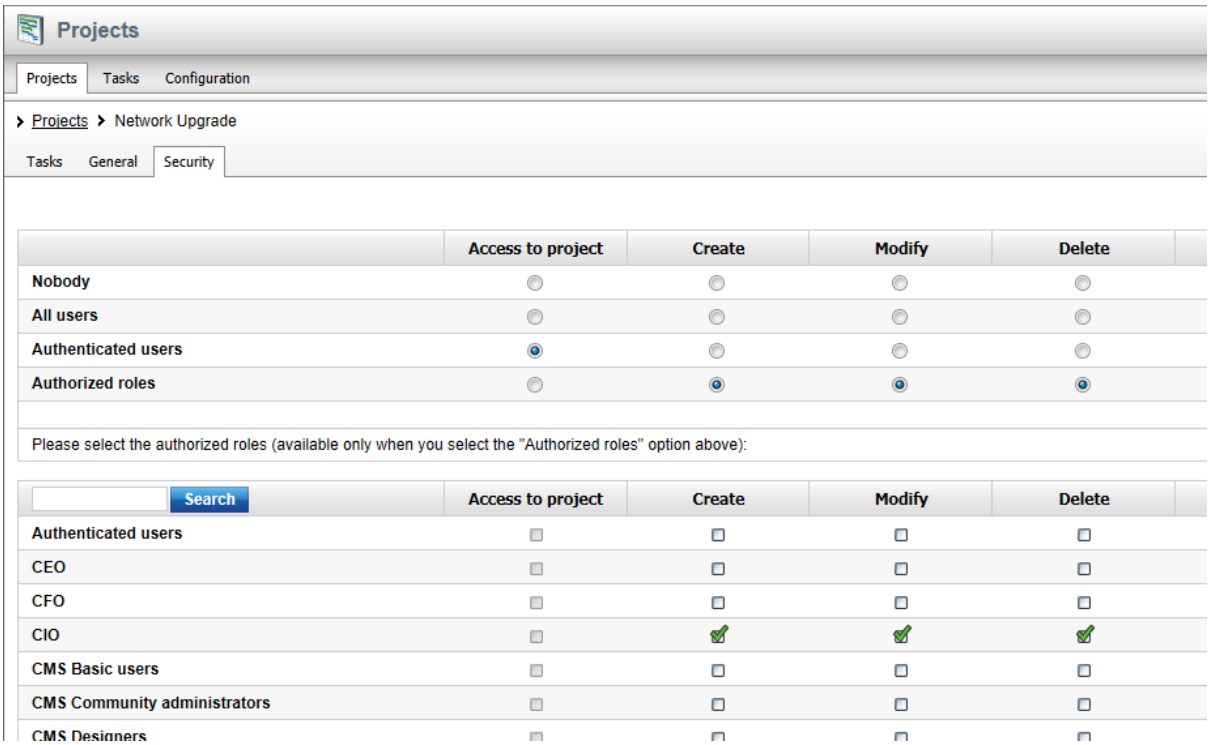
The following permissions can be assigned to members of the selected roles:

- **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.

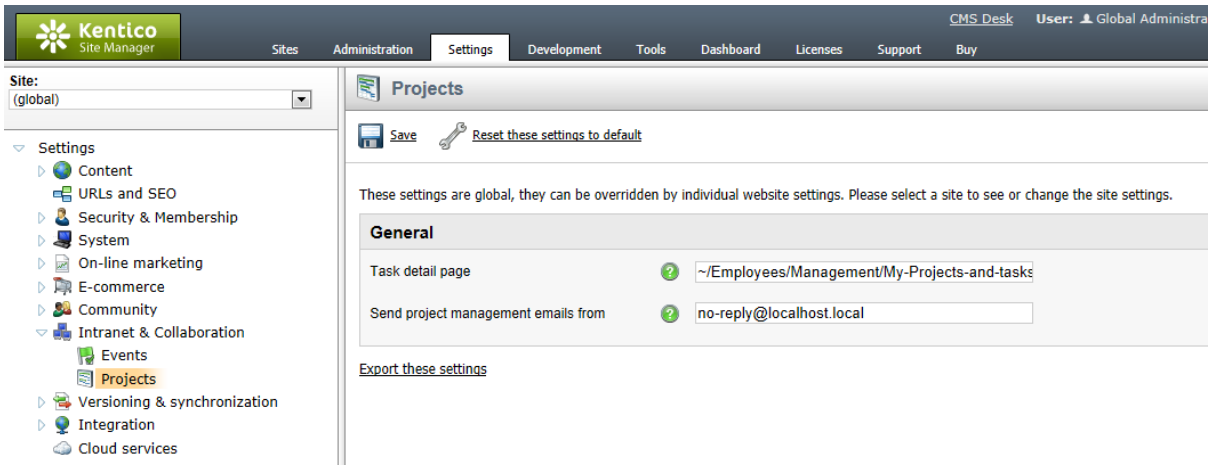
4. Users who belong to roles that can manage a group have full permissions for all projects under the given group.

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.

8.38.6 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.



8.38.7 Project management internals and API

8.38.7.1 Overview

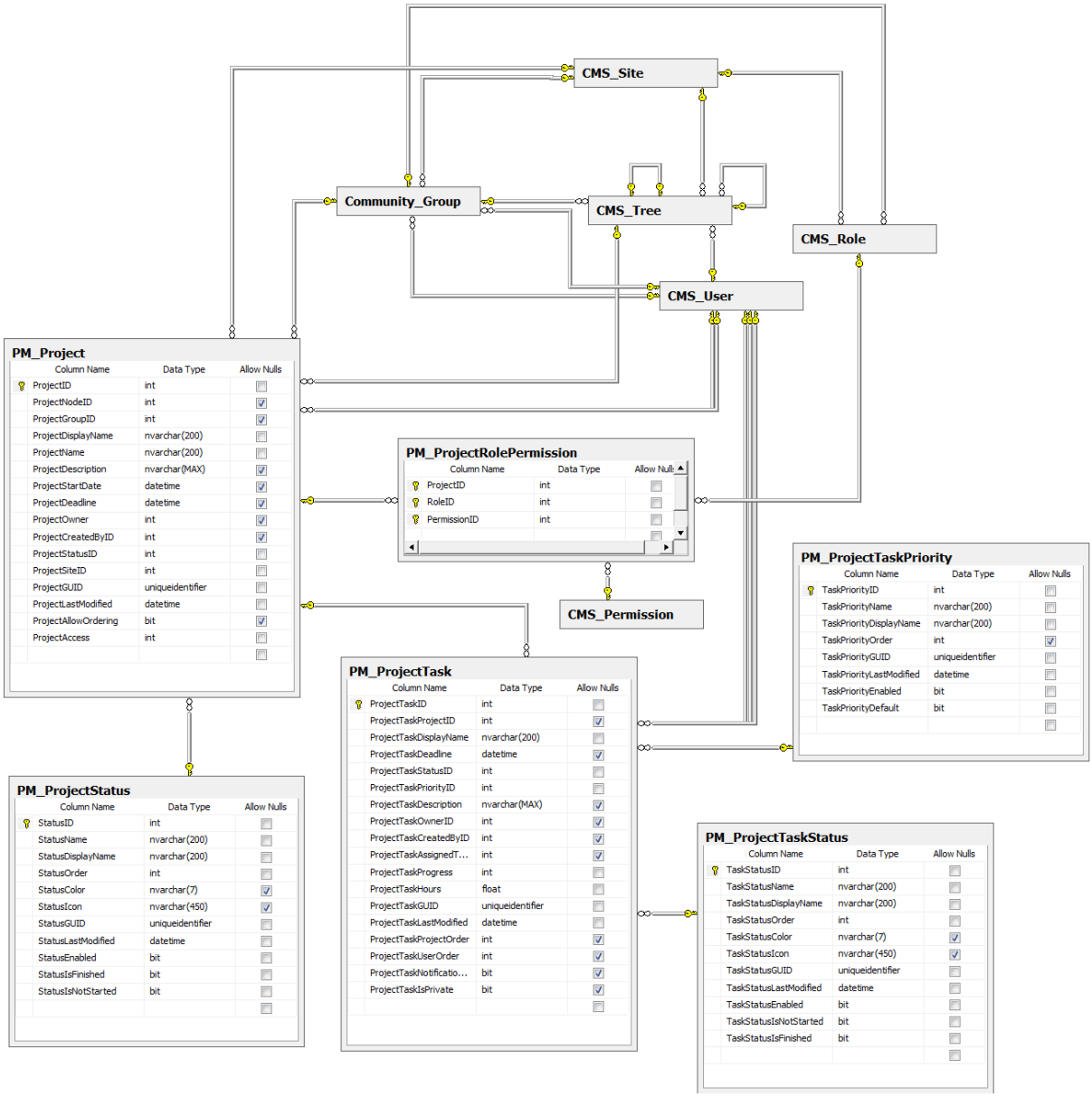
In this chapter, you will learn which database tables and API classes are used by the Project management module. You will also see the most common API examples.

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.

8.38.7.2 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.



8.38.7.3 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.

8.38.7.4 API examples

8.38.7.4.1 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;
```

8.38.7.4.2 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);
}
```

8.38.7.4.3 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;
}
```

8.38.7.4.4 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;
}
```

8.39 Reporting

8.39.1 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 a new report
- Defining report parameters
- Saving a report
- Displaying a report on the website

The Security topic describes how permissions can be set up for the 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!

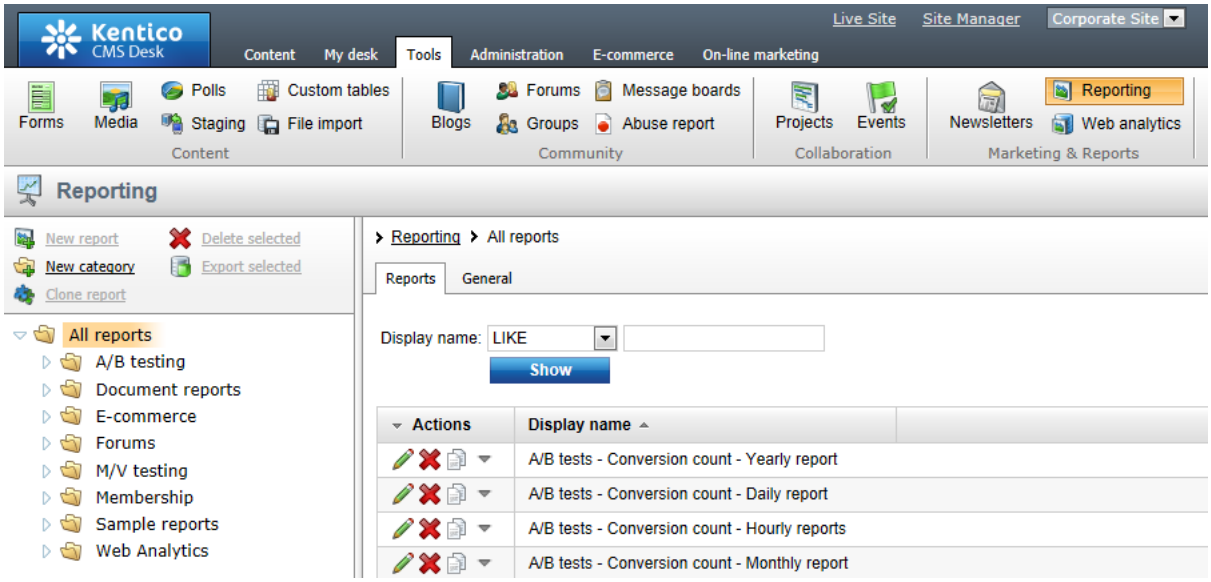
The graphs provided by the module are based on MS Charting Controls for ASP.NET.

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.

8.39.2 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 the following values:

- **Category display name:** User reports
- **Category code name:** UserReports

Click **OK**.

Continued in the example section of the Creating a new report topic.

8.39.3 Creating a new report

8.39.3.1 Creating a new report

New reports can be created by selecting a report category from the content tree at **CMS Desk -> Tools -> Reporting** and using the **New report** button. Existing reports can also be deleted, cloned or exported when selected.

When editing a report on its **General** tab, you can configure its properties, alter its layout using the built-in WYSIWYG editor and add file attachments. To retrieve and display data from the system in the report, some of the following objects must be added to the layout:

- Tables
- Graphs
- HTML graphs
- Values

Additionally, macro expressions as described in Development -> Macro expressions can be used in the report layout.

The output of the report can be seen on the **View** tab.

Localizing strings in reports

If you need to create a single report in multiple languages, please use Localization Expressions.

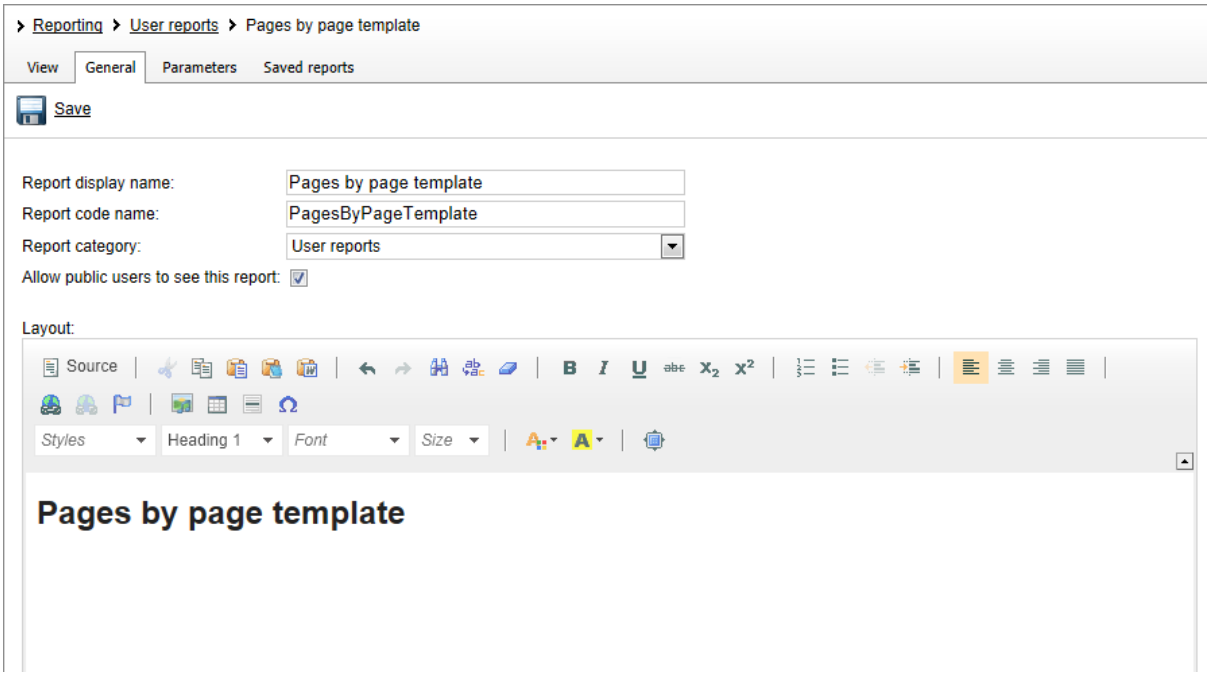
Example:

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. You will be redirected to the **General** tab of the report editing interface. Enter the following text in the **Layout** text area: *Pages by page template*. Select the text and use the WYSIWYG editor to set its **Format** to *Heading 1*.



Continued in the example section of the Creating a new Table topic.

8.39.3.2 Creating a new Table

A table can be placed into the layout of a report and can be used to display data retrieved from the Kentico CMS database by a query.

The following properties can be used to configure tables:

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 will be 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 will open a context menu with possible export actions.
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.
No record text	Text to be displayed if the query doesn't return any data.
Skin ID	ID of the .NET skin (stored in the *.skin* files in *~/AppThemes/<theme name>*) which will be used for the table.
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:**

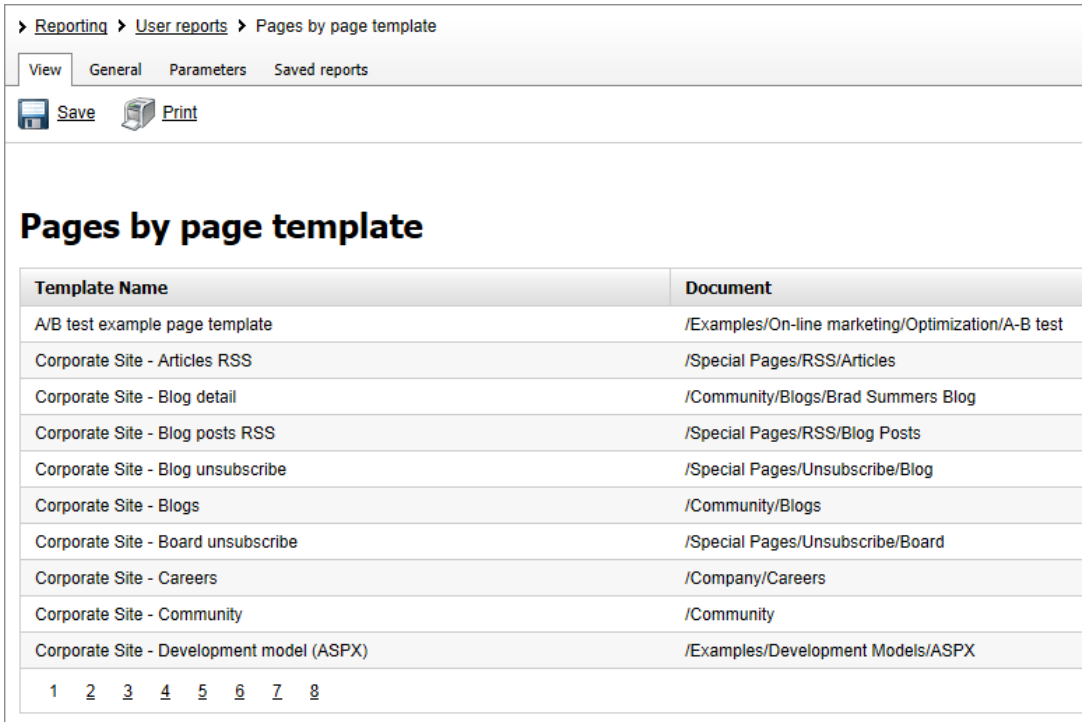
```
SELECT PageTemplateDisplayName AS [Template Name], DocumentNamePath AS [Document]
FROM View_CMS_Tree_Joined
LEFT JOIN CMS_PageTemplate
ON CMS_PageTemplate.PageTemplateID = View_CMS_Tree_Joined.DocumentPageTemplateID
WHERE PageTemplateDisplayName IS NOT NULL AND PageTemplateIsReusable = 1
ORDER BY PageTemplateDisplayName
```

- **Is stored procedure:** no
- **SkinID**: leave empty
- **Enable paging**: enabled
- **Page size**: 10
- **Page mode**: Page numbers

Click **OK**.

2. Now 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%%* will be added to the text area.

Click **Save** to save the changes and switch to the **View** tab. You will see a report like this:



Continued in the example section of the Creating a new Graph topic.

8.39.3.3 Creating a new Graph

A graph can be placed into the layout of a report and can be used to display data retrieved from the Kentico CMS database by a query.

The following properties can be used to configure graphs:

Default:

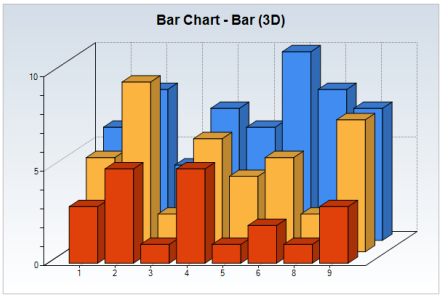
Display name	Display name of the graph shown in the user interface.
Code name	Code name of the graph.
Enable export	If enabled, users who view the graph will be able to export the displayed data to external files using the Microsoft Excel (XLSX), CSV or XML format. The data export feature may be accessed by right-clicking the graph in the report, which will open a context menu with possible export actions.

Query:

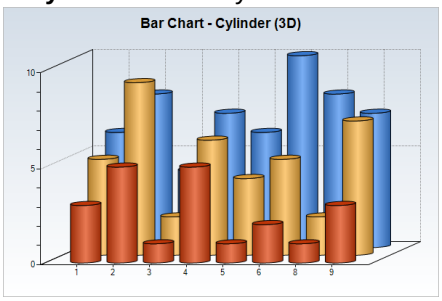
Query	Database query that extracts the data that will be displayed in the graph. It must return at least two columns: first one for categories, the other columns are used for values.
Is stored procedure	Indicates if the specified query is a stored procedure.
No record text	Text to be displayed if the 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

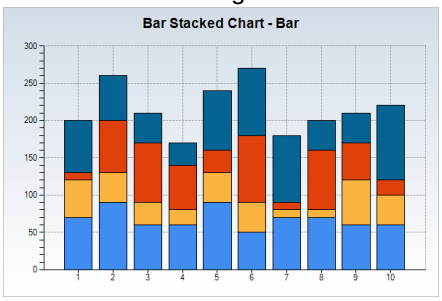


- **Cylinder** - uses cylinders

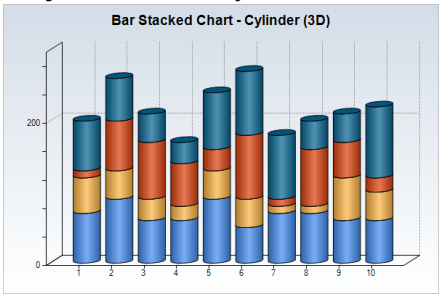


Bar stacked chart:

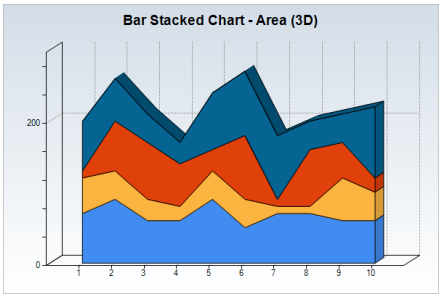
- **Bar** - uses rectangular bars



- **Cylinder** - uses cylinders

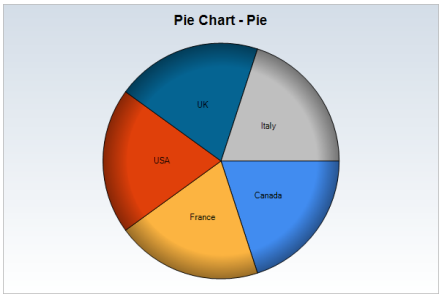


- **Area** - uses a line chart with the space under the lines filled with the respective color

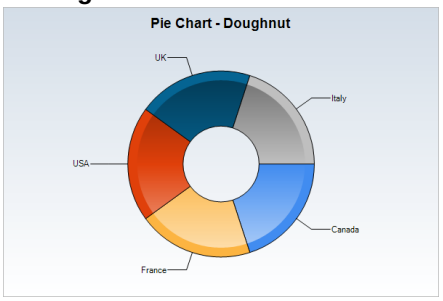


Pie Chart:

- **Pie** - uses a standard circular chart divided into sectors



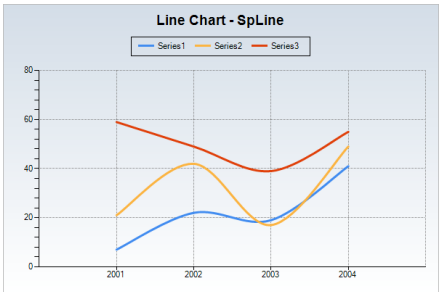
- **Doughnut** - uses a circular chart with a blank center



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 **New** in the **Graphs** section below the layout editor. Enter the following values:

- **Display name:** Favorite Page Templates
- **Code name:** FavoritePageTemplates
- **Query:**

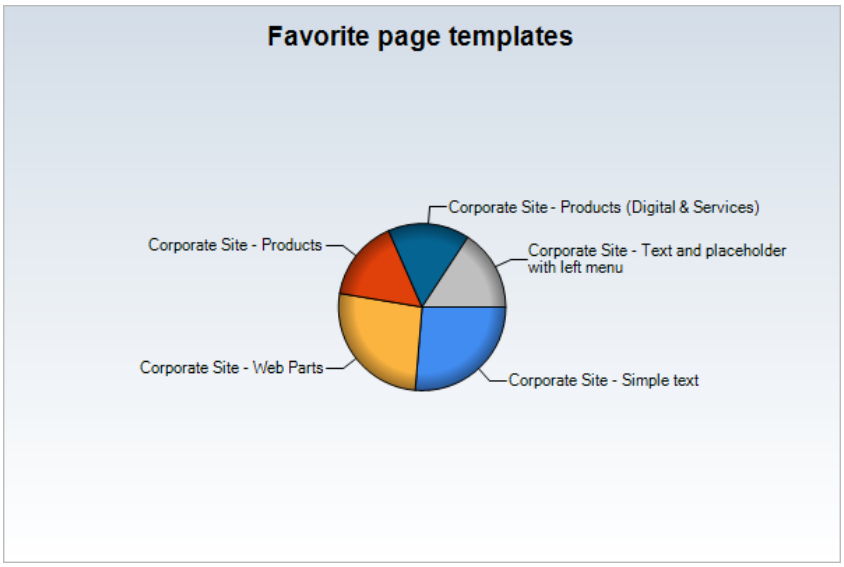
```
SELECT TOP 5 PageTemplateDisplayName AS [Template Name], COUNT
(PageTemplateDisplayName) AS [Usage]
FROM View_CMS_Tree_Joined
LEFT JOIN CMS_PageTemplate
ON CMS_PageTemplate.PageTemplateID = View_CMS_Tree_Joined.DocumentPageTemplateID
WHERE PageTemplateDisplayName IS NOT NULL AND PageTemplateIsReusable = 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%%* will be 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:



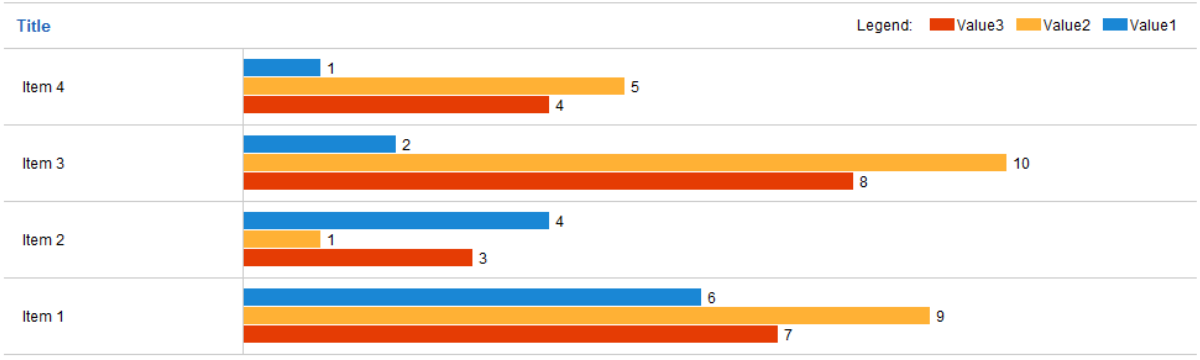
Continued in the example section of the Creating a new Value topic.

8.39.3.4 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 will be 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 will open a context menu with possible export actions.

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.
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).

8.39.3.5 Creating a new Value

A value is an object that can be placed 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:

Display name	The name of the item in the list
Code name	Name used in your code
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.
Formatting string	You can format the displayed value using the standard .NET expressions. Examples: • **{0}** - displays the value • **{0:F1}** - displays the value as a floating point number with one digit displayed after the decimal point You can find more details in .NET Framework documentation.

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 **New** in the **Values** section below the layout editor. Enter the following values:

- **Display name:** Number of pages with page template
- **Code name:** PagesWithTemplate

- **Query:**

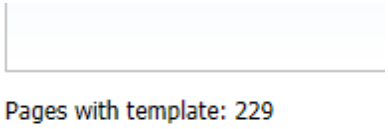
```
SELECT COUNT(DocumentID)
FROM View_CMS_Tree_Joined
WHERE DocumentPageTemplateID IS NOT NULL
```

- **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%%* will be added to the text area.

Click **Save** to save 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.

8.39.4 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:

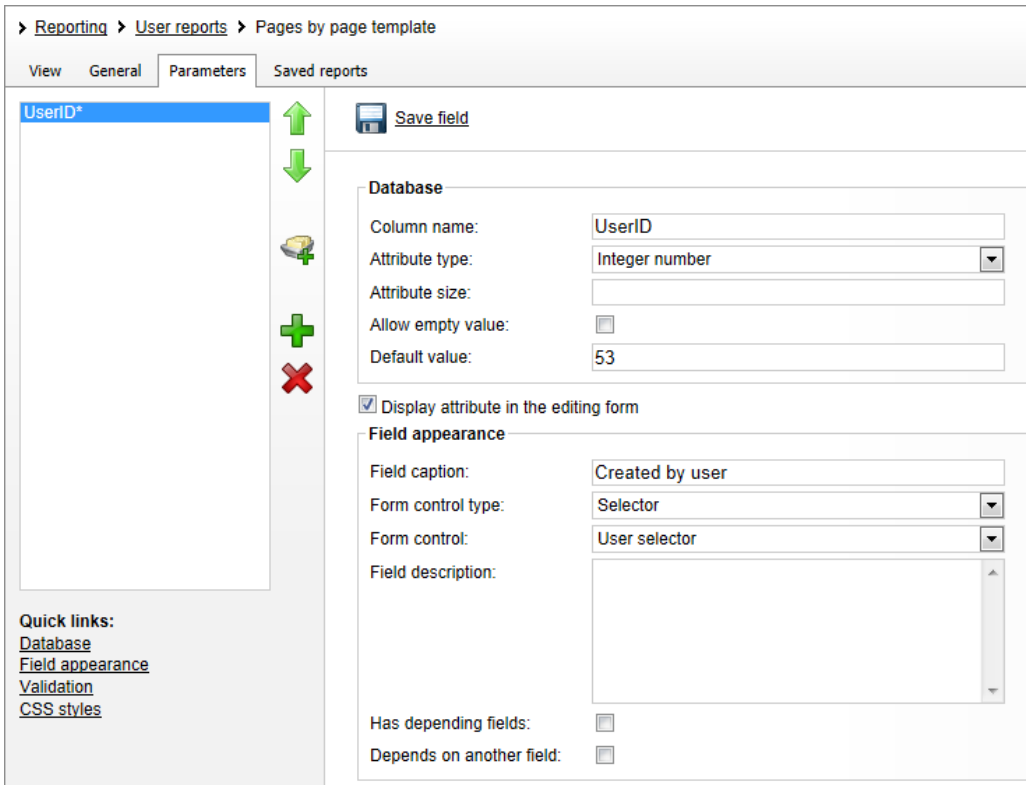
List of documents expired on or before 8/12/2007 12:06:49 PM

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
- **Attribute default value:** 53
- **Field caption:** Created by user
- **Form control type**: Selector
- **Form control:** User selector

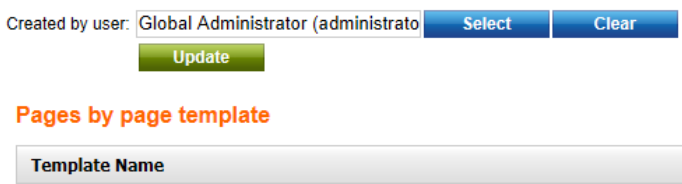


Click **Save field**.

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:



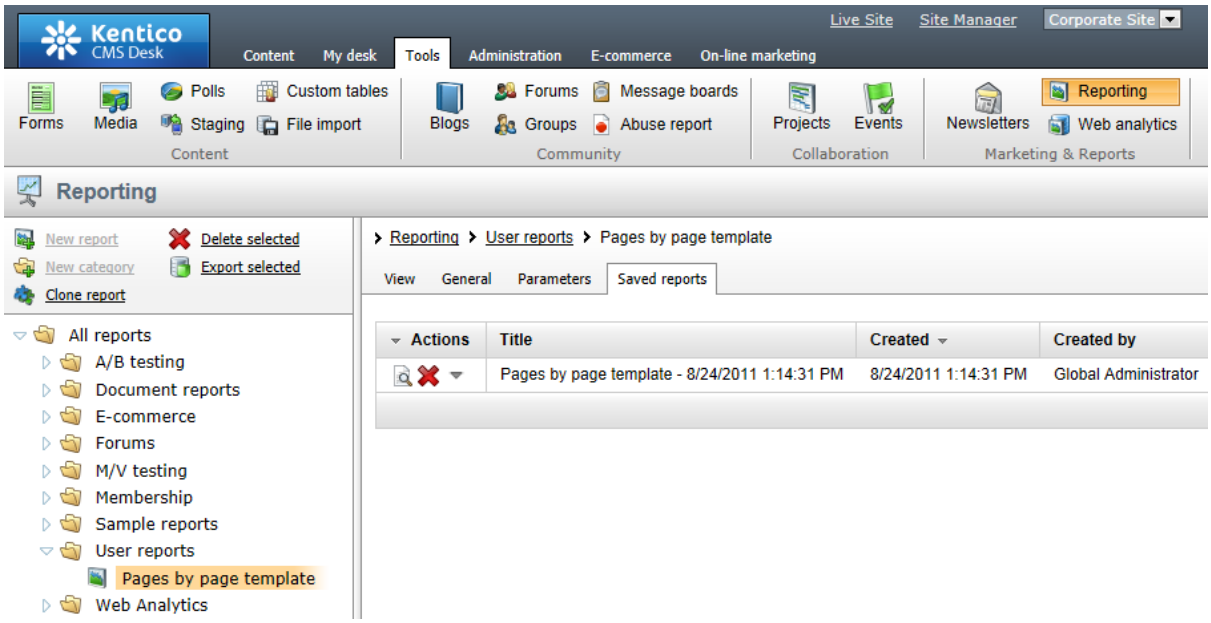
The table now only displays template names of documents that were created by the user specified in the filter.

Continued in the Saving a report topic.

8.39.5 Saving a report

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.



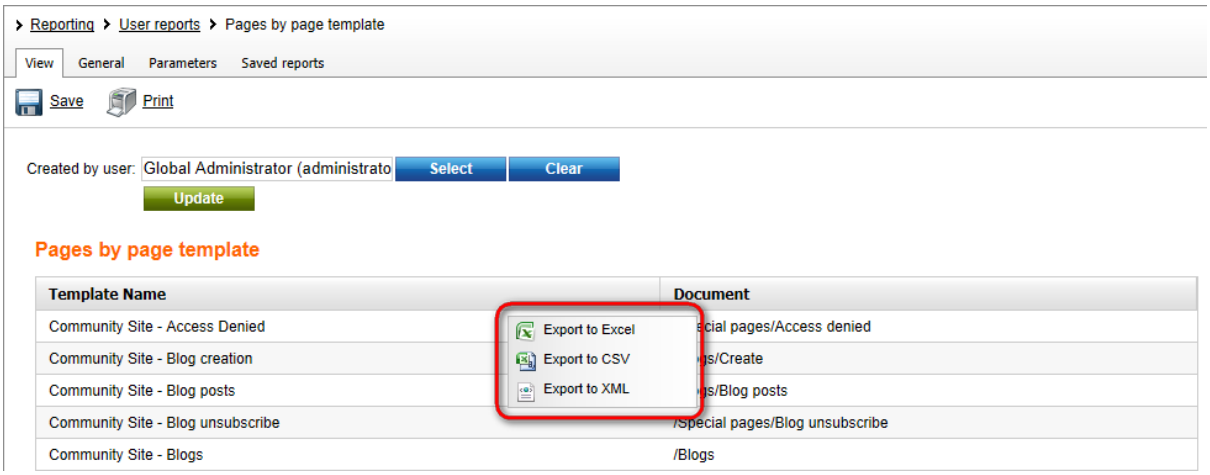
You can open the details of saved reports using the **Edit** ( ) action.

This tutorial is concluded in the example section of the Displaying a report 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.

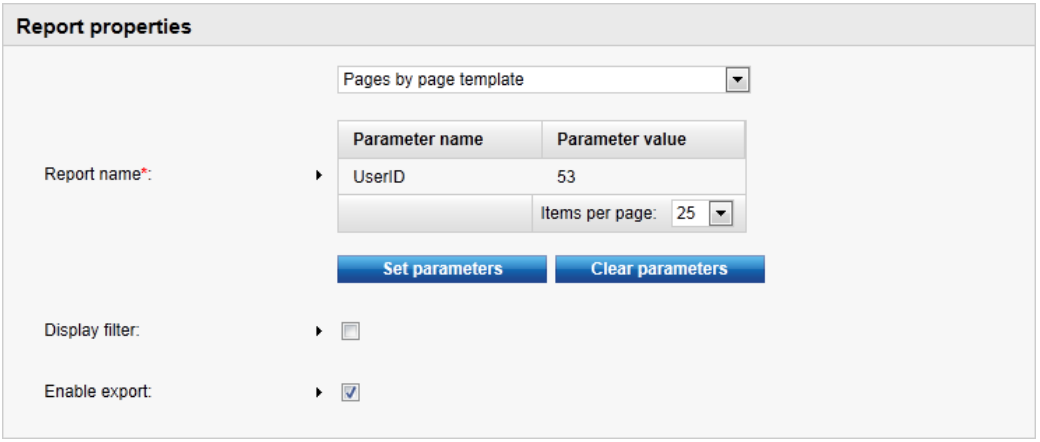
Please note that the data export function may be disabled for some tables and graphs, depending on the configuration of the given reporting object.

8.39.6 Displaying a report 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.

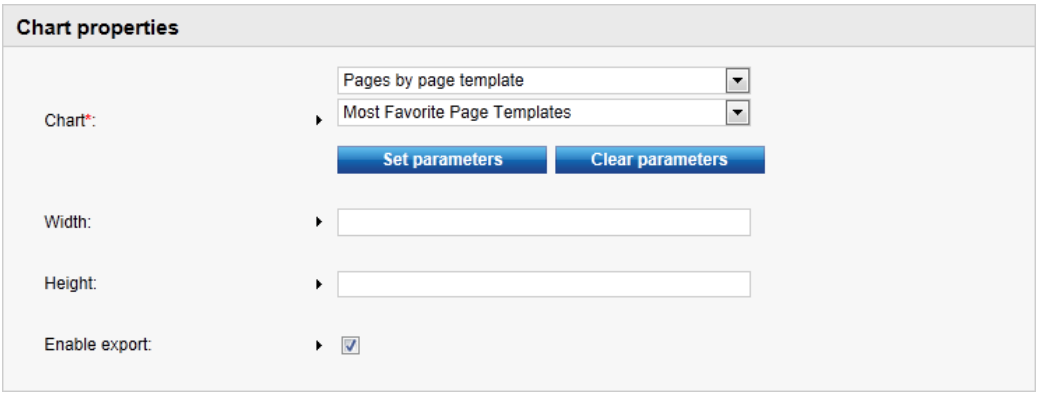
If the selected report has some parameters defined, their values can be entered here by using the **Set parameters** button. However, if the **Display filter** property is enabled, these settings will be ignored, as the parameters will be be configurable on the live site using the filter.



If you only wish to display a certain chart, 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. Parameters can be set using the **Set parameters** button.

These web parts are also available as Widgets.

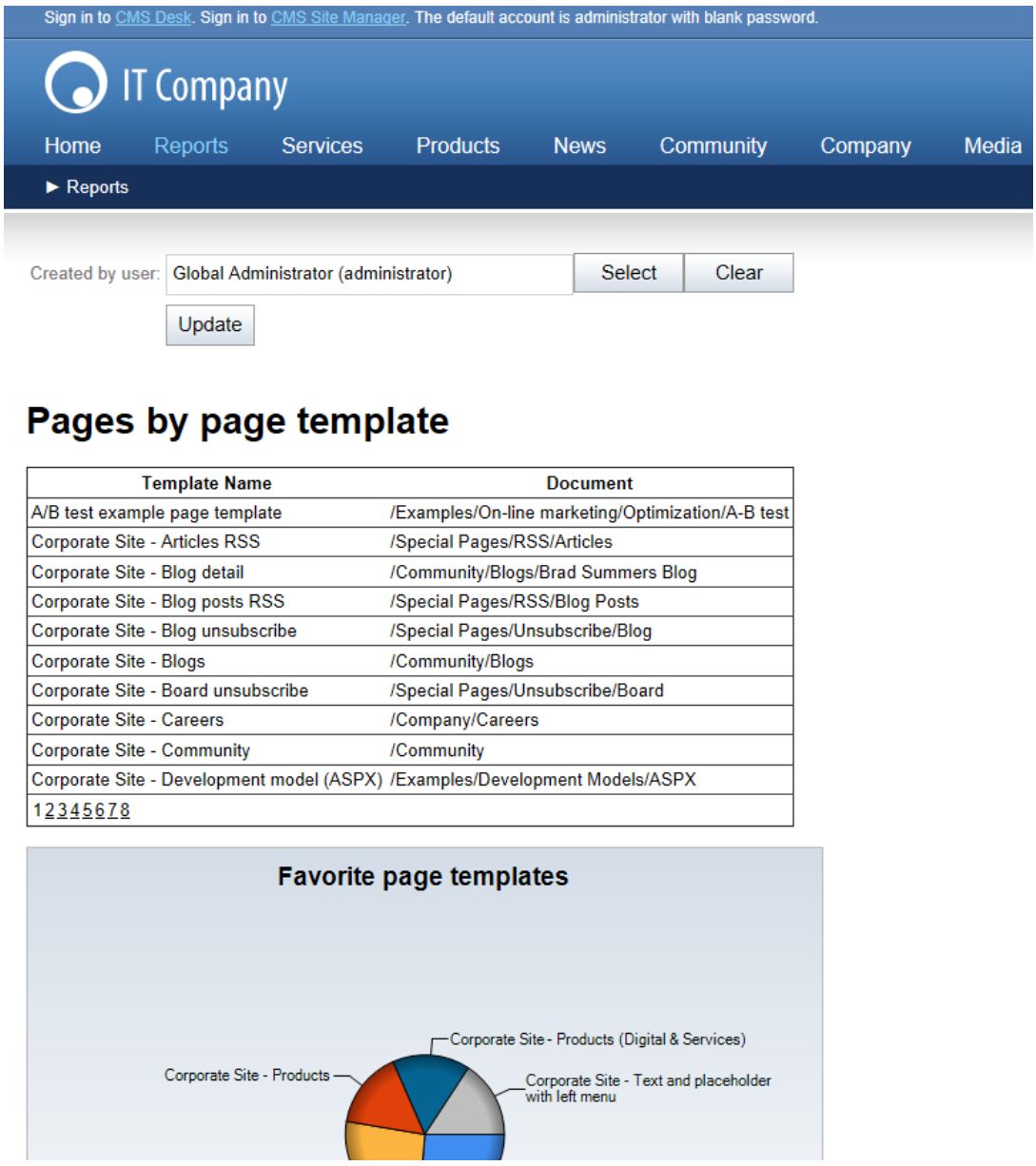
Example:

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:



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.

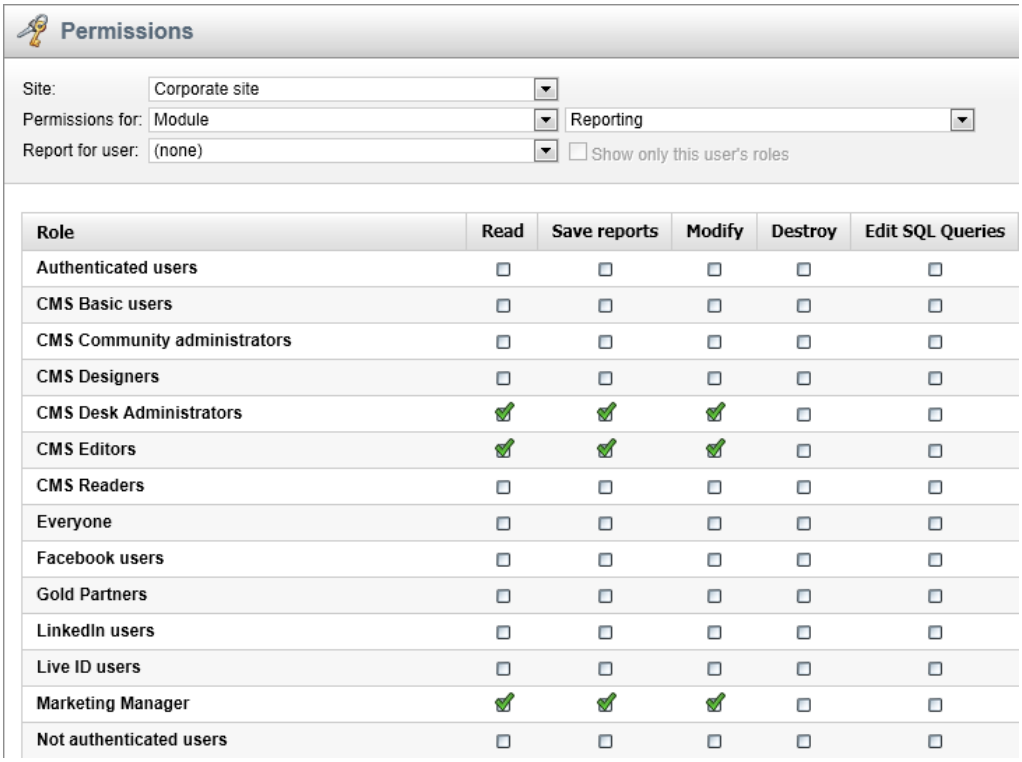
8.39.7 Security

You can configure access to the Reporting module in **CMS Desk -> Administration -> Permissions**. Choose the permission matrix for **Modules** -> **Reporting**.

You can assign roles with the following permissions:

- **Read** - allows members of the role to view existing reports.

- **Save reports** - allows members of the role to save reports.
- **Modify** - allows members of the role to create, modify and delete reports.
- **Destroy** - allows members of the role to delete the version history of reporting objects.
- **Edit SQL queries** - allows members of the role to edit the queries used to retrieve data for reporting tools (graphs, tables and values). This permission is necessary to create new reports, but it can be a security risk, since it allows users to run queries against the website's database.



Role	Read	Save reports	Modify	Destroy	Edit SQL Queries
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	☐	☐	☐	☐	☐

Making reports available on the live site

The **Allow public users to see this report** property, which can be found on the **General** tab when **Editing** () a report, indicates if the report can be displayed on the live site to non-authenticated (public) users. If it's set to false, the report and its graphs will be hidden from public users.

8.39.8 Reporting internals and API

8.39.8.1 Overview

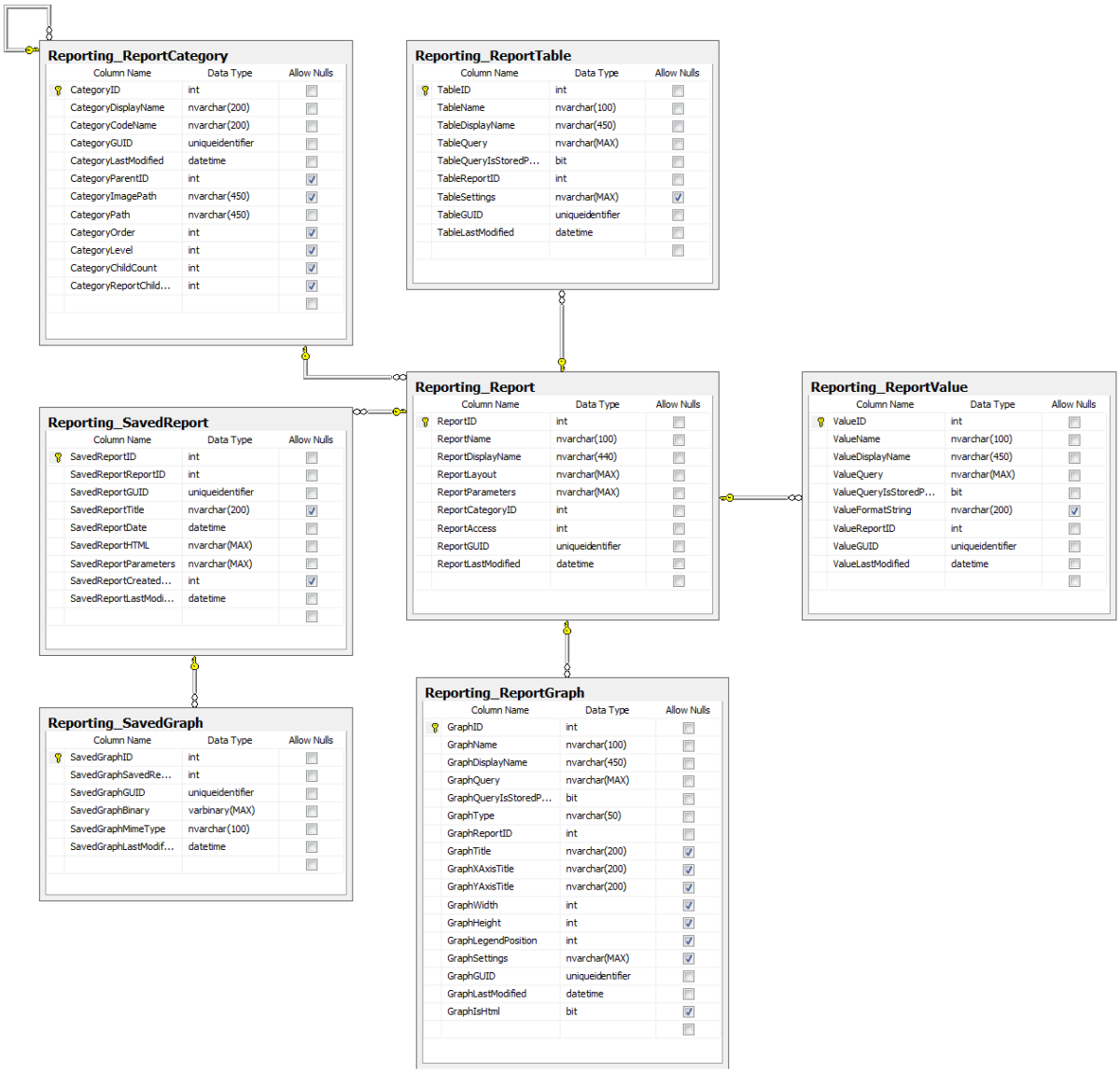
In this chapter, you will learn which database tables and API classes are used in the Reporting module. You will also see the most common API examples.

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.

8.39.8.2 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).



8.39.8.3 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 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.

8.39.8.4 API examples

8.39.8.4.1 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;
```

8.39.8.4.2 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);
}
```

8.39.8.4.3 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);
}
```

8.39.8.4.4 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);
}
```

8.39.8.4.5 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);
}
```

8.39.8.4.6 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);
}
```

8.39.8.4.7 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;
}
```

8.40 SharePoint integration

8.40.1 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.

8.40.2 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.

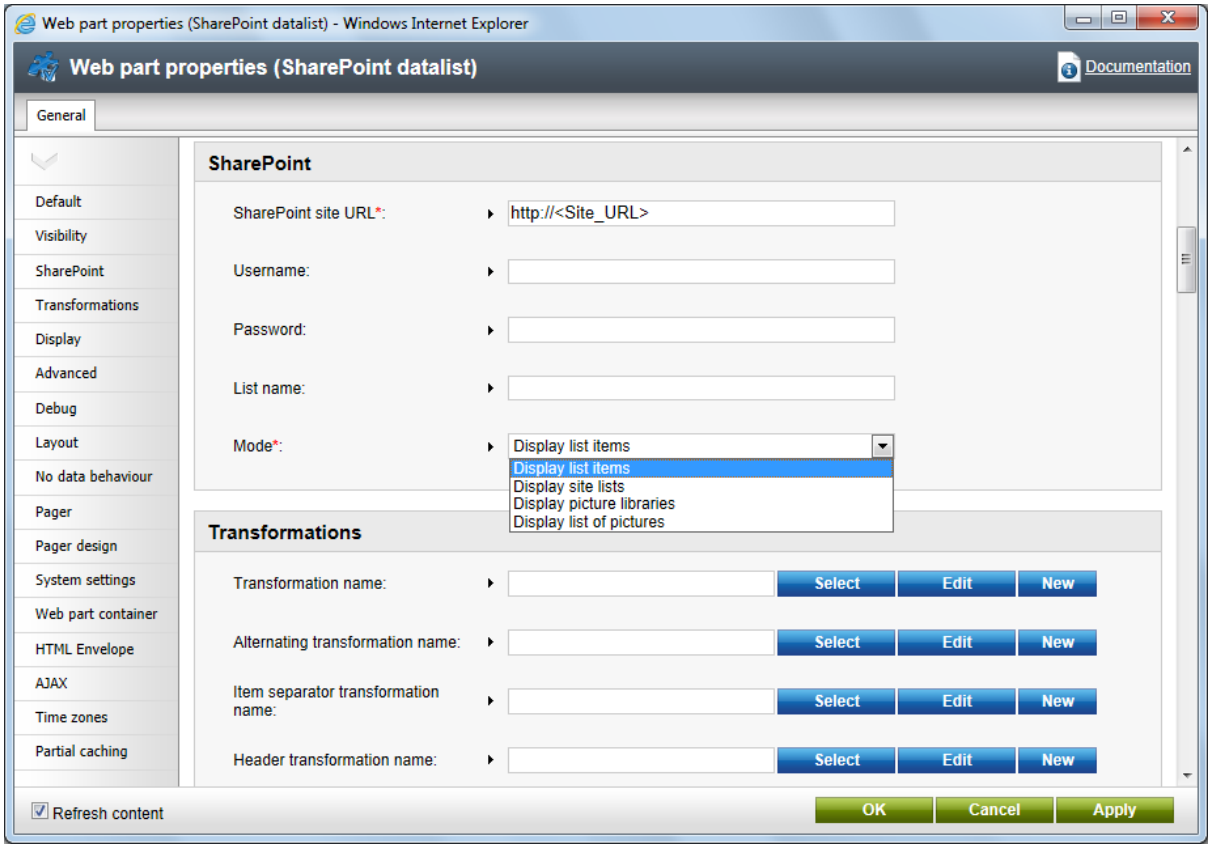
SharePoint

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.

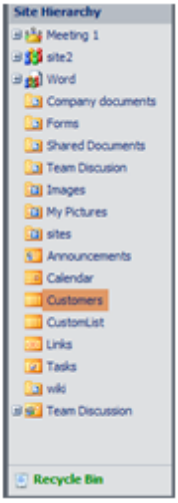
Debug

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.

8.40.3 Usage examples

8.40.3.1 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:

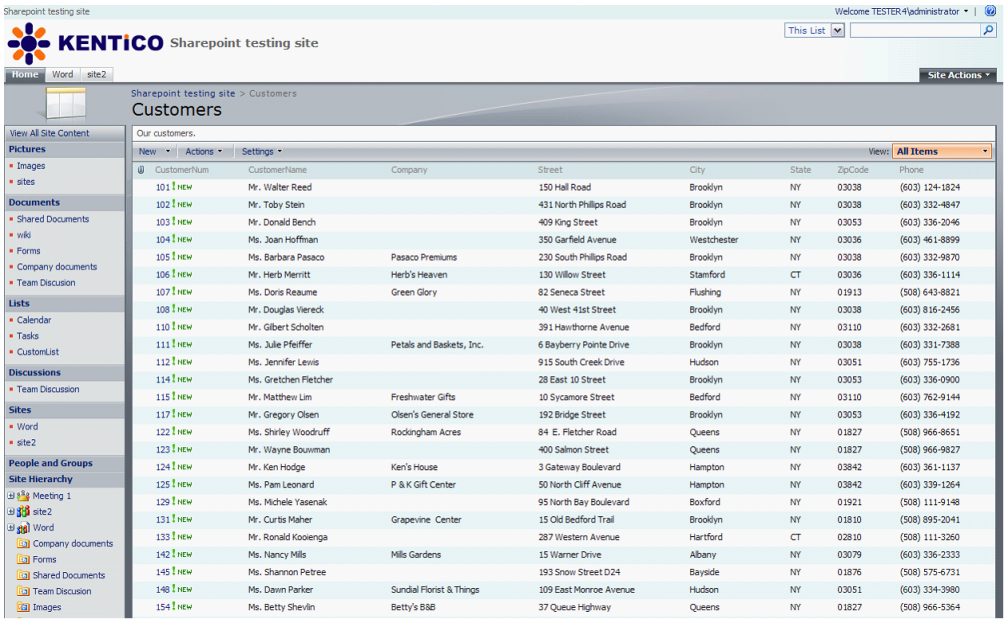
- **SharePoint site URL**: URL of your SharePoint server.
- **Mode**: Display site lists.
- **Username, password**: enter your username and password or leave the fields empty if you have them configured in *Site manager -> Settings -> Integration -> Microsoft SharePoint*.
- **Transformation**: there 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 > >|

8.40.3.2 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.

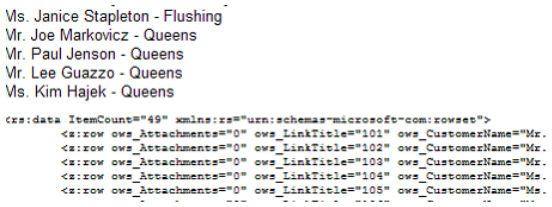
Replace the original **Transformation**:

```
<%# 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"
```

8.40.3.3 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:

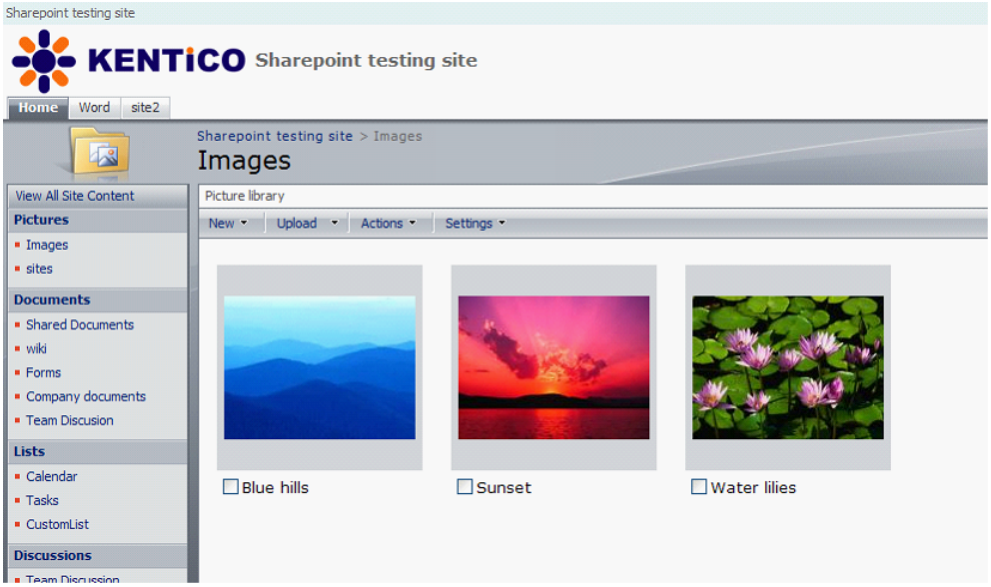
- **SharePoint site URL**: URL of your SharePoint server.
- **Mode**: Display picture libraries.
- **Username, password**: enter your username and password or leave the fields empty if you have it configured in *Site manager -> Settings -> Integration -> Microsoft SharePoint*.
- **Transformation**: there 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

8.40.3.4 List of pictures

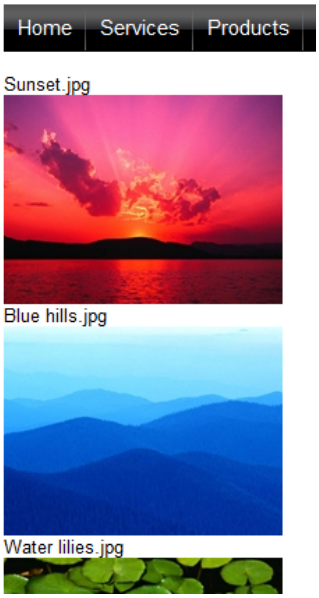
The *Display list of pictures* mode enables you to display pictures from picture libraries on a SharePoint server.



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.

8.40.3.5 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)**.

Important!

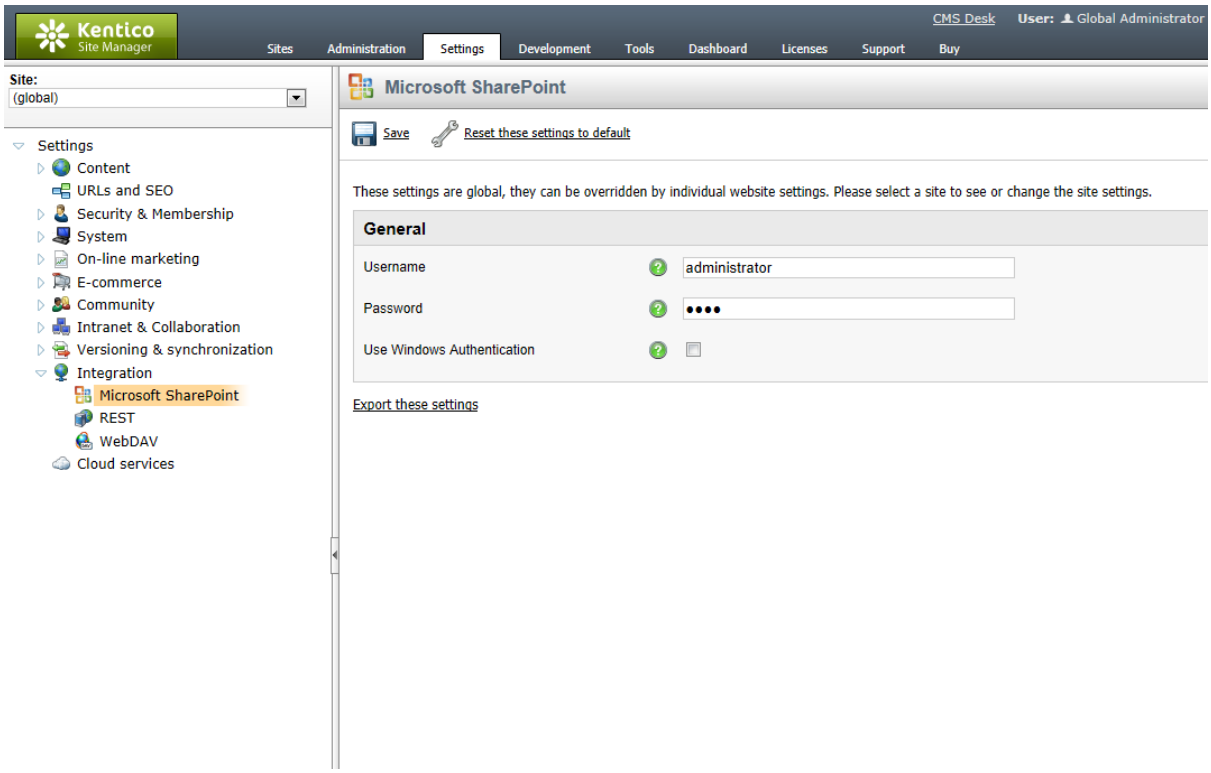
Because the settings configured in **Site Manager -> Settings -> Integration -> Microsoft SharePoint** are automatically used for authentication by the **GetSharePointFile.aspx** page and since the URL to access the page can be entered manually, it is highly recommended to enter the credentials of a user that is authorized to access only the files you want to display on your website.

8.40.4 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.

8.41 Smart search

8.41.1 Overview

The Smart search module allows index-based searching through the content of websites or other objects within the system. It is based on **Lucene.Net**, version 2.1.0, which is a source code, class-per-class, API-per-API and algorithmatic port of the Java **Lucene** search engine to the C# and .NET platform.

The module uses **indexes** to store information about the specified content. When a search request is sent to the system by a user, it is the index that gets searched, 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.

In order to allow the smart search functionality on your website, you need to perform the following three steps:

1. Enable Smart search indexing in the system. Learn how in the Enabling Smart search indexing topic.

2. Create an index. Assign it to a site (and culture) and define the content that should be indexed. Learn how in the Managing indexes sub-chapter.

3. Use some of the Smart search web parts on your site.

The module comes with several **web parts**. 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
- Security

In previous 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.

8.41.2 How it works

Information about the content specified for an index is stored in a physical **index file** on the local disk. The index files are located in **~/App_Data/CMSModules/SmartSearch/<Index code name>** within your web project's folder.

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 may contain fields that the corresponding Kentico CMS objects contain, based on the settings made on each object type's **Search fields** tab. Depending on these settings, a representation of the object is created in the index file by the **Index writer**. When an object included in the index is created, removed or has one of its fields modified, the index document is updated. The **Index searcher**, on the other hand, searches through the index file and returns the relevant results from it.

Brief example

Let's take the following model scenario to explain the life cycle of a document in the index file:

1. A new document is created and stored in the database. When the fields to be indexed are defined for the document type, a new indexing task is logged in the database upon the document's creation.

2. The Smart search checks the database automatically on a regular basis for the presence of indexing tasks.

3. When the Smart search module finds the task, it processes it and the new document is added to the

index.

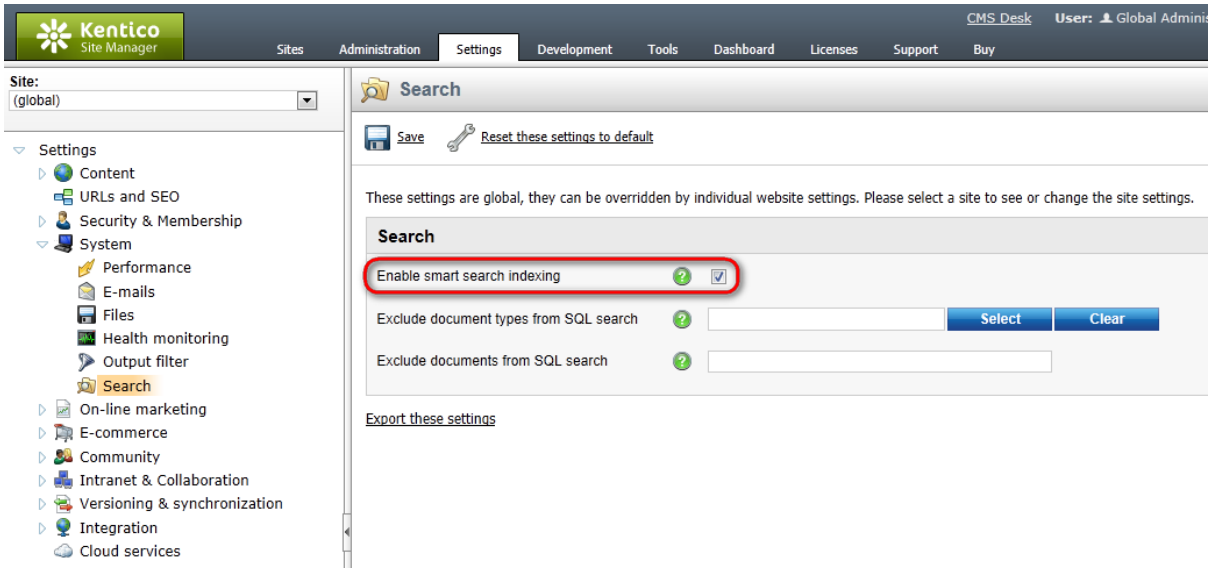
4. Now a user comes to the site and sends a search request via some Smart search web part.

5. The request gets passed to the Smart search module.

6. It modifies the request into a searchable form, searches the index and returns results based on what was found in the index.

8.41.3 Enabling Smart search indexing

To enable Smart search indexing, go to **Site Manager -> Settings -> System -> Search**, check the **Enable smart search indexing** check-box and click **Save**.



You also need to have the **write permission** assigned to the **~/AppData** folder. Indexes are stored within this folder, so the permission is necessary for the system to work with them. The procedure of granting a folder with the write permission is described in the Disk permission problems topic.

8.41.4 Managing indexes

8.41.4.1 Creating an index

Information about content is stored in indexes, which must be defined before any searches using the Smart search module can be performed.

There are seven types of indexes, click the link to learn about how indexes of the given type can be configured:

- Documents index - stores information about the content of documents in the content tree.
- Documents crawler index - stores information about the content of documents similar to a **Documents** index. However, the Documents crawler directly indexes the HTML output of documents.
- Forums index - stores information about the content of discussion forums in the system.
- Custom table index - stores information about data stored in custom tables.
- User index - stores information about site users.

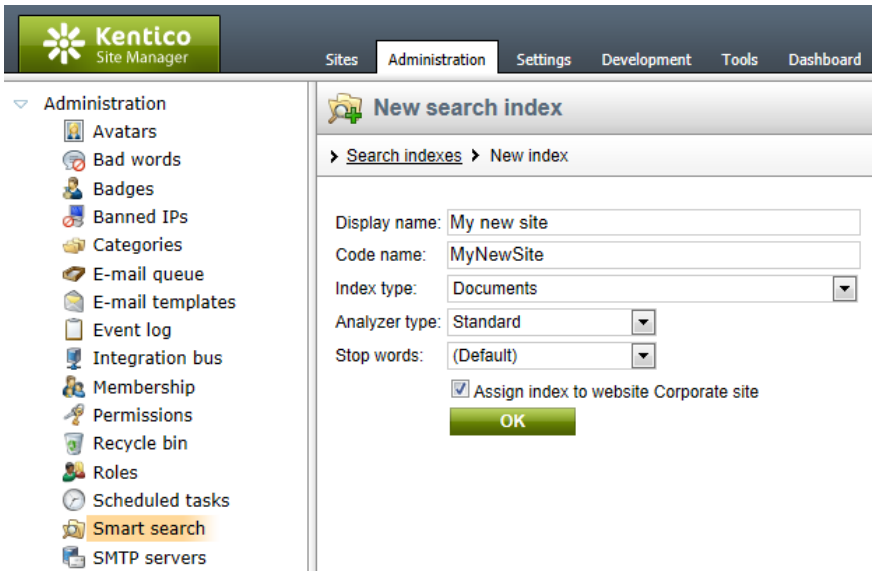
- General index - stores information about system objects of a specified type.
- Custom index - allows you to use your own custom-written search index, so it stores any kind of information depending on its implementation.

The following example describes the general procedure of search index creation and the options that are available. The procedure is applicable for all types of indexes and differences between them are noted in the text.

1. Go to **Site Manager -> Administration -> Smart search** and click the **New index** link. The **New search index** dialog will be displayed. You will be asked to enter the following details:

- **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 website code (the fully qualified file name must be less than 260 characters long, the directory name must be less than 241 characters long).
- **Index type** - sets the type of content to be indexed:
  - *Custom index* - indexes any kind of data depending on its implementation.
  - *Custom tables* - indexes records in custom tables.
  - *Documents* - indexes content of documents in the content tree.
  - *Documents crawler* - indexes the content of the HTML output generated by documents in the content tree.
  - *Forums* - indexes content of discussion forums.
  - *General* - indexes objects of a specified type. Any objects within the CMS can be searched this way.
  - *Users* - indexes details about system users (fields of the CMS_User system table).
- **Analyzer type** - type of analyzer that will be used when indexing content, the following types are available:
  - *Custom* - allows a custom-written analyzer to be specified. This gives you the option of performing tokenization according to your particular requirements. If selected, the names of the assembly and class that implement the custom analyzer must be entered into the **Assembly name** and **Class name** fields. An example can be found in the Using a custom analyzer topic.
  - *Keyword* - tokenizes the entire stream as a single token. This is useful for data like zip codes, ids, and some product names.
  - *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 entered string. Text is divided at whitespace characters. For example, searching for *test* returns words such as *test*, *tests*, *tester*, etc.
  - *Stop* - contains a collection of stop-words at which text is divided.
  - *Subset* - tokenizes all substrings in words, which allows searching for words that contain the entered string. Text is divided at whitespace characters. For example, searching for *net* returns words such as *net*, *Internet*, *network*, etc.
  - *White space* - divides text at whitespace characters.
- **Stop words** - dictionary containing words which will be omitted from indexing (e.g. 'and', 'or', ...) when a *Stop* or *Standard* analyzer is used. The dictionaries are stored in *~\App_Data\CMSModules\SmartSearch_StopWords*
- **Assign index to site <sitename>** - if checked, the index will be assigned to the site whose name is displayed.

Click **OK**.



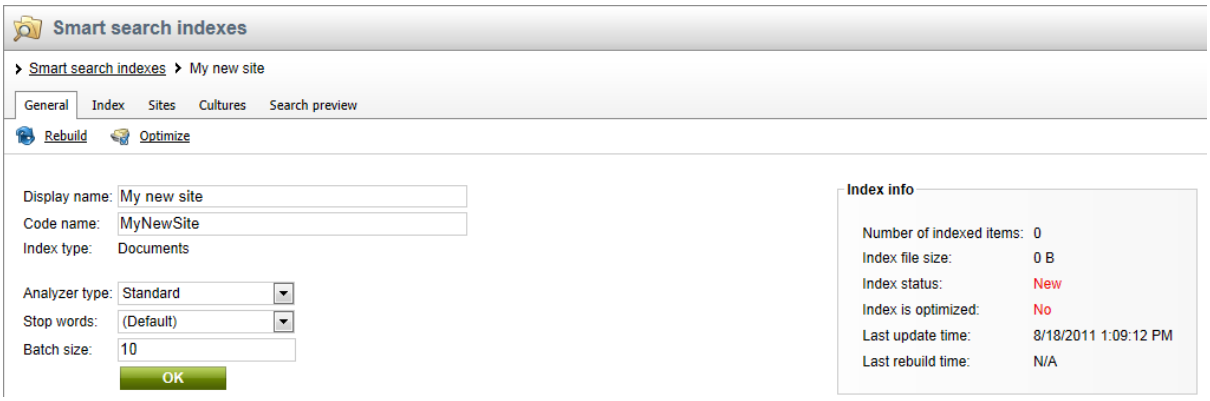
2. You will be redirected to the index's editing interface. The **General** tab, which will be displayed by default, allows the editing of the same properties entered when creating the index.

Additionally, the **Batch size** property can be set for the index, which limits the amount of records retrieved by a single query when rebuilding (or creating) the index. The purpose of this property is to help optimize indexing performance. Increasing the value of this property reduces the amount of required queries, which may improve performance, but doing this also increases memory consumption. The ideal value depends on the type of the indexed objects and on available resources. It is recommended to set a reasonable value when indexing large objects (e.g. documents).

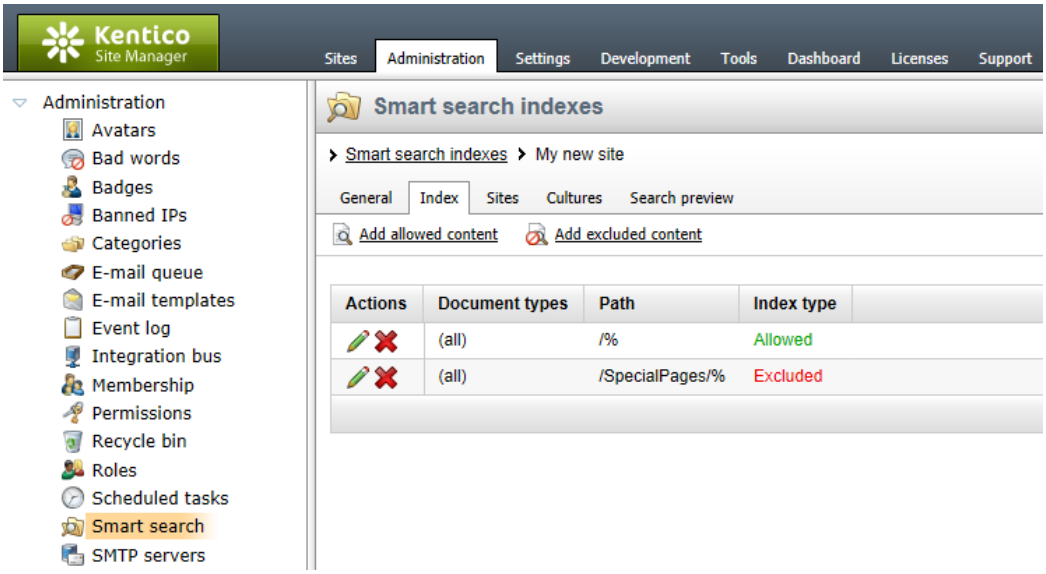
Notice the **Index info** box on the right, which displays current information about the status and properties of the index.

The following two actions can also be performed on this tab, but it would make no sense to perform them at this stage of creating a new index:

- **Rebuild** - deletes the original index and the specified content gets indexed again. Clicking this action button does not always guarantee that the index will be rebuilt immediately, the process may be delayed if e.g. another index is already being rebuilt or if the rebuilding tasks are configured to be handled by the scheduler. The index is automatically optimized after a successful rebuild.
- **Optimize** - de-fragments the index, which results in better search performance, particularly in the case of large indexes.

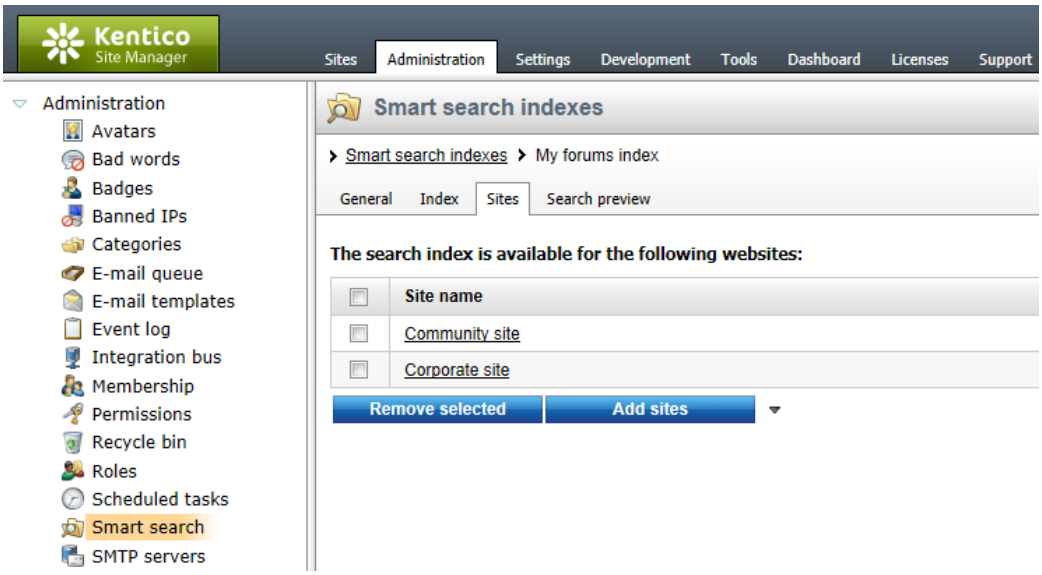


3. Switch to the **Index** tab. This is where you define which documents, forums, custom tables, users or other objects will be indexed. The content of this tab depends on the type of index that you are creating. Detailed information about defining index content is given in Defining document index content, Defining forums index content, Defining custom tables index content, Defining user index content, Defining general index content and Defining custom index content.

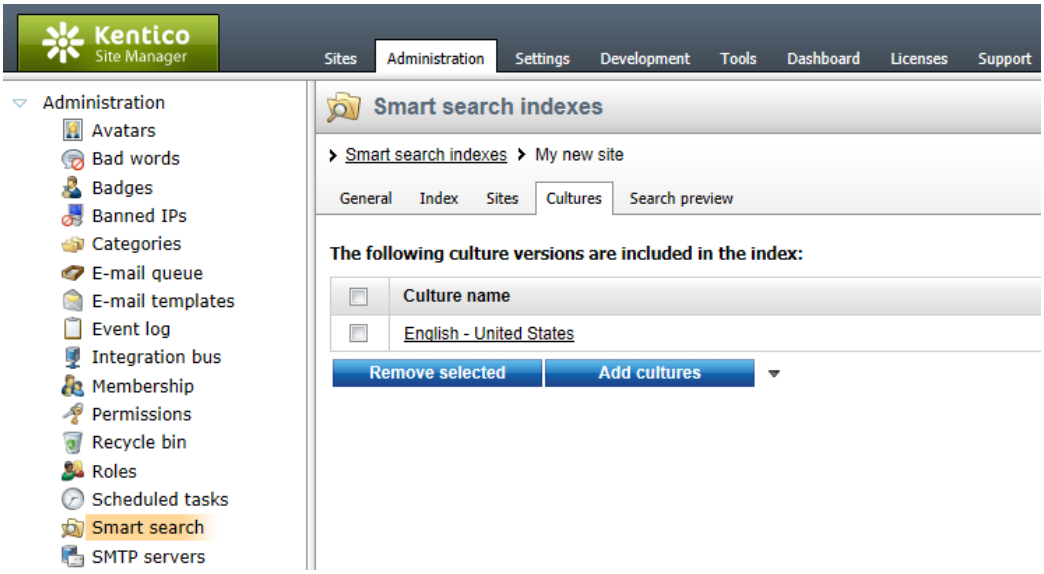


4. Now switch to the **Sites** tab. Make sure that the index is assigned to the appropriate website. You may also want to optionally assign the index to some other sites in order to have multi-site search results.

If the index is defined for global objects that are not site-specific, the selection made here will not affect the index's content. However, the index will still only be available for use (through smart search web parts) on the sites chosen on this tab.



5. Switch to the **Cultures** tab (only available for *Documents* and *Documents crawler* type indexes). This is where you can choose which cultural versions of documents will be indexed. At least one culture must be selected in order for the index to be functional. If you have a multi-site index, you can select the cultures separately for each site chosen by the **Select site** drop-down list.



6. Finally, go back to the **General** tab and choose to **Rebuild** the index. This needs to be done only for the first time. Any further changes made to the specified content will be indexed automatically. If you wish, you can quickly test the index by switching to the **Search preview** tab.

8.41.4.2 Defining document index content

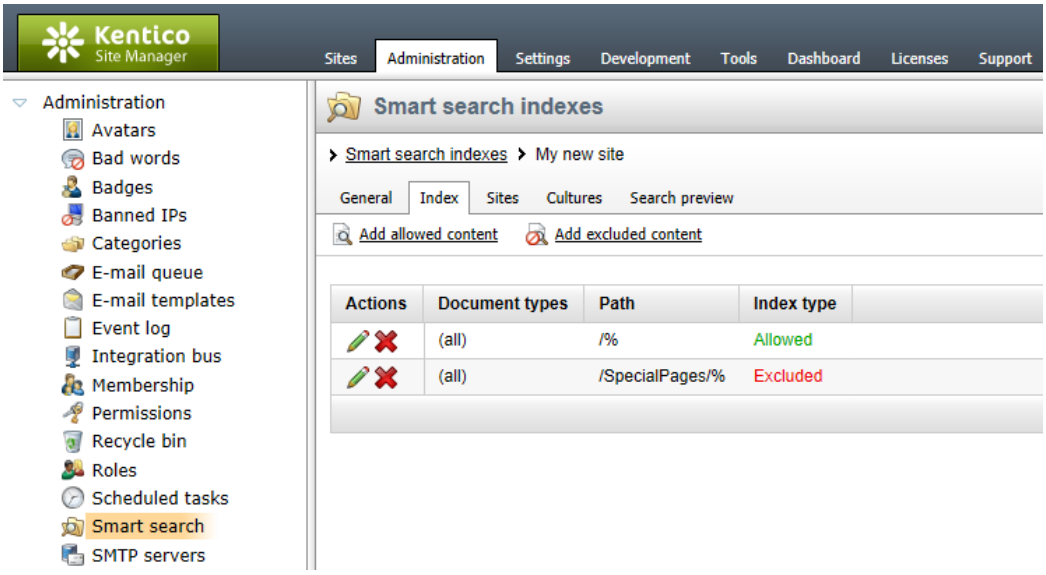
There are two types of search indexes available for the documents in a website's content tree. **Documents** type indexes include information from general document fields such as metadata, the text content of certain web parts placed on page (menu item) documents, as well as the selected fields of individual documents types as 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, which may be specified using keys in your project's web.config file. All keys related to smart search indexes may be found 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. This is done by specifying allowed and excluded content on the **Index** tab of the interface displayed when editing an index.

The dialogs for defining new allowed/excluded content can be accessed using the **Add allowed content** and **Add excluded content** links.



Adding allowed content

In this dialog, you are defining what part of the website will be indexed. You can specify this by choosing the **Path** to the indexed documents, **Document types** of the indexed documents or a combination of both. The following options can be specified:

- **Path** - path to the documents that should be indexed.
- **Document types** - document types that should be included in indexing.

The following properties define types of additional content that may be included in *Documents* search indexes. They are not available for *Documents crawler* indexes:

- **Including ad-hoc forums** - if checked, ad-hoc forums placed on documents specified by the properties above (if there are any) will be indexed too.
- **Including blog comments** - if checked, blog comments placed on blog posts specified by the

properties above (if there are any) will be indexed too.
- **Including message boards** - if checked, message boards placed on documents specified by the properties above will be indexed too.

Documents that have their **Exclude this document from search** property enabled will not be indexed. This property can be configured by selecting a document from the content tree in **CMS Desk** and going to **Content -> Edit -> Properties -> General**.

Examples:

- **Path**: /%
- **Document types**: empty

In this case, all documents on the site will be indexed.

- **Path:** /Partners
- **Document types**: empty

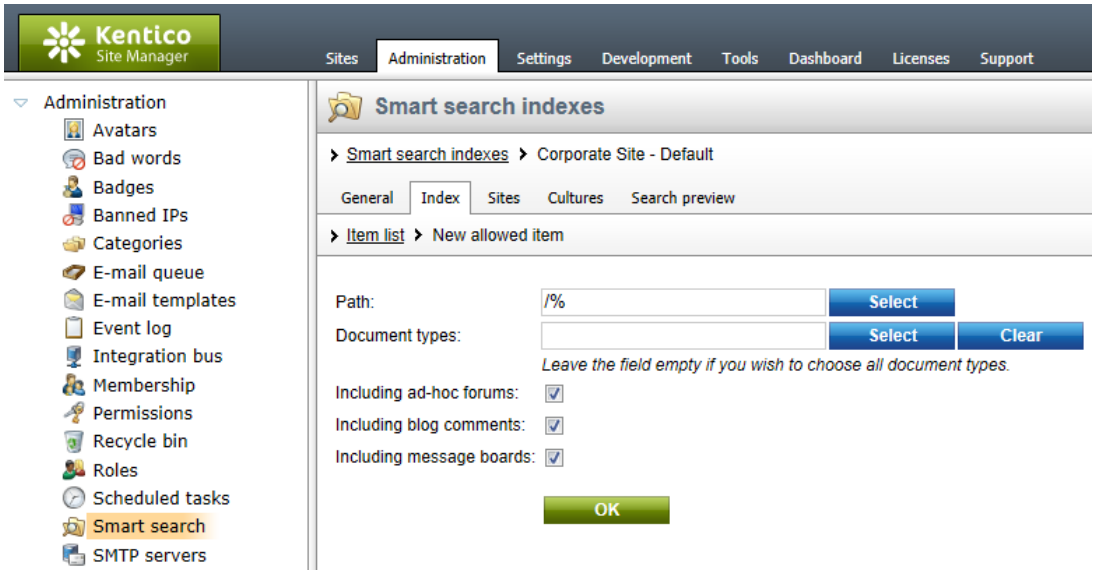
In this case, only the /Partners page will be indexed, without the pages under it.

- **Path**: empty
- **Document types**: CMS.News

In this case, all documents of the CMS.News document type on the whole site will be indexed. Please note that in this case, the empty path field is equal to **/%**.

- **Path**: /Products/%
- **Document types**: CMS.CellPhone;CMS.Pda

In this case, all documents of the CMS.CellPhone and CMS.Pda document types found under /Products will be indexed.



New excluded content

In this dialog, you are defining what part of the website will be excluded from indexing.

Excluded content is removed from the allowed content. So for example if you allow **/%** and exclude **/Special-pages/%** at the same time, it means that all pages on the site will be indexed, except for all pages found under the **/Special-pages** node.

The following options can be specified:

- **Path**: path to the documents that should be indexed
- **Document types**: document types that should be included in indexing

Examples:

- **Path:** /Partners
- **Document types**: empty

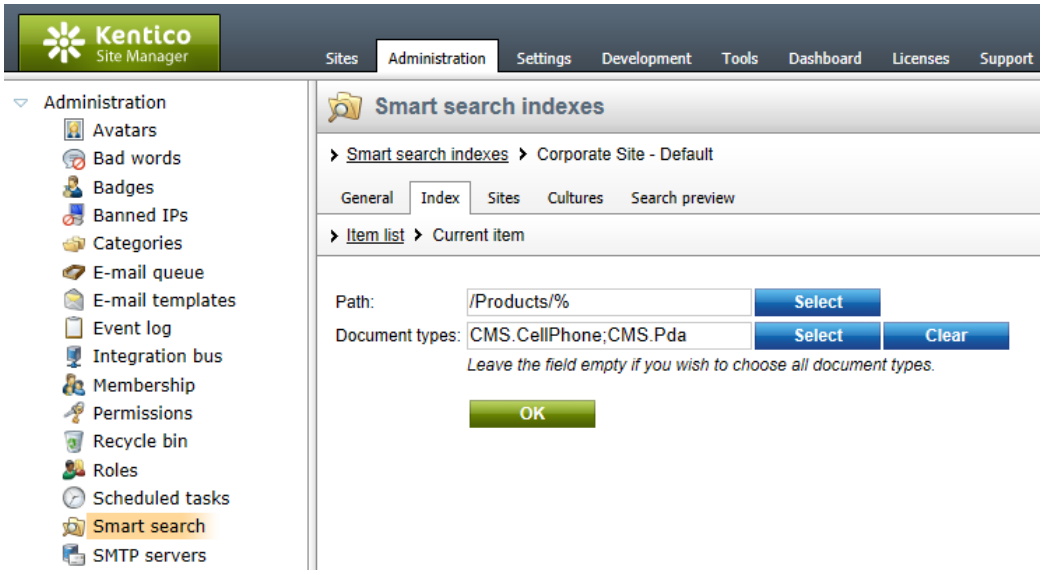
In this case, only the /Partners page will be excluded from indexing, not the pages under it.

- **Path**: empty
- **Document types**: CMS.News

In this case, all documents of the CMS.News document type on the whole site will be excluded from indexing. Please note that in this case, the empty path field is equal to **/%**.

- **Path**: /Products/%
- **Document types**: CMS.CellPhone;CMS.Pda

In this case, all documents of the CMS.CellPhone and CMS.Pda document types found under /Products will be excluded from indexing.



Setting the page content included by document crawler indexes (API)

By default, the output of documents is converted to plain text (stripped of all HTML tags, JavaScript and whitespace formatting) before it is saved to the index. It is possible to change this behavior if you wish to

store the content of any tags in documents crawler indexes. This can be done by implementing custom functionality in a handler for the **OnHtmlToPlainText** event of the **CMS.SiteProvider.SearchHelper** class, which is triggered when the HTML output is processed by a crawler.

For example, the customization can be performed by adding a custom class to the **~/App_Code** folder (or **~/Old_App_Code** if you installed the project as a web application). You can define the content of the class as shown below:

[C#]

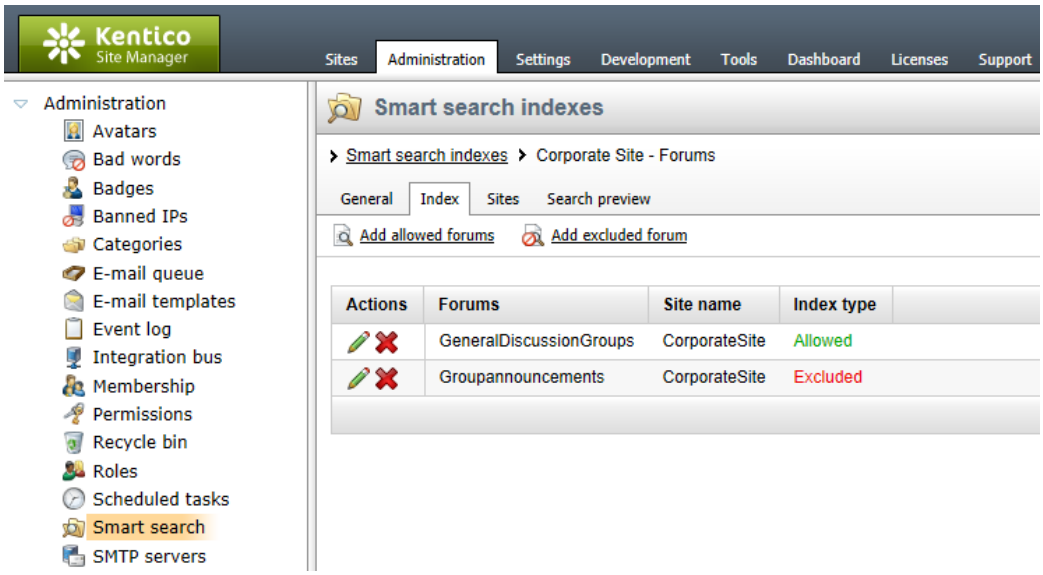
```
using CMS.SettingsProvider;
using CMS.SiteProvider;

[DocumentCrawlerContentLoader]
public partial class CMSModuleLoader
{
    /// <summary>
    /// Module registration
    /// </summary>
    private class DocumentCrawlerContentLoaderAttribute : CMSLoaderAttribute
    {
        /// <summary>
        /// Called automatically when the application starts
        /// </summary>
        public override void Init()
        {
            // Hook 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 **plainText** parameter contains the document output already stripped of all tags and converted to plain text, while **originalHTML** can be used to access the raw page code without any modifications.

8.41.4.3 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.

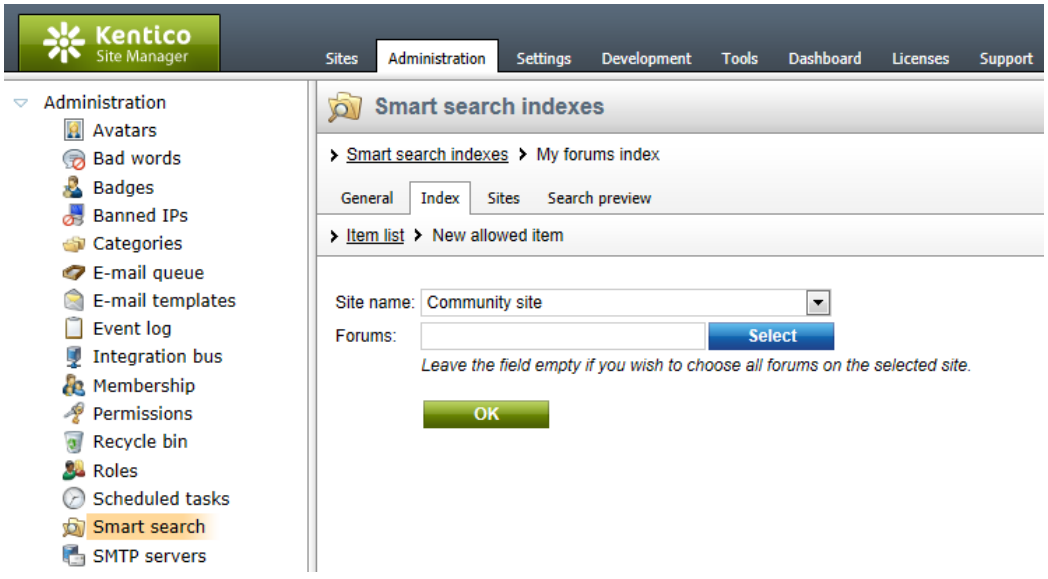
The dialogs for defining new allowed/excluded content can be accessed using the **Add allowed forums** and **Add excluded forum** links.



Adding allowed forums

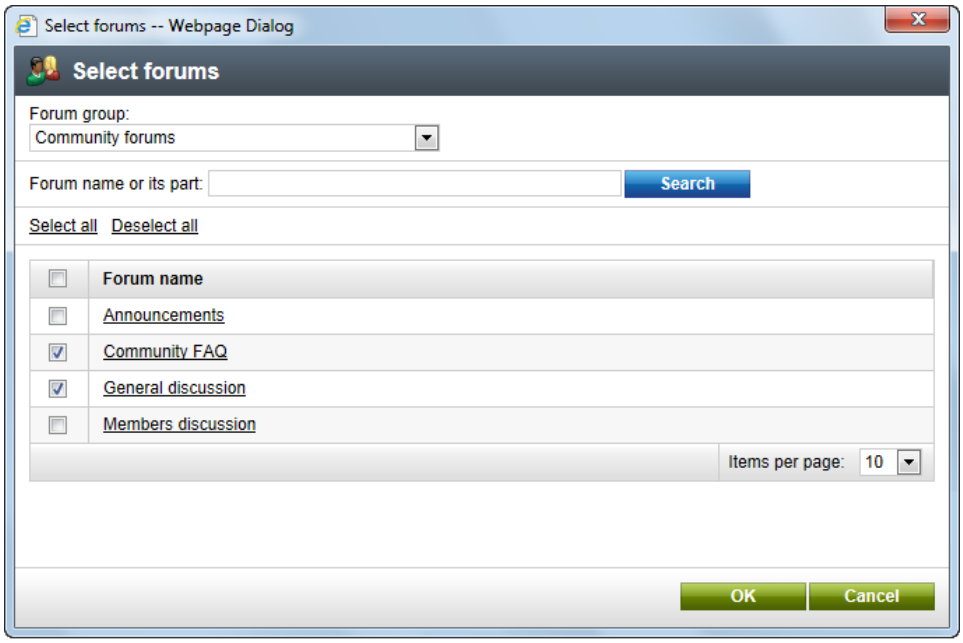
1. Click the **Add allowed forums** link.

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, the code names of forums, separated by semicolons, can be entered manually into the **Forums** field. 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** would add all forums that contain the string *community* in their code name to the index.

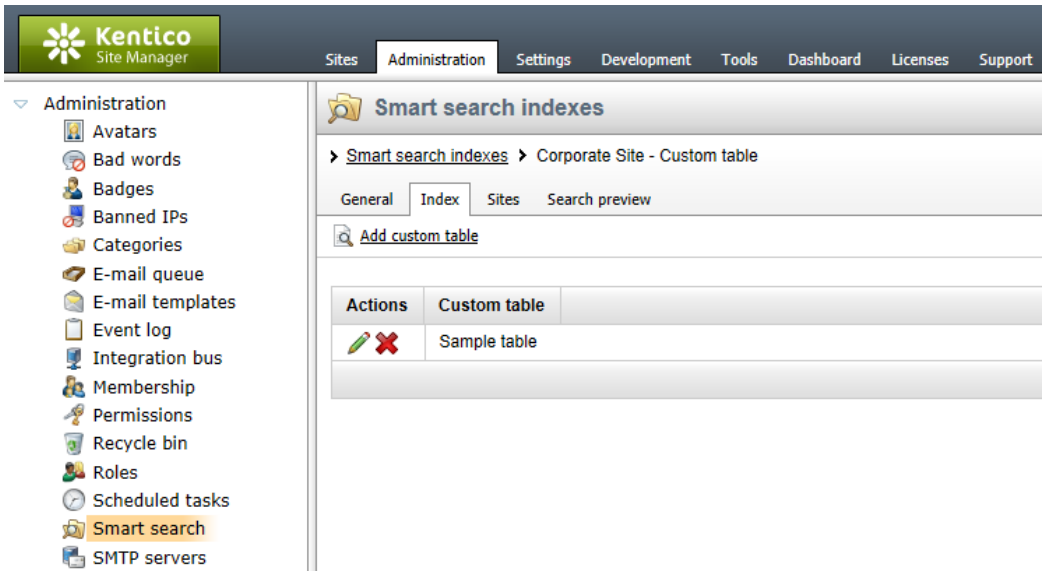
Adding excluded forums

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 the **Add excluded forums** link, while the procedure is the same as when adding allowed forums.

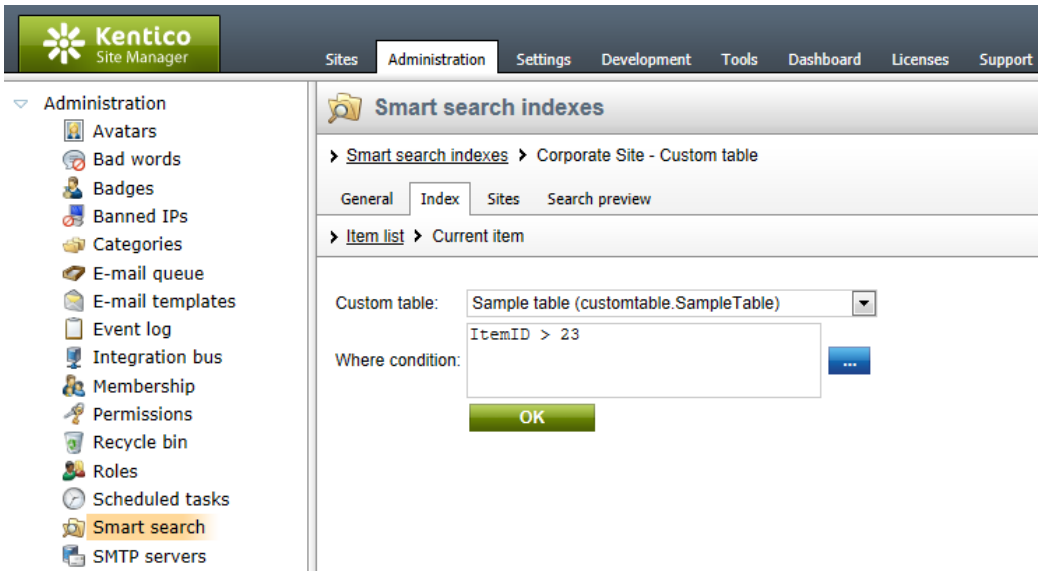
8.41.4.4 Defining custom tables index content

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** link. 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.

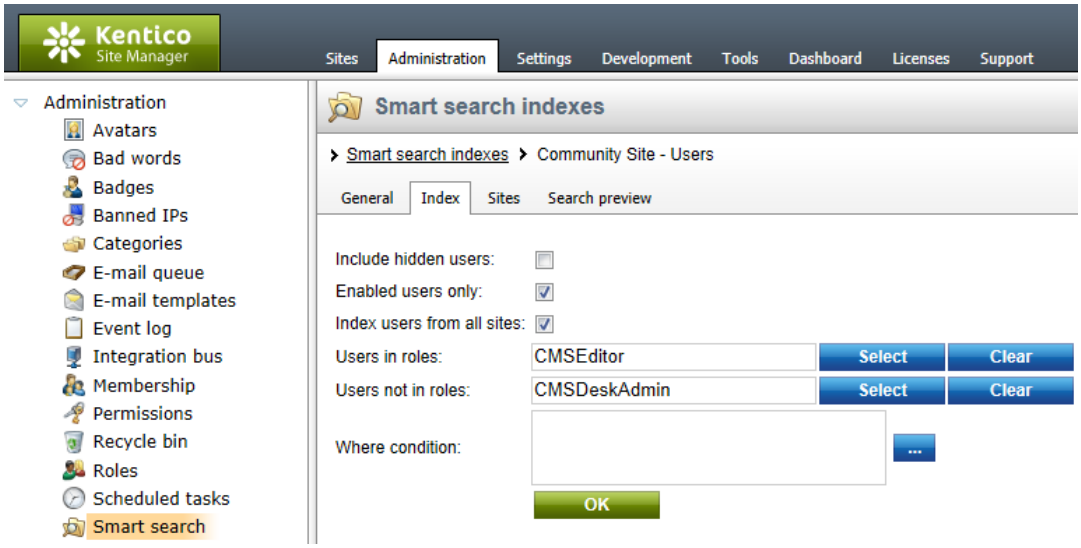


8.41.4.5 Defining user index content

When editing (✎) a user index on the **Index** tab in **Site Manager -> Administration -> Smart search**, you can limit which users will be indexed. The following limitations can be set:

- **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.



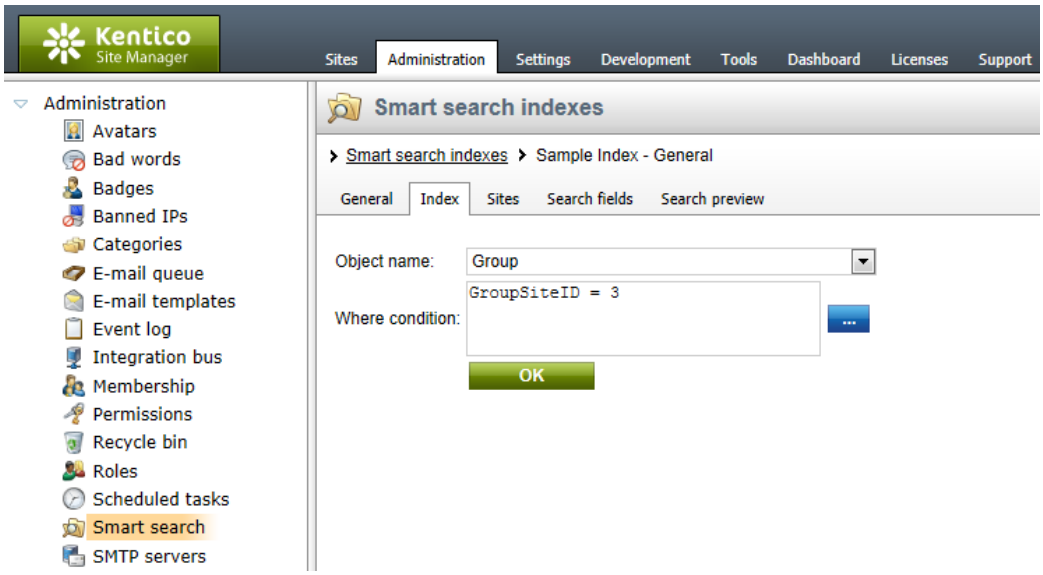
Please note: you can define which fields will be indexed in **Site Manager -> Development -> System tables ->** edit (✎) the **CMS_User** system table **-> Search fields**.

8.41.4.6 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.



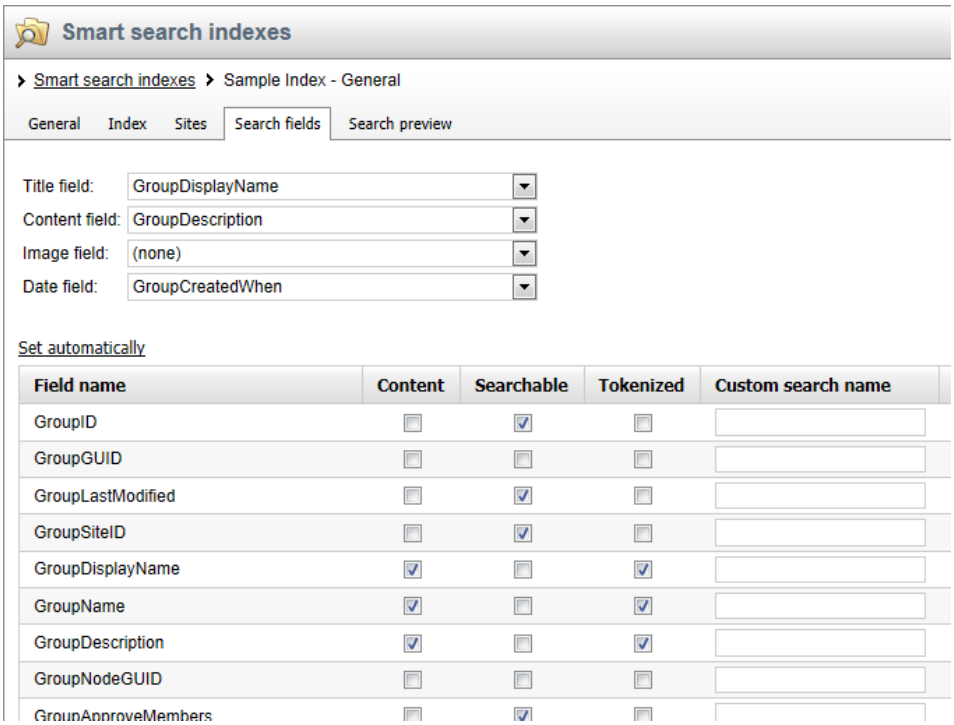
Once the object type is selected, it is necessary to configure which fields should be included in the index on the **Search fields** tab. The settings here affect how objects can be searched for.

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.

8.41.4.7 Defining custom index content

Custom indexes are implemented by writing code, so the content indexed by them is not directly defined on the **Index** tab. All that needs to be done in the administration interface is specify the names of the assembly and class that contain the custom index code.

Each custom index must be defined by a class that implements the **CMS.Siteprovider.ICustomSearchIndex** interface.

To integrate this type of class into the application, you can add a new assembly (Class library) to your project and include it there. When using this approach, it is necessary to add the appropriate references to both the new 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*).

Alternatively, you can define the custom index in *App_Code* without the need to compile an assembly, as described below.

Example:

The following example shows how you can create a custom index that searches the content of text files:

1. Open your web project, expand the **App_Code** folder (or **Old_App_Code** if you installed Kentico CMS as a web application), navigate to the **Samples\Classes** folder and edit the **MyIndex.cs** file. This file already contains a sample class that follows the basic structure required for a custom search index:

[C#]

```
using CMS.EventLog;
using CMS.SiteProvider;
using CMS.SettingsProvider;
using CMS.GlobalHelper;
using CMS.IO;

using Lucene.Net.Index;
using Lucene.Net.Documents;

namespace Custom
{
    /// <summary>
    /// Sample custom search index class.
    /// In this example, all text files from a specified directory are indexed.
    /// </summary>
    public class MyIndex : ICustomSearchIndex
    {
        ...
    }
}
```

Add any of the references listed above that are not already included in the file's code.

2. The most important part of an index's code that must always be included is its **Rebuild** method. It is used to fill the index with data, which determines what kind of searches will be possible. The method is called when the index is created and on each subsequent rebuild.

Place the following code into the method:

[C#]

```
/// <summary>
/// Implementation of the rebuild method.
/// </summary>
/// <param name="srchInfo">Search index info</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 a title 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("CustomIndex", "Rebuild", ex);
            }
            // Always close the index writer.
            finally
            {
                iw.Close();
            }
        }
    }
}
```

The exact code used to build a search index depends on its specific purpose, but the sample above shows the general principles that apply to all indexes. You will first need to create an **IndexWriter** object, then define appropriate search **Documents** (*Lucene.Net.Documents.Document* objects) and their fields. Once the documents are added to the index writer, you can optimize and close it.

The **SearchIndexInfo** parameter of the method can be used to access the data fields of the corresponding search index object. As you can see in the code above, the content of the **Index data**

field is loaded and used 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 can easily modify the behaviour of the index directly from the administration interface without having to edit its code.

Save the changes made to the **MyIndex.cs** file.

3. Next, it is necessary to ensure that the **MyIndex** class is loaded when the corresponding search index is created or rebuilt. Expand the **Samples\Modules** folder and open the **SampleClassLoaderModule.cs** file, which demonstrates how this can be done. You do not have to make any modifications for the purposes of this example, since the sample class loader handles the *Custom.MyIndex* class by default.

An object of the appropriate class is assigned by the **ClassHelper_OnGetCustomClass** event handler:

[C#]

```
/// <summary>
/// Gets a custom class object based on the given parameters.
/// </summary>
private void ClassHelper_OnGetCustomClass(object sender, ClassEventArgs e)
{
    if (e.Object == null)
    {
        // Passes on an object of the appropriate custom class.
        switch (e.ClassName)
        {
            ...

            // Gets an object of the MyIndex class implementing the
ICustomSearchIndex interface.
            case "Custom.MyIndex":
                e.Object = new Custom.MyIndex();
                break;

            ...

        }
    }
}
```

In the case of search indexes, the value of the **ClassName** property of the handler's *ClassEventArgs* parameter will match the **Class name** specified for the given index on its **Index** tab (*Custom.MyTask* in this example). When this value is identified, an instance of the appropriate class is created and passed on.

4. **Build** the project if you installed it as a web application.

5. Now you need to create a folder for the text files that this index will search. This example will use *C:\SearchExample*. Either create or copy some existing text file into this folder.

6. Open Kentico CMS, go to **Site Manager -> Administration -> Smart Search** and click **New Index**. Fill in the following properties:

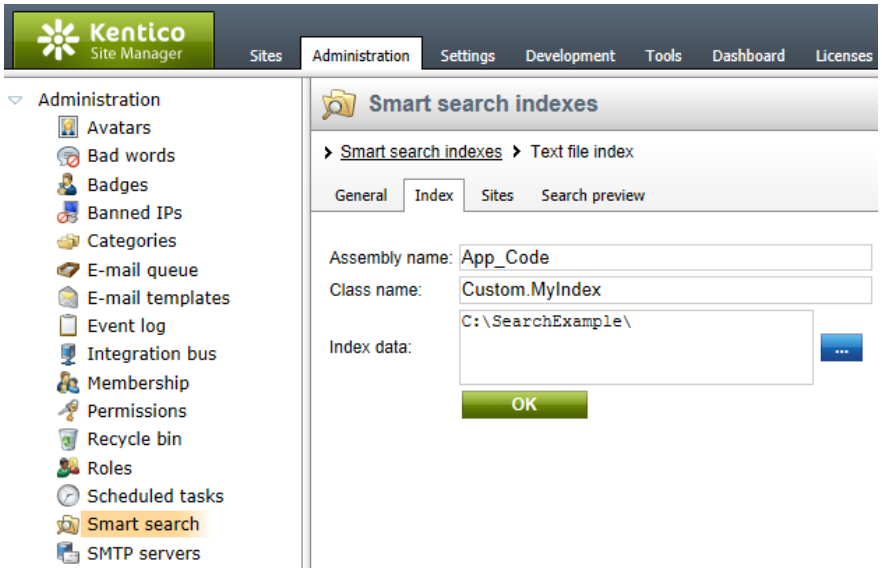
- **Display name**: Text file index

- **Code name**: TextFileIndex
- **Index type**: Custom Index

Click **OK**.

7. Now switch to the **Index** tab and enter the following:

- **Assembly name**: App_Code
- **Class name**: Custom.MyIndex
- **Index data**: C:\SearchExample\



These are the names of the assembly and class where the custom index is implemented. Click **OK**.

8. Finally, go to the **General** tab and **Rebuild** the index. The index should now be fully functional. If you wish, you can test the index by switching to the **Search preview** tab and trying to search for any words from the text files you created in the *SearchExample* folder.

8.41.4.8 Using a custom analyzer

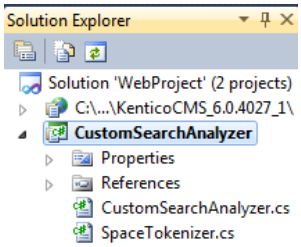
In case the selection of built-in indexing analyzers is insufficient, a custom-written or third-party analyzer may be specified for a search index. This gives you the option of performing tokenization according to your particular requirements.

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.

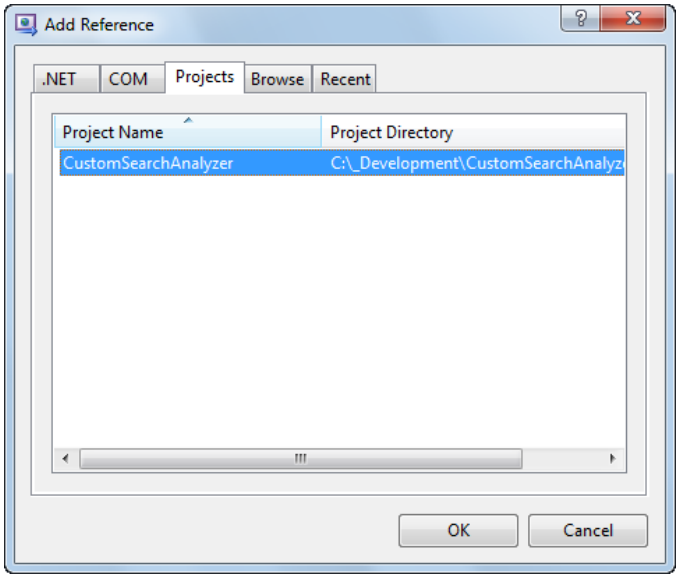
The following example demonstrates how a custom analyzer can be assigned to a smart search index. This sample analyzer divides text into tokens only at space characters and is purely for demonstrative purposes.

1. Copy the **CustomSearchAnalyzer** project from your Kentico CMS installation directory (typically *C:\Program Files\KenticoCMS\<version>\CodeSamples\CustomSearchAnalyzer*) to some 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:



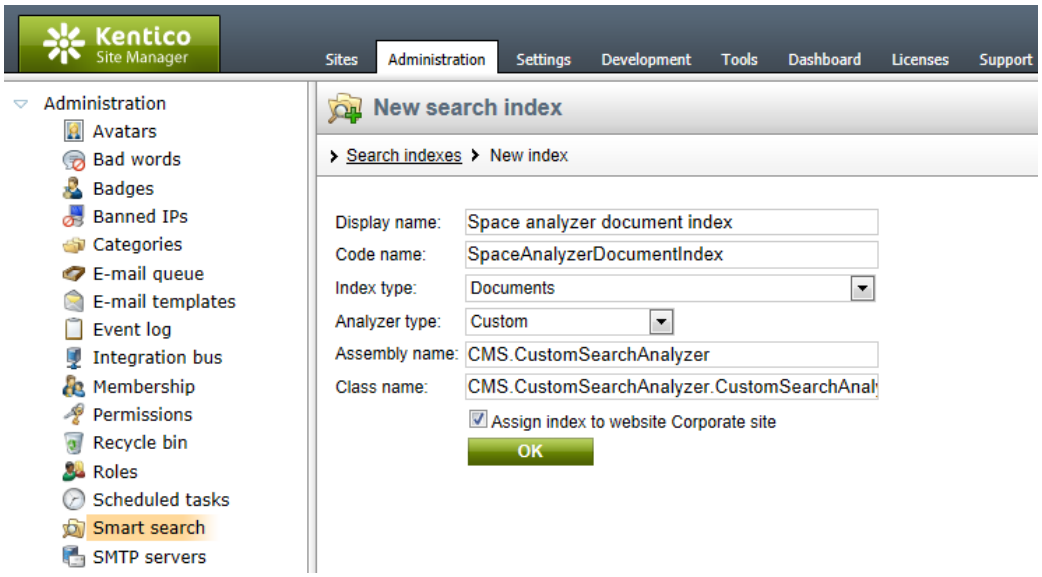
3. Right-click the CMS website object (or the **CMSApp** project if your installation is a web application) and choose **Add Reference**. Switch to the **Projects** tab, select the **CustomSearchAnalyzer** project and click **OK** to add the project reference.



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
- **Code name**: SpaceAnalyzerDocumentIndex
- **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 **OK**.

6. Next, switch to the **Index** tab, click **Add allowed content**, enter */%* into the **Path** field to index all documents and click **OK**. Switch over to the **Cultures** tab and add the culture(s) used by your website.

7. 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 step 3 of the Defining custom index content topic. In this case, the *Assembly* and *Class name* set for the analyzer will have to be changed accordingly.

8.41.5 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

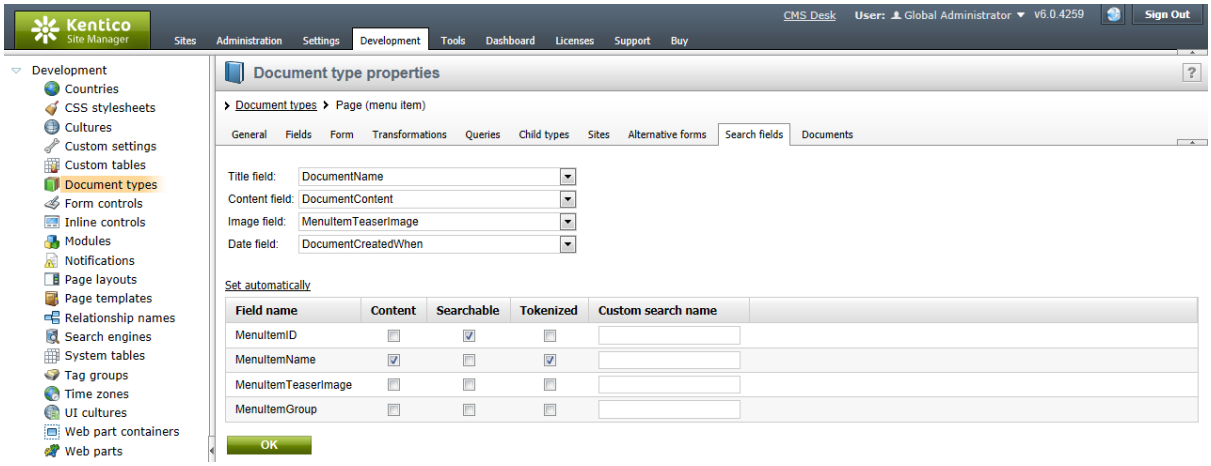
Settings for document types, custom tables and users are almost identical. You can find them on the **Search fields** tab in the editing interfaces at **Site Manager -> Development -> Document types** / **Custom tables** / **System tables -> Edit (🖉) User**.

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.
- **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.



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

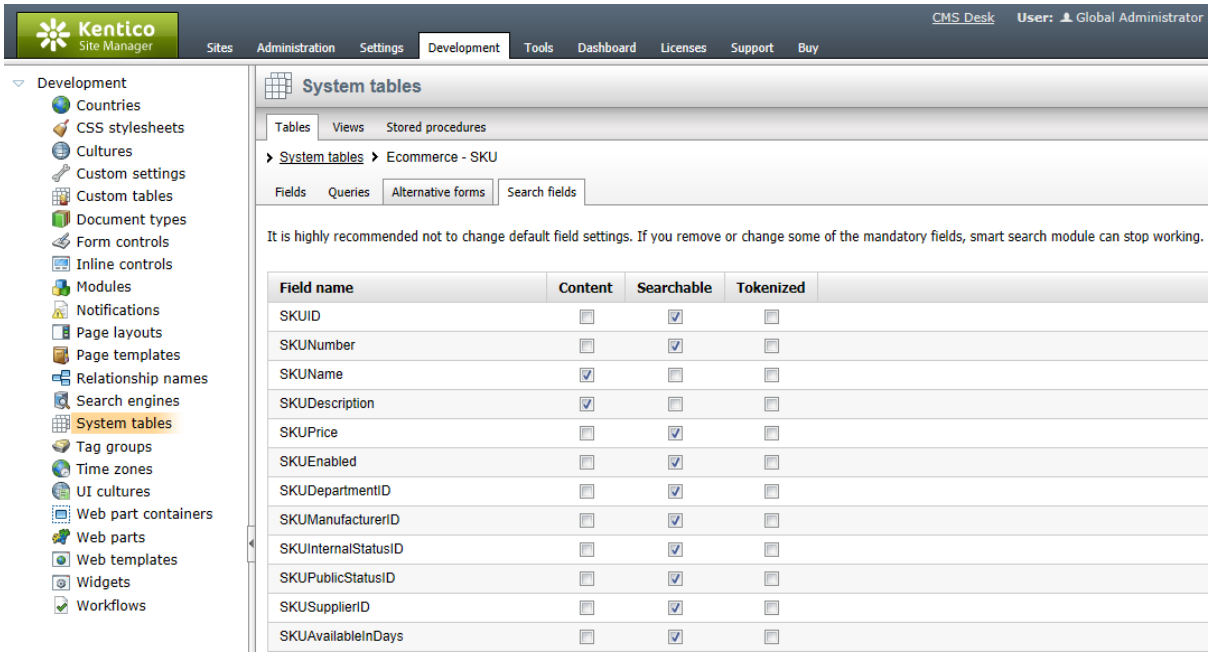
Similar settings are available for **E-commerce SKUs**. The settings can be done in **Site Manager -> Development -> System tables -> Edit (🖉) Ecommerce - SKU -> Search fields** tab. In this case,

only the following settings 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.



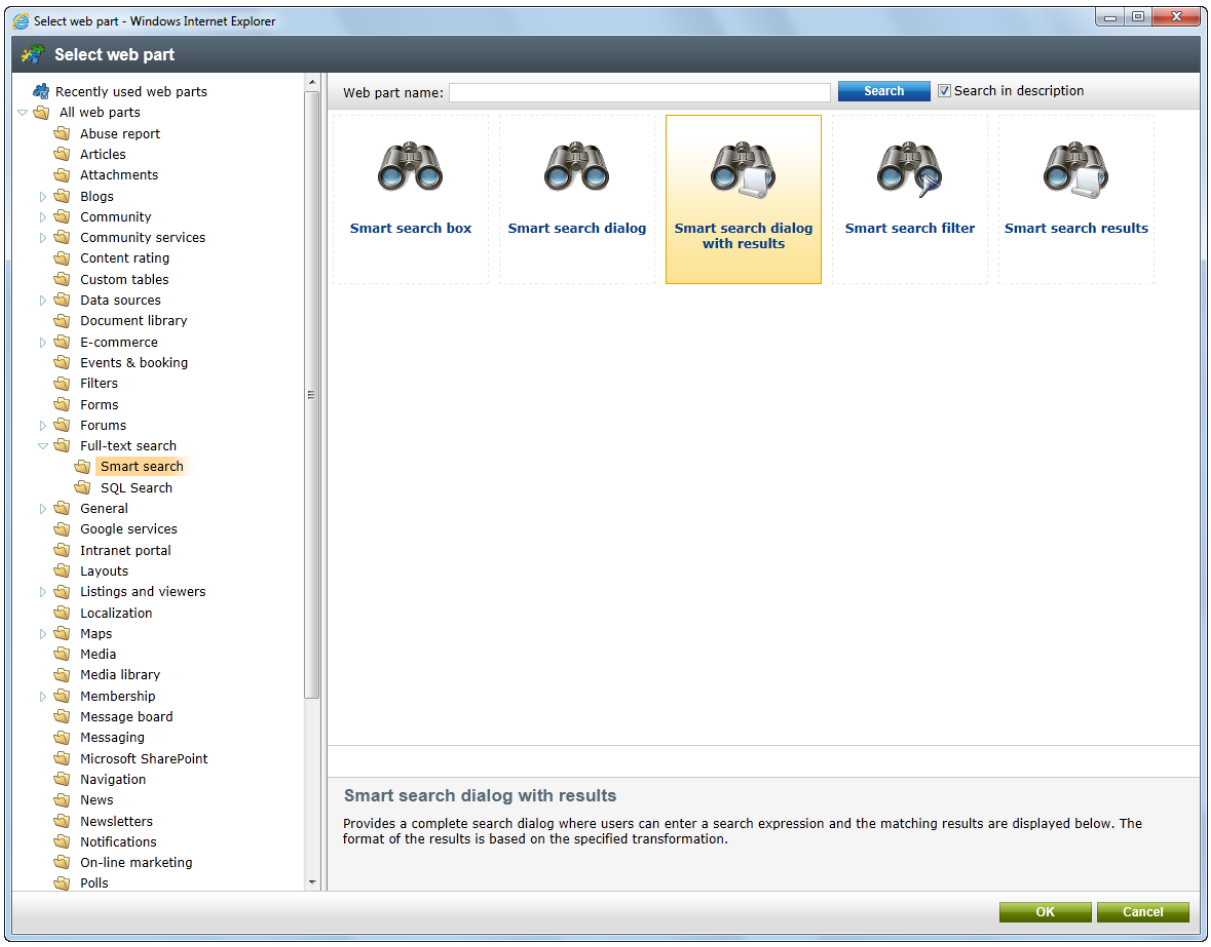
General objects

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).

8.41.6 Available web parts

The Smart search module comes with the web parts mentioned in the text below. Only the most important web part properties are mentioned here. For a complete list and explanations of web part 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.

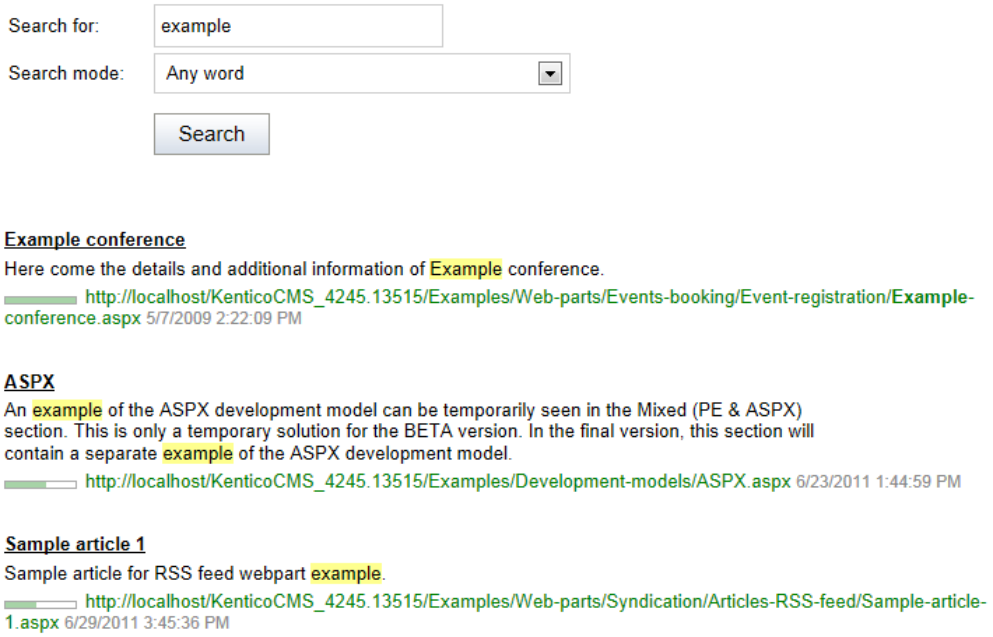
Smart search web parts are located in the **Full-text search** web part category.



The following web parts come with the Smart search module:

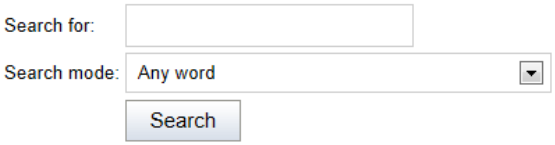
Smart search dialog with results

This is the all-in-one web part for both searching and displaying search results.



Smart search dialog

This web part needs to be placed on a page together with the **Smart search results** web part. The functionality of the two web parts will be identical to Smart search dialog with results, with the difference that these two web parts can be placed at different sections of the page. You can also connect the web part to a Smart search filter.

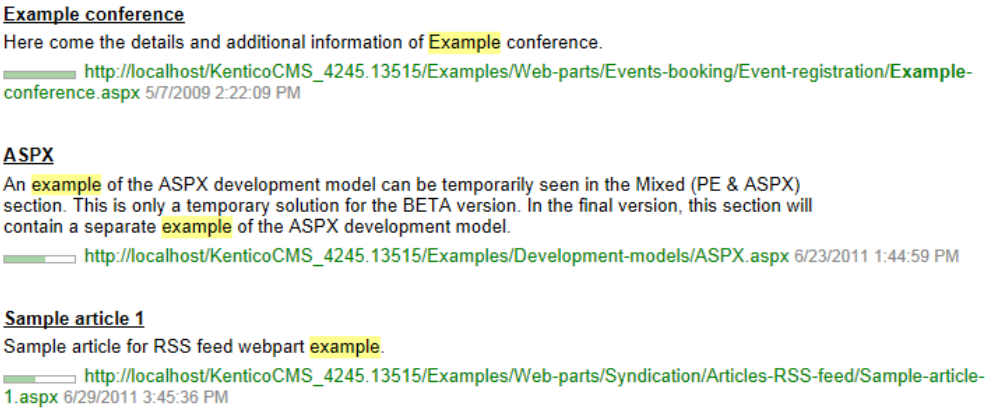


Smart search box

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.

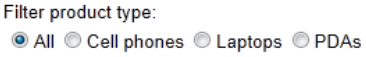
Smart search results

This web part can be 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 or on a different page as one of the two web parts. In case that it is placed on a different page, the page needs to be specified by their **Search results page URL** property.



Smart search filter

This web part can be used to enable users to search through a limited range of objects. You can find a detailed description of how this works in the Using the Smart search filter topic.



The following properties of **Smart search dialog with results** and **Smart search results** web parts are the most important:

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: **CMS.Root.SmartSearchResults** and **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	Using this property, you can limit which documents will be searched using the search syntax (e.g. +articleid:[(int)25 TO (int)150]).
Search results order	Defines the order in which search results are displayed. You can specify one or more document 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*

8.41.7 Using the Smart search filter

The **Smart search filter** web part can be used to allow users to limit the range of objects that will be searched (conditional filter), or define how the search results will be sorted. It is designed to be connected to the **Smart search dialog** or **Search dialog with results**. You can connect more than one filter to these web parts.

You can see an example of how this works on the sample Corporate Site, on the **Examples -> Web parts -> Full-text search -> Smart search filter** page. You can see three **Smart search filters** connected to a **Smart search dialog**, which is connected to **Smart search results**.

The behavior of the smart search filter is mainly defined by the properties explained in the table below. You can find descriptions of all the web part's properties in the Kentico CMS Web Parts reference or after clicking the **Documentation** link at 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.
Values	Using this property, you can specify the possible filtering options that can be selected. Enter one option per line in format: *<index field name>;<value of the field>;<displayed text>* When creating a conditional filter, the logical value of each filtering option must be specified. If the + symbol is added before the option, then only objects whose value in the given field matches the specified value will be included in the search. If the - symbol is added, only results that do **not** match the value will be returned. Please note, 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.cellphone;Cell phones • +_created;[{%CurrentTime|(add)-10080|(tosearchdatetime)%} TO {% CurrentTime|(tosearchdatetime)%}];Past week • ##SCORE##;;Score • documentcreatedwhen;;Creation date 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 field

	configuration of the given object type. Information about the search field configuration of objects can be found in the Settings for particular object types topic.
Query name	Name of the query which can be used instead of the *Values* property to dynamically create the filter options. The query must return three columns, which will be used in the following order: *<index field name>, <value of the field>,<displayed text>* Sample query to load all document types as the filter 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 range of objects that are searched (where condition). If false, the filter determines the order in which search results are displayed (order by condition).

8.41.8 Search syntax

A detailed description of Lucene query parser syntax can be found at http://lucene.apache.org/java/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) will be recognized except for field searching.
- **None** - no query syntax will be recognized. All text entered by users will be processed as a part of the searched for expression.
- **Full** - all search query syntax will be processed, including field searching.

Field search

By default, field searching can only be used to define an additional condition in a search expression. 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.

If you wish to allow direct field searches, you must add the following key into the **/configuration/appSettings** section of your **web.config** file:

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

Once this is done, expressions containing only search syntax can be used. This allows users to find records directly by entering an exact field value:

- *DocumentNodeID:(int)17* - the document with a nodeID equal to 17 will be returned.
- *NewsTitle:"New features"* - the news document titled New features will be returned.

Keep in mind that only fields that are set as **Searchable** in the field configuration of the given object type may be included in field searches. Information about the search field configuration of objects can be found in the Settings for particular object types topic.

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

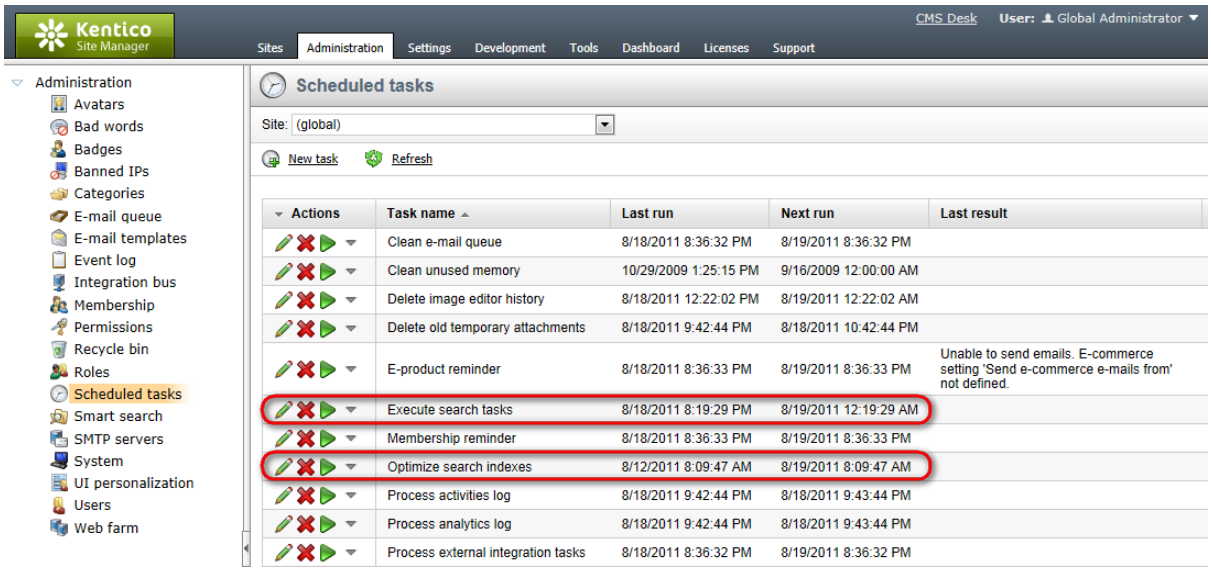
Indexes created by Stop and Simple analyzers cannot be searched using the standard field search format. This is by design, but you can use a workaround with a range query containing identical boundaries:

- newsid:[(int)22 TO (int)22]

8.41.9 Related scheduled tasks

The following two scheduled tasks are related to the Smart search module:

- **Optimize search indexes** - performs index optimization (defragmentation resulting in better performance, particularly in the case of large indexes). By default executed once per week.
- **Execute search tasks** - executes indexing tasks (created and executed automatically when the indexed content changes) that were not completed successfully on their automatic execution. By default performed every 4 hours.



Please refer to the Development -> Scheduler chapter for more information about scheduled tasks.

8.41.10 Searching attachments

Document attachments uploaded to the database can be searched as well. This is done by using the Microsoft SQL Server full-text search. You need to have your SQL server configured properly as described in the Installation and deployment -> Additional configuration tasks -> Configuration of full-text search in files chapter in order for attachment searching to be functional.

Once you have the SQL server configured, you need to enable attachment searching by using the following properties of the **Smart search dialog with results** or **Smart search results** web part:

Property Name	Description
Search in attachments	If enabled, document attachments will be searched.

WHERE condition	WHERE condition used to limit the scope of the attachment search for this web part. In addition to specifying which documents should have their attachments searched, you may 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 used to determine the order of documents retrieved by the attachment search in the results.

When a search is performed and a match is found in the attachment of a document, the given document is added to the returned results. These results are always interlaced with the other results provided by the specified smart search index(es). This is by design and cannot be modified.

Because the attachment search is performed by the SQL server, it is not affected by the settings and restrictions of the used index(es). If you wish to limit the attachment search, it is necessary to 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 would 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 will only return documents if they are directly connected to the matching attachment through one of the following methods:

- The attachment files are added to the document using fields that have their **Attribute type** set to *File* or *Document attachments* in the document type definition.
- The attachments are uploaded in CMS Desk via the **Properties -> Attachments** dialog of the given document.

8.41.11 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 what the field names in the found objects are, 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

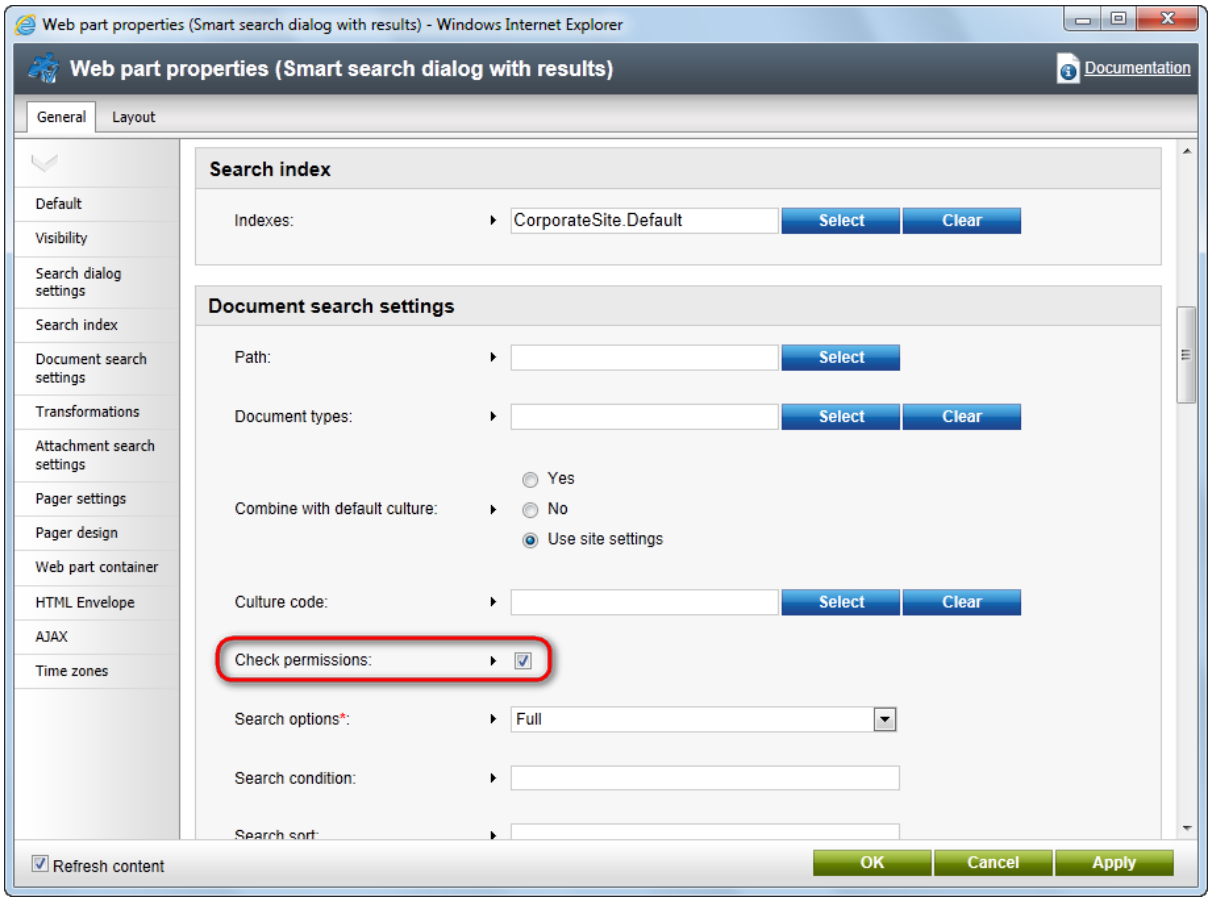
This method does not return valid URLs for search results produced by a general index, since the indexed objects are not documents and there is no default page where the details are displayed.

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.

8.41.12 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.



8.41.13 SQL search overview

The SQL search engine is kept in the system for backward compatibility with older versions and for attachment searching. It uses standard SQL queries to search for the given expression. It runs a separate query for each document type. You can find the search query for a document type at **Site Manager -> Development -> Document types -> ... select document type ... -> Queries -> Edit** ( ) the **searchtree** query.

Common fields (such as document name) are searched using the **cms.root.searchdocuments** query.

Uploaded files are searched using the **cms.root.searchattachments** query. If you want to search uploaded files, you need to configure the system as described in the Installation and deployment -> Additional configuration tasks -> Configuration of full-text search in files topic. In this case, the files are searched using the Microsoft SQL Server Search Engine.

Web parts and controls

The search dialog can be easily integrated into the website using the web parts in the Full-text search category (SQL search dialog, SQL search box, etc.) or using 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 the **Site Manager -> Settings -> System -> Search** dialog. You need to enter the code name, such as cms.article. You can enter multiple document types separated with a semicolon (;).

You can **exclude website sections** from search by setting the **Exclude documents from SQL search** in the **Site Manager -> Settings -> System -> Settings** dialog. You need to enter the alias path of the section that should be excluded. A single document can be excluded using e.g. **/news/news1**, whole website section can be excluded using e.g. **/news/%**. You can enter multiple values separated by semicolons (;).

The **Exclude this document from search property** of individual documents can also be used to exclude them from SQL search. This property can be configured by selecting a document from the content tree in **CMS Desk** and going to **Content -> Edit -> Properties -> General**.

Modifying the search result format

If you only need do modify the search results format, you can do that by modifying the transformation **searchresults** 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 standard search engine, you can develop your own search provider. You can find more details in the API programming and Kentico CMS internals-> Custom providers -> Custom Search Provider topic.

8.41.14 Smart search internals and API

8.41.14.1 Overview

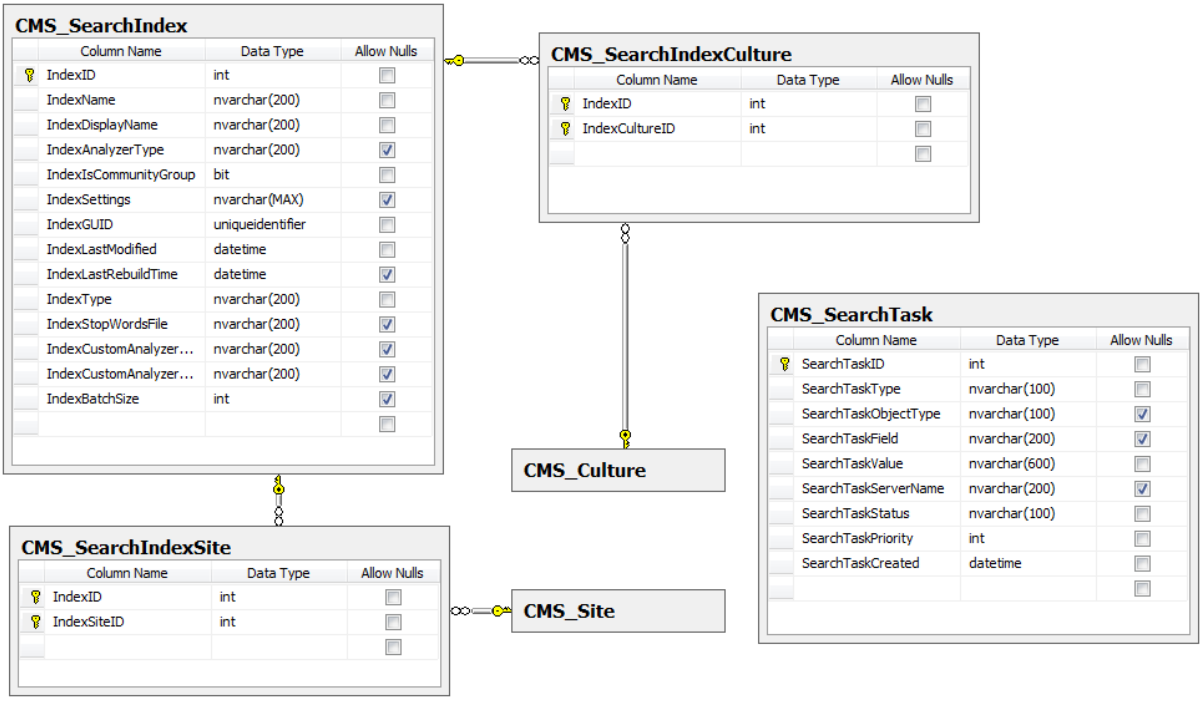
In this chapter, you will learn which database tables and API classes are used in the Smart search module. You will also see the most common API examples.

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.

8.41.14.2 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.



8.41.14.3 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.

8.41.14.4 API examples

8.41.14.4.1 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.TreeEngine;
```

8.41.14.4.2 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);
}
```

8.41.14.4.3 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;
}
```

8.41.14.4.4 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;
}
```

8.41.14.4.5 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 CMS.TreeEngine.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;
}
```

8.42 Syndication (RSS, Atom, XML)

8.42.1 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.

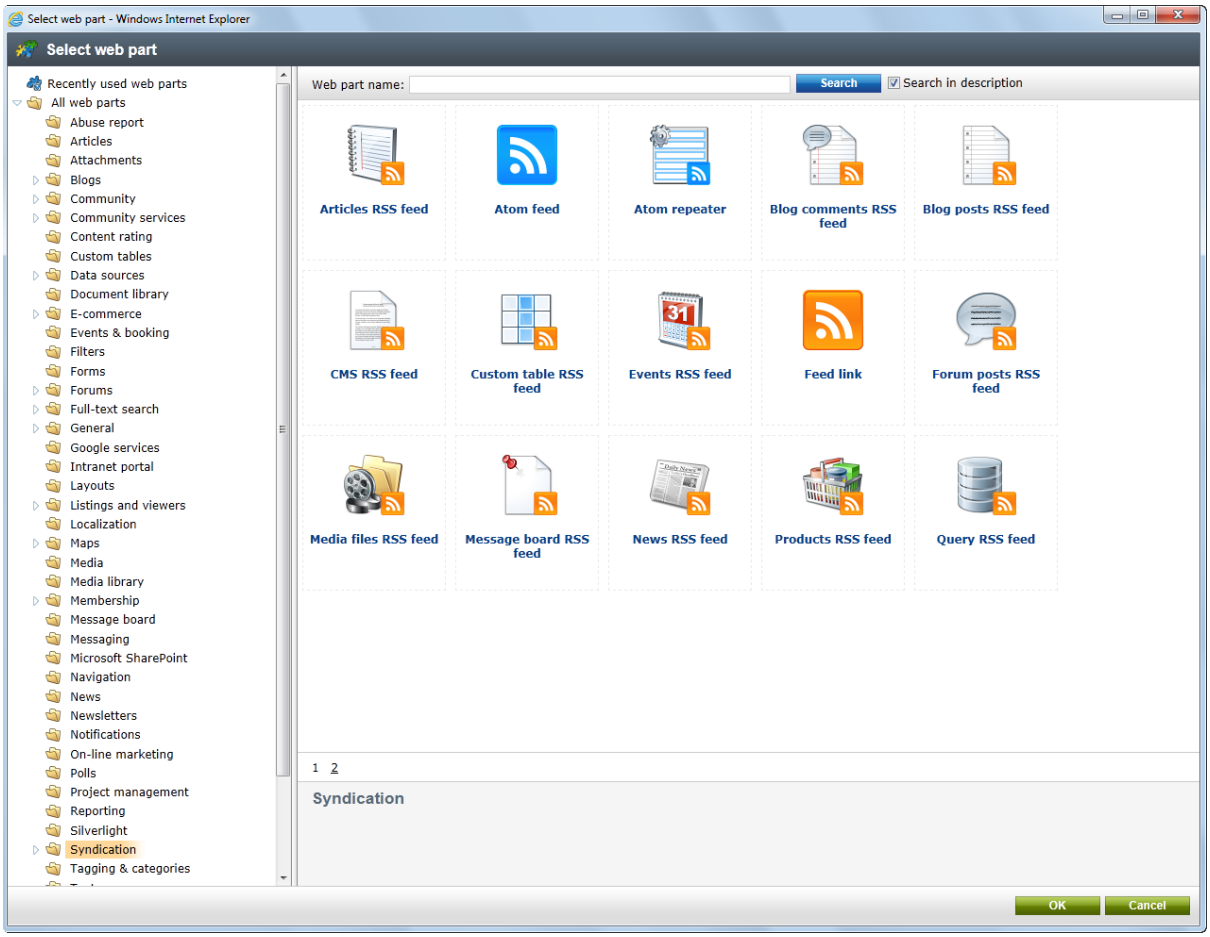
Detailed overview of the web parts can be found in the Syndication web parts and widgets topic. Use of transformations is essential with these web parts. Information related to use of transformations in syndication web parts can be found in the Syndication transformations topic.

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.

8.42.2 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

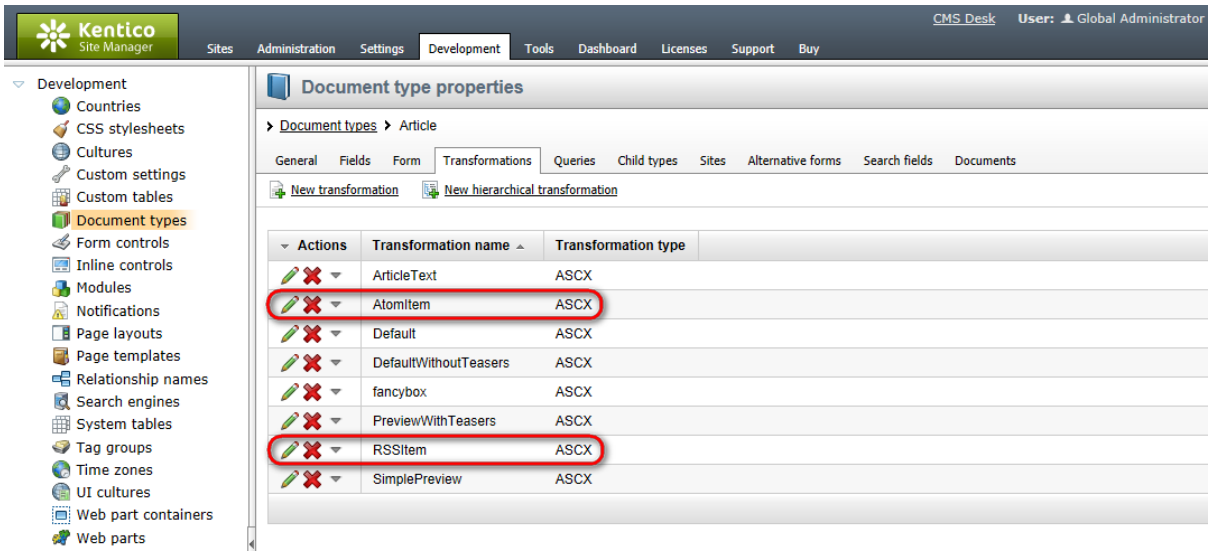
All specific properties of these web parts or widgets are explained in Kentico CMS Web Parts Reference or after clicking the **Documentation** ( ) link in the top right corner of the *Web part (widget) properties* dialog.

8.42.3 Syndication transformations

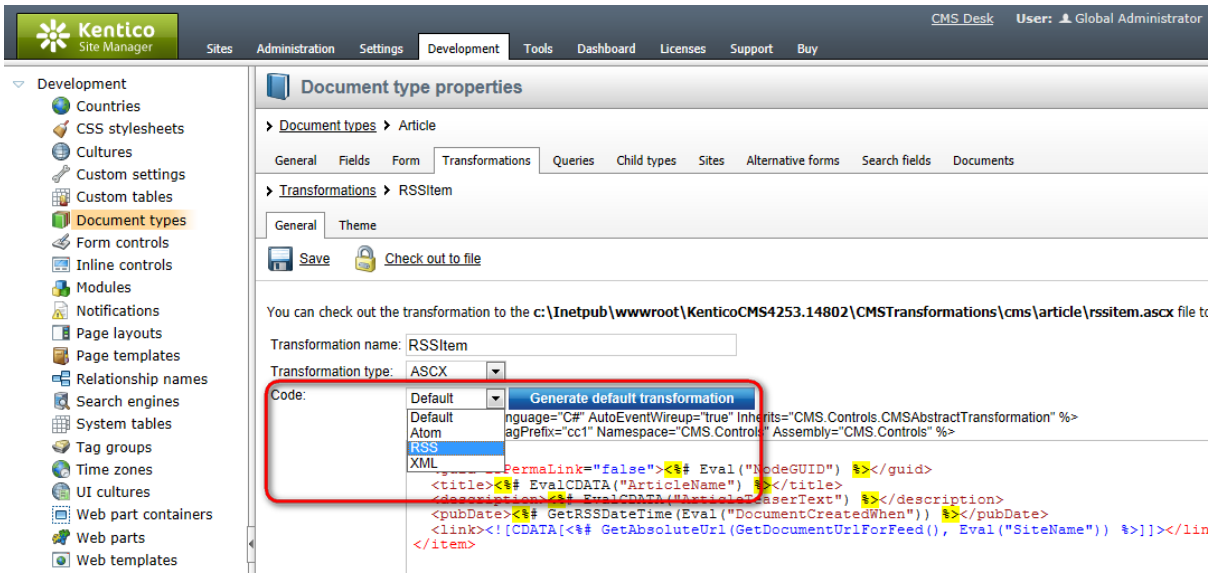
Use of transformations is of the highest importance when creating feeds, as all syndication web parts render the feeds using them. They need to be set in the **Transformation** property of each feed web part.

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.

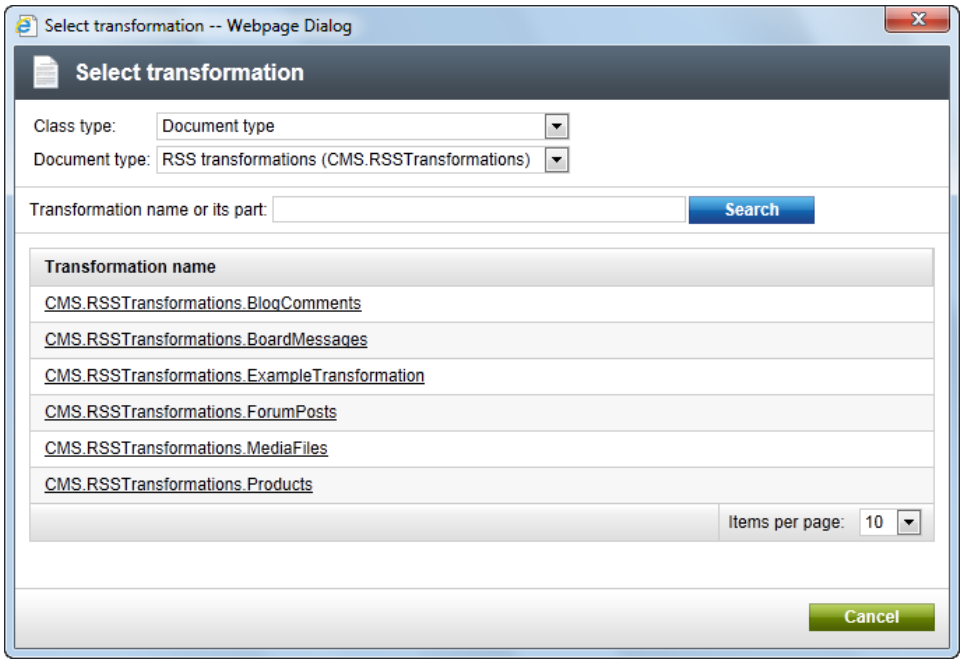


The default RSS, Atom and XML transformations can also be generated when editing or creating transformations of a document type or a custom table. To do this, you only need to select the required format from the **Code** drop-down list and click the **Generate default transformation** button.



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>

...
```

8.42.4 Usage examples

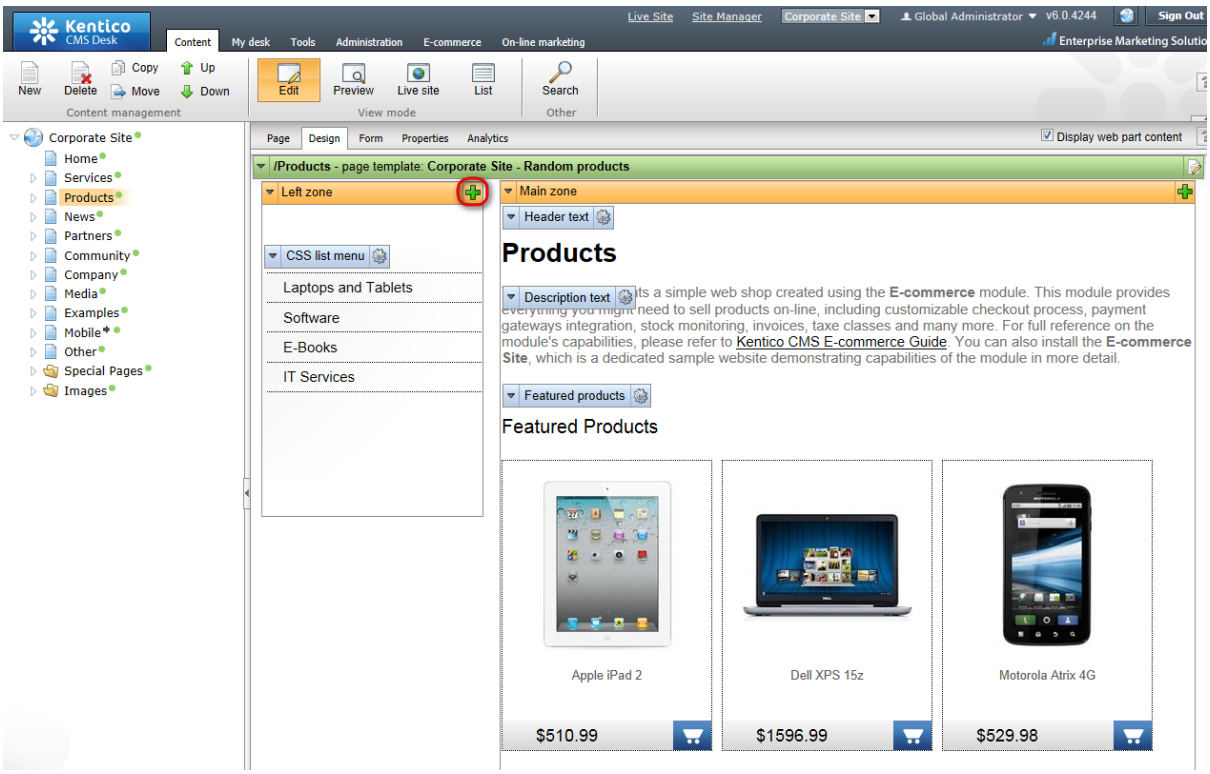
8.42.4.1 CMS RSS feed

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.

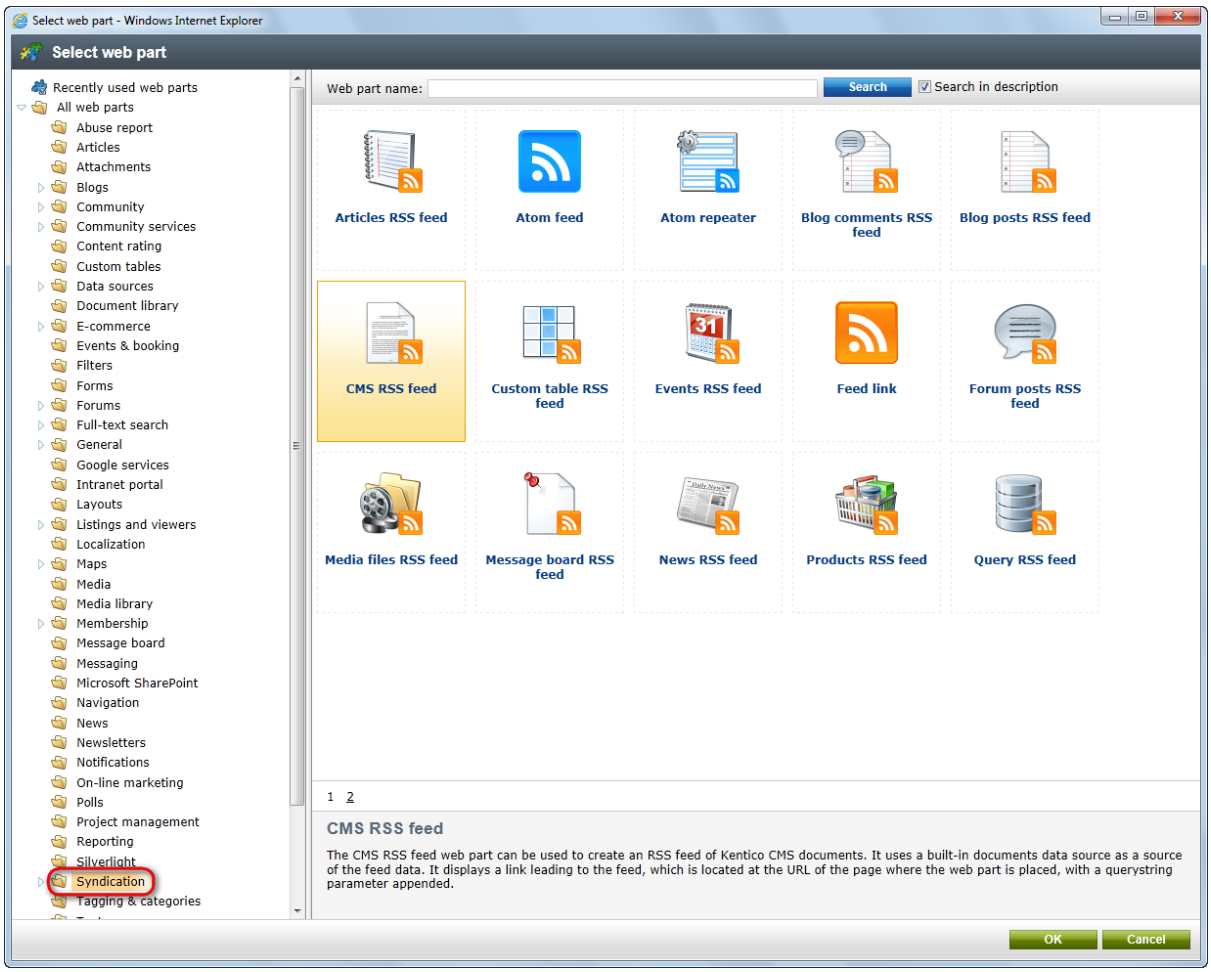
Apart from this universal web part, there are other web parts in the **Syndication** category which are pre-configured for one particular document type. See the listing in the Syndication web parts and widgets chapter.

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 *zoneLeft* web part zone.



2. In the **Select web part** dialog, select the **Syndication** category. Choose the **CMS RSS feed** web part and click **OK**.

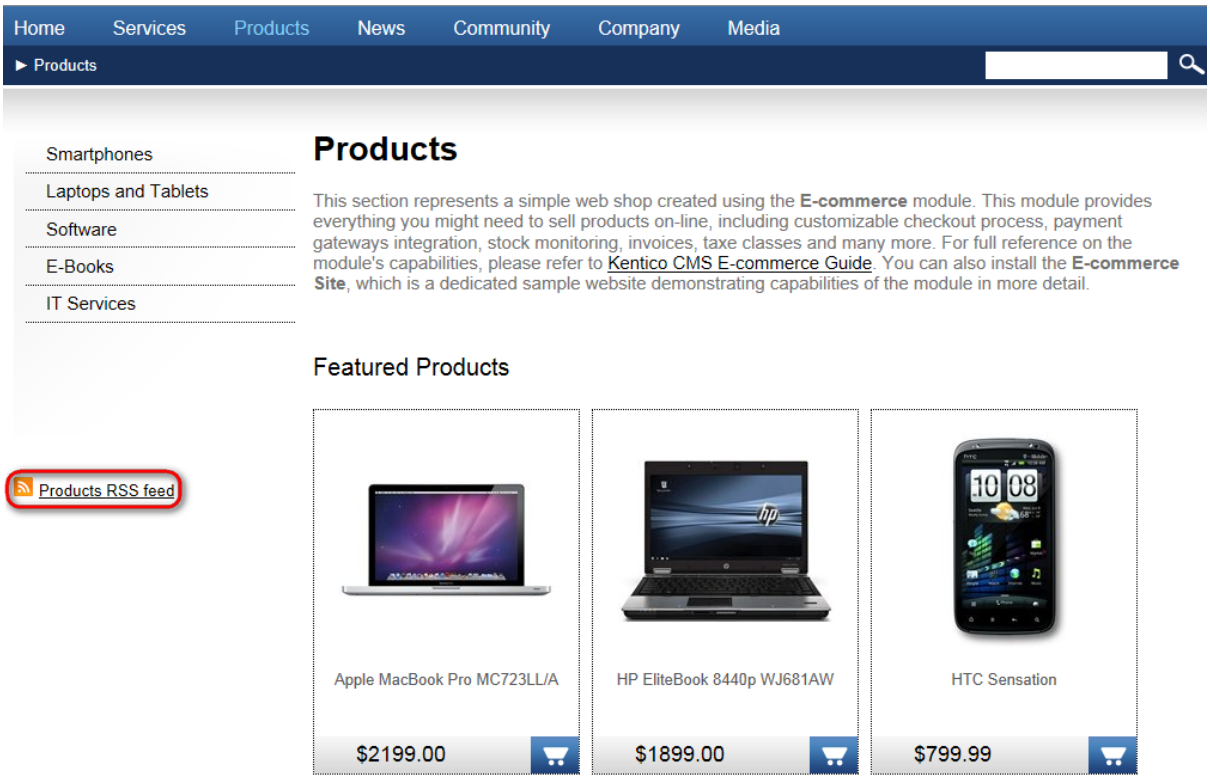


3. The **Web part properties** dialog opens. To achieve the required behavior, use the following configuration:

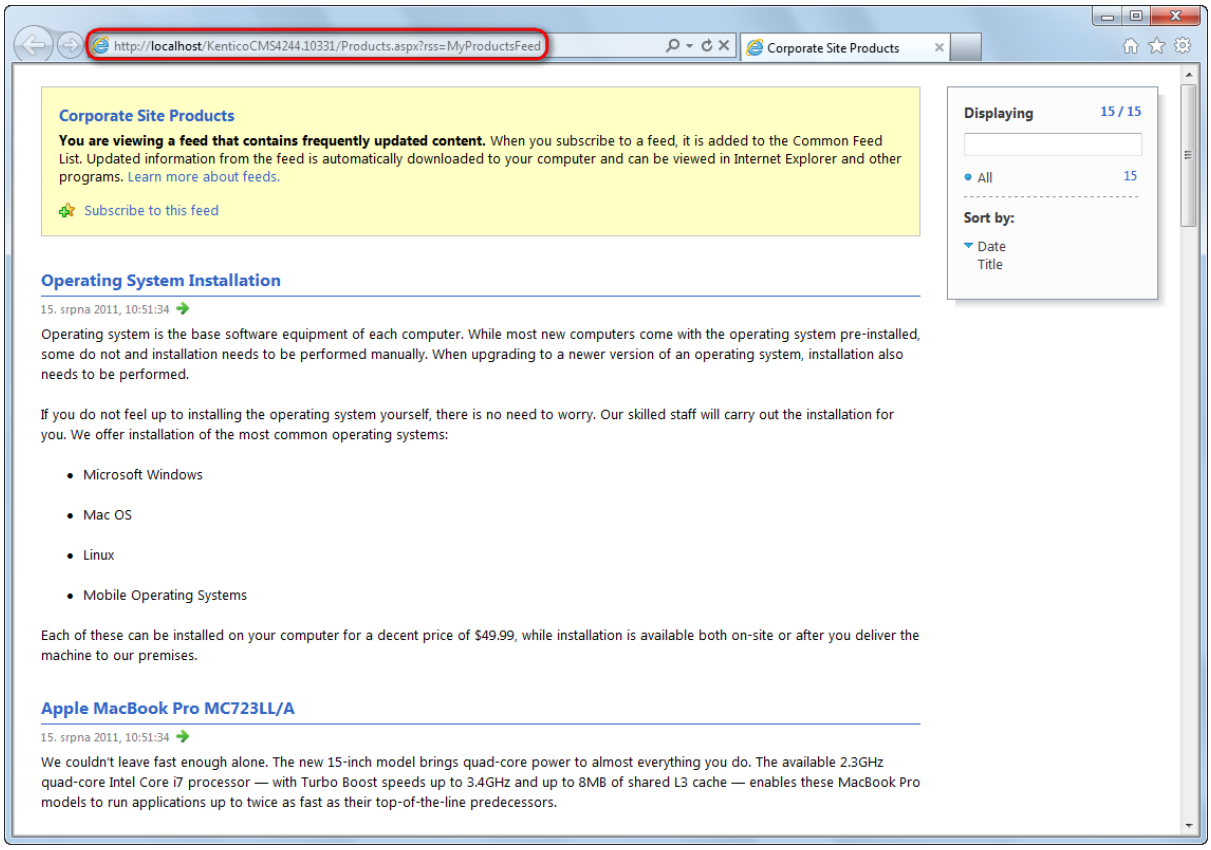
- **Link text**: *Products RSS feed*
- **Feed name**: *MyProductsFeed*
- **Feed title**: *Corporate Site Products*
- **Feed description**: *This is a sample RSS feed of products on the Corporate Site.*
- **Path**: */Products/%/%*
- **Transformation name**: *CMS.RSSTransformations.Products*

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.



8.42.4.2 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.

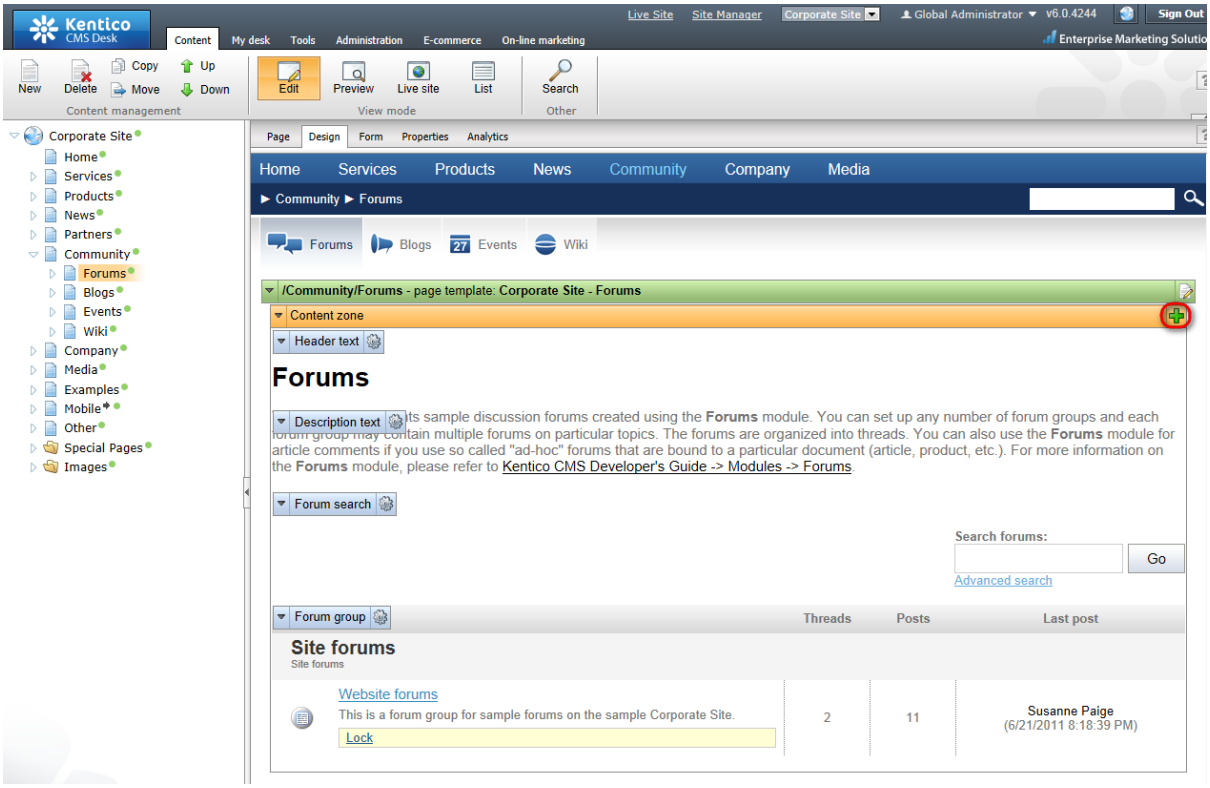
In the previous example, we have demonstrated how to use the *CMS RSS Feed* web part to create an RSS feed of **documents** stored in the content tree. The approach that we will explain now can also be used for that purpose, but you will typically use it to create feeds from Kentico CMS **objects** (forum posts, board comments, media files, ...).

In the following example, you will see how to create an RSS feed of forum posts in the site forums on the sample Corporate site. We will 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 handle the provided data by 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 listing here).

> The following example uses two separate web parts for educational purposes only.

1. Sign in to CMS Desk and select the **/Forums** page from the content tree. Switch to the **Design** tab and click the **Add web part** (✚) icon of the *zoneLeft* web part zone.

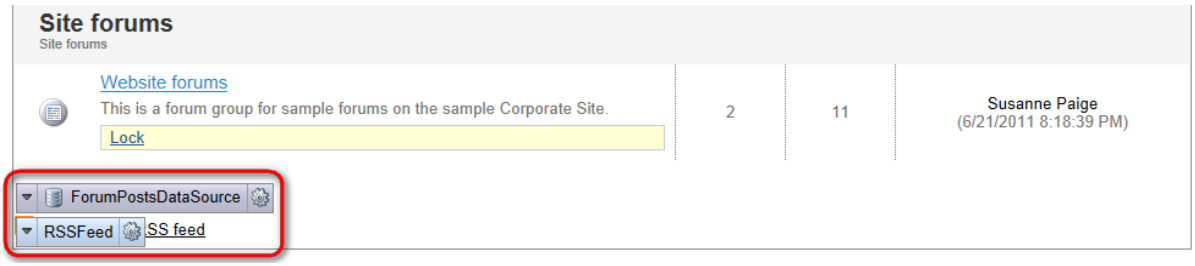


2. Select the **Data sources/Forum posts data source** web part and click **OK**. Leave all its properties at the default values, which will result in the data source providing all forum posts from all forum groups on the site. Click **OK**. The data source will be added to the bottom of *zoneLeft*.

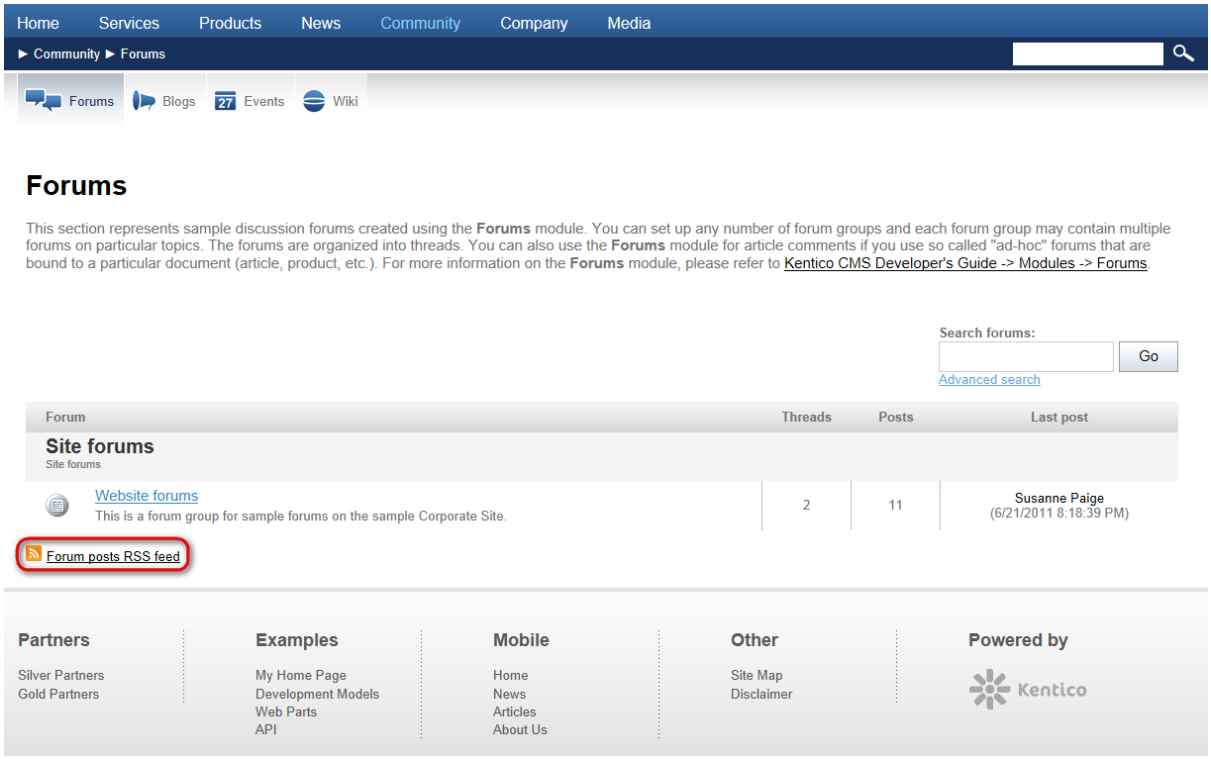
3. Click the **Add web part** (✚) icon again and select the **Syndication/RSS feed** web part. Click **OK** and enter the following in the *Web part properties* dialog:

- **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*

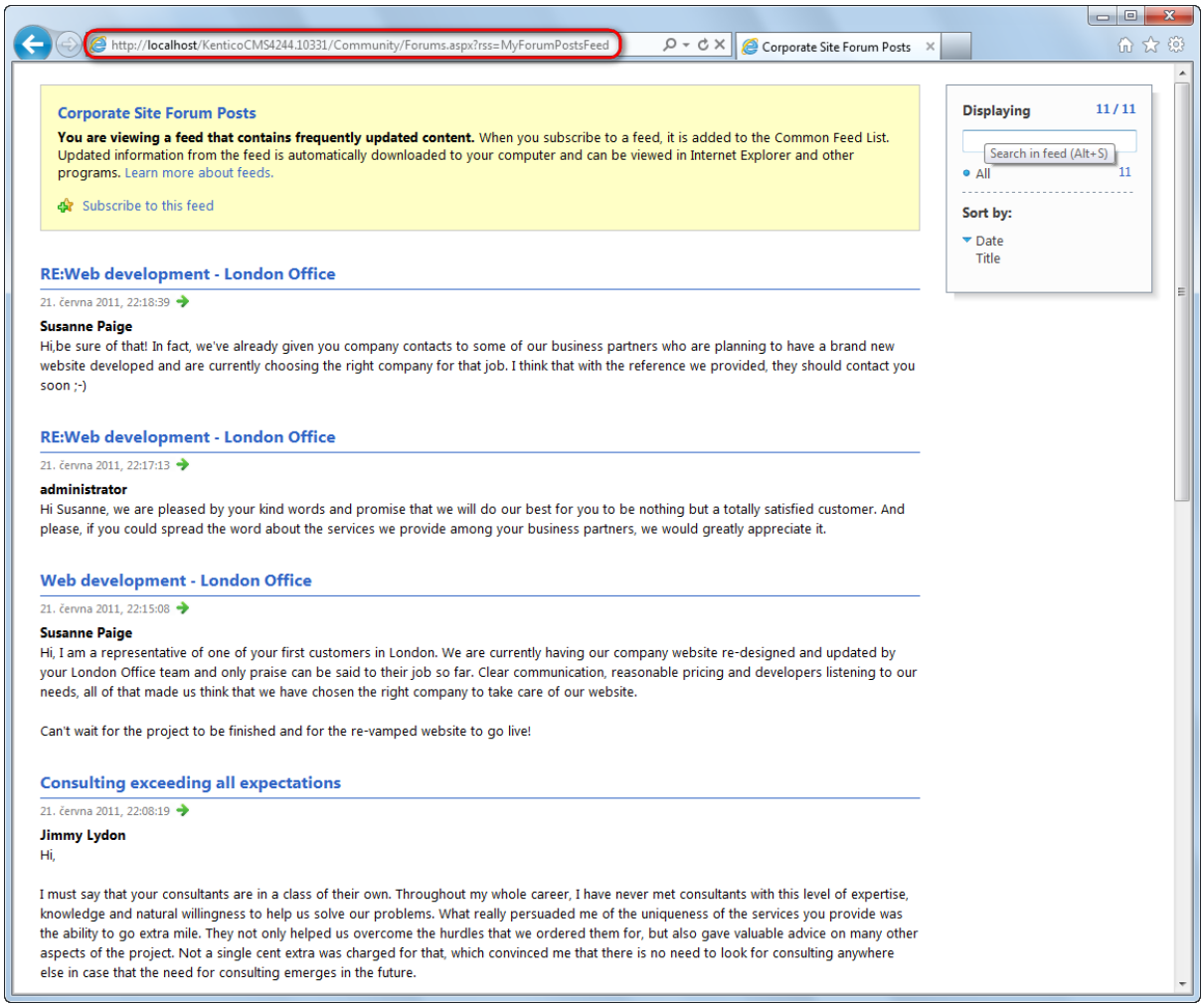
Leave defaults for the rest of the properties and click **OK**. The two web parts should now be at the bottom of the web part zone and fully configured for the required behavior.



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.



8.42.4.3 RSS repeater + Data source

The **RSS repeater** web part can be used to transform a page into an RSS feed. This means that when a page containing the web part is accessed, its response type is changed to *application/xml* and the content of the feed is rendered. This is useful typically if you want your feed to have a dedicated URL (i. e. not a URL of a page with 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.

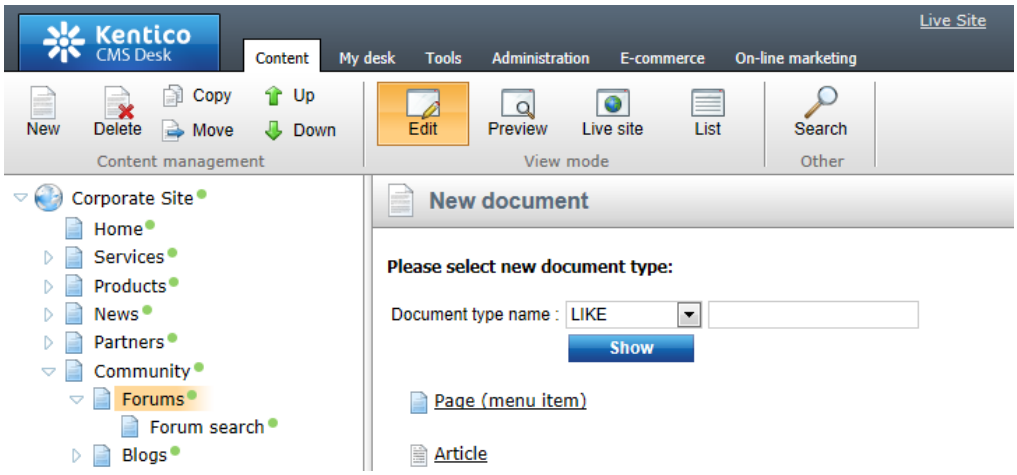
In the following example, we will create an RSS feed with forum posts, the same as in the previous example. But this time, we want the feed URL to be *<application path>/Forums/RSS.aspx*. Therefore, we will create a **dedicated page** with this URL which will contain the **RSS repeater** web part and which will be linked from the **/Forums** page.

> **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

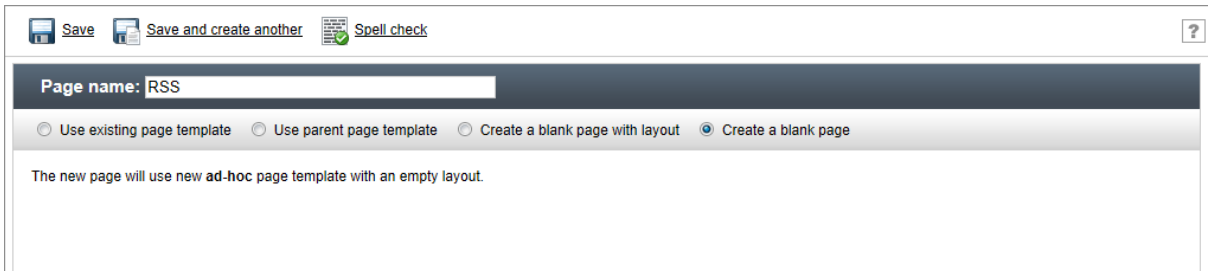
objects (see the listing here).

The following example uses two separate web parts for educational purposes only.

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* for **Page name** and click **Save**.

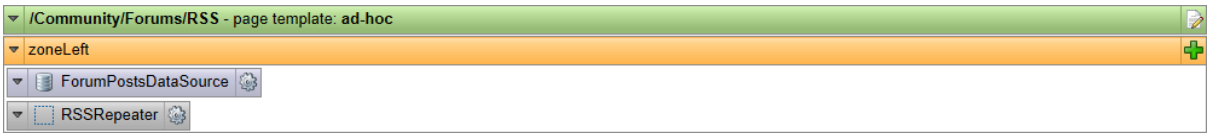


3. The page opens in **Design** mode. Click the **Add web part** (+) icon of the only web part zone on the page. Select the **Data sources/Forum posts data source** web part and click **OK**. Leave all its properties at the default values, which will result in the data source providing all forum posts from all forum groups on the site. Click **OK**.

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**.



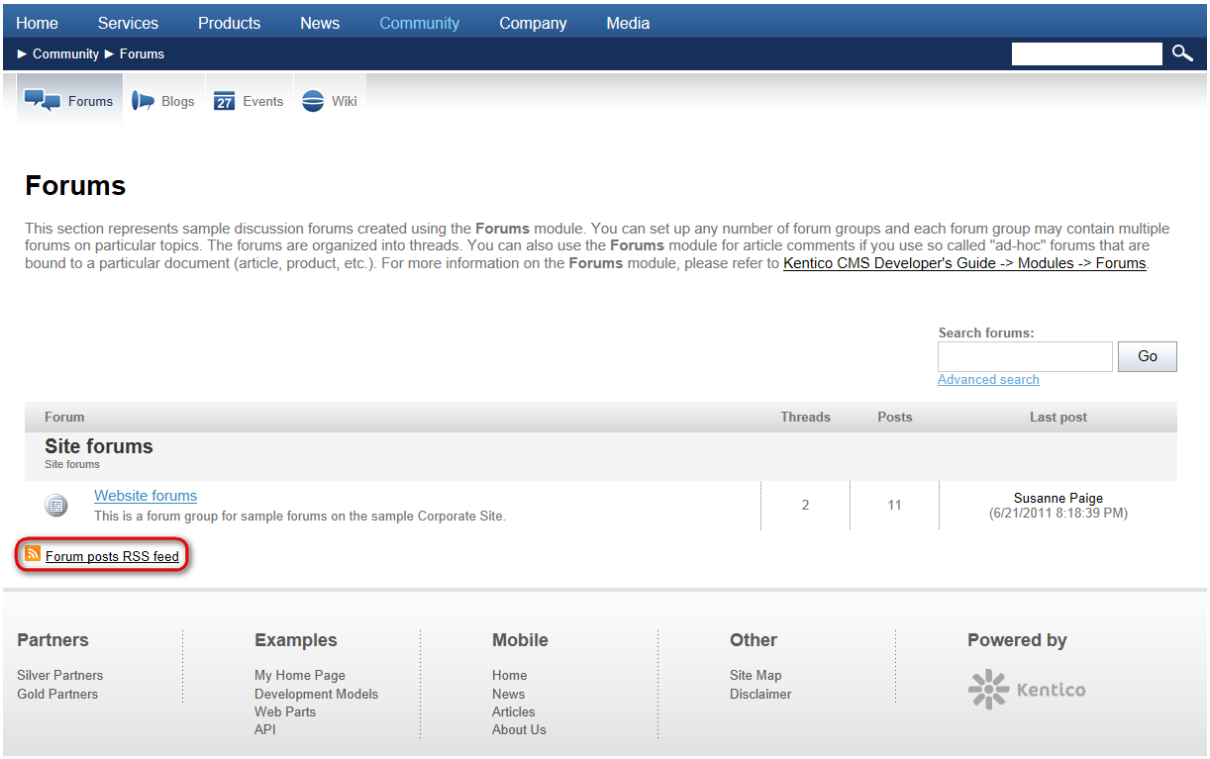
5. Now we have the RSS feed created, but without the knowledge of the page's URL, site visitors have no chance to access it. This is exactly when the **Feed link** web part comes in handy. It is a simple web part which displays the RSS icon with a link leading to a URL specified in its properties. We will add it to the */Forums* page.

Select the **/Forums** page in the content tree, switch to the **Design** mode and click the **Add web part** ( ) icon of the *zoneLeft* web part zone. Select **Syndication/Feed link** and click **OK**. Enter the following properties:

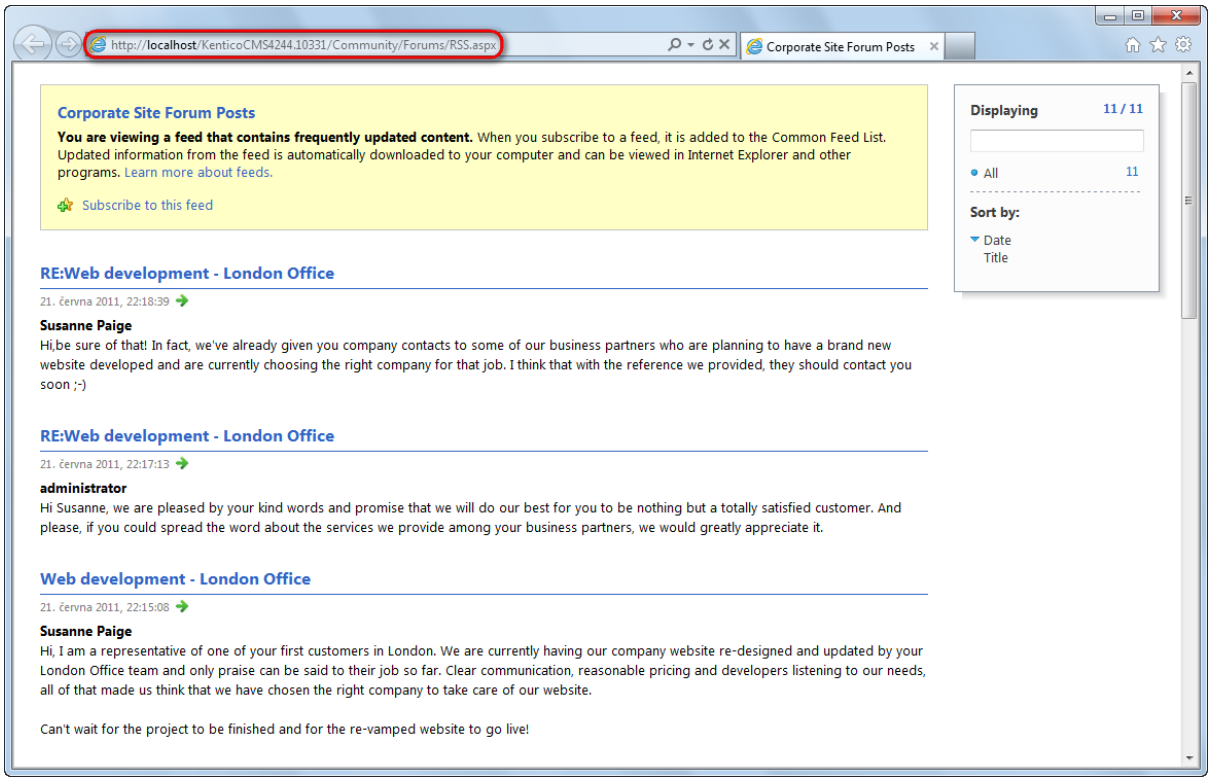
- **Link text**: *Forum posts RSS feed*
- **Feed URL**: *~/Forums/RSS.aspx*
- **Feed title**: *My Forum Posts Feed*

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 as highlighted in the screenshot below.



7. 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. The same URL can be used to access the feed directly both from a browser or a dedicated RSS reader program.

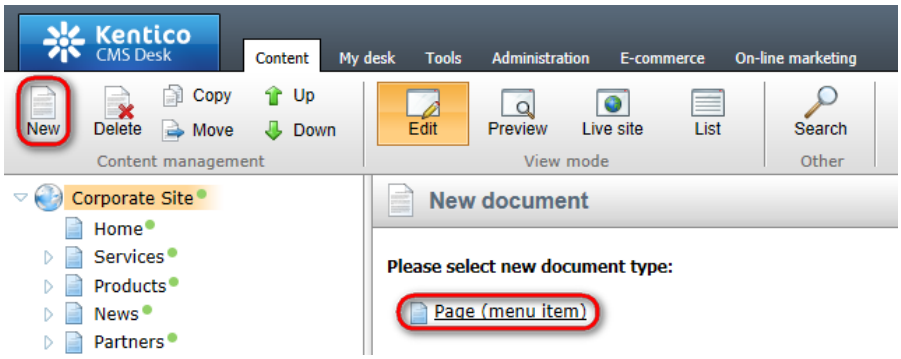


8.42.4.4 External RSS feed

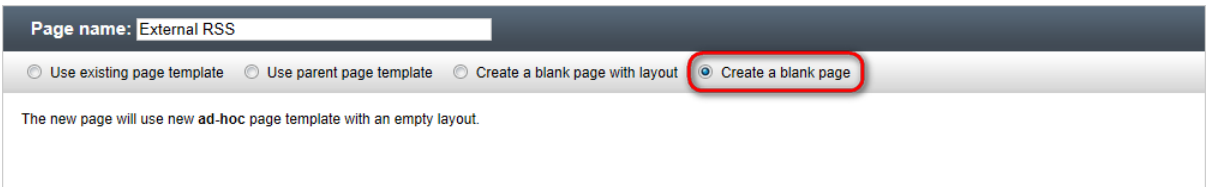
Apart from creating RSS feeds from the CMS content, it is also possible to use external RSS feeds (i.e. feeds published on other websites) as a source of data and display the data on your website.

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**.

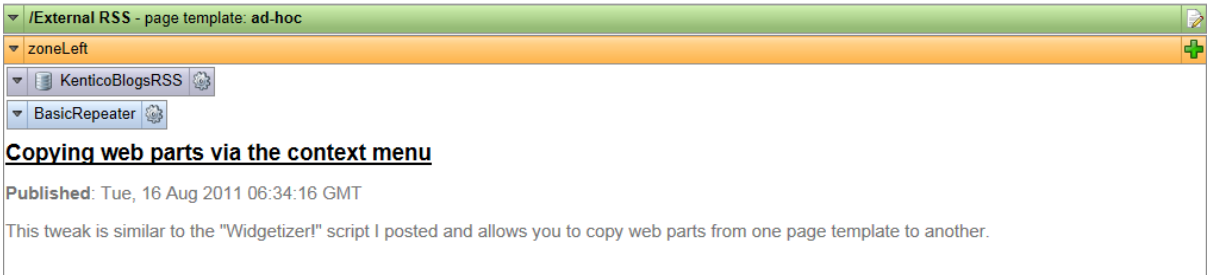
4. Click the **Add web part** () icon again. This time, choose the **Listings and viewers -> Basic repeater** web part and configure its properties as follows:

- **Data source name**: *KenticoBlogsRSS*;
- **Transformation name**: click **New** and create a new transformation with the code listed below.

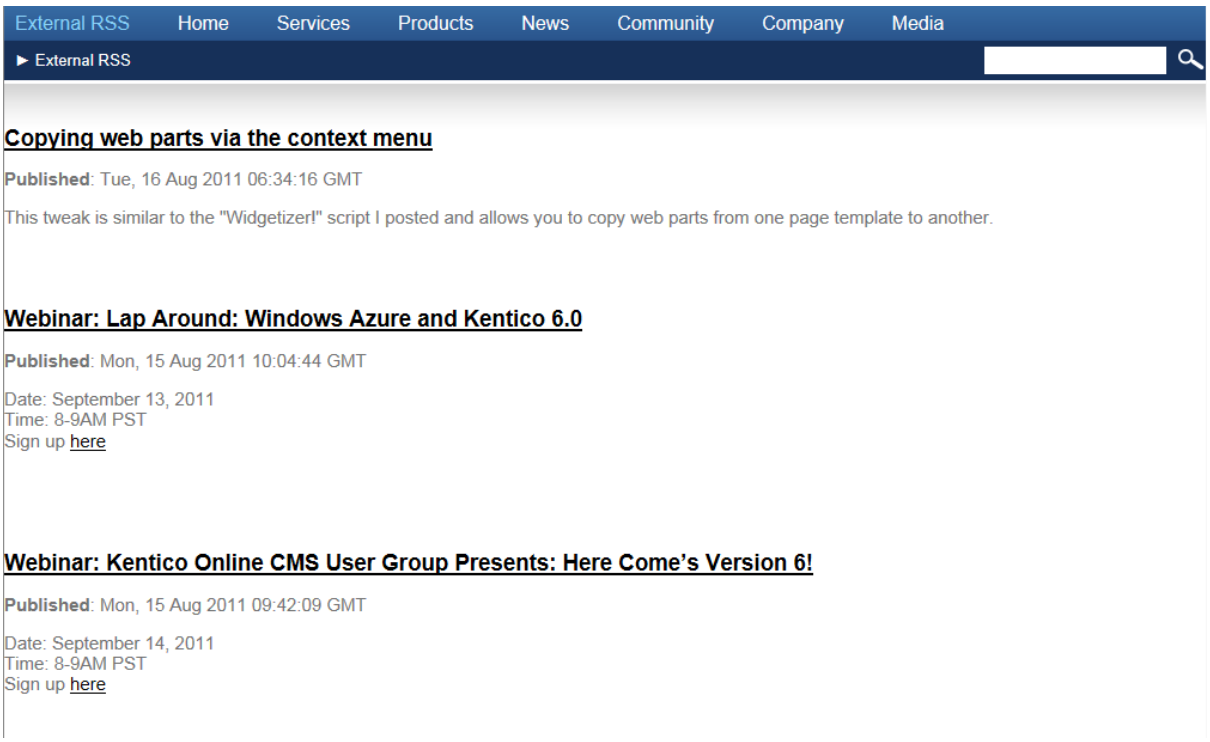
The following is a sample transformation code which transforms the *link*, *pubdate* and *description* elements of each record provided by the feed. Content of the elements is retrieved using the **Eval(string columnName)** method, while the element name is used in the parameter. This way, you can get 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.



8.42.5 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.

8.43 System integration bus

8.43.1 Overview

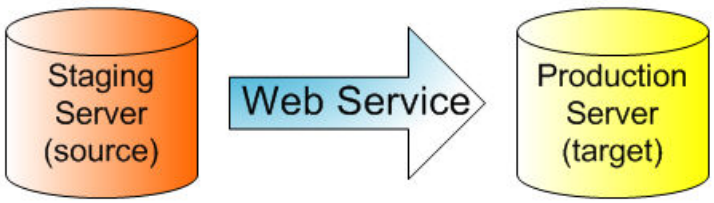
The System integration bus module enables integration of Kentico CMS with external systems. Detailed information about the module, implementation of custom integration connectors, management of integration tasks and usage examples of the module can be found in Kentico CMS Integration Guide.

8.44 Staging

8.44.1 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.

8.44.2 What can be synchronized

The Staging module **supports** synchronization of the following data:

- Document data - documents in the content tree
- Document attachments – if a document contains document attachments or file fields, the files are synchronized together with the document
- Document relationships – if specified relationship and related document exist on the target server, the relationship is also synchronized
- 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 data stored in them
- Media libraries and the files and folders in them
- ACLs (document-level permissions)
- Global objects - all global objects except the ones listed in the list of not-supported data below

The Staging module **does not support** synchronization of the following data:

- Forms data, the forms themselves are synchronized
- Forum posts, the forums themselves are synchronized
- Message board messages, the boards themselves are synchronized
- Blog comments, the blogs and blog posts are synchronized
- Friends
- Messaging module messages
- Abuse reports
- Web analytics

- Export history
- Event log
- Web templates

Staging of Media libraries content

Kentico CMS supports staging of the **physical files and folders** stored in the file system in a Media library. Staging tasks related to Media library files are logged in CMS Desk -> Tools -> Staging -> Objects.

You can limit the maximal size of synchronized media library files by adding the following key to the *appSettings* section of you *web.config* file. The value is entered in **kiloBytes** and files larger than the value will not be synchronized.

```
<add key="CMSMediaFileMaxStagingSize" value="1024" />
```

Please note

Changes in media library files and folders are logged as content staging tasks only when performed via the UI. If you make some changes directly in the library folder in the file system (e.g. upload or update some files via FTP), the changes are not logged.

Also, if you make some change to a file via the UI and then update the file via FTP, the current file (the one updated via 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.

8.44.3 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.

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

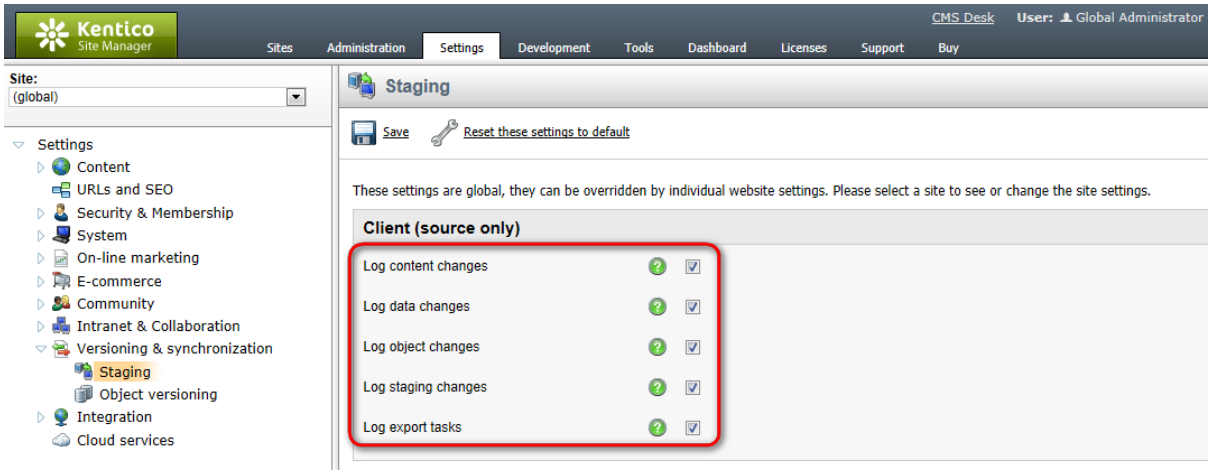
To enable content staging, you first need to open the **bin** folder under your web project and rename the **Microsoft.Web.Services3.dll.rename** file to **Microsoft.Web.Services3.dll**. This needs to be done on both source and target servers.

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.

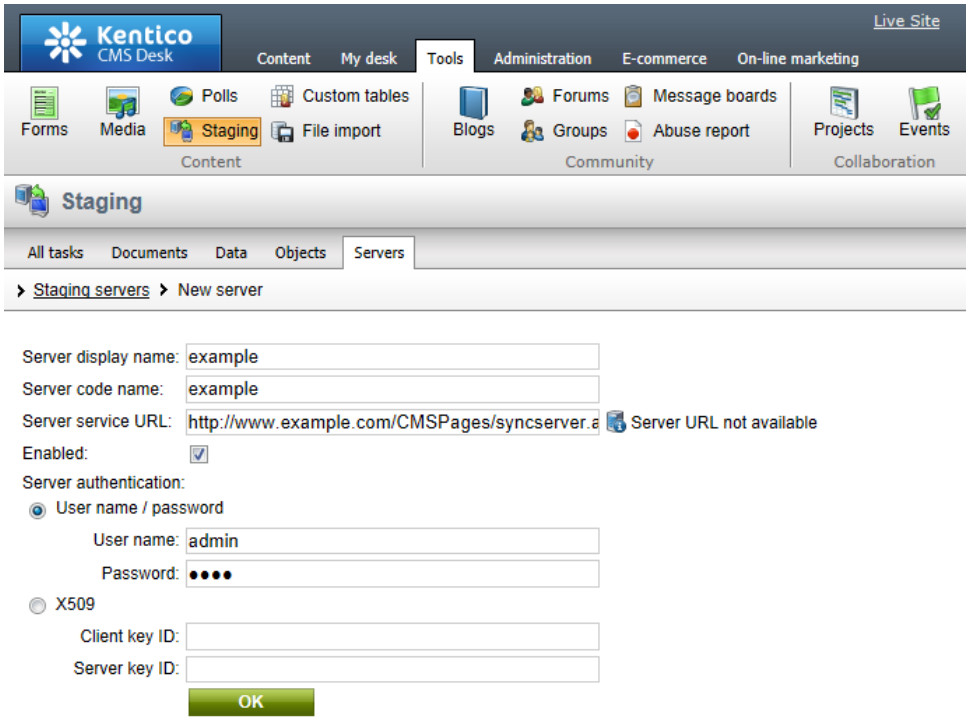


2. Then you need to specify the target server(s). Go to **CMSDesk –> Tools –> Staging -> Servers**.

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

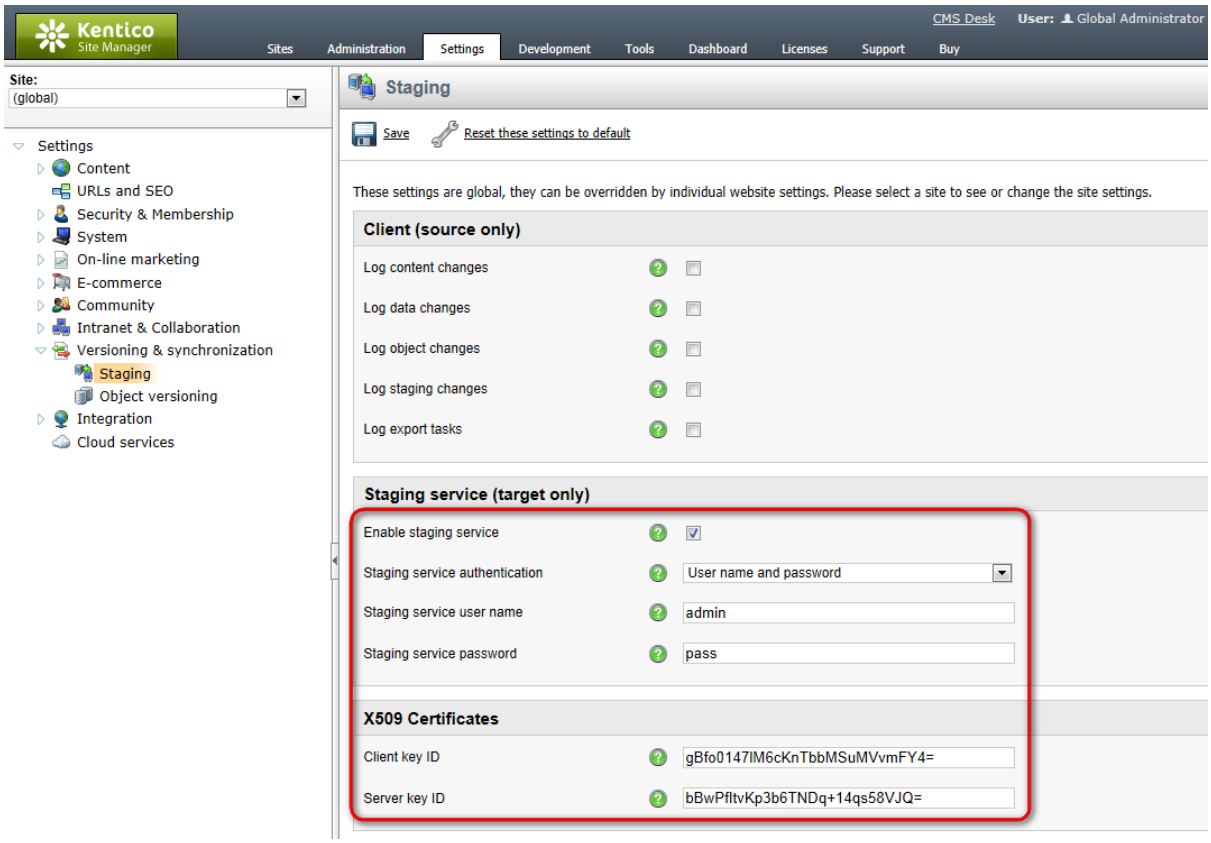
Tasks are logged only when there is at least one target server created and enabled. If there is no server created or if no server is running, no tasks are logged.

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



8.44.4 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.

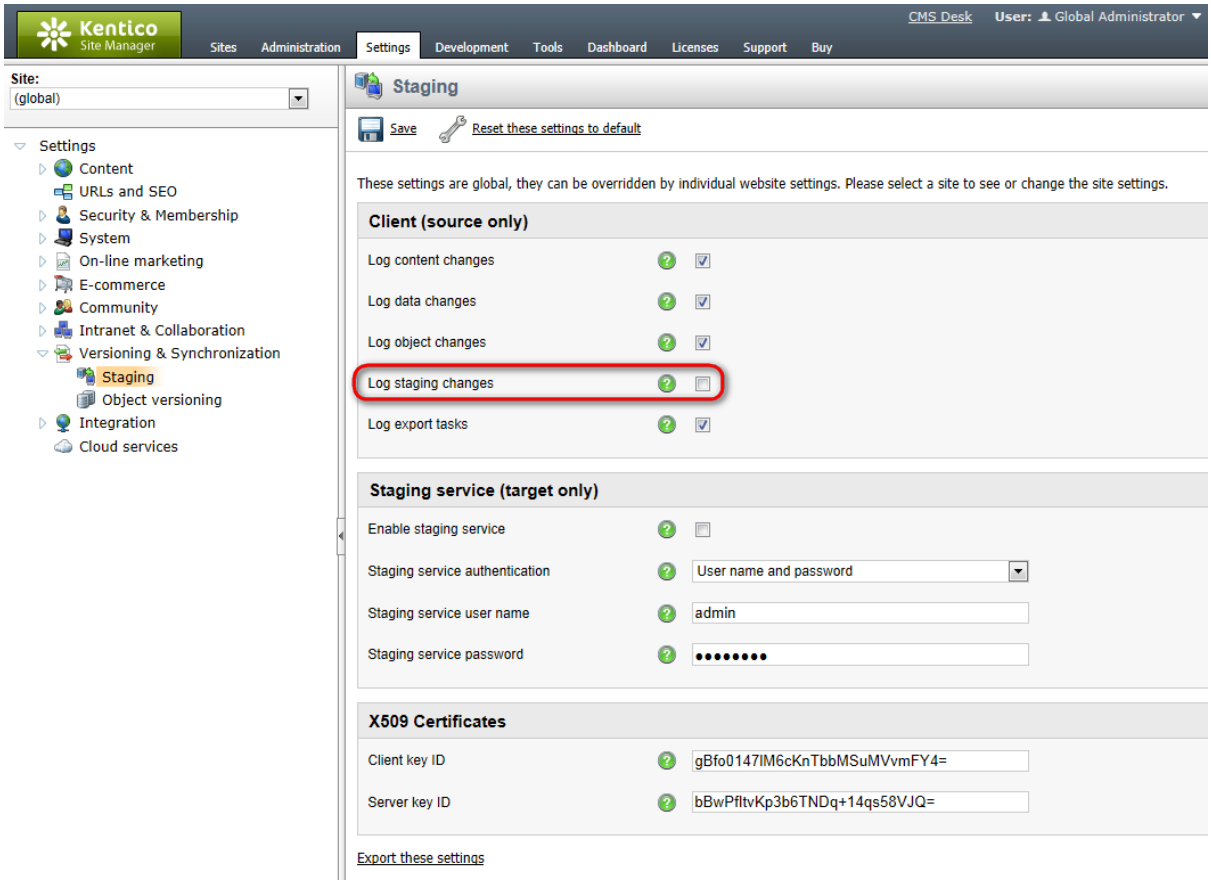
Simple bi-directional staging

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.

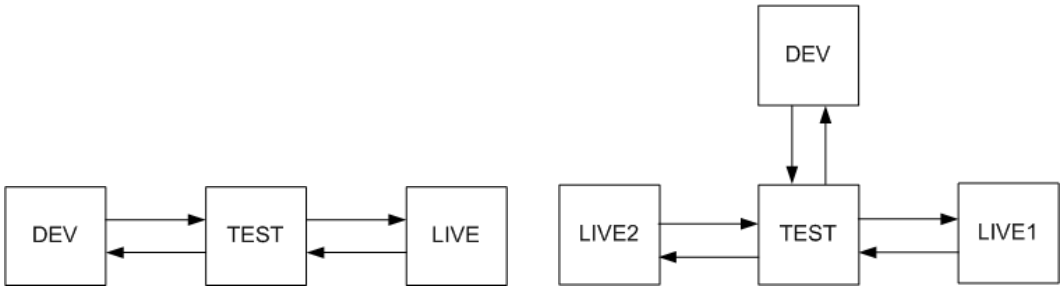


Advanced bi-directional staging

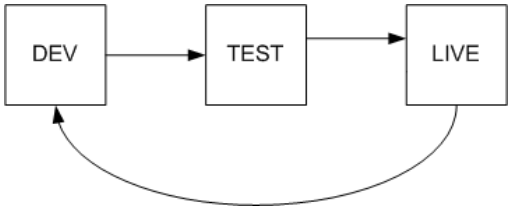
Bi-directional content staging is also possible on more than two servers. In this case, a list of servers where a staging task has already been processed is transferred with each task. This prevents redundant staging tasks from being logged for servers where the changes have already been performed.

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.

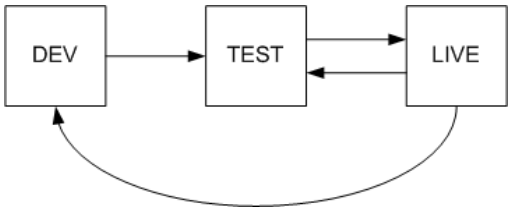
The first option is to have the servers connected using **star topology**.



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.

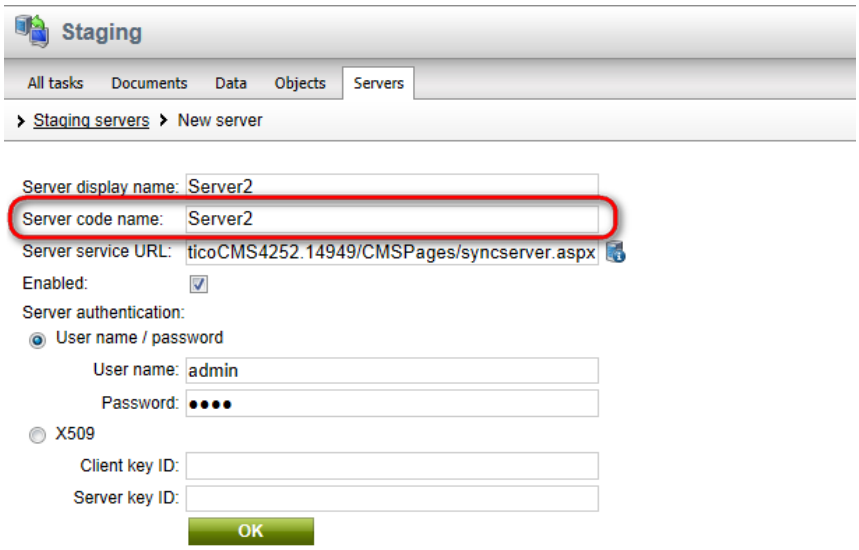


To enable advanced bi-directional staging, you need to go through the following steps:

1. Enter the following key into the *AppSettings* section of the *web.config* file on each server. The key should have a unique value on each server. The value of the key determines the name of the staging server which will be used as target server code name on the rest of the servers.

```
<add key="CMSStagingServerName" value="server2" />
```

2. Configure each server as both a target and a source server, as described in the Staging configuration topic. When specifying target servers in **CMS Desk -> Tools -> Staging -> Servers**, specify all remaining servers in the network. Use the respective values set by the *web.config* keys in the previous step for each **Server code name**.



With this configuration, content staging should work between all servers in both directions, whilst no redundant tasks should be generated.

8.44.5 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**.

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.

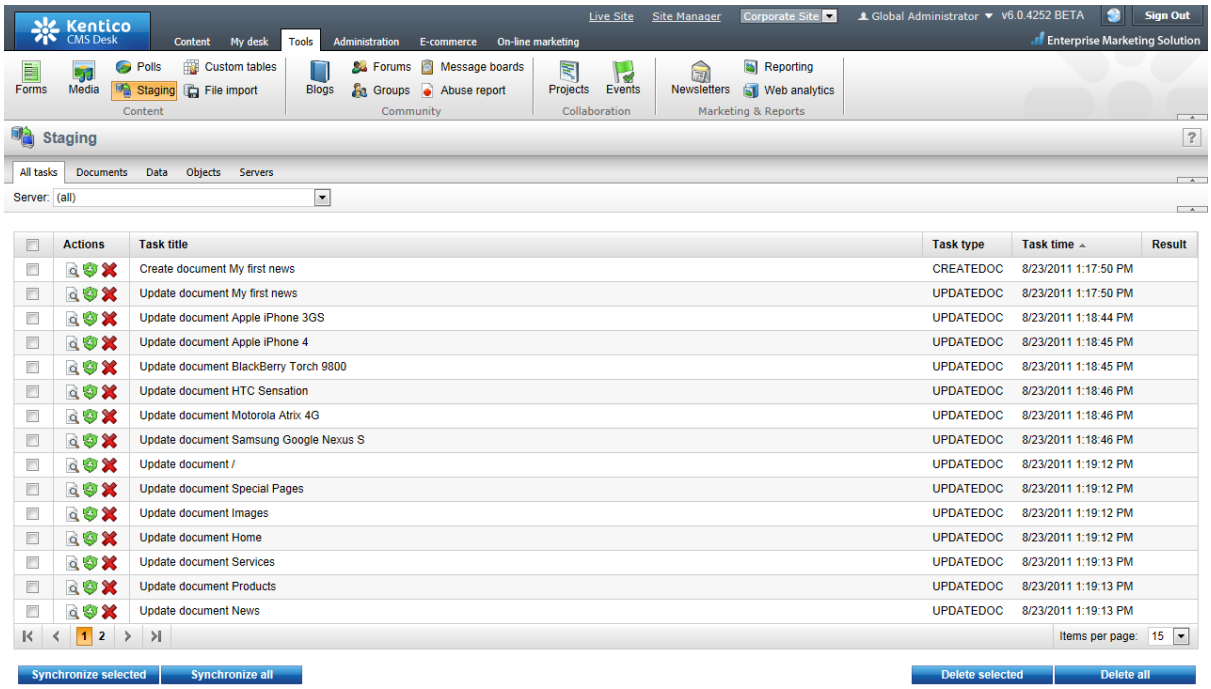
Using the **Server** drop-down, you can choose the target server that you want to synchronize. By choosing **(all)**, you perform synchronization for all available target servers. Then you can perform one of the following actions using the buttons at the bottom:

- **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.

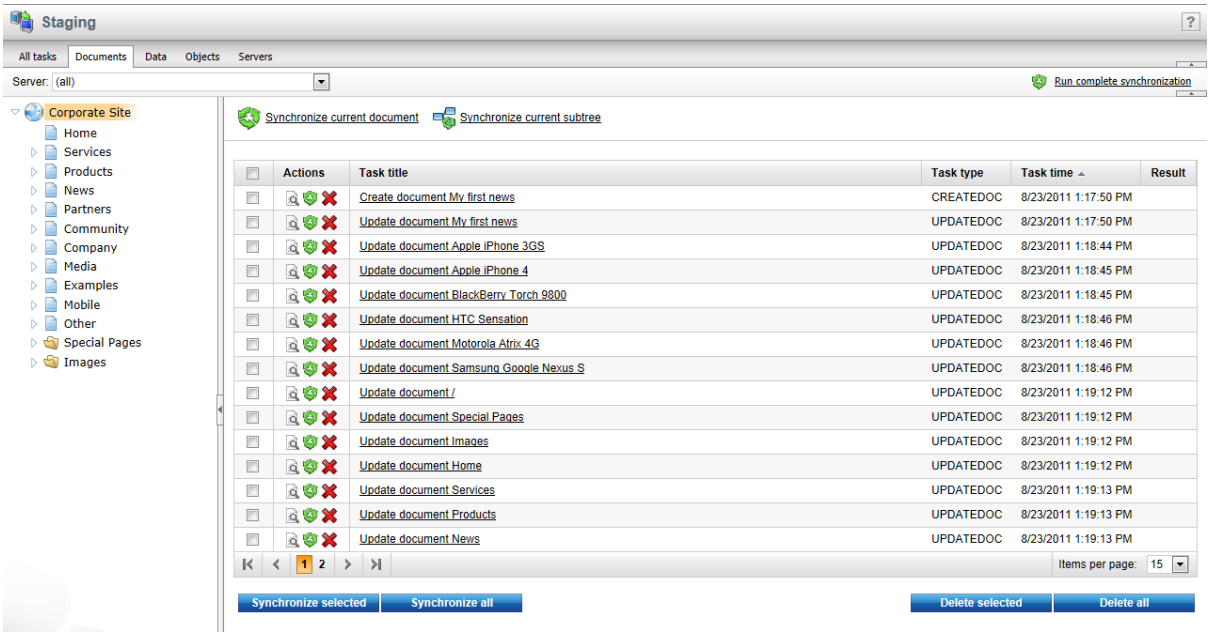
You can perform the same actions as on the **All tasks** tab, as described above.

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.

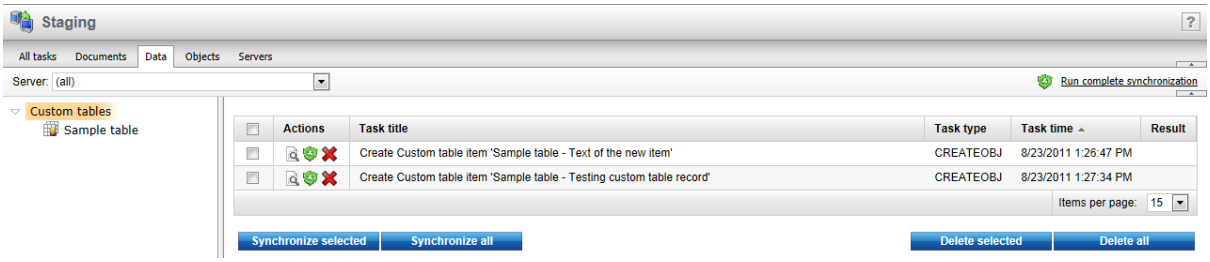
You can also perform the following manual actions. These actions are manual because they are not related to the listed tasks and they can be performed even if there are no synchronization tasks logged:

- **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* column. Items which do not have this column (typically in custom tables imported from older versions of Kentico CMS), changes made to them are not logged. However, you can edit the custom table in CMS Desk -> Tools -> Custom tables. You will see a warning message with a link letting you generate the GUIDs for the table.



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:

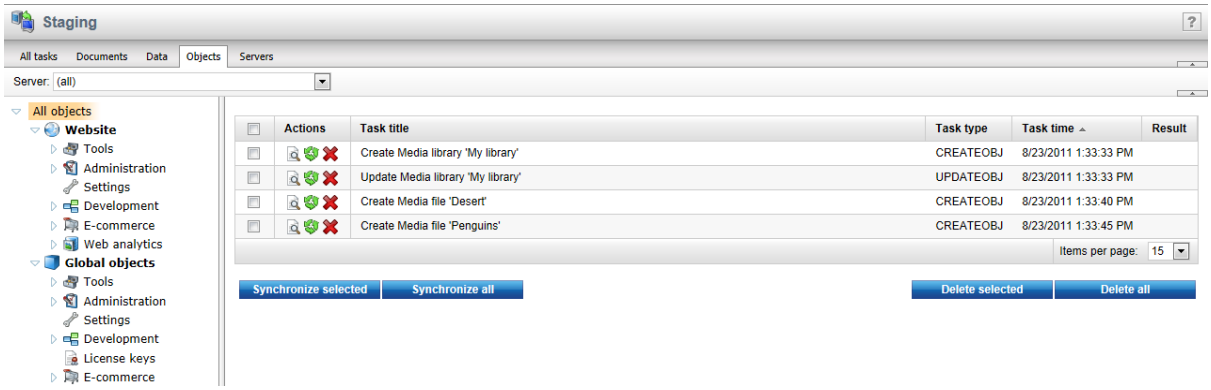
- **CREATEFOLDER** - folder was created
- **RENAMEFOLDER** - folder was renamed
- **COPYFOLDER** - folder was copied
- **MOVEFOLDER** - folder was moved
- **DELETEFOLDER** - folder was deleted

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

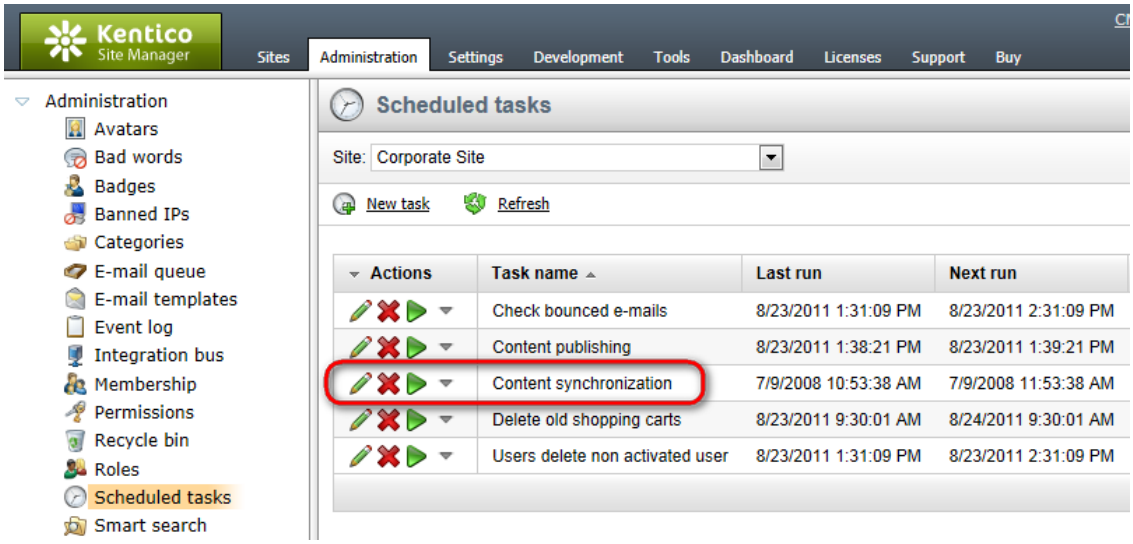
with the same name to prevent such possibility. The default value is false since staging can use multiple target instances targeted with the same names.



8.44.6 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.

8.44.7 Using X.509 authentication

In order to use **X.509** authentication, you need to install your own certificates, or you may use our sample ones.

Using the sample certificates

Kentico CMS is delivered with sample client and server private certificates. In order to install them, you need to do the following on the source server and on the target server:

1. Choose **Start -> Run**, type **mmc** and press **Enter**.

2. In the console window, choose **File -> Add/Remove Snap-in**.

3. Click **Add** and choose **Certificates**.

4. Choose **Computer account** in the next step.

5. Choose **Local computer** in the next step. Finish adding the **Certificates** snap-in.

6. Unfold **Certificates (Local Computer)** under the console root, right-click **Personal** and choose **All Tasks -> Import...** The **Certificate Import Wizard** starts.

7. Import the appropriate certificate from the appropriate **.pfx** file in **C:\Program Files\Kentico CMS\<version>\SampleCertificates**. Use *Client private.pfx* for the client certificate and *Server private.pfx* for the server certificate.

8. Enter the following password for the sample certificates (it is the same for the client and the server certificate): **wse2qs**

9. Now you need to grant the **READ** permissions for the certificate file to the ASP.NET account (names of the account under different operating systems are described in the Disk permissions problems chapter). You can do that using the **WseCertificate3.exe** tool that can also be found in **C:\Program Files\Kentico CMS\<version>\Sample Certificates**

10. Run the **WseCertificate3.exe** tool.

11. Choose **Local Computer** in the **Certificate Location** field.

12. Choose **Personal** in the **Store Name** field.

13. Click **Open Certificate** and choose either the client or the server certificate.

14. Click **View Private Key File Properties...** and grant the **READ** permission for this file to the ASP.NET account (names of the account under different operating systems are described in the Disk permissions problems chapter).

Using your own certificates

If you're using your own certificates (highly recommended), you will need to update the following values in the **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.

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.

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.

8.44.8 Security

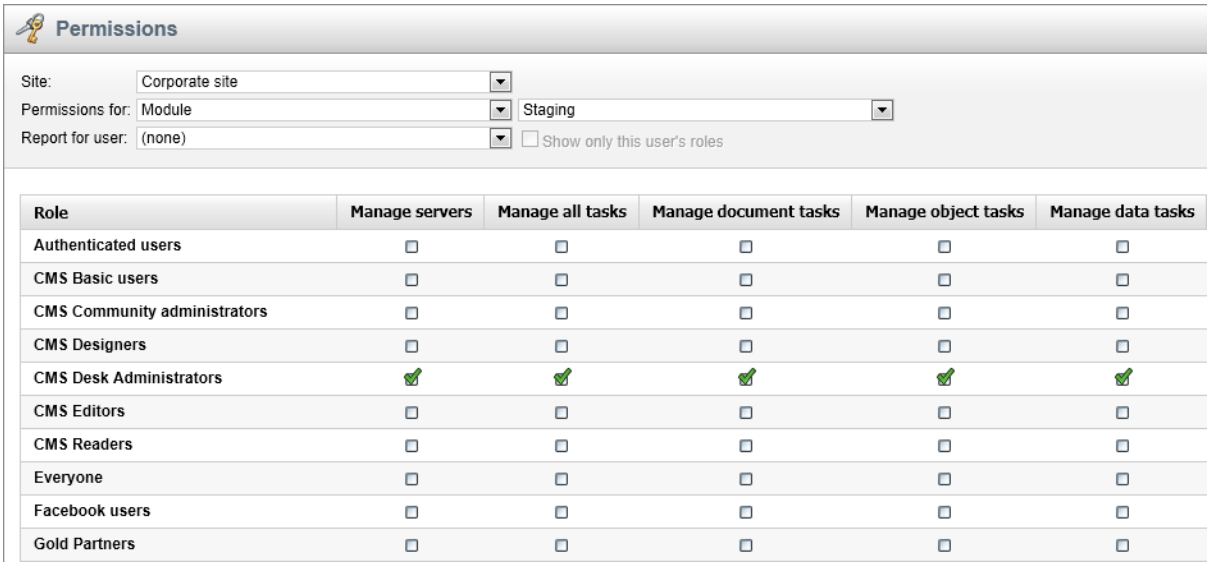
You can control the access to the Staging module in **Administration -> Permissions**. You need to select:

- **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



Role	Manage servers	Manage all tasks	Manage document tasks	Manage object tasks	Manage data tasks
Authenticated users	☐	☐	☐	☐	☐
CMS Basic users	☐	☐	☐	☐	☐
CMS Community administrators	☐	☐	☐	☐	☐
CMS Designers	☐	☐	☐	☐	☐
CMS Desk Administrators	✓	✓	✓	✓	✓
CMS Editors	☐	☐	☐	☐	☐
CMS Readers	☐	☐	☐	☐	☐
Everyone	☐	☐	☐	☐	☐
Facebook users	☐	☐	☐	☐	☐
Gold Partners	☐	☐	☐	☐	☐

8.44.9 Synchronization using API

In special cases, you may need to use Kentico CMS API to perform your own synchronization process. There are several methods you can use to work with synchronization.

WorkflowEngine.DocumentHelper. LogSynchronization	Creates a synchronization task for the specified document. This method does not actually perform the synchronization, it only creates the task which can be later synchronized. By default, the method creates the task for all the enabled servers. You can use the overriden version with serverId parameter to specify the particular server.
CMSStaging.StagingHelper. RunSynchronization	Runs the synchronization of specified task for the specified server or servers.

Here is an example code how to synchronize the content of the document "/Home" to server "Staging. Target1":

[C#]

```
using CMS.TreeEngine;
using CMS.CMSHelper;
using CMS.Synchronization;
using CMS.WorkflowEngine;
```

```
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 full document record (required for synchronization logging)
  node = tree.SelectSingleNode(CMSContext.CurrentSiteName, node.NodeAliasPath,
node.DocumentCulture, false, node.NodeClassName, false);
  if (node != null)
  {
    // Get the server
    ServerInfo si = ServerInfoProvider.GetServerInfo("Staging.Target1",
CMSContext.CurrentSiteID);
    if (si != null)
    {
      // Log the synchronization task for the given server
      TaskInfo ti = DocumentHelper.LogSynchronization(node, TaskTypeEnum.
UpdateDocument, tree, si.ServerID);
      if (ti != null)
      {
        // 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, but previous methods should be enough to perform any simple synchronization actions.

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)
   {
    // Creates the synchronization task
     TaskInfo ti = TaskInfoProvider.LogSynchronization(userObj, TaskTypeEnum.
UpdateObject, null, null, si.ServerID);
     if (ti != null)
```

```
    {
     // Runs the synchronization task
      StagingHelper.RunSynchronization(ti.TaskID, si.ServerID);
    }
  }
}
```

8.44.10 Staging internals and API

8.44.10.1 Overview

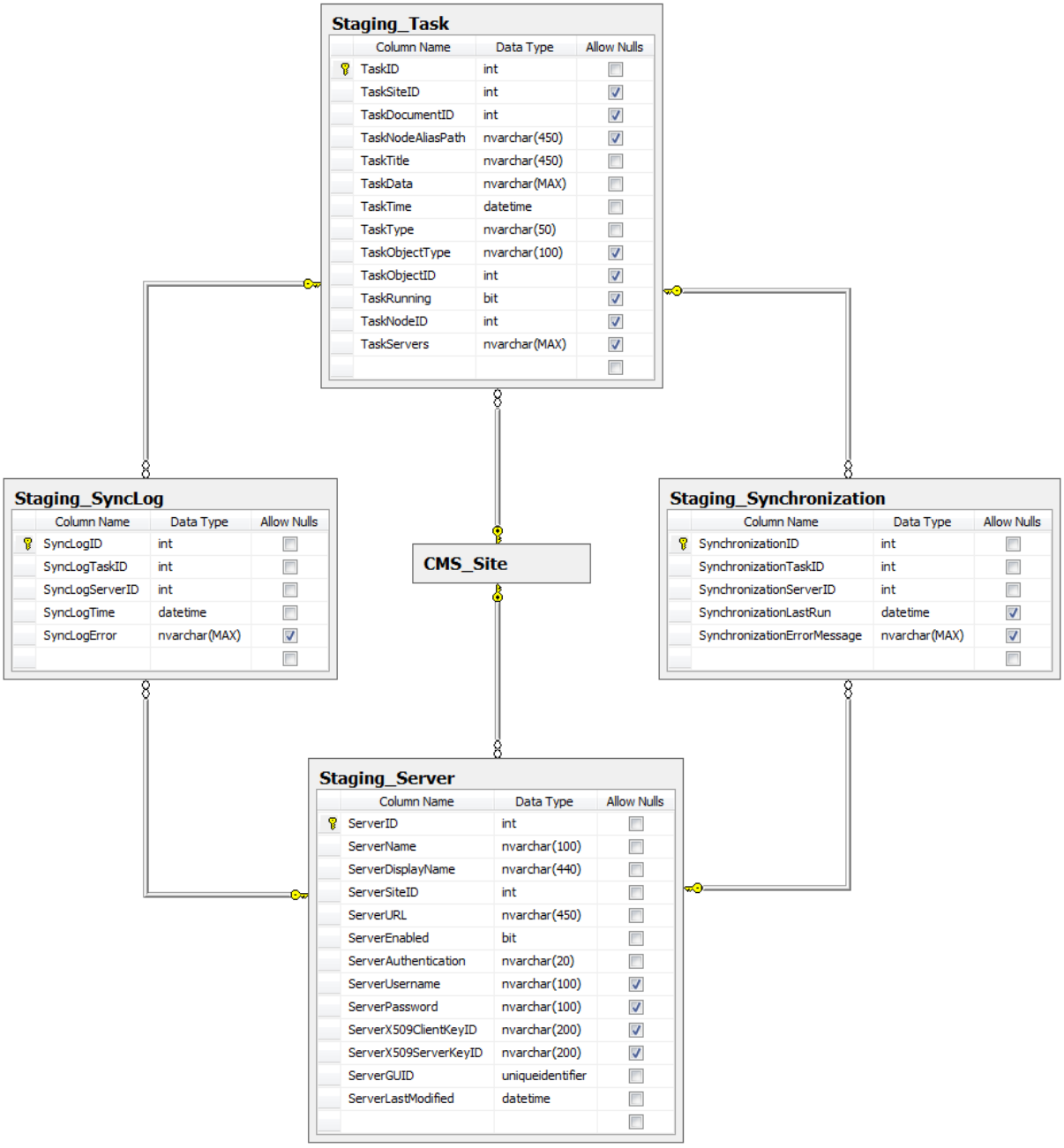
In this chapter, you will learn which database tables and API classes are used in the Staging module. You will also see the most common API examples.

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.

8.44.10.2 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.



8.44.10.3 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.

8.44.10.4 API examples

8.44.10.4.1 Overview

The following topic show examples of how the API of the Staging module can be used:

- 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;
```

8.44.10.4.2 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);
}
```

8.44.10.4.3 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.

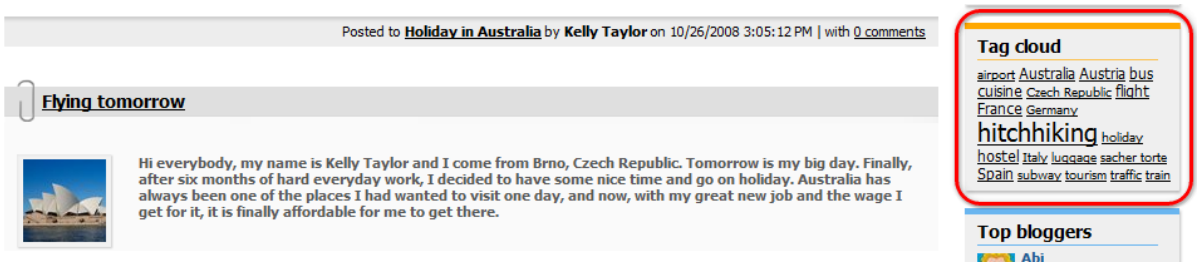
```
private bool DeleteTasks()
{
    // Get server
    ServerInfo server = ServerInfoProvider.GetServerInfo("MyNewServer", CMSContext
.CurrentSiteID);

    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;
}
```

8.45 Tags

8.45.1 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.

8.45.2 Tagging documents

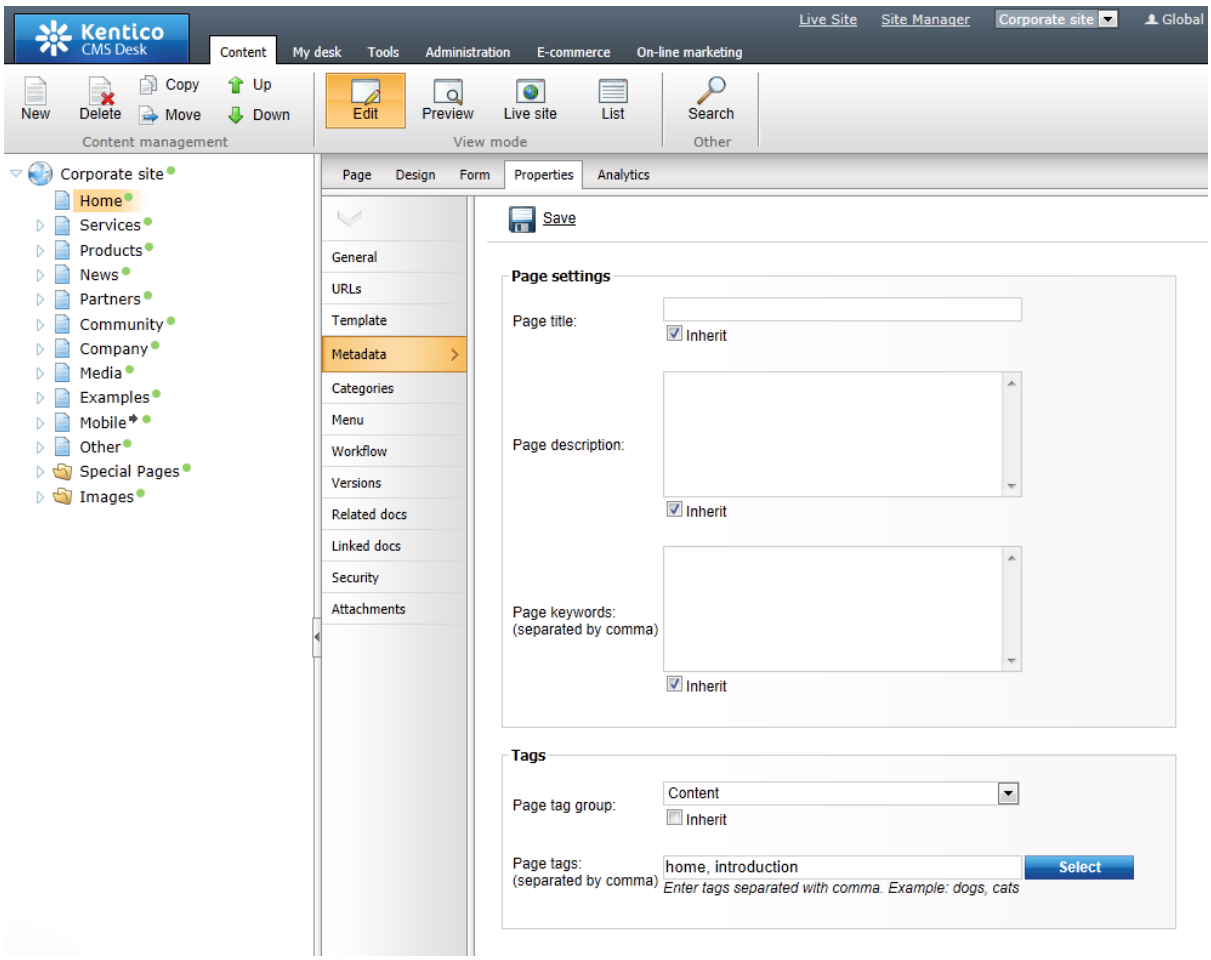
If you need to learn how to tag a document, the following examples describe this process. There are two ways of tagging documents:

- You can add tags to a document either 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.



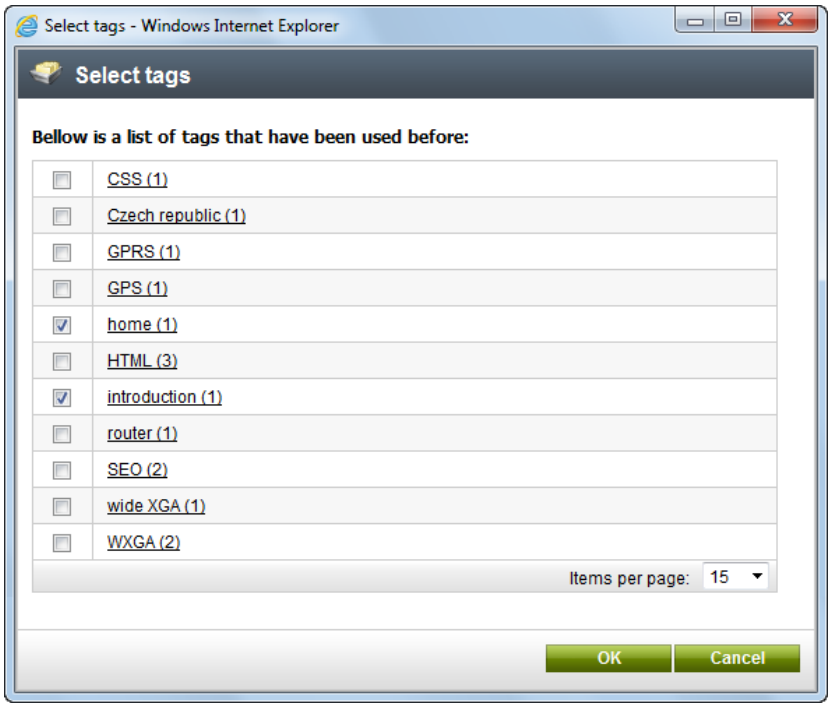
3. Choose a tag group using the **Page tag group** drop-down list. If you check the **Inherit** check-box, tag group will be inherited from the document's parent.

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

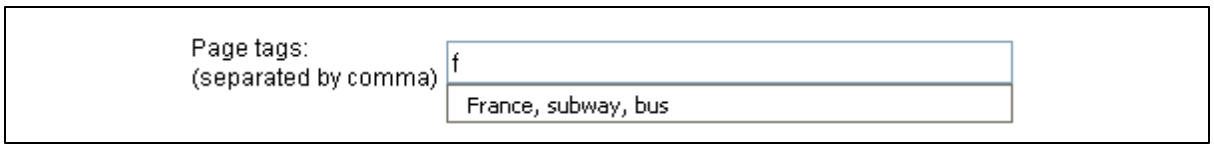
In case that you are entering a tag consisting of more than one word, you should enclose it within quotation marks. Multiple long tags can also be entered and can be also divided by both blank spaces and commas:

"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:

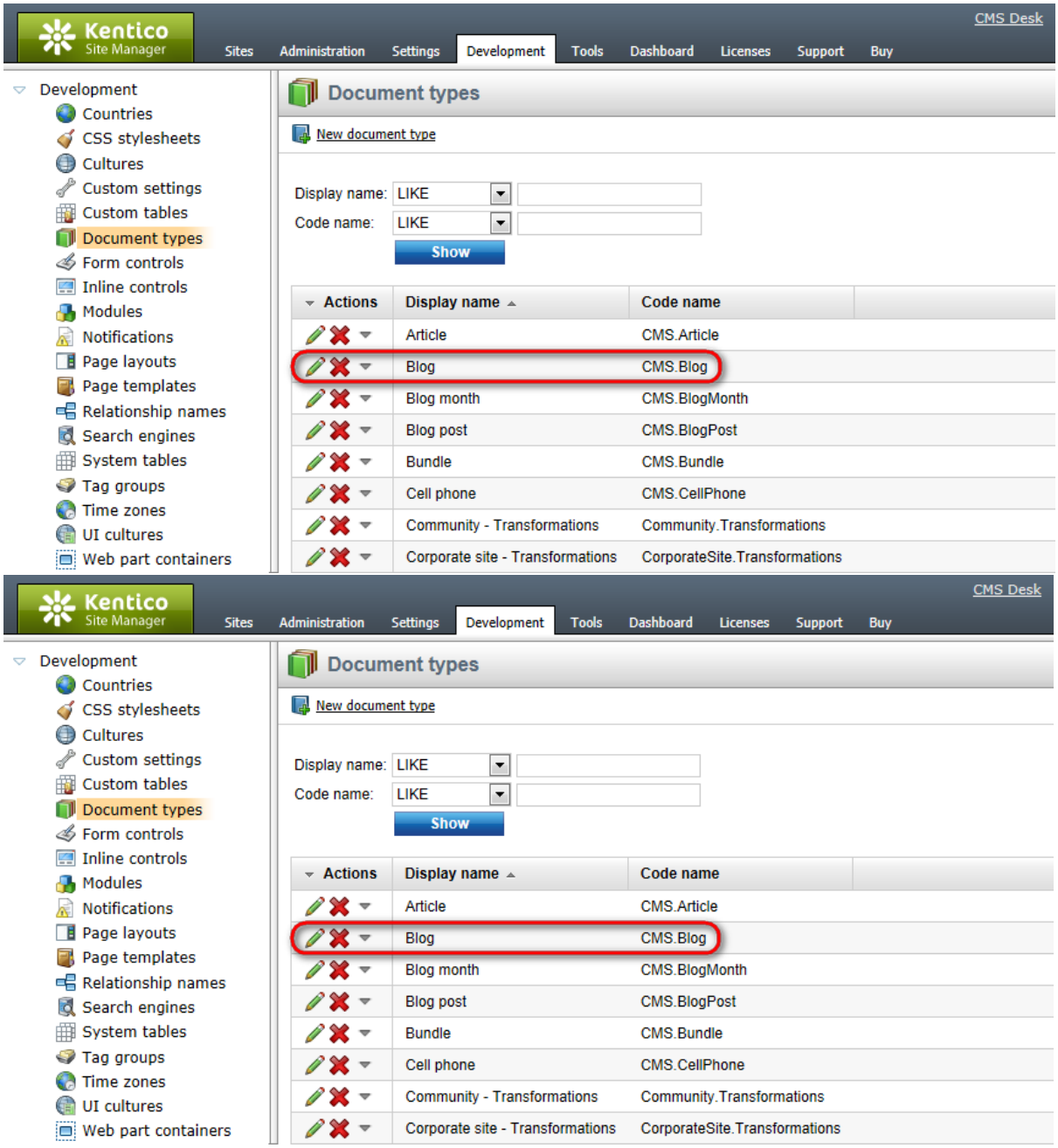
"tag@1", "tag#2", "long, strange: tag@#$"

The page tags field has also an insinuation function implemented. This functions offers you tags from the selected tag group while you are writing:



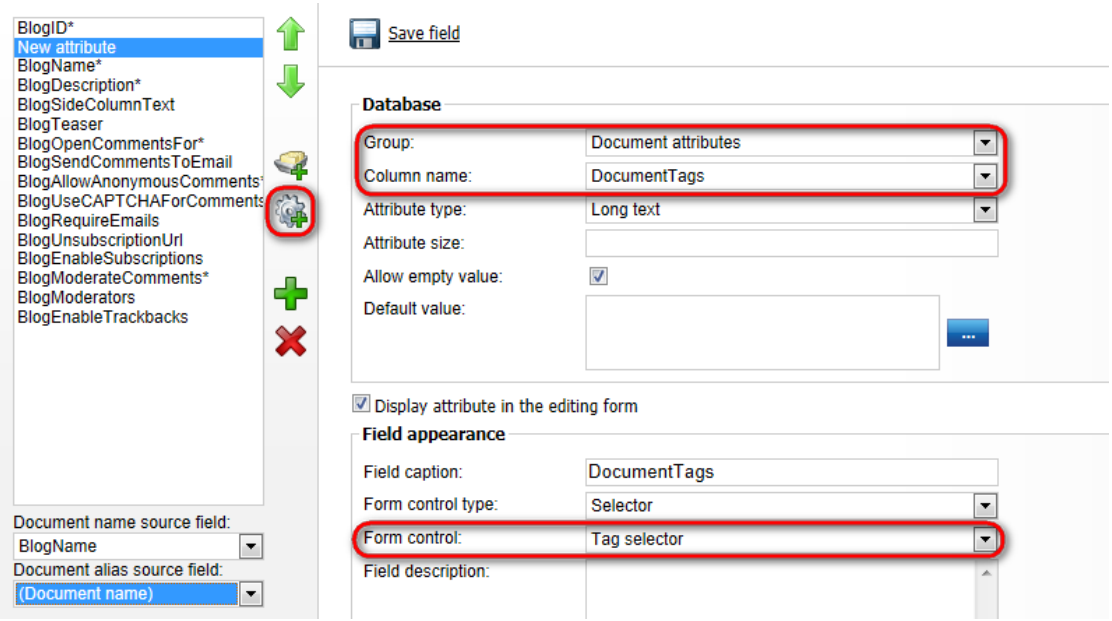
Tagging documents using Fields

1. Go to **Site Manager -> Development -> Document types** and choose to **Edit** (🖉) a document type.



2. Switch to the **Properties** tab, click the **New system attribute** ( ) icon to create a new system attribute and from the corresponding drop-down lists select the following values:

- **Group:** Document attributes
- **Column name:** DocumentTags
- **Form control type**: Selector
- **Form control:** Tag selector



Click **Save field** to save 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.

8.45.3 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.

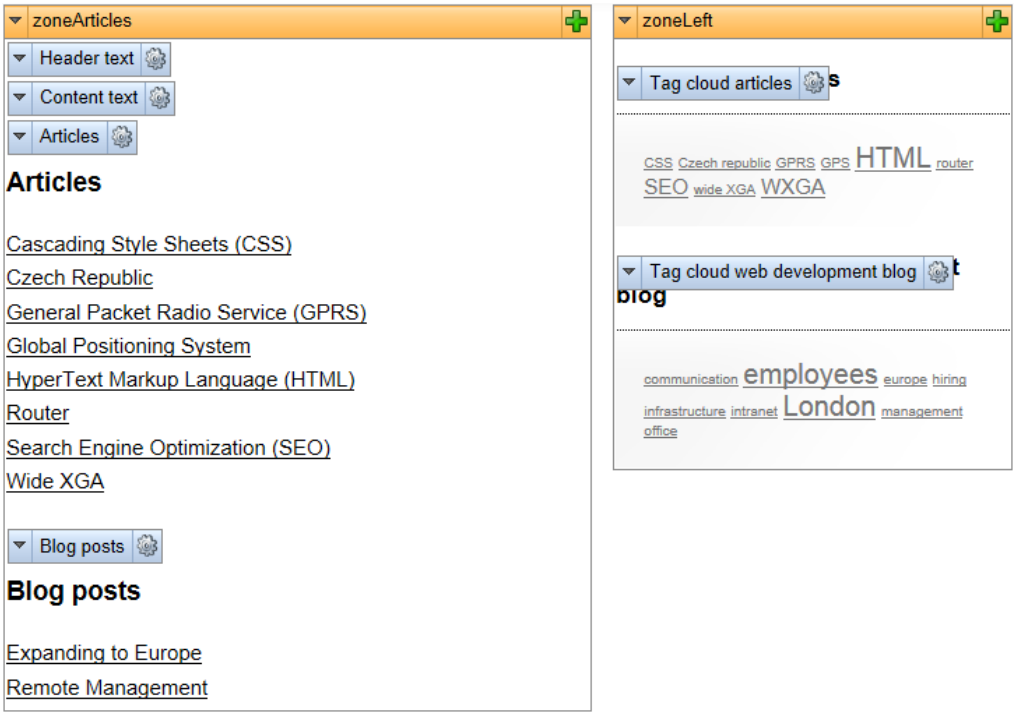
The web part has the following specific properties:

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.

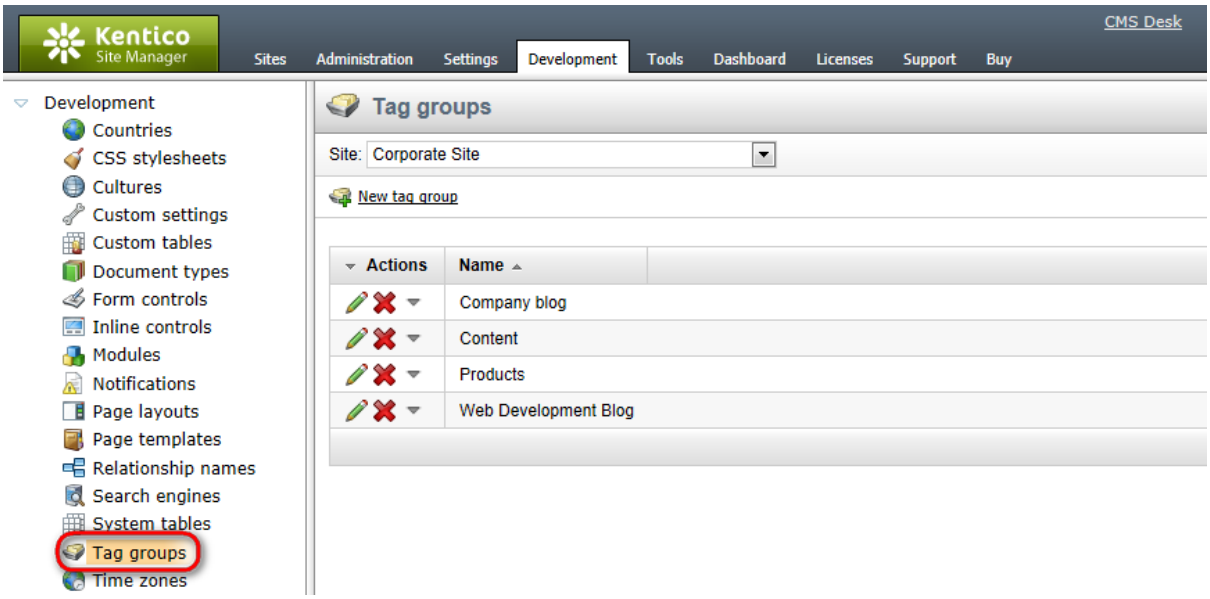
If you would like to see examples of how the **Tag cloud** web part is added to the site, please refer to:

- 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.

8.45.4 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.

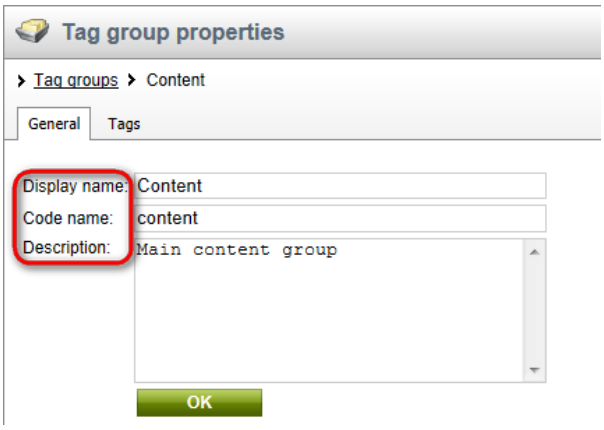
The default tag group **Content** is available for all Kentico CMS sample sites. It is inherited and should not under standard circumstances be deleted from the system.

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 the **New tag group** link. 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



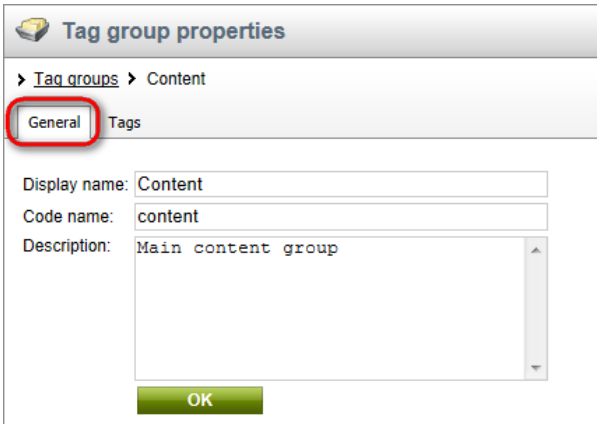
2. Click **OK** to confirm the values you have entered. The tag group will be created and become visible in the list.

Editing tag groups

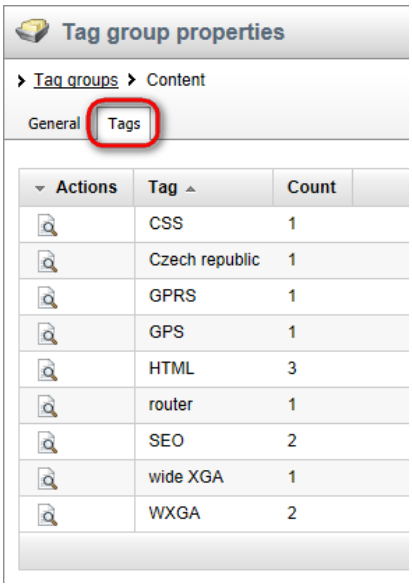
In this example, you will learn how to edit an existing tag group.

1. To edit an existing tag group, go to **Site Manager -> Development -> Tag groups** and choose to **Edit** ( ) one of the listed tag groups.

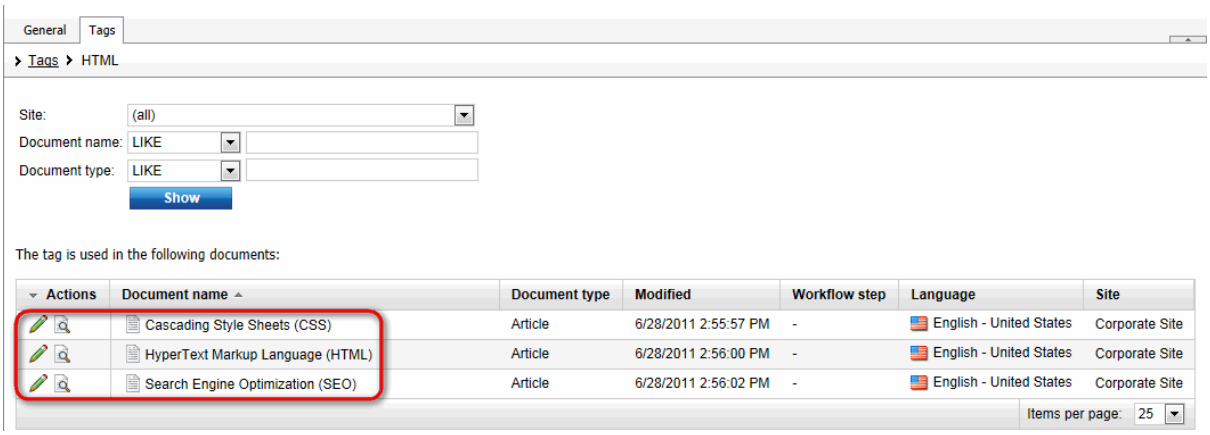
2. If you would like to change some of its properties, go to the **General** tab and change either the **Display name**, **Code name** or **Description** of the tag group.



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**.

8.45.5 Tags internals and API

8.45.5.1 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.

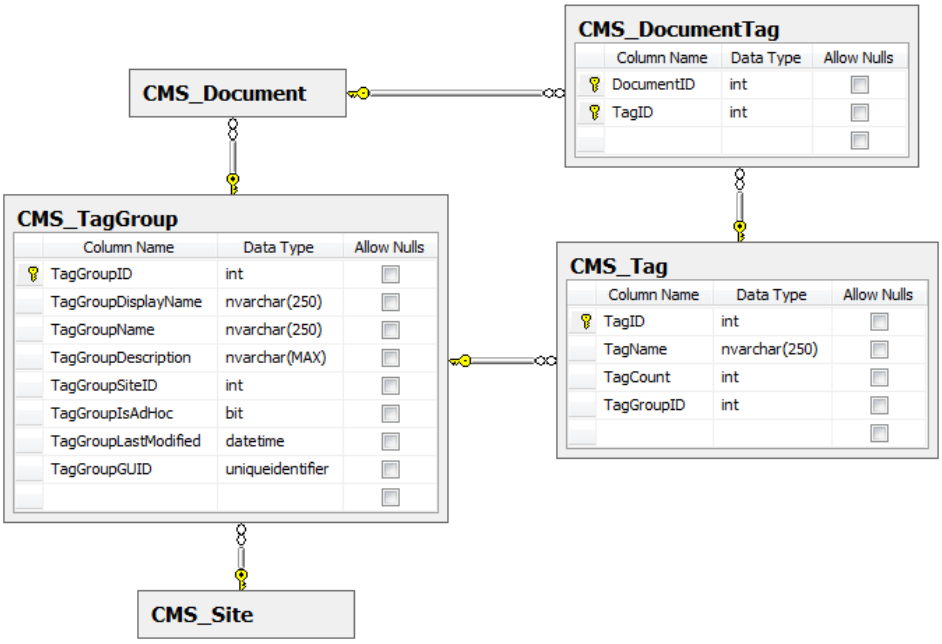
Advanced API examples can be found in the API programming and Kentico CMS internals section of this

guide. For detailed API documentation, such as a list of all methods from the module classes, please refer to Kentico CMS API Reference.

8.45.5.2 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.



8.45.5.3 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.

8.45.5.4 API examples

8.45.5.4.1 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\Development\Tags\Default.aspx.cs***.

8.46 Time zones

8.46.1 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.

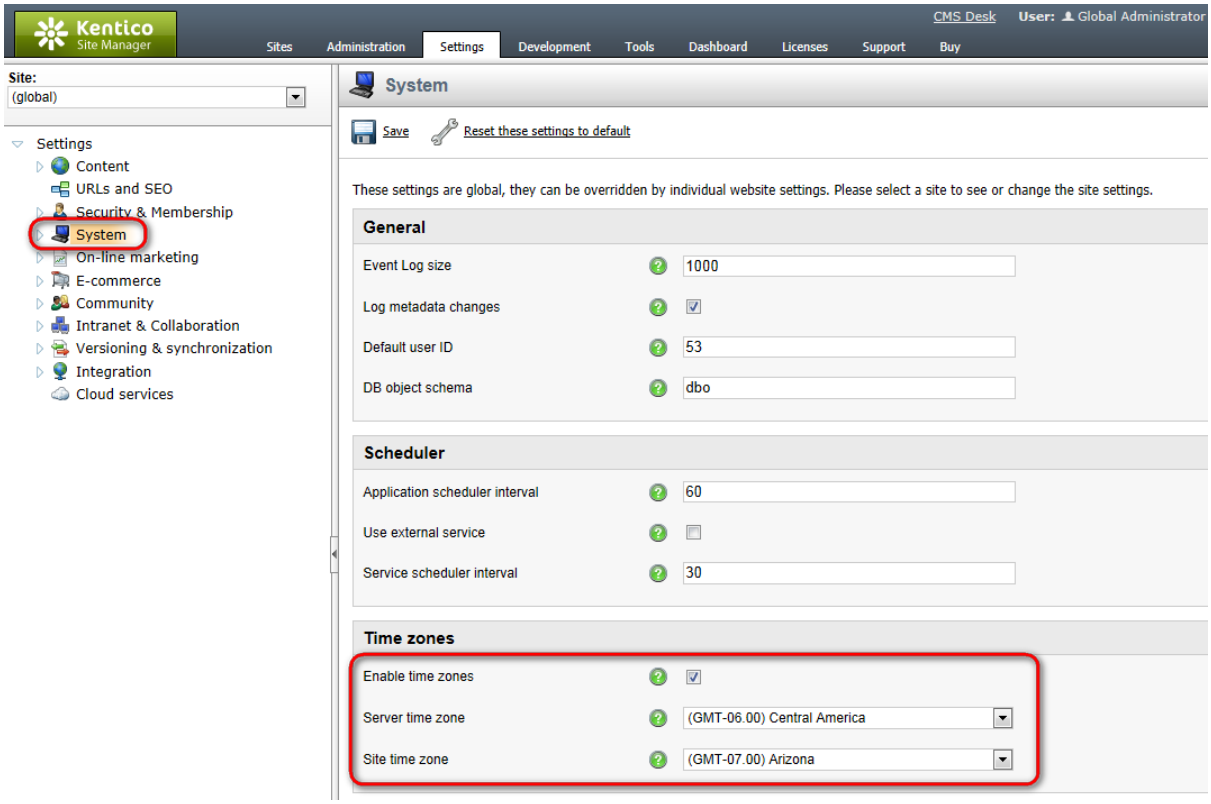
A typical example of use is displaying the time of forum posts when you have a global community – while the server may be located in New York (GMT -5:00), visitors coming from Paris (GMT +1:00) may see their new posts were added at 8am, while they would expect to see 2pm according to their current time.

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.

8.46.2 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:

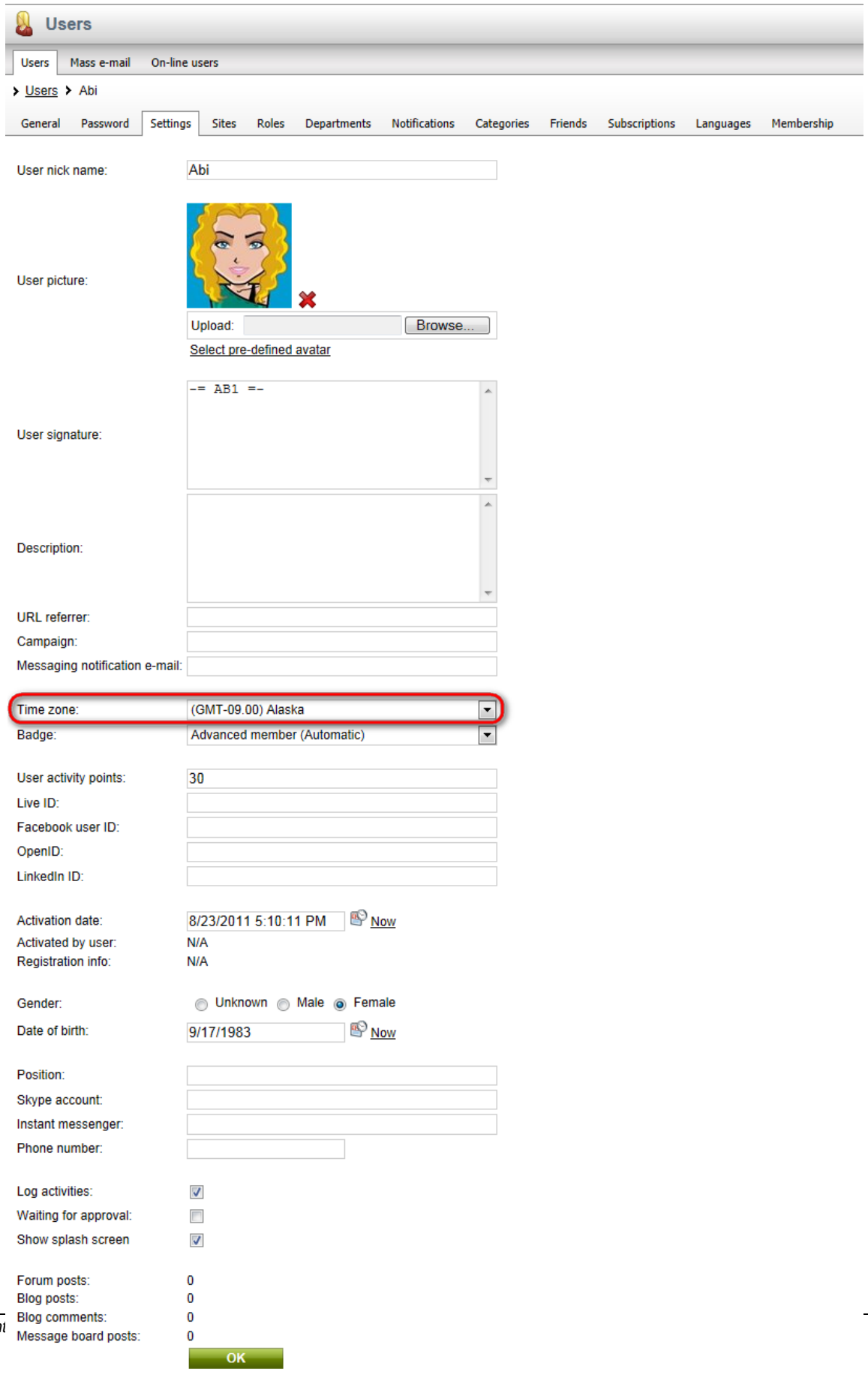
- **Server time zone** - time zone of the physical server location
- **Site time zone** - default time zone of the website selected by the **Site** drop-down list in the top-left part of the page; this time zone can also be set globally and inherited by sites that don't have it set differently



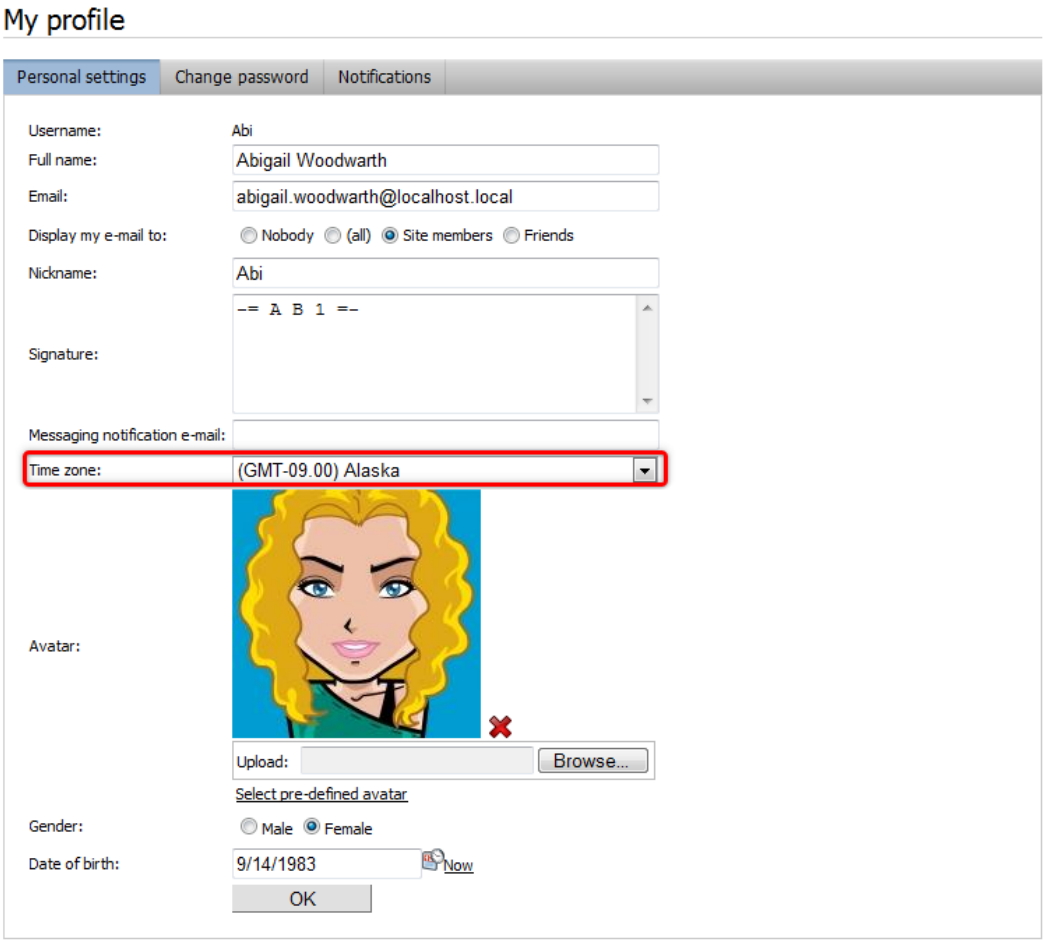
8.46.3 Setting user's time zone

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.

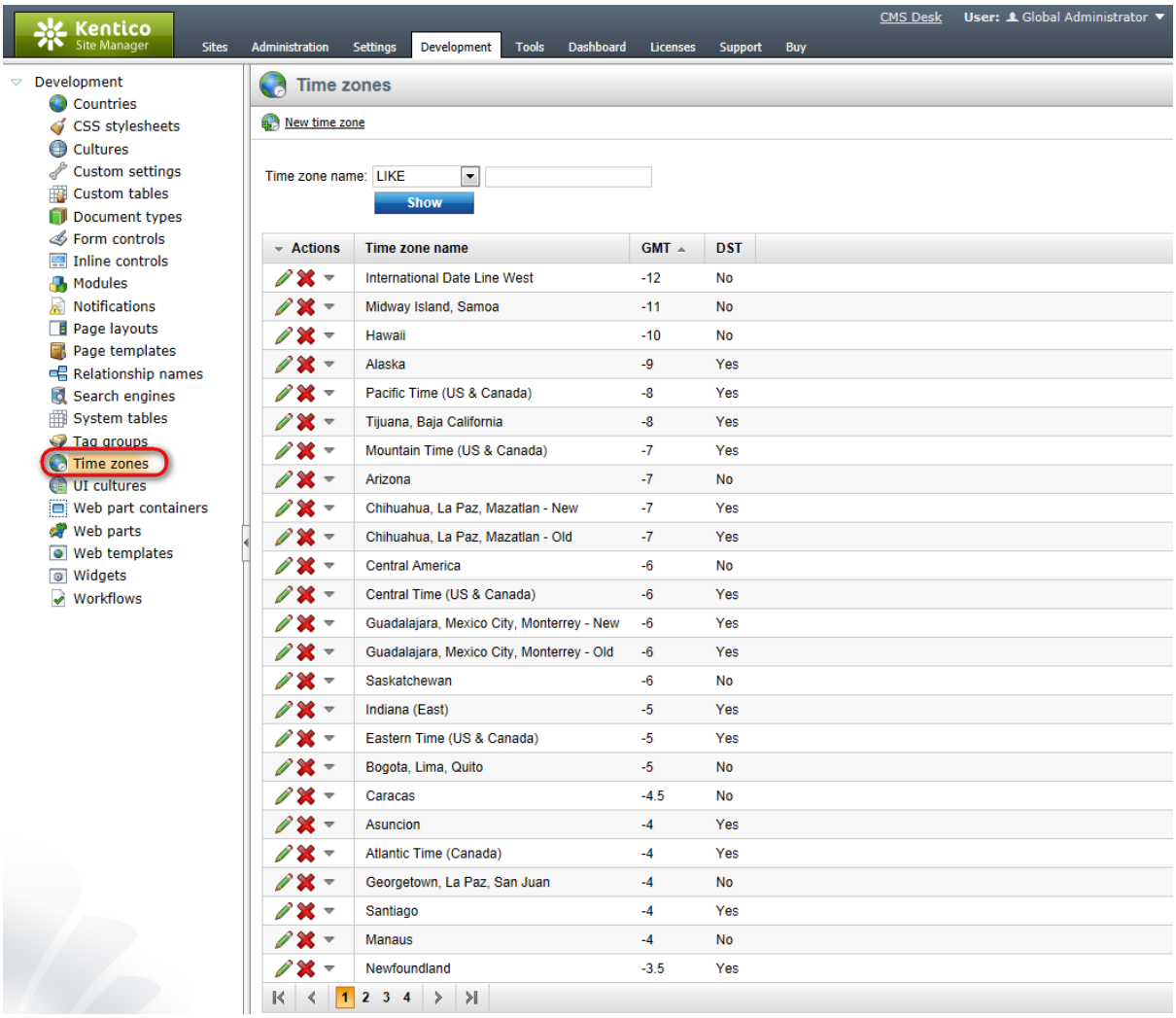

Users
Users
Mass e-mail
On-line users
Users › Abi
General
Password
Settings
Sites
Roles
Departments
Notifications
Categories
Friends
Subscriptions
Languages
Membership
User nick name:
Abi
User picture:
Upload:
Browse...
Select pre-defined avatar
User signature:
-= AB1 =-
Description:
URL referrer:
Campaign:
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.



8.46.4 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.

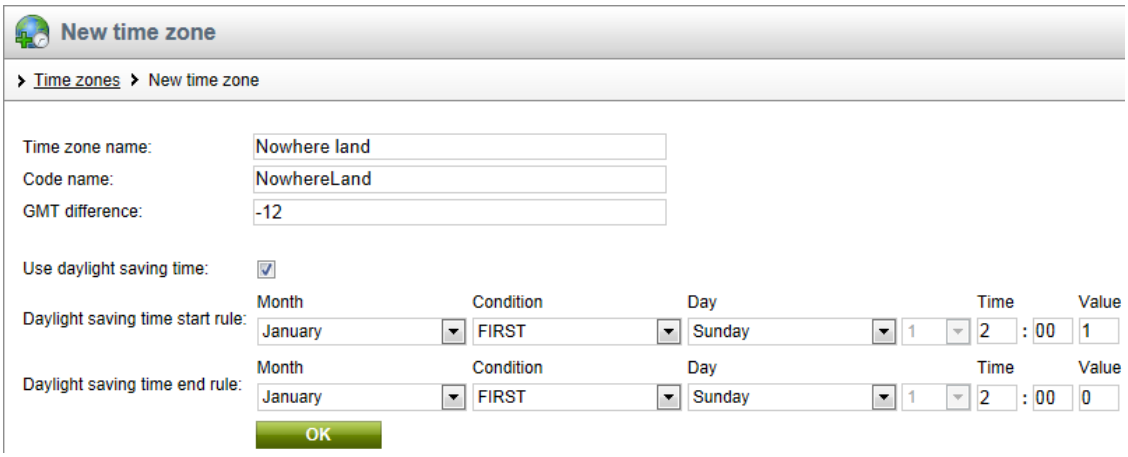
Time zones in the list can be **Edited** (✎) or **Deleted** (✖). You can also create new time zones by clicking the **New time zone** link.

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 the **New time zone** link.

2. Fill in the details that you can see in the following screenshot:



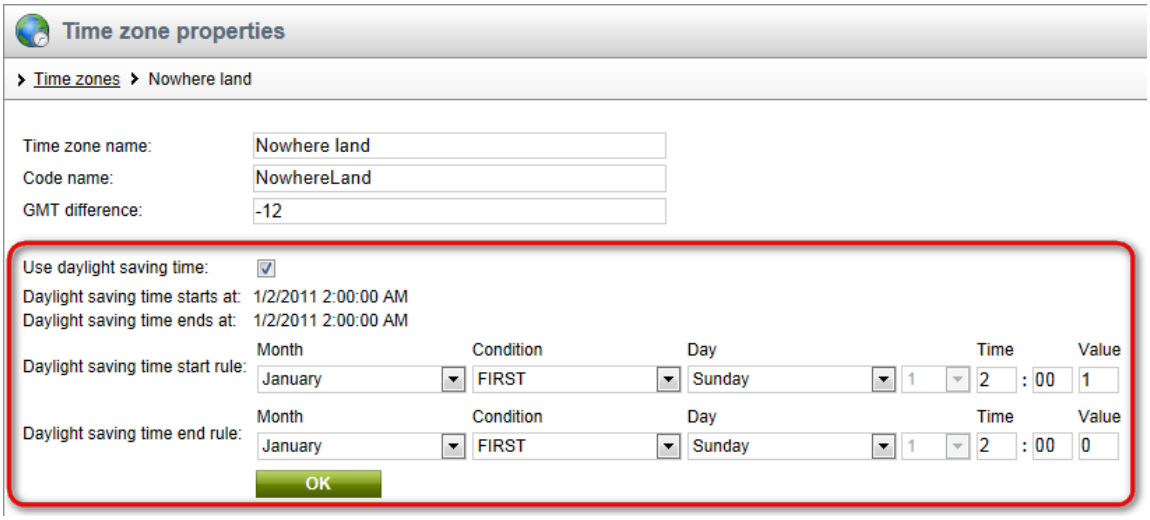
3. Click **OK**. You have just created the time zone. Now if you switch back to the time zones list, you should see the new time zone present among the records.

8.46.5 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.

8.46.6 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.

8.46.7 Time zones internals and API

8.46.7.1 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.

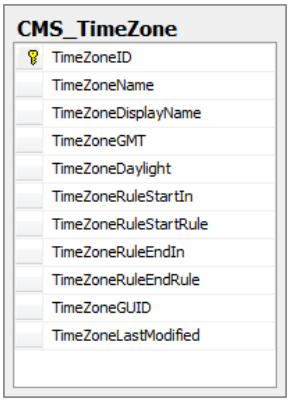
The Displaying correct time in your code topic demonstrates how the correct time can be displayed in transformations and by user controls depending on the current time zone settings.

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.

8.46.7.2 Database tables

The following database tables are used to store information for the time zones module:

- **CMS_TimeZone** - contains records representing time zones.



8.46.7.3 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.

8.46.7.4 API examples

8.46.7.4.1 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;
```

8.46.7.4.2 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;
}
```

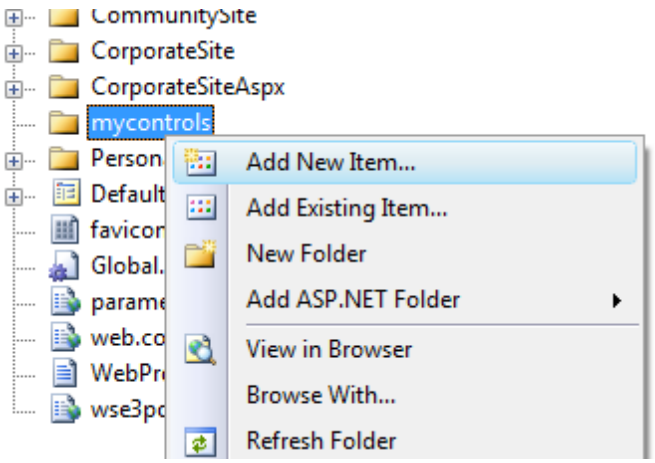
8.46.7.5 Displaying correct time in your code

The following methods can be used in transformation code to display the correct time according to time zone settings.

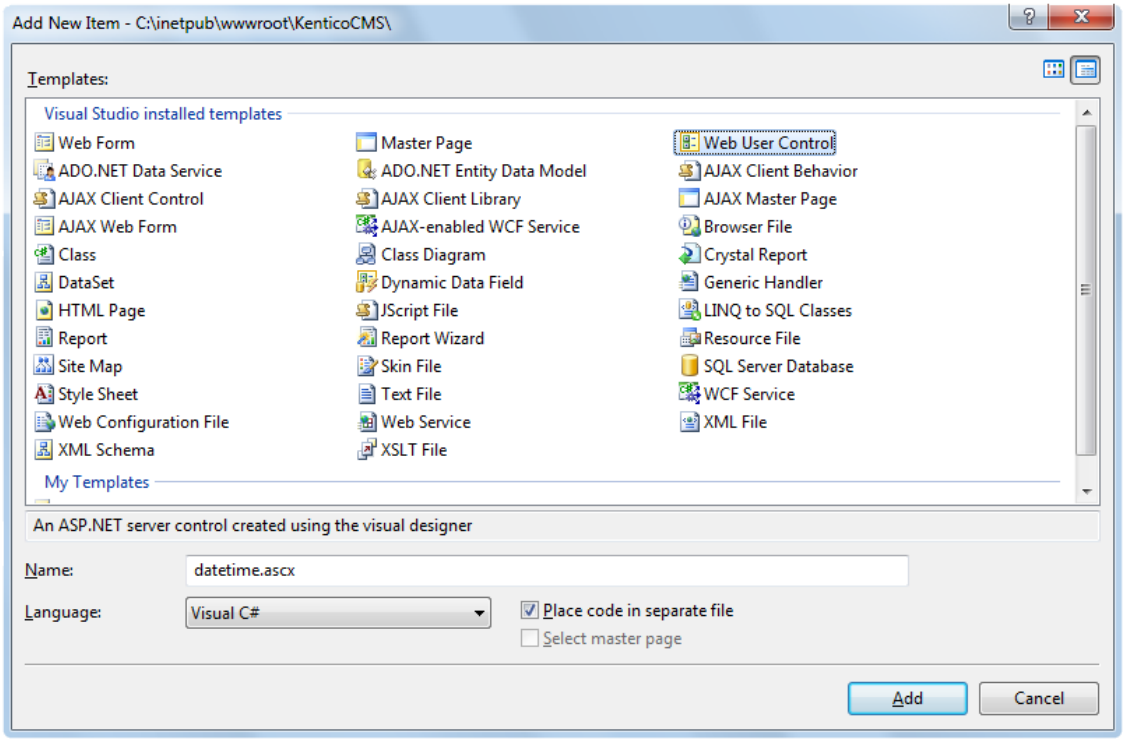
<%# GetDateTime (DateTime.Now) %>	Returns date-time value according to web part time zone settings.
<%# GetSiteDateTime (DateTime.Now) %>	Returns date-time according to site time zone settings.
<%# GetUserDateTime (DateTime.Now) %>	Returns date-time according to the current user's time zone settings.
<%# 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:

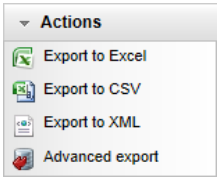
- **User control virtual path:** *~/mycontrols/datetime.ascx*
- **Time zone**: *Custom*
- **Custom time zone**: any time zone of your choice

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.

8.47 UI data export

8.47.1 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.



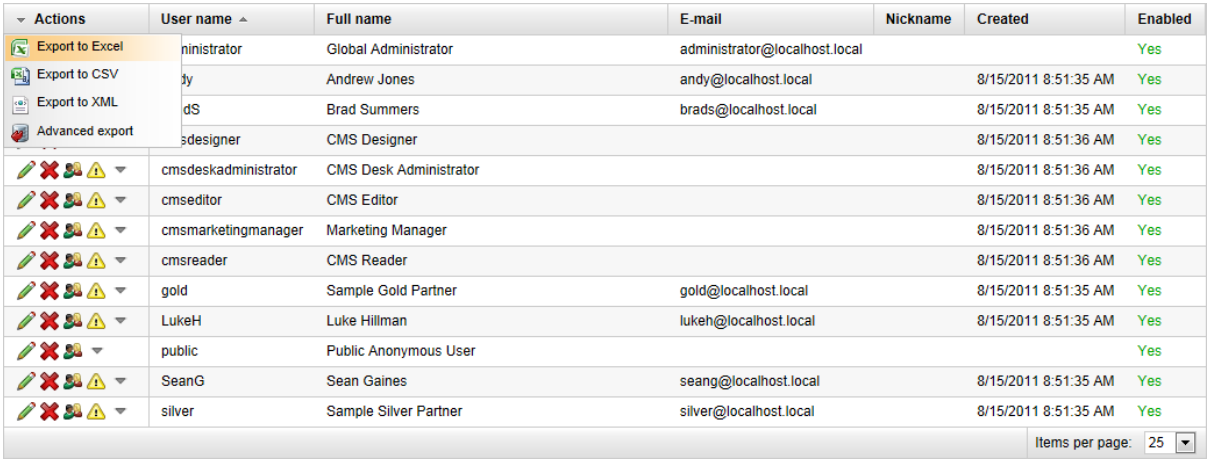
The Exporting UI data topic in this chapter provides you with information on the basics of this functionality. Export to the three formats mentioned above can either be performed in a single click with pre-defined options, or using the **Advanced export** dialog, which allows detailed configuration of 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.

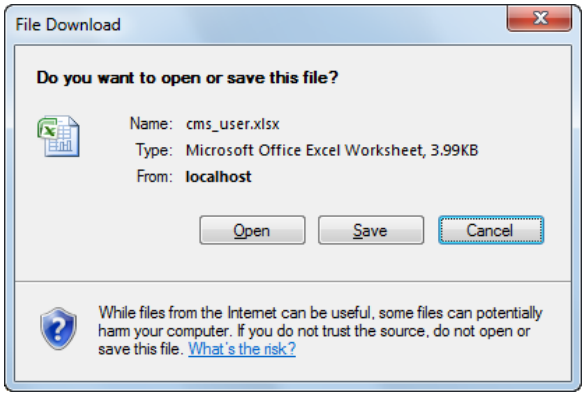
8.47.2 Exporting UI data

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.



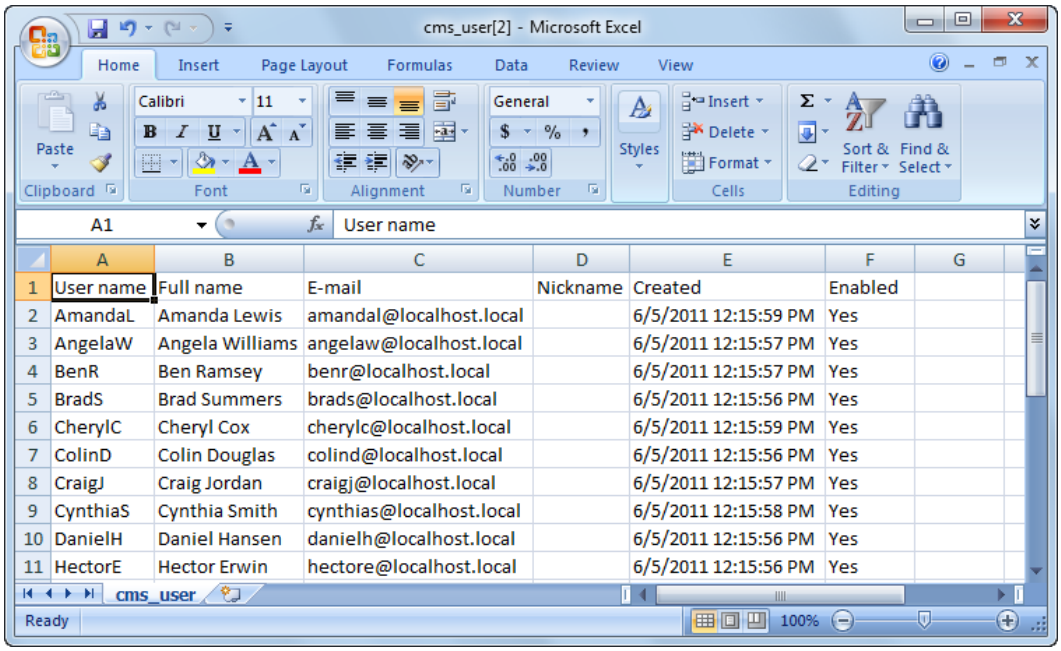
Export to Excel

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
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

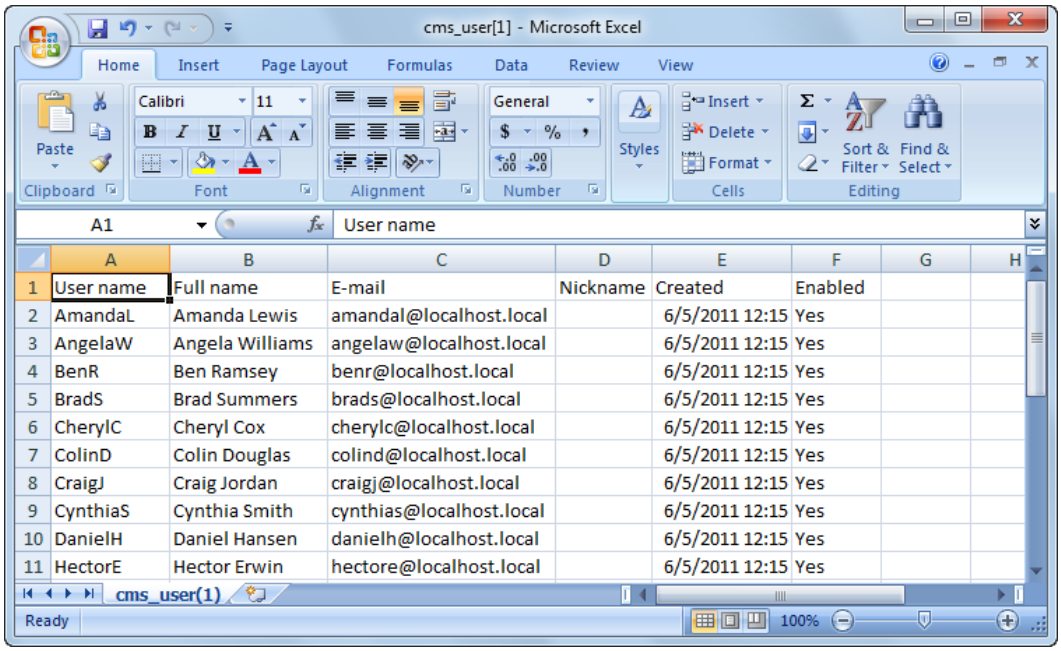
CSV is an abbreviation for Comma-separated values. It is a file format that stores tabular data in text form — each line represents one row of data, while particular values (columns) in each row are separated by a comma (,) or a semicolon (;). If a column value contains the delimiter, the whole value 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		

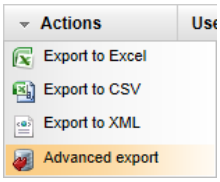
Export to XML

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>
```

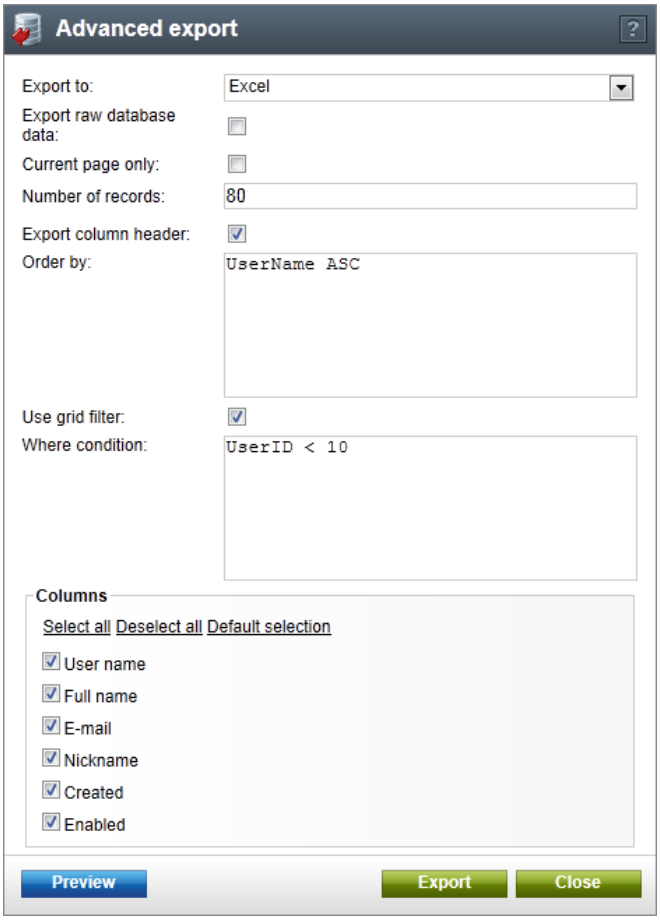
8.47.3 Advanced export

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.

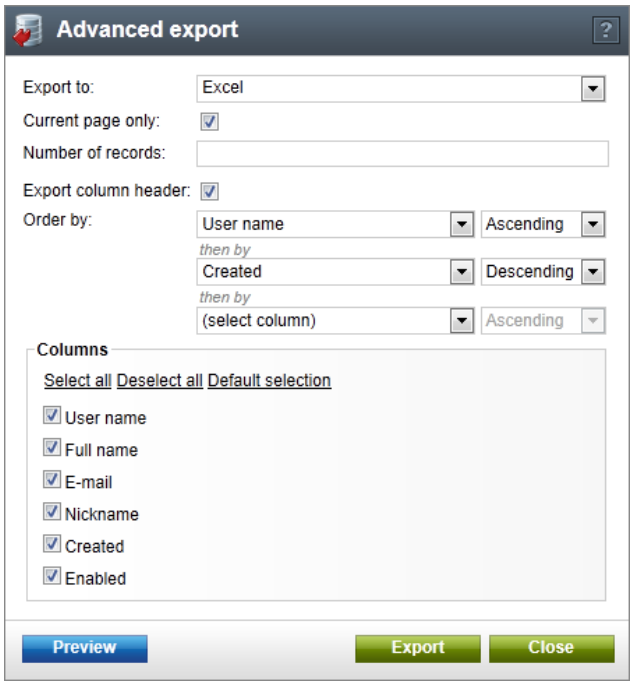
	• **CSV** - exports data to a CSV file. • **XML** - exports data to an XML file.
Delimiter	Displayed only when export to CSV is selected. Determines the value separation character used by the CSV file. The character can either be a comma (,) or a semicolon (;). See the CSV delimiters topic for more details.
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.

8.47.4 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.

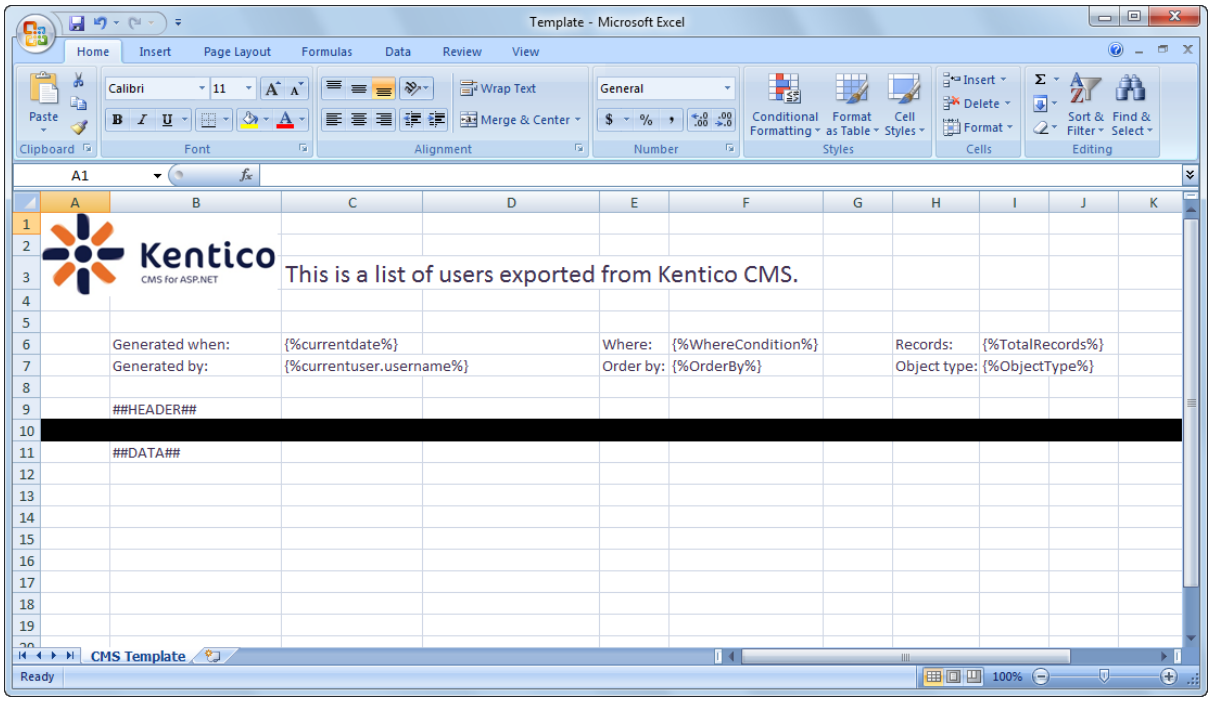
Template format

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:

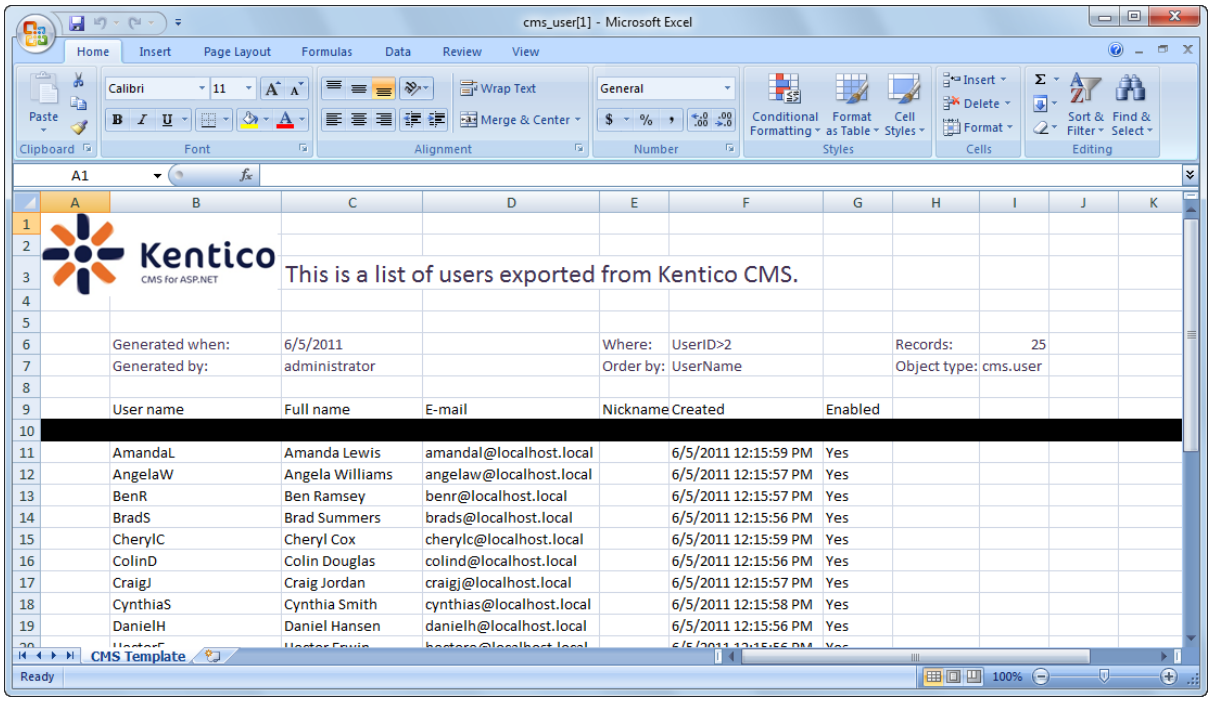
- **##HEADER##** - replaced with the header row. When advanced export is used, it is only replaced if the **Export header row** option is enabled.
- **##DATA##** - replaced with the actual exported data.
- **##TABLE##** - replaced with both the header row and the exported data.
- **{%WhereCondition%}** - 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.

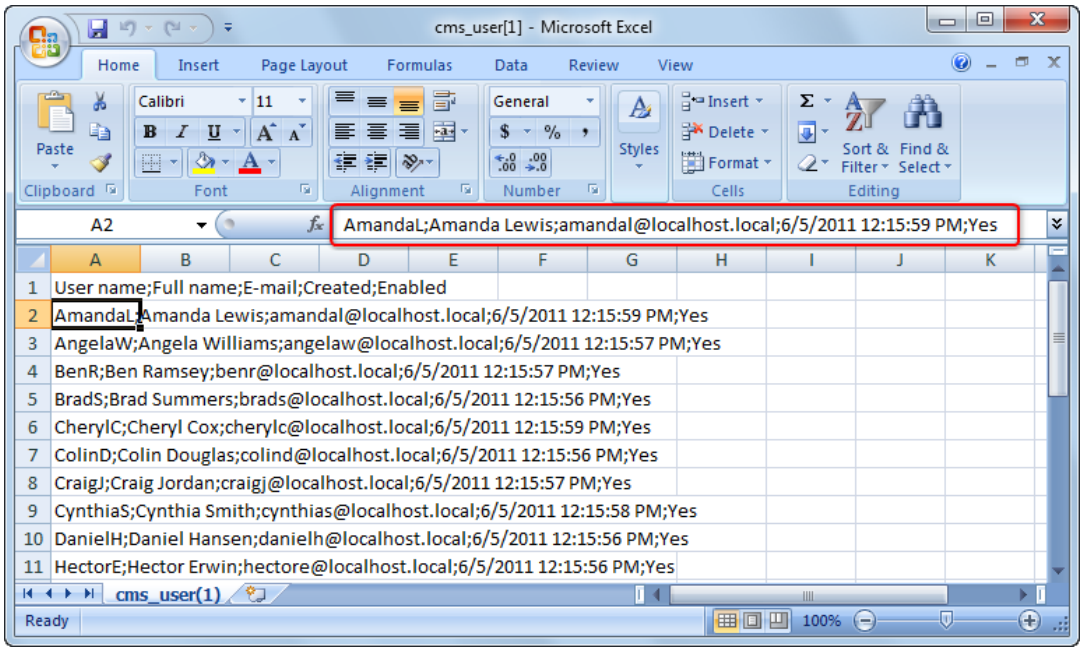


8.47.5 CSV delimiters

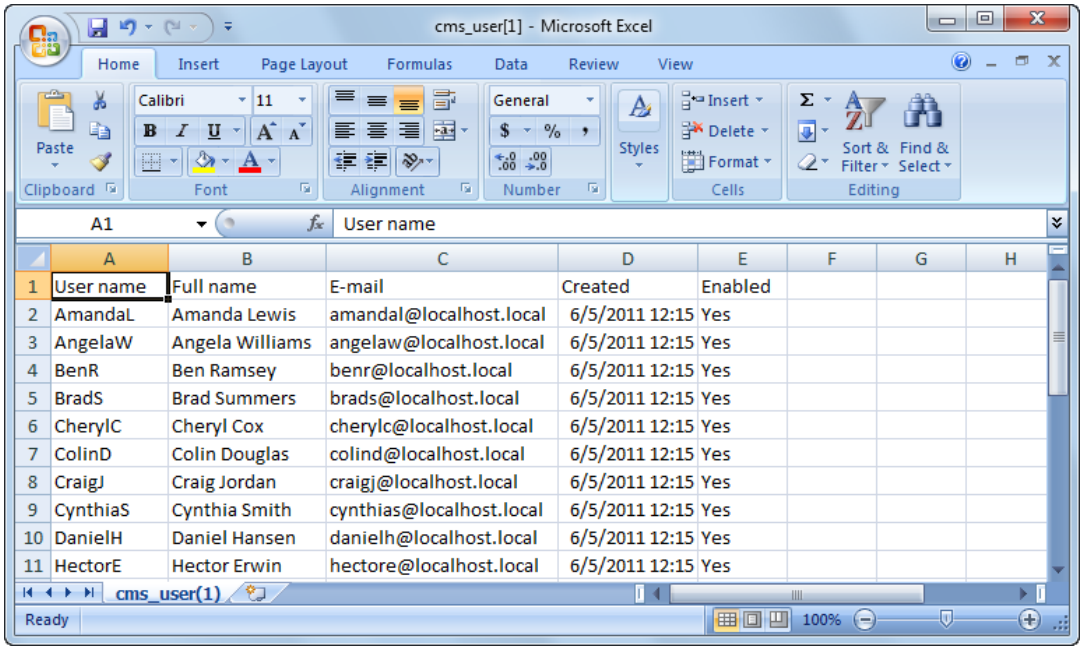
CSV is an abbreviation for Comma-separated values. It is a file format that stores tabular data in text form — each line represents one row of data, while particular values (columns) in each row are separated by a comma (,) or a semicolon (;). The comma is used as a column delimiter by default if you 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.



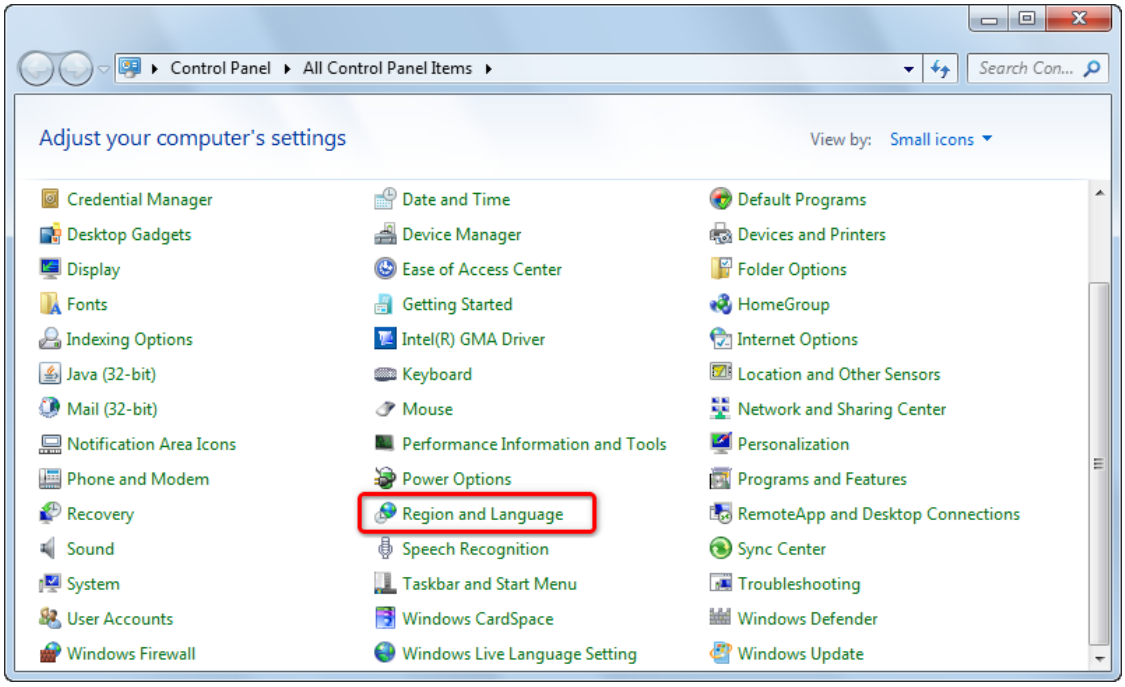
If the appropriate delimiter is used, data from each column will be displayed in individual cells as expected.



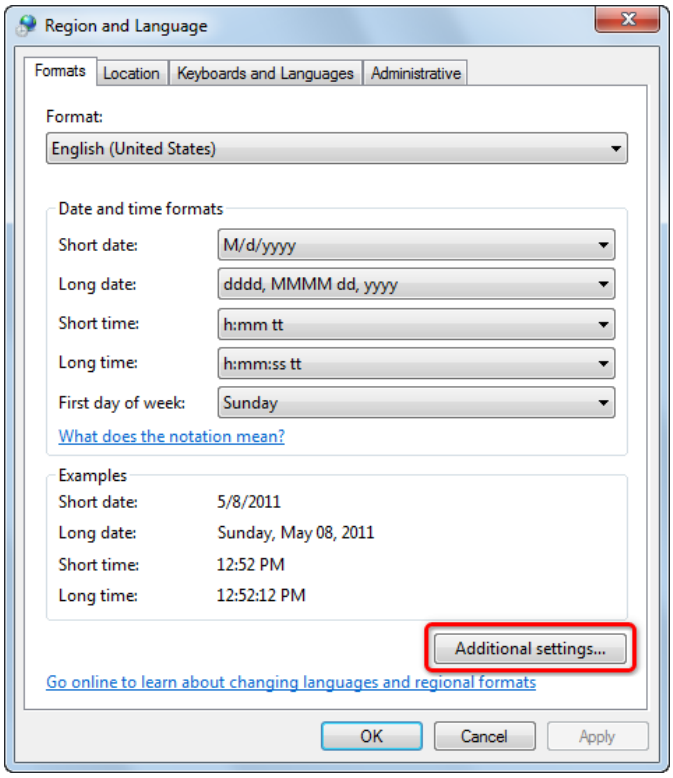
Delimiter settings on operating system level

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:

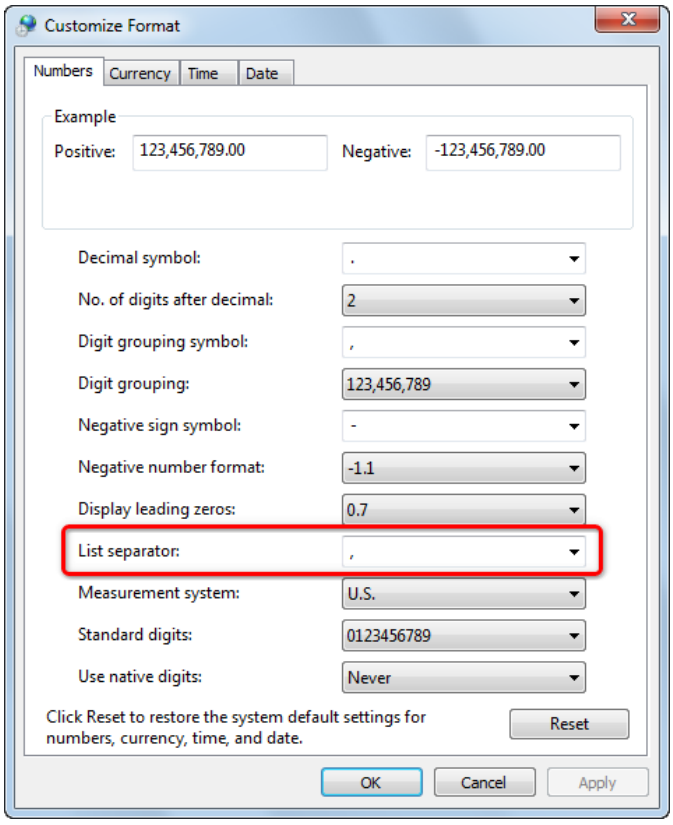
1. Go to **Start menu -> Control Panel** and open the **Region and Language** settings category.



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.



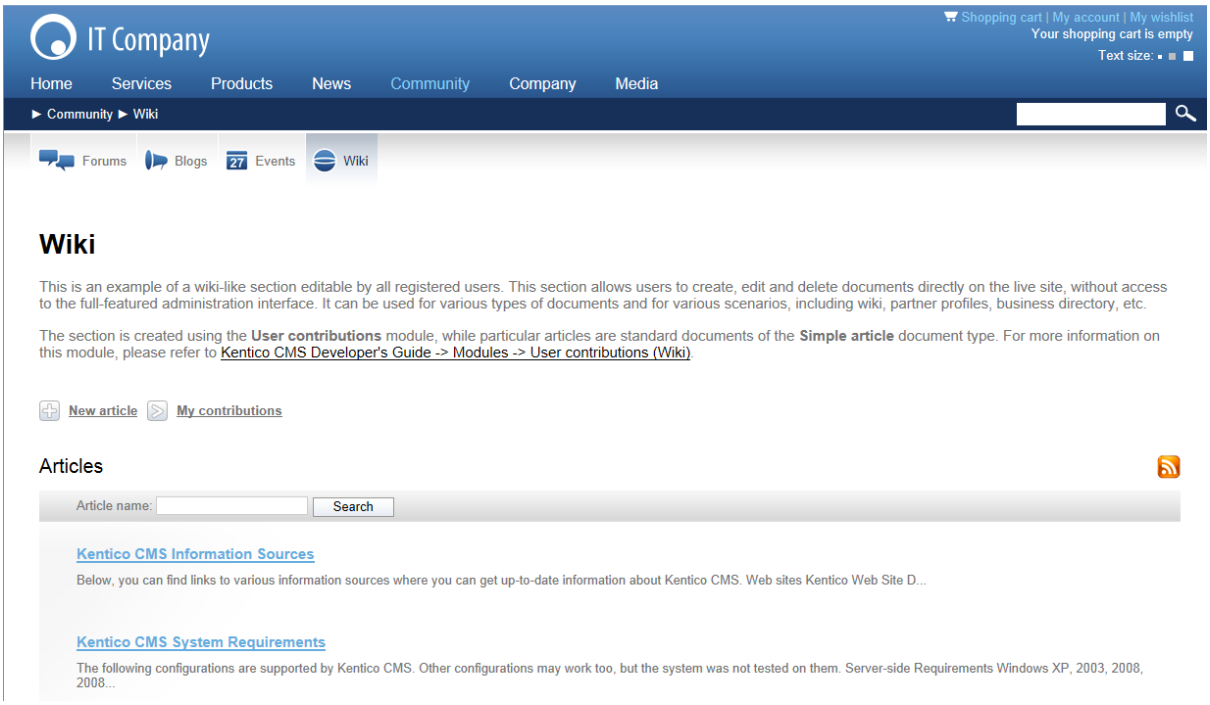
Important!

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.

8.48 User contributions (Wiki)

8.48.1 Overview

The User contributions module allows you to create 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.



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.

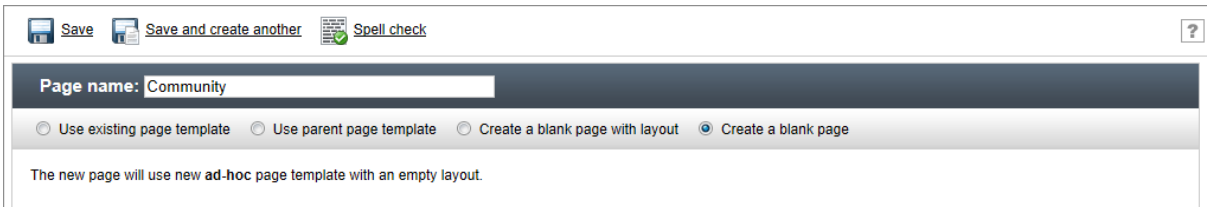
8.48.2 Example: Publishing community news

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

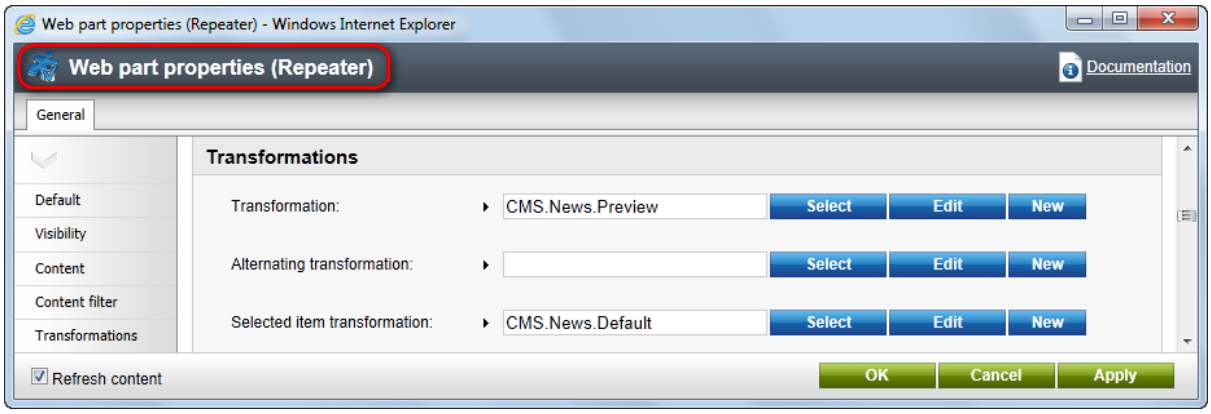
1. Sign in to **CMS Desk** as the administrator, go to the **Content** section and click **Examples** in the content tree. Click **New** and choose to create a new **Page (menu item)**. Enter the name **Community** and choose the **Create a blank page** option.



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



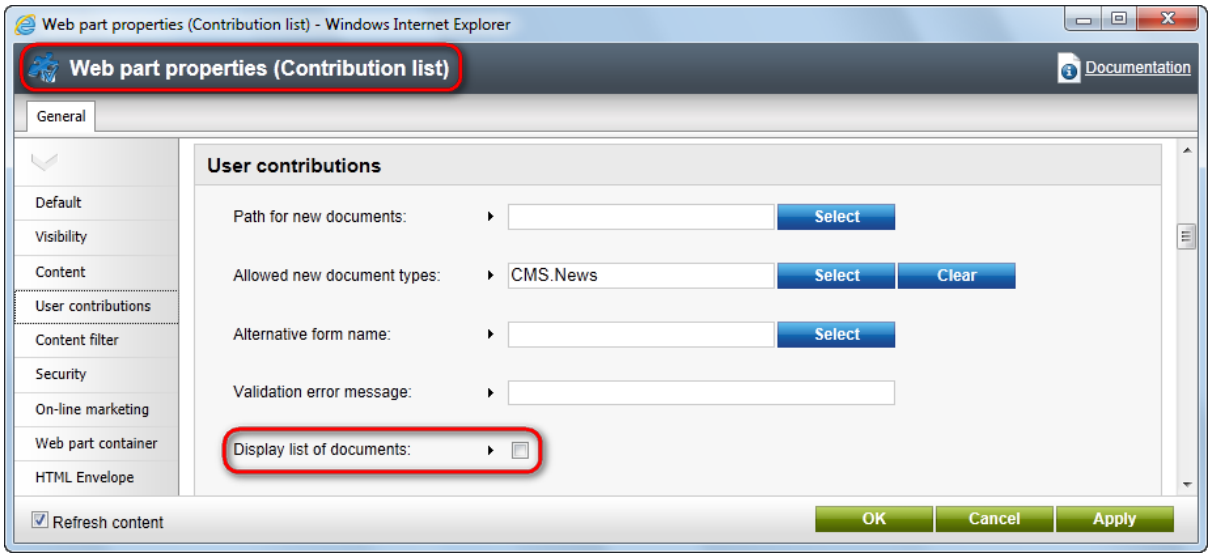
3. Click **OK**.

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.



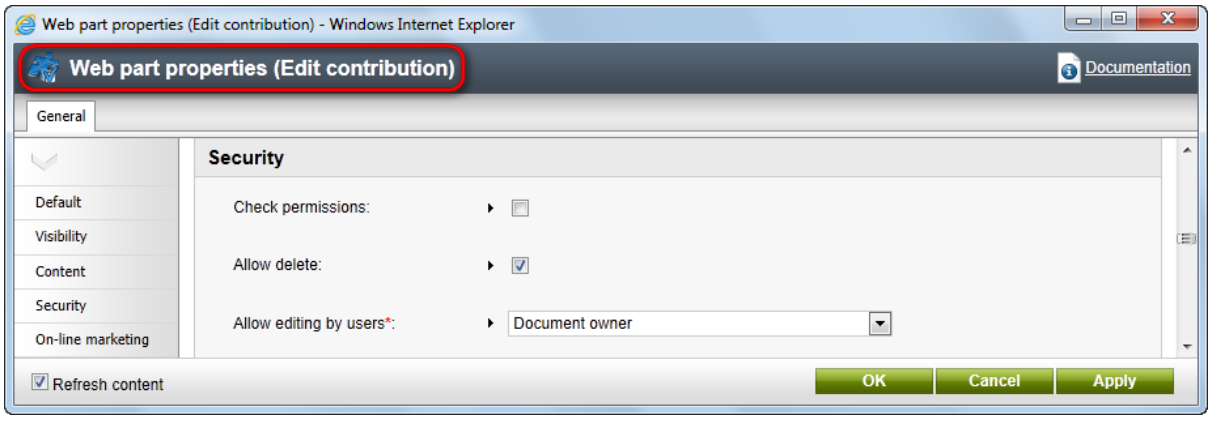
2. Click **OK**.

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

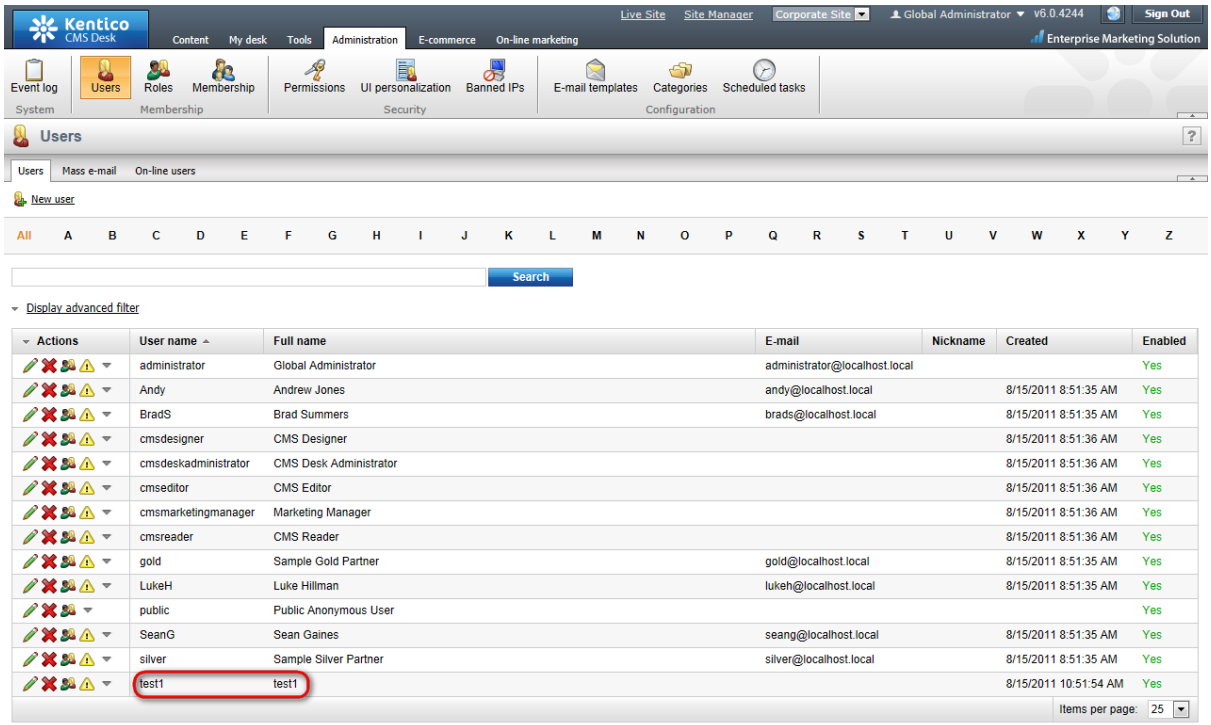
Please note that the **Edit contribution** web part is missing the **Allow insert/edit** properties.



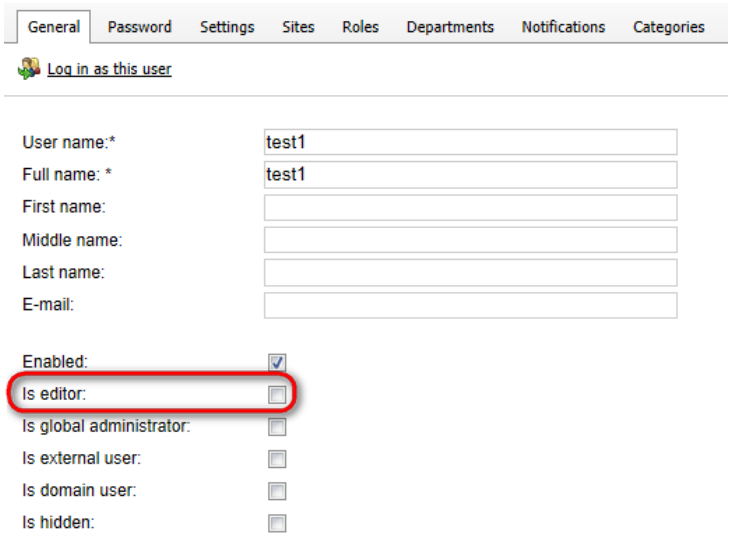
2. Click **OK**.

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.

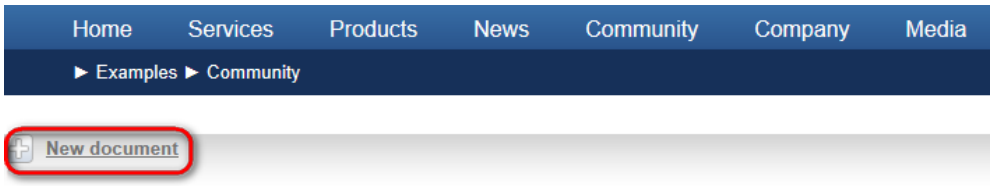


Click **OK** to save the changes.

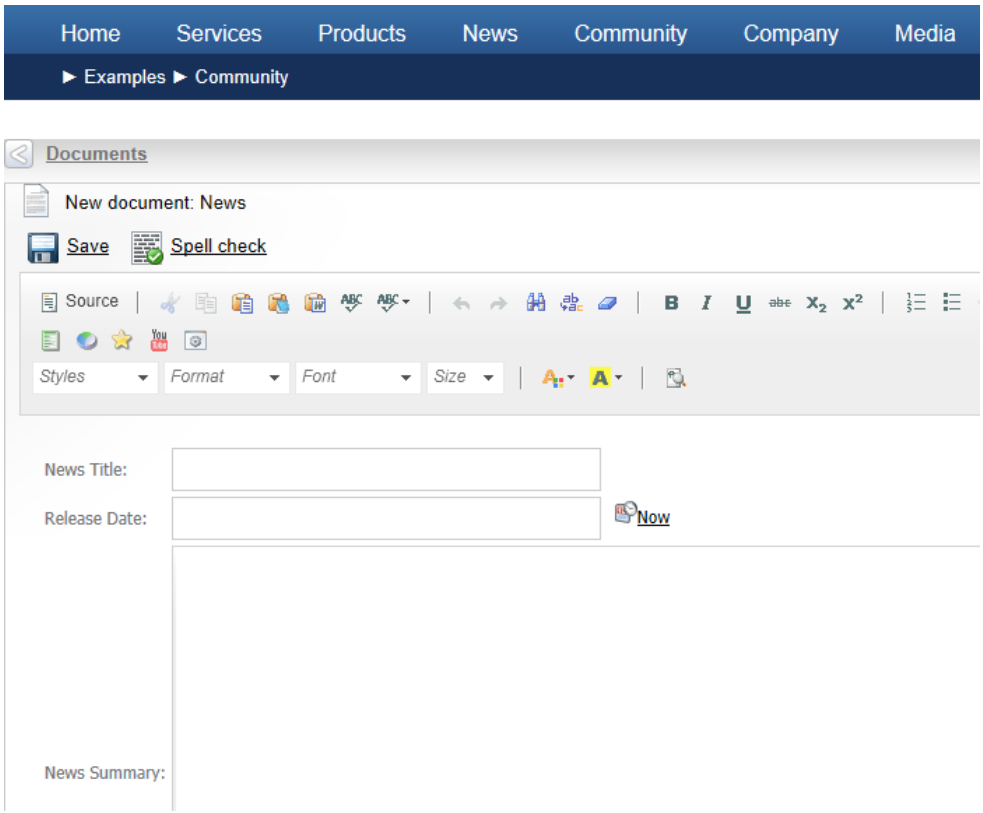
Creating your first news item

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

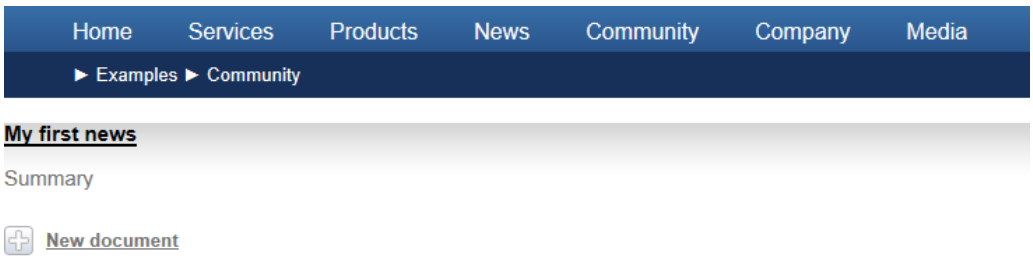
New document link:



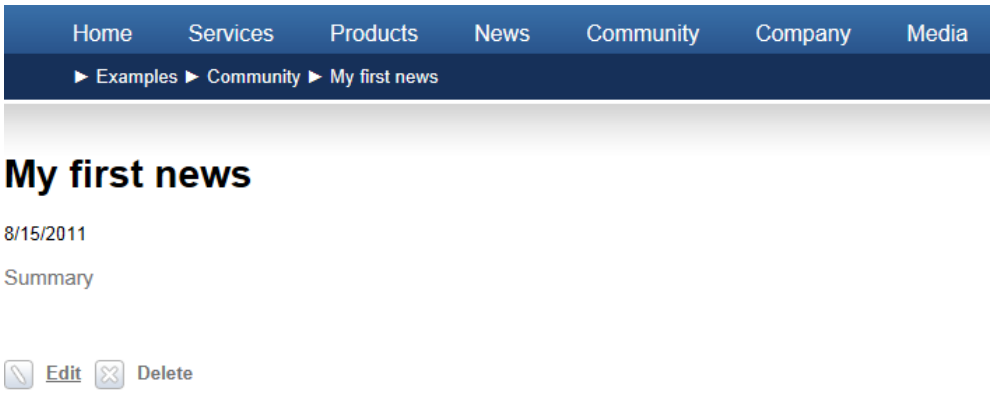
2. Click this link and you will be displayed with the news editing form:



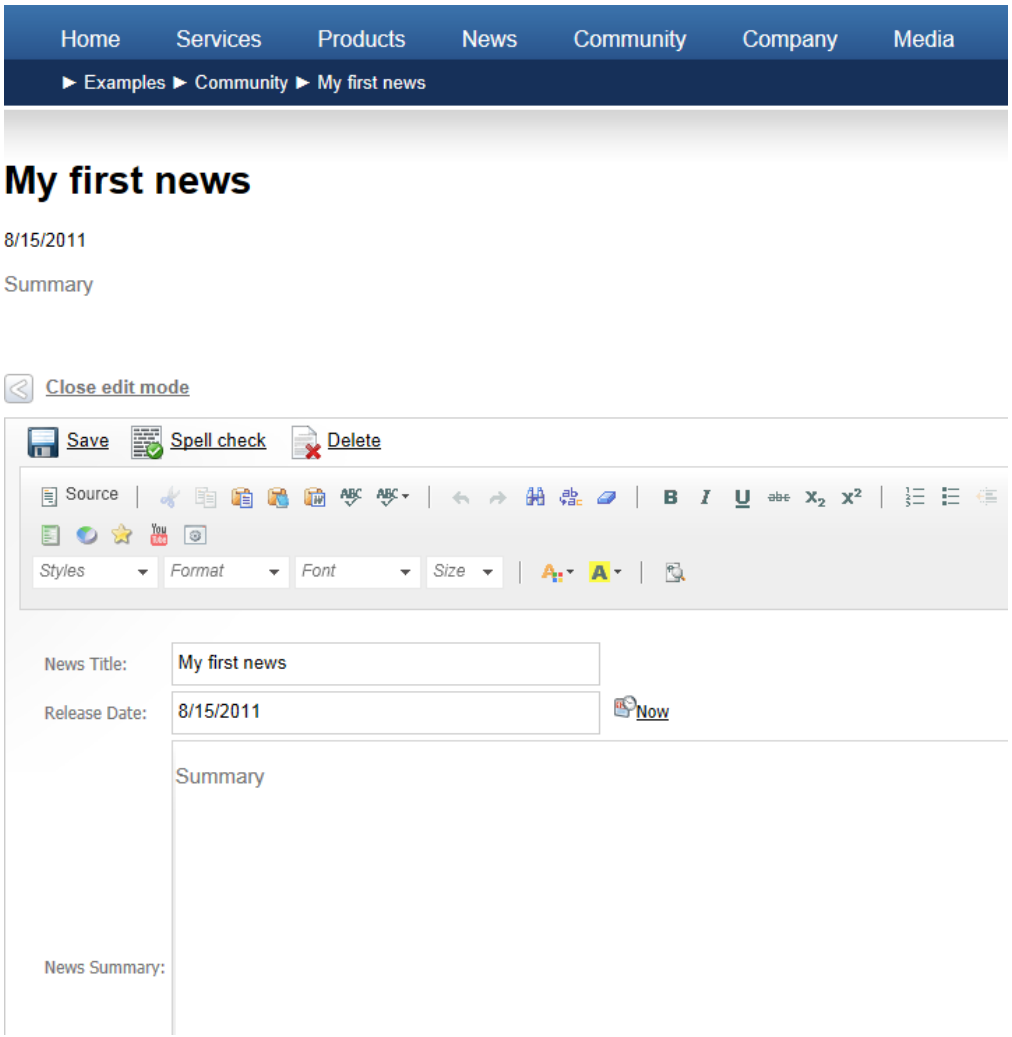
3. Enter some content and click **Save**. Now select **Examples -> Community** from the main menu at the top. You will see the **Community** page with your first news item:



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.

8.48.3 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.

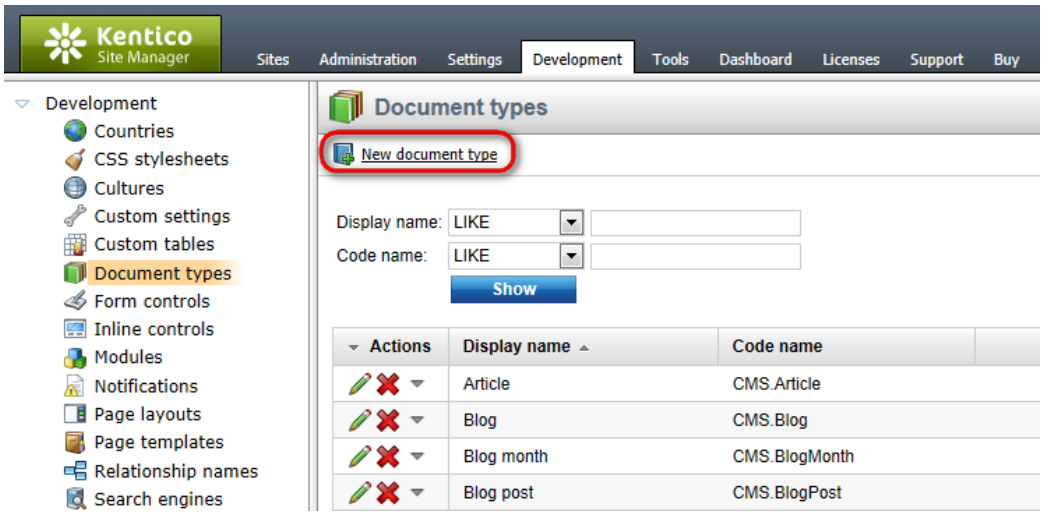
- Create the partner document type
- Create the partner list
- Create the partner user account
- Create the partner profile document
- Create the partner profile editing page
- Test the profile editing page

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.

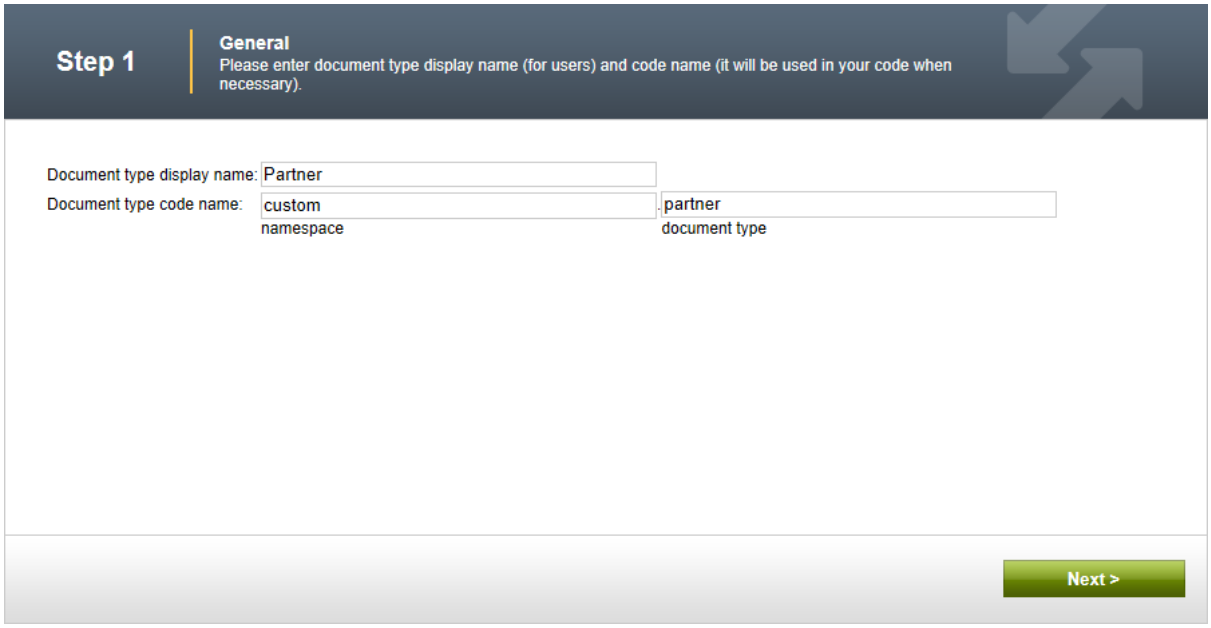
Before you create the page, it is necessary that you create a new document type **Partner**. Sign in to **Site Manager** and go to **Development -> Document types** where you need to click the **New document type** link.



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

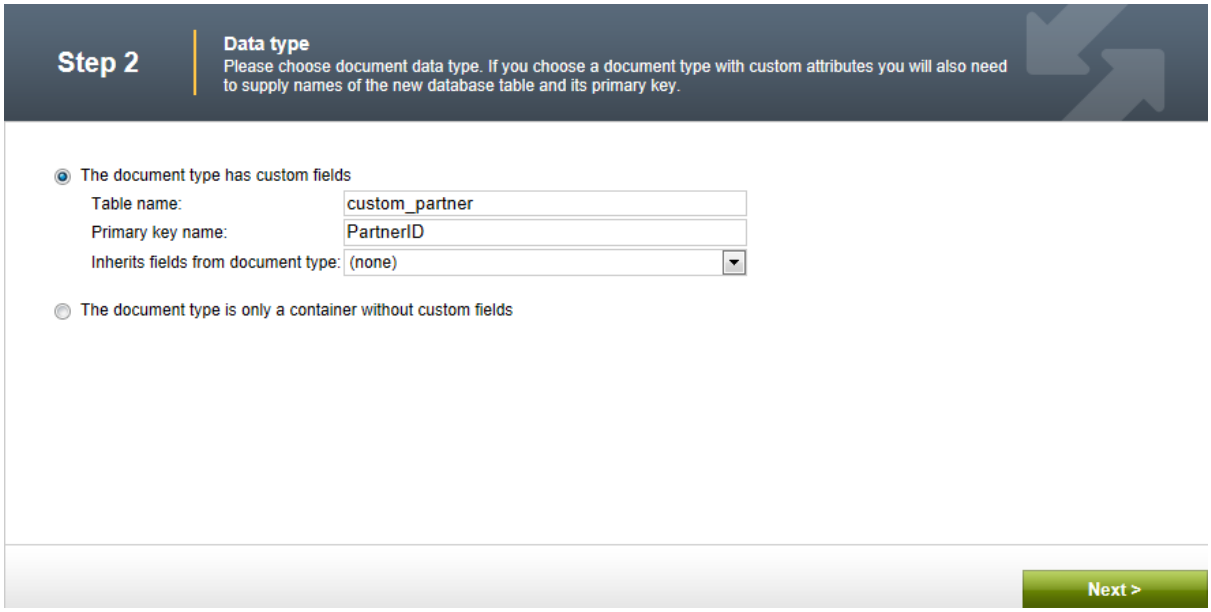


Please note that *custom* in the **Document type code name** field is your namespace to distinguish your document types from system types that use the *cms* namespace. You will use this value in web part properties later.

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**.

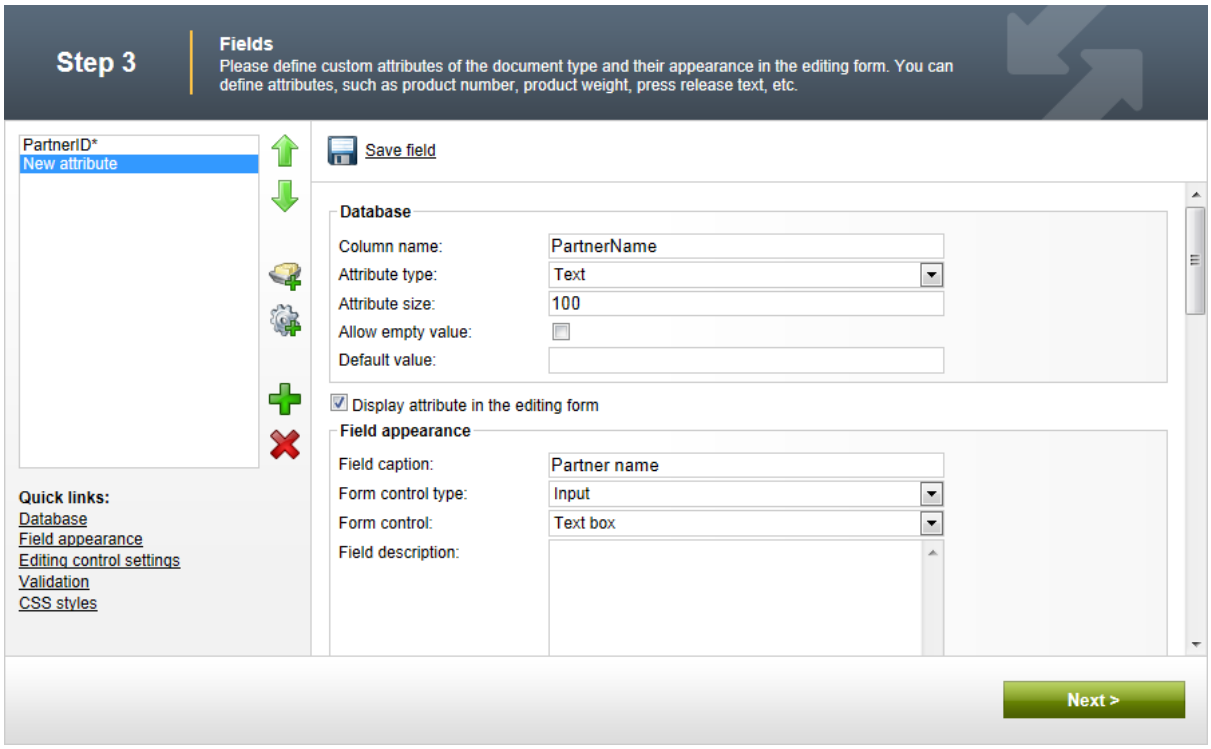
Step 3: Fields

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)

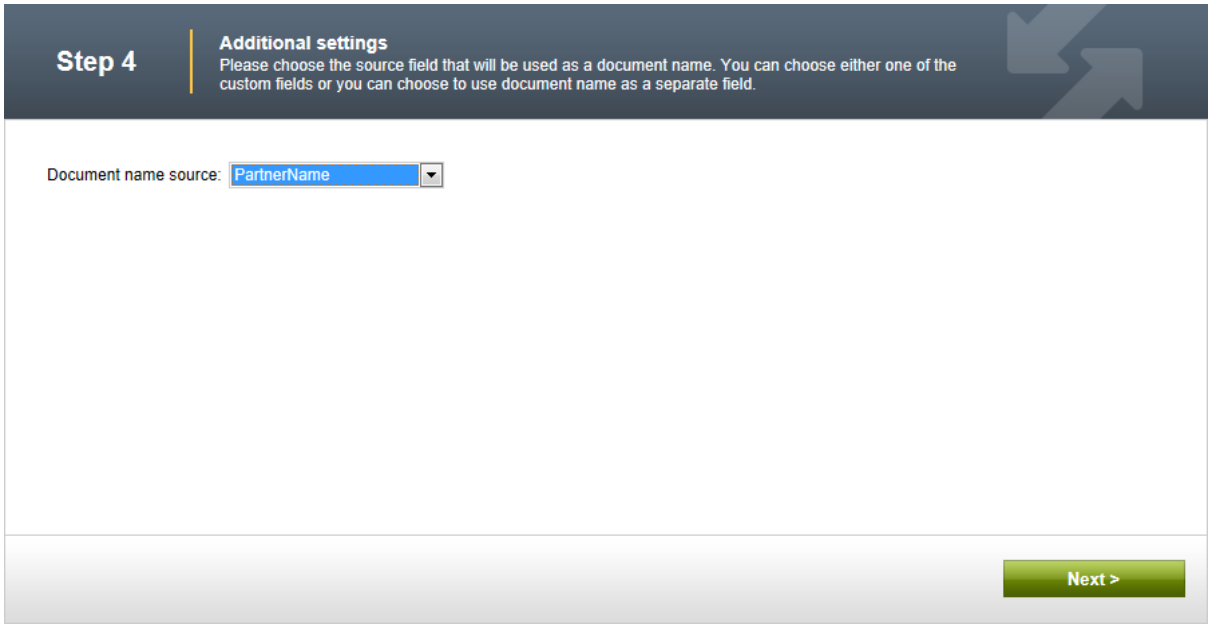
When you have defined both the fields, click **Next** again.



Step 4: Additional Settings

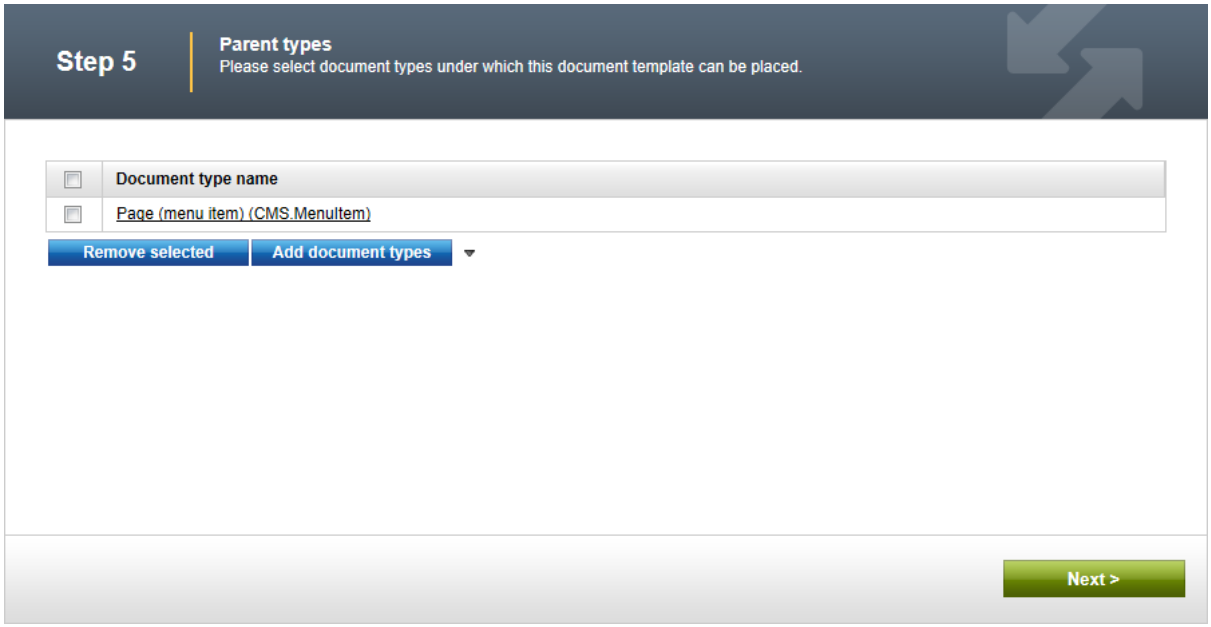
Now you need to choose the field that will be used as document name. Choose the *PartnerName* field option from the **Document name source** drop-down list and click **Next**.

- **Document name source:** PartnerName



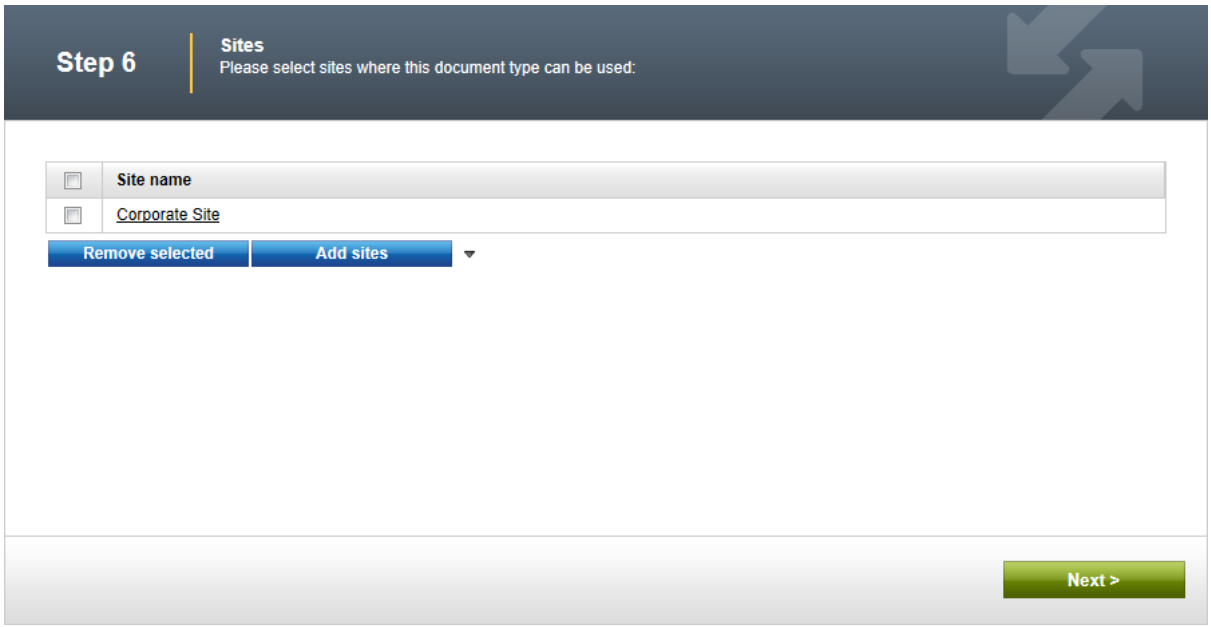
Step 5: Parent types

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.



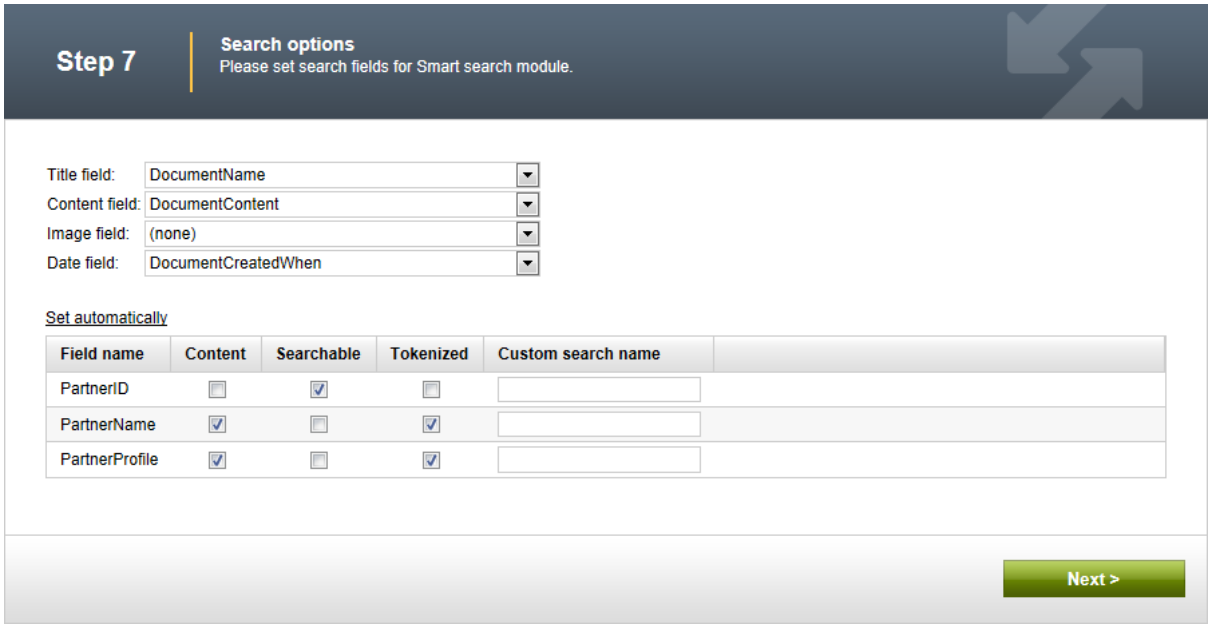
Step 6: Sites

Here you need to choose which websites will use this document type. Choose your current website and click **Next**.



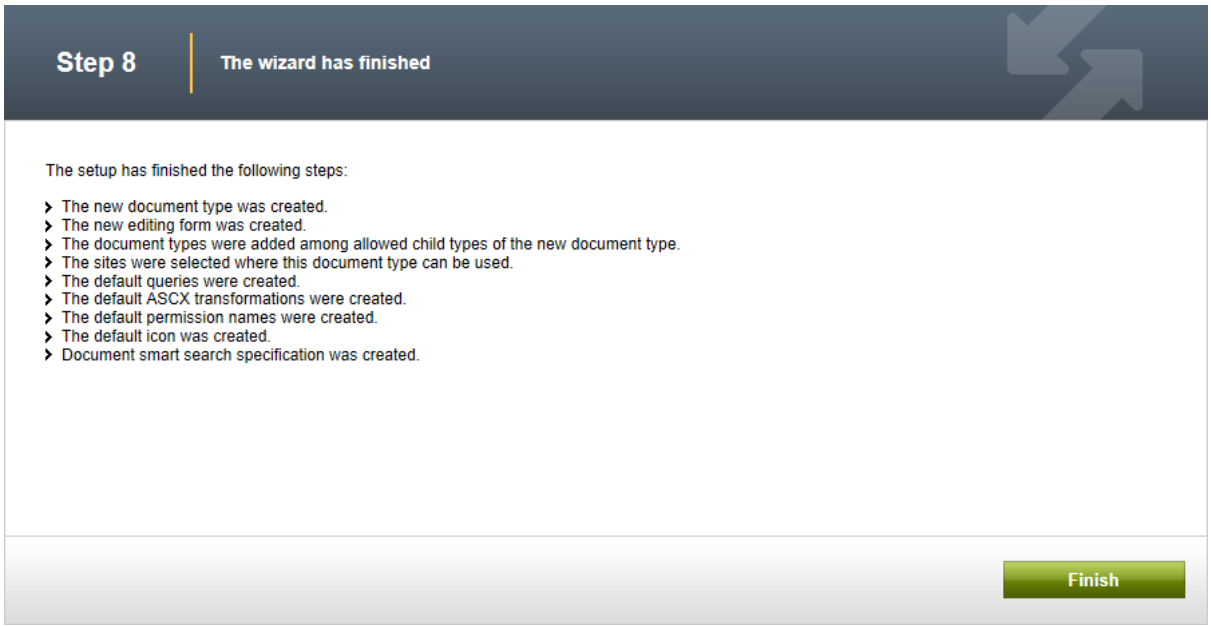
Step 7: Search options

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.



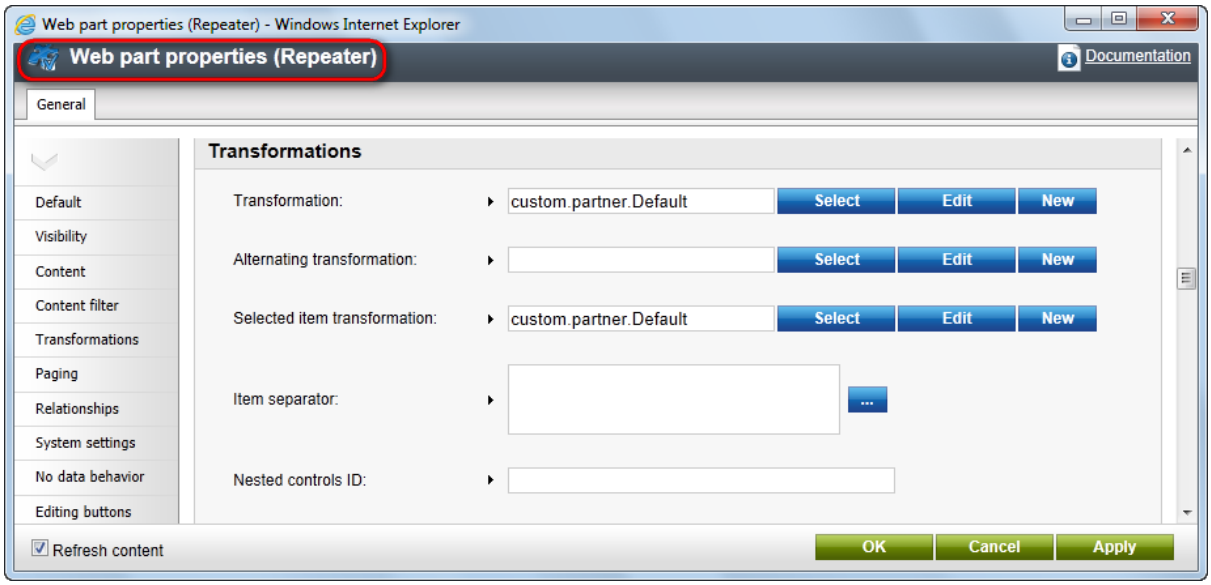
Creating the partner list

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

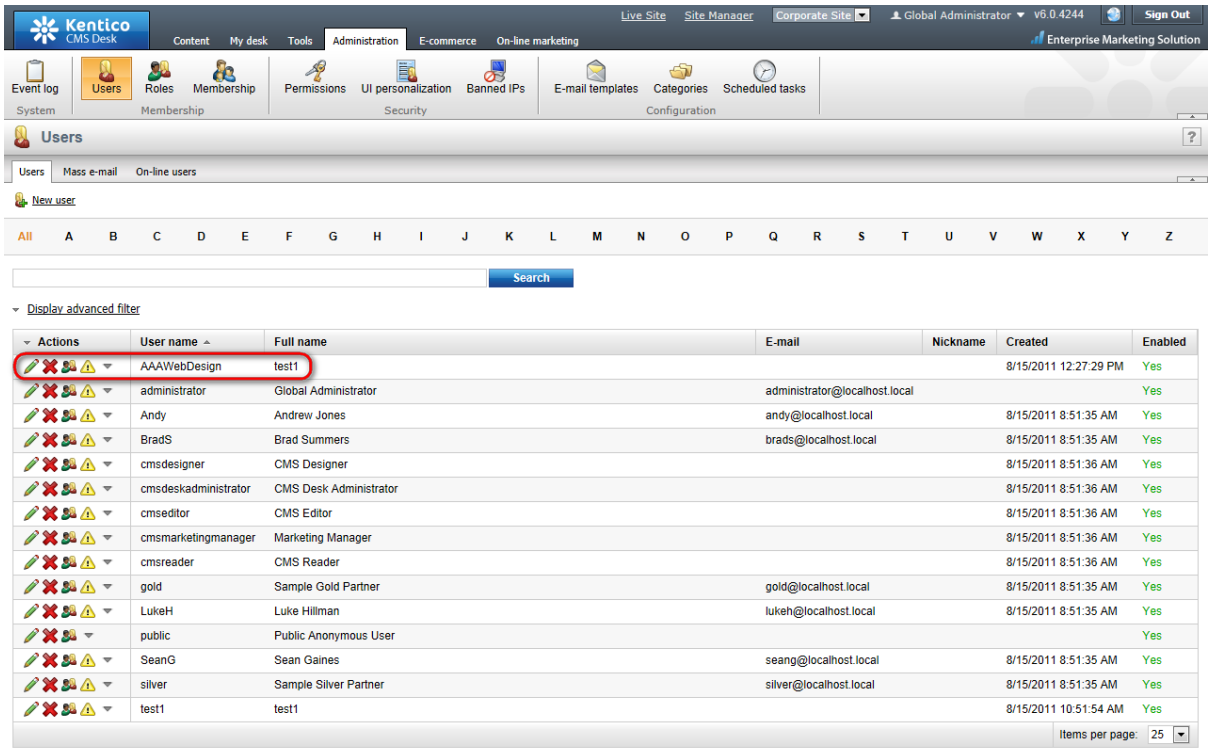


Please note that the default transformation generated by the system will be used. You can later modify the transformation so that it meets your design and layout requirements.

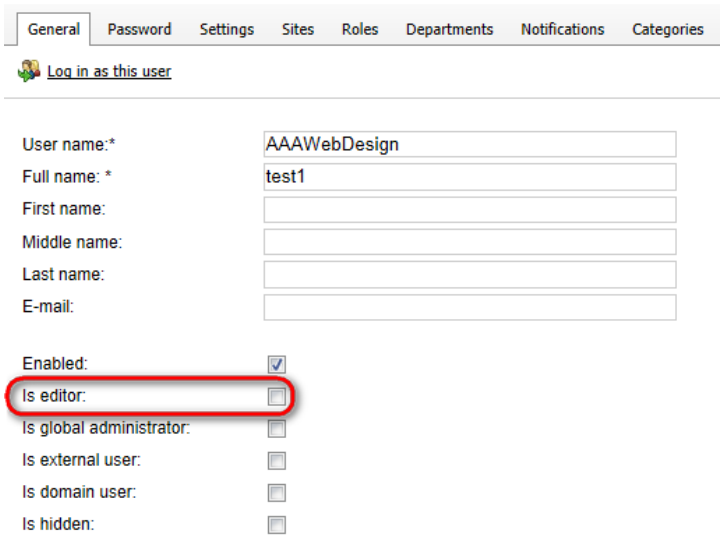
3. Click **OK**. Switch to the **Page** view and you will see an empty page now.

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**. Set the **Is editor** property to no (unchecked) and assign the user to the **CMS Basic users** role.



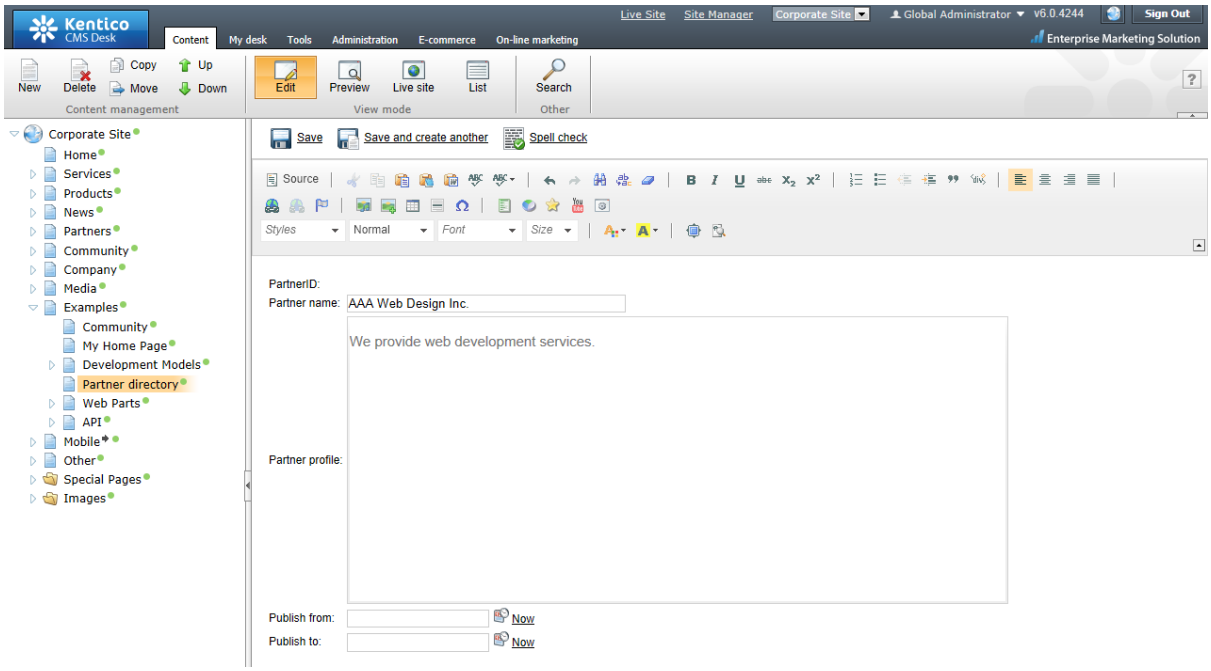
2. For the purpose of this example, choose no (uncheck the box) in the **Is editor** field - **AAA Web Design** is a common site member without access to Kentico CMS Desk. Click **OK** to save the changes.



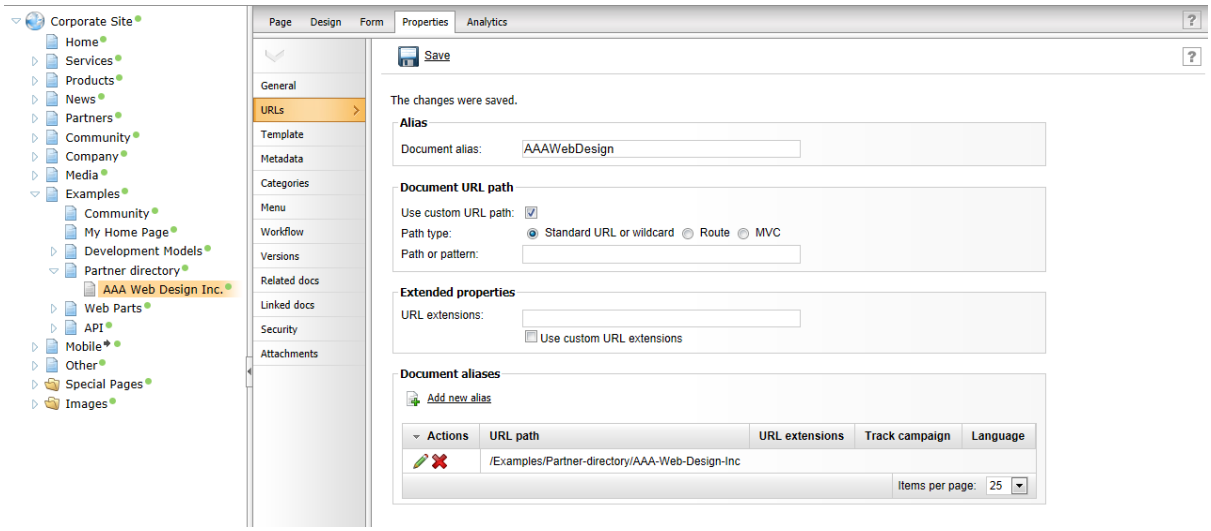
Creating the partner profile document

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:

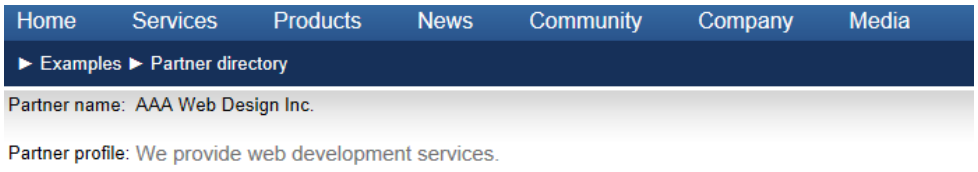
- **Partner name:** AAA Web Design Inc.
- **Partner profile:** We provide web development services.



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:



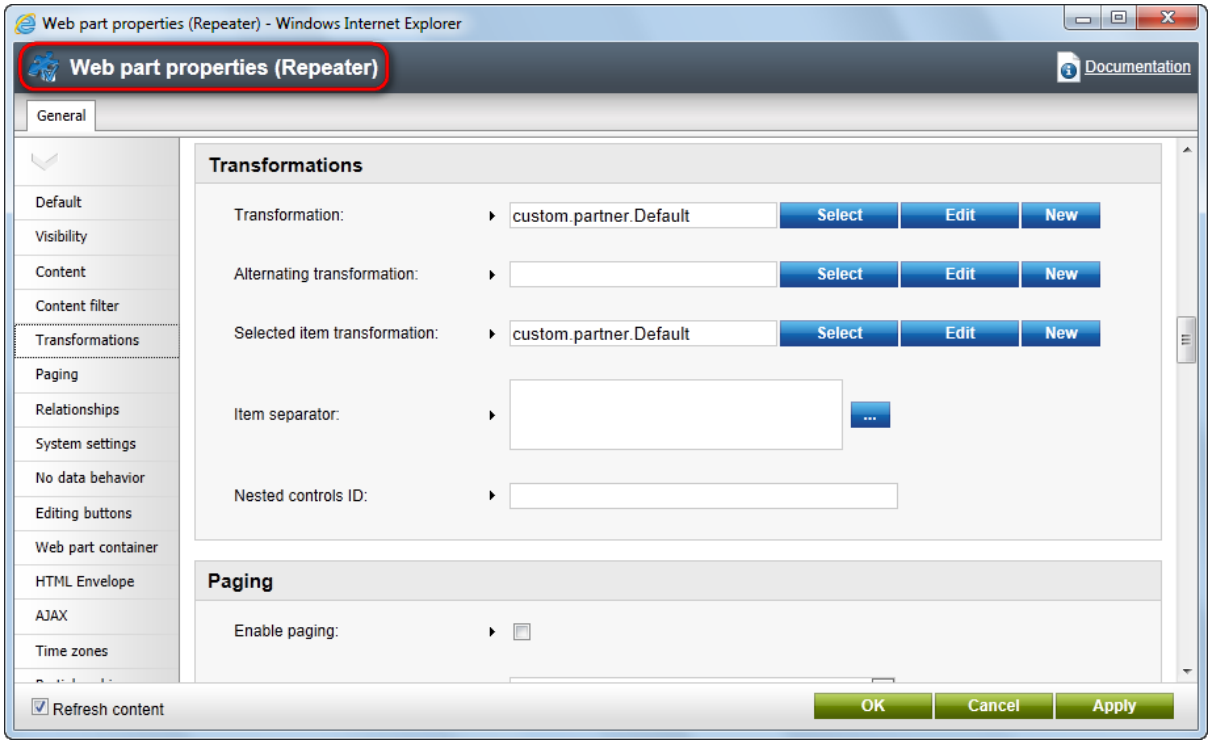
Creating the partner profile editing page

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>

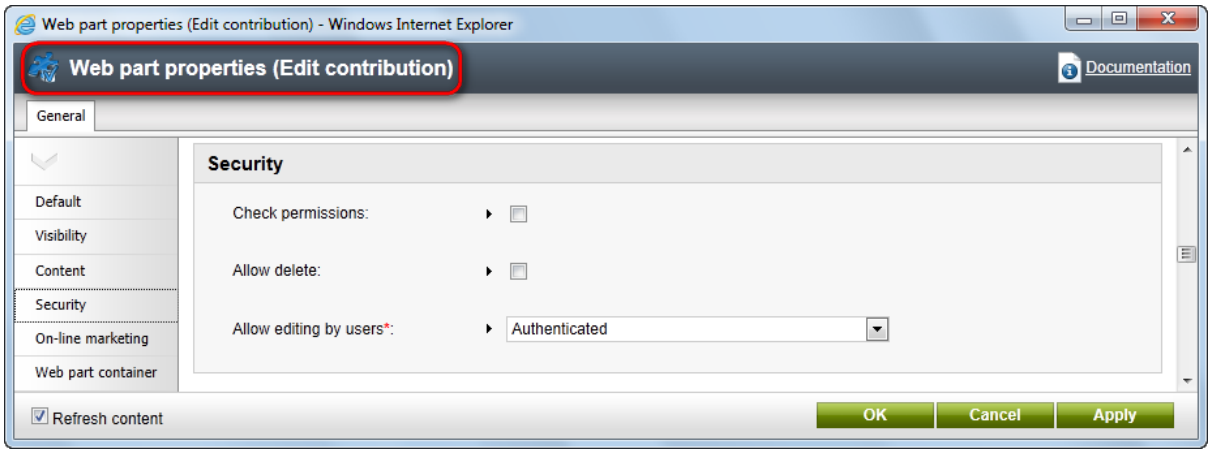


3. Add the **User contributions/Edit contribution** web part, enter the following values and click **OK**.

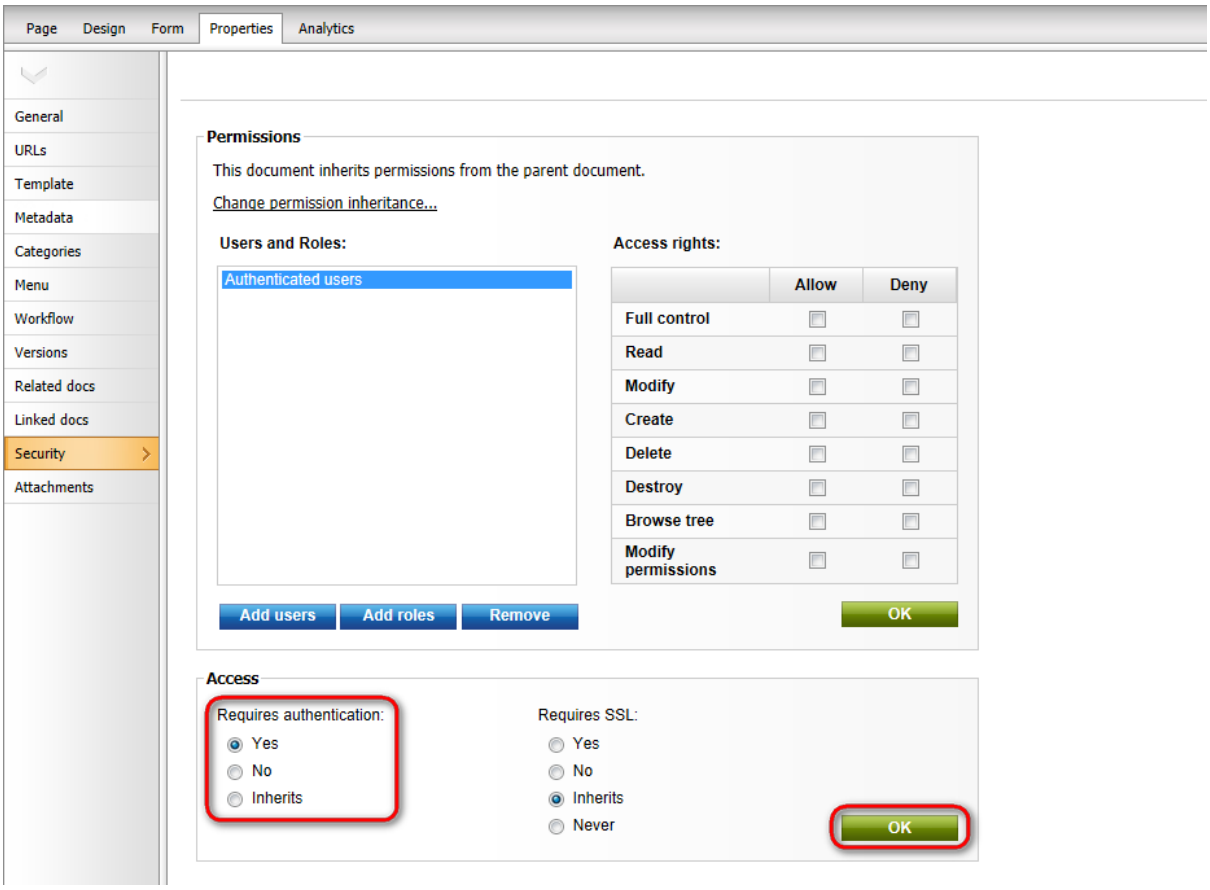
- **Path:** /Examples/Partner-directory/{%UserName%}
  - this ensures that the form edits the profile that matches the current user. The {%UserName%} macro is resolved as the user name of the current site visitor.

- **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.

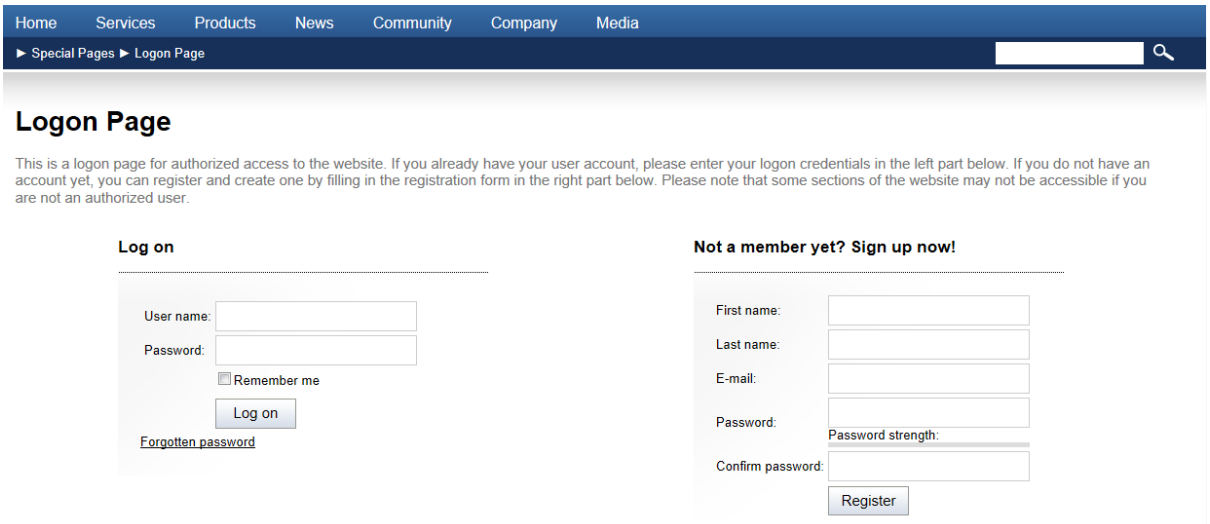


4. As the last step, go to the **Properties -> Security** dialog, set the **Requires authentication** value to **Yes** and click **OK** - the reason is that you want to allow access to this page only to authenticated partners so that you know their user name and can display their profile.

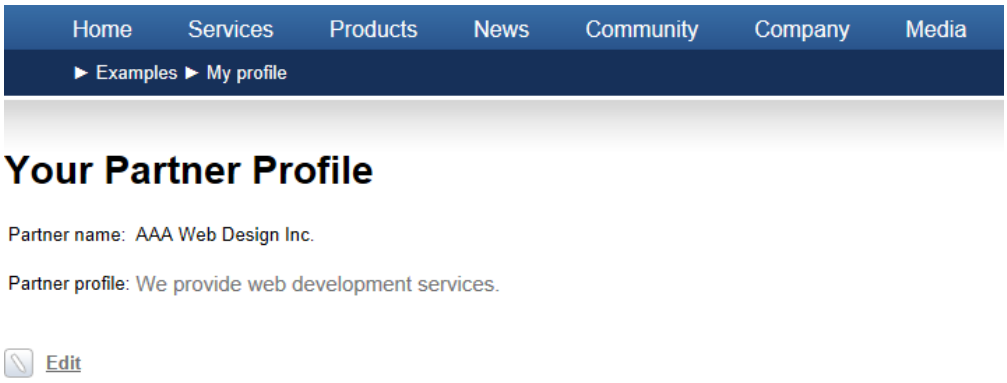


Testing the profile editing page

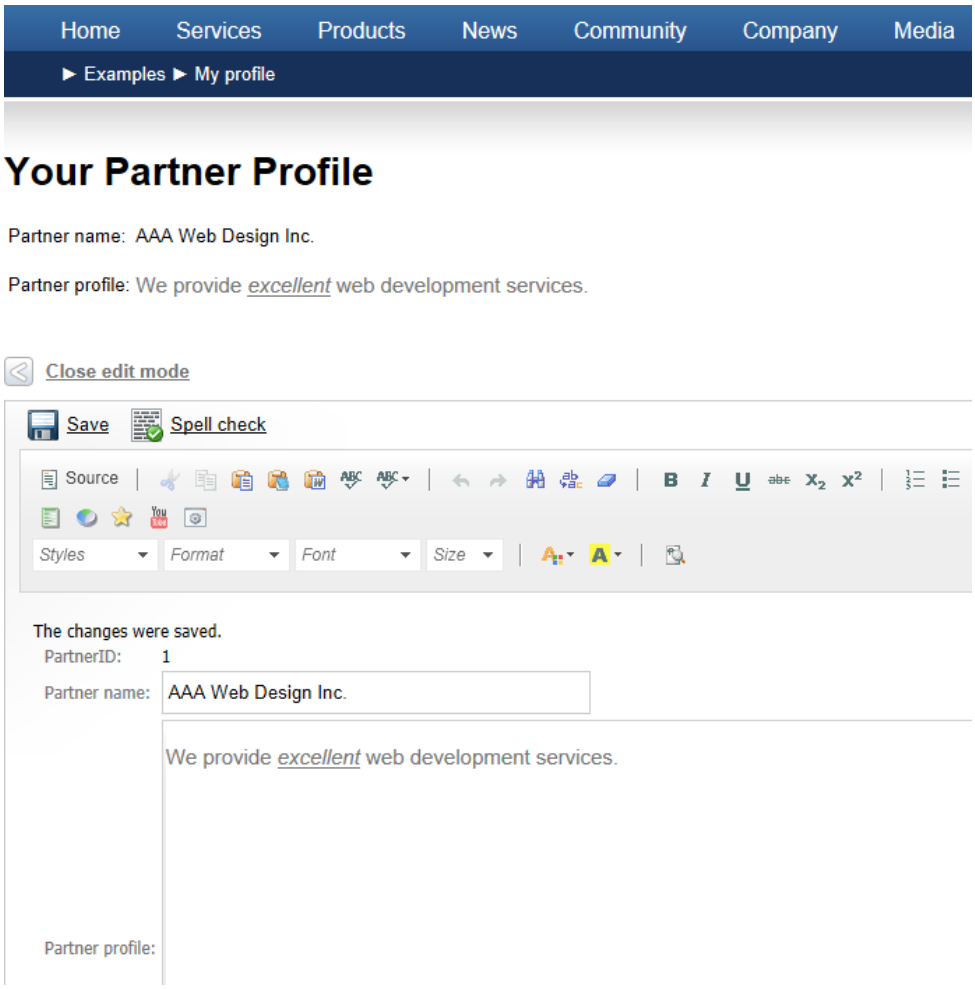
1. Sign out and go to **Examples -> My profile**. You will be displayed with logon form.



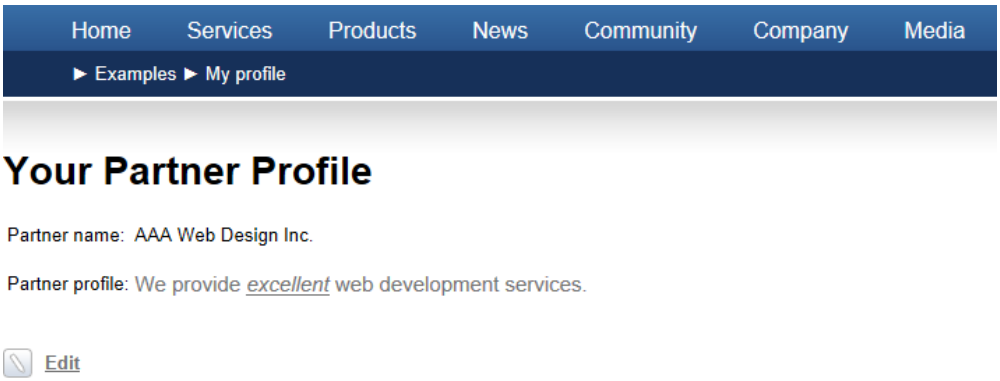
2. Sign in as user **AAAWebDesign** and you will see your partner profile with the **Edit** link.



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

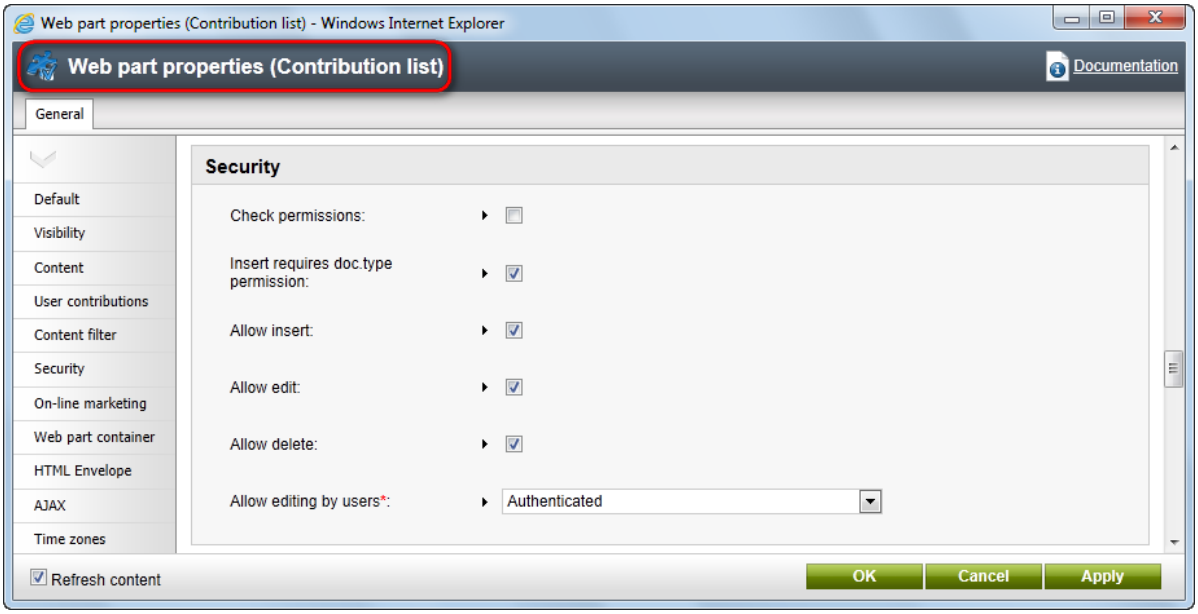
If the user name/document alias path mapping does not fit your needs or if you need to assign multiple users to the same partner profile, you can use another option: Go to **Site Manager -> Development -> System tables -> User** and create a custom text field **PartnerProfileAliasPath**. You will use 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.

8.48.4 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



8.48.5 User contributions and API

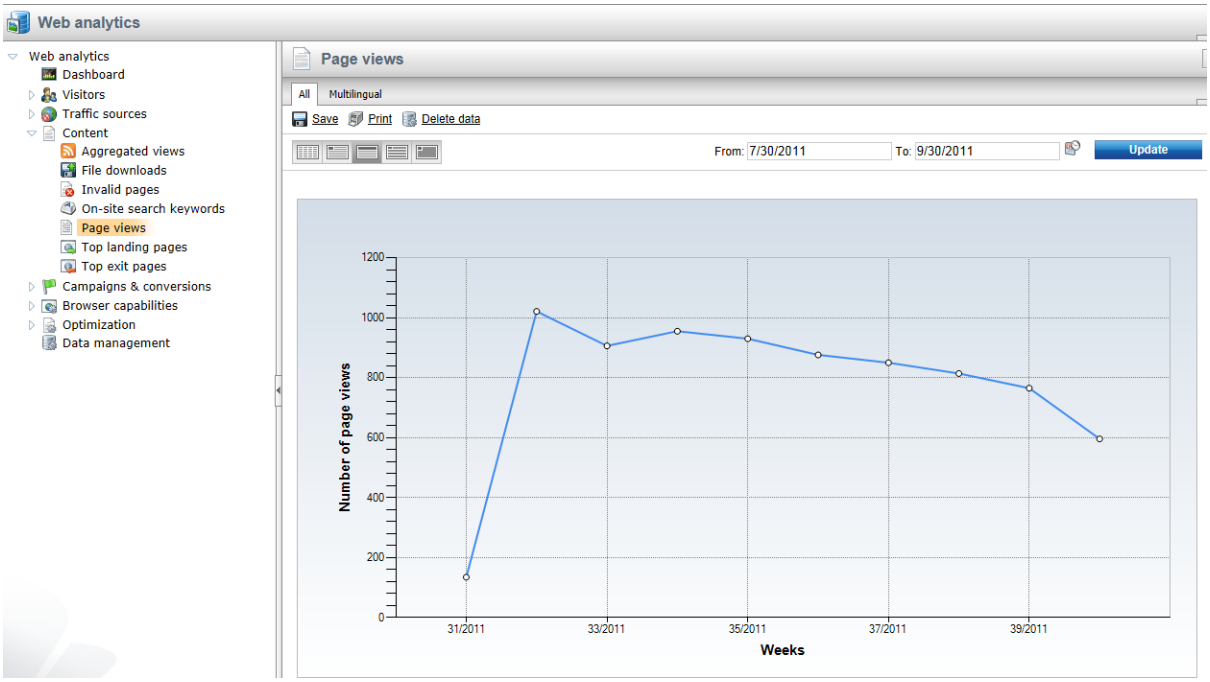
User contributions (Wiki) rely on standard Kentico CMS content management features. As such, they do not have a special programming interface and you need to use the document management API described in the Content management internals chapter.

8.49 Web analytics

8.49.1 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. Only activity on the live site will be measured, the statistics of the pages belonging to the Kentico CMS administration interface (CMS Desk and Site Manager) are not tracked.

The web analytics interface can be accessed in **CMS Desk -> Tools -> Web Analytics**. As you can see in the screenshot below, there are various types of statistics available in the tree on the left that track certain events that occur on the website. When you select a statistic, the corresponding data is displayed on the right.



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 in some cases wish to track custom events that are unique for your website. For instructions on how this can be done, please refer to the Adding custom analytics topic.

Enabling the Web analytics module

The Web analytics module is disabled by default. You can enable it in **Site Manager -> Settings -> On-line marketing -> Web Analytics** by checking the **Enable Web Analytics** checkbox. The other settings in this category are used to enable or disable tracking of various types of statistics and events, as described in the Configuration options topic.

How it works

When a tracked event (such as a page view, file download, etc.) occurs, a record of it is logged into a file in the *~/App_Data/CMSModules/WebAnalytics/* folder. The names of the log files use the following format: *<event type>_<date>_<time>.log*

A global scheduled task reads the content of all analytics files every minute (or other scheduled interval) and imports the processed data into the database. This data is then loaded and displayed in an easy to read format using web analytics reports defined through the Reporting module. The name of the scheduled task that performs this functionality is **Process analytics log** and you can view its status or configure its settings in **Site Manager -> Administration -> Scheduled tasks**.

The analytics data may also be managed directly from the interface, which can be used either to delete old data or generate sample values for statistics. Please see the Managing analytics data topic for further information.

Disk Permissions

Please note that the Web Analytics module requires that the Modify permission for the *~/App_Data* folder on your disk is granted to the ASP.NET account (see the Disk permissions problems chapter for the names of the account under various operating systems).

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

Since all tracked events are first stored in temporary files in the local file system and need to be processed by a scheduled task on a regular basis, there may be a delay between the time that an event occurs and the time it is reflected in the web analytics statistics.

8.49.2 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.
Visits	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 activities performed by a specific user over a day-long time period. Also displayed is the ratio between the number of new and returning visitors. 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 **Site Manager -> 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.
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 a link to the website and the corresponding number of page views gained from each URL.
Referring sites	Displays the total amount of page views gained through links from external websites. Additionally, the statistics for individual website domains can be viewed.
Search engines	Displays the amount of page views gained from traffic generated by search engines and the statistics of individual engines. Additional details can be found in the Monitoring traffic from search engines topic.
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

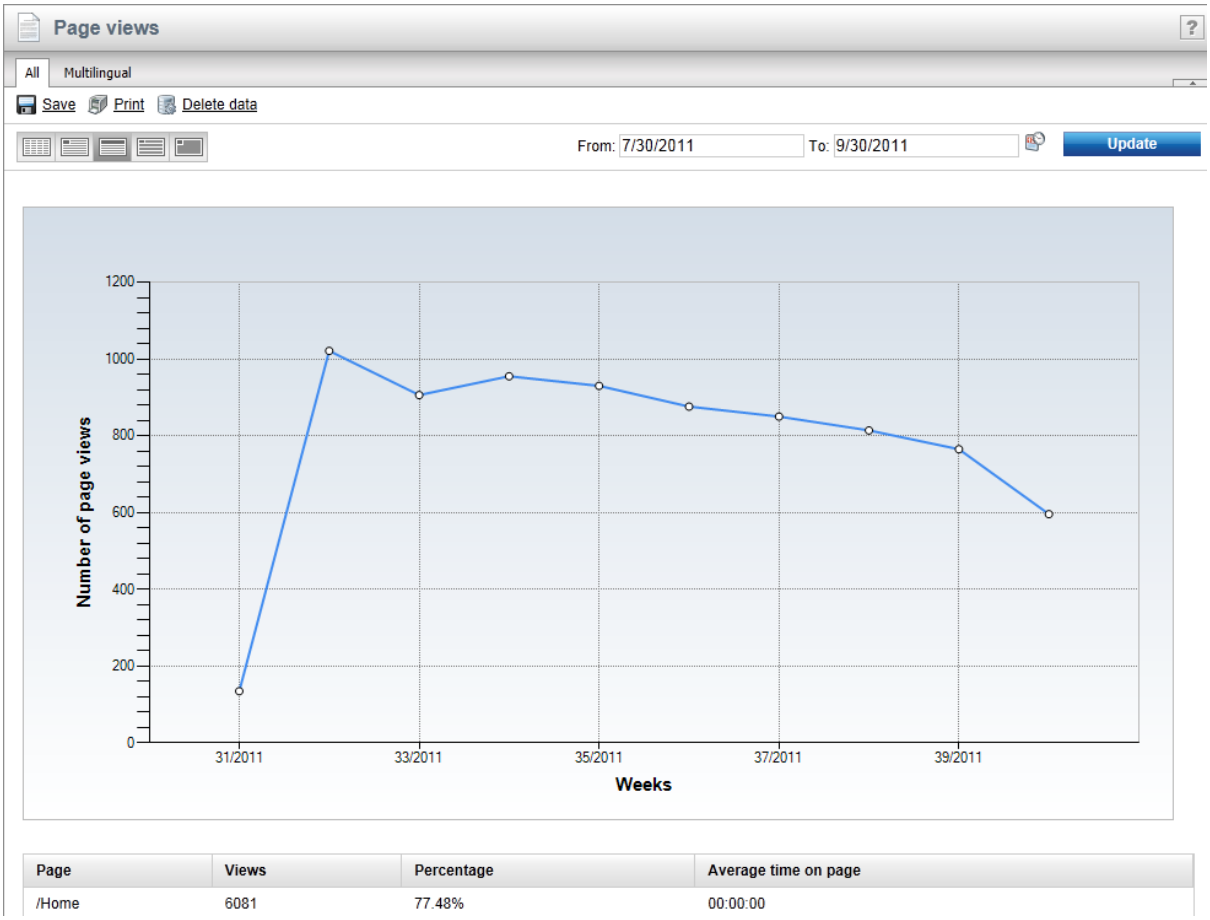
When a statistic is selected from the tree menu in the web analytics interface, its report will be displayed on the right. For most statistics, the data is organized into a graph showing the hits measured over time and a table containing detailed information for individual objects.

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.

The data displayed in the reports may also be exported into external files using various formats. This can

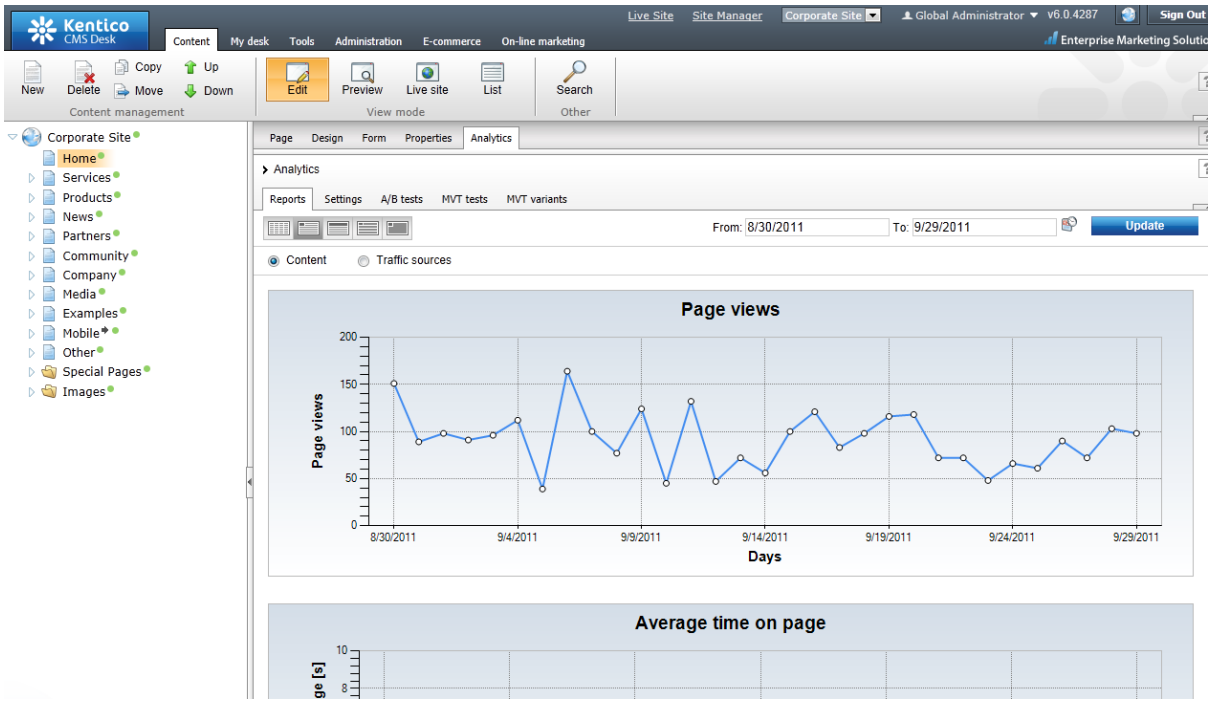
be done by right-clicking on a graph or table in the given report, which will open a context menu offering the following options:

- **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 and the statistics of individual sites.
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.

8.49.3 Conversions

8.49.3.1 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.

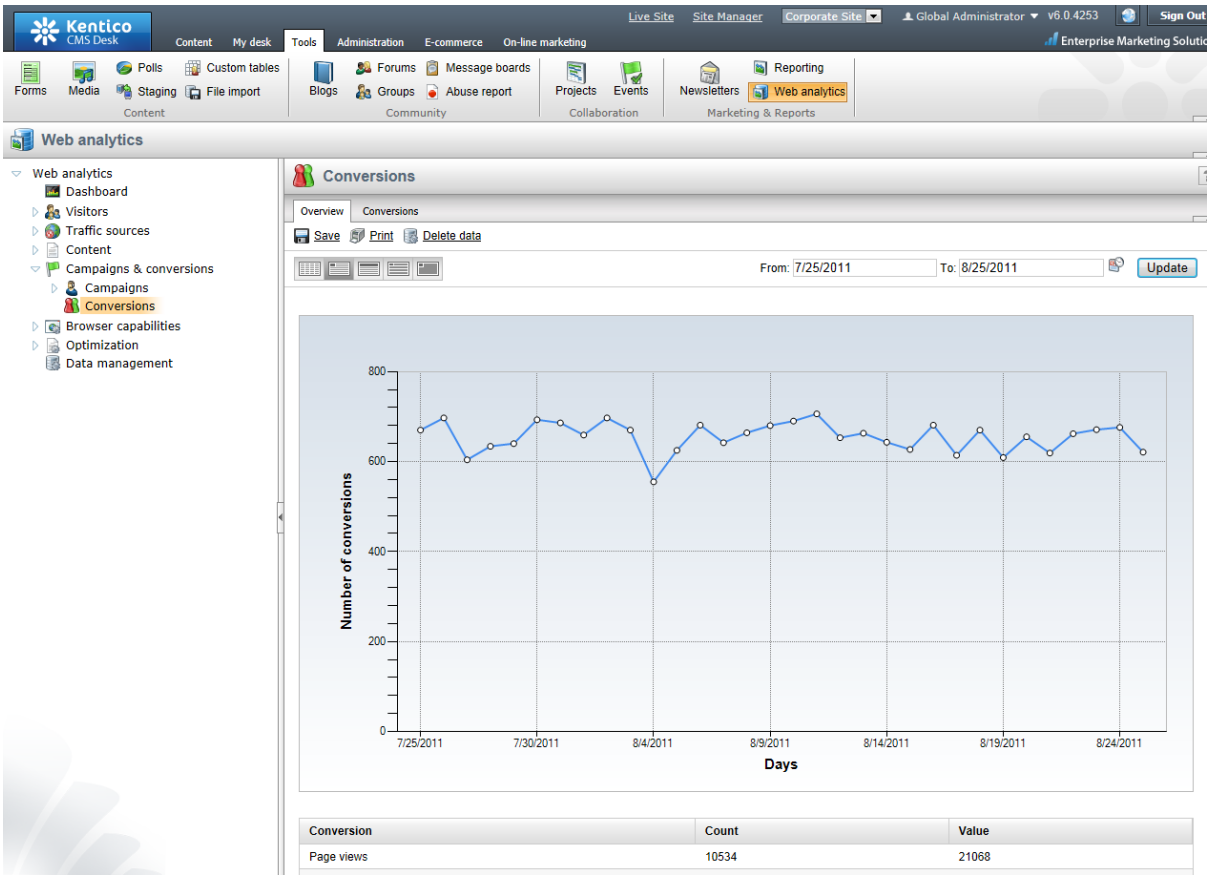
While simply tracking all conversions that occur on the website can be useful, in many cases you may also require additional information about the context in which the given actions occurred. For this reason, conversion tracking is integrated with several other web analytics and on-line marketing features. 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.

8.49.3.2 Managing conversions

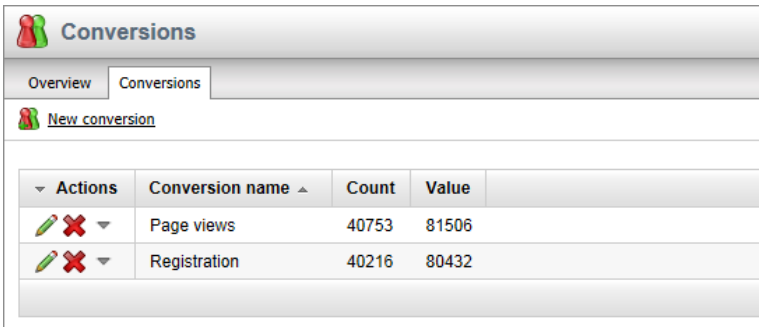
To access the management interface dedicated to conversions, go to **CMS Desk -> Tools -> Web analytics**, expand the **Campaigns & conversions** category in the tree on the left and select **Conversions**. This section of the UI may alternatively be reached by navigating to **On-line marketing -> Conversions**.

The **Overview** contains a web analytics report displaying the number of conversions that were logged on the website over the specified time period.



This report only provides a general summary of the total conversion statistics. Additional reports with more detailed data for conversions that were logged under special circumstances are available 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 the **New conversion** link 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.

8.49.3.3 Logging actions as conversions

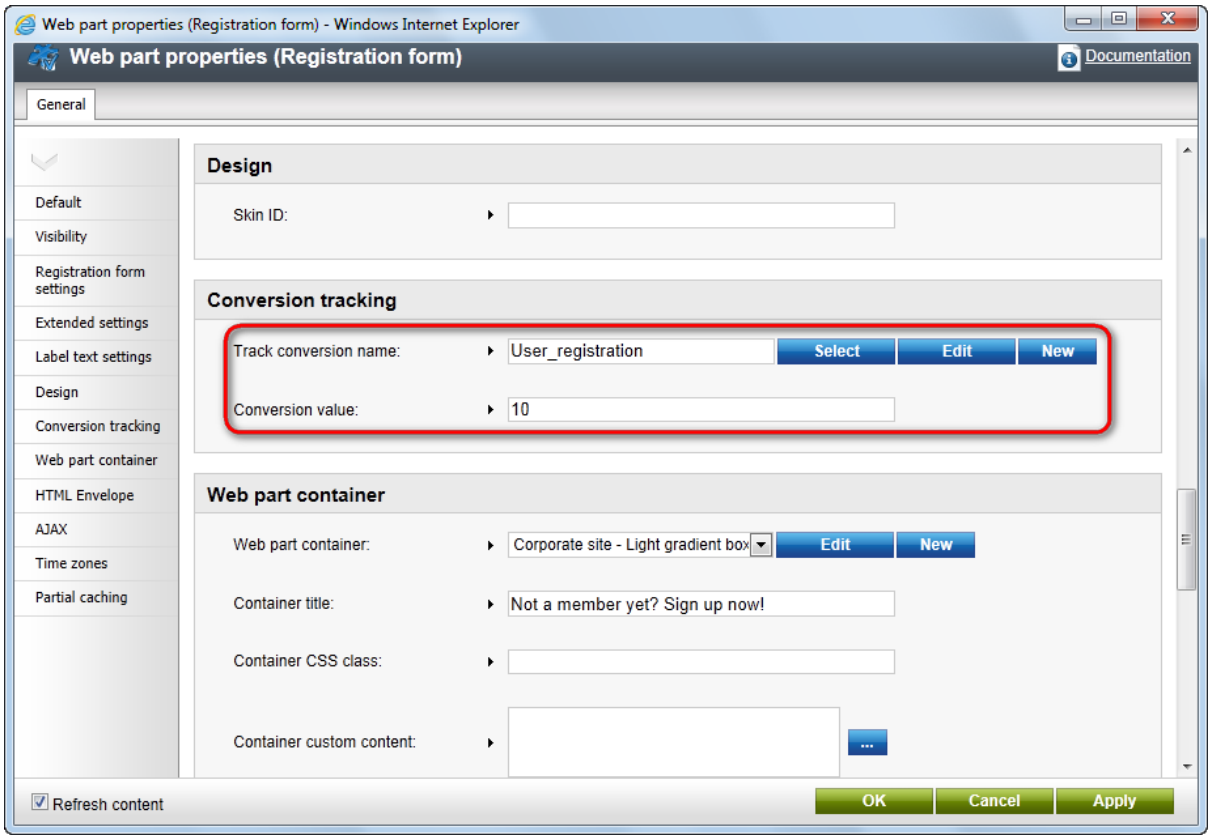
This topic describes the possible options that may be used to ensure that the system logs specific events as conversions. When assigning a conversion through the user interface, two types of fields are provided.

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.



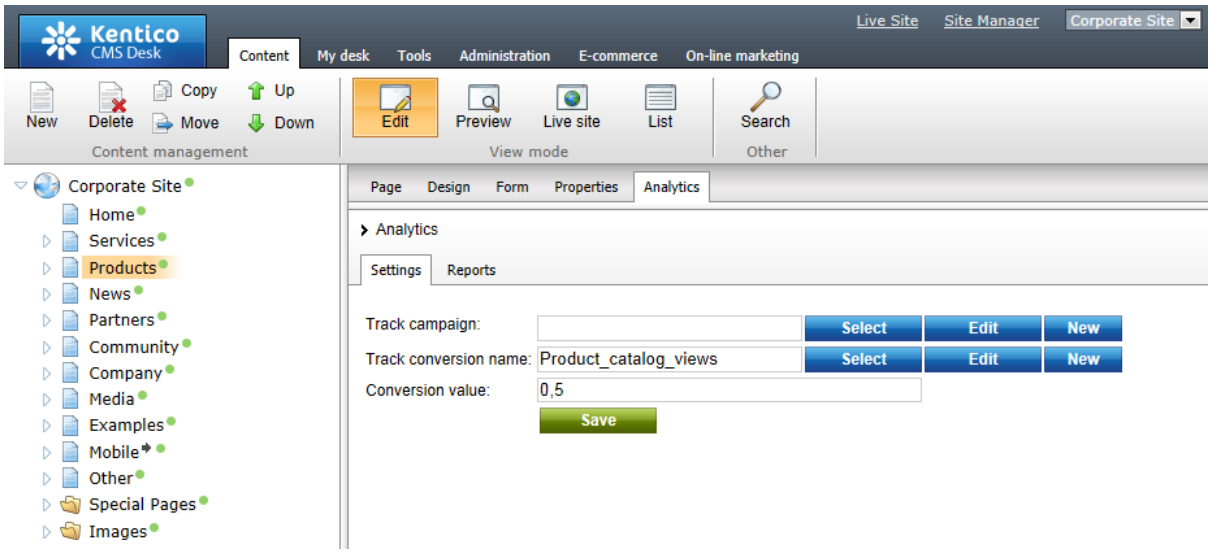
Below you can find a list of all types of actions that can be tracked as conversions through web parts:

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 web part. This can also be done for the widget based on this web part.
Shopping cart actions	The Shopping cart web part can be used to track three 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 through the shopping cart.

	• **Add to shopping cart** - occurs when a user adds a product to the shopping cart. You can assign a different conversion and value to each of these actions through the appropriate **conversion name** and **conversion value** properties. The conversion value properties support 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**: *{%ShoppingCartItemInfo.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.
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.

Page views of specific documents

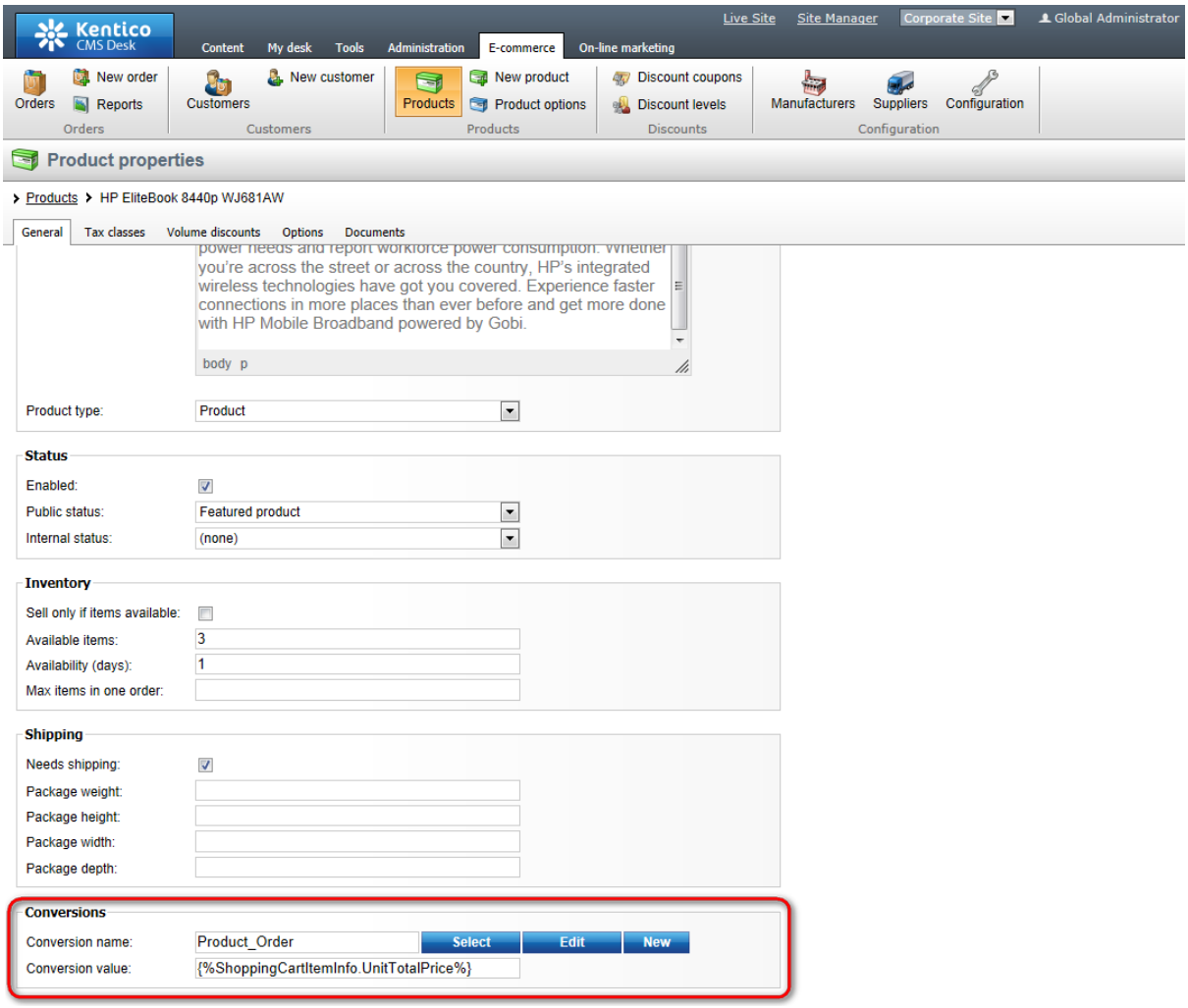
You can also use conversions to keep track of the amount of hits received by individual pages. To configure this behaviour for a page, go to **CMS Desk -> Content -> Edit**, select the document representing the given page from the content tree and switch to its **Analytics -> Settings** tab. To assign 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 product orders

The **Shopping cart** web part allows 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 editing (🖉) the particular product (or by selecting a product document in the content tree and editing it on the **Product -> General** 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 enter an appropriate value that will be recorded for each conversion hit. You may enter a macro expression here, for example: *{%ShoppingCartItemInfo.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.

8.49.4 Campaigns

8.49.4.1 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.

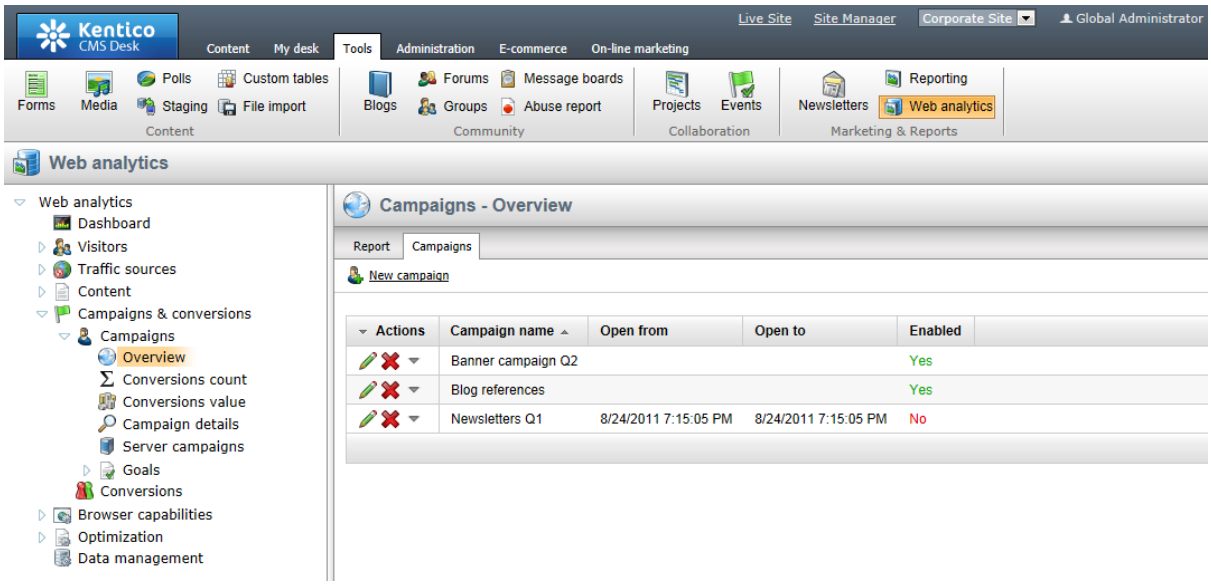
If the visitor registers on the website while the cookie is still valid, the name of the currently active campaign will be permanently stored in their user settings (administrators may view the value in **Administration -> edit ( ) user -> Settings -> Campaign**). This data field only serves to provide 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.

8.49.4.2 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**).

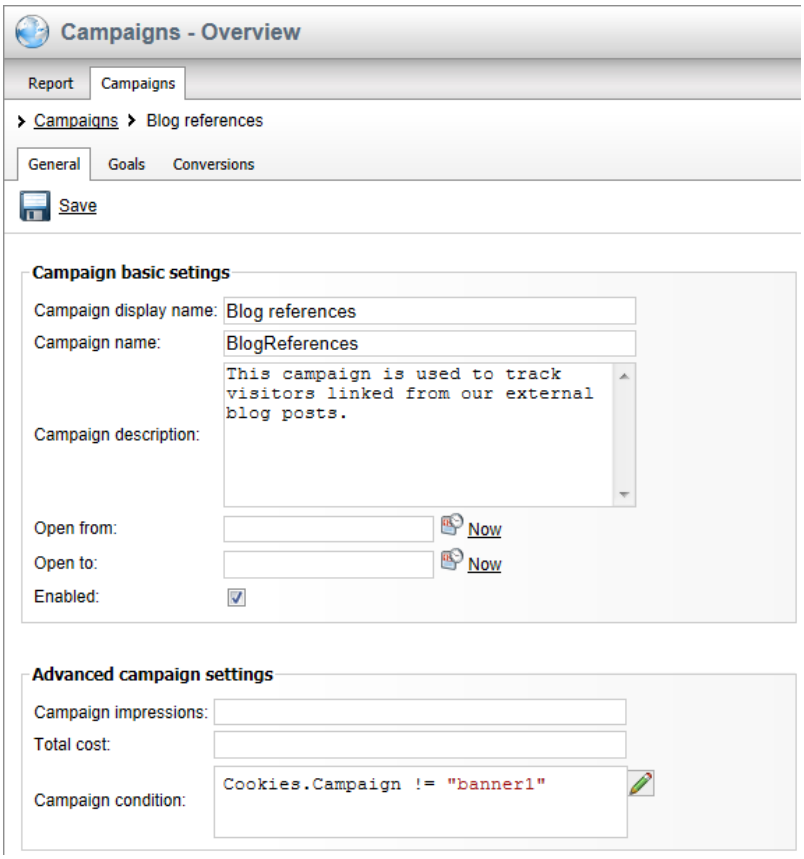
Here you can view the list of all campaigns defined for the current website and manage them as needed.



To create a new campaign, click the **New campaign** link 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 **OK**. 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).

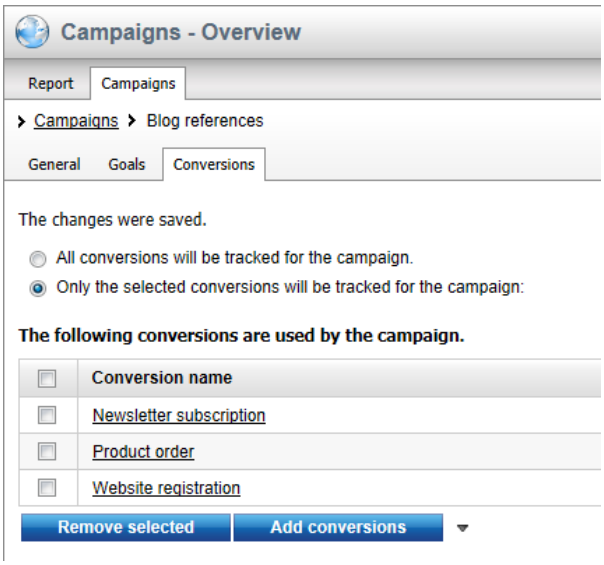
An important property is the **Campaign condition**, which allows you to specify a macro condition that must be fulfilled in order for visitors to be included in the campaign. Through the edit icon (), you can use the Macro condition editor, which provides a graphical interface that makes it easier to build complex conditions.

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.



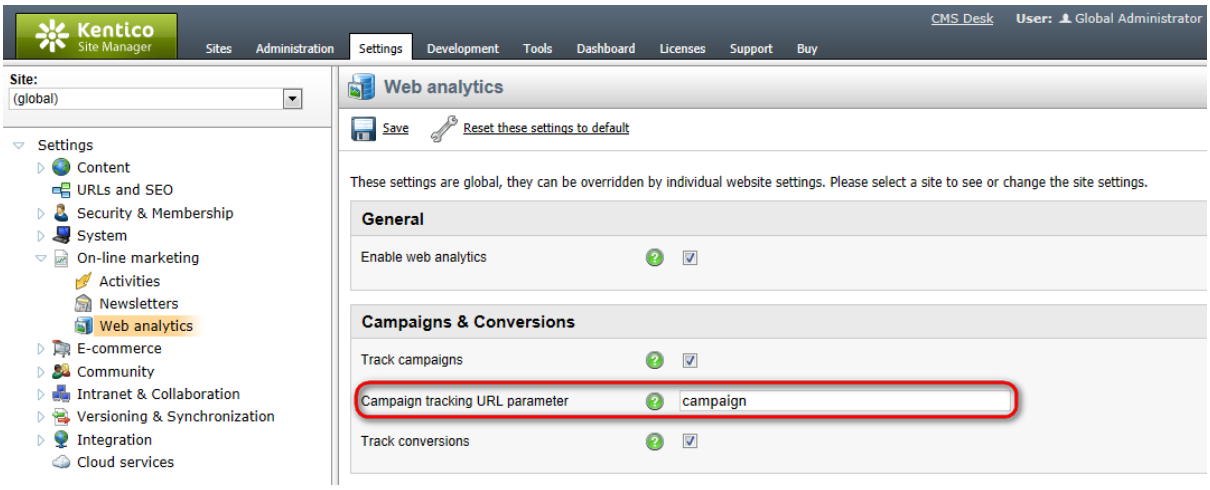
In both cases, only conversions performed by visitors who arrive on the website as a result of the given campaign will be logged.

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:

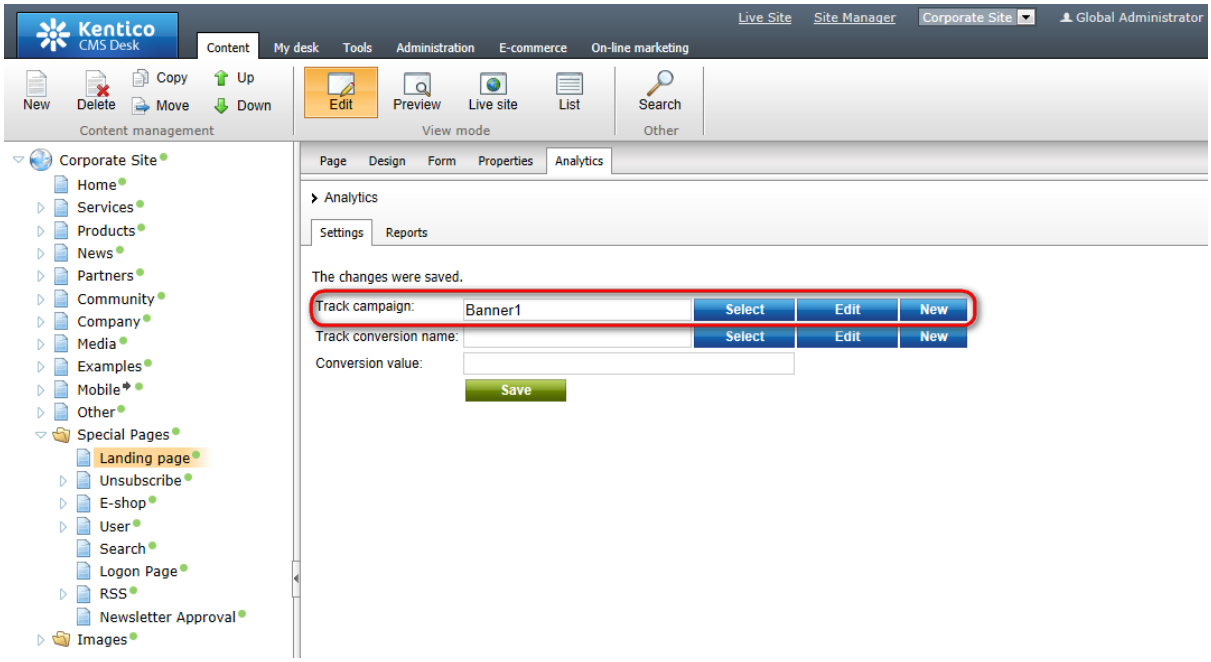
http://www.mywebsite.com/Home.aspx?campaign=banner1
http://www.mywebsite.com/News.aspx?campaign=newsletterJune

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.

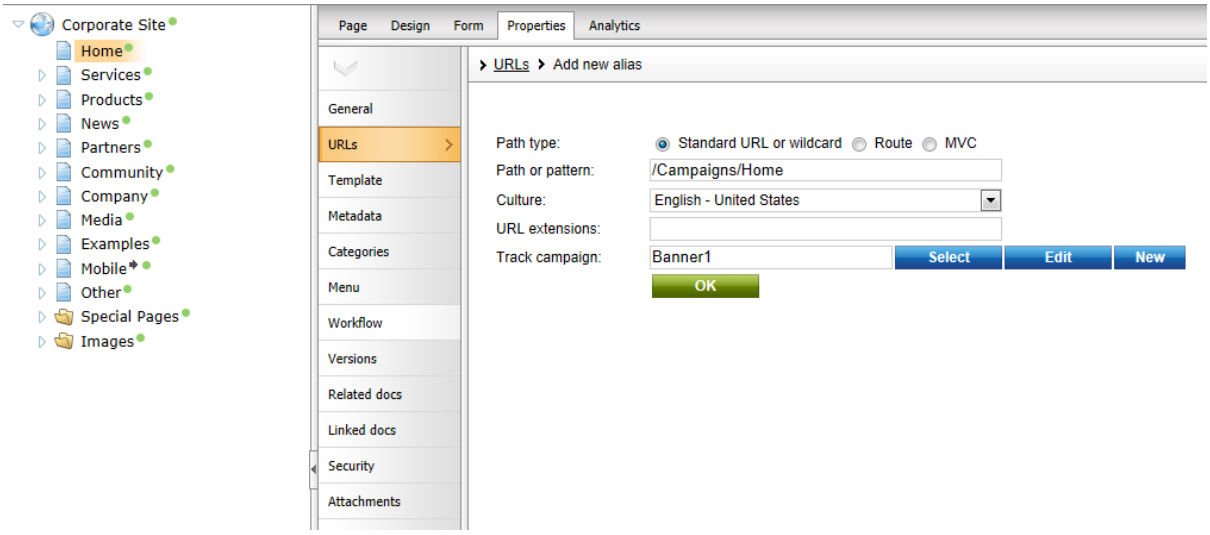


To ensure accurate campaign tracking, it is recommended to create a special page that will not be available in the website's standard navigation, since the campaign will be assigned to all users who view this page (and repeated views by the same user will also be logged as campaign visits). When using 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 the **Add new alias** link 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:

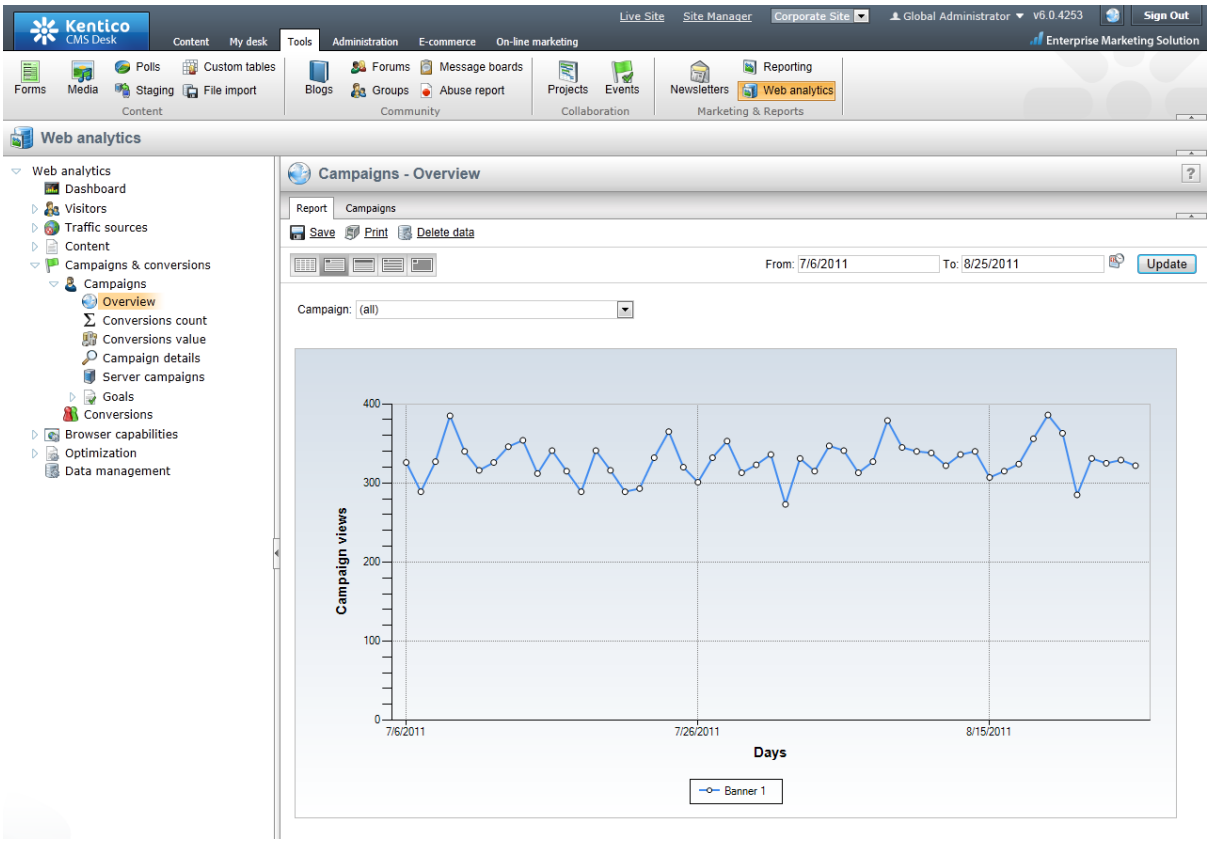
```
<add key="CMSEnableAutomaticCampaignCreate" value="true" />
```

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.

8.49.4.3 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

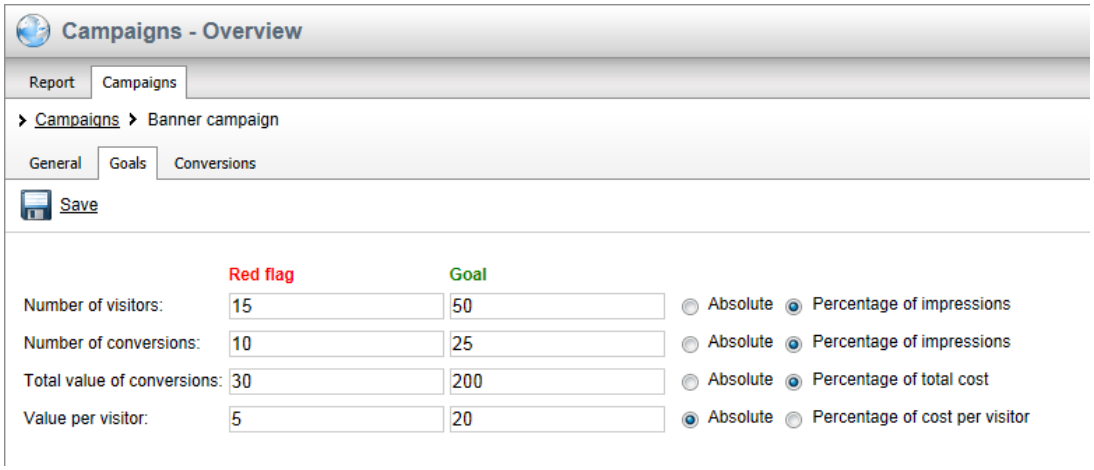
In addition to the data provided by the reports listed above, you can optionally specify goals that should be achieved by each campaign and then compare them with the actual results.

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*).

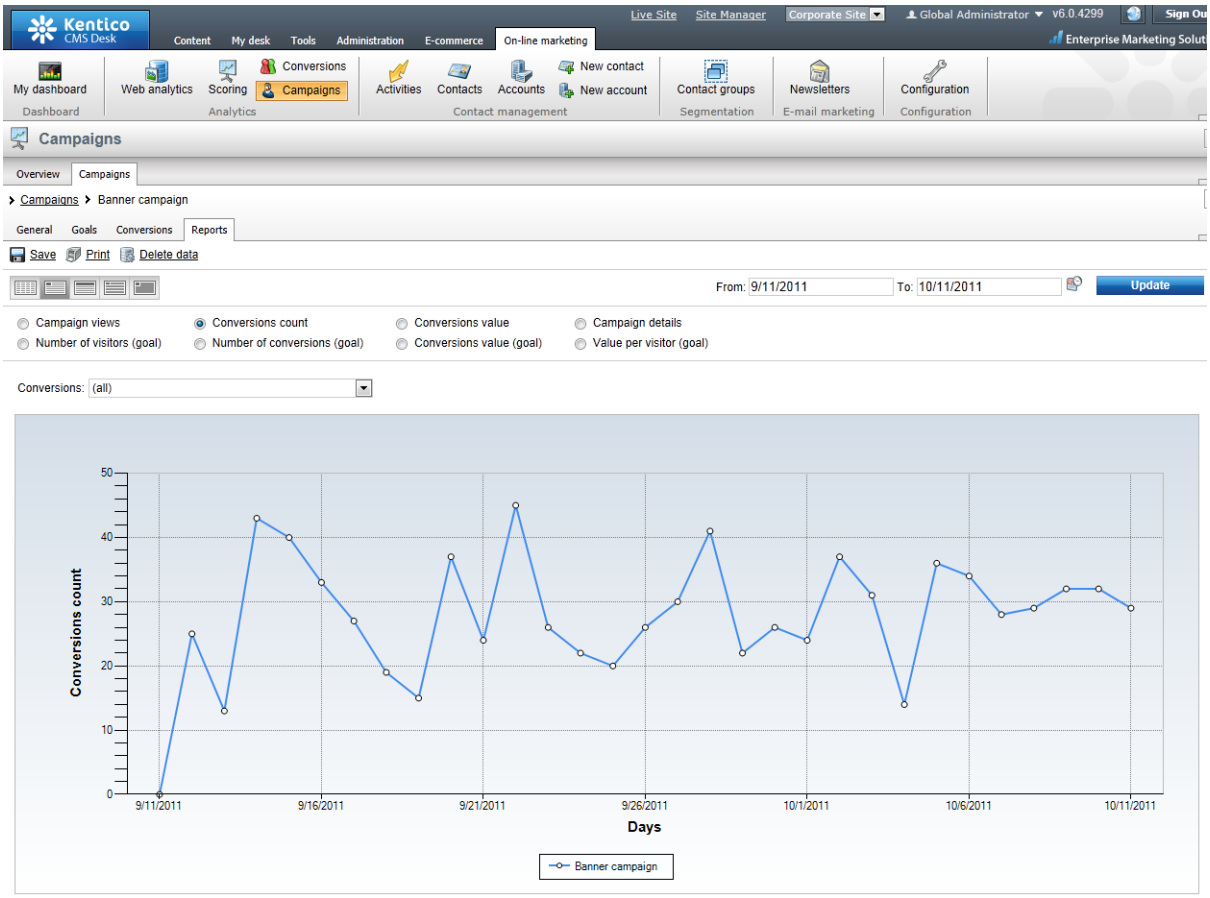


Each goal may have two values. The **Red flag** may be used to set a number that must be reached in order for the campaign to be at least partially successful (campaigns below this value are considered to have failed). The **Goal** value sets a target that should ideally be achieved by the campaign. 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.

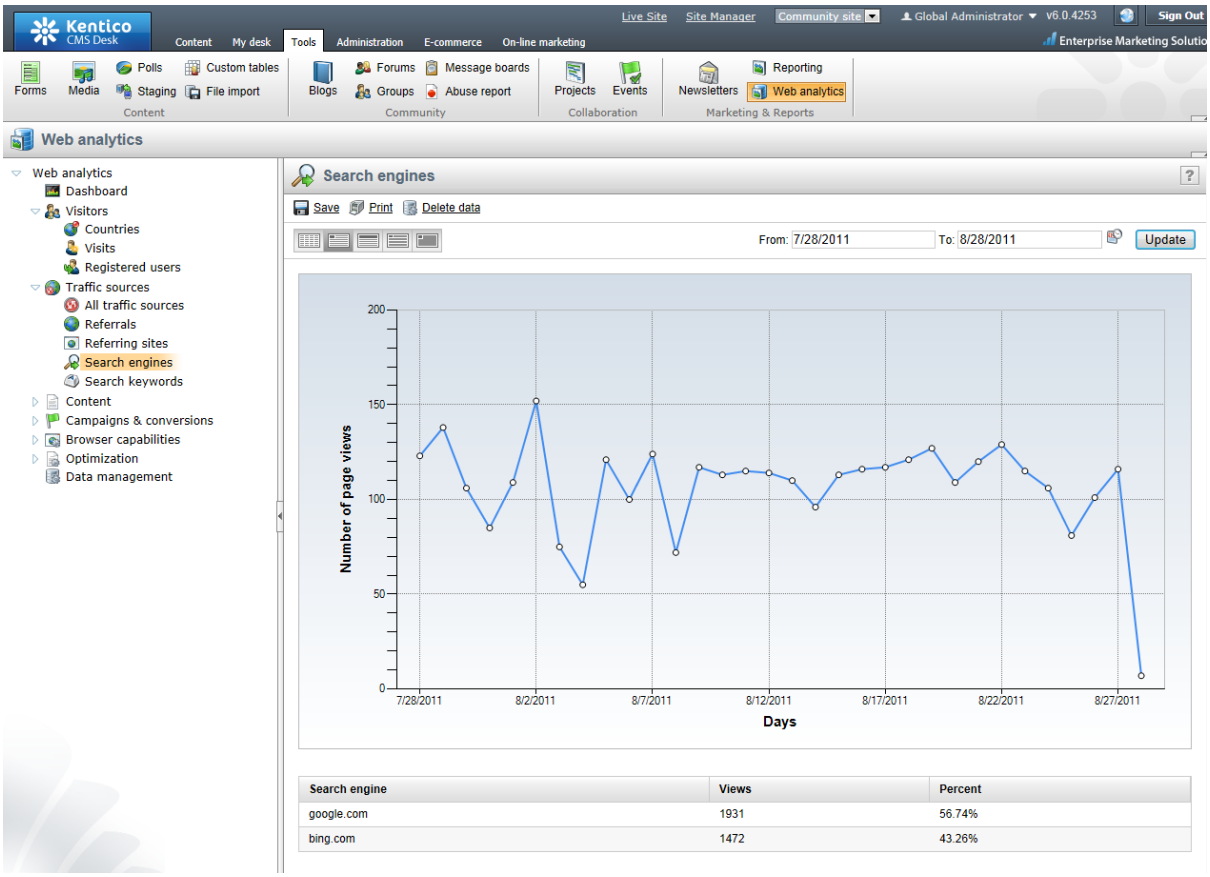


Only the data logged for the given campaign is displayed here.

8.49.5 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 found in the web analytics tree menu in **Traffic sources -> Search engines**. As shown below, it contains a graph showing the total amount of page views caused by search engine traffic and a breakdown of the statistics for individual search engines.



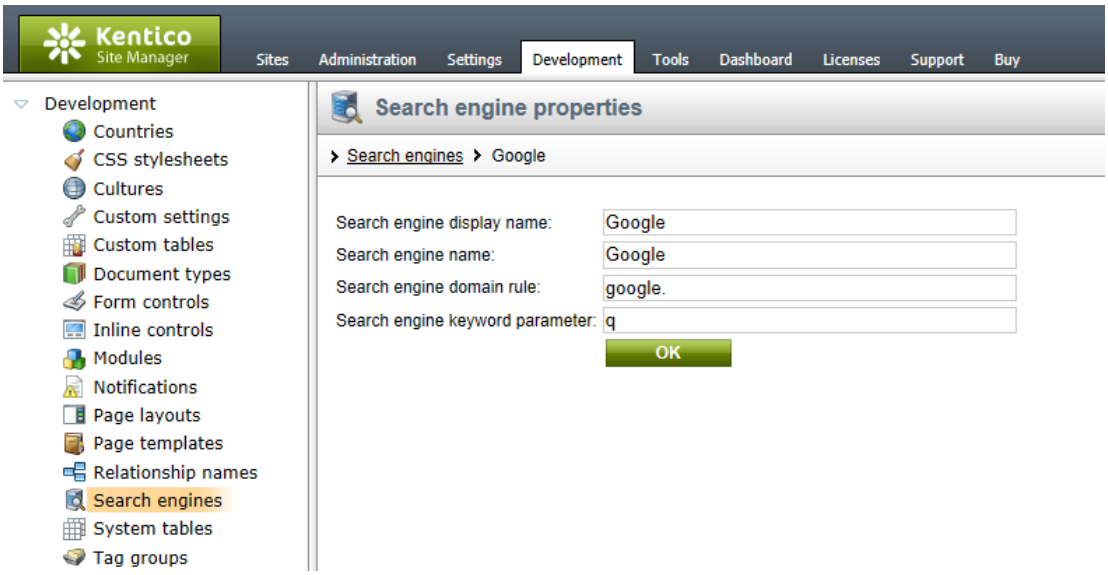
Additionally, it is possible to view the specific keywords that were entered into the search engines using the **Search keywords** statistic (also under the **Traffic sources** category).

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 must be specified:

- **Search engine display name** - sets the name of the search engine used in the administration and web analytics interface.
- **Search engine name** - sets a unique identifier for the search engine.
- **Search engine 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.
- **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.



If the site is accessed from an external website, the URL of the page that generated the request is parsed. First, it is checked for the presence of a valid **Search engine 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 **Search engine 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 way, website traffic that is gained from search engine results can be accurately tracked.

8.49.6 Adding custom analytics

This example describes in detail how you can track custom events using web analytics, measure them as statistics and display the results in reports. A simple button click event will be logged for this purpose, but you may use the same basic approach to implement tracking of any type of events on your website.

Logging the event

1. In this example, the event will be logged through a user control. Open your Kentico CMS web project in Visual Studio and 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). Right-click the folder and select **Add New Item**. Choose to create a **Web User Control** and name it *AnalyticsButton.ascx*. This location ensures that the control will be exported along with your website if it is deployed to a different server.

2. Add the following code to the control:

```
<asp:Button id="btnLog" runat="server" Text="Add analytics hit" onclick
="btnLog_Click" />
```

This inserts a simple Button control which will be used to add hits to the custom statistic.

3. Switch to the code behind file of the control and modify it according to the following. Please keep in mind that 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
{

    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);
        }
    }
}
```

As shown above, custom events for web analytics can be logged 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 should be logged.
- **culture** - specifies the culture code under which the event should be logged.
- **objectName / objectId** - used to specify the context in which the hit was logged. It is possible to use a name or ID of an object. For example, when logging page views, the alias path and document ID of the given page would be stored. In the current scenario, the name of the user who clicked the button is logged as the related object.
- **count** - sets the amount of hits that should be added to the statistic. This parameter is optional and the default value (1) is used if it is 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**).

When a user clicks the button, a log will be created and processed. The results containing the given user's name and the amount of clicks will be stored in the database, where they can be used by the web analytics module.

Save all changes to the user control's files.

4. Next, it is necessary to publish this user control on your website (this example will use the sample Corporate site). Go to **CMS Desk -> Content -> Edit**, select the **Home** document from the content tree, switch to the **Design** tab and add () a web part to the *zoneCenter* zone. Select the **General -> User control** from the web part catalog and enter *~/CorporateSite/AnalyticsButton.ascx* into its **User control virtual path** property. The button will now be displayed on the page.

There are also other ways to add custom web analytics functionality to the pages of your website. For example, you could implement the control as a custom web part, add it to your pages and configure it as necessary. If you are using the ASPX page template development model for the website, you can of course integrate the 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 by using the Web analytics -> Analytics custom statistics web part.

Once this web part is placed on a page, it will automatically add a hit to a custom statistic when the given page is viewed. The details of the statistic, such as its code name, the name of the related object or the hit value can be specified using the web part's properties. The site name and culture is automatically taken from the current context of the page.

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.

Before you can view the statistics of the button click event in **CMS Desk -> Tools -> Web Analytics**, the reports that will display the data of the statistics have to be created.

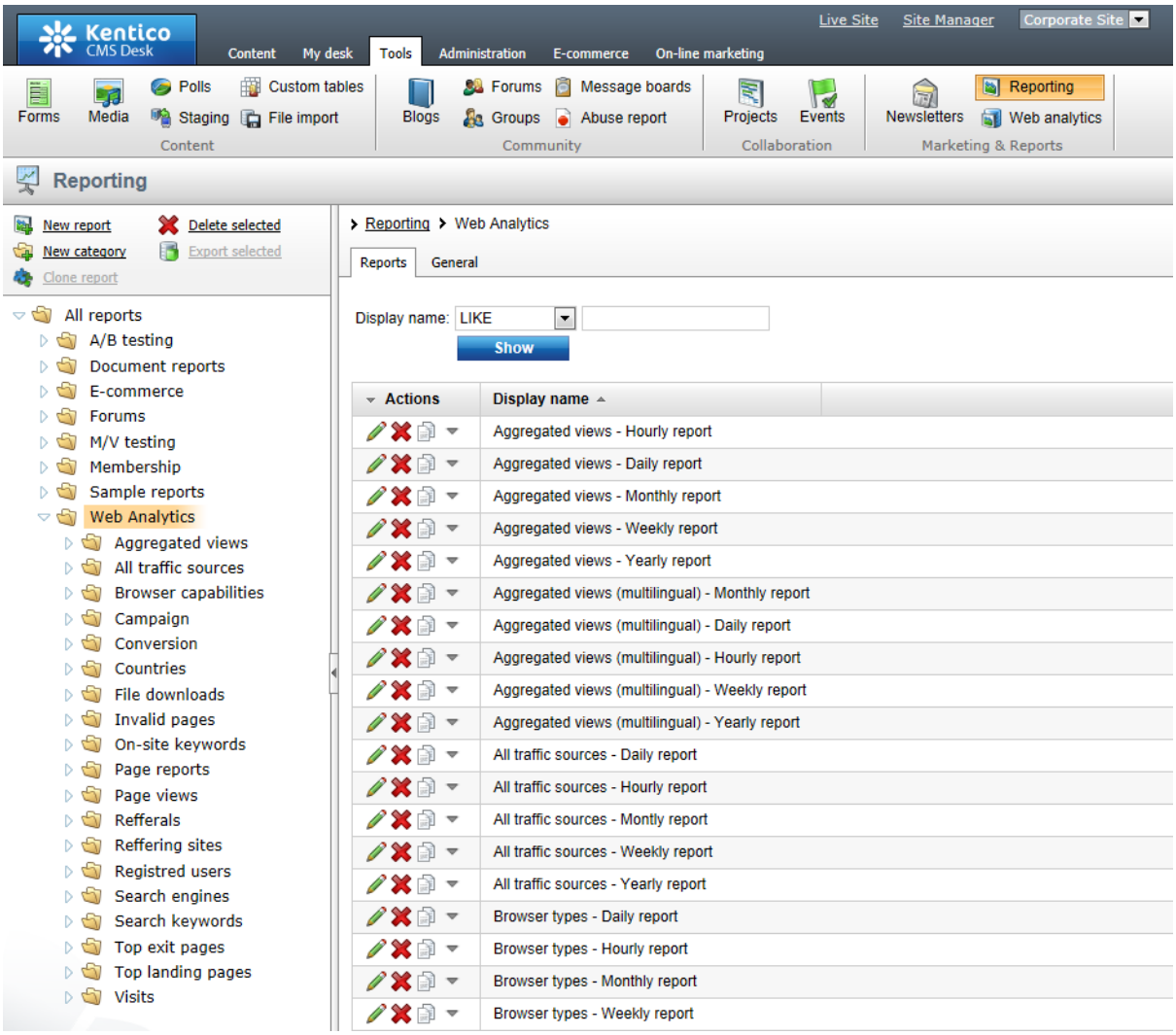
Creating reports for the statistic

6. Go to **CMS Desk -> Tools -> Reporting** and expand the **Web Analytics** category in the tree on the left. There are already quite a few sub-categories that contain reports used by 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.



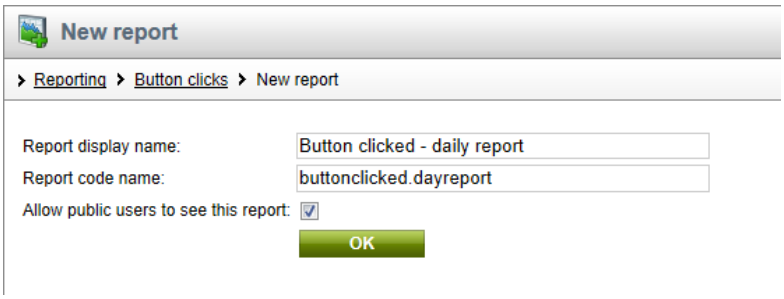
7. Add a **New category** under **Web Analytics** and enter the following names for it:

- **Category display name**: Button clicks
- **Category code name**: ButtonClicks

Click **OK**.

8. Now we will create a daily report for the new statistic. Click the **New report** link and enter the following values:

- **Report display name**: Button clicked - daily report
- **Report code name**: buttonclicked.dayreport



Click **OK** again.

9. 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:

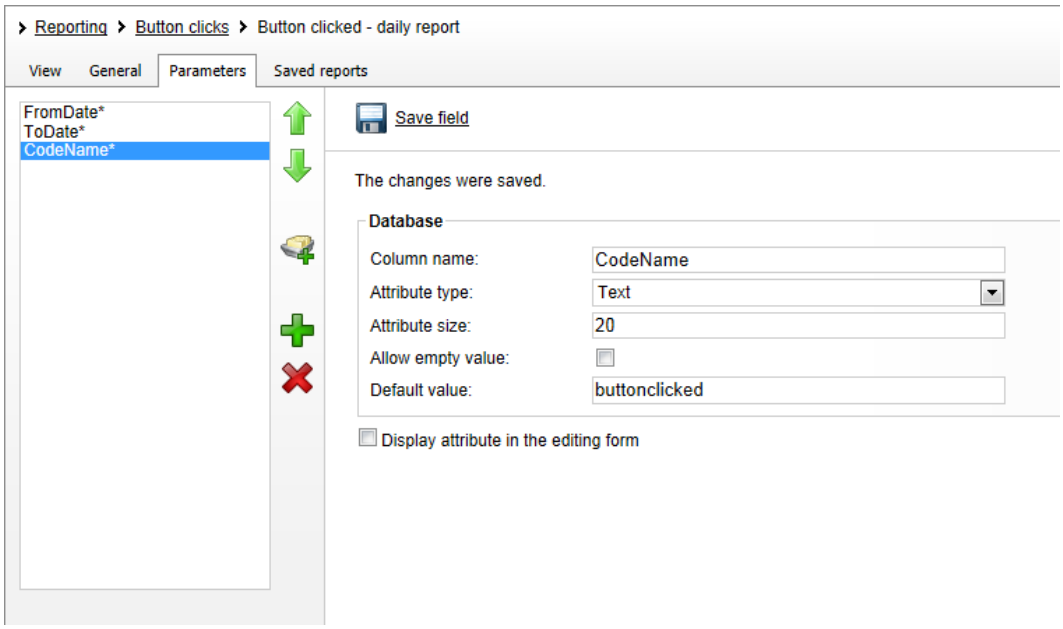
- **Column name:** FromDate
- **Attribute type**: Date and Time
- **Display attribute in the editing form**: disabled

- **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).



10. 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:

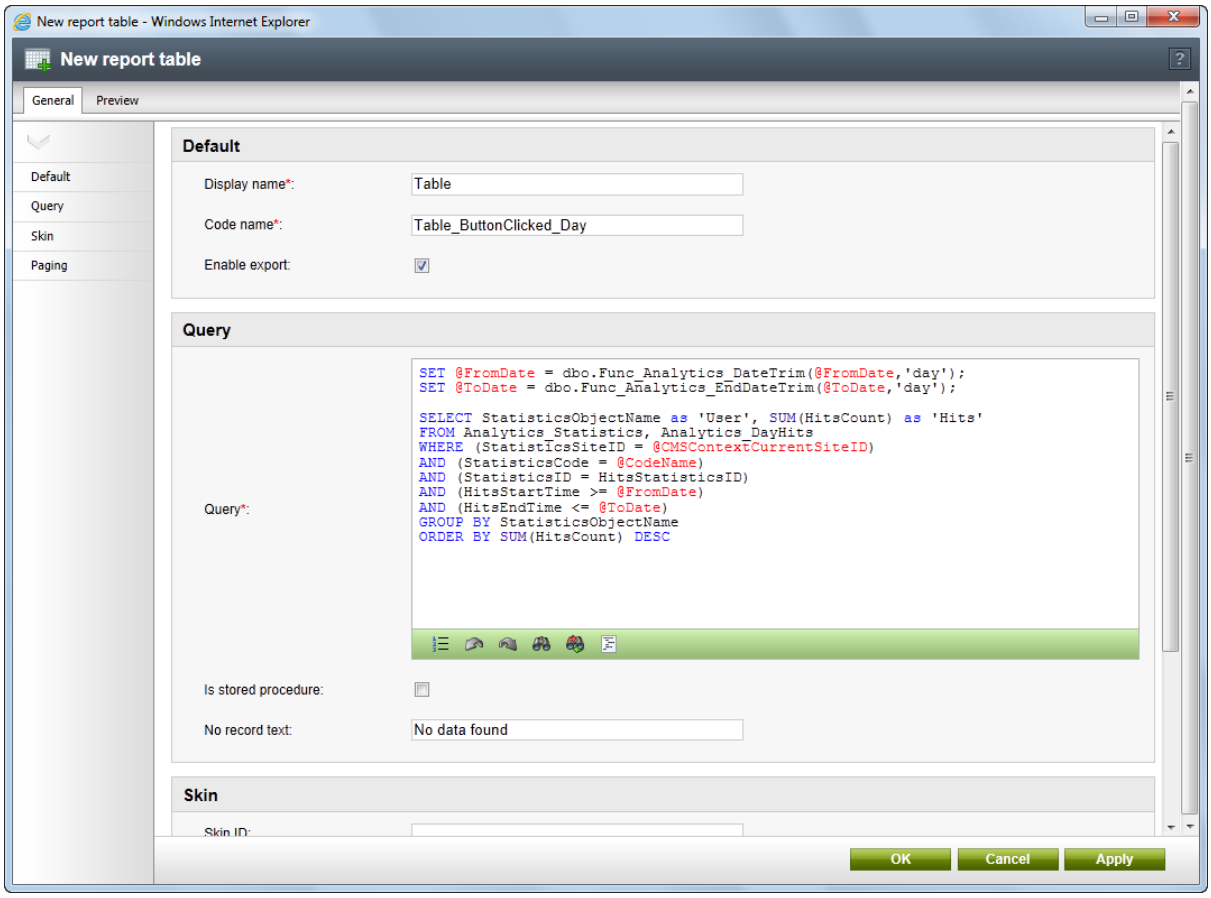
- **Display name**: Table
- **Code name**: Table_ButtonClicked_Day
- **Enable export**: true (checked)
- **Query**:

```
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).

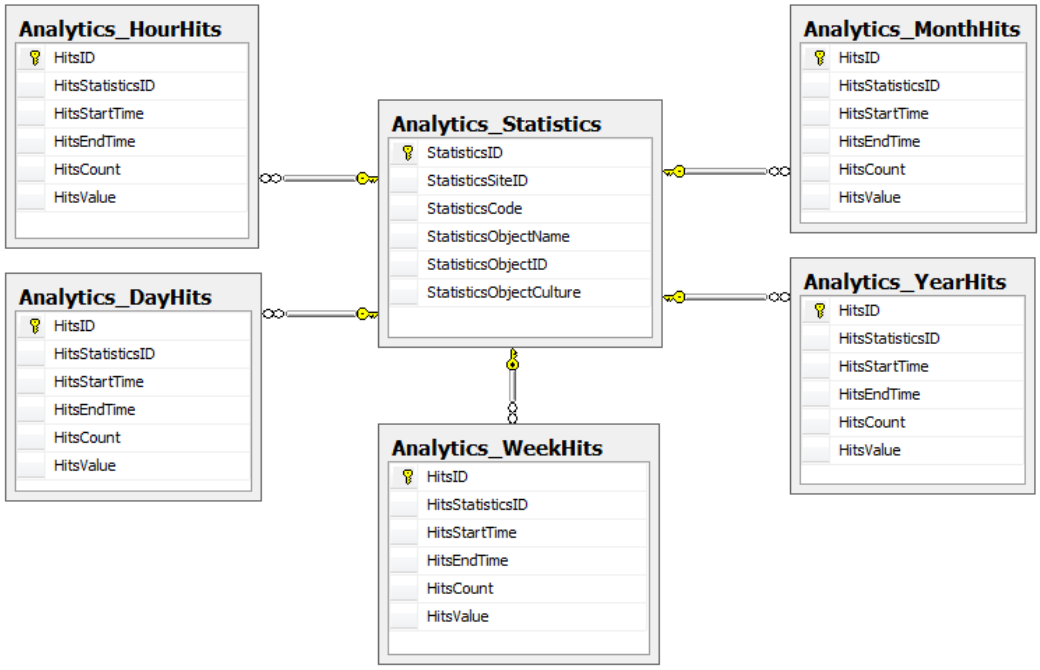
The second parameter of the trimming functions must match the time unit of the given report. The options are: *'hour'*, *'day'*, *'week'*, *'month'*, *'year'*.



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.

11. 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.

The reports for the remaining time intervals can be created 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.

Adjusting the web analytics interface

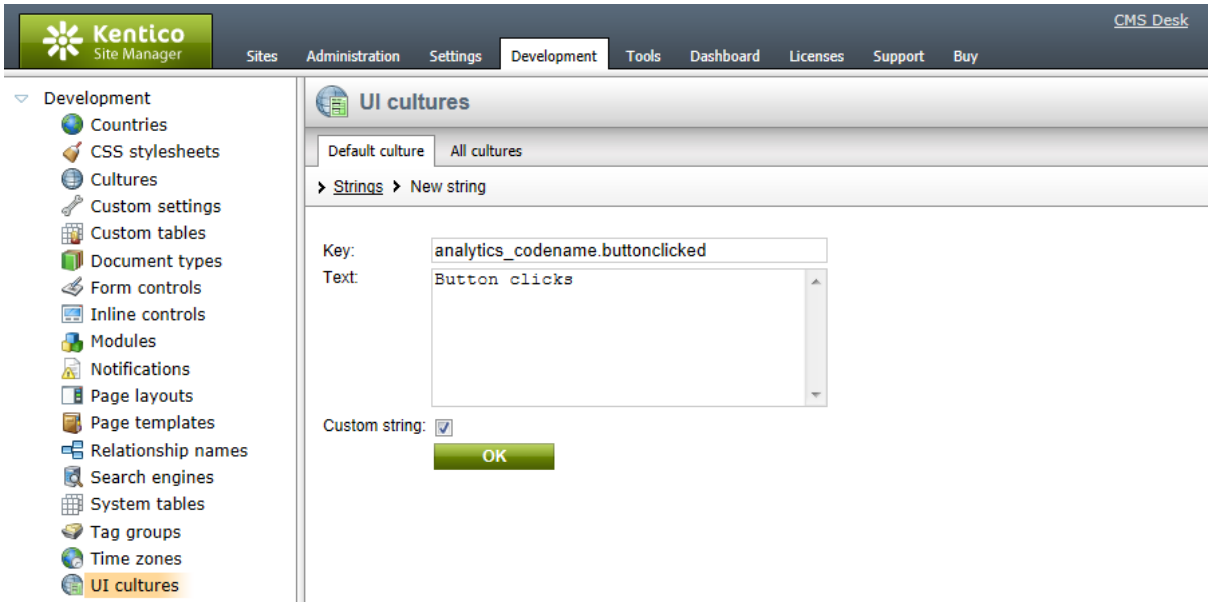
Once there are some hits logged in the database for a statistic, it will automatically be available under the **Custom** category of the tree in **CMS Desk -> Tools -> Web Analytics**. There are a couple more steps that should be taken to ensure that the statistic is displayed properly.

12. It is necessary to create the resource string that will be used as the title of the custom statistic. Go to **Site Manager -> Development -> UI cultures**, 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.

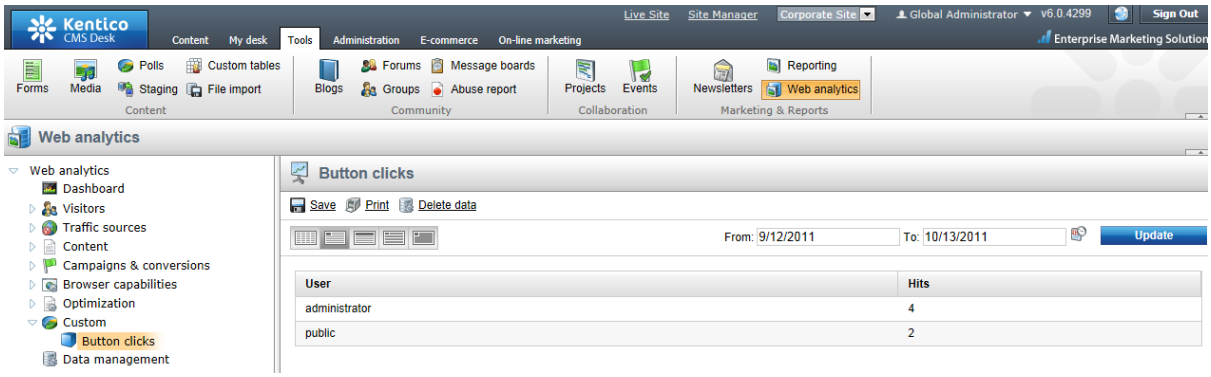
- **Custom string**: checked



Click **OK**.

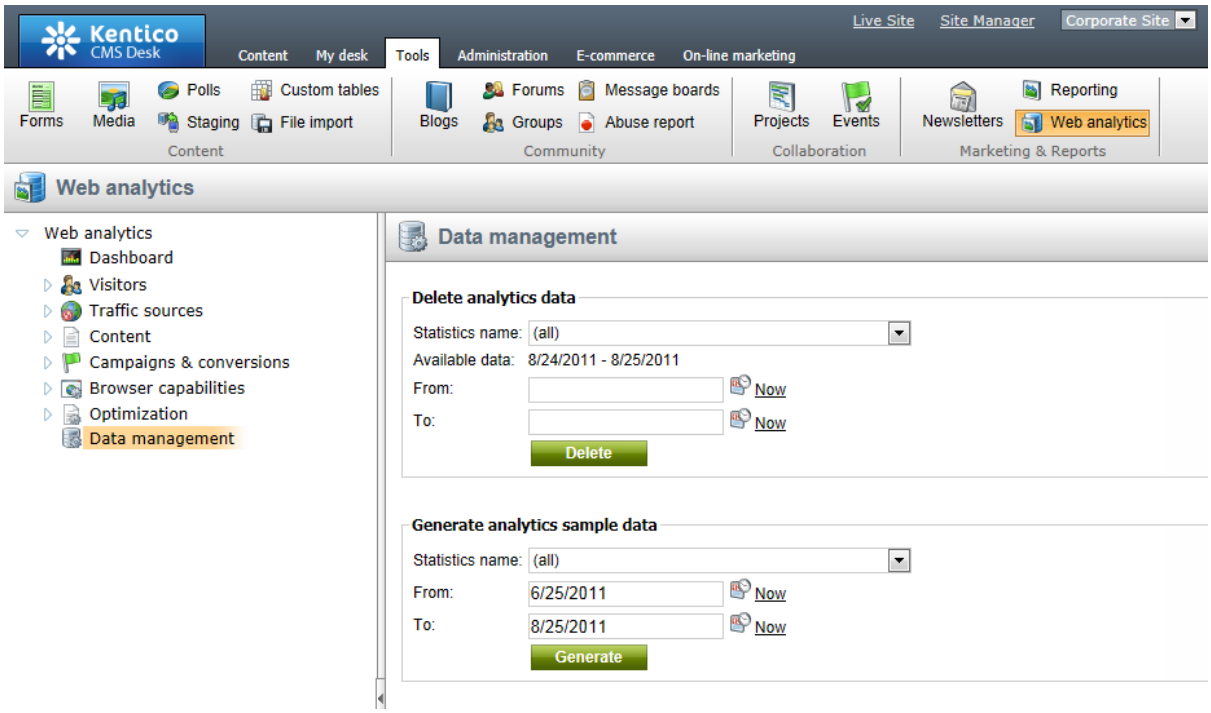
You may also add a custom icon for the new statistic. Such icons have to be stored in **~\App_Themes\Default\Images\CMSModules\CMS_WebAnalytics** and their name must be in format **<statistic code name>.<extension>** (e.g. *buttonclicked.png* in this scenario). If there are any period characters ('.') in the code name, you can replace them with an underscore ('_') to create a valid file name.

13. Finally, go to **CMS Desk -> Tools -> Web Analytics** and expand the **Custom** category in the tree. A new item will be available and its reports will display the statistics of the tracked button click event.



8.49.7 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!

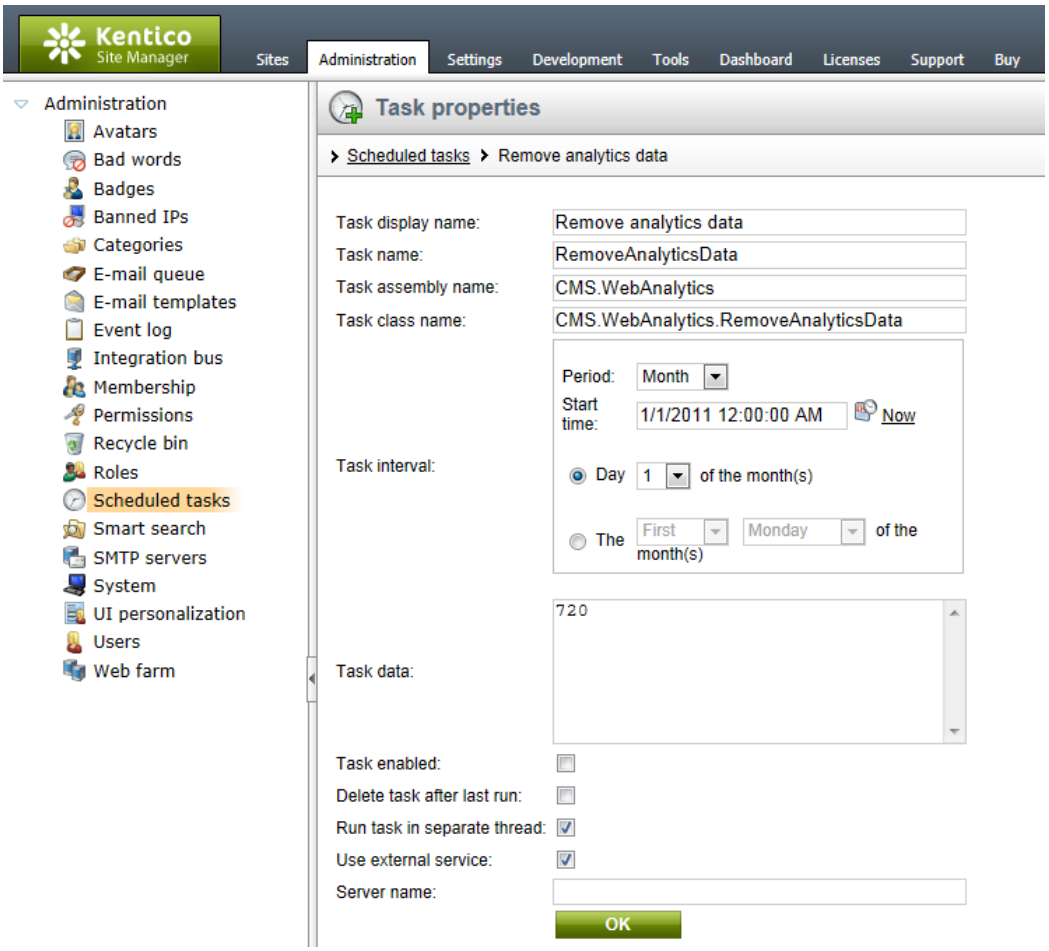
The data generation process is a highly resource-intensive operation, since it must create a very large amount of records in multiple database tables.

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.

8.49.8 Configuration options

The functionality of the web analytics module may be configured in **Site Manager -> Settings -> On-line marketing -> Web analytics**. Here, tracking of various types of events can be enabled or disabled and certain additional options may be specified. The following settings are available:

General	
Enable web analytics	Can be used to enable or disable the entire web analytics module. If this setting is disabled, none of the statistics listed below will be tracked.
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 will track which files were downloaded by website visitors. Please note that only files stored as documents in the website's content tree will be 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) will be 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 will log 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 will be tracked.
Visitors	
Track browser types	Indicates if the browser types and versions used by the website's visitors will be tracked.
Track visits	Indicates if site visits should be tracked. A single visit includes any number of page views or other activities performed by a specific user over a day-long time period. Visitors are recognized according to the presence of a browser cookie.
Remember visitors by IP (minutes)	If the entered value is higher than *0*, the IP addresses of the website's visitors will be 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 will be 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.
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 will track which links are used by visitors 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 will log when the 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.
Excluded	
Exclude search engines	If enabled, hits generated by search engine robots (crawlers) will not be included in tracking statistics.
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 will not be 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.

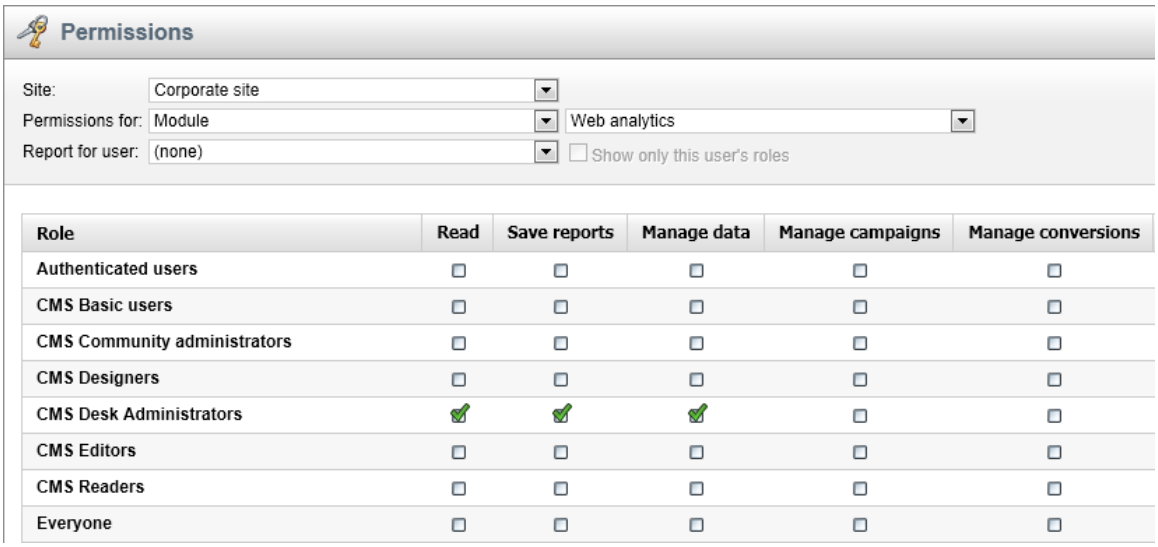
Please note: The values of these settings are checked at the moment that a tracked event occurs. Any changes will not affect previously logged data, only future events.

8.49.9 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.



8.49.10 Web analytics internals and API

8.49.10.1 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.

8.49.10.2 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

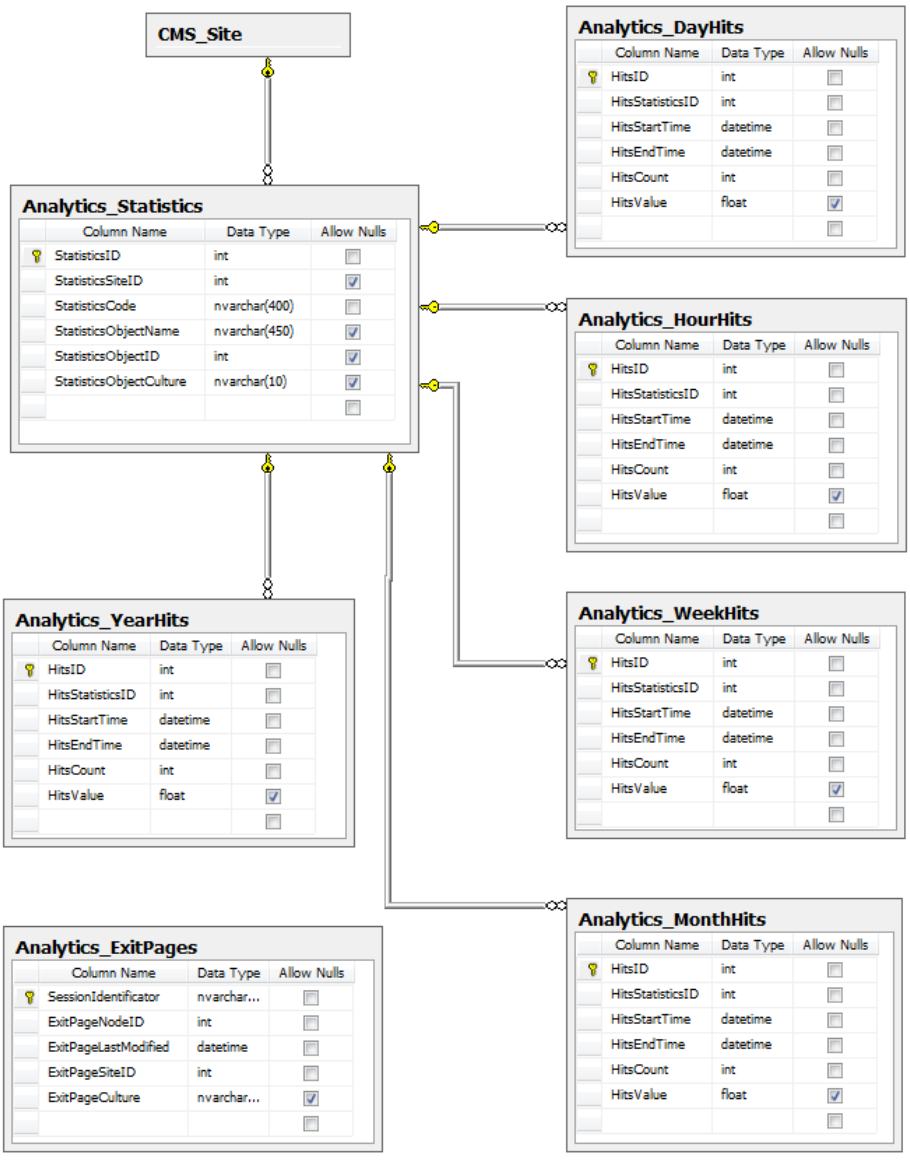
event within a certain context, i.e. related to a specific object, site and culture.

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.

8.49.10.3 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.

8.50 WebDAV integration

8.50.1 Overview

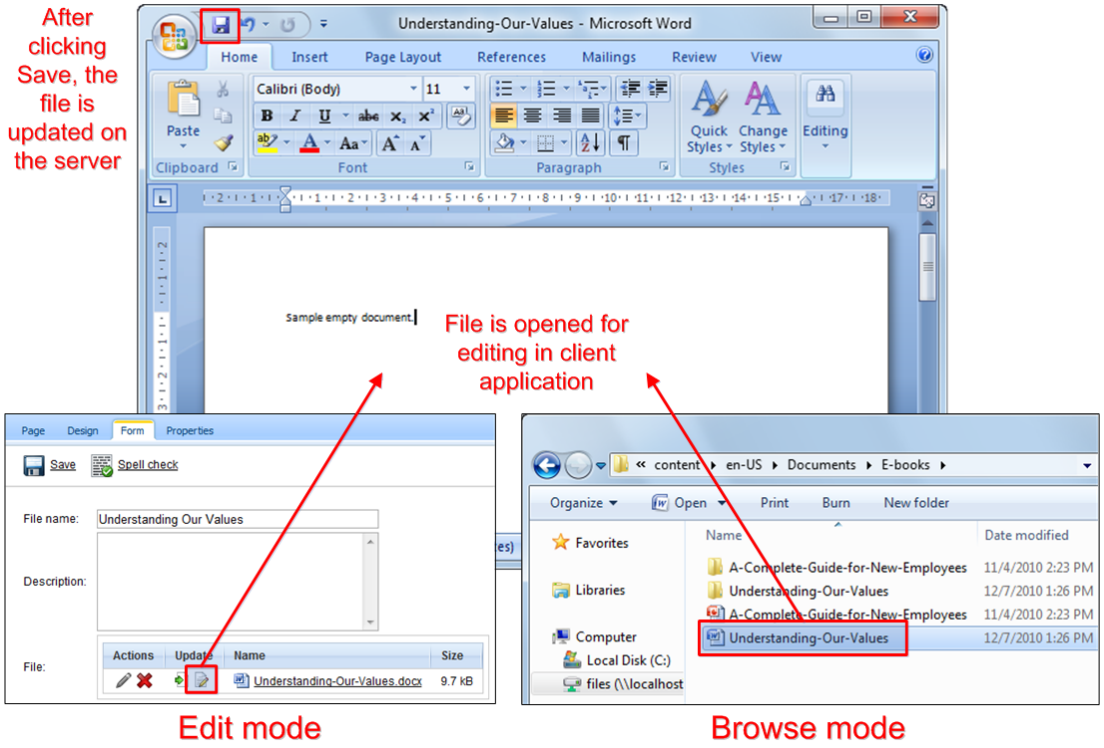
WebDAV stands for *Web-based Distributed Authoring and Versioning*. It is a set of extensions to the HTTP protocol which allows users to collaboratively edit and manage files on remote web servers. Detailed information, specifications and documentation related to this technology can be found on its official 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.

8.50.2 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.

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.

8.50.3 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.

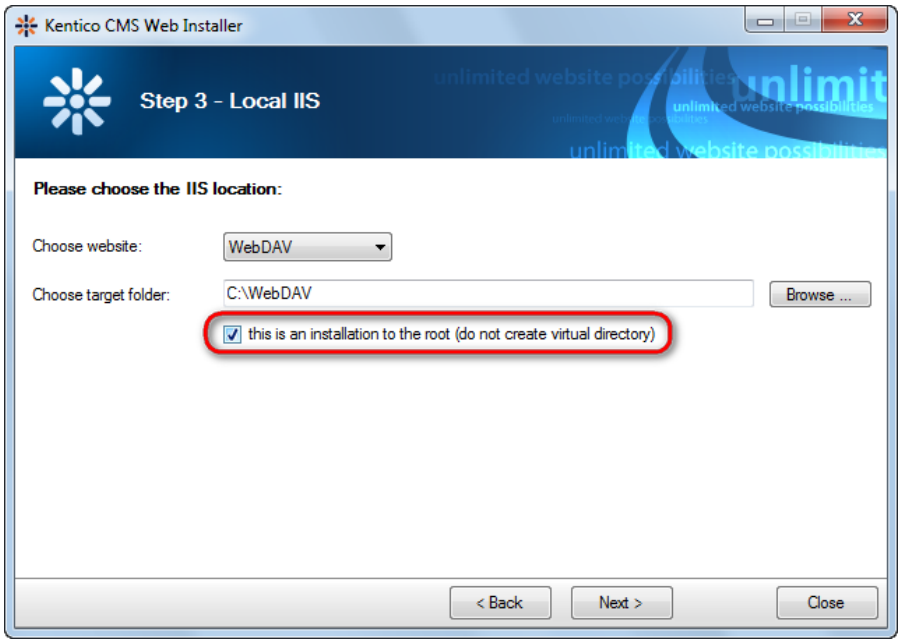
Step 1 on IIS versions prior to 7.0

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.

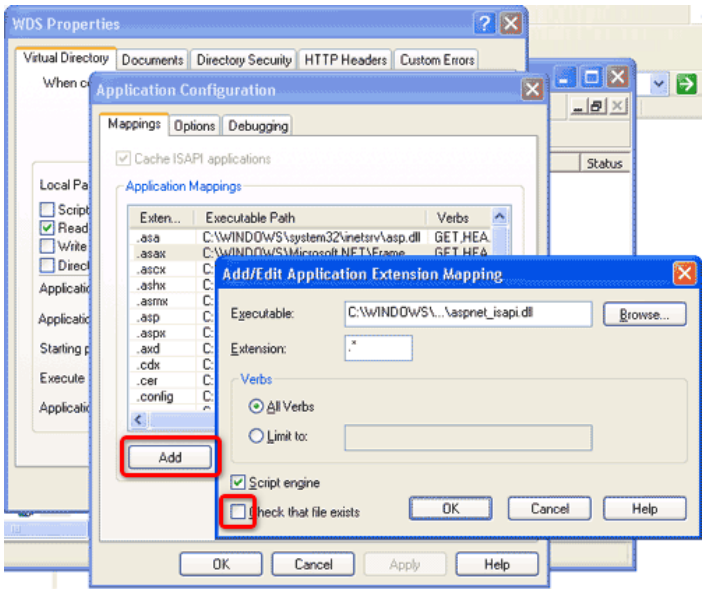


When using IIS versions older than 7.0, it is also necessary to correctly map the **aspnet_isapi.dll** file in IIS. The process of performing this configuration is described below.

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

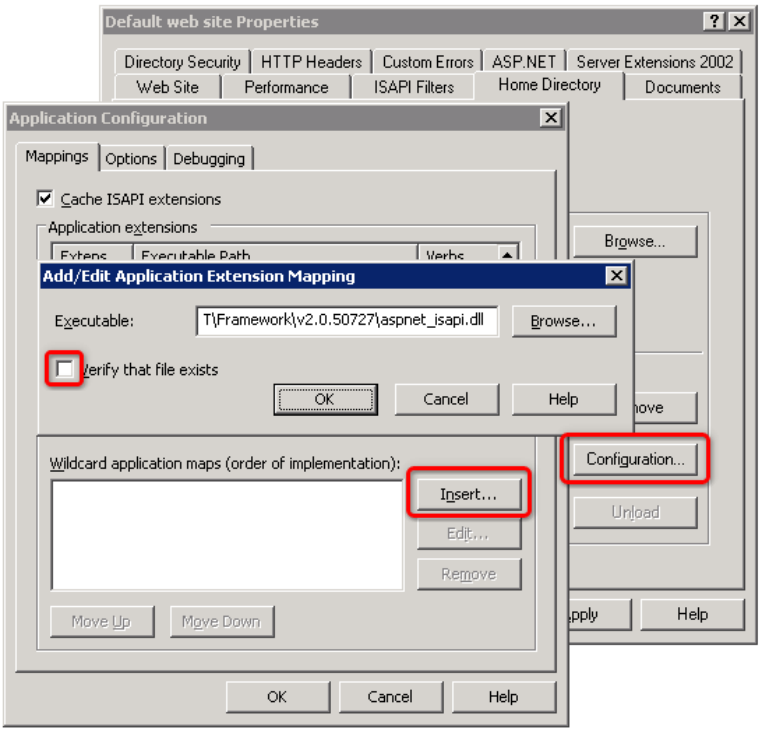


Click **OK** and return to the main management console. Now right-click the website under which your application is stored and select **Properties**. Switch to the **Home directory** tab, ensure that the **Read** box is checked and set the **Execute permissions** field to *Scripts only*.

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.

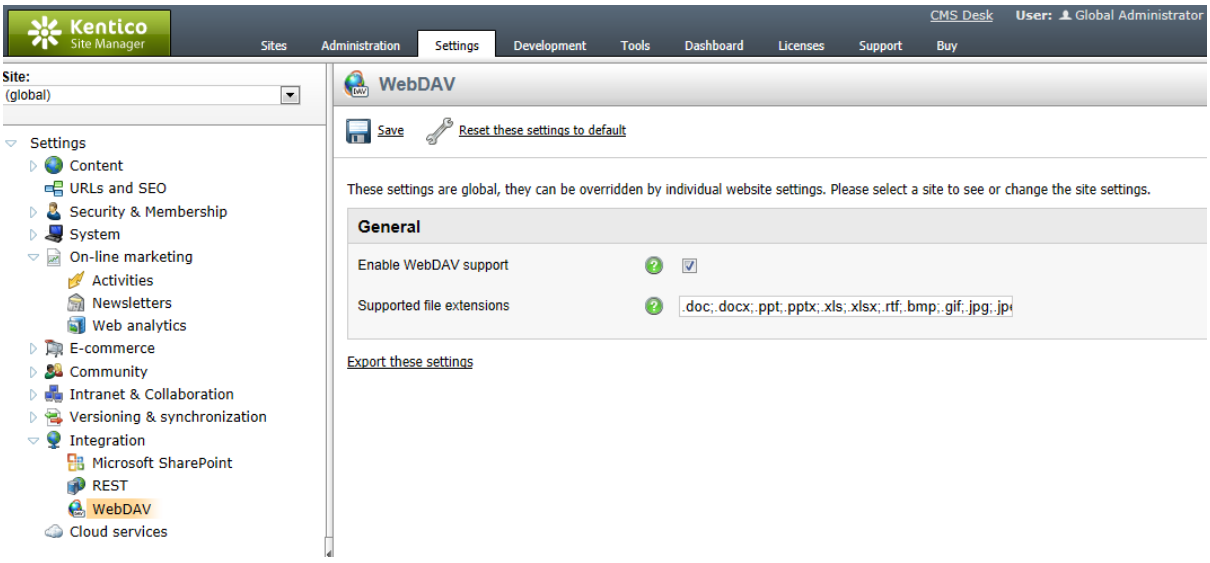
4. The last step is to enable **Windows authentication** for your website. Please follow the instructions in Development -> Membership -> Authentication -> Windows authentication -> Configuring Windows authentication.

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.

8.50.4 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.



8.50.5 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.

8.50.6 Edit mode

8.50.6.1 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:

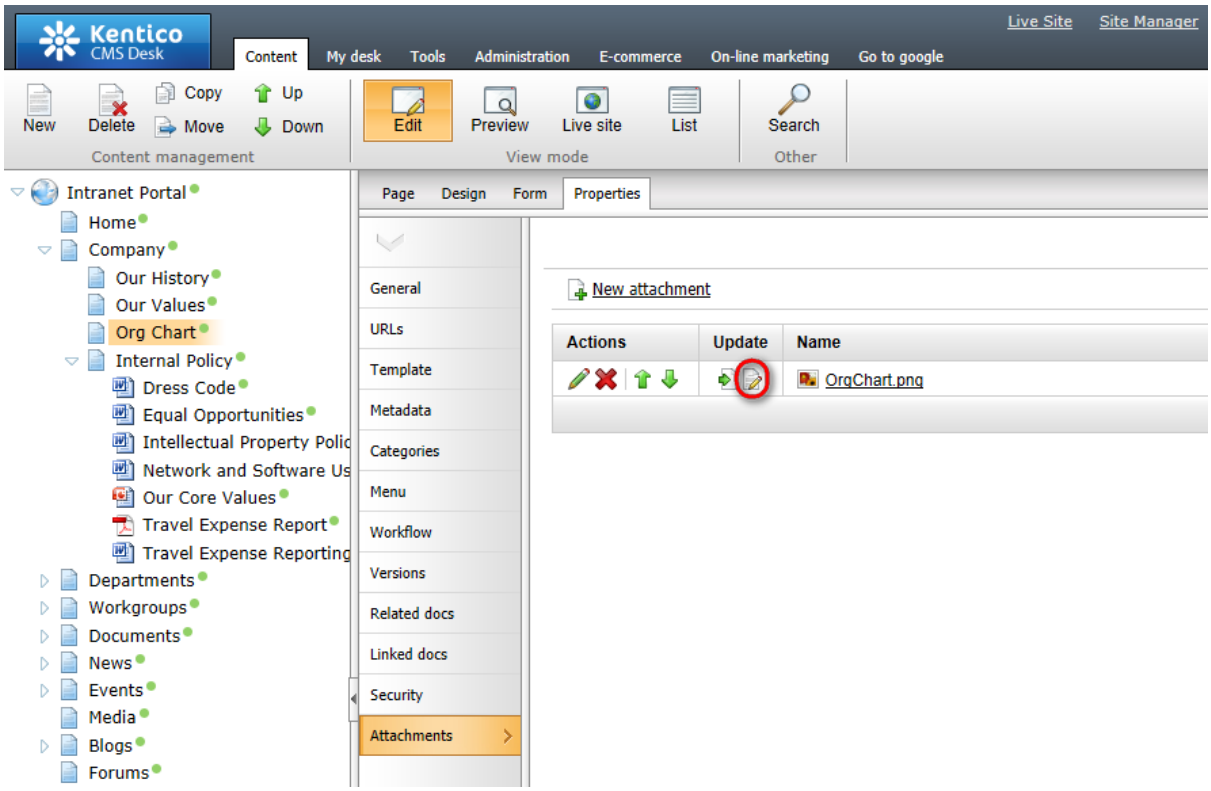
- Integration with Document library - explains how live site users can edit documents stored by means of the Document library module.
- Integration with User contributions - explains how live site users can edit files uploaded as part of documents submitted via the User contributions module.

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.

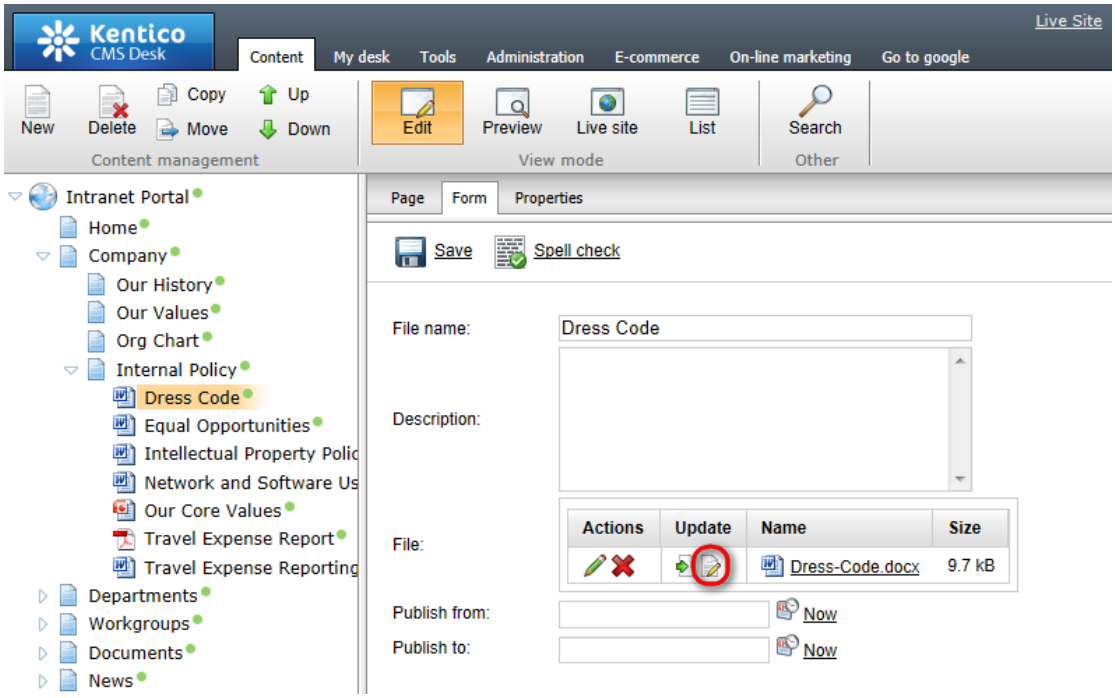
8.50.6.2 Editing files using WebDAV

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**.

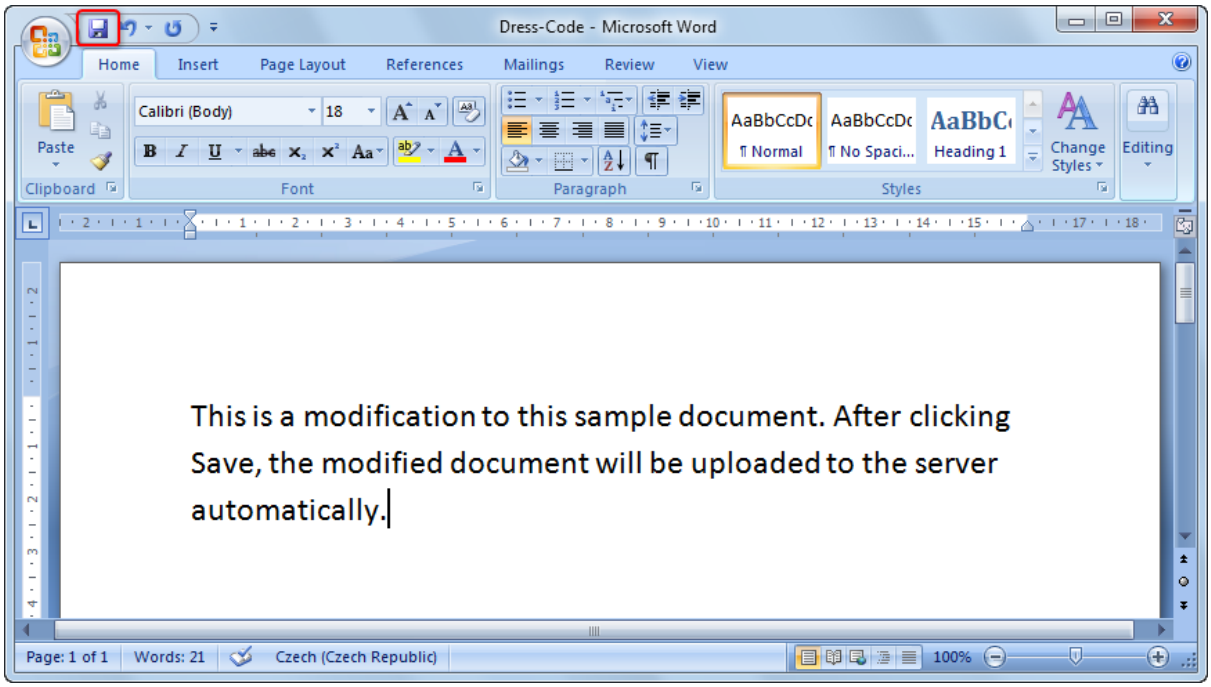


2. The same **Edit in client application ( )** icon is displayed on the **Form** tab of documents using the File field (e.g. *CMS.File* document type) where a file with a supported extension is uploaded.

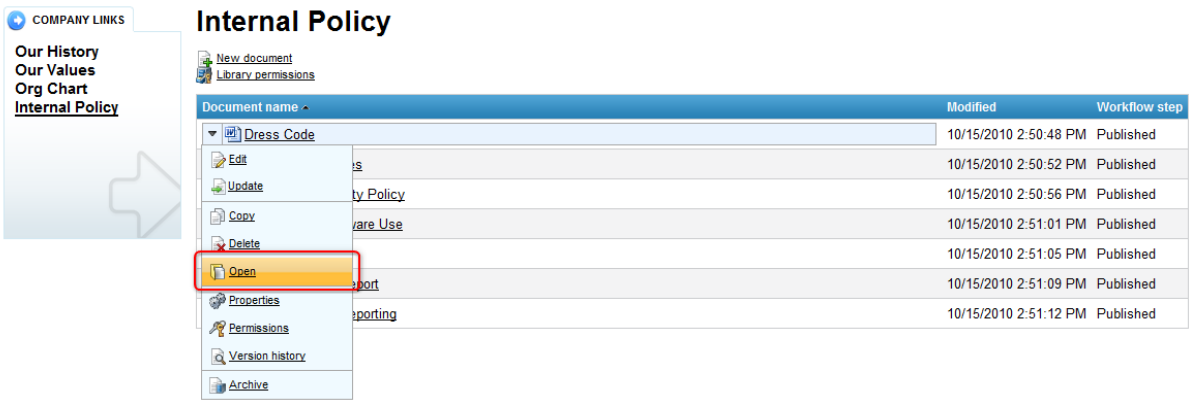


3. Clicking the icon opens the document in a client application associated with the particular file extension (e.g. *.docx* documents are opened in MS Office Word). Try editing the opened file and save

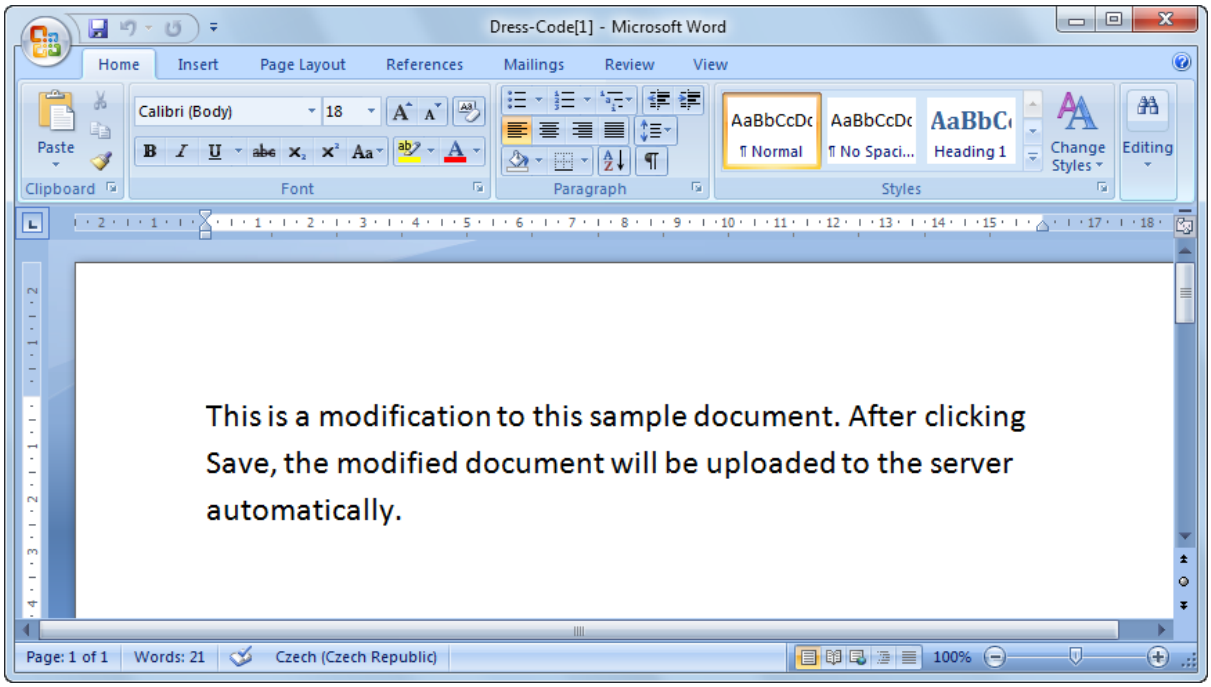
your changes using the **Save** button.



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

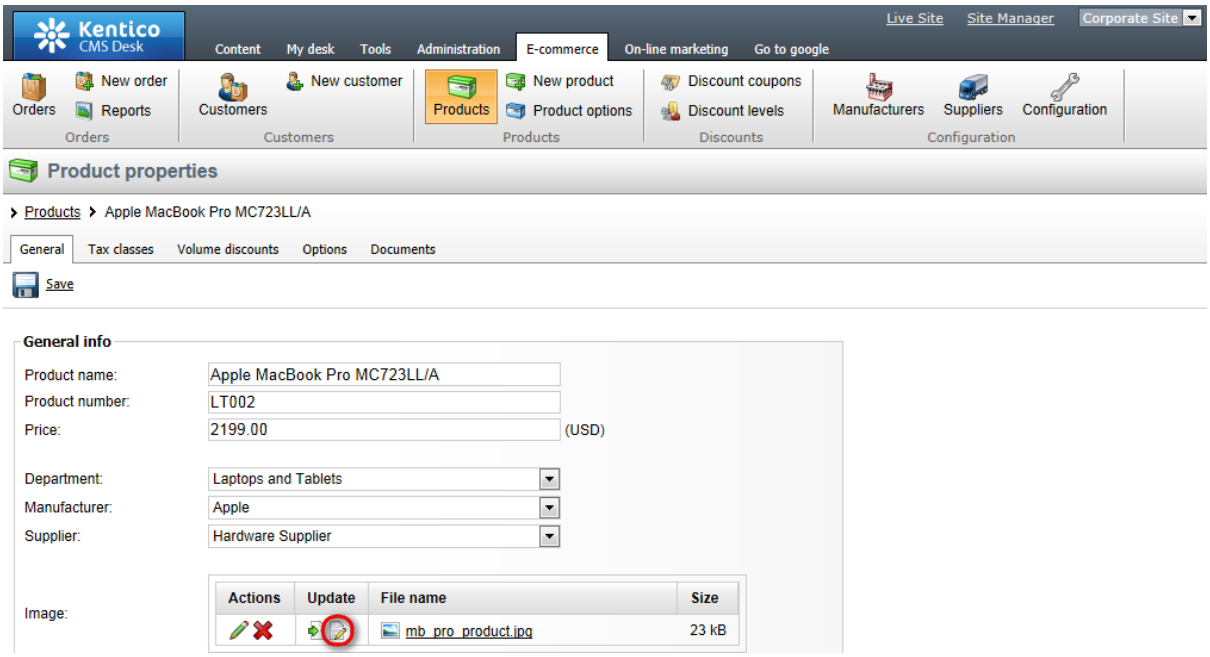
For the **Edit in client application ( )** icon to be enabled, the **Modify** permission for the current document must be granted to the current user (or one of their roles) on one of the three levels described in Development -> Membership -> Permissions -> Document permissions.

8.50.6.3 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).



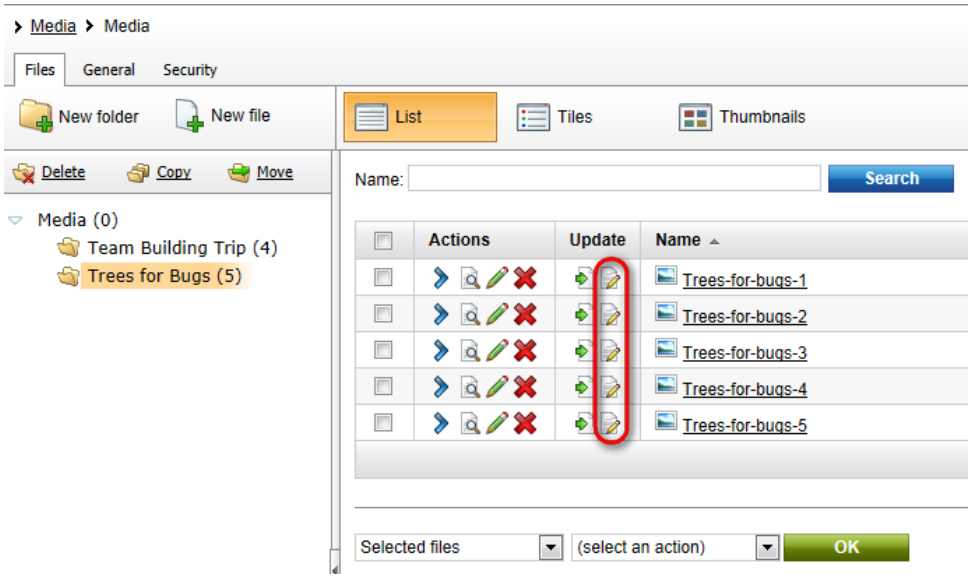
Required permissions

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**.		

8.50.6.4 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).

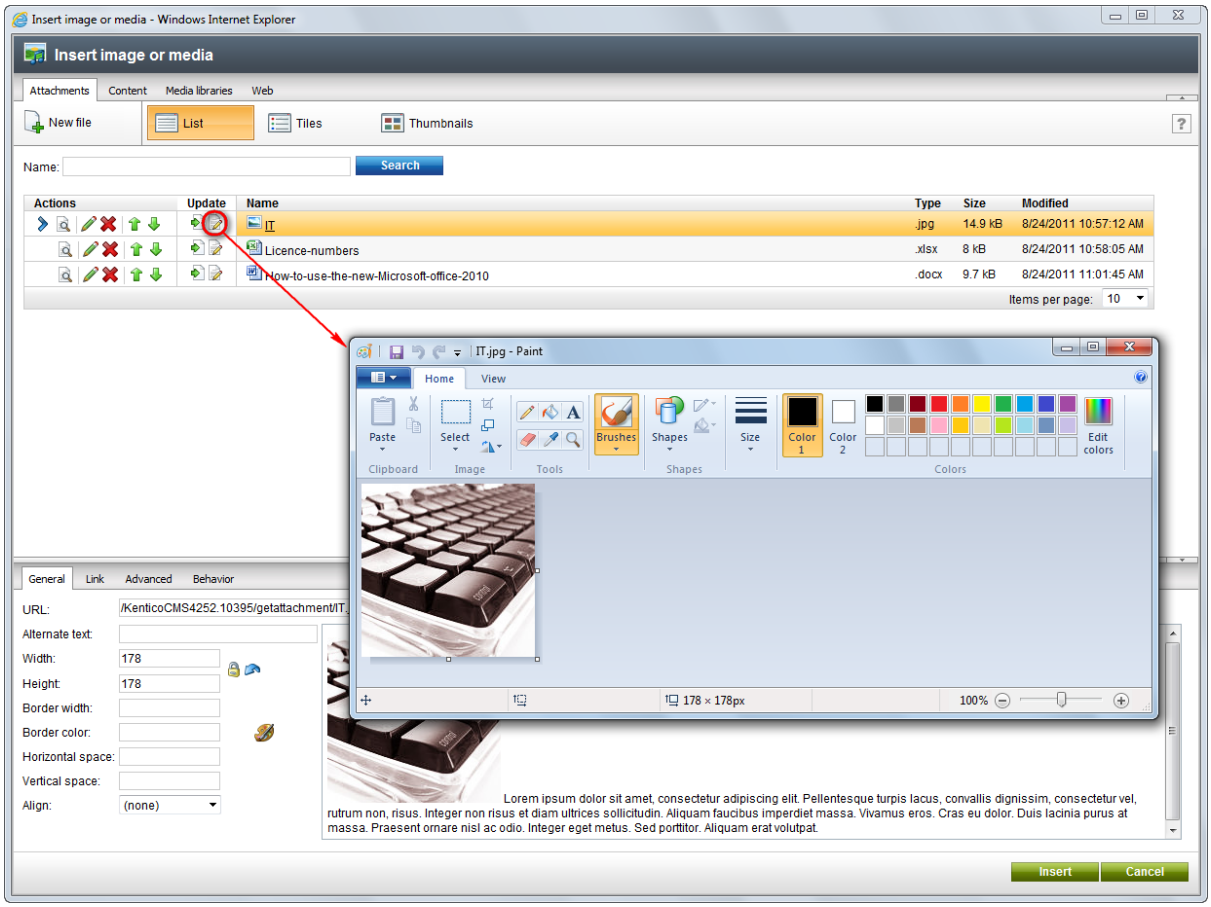


Required permissions

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.

8.50.6.5 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).



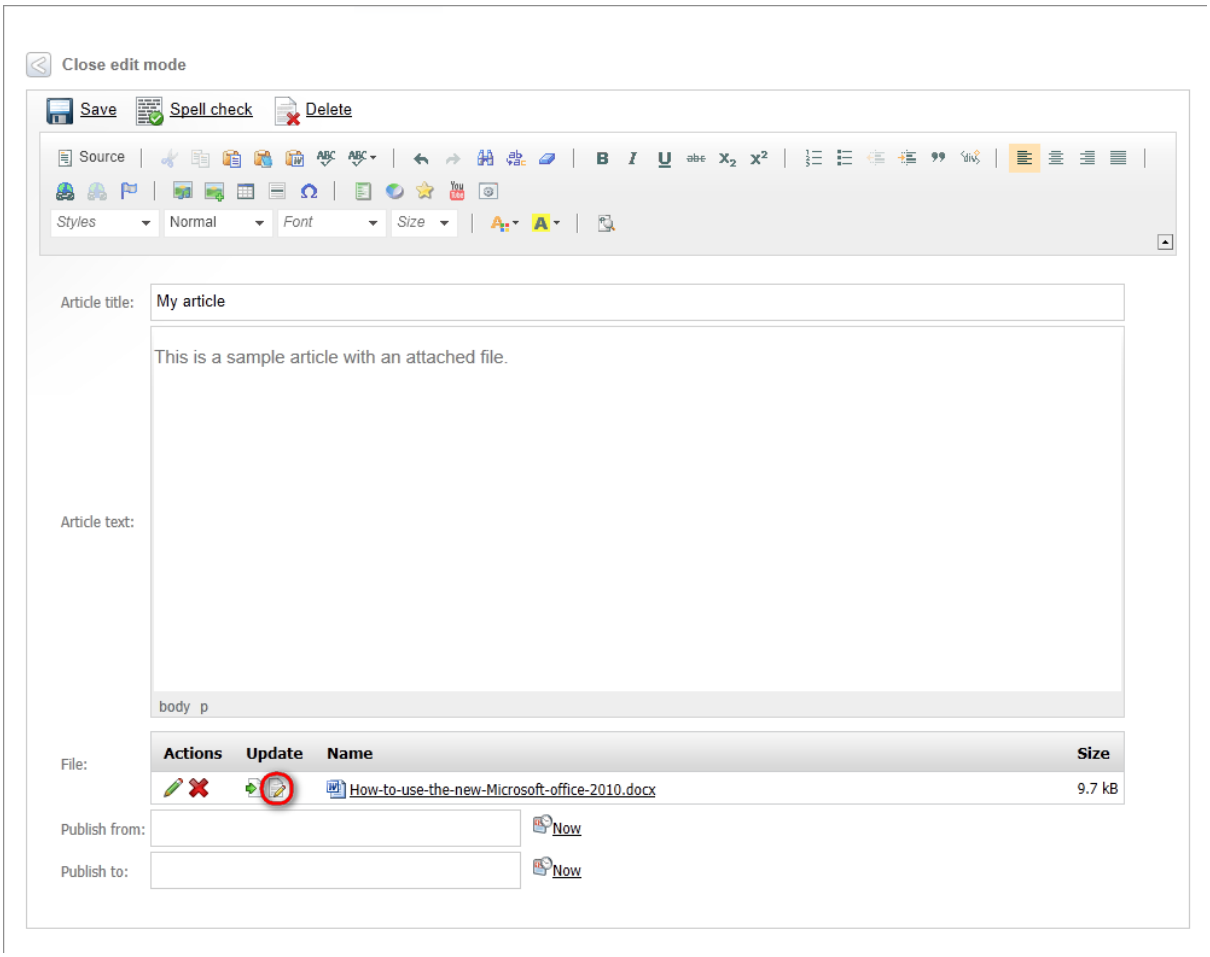
Required permissions

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.

8.50.6.6 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).

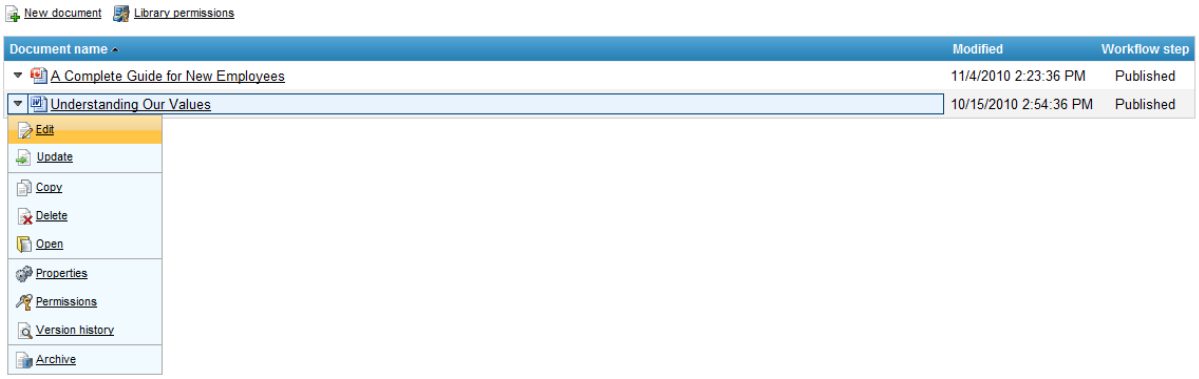


Required permissions

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.

8.50.6.7 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.

8.50.7 Browse mode

8.50.7.1 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.

8.50.7.2 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
```

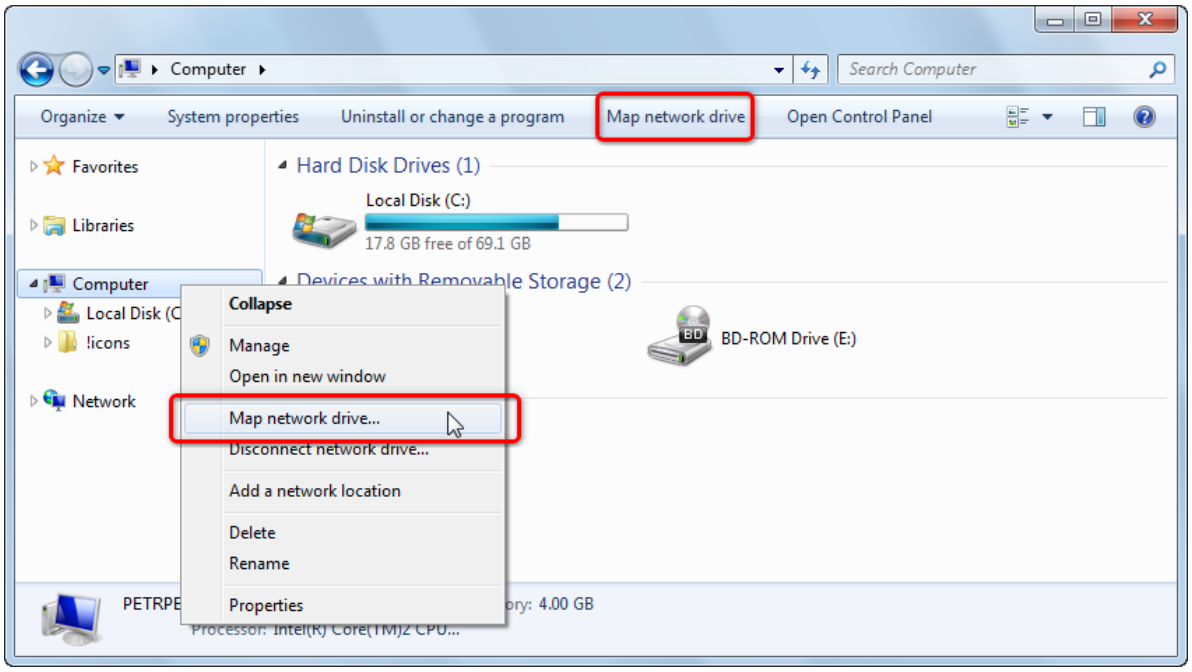
The *x:* part of the command determines which letter will be used for identification of the network drive in your system. Instead of *x*, you can use any letter that is not used for another drive yet.

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.

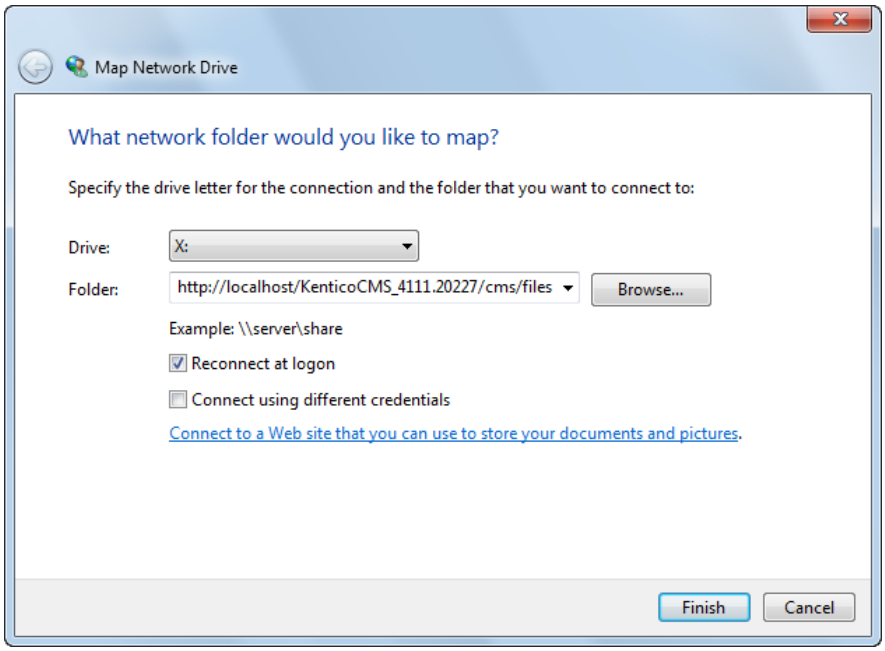
1. Open **Windows Explorer** and select **Computer** in the left menu. Right-click it and select **Map network drive**. Alternatively, the **Map network drive** action can be executed from the top menu, as highlighted in the screenshot below.



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

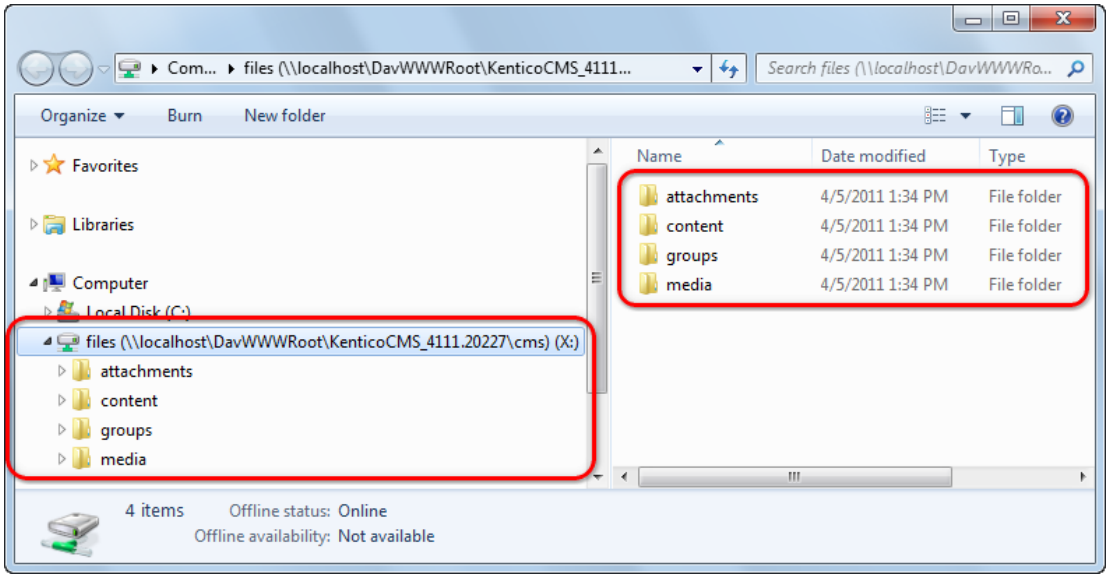
Once you have the network drive connected, it is recommended to check that you can access it. Open

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**.

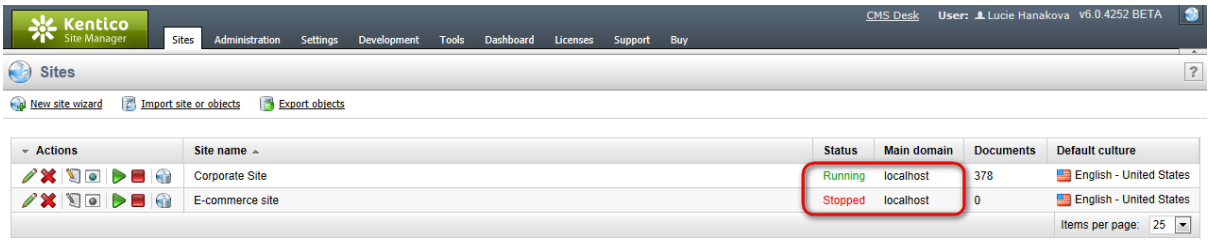
See the following article for more details: http://support.microsoft.com/kb/2445570

Multiple websites using the same domain

In a typical scenario, each website running in Kentico CMS uses a different domain. Therefore, the URL

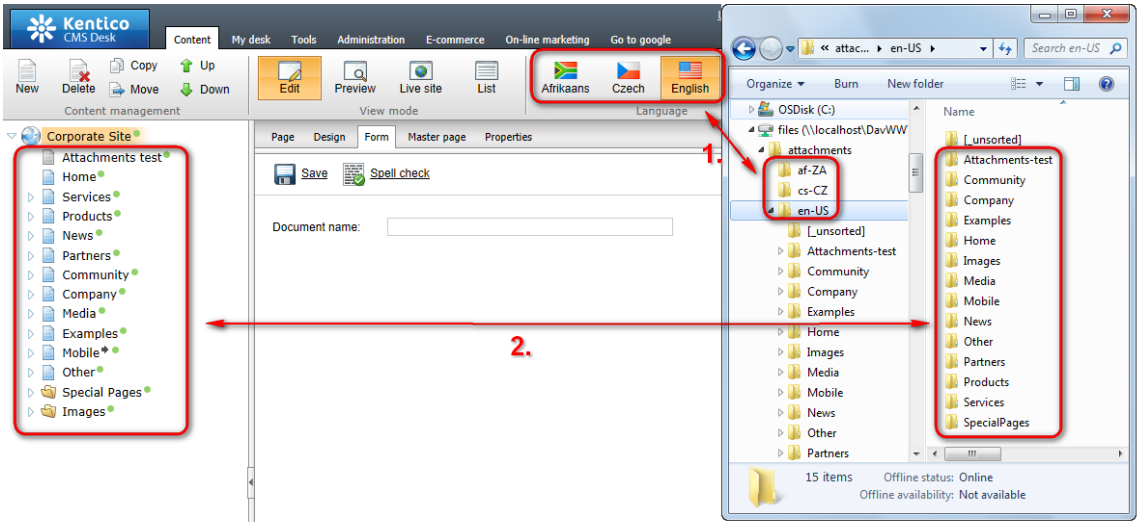
of the mapped network drive would be different for each site, letting you map a drive separately for each website.

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.

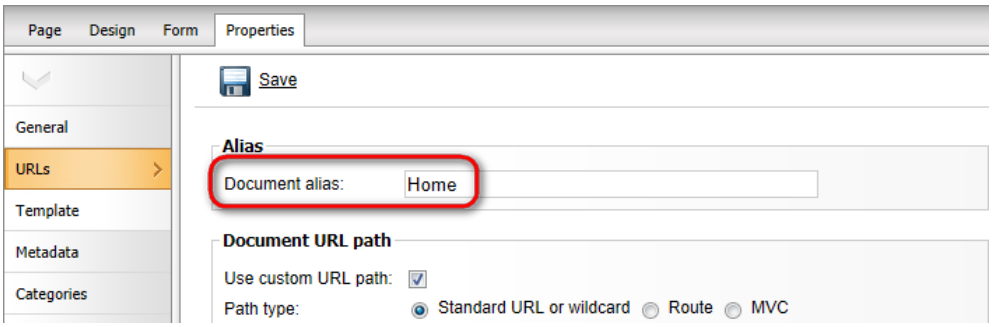


8.50.7.3 Attachments

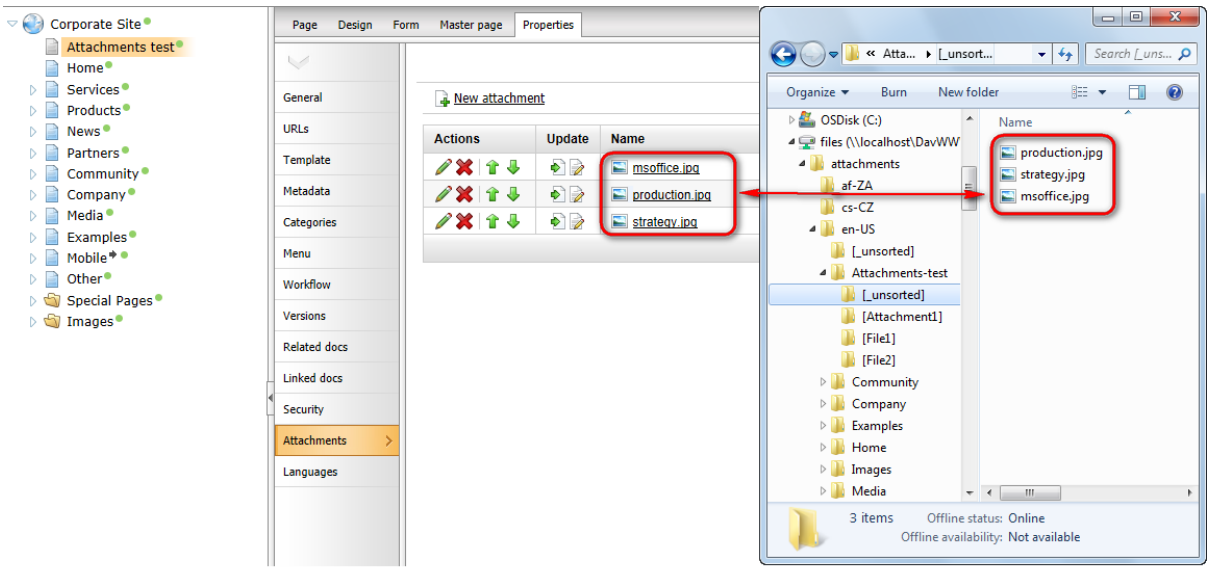
The **attachments** folder in the root of your WebDAV network drive contains document attachments of all documents in the website's content tree. Under the **attachments** folder, there is one folder for each culture used for the website (1. in the screenshot below). Under each of the folders, you can find one folder for each document in the content tree in the particular language version (2. in the screenshot below).



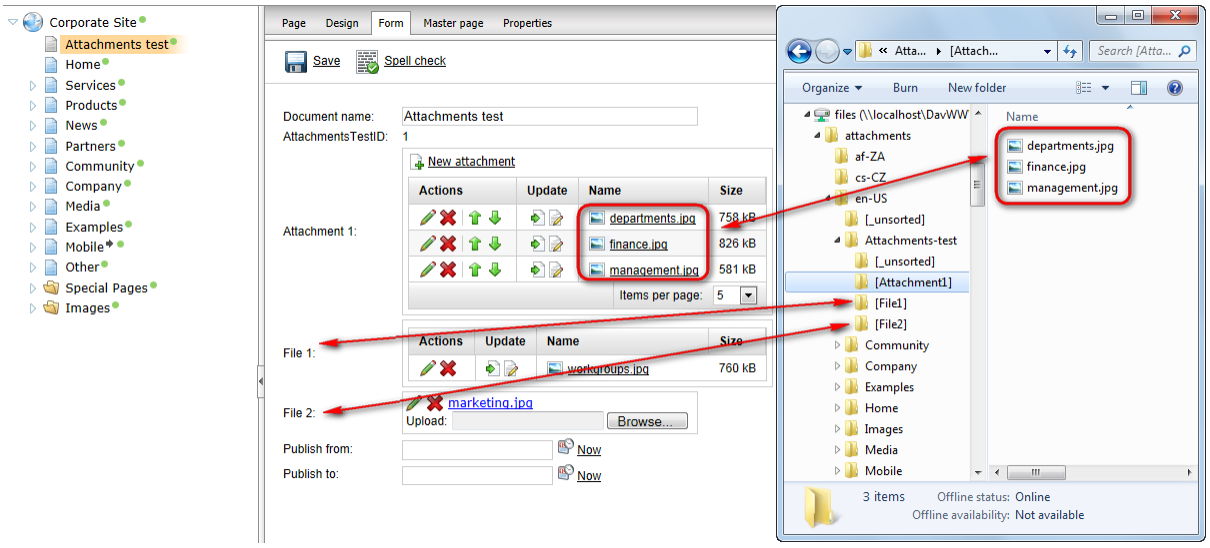
Please note that **folder names reflect node aliases** of particular documents, **not their document names**. This is necessary for unique identification of documents in URLs. Node aliases of documents can be viewed and modified on the **Properties -> URLs** tab, in the **Document alias** field.



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

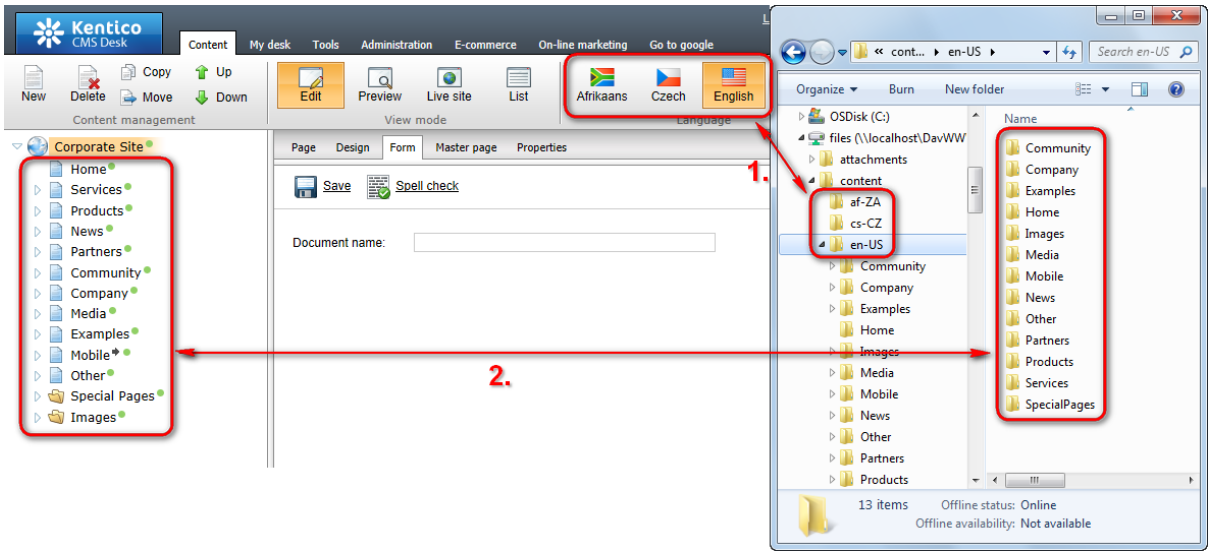
The following document permissions need to be granted to users (their roles) in order to perform respective actions with document attachments:

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.

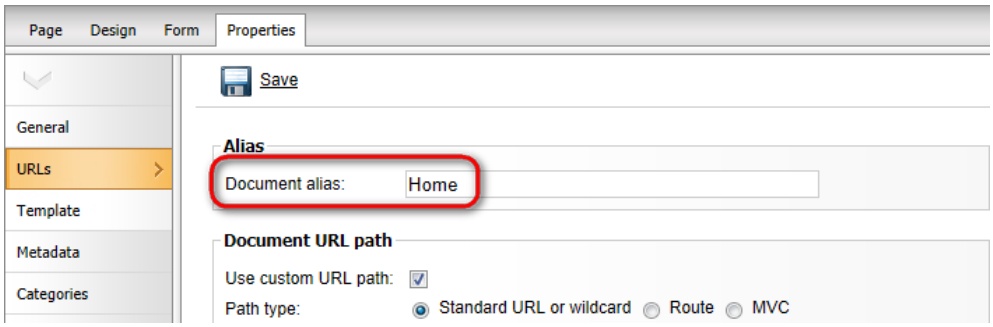
Please note that because WebDAV only works with Windows Authentication, the account used to log on to Windows needs to be imported as a user account in Kentico CMS. If you grant permissions to roles where the CMS user is member, Browse mode editing for the Windows account will be allowed.

8.50.7.4 Content

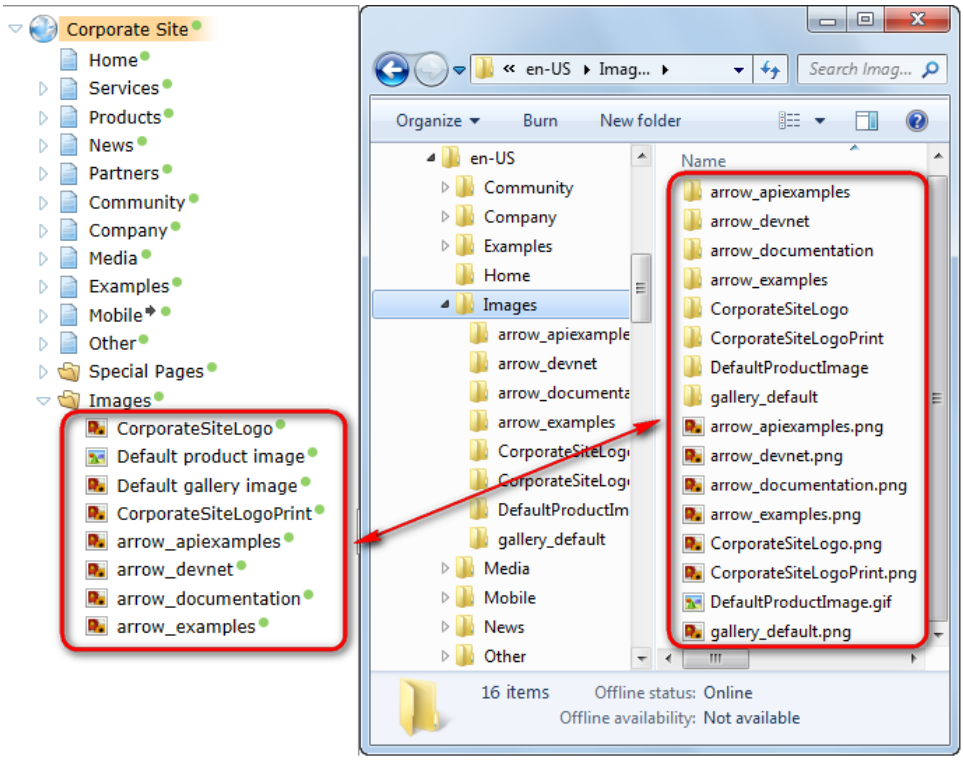
The **content** folder contains documents of the **CMS.File** document type (see Content management -> File management for more details). Under the **content** folder, there is one folder for each culture used for the website (1. in the screenshot below). Under each of the folders, you can find a folder 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

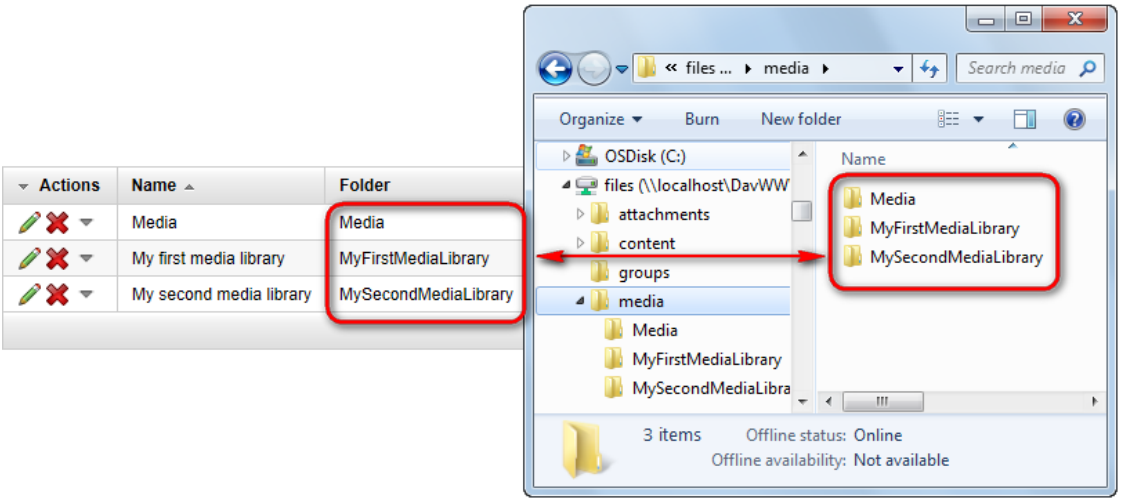
The following document permissions need to be granted to users (their roles) in order to perform respective actions with *CMS.File* documents:

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).

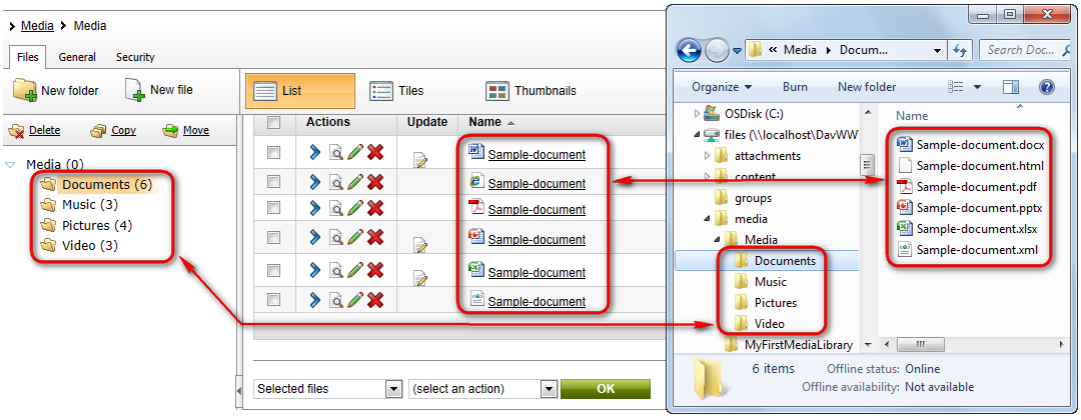
Please note that because WebDAV only works with Windows Authentication, the account used to log on to Windows needs to be imported as a user account in Kentico CMS. If you grant permissions to roles where the CMS user is member, Browse mode editing for the Windows account will be allowed.

8.50.7.5 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.



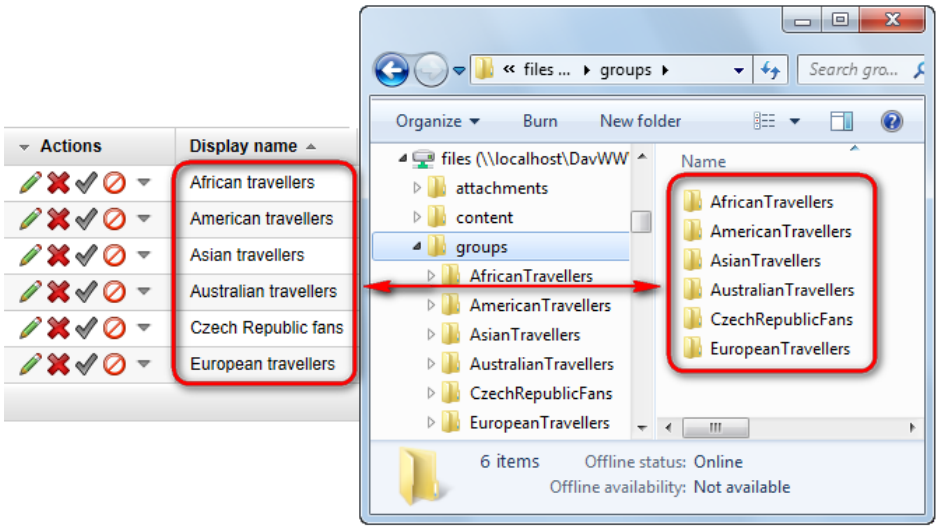
Possible actions and required permissions

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.

8.50.7.6 Groups

The **groups** folder contains a sub-folder for each group on the website.



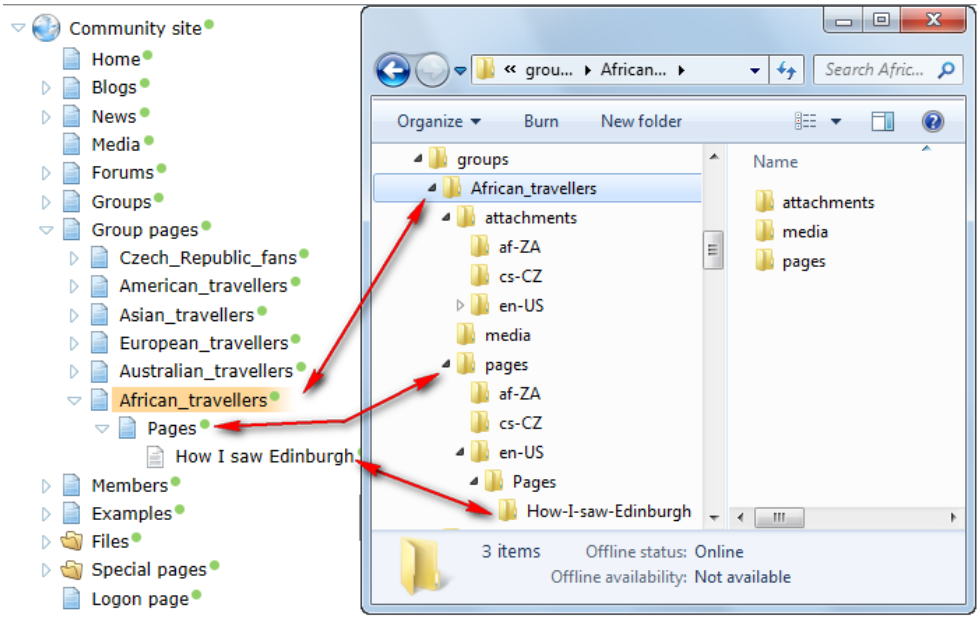
Under each group folder, you can find three folders: attachments, **pages** (similar behavior as the content folder in the network drive root) and media. The content and behavior of these folders is identical to the three folders of the same name that are located in the WebDAV network drive root. Click 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

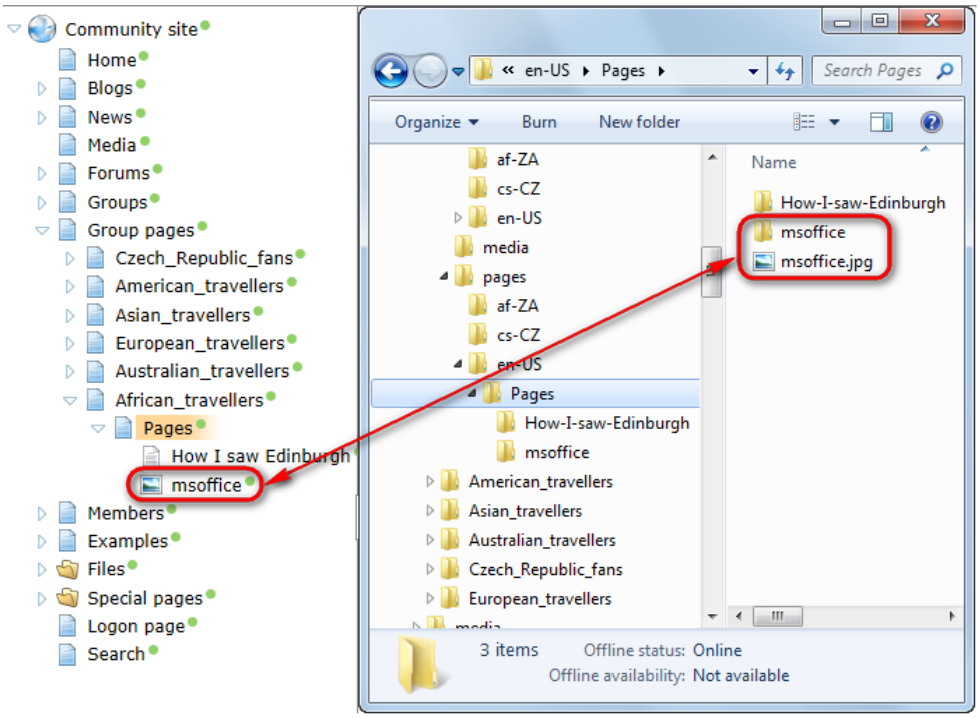
African_travellers document, while the **[MenuItemTeaserImage]** folder represents the **Menu item teaser image** file field of the **African_travellers** document. The same applies to the documents under it.



Group pages content

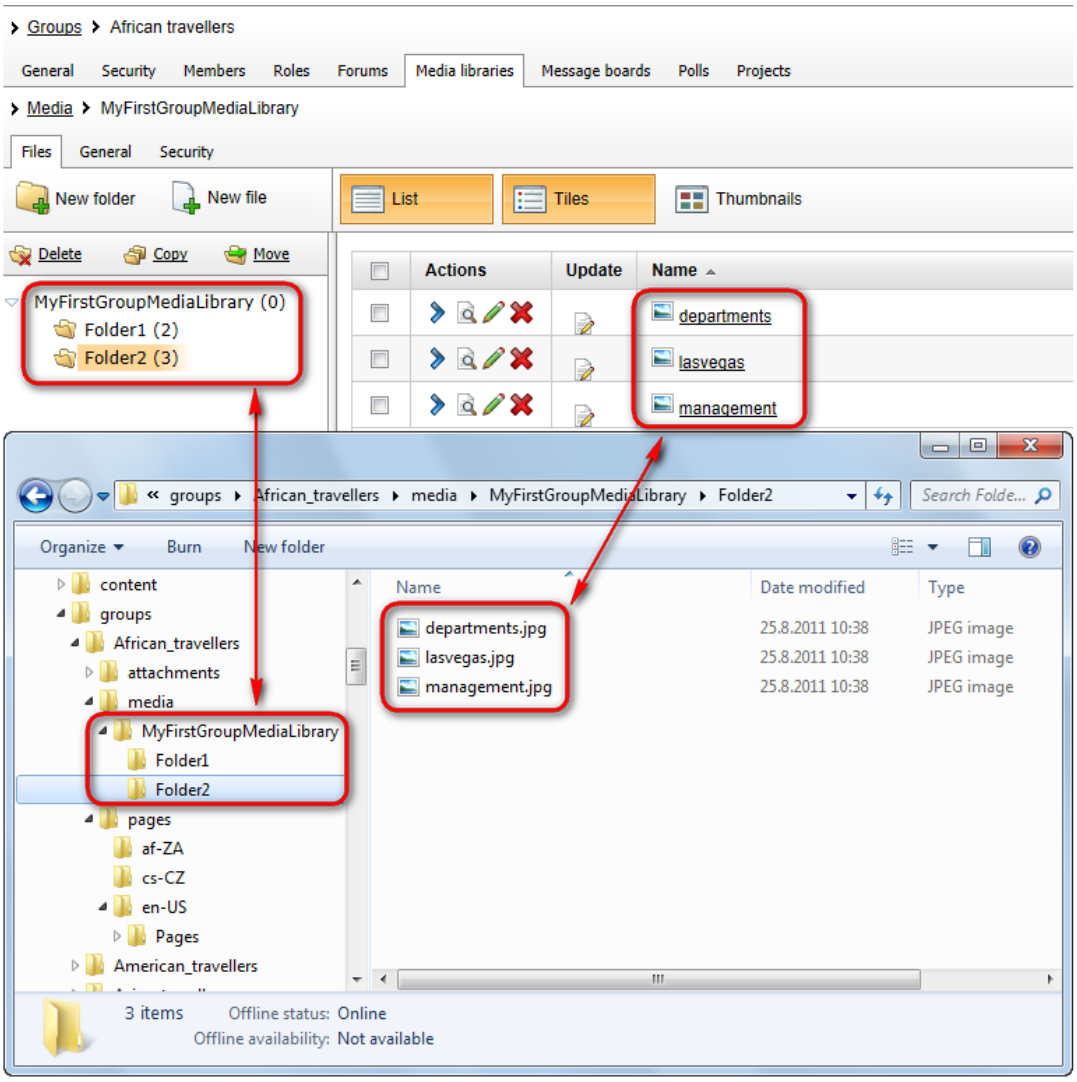
The **content** folder also contains a sub-folder for each website culture. Similarly to the content folder in the WebDAV network drive root, these folders are used to provide WebDAV editing of *CMS.File* documents. The only difference is that only documents belonging to the group found in the group 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.

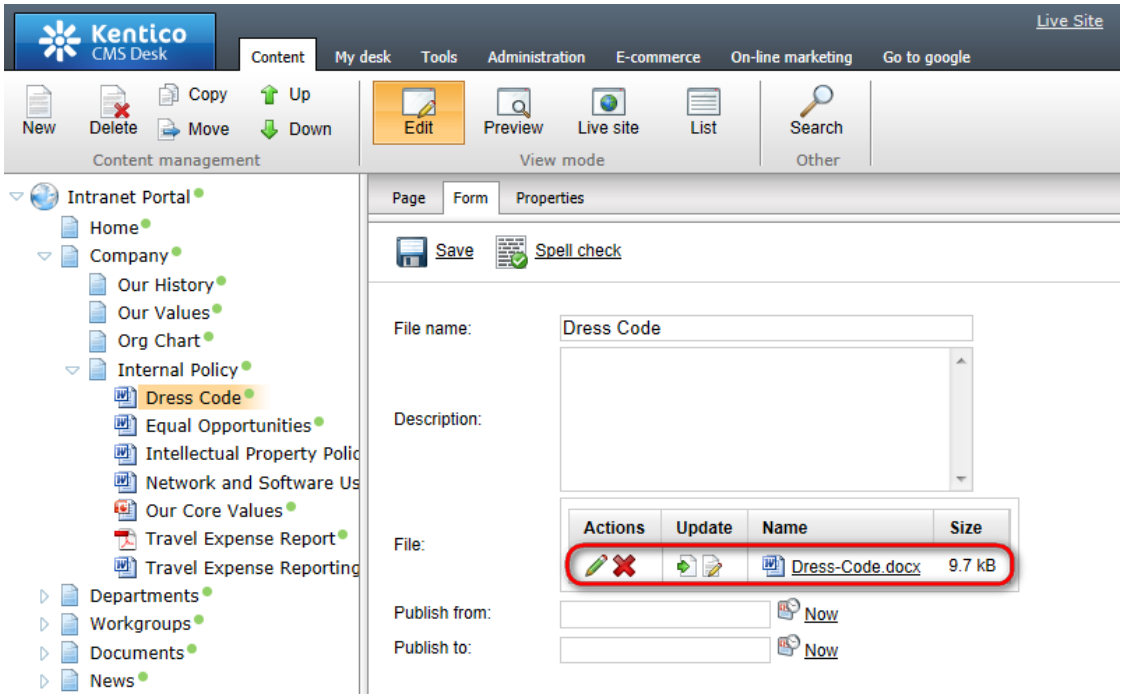
8.50.7.7 Example of Browse mode editing

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.

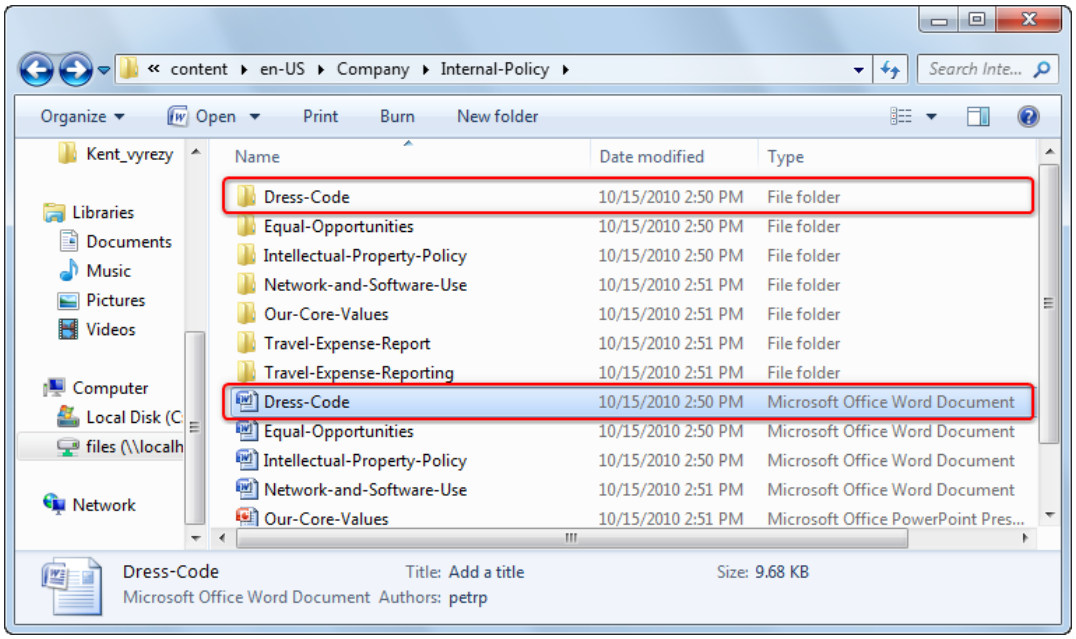
Before trying this example on your machine, please make sure that your system meets all requirements , that it is configured as described in the Configuration for WebDAV topic, that you have WebDAV settings adjusted properly and that you have mapped a WebDAV network drive.

Editing a CMS.File document

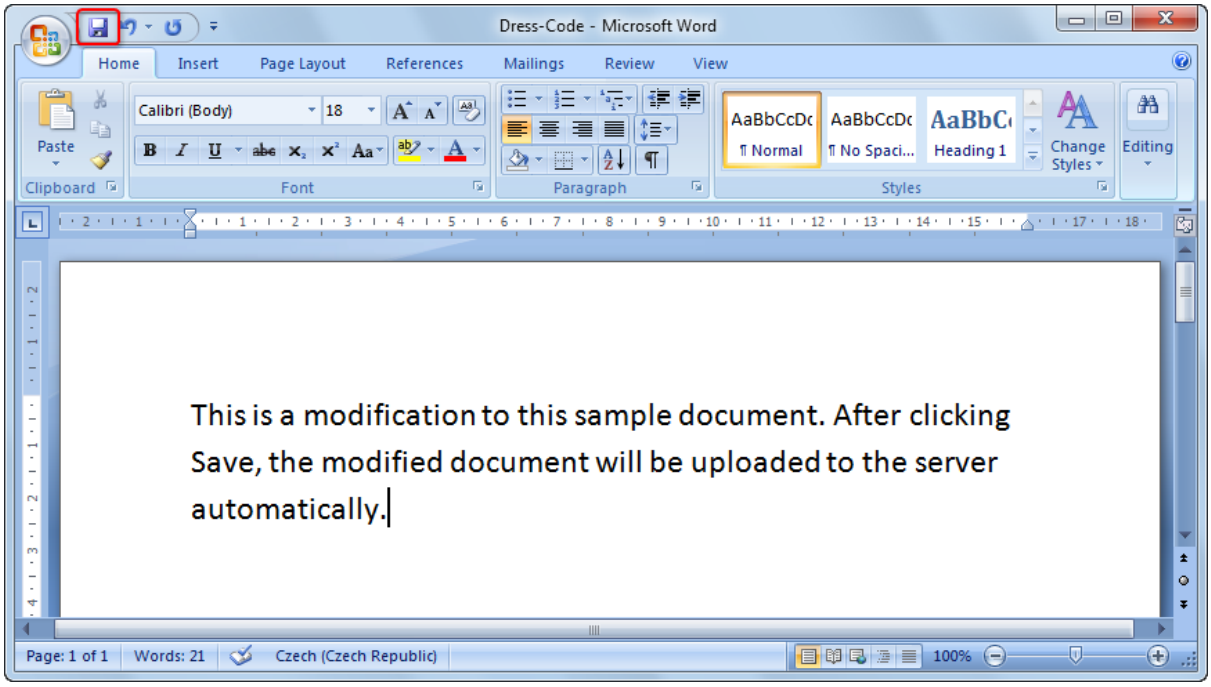
1. Go to **CMS Desk** and expand the **Company/Internal Policy** node in the content tree. From the documents under it, select **Dress Code** and switch to its **Form** tab. As it is a *CMS.File* document, you can see that it has a **Dress-Code.docx** file uploaded in its File field.



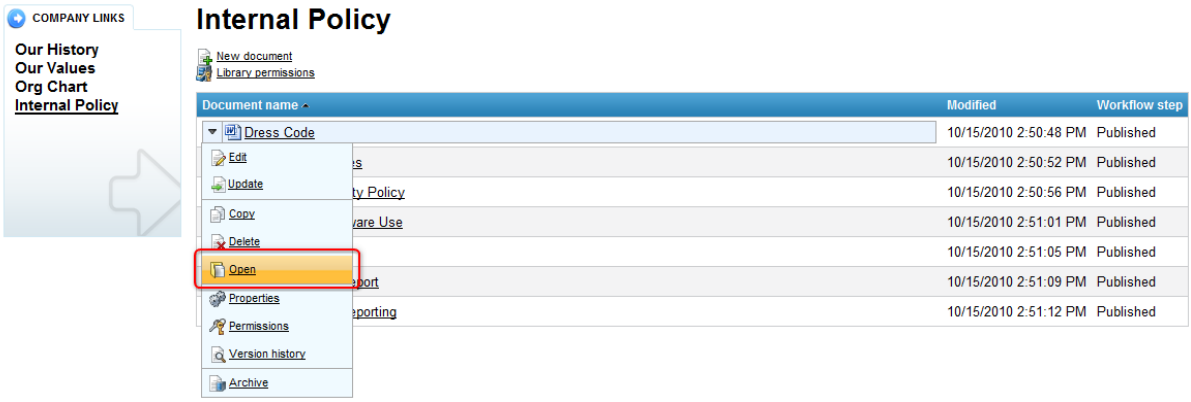
2. Let's take a look at the document on our network drive. Open **Windows Explorer** or any other file manager that you prefer. Open the WebDAV network drive and browse to *content\<your culture>\Company\Internal-Policy*. As you can see in the screenshot below, the **Dress-Code** document is represented 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.



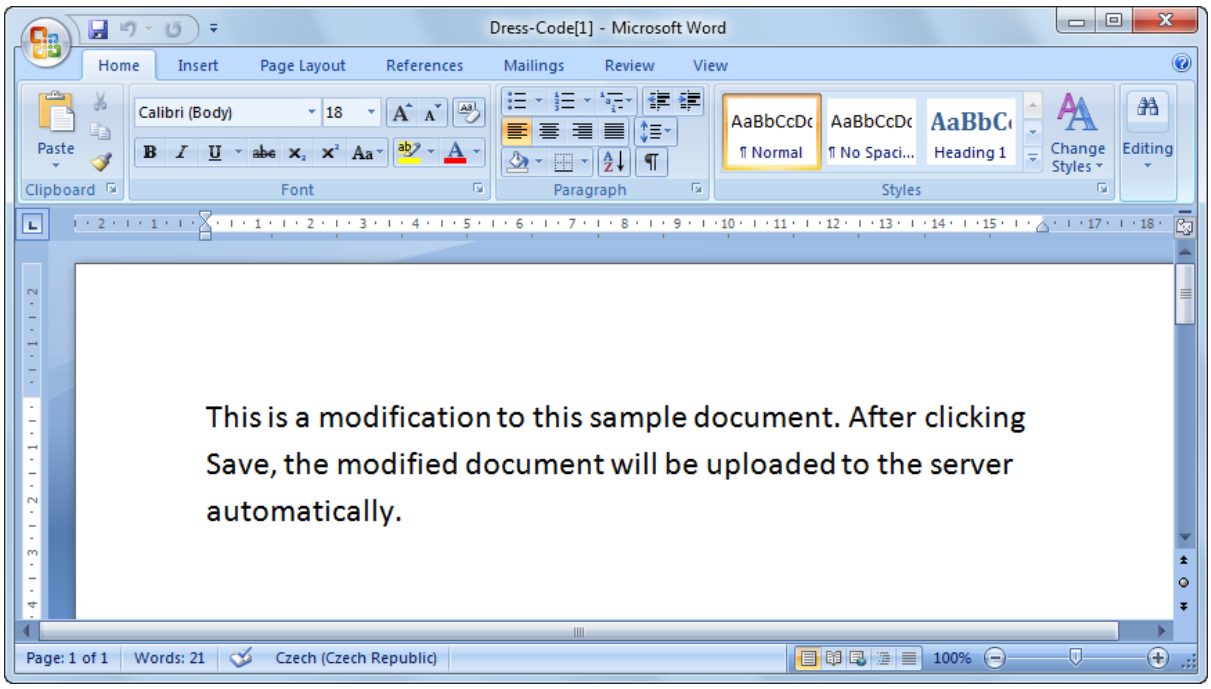
3. The **Dress-Code.docx** file gets opened in a client application (typically in Microsoft Office Word). Try editing the opened file and save your changes using the **Save** button.



4. When you have your modifications saved, try opening the file either from the UI or from the live site.

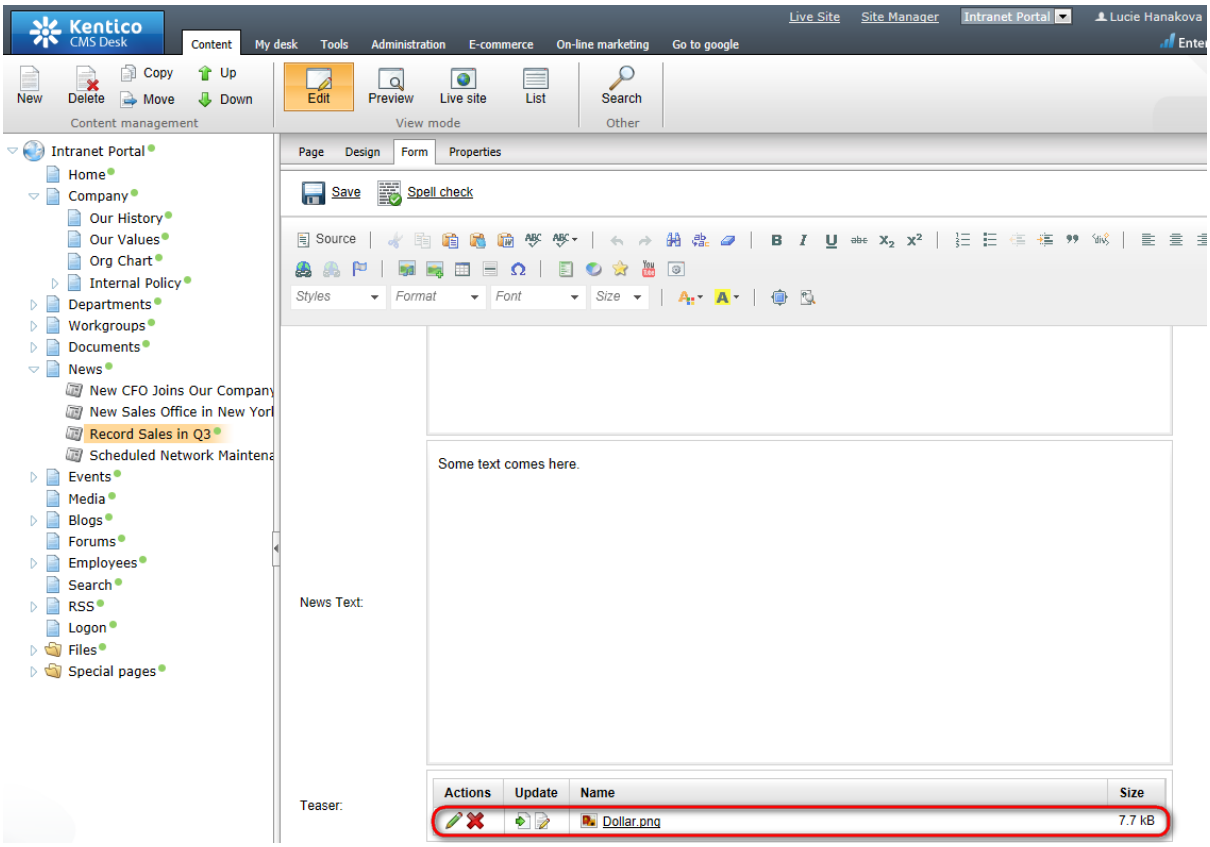


5. You should see that the opened file contains the modifications that you made to it.

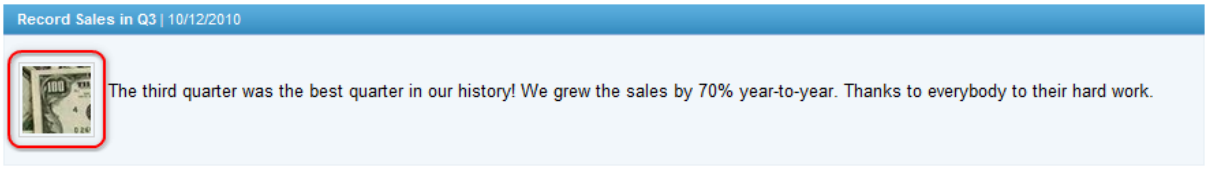


Updating a file in a File field

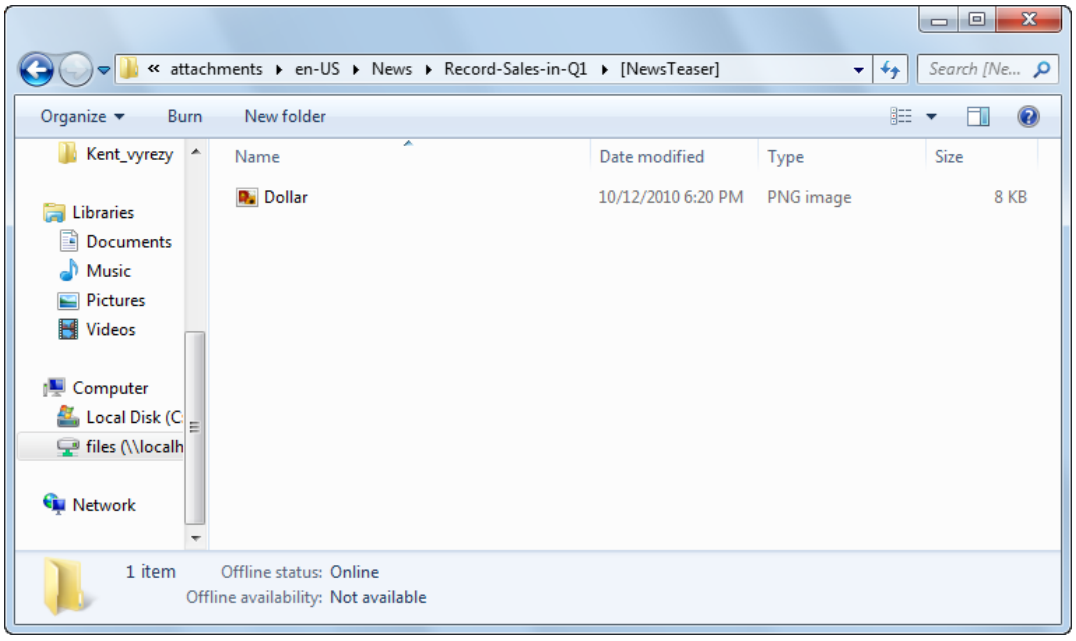
6. Next, we will try out how file uploaded in File fields can be replaced with other files. For this purpose, go back to **CMS Desk** and choose the **News/Record Sales in Q3** document in the content tree. On its **Form** tab, you can see a file uploaded in the **Teaser** File field.



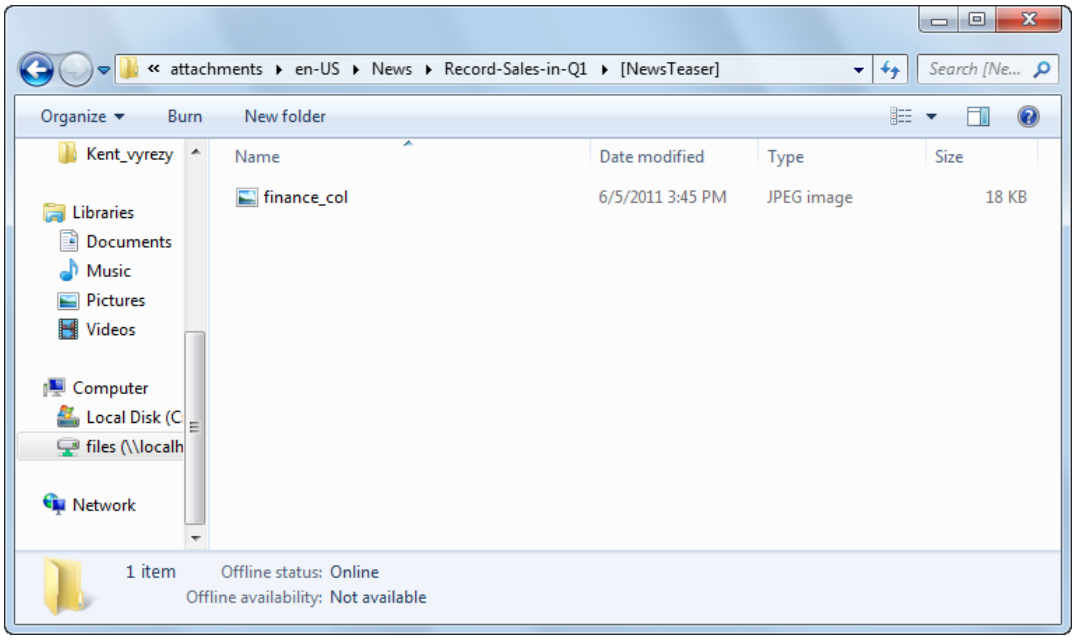
7. This file is displayed in the summary of the News item on the live site.



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.

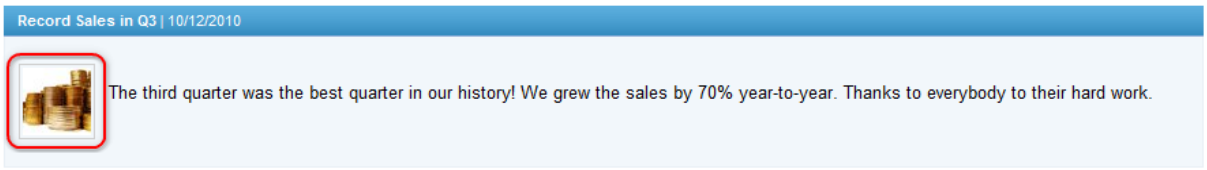


8. Try copying another image file into the folder. If you refresh the content of the folder (press F5) after a few seconds, you should see that the original file disappeared. This means that the original file in the File field has been replaced with the newly uploaded one.



9. The newly uploaded file should now also be visible in the UI ...

10. ... and on the live site.



8.50.7.8 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.

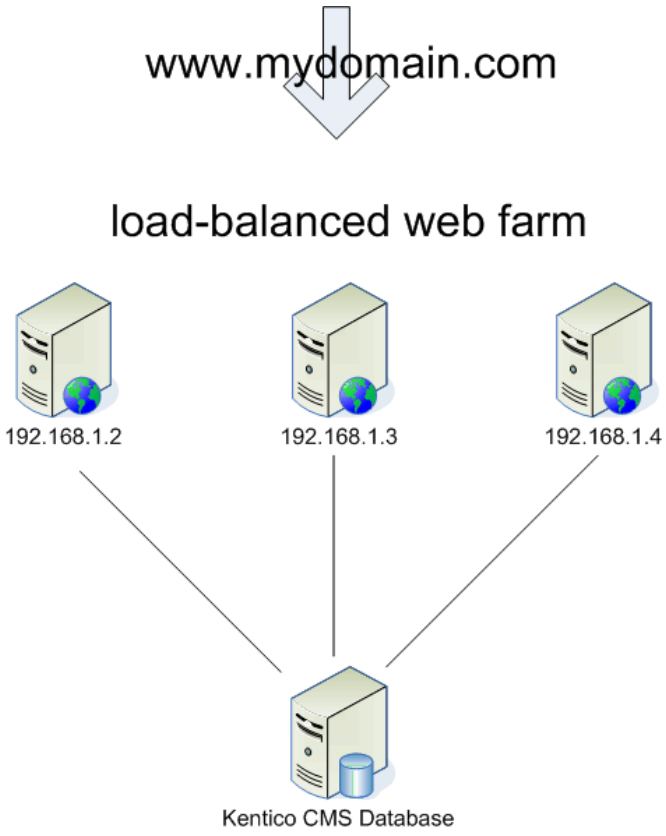
8.51 Web farm synchronization

8.51.1 Overview

Native web farm support in Kentico CMS provides the following features:

- It synchronizes the changes made to the site settings 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 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.

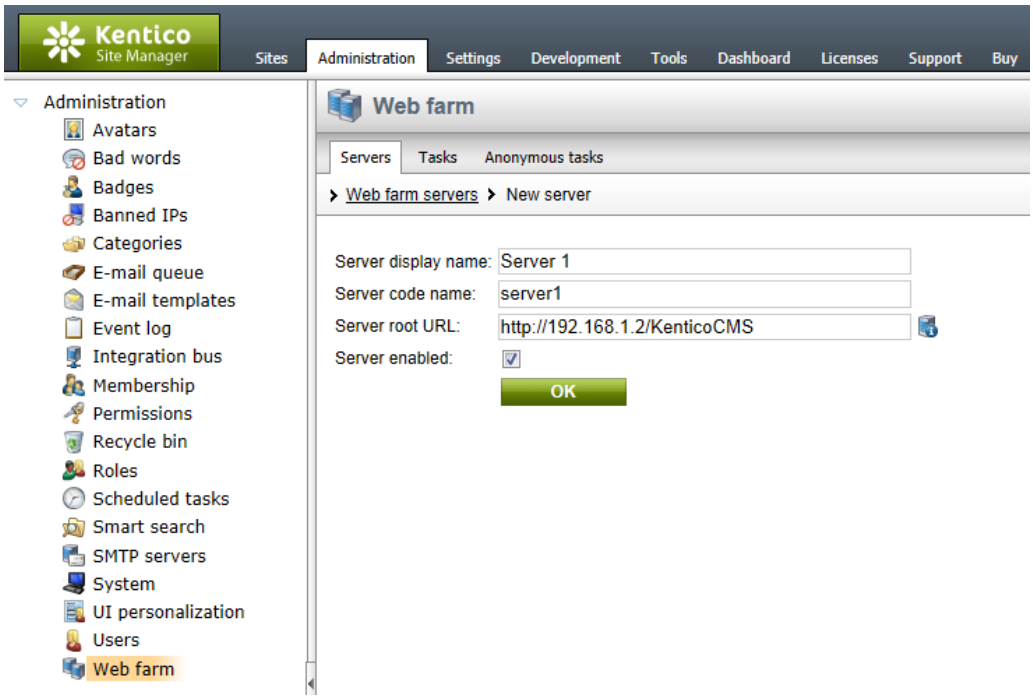
8.51.2 Defining web farm servers

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, go to **Site Manager -> Administration -> Web farm**, click **New server** and 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.* The availability of the server can be checked by using the *Check Server availability* ( ) button.

Click **OK** to save. Repeat the process for every server in your web farm.



Now you need to update the web.config file on each server and add the following keys into the **/configuration/appSettings** section:

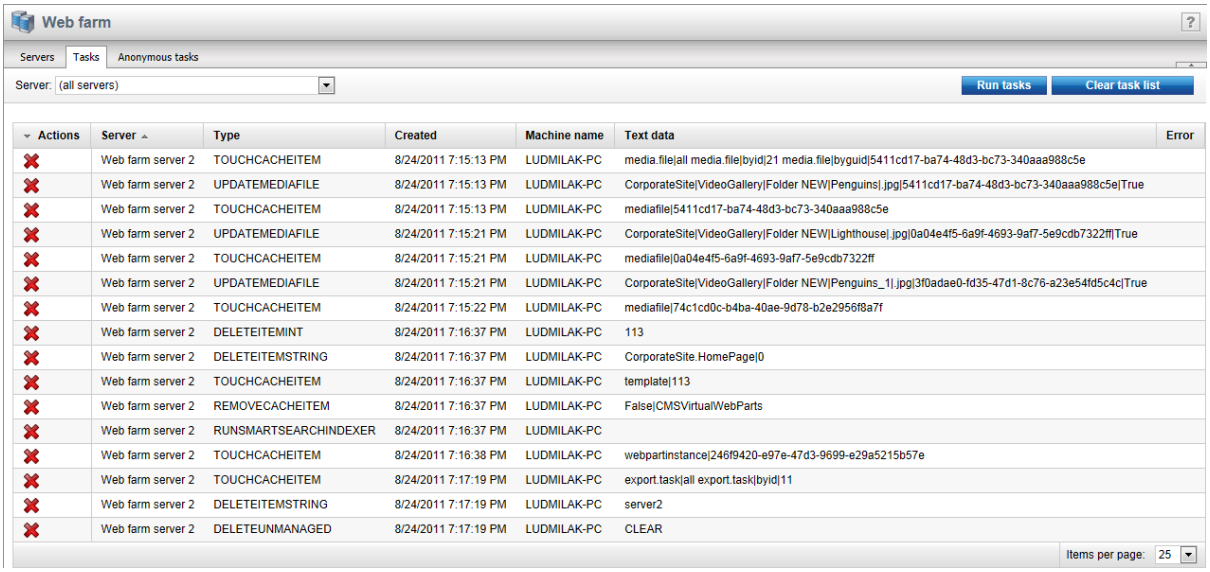
```
<add key="CMSWebFarmServerName" value="server1"/>
<add key="CMSWebFarmEnabled" value="true"/>
```

Where *server1* is the appropriate code name of the given server. Every server must contain only one such key with its own name.

Additional low-level settings of web farm synchronization can be done by adding the keys listed in the Web farm synchronization settings section of Appendix B - Web.config parameters into the **/configuration/appSettings** section of your *web.config* file.

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 currently active (waiting to be processed) synchronization tasks.

Tasks can be filtered using the **Server** drop-down list according to the server to which they are assigned, or **(all servers)** can be selected to show all synchronization tasks in the entire web farm. The **Run tasks** button can be used to manually execute tasks, regardless of how the synchronization interval is currently configured. The listed tasks can be removed all at once using the **Clear task list** button, or individually via the **Delete** (✖) action. Clearing the list is not necessary if the web farm is working correctly, since tasks are removed automatically when successfully processed.



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.

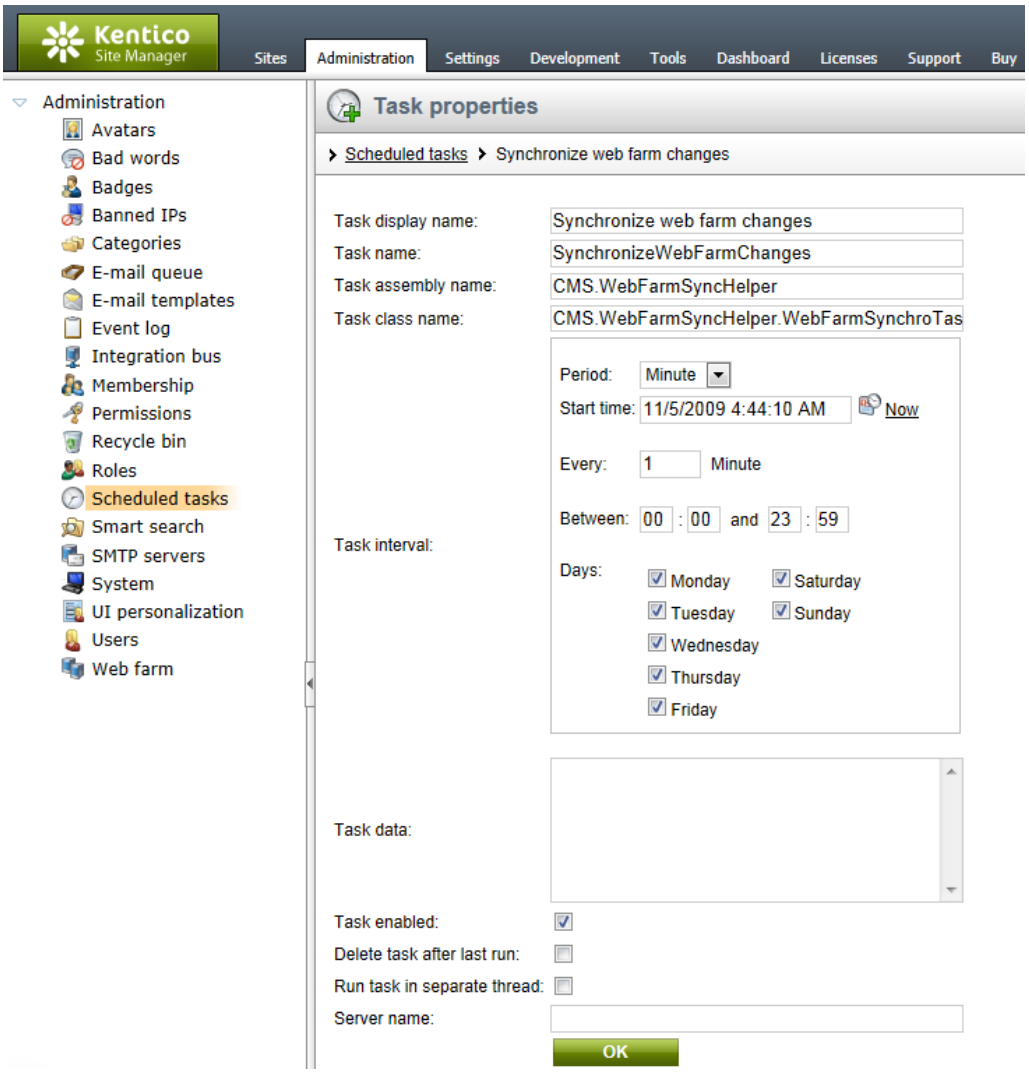
License keys

Please be sure to enter an appropriate license key for the internal URL of the server.

E.g. if the web farm servers are used for domain name *example.com* and you access them internally through URLs like *http://192.168.1.2*, *http://192.168.1.3*, etc., you need to enter separate license keys for *example.com*, *192.168.1.2* and *192.168.1.3* in **Site Manager -> Licenses** dialog (you need to do that on one server only and restart the other instances using **Site Manager -> Administration -> System -> Restart application**).

Configuring the server synchronization interval

By default, web farm servers are updated with changes made on other servers once per request. Synchronizing this frequently may however be impractical for high-traffic websites. The synchronization interval can be changed by setting the value of the **CMSWebFarmUpdateWithinRequest** web.config key to *false* and using the **Synchronize web farm changes** scheduled task instead, which has a configurable execution interval. To do this, go to **Site Manager -> Administration -> Scheduled tasks**, select *(global)* from the **Site** drop-down list at the top of the page, edit (✎) the mentioned task, set the required **Task interval**, check the **Task enabled** property and click **OK**.



The task must also be created for each server in the web farm, so return to the list of scheduled tasks, click the **New task** link, fill in the same properties as those of the **Synchronize web farm changes** task (the **Task name** must be different) and check the **Create tasks for all web farm servers** box below the **Server name** property. Click **OK** to complete the process.

8.51.3 Creating custom web farm synchronization tasks

It is possible to 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 you can 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. Next, 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**.

8.51.4 Web farm synchronization internals and API

8.51.4.1 Overview

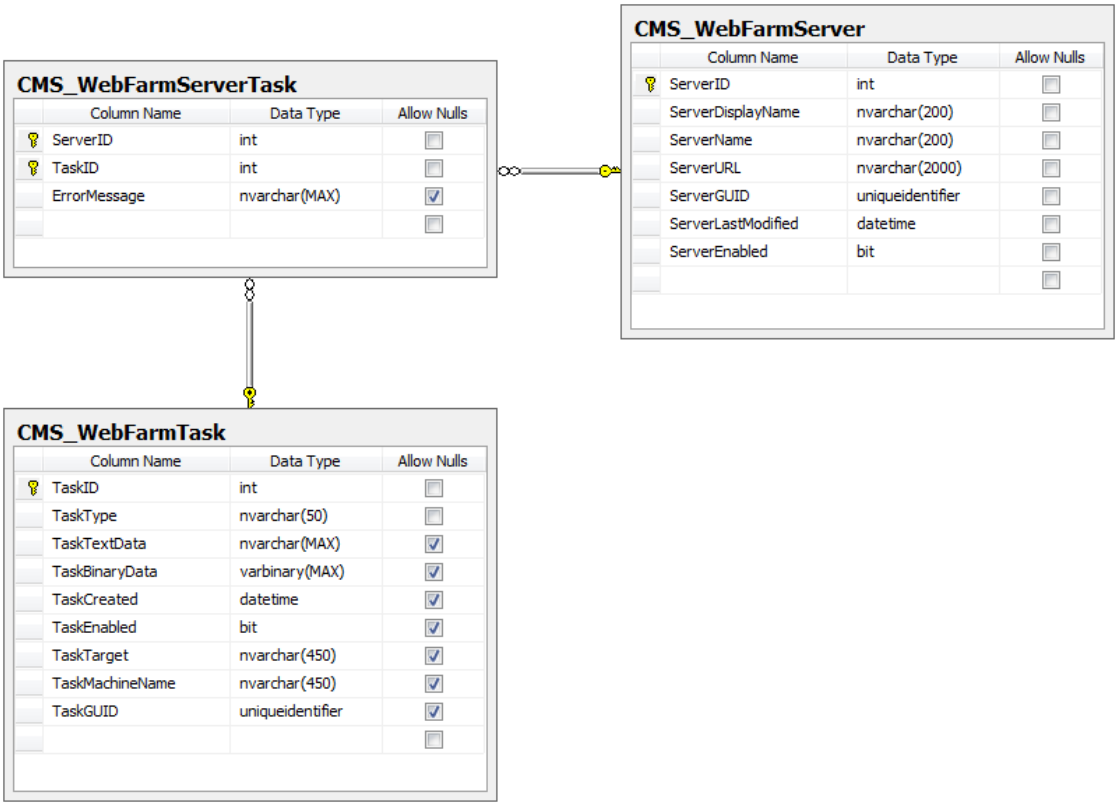
In this chapter, you will learn which database tables and API classes are used in the Web farm synchronization module. You will also see the most common API examples.

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.

8.51.4.2 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.



8.51.4.3 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.

8.51.4.4 API examples

8.51.4.4.1 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;
```

8.51.4.4.2 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);
}
```

8.51.4.4.3 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();
}
```

8.52 Website optimization

8.52.1 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.

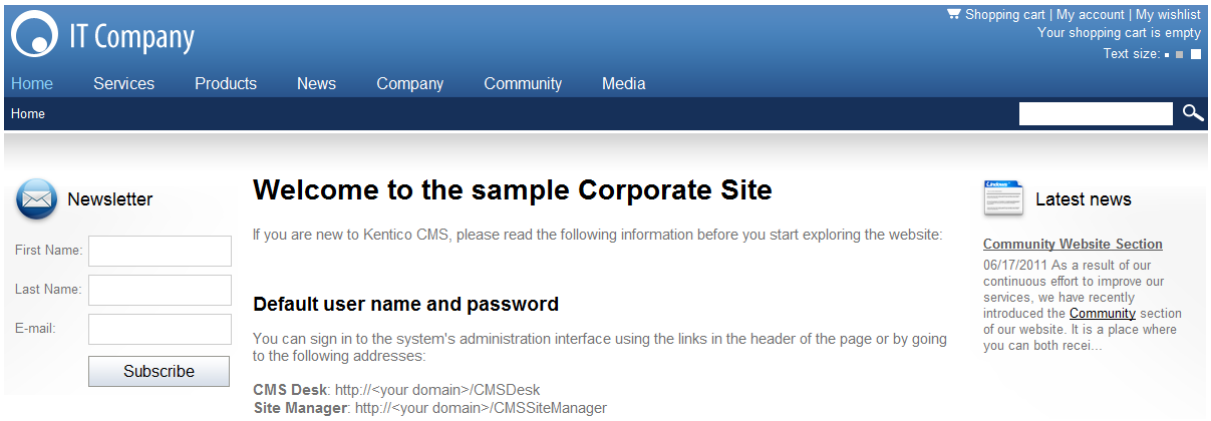
For both types of optimization testing, results are measured by tracking the activity of users after they access the tested page and view one of the different content versions. Actions that are desired from visitors are represented in the system as ***Conversions***. Typical examples of conversions are 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.

8.52.2 A/B testing

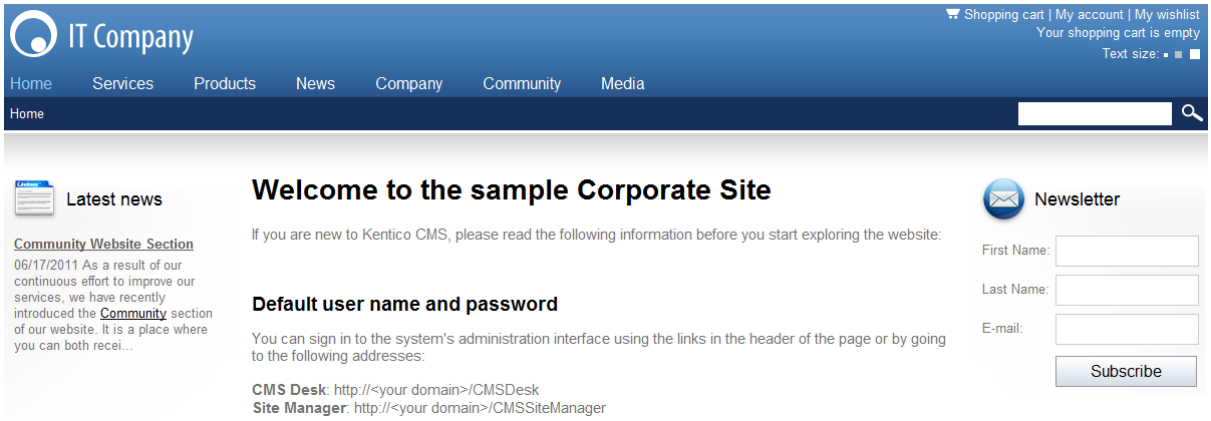
8.52.2.1 A/B testing overview

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 a visitor navigates 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.

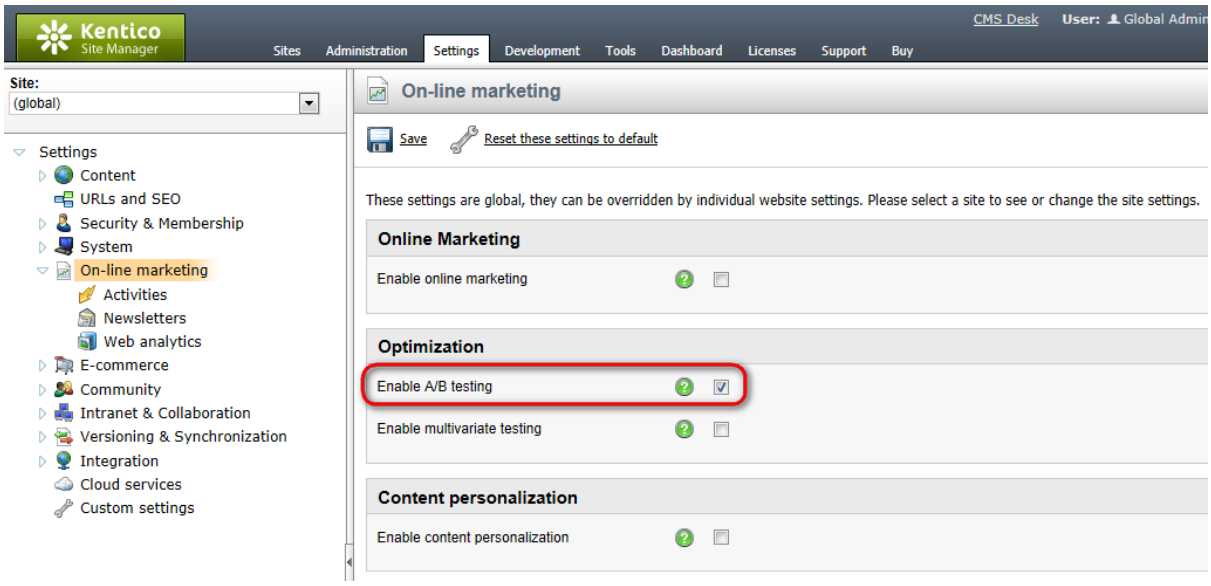
It is possible to run multiple A/B tests concurrently for different pages on the same website. Conversions will be logged for all tests defined for pages visited by a given user, according to the cookies present in the browser.

8.52.2.2 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.

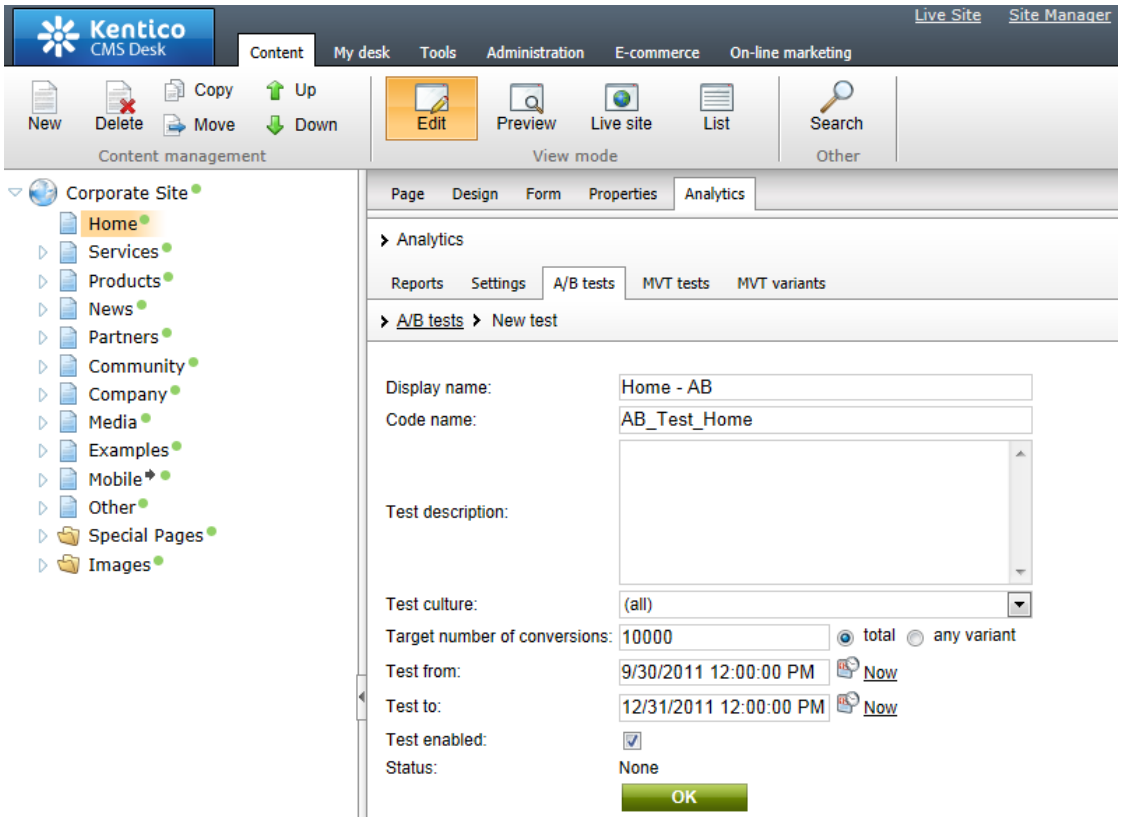
Defining an A/B test for a page

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 the **New test** link 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.

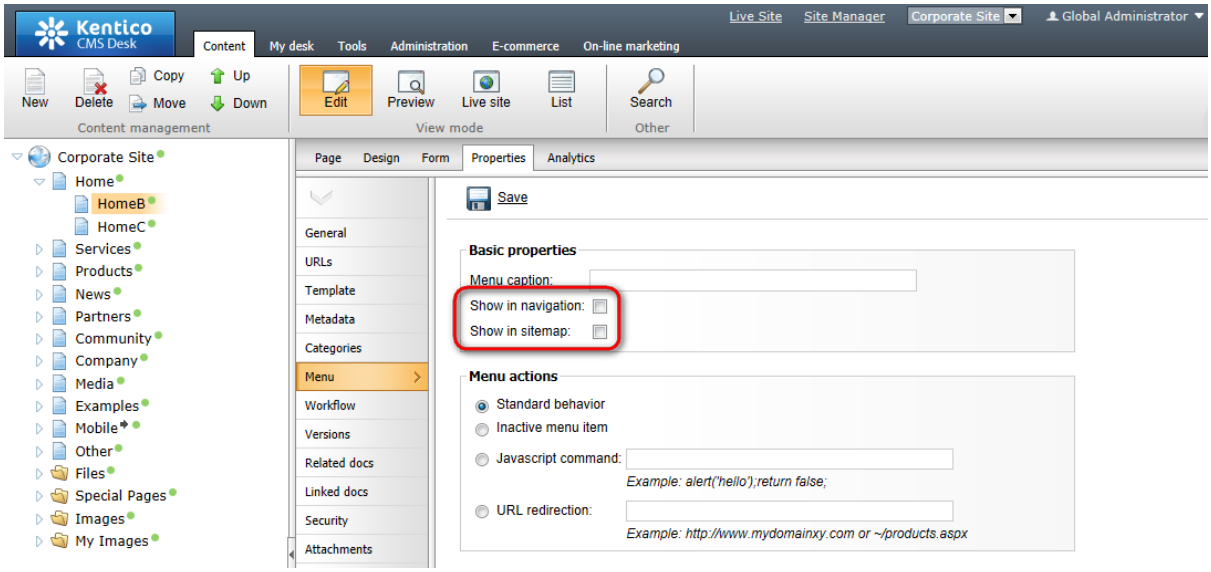
Usually, the page variants of an A/B test will all use the same basic concept, but have one key difference that sets them apart from the original page, such as:

- 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 -> Menu** 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 displayed by the Site map web part and included in the website's Google Sitemap.



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.

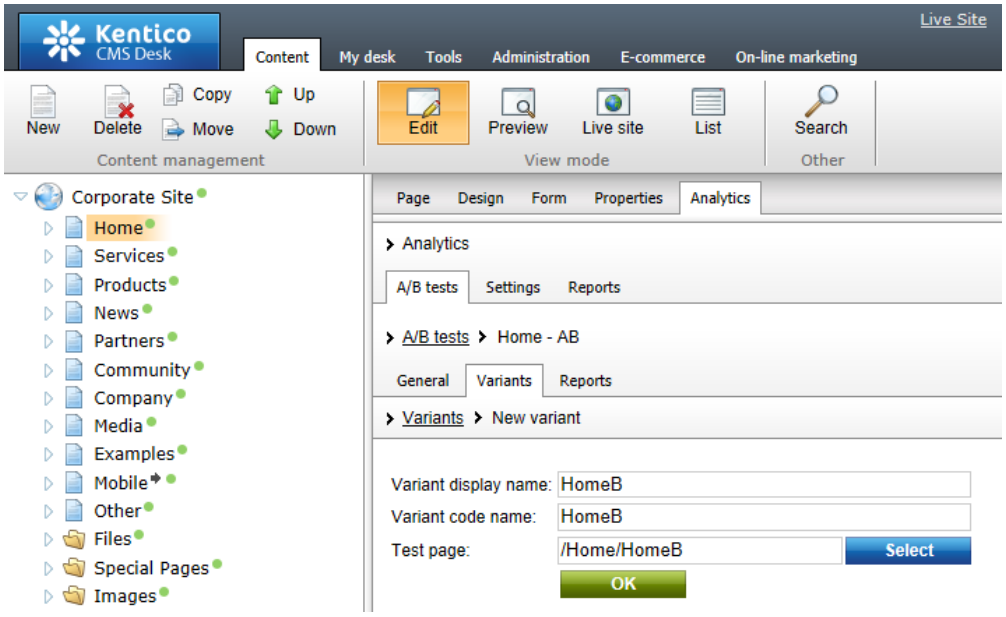
Additionally, the **Exclude from search** flag can be enabled for page variants on the **Properties -> General** tab. This will ensure that the given variant will not be returned in the results of searches performed on the website.

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 the **New variant** link and enter the following properties:

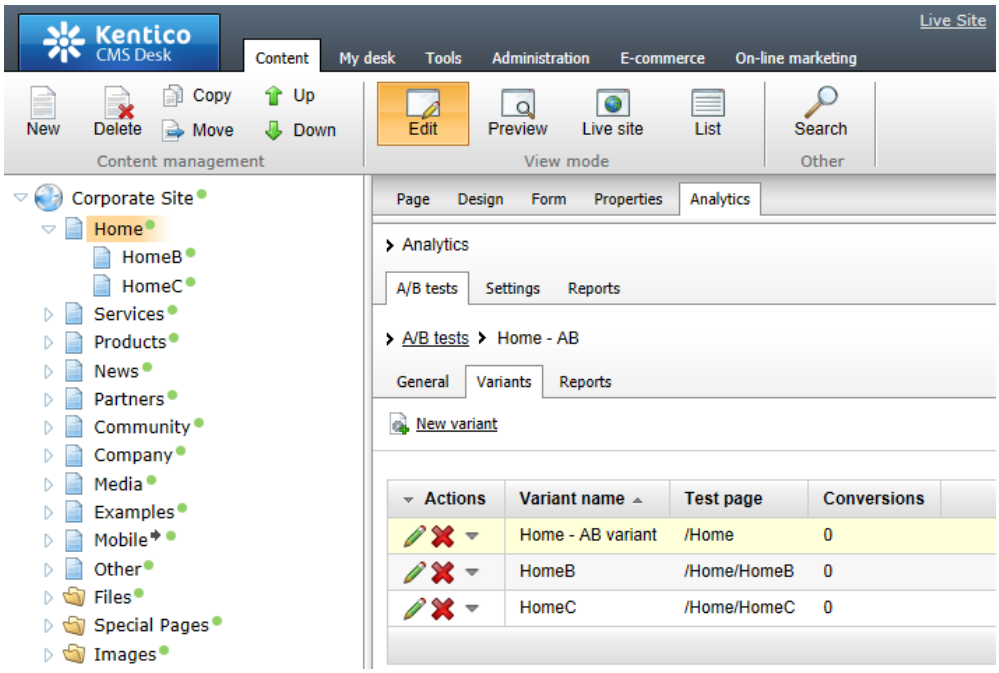
- **Variant display name** - the name of the variant displayed in the administration interface.
- **Variant code name** - sets a code name that serves as an identifier for the page variant. It is also

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.

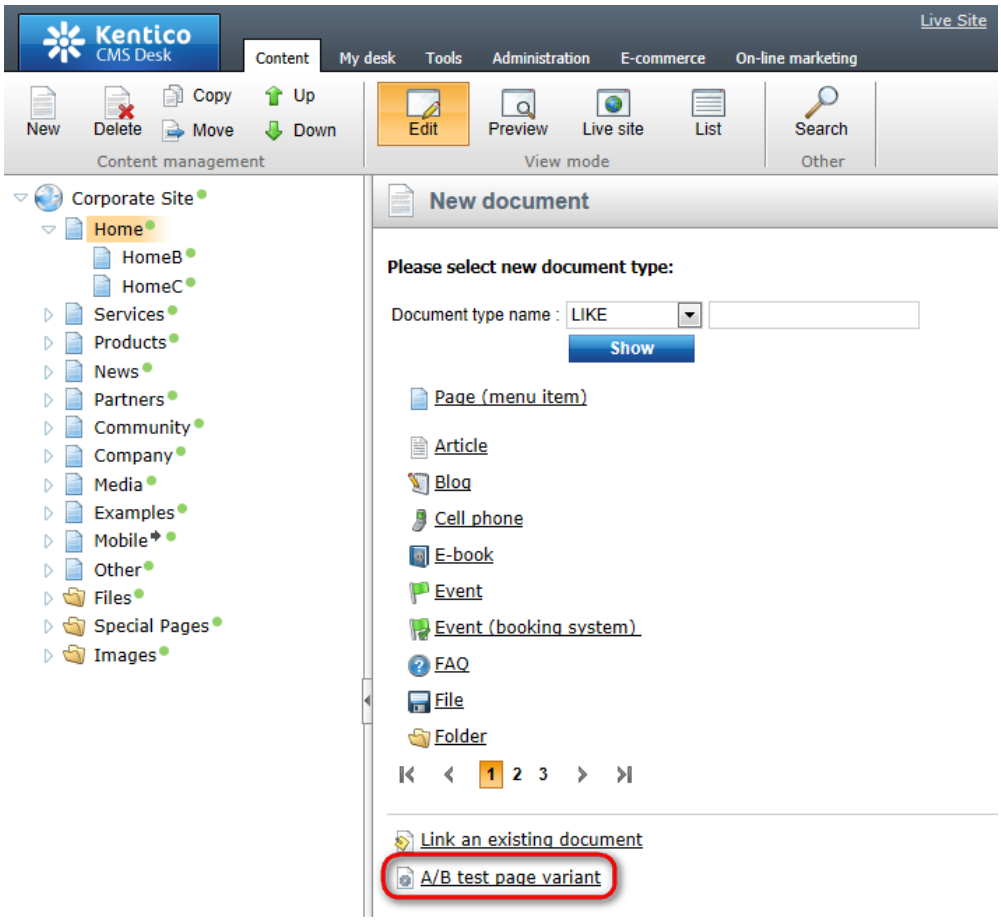


Important!

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.

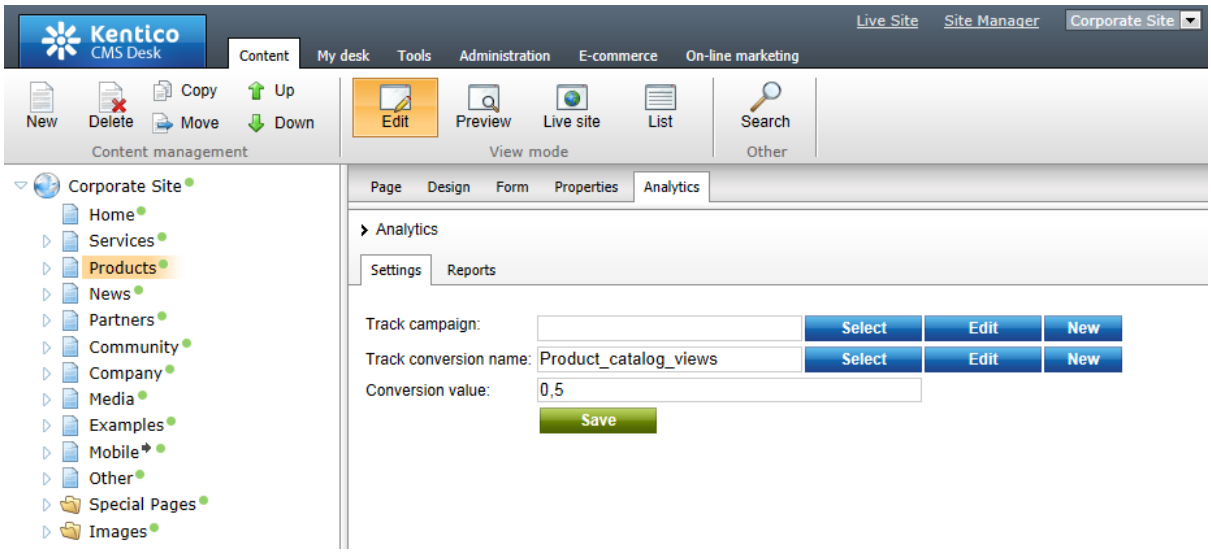
- **Assign to A/B test** - selects the A/B test to which the variant will be added.
- **Show in navigation, Show in site map, Exclude from search** - these options may be used to directly set the corresponding properties of the new document. The default values will hide the page variant from all standard 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.

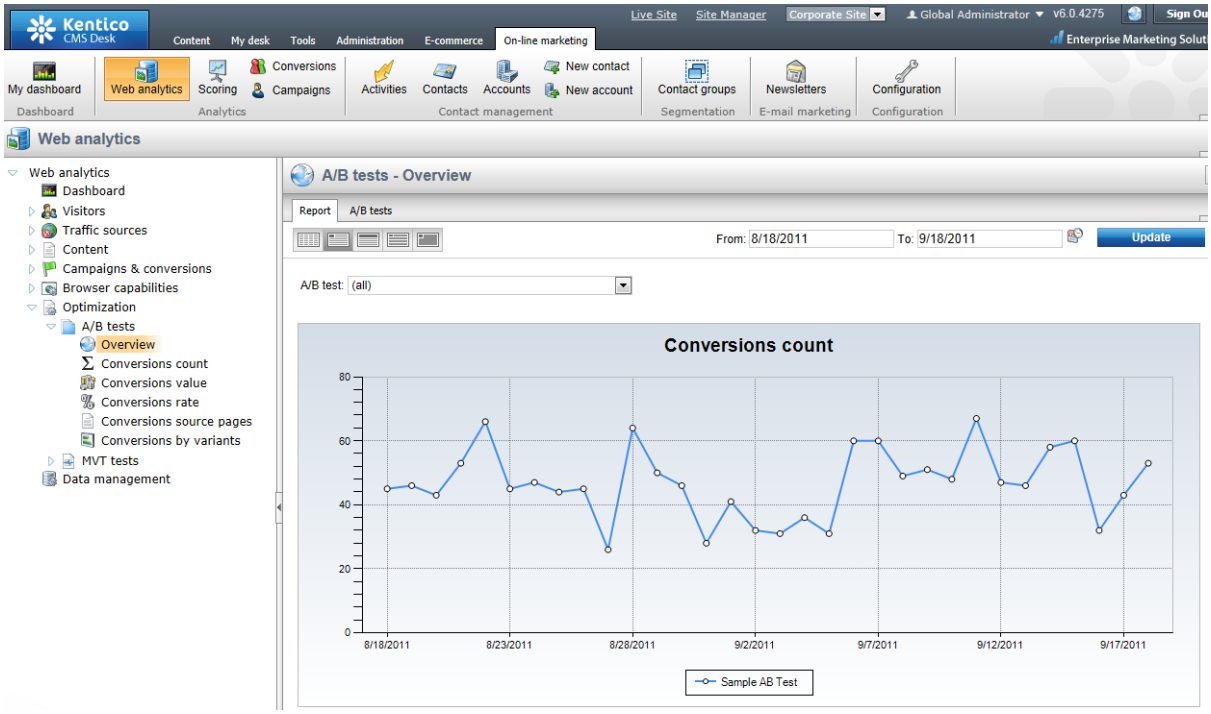
When an action designated as a conversion is performed anywhere on the website, the system checks if the given user has passed through a page with an A/B test (according to the presence of a cookie). If this is the case, the conversion hit will be logged for the page variant assigned to the user.

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.

8.52.2.3 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.

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 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.

8.52.3 Multivariate testing

8.52.3.1 Multivariate testing overview

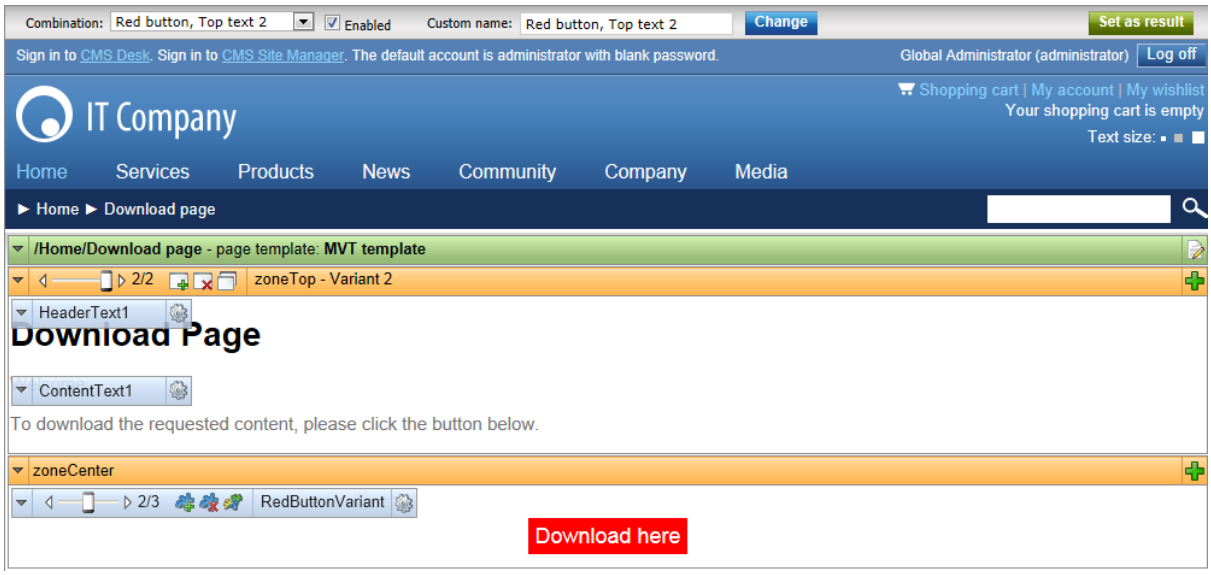
Multivariate testing (MVT) provides a way to optimize a website's pages based on the behaviour of its visitors. It allows you to define any number of elements on a page as variables and create different versions for each one. Once the test is started, users who view the page will see one of the possible 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.

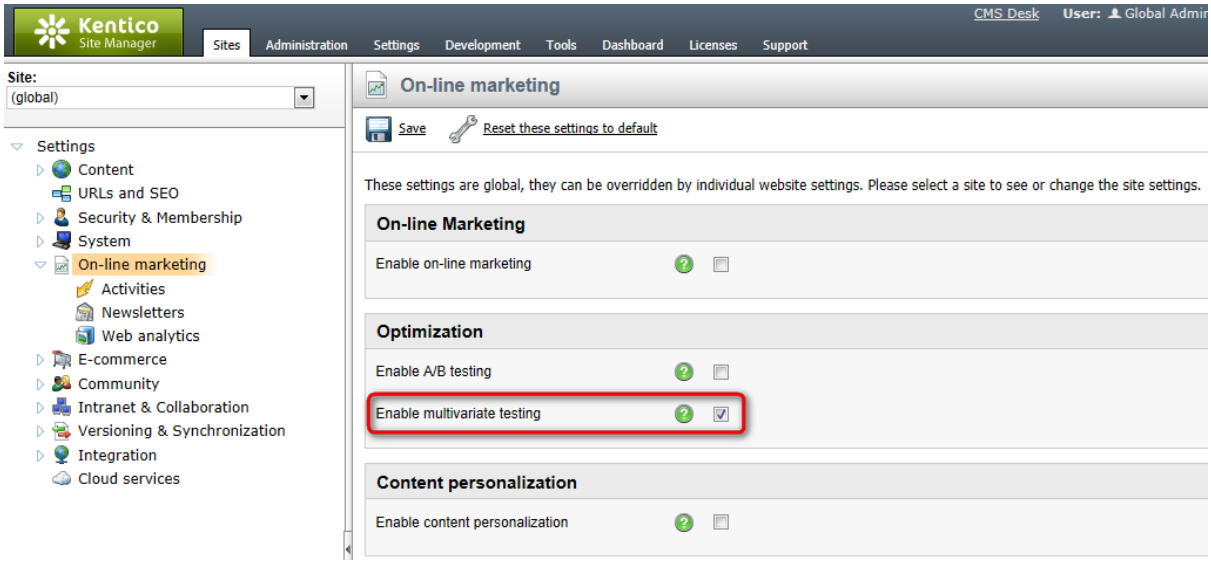
It is possible to run multiple MVT tests concurrently for different pages on the same website. Conversions will be logged for all tests defined on the pages visited by a given user, according to the cookies present in the browser.

8.52.3.2 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

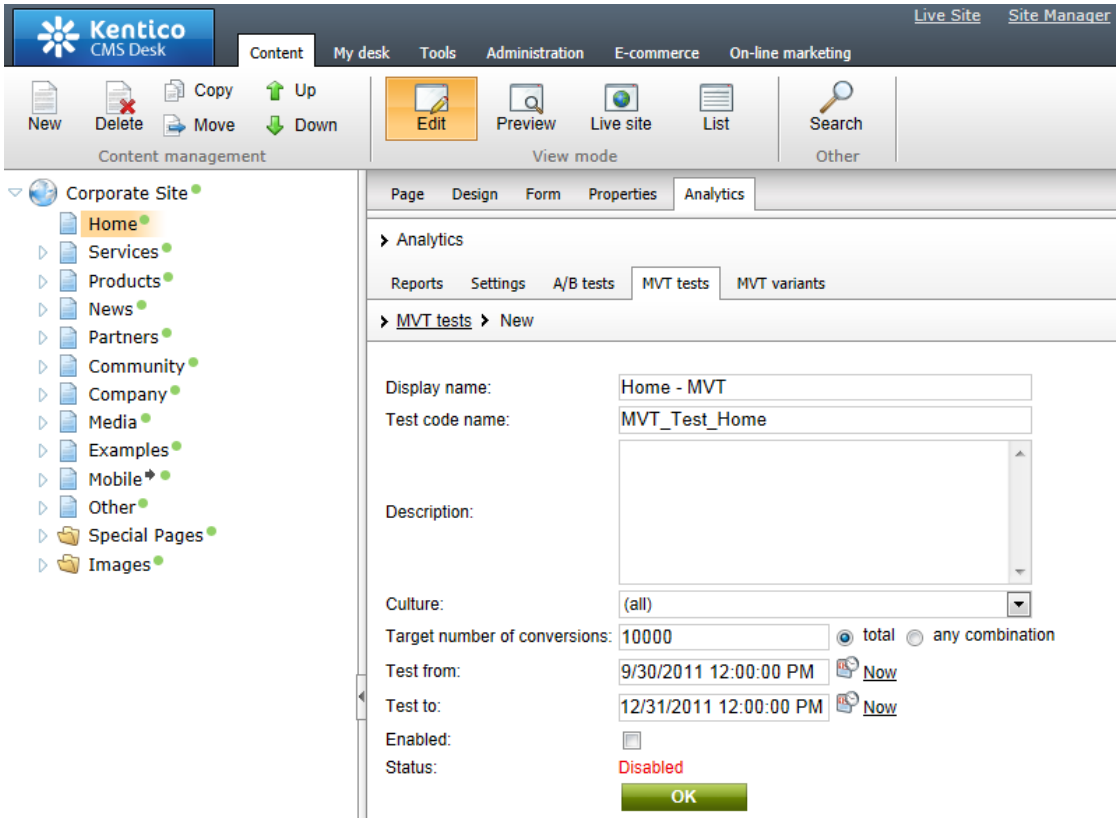
To start optimizing a page on your website via multivariate testing, create and configure an *MVT test*

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 the **New MVT test** link 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.

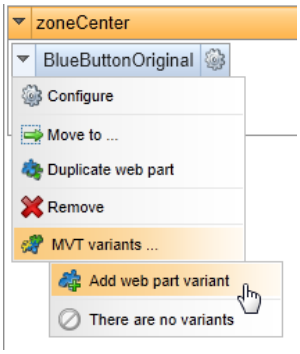
Creating testing variants on the page

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.

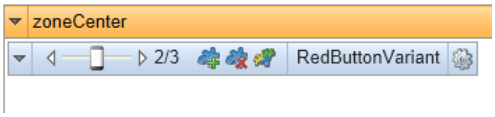


Then, fill in the following properties in the displayed dialog:

- **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.

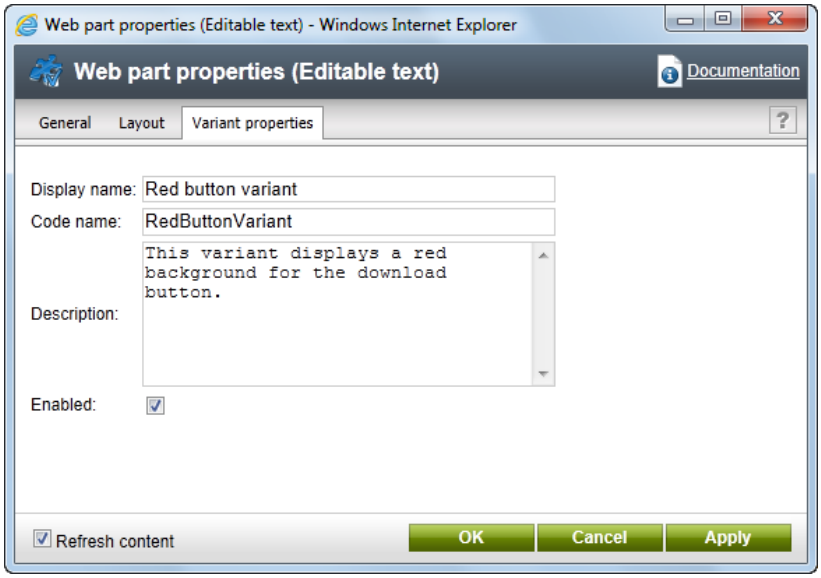
Each object for which you create testing variants will have a slider available in its header as shown below:



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.

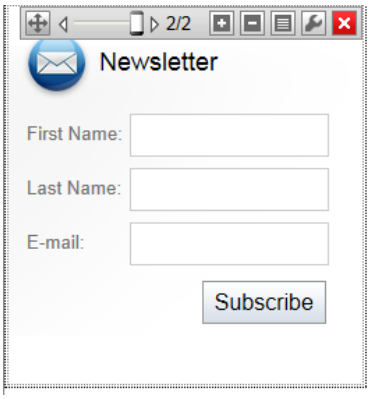
If you wish to view the content of variants while cycling through the slider on the **Design** tab, make sure that the **Display web part content** checkbox on the right side of the **Edit** mode header is checked. To make orientation easier, it is recommended to assign an appropriate **title** property 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.

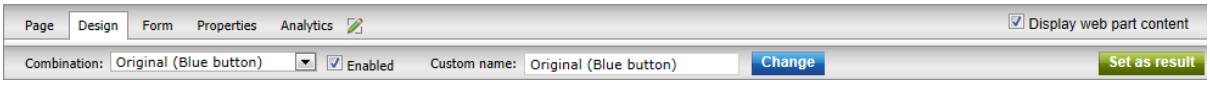
Multivariate testing and Content personalization

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.

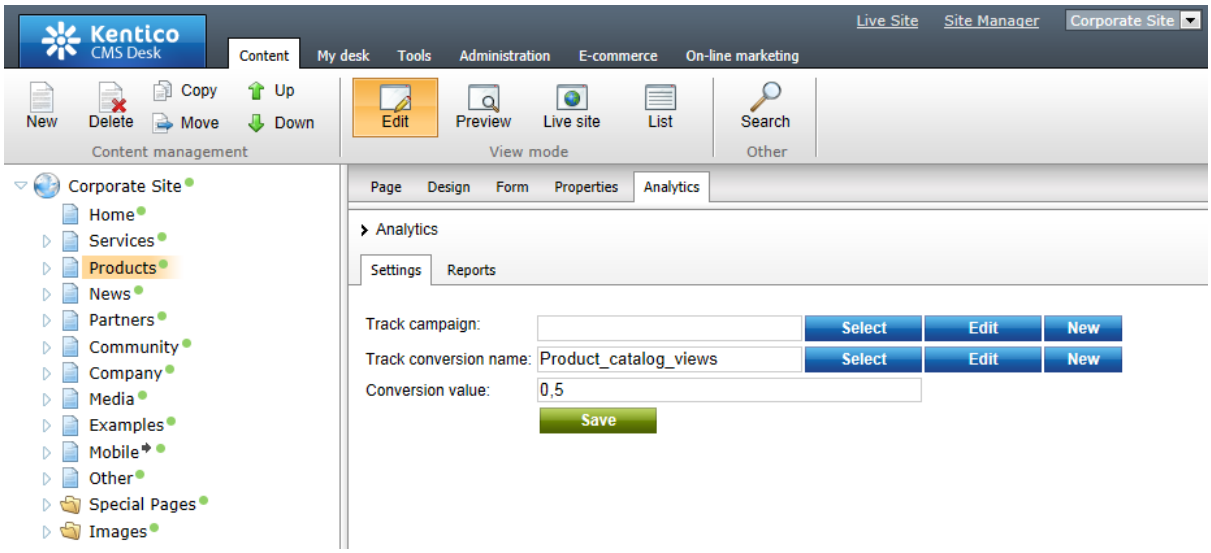
Once the page's variants and combinations are all configured as needed, you can enable your MVT test on the **Analytics -> MVT tests** tab of the document. Making further modifications on the page while an MVT test is running is not recommended, since this may affect the accuracy and relevance of the 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.



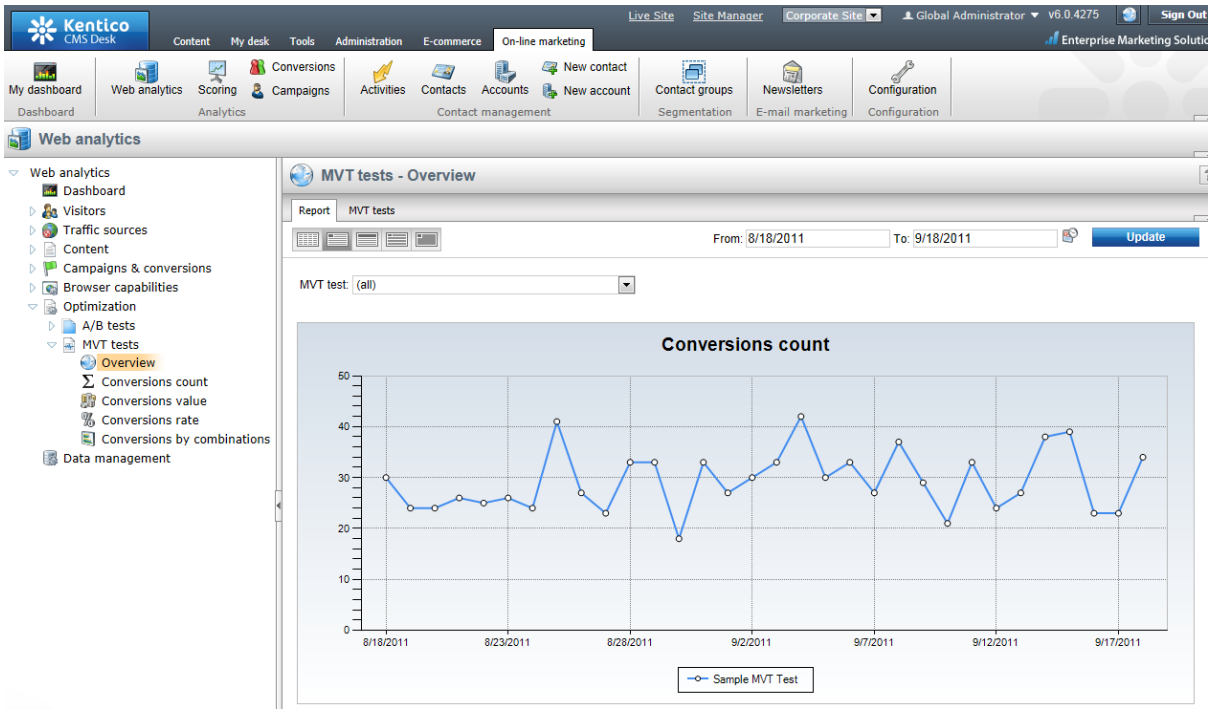
Additionally, many web parts and widgets that allow users to perform important actions (e.g. registration) also have the **Track conversion name** property, which can be used to specify a conversion in the same way. Further details about conversion management may be found in the Modules -> Web analytics -> 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.

8.52.3.3 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.

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 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.

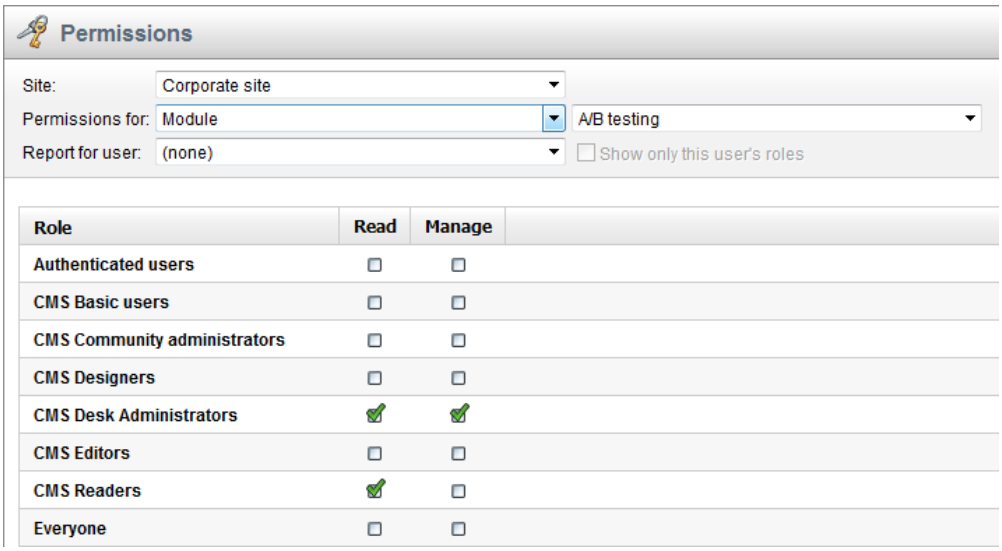
After you select an action from the menu, your browser's standard file download dialog will pop up, letting you open or save the file with the exported data just like when downloading any other type of file. For more details on the data export feature, please refer to the Modules -> UI data export chapter.

8.52.4 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.

8.53 Modules internals and API

8.53.1 Overview

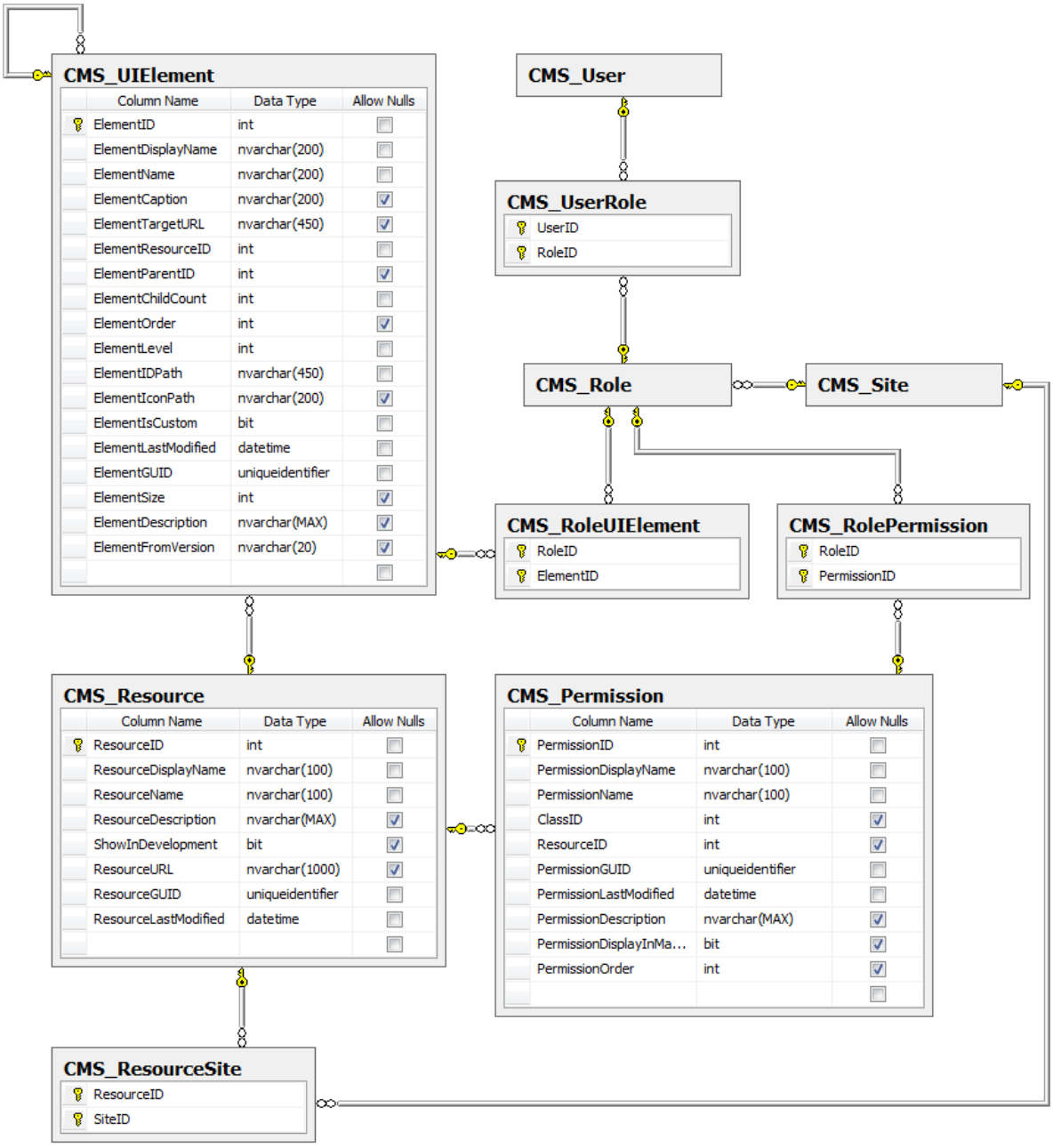
In this chapter, you will learn which database tables and API classes are used in Modules. You will also see the most common API examples.

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.

8.53.2 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.



8.53.3 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.

8.53.4 API examples

8.53.4.1 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;
```

8.53.4.2 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);
}
```

8.53.4.3 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);
}
```

8.53.4.4 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;
}
```

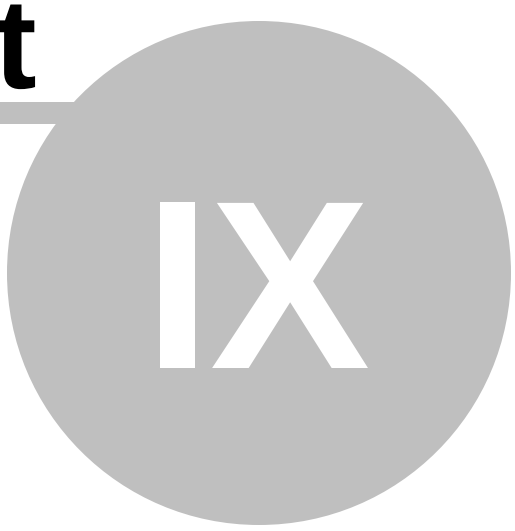
The following example removes a UI element from role.

```
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 it can 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 it can't 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**

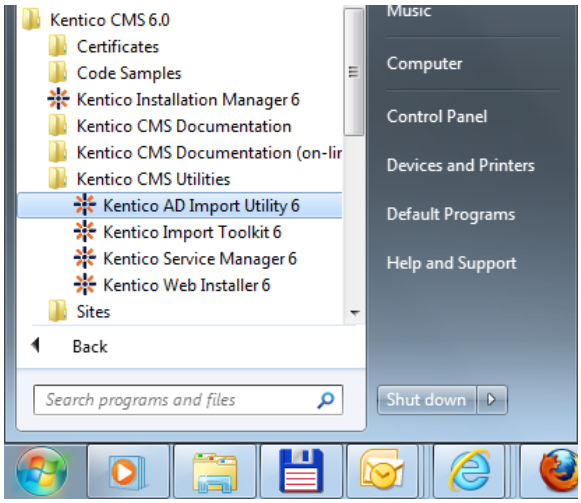
Launching the utility

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

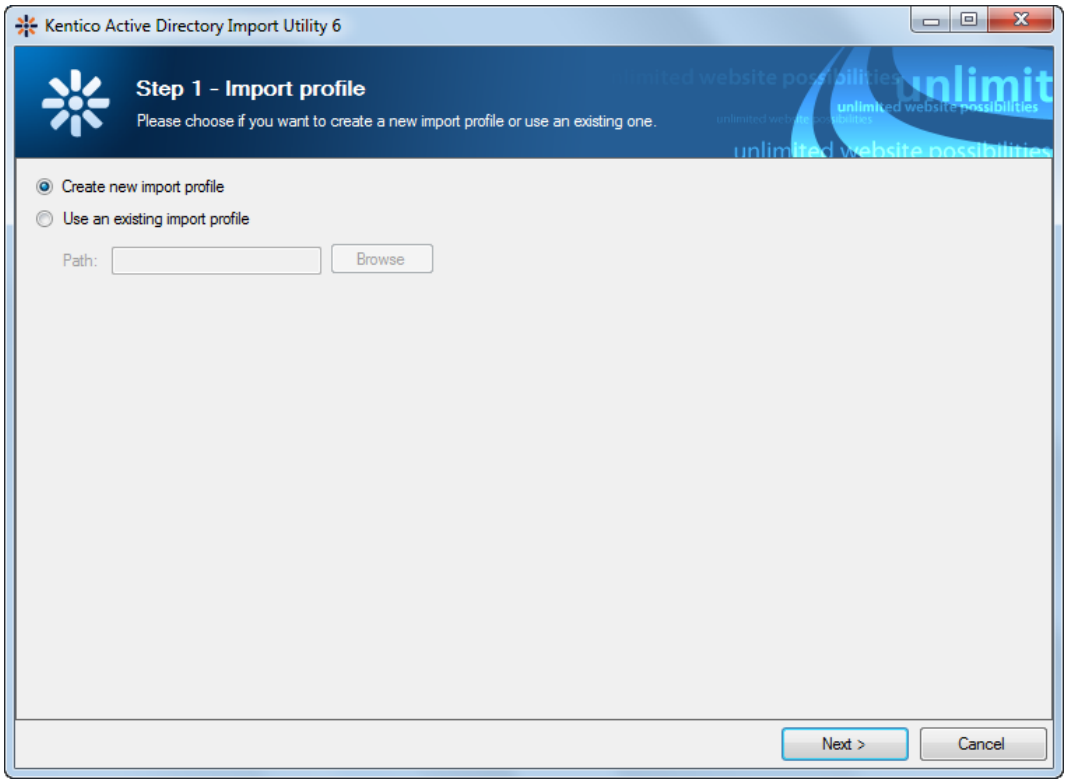
The AD Import Utility wizard can be executed either from **Start menu -> All programs -> Kentico CMS <version number> -> Kentico CMS Utilities**, or by launching the **ADImport.exe** file located in **<Kentico CMS installation folder>\Bin** (typically *c:\Program Files (x86)\KenticoCMS\<version number>\Bin*).



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.



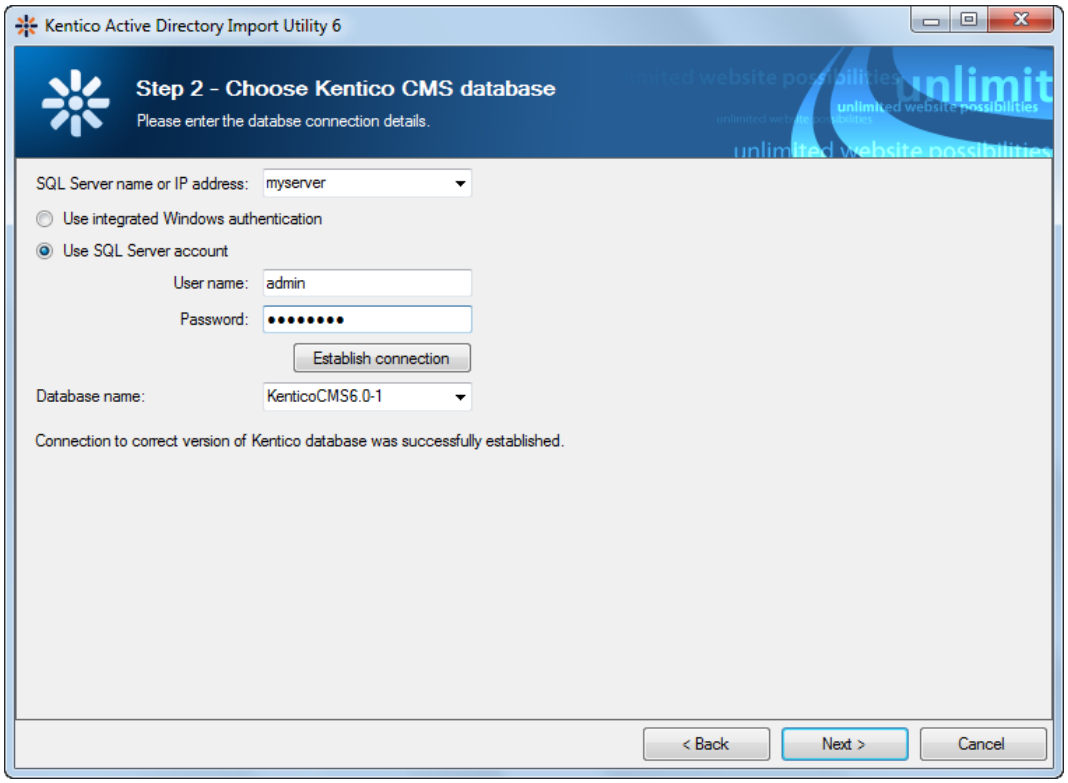
Step 2 – Kentico CMS DB Setup

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.

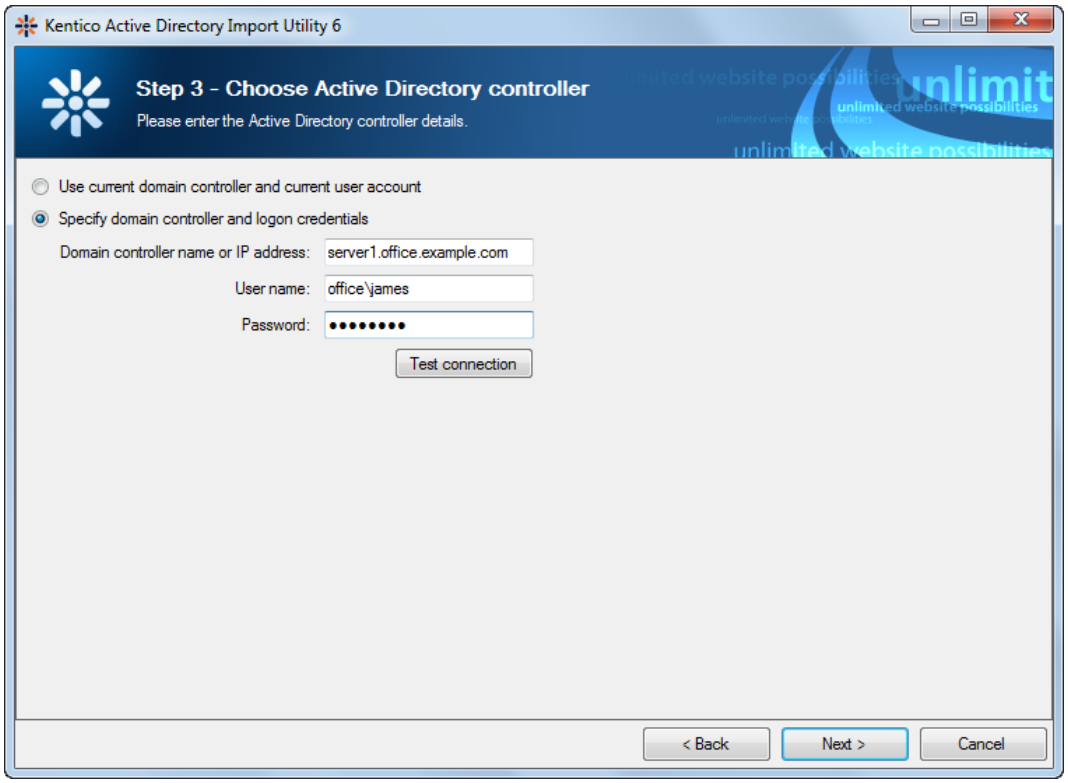


Step 3 – Active directory connection

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:

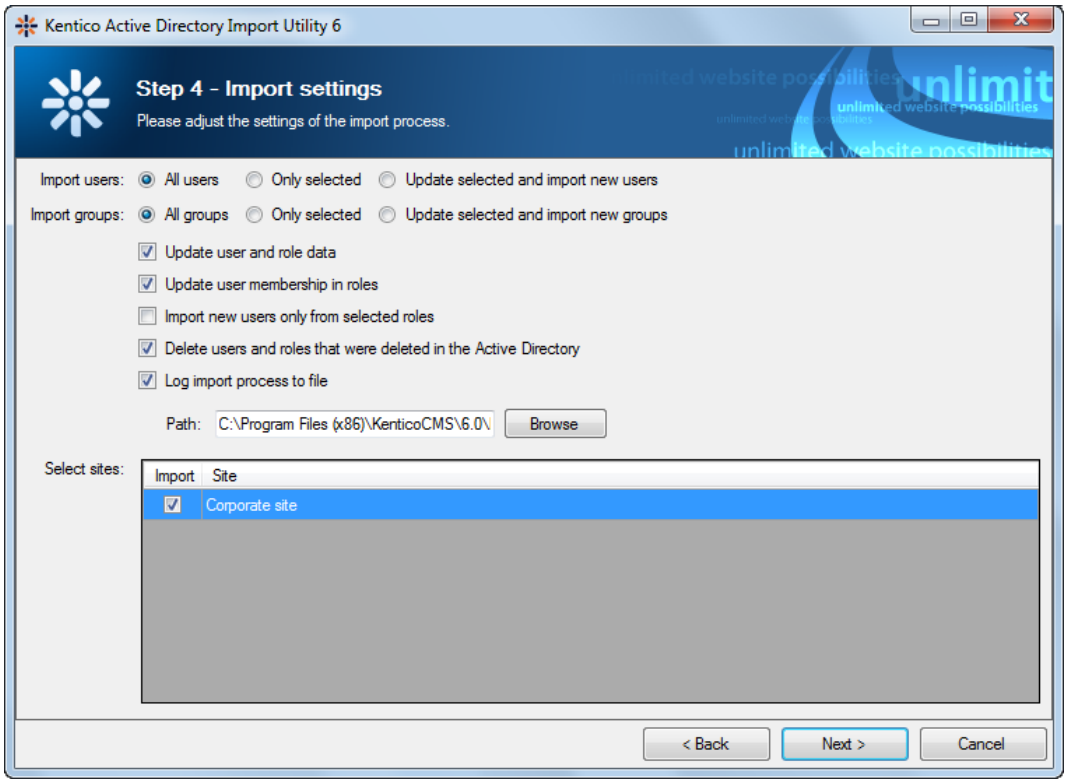
- **Import users/groups** - determines which users or groups (roles) will be pre-selected in Step 6, you have the following options:
  - **All** - all users or groups will be pre-selected.
  - **Only selected** - when using an existing import profile, selection stored in the profile will be used. Otherwise, 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.

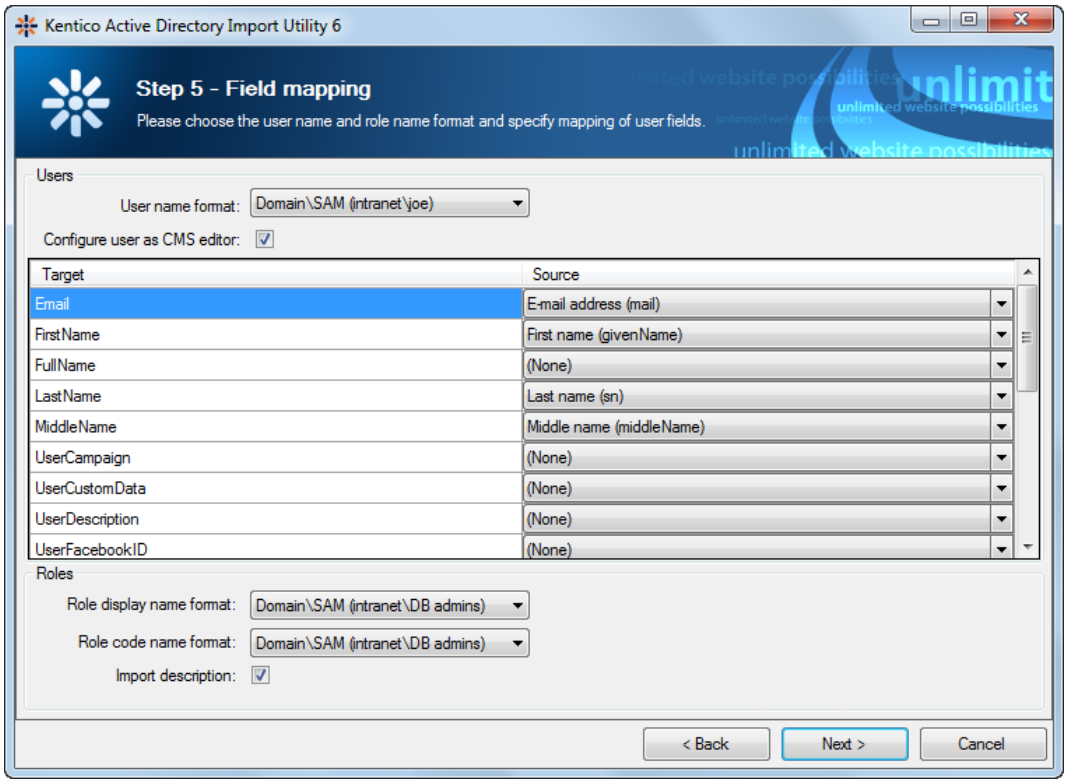


Step 5 – Import properties

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** - if enabled, all imported users will have the Is editor option enabled (this option is located in *Administration -> Users -> user edit -> General*).
- **Target/Source** - you can choose which properties from AD (Source) will be mapped to particular properties of the *CMS_User* role.

- **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

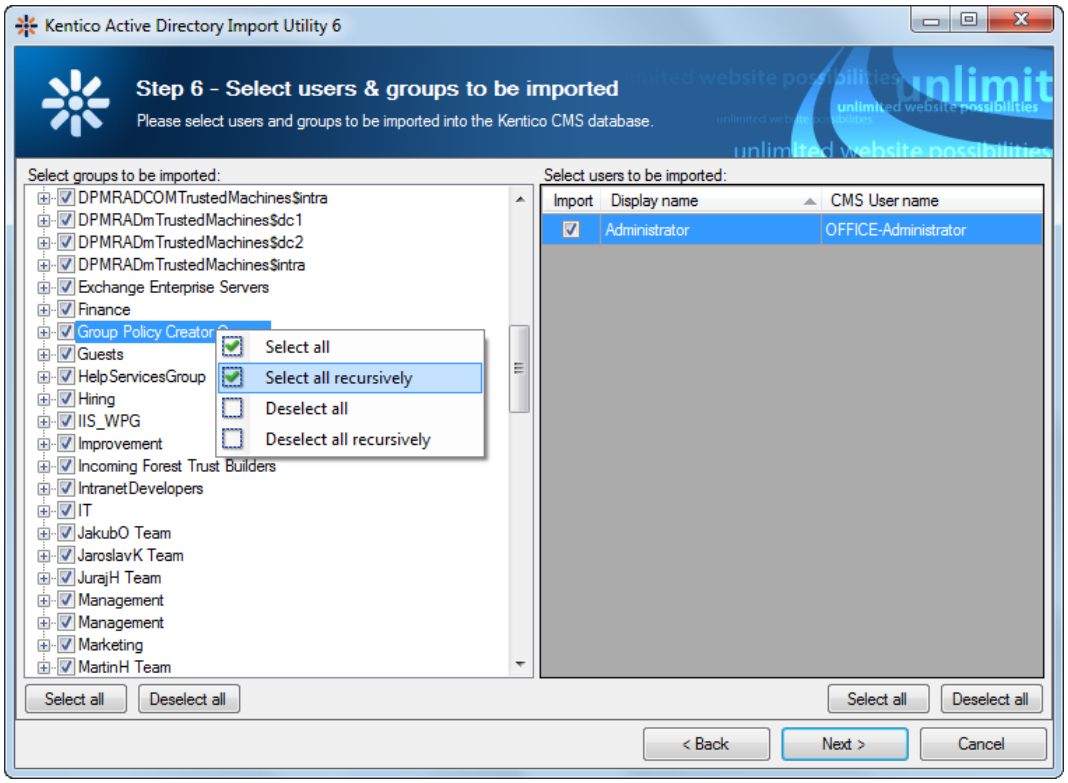
In the sixth step, you can select which roles and users will be imported. It will be possible to adjust the settings made here in the following two steps.

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:

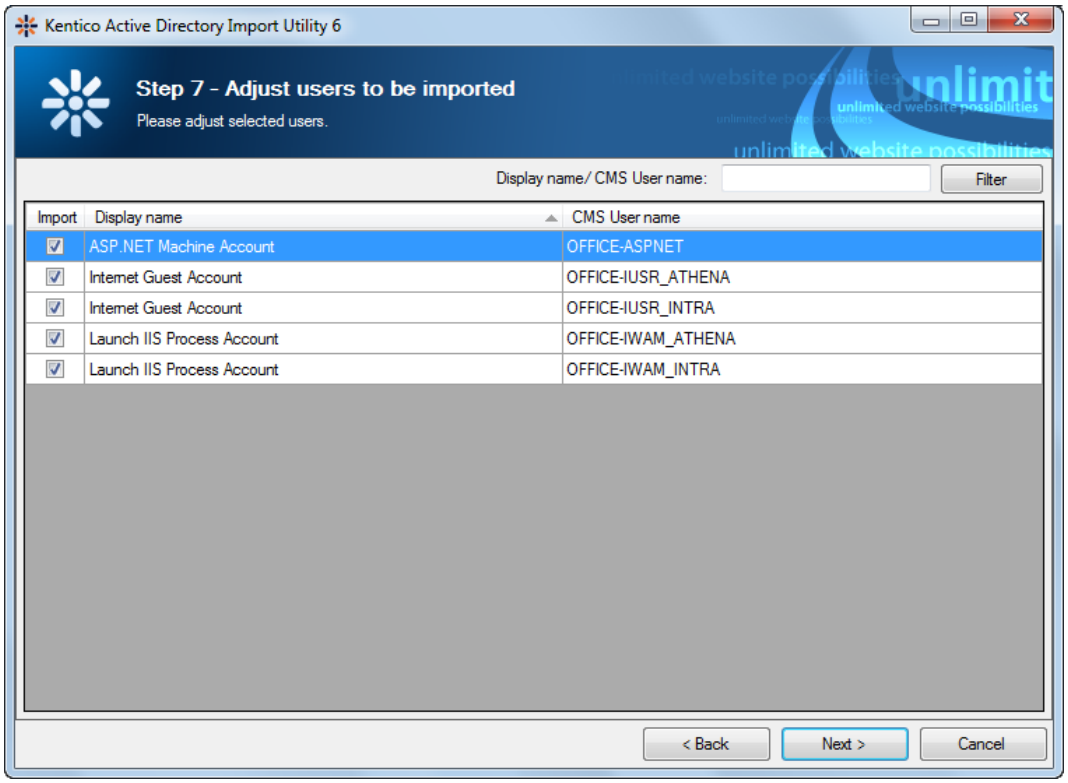
- **Select all** - selects all child groups directly under the selected group.
- **Select all recursively** - selects all child groups under the selected group until the last level.
- **Deselect all** - selects all groups directly under the selected group.
- **Deselect all recursively** - selects all 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.



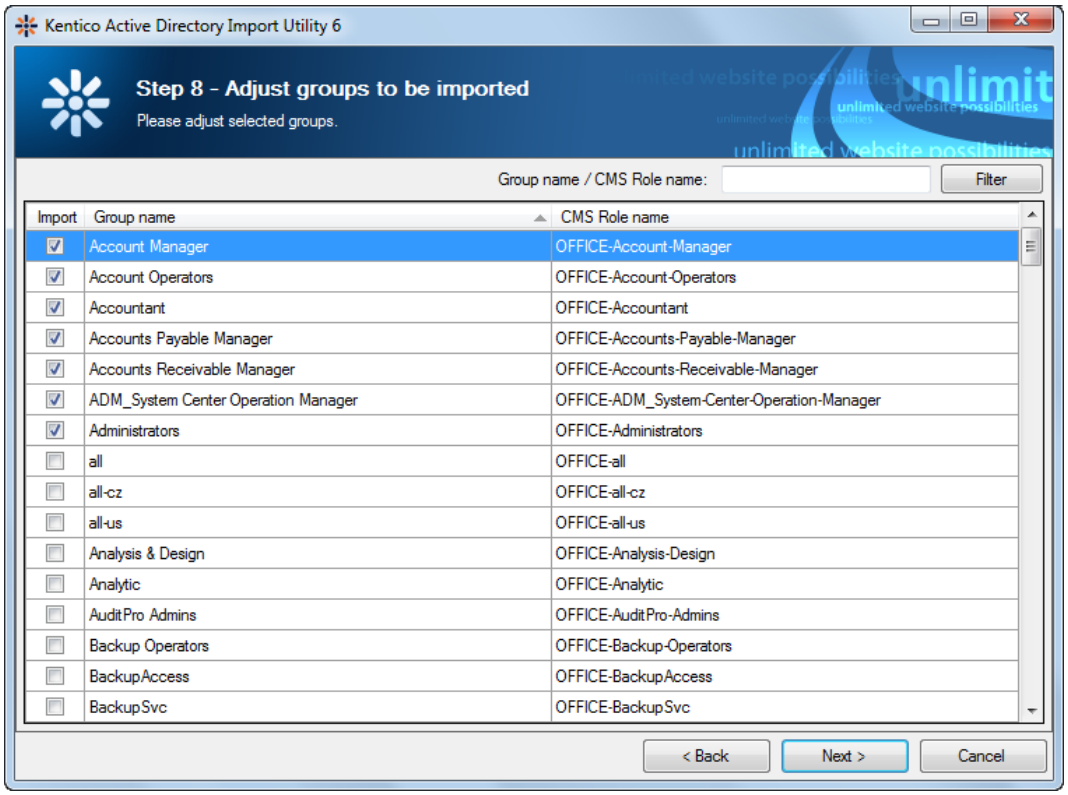
Step 7 – Adjust users to be imported

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.



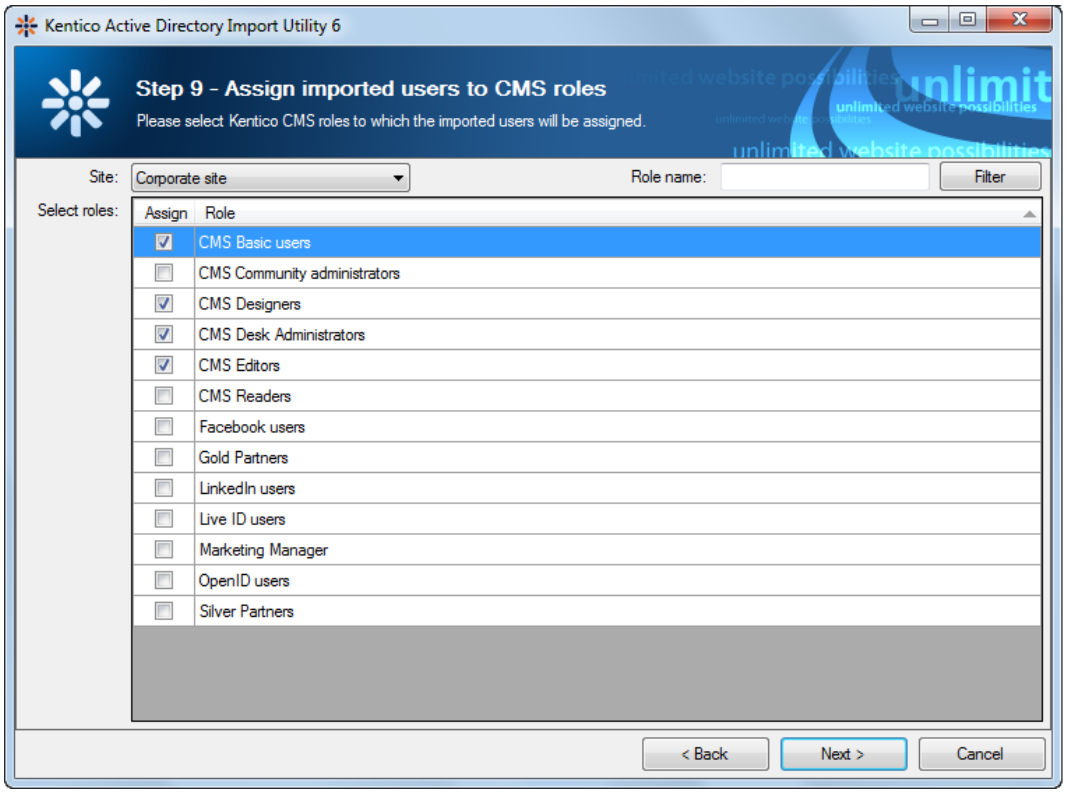
Step 8 – Adjust groups to be imported

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.



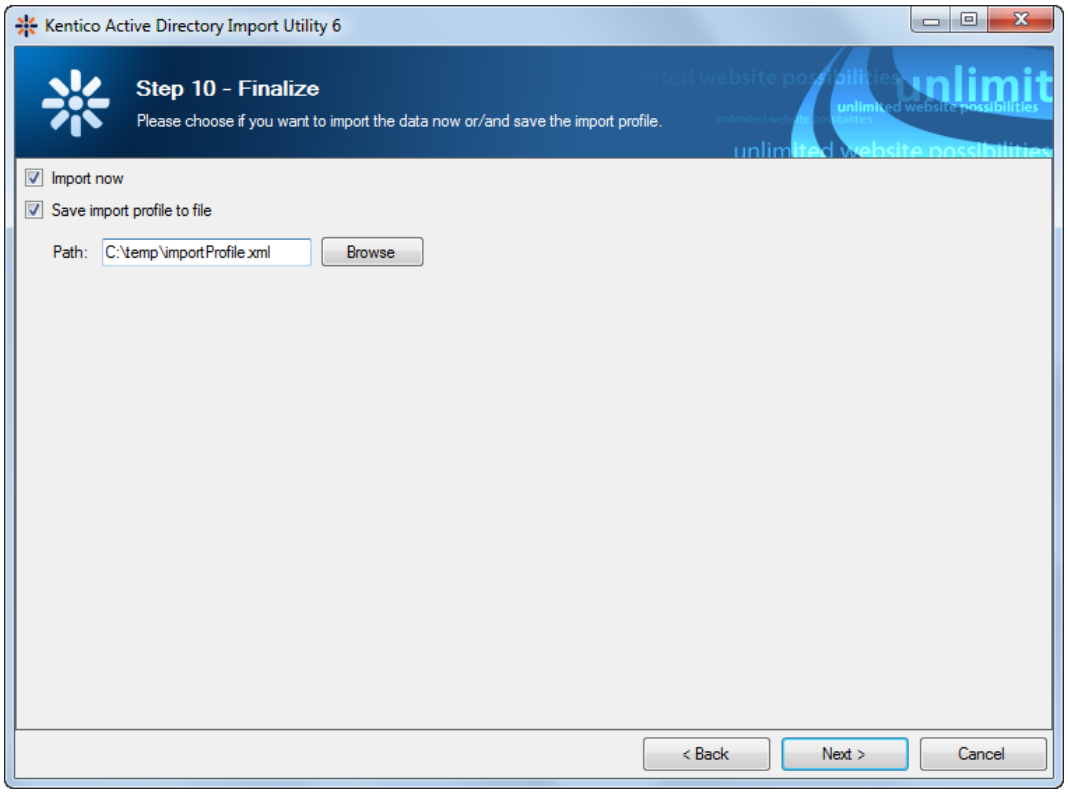
Step 9 – Assign to CMS roles

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.



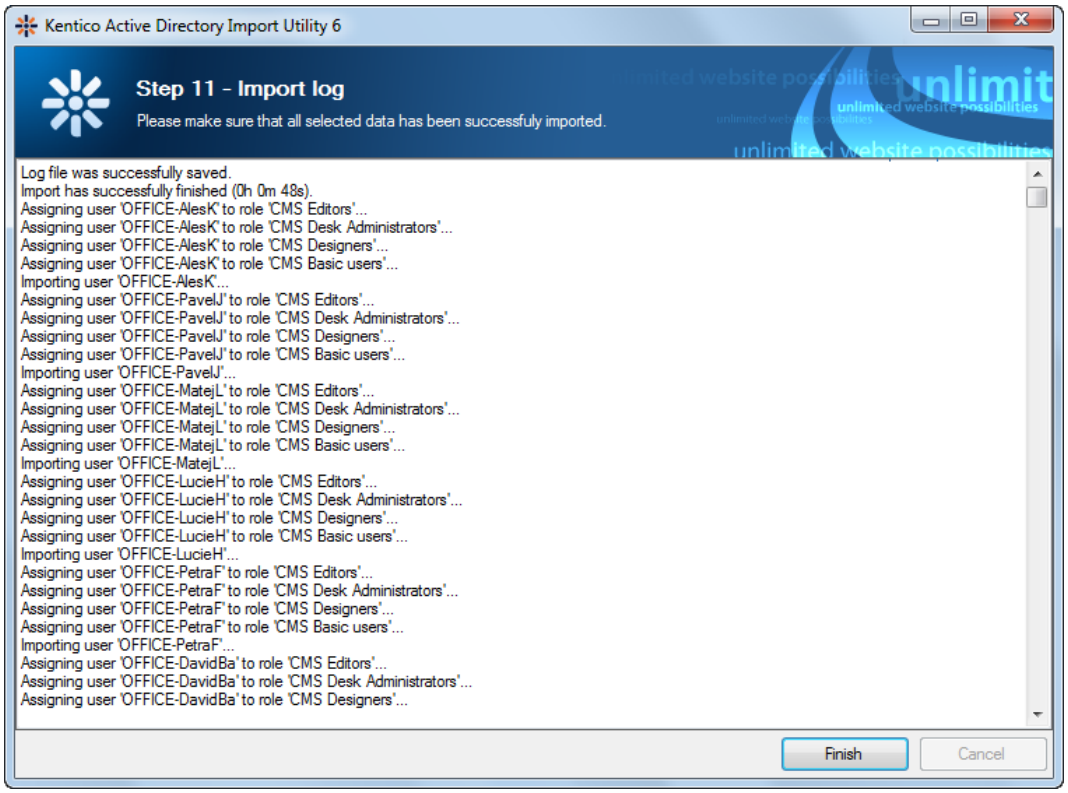
Step 10 - Finalize

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 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, both an absolute and 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.

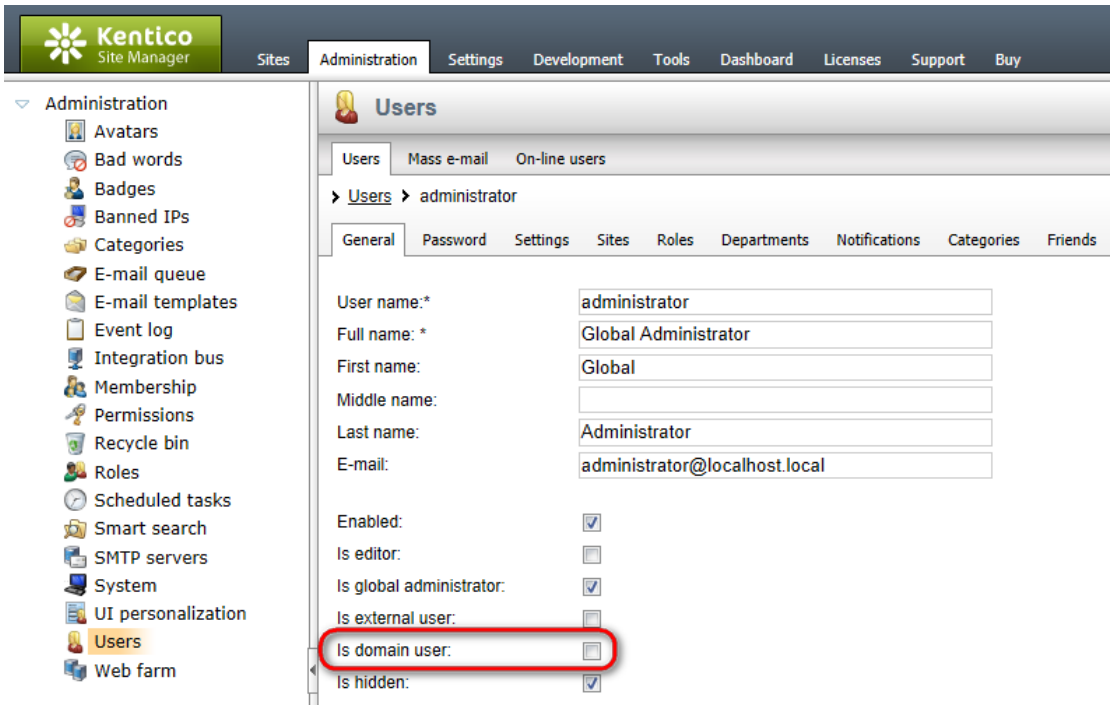
9.1.4 How to recognize imported users and roles

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**



9.2 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)**

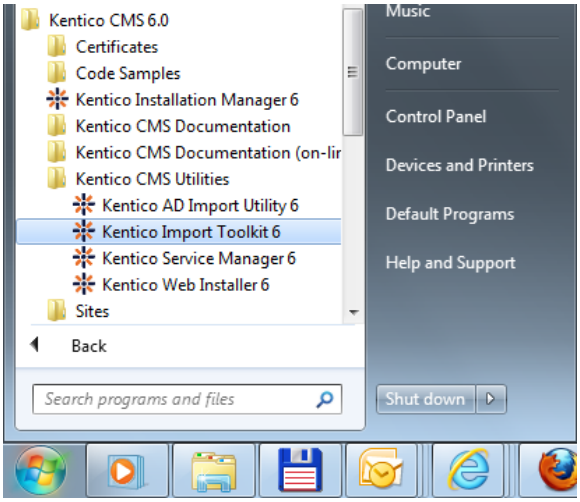
- **Custom table items**
- **On-line form items**
- **Resource strings**

To learn how to use the tool, choose please proceed to the Initial steps topic and follow the step-by-step guide through to the following topics.

9.2.2 Initial steps

This topic describes the initial steps of the Kentico Import Toolkit wizard.

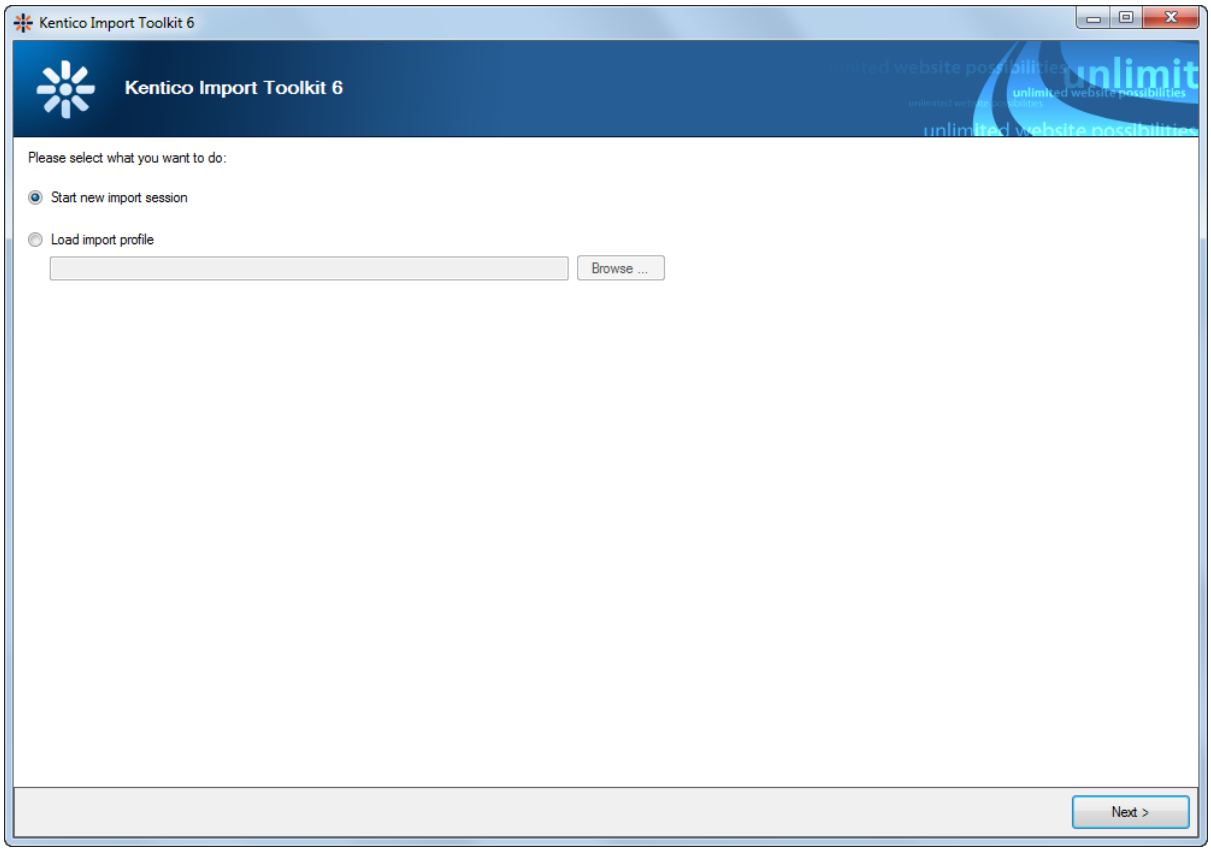
1. Kentico Import Toolkit can be launched from Windows Start menu. Its menu item is located under **Kentico CMS Utilities** in the Kentico CMS program group.



2. On the initial screen of the wizard, you can choose from the following options:

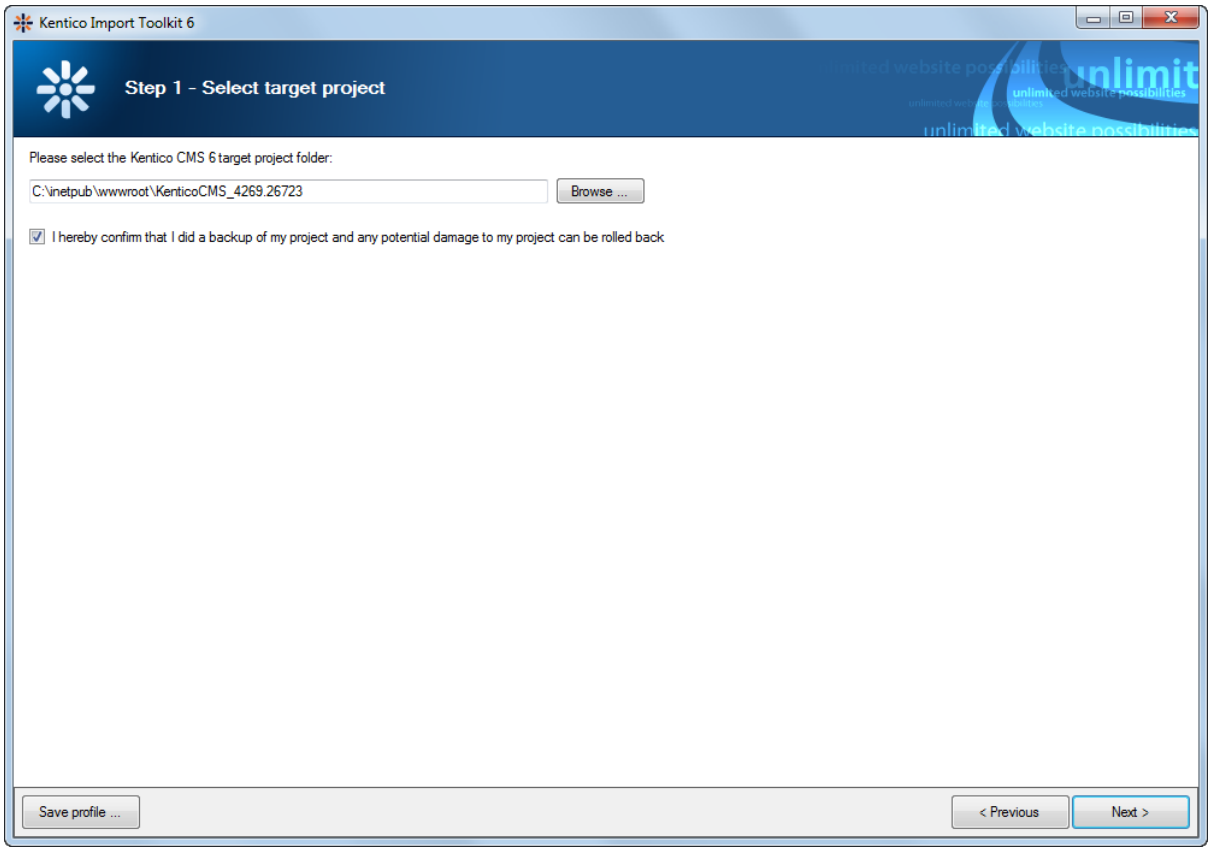
- **Start new import session** - starts a new import session where you can configure all options of the import.
- **Load import profile** - in each of the following steps, you can click the **Save profile** button to save configuration performed in all previous steps and in the current one into an *.iprofile* file. By selecting the **Load import profile** option, you can load such a file and have configuration contained in it pre-filled in the following steps instead of configuring all the options manually.

Click **Next** to proceed when selected.



3. In **Step 1**, you need to choose the root folder of the project into which you want to import your data. You can do it either by entering the path manually, or by clicking **Browse** and searching for the folder within your file system structure. In order to proceed, you also need to tick the check-box below the field. By doing this, you confirm that you performed a backup of your project so that you can roll back any potential damage caused by the import.

When you have the folder selected and confirmed that you backed up your project, click **Next** to proceed to the following step.



4. In **Step 2**, you need to select the data type that will be used to store the imported data in Kentico CMS. This can be selected from 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 as** drop-down list.
- **On-line form items** - imports the data as Form records of the on-line form selected in the **Import as** drop-down list.
- **Objects** - imports the data as Kentico CMS objects of a type selected in the **Import as** drop-down list. By default, only the most typically used objects are offered, while you can extend the selection by enabling the **Show all available objects** check-box.
- **Object attachments (metafiles)** - imports the data as metafiles of Kentico CMS objects of a type selected in the **Import as** drop-down list. By default, only the most typically used objects are offered, while you can extend the selection by enabling the **Show all available objects** check-box.
- **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, while 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.

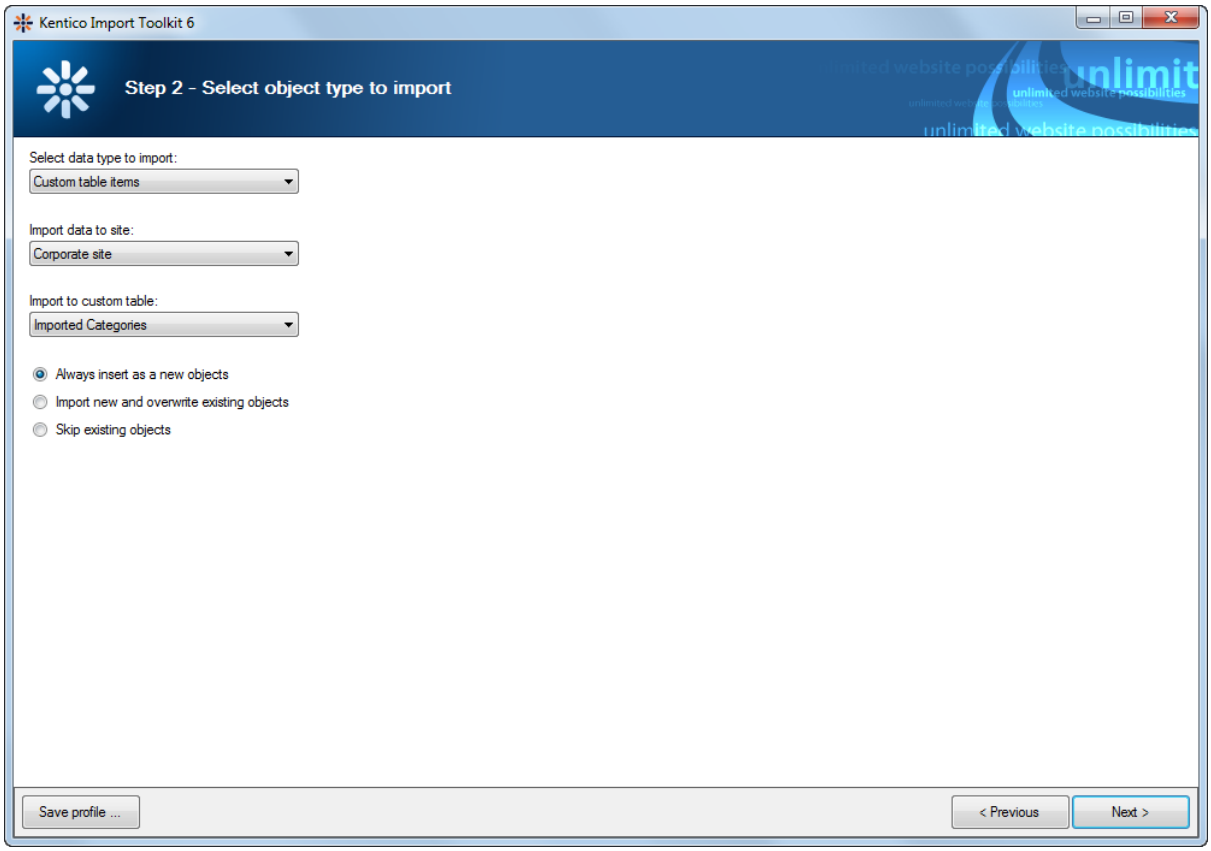
With data types that can be site-related, the **Import data to site** drop-down list is displayed. Using this drop-down, you can select a site to which the imported objects should be assigned. Where applicable, you can also choose **(global objects)** from the drop-down list to import the data as global objects not

bound to a specific website.

The radio buttons in the bottom part determine what will be done if the utility 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 them already exist in Kentico CMS.
- **Import new and overwrite existing objects/documents** - all imported data will be inserted, while those that already exist in Kentico CMS will be overwritten by the newly imported equivalents.
- **Skip existing objects/documents** - existing objects/documents will be skipped and only new ones will be imported.

Once you have performed the selection, click **Next** to proceed to the following step.



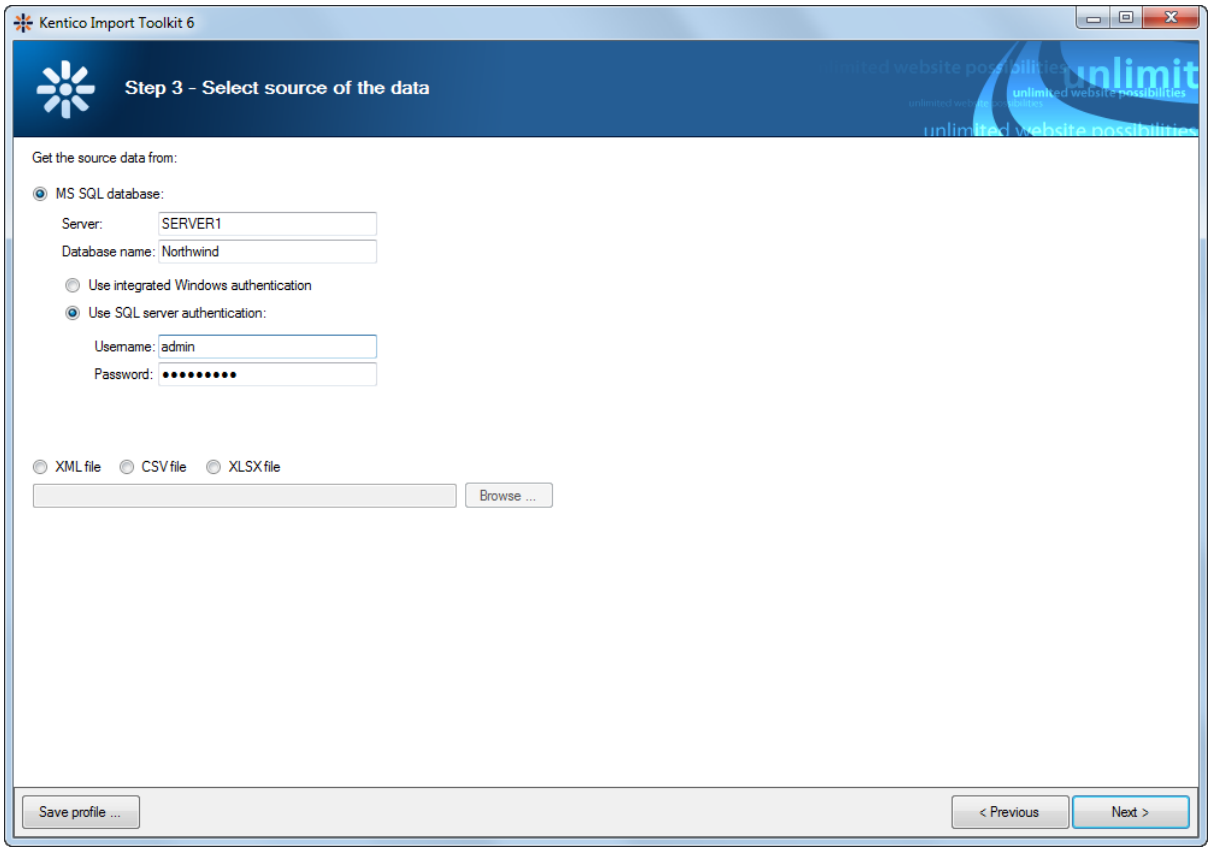
5. In **Step 3**, you need to specify the source of the imported data. Using the radio buttons, you can choose to import from:

- 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 when the respective source of data is selected.



9.2.3 Data selection from MS SQL database

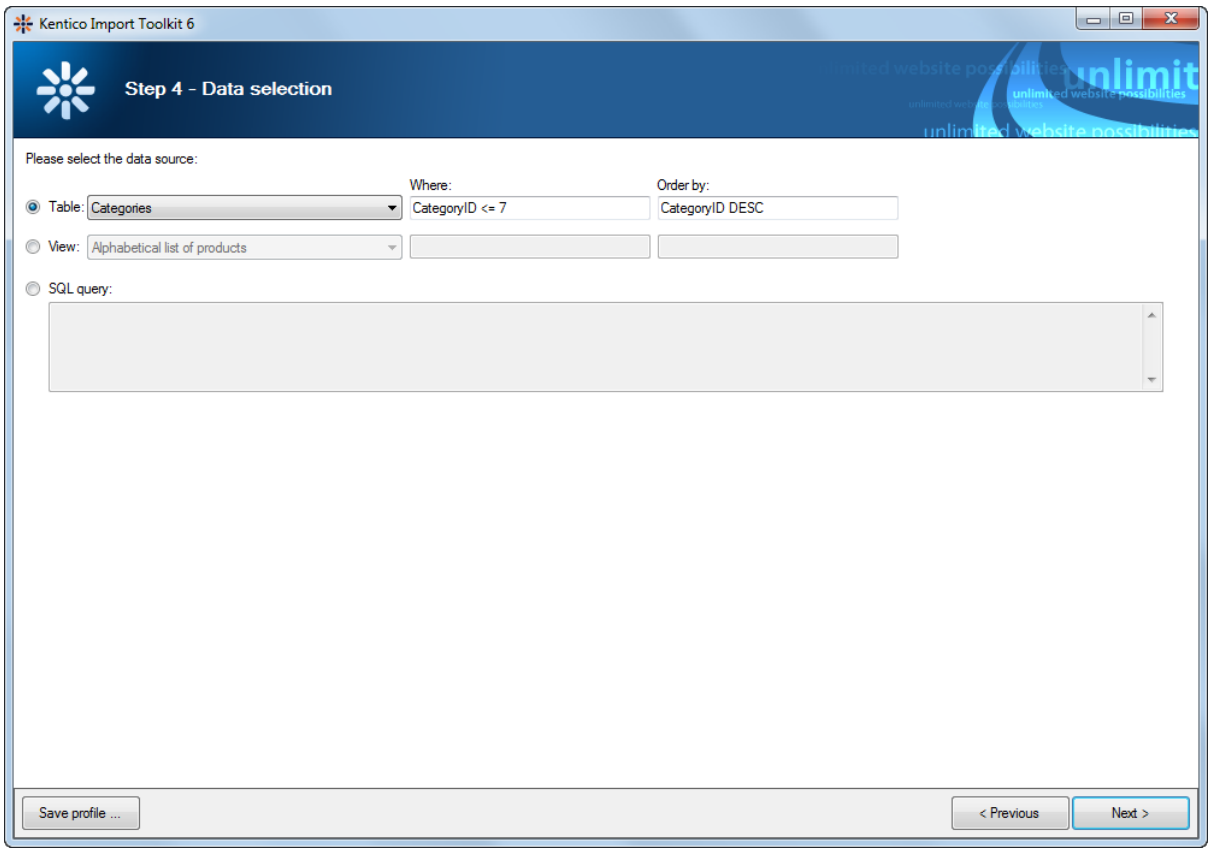
This topic contains information about the part of Kentico Import Toolkit wizard specific for import from an MS SQL database. The preceding steps of the wizard are explained in the Initial steps topic.

6. In **Step 4**, you can choose from the following options:

- **Table** - imports data from the table selected from the drop-down list, while imported data can be filtered using a WHERE condition and sorted using an ORDER BY expression entered into the respective fields.
- **View** - imports data from the view selected from the drop-down list, while imported data can be filtered using a WHERE condition and sorted 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.

The rest of the wizard is common for all sources of imported data and is explained in the Final steps topic.



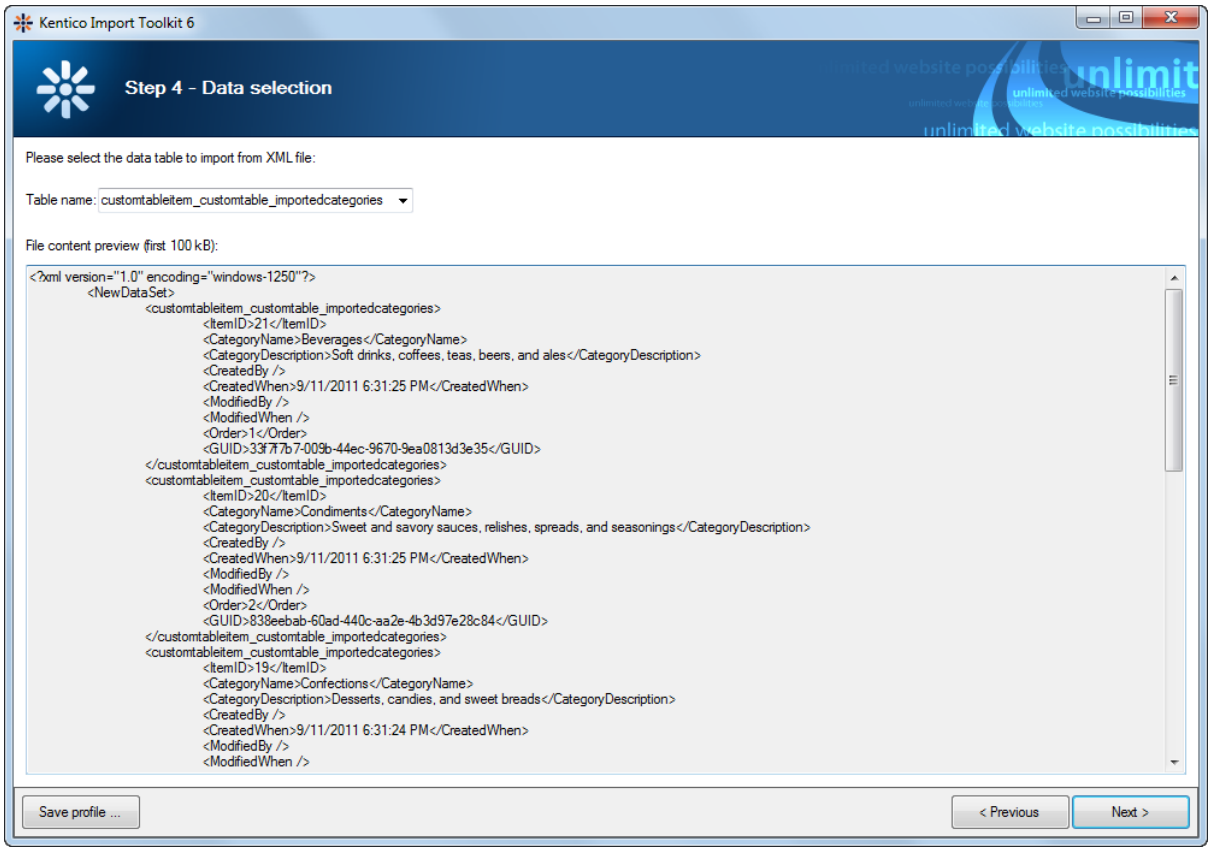
9.2.4 Data selection from XML file

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**, you need to 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.

The rest of the wizard is common for all sources of imported data and is explained in the Final steps topic.



9.2.5 Data selection from CSV or XLSX file

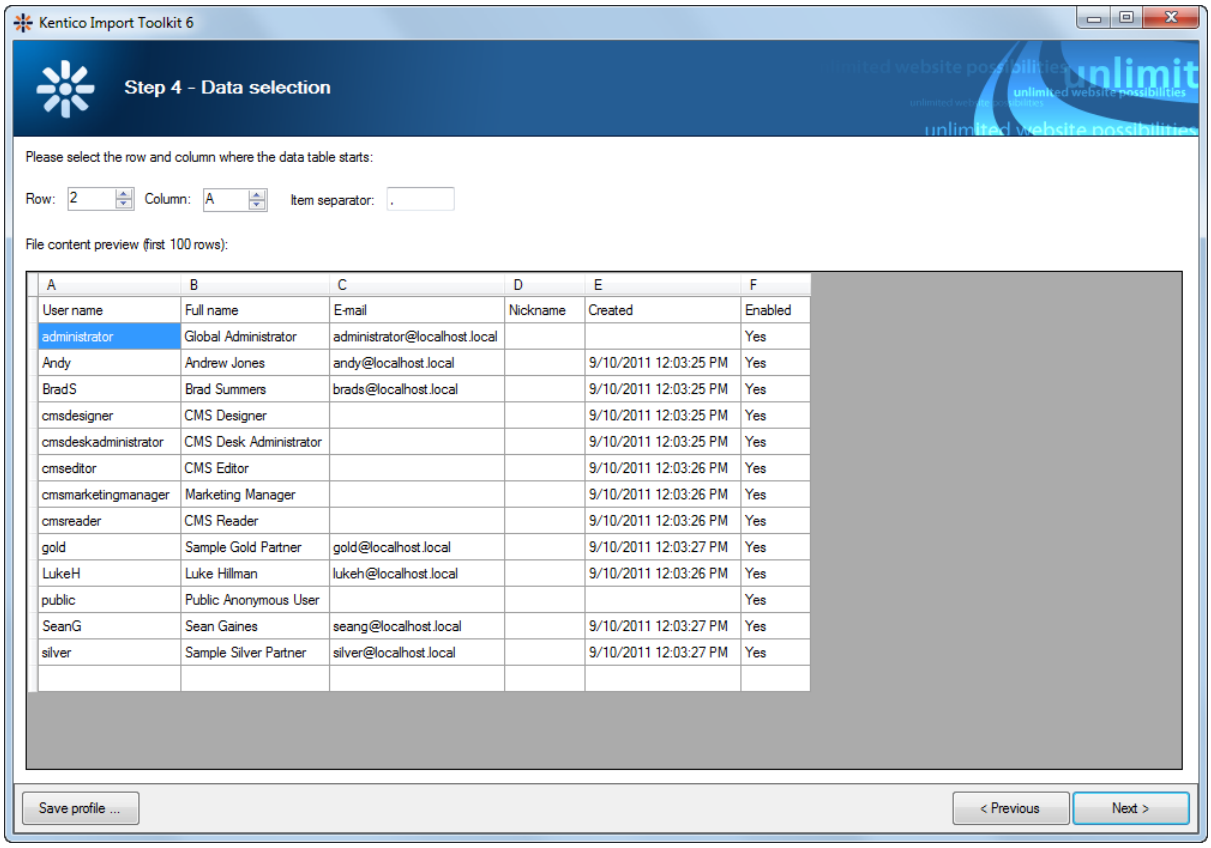
This topic contains information about the part of Kentico Import Toolkit wizard specific for import from a CSV of XLSX file. The preceding steps of the wizard are explained in the Initial steps topic.

6. In Step 4, you need to choose which row and column of the file will be taken as the beginning of the data grid. This is useful to exclude 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 mentioned above. Here, you can 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).

When selected, click **Next** to proceed to the following step.

The rest of the wizard is common for all sources of imported data and is explained in the Final steps topic.

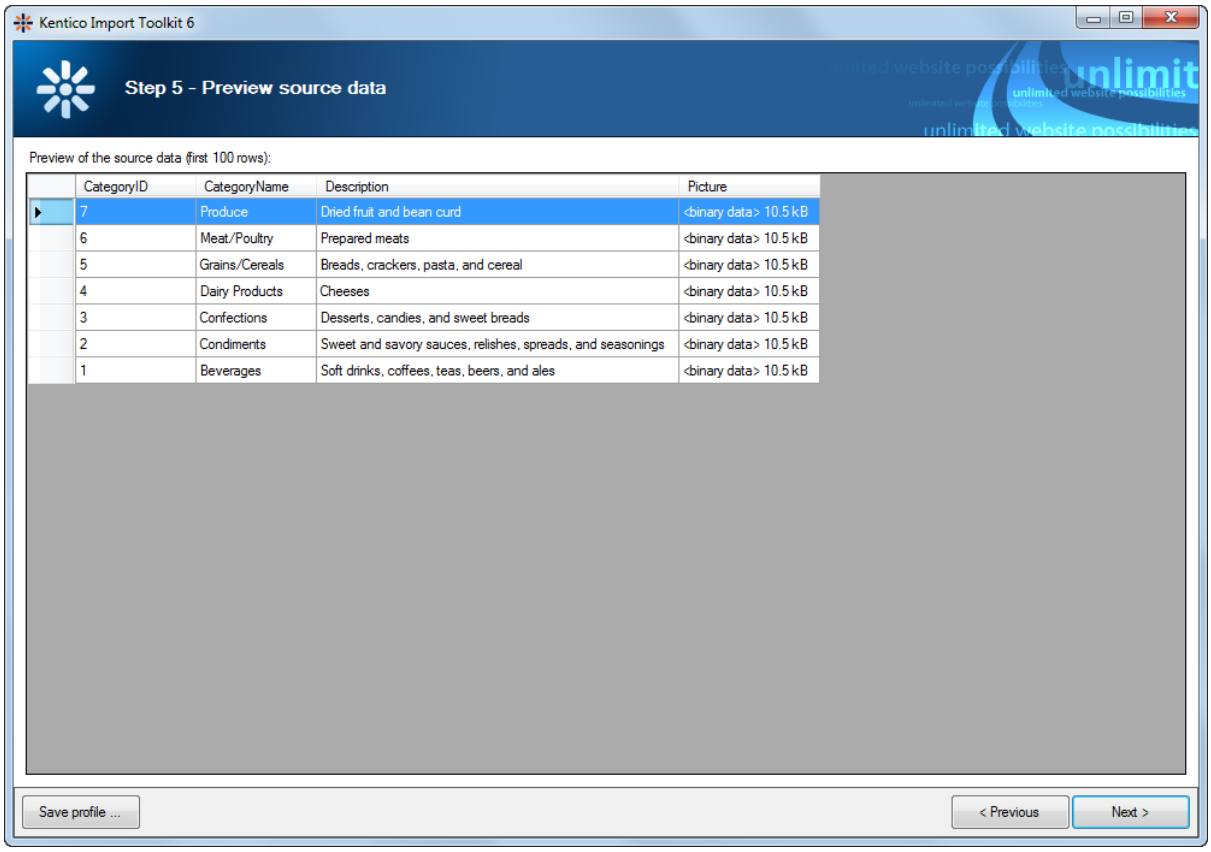


A	B	C	D	E	F
User name	Full name	E-mail	Nickname	Created	Enabled
administrator	Global Administrator	administrator@localhost.local			Yes
Andy	Andrew Jones	andy@localhost.local		9/10/2011 12:03:25 PM	Yes
BradS	Brad Summers	brads@localhost.local		9/10/2011 12:03:25 PM	Yes
cmsdesigner	CMS Designer			9/10/2011 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

9.2.6 Final steps

This topic explains the final steps of Kentico Import Toolkit wizard. To learn about the preceding steps, please 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**, a preview of the data to be imported is displayed. This step is only informative and you can use it to check if you specified the source data correctly in the previous steps. If so, click **Next** to proceed.



8. In **Step 6**, you 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 are matching, most of the fields should be mapped automatically and you only need to check and fine-tune the mappings. If you want to return to the default mappings after adjusting them, you can do it by clicking the **Reset to default mappings** link above the grid.

The mappings grid contains the following columns:

- **Target field** - field of the target object/document into which data should be imported. If you enable 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 its text to get the values dynamically:
  - **=macroexpression** - evaluates the specified macro expression using the source data and uses its result in the value. E.g. *={%ItemText%} - {%ItemOrder%}*.
  - **#<source>query** - executes a query against the source database and uses the result in the field. E.g. *#<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. E. g. *#<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 below for more details.

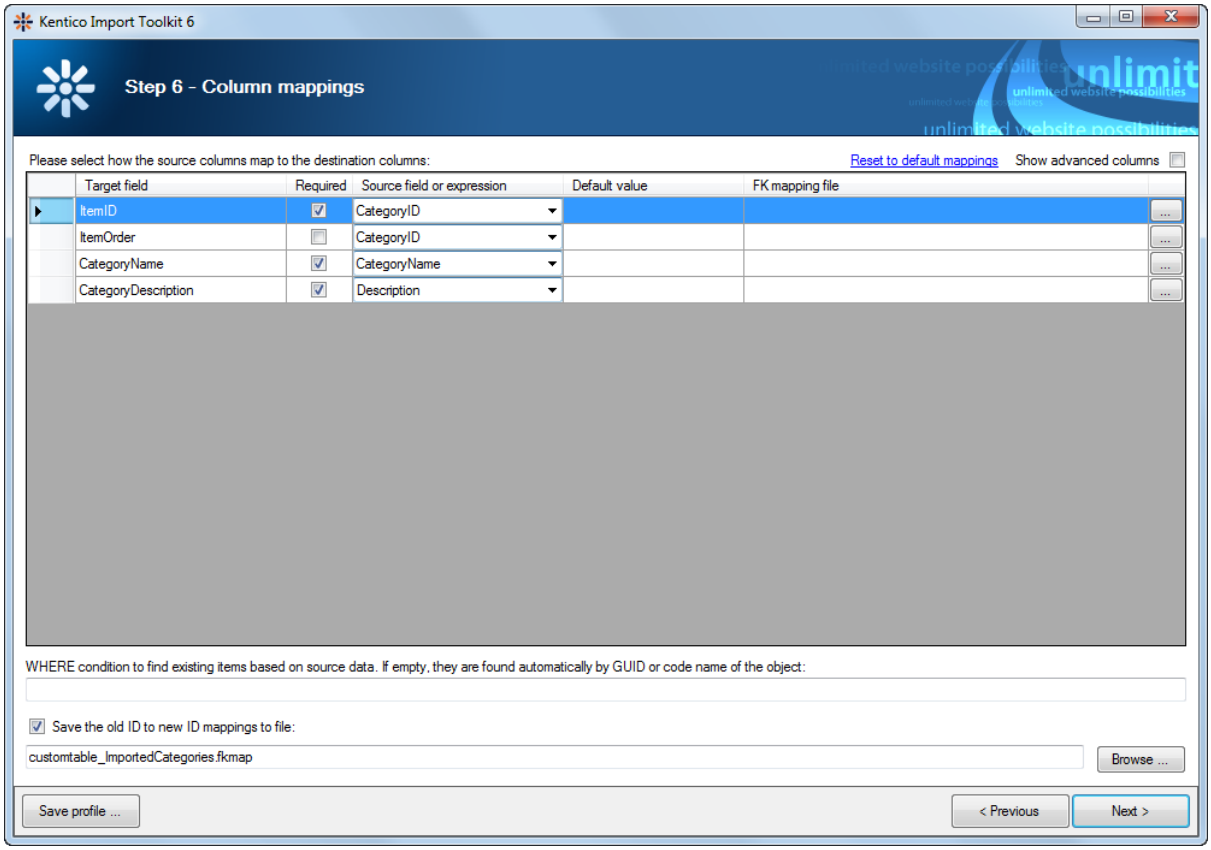
If you selected to skip or overwrite existing objects in Step 2 of the wizard, you can specify a WHERE condition in the field below the list. Based on this, the existing objects to be skipped or overwritten will be identified.

When importing the data as documents, there is an extra field below the grid where you need to specify a path to the document under which you want to created the imported documents. When importing the data as document attachments, there is also an extra field below the grid while in this case, you need to specify a path to the document to which you want to attach the imported attachments.

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. Let's explain the use of the file by the following example:

- Imagine you have a list of question and a list of answers where an answer is bound to a question with a foreign key.
- You want to import both and preserve their binding.
- You first import the questions, and save the ID mapping to a *questions.fkmap* file.
- Then you 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.

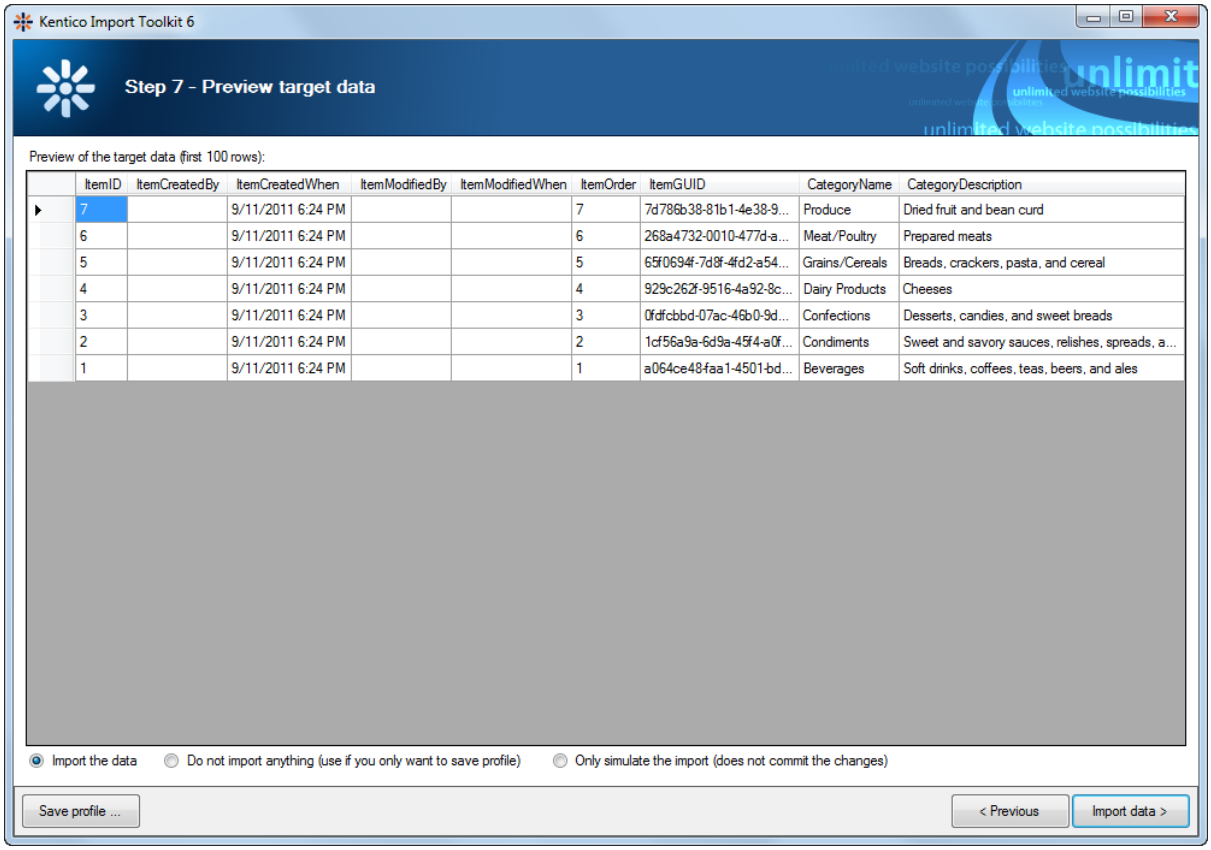


9. In **Step 7**, you can see a preview of the data as it will look when imported to the target. Before

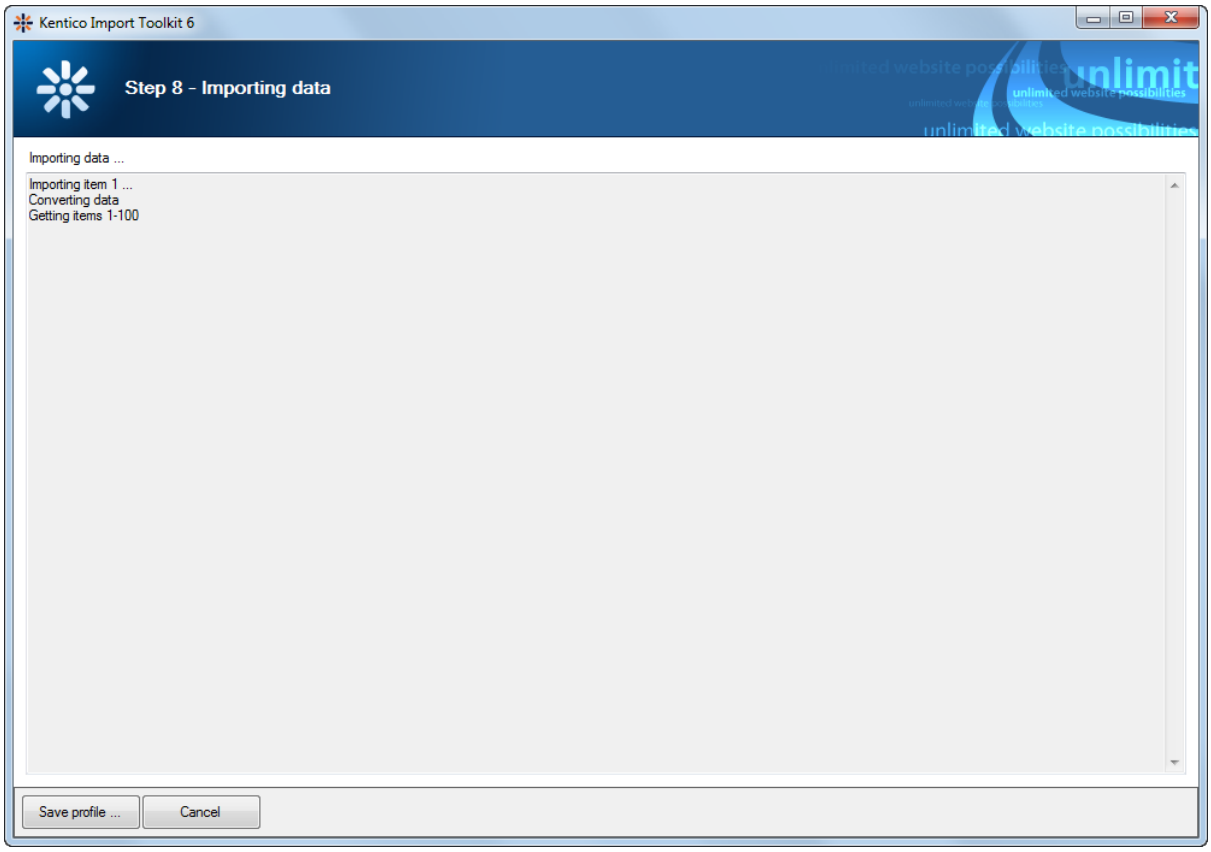
proceeding, you can choose how you want the data to be imported:

- **Import the data** - imports the specified data into Kentico CMS.
- **Do not import anything (use if you only want to save profile)** - skips the actual import. This is useful if you want to save the profile in the final step of the wizard. Please note that the profile can also be saved in any other step by clicking the **Save profile** button.
- **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.

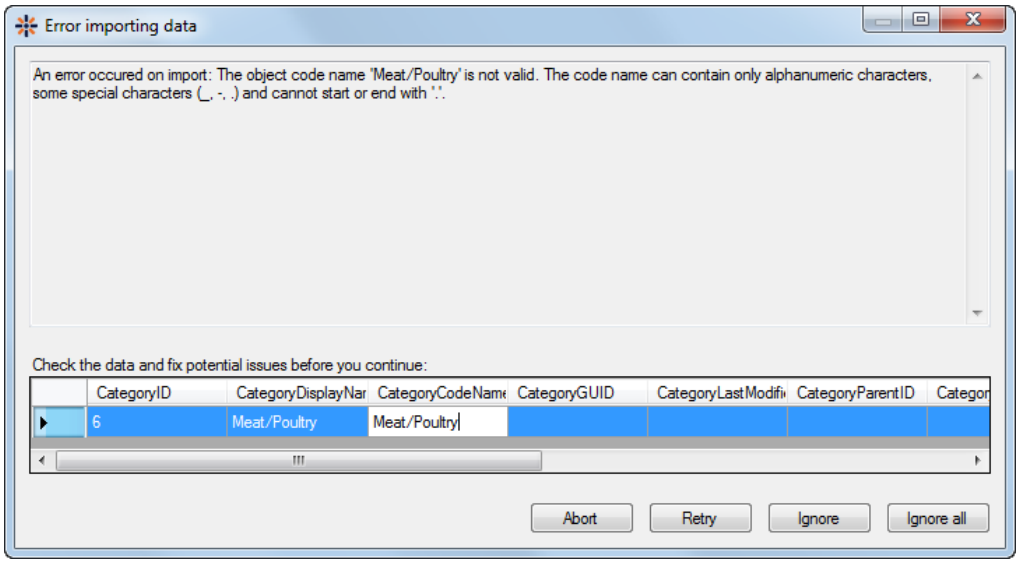


10. A log will be displayed, showing you the progress of the import.



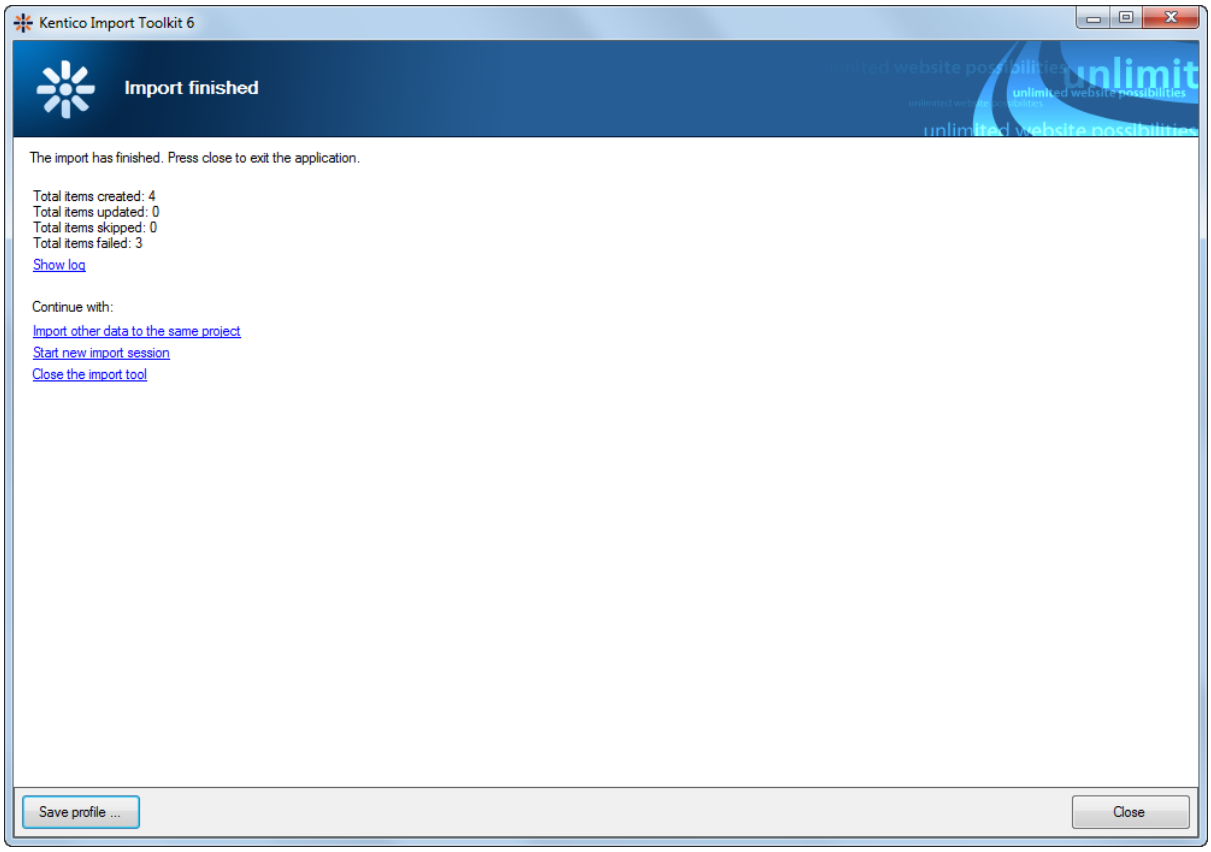
11. When an error occurs during the import (e.g. when there are restricted characters in imported data, NULL values in required columns, etc.), the dialog depicted in the screenshot below is displayed. In it, you can see an error message explaining the issue and you have the following options:

- **Abort** - aborts the whole import process.
- **Retry** - you can correct data in the displayed row and click this button to import the corrected data, without the need to correct the data in the source and go 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, without this window being displayed.



12. At the end, a final screen of the wizard is displayed, informing you about the result of the import. Imported data should already be visible in Kentico CMS at this point. The following actions can be performed on the final screen:

- **Show log** - opens the import log that was displayed while import was being performed.
- **Import other data to the same project** - redirects you to Step 2 of the wizard, with the web project used in the previous session preselected in Step 1.
- **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.

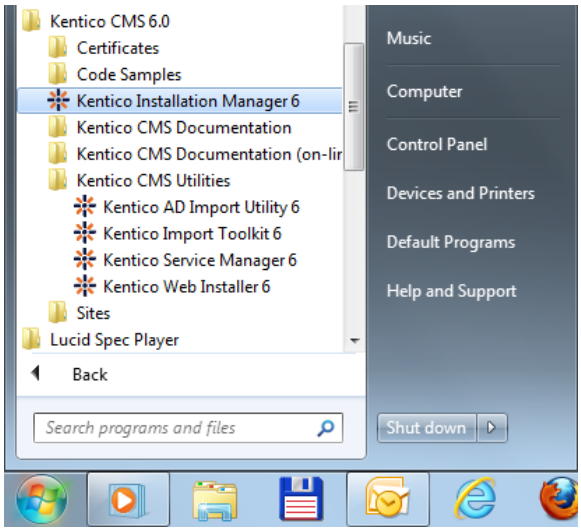


9.3 Kentico Installation Manager

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. All instances installed on a single machine can be managed from this single tool, which greatly simplifies their management on machines where a large number of instances is installed.

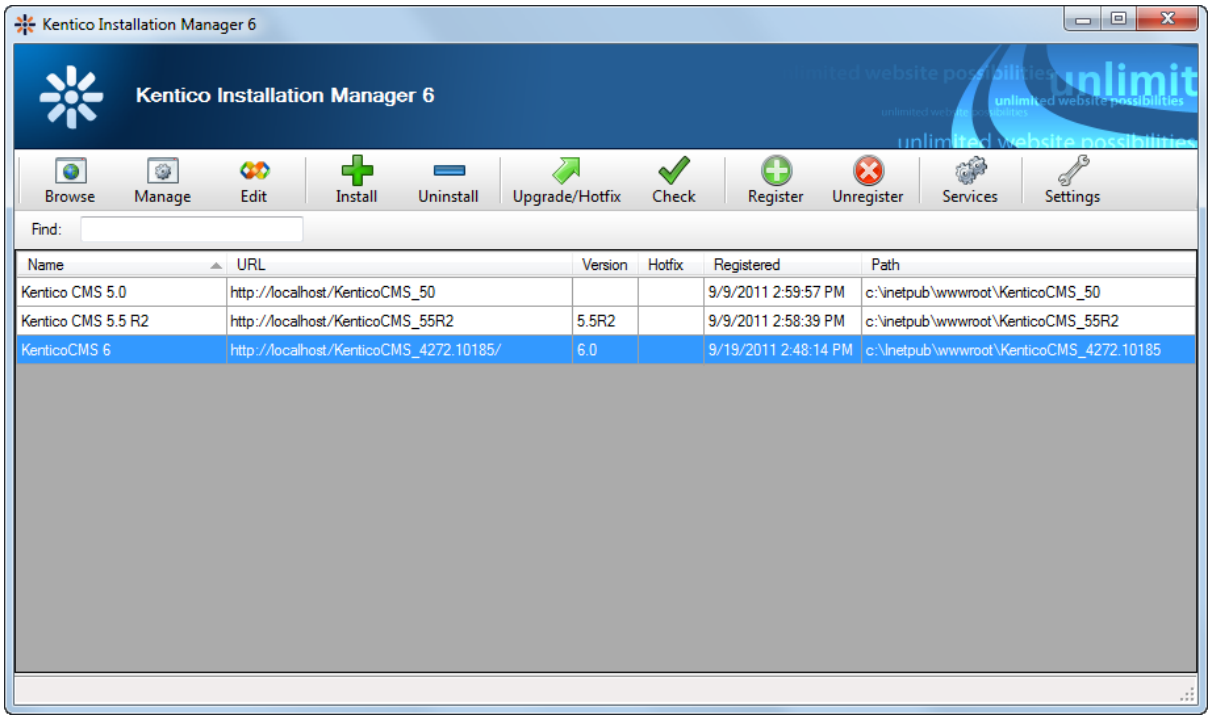
The utility can be launched by selecting **Kentico Installation Manager** from the Kentico CMS folder in Windows Start menu.



In the toolbar, you can find the following actions:

- **Browse** - opens the URL in the URL column of the selected instance in a new browser window. If the instance contains multiple websites and the **Show site selection after clicking Browse** option is enabled in configuration of the instance, a pop-up window is displayed after executing the action, letting you select which of the instance's websites should be opened.
- **Manage** - opens 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 can be used to install a new Kentico CMS instance.
- **Uninstall** - uninstalls the selected instance, optionally including removal from file system, IIS, database and from the instances list in KIM.
- **Upgrade/Hotfix** - opens a pop-up window where the currently selected instance can be upgraded or hotfixed. 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 a new instance can be registered for management by KIM (added to the list below). See Instance registration for more details.
- **Unregister** - unregisters the selected instance from KIM (removes it from the list below). See Instance registration for more details.
- **Services** - opens Kentico Service Manager, which can be used to install and manage Kentico CMS Windows services.
- **Settings** - opens a dialog where settings related to upgrading and hotfixing can be adjusted. See Upgrading and hotfixing for more details.

In cases when large numbers of instances is registered in the utility, you can use the **Find** textbox below the toolbar to search particular instances by the value stored in their **Name** column.



9.3.2 Instance registration

Installed Kentico CMS instances need to be registered in Kentico Installation Manager in order for their management by the utility to be possible. 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.
- **<showlist>** - indicates if a pop-up window where the website to be opened can be selected is displayed after executing the **Browse** action.

Here is a sample extract from the file:

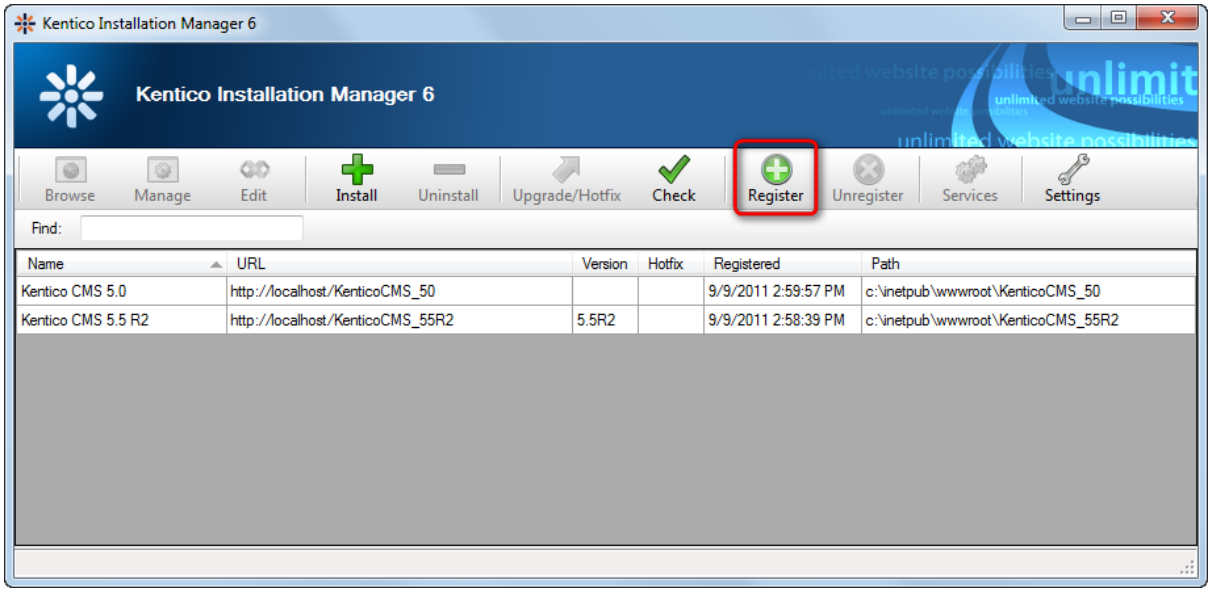
```
<?xml version="1.0"?>
<sites>
  <item>
    <guid>3720a09b-43f7-4d26-8fec-e4b26637a2f3</guid>
    <url>http://localhost/KenticoCMS_4269.16734</url>
    <name>Kentico CMS 6</name>
```

```
    <version>6.0</version>
    <path>c:\inetpub\wwwroot\KenticoCMS_4269.16734</path>
    <created>2011-09-09T14:21:17</created>
    <hotfix>0</hotfix>
    <netversion>4</netversion>
    <showlist>1</showlist>
  </item>
  ...
</sites>
```

Registering a Kentico CMS instance

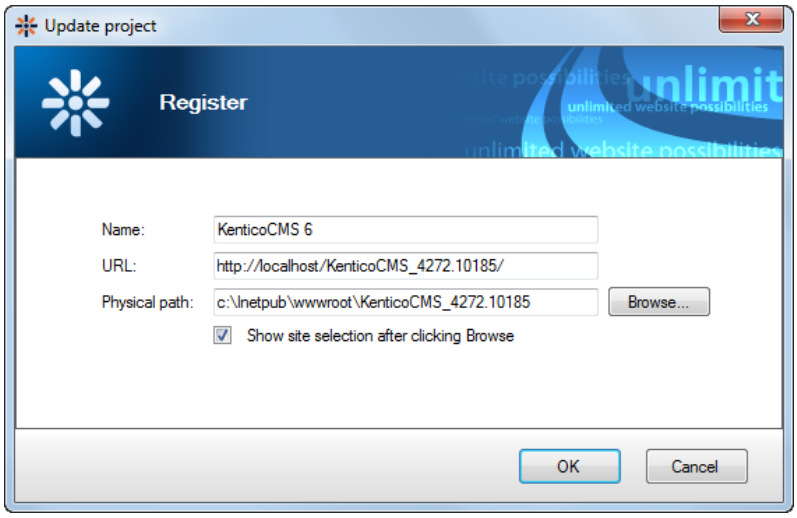
Registration of newly installed Kentico CMS instances is performed automatically. In case that you have Kentico CMS instances from versions prior to 6.0 already installed on your machine, you can register them by performing the following steps:

1. Click **Register** in the main toolbar.

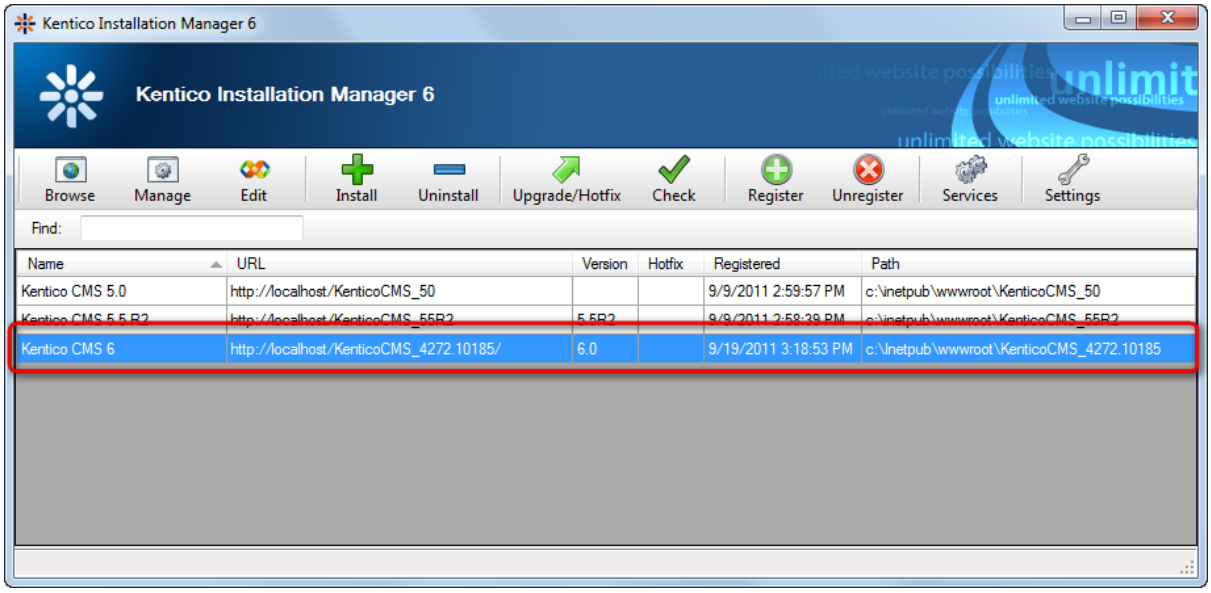


2. In the pop-up window, enter the following details:

- **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, a pop-up window where the website to be opened can be selected is displayed after executing the **Browse** action (useful for multi-site instances).



3. The instance should now be registered and visible in the list of managed instances.



9.3.3 Upgrading and hotfixing

Kentico CMS instances can be upgraded and hotfixed using Kentico Installation Manager. There are two actions related to these tasks on the main panel:

- **Upgrade/Hotfix** - opens a pop-up window where the currently selected instance can be upgraded or hotfixed.
- **Check** - checks if there are any new upgrades or hotfixes available for any of the managed instances.

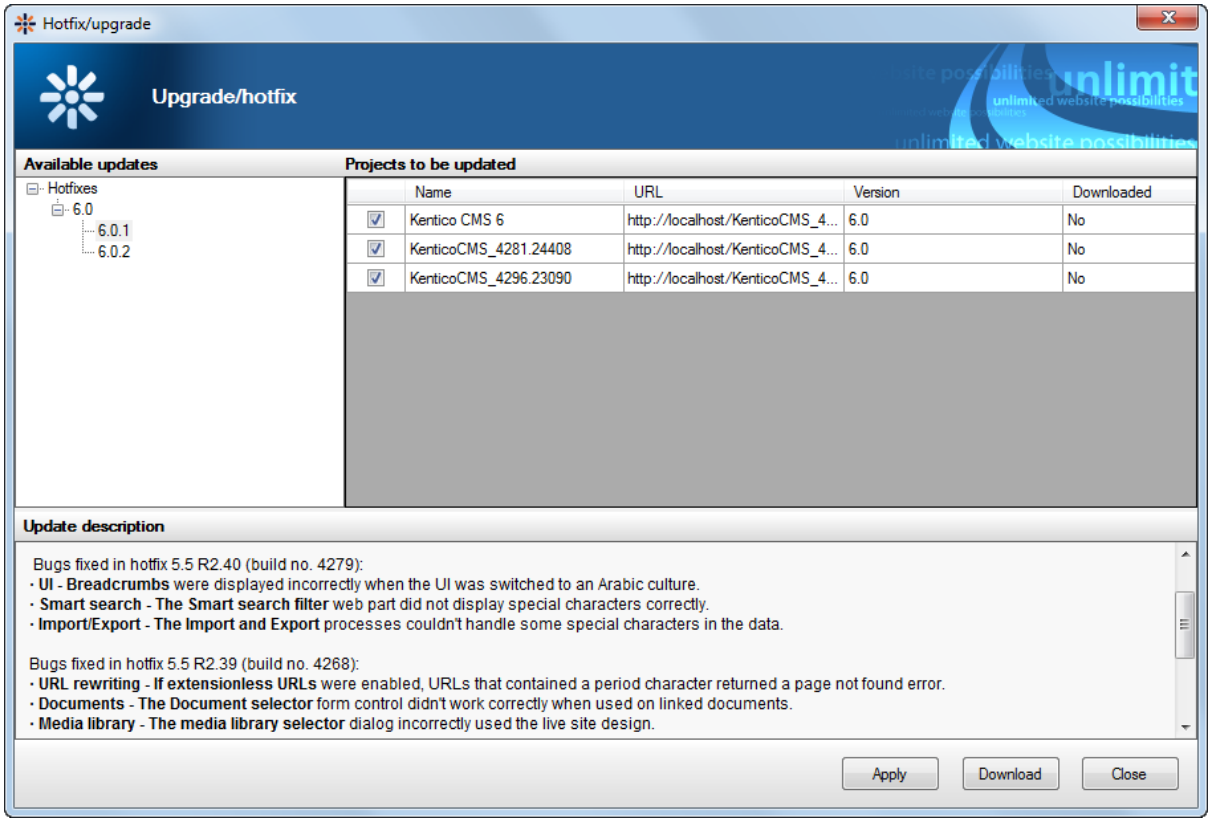
Upgrade/Hotfix action

You can upgrade or hotfix a Kentico CMS instance by selecting it in the list and clicking **Upgrade/Hotfix** in the main toolbar. After doing so, a pop-up window is displayed, listing all available upgrades

and hotfixes in the tree on the left. After selecting an upgrade or hotfix from the tree, a list of all instances to which it can be applied is displayed in the list on the right, with description of the upgrade or hotfix below the list. After selecting an instance from the list, the following options are available:

- **Apply** - the upgrade or hotfix will be downloaded and the Kentico Upgrade or Hotfix Utility contained in the package launched.
- **Download** - the upgrade or hotfix will only be downloaded, without being applied. It will be stored in *C:\ProgramData\KIM* and can be applied later.

When finished, click **Close** to return to the main window.

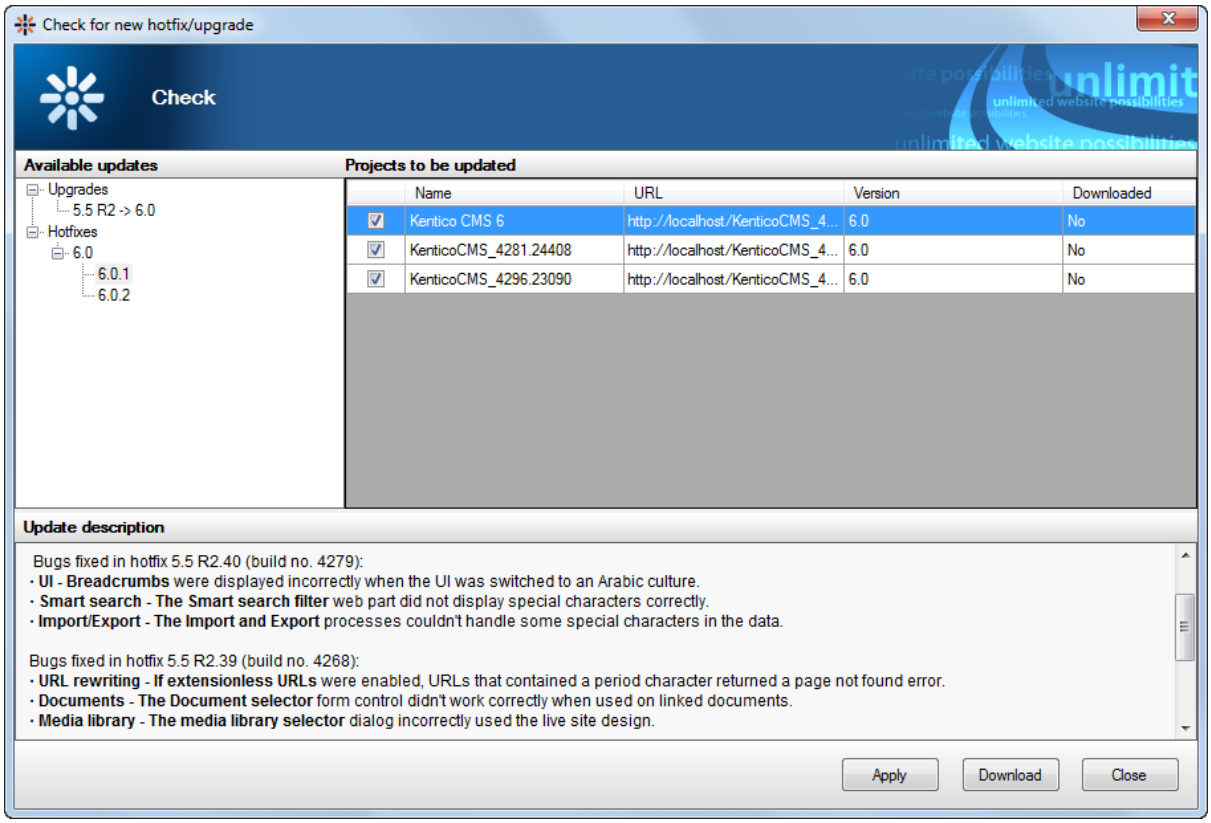


Check action

To check if any of the registered instances can be be upgraded or hotfixed, click the **Check** in the main toolbar. After doing so, the same pop-up window as described above for the **Upgrade/Hotfix** action is displayed. The difference is that in this case, upgrades and hotfixes for all registered instances is displayed in the tree on the right. Otherwise, the functionality is identical: after selecting a hotfix or upgrade, a list of all instances to which it can be applied is displayed in the list on the right. When an instance is selected, the same two actions are available:

- **Apply** - the upgrade or hotfix will be downloaded and the Kentico Upgrade or Hotfix Utility contained in the package launched.
- **Download** - the upgrade or hotfix will only be downloaded, without being applied. It will be stored in *C:\ProgramData\KIM* and can be applied later.

When finished, click **Close** to return to the main window.



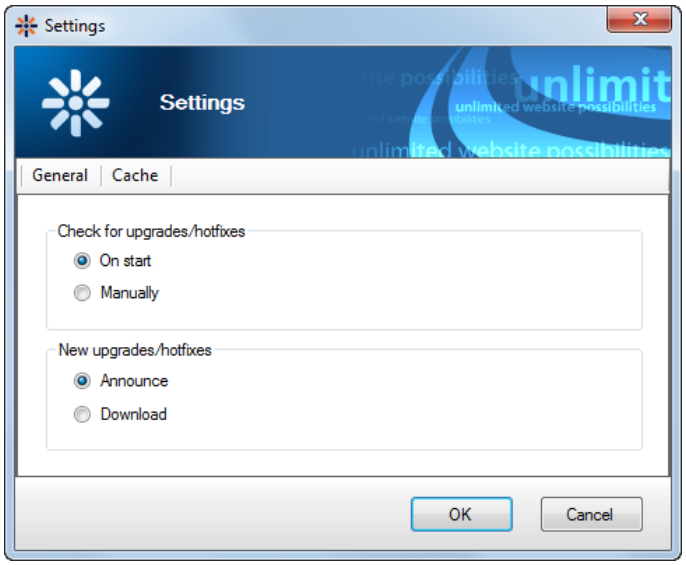
Upgrade and hotfix settings

Settings related to upgrading and hotfixing can be adjusted after clicking **Settings** in the main toolbar.

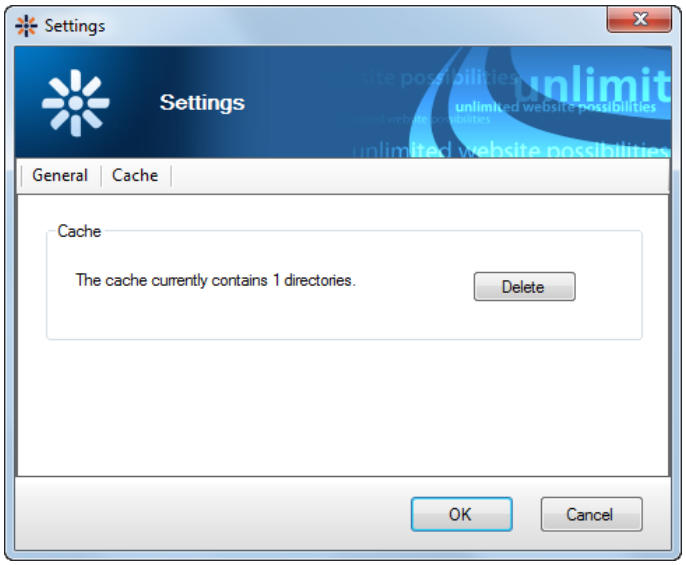
On the **General** tab, you can adjust the following settings:

- **Check for upgrades/hotfixes** - determines how KIM will check if new hotfixes or upgrades are available for registered instances.
  - **On start** - if selected, the check will be performed automatically on each start of KIM.
  - **Manually** - if selected, the check will only be performed after selecting the **Check** action from the toolbar.

- **New upgrades/hotfixes** - determines what should be done when new upgrades or hotfixes are found. Only applicable if **On start** is selected above.
  - **Announce** - a notification message will be displayed when new upgrades or hotfixes are found.
  - **Download** - all new upgrades and hotfixes will be downloaded automatically.



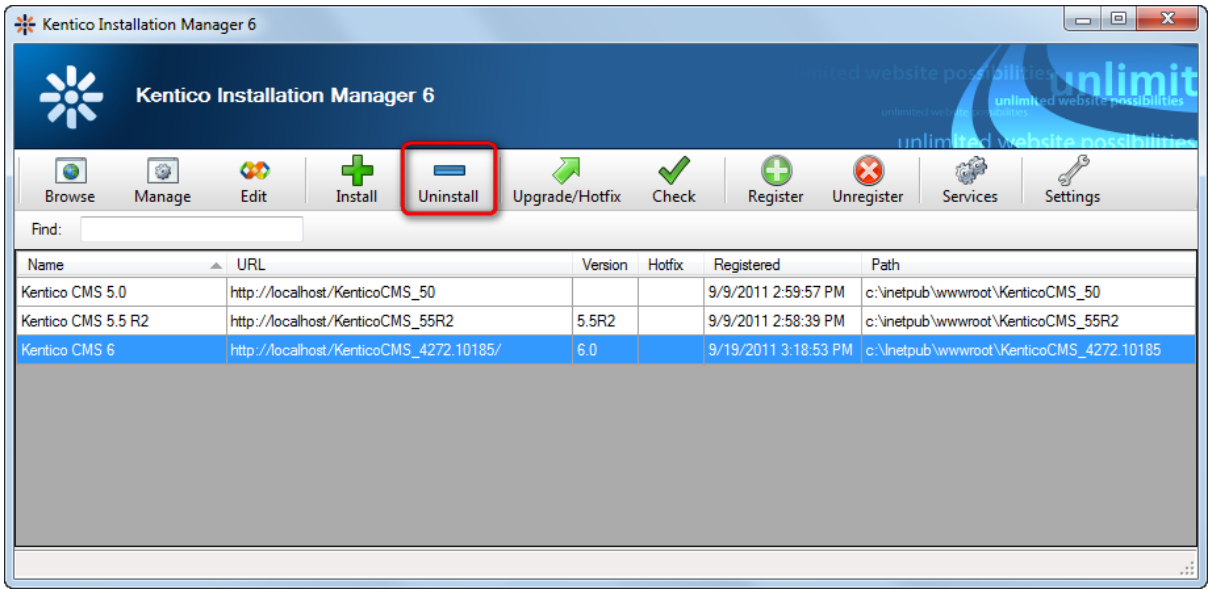
On the **Cache** tab, you can see the number of temporary upgrade and hotfix folders. These folders contain upgrade/hotfix data unpacked from downloaded upgrade/hotfix packages and stored are in *C:\ProgramData\KIM*. You can delete these temporary folders by clicking the **Delete** button.



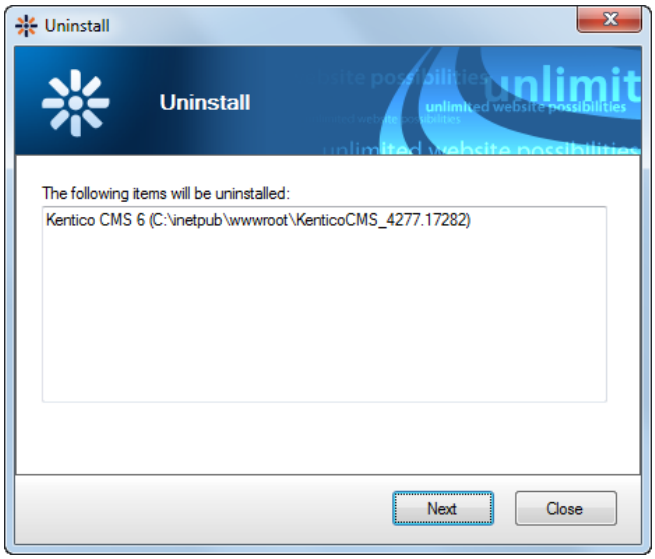
9.3.4 Uninstallation

Kentico Installation Manger can be used to uninstall a Kentico CMS instance, including its project folder in the file system, website in IIS and database on a database server. To uninstall an instance, perform the following steps:

1. Select the instance that you want to uninstall and click **Uninstall** in the main toolbar.



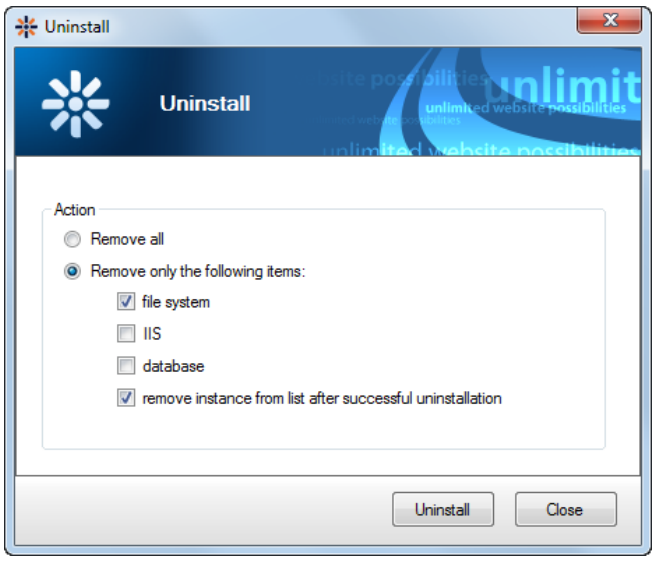
2. A pop-up window will be displayed, showing the name and path of the website to be uninstalled. Click **Next**.



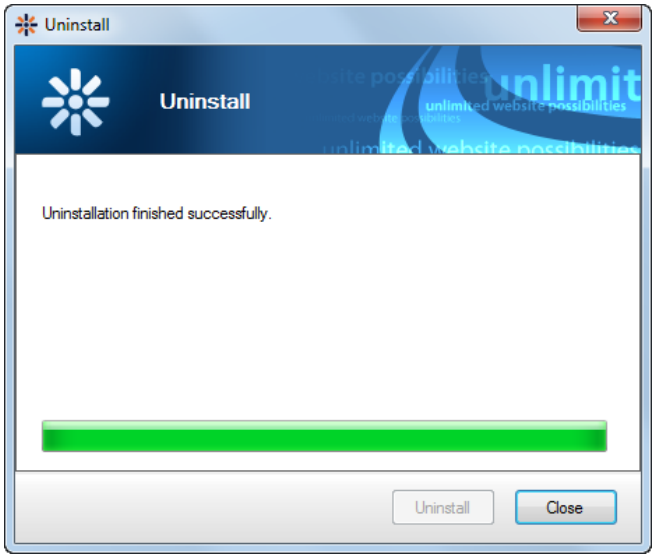
3. In the next step, choose which parts of the installation will be uninstalled:

- **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.

When selected, click **Uninstall**.



4. A log will be displayed, showing the progress of the uninstallation. When the process finishes successfully, you should be notified that the instance is uninstalled and you can click **Close** to return to the main window.

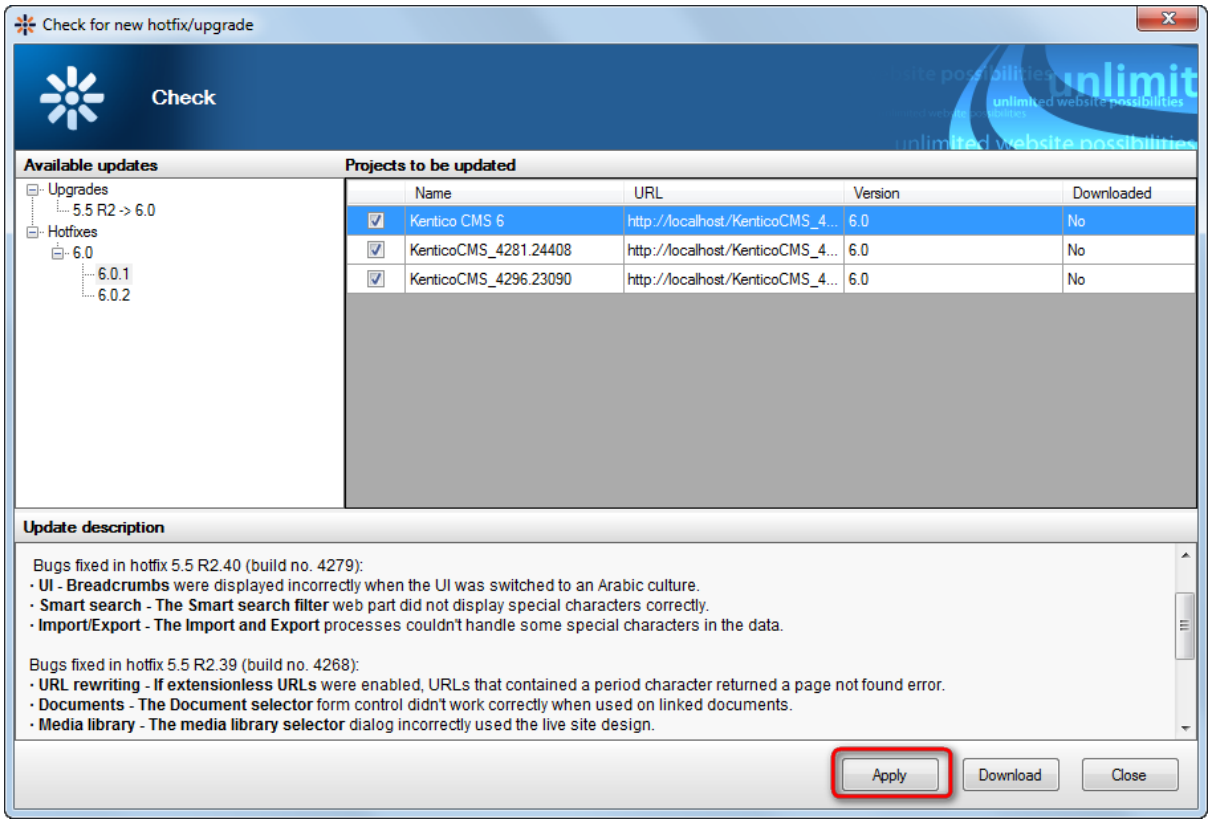


9.4 Kentico Hotfix and Upgrade Utility

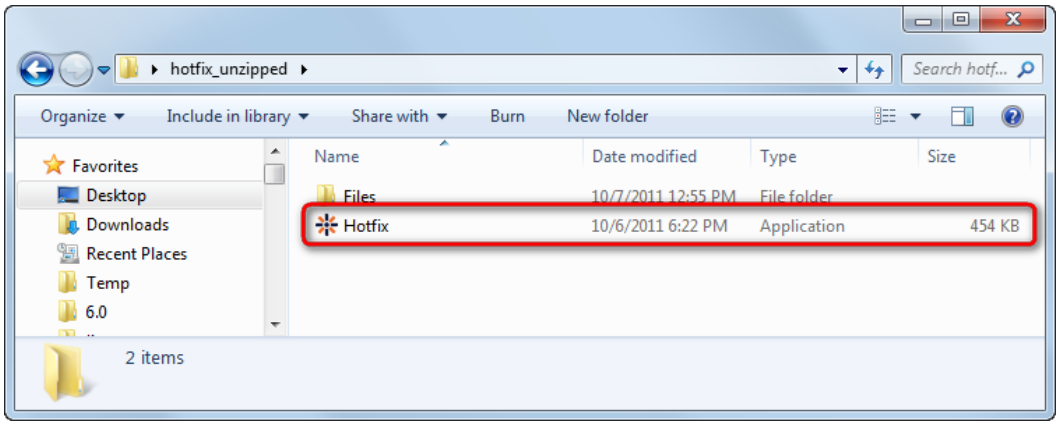
9.4.1 Overview

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 two ways:

1. By clicking the **Apply** button in the Upgrade/Hotfix and Check dialogs of Kentico Installation Manager.



2. By executing the file unpacked from the package. It must be placed in a folder together with the folder containing the actual hotfix or upgrade files.



Please refer to the Using the wizard topic for a step-by-step guide through individual steps of the utility's wizard.

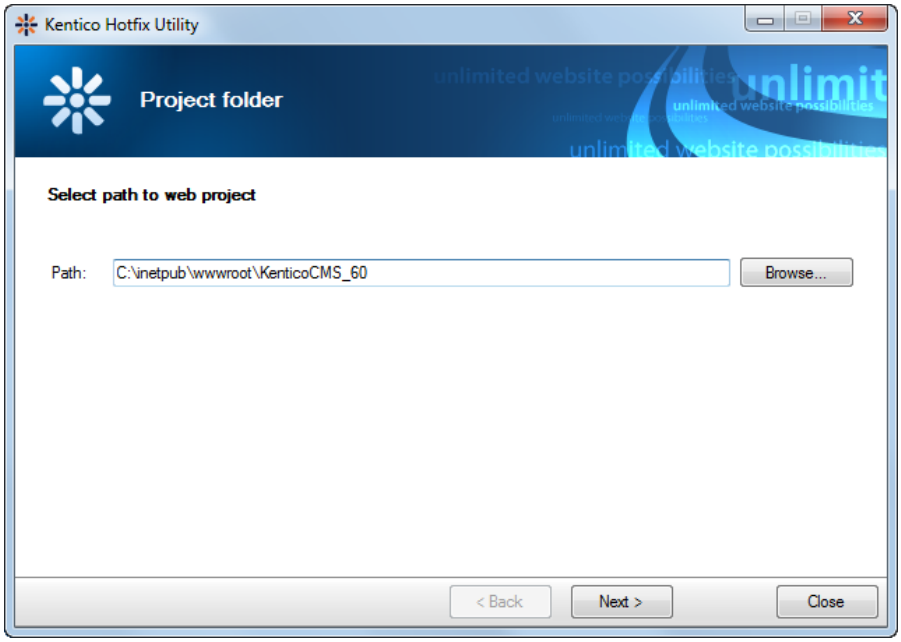
9.4.2 Using the wizard

The hotfix or upgrade process consists of the following steps:

1. In the first step, you need to specify the root folder of the Kentico CMS instance that you want to

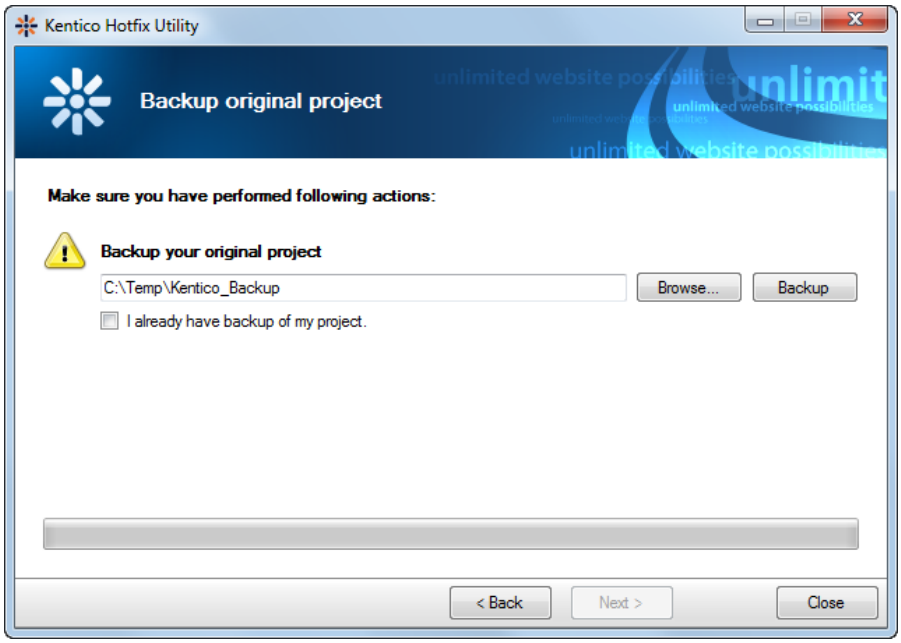
upgrade or hotfix. This step is only displayed when launching the utility manually. If you launch the utility from Kentico Installation Manager, the wizard will start with the following step, while the upgrade or hotfix will be applied to the instance selected in Kentico Installation Manager.

Once selected, click **Next** to proceed to the following step.



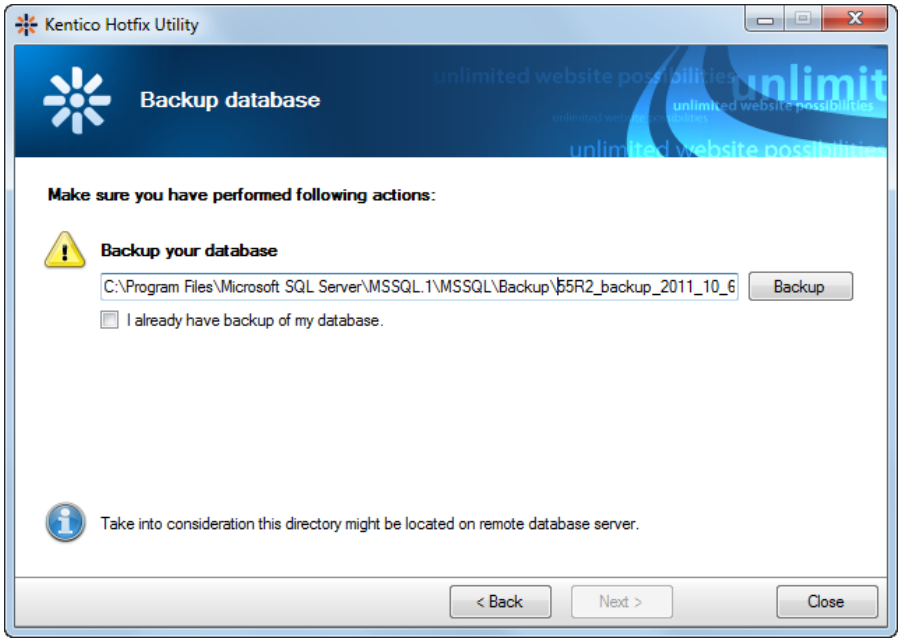
2. This step can be used to backup the project before it will be upgraded. This can be done by specifying a folder in the textbox and clicking the **Backup** button. If you have performed the backup before launching the utility, you can check the **I already have backup of my project** check-box and proceed.

After performing the backup or enabling the check-box, the **Next** button gets enabled. Click it to proceed to the subsequent step.



3. The third step is also intended for backup purposes, while this one offers the option to backup the instance's database. A path to a new folder under the MS SQL Server default backup folder is pre-filled by default, while you can change the path to any custom one. When a path is specified, click **Backup** to backup the database. If you already have a backup of the instance's database, you can check the **I already have backup of my database** check-box and proceed.

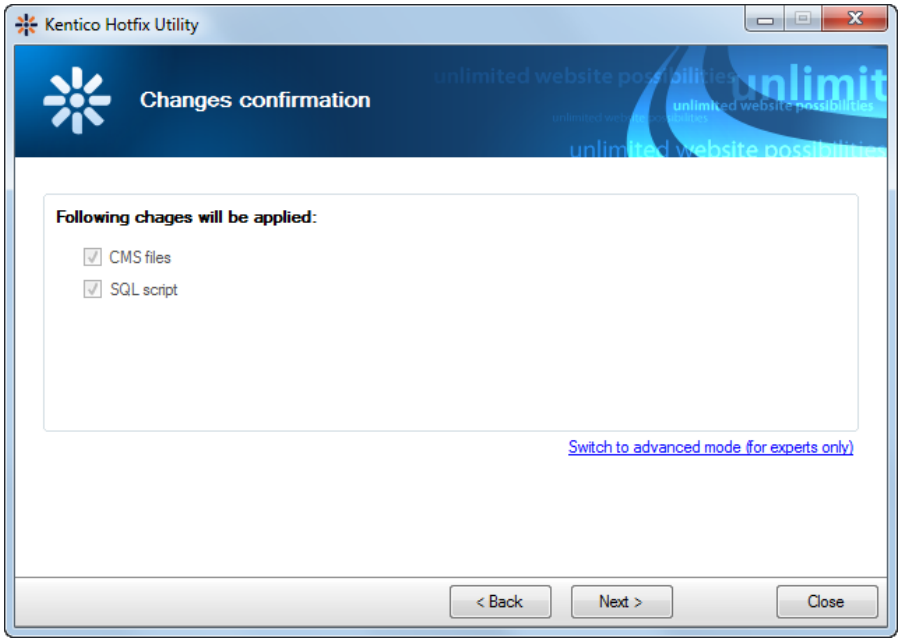
After performing the backup or enabling the check-box, the **Next** button gets enabled. Click it to proceed to the subsequent step.



4. The fourth step determines if the upgrade or hotfix should replace project files, execute SQL scripts, or both. Both options are enabled by default. If you click the **Switch to advanced mode (for experts**

only) link, the utility will let you select which of the two actions will be performed using the check-boxes.

Once selected, click **Next** to proceed.



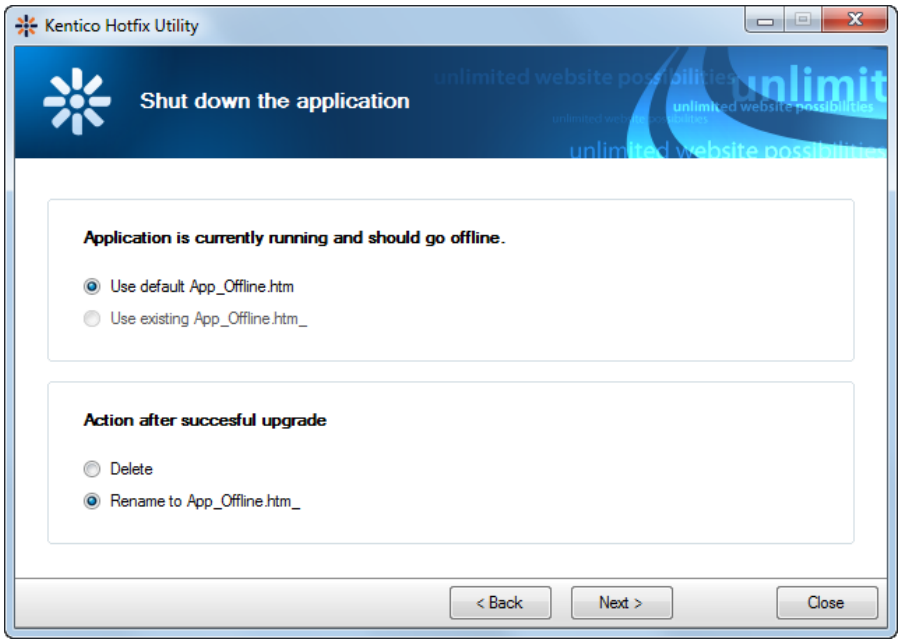
5. When applying a hotfix or upgrade, the Kentico CMS instance needs to go offline. To get the instance offline, the utility can place the *App_Offline.htm* file into the root of the web project (see http://msdn.microsoft.com/en-us/library/ff925031.aspx for more details on the use of the file).

- **Use default App_Offline.htm** - creates a new *App_Offline.htm* file in the root of the web project.
- **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.

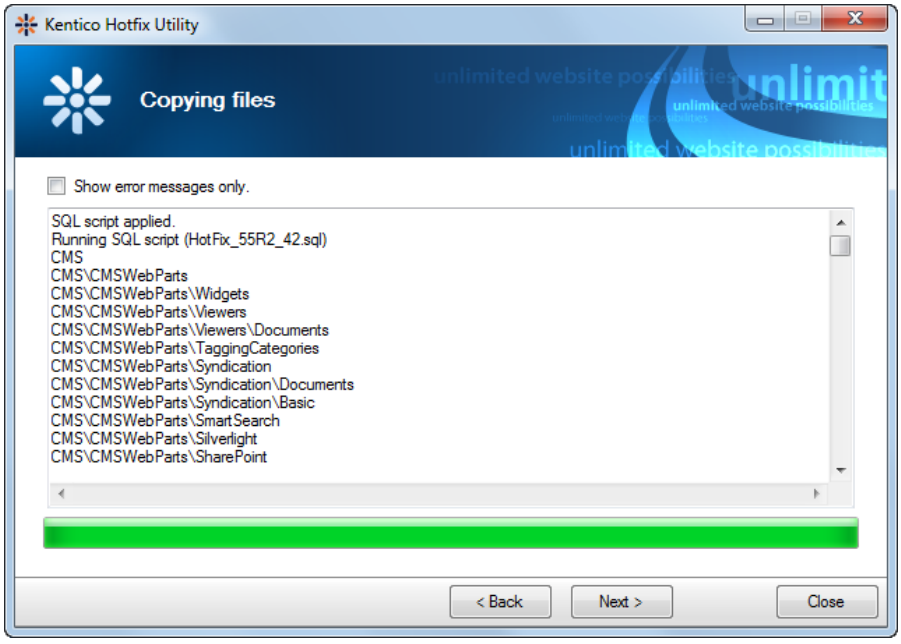
The second set of check-boxes lets you choose what to do with the *App_Offline.htm* file:

- **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 by choosing the *Use existing App_Offline.htm_* option above.

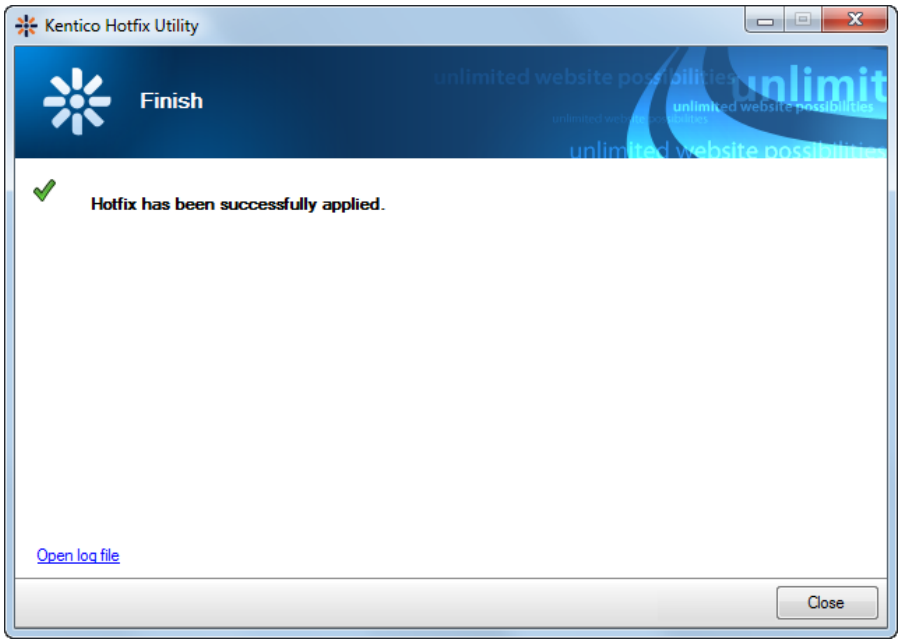
Once selected, click **Next** to execute the upgrade or hotfix procedure.



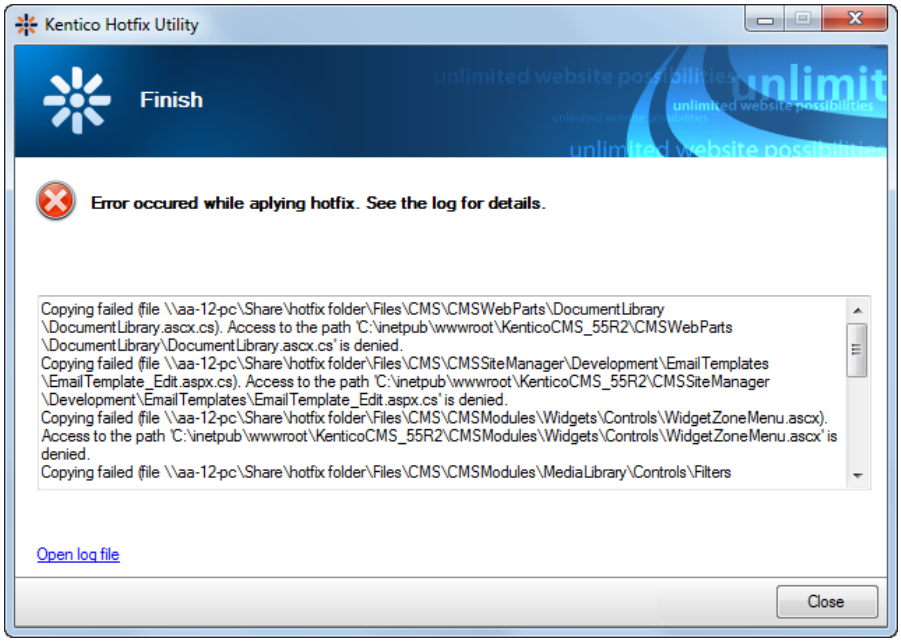
6. A log showing progress of the hotfix or upgrade will be displayed. You can enable the **Show error messages only** check-box to display only error messages in the log. Once the procedure finishes, the **Next** button gets enabled. Click it to proceed to the final step.



7. The final step only lets you know if the hotfix or upgrade finished successfully. By clicking the **Open log file** link, you 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.



If an error occurred while applying the hotfix or upgrade, the final step will notify about the fact and display the error log, as can be seen in the screenshot below.



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.

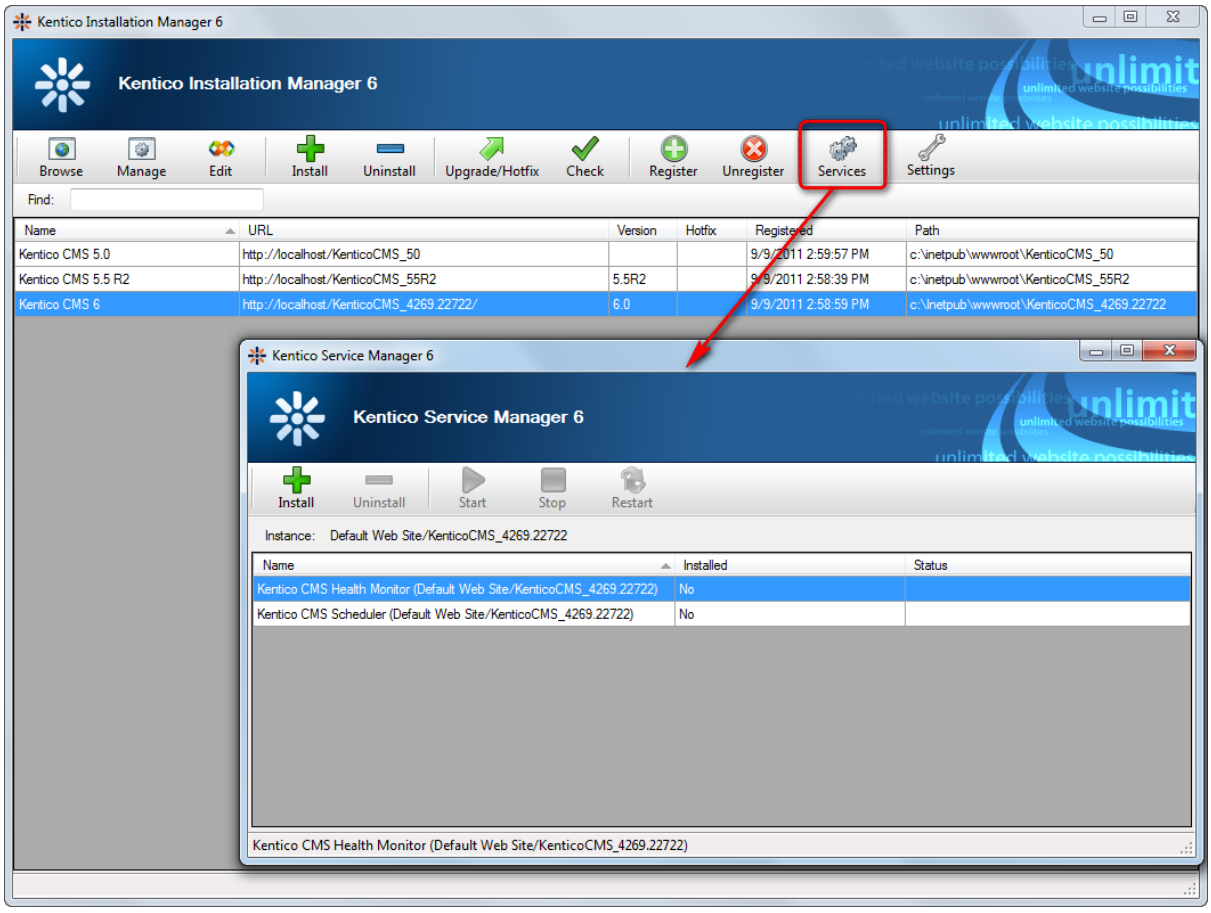
In the Using the utility topic, you can learn how services can be managed using the utility. Windows services for each particular Kentico CMS instance are defined in a dedicated XML file. Only services that are defined in the file can be managed by the Service Manager. To learn more about this file and its required format, please refer to the Services definition XML topic.

9.5.2 Using the utility

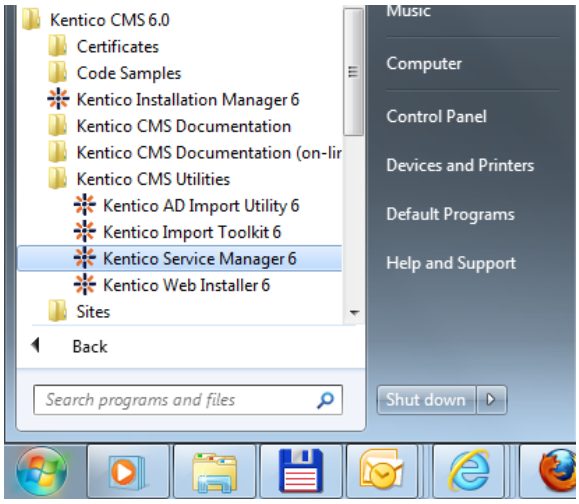
This topic explains how Kentico Service Manger can be launched, what problems may occur when launching it and how it can be used when launched successfully.

Launching the utility

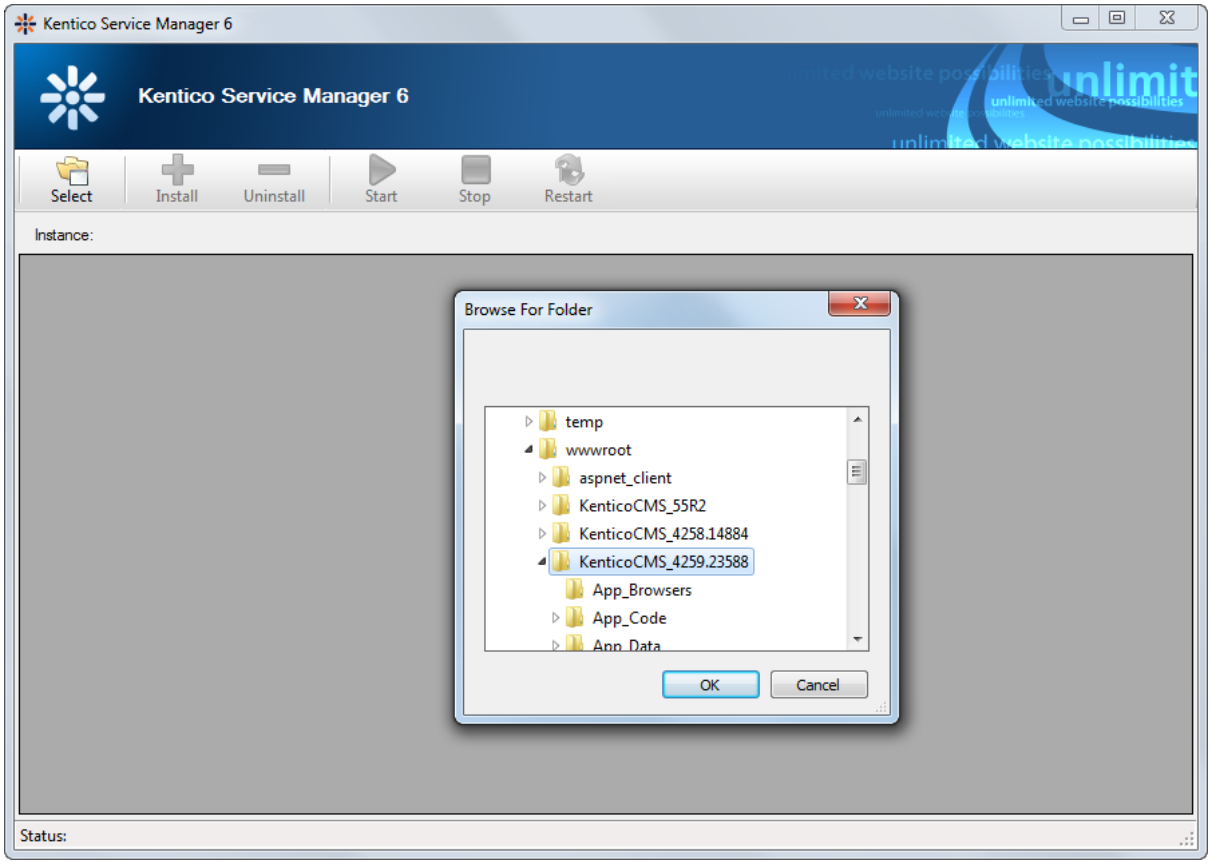
Kentico Service Manager can be launched two ways. The first option is to launch it by selecting an instance in Kentico Installation Manager and clicking the **Services** menu item in its toolbar. In this case, services of the selected instance will be managed.



Another options is to launch the Service Manager from Windows Start menu. It is located under **All programs -> Kentico CMS <version> -> Kentico CMS Utilities -> Kentico Service Manager**.



On each launch of the utility, the **Browse For Folder** dialog is displayed. In the dialog, you need to select the root of the Kentico CMS instance whose services you want to manage and click **OK**.

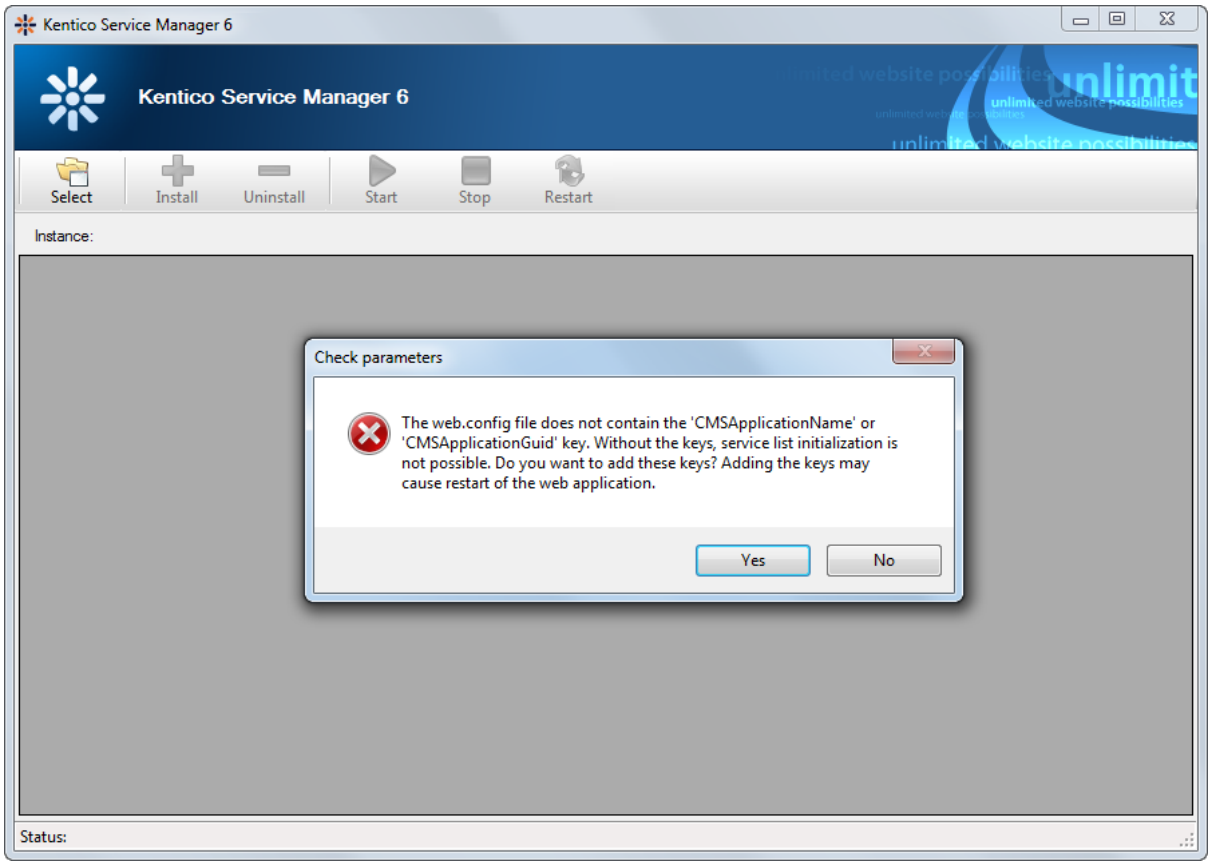


Problems with required web.config keys

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**, no keys will 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.



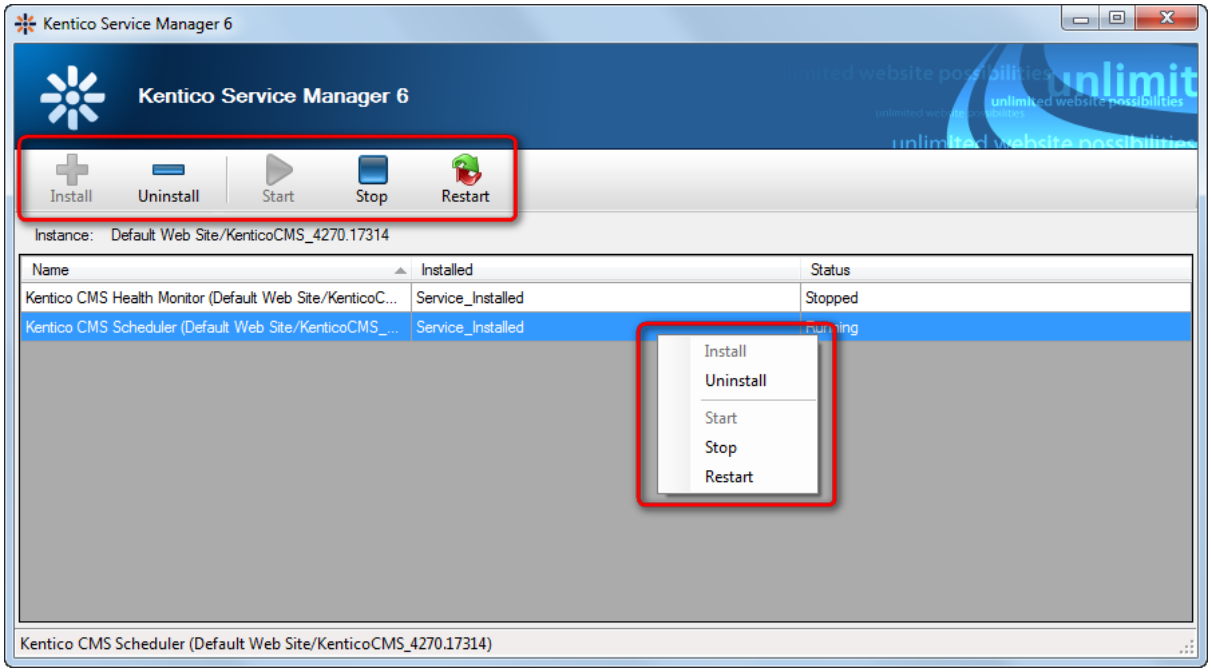
Using the utility

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 for it in the top toolbar. The same actions are also available in a context menu accessible by right-clicking a respective 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.



9.5.3 Services definition XML

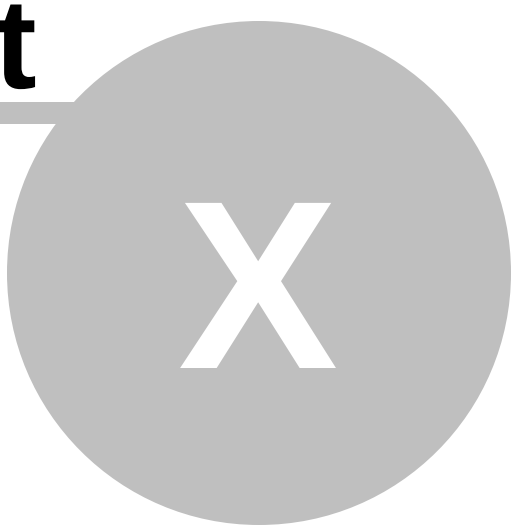
Windows services that can be managed by Kentico Service Manager are defined in *~/App_Data/CMSModules/WinServices/services.xml*. All health monitoring services that should be manageable by Kentico Service Manager and generally all Windows services that should work with the respective Kentico CMS instance (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 dabatase 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 Site management, import and export

10.4.1 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";
    }
```

10.4.2 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.

10.4.3 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);
        }
```

10.5 Custom Providers

10.5.1 Overview

Kentico CMS allows you to develop custom providers and use them instead of the standard ones. This way, you can modify the behaviour of the application or a specific module according to your exact requirements.

To perform this type of customization, you can either add a new assembly to your web project and define the necessary logic there or create the custom classes in the **App_Code** folder and ensure that they are registered correctly. Placing your custom code into the App_Code folder means that it will be compiled dynamically as needed and automatically referenced in all other parts of the application. Please see the Customizing providers from the App_Code folder and Registering custom classes via the web.config topics to learn more.

Examples of provider customizations and additional information can be found by following the links below:

- Custom E-mail Provider - demonstrates the typical customization process of a standard provider. The same approach can be used to customize most Kentico CMS provider classes (anything that inherits from the *CMS.SettingsProvider.AbstractProvider* class).
- Custom Data Provider - describes the specific steps that need to be taken to customize the Data provider.
- Custom Search Provider - contains an example of how the SQL search provider can be modified.
- Custom E-commerce Providers (the link leads to a dedicated chapter in the *Kentico CMS E-commerce guide*)

Customization is also possible for the following Helper classes:

- **CacheHelper** (CMS.GlobalHelper)
- **OutputHelper** (CMS.OutputFilter)
- **MediaHelper** (CMS.GlobalHelper)
- **DirectoryHelper** (CMS.IO)
- **TransformatonHelper** (CMS.Controls)
- **ClassHelper** (CMS.SettingsProvider)

Every class used to customize a provider or helper must inherit from the original class. This means that you can implement your modifications or additions by overriding the members of the given object. To ensure that everything works correctly when creating overrides for existing methods, it is recommended to call the original base method first and then add your own code (whenever possible). You can find information about provider and helper classes in the Kentico CMS API reference, including their inheritance hierarchy and lists of available methods.

You can also use global event handlers to customize the behaviour of the system, such as document modifications, the authentication process, etc.

10.5.2 Customizing providers from the App_Code folder

Custom providers can be defined in the **App_Code** folder, without the need to be compiled into an additional assembly. This process consists of two basic steps: creating a custom class and then registering it so that it is dynamically loaded when required. To see an example of how to properly write the provider class itself, please see the Custom E-mail provider topic in this chapter.

The steps below describe how you can register a custom class in the App_Code folder:

1. Open your project in Visual Studio and expand the **App_Code** folder (or **Old_App_Code** if you installed Kentico CMS as a web application). Then, navigate to the **Samples\Modules** directory and open the **SampleClassLoaderModule.cs** file. You can examine the code of this file to learn how to load a custom class and use it as inspiration for writing your own loaders.

[C#]

```
/// <summary>
/// Sample class loader. The CMSModuleLoader partial class ensures correct
registration.
```

```
/// </summary>
[SampleClassLoaderModuleLoader]
public partial class CMSModuleLoader
{
    /// <summary>
    /// Attribute that ensures the loading of custom classes.
    /// </summary>
    private class SampleClassLoaderModuleLoaderAttribute : CMSLoaderAttribute
    {
        ...
    }
}
```

Loading classes in *App_Code* is done by extending the **CMSModuleLoader** partial class and adding attributes to it. Individual attributes are then defined as private classes that inherit from **CMS.SettingsProvider.CMSLoaderAttribute**.

2. When this type of attribute is processed, its **Init()** method is called, so you can override it to load the classes that you need. The most direct way to register a custom class is to use the API as shown below:

[C#]

```
/// <summary>
/// Initializes the loading of a custom class. Called by the CMSModuleLoader when
the attribute is processed.
/// </summary>
public override void Init()
{
    // Assigns an object of the CustomSiteInfoProvider class as the provider.
    SiteInfoProvider.ProviderObject = new CustomSiteInfoProvider();
}
```

Specifically, this is done by assigning a new object of your class into the **ProviderObject** property of the provider that you wish to customize. The same applies when customizing helper classes, with the difference that the object must be assigned as the value of the **HelperObject** property instead. Remember to add a reference to the namespace that contains the given provider or helper.

3. An alternative approach is to use the **Init()** method's override to assign a handler for the **OnGetCustomClass** event of the **CMS.SettingsProvider.ClassHelper**, which can then be defined to return an object of the appropriate class. For example:

[C#]

```
/// <summary>
/// Initializes the loading of a custom class. Called by the CMSModuleLoader when
the attribute is processed.
/// </summary>
public override void Init()
{
    // Assigns a handler that registers custom provider or helper classes from
App_Code via the cms.extensibility web.config section.
```

```
    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)
    {
        // Passes on an object of the appropriate custom class.
        switch (e.ClassName)
        {
            // Gets an object of the CustomSiteInfoProvider class inheriting from
the SiteInfoProvider.
            case "CustomSiteInfoProvider":
                e.Object = new CustomSiteInfoProvider();
                break;

            // Gets an object of the CustomCacheHelper class inheriting from the
CacheHelper.
            case "CustomCacheHelper":
                e.Object = new CustomCacheHelper();
                break;

            // Gets an object of the CustomEmailProvider class inheriting from the
EmailProvider.
            case "CustomEmailProvider":
                e.Object = new CustomEmailProvider();
                break;

        }
    }
}
```

The handler checks the name of the requested provider or helper and then returns an instance of the corresponding custom class. The **OnGetCustomClass** event is only fired if you specify that a custom class should be used for the given provider or helper via a special extensibility section of the application's **web.config** file. There, the value of the **ClassName** property of the handler's *ClassEventArgs* parameter is also set. For details about how you can assign a custom class in the web.config, please see Registering custom classes via the web.config.

For more examples, you can check the code of the various files in the **~/App_Code/Samples** folder of your web project.

10.5.3 Registering custom classes via the web.config

The web.config of your Kentico CMS application may be used to map providers and helpers to custom classes that are included in your web project, i.e. either in a separate assembly or under the **App_Code** folder. To do this, please follow the steps described below:

1. Edit your **web.config** file and add the following 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 a new extensibility section, where you can register custom providers and helpers.

2. Next, create the actual **<cms.extensibility>** section under the <*configuration*> node and add any necessary providers or helpers into the appropriate sub-sections:

```
...

  <!-- 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>
```

As you can see, each custom class is assigned to a provider or helper through its own <*add*> element 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 the custom class is defined. Set the value to *App_Code* for classes located in this folder (even on web application installations, where the actual name of the folder is *Old_App_Code*).
- **type** - used to specify the name of the custom class. If your custom class is in the App_Code folder, the value of this attribute will be passed in the *ClassName* property of the *ClassEventArgs* parameter,

which is available for the *onGetCustomClass* event handler (as can be seen in step 3 of the Customizing providers from the App_Code folder topic).

10.5.4 Custom E-mail Provider

The custom e-mail provider allows you to use third-party e-mail components for sending e-mails or add custom actions when an e-mail is sent (e.g. logging the sent e-mails for auditing purposes). All e-mails sent by Kentico CMS and its modules will use your custom e-mail provider once it is configured.

Example

The following example demonstrates how a custom e-mail provider may be defined. This sample provider supports all of the default e-mail functionality, but in addition creates an entry in the Kentico CMS Event log every time an e-mail is sent.

1. Open the **web.config** file of your Kentico CMS application and add the following (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 a web.config section used to configure custom providers and helpers.

2. Next, create the actual **cms.extensibility** section and add the e-mail provider as shown below:

```
...

  <!-- Extensibility BEGIN -->
  <cms.extensibility>
    <providers>
      <add name="EmailProvider" assembly="App_Code" type="CustomEmailProvider" />
    </providers>
    <helpers>
    </helpers>
  </cms.extensibility>
  <!-- Extensibility END -->

...
```

```
</configuration>
```

Save the changes made to the web.config file.

3. Now expand the **App_Code** folder (or **Old_App_Code** if you installed your project as a web application), navigate to the **Samples\Modules\SampleClassLoaderModule.cs** file and open it. This file demonstrates how you can load a custom class and ensure that it is used instead of the default e-mail provider. You do not have to modify it for the purposes of this example, but the principles are the same when defining your own custom provider. Objects of the appropriate custom class are loaded by the **ClassHelper_OnGetCustomClass** handler:

[C#]

```
/// <summary>
/// Gets a custom class object based on the given parameters.
/// </summary>
private void ClassHelper_OnGetCustomClass(object sender, ClassEventArgs e)
{
    if (e.Object == null)
    {
        // Passes on an object of the appropriate custom class.
        switch (e.ClassName)
        {
            // Defines the MyTask class implementing ITask and you can create a
scheduled task in App_Code
            case "Custom.MyTask":
                e.Object = new Custom.MyTask();
                break;

            // Defines the MyCustomIndex class implementing ICustomSearchIndex and
you can create a custom smart search index in App_Code
            case "Custom.MyIndex":
                e.Object = new Custom.MyIndex();
                break;

            // Defines the MyCustomSiteInfoProvider class inheriting the
SiteInfoProvider so that you can customize this provider
            case "CustomSiteInfoProvider":
                e.Object = new CustomSiteInfoProvider();
                break;

            // Define the MyCustomCacheHelper class inheriting the CacheHelper so
that you can customize this helper
            case "CustomCacheHelper":
                e.Object = new CustomCacheHelper();
                break;

            // Gets an object of the CustomEmailProvider class inheriting the
            EmailProvider.
            case "CustomEmailProvider":
                e.Object = new CustomEmailProvider();
                break;

        }
```

```
    }
}
```

Notice that the **ClassName** property of the handler's parameter takes its value from the **type** attribute of the *provider* element in the web.config (*CustomEmailProvider* in this example). The example loads an object of the **CustomEmailProvider** class, but you may change the code here to use any class that you require when implementing a custom e-mail provider.

4. Now expand the **Samples\Classes** folder and open the **CustomEmailProvider.cs** class, which contains sample customizations of the e-mail provider. In this class, you can see how you can override the methods used to send out e-mails (the actual code in the file may not be exactly the same as shown below, but you can modify it as necessary).

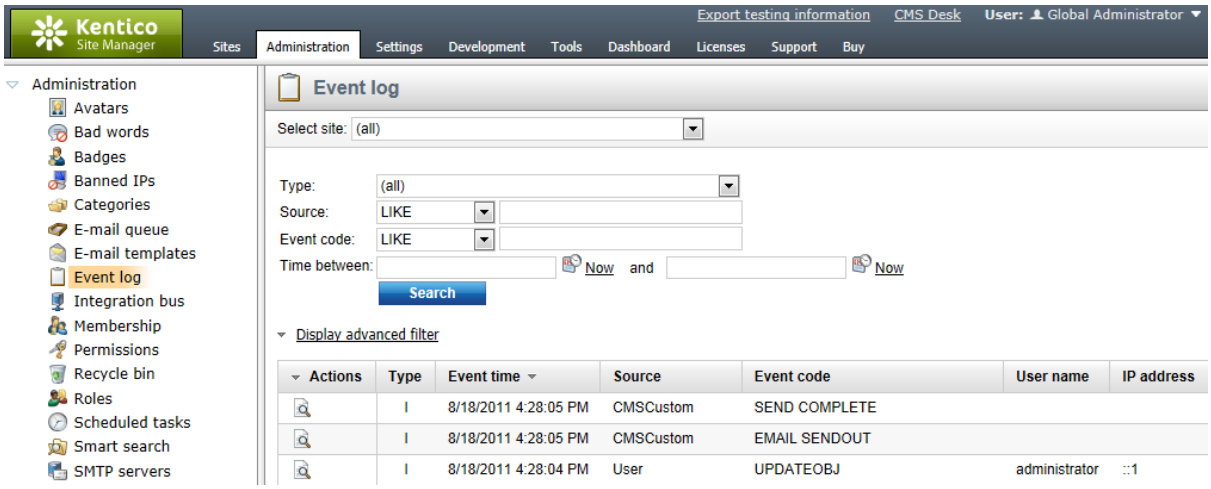
[C#]

```
/// <summary>
/// Sample 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} through {1} was sent
(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} through {1} was dispatched
(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 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);
        }
    }
```

When writing a class implementing a custom e-mail provider, it must always inherit from the **CMS.EmailEngine.EmailProvider** class. This means you can call the base methods as shown above and simply add the necessary functionality (logging information in the event log in this case).

5. To try out the sample custom e-mail provider, send some e-mails using Kentico CMS and check the log at **Site Manager -> Administration -> Event log**. Every successfully sent mail should have its own entry (two if it was sent asynchronously) as shown in the image below:



10.5.5 Custom Data Provider

The Custom Data Provider can be used to implement your own database connector. Customizations for this provider cannot be done in the **App_Code** folder. They must be added through a new assembly and registered via a specific web.config key as described below.

Please note: The Custom Data Provider is NOT intended for accessing non-Microsoft SQL Server database engines. It is only intended for minor modifications of the way the queries are executed against the Microsoft SQL Server.

1. Open the web project in Visual Studio.

2. Copy the **CustomDataProvider** project from *<installation directory>/CodeSamples* to your Solution directory.

3. Add the project **CustomDataProvider** to the solution.

4. Add a reference to the new project to your website project.

5. Add references to the **IDataConnectionLibrary**, **SettingsProvider**, **DirectoryUtilities** a **GlobalHelper** libraries located in the website project to the CustomDataProvider project.

6. Build the solution.

7. Add the following key to the AppSettings section of your **web.config** file:

```
<add key="CMSDataProviderAssembly" value="CMS.CustomDataProvider"/>
```

8. Run the website, it should now use the **CustomDataProvider** library.

9. If everything works correctly, you can modify the code of the Custom Data Provider as needed.

10.5.6 Custom Search Provider

Kentico CMS allows you to write your own SQL search provider that can use a third-party search engine. You can also write a custom search provider when you need to modify or filter search results returned by the standard search engine. Customizations for this provider cannot be done in the **App_Code** folder. They must be added through a new assembly and registered via a specific web.config key as described below.

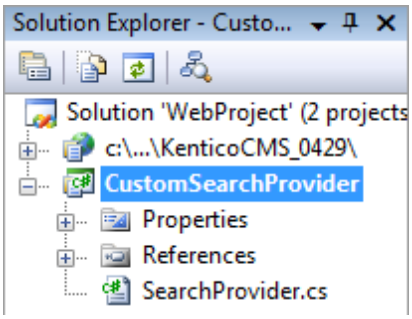
Please note

This provider only affects the SQL Search engine, the Smart Search engine will remain 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 explains how you can limit the standard search path to a subsection of your website:

1. Copy the **CustomSearchProvider** project from Kentico CMS installation (typically *C:\Program Files\KenticoCMS\<version>\CodeSamples\CustomSearchProvider*) to some development folder (not under the web project).

2. Open the **CMS** web project using the **WebProject.sln** file. Click **File -> Add -> Existing Project** and select the **CustomSearchProvider.csproj** file in the folder where you copied the **CustomSearchProvider** project. Your Solution Explorer window will look like this:

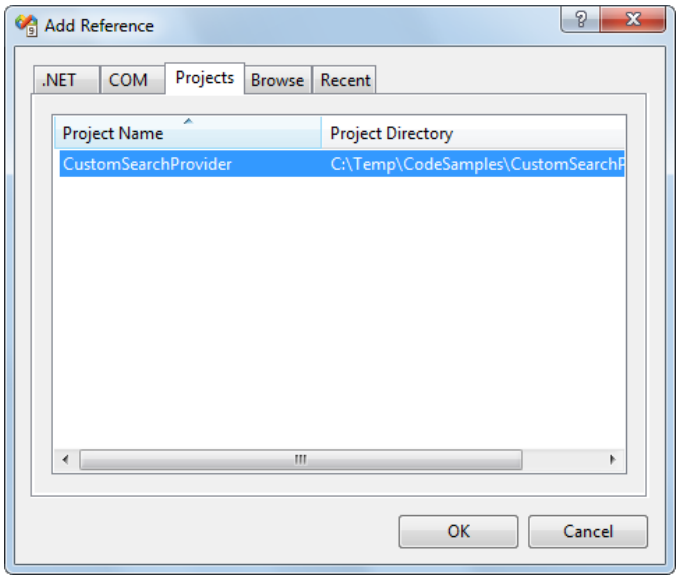


3. Unfold the **References** section of the **CustomSearchProvider** project and delete all invalid references.

4. Right-click the **CustomSearchProvider** 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.ISearchEngine.dll
- CMS.SearchProviderSQL.dll
- CMS.DataEngine.dll
- CMS.SettingsProvider.dll

5. Unfold the **bin** folder in the **CMS web project**. Right-click the **bin** folder and choose **Add Reference**. Choose the **Projects** tab and click **OK** to add the **CustomSearchProvider** project reference:



6. Now you can modify the **SearchProvider.cs** class and place your code into the **Search** method. Modify the code of the method to look like this:

[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 searchNodePath to the /Products section of the website only
    searchNodePath = "/Products/%";

    CMS.SearchProviderSQL.SearchProvider standardSearchProvider = new CMS.
SearchProviderSQL.SearchProvider();

    DataSet ds = standardSearchProvider.Search(siteName, searchNodePath,
cultureCode, searchExpression, searchMode, searchChildNodes, classNames,
filterResultsByReadPermission, searchOnlyPublished, whereCondition, orderBy,
combineWithDefaultCulture);

    return ds;

}
```

7. Add the following value to the **/configuration/appSettings** section of your web.config file:

```
<add key="CMSSearchProviderAssembly" value="CMS.CustomSearchProvider" />
```

8. Click **Build -> Rebuild solution**. Go to the live site and try to search for some text using the SQL Search engine, for example on the **Corporate Sample site** at **Examples -> Web Parts -> Full-Text Search -> SQL Search dialog with results**. You will notice that the search now only returns results from the **/Products** section of the website.

Tip

This is only an example to show how the standard search engine can be modified. The search path can easily be changed by setting the **Path** property of individual search web parts or controls.

10.6 Data layer

10.6.1 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 is currently working only with Microsoft SQL Server, but it’s designed for use with other data engines in 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

10.6.2 Code examples

The following examples show how you can use the CMS.DataEngine library 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;
...

SimpleDataClass userObj = new SimpleDataClass("cms.user");
userObj.SetValue("username", "johns");
userObj.SetValue("fullname", "John Smith");
userObj.Insert();
```

Selecting and updating an existing user

[C#]

```
using CMS.DataEngine;
...

SimpleDataClass userObj = new SimpleDataClass("cms.user", 10);
```

```
string userName = (string) userObj.GetValue("username");
userObj.SetValue("fullname", "John Smith Jr.");
userObj.Update();
```

Deleting a user

[C#]

```
using CMS.DataEngine;
...
SimpleDataClass userObj = new SimpleDataClass("cms.user", 10);
if (! userObj.IsEmpty())
{
   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 it's supported for the chosen entity).

[C#]

```
using CMS.DataEngine;
...
QueryDataParameters parameters = new QueryDataParameters();
parameters.Add("@UserName", "johns");

TypedDataSet ds = ExecuteQuery("cms.user.selectbyname", parameters);
```

10.6.3 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;
                        }
                    }
                }
            }
        }
```

10.7 Global events and their handling

10.7.1 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.

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*, it is an object which contains only properties and is used to store event data. It inherits from the *System.EventArgs* class.
- **<name>Events** - e.g. *DocumentEvents*, it is a static class containing definitions of handlers for various events as static properties.

Please note

If you are accustomed to using custom handler libraries, please note that these global events provide the same functionality in a much easier way. You can refer to the examples of the old handlers to see the difference between the two approaches.

Event handler registration

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.TreeEngine;
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
        }
    }
}
```

Available events

The following list contains available 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.

- **<name>Info.TYPEINFO.Events** - contains events that provide the same set of events as *ObjectEvents*, but specific for a particular object type, e.g. *UserInfo.TYPEINFO.Events*.

- **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.

- **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.

- **WorkflowEvents** – contains events related to specific workflow and versioning actions, replaces the old *CustomWorkflowHandler*.
  - Approve – fired when the document is approved to the next step.
  - Reject – fired when the document is rejected to a previous step
  - Publish – fired when the document is being published.

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 customization of 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.

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 of 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.

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 the After handler to make sure you 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

10.7.2 Event handler libraries

10.7.2.1 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 - TreeNode insert/update/delete 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" />
```

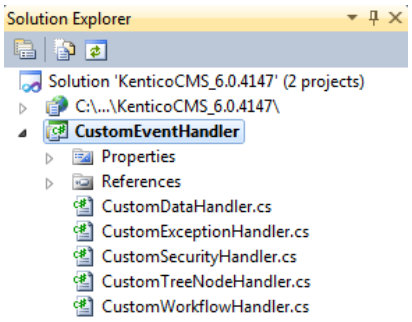
If you wanted to use this library in the following series of examples, it would require you to write the code in Visual Basic instead of C#.

How to write your own 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*).

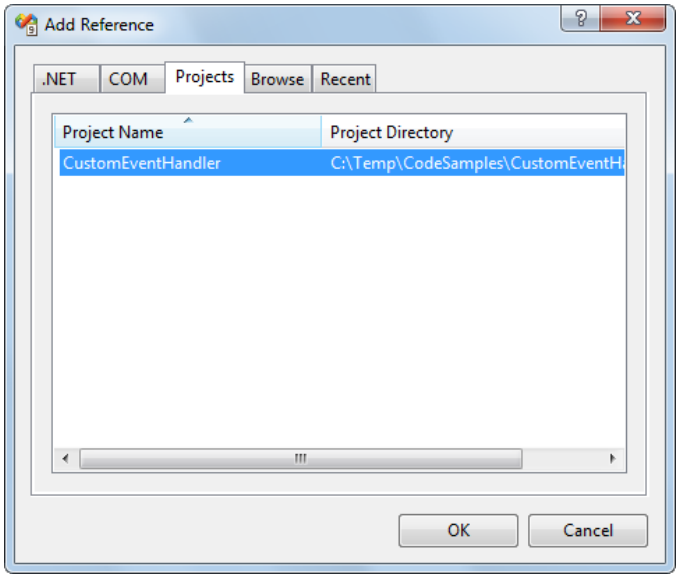
2. Open the **CMS** web project using the **WebProject.sln** file. Click **File -> Add -> Existing Project** and select the **CustomEventHandler.csproj** file located in the folder where you copied the **CustomEventHandler** project. Your Solution Explorer window should look like this:



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. Unfold the **bin** folder of the Kentico CMS web project and delete the standard *CMS.CustomEventHandler.dll* library.

5. Right-click the **bin** folder and choose **Add Reference**. Choose the **Projects** tab and click **OK** to add a reference to the **CustomEventHandler** project:



6. 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.TreeEngine.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.TreeEngine.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 needed to use another library in your custom handlers, please add a reference to it the same way as described in this step.

7. 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.

10.7.2.2 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 & Mebership**.

3. Set the From and Recipients e-mail addresses to your e-mail address.

4. If you don't have refference 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.

10.7.2.3 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.

10.7.2.4 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;

        // Initialize a hash table mapping roles to sites for the user
        Hashtable rolesTable = new Hashtable();

        // Get the code name of the current site
        string siteName = CMSContext.CurrentSite.SiteName;

        // Assign the user to the current site
        usr.SitesRoles[siteName.ToLower()] = rolesTable;
```

```
        // Add new role "external role" to the hash table to assign it to the user
        rolesTable["external role"] = 0;
    }

    // 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.

10.7.2.5 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.TreeEngine;
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.

10.7.2.6 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.TreeEngine;
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;
```

10.8 API examples

10.8.1 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.

10.8.2 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.

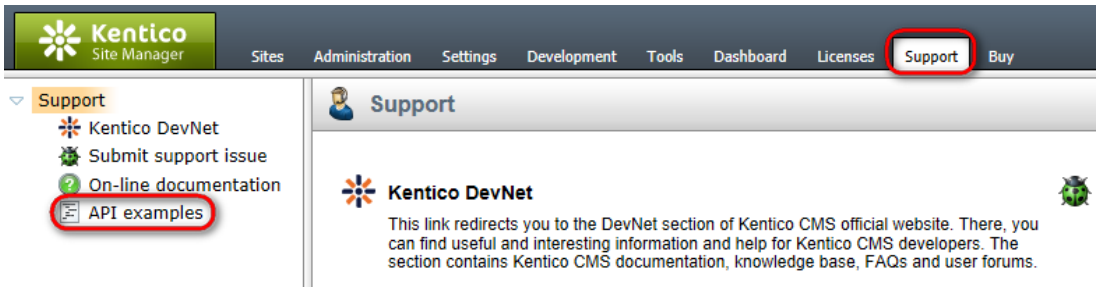
2. If you perform web project installation with the settings described above, all available API examples will be installed. API examples source code will be located in *<web project folder>/CMSAPIExamples* and the interface enabling users to view and run these examples will be located in **Site Manager -> Support -> API examples**.

> **Please note**
>
> You will need to have at least one site installed in order to be able to run API examples.

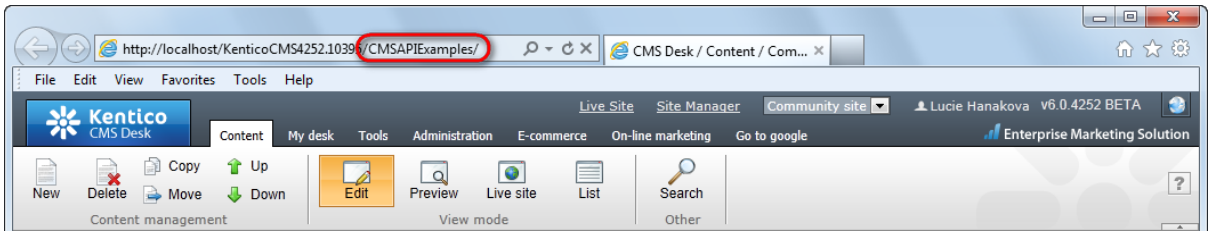
10.8.3 API examples user interface

If you have Kentico CMS API examples installed, you can access them by:

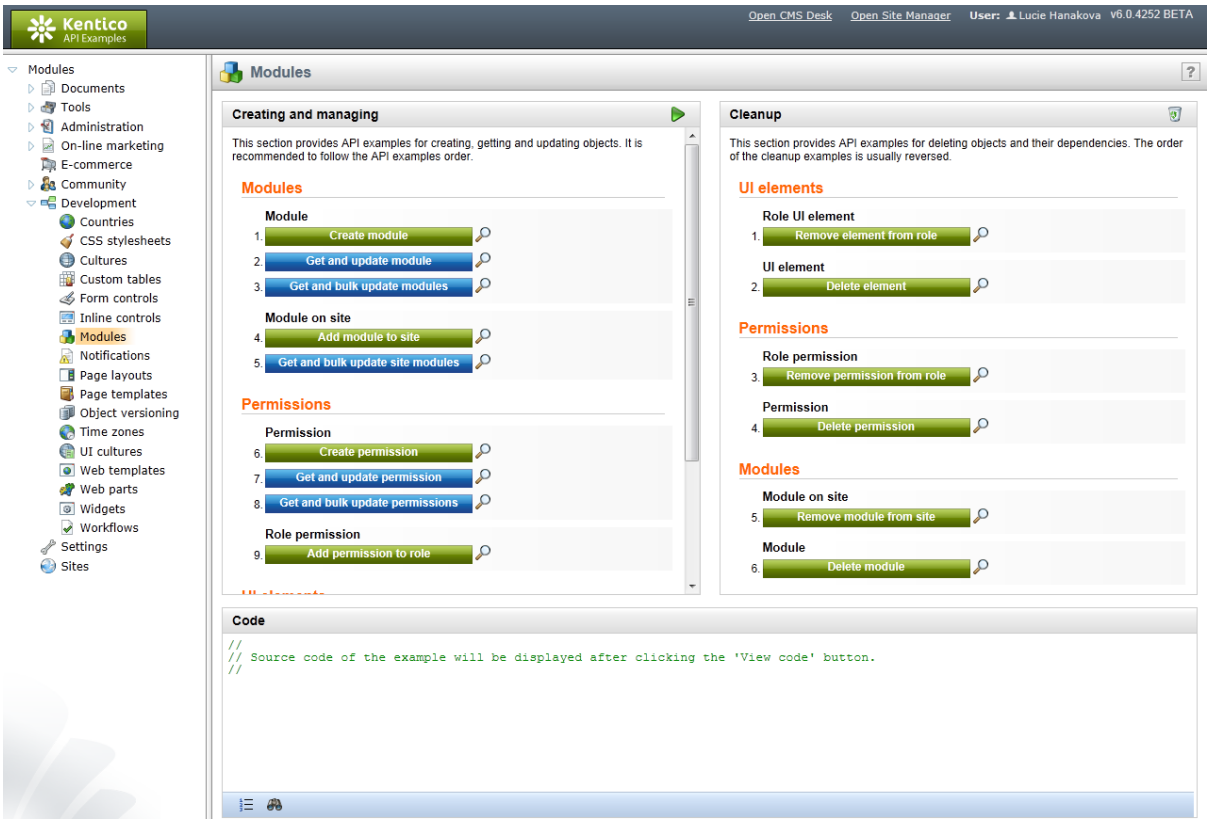
- Going to **Site Manager -> Support** and choosing **API examples** from the **Support** tree.



- 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.

10.8.4 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.

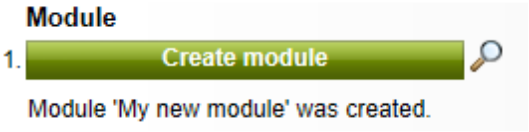
If other general objects are used in API examples, their list can be found in the **Internals and API** section of the respective module.

10.8.5 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.

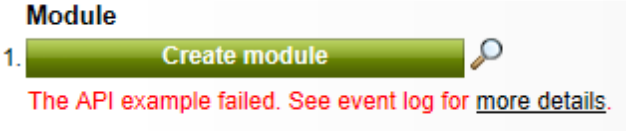
Running an API example may result in:

- **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.

10.8.6 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.

10.9 Using transactions and connections

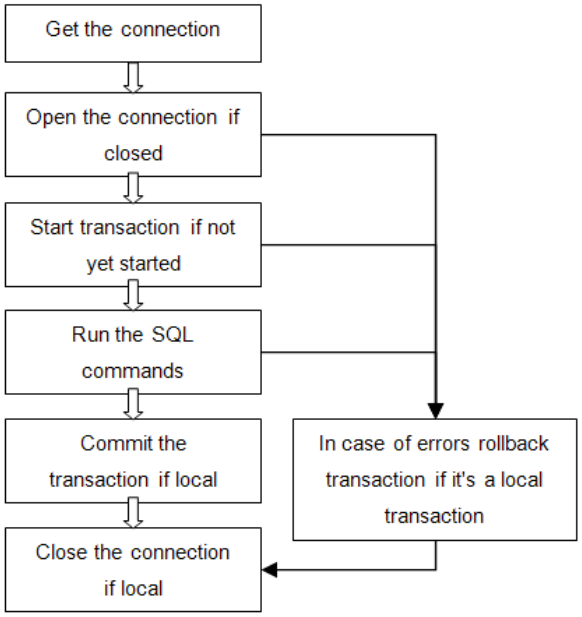
This chapter explains how to work with transactions in Kentico CMS API. Transactions are usually used to ensure database consistency.

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 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 method parameter if you call the method that

uses DB access within transaction.

Example

Please use the following example as a template in all the methods that work with transactions:

[C#]

```
using CMS.DataEngine;
using CMS.TreeEngine;
using CMS.FileManager;
using CMS.WorkflowEngine;

/// <summary>
/// Sample method
/// </summary>
/// <param name="conn">Data connection to use</param>

public void DoSomethingInTransaction()
{

    // 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 **TreeProvider** class instead of connection object. 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.

10.10 Customizing system objects with custom data or objects

In the example below, you will learn how to extend system objects with your own fields or data.

This extending is possible thanks to the **IRelatedData** interface and the **RelatedData** (object) property. If an object implements the **IRelatedData** interface, you can use its **RelatedData** (object) property to connect just about any object to it. The connected object is then shipped as a part of the system object.

On top of it, you can also make the related object behave as a native part of the system object and retrieve its properties as if they were properties of the system object. If the connected object implements the **IDataContainer** interface, it gets automatically connected to the system object's **GetValue** and **SetValue** methods.

Example

Let's presume that you want to customize the **SiteInfo** object with two additional properties and one method:

- **SiteOwner** - String, just the name of the site owner
- **SiteValidUntil** - DateTime until which the site is valid
- bool **IsValid**() - Returns true if the site is valid at the current time

1. First, you will need to create a custom class in the solution, containing the properties and the method as you can see in the code example below:

SiteRegistration.cs

[C#]

```
using System;
using CMS.SettingsProvider;

/// <summary>
/// Summary description for SiteRegistration
/// </summary>
public class SiteRegistration : IDataContainer
{
    #region "Variables"

    private string mSiteOwner = null;
    private DateTime mSiteValidUntil = DateTime.MinValue;

    #endregion


    #region "Properties"

    /// <summary>
    /// Gets or sets 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 the column names
    /// </summary>
    public List<string> ColumnNames
    {
        get
        {
            return new string[] { "SiteOwner", "SiteValidUntil" };
        }
    }


    /// <summary>
    /// Returns true
    /// </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 object value
    /// </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 field value
    /// </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>
    /// Tries to get the value from the object
    /// </summary>
    /// <param name="columnName">Column name</param>
    /// <param name="value">Returns the 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;
        }
    }

    #endregion


    #region "Methods"

    /// <summary>
    /// Returns true if the site is valid
    /// </summary>
    public bool IsValid()
    {
        return this.SiteValidUntil > DateTime.Now;
    }

    #endregion
}
```

2. Now that you have the class ready, you will want to bind it to the *SiteInfo* object. The binding will be dynamic, which means that the data will be loaded when it is requested. We will use the **OnLoadRelatedData** event which you can use for specific object types, including our *SiteInfo* objects. The event is part of the type information object referenced from the object info class.

What you need to do is 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\SystemObjectCustomizationModule.cs*, which will extend the *CMSModuleLoader* partial class. The class should contain the method ensuring the binding and its registration into the event in the *Init()* method, which is executed on each application start.

[C#]

```
using System;

using CMS.SettingsProvider;
using CMS.SiteProvider;

[SystemObjectCustomizationModuleLoader]
public partial class CMSModuleLoader
{
    /// <summary>
    /// Module registration
    /// </summary>
    private class SystemObjectCustomizationModuleLoaderAttribute :
CMSLoaderAttribute
```

```
    {
        /// <summary>
        /// Initializes the module
        /// </summary>
        public override void Init()
        {
            CMS.SiteProvider.SiteInfo.TYPEINFO.OnLoadRelatedData += new TypeInfo.
ObjectLoadRelatedDataEventHandler(SiteInfo_OnLoadRelatedData);
        }

        static object SiteInfo_OnLoadRelatedData(CMS.SettingsProvider.BaseInfo
infoObj)
        {
            // Get the related data from your external storage (make sure it
implements IDataContainer. Here are just some data created on-the-fly.
            SiteRegistration sr = new SiteRegistration();
            sr.SiteOwner = "Martin Hejtmanek";
            sr.SiteValidUntil = DateTime.Now.AddDays(1);

            return sr;
        }
    }
}
```

3. Now try to paste the following code to the layout of some page on your site. Notice the bold parts of the code. As you can see, data stored in the custom properties are retrieved using the GetValue method as if they were a native part of the SiteInfo object.

[C#]

```
<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>
```

As a result, you should see the following text on the page:

Site 'Corporate Site' is valid until 9/15/2009 1:09:50 PM and owned by Martin Hejtmanek.

4. It is also possible to get the values of the custom properties using macros. Try adding the **Text -> Static text** web part to your page and paste the following text to its **Text** property. The same text as in the previous example should be displayed on the live site.

```
Site '{%CMSContext.CurrentSite.SiteDisplayName%}' valid until {%CMSContext.
CurrentSite.SiteValidUntil%} and owned by {%CMSContext.CurrentSite.SiteOwner%}
```

10.11 Using API and CMS Controls outside the CMS project

You can use Kentico CMS API and Kentico CMS Controls also outside the standard CMS website project. This chapter explains how you can configure your ASP.NET application so that it can use Kentico CMS API and Kentico CMS Controls.

Start Visual Studio and create a new ASP.NET application or open your existing ASP.NET web project.

Kentico CMS Initialization

Please note: you may need to initialize the application to be able to use all the features of Kentico CMS API. To do that, call the method **CMSContext.Init()** before any other calls to the API. You can do that at the time your project starts or anytime later, the method handles all necessary initializations of the environment.

Configuring the web.config file

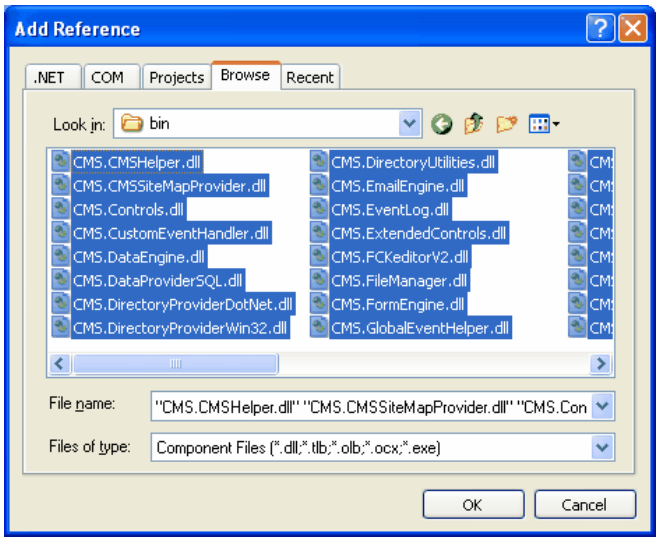
Add the connection string for the Kentico CMS database with name CMSConnectionString into the configuration/connectionStrings section of the web.config. The section will look like this:

```
<configuration>
  <connectionStrings>
    <add name="CMSConnectionString" connectionString="Persist Security
              Info=False;database=KenticoCMS;server=myserver;user id=sa;
password=mypassword;Current
              Language=English;Connection Timeout=120;" />
```

It's recommended that you copy the exact connection string line from the web.config file of the CMS web project.

Adding a reference to Kentico CMS API libraries

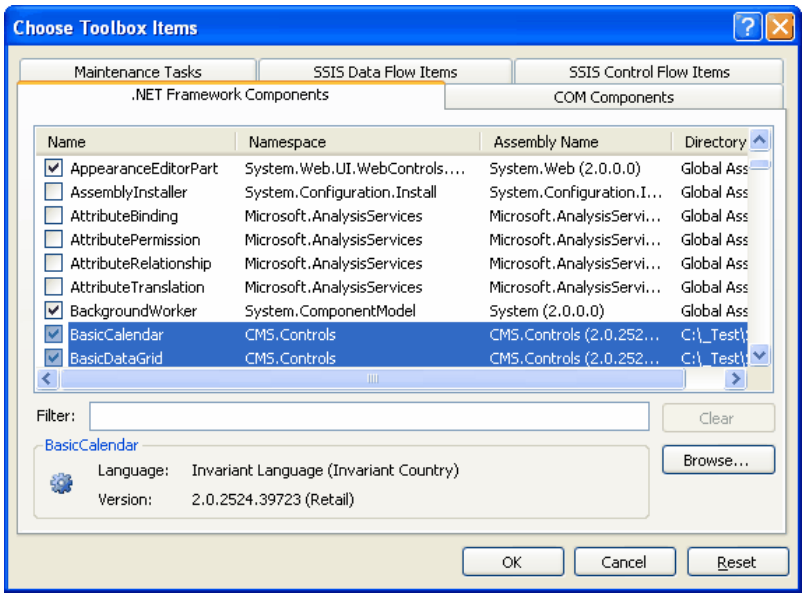
Now we will add a reference to Kentico CMS libraries. Right-click your web project in the **Solution Explorer** and choose **Add reference...** Choose the **Browse** tab and locate the CMS web project's bin folder. Choose all libraries and click OK:



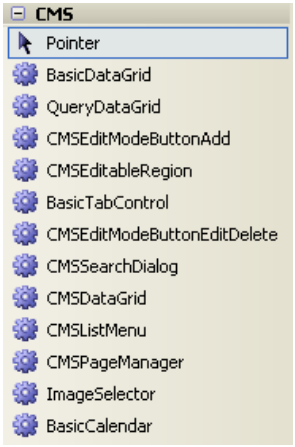
Adding Kentico CMS Controls to your toolbox

Now we will add Kentico CMS Controls to your toolbox so that you can easily drag-and-drop the controls on the web forms. Right-click the toolbox and choose **Add tab**. Use the name **CMS** and press Enter.

Right-click the new tab and click **Choose Items...** Click **Browse...** and locate the **CMS.Controls.dll** library in the **bin** folder of your application. Click **Open** and click **OK** on the **Choose Toolbox Items** dialog.



The controls are added to your tab:



Configuring your project for transformations (virtual path provider)

The transformations used by Kentico CMS Controls are retrieved using a virtual path provider. You need to add the following line to the **Application_Start** method in the **global.asax.cs/vb** file. If you installed the project as a web application, the file should be located in project root. If your project was installed as a web site, the file is located in the *<project_folder>\App_Code* folder.

[C#]

```
CMS.VirtualPathHelper.VirtualPathHelper.RegisterVirtualPathProvider();
```

[VB.NET]

```
CMS.VirtualPathHelper.VirtualPathHelper.RegisterVirtualPathProvider()
```

If you cannot use the virtual path provider (e.g. in a medium trust environment), you may need to save the virtual objects to disk using the **Site Manager -> Administration -> System -> Deployment -> Save all virtual objects to disk** button and copy the folder **~/CMSTransformations** to the root of your own web project.

Using Kentico CMS Controls to display content from Kentico CMS database

Now that we have configured the web project for Kentico CMS, we will create a testing page that will display news items from Kentico CMS.

Create a new web form (ASPX page) in your custom project using Visual Studio. Drag and drop the CMSRepeater control on your page and set the following properties:

- ClassNames: cms.news
- CultureCode: en-us (or other language version)
- SiteName: CorporateSite (or other site code name)
- TransformationName: cms.news.preview
- SelectedItemTransformationName: cms.news.default

Run the project and navigate to the newly created page. You should see a page like this:

However, the links may not work at this moment since they are using the Kentico CMS web friendly URLs by default. So we need to modify the transformation cms.news.preview so that it points the user to same page, but with URL parameter **aliasPath** that will contain the page aliasPath. You can alternatively use your own URL parameters and set the Path property appropriately.

Go to **Kentico CMS Site Manager -> Development -> Document types -> edit News -> Transformations -> edit preview**. Change the following line of the transformation code:

```
<b><a href="<%# GetDocumentUrl() %>">
```

Like this:

```
<b><a href="?aliasPath=<%# Eval("NodeAliasPath") %>">
```

Save the changes. Since the transformations are cached, you need to restart (rebuild) your custom web application now so that the change is applied to your website.

Now you need to add a short code to the page code behind:

[C#]

```
protected void Page_PreInit(object sender, EventArgs e)
{
    if (Request.QueryString["aliaspath"] != null)
    {
        CMSRepeater1.Path = Request.QueryString["aliaspath"];
    }
    else
    {
        CMSRepeater1.Path = "/%";
    }
}
```

[VB.NET]

```
Protected Sub Page_PreInit(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Me.PreInit
    If Not Request.QueryString("aliaspath") Is Nothing Then
        CMSRepeater1.Path = Request.QueryString("aliaspath")
    Else
```

```
        CMSRepeater1.Path = "/%"
    End If
End Sub
```

Setting CMS control properties

Kentico CMS Controls load the content at early stage of the page life cycle. It means that if you want to set the properties programmatically, you need to set them in the Page_PreInit event.

If you need to set them for some reason later in the page life cycle, you need to call the `CMSRepeater1.ReloadData(true)` method so that the data is reloaded and the changes are applied.

Using Kentico CMS API to retrieve content from Kentico CMS database

Kentico CMS API can be used to script any action in Kentico CMS, including content retrieval and modification. The following example explains how you can retrieve documents from Kentico CMS database as a DataSet and display them using standard ASP.NET repeater control (instead of using the CMSRepeater control).

Add a new web form (ASPX page) to your web project. Drag and drop the Repeater control on the web form. Switch to the **Source** mode of the page and add the Following item template code inside the <asp:Repeater> control element:

[C#]

```
<ItemTemplate>
  <b><a href="?aliasPath=<%# Eval("NodeAliasPath") %>">
  <%# Eval("NewsTitle") %></a></b> (<%# ( (DateTime) Eval("NewsReleaseDate")).
ToString("d") %>)<br />
  <i><%# Eval("NewsSummary") %></i>
  <br />
</ItemTemplate>
```

[VB.NET]

```
<ItemTemplate>
  <b><a href="?aliasPath=<%# Eval("NodeAliasPath") %>">
  <%# Eval("NewsTitle") %></a></b> (<%# ( (DateTime) Eval("NewsReleaseDate")).
ToString("d") %>)<br />
  <i><%# Eval("NewsSummary") %></i>
  <br />
</ItemTemplate>
```

Switch to the code behind and add the following code to the beginning of the page:

[C#]

```
using CMS.TreeEngine;
using System.Data;
```

[VB.NET]

```
Imports CMS.TreeEngine
Imports System.Data
```

Add the following code inside the Page_Load method of the page:

[C#]

```
TreeProvider tp = new TreeProvider();
DataSet ds = tp.SelectNodes("CorporateSite", "/news/%", "en-us", true, "cms.news",
" NewsReleaseDate <= GetDate() ", " NewsReleaseDate DESC ", -1, true);
Repeater1.DataSource = ds;
Repeater1.DataBind();
```

[VB.NET]

```
Dim tp As New TreeProvider
Dim ds As DataSet = tp.SelectNodes("CorporateSite", "/news/%", "en-us", True,
"cms.news", " NewsReleaseDate <= GetDate() ", " NewsReleaseDate DESC ", -1, True)
Repeater1.DataSource = ds
Repeater1.DataBind()
```

Please note

The code above works fine for just retrieving documents; if you want to perform some other operations with the documents (edit, delete, ...), you should initialize the tree provider with UserInfo, as shown below:

[C#]

```
UserInfo ui = UserInfoProvider.GetUserInfo("administrator");
TreeProvider tp = new TreeProvider(ui);
```

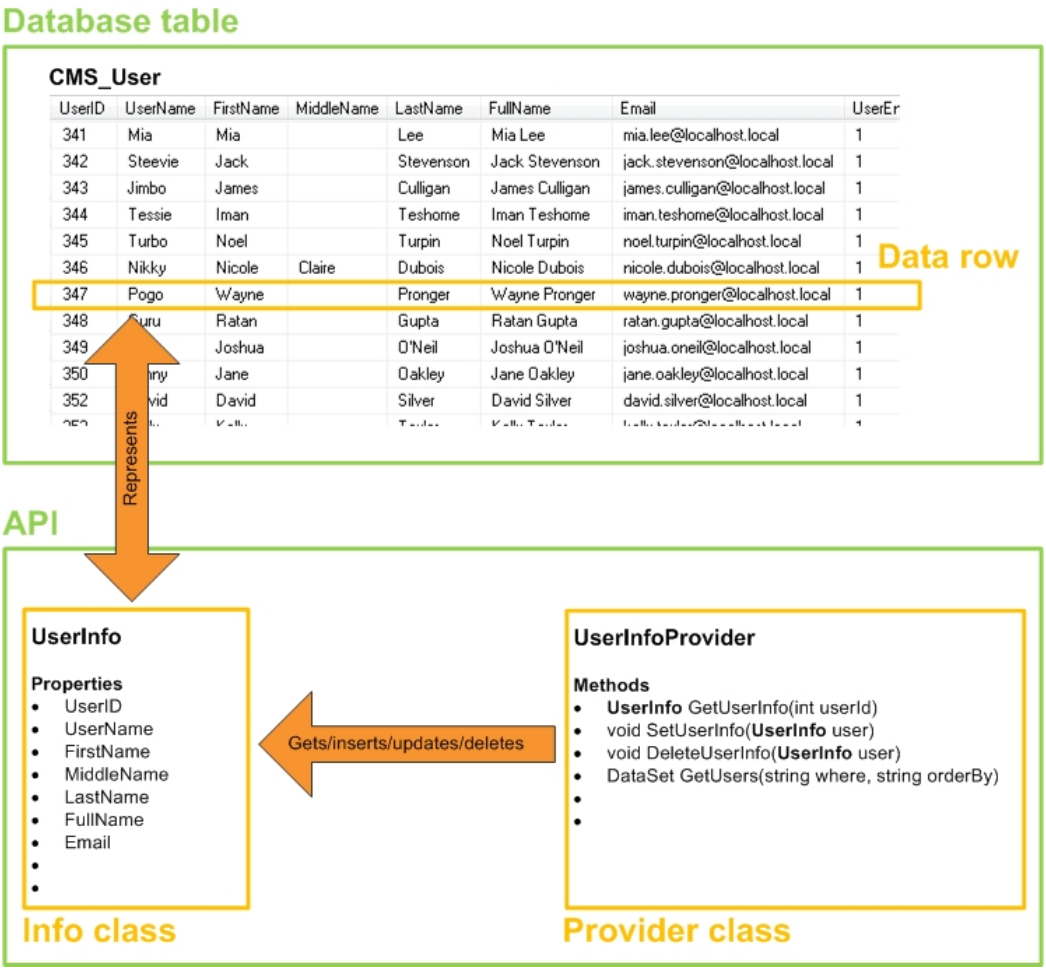
Run the website and navigate to your new page. You should see a page like this:

Your second news (11/6/2006)
Here comes your news summary.
Your first news (11/5/2006)
Here comes your news summary.

As you can see, you can use Kentico CMS API to retrieve content as a DataSet and display it with your custom .NET code.

10.12 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 bellow.



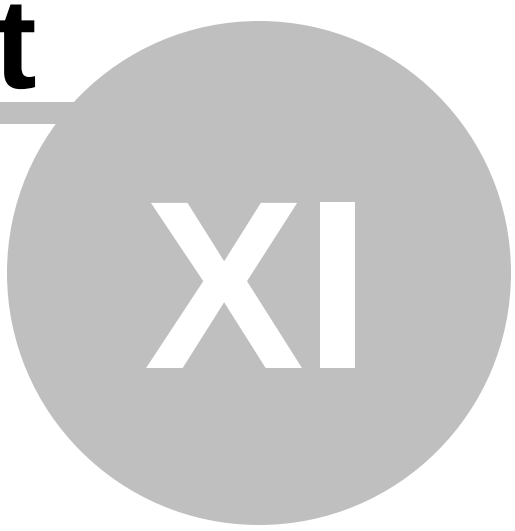
Info class

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



Appendix A - Path expressions

11 Appendix A - Path expressions

Many web parts and controls use a **Path** property that allows you to specify which documents should be displayed. This is the **AliasPath** property of the document. You can use either an exact path or you can use special characters for specifying multiple selection or relative paths:

Leaving the Path value empty

In many cases, **you can leave the Path value empty**. In this case, the Path value is set to the alias path of the currently displayed document.

In case of list controls/web parts, such as CMSRepeater/Repeater or CMSDataGrid/Grid, the path is set to *<current alias path>/%* if the current document is not of the same type as the required document in the **ClassNames** (document types) property. Otherwise, the path is set to the current alias path, which leads to automatic selection of the current document.

Using wildcard characters % and _

You can use **% as a wildcard character for any number of characters**, which allows you to select all documents under specified site section.

Examples:

/ - only root
/% - all documents

/Products - only the Products document.
/Products/% - all documents under the Products document.

You can also use **_ as a wildcard character for a single character**.

Examples:

/Product_ - selects documents /ProductA, /Product1, etc.

Using formatting string to get parts of the path

You can also use special expressions that **extract parts of the current path**, such as this.

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

Using relative paths

You can use relative paths expressions to specify **sub-documents or parent documents**:

Examples:

./Product - document Product under the current path

../Product - document Product under the parent document of the current path
. - current path
.. - 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).	
CMSUseCustomHandlers	Indicates if custom handlers should be executed to process system events. See the Global events and their handling	`<add key="CMSUseCustomHandlers" value="true" />`

	chapter for more details. The default value is *false*.	
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 is displayed in web part properties dialog in CMS Desk. This parameter can be used for backward compatibility purposes. Otherwise, using the **Code** tab is now obsolete. The default value is *false*.	`<add key="CMSShowWebPartCodeTab" value="true" />`
CMSShowWebPartBindingTab	Indicates if the Binding tab is displayed in web part properties dialog in CMS Desk. This parameter can be used for backward compatibility purposes. Otherwise, using the Binding tab is now obsolete. The dafault 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 of each page. The default value is *false*.	`<add key="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 possible length of a document's node alias. The default value is *50* characters and larger names are trimmed. This key can be used to increase the allowed length, which can be useful if you need to enter very long aliases. If you use the key, you also need to set the same size for the **NodeAlias** column in the **CMS_Tree** database table. However, keep in mind that the maximum	`<add key="CMSMaxNodeAliasLength" value="120" />`

	allowed length for the **Node alias path** is 450 characters, so documents deep in the content tree will have their alias path trimmed anyway.	
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*	`<add key="CMSAuthenticationType" value="windows" />`

	• **windows**: *IsFormsAuthentication* always returns *false*, *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 instance. Used by Kentico CMS Windows services to identify the Kentico CMS instance.	`<add 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="CMSAlwaysCacheResource`

	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**.	s" value="false"/>
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 displaying. The default value is *false*.	<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}$"/>
CMS55Compatibility	Ensures compatibility with CMS version 5.5 (includes the CMS.Compatibility namespace in transformations). The default value is *false*.	<add key ="CMS55Compatibility" value="true"/>
CMSMacrosCaseSensitiveComparison	If true, string comparisons are case sensitive (if not overriden by *(casesensitive)* macor parameter).	<add key ="CMSMacrosCaseSensitiveComparison" value="true"

	The default value is *false*.	/>
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. The default value is "".	<add key ="CMSVideoExtensions" 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	<add key="CMSURLPort" value="80" />

	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.	
CMSUseServerSideLiveIDAuthentication	This key is only available if **hotfix 6.0.8** or newer is applied to your installation of Kentico CMS. Determines which authentication mode should be used during Windows Live ID authentication. Each mode generates a different authentication token for the same 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" />`
CMSUseCurrentWildcardValueAsDefaultValue	If enabled, the current value of wildcards in document URLs will also be transferred to all other URLs on the page that contain the same wildcard (e.g. in navigation links). For example, imagine a scenario with two documents that use the same 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 not transferred to wildcards that already have a default value specified directly in the URL path.	`<add key="CMSUseCurrentWildcardValueAsDefaultValue" value="true" />`

	The key is *false* by default.	

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. See this topic for more details. The default assembly is *CMS.VirtualPathProvider*.	`<add key="CMSVirtualPathProviderAssembly" value="CMS.CustomVirtualPathProvider"/>`
CMSCustomEcommerceProviderAssembly	Specifies the custom e-commerce provider assembly. See this topic for more details.	`<add key="CMSCustomEcommerceProviderAssembly" value="CMS.CustomECommerceProvider"`

	The default assembly is *CMS.EcommerceProvider*.	/>

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 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*.	<add 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-	<add key="CMSUserValidationRegEx" value="([A-Za-z0-9-]+)" />

	9_\\-\\.\\\\@]+$".	

Special settings for forbidden character replacement in URLs

You can use the following keys to configure how forbidden URL characters are replaced:

Key	Description	Sample Value
CMSForbiddenURLValues	The characters entered as the value of this key are forbidden in URLs (document 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.	`<add key="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.	`<add key="CMSUseLimitReplacementsForUrlPath" value="false" />`

	The default value is *true*.	

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 through SSL (X.509). If false, only certificates generated by a certification authority will be accepted. The default value is *false*.	`<add key="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.	`<add key="CMSStagingServerName"`

	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).	`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" />`
CMSLineCountersThreshold	The line counter in the editor displays the number of the line where the cursor is currently positioned and the total number of lines in the displayed code. If the code contains more lines than the value specified for this key, the line counter will be disabled. This can be used to optimize editing performance, since the line counter may	`<add key="CMSLineCountersThreshold" value="100" />`

	slow down the response time of the editor if there is a very large amount of displayed lines. The default value is *500*.	

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. The default value is *false*.	`<add key="CMSUseCurrentSKUData" value="true"/>`
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.	`<add key="CMSLogEvents" value="true"/>`

	The default value is *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** option is enabled, details about changes of values in document fields are included in the respective log records. The default value is *false*.	`<add key="CMSLogDocumentFieldChanges" value="true"/>`
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.	`<add key="CMSExportExcludedFiles" value="test*;cms*" />`

	.scc files are excluded by default, even without this key added. More info here.	
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	`<add key="CMSListingShowDirectPage`

	entering a number will be included in pagers of listings in the UI with a large enough number of items. If disabled, the 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*.	`Control" 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 *lang* query string parameter so that you get *Home?sprache=de-de* instead of the default	`<add key="CMSLanguageParameterName" value="sprache" />`

	Home?lang=de-de. 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" />`
CMSLogActivityImmediatelyToDB	If *false*, activities are logged to temporary files on the server disk, which are then 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.	`<add key="CMSLogActivityImmediatelyToDB" 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* format, which can be configured in **Site Manager -> Settings -> Membership & Security -> Passwords -> Password format**.	`<add key="CMSPasswordSalt" value="SaltText" />`
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	`<add key="CMSSearchIndexPath"`

	physically stored.	value="App_Data\MyCustomIndexes" />
CMSSearchOnlyWhenContentPresent	Determines if smart searches should be performed only if a standard keyword is specified in the search field. The default value is *true*. Setting this key to *false* allows searches using expressions composed only of special Lucene query syntax, such as for examples direct field searches.	<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 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.	<add key="CMSCreateTemplateSearchTasks" value="false" />
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 you Event log.	<add key="CMSSmartSearchLockPollInterval" value="500" />

	Lock obtain timed out: CMS. SiteProvider.SearchLock	
CMSSmartSearchLockTimeout	Defines the timeout period (in milliseconds) during which the smart search indexer should try to acquire a lock on the index file. The default value of this key is 1000 milliseconds. You should increase the value if the following exception gets logged into you Event log. *Lock obtain timed out: CMS. SiteProvider.SearchLock*	`<add key="CMSSmartSearchLockTimeout" value="1000" />`

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 user interface cultures

By adding the following keys to your web.config, you can configure UI cultures:

Key	Description	Sample Value
CMSDefaultSpellCheckerCultur	Specifies the default culture of the built-in	`<add key=""`

e	spell-checker. This culture is used when the dictionary for the currently selected content culture is not found. The default value is *en-us*.	`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). The default value is *false*.	`<add key="CMSWebFarmEnabled" 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	`<add key="CMSWebFarmSynchronizeFil`

	enables synchronization of: • Attachments • Meta files • Media files • Form files • Avatars • Forum attachments The default value is *true*.	es" 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="CMSWebFarmSynchronizeForumAttachments" 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" />`

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	`<add key="CMSAzureTableEndPoint"`

	storage account. If you wish to use the default endpoint, remove the setting completely from the appropriate files.	`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.	`<add key="CMSAzureCachePath" value="C:\AzureCache" />`

	Do not use this key if the entire application is deployed as a Windows Azure hosted service.	

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

	enabled. The default value is *false*.	

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 -