TRY IT OUT Working with List Controls

In this exercise you add two list controls to a page. Additionally, you add a button that, when clicked, displays the selected items as text in a `Label` control.

1. In the `Demos` folder, create a new Web Form called `ListControls.aspx`. Make sure you create a Code Behind file by checking the Place Code in Separate File option.
2. Switch to Design View and drag a `DropDownList` from the Toolbox onto the design surface of the page within the dashed border of the `<div>` element that is already present in your page.
3. Notice that as soon as you drop the `DropDownList` control on the page, a pop-up menu appears that is labeled DropDownList Tasks, as shown in Figure 4-6.

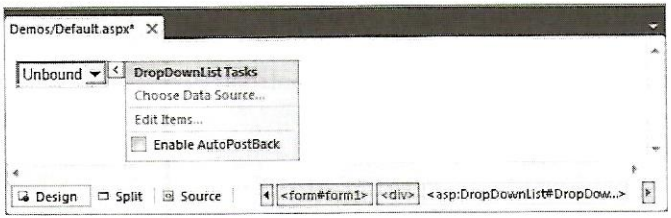

FIGURE 4-6

This pop-up menu is called the *Smart Tasks* panel. When it appears, it gives you access to the most common tasks of the control it belongs to. In the case of the `DropDownList`, you get three options. The first option enables you to bind the control to a data source, which is demonstrated in Chapter 13. The second item enables you to manually add items to the list, whereas the last option sets the `AutoPostBack` property of the control. With this option checked, the control will

submit the page it is contained in back to the server as soon as the user chooses a new item from the list.

The Smart Tasks panel only appears for the more complex controls that have a lot of features. You won't see it for simple controls like `Button` or `Label`. To reopen the Smart Tasks panel, right-click the control in the designer and choose Show Smart Tag. Alternatively, click the small arrow at the top-right corner of the control, visible in Figure 4-6.

On the Smart Tasks panel of the `DropDownList`, click the Edit Items link to bring up the ListItem Collection Editor, shown in Figure 4-7.

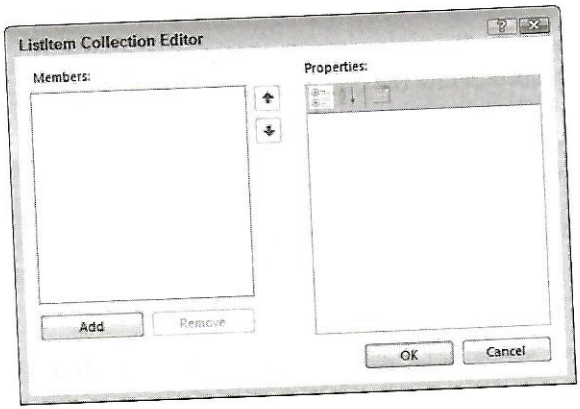

FIGURE 4-7

This dialog box enables you to add new items to the list control. The items you add through this window will be added as `<asp:ListItem>` elements between the tags for the control.

4. Click the Add button on the left side of the screen to insert a new list item. Then in the Properties Grid on the right, enter **C#** for the `Text` property and press Tab. As soon as you tab away from the `Text` property, the value is copied to the `Value` property as well. This is convenient if you want both the `Text` and the `Value` property to be the same. However, it's perfectly OK (and quite common) to assign a different value to the `Value` property.

5. Repeat step 4 twice, this time creating list items for Visual Basic and CSS. You can use the up and down arrows in the middle of the dialog box to change the order of the items in the list. Finally, click OK to insert the items in the page. You should end up with the following code:

```
<asp:DropDownList ID="DropDownList1" runat="server">
  <asp:ListItem>C#</asp:ListItem>
  <asp:ListItem>Visual Basic</asp:ListItem>
  <asp:ListItem>CSS</asp:ListItem>
</asp:DropDownList>
```

6. Switch to Markup View and drag a `CheckBoxList` control from the Toolbox directly into the code window, right after the `DropDownList`.

7. Copy the three `<asp:ListItem>` elements from the `DropDownList` you created in steps 4 and 5 and paste them between the opening and closing tags of the `CheckBoxList`. You should end up with this code:

```
  <asp:ListItem>CSS</asp:ListItem>
</asp:DropDownList>
<asp:CheckBoxList ID="CheckBoxList1" runat="server">
  <asp:ListItem>C#</asp:ListItem>
  <asp:ListItem>Visual Basic</asp:ListItem>
  <asp:ListItem>CSS</asp:ListItem>
</asp:CheckBoxList>
```

8. Switch to Design View and drag a `Button` from the Toolbox in Design View to the right of the `CheckBoxList` control. The `Button` will be placed below the `CheckBoxList`. Next, drag a `Label` control and drop it to the right of the `Button`. Create some room between the `Button` and the `Label` by positioning your cursor between the controls and then pressing Enter twice. Double-click the `Button` to open the Code Behind of the page.

9. In the code block that VWD added for you, add the following bolded code, which will be executed when the user clicks the button:

VB.NET

```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) _
          Handles Button1.Click
  Label1.Text = "In the DDL you selected " &
                        DropDownList1.SelectedValue & "<br />"

  For Each item As ListItem In CheckBoxList1.Items
    If item.Selected Then
      Label1.Text &= "In the CBL you selected " & item.Value & "<br />"
    End If
  Next
End Sub
```

C#

```
protected void Button1_Click(object sender, EventArgs e)
{
  Label1.Text = "In the DDL you selected " +
              DropDownList1.SelectedValue + "<br />";

  foreach (ListItem item in CheckBoxList1.Items)
  {
    if (item.Selected == true)
    {
      Label1.Text += "In the CBL you selected " + item.Value + "<br />";
    }
  }
}
```

Notice how in the VB.NET code the underscore is needed again to split the code over two lines. VB.NET requires the underscore if you want to move the `Handles` keyword to its own line.

10. Save the changes to the page and then request it in the browser. Choose an item from the `DropDownList`, check one or more items in the `CheckBoxList`, and click the button. You should see something similar to Figure 4-8, which shows the page in Firefox.

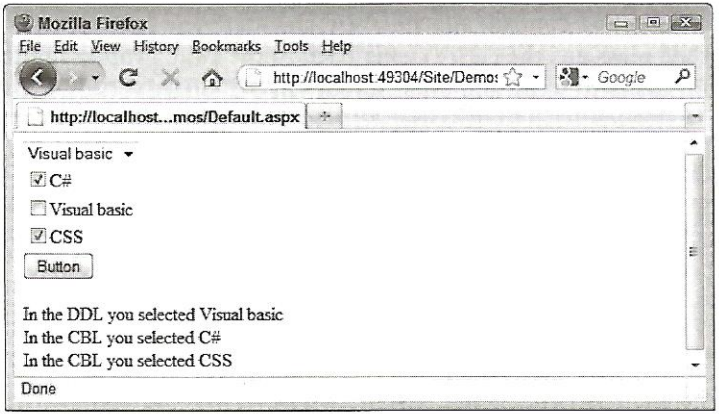

FIGURE 4-8