Web Controls

The mainstay of ASP.NET
Categories

• Simple (HTML equivalent)
• Buttons
• List Controls
• Data-Bound Controls
• Calendar Controls
• Validation Controls
TextBox Controls

• Use the Text property to move text back and forth.
• Set the type to password to mask entries.
• Usually don’t use with events although an event can be generated if the text changes. This does NOT cause a postback. The event is issued on the next postback.
• AutoPostBack can be set to cause a post back when the text changes. The postback does not actually take place until the control loses the input focus, e.g., the user selects another control.
TextBox Controls – contd.

• TextMode can be set to MultiLine to allow more than one line of text to be entered.
• The number of lines displayed is controlled by the Rows property.
• ToolTip is another neat property. You can set a text string to be displayed as a tiny flyout as the mouse is moved over the text.
• How is this done?
Flyout Implementation

• This is actually an HTML feature and not something out of the .NET world.
• The title="..." property of the tag is used to set the flyout (tooltip) text.
• Remember – ASP.NET has to serve up pages that have only HTML (static and dynamic) along with Javascript.
• Be sure to check out the other properties of the TextBox control.
General Features

• Visible – used to dynamically hide or show a control.
• ReadOnly – used to prevent data from being entered into a control by the user. Good for output that you want to have the appearance of a control. Use a Label otherwise.
• Font and Colors – Set to what you want.
• Dimensions – can change dynamically as well as setting up using the designer.
Data Conversion

• TextBox controls only handle text.
• Conversion to/from numerical values can be accomplished using the Convert class and the ToString method.
• Example:
  ```csharp
  string s = "123";
  int i = Convert.ToInt32(s);
  s = i.ToString();
  ```
• Careful! Convert throws an exception if the string is not in the proper format for the conversion.
Handling Exceptions

```csharp
try
{
    i = Convert.ToInt32(myBox.Text);
}
catch
{
    //failed to convert, use default
    i = 0;
}
```

- Do something reasonable from the user’s perspective.
Label Controls

- These make great placeholders for text to be output from the server side without embedding Response.Write in the HTML.
- Use foreground/background colors, font, and border styles to display alerts etc.
- Hide using the Visible property.
- Label controls emit <span>…</span> tags.
- There is no reason to use a Label control unless you wish to change the text from the server side.
CheckBox Controls

• The major difference between a regular checkbox and this control is that we can use a simple property, Checked, to determine if the box is checked.

• A regular HTML checkbox would only return its state as form data.

• The following example shows the difference and the advantage of web controls in general.
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" 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></title>
</head>
<body>
    <form id="form1" runat="server">
        <input id="Checkbox1" type="checkbox" name="Checkbox1" />
        HTML checkbox<br />
        <br />
        <asp:CheckBox ID="CheckBox2" runat="server" Text="Web control checkbox" />
        <br />
        <br />
        <asp:Button ID="Button1" runat="server" Text="Submit" />
    </form>
</body>
</html>
private void Page_Load(object sender, System.EventArgs e) {
    if(this.IsPostBack)
    {
        Response.Write("<br>Web control=
        + CheckBox2.Checked.ToString());
        string s = Request["Checkbox1"];  
        Response.Write("<br>HTML checkbox="+s);
    }
}

• The Request collection contains the key/value pairs for the form fields.
Note – the button state is NOT preserved.
List Controls

- ListBox
- DropDownList
- CheckBoxList
- RadioButtonList

- All list controls can use data binding. We will discuss data binding shortly.
List Control Properties

• The Items collection contains the ListItem’s to be displayed in the control.
• Each ListItem has a Text and Value property as well as a Selected boolean flag.
• A static list can be created with the VS 2015 collection editor.
• Optionally, the list can be created dynamically.
  MyListBox.Items.Add(“Itemtext”);
  or
  MyListBox.Items.Add(new ListItem(“Itemtext”, “ItemValue”));
Getting the Selected Index

• MyListBox.SelectedIndex is the index of the item.
• We can use this to access the item: 
  MyListBox.Items[index].Text
  or
  MyListBox.Items[index].Value
• MyListBox.SelectedValue can also be used.
HyperLink Controls

• Used like an HTML “href” tag.
• The advantage is that the link can be changed dynamically.
• Either text or an image can be used as the control.
• Set the Text property to display text.
• Set the ImageUrl property to display an image.
• The NavigateUrl property specifies the actual URL of the hyperlink.
Image Controls

- Similar to the standard image tag in HTML but allow dynamic control.
- The ImageUrl property specifies the location of the image.
- This is a useful control for displaying images in response to some user input.
- Don’t try to use this control in lieu of Javascript image flipping using the mouseover event. The response will be too slow.
RadioButton Controls

• Much as one would suspect.
• Must specify a group name for the radio buttons. This controls the single selection feature of the radio buttons.
• Must check each button individually.
• Radio button use is similar to check box controls.
• Can use RadioButtonList to get around this problem. (discussed later)
Table Controls

• Tables are a mainstay of HTML page design.
• The Table Control allows changing the cells dynamically.
• Tables can be created either statically or on-the-fly.
• See my online grade reporting system for an example of on-the-fly creation.
Panel Controls

• They don’t display anything.
• Are used to group other controls so you can turn their visibility on and off at will.
• One guess as to what HTML element is used for panels?

<div>
Button Controls

• Main use is to fire an event.
• Page_Load is invoked first and then the event handler for the button is called.
• The page is not posted back to the browser until both page load and the button event methods have been called.
Button Types

1. Button(plain)
   Like an HTML button.

2. Link button
   Looks like a link.

3. Image button
   Uses an image to render the button.
CheckBoxList Example
Selected Item(s):
One
Three
protected void Button1_Click(object sender, EventArgs e)
{
    Message.Text = "Selected Item(s):" + "<br / >" + "<br / >";
    for (int i = 0; i < checkboxlist1.Items.Count; i++)
    {
        if (checkboxlist1.Items[i].Selected)
            Message.Text = Message.Text +
            checkboxlist1.Items[i].Text + "<br / >";
    }
}
Data Binding

- Can bind data from a database or an XML file.
- In fact data can be bound from any type that supports IEnumerable or IListSource.
- You can implement these interfaces in your own classes.
- See example (converter2.aspx) from the Prosise book in the MELL collection.
Data Binding From an Array

protected string[] names = {"Tom", "Linda", "Erin", "Kristin"};

private void Page_Load(object sender, System.EventArgs e)
{
    dlist.DataSource = names;
    dlist.DataBind();
}

• The DataSource property of the list control must be set to the string array, names.
The Result
Data-Bound Controls

• Repeater
  – Simple, but not robust
• DataList
  – Support for multi-column formatting etc.
• DataGrid
  – Everything including the kitchen sink
• We will discuss these later
Calendar Controls

• Not just a static calendar display
• Allows the user to interact with the calendar to select dates.
• Relatively easy to use.
Validation Controls

• Allow validation of entries based on
  – Required entry
  – Range of values
  – Matches a regular expression
  – Custom

• Script code is generated to run on the client for performance.

• Backup checking is also done on the server if a client browser does not run scripts.
  – This also prevents hacking.

• Demo
HTML Controls

• Many standard HTML elements have an equivalent HTML control class in .NET.
• You can convert a regular HTML element to an HTML control by adding runat="server" to the elements tag.
• You can also do this from the VS 2011 designer by right clicking the control and selecting “Run As Server Control” menu.
HTML Controls

• There are some differences in the use of HTML controls vs. Web Controls.
• OnServerClick is used rather than OnClick as the click handler.
• The property Value is used rather than Text to access the control’s text.
• Numerous other differences in properties and events.
• HTML controls are most useful for adapting legacy pages for use with ASP.NET.