Kentico CMS Controls 4.1
Table of Contents

Part I Kentico CMS Controls

1 Overview ........................................................................................................................................... 5
2 Path specification in web parts and controls.................................................................................. 5
3 Caching ............................................................................................................................................. 6
4 Displaying related documents......................................................................................................... 8
5 Dynamic insertion of parameters in data web parts........................................................................ 11
6 CMS Controls .................................................................................................................................. 11

Controls overview ................................................................. 11
Configuring your project for Kentico CMS Controls ................................................................. 12
Using macro expressions in menu items ......................................................................................... 14
Using the CSSPrefix property for design of sub-menus .............................................................. 15
CMSBaseProperties - common properties .................................................................................... 16
CMSControlProperties - common properties .............................................................................. 17
CMSDataProperties - common properties .................................................................................... 18
CMSMenuProperties - common properties .................................................................................. 18
CMSQueryProperties - common properties .................................................................................. 19
BasicCalendar ................................................................................................................................. 19
BasicDataGrid .................................................................................................................................... 21
BasicDataList .................................................................................................................................... 23
BasicMultiColumnTable .................................................................................................................. 27
BasicRepeater .................................................................................................................................... 30
BasicTabControl ............................................................................................................................... 33
CMSBreadCrumbs ............................................................................................................................. 36
CMSCalendar ...................................................................................................................................... 37
CMSDataGrid ...................................................................................................................................... 39
CMSDataList ...................................................................................................................................... 40
CMSDocumentComparison .............................................................................................................. 42
CMSDocumentValue ....................................................................................................................... 44
CMSEditModeButtonAdd .................................................................................................................. 45
CMSEditModeButtonEditDelete ....................................................................................................... 46
CMSEditableImage .......................................................................................................................... 47
CMSEditableRegion ........................................................................................................................ 47
CMLstMenu ......................................................................................................................................... 49
CMSMenu ........................................................................................................................................... 54
CMSPageManager ............................................................................................................................ 58
CMSRepeater ...................................................................................................................................... 60
CMSSearchDialog, CMSSearchResults .......................................................................................... 63
CMSSiteMap ....................................................................................................................................... 66
CMSTabControl ................................................................................................................................. 69
CMSTreeMenu ..................................................................................................................................... 71
CMSTreeNode ....................................................................................................................................... 74
CMSViewer .......................................................................................................................................... 74
DataPager .......................................................................................................................................... 76
QueryDataGrid ..................................................................................................................................... 77
QueryDataList ..................................................................................................................................... 78
QueryRepeater .......................................................... 79
TemplateDataPager ..................................................... 80
Part I
1 Kentico CMS Controls

1.1 Overview

Kentico CMS Controls are standard ASP.NET 2.0 server controls that can be used in Visual Studio 2005 or 2008. You can place them on your custom webparts or on page templates and pages that do not use the portal engine. Some of them can also be used outside Kentico CMS.

1.2 Path specification in web parts and controls

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/webparts, such as CMSRepater/Document repeater or CMSDataGrid/Document datalist, 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

1.3 Caching

What is Caching

Caching allows you to minimize the number of performed database queries. The server can store the data in memory and next time a user requests the content, the server returns content from memory instead of performing a resource-intensive database query. Caching can improve the performance of your Web site typically 10- to 100-times depending on your application.

The content expires after specified time span and must be retrieved from the database again. Each cached item has its name and the cache memory is common for all pages in your Web application.

Caching Support in Kentico CMS

You can manage the caching either by yourself in your code (please see the .NET Framework SDK documentation for more details) or you can leverage caching features of the following Kentico CMS Controls that are also used in CMS web parts:

- CMSBreadCrumbs
- CMSDataGrid
- CMSDataList
- CMSMenu
- CMSRepeater
- CMSSiteMap
- CMSTabControl
- CMSTreeMenu
- CMSViewer
- QueryDataGrid
- QueryRepeater
- QueryDataList
All of these controls offer the following properties, that allow you to set up caching:

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CacheItemName</td>
<td>Name of the cache item the control will use.</td>
<td>“products_” &amp; request.querystring(“categoryid”)</td>
</tr>
<tr>
<td>CacheMinutes</td>
<td>Number of minutes the retrieved content is cached for.</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Zero indicates that the content will not be cached.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-1 indicates the site-level settings should be used.</td>
<td></td>
</tr>
</tbody>
</table>

**Cache Expiration Time**

By setting the CacheMinutes property to a value higher than zero, the control starts to cache its source data. You can configure caching for all Kentico CMS content using the Cache content parameter in Site Manager -> Settings -> Web site section. If you set any particular value to the CacheMinutes property of control/web part, it overrides the global settings. If you leave the value empty or set to -1 (minus one), the global settings apply.

The caching mechanism of Kentico CMS Controls uses absolute expiration time instead of sliding expiration. It means that the cache item expires after specified period of time regardless if it was requested or not. It ensures that content is updated from the database in a regular interval.

**Overriding the site-level caching settings**

If you need to cache most of the content on your web site, but still want to have a single control/web part that doesn't use cache, you can configure caching as described in the previous paragraph and set value 0 (zero) to the CacheMinutes property of the particular control. It will override the site-level settings and disable caching for the single control/web part.

**Cache Item Name**

It's important to understand the CacheItemName property: Since the cache is common for all pages in your application, the cache item name should be unique not only for all pages, but also inside one page (for case you use several Kentico CMS Controls with caching on one page).

When you leave the CacheItemName property empty, the control automatically generates a name in this form:

`URL including parameters/control ID`

If the content displayed on the page depends on some parameter, such as URL parameter or role of the current user, you need to adjust the CacheItemName value accordingly.
Example:
Your page products.aspx displays products according to the category that is passed through the URL parameter "category". You will need to use a code like this to ensure that the content will be cached "per category":

[C#]

```csharp
CMSDataGrid.CacheItemName = "products_grid1_" + Request.QueryString["category"];```

[VB.NET]

```vbnet
CMSDataGrid.CacheItemName = "products_grid1_" & Request.QueryString("category")
```

1.4 Displaying related documents

If you specify the related documents in the Relationships dialog of the Content module, you can display them using one of the following controls:
- CMSDataGrid
- CMSDataList
- CMSRepeater
- CMSViewer

All of them have three properties that need to be used in order to display only related documents (beside other properties, such as SelectNodesPath, SelectNodesClassName, etc.):

- RelationshipWithNodeGUID / Main document - NodeGUID value of the main document - it's typically the currently displayed document.
- RelationshipName / Relationship name - code name of the relationship.
- RelatedNodeIsOnTheLeftSide / Main document is on the left side - indicates if the main document is on the left or right side of the relationship.

Example

The following example shows how to display news items related to the product.

1. Go to CMS Desk -> Content and click /Products/LCD Displays/Acer AL1511s, click Properties -> Related docs.
2. Add two relationships with name **is related to** with documents /News/Your first news and /News/Your second news.

<table>
<thead>
<tr>
<th>Actions</th>
<th>Left document</th>
<th>Relationship name</th>
<th>Right document</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Acer AL1511s</td>
<td>is related to</td>
<td>Your first news</td>
</tr>
<tr>
<td>X</td>
<td>Acer AL1511s</td>
<td>is related to</td>
<td>Your second news</td>
</tr>
</tbody>
</table>

3. Edit the Products.aspx template and add a CMSRepeater control.

4. Set the following control properties:
   - Path: /% (we want to display related news items from the whole web site)
   - ClassNames: cms.news
   - TransformationName: cms.news.preview
   - RelationshipName: choose Display **documents related to the current document**, check the box **Main document is on the left side** and choose relationship name **is related to**. Save the changes.
5. Click /Products/LCD Displays/Acer AL1511s and click Live site. You will see the list of related news items below the product displayed using the cms.news.preview transformation:

Product short description comes here.

Product description comes here.

Price: $199

Related news

9/25/2006 - Your first news

Summary comes here.

9/25/2006 - Your second news

Summary comes here.
1.5 Dynamic insertion of parameters in data web parts

The following controls support dynamically inserted parameters in the WhereCondition / WHERE condition property:

- QueryDatalist
- QueryDataGrid
- QueryRepeater

You can use the following expressions that are resolved at run-time:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>{%currentaliaspath%}</td>
<td>Alias path of the current page.</td>
</tr>
<tr>
<td>{%currentculturecode%}</td>
<td>Culture code of the user’s preferred content culture.</td>
</tr>
<tr>
<td>{%currentsiteid%}</td>
<td>SiteID value of the current site.</td>
</tr>
</tbody>
</table>

###WHERE### macro expression

The custom query you are using must contain the ###WHERE### expression that is dynamically replaced with value of the WHERE condition property value at run-time.

1.6 CMS Controls

Kentico CMS Controls is a set of ASP.NET server controls that allow you to publish content from the Kentico CMS database on your Web site. Some of them also provide additional features, such as search dialog.

Before you can use them, please make sure you configured your project as described in Configuring your project for Kentico CMS Controls.

Corporate Site sample required

The examples in this chapter assume that your Kentico CMS database contains data for the sample Corporate Site web site.
1.6.2 Configuring your project for Kentico CMS Controls

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

1. Open the web site 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 web site. Click **Open** and then click **OK**.

![Choose Toolbox Items](image)

<table>
<thead>
<tr>
<th>Name</th>
<th>Namespace</th>
<th>Assembly Name</th>
<th>Directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute Translation</td>
<td>Microsoft.AnalysisServices</td>
<td>Microsoft.AnalysisServ...</td>
<td>Global Ass...</td>
</tr>
<tr>
<td>BackgroundWorker</td>
<td>System.ComponentModel</td>
<td>System (2.0.0.0)</td>
<td>Global Ass...</td>
</tr>
<tr>
<td>BasicCalendar</td>
<td>CMS.Controls</td>
<td>CMS.Controls (2.0.246...</td>
<td>C:_Test)</td>
</tr>
<tr>
<td>BasicDataGrid</td>
<td>CMS.Controls</td>
<td>CMS.Controls (2.0.246...</td>
<td>C:_Test)</td>
</tr>
<tr>
<td>BasicDataList</td>
<td>CMS.Controls</td>
<td>CMS.Controls (2.0.246...</td>
<td>C:_Test)</td>
</tr>
<tr>
<td>BasicMultiColumnTable</td>
<td>CMS.Controls</td>
<td>CMS.Controls (2.0.246...</td>
<td>C:_Test)</td>
</tr>
<tr>
<td>BasicRepeater</td>
<td>CMS.Controls</td>
<td>CMS.Controls (2.0.246...</td>
<td>C:_Test)</td>
</tr>
<tr>
<td>BasicTabControl</td>
<td>CMS.Controls</td>
<td>CMS.Controls (2.0.246...</td>
<td>C:_Test)</td>
</tr>
<tr>
<td>BehaviorEditorPart</td>
<td>System.Web.UI.WebControls</td>
<td>System.Web (2.0.0.0)</td>
<td>Global Ass...</td>
</tr>
</tbody>
</table>
6. The controls are now added to the Toolbox:

![Toolbox with added controls]

7. Now you can easily drag and drop the controls on your Web form.

1.6.3 Using macro expressions in menu items

You can use macro expressions in menu item URL or JavaScript command (when you're editing the menu item in Kentico CMS Desk). These macros allow you to dynamically replace macro commands with specified values of the current menu item, such as alias path, id path, node name.

You only need to put `{%ColumnName%}` macros in the menu item URL or JavaScript attribute. The URL will look like this:

```
products.aspx?show=brand&aliaspath={%AliasPath%}
```

The menu will redirect user e.g. to URL `products.aspx?show=brand&aliaspath=~/MobileStore/Products/Nokia`.

Please note: All apostrophes (``) in source data are escaped to `\` so that they do not break JavaScript.
1.6.4 Using the CSSPrefix property for design of sub-menus

The CSSPrefix property allows you not only to use several menu controls with different styles on one page, but also to specify style of menu sub-items at any chosen level.

Here's an example of how to specify various style for particular menu levels (it can be used for both CMSMenu and CMSTreeMenu):

First, you need to specify the list of prefixes for particular levels using the CSSPrefix property:

CMSMenu1.CSSPrefix = "MainMenu;MainMenuSubMenu;MainMenuOtherLevels"

... now you define the following styles:

.MainMenuCMSMenu
... for menu control

.MainMenuCMSMenuItem
.MainMenuCMSMenuMouseUp
... etc. for the first level of the menu (level 0)

.MainMenuSubMenuCMSMenu
.MainMenuSubMenuCMSMenuMouseUp
... etc. for the second level of the menu (level 1)

.MainMenuOtherLevelsCMSMenu
.MainMenuOtherLevelsCMSMenuMouseUp
... etc. for all underlying levels of the menu (level 2 and all higher levels)
### 1.6.5 CMSBaseProperties - common properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CacheItemName</td>
<td>Name of the cache item the control will use. By setting this name dynamically, you can achieve caching based on URL parameter or some other variable - simply put the value of the parameter to the CacheItemName property. If no value is set, the control stores its content to the item named &quot;URL</td>
<td>ControlID&quot;.</td>
</tr>
<tr>
<td>CacheMinutes</td>
<td>Number of minutes the retrieved content is cached for. Zero indicates that the content will not be cached. -1 indicates the site-level settings should be used. This parameter allows you to set up caching of content so that it's not retrieved from the database each time a user requests the page. The default value is retrieved from Site Settings.</td>
<td>10</td>
</tr>
<tr>
<td>ControlTagKey</td>
<td>Overrides the generation of the SPAN tag with custom tag. The default value is retrieved from Site Settings.</td>
<td></td>
</tr>
<tr>
<td>OrderBy</td>
<td>ORDER BY part of the SQL statement.</td>
<td>&quot;NewsReleaseDate DESC&quot;</td>
</tr>
<tr>
<td>SiteName</td>
<td>Site code name.</td>
<td>&quot;mywebsite&quot;</td>
</tr>
<tr>
<td>StopProcessing</td>
<td>Indicates if processing of the control should be stopped and the control should not retrieve or display any data.</td>
<td></td>
</tr>
<tr>
<td>WhereCondition</td>
<td>WHERE part of the SQL statement.</td>
<td>&quot;ProductPrice &gt; 100&quot;</td>
</tr>
</tbody>
</table>
### CMSControlProperties - common properties

Inherits: CMSBaseProperties - common properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CheckPermissions</td>
<td>Allows you to specify whether to check permissions of the current user. If the value is 'false' (default value) no permissions are checked. Otherwise, only nodes for which the user has read permission are displayed.</td>
<td>true</td>
</tr>
<tr>
<td>ClassNames</td>
<td>Classname value or several values separated by semicolon.</td>
<td>&quot;cms.news&quot; or &quot;cms.news;cms.article&quot;</td>
</tr>
<tr>
<td>CombineWithDefaultCulture</td>
<td>Indicates if the documents from the default culture version should be alternatively used.</td>
<td>true</td>
</tr>
<tr>
<td>CultureCode</td>
<td>Culture code, such as en-us. If not specified, it's read from the user's session or the default value is used.</td>
<td>&quot;en-us&quot;</td>
</tr>
<tr>
<td>MaxRelativeLevel</td>
<td>Relative level of child documents that should be selected. -1 selects all child documents.</td>
<td></td>
</tr>
<tr>
<td>Path</td>
<td>Path of the documents to be displayed.</td>
<td>See Path specification in web parts and controls for examples.</td>
</tr>
<tr>
<td>SelectOnlyPublished</td>
<td>Indicates if only published documents should be displayed.</td>
<td></td>
</tr>
<tr>
<td>TreeProvider</td>
<td>Tree provider instance used to access data. If no TreeProvider is assigned, a new TreeProvider instance is created.</td>
<td></td>
</tr>
<tr>
<td>WordWrap</td>
<td>Indicates if text displayed by the (navigation) control can use word wrapping.</td>
<td></td>
</tr>
</tbody>
</table>
## 1.6.7 CMSDataProperties - common properties

Inherits: CMSControlProperties - common properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>RelatedNodeIsOnTheLeftSide</td>
<td>If true, the returned nodes are on the right side of the relationship.</td>
<td></td>
</tr>
<tr>
<td>RelationshipName</td>
<td>Name of the relationship.</td>
<td>&quot;isrelatedto&quot;</td>
</tr>
<tr>
<td>RelationshipWithNodeGUID</td>
<td>Select nodes with given relationship with given node.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>If you use 00000000-0000-0000-0000-000000000000 value, the relationships are not applied.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>If you use 11111111-1111-1111-1111-111111111111 value, the current node NodeGUID value is used automatically.</td>
<td></td>
</tr>
<tr>
<td>SelectedItemTransformationName</td>
<td>Transformation name for selected item in format application.class.transformation.</td>
<td>&quot;cms.news.preview&quot;</td>
</tr>
<tr>
<td>SelectTopN</td>
<td>Select top N rows.</td>
<td>5</td>
</tr>
</tbody>
</table>

## 1.6.8 CMSMenuProperties - common properties

Inherits: CMSControlProperties - common properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSSPrefix</td>
<td>Specifies prefix of standard CMSMenu CSS classes. You can also use several values separated with semicolon ; for particular levels.</td>
<td>&quot;main;submenu1;submenu2&quot;</td>
</tr>
<tr>
<td>HighlightAllItemsInPath</td>
<td>Indicates if all items in the unfolded path should be displayed as highlighted.</td>
<td></td>
</tr>
<tr>
<td>SubmenuIndicator</td>
<td>Contains a path to image that will be used on the right of every item that contains subitems.</td>
<td></td>
</tr>
<tr>
<td>UseAlternatingStyles</td>
<td>Indicates if alternating styles should be used for even and odd items in the same level of the menu.</td>
<td></td>
</tr>
</tbody>
</table>
1.6.9 CMSQueryProperties - common properties

Inherits: CMSBaseProperties - common properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>GeneralConnection</td>
<td>GeneralConnection instance. If it's not specified, a new instance of the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GeneralConnection is created and returned.</td>
<td></td>
</tr>
<tr>
<td>QueryName</td>
<td>Query name in format application.class.query.</td>
<td>&quot;cms.news.selectlatest&quot;</td>
</tr>
<tr>
<td>QueryParameters</td>
<td>Query parameters. The first dimension contains the name of the parameter in</td>
<td></td>
</tr>
<tr>
<td></td>
<td>format @param, the second dimension contains its value.</td>
<td></td>
</tr>
<tr>
<td>SelectTopN</td>
<td>Select top N rows.</td>
<td>5</td>
</tr>
</tbody>
</table>

1.6.10 BasicCalendar

The BasicCalendar control allows you to display a calendar with events, news and other date-based documents specified in the DataSource property value. It's inherited from the standard ASP.NET Calendar control, which means it provides advanced formatting capabilities and it allows you to display additional information for appropriate days.

Sea also: CMSCalendar

Inherits

Calendar (ASP.NET control)

Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataMember</td>
<td>Name of the table when DataSet is used as a DataSource.</td>
<td>&quot;MyTable&quot;</td>
</tr>
<tr>
<td>DataSource</td>
<td>Data source with calendar events - either DataSet or DataTable object.</td>
<td></td>
</tr>
<tr>
<td>DayField</td>
<td>Name of the field in the DataSource that contains the datetime value.</td>
<td>&quot;NewsReleaseDate&quot;</td>
</tr>
<tr>
<td>DayWithEventsStyle</td>
<td>Style of the day with some event.</td>
<td></td>
</tr>
<tr>
<td>ItemTemplate</td>
<td>Template for displaying a day with event.</td>
<td></td>
</tr>
<tr>
<td>NoEventsTemplate</td>
<td>Template for displaying a day without any event.</td>
<td></td>
</tr>
</tbody>
</table>
Design

You can modify the design of the calendar control by setting the standard properties of the ASP.NET Calendar control. You can find more details on particular properties in the .NET Framework documentation.

Example

This example will show you how to display a calendar with news items released on particular date.

- Create a new Web form.
- Drag and drop the BasicCalendar control on the form.
- Switch to the HTML mode and add the following code inside the BasicCalendar element. The ItemTemplate section specifies the look of the event that will be displayed in the calendar control. The NoEventsTemplate section specifies the look of the day without any event.

[C#], [VB.NET]

```csharp
<ItemTemplate>
  <a href='<%#ResolveUrl(CMS.CMSHelper.CMSContext.GetUrl(Convert.ToString(Eval("NodeAliasPath")), Convert.ToString(Eval("DocumentUrlPath")))) %>'>
    <%#Eval("NewsTitle")%>
  </a>
</ItemTemplate>
<NoEventsTemplate>No Event</NoEventsTemplate>
```

- Switch to the code behind and add the following code at the beginning of the code:

[C#]

```csharp
using CMS.CMSHelper;
```

[VB.NET]

```vbnet
Imports CMS.CMSHelper
```

- Add the following code to the Page_Load method:

[C#]

```csharp
BasicCalendar1.DataSource = TreeHelper.SelectNodes("/", false, "cms.news", null, "NewsReleaseDate", -1, true);
BasicCalendar1.DayField = "NewsReleaseDate";
BasicCalendar1.DataBind();
```
What you did

You have added the code that retrieves all news items from the Kentico CMS database and assigns them to the BasicCalendar control. You have specified the DayField that contains the date/time value. Then you called the BasicCalendar.DataBind method.

- Run the project. You should see a page like this:

<table>
<thead>
<tr>
<th>July</th>
<th>August 2006</th>
<th>September</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sun</strong></td>
<td><strong>Mon</strong></td>
<td><strong>Tue</strong></td>
</tr>
<tr>
<td>30 No Event</td>
<td>31 No Event</td>
<td>1 No Event</td>
</tr>
<tr>
<td>6 No Event</td>
<td>7 No Event</td>
<td>8 No Event</td>
</tr>
<tr>
<td>13 No Event</td>
<td>14 No Event</td>
<td>15 No Event</td>
</tr>
<tr>
<td>20 No Event</td>
<td>21 No Event</td>
<td>22 No Event</td>
</tr>
<tr>
<td>27 No Event</td>
<td>28 No Event</td>
<td>29 No Event</td>
</tr>
<tr>
<td>3 No Event</td>
<td>4 No Event</td>
<td>5 No Event</td>
</tr>
</tbody>
</table>

1.6.11 BasicDataGrid

The BasicDataGrid control is inherited from the standard ASP.NET DataGrid control. It automatically ensures data binding, paging and sorting. You can use the standard DataGrid designer to set up BasicDataGrid style and behavior.

BasicDataGrid can be used with any bindable data source - it doesn't use Kentico CMS database or API.

See also: CMSDataGrid.
Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataBindByDefault</td>
<td>Indicates whether data binding should be performed by default.</td>
<td></td>
</tr>
<tr>
<td>HideControlForZeroRows</td>
<td>Hides the control when no data is loaded. Default value is False.</td>
<td></td>
</tr>
<tr>
<td>ProcessSorting</td>
<td>Indicates if sorting should be processed in DataView instead of sorting on the SQL level.</td>
<td></td>
</tr>
<tr>
<td>SortAscending</td>
<td>Direction of sorting. Default value is True.</td>
<td></td>
</tr>
<tr>
<td>SortField</td>
<td>Gets or sets the sort field. It can be used for setting the default sort field.</td>
<td>&quot;NewsReleaseDate&quot;</td>
</tr>
<tr>
<td>ZeroRowsText</td>
<td>Text to be shown when control is hidden by HideControlForZeroRows.</td>
<td>&quot;No records found&quot;</td>
</tr>
<tr>
<td>SetFirstPageAfterSortChange</td>
<td>Indicates if the current page should be set to the first page when the sorting is changed.</td>
<td></td>
</tr>
</tbody>
</table>

Design

The design can be modified in the same way as the standard DataGrid control.

Example

This example will show you how to read the list of products and display it in the grid.

1. Create a new Web form.
2. Drag and drop the BasicDataGrid control on the form.
3. In the Properties window, click Auto Format... and choose some color schema.
4. In the Properties window, click Property Builder..., the BasicDataGrid1 Properties dialog appears.
   - On the General tab, check the "Allow sorting" box.
   - Now we will specify the columns that will be displayed. On the Columns tab:
     - Uncheck the Create columns automatically at run time box.
     - Add a new Bound Column from the Available columns list to the Selected columns list. Enter the following values in the appropriate fields:
       - Header text: Name
       - Data field: ProductName
       - Sort expression: ProductName
     - Add another Bound Column from the Available columns list to the Selected columns list. Enter the following values in the appropriate fields:
       - Header text: Price
       - Data field: ProductPrice
       - Sort expression: ProductPrice
   - On the Paging tab check the box Allow Paging. Click OK.
5. Add the following code at the beginning of the Web form code-behind:

[C#]

```csharp
using CMS.CMSHelper;
```

[VB.NET]

```vbnet
Imports CMS.CMSHelper
```

What you did
You have included namespaces we will use.

6. Add the following code to the `Page_Load` method:

[C#]

```csharp
DataSet ds = TreeHelper.SelectNodes("/\%", false, "CMS.Product", "", "ProductName", -1, true);
BasicDataGrid1.DataSource = ds;
BasicDataGrid1.DataBind();
```

[VB.NET]

```vbnet
Dim ds As DataSet = TreeHelper.SelectNodes("/\%", False, "CMS.Product", "", "ProductName", -1, true)
BasicDataGrid1.DataSource = ds
BasicDataGrid1.DataBind()
```

What you did
You have added code that reads data from the database and provides them to the BasicDataGrid control.

7. Compile and run the project. You should see a page like this:

<table>
<thead>
<tr>
<th>Name</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acer AL1511s</td>
<td>199</td>
</tr>
<tr>
<td>FS V2030</td>
<td>249</td>
</tr>
</tbody>
</table>

1.6.12 BasicDataList

The BasicDataList control is inherited from the standard ASP.NET DataList control. It automatically ensures data binding. You can use the common ASP.NET DataList tags to set up BasicDataList style and behavior - please see the .NET Framework documentation for details.

BasicDataList can be used with any bindable data source - it doesn't use Kentico CMS database or API.
Unlike BasicRepeater, BasicDataList allows you to display data in several columns.

**Please note:** If you want to display data from Kentico CMS, please use the CMSDataList control that provides a more convenient way.

**Please note**
If you want to display data from Kentico CMS, please use the CMSDataList control that provides a more convenient way.

Inherits: **DataList (ASP.NET control)**

**Properties**

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataBindByDefault</td>
<td>Indicates whether data binding should be performed by default.</td>
<td></td>
</tr>
<tr>
<td>HideControlForZeroRows</td>
<td>Hides the control when no data is loaded. The default value is False.</td>
<td></td>
</tr>
<tr>
<td>ZeroRowsText</td>
<td>Text to be shown when the control is hidden by HideControlForZeroRows.</td>
<td>&quot;No records found.&quot;</td>
</tr>
</tbody>
</table>

**Design**

The design can be modified in the same way as for the standard DataList control.

**Example**

This example will show you how to read a list of products (from the sample Corporate Site) and display it using the BasicDataList control.

1. Create a new Web form.
2. Drag and drop the BasicDataList control on the form. Change its **RepeatColumns** property value to 3.
3. Switch to the HTML mode and add the following code inside the BasicDataList tags:
What you did

You have defined a template for one product displayed by the DataList control. The control dynamically replaces the `<%# ... %>` tags with values of the current record. This is repeated for each record in the SQL query result. The ResolveUrl(~) command generates the path to the root of your web project.

4. Add the following code at the beginning of the Web form code-behind:

[C#]

```csharp
using CMS.CMSHelper;
```

[VB.NET]

```vbnet
Imports CMS.CMSHelper
```

What you did

You have included namespaces we will use.

5. Add the following code to the Page_Load method:

[C#]

```csharp
DataSet ds = TreeHelper.SelectNodes("/%", false, "CMS.Product", ",", "ProductName", -1, true);
BasicDataList1.DataSource = ds;
BasicDataList1.DataBind();
```

[VB.NET]

```vbnet
Dim ds as DataSet = TreeHelper.SelectNodes("/%", false, "CMS.Product", ",", "ProductName", -1, true)
BasicDataList1.DataSource = ds
BasicDataList1.DataBind()
```

What you did

You have added code that reads data from the database and provides them to the BasicDataList control.

6. Compile and run the project. You should see a page like this:
1.6.13 BasicMultiColumnTable

The BasicMultiColumnTable control allows you to display a table where documents are rendered column-by-column instead of common row-by-row model used in the DataGrid and other controls. This control is useful if you want e.g. compare parameters of various products.

See also: CMSDocumentComparison

Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataSource</td>
<td>DataTable object containing rows to be displayed.</td>
<td></td>
</tr>
<tr>
<td>RenderFalseValueUsingEmptyTemplate</td>
<td>Indicates if false value of boolean type-columns should be displayed using the template for empty value.</td>
<td></td>
</tr>
<tr>
<td>TableParams</td>
<td>Params of the table to be displayed. A string array of size (x, 4) where x is the number of rows of the table and the second parameter is:</td>
<td>Standard row:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0, 0) = &quot;Product weight:&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0, 1) = &quot;Weight&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0, 2) = &quot;{%weight%} kg&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0, 3) = &quot;N/A&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Separator row:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0, 0) = &quot;This is separator&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - Row caption</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - Row source column</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - Non-empty-column template (if omitted, the value of the source column is used)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - Empty-column template (if omitted, the &quot; &quot;,&quot; value is used)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When specifying the template, you can use {%fieldname%} tags to insert value of any field of the source data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If you need to create a separator row, set only the (x, 0) value to the separator text and do not set the other values (x, 1), (x, 2) and (x, 3) - they must be nothing (null).</td>
</tr>
</tbody>
</table>
Design

You can modify the design of the table control by setting the following properties:

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TableCellSpacing</td>
<td>Table cell spacing.</td>
<td>0</td>
</tr>
<tr>
<td>TableCellPadding</td>
<td>Table cell padding.</td>
<td>0</td>
</tr>
<tr>
<td>TableCellCssClass</td>
<td>Table cell (TD) CSS class.</td>
<td>&quot;MulticolumnTableCell&quot;</td>
</tr>
<tr>
<td>TableCssClass</td>
<td>Table CSS class.</td>
<td>&quot;MulticolumnTable&quot;</td>
</tr>
<tr>
<td>TableFirstColumnCellCssClass</td>
<td>CSS class of the first table columns’ cells (TD).</td>
<td>&quot;MulticolumnTableFirstColumnCell&quot;</td>
</tr>
<tr>
<td>TableFirstRowCellCSSClass</td>
<td>Table cell (TD) CSS class for the first row of the table.</td>
<td>&quot;MulticolumnTableFirstRowCell&quot;</td>
</tr>
<tr>
<td>TableFirstRowCssClass</td>
<td>First table row (TR) CSS class.</td>
<td>&quot;MulticolumnTableFirstRow&quot;</td>
</tr>
<tr>
<td>TableRowCssClass</td>
<td>Table row (TR) CSS class.</td>
<td>&quot;MulticolumnTableRow&quot;</td>
</tr>
<tr>
<td>TableSeparatorCellCssClass</td>
<td>CSS class of the separator cell (TD).</td>
<td>&quot;MulticolumnTableSeparatorCell&quot;</td>
</tr>
<tr>
<td>TableSeparatorRowCssClass</td>
<td>CSS class of the separator row (TR).</td>
<td>&quot;MulticolumnTableSeparatorRow&quot;</td>
</tr>
</tbody>
</table>

Example

This example will show you how to display a product comparison:

1. Create a new web form.
2. Drag and drop the **BasicMultiColumnTable** control on the form.
3. Switch to the code behind and add the following code at the beginning of the code:

[C#]

```csharp
using CMS.CMSHelper;
```

[VB.NET]

```vbnet
Imports CMS.CMSHelper
```
4. Add the following code to the Page_Load method:

[C#]

```csharp
string[,]tableParams = new string[3, 4];
DataSet ds = null;

// set data source
ds = TreeHelper.SelectNodes("/%", false, "cms.product", ",", "ProductName", -1, true);
BasicMultiColumnTable1.DataSource = ds.Tables[0];

// define table header with product names
tableParams[0, 0] = "Product:";
tableParams[0, 1] = "productName";
// add row with product description
tableParams[1, 0] = "Description:";
tableParams[1, 1] = "ProductDescription";
// add row with product price
tableParams[2, 0] = "Price:";
tableParams[2, 1] = "productPrice";
tableParams[2, 2] = "USD {%ProductPrice%}";
tableParams[2, 3] = "N/A";
BasicMultiColumnTable1.TableParams = tableParams;
```

[VB.NET]

```vbnet
Dim tableParams(2, 3) As String
Dim ds As DataSet

'set data source
ds = TreeHelper.SelectNodes("/%", False, "cms.product", ",", "ProductName", -1, True)
BasicMultiColumnTable1.DataSource = ds.Tables(0)

'define table header with product names
tableParams(0, 0) = "Product:"
tableParams(0, 1) = "ProductName"
'add row with product description
tableParams(1, 0) = "Description:"
tableParams(1, 1) = "ProductDescription"
'add row with product weight
tableParams(2, 0) = "Price:"
tableParams(2, 1) = "ProductPrice"
tableParams(2, 2) = "USD {%ProductPrice%}"
tableParams(2, 3) = "N/A"
BasicMultiColumnTable1.TableParams = tableParams
```

What you did

You have added the code that retrieves all products from the Kentico CMS database and assigns them to the `BasicMultiColumnTable` control. The `tableParams` array defines how the data should be displayed.

5. Compile and run the project. You should see a page like this:
1.6.14 BasicRepeater

The BasicRepeater control is inherited from the standard ASP.NET Repeater control. It ensures data binding automatically. You can use the common ASP.NET Repeater tags to set up BasicRepeater style and behavior - please see the .NET Framework documentation for details.

BasicRepeater can be used with any bindable data source - it doesn't use Kentico CMS database or API.

Please note: If you want to display documents from Kentico CMS, please use the CMSRepeater control that provides a more convenient way.

Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataBindByDefault</td>
<td>Indicates whether data binding should be performed by default.</td>
<td></td>
</tr>
<tr>
<td>HideControlForZeroRows</td>
<td>Hides the control when no data is loaded. Default value is False.</td>
<td></td>
</tr>
<tr>
<td>ZeroRowsText</td>
<td>Text to be shown when the control is hidden by HideControlForZeroRows.</td>
<td>&quot;No records found.&quot;</td>
</tr>
</tbody>
</table>

Design

The design can be modified in the same way as for the standard Repeater control.

Example

This example will show you how to read a list of products and display it using the repeater.

- Create a new Web form.
- Drag and drop the BasicRepeater control on the form.
- Switch to the Source mode and add the following code inside the BasicRepeater tags:

[C#, VB.NET]

```
<ItemTemplate>
  <h3>  
    <%# Eval("ProductName") %>
  </h3>
  <img src='<%# ResolveUrl("~/CMSPages/GetFile.aspx?guid=" + Eval("ProductPhoto").ToString()) %>' />
</ItemTemplate>
```

What you did

You have defined a template for one record displayed by the repeater control. The control dynamically replaces the <%# ... %> tags with values of the current record. This is repeated for each record in the datasource.
• Add the following code at the beginning of the Web form code-behind:

[C#]

using CMS.CMSHelper;

[VB.NET]

Imports CMS.CMSHelper

What you did

You have included namespaces we will use.

• Add the following code to the Page_Load method:

[C#]

DataSet ds = TreeHelper.SelectNodes("/", false, "CMS.Product", "", "ProductName", -1, true);
this.BasicRepeater1.DataSource = ds;
this.BasicRepeater1.DataBind();

[VB.NET]

Dim ds As DataSet = TreeHelper.SelectNodes("/", False, "CMS.Product", "", "ProductName", -1, True)
BasicRepeater1.DataSource = ds
BasicRepeater1.DataBind()
What you did

You have added code that reads documents from the database and provides them to the BasicRepeater control.

- Compile and run the project. You should see a page like this:

**Acer AL1511s**

![Acer AL1511s](image1)

**FS V2030**

![FS V2030](image2)
1.6.15 BasicTabControl

The BasicTabControl control displays several tabs according to provided data. BasicTabControl does't use Kentico CMS database or API.

**Please note:** If you want to display a tab menu based on the data from Kentico CMS, please use the CMSTabControl control.

**Properties**

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SelectedTab</td>
<td>Index of the selected tab.</td>
<td>2</td>
</tr>
<tr>
<td>TabControlLayout</td>
<td>Horizontal or vertical layout.</td>
<td>TabControlLayoutEnum.Horizontal</td>
</tr>
</tbody>
</table>
| Tabs                   | 4- or 7-dimensional array of tabs:                    | tabs[0, 0] = "&nbsp;Home&nbsp;";  
tabs[0, 1] = "OnClick JavaScript"  
tabs[0, 2] = URL  
tabs[0, 3] = tooltip  
tabs[0, 4] = left image URL  
tabs[0, 5] = center image URL  
tabs[0, 6] = right image URL  
**Please note:**  
1. The image URLs in dimensions 4, 5 and 6 are optional.  
2. If you specify the center image URL, the image is displayed instead of the title. |
| UrlTarget              | If URL for tab items is set, this property specifies target frame for all URLs. | "_blank"                                                                    |
| UseClientScript        | Indicates if client script should be generated for each tab. |                                                                             |
| TabControlIdPrefix     | Prefix that will be used for all IDs in the HTML code rendered by the BasicTabControl. It's useful if you need to place multiple tab controls on the same page. | "FirstTab"                                                                  |
Design

You can modify the design using the following CSS Classes:

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TabControlTable</td>
<td>Table that contains the tabs (&lt;TABLE&gt; tag).</td>
</tr>
<tr>
<td>TabControlRow</td>
<td>Table row (&lt;TR&gt; tag).</td>
</tr>
<tr>
<td>TabControl</td>
<td>Tab item (&lt;TD&gt; tag).</td>
</tr>
<tr>
<td>TabControlSelected</td>
<td>Selected tab item (&lt;TD&gt; tag).</td>
</tr>
<tr>
<td>TabControlLink</td>
<td>Tab item link - for case a URL is specified (&lt;A&gt; tag).</td>
</tr>
<tr>
<td>TabControlLinkSelected</td>
<td>Selected tab item link - for case a URL is specified (&lt;A&gt; tag).</td>
</tr>
<tr>
<td>TabControlLeft</td>
<td>Left side of the tab item (&lt;TD&gt; tag).</td>
</tr>
<tr>
<td>TabControlRight</td>
<td>Right side of the tab item (&lt;TD&gt; tag).</td>
</tr>
<tr>
<td>TabControlSelectedLeft</td>
<td>Left side of the selected tab item (&lt;TD&gt; tag).</td>
</tr>
<tr>
<td>TabControlSelectedRight</td>
<td>Right side of the selected tab item (&lt;TD&gt; tag).</td>
</tr>
</tbody>
</table>

Example

This example will show you how to display a simple tab-menu.

- Create a new Web form.
- Drag and drop the BasicTabControl control on the form.
- Switch to the Source mode and add the following CSS styles inside the <BODY> tag. It will modify the appearance of the tabs.

[C#], [VB.NET]

```html
<style>
.TabControlTable { FONT-SIZE: 14px; FONT-FAMILY: Arial,Verdana }
.TabControl { BORDER-RIGHT: black 1px solid; BORDER-TOP: black 1px solid; FONT-WEIGHT: bold; BACKGROUND: #e7e7ff; BORDER-LEFT: black 1px solid; CURSOR: hand; COLOR: black }
.TabControlSelected { BORDER-RIGHT: black 1px solid; BORDER-TOP: black 1px solid; FONT-WEIGHT: bold; BACKGROUND: #4a3c8c; BORDER-LEFT: black 1px solid; CURSOR: default; COLOR: white }
.TabControlLinkSelected { COLOR: white; TEXT-DECORATION: none }
.TabControlLink { COLOR: black; TEXT-DECORATION: none }
.TabControlLeft { WIDTH: 1px }
.TabControlRight { WIDTH: 0px }
.TabControlSelectedLeft { WIDTH: 1px }
.TabControlSelectedRight { WIDTH: 0px }
</style>
```
Add the following table code just after the BasicTabControl tags. It will display a stripe under the tabs.

[C#], [VB.NET]

```html
<hr style="width:100%; height:2px; margin-top:0px;" />
```

Switch to the page code-behind and add the following code to the Page_Load method:

[C#]

```csharp
string[,] tabs = new string[3, 7];
tabs[0, 0] = "&nbsp;Home&nbsp;";
tabs[0, 1] = "alert(\"It is very simple!\")";
tabs[0, 2] = "http://www.kentico.com";
tabs[1, 0] = "&nbsp;Products&nbsp;";
tabs[1, 1] = "http://www.comparesql.com";
tabs[1, 2] = "&nbsp;Contact&nbsp;";
tabs[2, 0] = "http://www.syncserverfiles.com";
tabs[2, 3] = "Some tooltip";
BasicTabControl1.Tabs = tabs;
BasicTabControl1.SelectedTab = 0;
BasicTabControl1.UrlTarget = "_blank";
BasicTabControl1.UseClientScript = true;
```

[VB.NET]

```vbnet
Dim tabs(2, 6) As String
tabs(0, 0) = "&nbsp;Home&nbsp;"
tabs(0, 1) = "alert(\"It\'s very simple!\")"
tabs(0, 2) = "http://www.kentico.com"
tabs(1, 0) = "&nbsp;Products&nbsp;"
tabs(1, 1) = "http://www.comparesql.com"
tabs(1, 2) = "&nbsp;Contact&nbsp;"
tabs(2, 2) = "http://www.syncserverfiles.com"
tabs(2, 3) = "Some tooltip"
BasicTabControl1.Tabs = tabs
BasicTabControl1.SelectedTab = 0
BasicTabControl1.UrlTarget = "_blank"
BasicTabControl1.UseClientScript = True
```

What you did

You have added code that creates an array of tab items and assigns it to the BasicTabControl control. It also selects the first tab and sets the target frame to "_blank".

Compile and run the project. You will see a page like this:

```
Home  Products  Contact
```

© 2006-2008 Kentico Software
1.6.16 CMSBreadCrumbs

The CMSBreadCrumbs control allows you to display current user's position within the Web site in format

**Item 1 > Item 2 > Item 3**

where Item X is name of the document in the path.

**Inherits:** CMSControlProperties - common properties

**Properties**

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BreadCrumbSeparator</td>
<td>Character(s) that separate the bread crumbs. You can use HTML code here.</td>
<td>&quot;&gt;&gt;&quot;</td>
</tr>
<tr>
<td>DefaultPath</td>
<td>Path to the node whose path should be displayed. This value is used in case no Path value is provided and no alias path is provided through friendly URL.</td>
<td>&quot;/home&quot;</td>
</tr>
<tr>
<td>ShowCurrentItem</td>
<td>Indicates if the current (last) item should be displayed.</td>
<td></td>
</tr>
<tr>
<td>ShowCurrentItemAsLink</td>
<td>Indicates if the current (last) item should be displayed as a link.</td>
<td></td>
</tr>
<tr>
<td>StartingPath</td>
<td>Starting part of the path.</td>
<td>&quot;/products&quot;</td>
</tr>
<tr>
<td>UrlTarget</td>
<td>Specifies target frame for all URLs.</td>
<td>&quot;._blank&quot;</td>
</tr>
<tr>
<td>LoadDataAutomatically</td>
<td>Indicates if data for the control should be loaded automatically. By default, the data is loaded automatically. If you set this property to false, you can supply custom DataSet to the DataSource property and then call the ReloadData(false) method.</td>
<td></td>
</tr>
<tr>
<td>RenderedHTML</td>
<td>Allows you to get or set the HTML code rendered by the control.</td>
<td></td>
</tr>
</tbody>
</table>
  
You need to set this property before the Render event - e.g. in the OnLoad event.
Methods

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReloadData</td>
<td>Reloads the data. If the <code>forceLoad</code> parameter is set to false and the custom value is assigned to the DataSource property, the properties of the CMSListMenu control are not used and only the data from the DataSource are used.</td>
</tr>
</tbody>
</table>

Design

You can modify the design using the following CSS styles:

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMSBreadCrumbsLink</td>
<td>Link (A element) in the bread crumbs path.</td>
</tr>
<tr>
<td>CMSBreadCrumbsCurrentItem</td>
<td>Style of the last item representing the current location (it's not a link).</td>
</tr>
</tbody>
</table>

**CMSBreadCrumbs Example**

This example will show you how to display user’s current location within Web site structure. It assumes that you have configured your project for Kentico CMS Controls.

- Create a new Web form.
- Drag and drop the **CMSBreadCrumbs** control on the form.
- In the Properties window, set the following property value:
  - Path: /Products/Notebooks

Compile and run the project. You should see a page like this:

- Products > Notebooks

**1.6.17 CMSCalendar**

The CMSCalendar control allows you to display a calendar with events, news and other date-based documents from the Kentico CMS database. It inherits the BasicCalendar control.

**Data Source**

Data retrieved using the SelectDocuments query of the specified document type.

**Inherits:** BasicCalendar, CMSControlProperties - common properties
Own Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataSource</td>
<td>Gets or sets a DataSet containing values used to populate the items within the control. This value needn't be set.</td>
<td></td>
</tr>
<tr>
<td>NoEventTransformationName</td>
<td>No event transformation name in format application.class.transformation.</td>
<td>&quot;cms.news.noevent&quot;</td>
</tr>
<tr>
<td>RelatedNodesOnTheLeftSide</td>
<td>If true, the returned nodes are on the right side of the relationship.</td>
<td></td>
</tr>
<tr>
<td>RelationshipName</td>
<td>Name of the relationship.</td>
<td>&quot;isrelatedto&quot;</td>
</tr>
<tr>
<td>RelationshipWithNodeId</td>
<td>Select nodes with given relationship with given node.</td>
<td>10</td>
</tr>
<tr>
<td>TransformationName</td>
<td>Transformation name in format application.class.transformation.</td>
<td>&quot;cms.news.event&quot;</td>
</tr>
<tr>
<td>TreeProvider</td>
<td>Tree provider instance used to access data. If no TreeProvider is assigned, a new TreeProvider instance is created.</td>
<td></td>
</tr>
</tbody>
</table>

Design

You can modify the design of the calendar control by setting the standard properties of the ASP.NET Calendar control. You can find more details on particular properties in the .NET Framework documentation.

Example

This example will show you how to display a calendar with news items released on particular date.

1. Create a new Web form.
2. Drag and drop the CMSCalendar control on the form.
3. Set the following properties - they specify what should be displayed.

- **DayField** = "NewsReleaseDate"
- **ClassNames** = "cms.news"
- **Path** = "/%"
- **TransformationName** = "cms.news.calendarevent"
- **NoEventTransformationName** = "cms.news.calendarnoevent"
4. Run the project. You should see a page like this:

<table>
<thead>
<tr>
<th></th>
<th>Sun</th>
<th></th>
<th>Mon</th>
<th></th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th></th>
<th>Fri</th>
<th></th>
<th>Sat</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>30</td>
<td></td>
<td>31</td>
<td></td>
<td>1</td>
<td></td>
<td>2</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>No event</td>
<td>No event</td>
<td>No event</td>
<td></td>
<td>No event</td>
<td></td>
<td>No event</td>
<td></td>
<td>No event</td>
<td></td>
<td>No event</td>
</tr>
<tr>
<td>6</td>
<td>No event</td>
<td>7</td>
<td>No event</td>
<td>8</td>
<td>No event</td>
<td>9</td>
<td>No event</td>
<td>10</td>
<td>No event</td>
<td>11</td>
<td>No event</td>
</tr>
<tr>
<td>13</td>
<td>No event</td>
<td>14</td>
<td>No event</td>
<td>15</td>
<td>No event</td>
<td>16</td>
<td>No event</td>
<td>17</td>
<td>No event</td>
<td>18</td>
<td>No event</td>
</tr>
<tr>
<td>20</td>
<td>No event</td>
<td>21</td>
<td>No event</td>
<td>22</td>
<td>No event</td>
<td>23</td>
<td>No event</td>
<td>24</td>
<td>No event</td>
<td>25</td>
<td>No event</td>
</tr>
<tr>
<td>27</td>
<td>No event</td>
<td>28</td>
<td>No event</td>
<td>29</td>
<td>No event</td>
<td>30</td>
<td>No event</td>
<td>31</td>
<td>No event</td>
<td>1</td>
<td>No event</td>
</tr>
<tr>
<td>3</td>
<td>No event</td>
<td>4</td>
<td>No event</td>
<td>5</td>
<td>No event</td>
<td>6</td>
<td>No event</td>
<td>7</td>
<td>No event</td>
<td>8</td>
<td>No event</td>
</tr>
</tbody>
</table>

1.6.18 CMSDataGrid

The CMSDataGrid displays a table based on Kentico CMS data. It inherits the BasicDataGrid control. It automatically ensures databinding, paging and sorting.

It allows you to display Kentico CMS documents specified by path, depth, document type and WHERE condition. The CMSDataGrid control displays content without writing any additional code.

You can use the common DataGrid designer to set up CMSDataGrid style and behavior.

**Please note:** If you want to display data using custom query, please use the QueryDataGrid control.

**Data Source**

Data retrieved using the SelectDocuments query of the specified document type.

**Inherits:** CMSDataProperties - common properties, BasicDataGrid

**Design**

The design can be modified in the same way as the standard ASP.NET DataGrid control.

**Example**

This example will show you how to display a list of all mobile phones in the grid using the CMSDataGrid control.
1. Create a new Web form.

2. Drag and drop the CMSDataGrid control on the form.

3. In the **Properties** window, click **Auto Format**... and choose some color schema.

4. In the **Properties** window, click **Property Builder**..., the CMSDataGrid1 Properties dialog appears.

   - On the **General** tab check the "Allow sorting" box.
   - Now we will specify the columns that will be displayed. On the **Columns** tab:
     - Uncheck the Create columns automatically at run time box.
     - Add a new Bound Column from the Available columns list to the Selected columns list. Enter the following values in the appropriate fields:
       - **Header text**: Product Name
       - **Data field**: ProductName
       - **Sort expression**: ProductName
     - Add a new Bound Column from the Available columns list to the Selected columns list. Enter the following values in the appropriate fields:
       - **Header text**: Price
       - **Data field**: ProductPrice
       - **Sort expression**: ProductPrice
     - On the **Paging** tab check the box **Allow Paging**. Click **OK**.

5. In the **Properties** window set the following property values:

   - **ClassNames**: cms.product
   - **MaxRelativeLevel**: -1
   - **Path**: /products/%

6. Compile and run the project. You should see a page like this:

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>AcerAL1511s</td>
<td>199</td>
</tr>
<tr>
<td>FS V2030</td>
<td>249</td>
</tr>
<tr>
<td>HP nx6110</td>
<td>349</td>
</tr>
<tr>
<td>IPAQ HW6510</td>
<td>249</td>
</tr>
<tr>
<td>IPAQ RX1950</td>
<td>299</td>
</tr>
<tr>
<td>NEC 52vn</td>
<td>499</td>
</tr>
</tbody>
</table>
```

### 1.6.19 CMSDataList

The CMSDataList control is inherited from the BasicDataList control.

It allows you to display part of the CMS content specified by its path, depth, document template, WHERE condition and ORDER BY clause. The CMSDataList control displays content without writing any additional code. Unlike CMSRepeater, the CMSDataList control allows you to display content in several columns.

**Please note:** If you want to display data using custom query, please use the QueryDataList control.
**Data Source**

Data retrieved using the SelectDocuments query of the specified document template.

**Inherits:** CMSDataProperties - common properties, BasicDataList  
**See also:** DataPager

**Properties**

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlternatingTransformationName</td>
<td>Transformation name in format application.class.transformation applied to alternating items.</td>
<td>&quot;cms.product.alternatepreview&quot;</td>
</tr>
<tr>
<td>EnablePaging</td>
<td>Enables paging.</td>
<td></td>
</tr>
<tr>
<td>PagerControl</td>
<td>Represents pager control. See DataPager for more details.</td>
<td></td>
</tr>
<tr>
<td>TransformationName</td>
<td>Transformation name in format application.class.transformation.</td>
<td>&quot;cms.product.preview&quot;</td>
</tr>
<tr>
<td>NestedControlsID</td>
<td>IDs of the nested controls (CMSRepeater, CMSDataList), separated by semicolons.</td>
<td>&quot;CMSRepeaterNested; CMSDataListNested&quot;</td>
</tr>
</tbody>
</table>

**Please note:** you can find an example of datalist/repeater nesting in CMSRepeater chapter.

**Design**

The design can be modified in the same way as for the standard ASP.NET DataList control and by the transformations defined in the AlternatingTransformationName, TransformationName and SelectedItemTransformationName properties.

**Example**

This example will show you how to read a list of documents and display it using the repeater.

1. Create a new Web form.
2. Drag and drop the CMSDataList control on the form.
3. In the Properties window set the following property values:
   - ClassNames: cms.product
   - MaxRelativeLevel: -1
   - OrderBy: ProductName
   - Path: /products/%
   - RepeatColumns: 2
   - TransformationName: cms.product.preview
   - SelectedItemTransformationName: cms.product.default

   Please note: as you defined TransformationName property, it's not necessary to define the standard ItemTemplate element of the DataList control.
4. Compile and run the project. You should see a page like this:

**Acer AL1511s   FS V2030**

$199  $249

1.6.20 **CMSDocumentComparison**

The CMSDocumentComparison control allows you to display several documents in table columns for comparison. This control is useful if you want to compare parameters of various products. The user can add the available documents to the comparison table and remove selected displayed documents from the comparison.

CMSDocumentComparison control uses the BasicMultiColumnTable control.

**Data Source**

Kentico CMS documents specified using standard properties (Path, ClassNames, etc.).

**Inherits:** CMSControlProperties - common properties

**Properties**

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddButton</td>
<td>Add button control.</td>
<td></td>
</tr>
<tr>
<td>BasicMultiColumnTable</td>
<td>BasicMultiColumnTable control instance.</td>
<td></td>
</tr>
<tr>
<td>DocumentList</td>
<td>Drop-down list with all comparable documents.</td>
<td></td>
</tr>
<tr>
<td>DropDownListColumn</td>
<td>Column to be displayed in the drop-down list.</td>
<td>&quot;ProductName&quot;</td>
</tr>
<tr>
<td>RemoveAllText</td>
<td>Text of the link for removing all compared documents.</td>
<td>&quot;Remove all&quot;</td>
</tr>
<tr>
<td>RemoveText</td>
<td>Text of the link for removing selected document.</td>
<td>&quot;Remove&quot;</td>
</tr>
<tr>
<td>TableParams</td>
<td>See BasicMultiColumnTable</td>
<td></td>
</tr>
<tr>
<td>TagKey</td>
<td>Overrides the generation of the SPAN tag with custom tag.</td>
<td></td>
</tr>
</tbody>
</table>
Design

See BasicMultiColumnTable for details on design.

You can also modify the properties of the appropriate controls (BasicMultiColumnTable, AddButton, DocumentList).

Example

This example will show you how to display a product comparison.

1. Create a new web form.

2. Drag and drop the CMSDocumentComparison control on the form. Set the following properties:
   - ClassNames="cms.product"
   - Path="/Products/%" 
   - DropDownListColumn="ProductName"

3. Add the following code to the Page_Load method:

[C#]

```csharp
// define table rows
string[,] tableParameters = new string[4, 4];
tableParameters[0, 0] = "Product:";
tableParameters[0, 1] = "ProductName";
tableParameters[1, 0] = "Description:";
tableParameters[1, 1] = "ProductDescription";
tableParameters[2, 0] = "Price:";
tableParameters[2, 1] = "ProductPrice";
tableParameters[2, 2] = "USD {%ProductPrice%}";
tableParameters[2, 3] = "N/A";
this.CMSDocumentComparison1.TableName = tableParameters;

// set CSS styles
this.CMSDocumentComparison1.BasicMultiColumnTable.TableFirstRowCellCssClass = "MulticolumnTableFirstRowCell";
this.CMSDocumentComparison1.BasicMultiColumnTable.TableFirstColumnCellCssClass = "MulticolumnTableFirstColumnCell";
```
What you did

You have specified the rows of the BasicMultiColumnTable control and styles.

4. Compile and run the project. You should see a page like this:

![Page example](image)

Product: NEC 52vm
Description: Product description comes here.
Price: USD 499

1.6.21 CMSDocumentValue

This control displays the specified value of the currently displayed document. It's useful if you need to display e.g. the current document name on the page.

Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AttributeName</td>
<td>Name of the field to be displayed.</td>
<td>&quot;DocumentName&quot;</td>
</tr>
<tr>
<td>ClassNames</td>
<td>List of document types for which the value should be displayed, separated with a semicolon (;).</td>
<td>&quot;cms.article;cms.menuitem&quot;</td>
</tr>
<tr>
<td>FormattingString</td>
<td>.NET formatting expression for displaying the value.</td>
<td>&quot;Name: {0}&quot;</td>
</tr>
</tbody>
</table>

You can place this control into web parts, ASPX page templates or layouts.
Example

You only need to add the following code inside your ASPX page or web part:

[C#], [VB.NET]

```csharp
<cc1:CMSDocumentValue ID="docVal" runat="server" AttributeName="DocumentName" FormattingString="Document name: {0}" />
```

1.6.22 CMSEditModeButtonAdd

The CMSEditModeButtonAdd control displays a button that is shown in the editing mode and allows content editors to add a new document when they click it. It provides an intuitive way of adding new documents.

Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClassName</td>
<td>Document type in format cms.article that specifies the type of the document that should be created.</td>
<td>&quot;cms.article&quot;</td>
</tr>
<tr>
<td>Path</td>
<td>Alias path of the parent document under which the new document is created. If omitted, the documents are added under the currently selected document.</td>
<td>&quot;/whitepapers&quot;</td>
</tr>
<tr>
<td>Text</td>
<td>Custom caption of the button. If it's not set, the default text &quot;Add new&quot; is displayed.</td>
<td>&quot;Add new article&quot;</td>
</tr>
</tbody>
</table>

Design

You can modify the look of the control using the following CSS styles:

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMSEditModeButtonAdd</td>
<td>CSS style of the &lt;A&gt; tag.</td>
</tr>
</tbody>
</table>

Example

You only need to add the following code inside your ASPX page or web part:

[C#], [VB.NET]

```csharp
<cc1:CMSEditModeButtonAdd ID="CMSEditModeButtonAdd1" runat="server" ClassName="cms.article" />
```

The displayed button looks like this:

Add new
1.6.23 **CMSEditModeButtonEditDelete**

The CMSEditModeButtonEditDelete control displays a button that is shown in the editing mode and allows content editors to edit or delete a document when they click it. It provides an intuitive way of editing/deleting documents.

### Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>Alias path of the document to be edited/deleted.</td>
<td>/whitepapers/myfirstpaper</td>
</tr>
<tr>
<td>EditText</td>
<td>Custom caption of the button. If not set, the default text &quot;Edit&quot; is displayed.</td>
<td>&quot;Edit article&quot;</td>
</tr>
<tr>
<td>DeleteText</td>
<td>Custom caption of the button. If not set, the default text &quot;Delete&quot; is displayed.</td>
<td>&quot;Delete article&quot;</td>
</tr>
</tbody>
</table>

### Design

You can modify the look of the control using the following CSS styles:

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMSEditModeButtonEdit</td>
<td>CSS style of the &lt;A&gt; tag.</td>
</tr>
<tr>
<td>CMSEditModeButtonDelete</td>
<td>CSS style of the &lt;A&gt; tag.</td>
</tr>
</tbody>
</table>

### Example

You only need to add the following code inside your ASPX page or web part:

[C#], [VB.NET]

```csharp
<cc1:CMSEditModeButtonEditDelete runat="server" id="btnEditDelete" Path="/news/news1" />
```

If you want to display the buttons in the transformation, you can use the following code:

[C#], [VB.NET]

```csharp
<cc1:CMSEditModeButtonEditDelete runat="server" id="btnEditDelete" Path='@Eval("NodeAliasPath") />'
```

The displayed button looks like this:

![Edit](Edit) ![Delete](Delete)
**1.6.24 CMSEditableImage**

**Please note:**

Please note: This control is compatible only with ASPX page templates. On portal pages, use web part Editable image instead of this control.

The **Editable image** web part displays an editable region that enables content editors to choose an image.

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageWidth</td>
<td>Image width in pixels - the image will be resized to this width.</td>
<td>400</td>
</tr>
<tr>
<td>ImageHeight</td>
<td>Image height in pixels - the image will be resized to this height.</td>
<td>300</td>
</tr>
<tr>
<td>ImageTitle</td>
<td>Title displayed in the editing mode.</td>
<td>&quot;Main section image&quot;</td>
</tr>
<tr>
<td>AlternateText</td>
<td>ALT text of the image displayed on the website.</td>
<td>&quot;Main image&quot;</td>
</tr>
<tr>
<td>DisplaySelectorTextBox</td>
<td>Indicates if textbox with image path should be displayed in the editing mode.</td>
<td></td>
</tr>
</tbody>
</table>

**1.6.25 CMSEditableRegion**
Please note:
This control is compatible only with ASPX page templates. On portal pages, use web part Editable text instead of this control.

The CMSEditableRegion control defines the part of the page that should be editable. It ensures displaying in both edit and view mode. The flow of data to/from CMSEditableRegion control is managed by CMSPageManager control. There must be just one CMSPageManager control and any number of CMSEditableRegion controls.

Data Source

Data is set/read by the CMSPageManager control. The CMSEditableRegion control doesn't communicate with database directly.

Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DialogHeight</td>
<td>Height of the control.</td>
<td>400</td>
</tr>
<tr>
<td>DialogWidth</td>
<td>Width of the control.</td>
<td>500</td>
</tr>
<tr>
<td>HtmlAreaToolbar</td>
<td>HTML editor toolbar set name.</td>
<td>&quot;default&quot;</td>
</tr>
<tr>
<td>HtmlAreaToolbarLocation</td>
<td>HTML editor toolbar location.</td>
<td>&quot;In&quot; for inline,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Out:FCKToolbar&quot; for shared</td>
</tr>
<tr>
<td>MaxLength</td>
<td>Maximum length of the content (in number of characters).</td>
<td>10</td>
</tr>
<tr>
<td>MinLength</td>
<td>Minimum length of the content (in number of characters).</td>
<td>2</td>
</tr>
<tr>
<td>RegionTitle</td>
<td>Control title which is displayed in the editable mode.</td>
<td>&quot;Main text&quot;</td>
</tr>
<tr>
<td>RegionType</td>
<td>Type of server control which is displayed in the editable region. It can be a textbox, textarea of HTML editor.</td>
<td>CMSEditableRegionTypeEnum.TextBox</td>
</tr>
<tr>
<td>WordWrap</td>
<td>Wrap the text if using text area field.</td>
<td></td>
</tr>
<tr>
<td>InheritContent</td>
<td>Indicates if content of the editable region should be inherited from the parent page (menu item) document.</td>
<td></td>
</tr>
</tbody>
</table>

Please note: the other properties are used by the CMSPageManager control only and shouldn't be modified in your code.
CSS Styles

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMSEditableRegionEdit</td>
<td>Style of the main &lt;TABLE&gt; element.</td>
</tr>
<tr>
<td>CMSEditableRegionTitle</td>
<td>Style of the &lt;TD&gt; element containing the title.</td>
</tr>
<tr>
<td>CMSEditableRegionError</td>
<td>Style of the &lt;TD&gt; element containing the error message.</td>
</tr>
</tbody>
</table>

1.6.26 CMSListMenu

CMSListMenu control allows you to create a large variety of menus. It renders <UL> and <LI> tags and the design depends only on your CSS style sheet. This menu control provides several advantages:

- It's based only on CSS styles which makes it highly configurable.
- It renders shorter HTML code than CMSMenu.
- It's fully XHTML compliant.
- The list-based menu is better accessible.
- You can create drop-down menu using the list-based menu and CSS almost without any JavaScript.
- It automatically displays standard UL/LI listing with links if the browser does not support CSS styles so that the user can still navigate on the web site.

However, it requires advanced knowledge of CSS as it doesn't render any specific layout by itself.

It allows you to display part of the CMS content specified using its path, depth, document type and WHERE condition.

See also: Using macro expressions in menu items, Using the CSSPrefix property for design of sub-menus

Inherits: CMSMenuProperties - common properties
### Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DisplayHighlightedItemAsLink</td>
<td>Indicates if the highlighted item should be displayed as a link.</td>
<td></td>
</tr>
<tr>
<td>DisplayOnlySelectedPath</td>
<td>Specifies whether all submenus should be displayed or just submenu under highlighted (selected) item.</td>
<td></td>
</tr>
<tr>
<td>FirstItemCssClass</td>
<td>Specifies CSS class for the first item in every menu level.</td>
<td>&quot;ListMenuFirstItem&quot;</td>
</tr>
<tr>
<td>HighlightAllItemsInPath</td>
<td>Indicates if all items in the unfolded path should be displayed as highlighted.</td>
<td></td>
</tr>
<tr>
<td>HighlightedNodePath</td>
<td>Path of the item that will be highlighted like it was selected. The path type must be the same as PathType. If you omit this value, the control automatically uses the current alias path from the &quot;aliaspath&quot; querystring parameter.</td>
<td>&quot;/products/nokia&quot;</td>
</tr>
<tr>
<td>LastItemCssClass</td>
<td>Specifies CSS class for the last item in every menu level.</td>
<td>&quot;ListMenuLastItem&quot;</td>
</tr>
<tr>
<td>OnMouseOutScript</td>
<td>OnMouseOutScript script for menu items. You can use macro expressions here.</td>
<td>&quot;alert(this.innerText);&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;alert('{%nodealiaspath%}')&quot;</td>
</tr>
<tr>
<td>OnMouseOverScript</td>
<td>OnMouseOver script for menu items. You can use macro expressions here.</td>
<td>&quot;alert(this.innerText);&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;alert('{%nodealiaspath%}')&quot;</td>
</tr>
<tr>
<td>RenderCssClasses</td>
<td>Indicates if CSS classes should be rendered for every element. If set to false, only first and last item of menu level will use CSS class.</td>
<td></td>
</tr>
<tr>
<td>RenderItemID</td>
<td>Indicates if unique ID should be rendered for every menu item.</td>
<td></td>
</tr>
<tr>
<td>ItemIdPrefix</td>
<td>Prefix placed before each item ID. You can use it to keep ID's unique if you have several CSS list menu web parts on the same page.</td>
<td>&quot;submenu&quot;</td>
</tr>
<tr>
<td>HoverCSSClassName</td>
<td>Name of the surrounding CSS class that is used to define styles for the hover effect if you want to render a drop-down menu.</td>
<td>Horizontal</td>
</tr>
<tr>
<td>UrlTarget</td>
<td>Specifies target frame for all URLs.</td>
<td>&quot;_blank&quot;</td>
</tr>
<tr>
<td>RenderLinkTitle</td>
<td>Specifies if document name should be rendered as a TITLE tag of the link (for better accessibility).</td>
<td></td>
</tr>
<tr>
<td>RenderImageAlt</td>
<td>Indicates if ALT attribute should be rendered for images used in the menu (for XHTML compatibility).</td>
<td></td>
</tr>
<tr>
<td>LoadDataAutomatically</td>
<td>Indicates if data for the control should be loaded automatically. By default, the data is</td>
<td></td>
</tr>
</tbody>
</table>
Methods

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReloadData</td>
<td>Reloads the data. If the <strong>forceLoad</strong> parameter is set to false and the custom value is assigned to the <strong>DataSource</strong> property, the properties of the <strong>CMSListMenu</strong> control are not used and only the data from the <strong>DataSource</strong> are used.</td>
</tr>
</tbody>
</table>

Design

You can modify the design using the following CSS styles if the **RenderCssClasses** property is set to true:

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMSListMenuUL</td>
<td>UL element style</td>
</tr>
<tr>
<td>CMSListMenuLI</td>
<td>LI element style</td>
</tr>
<tr>
<td>CMSListMenuLink</td>
<td>A element style</td>
</tr>
<tr>
<td>CMSListMenuHighlightedLI</td>
<td>LI element style of a highlighted item</td>
</tr>
</tbody>
</table>

See also Using the **CSSPrefix** property for design of sub-menus to find out how to set different styles for particular menu levels.

Examples

**Simple menu without styles**

This example will show you how to display a simple menu based on the CMS content. Then, you will see how different CSS styles can render different menus.

- Create a new Web form.
- Drag and drop the CMSListMenu control on the form.
- Switch to the HTML mode and add the following line at the beginning of the page:
Creating a Horizontal Drop-Down Menu Using CSS Styles

Now we will modify the CSS style of the menu so that it displays a horizontal drop-down menu.

- Open the web page in VS.NET, in HTML mode and add the following style definition inside the `<HEAD>` element:

```html
<style type="text/css" media="screen">
  .Horizontal { border-right: #c2c2c2 1px solid; border-top: #c2c2c2 1px solid; font-size: 12px; font-family: Arial; background-color: #e2e2e2; display: block; text-decoration: none; }
  .Horizontal ul { padding-right: 0px; padding-left: 0px; padding-bottom: 0px; margin: 0px; width: 100%; padding-top: 0px; list-style-type: none; }
  .Horizontal li { border-right: #e2e2e2 1px solid; padding-right: 0px; border-top: #e2e2e2 1px solid; display: inline; padding-left: 0px; float: left; padding-bottom: 0px; border-left: #e2e2e2 1px solid; padding-top: 0px; border-bottom: #e2e2e2 1px solid; }
  .Horizontal a { padding-right: 3px; display: block; padding-left: 3px; padding-bottom: 2px; margin: 0px; width: 112px; color: black; padding-top: 2px; background-color: #e2e2e2; text-decoration: none; }
  .Horizontal ul ul { z-index: 500; width: 120px; border-bottom: #c2c2c2 2px solid; position: absolute; }
  .Horizontal ul ul li { clear: left; display: block; position: relative; }
  .Horizontal ul ul ul { border-right: #c2c2c2 2px solid; left: 100%; border-bottom: white 0px solid; top: -1px; }
  .Horizontal ul ul li { display: none; }
  .Horizontal ul li:hover ul ul { display: block; }
  .Horizontal ul ul ul li:hover ul { display: block; }
  .Horizontal ul ul ul li:hover ul { display: block; }
</style>
```

- Set the HoverCSSClassName property value to: Horizontal
Creating a Vertical Drop-Down Menu Using CSS Styles

Now we will modify the CSS style of the menu so that it displays a vertical drop-down menu.

- Open the web page in VS.NET, in HTML mode and add the following style definition inside the <HEAD> element:

```html
<STYLE type="text/css" media="screen">
.Vertical { BORDER-RIGHT: #c2c2c2 1px solid; BORDER-TOP: #c2c2c2 1px solid; FONT-SIZE: 12px; BORDER-LEFT: #c2c2c2 1px solid; WIDTH: 150px; BORDER-BOTTOM: #c2c2c2 1px solid; FONT-FAMILY: Arial; BACKGROUND-COLOR: #e2e2e2; }
.Vertical UL { PADDING-RIGHT: 0px; PADDING-LEFT: 0px; PADDING-BOTTOM: 0px; MARGIN: 0px; PADDING-TOP: 0px; LIST-LIST-STYLE-TYPE: none; }
.Vertical LI { POSITION: relative; FLOAT: left; WIDTH: 100%; }
.Vertical A { PADDING-RIGHT: 0px; BACKGROUND-POSITION: 0px 50%; DISPLAY: block; PADDING-LEFT: 10px; PADDING-BOTTOM: 2px; MARGIN: 0px; WIDTH: 140px; COLOR: black; PADDING-TOP: 2px; BACKGROUND-REPEAT: no-repeat; BACKGROUND-COLOR: #e2e2e2; TEXT-DECORATION: none; }
.Vertical A:hover { BACKGROUND: #808080 no-repeat 0px 50%; COLOR: white; }
.Vertical UL { BORDER-RIGHT: #c2c2c2 1px solid; BORDER-TOP: #c2c2c2 1px solid; Z-INDEX: 100; LEFT: 100%; BORDER-LEFT: #c2c2c2 1px solid; WIDTH: 100%; BORDER-BOTTOM: #c2c2c2 1px solid; POSITION: absolute; TOP: -1px; }
#Vertical1 UL { DISPLAY: none; }
#Vertical1 LI:hover UL UL { DISPLAY: none; }
#Vertical1 UL LI:hover UL UL { DISPLAY: none; }
#Vertical1 LI:hover UL { DISPLAY: block; }
#Vertical1 UL LI:hover UL { DISPLAY: block; }
#Vertical1 UL UL LI:hover UL { DISPLAY: block; }
</STYLE>
```

- Set the HoverCSSClassName property value to: **Vertical**

- Save the page and display it in the web browser. You should see a menu like this:
### CMSMenu

The CMSMenu control allows you to display multi-level DHTML menu based on data from Kentico CMS. It encapsulates the skmMenu control. This is a free menu control for ASP.NET that can be downloaded from http://skmmenu.com.

It allows you to display part of the CMS content specified using its path, depth, document type and WHERE condition. The items are sorted by MenuItemOrder and MenuItemCaption values.

**See also:** Using macro expressions in menu items, Using the CSSPrefix property for design of sub-menus

**Inherits:** CMSMenuProperties - common properties

### Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MouseCursor</td>
<td>Represents Cursor for menu (skmMenu) control.</td>
<td>MouseCursor.Default MouseCursor</td>
</tr>
<tr>
<td>ExternalScriptPath</td>
<td>Path of the external .js file with skmMenu scripts. The default path is ~/cmsscripts/skmmenu.js.</td>
<td>&quot;~/myscripts/skmmenu.js&quot;</td>
</tr>
<tr>
<td>HighlightedMenuItem</td>
<td>Highlighted menu item. You can use it to set its style.</td>
<td></td>
</tr>
<tr>
<td>HighlightedNodePath</td>
<td>Path of the item that will be highlighted like it was selected.</td>
<td>&quot;/products/nokia&quot;</td>
</tr>
<tr>
<td>Layout</td>
<td>Represents Layout for menu (skmMenu) control.</td>
<td>MenuLayout.Vertical MenuLayout.Horizontal</td>
</tr>
<tr>
<td>MenuControl</td>
<td>Menu (skmMenu) control that ensures rendering.</td>
<td></td>
</tr>
<tr>
<td>RenderItemName</td>
<td>Indicates if ItemName attribute should be rendered.</td>
<td></td>
</tr>
<tr>
<td>Padding</td>
<td>CMSMenu table padding.</td>
<td>1</td>
</tr>
<tr>
<td>Spacing</td>
<td>CMSMenu table spacing.</td>
<td>2</td>
</tr>
<tr>
<td>SeparatorText</td>
<td>Text of the separator placed between menu items of the first menu level.</td>
<td>&quot;</td>
</tr>
<tr>
<td>SeparatorCssClass</td>
<td>CSS class of the separator cell (TD element).</td>
<td></td>
</tr>
<tr>
<td>RenderImageAlt</td>
<td>Indicates if ALT attribute should be rendered for images used in the menu (for XHTML compatibility).</td>
<td></td>
</tr>
<tr>
<td>EnableMouseUpDownClass</td>
<td>Indicates if the menu should render different CSS classes for mouse-up and mouse-down events.</td>
<td></td>
</tr>
</tbody>
</table>

### Design
You can modify the design using the following CMSMenu properties.

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMSMenu.MenuControl.CssClass</td>
<td>Style of the whole control.</td>
</tr>
<tr>
<td>CMSMenu.MenuControl.DefaultCssClass</td>
<td>Default style of menu item. This style is applied unless you specify another style for particular (highlighted) item.</td>
</tr>
<tr>
<td>CMSMenu.MenuControl.DefaultMouseDownCssClass</td>
<td>Default style of the menu item when it's clicked.</td>
</tr>
<tr>
<td>CMSMenu.MenuControl.DefaultMouseOverCssClass</td>
<td>Default style of the menu item when you move mouse over it.</td>
</tr>
<tr>
<td>CMSMenu.MenuControl.DefaultMouseUpCssClass</td>
<td>Default style of the menu item when mouse up event occurs.</td>
</tr>
<tr>
<td>CMSMenu.HighlightedMenuItem.CssClass</td>
<td>Style of the highlighted item.</td>
</tr>
<tr>
<td>CMSMenu.HighlightedMenuItem.MouseDownCssClass</td>
<td>Style of the highlighted item when it's clicked.</td>
</tr>
<tr>
<td>CMSMenu.HighlightedMenuItem.MouseOverCssClass</td>
<td>Style of the highlighted item when you move mouse over it.</td>
</tr>
<tr>
<td>CMSMenu.HighlightedMenuItem.MouseUpCssClass</td>
<td>Style of the highlighted item when mouse up event occurs.</td>
</tr>
<tr>
<td>CMSMenu.MenuControl.DefaultTarget</td>
<td>This is rather a &quot;control behavior&quot;, but it's an important property - it allows you to specify the target frame of the menu item links.</td>
</tr>
</tbody>
</table>

You can also use the following **CSS classes**:

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMSMenu</td>
<td>CSS class of the menu table.</td>
</tr>
<tr>
<td>CMSMenuItem</td>
<td>CSS class of the menu item.</td>
</tr>
<tr>
<td>CMSMenuItemMouseDown</td>
<td>CSS class of the menu item when mouse button is down.</td>
</tr>
<tr>
<td>CMSMenuItemMouseOver</td>
<td>CSS class of the menu item when user moves mouse cursor over the menu item.</td>
</tr>
<tr>
<td>CMSMenuItemMouseUp</td>
<td>CSS class of the menu item when mouse button is released.</td>
</tr>
<tr>
<td>CMSMenuHighlightedMenuItem</td>
<td>CSS class of the highlighted menu item.</td>
</tr>
<tr>
<td>CMSMenuHighlightedMenuItemMouseDown</td>
<td>CSS class of the highlighted menu item when mouse button is down.</td>
</tr>
<tr>
<td>CMSMenuHighlightedMenuItemMouseOver</td>
<td>CSS class of the highlighted menu item when user moves mouse cursor over the menu item.</td>
</tr>
<tr>
<td>CMSMenuHighlightedMenuItemMouseUp</td>
<td>CSS class of the highlighted menu item when mouse button is released.</td>
</tr>
</tbody>
</table>

See also: Using the CSSPrefix property for design of sub-menus

**Example**
This example will show you how to display a simple DHTML menu based on the CMS content. It assumes that you have configured your project for Kentico CMS Controls.

- Create a new Web form.
- Drag and drop the CMSMenu control on the form.
- In the HTML mode, add the following CSS styles inside the <BODY> tag. It will modify the appearance of the menu.

[C#], [VB.NET]

```html
<style type="text/css">
/* horizontal menu - main menu and sub-menu*/
.MainCMSMenu { BORDER-RIGHT: 0px; TABLE-LAYOUT: fixed; BORDER-TOP: 0px; BORDER-LEFT: 0px; WIDTH: 100px; BORDER-BOTTOM: 0px; BACKGROUND-COLOR: #b8ba7e; }
.MainCMSMenuItem { PADDING-RIGHT: 15px; PADDING-LEFT: 5px; FONT-SIZE: 10pt; FONT-FAMILY: verdana; }
.MainCMSMenuItemMouseUp { PADDING-RIGHT: 15px; PADDING-LEFT: 5px; FONT-SIZE: 10pt; }
.PADDING-BOTTOM: 2px; WIDTH: 100px; COLOR: black; PADDING-TOP: 2px; FONT-FAMILY: verdana; }
.MainCMSMenuItemMouseOver { PADDING-RIGHT: 15px; PADDING-LEFT: 5px; FONT-SIZE: 10pt; }
.PADDING-BOTTOM: 2px; WIDTH: 100px; COLOR: white; PADDING-TOP: 2px; FONT-FAMILY: verdana; BACKGROUND-COLOR: #ff7315; }
.MainCMSMenuItemMouseDown { PADDING-RIGHT: 15px; PADDING-LEFT: 5px; FONT-SIZE: 10pt; }
.PADDING-BOTTOM: 2px; WIDTH: 100px; COLOR: black; PADDING-TOP: 2px; FONT-FAMILY: verdana; BACKGROUND-COLOR: #ff7315; }
.MainCMSMenuItemHighlightedMenuItemMouseUp { PADDING-RIGHT: 15px; PADDING-LEFT: 5px; FONT-SIZE: 10pt; }
.PADDING-BOTTOM: 2px; WIDTH: 100px; COLOR: black; PADDING-TOP: 2px; FONT-FAMILY: verdana; BACKGROUND-COLOR: #ff7315; }
.MainCMSMenuItemHighlightedMenuItemMouseOver { PADDING-RIGHT: 15px; PADDING-LEFT: 5px; FONT-SIZE: 10pt; }
.PADDING-BOTTOM: 2px; WIDTH: 100px; COLOR: black; PADDING-TOP: 2px; FONT-FAMILY: verdana; BACKGROUND-COLOR: #ff7315; }
.MainCMSMenuItemHighlightedMenuItemMouseDown { PADDING-RIGHT: 15px; PADDING-LEFT: 5px; FONT-SIZE: 10pt; }
.PADDING-BOTTOM: 2px; WIDTH: 100px; COLOR: black; PADDING-TOP: 2px; FONT-FAMILY: verdana; BACKGROUND-COLOR: #ff7315; }
.MainCMSMenuItemHighlightedMenuItem { PADDING-RIGHT: 15px; PADDING-LEFT: 5px; FONT-SIZE: 10pt; }
.PADDING-BOTTOM: 2px; WIDTH: 100px; COLOR: black; PADDING-TOP: 2px; FONT-FAMILY: verdana; BACKGROUND-COLOR: #ff7315; }
.SubCMSMenu { PADDING-RIGHT: 15px; PADDING-LEFT: 5px; FONT-SIZE: 10pt; }
.PADDING-BOTTOM: 2px; WIDTH: 100px; COLOR: black; PADDING-TOP: 2px; FONT-FAMILY: verdana; BACKGROUND-COLOR: #ff7315; }
.SubCMSMenuMouseUp { PADDING-RIGHT: 15px; PADDING-LEFT: 5px; FONT-SIZE: 10pt; }
.PADDING-BOTTOM: 2px; WIDTH: 100px; COLOR: black; PADDING-TOP: 2px; FONT-FAMILY: verdana; BACKGROUND-COLOR: #ff7315; }
.SubCMSMenuMouseOver { PADDING-RIGHT: 15px; PADDING-LEFT: 5px; FONT-SIZE: 10pt; }
.PADDING-BOTTOM: 2px; WIDTH: 100px; COLOR: black; PADDING-TOP: 2px; FONT-FAMILY: verdana; BACKGROUND-COLOR: #ff7315; }
.SubCMSMenuMouseDown { PADDING-RIGHT: 15px; PADDING-LEFT: 5px; FONT-SIZE: 10pt; }
.PADDING-BOTTOM: 2px; WIDTH: 100px; COLOR: black; PADDING-TOP: 2px; FONT-FAMILY: verdana; BACKGROUND-COLOR: #ff7315; }
.SubCMSMenuHighlightedMenuItem { PADDING-RIGHT: 15px; PADDING-LEFT: 5px; FONT-SIZE: 10pt; }
.PADDING-BOTTOM: 2px; WIDTH: 100px; COLOR: black; PADDING-TOP: 2px; FONT-FAMILY: verdana; BACKGROUND-COLOR: #ff7315; }
.SubCMSMenuHighlightedMenuItemMouseUp { PADDING-RIGHT: 15px; PADDING-LEFT: 5px; FONT-SIZE: 10pt; }
.PADDING-BOTTOM: 2px; WIDTH: 100px; COLOR: black; PADDING-TOP: 2px; FONT-FAMILY: verdana; BACKGROUND-COLOR: #ff7315; }
.SubCMSMenuHighlightedMenuItemMouseOver { PADDING-RIGHT: 15px; PADDING-LEFT: 5px; FONT-SIZE: 10pt; }
.PADDING-BOTTOM: 2px; WIDTH: 100px; COLOR: black; PADDING-TOP: 2px; FONT-FAMILY: verdana; BACKGROUND-COLOR: #ff7315; }
.SubCMSMenuHighlightedMenuItemMouseDown { PADDING-RIGHT: 15px; PADDING-LEFT: 5px; FONT-SIZE: 10pt; }
.PADDING-BOTTOM: 2px; WIDTH: 100px; COLOR: black; PADDING-TOP: 2px; FONT-FAMILY: verdana; BACKGROUND-COLOR: #ff7315; }
/* specifying different styles for particular items */
.MainCMSMenuMouseOver#CMSMenu1-000 { BACKGROUND-COLOR: #4a3c8c; }
.MainCMSMenuItemMouseOver#CMSMenu1-001 { BACKGROUND-COLOR: #5a4c9c; }
.MainCMSMenuItemMouseOver#CMSMenu1-002 { BACKGROUND-COLOR: #6a5cac; }
.MainCMSMenuItemMouseOver#CMSMenu1-003 { BACKGROUND-COLOR: #7a6cbc; }
.MainCMSMenuItemMouseOver#CMSMenu1-004 { BACKGROUND-COLOR: #8a7ccc; }
.MainCMSMenuItemMouseOver#CMSMenu1-005 { BACKGROUND-COLOR: #9a8cdc; }
/* vertical menu - all menu levels */
.CMSMenu { BORDER-RIGHT: 0px; TABLE-LAYOUT: fixed; BORDER-TOP: 0px; BORDER-LEFT: 0px; }
</style>
```
• Switch back to the Design mode.
• In the Properties window, set the following property values:
  - Path: %

• Run the project. You should see a page like this:
1.6.28 CMSPageManager

Please note:
This control is compatible only with ASPX page templates, do not use it on Portal engine templates. For portal pages, use Editable text and editable image web parts instead of CMSEditableRegion and CMSEditableImage controls.

The CMSPageManager control is used for pages with editable regions. It ensures loading and saving content to the database. It also displays the “Save” dialog. CMSPageManager manages the flow of data to/from CMSEditableRegions.

Data Source
Content is loaded from the nearest Page (menu item) document in the alias path specified through URL or through the DefaultPageAliasPath property. The content is stored in the following format:

```xml
<content>
  <region id="ID of the CMSEditableRegion control related to this content section">
    <![CDATA[ content of the editable region ]]> 
  </region>
  <region id="...">
    <![CDATA[ content of the editable region ]]> 
  </region>
</content>
```
## Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CacheItemName</td>
<td>Name of the cache item the control will use. By setting this name dynamically, you can achieve caching based on URL parameter or some other variable - simply put the value of the parameter to the CacheItemName property. If no value is set, the control stores its content to the item named &quot;URL</td>
<td>ControlID&quot;.</td>
</tr>
<tr>
<td>CacheMinutes</td>
<td>Number of minutes the retrieved content is cached for. Zero indicates that the content will not be cached. -1 indicates the site-level settings should be used. This parameter allows you to set up caching of content so that it's not retrieved from the database each time a user requests the page.</td>
<td>10</td>
</tr>
<tr>
<td>CheckPermissionsForUserID</td>
<td>Allows you to specify the UserID of the current user. If the value is 0 (default value) no permissions are checked. Otherwise, only nodes for which the user has read permission are displayed.</td>
<td>126</td>
</tr>
<tr>
<td>CmsDeskPath</td>
<td>CMS Desk virtual path. If no value is provided, it's read from CMSDeskVirtualPath web.config parameter or it uses default value &quot;~/cmsdesk&quot;.</td>
<td>&quot;~/cmsdesk&quot;</td>
</tr>
<tr>
<td>CombineWithDefaultCulture</td>
<td>Indicates if the results should be combined with default language versions in case the translated version is not available. This property is applied only if you do not set TreeProvider property manually.</td>
<td>False</td>
</tr>
<tr>
<td>CurrentPageAliasPath</td>
<td>Alias path that is used for retrieving editable regions content.</td>
<td>&quot;~/products/category1&quot;</td>
</tr>
<tr>
<td>DefaultPageAliasPath</td>
<td>Default path that is used if no alias path is provided in the query string or through friendly URL.</td>
<td>&quot;~/home&quot;</td>
</tr>
<tr>
<td>PreferredCultureCode</td>
<td>Code of the preferred culture (en-us, fr-fr, etc.). If it's not specified, it is read from the CMSPreferredCulture session variable and then from the CMSDefaultCultureCode configuration key.</td>
<td>&quot;en-us&quot;</td>
</tr>
<tr>
<td>TreeProvider</td>
<td>Tree provider instance. If it's not provided, it's created automatically.</td>
<td></td>
</tr>
</tbody>
</table>

### CSS Styles
<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMSPageManagerError</td>
<td>Style of the error label.</td>
</tr>
<tr>
<td>CMSPageManagerLabel</td>
<td>Style of the standard label.</td>
</tr>
<tr>
<td>CMSPageManagerTextLink</td>
<td>Style of the link.</td>
</tr>
<tr>
<td>CMSPageManagerTDLabel</td>
<td>Style of the TD element that contains text with save confirmation message.</td>
</tr>
</tbody>
</table>

### 1.6.29 CMSRepeater

The CMSRepeater control inherits from the BasicRepeater control. It allows you to display part of the CMS content specified using its path, depth, document template, WHERE condition and ORDER BY clause. The CMSRepeater control displays content without writing any additional code.

**Inherits:** BasicRepeater, CMSDataProperties - common properties  
**See also:** DataPager

#### Data Source

Data retrieved using the SelectDocuments query of the specified document type.

#### Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlternatingTransformationName</td>
<td>Transformation name in format application.class.transformation applied to alternating items.</td>
<td>&quot;cms.news.previewalternate&quot;</td>
</tr>
<tr>
<td>ItemSeparator</td>
<td>Item separator between displayed records.</td>
<td>&quot;&lt;hr/&gt;&quot;</td>
</tr>
<tr>
<td>PagerControl</td>
<td>DataPager control used for paging.</td>
<td></td>
</tr>
<tr>
<td>TransformationName</td>
<td>Transformation name in format application.class.transformation.</td>
<td>&quot;cms.news.preview&quot;</td>
</tr>
<tr>
<td>NestedControlsID</td>
<td>IDs of the nested controls (CMSRepeater, CMSDataList), separated by semicolons.</td>
<td>&quot;CMSRepeaterNested; CMSDataListNested&quot;</td>
</tr>
</tbody>
</table>

#### Design

The design can be modified using the transformations.
Example

This example will show you how to read a list of news and display it using the repeater.

- Create a new Web form.
- Drag and drop the **CMSRepeater** control on the form.
- Switch to the HTML edit mode and add the following line at the beginning of the page:
- In the Properties window set the following property values:
  - ClassNames: cms.news
  - OrderBy: NewsReleaseDate DESC
  - Path: /%
  - SelectedItemTransformationName: cms.news.default
  - TransformationName: cms.news.preview

- Compile and run the project. You should see a page like this:

  Your second news  (8/4/2006)

  Summary comes here.

  Your first news  (8/3/2006)

  Summary comes here.

Displaying a nested (hierarchical) repeater/datalist

This example explains how you can display a hierarchical repeater/datalist. The hierarchical repeater consists of the main repeater and the nested repeater. You can use it for example to display a list of product categories and a preview of products in each category. You can combine nested repeaters and datalists as you need.

You can use this approach for both server controls and web parts.

- Go to Site Manager -> Development -> Document types -> **Page (menu item)** -> Transformations and define the following ASCX transformation with name **Category**:

```html
<h1><%# Eval("DocumentName") %></h1>
<p>
<cc1:CMSDataList ID="CMSDataList1" runat="server" ClassNames="cms.pda;cms.cellphone;cms.laptop"
TransformationName="cms.product.preview" RepeatColumns="2" >
</p>
</cc1:CMSDataList>
</p>
```

The transformation contains the nested datalist control that displays the documents of type **Product** using the transformation **cms.product.preview**. Please note that the path is not specified - it's supplied dynamically by the parent repeater control/web part.

- Go to Site Manager -> Development -> Document types -> **Product** -> Transformations and change the **Preview** transformation the following way:

```html
<div class="ProductPreview">
  <div class="ProductBox">
    <div class="ProductTitle">
```

© 2006-2008 Kentico Software
• Add the **Repeater** web part to your page and set its following properties:
  
  · Path: /Products/%
  · Document types: cms.menuitem
  · Transformation: CMS.MenuItem.Category
  · Nested controls ID: CMSDataList1

  If you need to dynamically set properties of a nested control, you have to set its DelayedLoading property to 'True'. Please note that this setting can cause problems with ViewState.

• Go to the live site. You will see a page like this:

  **Laptops**

  ![Laptop 1](image1.png)  ![Laptop 2](image2.png)

  **PDA**

  ![PDA 1](image3.png)  ![PDA 2](image4.png)
1.6.30 CMSSearchDialog, CMSSearchResults

The CMSSearchDialog control allows users to enter searched words. The user can also (optionally) specify the search scope (where to search) and search mode (how to search).

CMSSearchDialog can be easily used with CMSSearchResults control that displays the search results according to the provided parameters from the CMSSearchDialog.

Note: Both controls can be used separately. Also, you can receive search results using the CMS.TreeProvider.Search() method.

Data Source

The CMSSearchDialog is not connected to any data source - it only communicates with user. The search method in the CMSSearchResults dialog uses the pre-defined queries "searchtree" in the document template definitions. It combines all results and returns them as one table.

CMSSearchDialog Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ShowSearchMode</td>
<td>Indicates if search mode settings should be displayed.</td>
<td></td>
</tr>
<tr>
<td>ShowSearchScope</td>
<td>Indicates if search scope settings should be displayed.</td>
<td></td>
</tr>
<tr>
<td>SearchExpression</td>
<td>Entered word(s) to be searched for.</td>
<td>&quot;asp.net cms&quot;</td>
</tr>
<tr>
<td>SearchMode</td>
<td>Search mode - any word, all words or exact phrase.</td>
<td>SearchModeEnum.AnyWord</td>
</tr>
<tr>
<td>SearchScope</td>
<td>Returns 0 for all content or 1 for the current section.</td>
<td>0</td>
</tr>
</tbody>
</table>

CMSSearchDialog Events

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DoSearch</td>
<td>Occurs when user submits the dialog.</td>
</tr>
</tbody>
</table>
**CMSSearchDialog Design**

The design can be modified by setting style of particular controls. All displayed controls can be accessed through the following properties:

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SearchForLabel</td>
<td>SearchFor label control.</td>
</tr>
<tr>
<td>SearchForTextBox</td>
<td>SearchFor textbox.</td>
</tr>
<tr>
<td>SearchModeLabel</td>
<td>SearchMode label.</td>
</tr>
<tr>
<td>SearchModeList</td>
<td>SearchMode drop-down list.</td>
</tr>
<tr>
<td>SearchScopeLabel</td>
<td>SearchScope label.</td>
</tr>
<tr>
<td>SearchScopeList</td>
<td>SearchScope drop-down list.</td>
</tr>
<tr>
<td>SearchButton</td>
<td>Search button.</td>
</tr>
</tbody>
</table>

You can also use the following CSS classes:

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMSSearchDialogSearchForLabel</td>
<td>CSS class of the &quot;Search for:&quot; label.</td>
</tr>
<tr>
<td>CMSSearchDialogSearchForTextBox</td>
<td>CSS class of the search expression text box.</td>
</tr>
<tr>
<td>CMSSearchDialogSearchModeLabel</td>
<td>CSS class of the &quot;Search Mode:&quot; label.</td>
</tr>
<tr>
<td>CMSSearchDialogSearchModeDropDownList</td>
<td>CSS class of the search mode drop down list.</td>
</tr>
<tr>
<td>CMSSearchDialogSearchScopeLabel</td>
<td>CSS class of the &quot;Search Scope:&quot; label.</td>
</tr>
<tr>
<td>CMSSearchDialogSearchScopeDropDownList</td>
<td>CSS class of the search scope drop down list.</td>
</tr>
<tr>
<td>CMSSearchDialogSearchButton</td>
<td>CSS class of the search button.</td>
</tr>
</tbody>
</table>
CMSSearchResults Properties

Inherits: CMSControlProperties - common properties
See also: DataPager

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PagerControl</td>
<td>DataPager control used for paging of the search results.</td>
<td></td>
</tr>
<tr>
<td>.QueryStringKey</td>
<td>Query string key used for data pager URL parameter.</td>
<td></td>
</tr>
<tr>
<td>SearchExpression</td>
<td>Word(s) to be searched for.</td>
<td>&quot;asp.net cms&quot;</td>
</tr>
<tr>
<td>SearchMode</td>
<td>Search mode - any word, all words or exact phrase.</td>
<td>SearchModeEnum.AnyWord</td>
</tr>
<tr>
<td>StartingPath</td>
<td>Starting path specifying the content to be searched.</td>
<td>&quot;/products&quot;</td>
</tr>
<tr>
<td>CMSSearchDialogID</td>
<td>Optionally, you can use this property to specify the ID of the source CMSSearchDialog control that provides search parameters.</td>
<td>&quot;CMSSearchDialog1&quot;</td>
</tr>
<tr>
<td>TransformationName</td>
<td>Transformation name in format application.class.transformation.</td>
<td>&quot;cms.searchresults&quot;</td>
</tr>
<tr>
<td></td>
<td>This transformation is used for displaying the search results.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The default transformation is &quot;cms.root.searchresults&quot;.</td>
<td></td>
</tr>
<tr>
<td>WhereCondition</td>
<td>WHERE condition used for the SQL search queries.</td>
<td>&quot; DocumentModifiedWhen &gt; '1/1/2007' &quot;</td>
</tr>
<tr>
<td>OrderBy</td>
<td>ORDER BY condition used for the SQL search queries.</td>
<td>&quot; DocumentModifiedWhen DESC &quot;</td>
</tr>
</tbody>
</table>

CMSSearchResults QueryString (URL) parameters

The CMSSearchResults control accepts the following URL parameters:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>searchtext</td>
<td>Searched text.</td>
<td>products</td>
</tr>
<tr>
<td>searchmode</td>
<td>Search mode.</td>
<td>allwords exactphrase anyword (default value)</td>
</tr>
</tbody>
</table>

CMSSearchResults Design

The search results are displayed using the transformation specified in the TransformationName property.
Example of Using CMSSearchDialog and CMSSearchResults Controls

This example will show you how to create a search form and display the search results.

- Create a new Web form.
- Drag and drop the CMSSearchDialog control on the form.
- Drag and drop the CMSSearchResults control on the form.
- Set the following properties of the CMSSearchResults control:
  - CMSSearchDialogID: CMSSearchDialog1
- Compile and run the project. You should see a page like this:

  Search for: product
  Search mode: Any Word
  Search Scope: All Content
  Products

  Acer AL1511s
  Modified when: 9/7/2006 10:38:52 AM

1.6.31 CMSSiteMap

The CMSSiteMap control allows you to display the whole navigation structure of the Web site or just its specified part. It reads CMS.MenuItem documents and renders their structure as a site map.

It allows you to display part of the menu structure specified using its path, depth, document type and WHERE condition. The items are sorted by MenuItemOrder and MenuItemCaption values.

See also: Using macro expressions in menu items

Inherits: CMSMenuProperties - common properties
## Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>UrlTarget</td>
<td>Specifies target frame for all URLs.</td>
<td>&quot;_blank&quot;</td>
</tr>
<tr>
<td>LoadDataAutomatically</td>
<td>Indicates if data for the control should be loaded automatically. By default, the data is loaded automatically. If you set this property to false, you can supply custom DataSet to the DataSource property and then call the ReloadData(false) method.</td>
<td></td>
</tr>
<tr>
<td>RenderedHTML</td>
<td>Allows you to get or set the HTML code rendered by the control. You need to set this property before the Render event - e.g. in the OnLoad event.</td>
<td></td>
</tr>
</tbody>
</table>

## Methods

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReloadData</td>
<td>Reloads the data. If the <code>forceLoad</code> parameter is set to false and the custom value is assigned to the DataSource property, the properties of the CMSListMenu control are not used and only the data from the DataSource are used.</td>
</tr>
</tbody>
</table>

## Design

You can modify the design using the following CSS styles:

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMSSiteMapList</td>
<td>The UL element in the site map.</td>
</tr>
<tr>
<td>CMSSiteMapListListItem</td>
<td>The LI element in the site map.</td>
</tr>
<tr>
<td>CMSSiteMapLink</td>
<td>Link (A element) in the site map.</td>
</tr>
</tbody>
</table>
Example

This example will show you how to display a site map based on the CMS content. It assumes that you have configured your project for Kentico CMS Controls.

- Create a new Web form.
- Drag and drop the CMSSiteMap control on the form.
- In the HTML mode, add the following CSS styles inside the <head> tag. It will modify the appearance of the menu.

```html
<style type="text/css">
  .CMSSiteMapList { }  
  .CMSSiteMapListItem { list-style-image: url(images/menuitem.gif); }  
  .CMSSiteMapLink { color: #C34C17; text-decoration:none; }
</style>
```

- Switch back to the Design mode.
- In the Properties window, set the following property values:
  - Path: /%
- Compile and run the project. You should see a page like this:

```
- Contact
- Home
- News
- Partners
- Products
  - LCD Displays
  - Notebooks
  - POAs
- Search
- Services
  - Service 1
  - Service 2
  - Overview
- Site map
```
1.6.32 CMSTabControl

The CMSTabControl control is inherited from the BasicTabControl control. It allows you to display one-level tab menu based on data from Kentico CMS. It reads specified documents and renders the menu according to their values.

It allows you to display menu items specified using their path, depth, document type and WHERE condition. The CMSTabControl control displays content without writing any additional code. The items are sorted by MenuItemOrder and MenuItemCaption values.

See also: Using macro expressions in menu items, Using the CSSPrefix property for design of sub-menus

Inherits: BasicTabControl, CMSMenuProperties - common properties

Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HighlightedNodePath</td>
<td>Path of the item that will be highlighted like it was selected.</td>
<td>&quot;products/notebooks&quot;</td>
</tr>
<tr>
<td>RenderImageAlt</td>
<td>Indicates if ALT attribute should be rendered for images used in the menu (for XHTML compatibility).</td>
<td></td>
</tr>
<tr>
<td>LoadDataAutomatically</td>
<td>Indicates if data for the control should be loaded automatically. By default, the data is loaded automatically. If you set this property to false, you can supply custom DataSet to the DataSource property and then call the ReloadData(false) method.</td>
<td></td>
</tr>
<tr>
<td>RenderedHTML</td>
<td>Allows you to get or set the HTML code rendered by the control. You need to set this property before the Render event - e.g. in the OnLoad event.</td>
<td></td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReloadData</td>
<td>Reloads the data. If the forceLoad parameter is set to false and the custom value is assigned to the DataSource property, the properties of the CMSListMenu control are not used and only the data from the DataSource are used.</td>
</tr>
</tbody>
</table>
Design

You can modify the design using the same classes as for the BasicTabControl control.

Example

This example will show you how to display a simple tab-menu based on the CMS content. It assumes that you have configured your project for Kentico CMS Controls.

- Create a new Web form.
- Drag and drop the **CMSTabControl** control on the form.
- In Source mode add the following CSS styles inside the `<BODY>` tag. It will modify the appearance of the tabs.

[C#], [VB.NET]

```html
<style type="text/css">
.TabControlTable { FONT-SIZE: 14px; FONT-FAMILY: Arial,Verdana }
.TabControlRow { } 
.TabControl { BORDER-RIGHT: black 1px solid; BORDER-LEFT: black 1px solid; CURSOR: hand; COLOR: black } 
.TabControlSelected { BORDER-RIGHT: black 1px solid; BORDER-LEFT: black 1px solid; CURSOR: default; COLOR: white } 
.TabControlLinkSelected { COLOR: white; TEXT-DECORATION: none } 
.TabControlLink { COLOR: black; TEXT-DECORATION: none } 
.TabControlLeft { WIDTH: 1px } 
.TabControlRight { WIDTH: 0px } 
.TabControlSelectedLeft { WIDTH: 1px } 
.TabControlSelectedRight { WIDTH: 0px } 
</style>
```

- Add the following table code just after the CMSTabControl tags. It will display a stripe under the tabs.

[C#], [VB.NET]

```html
<hr style="width:100%; height:2px; margin-top:0px;" />
```

- Switch back to the Design mode.
- In the **Properties** window set the following property values:
  - MaxRelativeLevel: 1
  - Path: /%
- Compile and run the project. You will see a page like this:
1.6.33 CMSTreeMenu

The **CMSTreeMenu** control allows you to display multi-level tree menu based on data from Kentico CMS. It allows you to display part of the menu structure specified using its path, depth, document type and WHERE condition.

**See also:** Using macro expressions in menu items, Using the CSSPrefix property for design of sub-menus

**Inherits:** CMSMenuProperties - common properties

**Properties**

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CellPadding</td>
<td>Cell padding of the table representing menu.</td>
<td></td>
</tr>
<tr>
<td>CellSpacing</td>
<td>Cell spacing of the table representing menu.</td>
<td></td>
</tr>
<tr>
<td>CollapseSelectedNodeOnClick</td>
<td>Indicates if the selected section of menu should be collapsed when it's clicked.</td>
<td></td>
</tr>
<tr>
<td>DisplayHighlightedItemAsLink</td>
<td>Indicates if the highlighted item should be displayed as a link.</td>
<td></td>
</tr>
<tr>
<td>GenerateIndentationInsideLink</td>
<td>Indicates if indentation spaces should be generated inside hyperlink (true) or outside (false). This applies only when you do not use images in the menu.</td>
<td></td>
</tr>
<tr>
<td>GenerateOnlyOuterLink</td>
<td>Indicates if only one outer link should be generated per each menu item.</td>
<td></td>
</tr>
<tr>
<td>HighlightAllItemsInPath</td>
<td>Indicates if all items in the currently selected path should be displayed as highlighted.</td>
<td></td>
</tr>
<tr>
<td>HighlightedNodePath</td>
<td>Path of the item that will be highlighted like it was selected. If you omit this value, the control automatically uses the current alias path from the &quot;aliaspath&quot; querystring parameter.</td>
<td>&quot;~/products&quot;</td>
</tr>
<tr>
<td>Indentation</td>
<td>Indentation of menu item levels. Number of spaces that will be placed before each level of menu items.</td>
<td></td>
</tr>
<tr>
<td>MenuItemImageUrl</td>
<td>URL address of the image that is displayed next to menu items. It may start with &quot;~/&quot; representing the virtual path of the current application.</td>
<td>&quot;~/images/menutem.gif&quot;</td>
</tr>
<tr>
<td>MenuItemOpenImageUrl</td>
<td>URL address of the image that is displayed next to open menu items. It may start with &quot;~/&quot; representing the virtual path of the current application.</td>
<td>&quot;~/images/openmenutem.gif&quot;</td>
</tr>
<tr>
<td>OnMouseOutScript</td>
<td>OnMouseOutScript script for menu items. You can use macro expressions here.</td>
<td></td>
</tr>
<tr>
<td>OnMouseOverScript</td>
<td>OnMouseOver script for menu items. You can use macro expressions here.</td>
<td></td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>UrlTarget</td>
<td>Specifies target frame for all URLs.</td>
<td></td>
</tr>
<tr>
<td>UseAlternatingStyles</td>
<td>Indicates if the control should render different styles for odd and even items.</td>
<td></td>
</tr>
<tr>
<td>RenderImageAlt</td>
<td>Indicates if ALT attribute should be rendered for images used in the menu (for XHTML compatibility).</td>
<td></td>
</tr>
<tr>
<td>LoadDataAutomaticaly</td>
<td>Indicates if data for the control should be loaded automatically. By default, the data is loaded automatically. If you set this property to false, you can supply custom DataSet to the DataSource property and then call the ReloadData(false) method.</td>
<td></td>
</tr>
<tr>
<td>RenderedHTML</td>
<td>Allows you to get or set the HTML code rendered by the control. You need to set this property before the Render event - e.g. in the OnLoad event.</td>
<td></td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReloadData</td>
<td>Reloads the data. If the <strong>forceLoad</strong> parameter is set to false and the custom value is assigned to the DataSource property, the properties of the CMSListMenu control are not used and only the data from the DataSource are used.</td>
</tr>
</tbody>
</table>
Design

You can modify the design using the following CSS styles:

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMSTreeMenuTable</td>
<td>The main table (TABLE element).</td>
</tr>
<tr>
<td>CMSTreeMenuItem</td>
<td>Tree menu item (TD element).</td>
</tr>
<tr>
<td>CMSTreeMenuItemAlt</td>
<td>Alternating style of the menu item (TD element). It's used only when you set the UseAlternatingStyles property to true.</td>
</tr>
<tr>
<td>CMSTreeMenuSelectedItem</td>
<td>Selected tree menu item (TD element).</td>
</tr>
<tr>
<td>CMSTreeMenuLink</td>
<td>Link (A element).</td>
</tr>
<tr>
<td>CMSTreeMenuLinkAlt</td>
<td>Alternating style of the link (A element). It's used only when you set the UseAlternatingStyles property to true.</td>
</tr>
<tr>
<td>CMSTreeMenuSelectedLink</td>
<td>Link of the selected item (A element).</td>
</tr>
<tr>
<td>CMSTreeMenuNestedTable</td>
<td>Nested table (TABLE element). It's used only when CollapseSelectedNodeOnClick is true.</td>
</tr>
</tbody>
</table>

See also: Using the CSSPrefix property for design of sub-menus to find out how to set different styles for particular menu levels.

Example

This example will show you how to display a tree menu based on the CMS content.

- Create a new Web form.
- Drag and drop the CMSTreeMenu control on the form.
- In the HTML mode, add the following CSS styles inside the <HEAD> element. It will modify the appearance of the menu.

<style>
.CMSTreeMenuTable { padding-right: 5px; padding-left: 5px; padding-bottom: 2px; padding-top: 2px; }
.CMSTreeMenuItem { background: #e7e7ff }
.CMSTreeMenuSelectedItem { background: #4a3c8c }
.CMSTreeMenuLink { color: black; text-decoration: none }
.CMSTreeMenuSelectedLink { color: white; text-decoration: none }
</style>

- Switch back to the Design mode.
- In the Properties window, set the following property values:
  - Path: /Products
  - MenuItemImageUrl: ~/images/menuitem.gif
  - MenuItemOpenImageUrl: ~/images/menuitem_selected.gif
  - Indentation: 3
Compile and run the project. You should see a page like this:

![CMSTreeView](attachment:CMSTreeView.png)

**1.6.34 CMSTreeView**

The CMSTreeView control allows you to display multi-level tree menu based on data from Kentico CMS. It allows you to display part of the menu structure specified using its path, depth, document type and WHERE condition.

*See also:* Using macro expressions in menu items

*Inherits:* CMSMenuProperties - common properties

This web part uses the standard ASP.NET TreeView control and enhances it with standard CMSMenuProperties set of properties. Please see the ASP.NET documentation for more details on the TreeView control properties, behavior and design.

**1.6.35 CMSViewer**

The CMSViewer control allows you to display part of the CMS content specified using its path, depth, document template, WHERE condition and ORDER BY clause. It uses XSLT transformations to display the content. The CMSViewer control displays content without writing any additional code.

*Inherits:* CMSDataProperties - common properties

**Properties**

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HideControlForZeroRows</td>
<td>Hides the control when no data is loaded. The default value is False.</td>
<td></td>
</tr>
<tr>
<td>TransformationName</td>
<td>Transformation name in format application.class.transformation.</td>
<td>&quot;cms.article.preview&quot;</td>
</tr>
<tr>
<td>ZeroRowsText</td>
<td>Text to be shown when the control doesn't display any data.</td>
<td></td>
</tr>
</tbody>
</table>

**Design**

The displayed content is completely driven by your XSLT transformation.

**Example**
This example will show you how to display specific news item using the CMSViewer control. It assumes that you have configured your project for Kentico CMS Controls.

- Create a new Web form.
- Drag and drop the CMSViewer control on the form.
- Switch back to the Design mode. In the Properties window set the following property values:
  - ClassNames: cms.news
  - Path: /News/Your-First-News
  - TransformationName: cms.news.default_xslt
- Compile and run the project. You should see a page like this:

NewsID: 1
News Title: Your first news
News Summary:

Summary comes here.

News Text: 

News text comes here.

Release Date: 2006-08-03T13:49:26+02:00
1.6.36 DataPager

The DataPager control ensures paging of some of the CMSControls:

- CMSDataList
- CMSRepeater
- CMSSearchResults
- QueryDataList
- QueryRepeater

It cannot be used separately.

See also: TemplateDataPager

Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrentPage</td>
<td>Current page index.</td>
<td></td>
</tr>
<tr>
<td>BackText</td>
<td>Back button/hyperlink text.</td>
<td></td>
</tr>
<tr>
<td>FirstText</td>
<td>First text.</td>
<td></td>
</tr>
<tr>
<td>HideOnSinglePage</td>
<td>If true, the pager is hidden if only one page is displayed.</td>
<td></td>
</tr>
<tr>
<td>IgnoreQueryString</td>
<td>Indicates if QueryString parameters should be ignored.</td>
<td></td>
</tr>
<tr>
<td>LabelText</td>
<td>Label text.</td>
<td></td>
</tr>
<tr>
<td>LastText</td>
<td>Last text.</td>
<td></td>
</tr>
<tr>
<td>MaxPages</td>
<td>Maximum number of pages to be displayed.</td>
<td></td>
</tr>
<tr>
<td>NextText</td>
<td>Next button/hyperlink text.</td>
<td></td>
</tr>
<tr>
<td>PageCount</td>
<td>Page count (read only).</td>
<td></td>
</tr>
<tr>
<td>PageNumbersSeparator</td>
<td>Page numbers separator.</td>
<td>&quot; &quot;</td>
</tr>
<tr>
<td>PageSize</td>
<td>Page size</td>
<td></td>
</tr>
<tr>
<td>PagingMode</td>
<td>Determines if PostBack or QueryString should be used for the paging.</td>
<td>PostBack QueryString</td>
</tr>
<tr>
<td>QueryStringKey</td>
<td>Query parameter name for the page index.</td>
<td>&quot;pagenumber&quot;</td>
</tr>
<tr>
<td>RecordEnd</td>
<td>Index of the last record on current page.</td>
<td></td>
</tr>
<tr>
<td>RecordStart</td>
<td>Index of the first record on current page.</td>
<td></td>
</tr>
<tr>
<td>ResultsFormat</td>
<td>Results text format.</td>
<td>&quot;Displaying results {0}-{1} (of {2})&quot;</td>
</tr>
<tr>
<td>SliderSize</td>
<td>Slider size</td>
<td></td>
</tr>
<tr>
<td>TotalRecords</td>
<td>Total data source records.</td>
<td></td>
</tr>
</tbody>
</table>
Design

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BackNextButtonStyle</td>
<td>Back/Next button style.</td>
<td></td>
</tr>
<tr>
<td>BackNextDisplay</td>
<td>Back/Next display mode.</td>
<td>Buttons Hyperlinks</td>
</tr>
<tr>
<td>BackNextLinkSeparator</td>
<td>Back/Next link separator.</td>
<td></td>
</tr>
<tr>
<td>BackNextLocation</td>
<td>Back/Next location.</td>
<td>Right Left Split None</td>
</tr>
<tr>
<td>BackNextStyle</td>
<td>Back/Next style.</td>
<td></td>
</tr>
<tr>
<td>ControlCssClass</td>
<td>CSS Class of the pager control</td>
<td></td>
</tr>
<tr>
<td>LabelStyle</td>
<td>Label style</td>
<td></td>
</tr>
<tr>
<td>PageNumbersDisplay</td>
<td>Page numbers display mode.</td>
<td>Number Results</td>
</tr>
<tr>
<td>PageNumbersStyle</td>
<td>Page numbers style.</td>
<td></td>
</tr>
<tr>
<td>PagerControlStyle</td>
<td>Pager control style.</td>
<td></td>
</tr>
<tr>
<td>PagerPosition</td>
<td>Pager position.</td>
<td>Bottom Top</td>
</tr>
<tr>
<td>ResultsLocation</td>
<td>Results location.</td>
<td>Top Bottom None</td>
</tr>
<tr>
<td>ResultsStyle</td>
<td>Results style.</td>
<td></td>
</tr>
<tr>
<td>SectionPadding</td>
<td>Section padding.</td>
<td></td>
</tr>
<tr>
<td>ShowFirstLast</td>
<td>Show first last labels.</td>
<td></td>
</tr>
<tr>
<td>ShowLabel</td>
<td>Show label.</td>
<td></td>
</tr>
<tr>
<td>ShowPageNumbers</td>
<td>Show page numbers.</td>
<td></td>
</tr>
<tr>
<td>UseSlider</td>
<td>Use slider.</td>
<td></td>
</tr>
</tbody>
</table>

1.6.37 QueryDataGrid

The QueryDataGrid control is inherited from the BasicDatagrid control. It automatically ensures data binding, paging and sorting. Moreover, it allows you to specify query and the WHERE condition and it displays data without any additional code.

You can use the common DataGrid designer to set up QueryDataGrid style and behavior.

QueryDataGrid can be only used with pre-defined queries stored in the document type configuration. If you want to display only data from particular part of the content tree, please use CMSDataGrid.

Data Source

Data retrieved using the predefined query specified in the QueryName property.

Inherits: CMSQueryProperties - common properties, BasicDataGrid
Design

The design can be modified in the same way as the standard DataGrid control.

Example

This example will show you how to read list of news and display it in the grid using the QueryDataGrid control.

- Create a new Web form.
- Drag and drop the QueryDataGrid control on the form.
- In the Properties window, click Auto Format... and choose some color schema.
- In the Properties window, click Property Builder..., the QueryDataGrid Properties dialog appears.
  - On the General tab check the Allow sorting box.
  - Now we will specify the columns that will be displayed. On the Columns tab:
    - Uncheck the Create columns automatically at run time box.
    - Add a new Bound Column from the Available columns list to the Selected columns list.
- In the Properties window set the property QueryName to cms.news.selectlatest and set ProcessSorting to True (sorting will be ensured by the control).

  Compile and run the project. You should see a page like this:

<table>
<thead>
<tr>
<th>News Title</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Your first news</td>
<td>8/3/2006 1:49:26 PM</td>
</tr>
<tr>
<td>Your second news</td>
<td>8/4/2006 1:50:42 PM</td>
</tr>
</tbody>
</table>

1.6.38 QueryDataList

The QueryDataList control is inherited from the BasicDataList control. It automatically ensures data binding. Moreover, it allows you to specify query, ORDER BY clause and the WHERE condition and it ensures displaying data without any additional code.

QueryDataList can be used only with pre-defined queries stored in the document type configuration.

If you want to display data only from particular part of the content tree, please use CMSDataList.

Data Source

Data retrieved using the predefined query specified in the QueryName property.

Inherits: CMSQueryProperties - common properties, BasicDataList
See also: DataPager

Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlternatingTransformationName</td>
<td>Transformation name in format application.class.transformation applied to alternating items.</td>
<td>&quot;cms.news.preview_alt&quot;</td>
</tr>
<tr>
<td>EnablePaging</td>
<td>Enables paging and show the DataPager control.</td>
<td></td>
</tr>
<tr>
<td>PagerControl</td>
<td>DataPager control.</td>
<td></td>
</tr>
<tr>
<td>TransformationName</td>
<td>Transformation name in format application.class.transformation.</td>
<td>&quot;cms.news.preview&quot;</td>
</tr>
</tbody>
</table>

Design

The design can be modified in the same way as for the standard ASP.NET DataList control. The design of the displayed content depends on used transformations.

Example

This example will show you how to read a list of the latest news and display them using the QueryDataList control.

- Create a new Web form.
- Drag and drop the QueryDataList control on the form.
- In the Properties window set the following property values:
  - QueryName: cms.news.selectlatest
  - RepeatColumns: 2
  - TransformationName: cms.news.preview
- Compile and run the project. You should see a page like this:

```
Summary comes here.                Summary comes here.
```

1.6.39 QueryRepeater

The QueryRepeater control is inherited from the BasicRepeater control. It automatically ensures data binding. Moreover, it allows you to specify query, ORDER BY clause and the WHERE condition and it ensures displaying data without any additional code.

QueryRepeater can be only used with pre-defined queries stored in the document type configuration.

If you want to display only data from particular part of the content tree, please use CMSRepeater instead.

Data Source
Data retrieved using the predefined query specified in the QueryName property.

**Inherits:** BasicRepeater, CMSQueryProperties - common properties  
**See also:** DataPager

## Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlternatingTransformationName</td>
<td>Transformation name in format application.class.transformation applied to alternating items.</td>
<td>&quot;cms.news.preview_alt&quot;</td>
</tr>
<tr>
<td>ItemSeparator</td>
<td>Item separator between displayed records.</td>
<td>&quot;&lt;hr/&gt;&quot;</td>
</tr>
<tr>
<td>EnablePaging</td>
<td>Enables paging a shows the DataPager control.</td>
<td></td>
</tr>
<tr>
<td>PagerControl</td>
<td>DataPager control that ensured paging.</td>
<td></td>
</tr>
<tr>
<td>TransformationName</td>
<td>Transformation name in format application.class.transformation.</td>
<td>&quot;cms.news.preview&quot;</td>
</tr>
</tbody>
</table>

## Design

The design depends on the transformations you use to display content.

## Example

This example will show you how to read list of news and display them using the QueryRepeater control.

- Create a new Web form.  
- Drag and drop the **QueryRepeater** control on the form.  
- In the Properties window set the following property values:  
  - QueryName: cms.news.selectlatest  
  - TransformationName: cms.news.preview  
- Compile and run the project. You should see a page like this:

  **Your second news** (8/4/2006)

  Summary comes here.

  **Your first news** (8/3/2006)

  Summary comes here.

### 1.6.40 TemplateDataPager

The **TemplateDataPager** control can be used for custom format of data pager. It automatically renders the list of numbers, but you need to write some code to bind it to a control that ensures
displaying of the content (e.g. CMSRepeater, CMSDataList or other).

Here's a simple example of how you can use it:

1. Add the following line to the beginning of your ASPX page:

[C#]

```csharp
<%@ Register Assembly="CMS.Controls" Namespace="CMS.Controls" TagPrefix="cc1" %>
```

2. Add the following code to the page, inside the `<form>` element:

[C#]

```csharp
<table style="border: solid 1px #CCCCCC; margin-left: auto; margin-right: auto;">
  <tr>
    <td style="border-bottom: solid 1px #CCCCCC; padding: 10px; text-align: center;">
      <cc1:CMSRepeater ID="CMSRepeater1" runat="server" Path="/" ClassNames="CMS.Product" TransformationName="CMS.Product.preview">
      </cc1:CMSRepeater>
    </td>
  </tr>
  <tr>
    <td style="padding: 10px; background-color: #D9D9D9;">
      <cc1:TemplateDataPager ID="TemplateDataPager1" runat="server">
        <NumberTemplate>
          <a href="?Page=<%# Eval("PageNumber") %>">
            <%# Eval("PageNumber") %>
          </a>
        </NumberTemplate>
        <SelectedNumberTemplate>
          <a href="?Page=<%# Eval("PageNumber") %>">
            <%# Eval("PageNumber") %>
          </a>
        </SelectedNumberTemplate>
        <SeparatorTemplate/>
        <FirstItemTemplate>
          <a href="?Page=1">First</a>
        </FirstItemTemplate>
        <LastItemTemplate>
          <a href="?Page=<%# pageCount %>">Last</a>
        </LastItemTemplate>
        <PreviousItemTemplate>
          <a href="?Page=<%# previousPage %>">Previous</a>
        </PreviousItemTemplate>
        <NextItemTemplate>
          <a href="?Page=<%# nextPage %>">Next</a>
        </NextItemTemplate>
      </cc1:TemplateDataPager>
    </td>
  </tr>
</table>
```

As you can see, the control uses the standard CMSRepeater control. The pager format is specified using the templates inside its definition.

3. Modify the code-behind so that it looks like this (the page class name and type may be different):

[C#]
using CMS.GlobalHelper;

public partial class CMSControlsExamples_TemplatedDataPager : ControlsExamplesPage
{
    public string pageCount = "1";
    public string previousPage = "1";
    public string nextPage = "";

    /// <summary>
    /// OnInit override
    /// </summary>
    /// <param name="e"></param>
    protected override void OnInit(EventArgs e)
    {
        // Disable repeater pager and databindbydefault
        CMSRepeater1.EnableFaging = false;
        CMSRepeater1.DataBindByDefault = false;
        base.OnInit(e);
    }

    protected void Page_Load(object sender, EventArgs e)
    {
        // Get repeater datasource
        TemplateDataPager1.DataSource = CMSRepeater1.DataSource;

        // Set page size
        TemplateDataPager1.PageSize = 1;

        // Set current page from query string
        TemplateDataPager1.CurrentPage = ValidationHelper.GetInteger(Request.QueryString["Page"], 1);

        // Get page number for last link
        pageCount = ((int)(TemplateDataPager1.PageCount - 1)).ToString();

        // Set default next page link
        nextPage = pageCount;

        // Set previous link
        if ((TemplateNameDataPager1.CurrentPage - 1) >= 1)
        {
            previousPage = ((int)(TemplateNameDataPager1.CurrentPage - 1)).ToString();
        }

        // Set next link
        if ((TemplateNameDataPager1.CurrentPage + 1) <= (TemplateNameDataPager1.PageCount - 1))
        {
            nextPage = ((int)(TemplateNameDataPager1.CurrentPage + 1)).ToString();
        }

        // Set paged datasource to the repeater and databind it
        CMSRepeater1.DataSource = TemplateDataPager1.PagedData;
        if (!DataSourceIsEmpty(CMSRepeater1.DataSource))
        {
            CMSRepeater1.DataBind();
        }
    }
}

4. Save all changes and see the page on the live site. It will look like this:
IPaq RX1950

our price: $ 699.00

more

First | Previous | 1 - 2 - 3 - 4 - 5 | Next | Last