Web Form Fundamentals

Today You Will Learn

- The Anatomy of an ASP.NET Application
- Introducing Server Controls
- Improving the Currency Converter

The Anatomy of an ASP.NET Application

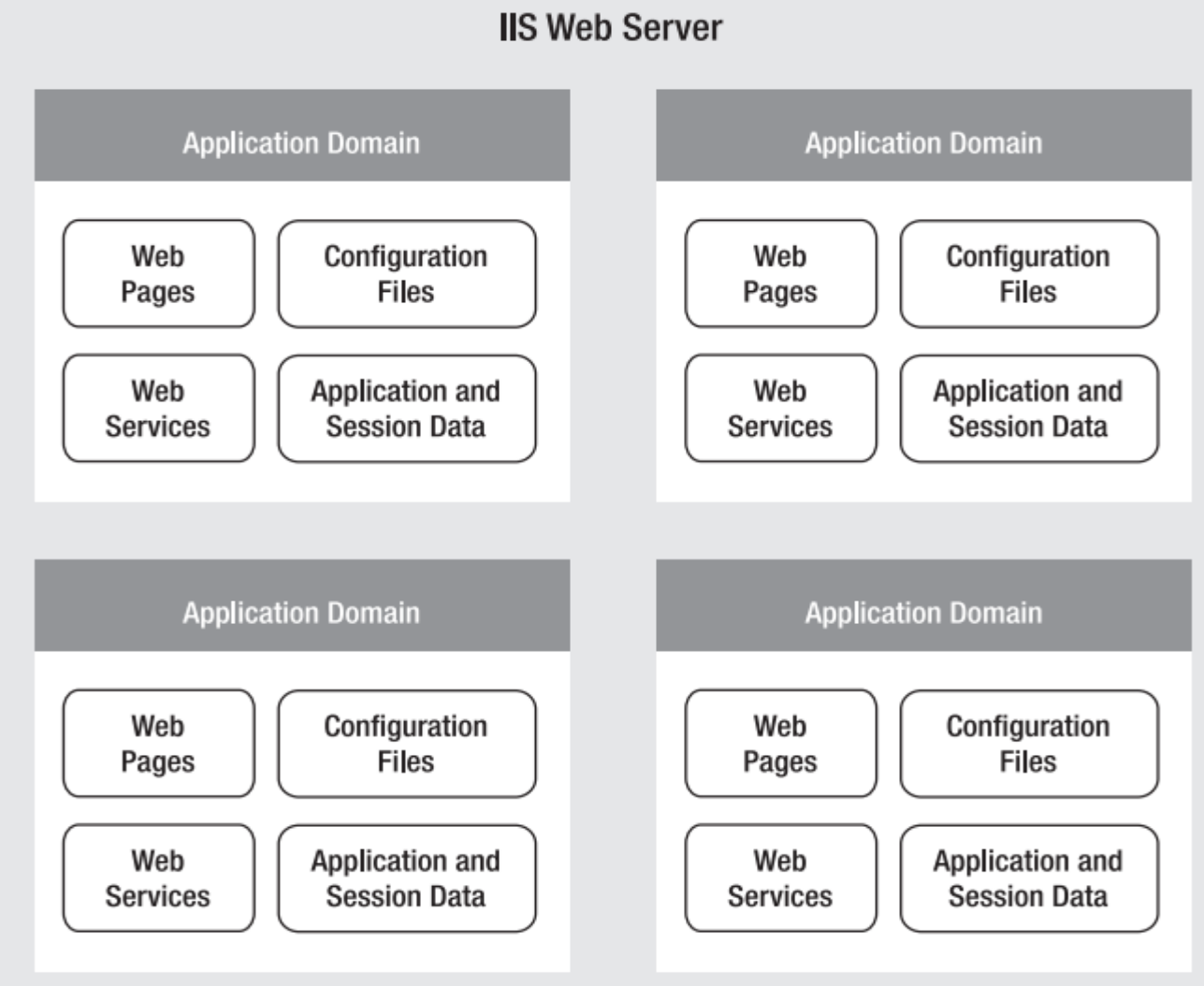
- ASP.NET applications are almost always divided into multiple web pages.
- Web pages from other ASP.NET applications don't share these resources, even if they're on the same web server.
- Every ASP.NET application is executed inside a separate application domain.
- Application domains are isolated areas in memory, and they ensure that even if one web application causes a fatal error, it's unlikely to affect any other application that is currently running on the same computer.

The Anatomy of an ASP.NET Application

- An ASP.NET application is a combination of files, pages, handlers, modules, and executable code that can be invoked from a virtual directory (and, optionally, its subdirectories) on a web server.

- The virtual directory is the basic grouping structure that delimits an application.

The Anatomy of an ASP.NET Application



The Anatomy of an ASP.NET Application

ASP.NET File Types

File Name	Description
Ends with .aspx	These are ASP.NET web pages. They contain the user interface and, optionally, the underlying application code. Users request or navigate directly to one of these pages to start your web application.
Ends with .ascx	These are ASP.NET user controls. User controls are similar to web pages, except that the user can't access these files directly. Instead, they must be hosted inside an ASP.NET web page. User controls allow you to develop a small piece of user interface and reuse it in as many web forms as you want without repetitive code. You'll learn about user controls in Chapter 11.
web.config	This is the configuration file for your ASP.NET application. It includes settings for customizing security, state management, memory management, and much more. You'll get an introduction to the web.config file in this chapter, and you'll explore its settings throughout this book.
global.asax	This is the global application file. You can use this file to define global variables (variables that can be accessed from any web page in the web application) and react to global events (such as when a web application first starts). You'll learn about it later in this chapter.
Ends with .cs	These are code-behind files that contain C# code. They allow you to separate the application logic from the user interface of a web page. We'll introduce the code-behind model in this chapter and use it extensively in this book.

The Anatomy of an ASP.NET Application

ASP.NET Application Directories

- Along with the directories you create, ASP.NET also uses a few specialized subfolders.
- You won't see all these subfolders in a typical application.

Directory	Description
App_Browsers	Contains .browser files that ASP.NET uses to identify the browsers that are using your application and determine their capabilities. Usually, browser information is standardized across the entire web server, and you don't need to use this folder. For more information about ASP.NET's browser support—which is an advanced features that most ordinary web developers can safely ignore—refer to *Pro ASP.NET 4 in VB 2010* (Apress).
App_Code	Contains source code files that are dynamically compiled for use in your application.
App_GlobalResources	Stores global resources that are accessible to every page in the web application. This directory is used in localization scenarios, when you need to have a website in more than one language. Localization isn't covered in this book, although you can refer to *Pro ASP.NET 4 in VB 2010* (Apress) for more information.

The Anatomy of an ASP.NET Application

Directory	Description
App_LocalResources	Serves the same purpose as App_GlobalResources, except these resources are accessible to a specific page only.
App_WebReferences	Stores references to web services, which are remote code routines that a web application can call over a network or the Internet.
App_Data	Stores data, including SQL Server Express database files (as you'll see in Chapter 14). Of course, you're free to store data files in other directories.
App_Themes	Stores the themes that are used to standardize and reuse formatting in your web application. You'll learn about themes in Chapter 12.
Bin	Contains all the compiled .NET components (DLLs) that the ASP.NET web application uses. For example, if you develop a custom component for accessing a database (see Chapter 22), you'll place the component here. ASP.NET will automatically detect the assembly, and any page in the web application will be able to use it. This seamless deployment model is far easier than working with traditional COM components, which must be registered before they can be used (and often reregistered when they change).

Introducing Server Controls

- In old-style web development, programmers had to master the quirks and details of HTML before they could design a dynamic web page.
- Pages had to be carefully tailored to a specific task, and the only way to generate additional content was to generate raw HTML tags.
- ASP.NET solves this problem with a higher-level model of server controls.
- These controls created and configured as an objects.

Introducing Server Controls

- ASP.NET provides two set of server-side controls:
  - **HTML server controls:** These are server-based equivalents for standard HTML elements. So, we work on familiar HTML tags. They are useful when migrating ordinary HTML pages or ASP pages to ASP.NET, because they require the fewest changes.
  - **Web Controls:** They provide a richer object model with a variety of properties for style, formatting details, more events, more closely resemble the controls used for Windows development. Web controls also feature some user interface elements that have no direct HTML equivalent, such as the GridView, Calendar and validation controls.

Introducing Server Controls

HTML Server Controls

- HTML server controls provide an object interface for standard HTML elements.

  They provide three key features:

  - **They generate their own interface:** You set properties in code, and the underlying HTML tag is created automatically when the page is rendered and sent to the client.
  - **They retain their state:** Because the Web is stateless, ordinary web pages need to do a lot of work to store information between requests. HTML server controls handle this task automatically.

Introducing Server Controls

HTML Server Controls

Ex: if the user selects an item in a list box, that item remains selected the next time the page is generated.

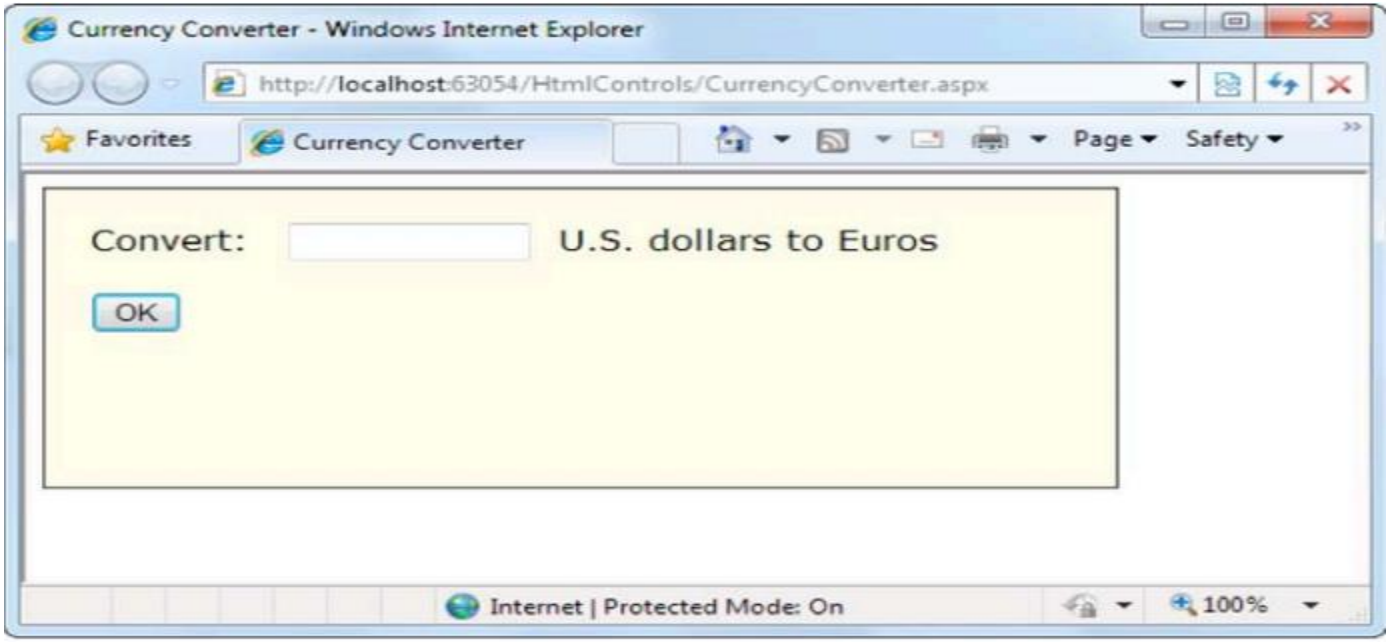
- **They fire server-side events:** With event-based programming, you can easily respond to individual user actions and create more structured code.

  Ex: Buttons fire an event when clicked, text boxes fire an event when the text they contain is modified, and so on.

HTML server controls are ideal when you're performing a quick translation to add server-side code to an existing HTML page.

Introducing Server Controls

Converting an HTML Page to an ASP.NET Page



- It is only HTML page, not a web form.

Introducing Server Controls

Converting an HTML Page to an ASP.NET Page

- It is a just HTML page to convert Dollar to Euro. When you click OK button nothing will happens.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
 "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Currency Converter</title>
 </head>
 <body>
  <form method="post">
   <div>
    Convert: 
    <input type="text" />
     U.S. dollars to Euros.
    <br /><br />
    <input type="submit" value="OK" />
   </div>
  </form>
 </body>
</html>
```

(There are two inputs text and button)
(tag is used for border)
(is used for adding extra space)

Introducing Server Controls

Converting an HTML Page to an ASP.NET Page

```
<%@ Page Language="C#" AutoEventWireup="false"
   CodeFile="CurrencyConverter.aspx.vb" Inherits="CurrencyConverter" %>
 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
 "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Currency Converter</title>
 </head>
 <body>
  <form method="post">
   <div>
    Convert:  
    <input type="text" />
      U.S. dollars to Euros.
    <br /><br />
    <input type="submit" value="OK" />
   </div>
  </form>
 </body>
</html>
```

Save this in .apsx file. Added page directive.

Introducing Server Controls

Converting an HTML Page to an ASP.NET Page

- Just add the attribute runat="server" to each tag that you want to transform into a server control.
- You should also add an ID attribute to each control that you need to interact with in code. The ID attribute assigns the unique name that you'll use to refer to the control in code.
- The <form> element must be processed as a server control to allow ASP.NET to access the controls it contains.

Introducing Server Controls

Converting an HTML Page to an ASP.NET Page

```
<%@ Page Language="C#" AutoEventWireup="false"
  CodeFile="CurrencyConverter.aspx.vb" Inherits="CurrencyConverter" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
 "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Currency Converter</title>
 </head>
 <body>
  <form runat="server">
   <div>
    Convert:  
    <input type="text" ID="US" runat="server" />
      U.S. dollars to Euros.
    <br /><br />
    <input type="submit" value="OK" ID="Convert" runat="server" />
   </div>
  </form>
 </body>
</html>
```

Introducing Server Controls

View State

- Now the static HTML elements converted to HTML server controls.
- By pressing F5, the website will run.
- Select View ➤ Source in your browser to look at the HTML that ASP.NET sent your way.
- Runat is removed, additional hidden field

```
<form id="form1" method="post" action="CurrencyConverter.aspx">
    <div>
     <input type="hidden" ID="__VIEWSTATE" name="__VIEWSTATE"
      value="dDw3NDg2NTI5MDg7Oz4=" />
    </div>
    <div>
     Convert:  
     <input type="text" ID="US" name="US" />
       U.S. dollars to Euros.
     <br /><br />
     <input type="submit" value="OK" ID="Convert" name="Convert" />
    </div>
</form>
```

Introducing Server Controls

View State

- If you enter information in the text box and click the submit button to post the page, the refreshed page will still contain the value you entered in the text box. (In the original example that uses ordinary HTML elements, the value will be cleared every time the page is submitted.) This change occurs because ASP.NET controls automatically retain their state.

Introducing Server Controls

The HTML Control Classes

- All the HTML server controls are defined in the System.Web.UI.HtmlControls namespace.
- Each kind of control has a separate class.

Class Name	HTML Element	Description
HtmlForm	<form>	Wraps all the controls on a web page. All ASP.NET server controls must be placed inside an HtmlForm control so that they can send their data to the server when the page is submitted. Visual Studio adds the <form> element to all new pages. When designing a web page, you need to ensure that every other control you add is placed inside the <form> section.

Introducing Server Controls

Class Name	HTML Element	Description
HtmlAnchor	<a>	A hyperlink that the user clicks to jump to another page.
HtmlImage	<img>	A link that points to an image, which will be inserted into the web page at the current location.
HtmlTable, HtmlTableRow, and HtmlTableCell	<table>, <tr>, <th>, and <td>	A table that displays multiple rows and columns of static text.
HtmlInputButton, HtmlInputSubmit, and HtmlInputReset	<input type="button">, <input type="submit">, and <input type="reset">	A button that the user clicks to perform an action (HtmlInputButton), submit the page (HtmlInputSubmit), or clear all the user-supplied values in all the controls (HtmlInputReset).
HtmlButton	<button>	A button that the user clicks to perform an action. This is not supported by all browsers, so HtmlInputButton is usually used instead. The key difference is that the HtmlButton is a container element. As a result, you can insert just about anything inside it, including text and pictures. The HtmlInputButton, on the other hand, is strictly text-only.

Introducing Server Controls

HtmlInputCheckBox	<input type="checkbox">	A check box that the user can check or clear. Doesn't include any text of its own.
HtmlInputRadioButton	<input type="radio">	A radio button that can be selected in a group. Doesn't include any text of its own.
HtmlInputText and HtmlInputPassword	<input type="text"> and <input type="password">	A single-line text box where the user can enter information. Can also be displayed as a password field (which displays bullets instead of characters to hide the user input).
HtmlTextArea	<textarea>	A large text box where the user can type multiple lines of text.
HtmlInputImage	<input type="image">	Similar to the <img> tag, but inserts a "clickable" image that submits the page. Using server-side code, you can determine exactly where the user clicked in the image—a technique you'll consider later in this chapter.
HtmlInputFile	<input type="file">	A Browse button and text box that can be used to upload a file to your web server, as described in Chapter 17.
HtmlInputHidden	<input type="hidden">	Contains text information that will be sent to the server when the page is posted back but won't be visible in the browser.

Introducing Server Controls

Class Name	HTML Element	Description
HtmlSelect	<select>	A drop-down or regular list box where the user can select an item.
HtmlHead and HtmlTitle	<head> and <title>	Represents the header information for the page, which includes information about the page that isn't actually displayed in the page, such as search keywords and the web page title. These are the only HTML server controls that aren't placed in the <form> section.
HtmlGenericControl	Any other HTML element.	This control can represent a variety of HTML elements that don't have dedicated control classes. For example, if you add the runat="server" attribute to a element, it's provided to your code as an HtmlGenericControl object. You can identify the type of element by reading the TagName property, which stores a string (for example, "div").

Introducing Server Controls

The HTML Control Classes

- There are two ways to add HTML controls:
  - Add HTML controls by hand to the markup in the .aspx file (simply insert the ordinary HTML element, and add the runat="server" attribute).
  - Drag the control from the HTML tab of the Toolbox, and drop it onto the design surface of a web page in Visual Studio.

  This approach doesn't work for every HTML server control, because they don't all appear in the HTML tab.

Introducing Server Controls

The HTML Control Classes

Table 5-4. Important HTML Control Properties

Control	Most Important Properties
HtmlAnchor	HRef, Name, Target, Title
HtmlImage	Src, Alt, Align, Border, Width, Height
HtmlInputCheckBox and HtmlInputRadioButton	Checked
HtmlInputText	Value
HtmlTextArea	Value
HtmlInputImage	Src, Alt, Align, Border
HtmlSelect	Items (collection)
HtmlGenericControl	InnerText and InnerHtml

Adding Additional Codes

The HTML Control Classes

- Web forms are event-driven, which means every piece of code acts in response to a specific event.
- Add another control that can display the result of the calculation.(you can use a <p> element named Result. The <p> element is one way to insert a block of formatted text into a web page.)

```
<p style="font-weight: bold" ID="Result" runat="server"></p>
```

Adding Additional Codes

Now your Currency converter web application having four server controls:

- A form (HtmlForm object). This is the only control you do not need to access in your code-behind class.
- An input text box named US (HtmlInputText object).
- A submit button named Convert (HtmlInputButton object).
- A <p> element named Result (HtmlGenericControl object).

Adding Additional Codes

CurrencyConverter.aspx

```
<%@ Page Language="VB" AutoEventWireup="false"
   CodeFile="CurrencyConverter.aspx.vb" Inherits="CurrencyConverter" %>
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Currency Converter</title>
 </head>
 <body>
  <form runat="server">
   <div>
    Convert:  
    <input type="text" ID="US" runat="server" />
      U.S. dollars to Euros.
    <br /><br />
    <input type="submit" value="OK" ID="Convert" runat="server" />
    <br /><br />
    <p style="font-weight: bold" ID="Result" runat="server"></p>    (Used to store result of conversion in paragraph)
   </div>
  </form>
 </body>
</html>
```

Adding Additional Codes

Code-behind class

CurrencyConverter.aspx .cs

```
public partial class CurrencyConverter : System.Web.UI.Page
{
    protected void Convert_ServerClick(Object sender, EventArgs e)
    {
            decimal USAmount = Decimal.Parse(US.Value);
            decimal euroAmount = USAmount * 0.85 M;
            Result.InnerText = USAmount.ToString()+ "U.S. dollars =";
            Result.InnerText + = euroAmount.ToString() + "Euros.";
    }
}
```

Adding Additional Codes

Code-behind class

- The code-behind class is a typical example of an ASP.NET page. You'll notice the following conventions:
  - The page class is defined with the Partial keyword. That's because your class code is merged with another code file that you never see. This extra code, which ASP.NET generates automatically.
  - The page defines a single event handler.
  - The event handler retrieves the value from the textbox,converts to numeric value, then conversion is done.
  - The += operator is used to quickly add information to the end of the label, without replacing the existing text.
  - The event handler uses ToString() to convert the numeric value to text.

Adding Additional Codes

Behind the Scenes with the Currency Converter

- ASP.NET performs following steps when receives a request from the CurrencyConverter.aspx page:
  - First, the request for the page is sent to the web server. When the page is running in Visual Studio, the request is sent to the built-in test server.
  - The web server determines that the .aspx file extension is registered with ASP.NET.
  - If this is the first time a page in this application has been requested, ASP.NET automatically creates the application domain. If already compiled, then ASP.NET will reuse the compiled version of the page.

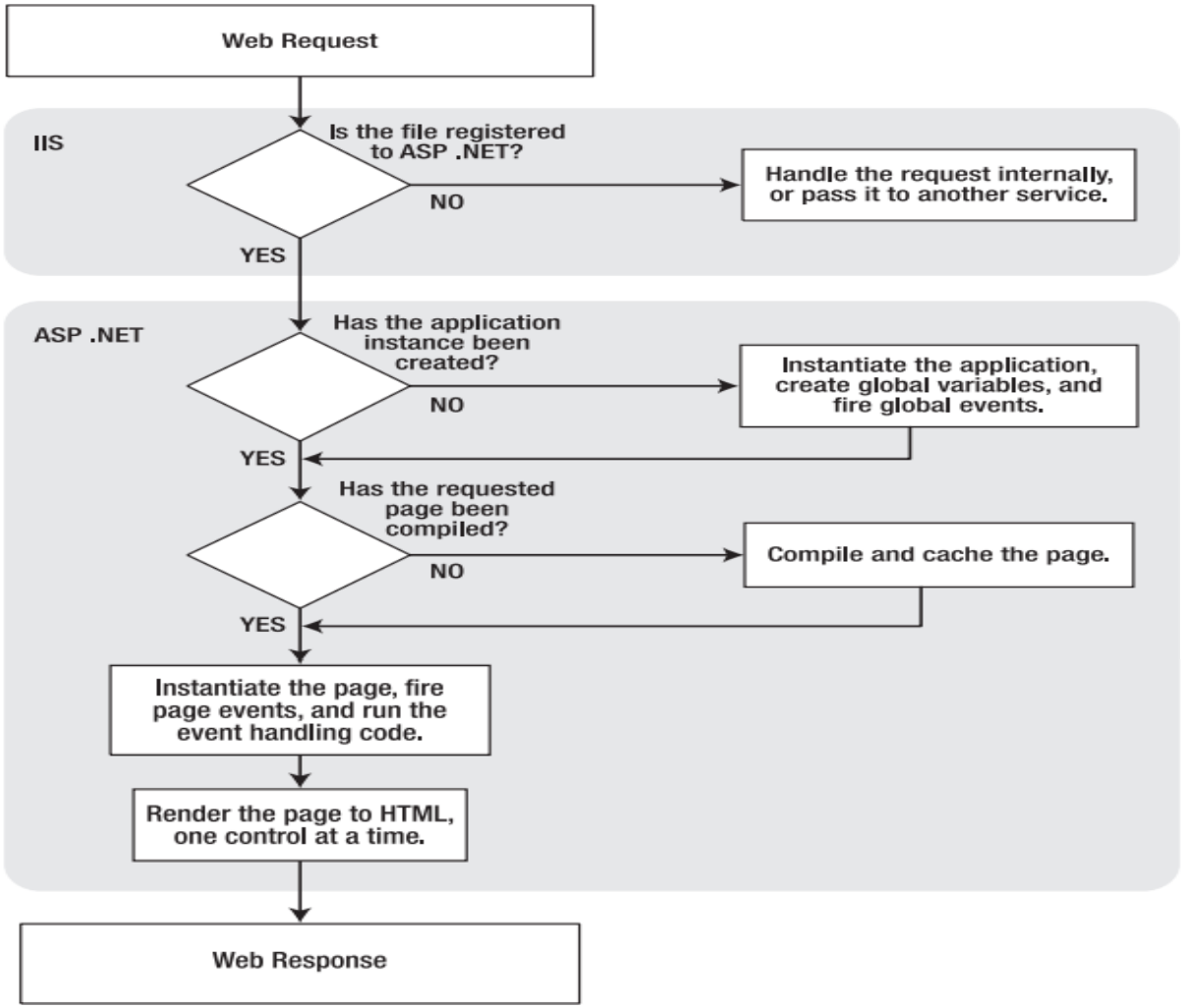
Adding Additional Codes

Behind the Scenes with the Currency Converter

- The compiled.aspx page acts like a miniature program. At this stage, everything is working together as a set of in-memory .NET objects.
- When the code is finished, ASP.NET asks every control in the web page to render itself into the corresponding HTML markup.
- The final page is sent to the user, and the application ends.

The stages in an ASP.NET request

Adding Additional Codes



Improving the Currency Converter

Adding Multiple Currencies

- The first task is to allow the user to choose a destination currency. (use a drop-down list box.)
- In HTML, a drop-down list is represented by a <select> element that contains one or more <option> elements.
- HTML allows you to use only <select> (by giving it an ID and adding the runat="server" attribute) as server tag without <option> tags,, you'll be able to interact with it in code and add the required items when the page loads.
- Add the <select> element to Currencyconverter.aspx page

```
<select ID="Currency" runat="server" />
```

Improving the Currency Converter

Adding Multiple Currencies

- The currency list can now be filled using code at runtime.
- The ideal event is the Page.Load event, which is fired at the beginning of the page processing sequence.

```
protected void Page_Load(Object sender, EventArgs e)
{
        if (this.IsPostBack == false)
        {
                //The HtmlSelect control accepts text or ListItem objects.
                Currency.Items.Add("Euros");
                Currency.Items.Add("Japanese Yen");
                Currency.Items.Add("Canadian Dollars");
        }
}
```

Improving the Currency Converter

Storing Information in the List

- To set the value attribute, you need to create a ListItem object for every item in the list and add that to the HtmlSelect control.
- The ListItem class provides a constructor that lets you specify the text and value at the same time that you create it.

```
protected void Page_Load(Object sender, EventArgs e)
{
        if (this.IsPostBack == false)
        {
                // The HtmlSelect control accepts text or ListItem objects.
                Currency.Items.Add(new ListItem("Euros", "0.85"));
                Currency.Items.Add(new ListItem("Japanese Yen", "110.33"));
                Currency.Items.Add(new ListItem("Canadian Dollars", "1.2"));
        }
}
```

END OF LECTURE