Beginning ASP.NET in C# and VB

Shortcuts

- Open in Browser CTRL + F5
- IntelliSense Invoke CTRL + Space, ENTER, TAB, '.'
- Update Cache CTRL + F5, CTRL + R

Chapter 2

- CTRL+F5, CTRL+R: Refresh

Chapter 3: CSS

- CSS Box Model

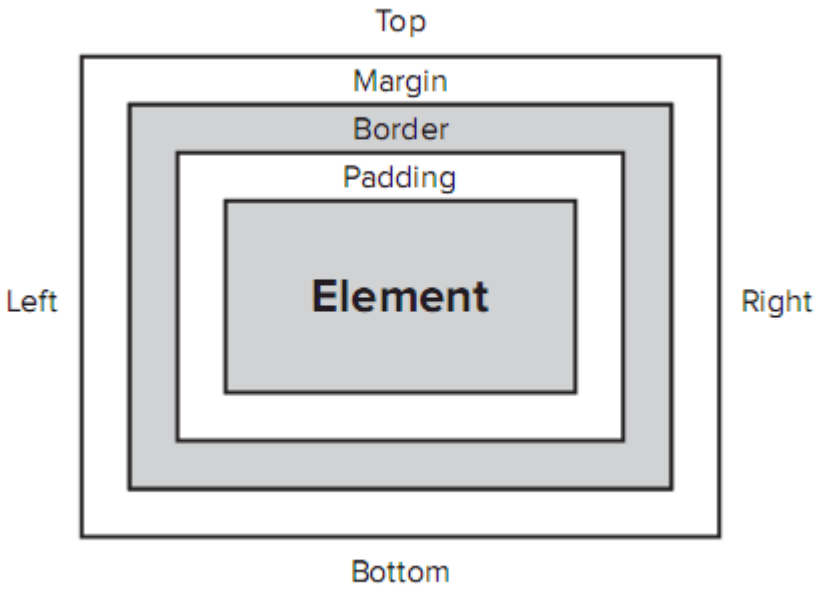


FIGURE 3-4

- Adding CSS to Pages
  - Linked
    - <link href=”StyleSheet.css” rel=”Stylesheet” type=”text/css” media=”screen” />
  - Embedded
    - <style type=”text/css”>
      h1
      {
      }
      </style>
  - Inline
    - <span style=”color: White;”>
- New Rules: Element, Class, ID

Chapter 4: Working with ASP.NET Server Controls

- With ASP.NET, the ASP Server converts the <asp:Control ID=”Unknown” Runt=”Server” /> into plain HTML. This reduces complexity at the browser level.

Property	Description
AccessKey	Enables you to set a key with which a control can be accessed at the client by pressing the associated letter.
BackColor ForeColor	Enables you to change the color of the background (BackColor) and text (ForeColor) of the control.

BorderColor BorderStyle BorderWidth	Changes the border of the control in the browser. The similarities with the CSS border properties you saw in the previous chapter are no coincidence. Each of these three ASP.NET properties maps directly to its CSS counterpart.
CssClass	Lets you define the HTML class attribute for the control in the browser. This class name then points to a CSS class you defined in the page or an external CSS file.
Enabled	Determines whether the user can interact with the control in the browser. For example, with a disabled text box (Enabled=”False”) you cannot change its text.
Font	Enables you to define different font-related settings, such as Font-Size, Font-Names, and Font-Bold.
Height Width	Determines the height and width of the control in the browser.
TabIndex	Sets the client-side HTML tabindex attribute that determines the order in which users can move through the controls in the page by pressing the Tab key.
ToolTip	Enables you to set a tooltip for the control in the browser. This tooltip, rendered as a title attribute in the HTML, is shown when the user hovers the mouse over the relevant HTML element.
Visible	Determines whether or not the control is sent to the browser. You should really see this as a server-side visibility setting because an invisible control is never sent to the browser at all. This means it’s quite different from the CSS display and visibility properties you saw in the previous chapter that hide the element at the client.

ASP

- <asp:TextBox AccessKey=”a” BackColor=”Black” ForeColor=”White” Font-Size=”30px” BorderColor=”Yellow” BorderStyle=”Dashed” BorderWidth=”4” CssClass=”TextBox” Enabled=”True” Height=”40” Width=”200” TabIndex=”1” ToolTip=”Hover text here” Visible=”True” ID=”TextBox1” runat=”server” Text=”Hello World”>
  </asp:TextBox>

HTML Rendering

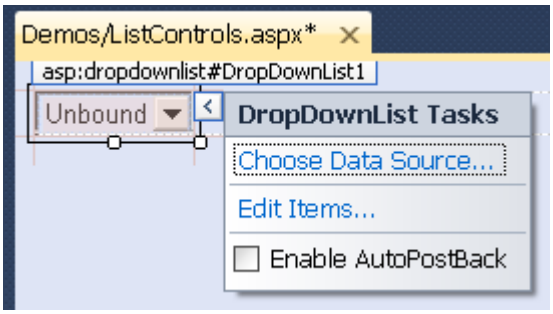
- <input name=”TextBox1” type=”text” value=”Hello World” id=”TextBox1” accesskey=”a” tabindex=”1” title=”Hover text here” class=”TextBox” style=”color:White; background-color:Black;border-color:Yellow;border-width:4px; border-style:Dashed;font-size:30px;height:40px;width:200px;”
  />

ASP wCSS

- <asp:TextBox ID=”TextBox1” AccessKey=”a” CssClass=”**TextBox**” TabIndex=”1” ToolTip=”Hover text here” runat=”server” Text=”Hello World”>
  </asp:TextBox>

```
.TextBox
{
 background-color: Black;
 color: White;
 font-size: 30px;
 border-color: Yellow;
 border-style: Dashed;
 border-width: 4px;
 height: 40px;
 width: 200px;
 }
```

Drop Downlists and Smart Tasks panel



ASP State Engine: __VIEWSTATE and Postback

- ASP packages all data not represented by HTML into a “HIDDEN” __VIEWSTATE field that is sent back to the ASP server. The server uses this hidden field to receive data sent by the Postback (i.e. pressing a form button).
- __VIEWSTATE and server processing of it causes latency, so disable it if it isn't needed on a particular control.

```
<form name=”form1” method=”post” action=”State.aspx” id=”form1” >

...

 <input type=”hidden” name=”__VIEWSTATE” id=”__VIEWSTATE”

      value=”/wEPDwULLTE5Njc4MzkzNDdkZI+lOWZMZpVv0hc7i/HFGMd OO8oc” />

</div>
```

- Disabling at the Site Level
    - At the web site level. You can do this in the **web.config** file in the root of the site by modi-fying the <pages> element under <system.web>, setting the enableViewState attribute to false:
      <pages enableViewState=”false”>

...
</pages>

- Page Level
  - At the page level. At the top of each page you find the page directive , a series of instructions that tell the ASP.NET runtime how the page should behave. In the page directive you can set **EnableViewState** to False:
    <%@ Page Language="VB" AutoEventWireup="False" CodeFile="State. aspx.vb" Inherits="Demos_State" EnableViewState="False" %>
- Control Level
  - **EnableViewState** = False
- ViewStateMode=Disabled at Page Level, then enable on control itself
  - <asp:Label ID="Label1" runat="server" Text="Label" ViewStateMode="Enabled" />
-

Chapter 5: Programming Your ASP.NET Web Pages

.NET Architecture

- .NET converts all Languages into an Intermediate Language (IL)
- CLR (Common Language Runtime) compiles the IL into DLL
- JIT (Just In Time) compiler reads local DLL and compiles it into machine code (based on target machine type) into HTML for sending to remote client

Data Types

- int
- char
- string
- bool

Converting and Casting Data types

- Label1.Text = System.DateTime.Now.ToString();
- bool myBoolean1 = Convert.ToBoolean("True"); // Results in true
- bool myBoolean2 = Convert.ToBoolean("False"); // Results in false

Commenting

- MSDN article on how to use commentig to create documentation automatically: **http://msdn.microsoft.com/magazine/cc302121.aspx**

- /*
  Multiline Comment
  */
- // Single Line Comment
- /// <summary>
  /// XML based multiline commenting
  ///<summary>

Assignments

- C# deals strictly with data types so...
  - 14.5 is interpreted as a 'double' so to force assignment at a 'decimal' use 14.5M
    - M(decimal)
    - D(double)
    - F(float0
    - L(long)
  - The following are equivalent
    - !!!! FIND IN TYPECASTING!!!! decimal myDecimal = 14.5 <--ConvertToDecimal;
    - decimal = 14.5M
- Use 'var' to force a particular datatype without forcing the type manually:
  - var myDecimal = 14.5M // this will force 'decimal' type

Strings and Escaped Characters

- \" = Double Quote
- \n = New Line
- \t = Horizontal tab
- \\ = backward slash
- @ = the following lines are equivalent:
  - path = @"c:\MyApp\Myfiles";
  - path = "c:\\MyApp\\Myfiles";
  -

Arrays

- Index starts at [0]

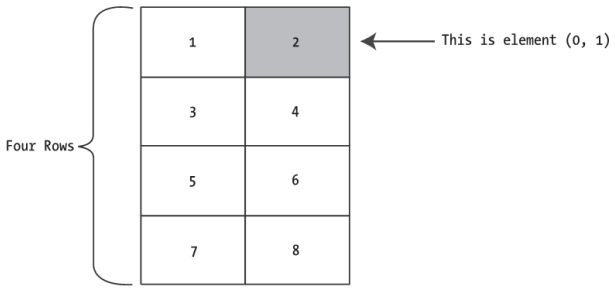
Single Dimensional

string[] stringArray = new string[4];

Two Dimensional Arrray

int[,] intArray = new int[2,4]
// Create a 4x2 array (a grid with four rows and two columns).
int[,] intArray = {{1, 2}, {3, 4}, {5, 6}, {7, 8}};

Figure 2-1 shows what this array looks like in memory.



Single Dimensional Array

int[] intArray = {1, 2, 3, 4};
int element = stringArray[2]; // element is now set to 3

Two Dimensional Array

int[,] intArray = {{1, 2}, {3, 4}, {5, 6}, {7, 8}};

// Access the value in row 0 (first row), column 1 (second column).
int element = intArray[0, 1]; // element is now set to 2.

ArrayList: Resizing Arrays

// Create an ArrayList object. It's a collection, not an array,
// so the syntax is slightly different.
ArrayList dynamicList = new ArrayList();

// Add several strings to the list.
// The ArrayList is not strongly typed, so you can add any data type
// although it's simplest if you store just one type of object
// in any given collection.
dynamicList.Add("one");
dynamicList.Add("two");
dynamicList.Add("three");

// Retrieve the first string. Notice that the object must be converted to a

```
// string, because there's no way for .NET to be certain what it is.
string item = Convert.ToString(dynamicList[0]);
```

Enumerations

While .NET uses enumeration extensively, programmers probably won't create too many variables that leverage enumeration.

Here's an example of an enumeration that defines different types of users:

```
// Define an enumeration type named UserType with three possible values.
enum UserType
{
   Admin,  // =0
   Guest,    // =1
   Invalid   // =2
}
```

Now you can use the UserType enumeration as a special data type that is restricted to one of three possible values. You assign or compare the enumerated value using the dot notation shown in the following example:

```
// Create a new value and set it equal to the UserType.Admin constant.
UserType newUserType = UserType.Admin;
```

Math Operations

```
int number;

number = 4 + 2 * 3;
// number will be 10.

number = (4 + 2) * 3;
// number will be 18.
```

Table 2-2. Arithmetic Operations

Operator	Description	Example
+	Addition	1 + 1 = 2
–	Subtraction	5 - 2 = 3
*	Multiplication	2 * 5 = 10
/	Division	5.0 / 2 = 2.5

%	ModulusGets	7 % 3 = 1