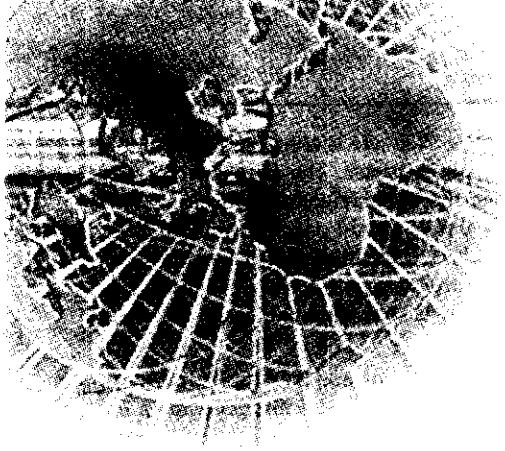


# 18

# Navigation Controls: TreeView, Menu, and SiteMapPath

Navigation controls allow users to navigate through the Web pages. If your Web site is composed of multiple pages, you need an interface that helps in making the navigation of pages easier. The user interface for navigation can be a simple, static, hyperlink or involve the use of trees and menus. Navigation controls, such as the `TreeView` control, `Menu` control, and `SiteMapPath` control, make it simple to add page hierarchies, quick links, and provide advanced navigational capabilities to your application. The Navigation controls display data in a consistent manner enabling the users to comprehend the application structure. These Navigation controls are the most recently introduced controls, and provide several features, such as consistent and easily managed navigation, to developers as well as users.

In this chapter we explore three navigation controls - TreeView control, Menu control, and SiteMapPath control, and also discusses about the base class of each of these navigation controls. This chapter also discusses about the various properties, methods, and events of the navigation controls and shows you the practical implementation of using these navigation controls.

## Using the **TreeView** Class

The `TreeView` class enables you to manage the TreeView control. The `TreeView` class can be used to create a tree-based hierarchical structure at runtime.

Here is the class hierarchy of the `TreeView` class:

```
System.Object
   System.Web.UI.Control
      System.Web.UI.WebControls.WebControl
         System.Web.UI.WebControls.BaseDataBoundControl
            System.Web.UI.WebControls.HierarchicalDataBoundControl
               System.Web.UI.WebControls.TreeView
```

The `TreeView` class contains a `Node` property that can be used to create the nodes of the `TreeView` control. The `Nodes` collection contains the `Add` method that can be used to add a text item or a `TreeNode` object. It can further have child nodes and consists of properties, such as `ExpandAll` and `CollapseAll`, which expand and collapse the tree view nodes. You can also insert check boxes against the nodes and can obtain the selected information dynamically by using the `TreeView` class.

You can find the noteworthy properties of the `TreeView` class in Table 18.1:

**Table 18.1: Noteworthy Properties of the TreeView Class**

| Property | Description |
|---|---|
| `AutoGenerateDataBindings` | Obtains or sets a value indicating whether the `TreeView` control automatically generates tree node bindings |
| `CheckedNodes` | Obtains a collection of `TreeNode` objects that represent the nodes in the `TreeView` control that display a selected check box |
| `CollapseImageToolTip` | Obtains or sets the `ToolTip` for the image that is displayed for the collapsible node indicator |
| `CollapseImageUrl` | Obtains or sets the URL to a custom image for the collapsible node indicator |
| `DataBindings` | Obtains a collection of `TreeNodeBinding` objects that define the relationship between a data item and the node to which the object is binding |
| `EnableClientScript` | Obtains or sets a value indicating whether the `TreeView` control renders client-side script to handle expanding and collapsing events |
| `ExpandDepth` | Obtains or sets the number of levels that are expanded when a `TreeView` control is displayed for the first time |
| `ExpandImageToolTip` | Obtains or sets the `ToolTip` for the image that is displayed for the expandable node indicator |

**Table 18.1: Noteworthy Properties of the TreeView Class**

| | |
|---|---|
| ExpandImageUrl | Obtains or sets the URL to a custom image for the expandable node indicator |
| HoverNodeStyle | Obtains a reference to the TreeNodeStyle object that allows you to set the appearance of a node when the mouse pointer is positioned over it |
| ImageSet | Obtains or sets the group of images to use for the TreeView control |
| LeafNodeStyle | Obtains a reference to the TreeNodeStyle object that allows you to set the appearance of leaf nodes |
| LevelStyles | Obtains a collection of Style objects that represent the node styles at the individual levels of the tree |
| LineImagesFolder | Obtains or sets the path to a folder that contains the line images that are used to connect child nodes to parent nodes |
| MaxDataBindDepth | Obtains or sets the maximum number of tree levels to bind to the TreeView control |
| NodeIndent | Obtains or sets the indentation amount (in pixels) for the child nodes of the TreeView control |
| Nodes | Obtains a collection of TreeNode objects that represents the root nodes in the TreeView control |
| NodeStyle | Obtains a reference to the TreeNodeStyle object that allows you to set the default appearance of the nodes in the TreeView control |
| NodeWrap | Obtains or sets a value indicating text wraps in a node when the node runs out of space |
| NoExpandImageUrl | Obtains or sets the URL to a custom image for the non-expandable node indicator |
| ParentNodeStyle | Obtains a reference to the TreeNodeStyle object that allows you to set the appearance of parent nodes in the TreeView control |
| PathSeparator | Obtains or sets the character that is used to delimit the node values that are specified by the ValuePath property |
| PopulateNodesFromClient | Obtains or sets a value indicating whether node data is populated on demand from the client |
| RootNodeStyle | Obtains a reference to the TreeNodeStyle object that allows you to set the appearance of the root node in the TreeView control |
| SelectedNode | Obtains a TreeNode object that represents the selected node in the TreeView control |
| SelectedNodeStyle | Obtains the TreeNodeStyle object that controls the appearance of the selected node in the TreeView control |
| SelectedValue | Obtains the value of the selected node |
| ShowCheckBoxes | Obtains or sets a value indicating which node types will display a check box in the TreeView control |
| ShowExpandCollapse | Obtains or sets a value indicating whether expansion node indicators are displayed on the Web page |
| ShowLines | Obtains or sets a value indicating whether lines connecting child nodes to parent nodes are displayed on the Web page |
| SkipLinkText | Obtains or sets a value that is used to render alternate text for screen readers to skip the content for the control |

| Table 18.1: Noteworthy Properties of the TreeView Class | |
|---|---|
| Target | Obtains or sets the target window or frame in which to display the Web page content that is associated with a node |
| Visible | Obtains or sets a value indicating whether the control is rendered as UI on the page |

You can find the noteworthy methods of `TreeView` class in Table 18.2:

| Table 18.2: Noteworthy Methods of TreeView Class | |
|---|---|
| CollapseAll | Closes every node in the tree |
| ExpandAll | Opens every node in the tree |
| FindNode | Find the TreeNode object in the TreeView control at the specified path |

You can find the noteworthy events of `TreeView` class in Table 18.3:

| Table 18.3: Noteworthy Events of the TreeView Class | |
|---|---|
| SelectedNodeChanged | Occurs when a node is selected in the TreeView control |
| TreeNodeCheckChanged | Occurs when a check box in the TreeView control changes state between posts to the server |
| TreeNodeCollapsed | Occurs when a node is collapsed in the TreeView control |
| TreeNodeDataBound | Occurs when a data item is bound to a node in the TreeView control |
| TreeNodeExpanded | Occurs when a node is expanded in the TreeView control |
| TreeNodePopulate | Occurs when a node with its PopulateOnDemand property set to true is expanded in the TreeView control |

# The **TreeView** Control

The `TreeView` control is used for logically displaying the data in a hierarchical structure, similar to Windows explorer. You can use this control when you need to display the navigation menu for displaying the files and folders. You can display an XML document and database records in a tree structure. To work with the `TreeView` control at runtime, you can programmatically access the `TreeView` Web server control.

The `TreeView` Web server control is used to create tree-based hierarchical structure, set properties, populate nodes, and so on. The `TreeView` control consists of the `TreeView` container and `TreeNode` properties. There can be multiple nodes in the tree view structure, which can be nested. The `TreeNode` property has a `Text` property that displays the text for the node.

In addition to the `Text` property, there is also an `Expanded` property for TreeNode, which is used to expand and collapse the nodes. The default value of this property is `False`. However, if you set it to `True`, the control will be displayed in the expanded view every time it is loaded.

You can access the `TreeView` control directly from the Navigation tab in the Toolbox. You can add the `TreeView` control to your Web page simply by either double clicking the `TreeView` control in the Toolbox or by dragging the control from the Toolbox to the design view of the Defalt.aspx page, which is the default page of your Web site. Figure 18.1 shows the default tree view node when the `TreeView` control is placed on the `Default.aspx` Web page of the TreeViewControl example:

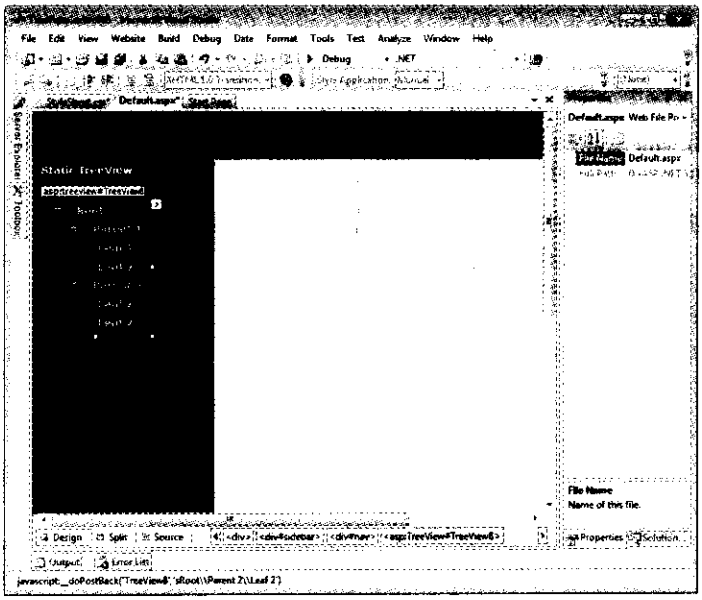


**Figure 18.1: Displaying the TreeView Control**

You can also add the `TreeView` control by adding the following code snippet in the `<form>` element of your Web page:

```
<asp:TreeView ID="TreeView1" runat="server">
</asp:TreeView>
```

**TIP**

*The TreeView control supports the AutoFormat feature, which allows you to select from the already existing formats. AutoFormat option can be selected from the Smart Tag of the TreeView control.*

The `TreeView` control is a powerful server control that is used to render the tree view user interface. It has many properties, such as `CheckedNodes`, `DataBindings`, `NodeIndent`, `Nodes`, and `NodeStyle` that are served by means of the `TreeView` class. The contents for the `TreeView` control can be added in the following ways:

- By adding programmatically at runtime.
- By dynamically loading from an external datasource, such as XML file.
- The `TreeView` control can also be added while designing an application.

## Adding Nodes to a TreeView Control Dynamically

The `TreeView` control can load the data dynamically at runtime. For this, you have to set the properties of the `TreeView` nodes dynamically. The `TreeView` control has a `Nodes` property, known as `TreeView.Nodes`. You can add nodes to the `TreeView` control by using the `add` method of the `TreeView.Nodes` property.

## Adding Nodes to a TreeView Control Using the DataSource Property

You can also add data to the `TreeView` control by using the `DataSource` property. In fact, this is a more flexible and easy-to-use property for binding data. By setting the `DataSourceID` property of the `TreeView` control to the desired file, you can load the content of the `TreeView` control from the specified file. The target file can be an XML file, a sitemap, or you can even populate a treeview from the database, such as SQL Server or MS-Access.

After an appropriate datasource is set for the `TreeView` control, the `TreeView` control displays the information in a tree structure. Using the datasource property is a flexible approach for the `TreeView` control as you can easily change the text of any of the nodes by changing the content of the underlying file.

You can only bind the `Treeview` control to the `XmlDataSource` and `SiteMapDataSource` controls using the wizards. However, if you want to bind the `TreeView` control to a database, you need to generate the `TreeView` control programmatically.

## Adding Nodes to a TreeView Control at Design Time

Adding node to a `TreeView` control at design time is easy to use but is not as flexible as hard-coding of properties and cannot be changed at runtime. You can also set the URLs and descriptions of the nodes by using the Tree View Node Editor option.

To add nodes to the `TreeView` control using the Tree View Node Editor, click the Show Smart Tag at the top-right corner to display the Smart Tag and select the Edit Nodes option, as shown in Figure 18.2:

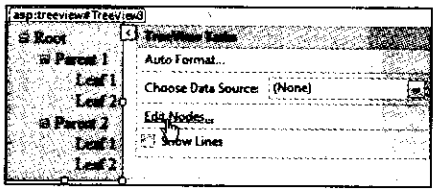



**Figure 18.2: Selecting the Edit Nodes Option**

The TreeView Node Editor opens, and allows you to add and delete nodes and set their properties, as shown in Figure 18.3:

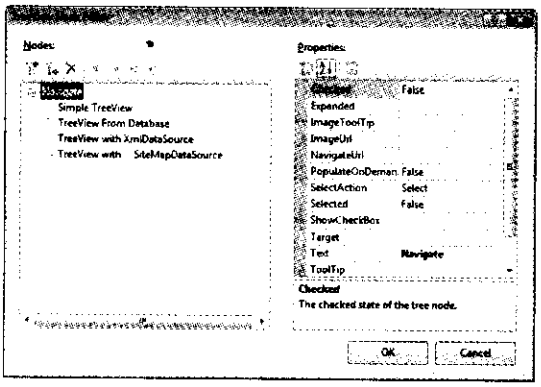



**Figure 18.3: TreeView Node Editor**

# Creating the TreeView Control

The `TreeView` control can be easily created in Visual Studio 2008. For this, simply drag and drop the control onto the Web page. The `TreeView` control is available under the Navigation tab in the Toolbox of VS 2008. We have used the `TreeView` control in the `TreeViewControlVB` application. You can find the code of `TreeViewControlVB` application in the Code\ASP.NET\Chapter 18\`TreeViewControlVB` folder on the CD. You can find the code for the `Default.aspx` page in Listing 18.1:

**Listing 18.1:** Showing the Code for the `Default.aspx` Page

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb"
  Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
```

```
    <title>TreeView Example</title>
    <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="mainForm" runat="server">
        <div>
        <div id="header">
            </div>
        <div id="sidebar">
        <div id="nav">
            <asp:Label ID="Label1" runat="server" Text="Static Treeview"
            Font-Bold="True" ForeColor="White"></asp:Label>
             <br />
            <br />
            <asp:TreeView ID="TreeView1" runat="server"
            ImageSet="Arrows">
            <ParentNodeStyle Font-Bold="False" />
            <HoverNodeStyle Font-Underline="True"
             ForeColor="#5555DD" />
            <SelectedNodeStyle Font-Underline="True"
             ForeColor="#5555DD" HorizontalPadding="0px"
            VerticalPadding="0px" />
            <Nodes>
            <asp:TreeNode Text="Navigate" Value="Navigate">
            <asp:TreeNode Text="Simple Treeview"
             Value="Simple Treeview"
             NavigateUrl="~/Default.aspx"></asp:TreeNode>
            <asp:TreeNode Text="TreeView From Database"
             Value="TreeView From Database"
             NavigateUrl="~/TreeViewDatabase.aspx">
            </asp:TreeNode>
            <asp:TreeNode Text="TreeView with XmlDataSource"
             Value="TreeView with XmlDataSource"
             NavigateUrl="~/TreeViewXMLData.aspx">
            </asp:TreeNode>
            <asp:TreeNode Text="TreeView with
             SiteMapDataSource" Value="TreeView with
             SiteMapDataSource"
             NavigateUrl="~/TreeViewSiteMapDataSource.aspx">
            </asp:TreeNode>
            </asp:TreeNode>
            </Nodes>
            <NodeStyle Font-Names="Tahoma" Font-Size="10pt"
             ForeColor="Black" HorizontalPadding="5px"
             NodeSpacing="0px" VerticalPadding="0px" />
            </asp:TreeView>
        </div>
        </div>
        <div id="content">
        <div class="itemContent">
            <br />
            <asp:Table ID="Table1" runat="server" Height="81px"
            Width="594px">
            <asp:TableRow ID="TableRow1" runat="server">
            <asp:TableCell ID="TableCell1" runat="server">
            <asp:TreeView ID="TreeView2" runat="server"
             ImageSet="Arrows">
            <Nodes>
            <asp:TreeNode Text="Root Node 1"
            Value="Root Node 1">
            <asp:TreeNode Text="Leaf Node"
             Value="Leaf Node"></asp:TreeNode>
```

```
        <asp:TreeNode Text="Leaf Node"
          Value="Leaf Node"></asp:TreeNode>
        </asp:TreeNode>
        <asp:TreeNode Text="Root Node 2"
         Value="Root Node 2"></asp:TreeNode>
        </Nodes>
        <ParentNodeStyle Font-Bold="False" />
        <HoverNodeStyle Font-Underline="True"
         ForeColor="#5555DD" />
        <SelectedNodeStyle Font-Underline="True"
         ForeColor="#5555DD" HorizontalPadding="0px"
         VerticalPadding="0px" />
        <NodeStyle Font-Names="Verdana" Font-
         Size="8pt" ForeColor="Black"
         HorizontalPadding="5px"
         NodeSpacing="0px" VerticalPadding="0px" />
        </asp:TreeView>
        </asp:TableCell>
        <asp:TableCell ID="TableCell2" runat="server">
        <asp:TreeView ID="TreeView3" runat="server"
         ImageSet="News" NodeIndent="10">
        <ParentNodeStyle Font-Bold="False" />
        <HoverNodeStyle Font-Underline="True" />
        <SelectedNodeStyle Font-Underline="True"
         HorizontalPadding="0px"
         VerticalPadding="0px" />
        <Nodes>
        <asp:TreeNode Text="Root Node 1"
         Value="Root Node 1">
        <asp:TreeNode Text="Leaf Node"
         Value="Leaf Node"></asp:TreeNode>
        <asp:TreeNode Text="Leaf Node"
         Value="Leaf Node"></asp:TreeNode>
        </asp:TreeNode>
        <asp:TreeNode Text="Root Node 2"
         Value="Root Node 2"></asp:TreeNode>
        </Nodes>
        <NodeStyle Font-Names="Arial" Font-
         Size="10pt" ForeColor="Black"
         HorizontalPadding="5px"
         NodeSpacing="0px" VerticalPadding="0px"/>
        </asp:TreeView>
        </asp:TableCell>
        <asp:TableCell ID="TableCell3" runat="server">
        <asp:TreeView ID="TreeView4" runat="server"
         ShowCheckBoxes="All" ShowLines="True">
        <Nodes>
        <asp:TreeNode Text="Root Node 1"
         Value="Root Node 1">
        <asp:TreeNode Text="Leaf Node"
         Value="Leaf Node"></asp:TreeNode>
        <asp:TreeNode Text="Leaf Node"
         Value="Leaf Node"></asp:TreeNode>
        </asp:TreeNode>
        <asp:TreeNode Text="Root Node 2"
         Value="Root Node 2"></asp:TreeNode>
        </Nodes>
        </asp:TreeView>
        </asp:TableCell>
        </asp:TableRow>
        <asp:TableRow ID="TableRow2" runat="server">
        <asp:TableCell ID="TableCell4" runat="server">
```

```
<asp:TreeView ID="TreeView6" runat="server"
 ImageSet="Contacts" NodeIndent="10">
<ParentNodeStyle Font-Bold="True"
 ForeColor="#5555DD" />
<HoverNodeStyle Font-Underline="False" />
<SelectedNodeStyle Font-Underline="True"
 HorizontalPadding="0px"
 VerticalPadding="0px" />
<Nodes>
<asp:TreeNode Text="Root Node 1"
 Value="Root Node 1" Expanded="False">
<asp:TreeNode Text="Leaf Node"
 Value="Leaf Node"></asp:TreeNode>
<asp:TreeNode Text="Leaf Node"
 Value="Leaf Node"></asp:TreeNode>
</asp:TreeNode>
<asp:TreeNode Text="Root Node 2"
 Value="Root Node 2"></asp:TreeNode>
</Nodes>
<NodeStyle Font-Names="Verdana" Font-
 Size="8pt" ForeColor="Black"
 HorizontalPadding="5px"
 NodeSpacing="0px" VerticalPadding="0px" />
</asp:TreeView>
</asp:TableCell>
<asp:TableCell ID="TableCell5" runat="server">
<asp:TreeView ID="TreeView5" runat="server"
 ShowLines="True">
<Nodes>
<asp:TreeNode Text="Root Node 1"
 Value="Root Node 1" Expanded="True">
<asp:TreeNode Text="Leaf Node"
 Value="Leaf Node"></asp:TreeNode>
<asp:TreeNode Text="Leaf Node"
 Value="Leaf Node"></asp:TreeNode>
</asp:TreeNode>
<asp:TreeNode Text="Root Node 2"
 Value="Root Node 2"></asp:TreeNode>
</Nodes>
<LeafNodeStyle BackColor="#FFC0C0"
 BorderColor="Blue" BorderStyle="Solid"
 Font-Bold="True"
 Font-Italic="True" />
</asp:TreeView>
</asp:TableCell>
<asp:TableCell ID="TableCell6" runat="server">
<asp:TreeView ID="TreeView7" runat="server"
 ShowCheckBoxes="All">
<Nodes>
<asp:TreeNode Text="Root Node 1"
 Value="Root Node 1" Checked="True"
 Expanded="False"
 ShowCheckBox="True">
<asp:TreeNode Text="Leaf Node"
 Value="Leaf Node"></asp:TreeNode>
<asp:TreeNode Text="Leaf Node"
 Value="Leaf Node"></asp:TreeNode>
</asp:TreeNode>
<asp:TreeNode Text="Root Node 2"
 Value="Root Node 2"></asp:TreeNode>
</Nodes>
</asp:TreeView>
</asp:TableCell>
```

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

As you can see in the preceding listing, we have placed numerous `TreeView` controls on the Web page. The first `TreeView` control contains a navigation structure for this application. After creating the navigation structure for the application, we have created six different `TreeView` controls, each one with different styles of formatting. The output of the preceding listing is shown in Figure 18.4:

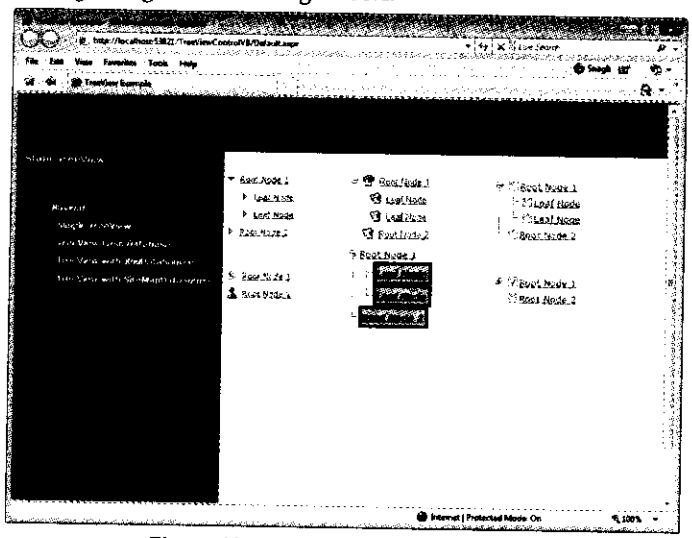


**Figure 18.4: Simple TreeView Example**

To add the nodes of the `TreeView` control at design time, choose the `Edit Nodes` option from the menu or `Nodes` property of the TreeView Control. Once you click the Nodes property option, a TreeView Node Editor appears, as shown earlier in Figure 18.3.

You can now add nodes by clicking the Add a root node button on the Toolbar of the TreeView Node Editor shown in Figure 18.5:

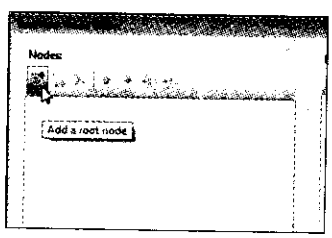



**Figure 18.5: Showing the Toolbar of the Editor Dialog Box**

You can add root and parent nodes by clicking the first button (Add a root node), and child nodes by clicking the second button (Add a child node). Set the `Text` property for the nodes under the Properties panel.

Creating the TreeView nodes using the TreeView Node Editor is the simplest form of `TreeView` you can generate with the help of code. You can also generate `TreeView` based on the XML file, `SiteMapDataSource` or even with databases such as SQL Server.

Now it's time to extend the example to generate the `TreeView` based on the XML data. To do so, we now add a new Web form named `TreeViewXMLData.aspx` to our solution. In order to generate a `TreeView` from the XML data, we need to bind that to an `XmlDataSource`. First, add an `XmlDataSource` control to the Web form as follows:

```
<asp:XmlDataSource ID="XmlDataSource1" runat="server"
  DataFile="~/Navigation.xml"></asp:XmlDataSource>
```

As you can see in the preceding code, we are using an XML file named `Navigation.xml` as the data file for the `XmlDataSource` control. Add the code in the `Navigation.xml` file as shown as follows:

```
<?xml version="1.0" encoding="utf-8" ?>
<application>
  <homepage title="Home Page" value="default.aspx">
      <subpage title ="TreeView From Database" value="treeviewdatabase.aspx"/>
      <subpage title ="TreeView with XMLDataSource"
      value="treeviewxmldata.aspx"/>
      <subpage title ="TreeView with SiteMapDataSource"
      value="treeviewsitemapdatasource.aspx"/>
  </homepage>
</application>
```

After configuring the `XmlDataSource` control its time now to add a `TreeView` control to the Web form and configuring it. Once you placed a `TreeView` control on the Web form, you need to configure it to work with XML data. To configure our `TreeView` control, first select `XmlDataSource` as its data source in the Smart Tag of the TreeView control and then click on the Edit TreeNode Databindings in Smart Tag, as shown in Figure 18.6:

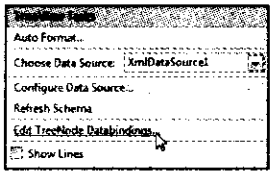



**Figure 18.6: Configuring TreeView Control**

This will open the TreeView Databindings Editor dialog box (Figure 18.7). Add the `homepage` and `subpage` from the Available data bindings section to the Selected data bindings section, as shown in Figure 18.7:

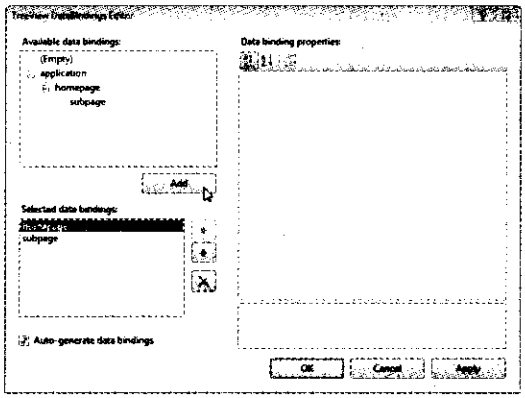



**Figure 18.7: TreeView Databindings Editor**

After selecting all the required data-bindings it is time to configure them. In other words, we will now define the structure of our `TreeView` control. Select one of your databinding; in this case, we are first selecting the `homepage` item and then setting properties for it, as shown in Figure 18.8:

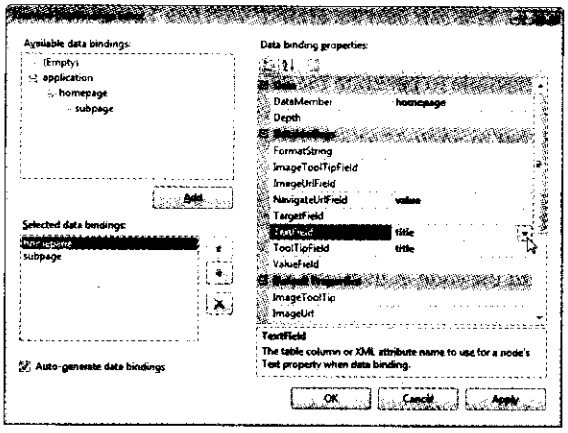



**Figure 18.8: Configuring DataBindings**

Repeat the same steps for the `subpage` item as well and the process is completed. This complete process generates the `TreeView` controls based on the `XmlDataSource`. We are using the `TreeViewXMLData.aspx` page for displaying the data as per the preceding requirement. You can find the code for the `TreeViewXMLData.aspx` page of the `TreeViewControl` application in the Listing 18.2.

**Listing 18.2:** Showing the Code for `TreeViewXMLData.aspx` Page

```
<%@ Page Language="VB" AutoEventWireup="false"
  CodeFile="TreeViewXMLData.aspx.vb"
Inherits="TreeViewXMLData" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
  <title>TreeView from XML Data</title>
  <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <form id="mainForm" runat="server">
      <div>
          <div id="header">
          </div>
          <div id="sidebar">
          <div id="nav">
           
                  <asp:Label ID="Label1" runat="server" Text="Static TreeView"
                   Font-Bold="True" ForeColor="White"></asp:Label>
                   <br />
                  <br />
                  <asp:TreeView ID="TreeView1" runat="server"
                   ImageSet="Arrows">
                  <ParentNodeStyle Font-Bold="False" />
                  <HoverNodeStyle Font-Underline="True"
                      ForeColor="#5555DD" />
                  <SelectedNodeStyle Font-Underline="True"
                   ForeColor="#5555DD" HorizontalPadding="0px"
                   VerticalPadding="0px" />
                  <Nodes>
                  <asp:TreeNode Text="Navigate" Value="Navigate">
```

```
                    <asp:TreeNode Text="Simple TreeView"
                        Value="Simple TreeView"
                        NavigateUrl="~/Default.aspx">
                    </asp:TreeNode>
                    <asp:TreeNode Text="TreeView From Database"
                        Value="TreeView From Database"
                        NavigateUrl="~/TreeViewDatabase.aspx">
                    </asp:TreeNode>
                    <asp:TreeNode Text="TreeView with XmlDataSource"
                        Value="TreeView with XmlDataSource"
                        NavigateUrl="~/TreeViewXMLData.aspx">
                    </asp:TreeNode>
                    <asp:TreeNode Text="TreeView with
                        SiteMapDataSource" Value="TreeView with
                        SiteMapDataSource"
                        NavigateUrl="~/TreeViewSiteMapDataSource.aspx">
                    </asp:TreeNode>
                    </asp:TreeNode>
                    </Nodes>
                    <NodeStyle Font-Names="Tahoma" Font-Size="10pt"
                     ForeColor="Black" HorizontalPadding="5px"
                     NodeSpacing="0px" VerticalPadding="0px" />
                    </asp:TreeView>
                </div>
                </div>
                <div id="content">
                <div class="itemContent">
                    <asp:XmlDataSource ID="XmlDataSource1" runat="server"
                     DataFile="~/Navigation.xml"></asp:XmlDataSource>
                    <br />
                    <asp:Label ID="Label2" runat="server" Text="TreeView
                     generated from XMLDataSource"></asp:Label><br />
                    <br />
                    <asp:TreeView ID="TreeView2" runat="server"
                     DataSourceID="XmlDataSource1">
                    <DataBindings>
                    <asp:TreeNodeBinding DataMember="homepage"
                     TextField="title" NavigateUrlField="value"
                     ToolTipField="title" />
                    <asp:TreeNodeBinding DataMember="subpage"
                     TextField="title" ToolTipField="value"
                     NavigateUrlField="value" />
                    </DataBindings>
                    </asp:TreeView>
                    <br />
                </div>
                <div id="footer">
                    <p class="left">
                    All content copyright &copy; Kogent Solutions Inc.</p>
                </div>
                </div>
        </div>
    </form>
</body>
</html>
```

Again run the `TreeViewControl` application and browse the `TreeViewXMLData.aspx` page by selecting the `TreeView with XmlDataSource` node from the `TreeView` on the left panel of the application. Figure 18.9 shows the output of `TreeViewXMLData.aspx`:

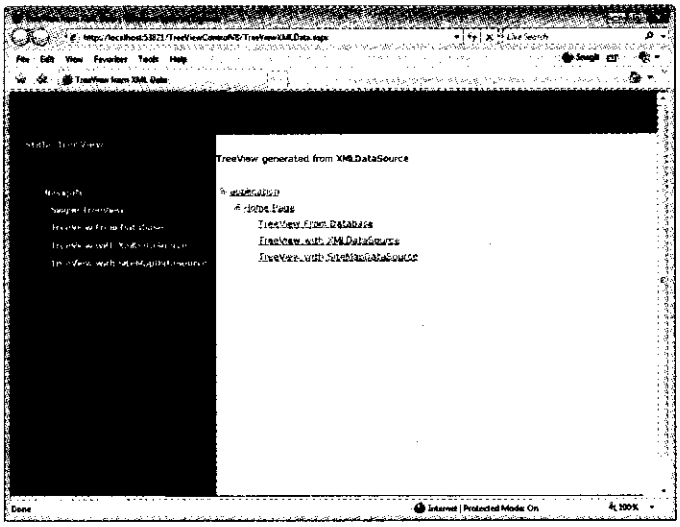



**Figure 18.9: Generating TreeView from XML Data**

We will now generate a `TreeView` control from the `SiteMapDataSource` control. To do this, first add a new Web form named `TreeViewSiteMapDataSource.aspx` to the solution. Next, we will add a sitemap file to our solution by right-clicking on the solution and selecting the Add New Item. In the Add New Item dialog box, select the Site Map Template from the list of available templates and name it as `Web.sitemap`, as shown in Figure 18.10:

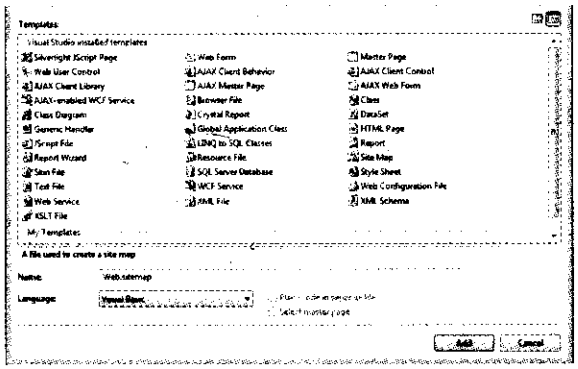


**Figure 18.10: Adding a Site Map to Solution**

Click the Add button of the Add New Item dialog box. This will open the `Web.sitemap` file and Replace the contents of the file with the following code:

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
  <siteMapNode title="TreeView Control Example" url="~/default.aspx" >
    <siteMapNode title="TreeView from database" url="~/treeviewdatabase.aspx"/>
    <siteMapNode title="TreeView with XMLDataSource"
     url="~/treeviewxmldata.aspx"/>
    <siteMapNode title="TreeView with SiteMapDataSource"
     url="~/TreeViewSiteMapDataSource.aspx"/>
    </siteMapNode>
</siteMap>
```

Now add the following line of code in the `TreeViewSiteMapDataSource.aspx` page to add a `SiteMapDataSource` control:

```
<asp:SiteMapDataSource ID="SiteMapSource1" runat="server" />
```

The `SiteMapDataSource` control automatically links itself to `Web.sitemap` file we added in this application. Let's add a `TreeView` control that will be populated based on this `SiteMapDataSource` control. You can find the code for the `TreeViewSiteMapDataSource.aspx` page of the `TreeViewControl` application in Listing 18.3:

**Listing 18.3:** Showing the Code for `TreeViewSiteMapDataSource.aspx` Page

```
<%@ Page Language="VB" AutoEventWireup="False"
  CodeFile="TreeViewSiteMapDataSource.aspx.vb"
  Inherits="TreeViewSiteMapDataSource" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
  <title>TreeView from SiteMapDataSource</title>
  <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <form id="mainForm" runat="server">
      <div>
          <div id="header">
          </div>
          <div id="sidebar">
          <div id="nav">
               
              <asp:Label ID="Label1" runat="server" Text="Static TreeView"
               Font-Bold="True" ForeColor="White"></asp:Label>
               <br />
              <br />
              <asp:TreeView ID="TreeView1" runat="server"
               ImageSet="Arrows">
              <ParentNodeStyle Font-Bold="False" />
              <HoverNodeStyle Font-Underline="True"
               ForeColor="#5555DD" />
              <SelectedNodeStyle Font-Underline="True"
               ForeColor="#5555DD" HorizontalPadding="0px"
               VerticalPadding="0px" />
              <Nodes>
              <asp:TreeNode Text="Navigate" Value="Navigate">
              <asp:TreeNode Text="Simple TreeView"
                  Value="Simple TreeView"
                  NavigateUrl="~/Default.aspx">
              </asp:TreeNode>
              <asp:TreeNode Text="TreeView From Database"
                  Value="TreeView From Database"
                  NavigateUrl="~/TreeViewDatabase.aspx">
              </asp:TreeNode>
              <asp:TreeNode Text="TreeView with XmlDataSource"
                  Value="TreeView with XmlDataSource"
                  NavigateUrl="~/TreeViewXMLData.aspx">
              </asp:TreeNode>
              <asp:TreeNode Text="TreeView with
                  SiteMapDataSource" Value="TreeView with
                  SiteMapDataSource"
                  NavigateUrl="~/TreeViewSiteMapDataSource.aspx">
              </asp:TreeNode>
              </asp:TreeNode>
              </Nodes>
              <NodeStyle Font-Names="Tahoma" Font-Size="10pt"
               ForeColor="Black" HorizontalPadding="5px"
               NodeSpacing="0px" VerticalPadding="0px" />
```

```
                    </asp:TreeView>
                </div>
                </div>
                <div id="content">
                <div class="itemContent">
                        <br />
                        <asp:Label ID="Label2" runat="server" Text="TreeView
                         generated from SiteMapDataSource"></asp:Label><br />
                        <br />
                        <asp:SiteMapDataSource ID="SiteMapSource1" runat="server" />
                        <asp:TreeView ID="TreeView2" DataSourceID="SiteMapSource1"
                         runat="server">
                        <DataBindings>
                        <asp:TreeNodeBinding TextField="Title"
                         NavigateUrlField="Url" />
                        </DataBindings>
                        </asp:TreeView>
                        <br />
                </div>
                <div id="footer">
                        <p class="left">
                        All content copyright &copy; Kogent Solutions Inc.</p>
                </div>
                </div>
        </div>
    </form>
</body>
</html>
```

You can see the output of the preceding listing in Figure 18.11:

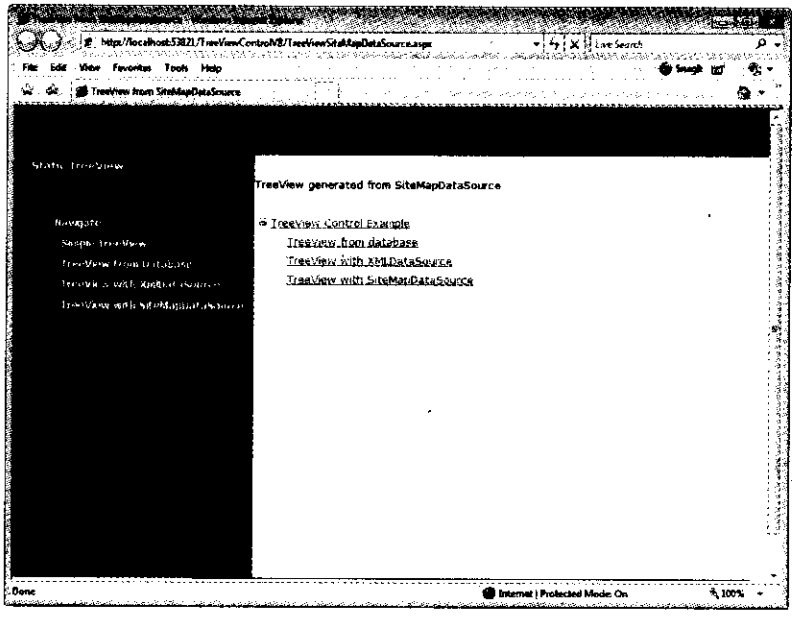



**Figure 18.11: Generating TreeView from SiteMapDataSource**

# Generating **TreeView** from a Database

In case you want to generate `TreeView` control based on the database, such as SQL Server, you need to generate the `TreeView` control programmatically by using the `TreeView` class since no wizard is available for this..

We will extend our `TreeViewControl` application for the current example. Just add a new Web form named `TreeViewDataBase.aspx` to the application. We are using Pubs database of the SQL Server. As a first step, drag a `SQLDataSource` control from the Toolbox to the Web form, and configure it to work with Pubs database existing in SQL Server (To know how to add work with the SQL Server database refer chapter 17: *Web Forms: Standard Controls* ). In this application we will display the first names of the authors and their books in `TreeView` control. You can find the code for the `TreeViewDataBase.aspx` page of the `TreeViewControl` application in Listing 18.4:

**Listing 18.4:** Code for the `TreeViewDataBase.aspx` Page

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

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
  <title>Treeview from DataBase</title>
  <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <form id="mainForm" runat="server">
        <div>
        <div id="header">
        </div>
        <div id="sidebar">
        <div id="nav">
                 
                <asp:Label ID="Label1" runat="server" Text="Static TreeView"
                Font-Bold="True" ForeColor="White"></asp:Label>
                 <br />
                <br />
                <asp:TreeView ID="TreeView1" runat="server"
                ImageSet="Arrows">
                <ParentNodeStyle Font-Bold="False" />
                <HoverNodeStyle Font-Underline="True"
                    ForeColor="#5555DD" />
                <SelectedNodeStyle Font-Underline="True"
                 ForeColor="#5555DD" HorizontalPadding="0px"
                 VerticalPadding="0px" />
                <Nodes>
                <asp:TreeNode Text="Navigate" Value="Navigate">
                <asp:TreeNode Text="Simple TreeView"
                    Value="Simple TreeView"
                    NavigateUrl="~/Default.aspx"></asp:TreeNode>
                <asp:TreeNode Text="TreeView From Database"
                    Value="TreeView From Database"
                    NavigateUrl="~/TreeViewDatabase.aspx">
                    </asp:TreeNode>
                <asp:TreeNode Text="TreeView with XmlDataSource"
                    Value="TreeView with XmlDataSource"
                    NavigateUrl="~/TreeViewXMLData.aspx">
                </asp:TreeNode>
                <asp:TreeNode Text="TreeView with
                    SiteMapDataSource" Value="TreeView with
                    SiteMapDataSource"
                    NavigateUrl="~/TreeViewSiteMapDataSource.aspx">
                </asp:TreeNode>
                </asp:TreeNode>
                </Nodes>
                <NodeStyle Font-Names="Tahoma" Font-Size="10pt"
                 ForeColor="Black" HorizontalPadding="5px"
                 NodeSpacing="0px" VerticalPadding="0px" />
```

```
                </asp:TreeView>
            </div>
            </div>
            <div id="content">
            <div class="itemContent">
                <br />
                <asp:SqlDataSource ID="SqlDataSource1" runat="server"
                ConnectionString="<%$
                ConnectionStrings:pubsConnectionString %>"
                SelectCommand="SELECT [au_fname] FROM
                [authors]"></asp:SqlDataSource>
                <asp:Label ID="Label2" runat="server" Text="Treeview Control
                bind to SQL Server  PUBS  database"></asp:Label><br />
                <br />
                <asp:TreeView ID="TreeView2" runat="server"
                MaxDataBindDepth="2">
                <Nodes>
                <asp:TreeNode PopulateOnDemand="True" Text="Books by
                authors" Value="booksbyauthors">
                </asp:TreeNode>
                </Nodes>
                </asp:TreeView>
                <br />
                <br />
            </div>
            <div id="footer">
                <p class="left">
                All content copyright &copy; Kogent Solutions Inc.</p>
            </div>
            </div>
    </div>
    </form>
</body>
</html>
```

Now, add the code in the code-behind file of the `TreeViewDataBase.aspx` page, as shown in Listing 18.5:

**Listing 18.5:** Code for the Code-Behind File of the `TreeViewDataBase.aspx` Page

```
Imports System.Data
Imports System.Data.SqlClient

Partial Class TreeViewDatabase
    Inherits System.Web.UI.Page

    Sub GetAuthorsName(ByVal node As TreeNode)
        Dim sqlQuery As New SqlCommand("SET ROWCOUNT 7 Select au_fname, au_id
        From authors")
        Dim ResultSet As DataSet
        ResultSet = PopulateDataFunction(sqlQuery)
        If ResultSet.Tables.Count > 0 Then
            Dim row As DataRow
            For Each row In ResultSet.Tables(0).Rows
                Dim NewNode As TreeNode = New TreeNode
                NewNode.Text = row("au_fname").ToString()
                NewNode.Value = row("au_id").ToString()
                NewNode.PopulateOnDemand = True
                NewNode.SelectAction = TreeNodeSelectAction.Expand
                node.ChildNodes.Add(NewNode)
            Next
        End If
    End Sub
```

```
    Sub GetBooksName(ByVal node As TreeNode)
        Dim sqlQuery As New SqlCommand
        sqlQuery.CommandText = "SELECT title, titleauthor.title_id,
           titleauthor.au_id FROM titles INNER JOIN titleauthor ON titles.title_id
           = titleauthor.title_id where titles.title_id = titleauthor.title_id AND
           titleauthor.au_id = @authorsid"
        sqlQuery.Parameters.Add("@authorsid", SqlDbType.Char).Value =
           node.Value.ToString
        Dim ResultSet As DataSet
        ResultSet = PopulateDataFunction(sqlQuery)

        If ResultSet.Tables.Count > 0 Then
                Dim row As DataRow
                For Each row In ResultSet.Tables(0).Rows
                    Dim NewNode As TreeNode = New TreeNode()
                    NewNode.Text = row("title").ToString()
                    NewNode.PopulateOnDemand = False
                    NewNode.SelectAction = TreeNodeSelectAction.None
                    node.ChildNodes.Add(NewNode)
                Next
        End If
    End Sub

    Function PopulateDataFunction(ByVal sqlQuery As SqlCommand) As DataSet
        Dim connectionString As String
        connectionString =
        ConfigurationManager.ConnectionStrings_
        ("PubsConnectionString").ConnectionString
        Dim dbConnection As New SqlConnection
        dbConnection.ConnectionString = connectionString
        Dim dbAdapter As New SqlDataAdapter
        dbAdapter.SelectCommand = sqlQuery
        sqlQuery.Connection = dbConnection
        Dim resultsDataSet As DataSet = New DataSet
        Try
                dbAdapter.Fill(resultsDataSet)
        Catch ex As Exception
                End Try
        Return resultsDataSet
    End Function
    Protected Sub TreeView2_TreeNodePopulate(ByVal sender As Object, ByVal e As
        System.Web.UI.WebControls.TreeNodeEventArgs) Handles
        TreeView2.TreeNodePopulate
    If e.Node.ChildNodes.Count = 0 Then
        Select Case e.Node.Depth
                Case 0
                    GetAuthorsName(e.Node)
                Case 1
                    GetBooksName(e.Node)
        End Select
    End If
    End Sub
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
        Handles Me.Load
    TreeView2.ExpandDepth = 1
    End Sub
End Class
```

To understand the preceding code, we will now discuss the code step by step. In the first step, we will add the code for the `TreeView2_TreeNodePopulate` method, as shown in Listing 18.6:

**Listing 18.6:** Code for the `TreeView2_TreeNodePopulate` Method

```
Protected Sub TreeView2_TreeNodePopulate(ByVal sender As Object, ByVal e As
  System.Web.UI.WebControls.TreeNodeEventArgs) Handles TreeView2.TreeNodePopulate
    If e.Node.ChildNodes.Count > 0 Then
        Select Case e.Node.Depth
            Case 0
                GetAuthorsName(e.Node)
            Case 1
                GetBooksName(e.Node)
        End Select
    End If
End Sub
```

The preceding listing checks the `TreeView2` control. If it is found that there does not exist any node in the `TreeView2` control, the `GetAuthorsName` function is called to generate the first-level nodes containing the author names. In the `GetAuthorsName` function, we query the database to retrieve the first seven records and generate the first-level tree nodes by using the `node.ChildNodes.Add` method, as shown in Listing 18.7:

**Listing 18.7:** Code for Generating the First-Level Tree Node

```
Sub GetAuthorsName(ByVal node As TreeNode)
Dim sqlQuery As New SqlCommand("SET ROWCOUNT 7 Select au_fname, au_id
  From authors")
  Dim ResultSet As DataSet
  ResultSet = PopulateDataFunction(sqlQuery)
      If ResultSet.Tables.Count > 0 Then
          Dim row As DataRow
          For Each row In ResultSet.Tables(0).Rows
          Dim NewNode As TreeNode = New TreeNode
          NewNode.Text = row("au_fname").ToString()
          NewNode.Value = row("au_id").ToString()
          NewNode.PopulateOnDemand = True
          NewNode.SelectAction = TreeNodeSelectAction.Expand
          node.ChildNodes.Add(NewNode)
          Next
      End If
End Sub
```

After generating the authors' `names`, the code again checks for the `TreeView2_TreeNodePopulate` event and this time the `GetBooksName` function is called to generate the second-level nodes containing the books written by the authors, whose names have been retrieved from the database by using the `GetAuthorsName` function, as shown in Listing 18.8:

**Listing 18.8:** Code for Retrieving the Author Names from the Database

```
Sub GetBooksName(ByVal node As TreeNode)
  Dim sqlQuery As New SqlCommand
  sqlQuery.CommandText = "SELECT title, titleauthor.title_id,
    titleauthor.au_id FROM titles INNER JOIN titleauthor ON titles.title_id
    = titleauthor.title_id where titles.title_id = titleauthor.title_id AND
    titleauthor.au_id = @authorsid"
  sqlQuery.Parameters.Add("@authorsid", SqlDbType.Char).Value =
    node.Value.ToString
  Dim ResultSet As DataSet
  ResultSet = PopulateDataFunction(sqlQuery)
  If ResultSet.Tables.Count > 0 Then
      Dim row As DataRow
      For Each row In ResultSet.Tables(0).Rows
          Dim NewNode As TreeNode = New TreeNode()
          NewNode.Text = row("title").ToString()
          NewNode.PopulateOnDemand = False
          NewNode.SelectAction = TreeNodeSelectAction.None
          node.ChildNodes.Add(NewNode)
```

```
            Next
        End If
    End Sub
```

In the preceding listing, we have used a parameterized query to get the book names only for those authors whose names have already been retrieved from the database and those authors that exist at the first level in the `TreeView2` control. This generates the complete tree view. The output of the TreeViewControl application is shown in Figure 18.12:

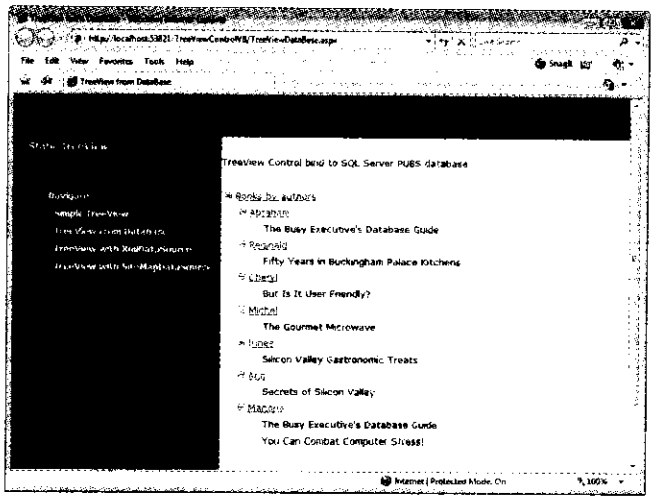



**Figure 18.12: Generating TreeView from the Database**

# Using the **Menu** Class

The `Menu` class provides various properties, methods, and events that provide a greater flexibility while displaying data using the `Menu` control.

Here is the inheritance hierarchy for the `Menu` class:

```
System.Object
    System.Web.UI.Control
        System.Web.UI.WebControls.WebControl
            System.Web.UI.WebControls.BaseDataBoundControl
                System.Web.UI.WebControls.HierarchicalDataBoundControl
                    System.Web.UI.WebControls.Menu
```

You can find the noteworthy properties of the `Menu` class in Table 18.4:

**Table 18.4: Noteworthy Properties of Menu Class**

| Property | Description |
|---|---|
| `DataBindings` | Obtains a collection of `MenuItemBinding` objects that define the relationship between a data item and the menu item to which the object is bound |
| `DisappearAfter` | Obtains or sets the duration for which a dynamic menu is displayed after the mouse pointer is no longer positioned over the menu |
| `DynamicBottomSeparatorImageUrl` | Obtains or sets the `URL` to an image to display at the bottom of each dynamic menu item to separate it from other menu items |
| `DynamicEnableDefaultPopOutImage` | Obtains or sets a value indicating whether default pop-out image is displayed in the dynamic part of the menu or not |
| `DynamicHorizontalOffset` | Obtains or sets the number of pixels that a dynamic menu item is shifted |

**Table 18.4: Noteworthy Properties of Menu Class**

| | |
|---|---|
| | relative to its parent menu item |
| DynamicHoverStyle | Obtains a reference to the MenuItemStyle object that allows you to set the appearance of a dynamic menu item when the mouse pointer is positioned over it |
| DynamicItemFormatString | Obtains or sets additional text shown with all menu items that are dynamically displayed |
| DynamicItemTemplate | Obtains or sets the template that contains the custom content to render for a dynamic menu |
| DynamicMenuItemStyle | Obtains a reference to the MenuItemStyle object that allows you to set the appearance of the menu items within a dynamic menu |
| DynamicMenuStyle | Obtains a reference to the MenuItemStyle object that allows you to set the appearance of a dynamic menu |
| DynamicPopOutImageTextFormatString | Obtains or sets the alternate text for the image used to indicate that a dynamic menu item has a submenu |
| DynamicPopOutImageUrl | Obtains or sets the URL to a custom image that is displayed in a dynamic menu item when the dynamic menu item has a submenu |
| DynamicSelectedStyle | Obtains a reference to the MenuItemStyle object that allows you to set the appearance of the dynamic menu item selected by the user |
| DynamicTopSeparatorImageUrl | Obtains or sets the URL to an image to display at the top of each dynamic menu item to separate it from other menu items |
| DynamicVerticalOffset | Obtains or sets the number of pixels that a dynamic menu item is shifted relative to its parent menu item |
| Items | Obtains a MenuItemCollection object that contains all menu items in the Menu control |
| ItemWrap | Obtains or sets a value indicating whether the text for menu items should wrap |
| LevelMenuItemStyles | Obtains a MenuItemStyleCollection object that contains the style settings that are applied to menu items based on their level in a Menu control |
| LevelSelectedStyles | Obtains a MenuItemStyleCollection object that contains the style settings that are applied to the selected menu item based on its level in a Menu control |
| LevelSubMenuStyles | Obtains a MenuItemStyleCollection object that contains the style settings that are applied to the submenu items in the static menu based on their level in a Menu control |
| MaximumDynamicDisplayLevels | Obtains or sets the number of menu levels to render for a dynamic menu |
| Orientation | Obtains or sets the direction in which to render the Menu control |
| PathSeparator | Obtains or sets the character used to delimit the path of a menu item in a Menu control |
| ScrollDownImageUrl | Obtains or sets the URL to an image displayed in a dynamic menu to indicate that the user can scroll down for additional menu items |
| ScrollDownText | Obtains or sets the alternate text for the image specified in the ScrollDownImageUrl property |

| Table 18.4: Noteworthy Properties of Menu Class | |
|---|---|
| ScrollUpImageUrl | Obtains or sets the URL to an image displayed in a dynamic menu to indicate that the user can scroll up for additional menu items |
| ScrollUpText | Obtains or sets the alternate text for the image specified in the ScrollUpImageUrl property |
| SelectedItem | Obtains the selected menu item |
| SelectedValue | Obtains the value of the selected menu item |
| SkipLinkText | Obtains or sets the alternate text for a hidden image read by screen readers to provide the ability to skip the list of links |
| StaticBottomSeparatorImageUrl | Obtains or sets the URL to an image displayed as the separator at the bottom of each static menu item |
| StaticDisplayLevels | Obtains or sets the number of menu levels to display in a static menu |
| StaticEnableDefaultPopOutImage | Obtains or sets a value indicating whether the built-in image is displayed to indicate that a static menu item has a submenu |
| StaticHoverStyle | Obtains a reference to the MenuItemStyle object that allows you to set the appearance of a static menu item when the mouse pointer is positioned over it |
| StaticItemFormatString | Obtains or sets additional text shown with all static menu items |
| StaticItemTemplate | Obtains or sets the template that contains the custom content to render for a static menu |
| StaticMenuItemStyle | Obtains a reference to the MenuItemStyle object that allows you to set the appearance of the menu items in a static menu |
| StaticMenuStyle | Obtains a reference to the MenuItemStyle object that allows you to set the appearance of a static menu |
| StaticPopOutImageTextFormatString | Obtains or sets the alternate text for the pop-out image used to indicate that a static menu item has a submenu |
| StaticPopOutImageUrl | Obtains or sets the URL to an image displayed to indicate that a static menu item has a submenu |
| StaticSelectedStyle | Obtains a reference to the MenuItemStyle object that allows you to set the appearance of the menu item selected by the user in a static menu |
| StaticSubMenuIndent | Obtains or sets the amount of space, in pixels, to indent submenus within a static menu |
| StaticTopSeparatorImageUrl | Obtains or sets the URL to an image displayed as the separator at the top of each static menu item |
| Target | Obtains or sets the target window or frame in which to display the Web page content associated with a menu item |

You can find the noteworthy methods of the Menu class in Table 18.5:

| Table 18.5: Noteworthy Methods of Menu Class | |
|---|---|
| FindItem | Retrieves the menu item at the specified value path |
| RenderBeginTag | Adds tag attributes and writes the markup for the opening Tag of the control to the output stream emitted to the browser or device |

| Table 18.5: Noteworthy Methods of Menu Class | |
|---|---|
| RenderEndTag | Performs final markup and writes the HTML closing tag of the control to the output stream emitted to the browser or device |

You can find the noteworthy events of the Menu class in Table 18.6:

| Table 18.6: Noteworthy Events of the Menu Class | |
|---|---|
| MenuItemclick | Occurs when a menu item in a Menu control is clicked |
| MenuItemDataBound | Occurs when a menu item in a Menu control is bound to data |

# The **Menu** Control

The Menu control is another navigation control, which is also used to display site navigation information. The Menu control can; however, display the site structure vertically as well as horizontally. It can be used as a databound control, for example, binding the menu control with a SiteMapDataSource control. It can also retrieve the data to be displayed from the items that are added to the Items collection of the Menu control at runtime. The Menu control supports binding data with hierarchical datasource, such as XML files. You can also add the database items to the Menu control at runtime using the Items collection of the Menu control. The Menu control consists of one or more MenuItems properties displayed at different levels of hierarchy. Each MenuItem, in turn, consists of properties to set the style of the individual MenuItem. The menu control contains two types of menus:

- **Static menu** – This type of menu is displayed completely with all the menu options on the screen. The entire structure of the static menu, including the parent menu items and their sub menus, is visible.
- **Dynamic menu** – This type of menu contains static part which is displayed on the scree as well as the dynamic part which appears when user passes the mouse pointer over the static part of the menu.

Figure 18.13 shows a simple Menu control from the MenuControl example:

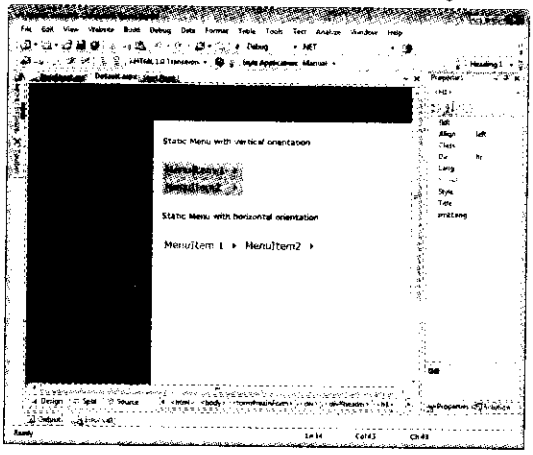



**Figure 18.13: MenuControl Example**

## *Using the **XmlDataSource** Class*

The XmlDataSource control is a data source used by the data-bound controls to display data in the tabular format. The XmlDataSource class contains the methods and properties for the XmlDataSource control.

Here is the inheritance hierarchy for the XmlDataSource class:

```
System.Object
    System.Web.UI.Control
        System.Web.UI.HierarchicalDataSourceControl
            System.Web.UI.WebControls.XmlDataSource
```

You can find the noteworthy properties of the `XmlDataSource` class in Table 18.7:

**Table 18.7: Noteworthy Properties of the XmlDataSource Class**

| | |
|---|---|
| `CacheDuration` | Obtains or set the time period, in seconds, to caches the data that the data source control has retrieved |
| `CacheExpirationPolicy` | Obtains or sets the cache expiration policy that is combined with the cache duration to describe the caching behavior of the cache that the data source control uses |
| `CacheKeyDependency` | Obtains or sets a user-defined key dependency that is linked to all data cache objects created by the data source control. All cache objects explicitly expire when the key expires |
| `Data` | Obtains or sets a block of XML data to which the data source control binds |
| `DataFile` | Sets the file name of an XML file to which the data source control binds |
| `EnableCaching` | Obtains or sets a value to enable or disable the data caching for `XmlDataSource` control |
| `Transform` | Obtains or sets a block of Extensible Stylesheet Language (XSL) data that defines an XSLT transformation to be performed on the XML data managed by the `XmlDataSource` control |
| `TransformArgumentList` | Provides a list of XSLT arguments that are used with the style sheet defined by the `Transform` or `TransformFile` properties to perform a transformation on the XML data |
| `TransformFile` | Sets the file name of an Extensible Stylesheet Language (XSL) file (.xsl) that defines an XSLT transformation to be performed on the XML data managed by the `XmlDataSource` control |
| `XPath` | Sets an XPath expression to be applied to the XML data contained by the Data property or by the XML file indicated by the DataFile property |

You can find the noteworthy methods of the `XmlDataSource` class in Table 18.8:

**Table 18.8: Noteworthy Methods of the XmlDataSource Class**

| | |
|---|---|
| `GetXmlDocument` | Loads the XML data into memory directly from the underlying data storage or from the cache, and return the data in the form of `XmlDataDocument` object |
| `Save` | Saves the XML data to the disk that is currently placed in the memory by the `XmlDataSource` control |

You can find the noteworthy events of the `XmlDataSource` class in Table 18.9:

**Table 18.9: Noteworthy Events of the XmlDataSource Class**

| | |
|---|---|
| `Transforming` | Occurs when the style sheet, that is defined by the `Transform` property or identified by the `TransformFile` property, is applied to the XML data |

## Using the XmlDataSource Control

You can use the `XmlDataSource` control to bind data with the `Menu` control. The `XmlDataSource` control binds the XML data to the `MenuItem` properties. The binded data can be read from the `sitemap.xml` file. The `XmlDataSource` control can be found under the Data tab in the Toolbox of the Smart Tag. To add the `XmlDataSource` control on the Web page, drag the control from the Toolbar and drop it control on the Web form designer. After this control is added to the Web form designer, the control is displayed as shown in Figure 18.14:

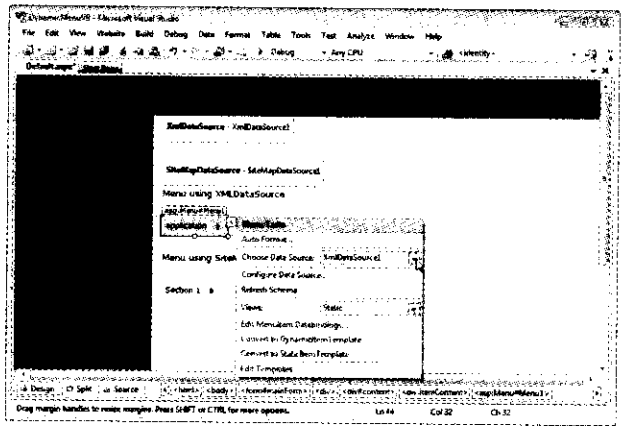


**Figure 18.14: Using XmlDataSource with Menu Control**

To set the XML data with the `MenuItems` property of the menu control, we need to bind the XML file with the `XmlDataSource` control. Click the Smart Tag at the upper-right corner of the control and select the `XmlDataSource` as the Data Source for the `Menu` control, as shown in Figure 18.14.

The `XmlDataSource` is using an XML file (`Data.xml`) as the source of data; this can be set through the Properties window, as shown in Figure 18.15:

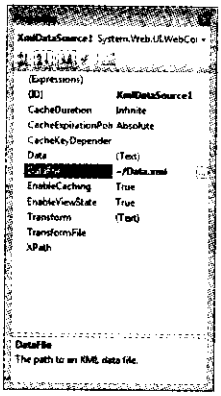


**Figure 18.15: Configuring the XmlDataSource**

## Menu Display Properties

You can change the look of the `Menu` control by setting the following properties:

- `Orientation`—You can set this property to horizontal to display menu either at the top or at the bottom of the page. The default orientation value is Vertical.
- `StaticItemFormatString`—You can set this property to display the value for the static menus.
- `DynamicItemFormatString`—You can set this property to display the value for the dynamic menus.

- `StaticPopOutImageUrl`—You can set the replacement image for `StaticPopOutImage`. This means that menu items can be hovered over to display all of its child items. The default image will appear dynamically and will continue to be used for menu items.

### *Menu Styles*

The `Menu` control supports various styles at different levels, such as on individual `MenuItems`, `MenuLevels`, or `MenuSubLevels`. You can set styles for all `MenuLevels` and `MenuItems` that appear as both statically and dynamically on a Web page. The following are the different style properties:

- `Static/DynamicMenuStyle`—It controls properties for all `MenuItem` property. It also controls properties, like `HorizontalPadding` and `VerticalPadding`.
- `DynamicMenuItemStyle`—It controls the style of the individual `MenuItems` property. It also includes style properties, such as `ItemSpacing`, `VerticalPadding`, and `HorizontalPadding`.
- `DynamicSelectedStyle`—It sets the style of the individual nodes in the `TreeView` control that are currently selected. In other words, it can be described as a collection of style properties for the selected `MenuItem`. It also includes style properties, such as `HorizontalPadding`, `VerticalPadding`, and `ItemSpacing`.
- `DynamicHoverStyle`—It sets the style properties of the individual `MenuItem` that is currently selected. In other words, it can be set for the selected `MenuItem`, while you hover your mouse over a `MenuItem`. It also includes properties, like `ItemSpacing`, `HorizontalPadding`, and `VerticalPadding`.

## Creating Static Menus

The menus in ASP.NET 3.5 can be easily created by simply dragging and dropping the `Menu` controls from the Toolbox on to the designer. You can find this as `MenuControlVB application`. You can find the code of `MenuControlVB` application in the Code\ASP.NET\Chapter 18\`MenuControlVB` folder on the CD. You can also add Static Menus by double clicking the `Menu` control in the Toolbox. The `Menu` control is added automatically to the designer. Click the Edit Menu Items in the list. A Menu Item Editor will appear. Insert nodes at different levels according to your requirements. You can also set the text for the nodes by setting the `Text` property in the Properties pane, as shown in Figure 18.16:

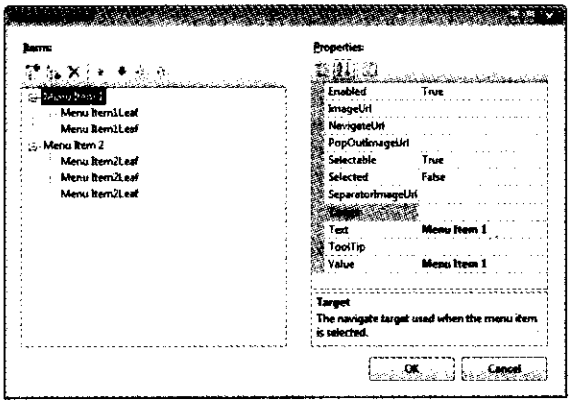



**Figure 18.16: Menu Item Editor**

After inserting the nodes, click the OK button to finish. Replace the code of the `Default.aspx` page of the `MenuControl` application with the code given in Listing 18.9:

**Listing 18.9:** Code for the `Default.aspx` Page

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

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
  <title>Menu Example</title>
  <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <form id="mainForm" runat="server">
  <div>
      <div id="header">
      </div>
      <div id="sidebar">
      <div id="nav">
         
      </div>
      </div>
      <div id="content">
      <div class="itemContent">
          <br />
          <asp:Label ID="Label1" runat="server" Text="Static Menu with vertical
           orientation"></asp:Label><br />
          <br />
          <asp:Menu ID="Menu1" runat="server" BackColor="#B5C7DE"
           DynamicHorizontalOffset="2"
           Font-Names="Verdana" Font-Size="Medium" ForeColor="#284E98"
           StaticSubMenuIndent="10px"
           StaticDisplayLevels="2">
          <StaticMenuItemStyle HorizontalPadding="5px" VerticalPadding="2px" />
          <DynamicHoverStyle BackColor="#284E98" ForeColor="White" />
          <DynamicMenuStyle BackColor="#B5C7DE" />
          <StaticSelectedStyle BackColor="#507CD1" />
          <DynamicSelectedStyle BackColor="#507CD1" />
          <DynamicMenuItemStyle HorizontalPadding="5px" VerticalPadding="2px" />
          <Items>
          <asp:MenuItem Text="MenuItem1" Value="MenuItem1">
          <asp:MenuItem Text="MenuItem1Leaf" Value="MenuItem1Leaf"></asp:MenuItem>
          <asp:MenuItem Text="MenuItem1Leaf" Value="MenuItem1Leaf"></asp:MenuItem>
          </asp:MenuItem>
          <asp:MenuItem Text="MenuItem2" Value="MenuItem2">
          <asp:MenuItem Text="MenuItem2Leaf" Value="MenuItem2Leaf"></asp:MenuItem>
          <asp:MenuItem Text="MenuItem2Leaf" Value="MenuItem2Leaf"></asp:MenuItem>
          <asp:MenuItem Text="MenuItem2Leaf" Value="MenuItem2Leaf"></asp:MenuItem>
          </asp:MenuItem>
          </Items>
          <StaticHoverStyle BackColor="#284E98" ForeColor="White" />
          </asp:Menu>
          <br />
          <asp:Label ID="Label2" runat="server" Text="Static Menu with horizontal
           orientation"></asp:Label><br />
          <br />
          <asp:Menu ID="Menu2" runat="server" BackColor="#FFFBD6"
           DynamicHorizontalOffset="2"
           Font-Names="Verdana" Font-Size="Medium" ForeColor="#990000"
           StaticSubMenuIndent="10px"
           Orientation="Horizontal" StaticDisplayLevels="2">
          <StaticMenuItemStyle HorizontalPadding="5px" VerticalPadding="2px" />
          <DynamicHoverStyle BackColor="#990000" ForeColor="White" />
          <DynamicMenuStyle BackColor="#FFFBD6" />
          <StaticSelectedStyle BackColor="#FFCC66" />
          <DynamicSelectedStyle BackColor="#FFCC66" />
          <DynamicMenuItemStyle HorizontalPadding="5px" VerticalPadding="2px" />
```

```
                <Items>
                <asp:MenuItem Text="MenuItem1" Value="MenuItem1">
                <asp:MenuItem Text="MenuItem1Leaf" Value="MenuItem1Leaf"></asp:MenuItem>
                </asp:MenuItem>
                <asp:MenuItem Text="MenuItem2" Value="MenuItem2">
                <asp:MenuItem Text="MenuItem2Leaf" Value="MenuItem2Leaf"></asp:MenuItem>
                <asp:MenuItem Text="MenuItem2Leaf" Value="MenuItem2Leaf"></asp:MenuItem>
                </asp:MenuItem>
                </Items>
                <StaticHoverStyle BackColor="#990000" ForeColor="White" />
                </asp:Menu>
                <br />
        </div>
        <div id="footer">
                <p class="left">
                All content copyright &copy; Kogent Solutions Inc.Inc.</p>
        </div>
        </div>
    </div>
    </form>
</body>
</html>
```

The output of the MenuControl application is shown in Figure 18.17:

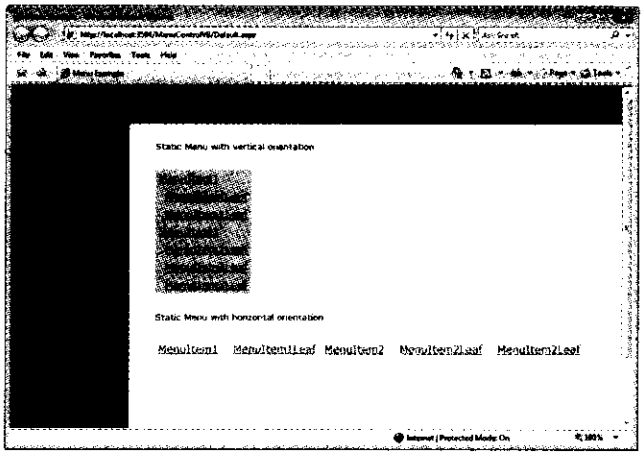



**Figure 18.17: MenuControl Example**

# Creating Dynamic Menus

In this section, we are going to use the `Menu` control with the XML and `SiteMapDataSource` as we did in the `TreeViewControl` example earlier. We have created an application named `DynamicMenu`, and you can find this application as `DynamicMenuVB`. You can find the code of `DynamicMenuVB` application in the `Code\ASP.NET\Chapter 18\DynamicMenuVB` folder on the CD.. Add the code, as shown in Listing 18.10, in the default page of the `DynamicMenu` application:

**Listing 18.10:** Code for the `Default.aspx` Page

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

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
```

```
    <title>DynamicMenu Example</title>
    <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="mainForm" runat="server">
    <div>
        <div id="header">
        </div>
        <div id="sidebar">
        <div id="nav">
             
        </div>
        </div>
        <div id="content">
        <div class="itemContent">
            <asp:XmlDataSource ID="XmlDataSource1" runat="server"
            DataFile="~/Data.xml"></asp:XmlDataSource>
            <br />
            <asp:SiteMapDataSource ID="SiteMapDataSource1"
            runat="server" />
            <asp:Label ID="Label1" runat="server" Text="Menu using
            XMLDataSource"></asp:Label><br />
            <br />
            <asp:Menu ID="Menu1" runat="server" BackColor="#FFFBD6"
            DataSourceID="XmlDataSource1"
            DynamicHorizontalOffset="2" Font-Names="Tahoma" Font-
             Size="Medium" ForeColor="#990000"
             StaticSubMenuIndent="10px" Orientation="Horizontal">
            <StaticMenuItemStyle HorizontalPadding="5px"
             VerticalPadding="2px" />
            <DynamicHoverStyle BackColor="#990000" ForeColor="White"
              />
            <DynamicMenuStyle BackColor="#FFFBD6" />
            <StaticSelectedStyle BackColor="#FFCC66" />
            <DynamicSelectedStyle BackColor="#FFCC66" />
            <DynamicMenuItemStyle HorizontalPadding="5px"
             VerticalPadding="2px" />
            <DataBindings>
            <asp:MenuItemBinding DataMember="setction"
             NavigateUrlField="value" TextField="title"
             ValueField="value" />
            <asp:MenuItemBinding DataMember="subpage"
             NavigateUrlField="value" TextField="title"
             ToolTipField="title" ValueField="value" />
            </DataBindings>
            <StaticHoverStyle BackColor="#990000" ForeColor="White"
               />
            </asp:Menu>
            <br />
            <asp:Label ID="Label2" runat="server" Text="Menu using
             SiteMapDataSource"></asp:Label>
            <br />
            <br />
            <asp:Menu ID="Menu2" runat="server" BackColor="#FFFBD6"
             DataSourceID="SiteMapDataSource1"
             DynamicHorizontalOffset="2" Font-Names="Tahoma" Font-
             Size="Medium" ForeColor="#990000"
             StaticSubMenuIndent="10px">
            <StaticMenuItemStyle HorizontalPadding="5px"
             VerticalPadding="2px" />
            <DynamicHoverStyle BackColor="#990000" ForeColor="White"
              />
```

```
                <DynamicMenuStyle BackColor="#FFFBD6" />
                <StaticSelectedStyle BackColor="#FFCC66" />
                <DynamicSelectedStyle BackColor="#FFCC66" />
                <DynamicMenuItemStyle HorizontalPadding="5px"
                 VerticalPadding="2px" />
                <DataBindings>
                <asp:MenuItemBinding DataMember="setction"
                 NavigateUrlField="value" TextField="title"
                 ValueField="value" />
                <asp:MenuItemBinding DataMember="subpage"
                 NavigateUrlField="value" TextField="title"
                 ToolTipField="title" ValueField="value" />
                </DataBindings>
                <StaticHoverStyle BackColor="#990000" ForeColor="White"
                    />
                </asp:Menu>
                  <br />
                <br />
            </div>
            <div id="footer">
                <p class="left">
                All content copyright &copy; Kogent Solutions Inc.</p>
            </div>
            </div>
        </div>
        </form>
    </body>
    </html>
```

Add an XML file to your Web application, say, `Data.xml` and add the code, as shown in Listing 18.11, in the `Data.xml` file:

**Listing 18.11:** Code for the `Data.xml` File

```
<?xml version="1.0" encoding="utf-8" ?>
<application>
  <setction title="Section 1" value="default.aspx">
        <subpage title ="Page 1" value="page1.aspx"/>
        <subpage title ="Page 2" value="page2.aspx"/>
        <subpage title ="Page 3" value="page3.aspx"/>
  </setction>
  <setction title="Section 2" value="default.aspx">
        <subpage title ="Page 1" value="page1.aspx"/>
        <subpage title ="Page 2" value="page2.aspx"/>
        <subpage title ="Page 3" value="page3.aspx"/>
  </setction>
  <setction title="Section 2" value="default.aspx">
        <subpage title ="Page 1" value="page1.aspx"/>
        <subpage title ="Page 2" value="page2.aspx"/>
        <subpage title ="Page 3" value="page3.aspx"/>
  </setction>
</application>
```

Add a `Web.sitemap` file to your Web application and add the code, as shown in Listing 18.12, in the `Web.sitemap` file:

**Listing 18.12:** Code for the `Web.sitemap` File

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
  <siteMapNode url= "Default.aspx" title= "Section 1"  description="">
        <siteMapNode url="Page1.aspx" title="Page 1"  description="" />
        <siteMapNode url="Page2.aspx" title="Page 2"  description="" />
```

```
        <siteMapNode url="Page3.aspx" title="Page 3"  description="" />
    </siteMapNode>
</siteMap>
```

The output is displayed in Figure 18.18, where we have generated two menus from `XmlDataSource` and `SiteMapPathDataSource`, respectively:

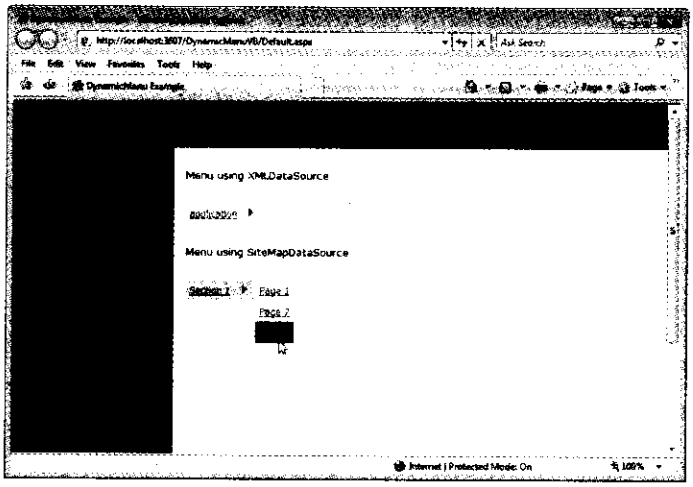



**Figure 18.18: DynamicMenu Example**

# Using the **SiteMapPath** Class

The `SiteMapPath` class displays a set of text or image hyperlinks that enable users to easily navigate a Web site.

Here is the inheritance hierarchy for the `SiteMapPath` class:

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

You can find the noteworthy properties of the `SiteMapPath` class in Table 18.10:

**Table 18.10: Noteworthy Properties of the SiteMapPath Class**

| | |
|---|---|
| `CurrentNodeStyle` | Obtains the style used for displaying text for the current node |
| `CurrentNodeTemplate` | Obtains or sets a control template to use for the node of a site navigation path that represents the currently displayed page |
| `NodeStyle` | Obtains the style used for the display text for all nodes in the site navigation path |
| `NodeTemplate` | Obtains or sets a control template to use for all functional nodes of a site navigation path |
| `ParentLevelsDisplayed` | Obtains or sets the number of levels of parent nodes the control displays, relative to the currently displayed node |
| `PathDirection` | Obtains or sets the order of the links displayed by the SiteMapPath control. Possible values are `RootToCurrent` (default) or `CurrentToRoot` |
| `PathSeparator` | Obtains or sets the string that delimits `SiteMapPath` nodes in the rendered navigation path |
| `PathSeparatorStyle` | Obtains the style used for the `PathSeparator` string |

| Table 18.10: Noteworthy Properties of the SiteMapPath Class | |
|---|---|
| PathSeparatorTemplate | Obtains or sets a control template to use for the path delimiter of a site navigation path |
| Provider | Obtains or sets a SiteMapProvider that is associated with the Web server control |
| RenderCurrentNodeAsLink | Indicates whether the site navigation node that represents the currently displayed page is rendered as a hyperlink |
| RootNodeStyle | Obtains the style for the root node display text |
| RootNodeTemplate | Obtains or sets a control template to use for the root node of a site navigation path |
| ShowToolTips | Obtains or sets a value indicating whether the SiteMapPath control writes an additional hyperlink attribute for hyperlinked navigation nodes. Depending on client support, when the mouse pointer is moved over a hyperlink that has the additional attribute set, a tool tip is displayed |
| SiteMapProvider | Obtains or sets the name of the SiteMapProvider used to render the site Navigation control |
| SkipLinkText | Obtains or sets a value that is used to render alternate text for screen readers to skip the control's content |

You can find the noteworthy methods of the SiteMapPath class in Table 18.11:

| Table 18.11: Noteworthy Methods of SiteMapPath Class | |
|---|---|
| DataBind | Binds a data source to the SiteMapPath control and all its child controls |

You can find the noteworthy events of the SiteMapPath class in Table 18.12:

| Table 18.12: Noteworthy Events of the SiteMapPath Class | |
|---|---|
| ItemCreated | Occurs when a SiteMapNodeItem is created by the SiteMapPath and is associated with its corresponding SiteMapNode. This event is raised by the OnItemCreated method |
| ItemDataBound | Occurs after a SiteMapNodeItem has been bound to its underlying SiteMapNode data by the SiteMapPath. This event is raised by the OnItemDataBound method |

# The SiteMapPath Control

The SiteMapPath control is also known as the breadcrumb Navigation control. Breadcrumb allows you to display the current page's context within the entire structure of a Web site. It also allows you to display links as a path back to the home page. It displays information regarding the user's current page along with the entire hierarchy of the pages, thereby enabling the user to navigate back to some other pages in the hierarchy.

However, the SiteMapPath control does not allow you to move forward from the current page to the next page deeper in the site hierarchy. The SiteMapPath control does not use the SiteMapDataSource control to display data. It relies on a SiteMapProvider to retrieve the data. SiteMapProvider contains an XML file Web.sitemap, which contains URLs and other information about the pages to be displayed. The SiteMapPath control can be localized by adding attributes to the Web.sitemap file. Figure 18.19 shows a SiteMapPath control on the Web form of the SiteMapPathControl example:

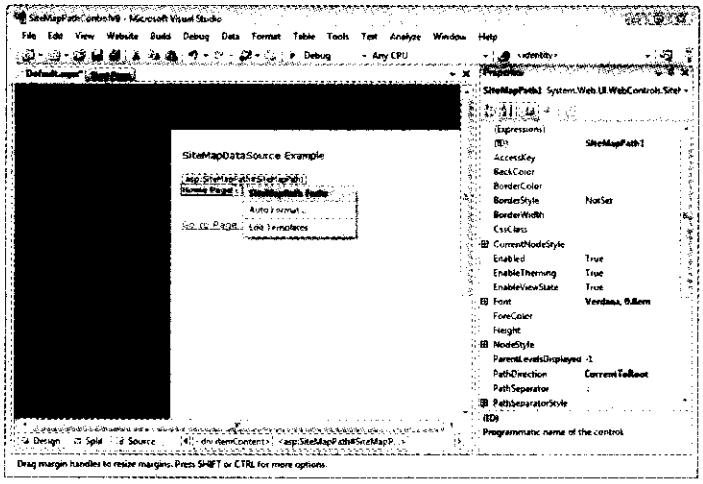



**Figure 18.19: SiteMapPath Control**

Earlier, to display the current page along with the entire hierarchy of the pages, you needed to create a manual code; which, in turn, used some external data source to display such breadcrumbs. However, this functionality is now built into .NET Framework. Therefore, all you need to do is use a one-line declaration in source code of the application using the `SiteMapPath` control.

## Data Retrieval Using the SiteMapPath Control

The `SiteMapPath` control uses `SiteMapProvider` to fetch the data and display it on the Web form. As you have seen earlier, both the `Menu` and `TreeView` controls rely on the `SiteMapDataSource` control The `SiteMapPath` control displays only the data whose URL is present in the specified `SiteMapProvider`. You should always check that there is no typing error in the `Web.sitemap` file, otherwise `SiteMapProvider` will not be able to display the desired output.

## SiteMapPath Behaviors

The `SiteMapPath` control displays various common properties to change the display behavior. Table 18.13 shows the noteworthy properties of the `SiteMapPath` class.

**Table 18.13: Noteworthy Properties of the SiteMapPath Class**

| Property | Description |
|---|---|
| `ParentLevelsDisplayed` | By default, this property is set to `-1`, which means that all parent levels in the XML file are displayed. You can set it to any integer value such as `1`, `2`, or `3` to indicate the N-level of the hierarchy along with the current page |
| `PathDirection` | Sets this property as `CurrentToRoot`. The default setting is such that it starts from the highest node in the current XML file and goes up to the current page. By setting this property, the links will be displayed in the opposite direction. This means starting from the current page and ending root page or first page |
| `PathSeparator` | Sets this property to : (Semicolon) as path separator. Its default value is > |
| `RenderCurrentNodeAsLink` | Sets this property to true, so that the current node text on the Web page will appear as hyperlink and not as plain text |
| `ShowToolTips` | Sets this property to false, so that description text from `SiteMapProvider` is not displayed automatically |

### SiteMapPath Appearance

You can set the appearance of a `SiteMapPath` control by using its style properties. You can also use skin and theme files for setting the appearance.

### SiteMapPath Style

You can set different style properties available with the `SiteMapPath` control. The following is the list of properties that can be used to set style for the SiteMapPath control:

- `NodeStyle`—It can be used to control the appearance of the text for all the nodes displayed by the `SiteMapPath` control. The default display text is that specified in the Title property of the SiteMapNode.
- `CurrentNodeStyle`—It sets the property of the node corresponding to the currently displayed Web page. The `CurrentNodeStyle` property often merges with the standard Web control style properties and the `NodeStyle` properties.
- `RootNodeStyle`—It is used to set the style for the absolute root in the hierarchy of `SiteMapProvider`. Depending on the levels of the hierarchy present on the current page, `RootNodeStyle` may not be displayed on every page of your Web site. It merges with the `NodeStyle` properties and the standard Web control style properties.
- `PathSeparatorStyle`—It is used to control the appearance of the path separator that appears between navigation path nodes. `PathSeparatorStyle` merges with the standard Web control style properties.

### SiteMapPath Templates

In addition to the style properties discussed above, the `SiteMapPath` control also offers templates, which can be used with a better flexibility to set the view of the control. Templates control the HTML rendering for the specific part of the control. Remember, if you define a template for a particular node, it will override the previous style setting defined for this node. To set the appearance of the `SiteMapPath` control, you can use the following templates:

- `NodeTemplate`—It can be applied on all the nodes in the hierarchy irrespective of their node type.
- `CurrentNodeTemplate`—It sets the appearance of the node corresponding to the current page displayed. `CurrentNodeTemplate` takes precedence over the `NodeTemplate`.
- `RootNodeTemplate`—It sets the appearance for the root node of the site hierarchy. `RootNodeTemplate` takes priority over `NodeTemplate` and `CurrentNodeTemplate`. This means that if the current page is the root page according to the site hierarchy, then only `RootNodeTemplate` will be used for the root node.
- `PathSeparatorTemplate`—It is used to separate the nodes with a separator between the nodes which are currently rendered on the Web page. It replaces the `PathSeparator` text value in your application.

## Creating SiteMapPath

As we have already discussed in this chapter, the `SiteMapPath` control can be automatically configured to the `SiteMapDataSource` control, which, in turn, can read the `Web.sitemap` file to display the data on the application. We have created an application named `SiteMapPathControl`. You can find this application as `SiteMapPathControlVB`. You can find the code of `SiteMapPathControlVB` application in the `Code\ASP.NET\Chapter 18\SiteMapPathControlVB` folder on the CD.

To create `Web.sitemap`, right-click the Project name. A context menu appears. Select the Add New Item option from the context menu. The Add New Item dialog box appears, as shown in Figure 18.10. Select Site Map from the options available. This will create a new `Web.sitemap` file in the project.

The `Web.sitemap` file consists of navigational details for an application. You can set the title, URL, and other information for each page by using the `Web.sitemap` file. Add the code, as shown in Listing 18.13, in the `Web.sitemap` file:

**Listing 18.13:** Code for the `Web.sitemap` File

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
    <siteMapNode url="default.aspx" title="Home Page"  description="">
        <siteMapNode url="page1.aspx" title="Section 1"  description="" />
        <siteMapNode url="page2.aspx" title="Section 2"  description="" />
    </siteMapNode>
</siteMap>
```

You also need to add the code, as shown in Listing 18.14, in the `Default.aspx` page of this application:

**Listing 18.14:** Code for the `Default.aspx` Page

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb"
    Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>SiteMapPath Example</title>
    <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="mainForm" runat="server">
    <div>
        <div id="header">
        </div>
        <div id="sidebar">
        <div id="nav">
             
        </div>
        </div>
        <div id="content">
                <div class="itemContent">
                <br />
                <asp:Label ID="Label1" runat="server" Text="SiteMapDataSource
                 Example"></asp:Label><br />
                <br />
                <asp:SiteMapPath ID="SiteMapPath1" runat="server" Font-Names="Verdana"
                 Font-Size="0.8em"
                 PathSeparator=" : " PathDirection="CurrentToRoot">
                <PathSeparatorStyle Font-Bold="True" ForeColor="#5D7B9D" />
                <CurrentNodeStyle ForeColor="#333333" />
                <NodeStyle Font-Bold="True" ForeColor="#7C6F57" />
                <RootNodeStyle Font-Bold="True" ForeColor="#5D7B9D" />
                </asp:SiteMapPath>
                <br />
                <br />
                <asp:HyperLink ID="HyperLink1" runat="server"
                 NavigateUrl="~/Page1.aspx">Go to Page 1</asp:HyperLink>
                <asp:HyperLink ID="HyperLink2" runat="server"
                 NavigateUrl="~/Page2.aspx">Go to Page 2</asp:HyperLink><br />
        </div>
        <div id="footer">
                <p class="left">
                    All content copyright &copy; Kogent Solutions Inc.</p>
        </div>
    </div>
    </form>
</body>
</html>
```

Now, add two Web forms in the project, named `Page1.aspx` and `Page2.aspx` and add the code, as shown in Listing 18.15, in the `Page1.aspx` page of the `SiteMapPathControl` application:

**Listing 18.15:** Code for the `Page1.aspx` Page

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Page1.aspx.vb" Inherits="Page1" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head id="Head1" runat="server">
  <title>SiteMapPath Example</title>
  <link href="Stylesheet.css" rel="stylesheet" type="text/css" />

</head>

<body>
  <form id="mainForm" runat="server">
  <div>
        <div id="header">
        </div>
        <div id="sidebar">
        <div id="nav">
               
        </div>
        </div>
        <div id="content">
        <div class="itemContent">
              <br />
              <asp:Label ID="Label1" runat="server" Text="SiteMapDataSource
               Example"></asp:Label><br />
              <br />
              <asp:SiteMapPath ID="SiteMapPath1" runat="server" Font-Names="Verdana"
               Font-Size="0.8em"
               PathSeparator=" : ">
              <PathSeparatorStyle Font-Bold="True" ForeColor="#5D7B9D" />
              <CurrentNodeStyle ForeColor="#333333" />
              <NodeStyle Font-Bold="True" ForeColor="#7C6F57" />
              <RootNodeStyle Font-Bold="True" ForeColor="#5D7B9D" />
              </asp:SiteMapPath>
               <br />
              <br />
              <asp:HyperLink ID="HyperLink2" runat="server"
               NavigateUrl="~/Page2.aspx">Go
               to Page 2</asp:HyperLink><br />
        </div>
        <div id="footer">
              <p class="left">
                    All content copyright &copy; Kogent Solutions Inc.</p>
        </div>
        </div>
  </div>
  </form>

</body>

</html>
```

Now, add the code, as shown in Listing 18.16, in the `Page2.aspx` page of the `SiteMapPathControl` application:

**Listing 18.16:** Code for the `Page2.aspx` Page

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Page2.aspx.vb" Inherits="Page2" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head id="Head1" runat="server">
    <title>SiteMapPath Example</title>
    <link href="StyleSheet.css" rel="stylesheet" type="text/css" />

</head>

<body>
    <form id="mainForm" runat="server">
    <div>
        <div id="header">
        </div>
        <div id="sidebar">
        <div id="nav">
                 
        </div>
        </div>
        <div id="content">
        <div class="itemContent">
                <br />
                <asp:Label ID="Label1" runat="server" Text="SiteMapDataSource
                Example"></asp:Label><br />
                <br />
                <asp:SiteMapPath ID="SiteMapPath1" runat="server" Font-Names="Verdana"
                Font-Size="0.8em"
                PathSeparator=" : ">
                <PathSeparatorStyle Font-Bold="True" ForeColor="#5D7B9D" />
                <CurrentNodeStyle ForeColor="#333333" />
                <NodeStyle Font-Bold="True" ForeColor="#7C6F57" />
                <RootNodeStyle Font-Bold="True" ForeColor="#5D7B9D" />
                </asp:SiteMapPath>
                <br />
                <br />
                <asp:HyperLink ID="HyperLink1" runat="server"
                 NavigateUrl="~/Page1.aspx">Go
                 to Page 1</asp:HyperLink> <br />
                </div>
                <div id="footer">
                <p class="left">
                        All content copyright &copy; Kogent Solutions Inc.</p>
        </div>
        </div>
    </div>
    </form>
</body>

</html>
```

Now, execute the fully functional `SiteMapPath` control. The output is shown in Figure 18.20:

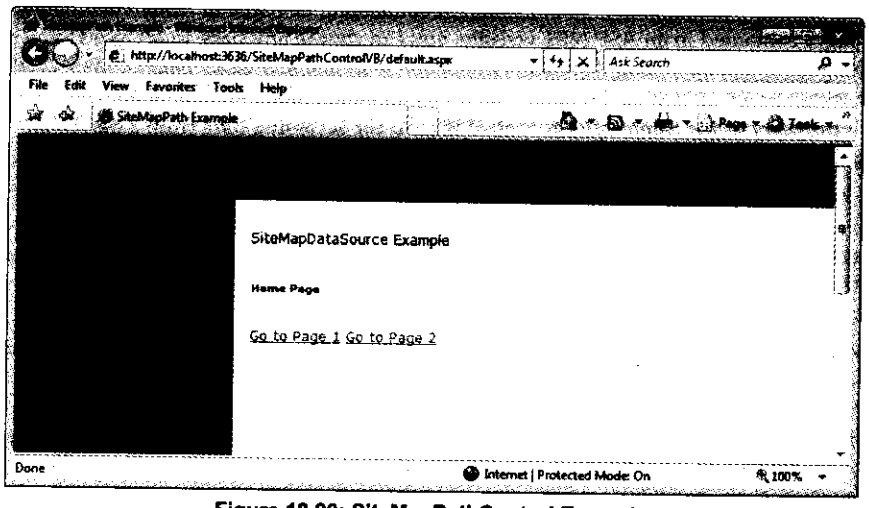



**Figure 18.20: SiteMapPathControl Example**

# Summary

In this chapter, we have described three navigation controls-- the `TreeView` control, `Menu` control, and `SiteMapPath` control of ASP.NET 3.5. We have also discussed about the base classes of the navigation controls along with its properties, methods, and events. In addition, the chapter discussed the implementation of these navigation controls.

In the next chapter, we will discuss validation controls available in ASP.NET 3.5.

# Quick Revies

**Q1. What is the use of navigation controls?**

Ans: Navigation controls are used in a Web application to provide users with the flexibility of moving back and forth between different Web pages of a website. These controls display the address of a Web page in the form of Menus and Hyperlinks, which users use to navigate to any Web page of a Web application from the current Web page.

**Q2. List some important features of navigation controls?**

Ans: Some of the important features of navigation controls are as follows:

- The navigation controls represent the structure of a Web application, that is, how different Web pages of a Web application are linked together.
- The navigation controls dynamically order the Web pages when a user traverses through different Web pages.
- The navigation controls recognize the user's current position in the website and shift the reference pointer to other pages accordingly when the user moves from a Web page to another.

**Q3. List all the navigation controls?**

Ans: Following is a list of the navigation controls:

- SiteMapPath control
- TreeView control
- Menu control

**Q4. What is the SiteMapPath control?**

Ans: The SiteMapPath control is a navigation control. This control displays the Map of the site related to its Web pages. This map includes the pages in the particular website and displays the name of those pages. You can click on that particular page in the Sitemap to navigate to that page. In other words, we can say

that the SiteMapPath control displays links for connecting to URLs of other Web pages. Out of all the Web pages contained in an ASP.NET website, you can choose any page to act as a Home page. This page is known as root node. A user can access the Home page and any other page of the website with the help of the SiteMapPath control.

**Q5. Which property of the SiteMapPath control is used for accessing data from a database?**

Ans: The SiteMapPath control uses a property called SiteMapProvider, for accessing data from a database.

**Q6. What is the TreeView control?**

Ans: The TreeView control is a navigation control, which is used in Web applications to display the location of different Web pages as a tree structure. The TreeView control is used in Web pages to enable the user to view the structure of the Web application by traversing its nodes.

**Q7. What is the Menu control? What are the views of Menu control to display the data?**

Ans: The Menu control is a Web Server control for representing data items. These data items can originate from a database, an XML file, or a sitemap file. The Menu control of ASP.NET has two views for displaying data:

- **Horizontal Orientation**—In this view, all the items displayed by the Menu control are aligned horizontally.
- **Vertical Orientation**—In this view, all the items displayed by the Menu control are aligned vertically.

**Q8. What are the categories of the Menu control?**

Ans: Based on the type of data they present, the two categories of menu are:

- **Static Menu**—The items of these menus are always displayed on the Web pages.
- **Dynamic Menu**—This is created when the user of a Web page accesses the main menu item; the dynamic menu items are presented as a submenu.

**Q9. List the various types of nodes of the TreeView control?**

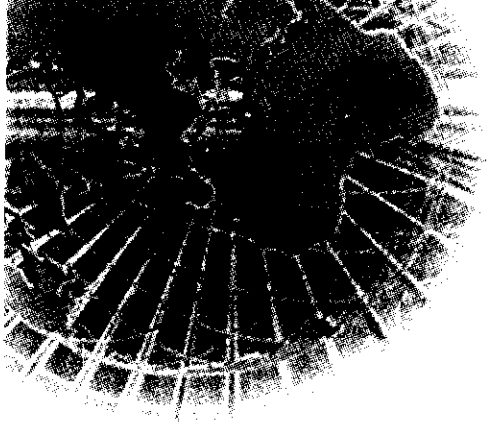
Ans: The TreeView control has four types of nodes:

- **Root**—It is a top-level node in the tree.
- **Parent**—These nodes are represented as parent nodes, which have child nodes shown as branches of the parent nodes.
- **Child**—These nodes are contained by its parent nodes.
- **Leaf**—It does not contain any child nodes.

**Q10. What are the node style properties of the SiteMapPath control?**

Ans: The node style properties of the SiteMapPath control are as follows:

- **CurrentNodeStyle**—Obtains the style used for the display text for the current node
- **RootNodeStyle**—Obtains the style for the root node display text
- **NodeStyle**—Obtains the style used for the display text for all nodes in the site navigation path



# 19

# Validation Controls

# Introduction

`Validation controls` are controls used for validating the data entered in an input control, such as the textbox of a Web page.

When a user enters invalid data in an associated control, the validation control displays an error message on the screen. The error message is defined as a property value of the validation control. The data being entered is validated each time it is entered by the user, and the error message disappears only when the data is valid. Validation controls help save time and enhance the efficiency of the applications by validating the data before a request is sent to the server. The following Validation controls are discussed in this chapter:

- The `RequiredFieldValidator` control
- The `RangeValidator` control
- The `RegularExpressionValidator` control
- The `CompareValidator` control
- The `CustomValidator` control
- The `ValidationSummary` control

Let's explore these controls in detail, starting with the `BaseValidator` class, which is the base of all validation controls.

# The BaseValidator Class

The `System.Web.UI.WebControls.BaseValidator` class provides basic implementation required for all validation controls. Here is the inheritance hierarchy of this class:

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

Noteworthy public properties of the `BaseValidator` class are given in Table 19.1:

**Table 19.1: Noteworthy Public Properties of the BaseValidator Class**

| | |
|---|---|
| `ControlToValidate` | Handles an input control, such as the `TextBox` control, which needs to be validated |
| `Display` | Handles the behavior of the error message in a validation control |
| `EnableClientScript` | Handles a value indicating whether or not client-side validation is enabled |
| `Enabled` | Handles a value that indicates whether or not the validation control is enabled or not |
| `ErrorMessage` | Handles the text for the error message displayed in a `ValidationSummary` control when validation fails |
| `ForeColor` | Handles the color of the message displayed when validation fails |
| `IsValid` | Handles a value that indicates if the associated input control passes validation or not |
| `SetFocusOnError` | Handles a value that indicates if the focus is set on the control or not, specified by the `ControlToValidate` property when validation fails |
| `Text` | Handles the text displayed in the validation control when the validation fails |
| `ValidationGroup` | Handles the name of the validation group to which this validation control belongs |

**NOTE**

*The ValidationGroup property, when set, validates only the validation controls within the specified group when the control is posted back to the server*

Noteworthy public methods of the `BaseValidator` class are listed in Table 19.2:

| Table 19.2: Noteworthy Public Methods of the BaseValidator Class | |
|---|---|
| `Validate` | Helps in performing validation on the associated input control and updates the `IsValid` property |
| `GetValidationProperty` | Helps in determining the validation property of a control, if it exists |

Now, let's discuss the Validation controls one by one in detail.

# The **RequiredFieldValidator** Control

The `RequiredFieldValidator` control is the simplest control, and is used to ensure that the user has entered the data into the input control, such as `TextBox` to which it is bound. For example, if you are entering data, for buying a T-shirt, in an input control, it is necessary to enter the size of the T-shirt that you want to buy. If you do not enter a value, the Validation control displays an error message. Here is the inheritance hierarchy for the `RequiredFieldValidator` class:

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

This class has no non-inherited methods or events. This class inherits the properties and methods of the `BaseValidator` class that are listed in Tables 19.1 and 19.2. Noteworthy public property of the `RequiredFieldValidator` class is listed in Table 19.3:

| Table 19.3: Noteworthy Public property of the RequiredFieldValidator Class | |
|---|---|
| `InitialValue` | Handles the initial value of the associated Input control |

# Using the RequiredFieldValidator Control

To add a `RequiredFieldValidator` control on a Web page (Default.aspx), add the following code to the source code of the Web page:

```
<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
  ErrorMessage="RequiredFieldValidator"></asp:RequiredFieldValidator>
```

Now, let's create an application named RequiredFieldValidatorVBExample. You can find the code of RequiredFieldValidatorVBExample application in the Code\ASP.NET\Chapter 19\ RequiredFieldValidatorVBExample folder on the CD. You can add a `RequiredFieldValidator` control to a Web page either by dragging and dropping it from the Standard tab of the Toolbox or by double-clicking this control in the Toolbox.

In this example, we are adding two `RequiredFieldValidator` controls on a Web page for validating two `TextBox` controls. The `RequiredFieldValidator` controls ensure that the two `TextBox` controls are not kept empty by the user at runtime. Listing 19.1 shows the Default.aspx page for the `RequiredFieldValidator` control:

**Listing 19.1:** Showing the Code for the `Default.aspx` Page

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

```
    <title>RequiredFieldValidator Control Example</title>
  <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <form id="Form" runat="server">
    <div id="header">

    </div>
    <div id="sidebar">
          <div id="nav">
                 
          </div>
    </div>
    <div id="content">
          <div class ="itemContent">
    </div>
    <div>
          <asp:Label ID="Label1" runat="server"
          Text="RequiredField Validator Control Example" Font-Bold="True"
          Font-Size="Medium" Font-Underline="True"></asp:Label>
          <br />
          <br />
          <asp:Label ID="Label2" runat="server" Font-Bold="True" Font-Size="Small"
          Text="User name:"></asp:Label>
            
          <asp:TextBox ID="TextBox1" runat="server" Height="22px"
          Width="175px"></asp:TextBox>
          <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
          ControlToValidate="TextBox1" ErrorMessage="User name can not be
          empty"></asp:RequiredFieldValidator>
          <br/>
          <br/>
          <asp:Label ID="Label3" runat="server" Font-Bold="True" Font-Size="Small"
          Text="Password:"></asp:Label>
              
          <asp:TextBox ID="TextBox2" runat="server" Height="22px" TextMode="Password"
          Width="175px"></asp:TextBox>
          <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server"
          ControlToValidate="TextBox2" ErrorMessage="Password can not be
          empty"></asp:RequiredFieldValidator>
          <br />
          <br />
          <br />
                       
                       
              
          <asp:Button ID="Button1" runat="server" BackColor="Black" Font-Bold="True"
          Font-Size="Medium" ForeColor="White" Text="Submit" />
          <div id="footer">
                <p class="left">
                      All content copyright &copy; Kogent Solutions Inc.</p>
          </div>
    </div>
    </div>
    </form>
  </body>
  </html>
```

If you run the application and click the Submit button without entering data in the two `TextBox` controls, both the `RequiredFieldValidator` controls display error messages, as shown in Figure 19.1:

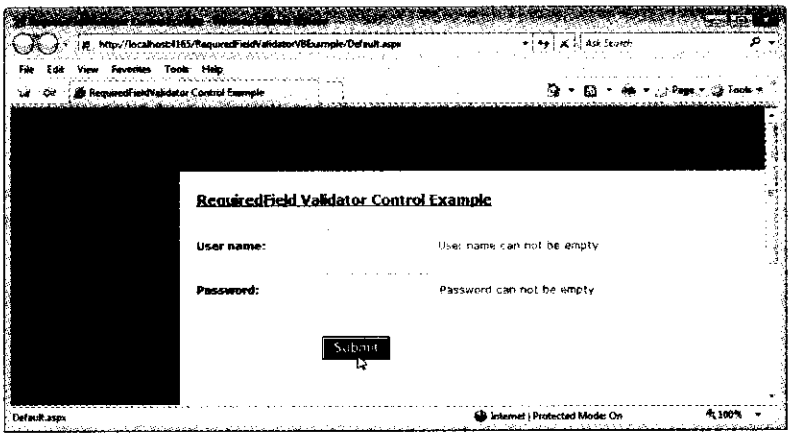



**Figure 19.1: Two RequiredFieldValidator Controls on a Web page Showing Error Messages**

# The **RangeValidator** Control

The `RangeValidator` control checks whether or not the value of an input control is inside a specified range of values. It has the following four three properties:

- `ControlToValidate` **property**—Contains the Input control to validate
- `MinimumValue` **property**—Holds the minimum value of the valid range
- `MaximumValue` **property**—Holds the maximum value of the valid range

If one of these properties is set, then the other property must also be set.

Do not forget to set the `Type` property to the data type of the values. The following data types can be used for the values:

- **String**—A string data type
- **Integer**—An integer data type
- **Double**—A double data type
- **Date**—A date data type
- **Currency**—A currency data type

Here is the inheritance hierarchy for the `RequiredFieldValidator` class:

```
System.Object
  System.Web.UI.Control
    System.Web.UI.WebControls.WebControl
      System.Web.UI.WebControls.Label
        System.Web.UI.WebControls.BaseValidator
          System.Web.UI.WebControls.BaseCompareValidator
            System.Web.UI.WebControls.RangeValidator
```

Noteworthy public properties of the `RangeValidator` class are listed in Table 19.4:

**Table 19.4: Noteworthy Public Properties of the RangeValidator Class**

| | |
|---|---|
| `MaximumValue` | Obtains or sets the maximum value of the validation range for the `RangeValidator` control. |
| `MinimumValue` | Obtains or sets the minimum value of the validation range for the `RangeValidator` control. |

# Using the RangeValidator Control

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

```
<asp:RangeValidator ID="RangeValidator1" runat="server"
    ErrorMessage="RangeValidator"></asp:RangeValidator>
```

Now, let's create an application named RangeValidatorVBExample. You can find the code of RangeValidatorVBExample application in the Code\ASP.NET\Chapter 19\ RangeValidatorVBExample folder on the CD. In this example, we are adding two `RangeValidator` controls on a Web page for validating two `TextBox` controls. The first `RangeValidator` control ensures that the age entered in the first `TextBox` control is in the range 20-40 yrs. The second `RangeValidator` control ensures that the date of birth is in the range 1988/1/1 to 2028/1/1. Listing 19.2 shows the Default.aspx page for the `RangeValidator` control:

**Listing 19.2:** Showing the Code for the `Default.aspx` Page

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

  </div>
  <div id="sidebar">
        <div id="nav">
               
        </div>
  </div>
  <div id="content">
        <div class ="itemContent">
        </div>
        <div>
        <asp:Label ID="Label1" runat="server" Text="RangeValidator Control Example"
        Font-Bold="True" Font-Size="Medium" Font-Underline="True"></asp:Label>
        <br />
        <br />
        <br />
        <asp:Label ID="Label2" runat="server" Font-Bold="True" Font-Size="Small"
        Text="Your Age:"></asp:Label>
          
        <asp:TextBox ID="TextBox1" runat="server" width="172px"></asp:TextBox>
        <asp:RangeValidator ID="RangeValidator1" runat="server"
        ControlToValidate="TextBox1"
        ErrorMessage="Invalid Age. Please enter the age between 20 to 40."
        MaximumValue="40" MinimumValue="20" Type="Integer"></asp:RangeValidator>
         <br />
        <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
        ControlToValidate="TextBox1" ErrorMessage="This field can not be
        blank."></asp:RequiredFieldValidator>
        <br />
        <br />
        <asp:Label ID="Label3" runat="server" Font-Bold="True" Font-Size="Small"
        Text="Date of Birth (YYYY/MM/DD format) :"></asp:Label>
           
        <asp:TextBox ID="TextBox2" runat="server" width="172px"></asp:TextBox>
        <asp:RangeValidator ID="RangeValidator2" runat="server"
        ControlToValidate="TextBox2"
        ErrorMessage="Invalid DOB. It must be between 1988/1/1 to 2028/1/1."
        MaximumValue="2028/1/1" MinimumValue="1988/1/1" Type="Date"></asp:RangeValidator>
        <br />
        <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server"
        ControlToValidate="TextBox2" ErrorMessage="This field can not be
        blank."></asp:RequiredFieldValidator>
```

```
            <br />
            <br />
                          
                          
                          
            <asp:Button ID="Button1" runat="server" BackColor="Black" Font-Bold="True"
            Font-Size="Small" ForeColor="White" Text="Submit" />
            <div id="footer">
                <p class="left">
                    All content copyright &copy; Kogent Solutions Inc.</p>
                </div>
            </div>
        </div>
    </form>
</body>
</html>
```

Now, enter values outside the specified range in the `TextBox` controls and click the Submit button. The `RangeValidator` controls display an error message, as shown in Figure 19.2:

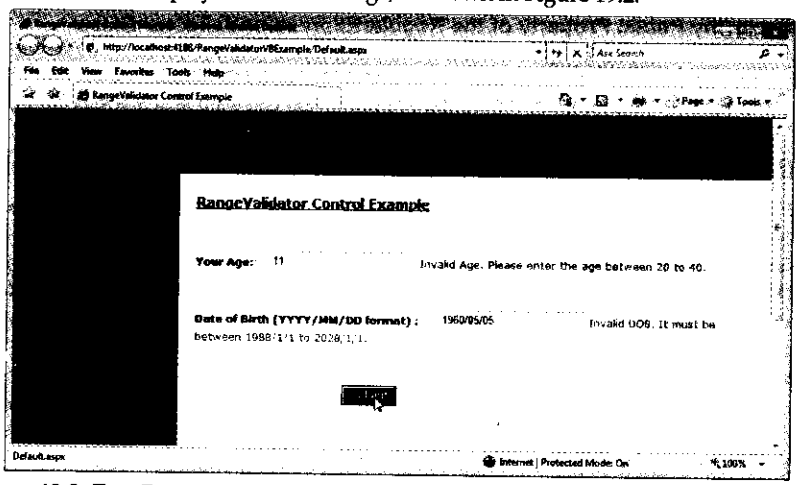



**Figure 19.2: Two RangeValidator Controls on a Web page Showing Error Messages**

**NOTE**

*It is mandatory to use a RequiredFieldValidator control with the RangeValidator, RegularExpressionValidator, CompareValidator, and CustomValidator controls because the latter do not work on blank Input controls.*

# The **RegularExpressionValidator** Control

Regular expressions are used to check whether or not the text matches a certain pattern. The `RegularExpressionValidator` control validates whether the value in the Input control (`text box`) matches the format specified by the regular expression, such as a US telephone number or a date format. The `RegularExpressionValidator` control exists within the `System.Web.UI.WebControls` namespace.

In general, regular expressions are made up of text with an embedded code listing that starts with a backslash (`\`) as well as other control code listings. The following is an example:

```
\b[A-Za-z]+\b
```

The code for a word boundary (where a word ends or starts) is `\b` and a `character` class is a set of characters surrounded by `[ and ]`. This allows you to specify what characters you want to accept. So, this regular expression matches a word consisting of uppercase or lowercase letters only. The + sign stands for one or more of, so we matching one or more uppercase and/or lowercase letters here.

For example, the regular expression to determine if the text matches a valid `e-mail address` is:

```
\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*
```

Here, \w stands for a word character (such as letters, underscores, and the rest) and * means zero or more of.

ASP.NET 3.5 includes some pre-built regular expressions that you can use to match well-known sequences of characters, such as, a French phone number, P.R.C. social security number, US social security number, e-mail address, Internet URL address, and the rest. Regular expressions are complex to understand as they require in-depth knowledge of the character relations required during custom expression definition.

Here is the inheritance hierarchy for the `RegularExpressionValidator` class:

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

This class has no non-inherited methods or events. Noteworthy public property of the `RegularExpressionValidator` class is listed in Table 19.5:

**Table 19.5: Noteworthy Public Property of the RegularExpressionValidator Class**

| Property | Description |
|---|---|
| `ValidationExpression` | Obtains or sets the regular expression that you want to match data against for validation |

## Using the RegularExpressionValidator Control

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

```
<asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server"
  ErrorMessage="RegularExpressionValidator"></asp:RegularExpressionValidator>
```

Now, let's create an application named RegularExpressionValidatorVBExample. You can find the code of RegularExpressionValidatorVBExample application in the Code\ASP.NET\Chapter 19\RegularExpressionValidatorVBExample folder on the CD. In this example, we are taking two `RegularExpressionValidator` controls on a Web page for validating two `TextBox` controls. It ensures that the data (URL and email address) entered in both the `TextBox` controls is valid. Listing 19.3 shows the Default.aspx page for the RegularExpressionValidator control:

**Listing 19.3:** Showing the Code for the `Default.aspx` Page

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb"
  Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>RegularExpressionValidator Control Example</title>
<link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <form id="form" runat="server">
  <div id="header">
  </div>
  <div id="sidebar">
      <div id="nav">
           
      </div>
  </div>
  <div id="content">
      <div class ="itemContent">
      </div>
      <asp:Label ID="Label1" runat="server" Font-Bold="True" Font-Size="Medium"
      Font-Underline="True" Text="RegularExpressionValidator Control
      Example"></asp:Label>
      <br />
```

```
            <br />
            <br />
            <asp:Label ID="Label2" runat="server" Font-Bold="True" Font-Size="Small"
            Text="Enter URL:"></asp:Label>
                        
                 
            <asp:TextBox ID="TextBox1" runat="server" Width="177px"></asp:TextBox>
            <asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server"
            ControlToValidate="TextBox1" ErrorMessage="Not a Valid URL."
            ValidationExpression="http(s)?://([\w-]+\.)+[\w-]+(/[\w-
            ./?%&amp;=]*)?"></asp:RegularExpressionValidator>
            <br />
            <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
            ControlToValidate="TextBox1" ErrorMessage="This field can not be
            blank."></asp:RequiredFieldValidator>
            <br />
            <br />
            <asp:Label ID="Label3" runat="server" Font-Bold="True" Font-Size="Small"
            Text="Enter E-mail address:"></asp:Label>
              
            <asp:TextBox ID="TextBox2" runat="server" Width="177px"></asp:TextBox>
            <asp:RegularExpressionValidator ID="RegularExpressionValidator2" runat="server"
            ControlToValidate="TextBox2" ErrorMessage="Not a Valid E-mail address."
            ValidationExpression="\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-
            .]\w+)*"></asp:RegularExpressionValidator>
            <br />
            <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server"
            ControlToValidate="TextBox2" ErrorMessage="This field can not be
            blank."></asp:RequiredFieldValidator>
            <br />
            <br />
            <br />
                        
                        
                     
            <asp:Button ID="Button1" runat="server" BackColor="Black" Font-Bold="True"
            Font-Size="Small" ForeColor="White" Text="Submit" Width="60px" />
            <br />
            </div>
            <div id="footer">
                  <p class="left">
                        All content copyright &copy; Kogent Solutions Inc.</p>
            </div>
      </div>
      </form>
</body>
</html>
```

Now, enter the values in the `TextBox` controls and click the Submit button. The `RegularExpressionValidator` controls display an error message, as shown in Figure 19.3:

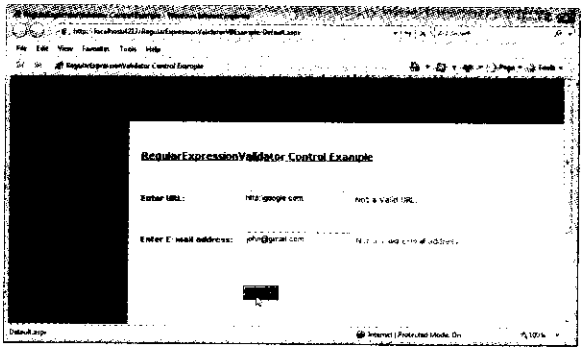



**Figure 19.3: RegularExpressionValidator Controls Displaying Error Message on a Web page**

# The CompareValidator Control

The CompareValidator control is used to compare the value entered by a user into one Input control with the value entered into another Input control or with an already existing value. The `CompareValidator` control exists within the `System.Web.UI.WebControls` namespace. Table 19.6 lists the various `Operator` property types, specifying the different types of comparisons:

**Table 19.6: Operator Properties of the CompareValidator Control**

| | |
|---|---|
| `Equal` | Checks whether the compared values are equal |
| `NotEqual` | Checks that controls are not equal to each other |
| `GreaterThan` | Checks for a greater than relationship |
| `GreaterThanEqual` | Checks for a greater than or equal to relationship |
| `LessThan` | Checks for a less than relationship |
| `LessThanEqual` | Checks for a less than or equal to relationship |
| `DataTypeCheck` | Compares data types between the value entered into the data-entry control being validated and the data type specified by the Type property |

The `Type` property is used to specify the data type of both the comparison values, where `String` is the default type. Both values are automatically converted to the String data type before the comparison operation is performed. The different data types that can be use are as follows:

- **String**—A string data type
- **Integer**—An integer data type
- **Double**—A double data type
- **Date**—A date data type
- **Currency**—A currency data type

Noteworthy public properties of the `CompareValidator` class are given in Table 19.7:

**Table 19.7: Noteworthy Public Properties of the CompareValidator Class**

| | |
|---|---|
| `ControlToCompare` | Obtains or sets the data entry control, which has to be compared with the data-entry control being validated. |
| `Operator` | Obtains or sets the comparison operation to perform. |
| `ValueToCompare` | Obtains or sets a constant value to compare with the value entered by a user in the data entry control being validated. |

# Using the CompareValidator Control

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

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

Now, let's create an application named CompareValidatorVBExample. You can find the code of CompareValidatorVBExample application in the Code\ASP.NET\Chapter 19\ CompareValidatorVBExample folder on the CD. In this example, we take a `CompareValidator` control on a Web page, which validates the third `TextBox` control. It ensures that the data (password) entered in the third `TextBox` control is exactly the same as the data (`password`) entered in the second `TextBox` control. Listing 19.4 shows the Default.aspx page for the CompareValidator control:

**Listing 19.4:** Showing the Code for the `Default.aspx` Page

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

  </div>
  <div id="sidebar">
        <div id="nav">
               
        </div>
  </div>
  <div id="content">
        <div class ="itemContent">
        </div>
        <div>
        <asp:Label ID="Label1" runat="server" Font-Bold="True" Font-Size="Medium"
        Font-Underline="True" Text="CompareValidator Control Example"></asp:Label>
        <br />
        <br />
        <br />
        <asp:Label ID="Label2" runat="server" Font-Bold="True" Font-Size="Small"
        Text="User Name:"></asp:Label>
                  
        <asp:TextBox ID="TextBox1" runat="server" width="173px"></asp:TextBox>
        <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
        ControlToValidate="TextBox1" ErrorMessage="This Field can not be
        blank."></asp:RequiredFieldValidator>
        <br />
        <br />
        <asp:Label ID="Label3" runat="server" Font-Bold="True" Font-Size="Small"
        Text="Password:"></asp:Label>
                    
        <asp:TextBox ID="TextBox2" runat="server" TextMode="Password"
        Width="173px"></asp:TextBox>
        <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server"
        ControlToValidate="TextBox2" ErrorMessage="This Field can not be
        blank."></asp:RequiredFieldValidator>
        <br />
        <br />
        <asp:Label ID="Label4" runat="server" Font-Bold="True" Font-Size="Small"
        Text="Re-type Password:"></asp:Label>
         <asp:TextBox ID="TextBox3" runat="server" TextMode="Password"
        Width="173px"></asp:TextBox>
        <asp:CompareValidator ID="CompareValidator1" runat="server"
        ControlToCompare="TextBox2" ControlToValidate="TextBox3"
        ErrorMessage="Password does not match."></asp:CompareValidator>
        <br />
        <br />
        <asp:RequiredFieldValidator ID="RequiredFieldValidator3" runat="server"
        ControlToValidate="TextBox3" ErrorMessage="This Field can not be
```

```
            blank."></asp:RequiredFieldValidator>
            <br />
            <br />
                       
                        
                      
            <asp:Button ID="Button1" runat="server" BackColor="Black" Font-Bold="True"
            Font-Size="Small" ForeColor="White" Text="Submit" />
            <br />
            <br />
            </div>
            <div id="footer">
                    <p class="left">
                    All content copyright &copy; Kogent Solutions Inc.</p>
            </div>
    </form>
</body>
</html>
```

Now, run the application, and enter John in the first `TextBox` control, abc@123 in the second `TextBox` control, abc@123 in the third `TextBox` control, and then click the Submit button. The `CompareValidator` control displays an error message, as shown in Figure 19.4:

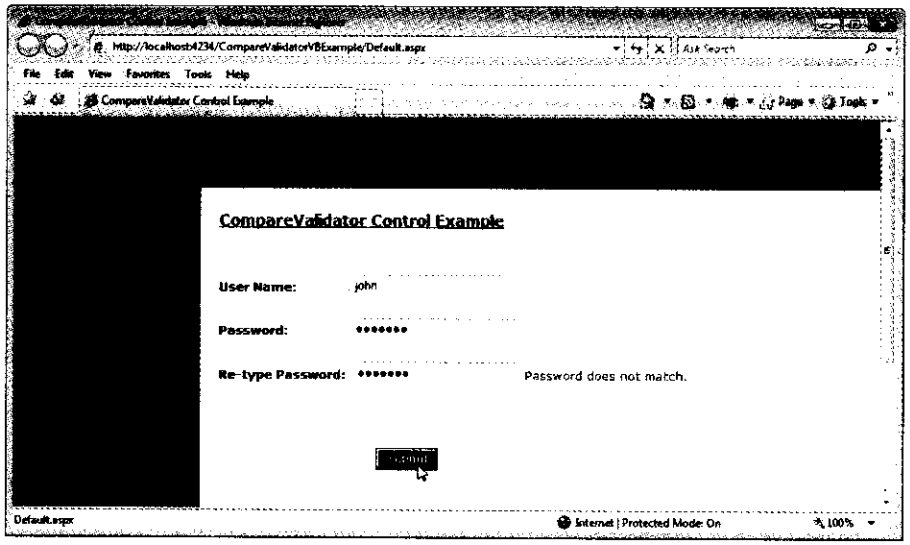



**Figure 19.4: CompareValidator Control Displaying Error Message on a Web page**

# The **CustomValidator** Control

The `CustomValidator` control is used to customize and implement data validation as per your requirement. Occasionally, you might need to validate your data in a way that is not possible directly with the help of existing Validation controls. For example, if you want to find out whether a number is odd or even, you cannot use an existing Validation control for doing that because this functionality is not included in any of the `Validation` controls. To solve this problem, you need a Validation control that can be customized to solve this problem. The `CustomValidator` control exists within the `System.Web.UI.WebControls` namespace. Here is the inheritance hierarchy for the `CustomValidator` class:

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

```
System.Web.UI.WebControls.BaseValidator
    System.Web.UI.WebControls.CustomValidator
```

The `CustomValidator` control checks whether or not the input you have given, such as prime, even, or odd number, matches a given condition. Noteworthy public properties of the `CustomValidator` class are listed in Table 19.8:

**Table 19.8: Noteworthy Public Properties of the CustomValidator Class**

| Property | Description |
|---|---|
| `ClientValidationFunction` | Obtains or sets the name of the custom client-side script function used for validation |
| `ValidateEmptyText` | Obtains or sets a `Boolean` value indicating whether empty text should be validated or not |

For using the `CustomValidator` control in a Web application, first create a function using a scripting language, such as `JavaScript` or `VBScript` (both are supported by Internet Explorer), in the source file of the Web page. You can then set the `ClientValidationFunction` property of the `CustomValidator` control to the name of the function you have created. The `CustomValidator` control validates the data entered by the user in the data entry control at the runtime on the basis of the code written inside this function.

A noteworthy public event of the `CustomValidator` class is listed in Table 19.9:

**Table 19.9: Noteworthy Public Event of the CustomValidator Class**

| Event | Description |
|---|---|
| `ServerValidate` | Occurs when validation takes place on the server |

# Using the CustomValidator Control

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

```
<asp:CustomValidator ID="CustomValidator1" runat="server"
  ErrorMessage="CustomValidator"></asp:CustomValidator>
```

Now, let's create an application named CustomValidatorVBExample. You can find the code of CustomValidatorVBExample application in the Code\ASP.NET\Chapter 19\ CustomValidatorVBExample folder on the CD. In this example, we are taking a `CustomValidator` control on a Web page, which validates the `TextBox` control and ensures that the user enters the correct data (username). Listing 19.5 shows the Default.aspx page for the CustomValidator control:

**Listing 19.5:** Showing the Code for the `Default.aspx` Page

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

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

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">
  <title>CustomValidator Control Example</title>
<link href="StyleSheet.css" rel="stylesheet" type="text/css" />

</head>

<body>
  <form id="Form" runat="server">
      <script language="vbscript" type="text/vbscript">
      Sub Validate (source, arguments)
```

```
            If (arguments.Value = "testuser")Then
                    arguments.IsValid = True
            Else
                    arguments.IsValid = False
            End IF
        End Sub
        </script>
        <div id="header">
        </div>
        <div id="sidebar">
        <div id="nav">
             
        </div>
        </div>
        <div id="content">
        <div class ="itemContent">
        </div>
        <div>
        <asp:Label ID="Label1" runat="server" Font-Bold="True" Font-Size="Medium"
        Font-Underline="True" Text="CustomValidator Control Example"></asp:Label>
        <br />
        <br />
        <br />
        <asp:Label ID="Label2" runat="server" Font-Bold="True" Font-Size="Small"
        Text="Enter User Name:"></asp:Label>
          
        <asp:TextBox ID="TextBox1" runat="server" Width="179px"></asp:TextBox>
        <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
        ControlToValidate="TextBox1" ErrorMessage="This field can not be
        blank."></asp:RequiredFieldValidator>
        <br />
        <asp:CustomValidator ID="CustomValidator1" runat="server"
        ErrorMessage="Invalid entry. Correct username is 'testuser'."
        ClientValidationFunction="validate"
        ControlToValidate="TextBox1"></asp:CustomValidator>
        <br />
        <br />
        <br />
                    
                    
        <asp:Button ID="Button1" runat="server" BackColor="Black" Font-Bold="True"
        Font-Size="Small" ForeColor="White" Text="Submit" />
        </div>
        <div id="footer">
                <p class="left">
                All content copyright &copy; Kogent Solutions Inc.</p>
        </div>
    </form>

</body>

</html>
```

In the source file, `Default.aspx`, we have created a procedure named Validate in the `<script>` tag. The string that the user enters should be same to the string defined, testuser, in the script. Now, run the application. The `CustomValidator` control displays an error message because the username we provided is incorrect, as shown in Figure 19.5:

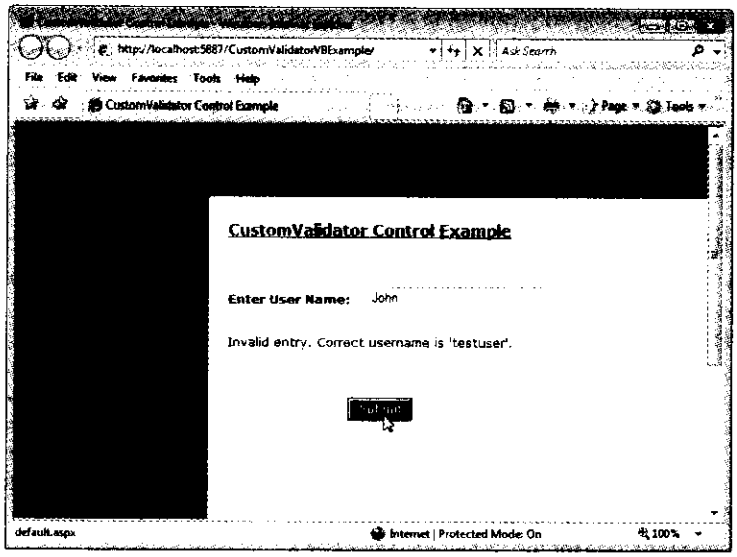



**Figure 19.5: CustomValidator Control Displaying Error Message on a Web page**

# The **ValidationSummary** Control

The `ValidationSummary` control collects all the Validation control error messages (known as summary) and displays it collectively on the screen. The display of the summary, which can be a list, a bulleted list, or a single paragraph, can be set by using the `DisplayMode` property. You can also specify whether the summary should be displayed in the Web page or in a message box by setting the `ShowSummary` and `ShowMessageBox` properties, respectively. The ValidationSummary control exists within the `System.Web.UI.WebControls` namespace.

Here is the inheritance hierarchy for the `ValidationSummary` class:

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

Noteworthy public properties of the `ValidationSummary` class are listed in Table 19.10:

**Table 19.10: Noteworthy Public Properties of ValidationSummary Class**

| | |
|---|---|
| `DisplayMode` | Obtains or sets the display mode of the `ValidationSummary` control |
| `EnableClientScript` | Obtains or sets a value indicating whether the `ValidationSummary` control updates itself using the client-side script |
| `ForeColor` | Obtains or sets the foreground color of the control |
| `HeaderText` | Obtains or sets the header text displayed at the top of the summary |
| `ShowMessageBox` | Obtains or sets a value showing whether the validation summary is displayed in a message box |
| `ShowSummary` | Obtains or sets a value showing whether the validation summary is displayed in line |
| `ValidationGroup` | Obtains or sets the group of controls for which the `ValidationSummary` object displays validation messages |

While using a `ValidationSummary` control in a Web application, if you want to display all the error messages on the same Web page, you must set the `ShowSummary` property of the `ValidationSummary` control to `True`. In case you want to display the error messages in a message box, set the `ShowMessageBox` property of the `ValidationSummary` control to `True`. By setting both these properties to `True`, you can display the error messages in both ways. You can also set the mode in which you want to display the error messages, by setting the `DisplayMode` property of the `ValidationSummary` control. The possible values for this property are `List`, `BulletList`, and `SingleParagraph`.

## Using the ValidationSummary Control

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

```
<asp:ValidationSummary ID="ValidationSummary2" runat="server" />
```

Now, let's create an application named ValidationSummaryVBExample. You can find the code of ValidationSummaryVBExample application in the Code\ASP.NET\Chapter 19\ ValidationSummaryVBExample folder on the CD. In this example, we are taking two `RequiredFieldValidator` controls and one `ValidationSummary` control on a Web page. The `ValidationSummary` control displays a summary of the error messages from all the other Validation controls. Listing 19.6 shows the Default.aspx page for the ValidationSummary control:

**Listing 19.6:** Showing the Code for the `Default.aspx` Page

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb"
  Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>ValidationSummary Control Example</title>
<link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <form id="Form" runat="server">
  <div id="header">
  </div>
  <div id="sidebar">
        <div id="nav">
               
        </div>
  </div>
  <div id="content">
        <div class ="itemContent">
        <div>
        <asp:Label ID="Label1" runat="server" Font-Bold="True" Font-Size="Medium"
        Font-Underline="True" Text="ValidationSummary Control Example"></asp:Label>
        <br />
        <br />
        <asp:Label ID="Label2" runat="server" Text="First Name:  "></asp:Label>
        <asp:TextBox ID="TextBox1" runat="server" Width="174px"></asp:TextBox>
        <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
        ControlToValidate="TextBox1" Display="None"
        ErrorMessage="First Name field can not be blank."></asp:RequiredFieldValidator>
        <br />
        <br />
        <asp:Label ID="Label3" runat="server" Text="Last Name:  "></asp:Label>
        <asp:TextBox ID="TextBox2" runat="server" Width="174px"></asp:TextBox>
        <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server"
        ControlToValidate="TextBox2" Display="None"
        ErrorMessage="Last Name field can not be blank."></asp:RequiredFieldValidator>
```

```
            <br />
            <br />
            <asp:ValidationSummary ID="ValidationSummary1" runat="server" Height="65px"
            Width="439px" />
            <br />
            <asp:Button ID="Button1" runat="server" BackColor="Black" Font-Bold="True"
            Font-Size="Small" ForeColor="white" Text="Submit" />
            </div>
            <div id="footer">
                  <p class="left">
                  All content copyright &copy; Kogent Solutions Inc.</p>
    </form>
</body>
</html>
```

Now, run the application and leave both the `TextBox` controls blank and click the Submit button. Notice that errors related to both the `RequiredFieldValidator` controls in the `ValidationSummary` control appear, as shown in Figure 19.6:

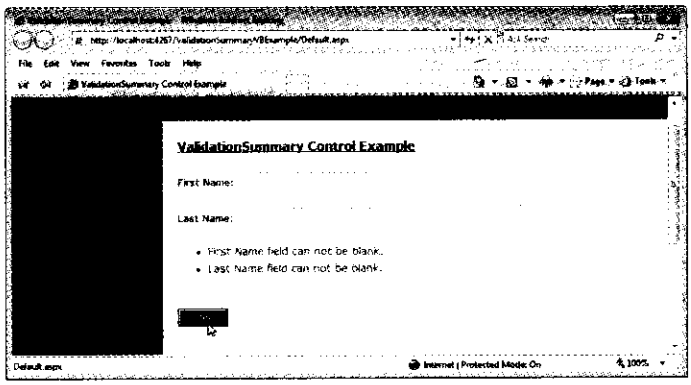



**Figure 19.6: ValidationSummary Control on a Web page Showing Error Messages**

# Summary

In this chapter, we have learned about the Validation controls, for example, `RequiredFieldValidator`, `CustomValidator`, and the rest. These controls are used to validate data entered by the user at the client-end. This validation, in turn, helps in reducing the load on the server by eliminating the need to send every piece of data to the server for validation. The associated control property for each control needs to be set for its association with standard controls. At the time of designing a website, all the validation controls require an expression property. This property helps validate the data entered by a user in the associated control.

In the next chapter, we describe the Web Parts controls.

# Quick Revise

**Q1. What are the validation controls available in ASP.NET?**

Ans: ASP.NET includes the following validation controls:

- `RequiredFieldValidator` – Validation succeeds as long as the input control does not contain an empty string.
- `RangeValidator` – Validation succeeds if the input control contains a value within a specific numeric, alphabetic, or date range.
- `CompareValidator` – Validation succeeds if the input control contains a value that matches the value in another specified input control.

- `RegularExpressionValidator` —Validation succeeds if the value in an input control matches a specified regular expression.
- `CustomValidator`—Validation is performed by a user-defined function.
- `ValidationSummary`—Displays summary of all current validation errors.

**Q2. How can we display all validation messages in one control?**

Ans: By using a `ValidationSummary` control.

**Q3. Which data type does the RangeValidator control support?**

Ans: Integer, String, and Date.

**Q4. Which control will you use if you need to ensure that the values in two different controls match?**

Ans: The `CompareValidator` Control.

**Q5. Which method is used to force all the validation controls to run?**

Ans: `Page.Validate()`.

**Q6. Which property validates only the validation controls within the specified group?**

Ans: `ValidationGroup` property.

**Q7. What does '+' sign signifies in regular expressions?**

Ans: The '+' sign stands for zero or more of.

**Q8. Mention operator property types of the CompareValidator Control?**

Ans:
- NotEqual
- GreaterThan
- GreaterThanEqual
- LessThan
- LessThanEqual
- DataTypeCheck

**Q9. Which validation control is used to validate whether number is even or not?**

Ans: `CustomValidator control`

**Q10. Which property of VaidationSummary control is used to display summary in a message box?**

Ans: `ShowMessageBox property`