Index

Coalesce

Chapter 1

Introduction

What is ASP.NET?.

ASP.NET is a programming framework built on the common language runtime that can be used on a server to build powerful Web applications. ASP.NET offers several important advantages over previous Web development models:

- **Enhanced Performance.** ASP.NET is compiled common language runtime code running on the server. Unlike its interpreted predecessors, ASP.NET can take advantage of early binding, just-in-time compilation, native optimization, and caching services right out of the box. This amounts to dramatically better performance before you ever write a line of code.

- **World-Class Tool Support.** The ASP.NET framework is complemented by a rich toolbox and designer in the Visual Studio integrated development environment. WYSIWYG editing, drag-and-drop server controls, and automatic deployment are just a few of the features this powerful tool provides.

- **Power and Flexibility.** Because ASP.NET is based on the common language runtime, the power and flexibility of that entire platform is available to Web application developers. The .NET Framework class library, Messaging, and Data Access solutions are all seamlessly accessible from the Web. ASP.NET is also language-independent, so you can choose the language that best applies to your application or partition your application across many languages. Further, common language runtime interoperability guarantees that your existing investment in COM-based development is preserved when migrating to ASP.NET.

- **Simplicity.** ASP.NET makes it easy to perform common tasks, from simple form submission and client authentication to deployment and site configuration. For example, the ASP.NET page framework allows you to build user interfaces that cleanly separate application logic from presentation code and to handle events in a simple, Visual Basic - like forms processing model. Additionally, the common language runtime simplifies development, with managed code services such as automatic reference counting and garbage collection.

- **Manageability.** ASP.NET employs a text-based, hierarchical configuration system, which simplifies applying settings to your server environment and Web applications. Because configuration information is stored as plain text, new settings may be applied without the aid of local administration tools. This "zero local administration" philosophy extends to deploying ASP.NET Framework applications as well. An ASP.NET Framework application is deployed to a server simply by copying the necessary files to the server. No server restart is required, even to deploy or replace running compiled code.

- **Scalability and Availability.** ASP.NET has been designed with scalability in mind, with features specifically tailored to improve performance in clustered and multiprocessor environments. Further, processes are closely monitored and managed by the ASP.NET runtime, so that if one misbehaves (leaks, deadlocks), a new process can be created in its place, which helps keep your application constantly available to handle requests.

- **Customizability and Extensibility.** ASP.NET delivers a well-factored architecture that allows developers to "plug-in" their code at the appropriate level. In fact, it is possible to extend or replace any subcomponent of the ASP.NET runtime with your own custom-written component. Implementing custom authentication or state services has never been easier.

- **Security.** With built in Windows authentication and per-application configuration, you can be assured that your applications are secure.

Language Support

The Microsoft .NET Platform currently offers built-in support for three languages: C#, Visual Basic, and JScript.

The exercises and code samples demonstrate how to use C#, Visual Basic, and JScript to build .NET applications. For information regarding the syntax of the other languages, refer to the complete documentation for the .NET Framework SDK.

The following table is provided to help you understand the code samples in this tutorial as well as the differences between the three languages:

Variable Declarations

```
     Dim x As Integer
Dim s As String
Dim s1, s2 As String
Dim o 'Implicitly Object
Dim obj As New Object()
Public name As String
```

Statements

```
Response.Write("foo")
```

Comments

```
' This is a comment
' This is
' a multiline
```

Coalesce

```
' comment
```

Accessing Indexed Properties

```
Dim s, value As String
s = Request.QueryString("Name")
value = Request.Cookies("Key").Value
'Note that default non-indexed properties
'must be explicitly named in VB
```

Declaring Indexed Properties

```
' Default Indexed Property
Public Default ReadOnly Property DefaultProperty(Name As String)
As String
    Get
        Return CStr(lookuptable(Name))
    End Get
End Property
```

Declaring Simple Properties

```
Public Property Name As String
  Get
    ...
    Return ...
  End Get
  Set
    ... = Value
  End Set
End Property
```

Declare and Use an Enumeration

```
' Declare the Enumeration
Public Enum MessageSize
    Small = 0
    Medium = 1
    Large = 2
End Enum
' Create a Field or Property
Public MsgSize As MessageSize
' Assign to the property using the Enumeration values
MsgSize = small
```

Enumerating a Collection

```
Dim S As String
For Each S In Coll
 ...
Next
```

Coalesce

Declare and Use Methods

```
' Declare a void return function
Sub VoidFunction()
 ...
End Sub
' Declare a function that returns a value
Function StringFunction() As String
 ...
    Return CStr(val)
End Function
' Declare a function that takes and returns values
Function ParmFunction(a As String, b As String) As String
 ...
    Return CStr(A & B)
End Function
' Use the Functions
VoidFunction()
Dim s1 As String = StringFunction()
Dim s2 As String = ParmFunction("Hello", "World!")
```

Custom Attributes

```
' Stand-alone attribute
<STAThread>
' Attribute with parameters
<Obsolete("Obsolete message here")>
' Attribute with named parameters
<Obsolete("Obsolete message here", true)>
```

Arrays

```
    Dim a(2) As String
    a(0) = "1"
    a(1) = "2"
    a(2) = "3"
    Dim a2(2,2) As String
    a(0,0) = "1"
    a(1,0) = "2"
    a(2,0) = "3"
```

Initialization

```
Dim s As String = "Hello World"
Dim i As Integer = 1
Dim a() As Double = { 3.00, 4.00, 5.00 }
```

If Statements

```
If Not (Request.QueryString = Nothing)
  ...
End If
```

Coalesce

Case Statements

```
Select Case FirstName
  Case "John"
    ...
  Case "Paul"
    ...
  Case "Ringo"
    ...
  Case Else
    ...
End Select
```

For Loops

```
  Dim I As Integer
  For I = 0 To 2
    a(I) = "test"
  Next
```

While Loops

```
Dim I As Integer
I = 0
Do While I < 3
  Console.WriteLine(I.ToString())
  I += 1
Loop
```

Exception Handling

```
Try
    ' Code that throws exceptions
Catch E As OverflowException
    ' Catch a specific exception
Catch E As Exception
    ' Catch the generic exceptions
Finally
    ' Execute some cleanup code
End Try
```

String Concatenation

```
' Using Strings
Dim s1, s2 As String
s2 = "hello"
s2 &= " world"
s1 = s2 & " !!!"
' Using StringBuilder class for performance
Dim s3 As New StringBuilder()
s3.Append("hello")
s3.Append(" world")
s3.Append(" !!!")
```

Coalesce

Event Handler Delegates

```
Sub MyButton_Click(Sender As Object,
                   E As EventArgs)
...
End Sub
```

Declare Events

```
' Create a public event
Public Event MyEvent(Sender as Object, E as
EventArgs)
' Create a method for firing the event
Protected Sub OnMyEvent(E As EventArgs)
    RaiseEvent MyEvent(Me, E)
End Sub
```

Add or Remove Event Handlers to Events

```
AddHandler Control.Change, AddressOf Me.ChangeEventHandler
RemoveHandler Control.Change, AddressOf
Me.ChangeEventHandler
```

Casting

```
Dim obj As MyObject
Dim iObj As IMyObject
obj = Session("Some Value")
iObj = CType(obj, IMyObject)
```

Conversion

```
Dim i As Integer
Dim s As String
Dim d As Double
i = 3
s = i.ToString()
d = CDbl(s)
' See also CDbl(...), CStr(...), ...
```

Class Definition with Inheritance

```
Imports System
Namespace MySpace
  Public Class Foo : Inherits Bar
    Dim x As Integer
    Public Sub New()
      MyBase.New()
      x = 4
    End Sub
    Public Sub Add(x As Integer)
      Me.x = Me.x + x
    End Sub
    Overrides Public Function
GetNum() As Integer
       Return x
    End Function
  End Class
End Namespace
' vbc /out:libraryvb.dll /t:library
' library.vb
```

Implementing an Interface

```
Public Class MyClass : Implements IEnumerable
 ...
    Function IEnumerable_GetEnumerator() As IEnumerator
_
        Implements IEnumerable.GetEnumerator
         ...
    End Function
End Class
```

Class Definition with a Main Method

```
Imports System
Public Class ConsoleVB
  Public Sub New()
    MyBase.New()
    Console.WriteLine("Object Created")
  End Sub
  Public Shared Sub Main()
    Console.WriteLine("Hello World")
    Dim cvb As New ConsoleVB
  End Sub
End Class
' vbc /out:consolevb.exe /t:exe console.vb
```

Standard Module

```
Imports System
Public Module ConsoleVB
  Public Sub Main()
    Console.WriteLine("Hello World")
  End Sub
End Module
' vbc /out:consolevb.exe /t:exe console.vb
```

Web Forms Syntax Reference

ASP.NET Web Forms Syntax Elements

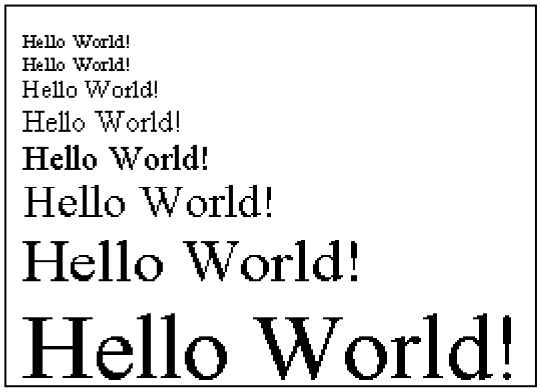
An ASP.NET Web Forms page is a declarative text file with an .aspx file name extension. In addition to static content, you can use eight distinct syntax markup elements. This section of the QuickStart reviews each of these syntax elements and provides examples demonstrating their use.

Rendering Code Syntax: <% %> and <%= %>

Code rendering blocks are denoted with <% ... %> elements, allow you to custom-control content emission, and execute during the render phase of Web Forms page execution. The following example demonstrates how you can use them to loop over HTML content.

```
    <% For I=0 To 7 %>
    <font size="<%=i%>"> Hello World! </font> <br>
<% Next %>
```

Figure 2.1 Reference1.aspx



Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!

Code for Figure 2.1 Reference1.aspx

```
<%@ Page Language="VB" %>
<html>
  <body>
    <% Dim I As Integer
      For I = 0 To 7 %>
      <font size="<%=I%>"> Hello World! </font> <br>
    <% Next %>
  </body>
</html>
```

Coalesce

Code enclosed by <% ... %> is just executed, while expressions that include an equal sign, <%= ... %>, are evaluated and the result is emitted as content. Therefore <%="Hello World" %> renders the same thing as the C# code <% Response.Write("Hello World"); %>.

Note: For languages that use marks to end or separate statements (for example, the semicolon (;) in C#), it is important to place those marks correctly depending on how your code should be rendered.

C# code

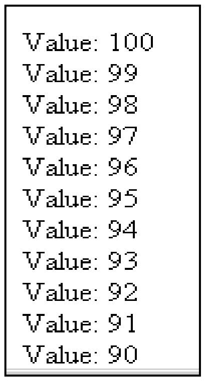
<% Response.Write("Hello World"); %>	A semicolon is necessary to end the statement.
<%="Hello World"; %>	Wrong: Would result in "Response.Write("Hello World";);".
<%="Hello World" %>	A semicolon is not necessary.

Declaration Code Syntax: <script runat="server">

Code declaration blocks define member variables and methods that will be compiled into the generated **Page** class. These blocks can be used to author page and navigation logic. The following example demonstrates how a **Subtract** method can be declared within a **<script runat="server">** block, and then invoked from the page.

```
<script language="VB" runat=server>
Function Subtract(num1 As Integer, num2 As Integer) As Integer
  Return(num1 - num2)
End Function
</script>
<%
  ...
  number = subtract(number, 1)
  ...
%>
```

Figure 2.2 Refrenece 2.aspx



```
Value: 100
Value: 99
Value: 98
Value: 97
Value: 96
Value: 95
Value: 94
Value: 93
Value: 92
Value: 91
Value: 90
```

Code for Figure 2.2 Refrenece 2.aspx

```
<html>
  <script language="VB" runat=server>
     Function Subtract(Num1 As Integer, Num2 As Integer) As Integer
       Return Num1-Num2
     End Function
  </script>
  <body>
    <%
      Dim Number As Integer = 100
      Do While Number > 0
        Response.Write("Value: " & Number & "<br>")
        Number = Subtract(Number, 1)
      Loop
    %>
  </body>
</html>
```

Important: Unlike ASP -- where functions could be declared within <% %> blocks -- all functions and global page variables must be declared in a **<script runat=server>** tag. Functions declared within <% %> blocks will now generate a syntax compile error.

ASP.NET Server Control Syntax

Custom ASP.NET server controls enable page developers to dynamically generate HTML user interface (UI) and respond to client requests. They are represented within a file using a declarative, tag-based syntax. These tags are distinguished from other tags because they contain a **"runat=server"** attribute. The following example demonstrates how an **<asp:label runat="server">** server control can be used within an ASP.NET page. This control corresponds to the **Label** class in the **System.Web.UI.WebControls** namespace, which is included by default.

Coalesce

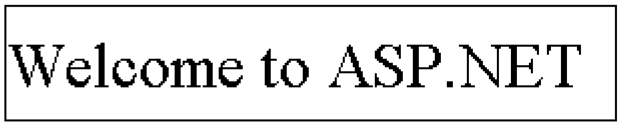
By adding a tag with the ID "Message", an instance of **Label** is created at run time:

<asp:label id="Message" font-size=24 runat="server"/>
The control can then be accessed using the same name. The following line sets the **Text** property of the control.

Figure 2.3 Refrenece 3.aspx

```
Message.Text = "Welcome to ASP.NET"
```

Code for Figure 2.3 Refrenece 3.aspx


Welcome to ASP.NET

Code for Refrenece 3.aspx

```
<html>
  <script language="VB" runat=server>
     Sub Page_Load(Sender As Object, E As EventArgs)
        Message.Text = "Welcome to ASP.NET"
     End Sub
  </script>
  <body>
    <asp:label id="Message" font-size=24 runat=server/>
  </body>
</html>
```

ASP.NET HTML Server Control Syntax

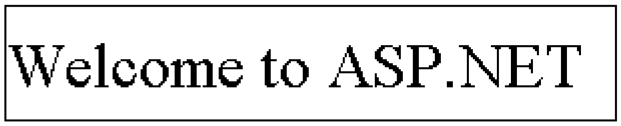
HTML server controls enable page developers to programmatically manipulate HTML elements within a page. An HTML server control tag is distinguished from client HTML elements by means of a **"runat=server"** attribute. The following example demonstrates how an HTML **<span runat=server>** server control can be used within an ASP.NET page.

As with other server controls, the methods and properties are accessible programmatically, as shown in the following example.

```
<script language="VB" runat="server">
 Sub Page_Load(sender As Object, e As EventArgs)
```

```
  Message.InnerHtml = "Welcome to ASP.NET"
 End Sub
</script>
...
<span id="Message" style="font-size:24" runat="server"/>
```

Figure 2.3 Refrenece 3.aspx


Welcome to ASP.NET

Code for Figure 2.4 Refrenece 4.aspx

```
<html>
  <script language="VB" runat=server>
     Sub Page_Load(Sender As Object, E As EventArgs)
        Message.InnerHtml = "Welcome to ASP.NET"
     End Sub
  </script>
  <body>
    <span id="Message" style="font-size:24" runat=server/>
  </body>
</html>
```

Data Binding Syntax: <%# %>

The data binding support built into ASP.NET enables page developers to hierarchically bind control properties to data container values. Code located within a <%# %> code block is only executed when the **DataBind** method of its parent control container is invoked. The following example demonstrates how to use the data binding syntax within an **<asp:datalist runat=server>** control.

Within the datalist, the template for one item is specified. The content of the item template is specified using a data binding expression and the Container.DataItem refers to the data source used by the datalist MyList.

```
<asp:datalist id="MyList" runat=server>  <ItemTemplate>    Here is a value:
<%# Container.DataItem %>  </ItemTemplate></asp:datalist>
```

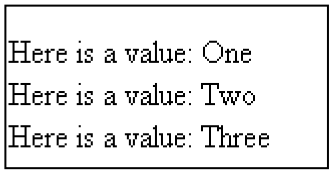
In this case the data source of the MyList control is set programmatically, and then DataBind() is called.

Code for Refrenece 4.aspx

```
Sub Page_Load(sender As Object, e As EventArgs)
 Dim items As New ArrayList()
 items.Add("One")
 items.Add("Two")
 items.Add("Three")
 MyList.DataSource = items
 MyList.DataBind()
End Sub
```

Calling the **DataBind** method of a control causes a recursive tree walk from that control on down in the tree; the **DataBinding** event is raised on each server control in that hierarchy, and data binding expressions on the control are evaluated accordingly. So, if the **DataBind** method of the page is called, then every data binding expression within the page will be called.

Figure 2.2 Refrenece 2.aspx



```
Here is a value: One
Here is a value: Two
Here is a value: Three
```

Code for Figure 2.2 Refrenece 2.aspx

```
 <html>
   <script language="VB" runat=server>
     Sub Page_Load(Sender As Object, E As EventArgs)
       Dim Items As New ArrayList
       Items.Add("One")
       Items.Add("Two")
       Items.Add("Three")
       MyList.DataSource = Items
       MyList.DataBind()
     End Sub
  </script>
 <body>
     <asp:datalist id="MyList" runat=server>
       <ItemTemplate>
         Here is a value: <%# Container.DataItem %>
       </ItemTemplate>
     </asp:datalist>
   </body>
 </html>
```

Object Tag Syntax: <object runat="server" />

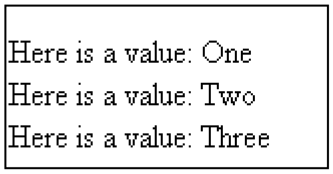
Object tags enable page developers to declare and create instances of variables using a declarative, tag-based syntax. The following example demonstrates how the object tag can be used to create an instance of an **ArrayList** class.

```
<object id="items" class="System.Collections.ArrayList" runat="server"/>
```

The object will be created automatically at run time and can then be accessed through the ID "items".

```
Sub Page_Load(sender As Object, e As EventArgs)
  items.Add("One")
  items.Add("Two")
  items.Add("Three")
        ...
     End Sub
```

Figure 2.3 Refrenece 3.aspx



```
Here is a value: One
Here is a value: Two
Here is a value: Three
```

Code for Figure 2.3 Refrenece 3.aspx

```
<html>
  <object id="Items" class="System.Collections.ArrayList" runat=server/>
  <script language="VB" runat=server>
    Sub Page_Load(Sender As Object, E As EventArgs)
      Items.Add("One")
      Items.Add("Two")
      Items.Add("Three")
      MyList.DataSource = Items
      MyList.DataBind()
    End Sub
  </script>
  <body>
    <asp:datalist id="MyList" runat=server>
      <ItemTemplate>
        Here is a value: <%# Container.DataItem %>
      </ItemTemplate>
    </asp:datalist>
  </body>
</html>
```

Server-Side Comment Syntax: <%-- Comment --%>

Server-side comments enable page developers to prevent server code (including server controls) and static content from executing or rendering. The following sample demonstrates how to block content from executing and being sent down to a client. Note that everything between <%-- and --%> is filtered out and only visible in the original server file, even though it contains other ASP.NET directives.

```
<%--
 <asp:calendar id="MyCal" runat=server/>
  <% For I=0 To 44 %>
         Hello World <br>
  <% Next %>
--%>
```

Code for   Refrenece 4.aspx

```
<html>
  <body>
    The below content has been hidden from browser clients using a server-side
comment
    (view the .aspx source to see what we mean :-)
      <%--
      <asp:calendar id="MyCal" runat=server/>
      <% For I = 0  To 44 %>
         Hello World <br>
      <% Next %>
    --%>
  </body>
</html>
```

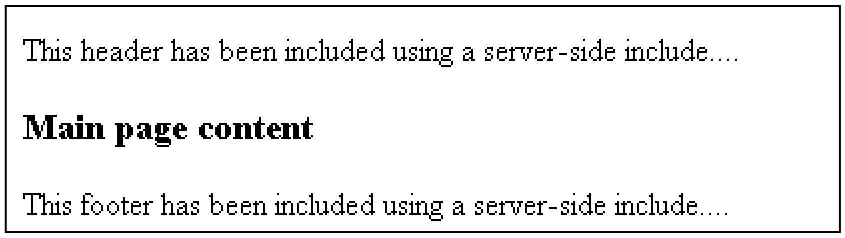
Server-Side Include Syntax: <-- #Include File="Locaton.inc" -->

Server-side #Includes enable developers to insert the raw contents of a specified file anywhere within an ASP.NET page. The following sample demonstrates how to insert a custom header and footer within a page.

```
<!-- #Include File="Header.inc" -->

...

<!-- #Include File="Footer.inc" -->
```

Coalesce

Figure 2.3 Refrenece 5.aspx


This header has been included using a server-side include....

Main page content

This footer has been included using a server-side include....

Code for Figure 2.3 Refrenece 5.aspx

```
<html>
  <body>
    <!-- #Include File="Header.inc" -->
        <p>
    <h3> Main page content </h3>
    <p>
    <!-- #Include File="Footer.inc" -->
  </body>

</html>
```

Chapter 3

Web Forms Controls

System.Web.UI.HtmlControls

HTML server controls are HTML elements exposed to the server so you can program against them. HTML server controls expose an object model that maps very closely to the HTML elements that they render.

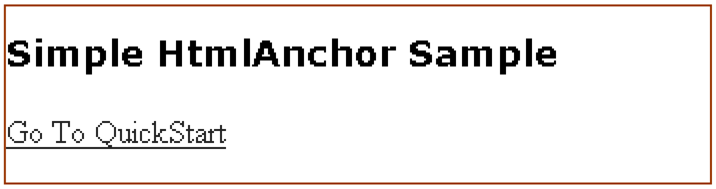
HtmlAnchor	HtmlButton	HtmlForm	HtmlGenericControl
HtmlImage	HtmlInputButton (Button)	HtmlInputButton (Reset)	HtmlInputButton (Submit)
HtmlInputCheckBox	HtmlInputFile	HtmlInputHidden	HtmlInputImage
HtmlInputRadioButton	HtmlInputText (Password)	HtmlInputText (Text)	HtmlSelect
HtmlTable	HtmlTableCell	HtmlTableRow	HtmlTextArea

HtmlAnchor

Working with HtmlAnchor

The following sample illustrates using the **HtmlAnchor** control (**<a>**). **HtmlAnchor** is used to navigate from the client page to another page.

Figure 3.1 HtmlAnchor1.aspx



Simple HtmlAnchor Sample

Go To QuickStart

Code for Figure 3.1 HtmlAnchor1.aspx

```
<html>
<script language="VB" runat="server">
  Sub Page_Load(sender As Object, e As EventArgs)
    anchor1.HRef = "/QuickStart"
  End Sub
</script>
<body>
  <h3><font face="Verdana">Simple HtmlAnchor Sample</font></h3>
  <form runat=server>

    <p>
    <a id=anchor1 runat="server">
      Go To QuickStart
    </a>
  </form>
</body>
```

```
</html>
```

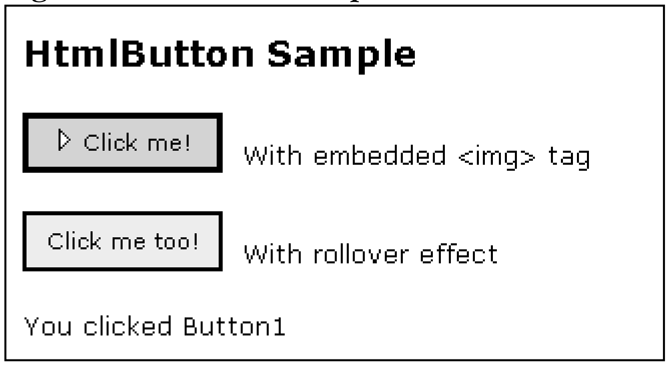
HtmlButton

Working with HtmlButton

The **HtmlButton** control renders as an HTML 4.0 **<button>**. This differs from <input type="button"> in that it enables Web developers to create rich user interface form buttons that can be composed from embedded HTML elements (and even other ASP.NET server controls).

The following sample illustrates using the **HtmlButton** control.

Figure 3.2 HtmlButton.aspx



You clicked Button1

Code for HtmlButton.aspx

```
<html>
<head>
   <script language="VB" runat="server">
     Sub Button1_OnClick(sender As Object, e As EventArgs)
        Span1.InnerHtml="You clicked Button1"
     End Sub
    Sub Button2_OnClick(sender As Object, e As EventArgs)
        Span1.InnerHtml="You clicked Button2"
     End Sub
   </script>
</head>
<body>
   <h3><font face="Verdana">HtmlButton Sample</font></h3>
   <form runat=server>
   <font face="Verdana" size="-1">
      <p>
      <button id="Button1" onServerClick="Button1_OnClick" style="font: 8pt
verdana;background-color:lightgreen;border-color:black;height=30;width:100"
runat="server">
```

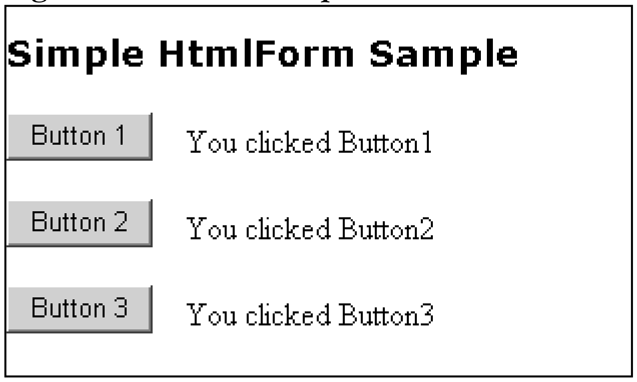
```
            <img src="/quickstart/aspplus/images/right4.gif"> Click me!
        </button>
        &nbsp;With embedded &lt;img&gt; tag
        <p>
        <button id=Button2 onServerClick="Button2_OnClick"
                style="font:          8pt          verdana;background-color:lightgreen;border-
color:black;height=30;width:100"
                onmouseover="this.style.backgroundColor='yellow'"
                onmouseout="this.style.backgroundColor='lightgreen'"
                runat="server">
            Click me too!
        </button>
        &nbsp;With rollover effect
        <p>
        <span id=Span1 runat=server />
    </font>
    </form>
</body>
</html>
```

HtmlForm

Working with HtmlForm

An **HtmlForm** control is required to process postback requests. A Web Forms page might only have one server side **<form>** tag; however, client forms (no **runat=server** attribute) can also postback to server-side logic as long as a server-side form is present on the page.

Figure 3.3 Htmlform.aspx


Simple HtmlForm Sample

Button 1   You clicked Button1

Button 2   You clicked Button2

Button 3   You clicked Button3

Code for Figure 3.3 Htmlform.aspx

```
<html>
<head>
   <script language="VB" runat="server">
      Sub Button1_OnClick(sender As Object, e As EventArgs)
         Span1.InnerHtml = "You clicked Button1"
      End Sub
      Sub Button2_OnClick(sender As Object, e As EventArgs)
         Span2.InnerHtml = "You clicked Button2"
      End Sub

      Sub Button3_OnClick(sender As Object, e As EventArgs)
         Span3.InnerHtml = "You clicked Button3"
      End Sub
   </script>
</head>
<body>
   <h3><font face="Verdana">Simple HtmlForm Sample</font></h3>
   <form id=ServerForm runat=server>
       <input value="Button 1" type="submit" id=Button1
        runat="server" onServerClick="Button1_OnClick">
       &nbsp;&nbsp;
       <span id=Span1 runat=server />
       <p>
       <input value="Button 2" type="submit" id=Button2
        runat="server" onServerClick="Button2_OnClick">
       &nbsp;&nbsp;
       <span id=Span2 runat=server />
       <p>
       <input value="Button 3" type="submit" id=Button3
        runat="server" onServerClick="Button3_OnClick">
       &nbsp;&nbsp;
       <span id=Span3 runat=server />
   </form>
</body>
</html>
```

HtmlGenericControl

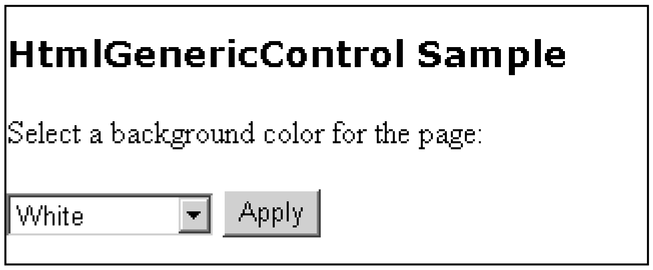
Working with HtmlGenericControl

The **HtmlGenericControl** provides an ASP.NET server control implementation for all unknown HTML server control tags not directly represented by a specific HTML server control (for example, **<span>**, **<div>**, **<body>**, and so on).

The following sample illustrates using the **HtmlGenericControl** control for the **<body>** tag.

Figure 3.4 HtmlGenericControl.aspx

Coalesce


HtmlGenericControl Sample
Select a background color for the page:
White
Apply

Code forFigure 3.3 Htmlform.aspx

```
<html>
<head>
   <script language="VB" runat="server">
   Sub SubmitBtn_Click(sender As Object, e As EventArgs)
      Body.Attributes("bgcolor") = ColorSelect.Value
   End Sub
   </script>
</head>
<body id=Body runat=server>
   <h3><font face="Verdana">HtmlGenericControl Sample</font></h3>
   <form runat=server>
     <p>
     Select a background color for the page: <p>
     <select id="ColorSelect" runat="server">
        <option>White</option>
        <option>Wheat</option>
        <option>Gainsboro</option>
        <option>LemonChiffon</option>
     </select>
     <input          type="submit"          runat="server"          Value="Apply"
OnServerClick="SubmitBtn_Click">
   </form>
</body>
</html>
```

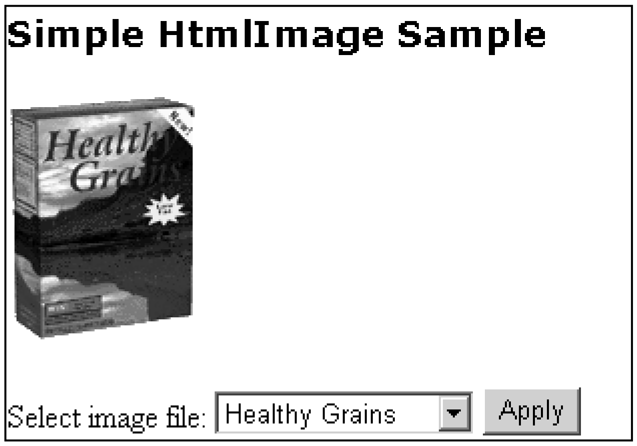
HtmlImage

Working with HtmlImage

An **HtmlImage** control renders the image file specified by its **Src** property in an HTML **<img>** tag.

The following sample illustrates using the **HtmlImage** control.

Figure 3.5 HtmlImage.aspx


Simple HtmlImage Sample

Select image file: Healthy Grains Apply

Code for Figure 3.5 HtmlImage.aspx

```
<html>
<head>
   <script language="VB" runat="server">
      Sub SubmitBtn_Click(sender As Object, e As EventArgs)
         Image1.Src="/quickstart/aspplus/images/" & Select1.Value
      End Sub
   </script>
</head>
<body>
   <h3><font face="Verdana">Simple   Sample</font></h3>
   <form runat=server>
      <img       ID="Image1"       src="/quickstart/aspplus/images/cereal1.gif"
runat="server"/>
      <p>
      Select image file:
      <select id="Select1" runat="server">
         <option Value="Cereal1.gif">Healthy Grains</option>
         <option Value="Cereal2.gif">Corn Flake Cereal</option>
         <option Value="Cereal3.gif">U.F.O.S</option>
         <option Value="Cereal4.gif">Oatey O's</option>
         <option Value="Cereal5.gif">Strike</option>
         <option Value="Cereal7.gif">Fruity Pops</option>
      </select>
      <input          type="submit"          runat="server"          Value="Apply"
OnServerClick="SubmitBtn_Click">
   </form>
</body>
</html>
```

Coalesce

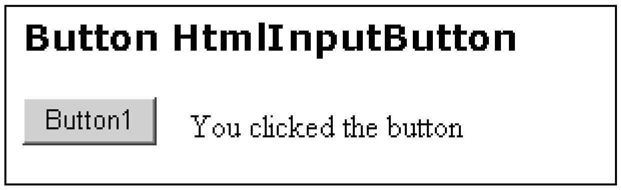
HtmlInputButton

Working with HtmlInputButton (Button)

The **HtmlInputButton** control (**<Input type=button>**) is similar in function to the <button> tag, except that it can target any browser.

The following sample illustrates using the **HtmlInputButton** control.

Figure 3.6 HtmlInputbutton1.aspx

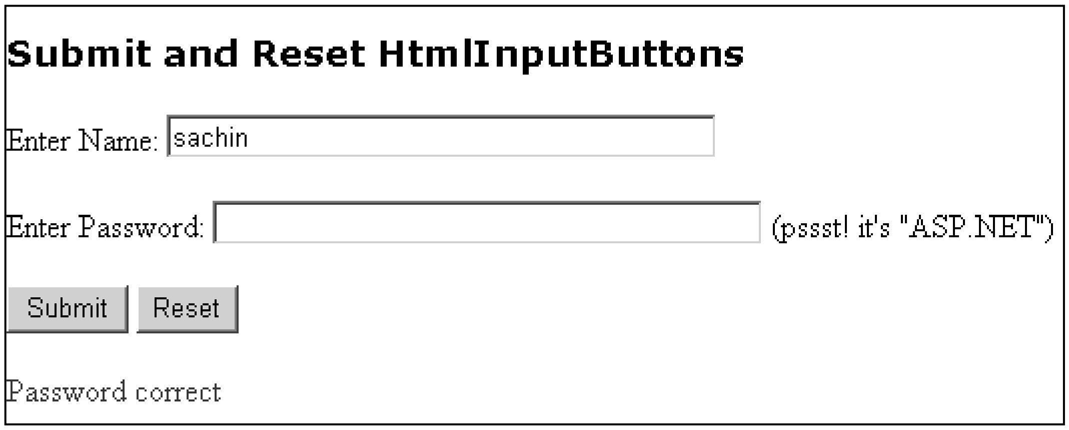


Submit and Reset HtmlInputButtons

The **HtmlInputButton** control also supports the **Reset** and **Submit** button types, which are used only with forms. **Submit** submits the form, whereas **Reset** restores all of the entry fields in a form to their initial values.

The following sample illustrates using **Submit** and **Reset HtmlInputButton** controls.

Figure 3.7 HtmlInputbutton2.aspx



Code for Figure 3.7 HtmlInputbutton2.aspx

Coalesce

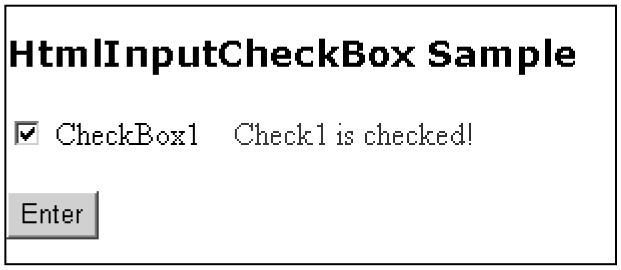
```
<html>
<head>
   <script language="VB" runat="server">
     Sub Button1_Click(sender As Object, e As EventArgs)
       Span1.InnerHtml = "You clicked the button"
     End Sub
   </script>
</head>
<body>
   <h3><font face="Verdana">Button HtmlInputButton</font></h3>
   <form runat=server>
      <p>
      <input type=button value="Button1" onServerClick="Button1_Click"
runat="server">
      &nbsp;&nbsp;
      <span id=Span1 runat=server />
   </form>
</body>
</html>
<html>
<head>
  <script language="VB" runat="server">
      Sub SubmitBtn_Click(sender As Object, e As EventArgs)
        If Password.Value = "ASP.NET" Then
           Span1.InnerHtml = "Password correct"
        Else
           Span1.InnerHtml="That password is not correct"
        End If
     End Sub
   </script>
</head>
<body>
   <h3><font face="Verdana">Submit and Reset
HtmlInputButtons</font></h3>
   <form runat=server>
      Enter Name: <input id="Name" type=text size=40 runat=server>
      <p>
      Enter Password: <input id="Password" type=password size=40
runat=server> (pssst! it's "ASP.NET")
      <p>
      <input type=submit value="Submit" OnServerClick="SubmitBtn_Click"
runat=server>
      <input type=reset runat=server>
      <p>
     <span id="Span1" style="color:red" runat=server></span>
   </form>
```

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

HtmlInputCheckBox

Working with HtmlInputCheckBox
The **HtmlInputCheckBox** control accepts Boolean (**true**/**false**) input. When selected, its **Checked** property is **true**. The following sample illustrates using the **HtmlInputCheckBox** control.

Figure 3.8 HtmlInputcheckbox.aspx


HtmlInputCheckBox Sample

☑ CheckBox1 Check1 is checked!

Code for Figure 3.8 HtmlInputcheckbox.aspx

```
<html>
<head>

   <script language="VB" runat="server">

     Sub Button1_Click(sender As Object, e As EventArgs)

       If Check1.Checked Then
          Span1.InnerHtml = "Check1 is checked!"
       Else
          Span1.InnerHtml = "Check1 is not checked!"
       End If
     End Sub

   </script>

</head>

<body>
```

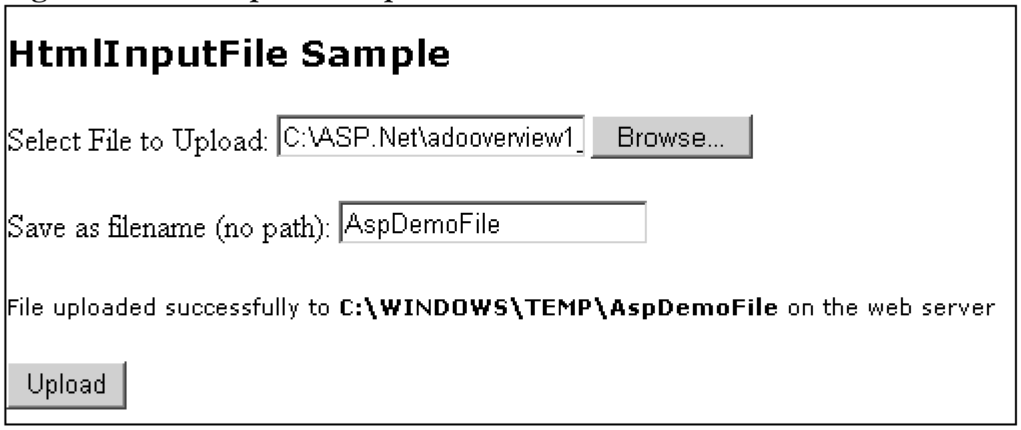
```
   <h3><font face="Verdana">HtmlInputCheckBox Sample</font></h3>

   <form runat=server>

      <input id="Check1" type=checkbox runat="server"> CheckBox1
&nbsp;&nbsp;

      <span id=Span1 style="color:red" runat=server />

      <p>

      <input type=button id="Button1" value="Enter"
OnServerClick="Button1_Click" runat=server>

   </form>

</body>
</html>
```

HtmlInputFile

Working with HtmlInputFile

An **HtmlInputFile** control handles uploading of binary or text files from a client browser to the server. File-upload works with all HTML 3.2 and later Web clients. Note that the **Enctype** attribute on the **<form>** tag must be set to **"multipart/form-data"**.

Figure 3.9 HtmlInputFile.aspx


HtmlInputFile Sample

Select File to Upload: C:\ASP.Net\adooverview1_

Save as filename (no path): AspDemoFile

File uploaded successfully to **C:\WINDOWS\TEMP\AspDemoFile** on the web server

Code for Figure 3.9 HtmlInputFile.aspx

Coalesce

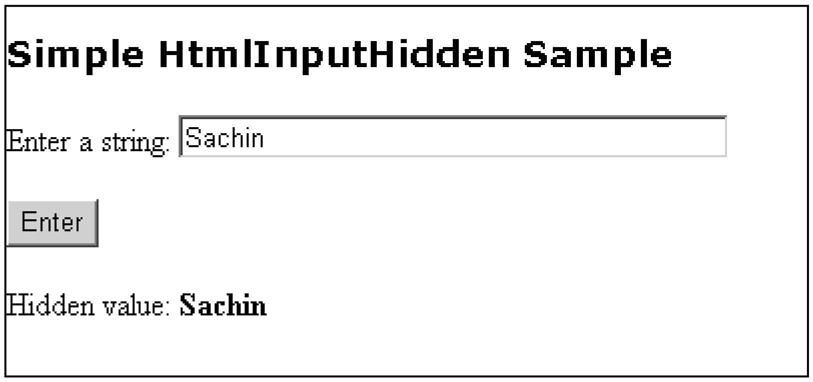
```
<%@ Import Namespace="System.IO" %>
<html>
<head>

   <script language="VB" runat="server">

     Sub Button1_Click(sender As Object, e As EventArgs)

       If Text1.Value = "" Then
         Span1.InnerHtml = "Error: you must enter a file name"
         Return
       End If

       If Not IsNothing(File1.PostedFile) Then
         Dim filepath As String = Path.Combine(Path.GetTempPath(),
Path.GetFileName(Text1.Value))

         Try
           File1.PostedFile.SaveAs(filepath)
           Span1.InnerHtml = "File uploaded successfully to <b>" & filepath &
"</b> on the web server"
         Catch Exc As Exception
           Span1.InnerHtml = "Error saving file <b>" & filepath &
"</b><br>" & Exc.ToString()
         End Try
       End If
     End Sub

   </script>

</head>
<body>

   <h3><font face="Verdana">HtmlInputFile Sample</font></h3>

   <form enctype="multipart/form-data" runat="server">

      Select File to Upload: <input id="File1" type=file runat="server">

      <p>
      Save as filename (no path): <input id="Text1" type="text"
runat="server">

      <p>
      <span id=Span1 style="font: 8pt verdana;" runat="server" />

      <p>
      <input          type=button          id="Button1"          value="Upload"
OnServerClick="Button1_Click" runat="server">

   </form>

</body>
</html>
```

HtmlInputHidden

Working with HtmlInputHidden

You can use hidden controls within HTML forms to embed non-visible information that will be sent back to the server the next time a user performs a postback. This technique is commonly used to persist session-dependent information without using cookies or session state. The Web Forms framework uses this feature of HTML to automatically store and restore the view state of ASP.NET server controls across round trips to the server.

The following sample illustrates using the **HtmlInputHidden** control.
Figure 3.10 HtmlInputhidden.aspx


Simple HtmlInputHidden Sample

Enter a string: Sachin

Enter

Hidden value: **Sachin**

Code for Figure 3.10 HtmlInputhidden.aspx

```
<html>
<head>
   <script language="VB" runat="server">
     Sub Page_Load(sender As Object, e As EventArgs)
        If IsPostBack Then
           Span1.InnerHtml="Hidden  value:  <b>"  &  HiddenValue.Value  &
"</b>"
        End If
```

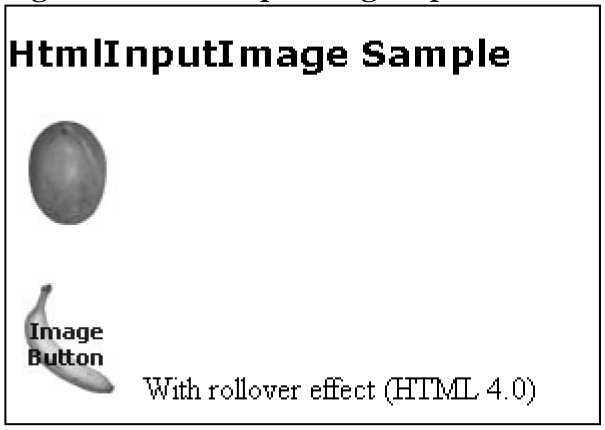
```
      End Sub
      Sub SubmitBtn_Click(sender As Object, e As EventArgs)
        HiddenValue.Value = StringContents.Value
      End Sub
   </script>
</head>
<body>
   <h3><font face="Verdana">Simple HtmlInputHidden
Sample</font></h3>
   <form runat=server>
      <input id="HiddenValue" type=hidden value="Initial Value"
runat=server>
      Enter a string: <input id="StringContents" type=text size=40
runat=server>
      <p>
      <input type=submit value="Enter"
OnServerClick="SubmitBtn_Click" runat=server>
      <p>
      <span id=Span1 runat=server>This label will display the previously
entered string.</span>
   </form>
</body>
</html>
```

HtmlInputImage

Working with HtmlInputImage

An **HtmlInputImage** control is used to create a graphical button. Unlike **HtmlButton** controls, all standard browser clients support image buttons.

Figure 3.11 HtmlInputImage.aspx



Colde for Figure 3.11 HtmlInputImage.aspx

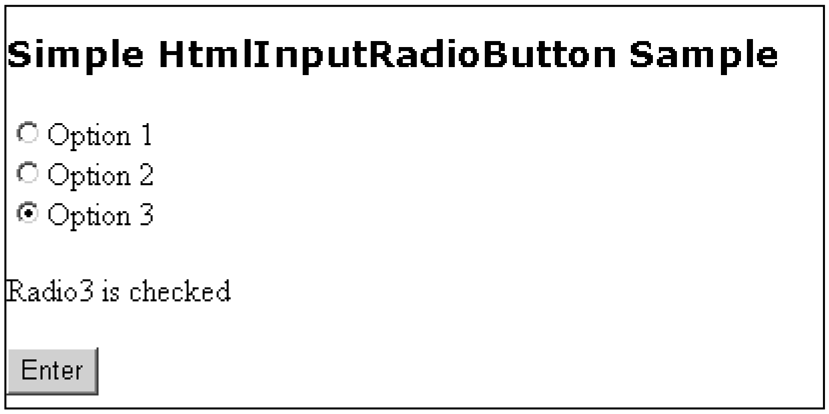
```
<html>
<head>
 <script language="VB" runat="server">

     Sub Button1_Click(sender As Object, e As ImageClickEventArgs)
       Span1.InnerHtml="You clicked button1"
     End Sub

     Sub Button2_Click(sender As Object, e As ImageClickEventArgs)
       Span1.InnerHtml="You clicked button2"
     End Sub
   </script>
</head>
<body>
   <h3><font face="Verdana">HtmlInputImage Sample</font></h3>
   <form runat=server>
      <input type=image id="InputImage1"
src="/quickstart/aspplus/images/mango.jpg"    OnServerClick="Button1_Click"
runat="server">
      <p>
      <input type=image id="InputImage2"
src="/quickstart/aspplus/images/mango.jpg"
         onmouseover="this.src='/quickstart/aspplus/images/banana.jpg';"
         onmouseout="this.src='/quickstart/aspplus/images/mango.jpg';"
         OnServerClick="Button2_Click"
         runat="server">
      &nbsp;With rollover effect (HTML 4.0)
      <p>
      <span id=Span1 runat=server />
   </form>
</body>
</html>
```

HtmlInputRadioButton

Working with HtmlInputRadioButton

An **HtmlInputRadioButton** control creates a single radio button input field. Setting the **Name** attribute the same way on each radio button forms a group in which only one radio button can be selected at a time. The selected state must be tested on the individual radio buttons, however.

Figure 3.12 HtmlInputRadioButton.aspx



Code for Figure 3.12 HtmlInputRadioButton.aspx

```
<html>
<head>
   <script language="VB" runat="server">
      Sub Button1_Click(sender As Object, e As EventArgs)
         If Radio1.Checked = True Then
            Span1.InnerHtml = "Radio1 is checked"
         Else If Radio2.Checked = True Then
            Span1.InnerHtml = "Radio2 is checked"
         Else If Radio3.Checked = true Then
            Span1.InnerHtml = "Radio3 is checked"
         End If
      End Sub
   </script>
</head>
<body>
   <h3><font        face="Verdana">Simple        HtmlInputRadioButton
Sample</font></h3>
   <form runat=server>
      <input  type="radio"  id="Radio1"  name="Mode"  runat="server"/>Option
1<br>
      <input  type="radio"  id="Radio2"  name="Mode"  runat="server"/>Option
2<br>
      <input type="radio" id="Radio3" name="Mode" runat="server"/>Option 3
      <p>
      <span id=Span1 runat=server />
      <p>
      <input type=button id="Button1" value="Enter"
OnServerClick="Button1_Click" runat=server>
   </form>
</body>
</html>
```

Coalesce

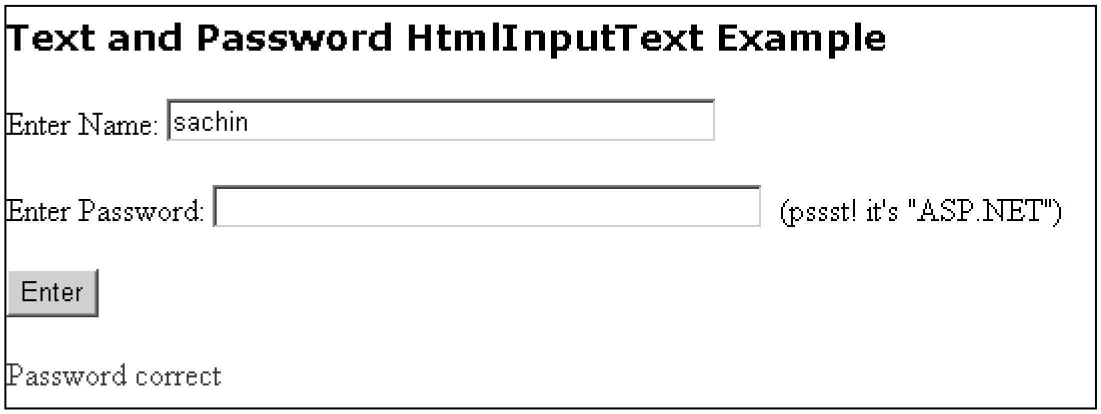
HtmlInputText

Working with HtmlInputText (Text and Password)

The **HtmlInputText** control is a single-line input control that lets the user enter text. **HtmlInputText** supports two behaviors. If **Type** is **Text**, **HtmlInputText** operates as a standard text box. If **Type** is **Password**, the user's input is masked by the "*" character to keep it private.

The following sample illustrates using the **HtmlInputText** control in both **Text** and **Password** modes.

Figure 3.13 HtmlTextandPassword.aspx



Text and Password HtmlInputText Example

Enter Name: sachin

Enter Password: (pssst! it's "ASP.NET")

Password correct

Code for Figure 3.13 HtmlTextandPassword.aspx

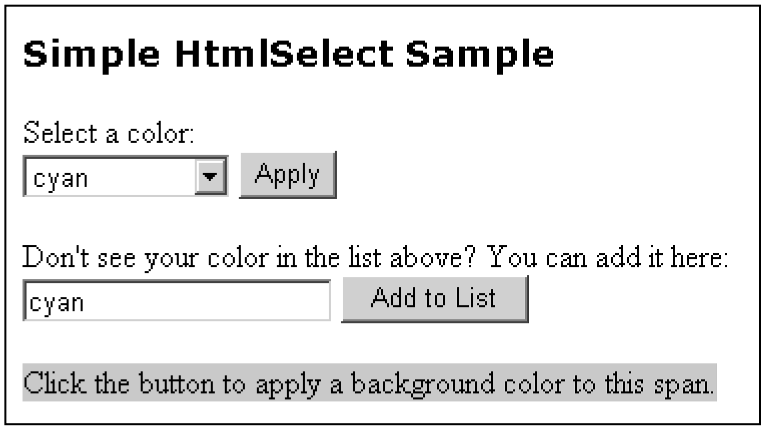
```
<html>
<head>
   <script language="VB" runat="server">
      Sub SubmitBtn_Click(sender As Object, e As EventArgs)
        If Password.Value = "ASP.NET" Then
           Span1.InnerHtml = "Password correct"
        Else
           Span1.InnerHtml = "That password is not correct"
        End If
      End Sub
   </script>
</head>
<body>
   <h3><font face="Verdana">Text and Password HtmlInputText
Example</font></h3>
   <form runat=server>
       Enter Name: <input id="Name" type=text size=40 runat=server>
       <p>
      Enter Password: <input id="Password" type=password size=40
runat=server> &nbsp;(pssst! it's "ASP.NET")
      <p>
      <input type=submit value="Enter" OnServerClick="SubmitBtn_Click"
runat=server>
      <p>
      <span id="Span1" style="color:red" runat=server></span>
   </form>
</body>
</html>
```

HtmlSelect

Working with HtmlSelect

The **HtmlSelect** control provides a drop-down list. The following sample illustrates using the **HtmlSelect** control.

Figure 3.14 HtmlSelect.aspx


Simple HtmlSelect Sample

Select a color:

Don't see your color in the list above? You can add it here:

Click the button to apply a background color to this span.

Code for Figure 3.14 HtmlSelect.aspx

```
<html>
<head>
   <script language="VB" runat="server">
     Sub Apply_Click(sender As Object, e As EventArgs)
       Span1.Style("background-color") = ColorSelect.Value
     End Sub
      Sub AddToList_Click(sender As Object, e As EventArgs)
       ColorSelect.Items.Add(Text1.Value)
     End Sub
   </script>
```

```
</head>
<body>
   <h3><font face="Verdana">Simple HtmlSelect Sample</font></h3>
   <form runat=server>
      Select a color:<br>
      <select id="ColorSelect" runat="server">
         <option>SkyBlue</option>
          <option>LightGreen</option>
          <option>Gainsboro</option>
          <option>LemonChiffon</option>
      </select>
      <input          type="button"          runat="server"          Value="Apply"
OnServerClick="Apply_Click">
      <p>
      Don't see your color in the list above?  You can add it here:<br>
      <input type="text" id="Text1" runat="server">
     <input     type="button"     runat="server"     Value="Add     to     List"
OnServerClick="AddToList_Click">
      <p>
      <span id="Span1" runat="server">Click the button to apply a background
color to this span.</span>
   </form>
</body>
</html>
<html>
<head>
   <script language="VB" runat="server">
     Sub Page_Load(sender As Object, e As EventArgs)
         If Not IsPostBack Then
            Dim values as ArrayList= new ArrayList()
            values.Add ("IN")
            values.Add ("KS")
            values.Add ("MD")
            values.Add ("MI")
            values.Add ("OR")
            values.Add ("TN")
            StateSelect.DataSource = values
            StateSelect.DataBind
         End If
      End Sub
      Sub SubmitBtn_Click(sender As Object, e As EventArgs)
         Span1.InnerHtml = "You chose: " & StateSelect.Value
      End Sub
   </script>
</head>
<body>
   <h3><font face="Verdana">DataBinding HtmlSelect</font></h3>
   <form runat=server>
```

```
        Select a state:<br>
        <select id="StateSelect" runat="server" />
        <input type="submit" value="Copy to span"
OnServerClick="SubmitBtn_Click" runat="server">
        <p>
        <span id="Span1" runat="server" />
   </form>
</body>
</html>
```

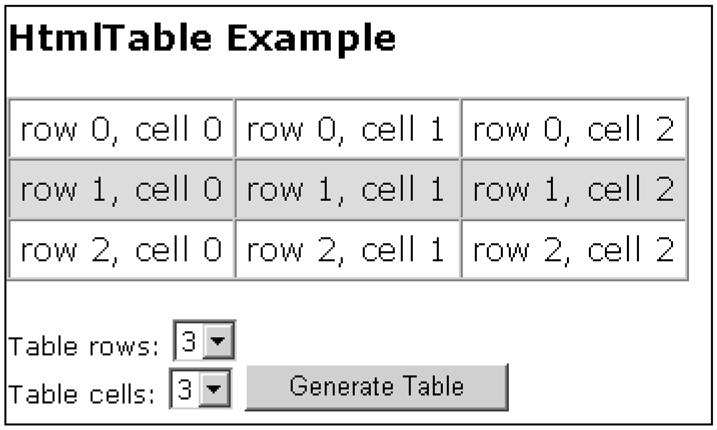
HtmlTable, HtmlTableRow, and HtmlTableCell

Working with HtmlTable, HtmlTableRow, and HtmlTableCell

The **HtmlTable** control lets you build up a table programmatically by adding **HtmlTableRow** controls to the table's Rows collection and **HtmlTableCell** controls to the row's Cells collection. You can add content to a table cell programmatically by adding controls to the cell's Controls collection.

The following sample illustrates using the **HtmlTable** control.

Figure 3.15 HtmlTable.aspx


HtmlTable Example

row 0, cell 0	row 0, cell 1	row 0, cell 2
row 1, cell 0	row 1, cell 1	row 1, cell 2
row 2, cell 0	row 2, cell 1	row 2, cell 2

Table rows: 3
Table cells: 3

Code for Figure 3.15 HtmlTable.aspx

```
<html>
<head>
   <script language="VB" runat="server">
     Sub Page_Load(sender As Object, e As EventArgs)

          Dim numrows As Integer
          Dim numcells As Integer
```

```
        Dim i As Integer = 0
        Dim j As Integer = 0
        Dim Row As Integer = 0
        Dim r As HtmlTableRow
        Dim c As HtmlTableCell

        ' Generate rows and cells
        numrows = CInt(Select1.Value)
        numcells = CInt(Select2.Value)
        For j = 0 To numrows-1
          r = new HtmlTableRow()
          If (row Mod 2 <> 0) Then
            r.BgColor = "Gainsboro"
          End If
          row += 1
          For i = 0 To numcells-1
            c = new HtmlTableCell()
            c.Controls.Add(new LiteralControl("row " & j & ", cell " & i))
            r.Cells.Add(c)
          Next i
          Table1.Rows.Add(r)
        Next j
     End Sub
  </script>
</head>
<body>
  <h3><font face="Verdana">HtmlTable Example</font></h3>
  <form runat=server>
  <font face="Verdana" size="-1">
      <p>
      <table    id="Table1"    CellPadding=5    CellSpacing=0    Border="1"
runat="server" />
      <p>
      Table rows:
      <select id="Select1" runat="server">
        <option Value="1">1</option>
        <option Value="2">2</option>
        <option Value="3">3</option>
        <option Value="4">4</option>
        <option Value="5">5</option>
      </select>
      <br>
      Table cells:
      <select id="Select2" runat="server">
        <option Value="1">1</option>
        <option Value="2">2</option>
        <option Value="3">3</option>
```

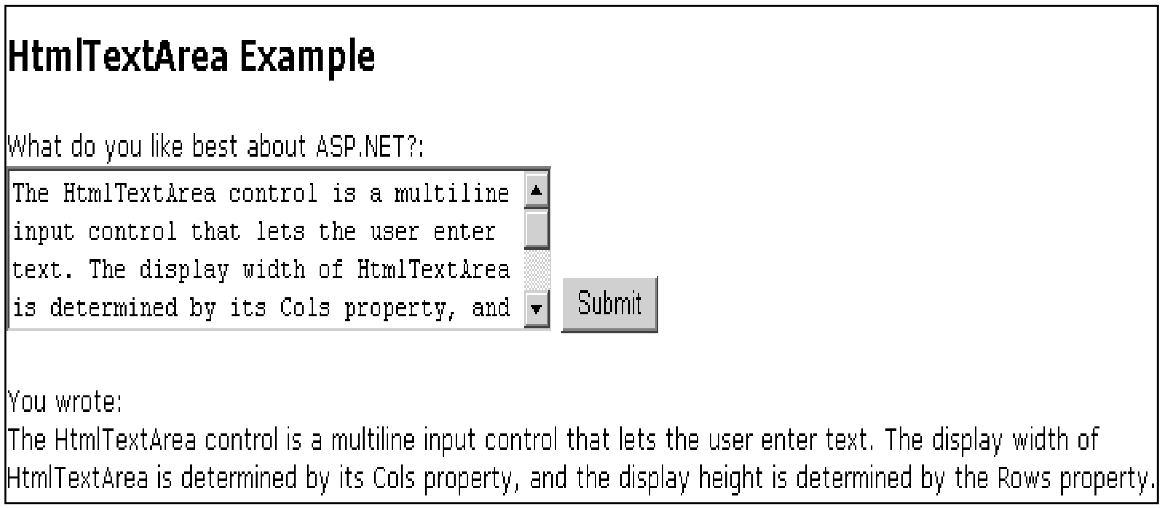
```
            <option Value="4">4</option>
            <option Value="5">5</option>
        </select>
        <input type="submit" value="Generate Table" runat="server">
    </font>
    </form>
</body>
</html>
```

HtmlTextArea

Working with HtmlTextArea

The **HtmlTextArea** control is a multiline input control that lets the user enter text. The display width of **HtmlTextArea** is determined by its **Cols** property, and the display height is determined by the **Rows** property.

Figure 3.16 HtmlTextarea.aspx



HtmlTextArea Example

What do you like best about ASP.NET?:

```
The HtmlTextArea control is a multiline
input control that lets the user enter
text. The display width of HtmlTextArea
is determined by its Cols property, and
```

You wrote:
The HtmlTextArea control is a multiline input control that lets the user enter text. The display width of HtmlTextArea is determined by its Cols property, and the display height is determined by the Rows property.

Code for Figure 3.16 HtmlTextarea.aspx

```
<html>
<head>
    <script language="VB" runat="server">
      Sub SubmitBtn_Click(sender As Object, e As EventArgs)
           Span1.InnerHtml = "You wrote: <br>" & TextArea1.Value
      End Sub
    </script>
</head>
<body>
```

```
   <h3><font face="Verdana">HtmlTextArea Example</font></h3>
   <form runat=server>
     <font face="Verdana" size="-1">
       What do you like best about ASP.NET?: <br>
       <textarea id="TextArea1" cols=40 rows=4 runat=server />
       <input   type=submit   value="Submit"   OnServerClick="SubmitBtn_Click"
runat=server>
       <p>
       <span id="Span1" runat="server" />
     </font>
   </form>
</body>
</html>
```

System.Web.UI.WebControls

Web server controls are ASP.NET server controls with an abstract, strongly-typed object model. Web server controls include not only form-type controls such as buttons and text boxes, but also special-purpose controls such as a calendar. Web server controls are more abstract than HTML server controls, in that their object model does not necessarily reflect HTML syntax.

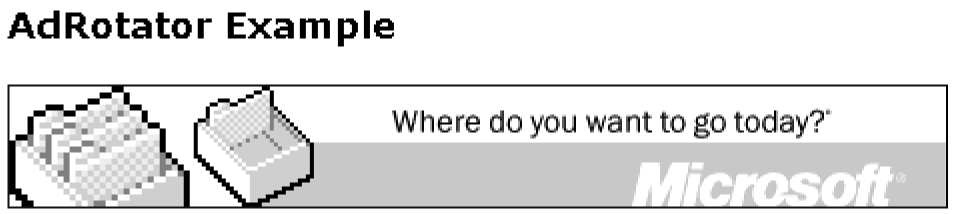
AdRotator	Button	Calendar	CheckBox
CheckBoxList	CompareValidator	CustomValidator	DataGrid
DataList	DropDownList	HyperLink	Image
ImageButton	Label	LinkButton	ListBox
Panel	PlaceHolder	RadioButton	RadioButtonList
RangeValidator	RegularExpressionValidator	Repeater	RequiredFieldValidator
Table	TableCell	TableRow	TextBox
ValidationSummary	XML		

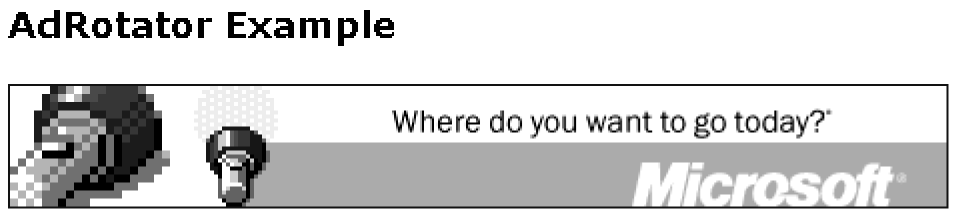
AdRotator

Working with AdRotator
The **AdRotator** control presents ad images that, when clicked, navigate to a new Web location. Each time the page is loaded into the browser, an ad is randomly selected from a predefined list. The following sample illustrates using the **AdRotator** control.

Figure 3.18 AdRotator1.aspx

Coalesce


AdRotator Example


AdRotator Example

The rotation schedule for ads is defined in an XML file. The following example demonstrates a rotation schedule in the file ads.xml.

```
   <Advertisements>
     <Ad>
       <ImageUrl>/quickstart/aspplus/images/banner1.gif</ImageUrl>
       <NavigateUrl>http://www.microsoft.com</NavigateUrl>
       <AlternateText>Microsoft.com</AlternateText>
       <Keyword>Computers</Keyword>
      <Impressions>80</Impressions>

    </Ad>    </Advertisements>
```

The rotation file defines the following attributes of each ad. Except for **ImageUrl**, these attributes are optional.

Attribute	**Description**
ImageUrl	An absolute or relative URL to the ad image file.
NavigateUrl	The Web location to navigate to when the image is clicked. If **NavigateUrl** is not set, the image is not clickable.
AlternateText	The text to render as the **ALT** attribute of the image. When the page is viewed with Microsoft Internet Explorer, this acts as a ToolTip for the ad.
Keyword	Specifies a category for the ad that the page can filter on.
Impressions	A number that indicates the "weight" of the ad in the schedule of rotation relative to the other ads in the file. The larger the number, the more often the ad will be displayed.

Code for Figure 3.18 AdRotator1.aspx

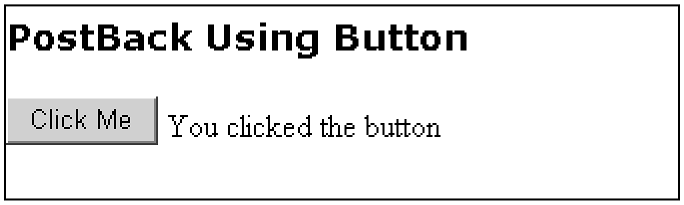
```
<html>
<body>
   <h3><font face="Verdana">AdRotator Example</font></h3>
   <form runat=server>
      <asp:AdRotator id="ar1" AdvertisementFile="Ads.xml"
BorderWidth="1" runat=server />
   </form>
</body>
</html>
```

Button

Postback Using Button

The **Button** control provides a command button-style control that is used to post a Web Forms page back to the server. The following sample illustrates using a simple **Button** control.

Figure 3.19 Button.aspx



Code for Figure 3.19 Button.aspx

```
<html>
<head>
   <script language="VB" runat="server">
      Sub Button1_Click(sender As Object, e As EventArgs)
         Label1.Text="You clicked the button"
      End Sub
   </script>
</head>
<body>
   <h3><font face="Verdana">PostBack Using Button</font></h3>
   <form runat=server>
      <asp:Button id=Button1 Text="Click Me"
onclick="Button1_Click" runat="server" />
      <asp:Label id=Label1 runat=server />
   </form>
</body>
</html>
```

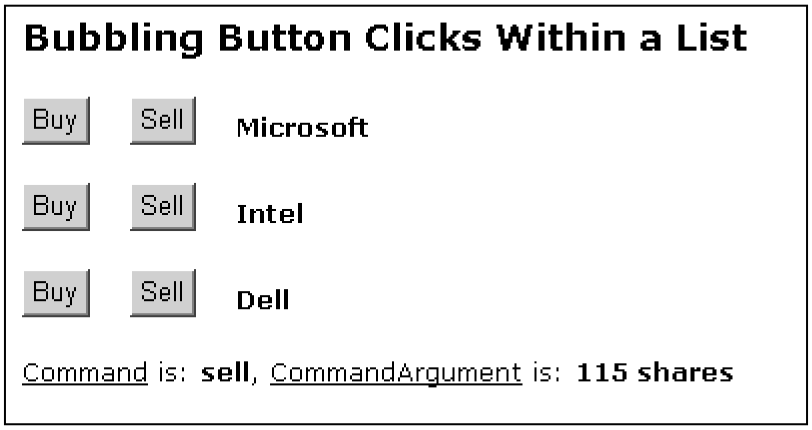
Bubbling Button Clicks Within a List

When used in a templated list such as a Repeater, DataList, or DataGrid, many **Button** controls might be rendered as the list iterates over its data source. For more information, see the Data Binding section. Because each of these **Button** controls shares the same ID, you cannot simply bind an event handler to each **Button** control's **OnClick** event to determine the particular **Button** that was clicked. To solve this, you use event bubbling to fire an event on the container control (in this case, the **Repeater**, **DataList**, or **DataGrid**), and let the container impart additional information to the event handler about the item that raised the event.

These events can be raised from a **Button** by specifying a **CommandName** property with the name of the event. When the **Button** is clicked, the command "bubbles" to the container control (such as **Repeater**), which fires its own event. The arguments for this event might contain additional information, such as a custom string or the index of the item that raised the event.

The following sample illustrates how a **Button** control's commands can bubble to the **OnItemCommand** event of a list. The **Button** control's **CommandName** and **CommandArgument** strings are passed to the **OnItemCommand** event, permitting the sample code to distinguish which button was clicked.

Figure 3.20 BubblingButton.aspx



Bubbling Button Clicks Within a List

Buy Sell **Microsoft**

Buy Sell **Intel**

Buy Sell **Dell**

Command is: **sell**, CommandArgument is: **115 shares**

Code for Figure 3.20 BubblingButton.aspx

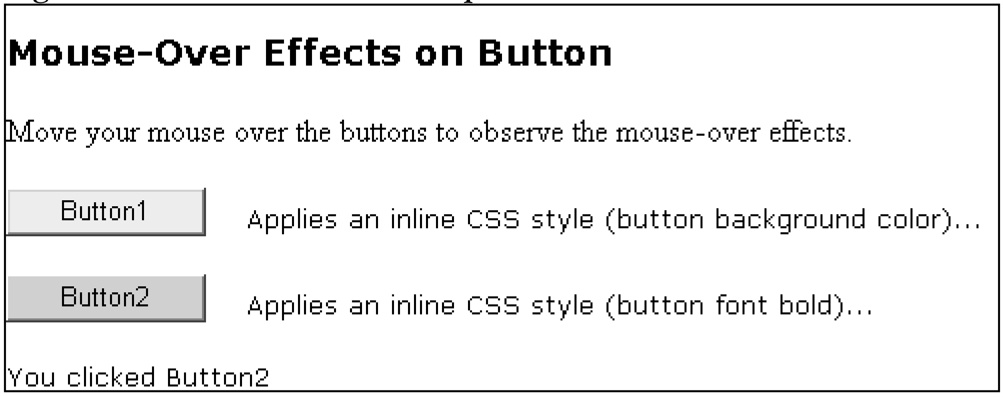
```
<html>
<head>
   <script language="VB" runat="server">
      Sub Page_Load(sender As Object, e As EventArgs)
      if Not IsPostBack Then
        dim values as ArrayList
        values = new ArrayList()
        values.Add(new PositionData("Microsoft", "Msft", "150 shares"))
        values.Add(new PositionData("Intel", "Intc", "25 shares"))
        values.Add(new PositionData("Dell", "Dell", "115 shares"))
        repeater1.DataSource = values
        repeater1.DataBind
      End If
    End Sub
    Sub    Repeater1_ItemCommand(sender    As    Object,    e    As
RepeaterCommandEventArgs)
      lblResult.Text = "<u>Command</u> is: <b>" & e.CommandName &
"</b>, <u>CommandArgument</u> is: <b>" & e.CommandArgument &
"</b>"
    End Sub
    class PositionData
      Dim m_name As String
      Dim m_ticker As String
      Dim m_shares As String
      Public Sub New(name As String, ticker As String, shares As String)
        MyBase.New
        m_name = name
        m_ticker = ticker
        m_shares = shares
      End Sub
      ReadOnly Property Name As String
       Get
        Return m_name
       End Get
      End Property
      ReadOnly Property Ticker As String
       Get
        Return m_ticker
       End Get
      End Property
      ReadOnly Property Shares As String
       Get
        Return m_shares
       End Get
      End Property
    End Class
  </script>
</head>
<body>
  <h3><font    face="Verdana">Bubbling    Button    Clicks    Within    a
List</font></h3>
```

```
    <p></p>
    <form runat=server>
      <font face="Verdana" size="-1">
        <asp:Repeater id=repeater1
onitemcommand="Repeater1_ItemCommand" runat="server">
          <ItemTemplate>
            <asp:Button id=btnBuy Text="Buy" CommandName="buy"
CommandArgument='<%# DataBinder.Eval(Container.DataItem, "Ticker")
%>' runat="server" />
            &nbsp;&nbsp;
            <asp:Button id=btnSell Text="Sell" CommandName="sell"
CommandArgument='<%# DataBinder.Eval(Container.DataItem, "Shares")
%>' runat="server" />
            &nbsp;&nbsp;
            <asp:Label id=lblCompany Text='<%#
DataBinder.Eval(Container.DataItem, "Name") %>' Font-Bold="true"
runat=server />
            <p> </ItemTemplate>
        </asp:Repeater>
        <asp:Label id=lblResult runat="server" />
      </font>
    </form>

</body></html>
```

Mouse-Over Effects on Button

You can hook the client script events **onmouseover** and **onmouseout** on a **Button** control to provide mouse-over effects such as changing the font or color of the button. Client attributes such as **onmouseover** are disregarded by ASP.NET on the server, and passed "as is" to the browser. If your application targets newer browsers that support DHTML, these events will fire in the browser as the cursor passes over the button. The following sample demonstrates buttons with mouse-over effects.

Figure 3.20 Mouse-OverEffects.aspx



Mouse-Over Effects on Button

Move your mouse over the buttons to observe the mouse-over effects.

Applies an inline CSS style (button background color)...

Applies an inline CSS style (button font bold)...

You clicked Button2

Code for Figure 3.20 Mouse-OverEffects.aspx

```
<html>
<head>
   <script language="VB" runat="server">
     Sub Button1_Click(sender As Object, e As EventArgs)
       Label1.Text="You clicked Button1"
     End Sub
     Sub Button2_Click(sender As Object, e As EventArgs)
       Label1.Text="You clicked Button2"
     End Sub
   </script>
</head>
<body>
   <h3><font face="Verdana">Mouse-Over Effects on Button</font></h3>
   <p>Move your mouse over the buttons to observe the mouse-over effects.</p>
   <form runat=server>
   <font face="Verdana" size="-1">

      <asp:Button id=Button1 runat="server"
        Text="Button1"
        Width="100px"
        onmouseover="this.style.backgroundColor='yellow'"
        onmouseout="this.style.backgroundColor='buttonface'"
        onclick="Button1_Click" />

        &nbsp;&nbsp;
        Applies an inline CSS style (button background color)...

      <p>

      <asp:Button id=Button2 runat="server"
        Text="Button2"
        Width="100px"
        onmouseover="this.style.fontWeight='bold'"
        onmouseout="this.style.fontWeight='normal'"
        onclick="Button2_Click" />

        &nbsp;&nbsp;
        Applies an inline CSS style (button font bold)...

      <p>

      <asp:Label id=Label1 runat=server />

   </font>
   </form>
```

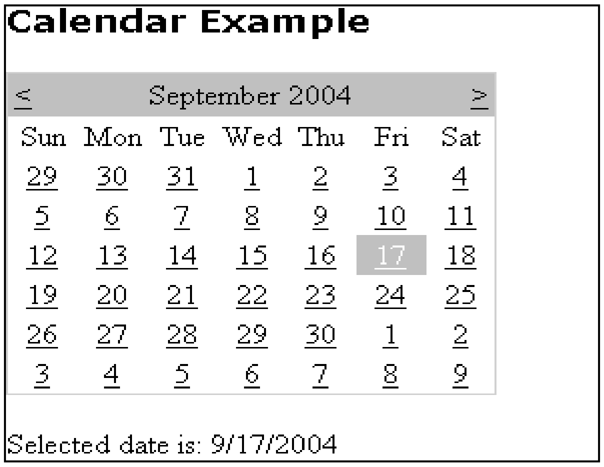
```
</body>
</html>
```

Calendar

Working With Calendar

The **Calendar** control displays a month calendar from which users can select dates. The following sample illustrates using a simple **Calendar** control.

Figure 3.21 Calendar.aspx



Calendar Example

<			September 2004			>
Sun	Mon	Tue	Wed	Thu	Fri	Sat
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	1	2
3	4	5	6	7	8	9

Selected date is: 9/17/2004

Code for Figure 3.21 Calendar.aspx

```
<html>
<head>
   <script language="VB" runat="server">
      Sub Date_Selected(sender As Object, e As EventArgs)
         Label1.Text = "Selected date is: " +
Calendar1.SelectedDate.ToShortDateString
      End Sub
   </script>
</head>
<body>
   <h3><font face="Verdana">Calendar Example</font></h3>
   <form runat=server>
      <asp:Calendar id=Calendar1 onselectionchanged="Date_Selected"
runat="server" />
      <p>
      <asp:Label id=Label1 runat="server" />
   </form>
```

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

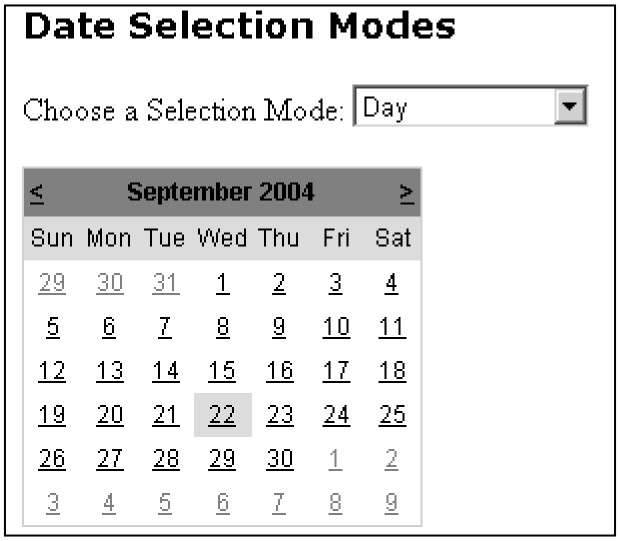
Date Selection Modes

Calendar supports four date selection modes, as described in the following table.

Mode	**Description**
Day	User can select any single day.
DayWeek	User can select a single day, or an entire week.
DayWeekMonth	User can select a single day, an entire week, or the entire visible month.
None	Date selection is disabled.

The following sample demonstrates mode selection with a **Calendar** control.

Figure 3.22 CalendarDateSelection.aspx



Date Selection Modes

Choose a Selection Mode: Day

<			September 2004			>
Sun	Mon	Tue	Wed	Thu	Fri	Sat
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	1	2
3	4	5	6	7	8	9

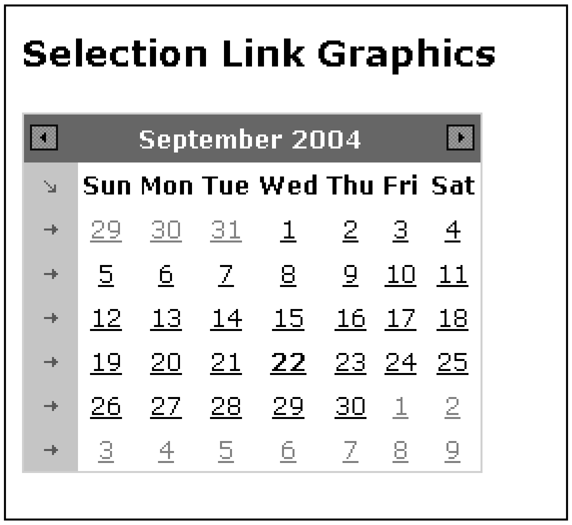
Code for Figure 3.22 CalendarDateSelection.aspx

```
<html>
<head>
   <script language="VB" runat="server">

      Sub Page_Load(sender As Object, e As EventArgs)
      Calendar1.SelectionMode = lstSelMode.SelectedIndex

      If Calendar1.SelectionMode = CalendarSelectionMode.None Then
        Calendar1.SelectedDates.Clear
      End If
    End Sub

    Sub Date_Selected(sender As Object, e As EventArgs)

      Select (Calendar1.SelectedDates.Count)
        Case 0:   'None
          Label1.Text = "No dates are currently selected"
        Case 1:   'Day
          Label1.Text    =    "The    selected    date    is    "    &
Calendar1.SelectedDate.ToShortDateString
        Case 7:   'Week
          Label1.Text  =  "The  selection  is  a  week  beginning  "  &
Calendar1.SelectedDate.ToShortDateString
        Case Else:  'Month
          Label1.Text  =  "The  selection  is  a  month  beginning  "  &
Calendar1.SelectedDate.ToShortDateString
      End Select

    End Sub

  </script>

</head>

<body>

  <h3><font face="Verdana">Date Selection Modes</font></h3>
  <p>

  <form runat=server>

    Choose a Selection Mode:
    <asp:DropDownList id="lstSelMode" runat=server
      AutoPostBack=true>
      <asp:ListItem Value="None" >None</asp:ListItem>
      <asp:ListItem Selected Value="Day" >Day</asp:ListItem>
      <asp:ListItem Value="DayWeek" >DayWeek</asp:ListItem>
      <asp:ListItem                    Value="DayWeekMonth"
>DayWeekMonth</asp:ListItem>
    </asp:DropDownList>

    <p>
    <asp:Calendar id=Calendar1 runat="server"
      onselectionchanged="Date_Selected"
      Font-Name="Arial" Font-Size="12px"
      Height="180px" Width="200px"
      SelectorStyle-BackColor="gainsboro"
      TodayDayStyle-BackColor="gainsboro"
      DayHeaderStyle-BackColor="gainsboro"
      OtherMonthDayStyle-ForeColor="gray"
      TitleStyle-BackColor="gray"
      TitleStyle-Font-Bold="True"
      TitleStyle-Font-Size="12px"
      SelectedDayStyle-BackColor="Navy"
      SelectedDayStyle-Font-Bold="True"
      />

    <p>
    <asp:Label id=Label1 runat="server" />
  </form>

</body>
</html>
```

Selection Link Graphics

The **Calendar** control can use either text or graphics for its selection links. The following sample shows how to use graphics to create a better-looking calendar.

Figure 3.23 CalendarDateSelectionLink.aspx



Code for Figure 3.23 CalendarDateSelectionLink.aspx

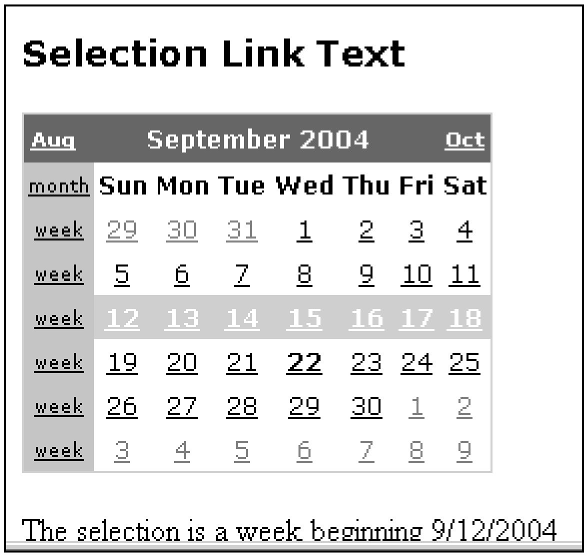
Coalesce

```
<html>
<head>
  <script language="VB" runat="server">

     Sub Date_Selected(sender As Object, e As EventArgs)

        Select (Calendar1.SelectedDates.Count)
           Case 0:   'None
              Label1.Text = "No dates are currently selected"
           Case 1:   'Day
              Label1.Text     =     "The     selected     date     is     "     &
Calendar1.SelectedDate.ToShortDateString
           Case 7:   'Week
              Label1.Text   =   "The   selection   is   a   week   beginning   "   &
Calendar1.SelectedDate.ToShortDateString
           Case Else:  'Month
              Label1.Text   =   "The   selection   is   a   month   beginning   "   &
Calendar1.SelectedDate.ToShortDateString
        End Select

     End Sub

  </script>
</head>

<body>

  <h3><font face="Verdana">Selection Link Graphics</font></h3>
  <p>

  <form runat=server>

     <asp:Calendar id=Calendar1 runat="server"
        onselectionchanged="Date_Selected"
        DayNameFormat="Short"
        SelectionMode="DayWeekMonth"
        Font-Name="Verdana;Arial" Font-Size="12px"
        Height="180px" Width="230px"
        TodayDayStyle-Font-Bold="True"
        DayHeaderStyle-Font-Bold="True"
        OtherMonthDayStyle-ForeColor="gray"
        TitleStyle-BackColor="#3366ff"
        TitleStyle-ForeColor="white"
        TitleStyle-Font-Bold="True"
        SelectedDayStyle-BackColor="#ffcc66"
        SelectedDayStyle-Font-Bold="True"
        NextMonthText = "<img
src='/quickstart/aspplus/images/monthright.gif' border=0>"
        PrevMonthText = "<img src='/quickstart/aspplus/images/monthleft.gif'
border=0>"
        SelectorStyle-BackColor="#99ccff"
        SelectWeekText = "<img src='/quickstart/aspplus/images/selweek.gif'
border=0 onmouseover=this.style.backgroundColor='#ffcc66'
onmouseout=this.style.backgroundColor='#99ccff'>"
        SelectMonthText = "<img src='/quickstart/aspplus/images/selmonth.gif'
border=0 onmouseover=this.style.backgroundColor='#ffcc66'
onmouseout=this.style.backgroundColor='#99ccff'>"
        />

      <p>

      <asp:Label id=Label1 runat="server" />

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

Selection Link Text

The **Calendar** control can also use text labels for week or month selection, as shown in the following example.

Figure 3.24 CalendarDateSelectionLink.aspx


Selection Link Text

Aug			September 2004				Oct
month	Sun	Mon	Tue	Wed	Thu	Fri	Sat
week	29	30	31	1	2	3	4
week	5	6	7	8	9	10	11
week	12	13	14	15	16	17	18
week	19	20	21	22	23	24	25
week	26	27	28	29	30	1	2
week	3	4	5	6	7	8	9

The selection is a week beginning 9/12/2004

Figure 3.24 CalendarDateSelectionLink.aspx

```
<html>
<head>
   <script language="VB" runat="server">
      Sub Date_Selected(sender As Object, e As EventArgs)
         Select (Calendar1.SelectedDates.Count)
            Case 0:   'None
               Label1.Text = "No dates are currently selected"
            Case 1:   'Day
               Label1.Text = "The selected date is " &
Calendar1.SelectedDate.ToShortDateString
            Case 7:   'Week
               Label1.Text = "The selection is a week beginning " &
Calendar1.SelectedDate.ToShortDateString
            Case Else:  'Month
               Label1.Text = "The selection is a month beginning " &
Calendar1.SelectedDate.ToShortDateString
         End Select
      End Sub
   </script>
</head>
<body>
   <h3><font face="Verdana">Selection Link Text</font></h3>
   <p>
   <form runat=server>
      <p>
      <asp:Calendar id=Calendar1 runat="server"
         onselectionchanged="Date_Selected"
         DayNameFormat="Short"
         SelectionMode="DayWeekMonth"
         Font-Name="Verdana;Arial" Font-Size="12px"
         Height="180px" Width="230px"
         TodayDayStyle-Font-Bold="True"
         DayHeaderStyle-Font-Bold="True"
         OtherMonthDayStyle-ForeColor="gray"
         TitleStyle-BackColor="#3366ff"
         TitleStyle-ForeColor="white"
         TitleStyle-Font-Bold="True"
         SelectedDayStyle-BackColor="#ffcc66"
         SelectedDayStyle-Font-Bold="True"
         NextPrevFormat="ShortMonth"
         NextPrevStyle-ForeColor="white"
         NextPrevStyle-Font-Size="10px"
         SelectorStyle-BackColor="#99ccff"
         SelectorStyle-ForeColor="navy"
         SelectorStyle-Font-Size="9px"
         SelectWeekText = "week"
```

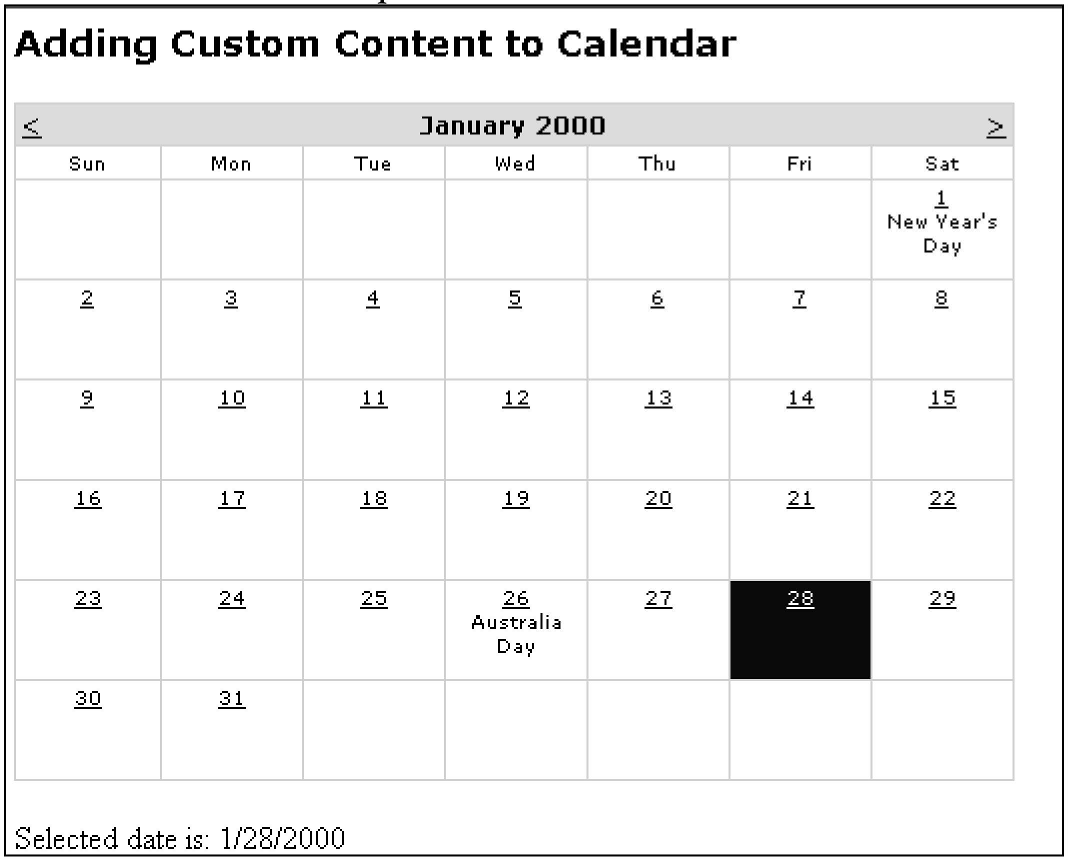
```
        SelectMonthText = "month"
        />
      <p>
      <asp:Label id=Label1 runat="server" />
   </form>
</body>
</html>
```

Adding Custom Content to Calendar

You can make appointment-style calendars by adding content in the **OnDayRender** event. Two of the arguments for **OnDayRender** are the **Day** that is being rendered and its **Cell** object. Custom text can be added to the cell for a particular day by adding it as a **LiteralControl** to the **Cell** object's Controls collection, as shown in the following example.

```
Dim Hol As String = GetHoliday(Day.Date)
If Hol <> String.Empty Then Cells.Controls.Add(New LiteralControl("<br>" +
Hol))
```

Figure 3.25
CalendarCustomContent.aspx



Adding Custom Content to Calendar

<			January 2000			>
Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1 New Year's Day
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26 Australia Day	27	28	29
30	31					

Selected date is: 1/28/2000

Code for Figure 3.25 CalendarCustomContent.aspx

Coalesce

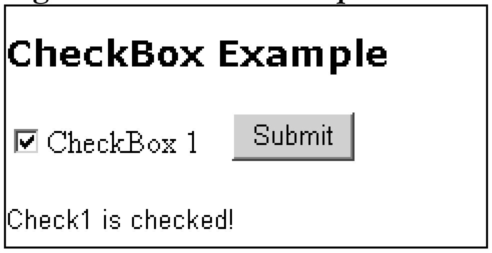
```
<html>
<head>
  <script language="VB" runat="server">

    Dim holidays(12,31) as String

    Sub Page_Load(sender As Object, e As EventArgs)
      holidays(1,1)   = "New Year's Day"
        holidays(1,26)  = "Australia Day"
      holidays(2,2)   = "Groundhog Day"
      holidays(2,14)  = "Valentine's Day"
      holidays(3,17)  = "St. Patrick's Day"
      holidays(4,1)   = "April Fool's Day"
      holidays(5,1)   = "May Day"
      holidays(6,15)  = "My Birthday"
      holidays(7,15)  = "My Anniversary"
      holidays(8,15)  = "My Mother's Birthday"
      holidays(9,24)  = "Autumnal Equinox"
      holidays(12,26) = "Boxing Day"
    End Sub

    Sub Calendar1_DayRender(sender As Object, e As DayRenderEventArgs)

      Dim d as CalendarDay
      Dim c as TableCell

      d = e.Day
      c = e.Cell

      If d.IsOtherMonth Then
        c.Controls.Clear
      Else
        Try
          Dim Hol As String = holidays(d.Date.Month,d.Date.Day)

          If Hol <> "" Then
            c.Controls.Add(new LiteralControl("<br>" + Hol))
          End If
        Catch exc as Exception
          Response.Write (exc.ToString())
        End Try
      End If
    End Sub

    Sub Date_Selected(sender As Object, e As EventArgs)
      Label1.Text      =      "Selected      date      is:      "      +
Calendar1.SelectedDate.ToShortDateString
      End Sub

   </script>

</head>

<body>

   <h3><font face="Verdana">Adding Custom Content to
Calendar</font></h3>
   <p><p>

   <form runat=server>

      <asp:Calendar id=Calendar1 runat="server"
         ondayrender="Calendar1_DayRender"
         onselectionchanged="Date_Selected"
         ShowGridLines="true"
         BorderWidth="1"
         Font-Name="Verdana"
         Font-Size="9px"
         Width="500px"
         VisibleDate="01/01/2000"
         TitleStyle-BackColor="Gainsboro"
         TitleStyle-Font-Size="12px"
         TitleStyle-Font-Bold="true"
         DayStyle-VerticalAlign="Top"
         DayStyle-Height="50px"
         DayStyle-Width="14%"
         SelectedDate="1/1/0001"
         SelectedDayStyle-BackColor="Navy"
         />

      <p>
      <asp:Label id=Label1 runat="server" />
   </form>

</body>
</html>
```

CheckBox

Working with CheckBox
The **CheckBox** server control accepts Boolean (**true** or **false**) input. When selected, its **Checked** property is **true**. Typically a check box is processed as one of several

fields in a form; however, it can be used to trigger postback to the server if its **AutoPostBack** property is **true**. The following sample illustrates using the **CheckBox** control.

Figure 3.26 Checkbox.aspx



Code for Figure 3.26 Checkbox.aspx

```
<html>
<head>
   <script language="VB" runat="server">
      Sub SubmitBtn_Click(sender As Object, e As EventArgs)
         If Check1.Checked = true Then
            Label1.Text = "Check1 is checked!"
         Else
            Label1.Text = "Check1 is not checked!"
         End If
      End Sub
   </script>
</head>
<body>
   <h3><font face="Verdana">CheckBox Example</font></h3>
   <form runat=server>
      <asp:CheckBox id=Check1 Text="CheckBox 1" runat="server" />

      &nbsp&nbsp

      <asp:button         text="Submit"         OnClick="SubmitBtn_Click"
runat=server/>

      <p>

      <asp:Label      id=Label1      font-name="arial"      font-size="10pt"
runat="server"/>
   </form>
</body>
</html>
```

Coalesce

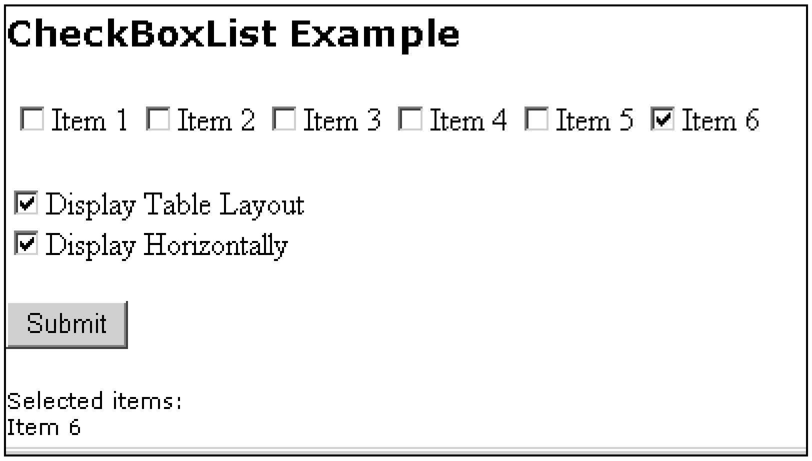
CheckBoxList

Working with CheckBoxList

The **CheckBoxList** control provides a multiple-selection checked list. Like other list controls, **CheckBoxList** has an **Items** collection with members that correspond to each item in the list. To determine which items are selected, test the **Selected** property of each item.

You can control the rendering of the list with the **RepeatLayout** and **RepeatDirection** properties. If **RepeatLayout** is **Table**, the list is rendered within a table. If it is set to **Flow**, the list is rendered without any table structure. By default, **RepeatDirection** is **Vertical**. Setting this property to **Horizontal** causes the list to be rendered horizontally.

Figure 3.27 CheckboxList.aspx


CheckBoxList Example

☐ Item 1 ☐ Item 2 ☐ Item 3 ☐ Item 4 ☐ Item 5 ☑ Item 6

☑ Display Table Layout
☑ Display Horizontally

Selected items:
Item 6

Code for Figure 3.27 CheckboxList.aspx

```
<html>
<head>
   <script language="VB" runat="server">
    Sub Button1_Click(sender As Object, e As EventArgs)
                Dim s As String = "Selected items:<br>"
       Dim i As Int32
       For i = 0 to Check1.Items.Count-1
          If Check1.Items(i).Selected Then
             ' List the selected items
             s = s & Check1.Items(i).Text
             s = s & "<br>"
          End If
       Next
       Label1.Text = s
     End Sub
```

```
      Sub chkLayout_CheckedChanged(sender As Object, e As EventArgs)
        If chkLayout.Checked = true Then
          Check1.RepeatLayout = RepeatLayout.Table
        Else
          Check1.RepeatLayout = RepeatLayout.Flow
        End If
      End Sub
      Sub chkDirection_CheckedChanged(sender As Object, e As EventArgs)
        If chkDirection.Checked = true Then
          Check1.RepeatDirection = RepeatDirection.Horizontal
        Else
          Check1.RepeatDirection = RepeatDirection.Vertical
        End If
      End Sub
   </script>
</head>
<body>
   <h3><font face="Verdana">CheckBoxList Example</font></h3>
   <form runat=server>
      <asp:CheckBoxList id=Check1 runat="server">
         <asp:ListItem>Item 1</asp:ListItem>
         <asp:ListItem>Item 2</asp:ListItem>
         <asp:ListItem>Item 3</asp:ListItem>
         <asp:ListItem>Item 4</asp:ListItem>
         <asp:ListItem>Item 5</asp:ListItem>
         <asp:ListItem>Item 6</asp:ListItem>
      </asp:CheckBoxList>
      <p>
      <asp:CheckBox id=chkLayout
OnCheckedChanged="chkLayout_CheckedChanged" Text="Display
Table Layout" Checked=true AutoPostBack="true" runat="server" />
      <br>
      <asp:CheckBox id=chkDirection
OnCheckedChanged="chkDirection_CheckedChanged" Text="Display
Horizontally" AutoPostBack="true" runat="server" />
      <p>
      <asp:Button id=Button1 Text="Submit" onclick="Button1_Click"
runat="server"/>
      <p>
      <asp:Label id=Label1 font-name="Verdana" font-size="8pt"
runat="server"/>
   </form>
</body>
</html>
```

Coalesce

CompareValidator

Working with CompareValidator

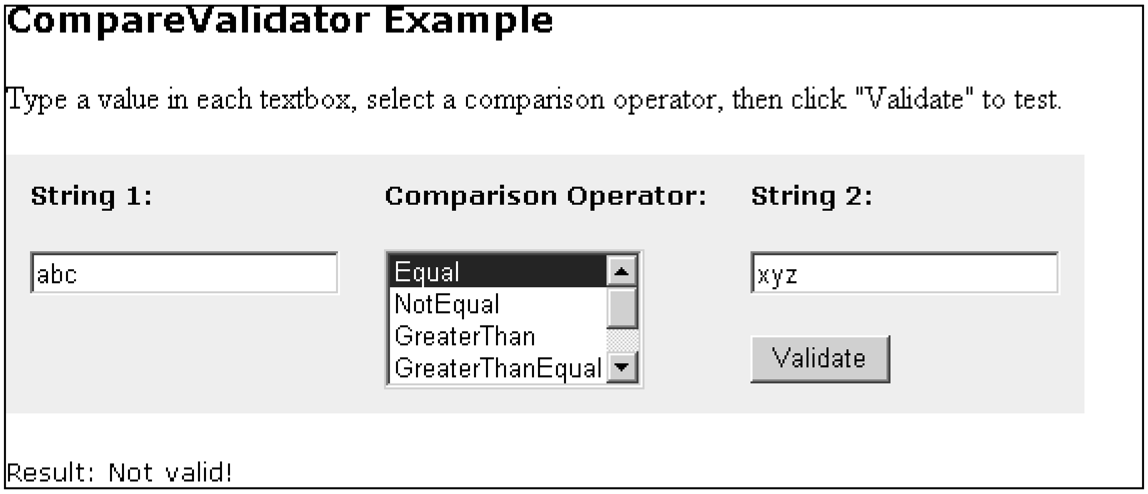
The **CompareValidator** control compares the value of one control to another, or to an explicit value in the control's **ValueToCompare** property.

Note: For the purpose of validation, a specific property on the control is designated as its "value". For more information, see the Server Control Form Validation section.

CompareValidator uses three key properties to perform its validation. **ControlToValidate** and **ControlToCompare** contain the values to compare. **Operator** defines the type of comparison to perform, for example, Equal or Not Equal. **CompareValidator** performs the validation by evaluating these properties as an expression, as shown in the following example.

If the expression evaluates **true**, the validation result is valid.
The following sample illustrates using the **CompareValidator** control.

Figure 3.28 CompareValidator.aspx


CompareValidator Example

Type a value in each textbox, select a comparison operator, then click "Validate" to test.

String 1: abc

Comparison Operator: Equal

String 2: xyz

Result: Not valid!

Code for Figure 3.28 CompareValidator.aspx

```
<%@ Page clienttarget=downlevel %>
<html>
<head>
   <script language="VB" runat="server">
      Sub Button1_OnSubmit(sender As Object, e As EventArgs)
         If Page.IsValid Then
            lblOutput.Text = "Result: Valid!"
         Else
```

```
            lblOutput.Text = "Result: Not valid!"
          End If
        End Sub
        Sub lstOperator_SelectedIndexChanged(sender As Object, e As EventArgs)
          comp1.Operator = lstOperator.SelectedIndex
          comp1.Validate
        End Sub
   </script>
</head>
<body>
   <h3><font face="Verdana">CompareValidator Example</font></h3>
   <p>Type a value in each textbox, select a comparison operator, then click
"Validate" to test.</p>
   <form runat=server>
     <table bgcolor="#eeeeee" cellpadding=10>
     <tr valign="top">
       <td>
          <h5><font face="Verdana">String 1:</font></h5>
          <asp:TextBox Selected id="txtComp"
runat="server"></asp:TextBox>
       </td>
       <td>
          <h5><font face="Verdana">Comparison Operator:</font></h5>
          <asp:ListBox id="lstOperator"
OnSelectedIndexChanged="lstOperator_SelectedIndexChanged"
runat="server">
                <asp:ListItem Selected Value="Equal" >Equal</asp:ListItem>
                <asp:ListItem Value="NotEqual" >NotEqual</asp:ListItem>
                <asp:ListItem Value="GreaterThan"
>GreaterThan</asp:ListItem>
                <asp:ListItem Value="GreaterThanEqual"
>GreaterThanEqual</asp:ListItem>
                <asp:ListItem Value="LessThan" >LessThan</asp:ListItem>
                <asp:ListItem Value="LessThanEqual"
>LessThanEqual</asp:ListItem>
          </asp:ListBox>
       </td>
       <td>
          <h5><font face="Verdana">String 2:</font></h5>
          <asp:TextBox id="txtCompTo"
runat="server"></asp:TextBox><p>
          <asp:Button runat=server Text="Validate" ID="Button1"
onclick="Button1_OnSubmit" />
       </td>
     </tr>
     </table>
    <asp:CompareValidator id="comp1" ControlToValidate="txtComp"
```

```
ControlToCompare = "txtCompTo" Type="String" runat="server"/>
    <br>
    <asp:Label ID="lblOutput" Font-Name="verdana" Font-Size="10pt"
runat="server"/>
   </form>
</body>
</html>
```

CustomValidator

Working with CustomValidator

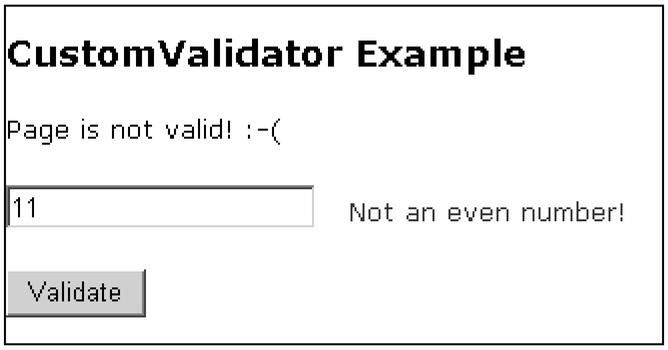
The **CustomValidator** control calls a user-defined function to perform validations that the standard validators can't handle. The custom function can execute on the server or in client-side script, such as JScript or VBScript. For client-side custom validation, the name of the custom function must be identified in the **ClientValidationFunction** property. The custom function must have the form

   function myvalidator(source, arguments)

Note that **source** is the client-side **CustomValidator** object, and **arguments** is an object with two properties, **Value** and **IsValid**. The **Value** property is the value to be validated and the **IsValid** property is a Boolean used to set the return result of the validation. You can view a client-side validation example in the ASP.NET Validation section.

For server-side custom validation, place your custom validation in the validator's **OnServerValidate** delegate.

Figure 3.29 CustomValidator.aspx


CustomValidator Example

Page is not valid! :-(

11   Not an even number!

Code for Figure 3.29 CustomValidator.aspx

```
<html>
<head>
   <script language="VB" runat=server>
      Sub ValidateBtn_OnClick(sender As Object, e As EventArgs)
```

```
        If (Page.IsValid) Then
          lblOutput.Text = "Page is valid!"
        Else
          lblOutput.Text = "Page is not valid! :-("
        End If
      End Sub
      Sub ServerValidate (sender As Object, value As ServerValidateEventArgs)
        Try
          Dim num As Int32 = Int32.Parse(value.Value)
          If num Mod 2 = 0 Then
            value.IsValid = True
            Exit Sub
          End If
        Catch E As Exception
          ' Do Nothing
        End Try
        value.IsValid = False
      End Sub
  </script>
</head>
<body>
<h3><font face="Verdana">CustomValidator Example</font></h3>
<form runat="server">
  <asp:Label id=lblOutput runat="server"
    Text="Enter an even number:"
    Font-Name="Verdana"
    Font-Size="10pt" /><br>
  <p>
  <asp:TextBox id=Text1 runat="server" />
  &nbsp&nbsp
  <asp:CustomValidator id="CustomValidator1" runat="server"
    ControlToValidate="Text1"
    OnServerValidate="ServerValidate"
    Display="Static"
    Font-Name="verdana" Font-Size="10pt">
      Not an even number!
  </asp:CustomValidator>
  <p>
  <asp:Button        text="Validate"        onclick="ValidateBtn_OnClick"
runat="server" />
</form>
</body>
</html>
```

Repeater

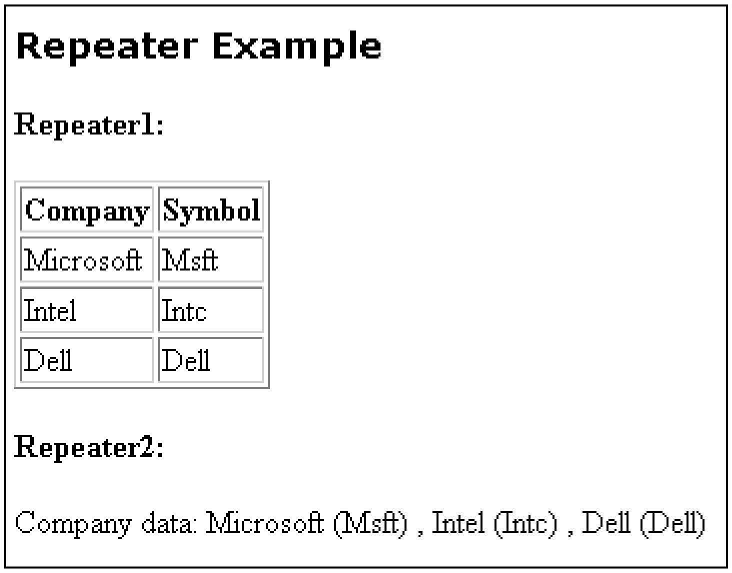
Working With Repeater

The **Repeater** control displays data items in a repeating list. Similar to DataList, the content and layout of list items in **Repeater** is defined using **templates**. At a minimum, every **Repeater** must define an **ItemTemplate**; however, the following optional templates may be used to customize the appearance of the list.

Template Name	Description
ItemTemplate	Defines the content and layout of items within the list. **Required**.
AlternatingItemTemplate	If defined, the **AlternatingItemTemplate** determines the content and layout of alternating items. If not defined, **ItemTemplate** is used.
SeparatorTemplate	If defined, the **SeparatorTemplate** is rendered between items (and alternating items). If not defined, a separator is not rendered.
HeaderTemplate	If defined, the **HeaderTemplate** determines the content and layout of the list header. If not defined, header is not rendered.
FooterTemplate	If defined, the **FooterTemplate** determines the content and layout of the list footer. If not defined, footer is not rendered.

Unlike **DataList**, **Repeater** has no built-in layout or styles. You must explicitly declare all HTML layout, formatting, and style tags within the templates of the control. For example, to create a list within an HTML table, you might declare the <table> tag in the **HeaderTemplate**, a table row (<tr> tags, <td> tags, and data-bound items) in the **ItemTemplate**, and the </table> tag in the **FooterTemplate**.

Figure 3.30 Repeater.aspx



Repeater Example

Repeater1:

Company	Symbol
Microsoft	Msft
Intel	Intc
Dell	Dell

Repeater2:

Company data: Microsoft (Msft) , Intel (Intc) , Dell (Dell)

Code for Figure 3.30 Repeater.aspx

```
<html>
<head>
    <script language="VB" runat="server">
        Sub Page_Load(sender As Object, e As EventArgs)
            If Not IsPostBack Then
                Dim values As ArrayList = New ArrayList()
                values.Add(new PositionData("Microsoft", "Msft"))
                values.Add(new PositionData("Intel", "Intc"))
                values.Add(new PositionData("Dell", "Dell"))
                Repeater1.DataSource = values
                Repeater1.DataBind
               Repeater2.DataSource = values
                Repeater2.DataBind
            End If
        End Sub
       class PositionData
            Dim m_name As String
            Dim m_ticker As String
            Public Sub New(name As String, ticker As String)
                MyBase.New
                m_name = name
                m_ticker = ticker
            End Sub
            ReadOnly Property Name As String
              Get
                Return m_name
              End Get
            End Property
            ReadOnly Property Ticker As String
```

```
          Get
            Return m_ticker
          End Get
        End Property
      End Class
    </script>
</head>
<body>
    <h3><font face="Verdana">Repeater Example</font></h3>
    <form runat=server>
      <b>Repeater1:</b>
        <p>
        <asp:Repeater id=Repeater1 runat="server">
            <HeaderTemplate>
                <table border=1>
                  <tr>
                    <td><b>Company</b></td>
                    <td><b>Symbol</b></td>
                  </tr>
            </HeaderTemplate>
            <ItemTemplate>
                <tr>
                  <td> <%# DataBinder.Eval(Container.DataItem, "Name") %> </td>
                  <td> <%# DataBinder.Eval(Container.DataItem, "Ticker") %> </td>
                </tr>
            </ItemTemplate>
            <FooterTemplate>
                </table>
            </FooterTemplate>
        </asp:Repeater>
        <p>
        <b>Repeater2:</b>
        <p>
        <asp:Repeater id=Repeater2 runat="server">
            <HeaderTemplate>
                Company data:
            </HeaderTemplate>
            <ItemTemplate>
                <%#       DataBinder.Eval(Container.DataItem,       "Name")       %>       (<%#
DataBinder.Eval(Container.DataItem, "Ticker") %>)
            </ItemTemplate>
            <SeparatorTemplate>, </SeparatorTemplate>
        </asp:Repeater>
    </form>
</body>
</html>
```

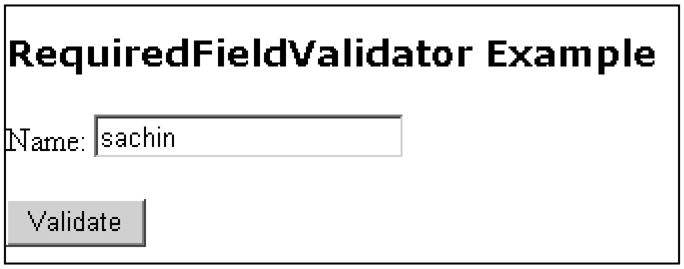
RequiredFieldValidator

For a detailed discussion of Web Forms validation, see Server Control Form Validation.

Working with RequiredFieldValidator

The **RequiredFieldValidator** control ensures that the user does not skip an entry. The control fails validation if the value it contains does not change from its initial value when validation is performed. If all the fields in the page are valid, the page is valid.

Figure 3.30 RequiredFieldValidator.aspx


RequiredFieldValidator Example

Name: sachin

Code for Figure 3.30 RequiredFieldValidator.aspx

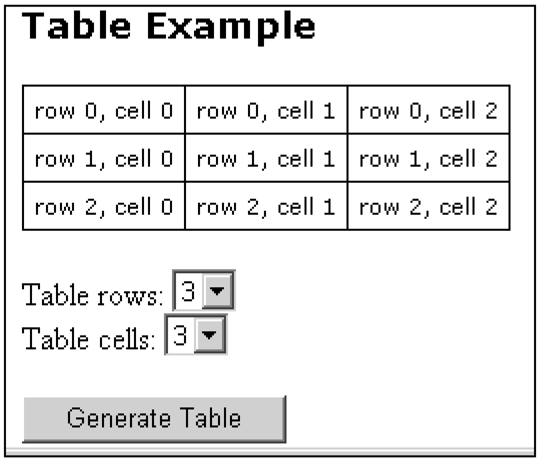
```
<html>
<body>
  <h3><font face="Verdana">RequiredFieldValidator Example</font></h3>
  <form runat=server>
     Name: <asp:TextBox id=Text1 runat="server"/>
     <asp:RequiredFieldValidator          id="RequiredFieldValidator1"
ControlToValidate="Text1"      Font-Name="Arial"      Font-Size="11"
runat="server">
        Required field!
     </asp:RequiredFieldValidator>
     <p>
     <asp:Button id="Button1" runat="server" Text="Validate" />
  </form>
</body>
</html>
```

Table, TableRow, and TableCell

Working with Table, TableRow, and TableCell

The **Table** control builds up a table programmatically by adding **TableRows** to the **Rows** collection of the table, and **TableCells** to the **Cells** collection of the row. You can add content to a table cell programmatically by adding controls to the **Controls** collection of the cell.

Figure 3.31Table.aspx



row 0, cell 0	row 0, cell 1	row 0, cell 2
row 1, cell 0	row 1, cell 1	row 1, cell 2
row 2, cell 0	row 2, cell 1	row 2, cell 2

Code for Figure 3.31Table.aspx

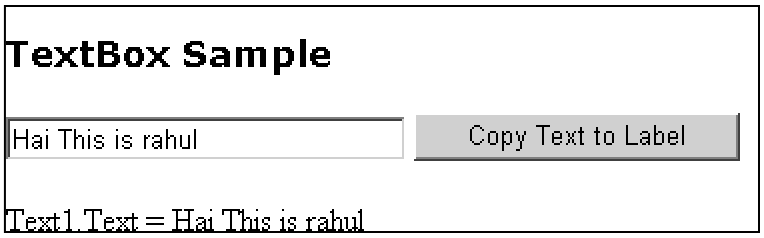
```
<html>
<head>
   <script language="VB" runat="server">
      Sub Page_Load(sender As Object, e As EventArgs)
         Dim numrows As Integer
         Dim numcells As Integer
         Dim i As Integer
         Dim j As Integer
         Dim r As TableRow
         Dim c As TableCell
         ' Generate rows and cells
         numrows = CInt(DropDown1.SelectedItem.Value)
         numcells = CInt(DropDown2.SelectedItem.Value)
         For j = 0 To numrows-1
            r = new TableRow()
            For i = 0  To numcells-1
               c = new TableCell()
               c.Controls.Add(new LiteralControl("row " & j & ", cell " & i))
               r.Cells.Add(c)
            Next i
            Table1.Rows.Add(r)
         Next j
      End Sub
   </script>
</head>
<body>
   <h3><font face="Verdana">Table Example</font></h3>
   <form runat=server>
      <asp:Table id="Table1" Font-Name="Verdana" Font-Size="8pt"
CellPadding=5 CellSpacing=0 BorderColor="black" BorderWidth="1"
Gridlines="Both" runat="server"/>
      <p>
      Table rows:
      <asp:DropDownList id=DropDown1 runat="server">
         <asp:ListItem Value="1">1</asp:ListItem>
         <asp:ListItem Value="2">2</asp:ListItem>
         <asp:ListItem Value="3">3</asp:ListItem>
         <asp:ListItem Value="4">4</asp:ListItem>
      </asp:DropDownList>
      <br>
      Table cells:
      <asp:DropDownList id=DropDown2 runat="server">
         <asp:ListItem Value="1">1</asp:ListItem>
         <asp:ListItem Value="2">2</asp:ListItem>
         <asp:ListItem Value="3">3</asp:ListItem>
         <asp:ListItem Value="4">4</asp:ListItem>
      </asp:DropDownList>
      <p>
      <asp:button Text="Generate Table" runat=server/>
   </form>
</body>
</html>
```

TextBox

Working with TextBox
The **TextBox** control enables the user to enter text. By default, the **TextMode** of **TextBox** is SingleLine, but you can modify the behavior of **TextBox** by setting the **TextMode** to Password or MultiLine.

The display width of **TextBox** is determined by its **Columns** property. If **TextMode** is MutliLine, the display height of **TextBox** is determined by the **Rows** property.

Figure 3.32 Textbox.aspx


TextBox Sample

Text1.Text = Hai This is rahul

Code for Figure 3.32 Textbox.aspx

```
<html>
<head>
   <script language="VB" runat="server">
      Sub SubmitBtn_Click(sender As Object, e As EventArgs)
         Label1.Text = "Text1.Text = " & Text1.Text
      End Sub
   </script>
</head>
<body>
   <h3><font face="Verdana">TextBox Sample</font></h3>
   <form runat="server">
    <asp:TextBox id="Text1" Text="Copy this text to the label" Width="200px" 
runat="server"/>
    <asp:Button OnClick="SubmitBtn_Click" Text="Copy Text to Label" 
Runat="server"/>
    <p>
       <asp:Label id="Label1" Text="Label1" runat="server"/>
   </form>
</body>
</html>
```

ValidationSummary

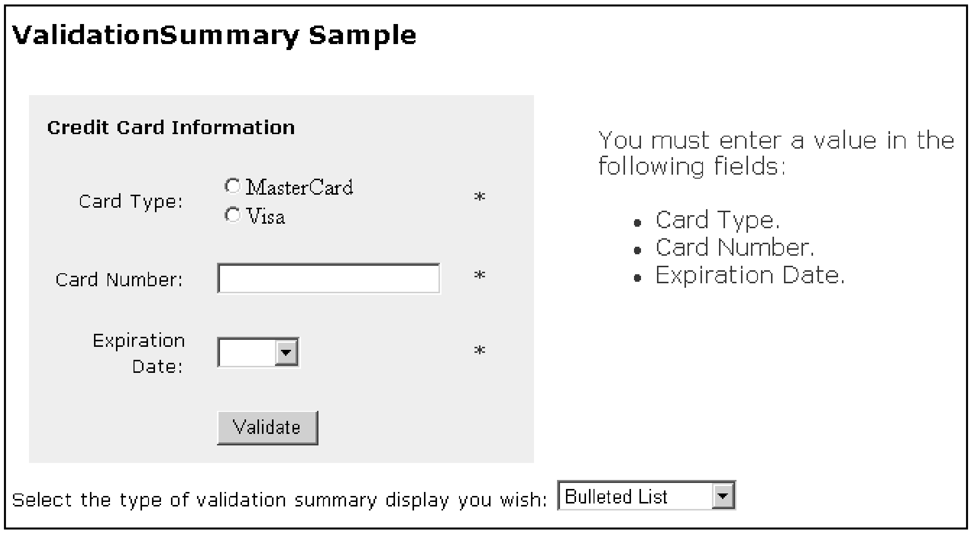
For a detailed discussion of Web Forms validation, see Server Control Form Validation.

Working with ValidationSummary

When the user's input is processed (for example, when the form is submitted), the Web Forms framework passes the user's entry to the associated validation control or controls. The validation controls test the user's input and set a property to indicate whether the entry passed the validation test. After all validation controls have been processed, the **IsValid** property on the page is set; if any of the controls shows that a validation check failed, the entire page is set to invalid.

A **ValidationSummary** control is displayed when the **IsValid** property of the page is false. It "polls" each of the validation controls on the page and aggregates the text messages exposed by each. The following sample illustrates displaying errors with a **ValidationSummary**.

Figure 3.32 ValidationSummary.aspx


ValidationSummary Sample

Credit Card Information

Card Type: ○ MasterCard ○ Visa *

Card Number: *

Expiration Date: *

Validate

You must enter a value in the following fields:

- Card Type.
- Card Number.
- Expiration Date.

Select the type of validation summary display you wish: Bulleted List

Code for Figure 3.32 ValidationSummary.aspx

```
<%@ Page clienttarget=downlevel %>
<html>
<head>
   <script language="VB" runat="server">
      Sub ListFormat_SelectedIndexChanged(sender As Object, e As EventArgs)
         ' Change display mode of the validator summary when a new option
        ' is selected from the "ListFormat" dropdownlist
         valSum.DisplayMode = ListFormat.SelectedIndex
      End Sub
  </script>
</head>
<body>
<h3><font face="Verdana">ValidationSummary Sample</font></h3>
<p>
<form runat="server">
<table cellpadding=10>
   <tr>
      <td>
         <table bgcolor="#eeeeee" cellpadding=10>
         <tr>
           <td colspan=3>
           <font face=Verdana size=2><b>Credit Card Information</b></font>
           </td>
         </tr>
         <tr>
           <td align=right>
             <font face=Verdana size=2>Card Type:</font>
```

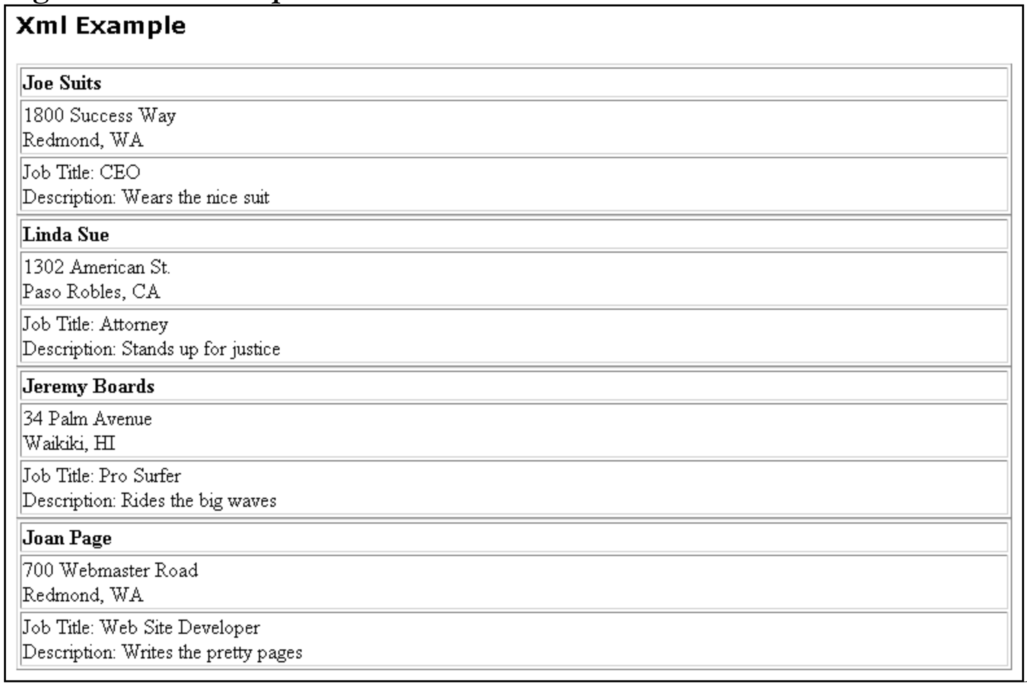
```
          </td>
          <td>
           <ASP:RadioButtonList                          id=RadioButtonList1
RepeatLayout="Flow" runat=server>
             <asp:ListItem>MasterCard</asp:ListItem>
             <asp:ListItem>Visa</asp:ListItem>
           </ASP:RadioButtonList>
          </td>
          <td align=middle rowspan=1>
           <asp:RequiredFieldValidator id="RequiredFieldValidator1"
             ControlToValidate="RadioButtonList1"
             ErrorMessage="Card Type. "
             Display="Static"
             InitialValue="" Width="100%" runat=server>
             *
           </asp:RequiredFieldValidator>
          </td>
        </tr>
        <tr>
          <td align=right>
           <font face=Verdana size=2>Card Number:</font>
          </td>
          <td>
           <ASP:TextBox id=TextBox1 runat=server />
          </td>
          <td>
           <asp:RequiredFieldValidator id="RequiredFieldValidator2"
             ControlToValidate="TextBox1"
             ErrorMessage="Card Number. "
             Display="Static"
             Width="100%" runat=server>
             *
           </asp:RequiredFieldValidator>
          </td>
        </tr>
        <tr>
          <td align=right>
           <font face=Verdana size=2>Expiration Date:</font>
          </td>
          <td>
           <ASP:DropDownList id=DropDownList1 runat=server>
             <asp:ListItem></asp:ListItem>
             <asp:ListItem >06/00</asp:ListItem>
             <asp:ListItem >07/00</asp:ListItem>
             <asp:ListItem >08/00</asp:ListItem>
             <asp:ListItem >09/00</asp:ListItem>
             <asp:ListItem >10/00</asp:ListItem>
```

```
            <asp:ListItem >11/00</asp:ListItem>
            <asp:ListItem >01/01</asp:ListItem>
            <asp:ListItem >02/01</asp:ListItem>
            <asp:ListItem >03/01</asp:ListItem>
            <asp:ListItem >04/01</asp:ListItem>
            <asp:ListItem >05/01</asp:ListItem>
            <asp:ListItem >06/01</asp:ListItem>
            <asp:ListItem >07/01</asp:ListItem>
            <asp:ListItem >08/01</asp:ListItem>
            <asp:ListItem >09/01</asp:ListItem>
            <asp:ListItem >10/01</asp:ListItem>
            <asp:ListItem >11/01</asp:ListItem>
            <asp:ListItem >12/01</asp:ListItem>
          </ASP:DropDownList>
        </td>
        <td>
          <asp:RequiredFieldValidator id="RequiredFieldValidator3"
            ControlToValidate="DropDownList1"
            ErrorMessage="Expiration Date. "
            Display="Static"
            InitialValue=""
            Width="100%"
            runat=server>
            *
          </asp:RequiredFieldValidator>
        </td>
        <td>
      </tr>
      <tr>
        <td></td>
        <td>
          <ASP:Button id=Button1 text="Validate" runat=server />
        </td>
        <td></td>
      </tr>
      </table>
    </td>
    <td valign=top>
      <table cellpadding=20><tr><td>
      <asp:ValidationSummary ID="valSum" runat="server"
        HeaderText="You must enter a value in the following fields:"
        Font-Name="verdana"
        Font-Size="12"
        />
      </td></tr></table>
    </td>
  </tr>
```

```
</table>
<font face="verdana" size="-1">Select the type of validation summary display you wish: </font>
<asp:DropDownList id="ListFormat" AutoPostBack=true
OnSelectedIndexChanged="ListFormat_SelectedIndexChanged"
runat=server >
   <asp:ListItem>List</asp:ListItem>
   <asp:ListItem selected>Bulleted List</asp:ListItem>
   <asp:ListItem>Single Paragraph</asp:ListItem>
</asp:DropDownList>
</form>
</body>
</html>
```

XML

Working with XML
The **XML** control can be used to write out an XML document or the results of an XSL Transform. The **DocumentSource** specifies the XML document to use. This document will be written directly to the output stream unless **TransformSource** is also specified. **TransformSource** must be a valid XSL Transform document and will be used to transform the XML document before its contents are written to the output stream. The following sample illustrates using a simple **XML** control.

Figure 3.33 XML.aspx

Xml Example

Joe Suits
1800 Success Way Redmond, WA
Job Title: CEO Description: Wears the nice suit
Linda Sue
1302 American St. Paso Robles, CA
Job Title: Attorney Description: Stands up for justice
Jeremy Boards
34 Palm Avenue Waikiki, HI
Job Title: Pro Surfer Description: Rides the big waves
Joan Page
700 Webmaster Road Redmond, WA
Job Title: Web Site Developer Description: Writes the pretty pages

A preloaded **XMLDocument** can be passed to the **Document** property of the **XML** control. You can also pass a preloaded **XSLTransform** to the **Transform** property of the **XML** control. The following sample illustrates creating custom **XMLDocument** and **XSLTransform** objects, then passing them into the **XML** control to be displayed.

Code for Figure 3.33 XML.aspx

```
<%@ Page Language="VB" %>
<html>
<body>
   <h3><font face="Verdana">Xml Example</font></h3>
   <form runat=server>
      <asp:Xml           id="xml1"           DocumentSource="people.xml"
TransformSource="peopletable.xsl" runat="server" />
   </form>
</body>
</html>
<%@ Page Language="VB" %>
<%@ Import Namespace="System.Xml" %>
<%@ Import Namespace="System.Xml.Xsl" %>
<html>
<script language="VB" runat="server">
   Sub Page_Load(Sender As Object, E As EventArgs)
      Dim doc As XmlDocument = New XmlDocument()
      doc.Load(Server.MapPath("people.xml"))
      Dim trans As XslTransform = new XslTransform()
      trans.Load(Server.MapPath("peopletable.xsl"))
      xml1.Document = doc
      xml1.Transform = trans
   End Sub
</script>
<body>
   <h3><font face="Verdana">Xml Example</font></h3>
   <form runat=server>
     <asp:Xml id="xml1" runat="server" />
   </form>
</body>
</html>
```

Chapter 4

Server Controls

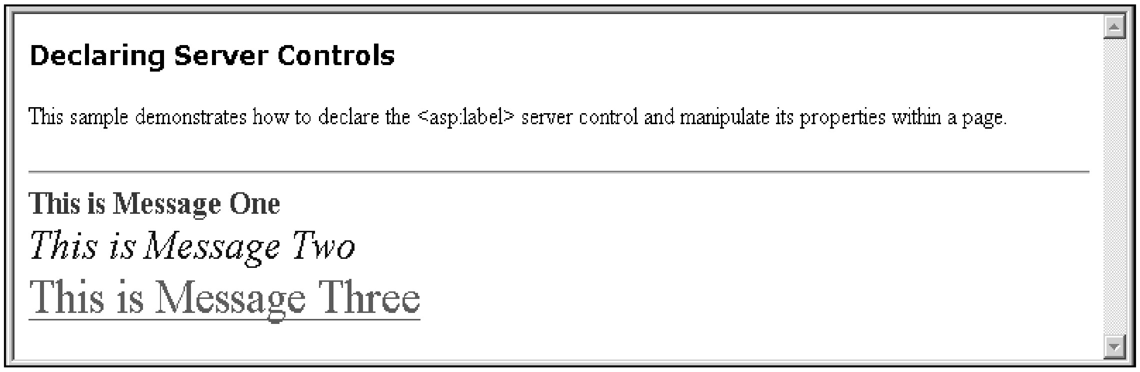
Working with Server Controls

This section of the QuickStart illustrates some common core concepts and common actions performed by end users when using ASP.NET server controls within a page.

Declaring Server Controls

ASP.NET server controls are identified within a page using declarative tags that contain a **runat="server"** attribute. The following example declares three **<asp:label runat="server">** server controls and customizes the text and style properties of each one individually.

Figure 4.1. Controls1.aspx



Declaring Server Controls

This sample demonstrates how to declare the <asp:label> server control and manipulate its properties within a page.

This is Message One

This is Message Two

This is Message Three

Code for Figure 4.1. Controls1.aspx

```
<html>
   <body>
      <h3><font face="Verdana">Declaring Server Controls</font></h3>
      This sample demonstrates how to declare the &lt;asp:label&gt; server control
and
      manipulate its properties within a page.
      <p>
      <hr>
      <asp:label id="Message1" font-size="16" font-bold="true"
forecolor="red" runat=server>This is Message One</asp:label>
      <br>
      <asp:label id="Message2" font-size="20" font-italic="true"
forecolor="blue" runat=server>This is Message Two</asp:label>
```

```
      <br>
      <asp:label    id="Message3"    font-size="24"    font-underline="true"
forecolor="green" runat=server>This is Message Three</asp:label>
   </body>
</html>
```

Manipulating Server Controls

You can programmatically identify an individual ASP.NET server control within a page by providing it with an **id** attribute. You can use this **id** reference to programmatically manipulate the server control's object model at run time. For example, the following sample demonstrates how a page developer could programmatically set an **<asp:label runat="server">** control's **Text** property within the **Page_Load** event.

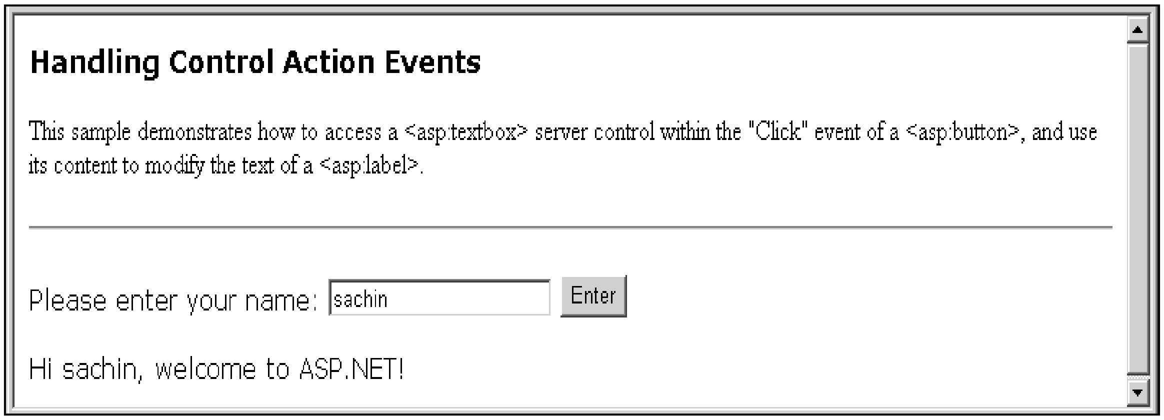
Code for Controls2.aspx

```
<html>
   <script language="VB" runat="server">
      Sub Page_Load(Sender As Object, E As EventArgs)
         Message.Text = "You last accessed this page at: " & DateTime.Now
      End Sub
   </script>
   <body>
      <h3><font face="Verdana">Manipulating Server Controls</font></h3>
      This sample demonstrates how to manipulate the &lt;asp:label&gt; server
control within
      the Page_Load event to output the current time.
      <p>
      <hr>
      <asp:label    id="Message"    font-size="24"    font-bold="true"
runat=server/>
   </body>
</html>
```

Handling Control Action Events

ASP.NET server controls can optionally expose and raise server events, which can be handled by page developers. A page developer may accomplish this by declaratively wiring an event to a control (where the attribute name of an event wireup indicates the event na
me and the attribute value indicates the name of a method to call). For example, the following code example demonstrates how to wire an **OnClick** event to a button control.

Figure 4.3. Controls3.aspx


Handling Control Action Events

This sample demonstrates how to access a <asp:textbox> server control within the "Click" event of a <asp:button>, and use its content to modify the text of a <asp:label>.

Please enter your name: sachin [Enter]

Hi sachin, welcome to ASP.NET!

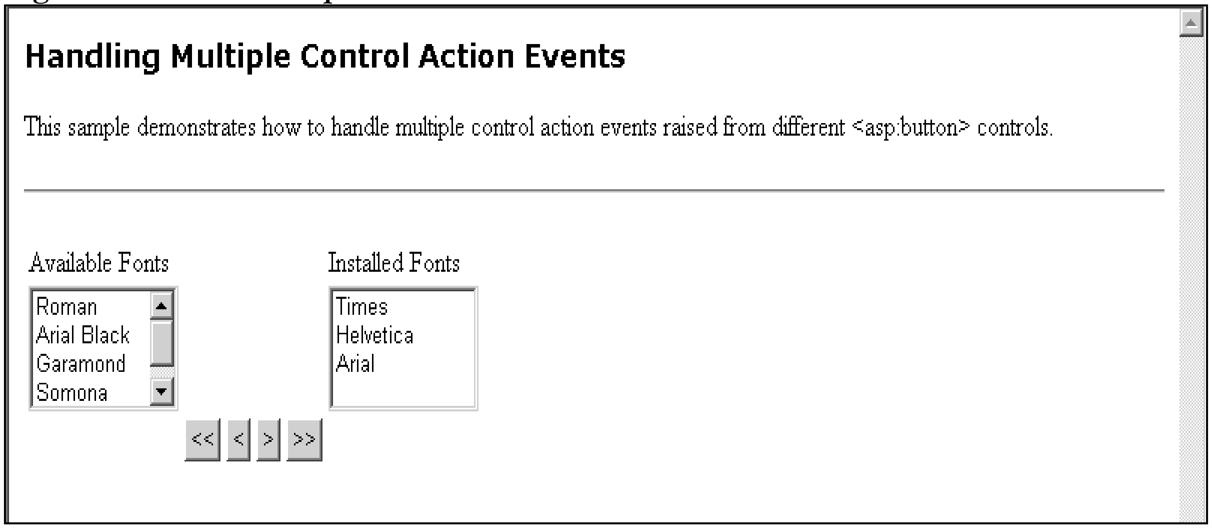
Code Figure 4.3. Controls3.aspx

```
<html>
   <script language="VB" runat="server">
      Sub EnterBtn_Click(Sender As Object, E As EventArgs)
         Message.Text = "Hi " & Name.Text & ", welcome to ASP.NET!"
      End Sub
   </script>
   <body>
      <h3><font face="Verdana">Handling Control Action
Events</font></h3>
      <p>
      This sample demonstrates how to access a &lt;asp:textbox&gt; server control
within the "Click"
      event of a &lt;asp:button&gt;, and use its content to modify the text of a
&lt;asp:label&gt;.
      <p>
      <hr>
      <form action="controls3.aspx" runat=server>
        <font face="Verdana">
          Please enter your name: <asp:textbox id="Name" runat=server/>
                         <asp:button text="Enter"
Onclick="EnterBtn_Click" runat=server/>
          <p>
          <asp:label id="Message"  runat=server/>
        </font>
      </form>
   </body>
</html>
```

Handling Multiple Control Action Events

Event handlers provide a clean way for page developers to structure logic within an ASP.NET page. For example, the following sample demonstrates how to wire and handle four button events on a single page.

Figure 4.3. Controls4.aspx



Handling Multiple Control Action Events

This sample demonstrates how to handle multiple control action events raised from different <asp:button> controls.

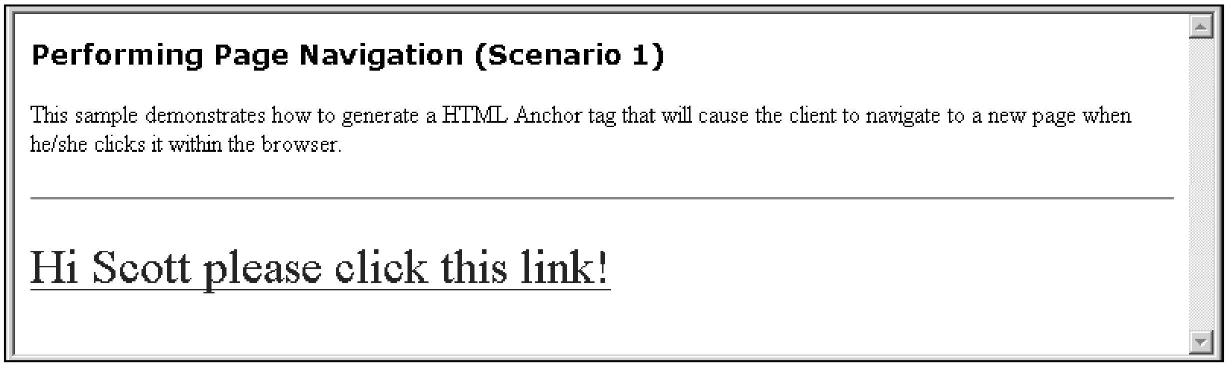
Code for Figure 4.3. Controls4.aspx

```
<html>
   <script language="VB" runat="server">
      Sub AddBtn_Click(Sender As Object, E As EventArgs)
         If Not (AvailableFonts.SelectedIndex = -1)
           InstalledFonts.Items.Add(New
ListItem(AvailableFonts.SelectedItem.Value))
           AvailableFonts.Items.Remove(AvailableFonts.SelectedItem.Value)
         End If
      End Sub
      Sub AddAllBtn_Click(Sender As Object, E As EventArgs)
         Do While Not (AvailableFonts.Items.Count = 0)
           InstalledFonts.Items.Add(New ListItem(AvailableFonts.Items(0).Value))
           AvailableFonts.Items.Remove(AvailableFonts.Items(0).Value)
         Loop
      End Sub
      Sub RemoveBtn_Click(Sender As Object, E As EventArgs)
         If Not (InstalledFonts.SelectedIndex = -1)
           AvailableFonts.Items.Add(New
ListItem(InstalledFonts.SelectedItem.Value))
           InstalledFonts.Items.Remove(InstalledFonts.SelectedItem.Value)
         End If
      End Sub
      Sub RemoveAllBtn_Click(Sender As Object, E As EventArgs)
         Do While Not (InstalledFonts.Items.Count = 0)
            AvailableFonts.Items.Add(New ListItem(InstalledFonts.Items(0).Value))
            InstalledFonts.Items.Remove(InstalledFonts.Items(0).Value)
         Loop
      End Sub
   </script>
   <body>
      <h3><font face="Verdana">Handling Multiple Control Action
Events</font></h3>
      <p>
      This sample demonstrates how to handle multiple control action events raised
from   different &lt;asp:button&gt; controls.
      <p>
      <hr>
      <form action="controls4.aspx" runat=server>
        <table>
          <tr>
            <td>
              Available Fonts
            </td>
            <td>
              <!-- Filler -->
            </td>
            <td>
              Installed Fonts
            </td>
          </tr>
          <tr>
            <td>
              <asp:listbox id="AvailableFonts" width="100px" runat=server>
                <asp:listitem>Roman</asp:listitem>
                <asp:listitem>Arial Black</asp:listitem>
                <asp:listitem>Garamond</asp:listitem>
                <asp:listitem>Somona</asp:listitem>
                <asp:listitem>Symbol</asp:listitem>
              </asp:listbox>
            </td>
            <td>
              <!-- Filler -->
            </td>
            <td>
              <asp:listbox id="InstalledFonts" width="100px" runat=server>
                <asp:listitem>Times</asp:listitem>
                <asp:listitem>Helvetica</asp:listitem>
                <asp:listitem>Arial</asp:listitem>
          </asp:listbox>
         </td>
        </tr>
        <tr>
         <td>
          <!-- Filler -->
         </td>
         <td>
          <asp:button      text="<<"      OnClick="RemoveAllBtn_Click"
runat=server/>
          <asp:button       text="<"       OnClick="RemoveBtn_Click"
runat=server/>
          <asp:button text=">" OnClick="AddBtn_Click" runat=server/>
          <asp:button       text=">>"       OnClick="AddAllBtn_Click"
runat=server/>
         </td>
         <td>
          <!-- Filler -->
         </td>
        </tr>
       </table>

     </form>

   </body>

</html>
```

Performing Page Navigation (Scenario 1)

Page navigation among multiple pages is a common scenario in virtually all Web applications. The following sample demonstrates how to use the **<asp:hyperlink runat=server>** control to navigate to another page (passing custom query string parameters along the way). The sample then demonstrates how to easily get access to these query string parameters from the target page.

Figure 4.4. Controls5.aspx


Performing Page Navigation (Scenario 1)

This sample demonstrates how to generate a HTML Anchor tag that will cause the client to navigate to a new page when he/she clicks it within the browser.

Hi Scott please click this link!

Code for Figure 4.4. Controls5.aspx

```
<html>

   <script language="VB" runat="server">

      Sub Page_Load(Sender As Object, E As EventArgs)

         Dim RandomGenerator As Random
         RandomGenerator = New Random(DateTime.Now.Millisecond)

         Dim RandomNum As Integer
         RandomNum = RandomGenerator.Next(0, 3)

         Select RandomNum

            Case 0:
              Name.Text = "Scott"

            Case 1:
              Name.Text = "Fred"

            Case 2:
              Name.Text = "Adam"

         End Select

         AnchorLink.NavigateUrl     =     "controls_navigationtarget.aspx?name="     &
System.Web.HttpUtility.UrlEncode(Name.Text)
      End Sub

   </script>

   <body>

     <h3><font    face="Verdana">Performing    Page    Navigation    (Scenario
1)</font></h3>
```

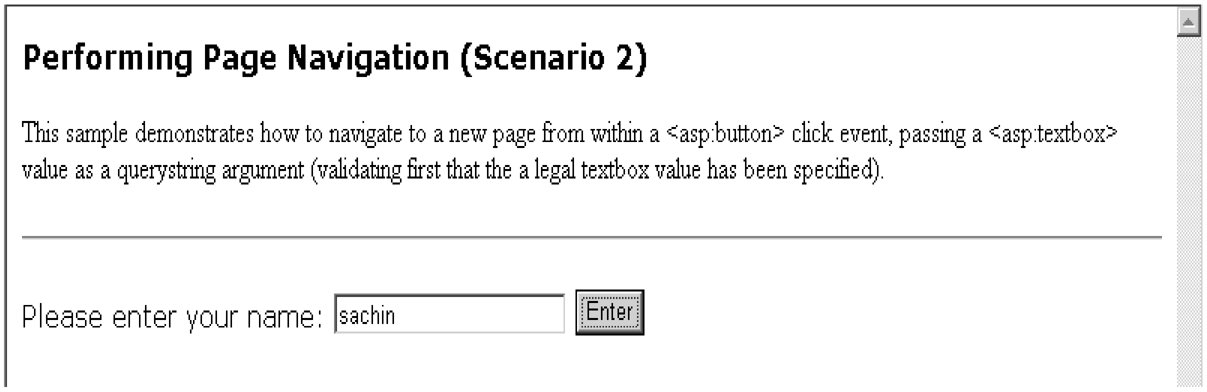
```
      <p>

      This sample demonstrates how to generate a HTML Anchor tag that will cause
the client to
      navigate to a new page when he/she clicks it within the browser.

      <p>

      <hr>

      <p>

      <asp:hyperlink id="AnchorLink" font-size=24 runat=server>
        Hi <asp:label id="Name" runat=server/> please click this link!
      </asp:hyperlink>

   </body>

</html>
```

Performing Page Navigation (Scenario 2)

Not all page navigation scenarios are initiated through hyperlinks on the client. Client-side page redirects or navigations can also be initiated from the server by an ASP.NET page developer by calling the **Response.Redirect(url)** method. This is typically done when server-side validation is required on some client input before the navigation actually takes place.

The following sample demonstrates how to use the **Response.Redirect** method to pass parameters to another target page. It also demonstrates how to easily get access to these parameters from the target page.

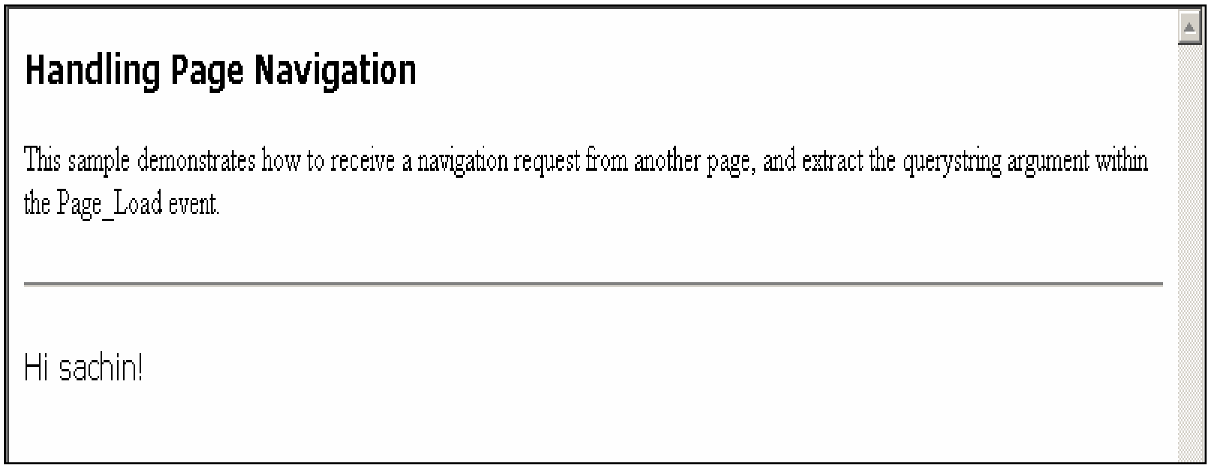
Figure 4.5. Controls6.aspx



Performing Page Navigation (Scenario 2)

This sample demonstrates how to navigate to a new page from within a <asp:button> click event, passing a <asp:textbox> value as a querystring argument (validating first that the a legal textbox value has been specified).

Please enter your name: sachin



Handling Page Navigation

This sample demonstrates how to receive a navigation request from another page, and extract the querystring argument within the Page_Load event.

Hi sachin!

Code for Figure 4.5. Controls6.aspx

```
<html>
   <script language="VB" runat="server">
      Sub EnterBtn_Click(Sender As Object, E As EventArgs)
         ' Navigate to a new page (passing name as a querystring argument) if
         ' user has entered a valid name value in the <asp:textbox>

         If Not (Name.Text = "")
           Response.Redirect("Controls_NavigationTarget.aspx?name=" &
System.Web.HttpUtility.UrlEncode(Name.Text))
         Else
           Message.Text = "Hey! Please enter your name in the textbox!"
         End If
      End Sub

   </script>

   <body>

     <h3><font face="Verdana">Performing Page Navigation (Scenario
2)</font></h3>

     <p>

     This sample demonstrates how to navigate to a new page from within a
&lt;asp:button&gt; click event,
     passing a &lt;asp:textbox&gt; value as a querystring argument (validating first
that the a legal
     textbox value has been specified).

     <p>

     <hr>
```

```
    <form action="controls6.aspx" runat=server>

      <font face="Verdana">

        Please enter your name: <asp:textbox id="Name" runat=server/>
                        <asp:button text="Enter"
Onclick="EnterBtn_Click" runat=server/>

        <p>

        <asp:label id="Message" forecolor="red" font-bold="true"
runat=server/>

      </font>

    </form>

  </body>
</html>
```

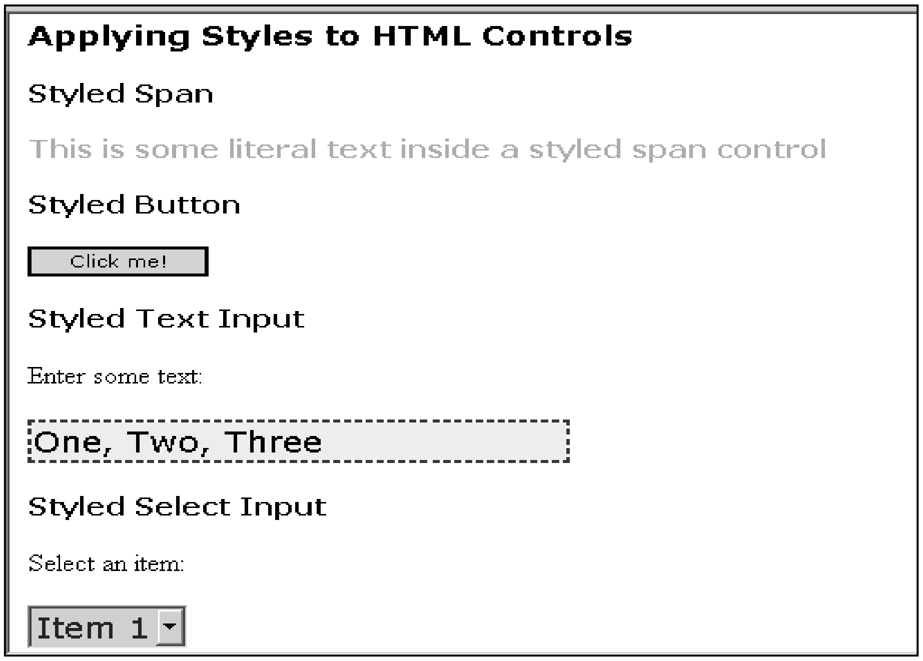
Applying Styles to Controls

The Web is a flexible environment for user interfaces, with extreme variations in the look and feel of different Web sites. The widespread adoption of cascading style sheets (CSS) is largely responsible for the rich designs encountered on the Web. All of ASP.NET's HTML server controls and Web server controls have been designed to provide first-class support for CSS styles. This section discusses how to use styles in conjunction with server controls and demonstrates the very fine control over the look and feel of your Web Forms that they provide.

Applying Styles to HTML Controls

Standard HTML tags support CSS through a style attribute, which can be set to a semicolon-delimited list of attribute/value pairs. For more information about the CSS attributes supported by the Internet Explorer browser, see MSDN Web Workshop's CSS Attributes Reference page. All of the ASP.NET HTML server controls can accept styles in exactly the same manner as standard HTML tags. The following example illustrates a number of styles applied to various HTML server controls. If you view the source code on the page returned to the client, you will see that these styles are passed along to the browser in the control's rendering.

Figure 4.6. Controls7.aspx



Applying Styles to HTML Controls

Styled Span

This is some literal text inside a styled span control

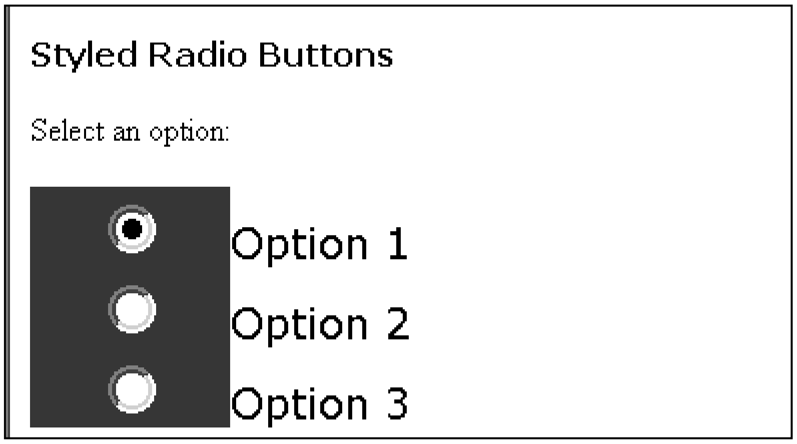
Styled Button

Styled Text Input

Enter some text:

Styled Select Input

Select an item:



Styled Radio Buttons

Select an option:

Code for Figure 4.6. Controls7.aspx

```
<html>
<body>
  <h3><font face="verdana">Applying Styles to HTML
Controls</font></h3>
  <p><font face="verdana"><h4>Styled
Span</h4></font><p>
  <span style="font: 12pt verdana; color:orange;font-weight:700"
runat="server">
      This is some literal text inside a styled span control
  </span>
  <p><font face="verdana"><h4>Styled
 Button</h4></font><p>
  <button style="font: 8pt verdana;background-
 color:lightgreen;border-color:black;width:100"
 runat="server">Click me!</button>
  <p><font face="verdana"><h4>Styled Text
 Input</h4></font><p>
  Enter some text: <p>
  <input type="text" value="One, Two, Three" style="font:
 14pt verdana;background-color:yellow;border-
 style:dashed;border-color:red;width:300;" runat="server"/>
  <p><font face="verdana"><h4>Styled Select
 Input</h4></font><p>
  Select an item: <p>
  <select style="font: 14pt verdana;background-
 color:lightblue;color:purple;" runat="server">
    <option>Item 1</option>
    <option>Item 2</option>
    <option>Item 3</option>
  </select>
  <p><font face="verdana"><h4>Styled Radio
 Buttons</h4></font><p>
  Select an option: <p>
  <span style="font: 16 pt verdana;font-weight:300">
  <input type="radio" name="Mode" checked
 style="width:50;background-color:red;zoom:200%"
 runat="server"/>Option 1<br>
  <input type="radio" name="Mode"
 style="width:50;background-color:red;zoom:200%"
 runat="server"/>Option 2<br>
  <input type="radio" name="Mode"
 style="width:50;background-color:red;zoom:200%"
 runat="server"/>Option 3
  </span>
 </body>
 </html>
```

CSS also defines a class attribute, which can be set to a CSS style definition contained in a <style>...</style> section of the document. The class attribute makes it easy to define styles once and apply them to several tags without having to redefine the style itself. Styles on HTML server controls also can be set in this manner, as demonstrated in the following sample.

Controls8.aspx

```
<html>
<head>
```

```
  <style>

      .spanstyle
      {
         font: 12pt verdana;
         font-weight:700;
         color:orange;
      }

      .buttonstyle
      {
         font: 8pt verdana;
         background-color:lightgreen;
         border-color:black;
         width:100
      }
      .inputstyle
      {
         font: 14pt verdana;
         background-color:yellow;
         border-style:dashed;
         border-color:red;
         width:300;
      }
      .selectstyle
      {
         font: 14pt verdana;
         background-color:lightblue;
         color:purple;
      }
      .radiostyle
      {
         width:50;
         background-color:red;
         zoom:200%
      }

  </style>
</head>
<body>
  <h3><font face="verdana">Applying Styles to HTML
Controls</font></h3>
  <p><font face="verdana"><h4>Styled Span</h4></font><p>
  <span class="spanstyle" runat="server">
      This is some literal text inside a styled span control
  </span>
```

```
 <p><font face="verdana"><h4>Styled Button</h4></font><p>
 <button class="buttonstyle" runat="server">Click me!</button>
 <p><font face="verdana"><h4>Styled Text Input</h4></font><p>
 Enter some text: <p>
 <input type="text" value="One, Two, Three" class="inputstyle"
runat="server"/>
 <p><font face="verdana"><h4>Styled Select Input</h4></font><p>
 Select an item: <p>
 <select class="selectstyle" runat="server">
   <option>Item 1</option>
   <option>Item 2</option>
   <option>Item 3</option>
 </select>
 <p><font face="verdana"><h4>Styled Radio Buttons</h4></font><p>
 Select an option: <p>
 <span style="font: 16 pt verdana;font-weight:300">
 <input type="radio" name="Mode" checked class="radiostyle"
runat="server"/>Option 1<br>
 <input type="radio" name="Mode" class="radiostyle"
runat="server"/>Option 2<br>
 <input type="radio" name="Mode" class="radiostyle"
runat="server"/>Option 3
 </span>
</body>
</html>
```

When an ASP.NET page is parsed, the style information is populated into a **Style** property (of type **CssStyleCollection**) on the **System.Web.UI.HtmlControls.HtmlControl** class. This property is essentially a dictionary that exposes the control's styles as a string-indexed collection of values for each style-attribute key. For example, you could use the following code to set and subsequently retrieve the **width** style attribute on an **HtmlInputText** server control.

Controls 9.aspx

```
 <script language="VB" runat="server" >
    Sub Page_Load(Sender As Object, E As EventArgs)
        MyText.Style("width") = "90px"
        Response.Write(MyText.Style("width"))
    End Sub
 </script>
 <input type="text" id="MyText" runat="server"/>
```

This next sample shows how you can programmatically manipulate the style for an HTML server control using this **Style** collection property.

Coalesce

Controls10.aspx

```
<html>
<script language="VB" runat="server">
   Sub Page_Load(Src As Object, E As EventArgs)
      Message.InnerHtml &= "<h5>Accessing Styles...</h5>"
      Message.InnerHtml &= "The color of the span is: " & MySpan.Style("color") &
"<br>"
      Message.InnerHtml &= "The width of the textbox is: " & MyText.Style("width")
& "<p>"
      Message.InnerHtml &= "MySelect's style collection is: <br>"
      Dim Keys As IEnumerator
      Keys = MySelect.Style.Keys.GetEnumerator()
      Do While (Keys.MoveNext())
         Dim Key As String
         Key = CStr(Keys.Current)
         Message.InnerHtml &= "<img
src='/quickstart/images/bullet.gif'>&nbsp;&nbsp;"
         Message.InnerHtml &= Key & "=" & MySelect.Style(Key) & "<br>"
      Loop
   End Sub
   Sub Submit_Click(Src As Object, E As EventArgs)
      Message.InnerHtml &= "<h5>Modifying Styles...</h5>"
      MySpan.Style("color") = ColorSelect.Value
      MyText.Style("width") = "600"
      Message.InnerHtml &= "The color of the span is: " & MySpan.Style("color") &
"<br>"
      Message.InnerHtml &= "The width of the textbox is: " & MyText.Style("width")
   End Sub
</script>
<body>
  <form runat="server">
     <h3><font face="verdana">Programmatically Accessing Styles</font></h3>
     <div style="font: 8pt verdana;background-color:cccccc;border-color:black;border-
width:1;border-style:solid;padding:1,10,25,10">
        <span id="Message" EnableViewState="false" runat="server"/>
        <p>
        Select a color for the span: <p>
        <select id="ColorSelect" style="font: 11pt verdana;font-weight:700;"
runat="server">
         <option>red</option>
         <option>green</option>
         <option>blue</option>
        </select>
        <input type="submit" runat="server" Value="Change Style"
OnServerClick="Submit_Click">
     </div>
```

```
    <p><font face="verdana"><h4>Styled Span</h4></font><p>
    <span id="MySpan" style="font: 12pt verdana; color:orange;font-weight:700"
runat="server">
       This is some literal text inside a styled span control
    </span>
    <p><font face="verdana"><h4>Styled Button</h4></font><p>
    <button id="MyButton" style="font: 8pt verdana;background-
color:lightgreen;border-color:black;width:100" runat="server">Click me!</button>
    <p><font face="verdana"><h4>Styled Text Input</h4></font><p>
    Enter some text: <p>
    <input id="MyText" type="text" value="One, Two, Three" style="font: 14pt
verdana;background-color:yellow;border-style:dashed;border-color:red;width:300px;"
runat="server"/>
    <p><font face="verdana"><h4>Styled Select Input</h4></font><p>
    Select an item: <p>
    <select id="MySelect" style="font: 14pt verdana;background-
color:lightblue;color:purple;" runat="server">
     <option>Item 1</option>
     <option>Item 2</option>
     <option>Item 3</option>
    </select>
    <p><font face="verdana"><h4>Styled Radio Buttons</h4></font><p>
    Select an option: <p>
    <span style="font: 16 pt verdana;font-weight:300">
       <input id="MyRadio1" type="radio" name="Mode" checked
style="width:50;background-color:red;zoom:200%" runat="server"/>Option 1<br>
       <input id="MyRadio2" type="radio" name="Mode"
style="width:50;background-color:red;zoom:200%" runat="server"/>Option 2<br>
       <input id="MyRadio3" type="radio" name="Mode"
style="width:50;background-color:red;zoom:200%" runat="server"/>Option 3
    </span>
   </form>
</body>
</html>
```

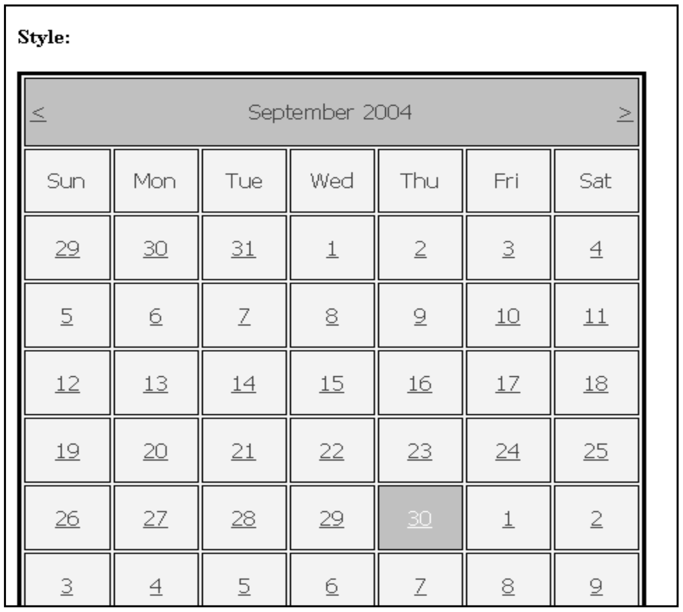
Applying Styles to Web Server Controls

Web server controls provide an additional level of support for styles by adding several strongly typed properties for commonly used style settings, such as background and foreground color, font name and size, width, height, and so on. These style properties represent a subset of the style behaviors available in HTML and are represented as "flat" properties exposed directly on the **System.Web.UI.WebControls.WebControl** base class. The advantage of using these properties is that they provide compile-time type checking and statement completion in development tools such as Microsoft Visual Studio .NET.

Coalesce

The following sample shows a **WebCalendar** control with several styles applied to it (a calendar without styles applied is included for contrast). Note that when setting a property that is a class type, such as **Font**, you need to use the subproperty syntax *PropertyName-SubPropertyName* .

Figure 4.7. Controls11.aspx



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

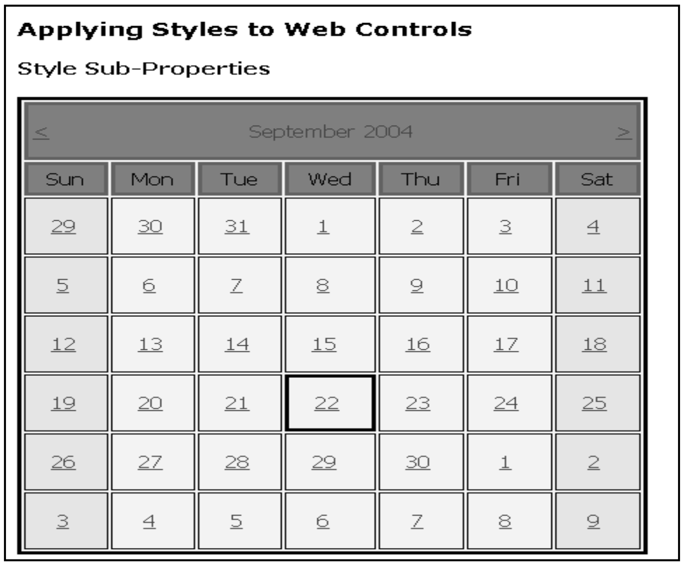
Code for Figure 4.7. Controls11.aspx

```
<html>
<body>

  <form runat="server">

     <h3><font face="verdana">Applying Styles to Web Controls</font></h3>

     <p><font face="verdana"><h4>Style Properties</h4></font><p>

     <b>No Style:</b>
     <p>
     <ASP:Calendar runat="server" />
     <p>

     <b>Style:</b>
     <p>
     <ASP:Calendar runat="server"

         BackColor="Beige"
         ForeColor="Brown"
```

```
        BorderWidth="3"
        BorderStyle="Solid"
        BorderColor="Black"
        Height="450"
        Width="450"
        Font-Size="12pt"
        Font-Name="Tahoma,Arial"
        Font-Underline="false"
        CellSpacing=2
        CellPadding=2
        ShowGridLines=true
      />

    </form>

</body>
</html>
```

The **System.Web.UI.WebControls** namespace includes a **Style** base class that encapsulates common style attributes (additional style classes, such as **TableStyle** and **TableItemStyle**, inherit from this common base class). Many Web server controls expose properties of this type for specifying the style of individual rendering elements of the control. For example, the **WebCalendar** exposes many such style properties: **DayStyle**, **WeekendDayStyle**, **TodayDayStyle**, **SelectedDayStyle**, **OtherMonthDayStyle**, and **NextPrevStyle**. You can set individual properties of these styles using the subproperty syntax *PropertyName-SubPropertyName*, as the following sample demonstrates.

Figure 4.8. Controls12.aspx


Applying Styles to Web Controls

Style Sub-Properties

<			September 2004			>
Sun	Mon	Tue	Wed	Thu	Fri	Sat
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	1	2
3	4	5	6	7	8	9

Code for Figure 4.8. Controls12.aspx

```
<html>
<body>

 <form runat="server">

   <h3><font face="verdana">Applying Styles to Web
Controls</font></h3>

   <p><font face="verdana"><h4>Style Sub-Properties</h4></font><p>

   <ASP:Calendar runat="server"

      BackColor="Beige"
      ForeColor="Brown"
      BorderWidth="3"
      BorderStyle="Solid"
      BorderColor="Black"
      Height="450"
      Width="450"
      Font-Size="12pt"
      Font-Name="Tahoma,Arial"
      Font-Underline="false"
      CellSpacing=2
      CellPadding=2
      ShowGridLines=true

      TitleStyle-BorderColor="darkolivegreen"
      TitleStyle-BorderWidth="3"
      TitleStyle-BackColor="olivedrab"
      TitleStyle-Height="50px"

      DayHeaderStyle-BorderColor="darkolivegreen"
      DayHeaderStyle-BorderWidth="3"
      DayHeaderStyle-BackColor="olivedrab"
      DayHeaderStyle-ForeColor="black"
      DayHeaderStyle-Height="20px"

      DayStyle-Width="50px"
      DayStyle-Height="50px"

      TodayDayStyle-BorderWidth="3"

      WeekEndDayStyle-BackColor="palegoldenrod"
      WeekEndDayStyle-Width="50px"
```

```
        WeekEndDayStyle-Height="50px"

        SelectedDayStyle-BorderColor="firebrick"
        SelectedDayStyle-BorderWidth="3"

        OtherMonthDayStyle-Width="50px"
        OtherMonthDayStyle-Height="50px"
      />

    </form>

</body>
</html>
```

A slightly different syntax allows each **Style** property to be declared as a child element nested within Web server control tags.

```
<ASP:Calendar ... runat="server">
   <TitleStyle BorderColor="darkolivegreen" BorderWidth="3"
         BackColor="olivedrab" Height="50px" />

</ASP:Calendar>
```

The following sample shows alternative syntax but is functionally equivalent to the preceding one.

Code for Controls13.aspx

```
<html>
<body>

  <form runat="server">

    <h3><font face="verdana">Applying Styles to Web
Controls</font></h3>

    <p><font face="verdana"><h4>Style Sub-Properties</h4></font><p>

    <ASP:Calendar id="MyCalendar" runat="server"

        BackColor="Beige"
        ForeColor="Brown"
        BorderWidth="3"
        BorderStyle="Solid"
        BorderColor="Black"
        Height="450"
        Width="450"
        Font-Size="12pt"
        Font-Name="Tahoma,Arial"
        Font-Underline="false"
        CellSpacing=2
        CellPadding=2
        ShowGridLines=true
     >

        <TitleStyle BorderColor="darkolivegreen" BorderWidth="3"
BackColor="olivedrab" Height="50px" />

        <DayHeaderStyle BorderColor="darkolivegreen" BorderWidth="3"
BackColor="olivedrab" ForeColor="black" Height="20px" />

        <WeekEndDayStyle BackColor="palegoldenrod" Width="50px"
Height="50px" />

        <DayStyle Width="50px" Height="50px" />

        <TodayDayStyle BorderWidth="3" />

        <SelectedDayStyle BorderColor="firebrick" BorderWidth="3" />

        <OtherMonthDayStyle Width="50px" Height="50px" />

     </ASP:Calendar>

   </form>

</body>
</html>
```

As with HTML server controls, you can apply styles to Web server controls using a CSS class definition. The **WebControl** base class exposes a **String** property named **CssClass** for setting the style class:

Code for Controls 14.aspx

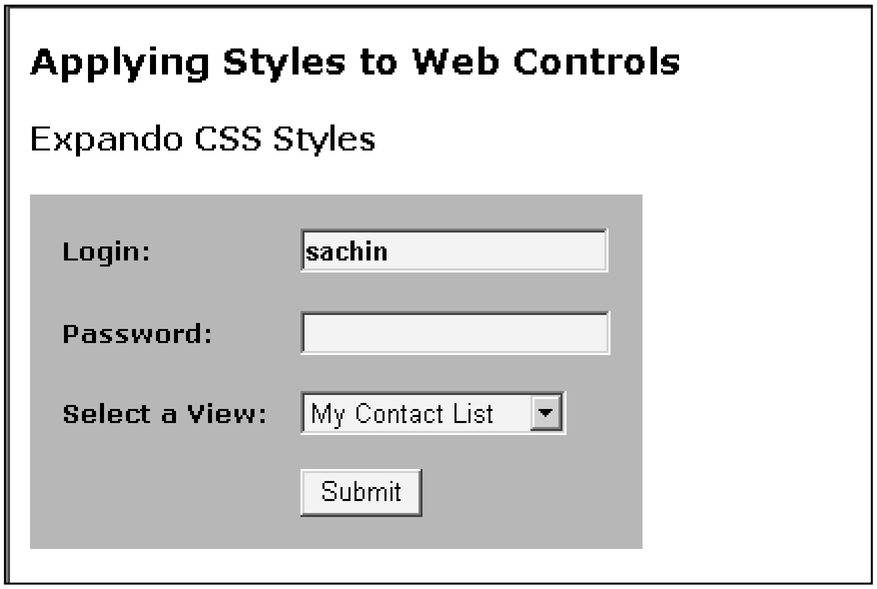
```
<html>
<head>

  <style>
   .calstyle { font-size:12pt; font-family:Tahoma,Arial; }
  </style>
```

```
</head>
<body>

 <form runat="server">

   <h3><font face="verdana">Applying Styles to Web
Controls</font></h3>

   <p><font face="verdana"><h4>The CssClass
Property</h4></font><p>

   <ASP:Calendar CssClass="calstyle" runat="server"

      BackColor="Beige"
      ForeColor="Brown"
      BorderWidth="3"
      BorderStyle="Solid"
      BorderColor="Black"
      Height="450"
      Width="450"
      CellSpacing=2
      CellPadding=2
      ShowGridLines=true

      TitleStyle-BorderColor="darkolivegreen"
      TitleStyle-BorderWidth="3"
      TitleStyle-BackColor="olivedrab"
      TitleStyle-Height="50px"

      DayHeaderStyle-BorderColor="darkolivegreen"
      DayHeaderStyle-BorderWidth="3"
      DayHeaderStyle-BackColor="olivedrab"
      DayHeaderStyle-ForeColor="black"
      DayHeaderStyle-Height="20px"

      DayStyle-Width="50px"
      DayStyle-Height="50px"

      TodayDayStyle-BorderWidth="3"

      WeekEndDayStyle-BackColor="palegoldenrod"
      WeekEndDayStyle-Width="50px"
      WeekEndDayStyle-Height="50px"

      SelectedDayStyle-BorderColor="firebrick"
      SelectedDayStyle-BorderWidth="3"
```

```
        OtherMonthDayStyle-Width="50px"
        OtherMonthDayStyle-Height="50px"
      />

   </form>

</body>

</html>
```

If an attribute is set on a server control that does not correspond to any strongly typed property on the control, the attribute and value are populated in the **Attributes** collection of the control. By default, server controls will render these attributes unmodified in the HTML returned to the requesting browser client. This means that the style and class attributes can be set on Web server controls directly instead of using the strongly typed properties. While this requires some understanding of the actual rendering of the control, it can be a flexible way to apply styles as well. It is especially useful for the standard form input controls, as illustrated in the following sample.

Figure 4.9. Controls 15.aspx



Code for Figure 4.9. Controls 15.aspx

```
<html>
<head>
  <style>
```

```
   .beige { background-color:beige }
  </style>

</head>

<body>

  <form runat="server">

      <h3><font face="verdana">Applying Styles to Web Controls</font></h3>

      <p><font face="verdana"><h4>Expando CSS Styles</h4></font><p>

      <table style="font: 10pt verdana; background-color:tan" cellspacing=15>
        <tr>
          <td><b>Login: </b></td>
          <td><ASP:TextBox runat="server" class="beige" style="font-
weight:700;"/></td>
        </tr>
        <tr>
          <td><b>Password: </b></td>
          <td><ASP:TextBox TextMode="Password" runat="server"
class="beige"/></td>
        </tr>
        <tr>
          <td><b>Select a View: </b></td>
          <td>
            <ASP:DropDownList class="beige" runat="server">
               <ASP:ListItem>Default Desktop</ASP:ListItem>
               <ASP:ListItem>My Stock Portfolio</ASP:ListItem>
               <ASP:ListItem>My Contact List</ASP:ListItem>
            </ASP:DropDownList>
          </td>
        </tr>
        <tr>
          <td>&nbsp;</td>
          <td><ASP:Button Text="Submit" runat="server"
class="beige"/></td>
        </tr>
      </table>

    </form>

</body>
</html>
```

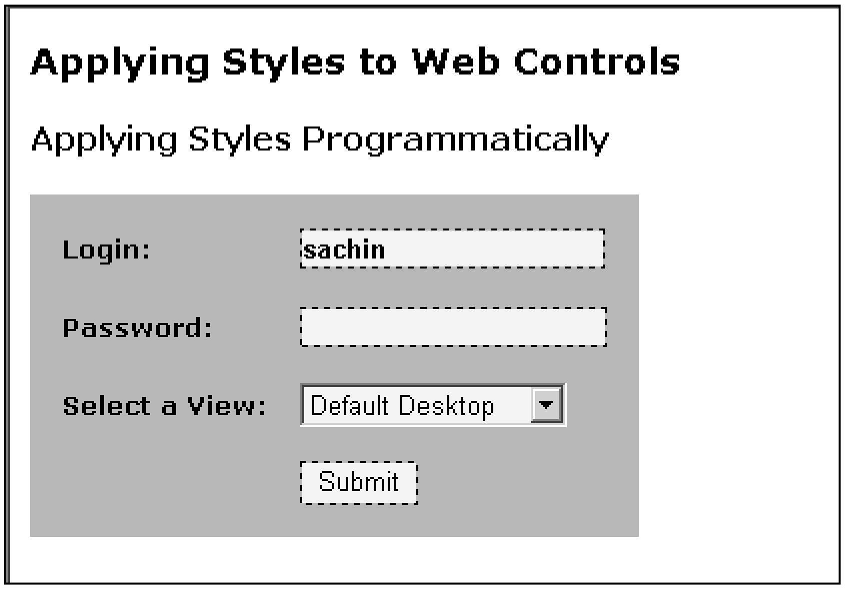
Coalesce

Web server control styles can also be set programmatically, using the **ApplyStyle** method of the base **WebControl** class, as in the following code.

Code for Controls 16.aspx

```
<script language="VB" runat="server">
   Sub Page_Load(Src As Object, E As EventArgs)
      Dim MyStyle As New Style
      MyStyle.BorderColor = Color.Black
      MyStyle.BorderStyle = BorderStyle.Dashed
      MyStyle.BorderWidth = New Unit(1)

      MyLogin.ApplyStyle (MyStyle)
      MyPassword.ApplyStyle (MyStyle)
      MySubmit.ApplyStyle (MyStyle)
   End Sub
</script>

Login: <ASP:TextBox id="MyLogin" runat="server" />/<p/>
Password:     <ASP:TextBox     id="MyPassword"     TextMode="Password"
runat="server" />
View:       <ASP:DropDownList    id="MySelect"    runat="server">       ...
</ASP:DropDownList>
```

Figure 4.10. Controls 16.aspx


Applying Styles to Web Controls

Applying Styles Programmatically

Login: sachin

Password:

Select a View: Default Desktop

Coalesce

Code for Figure 4.10. Controls 16.aspx

```
<%@ Import Namespace="System.Drawing" %>

<html>

<head>

  <style>

    .beige { background-color:beige }

  </style>

</head>

<script language="VB" runat="server">

    Sub Page_Load(Src As Object, E As EventArgs)

        Dim MyStyle As System.Web.UI.WebControls.Style

        MyStyle = New System.Web.UI.WebControls.Style()
        MyStyle.BorderColor = Color.Black
        MyStyle.BorderStyle = BorderStyle.Dashed
        MyStyle.BorderWidth = New Unit(1)

        MyLogin.ApplyStyle (MyStyle)
        MyPassword.ApplyStyle (MyStyle)
        MySubmit.ApplyStyle (MyStyle)
    End Sub

</script>

<body>

  <form runat="server">

      <h3><font face="verdana">Applying Styles to Web Controls</font></h3>

      <p><font                    face="verdana"><h4>Applying                    Styles
Programmatically</h4></font><p>

      <table style="font: 10pt verdana; background-color:tan" cellspacing=15>
        <tr>
          <td><b>Login: </b></td>
          <td><ASP:TextBox   id="MyLogin"   runat="server"   class="beige"
```

```
style="font-weight:700;"/></td>
      </tr>
      <tr>
       <td><b>Password: </b></td>
       <td><ASP:TextBox id="MyPassword" TextMode="Password"
runat="server" class="beige"/></td>
      </tr>
      <tr>
       <td><b>Select a View: </b></td>
       <td>
        <ASP:DropDownList id="MySelect" class="beige"
runat="server">
          <ASP:ListItem>Default Desktop</ASP:ListItem>
          <ASP:ListItem>My Stock Portfolio</ASP:ListItem>
          <ASP:ListItem>My Contact List</ASP:ListItem>
        </ASP:DropDownList>
       </td>
      </tr>
      <tr>
       <td>&nbsp;</td>
       <td><ASP:Button id="MySubmit" Text="Submit" runat="server"
class="beige"/></td>
      </tr>
     </table>

   </form>

</body>
</html>
```

Section Summary

1. ASP.NET's HTML server control and Web server control families provide first-class support for CSS styles.
2. Styles may be applied by setting either the style or the class attributes of a control. These settings are accessible programmatically through the control's **Attributes** collection. In the case of HTML server controls, individual values for style-attribute keys can be retrieved from the control's **Style** collection.
3. Most commonly used style settings are exposed on Web server controls as strongly typed properties of the control itself.
4. The **System.Web.UI.WebControls** namespace includes a **Style** base class that encapsulates common style attributes. Many Web server controls expose properties of this type to control individual rendering elements.

5. Styles may be set programmatically on Web server controls using the **ApplyStyle** method of the **WebControl** base class.

Server Control Form Validation

Introduction to Validation

The Web Forms framework includes a set of validation server controls that provide an easy-to-use but powerful way to check input forms for errors and, if necessary, display messages to the user.

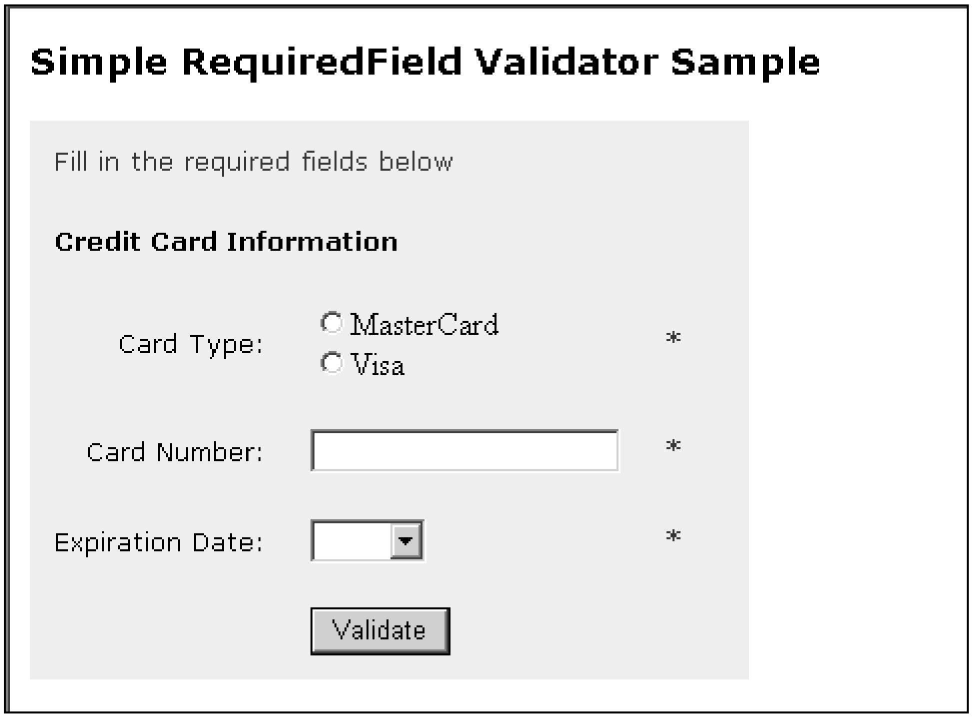
Validation controls are added to a Web Forms page like other server controls. There are controls for specific types of validation, such as range checking or pattern matching, plus a **RequiredFieldValidator** that ensures that a user does not skip an entry field. You can attach more than one validation control to an input control. For example, you might specify both that an entry is required and that it must contain a specific range of values.

Validation controls work with a limited subset of HTML and Web server controls. For each control, a specific property contains the value to be validated. The following table lists the input controls that may be validated.

Control	**Validation Property**
HtmlInputText	Value
HtmlTextArea	Value
HtmlSelect	Value
HtmlInputFile	Value
TextBox	Text
ListBox	SelectedItem.Value
DropDownList	SelectedItem.Value
RadioButtonList	SelectedItem.Value

Types of Validation Controls
The simplest form of validation is a required field. If the user enters any value in a field, it is valid. If all of the fields in the page are valid, the page is valid. The following example illustrates this using the **RequiredFieldValidator**.

Figure 4.10. Controls 17.aspx


Simple RequiredField Validator Sample

Fill in the required fields below

Credit Card Information

Code for Figure 4.10. Controls 17.aspx

```
<html>
<head>
   <script language="VB" runat="server">

      Sub ValidateBtn_Click(sender As Object, e As EventArgs)
         If (Page.IsValid) Then
            lblOutput.Text = "Page is Valid!"
         Else
            lblOutput.Text = "Some of the required fields are empty"
         End If
      End Sub

  </script>

</head>
<body>

<h3><font          face="Verdana">Simple          RequiredField          Validator
Sample</font></h3>
<p>
```

Coalesce

```
<form runat="server">

  <table bgcolor="#eeeeee" cellpadding=10>
  <tr valign="top">
    <td colspan=3>
      <asp:Label ID="lblOutput" Text="Fill in the required fields below"
ForeColor="red" Font-Name="Verdana" Font-Size="10" runat=server
/><br>
    </td>
  </tr>

  <tr>
    <td colspan=3>
    <font face=Verdana size=2><b>Credit Card Information</b></font>
    </td>
  </tr>
  <tr>
    <td align=right>
      <font face=Verdana size=2>Card Type:</font>
    </td>
    <td>
      <ASP:RadioButtonList id=RadioButtonList1 RepeatLayout="Flow"
runat=server>
          <asp:ListItem>MasterCard</asp:ListItem>
          <asp:ListItem>Visa</asp:ListItem>
      </ASP:RadioButtonList>
    </td>
    <td align=middle rowspan=1>
      <asp:RequiredFieldValidator id="RequiredFieldValidator1"
        ControlToValidate="RadioButtonList1"
        Display="Static"
        InitialValue="" Width="100%" runat=server>
        *
      </asp:RequiredFieldValidator>
    </td>
  </tr>
  <tr>
    <td align=right>
      <font face=Verdana size=2>Card Number:</font>
    </td>
    <td>
      <ASP:TextBox id=TextBox1 runat=server />
    </td>
    <td>
      <asp:RequiredFieldValidator id="RequiredFieldValidator2"
        ControlToValidate="TextBox1"
        Display="Static"
        Width="100%" runat=server>
        *
      </asp:RequiredFieldValidator>

    </td>
  </tr>
  <tr>
    <td align=right>
      <font face=Verdana size=2>Expiration Date:</font>
    </td>
    <td>
      <ASP:DropDownList id=DropDownList1 runat=server>
        <asp:ListItem></asp:ListItem>
        <asp:ListItem >06/00</asp:ListItem>
        <asp:ListItem >07/00</asp:ListItem>
        <asp:ListItem >08/00</asp:ListItem>
        <asp:ListItem >09/00</asp:ListItem>
        <asp:ListItem >10/00</asp:ListItem>
        <asp:ListItem >11/00</asp:ListItem>
        <asp:ListItem >01/01</asp:ListItem>
        <asp:ListItem >02/01</asp:ListItem>
        <asp:ListItem >03/01</asp:ListItem>
        <asp:ListItem >04/01</asp:ListItem>
        <asp:ListItem >05/01</asp:ListItem>
        <asp:ListItem >06/01</asp:ListItem>
        <asp:ListItem >07/01</asp:ListItem>
        <asp:ListItem >08/01</asp:ListItem>
        <asp:ListItem >09/01</asp:ListItem>
        <asp:ListItem >10/01</asp:ListItem>
        <asp:ListItem >11/01</asp:ListItem>
        <asp:ListItem >12/01</asp:ListItem>
      </ASP:DropDownList>
    </td>
    <td>
      <asp:RequiredFieldValidator id="RequiredFieldValidator3"
       ControlToValidate="DropDownList1"
       Display="Static"
       InitialValue="" Width="100%" runat=server>
       *
      </asp:RequiredFieldValidator>
    </td>
    <td>
  </tr>
  <tr>
    <td></td>
    <td>
```

```
     <ASP:Button                 id=Button1                 text="Validate"
OnClick="ValidateBtn_Click" runat=server />
    </td>
    <td></td>
   </tr>
   </table>
</form>
</body>
</html>
```

There are also validation controls for specific types of validation, such as range checking or pattern matching. The following table lists the validation controls.

Control Name	**Description**
RequiredFieldValidator	Ensures that the user does not skip an entry.
CompareValidator	Compares a user's entry with a constant value or a property value of another control using a comparison operator (less than, equal to, greater than, and so on).
RangeValidator	Checks that a user's entry is between specified lower and upper boundaries. You can check ranges within pairs of numbers, alphabetic characters, or dates. Boundaries can be expressed as constants.
RegularExpressionValidator	Checks that the entry matches a pattern defined by a regular expression. This type of validation allows you to check for predictable sequences of characters, such as those in social security numbers, e-mail addresses, telephone numbers, postal codes, and so on.
CustomValidator	Checks the user's entry using validation logic that you code yourself. This type of validation allows you to check for values derived at run time.
ValidationSummary	Displays the validation errors in summary form for all of the

validators on a page.

Client-Side Validation

The validation controls always perform validation checking in server code. However, if the user is working with a browser that supports DHTML, the validation controls can also perform validation using client script. With client-side validation, any errors are detected on the client when the form is submitted to the server. If any of the validators are found to be in error, the submission of the form to the server is cancelled and the validator's **Text** property is displayed. This permits the user to correct the input before submitting the form to the server. Field values are revalidated as soon as the field containing the error loses focus, thus providing the user with a rich, interactive validation experience.

Note that the Web Forms page framework always performs validation on the server, even if the validation has already been performed on the client. This helps prevent users from being able to bypass validation by impersonating another user or a preapproved transaction.

Client-side validation is enabled by default. If the client is capable, uplevel validation will be performed automatically. To disable client-side validation, set the page's **ClientTarget** property to "Downlevel" ("Uplevel" forces client-side validation).

Controls 18.aspx

```
<%@ Page ClientTarget=UpLevel %>

<html>
<head>

  <script language="VB" runat="server">

    Sub Page_Load
      If Not IsPostBack
        'Validate intially to force *s to appear before the first round-trip
        Validate()
      End If
    End Sub

    Sub ValidateBtn_Click(sender As Object, e As EventArgs)
      If (Page.IsValid) Then
        lblOutput.Text = "Page is Valid!"
      Else
        lblOutput.Text = "Some of the required fields are empty"
      End If
    End Sub

  </script>

</head>
<body>

<h3><font face="Verdana">Client-Side RequiredFieldValidator
Sample</font></h3>
<p>
<form runat="server">
   <table bgcolor="#eeeeee" cellpadding=10>
   <tr valign="top">
     <td colspan=3>
      <asp:Label ID="lblOutput" Name="lblOutput" Text="Fill in the
required fields below" ForeColor="red" Font-Name="Verdana" Font-
Size="10" runat=server /><br>
     </td>
   </tr>
   <tr>
     <td colspan=3>
     <font face=Verdana size=2><b>Credit Card Information</b></font>
     </td>
   </tr>
   <tr>
     <td align=right>
      <font face=Verdana size=2>Card Type:</font>
     </td>
     <td>
      <ASP:RadioButtonList id=RadioButtonList1 RepeatLayout="Flow"
onclick="ClientOnChange();" runat=server>
         <asp:ListItem>MasterCard</asp:ListItem>
         <asp:ListItem>Visa</asp:ListItem>
      </ASP:RadioButtonList>
     </td>
     <td align=middle rowspan=1>
      <asp:RequiredFieldValidator id="RequiredFieldValidator1"
runat="server"
         ControlToValidate="RadioButtonList1"
         ErrorMessage="*"
         Display="Static"
         InitialValue=""
         Width="100%">
      </asp:RequiredFieldValidator>
     </td>
   </tr>
   <tr>
     <td align=right>
      <font face=Verdana size=2>Card Number:</font>
     </td>
     <td>
      <ASP:TextBox     id=TextBox1     onchange="ClientOnChange();"
runat=server />
     </td>
     <td>
      <asp:RequiredFieldValidator            id="RequiredFieldValidator2"
runat="server"
         ControlToValidate="TextBox1"
         ErrorMessage="*"
         Display="Static"
         Width="100%">
      </asp:RequiredFieldValidator>
     </td>
   </tr>
   <tr>
     <td align=right>
      <font face=Verdana size=2>Expiration Date:</font>
     </td>
     <td>
      <ASP:DropDownList                                   id=DropDownList1
onchange="ClientOnChange();" runat=server>
         <asp:ListItem></asp:ListItem>
         <asp:ListItem >06/00</asp:ListItem>
         <asp:ListItem >07/00</asp:ListItem>
         <asp:ListItem >08/00</asp:ListItem>
         <asp:ListItem >09/00</asp:ListItem>
         <asp:ListItem >10/00</asp:ListItem>
         <asp:ListItem >11/00</asp:ListItem>
         <asp:ListItem >01/01</asp:ListItem>
         <asp:ListItem >02/01</asp:ListItem>
         <asp:ListItem >03/01</asp:ListItem>
         <asp:ListItem >04/01</asp:ListItem>
         <asp:ListItem >05/01</asp:ListItem>
         <asp:ListItem >06/01</asp:ListItem>
         <asp:ListItem >07/01</asp:ListItem>
         <asp:ListItem >08/01</asp:ListItem>
         <asp:ListItem >09/01</asp:ListItem>
         <asp:ListItem >10/01</asp:ListItem>
         <asp:ListItem >11/01</asp:ListItem>
         <asp:ListItem >12/01</asp:ListItem>
      </ASP:DropDownList>
     </td>
     <td>
```

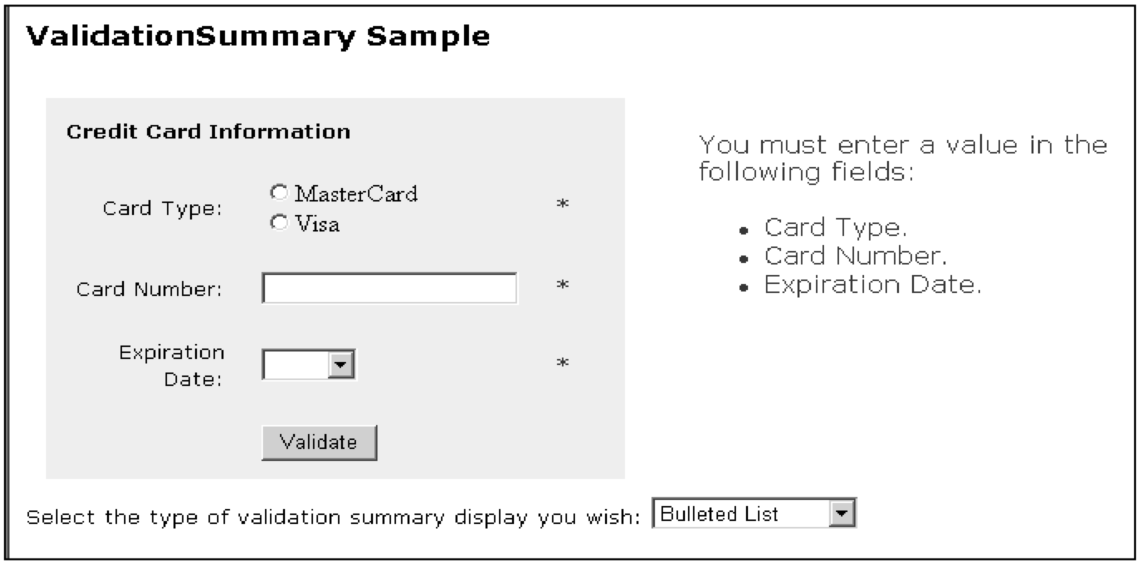
```
     <asp:RequiredFieldValidator              id="RequiredFieldValidator3"
runat="server"
       ControlToValidate="DropDownList1"
       ErrorMessage="*"
       Display="Static"
       InitialValue=""
       Width="100%">
     </asp:RequiredFieldValidator>
    </td>
    <td>
   </tr>
   <tr>
    <td></td>
    <td>
     <ASP:Button              id=Button1              text="Validate"
OnClick="ValidateBtn_Click" runat="server" />
    </td>
    <td></td>
   </tr>
   </table>
</form>
<script language=javascript>
<!--
   function ClientOnChange() {
     if (typeof(Page_Validators) == "undefined")
       return;
     document.all["lblOutput"].innerText = Page_IsValid ? "Page is Valid!" :
"Some of the required fields are empty";
   }
// -->
</script>
</body>
</html>
```

Displaying Validation Errors

When the user's input is processed (for example, when the form is submitted), the Web Forms page framework passes the user's entry to the associated validation control or controls. The validation controls test the user's input and set a property to indicate whether the entry passed the validation test. After all validation controls have been processed, the **IsValid** property on the page is set; if any of the controls shows that a validation check failed, the entire page is set to invalid.

If a validation control is in error, an error message may be displayed in the page by that validation control or in a **ValidationSummary** control elsewhere on the page. The **ValidationSummary** control is displayed when the **IsValid** property of the page is false. It polls each of the validation controls on the page and aggregates the text messages exposed by each. The following example illustrates displaying errors with a **ValidationSummary** control.

Figure 4.11. Controls 19.aspx



ValidationSummary Sample

Credit Card Information

Card Type: ○ MasterCard ○ Visa *

Card Number: *

Expiration Date: *

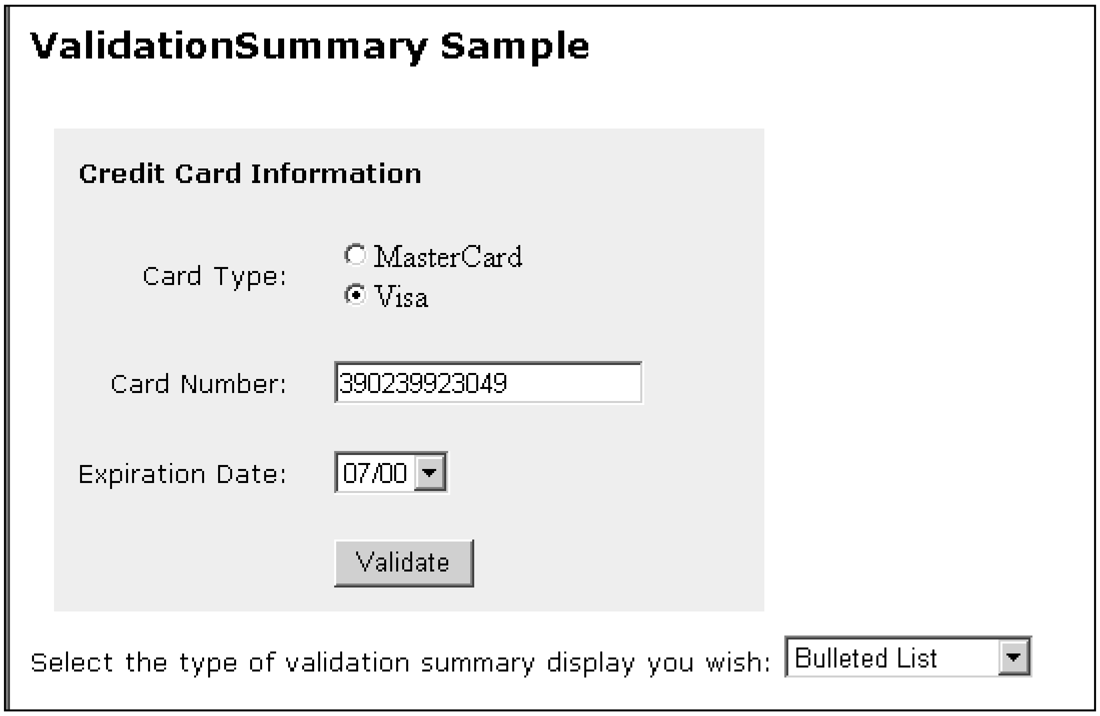
Validate

You must enter a value in the following fields:

- Card Type.
- Card Number.
- Expiration Date.

Select the type of validation summary display you wish: Bulleted List

Figure 4.12. Controls 20.aspx



ValidationSummary Sample

Credit Card Information

Card Type: ○ MasterCard [✓ Visa]

Card Number: 390239923049

Expiration Date: 07/00

Validate

Select the type of validation summary display you wish: Bulleted List

Code for Figure 4.12. Controls 21.aspx

```
<%@ Page clienttarget=downlevel %>

<html>
<head>
   <script language="VB" runat="server">

      Sub ListFormat_SelectedIndexChanged(sender As Object, e As EventArgs)

         ' Change display mode of the validator summary when a new option
         ' is selected from the "ListFormat" dropdownlist

         valSum.DisplayMode = ListFormat.SelectedIndex
      End Sub

   </script>

</head>
<body>

<h3><font face="Verdana">ValidationSummary Sample</font></h3>
<p>

<form runat="server">

<table cellpadding=10>
   <tr>
      <td>
         <table bgcolor="#eeeeee" cellpadding=10>

         <tr>
           <td colspan=3>
           <font face=Verdana size=2><b>Credit Card Information</b></font>
           </td>
         </tr>
         <tr>
           <td align=right>
             <font face=Verdana size=2>Card Type:</font>
           </td>
           <td>
             <ASP:RadioButtonList                        id=RadioButtonList1
RepeatLayout="Flow" runat=server>
                <asp:ListItem>MasterCard</asp:ListItem>
                <asp:ListItem>Visa</asp:ListItem>
         </ASP:RadioButtonList>
        </td>
        <td align=middle rowspan=1>
         <asp:RequiredFieldValidator id="RequiredFieldValidator1"
            ControlToValidate="RadioButtonList1"
            ErrorMessage="Card Type. "
            Display="Static"
            InitialValue="" Width="100%" runat=server>
            *
         </asp:RequiredFieldValidator>
        </td>
       </tr>
       <tr>
        <td align=right>
         <font face=Verdana size=2>Card Number:</font>
        </td>
        <td>
         <ASP:TextBox id=TextBox1 runat=server />
        </td>
        <td>
         <asp:RequiredFieldValidator id="RequiredFieldValidator2"
            ControlToValidate="TextBox1"
            ErrorMessage="Card Number. "
            Display="Static"
            Width="100%" runat=server>
            *
         </asp:RequiredFieldValidator>

        </td>
       </tr>
       <tr>
        <td align=right>
         <font face=Verdana size=2>Expiration Date:</font>
        </td>
        <td>
         <ASP:DropDownList id=DropDownList1 runat=server>
            <asp:ListItem></asp:ListItem>
            <asp:ListItem >06/00</asp:ListItem>
            <asp:ListItem >07/00</asp:ListItem>
            <asp:ListItem >08/00</asp:ListItem>
            <asp:ListItem >09/00</asp:ListItem>
            <asp:ListItem >10/00</asp:ListItem>
            <asp:ListItem >11/00</asp:ListItem>
            <asp:ListItem >01/01</asp:ListItem>
            <asp:ListItem >02/01</asp:ListItem>
            <asp:ListItem >03/01</asp:ListItem>
            <asp:ListItem >04/01</asp:ListItem>
            <asp:ListItem >05/01</asp:ListItem>
            <asp:ListItem >06/01</asp:ListItem>
            <asp:ListItem >07/01</asp:ListItem>
            <asp:ListItem >08/01</asp:ListItem>
            <asp:ListItem >09/01</asp:ListItem>
            <asp:ListItem >10/01</asp:ListItem>
            <asp:ListItem >11/01</asp:ListItem>
            <asp:ListItem >12/01</asp:ListItem>
          </ASP:DropDownList>
         </td>
         <td>
          <asp:RequiredFieldValidator id="RequiredFieldValidator3"
           ControlToValidate="DropDownList1"
           ErrorMessage="Expiration Date. "
           Display="Static"
           InitialValue=""
           Width="100%"
           runat=server>
           *
          </asp:RequiredFieldValidator>
         </td>
         <td>
        </tr>
        <tr>
         <td></td>
         <td>
          <ASP:Button id=Button1 text="Validate" runat=server />
         </td>
         <td></td>
        </tr>
        </table>
      </td>
      <td valign=top>
        <table cellpadding=20><tr><td>
        <asp:ValidationSummary ID="valSum" runat="server"
          HeaderText="You must enter a value in the following fields:"
          Font-Name="verdana"
          Font-Size="12"
          />
        </td></tr></table>
      </td>
   </tr>
</table>
<font face="verdana" size="-1">Select the type of validation summary display you wish: </font>
<asp:DropDownList id="ListFormat" AutoPostBack=true
OnSelectedIndexChanged="ListFormat_SelectedIndexChanged"
```

```
runat=server >
   <asp:ListItem>List</asp:ListItem>
   <asp:ListItem selected>Bulleted List</asp:ListItem>
   <asp:ListItem>Single Paragraph</asp:ListItem>
</asp:DropDownList>
</form>
</body>
</html>
```

Working with CompareValidator

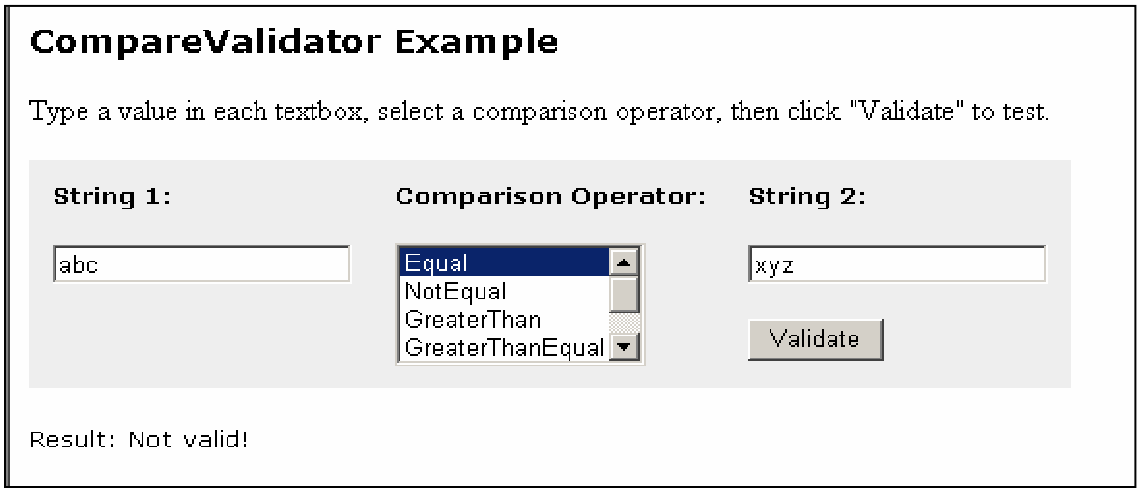
The **CompareValidator** server control compares the values of two controls. **CompareValidator** uses three key properties to perform its validation. **ControlToValidate** and **ControlToCompare** contain the values to compare. **Operator** defines the type of comparison to perform--for example, Equal or Not Equal. **CompareValidator** performs the validation by evaluating these properties as an expression, as follows:

(ControlToValidate ControlToCompare)
If the expression evaluates true, the validation result is valid.

The **CompareValidator** server control could also be used to do Datatype validation.For example, if birth date information has to be collected from a user registration page, **CompareValidator** control could be used to make sure that the date is in a recognized format before it is submitted to the database.

The following sample shows how to use the **CompareValidator** control.

Figure 4.13. Controls 21.aspx


CompareValidator Example

Type a value in each textbox, select a comparison operator, then click "Validate" to test.

String 1: abc

Comparison Operator: Equal (selected), NotEqual, GreaterThan, GreaterThanEqual

String 2: xyz

Result: Not valid!

Figure 4.14. Controls 22.aspx

```
<%@ Page clienttarget=downlevel %>

<html>
<head>
   <script language="VB" runat="server">

      Sub Button1_OnSubmit(sender As Object, e As EventArgs)
        If (Page.IsValid) Then
          lblOutput.Text = "Result: Valid!"
        Else
          lblOutput.Text = "Result: Not valid!"
        End If
      End Sub

      Sub lstOperator_SelectedIndexChanged(sender As Object, e As EventArgs)
         comp1.Operator = lstOperator.SelectedIndex
         comp1.Validate
      End Sub

  </script>

</head>
<body>

   <h3><font face="Verdana">CompareValidator Example</font></h3>
   <p>Type a value in each textbox, select a comparison operator, then click
"Validate" to test.</p>

   <form runat=server>

     <table bgcolor="#eeeeee" cellpadding=10>
     <tr valign="top">
       <td>
          <h5><font face="Verdana">String 1:</font></h5>
          <asp:TextBox id="txtComp" runat="server"></asp:TextBox>
       </td>
       <td>
          <h5><font face="Verdana">Comparison Operator:</font></h5>

          <asp:ListBox                                    id="lstOperator"
```

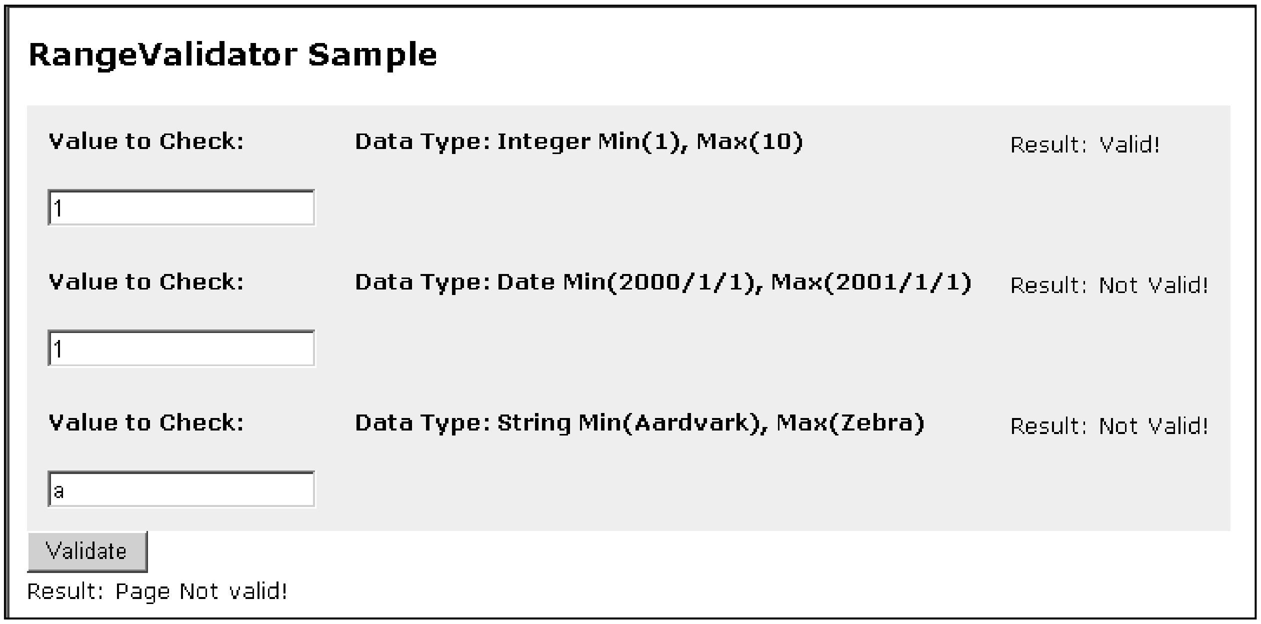
```
OnSelectedIndexChanged="lstOperator_SelectedIndexChanged"
runat="server">
            <asp:ListItem Selected Value="Equal" >Equal</asp:ListItem>
            <asp:ListItem Value="NotEqual" >NotEqual</asp:ListItem>
            <asp:ListItem                    Value="GreaterThan"
>GreaterThan</asp:ListItem>
            <asp:ListItem                 Value="GreaterThanEqual"
>GreaterThanEqual</asp:ListItem>
            <asp:ListItem Value="LessThan" >LessThan</asp:ListItem>
            <asp:ListItem                   Value="LessThanEqual"
>LessThanEqual</asp:ListItem>
        </asp:ListBox>
      </td>
      <td>
        <h5><font face="Verdana">String 2:</font></h5>
        <asp:TextBox                             id="txtCompTo"
runat="server"></asp:TextBox><p>
        <asp:Button    runat=server    Text="Validate"    ID="Button1"
onclick="Button1_OnSubmit" />
      </td>
    </tr>
    </table>

    <asp:CompareValidator   id="comp1"   ControlToValidate="txtComp"
ControlToCompare = "txtCompTo" Type="String" runat="server"/>

    <br>

    <asp:Label  ID="lblOutput"  Font-Name="verdana"  Font-Size="10pt"
runat="server"/>

  </form>

</body>
</html>
```

Working with RangeValidator

The **RangeValidator** server control tests whether an input value falls within a given range. **RangeValidator** uses three key properties to perform its validation. **ControlToValidate** contains the value to validate. **MinimumValue** and **MaximumValue** define the minimum and maximum values of the valid range.

This sample shows how to use the **RangeValidator** control.

Figure 4.14. Controls 22.aspx



RangeValidator Sample

Value to Check: 1 — Data Type: Integer Min(1), Max(10) — Result: Valid!

Value to Check: 1 — Data Type: Date Min(2000/1/1), Max(2001/1/1) — Result: Not Valid!

Value to Check: a — Data Type: String Min(Aardvark), Max(Zebra) — Result: Not Valid!

Result: Page Not valid!

Code for Figure 4.14. Controls 22.aspx

```
<%@ Page clienttarget=downlevel %>

<html>
<head>
   <script language="VB" runat="server">

      Sub Button1_Click(sender As Object, e As EventArgs)
         rangeValInteger.Validate()
         If (rangeValInteger.IsValid) Then
            lblOutput1.Text = "Result: Valid!"
         Else
            lblOutput1.Text = "Result: Not Valid!"
         End If

         rangeValDate.Validate()
         If (rangeValDate.IsValid) Then
            lblOutput2.Text = "Result: Valid!"
         Else
            lblOutput2.Text = "Result: Not Valid!"
         End If

         rangeValString.Validate()
```

```
        If (rangeValString.IsValid) Then
           lblOutput3.Text = "Result: Valid!"
        Else
           lblOutput3.Text = "Result: Not Valid!"
        End If

        If (Page.IsValid) Then
          lblOutput.Text = "Result: Page Valid!"
        Else
          lblOutput.Text = "Result: Page Not valid!"
        End If
      End Sub

   </script>

</head>
<body>

   <h3><font face="Verdana">RangeValidator Sample</font></h3>
   <p>

   <form runat="server">

     <table bgcolor="#eeeeee" cellpadding=10>
     <tr valign="top">
      <td>
         <h5><font face="Verdana">Value to Check:</font></h5>
         <asp:TextBox id="txtComp1" runat="server"/>
      </td>
      <td>
         <h5><font face="Verdana">Data Type: Integer Min(1), Max(10)</font></h5>
      </td>
      <td>
          <asp:Label id="lblOutput1" Font-Name="verdana" Font-Size="10pt"
runat="server" />
      </td>
     </tr>
     <tr valign="top">
      <td>
         <h5><font face="Verdana">Value to Check:</font></h5>
         <asp:TextBox id="txtComp2" runat="server"/>
      </td>
      <td>
         <h5><font face="Verdana">Data Type: Date Min(2000/1/1),
Max(2001/1/1)</font></h5>
      </td>
      <td>
```

```
        <asp:Label    id="lblOutput2"    Font-Name="verdana"    Font-Size="10pt"
runat="server" />
      </td>
    </tr>
    <tr valign="top">
      <td>
        <h5><font face="Verdana">Value to Check:</font></h5>
        <asp:TextBox id="txtComp3" runat="server"/>
      </td>
      <td>
        <h5><font      face="Verdana">Data      Type:      String      Min(Aardvark),
Max(Zebra)</font></h5>
      </td>
      <td>
        <asp:Label    id="lblOutput3"    Font-Name="verdana"    Font-Size="10pt"
runat="server" />
      </td>
    </tr>
   </table>

   <asp:Button     Text="Validate"     ID="Button1"     onclick="Button1_Click"
runat="server" />

   <asp:RangeValidator
     id="rangeValInteger"
     Type="Integer"
     ControlToValidate="txtComp1"
     MaximumValue="10"
     MinimumValue="1"
     runat="server"/>

   <asp:RangeValidator
     id="rangeValDate"
     Type="Date"
     ControlToValidate="txtComp2"
     MaximumValue="2001/1/1"
     MinimumValue="2000/1/1"
     runat="server"/>

   <asp:RangeValidator
     id="rangeValString"
     Type="String"
     ControlToValidate="txtComp3"
     MaximumValue="Zebra"
     MinimumValue="Aardvark"
     runat="server"/>
   <br>
```

```
    <asp:Label      id="lblOutput"      Font-Name="verdana"      Font-Size="10pt"
runat="server" />

   </form>

</body>
</html>
```

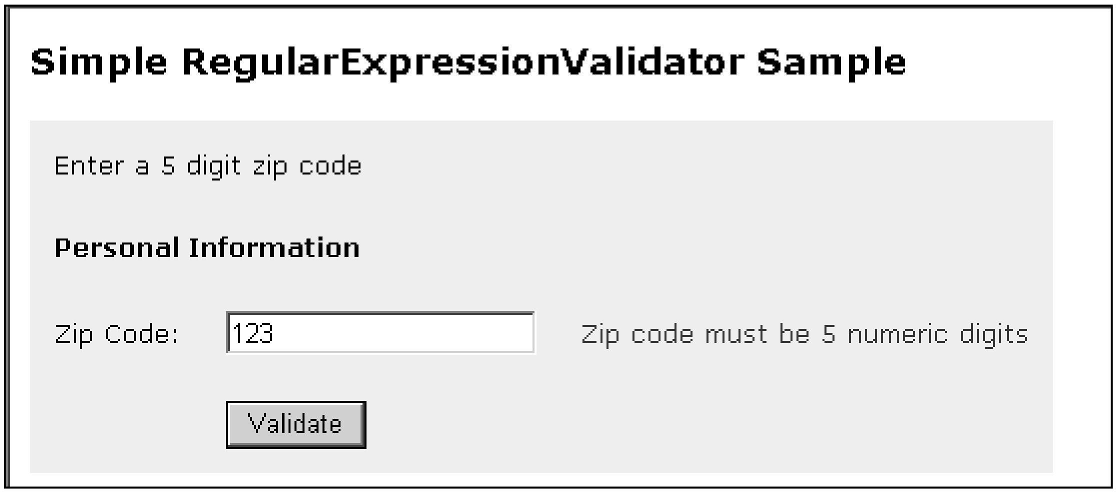
Working with Regular Expressions

The **RegularExpressionValidator** server control checks that the entry matches a pattern defined by a regular expression. This type of validation allows you to check for predictable sequences of characters, such as those in social security numbers, e-mail addresses, telephone numbers, postal codes, and so on.

RegularExpressionValidator uses two key properties to perform its validation. **ControlToValidate** contains the value to validate. **ValidationExpression** contains the regular expression to match.

These samples illustrates using the RegularExpressionValidator control.

Figure 4.15. Controls 23.aspx



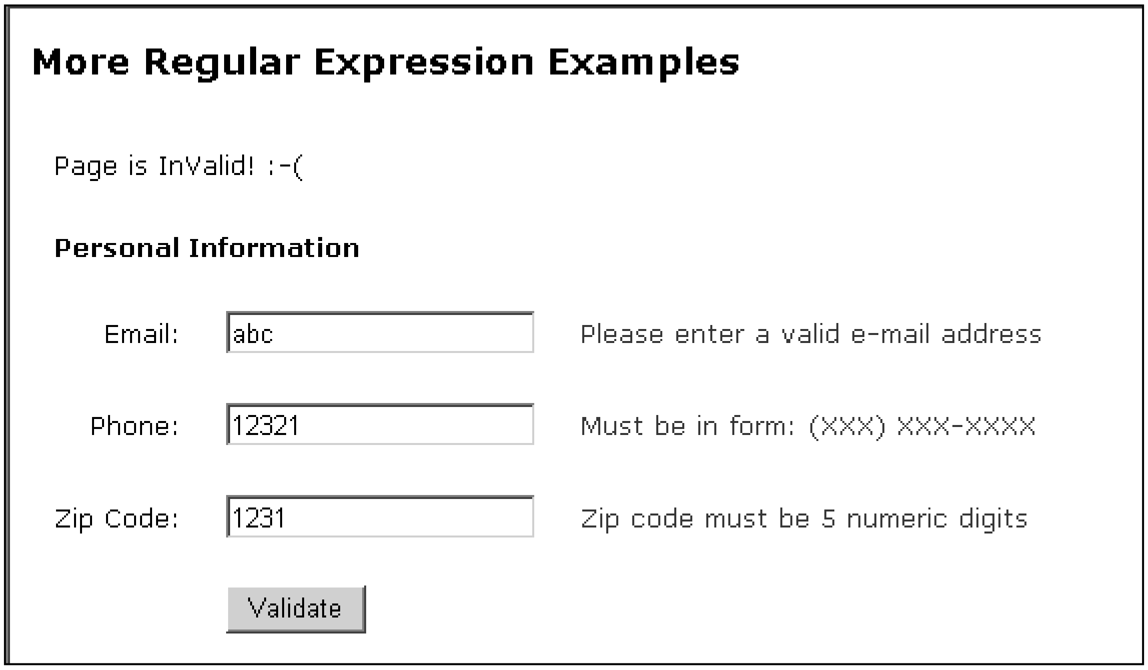
Simple RegularExpressionValidator Sample

Enter a 5 digit zip code

Personal Information

Zip Code: 123 Zip code must be 5 numeric digits

Figure 4.15. Controls 24.aspx



More Regular Expression Examples

Page is InValid! :-(

Personal Information

Email: abc — Please enter a valid e-mail address

Phone: 12321 — Must be in form: (XXX) XXX-XXXX

Zip Code: 1231 — Zip code must be 5 numeric digits

Code for Figure 4.14. Controls 23.aspx

```
<html>
<head>
 <script language="VB" runat="server">

      Sub ValidateBtn_Click(sender As Object, e As EventArgs)
        If (Page.IsValid) Then
          lblOutput.Text = "Page is Valid!"
        Else
          lblOutput.Text = "Page is InValid! :-("
        End If
      End Sub

   </script>

</head>
<body>

<h3><font face="Verdana">Simple RegularExpressionValidator Sample</font></h3>
<p>
```

```
<form runat="server">

   <table bgcolor="#eeeeee" cellpadding=10>
   <tr valign="top">
     <td colspan=3>
       <asp:Label ID="lblOutput" Text="Enter a 5 digit zip code" Font-
Name="Verdana" Font-Size="10pt" runat="server"/>
     </td>
   </tr>

   <tr>
     <td colspan=3>
     <font face=Verdana size=2><b>Personal Information</b></font>
     </td>
   </tr>
   <tr>
     <td align=right>
       <font face=Verdana size=2>Zip Code:</font>
     </td>
     <td>
       <ASP:TextBox id=TextBox1 runat=server />
     </td>
     <td>
       <asp:RegularExpressionValidator            id="RegularExpressionValidator1"
runat="server"
          ControlToValidate="TextBox1"
          ValidationExpression="^\d{5}$"
          Display="Static"
          Font-Name="verdana"
          Font-Size="10pt">
             Zip code must be 5 numeric digits
       </asp:RegularExpressionValidator>
     </td>
   </tr>
   <tr>
     <td></td>
     <td>
       <ASP:Button text="Validate" OnClick="ValidateBtn_Click" runat=server />
     </td>
     <td></td>
   </tr>
   </table>

</form>

</body>
```

```
</html>
```

Code for Figure 4.15. Controls 24.aspx

```
<%@ Page clienttarget="downlevel" %>

<html>
<head>

   <script language="VB" runat="server">

      Sub ValidateBtn_Click(sender As Object, e As EventArgs)
          If (Page.IsValid) Then
            lblOutput.Text = "Page is Valid!"
         Else
            lblOutput.Text = "Page is InValid! :-("
         End If
      End Sub

  </script>

</head>
<body>

<h3><font face="Verdana">More Regular Expression Examples</font></h3>
<p>

<form runat="server">

   <table cellpadding=10>
   <tr valign="top">
     <td colspan=3>
      <asp:Label ID="lblOutput" Text="Enter values for each field" Font-
Name="Verdana" Font-Size="10pt" runat="server" />
     </td>
   </tr>

   <tr>
     <td colspan=3>
     <font face=Verdana size=2><b>Personal Information</b></font>
     </td>
   </tr>
   <tr>
     <td align=right>
```

```
      <font face=Verdana size=2>Email:</font>
     </td>
     <td>
      <ASP:TextBox id=TextBox1 runat=server />
     </td>
     <td>
      <asp:RequiredFieldValidator                   id="RequiredFieldValidator1"
runat="server"
         ControlToValidate="TextBox1"
         Display="Dynamic"
         Font-Name="Verdana" Font-Size="10pt"
         >
         *
      </asp:RequiredFieldValidator>

      <asp:RegularExpressionValidator      id="RegularExpressionValidator1"
runat="server"
         ControlToValidate="TextBox1"
         ValidationExpression="^[\w-]+@[\w-
]+\.(com|net|org|edu|mil)$"
         Display="Static"
         Font-Name="verdana" Font-Size="10pt">
            Please enter a valid e-mail address
      </asp:RegularExpressionValidator>
     </td>
   </tr>
   <tr>
     <td align=right>
      <font face=Verdana size=2>Phone:</font>
     </td>
     <td>
      <ASP:TextBox id=TextBox2 runat=server />
     </td>
     <td>
      <asp:RequiredFieldValidator                   id="RequiredFieldValidator2"
runat="server"
         ControlToValidate="TextBox2"
         Display="Dynamic"
         Font-Name="Verdana" Font-Size="10pt">
         *
      </asp:RequiredFieldValidator>
      <asp:RegularExpressionValidator id="RegularExpressionValidator2"
         ControlToValidate="TextBox2"
         ValidationExpression="(^x\s*[0-9]{5}$)|(^(\([1-9][0-9]{2}\)\s)?[1-
9][0-9]{2}-[0-9]{4}(\sx\s*[0-9]{5})?$)"
         Display="Static"
         Font-Name="verdana" Font-Size="10pt"
```

```
          runat=server>
            Must be in form: (XXX) XXX-XXXX
        </asp:RegularExpressionValidator>
      </td>
    </tr>
    <tr>
      <td align=right>
        <font face=Verdana size=2>Zip Code:</font>
      </td>
      <td>
        <ASP:TextBox id=TextBox3 runat=server />
      </td>
      <td>
        <asp:RequiredFieldValidator          id="RequiredFieldValidator3"
runat="server"
          ControlToValidate="TextBox3"
          Display="Dynamic"
          Font-Name="Verdana" Font-Size="10pt">
            *
        </asp:RequiredFieldValidator>

        <asp:RegularExpressionValidator id="RegularExpressionValidator3"
          ControlToValidate="TextBox3"
          ValidationExpression="^\d{5}$"
          Display="Static"
          Width="100%"
          Font-Name="verdana" Font-Size="10pt"
          runat=server>
            Zip code must be 5 numeric digits
        </asp:RegularExpressionValidator>
      </td>
    </tr>
    <tr>
      <td></td>
      <td>
        <ASP:Button     text="Validate"     OnClick="ValidateBtn_Click"
runat=server />
      </td>
      <td></td>
    </tr>
    </table>
</form>
</body>
</html>
```

Performing Custom Validation

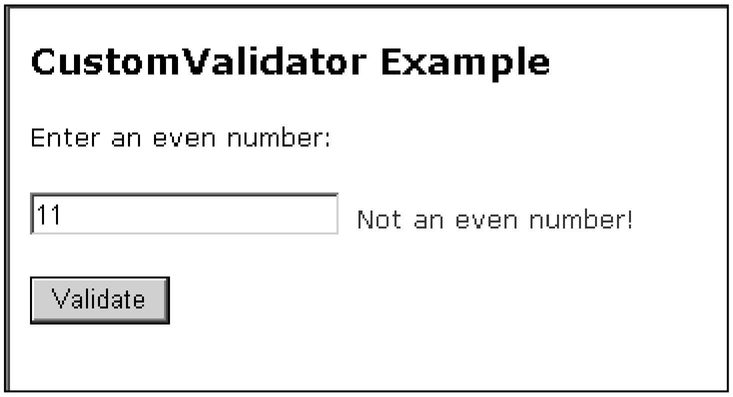
Coalesce

The **CustomValidator** server control calls a user-defined function to perform validations that the standard validators can't handle. The custom function can execute on the server or in client-side script, such as JScript or VBScript. For client-side custom validation, the name of the custom function must be identified in the **ClientValidationFunction** property. The custom function must have the form

```
function myvalidator(source, arguments)
```

Note that **source** is the client-side **CustomValidator** object, and **arguments** is an object with two properties, **Value** and **IsValid**. The **Value** property is the value to be validated and the **IsValid** property is a Boolean used to set the return result of the validation. For server-side custom validation, place your custom validation in the validator's **OnServerValidate** delegate.
The following sample shows how to use the **CustomValidator** control.

Figure 4.16. Controls 25.aspx



Code for Figure 4.16. Controls 25.aspx

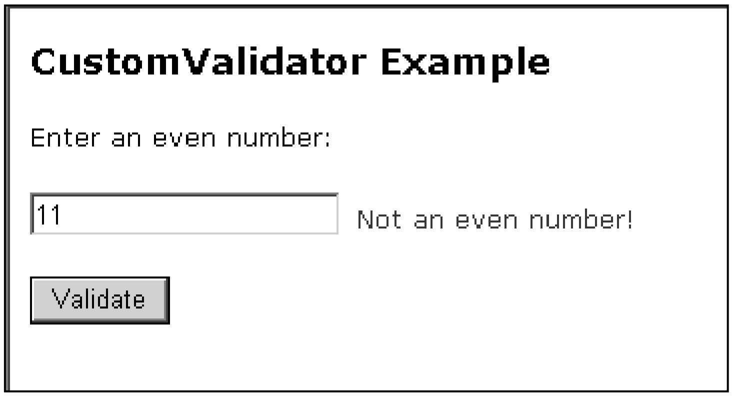
```
<html>
<head>
   <script language="VB" runat="server">

      Sub ValidateBtn_OnClick(sender As Object, e As EventArgs)
         If (Page.IsValid) Then
           lblOutput.Text = "Page is Valid!"
         Else
           lblOutput.Text = "Page is InValid! :-("
         End If
      End Sub

      Sub ServerValidate (sender As Object, value As ServerValidateEventArgs)
         Try
```

```
            Dim num As Int32 = Int32.Parse(value.Value)
            If num Mod 2 = 0 Then
                value.IsValid = True
                Exit Sub
            End If
        Catch exc As Exception
        End Try

        value.IsValid = False
    End Sub

  </script>

</head>
<body>

<h3><font face="Verdana">CustomValidator Example</font></h3>
<p>

<form runat="server">

   <asp:Label id=lblOutput runat="server" Text="Enter an even number:"
Font-Name="Verdana" Font-Size="10pt" /><br>

   <p>

   <asp:TextBox id=Text1 runat="server" />

   <asp:RequiredFieldValidator                    id="RequiredFieldValidator1"
runat="server"
       ControlToValidate="Text1"
       ErrorMessage="Please enter a number"
       Display="Dynamic"
       Font-Name="verdana" Font-Size="10pt">
   </asp:RequiredFieldValidator>

   <asp:CustomValidator id="CustomValidator1" runat="server"
       ControlToValidate="Text1"
              ClientValidationFunction="ClientValidate"
       OnServerValidate="ServerValidate"
       Display="Static"
       Font-Name="verdana" Font-Size="10pt">
         Not an even number!
   </asp:CustomValidator>

   <p>
   <asp:Button text="Validate" onclick="ValidateBtn_OnClick"
runat="server" />

   <script language="javascript">

      function ClientValidate(source, arguments)
      {
        // even number?
        if (arguments.Value%2 == 0)
          arguments.IsValid = true;
        else
          arguments.IsValid = false;
      }

   </script>

</form>


</body>
</html>
```

Bringing It All Together

This sample shows a typical registration form, using the variations of validation controls discussed in this topic.

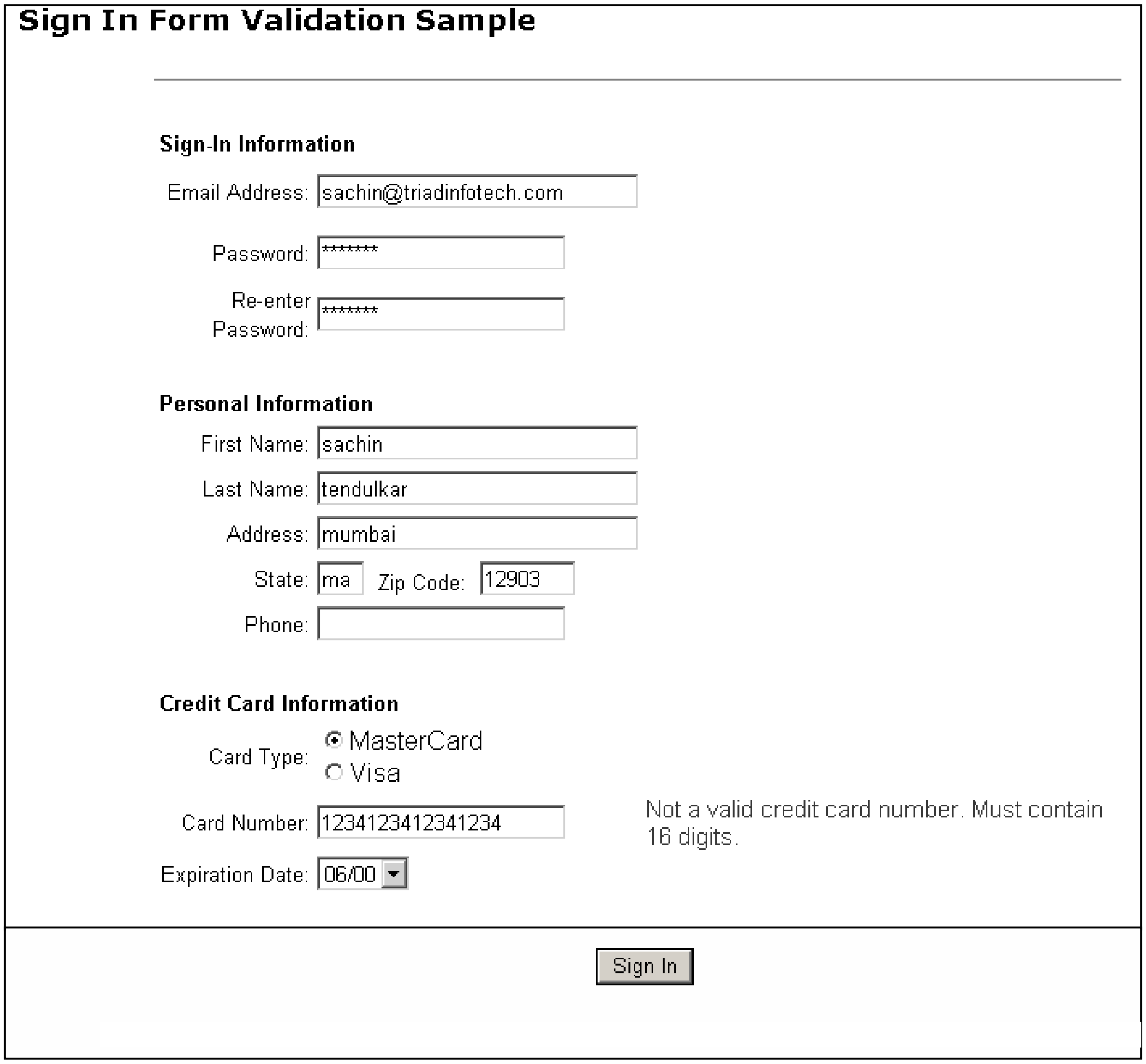
Figure 4.17. Controls 26.aspx


CustomValidator Example

Enter an even number:

11 Not an even number!

Figure 4.18. Controls 27.aspx



Sign In Form Validation Sample

Sign-In Information

Email Address:	sachin@triadinfotech.com
Password:	*******
Re-enter Password:	*******

Personal Information

First Name:	sachin
Last Name:	tendulkar
Address:	mumbai
State:	ma Zip Code: 12903
Phone:	

Credit Card Information

Card Type:	[✓ MasterCard] Visa
Card Number:	1234123412341234
Expiration Date:	06/00

Not a valid credit card number. Must contain 16 digits.

Figure 4.18. Controls 28.aspx

```
<%@ Page Language="VB" %>

<html>
<body>
```

```
<h3><font face="Verdana">Sign In Form Validation Sample</font></h3>

<form method=post runat=server>
<hr width=600 size=1 noshade>

<center>
<asp:ValidationSummary ID="valSum" runat="server"
  HeaderText="You must enter a valid value in the following fields:"
  DisplayMode="SingleParagraph"
  Font-Name="verdana"
  Font-Size="12"
  />
<p>

<!-- sign-in -->
<table border=0 width=600>
<tr><td colspan=3>
  <table border=0 cellpadding=0 cellspacing=0 width="100%">
  <tr><td>
    <font face=geneva,arial size=-1><b>Sign-In Information</b></font>
  </td></tr>
  </table>
</td></tr>
<tr>
 <td align=right>
  <font face=Arial size=2>Email Address:</font>
 </td>
 <td>
  <asp:TextBox id=email width=200px maxlength=60 runat=server />
 </td>
 <td>
  <asp:RequiredFieldValidator id="emailReqVal"
    ControlToValidate="email"
    ErrorMessage="Email. "
    Display="Dynamic"
    Font-Name="Verdana" Font-Size="12"
    runat=server>
    *
  </asp:RequiredFieldValidator>
  <asp:RegularExpressionValidator id="emailRegexVal"
    ControlToValidate="email"
    ErrorMessage="Email. "
    Display="Static"
    ValidationExpression="^[\w-]+@[\w-]+\.(com|net|org|edu|mil)$"
    Font-Name="Arial" Font-Size="11"
    runat=server>
```

```
        Not a valid e-mail address.  Must follow email@host.domain.
      </asp:RegularExpressionValidator>
    </td>
   </tr>

   <tr>
    <td align=right>
      <font face=Arial size=2>Password:</font>
    </td>
    <td>
      <asp:TextBox     id=passwd     TextMode="Password"     maxlength=20
runat=server/>
    </td>
    <td>
       <asp:RequiredFieldValidator id="passwdReqVal"
          ControlToValidate="passwd"
          ErrorMessage="Password. "
          Display="Dynamic"
          Font-Name="Verdana" Font-Size="12"
          runat=server>
          *
       </asp:RequiredFieldValidator>
       <asp:RegularExpressionValidator id="passwdRegexBal"
          ControlToValidate="passwd"
          ErrorMessage="Password. "
          ValidationExpression=".*[!@#$%^&*+;:].*"
          Display="Static"
          Font-Name="Arial" Font-Size="11"
          Width="100%" runat=server>
          Password must include one of these (!@#$%^&amp;*+;:)
       </asp:RegularExpressionValidator>
    </td>
   </tr>
   <tr>
    <td align=right>
      <font face=Arial size=2>Re-enter Password:</font>
    </td>
    <td>
      <asp:TextBox     id=passwd2     TextMode="Password"     maxlength=20
runat=server/>
    </td>
    <td>
      <asp:RequiredFieldValidator id="passwd2ReqVal"
        ControlToValidate="passwd2"
        ErrorMessage="Re-enter Password. "
        Display="Dynamic"
        Font-Name="Verdana" Font-Size="12"
        runat=server>
        *
    </asp:RequiredFieldValidator>
    <asp:CompareValidator id="CompareValidator1"
        ControlToValidate="passwd2" ControlToCompare="passwd"
        ErrorMessage="Re-enter Password. "
       Display="Static"
        Font-Name="Arial" Font-Size="11"
        runat=server>
        Password fields don't match
    </asp:CompareValidator>
  </td>
</tr>
<tr><td colspan=3>&nbsp;</td></tr>


<!-- personalization information -->
<tr><td colspan=3>
    <table border=0 cellpadding=0 cellspacing=0 width="100%">
    <tr><td><font face=geneva,arial size=-1>
        <b>Personal Information</b></font>
    </td></tr>
    </table>
</td></tr>
<tr>
  <td align=right>
    <font face=Arial size=2>First Name:</font>
  </td>
  <td>
    <asp:TextBox id=fn maxlength=20 width=200px runat=server />
  </td>
  <td>
  </td>
</tr>
<tr>
  <td align=right>
    <font face=Arial size=2>Last Name:</font>
  </td>
  <td>
    <asp:TextBox id=ln maxlength=40 width=200px runat=server />
  </td>
  <td>
  </td>
</tr>
<tr>
  <td align=right>
    <font face=Arial size=2>Address:</font>
    </td>
    <td>
     <asp:TextBox id=address width=200px runat=server />
    </td>
    <td>
    </td>
   </tr>
   <tr>
    <td align=right>
     <font face=Arial size=2>State:</font>
    </td>
    <td>
     <asp:TextBox id=state width=30px maxlength=2 runat=server />&nbsp;
     <font face=Arial size=2>Zip Code:</font>&nbsp;
     <ASP:TextBox id=zip width=60px maxlength=5 runat=server />
    </td>
    <td>
     <asp:RegularExpressionValidator id="RegularExpressionValidator1"
       ControlToValidate="zip"
       ErrorMessage="Zip Code. "
       ValidationExpression="^\d{5}$"
       Display="Static"
       Font-Name="Arial" Font-Size="11"
       runat=server>
       Zip code must be 5 numeric digits
     </asp:RegularExpressionValidator>
    </td>
   </tr>
   <tr>
    <td align=right>
     <font face=Arial size=2>Phone:</font>
    </td>
    <td>
     <asp:TextBox id="phone" maxlength=20 runat="server" />
    </td>
    <td>
     <asp:RequiredFieldValidator id="phoneReqVal"
       ControlToValidate="phone"
       ErrorMessage="Phone. "
       Display="Dynamic"
       Font-Name="Verdana" Font-Size="12"
       runat=server>
       *
     </asp:RequiredFieldValidator>
     <asp:RegularExpressionValidator id="phoneRegexVal"
       ControlToValidate="phone"
       ErrorMessage="Phone. "
```

```
        ValidationExpression="(^x\s*[0-9]{5}$)|(^(\([1-9][0-9]{2}\)\s)?[1-9][0-9]{2}-
[0-9]{4}(\sx\s*[0-9]{5})?$)"
        Display="Static"
        Font-Name="Arial" Font-Size="11"
        runat=server>
        Must be in form: (XXX) XXX-XXXX
      </asp:RegularExpressionValidator>
    </td>
  </tr>
  <tr><td colspan=3>&nbsp;</td></tr>

  <!-- Credit Card Info -->
  <tr>
    <td colspan=3>
    <font face=Arial size=2><b>Credit Card Information</b></font>
    </td>
  </tr>
  <tr>
    <td align=right>
      <font face=Arial size=2>Card Type:</font>
    </td>
    <td>
      <ASP:RadioButtonList id=ccType Font-Name="Arial" RepeatLayout="Flow"
runat=server>
        <asp:ListItem>MasterCard</asp:ListItem>
        <asp:ListItem>Visa</asp:ListItem>
      </ASP:RadioButtonList>
    </td>
    <td>
      <asp:RequiredFieldValidator id="ccTypeReqVal"
        ControlToValidate="ccType"
        ErrorMessage="Card Type. "
        Display="Static"
        InitialValue=""
        Font-Name="Verdana" Font-Size="12"
        runat=server>
        *
      </asp:RequiredFieldValidator>
    </td>
  </tr>
  <tr>
    <td align=right>
      <font face=Arial size=2>Card Number:</font>
    </td>
    <td>
      <ASP:TextBox id="ccNum" runat=server />
    </td>
    <td>
      <asp:RequiredFieldValidator id="ccNumReqVal"
        ControlToValidate="ccNum"
        ErrorMessage="Card Number. "
        Display="Dynamic"
        Font-Name="Verdana" Font-Size="12"
        runat=server>
        *
      </asp:RequiredFieldValidator>
      <asp:CustomValidator id="ccNumCustVal"
        ControlToValidate="ccNum"
        ErrorMessage="Card Number. "
        clientvalidationfunction="ccClientValidate"
        Display="Static"
        Font-Name="Arial" Font-Size="11"
        runat=server>
        Not a valid credit card number.  Must contain 16 digits.
      </asp:CustomValidator>
    </td>
  </tr>
  <tr>
    <td align=right>
      <font face=Arial size=2>Expiration Date:</font>
    </td>
    <td>
      <ASP:DropDownList id=expDate runat=server>
        <asp:ListItem></asp:ListItem>
        <asp:ListItem >06/00</asp:ListItem>
        <asp:ListItem >07/00</asp:ListItem>
        <asp:ListItem >08/00</asp:ListItem>
        <asp:ListItem >09/00</asp:ListItem>
        <asp:ListItem >10/00</asp:ListItem>
        <asp:ListItem >11/00</asp:ListItem>
        <asp:ListItem >01/01</asp:ListItem>
        <asp:ListItem >02/01</asp:ListItem>
        <asp:ListItem >03/01</asp:ListItem>
        <asp:ListItem >04/01</asp:ListItem>
        <asp:ListItem >05/01</asp:ListItem>
        <asp:ListItem >06/01</asp:ListItem>
        <asp:ListItem >07/01</asp:ListItem>
        <asp:ListItem >08/01</asp:ListItem>
        <asp:ListItem >09/01</asp:ListItem>
        <asp:ListItem >10/01</asp:ListItem>
        <asp:ListItem >11/01</asp:ListItem>
        <asp:ListItem >12/01</asp:ListItem>
      </ASP:DropDownList>
    </td>
    <td>
     <asp:RequiredFieldValidator id="expDateReqVal"
      ControlToValidate="expDate"
      ErrorMessage="Expiration Date. "
      Display="Static"
      InitialValue=""
      Font-Name="Verdana" Font-Size="12"
      runat=server>
      *
     </asp:RequiredFieldValidator>
    </td>
   </tr>
   </table>

   <p>
   <input runat="server" type=submit value="Sign In">
   <p>

   <hr width=600 size=1 noshade>

   <script language="javascript">

     function ccClientValidate(source, arguments)
     {
       var cc = arguments.Value;
       var ccSansSpace;
       var i, digits, total;

       // SAMPLE ONLY.  Not a real world actual credit card algo.
       // Based on ANSI X4.13, the LUHN formula (also known as the modulus 10 -- or
mod 10 -- algorithm )
       // is used to generate and/or validate and verify the accuracy of some credit-card
numbers.

       // Get the number, parse out any non digits, should have 16 left
       ccSansSpace = cc.replace(/\D/g, "");
       if(ccSansSpace.length != 16) {
        arguments.IsValid = false;
        return;   // invalid ccn
       }

       // Convert to array of digits
       digits = new Array(16);
       for(i=0; i<16; i++)
        digits[i] = Number(ccSansSpace.charAt(i));

       // Double & sum digits of every other number
```

```
        for(i=0; i<16; i+=2) {
            digits[i] *= 2;
            if(digits[i] > 9)   digits[i] -= 9;
        }

        // Sum the numbers
        total = 0;
        for(i=0; i<16; i++)     total += digits[i];

        // Results
        if( total % 10 == 0 )   {
            arguments.IsValid = true;
            return;   // valid ccn
        }
        else  {
            arguments.IsValid = false;
            return;   // invalid ccn
        }
    }

  </script>
</form>
</center>


</body>
</html>
```

Section Summary

1. Validator controls can be used to validate input on any Web Forms page.
2. More than one control can be used on a given input field.
3. Client-side validation may be used in addition to server validation to improve form usability.
4. The **ValidationSummary** control can be used to provide centralized error feedback by querying all validation controls for error messages
5. Simple validation can be performed using the **CompareValidator** and **RangeValidator** classes. These are commonly used on numeric data.
6. Complex pattern validation can be performed using the **RegularExpressionValidator**. Pattern validation is useful for strings like names, address, phone numbers, and email addresses.
7. The **CustomValidator** control lets the user define custom validation criteria.

Chapter 5

Web Forms User Controls

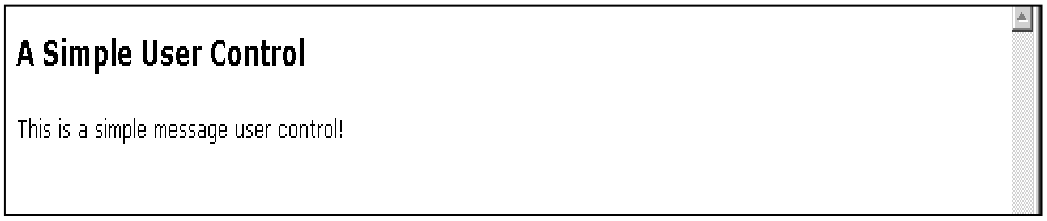
In addition to the built-in server controls provided by ASP.NET, you can easily define your own controls using the same programming techniques that you have already learned for writing Web Forms pages. In fact, with just a few modifications, almost any Web Forms page can be reused in another page as a server control (note that a user control is of type **System.Web.UI.UserControl**, which inherits directly from **System.Web.UI.Control**). A Web Forms page used as a server control is named a user control for short. As a matter of convention, the .ascx extension is used to indicate such controls. This ensures that the user control's file cannot be executed as a standalone Web Forms page (you will see a little that there are a few, albeit important, differences between a user control and a Web Forms page). User controls are included in a Web Forms page using a **Register** directive:

<%@ Register TagPrefix="Acme" TagName="Message" Src="pagelet1.ascx" %>

The **TagPrefix** determines a unique namespace for the user control (so that multiple user controls with the same name can be differentiated from each other). The **TagName** is the unique name for the user control (you can choose any name). The **Src** attribute is the virtual path to the user control--for example "MyPagelet.ascx" or "/MyApp/Include/MyPagelet.ascx". After registering the user control, you may place the user control tag in the Web Forms page just as you would an ordinary server control (including the **runat="server"** attribute):

<Acme:Message runat="server"/>
The following example shows a user control imported into another Web Forms page. Note that the user control in this case is just a simple static file.

Figure 5.1 SimpleUserControl.aspx



A Simple User Control

This is a simple message user control!

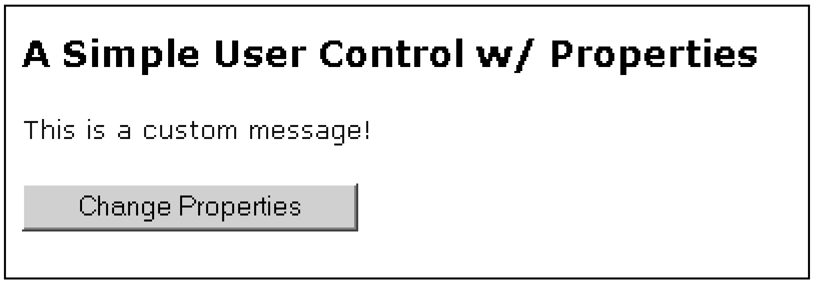
Code for Figure 5.1 SimpleUserControl.aspx

```
<%@ Register TagPrefix="Acme" TagName="Message"
Src="pagelet1.ascx" %>

<html>
<body style="font: 10pt verdana">
```

```
  <h3>A Simple User Control</h3>

  <Acme:Message runat="server"/>

</body>
</html>
```

Introduction to User Controls Exposing User Control Properties
When a Web Forms page is treated as a control, the public fields and methods of that Web Form are promoted to public properties (that is, tag attributes) and methods of the control as well. The following example shows an extension of the previous user control example that adds two public **String** fields. Notice that these fields can be set either declaratively or programmatically in the containing page.

Figure 5.2 SimpleUserControl.aspx


A Simple User Control w/ Properties
This is a custom message!
Change Properties

Code for Figure 5.2 SimpleUserControl.aspx

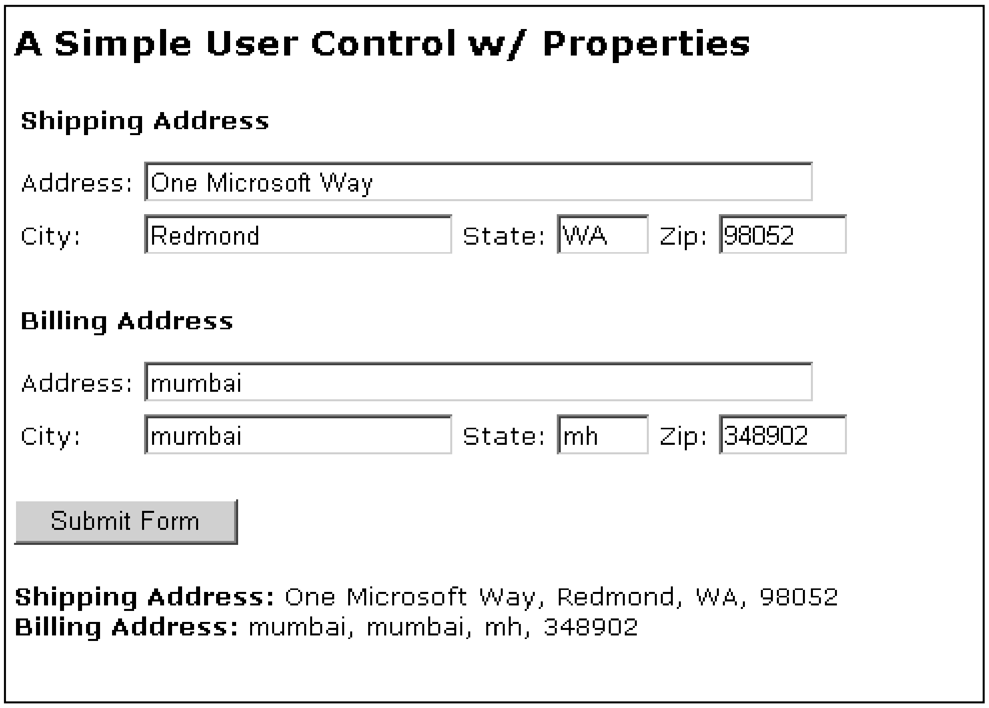
```
<%@ Register TagPrefix="Acme" TagName="Message" Src="pagelet2.ascx" %>

<html>

  <script language="VB" runat="server">

      Sub SubmitBtn_Click(Sender As Object, E As EventArgs)
          MyMessage.MessageText = "Message text changed!"
          MyMessage.Color = "red"
      End Sub
  </script>
<body style="font: 10pt verdana">
  <h3>A Simple User Control w/ Properties</h3>
  <form runat="server">
    <Acme:Message id="MyMessage" MessageText="This is a custom message!"
Color="blue" runat="server"/>
    <p>
    <asp:button text="Change Properties" OnClick="SubmitBtn_Click"
runat=server/>
```

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

In addition to promoting public fields to control properties, the property syntax may be used. Property syntax has the advantage of being able to execute code when properties are set or retrieved. The following example demonstrates an **Address** user control that wraps the text properties of **TextBox** controls within it. The benefit of doing this is that the control inherits the automatic state management of the **TextBox** control for free.

Notice that there are two **Address** user controls on the containing Web Forms page that set the **Caption** property to "Billing Address" and "Shipping Address", respectively. The real power of user controls is in this type of reusability.

Figure 5.2 SimpleUserControl.aspx



A Simple User Control w/ Properties

Shipping Address

Address:	One Microsoft Way				
City:	Redmond	State:	WA	Zip:	98052

Billing Address

Address:	mumbai				
City:	mumbai	State:	mh	Zip:	348902

Shipping Address: One Microsoft Way, Redmond, WA, 98052
Billing Address: mumbai, mumbai, mh, 348902

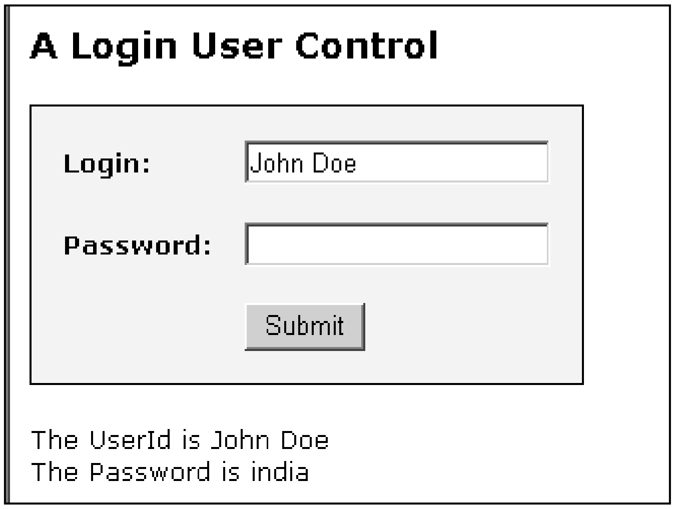
Code for Figure 5.2 SimpleUserControl.aspx

```
<%@ Register TagPrefix="Acme" TagName="Address"
Src="pagelet3.ascx" %>
<html>

   <script language="VB" runat="server">

      Sub SubmitBtn_Click(Sender As Object, E As EventArgs)
```

```
        MyLabel.Text &= "<b>Shipping Address:</b> " _
                & ShipAddr.Address & ", " _
                & ShipAddr.City & ", " _
                & ShipAddr.StateName & ", " _
                & ShipAddr.Zip & "<br>"

        MyLabel.Text &= "<b>Billing Address:</b> " _
                & BillAddr.Address & ", " _
                & BillAddr.City & ", " _
                & BillAddr.StateName & ", " _
                & BillAddr.Zip & "<br>"
      End Sub

   </script>

<body style="font: 10pt verdana">

  <h3>A Simple User Control w/ Properties</h3>

  <form runat="server">

   <Acme:Address id="ShipAddr" Caption="Shipping Address" Address="One
Microsoft Way" City="Redmond" StateName="WA" Zip="98052"
runat="server"/>

   <p>

   <Acme:Address id="BillAddr" Caption="Billing Address" runat="server"/>

   <p>

   <asp:button Text="Submit Form" OnClick="SubmitBtn_Click"
runat=server/>

  </form>

  <asp:Label id="MyLabel" runat="server"/>

</body>
</html>
```

Another useful user control is a **Login** control for collecting user names and passwords.

Figure 5.3 LoginUserControl.aspx



```
The UserId is John Doe
The Password is india
```

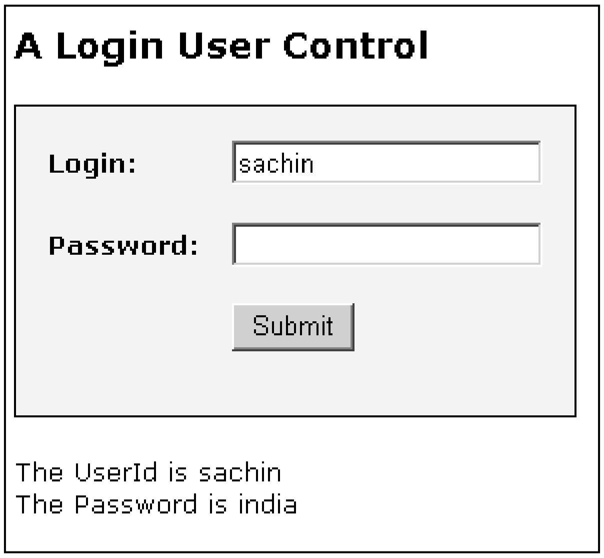
Code for Figure 5.3 LoginUserControl.aspx

```
<%@ Register TagPrefix="Acme" TagName="Login"
Src="pagelet4.ascx" %>

<html>

<script language="VB" runat="server">

  Sub Page_Load(Sender As Object, E As EventArgs)

    If (Page.IsPostBack)
      MyLabel.Text &= "The UserId is " & MyLogin.UserId & "<br>"
      MyLabel.Text &= "The Password is " & MyLogin.Password & "<br>"
    End If
  End Sub

</script>

<body style="font: 10pt verdana">

  <h3>A Login User Control</h3>

  <form runat="server">

    <Acme:Login id="MyLogin" UserId="John Doe" Password="Secret"
BackColor="beige" runat="server"/>

  </form>

  <asp:Label id="MyLabel" runat="server"/>
```

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

In this example, form validation controls are added to the **Login** user control.

Figure 5.4 LoginUserControl.aspx



The UserId is sachin
The Password is india

Code for Figure 5.4 LoginUserControl.aspx

```
<%@ Register TagPrefix="Acme" TagName="Login" Src="pagelet4.ascx"
%>

<html>

<script language="VB" runat="server">

  Sub Page_Load(Sender As Object, E As EventArgs)

    If (Page.IsPostBack)
      MyLabel.Text &= "The UserId is " & MyLogin.UserId & "<br>"
      MyLabel.Text &= "The Password is " & MyLogin.Password & "<br>"
    End If
  End Sub

</script>

<body style="font: 10pt verdana">

  <h3>A Login User Control</h3>
  <form runat="server">

    <Acme:Login id="MyLogin" UserId="John Doe" Password="Secret"
BackColor="beige" runat="server"/>

  </form>

  <asp:Label id="MyLabel" runat="server"/>

</body>
</html>
```

Encapsulating Events in a User Control

User controls participate in the complete execution lifecycle of the request, much the way ordinary server controls do. This means that a user control can handle its own events, encapsulating some of the page logic from the containing Web Forms page. The following example demonstrates a product-listing user control that internally handles its own postbacks. Note that the user control itself has no wrapping **<form runat="server">** control. Because only one form control may be present on a page (ASP.NET does not allow nested server forms), it is left to the containing Web Forms page to define this.

Code for 5.4 LoginUserControl.aspx

```
<%@ Register TagPrefix="Acme" TagName="BookList"
Src="pagelet6.ascx" %>

<html>
<body style="font: 10pt verdana">

  <h3>A User Control w/ an Event</h3>

  <form runat="server">

    <Acme:BookList runat="server"/>

  </form>

</body>
</html>
```

Creating User Controls Programmatically

Coalesce

Just as ordinary server controls can be created programmatically, so user controls can be. The page's **LoadControl** method is used to load the user control, passing the virtual path to the user control's source file:

```
Dim c1 As Control =
LoadControl("pagelet7.ascx")
CType(c1, (Pagelet7VB)).Category = "business"
Page.Controls.Add(c1)
```

The type of the user control is determined by a **ClassName** attribute on the **Control** directive. For example, a user control saved with the file name "pagelet7.ascx" is assigned the strong type "Pagelet7CS" as follows:

<%@ Control ClassName="Pagelet7CS" %>
Because the **LoadControl** method returns a type of **System.Web.UI.Control**, it must be cast to the appropriate strong type in order to set individual properties of the control. Finally, the user control is added to the base page's **ControlCollection**.

Code for LoginUserControl.aspx

```
<%@ Register TagPrefix="Acme" TagName="BookList"
Src="pagelet7.ascx" %>
<html>

  <script language="VB" runat="server">

    Sub Page_Load(Sender As Object, E As EventArgs)

      Page.Controls.Add(New HtmlGenericControl("hr"))

      Dim c1 As Control = LoadControl("pagelet7.ascx")
      CType(c1, Pagelet7VB).Category = "business"
      Page.Controls.Add(c1)

      Page.Controls.Add(New HtmlGenericControl("hr"))

      Dim c2 As Control = LoadControl("pagelet7.ascx")
      CType(c2, Pagelet7VB).Category = "trad_cook"
      Page.Controls.Add(c2)

      Page.Controls.Add(New HtmlGenericControl("hr"))

      Dim c3 As Control = LoadControl("pagelet7.ascx")
      CType(c3, Pagelet7VB).Category = "mod_cook"
      Page.Controls.Add(c3)
```

```
        End Sub

     </script>

  <body style="font: 10pt verdana">

    <h3>Creating User Controls Programmatically</h3>

  </body>
  </html>
```

Important The strong type for a user control is available to the containing Web Forms page only if a **Register** directive is included for the user control (even if there are no user control tags actually declared).

Section Summary

1. User controls allow developers to easily define custom controls using the same programming techniques as for writing Web Forms pages.
2. As a matter of convention, an .ascx file name extension is used to indicate such controls. This ensures that a user control file cannot be executed as a standalone Web Forms page.
3. User controls are included into another Web Forms page using a **Register** directive, which specifies a **TagPrefix**, **TagName**, and **Src location**.
4. After the user control has been registered, a user control tag may be placed in a Web Forms page as an ordinary server control (including the **runat="server"** attribute).
5. The public fields, properties, and methods of a user control are promoted to public properties (tag attributes) and methods of the control in the containing Web Forms page.
6. User controls participate in the complete execution lifecycle of every request and can handle their own events, encapsulating some of the page logic from the containing Web Forms page.
7. User controls should not contain any form controls but should instead rely on their containing Web Forms page to include one if necessary.
8. User controls may be created programmatically using the **LoadControl** method of the **System.Web.UI.Page** class. The type of the user control is determined by the ASP.NET runtime, following the convention *filename_extension*.
9. The strong type for a user control is available to the containing Web Forms page only if a **Register** directive is included for the user control (even if there are no user control tags actually declared).

Coalesce

Chapter 6

Data Binding Server Controls

Data Binding Overview and Syntax

ASP.NET introduces a new declarative data binding syntax. This extremely flexible syntax permits the developer to bind not only to data sources, but also to simple properties, collections, expressions, and even results returned from method calls. The following table shows some examples of the new syntax.

Simple property	Customer: <%# custID %>
Collection	Orders: <asp:ListBox id="List1" datasource='<%# myArray %>' runat="server">
Expression	Contact: <%# (customer.FirstName + " " + customer.LastName) %>
Method result	Outstanding Balance: <%# GetBalance(custID) %>

Although this syntax looks similar to the ASP shortcut for **Response.Write** -- <%= %> -- its behavior is quite different. Whereas the ASP **Response.Write** shortcut syntax was evaluated when the page was processed, the ASP.NET data binding syntax is evaluated only when the **DataBind** method is invoked.

DataBind is a method of the **Page** and all server controls. When you call **DataBind** on a parent control, it cascades to all of the children of the control. So, for example, DataList1.DataBind() invokes the **DataBind** method on each of the controls in the **DataList** templates. Calling **DataBind** on the **Page** -- Page.DataBind() or simply DataBind() -- causes all data binding expressions on the page to be evaluated. **DataBind** is commonly called from the **Page_Load** event, as shown in the following example.

```
Protected Sub Page_Load(Src As Object, E As EventArgs)
   DataBind()
End Sub
```

You can use a binding expression almost anywhere in the declarative section of an .aspx page, provided it evaluates to the expected data type at run time. The simple property, expression, and method examples above display text to the user when evaluated. In these cases, the data binding expression must evaluate to a value of type **String**. In the collection example, the data binding expression evaluates to a value of valid type for the **DataSource** property of **ListBox**. You might find it necessary to coerce the type of value in your binding expression to produce the desired result. For example, if count is an integer:

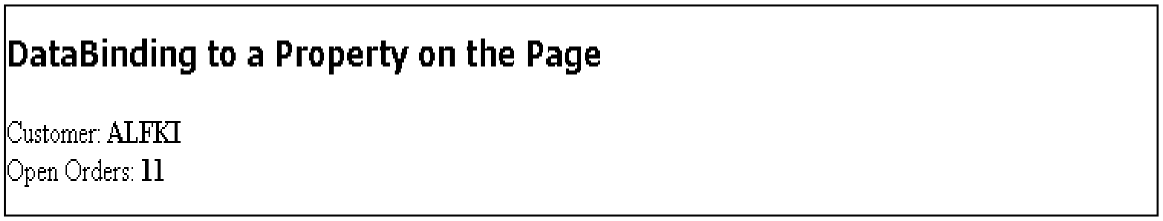
Number of Records: <%# count.ToString() %>

Binding to Simple Properties

The ASP.NET data binding syntax supports binding to public variables, properties of the **Page**, and properties of other controls on the page.

The following example illustrates binding to a public variable and simple property on the page. Note that these values are initialized before DataBind() is called.

Figure 6.1 Databinding.aspx



DataBinding to a Property on the Page

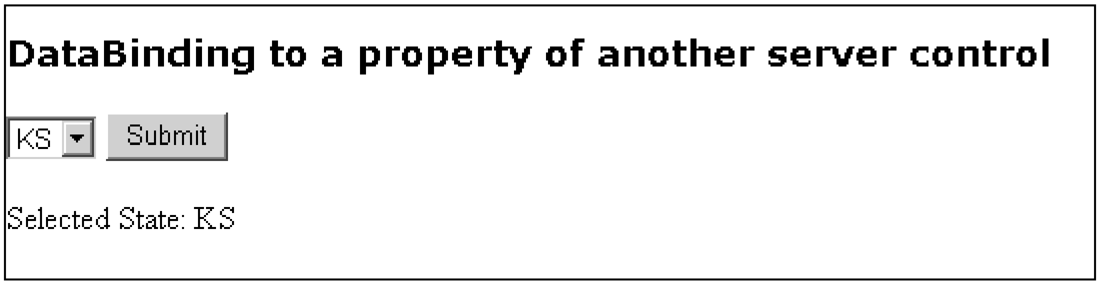
Customer: ALFKI
Open Orders: 11

Code for Databinding.aspx

```
<html>
<head>
   <script language="VB" runat="server">

      Sub Page_Load(sender As Object, e As EventArgs)
         Page.DataBind
      End Sub

      ReadOnly Property custID() As String
         Get
            Return "ALFKI"
         End Get
      End Property

      ReadOnly Property orderCount() As Integer
         Get
            Return 11
         End Get
      End Property
   </script>
</head>
<body>

   <h3><font face="Verdana">DataBinding to a Property on the
Page</font></h3>

   <form runat=server>

      Customer: <b><%# custID %></b><br>
      Open Orders: <b><%# orderCount %></b>

   </form>

</body>
</html>
```

The following example illustrates binding to a property of another control.

Figure 6.2 Databindinganotherserver.aspx


DataBinding to a property of another server control

Selected State: KS

Code for Figure 6.2 Databindinganotherserver.aspx

```
<html>
<head>
   <script language="VB" runat="server">

      Sub SubmitBtn_Click(sender As Object, e As EventArgs)

        ' Rather than explictly pull out the variable from the "StateList"
        ' and then manipulate a label control, just call "Page.DataBind".
        ' This will evaluate any <%# %> expressions within the page

        Page.DataBind
      End Sub

   </script>
```

```
</head>
<body>

   <h3><font face="Verdana">DataBinding to a property of another server
control</font></h3>

   <form runat=server>

      <asp:DropDownList id="StateList" runat="server">
        <asp:ListItem>CA</asp:ListItem>
        <asp:ListItem>IN</asp:ListItem>
        <asp:ListItem>KS</asp:ListItem>
        <asp:ListItem>MD</asp:ListItem>
        <asp:ListItem>MI</asp:ListItem>
        <asp:ListItem>OR</asp:ListItem>
        <asp:ListItem>TN</asp:ListItem>
        <asp:ListItem>UT</asp:ListItem>
      </asp:DropDownList>

      <asp:button      Text="Submit"      OnClick="SubmitBtn_Click"
runat=server/>

      <p>

      Selected State: <asp:label text='<%# StateList.SelectedItem.Text %>'
runat=server/>

   </form>

</body>
</html>
```

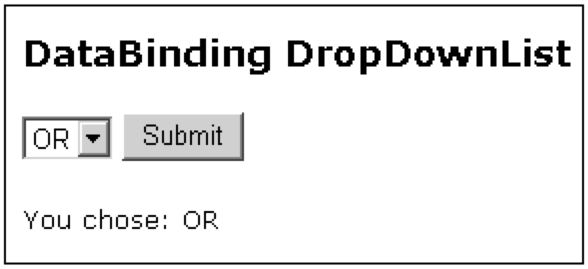
Binding to Collections and Lists

List server controls like **DataGrid**, **ListBox** and **HTMLSelect** use a collection as a data source. The following examples illustrate binding to usual common language runtime collection types. These controls can bind only to collections that support the **IEnumerable**, **ICollection**, or **IListSource** interface. Most commonly, you'll bind to **ArrayList**, **Hashtable**, **DataView** and **DataReader**.

The following example illustrates binding to an **ArrayList**.

Coalesce

Figure 6.3 DatabindingDropDownlist.aspx


DataBinding DropDownList

You chose: OR

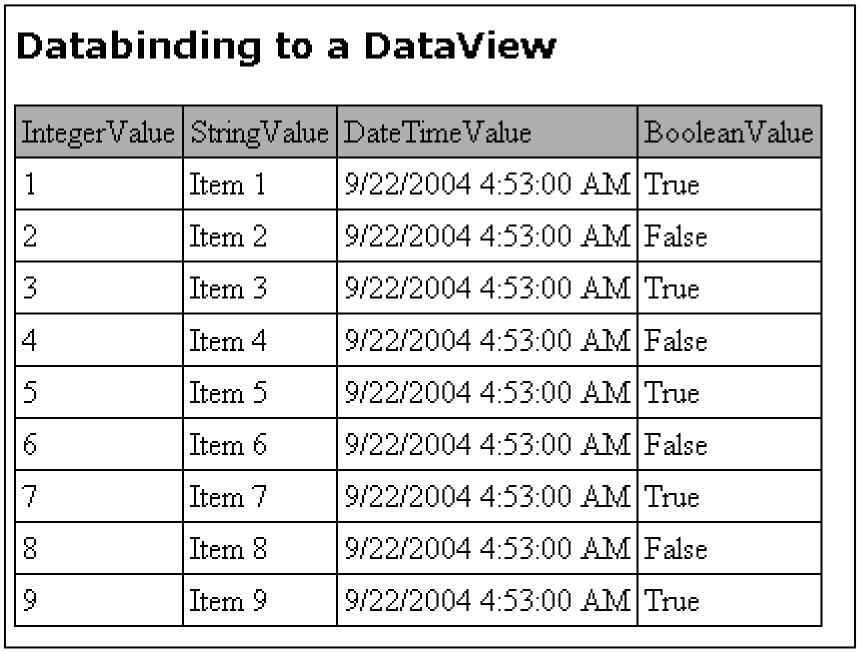
Code for Figure 6.3 DatabindingDropDownlist.aspx

```
<html>
<head>

   <script language="VB" runat="server">

      Sub Page_Load(sender As Object, e As EventArgs)
         If Not IsPostBack Then

            Dim values as ArrayList= new ArrayList()

            values.Add ("IN")
            values.Add ("KS")
            values.Add ("MD")
            values.Add ("MI")
            values.Add ("OR")
            values.Add ("TN")

            DropDown1.DataSource = values
            DropDown1.DataBind
         End If
      End Sub

      Sub SubmitBtn_Click(sender As Object, e As EventArgs)
        Label1.Text = "You chose: " + DropDown1.SelectedItem.Text
      End Sub

   </script>

</head>
<body>

   <h3><font face="Verdana">DataBinding DropDownList</font></h3>
```

```
  <form runat=server>

    <asp:DropDownList id="DropDown1" runat="server" />

    <asp:button Text="Submit" OnClick="SubmitBtn_Click"
runat=server/>

    <p>

    <asp:Label id=Label1 font-name="Verdana" font-size="10pt"
runat="server" />

  </form>

</body>
</html>
```

The following example illustrates binding to a **DataView**. Note that the **DataView** class is defined in the **System.Data** namespace.

Figure 6.4 DatabindingtoDataview.aspx



Databinding to a DataView

IntegerValue	StringValue	DateTimeValue	BooleanValue
1	Item 1	9/22/2004 4:53:00 AM	True
2	Item 2	9/22/2004 4:53:00 AM	False
3	Item 3	9/22/2004 4:53:00 AM	True
4	Item 4	9/22/2004 4:53:00 AM	False
5	Item 5	9/22/2004 4:53:00 AM	True
6	Item 6	9/22/2004 4:53:00 AM	False
7	Item 7	9/22/2004 4:53:00 AM	True
8	Item 8	9/22/2004 4:53:00 AM	False
9	Item 9	9/22/2004 4:53:00 AM	True

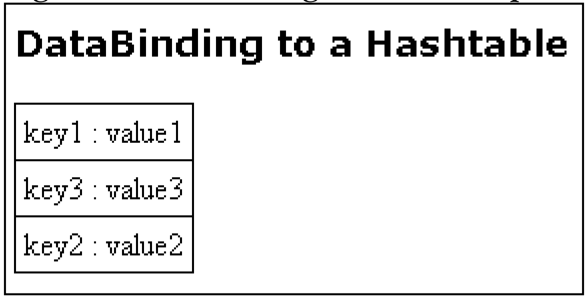
Code for Figure 6.5 DatabindingtoDataview.aspx

Coalesce

```
<%@ Import namespace="System.Data" %>
<html>
<head>
  <script language="VB" runat="server">

    Sub Page_Load(sender As Object, e As EventArgs)
      If Not IsPostBack Then
        Dim dt As DataTable
        Dim dr As DataRow
        Dim i As Integer

        'create a DataTable
        dt = New DataTable
        dt.Columns.Add(New DataColumn("IntegerValue", GetType(Integer)))
        dt.Columns.Add(New DataColumn("StringValue", GetType(String)))
        dt.Columns.Add(New                    DataColumn("DateTimeValue",
GetType(DateTime)))
        dt.Columns.Add(New                     DataColumn("BooleanValue",
GetType(Boolean)))

        'Make some rows and put some sample data in

        For i = 1 To 9
          dr = dt.NewRow()
          dr(0) = i
          dr(1) = "Item " + i.ToString()
          dr(2) = DateTime.Now.ToShortTimeString
          If (i Mod 2 <> 0) Then
            dr(3) = True
          Else
            dr(3) = False
          End If
          'add the row to the datatable
          dt.Rows.Add(dr)
        Next

        dataGrid1.DataSource = new DataView(dt)
        dataGrid1.DataBind

      End If
    End Sub

  </script>

</head>
<body>
```

```
   <h3><font face="Verdana">Databinding to a DataView</font></h3>

   <form runat=server>

      <asp:DataGrid id="dataGrid1" runat="server"
        BorderColor="black"
        BorderWidth="1"
        GridLines="Both"
        CellPadding="3"
        CellSpacing="0"
        HeaderStyle-BackColor="#aaaadd"
      />

   </form>

</body>
</html>
```

Figure 6.6 DatabindingtoDataview.aspx



DataBinding to a Hashtable

key1 : value1
key3 : value3
key2 : value2

Code for Figure 6.6 DatabindingtoDataview.aspx

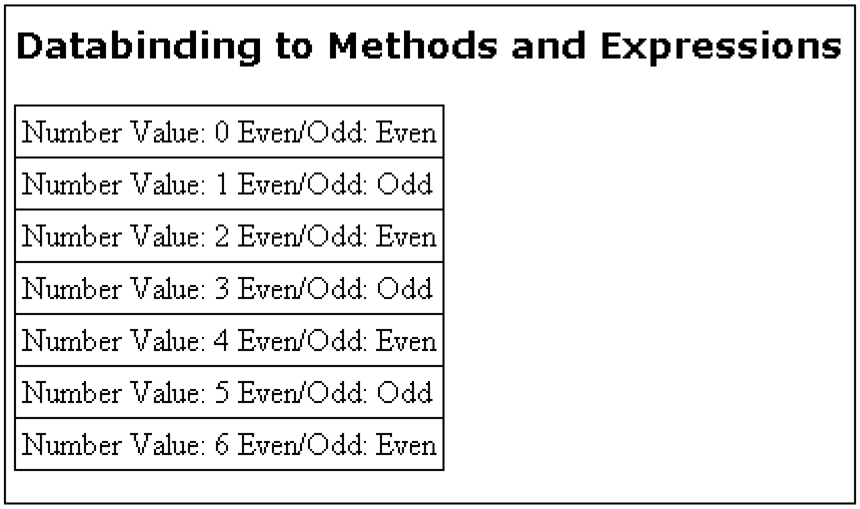
```
<html>
<head>
   <script language="VB" runat="server">
      Sub Page_Load(sender As Object, e As EventArgs)
         If Not IsPostBack Then
            Dim h As Hashtable = new Hashtable()
            h.Add ("key1", "value1")
            h.Add ("key2", "value2")
            h.Add ("key3", "value3")
            MyDataList.DataSource = h
            MyDataList.DataBind
         End If
      End Sub
   </script>
</head>
<body>
```

```
    <h3><font face="Verdana">DataBinding to a Hashtable</font></h3>
    <form runat=server>
       <asp:DataList id="MyDataList" runat="server"
         BorderColor="black"
         BorderWidth="1"
         GridLines="Both"
         CellPadding="4"
         CellSpacing="0"
         >
           <ItemTemplate>
              <%# Container.DataItem.Key %> :
              <%# Container.DataItem.Value %>
           </ItemTemplate>
       </asp:DataList>
    </form>
</body>
</html>
```

Binding Expressions or Methods

Often, you'll want to manipulate data before binding to your page or a control. The following example illustrates binding to an expression and the return value of a method.

Figure 6.7 DatabindingtoMethods.aspx



Databinding to Methods and Expressions

Number Value: 0 Even/Odd: Even
Number Value: 1 Even/Odd: Odd
Number Value: 2 Even/Odd: Even
Number Value: 3 Even/Odd: Odd
Number Value: 4 Even/Odd: Even
Number Value: 5 Even/Odd: Odd
Number Value: 6 Even/Odd: Even

Code for Figure 6.7 DatabindingtoMethods.aspx

```
<html>
<head>
```

Coalesce

```
  <script language="VB" runat="server">
    Sub Page_Load(sender As Object, e As EventArgs)
      If Not IsPostBack Then
        Dim values as ArrayList= new ArrayList()
        values.Add (0)
        values.Add (1)
        values.Add (2)
        values.Add (3)
        values.Add (4)
        values.Add (5)
        values.Add (6)
        DataList1.DataSource = values
        DataList1.DataBind
      End If
    End Sub
    Function EvenOrOdd(number As Integer) As String
      If (number Mod 2 <> 0) Then
       Return "Odd"
      Else
       Return "Even"
      End If
    End Function
  </script>
</head>
<body>
  <h3><font face="Verdana">Databinding to Methods and
Expressions</font></h3>
  <form runat=server>
   <asp:DataList id="DataList1" runat="server"
    BorderColor="black"
    BorderWidth="1"
    GridLines="Both"
    CellPadding="3"
    CellSpacing="0"
    >
    <ItemTemplate>
     Number Value: <%# Container.DataItem %>
     Even/Odd: <%# EvenOrOdd(Container.DataItem) %>
    </ItemTemplate>
   </asp:datalist>
  </form>
</body>
</html>
```

DataBinder.Eval

Coalesce

The ASP.NET framework supplies a static method that evaluates late-bound data binding expressions and optionally formats the result as a string. **DataBinder.Eval** is convenient in that it eliminates much of the explicit casting the developer must do to coerce values to the desired data type. It is particularly useful when data binding controls within a templated list, because often both the data row and the data field must be cast.

Consider the following example, where an integer will be displayed as a currency string. With the standard ASP.NET data binding syntax, you must first cast the type of the data row in order to retrieve the data field, IntegerValue. Next, this is passed as an argument to the **String.Format** method.

```
<%# String.Format("{0:c}", (CType(Container.DataItem,
DataRowView)("IntegerValue"))) %>
```

This syntax can be complex and difficult to remember. In contrast, **DataBinder.Eval** is simply a method with three arguments: the naming container for the data item, the data field name, and a format string. In a templated list like **DataList**, **DataGrid**, or **Repeater**, the naming container is always Container.DataItem. **Page** is another naming container that can be used with **DataBinder.Eval**.

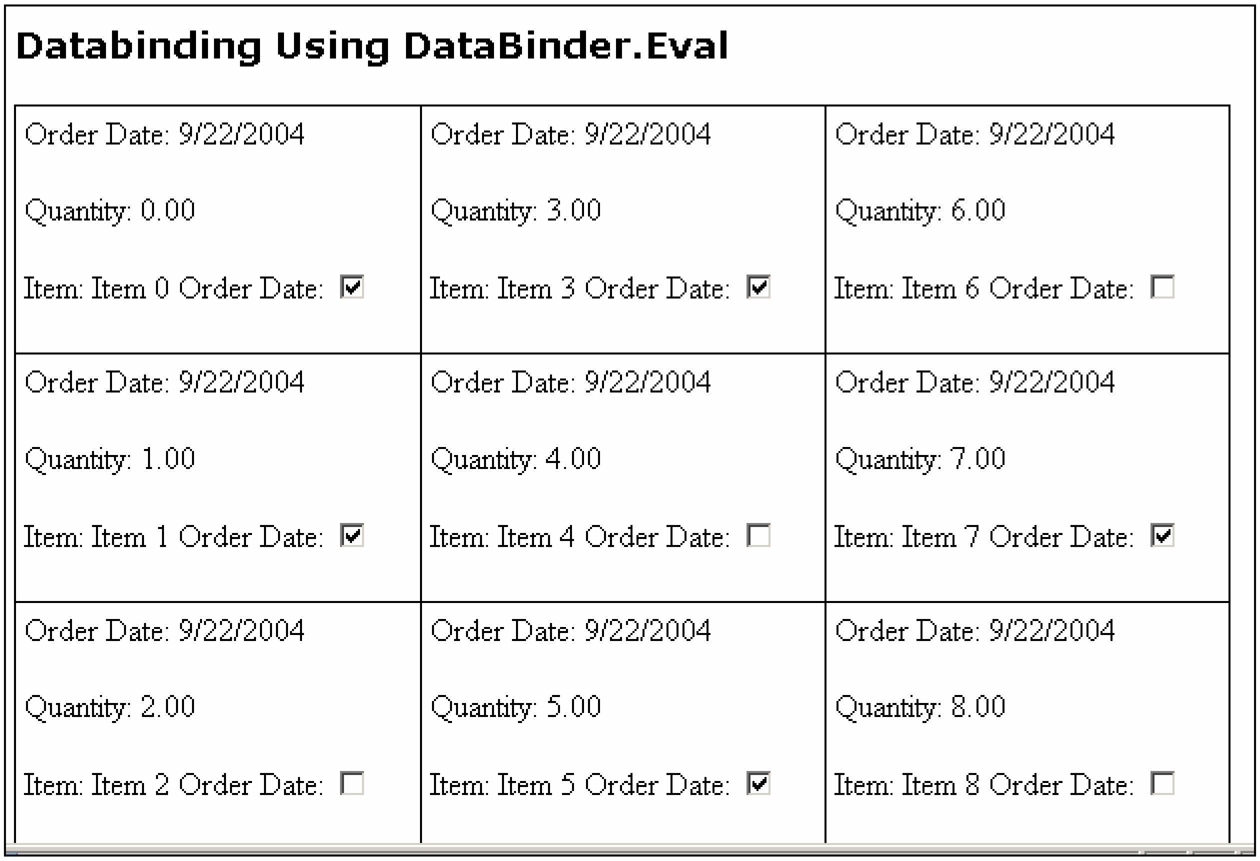
```
<%# DataBinder.Eval(Container.DataItem, "IntegerValue", "{0:c}") %>
```

The format string argument is optional. If it is omitted, **DataBinder.Eval** returns a value of type object, as shown in the following example.

```
<%# CType(DataBinder.Eval(Container.DataItem, "BoolValue"), Boolean) %>
```

It is important to note that **DataBinder.Eval** can carry a noticeable performance penalty over the standard data binding syntax because it uses late-bound reflection. Use **DataBinder.Eval** judiciously, especially when string formatting is not required.

Figure 6.8 DatabindingusingDataBinder.aspx



Databinding Using DataBinder.Eval

Order Date: 9/22/2004 Quantity: 0.00 Item: Item 0 Order Date: ✓	Order Date: 9/22/2004 Quantity: 3.00 Item: Item 3 Order Date: ✓	Order Date: 9/22/2004 Quantity: 6.00 Item: Item 6 Order Date: ☐
Order Date: 9/22/2004 Quantity: 1.00 Item: Item 1 Order Date: ✓	Order Date: 9/22/2004 Quantity: 4.00 Item: Item 4 Order Date: ☐	Order Date: 9/22/2004 Quantity: 7.00 Item: Item 7 Order Date: ✓
Order Date: 9/22/2004 Quantity: 2.00 Item: Item 2 Order Date: ☐	Order Date: 9/22/2004 Quantity: 5.00 Item: Item 5 Order Date: ✓	Order Date: 9/22/2004 Quantity: 8.00 Item: Item 8 Order Date: ☐

Code for Figure 6.8 DatabindingusingDataBinder.aspx

```
<html>
<head>
   <script language="VB" runat="server">
      Sub Page_Load(sender As Object, e As EventArgs)
         If Not IsPostBack Then
            Dim values as ArrayList= new ArrayList()
            values.Add (0)
            values.Add (1)
            values.Add (2)
            values.Add (3)
            values.Add (4)
            values.Add (5)
            values.Add (6)
            DataList1.DataSource = values
            DataList1.DataBind
         End If
      End Sub
```

```
     Function EvenOrOdd(number As Integer) As String
        If (number Mod 2 <> 0) Then
          Return "Odd"
        Else
          Return "Even"
        End If
     End Function
  </script>
</head>
<body>
  <h3><font face="Verdana">Databinding to Methods and Expressions</font></h3>
  <form runat=server>
   <asp:DataList id="DataList1" runat="server"
     BorderColor="black"
     BorderWidth="1"
     GridLines="Both"
     CellPadding="3"
     CellSpacing="0"
     >
     <ItemTemplate>
       Number Value: <%# Container.DataItem %>
       Even/Odd: <%# EvenOrOdd(Container.DataItem) %>
     </ItemTemplate>
   </asp:datalist>
  </form>
</body>
</html>
```

Section Summary

1. The ASP.NET declarative data binding syntax uses the <%# %> notation.
2. You can bind to data sources, properties of the page or another control, collections, expressions, and results returned from method calls.
3. List controls can bind to collections that support the **ICollection**, **IEnumerable**, or **IListSource** interface, such as **ArrayList**, **Hashtable**, **DataView**, and **DataReader**.
4. **DataBinder.Eval** is a static method for late binding. Its syntax can be simpler than the standard data binding syntax, but performance is slower.

Chapter 7

DataGrid ,Data Access and Template Controls

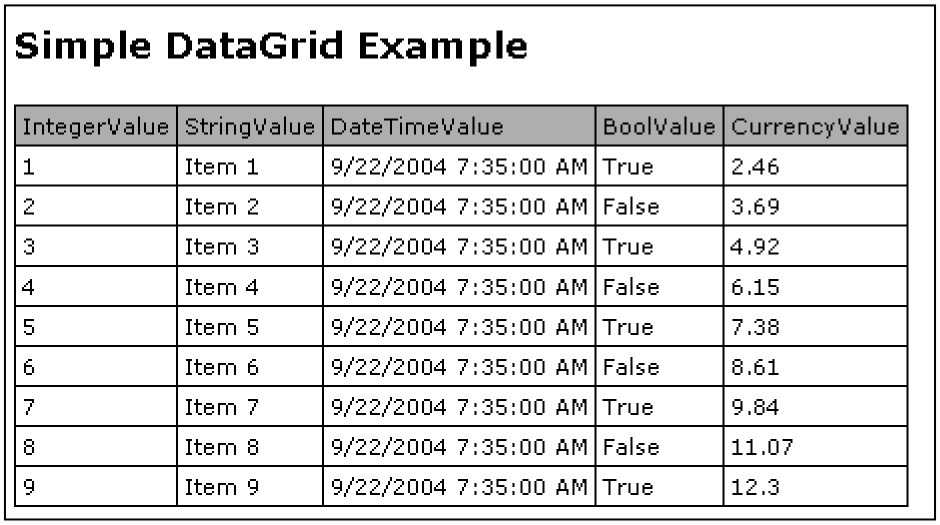
These samples illustrate using the **DataGrid** control. These examples use sample data rather than data from a real database. Please see the Server-Side Data Access section for examples of **DataGrid** bound to live data.

Working With DataGrid

The **DataGrid** control displays tabular data and optionally supports selecting, sorting, paging, and editing the data. By default, **DataGrid** generates a **BoundColumn** for each field in the data source (**AutoGenerateColumns=true**). Each field in the data is rendered in a separate column, in the order it occurs in the data. Field names appear in the grid's column headers, and values are rendered in text labels. A default format is applied to non-string values.

The following sample illustrates using a simple **DataGrid** control.

Figure 7.1 SimpleDataGrid.aspx



Simple DataGrid Example

IntegerValue	StringValue	DateTimeValue	BoolValue	CurrencyValue
1	Item 1	9/22/2004 7:35:00 AM	True	2.46
2	Item 2	9/22/2004 7:35:00 AM	False	3.69
3	Item 3	9/22/2004 7:35:00 AM	True	4.92
4	Item 4	9/22/2004 7:35:00 AM	False	6.15
5	Item 5	9/22/2004 7:35:00 AM	True	7.38
6	Item 6	9/22/2004 7:35:00 AM	False	8.61
7	Item 7	9/22/2004 7:35:00 AM	True	9.84
8	Item 8	9/22/2004 7:35:00 AM	False	11.07
9	Item 9	9/22/2004 7:35:00 AM	True	12.3

Code for Figure 7.1 SimpleDataGrid.aspx

```
<%@ Import Namespace="System.Data" %>
<html>
<script language="VB" runat="server">
   Function CreateDataSource() As ICollection
      Dim dt As DataTable
      Dim dr As DataRow
      Dim i As Integer

      'create a DataTable
```

```
        dt = New DataTable
        dt.Columns.Add(New DataColumn("IntegerValue", GetType(Integer)))
        dt.Columns.Add(New DataColumn("StringValue", GetType(String)))
        dt.Columns.Add(New DataColumn("DateTimeValue", GetType(DateTime)))
        dt.Columns.Add(New DataColumn("BoolValue", GetType(Boolean)))
        dt.Columns.Add(new DataColumn("CurrencyValue", GetType(Double)))

        'Make some rows and put some sample data in
        For i = 1 To 9
            dr = dt.NewRow()
            dr(0) = i
            dr(1) = "Item " + i.ToString()
            dr(2) = DateTime.Now.ToShortTimeString
            If (i Mod 2 <> 0) Then
                dr(3) = True
            Else
                dr(3) = False
            End If
            dr(4) = 1.23 * (i+1)
            'add the row to the datatable
            dt.Rows.Add(dr)
        Next

        'return a DataView to the DataTable
        CreateDataSource = New DataView(dt)

    End Function

    Sub Page_Load(sender As Object, e As EventArgs)
        MyDataGrid.DataSource = CreateDataSource()
        MyDataGrid.DataBind
    End Sub

</script>

<body>

    <h3><font face="Verdana">Simple DataGrid Example</font></h3>

    <form runat=server>

      <ASP:DataGrid id="MyDataGrid" runat="server"
        BorderColor="black"
        BorderWidth="1"
        GridLines="Both"
        CellPadding="3"
        CellSpacing="0"
```

```
     Font-Name="Verdana"
     Font-Size="8pt"
     HeaderStyle-BackColor="#aaaadd"
    />

  </form>

</body>
</html>
```

Defining Columns in DataGrid

You can control the order, behavior, and rendering of individual columns by directly manipulating the grid's Columns collection. The standard column type -- **BoundColumn** -- renders the values in text labels. The grid also supports other column types that render differently. Any of the column types can be used together with the Columns collection of a **DataGrid**.

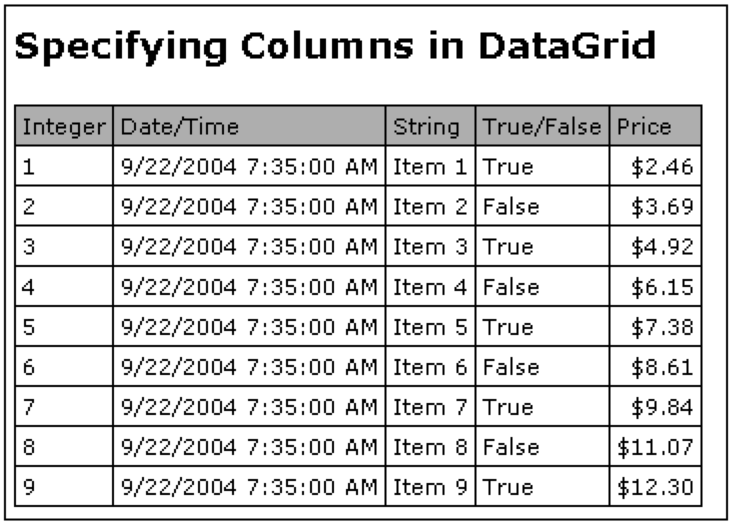
Note that you can use explicitly-declared columns together with auto-generated columns (**AutoGenerateColumns=true**). When used together, the explicitly-declared columns in the Columns collection are rendered first, and then the auto-generated columns are rendered. The auto-generated columns are not added to the Columns collection.

Column Name	**Description**
BoundColumn	Lets you control the order and rendering of the columns.
HyperLinkColumn	Presents the bound data in **HyperLink** controls.
ButtonColumn	Bubbles a user command from within a row to an event handler on the grid.
TemplateColumn	Lets you control which controls are rendered in the column.
EditCommandColumn	Displays Edit, Update, and Cancel links in response to changes in the **DataGrid** control's **EditItemIndex** property.

By explicitly creating a **BoundColumn** in the grid's Columns collection, you can control the order and rendering of each column. The following example shows how to use **BoundColumn**.

Figure 7.2 SpecificColumnsDataGrid.aspx

Coalesce



Specifying Columns in DataGrid

Integer	Date/Time	String	True/False	Price
1	9/22/2004 7:35:00 AM	Item 1	True	$2.46
2	9/22/2004 7:35:00 AM	Item 2	False	$3.69
3	9/22/2004 7:35:00 AM	Item 3	True	$4.92
4	9/22/2004 7:35:00 AM	Item 4	False	$6.15
5	9/22/2004 7:35:00 AM	Item 5	True	$7.38
6	9/22/2004 7:35:00 AM	Item 6	False	$8.61
7	9/22/2004 7:35:00 AM	Item 7	True	$9.84
8	9/22/2004 7:35:00 AM	Item 8	False	$11.07
9	9/22/2004 7:35:00 AM	Item 9	True	$12.30

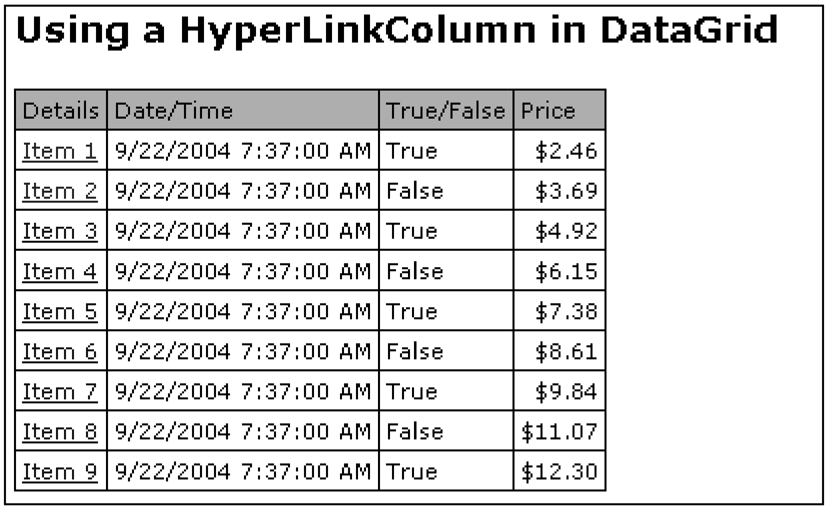
Code for Figure 7.2 SpecificColumnsDataGrid.aspx

```
<%@ Page Debug="True" %>
<%@ Import Namespace="System.Data" %>

<html>
<script language="VB" runat="server">

  Function CreateDataSource() As ICollection

    Dim dt As DataTable
    Dim dr As DataRow
    Dim i As Integer

    'create a DataTable
    dt = New DataTable
    dt.Columns.Add(New DataColumn("IntegerValue", GetType(Integer)))
    dt.Columns.Add(New DataColumn("StringValue", GetType(String)))
    dt.Columns.Add(New DataColumn("DateTimeValue", GetType(DateTime)))
    dt.Columns.Add(New DataColumn("BoolValue", GetType(Boolean)))
    dt.Columns.Add(new DataColumn("CurrencyValue", GetType(Double)))

    'Make some rows and put some sample data in
    For i = 1 To 9
      dr = dt.NewRow()
      dr(0) = i
      dr(1) = "Item " + i.ToString()
      dr(2) = DateTime.Now.ToShortTimeString
      If (i Mod 2 <> 0) Then
        dr(3) = True
      Else
        dr(3) = False
```

```
            End If
            dr(4) = 1.23 * (i+1)
            'add the row to the datatable
            dt.Rows.Add(dr)
        Next

        'return a DataView to the DataTable
        CreateDataSource = New DataView(dt)

    End Function

    Sub Page_Load(sender As Object, e As EventArgs)
        MyDataGrid.DataSource = CreateDataSource()
        MyDataGrid.DataBind
    End Sub

</script>

<body>

    <h3><font face="Verdana">Specifying Columns in DataGrid</font></h3>

    <form runat=server>

      <ASP:DataGrid id="MyDataGrid" runat="server"
        BorderColor="black"
        BorderWidth="1"
        GridLines="Both"
        CellPadding="3"
        CellSpacing="0"
        Font-Name="Verdana"
        Font-Size="8pt"
        HeaderStyle-BackColor="#aaaadd"
        AutoGenerateColumns="false">
          <Columns>
            <asp:BoundColumn                    HeaderText="Integer"
DataField="IntegerValue" />
            <asp:BoundColumn                    HeaderText="Date/Time"
DataField="DateTimeValue"/>
            <asp:BoundColumn                    HeaderText="String"
DataField="StringValue"/>
            <asp:BoundColumn                    HeaderText="True/False"
DataField="BoolValue"/>
            <asp:BoundColumn                    HeaderText="Price"
DataField="CurrencyValue"     DataFormatString="{0:c}"     ItemStyle-
HorizontalAlign="right" />
          </Columns>
    </asp:DataGrid>

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

A **HyperLinkColumn** presents the bound data in **HyperLink** controls. This is typically used to navigate from an item in the grid to a Details view on another page. In the following example, the value **IntegerValue** data field is passed as an argument in the URL to another page, and the **StringValue** data field is used as the display text of the hyperlink.

Figure 7.3 HyperLinkcolumn.aspx



Using a HyperLinkColumn in DataGrid

Details	Date/Time	True/False	Price
Item 1	9/22/2004 7:37:00 AM	True	$2.46
Item 2	9/22/2004 7:37:00 AM	False	$3.69
Item 3	9/22/2004 7:37:00 AM	True	$4.92
Item 4	9/22/2004 7:37:00 AM	False	$6.15
Item 5	9/22/2004 7:37:00 AM	True	$7.38
Item 6	9/22/2004 7:37:00 AM	False	$8.61
Item 7	9/22/2004 7:37:00 AM	True	$9.84
Item 8	9/22/2004 7:37:00 AM	False	$11.07
Item 9	9/22/2004 7:37:00 AM	True	$12.30

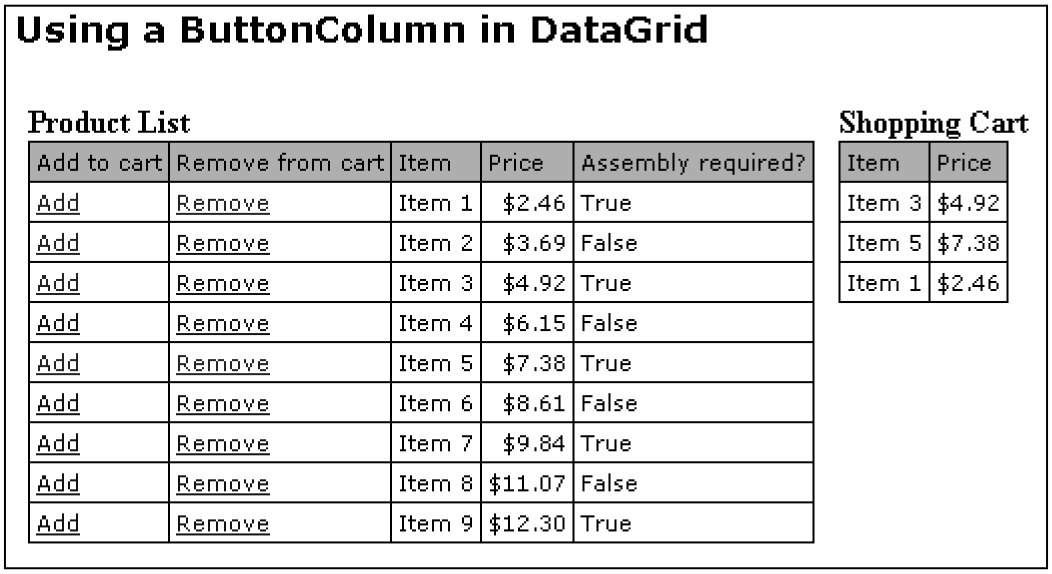
Datagrid for Figure 7.3 HyperLinkcolumn.aspx

```
<%@ Import Namespace="System.Data" %>
<html>
<script language="VB" runat="server">
  Function CreateDataSource() As ICollection
    Dim dt As DataTable
    Dim dr As DataRow
    Dim i As Integer
    'create a DataTable
    dt = New DataTable
    dt.Columns.Add(New DataColumn("IntegerValue", GetType(Integer)))
    dt.Columns.Add(New DataColumn("StringValue", GetType(String)))
    dt.Columns.Add(New DataColumn("DateTimeValue", GetType(DateTime)))
    dt.Columns.Add(New DataColumn("BoolValue", GetType(Boolean)))
    dt.Columns.Add(new DataColumn("CurrencyValue", GetType(Double)))
    'Make some rows and put some sample data in
```

```
        For i = 1 To 9
            dr = dt.NewRow()
            dr(0) = i
            dr(1) = "Item " + i.ToString()
            dr(2) = DateTime.Now.ToShortTimeString
            If (i Mod 2 <> 0) Then
                dr(3) = True
            Else
                dr(3) = False
            End If
            dr(4) = 1.23 * (i+1)
            'add the row to the datatable
            dt.Rows.Add(dr)
        Next
        'return a DataView to the DataTable
        CreateDataSource = New DataView(dt)
    End Function
    Sub Page_Load(sender As Object, e As EventArgs)
        MyDataGrid.DataSource = CreateDataSource()
        MyDataGrid.DataBind
    End Sub
</script>
<body>
    <h3><font face="Verdana">Using a HyperLinkColumn in DataGrid</font></h3>
   <form runat=server>
      <ASP:DataGrid id="MyDataGrid" runat="server"
        BorderColor="black"
        BorderWidth="1"
        GridLines="Both"
        CellPadding="3"
        CellSpacing="0"
        Font-Name="Verdana"
        Font-Size="8pt"
        HeaderStyle-BackColor="#aaaadd"
        AutoGenerateColumns="false"
      >
          <Columns>
            <asp:HyperLinkColumn
                HeaderText="Details"
                DataNavigateUrlField="IntegerValue"
                DataNavigateUrlFormatString="detailspage.aspx?id={0}"
                DataTextField="StringValue"
                Target="_new"
            />
            <asp:BoundColumn                                    HeaderText="Date/Time"
DataField="DateTimeValue"/>
            <asp:BoundColumn HeaderText="True/False" DataField="BoolValue"/>
          <asp:BoundColumn HeaderText="Price"
            DataField="CurrencyValue"
            DataFormatString="{0:c}"
            ItemStyle-HorizontalAlign="right"
            />
        </Columns>
      </asp:DataGrid>
  </form>
</body>
</html>
```

A **ButtonColumn** is used to bubble a user command from within a row to an event handler on the grid. In the following example, the "Add To Cart" and "Remove From Cart" commands cause the item from the row where the button was clicked to be added or removed from a simple shopping cart.

Figure 7.4 Buttoncolumn.aspx



Using a ButtonColumn in DataGrid

Product List

Add to cart	Remove from cart	Item	Price	Assembly required?
Add	Remove	Item 1	$2.46	True
Add	Remove	Item 2	$3.69	False
Add	Remove	Item 3	$4.92	True
Add	Remove	Item 4	$6.15	False
Add	Remove	Item 5	$7.38	True
Add	Remove	Item 6	$8.61	False
Add	Remove	Item 7	$9.84	True
Add	Remove	Item 8	$11.07	False
Add	Remove	Item 9	$12.30	True

Shopping Cart

Item	Price
Item 3	$4.92
Item 5	$7.38
Item 1	$2.46

Code for Figure 7.4 Buttoncolumn.aspx

```
<%@ Import Namespace="System.Data" %>

<html>
<script language="VB" runat="server">

   Dim Cart As DataTable
   Dim CartView As DataView

   Function CreateDataSource() As ICollection
```

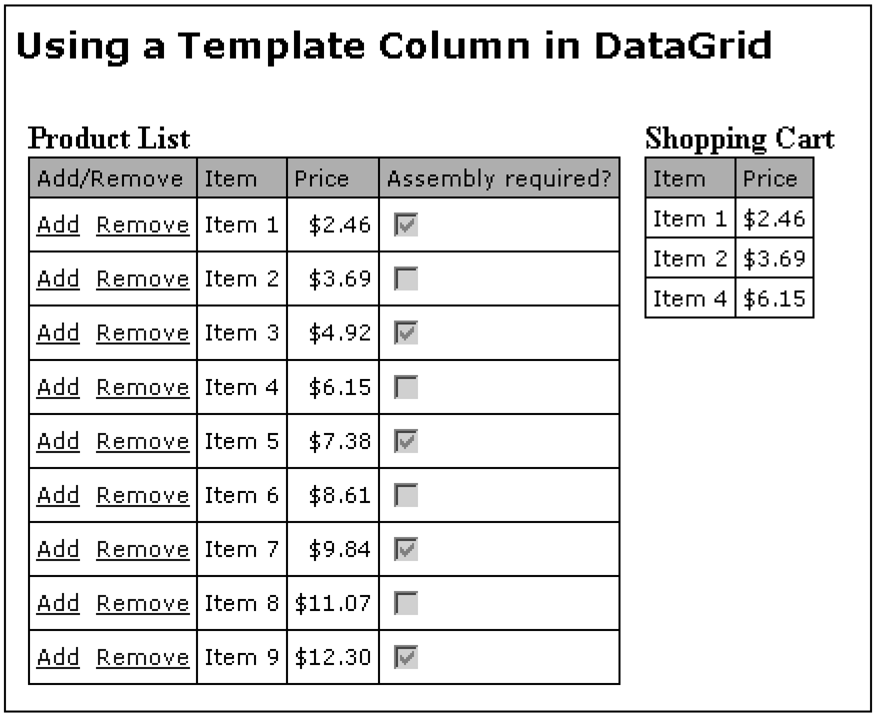
```
        Dim dt As DataTable
        Dim dr As DataRow
        Dim i As Integer

        'create a DataTable
        dt = New DataTable
        dt.Columns.Add(New DataColumn("IntegerValue", GetType(Integer)))
        dt.Columns.Add(New DataColumn("StringValue", GetType(String)))
        dt.Columns.Add(New                    DataColumn("DateTimeValue",
GetType(DateTime)))
        dt.Columns.Add(New DataColumn("BoolValue", GetType(Boolean)))
        dt.Columns.Add(new DataColumn("CurrencyValue", GetType(Double)))

        'Make some rows and put some sample data in
        For i = 1 To 9
            dr = dt.NewRow()
            dr(0) = i
            dr(1) = "Item " & i.ToString()
            dr(2) = DateTime.Now.ToShortTimeString
            If (i Mod 2 <> 0) Then
                dr(3) = True
            Else
                dr(3) = False
            End If
            dr(4) = 1.23 * (i + 1)
            'add the row to the datatable
            dt.Rows.Add(dr)
        Next

        'return a DataView to the DataTable
        CreateDataSource = New DataView(dt)

    End Function

    Sub Page_Load(sender As Object, e As EventArgs)

        If Session("DG4VB_ShoppingCart") Is Nothing Then
            Cart = New DataTable()
            Cart.Columns.Add(new DataColumn("Item", GetType(string)))
            Cart.Columns.Add(new DataColumn("Price", GetType(string)))
            Session("DG4VB_ShoppingCart") = Cart
        Else
            Cart = Session("DG4VB_ShoppingCart")
        End If
        CartView = New DataView(Cart)
        ShoppingCart.DataSource = CartView
        ShoppingCart.DataBind
```

```
      If Not IsPostBack Then
         ' need to load this data only once
         MyDataGrid.DataSource = CreateDataSource()
         MyDataGrid.DataBind
      End If
   End Sub

   Sub     Grid_CartCommand(sender     As     Object,     e     As
DataGridCommandEventArgs)

      Dim dr As DataRow = Cart.NewRow()

      ' e.Item is the row of the table where the command fired
      ' For bound columns the value is stored in the Text property of TableCell
      Dim itemCell As TableCell = e.Item.Cells(2)
      Dim priceCell As TableCell = e.Item.Cells(3)
      Dim item As String = itemCell.Text
      Dim price As String = priceCell.Text

      If e.CommandSource.CommandName = "AddToCart" Then
         dr(0) = item
         dr(1) = price
         Cart.Rows.Add(dr)
      Else   'Remove from Cart

         CartView.RowFilter = "Item='" & item & "'"
         If CartView.Count > 0 Then
            CartView.Delete(0)
         End If
         CartView.RowFilter = ""
      End If
      ShoppingCart.DataBind()

   End Sub


</script>

<body>

   <h3><font     face="Verdana">Using     a     ButtonColumn     in
DataGrid</font></h3>

   <form runat=server>

   <table cellpadding="5">
   <tr valign="top"><td>

   <b>Product List</b>
   <ASP:DataGrid id="MyDataGrid" runat="server"
      BorderColor="black"
      BorderWidth="1"
      GridLines="Both"
      CellPadding="3"
      CellSpacing="0"
      Font-Name="Verdana"
      Font-Size="8pt"
      HeaderStyle-BackColor="#aaaadd"
      AutoGenerateColumns="false"
      OnItemCommand="Grid_CartCommand"
      >
      <Columns>
         <asp:ButtonColumn HeaderText="Add to cart" Text="Add"
CommandName="AddToCart" />
         <asp:ButtonColumn HeaderText="Remove from cart"
Text="Remove" CommandName="RemoveFromCart" />
         <asp:BoundColumn HeaderText="Item"
DataField="StringValue"/>
         <asp:BoundColumn HeaderText="Price"
DataField="CurrencyValue" DataFormatString="{0:c}" ItemStyle-
HorizontalAlign="right" />
         <asp:BoundColumn HeaderText="Assembly required?"
DataField="BoolValue"/>
      </Columns>
   </asp:DataGrid>

   </td><td>

   <b>Shopping Cart</b>
   <ASP:DataGrid id="ShoppingCart" runat="server"
      BorderColor="black"
      BorderWidth="1"
      CellPadding="3"
      Font-Name="Verdana"
      Font-Size="8pt"
      HeaderStyle-BackColor="#aaaadd"
      />

   </td></tr>
   </table>

  </form>
```

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

With a **TemplateColumn**, you completely control which controls are rendered in the column, and which data fields are bound to the controls. The following example includes two **TemplateColumn** objects. The first column renders two **LinkButton** controls. These bubble commands to the grid's **ItemCommand**, just as a **ButtonColumn** does. The last column binds the **true/false** value to a read-only **CheckBox**.

Figure 7.5 Templatecolumn.aspx



Using a Template Column in DataGrid

Product List

Add/Remove	Item	Price	Assembly required?
Add Remove	Item 1	$2.46	✓
Add Remove	Item 2	$3.69	☐
Add Remove	Item 3	$4.92	✓
Add Remove	Item 4	$6.15	☐
Add Remove	Item 5	$7.38	✓
Add Remove	Item 6	$8.61	☐
Add Remove	Item 7	$9.84	✓
Add Remove	Item 8	$11.07	☐
Add Remove	Item 9	$12.30	✓

Shopping Cart

Item	Price
Item 1	$2.46
Item 2	$3.69
Item 4	$6.15

Code for Figure 7.5 Templatecolumn.aspx

```
<%@ Import Namespace="System.Data" %>
<html>
<script language="VB" runat="server">
  Dim Cart As DataTable
  Dim CartView As DataView
  Function CreateDataSource() As ICollection
    Dim dt As DataTable
    Dim dr As DataRow
    Dim i As Integer
    'create a DataTable
    dt = New DataTable
    dt.Columns.Add(New DataColumn("IntegerValue", GetType(Integer)))
    dt.Columns.Add(New DataColumn("StringValue", GetType(String)))
    dt.Columns.Add(New DataColumn("DateTimeValue", GetType(DateTime)))
    dt.Columns.Add(New DataColumn("BoolValue", GetType(Boolean)))
    dt.Columns.Add(new DataColumn("CurrencyValue", GetType(Double)))
    'Make some rows and put some sample data in
    For i = 1 To 9
      dr = dt.NewRow()
      dr(0) = i
      dr(1) = "Item " & i.ToString()
      dr(2) = DateTime.Now.ToShortTimeString
      If (i Mod 2 <> 0) Then
        dr(3) = True
      Else
        dr(3) = False
      End If
      dr(4) = 1.23 * (i + 1)
      'add the row to the datatable
      dt.Rows.Add(dr)
    Next
    'return a DataView to the DataTable
    CreateDataSource = New DataView(dt)
  End Function
  Sub Page_Load(sender As Object, e As EventArgs)
    If Session("DG5VB_ShoppingCart") Is Nothing Then
      Cart = New DataTable()
      Cart.Columns.Add(new DataColumn("Item", GetType(string)))
      Cart.Columns.Add(new DataColumn("Price", GetType(string)))
      Session("DG5VB_ShoppingCart") = Cart
    Else
      Cart = Session("DG5VB_ShoppingCart")
    End If
    CartView = New DataView(Cart)
    ShoppingCart.DataSource = CartView
    CartView.Sort="Item"
   ShoppingCart.DataBind
    MyDataGrid.DataSource = CreateDataSource()
    MyDataGrid.DataBind
  End Sub
  Sub Grid_CartCommand(sender As Object, e As DataGridCommandEventArgs)
    Dim dr As DataRow = Cart.NewRow()
    ' e.Item is the row of the table where the command fired
    ' For bound columns the value is stored in the Text property of TableCell
    Dim itemCell As TableCell = e.Item.Cells(1)
    Dim priceCell As TableCell = e.Item.Cells(2)
    Dim item As String = itemCell.Text
```

```
      Dim price As String = priceCell.Text
      If e.CommandSource.CommandName = "AddToCart" Then
         dr(0) = item
         dr(1) = price
         Cart.Rows.Add(dr)
      Else   'Remove from Cart
         CartView.RowFilter = "Item='" & item & "'"
         If CartView.Count > 0 Then
            CartView.Delete(0)
         End If
         CartView.RowFilter = ""
      End If
      ShoppingCart.DataBind()
   End Sub
</script>
<body>
   <h3><font face="Verdana">Using a Template Column in
DataGrid</font></h3>
   <form runat=server>
   <table cellpadding="5">
   <tr valign="top">
   <td>
   <b>Product List</b>
   <asp:DataGrid id="MyDataGrid" runat="server"
      BorderColor="black"
      BorderWidth="1"
      GridLines="Both"
      CellPadding="3"
      CellSpacing="0"
      Font-Name="Verdana"
      Font-Size="8pt"
      HeaderStyle-BackColor="#aaaadd"
      AutoGenerateColumns="false"
      OnItemCommand="Grid_CartCommand"
      >
      <Columns>
         <asp:TemplateColumn HeaderText="Add/Remove">
            <ItemTemplate>
               <asp:LinkButton ID=AddButton Text="Add"
CommandName="AddToCart" ForeColor="blue" runat="server"
/>&nbsp;
               <asp:LinkButton ID=RemoveButton Text="Remove"
CommandName="RemoveFromCart" ForeColor="blue" runat="server" />
            </ItemTemplate>
         </asp:TemplateColumn>
        <asp:BoundColumn HeaderText="Item"
DataField="StringValue"/>
```

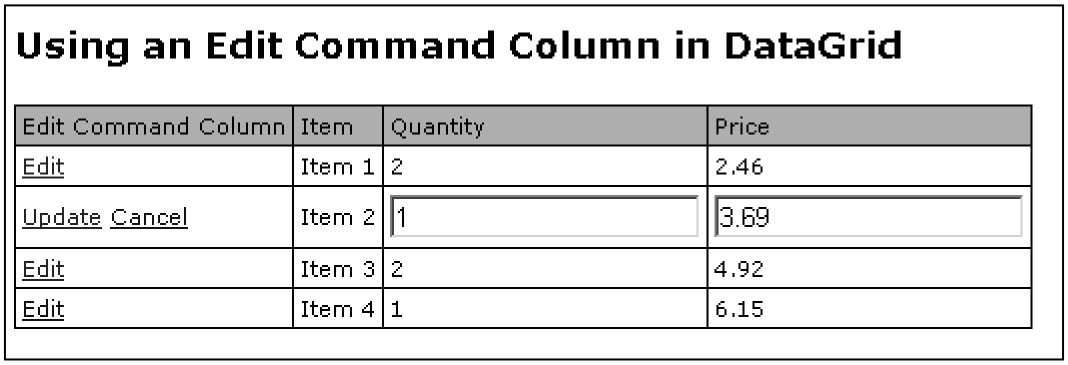
```
        <asp:BoundColumn HeaderText="Price" DataField="CurrencyValue" DataFormatString="{0:c}" ItemStyle-HorizontalAlign="right" />
        <asp:TemplateColumn HeaderText="Assembly required?">
          <ItemTemplate>
            <asp:CheckBox ID=Chk1 Checked='<%# DataBinder.Eval(Container.DataItem, "BoolValue") %>' Enabled="false" runat="server" />
          </ItemTemplate>
        </asp:TemplateColumn>
      </Columns>
    </asp:DataGrid>
    </td><td>
    <b>Shopping Cart</b>
    <asp:DataGrid id="ShoppingCart" runat="server"
      BorderColor="black"
      BorderWidth="1"
      CellPadding="3"
      Font-Name="Verdana"
      Font-Size="8pt"
      HeaderStyle-BackColor="#aaaadd"
      />
    </td>
    </tr>
    </table>
  </form>
</body>
</html>
```

The **EditCommandColumn** is a special column type that supports in-place editing of the data in one row in the grid. **EditCommandColumn** interacts with another property of the grid: **EditItemIndex**. By default the value of **EditItemIndex** is -1, meaning none of the rows (items) in the grid is being edited. If **EditItemIndex** is -1, an "edit" button is displayed in the **EditCommandColumn** for each of the rows in the grid.

When the "edit" button is clicked, the grid's **EditCommand** event is thrown. It's up to you to handle this event in your code. The typical logic sets **EditItemIndex** to the selected row, and then rebinds the data to the grid.

When **EditItemIndex** is set to a particular row, the **EditCommandColumn** displays "update" and "cancel" buttons for that row ("edit" is still displayed for the other rows). These buttons cause the **UpdateCommand** and **CancelCommand** event to be thrown, respectively. The following sample demonstrates this functionality.

Figure 7.6 Editcommandcolumn.aspx



Using an Edit Command Column in DataGrid

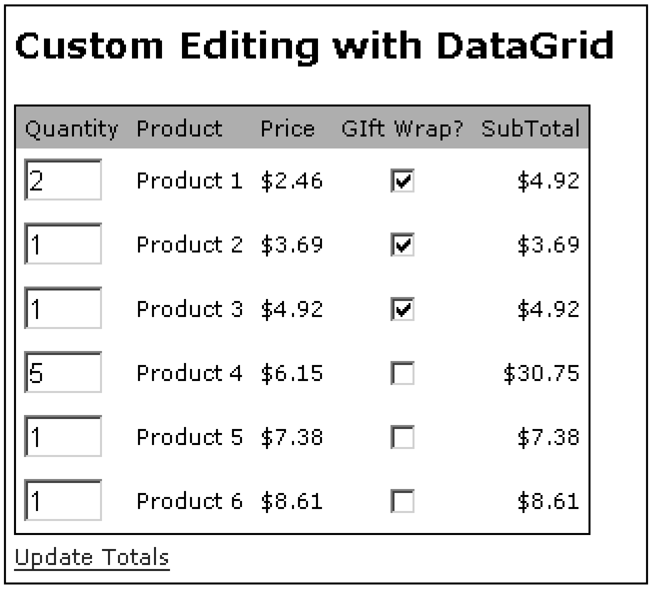
Edit Command Column	Item	Quantity	Price
Edit	Item 1	2	2.46
Update Cancel	Item 2	1	3.69
Edit	Item 3	2	4.92
Edit	Item 4	1	6.15

Editing Data in DataGrid

In the previous example, the **EditCommandColumn** was used to support in-place editing of a single row of data. When you use **EditItemIndex**, the grid automatically inserts the values to be edited into **TextBox** and **CheckBox** controls.

By using **TemplateColumn** objects for the fields you want to edit, you can precisely control how the data is edited. In the following example, the Quantity and Gift Wrap fields are editable in all rows. When the "Update Totals" button is clicked, the grid's Items collection is traversed to extract the current values for these fields, and the data source is updated.

Figure 7.7 CustomEdit.aspx



Custom Editing with DataGrid

Quantity	Product	Price	GIft Wrap?	SubTotal
2	Product 1	$2.46	✓	$4.92
1	Product 2	$3.69	✓	$3.69
1	Product 3	$4.92	✓	$4.92
5	Product 4	$6.15	☐	$30.75
1	Product 5	$7.38	☐	$7.38
1	Product 6	$8.61	☐	$8.61

Code for Figure 7.7 CustomEdit.aspx

```
<%@ Import Namespace="System.Data" %>
```

Coalesce

```
<html>
<script language="VB" runat="server">
  Dim CartView As DataView
  Dim runningTotal As Double = 0
  'Cart is a property on the Page
  ReadOnly Property Cart As DataTable
    Get
      Dim tmpCart As DataTable
      Dim i As Integer
      Dim dr As DataRow
      If Session("DG_ShoppingCart") Is Nothing Then
        tmpCart = new DataTable()
        tmpCart.Columns.Add(new DataColumn("Qty", GetType(String)))
        tmpCart.Columns.Add(new DataColumn("Product", GetType(String)))
        tmpCart.Columns.Add(new DataColumn("Price", GetType(Double)))
        tmpCart.Columns.Add(new                    DataColumn("GiftWrap",
GetType(Boolean)))
        Session("DG_ShoppingCart") = tmpCart

        ' first load -- prepopulate with some data
        For i= 1 to 6
          dr = tmpCart.NewRow()
          dr(0) = "1"
          dr(1) = "Product " & i.ToString
          dr(2) = 1.23 * (i+1)
          dr(3) = false
          tmpCart.Rows.Add(dr)
        Next
        Return tmpCart
      Else
        Return Session("DG_ShoppingCart")
      End If
    End Get
  End Property

  'Sub Page_Init(sender As Object, e As EventArgs)
  '   MyDataGrid.EnableViewState = true
  'End Sub

  Sub Page_Load(sender As Object, e As EventArgs)
    CartView = Cart.DefaultView
    If Not IsPostBack Then
      BindGrid
    End If
  End Sub

  Sub BindGrid()
```

```
      MyDataGrid.DataSource = CartView
      MyDataGrid.DataBind()
   End Sub

   Sub btnUpdate_click(sender As Object, e As EventArgs)
      Dim i As Integer
      Dim _item As DataGridItem
      Dim dr As DataRow
      For i = 0 To MyDataGrid.Items.Count - 1
         _item = MyDataGrid.Items(i)
         Dim      qtyTextBox      As      System.Web.UI.WebControls.TextBox=
_item.FindControl("txtQty")
         Dim giftCheckBox As CheckBox = _item.FindControl("chkGIft")

         ' with a database, we'd use an update command.
         ' since this is an in-memory datatable, we'll just change the in-memory row.
         dr = Cart.Rows(i)
         dr(0) = qtyTextBox.Text
         dr(3) = giftCheckBox.Checked
      Next
      BindGrid
   End Sub

   Function CalcTotal (count As Integer, price As Double) As Double
      Dim total As Double
      total = count * price
      runningTotal += total
      CalcTotal = total
   End Function
</script>
<body>
   <h3><font face="Verdana">Custom Editing with DataGrid</font></h3>
   <form runat=server>
    <ASP:DataGrid id="MyDataGrid" runat="server"
      BorderColor="black"
      BorderWidth="1"
      GridLines="none"
      CellPadding="4"
      Font-Name="Verdana"
      Font-Size="8pt"
      HeaderStyle-BackColor="#aaaadd"
      AutoGenerateColumns="false"
    >
       <Columns>
         <asp:TemplateColumn HeaderText="Quantity">
            <ItemTemplate>
               <asp:TextBox id=txtQty runat="server"
```

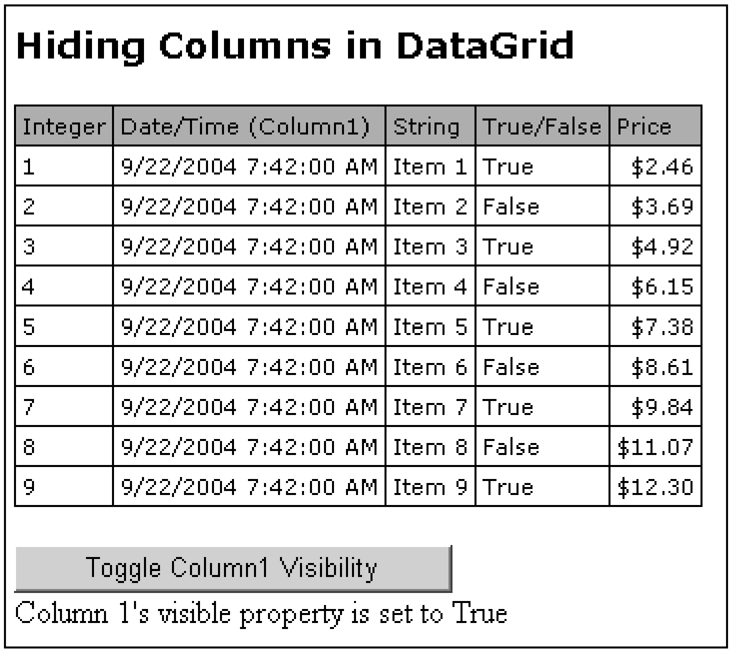
```
                    Text='<%#   DataBinder.Eval(Container.DataItem,    "Qty")
%>'
                    Width="40px"
                    />
                </ItemTemplate>
            </asp:TemplateColumn>
            <asp:BoundColumn                              HeaderText="Product"
DataField="Product"/>
            <asp:BoundColumn     HeaderText="Price"     DataField="Price"
DataFormatString="{0:c}" />
            <asp:TemplateColumn HeaderText="GIft Wrap?">
                <ItemTemplate>
                    <center>
                    <asp:CheckBox id=chkGIft runat="server"
                        Checked='<%#           DataBinder.Eval(Container.DataItem,
"GiftWrap") %>'
                        />
                    </center>
                </ItemTemplate>
            </asp:TemplateColumn>
            <asp:TemplateColumn HeaderText="SubTotal">
                <ItemTemplate>
                    <p align="right">
                    <asp:Label runat="server"
                        Text='<%# System.String.Format("{0:c}", _
                         CalcTotal(Int32.Parse(DataBinder.Eval(Container.DataItem,
"Qty")), _                    DataBinder.Eval(Container.DataItem, "Price")))
%>'
                        />
                    </p>
                </ItemTemplate>
                <FooterTemplate>
                    <p align="right"><b>
                    <asp:Label runat="server"
                        Text='<%# System.String.Format("{0:c}", runningTotal) %>'
                        />
                    </b></p>
               </FooterTemplate>
            </asp:TemplateColumn>
          </Columns>
      </asp:DataGrid>
    <asp:LinkButton id=btnUpdate runat="server"
        Text="Update Totals"
        Font-Name="Verdana"
        Font-Size="8pt"
        onClick="btnUpdate_click"
        />
```

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

Hiding Columns in DataGrid

Each column in the grid has a **Visible** property. Setting **Visible** to **false** hides a column.

Figure 7.7 HidingColumn.aspx



Integer	Date/Time (Column1)	String	True/False	Price
1	9/22/2004 7:42:00 AM	Item 1	True	$2.46
2	9/22/2004 7:42:00 AM	Item 2	False	$3.69
3	9/22/2004 7:42:00 AM	Item 3	True	$4.92
4	9/22/2004 7:42:00 AM	Item 4	False	$6.15
5	9/22/2004 7:42:00 AM	Item 5	True	$7.38
6	9/22/2004 7:42:00 AM	Item 6	False	$8.61
7	9/22/2004 7:42:00 AM	Item 7	True	$9.84
8	9/22/2004 7:42:00 AM	Item 8	False	$11.07
9	9/22/2004 7:42:00 AM	Item 9	True	$12.30

Column 1's visible property is set to True

Code for Figure 7.7 HidingColumn.aspx

```
<%@ Import Namespace="System.Data" %>

<html>
<script language="VB" runat="server">

   Function CreateDataSource() As ICollection

      Dim dt As DataTable
      Dim dr As DataRow
      Dim i As Integer

      'create a DataTable
      dt = New DataTable
      dt.Columns.Add(New DataColumn("IntegerValue", GetType(Integer)))
```

```
        dt.Columns.Add(New DataColumn("StringValue", GetType(String)))
        dt.Columns.Add(New DataColumn("DateTimeValue", GetType(DateTime)))
        dt.Columns.Add(New DataColumn("BoolValue", GetType(Boolean)))
        dt.Columns.Add(new DataColumn("CurrencyValue", GetType(Double)))

        'Make some rows and put some sample data in
        For i = 1 To 9
            dr = dt.NewRow()
            dr(0) = i
            dr(1) = "Item " & i.ToString()
            dr(2) = DateTime.Now.ToShortTimeString
            If (i Mod 2 <> 0) Then
                dr(3) = True
            Else
                dr(3) = False
            End If
            dr(4) = 1.23 * (i + 1)
            'add the row to the datatable
            dt.Rows.Add(dr)
        Next

        'return a DataView to the DataTable
        CreateDataSource = New DataView(dt)

    End Function

    Sub Page_Load(sender As Object, e As EventArgs)

        If Not IsPostBack Then
            MyDataGrid.DataSource = CreateDataSource()
            MyDataGrid.DataBind
        End If

    End Sub

    Sub Button1_Col1Vis(sender As Object, e As EventArgs)

        MyDataGrid.Columns(1).Visible = Not MyDataGrid.Columns(1).Visible
        Label1.Text = "Column 1's visible property is set to " &
MyDataGrid.Columns(1).Visible.ToString

    End Sub

</script>

<body>
```

```
   <h3><font face="Verdana">Hiding Columns in DataGrid</font></h3>

   <form runat=server>

     <ASP:DataGrid id="MyDataGrid" runat="server"
       BorderColor="black"
       BorderWidth="1"
       GridLines="Both"
       CellPadding="3"
       CellSpacing="0"
       Font-Name="Verdana"
       Font-Size="8pt"
       HeaderStyle-BackColor="#aaaadd"
       AutoGenerateColumns="false"
     >
        <Columns>
          <asp:BoundColumn HeaderText="Integer" DataField="IntegerValue" />
          <asp:BoundColumn            HeaderText="Date/Time            (Column1)"
DataField="DateTimeValue"/>
          <asp:BoundColumn HeaderText="String" DataField="StringValue"/>
          <asp:BoundColumn HeaderText="True/False" DataField="BoolValue"/>
          <asp:BoundColumn       HeaderText="Price"       DataField="CurrencyValue"
DataFormatString="{0:c}" ItemStyle-HorizontalAlign="right" />
        </Columns>
     </asp:DataGrid>

     <br>
     <asp:Button       id="Button1"       Text="Toggle       Column1       Visibility"
OnClick="Button1_Col1Vis" runat="server" />

     <br>
     <asp:Label id="Label1" runat="server" />
  </form>

</body>
</html>
```

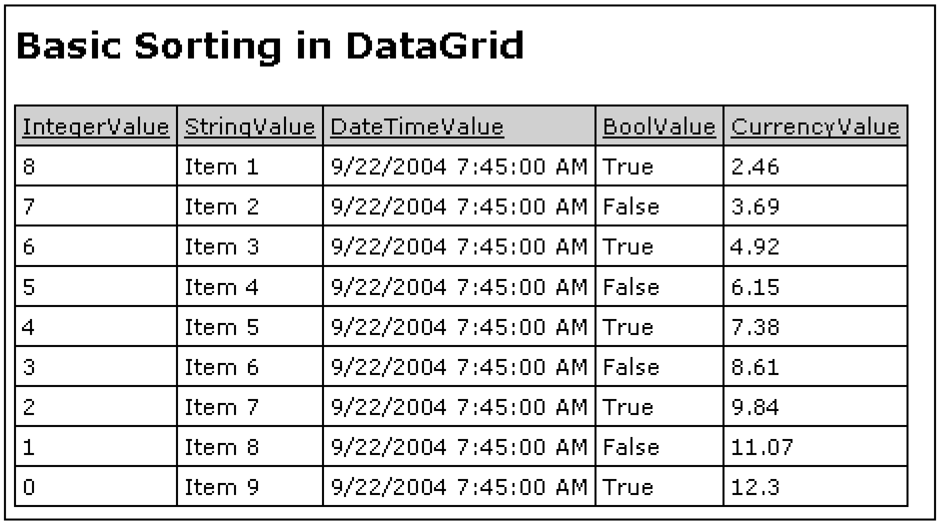
Sorting Columns in DataGrid

Data in a grid is commonly sorted by clicking the header of the column you wish to sort. You can enable sorting in **DataGrid** by setting **AllowSorting** to **true**. When enabled, the grid renders **LinkButton** controls in the header for each column. When the button is clicked, the grid's **SortCommand** event is thrown. It's up to you to handle this event in your code. Because **DataGrid** always displays the data in the same order it occurs in the data source, the typical logic sorts the data source, and then rebinds the data to the grid.

Coalesce

The following example shows how to implement simple sorting in **DataGrid**.

Figure 7.8 SortDatagrid.aspx



Basic Sorting in DataGrid

IntegerValue	StringValue	DateTimeValue	BoolValue	CurrencyValue
8	Item 1	9/22/2004 7:45:00 AM	True	2.46
7	Item 2	9/22/2004 7:45:00 AM	False	3.69
6	Item 3	9/22/2004 7:45:00 AM	True	4.92
5	Item 4	9/22/2004 7:45:00 AM	False	6.15
4	Item 5	9/22/2004 7:45:00 AM	True	7.38
3	Item 6	9/22/2004 7:45:00 AM	False	8.61
2	Item 7	9/22/2004 7:45:00 AM	True	9.84
1	Item 8	9/22/2004 7:45:00 AM	False	11.07
0	Item 9	9/22/2004 7:45:00 AM	True	12.3

Code for Figure 7.8 SortDatagrid.aspx

```
<%@ Import Namespace="System.Data" %>

<html>
<script language="VB" runat="server">

   Dim SortField As String

   Function CreateDataSource() As ICollection

      Dim dt As DataTable
      Dim dr As DataRow
      Dim i As Integer

      'create a DataTable
      dt = New DataTable
      dt.Columns.Add(New DataColumn("IntegerValue", GetType(Integer)))
      dt.Columns.Add(New DataColumn("StringValue", GetType(String)))
      dt.Columns.Add(New DataColumn("DateTimeValue", GetType(DateTime)))
      dt.Columns.Add(New DataColumn("BoolValue", GetType(Boolean)))
      dt.Columns.Add(new DataColumn("CurrencyValue", GetType(Double)))

      'Make some rows and put some sample data in
      For i = 1 To 9
         dr = dt.NewRow()
         dr(0) = 9-i
```

```
            dr(1) = "Item " & i.ToString()
            dr(2) = DateTime.Now.ToShortTimeString
            If (i Mod 2 <> 0) Then
               dr(3) = True
            Else
               dr(3) = False
            End If
            dr(4) = 1.23 * (i + 1)
            'add the row to the datatable
            dt.Rows.Add(dr)
         Next

         'return a DataView to the DataTable
         Dim dv as DataView = New DataView(dt)
         dv.Sort = SortField
         CreateDataSource = dv
      End Function

      Sub Page_Load(sender As Object, e As EventArgs)
         If Not IsPostBack Then
            If SortField = "" Then
               SortField = "IntegerValue"
            End If
            BindGrid
         End If
      End Sub

      Sub      MyDataGrid_Sort(sender      As      Object,      e      As
DataGridSortCommandEventArgs)
         SortField = e.SortExpression
         BindGrid
      End Sub

      Sub BindGrid()
         MyDataGrid.DataSource = CreateDataSource()
         MyDataGrid.DataBind
      End Sub


</script>

<body>

   <h3><font face="Verdana">Basic Sorting in DataGrid</font></h3>

   <form runat=server>
```

```
    <ASP:DataGrid id="MyDataGrid" runat="server"
     AllowSorting="true"
     OnSortCommand="MyDataGrid_Sort"
     BorderColor="black"
     BorderWidth="1"
     CellPadding="3"
     Font-Name="Verdana"
     Font-Size="8pt"
     HeaderStyle-BackColor="#ccccff"
     HeaderStyle-ForeColor="black"
    />

  </form>

</body>
</html>
```

Data Access and Customization

Introduction to Templated Controls

While the **DataGrid** server control demonstrated in the previous section is suitable for many Web application scenarios where a grid-like representation of data is appropriate, many times the presentation of data needs to be much richer. ASP.NET offers two controls, **DataList** and **Repeater**, that give you greater flexibility over the rendering of list-like data. These controls are template-based, and so have no default rendering of their own. The way data is rendered is completely determined by the your implementation of the control's templates, which describe how to present data items.

Like the **DataGrid** control, **DataList** and **Repeater** support a **DataSource** property, which can be set to any **ICollection**, **IEnumerable**, or **IListSource** type. The data in this **DataSource** is bound to the control using its **DataBind** method. Once the data is bound, the format of each data item is described by a template.

The **ItemTemplate** property controls the rendering of each item in the DataSource collection. Inside an **ItemTemplate**, you can define any arbitrary presentation code (HTML or otherwise). Using the ASP.NET data binding syntax, you can insert values from the data bound to the **DataList** or **Repeater** control, as shown in the following example.

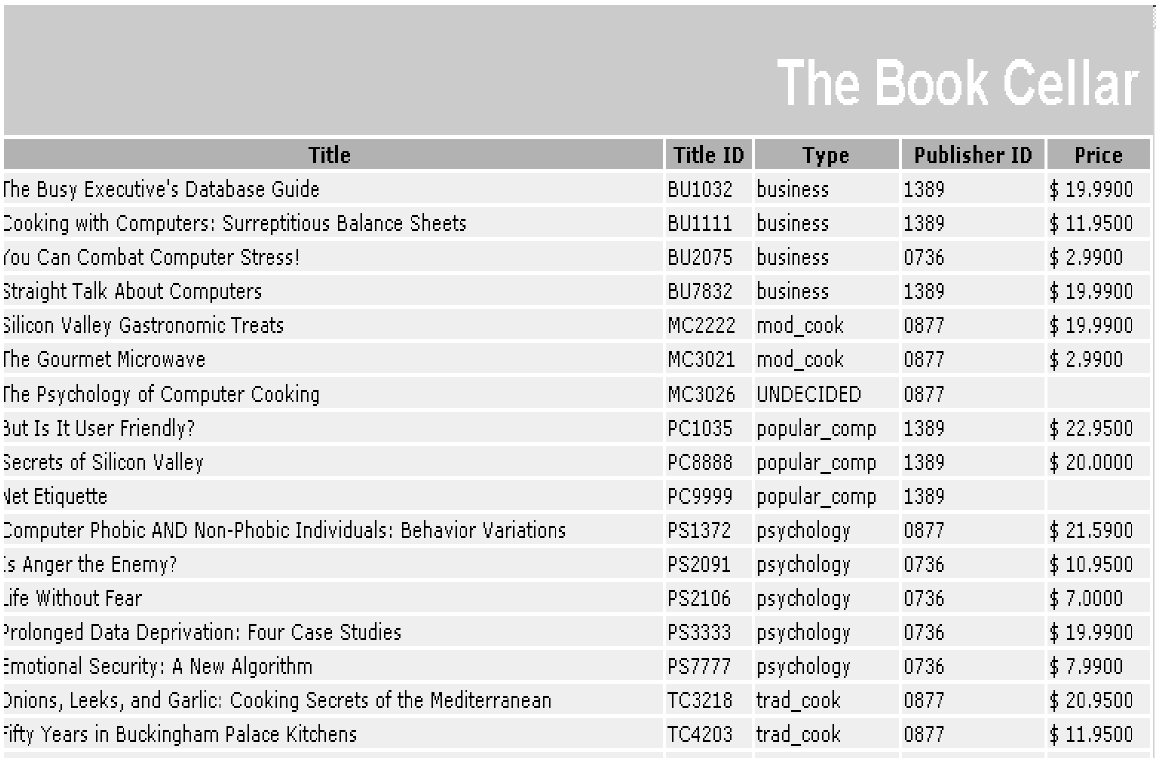
```
<ASP:Repeater id="MyRepeater" runat="server">
   <ItemTemplate>
      Hello <%# DataBinder.Eval(Container.DataItem, "name") %> !
```

```
    </ItemTemplate>
</ASP:Repeater>
```

The **Container** represents the first control in the immediate hierarchy that supports the **System.Web.UI.INamingContainer** marker interface. In this case, the **Container** resolves to an object of type **System.Web.UI.WebControls.RepeaterItem**, which has a **DataItem** property. As the **Repeater** iterates over the DataSource collection, the **DataItem** contains the current item in this collection. For example, if the data source is set to an **ArrayList** of Employee objects, the **DataItem** is of type Employees. When bound to a **DataView**, the **DataItem** is of type **DataRowView**.

The following example demonstrates a **Repeater** control bound to a **DataView** (returned from a SQL query). **HeaderTemplate** and **FooterTemplate** have also been defined and render at the beginning and end of the list, respectively.

Figure 7.8 RepeaterDataview.aspx



The Book Cellar

Title	Title ID	Type	Publisher ID	Price
The Busy Executive's Database Guide	BU1032	business	1389	$ 19.9900
Cooking with Computers: Surreptitious Balance Sheets	BU1111	business	1389	$ 11.9500
You Can Combat Computer Stress!	BU2075	business	0736	$ 2.9900
Straight Talk About Computers	BU7832	business	1389	$ 19.9900
Silicon Valley Gastronomic Treats	MC2222	mod_cook	0877	$ 19.9900
The Gourmet Microwave	MC3021	mod_cook	0877	$ 2.9900
The Psychology of Computer Cooking	MC3026	UNDECIDED	0877	
But Is It User Friendly?	PC1035	popular_comp	1389	$ 22.9500
Secrets of Silicon Valley	PC8888	popular_comp	1389	$ 20.0000
Net Etiquette	PC9999	popular_comp	1389	
Computer Phobic AND Non-Phobic Individuals: Behavior Variations	PS1372	psychology	0877	$ 21.5900
Is Anger the Enemy?	PS2091	psychology	0736	$ 10.9500
Life Without Fear	PS2106	psychology	0736	$ 7.0000
Prolonged Data Deprivation: Four Case Studies	PS3333	psychology	0736	$ 19.9900
Emotional Security: A New Algorithm	PS7777	psychology	0736	$ 7.9900
Onions, Leeks, and Garlic: Cooking Secrets of the Mediterranean	TC3218	trad_cook	0877	$ 20.9500
Fifty Years in Buckingham Palace Kitchens	TC4203	trad_cook	0877	$ 11.9500

Code for Figure 7.8 RepeaterDataview.aspx

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>
<html>
<script language="VB" runat="server">
   Sub Page_Load(Sender As Object, E As EventArgs)
```

```
      Dim DS As DataSet
      Dim MyConnection As SqlConnection
      Dim MyCommand As SqlDataAdapter
      MyConnection                                    =                                    New
SqlConnection(System.Configuration.ConfigurationSettings.AppSettings("PubsString"))
      MyCommand = New SqlDataAdapter("select * from Titles", MyConnection)

      DS = New DataSet()
      MyCommand.Fill(DS, "Titles")

      MyRepeater.DataSource = DS.Tables("Titles").DefaultView
      MyRepeater.DataBind()
   End Sub

</script>

<body topmargin="0" leftmargin="0" marginwidth="0" marginheight="0">

  <!--                                                                              #include
virtual="/quickstart/aspplus/samples/webforms/customize/header.inc" -->

  <ASP:Repeater id="MyRepeater" runat="server">

     <HeaderTemplate>

       <table width="100%" style="font: 8pt verdana">
         <tr style="background-color:DFA894">
           <th>
             Title
           </th>
           <th>
             Title ID
           </th>
           <th>
             Type
           </th>
           <th>
             Publisher ID
           </th>
           <th>
             Price
           </th>
         </tr>

     </HeaderTemplate>

     <ItemTemplate>
```

```
      <tr style="background-color:FFECD8">
        <td>
          <%# DataBinder.Eval(Container.DataItem, "title") %>
        </td>
        <td>
          <%# DataBinder.Eval(Container.DataItem, "title_id") %>
        </td>
        <td>
          <%# DataBinder.Eval(Container.DataItem, "type") %>
        </td>
        <td>
          <%# DataBinder.Eval(Container.DataItem, "pub_id") %>
        </td>
        <td>
          <%# DataBinder.Eval(Container.DataItem, "price", "$ {0}") %>
        </td>
      </tr>

    </ItemTemplate>

    <FooterTemplate>

      </table>

    </FooterTemplate>

  </ASP:Repeater>

  <!--                                                          #include
virtual="/quickstart/aspplus/samples/webforms/customize/footer.inc" -->

</body>
</html>
```

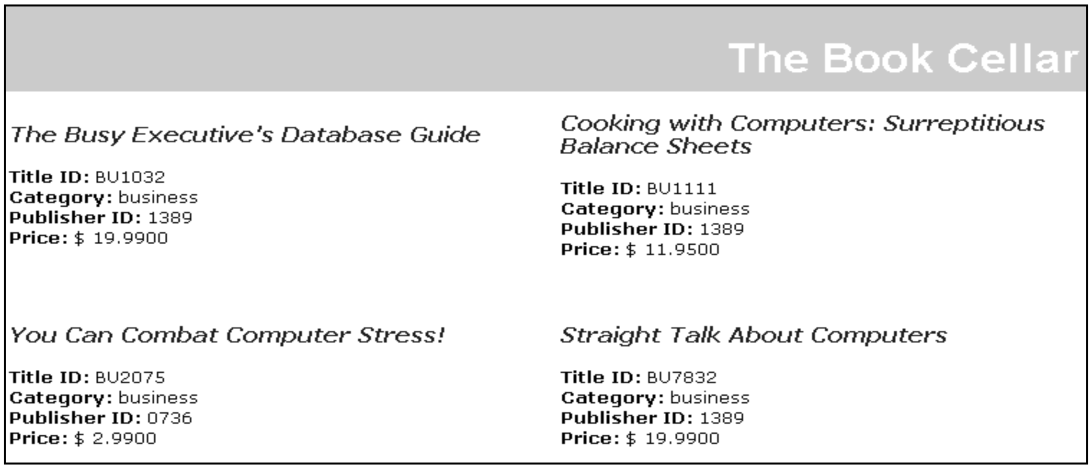
The **Repeater** control just iterates over the bound data, rendering the **ItemTemplate** once for each item in the DataSource collection. It does not render anything besides the elements contained in its templates. While the **Repeater** is a general purpose iterator, the **DataList** provides some additional features for controlling the layout of the list. Unlike the **Repeater**, **DataList** renders additonal elements, like table rows and cells and spans containing style attributes, outside of the template definition to enable this richer formatting. For example, **DataList** supports **RepeatColumns** and **RepeatDirection** properties that specify whether data should be rendered in multiple columns, and in which direction (vertical or horizontal) the data items should be rendered. **DataList** also supports style attributes, as shown in the following example.

```
<ASP:DataList runat="server" DataSource="<%#MyData%>"
   RepeatColumns="2"
   RepeatDirection="Horizontal"
   ItemStyle-Font-Size="10pt"
   ItemStyle-Font-Name="Verdana"
>
   ...

</ASP:DataList>
```

Note: The remainder of this section concentrates on the many features of the **DataList** control. For more information about the **Repeater** control, refer to the Repeater topic in the Web Forms Controls Reference section of this tutorial.

The following sample demonstrates the use of the **DataList** control. Note that the look of the data items has been changed from the previous example, simply by changing the contents of the control's **ItemTemplate** property. The **RepeatDirection** and **RepeatColumns** properties determine how the **ItemTemplates** are laid out.

Figure 7.9 DataList.aspx


The Book Cellar

The Busy Executive's Database Guide

Title ID: BU1032
Category: business
Publisher ID: 1389
Price: $ 19.9900

Cooking with Computers: Surreptitious Balance Sheets

Title ID: BU1111
Category: business
Publisher ID: 1389
Price: $ 11.9500

You Can Combat Computer Stress!

Title ID: BU2075
Category: business
Publisher ID: 0736
Price: $ 2.9900

Straight Talk About Computers

Title ID: BU7832
Category: business
Publisher ID: 1389
Price: $ 19.9900

Code for Figure 7.9 DataList.aspx

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>
<html>
<script language="VB" runat="server">
```

```
   Sub Page_Load(Sender As Object, E As EventArgs)

      Dim DS As DataSet
      Dim MyConnection As SqlConnection
      Dim MyCommand As SqlDataAdapter

      MyConnection = New
SqlConnection(System.Configuration.ConfigurationSettings.AppSettings("PubsString"))
      MyCommand = New SqlDataAdapter("select * from Titles", MyConnection)

      DS = New DataSet()
      MyCommand.Fill(DS, "Titles")

      MyDataList.DataSource = DS.Tables("Titles").DefaultView
      MyDataList.DataBind()
   End Sub
</script>

<body topmargin="0" leftmargin="0" marginwidth="0" marginheight="0">

 <!-- #include
virtual="/quickstart/aspplus/samples/webforms/customize/header.inc" -->

 <ASP:DataList id="MyDataList" RepeatColumns="2"
RepeatDirection="Horizontal" runat="server">

    <ItemTemplate>

      <div style="padding:15,15,15,15;font-size:10pt;font-family:Verdana">

       <div style="font:12pt verdana;color:darkred">
        <i><b><%# DataBinder.Eval(Container.DataItem, "title")
%></i></b>
       </div>

       <br>

       <b>Title ID: </b><%# DataBinder.Eval(Container.DataItem, "title_id")
%><br>
       <b>Category: </b><%# DataBinder.Eval(Container.DataItem, "type")
%><br>
       <b>Publisher ID: </b><%# DataBinder.Eval(Container.DataItem,
"pub_id") %><br>
       <b>Price: </b><%# DataBinder.Eval(Container.DataItem, "price", "$
{0}") %><p>

      </div>
    </ItemTemplate>

  </ASP:DataList>

  <!--                                                              #include
virtual="/quickstart/aspplus/samples/webforms/customize/footer.inc" -->

</body>
</html>
```

The following example further demonstrates the infinite flexibility of templates by changing the **ItemTemplate** yet again. This time, one of the **DataItem** values has been substituted for the "src" attribute of an **<img>** tag. The *format* **String** parameter of **DataBinder.Eval** has also been used to substitute a **DataItem** value in the query string for a URL.

Code for DataBinder.aspx

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>
<html>

<script language="VB" runat="server">

   Sub Page_Load(Sender As Object, E As EventArgs)

       Dim DS As DataSet
       Dim MyConnection As SqlConnection
       Dim MyCommand As SqlDataAdapter

       MyConnection                         =                         New
SqlConnection(System.Configuration.ConfigurationSettings.AppSettings("PubsString"))
       MyCommand = New SqlDataAdapter("select * from Titles", MyConnection)

       DS = New DataSet()
       MyCommand.Fill(DS, "Titles")

       MyDataList.DataSource = DS.Tables("Titles").DefaultView
       MyDataList.DataBind()
   End Sub

</script>

<body topmargin="0" leftmargin="0" marginwidth="0" marginheight="0">
```

```
  <!--                                                   #include
virtual="/quickstart/aspplus/samples/webforms/customize/header.inc" -->

  <ASP:DataList id="MyDataList" RepeatColumns="2" runat="server">

      <ItemTemplate>

        <table cellpadding=10 style="font: 10pt verdana">
          <tr>
            <td width=1 bgcolor="BD8672"/>

            <td valign="top">
              <img align="top" src='<%# DataBinder.Eval(Container.DataItem,
"title_id", "/quickstart/aspplus/images/title-{0}.gif") %>' >
            </td>

            <td valign="top">

              <b>Title: </b><%# DataBinder.Eval(Container.DataItem, "title")
%><br>
              <b>Category: </b><%# DataBinder.Eval(Container.DataItem, "type")
%><br>
              <b>Publisher ID: </b><%# DataBinder.Eval(Container.DataItem,
"pub_id") %><br>
              <b>Price: </b><%# DataBinder.Eval(Container.DataItem, "price", "$
{0}") %>

              <p>

              <a href='<%# DataBinder.Eval(Container.DataItem, "title_id",
"purchase.aspx?titleid={0}") %>' >
                <img border="0" src="/quickstart/aspplus/images/purchase_book.gif" >
              </a>

            </td>
          </tr>
        </table>

      </ItemTemplate>

  </ASP:DataList>

  <!--                                                   #include
virtual="/quickstart/aspplus/samples/webforms/customize/footer.inc" -->
</body>
</html>
```

Handling Postbacks from a Template

As in the **DataGrid**, you can fire a command from inside a **DataList** template that is passed to an event handler wired to the **DataList** itself. For example, a **LinkButton** inside the **ItemTemplate** might fire a **Select** command. By setting the **OnSelectedIndexChanged** property of the **DataList**, you can call an event handler in response to this command. The following example demonstrates this process.

```
<ASP:DataList id="MyDataList" OnSelectedIndexChanged="MyDataList_Select"
runat="server">

   <ItemTemplate>

      <asp:linkbutton CommandName="Select" runat="server">
         <%# DataBinder.Eval(Container.DataItem, "title") %>
      </asp:linkbutton>

   </ItemTemplate>

</ASP:DataList>
```

The following sample demonstrates this code in action. In the MyDataList_Select event handler, you populate several other server controls with the details about the particular selected item.

Code for DataList1.aspx

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>

<html>

<script language="VB" runat="server">

   Dim MyConnection As SqlConnection

   Sub Page_Load(Sender As Object, E As EventArgs)

      MyConnection                              =                              New
SqlConnection(System.Configuration.ConfigurationSettings.AppSettings("PubsString"))

      If Not (Page.IsPostBack)

         Dim DS As New DataSet
```

```
          Dim MyCommand As New SqlDataAdapter("select * from Titles where type =
'business'", MyConnection)

          MyCommand.Fill(DS, "Titles")

          MyDataList.DataSource = DS.Tables("Titles").DefaultView
          MyDataList.DataBind()

       End If
    End Sub

    Sub MyDataList_Select(Sender As Object, E As EventArgs)

       Dim              Title              As              String              =
MyDataList.DataKeys(MyDataList.SelectedItem.ItemIndex)
       Dim MyCommand As New SqlDataAdapter("select * from Titles where title_id =
'" & Title & "'" , MyConnection)

       Dim DS As New DataSet
       MyCommand.Fill(DS, "TitleDetails")

       Dim RowView As DataRowView = DS.Tables("TitleDetails").DefaultView(0)

       DetailsImage.Src = "/quickstart/aspplus/images/title-" & RowView("title_id") &
".gif"
       DetailsPubId.InnerHtml     =     "<b>Publisher     ID:     </b>"     &
RowView("pub_id").ToString() & "<br>"
       DetailsTitleId.InnerHtml     =     "<b>Title     ID:     </b>"     &
RowView("title_id").ToString() & "<br>"
       DetailsType.InnerHtml = "<b>Category: </b>" & RowView("type").ToString()
+ "<br>"
       DetailsPrice.InnerHtml = "<b>Price: </b> $ " & RowView("price").ToString()
+ "<p>"
       PurchaseLink.InnerHtml          =          "<img          border='0'
src='/quickstart/aspplus/images/purchase_book.gif' >"
       PurchaseLink.HRef="purchase.aspx?titleid=" & RowView("title_id").ToString()
       DetailsTitle.InnerHtml = RowView("title").ToString()

       DetailsImage.Visible = true

    End Sub

</script>

<body topmargin="0" leftmargin="0" marginwidth="0" marginheight="0">

  <form runat="server">
```

Coalesce

```
 <!-- #include virtual="/quickstart/aspplus/samples/webforms/customize/header.inc" -->

 <table width="100%">
   <tr>
     <td width="50%">

       <ASP:DataList id="MyDataList" OnSelectedIndexChanged="MyDataList_Select" DataKeyField="title_id" runat="server">

         <ItemTemplate>

           <table cellpadding=10 style="font: 10pt verdana">
             <tr>
               <td valign="top">
                 <img align="top" width="25" border=1 src='<%# DataBinder.Eval(Container.DataItem, "title_id", "/quickstart/aspplus/images/title-{0}.gif") %>' runat="server"/>
               </td>
               <td valign="top">
                 <b>Title: </b>
                 <asp:linkbutton Text='<%# DataBinder.Eval(Container.DataItem, "title") %>' CommandName="Select" style="color:darkred" runat="server"/>
                 <br>
                 <b>Price: </b><%# DataBinder.Eval(Container.DataItem, "price", "$ {0}") %><br>
               </td>
             </tr>
           </table>

         </ItemTemplate>

       </ASP:DataList>

     </td>

     <td valign="top" style="padding-top:15" width="50%">
       <table cellpadding="5" width="100%" style="font: 10pt verdana">
         <tr>
           <td>
             <img id="DetailsImage" visible="false" runat="server">
           </td>
           <td valign="top" width="400">
             <div style="font: 12pt verdana;color:darkred">
```

```
          <i><b><span id="DetailsTitle" runat="server"/></i></b><br>
         </div>
         <span id="DetailsTitleId" runat="server"/>
         <span id="DetailsPubId" runat="server"/>
         <span id="DetailsType" runat="server"/>
         <span id="DetailsPrice" runat="server"/>
         <a id="PurchaseLink" runat="server"/>
        </td>
       </tr>
      </table>
     </td>

   </tr>
  </table>

  <!--                                                                   #include
virtual="/quickstart/aspplus/samples/webforms/customize/footer.inc" -->

  </form>

</body>
</html>
```

Note that while the **DataList** recognizes a few special commands such as **Select** and **Edit/Update/Cancel**, the command string fired inside a template can be any arbitrary string. For all commands, the **DataList**'s **OnItemCommand** is fired. You can wire this event to a handler as in the previous example; the following example shows how to do this.

Code for DataList1.aspx

```
<script runat="server">

   Protected Sub MyDataList_ItemCommand(Sender As Object, E As
DataListCommandEventArgs)
      Dim Command As String = E.CommandName

      Select Case Command
         Case "Discuss"
            ShowDiscussions(E.Item.DataItem)
         Case "Ratings"
```

```
            ShowRatings(E.Item.DataItem)
        End Select
    End Sub

</script>

<ASP:DataList id="MyDataList"
OnItemCommand="MyDataList_ItemCommand" runat="server">

    <ItemTemplate>

        <asp:linkbutton CommandName="Ratings" runat="server">
            View Ratings
        </asp:linkbutton>
        |
        <asp:linkbutton CommandName="Discuss" runat="server">
            View Discussions
        </asp:linkbutton>

    </ItemTemplate>

</ASP:DataList>
```

Note that because more than one command can fire this event handler, you must employ a switch statement to determine the particular command that was fired. The following sample demonstrates this code in action.

Code for DataList2.aspx

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>

<html>

<script language="VB" runat="server">

    Dim MyConnection As SqlConnection

    Sub Page_Load(Sender As Object, E As EventArgs)

        MyConnection = New
SqlConnection(System.Configuration.ConfigurationSettings.AppSettings("PubsString"))

        If Not (Page.IsPostBack)

            Dim DS As New DataSet
            Dim MyCommand As New SqlDataAdapter("select * from Titles where type =
'business'", MyConnection)

            MyCommand.Fill(DS, "Titles")

            MyDataList.DataSource = DS.Tables("Titles").DefaultView
            MyDataList.DataBind()

        End If
    End Sub

    Sub MyDataList_Select(Sender As Object, E As EventArgs)

        Dim Title As String =
MyDataList.DataKeys(MyDataList.SelectedItem.ItemIndex)
        Dim MyCommand As New SqlDataAdapter("select * from Titles where title_id =
'" & Title & "'" , MyConnection)

        Dim DS As New DataSet
        MyCommand.Fill(DS, "TitleDetails")

        Dim RowView As DataRowView = DS.Tables("TitleDetails").DefaultView(0)

        DetailsImage.Src = "/quickstart/aspplus/images/title-" & RowView("title_id") &
".gif"
        DetailsPubId.InnerHtml = "<b>Publisher ID: </b>" &
RowView("pub_id").ToString() & "<br>"
        DetailsTitleId.InnerHtml = "<b>Title ID: </b>" &
RowView("title_id").ToString() & "<br>"
        DetailsType.InnerHtml = "<b>Category: </b>" & RowView("type").ToString()
+ "<br>"
        DetailsPrice.InnerHtml = "<b>Price: </b> $ " & RowView("price").ToString()
+ "<p>"
        PurchaseLink.InnerHtml = "<img border='0'
src='/quickstart/aspplus/images/purchase_book.gif' >"
        PurchaseLink.HRef="purchase.aspx?titleid=" & RowView("title_id").ToString()
        DetailsTitle.InnerHtml = RowView("title").ToString()

        DetailsImage.Visible = true

    End Sub

</script>

<body topmargin="0" leftmargin="0" marginwidth="0" marginheight="0">

  <form runat="server">
```

```
 <!-- #include virtual="/quickstart/aspplus/samples/webforms/customize/header.inc" -->

 <table width="100%">
  <tr>
   <td width="50%">

    <ASP:DataList id="MyDataList" OnSelectedIndexChanged="MyDataList_Select" DataKeyField="title_id" runat="server">

     <ItemTemplate>

      <table cellpadding=10 style="font: 10pt verdana">
       <tr>
        <td valign="top">
         <img align="top" width="25" border=1 src='<%# DataBinder.Eval(Container.DataItem, "title_id", "/quickstart/aspplus/images/title-{0}.gif") %>' runat="server"/>
        </td>
        <td valign="top">
         <b>Title: </b>
         <asp:linkbutton Text='<%# DataBinder.Eval(Container.DataItem, "title") %>' CommandName="Select" style="color:darkred" runat="server"/>
         <br>
         <b>Price: </b><%# DataBinder.Eval(Container.DataItem, "price", "$ {0}") %><br>
        </td>
       </tr>
      </table>

     </ItemTemplate>

    </ASP:DataList>

   </td>

   <td valign="top" style="padding-top:15" width="50%">
    <table cellpadding="5" width="100%" style="font: 10pt verdana">
     <tr>
      <td>
       <img id="DetailsImage" visible="false" runat="server">
      </td>
      <td valign="top" width="400">
       <div style="font: 12pt verdana;color:darkred">
```

```
            <i><b><span id="DetailsTitle" runat="server"/></i></b><br>
          </div>
          <span id="DetailsTitleId" runat="server"/>
          <span id="DetailsPubId" runat="server"/>
          <span id="DetailsType" runat="server"/>
          <span id="DetailsPrice" runat="server"/>
          <a id="PurchaseLink" runat="server"/>
        </td>
      </tr>
    </table>
  </td>

 </tr>
</table>

<!--                                                            #include
virtual="/quickstart/aspplus/samples/webforms/customize/footer.inc" -->

</form>

</body>
</html>
```

Using Select and Edit Templates

In addition to handling the **Select** command using a page-level event handler, the **DataList** can respond to this event internally. If a **SelectedItemTemplate** is defined for the **DataList**, the **DataList** renders this template for the item that fired the **Select** command. The following example uses the **SelectedItemTemplate** to make the title of the selected book bold.

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>

<html>

<script language="VB" runat="server">

  Dim MyConnection As SqlConnection

  Sub Page_Load(Sender As Object, E As EventArgs)

    MyConnection                        =                        New
SqlConnection(System.Configuration.ConfigurationSettings.AppSettings("PubsString"))
```

```
        Dim DS As New DataSet
        Dim MyCommand As New SqlDataAdapter("select * from Titles where type =
'business'", MyConnection)

        MyCommand.Fill(DS, "Titles")

        MyDataList.DataSource = DS.Tables("Titles").DefaultView

        If Not (Page.IsPostBack)
           MyDataList.DataBind()
        End If
     End Sub

     Sub MyDataList_Select(Sender As Object, E As EventArgs)

        Dim Title As String =
MyDataList.DataKeys(MyDataList.SelectedItem.ItemIndex)
        Dim MyCommand As New SqlDataAdapter("select * from Titles where title_id =
'" & Title & "'" , MyConnection)

        Dim DS As New DataSet
        MyCommand.Fill(DS, "TitleDetails")

        Dim RowView As DataRowView = DS.Tables("TitleDetails").DefaultView(0)

        DetailsImage.Src = "/quickstart/aspplus/images/title-" & RowView("title_id") &
".gif"
        DetailsPubId.Text = "<b>Publisher ID: </b>" &
RowView("pub_id").ToString() & "<br>"
        DetailsTitleId.Text = "<b>Title ID: </b>" & RowView("title_id").ToString() &
"<br>"
        DetailsType.Text = "<b>Category: </b>" & RowView("type").ToString() +
"<br>"
        DetailsPrice.Text = "<b>Price: </b> $ " & RowView("price").ToString() +
"<p>"
        PurchaseLink.Text = "<img border='0'
src='/quickstart/aspplus/images/purchase_book.gif' >"
        PurchaseLink.NavigateUrl = "purchase.aspx?titleid=" &
RowView("title_id").ToString()
        DetailsTitle.Text = RowView("title").ToString()

        DetailsImage.Visible = true

        MyDataList.DataBind()
     End Sub

     Sub MyDataList_ItemCommand(Sender As Object, E As
DataListCommandEventArgs)

        Dim Title As String = MyDataList.DataKeys(E.Item.ItemIndex)
        Dim MyLinkButton As LinkButton = E.CommandSource

        Select (MyLinkButton.Text)

          Case "Discussions" :
            ShowDiscussions(Title)

          Case "Ratings" :
            ShowRatings(Title)
       End Select
    End Sub

    Sub ShowRatings(Title As String)

        Message.InnerHtml = "<h5>Ratings for """ & Title & """</h5>"
        Message.InnerHtml &= "Print Ratings here..."
    End Sub

    Sub ShowDiscussions(Title As String)

        Message.InnerHtml = "<h5>Discussions for """ & Title & """</h5>"
        Message.InnerHtml &= "Print Discussions here..."
    End Sub

</script>

<body topmargin="0" leftmargin="0" marginwidth="0" marginheight="0">

  <form runat="server">

  <!--                                                           #include
virtual="/quickstart/aspplus/samples/webforms/customize/header.inc" -->

  <table width="100%">
    <tr>
      <td width="50%">

        <ASP:DataList                                    id="MyDataList"
OnSelectedIndexChanged="MyDataList_Select"
OnItemCommand="MyDataList_ItemCommand"     DataKeyField="title_id"
runat="server">

          <ItemTemplate>
          <table cellpadding=10 style="font: 10pt verdana">
            <tr>
              <td valign="top">
                <img align="top" width="25" border=1 src='<%# DataBinder.Eval(Container.DataItem, "title_id", "/quickstart/aspplus/images/title-{0}.gif") %>' runat="server"/>
              </td>
              <td valign="top">
                <b>Title: </b>
                <asp:linkbutton Text='<%# DataBinder.Eval(Container.DataItem, "title") %>' CommandName="Select" style="color:darkred" runat="server"/>
                <br>
                <b>Price: </b><%# DataBinder.Eval(Container.DataItem, "price", "$ {0}") %>
                <br>
                <asp:linkbutton Text="Discussions" CommandName="Discuss" style="color:darkred;font:8pt tahoma" runat="server"/>
                |
                <asp:linkbutton Text="Ratings" CommandName="Ratings" style="color:darkred;font:8pt tahoma" runat="server"/>
              </td>
            </tr>
          </table>

        </ItemTemplate>

        <SelectedItemTemplate>

          <table cellpadding=10 style="font: 10pt verdana" >
            <tr>
              <td valign="top">
                <img src='<%# DataBinder.Eval(Container.DataItem, "title_id", "/quickstart/aspplus/images/title-{0}.gif") %>' align="top" width="25" border=1 runat="server"/>
              </td>
              <td valign="top">
                <b>Title: </b>
                <asp:linkbutton Font-Bold="true" Text='<%# DataBinder.Eval(Container.DataItem, "title") %>' CommandName="Select" style="color:darkred" runat="server"/>
                <br>
                <b>Price: </b><%# DataBinder.Eval(Container.DataItem, "price", "$ {0}") %>
                <br>
                <asp:linkbutton Text="Discussions" Command="Discuss" style="color:darkred;font:8pt tahoma" runat="server"/>
```

```
             |
             <asp:linkbutton           Text="Ratings"          Command="Ratings"
style="color:darkred;font:8pt tahoma" runat="server"/>
           </td>
         </tr>
       </table>

     </SelectedItemTemplate>

   </ASP:DataList>

  </td>

  <td valign="top" style="padding-top:15" width="50%">
    <table cellpadding="5" width="100%" style="font: 10pt verdana">
      <tr>
        <td>
          <img id="DetailsImage" visible="false" runat="server">
        </td>
        <td valign="top" width="400">
          <div style="font: 12pt verdana;color:darkred">
            <i><b><asp:Label                                    id="DetailsTitle"
runat="server"/></i></b><br>
          </div>
          <asp:Label id="DetailsTitleId" runat="server"/>
          <asp:Label id="DetailsPubId" runat="server"/>
          <asp:Label id="DetailsType" runat="server"/>
          <asp:Label id="DetailsPrice" runat="server"/>
          <asp:HyperLink id="PurchaseLink" runat="server"/>
        </td>
      </tr>
    </table>
  </td>

 </tr>
</table>

<!--                                                                  #include
virtual="/quickstart/aspplus/samples/webforms/customize/footer.inc" -->

<div id="Message" style="font: 10pt verdana;padding:0,15,15,15" runat="server"/>

</form>

</body>
</html>
```

Coalesce

DataList also supports an **EditItemTemplate** for rendering an item whose index is equal to the **DataList**'s **EditItemIndex** property. For details about how editing and updating works, refer to the Updating Data topic of the Data Access section of this tutorial.

Code for DataListEditItemTemplate.aspx

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>

<html>

<script language="VB" runat="server">

   Dim MyConnection As SqlConnection

   Sub Page_Load(Sender As Object, E As EventArgs)

      MyConnection = New
SqlConnection(System.Configuration.ConfigurationSettings.AppSettings("PubsString"))

      Dim DS As New DataSet
      Dim MyCommand As New SqlDataAdapter("select * from Titles where type =
'business'", MyConnection)

      MyCommand.Fill(DS, "Titles")

      MyDataList.DataSource = DS.Tables("Titles").DefaultView

      If Not (Page.IsPostBack)
         MyDataList.DataBind()
      End If
   End Sub

   Sub MyDataList_Select(Sender As Object, E As EventArgs)

      Dim Title As String = MyDataList.DataKeys(MyDataList.SelectedItem.ItemIndex)
      Dim MyCommand As New SqlDataAdapter("select * from Titles where title_id = '"
& Title & "'" , MyConnection)

      Dim DS As New DataSet
      MyCommand.Fill(DS, "TitleDetails")

      Dim RowView As DataRowView = DS.Tables("TitleDetails").DefaultView(0)

      DetailsImage.Src = "/quickstart/aspplus/images/title-" & RowView("title_id") &
".gif"
```

```
      DetailsPubId.Text = "<b>Publisher ID: </b>" & RowView("pub_id").ToString()
& "<br>"
      DetailsTitleId.Text = "<b>Title ID: </b>" & RowView("title_id").ToString() &
"<br>"
      DetailsType.Text = "<b>Category: </b>" & RowView("type").ToString() +
"<br>"
      DetailsPrice.Text = "<b>Price: </b> $ " & RowView("price").ToString() + "<p>"
      PurchaseLink.Text = "<img border='0'
src='/quickstart/aspplus/images/purchase_book.gif' >"
      PurchaseLink.NavigateUrl = "purchase.aspx?titleid=" &
RowView("title_id").ToString()
      DetailsTitle.Text = RowView("title").ToString()

      DetailsImage.Visible = true

      MyDataList.DataBind()
   End Sub

   Sub MyDataList_ItemCommand(Sender As Object, E As
DataListCommandEventArgs)

      Dim Title As String = MyDataList.DataKeys(E.Item.ItemIndex)
      Dim MyLinkButton As LinkButton = E.CommandSource

      Select (MyLinkButton.Text)

        Case "Discussions" :
          ShowDiscussions(Title)

        Case "Ratings" :
          ShowRatings(Title)
     End Select
   End Sub

   Sub ShowRatings(Title As String)

      Message.InnerHtml = "<h5>Ratings for """ & Title & """</h5>"
      Message.InnerHtml &= "Print Ratings here..."
   End Sub

   Sub ShowDiscussions(Title As String)

      Message.InnerHtml = "<h5>Discussions for """ & Title & """</h5>"
      Message.InnerHtml &= "Print Discussions here..."
   End Sub

</script>
```

Coalesce

```
<body topmargin="0" leftmargin="0" marginwidth="0" marginheight="0">

  <form runat="server">

  <!-- #include virtual="/quickstart/aspplus/samples/webforms/customize/header.inc" -
->

  <table width="100%">
    <tr>
      <td width="50%">

        <ASP:DataList                                                    id="MyDataList"
OnSelectedIndexChanged="MyDataList_Select"
OnItemCommand="MyDataList_ItemCommand"          DataKeyField="title_id"
runat="server">

          <ItemTemplate>

            <table cellpadding=10 style="font: 10pt verdana">
              <tr>
                <td valign="top">
                  <img       align="top"       width="25"       border=1       src='<%#
DataBinder.Eval(Container.DataItem,                                         "title_id",
"/quickstart/aspplus/images/title-{0}.gif") %>' runat="server"/>
                </td>
                <td valign="top">
                  <b>Title: </b>
                  <asp:linkbutton    Text='<%#    DataBinder.Eval(Container.DataItem,
"title") %>' CommandName="Select" style="color:darkred" runat="server"/>
                  <br>
                  <b>Price: </b><%# DataBinder.Eval(Container.DataItem, "price", "$
{0}") %>
                  <br>
                  <asp:linkbutton    Text="Discussions"    CommandName="Discuss"
style="color:darkred;font:8pt tahoma" runat="server"/>
                  |
                  <asp:linkbutton      Text="Ratings"      CommandName="Ratings"
style="color:darkred;font:8pt tahoma" runat="server"/>
                </td>
              </tr>
            </table>

          </ItemTemplate>

          <SelectedItemTemplate>
            <table cellpadding=10 style="font: 10pt verdana" >
              <tr>
                <td valign="top">
                  <img src='<%# DataBinder.Eval(Container.DataItem, "title_id", "/quickstart/aspplus/images/title-{0}.gif") %>' align="top" width="25" border=1 runat="server"/>
                </td>
                <td valign="top">
                  <b>Title: </b>
                  <asp:linkbutton Font-Bold="true" Text='<%# DataBinder.Eval(Container.DataItem, "title") %>' CommandName="Select" style="color:darkred" runat="server"/>
                  <br>
                  <b>Price: </b><%# DataBinder.Eval(Container.DataItem, "price", "${0}") %>
                  <br>
                  <asp:linkbutton Text="Discussions" Command="Discuss" style="color:darkred;font:8pt tahoma" runat="server"/>
                  |
                  <asp:linkbutton Text="Ratings" Command="Ratings" style="color:darkred;font:8pt tahoma" runat="server"/>
                </td>
              </tr>
            </table>

          </SelectedItemTemplate>

        </ASP:DataList>

      </td>

      <td valign="top" style="padding-top:15" width="50%">
        <table cellpadding="5" width="100%" style="font: 10pt verdana">
          <tr>
            <td>
              <img id="DetailsImage" visible="false" runat="server">
            </td>
            <td valign="top" width="400">
              <div style="font: 12pt verdana;color:darkred">
                <i><b><asp:Label id="DetailsTitle" runat="server"/></i></b><br>
              </div>
              <asp:Label id="DetailsTitleId" runat="server"/>
              <asp:Label id="DetailsPubId" runat="server"/>
              <asp:Label id="DetailsType" runat="server"/>
              <asp:Label id="DetailsPrice" runat="server"/>
              <asp:HyperLink id="PurchaseLink" runat="server"/>
            </td>
        </tr>
       </table>
      </td>

    </tr>
   </table>

   <!-- #include virtual="/quickstart/aspplus/samples/webforms/customize/footer.inc" --
>

   <div id="Message" style="font: 10pt verdana;padding:0,15,15,15" runat="server"/>

   </form>

</body>
</html>
```

Finding a Control Inside a Template

Sometimes it is necessary to locate a control contained inside a template. If a control is given an ID in a template, that control can be retrieved from its container (the first control in the parent hierarchy that supports **INamingContainer**). In this case, the container is the **DataListItem** control. Note that even though there are several controls with the same ID (by virtue of the **DataList**'s repetition), each is contained logically in the namespace of the **DataListItem** container control.

You can go through the **DataList**'s **Items** collection to retrieve the **DataListItem** for a given index, and then call the **DataListItem**'s **FindControl** method (inherited from the base **Control** class) to retrieve a control with a particular ID.

```
   <script runat="server">

      Public Sub Page_Load(sender As Object, E As EventArgs))
         ' set datasource and call databind here

         For I=0 To MyDataList.Items.Count-1
           Dim          IsChecked          As          String          =
   MyDataList.Items(i).FindControl("Save").Checked.ToString()
           If IsChecked = "True" Then
             ...
           End If
         Next
      End Sub
```

```
    </script>

    <ASP:DataList id="MyDataList" runat="server">

       <ItemTemplate>
           <asp:CheckBox id="Save" runat="server"/> <b>Save to Favorites</b>
       </ItemTemplate>

    </ASP:DataList>
```

Code for DataListFindControl.aspx

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>

<html>

<script language="VB" runat="server">

   Sub Page_Load(Src As Object, E As EventArgs)

      If Not (Page.IsPostBack)

         Dim DS As DataSet
         Dim MyConnection As SqlConnection
         Dim MyCommand As SqlDataAdapter

         MyConnection                              =                              New
SqlConnection(System.Configuration.ConfigurationSettings.AppSettings("PubsString"))
         MyCommand = New SqlDataAdapter("select * from Titles where type =
'business'", MyConnection)

         DS = New DataSet()
         MyCommand.Fill(DS, "Titles")

         MyDataList.DataSource = DS.Tables("Titles").DefaultView
         MyDataList.DataBind()
      End If
   End Sub

   Sub Submit_Click(Src As Object, E As EventArgs)
      Dim I As Long

       For I=0 To MyDataList.Items.Count -1

          Dim CurrentCheckBox As CheckBox
         CurrentCheckBox = MyDataList.Items(I).FindControl("Save")
         Message.InnerHtml &= "Item(" & i & "): " &
CurrentCheckBox.Checked.ToString() & "<br>"
       Next
   End Sub

</script>

<body topmargin="0" leftmargin="0" marginwidth="0" marginheight="0">

  <form runat="server">

  <!-- #include
virtual="/quickstart/aspplus/samples/webforms/customize/header.inc" -->

  <ASP:DataList id="MyDataList" RepeatColumns="2" runat="server">

     <ItemTemplate>

       <table cellpadding=10 style="font: 10pt verdana">
         <tr>
           <td width=1 bgcolor="BD8672"/>
           <td valign="top">
             <img align="top" src='<%# DataBinder.Eval(Container.DataItem,
"title_id", "/quickstart/aspplus/images/title-{0}.gif") %>' >
           </td>
           <td valign="top">
             <b>Title: </b><%# DataBinder.Eval(Container.DataItem, "title")
%><br>
             <b>Category: </b><%# DataBinder.Eval(Container.DataItem, "type")
%><br>
             <b>Publisher ID: </b><%# DataBinder.Eval(Container.DataItem,
"pub_id") %><br>
             <b>Price: </b><%# DataBinder.Eval(Container.DataItem, "price", "$
{0}") %>
             <p>
             <asp:CheckBox id="Save" runat="server"/> <b>Save to
Favorites</b>
           </td>
         </tr>
       </table>

     </ItemTemplate>

  </ASP:DataList>
  <p>
  <div style="padding:0,15,0,15">
    <input type="submit" Value="Update Favorites" OnServerClick="Submit_Click"
runat="server"/>
  </div>

  <p>

  <!--                                                          #include
virtual="/quickstart/aspplus/samples/webforms/customize/footer.inc" -->

  </form>

  <div   style="font:   10pt   verdana"   EnableViewState="false"   id="Message"
runat="server"/>

</body>
</html>
```

Section Summary

1. The **DataList** and **Repeater** controls provide developers fine-tuned control over the rendering of data-bound lists.
2. Rendering of bound data is controlled using a template, such as the **HeaderTemplate**, **FooterTemplate**, or **ItemTemplate**.
3. The **Repeater** control is a general-purpose iterator, and does not insert anything in its rendering that is not contained in a template.
4. The **DataList** control offers more control over the layout and style of items, and outputs its own rendering code for formatting.
5. The **DataList** supports the **Select**, **Edit/Update/Cancel**, and **Item Command** events, which can be handled at the page level by wiring event handlers to the **DataList**'s **Command** events.
6. **DataList** supports a **SelectedItemTemplate** and **EditItemTemplate** for control over the rendering of a selected or editable item.
7. Controls can be programmatically retrieved from a template using the **Control.FindControl** method. This should be called on a **DataListItem** retrieved from the **DataList**'s Items collection.

Chapter 8

Using the Global.asax File

The Global.asax File

In addition to writing UI code, developers can also add application level logic and event handling code into their Web applications. This code does not handle generating UI and is typically not invoked in response to individual page requests. Instead, it is responsible for handling higher-level application events such as **Application_Start**, **Application_End**, **Session_Start**, **Session_End**, and so on. Developers author this logic using a **Global.asax** file located at the root of a particular Web application's virtual directory tree. ASP.NET automatically parses and compiles this file into a dynamic .NET Framework class--which extends the **HttpApplication** base class--the first time any resource or URL within the application namespace is activated or requested.

The Global.asax file is parsed and dynamically compiled by ASP.NET into a .NET Framework class the first time any resource or URL within its application namespace is activated or requested. The Global.asax file is configured to automatically reject any direct URL request so that external users cannot download or view the code within.

Application or Session-Scoped Events

Developers can define handlers for events of the **HttpApplication** base class by authoring methods in the Global.asax file that conform to the naming pattern "Application_EventName(AppropriateEventArgumentSignature)". For example:

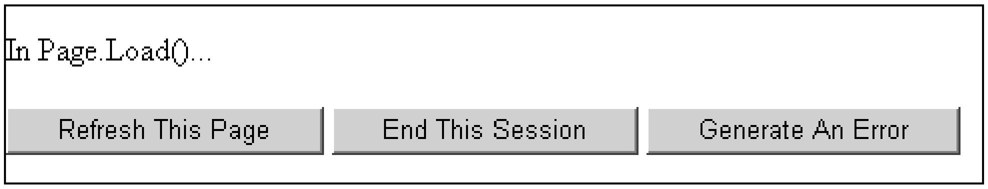
```
    <script language="VB" runat="server">

Sub Application_Start(Sender As Object, E As EventArgs)
  ' Application startup code goes here
End Sub
</script>
```

If the event handling code needs to import additional namespaces, the **@ import** directive can be used on an .aspx page, as follows:

```
 <%@ Import Namespace="System.Text" %>
```

Coalesce

Figure 8.1 Lifetime of Applicatio,Session and Request Object


In Page.Load()...

Code for Figure 8.1

```
<script language="VB" runat="server">

   Sub Application_Start(Sender As Object, E As EventArgs)
      ' Do application startup code here
   End Sub

   Sub Application_End(Sender As Object, E As EventArgs)
      ' Clean up application resources here
   End Sub

   Sub Session_Start(Sender As Object, E As EventArgs)
      Response.Write("Session is Starting...<br>")
   End Sub

   Sub Session_End(Sender As Object, E As EventArgs)
      ' Clean up session resources here
   End Sub

   Sub Application_BeginRequest(Sender As Object, E As EventArgs)
      Response.Write("<h3><font face='Verdana'>Using the Global.asax
File</font></h3>")
      Response.Write("Request is Starting...<br>")
   End Sub

   Sub Application_EndRequest(Sender As Object, E As EventArgs)
      Response.Write("Request is Ending...<br>")
   End Sub

   Sub Application_Error(Sender As Object, E As EventArgs)
      Context.ClearError()
      Response.Redirect("errorpage.htm")
   End Sub

</script>
```

The first time the page is opened, the **Start** event is raised for the application and the session:

Code for Application_Session_StartEvent.aspx

```
Sub Application_Start(Sender As Object, E As EventArgs)
  ' Application startup code goes here
End Sub

Sub Session_Start(Sender As Object, E As EventArgs)
  Response.Write("Session is Starting...<br>")
  Session.Timeout = 1
End Sub
```

The **BeginRequest** and **EndRequest** events are raised on each request. When the page is refreshed, only messages from **BeginRequest**, **EndRequest**, and the **Page_Load** method will appear. Note that by abandoning the current session (click the "End this session" button) a new session is created and the **Session_Start** event is raised again.

Application or Session-Scoped Objects

Static objects, .NET Framework classes, and COM components all can be defined in the Global.asax file using the object tag. The scope can be **appinstance**, **session**, or **application**. The **appinstance** scope denotes that the object is specific to one instance of **HttpApplication** and is not shared.

```
<object id="id" runat="server" class=".NET Framework class Name" scope="appinstance"/>
<object id="id" runat="server" progid="COM ProgID" scope="session"/>
<object id="id" runat="server" classid="COM ClassID" scope="application"/>
```

Section Summary

1. ASP.NET Framework applications can define event handlers with application-wide or session-wide scope in the Global.asax file.
2. ASP.NET Framework applications can define objects with application-wide or session-wide scope in the Global.asax file.

```
<script language="VB" runat="server">

  Sub Application_Start(Sender As Object, E As EventArgs)
     ' Do application startup code here
  End Sub
```

```
   Sub Application_End(Sender As Object, E As EventArgs)
      ' Clean up application resources here
   End Sub

   Sub Session_Start(Sender As Object, E As EventArgs)
      Response.Write("Session is Starting...<br>")
   End Sub

   Sub Session_End(Sender As Object, E As EventArgs)
      ' Clean up session resources here
   End Sub

   Sub Application_BeginRequest(Sender As Object, E As EventArgs)
      Response.Write("<h3><font face='Verdana'>Using the Global.asax
File</font></h3>")
      Response.Write("Request is Starting...<br>")
   End Sub

   Sub Application_EndRequest(Sender As Object, E As EventArgs)
      Response.Write("Request is Ending...<br>")
   End Sub

   Sub Application_Error(Sender As Object, E As EventArgs)
      Context.ClearError()
      Response.Redirect("errorpage.htm")
   End Sub
</script>
```

Managing Application State

Using Application State

This sample illustrates the use of application state to read a dataset in **Application_Start**.

Because an application and all the objects it stores can be concurrently accessed by different threads, it is better to store only infrequently modified data with application scope. Ideally an object is initialized in the **Application_Start** event and further access is read-only.

In the following sample a file is read in **Application_Start** (defined in the Global.asax file) and the content is stored in a **DataView** object in the application state.

Code for Application_ StartEvent.aspx

```
      Sub Application_Start()
  Dim ds As New DataSet()
  Dim fs As New
FileStream(Server.MapPath("schemadata.xml"),FileMode.Open,FileAccess.Read)
  Dim reader As New StreamReader(fs)
  ds.ReadXml(reader)
  fs.Close()

  Dim view As New DataView (ds.Tables(0))
  Application("Source") = view
End Sub
```

In the **Page_Load** method, the **DataView** is then retrieved and used to populate a **DataGrid** object:

```

Sub Page_Load(sender As Object, e As EventArgs)
  Dim Source As New DataView  = CType(Application("Source"), DataView)
  ...
  MyDataGrid.DataSource = Source
  ...
End Sub

```

The advantage of this solution is that only the first request pays the price of retrieving the data. All subsequent requests use the already existing **DataView** object. As the data is never modified after initialization, you do not have to make any provisions for serializing access.

Using Session State

The following sample illustrates the use of session state to store volatile user preferences.

```
<script language="VB" runat="server">

   Sub Session_Start(Sender As Object, E As EventArgs)

     Session("BackColor") = "beige"
     Session("ForeColor") = "black"
     Session("LinkColor") = "blue"
     Session("FontSize") = "8pt"
     Session("FontName") = "verdana"
   End Sub
</script>
```

Coalesce

To provide individual data for a user during a session, data can be stored with session scope. In the following sample, values for user preferences are initialized in the **Session_Start** event in the Global.asax file.

```
Sub Session_Start()
   Session("BackColor") = "beige"
   ...
End Sub
```

In the following customization page, values for user preferences are modified in the **Submit_Click** event handler according to user input.

```
Protected Sub Submit_Click(sender As Object, e As EventArgs)
   Session("BackColor") = BackColor.Value
   ...

   Response.Redirect(State("Referer").ToString())
End Sub
```

The individual values are retrieved using the **GetStyle** method:

```
Protected GetStyle(key As String) As String
   Return(Session(key).ToString())
End Sub
```

The **GetStyle method** is used to construct session-specific styles:

```
<style>
   body
   {
     font: <%=GetStyle("FontSize")%> <%=GetStyle("FontName")%>;
     background-color: <%=GetStyle("BackColor")%>;
   }
   a
   {
      color: <%=GetStyle("LinkColor")%>
   }

</style>
```

To verify that the values are really stored with session scope, open the sample page twice, then change one value in the first browser window and refresh the second one. The second window picks up the changes because both browser instances share a common **Session** object.

Configuring session state: Session state features can be configured via the **<sessionState>** section in a web.config file. To double the default timeout of 20 minutes, you can add the following to the web.config file of an application:

If cookies are not available, a session can be tracked by adding a session identifier to the URL. This can be enabled by setting the following:

By default, ASP.NET will store the session state in the same process that processes the request, just as ASP does. Additionally, ASP.NET can store session data in an external process, which can even reside on another machine. To enable this feature:

- Start the ASP.NET state service, either using the Services snap-in or by executing "net start aspnet_state" on the command line. The state service will by default listen on port 42424. To change the port, modify the registry key for the service: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\aspnet_state\Parameters\Port
- Set the **mode** attribute of the **<sessionState>** section to "StateServer".
- Configure the **stateConnectionString** attribute with the values of the machine on which you started aspnet_state.

The following sample assumes that the state service is running on the same machine as the Web server ("localhost") and uses the default port (42424):

```
<sessionState
  mode="StateServer"
  stateConnectionString="tcpip=localhost:42424"
/>
```

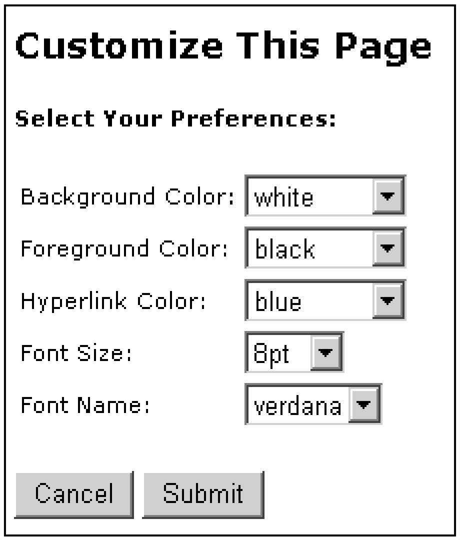
Note that if you try the sample above with this setting, you can reset the Web server (enter iisreset on the command line) and the session state value will persist.

Using Client-Side Cookies

The following sample illustrates the use of client-side cookies to store volatile user preferences.

Coalesce

Figure 8.2 for Cookies1.aspx


Customize This Page

Select Your Preferences:

Background Color: white

Foreground Color: black

Hyperlink Color: blue

Font Size: 8pt

Font Name: verdana

Storing cookies on the client is one of the methods that ASP.NET's session state uses to associate requests with sessions. Cookies can also be used directly to persist data between requests, but the data is then stored on the client and sent to the server with every request. Browsers place limits on the size of a cookie; therefore, only a maximum of 4096 bytes is guaranteed to be acceptable.

When the data is stored on the client, the **Page_Load** method in the file cookies1.aspx checks whether the client has sent a cookie. If not, a new cookie is created and initialized and stored on the client:

Code for Cookies1.aspx

```
Protected Sub Page_Load(sender As Object, e As EventArgs)
   If Request.Cookies("preferences1") = Null Then
      Dim cookie As New HttpCookie("preferences1")
      cookie.Values.Add("ForeColor", "black")
      ...
      Response.Cookies.Add(cookie)
   End If
     End Sub
```

On the same page, a **GetStyle** method is used again to provide the individual values stored in the cookie:

Code for Customization.aspx

```
Protected Function GetStyle(key As String) As String
 Dim cookie As HttpCookie = Request.Cookies("preferences1")
 If cookie <> Null Then
   Select Case key
     Case "ForeColor"
       Return(cookie.Values("ForeColor"))
     Case ...
   End Select
 End If
 Return("")
End Function
```

Verify that the sample works by opening the cookies1.aspx page and modifying the preferences. Open the page in another window, it should pick up the new preferences. Close all browser windows and open the cookies1.aspx page again. This should delete the temporary cookie and restore the default preference values.

To make a cookie persistent between sessions, the **Expires** property on the **HttpCookie** class has to be set to a date in the future. The following code on the customization.aspx page is identical to the previous sample, with the exception of the assignment to **Cookie.Expires**:

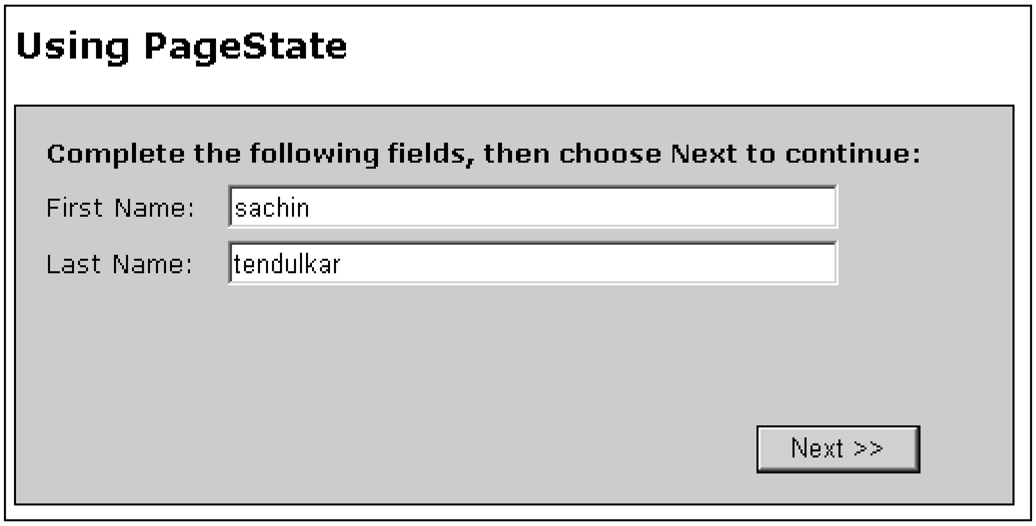
```
Protected Sub Submit_Click(sender As Object, e As EventArgs)
   Dim cookie As New HttpCookie("preferences2")
   cookie.Values.Add("ForeColor",ForeColor.Value)
   ...
   cookie.Expires = DateTime.MaxValue ' Never Expires

   Response.Cookies.Add(cookie)

   Response.Redirect(State("Referer").ToString())
End Sub
```

Verify that the sample is working by modifying a value, closing all browser windows, and opening cookies2.aspx again. The window should still show the customized value.

Using ViewState

This sample illustrates the use of the **ViewState** property to store request-specific values.

Figure 8.3 PageState1.aspx


Using PageState

Complete the following fields, then choose Next to continue:

First Name: sachin

Last Name: tendulkar

```
<%@ Register TagPrefix="Acme" TagName="Address"
Src="address.ascx" %>

<html>

 <script language="VB" runat="server">

    Sub Page_Load(Src As Object, E As EventArgs)

      If Not (IsPostBack)
        ViewState("PanelIndex") = 0
      End If
    End Sub

    Sub Next_Click(Src As Object, E As EventArgs)

      Dim PrevPanelId As String = "Panel" & ViewState("PanelIndex").ToString()
      ViewState("PanelIndex") = CInt(ViewState("PanelIndex")) + 1
      Dim PanelId As String = "Panel" & ViewState("PanelIndex").ToString()

      Dim P As System.Web.UI.WebControls.Panel
      P = FindControl(PanelId)
      P.Visible=true

      P = FindControl(PrevPanelId)
      P.Visible=false
    End Sub
    Sub Prev_Click(Src As Object, E As EventArgs)

      Dim PanelId As String = "Panel" & ViewState("PanelIndex").ToString()
      ViewState("PanelIndex") = CInt(ViewState("PanelIndex")) - 1
      Dim PrevPanelId As String = "Panel" & ViewState("PanelIndex").ToString()

      Dim P As System.Web.UI.WebControls.Panel
      P = FindControl(PanelId)
      P.Visible=false

      P = FindControl(PrevPanelId)
      P.Visible=true
    End Sub

    Sub Finish_Click(Src As Object, E As EventArgs)

      Dim P As System.Web.UI.WebControls.Panel
      Dim PanelId As String = "Panel" & ViewState("PanelIndex").ToString()
      P = FindControl(PanelId)
      P.Visible=false

      MyLabel.Text &= "<b>Thank You!  We received the following information:
</b><p>"
      MyLabel.Text &= "First Name: " & FirstName.Value & "<br>"
      MyLabel.Text &= "Last Name: " & LastName.Value & "<br>"
      MyLabel.Text &= "Address: " & Address.Address & "<br>"
      MyLabel.Text &= "City: " & Address.City & "<br>"
      MyLabel.Text &= "State: " & Address.StateName & "<br>"
      MyLabel.Text &= "Zip: " & Address.Zip & "<br>"
      MyLabel.Text &= "Card Number: " & CardNum.Value & "<br>"
      MyLabel.Text &= "Card Type: " & CardType.SelectedItem.Value & "<br>"
      MyLabel.Text &= "Expires: " & Expires.Value & "<br>"
    End Sub

  </script>

  <body style="font: 10pt verdana">

   <h3><font face="Verdana">Using PageState</font></h3>

   <form runat="server">

   <ASP:Panel id="Panel0" Visible="true" runat="server">
    <table width="500" height="200" style="font:10pt verdana;background-
color:cccccc;border-width:1;border-style:solid;border-color:black">
      <tr>
       <td style="padding:10,10,10,10" valign="top">
```

```
        <table height="100%" style="font:10pt verdana;">
          <tr>
            <td colspan="2"><b>Complete the following fields, then choose Next
to continue:</b></td>
          </tr>
          <tr height="20"/>
          <tr>
            <td>First Name:</td>
            <td><input          id="FirstName"          type="text"          size="45"
runat="server"></td>
          </tr>
          <tr>
            <td>Last Name:</td>
            <td><input          id="LastName"          type="text"          size="45"
runat="server"></td>
          </tr>
          <tr>
            <td colspan="2" align="right" height="100%" valign="bottom">
              <input          type="submit"          Value="          Next          >>          "
OnServerClick="Next_Click" runat="server">
            </td>
          </tr>
        </table>
      </td>
    </tr>
  </table>
</ASP:Panel>

<ASP:Panel id="Panel1" Visible="false" runat="server">
  <table   width="500"   height="200"   style="font:10pt   verdana;background-
color:cccccc;border-width:1;border-style:solid;border-color:black">
    <tr>
      <td style="padding:10,10,10,10" valign="top">
        <table height="100%" style="font:10pt verdana;">
          <tr>
            <td colspan="2"><b>Complete the following fields, then choose Next
to continue:</b></td>
          </tr>
          <tr height="20"/>
          <tr>
            <td colspan="2">
              <Acme:Address id="Address" ShowCaption="false" runat="server"/>
            </td>
          </tr>
          <tr>
            <td colspan="2" align="right" valign="bottom" height="100%">
              <input          type="submit"          Value="          <<          Back          "
```

```
OnServerClick="Prev_Click" runat="server">
          <input type="submit" Value=" Next >> "
OnServerClick="Next_Click" runat="server">
         </td>
        </tr>
       </table>
      </td>
     </tr>
    </table>
  </ASP:Panel>

  <ASP:Panel id="Panel2" Visible="false" runat="server">
   <table width="500" height="200" style="font:10pt verdana;background-
color:cccccc;border-width:1;border-style:solid;border-color:black">
     <tr>
      <td style="padding:10,10,10,10" valign="top">
       <table height="100%" style="font:10pt verdana;">
        <tr>
         <td colspan="2"><b>Complete the following fields, then choose Next
to continue:</b></td>
        </tr>
        <tr height="20"/>
        <tr>
         <td>Card Number: </td>
         <td><input id="CardNum" size="45" type="text"
runat="server"/></td>
        </tr>
        <tr>
         <td>Card Type: </td>
         <td>
          <asp:DropDownList id="CardType" runat="server">
           <asp:ListItem>Visa</asp:ListItem>
           <asp:ListItem>Mastercard</asp:ListItem>
           <asp:ListItem>Discover</asp:ListItem>
          </asp:DropDownList>
         </td>
        </tr>
        <tr>
         <td>Expires: </td>
         <td><input id="Expires" type="text" runat="server"/></td>
        </tr>
        <tr>
         <td colspan="2" align="right" valign="bottom" height="100%">
          <input type="submit" Value=" << Back "
OnServerClick="Prev_Click" runat="server">
          <input type="submit" Value=" Finish "
OnServerClick="Finish_Click" runat="server">
```

```
            </td>
          </tr>
        </table>
      </td>
    </tr>
  </table>
  </ASP:Panel>
  </form>

  <asp:Label id="MyLabel" EnableViewState="false" runat="server"/>
</body></html>
```

ASP.NET provides the server-side notion of a view state for each control. A control can save its internal state between requests using the **ViewState** property on an instance of the class **StateBag**. The **StateBag** class provides a dictionary-like interface to store objects associated with a string key.

The file pagestate1.aspx displays one visible panel and stores the index of it in the view state of the page with the key **PanelIndex**:

```
Protected Sub Next_Click(sender As Object, e As EventArgs)
   Dim PrevPanelId As String = "Panel" + ViewState("PanelIndex").ToString()
   ViewState("PanelIndex") = CType(ViewState("PanelIndex") + 1, Integer)
   Dim PanelId As String = "Panel" + ViewState("PanelIndex").ToString()
   ...
End Sub
```

Note that if you open the page in several browser windows, each browser window will initially show the name panel. Each window can independently navigate between the panels.

Section Summary

1. Use application state variables to store data that is modified infrequently but used often.
2. Use session state variables to store data that is specific to one session or user. The data is stored entirely on the server. Use it for short-lived, bulky, or sensitive data.
3. Store small amounts of volatile data in a nonpersistent cookie. The data is stored on the client, sent to the server on each request, and expires when the client ends execution.

4. Store small amounts of non-volatile data in a persistent cookie. The data is stored on the client until it expires and is sent to the server on each request.
5. Store small amounts of request-specific data in the view state. The data is sent from the server to the client and back.

HTTP Handlers and Factories

Overview

ASP.NET provides a low-level request/response API that enables developers to use .NET Framework classes to service incoming HTTP requests. Developers accomplish this by authoring classes that support the **System.Web.IHTTPHandler** interface and implement the **ProcessRequest()** method. Handlers are often useful when the services provided by the high-level page framework abstraction are not required for processing the HTTP request. Common uses of handlers include filters and CGI-like applications, especially those that return binary data.

Each incoming HTTP request received by ASP.NET is ultimately processed by a specific instance of a class that implements **IHTTPHandler**.

IHttpHandlerFactory provides the infrastructure that handles the actual resolution of URL requests to **IHttpHandler** instances. In addition to the default **IHttpHandlerFactory** classes provided by ASP.NET, developers can optionally create and register factories to support

rich request resolution and activation scenarios.

Configuring HTTP Handlers and Factories

HTTP handlers and factories are declared in the ASP.NET configuration as part of a web.config file. ASP.NET defines an **<httphandlers>** configuration section where handlers and factories can be added and removed. Settings for **HttpHandlerFactory** and **HttpHandler** are inherited by subdirectories.

For example, ASP.NET maps all requests for .aspx files to the **PageHandlerFactory** class in the global machine.config file:

```
<httphandlers>
  ...
  <add verb="*" path="*.aspx"
type="System.Web.UI.PageHandlerFactory,System.Web" />
  ...
</httphandlers>
```

Creating a Custom HTTP Handler

The following sample creates a custom **HttpHandler** that handles all requests to "SimpleHandler.aspx".

A custom HTTP handler can be created by implementing the **IHttpHandler** interface, which contains only two methods. By calling **IsReusable**, an HTTP factory can query a handler to determine whether the same instance can be used to service multiple requests. The **ProcessRequest** method takes an **HttpContext** instance as a parameter, which gives it access to the **Request** and **Response** intrinsics. In the following sample, request data is ignored and a constant string is sent as a response to the client.

```
Public Class SimpleHandler : Inherits IHttpHandler
 Public Sub ProcessRequest(context As HttpContext)
   context.Response.Write("Hello World!")
 End Sub

 Public Function IsReusable() As Boolean
   Return(True)
 End Function
End Class
```

After placing the compiled handler assembly in the application's \bin directory, the handler class can be specified as a target for requests. In this case, all requests for "SimpleHandler.aspx" will be routed to an instance of the **SimpleHandler** class, which lives in the namespace **Acme.SimpleHandler**.

Section Summary

1. HTTP Handlers and factories are the backbone of the ASP.NET page framework.
2. Factories assign each request to one handler, which processes the request.
3. Factories and handlers are defined in the web.config file. Settings for factories are inherited by subdirectories.
4. To create a custom handler, implement **IHttpHandler** and add the class in the **<httphandlers>** section of the web.config in the directory.

```
'---------------------------------------------------------------------
'  This file is part of the Microsoft .NET SDK Code Samples.
'
'  Copyright (C) Microsoft Corporation.  All rights reserved.
'
'This source code is intended only as a supplement to Microsoft
```

```
'Development Tools and/or on-line documentation.  See these other
'materials for detailed information regarding Microsoft code samples.
'
'THIS CODE AND INFORMATION ARE PROVIDED AS IS WITHOUT WARRANTY OF ANY
'KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
'IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
'PARTICULAR PURPOSE.
'---------------------------------------------------------------------
Imports System.Web

Namespace Acme

   Public Class SimpleHandlerVB : Implements IHttpHandler

      Public Sub ProcessRequest(Context As HttpContext) Implements IHttpHandler.ProcessRequest
         Context.Response.Write("Hello World!")
      End Sub

      Public ReadOnly Property IsReusable As Boolean Implements IHttpHandler.IsReusable
         Get
            Return true
         End Get
      End Property
   End Class
End Namespace
```

Chapter 9

ASP.NET APPLICATION

Application Overview

What is an ASP.NET Application?

ASP.NET defines an application as the sum of all files, pages, handlers, modules, and executable code that can be invoked or run in the scope of a given virtual directory (and its subdirectories) on a Web application server. For example, an "order" application might be published in the "/order" virtual directory on a Web server computer. For IIS the virtual directory can be set up in the Internet Services Manager; it contains all subdirectories, unless the subdirectories are virtual directories themselves.

Each ASP.NET Framework application on a Web server is executed within a unique .NET Framework application domain, which guarantees class isolation (no versioning or naming conflicts), security sandboxing (preventing access to certain machine or network resources), and static variable isolation.

ASP.NET maintains a pool of **HttpApplication** instances over the course of a Web application's lifetime. ASP.NET automatically assigns one of these instances to process each incoming HTTP request that is received by the application. The particular **HttpApplication** instance assigned is responsible for managing the entire lifetime of the request and is reused only after the request has been completed. This means that user code within the **HttpApplication** does not need to be reentrant.

Creating an Application

To create an ASP.NET Framework application you can use an existing virtual directory or create a new one. For example, if you installed Windows 2000 Server including IIS, you probably have a directory C:\InetPub\WWWRoot. You can configure IIS using the Internet Services Manager, available under Start -> Programs -> Administrative Tools. Right-click on an existing directory and choose either New (to create a new virtual directory) or Properties (to promote an existing regular directory).

By placing a simple .aspx page like the following in the virtual directory and accessing it with the browser, you trigger the creation of the ASP.NET application.

```
    <%@Page Language="VB"%>
<html>
<body>
<h1>hello world, <% Response.Write(DateTime.Now.ToString()) %></h1>
```

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

Now you can add appropriate code to use the Application object--to store objects with application scope, for example. By creating a global.asax file you also can define various event handlers-- for the **Application_Start** event, for example.

Lifetime of an Application

An ASP.NET Framework application is created the first time a request is made to the server; before that, no ASP.NET code executes. When the first request is made, a pool of **HttpApplication** instances is created and the **Application_Start** event is raised. The **HttpApplication** instances process this and subsequent requests, until the last instance exits and the **Application_End** event is raised.

Note that the **Init** and **Dispose** methods of **HttpApplication** are called per instance and thus can be called several times between **Application_Start** and **Application_End**. Only these events are shared among all instances of **HttpApplication** in one ASP.NET application.

A Note on Multiple Threads

If you use objects with application scope, you should be aware that ASP.NET processes requests concurrently and that the **Application** object can be accessed by multiple threads. Therefore the following code is dangerous and might not produce the expected result, if the page is repeatedly requested by different clients at the same time.

```
<%
Application("counter") = CType(Application("counter"), Int32) + 1
%>
```

To make this code thread safe, serialize the access to the **Application** object using the **Lock** and **UnLock** methods. However, doing so also means accepting a considerable performance hit:

```
<%
Application.Lock()
Application("counter") = CType(Application("counter"), Int32) + 1
Application.UnLock()
%>
```

Coalesce

Another solution is to make the object stored with an application scope thread safe. For example, note that the collection classes in the **System.Collections** namespace are not thread safe for performance reasons.

Section Summary

1. ASP.NET Framework applications consist of everything under one virtual directory of the Web server.
2. You create an ASP.NET Framework application by adding files to a virtual directory on the Web server.
3. The lifetime of an ASP.NET Framework application is marked by **Application_Start** and **Application_End** events.
4. Access to application-scope objects must be safe for multithreaded access.