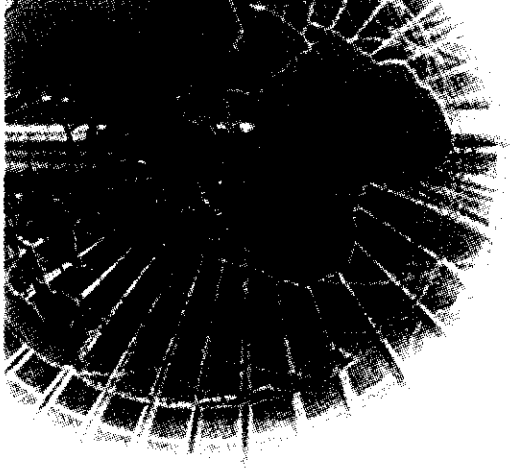


17

Web Forms: Standard Controls

In the previous chapters, you have already learned how to create Web applications. Now, let's discuss about Web Form controls. The Web Form controls are closely designed to resemble standard Visual Basic Windows forms controls. These controls are used for designing the interface for any Web application, for example, when you visit the website of Google, you type your search query in a textbox, which is a control. ASP.NET provides a standard set of controls that can be used for the development of Web applications. You can access all these controls from the Toolbox present in the Visual Studio Integrated Development Environment (IDE). These controls can easily be used by just dragging and dropping them at any desired location on the Web Form. Based on the tasks performed by them, these controls on the Toolbox are grouped under various categories known as tabs. For example, controls for validating the data are put under the Validation tab and controls used for logging on to websites are put under the Login tab. Similarly, controls for common use are put under the Standard tab and are known as Standard controls. All these controls come under the Control class. All the Standard/ Web server controls are based on the `WebControl` class, which, in turn, is based on the Control class. In other words, the `WebControl` class has originated from the Control class.

In this chapter, we describe the inheritance hierarchy, public properties, public methods, and public events of the `Control` and the `WebControl` classes. We also get to know about the various controls such as `Label`, `Button`, `TextBox`, `Literal`, `PlaceHolder`, `Hidden Field`, `FileUpload`, Image, ImageMap, ListBox, DropDownList, BulletedList, HyperLink,LinkButton, CheckBox, RadioButton, Table,and Wizard that originate from the `WebControl` class, along with their implementations. Chapter also describes the user controls and custom controls along with their implementations. Let's start with the Control class.

The **Control** Class

The `System.Web.UI.Control` class is the base class for all Web server controls. This class is derived directly from the `Object` class, which resides in the `System` namespace. Here is the hierarchy of the `Control` class:

```
System.Object
  System.Web.UI.Control
```

Noteworthy public properties of the `Control` class are listed in Table 17.1:

Table 17.1: Noteworthy Public Properties of Control Class

`AppRelativeTemplateSource Directory`	Obtains the application-relative virtual directory of the `Page` or `UserControl` that contains this control
`BindingContainer`	Obtains the control that contains this control's data binding
`ClientID`	Obtains the server control identifier generated by ASP.NET
`Controls`	Obtains a `ControlCollection` object that represents the child controls for a specified server control in the `UI` hierarchy
`EnableTheming`	Obtains a value indicating whether themes apply to this control
`EnableViewState`	Obtains a value indicating whether the server control persists its view-state, and the view-state of any child controls it contains
`ID`	Obtains the programmatic identifier assigned to the server control
`NamingContainer`	Obtains a reference to the server control's naming container, which creates a unique namespace for differentiating between server controls with the same `System.Web.UI.Control.ID` property value
`Page`	Obtains a reference to the `Page` instance that contains the server control
`Parent`	Obtains a reference to the server control's parent control in the page control hierarchy
`Site`	Obtains information about the container that hosts the current control when rendered on a design surface

Table 17.1: Noteworthy Public Properties of Control Class	
SkinID	Obtains the skin to apply to the control
TemplateControl	Obtains a reference to the template that contains this control
TemplateSourceDirectory	Obtains the virtual directory of the Page or UserControl that contains the current server control
UniqueID	Obtains the unique, hierarchically-qualified identifier for the server control
Visible	Obtains a value that indicates whether a server control is rendered as UI on the page

Noteworthy public methods of the Control class are listed in Table 17.2:

Table 17.2: Noteworthy Public Methods of Control Class	
ApplyStyleSheetSkin	Helps in applying the style properties defined in the page style sheet to the control
DataBind	Helps in binding a data source to the invoked server control and all its child controls
Dispose	Helps in enabling a server control to perform final clean up before it is released from memory
FindControl	Helps in searching the current naming container for the specified server control
Focus	Helps in setting input focus to a control
HasControls	Helps in determining if the server control contains any child controls
RenderControl	Helps in output server control content and stores tracing information about the control if tracing is enabled
ResolveClientUrl	Helps in getting a URL that can be used by the browser
ResolveUrl	Helps in converting a URL into the desired format
SetRenderMethodDelegate	Helps in assigning an event handler delegate to render the server control and its content into its parent control

Noteworthy public events of the Control class are listed in Table 17.3:

Table 17.3: Noteworthy Public Events of Control Class	
DataBinding	Occurs when the server control binds to a data source
Disposed	Occurs when a server control is released from memory, which is the last stage of the server control life cycle when an ASP.NET page is requested
Init	Occurs when the server control is initialized, which is the first step in the life cycle
Load	Occurs when the server control is loaded into the Page object
PreRender	Occurs when the server control is about to render to its containing Page object
Unload	Occurs when the server control is unloaded from memory

After discussing the major properties, methods, and events of the Control class, let's move on to discuss the WebControl class; since it is the class on which all the controls directly depend.

The **WebControl** Class

All Web server controls exist within the System.Web.UI.WebControls.WebControl namespace and can be used on any Web form. Here is the hierarchy of `WebControl` class:

```
System.Object
    System.web.UI.Control
        System.web.UI.WebControls.WebControl
```

Noteworthy public properties of the `WebControl` class are listed in Table 17.4:

Table 17.4: Noteworthy Public Properties of WebControl Class

`AccessKey`	Obtains the access key that allows you to quickly navigate to the Web server control
`Attributes`	Obtains the collection of arbitrary attributes (for rendering only) that do not correspond to properties on the control
`BackColor`	Obtains the background color of the Web server control
`BorderColor`	Obtains the border color of the Web control
`BorderStyle`	Obtains the border style of the Web server control
`BorderWidth`	Obtains the border width of the Web server control
`ControlStyle`	Obtains the style of the Web server control. This property is used primarily by control developers
`ControlStyleCreated`	Gets a value indicating whether a `Style` object has been created for the `ControlStyle` property. This property is primarily used by control developers
`CssClass`	Obtains the Cascading Style Sheet (CSS) class rendered by the Web server control on the client
`Enabled`	Obtains a value indicating whether the Web server control is enabled
`EnableTheming`	Obtains a value indicating whether themes apply to this control
`Font`	Obtains the font properties associated with the Web server control
`ForeColor`	Obtains the foreground color (typically the color of the text) of the Web server control
`HasAttributes`	Sets a value indicating whether the control has attributes set
`Height`	Obtains the height of the Web server control
`SkinID`	Obtains the skin to apply to the control
`Style`	Obtains a collection of text attributes that will be rendered as a style attribute on the outer tag of the Web server control
`TabIndex`	Obtains the tab index of the Web server control
`ToolTip`	Obtains the text displayed when the mouse pointer hovers over the Web server control
`Width`	Obtains the width of the Web server control

Noteworthy public methods of the `WebControl` class are listed in Table 17.5:

Table 17.5: Noteworthy Public Methods of WebControl Class

`ApplyStyle`	Copies any non-blank elements of the specified style to the Web control. It also overwrites any existing style elements of the control

Table 17.5: Noteworthy Public Methods of WebControl Class	
`CopyBaseAttributes`	Copies the properties that are not encapsulated by the `Style` object from the specified Web server control to the Web server control that this method is called from. This method is used primarily by control developers
`MergeStyle`	Copies any non-blank elements of the specified style to the Web control, but will not overwrite any existing style elements of the control. This method is used primarily by control developers
`RenderBeginTag`	Writes the opening tag of the specified markup element to the output stream
`RenderEndTag`	Writes the end tag of a markup element to the output stream

Some of the basic standard controls discussed in this chapter are as follows:

- The Label control
- The Button control
- The TextBox control
- The Literal control
- The PlaceHolder control
- The Hidden Field control
- The FileUpload control
- Image control
- ImageMap control
- ListBox control
- DropDownList Control
- BulletedList Control
- HyperLink Control
- LinkButton Control
- CheckBox Control
- RadioButton Control
- Table Control
- Wizard Control
- Before moving to standard controls, let's go through the cascading style sheet (CSS).

Using CSS in Web Applications

CSS stands for cascading style sheet, which helps in giving a better look and feel to applications.

NOTE

Now onwards, you will find the same CSS in all the applications of this book.

Now, let's learn the procedure to create CSS in any Web application. To create a CSS, right-click the Web application's name in the Solution Explorer and select the Add New Item option from the context menu, as shown in Figure 17.1:

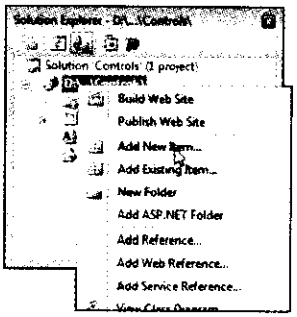


Figure 17.1: Add New Item Option

The Add New Item dialog box appears. Select the Style Sheet template from the list of given templates and click the Add button, as shown in Figure 17.2:

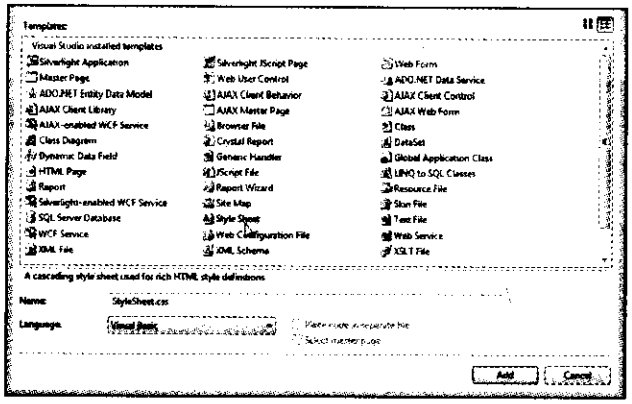


Figure 17.2: Add New Item Dialog Box Showing the Style Sheet Template

NOTE

You can also create the CSS file by performing the following steps:

1. *Create a file with the .css extension using text editor, such as Notepad.*
2. *Add the code (provided in Listing 17.1 later) to the .css file and save it as StyleSheet.css. You can find the code of StyleSheet.css application in the Code\ASP.NET\Chapter 17\StyleSheet.css folder on the CD.*
3. *Now, create a new Web application or open an existing one.*
4. *Right-click the Solution Explorer and select the Add Existing Item option, as shown in Figure 17.3:*

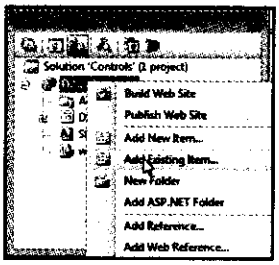


Figure 17.3: Add Existing Item Option in Context Menu

The Add Existing Item dialog box opens.

5. *Browse the CSS file you have created and click the Add button, as shown in Figure 17.4:*

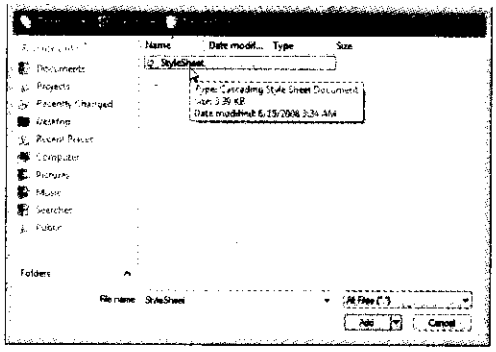


Figure 17.4: Add Existing Item Dialog Box

It will add the CSS file to the Solution Explorer.
Now, follow the procedure given after this note for linking the CSS file in the Default.aspx page.

Now, you can see this file (`StyleSheet.css`) in the Solution Explorer. Listing 17.1 shows the code for the CSS in StyleSheet.css file:

Listing 17.1: Showing the Code for the `StyleSheet.css` File

```
html, body {
   background-color: #FFFFFF;
   color: #000000;
   font: normal 90%/1.8em 'Lucida Grande', Verdana, Geneva, Lucida, Helvetica, Arial,
    sans-serif;
   margin: 0;
   height: 100%;
}
h1 {
   font-size: 1.8em;
   font-weight: bold;
   margin-top: 0em;
   margin-bottom: 0em;
   color: #a83930;
}
h2 {
   font-size: 1.6em;
   margin: 1.0em 0em 1.0em 0em;
   font-weight: bold;
   color: #a83930;
}
h3 {
   font-size: 1.2em;
   margin: 1.0em 0em 1.0em 0em;
   font-weight: bold;
   color: #a83930;
}
p {
   font-size: 1.1em;
   line-height: 1.8em;
   margin: 1.1em 0em 1.1em 0em;
   text-align: left;
}
```

```
ul
{
  font-size: 1.1em;
}
a:link, a:visited {
  color: #cc3300;
  text-decoration: underline;
}
a:hover {
  text-decoration: none;
}
a:active {
  color: #ff9900;
  text-decoration: underline;
}
title {
  color: #a83930;
}
#header {
  height: 67px;
  padding: 0 0px 0 0px;
  color: #a83930;
  background-color: #000000;
  border-bottom: 8px solid #d30a1a;
}
#header h1 {
  padding: 10px 0 0 0;
  margin: 10px 0 0 0px;
  color: #FFFFFF;
  margin-left: 10px;
  margin-bottom: 10px;
}
#logo {
  background-color: #000000;
  background: url(images/logo.gif);
  background-position: top left;
  background-repeat: no-repeat;
  height: 100px;
  width: 100px;
  border: 0;
  float: left;
}
#content {
  margin-left: 210px;
  padding: 0 20px 1.8em 10px;
  background-color: #fff;
}
.code {
  color: #a83930;
  background-color: #fdea12;
  font: 0.9em/1.4em verdana, arial, helvetica, sans-serif;
  float: right;
  border: solid 1px #a83930;
  padding: 10px;
}

#sidebar {
  float:left;
  padding: 0 10px 10px 10px;
```

```
    background-color: #656565;
    /*background-image: url(images/bg_2.gif);*/
    background-repeat: repeat-y;
    background-position: top left;
    border-top: 2px solid #656565;
    border-bottom: 2px solid #656565;
    width: 168px;  /* ds */
    height:500px;
  }
  html>body #sidebar {
    width: 180px;
  }
  #sidebar h3 {
    font-weight: bold;
    padding-bottom: 0.5em;
    border-bottom: 1px dashed #fdea12;
    color: #fdea12;
  }

  #nav a:link, #nav a:visited {
    /*display: block;*/
    width: 99.9%; /* for IE5 */
    color: #FFFFFF;
    text-decoration: none;
    padding: 0.25em 0.5em 0.25em 0.5em;
    font-weight: bold;
  }
  #nav a:hover {
    text-decoration: none;
    color: #FFFFFF;
    /*background-color: #fdea12;  */
  }

  .title {
    color: #a83930;
    font: bold 1.0em/1.0em verdana, arial, helvetica, sans-serif;
    text-align: center;
    padding-bottom: 5px;
    border-bottom: #a83930 thin solid;
    margin-bottom: 5px;
  }
  #footer {
    /*margin-left: 210px; */
    padding: 0 20px 1.5em 10px;
    border-top: 1px solid #000000;
    clear: both;
  }
  #footer p {
    font: normal 0.8em/0.9em verdana, arial, helvetica, sans-serif;
    color: #666;
  }
  #footer p.left {
    float: left;
    clear: left;
  }
  #footer p.right {
    float: right;
    clear: right;
  }
```

```
.spacer {
  clear: both;
}
dd, dt {
  font-size: 0.95em;
}
#mainTitle {
  font-size: 2.0em;
  font-weight: bold;
  visibility: hidden;
}
.pageTitle {
  font-size: 1.5em;
  font-weight: bold;
}
.itemTitle {
  border-bottom: 1px solid #a83930;
  font-size: 1.5em;
}
.itemContent {
  padding-bottom: 1.8em;
}
.box {
  border: solid 1px #a83930;
}
.boxFloat {
  /* border: solid 1px #a83930; */
  float: left;
}
.boxFloatRight {
  border: solid 1px #a83930;
  float: right;
}
.floatRight {
  /* border: solid 1px #a83930; */
  float: right;
}
```

Now, let's link this CSS file to the Web page. To do so, add the following code in the <head> section of the Default.aspx file:

```
<link href="StyleSheet.css" rel="stylesheet" type="text/css" />
```

Listing 17.2: Showing the Code for the Default.aspx Page

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb"
  Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>Linking CSS file</title>
  <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <form id="form" runat="server">
      <div id="header">

      </div>
      <div id="sidebar">
          <div id="nav">
               
```

```
                </div>
            </div>
            <div id="content">
                <div class ="itemContent">
                    <div id="footer">
                        <p class="left">
                        All content copyright &copy; Kogent Solutions Inc.</p>
                    </div>
                </div>
            </div>

    </form>

</body>
</html>
```

Now, run the application by pressing the F5 key. The output will be as shown in Figure 17.5:

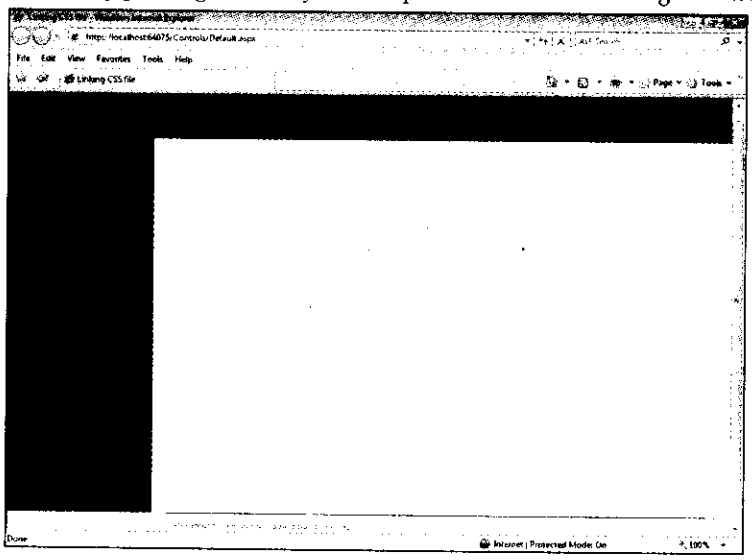


Figure 17.5: CSS File Attached with the Web Page

The Label Control

The Label control is used to display the text that the user cannot edit. The Label control exists within the `System.Web.UI.WebControls` namespace. Here is the class hierarchy of the Label class:

```
System.Object
  System.Web.UI.Control
    System.Web.UI.WebControls.WebControl
      System.Web.UI.WebControls.Label
```

Noteworthy public properties of the `Label` class are listed in Table 17.6:

Table 17.6: Noteworthy Public Properties of the Label Class

Property	Description
`AssociatedControlID`	Obtains or sets the identifier for a server control the `Label` control is associated with
`Text`	Obtains or sets the text content of the `Label` control

Working with Label Controls

When you add a Label control on a Web page, it adds the following code to the source code of the Web page:

```
<asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
```

Now, let's create applications named LabelVBExample. You can find the code of LabelVBExample application in the Code\ASP.NET\Chapter 17\LabelVBExample folder on the CD. You can add a `Label` control to a Web page either by dragging and dropping it from the Standard tab of the Toolbox or by double-clicking this control in the Toolbox. In this example, we are setting the `Text` and `BackColor` properties of a Label control at the `Page_Load` event. Listing 17.3 shows the `Default.aspx` page for the Label controls:

Listing 17.3: Showing the Code for the `Default.aspx` Page

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb"
  Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
  <title>Label Control Example</title>
  <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <form id="Form" runat="server">
        <div id="header">
        </div>
        <div id="sidebar">
              <div id="nav">
                     
              </div>
        </div>
        <div id="content">
              <div class ="itemContent">
              <asp:Label ID="Label1" runat="server"
              Text="Example showing some of the properties of the Label Control"
              Font-Bold="True" Font-Underline="True"></asp:Label>
              <br />
              <br />
              <asp:Label ID="Label3" runat="server"></asp:Label>
              <br />
              <br />
              <asp:Label ID="Label4" runat="server"></asp:Label>
              <br />
              <div id="footer">
                    <p class="left">
                          All content copyright &copy; Kogent Solutions Inc.
                    </p>
              </div>
              </div>
        </div>
  </form>
</body>
</html>
```

Now, add the code in the code-behind file of the `Default.aspx page for the Label controls`, as shown in Listing 17.4:

Listing 17.4: Showing the code for the Code-Behind File of the `Default.aspx Page`

```
Partial Class _Default
  Inherits System.Web.UI.Page
  Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles
  Me.Load
```

```
        Label3.Text = "Welcome to the World of ASP.NET 3.5"
        Label4.Text = "I have Olive Color"
        Label4.BackColor = System.Drawing.Color.Olive
    End Sub
End Class
```

Now, run the application by pressing the F5 key. The output is as shown in Figure 17.6:

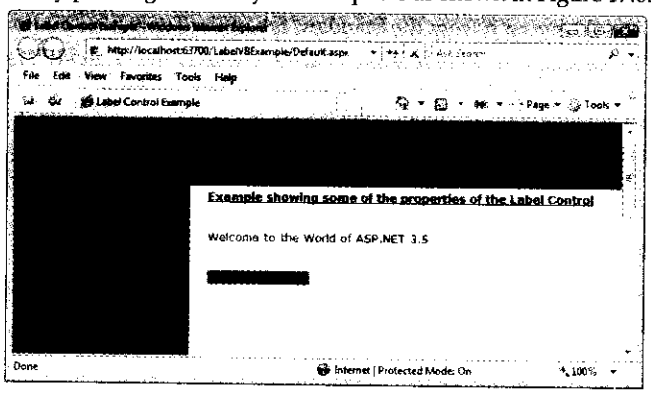


Figure 17.6: Label Control Showing Its Properties at Runtime

The Button Control

The Button control is used to create a button that sends a request to a Web page. The Button control exists within the `System.Web.UI.WebControls` namespace. The Button controls post data to the server when they are clicked.

Here is the class hierarchy of the Button class:

```
System.Object
    System.Web.UI.Control
        System.Web.UI.WebControls.WebControl
            System.Web.UI.WebControls.Button
```

Noteworthy public properties of the `Button` class are listed in Table 17.7:

Table 17.7: Noteworthy Public Properties of the Button Class

Property	Description
`CausesValidation`	Obtains or sets a value indicating whether validation is performed when the `Button` control is clicked
`CommandArgument`	Obtains or sets an optional parameter passed to the Command event along with the associated CommandName
`CommandName`	Obtains or sets the command name associated with the Button control that is passed to the Command event
`OnClientClick`	Obtains or sets the client-side script that executes when a Button control's Click event is raised
`PostBackUrl`	Obtains or sets the URL of the page to post when the Button control is clicked
`Text`	Obtains or sets the text caption displayed in the Button control
`UseSubmitBehavior`	Obtains or sets a value indicating whether the `Button` control uses the client browser's submit mechanism or the ASP.NET postback mechanism
`ValidationGroup`	Obtains or sets the group of controls for which the `Button` control causes validation when it posts back to the server

Noteworthy public events of the `Button` class are listed in Table 17.8:

Table 17.8: Noteworthy Public Events of the Button Class

Event	Description
`Click`	Occurs when the `Button` control is dicked
`Command`	Occurs when the `Button` control is dicked

The Button controls are also used to create push buttons on the Web page. Push buttons can be either Submit buttons or Command buttons. By default, a Button control is a Submit button. A Submit button does not have a Command name element, which you specify by the CommandName property, associated with the Button control. You can provide an event handler with the Click event to programmatically control the actions performed when the Submit button is clicked.

A Command button has a command name associated with the button, which you specify by the CommandName property, such as Sort. This allows you to create multiple Button controls on a Web page and programmatically find out that which of the Button control is clicked. You can provide an event handler for the Command event to programmatically control the actions performed when the Command button is clicked.

Working with Button Controls

When you add a Button control on a Web page, it adds the following code to the source code of the Web page:

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

Now, let's create an application named ButtonVBExample. You can find the code of ButtonVBExample application in the Code\ASP.NET\Chapter 17\ButtonVBExample folder on the CD. You can add a Button control to a Web page either by dragging and dropping it from the Standard tab of the Toolbox or by double-clicking this control in the Toolbox. In this example, we are setting the `Text` and `BackColor` properties of a Label control at the `Button_Click` event. Listing 17.5 shows the `Default.aspx` page for the Button controls:

Listing 17.5: Showing the Code for the `Default.aspx` Page

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb"
  Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
  <title>Button Control Example</title>
  <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <form id="Form" runat="server">
        <div id="header">

        </div>
        <div id="sidebar">
              <div id="nav">
                     
              </div>
        </div>
        <div id="content">
              <div class ="itemContent">
              <asp:Label ID="Label1" runat="server" Font-Bold="True" Font-
              Underline="True" Text="Button control example showing text on button
              click event"></asp:Label>
              <br />
              <br />
              <br />
              <asp:Button ID="Button1" runat="server" Font-Bold="True"
```

```
                    Text="Click Me, I will Welcome You !!" />
                     <br />
                    <br />
                    <asp:Label ID="Label2" runat="server"></asp:Label>
                    <div id="footer">
                            <p class="left">
                            All content copyright &copy; Kogent Solutions Inc.</p>
                    </div>
                    </div>
            </div>
    </form>
</body>
</html>
```

Now, add the code in the code-behind file of the `Default.aspx` page for the Button controls, as shown in Listing 17.6:

Listing 17.6: Showing the code for the Code-Behind File of the `Default.aspx` Page

```
Partial Class _Default
    Inherits System.Web.UI.Page
    Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
      Handles Button1.Click
        Label2.Text = " Welcome to the World of ASP.NET 3.5"
        Label2.ForeColor = System.Drawing.Color.RosyBrown
    End Sub
End Class
```

Now, run the application by pressing the F5 key and click the button. The output is as shown in Figure 17.7:

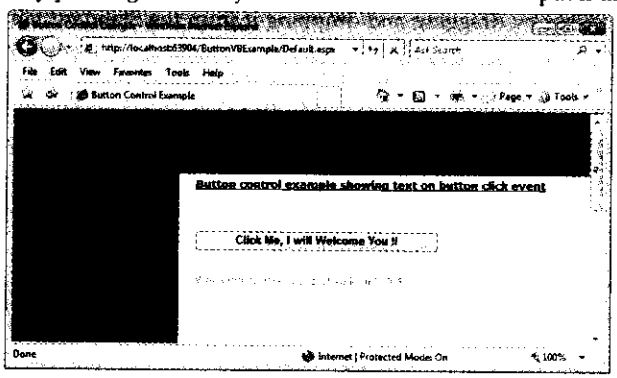


Figure 17.7: Button Control Showing Output Text

The TextBox Control

The TextBox control is an input control, which allows you to enter text. The TextBox control exists within the `System.Web.UI.WebControls` namespace. Here is the class hierarchy of this class:

```
System.Object
  System.Web.UI.Control
      System.Web.UI.WebControls.WebControl
          System.Web.UI.WebControls.TextBox
```

You can set the style of the TextBox by using the `TextMode` property. By default, the `TextMode` property is set to `SingleLine` to create a single-line HTML text field but it can also be set to `MultiLine` for a multiline text box.

You can convert a mode of TextBox control to a Password control, where the text, which the user types is masked with special symbols such as asterisks (*).

The display width of a text box is set with its `Columns` property and if it is a multiline text box, the display height is set with the `Rows` property. Noteworthy public properties of the `TextBox` class are listed in Table 17.9:

Table 17.9: Noteworthy Public Properties of the TextBox Class

Property	Description
`AutoCompleteType`	Obtains or sets a value that indicates the AutoComplete behavior of the TextBox control
`AutoPostBack`	Obtains or sets a value that indicates whether an automatic postback to the server occurs when the TextBox control loses focus
`CausesValidation`	Obtains or sets a value indicating whether validation is performed when the TextBox control is set to validate when a postback occurs
`Columns`	Obtains or sets the display width of the text box in characters
`MaxLength`	Obtains or sets the maximum number of characters allowed in the text box
`ReadOnly`	Obtains or sets a value indicating whether the contents of the TextBox control can be changed
`Rows`	Obtains or sets the number of rows displayed in a multiline text box
`Text`	Obtains or sets the text content of the TextBox control
`TextMode`	Obtains or sets the behavior mode (single-line, multiline, or password) of the TextBox control
`ValidationGroup`	Obtains or sets the group of controls for which the TextBox control causes validation when it posts back to the server
`Wrap`	Obtains or sets a value indicating whether the text content wraps within a multiline text box

The public event of the `TextBox` class is listed in Table 17.10:

Table 17.10: The Public Event of the TextBox Class

Event	Description
`TextChanged`	Occurs when the user changes the text of a TextBox

Working with TextBox Controls

When you add a TextBox control on a Web page, it adds the following code to the source code of the Web page:

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
```

Now, let's create an application named TextBoxVBExample. You can find the code of TextBoxVBExample application in the Code\ASP.NET\Chapter 17\TextBoxVBExample folder on the CD. This example shows how to work with single-line TextBox, multiline TextBox, and Textbox showing masked text (password). When you click the Button control in this example, the message "You have clicked the button appears in a single-line `TextBox`" and a multiline `TextBox` shows the message "This is a multi line textbox. This is a multi line textbox. This is a multi line textbox. This is a multi line textbox". In addition, the masked text (password) in a `TextBox` control is moved and displayed in another `TextBox` control which is below that control. Listing 17.7 shows the `Default.aspx` page for the TextBox controls:

Listing 17.7: Showing the Code for the `Default.aspx` Page

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb"
  Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
  <title>TextBox Control Example</title>
  <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
```

```
</head>
<body>
    <form id="Form" runat="server">
        <div id="header">
        </div>
        <div id="sidebar">
            <div id="nav">
                 
            </div>
        </div>
        <div id="content">
            <div class ="itemContent">
                <asp:Label ID="Label2" runat="server" Font-Bold="True"
                Text="TextBox example showing types of TextBox controls"
                Font-Underline="True"></asp:Label>
                <br />
                <br />
                <br />
                <asp:Button ID="Button1" runat="server" Text="Click me"
                  Width="125px" />
                 
                <asp:TextBox ID="TextBox1" runat="server" Width="262px"
                  ></asp:TextBox><br />

                <asp:TextBox ID="TextBox4" runat="server" Rows="10"
                  TextMode="MultiLine"></asp:TextBox><br />
            </div>
            <asp:Label ID="Label1" runat="server" Text="Enter Password"></asp:Label>
            <br />
            <asp:TextBox ID="TextBox2" runat="server" TextMode="Password"
              Width="172px"
            ></asp:TextBox> 
             <br />
            <asp:TextBox ID="TextBox3" runat="server" Width="173px">
            </asp:TextBox><br />
        </div>

        <div id="footer">
            <p class="left">
            All content copyright &copy; Kogent Solutions Inc.</p>
        </div>
    </form>
</body>
</html>
```

NOTE

The highlighted code shows the TextMode property of TextBox controls. The SingleLine TextMode is a default property of the TextBox control and does not appear in the Default.aspx file.

Now, add the code in the code-behind file of the `Default.aspx` page for the TextBox controls, as shown in Listing 17.8:

Listing 17.8: Showing the code for the Code-Behind File of the `Default.aspx Page`

```
Partial Class _Default
    Inherits System.Web.UI.Page
    Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
      Handles Button1.Click
        TextBox1.Text = "You have clicked the button"
        TextBox4.Text = "This is a multi line textbox. This is a multi line textbox.
          This is a multi line textbox. This is a multi line textbox. "
        TextBox3.Text = TextBox2.Text
    End Sub
End Class
```

Now, run the application by pressing the F5 key. Enter the password in the Password textbox and click the Click me button. The output is as shown in Figure 17.8:

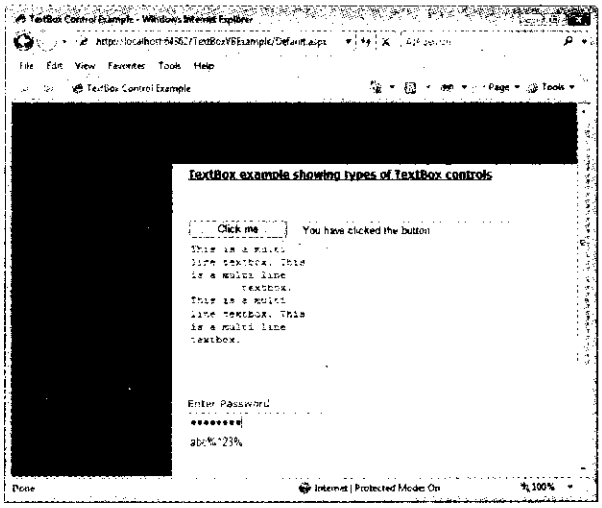


Figure 17.8: TextBox Control Showing its TextMode Properties

NOTE

When you click the Button control, the Enter Password textbox gets empty displaying the masked text in the textbox below it

The Literal Control

The `Literal` control is another Web server control used for displaying text on a Web page, similar to that done by a `Label` control. In ASP.NET, you can add HTML in a Web page with the help of `Literal` control, without switching to the `.aspx` page. The Literal control exists within the `System.Web.UI.WebControls` namespace. Here is the class hierarchy of this class:

```
System.Object
  System.Web.UI.Control
      System.Web.UI.WebControls.Literal
```

Noteworthy public properties of the `Literal` class are listed in Table 17.11:

Table 17.11: Noteworthy Public Properties of the Literal Class

Property	Description
`Mode`	Obtains or sets an enumeration value that specifies how the content in the Literal control is rendered
`Text`	Obtains or sets the caption displayed in the Literal control

You can directly set the text by using the `Text` property of the `Literal` control. This control does not have any visual appearance in a Web page.

Working with Literal Controls

When you add a Literal control on a Web page, the following code is added to the source code of the Web page:

```
<asp:Literal ID="Literal1" runat="server"></asp:Literal>
```

Now, let's create an application named LiteralVBExample You can find the code of LiteralVBExample application in the Code\ASP.NET\Chapter 17\LiteralVBExample folder on the CD. You can add a Literal control to a Web page either by dragging and dropping it from the Standard tab of the Toolbox or by double-

clicking this control in the **Toolbox**. When you click the Button control in this example, the code (code-behind file) inserts the HTML code needed to display the message. The Literal control does not have any visual appearance in a Web page. You just set the `Text` property and that text is inserted into the Web Form without any HTML coding prefixes. Here, we will insert the HTML code for an `<h1>` header and `<marquee>` tag into the Button `Click` event of a Web Form, using a `Literal` control. Listing 17.9 shows the `Default.aspx` page for the Literal controls:

Listing 17.9: Showing the Code for the `Default.aspx` Page

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb"
  Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
  <title>Literal Control Example</title>
  <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <form id="Form" runat="server">
        <div id="header">

        </div>
        <div id="sidebar">
              <div id="nav">
                     
              </div>
        </div>
        <div id="content">
              <div class ="itemContent">
                    <asp:Label ID="Label1" runat="server"  Font-Bold="True" Font-
                      Underline="True"
                    Text="Literal Control Example showing Text on Button Click
                      Event"></asp:Label>
                    <br />
                    <br />
                    <br />
                    <asp:Button ID="Button1" runat="server" Font-Bold="True"
                      Text="Click Me!!" />
                    <br />
                    <br />
                    <br />
                    <asp:Literal ID="Literal1" runat="server"></asp:Literal>
                    <div id="footer">
                          <p class="left">
                          All content copyright &copy; Kogent Solutions Inc.</p>
                    </div>
              </div>
        </div>
  </form>
</body>
</html>
```

Now, add the code in the code-behind file of the `Default.aspx` page for Literal controls, as shown in Listing 17.10:

Listing 17.10: Showing the Code for the Code-Behind File of the `Default.aspx`

```
Partial Class _Default
  Inherits System.Web.UI.Page
  Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    Handles Button1.Click
      Literal1.Text = "<h1><marquee>Welcome to ASP.NET 3.5</marquee></h1>"
  End Sub
End Class
```

Now, run the application by pressing the F5 key and then click the Click Me!! button. The output is as shown in Figure 17.9:

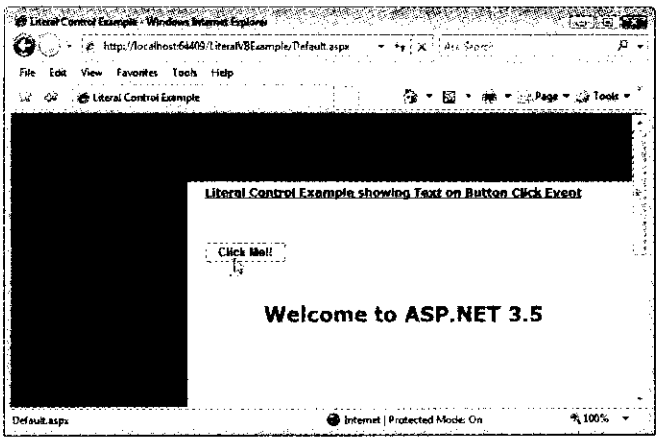


Figure 17.9: Literal Control Showing HTML Functioning on Button Click Event

The PlaceHolder Control

The PlaceHolder control is used as a container to store server controls that are added to the Web page at runtime. The PlaceHolder control does not produce any visible output and is used only as a container for other controls on the Web page. Here is the class hierarchy for the `PlaceHolder` class:

```
System.Object
  System.Web.UI.Control
      System.Web.UI.WebControls.PlaceHolder
```

A noteworthy public property of the `PlaceHolder` class is listed in Table 17.12:

Table 17.12: Public Property of the PlaceHolder Class

Property	Description
`EnableTheming`	Obtains or sets a value indicating whether themes apply to this control

Working with PlaceHolder Controls

When you add a `PlaceHolder` control on a Web page, the following code is added to the source code of the Web page:

```
<asp:PlaceHolder ID="PlaceHolder1" runat="server"></asp:PlaceHolder>
```

A control can act as a container for other controls. For example, the Web form Page class is derived from the `Control` class, so you can add controls to a Web Form with the form's `Controls.Add()` method. In fact, the `PlaceHolder` controls are designed in a way that they do not insert any HTML code into a Web page by themselves. You can add controls to them by using the `Controls.Add()` method.

Now, let's create an application named PlaceHolderVBExample. You can find the code of PlaceHolderVBExample application in the Code\ASP.NET\Chapter 17\PlaceHolderVBExample folder on the CD. When you click the `Button` control in this example, one `TextBox` control is added to the Web Form at runtime within `PlaceHolder` control. Listing 17.11 shows the `Default.aspx` page for the PlaceHolder controls:

Listing 17.11: Showing the Code for the `Default.aspx` Page

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb"
  Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head id="Head1" runat="server">
<title>Linking CSS file</title>
        <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>

<body>
  <form id="Form" runat="server">
        <div id="header">
        </div>
        <div id="sidebar">
                <div id="nav">
                         
                </div>
        </div>
        <div id="content">
                <div class ="itemContent">
                        <asp:Label ID="Label1" runat="server" Text="PlaceHolder Control
                          Example showing TextBox at Runtime"
                        Font-Bold="True" Font-Underline="True"></asp:Label>
                        <br />
                        <br />
                        <br />
                        <asp:Button ID="Button1" runat="server" Text="Click Me!!" Font-
                          Bold="True"
                        />

                        <br />
                        <br />
                        <asp:PlaceHolder ID="PlaceHolder1"
                          runat="server"></asp:PlaceHolder>
                        <div id="footer">
                                <p class="left">
                                All content copyright &copy; Kogent Solutions Inc.</p>
                        </div>
                </div>
        </div>

  </form>
</body>

</html>
```

Now, add the code in the code-behind file of the `Default.aspx` page for PlaceHolder controls, as shown in Listing 17.12:

Listing 17.12: Showing the code for the Code-Behind File of the `Default.aspx Page`

```
Partial Class _Default
  Inherits System.Web.UI.Page
  Dim Textbox1 As New System.Web.UI.WebControls.TextBox
  Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    Handles Button1.Click
      Textbox1.Text = "This is TextBox inside the PlaceHolder control"
      Textbox1.Style("width") = "280PX"
      PlaceHolder1.Controls.Add(Textbox1)
  End Sub
End Class
```

Now, run the application by pressing the F5 key and click the Click Me!! button. The output is as shown in Figure 17.10:

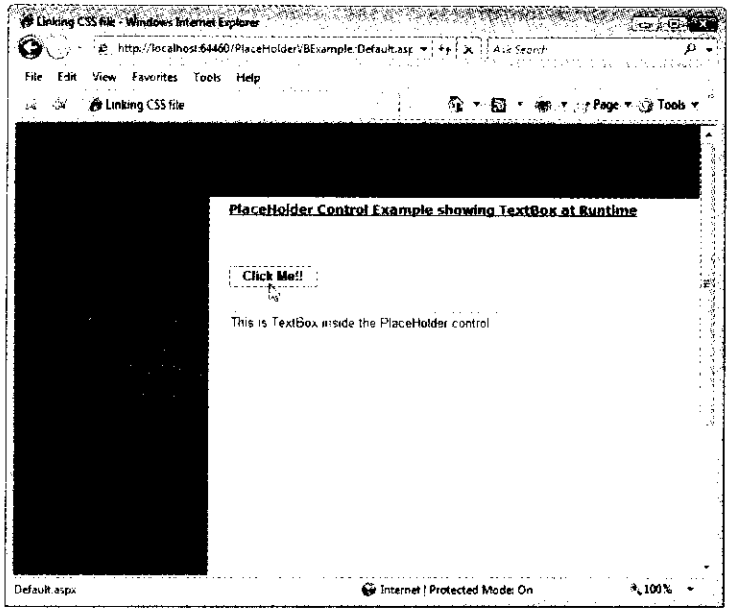


Figure 17.10: PlaceHolder Control Showing TextBox at Runtime

The HiddenField Control

The `HiddenField` control is used to store a value that needs to persist across posts to the server. Normally, view-state, session-state, and cookies are used to maintain the state of a Web form page. In case, if these methods are disabled or are not available, you can use the `HiddenField` control to store state values. Here is the class hierarchy for the `HiddenField` class:

```
System.Object
  System.Web.UI.Control
      System.Web.UI.WebControls.HiddenField
```

Noteworthy public properties of the HiddenField class are listed in Table 17.13:

Table 17.13: Public Properties of the HiddenField Class

Property	Description
`EnableTheming`	Obtains or sets a value indicating whether themes apply to this control
`SkinID`	Obtains or sets the skin to apply to the control
`Value`	Obtains or sets the value of the hidden field

Working with Hidden Field Controls

When you add a `HiddenField` control on a Web page, the following code is added to the source code of the Web page:

```
<asp:HiddenField ID="HiddenField1" runat="server" />
```

Now, let's create an application named HiddenFieldVBExample. You can find the code of HiddenFieldVBExample application in the Code\ASP.NET\Chapter 17\HiddenFieldVBExample folder on the CD. You can add a HiddenField control to a Web page either by dragging and dropping it from the Standard tab of the Toolbox or by double-clicking this control in the Toolbox. When you click the `Button` control in this example, it displays the value stored in the `HiddenField` control to the `Label` control. Listing 17.13 shows the `Default.aspx` page for the HiddenField controls:

Listing 17.13: Showing the Code for the `Default.aspx` Page

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb"
  Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
  <title> HiddenField Control Example</title>
  <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <form id="Form" runat="server">
        <div id="header">
        </div>
        <div id="sidebar">
              <div id="nav">
                     
              </div>
        </div>
        <div id="content">
              <div class ="itemContent">
                     <asp:Label ID="Label1" runat="server"
                    Text="HiddenField Control showing value on Button click" Font-
                      Bold="True"
                    Font-Underline="True"></asp:Label>
                    <br />
                    <br />
                    <br />
                    <br />
                    <asp:Button ID="Button1" runat="server" Text="Click Me!!" Font-
                      Bold="True" />
                    <br />
                    <br />
                    <asp:Label ID="Label2" runat="server"></asp:Label>
                    <br />
                    <br />
                    <asp:HiddenField ID="HiddenField1" runat="server" />
                    <div id="footer">
                          <p class="left">
                          All content copyright © Kogent Solutions Inc.</p>
                    </div>
              </div>
        </div>
  </form>
</body>
</html>
```

Now, add the code in the code-behind file of the `Default.aspx page for the HiddenField controls`, as shown in Listing 17.14:

Listing 17.14: Showing the Code for the Code-Behind File of the `Default.aspx Page`

```
Partial Class _Default
  Inherits System.Web.UI.Page
  Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    Handles Button1.Click
        Label2.Text = HiddenField1.Value
  End Sub
  Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles
    Me.Load
        HiddenField1.Value = "Welcome to the World of ASP.NET 3.5"
  End Sub
End Class
```

Now, run the application by pressing the F5 key and click the Click Me!! button. The output is as shown in Figure 17.11:

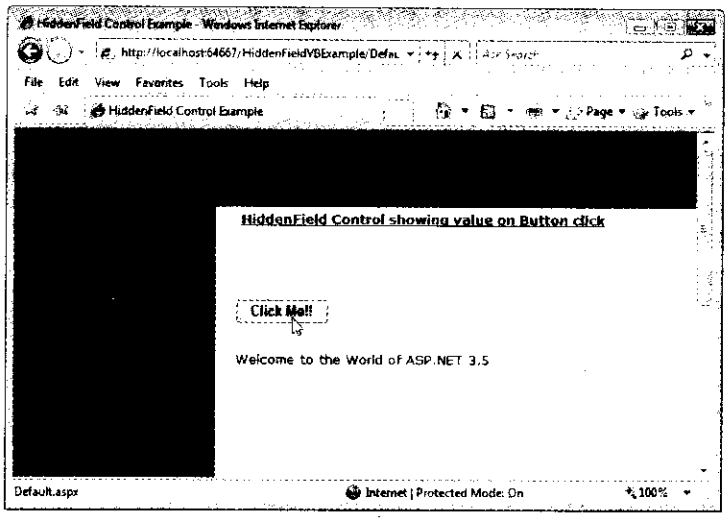


Figure 17.11: Button Click Event Showing Value of the HiddenField Control

The FileUpload Control

The `FileUpload` control displays a text box control and a browse button that enable users to browse a file from the local or remote machine to upload it on the Web server. You can upload a file on the Web server by specifying the full path of the file to be uploaded (for example, `D:\MyFiles\Test.txt`) in the textbox of this control. Alternatively, you can select the file by clicking the Browse button, and then locating it in the Choose File dialog box. Here is the class hierarchy for the `FileUpLoad` class:

```
System.Object
  System.Web.UI.Control
      System.Web.UI.WebControls.WebControl
          System.Web.UI.WebControls.FileUpload
```

Noteworthy public properties of the `FileUpLoad` class are listed in Table 17.14:

Table 17.14: Noteworthy Public Properties of the FileUpload Class	
Name	**Description**
`FileBytes`	Obtains an array of the bytes in a file that is specified by using a `FileUpload` control
`FileContent`	Obtains a Stream object that points to a file to upload using the `FileUpload` control
`FileName`	Obtains the name of a file on a client to upload using the `FileUpload` control
`HasFile`	Obtains a value indicating whether the `FileUpload` control contains a file
`PostedFile`	Obtains the underlying `HttpPostedFile` object for a file that is uploaded by using the `FileUpload` control

Noteworthy public methods of the `FileUpLoad` class are listed in Table 17.15:

Table 17.15: Noteworthy Public Method of the FileUpload Class	
Name	**Description**
`SaveAs`	Saves the data of an uploaded file to a specified path on the Web server

Working with FileUpload Control

When you add a `FileUpload` control on a Web page, the following code is added to the source code of the Web page:

```
<asp:FileUpload ID="FileUpload1" runat="server" />
```

Now, let's create applications named FileUploadVBExample. You can find the code of LabelVBExample application in the Code\ASP.NET\Chapter 17\LabelVBExample folder on the CD. You can add a FileUpload control to a Web page either by dragging and dropping it from the Standard tab of the Toolbox or by double-clicking this control in the Toolbox. Here, we are using a `FileUpload` control to browse the file and upload it on the server using the `SaveAs()` method on the `Button Click` event. Listing 17.15 shows the `Default.aspx` page for the FileUpload controls:

Listing 17.15: Showing the Code for the `Default.aspx` Page

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb"
  Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>FileUpload Control Example</title>
  <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <form id="Form" runat="server">
        <div id="header">
              </div>
              <div id="sidebar">
                    <div id="nav">
                           
                    </div>
              </div>
              <div id="content">
                    <div class ="itemContent">
                    <asp:Label ID="Label2" runat="server" Font-Bold="True"
                      Font-Underline="True"
                    Text="FileUpload Control Example"></asp:Label>
                    <br />
                    <br />
                    <br />
                     <asp:Label ID="Label1" runat="server" Text="Please
                      select a file to Upload:"></asp:Label>
                    <asp:FileUpload ID="FileUpload1" runat="server" />
                    <br />
                    <br />
                     <br />
                    <asp:Button ID="Button1" runat="server" Font-Bold="True"
                      Font-Underline="False"
                    Text="Click to Upload" Height="26px" Width="144px" />
                            
                    <asp:Label ID="Label3" runat="server"></asp:Label>
                    <div id="footer">
                    <p class="left">
                    All content copyright &copy; Kogent Solutions Inc.</p>
                    </div>
              </div>
        </div>

  </form>
</body>
</html>
```

Now, add the code in the code-behind file of the `Default.aspx` page for the FileUpload controls, as shown in Listing 17.16:

Listing 17.16: Showing the code for the Code-Behind File of the `Default.aspx` Page

```
Partial Class _Default
  Inherits System.Web.UI.Page
  Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles
    Me.Load
      If Not Page.IsPostBack Then
        Label3.Text = ""
      End If
  End Sub
  Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    Handles Button1.Click
      If FileUpload1.HasFile Then
        FileUpload1.SaveAs("D:\Projects\" & FileUpload1.FileName)
        Label3.Text = "File Uploaded"
      Else
        Label3.Text = "No uploaded file"
      End If
  End Sub
End Class
```

Now, run the application by pressing the F5 key, browse the file, and click the Click to Upload button. The output is as shown in Figure 17.12:

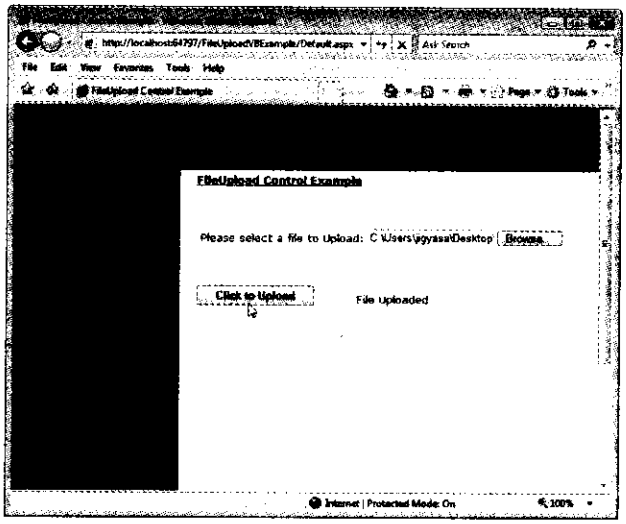


Figure 17.12: Button Click Event Uploading File on the Server Using the FieldUpload Control

NOTE

On clicking the Button control, the text box in the FileUpload Control, showing path of the uploaded file, gets empty.

The **Image** Control

The `Image` control is a standard Web server control, which is used to display an image on a Web page. This control exists within the `System.Web.UI.WebControls` namespace. You set the Uniform Resource Locator (URL) of the image with the `ImageUrl` property of the Image control. The alignment of the image in relation to other elements on the Web page is specified by setting the `ImageAlign` property. Here is the inheritance hierarchy for the `Image` class:

```
System.Object
  System.Web.UI.Control
    System.Web.UI.WebControls.WebControl
      System.Web.UI.WebControls.Image
```

This class has no non-inherited methods or events.

Noteworthy public properties of the `Image` class are listed in Table 17.16:

Table 17.16: Noteworthy Public Properties of the Image Class

`AlternateText`	Obtains or sets the alternate text displayed in the Image control when the image is not available. Browsers that support the ToolTips feature display this text as a tool tip.
`DescriptionUrl`	Obtains or sets the location to a detailed description for the image.
`Enabled`	Obtains or sets a value indicating whether the control is enabled.
`Font`	Obtains or sets the font properties for the text associated with the control.
`GenerateEmptyAlternateText`	Obtains or sets a value indicating whether the control generates an alternate text attribute for an empty string value.
`ImageAlign`	Obtains or sets the alignment of the Image control in relation to other controls on the Web page.
`ImageUrl`	Obtains or sets the location of an image to where it display in the `Image` control.

Working with **Image** Controls

When you add an `Image` control on a Web page, it adds the following code to the source code of the Web page:

```
<asp:Image ID="Image1" runat="server" />
```

Let's create an application named ImageVBExample. You can find the code of ImageVBExample application in the Code\ASP.NET\Chapter 17\ImageVBExample folder on the CD. You can add an Image control to a Web page (`Default.aspx`) either by dragging and dropping it from the Standard tab of the Toolbox or by double-clicking this control in the Toolbox. In this example, we will display the image on the `click` event of a `Button` control. To associate an image with an `Image` control, you just assign the image's URL to the `Image` control's `ImageUrl` property (if the image file is local to the project, you just set the `ImageUrl` property to the name of the image file). You can set the image's width and height with the `Width` and `Height` properties, respectively. You can also set the `AlternateText` property to the text that you want to show in case your image is not visible due to any reason.

Perform the following steps to work with an Image control:

1. Place an image in the project folder and assign the image to the `Image` control with the help of the `Select Image` dialog box, which can be opened by selecting the `Image` control in the design-view and clicking the ellipsis (...) button in front of its `ImageUrl` property in the `Properties` window. Figure 17.13 displays how to select an image with the help of `Select Image` window:

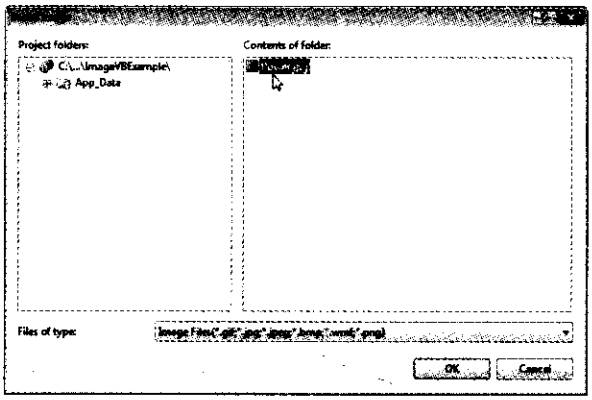


Figure 17.13: Selecting an Image in the Select Image Dialog Box

Listing 17.17 shows the `Default.aspx` page for the Image controls:

Listing 17.17: Showing the Code for the `Default.aspx` Page

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb"
  Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>Image Control Example</title>
  <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <form id="Form" runat="server">
      <div id="header">
      </div>
      <div id="sidebar">
            <div id="nav">
                   
            </div>
      </div>
      <div id="content">
            <div class ="itemContent">
                  <asp:Label ID="Label1" runat="server" Font-Bold="True" Font-
                    Underline="True"
                  Text="Image Control Example"></asp:Label>
                  <br />
                  <br />
                  <br />
                  <br />
                  <asp:Button ID="Button1" runat="server" BackColor="#000066" Font-
                    Bold="True"
                  ForeColor="#66FFFF" Height="51px"
                  Text="Click Me to See your Gift!!" />
                  <br />
                  <br />
                  <asp:Image ID="Image1" runat="server" BorderColor="#660066"
                    Height="300px"
                  ImageUrl="~/flower.jpg" ToolTip="Image control example"
                    Visible="False"
                  Width="300px" />
                   <div id="footer">
                  <p class="left">
                  All content copyright &copy; Kogent Solutions Inc.</p>
                  </div>
            </div>
      </div>

  </form>
</body>
</html>
```

2. Now, add the code in the code-behind file of the `Default.aspx` page for Image controls, as shown in Listing 17.18:

Listing 17.18: Showing the code for the Code-Behind File of the `Default.aspx` Page

```
Partial Class _Default
  Inherits System.Web.UI.Page
  Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    Handles Button1.Click
      Image1.Visible = True
      End Sub
End Class
```

3. Now, run the application by pressing the F5 key and click the Click Me to See your Gift button. After clicking the Button control, the output is as shown in Figure 17.14:

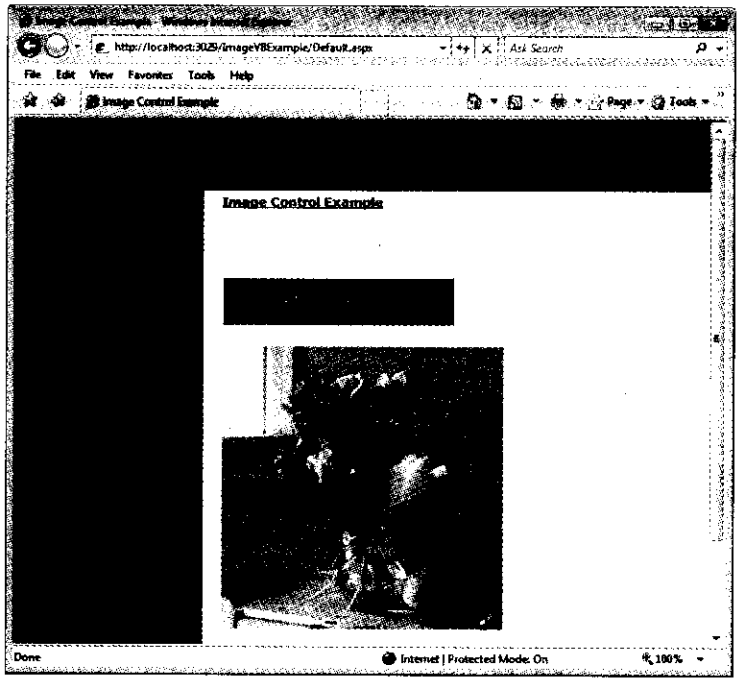


Figure 17.14: Image Control on Clicking the Button Control

The **ImageMap** Control

An image map is a picture on a Web page that provides various links, called hotspots, to navigate to other Web pages, depending on the place where the user clicks (on the single image). Web designers frequently use image maps in their websites. An image map can be included in your Web page by using the `ImageMap` control.

An `ImageMap` control exists within the `System.Web.UI.WebControls` namespace. Image maps are often real maps; for example, you can display the map of the USA and define different hotspot regions for each of its state and then navigate to the corresponding page containing the information for the selected state.

The `ImageMap` control uses the `ImageURL` property to specify the path of the image needed to be displayed and the `HotSpots` property defines a hotspot region on the image. The inheritance hierarchy for the `ImageMap` class is:

```
System.Object
    System.Web.UI.Control
        System.Web.UI.WebControls.WebControl
            System.Web.UI.WebControls.Image
                System.Web.UI.WebControls.ImageMap
```

Noteworthy public properties of the ImageMap class are listed in Table 17.17:

Table 17.17: Noteworthy Public Properties of the ImageMap Class

`Enabled`	Obtains or sets a value indicating whether the control can respond to user interaction
`HotSpotMode`	Obtains or sets the default behavior for the `HotSpot` objects of an `ImageMap` control when the `HotSpot` objects are clicked
`HotSpots`	Obtains or sets a group of `HotSpot` objects that represents the defined hotspot regions in an `ImageMap` control

Table 17.17: Noteworthy Public Properties of the ImageMap Class	
Property	Description
Target	Obtains or sets the target window to show the Web page content when the ImageMap control is clicked

A noteworthy public event of the ImageMap class is listed in Table 17.18:

Table 17.18: Noteworthy Public Event of the ImageMap Class	
Method	Description
Click	Occurs when a HotSpot object in an ImageMap control is clicked

Working with **ImageMap** Controls

When you add an ImageMap control on a Web page (Default.aspx), it adds the following code to the source code of the Web page:

```
<asp:ImageMap ID="ImageMap1" runat="server"> </asp:ImageMap>
```

Let's create an application named ImageMapVBExample. You can find the code of ImageMapVBExample application in the Code\ASP.NET\Chapter 17\ImageMapVBExample folder on the CD. Add an ImageMap control to a Web page. Now, let's see how it works by performing the following steps:

1. First, we put an image file inside the project folder and then we assign the image to the ImageMap control with the help of the Select Image dialog box, as shown in Figure 17.15, as we have done in the case of the Image control:

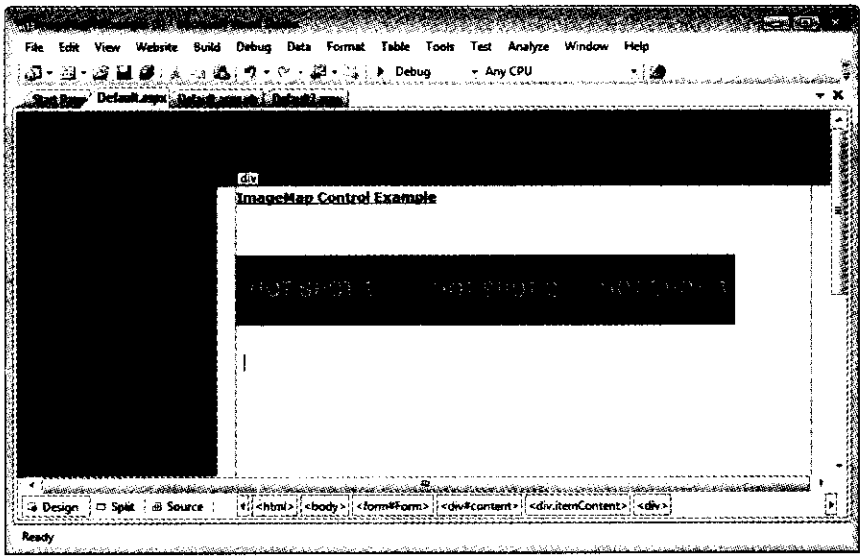


Figure 17.15: The ImageMap Control at Design-time

2. We then add three hotspots to the ImageMap control with the help of the HotSpot Collection Editor dialog box, which can be opened by selecting the ImageMap control in the design-view and clicking the ellipsis (...) button in front of its HotSpots property in the Properties window.
3. Right-click the Add button (Figure 17.16) and select RectangleHotSpot because the image that we have taken in our example is rectangular. Follow this step three times and add three RectangleHotSpots. Figure 17.16 displays the HotSpot Collection Editor dialog box with three hotspots added to it:

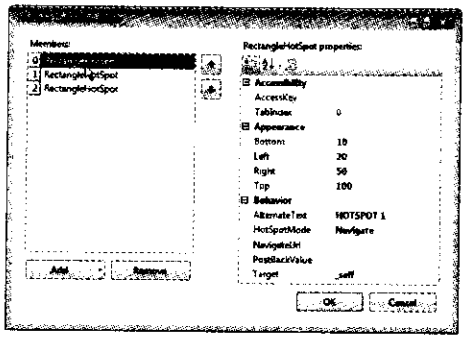


Figure 17.16: HotSpot Collection Editor Window

Add one more Web form, named as Default2.aspx, in the application. The Web form will be displayed when a user clicks the Hotspot. Listing 17.19 shows the Default.aspx page for the ImageMap controls:

Listing 17.19: Showing the Code for the `Default.aspx` Page

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb"
    Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>ImageMap Control Example</title>
    <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
        <div id="header">
        </div>
        <div id="sidebar">
            <div id="nav">
                Content
            </div>
        </div>
        <div id="content">
            <div class="itemContent">
                <div>
                    <asp:label id="Label1" runat="server" font-bold="True"
                      font-underline="True" text="ImageMap Control
                      Example"></asp:label>
                    <br />
                    <br />
                    <br />
                    <asp:imagemap id="ImageMap1" runat="server"
                      imageurl="~/SampleImageMap.jpg">
                    <asp:RectangleHotSpot AlternateText="HOTSPOT 1"
                      Bottom="10"
                    HotSpotMode="Navigate" Left="20"
                      NavigateUrl="~/Default2.aspx" Right="50"
                    Target="_self" Top="100" />
                    </asp:imagemap>
                    <br />
                    <br />
                    <br />
                    <asp:label id="Label2" runat="server"></asp:label>
                    <div id="footer">
                    <p class="left">
                    All content copyright &copy; Kogent Solutions Inc.</p>
                    </div>
                </div>
            </div>
```

```
        </div>
    </form>
</body>
</html>
```

Now, add the code in the `Default2.aspx` page for the ImageMap controls, as shown in Listing 17.20:

Listing 17.20: Showing the code for the `Default2.aspx` Page

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb"
  Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
<title>ImageMap Control Example</title>
  <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <form id="Form" runat="server">
      <div id="header">
      </div>
      <div id="sidebar">
          <div id="nav">
               
          </div>
      </div>
      <div id="content">
          <div class ="itemContent" >
          <br />
           <br />
          <br />
          <br />
          <div>
          <asp:Label ID="Label1" runat="server" Text="Welcome to the Home Page"
          Font-Bold="True" Font-Names="Elephant" Font-Size="Larger" ForeColor="Red"
          Height="104px" Width="388px"></asp:Label>
          <div id="footer">
          <p class="left">
          All content copyright &copy; Kogent Solutions Inc.</p>
          </div>
          </div>
          </div>
      </div>
  </form>
</body>
</html>
```

4. Set the `Default.aspx` page as the start page by right-clicking it, and selecting the option `Set As Start Page` from the context menu.

Now, run the application by pressing the F5 key and the output is as shown in Figure 17.17:

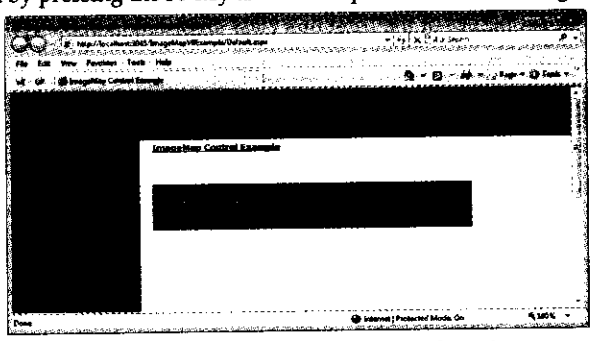


Figure 17.17: Before Clicking HotSpot 1

Click the HOT SPOT 1 hot spot; the result is as shown in Figure 17.18:

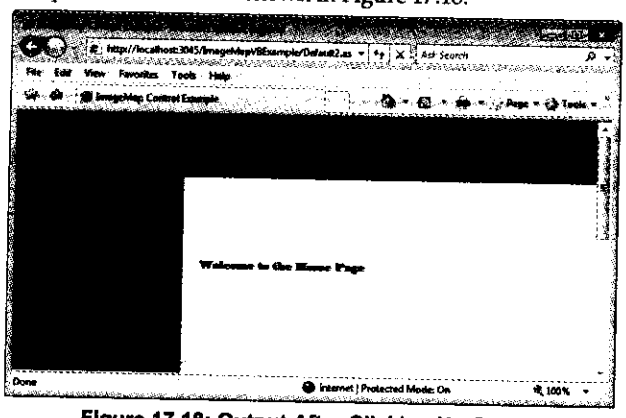


Figure 17.18: Output After Clicking HotSpot 1

NOTE

We have not provided functionality to HOT SPOT 2 and HOT SPOT 3 in this application

The **ListBox** Control

The `ListBox` control is a standard Web server control used to select one or more items from a list of items on a Web page at runtime. The inheritance hierarchy for the `ListBox` class is:

```
System.Object
    System.Web.UI.Control
        System.Web.UI.WebControls.WebControl
            System.Web.UI.WebControls.BaseDataBoundControl
                System.Web.UI.WebControls.DataBoundControl
                    System.Web.UI.WebControls.ListControl
                        System.Web.UI.WebControls.ListBox
```

You can use the `Rows` property of the ListBox control to specify the height of the control. To enable multiple item selection, you can set the `SelectionMode` property to `ListSelectionMode.Multiple`. You can find the selected item in a single-selection `ListBox` control with the help of `SelectedItem` and `SelectedIndex` properties. The `SelectedItem` property returns the selected item as a `ListItem` object, which supports `Text`, `Value`, and `Selected` properties. In multiple-selection `ListBox` controls, the loop runs over the selected items in the ListBox controls and returns each selected item as a ListItem object by using the SelectedIndex property.

This ListBox class inherits the `ListControl` class and does not have any non-inherited events. The `ListControl` class does not have any non-inherited methods either. Note that the `Items` property of the `ListControl` class returns a collection of `ListItem` objects, which you can use to access an item in a `ListBox`. Noteworthy public properties of the `ListBox` class are listed in Table 17.19:

Table 17.19: Noteworthy Public Properties of the ListBox Class

`BorderColor`	Obtains or sets the border color of the control
`BorderStyle`	Obtains or sets the border style of the control
`BorderWidth`	Obtains or sets the border width for the control
`Rows`	Obtains or sets the number of rows displayed in the ListBox control
`SelectionMode`	Obtains or sets the selection mode of the `ListBox` control

Working with **ListBox** Controls

When you add a `ListBox` control on a Web page (`Default.aspx`), it adds the following code to the source code of the Web page:

```
<asp:ListBox ID="ListBox1" runat="server"></asp:ListBox>
```

Let's create an application named ListBoxVBExample. You can find the code of ListBoxVBExample application in the Code\ASP.NET\Chapter 17\ListBoxVBExample folder on the CD. You can add a ListBox control to a Web page either by dragging and dropping it from the Standard tab of the Toolbox or by double-clicking this control in the Toolbox.

In this example, we have taken a `ListBox` control on a Web page and adding data to it dynamically at runtime from `Northwind` database through a `Data Source Configuration Wizard`. We have taken one more `ListBox` control to display selected items of the first `ListBox` control. Let's see it's functioning by following these steps:

1. Click the `Choose Data Source` option in the `ListBox Tasks` pane, as shown in Figure 17.19:

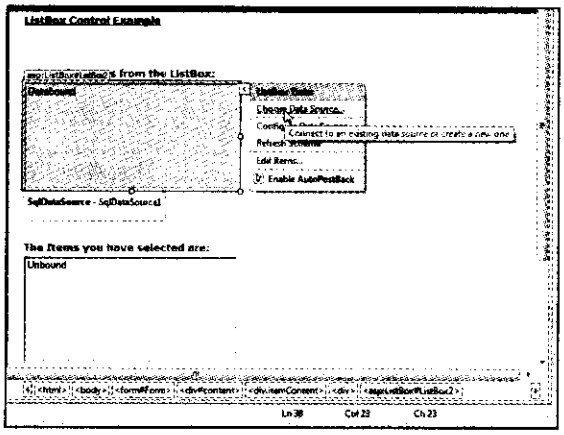


Figure 17.19: Choosing Data Source Option in the ListBox Tasks Pane

The Data Source Configuration Wizard opens.

2. Select the `New Data Source` option from the `Data Source Configuration Wizard`, as shown in Figure 17.20:

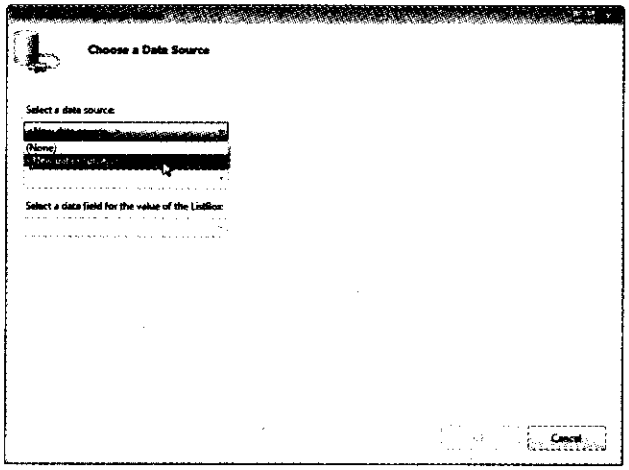


Figure 17.20: Selecting a New Data Source

It opens the Data Source Configuration Wizard showing data source types.

3. Select `Database` option as data source, as shown in Figure 17.21:

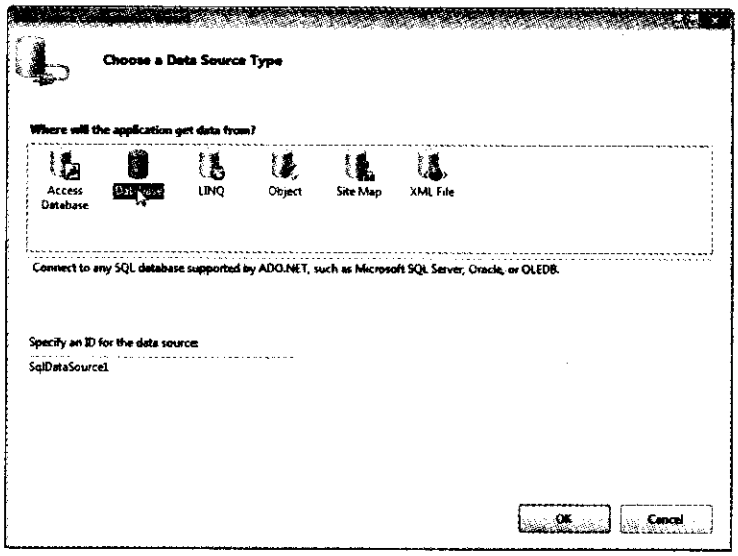


Figure 17.21: Selecting Database Option

4. Choose the existing data connection or create new one. We have selected a New Connection in our example as shown in Figure 17.22:

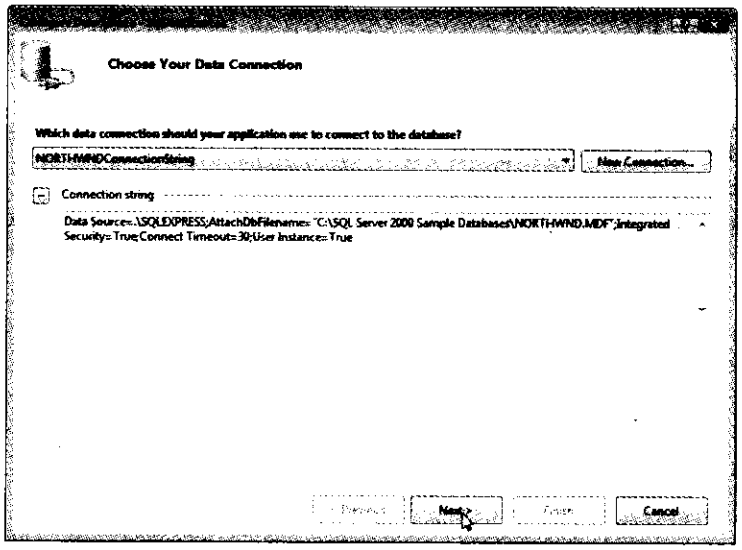


Figure 17.22: Showing Connection Options

5. Select the table and column name; here, we have selected Products table's ProductName column, as shown in Figure 17.23.
6. Finally, click the `Finish` button to complete the wizard, as shown in Figure 17.23:

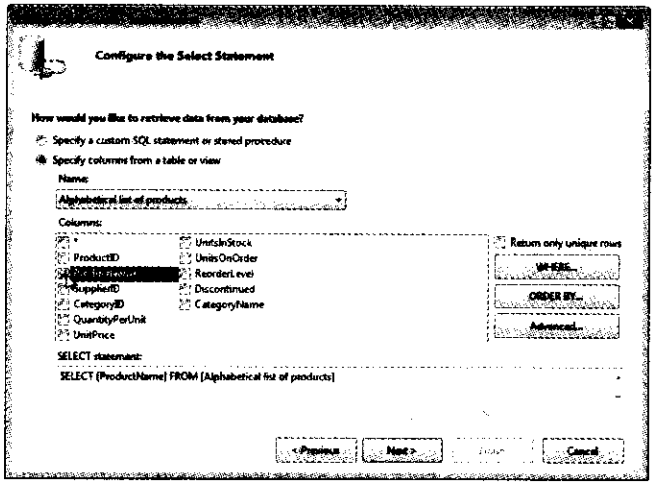


Figure 17.23: Selecting ProductName Column

It adds the SqlDataSource control at design-time. Listing 17.21 shows the Default.aspx page for the ListBox controls:

NOTE

Create / configure the connection string according to your own system throughout the chapter.

Listing 17.21: Showing the Code for the `Default.aspx` Page

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb"
  Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<head id="Head1" runat="server">
  <title>ListBox Control Example</title>
  <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <form id="form" runat="server">
        <div id="header">
        </div>
        <div id="sidebar">
              <div id="nav">
                     
              </div>
        </div>
        <div id="content">
              <div class ="itemContent" >
                    <div>
                          <asp:Label ID="Label1" runat="server" Font-Bold="True"
                            Font-Underline="True"
                          Text="ListBox Control Example"></asp:Label>
                          <br />
                          <br />
                          <br />
                          <asp:Label ID="Label2" runat="server" Font-Bold="True"
                            Font-Size="Small"
```

```
                ForeColor="#FF0066" Text="Select the Items from the
                  ListBox:"></asp:Label>
                <br />
                <asp:ListBox ID="ListBox2" runat="server"
                  AutoPostBack="True"
                DataSourceID="SqlDataSource1" DataTextField="ProductName"
                DataValueField="ProductName" Height="139px"
                Width="280px">
                </asp:ListBox>
                <asp:SqlDataSource ID="SqlDataSource1" runat="server"
                ConnectionString="<%$
                  ConnectionStrings:NORTHWNDConnectionString %>"
                SelectCommand="SELECT [ProductName] FROM [Alphabetical
                  list of products]">
                </asp:SqlDataSource>

                <br />
                <asp:Label ID="Label3" runat="server" Font-Bold="True"
                  Font-Size="Small"
                ForeColor="#FF0066" Text="The Items you have selected
                  are:"></asp:Label>
                <br />
                <asp:ListBox ID="ListBox1" runat="server" Height="139px"
                  Width="280px">
                </asp:ListBox>
                <div id="footer">
                <p class="left">
                All content copyright &copy; Kogent Solutions Inc.</p>
                </div>
            </div>
        </div>
    </div>
</form>

</body>

</html>
```

Now, add the code in the code-behind file of the `Default.aspx` page for the ListBox controls, as shown in Listing 17.22:

Listing 17.22: Showing the code for the Code-Behind File of the `Default.aspx` Page

```
Partial Class _Default
  Inherits System.Web.UI.Page

  Protected Sub ListBox2_SelectedIndexChanged(ByVal sender As Object, ByVal e As
    System.EventArgs) Handles ListBox2.SelectedIndexChanged
      Dim dt As ListItem
      For Each dt In ListBox2.Items
          If dt.Selected = True Then
              ListBox1.Items.Add(dt.Text)
          End If
      Next

  End Sub
End Class
```

Now, run the application by pressing the F5 key and select items from the listbox. The output is as shown in Figure 17.24:

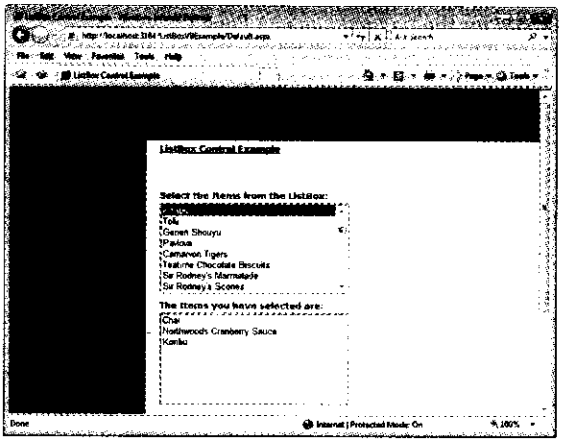


Figure 17.24: ListBox Control at Runtime

The **DropDownList** Control

The `DropDownList` control displays the list of data as a drop-down list from which you can make a single selection. The `DropDownList` control exists within the `System.Web.UI.WebControls` namespace. You cannot select multiple items in this control because when you make a selection from the list, the list closes automatically. Here is the inheritance hierarchy for the `DropDownList` class:

```
System.Object
    System.Web.UI.Control
        System.Web.UI.WebControls.WebControl
            System.Web.UI.WebControls.BaseDataBoundControl
                System.Web.UI.WebControls.DataBoundControl
                    System.Web.UI.WebControls.ListControl
                        System.Web.UI.WebControls.DropDownList
```

The `DropDownList` control has no non-inherited methods or events. This class inherits the `ListControl` class. Noteworthy public properties of the `DropDownList` class are listed in Table 17.20:

Table 17.20: Noteworthy Public Properties of the DropDownList Class

`BorderColor`	Obtains or sets the border color of the control
`BorderStyle`	Obtains or sets the border style of the control
`BorderWidth`	Obtains or sets the border width for the control
`SelectedIndex`	Obtains or sets the index of the selected item in the control

Working with **DropDownList** Controls

When you add a `DropDownList` control on a Web page (`Default.aspx`), it adds the following code to the source code of the Web page:

```
<asp:DropDownList ID="DropDownList1" runat="server"></asp:DropDownList>
```

Let's create an application named DropDownListVBExample. You can find the code of DropDownListVBExample application in the Code\ASP.NET\Chapter 17\DropDownListVBExample folder on the CD. To bind data dynamically at runtime from the `Northwind` database, follow the procedure, we have used for the `ListBox` control previously. In this example, we display data of Products table's ProductName column of the `Northwind` database, and also display the selected item on the Web page. Listing 17.23 shows the Default.aspx page for the `DropDownList` controls:

Listing 17.23: Showing the Code for the `Default.aspx` Page

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb"
  Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>DropDownList Control Example</title>
  <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <form id="Form" runat="server">
        <div id="header">
        </div>
        <div id="sidebar">
              <div id="nav">
                     
              </div>
        </div>
        <div id="content">
              <div class ="itemContent" >
                    <div>
                    <asp:Label ID="Label1" runat="server" Font-Bold="True"
                      Font-Underline="True"
                    Text="DropDownList Control Example" Font-Size="Medium"
                      ForeColor="Black"></asp:Label>
                    <br />
                    <br />
                    <asp:Label ID="Label2" runat="server" Font-Size="Small"
                      ForeColor="#FF0066"
                    Text="Select the Product Name:"></asp:Label>
                    <br />
                    <asp:DropDownList ID="DropDownList1" runat="server"
                      AutoPostBack="True"
                    DataSourceID="SqlDataSource1" DataTextField="ProductName"
                    DataValueField="ProductName" Height="76px"
                    onselectedindexchanged="DropDownList1_SelectedIndexChanged"
                      Width="227px"
                    style="margin-bottom: 0px">
                    </asp:DropDownList>
                    <asp:SqlDataSource ID="SqlDataSource1" runat="server"
                    ConnectionString="<%$ ConnectionStrings:NORTHWNDConnectionString %>"
                    SelectCommand="SELECT [ProductName] FROM [Alphabetical list of
                      products]">
                    </asp:SqlDataSource>
                    <br />
                    <br />
                    <br />
                    <br />
                    <br />
                    <br />
                    <asp:Label ID="Label4" runat="server" Font-Size="Small"
                    ForeColor="Red"></asp:Label>
                    <div id="footer">
                          <p class="left">
                        All content copyright &copy; Kogent Solutions Inc.</p>
                    </div>
                    </div>
              </div>
        </div>
```

```
    </form>
</body>
</html>
```

Now, add the code in the code-behind file of the `Default.aspx` page for the `DropDownList` controls, as shown in Listing 17.24:

Listing 17.24: Showing the code for the Code-Behind File of the `Default.aspx` Page

```
Partial Class _Default
    Inherits System.Web.UI.Page
    Protected Sub DropDownList1_SelectedIndexChanged(ByVal sender As Object, ByVal e As
      System.EventArgs) Handles DropDownList1.SelectedIndexChanged
        Label4.Text = "You have selected: " + DropDownList1.SelectedItem.Text
    End Sub
End Class
```

Now, run the application by pressing the F5 key and select the item from the drop down list. The output is as shown in Figure 17.25:

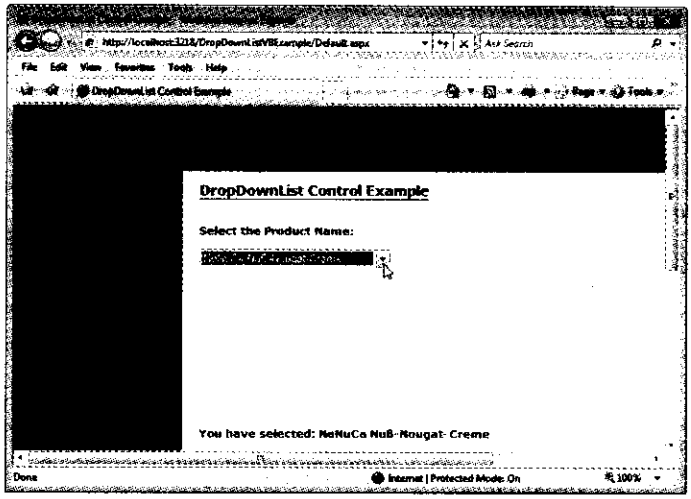


Figure 17.25: DropDownList Control at Runtime

The **BulletedList** Control

The `BulletedList` control is a standard Web server control used to display items in the form of a bulleted list on a Web page. Here is the inheritance hierarchy for the `BulletedList` class:

```
System.Object
    System.Web.UI.Control
        System.Web.UI.WebControls.WebControl
            System.Web.UI.WebControls.BaseDataBoundControl
                System.Web.UI.WebControls.DataBoundControl
                    System.Web.UI.WebControls.ListControl
                        System.Web.UI.WebControls.BulletedList
```

This control has no non-inherited methods. The BulletedList class inherits the `ListControl` class. Noteworthy public properties of the `BulletedList` class are listed in Table 17.21:

Table 17.21: Noteworthy Public Properties of the BulletedList Class

`AutoPostBack`	Obtains or sets the value of the `AutoPostBack` property of the control for the base class
`BulletImageUrl`	Obtains or sets the path to an image to display each bullet in a control

Table 17.21: Noteworthy Public Properties of the BulletedList Class	
BulletStyle	Obtains or sets the style of the bullets for the BulletedList control
Controls	Obtains a ControlCollection collection for the control
DisplayMode	Obtains or sets the display mode of the list items appearing in a BulletedList control
FirstBulletNumber	Obtains or sets the value that starts the numbering of list items in an ordered BulletedList control
SelectedIndex	Obtains or sets the zero-based index of the currently selected item in a BulletedList control
SelectedItem	Obtains the currently selected item in a BulletedList control
SelectedValue	Obtains or sets the Value property of the selected ListItem object in the BulletedList control
Target	Obtains or sets the target window to display the data of the Web page that is linked to when a hyperlink in a BulletedList control is clicked
Text	Obtains or sets the text for the BulletedList control

Noteworthy public event of the BulletedList class is listed in Table 17.22:

Table 17.22: Noteworthy Public Event of the BulletedList Class	
Click	Occurs when a link button in a BulletedList control is clicked

To specify the bullet type to display the list items in a BulletedList control, you can set the BulletStyle property to one of the bullet types that are defined by the BulletStyle enumeration. Table 17.23 lists the available bullet styles:

Table 17.23: Bullet styles of the BulletedList Control	
NotSet	Represents an unspecified bulleted list style; the browser automatically sets the style of the bulleted list
Numbered	Represents the bullets in the form of numbers
LowerAlpha	Represents the bullets in the form of lowercase letters
UpperAlpha	Represents the bullets in the form of uppercase letters
LowerRoman	Represents the bullets in the form of lowercase Roman numerals
UpperRoman	Represents the bullets in the form of uppercase Roman numerals
Disc	Represents the bullets in the form of a filled circle
Circle	Represents the bullets in the form of a empty circle
Square	Represents the bullets in the form of a filled square
CustomImage	Represents the bullets in the form of a custom image

To specify the display behavior of the list items in a BulletedList control, you can set the DisplayMode property to one of the Text, HyperLink, and LinkButton options.

Working with **BulletedList** Controls

When you add a `BulletedList` control on a Web page (`Default.aspx`), it adds the following code to the source code of the Web page:

```
<asp:BulletedList ID="BulletedList1" runat="server"></asp:BulletedList>
```

Let's create an application named BulletedListVBExample. You can find the code of BulletedListVBExample application in the Code\ASP.NET\Chapter 17\BulletedListVBExample folder on the CD. To bind the data dynamically at runtime from the `PUBS` database, follow the procedure that we have used for the ListBox control. In this example, we have displayed data from `authors` table's `au_fname` column of the `Pubs` database, and also displaying the photo of the employees by clicking the `BulletedList` control items, whose `DisplayMode` property is set to `LinkButton`. Listing 17.25 shows the Default.aspx page for the `BulletedList` controls:

Listing 17.25: Showing the Code for the `Default.aspx` Page

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb"
  Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>BulletedList Control Example</title>
<link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <form id="Form" runat="server">
        <div id="header">
        </div>
        <div id="sidebar">
              <div id="nav">
                     
              </div>
        </div>
        <div id="content">
              <div class ="itemContent" >
              <asp:Label ID="Label1" runat="server" Font-Bold="True" Font-Size="Medium"
              Font-Underline="True" ForeColor="Black" Text="BulletedList Control
                Example"></asp:Label>
               
              <br />
                         
                           
                           
                           
                           
                        
              <br />
              <br />
                
              <asp:Image ID="Image1" runat="server" Height="150px" Visible="False"
              Width="100px" />
                         
                           
                           
                           
                           
                           
                      
              <br />
              <br />
              <asp:Label ID="Label2" runat="server" Font-Size="Small"
                ForeColor="#FF0066"
              Text="Click the name to see the employee's photo"></asp:Label>
```

```
                <br />
                <asp:BulletedList ID="BulletedList1" runat="server"
                DataSourceID="SqlDataSource1" DataTextField="au_fname"
                DataValueField="au_fname" DisplayMode="LinkButton" Font-Size="X-Small"
                ForeColor="Black" onclick="BulletedList1_Click1">
                </asp:BulletedList>
                <asp:SqlDataSource ID="SqlDataSource1" runat="server"
                ConnectionString="<%$ ConnectionStrings:PUBSConnectionString %>"
                SelectCommand="SELECT [au_fname] FROM [authors]"></asp:SqlDataSource>
                <div id="footer">
                        <p class="left">
                        All content copyright &copy; Kogent Solutions Inc.</p>
                </div>
                </div>
        </div>
    </form>
</body>
</html>
```

Now, add the code in the code-behind file of the `Default.aspx` page for the `BulletedList controls`, as shown in Listing 17.26:

Listing 17.26: Showing the code for the Code-Behind File of the `Default.aspx Page`

```
Partial Class _Default
    Inherits System.Web.UI.Page
    Protected Sub BulletedList1_Click1(ByVal sender As Object, ByVal e As
     System.Web.UI.WebControls.BulletedListEventArgs) Handles BulletedList1.Click
        Image1.Visible = True
        If e.Index = 0 Then
            Image1.ImageUrl = "Abraham.jpg"
        End If
        If e.Index = 1 Then
            Image1.ImageUrl = "Reginald.jpg"
        End If
    End Sub
End Class
```

Now, run the application by pressing the F5 key. The output is as shown in Figure 17.26:

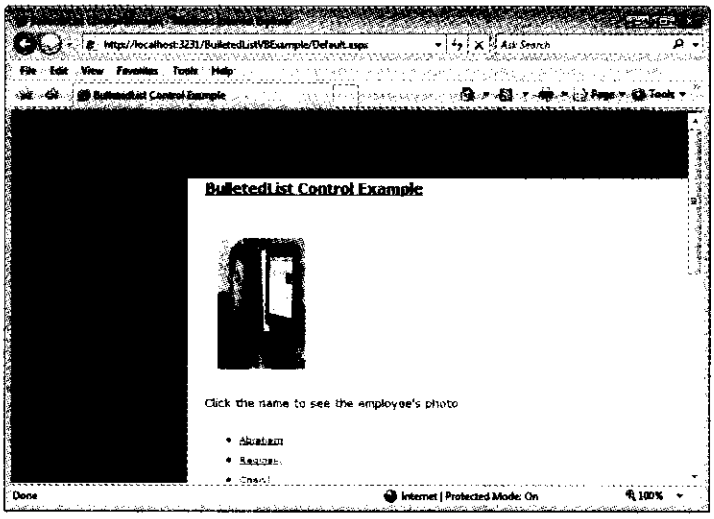


Figure 17.26: Displaying the Image on the Click Event of the BulletedList Item2

NOTE

In this example, we have set images for only index numbers zero(0) and one(1). You can set images for other indexes also by following the same procedure.

The **HyperLink** Control

The `HyperLink` control is used to create a link to another Web page that can be a page in your Web application or anywhere else on the World Wide Web (WWW). The `HyperLink` control exists within the `System.Web.UI.WebControls` namespace. Here is the inheritance hierarchy for the `HyperLink` class:

```
System.Object
   System.Web.UI.Control
      System.Web.UI.WebControls.webControl
         System.Web.UI.WebControls.HyperLink
```

You can specify the location of the linked page by specifying its URL on the current page. You can use text as well as an image in the `HyperLink` control. Text is specified with the `Text` property and image is specified by the `ImageUrl` property. By default, when you click a `HyperLink` control, the hyperlinked content appears in a new browser window. You can set the `Target` property to the name of a window or frame, as listed in Table 17.24:

Table 17.24: Target Property Options of the HyperLink Class

`_blank`	Displays the hyperlinked content in a new window without frames
`_parent`	Displays the hyperlinked content in the immediate frameset parent
`_self`	Displays the hyperlinked content in the frame with focus
`_top`	Displays the hyperlinked content in the full window without frames

This class has no non-inherited method or event. Noteworthy public properties of the `HyperLink` class are listed in Table 17.25:

Table 17.25: Noteworthy Public Properties of the HyperLink Class

`ImageUrl`	Obtains the path to an image to be displayed for the `HyperLink` control
`NavigateUrl`	Obtains the URL to link to the Web page when the `HyperLink` control is clicked
`Target`	Obtains the target window to display the Web page data when the `HyperLink` control is clicked
`Text`	Obtains the text caption for the `HyperLink` control

Working with **HyperLink** Controls

When you add a `HyperLink` control to a Web page (`Default.aspx`), it adds the following code to the source code of the Web page:

```
<asp:HyperLink ID="HyperLink1" runat="server">HyperLink</asp:HyperLink>
```

Let's create an application named HyperLinkVBExample. You can find the code of HyperLinkVBExample application in the Code\ASP.NET\Chapter 17\HyperLinkVBExample folder on the CD. In this example, we have set the `NavigateUrl` property of the `Hyperlink` control to `http:\\www.google.co.in` Web page. Listing 17.27 shows the Default.aspx page for the HyperLink controls:

Listing 17.27: Showing the Code for the `Default.aspx` Page

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb"
    Inherits="_Default" %>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>HyperLink Control Example</title>
    <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="Form" runat="server">
        <div id="header">
        </div>
        <div id="sidebar">
            <div id="nav">
                 
            </div>
        </div>
        <div id="content">
            <div class ="itemContent" >
            <asp:Label ID="Label1" runat="server" Font-Bold="True" Font-Size="Medium"
            Font-Underline="True" ForeColor="Black" Text="HyperLink Control
              Example"></asp:Label>
            <br />
            <br />
            <br />
            <asp:HyperLink ID="HyperLink1" runat="server" Font-Size="Small"
            ForeColor="Blue" Target="_blank"
              NavigateUrl="http://www.google.co.in">Click me to move to the google
              website</asp:HyperLink>
            <br />
            <div id="Footer">
                <p class="left">
                All content copyright &copy; Kogent Solutions Inc.</p>
            </div>
            </div>
        </div>
    </form>
</body>
</html>
```

Now, run the application by pressing the F5 key. The output is as shown in Figure 17.27:

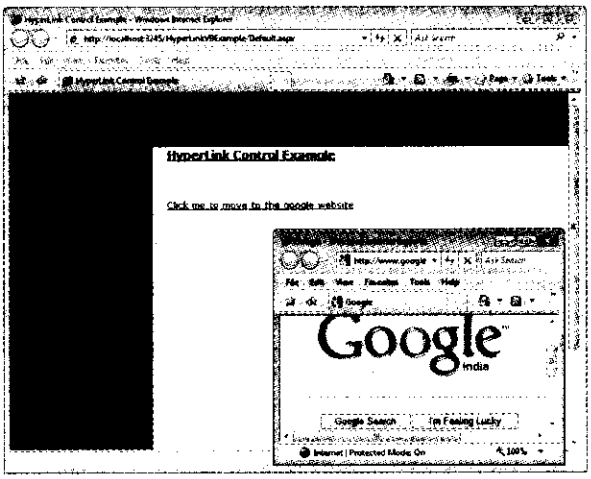


Figure 17.27: Google Page After Clicking HyperLink Control

The **LinkButton** Control

The `LinkButton` control is another standard Web server control used to link the current Web page to some other Web page, similar to the `HyperLink` control. The only difference between the two is that the `HyperLink` control just allows the browser to navigate to a new Web page; whereas in the `LinkButton` control, we can also perform some other action by handling the `Click` and `Command` events of this control. Here is the inheritance hierarchy for the `LinkButton` class:

```
System.Object
    System.Web.UI.Control
        System.Web.UI.WebControls.WebControl
            System.Web.UI.WebControls.LinkButton
```

This class has no non-inherited method. Noteworthy public properties of the `LinkButton` class are listed in Table 17.26:

Table 17.26: Noteworthy Public Properties of the LinkButton Class

`CausesValidation`	Obtains a value showing, if validation is performed whether the `LinkButton` control is clicked or not
`CommandArgument`	Obtains an optional argument passed to the Command event handler along with the related CommandName property
`CommandName`	Retrieves the command name associated with the `LinkButton` control. This value is passed to the Command event handler along with the CommandArgument property
`OnClientClick`	Obtains the client-side script that executes when a `LinkButton` control's Click event is fired
`PostBackUrl`	Obtains the URL of the page from the current page when the `LinkButton` control is clicked
`Text`	Obtains the text caption displayed on the `LinkButton` control
`ValidationGroup`	Obtains the collection of controls for which the `LinkButton` control causes validation when it posts back to the server

Noteworthy public events of the `LinkButton` control are listed in Table 17.27:

Table 17.27: Noteworthy Public Events of the LinkButton Class

`Click`	Raises when the `LinkButton` control is clicked and no command name is associated with this control
`Command`	Raises when the `LinkButton` control is clicked and a command name is associated with this control

Working with LinkButton Controls

When you add a `LinkButton` control on a Web page (`Default.aspx`), it adds the following code to the source code of the Web page:

```
<asp:LinkButton ID="LinkButton1" runat="server">LinkButton</asp:LinkButton>
```

Let's create an application named LinkButtonVBExample. You can find the code of LinkButtonVBExample application in the Code\ASP.NET\Chapter 17\LinkButtonVBExample folder on the CD. In this example, we have redirected the current page to the `http://www.google.co.in` Web page on the `click` event of the `LinkButton` control. Listing 17.28 shows the `Default.aspx` page for the LinkButton controls:

Listing 17.28: Showing the Code for the `Default.aspx` Page

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb"
    Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>LinkButton Control Example</title>
<link href="styleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="Form" runat="server">
        <div id="header">
        </div>
        <div id="sidebar">
            <div id="nav">
                 
            </div>
        </div>
        <div id="content">
            <div class ="itemContent" >
                <div>
                <asp:Label ID="Label1" runat="server" Font-Bold="True" Font-
                  Size="Medium"
                Font-Underline="True" ForeColor="Black" Text="LinkButton Control
                  Example"></asp:Label>
                <br />
                <br />
                <br />
                <asp:LinkButton ID="LinkButton1" runat="server" BackColor="White"
                BorderColor="#00CCFF" Font-Names="Tahoma" Font-Size="Small"
                  ForeColor="#0000CC"
                ToolTip="Google Website">LinkButton</asp:LinkButton>
                <div id="footer">
                    <p class="left">
                    All content copyright &copy; Kogent Solutions Inc.</p>
                </div>
                </div>
            </div>
        </div>
    </form>
</body>
</html>
```

Now, add the code in the code-behind file of the `Default.aspx` page for the LinkButton controls, as shown in Listing 17.29:

Listing 17.29: Showing the code for the Code-Behind File of the `Default.aspx Page`

```
Partial Class _Default
    Inherits System.Web.UI.Page
    Protected Sub LinkButton1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
      Handles LinkButton1.Click
        Response.Redirect("http://www.google.co.in")
    End Sub
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles
      Me.Load
        LinkButton1.Text = "Click me to move to Google website"
    End Sub
End Class
```

Now, run the application by pressing the F5 key. The output is as shown in Figure 17.28:

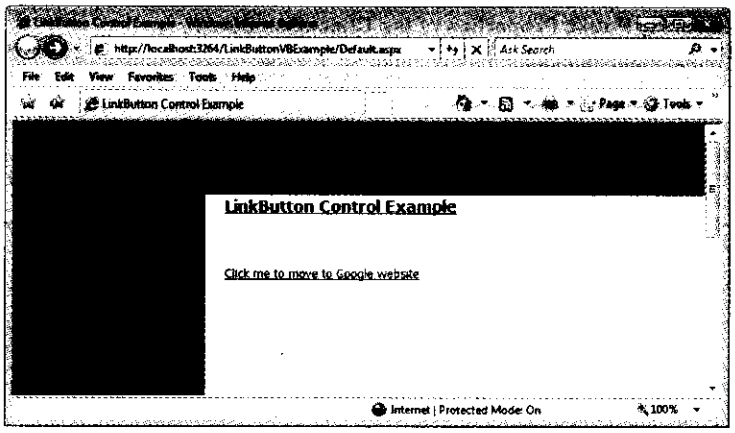


Figure 17.28: Showing the LinkButton Control at Runtime

Now, click the `LinkButton` control. It will redirect you to the `Google` website, as shown in Figure 17.29:

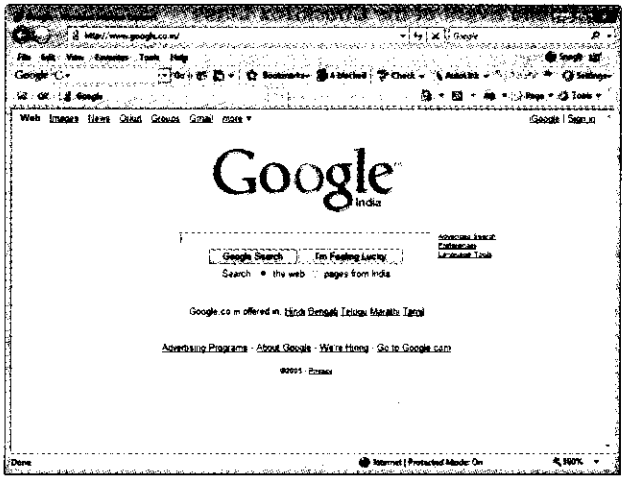


Figure 17.29: Showing the Google Website After Clicking the LinkButton Control

The **CheckBox** Control

The `CheckBox` control creates a check box that can be selected by clicking it. This control exists within the `System.Web.UI.WebControls` namespace. The `CheckBox` control displays checkmarks that allow the user to toggle between a True or False condition. Here is the class hierarchy of the `CheckBox` class:

```
System.Object
    System.Web.UI.Control
        System.Web.UI.WebControls.WebControl
            System.Web.UI.WebControls.CheckBox
```

Noteworthy public properties of the `CheckBox` class are listed in Table 17.28:

Table 17.28: Noteworthy Public Properties of the CheckBox Class

`AutoPostBack`	Obtains a value showing, if the `CheckBox` state automatically posts back to the server when clicked or not
`CausesValidation`	Obtains or sets a `Boolean` value showing if validation is performed whether the `CheckBox`

Table 17.28: Noteworthy Public Properties of the CheckBox Class	
[illegible]	control is clicked or not
Checked	Obtains a value showing, if the CheckBox control is checked or not
InputAttributes	Obtains a reference to the collection of attributes for all the rendered input elements of the CheckBox control
LabelAttributes	Obtains a reference to the collection of attributes for all the rendered LABEL element of the CheckBox control
Text	Obtains the text label associated with the CheckBox control
TextAlign	Obtains the placement of the text label associated with the CheckBox control
ValidationGroup	Obtains or sets the group of controls for which the CheckBox control will cause validation on its post back

A noteworthy public event of the CheckBox class is listed in Table 17.29:

Table 17.29: Noteworthy Public Event of the CheckBox Class	
CheckedChanged	Occurs when the value of the Checked property changes between various posts to the server

Working with **CheckBox** Controls

When you add a CheckBox control on a Web page (Default.aspx), it adds the following code to the source code of the Web page:

```
<asp:CheckBox ID="CheckBox1" runat="server" />
```

Let's create an application named CheckBoxVBExample. You can find the code of CheckBoxVBExample application in the Code\ASP.NET\Chapter 17\CheckBoxVBExample folder on the CD. In this example, we have placed five CheckBox controls on the Web page showing car names. The item attached to the CheckBox control appears in the ListBox control whenever you select the item, and is removed from the ListBox control when you clear the CheckBox control. Listing 17.30 shows the Default.aspx page for the CheckBox controls:

Listing 17.30: Showing the Code for the Default.aspx Page

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb"
  Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>CheckBox Control Example</title>
<link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <form id="Form" runat="server">

      <div id="header">
      </div>
      <div id="sidebar">
            <div id="nav">
                   
            </div>
      </div>
      <div id="content">
            <div class ="itemContent" >
```

```
                <div>
                <asp:Label ID="Label1" runat="server" Font-Bold="True" Font-Size="Medium"
                Font-Underline="True" Text="CheckBox Example"></asp:Label>
                <br />
                <br />
                <asp:Label ID="Label2" runat="server" Font-Size="Small"
                  ForeColor="#FF0066"
                Text="which of these cars you own?"></asp:Label>
                <br />
                <br />
                <asp:CheckBox ID="CheckBox1" runat="server" AutoPostBack="True"
                Font-Size="Small" Text="Maruti Zen" />
                          
                           
                          
                <br />
                <asp:CheckBox ID="CheckBox2" runat="server" AutoPostBack="True"
                Font-Size="Small" Text="Honda City" />
                <br />
                <asp:CheckBox ID="CheckBox3" runat="server" AutoPostBack="True"
                Font-Size="Small" Text="Toyota Corolla" />
                <br />
                <asp:CheckBox ID="CheckBox4" runat="server" AutoPostBack="True"
                Font-Size="Small" Text="Hundai Santro" />
                <br />
                <asp:CheckBox ID="CheckBox5" runat="server" AutoPostBack="True"
                Font-Size="Small" Text="Maruti Swift" />
                <br />
                <br />
                <br />
                <asp:ListBox ID="ListBox1" runat="server" Height="102px" Width="118px">
                </asp:ListBox>
                <div id="footer">
                        <p class="left">
                        All content copyright &copy; Kogent Solutions Inc.</p>
                </div>
                </div>
                </div>
        </div>
    </form>
</body>
</html>
```

Now, add the code in the code-behind file of the `Default.aspx` page for the CheckBox controls, as shown in Listing 17.31:

Listing 17.31: Showing the code for the Code-Behind File of the `Default.aspx` Page

```
Partial Class _Default
    Inherits System.Web.UI.Page
    Protected Sub CheckBox1_CheckedChanged(ByVal sender As Object, ByVal e As
     System.EventArgs) Handles CheckBox1.CheckedChanged
        If CheckBox1.Checked = True Then
            ListBox1.Items.Add(CheckBox1.Text)
        Else
            ListBox1.Items.Remove(ListBox1.Items.FindByText(CheckBox1.Text))
        End If
    End Sub
    Protected Sub CheckBox2_CheckedChanged(ByVal sender As Object, ByVal e As
     System.EventArgs) Handles CheckBox2.CheckedChanged
        If CheckBox2.Checked = True Then
            ListBox1.Items.Add(CheckBox2.Text)
```

```
        Else
            ListBox1.Items.Remove(ListBox1.Items.FindByText(CheckBox2.Text))
        End If
    End Sub
    Protected Sub CheckBox3_CheckedChanged(ByVal sender As Object, ByVal e As
     System.EventArgs) Handles CheckBox3.CheckedChanged
        If CheckBox3.Checked = True Then
            ListBox1.Items.Add(CheckBox3.Text)
        Else
            ListBox1.Items.Remove(ListBox1.Items.FindByText(CheckBox3.Text))
        End If
    End Sub
    Protected Sub CheckBox4_CheckedChanged(ByVal sender As Object, ByVal e As
     System.EventArgs) Handles CheckBox4.CheckedChanged
        If CheckBox4.Checked = True Then
            ListBox1.Items.Add(CheckBox4.Text)
        Else
            ListBox1.Items.Remove(ListBox1.Items.FindByText(CheckBox4.Text))
        End If
    End Sub
    Protected Sub CheckBox5_CheckedChanged(ByVal sender As Object, ByVal e As
     System.EventArgs) Handles CheckBox5.CheckedChanged
        If CheckBox5.Checked = True Then
            ListBox1.Items.Add(CheckBox5.Text)
        Else
            ListBox1.Items.Remove(ListBox1.Items.FindByText(CheckBox5.Text))
        End If
    End Sub
End Class
```

Now, run the application by pressing the F5 key and select the check box. The output is as shown in Figure 17.30:

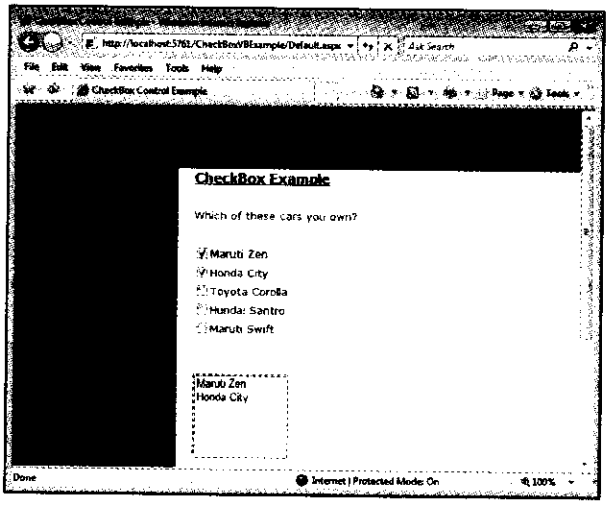


Figure 17.30: Selecting CheckBox Controls in ListBox Control

The **RadioButton** Control

Similar to the `CheckBox` control, a `RadioButton` control creates a single radio button. The `RadioButton` control exists within the `System.Web.UI.WebControls` namespace. The `RadioButton` controls can be grouped, where only one `RadioButton` control can be selected at a time. In Web Forms, you have to set the

RadioButton' s GroupName property to the same value to associate them into a group. Here is the hierarchy of the RadioButton class:

```
System.Object
   System.Web.UI.Control
         System.Web.UI.WebControls.WebControl
               System.Web.UI.WebControls.CheckBox
                     System.Web.UI.WebControls.RadioButton
```

This class is also inherited from the CheckBox class. Noteworthy public property of the RadioButton class is listed in Table 17.30:

Table 17.30: Noteworthy Public Properties of the RadioButton Class

GroupName	Obtains the name of the group to which radio button belongs

Working with **RadioButton** Controls

When you add a RadioButton control on a Web page (Default.aspx), it adds the following code to the source code of the Web page:

```
<asp:RadioButton ID="RadioButton1" runat="server" />
```

Let's create an application named RadioButtonVBExample. You can find the code of RadioButtonVBExample application in the Code\ASP.NET\Chapter 17\RadioButtonVBExample folder on the CD. In this example, we have placed five RadioButton controls on the Web page showing car names. The item attached to the RadioButton control appears in the Label control whenever you select it. Listing 17.32 shows the Default.aspx page for the RadioButton controls:

Listing 17.32: Showing the Code for the Default.aspx Page

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb"
  Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>RadioButton Control Example</title>
<link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <form id="form" runat="server">
        <div id="header">
        </div>
        <div id="sidebar">
              <div id="nav">
                     
              </div>
        </div>
        <div id="content">
              <div class ="itemContent">
              <asp:Label ID="Label1" runat="server" Font-Bold="True" Font-Size="Medium"
              Font-Underline="True" Text="RadioButton Control Example"></asp:Label>
              <br />
              <br />
              <asp:Label ID="Label2" runat="server" Font-Size="Small" ForeColor="Red"
              Text="Select the car you want to buy:"></asp:Label>
              <br />
              <asp:RadioButton ID="RadioButton1" runat="server" AutoPostBack="True"
              Font-Size="Small" ForeColor="#0000CC"
              Text="Maruti Zen" />
              <br />
```

```
                <asp:RadioButton ID="RadioButton2" runat="server" AutoPostBack="True"
                Font-Size="Small" ForeColor="#0000CC"
                Text="Toyota Corolla" />
                <br />
                <asp:RadioButton ID="RadioButton3" runat="server" AutoPostBack="True"
                Font-Size="Small" ForeColor="#0000CC"
                Text="Maruti Esteem" />
                <br />
                <asp:RadioButton ID="RadioButton4" runat="server" AutoPostBack="True"
                Font-Size="Small" ForeColor="#0000CC"
                Text="Honda City" />
                <br />
                <asp:RadioButton ID="RadioButton5" runat="server" AutoPostBack="True"
                Font-Size="Small" ForeColor="#0000CC"
                Text="Hundai Santro" />
                <br />
                <br />
                <asp:Label ID="Label3" runat="server" Font-Size="Small"
                  ForeColor="#FF6600"></asp:Label>
                <div id="footer">
                        <p class="left">
                        All content copyright &copy; Kogent Solutions Inc.</p>
                </div>
                </div>
                </div>
        </div>
    </form>
</body>
</html>
```

Now, add the code in the code-behind file of the `Default.aspx` page for the `RadioButton controls`, as shown in Listing 17.33:

Listing 17.33: Showing the Code for the Code-Behind File of the `Default.aspx` Page

```
Partial Class _Default
    Inherits System.Web.UI.Page
    Protected Sub RadioButton1_CheckedChanged(ByVal sender As Object, ByVal e As
      System.EventArgs) Handles RadioButton1.CheckedChanged
        Label3.Text = "<font color=black>You have selected: </font>" + RadioButton1.Text
        RadioButton2.Checked = False
        RadioButton3.Checked = False
        RadioButton4.Checked = False
        RadioButton5.Checked = False
    End Sub
    Protected Sub RadioButton2_CheckedChanged(ByVal sender As Object, ByVal e As
      System.EventArgs) Handles RadioButton2.CheckedChanged
        Label3.Text = "<font color=black>You have selected: </font>" + RadioButton2.Text
        RadioButton1.Checked = False
        RadioButton3.Checked = False
        RadioButton4.Checked = False
        RadioButton5.Checked = False
    End Sub
    Protected Sub RadioButton3_CheckedChanged(ByVal sender As Object, ByVal e As
      System.EventArgs) Handles RadioButton3.CheckedChanged
        Label3.Text = "<font color=black>You have selected: </font>" + RadioButton3.Text
        RadioButton2.Checked = False
        RadioButton1.Checked = False
        RadioButton4.Checked = False
        RadioButton5.Checked = False
    End Sub
```

```
    Protected Sub RadioButton4_CheckedChanged(ByVal sender As Object, ByVal e As
      System.EventArgs) Handles RadioButton4.CheckedChanged
        Label3.Text = "<font color=black>You have selected: </font>" + RadioButton4.Text
        RadioButton2.Checked = False
        RadioButton3.Checked = False
        RadioButton1.Checked = False
        RadioButton5.Checked = False
    End Sub
    Protected Sub RadioButton5_CheckedChanged(ByVal sender As Object, ByVal e As
      System.EventArgs) Handles RadioButton5.CheckedChanged
        Label3.Text = "<font color=black>You have selected: </font>" + RadioButton5.Text
        RadioButton2.Checked = False
        RadioButton3.Checked = False
        RadioButton4.Checked = False
        RadioButton1.Checked = False
    End Sub
End Class
```

Now, run the application by pressing the F5 key and select the radio button. The output is as shown in Figure 17.31:

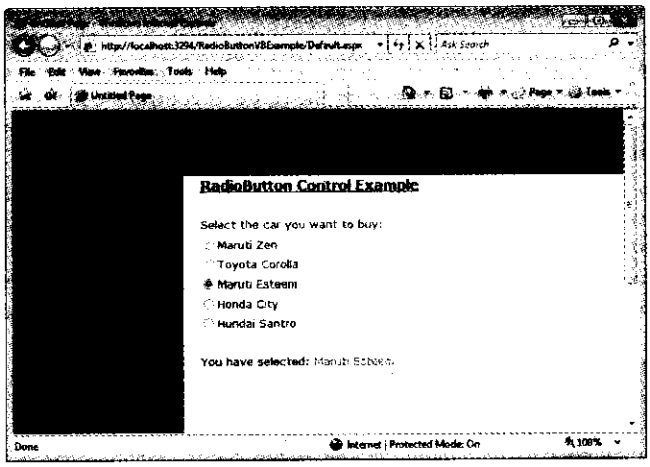


Figure 17.31: Selecting RadioButton Control

The **Table** Control

The `Table` control helps create a table. This control is useful when you want to present data in a tabular format. This control exists within the `System.Web.UI.WebControls` namespace. Web designers often use tables for the right placement of elements in Web pages. You can create a table at design-time or runtime, depending on the requirement of the project. Here is the hierarchy of this class:

```
System.Object
  System.Web.UI.Control
    System.Web.UI.WebControls.WebControl
      System.Web.UI.WebControls.Table
```

Noteworthy public properties of the `Table` class are listed in Table 17.31:

Table 17.31: Noteworthy Public Properties of the Table Class

`BackImageUrl`	Obtains or sets the URL of the background image to display at the back of the `Table` control
`Caption`	Obtains or sets the text to render in an HTML caption element in a `Table` control

Table 17.31: Noteworthy Public Properties of the Table Class	
CaptionAlign	Obtains or sets the horizontal or vertical positioning of the HTML caption element in a Table control
CellPadding	Obtains or sets the amount of space between the data of a cell and the cell's border
CellSpacing	Obtains or sets the amount of space between the cells
GridLines	Obtains or sets the grid line style to display in the Table control
HorizontalAlign	Obtains or sets the horizontal alignment of the Table control on the Web page
Rows	Retrieves the collection of rows in the Table control

Working with Table Controls

When you add a Table control on a Web page (Default.aspx), it adds the following code to the source code of the Web page:

```
<asp:Table ID="Table1" runat="server"></asp:Table>
```

Let's create an application named TableVBExample. You can find the code of TableVBExample application in the Code\ASP.NET\Chapter 17\TableVBExample folder on the CD. In this example, we have placed one Table control on the Web page showing student details on the Button control's click event. The item attached to RadioButtonList control appears in Label control whenever, you select it. Perform the following steps to work with the Table control:

1. Add table rows to the Table control with the help of TableRow Collection Editor dialog box by selecting the Table control in the design-mode and clicking the ellipsis (...) button in front of its Rows property in the Properties window.
2. For adding new table rows to the Table control, click the Add button in the TableRow Collection Editor dialog box. Figure 17.32 displays the TableRow Collection Editor dialog box with four table rows added to it:

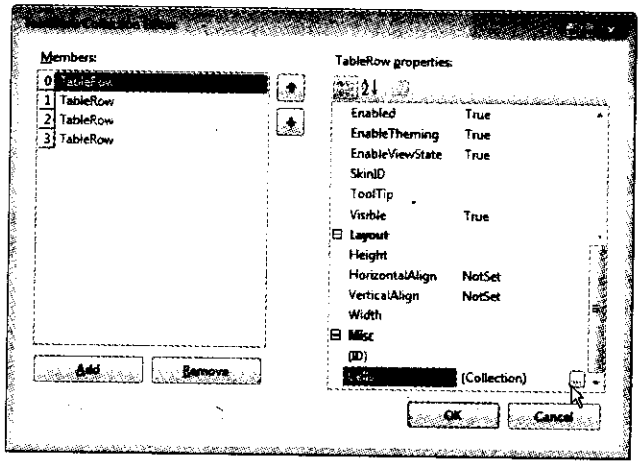


Figure 17.32: TableRow Collection Editor dialog box

3. To add table cells to a table row, click the ellipsis (...) button in front of its Cells property on the right side of the TableRow Collection Editor dialog box.

This opens the TableCell Collection Editor dialog box, as shown in Figure 17.33:

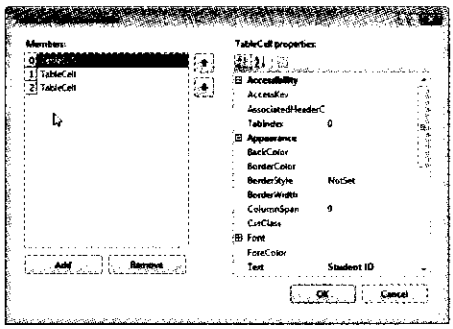


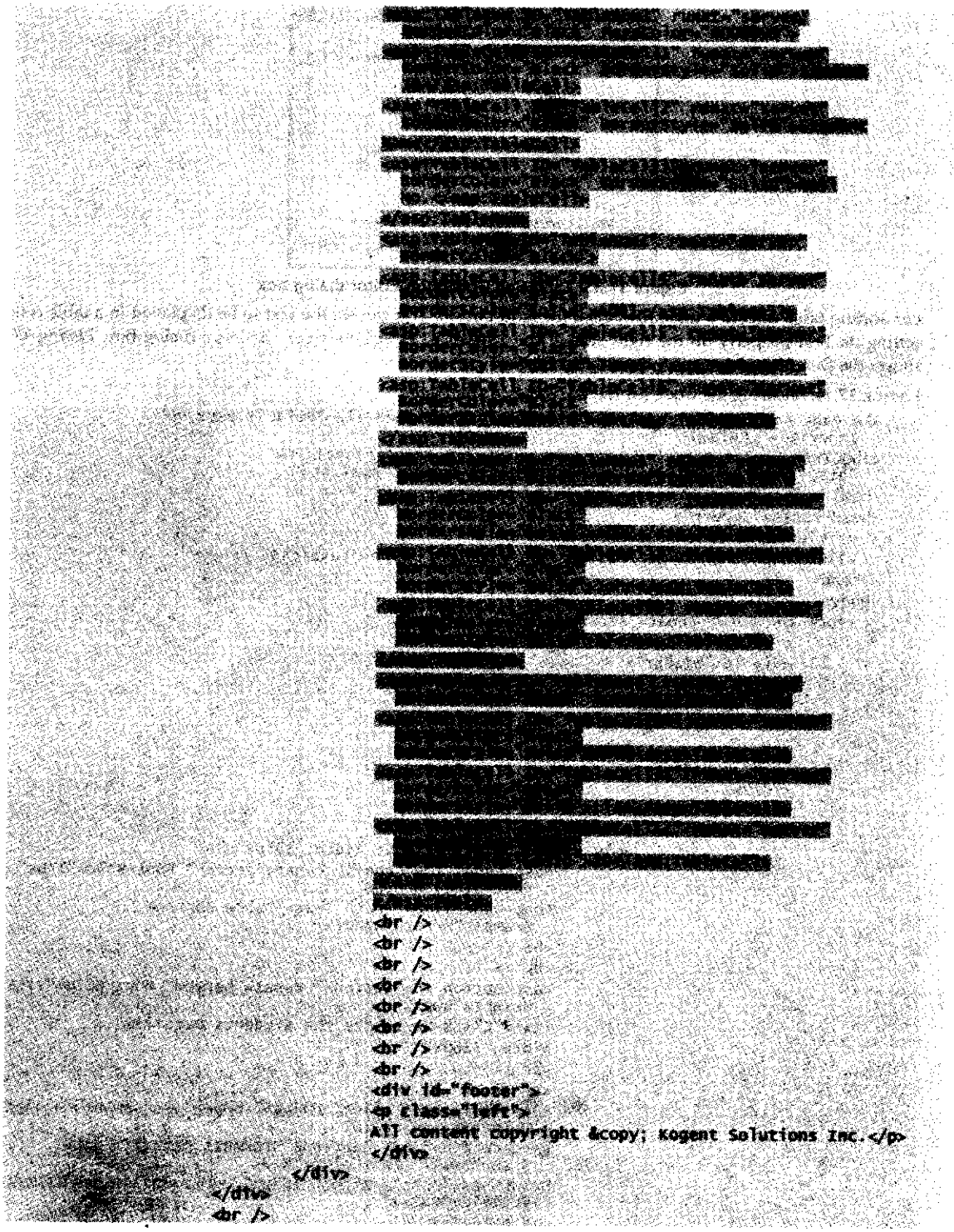
Figure 17.33: TableCell Collection Editor dialog box

For adding table cells to a table row, click the Add button. You can set the text to be displayed in a table cell by setting its Text property on the right side of the `TableCell Collection Editor` dialog box. Listing 17.34 shows the Default.aspx page for the Table controls:

Listing 17.34: Showing the Code for the `Default.aspx` Page

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb"
  Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>Table Control Example</title>
  <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <form id="Form" runat="server">

      <div id="header">
      </div>
      <div id="sidebar">
          <div id="nav">
               
          </div>
      </div>
      <div id="content">
          <div class ="itemContent" >
              <div style="height: 114px; width: 335px">
                  <asp:Label ID="Label1" runat="server" Font-Bold="True"
                    Font-Size="Medium"
                  Font-Underline="True" Text="Table Control
                    Example"></asp:Label>
                  <br />
                  <br />
                  <asp:Button ID="Button1" runat="server" Font-Bold="True"
                    Height="60px"
                  Text="Click Me to See the Students Detail!!"
                  Width="316px" />
                  <br />
                  <br />
```



```
<br />
<br />
<br />
<br />
<br />
<br />
<br />
<br />
<div id="footer">
<p class="left">
All content copyright &copy; Kogent Solutions Inc.</p>
</div>
```

```
                <br />
            </div>
        </form>
    </body>
    </html>
```

The highlighted part in the preceding Listing 17.34 shows how to insert the data into the Table control.

Now, add the code in the code-behind file of the `Default.aspx` page for the Table controls, as shown in Listing 17.35:

Listing 17.35: Showing the Code for the Code-Behind File of the `Default.aspx Page`

```
Partial Class _Default
  Inherits System.Web.UI.Page
  Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
   Handles Button1.Click
        Table1.Visible = True
  End Sub
End Class
```

Run the application by pressing the F5 key an click the button. The output is as shown in Figure 17.34:

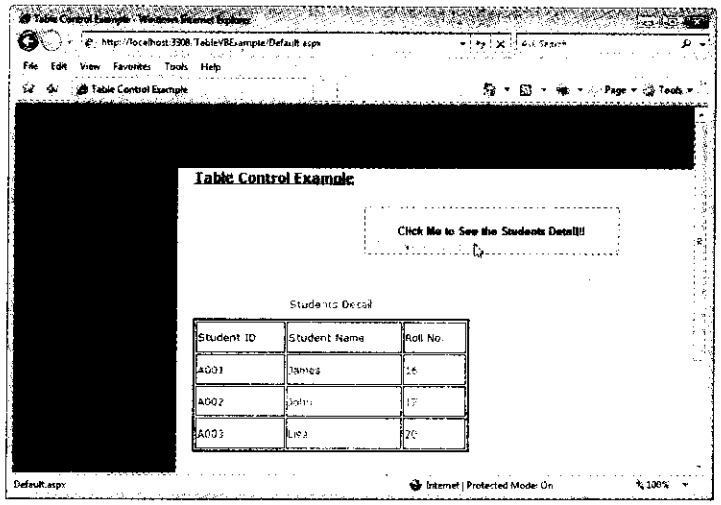


Figure 17.34: Showing Table Control Along with Records on Button Click Event

The **Wizard** Control

The `Wizard` control provides navigation and a User Interface (UI) to collect related data across multiple steps. It also allows you to implement linear or non-linear navigation through the steps such as skipping unnecessary steps or returning to a previously completed step to change some value. The appearance of the `Wizard` control is fully customizable by using templates, skins, and style settings; for example, you can use the `HeaderTemplate`, `SideBarTemplate`, `StartNavigationTemplate`, `FinishNavigationTemplate`, and `StepNavigationTemplate` properties to customize the interface of the `Wizard` control. Here is the inheritance hierarchy of the `Wizard` class:

```
System.Object
  System.Web.UI.Control
      System.Web.UI.WebControls.WebControl
          System.Web.UI.WebControls.CompositeControl
              System.Web.UI.WebControls.Wizard
```

Noteworthy public properties of the `Wizard` class are listed in Table 17.32:

Table 17.32: Noteworthy Public Properties of the Wizard Class

Property	Description
ActiveStep	Obtains the step in the WizardSteps collection, which is currently displayed to the end-user
ActiveStepIndex	Obtains or sets the index value of the current WizardStepBase object
CancelButtonImageUrl	Obtains or sets the URL of the image displayed for the Cancel button
CancelButtonStyle	Obtains a reference to a collection of style properties that describe the appearance of the Cancel button
CancelButtonText	Obtains or sets the text caption that is displayed for the Cancel button
CancelButtonType	Obtains or sets the type of button that is rendered as the Cancel button
CancelDestinationPageUrl	Obtains or sets the URL that the end user is directed to when they click the Cancel button
CellPadding	Obtains or sets the amount of space between the data of the cell and the cell border
CellSpacing	Obtains or sets the space between the cells
DisplayCancelButton	Obtains or sets a Boolean value showing whether to display a Cancel button
DisplaySideBar	Obtains or sets a Boolean value showing whether to display the sidebar area on the Wizard control
FinishCompleteButtonImageUrl	Obtains or sets the URL of the image, which is displayed for the Finish button
FinishCompleteButtonStyle	Obtains a reference to a Style object that describes the settings for the Finish button
FinishCompleteButtonText	Obtains or sets the text caption that is displayed for the Finish button
FinishCompleteButtonType	Obtains or sets the type of button that is rendered as the Finish button
FinishDestinationPageUrl	Obtains or sets the URL that the end user is redirected to when they click the Finish button
FinishNavigationTemplate	Obtains or sets the template that is used to display the navigation area on the Finish step
FinishPreviousButtonImageUrl	Obtains or sets the URL of the image that is displayed for the Previous button on the Finish step
FinishPreviousButtonStyle	Obtains a reference to a Style object that describes the settings for the Previous button on the Finish step
FinishPreviousButtonText	Obtains or sets the text caption, which is displayed for the Previous button on the Finish step
FinishPreviousButtonType	Obtains or sets the type of button, which is rendered as the Previous button on the Finish step
HeaderStyle	Obtains a reference to a Style object, which defines the settings for the header area on the control
HeaderTemplate	Obtains or sets the template, which is used to display the header area on the control
HeaderText	Obtains or sets the text caption, which is displayed for the header area on the control
NavigationButtonStyle	Obtains a reference to a Style object, which describes the settings for the buttons in the navigation area on the control

Table 17.32: Noteworthy Public Properties of the Wizard Class

Property	Description
`NavigationStyle`	Obtains a reference to a `Style` object, which describes the settings for the navigation area on the control
`SideBarButtonStyle`	Obtains a reference to a `Style` object, which describes the settings for the buttons on the sidebar
`SideBarStyle`	Obtains a reference to a `Style` object, which describes the settings for the sidebar area on the control
`SideBarTemplate`	Obtains or sets the template, which is used to display the sidebar area on the control
`SkipLinkText`	Obtains or sets a value, which is used to render alternate text that notifies screen readers to skip the content in the sidebar area
`StartNavigationTemplate`	Obtains or sets the template, which is used to display the navigation area on the `Start` step of the `Wizard` control
`StartNextButtonImageUrl`	Obtains or sets the URL of the image, which is displayed for the Next button on the `Start` step
`StartNextButtonStyle`	Obtains a reference to a `Style` object, which describes the settings for the Next button on the `Start` step
`StartNextButtonText`	Obtains or sets the text caption, which is displayed for the Next button on the `Start` step
`StartNextButtonType`	Obtains or sets the type of button, which is rendered as the Next button on the `Start` step
`StepNavigationTemplate`	Obtains or sets the template, which is used to display the navigation area on any `WizardStepBase`-derived objects other than the `Start`, the `Finish`, or the `Complete` step
`StepNextButtonImageUrl`	Obtains or sets the URL of the image, which is displayed for the Next button
`StepNextButtonStyle`	Obtains a reference to the `Style` object, which describes the settings for the Next button
`StepNextButtonText`	Obtains or sets the text caption, which is displayed for the Next button
`StepNextButtonType`	Obtains or sets the type of button, which is rendered as the Next button
`StepPreviousButtonImageUrl`	Obtains or sets the URL of the image, which is displayed for the Previous button
`StepPreviousButtonStyle`	Obtains a reference to a `Style` object, which describes the settings for the Previous button
`StepPreviousButtonText`	Obtains or sets the text caption which is displayed for the Previous button
`StepPreviousButtonType`	Obtains or sets the type of button which is rendered as the Previous button
`StepStyle`	Obtains a reference to a `Style` object which describes the settings for the `WizardStep` objects
`WizardSteps`	Obtains a collection containing all the `WizardStepBase` objects that are described for the control

Noteworthy public methods of the `Wizard` class are listed in Table 17.33:

Table 17.33: Noteworthy Public Methods of the Wizard Class

Method	Description
`GetHistory`	Obtains a collection of WizardStepBase objects that have been accessed
`GetStepType`	Obtains the WizardStepType value for the specified WizardStepBase object
`MoveTo`	Sets the specified WizardStepBase-derived object as the value for the ActiveStep property of the Wizard control

Noteworthy public events of the `Wizard` class are listed in Table 17.34:

Table 17.34: Noteworthy Public Events of the Wizard Class

Event	Description
`ActiveStepChanged`	Raises when the user switches to a new step in the control
`CancelButtonClick`	Raises when the Cancel button is clicked
`FinishButtonClick`	Raises when the Finish button is clicked
`NextButtonClick`	Raises when the Next button is clicked
`PreviousButtonClick`	Raises when the Previous button is clicked
`SideBarButtonClick`	Raises when a button in the sidebar area is clicked

Working with **Wizard** Controls

When you add a `Wizard` control on a Web page (`Default.aspx`), it adds the following code to the source code of the Web page:

```
<asp:Wizard ID="Wizard1" runat="server">
  <WizardSteps>
        <asp:WizardStep runat="server" title="Step 1">
        </asp:WizardStep>
        <asp:WizardStep runat="server" title="Step 2">
        </asp:WizardStep>
  </WizardSteps>
</asp:Wizard>
```

Let's create an application named WizardVBExample. You can find the code of WizardVBExample application in the Code\ASP.NET\Chapter 17\WizardVBExample folder on the CD. First, you have to add one Wizard control to the Web page. In this example, we have also placed two `Label` and two `DropDownList` controls within the Wizard control. To add a Wizard control with the pre-defined style formats, select the AutoFormat option from the Smart Tag. This will display the `AutoFormat` dialog box, as shown in Figure 17.35:

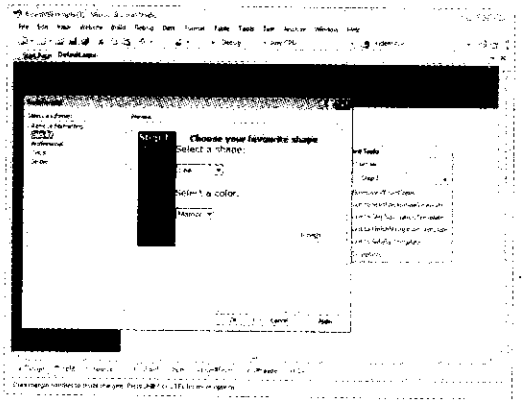


Figure 17.35: Auto Format Dialog Box

Here, the user is prompted to select a shape and color to fill the shape from the `DropDownList` controls, as shown in Figure 17.36:

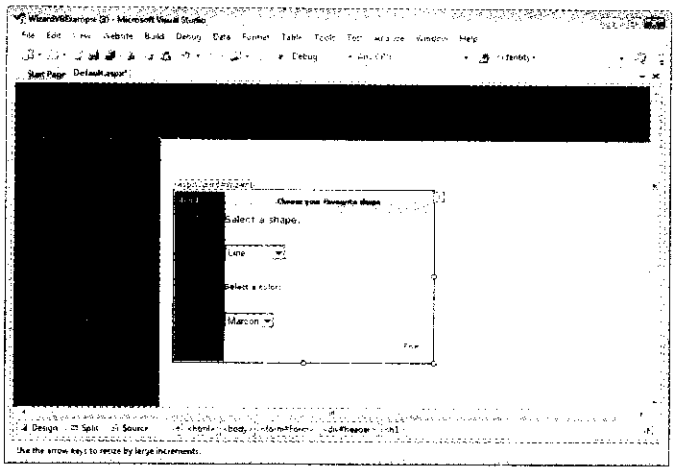


Figure 17.36: Wizard Control at Design-Time After Selecting Color and Shape

Listing 17.36 shows the Default.aspx page for the Wizard controls:

Listing 17.36: Showing the Code for the `Default.aspx` Page

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb"
  Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>Wizard Control Example</title>
<link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <form id="Form" runat="server">

        <div id="header">
        </div>
        <div id="sidebar">
              <div id="nav">
                     
              </div>
        </div>
        <div id="content">
              <div class ="itemContent" >
              <asp:Label ID="Label1" runat="server" Font-Bold="True" Font-Size="Medium"
              Font-Underline="True" Text="Wizard Control Example"></asp:Label>
              <br />
              <br />
              <br />
              <asp:Wizard ID="Wizard1" runat="server" ActiveStepIndex="0"
              Height="163px" Width="356px" HeaderText ="Choose your favourite
              shape" BorderColor ="#FFDFAD" BackColor="#FFFBD6"
              BorderWidth="1px" Font-Names="Verdana" Font-Size="Small" >
              <WizardSteps>
                    <asp:WizardStep ID="WizardStep1" runat="server" Title="Step 1" >
                          <asp:Label ID="Label2" runat="server" Text="Select a
                            shape:" Font-Size="Small"></asp:Label><br />
```

```
                                <br />
                                <asp:DropDownList ID="DropDownList1" runat="server">
                                        <asp:ListItem>Line</asp:ListItem>
                                        <asp:ListItem>Ellipse</asp:ListItem>
                                        <asp:ListItem>Rectangle</asp:ListItem>
                                </asp:DropDownList><br />
                                <br />
                                <asp:Label ID="Label3" runat="server"
                                Text="Select a color:"></asp:Label><br />
                                <br />
                                <asp:DropDownList ID="DropDownList2"
                                runat="server">
                                        <asp:ListItem>Maroon</asp:ListItem>
                                        <asp:ListItem>Purple</asp:ListItem>
                                        <asp:ListItem>Blue</asp:ListItem>
                                        <asp:ListItem>Pink</asp:ListItem>
                                </asp:DropDownList>
                        </asp:WizardStep>
                        <asp:WizardStep ID="WizardStep2" runat="server" Title="Step 2"
                        AllowReturn="False" StepType="Complete">
                        </asp:WizardStep>
                </WizardSteps>
                <SideBarStyle BackColor="#990000" Font-Size="0.9em"
                VerticalAlign="Top" />
                <NavigationButtonStyle BackColor="White"
                BorderColor="#CC9966" BorderStyle="Solid"
                BorderWidth="1px" Font-Names="Verdana" Font-Size="0.8em"
                ForeColor="#990000" />
                <SideBarButtonStyle ForeColor="White" />
                <HeaderStyle BackColor="#FFCC66" BorderColor="#FFFBD6"
                BorderStyle="Solid" BorderWidth="2px"
                Font-Bold="True" Font-Size="0.9em" ForeColor="#333333"
                HorizontalAlign="Center" />
                </asp:Wizard>
                <div id="footer">
                        <p class="left">
                        All content copyright &copy; Kogent Solutions Inc.</p>
                </div>
                </div>
                </div>
                <div>

        </div>
    </form>
</body>
</html>
```

Now, add the code in the code-behind file of the `Default.aspx` page for the Wizard controls, as shown in Listing 17.37:

Listing 17.37: Showing the code for the Code-Behind File of the `Default.aspx` Page

```
Imports System.Drawing
Imports System.Drawing.Imaging
Imports System.Drawing.Text
Imports System.Drawing.Drawing2D
Partial Class _Default
  Inherits System.Web.UI.Page
  Private Sub DrawShape()
        Dim BitMapObject As System.Drawing.Bitmap
        Dim g As System.Drawing.Graphics
```

```
        BitMapObject = New Bitmap(300, 300)
        g = System.Drawing.Graphics.FromImage(BitMapObject)
        g.SmoothingMode = Drawing2D.SmoothingMode.HighQuality
        g.TextRenderingHint = TextRenderingHint.AntiAlias
        g.Clear(Color.Silver)
        If DropDownList1.SelectedItem.Text = "Line" Then
                CallDrawLine(g)
        ElseIf DropDownList1.SelectedItem.Text = "Ellipse" Then
                CallDrawEllipse(g)
        ElseIf DropDownList1.SelectedItem.Text = "Rectangle" Then
                CallDrawRectangle(g)
        End If
        Response.ContentType = "image/jpeg"
        BitMapObject.Save(Response.OutputStream, ImageFormat.Jpeg)
        BitMapObject.Dispose()
        g.Dispose()
        Response.End()
    End Sub
    Private Sub CallDrawLine(ByVal g As System.Drawing.Graphics)
        Dim cl As Color
        cl = ColorTranslator.FromHtml(DropDownList2.SelectedItem.Text)
        Dim RPen As New Pen(cl, 8)
        Dim p1 As New Point(95, 55)
        Dim p2 As New Point(196, 55)
        Dim joinPoints As Point() = {p1, p2}
        g.DrawLines(RPen, joinPoints)
    End Sub
    Private Sub CallDrawRectangle(ByVal g As System.Drawing.Graphics)
        Dim RPen As New Pen(Color.Yellow, 15)
        Dim rect As New Rectangle(95, 109, 100, 90)
        g.DrawRectangle(RPen, rect)
        Dim cl As Color
        cl = ColorTranslator.FromHtml(DropDownList2.SelectedItem.Text)
        g.FillRectangle(New SolidBrush(cl), rect)
    End Sub
    Private Sub CallDrawEllipse(ByVal g As System.Drawing.Graphics)
        Dim RPen As New Pen(Color.Yellow, 15)
        Dim rectF As New RectangleF(100, 109, 100, 90)
        g.DrawEllipse(RPen, rectF)
        Dim cl As Color
        cl = ColorTranslator.FromHtml(DropDownList2.SelectedItem.Text)
        g.FillEllipse(New SolidBrush(cl), rectF)
    End Sub
    Protected Sub Wizard1_FinishButtonClick(ByVal sender As Object, ByVal e As
     System.Web.UI.WebControls.WizardNavigationEventArgs) Handles Wizard1.FinishButtonClick
        DrawShape()
    End Sub
End Class
```

NOTE

Notice that in the code-behind file (Default.aspx.vb) we have added four additional namespaces, such as System.Drawing, System.Drawing.Imaging, System.Drawing.Text, and System.Drawing.Drawing2D, which is mandatory, for performing operations related to drawing.

When you run this application by pressing the F5 key, you will be prompted to select the shape and color of your choice, as shown in Figure 17.37:

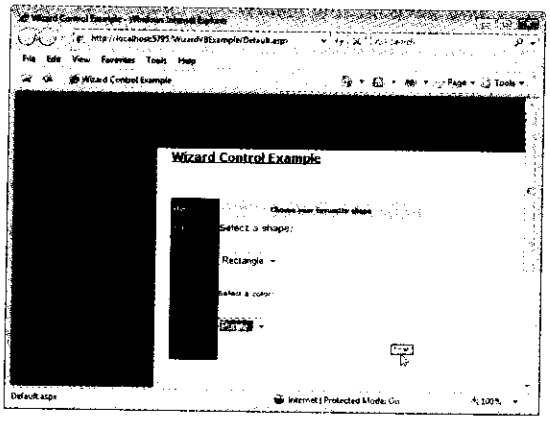


Figure 17.37: First Step of Wizard

On clicking the Finish button, the DrawShape method, which further calls the CallDrawRectangle method. For example, in the output (Figure 17.38), if you select the shape as Rectangle and color as Purple, the following method is called from Listing 17.37:

```
Private Sub CallDrawRectangle(ByVal g As System.Drawing.Graphics)
    Dim RPen As New Pen(Color.Yellow, 15)
    Dim rect As New Rectangle(95, 109, 100, 90)
    g.DrawRectangle(RPen, rect)
    Dim cl As Color
    cl = ColorTranslator.FromHtml(DropDownList2.SelectedItem.Text)
    g.FillRectangle(New SolidBrush(cl), rect)
End Sub
```

In the preceding code, the `CallDrawRectangle` method is called when the user selects Rectangle from the `DropDownList` control. Using the object of the Pen class, you can specify the color and the width of the selected shape's border. The `DrawRectangle` method of the `Graphics` class takes two parameters to draw a rectangle. One is the object of the `Pen` class and the other is the object of the `Rectangle` class with the coordinates specified to draw the rectangle. The `FillRectangle` method of the `Graphics` class then takes two objects as parameters to fill the rectangle with the color; one is the object of the `Brush` class and the other is the object of the `Rectangle` class. Finally, you can see the output, as shown in Figure 17.38:

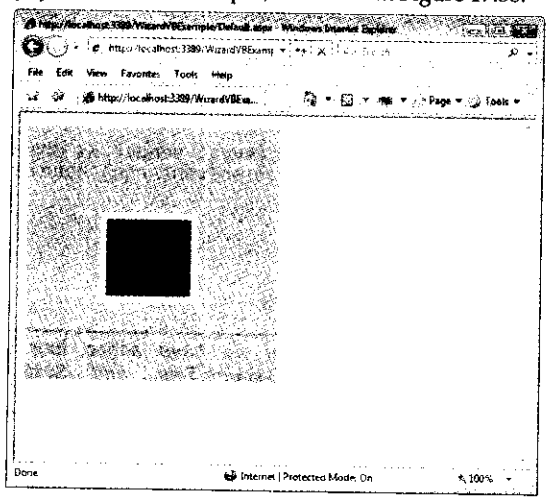


Figure 17.38: Dynamically Generated Shape

User Controls and Custom Controls

There are situations, when you need some specific functionality in a control, which is not provided by the built-in ASP.NET Web server controls. In such situations, you can create user controls and custom controls according to the requirement. User controls are basically containers into which you can put markup and Web server controls. Later on you can use the user control as a unit in various ASP.NET pages whereas, custom control is a class, which you write, and it derives from Control or WebControl classes. Creating user controls is comparatively easier task to create than custom controls, because user controls are similar to an ASP.NET Web page (.aspx file), with both a user interface page and code, while custom control does not have any user interface.

Working with User Controls

Let's create a Web application named UserControlVB. You can find the code of UserControlVB application in the Code\ASP.NET\Chapter 17\UserControlVB folder on the CD. In the Web application, we will learn how to create a user control, in which user will enter the name in textbox and it displays in label on button click event and after that using that user control in an ASP.NET Web page (.aspx). To create a user control, perform the following steps:

1. Create ASP.NET Web application named as UserControlVB in Visual Studio 2008 by selecting ASP.NET Web Application template from the New Project window.
2. Right-click the application name in the Solution Explorer, and select Web User Control template from the Add New Item dialog box.

It creates WebUserControl.ascx file by default. Listing 17.38 shows the code for the `WebUserControl.ascx` page:

Listing 17.38: Showing the Code for the `WebUserControl.ascx` Page

```
<%@ Control Language="VB" AutoEventWireup="false" CodeFile="WebUserControl.ascx.vb"
  Inherits="WebUserControl" %>
<asp:Label ID="Label2" runat="server" Text="Enter Your Name:"></asp:Label>
  

<asp:TextBox ID="TextBox1" runat="server" Width="149px"></asp:TextBox>
<br />
<br />

<asp:Button ID="Button1" runat="server" Text="Submit" />
<br />
<br />
<br />
<asp:Label ID="Label1" runat="server"></asp:Label>
```

Now, add the code in the code-behind file of the `WebUserControl.ascx` page, as shown in Listing 17.39:

Listing 17.39: Showing the code for the `WebUserControl.ascx.vb` Page

```
Public Partial Class WebUserControl
  Inherits System.Web.UI.UserControl
  Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles
      Me.Load
  Label1.Text = "Welcome " & TextBox1.Text
  End Sub
End Class
```

3. Now, build the Web application by pressing F5 key and then create a Web page to host the control. In our application, we already have a Default.aspx page to host the control.

NOTE

If you make any changes to the user control, then re-compile the application again.

To use the control, add a @Register directive to the Web page that specified the tag prefix we use, the tag name and the respective location (Src attribute) of the user control's page. Listing 17.40 shows the code for the Default.aspx page:

Listing 17.40: Showing the Code for the `Default.aspx` Page

```
<%@ Page Language="vb" AutoEventWireup="false" CodeBehind="Default.aspx.vb"
  Inherits="UserControlVB._Default" %>
<%@ Register TagPrefix="TP" TagName="Upload" Src="WebUserControl.ascx" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
  <title></title>
</head>
<body>
  <form id="form1" runat="server">
        <div>
              <TP:Upload ID="Upload1" runat="server" />
        </div>
  </form>
</body>
</html>
```

In our case, the tag prefix is TP (you can change it as well). Now set the Default.aspx as start page.

4. Now, run the application by pressing the F5 key and click the Submit button. The output is as shown in Figure 17.39:

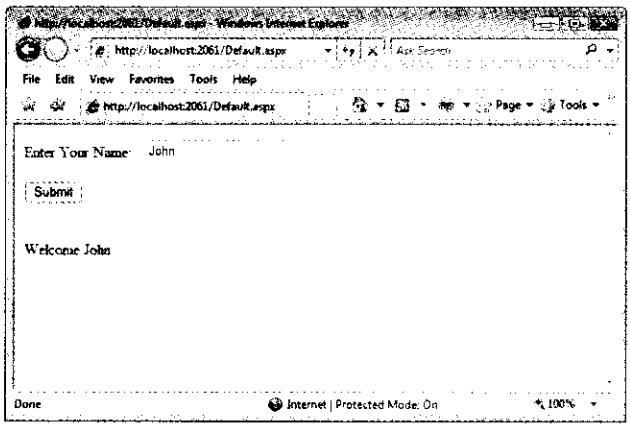


Figure 17.39: User Control on Default.aspx Page

Working with Custom Controls

Let's create a Web application named CustomControlVB. You can find the code of CustomControlVB application in the Code\ASP.NET\Chapter 17\CustomControlVB folder on the CD. In the Web application, we will learn how to create a custom control, which will display server name, current date, and time on the page load event. To create a custom control, perform the following steps:

1. Create ASP.NET Web application named as CustomControlVB in Visual Studio 2008 by selecting ASP.NET Web Application template from the New Project window.
2. Add a class file to the application and rename it to MyControl.vb. Listing 17.41 shows the code for the MyControl.vb class:

Listing 17.41: Showing the Code for the `MyControl.vb` Class

```
Imports System

Imports System.ComponentModel
Imports System.Security.Permissions
Imports System.Web
Imports System.Web.UI
Imports System.Web.UI.WebControls
Namespace CustomControlSample.AspNetControl

  Public Class MyControl
        Inherits WebControl
        Public Overridable Property Text() As String
                Get
                        Dim comp As String = CStr(ViewState("CompName"))
                        If comp Is Nothing Then comp = String.Empty
                        Return comp
                End Get
                        Set(ByVal result As String)
                        ViewState("CompName") = result
                End Set
        End Property
        Protected Overrides Sub RenderContents( _
                ByVal writer As HtmlTextWriter)
                writer.WriteEncodedText(Text)
                If Context IsNot Nothing Then
                        Dim comp As String = Context.User.Identity.Name
                        If (comp IsNot Nothing) AndAlso (comp <> String.Empty) Then
                                Dim split() As String = comp.Split("\".ToCharArray)
                                If (split(0) <> String.Empty) Then
                                        writer.Write(split(0))
                                End If
                        End If
                End If
                writer.Write("!" + "<BR>")
                writer.Write("Today's date and current time is " +
                System.DateTime.Now.ToString() + "</font>")
        End Sub

  End Class

End Namespace
```

In the preceding listing, the custom control we are creating, MyControl is similar to the standard Label control. The MyControl class derives from WebControl class and the Context.User.Identity.Name property fetches your computer name. The WebControl class defines a Text property that allows you to provide a text string, for example, if you set " Computer Name: "as the value of the Text property, MyControl custom control results" Computer Name: Computer Name!" or "Computer Name: " based on whether the computer name is present or not. In addition to the computer name we are also displaying current date and time after a line break.

3. Now add the following code in the <controls> section in the Web.config file:
   `<add tagPrefix="aspSample" namespace="CustomControlSample.AspNetControl"/>`
4. Now, build the Web application by pressing F5 key and then create a Web page to host the control. In our application, we already have a Default.aspx page to host the control.

You can see the custom control (MyControl) added to the Toolbox at design-time of the Default.aspx page as shown in Figure 17.40:

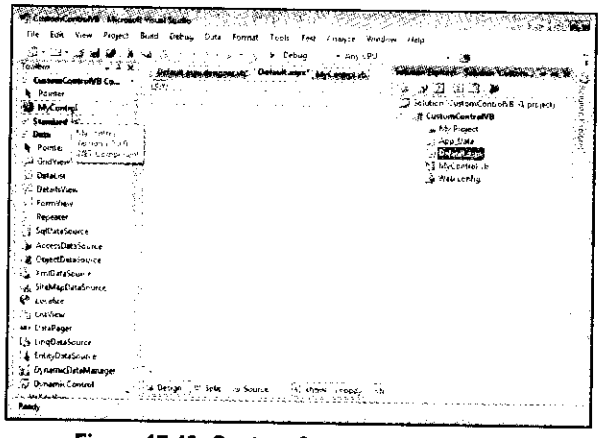


Figure 17.40: Custom Control in the Toolbox

5. Drag and drop the custom control to the Web page, the @Register directive automatically added to the Web page that specifies the assembly, namespace, and the tag prefix we use. You can add more attributes to customize the label. Listing 17.42 shows the code for the Default.aspx page:

Listing 17.42: Showing the Code for the `Default.aspx` Page

```
<%@ Page Language="vb" AutoEventWireup="false" CodeBehind="Default.aspx.vb"
  Inherits="CustomControlVB._Default" %>
<%@ Register Assembly="CustomControlVB"
  Namespace="CustomControlVB.CustomControlSample.AspNetControl" TagPrefix="cc1" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
  <title></title>
</head>
<body>
  <form id="form1" runat="server">
      <div>
      </div>
      <cc1:MyControl ID="MyControl1" Text="Computer Name: " ForeColor="Blue"
          BackColor="Black" runat="server" />
  </form>
</body>
</html>
```

In our case, the tag prefix is cc1 (you can change it as well). Now set the Default.aspx as start page.

4. Now, run the application by pressing the F5 key. The output is as shown in Figure 17.41:

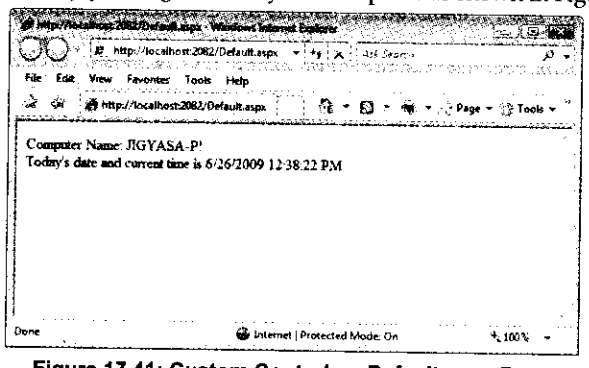


Figure 17.41: Custom Control on Default.aspx Page

Summary

In this chapter, you learned about various Standard Web controls, such as the Button control, TextBox control, Label control, Literal control, Image, FileUpload, and Wizard control. These controls are mostly used for designing user interfaces for ASP.NET Web applications. This chapter also focuses on customizing these controls, and setting their common properties. We have also discussed the implementation of these controls. Chapter also focused on the user controls and custom controls along with their implementations.

In the next chapter, we explore few more controls that appear under the Navigation tab of the ToolBox and are used for navigational purpose; such as `Treeview`, `Menu`, and `SiteMapPath`.

Quick Revise

Q1. Which class gets inherited when an ASP.NET server control is added to a Web form?

1. **Control**
2. **WebControl**
3. **System**
4. **ASP**

Ans: WebControl

Q2. What does the "EnableViewState" property do? Why do we want it On or Off?

Ans: The EnableViewState property enables the ViewState property on the page. It is set to On to allow the page to save the users input between postback requests of a Web page; that is, between the Request and corresponding Response objects. When this property is set to Off, the page does not store the users input during postback.

Q3. What is the difference between a Label control and a Literal control?

Ans: The final html code of a Label control has an HTML tag whereas final html code of a Literal control contains only text, which is not surrounded by any HTML tag.

Q4. What is the difference between user control and custom control?

Ans: User controls are basically containers into which you can put markup and Web server controls, whereas, custom control is a class, which you write, and it derives from Control or WebControl classes. User controls are similar to an ASP.NET Web page (.aspx file), with both a user interface page and code, while custom control does not have any user interface.

Q5. What is the purpose of PostBackUrl property of Button control?

Ans: It gets or sets the URL of the page to post when the Button control is clicked.

Q6. Which is the default TextMode property of the TextBox control?

1. **Text**
2. **SingleLine**
3. **MultiLine**
4. **Password**

Ans: SingleLine

Q7. What is the main purpose of using PlaceHolder control?

Ans: The PlaceHolder control is used as a container to store server controls that are added to the Web page at runtime. The PlaceHolder control does not produce any visible output and is used only as a container for other controls on the Web page.

Q8. **The ..... control is used to store a value that needs to persist across posts to the server.**

1. **FileUpload**
2. **HiddenField**
3. **PlaceHolder**
4. **Button**

Ans: HiddenField

Q9. What is the difference between AlternateText and DescriptionUrl properties of the Image control?

Ans: The AlternateText property gets or sets the alternate text displayed in the Image control when the image is not available whereas DescriptionUrl property gets or sets the location to a detailed description for the image.

Q10. What is the purpose of MergeStyle method?

Ans: The MergeStyle method copies any non-blank elements of the specified style to the Web control, but not overwrites any existing style elements of the control.