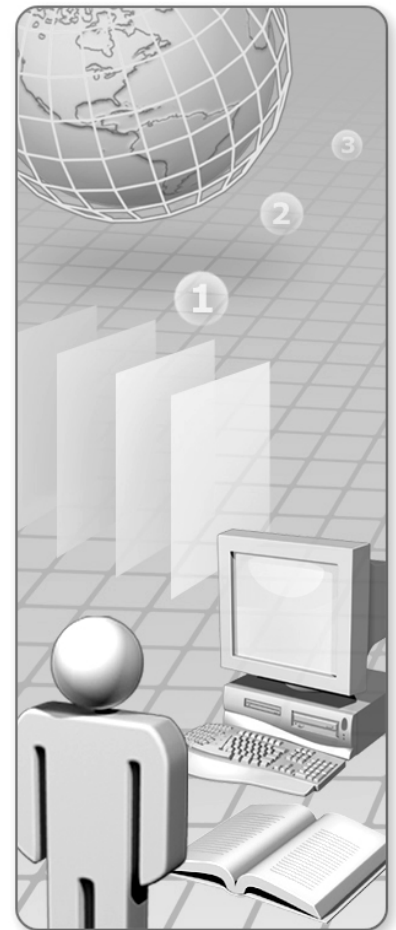


Unit 3: Adding and Configuring Server Controls

Contents

Overview	1
HTML Controls and Web Server Controls	2
Types of Web Server Controls	4
Working with Web Server Controls	7
The ASP.NET 2.0 Page Postback Model	9
Lab Scenario	11
Lab Tasks and Objectives	12
Lab: Adding and Configuring Server Controls	14
Lab Discussion	35



Overview



- HTML Controls and Web Server Controls
- Types of Web Server Controls
- Working with Web Server Controls
- The ASP.NET 2.0 Page Postback Model
- Lab Scenario
- Lab Tasks and Objectives
- Lab: Adding and Configuring Server Controls

Introduction

This unit explains how to use the HTML controls and Web server controls provided by Microsoft® Visual Studio® 2005 and Microsoft ASP.NET 2.0. It shows you how to design and build Web-based user interfaces, and it teaches you how to program Web server controls. This unit also describes how the ASP.NET 2.0 postback model works and how you can use it.

Objectives

After completing this unit, you will be able to:

- Explain the differences between HTML controls and Web server controls.
- Describe the different types of Web server controls.
- Explain how to use HTML controls and Web server controls.
- Explain how the postback model of ASP.NET 2.0 works.
- Create Web-based user interfaces with HTML controls and Web server controls.
- Write code that interacts with Web server controls.
- Write code that interacts with the postback model of ASP.NET 2.0.

HTML Controls and Web Server Controls



- What Are HTML Controls?
- What Are Web Server Controls?
- Comparing HTML Controls with Web Server Controls

Introduction

When you create a Web page by using Visual Studio 2005 and ASP.NET, you can define the page's appearance with a mixture of static HTML text, HTML controls, Web server controls, and the output generated by using server-side processing, such as the output from the **Response.Write** method.

What Are HTML Controls?

HTML controls are representations of HTML markup that is rendered by Web browsers. You use HTML controls to add static HTML elements to an ASP.NET Web page. Visual Studio 2005 enables you to add HTML controls to your Web page by dragging and dropping them from the Toolbox to your page. You can manipulate the attributes of the HTML elements by using the Properties window or by editing the HTML markup in Source view. Furthermore, Visual Studio 2005 provides IntelliSense® features that help you edit the HTML markup when you work directly with HTML elements in Source view.

By default, HTML controls are not accessible to the server-side code in the Web page. However, in many circumstances, it will be necessary to manipulate an HTML control's properties in your server-side code. To enable your server-side code to interact with HTML controls, you must specify that the control runs on the server. To do this, you can either:

- Right-click the control in Design view and then click **Run as Server Control** from the shortcut menu.
- Edit the HTML markup for the control by adding the **runat="server"** attribute.

When a browser requests a page that contains HTML controls, ASP.NET returns the static HTML text defined by the controls. For example, if you have added an HTML text input control to a Web page, the HTML markup for that control is sent to the browser.

What Are Web Server Controls?

Web server controls are .NET Framework objects that are converted by ASP.NET to HTML elements at run time. When a browser requests a page that contains Web server controls, ASP.NET generates HTML output based on the definition of the Web server controls on the page and returns that output to the browser. For example, if you have added a **DropDownList** Web server control to a Web page, ASP.NET sends a **SELECT** HTML element to browser. The **SELECT** element is the HTML equivalent of a **DropDownList** Web server control.

Web server controls include not only form-type controls such as buttons and text boxes, but also special-purpose controls such as calendars and tree views.

Comparing HTML Controls with Web Server Controls

When you design Web pages with Visual Studio 2005 and ASP.NET, you will encounter situations in which you can include either an HTML control or an equivalent Web server control. For example, to enable users to enter their names on a Web page, you can use an **HtmlInputText** HTML control or a **TextBox** Web server control.

The type of control you choose depends on how your application uses it. Web server controls are more abstract than HTML server controls in that their object model does not necessarily reflect HTML syntax. This means that Web server controls are more versatile if you are going to run server-side code associated with the controls, but they incur more overhead when processing the page. HTML controls incur less processing overhead on the server but do not support such a rich programming model as their Web server control equivalents.

Types of Web Server Controls



- Standard Controls
- Data Controls
- Validation Controls
- Navigation Controls
- Login Controls
- WebPart Controls

Introduction

Visual Studio 2005 groups Web server controls together in the Toolbox, based on their functionality.

Standard Controls

The **Standard** group in the Toolbox contains controls that provide common user interface elements. Controls in this group include the following:

- **TextBox**
- **ListBox**
- **DropDownList**
- **Checkbox**
- **RadioButton**
- **Button**
- **Image**
- **Table**
- **Calendar**

There are other types of standard controls in addition to those on this list. Refer to this page in the Microsoft Visual Studio 2005 documentation for a full list of standard Web server controls: ms-help://MS.MSDNQTR.v80.en/MS.MSDN.v80/MS.VisualStudio.v80.en/dv_ywdcon/html/241dbc4a-7046-404f-827b-bafc51ec5410.htm.

Data Controls

The **Data** group in the Toolbox contains controls that you can use to manipulate data stored in relational databases, XML files, and other .NET Framework objects. Controls in this group include:

- Those that can provide visual representation of data, such as the **GridView** control.
- Those that provide access to data, such as the **SqlDataSource** control.

There are other types of data controls in addition to those in this list. Refer to the following page in the Microsoft Visual Studio 2005 documentation for a full list of data Web server controls:

ms-help://MS.MSDN.QTR.v80.en/MS.MSDN.v80/MS.VisualStudio.v80.en/dv_vwdcon/html/e1363987-bacd-4e79-bbc3-1a49c5ed68fa.htm.

Validation Controls

The **Validation** group in the Toolbox contains controls that you can use to validate user input. Controls in this group include:

- The **RequiredFieldValidator** control that ensures that the user does not omit a mandatory field.
- The **CompareValidator** control that validates a user's entry against a constant value, or a property value of another control.

The validation controls can perform simple validation in the user's Web browser by generating JavaScript code (if the browser supports it), as well as more complex validation on the Web server. Performing validation in the Web browser can help to reduce the load on the Web server as well as improve the responsiveness of the Web page.

There are other types of validation controls in addition to those in this list. Refer to the following page in the Microsoft Visual Studio 2005 documentation for a full list of validation Web server controls:

ms-help://MS.MSDN.QTR.v80.en/MS.MSDN.v80/MS.VisualStudio.v80.en/dv_aspnetcon/html/df04bdd2-7411-45c8-8f35-d9ccf3d80a66.htm.

Navigation Controls

The **Navigation** group in the Toolbox contains controls that you can use to help the user move through the Web site. Controls in this group include:

- The **Menu** control for building navigational menus.
- The **TreeView** control for displaying hierarchical information (such as a Web site navigational structure).
- The **SiteMapPath** control used for creating breadcrumb trails to current Web pages.
A *breadcrumb* trail is a graphical display of the path from the home page of the Web application to the current page. Each step in the breadcrumb trail can be rendered as a hyperlink, so users can easily find their way back to the home page.

Login Controls

The **Login** group in the Toolbox contains controls that you can use to create login and sign-up pages for your Web sites. Controls in this group include:

- The **Login** control, which provides a graphical user interface for authentication, including text boxes for the user name and password.
- The **ChangePassword** control, which provides text boxes for the original password, the new password, and a confirmation of the new password.
- The **PasswordRecovery** control, which allows user passwords to be retrieved based on the e-mail address that was used when the account was created. The **PasswordRecovery** control sends an e-mail containing the user password.

There are also other types of login controls. They all integrate with the ASP.NET membership systems to help automate management of user membership and authentication for your Web site. Refer to the following page in the Microsoft Visual Studio 2005 documentation for a full list of login Web server controls:

ms-help://MS.MSDNQTR.v80.en/MS.MSDN.v80/MS.VisualStudio.v80.en/dv_ywdcon/html/f4ee4ece-c984-4bf7-bfde-f2365c06026d.htm

WebPart Controls

The WebPart group in the Toolbox contains controls that are typically used in portal-type Web applications. At run time, the user can modify the properties of **WebPart** controls (and even choose to add and remove **WebPart** controls) so that they experience a highly personalized view of the application. For example, one user might choose a selection of **WebPart** controls for the home page that they find most useful, whereas another user might select a completely different set of **WebPart** controls.

The **WebPart** group of the Toolbox provides controls that you can use to build the framework for dynamic Web pages. Controls in this group include:

- The **WebPartManager** control, which enables **WebPart** controls to be added and managed on a **WebPart** page.
- The **WebPartZone** control, which defines an area of the page that users can add **WebPart** controls to.

There are other types of **WebPart** controls in addition to those in this list. Refer to the following page in the Visual Studio 2005 documentation for a full list of **WebPart** controls:

ms-help://MS.MSDNQTR.v80.en/MS.MSDN.v80/MS.VisualStudio.v80.en/dv_aspnetcon/html/ff0149e4-695a-401a-8cda-53df6d7d2668.htm.

Working with Web Server Controls



- Adding Web Server Controls to Web Forms
- Setting Web Server Control Properties at Design Time
- Manipulating Web Server Controls at Run Time

Introduction

Visual Studio 2005 enables you to add Web server controls in a variety of ways. You can choose a method that suits your way of working.

Adding Web Server Controls to Web Forms

You can add Web server controls to your forms by using any of the following methods:

- Dragging and dropping them from the Toolbox onto the Design view of the Web page.
This approach enables you to see the page layout and positioning of the controls as you add them to the page.
- Dragging and dropping them from the Toolbox into the Source view of the Web page.
This approach enables you to more accurately place the controls in the flow of the HTML markup. However, be aware that it is possible to drag and drop a control into an inappropriate area of the HTML markup, such as inside the element definitions of other controls or HTML fragments.
- Typing the markup text for the control directly into the Source view of the Web page.
This approach enables you to set only the attributes for the controls that you want to. The IntelliSense features of Visual Studio 2005 help you to specify the correct attributes for each type of Web server control, but this approach relies on your being familiar with the required structure and attributes for each control.

Setting Web Server Control Properties at Design Time

You can set the properties of Web server controls at design time by using any of the following methods:

- Selecting the control in the Design view of the Web page, and then setting property values in the Properties window
- Selecting the markup text for the control in the Source view of the Web page and then setting the property values in the Properties window
- Editing the markup text for the control directly in the Source view of the Web page

Manipulating Web Server Controls at Run Time

You can set writeable properties and invoke methods of Web server controls in your application code at run time. You can also read the values of their readable properties. The following examples show how to manipulate Web server controls at run time:

```
'[Visual Basic]
'Set a writeable property
TextBox1.Text="Hello World!"
'Invoke a method
TextBox1.Focus()
'Read a readable property
Dim sName as String = TextBox1.Text
//[C#]
//Set a writeable property
TextBox1.Text="Hello World!";
//Invoke a method
TextBox1.Focus();
//Read a readable property
string sName = TextBox1.Text;
```

The ASP.NET 2.0 Page Postback Model



- Round-Trip Processing
- AutoPostBack Properties
- EnableViewState Properties
- Cross-Page Postbacks

Introduction

The ASP.NET Postback model provides a mechanism for sending control properties on Web pages from the Web browser to the Web server and for restoring those values when a response is sent back from the Web server to the Web browser. The postback model is designed to provide stateful rendering of dynamic content on a Web page, requiring minimal developer effort.

Round-Trip Processing

The ASP.NET Postback model enables Web server controls to retain their values over multiple requests to the server, even though the underlying HTTP mechanisms are stateless. The ASP.NET Postback model enables you to develop Web pages as if they are part of a stateful application, by posting information from the browser to the server, where server-side events will then be raised; and by repopulating control properties in response to these posting operations.

AutoPostBack Property

Some Web server controls support the **AutoPostBack** property. This property controls whether user interaction with the control should invoke a round-trip request to the server. For example, if the values in a **ListBox** control are dependent on the selected item in a **DropDownList** control, and you can manage this dependency only in server-side code, set the **AutoPostBack** property of the **DropDownList** control to **true**. Then add server-side code to respond to the user changing the selection in the **DropDownList**. The server-side code should evaluate the newly selected item and populate the **ListBox** control accordingly.

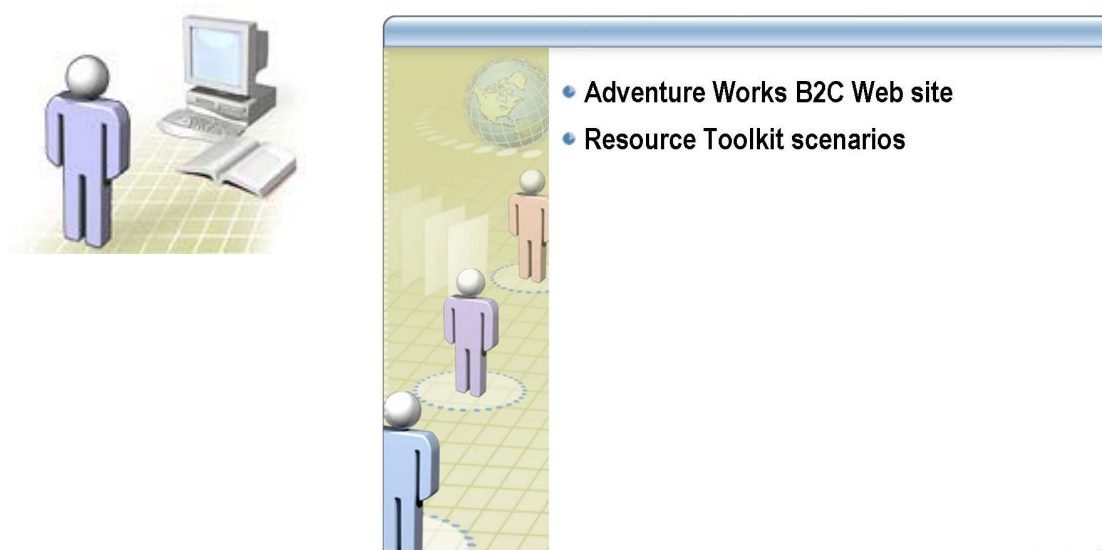
EnableViewState Property

The **EnableViewState** property of a Web server control determines whether the control should retain its state for the duration of the postback. You need to set this property to **true** only for controls whose properties you set in server-side code.

Cross-Page Postbacks

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. In some situations, however, you might want to post the values on one page to another page for processing. In this case, you can configure controls to post requests to a different page by setting their **PostBackUrl** properties.

Lab Scenario



Adventure Works B2C Web Site

You are a developer in the Adventure Works organization, a fictitious bicycle manufacturer. You have been asked to assist in the development of the Business-to-Consumer (B2C) Web application and a related Business-to-Employee (B2E) extranet portal.

Decisions on the design of the application have already been taken. You have been asked to carry out a number of specific tasks to implement various elements of this design. As part of the first phase of the B2C development, you have been asked to complete the prototypes for the following pages:

- *Survey.aspx*. You will design a rich graphical user interface for this page that enables users to submit responses to an online survey. You will also add a **SiteMapPath** control to this page to aid user navigation.
- *SurveyReceipt.aspx*. You will design this page to receive the information provided by the user in the *Survey.aspx* page.
- *FeedbackForm.aspx*. You will design this page to receive the information provided by the user in the existing *Feedback.aspx* page.
- *Default.aspx*. You will add a **Menu** control to this page to aid user navigation.

You will also add and review a *Web.sitemap* file that will be used to add navigation features to the Web application. Additionally, you will add and review an advertising schedule file that will be used to add dynamic image display features to the Web application.

Resource Toolkit Scenarios

Before you start work on the lab, you should review the Scenario tab in the Resource Toolkit. The instructor will lead a group discussion of the scenarios for this lab.

Lab Tasks and Objectives



Task	Objectives
Build graphical user interfaces with HTML controls	<ul style="list-style-type: none"> Define a page layout with HTML controls Add interactive HTML controls to a page
Build graphical user interfaces with Web server controls	<ul style="list-style-type: none"> Define a page layout with Web server controls Add display-oriented Web server controls to the page Add interactive Web server controls to the page
Write code that controls the graphical user interface	<ul style="list-style-type: none"> Manipulate Web server control properties at run time Call Web server control methods at run time Write code for page postbacks Write code for cross-page postbacks

Lab Tasks

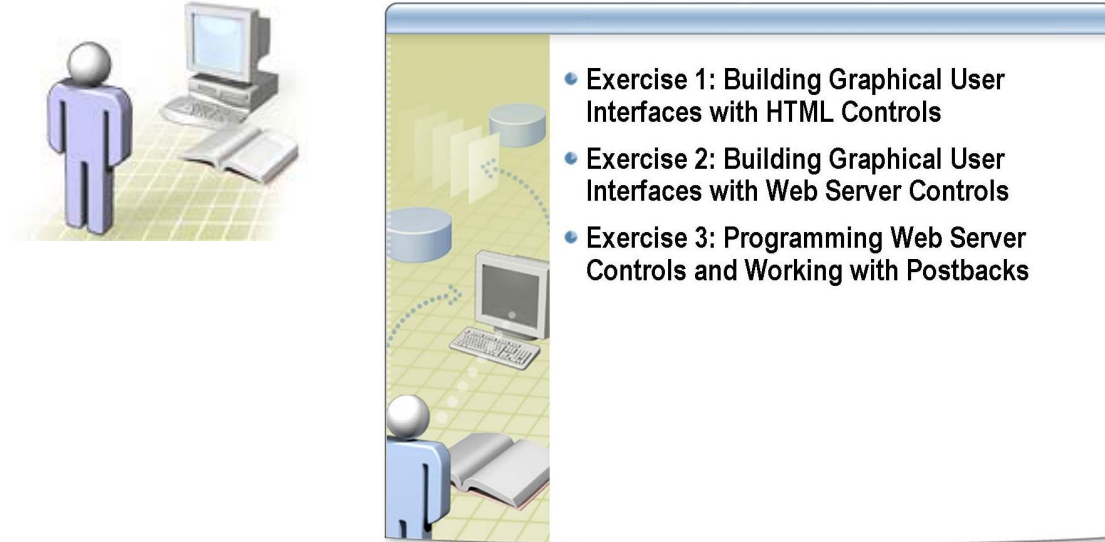
There are three main tasks in this lab. Each one is designed to help you achieve one or more learning objectives. Resources are provided in the Resource Toolkit to help you complete the tasks. You should try to complete all of the tasks.

The tasks in this lab are:

- *Build Graphical User Interfaces with HTML Controls.* In this task, you will define a page layout and add interactivity to the page with HTML controls. The resources for this task are titled:
 - *Survey.aspx Page Design–HTML Controls*
 - *How to: Add HTML Server Controls to a Web Forms Page by Using the Web Forms Designer*
 - *HTML Controls for ASP.NET Web Pages*
 - *How to: Add HTML Markup to Web Pages*
- *Build Graphical User Interfaces with Web Server Controls.* In this task, you will define a page layout and add interactivity to the page with Web server controls. The resources for this task are titled:
 - *Survey.aspx Page Design–Web Server Controls*
 - *Introduction to ASP.NET Web Server Controls*
 - *Standard Toolbox Controls*
 - *How to: Create a Simple Web Page by Using Visual Web Developer*
 - *How to: Set ASP.NET Control Properties*
 - *How to: Add Web Server Controls to a Web Forms Page by Using the Web Forms Designer*
 - *Navigation Controls*

-
- *Write Code that Controls the Graphical User Interface.* In this task, you will write code to manipulate the properties of Web server controls and call Web server control methods at run time. You will also add server-side code for the page postbacks and cross-page postback. The resources for this task are titled:
 - *The Button.OnClientClick Property*
 - *How to: Determine How an ASP.NET Web Page was Invoked*
 - *Cross Page Posting in an ASP.NET Web Page*
 - *How to: Post an ASP.NET Page to a Different Page*

Lab: Adding and Configuring Server Controls



After completing this lab, you will be able to:

- Create Web-based user interfaces with HTML controls and Web server controls.
- Write code that interacts with Web server controls.
- Write code that interacts with the postback model of ASP.NET 2.0.

Estimated time to complete this lab: **120 minutes**



Important You can choose to perform this workshop by using either Microsoft Visual Basic® or Microsoft Visual C#®. Code samples and lab solutions are provided in both languages.

Lab Solutions

There are Visual Basic and Visual C# solution files associated with the labs in this workshop. You can find the lab solution files in the folder E:\Labfiles\Solution on the virtual machines.

Lab Setup

For this lab, you will use the available Microsoft® Virtual PC environment. Before you begin the lab, you must:


1. Start the **2543B-LON-DEV-01-03** virtual machine.
2. Log on to the **2543B-LON-DEV-01-03** virtual machine with the user name **Student** and the password **Pa\$\$w0rd**.

Exercise 1




Building Graphical User Interfaces with HTML Controls

In this exercise, you will start to build an online survey form for the Adventure Works Web application. You will define the layout of the Survey.aspx page, and you will add HTML controls to provide a rich graphical user interface.




Scenario

Tasks	Supporting information
1. Review the proposed design of the Survey.aspx page.	 See the resource in the Resource Toolkit, “Survey.aspx Page Design–HTML Controls.”
2. Open the Adventure Works Web site.	<ol style="list-style-type: none"> If you want to complete this lab by using Visual Basic: <ul style="list-style-type: none"> Use Microsoft Windows® Explorer to browse to the E:\Labfiles\Starter\VB folder, and then double-click the VB.sln file. If you want to complete this lab by using Visual C#: <ul style="list-style-type: none"> Use Windows Explorer to browse to the E:\Labfiles\Starter\CS folder, and then double-click the CS.sln file.
3. Define the Survey.aspx page layout by using HTML controls in Design view.	<ol style="list-style-type: none"> If you are working with Visual Basic, in Solution Explorer, expand the E:\...\VB project. If you are working with Visual C#, in Solution Explorer, expand the E:\...\CS project. Display Survey.aspx in Design view. Drag a Div control from the HTML group in the Toolbox to the empty cell in the table in the middle of the Web page. Ensure that the Div control you have just added to the page is selected, and then click the Style property in the Properties window. Click the ellipsis (...) button in the Style property, and then use the Style Builder to set the following properties: <ul style="list-style-type: none"> Font Family: Impact (Hint: Use the Font tab) Font Color: #ffffff Font Style: Italic Font Size: 36pt Background Color: Maroon (Hint: Use the Background tab) Width: 100% (Hint: Use the Position tab)


(continued)

Tasks	Supporting information
3. <i>(continued)</i>	<p>g. Click the Div object in the Page Designer so that the insertion point is active in the object, and then type Survey</p> <p> See the following resources in the Resource Toolkit:</p> <ul style="list-style-type: none"> ▪ “HTML Controls for ASP.NET Web Pages” ▪ “How to: Add HTML Markup to Web Pages”
4. Define the Survey.aspx page layout by modifying HTML markup in Source view.	<p>a. Switch to Source view.</p> <p>b. Directly below the HTML for the Div element you have just added, type the HTML markup for a table that has the following characteristics:</p> <ul style="list-style-type: none"> • Style of Color : #0E0E6C; Width: 100%. • Contains six rows. • The first row has one cell. • The other five rows each have two cells. <p>c. Set the attributes of the cell in the first row of the table to:</p> <pre>align="left" valign="top" colspan="2" style="background-color:Maroon; color:white"</pre> <p> Note As you start typing the attributes of HTML element and the tags for sub elements, the Visual Studio IntelliSense features provide guidance to help you.</p> <p>d. In the first cell of the table, add the HTML markup for a SPAN element and HTML text from the following file:</p> <ul style="list-style-type: none"> • E:\Labfiles\Starter\StarterText\Ex1-Step4-Span.txt <p>e. For the first cell in each of the other rows in the table, set the align attribute to right and the valign attribute to top.</p> <p> Tip You can either add the attributes by typing the HTML markup in Source view, or you can position the insertion point inside the opening tag for an element, and then use the Properties window to set the properties.</p> <p>f. For the second cell in each of the rows in the table (apart from the first row), set the valign attribute to top.</p> <p>g. Type the text Name: in the first cell in the second row of the table.</p> <p>h. Type the text Gender: in the first cell in the third row of the table.</p> <p>i. Type the text Mountain Bike Rider? in the first cell in the fourth row of the table.</p> <p>j. Type the text Road Rider? in the first cell in the fifth row of the table.</p>

(continued)

Tasks	Supporting information
4. <i>(continued)</i>	<p>k. Type the text Favorite Trails from around the world: in the first cell in the sixth row of the table.</p> <p> See the following resources in the Resource Toolkit:</p> <ul style="list-style-type: none"> “HTML Controls for ASP.NET Web Pages” “How to: Add HTML Markup to Web Pages”
5. Add HTML Input controls to the Web page.	<p> Tip Use either Source view or Design view to complete the following steps, depending on which approach you find most useful.</p> <p>a. Add an HTML Text Input control to the second cell in the second row of the table. Specify the following attributes and values:</p> <ul style="list-style-type: none"> id="txtName" type="text" name="txtName" runat="server" enableviewstate="true" <p> Note Although you can use the Properties window to set most attributes for HTML controls in either Source view or Design view, you cannot use the Properties window to set the runat attribute for controls in Design view. Instead, you can use any of the following techniques:</p> <ul style="list-style-type: none"> Edit the HTML markup in Source view. Right-click the control in Design view, and then click Run as Server Control from the shortcut menu. <p>b. Add the markup for a radio element from the following file to the second cell in the third row of the table:</p> <ul style="list-style-type: none"> E:\Labfiles\Starter\StarterText\Ex1-Step5-Radio.txt <p>c. Add the markup for a checkbox element from the following file to the second cell in the fourth row of the table:</p> <ul style="list-style-type: none"> E:\Labfiles\Starter\StarterText\Ex1-Step5-MBRCheckBox.txt <p>d. Add the markup for a checkbox element from the following file to the second cell in the fifth row of the table:</p> <ul style="list-style-type: none"> E:\Labfiles\Starter\StarterText\Ex1-Step5-RRCheckBox.txt <p>e. Add the markup for a TextArea element from the following file to the second cell in the sixth row of the table:</p> <ul style="list-style-type: none"> E:\Labfiles\Starter\StarterText\Ex1-Step5-TextArea.txt <p>f. Add a Horizontal Rule (<hr/>) element directly below the HTML table.</p>

(continued)



Tasks	Supporting information
5. <i>(continued)</i>	<p data-bbox="656 338 1409 401">g. If you are currently working in Source view, switch to Design view to review the page.</p> <div data-bbox="641 426 699 485"></div> <p data-bbox="727 422 1256 453">See the following resources in the Resource Toolkit:</p> <ul data-bbox="732 464 1386 611" style="list-style-type: none"><li data-bbox="732 464 1386 527">▪ “How to: Add HTML Server Controls to a Web Forms Page by Using the Web Forms Designer”<li data-bbox="732 537 1224 569">▪ “HTML Controls for ASP.NET Web Pages”<li data-bbox="732 579 1240 611">▪ “How to: Add HTML Markup to Web Pages”

Exercise 2


Building Graphical User Interfaces with Web Server Controls

In this exercise, you will add ASP.NET Web server controls to the online survey form you used in Exercise 1. These controls will provide additional layout features and additional rich graphical user interface elements.



Scenario

Tasks	Supporting information
1. Review the proposed design of the Survey.aspx page.	 See the resource in the Resource Toolkit, “Survey.aspx Page Design–Web Server Controls.”
2. Refine the Survey.aspx page layout by using an ASP.NET Table Web server control in Design view.	<ol style="list-style-type: none"> Display the Survey.aspx Web page in Design view. Drag an ASP.NET Table Web server control from the Standard group in the Toolbox to the area below the horizontal rule control you added in Exercise 1. Use the Properties window to modify the following properties of the Table control: <ul style="list-style-type: none"> Set the ID property to tblOuter. Set the ForeColor property to #0E0E6C. Set the Width property to 100%. Click the Rows property in the Properties window, and then click the ellipsis button (...). Click Add three times to add three rows to the table. Click the first row you have just added, and then use the TableRow properties list to set its ID property to outerR1. Click the ellipsis button (...) in the Cells property for the first row. Click Add to add a cell. Set the ID property of the new cell to outerR1C1, and then set the ColumnSpan property to 2. Click OK to accept the settings in the TableCell Collection Editor dialog box.  See the following resources in the Resource Toolkit: <ul style="list-style-type: none"> “Introduction to ASP.NET Web Server Controls” “Standard Toolbox Controls” “How to: Add Web Server Controls to a Web Forms Page by Using the Web Forms Designer” “How to: Create a Simple Web Page by Using Visual Web Developer” “How to: Set ASP.NET Control Properties”




(continued)

Tasks	Supporting information
<p>3. Modify the properties of the Table Web server control.</p>	<ol style="list-style-type: none"> a. In the TableRow Collection Editor, set the ID properties of the second and third rows of the table to outerR2 and outerR3, respectively. b. Add two cells to the second row. c. Set the following properties for the first cell in the second row: <ul style="list-style-type: none"> • Text: Rides: • VerticalAlign: Middle • Width: 1% • Wrap: False • ID: outerR2C1 d. Set the ID property of the second cell in the second row to outerR2C2. e. Add two cells to the third row. f. Set the following properties for the first cell in the third row: <ul style="list-style-type: none"> • Text: Your Ability: • HorizontalAlign: Left • VerticalAlign: Top • Width: 1% • Wrap: False • ID: outerR3C1 g. Set the ID property of the second cell in the third row to outerR3C2. <p> See the following resources in the Resource Toolkit:</p> <ul style="list-style-type: none"> ▪ “Introduction to ASP.NET Web Server Controls” ▪ “Standard Toolbox Controls” ▪ “How to: Add Web Server Controls to a Web Forms Page by Using the Web Forms Designer” ▪ “How to: Create a Simple Web Page by Using Visual Web Developer” ▪ “How to: Set ASP.NET Control Properties”



(continued)

Tasks	Supporting information
<p>4. Add rows to the Table Web Server control by using Source view.</p>	<p>a. Switch to Source view.</p> <p>b. Select the markup code for the asp:TableRow element with the ID of outerR3.</p> <p> Important Select the entire element including its closing tag and both asp:TableCell sub elements.</p> <p>c. Copy the select text to the clipboard, and then position the cursor after the text you have just copied, but before the closing asp:Table tag.</p> <p>d. Paste the contents of the clipboard five times to create five new rows.</p> <p>e. Modify the ID properties of each new row so that it follows the naming convention used for the first three rows. For example, the fourth row should have an ID of outerR4, the fifth row should have an ID of outerR5, and so on.</p> <p>f. Modify the ID properties of each new cell so that it follows the naming convention used for the first three rows. For example, the first cell in the fourth row should have an ID of outerR4C1.</p> <p>g. Modify the textual content of each cell so that it matches the following:</p> <ul style="list-style-type: none"> • outerR4C1: Your Experience: • outerR5C1: Your Goals: • outerR6C1: Contact: • outerR7C1: Where do you normally ride? <p>h. Delete the textual content of the outerR8C1 cell.</p> <p> See the following resources in the Resource Toolkit:</p> <ul style="list-style-type: none"> ▪ “Introduction to ASP.NET Web Server Controls” ▪ “Standard Toolbox Controls” ▪ “How to: Add Web Server Controls to a Web Forms Page by Using the Web Forms Designer” ▪ “How to: Create a Simple Web Page by Using Visual Web Developer” ▪ “How to: Set ASP.NET Control Properties”





(continued)

Tasks	Supporting information
<p>5. Add a nested Table Web server control in Source view.</p>	<p> Note You cannot drag and drop controls into the cells of a Table Web server control in Design view. You must complete the following steps in Source view.</p> <p>a. Add the markup for an asp:Table from the following file so that it is between the start and end tags for the outerR2C2 asp:TableCell element:</p> <ul style="list-style-type: none"> E:\Labfiles\Starter\StarterText\Ex2-Step5-ASPTable.txt <p> See the following resources in the Resource Toolkit:</p> <ul style="list-style-type: none"> “Introduction to ASP.NET Web Server Controls” “Standard Toolbox Controls” “How to: Add Web Server Controls to a Web Forms Page by Using the Web Forms Designer” “How to: Create a Simple Web Page by Using Visual Web Developer” “How to: Set ASP.NET Control Properties” <p>b. Switch to Design view to review the page design.</p>
<p>6. Add Web server controls to the Table control.</p>	<p>a. Switch back to Source view.</p> <p> Note You cannot drag controls into the cells of a Table Web server control in Design view. You must complete the following steps in Source view.</p> <p>b. Drag a Panel control from the Standard group in the Toolbox to the area between the start and end tags of the outerR1C1 asp:TableCell element. With the HTML markup for the Panel control selected, use the Properties window to set the following properties:</p> <ul style="list-style-type: none"> ID: pnlActivity BackColor: Maroon ForeColor: White HorizontalAlign: Left Width: 100% Clear the Height attribute



(continued)

Tasks	Supporting information
6. <i>(continued)</i>	<p>c. Type the text Recent and Planned Activity between the start and end tags for the Panel control.</p> <p>d. Drag a Literal control from the Standard group in the Toolbox to the area between the start and end tags of the innerR1C1 asp:TableCell element. With the HTML markup for the Literal control selected, set the following properties in the Properties window:</p> <ul style="list-style-type: none"> • ID: ltrlLast • Text: Last Trip>>> <p>e. Add a Calendar control to the innerR1C2 asp:TableCell element, and then set the following properties for the Calendar control:</p> <ul style="list-style-type: none"> • ID: callLast • TodayDayStyle-BackColor: Maroon <p>f. Add the markup for a Literal control from the following file to the innerR2C1 asp:TableCell element:</p> <ul style="list-style-type: none"> • E:\Labfiles\Starter\StarterText\Ex2-Step6-Literal.txt <p>g. Add the markup for a Calendar control from the following file to the innerR2C2 asp:TableCell element:</p> <ul style="list-style-type: none"> • E:\Labfiles\Starter\StarterText\Ex2-Step6-Calendar.txt <p>h. Add a DropDownList control to the outerR3C2 asp:TableCell element, and then set the ID property of the DropDownList to ddAbility. Add the following text between the start and end tags of the ddAbility DropDownList control:</p> <pre><asp:ListItem Text="Beginner"></asp:ListItem> <asp:ListItem Text="Competent"></asp:ListItem> <asp:ListItem Text="Expert"></asp:ListItem></pre> <p> Note These ListItem elements provide the items for the DropDownList.</p> <p>i. Add the markup for a ListBox control from the following file to the outerR4C2 asp:TableCell element:</p> <ul style="list-style-type: none"> • E:\Labfiles\Starter\StarterText\Ex2-Step6-List.txt <p> Note The ListItem elements you have just added provide the items for the ListBox control.</p>



(continued)

Tasks	Supporting information
6. (continued)	<p>j. Add a CheckBoxList control to the outerR5C2 asp:TableCell element, and then set the ID property of the CheckBoxList to chkGoals. Add the following list items to the chkGoals CheckBoxList control:</p> <ul style="list-style-type: none"> • To get more road-riding experience • To get more mountain-biking experience • To upgrade my bike • To get the latest accessories <p> Note These ListItem elements provide the individual check box items for the CheckBoxList.</p> <p>k. Add the markup for a RadioButtonList control from the following file to the outerR6C2 asp:TableCell element:</p> <ul style="list-style-type: none"> • E:\Labfiles\Starter\StarterText\Ex2-Step6-RadioList.txt <p> Note The ListItem elements provide the individual radio button items for the RadioButtonList. Note also that the first radio option in the list is selected by default.</p> <p>l. Add an ImageMap control to the outerR7C2 asp:TableCell element, and then set the following properties for the ImageMap control:</p> <ul style="list-style-type: none"> • ID: imgMap • ImageUrl: ~/Images/BikeTheWorld.gif • HotSpotMode: PostBack <p> Note The HotSpotMode property controls the action to take when the user clicks a HotSpot in the ImageMap control. In this case, you have specified that a postback event will occur. You will define the HotSpot items later in this exercise.</p> <p>m. Add the following attribute to the start tag for the imgMap asp:ImageMap element:</p> <pre>OnClick = "imgMap_Click"</pre> <p> Note This property specifies the event-handler procedure to be used when a user clicks a HotSpot in the ImageMap. You will write the code for this event later in this lab, and you will define the HotSpot later in this exercise.</p> <p>n. Switch to Design view to review the current page design. Review the image displayed in the ImageMap control; you will create clickable areas on this image that cover the shapes of North America and Europe in the image.</p> <p>o. Switch back to Source view.</p>



(continued)

Tasks	Supporting information
6. <i>(continued)</i>	<p>p. Add the following text between the start and end tags of the imgMap asp:ImageMap control:</p> <pre><asp:PolygonHotSpot Coordinates="0,0,150,0,75,60,75,100,0,75" HotSpotMode="PostBack" PostBackValue="USA" TabIndex="1" /> <asp:PolygonHotSpot Coordinates="160,0,225,0,200,60,125,60,125,25" HotSpotMode="PostBack" PostBackValue="Europe" TabIndex="2" /></pre> <p> Note An ImageMap control supports three different types of HotSpot areas: RectangleHotSpot, CircleHotSpot, and PolygonHotSpot. You have just added two PolygonHotSpots, each with five sides.</p> <p>The shape of a PolygonHotSpot is defined by pairs of comma-separated x and y coordinates that relate to the underlying image used in the ImageMap control. The coordinates define the outline of the polygon shapes by representing where each side of the polygon begins.</p> <p>The two polygons you have added cover North America and Europe of the image used in the ImageMap control.</p> <p>q. Add a HiddenField control, and then drop it <i>after</i> the closing tag of the imgMap ImageMap control. Set the ID property of the HiddenField control to hdnRegion.</p> <p>r. Add an ImageButton control to the outerR8C2 asp:TableCell element, and then set the following properties of the ImageButton control:</p> <ul style="list-style-type: none"> • ID: ibtnSubmit • ImageUrl: ~/Images/Submit.GIF • PostBackUrl: ~/SurveyReceipt.aspx <p>s. Switch to Design view, and then review the Survey.aspx Web page design.</p> <p> See the following resources in the Resource Toolkit:</p> <ul style="list-style-type: none"> ▪ “Introduction to ASP.NET Web Server Controls” ▪ “Standard Toolbox Controls” ▪ “How to: Add Web Server Controls to a Web Forms Page by Using the Web Forms Designer” ▪ “How to: Create a Simple Web Page by Using Visual Web Developer” ▪ “How to: Set ASP.NET Control Properties”

(continued)

Tasks	Supporting information
<p>7. Add site navigation functionality by using Web server controls.</p>	<ol style="list-style-type: none"> a. On the Website menu, click Add Existing Item. b. In the Add Existing Item dialog box, browse to the E:\Labfiles\Starter folder, locate the Web.sitemap file, and then add the file to your project. c. Double-click Web.sitemap in Solution Explorer to open it. Review the contents of the file. <p> Note The file contains XML that defines a hierarchy of Web pages used in the Web application. Note that the hierarchy does not depend on the physical locations of the files referenced in the Web.sitemap. You can use a Web.sitemap file to define a virtual hierarchy and to bind navigation controls to that hierarchy.</p> <ol style="list-style-type: none"> d. Return to the Survey.aspx page in Design view. e. Drag a SiteMapPath control from the Navigation group in the Toolbox to the empty line below the Adventure Works logo on the right side of the page. f. Set the ForeColor property of the SiteMapPath control to #0E0E6C. g. Switch to Source view, and then add the following text between the start and end tags of the SiteMapPath1 asp:SiteMapPath control: <pre><NodeStyle ForeColor="#0E0E6C" /></pre> h. Display the Default.aspx page in Design view. i. Drag a Menu control from the Navigation group in the Toolbox to the empty table row below the Adventure Works image at the top of the page. A Smart Tag menu appears. j. On the Smart Tag menu, click New Data Source in the Choose Data Source list. The Data Source Configuration Wizard appears. k. Click Site Map in the Choose a Data Source page, and then click OK. l. Set the following properties of the Menu control in the Properties window: <ul style="list-style-type: none"> • BackColor: #0E0E6C • DynamicHoverStyle-BackColor: #0E0E6C • DynamicMenuItemStyle-BackColor: #0E0E6C • ForeColor: White <p> See the following resources in the Resource Toolkit:</p> <ul style="list-style-type: none"> ▪ “Introduction to ASP.NET Web Server Controls” ▪ “Navigation Controls” ▪ “How to: Add Web Server Controls to a Web Forms Page by Using the Web Forms Designer” ▪ “How to: Create a Simple Web Page by Using Visual Web Developer” ▪ “How to: Set ASP.NET Control Properties”

(continued)

Tasks	Supporting information
<p>8. Incorporate advertising functionality with Web server controls.</p>	<ol style="list-style-type: none"> a. On the Website menu, click Add Existing Item. b. In the Add Existing Item dialog box, browse to the E:\Labfiles\Starter folder, locate the AdSchedule.xml file, and then add the file to your project. c. Double-click AdSchedule.xml in Solution Explorer to open it. Review the contents of the file. <p> Note The file contains XML that defines a schedule for displaying images. AdRotator controls can be bound to this file and will display the images according to the schedule information in this file. The schedule in the file specifies that the BikeTheWorld.gif image will be displayed three times more frequently than the WaterBottle.gif image, because their weightings specified by the Impressions element are 75 and 25, respectively.</p> <p>As its name implies, the AdRotator is usually used to control the schedule for banner advertisements on Web sites. However, the Adventure Works Web application simply uses the scheduling functionality to provide dynamic image displays on the Home page.</p> <ol style="list-style-type: none"> d. Switch back to Default.aspx. e. Drag an AdRotator control from the Standard group in the Toolbox to the empty space in the table below the Biking Services section, on the right side of the Default.aspx Web page. f. Using the Properties window, set the AdvertisementFile property to ~/AdSchedule.xml. <p> See the following resources in the Resource Toolkit:</p> <ul style="list-style-type: none"> ▪ “Introduction to ASP.NET Web Server Controls” ▪ “Standard Toolbox Controls” ▪ “How to: Add Web Server Controls to a Web Forms Page by Using the Web Forms Designer” ▪ “How to: Create a Simple Web Page by Using Visual Web Developer” ▪ “How to: Set ASP.NET Control Properties”


Exercise 3

Programming Web Server Controls and Working with Postbacks




In this exercise, you will write code for the `FeedbackForm.aspx` page to manipulate the properties of controls on that page so that an appropriate response is sent to the browser in specific situations. For example, you will set the **Text** property of **Label** controls to provide information to users. You will also write code that determines whether the `FeedBackForm.aspx` page was loaded in response to a postback action.

You will also write code that retrieves survey information from the `Survey.aspx` page when a user submits an online survey. This will involve a cross-page postback scenario.


Scenario

Tasks	Supporting information
<ol style="list-style-type: none"> 1. Manage the Click event for the ImageMap object. 	<ol style="list-style-type: none"> a. Return to the Survey.aspx page. b. On the View menu, click Code. c. Add a protected method named imgMap_Click to the Survey class. If you are working with Visual Basic, the method should be written as a Sub procedure. If you are working with Visual C#, the method should return the void type. In addition, the procedure must accept two parameters: <ul style="list-style-type: none"> • The first parameter must be of type object (for Visual C#) or Object (for Visual Basic). Name this parameter sender. • The second parameter must be of type ImageMapEventArgs. Name this parameter e. d. Add a statement to the method that sets the Value property of the hdnRegion control to the PostBackValue property of the ImageMapEventArgs parameter. <div style="margin-top: 10px;">  Note This method handles the Click event of the ImageMap control you added to the Web page in Exercise 2. </div>


(continued)

Tasks	Supporting information
<p>2. Create the OnClientClick property and the server-side Click event for the Submit button.</p>	<p>a. Display the Web page FeedbackForm.aspx form in Design view.</p> <p>b. Click the Submit button, and then in the Properties window, set the button's OnClientClick property to the following JavaScript code:</p> <pre>return confirm('Are you sure you want to submit this survey?')</pre> <p> Note The code you have just written runs in the Web browser. It enables users to decide whether they are ready to submit their feedback without invoking a round-trip to the server.</p> <p> See the resource in the Resource Toolkit, “The Button.OnClientClick Property.”</p> <p>c. Double-click the Submit button to view its default server-side event-handler.</p> <p>d. Add code to the event handler to set the Text property of the lblTo label object to a concatenation of:</p> <ul style="list-style-type: none"> • Thank-you for your feedback. Your comment<code>
</code> • The Text property of TextBox1 • <code></code><code>
</code> has been noted. <p>e. Add code to the event handler to perform the following actions:</p> <ul style="list-style-type: none"> • Set the Visible property of TextBox1 to false. • Set the Visible property of Label8 to false. • Set the Visible property of Button1 to false. <p> Note The code you have just written manipulates the properties of controls so that they are hidden when the user has submitted feedback.</p>


(continued)

Tasks	Supporting information
<p>3. Determine if an ASP.NET Web page was invoked as part of the postback process.</p>	<p>a. Return to the Design view of the FeedbackForm.aspx page, and then double-click the page background. Ensure that you double-click outside the table. Visual Studio 2005 creates an event-handling method for the Load event of the page.</p> <p>b. Add a decision-making statement that examines the Page.IsPostBack property of the Web page. Add code that performs the following actions if the Page.IsPostBack property is <i>not</i> true:</p> <ul style="list-style-type: none"> • Sets the Text property of the lblTo Label control to Adventure Works: General Feedback if the Department member of the Page.Request.QueryString collection has not been populated • Sets the Text property of the lblTo Label control to Adventure Works: Web Site Feedback if the uppercase version of the Department member of the Page.Request.QueryString collection is equal to the string "WEB" • Sets the Text property of the lblTo Label control to Adventure Works: Product Feedback if the uppercase version of the Department member of the Page.Request.QueryString collection is equal to the string "PRODUCTS" • Sets the Text property of the lblTo Label control to Adventure Works: General Feedback for all other cases <p> See the resource in the Resource Toolkit, “How to: Determine How an ASP.NET Web Page Was Invoked.”</p>
<p>4. Create a page that can receive and process information submitted as part of a cross-page postback.</p>	<p>a. Display the SurveyReceipt.aspx Web page in Design view.</p> <p>b. Generate the default Page_Load event handler for this Web page.</p> <p>c. In the Page_Load method, write code that calls the PreviousPage.FindControl method, passing in the literal string txtName as a parameter.</p> <p>d. Add code that converts the return value of this method call to an HtmlInputText object.</p> <p>e. Add code that declares a variable named txtName as an HtmlInputText object and sets it to the code you have already added.</p>

(continued)

Tasks	Supporting information
4. <i>(continued)</i>	<div data-bbox="704 338 764 394"></div> <p>Note If you are working with Visual Basic, the following sample code achieves these tasks:</p> <pre>Dim txtName As HtmlInputText = _ CType(PreviousPage.FindControl("txtName"), _ HtmlInputText)</pre> <p>If you are working with Visual C#, the following sample code achieves these tasks:</p> <pre>HtmlInputText txtName = (HtmlInputText)PreviousPage.FindControl ("txtName");</pre> <p>f. If you are working with Visual Basic, add the code from the following file to declare 10 more variables directly beneath the code you have just added:</p> <ul style="list-style-type: none"> E:\Labfiles\Starter\StarterText\Ex3-Step4-VB_Variables.txt <p>g. If you are working with Visual C#, add the code from the following file to declare 10 more variables directly beneath the code you have just added:</p> <ul style="list-style-type: none"> E:\Labfiles\Starter\StarterText\Ex3-Step4-CS_Variables.txt <p>h. Declare a string variable named sName, and then set it to txtName.Value.ToString().</p> <p>i. If you are working with Visual Basic, declare a string variable named sGender, and then set it to Request.Form("optGender").</p> <p>j. If you are working with Visual C#, declare a string variable named sGender, and then set it to Request.Form["optGender"].</p> <p>k. Declare a Boolean variable named bMBR, and then set it to chkMBR.Checked.</p> <p>l. Declare a Boolean variable named bRR, and then set it to chkRR.Checked.</p> <p>m. Declare a string variable named sTrails, and then set it to txtTrails.Value.</p> <p>n. Declare a string variable named sLast, and then set it to calLast.SelectedDate.ToLongDateString().</p> <p>o. Declare a string variable named sNext, and then set it to calNext.SelectedDate.ToLongDateString().</p> <p>p. Declare a string variable named sAbility, and then set it to ddAbility.SelectedValue.ToString().</p> <p>q. Declare a string variable named sExperience, and then set it to lstExperience.SelectedValue.ToString().</p> <p>r. Declare a string variable named sGoals, and then set it to "".</p>

(continued)

Tasks	Supporting information
4. <i>(continued)</i>	<p>s. Construct a loop that iterates through each ListItem object in chkGoals.Items. In the loop, create a decision-making structure to check if the Selected property of the ListItem object is true. If so, your code should concatenate the Text property of the ListItem object with the literal string <code>
</code>, and append the concatenated string to the sGoals variable.</p> <p>t. Declare a string variable named sMarketing, and then set it to optMarketing.Selected.Value.ToString().</p> <p>u. Declare a string variable named sRegion, and then set it to hdnRegion.Value.</p> <p>v. If you are working with Visual Basic, add the code from the following file to set the Text property of the lblSurveyReceipt label object to a concatenation of literal text and variables:</p> <ul style="list-style-type: none"> • E:\Labfiles\Starter\StarterText\Ex3-Step4-VB_Concat.txt <p>w. If you are working with Visual C#, add the code from the following file to set the Text property of the lblSurveyReceipt label object to a concatenation of literal text and variables:</p> <ul style="list-style-type: none"> • E:\Labfiles\Starter\StarterText\Ex3-Step4-CS_Concat.txt <p> See the following resources in the Resource Toolkit:</p> <ul style="list-style-type: none"> ▪ “Cross Page Posting in an ASP.NET Web Page” ▪ “How To: Post an ASP.NET Page to a Different Page” ▪ “How To: Determine How an ASP.NET Web Page was Invoked”

(continued)

Tasks	Supporting information
5. Test the Web application.	<ul style="list-style-type: none">a. Save all files.b. In Solution Explorer, right-click the Web project, and then click View in Browser from the shortcut menu to run the Web application. Microsoft Internet Explorer starts and displays the default page.c. Note the graphic type displayed on the right side of the page. Press F5 repeatedly to refresh the browser until the image changes. This dynamic image functionality is provided by a combination of the AdRotator Web server control that you added to the page, and the AdSchedule.xml file that contains details of the advertisements.d. At the top-left of the page, point to Home, point to Contact Us, and then click Complete an Online Survey. This menu structure is provided by a combination of the Menu Web server control for the user interface, and the Web.Sitemap file that contains the menu definition.e. When the Survey.aspx page has loaded, notice the navigation features below the Adventure Works logo. This navigation structure is provided by a combination of the SiteMapPath Web server control for the user interface, and the Web.Sitemap file that contains the menu definition. Click the Home link in the SiteMapPath navigation control.f. Navigate to the Survey.aspx page again. Notice the survey layout and the controls that it contains.

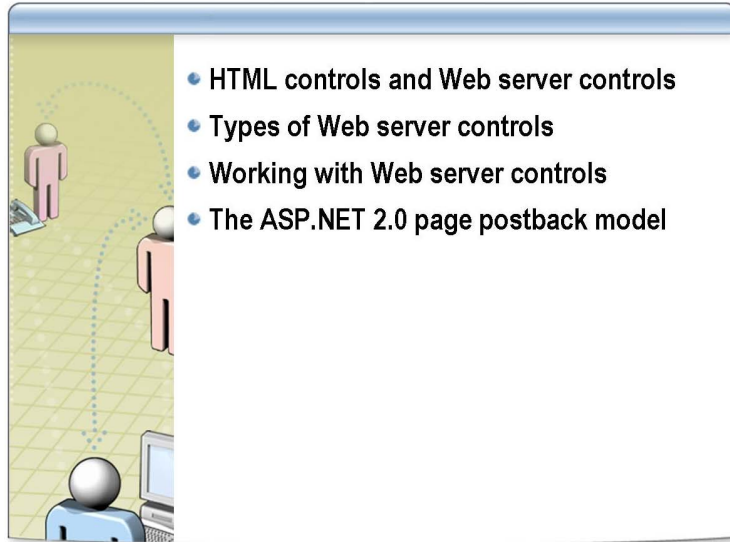
(continued)

Tasks	Supporting information
5. <i>(continued)</i>	<p>g. Enter the following information in the online survey:</p> <ul style="list-style-type: none"> Name: <i><Enter your name></i> Gender: <i><Click your gender ></i> Mountain Bike Rider? Checked Road Rider? Not checked Favorite Trails: North Seattle Trail, Giro Alps, Giro Napoli Last Trip: <i><Click yesterday's date></i> Next Trip: <i><Click tomorrow's date></i> Your Ability: Expert Your Experience: Raced down mountains To get more road-riding experience: Not checked To get more mountain-biking experience: Checked To upgrade my bike: Checked To get the latest accessories: Not checked Contact: Only send me sales information from Adventure Works Where do you normally ride?: <i><Click Europe></i> <p>h. Click Submit. The survey information is posted to the SurveyReceipt.aspx page, where it is rendered to provide feedback.</p> <p>i. Review the details of the submitted survey.</p> <p>j. In the list of links on the left side of the page, click Provide Feedback.</p> <p>k. Click Provide feedback about our Web site. When the FeedbackForm.aspx page loads, note that the label reads <i>Adventure Works: Web site feedback</i>. This occurs because the page loaded in response to a normal request, not as part of a page postback.</p> <p>l. Type feedback of Great Web Site!, and then click Submit.</p> <p>m. A client-side message box appears asking you to confirm that you want to submit the survey. Click OK. You defined this client side code for the OnClientClick event of the Submit button in Exercise 3. Note the information displayed on the Web page, and note also that the previously visible label, text box, and button are no longer visible. This occurs because the page was loaded as part of a postback, and you wrote code to hide the controls for this scenario.</p> <p>n. Close Internet Explorer.</p>

Lab Shutdown

After you complete the lab, you must shut down the **2543B-LON-DEV-01-03** virtual machine and discard any changes.

Lab Discussion



- HTML controls and Web server controls
- Types of Web server controls
- Working with Web server controls
- The ASP.NET 2.0 page postback model

In the lab, you:

- Created Web-based user interfaces with HTML controls and Web server controls.
- Wrote code that interacted with Web server controls.
- Wrote code that interacted with the postback model of ASP.NET 2.0.

The instructor will lead a group discussion of these tasks, and you should be prepared to contribute to the discussion, especially if you encountered problems completing the exercises.

