CHAPTER 2: ASP.NET BASICS

BUILD YOUR OWN ASP.NET 3.5 WEB SITE USING C# & VB

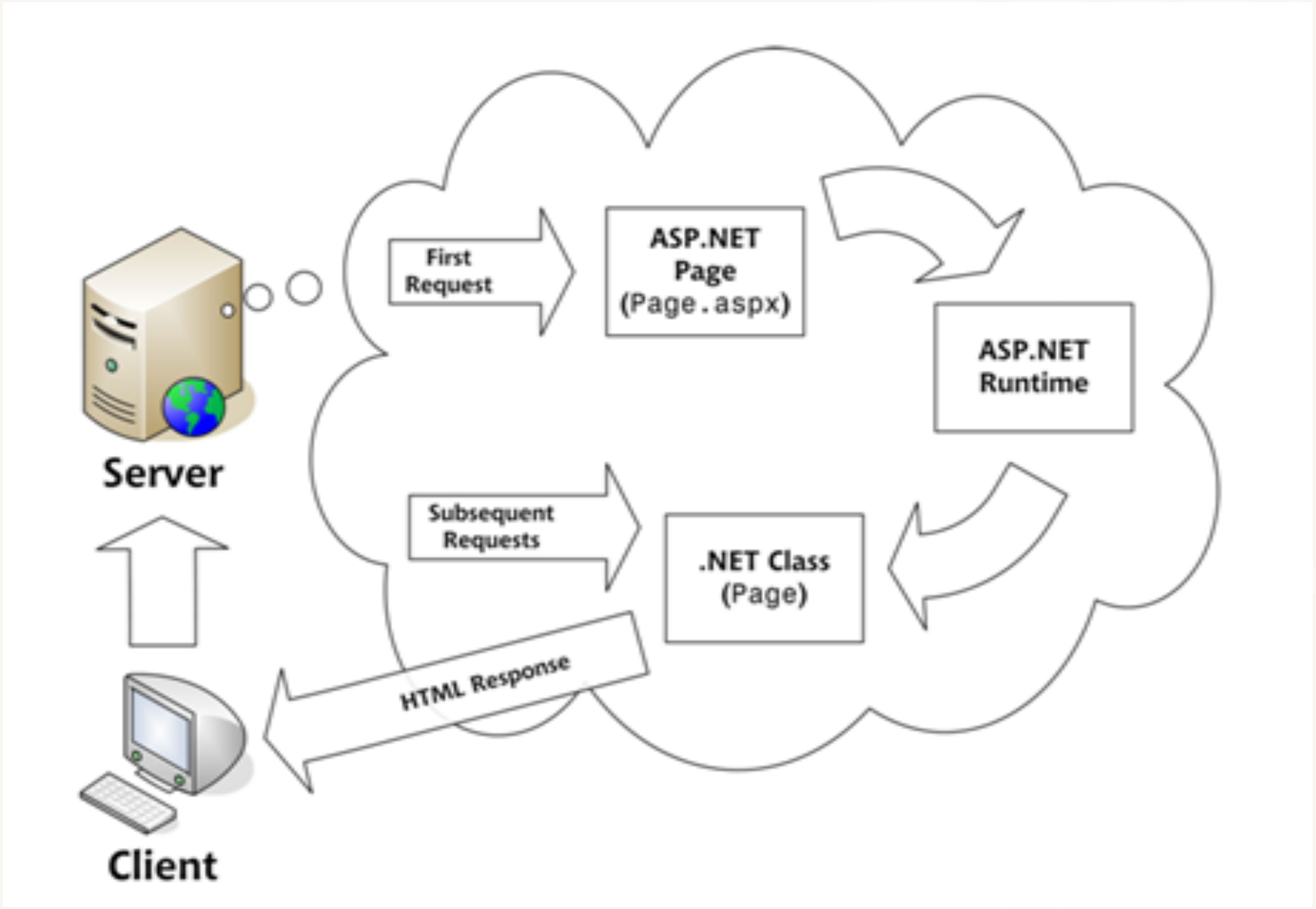
Slides adapted from **Marwa Fouad**

Outlines of today's lecture

So in this chapter, we'll talk about some key mechanisms of an ASP.NET page, specifically:

- page structure
- directives
- namespaces
- view state

The life cycle of the ASP.NET page



ASP.NET Page Structure

- ASP.NET pages are simply text files that have the **.aspx** file name extension, and can be placed on any web server equipped with ASP.NET.

- When a client requests an ASP.NET page, the web server passes the page to the **ASP.NET runtime**, a program that runs on the web server that's responsible for reading the page and compiling it in to a .NET class.

- This class is then used to produce the HTML that's sent back to the user.

- Each subsequent request for this page avoids the compilation process: the .NET class can respond directly to the request, producing the page's HTML and sending it to the client until such time as the .aspx file changes.

ASP.NET Page Structure (Cont.)

An ASP.NET page consists of the following elements:

- directives
- code declaration blocks
- code render blocks
- ASP.NET server controls
- server-side comments
- literal text and HTML tags

```
<%@ Page Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">
  protected void Page_Load(object sender, EventArgs e)
  {
    myTimeLabel.Text = DateTime.Now.ToString();
  }
</script>

<html xmlns="http://www.w3.org/1999/xhtml">
  <head runat="server">
    <title>Welcome to Build Your Own ASP.NET 3.5 Web Site!</title>
  </head>
  <body>
    <form id="form1" runat="server">
    <div>
      <p>Hello there!</p>
      <p>
        The time is now:
        <%-- Display the current date and time --%>
        <asp:Label ID="myTimeLabel" runat="server" />
      </p>
      <p>
        <%-- Declare the title as string and set it --%>
        <% string Title = "This is generated by a code render
            block."; %>
        <%= Title %>
      </p>
    </div>
    </form>
  </body>
</html>
```

```
<%@ Page Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">
  protected void Page_Load(object sender, EventArgs e)
  {
    myTimeLabel.Text = DateTime.Now.ToString();
  }
</script>

<html xmlns="http://www.w3.org/1999/xhtml">
  <head runat="server">
    <title>Welcome to Build Your Own ASP.NET 3.5 Web Site!</title>
  </head>
  <body>
    <form id="form1" runat="server">
    <div>
      <p>Hello there!</p>
      <p>
        The time is now:
        <%-- Display the current date and time --%>
        <asp:Label ID="myTimeLabel" runat="server" />
      </p>
      <p>
        <%-- Declare the title as string and set it --%>
        <% string Title = "This is generated by a code render
            block."; %>
        <%= Title %>
      </p>
    </div>
    </form>
  </body>
</html>
```

1. Directives

- The **directives** section is one of the most important parts of an ASP.NET page.
- They control how a page is compiled, specify how a page is cached by web browsers, aid debugging, and allow to import classes ..etc.
- Each directive starts with<%@ followed by the directive name, plus any attributes and their corresponding values.
- The directive then ends with %>.
- ASP.NET directives can appear anywhere on a page, but they're commonly included at its very **beginning**.

1. Directives (Cont.)

- Import and Page directives are the most useful for ASP.NET development. Looking at our sample ASP.NET page, we can see that a Page directive was used at the top of the page like so:

```
<%@ Page Language="C#" %>
```

- The Page directive specifies the language that's to be used for the application logic by setting the **Language** attribute.

1. Directives–Examples

- **Page directive**
  - Defines page-specific attributes for the ASP.NET page, such as the language used for server-side code.

```
<%@ Page Language="C#" %>
```

- **Import directive**
  - Makes functionality that's been defined elsewhere available in a given page.

```
<%@ Import Namespace="System.Web.Mail" %>
```

  - **Namespaces** are simply .NET's way of keeping all its functionality neatly organized.
- **Register directive**

  Allows you to register a user control for use on your page.

```
<%@ Register TagPrefix="uc" TagName="footer"
Src="footer.ascx" %>
```

```
<%@ Page Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">
  protected void Page_Load(object sender, EventArgs e)
  {
    myTimeLabel.Text = DateTime.Now.ToString();
  }
</script>

<html xmlns="http://www.w3.org/1999/xhtml">
  <head runat="server">
    <title>Welcome to Build Your Own ASP.NET 3.5 Web Site!</title>
  </head>
  <body>
    <form id="form1" runat="server">
    <div>
      <p>Hello there!</p>
      <p>
        The time is now:
        <%-- Display the current date and time --%>
        <asp:Label ID="myTimeLabel" runat="server" />
      </p>
      <p>
        <%-- Declare the title as string and set it --%>
        <% string Title = "This is generated by a code render
            block."; %>
        <%= Title %>
      </p>
    </div>
    </form>
  </body>
</html>
```

2. Code Declaration Blocks

- Code-behind pages let us separate our application logic from an ASP.NET page's HTML. However, if you're not working with code-behind pages, you must use **code declaration blocks** to contain all the application logic of your ASP.NET page. This application logic defines **variables, subroutines, functions,** and more.
- In our sample page, we've placed the code inside<script> tags with the runat="server" attribute, like so:

C# LearningASP\CS\Hello.aspx *(excerpt)*

```
<script runat="server">
  protected void Page_Load(object sender, EventArgs e)
  {
    //set the label text to the current time
    myTimeLabel.Text = DateTime.Now.ToString();
  }
</script>
```

C# Comments

```
//set the label text to the current time
```

- C# code also lets us span a comment over multiple lines if we begin it with /* and end it with*/, as in this example:

```
/*set the label text
to the current time */
```

<script> Tag Attribute

- The<script runat="server"> tag accepts two other attributes: **language** and **src.**
- **language:**

  <script runat="server" language="C#”>

  •If you don’t specify a language within the code declaration block, the ASP.NET Page will use the language provided by the language attribute of the Page directive.
- **src:**

  •The second attribute that’s available to us is src; this lets us specify an external code file for use within the ASP.NET page:

  <script runat="server" language="C#" src="mycodefile.cs">

```
<%@ Page Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">
  protected void Page_Load(object sender, EventArgs e)
  {
    myTimeLabel.Text = DateTime.Now.ToString();
  }
</script>

<html xmlns="http://www.w3.org/1999/xhtml">
  <head runat="server">
    <title>Welcome to Build Your Own ASP.NET 3.5 Web Site!</title>
  </head>
  <body>
    <form id="form1" runat="server">
    <div>
      <p>Hello there!</p>
      <p>
        The time is now:
        <%-- Display the current date and time --%>
        <asp:Label ID="myTimeLabel" runat="server" />
      </p>

      <p>
        <%-- Declare the title as string and set it --%>
        <% string Title = "This is generated by a code render
            block."; %>
        <%= Title %>
      </p>
    </div>
    </form>
  </body>
</html>
```

3. Code Render Blocks

- You can use code render blocks to define **inline code** or **expressions** that will execute when a page is rendered
- Code within a code render block is executed immediately when it is encountered during page rendering. On the other hand, code within a code declaration block (within <script> tags) is executed only when it is called or triggered by user or page interactions.
- There are two types of code render blocks—inline code, and inline expressions—both of which are typically written within the body of the ASP.NET page.
- Inline code render blocks execute one or more statements, between<%and%>

```
<% string Title = "This is generated by a code render block."; %>
```

- Inline expression render blocks are used to display the values of variables and the results of methods on a page.

```
<%= Title %>
```

```
<%@ Page Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">
  protected void Page_Load(object sender, EventArgs e)
  {
    myTimeLabel.Text = DateTime.Now.ToString();
  }
</script>

<html xmlns="http://www.w3.org/1999/xhtml">
  <head runat="server">
    <title>Welcome to Build Your Own ASP.NET 3.5 Web Site!</title>
  </head>
  <body>
    <form id="form1" runat="server">
    <div>
      <p>Hello there!</p>
      <p>
        The time is now:
        <%-- Display the current date and time --%>
        <asp:Label ID="myTimeLabel" runat="server" />
      </p>
      <p>
        <%-- Declare the title as string and set it --%>
        <% string Title = "This is generated by a code render
            block."; %>
        <%= Title %>
      </p>
    </div>
    </form>
  </body>
</html>
```

4. ASP.NET Server Controls

- They represent dynamic elements with which your users can interact.
- There are three basic types of server control: **ASP.NET controls, HTML controls, and web user controls**.
- Usually, an ASP.NET control **must reside within a <form runat="server">** tag in order to function correctly.
- Controls offer the following advantages to ASP.NET developers:
  - They give us the ability to access HTML elements easily from within our code.
  - ASP.NET controls retain their properties thanks to a mechanism called **view state**.
  - With ASP.NET controls, developers are able to separate a page's presentational elements (everything the user sees) from its application logic (the dynamic por- tions of the ASP.NET page).
  - Many ASP.NET controls can be "bound" to the data sources from which they will extract data for display with minimal (if any) coding effort.

```
<%@ Page Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">
  protected void Page_Load(object sender, EventArgs e)
  {
    myTimeLabel.Text = DateTime.Now.ToString();
  }
</script>

<html xmlns="http://www.w3.org/1999/xhtml">
  <head runat="server">
    <title>Welcome to Build Your Own ASP.NET 3.5 Web Site!</title>
  </head>
  <body>
    <form id="form1" runat="server">
    <div>
      <p>Hello there!</p>
      <p>
        The time is now:
        <%-- Display the current date and time --%>
        <asp:Label ID="myTimeLabel" runat="server" />
      </p>

      <p>
        <%-- Declare the title as string and set it --%>
        <% string Title = "This is generated by a code render
            block."; %>
        <%= Title %>
      </p>
    </div>
    </form>
  </body>
</html>
```

5. Server-side Comments

- Server-side comments allow you to include within the page comments or notes that won’t be processed by ASP.NET.
- They use the sequences <%-- and --%>.

```
<%-- Display the current date and time --%>
```

- The difference between ASP.NET comments and HTML comments is that ASP.NET comments are not sent to the client at all; HTML comments are.

- Consider the following example, what would happen?

```
<!–
<% string Title = "This is generated by a code render block."; %>
 <%= Title %>
 -->
```

6. Literal Text and HTML Tags

- The final elements of an ASP.NET page are plain old text and HTML.

- HTML allows the display of the information in your ASP.NET controls and code in a way that’s suitable for users and their browsers.

View State

- ASP.NET controls automatically retain their data when a page is sent to the server in response to an event (such as a user clicking a button).
- Microsoft calls this persistence of data **view state**.

C# LearningASP\CS\ViewState.aspx

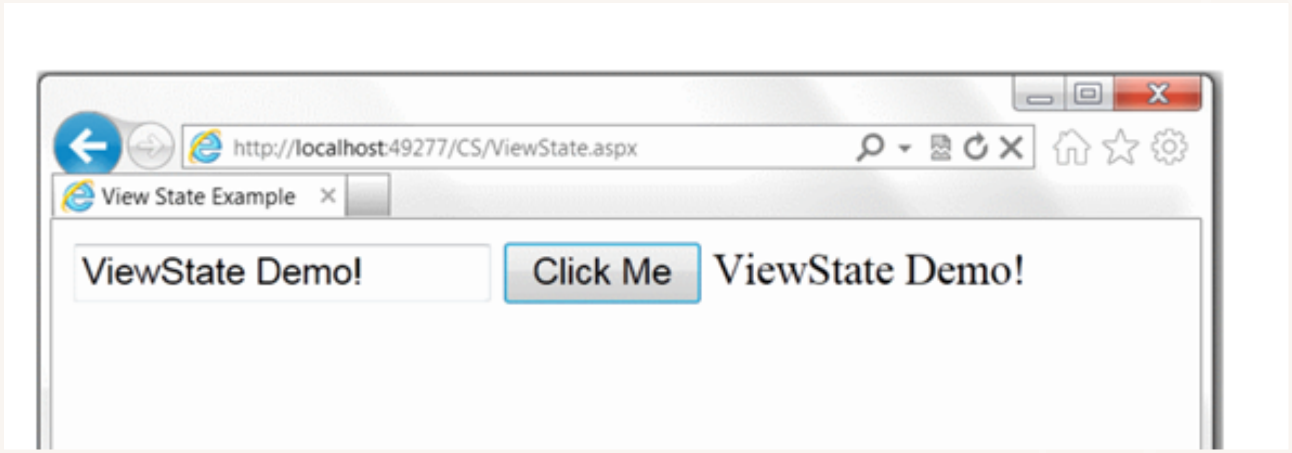
```
<%@ Page Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">
  void Click(Object s, EventArgs e)
  {
    messageLabel.Text = nameTextBox.Text;
  }
</script>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>View State Example</title>
</head>
<body>
  <form id="form1" runat="server">
    <div>
      <asp:TextBox id="nameTextBox" runat="server" />
      <asp:Button id="submitButton" runat="server"
          Text="Click Me" OnClick="Click" />
      <asp:Label id="messageLabel" runat="server" />
    </div>
  </form>
</body>
</html>
```

View State–Example



View State–hidden field

- Where's all that information stored?
- ASP.NET pages maintain view state by encrypting the data within a **hidden form field.**

```
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
    value="/wEPDwUKLTEwNDY1NzgOMQ9…0fMCR+FN5P6v5pkTQwNEl5xhBk" />
```

- This is a standard HTML hidden form field. All information that's relevant to the view state of the page is stored within this hidden form field as an encrypted string.
- View state is enabled for every page by default. If you don't intend to use view state, you can turn it off, which will result in a slight performance gain in your pages. To do this, set the EnableViewState property of the Page directive to false:

```
<%@ Page EnableViewState="False" %>
```