Chapter 3
Working with the Standard Web Server Controls
Overview

- This presentation covers the initial overview of the different server controls.
Introducing Server Controls

- Normal HTML tags such as `<input>`, `<H1>` and `<select>` are not processed by the server but are sent to and displayed by the browser.
- Server controls, in contrast, are tags that can be understood by the server.
- Each ASP.NET server control has an object model containing properties, methods, and events.
Introducing Server Controls

- There are five kinds of server controls:
  - HTML server controls
  - Web server controls
  - Validation server controls
  - User controls
  - Custom server controls

- All five of these different types of controls can be used within a single given Web form.
HTML Server Controls

- Most standard HTML tags can be turned into HTML server controls simply by adding the `runat="server"` attribute.
  - This `runat` attribute indicates that the element is to be processed on the server.
  - You can programmatically respond to events or bind data.
HTML Server Controls

- HTML server controls can be useful for:
  - situations in which you need complete control over how the HTML element will be rendered in the browser, or
  - when migrating an existing ASP page to ASP.NET.

- Web server controls are almost always preferable to HTML server controls due to their richly typed object model.
Web Server Controls

- Web server controls are also created on the server and they require a `runat="server"` attribute to work.

```xml
<asp:Button ID="myButton" runat="server" />
```
Web Server Controls

- Some Web server controls represent traditional HTML form elements such as buttons and drop-down lists.
- Other Web server controls represent more complex or abstract elements such as calendars, data lists, data sources, and tree views.
  - These more complex Web server controls do not necessarily map one-to-one (or even at all) to any existing HTML tags and can in fact be realized by dozens if not hundreds of HTML tags and many lines of Javascript code.
Validation Controls

- These controls allow you to test a user’s input for validity.
  - They are actually a special type of Web server control.

- Validation controls encapsulate common user input validation checks required by most Web applications:
  - ensuring that a required field is not empty,
  - comparing an input value against another value,
  - checking if a value falls within a range,
  - verifying that an input value matches a given pattern.

- Covered in Chapter 5.
User Controls

- These are developer-created controls that use the same programming techniques used to write Web forms pages.
- They typically allow the encapsulation of the functionality of multiple server controls along with other ASP.NET or HTML content in a single unit.
- Covered in Chapter 6.
Custom Server Controls

- A custom server control is a Web server control that you can create. A custom control is a compiled class and may combine multiple existing server controls.

- A custom control, unlike a user control, contains no declarative elements, and can be extended, use templates, support data binding, and be redistributed in a precompiled assembly.

- Partially covered in Chapter 14.
Web Server Control Overview

- Web server controls are added to a Web form in the same way as any HTML element:
  - You can type the markup code in Source view in VS.
  - You can use drag-and-drop from Source or Design view.
  - As well, you can programmatically add controls at runtime.
Web Server Control Syntax

<asp:controlName id="some_id" runat="server"
    «other attributes»>
</asp:controlName>
# List of the Common Web Server Controls

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BulletedList</td>
<td>Displays a bulleted list of items.</td>
</tr>
<tr>
<td>Button</td>
<td>Displays a push button that posts a Web form page back to the server.</td>
</tr>
<tr>
<td>Calendar</td>
<td>Displays a month calendar from which the user can select dates.</td>
</tr>
<tr>
<td>CheckBox</td>
<td>Displays a check box for selected true or false values.</td>
</tr>
<tr>
<td>CheckBoxList</td>
<td>Displays a multiselection check box group.</td>
</tr>
<tr>
<td>DropDownList</td>
<td>Displays a drop-down list for selecting a value from a list of values.</td>
</tr>
</tbody>
</table>
## List of the Common Web Server Controls

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HiddenField</td>
<td>Stores a nondisplayed value in a form that needs to be persisted across posts.</td>
</tr>
<tr>
<td>HyperLink</td>
<td>Displays a hyperlink that when clicked requests a different page.</td>
</tr>
<tr>
<td>Image</td>
<td>Displays an image.</td>
</tr>
<tr>
<td>ImageButton</td>
<td>Displays an image that posts the form back to the server</td>
</tr>
<tr>
<td>ImageMap</td>
<td>Displays an image with predefined hot spot regions that post back to the server or navigate to a different page.</td>
</tr>
<tr>
<td>Label</td>
<td>Displays static content that can be set programmatically and whose content can be styled.</td>
</tr>
</tbody>
</table>
## List of the Common Web Server Controls

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LinkButton</td>
<td>Creates a hyperlink-style button that posts the form back to the server.</td>
</tr>
<tr>
<td>ListBox</td>
<td>Displays a hyperlink that when clicked requests a different page. Creates a single- or multiselection list.</td>
</tr>
<tr>
<td>Literal</td>
<td>Like the Label, displays static content that is programmable. Unlike the Label control, it does not let you apply styles to its content.</td>
</tr>
<tr>
<td>RadioButton</td>
<td>Creates a radio button form element.</td>
</tr>
<tr>
<td>RadioButtonList</td>
<td>Creates a group of radio button form elements.</td>
</tr>
<tr>
<td>Table</td>
<td>Creates an HTML table; principally used for programmatically constructing a table.</td>
</tr>
<tr>
<td>TextBox</td>
<td>Creates a text box form element.</td>
</tr>
</tbody>
</table>
Common Members

- Learning how to work with all of these controls might seem a daunting prospect.
- Thankfully, Web server controls share a common object model.
Object Model
Common Members

- Most Web server controls inherit from the `WebControl` class.
  - This `WebControl` class in turn inherits from the `Control` class.

- Both of these base classes define a variety of properties, methods, and events that are available to all controls derived from them.
  - Most of these modify the formatting and display of the controls.
## Some Properties of the WebControl Class

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BackColor</td>
<td>The background color (either standard HTML color identifier or the name of the color) of the control.</td>
</tr>
<tr>
<td>BorderWidth</td>
<td>The thickness (in pixels) of the control's border.</td>
</tr>
<tr>
<td>CssClass</td>
<td>The CSS class name assigned to the control.</td>
</tr>
<tr>
<td>Font</td>
<td>Font information for the control. This property contains subproperties.</td>
</tr>
<tr>
<td>ToolTip</td>
<td>The tool tip text (i.e., the text that appears when the mouse rests over control) for the control.</td>
</tr>
<tr>
<td>Width</td>
<td>The width of the control.</td>
</tr>
</tbody>
</table>
## Some Properties of the Control Class

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>The unique identifier for the control.</td>
</tr>
<tr>
<td>Visible</td>
<td>Specifies whether the control is visible.</td>
</tr>
</tbody>
</table>
Subproperties

- Properties can also have properties; these are called **subproperties**.

- For instance, the **Font** property is a complex object with properties of its own, such as **Name** and **Size**.

- When working with subproperties programmatically, you use dot notation.
  - e.g.,
  ```csharp
  somecontrol.Font.Size=10;
  ```

- When working with subproperties declaratively, you use hyphen (-) notation.
  - e.g.,
  ```xml
  <asp:Label id="acontrol" Font-Size="10"
  runat="server" />
  ```
Manipulating Properties Programmatically

- You can retrieve the value of a property or set the value of a property at runtime.
- These properties are strongly typed and vary depending upon the property.
  - Some properties have a primitive data type such as a Boolean or a numeric.
  - Other property values are defined by an enumerated type or some other type.

```csharp
// Using a primitive
myLabel.Text = "Randy";
string abc = myTextBox.Text;
myLabel.Visible = false;

// Using an enumeration
myBulletedList.BulletStyle = BulletStyle.Circle;
TextBoxMode mode = myTextBox.TextMode;
```
Event Properties

- All controls support events.
- You can specify the event handler method for a given event declaratively by affixing the `On` prefix to the event property name.
  - E.g., if you have an event handling method named `btnOne_Click` that you want to run when the user clicks a button, you would use:

```xml
<asp:button id="btnOne" runat="server"
            Onclick="btnOne_Click" />
```
Event Methods

- All event methods in the .NET Framework are `void` methods with two parameters:
  - the source object that raised the event
  - the data for the event, usually contained within an EventArgs object (or an object derived from it).
- Thus, the method signature for the event handler would be:

```csharp
public void btnOne_Click(object source, EventArgs e)
{
    // code goes here
}
```
Unit-Based Measurement Properties

- Various measurement properties such as Width, Height, and Font.Size are implemented using the Unit structure.
- This Unit structure allows you to use any HTML- or CSS-compatible size unit, such as cm, inch, and pixels.
- These size units are defined in the UnitType enumeration.
Color-Based Properties

- All color-based properties use the Color structure.
- The Color structure defines:
  - all the HTML 4.0 system-defined colors
  - the color names supported by most browsers
  - methods for retrieving and specifying color using different color models (such as HSB and RGB).
Collection Properties

- Some properties are not a single structure, primitive, or enumerated type, but a collection of other objects.
- An example of such a property is the Items collection of the DropDownList control.
  - This collection contains zero or more ListItem objects.
  - These collection properties have their own methods and properties for determining the size of the collection, as well for adding, retrieving, and removing items from the collection.

```csharp
DropDownList drpSample = new DropDownList();

// Create the item, then add to collection
ListItem li = new ListItem("Item 2");
drpSample.Items.Add(li);

// Combine the creation and addition to collection steps
drpSample.Items.Add(new ListItem("Item 1"));
```
Additional Control Attributes

- Any given control can be given additional HTML attributes.
  - Attributes are name-value pairs that will be added to the rendered HTML.
  - Generally used to provide additional client-side scripting behavior to a control.

```javascript
myButton.Attributes["onmouseout"] = "document.bgColor='green';";
myButton.Attributes["onmouseover"] = "document.bgColor='blue';";
```

**rendered as**

```html
<input type="submit" name="myButton" value="Click Me" id="myButton"
    onmouseout="document.bgColor='green';"
    onmouseover="document.bgColor='blue';" />
```
Essential Controls

- See pages 107-176 for details.
Label controls are (by default) rendered in the browser as static text within an HTML `<span>` element, e.g.,

```html
<asp:Label id="labMsg" runat="server" text="hello"/>
```

will be rendered as

```html
<span id="labMsg">hello</span>
```

The Literal control does not add any HTML elements to the text, e.g.,

```html
<asp:Literal id="litMsg" runat="server" text="hello"/>
```

will be rendered as

`hello`
You can change the rendering of a Label control via the AccessKey attribute.

It allows the user to use a combination of keyboard keys (on Windows machines, usually the ALT key plus some other key) to move the focus to an associated input control.
<asp:Label ID="labName" runat="server"
    AccessKey="N"
    AssociatedControlID="txtName"
    Text="&lt;u&gt;N&lt;/u&gt;ame" />
<asp:TextBox ID="txtName" runat="server" />

rendered as

<label for="txtName" id="labName" AccessKey="N" >
    &lt;u&gt;N&lt;/u&gt;ame
</label>
<input type="text" id="txtName" />
**Text Box**

```
Example of TextBoxes - Windows Internet Explorer

http://localhost:1083/Chapter3/TextBoxTest.aspx

Name:
John Locke

State:
AB

Comments:
He is the author of "An Essay Concerning Human Understanding"

Password:
****

MaxLength="2"

TextMode="MultiLine"

TextMode="Password"
```
Button-Style Controls

- Support two main events. Both are raised when user clicks on the button.
  - Click
    - Used for single buttons
  - Command
    - Used when you want several related buttons to share the same event handler.
### Key Properties of ListControl

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataTextField</td>
<td>Specifies the field name of the data source that will provide the textual content for the list.</td>
</tr>
<tr>
<td>DataTextFormatString</td>
<td>Specifies the formatting string that controls the visual display of the list content.</td>
</tr>
<tr>
<td>DataValueField</td>
<td>Specifies the field name of the data source that will provide the value for each list item.</td>
</tr>
<tr>
<td>Items</td>
<td>The collection of <code>ListItem</code>s in the control.</td>
</tr>
<tr>
<td>SelectedIndex</td>
<td>The index (starting with 0) of the selected list item(s).</td>
</tr>
<tr>
<td>SelectedItem</td>
<td>The list item that was selected.</td>
</tr>
<tr>
<td>SelectedValue</td>
<td>The value of the list item that was selected.</td>
</tr>
</tbody>
</table>
HyperLink vs LinkButton

- Both are visually displayed in the browser as a hyperlink (i.e., the `<a>` tag).
- HyperLink
  - simply a link that can be programmatically manipulated.
  - Does not generate a postback.
- LinkButton
  - Generates a postback and thus supports postback event handling (e.g., OnClick).
The Table web server control is used for creating server-programmable tables.

It is usually easier to use the HTML `<table>` element for static tables.

The Table server control can be used to dynamically add rows or columns to a table at run-time.
Table Control

- The Table control can also make it easier to create accessible tables.
- ASP.NET 2.0 now supports those features in HTML 4.0 that help those who use screen readers or assistive technologies to understand a web page’s tables.
Calendar Control

- The Calendar control is perhaps the most complex of the basic server controls covered in this chapter.
- It displays a single month calendar that allows a user to navigate from month to month and to select dates, weeks, or entire months.
Style Elements

- The appearance of the Calendar control can be further customized by setting various style elements.
- Many of the more complex web server controls in ASP.NET use style elements.
  - These are additional tags that are embedded within the parent control and which allow you to customize the appearance of that control.

```xml
<asp:calendar id="calTest" runat="server">
  <titlestyle font-size="14px" font-bold="true" />
</asp:calendar>
```