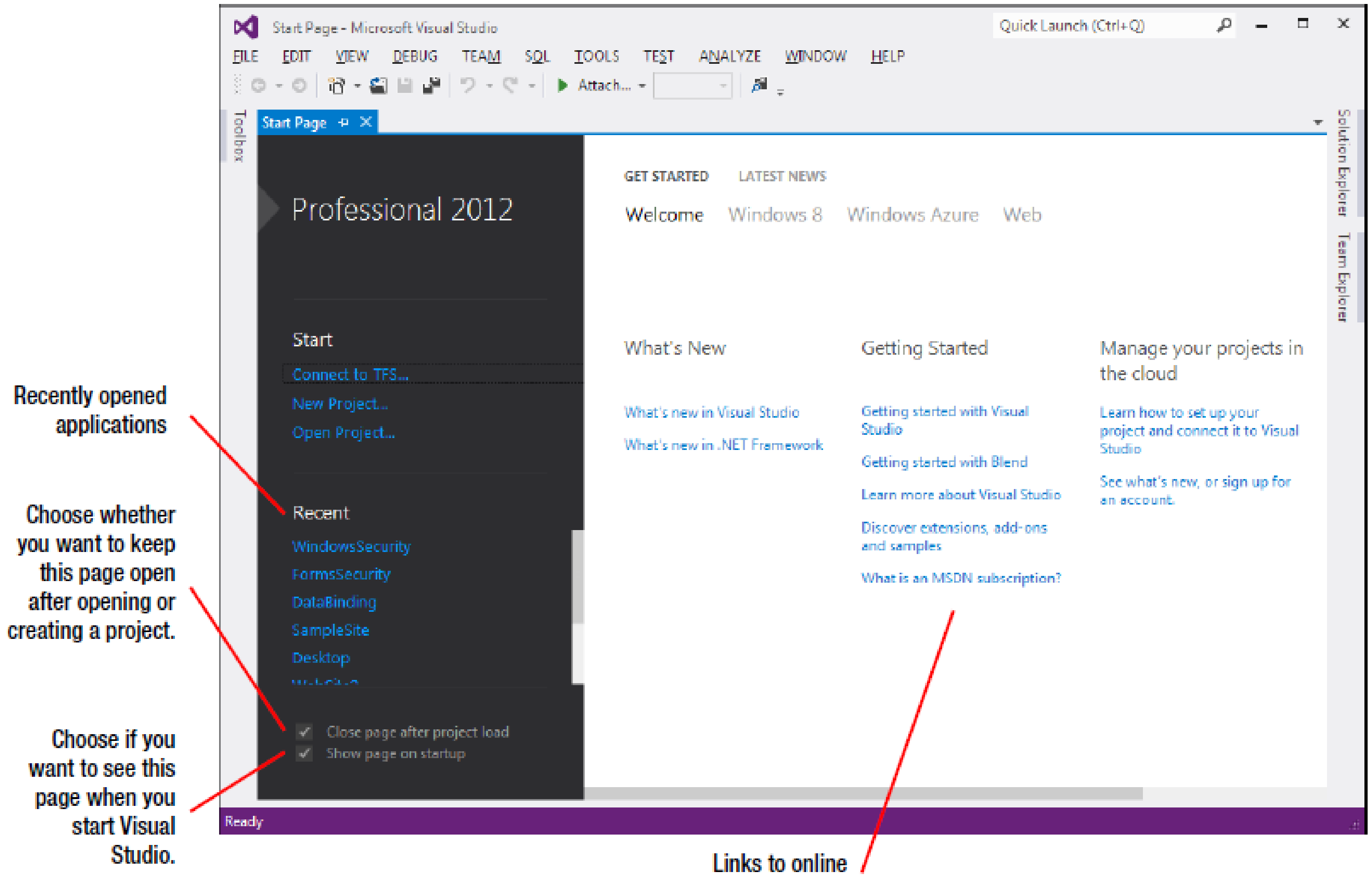
A Tour to Visual Studio

- Creating Websites

Creating Websites

You start Visual Studio by choosing Start ➤ All Programs ➤ Microsoft Visual Studio 2012 ➤ Microsoft Visual Studio 2012. When Visual Studio first loads, it shows its Start page (Figure 4-1).


Start Page - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
FILE EDIT VIEW DEBUG TEAM SQL TOOLS TEST ANALYZE WINDOW HELP
Attach...
Toolbox
Start Page
Professional 2012
Start
Connect to TFS...
New Project...
Open Project...
Recent
WindowsSecurity
FormsSecurity
DataBinding
SampleSite
Desktop
Close page after project load
Show page on startup
GET STARTED
LATEST NEWS
Welcome
Windows 8
Windows Azure
Web
What's New
What's new in Visual Studio
What's new in .NET Framework
Getting Started
Getting started with Visual Studio
Getting started with Blend
Learn more about Visual Studio
Discover extensions, add-ons and samples
What is an MSDN subscription?
Manage your projects in the cloud
Learn how to set up your project and connect it to Visual Studio
See what's new, or sign up for an account.
Solution Explorer
Team Explorer
Ready
Recently opened applications
Choose whether you want to keep this page open after opening or creating a project.
Choose if you want to see this page when you start Visual Studio.
Links to online

Creating an Empty Web Application

To create your first Visual Studio application, follow these steps:

1. Choose File ➤ New ➤ Web Site from the Visual Studio menu. The New Web Site dialog box appears, as shown in Figure 4-2.

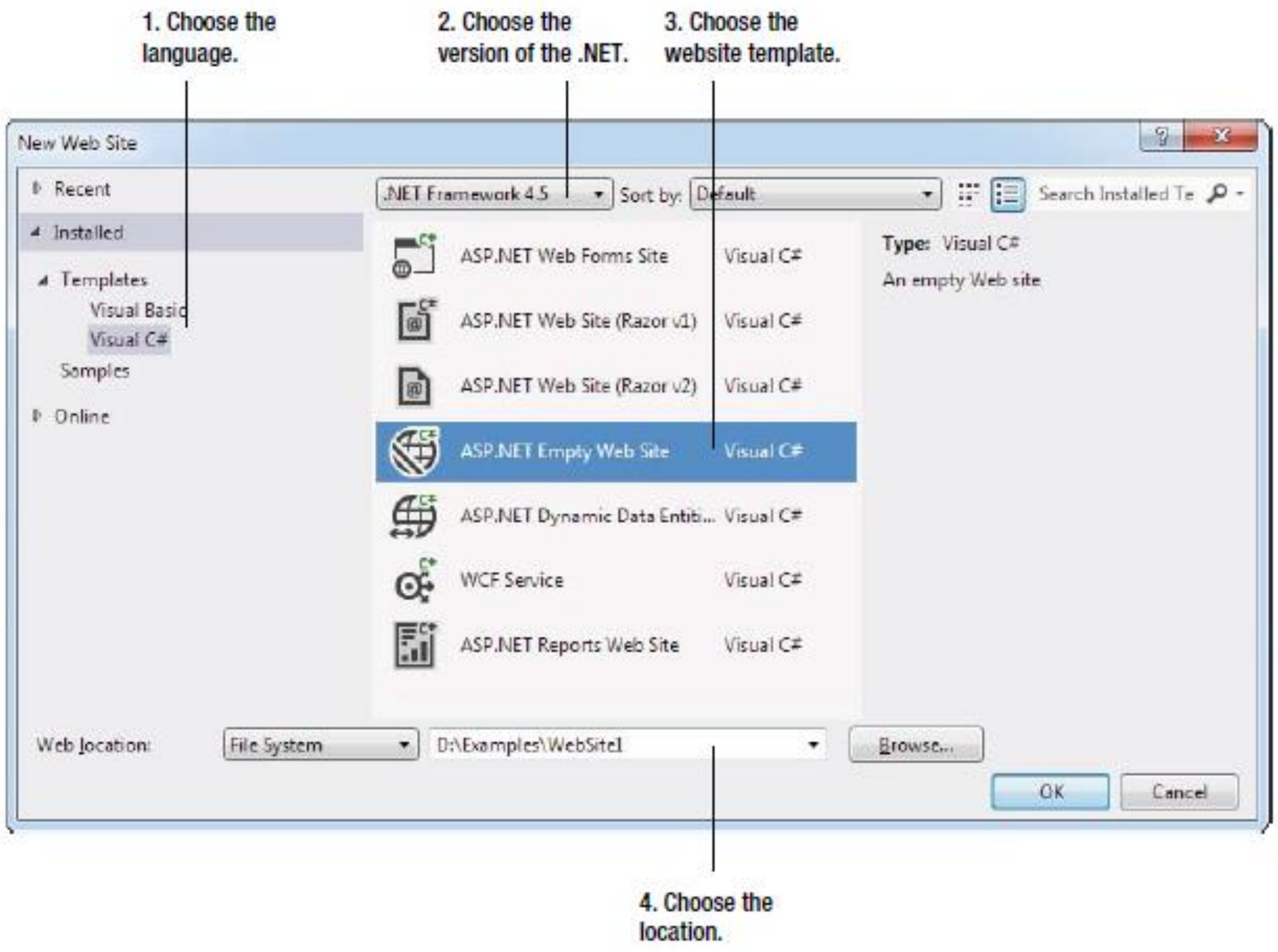
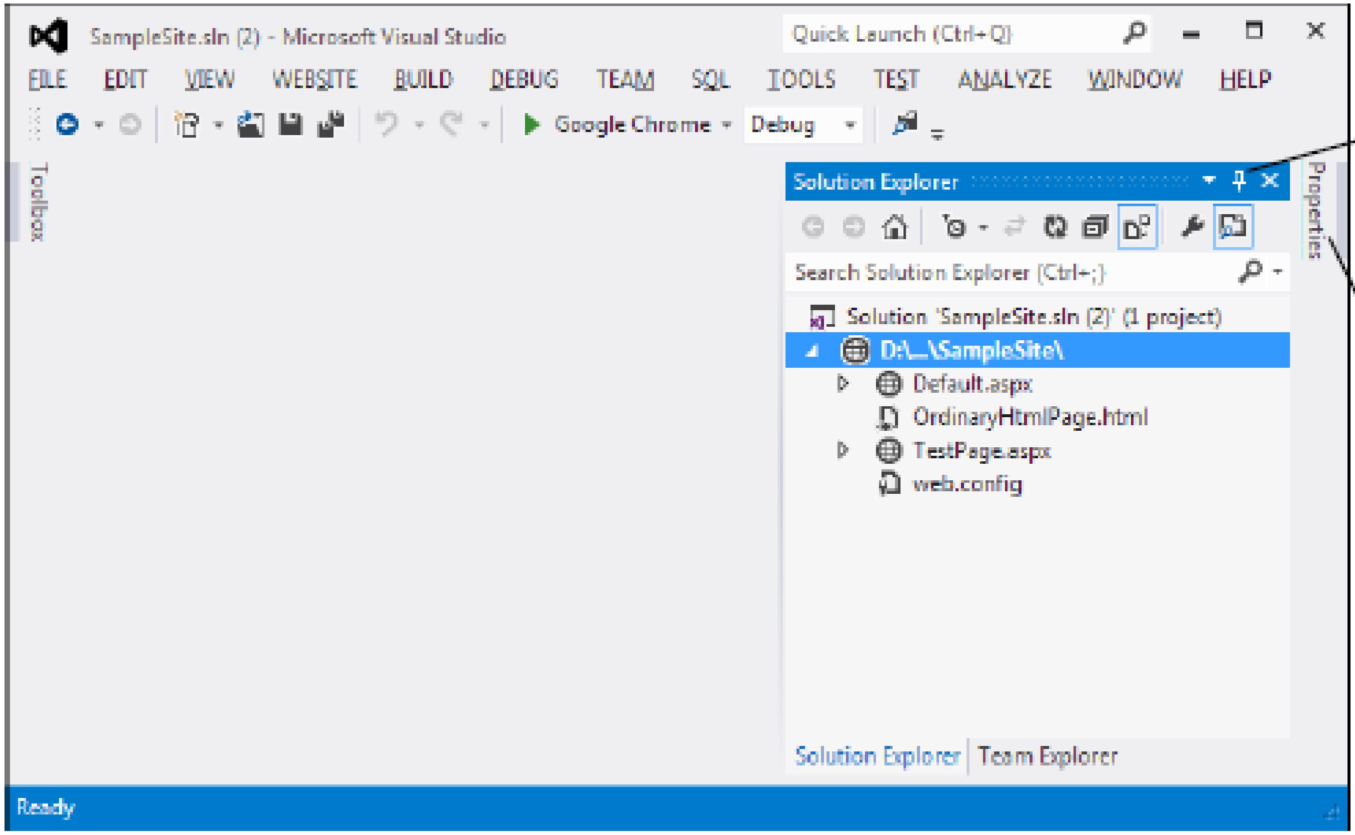

1. Choose the language.
2. Choose the version of the .NET.
3. Choose the website template.
4. Choose the location.
New Web Site
Recent
Installed
Templates
Visual Basic
Visual C#
Samples
Online
.NET Framework 4.5
Sort by: Default
Search Installed Te
ASP.NET Web Forms Site Visual C#
ASP.NET Web Site (Razor v1) Visual C#
ASP.NET Web Site (Razor v2) Visual C#
ASP.NET Empty Web Site Visual C#
ASP.NET Dynamic Data Entiti... Visual C#
WCF Service Visual C#
ASP.NET Reports Web Site Visual C#
Type: Visual C#
An empty Web site
Web location: File System
D:\Examples\WebSite1
Browse...
OK
Cancel

Figure 4-2. The New Web Site dialog box

Using the Solution Explorer

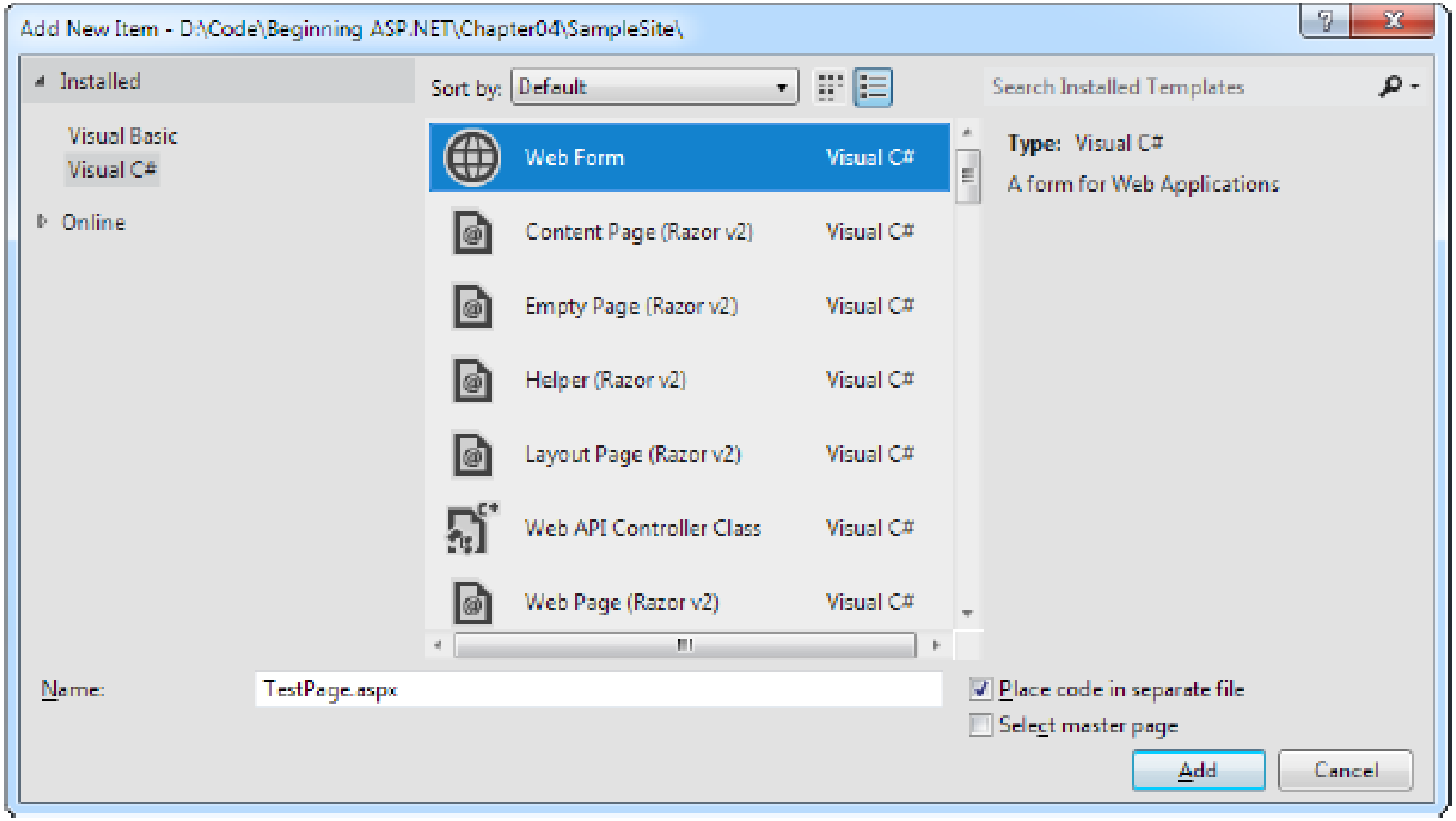

SampleSite.sln (2) - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
FILE EDIT VIEW WEBSITE BUILD DEBUG TEAM SQL TOOLS TEST ANALYZE WINDOW HELP
Google Chrome
Debug
Toolbox
Solution Explorer
Properties
Search Solution Explorer (Ctrl+;)
Solution 'SampleSite.sln (2)' (1 project)
D:\...\SampleSite\
Default.aspx
OrdinaryHtmlPage.html
TestPage.aspx
web.config
Solution Explorer
Team Explorer
Ready

Click the thumbtack to "pin" a window in place.

Click a tab to expand a hidden window.

Figure 4-4. The Solution Explorer

ADDING WEB FORMS



Add New Item - D:\Code\Beginning ASP.NET\Chapter04\SampleSite\

Web Form	Visual C#
Content Page (Razor v2)	Visual C#
Empty Page (Razor v2)	Visual C#
Helper (Razor v2)	Visual C#
Layout Page (Razor v2)	Visual C#
Web API Controller Class	Visual C#
Web Page (Razor v2)	Visual C#

Type: Visual C#

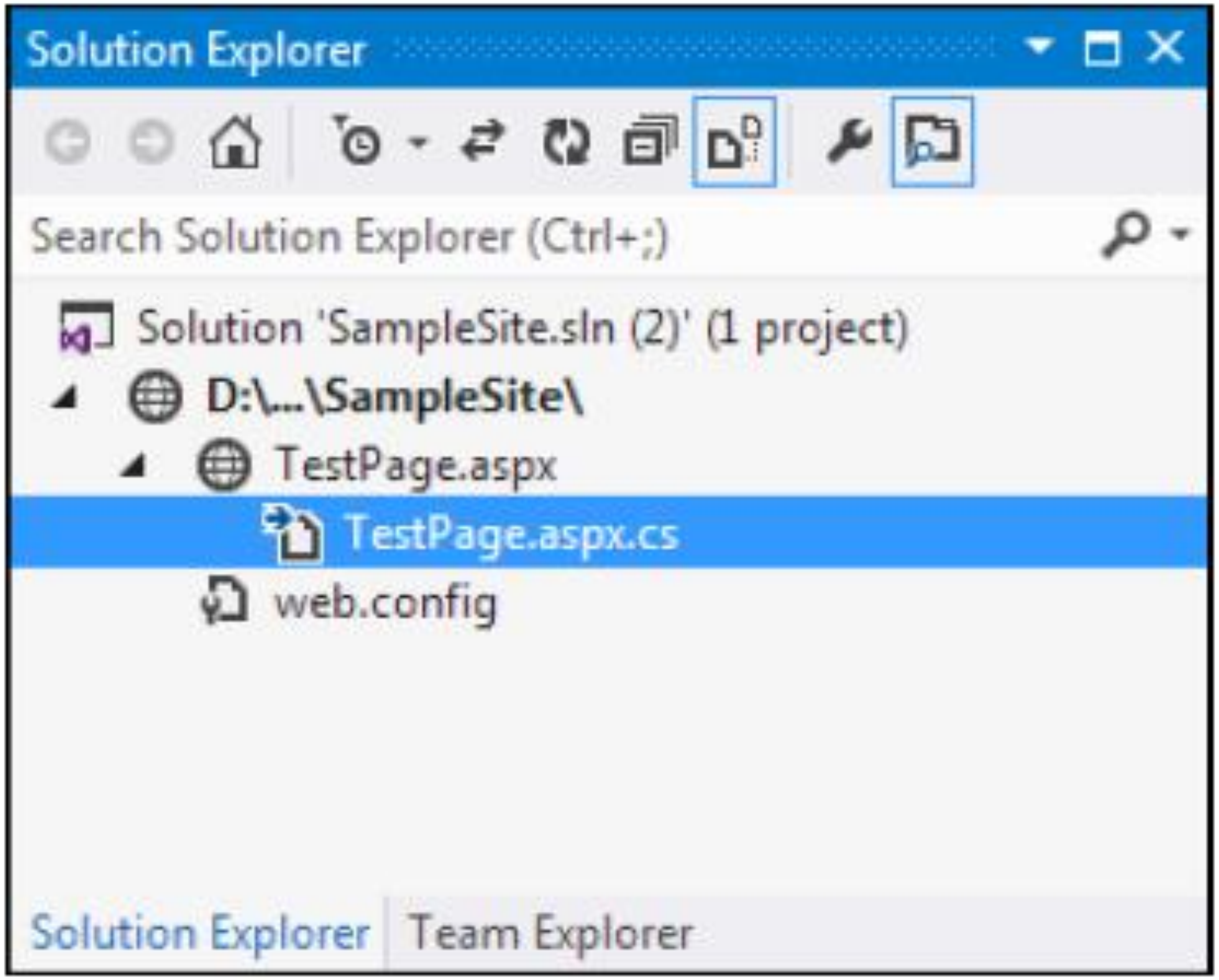
A form for Web Applications

Name: TestPage.aspx

[✓] Place code in separate file

[ ] Select master page

Figure 4-5. Adding an ASP.NET web form


Solution Explorer

Search Solution Explorer (Ctrl+;)

Solution 'SampleSite.sln (2)' (1 project)
D:\...\SampleSite\
TestPage.aspx
TestPage.aspx.cs
web.config

Solution Explorer | Team Explorer

Figure 4-6. A code file for a web page

Adding Web Controls

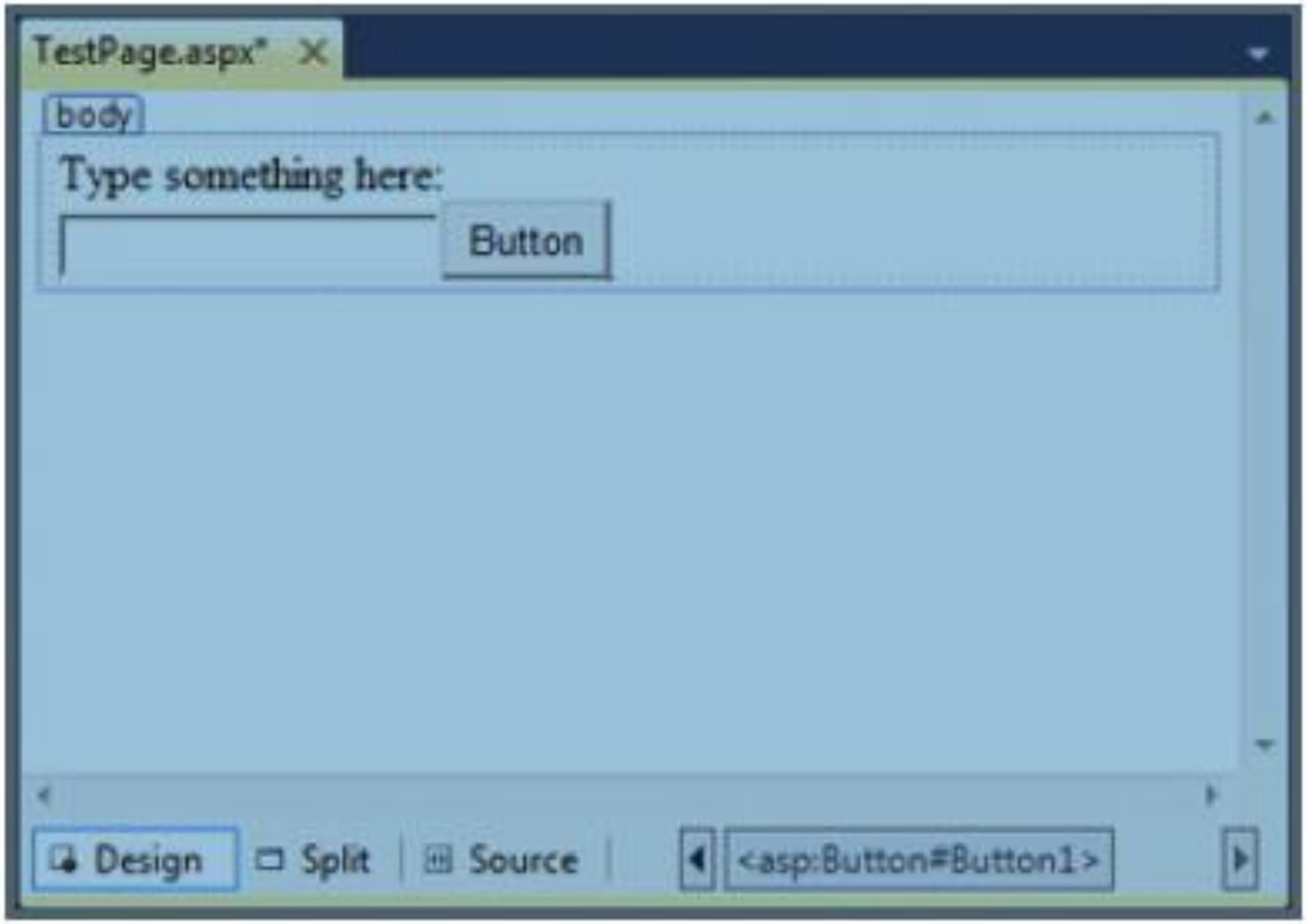
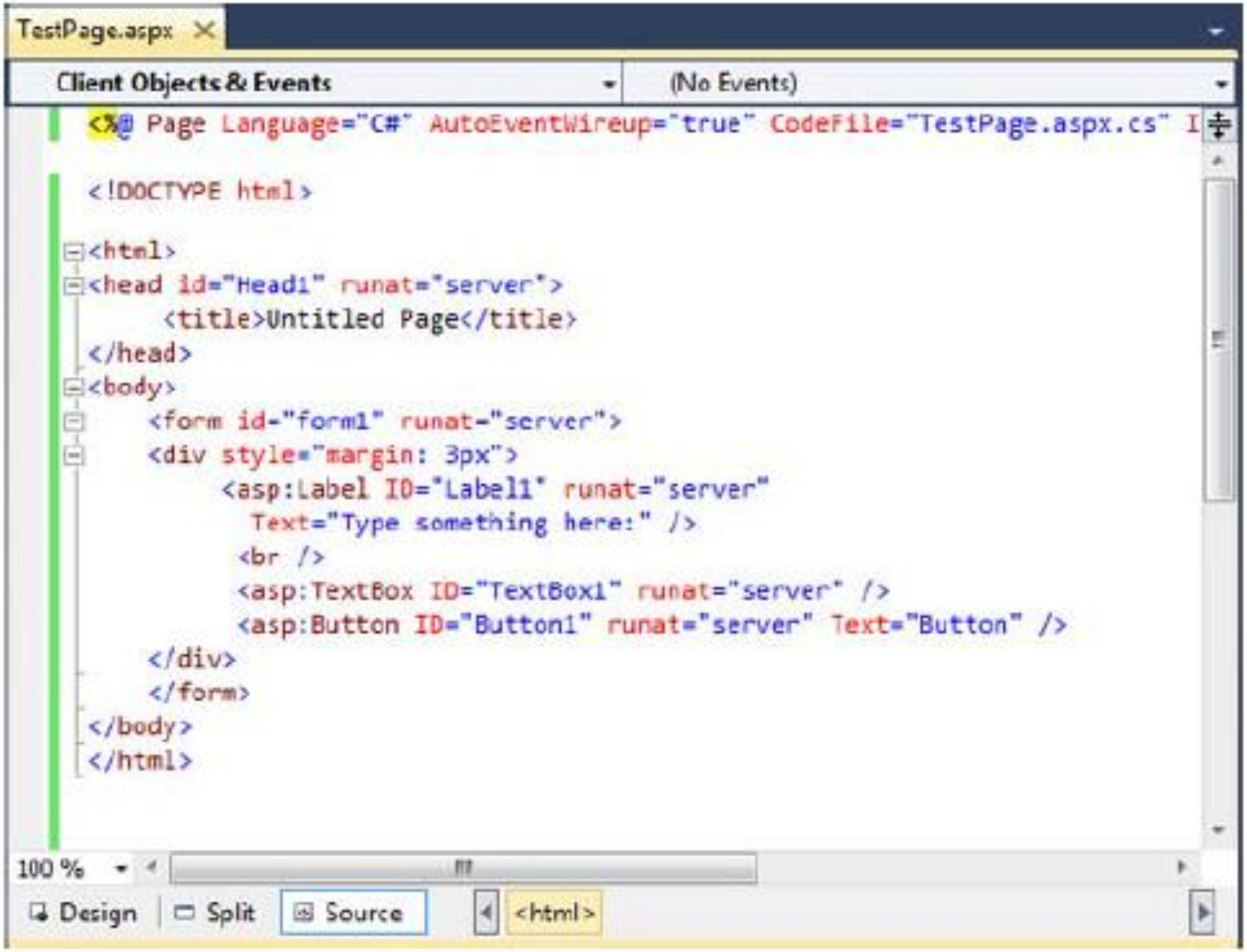


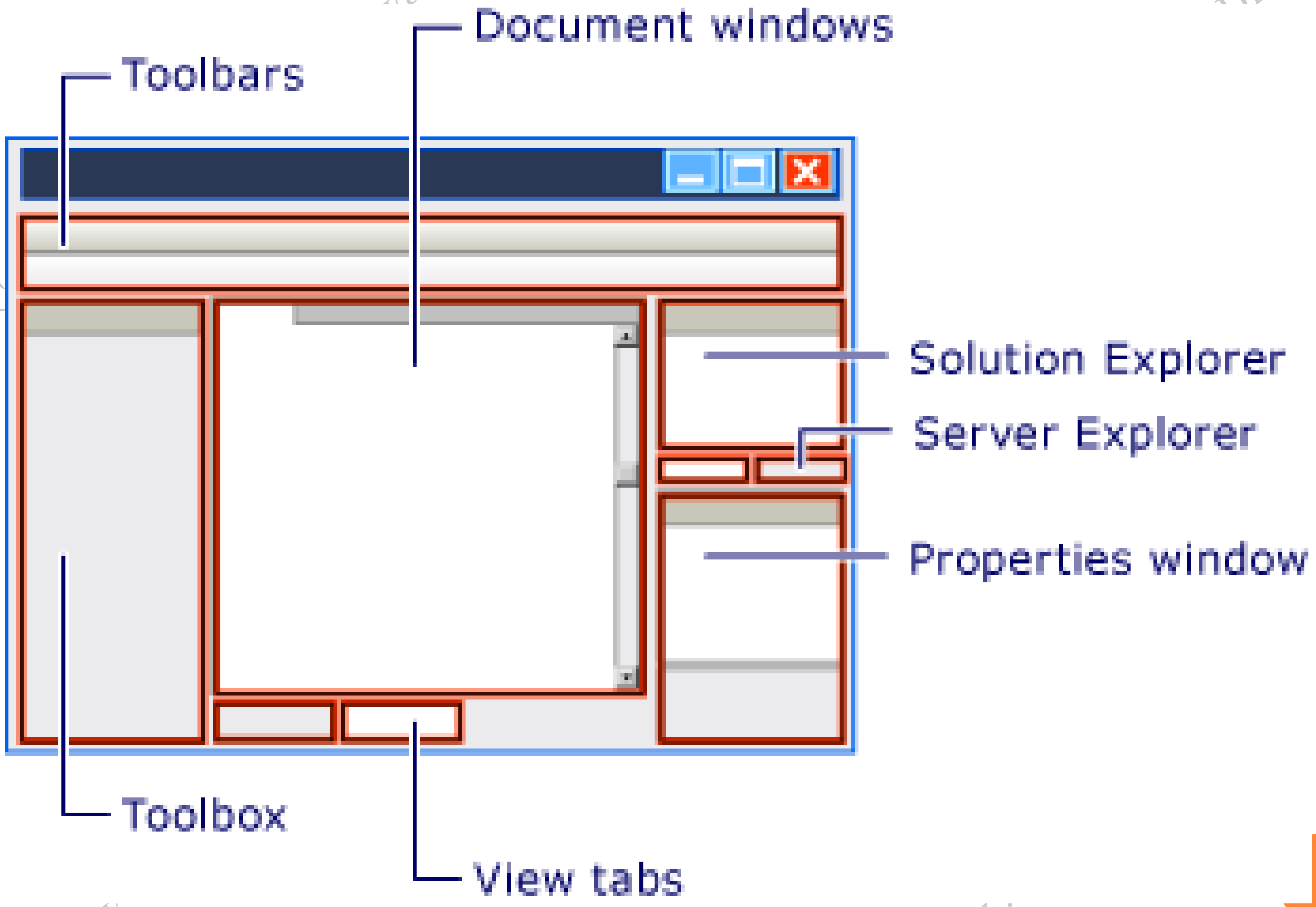
Figure 4-9. The design view for a page



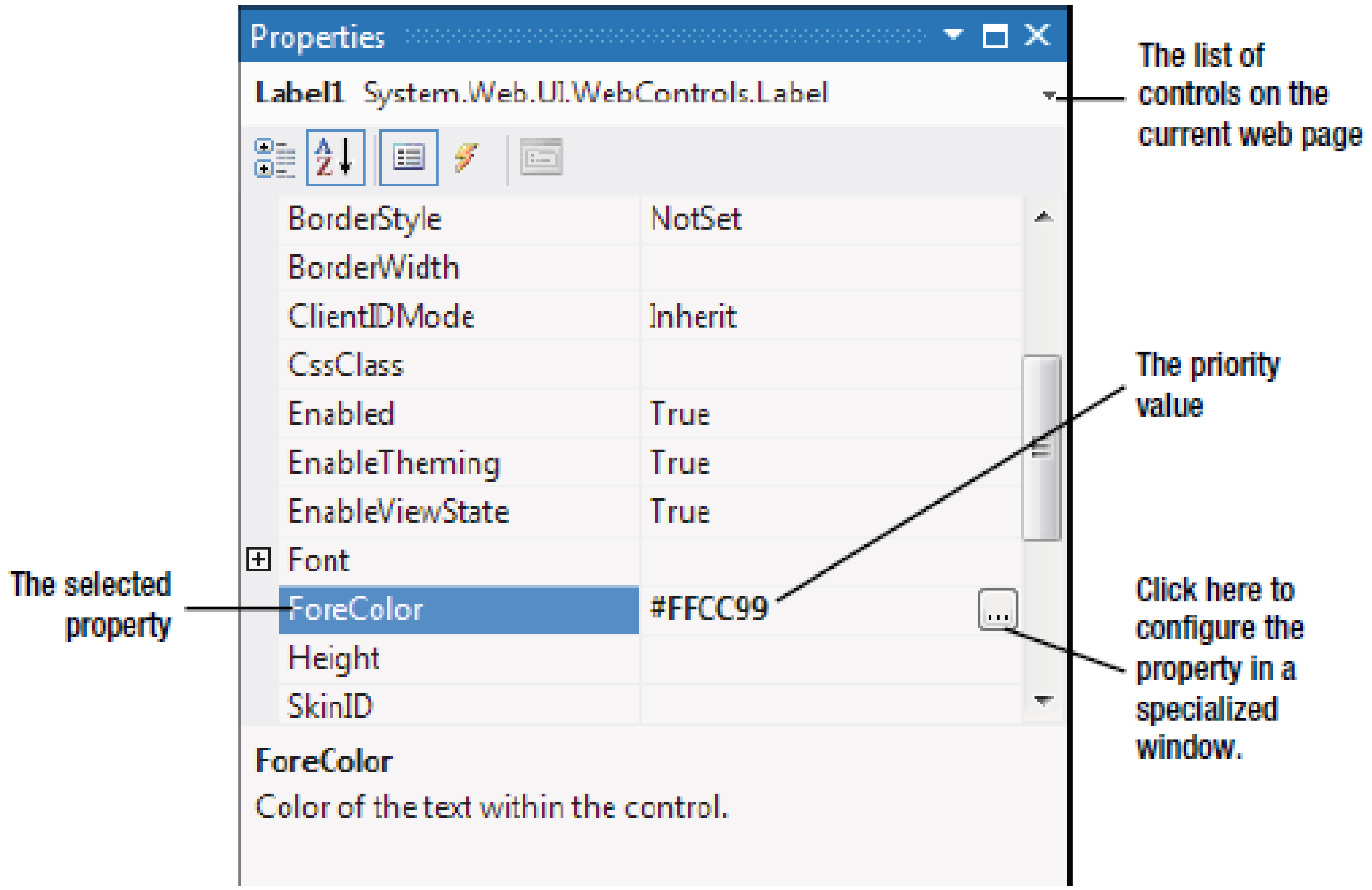
```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="TestPage.aspx.cs" I

<!DOCTYPE html>

<html>
<head id="Head1" runat="server">
     <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
    <div style="margin: 3px">
         <asp:Label ID="Label1" runat="server"
          Text="Type something here:" />
          <br />
          <asp:TextBox ID="TextBox1" runat="server" />
          <asp:Button ID="Button1" runat="server" Text="Button" />
    </div>
    </form>
</body>
</html>
```

Figure 4-10. The source view for a page

The Visual Studio environment



Using the Properties Window



Properties

Label1 System.Web.UI.WebControls.Label

BorderStyle	NotSet
BorderWidth	
ClientIDMode	Inherit
CssClass	
Enabled	True
EnableTheming	True
EnableViewState	True
⊞ Font	
ForeColor	#FFCC99
Height	
SkinID	

ForeColor

Color of the text within the control.

The list of controls on the current web page
The priority value
The selected property
Click here to configure the property in a specialized window.

Figure 4-11. The ForeColor property in the Properties window

INLINE CODE

```
<%@ Page Language="C#" AutoEventWireup="true"
 CodeFile="Default.aspx.cs" Inherits="_Default" %>

<!DOCTYPE html>

<html>
 <head runat="server">
  <title>Untitled Page</title>
 </head>
 <body>
  <form ID="form1" runat="server">
   <div>
    <asp:Label ID="Label1" runat="server"
     Text="Type something here:" />
    <br />
    <asp:TextBox ID="TextBox1" runat="server" />
    <asp:Button ID="Button1" runat="server" Text="Button" />
   </div>
  </form>
 </body>
</html>
```

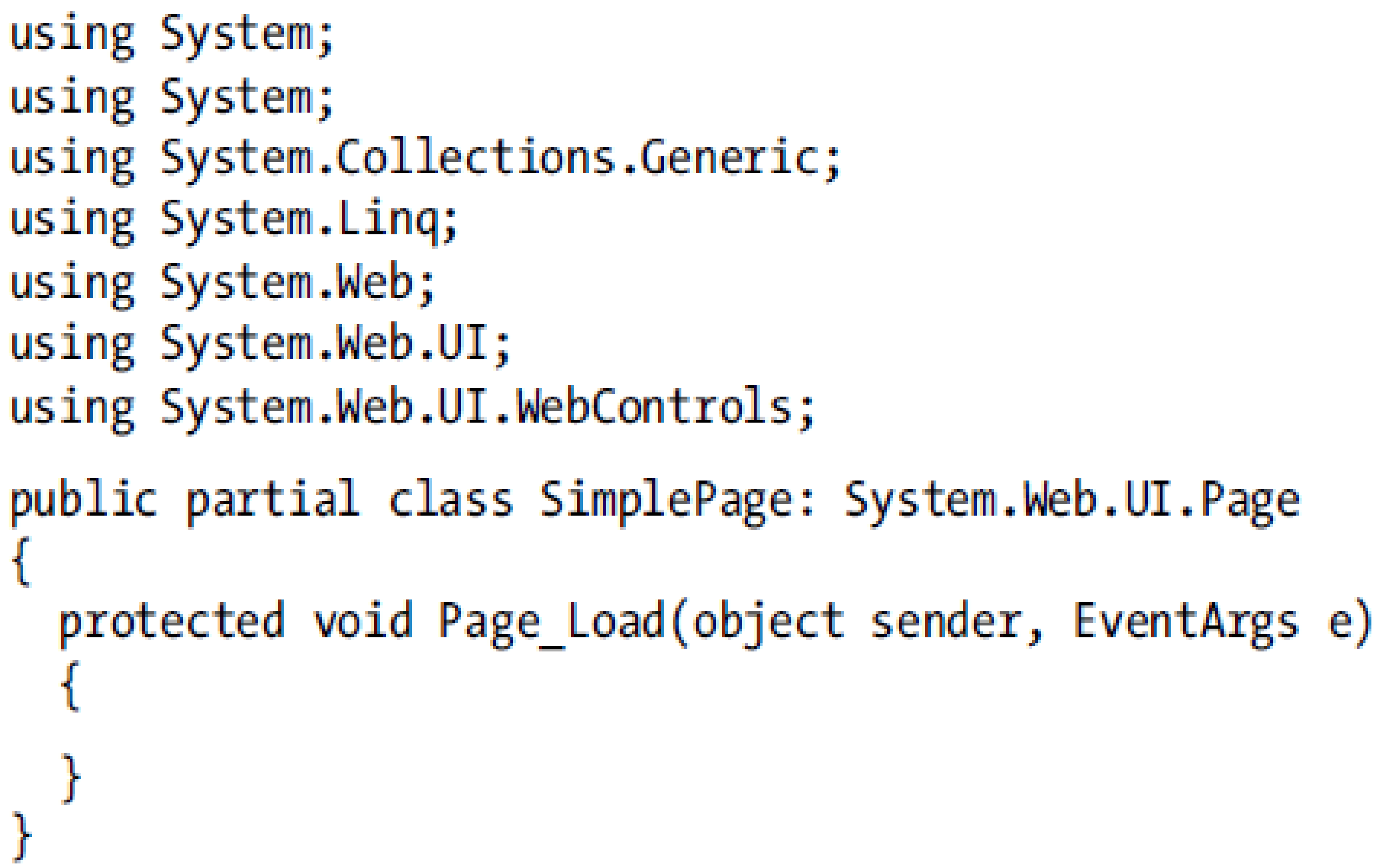
The Page Directive

- Defines **page**-specific attributes that guide the behavior of the **page** compiler and the language parser that will preprocess the **page**.

```
<%@ Page Language = "C#" AutoEventWireup = "true"
CodeFile = "Default.aspx.cs" Inherits = "_Default" %>
```

- CodeFile: The code behind file name
- Language: The programming language for the code to be added
- AutoEventWireup: It states that the page is automatically bound to the methods. Indicated by a Boolean value.
- Inherits : The name of the class to inherit from

Code-Behind Class



```
using System;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class SimplePage: System.Web.UI.Page
{
  protected void Page_Load(object sender, EventArgs e)
  {

  }
}
```

Adding Event Handlers

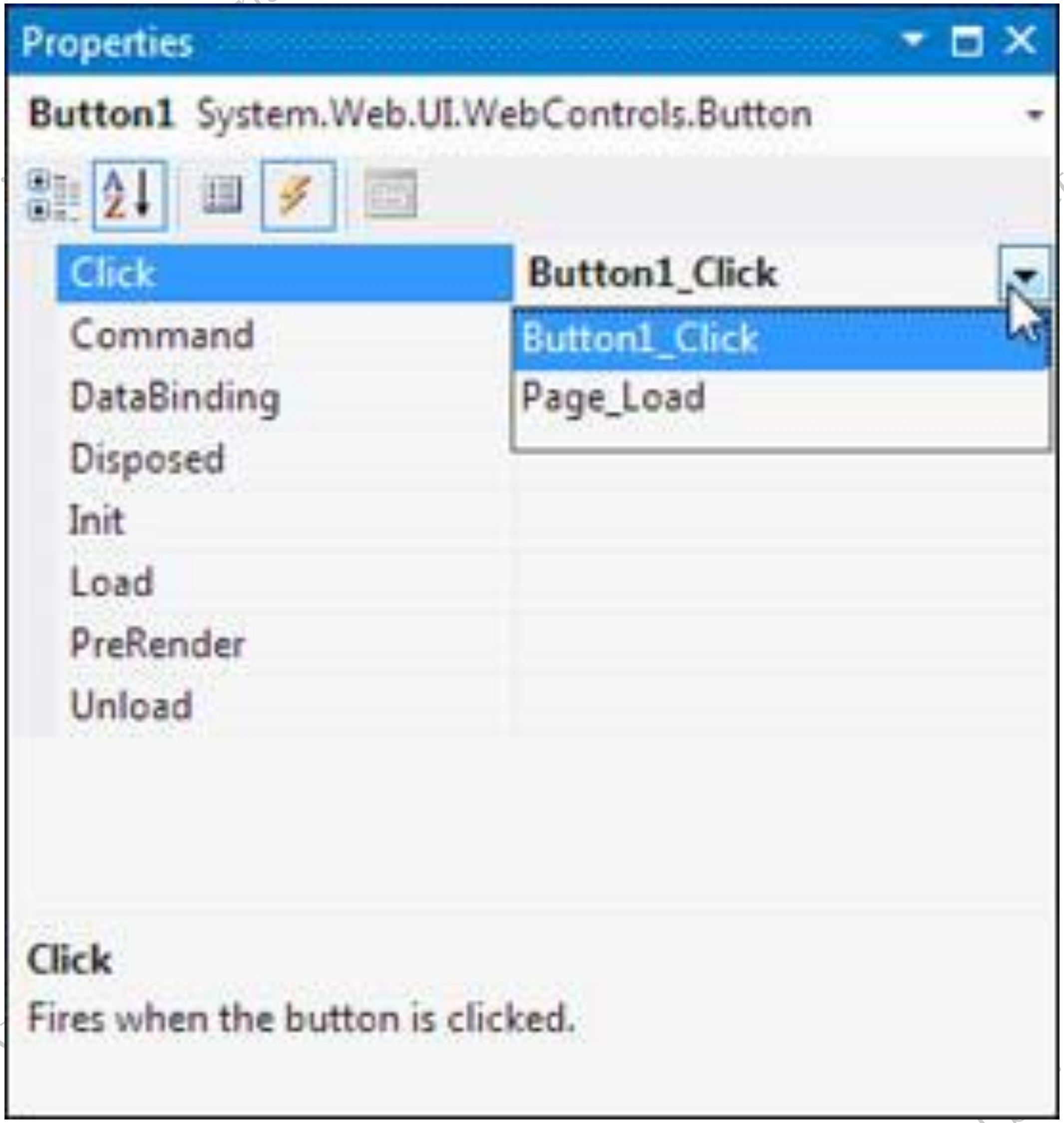
- *Type it in manually*
- *Double-click a control in design view*
- *Choose the event from the Properties window*

```
protected void Button1_Click(object sender, EventArgs e)
{
// Your code for reacting to the button click goes here.
}
```

```
<asp:Button ID = "Button1" runat = "server" Text = "Button" OnClick
= "Button1_Click" />
```

```
protected void Button1_Click(object sender, EventArgs e)
{
TextBox1.Text = “Here is some sample text.”;
}
```

Adding Event Handlers

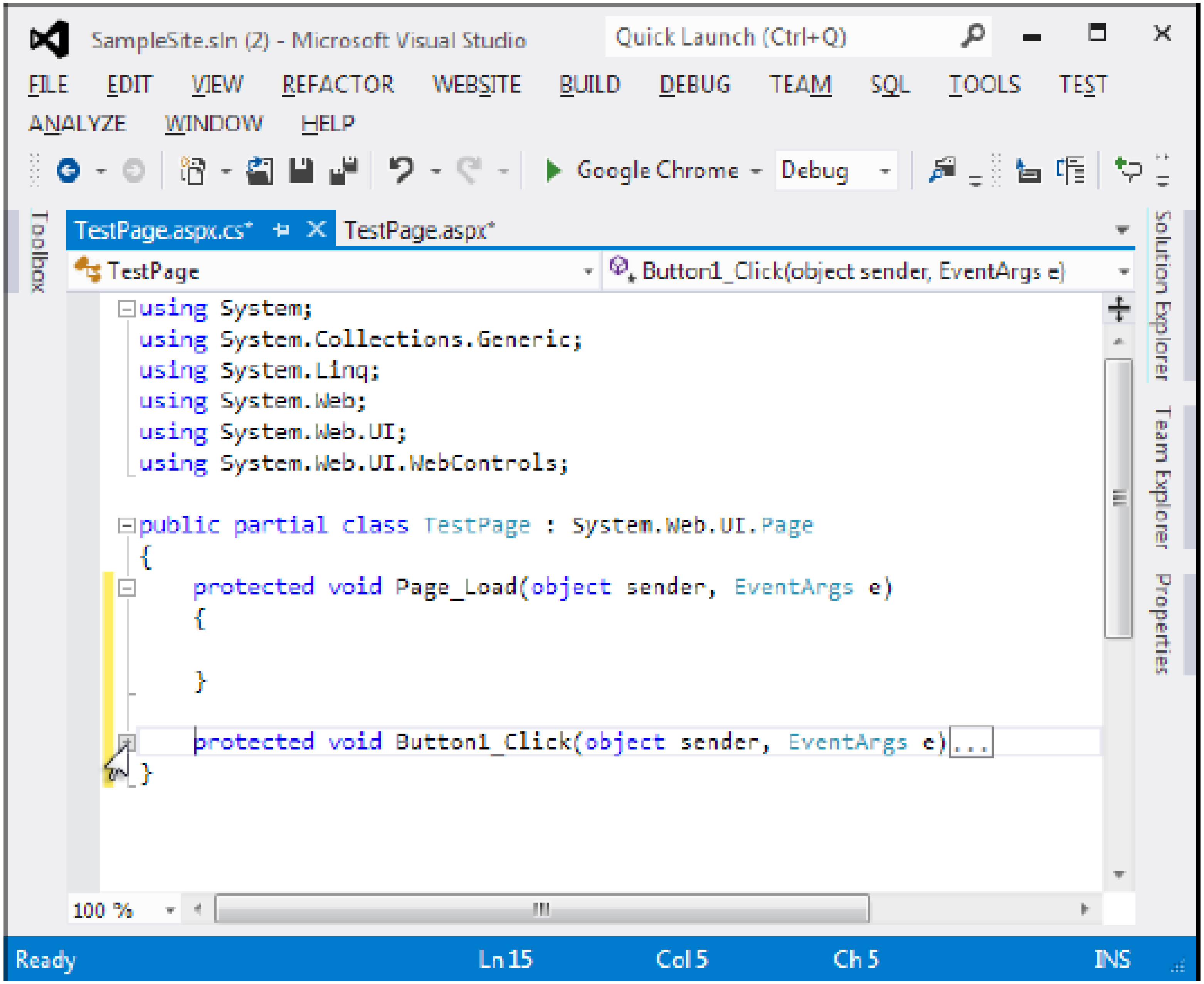

Properties

Button1 System.Web.UI.WebControls.Button

Click	Button1_Click
Command	
DataBinding	
Disposed	
Init	
Load	
PreRender	
Unload	

Click

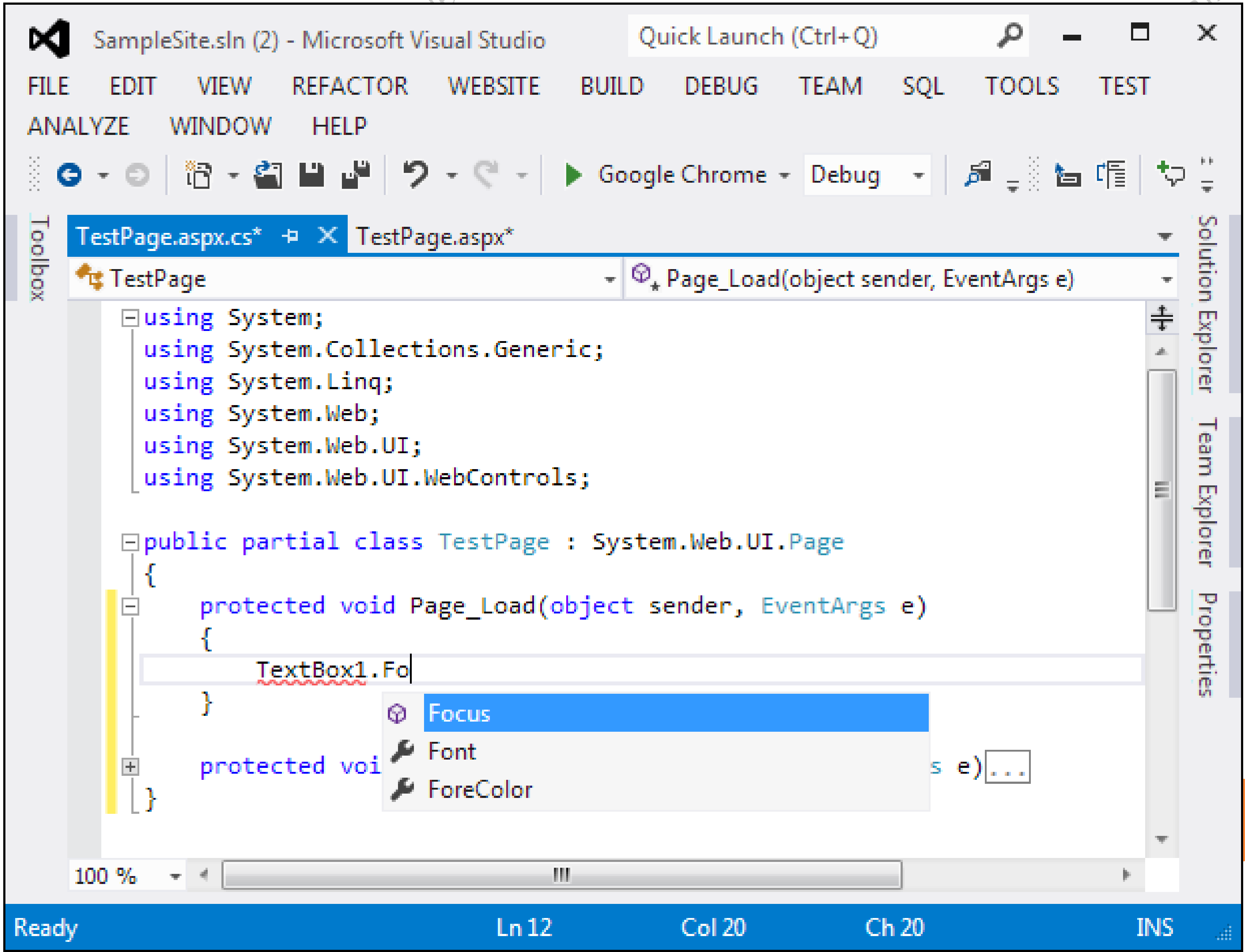
Fires when the button is clicked.


SampleSite.sln (2) - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
FILE EDIT VIEW REFACTOR WEBSITE BUILD DEBUG TEAM SQL TOOLS TEST ANALYZE WINDOW HELP
Google Chrome
Debug
Toolbox
TestPage.aspx.cs*
TestPage.aspx*
TestPage
Button1_Click(object sender, EventArgs e)
Solution Explorer
Team Explorer
Properties
100 %
Ready
Ln 15
Col 5
Ch 5
INS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class TestPage : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void Button1_Click(object sender, EventArgs e)...
}
```

Figure 4-14. Collapsing code

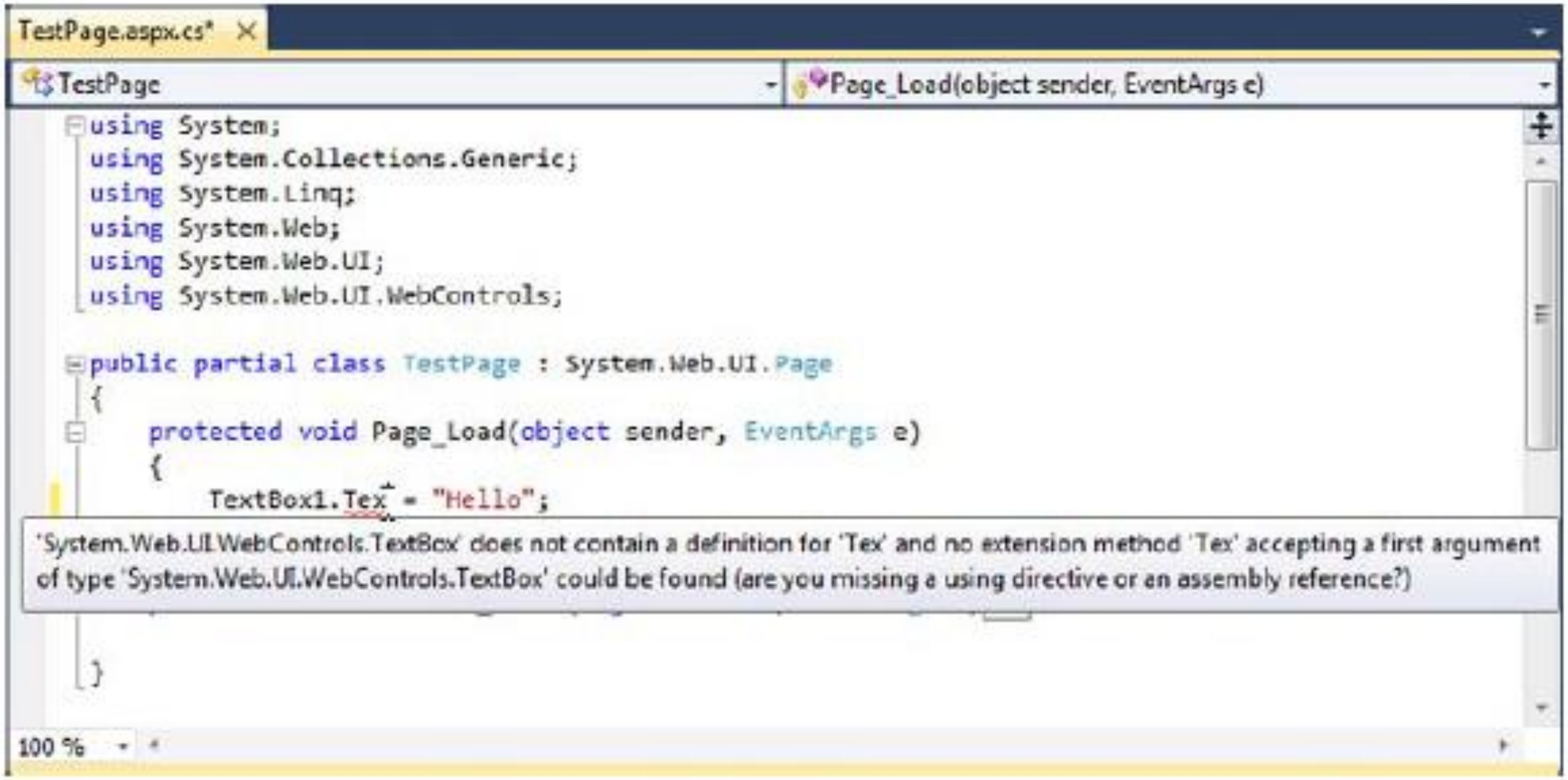
IntelliSense


SampleSite.sln (2) - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
FILE EDIT VIEW REFACTOR WEBSITE BUILD DEBUG TEAM SQL TOOLS TEST ANALYZE WINDOW HELP
Google Chrome
Debug
TestPage.aspx.cs*
TestPage.aspx*
TestPage
Page_Load(object sender, EventArgs e)
Toolbox
Solution Explorer
Team Explorer
Properties
100 %
Ready
Ln 12
Col 20
Ch 20
INS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class TestPage : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        TextBox1.Fo
    }

    protected voi                                                s e)...
}
```

```
Focus
Font
ForeColor
```

Catching Errors in Code

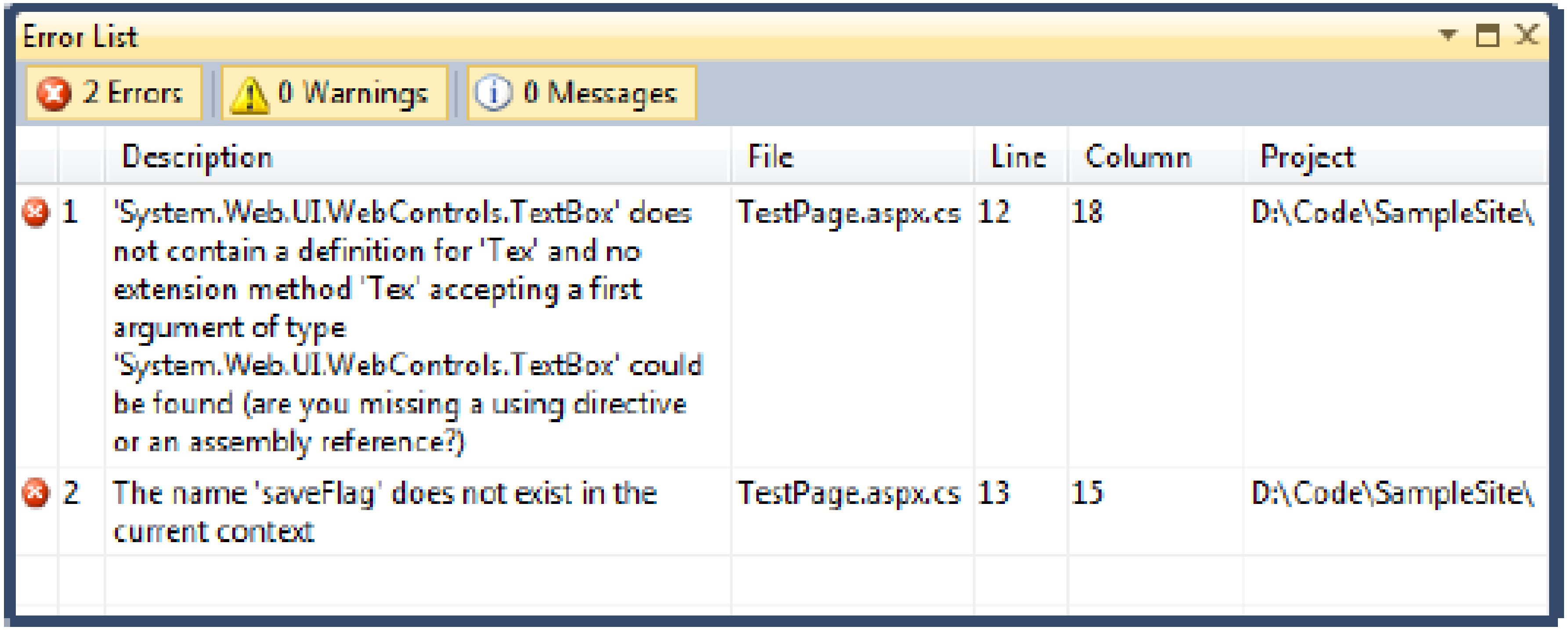


```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class TestPage : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        TextBox1.Tex = "Hello";
```

'System.Web.UI.WebControls.TextBox' does not contain a definition for 'Tex' and no extension method 'Tex' accepting a first argument of type 'System.Web.UI.WebControls.TextBox' could be found (are you missing a using directive or an assembly reference?)

```
}
```

***Figure 4-17.** Highlighting errors at design time*

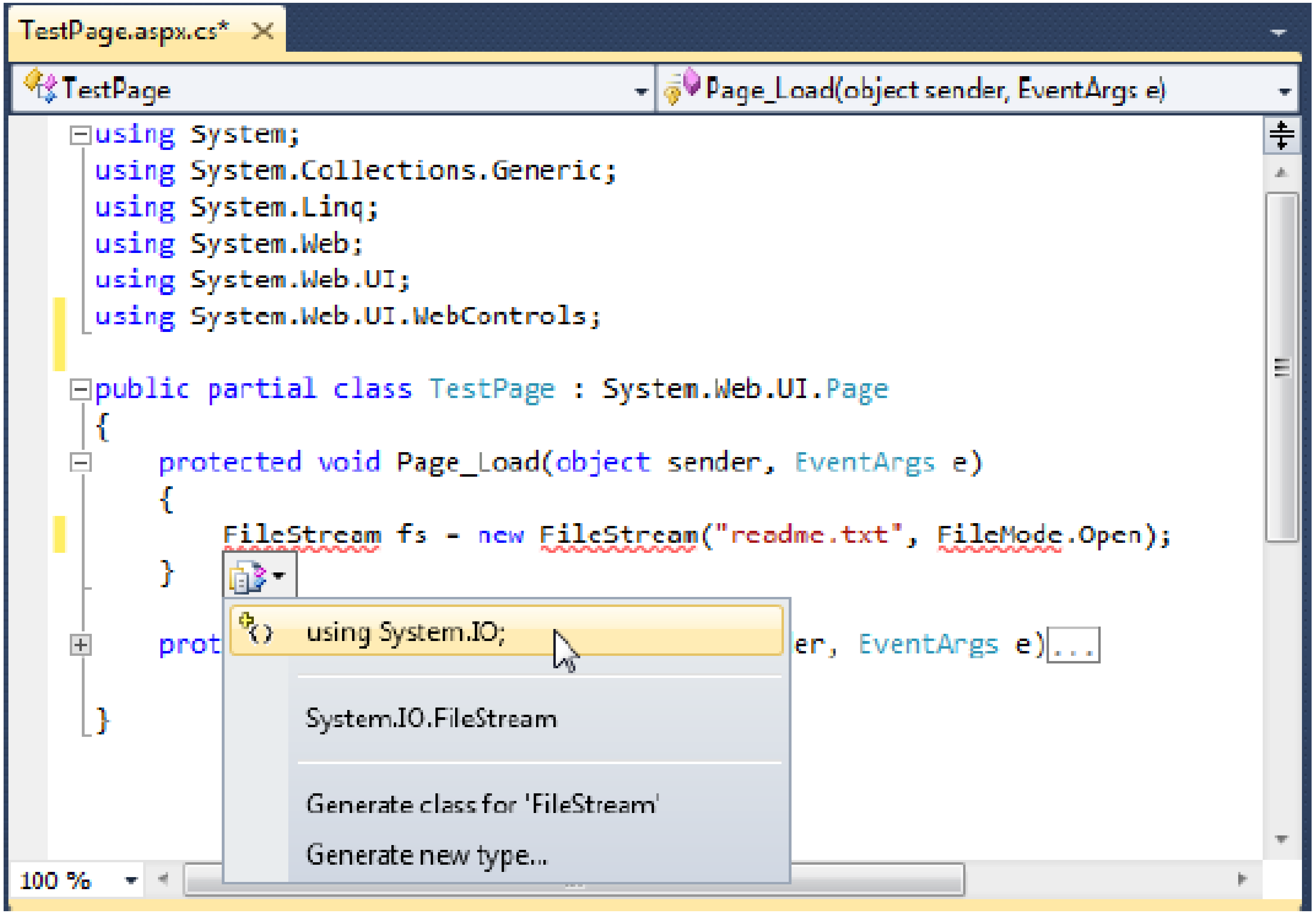

Error List

2 Errors | 0 Warnings | 0 Messages

		Description	File	Line	Column	Project
●	1	'System.Web.UI.WebControls.TextBox' does not contain a definition for 'Tex' and no extension method 'Tex' accepting a first argument of type 'System.Web.UI.WebControls.TextBox' could be found (are you missing a using directive or an assembly reference?)	TestPage.aspx.cs	12	18	D:\Code\SampleSite\
●	2	The name 'saveFlag' does not exist in the current context	TestPage.aspx.cs	13	15	D:\Code\SampleSite\

Figure 4-18. The Error List window

AUTOMATICALLY IMPORTING NAMESPACES



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class TestPage : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        FileStream fs = new FileStream("readme.txt", FileMode.Open);
    }

    prot                                    er, EventArgs e)...

}
```

***Figure 4-20.** Importing a missing namespace*

ATTRIBUTES

- <img src = "happy.gif" alt = "Happy Face" />
- <p>
- Click < a href = "http://www.prosetech.com">here</a> to visit my website.
- </p>

Web Form Fundamentals

- **Anatomy of an ASP.NET Application**
  - Every ASP.NET application shares a **common set of resources and configuration settings**.
  - Web pages from other ASP.NET applications don't share these resources, even if they're on the same web server.
  - Every ASP.NET application is executed inside a separate ***application domain***.
  - Application domains are isolated areas in memory, and they ensure that even if one web application causes a fatal error, it's unlikely to affect any other application that is currently running on the same computer.
  - Application domains restrict a web page in one application from accessing the in-memory information of another application.
  - Each web application is maintained separately and has its own set of cached, application, and session data.

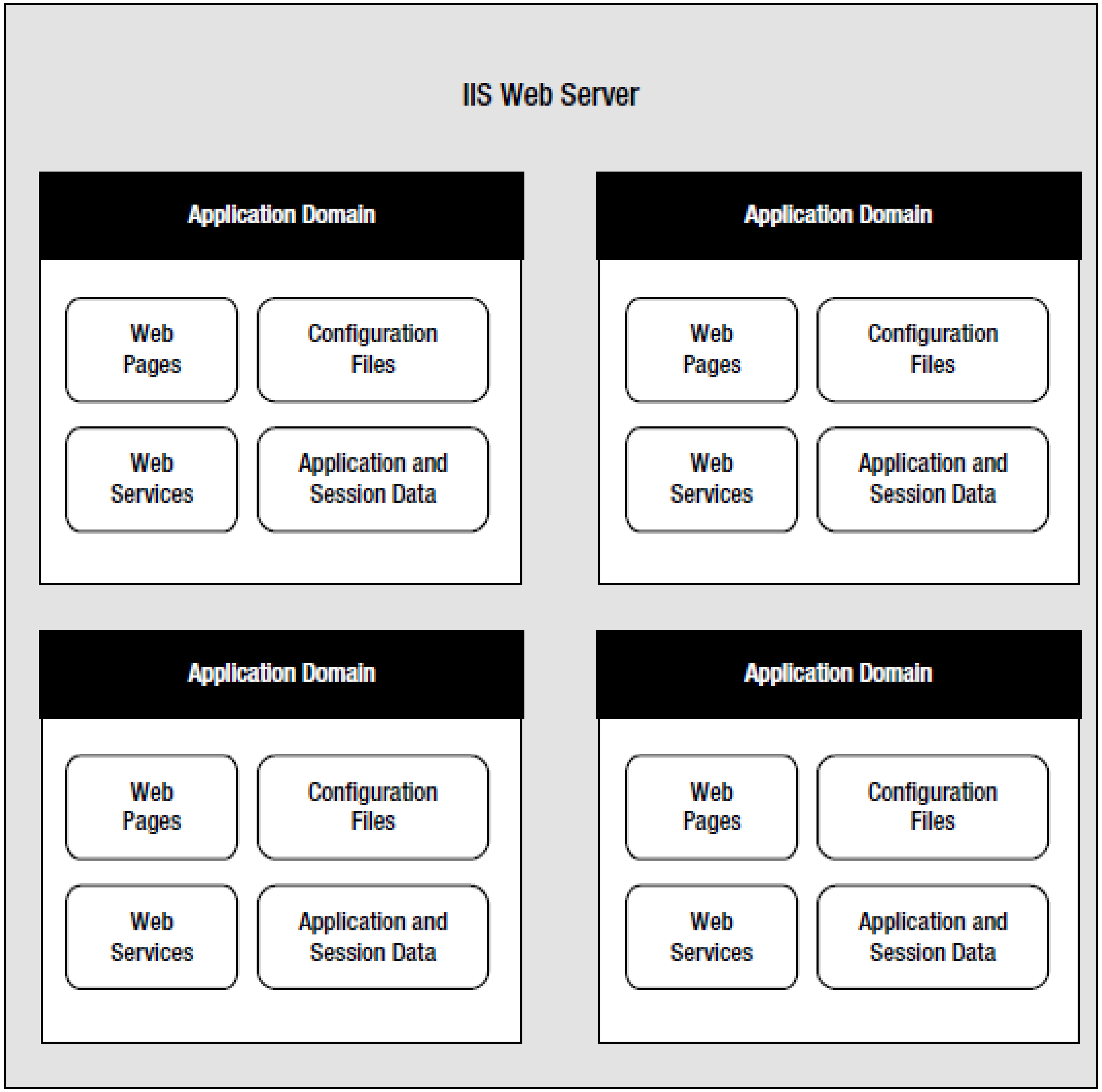


***Figure 5-1.** ASP.NET applications*

ASP.NET File Types

File Name	Description
Ends with .aspx	These are ASP.NET web pages.
Ends with .ascx	These are ASP.NET user controls User controls allow you to develop a small piece of user interface and reuse it in as many web forms as you want without repetitive code
web.config	This is the configuration file for your ASP.NET application. It includes settings for customizing security, state management, memory management
global.asax	This is the global application file. You can use this file to define global variables and react to global events
Ends with .cs	These are code-behind files that contain C# code. They allow you to separate the application logic from the user interface of a web page.

ASP.NET Web Folders

Directory	Description
App_Browsers	Contains .browser files that ASP.NET uses to identify the browsers that are using your application and determine their capabilities.
App_Code	Contains source code files that are dynamically compiled for use in your application
App_GlobalResources	Stores global resources that are accessible to every page in the web application.
App_LocalResources	Serves the same purpose as App_GlobalResources, except these resources are accessible to a specific page only
App_WebReferences	Stores references to web services, which are remote code routines that a webapplication can call over a network or the Internet.
App_Data	Stores data, including SQL Server Express database files
App_Themes	Stores the themes that are used to standardize and reuse formatting in your web application.
Bin	Contains all the compiled .NET components (DLLs) that the ASP.NET web applicationuses.

CONTROLS

HTML Server controls

Web Server Controls or ASP.NET controls

- Web Server Controls are group of controls derived directly from the System.Web.UI.WebControls base class.
- They are executed on the server side and output HTML sent back to the client browser.
- These controls are programmable and reusable.
- Web Server Controls can detect the target browser's capabilities and render themselves accordingly.

Table 4-1. Basic HTML Elements

Tag	Name	Type	Description
<b>, <i>, <u>	Bold, Italic, Underline	Container	These elements are used to apply basic formatting and make text bold, italic, or underlined. Some web designers prefer to use <strong> instead of <b>, and <emphasis> instead of <i>. Although these elements have the same standard rendering (bold and italic, respectively), they make more sense if you plan to use styles to change the formatting sometime in the future.
<p>	Paragraph	Container	The paragraph groups a block of free-flowing text together. The browser automatically adds a bit of space between paragraphs and other elements (such as headings) or between subsequent paragraphs.
<h1>, <h2>, <h3>, <h4>, <h5>, <h6>	Heading	Container	These elements are headings, which give text bold formatting and a large font size. The lower the number, the larger the text, so <h1> is for the largest heading. The <h5> heading is normal text size, and <h6> is actually a bit smaller than ordinary text.
<img>	Image	Stand-alone	The image element shows an external image file (specified by the src attribute) in a web page.
<br>	Line Break	Stand-alone	This element adds a single line break, with no extra space.
<hr>	Horizontal Line	Stand-alone	This element adds a horizontal line (which gets the full width of the containing element). You can use the horizontal line to separate different content regions.
<a>	Anchor	Container	The anchor element wraps a piece of text and turns it into a link. You set the link target by using the href attribute.

<ul>, <li>	Unordered List, List Item	Container	These elements allow you to build bulleted lists. The <ul> element defines the list, while the <li> element defines an item in the list (you nest the actual content for that item inside).
<ol>, <li>	Ordered List, List Item	Container	These elements allow you to build numbered lists. The <ol> element defines the list, while the <li> element defines an item in the list (you nest the actual content for that item inside).
<table>, <tr>, <td>, <th>	Table	Container	The <table> element allows you to create a multicolumn, multirow table. Each row is represented by a <tr> element inside the <table>. Each cell in a row is represented by a <td> element inside a <tr>. You place the actual content for the cell in the individual <td> elements (or, in the case of the header cells that sit at the top of the table, you can use <th> elements instead).
<div>	Division	Container	This element is an all-purpose container for other elements. It's used to separate different regions on the page, so you can format them or position them separately. For example, you can use <div> to create a shaded box around a group of elements.
<span>	Span	Container	This element is an all-purpose container for bits of text content inside other elements (such as headings or paragraphs). It's most commonly used to format those bits of text. For example, you can use <span> to change the color of a few words in a sentence.
<form>	Form	Container	This element is used to hold all the controls on a web page. Controls are HTML elements that can send information back to the web server when the page is submitted. For example, text boxes submit their text, list boxes submit the currently selected item in the list, and so on.

HTML Controls	ASP.Net Controls
HTML control runs at client side.	ASP.Net controls run at server side.
You can run HTML controls at server side by adding attribute runat=”server”.	You can not run ASP.Net Controls on client side as these controls have this attribute runat=”server” by default.
HTML controls are client side controls, so it does not provide STATE management.	ASP.Net Controls are Server side controls, provides STATE management.
HTML control does not require rendering.	ASP.Net controls require rendering.
As HTML controls runs on client side, execution is fast.	As ASP.Net controls run on server side, execution is slow.
HTML controls do not have any separate class for its controls.	ASP.Net controls have separate class for its each control.
HTML controls does not support Object Oriented paradigm.	With ASP.Net controls, you have full support of Object oriented paradigm.
HTML controls can not be accessed form code behind files.	ASP.Net controls can be directly worked and accessed from code behind files.
HTML control have limited set of properties and/or methods.	ASP.Net controls have rich set of properties and/or methods.

SERVER CONTROLS

- are created and configured as ***objects.***
- They run on the web server and automatically provide their own HTML output
- key features :
- *They generate their own interface*
- *They retain their state*
- *They fire server-side events*

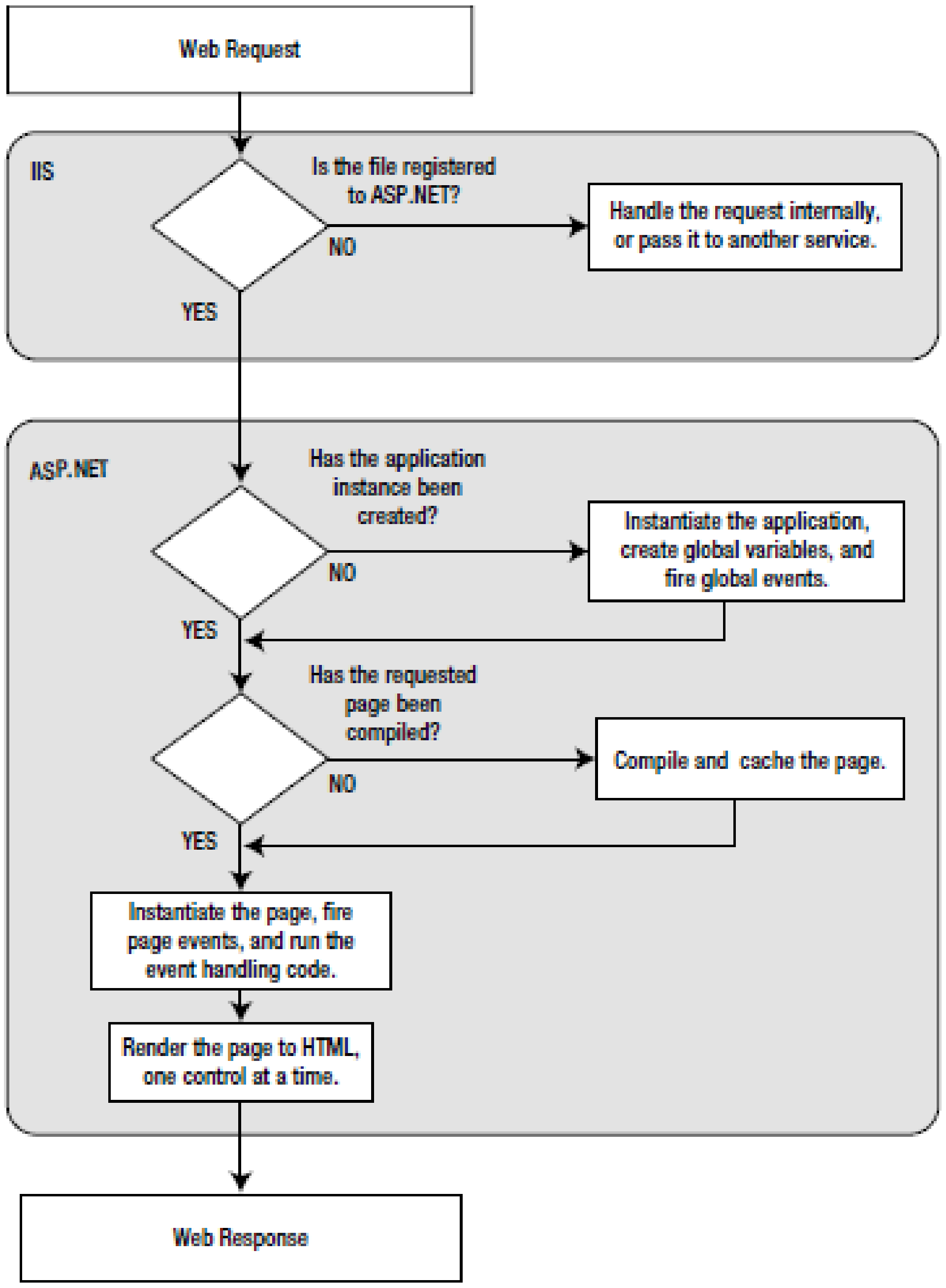
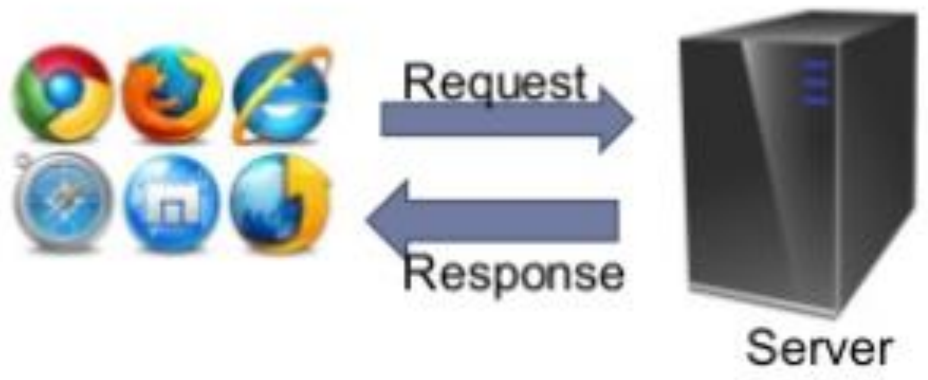

Web Request
IIS
Is the file registered to ASP.NET?
NO
Handle the request internally, or pass it to another service.
YES
ASP.NET
Has the application instance been created?
NO
Instantiate the application, create global variables, and fire global events.
YES
Has the requested page been compiled?
NO
Compile and cache the page.
YES
Instantiate the page, fire page events, and run the event handling code.
Render the page to HTML, one control at a time.
Web Response

Figure 5-4. The stages in an ASP.NET request

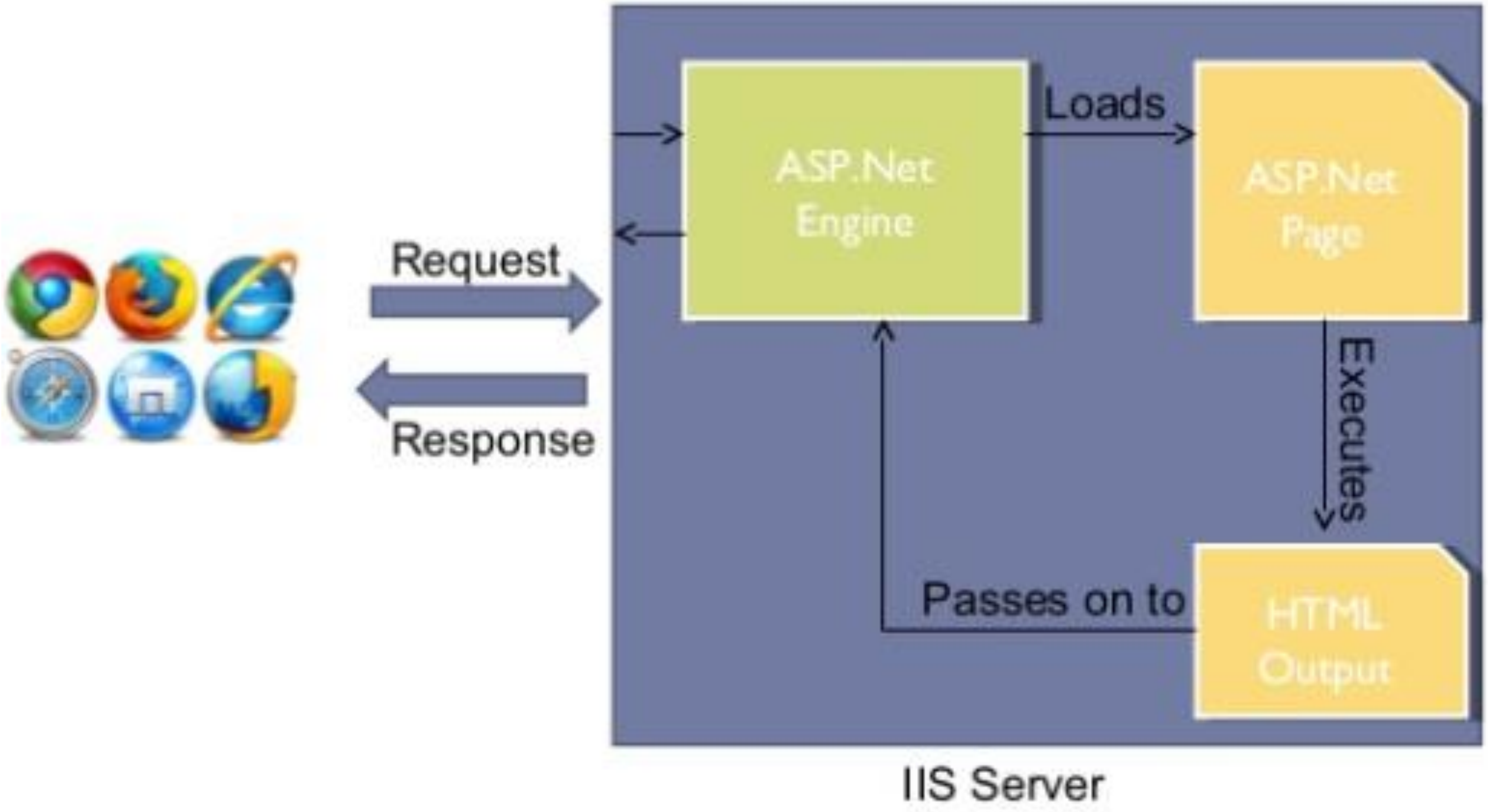
How browsers understand ASP.Net pages?

- If the client requests an HTML file:

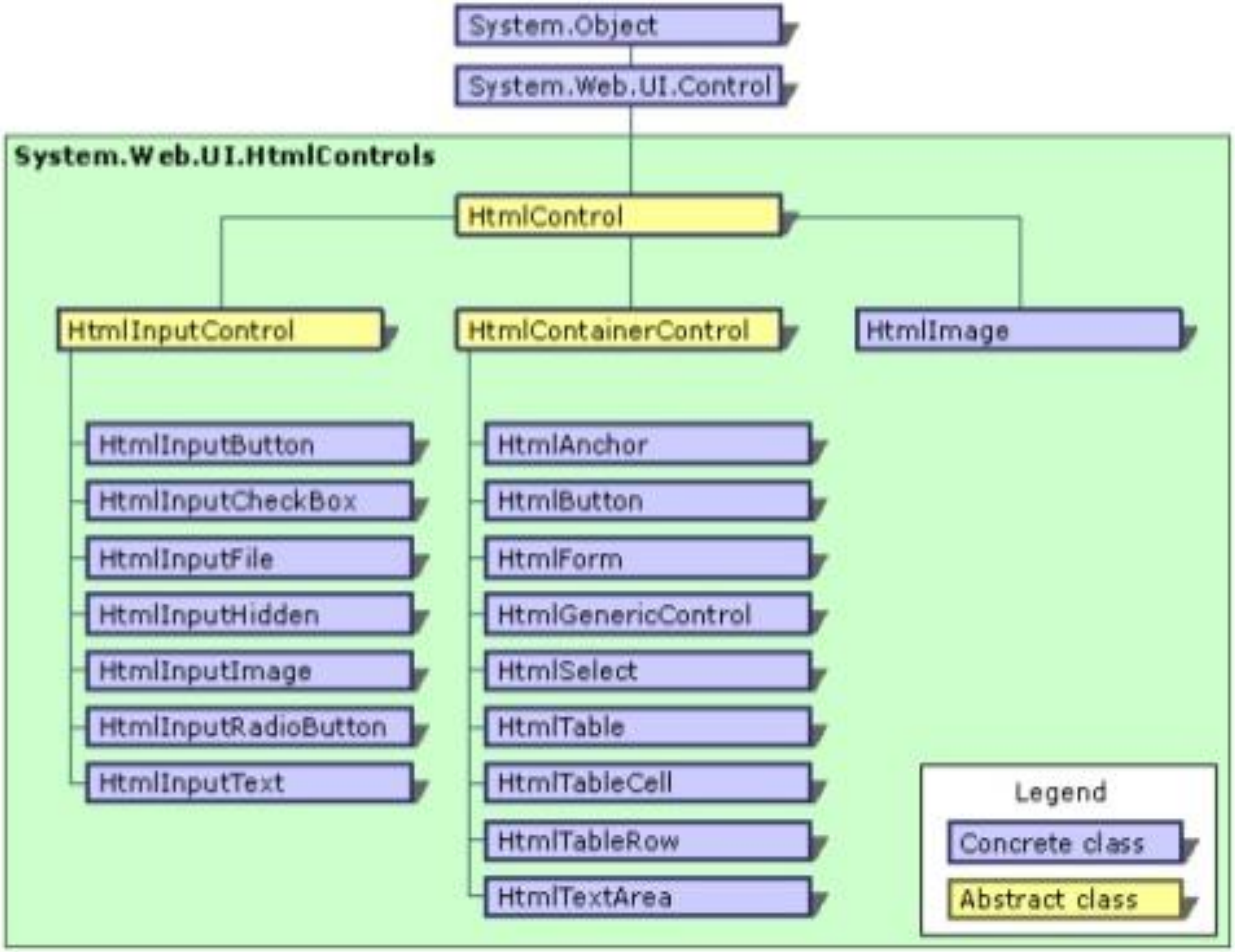


How browsers understand ASP.Net pages?

- If the client requests an ASP.Net page:



HTML Control Classes



HTML Control Events

- HTML server controls also provide one of two possible events: ServerClick or ServerChange.
- The ServerClick event is simply a click that's processed on the server side.
- The ServerChange event responds when a change has been made to a text or selection control

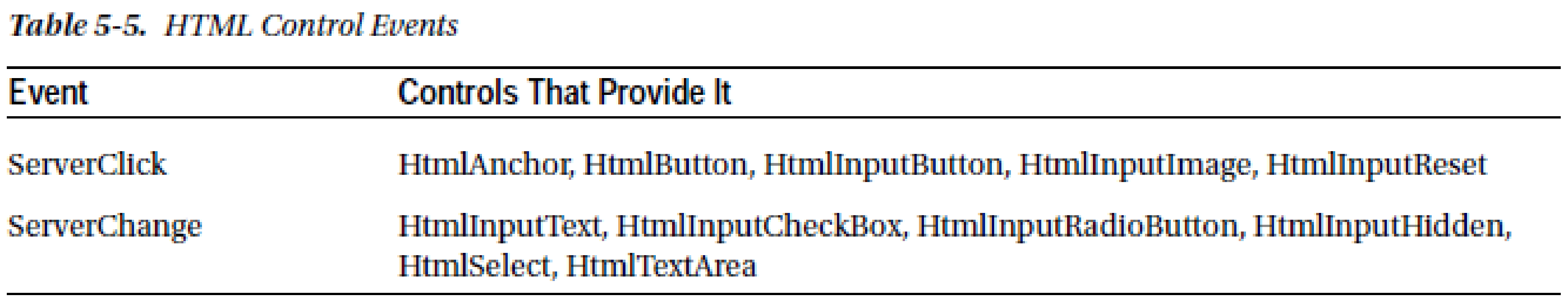


Table 5-5. HTML Control Events

Event	Controls That Provide It
ServerClick	HtmlAnchor, HtmlButton, HtmlInputButton, HtmlInputImage, HtmlInputReset
ServerChange	HtmlInputText, HtmlInputCheckBox, HtmlInputRadioButton, HtmlInputHidden, HtmlSelect, HtmlTextArea

The HtmlControl Base Class

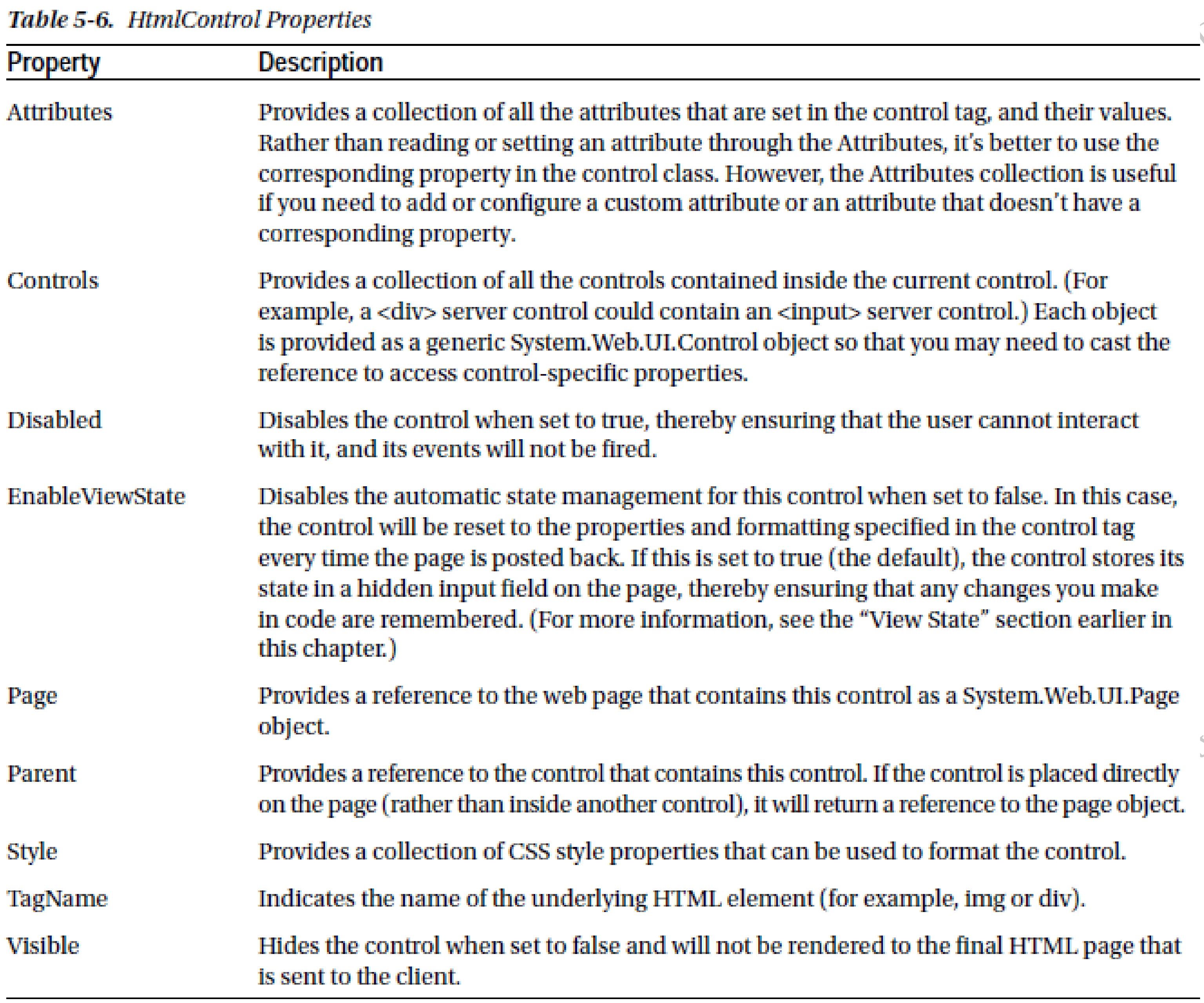


Table 5-6. HtmlControl Properties

Property	Description
Attributes	Provides a collection of all the attributes that are set in the control tag, and their values. Rather than reading or setting an attribute through the Attributes, it's better to use the corresponding property in the control class. However, the Attributes collection is useful if you need to add or configure a custom attribute or an attribute that doesn't have a corresponding property.
Controls	Provides a collection of all the controls contained inside the current control. (For example, a <div> server control could contain an <input> server control.) Each object is provided as a generic System.Web.UI.Control object so that you may need to cast the reference to access control-specific properties.
Disabled	Disables the control when set to true, thereby ensuring that the user cannot interact with it, and its events will not be fired.
EnableViewState	Disables the automatic state management for this control when set to false. In this case, the control will be reset to the properties and formatting specified in the control tag every time the page is posted back. If this is set to true (the default), the control stores its state in a hidden input field on the page, thereby ensuring that any changes you make in code are remembered. (For more information, see the "View State" section earlier in this chapter.)
Page	Provides a reference to the web page that contains this control as a System.Web.UI.Page object.
Parent	Provides a reference to the control that contains this control. If the control is placed directly on the page (rather than inside another control), it will return a reference to the page object.
Style	Provides a collection of CSS style properties that can be used to format the control.
TagName	Indicates the name of the underlying HTML element (for example, img or div).
Visible	Hides the control when set to false and will not be rendered to the final HTML page that is sent to the client.

THE HTMLINPUTCONTROL CLASS

- The HtmlInputControl class inherits from HtmlControl and adds some properties that are used for the <input> element.
- The <input> element can represent different controls,
- depending on the type attribute.
- The <input type = “text”> element is a text box, and <input type = “submit”> is a button.

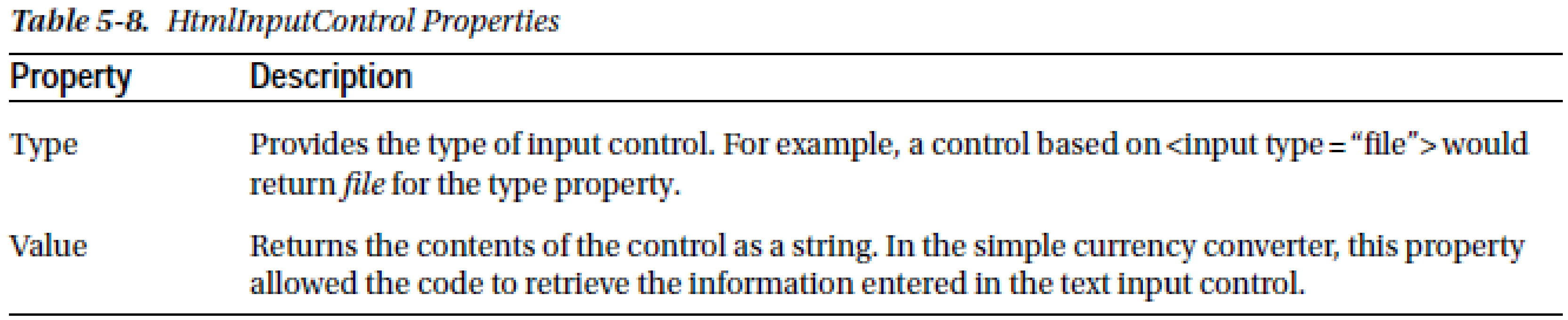


Table 5-8. HtmlInputControl Properties

Property	Description
Type	Provides the type of input control. For example, a control based on <input type = “file”> would return *file* for the type property.
Value	Returns the contents of the control as a string. In the simple currency converter, this property allowed the code to retrieve the information entered in the text input control.

THE HTMLCONTAINERCONTROL CLASS

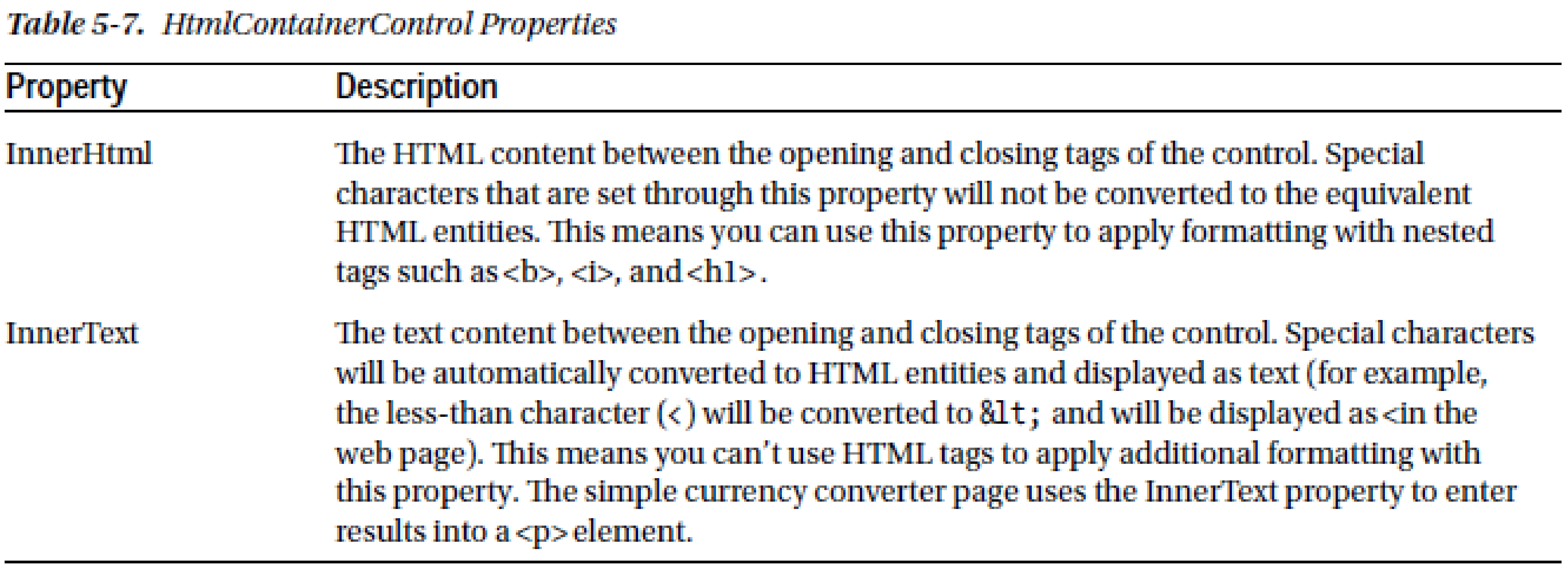
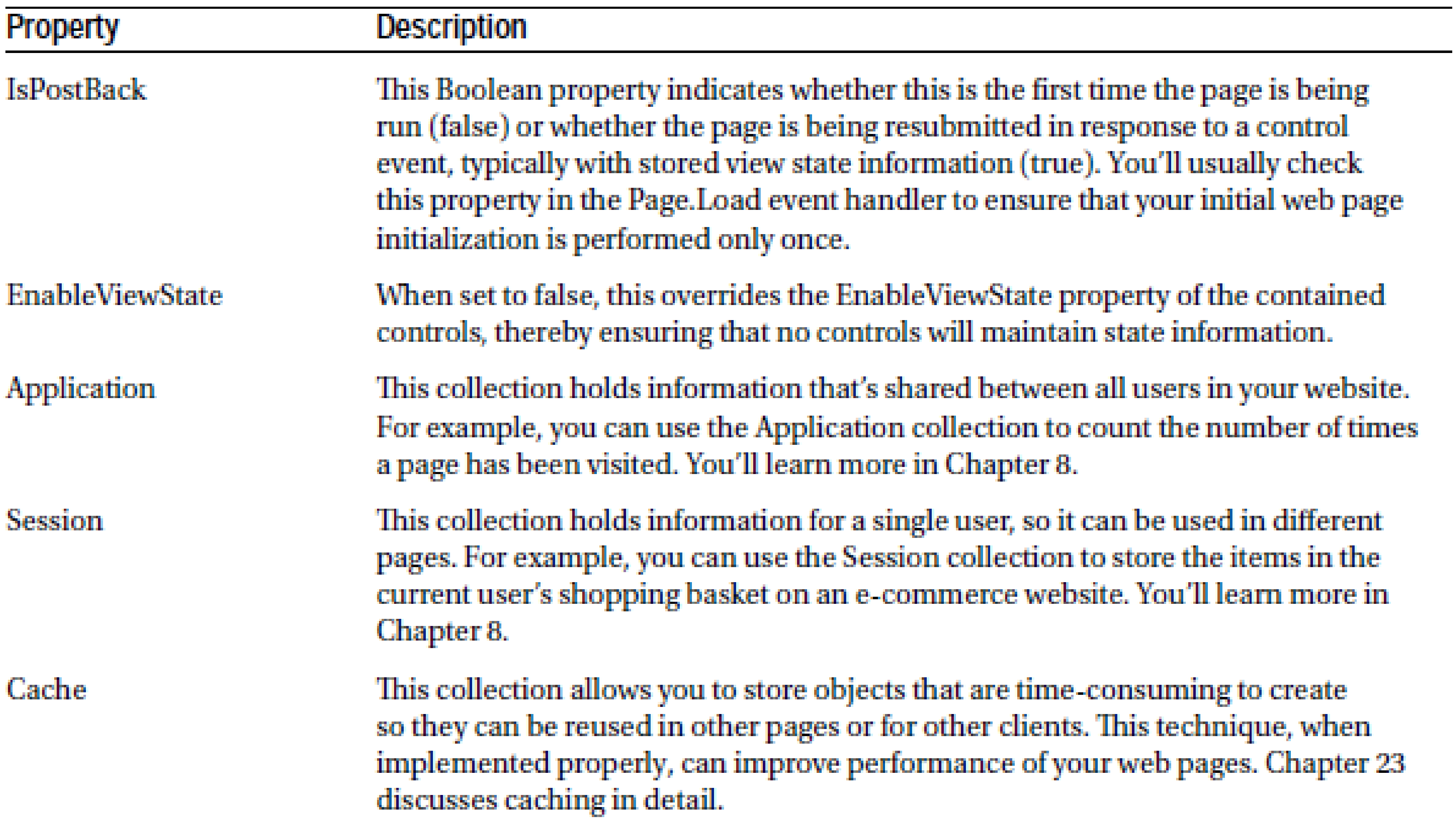


Table 5-7. HtmlContainerControl Properties

Property	Description
InnerHtml	The HTML content between the opening and closing tags of the control. Special characters that are set through this property will not be converted to the equivalent HTML entities. This means you can use this property to apply formatting with nested tags such as <b>, <i>, and <h1>.
InnerText	The text content between the opening and closing tags of the control. Special characters will be automatically converted to HTML entities and displayed as text (for example, the less-than character (<) will be converted to `&lt;` and will be displayed as < in the web page). This means you can't use HTML tags to apply additional formatting with this property. The simple currency converter page uses the InnerText property to enter results into a <p> element.

PAGE CLASS



Property	Description
IsPostBack	This Boolean property indicates whether this is the first time the page is being run (false) or whether the page is being resubmitted in response to a control event, typically with stored view state information (true). You'll usually check this property in the Page.Load event handler to ensure that your initial web page initialization is performed only once.
EnableViewState	When set to false, this overrides the EnableViewState property of the contained controls, thereby ensuring that no controls will maintain state information.
Application	This collection holds information that's shared between all users in your website. For example, you can use the Application collection to count the number of times a page has been visited. You'll learn more in Chapter 8.
Session	This collection holds information for a single user, so it can be used in different pages. For example, you can use the Session collection to store the items in the current user's shopping basket on an e-commerce website. You'll learn more in Chapter 8.
Cache	This collection allows you to store objects that are time-consuming to create so they can be reused in other pages or for other clients. This technique, when implemented properly, can improve performance of your web pages. Chapter 23 discusses caching in detail.

Request	This refers to an HttpRequest object that contains information about the current web request. You can use the HttpRequest object to get information about the user's browser, although you'll probably prefer to leave these details to ASP. NET. You'll use the HttpRequest object to transmit information from one page to another with the query string in Chapter 8.
Response	This refers to an HttpResponse object that represents the response ASP.NET will send to the user's browser. You'll use the HttpResponse object to create cookies in Chapter 8, and you'll see how it allows you to redirect the user to a different web page later in this chapter.
Server	This refers to an HttpServerUtility object that allows you to perform a few miscellaneous tasks. For example, it allows you to encode text so that it's safe to place it in a URL or in the HTML markup of your page. You'll learn more about these features in this chapter.
User	If the user has been authenticated, this property will be initialized with user information. Chapter 19 describes this property in more detail.

Sending the User to a New Page

- Click <a href = "newpage.aspx" > here</a > to go to newpage.aspx.
- Response.Redirect("newpage.aspx");
- Response.Redirect("http://www.prosetech.com");
- Server.Transfer("newpage.aspx");

APPLICATION EVENTS

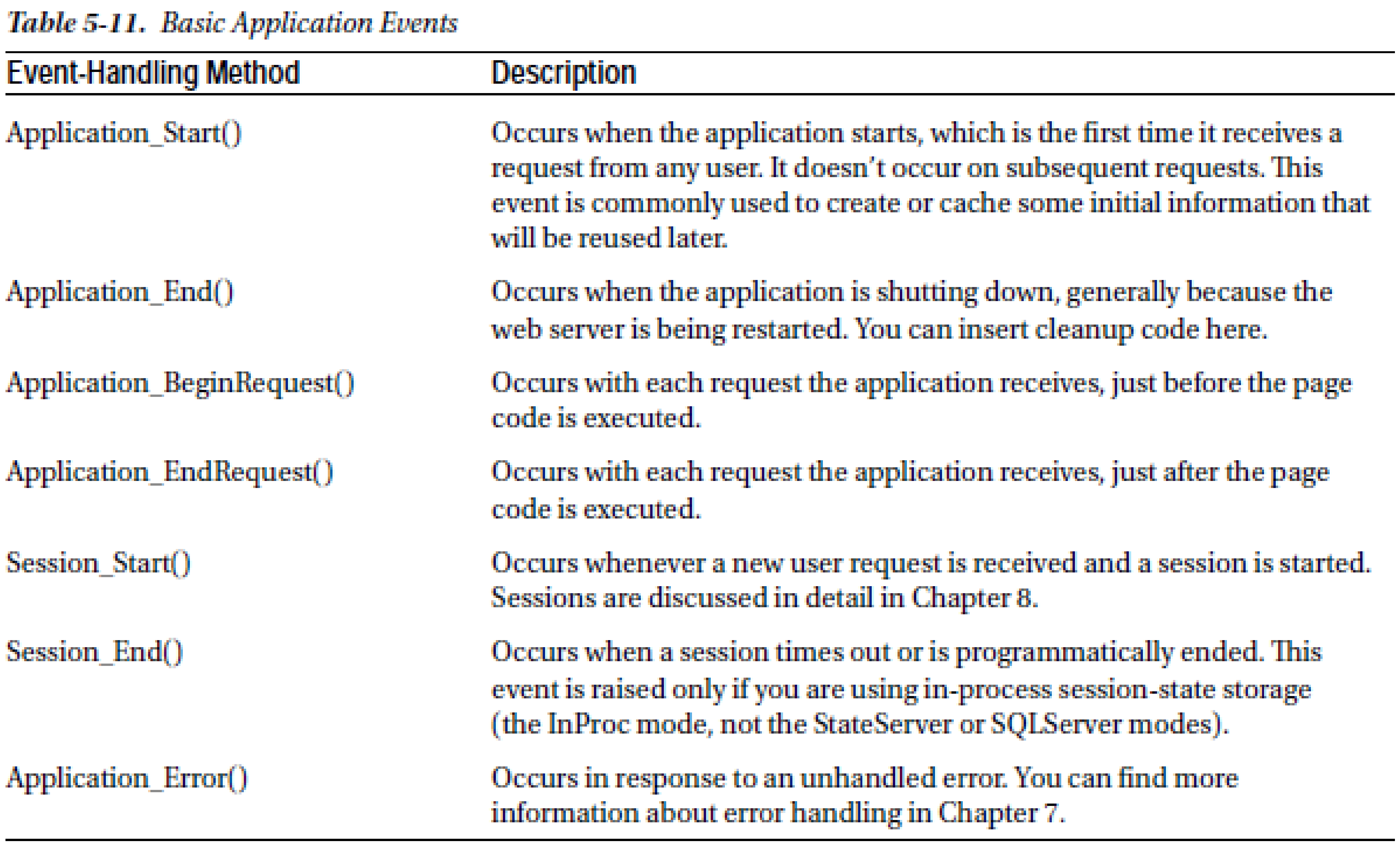


Table 5-11. Basic Application Events

Event-Handling Method	Description
Application_Start()	Occurs when the application starts, which is the first time it receives a request from any user. It doesn't occur on subsequent requests. This event is commonly used to create or cache some initial information that will be reused later.
Application_End()	Occurs when the application is shutting down, generally because the web server is being restarted. You can insert cleanup code here.
Application_BeginRequest()	Occurs with each request the application receives, just before the page code is executed.
Application_EndRequest()	Occurs with each request the application receives, just after the page code is executed.
Session_Start()	Occurs whenever a new user request is received and a session is started. Sessions are discussed in detail in Chapter 8.
Session_End()	Occurs when a session times out or is programmatically ended. This event is raised only if you are using in-process session-state storage (the InProc mode, not the StateServer or SQLServer modes).
Application_Error()	Occurs in response to an unhandled error. You can find more information about error handling in Chapter 7.

Configuring an ASP.NET Application

- <?xml version = "1.0" ?>
- <configuration>
- <appSettings > . . .</appSettings>
- <connectionStrings > . . .</connectionStrings>
- <system.web > . . .</system.web>
- </configuration>

Web Controls

Table 6-1. Basic Web Controls

Control Class	Underlying HTML Element
Label	<span>
Button	<input type="submit"> or <input type="button">
TextBox	<input type="text">, <input type="password">, or <textarea>
CheckBox	<input type="checkbox">
RadioButton	<input type="radio">
Hyperlink	<a>
LinkButton	<a> with a contained <img> tag
ImageButton	<input type="image">
Image	<img>
ListBox	<select size="X"> where X is the number of rows that are visible at once
DropDownList	<select>
CheckBoxList	A list or <table> with multiple <input type="checkbox"> tags
RadioButtonList	A list or <table> with multiple <input type="radio"> tags
BulletedList	An <ol> ordered list (numbered) or <ul> unordered list (bulleted)
Panel	<div>
Table, TableRow, and TableCell	<table>, <tr>, and <td> or <th>

The Web Control Tags

- ASP.NET tags always begin with the prefix asp: followed by the class name.
- If there is no closing tag, the tag must end with />.
- <asp:TextBox ID="txt" runat="server" />

Web Control Classes

- Web control classes are defined in the **System.Web.UI.WebControls** namespace

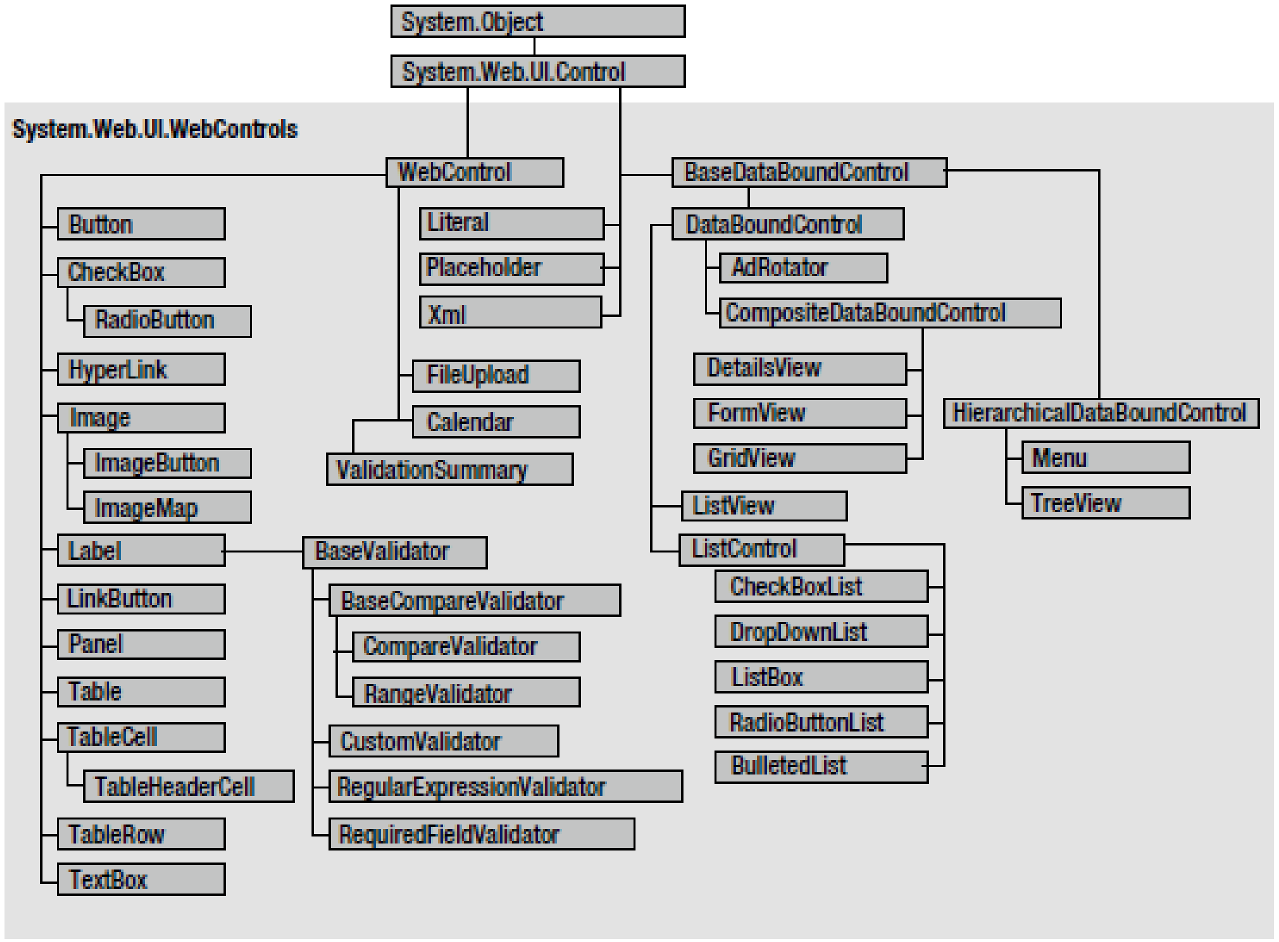


Figure 6-2. The web control hierarchy

Table 6-2. WebControl Properties

Property	Description
AccessKey	Specifies the keyboard shortcut as one letter. For example, if you set this to Y, the Alt+Y keyboard combination will automatically change focus to this web control (assuming the browser supports this feature).
BackColor, ForeColor, and BorderColor	Sets the colors used for the background, foreground, and border of the control. In most controls, the foreground color sets the text color.
BorderWidth	Specifies the size of the control border.
BorderStyle	One of the values from the BorderStyle enumeration, including Dashed, Dotted, Double, Groove, Ridge, Inset, Outset, Solid, and None.
Controls	Provides a collection of all the controls contained inside the current control. Each object is provided as a generic System.Web.UI.Control object, so you will need to cast the reference to access control-specific properties.
Enabled	When set to false, the control will be visible, but it will not be able to receive user input or focus.
EnableViewState	Set this to false to disable the automatic state management for this control. In this case, the control will be reset to the properties and formatting specified in the control tag (in the .aspx page) every time the page is posted back. If this is set to true (the default), the control uses the hidden input field to store information about its properties, ensuring that any changes you make in code are remembered.

Font	Specifies the font used to render any text in the control as a System.Web.UI.WebControls.FontInfo object.
Height and Width	Specifies the width and height of the control. For some controls, these properties will be ignored when used with older browsers.
ID	Specifies the name that you use to interact with the control in your code (and also serves as the basis for the ID that's used to name the top-level element in the rendered HTML).
Page	Provides a reference to the web page that contains this control as a System.Web.UI.Page object.
Parent	Provides a reference to the control that contains this control. If the control is placed directly on the page (rather than inside another control), it will return a reference to the page object.
TabIndex	A number that allows you to control the tab order. The control with a TabIndex of 0 has the focus when the page first loads. Pressing Tab moves the user to the control with the next lowest TabIndex, provided it is enabled. This property is supported only in Internet Explorer.
ToolTip	Displays a text message when the user hovers the mouse above the control. Many older browsers don't support this property.
Visible	When set to false, the control will be hidden and will not be rendered to the final HTML page that is sent to the client.

Units

- All the properties that use measurements, including BorderWidth, Height, and Width, require the Unit structure, which combines a numeric value with a type of measurement (pixels, percentage, and so on).
- This means when you set these properties in a control tag, you must make sure to append px (pixel) or % (for percentage) to the number to indicate the type of unit.
- <asp:Panel Height="300px" Width="50%" ID="pnl" runat="server" />

- // Convert the number 300 to a Unit object
- // representing pixels, and assign it.
- pnl.Height = Unit.Pixel(300);
- // Convert the number 50 to a Unit object
- // representing percent, and assign it.
- pnl.Width = Unit.Percentage(50);

ENUMERATIONS

- ctrl.BorderStyle = BorderStyle.Dashed;

- <asp:Label BorderStyle="Dashed" Text="Border Test" ID="lbl“ runat="server" />

Colors

- The Color property refers to a Color object from the System.Drawing namespace.
- You can create color objects in several ways:
- *Using an ARGB (alpha, red, green, blue) color value*:
  - You specify each value as an integer from 0 to 255.
  - The alpha component represents the transparency of a color, and usually you'll use 255 to make the color completely opaque.
- *Using a predefined .NET color name*:
  - You choose the correspondingly named readonly property from the Color structure.
  - These properties include the 140 HTML color names.
- *Using an HTML color name*:
  - You specify this value as a string by using the ColorTranslator class.

- using System.Drawing;
- The following code shows several ways to specify a color in code:
- // Create a color from an ARGB value
- int alpha = 255, red = 0, green = 255, blue = 0;
- ctrl.ForeColor = Color.FromArgb(alpha, red, green, blue);
- // Create a color using a .NET name
- ctrl.ForeColor = Color.Crimson;
- // Create a color from an HTML code
- ctrl.ForeColor = ColorTranslator.FromHtml("Blue");

- <asp:TextBox ForeColor="Red" Text="Test" ID="txt" runat="server" />
- <asp:TextBox ForeColor="#ff50ff" Text="Test"
- ID="txt" runat="server" />

FONTS

- The Font property actually references a full FontInfo object, which is defined in the System.Web.UI.WebControls namespace.
- Every FontInfo object has several properties that define its name, size, and style.

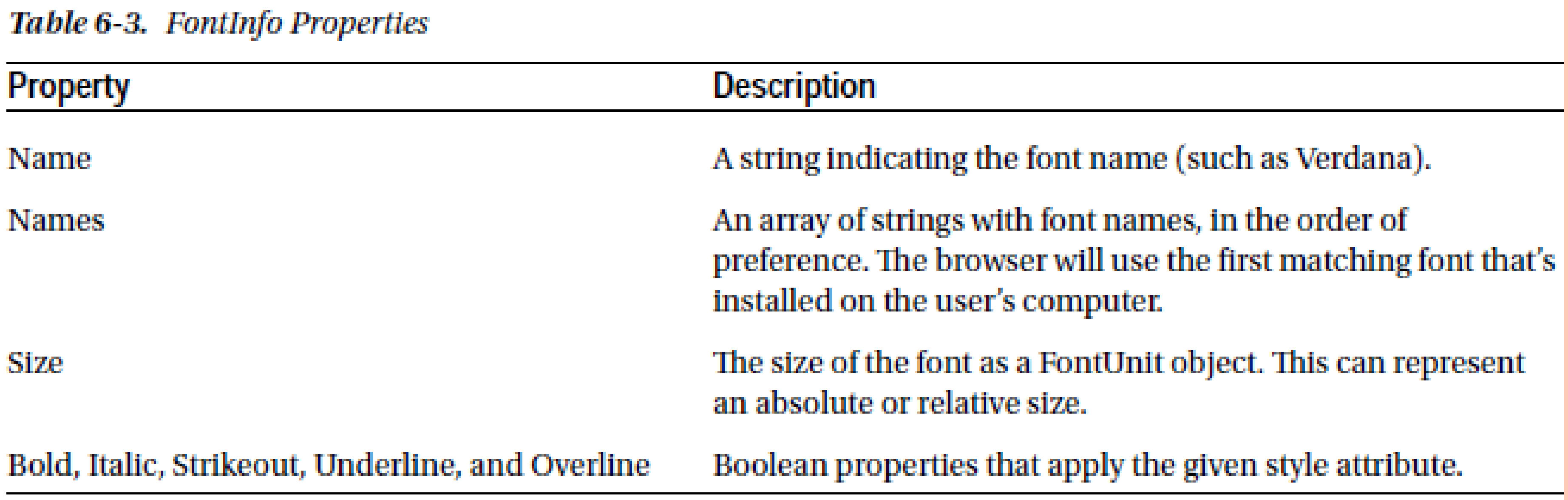


Table 6-3. FontInfo Properties

Property	Description
Name	A string indicating the font name (such as Verdana).
Names	An array of strings with font names, in the order of preference. The browser will use the first matching font that's installed on the user's computer.
Size	The size of the font as a FontUnit object. This can represent an absolute or relative size.
Bold, Italic, Strikeout, Underline, and Overline	Boolean properties that apply the given style attribute.

- ctrl.Font.Name = "Verdana";
- ctrl.Font.Bold = true;

- // Specifies a relative size.
- ctrl.Font.Size = FontUnit.Small;
- // Specifies an absolute size of 14 pixels.
- ctrl.Font.Size = FontUnit.Point(14);

- <asp:TextBox Font-Name="Tahoma" Font-Size="40" Text="Size Test" ID="txt“ runat="server" />
- Or you could set a relative size like this:
- <asp:TextBox Font-Name="Tahoma" Font-Size="Large" Text="Size Test“ ID="txt" runat="server" />
- <asp:TextBox Font-Names="Verdana,Tahoma,Arial"
- Text="Size Test" ID="txt" runat="server" />

Focus

- <form DefaultFocus="TextBox2" runat="server">

Control PreFIxes

- Button: cmd (or btn)
- • CheckBox: chk
- • Image: img
- • Label: lbl
- • List control: lst
- • Panel: pnl
- • RadioButton: opt
- • TextBox: txt

List Controls

- The list controls include
  - ListBox
  - DropDownList
  - CheckBoxList
  - RadioButtonList
  - BulletedList

  - SelectedIndex
  - SelectedItem
  - Text (the displayed content)
  - Value (the hidden value from the HTMLmarkup)
  - Selected
  - SelectionMode
  - Items collection

Table 6-4. Added BulletedList Properties

Property	Description
BulletStyle	Determines the type of list. Choose from Numbered (1, 2, 3, . . .); LowerAlpha (a, b, c, . . .) and UpperAlpha (A, B, C, . . .); LowerRoman (i, ii, iii, ; . . .) and UpperRoman (I, II, III, . . .); and the bullet symbols Disc, Circle, Square, or CustomImage (in which case you must set the BulletImageUrl property).
BulletImageUrl	If the BulletStyle is set to CustomImage, this points to the image that is placed to the left of each item as a bullet.
FirstBulletNumber	In an ordered list (using the Numbered, LowerAlpha, UpperAlpha, LowerRoman, and UpperRoman styles), this sets the first value. For example, if you set FirstBulletNumber to 3, the list might read 3, 4, 5 (for Numbered) or C, D, E (for UpperAlpha).
DisplayMode	Determines whether the text of each item is rendered as text (use Text, the default) or a hyperlink (use LinkButton or HyperLink). The difference between LinkButton and HyperLink is how they treat clicks. When you use LinkButton, the BulletedList fires a Click event that you can react to on the server to perform the navigation. When you use HyperLink, the BulletedList doesn't fire the Click event—instead, it treats the text of each list item as a relative or absolute URL, and renders them as ordinary HTML hyperlinks. When the user clicks an item, the browser attempts to navigate to that URL.

Table Controls

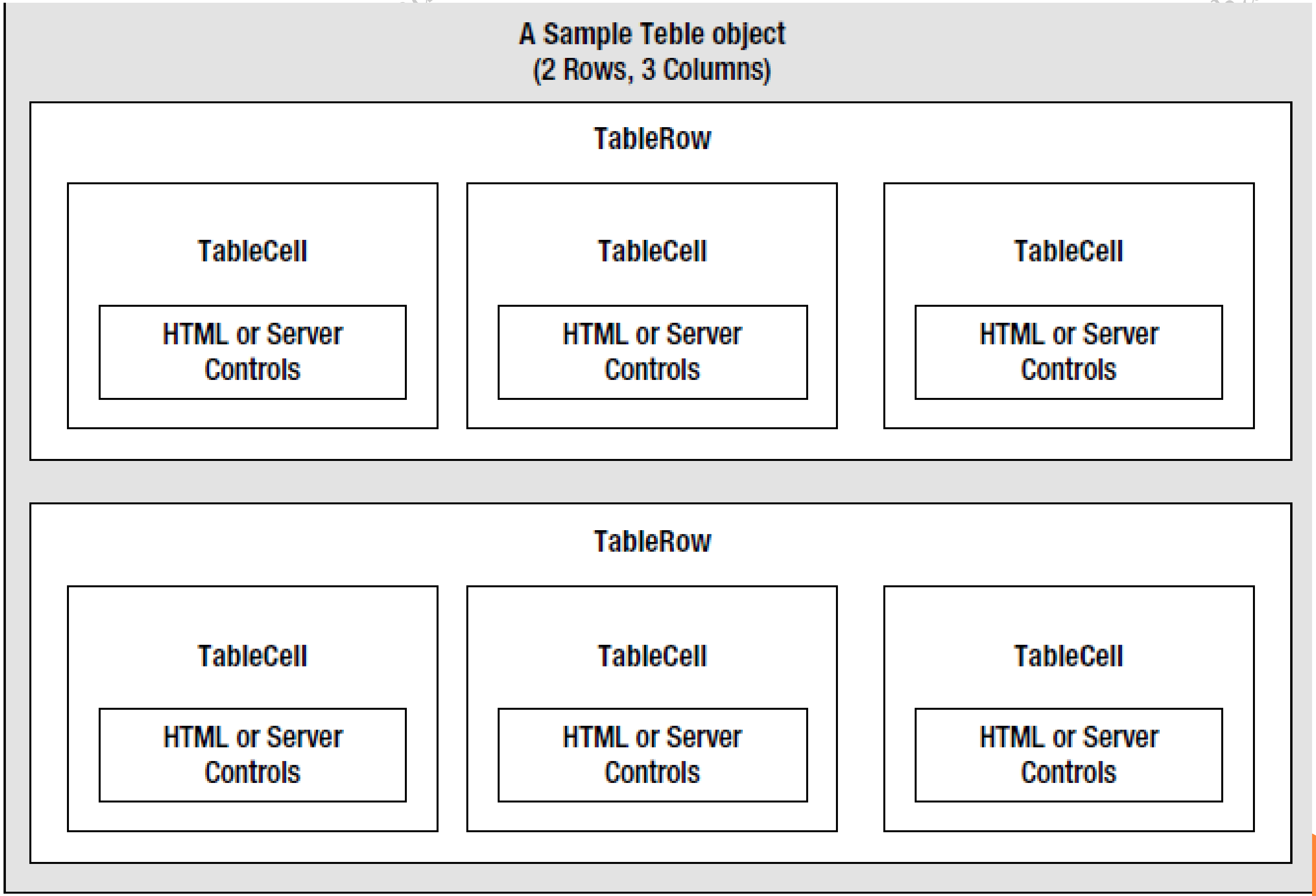


Figure 6-7. *Table control containment*

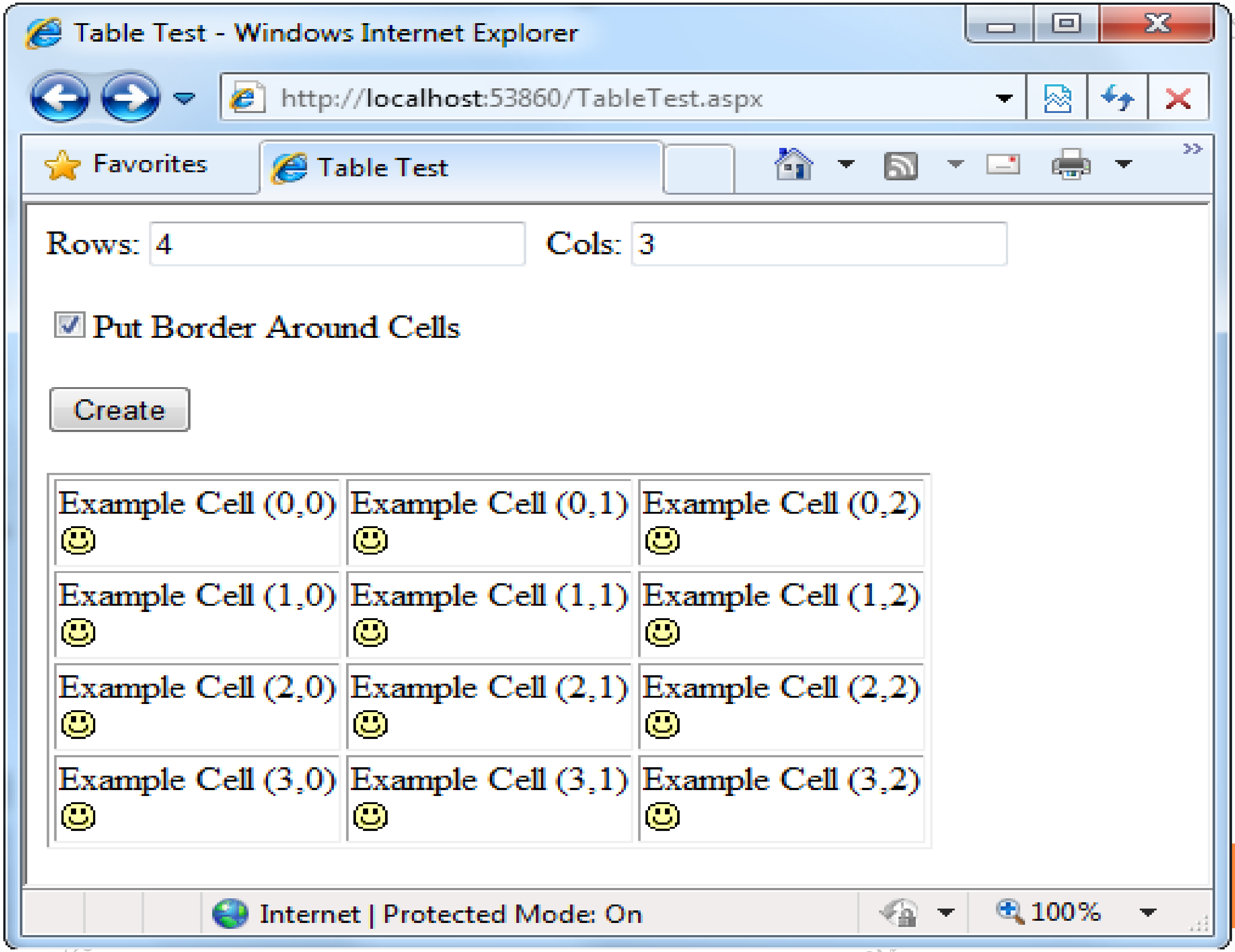


Table Test - Windows Internet Explorer

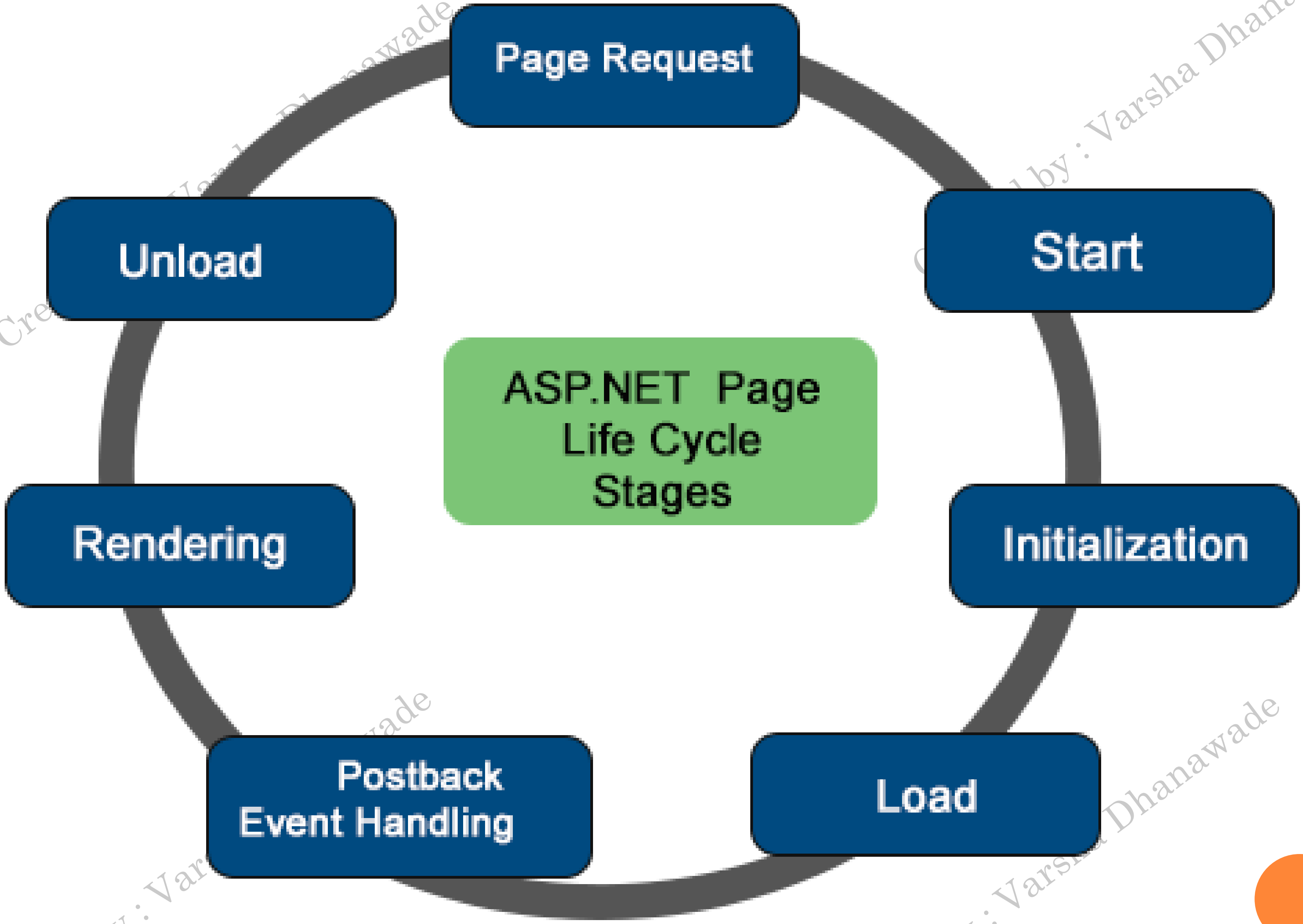
http://localhost:53860/TableTest.aspx

Rows: 4 Cols: 3

[✓] Put Border Around Cells

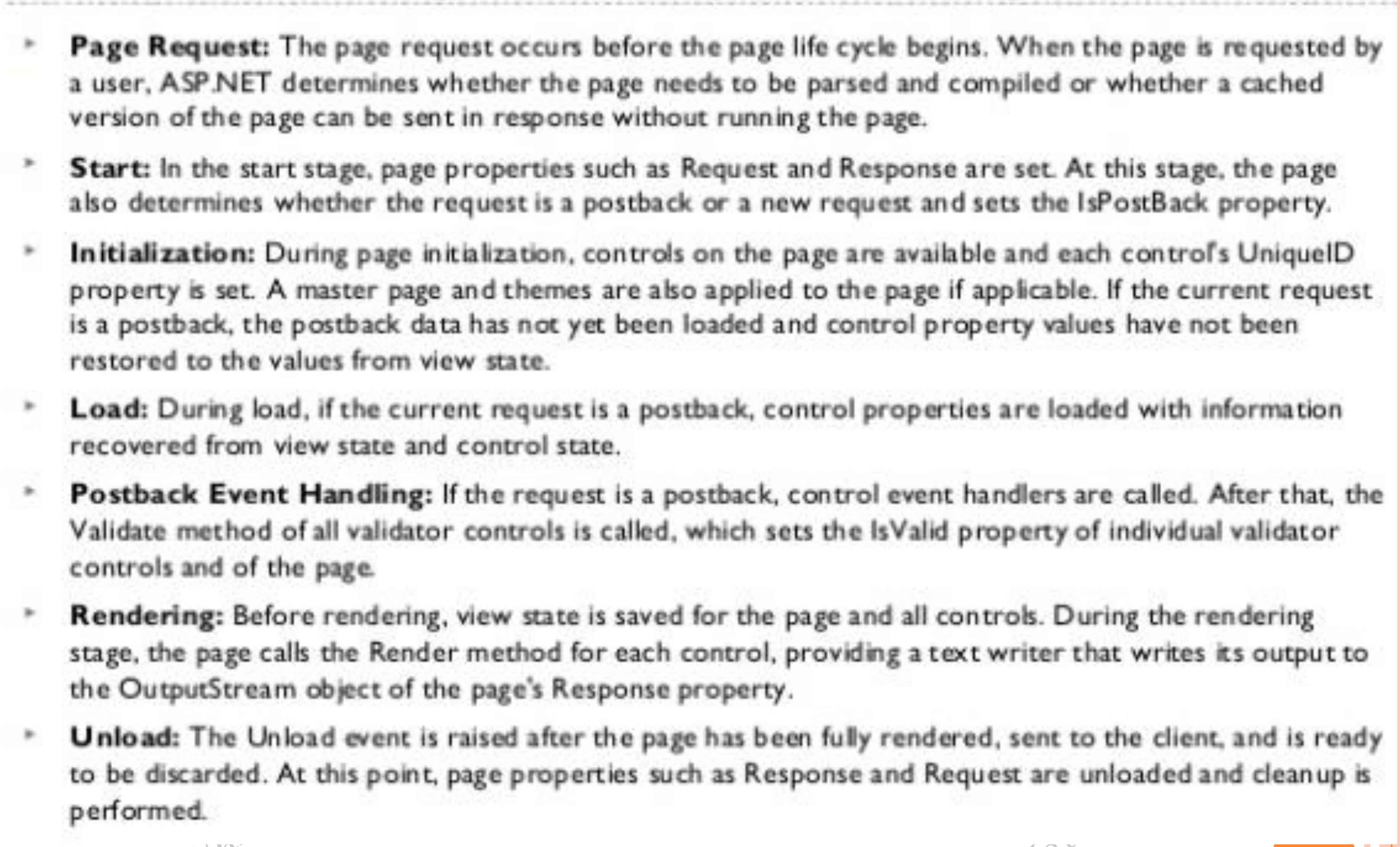
Create

Example Cell (0,0) ☺	Example Cell (0,1) ☺	Example Cell (0,2) ☺
Example Cell (1,0) ☺	Example Cell (1,1) ☺	Example Cell (1,2) ☺
Example Cell (2,0) ☺	Example Cell (2,1) ☺	Example Cell (2,2) ☺
Example Cell (3,0) ☺	Example Cell (3,1) ☺	Example Cell (3,2) ☺


Page Request
Start
Initialization
Load
Postback Event Handling
Rendering
Unload
ASP.NET Page Life Cycle Stages

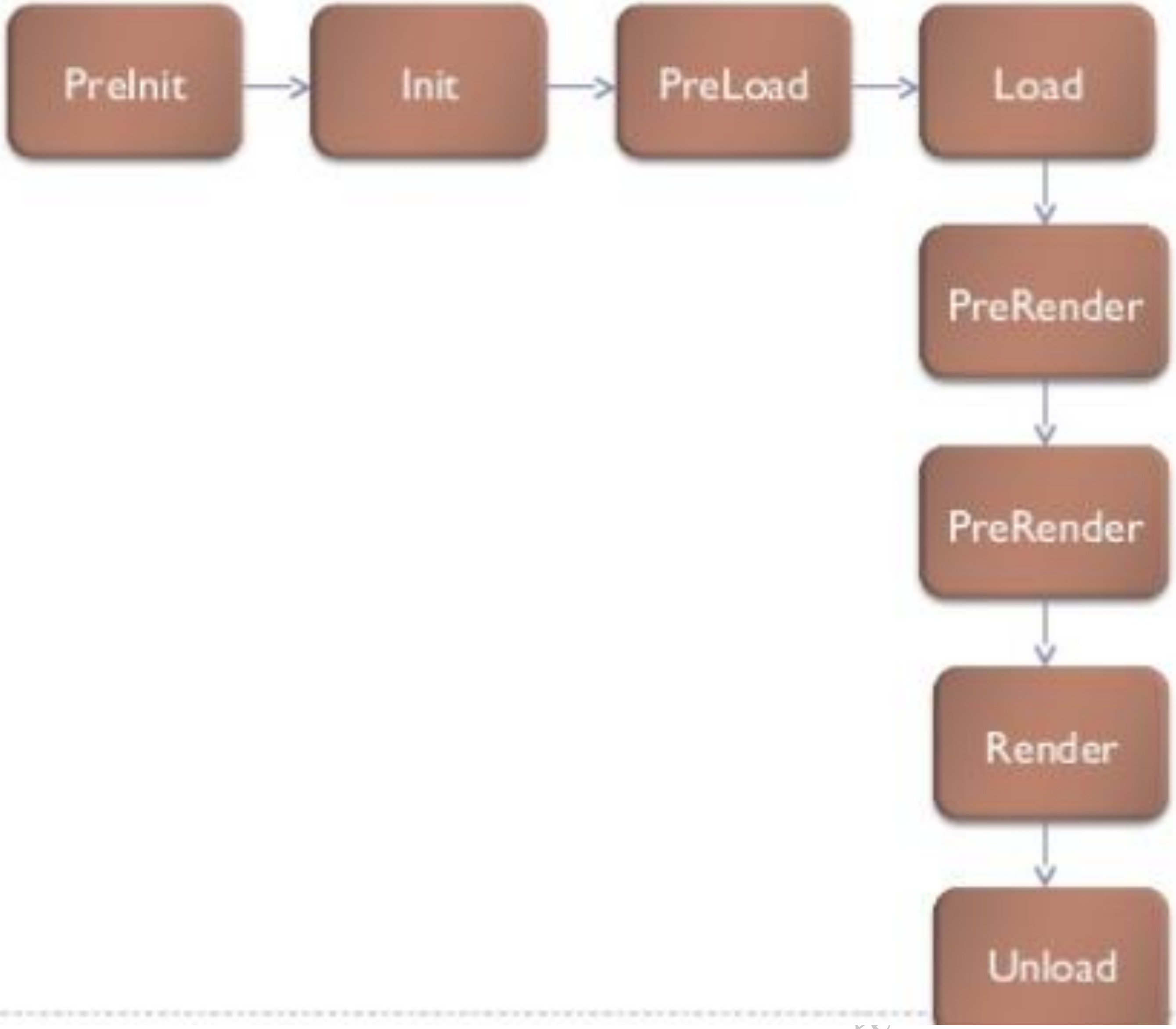
- Page Request
- Page Start
  - Request and Response
- Page Initialization
  - Control Initilization
  - Master Page and themes
- Page Load
- Validation
- Postback event handling
  - Control Event handlers
- Page Rendering
  - ViewState
  - Respose object is ready
- Page Unload
  - Unwanted objects are removed

ASP .Net Page Life Cycle: Stages



- **Page Request:** The page request occurs before the page life cycle begins. When the page is requested by a user, ASP.NET determines whether the page needs to be parsed and compiled or whether a cached version of the page can be sent in response without running the page.
- **Start:** In the start stage, page properties such as Request and Response are set. At this stage, the page also determines whether the request is a postback or a new request and sets the IsPostBack property.
- **Initialization:** During page initialization, controls on the page are available and each control's UniqueID property is set. A master page and themes are also applied to the page if applicable. If the current request is a postback, the postback data has not yet been loaded and control property values have not been restored to the values from view state.
- **Load:** During load, if the current request is a postback, control properties are loaded with information recovered from view state and control state.
- **Postback Event Handling:** If the request is a postback, control event handlers are called. After that, the Validate method of all validator controls is called, which sets the IsValid property of individual validator controls and of the page.
- **Rendering:** Before rendering, view state is saved for the page and all controls. During the rendering stage, the page calls the Render method for each control, providing a text writer that writes its output to the OutputStream object of the page's Response property.
- **Unload:** The Unload event is raised after the page has been fully rendered, sent to the client, and is ready to be discarded. At this point, page properties such as Response and Request are unloaded and cleanup is performed.

ASP .Net Page Life Cycle: Events



- PreInit
- Init
- PreLoad
- Load
- PreRender
- Render
- Unload

ASP .Net Page Life Cycle: Events

- **PreInit:** Raised after the start stage is complete and before the initialization stage begins. Check the IsPostBack property to determine whether this is the first time the page is being processed. Set a master page dynamically. Set the Theme property dynamically.
- **Init:** Raised after all controls have been initialized and any skin settings have been applied. The Init event of individual controls occurs before the Init event of the page. Use this event to read or initialize control properties.
- **PreLoad:** Raised after the page loads view state for itself and all controls, and after it processes postback data that is included with the Request instance.
- **Load:** The Page object calls the OnLoad method on the Page object, and then recursively does the same for each child control until the page and all controls are loaded. The Load event of individual controls occurs after the Load event of the page. Use the OnLoad event method to set properties in controls and to establish database connections.
- **Control events:** Use these events to handle specific control events, such as a Button control's Click event or a TextBox control's TextChanged event.
- **PreRender:** Raised after the Page object has created all controls that are required in order to render the page, including child controls of composite controls. The Page object raises the PreRender event on the Page object, and then recursively does the same for each child control. The PreRender event of individual controls occurs after the PreRender event of the page.
- **Render:** This is not an event; instead, at this stage of processing, the Page object calls this method on each control. All ASP.NET Web server controls have a Render method that writes out the control's markup to send to the browser. If you create a custom control, you typically override this method to output the control's markup. However, if your custom control incorporates only standard ASP.NET Web server controls and no custom markup, you do not need to override the Render method.
- **Unload:** Raised for each control and then for the page.

- **Autopostback** is Boolean value that specifies whether the control is automatically posted back to the server when the contents change or not. Default is false.
- ASP.NET also adds two hidden input fields that are used to pass information back to the server.
- This information consists of the ID of the control that raised the event and any additional information that might be relevant.
- These fields are initially empty, as shown here:
- <input type="hidden" name="__EVENTTARGET" ID="__EVENTTARGET" value="" />
- <input type="hidden" name="__EVENTARGUMENT" ID="__EVENTARGUMENT" value="" />

- The __doPostBack() function has the responsibility of setting these values with the appropriate information about the event and then submitting the form.
- The __doPostBack() function is shown here:
- <script language="text/javascript">
- function __doPostBack(eventTarget, eventArgument) {
- if (!theForm.onsubmit | | (theForm.onsubmit() != false)) {
- theForm.__EVENTTARGET.value = eventTarget;
- theForm.__EVENTARGUMENT.value = eventArgument;
- theForm.submit();
- }
- ...
- }
- </script>

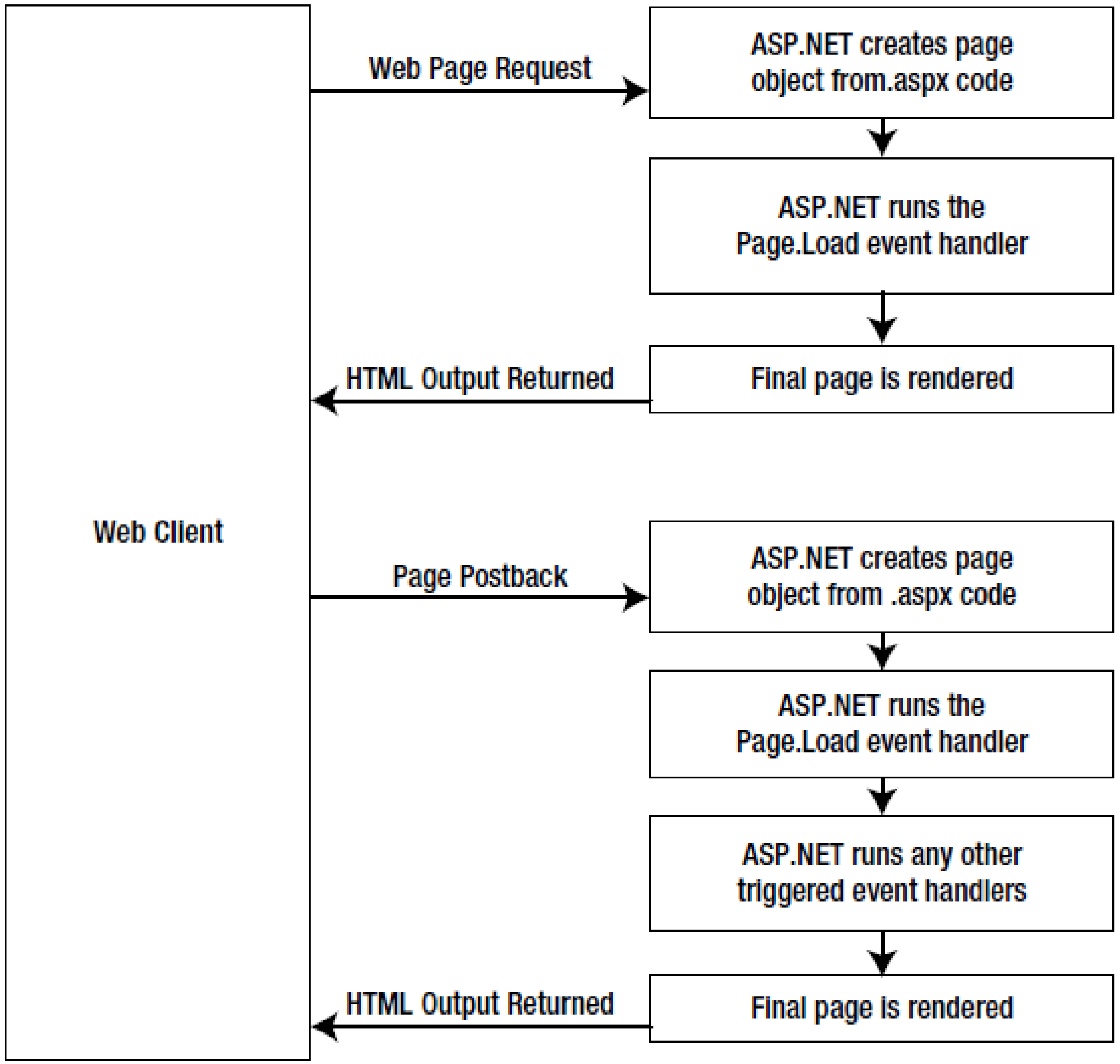
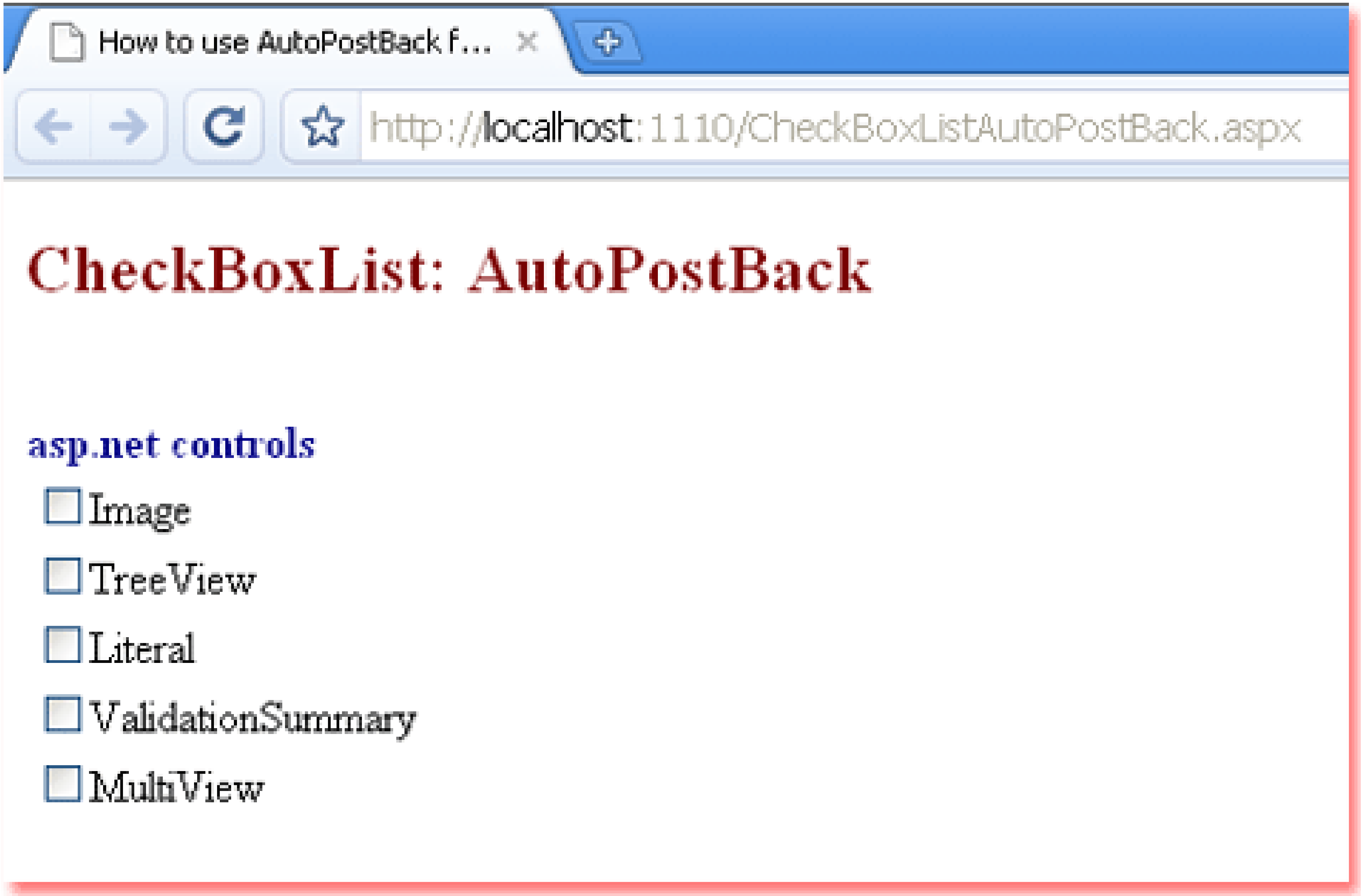

Web Client
Web Page Request
ASP.NET creates page object from.aspx code
ASP.NET runs the Page.Load event handler
Final page is rendered
HTML Output Returned
Page Postback
ASP.NET creates page object from .aspx code
ASP.NET runs the Page.Load event handler
ASP.NET runs any other triggered event handlers
Final page is rendered
HTML Output Returned

Figure 6-11. *The page-processing sequence*


How to use AutoPostBack f...

http://localhost:1110/CheckBoxListAutoPostBack.aspx

CheckBoxList: AutoPostBack

asp.net controls

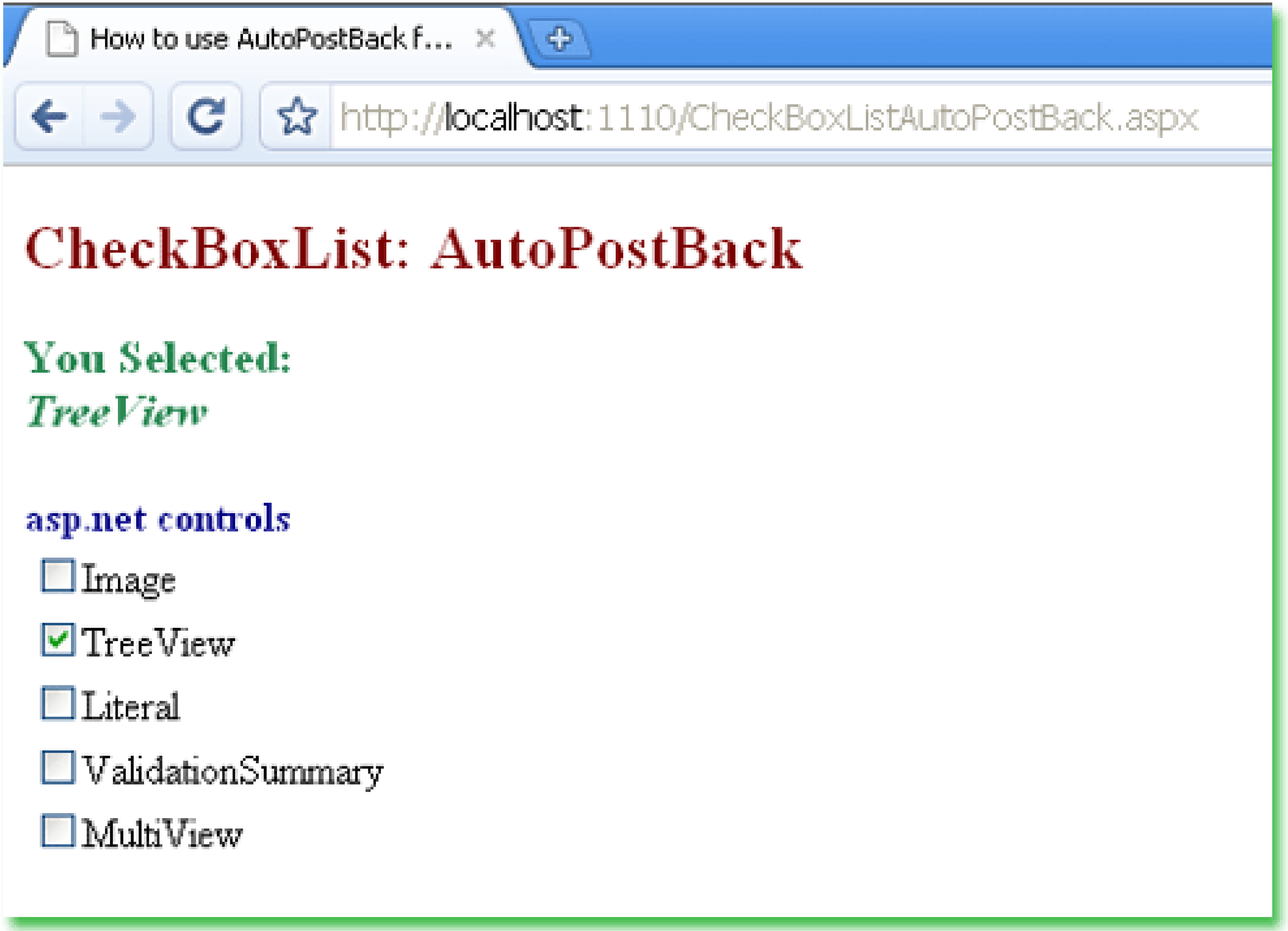
☐ Image

☐ TreeView

☐ Literal

☐ ValidationSummary

☐ MultiView


How to use AutoPostBack f...

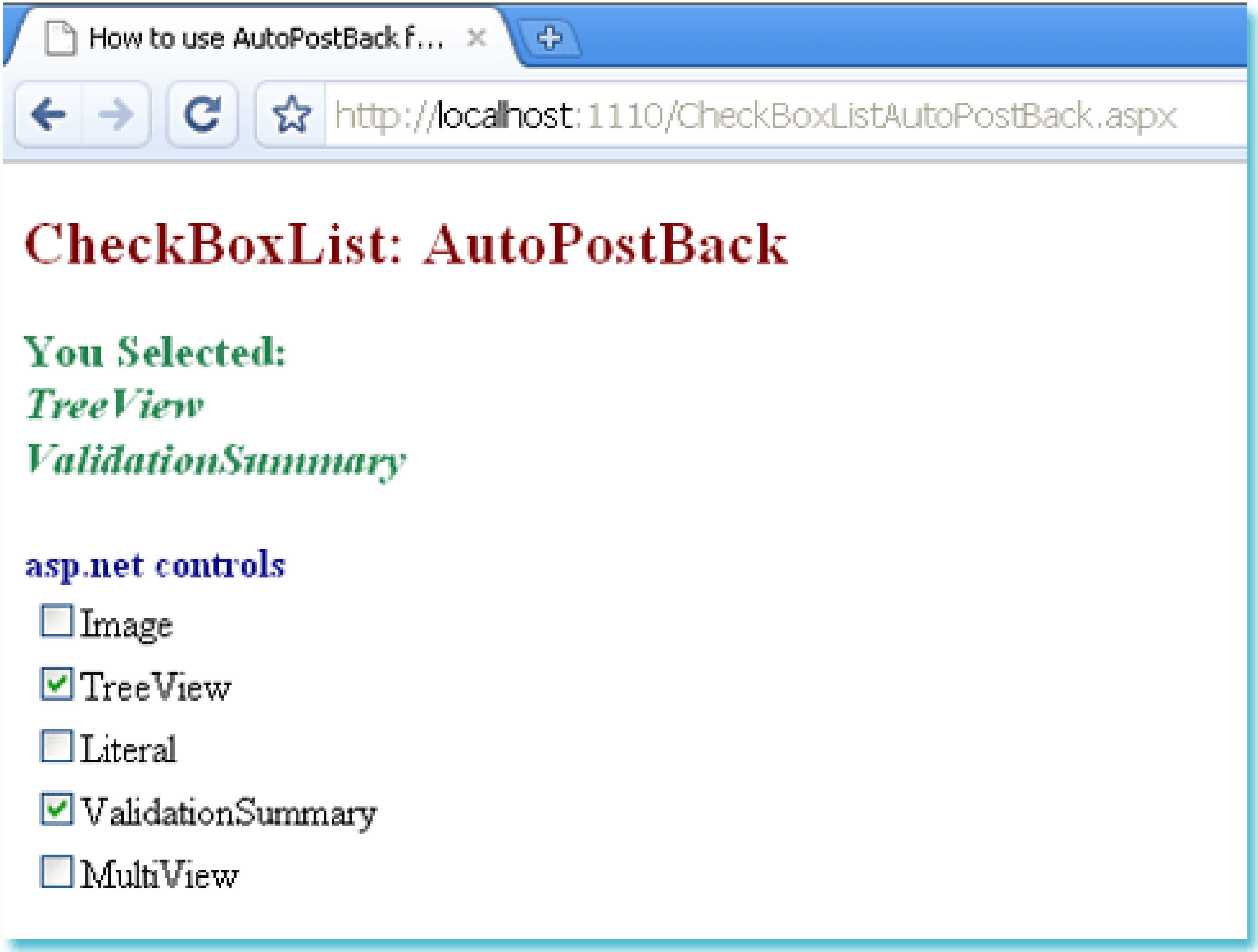
http://localhost:1110/CheckBoxListAutoPostBack.aspx

CheckBoxList: AutoPostBack

You Selected:
TreeView

asp.net controls

- [ ] Image
- [✓] TreeView
- [ ] Literal
- [ ] ValidationSummary
- [ ] MultiView



How to use AutoPostBack f...

http://localhost:1110/CheckBoxListAutoPostBack.aspx

CheckBoxList: AutoPostBack

You Selected:

TreeView

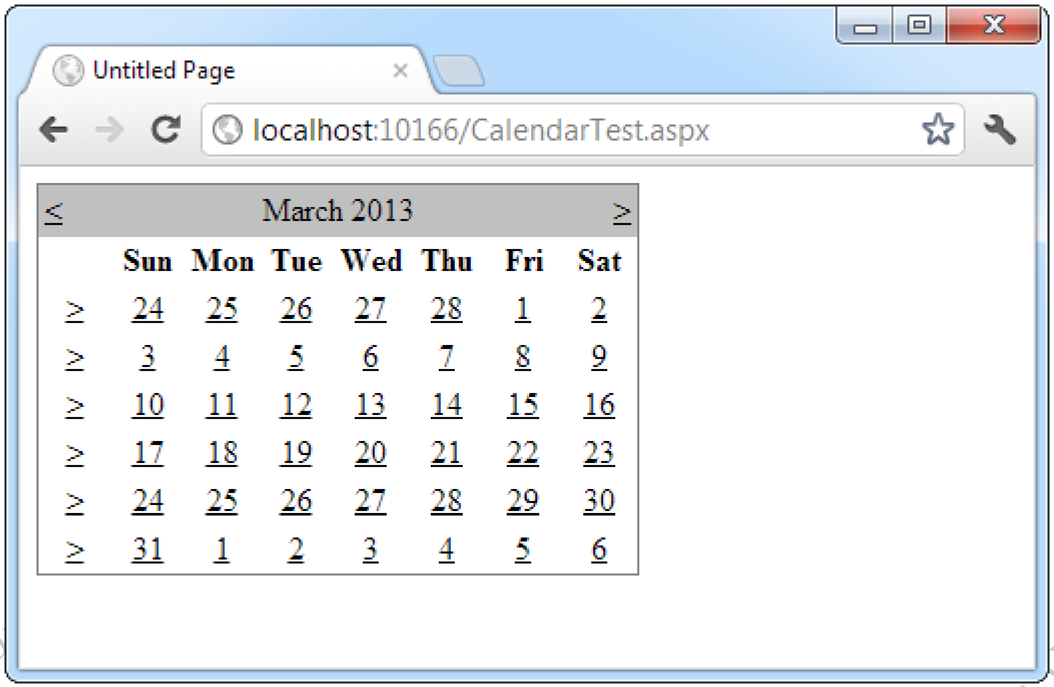
ValidationSummary

asp.net controls

- [ ] Image
- [✓] TreeView
- [ ] Literal
- [✓] ValidationSummary
- [ ] MultiView

Rich Controls

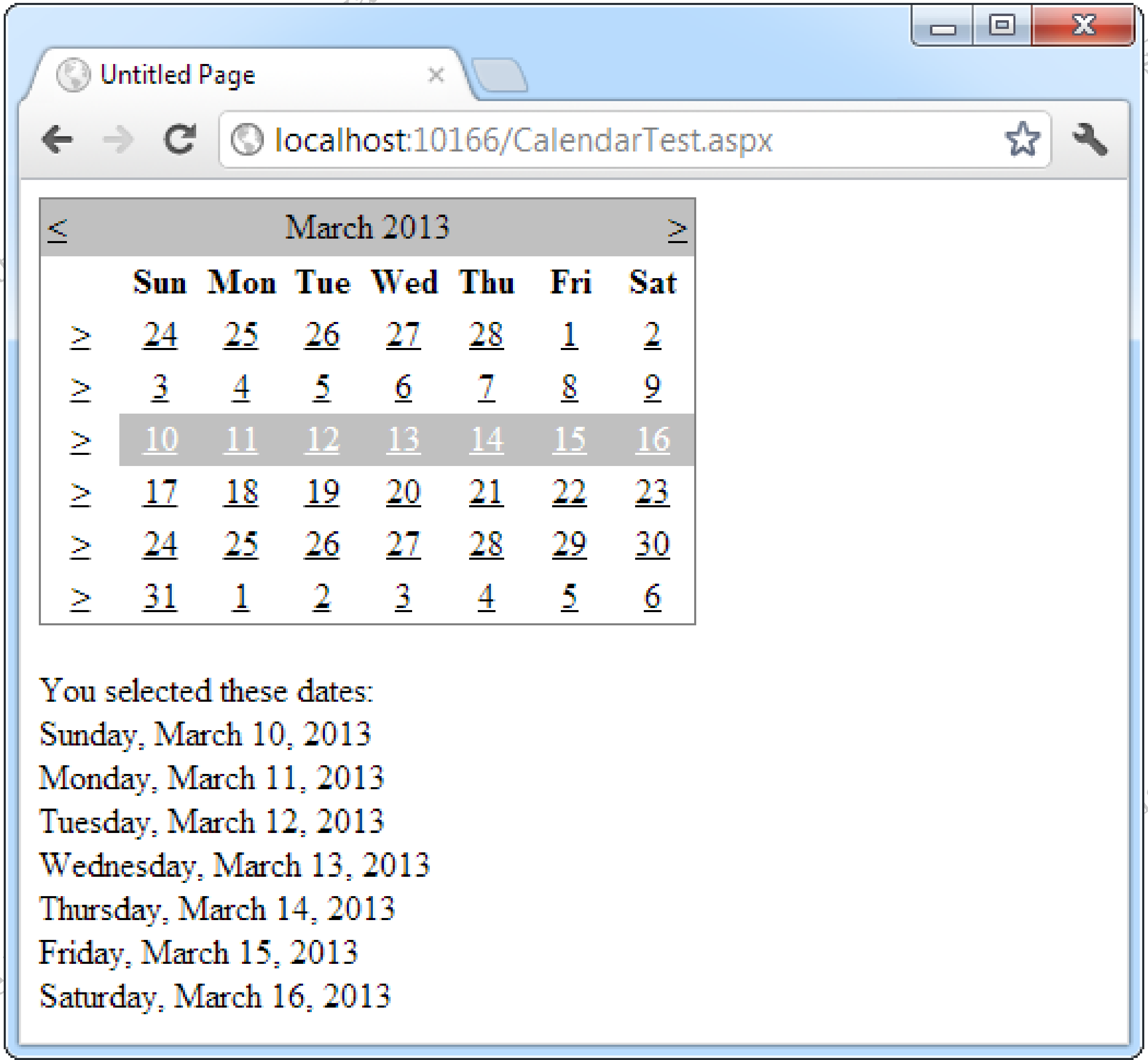
- The Calendar
  - The *Calendar control* presents a miniature calendar that you can place in any web page.
  - <asp:Calendar id="MyCalendar" runat="server" />



<			March 2013				>
	Sun	Mon	Tue	Wed	Thu	Fri	Sat
>	24	25	26	27	28	1	2
>	3	4	5	6	7	8	9
>	10	11	12	13	14	15	16
>	17	18	19	20	21	22	23
>	24	25	26	27	28	29	30
>	31	1	2	3	4	5	6

- lblDates.Text = "You selected these dates:<br />";
- foreach (DateTime dt in MyCalendar.SelectedDates)
- {
- lblDates.Text += dt.ToLongDateString() + "<br />";
- }

Figure 10-2. Selecting multiple dates



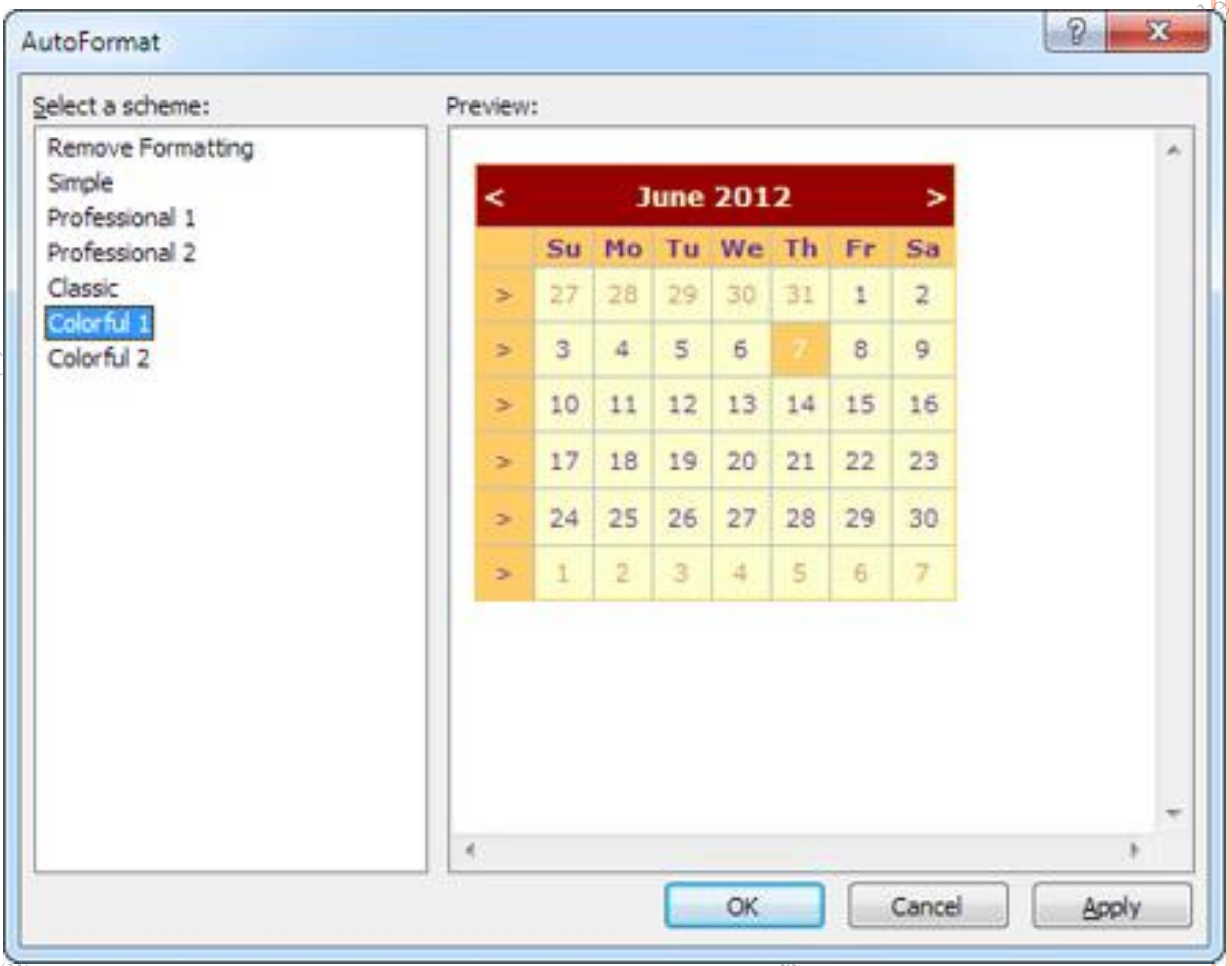
<			March 2013				>
	Sun	Mon	Tue	Wed	Thu	Fri	Sat
>	24	25	26	27	28	1	2
>	3	4	5	6	7	8	9
>	10	11	12	13	14	15	16
>	17	18	19	20	21	22	23
>	24	25	26	27	28	29	30
>	31	1	2	3	4	5	6

You selected these dates:
Sunday, March 10, 2013
Monday, March 11, 2013
Tuesday, March 12, 2013
Wednesday, March 13, 2013
Thursday, March 14, 2013
Friday, March 15, 2013
Saturday, March 16, 2013

FORMATTING THE CALENDAR

Table 10-1. Properties for Calendar Styles

Member	Description
DayHeaderStyle	The style for the section of the Calendar that displays the days of the week (as column headers).
DayStyle	The default style for the dates in the current month.
NextPrevStyle	The style for the navigation controls in the title section that move from month to month.
OtherMonthDayStyle	The style for the dates that aren't in the currently displayed month. These dates are used to "fill in" the calendar grid. For example, the first few cells in the topmost row may display the last few days from the previous month.
SelectedDayStyle	The style for the selected dates on the calendar.
SelectorStyle	The style for the week and month date selection controls.
TitleStyle	The style for the title section.
TodayDayStyle	The style for the date designated as today (represented by the TodaysDate property of the Calendar control).
WeekendDayStyle	The style for dates that fall on the weekend.



Remove Formatting
Simple
Professional 1
Professional 2
Classic
[✓ Colorful 1]
Colorful 2

<			June 2012				>
	Su	Mo	Tu	We	Th	Fr	Sa
>	27	28	29	30	31	1	2
>	3	4	5	6	7	8	9
>	10	11	12	13	14	15	16
>	17	18	19	20	21	22	23
>	24	25	26	27	28	29	30
>	1	2	3	4	5	6	7

Restricting Dates

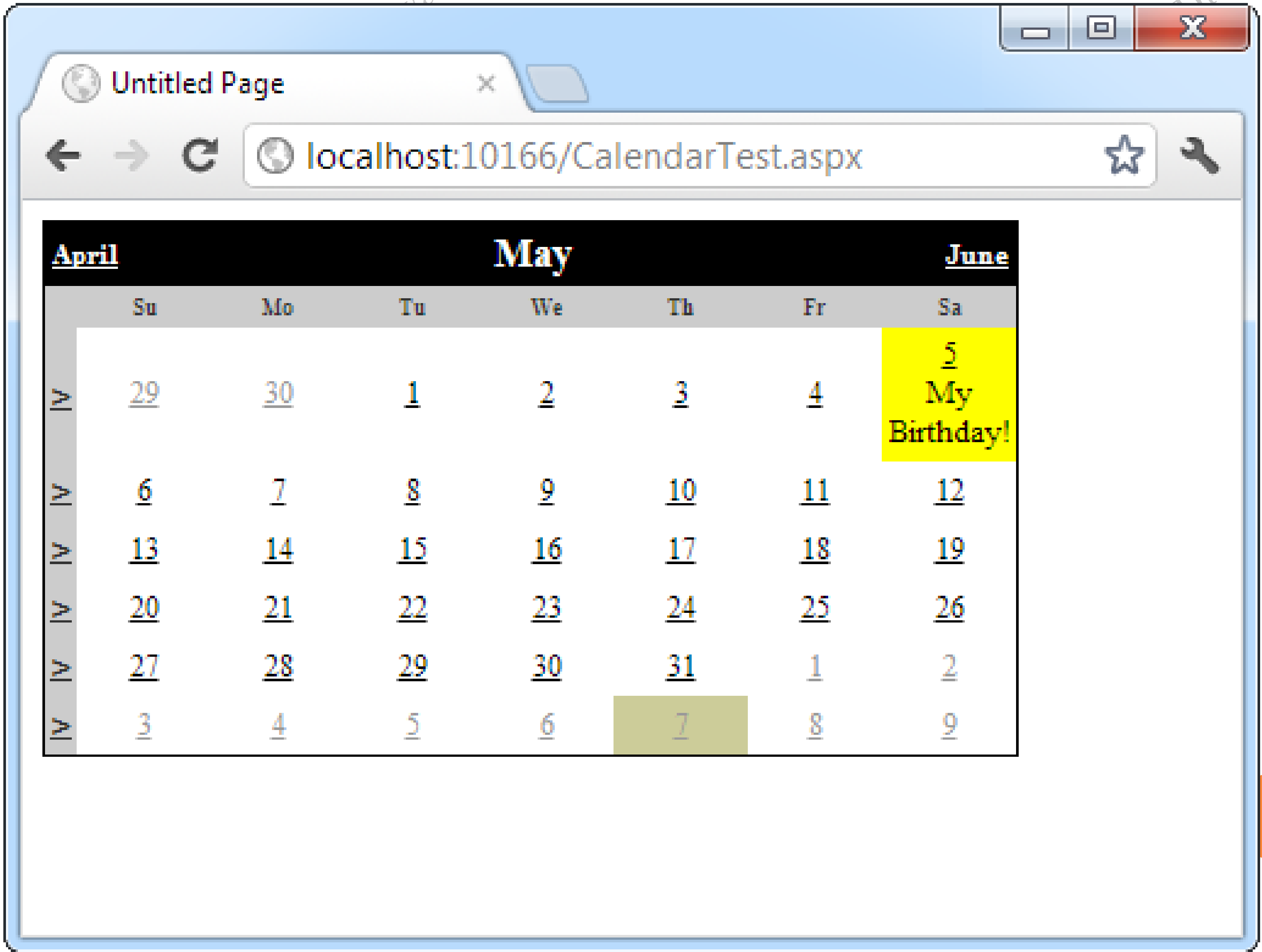
- protected void MyCalendar_DayRender(Object source, DayRenderEventArgs e)
- {
- // Restrict dates after the year 2013 and those on the weekend.
- if (e.Day.IsWeekend | | e.Day.Date.Year > 2013)
- {
- e.Day.IsSelectable = false;
- }

Table 10-2. CalendarDay Properties

Property	Description
Date	The DateTime object that represents this date.
IsWeekend	True if this date falls on a Saturday or Sunday.
IsToday	True if this value matches the Calendar.TodaysDate property, which is set to the current day by default.
IsOtherMonth	True if this date doesn't belong to the current month but is displayed to fill in the first or last row. For example, this might be the last day of the previous month or the next day of the following month.
IsSelectable	Allows you to configure whether the user can select this day.

- protected void MyCalendar_DayRender(Object source, DayRenderEventArgs e)
- {
- // Check for May 5 in any year, and format it.
- if (e.Day.Date.Day == 5 && e.Day.Date.Month == 5)
- {
- e.Cell.BackColor = System.Drawing.Color.Yellow;
- // Add some static text to the cell.
- Label lbl = new Label();
- lbl.Text = “<br />My Birthday!”;
- e.Cell.Controls.Add(lbl);
- }
- }

FIGURE 10-4. HIGHLIGHTING A DAY



April			May				June
	Su	Mo	Tu	We	Th	Fr	Sa
>	29	30	1	2	3	4	5 My Birthday!
>	6	7	8	9	10	11	12
>	13	14	15	16	17	18	19
>	20	21	22	23	24	25	26
>	27	28	29	30	31	1	2
>	3	4	5	6	7	8	9

```
protected void MyCalendar_SelectionChanged(Object source, EventArgs e)
{
lstTimes.Items.Clear();
switch (MyCalendar.SelectedDate.DayOfWeek)
{
case DayOfWeek.Monday:
// Apply special Monday schedule.
lstTimes.Items.Add("10:00");
lstTimes.Items.Add("10:30");
lstTimes.Items.Add("11:00");
break;
default:
lstTimes.Items.Add("10:00");
lstTimes.Items.Add("10:30");
lstTimes.Items.Add("11:00");
lstTimes.Items.Add("11:30");
lstTimes.Items.Add("12:00");
lstTimes.Items.Add("12:30");
break;
}
}
```

Table 10-3. *Calendar Members*

Member	Description
Caption and CaptionAlign	Gives you an easy way to add a title to the calendar. By default, the caption appears at the top of the title area, just above the month heading. However, you can control this to some extent with the CaptionAlign property. Use Left or Right to keep the caption at the top but move it to one side or the other, and use Bottom to place the caption under the calendar.
CellPadding	ASP.NET creates a date in a separate cell of an invisible table. CellPadding is the space, in pixels, between the border of each cell and its contents.
CellSpacing	The space, in pixels, between cells in the same table.
DayNameFormat	Determines how days are displayed in the calendar header. Valid values are Full (as in Sunday), FirstLetter (S), FirstTwoLetters (Su), and Short (Sun), which is the default.
FirstDayOfWeek	Determines which day is displayed in the first column of the calendar. The values are any day name from the FirstDayOfWeek enumeration (such as Sunday). By default, this is Sunday.
NextMonthText and PrevMonthText	Sets the text that the user clicks to move to the next or previous month. These navigation links appear at the top of the calendar and are the greater-than (>) and less-than (<) signs by default. This setting is applied only if NextPrevFormat is set to CustomText.

NextPrevFormat	Sets the text that the user clicks to move to the next or previous month. This can be FullMonth (for example, December), ShortMonth (Dec), or CustomText, in which case the NextMonthText and PrevMonthText properties are used. CustomText is the default.
SelectedDate and SelectedDates	Sets or gets the currently selected date as a DateTime object. You can specify this in the control tag in a format like this: `12:00:00 AM, 12/31/2010` (depending on your computer's regional settings). If you allow multiple date selection, the SelectedDates property will return a collection of DateTime objects, one for each selected date. You can use collection methods such as Add, Remove, and Clear to change the selection.
SelectionMode	Determines how many dates can be selected at once. The default is Day, which allows one date to be selected. Other options include DayWeek (a single date or an entire week) or DayWeekMonth (a single date, entire week, or entire month). You have no way to allow the user to select multiple noncontiguous dates. You also have no way to allow larger selections without also including smaller selections. (For example, if you allow full months to be selected, you must also allow week selection and individual day selection.)
SelectMonthText and SelectWeekText	The text shown for the link that allows the user to select an entire month or week. These properties don't apply if the SelectionMode is Day.

Table 10-3. *(continued)*

Member	Description
ShowDayHeader, ShowGridLines, ShowNextPrevMonth, and ShowTitle	These Boolean properties allow you to configure whether various parts of the calendar are shown, including the day titles, gridlines between every day, the previous/next month navigation links, and the title section. Note that hiding the title section also hides the next and previous month navigation controls.
TitleFormat	Configures how the month is displayed in the title area. Valid values include Month and MonthYear (the default).
TodaysDate	Sets which day should be recognized as the current date and formatted with the TodayDayStyle. This defaults to the current day on the web server.
VisibleDate	Gets or sets the date that specifies what month will be displayed in the calendar. This allows you to change the calendar display without modifying the current date selection.
DayRender event	Occurs once for each day that is created and added to the currently visible month before the page is rendered. This event gives you the opportunity to apply special formatting, add content, or restrict selection for an individual date cell. Keep in mind that days can appear in the calendar even when they don't fall in the current month, provided they fall close to the end of the previous month or close to the start of the following month.
SelectionChanged event	Occurs when the user selects a day, a week, or an entire month by clicking the date selector controls.
VisibleMonthChanged event	Occurs when the user clicks the next or previous month navigation controls to move to another month.

THE ADROTATOR

- provides a graphic on a page that is chosen randomly from a group of possible images.
- Every time the page is requested, an image is selected at random and displayed, which is the *rotation* indicated by the name AdRotator.
- One use of the AdRotator is to show banner-style advertisements on a page, but you can use it anytime you want to vary an image randomly.

The Advertisement File

- The AdRotator stores its list of image files in an XML file. This file uses the format shown here:

```
<Advertisements>
        <Ad>
                <ImageUrl>prosetech.jpg</ImageUrl>
                <NavigateUrl>http://www.prosetech.com</NavigateUrl>
                <AlternateText>ProseTech Site</AlternateText>
                <Impressions>1</Impressions>
                <Keyword>Computer</Keyword>
        </Ad>
</Advertisements>
```

- AdRotator control picks at random from the list of advertisements

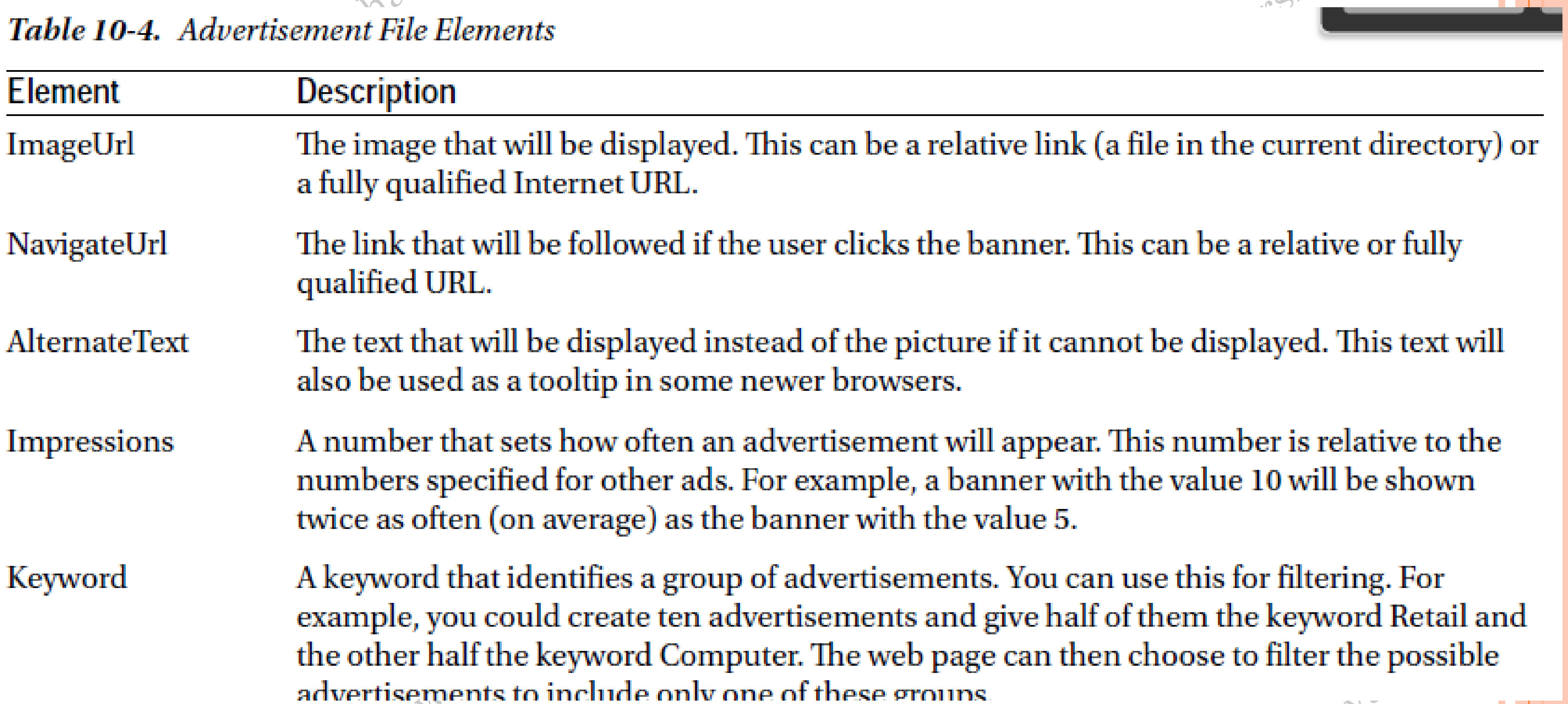


Table 10-4. Advertisement File Elements

Element	Description
ImageUrl	The image that will be displayed. This can be a relative link (a file in the current directory) or a fully qualified Internet URL.
NavigateUrl	The link that will be followed if the user clicks the banner. This can be a relative or fully qualified URL.
AlternateText	The text that will be displayed instead of the picture if it cannot be displayed. This text will also be used as a tooltip in some newer browsers.
Impressions	A number that sets how often an advertisement will appear. This number is relative to the numbers specified for other ads. For example, a banner with the value 10 will be shown twice as often (on average) as the banner with the value 5.
Keyword	A keyword that identifies a group of advertisements. You can use this for filtering. For example, you could create ten advertisements and give half of them the keyword Retail and the other half the keyword Computer. The web page can then choose to filter the possible advertisements to include only one of these groups

Table 10-5. Special Frame Targets

Target	Description
_blank	The link opens a new unframed window.
_parent	The link opens in the parent of the current frame.
_self	The link opens in the current frame.
_top	The link opens in the topmost frame of the current window (so the link appears in the full window).

- <asp:AdRotator ID="Ads" runat="server" AdvertisementFile="MainAds.xml"
- Target=”_blank” KeywordFilter=”Computer” />

The Validation Controls

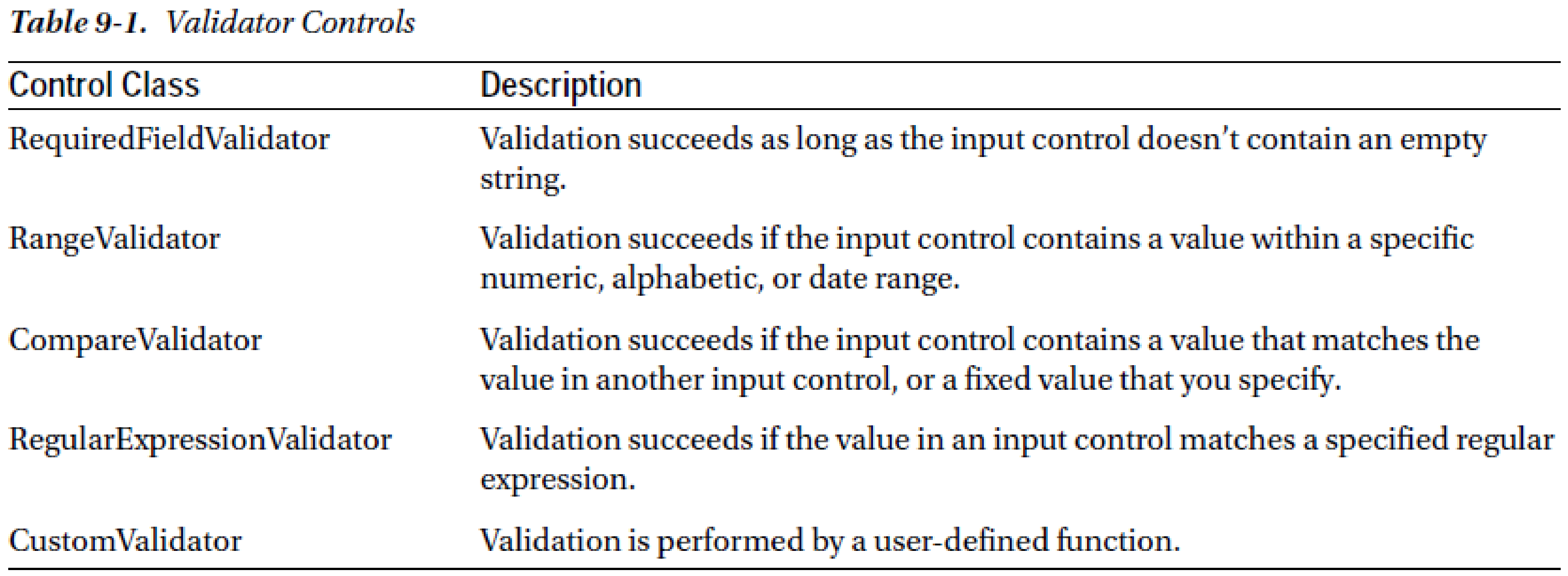


Table 9-1. Validator Controls

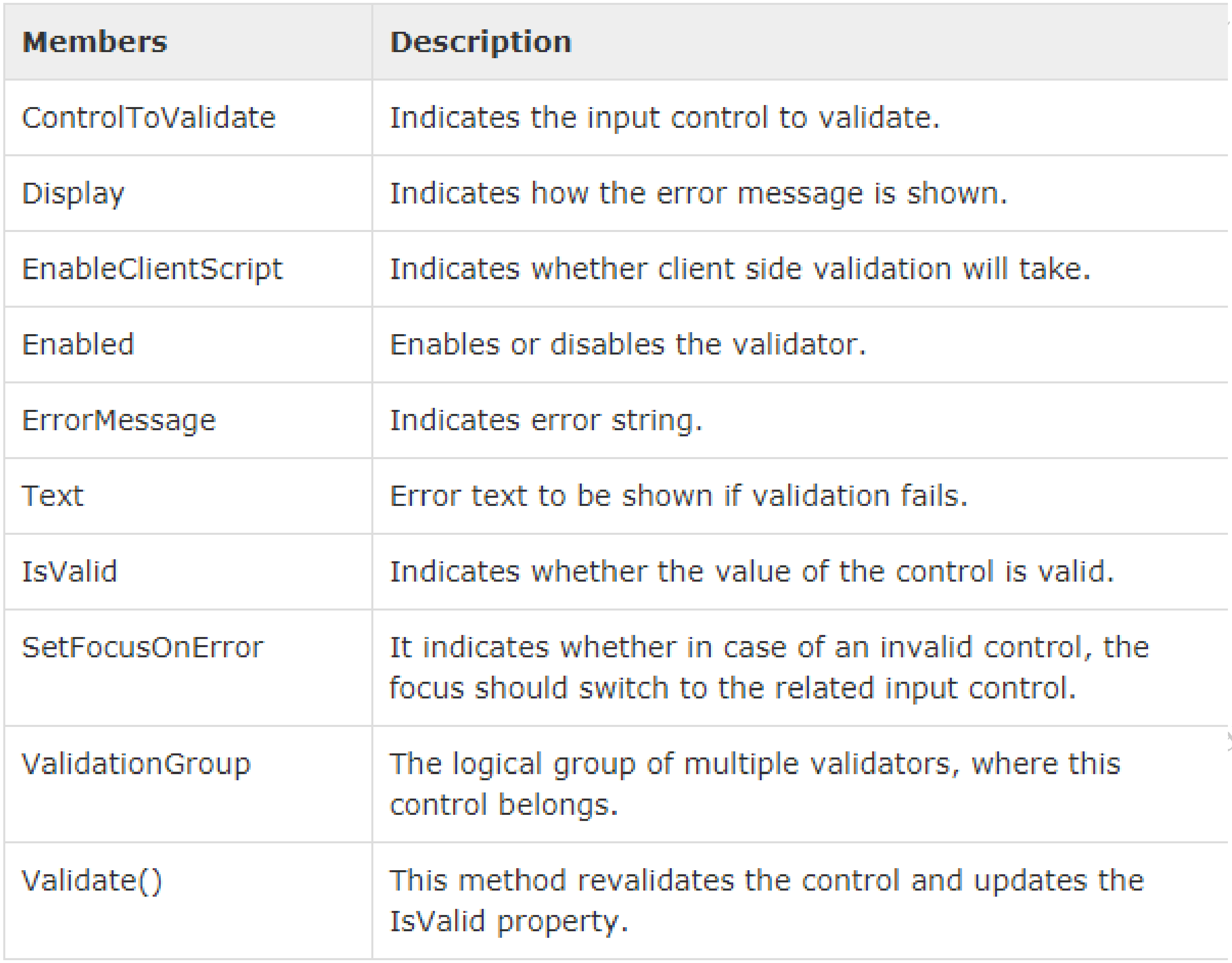
Control Class	Description
RequiredFieldValidator	Validation succeeds as long as the input control doesn't contain an empty string.
RangeValidator	Validation succeeds if the input control contains a value within a specific numeric, alphabetic, or date range.
CompareValidator	Validation succeeds if the input control contains a value that matches the value in another input control, or a fixed value that you specify.
RegularExpressionValidator	Validation succeeds if the value in an input control matches a specified regular expression.
CustomValidator	Validation is performed by a user-defined function.

- Each validation control can be bound to a single input control.
- In addition, you can apply more than one validation control to the same input control to provide multiple types of validation.

VALIDATION CONTROLS

- RequiredFieldValidator
- RangeValidator
- CompareValidator
- RegularExpressionValidator
- CustomValidator
- ValidationSummary

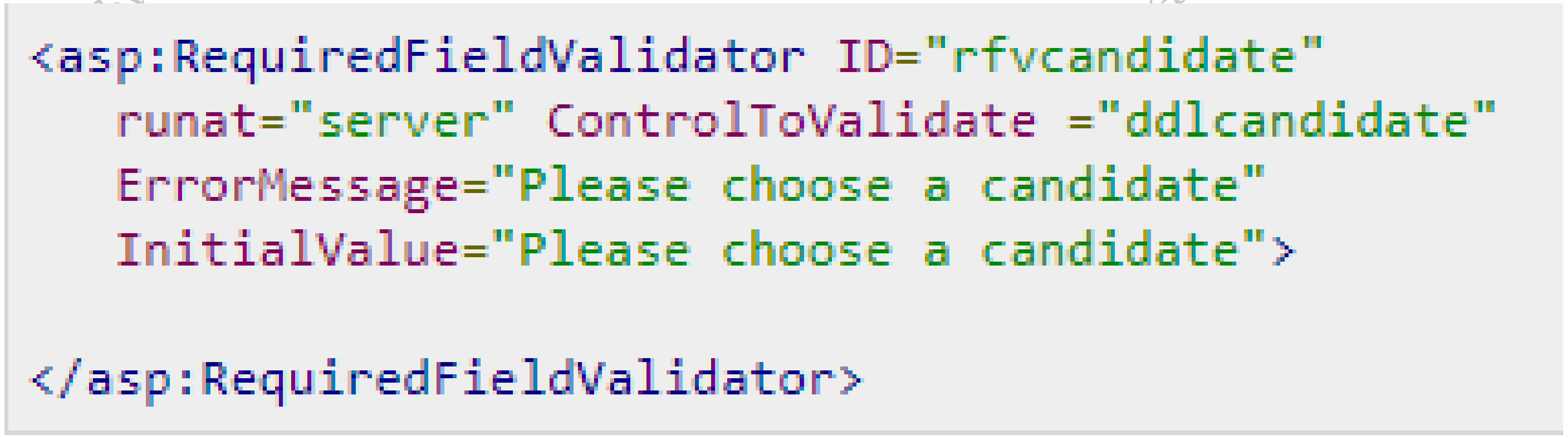
BaseValidator Class



Members	Description
ControlToValidate	Indicates the input control to validate.
Display	Indicates how the error message is shown.
EnableClientScript	Indicates whether client side validation will take.
Enabled	Enables or disables the validator.
ErrorMessage	Indicates error string.
Text	Error text to be shown if validation fails.
IsValid	Indicates whether the value of the control is valid.
SetFocusOnError	It indicates whether in case of an invalid control, the focus should switch to the related input control.
ValidationGroup	The logical group of multiple validators, where this control belongs.
Validate()	This method revalidates the control and updates the IsValid property.

RequiredFieldValidator Control

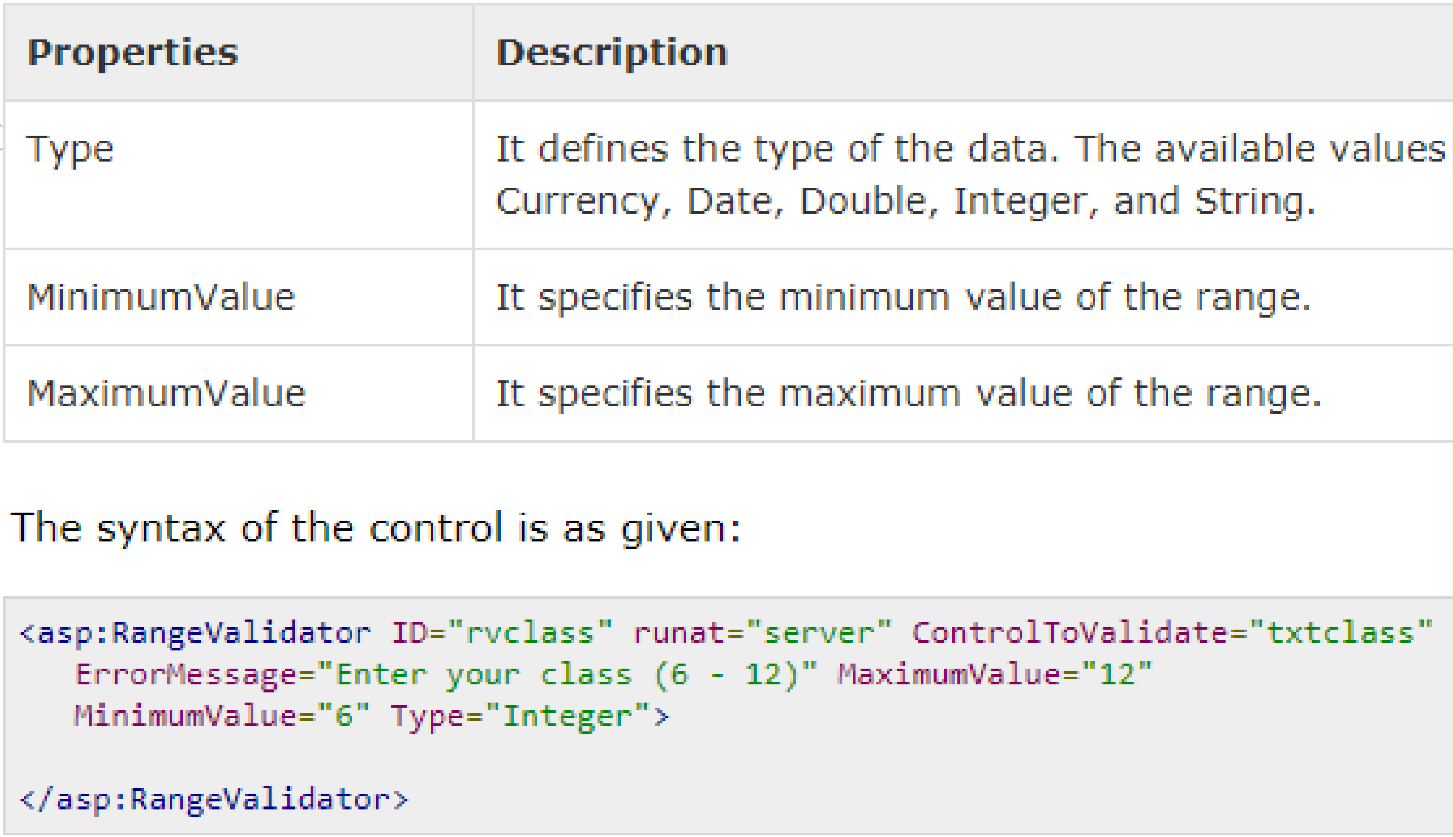
- The RequiredFieldValidator control ensures that the required field is not empty.



```
<asp:RequiredFieldValidator ID="rfvcandidate"
   runat="server" ControlToValidate ="ddlcandidate"
   ErrorMessage="Please choose a candidate"
   InitialValue="Please choose a candidate">

</asp:RequiredFieldValidator>
```

RangeValidator Control

- The RangeValidator control verifies that the input value falls within a predetermined range.



Properties	Description
Type	It defines the type of the data. The available values Currency, Date, Double, Integer, and String.
MinimumValue	It specifies the minimum value of the range.
MaximumValue	It specifies the maximum value of the range.

The syntax of the control is as given:

```
<asp:RangeValidator ID="rvclass" runat="server" ControlToValidate="txtclass"
   ErrorMessage="Enter your class (6 - 12)" MaximumValue="12"
   MinimumValue="6" Type="Integer">

</asp:RangeValidator>
```

CompareValidator Control

- The CompareValidator control compares a value in one control with a fixed value or a value in another control.

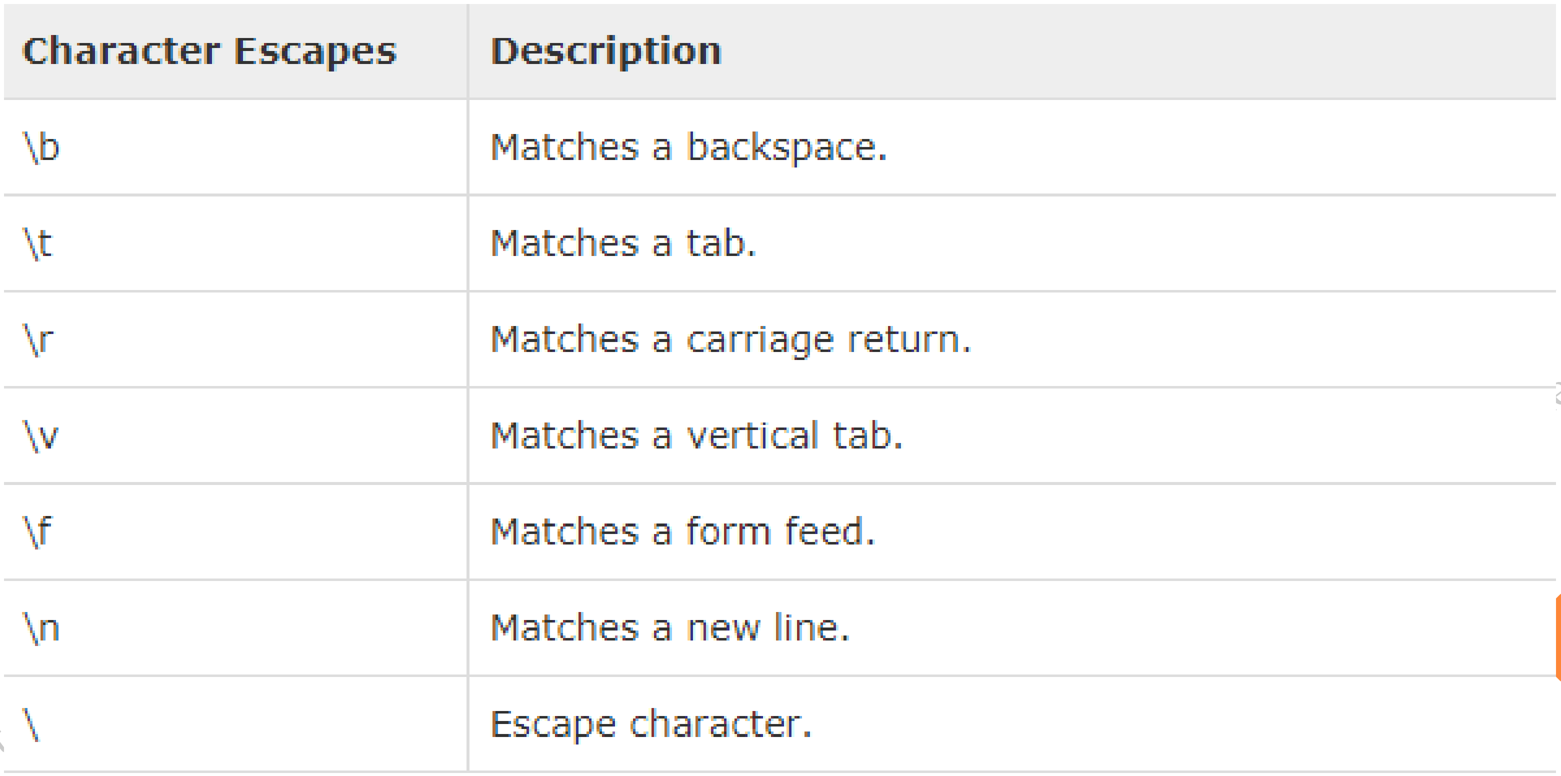
Properties	Description
Type	It specifies the data type.
ControlToCompare	It specifies the value of the input control to compare with.
ValueToCompare	It specifies the constant value to compare with.
Operator	It specifies the comparison operator, the available values are: Equal, NotEqual, GreaterThan, GreaterThanEqual, LessThan, LessThanEqual, and DataTypeCheck.

The basic syntax of the control is as follows:

```
<asp:CompareValidator ID="CompareValidator1" runat="server"
   ErrorMessage="CompareValidator">

</asp:CompareValidator>
```

RegularExpressionValidator

- The RegularExpressionValidator allows validating the input text by matching against a pattern of a regular expression.
- The regular expression is set in the ValidationExpression property.



Character Escapes	Description
\b	Matches a backspace.
\t	Matches a tab.
\r	Matches a carriage return.
\v	Matches a vertical tab.
\f	Matches a form feed.
\n	Matches a new line.
\	Escape character.

Metacharacters	Description
.	Matches any character except \n.
[abcd]	Matches any character in the set.
[^abcd]	Excludes any character in the set.
[2-7a-mA-M]	Matches any character specified in the range.
\w	Matches any alphanumeric character and underscore.
\W	Matches any non-word character.
\s	Matches whitespace characters like, space, tab, new line etc.
\S	Matches any non-whitespace character.
\d	Matches any decimal character.
\D	Matches any non-decimal character.

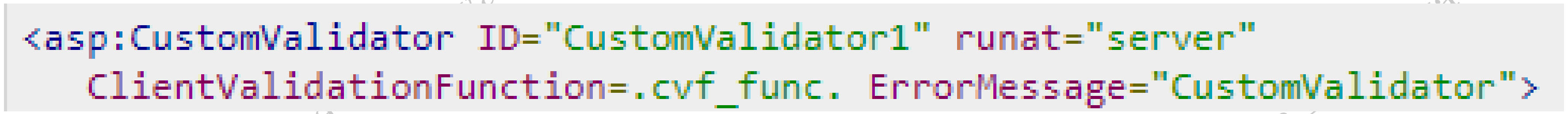
Quantifier	Description
*	Zero or more matches.
+	One or more matches.
?	Zero or one matches.
{N}	N matches.
{N,}	N or more matches.
{N,M}	Between N and M matches.

The syntax of the control is as given:

```
<asp:RegularExpressionValidator ID="string" runat="server" ErrorMessage="string"
   ValidationExpression="string" ValidationGroup="string">

</asp:RegularExpressionValidator>
```

CustomValidator

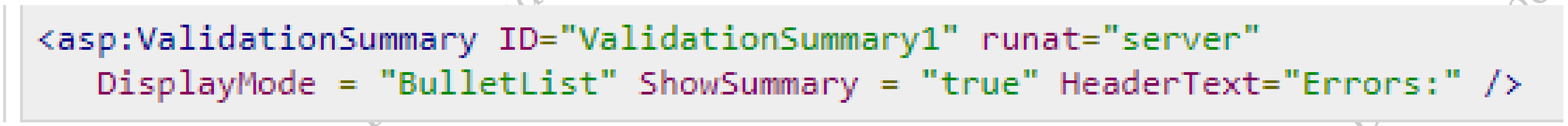
- The CustomValidator control allows writing application specific custom validation routines for both the client side and the server side validation.
- The client side validation routine should be written in a scripting language, such as JavaScript or VBScript, which the browser can understand.
- The server side validation routine must be called from the control's ServerValidate event handler.
- The server side validation routine should be written in any .Net language, like C# or VB.Net.



```
<asp:CustomValidator ID="CustomValidator1" runat="server"
   ClientValidationFunction=.cvf_func. ErrorMessage="CustomValidator">
```

VALIDATIONSUMMARY

- The ValidationSummary control does not perform any validation but shows a summary of all errors in the page.
- The summary displays the values of the ErrorMessage property of all validation controls that failed validation.
- The following two mutually inclusive properties list out the error message:
- **ShowSummary** : shows the error messages in specified format.
- **ShowMessageBox** : shows the error messages in a separate window.



```
<asp:ValidationSummary ID="ValidationSummary1" runat="server"
   DisplayMode = "BulletList" ShowSummary = "true" HeaderText="Errors:" />
```

VALIDATION GROUPS

- Complex pages have different groups of information provided in different panels.
- In such situation, a need might arise for performing validation separately for separate group.
- This kind of situation is handled using validation groups.
- To create a validation group, you should put the input controls and the validation controls into the same logical group by setting their *ValidationGroup*property.