

EPiServer Web Controls

EPiServer Web Controls are a number of ASP.NET Web Custom Controls collected in the name space EPiServer.WebControls. ASP.NET Web Custom Controls differ from ASP.NET Web User Controls in several ways. One is the fact that Web Custom Controls are created solely in code; there's no visual part (no ascx file).

It's a safe bet to state that there is a definite pattern to the control types found in the name space EPiServer.WebControls. Very many of them deal with the Web Page Tree that's created in the EPiServer Edit mode and indeed make up your Web site. Thus, you'll find control classes that deal with creating menus from part of or all of the Web Page Tree, as well as control classes used to create various kinds of lists. Most of these controls are templated controls giving you excellent control of their visual behaviour without the need for code in the code-behind file. (ASP.NET templated controls are discussed below.)

Table 7-1: EPiServer ASP.NET Web Custom Controls

<i>Class</i>	<i>Description</i>
Calendar	Displays a template-based calendar where the calendar events are pages in EPiServer.
CalendarEventTemplate-Container	Template for a Calendar Event.
CategoryTemplate-Container	Used for template context at category level.
CategoryTree	Renders a tree of categories from a given root category.
ChangedPages	Lists changed pages for the specified root page and all child pages.
Clear	Simple control to put an HTML Img tag with a transparent GIF picture of the given dimensions.
Content	Content in ContentFramework on a Web Form that will populate controls to Region.

Table 7-1: EPiServer ASP.NET Web Custom Controls

<i>Class</i>	<i>Description</i>
ContentFramework	Provides support for visual inheritance. A skeleton user control populated with Region controls, the contents of which can be replaced by Content controls contents in a ContentFramework on a Web Form.
ContentFramework-Preview	Used to create a miniature model of a table structure.
ContentFramework-Selector	Select framework to use when multiple frameworks are defined
DayTemplateContainer	Template for rendering information about a single day.
DirectoryTemplate-Container	Template class used by FileTree to display directories
DynamicCell	A TableCell whose width can be changed.
DynamicResizeCell	A TableCell that can be used to change width on DynamicCells.
DynamicResizeRow	A TableRow that can be used to change width on DynamicRows.
DynamicRow	A TableRow that can change height.
DynamicTable	A Table that contains resizable Rows and Cells. These can be personalized and saved by authenticated users.
ExplorerTree	Displays a tree Explorer style.
FileTemplateContainer	Template class used by FileTree to display files
FileTree	Displays a file tree from the server.
FormFieldStatistic	Helper class to keep track of information for a field.
FormPostings	Gets a list of form postings as created by the PropertyForm control
FormStatistics	Used to create a statistics view of data posted with a PropertyForm control
InputBase	Base class for simple input types.

Table 7-1: EPiServer ASP.NET Web Custom Controls

<i>Class</i>	<i>Description</i>
InputCategoryTree	Simple input type that displays a page selector.
InputEditorOptions	Simple input type that displays editor options.
InputFrame	Simple input type that displays a frame drop-down list.
InputLanguage	Simple input control used to select a language
InputPageDefinitionType	Simple input type that displays a page definition type drop-down list.
InputPageReference	Simple input type that displays a page selector.
InputPassword	Simple input control that allows a user to enter / change passwords
InputSortOrder	Simple input type that displays available sort orders.
InputTab	Simple input type that displays a tabbed drop-down list.
InputTimeSpan	Simple input type that displays a time span.
MenuList	Renders a menu list of top level items; useful for navigations that contain a top level menu displaying sub-trees as clicked.
NewsList	Control for rendering news list with specialised template for top level news.
PageControlBase	Serves as a base class for all controls that generate any type of PageData collection.
PageList	Control for rendering page list; extends PageListData with templates.
PageListData	Base data control for accessing page list.
PageSearch	WebControl that handles text searches against EPiServer pages in the database and files indexed by Microsoft Index Server.
PageTemplateContainer	Used for template context at page level.

Table 7-1: EPiServer ASP.NET Web Custom Controls

<i>Class</i>	<i>Description</i>
PageTree	Control for rendering page trees; extends PageTreeData with templates.
PageTreeData	Base data control for accessing page trees.
PortalFramework	A Web control that renders a table structure containing IFrames. This control will load an XML file which has to be located in the /Util/PortalFramework folder.
PortalRegion	A region in ContentFramework which can be populated with controls from a Content.
Property	WebControl for rendering page properties.
PropertyCriteriaControl	Holder of criteria information used by property searching.
PropertyCriteriaControl-Builder	Builds child controls of the type PropertyCriteriaControl.
PropertySearch	Advanced property search in the complete database.
PropertyTemplateContainer	Used for template context at property level.
Region	A region in ContentFramework that can be populated with controls from a Content.
ResetDynamicTableButton	A LinkButton that will remove any personalized values from all DynamicTables on the page.
SaveDynamicTableButton	A LinkButton that will collect and save changed values to height and width for all DynamicTables on the page.
SiteMap	Displays a site map using templates and predefined design options.
SubscriptionList	Displays a list of subscription options.
Translate	WebControl for language specific strings.
XmlNameValidator	Checks that the validated controls value conforms with naming rules for an XML identifier

Inheritance Tree for EPiServer.WebControls

The inheritance tree for EPiServer.WebControls is quite extensive.

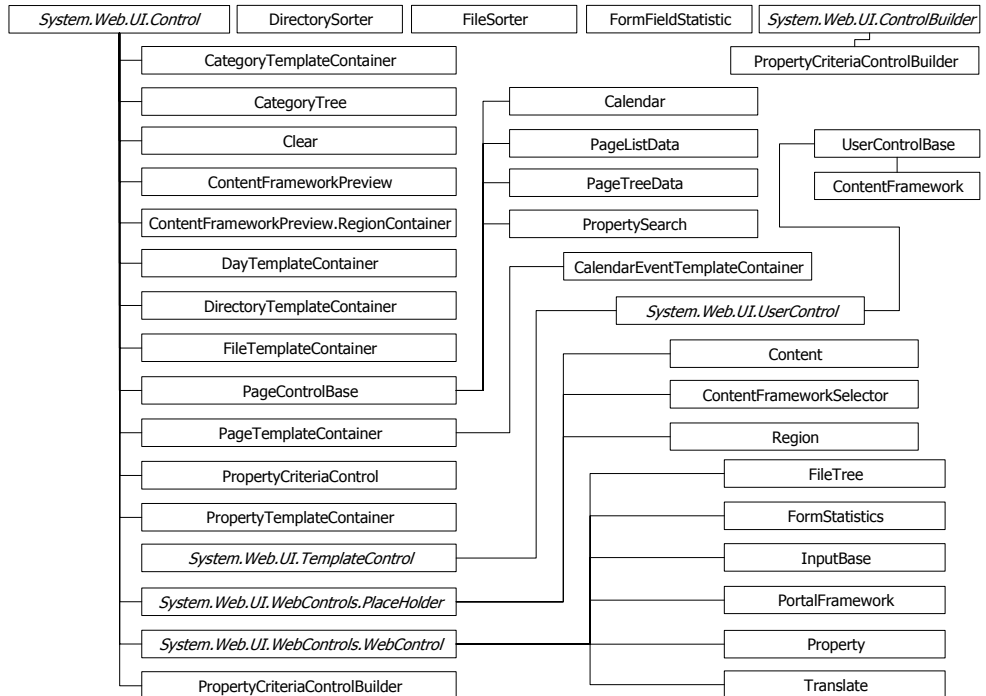


Figure 7-1: Inheritance Tree for EPiServer.WebControls name space, first three levels shown for all, more for UserControlBase and ContentFramework.

All of the five top level classes inherit directly from System.Object. The inheritance tree for EPiServer.WebControls is at most seven levels deep (including System.Object). Only the classes EPiServerValidator and XmlNameValidator have a seven level deep inheritance chain: (not shown in figure 7-1).

ASP.NET Templated Controls

ASP.NET Templated Controls represent an elegant way to separate presentation and data whilst keeping some connection between the two. In short, templated controls offer HTML template tags with given names. Each control class can define their own names, which are handled by code in the control's class.

For example, the built-in DataList server control (System.Web.UI.WebControls.DataList) defines these template names:

- HeaderTemplate
- FooterTemplate

- ❑ ItemTemplate
- ❑ AlternatingItemTemplate
- ❑ SeparatorTemplate
- ❑ SelectedItemTemplate
- ❑ EditItemTemplate

Visual Studio .NET has support for the visual layout of templated controls, but there's no problem adding them directly to a Web Forms page. This example comes from the help file:

Example 7-1: Using templated control DataList.

```
<asp:datalist ID="DataList1" runat="server">
  <HeaderTemplate>
    Employee List
  </HeaderTemplate>
  <ItemTemplate>
    <asp:label id=Label1 runat="server" Text='<%# DataBinder.Eval( Container,
      "DataItem.EmployeeName") %>' />
    <asp:label id=Label2 runat="server" Text='<%# DataBinder.Eval( Container,
      "DataItem.PhoneNumber") %>' />
    <asp:Hyperlink id=Hyperlink1 runat="server"
      Text='<%# DataBinder.Eval( Container, "DataItem.Email") %>'
      NavigateURL='<%# DataBinder.Eval( Container, "DataItem.Link") %>' />
  </ItemTemplate>
</asp:datalist>
```

Templated Controls Have an Imaginary Foreach Statement

It is possible to envisage a C# language foreach-statement enclosing parts of the templated control. Rewriting the example above results in this:

Example 7-2: Pseudo code for templated control with an imaginary foreach statement.

```
<DataList control>
  <!-- ItemTemplate -->
  foreach ( object DataItem in Container.DataItems ) {
    use DataBinder.Eval, Container and Container.DataItem
  }
  <!-- ItemTemplate -->
</DataList>
```

The Container Property

ASP.NET templated controls seem to have this mysterious property Container which is used inside the control (you can see it in both example 7-1 and 7-2).

'Container' is actually a code-generated property which refers to the individual row/item in the outside control. As we are using DataList in the example, Container is a DataListItem. Container only exists for autogenerated templates, and you must implement it yourself in the ITemplate class. Generally, it's the NamingContainer of the control you get sent to instantiate into. Luckily, this has been done in all of EPiServer templated controls.

EPiServer Templated Controls

You can make good use of Templated Controls when developing solutions with EPiServer. Most of the controls in EPiServer.WebControls are templated controls. We'll take a look at NewsList (EPiServer.WebControls.NewsList). This control class defines these templates:

- FirstNewsTemplate, template for the first news item
- SecondNewsTemplate, template for the second news item
- ThirdNewsTemplate, template for the third news item
- FourthNewsTemplate, template for the fourth news item
- NewsTemplate, default template for news items

In addition to the listed templates, there are also FooterTemplate and HeaderTemplate, with obvious uses.

An example of how to use this control in a Web Form or a User Control might look like this:

Example 7-3: Using EPiServer templated control NewsList.

```
<EPiServer:Newslist ID=Newslistnew Pagelinkproperty="NewsContainer" runat="Server"
  MaxCount='<%# GetNewsCount() %>'>
...
  <NewsTemplate>
    <tr>
      <td class="DateListingText"><%# Container.CurrentPage[ "PageStartPublish" ] %></td>
    </tr>
    <tr>
      <td>
        <a href="<%# Container.CurrentPage.LinkURL %>" class="StartPageHeading">
          <%# Container.CurrentPage.PageName %></a>&nbsp;<br />
          <span class="Normal"><%# Container.CurrentPage[ "MainIntro" ] %></span>
        </td>
      </tr>
      <tr><td><EPiServer:Clear height="6" runat="server" /></td></tr>
    </NewsTemplate>
  </EPiServer:Newslist>
```

The code in example 7-3 uses just NewsTemplate, meaning that all news items will look the same in the list. Each news item will be displayed inside an HTML table using three rows each. On the first row of the table, the contents of the PageStartPublish property for the page are displayed. The second row will show the contents of the MainIntro property and the third row is simply a spacer to offset the news items vertically from each other. The use of classes in the Table Data tag and the Span tag is linked to the Cascading Style Sheet used. Looking at class DateListingText in the style sheet actually used, we find that it is set using a different foreground colour (#606060) and font (7 pt, italic).

```
2/11/2004 11:39:21 AM
Mimic Almost as Good as
Any Example
The Mimic web site has
been found to equal any
example web site in
appearance.

2/11/2004 11:38:18 AM
To Be Reckoned with
Creators of web sites should
pay close attention to Mimic.

2/11/2004 11:29:23 AM
Mimic Up And Coming
Web site Mimic is a strong
contender says Reuters.
```

Figure 7-2: Actual news items presented using the templated control NewsList.

Comparing the code in 7-3 with the news items in figure 7-2 reveals a high degree of correspondence between the two.

In the resulting HTML code, this news list will be distinguished by the first news item being the only one which is displayed.

The Container Property in EPiServer Templated Controls

For the EPiServer templated controls in EPiServer.WebControls, the natural naming container is hosted on a Web page most of the time. This means that the Container property has a type of EPiServer.WebControls.PageTemplateContainer. For Calendar events, EPiServer.WebControls.Calendar, the Container property type is EPiServer.WebControls.CalendarEventTemplateContainer.

EPiServer.WebControls.Clear

Not many EPiServer controls are designed to produce invisible results, but that's exactly what Clear does. Including an object of the Clear class in HTML produces an invisible, transparent, area of specified dimensions using a transparent GIF image. Its primary use is to avoid cluttering, to make the visual impression of the Web page lighter.

Example 7-4: Clear control object using in HTML.

```
<%@ Register TagPrefix="EPiServer" Namespace="EPiServer.WebControls" Assembly="EPiServer" %>
...
<EPiServer:Clear Width="150" Height="10" runat="server" />
```


The Clear control object included in example 7-4 would result in the final rendering of a transparent GIF image with the specified dimensions.

Example 7-5: HTML IMG tag equivalent to Clear control object.

```

```

The HTML IMG tag in example 7-5 and the EPiServer.WebControls.Clear control object in example 7-4 are equivalent.

If you don't specify either or both of Height or Width, they will assume their default values of '1'.

Example 7-6: Clear control object using in HTML.

```
<%@ Register TagPrefix="EPiServer" Namespace="EPiServer.WebControls" Assembly="EPiServer" %>
...
<EPiServer:Clear Height="10" runat="server" />
```

The Clear control object in example 7-11 produces a GIF image which is 10 pixels high and 1 pixel wide.

As can be seen in figure 7-1, Clear is a direct descendant of System.Web.UI.Control; it only adds the Height and Width attributes.

The Visible attribute can be used to decide whether or not to actually render an image.

Example 7-7: Using HasChildren attribute to control attribute Clear.Visible.

```
<%@ Register TagPrefix="EPiServer" Namespace="EPiServer.WebControls" Assembly="EPiServer" %>
...
<EPiServer:Clear Visible=<%=# Container.HasChildren %> Width="7" Height="7" runat="server" />
```

EPiServer.WebControls.Content

Objects instantiated from the Content control class are always used in conjunction with Region control object (read more about the Region class on page 197, onwards). Content objects define and name an area in the HTML landscape which can be claimed by Region objects simply by stating the name of the pertinent area. There's a clear distinction between Content and Region objects though: Region objects are used in Framework Definition Files and Content objects are found in Page Template Files. Together, they form the building blocks of the EPiServer Content Framework.

Two of the attributes in the Content class are important:

Table 7-2: The two important attributes in EPiServer.WebControls.Content.

<i>Attribute Name</i>	<i>Purpose</i>
ID	Unique ID for this object (this particular instance of Content) [string].
Region	Name of the Region whose contents will be replaced by the contents of Content [string].

Using Content objects is easy:

Example 7-8: Using Content control in a Page Template File.

```
<%@ Page ... Inherits="development.Default" %>
<%@ Register TagPrefix="EPiServer" Namespace="EPiServer.WebControls" Assembly="EPiServer" %>
<%@ Register TagPrefix="development" TagName="DefaultFramework ... %>
...
<development:DefaultFramework id="DefaultFramework" runat="server">
  <EPiServer:Content ID="NewsListing" Region="menuRegion" runat="server">
    <table>
      <tr>
        <td>
          <EPiServer:Newslist ID=Newslistnew Pagelinkproperty="NewsContainer"
            runat="Server" MaxCount='<%=# GetNewsCount() %>'>
            <Newstemplate>
              <tr><td><%=# Container.CurrentPage[ "MainIntro" ] %></td></tr>
            </Newstemplate>
          </EPiServer:Newslist>
        </td>
      </tr>
    </table>
  </EPiServer:Content>
...
</development:DefaultFramework>
```

Comparing the HTML code in example 7-8 with that in example 7-30 on page 197, we see that the Web User Control Menu.ascx that is used in example 7-30 is replaced by a very simple news listing in example 7-8.

Even if you're working with nested Region controls, there's no need to nest the Content controls, as they will replace the proper Region control contents anyway.

Remember, in the C# world many strings are case-sensitive. To improve portability always treat them as such, no matter which programming language you are using.

EPiServer.WebControls.ContentFramework

The only purpose for EPiServer.WebControls.ContentFramework is to act as a base class for classes in Framework Definition Files. Remember that Framework Definition Files are ASP.NET Web User Controls (they're just not used like that in the EPiServer Content Framework), a point which is made even clearer by the fact that ContentFramework inherits EPiServer.UserControlBase.

Example 7-9: Code-behind file from a Framework Definition File.

```
namespace development.Frameworks {
    /// <summary>SimpleFramework is a simple Framework Definition
    /// File.</summary>
    public abstract class SimpleFramework : EPiServer.WebControls.ContentFramework {
        protected EPiServer.WebControls.Region topRegion;
        protected EPiServer.WebControls.Region leftColumn;
        protected EPiServer.WebControls.Region centreRegion;
        protected EPiServer.WebControls.Region footerRegion;

        private void Page_Load( object sender, System.EventArgs e ) {
            // Put user code to initialize the page here
        }

        // Web Form Designer generated code
    }
}
```

EPiServer.WebControls.ContentFrameworkSelector

Its name is a dead give-away: ContentFrameworkSelector enables you to adapt to different Framework Definition Files, e.g. ordinary Framework Definition Files and those which have portal functionality.

In the example Web site, the Page Template File Calendar.aspx uses ContentFrameworkSelector to render either of two Web User Controls: Calendar.ascx or PortalCalendar.ascx.

Example 7-10: ContentFrameworkSelector in Calendar.aspx.

```
<%@ Page language="c#" Codebehind="Calendar.aspx.cs" AutoEventWireup="false"
Inherits="development.Templates.calendar" %>
<%@ Register TagPrefix="EPiServer" Namespace="EPiServer.WebControls" Assembly="EPiServer" %>
<%@ Register TagPrefix="development" TagName="DefaultFramework"
    Src="~/templates/Frameworks/DefaultFramework.ascx"%>
<%@ Register TagPrefix="development" TagName="PortalFramework"
    Src="~/templates/Frameworks/PortalUnitFramework.ascx"%>
```

```

<%@ Register TagPrefix="development" TagName="Calendar"
    Src="~/templates/Units/Calendar.ascx"%>
<%@ Register TagPrefix="development" TagName="PortalCalendar"
    Src="~/templates/Units/PortalCalendar.ascx"%>

<EPiServer:ContentFrameworkSelector runat="server" FrameworkKeyName="MyMode"
    ID="FrameworkSelector" DefaultFramework="DefaultFramework">
    <development:DefaultFramework id="DefaultFramework" runat="server">
        <EPiServer:Content Region="mainRegion" ID="calendarContent" runat="server">
            <development:Calendar id="Calendar" runat="server"></development:Calendar>
        </EPiServer:Content>
        <EPiServer:Content Region="rightColumnRegion" ID="emptyContent" runat="server" />
    </development:DefaultFramework>
    <development:PortalFramework id="portalmode" runat="server">
        <EPiServer:Content Region="mainRegion" ID="calendarContent2" runat="server">
            <development:PortalCalendar id="PortalCalendar" runat="server" />
        </EPiServer:Content>
    </development:PortalFramework>
</EPiServer:ContentFrameworkSelector>

```

When DefaultFramework is active, Calendar.aspx (watch out for the small difference between Web Form file Calendar.aspx and Web User Control file Calendar.ascx) replaces the contents of Region 'mainRegion' with Web User Control Calendar and also replaces the contents of Region 'rightColumnRegion' with nothing (an empty Content control). On the other hand, when the Framework PortalFramework is active, another Web User Control, PortalCalendar.ascx is used.

EPiServer.WebControls.ExplorerTree

Windows Explorer is the visual ideal for ExplorerTree. It displays a tree, such as the Web Page tree, just like the Windows Explorer (in its left pane). ExplorerTree control objects are used in both Admin and Edit modes of EPiServer.

ExplorerTree is a direct descendant of EPiServer.WebControls.PageTreeData (see figure 7-1).

Figure 7-3 below shows the visual appearance of ExplorerTree. (You should recognise the Web Page Tree from the example Web site; it's the tree shown in the left pane of Edit mode.)

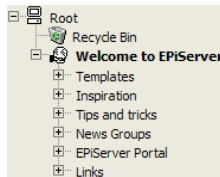


Figure 7-3: Using ExplorerTree in EPiServer Edit mode.

As with most other EPiServer Web Custom Controls, ExplorerTree is very easy to use. You only have to decide on a root page for the control. In the code-behind file, you call the DataBind method for ExplorerTree which connects the control with the EPiServer Web Page Tree.

Example 7-11: ExplorerTree in the code-behind file.

```
protected EPiServer.WebControls.ExplorerTreeExplorerTreeObj;

private void Page_Load( object sender, System.EventArgs e ) {
    if ( ! IsPostBack ) {
        ExplorerTreeObj.DataBind();
    }
}
```

In the HTML file:

Example 7-12: HTML code needed for ExplorerTree.

```
<EPiServer:ExplorerTree EnableVisibleInMenu="False" ShowRootPage="True"
    PageLink='<%# ( EPiServer.PageBase ) Page ).Configuration.StartPage %>' ShowIcons="True"
    ClickScript="window.location.href = '{PageLinkURL}'" id="ExplorerTreeObj" runat="server"
/>
```

This results in the following rendition:



Figure 7-4: Using ExplorerTree control on a Web page.

Some of ExplorerTree's attributes are more important than others.

Table 7-3: ExplorerTree attributes.

<i>Attribute</i>	<i>Description</i>
BranchUrl	The URL that will be called to load child pages; default is 'Util/ExplorerTreeBranch.aspx'.
ClickScript	Set script code to be executed when the page name is clicked.
EnableVisibleInMenu	(Inherited from PageTreeData) If tree should check that page should be 'visible in menus' to appear in tree.
ExpandAll	(Inherited from PageTreeData) Expand all tree nodes.
ImageDirectory	Path to image folder.
PageLink	(Inherited from PageControlBase) The root page to read data from
ShowIcons	Show icons on system pages
ShowRootPage	(Inherited from PageTreeData) If root page should be loaded in the page list
ShowStatusIcons	Show status icons, indicating access rights and published status

EPiServer.WebControls.MenuList

Having a name like MenuList destines a class for handling menus. Of course, this is what EPiServer.WebControls.MenuList does. MenuList is a direct descendant of EPiServer.WebControls.PageTreeData and thus a sibling to EPiServer.WebControls.ExplorerTree, EPiServer.WebControls.PageTree and EPiServer.WebControls.SiteMap.

In the User Control TopMenu that ships with EPiServer, MenuList is used to create a horizontal top-level menu.

TopMenu.ascx itself is used in the Framework Definition File DefaultFramework outside of any Regions, meaning that all Page Template Files that use DefaultFramework get TopMenu.ascx.

TEMPLATES **INSPIRATION** TIPS & TRICKS NEWS GROUPS EPISERVER PORTAL LINKS

Figure 7-5: Result of using TopMenu.ascx in the example Web site.

TopMenu.ascx.cs is not terribly complex:

Example 7-13: TopMenu.ascx.

```
namespace development.Templates.Units {
    /// <summary>TopMenu implements a horizontal top menu.
    /// </summary>
    public abstract class TopMenu : EPiServer.UserControlBase {
        protected EPiServer.WebControls.PropertyProperty1;
        protected EPiServer.WebControls.PropertyProperty2;
        public EPiServer.WebControls.MenuListMenuListControl;

        private void Page_Load( object sender, System.EventArgs e ) {
            if ( ! IsPostBack ) {
                MenuListControl.DataBind();
            }
        }

        protected EPiServer.Core.PageReference MenuRoot {
            get {
                if ( CurrentPage[ "MainMenuContainer" ] != null ) {
                    return (EPiServer.Core.PageReference) CurrentPage[ "MainMenuContainer" ];
                } else {
                    return Configuration.StartPage;
                }
            }
        }
    }
}
```

Nor is TopMenu.ascx very complex:

Example 7-14: TopMenu.ascx.

```
<%@ Register TagPrefix="EPiServer" Namespace="EPiServer.WebControls" Assembly="EPiServer" %>
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="TopMenu.ascx.cs"
    Inherits="development.Templates.Units.TopMenu"
    TargetSchema="http://schemas.microsoft.com/intellisense/ie5"%>
<table width="100%" border="0" cellspacing="0" cellpadding="0"
    xmlns:EPiServer="http://schemas.episerver.com/WebControls">
    <tr valign="middle" align="center">
        <EPiServer:MenuList runat="server" ID="MenuListControl" PageLink="<%=#MenuRoot%>">
            <ItemTemplate>
                <td height="23">
                    <EPiServer:Property id="Property1" runat="server" PropertyName="PageLink"
                        CssClass="MenuHead"></EPiServer:Property>
                </td>
            </ItemTemplate>
        <SelectedTemplate>
```

```

        <td height="23">
            <EPiServer:Property id="Property2" runat="server" PropertyName="PageLink"
                CssClass="ActiveMenuHead"></EPiServer:Property>
        </td>
    </SelectedTemplate>
</EPiServer:MenuList>
</tr>
</table>

```

As you can see in example 7-14, all TopMenu.ascx does is create one row in an assumed outer table containing one table data cell for every top level record. Menu items are displayed according to the style sheet style MenuHead (which among other settings specifies ‘text-transform : uppercase’). Selected menu items take on the style ActiveMenuHead, which is the same as MenuHead except for using a bold font. ‘font-weight:bold’.

EPiServer.WebControls.NewsList

The rationale for the NewsList class is to provide your Web site with an easy, yet powerful way to display news items.

Again looking at the Mimic Web site, we see that a NewsList control object is used on the start page.

NEWS
2/11/2004 11:39:21 AM
Mimic Almost as Good as Any Example
The Mimic web site has been found to equal any example web site in appearance.

2/11/2004 11:38:18 AM
To Be Reckoned with
Creators of web sites should pay close attention to Mimic.

2/11/2004 11:29:23 AM
Mimic Up And Coming
Web site Mimic is a strong contender says Reuters.

Figure 7-6: Using NewsList on Mimic's start page.

As NewsList is a templated control, the corresponding HTML code might seem a bit large.

Example 7-15: HTML code to render NewsList control (from the Mimic start page).

```

<table cellspacing="0" cellpadding="0" border="0" width="100%">
    <tr>
        <td>
            <EPiServer:Newslist ID=Newslistnew Pagelinkproperty="NewsContainer" runat="Server"
                MaxCount='<%# GetNewsCount() %>'>
            <HeaderTemplate>
                <tr>
                    <td height="15" background="images/L_triangleBG.gif">

```



```

        <a class="ListHeading" href="< %# Container.CurrentPage.LinkURL %>">
            < %# Container.CurrentPage.PageName.ToUpper() %>&nbsp;
        </a>
    </td>
</tr>
</HeaderTemplate>
<NewsTemplate>
    <tr>
        <td class="DateListingText">
            < %# Container.CurrentPage[ "PageStartPublish" ] %>
        </td>
    </tr>
    <tr>
        <td>
            <a href="< %# Container.CurrentPage.LinkURL %>"
                class="StartPageHeading">
                < %# Container.CurrentPage.PageName %>
            </a>&nbsp;<br />
            <span class="Normal">
                < %# Container.CurrentPage[ "MainIntro" ] %>
            </span>
        </td>
    </tr>
    <tr>
        <td><EPiServer:Clear height="6" runat="server" /></td>
    </tr>
</NewsTemplate>
<FooterTemplate>
    <tr>
        <td bgcolor="#DEDEDE"><EPiServer:Clear height="1" runat="server" /></td>
    </tr>
</FooterTemplate>
</EPiServer:Newslist>
</td>
</tr>
</table>

```

A discussion of the templates used in `NewsList` is presented on page 171 and following pages. Apart from these templates, the `NewsList` class only offers inherited attributes, methods and events.

EPiServer.WebControls.PageList

EPiServer is currently geared towards presentation information in a Web site context, which makes it only natural that so many classes and controls deal with handling Web pages. EPiServer.WebControls.PageList is one such page handling control. It is used to present a list of the Web pages that form the Web site. It's an easy way to render a list of references (HTML Anchor tags) to all the Web pages that belong to the site.

[Templates](#)
[Inspiration](#)
[Tips & Tricks](#)
[News Groups](#)
[EPiServer Portal](#)
[Links](#)

Figure 7-7: Example of PageList on Web page.

The PageList control object as seen in figure 7-7 was placed on a Web Form using the HTML code, as seen in example 7-16.

Example 7-16: EPiServer.WebControls.PageList used on a Web Form.

```
<EPiServer:PageList PageLink="<%# (EPiServer.PageBase) Page ).Configuration.StartPage %>"
  runat="server" ID="PageList1">
  <ItemTemplate>
    <tr>
      <td>
        <EPiServer:Property PropertyName="PageLink" runat="server" ID="Property2"/>
      </td>
    </tr>
  </ItemTemplate>
</EPiServer:PageList>
```

PageList has three templates: HeaderTemplate, ItemTemplate and FooterTemplate.

The PageLink property of PageList is used to link the control object to the proper place in the Web Page Tree. To link it to the current page, and its child pages, you would use 'PageLink=<%# CurrentPage.PageLink %>'. Instead.

EPiServer.WebControls.PageSearch

It is not searching for Pages but for information on Pages that PageSearch is used for. It also cooperates with Microsoft Index Server and thus allows searching outside of the Web site proper.

Search

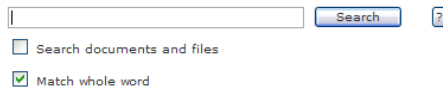
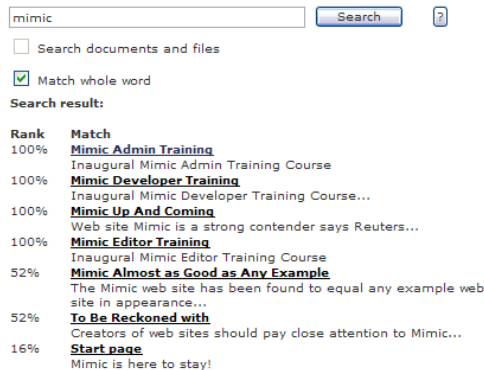


Figure 7-8: Using PageSearch on a Web Form Search.aspx.

A good example of using PageSearch is the Search.aspx Web Form that is shipped with EPiServer. Actually Search.aspx is simply a host for the Web User Control Search.ascx. As always, the reason for using both a User Control and a Web Form is the EPiServer Content Framework; the Web Form is an EPiServer Page which is using a Framework Definition File.

This is what it looks like when searching the Mimic Web site for the word ‘mimic’.

Search



Rank	Match
100%	Mimic Admin Training Inaugural Mimic Admin Training Course
100%	Mimic Developer Training Inaugural Mimic Developer Training Course...
100%	Mimic Up And Coming Web site Mimic is a strong contender says Reuters...
100%	Mimic Editor Training Inaugural Mimic Editor Training Course
52%	Mimic Almost as Good as Any Example The Mimic web site has been found to equal any example web site in appearance...
52%	To Be Reckoned with Creators of web sites should pay close attention to Mimic...
16%	Start page Mimic is here to stay!

Figure 7-9: Result presented when searching the Mimic Web site for the word ‘mimic’.

Looking at the screen dump in figure 7-9, you’d be quite right in assuming that the Search Web Custom Control Class is a templated control.

All the action takes place in the Web User Control, Search.ascx, and its code-behind file, Search.ascx.cs.

Example 7-17: Using EPiServer.WebControls.PageList in Web User Control Search.ascx.

```
<%@ Register TagPrefix="EPiServer" Namespace="EPiServer.WebControls" Assembly="EPiServer" %>
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="Search.ascx.cs"
    Inherits="development.Templates.Units.Search"
    TargetSchema="http://schemas.microsoft.com/intellisense/ie5" %>
```

```

<table cellpadding="2" border="0" xmlns:EPiServer="http://schemas.episerver.com/WebControls">
...
  <!-- Note: PageLink is the default start page for search, used if MainContainer is empty -->
  <EPiServer:PageSearch
    Runat="server"
    ID="SearchResults"
    SearchQuery='<%# SearchQuery.Text %>'
    SearchFiles='<%# SearchFiles.Checked %>'
    OnlyWholeWords='<%# OnlyWholeWords.Checked %>'
    MainScope='<%# CurrentPage[ "MainScope" ] %>'
    MainCatalog='<%# CurrentPage[ "MainCatalog" ] %>'
    PageLink='<%# Configuration.StartPage %>'
    PageLinkProperty="MainContainer"
  >
  <HeaderTemplate>
    <tr>
      <td colspan="2">
        <b>
          <EPiServer:Translate id="Translate6" runat="server"
            Text="/templates/search/searchresult" CssClass="EP-tableHeading" />
        </b>
      </td>
    </tr>
    <tr>
      <td colspan="2">&nbsp;</td>
    </tr>
    <tr>
      <td align="left" width="50">
        <b><EPiServer:Translate id="Translate4" runat="server"
          Text="/templates/search/rank" /></b>
      </td>
      <td>
        <b><EPiServer:Translate id="Translate5" runat="server"
          Text="/templates/search/match" /></b>
      </td>
    </tr>
  </HeaderTemplate>
  <ItemTemplate>
    <tr>
      <td align="left" width="50">
        <%# (int) Container.CurrentPage[ "PageRank" ] / 10 %>%
      </td>
      <td align="left">

```

```

        <b>
        <EPiServer:Property id="Property1" runat="server" PropertyName="PageLink" />
        </b>
    </td>
</tr>
<tr>
    <td>&nbsp;   </td>
    <td><%# Container.PreviewText %></td>
</tr>
</ItemTemplate>
<FileTemplate>
    <tr>
        <td align="left" width="50">
            <%# (int) Container.CurrentPage[ "PageRank" ] / 10 %>%</td>
        <td align="left">
            <img src='<%# Container.CurrentPage[ "IconPath" ] %>' >
            <a href='<%# Container.CurrentPage[ "PageLinkURL" ]%>'>
                <b><%# Container.CurrentPage.PageName %></b></a>
        </td>
    </tr>
</FileTemplate>
<NoMatchesTemplate>
    <tr>
        <td colspan="2">
            <br/>
            <br/>
            <EPiServer:Translate CssClass="EP-tableHeading"
                Text="/templates/search/nomatches" runat="server" ID="Translate3" />
        </td>
    </tr>
</NoMatchesTemplate>
</EPiServer:PageSearch>
</table>

```

Of course, there are many opportunities for those who like more control. These are some of the pertinent attributes in the PageList class:

Table 7-4: PageList attributes.

<i>Attribute Name</i>	<i>Description</i>
FileTemplate	Template for files
ItemTemplate	(Inherited from PageList) The default template for pages
MainCatalog	The index server catalog to search in.

Table 7-4: PageList attributes.

<i>Attribute Name</i>	<i>Description</i>
MainScope	The scope parameter for Index Server searches. The default is deep traversal under the catalog root.
MaxCount	(Inherited from PageListData) Restrict listing to a maximum number of pages.
NoMatchesTemplate	The template for no search matches
OnlyWholeWords	A flag to control if the search will match only whole words. If False, the search will match all words that begin with the words in the query (word* match).
SearchFiles	A flag to indicate whether or not files should be included in the search.
SearchQuery	Search query string, i.e. whatever the user types.

EPiServer.WebControls.PageTree

EPiServer.WebControls.PageTree is a direct descendant of EPiServer.WebControls.PageTreeData (see figure 7-1). PageTree is a templated control; templates are its only extension to PageTreeData. The names of the templates are self-explanatory:

- ❑ ExpandedItemTemplate
- ❑ ExpandedTopTemplate
- ❑ FooterTemplate
- ❑ HeaderTemplate
- ❑ ItemTemplate
- ❑ SelectedExpandedItemTemplate
- ❑ SelectedExpandedTopTemplate
- ❑ SelectedItemTemplate
- ❑ SelectedTopTemplate
- ❑ TopTemplate

FooterTemplate and HeaderTemplate are simply used to define the areas immediately above and below the PageTree control area.

PageTree is often used in conjunction with EPiServer.MenuList, where it uses the MenuList as its data source to create hierarchical menus.

One example of using PageTree, and Menu, is the DefaultFramework file shipped with EPiServer. The menu displayed along the left side of the Web page is created by using EPiServer.WebControls.PageTree and EPiServer.WebControls.MenuList in an ASP.NET User Control, Menu.ascx.

This is what the final result looks like:



Figure 7-10: Menu created by Web Control Menu.ascx using Custom Controls PageTree and MenuList. The right-hand picture is the result of expanding the item News.

Looking into Menu.ascx

Example 7-18: Menu.ascx.cs

```
namespace development.Templates.Units {
    /// <summary>Menu uses EPiServer.WebControls.MenuList
    /// and EPiServer.WebControls.PageTree to implement a
    /// vertical menu.</summary>
    public abstract class Menu : EPiServer.UserControlBase {
        protected EPiServer.WebControls.PageTree PageTreeControl;
        private EPiServer.WebControls.MenuList _menuListControl;

        private void Page_Load( object sender, System.EventArgs e ) {
            if ( MenuListControl != null ) {
                PageTreeControl.DataSource = MenuListControl;
            }
            if ( ! IsPostBack ) {
                PageTreeControl.DataBind();
            }
        }

        public EPiServer.WebControls.MenuList MenuListControl {
            set { _menuListControl = value; }
            get { return (EPiServer.WebControls.MenuList) _menuListControl; }
        }
    }
}
...
}
```

The HTML code looks like this (just browse it, a more readable version follows):

Example 7-19: Menu.ascx.

```

<%@ Control Language="c#" AutoEventWireup="false" Codebehind="Menu.ascx.cs"
    Inherits="development.Templates.Units.Menu"
    TargetSchema="http://schemas.microsoft.com/intellisense/ie5" %>
<%@ Register TagPrefix="EpiServer" Namespace="EpiServer.WebControls" Assembly="EpiServer" %>
<table cellpadding="0" cellspacing="0" border="0" width="160"
    xmlns:EpiServer="http://schemas.episerver.com/WebControls">
    <EpiServer:PageTree runat="server" id="PageTreeControl">
        <HeaderTemplate>
            <tr>
                <td valign="middle" height="15" colspan="6">
                    <a class="ListHeading" href="< %# Container.CurrentPage.LinkURL %>">
                        < %# Container.CurrentPage.PageName.ToUpper() %>
                        &nbsp; &nbsp; </a>
                </td>
            </tr>
            <tr>
                <td colspan="6"><EpiServer:Clear height="5" runat="server" /></td>
            </tr>
            <tr>
                <td width="12" height="1"></td>
                <td width="12" height="1"></td>
                <td width="12" height="1"></td>
                <td width="12" height="1"></td>
                <td width="12" height="1"></td>
                <td width="100" height="1"></td>
            </tr>
        </HeaderTemplate>
        <ExpandedItemTemplate>
            <tr>
                <td height="20" colspan='< %# Container.CurrentPage.Indent %>' align="right">
                    <asp:Image Visible="< %# Container.HasChildren %>"
                        ImageUrl="~/images/openMenuArrow.gif" Width="7" Height="7"
                        AlternateText="" runat="server" />
                    <EpiServer:Clear Visible="< %# ! Container.HasChildren %>" Width="7" Height="7"
                        runat="server" />
                    &nbsp; &nbsp;
                </td>
                <td height="20" colspan='< %# 6-Container.CurrentPage.Indent %>'>
                    <EpiServer:Property CssClass="MenuLink" runat="server"
                        PropertyName="PageLink" id="Property1" name="Property1" />
                </td>
            </tr>
        </ExpandedItemTemplate>
    </EpiServer:PageTree>
</table>

```



```

<tr>
  <td bgcolor="#dedede" colspan="6"><EPiServer:Clear runat="server" /></td>
</tr>
</ExpandedItemTemplate>
<ItemTemplate>
  <tr>
    <td height="20" colspan="< %# Container.CurrentPage.Indent%>" align="right">
      <asp:Image Visible="< %# Container.HasChildren %>"
        ImageUrl="~/images/closedMenuArrow.gif" Width="7" Height="7"
        AlternateText="" runat="server" />
      <EPiServer:Clear Visible="< %# ! Container.HasChildren %>" Width="7"
        Height="7" runat="server" />
      &nbsp;
    </td>
    <td height="20"
      colspan="< %# 6-Container.CurrentPage.Indent%>"
      <EPiServer:Property CssClass="MenuItem" runat="server"
        PropertyName="PageLink" id="Property2" name="Property2" />
    </td>
  </tr>
</tr>
  <td bgcolor="#dedede" colspan="6"><EPiServer:Clear runat="server" /></td>
</tr>
</ItemTemplate>
<SelectedItemTemplate>
  <tr bgcolor="#dedede">
    <td height="20" colspan="< %# Container.CurrentPage.Indent%>"
      align="right">
      <asp:Image Visible="< %# Container.HasChildren %>"
        ImageUrl="~/images/openMenuArrow.gif" Width="7" Height="7"
        AlternateText="" runat="server" />
      <EPiServer:Clear Visible="< %# ! Container.HasChildren %>" Width="7"
        Height="7" runat="server" />
      &nbsp;
    </td>
    <td height="20" colspan="< %# 6-Container.CurrentPage.Indent%>"
      <EPiServer:Property CssClass="MenuItem" runat="server"
        PropertyName="PageLink" ID="Property3" NAME="Property3" />
    </td>
  </tr>
</tr>
  <td bgcolor="#dedede" colspan="6"><EPiServer:Clear runat="server" /></td>
</tr>

```

```

        </SelectedItemTemplate>
    </EPiServer:PageTree>
</table>

```

Now, that's a mouthful. Let's simplify it:

Example 7-20: Menu.ascx in much simplified form.

```

<%@ Control Language="c#" AutoEventWireup="false" Codebehind="Menu.ascx.cs"
    Inherits="development.Templates.Units.Menu"
    TargetSchema="http://schemas.microsoft.com/intellisense/ie5" %>
<%@ Register TagPrefix="EPiServer" Namespace="EPiServer.WebControls" Assembly="EPiServer" %>
<table cellpadding="0" cellspacing="0" border="0" width="160"
    xmlns:EPiServer="http://schemas.episerver.com/WebControls">
    <EPiServer:PageTree runat="server" id="PageTreeControl">
        <HeaderTemplate>
            <tr>
                <td>
                    <td>
                        <a href="<%# Container.CurrentPage.LinkURL %>"
                            href="<%# Container.CurrentPage.PageName.ToUpper() %>"
                        </a>
                    </td>
                </tr>
            </HeaderTemplate>
            <ItemTemplate>
                <tr>
                    <td>
                        <a href="<%# Container.CurrentPage.LinkURL %>"
                            <%# Container.CurrentPage.PageName %>"
                        </a>
                    </td>
                </tr>
            </ItemTemplate>
        </EPiServer:PageTree>
    </table>

```

You probably agree that the latter is much easier to read. It still retains the basic functionality of the code in example 7-19.

INSPIRATION
News
Search
Calendar

INSPIRATION
News
Mimic Almost as Good as
Any Example
To Be Reckoned with
Mimic Up And Coming
Search
Calendar

Figure 7-11: Menu created by the simplified Web Control Menu.ascx using Custom Controls PageTree and MenuList. The right-hand picture is the result of expanding the item News.

Underlining is controlled by the style sheet, `episerver.css`, which by default underlines all links (anchor tags).

The original `Menu.ascx` (see example 7-19) uses four out of PageTree's ten templates: `HeaderTemplate`, to create a header for the menu; `ItemTemplate`, for all ordinary items whether they are mother or children items; `ExpandedItemTemplate` for selected items that are expanded, again whether or not they have any children items; and `SelectedItemTemplate` for the currently selected item.

There are several examples of elegant coding in example 7-19. One is the use of `EPiServer.WebControls.Property` instead of the HTML anchor tag.

Example 7-21: Using EPiServer.WebControls.Property instead of HTML anchor tag.

```
<EPiServer:Property runat="server" PropertyName="PageLink" ID="Property1" />
<!-- Equivalent to: -->
<a href="< %# Container.CurrentPage.LinkURL %>">< %# Container.CurrentPage.PageName %></a>
```

EPiServer.WebControls.Property

`Property` is probably the most frequently used of all control types in the `EPiServer.WebControls` name space. It provides access to all page properties, whether they are built-in, changes automatically or added in EPiServer Admin mode. The class name, 'Property', must be understood in an historical context. Had EPiServer been written in and for an object-oriented environment, such as Microsoft .NET, today, this class would most likely have been called 'Value' or perhaps 'Setting'. It will be interesting to see whether a name change is feasible for future versions of EPiServer.

Table 7-5: Attributes for EPiServer.WebControls.Property.

<i>Attribute Name</i>	<i>Description</i>
<code>DisplayMissingMessage</code>	Hide or show error message when property is missing
<code>Editable</code>	Determines whether the property is editable with DOPE
<code>EditMode</code>	Determines whether the property should render its edit mode
<code>InnerProperty</code>	Set or get the inner property used by this control
<code>PageLink</code>	The root page to read data from, if different from current
<code>PageLinkProperty</code>	The property that contains the root page to read data from, if different from current
<code>PageSource</code>	Returns the <code>IPageSource</code> implementation that this property control uses to read page data

Table 7-5: Attributes for EPiServer.WebControls.Property.

<i>Attribute Name</i>	<i>Description</i>
PropertyName	The name of the property to load automatically
PropertyValue	The value of the loaded property

Property objects, like Content and Region objects, are most often found in HTML part of Web Forms and User Controls, but they also are used to a great extent in code-behind files.

Properties on pages are collected in an EPiServer.Core.PropertyDataCollection collection called Property, provided that the template used implements IPageSource (see page 153).

Property objects used in HTML only have to include the name of the Property:

Example 7-22: Using a Property object to display contents of Property MainBodyHeading.

```
<EPiServer:Property PropertyName="MainBodyHeading"
  DisplayMissingMessage="false" class="ListHeading" runat="server" />
```

Example 7-23: Display the name of the page.

```
<EPiServer:Property propertyname="PageName" runat="server" />
```

For the Property control to be able to read the properties from the page, it needs to be hosted on a Web form or a control that implements the IPageSource interface. The control will iterate through the control hierarchy looking for this interface, and when it finds one, it will use the CurrentPage property to read the information about the specific built-in or custom property.

If you put a Property control inside a templated control like the PageList control which implements IPageSource, the Property control will use the CurrentPage property of the template control instead. The PageList then points the Property control to the current PageData object in its internal PageDataCollection. This is why the two following examples will print the same:

Example 7-24: Using Property inside a templated control.

```
<EPiServer:PageList PageLink=<%# CurrentPage.Configuration.StartPage %> runat="server">
  <ItemTemplate>
    <EPiServer:Property PropertyName="PageName" runat="server"/>
  </ItemTemplate>
</EPiServer:PageList>
```

Example 7-25: Using Container inside a templated control.

```
<EPiServer:PageList PageLink=<%# CurrentPage.Configuration.StartPage %> runat="server">
```

```

<ItemTemplate>
  <%# Container.CurrentPage.PageName %>
</ItemTemplate>
</epi:PageList>

```

The Property control will also give you DOPE (Direct On Page Editing) support if the underlying template page supports it (meaning any Web page that inherits directly or indirectly from TemplatePage). If you do not want DOPE support, you can either inherit from SimplePage or set the Editable attribute to false, like this:

Example 7-26: Switching off DOPE support for an EPiServer Property.

```
<EPiServer:Property PropertyName="PageName" runat="server" Editable='false' />
```

Note that DOPE support will only be available for properties on the current page, not properties rendered inside a PageList or any other templated control.

EPiServer.WebControls.PropertyCriteriaControl

The HTML code in example 7-27 uses a PropertySearch object as data source for a PageList object. Enclosed in the PropertySearch object are two PropertyCriteriaControl objects, specifying that the search proper concerns the Property PageName and that the contents of the PageName properties should start with one of the letters 'A', 'a', 'B' or 'b'. (Compare this with example 7-28, which puts the logic in the code-behind file.)

Example 7-27: Using PropertySearch, PropertyCriteriaControl and PageList to produce a list of pages meeting certain criteria.

```

<%@ Register TagPrefix="EPiServer" Namespace="EPiServer.WebControls" Assembly="EPiServer" %>
...
<EPiServer:PropertySearch PageLink=<%# Configuration.StartPage %> runat="server"
  ID="PropSearch" >
  <EPiServer:PropertyCriteriaControl Name="PageName" Value="a" Type="String"
    StringCondition="StartsWith" />
  <EPiServer:PropertyCriteriaControl Name="PageName" Value="b" Type="String"
    StringCondition="StartsWith" />
</EPiServer:PropertySearch>
<EPiServer:PageList SortBy="PageName" DataSource="<%# PropSearch %>" runat="server"
  ID="PL">
  <ItemTemplate>
  <tr>
    <td><EPiServer:Property PropertyName="PageName" runat="server" ID="Prop3" /></td>
  </tr>
  </ItemTemplate>

```

</EPiServer:PageList>

EPiServer.WebControls.PropertySearch

Having a lot of Web Pages in the Web Page and perhaps a lot of information on those pages makes it absolutely necessary to be able to search these pages using almost any attributes and any number of search criteria. PropertySearch is used when the capabilities of PageSearch (see page 183) just aren't enough. In addition, PropertySearch is not limited to searching for, or in, Web Pages in the Web Page Tree; it searches all of page database, in the database table tblProperty (see page 315). So pages such as user registrations and other kinds are part of the search. (For an illustration, open the example Web site and look at its site map, which shows only pages in the Web Page Tree. Then open Templates and select 'Alphabetical table of contents'. In the latter case, PropertySearch is used and thus all pages are returned, not just those in the Web Page Tree.)

Central to PropertySearch is its attribute *Criteria*s (this is the only attribute which is not inherited). *Criteria*s is of the type `EPiServer.PropertyCriteriaCollection`, i.e. a collection of `EPiServer.PropertyCriteria`. Proper use of the *PropertyCriteria* objects in *Criteria*s is key to successfully using PropertySearch.

EPiServer.PropertyCriteria

PropertyCriteria inherits from `System.Object` and extends the mother class only by adding seven attributes. It is used for two different kinds of searches: string and non-string searches. When used for string searches, the search condition attribute `StringCondition` is employed whilst for non-string searches `Condition` is the attribute of choice.

Table 7-6: PropertyCriteria attributes.

<i>Attribute Name</i>	<i>Description</i>
Condition	The type of comparison condition. <code>EPiServer.Filters.CompareCondition</code> , specifies <code>Equal</code> , <code>NotEqual</code> , <code>GreaterThan</code> and <code>LessThan</code> .
IsNull	Test for value set to null. Boolean.
Name	Name of property. String.
Required	Determines whether this criterion is required for a match. Boolean. Default is false, meaning that this criterion is logically <code>Or</code> :ed with other criteria.
StringCondition	Comparison of strings when <code>Equal</code> is used. Type is <code>EPiServer.Filters.StringCompareMethod</code> (<code>Identical</code> , <code>StartsWith</code> , <code>EndsWith</code> and <code>Contained</code>). Comparison is case insensitive.

Table 7-6: PropertyCriteria attributes.

<i>Attribute Name</i>	<i>Description</i>
Type	Type of criterion, one of EPiServer.Core.PropertyType enumeration.
Value	Value of criterion. String. If you're using Microsoft SQL Server, its wild cards '%' and '?' may be used.

Using PropertySearch

Example 7-28: Using PropertyCriteria when searching for Web Pages by their name (from templates\Units\AlphanumericListing.ascx.cs).

```

...
protected EPiServer.WebControls.PropertySearchPropertySearchControl;
...
protected void ChangeLetters( object sender, System.EventArgs e ) {
...
    AddLetter( "A" );
    AddLetter( "B" );
    AddLetter( "C" );
    AddLetter( "D" );
    AddLetter( "E" );
    AddLetter( "F" );
...
    DataBind();
}

private void AddLetter( string letter ) {
    EPiServer.PropertyCriteria criterion = new EPiServer.PropertyCriteria();
    criterion.StringCondition = EPiServer.Filters.StringCompareMethod.StartsWith;
    criterion.Type = EPiServer.Core.PropertyType.String;
    criterion.Value = letter;
    criterion.Name = "PageName";
    PropertySearchControl.Criteria.Add( criterion );
}

```

The code in example 7-28 is extracted from the code-behind file of Web User Control AlphanumericListing, AlphanumericListing.ascx.cs.

Taking a look at the function AddLetter, we see that it's building up for a string search; criterion.StringCondition is given a value rather than criterion.Condition. Moreover, we're using a string value as a search criterion, 'criterion.Type = EPiServer.Core.PropertyType.String'; the string value proper is set to the formal argument 'letter' passed to AddLetter. The EPiServer Property which

we'll be comparing `criterion.Value` against `PageName`. In other words, the search is performed against every Web Page comparing EPiServer Property `PageName` to the criteria in `Criteria`s.

In this particular instance, you can also see that `AddLetter` is called multiple times. As `criterion.Required` is not altered, this means that `PageName` is matched against any of 'A' through 'F'. Setting `criterion.Required` to true for this kind of comparison would be non-sensical (`criterion.StringCondition` is set to `EPiServer.Filters.StringCompareMethod.StartsWith`), but for other kinds it is essential, e.g. when searching for Properties which should contain more than one letter.

The HTML code in `AlphanumericListing.ascx` is quite simple.

Example 7-29: AlphanumericListing.ascx.

```
<%@ Register TagPrefix="EPiServer" Namespace="EPiServer.WebControls" Assembly="EPiServer" %>
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="AlphanumericListing.ascx.cs"
    Inherits="development.Templates.Units.AlphanumericListing"
    TargetSchema="http://schemas.microsoft.com/intellisense/ie5"%>
<table border="0" height="90%" cellpadding="0" cellspacing="0"
    xmlns:EPiServer="http://schemas.episerver.com/WebControls">
  <tr>
    <td valign="top">
      <table bgcolor="#cccccc" width="400" cellpadding="2" cellspacing="1">
        <tr>
          <td bgcolor="#cccccc" Class="Heading2">
            [<asp:LinkButton ID="Alphanumeric1" Runat="server"
              OnClick="ChangeLetters" CssClass="Heading2">a-f</asp:LinkButton>]&nbsp;
            [<asp:LinkButton ID="Alphanumeric2" Runat="server"
              OnClick="ChangeLetters" CssClass="Heading2">g-l</asp:LinkButton>]&nbsp;
            [<asp:LinkButton ID="Alphanumeric3" Runat="server"
              OnClick="ChangeLetters" CssClass="Heading2">m-r</asp:LinkButton>]&nbsp;
            [<asp:LinkButton ID="Alphanumeric4" Runat="server"
              OnClick="ChangeLetters" CssClass="Heading2">s-z</asp:LinkButton>]
          </td>
        </tr>
      </table>
      <EPiServer:PropertySearch PageLink="<%# EPiServer.Global.EPConfig.StartPage %>"
        runat="server" ID="PropertySearchControl" />
      <EPiServer:PageList SortBy="PageName"
        DataSource="<%# PropertySearchControl %>"runat="server"
        ID="PageListControl">
        <ItemTemplate>
          <tr>
            <td bgColor="#ffffff">
              <EPiServer:Property id="Property1" runat="server"
                PropertyName="PageLink" />
            </td>
          </tr>
        </ItemTemplate>
      </EPiServer:PageList>
    </td>
  </tr>
</table>
```



```

        </td>
      </tr>
    </ItemTemplate>
  </EPiServer:PageList>
</table>
</td>
</tr>
</table>

```

EPiServer.WebControls.Region

Framework Definition Files is the only type of place where you'll find EPiServer.WebControls.Region objects. Region objects are used to define areas in a Framework Definition File which may have their contents exchanged by Content object in Page Template Files (yes, we're talking about the EPiServer Content Framework).

As such, Region objects are never instantiated in code-behind files, they simply have no purpose there.

Example 7-30: Use of EPiServer.WebControls.Region in a Framework Definition File.

```

<%@ Control Language="c#" ... %>
<%@ Register TagPrefix="EPiServer" Namespace="EPiServer.WebControls" Assembly="EPiServer" %>
<%@ Register TagPrefix="development" TagName="LeftMenu" Src="~/templates/Units/Menu.ascx"%>
...
<EPiServer:Region id="menuRegion" runat="server">
    <development:LeftMenu id="LeftMenu" runat="server"/>
    <br/>
</EPiServer:Region>

```

Region control objects may be nested to any depth you need.

Example 7-31: Nested Region control objects.

```

<%@ Control Language="c#" ... %>
<%@ Register TagPrefix="EPiServer" Namespace="EPiServer.WebControls" Assembly="EPiServer" %>
<%@ Register TagPrefix="development" TagName="LeftMenu" Src="~/templates/Units/Menu.ascx"%>
...
<EPiServer:Region id="fullRegion" runat="server">
    <table>
        <tr>
            <td>
                <EPiServer:Region id="menuRegion" runat="server">
                    <development:LeftMenu id="LeftMenu" runat="server" />
                    <br/>

```

```

                </EPiServer:Region>
            </td>
...
        </tr>
    </table>
</EPiServer:Region>

```

The only important attribute for Region controls is ID, the unique identifier. Only by using the proper ID string can a Content control exchange its own contents for that of a pre-defined Region.

EPiServer.WebControls.SiteMap

It's not surprising that EPiServer ships with such an easy way to produce a site map as using SiteMap control object.

SiteMap is used in just two ASP.NET objects in the example Web site: the Page Template File SiteMap.aspx and the Web User Control SiteMap.ascx (in the template\Units folder).

The Page Template File SiteMap.aspx is not very exciting; its code-behind file is all but empty. The HTML contents look like this:

Example 7-32: HTML contents of Page Template File SiteMap.aspx.

```

<%@ Page language="c#" Codebehind="SiteMap.aspx.cs" AutoEventWireup="false"
    Inherits="development.Templates.SiteMap" %>
<%@ Register TagPrefix="EPiServer" Namespace="EPiServer.WebControls" Assembly="EPiServer" %>
<%@ Register TagPrefix="development" TagName="DefaultFramework"
    Src="~/templates/Frameworks/DefaultFramework.ascx"%>
<%@ Register TagPrefix="development" TagName="Sitemap"
    Src="~/templates/Units/Sitemap.ascx"%>

<development:DefaultFramework ID="defaultframework" runat="server">
    <EPiServer:Content Region="fullRegion" ID="SiteMapContent" runat="server">
        <development:Sitemap id="Sitemap" runat="server"></development:Sitemap>
    </EPiServer:Content>
</development:DefaultFramework>

```

As can be seen in example 7-32, the Page Template File SiteMap.aspx replaces everything in the Region 'fullRegion' with the Web User Control SiteMap.ascx.

EPiServer.WebControls.Translate

In order to make your EPiServer solution as portable as possible, you should use the Translate class. With Translate and a set of language files, you can furnish groups of users with an EPiServer solution which literally speaks their own language.

Translate is a direct descendant of `System.Web.UI.WebControls.WebControl` (see figure 7-1).

There are only three attributes that Translate implements itself:

Table 7-7: Attributes implemented by EPiServer.Translate.

<i>Attribute Name</i>	<i>Description</i>
LocalizedText	Used to access the text to be translated
StringFormatObjects	Set object references if the localized text is a format string with '{n}' style instructions
Text	The property that holds the text to be translated.

Use the Translate control to translate your own strings or one of the built-in strings in the `/lang` folder of your site. The text displayed will vary according to the current user's chosen language (if chosen) or the system default language.

The control uses the `LanguageManager` to translate the text. You can access the `LanguageManager` through code using the `Global.EPLang` static property.

Example 7-33: Using EPiServer.Translate in HTML part of a form or control.

```
<EPiServer:Translate text="/button/search" runat="server"/>
```

This will show the translated value for the `/button/search` language resource.

Translating ASP Intrinsic controls

Translate can also be used to translate ASP.NET control objects such as the Label Web control. In the `PreRender` event, EPiServer will search the Controls hierarchy for controls with a 'Translate' attribute (case insensitive) and subsequently translate the textual value of the control.

Example 7-34: Using Translate attribute in an ASP.NET control.

```
<asp:button id="QuicksearchButton" runat="server" translate="/button/search" />
```

Automatic translation can be turned off by setting the `AutomaticTranslation` property in your template to false.

