**ASP.NET Controls**

ASP.NET Web server controls are objects on ASP.NET Web pages that run when the page is requested and that render markup to a browser. Many Web server controls resemble familiar HTML elements, such as buttons and text boxes. Other controls encompass complex behavior, such as a calendar controls, and controls that manage data connections.

The topics in this section describe what ASP.NET Web controls are and how to work with them.

http://i.msdn.microsoft.com/Global/Images/clear.gif In This Section

[ASP.NET User Controls](http://msdn.microsoft.com/en-us/library/y6wb1a0e.aspx)

[ASP.NET Web Server Controls](http://msdn.microsoft.com/en-us/library/bb386416.aspx)

[ASP.NET Web Parts Controls](http://msdn.microsoft.com/en-us/library/e0s9t4ck.aspx)

.NET Framework 4 - ASP.NET

**ASP.NET User Controls Overview**

At times, you might need functionality in a control that is not provided by the built-in ASP.NET Web server controls. In those cases, you can create your own controls. You have two options. You can create:

* User controls. User controls are containers into which you can put markup and Web server controls. You can then treat the user control as a unit and define properties and methods for it.
* Custom controls. A custom control is a class that you write that derives from [[System.Web.UI](http://msdn.microsoft.com/en-us/library/system.web.ui.aspx).Control](http://msdn.microsoft.com/en-us/library/system.web.ui.control.aspx) or [System.Web.UI.WebControls](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.aspx).[WebControl](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.webcontrol.aspx).

User controls are substantially easier to create than custom controls, because you can reuse existing controls. They make it particularly easy to create controls with complex user interface elements.

This topic provides an overview of working with ASP.NET user controls.

http://i.msdn.microsoft.com/Global/Images/clear.gif User Control Structure

An ASP.NET Web user control is similar to a complete ASP.NET Web page (.aspx file), with both a user interface page and code. You create the user control in much the same way you create an ASP.NET page and then add the markup and child controls that you need. A user control can include code to manipulate its contents like a page can, including performing tasks such as data binding.

A user controls differs from an ASP.NET Web page in these ways:

* The file name extension for the user control is .ascx.
* Instead of an [@ Page](http://msdn.microsoft.com/en-us/library/ydy4x04a.aspx) directive, the user control contains an [@ Control](http://msdn.microsoft.com/en-us/library/d19c0t4b.aspx) directive that defines configuration and other properties.
* User controls cannot run as stand-alone files. Instead, you must add them to ASP.NET pages, as you would any control.
* The user control does not have **html**, **body**, or **form** elements in it. These elements must be in the hosting page.

You can use the same HTML elements (except the **html**, **body**, or **form** elements) and Web controls on a user control that you do on an ASP.NET Web page. For example, if you are creating a user control to use as a toolbar, you can put a series of [Button](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.button.aspx) Web server controls onto the control and create event handlers for the buttons.

The following example shows a user control that implements a spinner control where users can click up and down buttons to rotate through a series of choices in a text box.

|  |
| --- |
| **Security noteSecurity Note** |
| This example has a text box that accepts user input, which is a potential security threat. By default, ASP.NET Web pages validate that user input does not include script or HTML elements. For more information, see [Script Exploits Overview](http://msdn.microsoft.com/en-us/library/w1sw53ds.aspx). |

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl34_ctl00_ctl05_code');" \o "Copy Code)

<%@ Control Language="VB" ClassName="UserControl1" %>

<script runat="server">

Protected colors As String() = {"Red", "Green", "Blue", "Yellow"}

Protected currentColorIndex As Integer = 0

Protected Sub Page\_Load(ByVal sender As Object, \_

ByVal e As System.EventArgs)

If IsPostBack Then

currentColorIndex = CInt(ViewState("currentColorIndex"))

Else

currentColorIndex = 0

DisplayColor()

End If

End Sub

Protected Sub DisplayColor()

textColor.Text = colors(currentColorIndex)

ViewState("currentColorIndex") = currentColorIndex.ToString()

End Sub

Protected Sub buttonUp\_Click(ByVal sender As Object, \_

ByVal e As System.EventArgs)

If currentColorIndex = 0 Then

currentColorIndex = colors.Length - 1

Else

currentColorIndex -= 1

End If

DisplayColor()

End Sub

Protected Sub buttonDown\_Click(ByVal sender As Object, \_

ByVal e As System.EventArgs)

If currentColorIndex = colors.Length - 1 Then

currentColorIndex = 0

Else

currentColorIndex += 1

End If

DisplayColor()

End Sub

</script>

<asp:TextBox ID="textColor" runat="server"

ReadOnly="True" />

<asp:Button Font-Bold="True" ID="buttonUp" runat="server"

Text="^" OnClick="buttonUp\_Click" />

<asp:Button Font-Bold="True" ID="buttonDown" runat="server"

Text="v" OnClick="buttonDown\_Click" />

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl34_ctl00_ctl06_code');" \o "Copy Code)

<% @ Control Language="C#" ClassName="UserControl1" %>

<script runat="server">

protected int currentColorIndex;

protected String[] colors = {"Red", "Blue", "Green", "Yellow"};

protected void Page\_Load(object sender, EventArgs e)

{

if (IsPostBack)

{

currentColorIndex =

Int16.Parse(ViewState["currentColorIndex"].ToString());

}

else

{

currentColorIndex = 0;

DisplayColor();

}

}

protected void DisplayColor()

{

textColor.Text = colors[currentColorIndex];

ViewState["currentColorIndex"] = currentColorIndex.ToString();

}

protected void buttonUp\_Click(object sender, EventArgs e)

{

if(currentColorIndex == 0)

{

currentColorIndex = colors.Length - 1;

}

else

{

currentColorIndex -= 1;

}

DisplayColor();

}

protected void buttonDown\_Click(object sender, EventArgs e)

{

if(currentColorIndex == (colors.Length - 1))

{

currentColorIndex = 0;

}

else

{

currentColorIndex += 1;

}

DisplayColor();

}

</script>

<asp:TextBox ID="textColor" runat="server"

ReadOnly="True" />

<asp:Button Font-Bold="True" ID="buttonUp" runat="server"

Text="^" OnClick="buttonUp\_Click" />

<asp:Button Font-Bold="True" ID="buttonDown" runat="server"

Text="v" OnClick="buttonDown\_Click" />

Notice that the user control looks much like an ASP.NET page — it contains several controls (a [TextBox](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.textbox.aspx) control and two [Button](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.button.aspx) controls) and code that handles the buttons' [Click](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.button.click.aspx) events and the page’s [Load](http://msdn.microsoft.com/en-us/library/system.web.ui.control.load.aspx) event. However, the control contains no markup except for the controls, and instead of an [@ Page](http://msdn.microsoft.com/en-us/library/ydy4x04a.aspx) directive it contains an [@ Control](http://msdn.microsoft.com/en-us/library/d19c0t4b.aspx) directive.

http://i.msdn.microsoft.com/Global/Images/clear.gif Adding a User Control to a Page

You add a user control to a page by registering it on the host page. When you register it, you specify the .ascx file that contains the user control, a tag prefix, and a tag name that you will use to declare the user control on the page. For details, see [How to: Include a User Control in an ASP.NET Web Page](http://msdn.microsoft.com/en-us/library/sbz9etab.aspx).

http://i.msdn.microsoft.com/Global/Images/clear.gif Defining Properties and Methods for a User Control

You can define properties and methods for a user control the same way you do for a page. By defining a property for a user control, you make it possible to set its properties declaratively and in code.

http://i.msdn.microsoft.com/Global/Images/clear.gif Events in User Controls

When a user control contains Web server controls, you can write code in the user control to handle the events raised by the child controls. For example, if your user control contains a [Button](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.button.aspx) control, you can create a handler in the user control for the button's [Click](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.button.click.aspx) event.

By default, events raised by child controls in a user control are not available to the host page. However, you can define events for your user control and raise them so that the host page is notified of the event. You do this in the same way that you define events for any class. For more information, see [Raising an Event](http://msdn.microsoft.com/en-us/library/wkzf914z.aspx).

http://i.msdn.microsoft.com/Global/Images/clear.gif Referencing External Resources

When a user control runs, references to external resources such as images or anchors to other pages are resolved using the URL of the user control as the base URL. For example, if the user control contains an [Image](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.image.aspx) control whose [ImageUrl](http://msdn.microsoft.com/en-us/library/system.web.ui.mobilecontrols.image.imageurl.aspx) property is set to Images/Button1.gif, the URL of the image is added to the URL of the user control to resolve the complete path to the image. If the user control references a resource that is not in a subfolder of the user control itself, you must provide a path that resolves to the correct folder at run time. For more information on specifying paths for ASP.NET server controls, see [ASP.NET Web Project Paths](http://msdn.microsoft.com/en-us/library/ms178116.aspx).

http://i.msdn.microsoft.com/Global/Images/clear.gif Caching and User Controls

User controls can support caching directives that are separate from the host page. You can therefore add user controls to pages and to cache portions of a page. For details, see [Caching Portions of an ASP.NET Page](http://msdn.microsoft.com/en-us/library/h30h475z.aspx).

.NET Framework 4 - ASP.NET

**ASP.NET Web Server Controls Overview**

When you create ASP.NET Web pages, you can use these types of controls:

* **HTML server controls**   HTML elements exposed to the server so you can program them. HTML server controls expose an object model that maps very closely to the HTML elements that they render.
* **Web server controls**   Controls with more built-in features than HTML server controls. Web server controls include not only form controls such as buttons and text boxes, but also special-purpose controls such as a calendar, menus, and a tree view control. Web server controls are more abstract than HTML server controls in that their object model does not necessarily reflect HTML syntax.
* **Validation controls**   Controls that incorporate logic to enable you to what users enter for input controls such as the [TextBox](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.textbox.aspx) control. Validation controls enable you to check for a required field, to test against a specific value or pattern of characters, to verify that a value lies within a range, and so on. For more information, see [ASP.NET Validation Controls](http://msdn.microsoft.com/en-us/library/debza5t0.aspx).
* **User controls**   Controls that you create as ASP.NET Web pages. You can embed ASP.NET user controls in other ASP.NET Web pages, which is an easy way to create toolbars and other reusable elements. For more information, see [ASP.NET User Controls](http://msdn.microsoft.com/en-us/library/y6wb1a0e.aspx).

|  |
| --- |
| **NoteNote** |
| You can also create output for mobile devices. To do so, you use the same ASP.NET page framework, but you create Mobile ASP.NET Web pages instead of ASP.NET Web pages and use controls specifically designed for mobile devices. For details, see [Creating ASP.NET Mobile Web Pages](http://msdn.microsoft.com/en-us/library/8htez1ds.aspx). |

You can use all types of controls on the same page. The following sections provide more detail about ASP.NET server controls.

|  |
| --- |
| **NoteNote** |
| In some situations, server controls require client script in order to function properly. If a user has disabled scripting in the browser, the controls might not function as you intend. For details, see [ASP.NET Web Server Controls and Browser Capabilities](http://msdn.microsoft.com/en-us/library/x3k2ssx2.aspx). |

http://i.msdn.microsoft.com/Global/Images/clear.gif HTML Server Controls

HTML server controls are HTML elements (or elements in other supported markup, such as XHTML) containing attributes that make them programmable in server code. By default, HTML elements on an ASP.NET Web page are not available to the server. Instead, they are treated as opaque text and passed through to the browser. However, by converting HTML elements to HTML server controls, you expose them as elements you can program on the server.

The object model for HTML server controls maps closely to that of the corresponding elements. For example, HTML attributes are exposed in HTML server controls as properties.

Any HTML element on a page can be converted to an HTML server control by adding the attribute **runat="server"**. During parsing, the ASP.NET page framework creates instances of all elements containing the **runat="server"** attribute. If you want to reference the control as a member within your code, you should also assign an **id** attribute to the control.

The page framework provides predefined HTML server controls for the HTML elements most commonly used dynamically on a page: the **form** element, the **input** elements (text box, check box, Submit button), the **select** element, and so on. These predefined HTML server controls share the basic properties of the generic control, and in addition, each control typically provides its own set of properties and its own event.

HTML server controls offer the following features:

* An object model that you can program against on the server using familiar object-oriented techniques. Each server control exposes properties that enable you to manipulate the control's markup attributes programmatically in server code.
* A set of events for which you can write event handlers in much the same way you would in a client-based form, except that the event is handled in server code.
* The ability to handle events in client script.
* Automatic maintenance of the control's state. When the page makes a round trip to the server, the values that the user entered into HTML server controls are automatically maintained and sent back to the browser.
* Interaction with ASP.NET validation controls so you can verify that a user has entered appropriate information into a control.
* Data binding to one or more properties of the control.
* Support for styles if the ASP.NET Web page is displayed in a browser that supports cascading style sheets.
* Pass-through of custom attributes. You can add any attributes you need to an HTML server control and the page framework will render them without any change in functionality. This enables you to add browser-specific attributes to your controls.

For details about how to convert an HTML element to an HTML server control, see [How to: Add HTML Server Controls to a Web Page Using ASP.NET Syntax](http://msdn.microsoft.com/en-us/library/s37470tf.aspx).

http://i.msdn.microsoft.com/Global/Images/clear.gif Web Server Controls

Web server controls are a second set of controls designed with a different emphasis. They do not necessarily map one-to-one to HTML server controls. Instead, they are defined as abstract controls in which the actual markup rendered by the control can be quite different from the model that you program against. For example, a [RadioButtonList](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.radiobuttonlist.aspx) Web server control might be rendered in a table or as inline text with other markup.

Web server controls include traditional form controls such as buttons and text boxes as well as complex controls such as tables. They also include controls that provide commonly used form functionality such as displaying data in a grid, choosing dates, displaying menus, and so on.

Web server controls offer all of the features described above for HTML server controls (except one-to-one mapping to elements) and these additional features:

* A rich object model that provides type-safe programming capabilities.
* Automatic browser detection. The controls can detect browser capabilities and render appropriate markup.
* For some controls, the ability to define your own layout for the control using [Templates](http://msdn.microsoft.com/en-us/library/system.web.ui.design.templategroup.templates.aspx).
* For some controls, the ability to specify whether a control's event causes immediate posting to the server or is instead cached and raised when the page is submitted.
* Support for themes, which enable you to define a consistent look for controls throughout your site. For details, see [ASP.NET Themes and Skins](http://msdn.microsoft.com/en-us/library/wcyt4fxb.aspx).
* Ability to pass events from a nested control (such as a button in a table) to the container control.

The controls use syntax such as the following:

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl25_ctl00_ctl03_code');" \o "Copy Code)

<asp:button attributes runat="server" id="Button1" />

The attributes in this case are not those of HTML elements. Instead, they are properties of the Web control.

When the ASP.NET Web page runs, the Web server control is rendered on the page using appropriate markup, which often depends not only on the browser type but also on settings that you have made for the control. For example, a [TextBox](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.textbox.aspx) control might render as an **input** tag or a **textarea** tag, depending on its properties.

Detailed how-to and reference documentation is available for each control separately. For more information, see [Individual ASP.NET Web Server Controls](http://msdn.microsoft.com/en-us/library/fxh7k08z.aspx).

.NET Framework 4 - ASP.NET

**ASP.NET Web Parts Overview**

ASP.NET Web Parts is an integrated set of controls for creating Web sites that enable end users to modify the content, appearance, and behavior of Web pages directly from a browser. The modifications can be applied to all users on the site or to individual users. When users modify pages and controls, the settings can be saved to retain a user's personal preferences across future browser sessions, a feature called personalization. These Web Parts capabilities mean that developers can empower end users to personalize a Web application dynamically, without developer or administrator intervention.

Using the Web Parts control set, you as a developer can enable end users to:

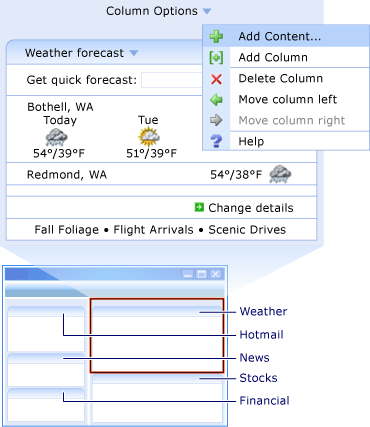
* Personalize page content. Users can add new Web Parts controls to a page, remove them, hide them, or minimize them like ordinary windows.
* Personalize page layout. Users can drag a Web Parts control to a different zone on a page, or change its appearance, properties, and behavior.
* Export and import controls. Users can import or export Web Parts control settings for use in other pages or sites, retaining the properties, appearance, and even the data in the controls. This reduces data entry and configuration demands on end users.
* Create connections. Users can establish connections between controls so that, for example, a chart control could display a graph for the data in a stock ticker control. Users could personalize not only the connection itself, but the appearance and details of how the chart control displays the data.
* Manage and personalize site-level settings. Authorized users can configure site-level settings, determine who can access a site or page, set role-based access to controls, and so on. For example, a user in an administrative role could set a Web Parts control to be shared by all users, and prevent users who are not administrators from personalizing the shared control.

 Web Parts Essentials

The Web Parts control set consists of three main building blocks: personalization, user interface (UI) structural components, and actual Web Parts UI controls. For more details, see [Web Parts Control Set Overview](http://msdn.microsoft.com/en-us/library/k3w2y2tf.aspx). Much of your development effort will focus on Web Parts controls, which are simply ASP.NET controls that can use the features of the Web Parts control set.

As an example of how Web Parts controls can be used to build personalizable Web pages, examine the following screen shot.

**Typical Web Parts page**



This page contains several basic elements of a Web Parts application:

* Use of zones for page layout. There are two columns that can contain controls: one has the Weather and Stock Quotes controls, the other has Hotmail and News controls. These columns in Web Parts terminology are called zones--regions on a page that contain Web Parts controls. Zones exist to lay out Web Parts controls on a page, and to provide a common UI for the controls. There can be one or many zones on a page, each zone can contain one or many Web Parts controls, and each zone can have a vertical or horizontal orientation for page layout.
* Web Parts controls within the zones. Each control has UI verbs (actions that a user can perform) that can appear as links, buttons, or clickable images on the control. In the preceding screen shot, notice that each control has a button in its title bar that exposes a drop-down menu. In the menus for each control are options to change details particular to that control, and other options to carry out common actions such as moving or deleting a control, and getting help. Some controls, such as the Weather control, allow users to personalize them so the controls display only information relevant to the user.
* Links to enable extensive personalization. These allow users to change the content, color, and layout of the page. For instance, if users click the **Add Column** link, a Web Parts application could enable them to add another column to a page. Or users could click the **Add Content** link, which displays a catalog of controls that that they can optionally add to the page. One of those could be a stock charting control. A user could add that control to one of the zones on the page, and could then connect it to the existing Stock Quotes control to chart the stock data it contains.

 Developer Scenarios for Using Web Parts

You will typically work with Web Parts in one of three ways: creating pages that use Web Parts controls, creating individual Web Parts controls, or creating complete, personalizable Web applications, such as a portal.

**Page Development**

Page developers can use visual design tools such as Microsoft Visual Studio 2005 to create pages that use Web Parts. One advantage in using a tool such as Visual Studio is that the Web Parts control set provides features for drag-and-drop creation and configuration of Web Parts controls in a visual designer. For example, you can use the designer to drag a Web Parts zone, or a Web Parts editor control, onto the design surface, and then configure the control right in the designer using the UI provided by the Web Parts control set. This can speed development of Web Parts applications and reduce the amount of code you have to write.

**Control Development**

You can use any existing ASP.NET control as a Web Parts control, including standard Web server controls, custom server controls, and user controls. For maximum programmatic control of your environment, you can also create custom Web Parts controls that derive from the [WebPart](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.webparts.webpart.aspx) class. For individual Web Parts control development, you will typically either create a user control and use it as a Web Parts control, or develop a custom Web Parts control.

As an example of developing a custom Web Parts control, you could create a control to provide any of the features provided by other ASP.NET server controls that might be useful to package as a personalizable Web Parts control: calendars, lists, financial information, news, calculators, rich text controls for updating content, editable grids that connect to databases, charts that dynamically update their displays, or weather and travel information. If you provide a visual designer with your control, then any page developer using Visual Studio can simply drag your control into a Web Parts zone and configure it at design time without having to write additional code.

**Web Application Development**

Developing fully integrated and personalizable Web applications--such as a portal-- involves the most comprehensive use of Web Parts. You can develop a Web site that allows extensive user personalization of the UI and content--with features similar to [MSN](http://www.msn.com/). Or you can even develop a packaged application that can be shipped and used by companies or fee-based ISPs that provide portal hosting services.

In a Web application scenario, you could offer a complete solution for end users to manage and personalize the application. This could include a set of Web Parts controls that provide the desired features for the site, a consistent set of themes and styles that allow end users to personalize the UI in a consistent way, catalogs of Web Parts controls from which users can select the ones they want to appear on a page, authentication services, and role-based management (for example, allowing administrative users to personalize Web Parts controls and site settings for all users).

For each part of your application, you can extend the Web Parts control set as needed to provide greater control over the environment. For example, besides authoring custom Web Parts controls for the primary UI of your pages, you might also want to develop a custom Web Parts catalog that is consistent with the look and feel of your application, and gives users more flexibility to choose how controls are added to a page. Or you could extend a zone control to provide additional UI options for the Web Parts controls it contains. You could also write a custom personalization provider to give more flexibility and control over how the personalization data is stored and managed.

**Individual ASP.NET Web Server Controls**

This section contains links to topics that provide information about each Web server control. If you want to use HTML server controls in your application, see [HTML Server Controls](http://msdn.microsoft.com/en-us/library/620b4fzf(VS.85).aspx).

|  |
| --- |
| **NoteNote** |
| You can also create output for mobile devices. To do so, you use the same ASP.NET page framework, but you create ASP.NET mobile Web Forms instead of Web Forms pages and use controls specifically designed for mobile devices. For details, see [Creating ASP.NET Mobile Web Pages](http://msdn.microsoft.com/en-us/library/8htez1ds(VS.85).aspx). |

# In This Section

[Standard ASP.NET Web Server Controls](http://msdn.microsoft.com/en-us/library/tt9b3d0f(VS.85).aspx)

[Data Web Server (ASP.NET) Controls](http://msdn.microsoft.com/en-us/library/a63e36w2(VS.85).aspx)

[Data Source Web Server Controls](http://msdn.microsoft.com/en-us/library/ms227437(VS.85).aspx)

[Validation ASP.NET Controls](http://msdn.microsoft.com/en-us/library/debza5t0(VS.85).aspx)

[Navigation ASP.NET Controls](http://msdn.microsoft.com/en-us/library/w25ef46s(VS.85).aspx)

[Login ASP.NET Controls](http://msdn.microsoft.com/en-us/library/d51ttbhx(VS.85).aspx)

**Standard ASP.NET Web Server Controls**

ASP.NET includes a large selection of Web server controls that you can use on ASP.NET Web pages. The topics in this section describe the basic controls used to create forms, such as buttons, text boxes, and more complex controls such as a calendar.

For complete syntax information for Web server controls, see [Web Server Control Syntax](http://msdn.microsoft.com/en-us/library/zfzfkea6(VS.85).aspx).

# In This Section

[AdRotator Web Server Control](http://msdn.microsoft.com/en-us/library/ffxffhs2(VS.85).aspx)

[BulletedList Web Server Control](http://msdn.microsoft.com/en-us/library/5h5ywbxe(VS.85).aspx)

[Button Web Server Controls](http://msdn.microsoft.com/en-us/library/44k478ff(VS.85).aspx)

[Calendar Web Server Control](http://msdn.microsoft.com/en-us/library/tc52hey5(VS.85).aspx)

[CheckBox and CheckBoxList Web Server Controls](http://msdn.microsoft.com/en-us/library/bwxbfx4b(VS.85).aspx)

[DropDownList Web Server Control](http://msdn.microsoft.com/en-us/library/dtx91y0z(VS.85).aspx)

[FileUpload Web Server Control](http://msdn.microsoft.com/en-us/library/fey4ds4e(VS.85).aspx)

[HiddenField Web Server Control](http://msdn.microsoft.com/en-us/library/wwesd2s9(VS.85).aspx)

[HyperLink Web Server Control](http://msdn.microsoft.com/en-us/library/5yf1092x(VS.85).aspx)

[Image Web Server Control](http://msdn.microsoft.com/en-us/library/a8264z60(VS.85).aspx)

[ImageMap Web Server Control](http://msdn.microsoft.com/en-us/library/4cac8b5e(VS.85).aspx)

[Label Web Server Control](http://msdn.microsoft.com/en-us/library/hdd2a3db(VS.85).aspx)

[ListBox Web Server Control](http://msdn.microsoft.com/en-us/library/a8ec8691(VS.85).aspx)

[Literal Web Server Control](http://msdn.microsoft.com/en-us/library/8b98t0zw(VS.85).aspx)

[Localize Web Server Control](http://msdn.microsoft.com/en-us/library/ms227554(VS.85).aspx)

[MultiView and View Web Server Controls](http://msdn.microsoft.com/en-us/library/3ek2z7fh(VS.85).aspx)

[Panel Web Server Control](http://msdn.microsoft.com/en-us/library/4k9563fh(VS.85).aspx)

[PlaceHolder Web Server Control](http://msdn.microsoft.com/en-us/library/3h1c92ts(VS.85).aspx)

[RadioButton and RadioButtonList Web Server Controls](http://msdn.microsoft.com/en-us/library/5xf4ea25(VS.85).aspx)

[Substitution Web Server Control](http://msdn.microsoft.com/en-us/library/hkae7by3(VS.85).aspx)

[Table, TableRow, and TableCell Web Server Controls](http://msdn.microsoft.com/en-us/library/3k184tsx(VS.85).aspx)

[TextBox Web Server Control](http://msdn.microsoft.com/en-us/library/91katfcc(VS.85).aspx)

[Wizard Web Server Control](http://msdn.microsoft.com/en-us/library/s2etd1ek(VS.85).aspx)

[XML Web Server Control](http://msdn.microsoft.com/en-us/library/91z3e26a(VS.85).aspx)

[Securing Standard Controls](http://msdn.microsoft.com/en-us/library/ms178270(VS.85).aspx)

**Data Web Server (ASP.NET) Controls**

This section contains topics that describe the ASP.NET controls you can use to display data on a Web page.

For complete syntax information for Web server controls, see [Web Server Control Syntax](http://msdn.microsoft.com/en-us/library/zfzfkea6(VS.85).aspx).

# In This Section

[ASP.NET Data-Bound Web Server Controls Overview](http://msdn.microsoft.com/en-us/library/ms228214(VS.85).aspx)

[GridView Web Server Control](http://msdn.microsoft.com/en-us/library/8bb3971z(VS.85).aspx)

[DetailsView Web Server Control](http://msdn.microsoft.com/en-us/library/8z5637sx(VS.85).aspx)

[FormView Web Server Control](http://msdn.microsoft.com/en-us/library/fyf1dk77(VS.85).aspx)

[Repeater Web Server Control](http://msdn.microsoft.com/en-us/library/6weyd81h(VS.85).aspx)

[DataList Web Server Control](http://msdn.microsoft.com/en-us/library/9cx2f3ks(VS.85).aspx)

.NET Framework 4 - ASP.NET

**Data Source Controls Overview**

ASP.NET includes data source controls that allow you to work with different types of data sources such as a database, an XML file, or a middle-tier business object. Data source controls connect to and retrieve data from a data source and make it available for other controls to bind to, without requiring code. They can also support modifying data.

This topic provides information about the different types of data source controls in ASP.NET. The data source control model is extensible, so you can also create your own data source controls that interact with different data sources or that provide additional functionality for an existing data source.

 Data Source Control Comparison

The .NET Framework includes data source controls to support different data-binding scenarios. The following table describes the built-in data source controls. More detail about each type of data source control is provided later in this topic.

|  |  |
| --- | --- |
| **Data source control** | **Description** |
| [LinqDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.linqdatasource.aspx) | Enables you to use Language-Integrated Query (LINQ) in an ASP.NET Web page through declarative markup in order to retrieve and modify data from a data object. Supports automatic generation of select, update, insert, and delete commands. The control also supports sorting, filtering, and paging. |
| [EntityDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.entitydatasource.aspx) | Enables you to bind to data that is based on the Entity Data Model (EDM). Supports automatic generation of update, insert, delete, and select commands. The control also supports sorting, filtering and paging. |
| [ObjectDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.objectdatasource.aspx) | Enables you to work with a business object or other class and create Web applications that rely on middle-tier objects to manage data. Supports advanced sorting and paging scenarios unavailable with the other data source controls. |
| [SqlDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.sqldatasource.aspx) | Enables you to work with Microsoft SQL Server, OLE DB, ODBC, or Oracle databases. When used with SQL Server, supports advanced caching capabilities. The control also supports sorting, filtering, and paging when data is returned as a [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx) object. |
| [AccessDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.accessdatasource.aspx) | Enables you to work with a Microsoft Access database. Supports sorting, filtering, and paging when data is returned as a [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx) object. |
| [XmlDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.xmldatasource.aspx) | Enables you to work with an XML file, especially for hierarchical ASP.NET server controls such as the [TreeView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.treeview.aspx) or [Menu](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.menu.aspx) control. Supports filtering capabilities using XPath expressions and enables you to apply an XSLT transformation to the data. The [XmlDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.xmldatasource.aspx) allows you to update data by saving the entire XML document with changes. |
| [SiteMapDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.sitemapdatasource.aspx) | Used with ASP.NET site navigation. For more information, see [ASP.NET Site Navigation](http://msdn.microsoft.com/en-us/library/e468hxky.aspx). |

 LinqDataSource Control

The [LinqDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.linqdatasource.aspx) control enables you to use LINQ in an ASP.NET page to retrieve data from a database table or an in-memory data collection. You can use declarative markup to write all the conditions that are required in order retrieve, filter, order, and group the data. When you retrieve data from a SQL database table, you can also configure a [LinqDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.linqdatasource.aspx) control to handle update, insert, and delete operations. You can do this writing SQL commands to perform these tasks. By using the [LinqDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.linqdatasource.aspx) control, you can reduce the amount of code that is required for data operations, compared to performing the same operations in other data source controls.

For more information, see [LinqDataSource Web Server Control Overview](http://msdn.microsoft.com/en-us/library/bb547113.aspx).

 SqlDataSource Control

The [SqlDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.sqldatasource.aspx) control retrieves and modifies data using SQL commands. The [SqlDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.sqldatasource.aspx) control works with Microsoft SQL Server, OLE DB, ODBC, and Oracle databases.

The [SqlDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.sqldatasource.aspx) control can return results as a DataReader or a DataSet object. It supports sorting, filtering, and caching when the results are returned as a [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx). When you are working with Microsoft SQL Server, the control has the added benefit that cache results can be invalidated when the database changes, using a [SqlCacheDependency](http://msdn.microsoft.com/en-us/library/system.web.caching.sqlcachedependency.aspx) object.

For more information, see [SqlDataSource Web Server Control Overview](http://msdn.microsoft.com/en-us/library/dz12d98w.aspx).

 EntityDataSource Control

The [EntityDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.entitydatasource.aspx) control supports data binding scenarios based on the Entity Data Model (EDM). This data specification represents data as sets of entities and relationships. The Entity Framework uses the EDM in object-relational mapping and in other scenarios such as WCF Data Services. The [EntityDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.entitydatasource.aspx) control supports Entity-SQL (eSQL) as the query language, and it supports the query specification that is exposed by the [ObjectQuery<(Of <(T>)>)](http://msdn.microsoft.com/en-us/library/bb345303.aspx) class.

For more information, see [EntityDataSource Web Server Control Overview](http://msdn.microsoft.com/en-us/library/cc488502.aspx).

 ObjectDataSource Control

The [ObjectDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.objectdatasource.aspx) control works with a business object or other class in Web applications that rely on middle-tier business objects to manage data. The control is designed to interact with an object that implements one or more methods to retrieve or modify data. When data-bound controls interact with the [ObjectDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.objectdatasource.aspx) control to retrieve or modify data, the [ObjectDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.objectdatasource.aspx) control passes values from the bound control to the source object as parameters in method calls.

The source object's data-retrieval methods must return a [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx), [DataTable](http://msdn.microsoft.com/en-us/library/system.data.datatable.aspx), or [DataView](http://msdn.microsoft.com/en-us/library/system.data.dataview.aspx) object, or an object that implements the [IEnumerable](http://msdn.microsoft.com/en-us/library/system.collections.ienumerable.aspx) interface. If the data is returned as a [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx), [DataTable](http://msdn.microsoft.com/en-us/library/system.data.datatable.aspx), or [DataView](http://msdn.microsoft.com/en-us/library/system.data.dataview.aspx) object, the [ObjectDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.objectdatasource.aspx) control can cache and filter the data. You can also implement advanced paging scenarios if the source object accepts page size and record index information from the [ObjectDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.objectdatasource.aspx) control.

For more information, see [ObjectDataSource Web Server Control Overview](http://msdn.microsoft.com/en-us/library/9a4kyhcx.aspx).

 XmlDataSource Control

The [XmlDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.xmldatasource.aspx) control reads and writes XML data so that you can work with it using controls such as the [TreeView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.treeview.aspx) and [Menu](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.menu.aspx) controls. The [XmlDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.xmldatasource.aspx) control can read either an XML file or string of XML. If the control is working with an XML file, it can write modified XML back to the source file. If a schema is available that describes the data, the [XmlDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.xmldatasource.aspx) control can use the schema to expose data using typed members.

You can apply an XSLT transformation to the XML data, which allows you to restructure the raw data from the XML file into a format better suited to the control you want to bind to the XML data.

You can also apply XPath expressions to the XML data, which allows you to filter the XML data to return only certain nodes in the XML tree, to look for nodes that have specific values in them, and so on. Using an XPath expression disables the ability to insert new data.

For more information, see [XmlDataSource Web Server Control Overview](http://msdn.microsoft.com/en-us/library/494y92bs.aspx).

 AccessDataSource Control

The [AccessDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.accessdatasource.aspx) control is a specialized version of the [SqlDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.sqldatasource.aspx) control, designed to work specifically with Microsoft Access .mdb files. As with the [SqlDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.sqldatasource.aspx) control, you use SQL statements to define how the control fetches and retrieves data.

For more information, see [AccessDataSource Web Server Control Overview](http://msdn.microsoft.com/en-us/library/b277ts6z.aspx).

 SiteMapDataSource Control

The [SiteMapDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.sitemapdatasource.aspx) control works with ASP.NET site maps and provides site navigation data. It is most commonly used with the [Menu](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.menu.aspx) control. The [SiteMapDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.sitemapdatasource.aspx) control is also useful when you want to customize site navigation using site map data with Web server controls that are not specifically designed for navigation, such as the [TreeView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.treeview.aspx) or [DropDownList](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.dropdownlist.aspx) controls.

.NET Framework 4 - ASP.NET

**Types of Validation for ASP.NET Server Controls**

The following table lists ASP.NET validation controls and how you can use them.

|  |  |  |
| --- | --- | --- |
| **Security noteSecurity Note** | | |
| By default, ASP.NET Web pages automatically validate that malicious users are not attempting to send script to your application. For more information, see [Script Exploits Overview](http://msdn.microsoft.com/en-us/library/w1sw53ds.aspx). | | |
| **Type of validation** | **Control to use** | **Description** |
| Required entry | [RequiredFieldValidator](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.requiredfieldvalidator.aspx) | Ensures that the user does not skip an entry. For details, see [How to: Validate Required Entries for ASP.NET Server Controls](http://msdn.microsoft.com/en-us/library/e78xxk8k.aspx). |
| Comparison to a value | [CompareValidator](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.comparevalidator.aspx) | Compares a user's entry against a constant value, against the value of another control (using a comparison operator such as less than, equal, or greater than), or for a specific data type. For details, see [How to: Validate Against a Specific Value for ASP.NET Server Controls](http://msdn.microsoft.com/en-us/library/e303tf05.aspx) and [How to: Validate Against a Data Type for ASP.NET Server Controls](http://msdn.microsoft.com/en-us/library/ad548tzy.aspx). |
| Range checking | [RangeValidator](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.rangevalidator.aspx) | Checks that a user's entry is between specified lower and upper boundaries. You can check ranges within pairs of numbers, alphabetic characters, and dates. For details, see [How to: Validate Against a Range of Values for ASP.NET Server Controls](http://msdn.microsoft.com/en-us/library/bk58fdzx.aspx). |
| Pattern matching | [RegularExpressionValidator](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.regularexpressionvalidator.aspx) | Checks that the entry matches a pattern defined by a regular expression. This type of validation enables you to check for predictable sequences of characters, such as those in e-mail addresses, telephone numbers, postal codes, and so on. For details, see [How to: Validate Against Patterns for ASP.NET Server Controls](http://msdn.microsoft.com/en-us/library/6xh899wy.aspx). |
| User-defined | [CustomValidator](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.customvalidator.aspx) | Checks the user's entry using validation logic that you write yourself. This type of validation enables you to check for values derived at run time. For details, see [How to: Validate with a Custom Function for ASP.NET Server Controls](http://msdn.microsoft.com/en-us/library/f5db6z8k.aspx) and [How to: Validate Against Values in a Database for ASP.NET Server Controls](http://msdn.microsoft.com/en-us/library/s5z00s5e.aspx). |

You can attach more than one validation control to an input control. For example, you might specify that a control is required and that it also contains a specific range of values.

A related control, the [ValidationSummary](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.validationsummary.aspx) control, does not perform validation, but is often used in conjunction with other validation controls to display the error messages from all the validation controls on the page together. For more information, see [How to: Control Validation Error Message Display for ASP.NET Server Controls](http://msdn.microsoft.com/en-us/library/1ze30x3t.aspx).

**Navigation ASP.NET Controls**

The topics in this section provide information on ASP.NET Web server controls that you can use to create menus and other navigation aids on ASP.NET Web pages.

For complete syntax information for Web server controls, see [Web Server Control Syntax](http://msdn.microsoft.com/en-us/library/zfzfkea6(VS.85).aspx).

# In This Section

[Menu Web Server Control](http://msdn.microsoft.com/en-us/library/bz09dy46(VS.85).aspx)

[SiteMapPath Web Server Control Overview](http://msdn.microsoft.com/en-us/library/x20z8c51(VS.85).aspx)

[TreeView Web Server Control](http://msdn.microsoft.com/en-us/library/3eafky27(VS.85).aspx)

.NET Framework 4 - ASP.NET

**ASP.NET Login Controls Overview**

ASP.NET provides robust login (authentication) functionality for ASP.NET Web applications without requiring programming. The default Visual Studio project templates for Web applications and for Web sites include prebuilt pages that let users register a new account, log in, and change their passwords For information about how to use the built-in login page templates, see [Walkthrough: Creating an ASP.NET Web Site with Basic User Login](http://msdn.microsoft.com/en-us/library/ff184050.aspx).

You can also create your own pages that you can add ASP.NET login controls to in order to add login functionality. To use the login controls, you create a Web pages and then add the login controls to them from the **Toolbox**.

Typically, you restrict access to ASP.NET pages by putting them into a protected folder. You then configure the folder to deny access to anonymous users (users who are not logged in) and to grant access to authenticated (logged-in) users. (The default project template for Web projects includes a folder named Accounts that is already configured to allow access only to logged-in users.) Optionally, you can assign users to roles and then authorize users to access specific Web pages by role.

By default, login controls integrate with ASP.NET membership and ASP.NET forms authentication to help automate user authentication for a Web site. For information about how to use ASP.NET membership with forms authentication, see [Introduction to Membership](http://msdn.microsoft.com/en-us/library/yh26yfzy.aspx).

By default, the ASP.NET login controls work in plain text over HTTP. If you are concerned about security, use HTTPS with SSL encryption. For more information about SSL, see [Configuring SSL on a Web Server or a Web Site](http://go.microsoft.com/fwlink/?LinkId=38553) in the IIS documentation.

|  |
| --- |
| **NoteNote** |
| Login controls might not function correctly if the [Method](http://msdn.microsoft.com/en-us/library/system.web.ui.htmlcontrols.htmlform.method.aspx) of the ASP.NET Web page is changed from POST (the default) to GET. |

This topic describes each ASP.NET login control and provides links to the control's reference documentation.

This topic contains the following sections:

* [Built-in Login Pages](http://msdn.microsoft.com/en-us/library/ms178329.aspx#builtin_login_pages)
* [The Login Control](http://msdn.microsoft.com/en-us/library/ms178329.aspx#the_login_control)
* [The LoginView Control](http://msdn.microsoft.com/en-us/library/ms178329.aspx#the_loginview_control)
* [The LoginStatus Control](http://msdn.microsoft.com/en-us/library/ms178329.aspx#the_loginstatus_control)
* [The LoginName Control](http://msdn.microsoft.com/en-us/library/ms178329.aspx#the_loginname_control)
* [The PasswordRecovery Control](http://msdn.microsoft.com/en-us/library/ms178329.aspx#the_passwordrecovery_control)
* [The CreateUserWizard Control](http://msdn.microsoft.com/en-us/library/ms178329.aspx#the_createuserwizard_control)
* [The ChangePassword Control](http://msdn.microsoft.com/en-us/library/ms178329.aspx#the_changepassword_control)

 Built-in Login Pages

When you use the default Visual Studio template to create a Web site or Web application, pages that support login functionality are created in the Account folder. By default, pages in the Account folder are not accessible to anonymous users, except the registration page (Register.aspx) and the login page (Login.aspx). The settings that define access to pages in the Account folder are configured in the Web.config file in that folder. The settings that define access the Login page are configured in the root Web.config file.

The following illustration shows the contents of the Account folder.

**Account Folder**

The following table lists the contents of the Account folder:

|  |  |
| --- | --- |
| **Page** | **Description** |
| Login.aspx | Enables users who have an account to log in by providing a user name and password. This page includes a link to the registration page. The Login.aspx page is accessible to anonymous users.  Contains the [Login](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.login.aspx) control.  The following illustration shows the login page in the browser.  Login Page Template |
| Register.aspx | Enables users to register and create a new account. This page is accessible to anonymous users and to authenticated users.  Contains the [CreateUserWizard](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.createuserwizard.aspx) control.  The following illustration shows the create new account (registration) page in the browser.  Create New Account Page Template |
| ChangePassword.aspx | Enables logged-in users to change their password. This page is accessible only to authenticated users.  Contains the [ChangePassword](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.changepassword.aspx) control.  The following illustration shows the change password page in the browser.  Change Password Page Template |
| ChangePasswordSuccess.aspx | Displays a message that indicates that the password was changed successfully. This page is accessible to authenticated users only.  This page does not use any ASP.NET login controls. |
| Web.config | Contains the settings that define access to pages in the Account folder |

**Login Functionality in the Master Page**

When you use the default Visual Studio template to create a Web site or Web application, the default master page (Site.master) contains the following controls that provide login functionality.

* [LoginView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.loginview.aspx)
* [LoginStatus](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.loginstatus.aspx)

The Site.master page also includes a hyperlink to the login page, which is accessible to all users.

**Storing the Login Information**

Login (membership) information is stored in a database. By default, this is a local database in the ASPNETDB.mdf file in the App\_Data folder of the Web application. The database can be created in the following ways:

* You can use the Web Site Administration Tool to configure membership and roles manually, which automatically creates the database. For more information, see [Walkthrough: Creating a Web Site with Membership and User Login](http://msdn.microsoft.com/en-us/library/879kf95c.aspx).
* When the first user registers and creates an account, and if the database does not already exist, ASP.NET creates the database automatically. This option does not provide a way to specify roles for a user.

**Modifying the Membership Attributes**

The settings for ASP.NET membership are in the **membership** section of the root Web.config file. In the **providers** section you can change attributes such as the number of invalid login attempts to allow, password length, and so on. For more information, see [Introduction to Membership](http://msdn.microsoft.com/en-us/library/yh26yfzy.aspx).

 The Login Control

The [Login](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.login.aspx) control displays a user interface for user authentication. The [Login](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.login.aspx) control contains text boxes for the user name and password and a check box that allows users to indicate whether they want the server to store their identity using ASP.NET membership and automatically be authenticated the next time they visit the site.

The [Login](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.login.aspx) control has properties for customized display, for customized messages, and for links to other pages where users can change their password or recover a forgotten password. The [Login](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.login.aspx) control can be used as a standalone control on a main or home page, or you can use it on a dedicated login page.

If you use the [Login](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.login.aspx) control with ASP.NET membership, you do not need to write code to perform authentication. However, if you want to create your own authentication logic, you can handle the [Login](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.login.aspx) control's [Authenticate](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.login.authenticate.aspx) event and add custom authentication code.

 The LoginView Control

The [LoginView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.loginview.aspx) control allows you to display different information to anonymous and logged-in users. The control displays one of two templates: the [AnonymousTemplate](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.loginview.anonymoustemplate.aspx) or the [LoggedInTemplate](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.loginview.loggedintemplate.aspx). In the templates, you can add markup and controls that display information appropriate for anonymous users and authenticated users, respectively.

The [LoginView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.loginview.aspx) control also includes events for [ViewChanging](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.loginview.viewchanging.aspx) and [ViewChanged](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.loginview.viewchanged.aspx), which allow you to write handlers for when the user logs in and changes status.

 The LoginStatus Control

The [LoginStatus](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.loginstatus.aspx) control displays a login link for users who are not authenticated and a logout link for users who are authenticated. The login link takes the user to a login page. The logout link resets the current user's identity to be an anonymous user.

You can customize the appearance of the [LoginStatus](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.loginstatus.aspx) control by setting the [LoginText](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.loginstatus.logintext.aspx) and [LoginImageUrl](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.loginstatus.loginimageurl.aspx) properties.

 The LoginName Control

The [LoginName](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.loginname.aspx) control displays a user's login name if the user has logged in using ASP.NET membership. Alternatively, if your site uses integrated Windows authentication, the control displays the user's Windows account name.

 The PasswordRecovery Control

The [PasswordRecovery](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.passwordrecovery.aspx) control allows user passwords to be retrieved based on the e-mail address that was used when the account was created. The [PasswordRecovery](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.passwordrecovery.aspx) control sends an e-mail message containing a password to the user.

You can configure ASP.NET membership to store passwords using non-reversible encryption. In that case, the [PasswordRecovery](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.passwordrecovery.aspx) control generates a new password instead of sending the original password to the user.

You can also configure membership to include a security question that the user must answer to recover a password. If you do, the [PasswordRecovery](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.passwordrecovery.aspx) control asks the question and checks the answer before recovering the password.

The [PasswordRecovery](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.passwordrecovery.aspx) control requires that your application can forward e-mail message to a Simple Mail Transfer Protocol (SMTP) server. You can customize the text and format of the e-mail message sent to the user by setting the [MailDefinition](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.passwordrecovery.maildefinition.aspx) property.

|  |
| --- |
| **NoteNote** |
| Password information sent in an e-mail message is sent as clear text. |

The following example shows a [PasswordRecovery](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.passwordrecovery.aspx) control declared in an ASP.NET page with [MailDefinition](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.passwordrecovery.maildefinition.aspx) property settings to customize the e-mail message.

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl77_ctl00_ctl09_code');" \o "Copy Code)

<asp:PasswordRecovery ID="PasswordRecovery1" Runat="server"

SubmitButtonText="Get Password" SubmitButtonType="Link">

<MailDefinition From="administrator@Contoso.com"

Subject="Your new password"

BodyFileName="PasswordMail.txt" />

</asp:PasswordRecovery>

 The CreateUserWizard Control

The [CreateUserWizard](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.createuserwizard.aspx) control collects information from potential users. By default, the [CreateUserWizard](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.createuserwizard.aspx) control adds the new user to the ASP.NET membership system.

The [CreateUserWizard](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.createuserwizard.aspx) control gathers the following user information:

* User name
* Password
* Confirmation of password
* E-mail address
* Security question
* Security answer

This information is used to authenticate users and recover user passwords, if necessary.

|  |
| --- |
| **NoteNote** |
| The [CreateUserWizard](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.createuserwizard.aspx) control is inherited from the [Wizard](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.wizard.aspx) control. |

The following example shows a typical ASP.NET declaration for the [CreateUserWizard](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.createuserwizard.aspx) control:

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl78_ctl00_ctl07_code');" \o "Copy Code)

<asp:CreateUserWizard ID="CreateUserWizard1" Runat="server"

ContinueDestinationPageUrl="~/Default.aspx">

<WizardSteps>

<asp:CreateUserWizardStep Runat="server"

Title="Sign Up for Your New Account">

</asp:CreateUserWizardStep>

<asp:CompleteWizardStep Runat="server"

Title="Complete">

</asp:CompleteWizardStep>

</WizardSteps>

</asp:CreateUserWizard>

 The ChangePassword Control

The [ChangePassword](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.changepassword.aspx) control allows users to change their password. The user must first supply the original password and then create and confirm the new password. If the original password is correct, the user password is changed to the new password. The control also includes support for sending an e-mail message about the new password.

The [ChangePassword](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.changepassword.aspx) control includes two templated views that are displayed to the user. The first is the [ChangePasswordTemplate](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.changepassword.changepasswordtemplate.aspx), which displays the user interface used to gather the data required to change the user password. The second template is the [SuccessTemplate](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.changepassword.successtemplate.aspx), which defines the user interface that is displayed after a user password has been successfully changed.

The [ChangePassword](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.changepassword.aspx) control works with authenticated and non-authenticated users. If a user has not been authenticated, the control prompts the user for a login name. If the user is authenticated, the control populates the text box with the user's login name.