**How to: Pass Values Between 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/6c3yckfw(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/6c3yckfw(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/6c3yckfw(d=printer,v=vs.80).aspx)
* [.NET Framework 1.1](http://msdn.microsoft.com/en-us/library/6c3yckfw(d=printer,v=vs.71).aspx)

If your application redirects (navigates) from one ASP.NET Web page to another, you will frequently want to pass information from the source page to the target page. For example, you might have a page where users can select items to purchase. When users submit the page, you want to call another page that can process the information that the user has entered.

A Visual Studio Web application project with source code is available to accompany this topic: [Download](http://go.microsoft.com/fwlink/?LinkId=214100).

|  |
| --- |
| **NoteNote** |
| For information about ways to navigate from one page to another in an ASP.NET Web site, see [How to: Redirect Users to Another Page](http://msdn.microsoft.com/en-us/library/540y83hx.aspx). |

You can pass information between pages in various ways, some of which depend on how the redirection occurs. The following options are available even if the source page is in a different ASP.NET Web application from the target page, or if the source page is not an ASP.NET Web page:

* Use a query string.
* Get HTTP POST information from the source page.

The following options are available only when the source and target pages are in the same ASP.NET Web application.

* Use session state.
* Create public properties in the source page and access the property values in the target page.
* Get control information in the target page from controls in the source page.

[Using a Query String](javascript:void(0))

When you use a hyperlink or Response.Redirect to navigate from one page to another, you can add information in a query string at the end of the URL.

|  |
| --- |
| **NoteNote** |
| Never pass sensitive data using a query string, because the information is visible to users and can easily be modified, thus representing a potential security risk. |

For more information, see [QueryString](http://msdn.microsoft.com/en-us/library/system.web.httprequest.querystring.aspx).

**To use a query string to pass information**

1. In the source page when you specify the URL of the target page, include the information that you want to pass in the form of key-value pairs at the end of the URL. The first pair is preceded by a question mark (?) and subsequent pairs are preceded by ampersands(&), as shown in the following example:

http://contoso.com/products.aspx?field1=value1

http://contoso.com/products.aspx?field1=value1&field2=value2

1. In the target page, access query string values by using the [QueryString](http://msdn.microsoft.com/en-us/library/system.web.httprequest.querystring.aspx) property of the [HttpRequest](http://msdn.microsoft.com/en-us/library/system.web.httprequest.aspx) object, as shown in the following example:

VB

Dim s As String

s = Request.QueryString("field1")

C#

String s = Request.QueryString["field1"];

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

When the source page uses the HTTP POST action to navigate to the target page, you can retrieve posted values from the [Form](http://msdn.microsoft.com/en-us/library/system.web.httprequest.form.aspx) collection in the target page. Note that you can get only the post values; you cannot read the values of arbitrary controls on the page.

|  |
| --- |
| **Security noteSecurity Note** |
| Do not pass sensitive data in posted values. Although posted values are not as readily available to users as query strings are, they can be viewed and changed using commonly available browser tools. |

**To get the values of controls from the source page in another application**

1. In the source page, include a form element that contains HTML elements (such as input or textarea) or ASP.NET server controls (such as [TextBox](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.textbox.aspx) or [DropDownList](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.dropdownlist.aspx) controls) that post values when the form is submitted.
2. In the target page, read the [Form](http://msdn.microsoft.com/en-us/library/system.web.httprequest.form.aspx) collection, which returns a dictionary of name/value pairs, one pair for each posted value.

The following code example displays the ID and value of every posted control in the source page and displays the posted values in a label named Label1.

|  |
| --- |
| **NoteNote** |
| Post information from an ASP.NET Web pages includes the values of hidden fields, such as \_\_VIEWSTATE, \_\_EVENTTARGET, and \_\_EVENTARGUMENT, which are used for internal processing in the page. The following code example excludes the values of posted fields that are named with a leading double underscore (\_\_). |

VB

Sub Page\_Load(ByVal sender As Object, ByVal e As System.EventArgs) \_

Handles Me.Load

Dim displayValues As New StringBuilder()

Dim postedValues As NameValueCollection = Request.Form

Dim nextKey As String

For i As Integer = 0 To postedValues.AllKeys.Length - 1

nextKey = postedValues.AllKeys(i)

If nextKey.Substring(0, 2) <> "\_\_" Then

displayValues.Append("<br>")

displayValues.Append(nextKey)

displayValues.Append(" = ")

displayValues.Append(postedValues(i))

End If

Next

Label1.Text = displayValues.ToString()

End Sub

C#

void Page\_Load(object sender, EventArgs e)

{

System.Text.StringBuilder displayValues =

new System.Text.StringBuilder();

System.Collections.Specialized.NameValueCollection

postedValues = Request.Form;

String nextKey;

for(int i = 0; i < postedValues.AllKeys.Length; i++)

{

nextKey = postedValues.AllKeys[i];

if(nextKey.Substring(0, 2) != "\_\_")

{

displayValues.Append("<br>");

displayValues.Append(nextKey);

displayValues.Append(" = ");

displayValues.Append(postedValues[i]);

}

}

Label1.Text = displayValues.ToString();

}

[Using Session State](javascript:void(0))

Information in session state is available to all ASP.NET Web pages in the current application. However, session state takes server memory, and the information is stored until the session expires, which can be more overhead than you want for simply passing information to the next page. For more information, see [How to: Save Values in Session State](http://msdn.microsoft.com/en-us/library/6ad7zeeb.aspx) and [How to: Read Values from Session State](http://msdn.microsoft.com/en-us/library/03sekbw5.aspx).

**To use session state to pass information**

1. In the source page, save the information that you want to pass in session state , as shown in the following example:

VB

Session("field1") = "value1"

C#

Session["field1"] = "value1";

1. In the target page, read the saved information from session state, as shown in the following example:

C#

[VB](http://msdn.microsoft.com/en-us/library/6c3yckfw(d=printer).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-8)

string field1 = (string)(Session["field1"]);

[Getting Public Property Values from the Source Page](javascript:void(0))

If the source page and target page are both ASP.NET Web pages in the same Web application, and if you transfer execution from the source page to the target page on the server by using the [Transfer](http://msdn.microsoft.com/en-us/library/system.web.httpserverutility.transfer.aspx) method, the target page can access public properties in the source page.

**To get public property values from the source page**

1. On the source page, create one or more public properties and save the page.

The following code example shows a 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.

VB

Public ReadOnly Property CurrentCity() As String

Get

Return textCity.Text

End Get

End Property

C#

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. |

1. On the target page, add a [@ PreviousPageType](http://msdn.microsoft.com/en-us/library/ms228169.aspx) page directive that points to the source page.

The following code example shows a PreviousPageType directive that references a source page named SourcePage.aspx.

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

The PreviousPageType directive causes the page's PreviousPage property to be typed to the source page class.

1. In target page code, use strongly typed members of the PreviousPage property to read the source code properties.

The following code example reads the value of the CurrentCity property that is defined in the source page.

VB

Label1.Text = PreviousPage.CurrentCity

C#

C#

Label1.Text = PreviousPage.CurrentCity;

[Getting Control Information from the Source Page in the Same Application](javascript:void(0))

If the source and target pages are both ASP.NET Web pages and in the same Web application, you can read the values of controls on the source page while in the target page. You might use this strategy if the source page does not expose public properties containing the information you need.

**To get the values of controls from the source page in the same application**

* On the target page, get a reference to the source page by using the target page's [PreviousPage](http://msdn.microsoft.com/en-us/library/system.web.ui.page.previouspage.aspx) property, and then call the [FindControl](http://msdn.microsoft.com/en-us/library/system.web.ui.control.findcontrol.aspx) method to get a reference to the control you want.

The following code example gets the value of the source page's TextBox1 control and displays it in the control named Label1:

VB

If Not Page.PreviousPage Is Nothing Then

Dim SourceTextBox As TextBox

SourceTextBox = CType(PreviousPage.FindControl("TextBox1"), \_

TextBox)

If Not SourceTextBox Is Nothing Then

Label1.Text = SourceTextBox.Text

End If

End If

C#

if (PreviousPage != null)

{

TextBox SourceTextBox =

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

if (SourceTextBox != null)

{

Label1.Text = SourceTextBox.Text;

}

}

The FindControl method finds controls in the current naming container. If the control you are looking for is inside another control, you must first get a reference to the container, and then search the container to find the control that you want. A typical example of this situation is when the previous page is a master page and the control that you want to find is inside a [ContentPlaceHolder](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.contentplaceholder.aspx) control. The following example is similar to the previous one except that it assumes that TextBox1 is located in a [ContentPlaceHolder](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.contentplaceholder.aspx) control that is named ContentPlaceHolder1:

VB

If Not Page.PreviousPage Is Nothing Then

Dim placeHolder As Control =

PreviousPage.Controls(0).FindControl("ContentPlaceHolder1")

Dim SourceTextBox As TextBox =

CType(placeHolder.FindControl("TextBox1"), TextBox)

If Not SourceTextBox Is Nothing Then

Label1.Text = SourceTextBox.Text

End If

End If

C#

if (PreviousPage != null)

{

Control placeHolder =

PreviousPage.Controls[0].FindControl("ContentPlaceHolder1")

TextBox SourceTextBox =

(TextBox)placeHolder.FindControl("TextBox1");

if (SourceTextBox != null)

{

Label1.Text = SourceTextBox.Text;

}

}

For information about how to get a reference to a control when you do not have a reference to the naming container, see [How to: Access Server Controls by ID](http://msdn.microsoft.com/en-us/library/y81z8326.aspx).

**How to: Access Server Controls by ID**

**.NET Framework 4**

[Other Versions](javascript:;)



* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/y81z8326(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/y81z8326(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/y81z8326(d=printer,v=vs.80).aspx)

When a control is not inside a naming container, you can get a reference to it by using the control's ID. When a control is inside a naming container, you must call a method that searches the naming container for the control's ID. A control might also be inside a naming container that you do not have direct access to.

**To locate a control that is not inside a naming container**

* Reference the control's ID to access the object.

The following example includes code that shows how to access a control that is not inside a naming container. The Label control named Message is not within a naming container and therefore can be accessed by ID.

C#

[VB](http://msdn.microsoft.com/en-us/library/y81z8326(d=printer).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-1)

<%@ Page language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">

void Page\_Load()

{

Message.Text = String.Empty;

}

void ProductsListView\_SelectedIndexChanging(Object sender, ListViewSelectEventArgs e)

{

ListViewItem item = (ListViewItem)ProductsListView.Items[e.NewSelectedIndex];

Label l = (Label)item.FindControl("DiscontinuedDateLabel");

if (String.IsNullOrEmpty(l.Text))

{

return;

}

DateTime discontinued = DateTime.Parse(l.Text);

if (discontinued < DateTime.Now)

{

Message.Text = "You cannot select a discontinued item.";

e.Cancel = true;

}

}

protected void ProductsListView\_PagePropertiesChanging(object sender, PagePropertiesChangingEventArgs e)

{

// Clear selection.

ProductsListView.SelectedIndex = -1;

}

</script>

<html xmlns="http://www.w3.org/1999/xhtml" >

<head id="Head1" runat="server">

<title>ListView.SelectedIndexChanging Example</title>

</head>

<body>

<form id="form1" runat="server">

<h3>ListView.SelectedIndexChanging Example</h3>

<asp:Label ID="Message"

ForeColor="Red"

runat="server"/>

<br/>

<asp:ListView ID="ProductsListView"

DataSourceID="ProductsDataSource"

DataKeyNames="ProductID"

OnSelectedIndexChanging="ProductsListView\_SelectedIndexChanging"

OnPagePropertiesChanging="ProductsListView\_PagePropertiesChanging"

runat="server" >

<LayoutTemplate>

<table cellpadding="2" runat="server" id="tblProducts" width="640px">

<tr runat="server" id="itemPlaceholder" />

</table>

<asp:DataPager runat="server" ID="ProductsDataPager" PageSize="12">

<Fields>

<asp:NextPreviousPagerField

ShowFirstPageButton="true" ShowLastPageButton="true"

FirstPageText="|&lt;&lt; " LastPageText=" &gt;&gt;|"

NextPageText=" &gt; " PreviousPageText=" &lt; " />

</Fields>

</asp:DataPager>

</LayoutTemplate>

<ItemTemplate>

<tr runat="server">

<td valign="top">

<asp:LinkButton ID="SelectButton" runat="server" Text="..." CommandName="Select" />

</td>

<td valign="top">

<asp:Label ID="NameLabel" runat="server" Text='<%#Eval("Name") %>' />

</td>

<td valign="top">

<asp:Label ID="ProductNumberLabel" runat="server" Text='<%#Eval("ProductNumber") %>' />

</td>

<td>

<asp:Label ID="DiscontinuedDateLabel" runat="server" Text='<%#Eval("DiscontinuedDate", "{0:d}") %>' />

</td>

</tr>

</ItemTemplate>

<SelectedItemTemplate>

<tr runat="server" style="background-color:#ADD8E6">

<td>&nbsp;</td>

<td valign="top">

<asp:Label ID="NameLabel" runat="server" Text='<%#Eval("Name") %>' />

</td>

<td valign="top">

<asp:Label ID="ProductNumberLabel" runat="server" Text='<%#Eval("ProductNumber") %>' />

</td>

<td>

<asp:Label ID="DiscontinuedDateLabel" runat="server" Text='<%#Eval("DiscontinuedDate", "{0:d}") %>' />

</td>

</tr>

</SelectedItemTemplate>

</asp:ListView>

<asp:SqlDataSource ID="ProductsDataSource" runat="server"

ConnectionString="<%$ ConnectionStrings:AdventureWorks\_DataConnectionString %>"

SelectCommand="SELECT [ProductID], [Name], [ProductNumber], [DiscontinuedDate]

FROM Production.Product"

UpdateCommand="UPDATE Production.Product

SET Name = @Name, ProductNumber = @ProductNumber, DiscontinuedDate = @DiscontinuedDate

WHERE ProductID = @ProductID">

</asp:SqlDataSource>

</form>

</body>

</html>

**To locate a control that is inside a naming container when you have a reference to the naming container**

* Call the [FindControl](http://msdn.microsoft.com/en-us/library/system.web.ui.control.findcontrol.aspx) method of the naming container, passing a string that contains the ID of the control that you want to use. The method returns an object of type [Control](http://msdn.microsoft.com/en-us/library/system.web.ui.control.aspx) that you must cast to the appropriate type.

The example in the previous procedure also shows how you can access a control that is inside a naming container. The Label control named DiscontinuedDateLabel is inside a [ListView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.listview.aspx) control. Therefore, to access the Label control ,you must call the [FindControl](http://msdn.microsoft.com/en-us/library/system.web.ui.control.findcontrol.aspx) method of the [ListView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.listview.aspx) control.

[Locating a Control Inside a Hierarchy of Naming Containers](javascript:void(0))

Sometimes, a control is inside a naming container but you do not have a reference to the naming container. In that case, one way to get a reference to the control is to write a custom method that searches the controls in a hierarchy of naming containers.

**To locate a control by searching through a hierarchy of naming containers**

* Create a method that can be called recursively to search a naming control and its child naming containers.

The following example shows one way to write a search method.

C#

[VB](http://msdn.microsoft.com/en-us/library/y81z8326(d=printer).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-2)

private Control FindControlRecursive(Control rootControl, string controlID)

{

if (rootControl.ID == controlID) return rootControl;

foreach (Control controlToSearch in rootControl.Controls)

{

Control controlToReturn =

FindControlRecursive(controlToSearch, controlID);

if (controlToReturn != null) return controlToReturn;

}

return null;

}

This method accepts a reference to a naming container. If you do not know which naming container on the page has the control that you are looking for, you can pass in the page itself as the top-level naming container. The method looks through all controls in the naming container that you pass to it. If it does not find the requested control, the method calls itself recursively for each lower-level naming container.