

Whitepaper

Unigrid

This article gives a more advanced example on how to use the UniGrid control which is a listing control used in the user interface of Kentico.

Configuration for unigrid.

Property Name	Description	Sample Value
Columns	<p>Specifies the columns that should be loaded from the data source specified in the DataSource property.</p> <p>By default, the values of the first column are passed as the actionArgument parameter of the OnAction event handler. This can be overridden in the definition by specifying a column name in the commandargument attribute of individual <action> elements.</p>	
CompleteWhereCondition	Can be used to get the used WHERE clause including any modifications applied by the filter.	
DataSource	Can be used to gets or set an object (DataSet or DataTable) containing the data to be displayed by the UniGrid control.	
DelayedReload	If enabled, data will not be loaded automatically during the Load event of the page and the ReloadData() method must be called manually instead.	
FilterDirectoryPath	Path to the control (.ascx file) that should be used instead of the default filter. The default relative path is <i>~/CMSAdminControls/UI/UniGrid/Filters/</i> .	
FilteredZeroRowsText	Text to be shown when no rows are displayed after the filter is applied.	
FilterLimit	Determines the minimum amount of rows that must be displayed in the UniGrid before a filter is shown. The default value is read from the CMSDefaultListingFilterLimit web.config key.	
GridName	Contains the name of an external XML file that defines the structure and behaviour of the UniGrid control. For more information, please refer to the UniGrid definition topic.	
GridView	Can be used to access the GridView control encapsulated by the UniGrid.	
HideControlForZeroRows	Indicates whether the control should be hidden when no rows are loaded. The control is not hidden if the filter causes zero	

	rows to be displayed.	
ImageDirectoryPath	Path to the directory that contains images used by the control. The default value is <i>~/App_Themes/Default/Images/Design/Controls/UniGrid/Actions.</i>	
NamedColumns	<p>Gets a dictionary mapping custom names to <i>DataControlField</i> objects that represent the columns of the UniGrid.</p> <p>The names of columns can be specified in the UniGrid's definition through the name attribute of individual <column> elements.</p> <p>This can be used to access the grid's columns in your code.</p> <p>For example:</p> <p>[C#]</p> <pre>UniGrid.NamedColumns["column1"].Visible = false;</pre> <p>When executed, this code would hide the column named <i>column1</i>.</p>	
ObjectType	<p>Can be used to define the data class of the objects that should be loaded as the data source and displayed by the UniGrid control. A list of all available data classes and related information can be found in the <i>CMS_Class</i> database table.</p> <p>Alternatively, the same can be defined in the UniGrid's definition through the <objecttype> element as described in</p>	

	<p>the following topic.</p> <p>Please note that this approach is not supported for classes representing document types (i.e. those whose value in the ClassIsDocumentType column is <i>1</i>). In these cases, you can load the required data by specifying an appropriate query through the Query property.</p>	
OrderBy	The ORDER BY clause used to determine how the UniGrid rows are sorted when the page is first loaded.	
Pager	Can be used to access the UniGridPager control used for paging.	
PageSize	<p>This setting can be used to override the default values offered by the page size selection drop-down list. Values must be separated by commas.</p> <p>The ##ALL## macro can be used as a value to indicate that all rows should be displayed.</p> <p>The default value is “25,50,100,##ALL##”.</p>	"10,20,##ALL##"
Query	<p>Can be used to specify the name of the query that should be used to retrieve data from the Kentico CMS database to be displayed by the UniGrid control. The name is entered in format <i><class name>.<query name></i>.</p> <p>Alternatively, the same can be defined in the definition through the <query> element as described in the following topic.</p>	"cms.user.selectallview"
SelectedItems	Gets (as an ArrayList) or sets the currently selected rows from the UniGrid.	
ShowActionsMenu	Indicates whether the header of the actions column should contain a context menu that provides the option to export the data displayed in the grid into various other formats (Excel, CSV or XML).	
ShowObjectMenu	Indicates if an action providing a context menu with object actions should automatically be added to the displayed grid.	

	<p>This requires the data source of the UniGrid to be an object type, specified either through the <i><objecttype></i> element or the ObjectType property.</p> <p>This menu provides options that can be used to Export, Backup, Restore or Destroy individual listed objects. Some types of objects may not have all menu options available.</p> <p>This action is not added if there is another action specified that has a contextmenu attribute or in cases where there are no actions at all defined for the grid.</p> <p>The default value is <i>true</i>.</p>	
SortDirect	The ORDER BY clause reflecting the current row sorting being used by the UniGrid.	
TopN	Specifies the maximum amount of rows that should be selected.	
WhereCondition	Can be used to get the used WHERE clause without modifications applied by the filter.	
ZeroRowsText	Text to be shown when the control is hidden by the HideControlForZeroRows property.	

The following events of the UniGrid control are available:

Event Name	Description
OnAction	Occurs when one of the actions of the control is used. The name of the given action is passed as a parameter to the handlers of the event. An example of how it is used can be found in the tutorial found in the Getting started topic.
OnExternalDataBound	Occurs after data is loaded. It is used to implement a custom design or functionality for UniGrid columns, including the action column. An example of how it is used can be found in the Implementing custom functionality topic.

OnBeforeDataReload	This event can be used to perform any actions before the <i>ReloadData()</i> method is executed.
OnAfterDataReload	This event can be used to perform any actions after the <i>ReloadData()</i> method is executed.

It may come in handy when building custom functionality for your site. We will define a basic UniGrid; change dynamically the editing (deleting) button image and finally we will create a custom column for editing the object.

The UniGrid is used to display and to edit the data across the user interface. Everywhere in the user interface you see a table with editing buttons you can be pretty sure, that this table is generated by a UniGrid. The UniGrid needs basically two inputs. The data you want to display and the UniGrid definition. The UniGrid definition can be defined either directly in the designer file or in an XML file. The designer file approach is recommended and that's why we will use it in our example.

At first let's create a UniGrid for testing purposes. You can place it for example into a new custom web part created according to developer's guide. At first you need to register the user control.

```
<%@ Register src="~/CMSAdminControls/UI/UniGrid/UniGrid.ascx"
tagname="UniGrid" tagprefix="cms" %>
```

In this example we will be using the users as testing data. So let's declare a userGrid UniGrid:

```
<cms:UniGrid ID="userGrid" runat="server"/>
```

This simple code places a Unigrid on your custom web part. The UniGrid can be populated in three different ways:

- A. You need to specify the query name in the UniGrid definition if you want to populate the UniGrid with a query. You can create the query in the CMSSiteManager / Development / Document types / <select a document type> / Queries section. You can also create a new document type which will be only a container for custom queries without any custom fields.

```
<cms:UniGrid ID="userGrid" runat="server" Query="cms.user.selectall"/>
```

- B. If you want to use the object approach, you need to define the edited object with the ObjectType attribute of the UniGrid:

```
<cms:UniGrid ID="userGrid" runat="server" ObjectType="cms.user" />
```

- C. The last approach is to populate the UniGrid in the CodeBehind file of your web part. This approach can be used for example if you need to add additional data to the displayed dataset:

```
userGrid.DataSource = CMS.SiteProvider.UserInfoProvider.GetAllUsers();
```

Now let's check how to define the layout and functionality of a UniGrid. The first approach would be to use an XML file:

```
<cms:UniGrid ID="userGrid" runat="server"
GridName="YourConfigurationFile.xml" />
```

In this case needs the file to be located in the same directory as the web part itself otherwise you need to specify the path to the file (the tilde sign is supported). There are a lot of examples through the system of such a setup. For example this approach is used in the BadWords module in the file

CMSModules\BadWords\BadWords_List.aspx. You can check it out if you would like to use this approach.

The next step in our example is to populate the UniGrid by placing the code from the data population approach C into your web part code behind file into the Page_Load method:

```
protected void Page_Load(object sender, EventArgs e)
{
userGrid.DataSource = CMS.MembershipProvider.UserInfoProvider.GetAllUsers();
}
```

Let's say you want only to display two columns, the UserName and the UserID. We populate the data programmatically, so you can use the UniGrid definition directly in the designer file. So let's add the mentioned columns to the GridColumns section of the UniGrid definition. Please add the columns so your code will look the following way:

```
<%@ Register src="~/CMSAdminControls/UI/UniGrid/UniGrid.ascx"
tagname="UniGrid" tagprefix="cms" %>
<%@ Register Namespace="CMS.UIControls.UniGridConfig" TagPrefix="ug"
Assembly="CMS.UIControls" %>

<cms:UniGrid ID="userGrid" runat="server" >
<GridColumns>
<ug:Column Source="UserName" Caption="User name" Wrap="false">
</ug:Column>
<ug:Column Source="UserID" Caption="User ID" Wrap="false">
</ug:Column>
</GridColumns>
</cms:UniGrid>
```

You can see that we had to add a new assembly so we can define the UniGrid actions. Your UniGrid should now list all the users if you access the page where your web part is placed.

In this example we want to be able to delete a selected entry, so we need to add a new UniGrid action. The **first column** in the dataset used as an identifier of the edited entry for all UniGrid actions if not specified otherwise by the commandargument property on the Action definition (please see the referenced documentation at the end for details). In our case it's the UserID. Additionally to distinguish between the actions which are called you

need to specify the Name property (this is useful if you have multiple actions defined). This value will be later used in the switch statement. By adding the GridActions section to the designer file code will look something like this:

```
<%@ Register src="~/CMSAdminControls/UI/UniGrid/UniGrid.ascx"
tagname="UniGrid" tagprefix="cms" %>
<%@ Register Namespace="CMS.UIControls.UniGridConfig" TagPrefix="ug"
Assembly="CMS.UIControls" %>

<cms:UniGrid ID="userGrid" runat="server">
<GridActions>
<ug:Action Name="deleteaction" Caption="$General.Delete$"
Icon="Delete.png" />
</GridActions>
<GridColumns>
<ug:Column Source="UserName" Caption="User name" Wrap="false">
</ug:Column>
<ug:Column Source="UserID" Caption="User ID" Wrap="false">
</ug:Column>
</GridColumns>
</cms:UniGrid>
```

The Source property of within the Column tag defines which columns are used from the supplied dataset. Our next step will be to execute our own code on this action. At first you need to register the OnAction event in the code behind file in the Page_Load method:

```
userGrid.OnAction += OnAction;
```

Now we have registered our new OnAction method. The method itself can look the following way:

```
private void OnAction(string actionName, object actionArgument)
{
switch (actionName.ToLower())
{
case "deleteaction":
// Code for deleting the user by the actionArgument which is the passed
UserID
UserInfoProvider.DeleteUser(ValidationHelper.GetInteger( actionArgument,
0));
// Repopulate the UniGrid with the new data
userGrid.DataSource = (CMS.SiteProvider.UserInfoProvider.GetAllUsers());
break;
}
}
```

You have probably noticed that the case “deleteaction” corresponds with the Name of the action in the markup file. In our example the UserID is used to delete the given user from the system. The actionArgument variable holds the UserID. If you check your page now, you should see a new delete button for each UniGrid entry. You

can try to delete some testing user from the system.

An expansion of our example could be to change the image used for the button. This could be used for example because you want to display different buttons for deleting specific users. Let's say you want to display a different image if the user is a global administrator. In that case you will need to dynamically change the button image during the creation of the UniGrid. For this you need to register the `OnExternalDataBound` event, where you will dynamically check the `UserInfo` object and change the icon according to this value. Please add this code into your Page Load method to register a new event:

```
userGrid.OnExternalDataBound += new
CMS.UIControls.OnExternalDataBoundEventHandler(userGrid_OnExternalDataBound);
```

The corresponding method will have to check for the action type and change the passed down `ImageButton` control to use a different image, if the user is a global administrator. One approach to get the information, if the User is a global administrator is to use our API. The more efficient approach is to check the data you already have available:

```
protected object userGrid_OnExternalDataBound(object sender, string
sourceName, object parameter)
{
    string id = ((System.Data.DataRowView)parameter)["UserID"].ToString();
}
```

In our dataset we retrieved all the columns for the `UserInfo` object, so we can use the following code to check if the user is a global administrator:

```
string isAdmin =
((System.Data.DataRowView)parameter)["UserIsGlobalAdministrator"].ToString();
```

Now let's use this information to achieve our desired functionality. We need to get a different image URL if the user is an administrator and we need to modify the sender object. In this case this will be an `ImageButton` control:

```
protected object userGrid_OnExternalDataBound(object sender, string
sourceName, object parameter)
{
    // User ID is used as actions parameter editedItemId
    string editedItemId = "";
    bool isAdmin = false;
    object param = null;

    if (parameter is System.Web.UI.WebControls.GridViewRow)
    {
        param = ((System.Web.UI.WebControls.GridViewRow)parameter).DataItem;
    }
}
```

```

editedItemId =
((System.Data.DataRowView) (param)).Row["UserId"].ToString();
isAdmin =
ValidationHelper.GetBoolean(((System.Data.DataRowView) (param)).Row["UserIs
GlobalAdministrator"], false);

string sImageUrl =
GetImageUrl("Design/Controls/UniGrid/Actions/Delete.png");
if (isAdmin)
{
sImageUrl = GetImageUrl("Design/Controls/UniGrid/Actions/Edit.png");
}

switch (sourceName)
{
case "deleteaction":
ImageButton btn = ((ImageButton) sender);
btn.ImageUrl = sImageUrl;
return btn;
}
return parameter;
}

```

Additionally you need to register the ExternalSourceName for this deleteaction from the switch statement in the upper code the following way:

```

<GridActions>
<ug:Action Name="deleteaction" ExternalSourceName="deleteaction"
Caption="$General.Delete$" Icon="Delete.png" />
</GridActions>

```

If you now load the UniGrid then you should get different icons for users which are global administrators.

The last example will be how to display the editing buttons also in the last column. By default are the buttons displayed in the first column, but in some cases this isn't desired. To achieve this we will have to change the UniGrid definition to include an additional column for our actions. To be able to get all the data from the row we will use the ##ALL## macro which is a substitute for all columns in our dataset. The ExternalSourceName is different since we need to create a different control for the action processing:

```

<%@ Register src="~/CMSAdminControls/UI/UniGrid/UniGrid.ascx"
tagname="UniGrid" tagprefix="cms" %>
<%@ Register Namespace="CMS.UIControls.UniGridConfig" TagPrefix="ug"
Assembly="CMS.UIControls" %>

<cms:UniGrid ID="userGrid" runat="server">
<GridActions>
<ug:Action Name="deleteaction" ExternalSourceName="deleteaction"
Caption="$General.Delete$" Icon="Delete.png" />

```

```

</GridActions>
<GridColumns>
<ug:Column Source="UserName" Caption="User name" Wrap="false">
</ug:Column>
<ug:Column Source="UserID" Caption="User ID" Wrap="false">
</ug:Column>
<ug:Column source="##ALL##" ExternalSourceName="deleteactioncolumn"
caption="Additional Actions" wrap="false">
</ug:Column>
</GridColumns>
</cms:UniGrid>

```

Now let's add a button to this binding event with a custom click event handler. To simulate an action button in a column we will need to define a JavaScript function for the click event. Since we still want to display the first column for deleting the user, we will need to modify the OnExternalDataBound handling method to return a clickable image if the column is used as an action. We will use the different ExternalSourceName to perform the additional processing. Also, the parameter object id of a different type so we will have to check its type to process it correctly. The JavaScript function name for our clickable image needs to be in the following form:

```
UG_Cmd_<UniGrid control ID>
```

The parameters of the method are the action name and the ID to identify the object in the UniGrid which is edited (deleted by your code). In our case it can be the UserID which is assigned to the variable editedItemId:

```

btnImage.Attributes["onclick"] = String.Format("return
UG_Cmd_{0}('deleteAction', {1});", userGrid.ClientID,
ScriptHelper.GetString(editedItemId));

```

When putting all those things together, here is the complete code of the OnExternalDataBound method:

```

protected object userGrid_OnExternalDataBound(object sender, string
sourceName, object parameter)
{
// User ID is used as actions parameter editedItemId
string editedItemId = "";
bool isAdmin = false;
object param = null;

if (parameter is System.Web.UI.WebControls.GridViewRow)
{
param = ((System.Web.UI.WebControls.GridViewRow)parameter).DataItem;
}
else if (parameter is System.Data.DataRowView) {
param = parameter;
}

editedItemId =

```

```

((System.Data.DataRowView) (param)).Row["UserId"].ToString();
isAdmin =
ValidationHelper.GetBoolean(((System.Data.DataRowView) (param)).Row["UserIs
GlobalAdministrator"], false);

string sImageUrl =
GetImageUrl("Design/Controls/UniGrid/Actions/Delete.png");
if (isAdmin)
{
sImageUrl = GetImageUrl("Design/Controls/UniGrid/Actions/Edit.png");
}

switch (sourceName)
{
case "deleteaction":
ImageButton btn = (ImageButton)sender;
btn.ImageUrl = sImageUrl;
return btn;

case "deleteactioncolumn":
Image btnImage = new Image()
{
ID = "deleteaction",
ImageUrl = sImageUrl,
Width = 16,
Height = 16
};
btnImage.Attributes["onclick"] = String.Format("return
UG_Cmd_{0}&#39;{deleteAction}&#39;;, {1});", userGrid.ClientID,
ScriptHelper.GetString(editedItemId));
return btnImage;
}
return parameter;
}


















```

Please use apostrophes instead of ' in the code.

Tip #1: If you want to add multiple controls use a Panel control to encapsulate them and return the Panel control, because only one control can be returned.

Tip #2: You can return also a string (for example custom HTML code), not only controls.

Here is the final table generated by the UniGrid:

Actions	User name	User ID	Additional Actions
	administrator	53	
	public	65	
	Andy	66	
	cmsdesigner	73	
	cmsdeskadministrator	74	
	cmseditor	75	
	cmsreader	76	
	CustomUser	111	
			Items per page: 25 

Uni-Grid Definition (XML Configuration).

Many configuration options that determine the behavior, design and content of the UniGrid control must be specifically defined. This can either be done either in an external XML configuration file, which is then assigned to the control through its **GridName** property, or directly within the definition of the control in the ASPX markup of the page or user control where the UniGrid is placed.

When using an external XML file, it must be organized according to the structure shown below (some elements are optional):

```
<?xml version="1.0" encoding="utf-8" ?>
<grid>

    <actions>
        <action />
        <separator />
        ...
    </actions>

    <columns>
        <column>
            <tooltip />
            <filter />
            ...
        </column>
        ...
    </columns>

    <objecttype />

    <query>
```

```

    <parameter />
    ...
</query>

<pager>
  <key name="DefaultPageSize" value="10" />
  ...
</pager>

<options>
  <key name="DisplayFilter" value="true" />
  ...
</options>
</grid>

```



Please note

If you use an external XML configuration file to specify the UniGrid's definition, the names of elements and their attributes used must be written in **lower case** to be recognized correctly, since it is case sensitive.

To define the UniGrid directly in the ASPX markup, it is first necessary to register the following namespace at the start of the code (in addition to the UniGrid control):

```
<%@ Register Namespace="CMS.UIControls.UniGridConfig" TagPrefix="ug" Assembly="CMS.UIControls" %>
```

Then you can simply add elements under the control according to the following structure:

```

<cms:UniGrid runat="server" ID="UniGrid" ... >
  <GridActions>
    <ug:Action />
  </GridActions>
</cms:UniGrid>

```

```

        <ug:ActionSeparator />
        ...
    </GridActions>

    <GridColumns>
        <ug:Column>
            <Tooltip />
            <Filter />
            ...
        </ug:Column>
        ...
    </GridColumns>

    <PagerConfig DisplayPager="true" ... />

    <GridOptions DisplayFilter="true" ... />
</cms:UniGrid>

```

When using this approach, the data source of the control must be specified directly through the UniGrid's properties (**Query**, **ObjectType** or **DataSource**). An advantage of this option is that you may use the IntelliSense in Visual Studio to help find the appropriate elements and attributes.

Individual elements that can be defined for the UniGrid and their attributes are described below:

- [<actions>](#)
- [<columns>](#)
- [<objecttype>](#)
- [<query>](#)
- [<pager>](#)
- [<options>](#)

<actions> (<GridActions>):

This element is used to define a column that contains various possible actions (e.g. Edit, Delete, View...) represented by icons for every row of the UniGrid. Individual actions must be defined by child **<action>** elements.

The following attributes of the **<actions>** element are available:

Attribute Name	Description	Sample Value
cssclass	Specifies the name of a CSS class from the assigned stylesheet to be used to style the appearance of the actions column.	"UniGridCustomActionsColumn"
parameters	A list of columns separated by semicolons that will be usable as parameters in the onclick or menuparameter attributes of child <action> elements .	"AttachmentGUID;AttachmentFormGUID"
showheader	Indicates whether the header of the actions column should be displayed. The default value is true.	
width	Determines the width of the actions column in the UniGrid.	"30% " "100px"

This element may contain **<action>** and **<separator>** child elements.

<action> (<ug:Action>):

This element is used to define individual actions. The implementation of individual actions is handled during the **OnAction** event of the UniGrid control. Any advanced features of individual action buttons, such as defining when a button should be functional, can be implemented in the handler of the **OnExternalDataBound** event.

The following attributes are available:

Attribute Name	Description	Sample Value
caption	Specifies the text used as the tooltip of the	"\$General.Delete\$"

	image defined in the icon attribute. You can enter the name of a resource string if you start and end it with the \$ character.	
commandargument	<p>The name of the column whose value should be passed as the actionArgument parameter of the OnAction event handler.</p> <p>If not defined, the first column of the data source will be used.</p>	
confirmation	The text used in a JavaScript confirmation for the action. Most commonly used as a confirmation for delete type actions. You can enter the name of a resource string if you start and end it with the \$ character.	"\$General.ConfirmDelete\$"
contextmenu	The relative path to a control (.ascx file) that implements a context menu for the action. Controls created for this purpose must inherit from the CMS.ExtendedControls.CMSContextMenuControl class.	"~/CMSAdminControls/UI/UniGrid/Controls/ObjectMenu.ascx"
externalsource name	Name of the action that is passed as the sourceName parameter of the OnExternalDataBound event handler.	"deletefile"
icon	Name of the image that should be used as the icon of the action. The image must be located in the folder defined by the ImageDirectoryPath property of the UniGrid.	"delete.png"
menuparameter	<p>Contains an array of parameters passed to the control implementing the action's context menu (the path to this control must be specified in the contextmenu attribute). These parameters may be retrieved in the control's code using the GetContextMenuParameter JavaScript function.</p> <p>The columns defined in the parameters attribute of the <actions> element may be entered as parameters using the following</p>	"new Array('cms.site', '{0}')"

	<p>expressions:</p> <p>{0} - first parameter</p> <p>{1} - second parameter</p> <p>and so forth.</p>	
mousebutton	<p>Specifies which mouse button causes the action's context menu to appear (if a context menu is enabled via the contextmenu attribute).</p> <p>If not defined, both mouse buttons open the context menu.</p>	<p>"left"</p> <p>"right"</p>
name	<p>Name of the action. This is passed to the handler of the OnAction event as the actionName parameter.</p>	<p>"delete"</p>
onclick	<p>The JavaScript OnClick function for the given action. It may use the columns defined in the parameters attribute of the <actions> element as parameters, which can be called by using the following expressions:</p> <p>{0} - first parameter</p> <p>{1} - second parameter</p> <p>and so forth.</p>	<p>"alert('{0}');"</p>
Action security		
modulename	<p>This attribute (and the two listed below) may be specified to leverage the security model of Kentico CMS to make the action usable only by a limited group of users.</p>	<p>"cms.ecommerce"</p>

	<p>Enter the code name of the module related to the action.</p> <p>You can find information about modules, their permissions and UI elements in the Site Manager -> Development -> Modules interface.</p>	
permissions	Sets the code name of the permission that users must have to be allowed to perform the action. The permissions must belong to the module specified in the modulename attribute.	"modifyorders"
ui elements	If specified, users will need to be allowed to view the given UI element in order to perform the action. The given user interface element must belong to the module specified in the modulename attribute.	"orders.general"
hide if not authorized	Indicates if the action should be hidden for users who are not allowed to perform it (as defined by the attributes above).	



Default object menu action

If your UniGrid control uses an object type data source (specified either through the `<objecttype>` definition element or the **ObjectType** property), then an action providing a context menu will automatically be added to the displayed grid.

This menu provides options that can be used to Export, Backup, Restore or Destroy the listed objects. Some types of objects may not have all menu options available.

This does not occur if you manually specify another action with a **contextmenu** attribute or in cases where there are no actions at all defined for the grid. You can also disable this action by setting the **ShowObjectMenu** property of the UniGrid to *false*.

<separator> (<ug:ActionSeparator>):

This element is used to define a separator between actions. The following attribute is available for it:

Attribute Name	Description	Sample Value
text	Text to be generated in the Literal control between actions.	<pre>"&lt;span class=&quot; UniGridActionSeparator&quot; &gt;&amp;nbsp;&lt;/span&gt;"</pre>

<columns> (<GridColumns>):

This element represents the main section of the UniGrid. The <columns> element itself has no attributes as each column can have its own settings. Individual columns are defined by child **<column>** elements.

<column> (<ug:Column>):

This element is used to define columns. Any advanced functionality of the cells in the given column can be implemented in the handler of the **OnExternalDataBound** event.

The following attributes are available for it:

Attribute Name	Description	Sample Value
action	Can be used to set the name of an action that will be performed when the content of this column's cells is clicked. An action with this name must be defined for the UniGrid via the name attribute of an <action> element.	
allowsorting	Indicates whether the column can be used to sort the rows of the UniGrid.	
caption	Specifies the text used as the header for the column. You can enter the name of a resource string if you start and end it with the \$ character.	"\$general.name\$"
commandargument	<p>The name of the column whose value should be passed as the actionArgument parameter of the OnAction event handler when the action specified via the action attribute is used.</p> <p>If not defined, the first column of the data source will be used.</p>	
cssclass	Specifies the name of a CSS class from the assigned stylesheet to be used to style the appearance of the given column.	"UniGridCustomColumn"
externalsourcename	<p>Sets a name for the column that will be passed as the sourceName parameter of the OnExternalDataBound event handler. Used for implementing custom functionality in the cells of the given column.</p> <p>You can use the following values to call built-in functions of the UniGrid to format the content of the column without having to write any code:</p> <p>•#yesno - can be set for columns with a source that uses the bit (boolean) data type. The values will be displayed as <i>Yes</i> (colored green) or <i>No</i> (colored red).</p>	

	<p>•#sitename - converts site ID (integer) values into the appropriate site display name for each row.</p> <p>•#sitenameorglobal - converts site ID values into the appropriate site display name for each row. If a record is not related to a specific site (i.e. the site ID is <i>null</i>), then the given cell will display (<i>global</i>) as its value.</p> <p>•#countryname - converts ID (integer) values into the display name of the Country object with the given ID.</p>	
href	If a URL is entered here, a link to this URL is generated around the content of the cells in this column. Macros {0}, {1}, ... can be used to access parameters defined by the parameters attribute.	"~/page.aspx"
icon	Name of an image that should be added into the column cells after the loaded data. The image must be located in the folder defined by the ImageDirectoryPath property of the UniGrid.	"edit.png"
istext	Indicates whether the content of the column is of type Text or nText. This is used to generate a special OrderBy clause of the query, so it must be set if sorting is enabled for the column.	
localize	Indicates whether localization should be enabled for string values in the column.	
maxlength	Sets the maximum number of characters that can be displayed in the column's cells. The last 3 characters will be replaced by periods.	
name	Can be used to set a custom name for the column, which will be used in the column dictionary accessible through the NamedColumns property of the UniGrid control.	
parameters	Names of the columns used as parameters of the URL generated by the Href attribute. Separated by semicolons.	
source	Name of the column from the data source of the UniGrid that is used as the source for the content of this column. The special macro ##ALL## can be used to specify all columns.	

sort	Used to define the column name to be used for sorting if the ##ALL## macro is used in the source attribute.	
style	The style used for the entire column.	"padding:10px"
visible	Indicates whether the column should be visible.	
width	Determines the width of the column.	"20% " "200px"
wrap	Indicates whether word wrapping is used in the column.	

The column element may contain child **<tooltip>** and **<filter>** elements.

<tooltip>:

When this element is added, a tooltip is displayed when the mouse hovers over the content of the cells in this column. If an icon is present in the cell, the tooltip is displayed over the icon instead of the text. The content of the tooltip can be defined and configured by the following attributes:

Attribute Name	Description	Sample Value
encode	Indicates whether the output of the tooltip should be encoded.	
externalsourcename	Name used in the OnExternalDataBound event for changing the appearance of the tooltip. This can be used to create complex tooltips including images, panels etc.	
source	Name of the column from the data source of the UniGrid that is used as the source of the tooltip.	
width	Determines the width of the tooltip.	

<filter>:

When this element is added, the given column will be used in the UniGrid filter. The following attributes are available to configure the filter:

Attribute Name	Description	Sample Value
format	<p>Can be used to define a custom WHERE clause format to be generated by the default filter. The following expressions can be used:</p> <p>{0} - is resolved into the column name</p> <p>{1} - is resolved into the operator selected in the drop-down list of the default filter</p> <p>{2} - is resolved into the value entered into the textbox of the default filter</p>	" [{0}] {1} '{2}' "
size	Determines the maximum amount of characters that can be entered into the textbox of the default filter. Available for <i>Text</i> , <i>Integer</i> and <i>Double</i> filter types . The default value is 1000.	
source	Name of the column used in the WHERE clause generated by the filter.	
path	Path to the control (.ascx file) that should be used instead of the default filter for the column. If filled, the type attribute is ignored. The default relative path is ~/CMSAdminControls/UI/UniGrid/Filters/.	
type	The filter type that should be created for the given column.	"Text" "Bool" "Integer" "Double"

<objecttype>:

This element can be used to define the data class of the objects that should be loaded as the data source and displayed by the UniGrid control. A list of all data classes and related information can be found in the *CMS_Class* database table. Please note that this approach is not supported for classes representing document types (those whose value in the **ClassIsDocumentType** column is *1*).

If this element isn't used, a data source must be retrieved by means of the <query> element or assigned through the UniGrid control's **DataSource** property before its *ReloadData()* method is called. Alternatively, the **ObjectType** property of the UniGrid control can be used for the same purpose.

The following attributes can be used to define the object type:

Attribute Name	Description	Sample Value
columns	Names of the columns that should be retrieved separated by commas. If empty, all columns will be retrieved. By default, the values of the first column are passed as the actionArgument parameter of the OnAction event handler. This can be overridden for actions by specifying a column name in the commandargument attribute of individual <action> elements.	
name	Code name of the used data class.	"cms.user"

<query>:

This element can be used to specify the system query that will retrieve data from the Kentico CMS database to be displayed by the UniGrid control. If it isn't used, an external data source must be assigned through the UniGrid control's **DataSource** property before its *ReloadData()* method is called. Alternatively, the **Query** property of the UniGrid control can be used for the same purpose.

The following attributes can be used to define the query:

Attribute Name	Description	Sample Value
columns	Names of the columns that should be retrieved by the query separated by commas. If empty, all database columns will be retrieved. By default, the values of the first column are passed as the actionArgument parameter of the OnAction event handler. This can be overridden for actions by specifying a column name in the commandargument attribute of individual <action> elements.	
name	Code name of the used system query in format <i><class name>.<query name></i> .	"cms.site.selectsitelist"

The query element may contain **<parameter>** child elements:

<parameter>:

This element can be used to define the value of a parameter inside the specified query.

The following attributes must be filled to define the parameter:

Attribute Name	Description	Sample Value
name	Name of the parameter. Parameters are placed into queries using the following syntax: <i>@<paramater name></i>	

	<p>For example, if the specified query looked like this:</p> <pre>SELECT TOP @customTop FROM CMS_User</pre> <p>Then entering <i>customTop</i> into this attribute would cause the value of this element to be used by the query instead of the @customTop expression.</p>	
type	The type of the parameter.	"String" "Int" "Double" "Bool"
value	The value of the parameter.	

<pager> (<PagerConfig>):

This element is used to define the behaviour of the UniGrid pager. You can either add the settings as child **<key>** elements in the XML configuration file, or as attributes of the **<PagerConfig>** element when defined directly in the code. The following are available:

Key name	Description	Sample Value
DisplayPager	Indicates if a pager should be included below the UniGrid. <i>True</i> by default.	<code><key name="DisplayPager" value="false" /></code>
DefaultPageSize	Defines the default amount of rows displayed on one UniGrid page.	<code><key name="DefaultPageSize" value="10" /></code>

	The value must be one of the options offered by the page size selection drop-down list. These values are defined by the PageSizeOptions key.	
PageSizeOptions	<p>This setting can be used to override the default values offered by the page size selection drop-down list. Values must be separated by commas.</p> <p>The ##ALL## macro can be used as a value to indicate that all rows should be displayed.</p> <p>The default value is “25,50,100,##ALL##”.</p>	<pre><key name="PageSizeOptions" value="10,20,##ALL##" /></pre>
ShowDirectPageControl	Indicates whether a drop-down list used for direct page selection should be displayed.	<pre><key name="ShowDirectPageControl" value="true" /></pre>
ShowFirstLastButtons	Indicates whether the buttons that link to the first and last page should be displayed.	<pre><key name="ShowFirstLastButtons" value="false" /></pre>
ShowPageSize	Indicates whether the page size selection drop-down list should be displayed.	<pre><key name="ShowPageSize" value="false" /></pre>
ShowPreviousNextButtons	Indicates whether the buttons that link to the previous and next page should be displayed.	<pre><key name="ShowPreviousNextButtons" value="false" /></pre>
ShowPreviousNextPageGroup	Indicates whether the buttons that link to the next group of page links should be displayed.	<pre><key name="ShowPreviousNextPageGroup" value="false" /></pre>
VisiblePages	Determines the amount of displayed page links in one group.	<pre><key name="VisiblePages" value="5" /></pre>

<options> (<GridOptions>):

This element is used to define additional settings and special features of the UniGrid control. You can either add the settings as child **<key>** elements in the XML configuration file, or as attributes of the **<GridOptions>** element when defined directly in the code. The following are available:

Key name	Description	Sample Value
DisplayFilter	Indicates whether a filter should be displayed above the UniGrid. If the amount of displayed rows is lower than the value of the FilterLimit key, the filter will be hidden despite this setting.	<code><key name="DisplayFilter" value="true" /></code>
FilterLimit	Determines the minimum amount of rows that must be displayed in the UniGrid before a filter is shown. The default value is read from the CMSDefaultListingFilterLimit web.config key.	<code><key name="FilterLimit" value="10" /></code>
ShowSelection	Indicates whether a column allowing the selection of rows should be displayed on the left of the UniGrid. This can be used to perform mass actions affecting multiple rows. The selected rows can be accessed through the SelectedItems property of the UniGrid.	<code><key name="ShowSelection" value="true" /></code>
SelectionColumn	Name of the column used as an item in the array of selected rows which can be accessed through the SelectedItems property of the UniGrid. By default the first column in the data source is used.	<code><key name="SelectionColumn" value="SiteName" /></code>
ShowSortDirection	Determines if an arrow showing the sorting direction should be displayed next to the header of the column used for sorting.	<code><key name="ShowSortDirection" value="false" /></code>

About Ray Business Technologies

Ray Business Technologies is a leading Global Information Technology (IT) Services and Solutions, a CMMI Level 3, ISO 27001:2013 and ISO 9001:2015 Certified Company. We are a Member of NASSCOM, HYSEA, NJTC, and AIIA. Ray Business Technologies offers comprehensive end-to-end IT Services for Business Application Development, Enterprise Solutions, Enterprise Collaboration Services, Testing and Quality Assurance Services, Cloud Computing and IT Infrastructure Management to organizations in the Banking & Finance, Insurance, Healthcare, Manufacturing, Retail, Media & Entertainment, Leisure & Travel, Telecom and Energy & Utilities verticals as well as Independent Software Vendors.