**Cross-Page Posting in ASP.NET Web Pages**

**.NET Framework 4**

[Other Versions](javascript:;)

http://i.msdn.microsoft.com/Areas/Epx/Content/Images/ImageSprite.png

* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/ms178139(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/ms178139(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/ms178139(d=printer,v=vs.80).aspx)

By default, buttons and other controls that cause a postback on an ASP.NET Web page submit the page back to itself. This is part of the round-trip cycle that ASP.NET Web pages go through as part of their normal processing. For details, see [Introduction to Programming ASP.NET Web Pages](http://msdn.microsoft.com/en-us/library/ms178125.aspx).

Under some circumstances, you might want to post one page to another page. For example, you might be creating a multi-page form that collects different information on each page. In that case, you can configure certain controls (those that implement the [IButtonControl](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.ibuttoncontrol.aspx) interface, such as the [Button](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.button.aspx) control) on the page to post to a different target page. This is referred to as cross-page posting. Cross-page posting provides some advantages over using the [Transfer](http://msdn.microsoft.com/en-us/library/system.web.httpserverutility.transfer.aspx) method to redirect to another page. For details, see [Redirecting Users to Another Page](http://msdn.microsoft.com/en-us/library/x3x8t37x.aspx).

|  |
| --- |
| **NoteNote** |
| You can also use the [Wizard](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.wizard.aspx) control to create multi-view forms. For details, see [Wizard Web Server Control Overview](http://msdn.microsoft.com/en-us/library/fs0za4w6.aspx). |

Because cross-page posting is configured for individual controls, you can create a page that posts to different pages depending on which button the user clicks.

[Getting Information from the Source Page](javascript:void(0))

When you configure a page for cross-page posting, you frequently want to get information from the source page. This might include the information from controls on the page—that is, the information being posted by the browser—as well as public properties of the source page.

**Getting Control Values**

The [Page](http://msdn.microsoft.com/en-us/library/system.web.ui.page.aspx) class exposes a property named [PreviousPage](http://msdn.microsoft.com/en-us/library/system.web.ui.page.previouspage.aspx). If the source page and target page are in the same ASP.NET application, the [PreviousPage](http://msdn.microsoft.com/en-us/library/system.web.ui.page.previouspage.aspx) property in the target page contains a reference to the source page. (If the page is not the target of a cross-page posting, or if the pages are in different applications, the [PreviousPage](http://msdn.microsoft.com/en-us/library/system.web.ui.page.previouspage.aspx) property is not initialized.) By default, the [PreviousPage](http://msdn.microsoft.com/en-us/library/system.web.ui.page.previouspage.aspx) property is typed as [Page](http://msdn.microsoft.com/en-us/library/system.web.ui.page.aspx).

|  |
| --- |
| **NoteNote** |
| If the source and target page are in different applications, you cannot directly get the values of controls on the page, but you can read the posted data from the [Form](http://msdn.microsoft.com/en-us/library/system.web.httprequest.form.aspx) dictionary. You cannot read view state from the source page, because it is hashed. If you want to store values in the source page and make them available in a target page in another application, you can store the values as strings inside hidden fields on the source page and access them through Request.Form on the target page. |

Using the reference in the [PreviousPage](http://msdn.microsoft.com/en-us/library/system.web.ui.page.previouspage.aspx) property, you can search for controls on the source page and extract their value. You typically do this with the [FindControl](http://msdn.microsoft.com/en-us/library/system.web.ui.page.findcontrol.aspx) method.

|  |
| --- |
| **NoteNote** |
| If you are coding the source page specifically to be able to share information with target pages, an easier way to make control values available to the target page is to expose them as public properties. For details, see [Getting Public Property Values from the Source Page](http://msdn.microsoft.com/en-us/library/ms178139(d=printer).aspx#GettingPublicPropertyValuesFromSourcePage) later in this topic. |

The following code example shows how you can get the value of the TextBox1 control on the source page.

C#

[VB](http://msdn.microsoft.com/en-us/library/ms178139(d=printer).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-1)

if (Page.PreviousPage != null)

{

TextBox SourceTextBox =

(TextBox)Page.PreviousPage.FindControl("TextBox1");

if (SourceTextBox != null)

{

Label1.Text = SourceTextBox.Text;

}

}

The [FindControl](http://msdn.microsoft.com/en-us/library/system.web.ui.page.findcontrol.aspx) method finds controls in the current naming container. If the control you are looking for is inside another control (typically, inside a template), you must first get a reference to the container and then search the container to find the control you want to get. In the following code example, the source page contains a [Login](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.login.aspx) control with a [LayoutTemplate](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.login.layouttemplate.aspx) container that in turn contains a [TextBox](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.textbox.aspx) control named UserName. The code gets the value of the UserName control.

C#

[VB](http://msdn.microsoft.com/en-us/library/ms178139(d=printer).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-2)

Login LoginControl = (Login)PreviousPage.FindControl("Login1");

if (LoginControl != null)

{

TextBox UserName = (TextBox)LoginControl.FindControl("UserName");

if (UserName != null)

{

Label1.Text = UserName.Text;

}

}

else

{

Label1.Text = "Cannot find user name in Login control.";

}

**Getting Public Property Values from the Source Page**

In the target page of a cross-page posting, you can also get the values of public members of the source page. The most common scenario is that the source page defines public properties and you want to get their values on the target page.

|  |
| --- |
| **Security noteSecurity Note** |
| It is recommended that you expose only the information you need as public properties to reduce the amount of information available to potentially malicious users. |

To get public members of the source page, you must first get a strongly typed reference to the source page.

You can do so in a number of ways. The first is to include an [@ PreviousPageType](http://msdn.microsoft.com/en-us/library/ms228169.aspx) directive in the target page, which allows you to specify the source page, as in this example:

<%@ PreviousPageType VirtualPath="~/SourcePage.aspx" %>

When this directive is included, the [PreviousPage](http://msdn.microsoft.com/en-us/library/system.web.ui.page.previouspage.aspx) property is strongly typed to the class of the referenced source page. As a consequence, you can directly reference public members of the source page. You can specify the type of the source page either directly, using a type attribute, or indirectly by explicitly referencing the source page in a VirtualPath attribute, as shown in the example.

The following code example shows a portion of a source page containing a public property named CurrentCity that exposes the value of a [TextBox](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.textbox.aspx) control named textCity.

C#

[VB](http://msdn.microsoft.com/en-us/library/ms178139(d=printer).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-4)

public String CurrentCity

{

get

{

return textCity.Text;

}

}

|  |
| --- |
| **NoteNote** |
| Properties on the source page that are created primarily to expose values for cross-page posting are usually read-only properties. Although the source page can contain public read/write properties, setting a source page property from the target page property generally has no purpose, because the value will not be persisted. |

If the target page contains a PreviousPageType directive that points to the source page, you can access the source page's CurrentCity property using code such as the following.

C#

[VB](http://msdn.microsoft.com/en-us/library/ms178139(d=printer).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-5)

Label1.Text = PreviousPage.CurrentCity;

Another way to get a strongly typed reference to the source page is to include an [@ Reference](http://msdn.microsoft.com/en-us/library/w70c655a.aspx) directive in the target page that references the source page, as you would reference any type that you wanted to use in your page. In that case, in the target page you can get the target page's [PreviousPage](http://msdn.microsoft.com/en-us/library/system.web.ui.page.previouspage.aspx) property and cast it to the source page type, as in the following code example.

C#

[VB](http://msdn.microsoft.com/en-us/library/ms178139(d=printer).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-6)

SourcePage\_aspx sourcePage;

sourcePage = (SourcePage\_aspx) PreviousPage;

Label1.Text = sourcePage.CurrentCity;

**Checking for Postbacks in the Target Page**

During a cross-page postback, the contents of the source page's controls are posted to the target page, and the browser executes an HTTP POST operation (not a GET operation). However, in the target page, the [IsPostBack](http://msdn.microsoft.com/en-us/library/system.web.ui.page.ispostback.aspx) property is false immediately after a cross-page post. Although the behavior is that of a POST, the cross-posting is not a postback to the target page. Therefore, [IsPostBack](http://msdn.microsoft.com/en-us/library/system.web.ui.page.ispostback.aspx) is set to false and the target page can go through its first-time code.

If it is useful in your application, you can determine whether the target page is running as the result of a cross-page post. To do so, you can test the [IsCrossPagePostBack](http://msdn.microsoft.com/en-us/library/system.web.ui.page.iscrosspagepostback.aspx) property of the page reference returned by the target page's [PreviousPage](http://msdn.microsoft.com/en-us/library/system.web.ui.page.previouspage.aspx) property, as in the following code example.

C#

[VB](http://msdn.microsoft.com/en-us/library/ms178139(d=printer).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-7)

if(PreviousPage != null)

{

if(PreviousPage.IsCrossPagePostBack == true)

{

Label1.Text = "Cross-page post.";

}

}

else

{

Label1.Text = "Not a cross-page post.";

}

Note that if the current page is not the target of a cross-page post, the [PreviousPage](http://msdn.microsoft.com/en-us/library/system.web.ui.page.previouspage.aspx) property returns null (Nothing in Visual Basic).

For more information, see [How to: Determine How ASP.NET Web Pages Were Invoked](http://msdn.microsoft.com/en-us/library/ms178141.aspx).

**Cross-Page Posting versus Server.Transfer**

The [PreviousPage](http://msdn.microsoft.com/en-us/library/system.web.ui.page.previouspage.aspx) property and the PreviousPageType directive are useful in two situations where you invoke the target page: in a cross-page postback, which is a client-based transfer, and with the [Transfer](http://msdn.microsoft.com/en-us/library/system.web.httpserverutility.transfer.aspx) method, which is a server-based operation. In both operations, code in the target page can get a reference to the source page using the [PreviousPage](http://msdn.microsoft.com/en-us/library/system.web.ui.page.previouspage.aspx) property.

It might be important in the target page to determine whether the page was invoked from a cross-page posting or a Server.Transfer operation. To help you do this, the [Page](http://msdn.microsoft.com/en-us/library/system.web.ui.page.aspx) class exposes a property named [IsCrossPagePostBack](http://msdn.microsoft.com/en-us/library/system.web.ui.page.iscrosspagepostback.aspx). For details, see [How to: Determine How ASP.NET Web Pages Were Invoked](http://msdn.microsoft.com/en-us/library/ms178141.aspx). For details about Server.Transfer, see [Redirecting Users to Another Page](http://msdn.microsoft.com/en-us/library/x3x8t37x.aspx).

**How to: Determine How ASP.NET Web Pages Were Invoked**

**.NET Framework 4**

[Other Versions](javascript:;)



* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/ms178141(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/ms178141(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/ms178141(d=printer,v=vs.80).aspx)

It is often useful to know how an ASP.NET Web page was invoked: whether by an original request (an HTTP GET), a postback (an HTTP POST), a cross-page post from another page (an HTTP POST), or a transfer from another page using the [Transfer](http://msdn.microsoft.com/en-us/library/system.web.httpserverutility.transfer.aspx) method or using a callback from the browser. The [Page](http://msdn.microsoft.com/en-us/library/system.web.ui.page.aspx) class exposes a set of properties that you can use to determine how a page was invoked.

**To determine how an ASP.NET Web page was invoked**

* Examine the values of the following [Page](http://msdn.microsoft.com/en-us/library/system.web.ui.page.aspx) class properties, and then refer the table to determine how the page was invoked:
  + [IsPostBack](http://msdn.microsoft.com/en-us/library/system.web.ui.page.ispostback.aspx)
  + [PreviousPage](http://msdn.microsoft.com/en-us/library/system.web.ui.page.previouspage.aspx)
  + [IsCrossPagePostBack](http://msdn.microsoft.com/en-us/library/system.web.ui.page.iscrosspagepostback.aspx)
  + [IsCallback](http://msdn.microsoft.com/en-us/library/system.web.ui.page.iscallback.aspx)

The following table lists ways in which a page can be invoked and the corresponding [Page](http://msdn.microsoft.com/en-us/library/system.web.ui.page.aspx) property values.

|  |  |
| --- | --- |
| **Invocation method** | **Property values** |
| Original request | * + [IsPostBack](http://msdn.microsoft.com/en-us/library/system.web.ui.page.ispostback.aspx) is set to false.   + [PreviousPage](http://msdn.microsoft.com/en-us/library/system.web.ui.page.previouspage.aspx) is set to null (Nothing in Visual Basic).   + [IsCallback](http://msdn.microsoft.com/en-us/library/system.web.ui.page.iscallback.aspx) is set to false. |
| Postback | * + [IsPostBack](http://msdn.microsoft.com/en-us/library/system.web.ui.page.ispostback.aspx) is set to true.   + [PreviousPage](http://msdn.microsoft.com/en-us/library/system.web.ui.page.previouspage.aspx) is set to null (Nothing in Visual Basic).   + [IsCallback](http://msdn.microsoft.com/en-us/library/system.web.ui.page.iscallback.aspx) is set to false. |
| Cross-page posting | * + [IsPostBack](http://msdn.microsoft.com/en-us/library/system.web.ui.page.ispostback.aspx) is set to false.   + [PreviousPage](http://msdn.microsoft.com/en-us/library/system.web.ui.page.previouspage.aspx) references the source page.   + [IsCrossPagePostBack](http://msdn.microsoft.com/en-us/library/system.web.ui.page.iscrosspagepostback.aspx) is set to true.   + [IsCallback](http://msdn.microsoft.com/en-us/library/system.web.ui.page.iscallback.aspx) is set to false. |
| Server transfer | * + [IsPostBack](http://msdn.microsoft.com/en-us/library/system.web.ui.page.ispostback.aspx) is set to false.   + [PreviousPage](http://msdn.microsoft.com/en-us/library/system.web.ui.page.previouspage.aspx) references the source page.   + [IsCrossPagePostBack](http://msdn.microsoft.com/en-us/library/system.web.ui.page.iscrosspagepostback.aspx) that is referenced in the [PreviousPage](http://msdn.microsoft.com/en-us/library/system.web.ui.page.previouspage.aspx) is set to false.   + [IsCallback](http://msdn.microsoft.com/en-us/library/system.web.ui.page.iscallback.aspx) is set to false. |
| Callback | * + [IsPostBack](http://msdn.microsoft.com/en-us/library/system.web.ui.page.ispostback.aspx) is set to false.   + [PreviousPage](http://msdn.microsoft.com/en-us/library/system.web.ui.page.previouspage.aspx) is set to null (Nothing in Visual Basic).   + [IsCallback](http://msdn.microsoft.com/en-us/library/system.web.ui.page.iscallback.aspx) is set to true. |

|  |
| --- |
| **NoteNote** |
| Be sure you test the [IsCrossPagePostBack](http://msdn.microsoft.com/en-us/library/system.web.ui.page.iscrosspagepostback.aspx) property of the page that is referenced in [PreviousPage](http://msdn.microsoft.com/en-us/library/system.web.ui.page.previouspage.aspx). The [IsCrossPagePostBack](http://msdn.microsoft.com/en-us/library/system.web.ui.page.iscrosspagepostback.aspx) property of the current page always returns false. |

**Redirecting Users to Another Page**

**.NET Framework 4**

[Other Versions](javascript:;)

http://i.msdn.microsoft.com/Areas/Epx/Content/Images/ImageSprite.png

* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/x3x8t37x(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/x3x8t37x(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/x3x8t37x(d=printer,v=vs.80).aspx)

You will often want to redirect users to other pages as part of your Web application. ASP.NET provides the following ways for you to build redirection into your Web pages:

* Using hyperlinks on pages.
* Configuring cross-page posting, which enables you to specify an alternate target page when the current page is submitted.
* Redirecting programmatically by forcing the browser to request a different page.
* Redirecting programmatically by transferring control to a different page in the same Web application.

Each of these options is described below. A table at the end of the topic summarizes the options and provides guidelines to help you decide when to use each option.

[Hyperlinks](javascript:void(0))

You can use an HTML anchor tag (<a>) on an ASP.NET Web page to create static links, or you can programmatically control the link text and target URL of hyperlinks using the [HyperLink](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.hyperlink.aspx) control. In this scenario, the user explicitly clicks a link and the browser transfers to the target page. The target page is invoked using an HTTP GET command. Therefore, no information about the source page is passed to the target page unless you specify a query string on the URL of the target page. If the source and target page are in the same Web application, they can share information using session state or application state.

[Cross-Page Posting](javascript:void(0))

By default, buttons in an ASP.NET Web page post the page to itself. Cross-page posting enables you to configure a button on an ASP.NET Web page to post the current page to a different page. A typical example is when creating a multi-page form. You can configure buttons on the page to move to the next and previous pages of the form.

Cross-page posting is similar to hyperlinks in that the transfer is initiated by a user action. However, in cross-page posting, the target page is invoked using an HTTP POST command, which sends the values of controls on the source page to the target page. In addition, if the source and target page are in the same Web application, the target page can access public properties of the source page. As always, all of the pages in the application can share information stored in session state or application state.

For more information, see [Cross-Page Posting in ASP.NET Web Pages](http://msdn.microsoft.com/en-us/library/ms178139.aspx) and [How to: Post ASP.NET Web Pages to a Different Page](http://msdn.microsoft.com/en-us/library/ms178140.aspx).

[Redirecting Programmatically Using the Browser](javascript:void(0))

You can redirect users to another page using the capabilities of the user's browser. In a browser redirect, the browser issues a new request to the target server in the form of an HTTP GET request.

You can trigger the redirect programmatically in client script or server code. In client script, you can call the form.submit method, provided the <form> element's method attribute value is get. In that case, if the current page contains form data, it is passed to the target server by appending it as a query string onto the requested URL.

In server code, you can programmatically redirect by calling the [Redirect](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.redirect.aspx) method. The method sends a command to the user's browser that causes the browser to issue an HTTP GET command for the target page. Calling the server Redirect method is the programmatic equivalent of clicking a hyperlink, in that it results in a new request for the target page. Because you are calling the methods from your own code, you can dynamically define the target URL, including any query-string information. If the source and target pages are in the same Web application, you can share data between the source and target pages by adding server code to store the data in session state.

|  |
| --- |
| **NoteNote** |
| Internet Explorer through version 6.0 can process only up to 2,048 characters in the URL, including data in the query string. If the URL exceeds 2,048 characters, an error may result, or data in the query string might be truncated or not sent with the request. In ASP.NET Web pages, a GET request that includes post data can easily exceed 2,048 characters if view state information (which is stored in a hidden field) is part of the request, resulting in errors. Other browsers may not have this limitation. For more information, see article 208427, "Maximum URL Length Is 2,083 Characters in Internet Explorer" in the Microsoft Knowledge Base at http://support.microsoft.com. To work around this limitation when you are trying to share information among pages, you can redirect users by using HTTP POST requests as discussed in this topic. If your application requires an HTTP GET request, you can store information in an alternative way, without using a query string, such as in session state. |

[Redirecting Programmatically on the Server](javascript:void(0))

You can also redirect programmatically to a target page on the server by calling the [Transfer](http://msdn.microsoft.com/en-us/library/system.web.httpserverutility.transfer.aspx) method. In this scenario, the server simply transfers the current source page context to the target page. The target page then renders in place of the source page. The source and target pages must be in the same Web application. As with cross-page posting, the Transfer method has the advantage that it enables the target page to read control values and public property values from the source page.

Because the transfer between source and target pages happens on the server, the browser has no information about the changed page, and it retains information about the original (source) URL. For example, the Address box in Internet Explorer does not change after a transfer, and instead continues to show the URL of the page it most recently requested (which is usually the source page). The browser's history is not updated to reflect the transfer. This can result in unexpected behavior if the user refreshes the page in the browser or clicks the browser's back button. Therefore, calling the Transfer method is a strategy that is best used in applications where you are presenting pages to the user with the URL hidden.

[Selecting a Redirect Option](javascript:void(0))

The following table summarizes the possible ways to redirect between pages.

|  |  |  |
| --- | --- | --- |
| **Strategy** | **Characteristics** | **Usage** |
| Hyperlinks | * Performs new request on the target page. * Does not pass current page information to the target page. * Requires user initiation. * Redirects to any page, not just pages in the same Web application. * Enables you to share information between pages using query string or session state. (The [HyperLink](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.hyperlink.aspx) control enables you to create URL and query strings programmatically.) | * For navigation without any additional processing, as in menus or lists of links. * When navigation should be under user control. |
| Cross-page posting | * Posts current page information to the target page. * Makes post information available in the target page. * Requires user initiation. * Redirects to any page, not just pages in the same Web application. * Enables the target page to read public properties of the source page if the pages are in the same Web application. | * To pass current page information to the target page (as in multi-page forms). * When navigation should be under user control. |
| Browser redirect | * Performs a new HTTP GET request on the target page. * Passes the query string (if any) to the target page. In Internet Explorer, the size of the query string is limited to 2,048 characters. * Provides programmatic and dynamic control over the target URL and query string. * Enables you to redirect to any page, not just pages in the same Web application. * Enables you to share information between source and target pages using session state. | * For conditional navigation, when you want to control the target URL and control when navigation takes place. For example, use this option if the application must determine which page to navigate to based on data provided by the user. |
| Server transfer | * Transfers control to a new page that renders in place of the source page. * Redirects only to target pages that are in the same Web application as the source page. * Enables you to read values and public properties from source page. * Does not update browser information with information about the target page. Pressing the refresh or back buttons in the browser can result in unexpected behavior. | * For conditional navigation, when you want to control when navigation takes place and you want access to the context of the source page. * Best used in situations where the URL is hidden from the user. |

# How to: Post ASP.NET Web Pages to a Different Page

**.NET Framework 4**

[Other Versions](javascript:;)

http://i.msdn.microsoft.com/Areas/Epx/Content/Images/ImageSprite.png

* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/ms178140(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/ms178140(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/ms178140(d=printer,v=vs.80).aspx)

11 out of 25 rated this helpful - [Rate this topic](http://msdn.microsoft.com/en-us/library/ms178140(d=printer).aspx#feedback)

By default, controls on ASP.NET Web pages that cause a postback, such as the Button control, post back to the page for processing. However, you can configure controls to post to a different page. For example, you might be creating a multi-page form that collects different information on each page.

On the target page, you can read the values of controls or public properties from the source page. For more information, see [Cross-Page Posting in ASP.NET Web Pages](http://msdn.microsoft.com/en-us/library/ms178139.aspx).

### To post an ASP.NET Web page to another page

1. Add a button control to your Web page, such as a [Button](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.button.aspx), [LinkButton](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.linkbutton.aspx), or [ImageButton](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.imagebutton.aspx) control.
2. Set the [PostBackUrl](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.ibuttoncontrol.postbackurl.aspx) property for the control to the URL of the page to which you want to post the ASP.NET Web page.

The following code example illustrates a [Button](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.button.aspx) control that is configured to post to a page named TargetPage in the root of the Web site.

<asp:Button

ID="Button1"

PostBackUrl="~/TargetPage.aspx"

runat="server"

Text="Submit" />