Chapter 23: Web App Development with ASP.NET in VB

Internet & World Wide Web
How to Program, 5/e

Note: This chapter is a copy of Chapter 13 of our book *Visual Basic 2010 How to Program*. For that reason, we simply copied the PowerPoint slides for this chapter and *did not* re-number them

OBJECTIVES

In this chapter you'll learn:

- Web application development using ASP.NET.
- To handle the events from a Web Form's controls.
- To use validation controls to ensure that data is in the correct format before it's sent from a client to the server.
- To maintain user-specific information.
- To create a data-driven web application using ASP.NET and LINQ to SQL.

13.1 Introduction

- In this chapter, we introduce web-application development with Microsoft's ASP.NET technology.
- Web-based applications create web content for web-browser clients.
- We present several examples that demonstrate web-application development using Web Forms, web controls (also called ASP.NET server controls) and Visual Basic programming.
- Web Form files have the file-name extension .aspx and contain the web page's GUI.
- You customize Web Forms by adding web controls including labels, textboxes, images, buttons and other GUI components.

13.1 Introduction

- The Web Form file represents the web page that is sent to the client browser.
- We often refer to Web Form files as ASPX files.
- An ASPX file created in Visual Studio has a corresponding class written in a .NET language—we use Visual Basic in this book.
- This class contains event handlers, initialization code, utility methods and other supporting code.
- The file that contains this class is called the code-behind file and provides the ASPX file's programmatic implementation.

13.1 Introduction

- To develop the code and GUIs in this chapter, we used Microsoft's Visual Web Developer 2010 Express—a free IDE designed for developing ASP.NET web applications.
- The full version of Visual Studio 2010 includes the functionality of Visual Web Developer, so the instructions we present for Visual Web Developer also apply to Visual Studio 2010.
- The database example (Section 13.8) also requires SQL Server 2008 Express.
- See the *Before You Begin* section of the book for additional information on this software.

13.1 Introduction

- In the online chapter, Web App Development: A Deeper Look, we present several additional web-application development topics, including:
  - master pages to maintain a uniform look-and-feel across the Web Forms in a web application
  - creating password-protected websites with registration and login capabilities
  - using the Web Site Administration Tool to specify which parts of a website are password protected
  - using ASP.NET AJAX to quickly and easily improve the user experience for your web applications, giving them responsiveness comparable to that of desktop applications.

13.2 Web Basics

- In this section, we discuss what occurs when a user requests a web page in a browser.
- In its simplest form, a web page is nothing more than an HTML (HyperText Markup Language) document (with the extension .html or .htm) that describes to a web browser the document's content and how to format it.
- HTML documents normally contain hyperlinks that link to different pages or to other parts of the same page.
- When the user clicks a hyperlink, a web server locates the requested web page and sends it to the user's web browser.

13.2 Web Basics

- Similarly, the user can type the address of a web page into the browser's address field and press Enter to view the specified page.
- Web development tools like Visual Web Developer typically use a "stricter" version of HTML called XHTML (Extensible HyperText Markup Language).
- ASP.NET produces web pages as XHTML documents.

13.2 Web Basics

- URIs and URLs
  - URIs (Uniform Resource Identifiers) identify resources on the Internet.
  - URIs that start with `http://` are called URLs (Uniform Resource Locators).
  - Common URLs refer to files, directories or server-side code that performs tasks such as database lookups, Internet searches and business application processing.
  - If you know the URL of a publicly available resource anywhere on the web, you can enter that URL into a web browser's address field and the browser can access that resource.

13.2 Web Basics

- Parts of a URL
  - A URL contains information that directs a browser to the resource that the user wishes to access.
  - Web servers make such resources available to web clients.
  - Popular web servers include Microsoft's Internet Information Services (IIS) and Apache's HTTP Server.
  - Let's examine the components of the URL

    ```
    http://www.deitel.com/books/downloads.html
    ```

  - The `http://` indicates that the HyperText Transfer Protocol (HTTP) should be used to obtain the resource.

13.2 Web Basics

- HTTP is the web protocol that enables clients and servers to communicate.
- Next in the URL is the server's fully qualified hostname (`www.deitel.com`)—the name of the web server computer on which the resource resides.
- This computer is referred to as the host, because it houses and maintains resources.
- The hostname `www.deitel.com` is translated into an IP (Internet Protocol) address—a numerical value that uniquely identifies the server on the Internet.

13.2 Web Basics

- A Domain Name System (DNS) server maintains a database of hostnames and their corresponding IP addresses, and performs the translations automatically.
- The remainder of the URL (`/books/downloads.html`) specifies the resource's location (`/books`) and name (`downloads.html`) on the web server.
- The location could represent an actual directory on the web server's file system.
- For security reasons, however, the location is typically a virtual directory.
- The web server translates the virtual directory into a real location on the server, thus hiding the resource's true location.

13.2 Web Basics

- Making a Request and Receiving a Response
  - When given a URL, a web browser uses HTTP to retrieve and display the web page found at that address.
  - Figure 13.1 shows a web browser sending a request to a web server.
  - Figure 13.2 shows the web server responding to that request.

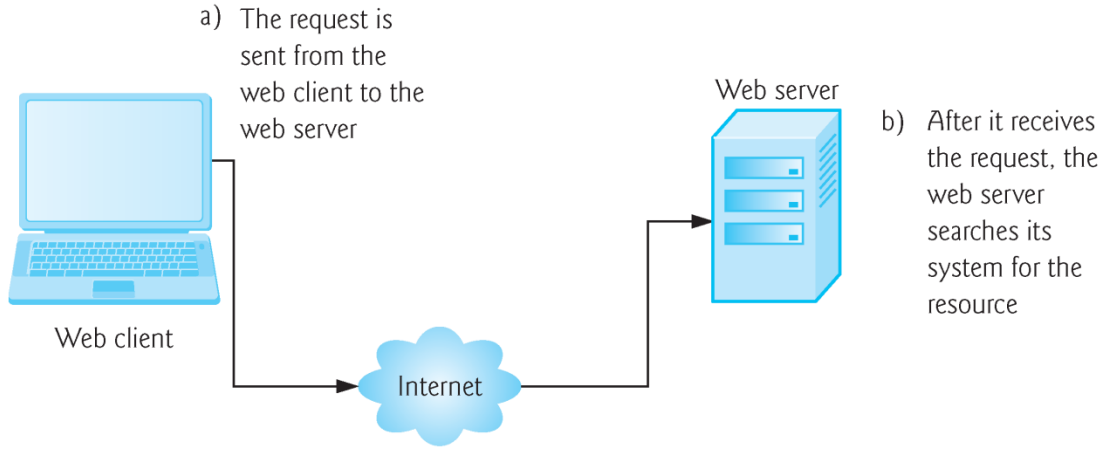

Fig. 13.1 | Client requesting a resource from a web server.

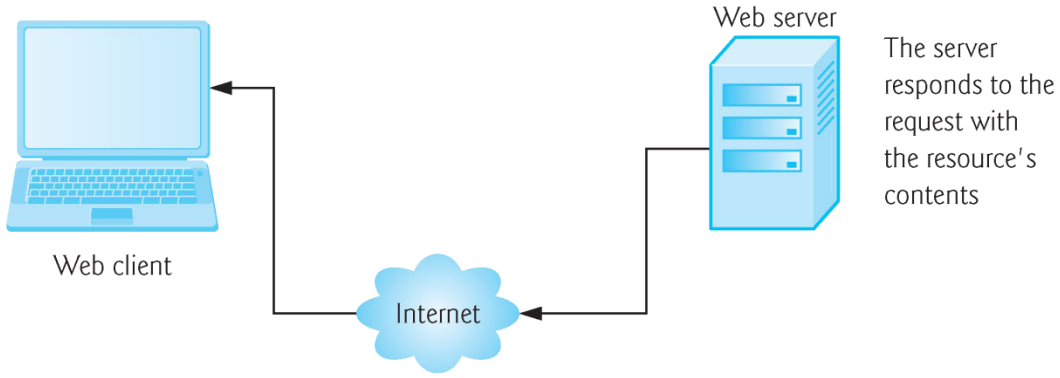

Fig. 13.2 | Client receiving a response from the web server.

13.3 Multitier Application Architecture

- Web-based applications are multitier applications (sometimes referred to as ***n-tier applications*).**
- Multitier applications divide functionality into separate tiers (that is, logical groupings of functionality).
- Although tiers can be located on the same computer, the tiers of web-based applications commonly reside on separate computers for security and scalability.
- Figure 13.3 presents the basic architecture of a three-tier web-based application.

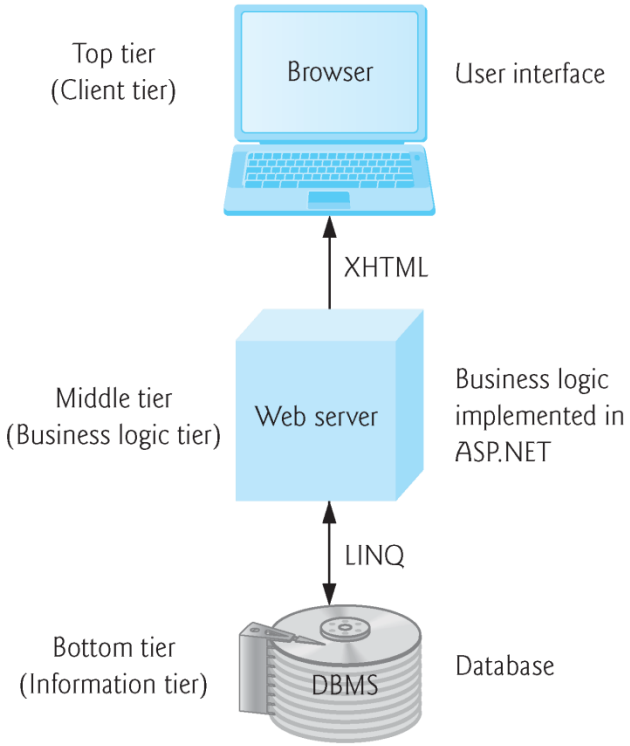

Fig. 13.3 | Three-tier architecture.

13.3 Multitier Application Architecture

- Information Tier
  - The information tier (also called the bottom tier) maintains the application's data.
  - This tier typically stores data in a relational database management system.
  - For example, a retail store might have a database for storing product information, such as descriptions, prices and quantities in stock.
  - The same database also might contain customer information, such as user names, billing addresses and credit card numbers.
  - This tier can contain multiple databases, which together comprise the data needed for an application.

13.3 Multitier Application Architecture

- Business Logic
  - The middle tier implements business logic, controller logic and presentation logic to control interactions between the application's clients and its data.
  - The middle tier acts as an intermediary between data in the information tier and the application's clients.
  - The middle-tier controller logic processes client requests (such as requests to view a product catalog) and retrieves data from the database.
  - The middle-tier presentation logic then processes data from the information tier and presents the content to the client.

13.3 Multitier Application Architecture

- Web applications typically present data to clients as web pages.
- Business logic in the middle tier enforces business rules and ensures that data is reliable before the server application updates the database or presents the data to users.
- Business rules dictate how clients can and cannot access application data, and how applications process data.
- For example, a business rule in the middle tier of a retail store's web-based application might ensure that all product quantities remain positive.
- A client request to set a negative quantity in the bottom tier's product information database would be rejected by the middle tier's business logic.

13.3 Multitier Application Architecture

- Client Tier
  - The client tier, or top tier, is the application's user interface, which gathers input and displays output.
  - Users interact directly with the application through the user interface (typically viewed in a web browser), keyboard and mouse.
  - In response to user actions (for example, clicking a hyperlink), the client tier interacts with the middle tier to make requests and to retrieve data from the information tier.
  - The client tier then displays to the user the data retrieved from the middle tier.
  - The client tier never directly interacts with the information tier.

13.4 Your First Web Application

- Our first example displays the web server's time of day in a browser window (Fig. 13.4).
- When this application executes—that is, a web browser requests the application's web page—the web server executes the application's code, which gets the current time and displays it in a `Label`.
- The web server then returns the result to the web browser that made the request, and the web browser renders the web page containing the time.
- We executed this application in both the Internet Explorer and Firefox web browsers to show you that the web page renders identically in each.

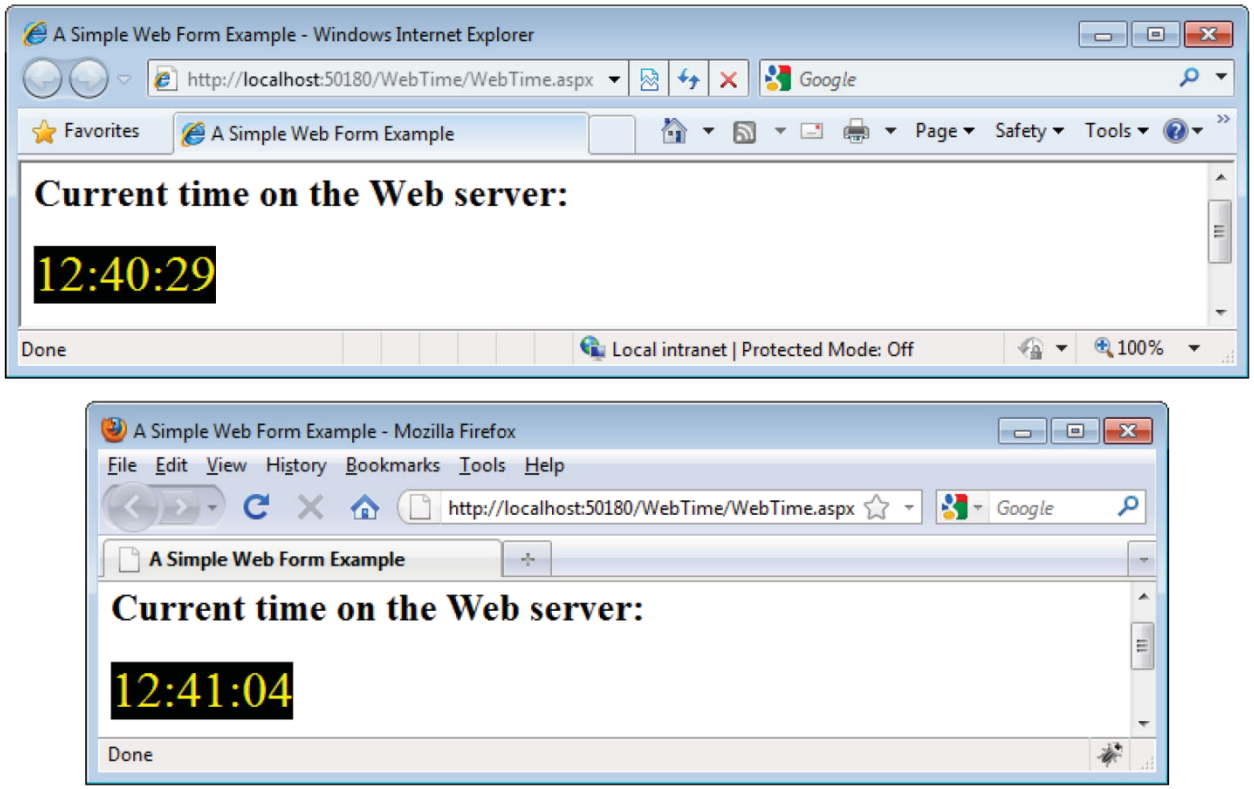

Fig. 13.4 | `WebTime` web application running in both Internet Explorer and Firefox.

13.4 Your First Web Application

- Testing the Application in Your Default Web Browser
- To test this application in your default web browser, perform the following steps:
  - Open Visual Web Developer.
  - Select Open Web Site… from the File menu.
  - In the Open Web Site dialog (Fig. 13.5), ensure that File System is selected, then navigate to this chapter's examples, select the `WebTime` folder and click the Open `Button`.
  - Select `WebTime.aspx` in the Solution Explorer, then type *Ctrl + F5* to execute the web application.

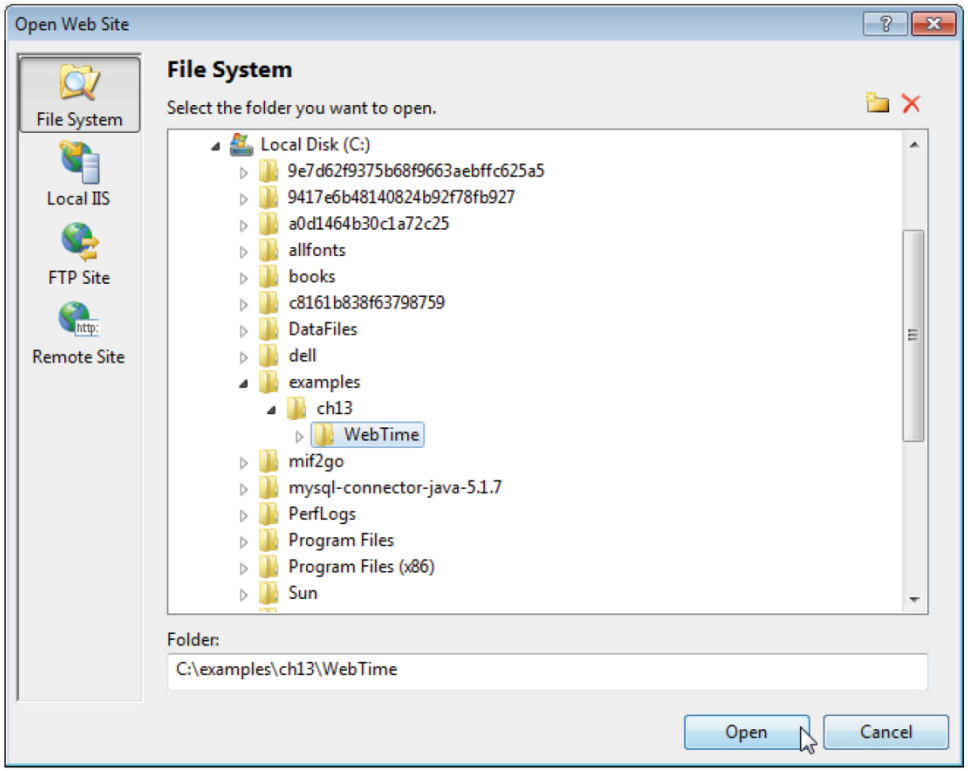

Fig. 13.5 | Open Web Site dialog.

13.4 Your First Web Application

- Testing the Application in a Selected Web Browser
  - If you wish to execute the application in another web browser, you can copy the web page's address from your default browser's address field and paste it into another browser's address field, or you can perform the following steps:
  - In the Solution Explorer, right click `WebTime.aspx` and select Browse With… to display the Browse With dialog (Fig. 13.6).
  - From the Browsers list, select the browser in which you'd like to test the web application and click the Browse `Button`.
- If the browser you wish to use is not listed, you can use the Browse With dialog to add items to or remove items from the list of web browsers.

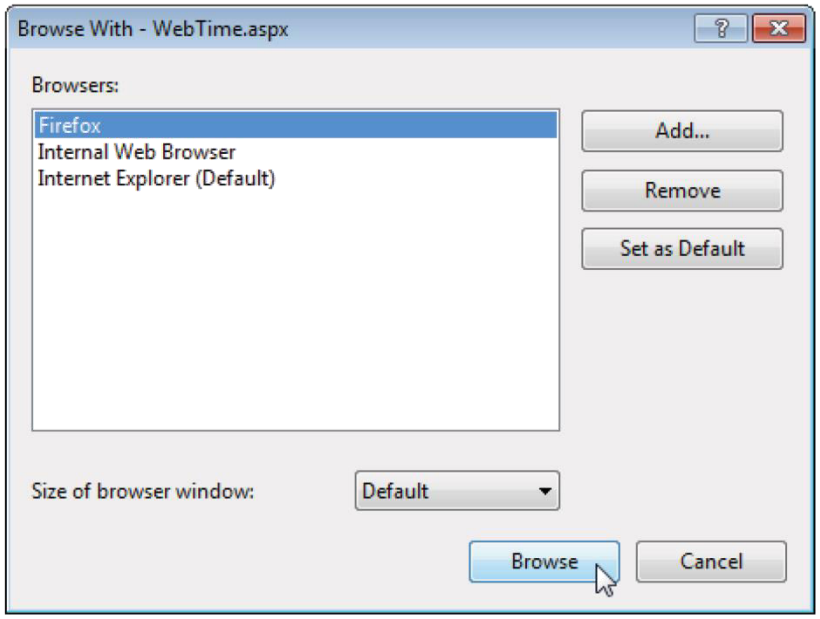

Fig. 13.6 | Selecting another web browser to execute the web application.

13.4.1 Building the `WebTime` Application

- Now that you've tested the application, let's create it in Visual Web Developer.
- Step 1: Creating the Web Site Project
  - Select File > New Web Site...
  - to display the New Web Site dialog (Fig. 13.7).
  - In the left column of this dialog, ensure that Visual Basic is selected, then select Empty Web Site in the middle column.
  - At the bottom of the dialog you can specify the location and name of the web application.

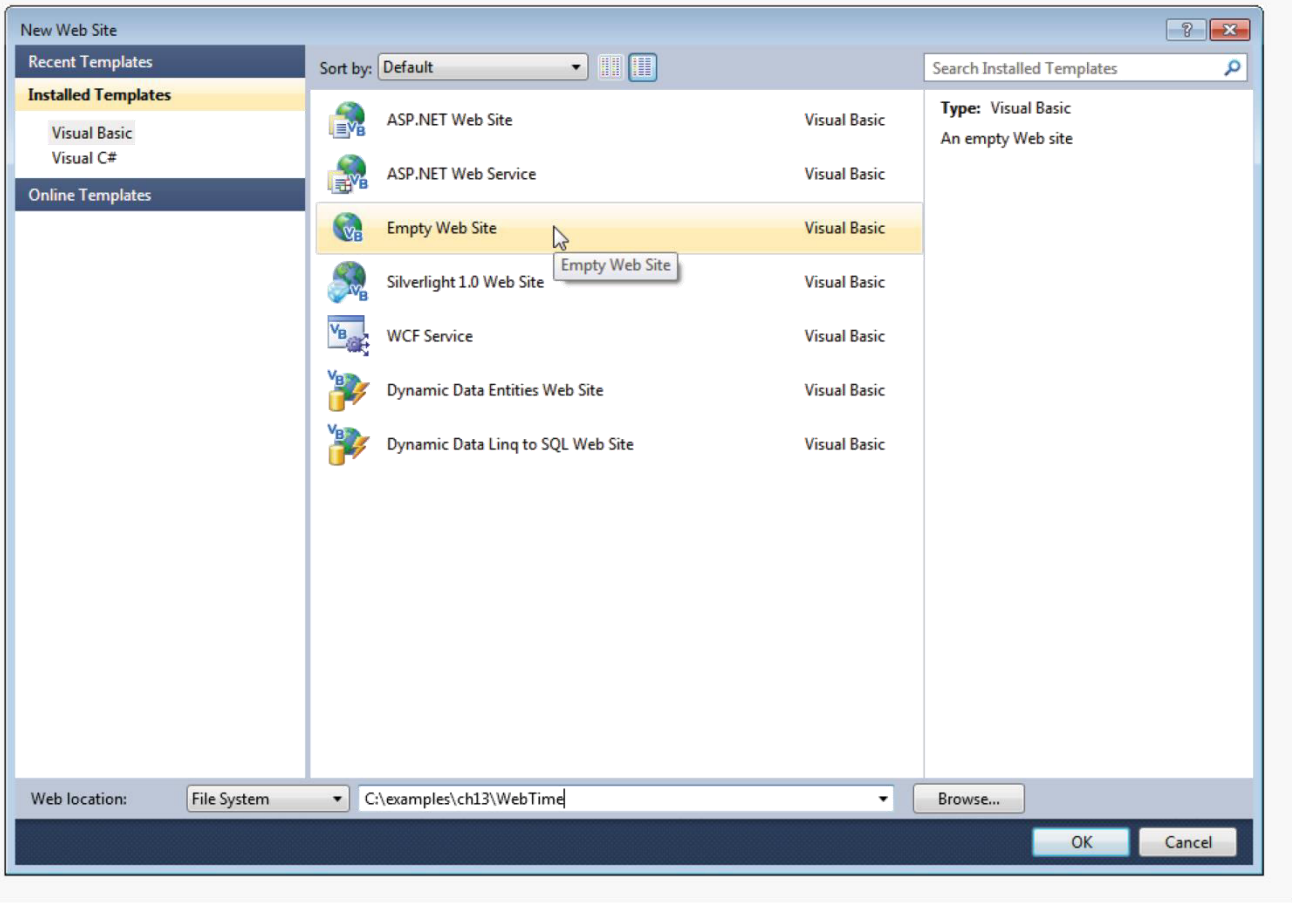

Fig. 13.7 | Creating an **ASP.NET Web Site** in Visual Web Developer

13.4.1 Building the `WebTime` Application

- The Web location: `ComboBox` provides the following options:
  - File System: Creates a new website for testing on your local computer. Such websites execute in Visual Web Developer's built-in ASP.NET Development Server and can be accessed only by web browsers running on the same computer. You can later "publish" your website to a production web server for access via a local network or the Internet. Each example in this chapter uses the File System option, so select it now.

13.4.1 Building the `WebTime` Application

- HTTP: Creates a new website on an IIS web server and uses HTTP to allow you to put your website's files on the server. IIS is Microsoft's software that is used to run production websites. If you own a website and have your own web server, you might use this to build a new website directly on that server computer. You must be an Administrator on the computer running IIS to use this option.
- FTP: Uses File Transfer Protocol (FTP) to allow you to put your website's files on the server. The server administrator must first create the website on the server for you. FTP is commonly used by so-called "hosting providers" to allow website owners to share a server computer that runs many websites.

- Change the name of the web application from `WebSite1` to `WebTime`, then click the OK `Button` to create the website.

13.4.1 Building the `WebTime` Application

- Step 2: Adding a Web Form to the Website and Examining the *Solution Explorer*
  - A Web Form represents one page in a web application—we'll often use the terms "page" and "Web Form" interchangeably.
  - A Web Form contains a web application's GUI.
- To create the `WebTime.aspx` Web Form:
  - Right click the project name in the Solution Explorer and select Add New Item... to display the Add New Item dialog (Fig. 13.8).
  - In the left column, ensure that Visual Basic is selected, then select Web Form in the middle column.
  - In the Name: `TextBox`, change the file name to `WebTime.aspx`, then click the Add `Button`.
- After you add the Web Form, the IDE opens it in Source view by default (Fig. 13.9).
- This view displays the markup for the Web Form.

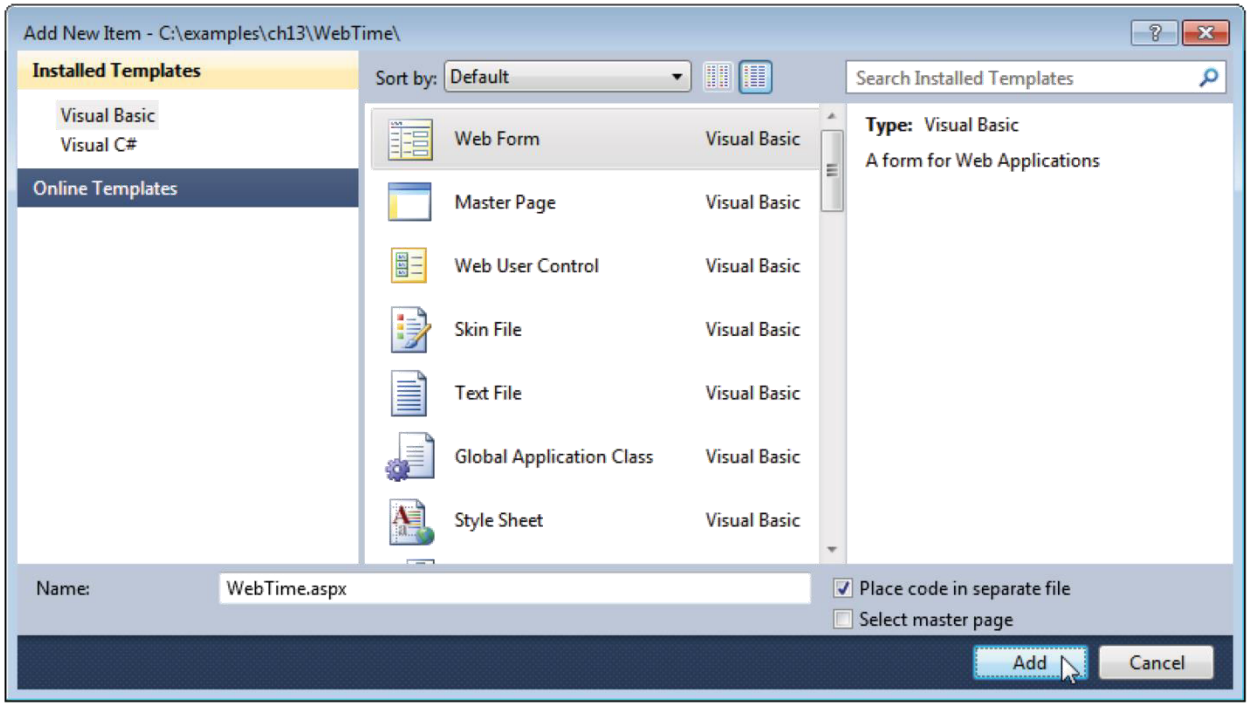

Fig. 13.8 | Adding a new **Web Form** to the website with the **Add New Item** dialog.

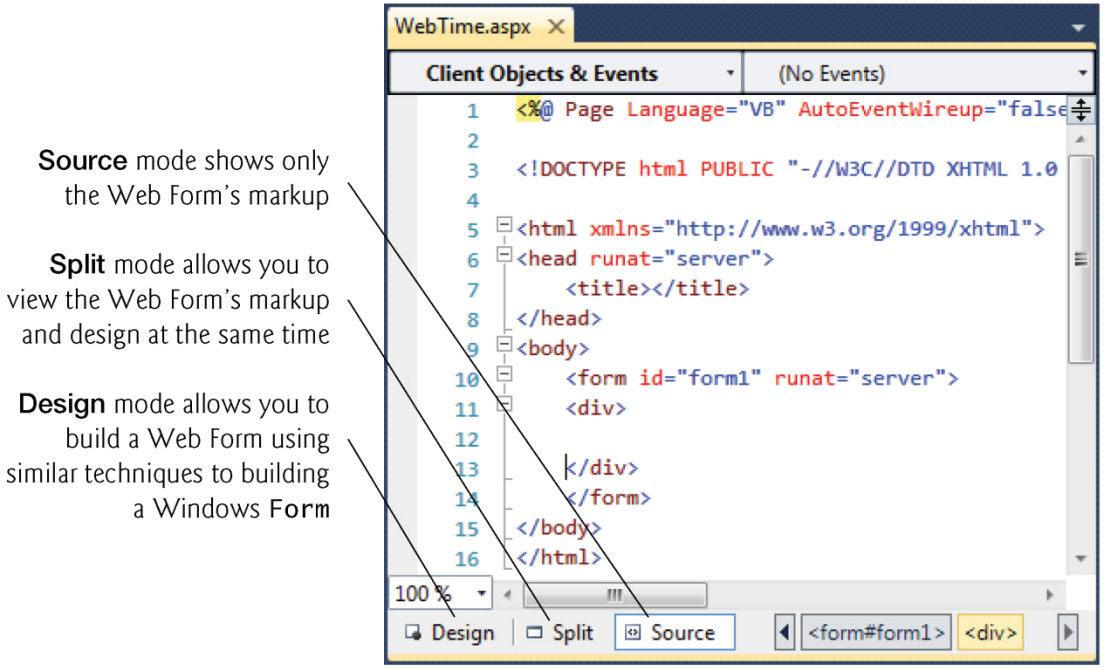

Fig. 13.9 | Web Form in **Source** view.

13.4.1 Building the `WebTime` Application

- As you become more familiar with ASP.NET and building web sites in general, you might use Source view to perform high precision adjustments to your design or to program in the JavaScript language that executes in web browsers.
- For the purposes of this chapter, we'll keep things simple by working exclusively in Design mode.
- To switch to Design mode, you can click the Design `Button` at the bottom of the code editor window.

13.4.1 Building the `WebTime` Application

- The Solution Explorer
  - The Solution Explorer (Fig. 13.10) shows the contents of the website.
  - We expanded the node for `WebTime.aspx` to show you its code-behind file `WebTime.aspx.vb`.
  - Visual Web Developer's Solution Explorer contains several buttons that differ from Visual Basic Express.
  - The View Designer button allows you to open the Web Form in Design mode.

13.4.1 Building the WebTime Application

- The Copy Web Site button opens a dialog that allows you to move the files in this project to another location, such as a remote web server.
- This is useful if you're developing the application on your local computer but want to make it available to the public from a different location.
- Finally, the ASP.NET Configuration button takes you to a web page called the Web Site Administration Tool, where you can manipulate various settings and security options for your application.

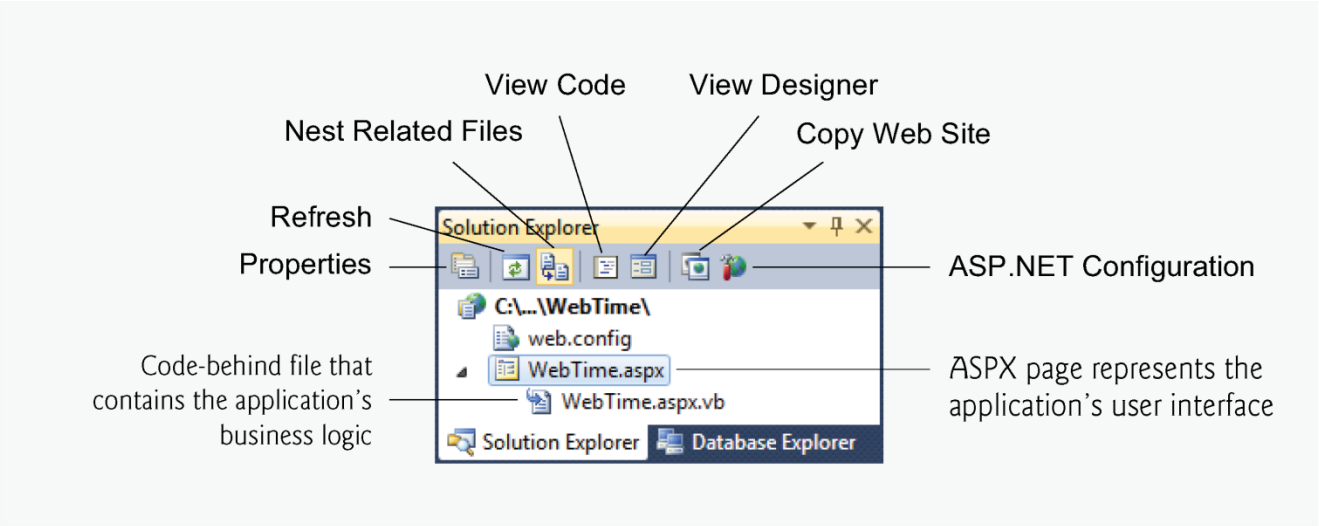

Fig. 13.10 | **Solution Explorer** window for an **Empty Web Site** project.

13.4.1 Building the `WebTime` Application

- If the ASPX file is not open in the IDE, you can open it in Design mode three ways:
  - double click it in the Solution Explorer
  - select it in the Solution Explorer and click the View Designer () `Button`
  - right click it in the Solution Explorer and select View Designer

13.4.1 Building the `WebTime` Application

- To open the code-behind file in the code editor, you can
  - double click it in the Solution Explorer
  - select the ASPX file in the Solution Explorer, then click the View Code () `Button`
  - right click the code-behind file in the Solution Explorer and select Open

13.4.1 Building the `WebTime` Application

- The *Toolbox*
  - Figure 13.11 shows the Toolbox displayed in the IDE when the project loads.
  - Part (a) displays the beginning of the Standard list of web controls, and part (b) displays the remaining web controls and the list of other control groups.
  - We discuss specific controls listed in Fig. 13.11 as they're used throughout the chapter.
  - Many of the controls have similar or identical names to Windows `Form`s controls presented earlier in the book.

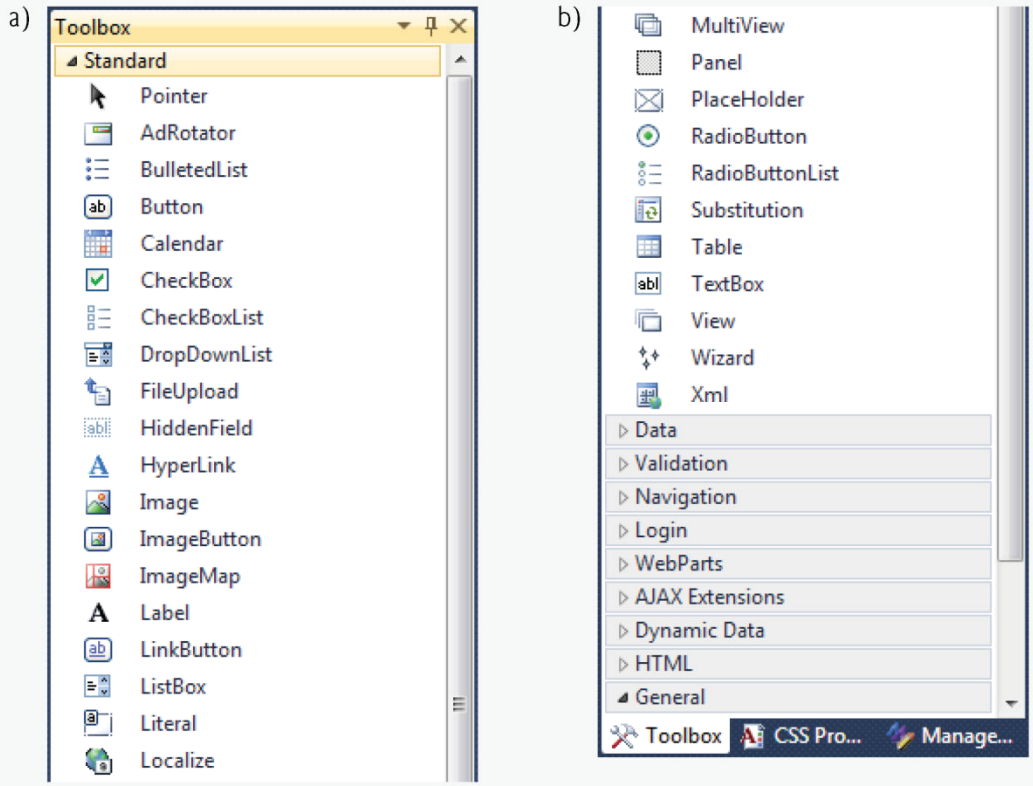

Fig. 13.11 | **Toolbox** in Visual Web Developer.

13.4.1 Building the `WebTime` Application

- The Web Forms Designer
  - Figure 13.12 shows the initial Web Form in Design mode.
  - You can drag and drop controls from the Toolbox onto the Web Form.
  - You can also type at the current cursor location to add so-called static text to the web page.
  - In response to such actions, the IDE generates the appropriate markup in the ASPX file.

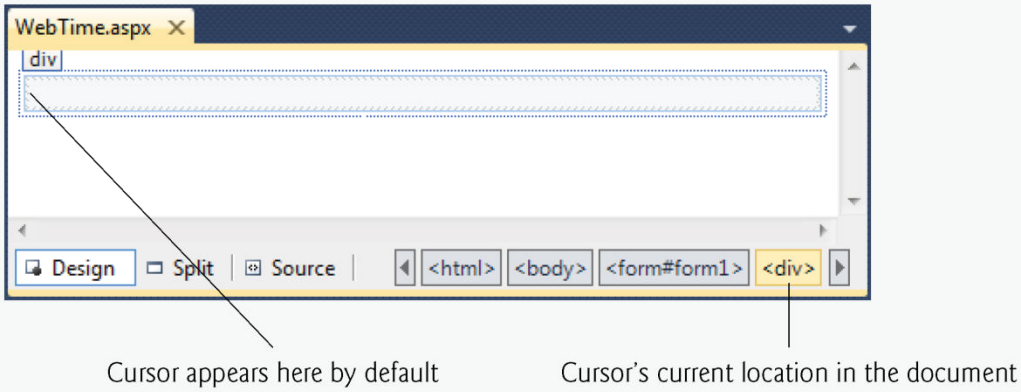

Fig. 13.12 | **Design** mode of the Web Forms Designer.

13.4.1 Building the `WebTime` Application

- Step 3: Changing the Title of the Page
  - Before designing the Web Form's content, you'll change its title to `A Simple Web Form Example`.
  - This title will be displayed in the web browser's title bar (see Fig. 13.4).
  - It's typically also used by search engines like Google and Bing when they index real websites for searching.
  - Every page should have a title.

13.4.1 Building the `WebTime` Application

- To change the title:
  - Ensure that the ASPX file is open in **Design** view.
  - View the Web Form's properties by selecting DOCUMENT, which represents the Web Form, from the drop-down list in the **Properties** window.
  - Modify the Title property in the **Properties** window by setting it to `A Simple Web Form Example`.

13.4.1 Building the `WebTime` Application

- Designing a Page
  - Designing a Web Form is similar to designing a Windows Form.
  - To add controls to the page, drag-and-drop them from the **Toolbox** onto the Web Form in **Design** view.
  - The Web Form and each control are objects that have properties, methods and events.
  - You can set these properties visually using the **Properties** window or programmatically in the code-behind file.
  - You can also type text directly on a Web Form at the cursor location.

13.4.1 Building the `WebTime` Application

- Controls and other elements are placed sequentially on a Web Form one after another in the order in which you drag-and-drop them onto the Web Form.
- The cursor indicates the insertion point in the page.
- If you want to position a control between existing text or controls, you can drop the control at a specific position between existing page elements.
- You can also rearrange controls with drag-and-drop actions in Design view.

13.4.1 Building the WebTime Application

- The positions of controls and other elements are relative to the Web Form's upper-left corner.
- This type of layout is known as relative positioning and it allows the browser to move elements and resize them based on the size of the browser window.
- Relative positioning is the default, and we'll use it throughout this chapter.
- For precise control over the location and size of elements, you can use absolute positioning in which controls are located exactly where you drop them on the Web Form.

13.4.1 Building the WebTime Application

- If you wish to use absolute positioning:
  - Select Tools > Options…., to display the Options dialog.
  - If it isn't checked already, check the Show all settings checkbox.
  - Next, expand the HTML Designer > CSS Styling node and ensure that the checkbox labeled Change positioning to absolute for controls added using Toolbox, paste or drag and drop is selected.

13.4.1 Building the WebTime Application

- Step 4: Adding Text and a **Label**
- You'll now add some text and a **Label** to the Web Form. Perform the following steps to add the text:
  - Ensure that the Web Form is open in **Design** mode.
  - Type the following text at the current cursor location:
    - `Current time on the Web server:`
  - Select the text you just typed, then select **Heading 2** from the **Block Format ComboBox** (Fig. 13.13) to format this text as a heading that will appear in a larger bold font. In more complex pages, headings help you specify the relative importance of parts of that content—like sections in a book chapter.

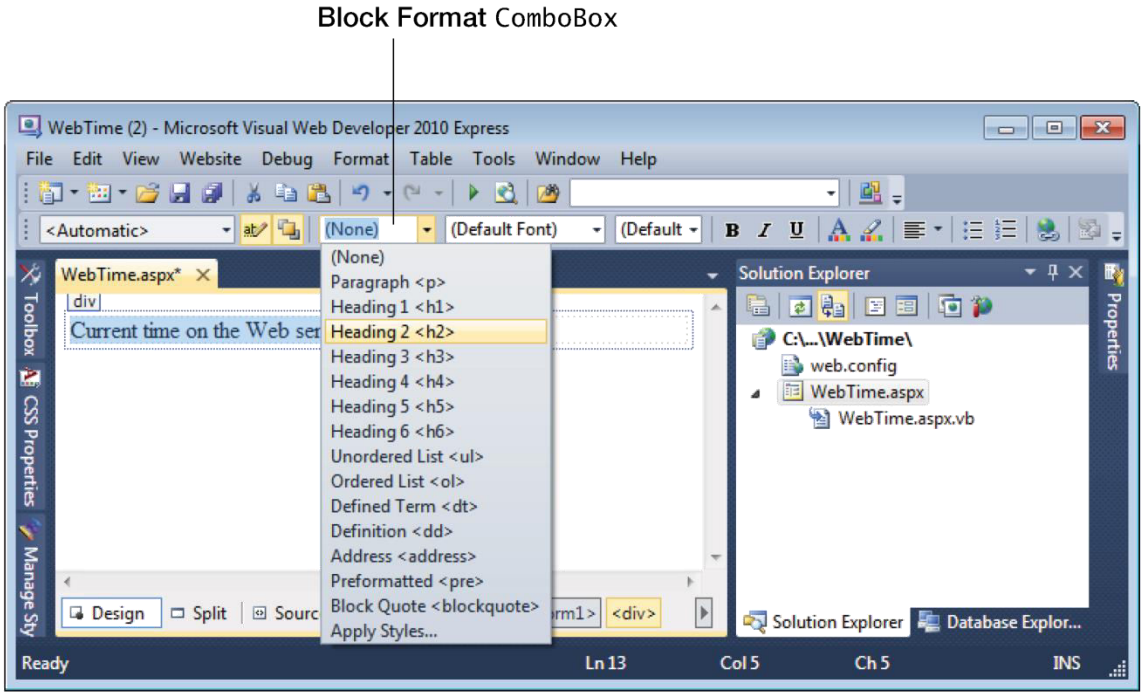

Fig. 13.13 | Changing the text to **Heading 2** heading.

13.4.1 Building the WebTime Application

- Click to the right of the text you just typed and press the *Enter key to start a new paragraph in the page. The Web Form should now appear as in Fig. 13.14.*
- Next, drag a `Label` control from the Toolbox into the new paragraph or double click the `Label` control in the Toolbox to insert the `Label` at the current cursor position.
- Using the Properties window, set the `Label`'s `(ID)` property to `timeLabel`. This specifies the variable name that will be used to programmatically change the `Label`'s `Text`.

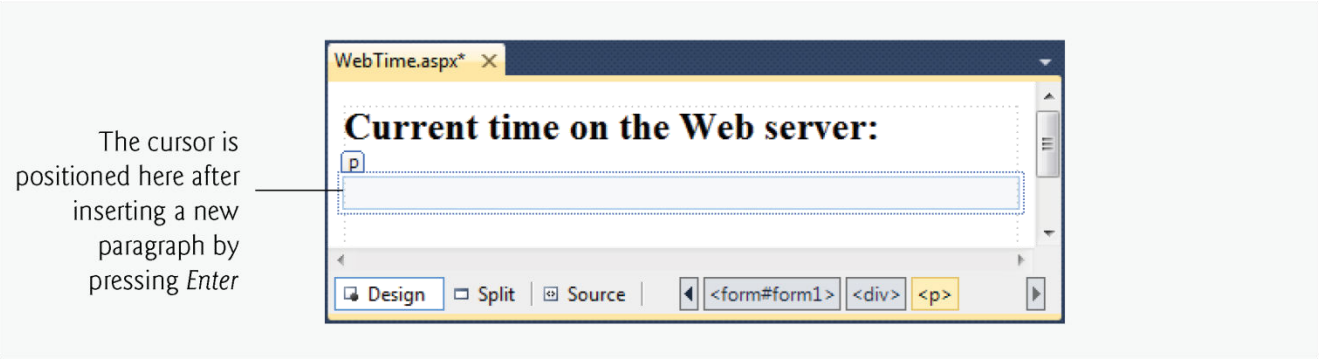

Fig. 13.14 | `WebTime.aspx` after inserting text and a new paragraph.

13.4.1 Building the WebTime Application

- Because, the `Label`'s `Text` will be set programmatically, delete the current value of the `Label`'s `Text` property. When a `Label` does not contain text, its name is displayed in square brackets in Design view (Fig. 13.15) as a placeholder for design and layout purposes. This text is not displayed at execution time.

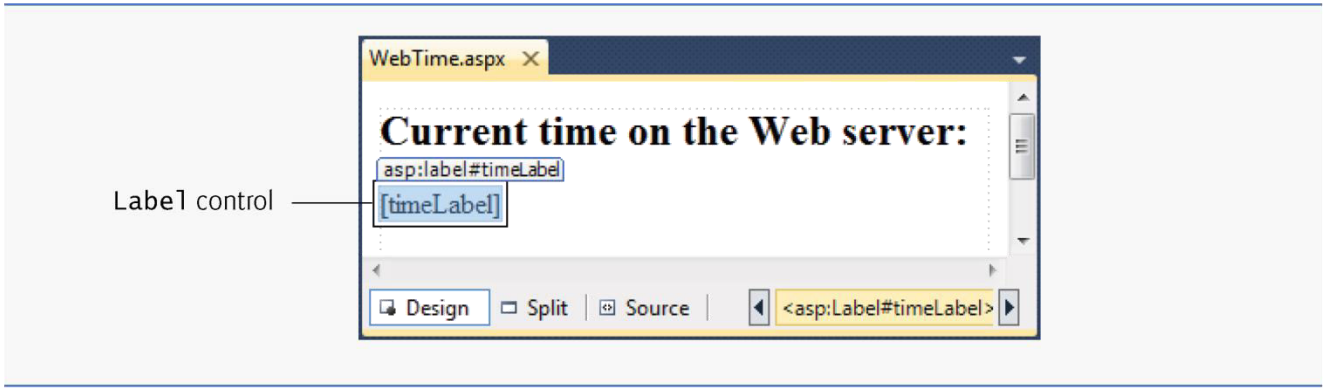

Fig. 13.15 | `WebTime.aspx` after adding a `Label`.

13.4.1 Building the WebTime Application

- Step 5: Formatting the `Label`
  - Formatting in a web page is performed with CSS (Cascading Style Sheets).
  - The details of CSS are beyond the scope of this book.
  - However, it's easy to use CSS to format text and elements in a Web Form via the tools built into Visual Web Developer.
  - In this example, we'd like to change the `Label`'s background color to black, its foreground color yellow and make its text size larger.

13.4.1 Building the `WebTime` Application

- To format the `Label`, perform the following steps:
  - Click the `Label` in Design view to ensure that it's selected.
  - Select View > Other Windows > CSS Properties to display the CSS Properties window at the left side of the IDE (Fig. 13.16).
  - Right click in the Applied Rules box and select New Style… to display the New Style dialog (Fig. 13.17).

13.4.1 Building the `WebTime` Application

- Type the new style's name—`.timeStyle`—in the Selector: `ComboBox`. Styles that apply to specific elements must be named with a dot (`.`) preceding the name. Such a style is called a CSS class.
- Each item you can set in the New Style dialog is known as a CSS attribute. To change `timeLabel`'s foreground color, select the Font category from the Category list, then select the yellow color swatch for the color attribute.
- Next, change the font-size attribute to `xx-large`.

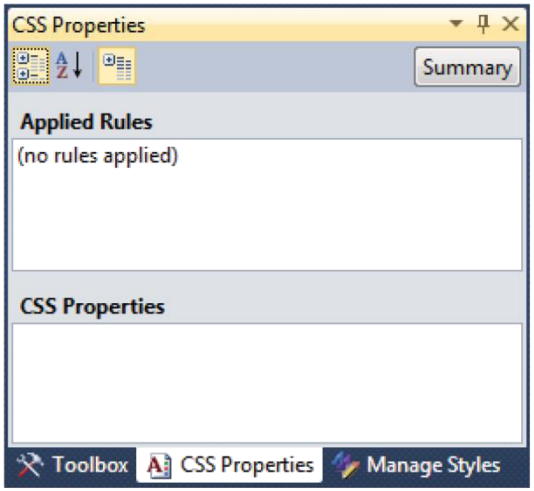

Fig. 13.16 | CSS Properties window.

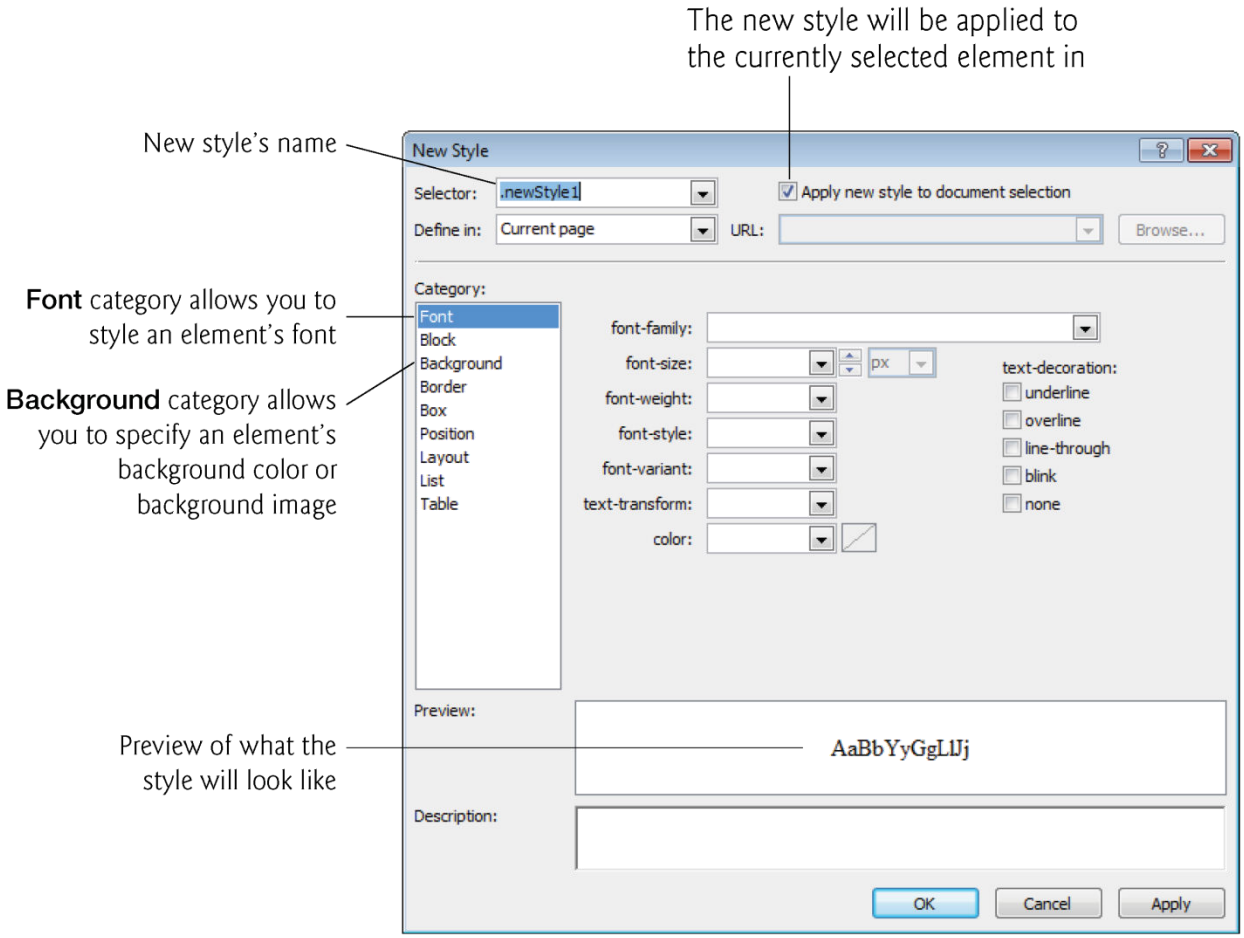

Fig. 13.17 | New Style dialog.

13.4.1 Building the WebTime Application

- To change timeLabel's background color, select the Background category, then select the black color swatch for the background-color attribute.

- The New Style dialog should now appear as shown in (Fig. 13.18).
- Click the OK Button to apply the style to the timeLabel so that it appears as shown in Fig. 13.19.
- Also, notice that the Label's CssClass property is now set to timeStyle in the Properties window.

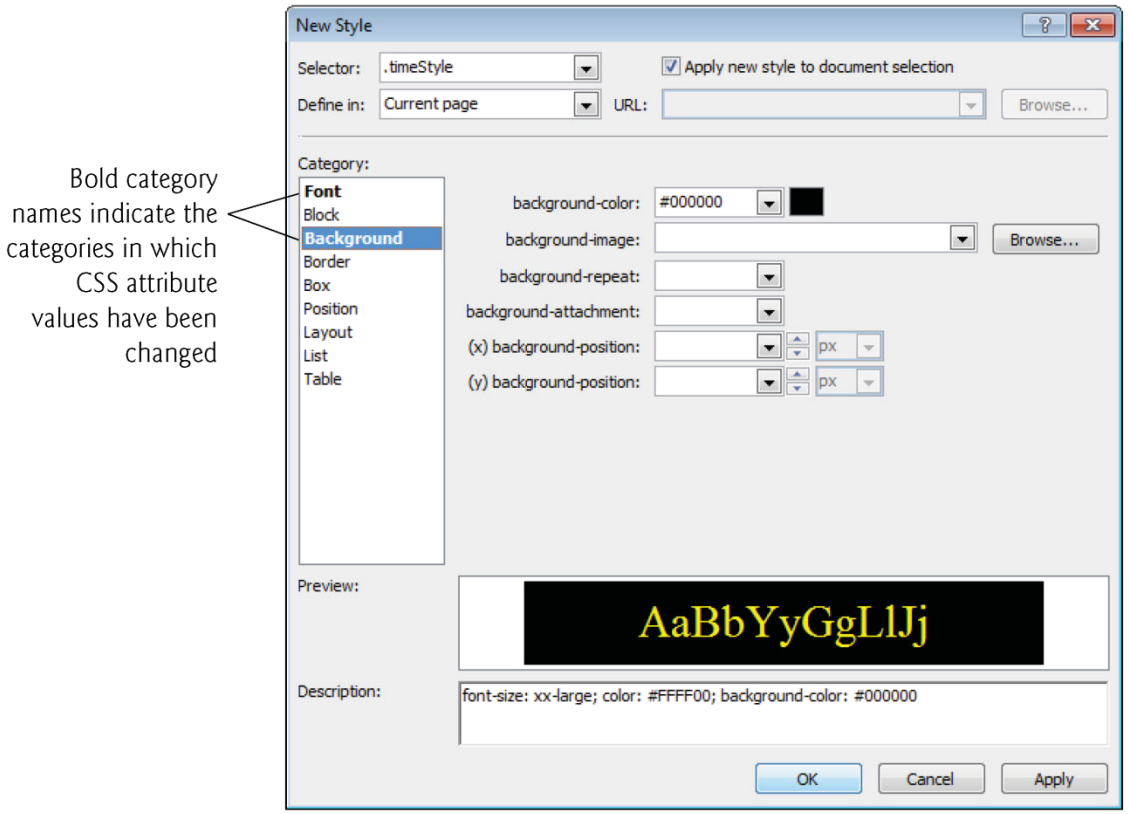

Fig. 13.18 | New Style dialog after changing the Label's font size, foreground color and background color.

Fig. 13.19 | **Design** view after changing the Label's style.

13.4.1 Building the `WebTime` Application

- Step 6: Adding Page Logic
  - Now you'll write code in the code-behind file to obtain the server's time and display it on the `Label`.
  - First, open `WebTime.aspx.vb` by double clicking its node in the Solution Explorer.
  - In this example, you'll add an event handler to the code-behind file to handle the Web Form's Init event, which occurs when the page is first requested by a web browser.
  - The event handler for this event—named Page_Init—initializes the page.
  - The only initialization required for this example is to set the `timeLabel`'s `Text` property to the time on the web server computer.

13.4.1 Building the WebTime Application

- To create the `Page_Init` event handler:
  - Select (Page Events) from the left `ComboBox` at the top of the code editor window.
  - Select Init from the right `ComboBox` at the top of the code editor window.
  - Complete the event handler by inserting the following code in the `Page_Init` event handler:

```
' display the server's current time in timeLabel
timeLabel.Text = DateTime.Now.ToString("hh:mm:ss")
```

13.4.1 Building the `WebTime` Application

- Step 7: Setting the Start Page and Running the Program
  - To ensure that `WebTime.aspx` loads when you execute this application, right click it in the Solution Explorer and select Set As Start Page.
  - You can now run the program in one of several ways.
  - At the beginning of Fig. 13.4, you learned how to view the Web Form by typing *Ctrl* + *F5* to run the application.
  - You can also right click an ASPX file in the Solution Explorer and select View in Browser.

13.4.1 Building the `WebTime` Application

- Both of these techniques execute the ASP.NET Development Server, open your default web browser and load the page into the browser, thus running the web application.
- The development server stops when you exit Visual Web Developer.
- If problems occur when running your application, you can run it in debug mode by selecting Debug > Start Debugging, by clicking the Start Debugging `Button` () or by typing *F5* to view the web page in a web browser with debugging enabled.

13.4.1 Building the `WebTime` Application

- You cannot debug a web application unless debugging is explicitly enabled in the application's Web.config file—a file that is generated when you create an ASP.NET web application.
- This file stores the application's configuration settings.
- You'll rarely need to manually modify `web.config`.
- The first time you select Debug > Start Debugging in a project, a dialog appears and asks whether you want the IDE to modify the `web.config` file to enable debugging.

13.4.1 Building the `WebTime` Application

- After you click OK, the IDE executes the application.
- You can stop debugging by selecting Debug > Stop Debugging.
- Regardless of how you execute the web application, the IDE will compile the project before it executes.
- In fact, ASP.NET compiles your web page whenever it changes between HTTP requests.
- For example, suppose you browse the page, then modify the ASPX file or add code to the code-behind file.
- When you reload the page, ASP.NET recompiles the page on the server before returning the response to the browser.
- This important behavior ensures that clients always see the latest version of the page.
- You can manually compile an entire website by selecting Build Web Site from the Debug menu in Visual Web Developer.

13.4.2 Examining `WebTime.aspx`'s Code-Behind File

- Figure 13.20 presents the code-behind file `WebTime.aspx.vb`.
- Line 3 of Fig. 13.20 begins the declaration of class `WebTime`.
- In Visual Basic, a class declaration can span multiple source-code files—the separate portions of the class declaration in each file are known as partial classes.
- The Partial modifier indicates that the code-behind file is part of a larger class.
- Like Windows `Forms` applications, the rest of the class's code is generated for you based on your visual interactions to create the application's GUI in Design mode.

13.4.2 Examining `WebTime.aspx`'s Code-Behind File

- That code is stored in other source code files as partial classes with the same name.
- The compiler assembles all the partial classes that have the same into a single class declaration.
- Line 4 indicates that `WebTime` inherits from class Page in namespace System.Web.UI.
- This namespace contains classes and controls for building web-based applications.
- Class `Page` represents the default capabilities of each page in a web application—all pages inherit directly or indirectly from this class.

```
1  ' Fig. 13.20: WebTime.aspx.vb
2  ' Code-behind file for a page that displays the current time.
3  Partial Class WebTime
4     Inherits System.Web.UI.Page
5
6     ' initializes the contents of the page
7     Protected Sub Page_Init(ByVal sender As Object, _
8        ByVal e As System.EventArgs) Handles Me.Init
9
10       ' display the server's current time in timeLabel
11       timeLabel.Text = DateTime.Now.ToString("hh:mm:ss")
12    End Sub ' Page_Init
13 End Class ' WebTime
```

Fig. 13.20 | Code-behind file for a page that displays the web server's time.

13.4.2 Examining `WebTime.aspx`'s Code-Behind File

- Lines 7–12 define the `Page_Init` event handler, which initializes the page in response to the page's `Init` event.
- The only initialization required for this page is to set the `timeLabel`'s `Text` property to the time on the web server computer.
- The statement in line 11 retrieves the current time (`DateTime.Now`) and formats it as *hh*`:`*mm*`:`*ss*.
- For example, 9 AM is formatted as 09:00:00, and 2:30 PM is formatted as 02:30:00.
- As you'll see, variable `timeLabel` represents an ASP.NET `Label` control.
- The ASP.NET controls are defined in namespace System.Web.UI.WebControls.

13.5 Standard Web Controls: Designing a Form

- This section introduces some of the web controls located in the Standard section of the Toolbox (Fig. 13.11).
- Figure 13.21 summarizes the controls used in the next example.

Web control	Description
`TextBox`	Gathers user input and displays text.
`Button`	Triggers an event when clicked.
`HyperLink`	Displays a hyperlink.
`DropDownList`	Displays a drop-down list of choices from which a user can select an item.
`RadioButtonList`	Groups radio buttons.
`Image`	Displays images (for example, PNG, GIF and JPG).

Fig. 13.21 | Commonly used web controls.

13.5 Examining WebTime.aspx's Code-Behind File

- A Form Gathering User Input
  - Figure 13.22 depicts a form for gathering user input.
  - This example does not perform any tasks—that is, no action occurs when the user clicks Register.
  - As an exercise, we ask you to provide the functionality.
  - Here we focus on the steps for adding these controls to a Web Form and for setting their properties.
  - Subsequent examples demonstrate how to handle the events of many of these controls.
  - To execute this application:

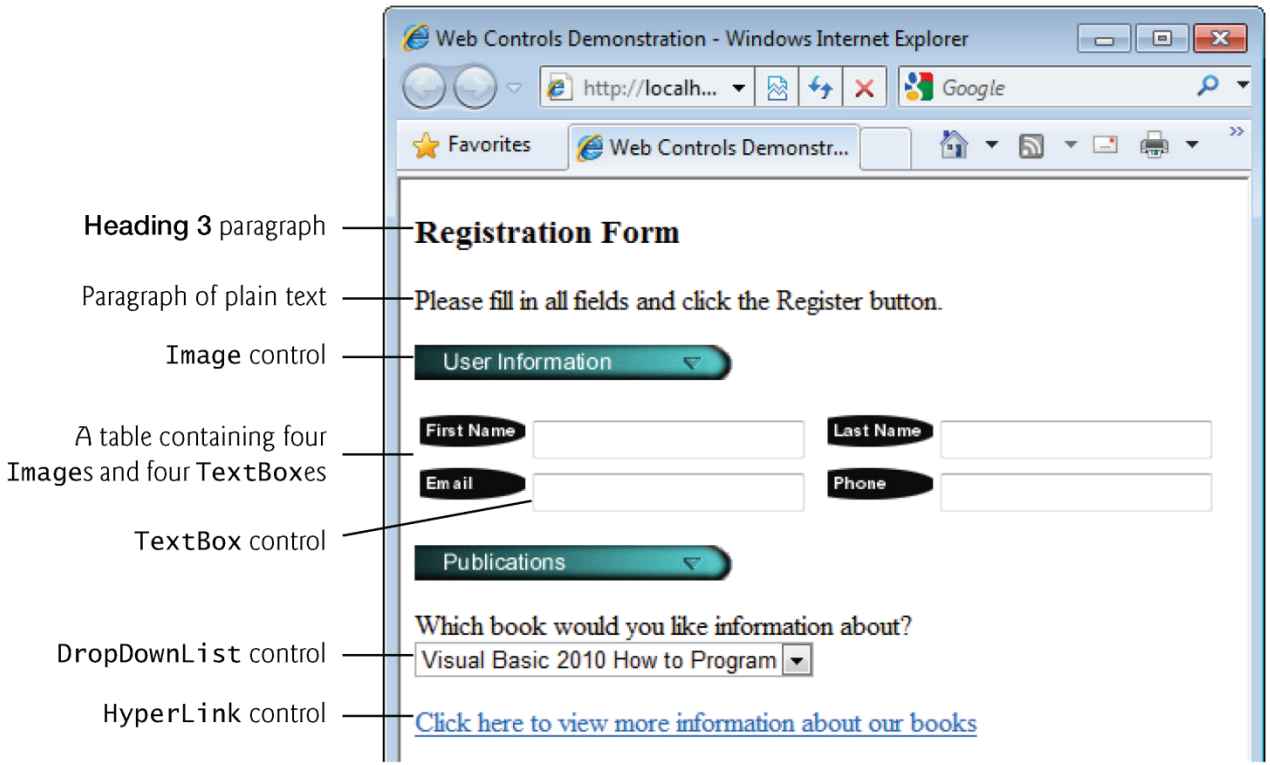

Fig. 13.22 | Web Form that demonstrates web controls. (Part 1 of 2.)

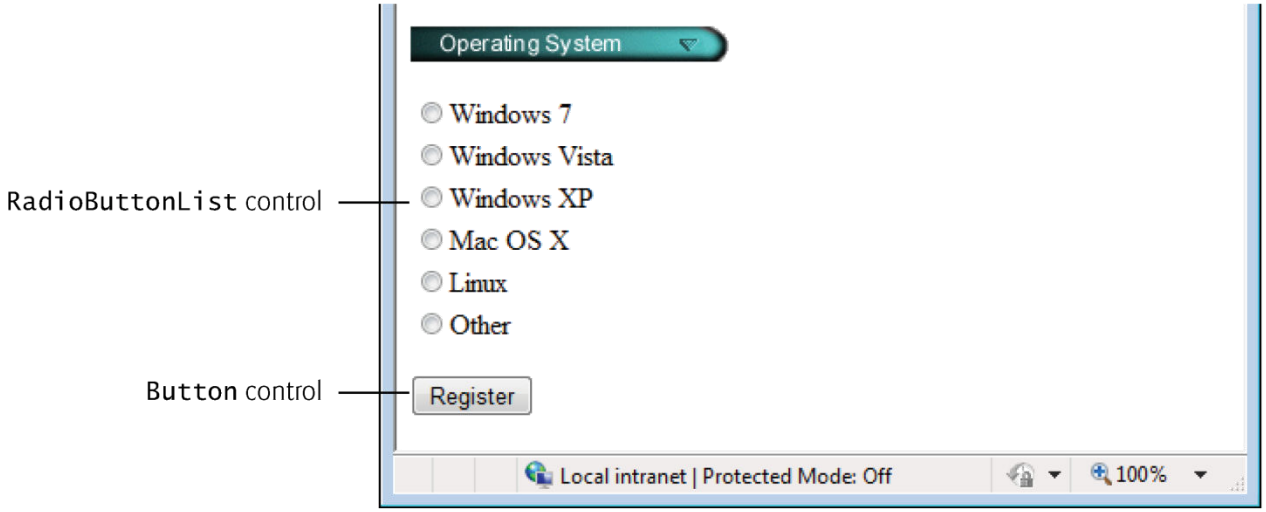

Fig. 13.22 | Web Form that demonstrates web controls. (Part 2 of 2.)

13.5 Examining `WebTime.aspx`'s Code-Behind File

- Select Open Web Site… from the File menu.
- In the Open Web Site dialog, ensure that File System is selected, then navigate to this chapter's examples, select the `WebControls` folder and click the Open `Button`.
- Select `WebControls.aspx` in the Solution Explorer, then type *Ctrl + F5* to execute the web application in your default web browser.

13.5 Examining WebTime.aspx's Code-Behind File

- Create the Web Site
  - To begin, follow the steps in Section 13.4.1 to create an Empty Web Site named `WebControls`, then add a Web Form named `WebControls.aspx` to the project.
  - Set the document's `Title` property to `"Web Controls Demonstration"`.
  - To ensure that `WebControls.aspx` loads when you execute this application, right click it in the Solution Explorer and select Set As Start Page.

13.5 Examining WebTime.aspx's Code-Behind File

- Adding the Images to the Project
  - The images used in this example are located in the `images` folder with this chapter's examples.
  - Before you can display images in the Web Form, they must be added to your project.

13.5 Examining WebTime.aspx's Code-Behind File

- To add the `images` folder to your project:
  - Open Windows Explorer.
  - Locate and open this chapter's examples folder (`ch13`).
  - Drag the `images` folder from Windows Explorer into Visual Web Developer's Solution Explorer window and drop the folder on the name of your project.
- The IDE will automatically copy the folder and its contents into your project.

13.5 Examining WebTime.aspx's Code-Behind File

- Adding Text and an Image to the Form
- Next, you'll begin creating the page. Perform the following steps:
  - First create the page's heading. At the current cursor position on the page, type the text `"Registration Form"`, then use the Block Format `ComboBox` in the IDE's toolbar to change the text to Heading 3 format.
  - Press *Enter* to start a new paragraph, then type the text `"Please fill in all fields and click the Register button"`.

13.5 Examining WebTime.aspx's Code-Behind File

- Press *Enter* to start a new paragraph, then double click the Image control in the Toolbox. This control inserts an image into a web page, at the current cursor position. Set the Image's (ID) property to userInformationImage. The ImageUrl property specifies the location of the image to display. In the Properties window, click the ellipsis for the ImageUrl property to display the Select Image dialog. Select the images folder under Project folders: to display the list of images. Then select the image user.png.
- Click OK to display the image in Design view, then click to the right of the Image and press *Enter* to start a new paragraph.

13.5 Examining WebTime.aspx's Code-Behind File

- Adding a Table to the Form
- Form elements are often placed in tables for layout purposes—like the elements that represent the first name, last name, e-mail and phone information in Fig. 13.22.
  - Next, you'll create a table with two rows and two columns in Design mode.
  - Select Table > Insert Table to display the Insert Table dialog (Fig. 13.23). This dialog allows you to configure the table's options.
  - Under Size, ensure that the values of Rows and Columns are both 2—these are the default values.
  - Click OK to close the Insert Table dialog and create the table.

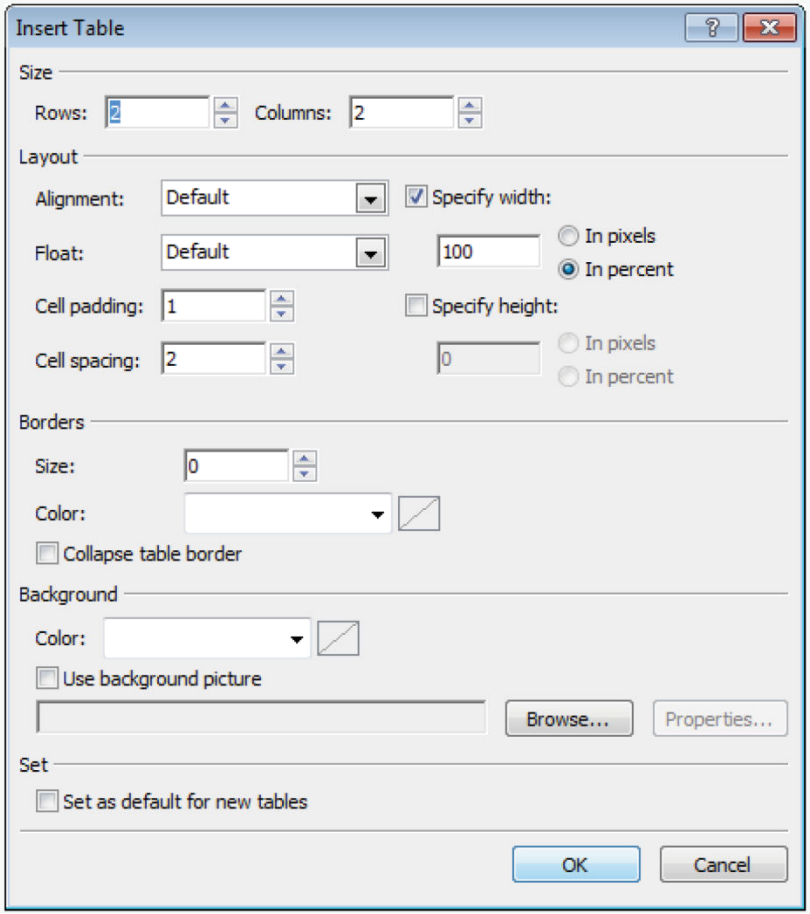

Fig. 13.23 | **Insert Table** dialog.

13.5 Examining WebTime.aspx's Code-Behind File

- By default, the contents of a table cell are aligned vertically in the middle of the cell.
- We changed the vertical alignment of all cells in the table by setting the `valign` property to `top` in the Properties window.
- This causes the content in each table cell to align with the top of the cell.
- You can set the `valign` property for each table cell individually or by selecting all the cells in the table at once, then changing the `valign` property's value.

13.5 Examining WebTime.aspx's Code-Behind File

- After creating the table, controls and text can be added to particular cells to create a neatly organized layout.
- Next, add `Image` and `TextBox` controls to each the four table cells as follows:
  - Click the table cell in the first row and first column of the table, then double click the `Image` control in the Toolbox. Set its `(ID)` property to `firstNameImage` and set its `ImageUrl` property to the image `fname.png`.

13.5 Examining WebTime.aspx's Code-Behind File

- Next, double click the `TextBox` control in the Toolbox. Set its `(ID)` property to `firstNameTextBox`. As in Windows Forms, a TextBox control allows you to obtain text from the user and display text to the user
- Repeat this process in the first row and second column, but set the `Image`'s `(ID)` property to `lastNameImage` and its `ImageUrl` property to the image `lname.png`, and set the `TextBox`'s `(ID)` property to `lastNameTextBox`.

13.5 Examining WebTime.aspx's Code-Behind File

- Repeat *Steps 1* and *2* in the second row and first column, but set the `Image`'s `(ID)` property to `emailImage` and its `ImageUrl` property to the image `email.png`, and set the `TextBox`'s `(ID)` property to `emailTextBox`.
- Repeat *Steps 1* and *2* in the second row and second column, but set the `Image`'s `(ID)` property to `phoneImage` and its `ImageUrl` property to the image `phone.png`, and set the `TextBox`'s `(ID)` property to `phoneTextBox`.

13.5 Examining WebTime.aspx's Code-Behind File

- Creating the Publications Section of the Page
- This section contains an `Image`, some text, a `DropDownList` control and a `HyperLink` control.
- Perform the following steps to create this section:
  - Click below the table, then use the techniques you've already learned in this section to add an `Image` named `publicationsImage` that displays the `publications.png` image.

13.5 Examining WebTime.aspx's Code-Behind File

- Click to the right of the `Image`, then press *Enter* and type the text `"Which book would you like information about?"` in the new paragraph.
- Hold the *Shift* key and press *Enter* to create a new line in the current paragraph, then double click the DropDownList control in the `Toolbox`. Set its `(ID)` property to `booksDropDownList`. This control is similar to the Windows `Forms` `ComboBox` control, but doesn't allow users to type text. When a user clicks the drop-down list, it expands and displays a list from which the user can make a selection.

13.5 Examining WebTime.aspx's Code-Behind File

- You can add items to the `DropDownList` using the ListItem Collection Editor, which you can access by clicking the ellipsis next to the `DropDownList`'s `Items` property in the Properties window, or by using the DropDownList Tasks smart-tag menu. To open this menu, click the small arrowhead that appears in the upper-right corner of the control in Design mode (Fig. 13.24). Visual Web Developer displays smart-tag menus for many ASP.NET controls to facilitate common tasks.

13.5 Examining WebTime.aspx's Code-Behind File

- Clicking Edit Items... in the DropDownList Tasks menu opens the ListItem Collection Editor, which allows you to add `ListItem` elements to the `DropDownList`. Add items for `"Visual Basic 2010 How to Program"`, `"Visual C# 2008 How to Program"`, `"Java How to Program"` and `"C++ How to Program"` by clicking the Add `Button` four times. For each item, select it, then set its `Text` property to one of the four book titles.

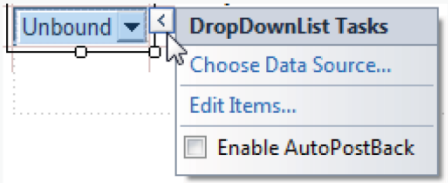

Fig. 13.24 | **DropDownList Tasks** smart-tag menu.

13.5 Examining WebTime.aspx's Code-Behind File

- Click to the right of the `DropDownList` and press *Enter* to start a new paragraph, then double click the HyperLink control in the Toolbox to add a hyperlink to the web page. Set its `(ID)` property to `booksHyperLink` and its `Text` property to `"Click here to view more information about our books"`.

13.5 Examining WebTime.aspx's Code-Behind File

- Set the NavigateUrl property to `http://www.deitel.com`. This specifies the resource or web page that will be requested when the user clicks the `HyperLink`. Setting the Target property to `_blank` specifies that the requested web page should open in a new browser window. By default, `HyperLink` controls cause pages to open in the same browser window.

13.5 Examining WebTime.aspx's Code-Behind File

- Completing the Page
  - Next you'll create the Operating System section of the page and the Register `Button`.
  - This section contains a RadioButtonList control, which provides a series of radio buttons from which the user can select only one.
  - The RadioButtonList Tasks smart-tag menu provides an Edit Items… link to open the ListItem Collection Editor so that you can create the items in the list.

13.5 Examining WebTime.aspx's Code-Behind File

- Perform the following steps:
  - Click to the right of the `HyperLink` control and press *Enter* to create a new paragraph, then add an `Image` named `osImage` that displays the `os.png` image.
  - Click to the right of the `Image` and press *Enter* to create a new paragraph, then add a `RadioButtonList`. Set its `(ID)` property to `osRadioButtonList`. Use the ListItem Collection Editor to add the items shown in Fig. 13.22.

13.5 Examining WebTime.aspx's Code-Behind File

- Finally, click to the right of the `RadioButtonList` and press *Enter* to create a new paragraph, then add a Button. A `Button` web control represents a button that triggers an action when clicked. Set its `(ID)` property to `registerButton` and its `Text` property to `Register`. As stated earlier, clicking the Register button in this example does not do anything.

- You can now execute the application (*Ctrl + F5*) to see the Web Form in your browser.

13.6 Validation Controls

- This section introduces a different type of web control, called a validation control or validator, which determines whether the data in another web control is in the proper format.
- For example, validators can determine whether a user has provided information in a required field or whether a zip-code field contains exactly five digits.
- Validators provide a mechanism for validating user input on the client.
- When the page is sent to the client, the validator is converted into JavaScript that performs the validation in the client web browser.

13.6 Examining WebTime.aspx's Code-Behind File

- JavaScript is a scripting language that enhances the functionality of web pages and is typically executed on the client.
- Unfortunately, some client browsers might not support scripting or the user might disable it.
- For this reason, you should always perform validation on the server.
- ASP.NET validation controls can function on the client, on the server or both.

13.6 Validation Controls

- Validating Input in a Web Form
  - The Web Form in Fig. 13.25 prompts the user to enter a name, e-mail address and phone number.
  - A website could use a form like this to collect contact information from visitors.
  - After the user enters any data, but before the data is sent to the web server, validators ensure that the user entered a value in each field and that the e-mail address and phone-number values are in an acceptable format.

13.6 Validation Controls

- In this example, (555) 123-4567, 555-123-4567 and 123-4567 are all considered valid phone numbers.
- Once the data is submitted, the web server responds by displaying a message that repeats the submitted information.
- A real business application would typically store the submitted data in a database or in a file on the server.
- We simply send the data back to the client to demonstrate that the server received the data.

13.6 Validation Controls

- To execute this application:
  - Select Open Web Site… from the File menu.
  - In the Open Web Site dialog, ensure that File System is selected, then navigate to this chapter's examples, select the `Validation` folder and click the Open `Button`.
  - Select `Validation.aspx` in the Solution Explorer, then type *Ctrl + F5* to execute the web application in your default web browser.

13.6 Validation Controls

- In the sample output:
  - Fig. 13.25(a) shows the initial Web Form
  - Fig. 13.25(b) shows the result of submitting the form before typing any data in the `TextBox`es
  - Fig. 13.25(c) shows the results after entering data in each `TextBox`, but specifying an invalid e-mail address and invalid phone number
  - Fig. 13.25(d) shows the results after entering valid values for all three `TextBox`es and submitting the form.

a) Initial Web Form

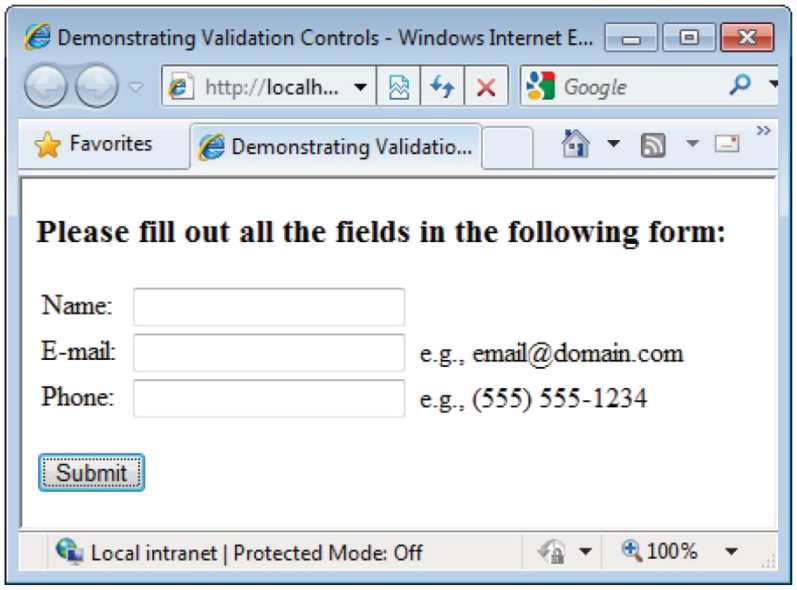

Fig. 13.25 | Validators in a Web Form that retrieves user contact information. (Part 1 of 4.)

b) Web Form after the user presses the **Submit** `Button` without having entered any data in the `TextBox`es; each `TextBox` is followed by an error message that was displayed by a validation control

`RequiredFieldValidator` controls

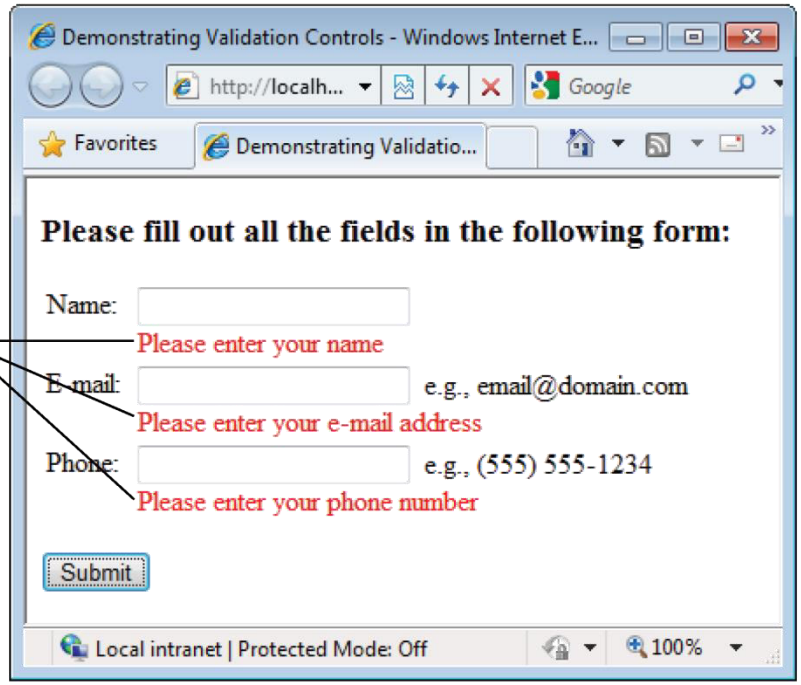

Fig. 13.25 | Validators in a Web Form that retrieves user contact information. (Part 2 of 4.)

c) Web Form after the user enters a name, an invalid e-mail address and an invalid phone number in the `TextBox`es, then presses the **Submit** `Button`; the validation controls display error messages in response to the invalid e-mail and phone number values

`RegularExpressionValidator` controls

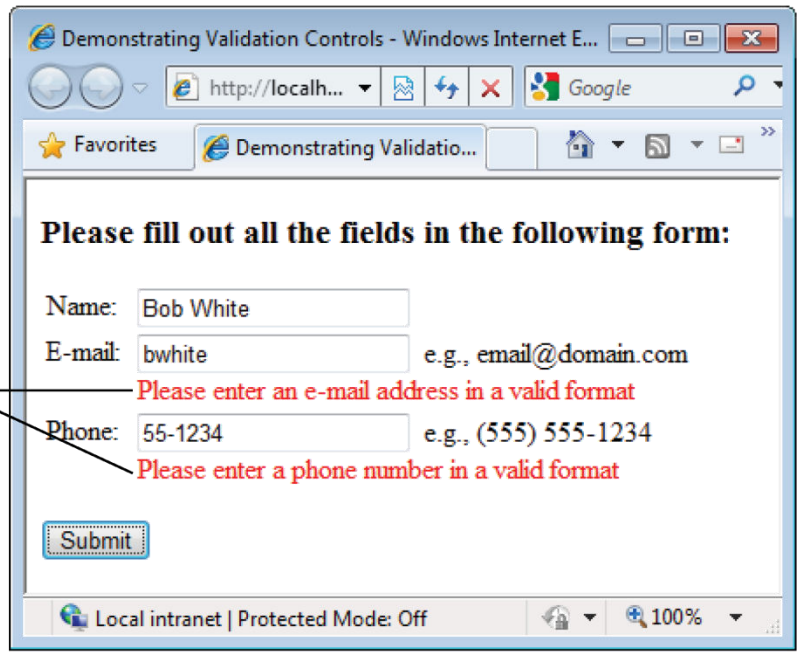

Fig. 13.25 | Validators in a Web Form that retrieves user contact information. (Part 3 of 4.)

d) The Web Form after the user enters valid values for all three `TextBox`es and presses the **Submit** `Button`

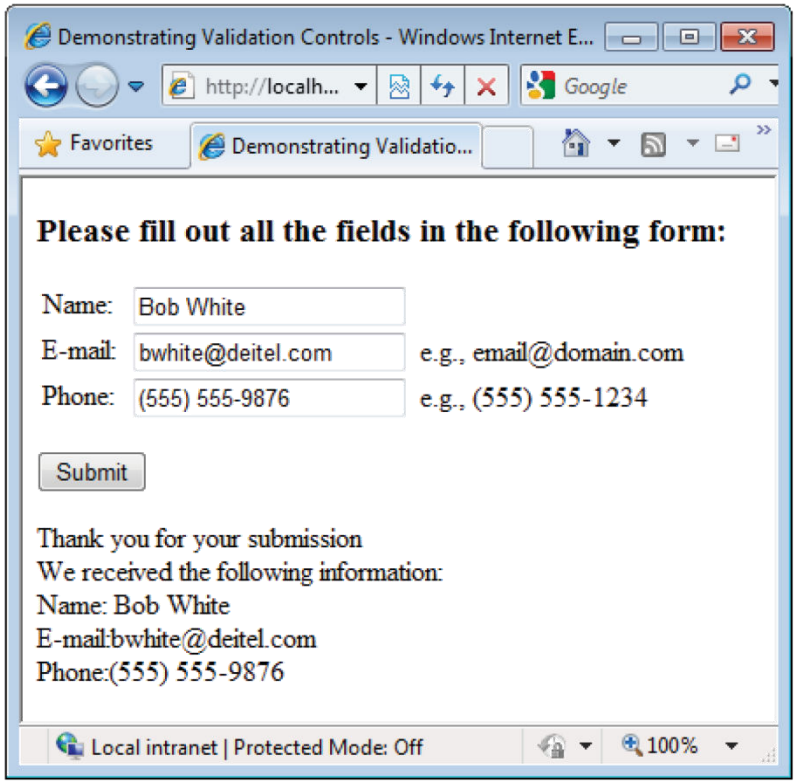

Fig. 13.25 | Validators in a Web Form that retrieves user contact information. (Part 4 of 4.)

13.6 Validation Controls

- Creating the Web Site
  - To begin, follow the steps in Section 13.4.1 to create an Empty Web Site named `Validation`, then add a Web Form named `Validation.aspx` to the project.
  - Set the document's `Title` property to `"Demonstrating Validation Controls"`.
  - To ensure that `Validation.aspx` loads when you execute this application, right click it in the Solution Explorer and select Set As Start Page.

13.6 Validation Controls

- Creating the GUI
- To create the page, perform the following steps:
  - Type `"Please fill out all the fields in the following form:"`, then use the Block Format `ComboBox` in the IDE's toolbar to change the text to Heading 3 format and press *Enter* to create a new paragraph.
  - Insert a three row and two column table. You'll add elements to the table momentarily.

13.6 Validation Controls

- Click below the table and add a `Button`. Set its `(ID)` property to `submitButton` and its `Text` property to `Submit`. Press *Enter* to create a new paragraph. By default, a `Button` control in a Web Form sends the contents of the form back to the server for processing.

13.6 Validation Controls

- Add a `Label`. Set its `(ID)` property to `outputLabel` and clear its `Text` property—you'll set it programmatically when the user clicks the `submitButton`. Set the `outputLabel`'s Visible property to `False`, so the `Label` does not appear in the client's browser when the page loads for the first time. You'll programmatically display this `Label` after the user submits valid data.

- Next you'll add text and controls to the table you created in *Step 2 above*.

13.6 Validation Controls

- Perform the following steps:
  - In the left column, type the text `"Name:"` in the first row, `"E-mail:"` in the second row and `"Phone:"` in the row column.
  - In the right column of the first row, add a `TextBox` and set its `(ID)` property to `nameTextBox`.

13.6 Validation Controls

- In the right column of the second row, add a `TextBox` and set its `(ID)` property to `emailTextBox`. Then type the text `"e.g., email@domain.com"` to the right of the `TextBox`.
- In the right column of the third row, add a `TextBox` and set its `(ID)` property to `phoneTextBox`. Then type the text `"e.g., (555) 555-1234"` to the right of the `TextBox`.

13.6 Validation Controls

- Using `RequiredFieldValidator` Controls
- We use three RequiredFieldValidator controls (found in the Validation section of the Toolbox) to ensure that the name, e-mail address and phone number `TextBox`es are not empty when the form is submitted.
- A `RequiredFieldValidator` makes an input control a required field.
- If such a field is empty, validation fails.

13.6 Validation Controls

- Add a `RequiredFieldValidator` as follows:
  - Click to the right of the `nameTextBox` in the table and press *Enter* to move to the next line.
  - Add a `RequiredFieldValidator`, set its `(ID)` to `nameRequiredFieldValidator` and set the `ForeColor` property to `Red`.
  - Set the validator's ControlToValidate property to `nameTextBox` to indicate that this validator verifies the `nameTextBox`'s contents.

13.6 Validation Controls

- Set the validator's ErrorMessage property to `"Please enter your name"`. This is displayed on the Web Form only if the validation fails.
- Set the validator's Display property to `Dynamic`, so the validator occupies space on the Web Form only when validation fails. When this occurs, space is allocated dynamically, causing the controls below the validator to shift downward to accommodate the `ErrorMessage`, as seen in Fig. 13.25(a)–(c).

13.6 Validation Controls

- Repeat these steps to add two more `RequiredFieldValidator`s in the second and third rows of the table.
- Set their `(ID)` properties to `emailRequiredFieldValidator` and `phoneRequiredFieldValidator`, respectively, and set their `ErrorMessage` properties to `"Please enter your email address"` and `"Please enter your phone number"`, respectively.

13.6 Validation Controls

- Using `RegularExpressionValidator` Controls
  - This example also uses two RegularExpressionValidator controls to ensure that the e-mail address and phone number entered by the user are in a valid format.
  - Regular expressions are beyond the scope of this book; however, Visual Web Developer provides several predefined regular expressions that you can simply select to take advantage of this powerful validation control.

13.6 Validation Controls

- Add a `RegularExpressionValidator` as follows:
  - Click to the right of the `emailRequiredFieldValidator` in the second row of the table and add a `RegularExpressionValidator`, then set its `(ID)` to `emailRegularExpressionValidator` and its `ForeColor` property to `Red`.
  - Set the `ControlToValidate` property to `emailTextBox` to indicate that this validator verifies the `emailTextBox`'s contents.

13.6 Validation Controls

- Set the validator's `ErrorMessage` property to `"Please enter an e-mail address in a valid format"`.
- Set the validator's `Display` property to `Dynamic`, so the validator occupies space on the Web Form only when validation fails.

13.6 Validation Controls

- Repeat the preceding steps to add another `RegularExpressionValidator` in the third row of the table.
- Set its `(ID)` property to `phoneRequiredFieldValidator` and its `ErrorMessage` property to `"Please enter a phone number in a valid format"`, respectively.
- A `RegularExpressionValidator`'s ValidationExpression property specifies the regular expression that validates the `ControlToValidate`'s contents.

13.6 Validation Controls

- Clicking the ellipsis next to property `ValidationExpression` in the Properties window displays the Regular Expression Editor dialog, which contains a list of Standard expressions for phone numbers, zip codes and other formatted information.
- For the `emailRegularExpressionValidator`, we selected the standard expression Internet e-mail address.
- If the user enters text in the `emailTextBox` that does not have the correct format and either clicks in a different text box or attempts to submit the form, the `ErrorMessage` text is displayed in red.

13.6 Validation Controls

- For the `phoneRegularExpressionValidator`, we selected U.S. phone number to ensure that a phone number contains an optional three-digit area code either in parentheses and followed by an optional space or without parentheses and followed by a required hyphen.
- After an optional area code, a phone number must contain three digits, a hyphen and another four digits.
- For example, `(555) 123-4567`, `555-123-4567` and `123-4567` are all valid phone numbers.

13.6 Validation Controls

- Submitting the Web Form's Contents to the Server
  - If all five validators are successful (that is, each `TextBox` is filled in, and the e-mail address and phone number provided are valid), clicking the Submit button sends the form's data to the server.
  - As shown in Fig. 13.25(d), the server then responds by displaying the submitted data in the `outputLabel`.

13.6 Validation Controls

- Examining the Code-Behind File for a Web Form That Receives User Input
  - Figure 13.26 shows the code-behind file for this application.
  - Notice that this code-behind file does not contain any implementation related to the validators.
  - We say more about this soon.
  - In this example, we respond to the page's Load event to process the data submitted by the user.
  - This event occurs each time the page loads into a web browser—as opposed to the `Init` event, which executes only the first time the page is requested by the user.

13.6 Validation Controls

- The event handler for this event is Page_Load (lines 7–30).
- To create the event handler, open `Validation.aspx.vb` in the code editor and perform the following steps:
  - Select (Page Events) from the left `ComboBox` at the top of the code editor window.
  - Select Load from the right `ComboBox` at the top of the code editor window.
  - Complete the event handler by inserting the code from Fig. 13.26.

```
1  ' Fig. 13.26: Validation.aspx.vb
2  ' Code-behind file for the form demonstrating validation controls.
3  Partial Class Validation
4     Inherits System.Web.UI.Page
5
6     ' Page_Load event handler executes when the page is loaded
7     Protected Sub Page_Load(ByVal sender As Object,
8        ByVal e As System.EventArgs) Handles Me.Load
9
10       ' if this is not the first time the page is loading
11       ' (i.e., the user has already submitted form data)
12       If IsPostBack Then
13          Validate() ' validate the form
14
15          If IsValid Then
16             ' retrieve the values submitted by the user
17             Dim name As String = nameTextBox.Text
18             Dim email As String = emailTextBox.Text
19             Dim phone As String = phoneTextBox.Text
20
```

Fig. 13.26 | Code-behind file for a Web Form that obtains a user's contact information. (Part 1 of 2.)

```
            ' create a table indicating the submitted values
            outputLabel.Text = "Thank you for your submission<br/>" &
               "We received the following information:<br/>"
            outputLabel.Text &=
               String.Format("Name: {0}{1}E-mail:{2}{1}Phone:{3}",
                  name, "<br/>", email, phone)
            outputLabel.Visible = True ' display the output message
         End If
      End If
   End Sub ' Page_Load
End Class ' Validation
```

Fig. 13.26 | Code-behind file for a Web Form that obtains a user's contact information. (Part 2 of 2.)

13.6 Validation Controls

- Differentiating Between the First Request to a Page and a Postback
  - Web programmers using ASP.NET often design their web pages so that the current page reloads when the user submits the form; this enables the program to receive input, process it as necessary and display the results in the same page when it's loaded the second time.
  - These pages usually contain a form that, when submitted, sends the values of all the controls to the server and causes the current page to be requested again.

13.6 Validation Controls

- This event is known as a postback.
- Line 12 uses the IsPostBack property of class `Page` to determine whether the page is being loaded due to a postback.
- The first time that the web page is requested, `IsPostBack` is `False`, and the page displays only the form for user input.
- When the postback occurs (from the user clicking Submit), `IsPostBack` is `True`.

13.6 Validation Controls

- Server-Side Web Form Validation
  - Server-side Web Form validation must be implemented programmatically.
  - Line 13 calls the current `Page`'s Validate method to validate the information in the request.
  - This validates the information as specified by the validation controls in the Web Form.

13.6 Validation Controls

- Line 15 uses the IsValid property of class `Page` to check whether the validation succeeded.
- If this property is set to `True` (that is, validation succeeded and the Web Form is valid), then we display the Web Form's information.
- Otherwise, the web page loads without any changes, except any validator that failed now displays its `ErrorMessage`.

13.6 Validation Controls

- Processing the Data Entered by the User
  - Lines 17–19 retrieve the values of `nameTextBox`, `emailTextBox` and `phoneTextBox`.
  - When data is posted to the web server, the data that the user entered is accessible to the web application through the web controls' properties.

13.6 Validation Controls

- Next, lines 22–27 set `outputLabel`'s `Text` to display a message that includes the name, e-mail and phone information that was submitted to the server.
- In lines 22, 23 and 26, notice the use of `<br/>` rather than `vbCrLf` to start new lines in the `outputLabel`—`<br/>` is the markup for a line break in a web page.
- Line 27 sets the `outputLabel`'s `Visible` property to `True`, so the user can see the thank-you message and submitted data when the page reloads in the client web browser.

13.7 Session Tracking

- Originally, critics accused the Internet and business of failing to provide the customized service typically experienced in "brick-and-mortar" stores.
- To address this problem, businesses established mechanisms by which they could personalize users' browsing experiences, tailoring content to individual users.
- Businesses achieve this level of service by tracking each customer's movement through the Internet and combining the collected data with information provided by the consumer, including billing information, personal preferences, interests and hobbies.

13.7 Session Tracking

- Personalization
  - Personalization makes it possible for businesses to communicate effectively with their customers and also improves users' ability to locate desired products and services.
  - Companies that provide content of particular interest to users can establish relationships with customers and build on those relationships over time.
  - Furthermore, by targeting consumers with personal offers, recommendations, advertisements, promotions and services, businesses create customer loyalty.

13.7 Session Tracking

- Websites can use sophisticated technology to allow visitors to customize home pages to suit their individual needs and preferences.
- Similarly, online shopping sites often store personal information for customers, tailoring notifications and special offers to their interests.
- Such services encourage customers to visit sites more frequently and make purchases more regularly.

13.7 Session Tracking

- Privacy
  - A trade-off exists between personalized business service and protection of privacy.
  - Some consumers embrace tailored content, but others fear the possible adverse consequences if the info they provide to businesses is released or collected by tracking technologies.
  - Consumers and privacy advocates ask:
  - What if the business to which we give personal data sells or gives that information to another organization without our knowledge?

13.7 Session Tracking

- What if we do not want our actions on the Internet—a supposedly anonymous medium—to be tracked and recorded by unknown parties?
- What if unauthorized parties gain access to sensitive private data, such as credit-card numbers or medical history?
- These are questions that must be addressed by programmers, consumers, businesses and lawmakers alike.

13.7 Session Tracking

- Recognizing Clients
  - To provide personalized services to consumers, businesses must be able to recognize clients when they request information from a site.
  - As we have discussed, the request/response system on which the web operates is facilitated by HTTP.
  - Unfortunately, HTTP is a stateless protocol—it does not provide information that would enable web servers to maintain state information regarding particular clients.

13.7 Session Tracking

- This means that web servers cannot determine whether a request comes from a particular client or whether the same or different clients generate a series of requests.
- To circumvent this problem, sites can provide mechanisms by which they identify individual clients.
- A session represents a unique client on a website.
- If the client leaves a site and then returns later, the client will still be recognized as the same user.

13.7 Session Tracking

- When the user closes the browser, the session ends.
- To help the server distinguish among clients, each client must identify itself to the server.
- Tracking individual clients is known as session tracking.
- One popular session-tracking technique uses cookies (discussed in Section 13.7.1); another uses ASP.NET's `HttpSessionState` object (used in Section 13.7.1).
- Additional session-tracking techniques are beyond this book's scope.

13.7.1 Cookies

- Cookies provide you with a tool for personalizing web pages.
- A cookie is a piece of data stored by web browsers in a small text file on the user's computer.
- A cookie maintains information about the client during and between browser sessions.
- The first time a user visits the website, the user's computer might receive a cookie from the server; this cookie is then reactivated each time the user revisits that site.
- The collected information is intended to be an anonymous record containing data that is used to personalize the user's future visits to the site.

13.7.1 Cookies

- For example, cookies in a shopping application might store unique identifiers for users.
- When a user adds items to an online shopping cart or performs another task resulting in a request to the web server, the server receives a cookie containing the user's unique identifier.
- The server then uses the unique identifier to locate the shopping cart and perform any necessary processing.
- In addition to identifying users, cookies also can indicate users' shopping preferences.

13.7.1 Cookies

- When a Web Form receives a request from a client, the Web Form can examine the cookie(s) it sent to the client during previous communications, identify the user's preferences and immediately display products of interest to the client.
- Every HTTP-based interaction between a client and a server includes a header containing information either about the request (when the communication is from the client to the server) or about the response (when the communication is from the server to the client).

13.7.1 Cookies

- When a Web Form receives a request, the header includes information such as the request type and any cookies that have been sent previously from the server to be stored on the client machine.
- When the server formulates its response, the header information contains any cookies the server wants to store on the client computer and other information, such as the MIME type of the response.
- The expiration date of a cookie determines how long the cookie remains on the client's computer.

13.7.1 Cookies

- If you do not set an expiration date for a cookie, the web browser maintains the cookie for the duration of the browsing session.
- Otherwise, the web browser maintains the cookie until the expiration date occurs.
- Cookies are deleted when they expire.

Portability Tip 13.1

Users may disable cookies in their web browsers to help ensure their privacy. Such users will experience difficulty using web applications that depend on cookies to maintain state information.

13.7.1 Session Tracking with `HttpSessionState`

- The next web application demonstrates session tracking using the .NET class HttpSessionState.
- When you execute this application, the `Options.aspx` page (Fig. 13.27(a)), which is the application's Start Page, allows the user to select a programming language from a group of radio buttons.
- When the user clicks Submit, the selection is sent to the web server for processing.

13.7.1 Cookies

- The web server uses an `HttpSessionState` object to store the chosen language and the ISBN number for one of our books on that topic.
- Each user that visits the site has a unique `HttpSessionState` object, so the selections made by one user are maintained separately from all other users.
- After storing the selection, the server returns the page to the browser (Fig. 13.27(b)) and displays the user's selection and some information about the user's unique session (which we show just for demonstration purposes).

13.7.1 Cookies

- The page also includes links that allow the user to choose between selecting another programming language or viewing the `Recommendations.aspx` page (Fig. 13.27(e)), which lists recommended books pertaining to the programming language(s) that the user selected previously.
- If the user clicks the link for book recommendations, the information stored in the user's unique `HttpSessionState` object is read and used to form the list of recommendations.

13.7.2 Session Tracking with `HttpSessionState`

- To test this application:
  - Select Open Web Site… from the File menu.
  - In the Open Web Site dialog, ensure that File System is selected, then navigate to this chapter's examples, select the `Sessions` folder and click the Open `Button`.
  - Select `Options.aspx` in the Solution Explorer, then type *Ctrl + F5* to execute the web application in your default web browser.
  -

a) User selects a language from the `Options.aspx` page, then presses **Submit** to send the selection to the server

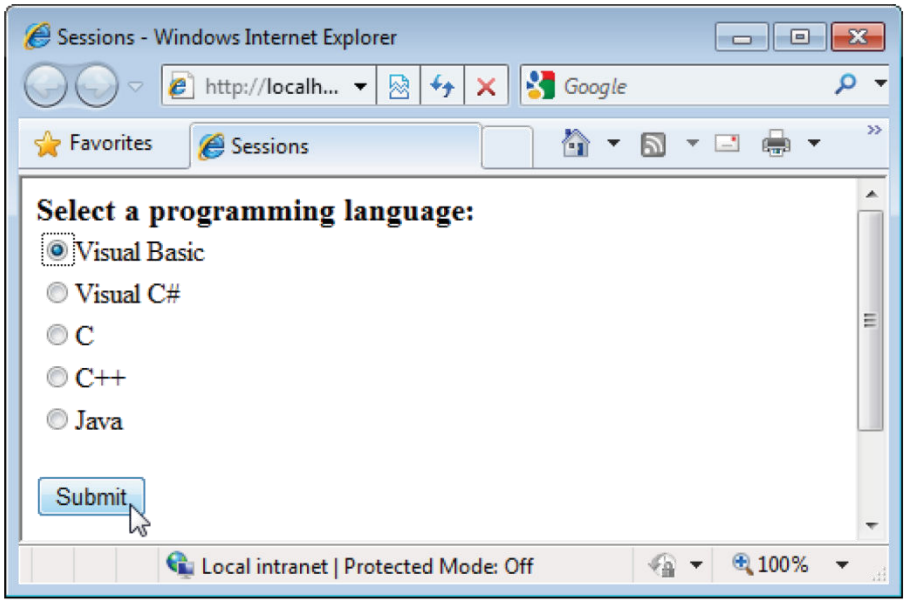

Fig. 13.27 | ASPX file that presents a list of programming languages. (Part 1 of 5.)

b) `Options.aspx` page is updated to hide the controls for selecting a language and to display the user's selection; the user clicks the hyperlink to return to the list of languages and make another selection

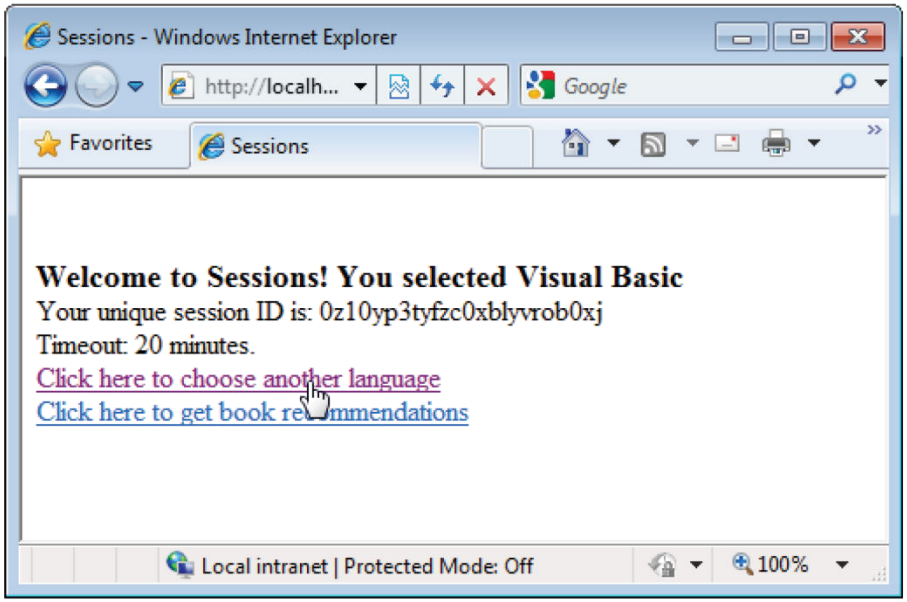

Fig. 13.27 | ASPX file that presents a list of programming languages. (Part 2 of 5.)

c) User selects another language from the `Options.aspx` page, then presses **Submit** to send the selection to the server

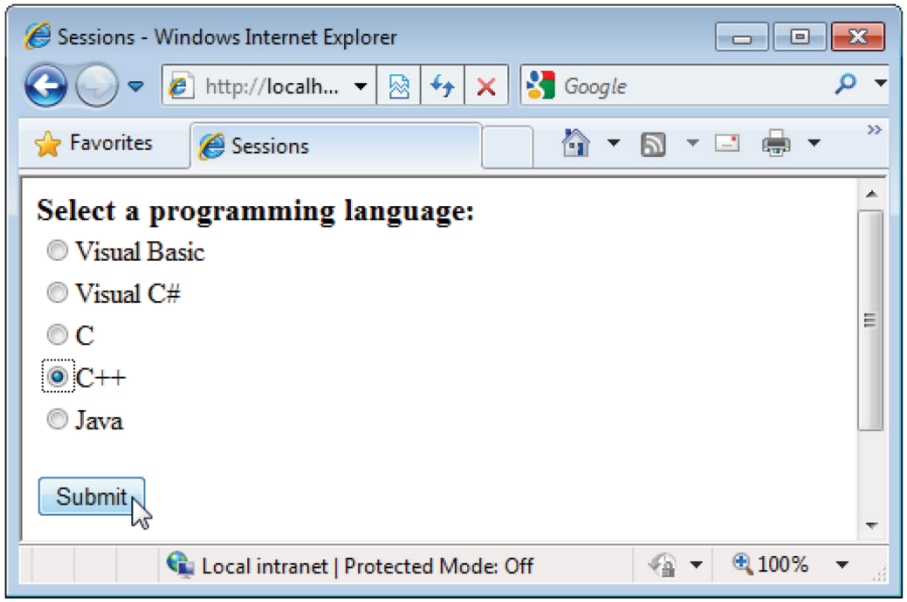

Fig. 13.27 | ASPX file that presents a list of programming languages. (Part 3 of 5.)

d) `Options.aspx` page is updated to hide the controls for selecting a language and to display the user's selection; the user clicks the hyperlink to get a list of book recommendations

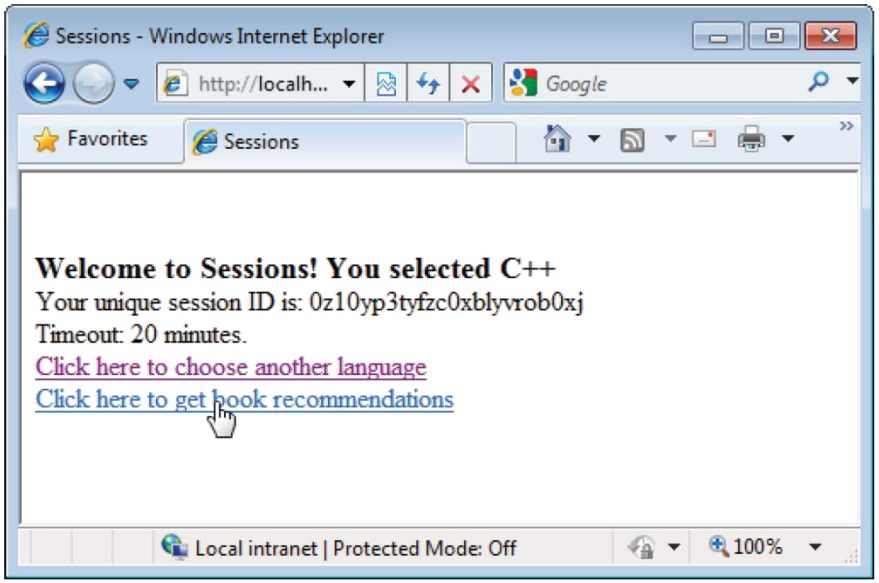

Fig. 13.27 | ASPX file that presents a list of programming languages. (Part 4 of 5.)

e) `Recommendations.aspx` displays the list of recommended books based on the user's selections

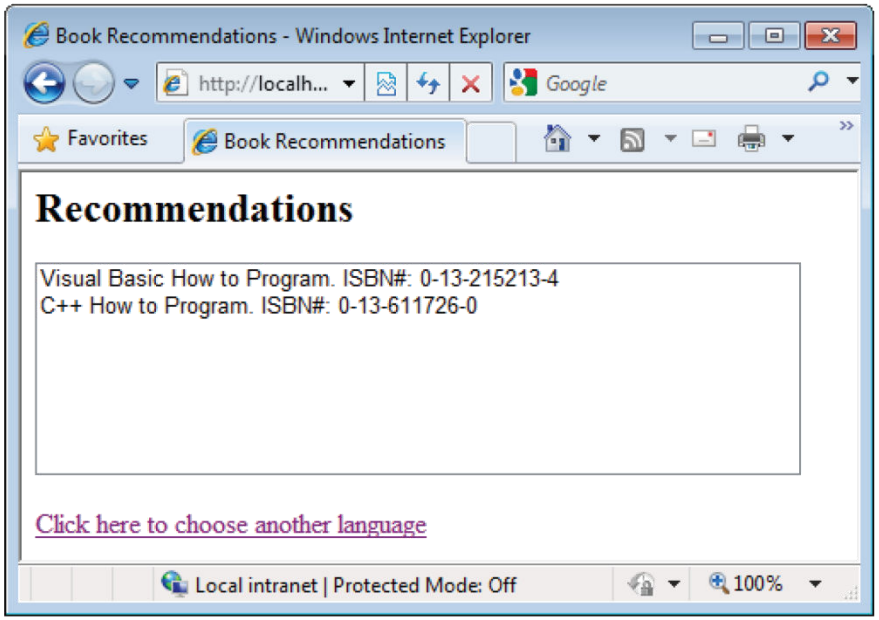

Fig. 13.27 | ASPX file that presents a list of programming languages. (Part 5 of 5.)

13.7.2 Session Tracking with `HttpSessionState`

- Creating the Web Site
  - To begin, follow the steps in Section 13.4.1 to create an Empty Web Site named `Sessions`, then add two Web Forms named `Options.aspx` and `Recommendations.aspx` to the project.
  - Set the `Options.aspx` document's `Title` property to `"Sessions"` and the `Recommendations.aspx` document's `Title` property to `"Book Recommendations"`.
  - To ensure that `Options.aspx` is the first page to load for this application, right click it in the Solution Explorer and select Set As Start Page.

13.7.3 Options.aspx: Selecting a Programming Language

- The `Options.aspx` page Fig. 13.27(a) contains the following controls arranged vertically:
  - A `Label` with its `(ID)` property set to `promptLabel` and its `Text` property set to `"Select a programming language:"`. We used the techniques shown in *Step 5* of Section 13.4.1 to create a CSS style for this label named `.labelStyle`, and set the style's `font-size` attribute to `large` and the `font-weight` attribute to `bold`.

13.7.3 Options.aspx: Selecting a Programming Language

- The user selects a programming language by clicking one of the radio buttons in a `RadioButtonList`. Each radio button has a `Text` property and a `Value` property. The `Text` property is displayed next to the radio button and the `Value` property represents a value that is sent to the server when the user selects that radio button and submits the form. In this example, we'll use the `Value` property to represent the ISBN for the recommended book.

13.7.3 Options.aspx: Selecting a Programming Language

- Create a `RadioButtonList` with its `(ID)` property set to `languageList`. Use the ListItem Collection Editor to add five radio buttons with their `Text` properties set to `Visual Basic`, `Visual C#`, `C`, `C++` and `Java`, and their `Value` properties set to `0-13-215213-4`, `0-13-605322-X`, `0-13-512356-2`, `0-13-611726-0` and `0-13-605306-8`, respectively

13.7.3 Options.aspx: Selecting a Programming Language

- A `Button` with its `(ID)` property set to `submitButton` and its `Text` property set to `Submit`. In this example, we'll handle this `Button`'s `Click` event. You can create its event handler by double clicking the `Button` in Design view.

13.7.3 Options.aspx: Selecting a Programming Language

- A `Label` with its `(ID)` property set to `responseLabel` and its `Text` property set to `"Welcome to Sessions!"`. This `Label` should be placed immediately to the right of the `Button` so that the `Label` appears at the top of the page when we hide the preceding controls on the page. Reuse the CSS style you created in *Step 1* by setting this `Label`'s `CssClass` property to `labelStyle`.

13.7.3 Options.aspx: Selecting a Programming Language

- Two more `Label`s with their `(ID)` properties set to `idLabel` and `timeoutLabel`, respectively. Clear the text in each `Label`'s `Text` property—you'll set these programmatically with information about the current user's session.
- A `HyperLink` with its `(ID)` property set to `languageLink` and its `Text` property set to `"Click here to choose another language"`. Set its `NavigateUrl` property by clicking the ellipsis next to the property in the Properties window and selecting `Options.aspx` from the Select URL dialog.

13.7.3 Options.aspx: Selecting a Programming Language

- A `HyperLink` with its `(ID)` property set to `recommendationsLink` and its `Text` property set to `"Click here to get book recommendations"`. Set its `NavigateUrl` property by clicking the ellipsis next to the property in the Properties window and selecting `Recommendations.aspx` from the Select URL dialog.
- Initially, the controls in *Steps 4–7* will not be displayed, so set each control's `Visible` property to `False`.

13.7.3 Options.aspx: Selecting a Programming Language

- `Session` Property of a `Page`
  - Every Web Form includes a user-specific `HttpSessionState` object, which is accessible through property Session of class `Page`.
  - Throughout this section, we use this property to manipulate the current user's `HttpSessionState` object.
  - When a page is first requested, a unique `HttpSessionState` object is created by ASP.NET and assigned to the `Page`'s `Session` property.

13.7.3 Options.aspx: Selecting a Programming Language

- Code-Behind File for `Options.aspx`
  - Fig. 13.28 presents the code-behind file for the `Options.aspx` page.
  - When this page is requested, the `Page_Load` event handler (lines 9–40) executes before the response is sent to the client.
  - Since the first request to a page is not a postback, the code in lines 12–39 does not execute the first time the page loads.

13.7.3 `Options.aspx`: Selecting a Programming Language

- Postback Processing
  - When the user presses Submit, a postback occurs.
  - The form is submitted to the server and the `Page_Load` event handler executes.
  - Lines 15–19 display the controls shown in Fig. 13.27(b) and lines 22–24 hide the controls shown in Fig. 13.27(a).
  - Next, lines 27–32 ensure that the user selected a language and, if so, display a message in the `responseLabel` indicating the selection.
  - Otherwise, the message `"You did not select a language"` is displayed.

```
1  ' Fig. 13.28: Options.aspx.vb
2  ' Process user's selection of a programming language by displaying
3  ' links and writing information in an HttpSessionState object.
4  Partial Class Options
5     Inherits System.Web.UI.Page
6
7     ' if postback, hide form and display links to make additional
8     ' selections or view recommendations
9     Protected Sub Page_Load(ByVal sender As Object,
10       ByVal e As System.EventArgs) Handles Me.Load
11
12       If IsPostBack Then
13          ' user has submitted information, so display message
14          ' and appropriate hyperlinks
15          responseLabel.Visible = True
16          idLabel.Visible = True
17          timeoutLabel.Visible = True
18          languageLink.Visible = True
19          recommendationsLink.Visible = True
20
```

Fig. 13.28 | Process user's selection of a programming language by displaying links and writing information in an `HttpSessionState` object. (Part 1 of 3.)

```
21          ' hide other controls used to make language selection
22          promptLabel.Visible = False
23          languageList.Visible = False
24          submitButton.Visible = False
25
26          ' if the user made a selection, display it in responseLabel
27          If languageList.SelectedItem IsNot Nothing Then
28             responseLabel.Text &= " You selected " &
29                languageList.SelectedItem.Text
30          Else
31             responseLabel.Text &= "You did not select a language."
32          End If
33
34          ' display session ID
35          idLabel.Text = "Your unique session ID is: " & Session.SessionID
36
37          ' display the timeout
38          timeoutLabel.Text = "Timeout: " & Session.Timeout & " minutes."
39       End If
40    End Sub ' Page_Load
41
```

Fig. 13.28 | Process user's selection of a programming language by displaying links and writing information in an `HttpSessionState` object. (Part 2 of 3.)

```
42      ' record the user's selection in the Session
43      Protected Sub submitButton_Click(ByVal sender As Object,
44         ByVal e As System.EventArgs) Handles submitButton.Click
45
46         ' if the user made a selection
47         If languageList.SelectedItem IsNot Nothing Then
48            ' add name/value pair to Session
49            Session.Add(languageList.SelectedItem.Text,
50               languageList.SelectedItem.Value)
51         End If
52      End Sub ' submitButton_Click
53   End Class ' Options
```

Fig. 13.28 | Process user's selection of a programming language by displaying links and writing information in an `HttpSessionState` object. (Part 3 of 3.)

13.7.3 Options.aspx: Selecting a Programming Language

- The ASP.NET application contains information about the `HttpSessionState` object (`Session`) for the current client.
- Property SessionID (displayed in line 35) contains the unique session ID—a sequence of random letters and numbers.
- The first time a client connects to the web server, a unique session ID is created for that client and a temporary cookie is written to the client so the server can identify the client on subsequent requests.
- When the client makes additional requests, the client's session ID from that temporary cookie is compared with the session IDs stored in the web server's memory to retrieve the client's `HttpSessionState` object.

13.7.3 Options.aspx: Selecting a Programming Language

- `HttpSessionState` property Timeout (displayed in line 38) specifies the maximum amount of time that an `HttpSessionState` object can be inactive before it's discarded.
- By default, if the user does not interact with this web application for 20 minutes, the `HttpSessionState` object is discarded by the server and a new one will be created if the user interacts with the application again.
- Figure 13.29 lists some common `HttpSessionState` properties.

Properties	Description
`Count`	Specifies the number of key/value pairs in the `Session` object.
`IsNewSession`	Indicates whether this is a new session (that is, whether the session was created during loading of this page).
`Keys`	Returns a collection containing the `Session` object's keys.
`SessionID`	Returns the session's unique ID.
`Timeout`	Specifies the maximum number of minutes during which a session can be inactive (that is, no requests are made) before the session expires. By default, this property is set to 20 minutes.

Fig. 13.29 | `HttpSessionState` properties.

13.7.3 Options.aspx: Selecting a Programming Language

- Method `submitButton_Click`
  - In this example, we wish to store the user's selection in an `HttpSessionState` object when the user clicks the Submit `Button`.
  - The `submitButton_Click` event handler (lines 43–52) adds a key/value pair to the `HttpSessionState` object for the current user, specifying the language chosen and the ISBN number for a book on that language.
  - The `HttpSessionState` object is a dictionary—a data structure that stores key/value pairs.

13.7.3 `Options.aspx`: Selecting a Programming Language

- A program uses the key to store and retrieve the associated value in the dictionary.
- We cover dictionaries in more depth in the online Collections chapter.
- The key/value pairs in an `HttpSessionState` object are often referred to as session items.
- They're placed in an `HttpSessionState` object by calling its Add method.

13.7.3 `Options.aspx`: Selecting a Programming Language

- If the user made a selection (line 47), lines 49–50 get the selection and its corresponding value from the `languageList` by accessing its `SelectedItem`'s `Text` and `Value` properties, respectively, then call `HttpSessionState` method `Add` to add this name/value pair as a session item in the `HttpSessionState` object (`Session`).

13.7.3 Options.aspx: Selecting a Programming Language

- If the application adds a session item that has the same name as an item previously stored in the `HttpSessionState` object, the session item is replaced—the names in session items must be unique.
- Another common syntax for placing a session item in the `HttpSessionState` object is `Session(`*Name*`) = Value`.
- For example, we could have replaced lines 49–50 with

```
Session(languageList.SelectedItem.Text) =
   languageList.SelectedItem.Value
```

Software Engineering Observation 13.1

A Web Form should not use instance variables to maintain client state information, because each new request or postback is handled by a new instance of the page. Instead, maintain client state information in `HttpSessionState` objects, because such objects are specific to each client.

Software Engineering Observation 13.2

A benefit of using `HttpSessionState` objects (rather than cookies) is that `HttpSessionState` objects can store any type of object (not just `String`s) as attribute values. This provides you with increased flexibility in determining the type of state information to maintain for clients.

13.7.4 Recommendations.aspx: Displaying Recommendations Based on Session Values

- After the postback of `Options.aspx`, the user may request book recommendations.
- The book-recommendations hyperlink forwards the user to the page `Recommendations.aspx` (Fig. 13.27(e)) to display the recommendations based on the user's language selections.

13.7.4 Recommendations.aspx: Displaying Recommendations Based on Session Values

- The page contains the following controls arranged vertically:
  - A `Label` with its `(ID)` property set to `recommendationsLabel` and its `Text` property set to `"Recommendations:"`. We created a CSS style for this label named `.labelStyle`, and set the `font-size` attribute to `x-large` and the `font-weight` attribute to `bold`. (See *Step 5* in Section 13.4.1 for information on creating a CSS style.)

13.7.4 Recommendations.aspx: Displaying Recommendations Based on Session Values

- A `ListBox` with its `(ID)` property set to `booksListBox`. We created a CSS style for this label named `.listBoxStyle`. In the `Position` category, we set the `width` attribute to `450px` and the `height` attribute to `125px`. The `px` indicates that the measurement is in pixels.

13.7.4 Recommendations.aspx: Displaying Recommendations Based on Session Values

- A `HyperLink` with its `(ID)` property set to `languageLink` and its `Text` property set to `"Click here to choose another language"`. Set its `NavigateUrl` property by clicking the ellipsis next to the property in the Properties window and selecting `Options.aspx` from the Select URL dialog. When the user clicks this link, the `Options.aspx` page will be reloaded. Requesting the page in this manner is not considered a postback, so the original form in Fig. 13.27(a) will be displayed.

13.7.4 Recommendations.aspx: Displaying Recommendations Based on Session Values

- Code-Behind File for `Recommendations.aspx`
  - Figure 13.30 presents the code-behind file for `Recommendations.aspx`.
  - Event handler `Page_Init` (lines 7–27) retrieves the session information.
  - If a user has not selected a language in the `Options.aspx` page, the `HttpSessionState` object's Count property will be `0` (line 11).
  - This property provides the number of session items contained in a `HttpSessionState` object.
  - If the `Count` is `0`, then we display the text `No Recommendations` (line 20), clear the `ListBox` and hide it (lines 21–22), and update the `Text` of the `HyperLink` back to `Options.aspx` (line 25).

```
1  ' Fig. 13.30: Recommendations.aspx.vb
2  ' Creates book recommendations based on a Session object.
3  Partial Class Recommendations
4     Inherits System.Web.UI.Page
5
6     ' read Session items and populate ListBox with any book recommendations
7     Protected Sub Page_Init(ByVal sender As Object,
8        ByVal e As System.EventArgs) Handles Me.Init
9
10       ' determine whether Session contains any information
11       If Session.Count <> 0 Then
12          For Each keyName In Session.Keys
13             ' use keyName to display one of Session's name/value pairs
14             booksListBox.Items.Add(keyName &
15                " How to Program. ISBN#: " & Session(keyName))
16          Next
17       Else
18          ' if there are no session items, no language was chosen, so
19          ' display appropriate message and clear and hide booksListBox
20          recommendationsLabel.Text = "No Recommendations"
21          booksListBox.Items.Clear()
22          booksListBox.Visible = False
```

Fig. 13.30 | Session data used to provide book recommendations to the user. (Part 1 of 2.)

```
23
24             ' modify languageLink because no language was selected
25             languageLink.Text = "Click here to choose a language"
26          End If
27       End Sub ' Page_Init
28    End Class ' Recommendations
```

Fig. 13.30 | Session data used to provide book recommendations to the user. (Part 2 of 2.)

13.7.4 Recommendations.aspx: Displaying Recommendations Based on Session Values

- If the user chose at least one language, the loop in lines 12–16 iterates through the `HttpSessionState` object's keys (line 12) by accessing the `HttpSessionState`'s Keys property, which returns a collection containing all the keys in the session.
- Lines 14–15 concatenate the `keyName`, the `String` `" How to Program`.
- `ISBN#: "` and the key's corresponding value, which is returned by `Session(keyName)`.
- This `String` is the recommendation that is added to the `ListBox`.

13.8 Case Study: Database-Driven ASP.NET Guestbook

- Many websites allow users to provide feedback about the website in a guestbook.
- Typically, users click a link on the website's home page to request the guestbook page.
- This page usually consists of a form that contains fields for the user's name, e-mail address, message/feedback and so on.
- Data submitted on the guestbook form is then stored in a database located on the server.
- In this section, we create a guestbook Web Form application.

13.8 Case Study: Database-Driven ASP.NET Guestbook

- The GUI (Fig. 13.31) contains a GridView data control, which displays all the entries in the guestbook in tabular format.
- This control is located in the Toolbox's Data section.
- We explain how to create and configure this data control shortly.
- The `GridView` displays abc in Design mode to indicate data that will be retrieved from a data source at runtime.
- You'll learn how to create and configure the `GridView` shortly.

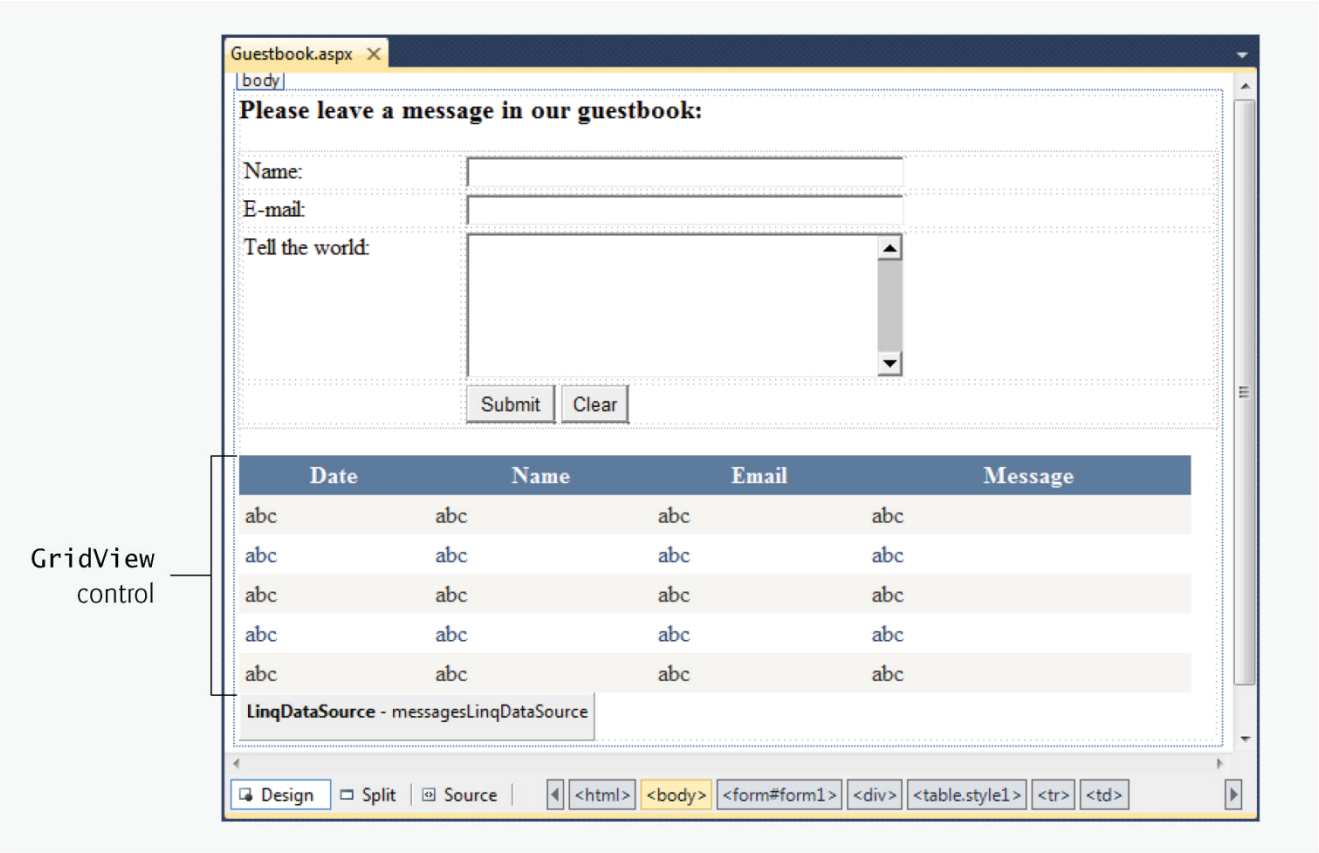

Fig. 13.31 | Guestbook application GUI in Design mode.

13.8 Case Study: Database-Driven ASP.NET Guestbook

- The Guestbook Database
  - The application stores the guestbook information in a SQL Server database called `Guestbook.mdf` located on the web server.
  - (We provide this database in the `databases` folder with this chapter's examples.) The database contains a single table named `Messages`.

13.8 Case Study: Database-Driven ASP.NET Guestbook

- Testing the Application
  - To test this application: Select Open Web Site… from the File menu.
  - In the Open Web Site dialog, ensure that File System is selected, then navigate to this chapter's examples, select the `Guestbook` folder and click the Open `Button`.
  - Select `Guestbook.aspx` in the Solution Explorer, then type *Ctrl* + *F5* to execute the web application in your default web browser.
- Figure 13.32(a) shows the user submitting a new entry.
- Figure 13.32(b) shows the new entry as the last row in the `GridView`.

a) User enters data for the name, e-mail and message, then presses Submit to send the data to the server

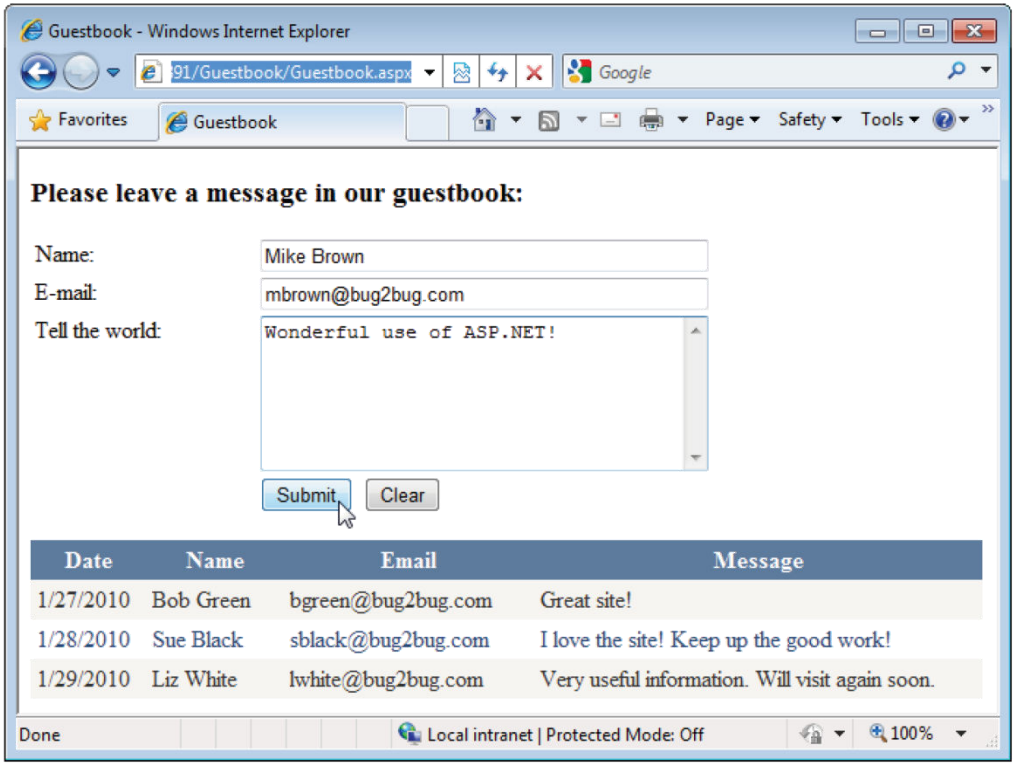

Fig. 13.32 | Sample execution of the Guestbook application.

b) Server stores the data in the database, then refreshes the `GridView` with the updated data

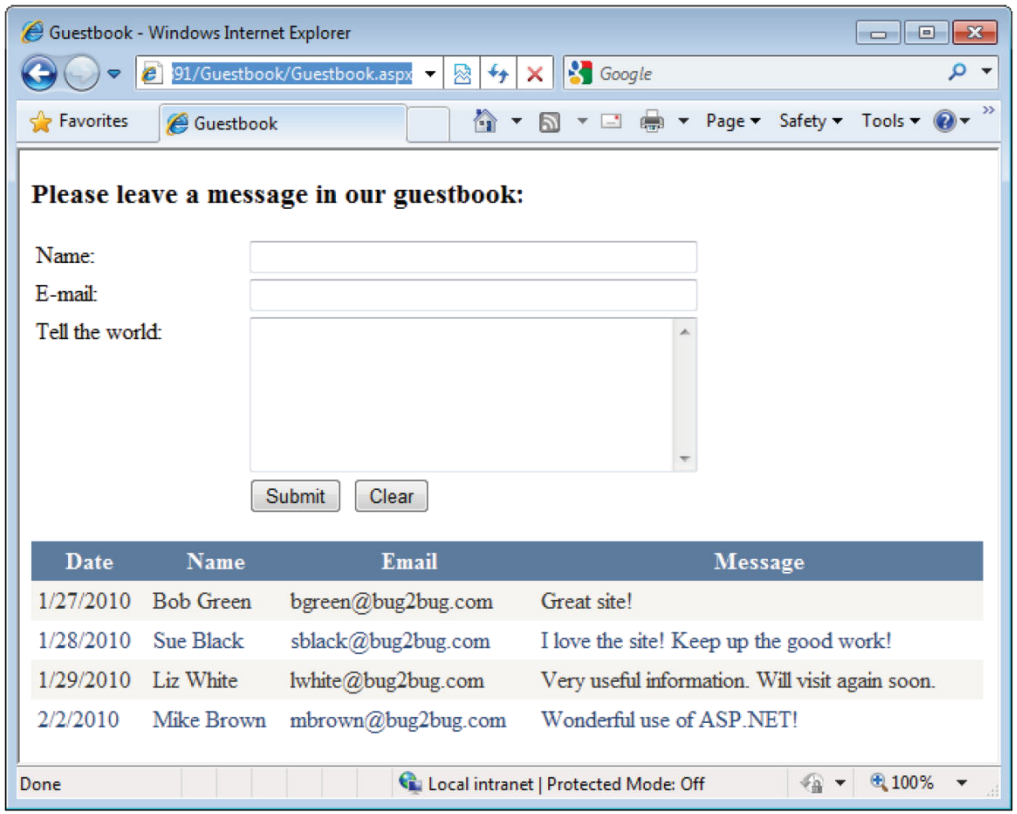

Fig. 13.32 | Sample execution of the Guestbook application.

13.8.1 Building a Web Form that Displays Data from a Database

- We now explain how to build this GUI and set up the data binding between the `GridView` control and the database.
- Many of these steps are similar to those performed in Chapter 12 to access and interact with a database in a Windows application.
- We discuss the code-behind file in Section 13.8.2.
- To build the guestbook application, perform the following steps:

13.8.1 Building a Web Form that Displays Data from a Database

- Step 1: Creating the Web Site
  - To begin, follow the steps in Section 13.4.1 to create an Empty Web Site named `Guestbook` then add a Web Form named `Guestbook.aspx` to the project.
  - Set the document's `Title` property to `"Guestbook"`.
  - To ensure that `Guestobook.aspx` loads when you execute this application, right click it in the Solution Explorer and select Set As Start Page.

13.8.1 Building a Web Form that Displays Data from a Database

- Step 2: Creating the Form for User Input
  - In Design mode, add the text `Please leave a message in our guestbook:`, then use the Block Format `ComboBox` in the IDE's toolbar to change the text to Heading 3 format.
  - Insert a table with four rows and two columns, configured so that the text in each cell aligns with the top of the cell.
  - Place the appropriate text (see Fig. 13.31) in the top three cells in the table's left column.
  - Then place `TextBox`es named `nameTextBox`, `emailTextBox` and `messageTextBox` in the top three table cells in the right column.

13.8.1 Building a Web Form that Displays Data from a Database

- Configure the `TextBox`es as follows:
  - Set the `nameTextBox`'s width to `300px`.
  - Set the `emailTextBox`'s width to `300px`.
  - Set the `messageTextBox`'s width to `300px` and height to `100px`. Also set this control's `TextMode` property to `MultiLine` so the user can type a message containing multiple lines of text.

13.8.1 Building a Web Form that Displays Data from a Database

- Finally, add `Button`s named `submitButton` and `clearButton` to the bottom-right table cell.
- Set the buttons' `Text` properties to `Submit` and `Clear`, respectively.
- We discuss the buttons' event handlers when we present the code-behind file.
- You can create these event handlers now by double clicking each `Button` in Design view.

13.8.1 Building a Web Form that Displays Data from a Database

- Step 3: Adding a `GridView` Control to the Web Form
  - Add a `GridView` named `messagesGridView` that will display the guestbook entries.
  - This control appears in the Data section of the Toolbox.
  - The colors for the `GridView` are specified through the Auto Format...
  - link in the GridView Tasks smart-tag menu that opens when you place the `GridView` on the page.
  - Clicking this link displays an AutoFormat dialog with several choices.
  - In this example, we chose Professional.
  - We show how to set the `GridView`'s data source (that is, where it gets the data to display in its rows and columns) shortly.

13.8.1 Building a Web Form that Displays Data from a Database

- Step 4: Adding a Database to an ASP.NET Web Application
  - To use a SQL Server Express database file in an ASP.NET web application, you must first add the file to the project's `App_Data` folder.
  - For security reasons, this folder can be accessed only by the web application on the server—clients cannot access this folder over a network.

13.8.1 Building a Web Form that Displays Data from a Database

- The web application interacts with the database on behalf of the client.
- The Empty Web Site template does not create the `App_Data` folder.
- To create it, right click the project's name in the Solution Explorer, then select Add ASP.NET Folder > App_Data.
- Next, add the `Guestbook.mdf` file to the `App_Data` folder.

13.8.1 Building a Web Form that Displays Data from a Database

- You can do this in one of two ways:
  - Drag the file from Windows Explorer and drop it on the `App_Data` folder.
  - Right click the `App_Data` folder in the Solution Explorer and select Add Existing Item… to display the Add Existing Item dialog, then navigate to the databases folder with this chapter's examples, select the `Guestbook.mdf` file and click Add. [*Note:* Ensure that Data Files is selected in the `ComboBox` above or next to the Add `Button` in the dialog; otherwise, the database file will not be displayed in the list of files.]

13.8.1 Building a Web Form that Displays Data from a Database

- Step 5: Creating the LINQ to SQL Classes
  - As in Chapter 12, you'll use LINQ to interact with the database.
  - To create the LINQ to SQL classes for the `Guestbook` database:
  - Right click the project in the Solution Explorer and select Add New Item… to display the Add New Item dialog.

13.8.1 Building a Web Form that Displays Data from a Database

- In the dialog, select LINQ to SQL Classes, enter `Guestbook.dbml` as the Name, and click Add. A dialog appears asking if you would like to put your new LINQ to SQL classes in the `App_Code` folder; click Yes. The IDE will create an `App_Code` folder and place the LINQ to SQL classes information in that folder.
- In the Database Explorer window, drag the `Guestbook` database's `Messages` table from the Database Explorer onto the Object Relational Designer. Finally, save your project by selecting File > Save All.

13.8.1 Building a Web Form that Displays Data from a Database

- Step 6: Binding the `GridView` to the `Messages` Table of the `Guestbook` Database
  - You can now configure the `GridView` to display the database's data.
  - Open the GridView Tasks smart-tag menu, then select <New data source...> from the Choose Data Source `ComboBox` to display the Data Source Configuration Wizard dialog.

13.8.1 Building a Web Form that Displays Data from a Database

- In this example, we use a LinqDataSource control that allows the application to interact with the `Guestbook.mdf` database through LINQ. Select LINQ, then set the `ID` of the data source to `messagesLinqDataSource` and click OK to begin the Configure Data Source wizard.
- In the Choose a Context Object screen, ensure that `GuestbookDataContext` is selected in the `ComboBox`, then click Next >.

13.8.1 Building a Web Form that Displays Data from a Database

- The Configure Data Selection screen (Fig. 13.33) allows you to specify which data the `LinqDataSource` should retrieve from the data context. Your choices on this page design a `Select` LINQ query. The Table drop-down list identifies a table in the data context. The `Guestbook` data context contains one table named `Messages`, which is selected by default. If you haven't saved your project since creating your LINQ to SQL classes (Step 5), the list of tables will not appear. In the Select pane, ensure that the checkbox marked with an asterisk (*) is selected to indicate that you want to retrieve all the columns in the `Messages` table.

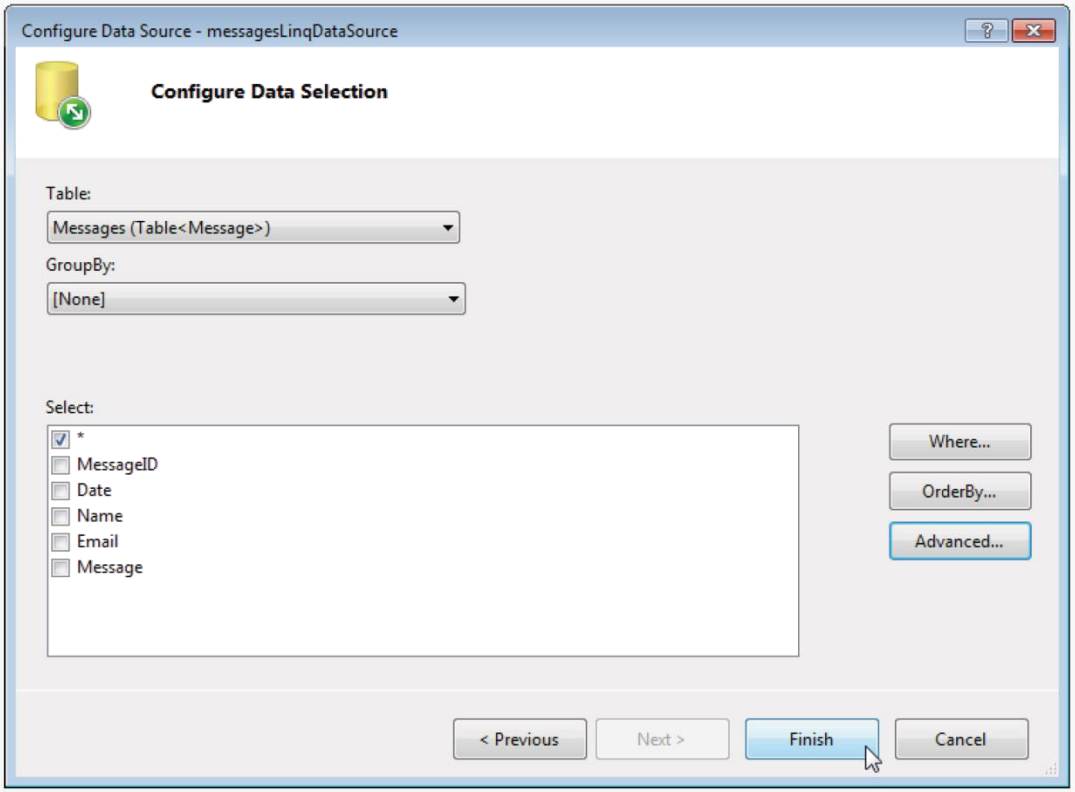

Fig. 13.33 | Configuring the query used by the `LinqDataSource` to retrieve data.

13.8.1 Building a Web Form that Displays Data from a Database

- Click the Advanced… button, then select the Enable the LinqDataSource to perform automatic inserts `CheckBox` and click OK. This configures the `LinqDataSource` control to automatically insert new data into the database when new data is inserted in the data context. We discuss inserting new guestbook entries based on users' form submissions shortly.
- Click Finish to complete the wizard.

13.8.1 Building a Web Form that Displays Data from a Database

- A control named `messagesLinqDataSource` now appears on the Web Form directly below the `GridView` (Fig. 13.34).
- This control is represented in Design mode as a gray box containing its type and name.
- It will *not appear on the web page—the gray box simply provides a way to manipulate the control visually through* Design *mode—similar to how the objects in the component tray are used in* Design *mode for a Windows Forms application.*

13.8.1 Building a Web Form that Displays Data from a Database

- The `GridView` now has column headers that correspond to the columns in the `Messages` table.
- The rows each contain either a number (which signifies an autoincremented column) or `abc` (which indicates string data).
- The actual data from the `Guestbook.mdf` database file will appear in these rows when you view the ASPX file in a web browser.

13.8.1 Building a Web Form that Displays Data from a Database

- Step 7: Modifying the Columns of the Data Source Displayed in the `GridView`
  - It's not necessary for site visitors to see the `MessageID` column when viewing past guestbook entries—this column is merely a unique primary key required by the `Messages` table within the database.
  - So, let's modify the `GridView` to prevent this column from displaying on the Web Form.
  - In the GridView Tasks smart tag menu, click Edit Columns to display the Fields dialog (Fig. 13.35).

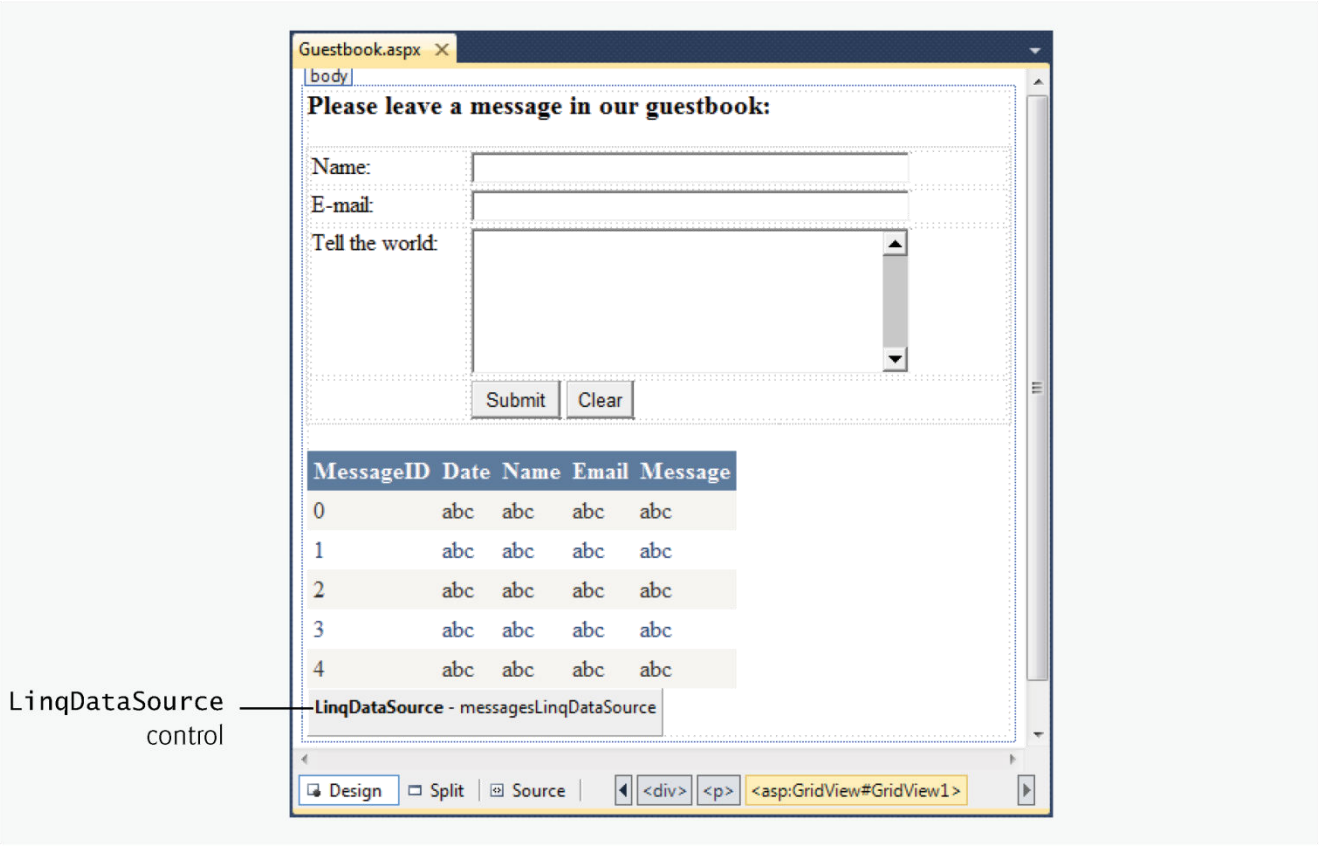

Fig. 13.34 | **Design** mode displaying `LinqDataSource` control for a `GridView`.

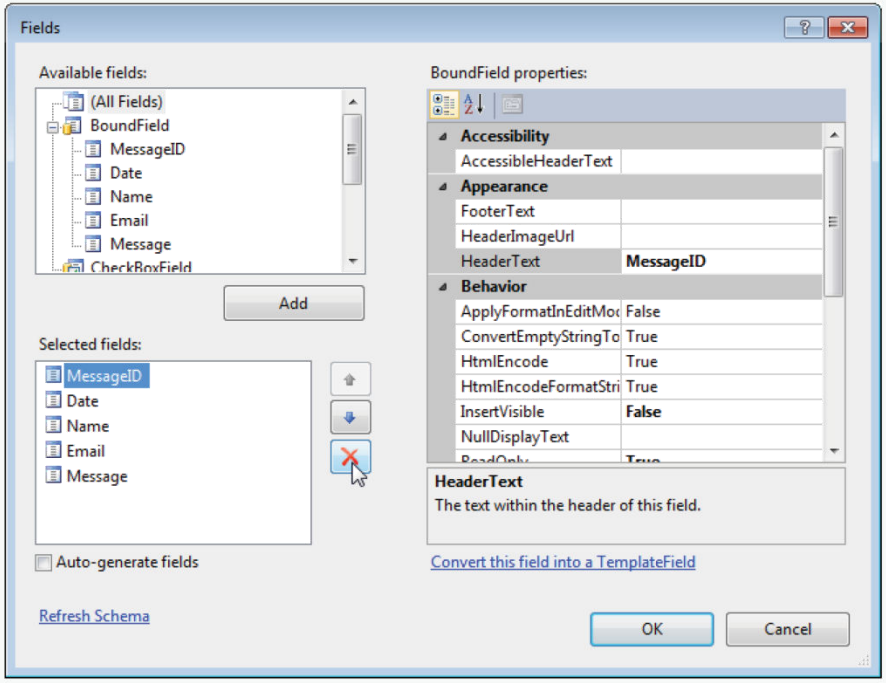

Fig. 13.35 | Removing the `MessageID` column from the `GridView`.

13.8.1 Building a Web Form that Displays Data from a Database

- Select MessageID in the Selected fields pane, then click the `Button`. This removes the `MessageID` column from the `GridView`.
- Click OK to return to the main IDE window, then set the `Width` property of the `GridView` to `650px`.

- The `GridView` should now appear as shown in Fig. 13.31.

13.8.2 Modifying the Code-Behind File for the Guestbook Application

- After building the Web Form and configuring the data controls used in this example, double click the Submit and Clear buttons in Design view to create their corresponding `Click` event handlers in the code-behind file (Fig. 13.36).
- The IDE generates empty event handlers, so we must add the appropriate code to make these buttons work properly.
- The event handler for `clearButton` (lines 36–41) clears each `TextBox` by setting its `Text` property to an empty string.
- This resets the form for a new guestbook submission.

```
1  ' Fig. 13.36: Guestbook.aspx.vb
2  ' Code-behind file that defines event handlers for the guestbook.
3  Partial Class Guestbook
4     Inherits System.Web.UI.Page
5
6     ' Submit Button adds a new guestbook entry to the database,
7     ' clears the form and displays the updated list of guestbook entries
8     Protected Sub submitButton_Click(ByVal sender As Object, _
9        ByVal e As System.EventArgs) Handles submitButton.Click
10
11       ' create dictionary of parameters for inserting
12       Dim insertParameters As New ListDictionary()
13
14       ' add current date and the user's name, e-mail address and message
15       ' to dictionary of insert parameters
16       insertParameters.Add("Date", Date.Now.ToShortDateString())
17       insertParameters.Add("Name", nameTextBox.Text)
18       insertParameters.Add("Email", emailTextBox.Text)
19       insertParameters.Add("Message", messageTextBox.Text)
20
```

Fig. 13.36 | Code-behind file for the guestbook application. (Part 1 of 2.)

```
21          ' execute an INSERT LINQ statement to add a new entry to the
22          ' Messages table in the Guestbook data context that contains the
23          ' current date and the user's name, e-mail address and message
24          messagesLinqDataSource.Insert(insertParameters)
25
26          ' clear the TextBoxes
27          nameTextBox.Text = String.Empty
28          emailTextBox.Text = String.Empty
29          messageTextBox.Text = String.Empty
30
31          ' update the GridView with the new database table contents
32          messagesGridView.DataBind()
33       End Sub ' submitButton_Click
34
35       ' Clear Button clears the Web Form's TextBoxes
36       Protected Sub clearButton_Click(ByVal sender As Object, _
37          ByVal e As System.EventArgs) Handles clearButton.Click
38          nameTextBox.Text = String.Empty
39          emailTextBox.Text = String.Empty
40          messageTextBox.Text = String.Empty
41       End Sub ' clearButton_Click
42    End Class ' Guestbook
```

Fig. 13.36 | Code-behind file for the guestbook application. (Part 2 of 2.)

13.8.3 Modifying the Code-Behind File for the Guestbook Application

- Lines 8–33 contain `submitButton`'s event-handling code, which adds the user's information to the `Guestbook` database's `Messages` table.
- To use the values of the `TextBox`es on the Web Form as the parameter values inserted into the database, we must create a ListDictionary of insert parameters that are key/value pairs.
- Line 12 creates a `ListDictionary` object.

13.8.3 Building a Web Form that Displays Data from a Database

- Lines 16–19 used the `ListDictionary`'s `Add` method to store key/value pairs that represent each of the four insert parameters—the current date and the user's name, e-mail address, and message.
- Invoking the `LinqDataSource` method `Insert` (line 24) inserts the data in the data context, adding a row to the `Messages` table and automatically updating the database.

13.8.3 Building a Web Form that Displays Data from a Database

- We pass the `ListDictionary` object as an argument to the `Insert` method to specify the insert parameters.
- After the data is inserted into the database, lines 27–29 clear the `TextBox`es, and line 32 invokes `messagesGridView`'s DataBind method to refresh the data that the `GridView` displays.
- This causes `messagesLinqDataSource` (the `GridView`'s source) to execute its `Select` command to obtain the `Messages` table's newly updated data.

13.9 Online Case Study: ASP.NET AJAX

- In the online chapter, Web App Development: A Deeper Look, you learn the difference between a traditional web application and an Ajax (Asynchronous JavaScript and XML) web application.
- You also learn how to use ASP.NET AJAX to quickly and easily improve the user experience for your web applications, giving them responsiveness comparable to that of desktop applications.
- To demonstrate ASP.NET AJAX capabilities, you enhance the validation example by displaying the submitted form information without reloading the entire page.
- The only modifications to this web application appear in `Validation.aspx` file.
- You use Ajax-enabled controls to add this feature.

13.10 Online Case Study: Password-Protected Books Database Application

- In the online chapter, Web App Development: A Deeper Look, we include a web application case study in which a user logs into a password-protected website to view a list of publications by a selected author.
- The application consists of several pages and provides website registration and login capabilities.
- You'll learn about ASP.NET master pages, which allow you to specify a common look-and-feel for all the pages in your app.
- We also introduce the Web Site Administration Tool and use it to configure the portions of the application that can be accessed only by users who are logged into the website.