Web Form Fundamentals

Today You Will Learn

- Improving the Currency Converter
- A Deeper Look at HTML Control Classes
- The Page Class
- Application Events
- ASP.NET Configuration

Improving the Currency Converter

Adding Linked Images

- Add a new button, which will display the currency conversion rate graph.
- The program would need an additional button and image control.
- Add following code in CurrencyConverter.aspx

```
<input type="submit" value="Show Graph" ID="ShowGraph" runat="server" />
<img ID="Graph" src="" alt="Currency Graph" runat="server" />
```

- As it's currently declared, the image doesn't refer to a picture. For that reason, it makes sense to hide it when the page is first loaded.
- Add `Graph.Visible = False` to Page.Load Event in CurrencyConverter.aspx.vb
- When a server control is hidden, ASP.NET omits it from the final HTML page.

Improving the Currency Converter

- The currency converter has three possible picture fi les—pic0.png, pic1.png, and pic2.png.
- It chooses one based on the index of the selected item.

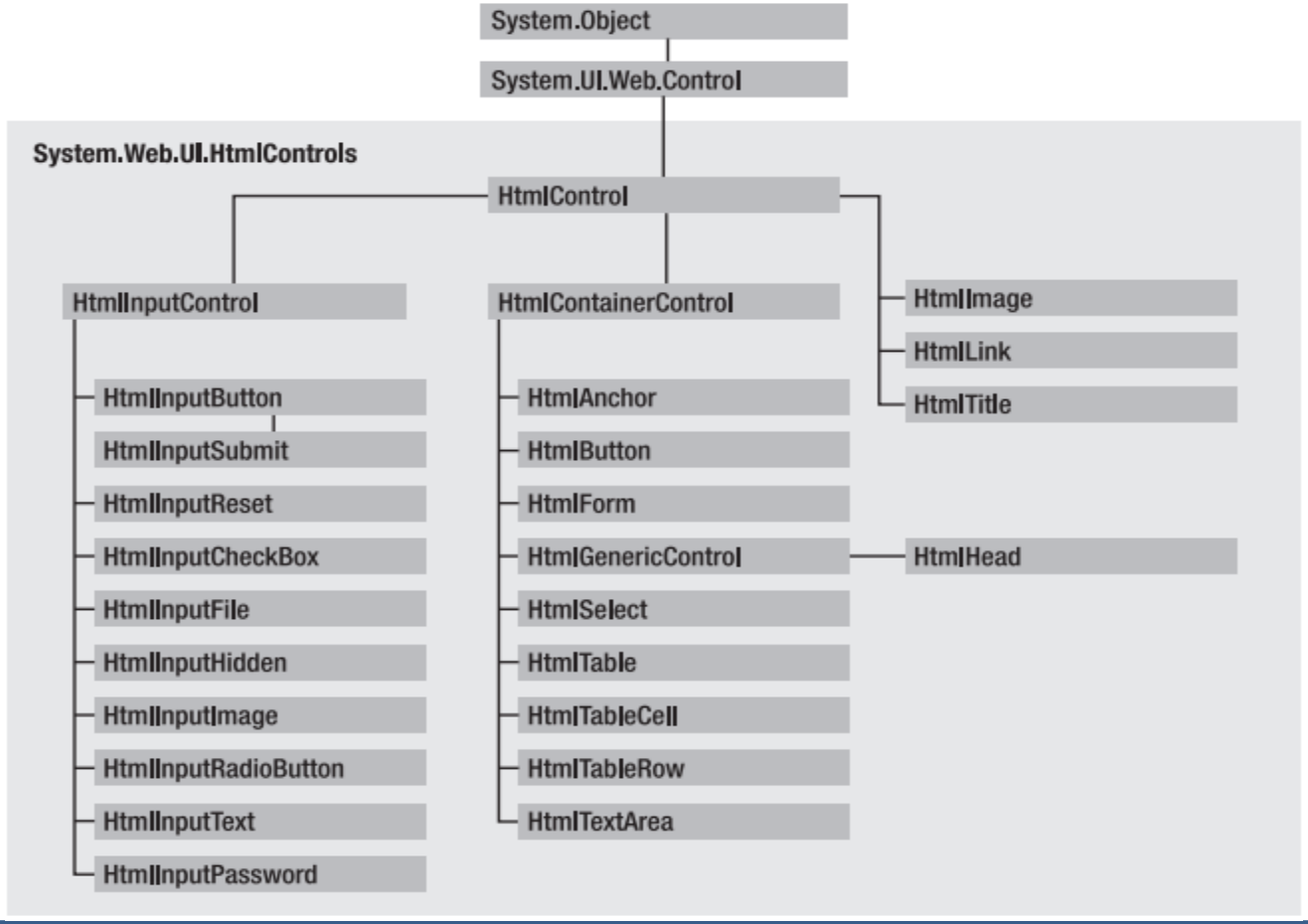
```
protected void ShowGraph_ServerClick(Object sender, EventArgs e)
{
        Graph.Src = "Pic" + Currency.SelectedIndex.ToString() + ".png";
        Graph.Visible = true;
}
```

To change the graph picture when the currency is changed, using this line of code in the Convert_ServerClick()method:

```
Graph.Src = "Pic" & Currency.SelectedIndex.ToString() & ".png“;
```

A Deeper Look at HTML Control Classes

- Related classes in the .NET Framework use inheritance to share functionality.



A Deeper Look at HTML Control Classes

HTML Control Events

- It provides one of two possible events: ServerClick or ServerChange.
- The ServerClick is simply a click that's processed on the server side. It's provided by most button controls, and it allows your code to take immediate action.
- The ServerChange event responds when a change has been made to a text or selection control. This event isn't as useful as it appears because it doesn't occur until the page is posted back.

Event	Controls That Provide It
ServerClick	HtmlAnchor, HtmlButton, HtmlInputButton, HtmlInputImage, HtmlInputReset
ServerChange	HtmlInputText, HtmlInputCheckBox, HtmlInputRadioButton, HtmlInputHidden, HtmlSelect, HtmlTextArea

A Deeper Look at HTML Control Classes

Advanced Events with the HtmlInputImage Control

- The .NET event standard indicates that every event should pass exactly two pieces of information.
- The first parameter identifies the object that fired the event.
- The second parameter is a special object that can include additional information about the event.

```
protected void Convert_ServerClick(object sender, EventArgs e)
{
        ...
}
```

The second parameter (e) has always been used to pass an empty System.EventArgs object. This object doesn't contain any additional information—it's just a glorified placeholder.

A Deeper Look at HTML Control Classes

- **O**nly one HTML server control sends additional information: the HtmlInputImage control.

```
protected void ImgButton_ServerClick(Object sender, ImageClickEventArgs e)
{ . . . }
```

An ImageClickEventArgs object provides X and Y properties representing the location where the image was clicked.

- Using this additional information, you can replace multiple button controls and image maps with a single, intelligent HtmlInputImage control.

A Deeper Look at HTML Control Classes

The HtmlControl Base Class Properties

Property	Description
Attributes	Provides a collection of all the attributes that are set in the control tag, and their values. Rather than reading or setting an attribute through the Attributes, it's better to use the corresponding property in the control class. However, the Attributes collection is useful if you need to add or configure a custom attribute or an attribute that doesn't have a corresponding property.
Controls	Provides a collection of all the controls contained inside the current control. (For example, a server control could contain an <input> server control.) Each object is provided as a generic System.Web.UI.Control object so that you may need to cast the reference to access control-specific properties.
Disabled	Disables the control when set to True, thereby ensuring that the user cannot interact with it, and its events will not be fired.
EnableViewState	Disables the automatic state management for this control when set to False. In this case, the control will be reset to the properties and formatting specified in the control tag every time the page is posted back. If this is set to True (the default), the control stores its state in a hidden input field on the page, thereby ensuring that any changes you make in code are remembered. (For more information, see the "View State" section earlier in this chapter.)

A Deeper Look at HTML Control Classes

Property	Description
Page	Provides a reference to the web page that contains this control as a System.Web.UI.Page object.
Parent	Provides a reference to the control that contains this control. If the control is placed directly on the page (rather than inside another control), it will return a reference to the page object.
Style	Provides a collection of CSS style properties that can be used to format the control.
TagName	Indicates the name of the underlying HTML element (for example, img or div).
Visible	Hides the control when set to False and will not be rendered to the final HTML page that is sent to the client.

- The HtmlControl class also provides built-in support for data binding

A Deeper Look at HTML Control Classes

The HtmlContainerControl Class

- Any HTML control that requires a closing tag inherits from the HtmlContainer class.
- Elements such as <a>, <form>, and always use a closing tag, because they can contain other HTML elements.
- Elements such as <img> and <input> are used only as stand-alone tags.
- Thus, the HtmlAnchor, HtmlForm, and HtmlGenericControl classes inherit from HtmlContainerControl , while HtmlInputImage and HtmlInputButton do not.

A Deeper Look at HTML Control Classes

- HtmlContainerControl Class support for some properties:

Property	Description
InnerHtml	The HTML content between the opening and closing tags of the control. Special characters that are set through this property will not be converted to the equivalent HTML entities. This means you can use this property to apply formatting with nested tags such as <b>, <i>, and <h1>.
InnerText	The text content between the opening and closing tags of the control. Special characters will be automatically converted to HTML entities and displayed like text (for example, the less-than character (<) will be converted to `<` and will be displayed as < in the web page). This means you can't use HTML tags to apply additional formatting with this property. The simple currency converter page uses the InnerText property to enter results into a <p> element.

A Deeper Look at HTML Control Classes

The HtmlInputControl Class

- The HtmlInputControl class inherits from HtmlControl and adds some properties that are used for the <input> element.
- The <input type="text"> element is a text box and <input type="submit"> is a button.

Property	Description
Type	Provides the type of input control. For example, a control based on <input type="file"> would return *file* for the type property.
Value	Returns the contents of the control as a string. In the simple currency converter, this property allowed the code to retrieve the information entered in the text input control.

The Page Class

- Every web page is a custom class that inherits from System.Web.UI.Page.
- The properties of Page Class are:

Property	Description
IsPostBack	This Boolean property indicates whether this is the first time the page is being run (False) or whether the page is being resubmitted in response to a control event, typically with stored view state information (True). You'll usually check this property in the Page.Load event handler to ensure that your initial web page initialization is only performed once.
EnableViewState	When set to False, this overrides the EnableViewState property of the contained controls, thereby ensuring that no controls will maintain state information.
Application	This collection holds information that's shared between all users in your website. For example, you can use the Application collection to count the number of times a page has been visited. You'll learn more in Chapter 8.

The Page Class

Property	Description
Request	This refers to an HttpRequest object that contains information about the current web request. You can use the HttpRequest object to get information about the user's browser, although you'll probably prefer to leave these details to ASP.NET. You'll use the HttpRequest object to transmit information from one page to another with the query string in Chapter 8.
Response	This refers to an HttpResponse object that represents the response ASP.NET will send to the user's browser. You'll use the HttpResponse object to create cookies in Chapter 8, and you'll see how it allows you to redirect the user to a different web page later in this chapter.
Session	This collection holds information for a single user, so it can be used in different pages. For example, you can use the Session collection to store the items in the current user's shopping basket on an e-commerce website. You'll learn more in Chapter 8.
Cache	This collection allows you to store objects that are time-consuming to create so they can be reused in other pages or for other clients. This technique, when implemented properly, can improve performance of your web pages. Chapter 23 discusses caching in detail.
User	If the user has been authenticated, this property will be initialized with user information. Chapter 19 describes this property in more detail.

The Page Class

Sending the User to a New Page

- There are several ways to transfer a user from one page to another.
  - Use an ordinary <a> anchor element:

  Click here to go to newpage.aspx.

  - Use code:
    - First we need a control that causes the page to be posted back.

    Use event handler that reacts to the ServerClick event of a control such as HtmlInputButton or HtmlAnchor.

    - When the page is posted back and your event handler runs, you can use the a method to send the user to the new page.

      HttpResponse.Redirect() method

The Page Class

When you use the Redirect() method, ASP.NET immediately stops processing the page and sends a redirect message back to the browser.

You can also send the user to another website using an absolute URL (a URL that starts with http://).

```
Response.Redirect("http://www.prosetech.com");
```

- Use the HttpServerUtility.Transfer() method:

```
Server.Transfer("newpage.aspx") ;
```

The advantage of using the Transfer() method is the fact that it doesn't involve the browser.

Instead of sending a redirect message back to the browser, ASP.NET simply starts processing the new page as though the user had originally requested that page.

The Page Class

HTML Encoding

- There are certain characters that have a special meaning in HTML.

  Ex: <> are used to create tags.

  Input: Enter a word <here>

  Output: Enter a word (<here> is treated as tag)

- To avoid this problem HTML uses Encoded Special Characters.

Result	Description	Encoded Entity
	Nonbreaking space	
<	Less-than symbol	&lt;
>	Greater-than symbol	&gt;
&	Ampersand	&amp;
"	Quotation mark	&quot;

The Page Class

HTML Encoding

- Use the HttpServerUtility.HtmlEncode() method to replace the special characters.

  Ex: Replace < eith &lt anf > with &gt

  Input:

  ctrl.InnerHtml = Server.HtmlEncode("Enter a word <here>");

  Output:

  "Enter a word <here>“ (in HTML file)

  Enter a word <here> (in Browser window)

Application Events

- Application events are used to perform additional processing tasks.
- Basic ASP.NET features like session state and authentication use application events to plug into the ASP.NET processing pipeline.
- To handle application events you need the help of another ingredient: the global.asax file.

The global.asax File

- The global.asax file allows you to write code that responds to global application events.
- These events fire at various points during the lifetime of a web application, including when the application domain is first created.
- To add a global.asax file to an application in Visual Studio, choose Website ➤ Add New Item, and select the Global Application Class file type. Then, click OK.

Application Events

- It contains event handlers that respond to application events.

```
<%@ Application Language = "C#" %>
<script language = "c#" runat = "server">
protected void Application_EndRequest(object sender, EventArgs e)
{
        Response.Write(" < hr />This page was served at " +
        DateTime.Now.ToString());
}
</script>
```

Additional Application Events

- Application.EndRequest is only one of more than a dozen events you can respond to in your code.
- To create a different event handler, you simply need to create a subroutine with the defined name.

Application Events

Event Handling Method	Description
Application_Start()	Occurs when the application starts, which is the first time it receives a request from any user. It doesn't occur on subsequent requests. This event is commonly used to create or cache some initial information that will be reused later.
Application_End()	Occurs when the application is shutting down, generally because the web server is being restarted. You can insert cleanup code here.
Application_BeginRequest()	Occurs with each request the application receives, just before the page code is executed.
Application_EndRequest()	Occurs with each request the application receives, just after the page code is executed.
Session_Start()	Occurs whenever a new user request is received and a session is started. Sessions are discussed in detail in Chapter 8.
Session_End()	Occurs when a session times out or is programmatically ended. This event is only raised if you are using in-process session state storage (the InProc mode, not the StateServer or SQLServer modes).
Application_Error()	Occurs in response to an unhandled error. You can find more information about error handling in Chapter 7.

ASP.NET Configuration

- Every web application includes a web.config file.

The ASP.NET configuration files have several key advantages:

- **They are never locked:** You can update web.config settings at any point, even while your application is running.
- **They are easily accessed and replicated:** Provided you have the appropriate network rights, you can change a web.config file from a remote computer. You can also copy the web.config file and use it to apply identical settings to another application or another web server that runs the same application in a web farm scenario
- **The settings are easy to edit and understand:** The settings in the web.config file are human-readable, which means they can be edited and understood without needing a special configuration tool.

ASP.NET Configuration

The web.config File

- The web.config file uses a predefined XML format.
- The entire content of the file is nested in a root <configuration> element.

```
<?xml version="1.0" ?>
<configuration>
        <appSettings>…</appSettings>
        <connectionStrings>…</connectionStrings>
        <system.web>…</system.web>
</configuration>
```

- Note that the web.config file is case-sensitive, like all XML documents, and starts every setting with a lowercase letter. This means you cannot write <AppSettings> instead of <appSettings>.

ASP.NET Configuration

- There are three sections in the web.config file.
  - The <appSettings> section allows you to add your own miscellaneous pieces of information.
  - The <connectionStrings> section allows you to define the connection information for accessing a database.
  - The <system.web> section holds every ASP.NET setting you'll need to configure.

<appSettings>

- When you create a new website, there's just one element in the <system.web> section, named <compilation>:

ASP.NET Configuration

```
<?xml version="1.0" ?>
<configuration>
        <system.web>
                <compilation debug="true" targetFramework="4.0">
                </compilation>
        </system.web>
</configuration>
```

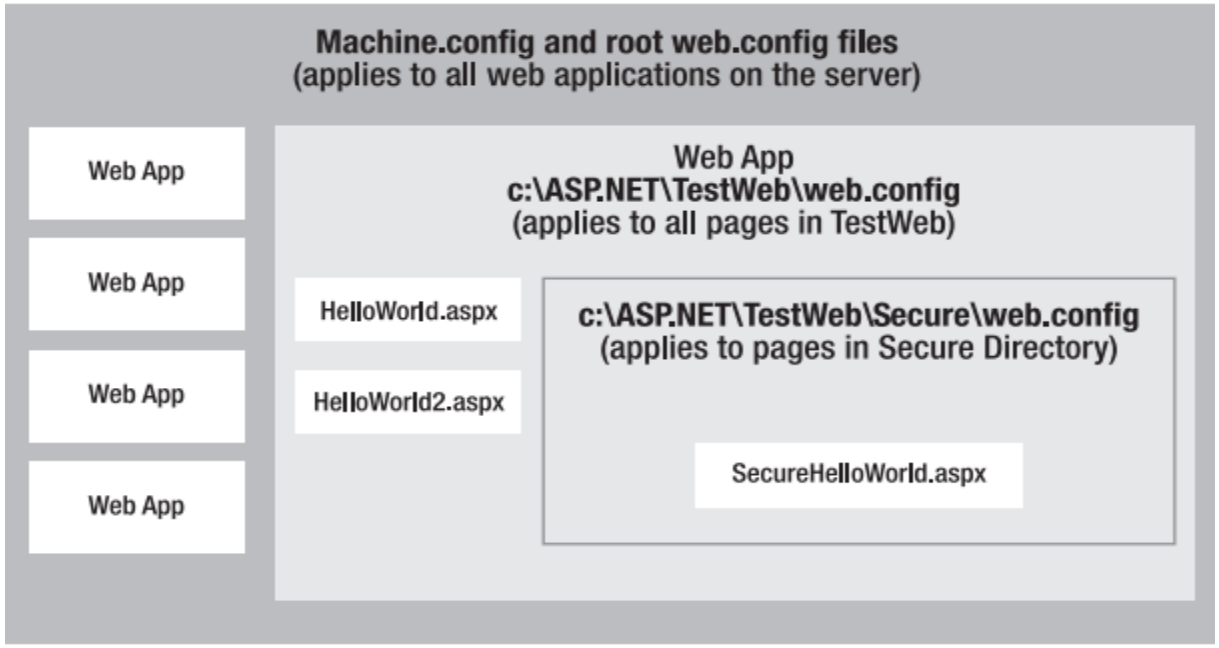
The <compilation> element specifies two settings using two attributes:

- **debug:** This attribute tells ASP.NET whether to compile the application in debug mode so you can use Visual Studio's debugging tools.
- **targetFramework:** This attribute tells Visual Studio what version of .NET you're planning to use on your web server.

ASP.NET Configuration

Nested Configuration

- ASP.NET uses a multi-layered configuration system that allows you to set settings at different levels.
- The Config folder contains two files, named machine.config and web.config.



ASP.NET Configuration

Storing Custom Settings in the web.config File

- ASP.NET also allows you to store your own settings in the web.config file, in an element called <appSettings>.

```
<?xml version="1.0" ?>
<configuration>
        <appSettings>
                <!-- Custom application settings go here. -->
        </appSettings>
        <system.web>
                <!-- ASP.NET Configuration sections go here. -->
         </system.web>
</configuration>
```

- You can enter custom settings using an <add> element that identifies a unique variable name (key) and the variable contents (value).

ASP.NET Configuration

```
<appSettings>
  <add key="DataFilePath"  value="e:\NetworkShare\Documents\WebApp\Shared" />
</appSettings>
```

Reasons to do special settings in web.config

- To centralize an important setting that needs to be used in many different pages.
- To make it easy to quickly switch between different modes of operation.
- To set some initial values.

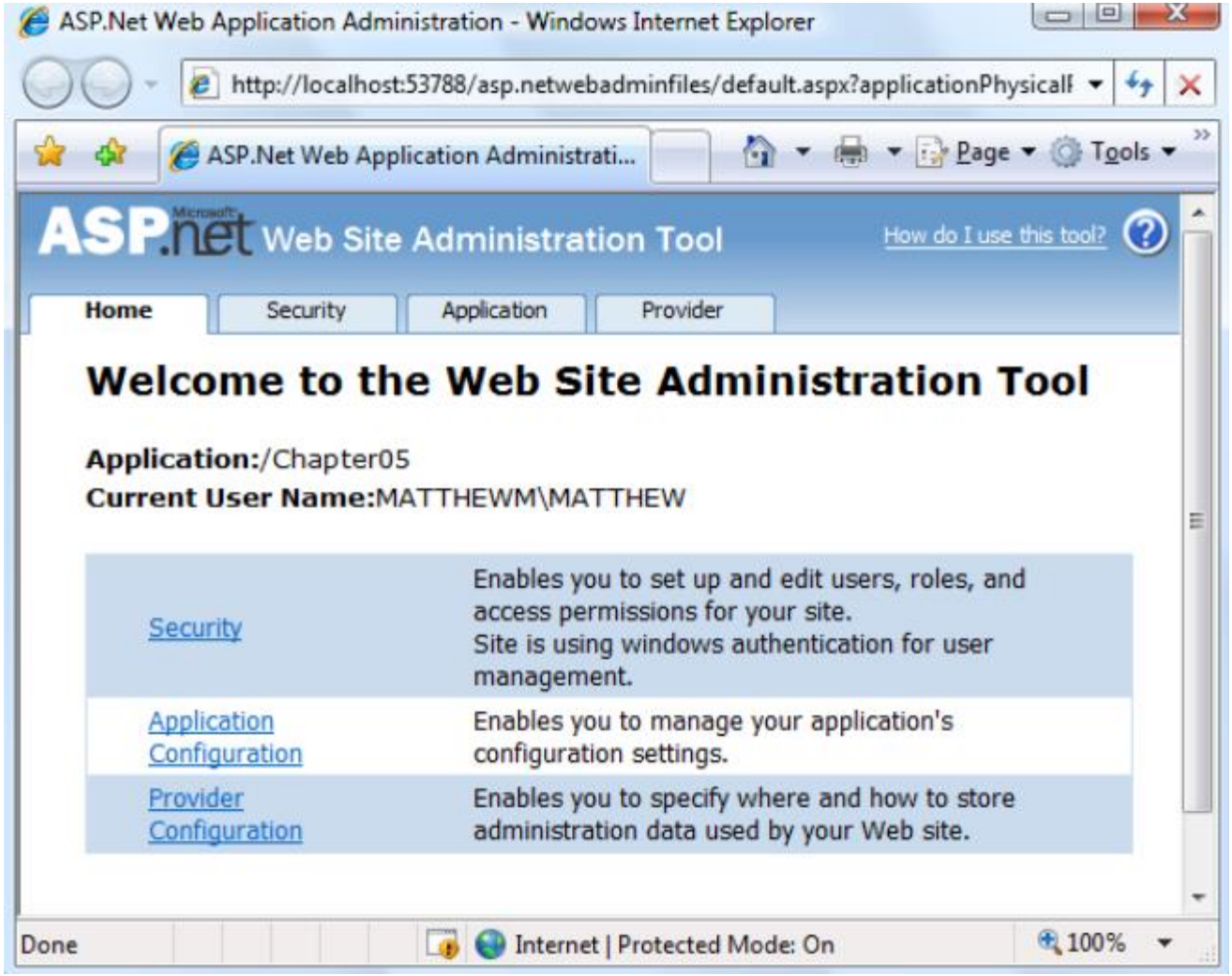
ASP.NET is configured, by default, to deny any requests for .config files. That means remote users will not be able to access the file.

ASP.NET Configuration

The Website Administration Tool (WAT)

- To edit web.config file, ASP.NET includes a graphical configuration tool called the **Website Administration Tool (WAT).**
- You configure various parts of the web.config file using a web page interface by WAT.
- To run the WAT to configure the current web project in Visual Studio, select Website ➤ ASP.NET Configuration.
- Using Application tab, you can create a new setting.
- The WAT generates the settings you need and adds them to the web.config file for your application behind the scenes.

ASP.NET Configuration



END OF LECTURE