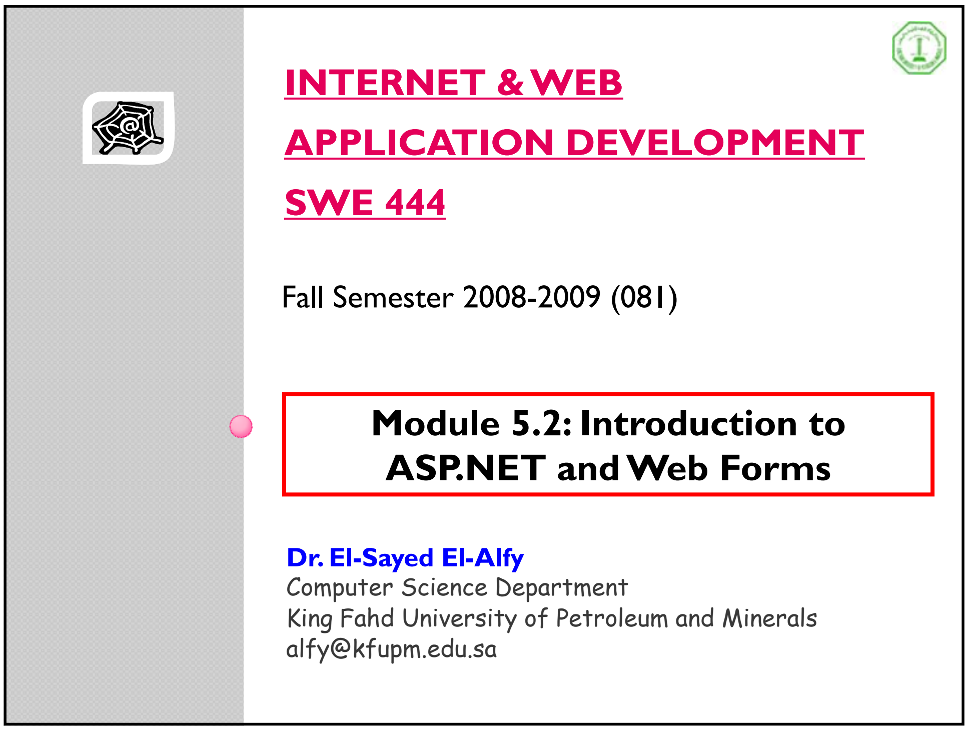
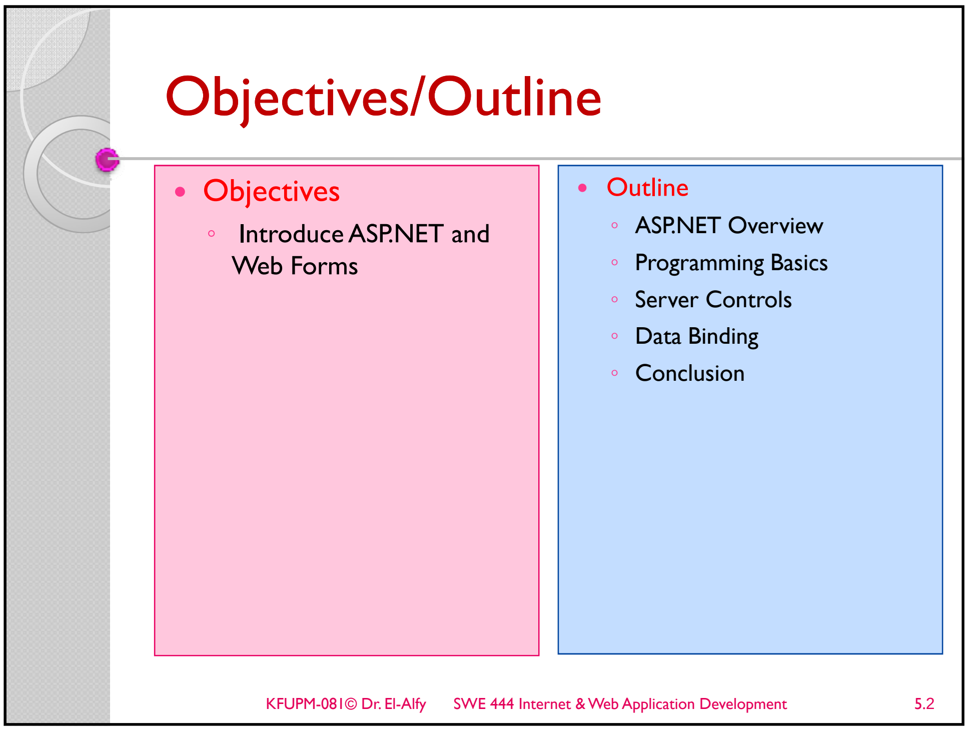


INTERNET & WEB APPLICATION DEVELOPMENT SWE 444

Fall Semester 2008-2009 (081)

Module 5.2: Introduction to ASP.NET and Web Forms

Dr. El-Sayed El-Alfy
Computer Science Department
King Fahd University of Petroleum and Minerals
alfy@kfupm.edu.sa



Objectives/Outline

- Objectives
  - Introduce ASP.NET and Web Forms

- Outline
  - ASP.NET Overview
  - Programming Basics
  - Server Controls
  - Data Binding
  - Conclusion

ASP.NET Overview

- ASP.NET is the next generation ASP, but it's not an upgraded version of ASP
  - Early named ASP+
- Like ASP, ASP.NET is a server-side technology
- Classic ASP restricts developers to using scripting languages (with their inherent limitations)
- ASP.NET provides the most advanced and more flexible Web development platform to date
  - allows the creation, deployment, and execution of Web Applications and Web Services
- With ASP.NET, you can work with any .NET-compliant language, i.e.
  - the code in ASP.NET is compiled for better performance
  - full advantage of advanced language features
- Web Applications are built using Web Forms
- Web Forms are designed to make building Web-based applications easy

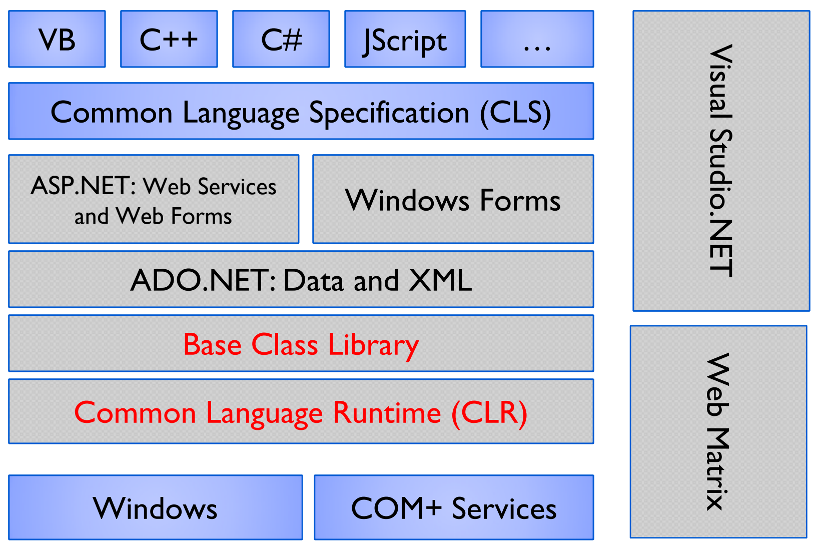
Key Features

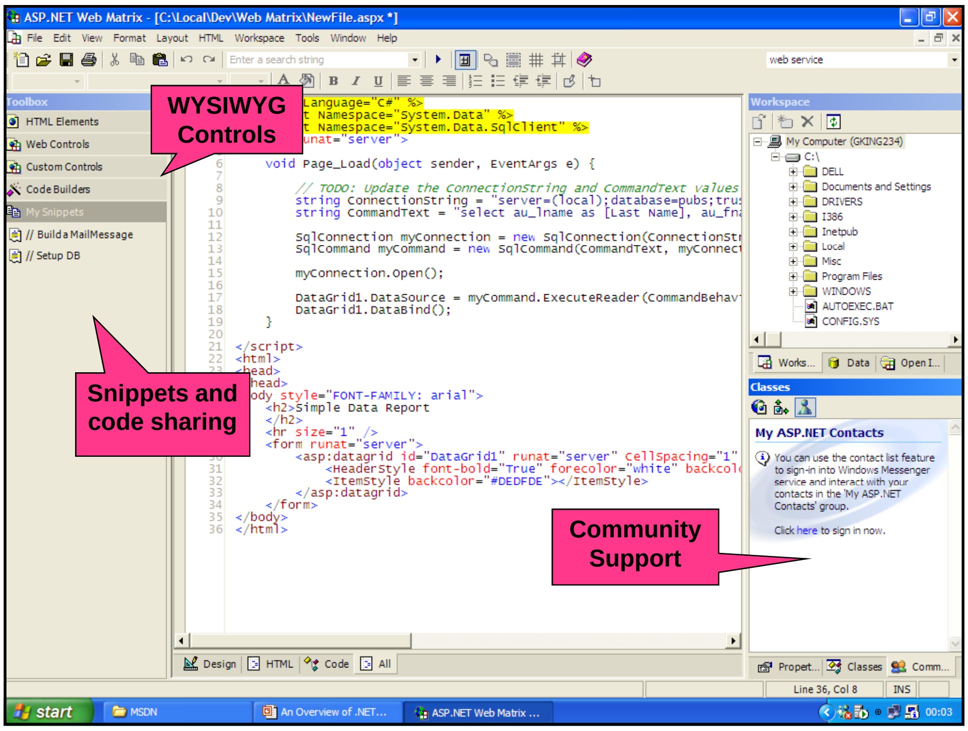
- Built on .NET framework
  - Supports C++, C#, Visual Basic, and JScript (Microsoft's version of JavaScript)
- ASP.NET is designed to run side by side with classic ASP.
  - for the most part you can write an ASP.NET page exactly the same way you would write a classic ASP page.
  - don't need to migrate all of your existing ASP applications at once.
- New programming model of ASP.NET
  - Simpler and combines the best of ASP with the ease of development
  - Separation of code and UI
- ASP.NET includes a powerful new caching engine
  - allow developers to improve the performance of their applications by reducing the Web server and database server processing loads.
- ASP.NET uses a new method of storing configuration information for Web applications
  - Instead of having IIS store this information in a hard-to-access database, it's stored in XML-based human- and machine-readable configuration files,
- State management improvements
  - Providing support for distributing session state across Web servers, persisting state information in a Microsoft SQL Server database, and providing state management without the use of cookies.
- Improved security model in ASP.NET, including new and improved authentication methods, code access security, and role-based authorization
- Built-in support for the ASP.NET Mobile controls
- Simplified form validation

Development Environments

- Visual Studio.NET
  - Installed in college labs and available on CD from PC Admin support
  - Provide simple, faster and unified integrated development environment (IDE) for all of Microsoft's .NET languages and for both Windows, Web applications and Web services
  - http://msdn.microsoft.com/en-us/vstudio/default.aspx
  - http://www.learnvisualstudio.net/
- Visual Web Developer
  - http://www.microsoft.com/express/vwd/
- ASP.NET Web Matrix
  - Lightweight, simple, community-oriented tool for building ASP.NET apps
  - Full WYSIWYG support
    - "What You See Is What You Get"
  - Small (~ 1.4 Mb)
  - Community features
    - IM integration, code sharing, chat features
  - Available free-of-charge at www.asp.net

.NET Platform Architecture





```
Language="C#" %>
t Namespace="System.Data" %>
t Namespace="System.Data.SqlClient" %>
unat="server">

    void Page_Load(object sender, EventArgs e) {

        // TODO: Update the ConnectionString and CommandText values
        string ConnectionString = "server=(local);database=pubs;tru
        string CommandText = "select au_lname as [Last Name], au_fn

        SqlConnection myConnection = new SqlConnection(ConnectionStr
        SqlCommand myCommand = new SqlCommand(CommandText, myConnect

        myConnection.Open();

        DataGrid1.DataSource = myCommand.ExecuteReader(CommandBehav
        DataGrid1.DataBind();
    }

</script>
<html>
<head>
head>
ody style="FONT-FAMILY: arial">
    <h2>Simple Data Report
    </h2>
    <hr size="1" />
    <form runat="server">
        <asp:datagrid id="DataGrid1" runat="server" CellSpacing="1"
            <HeaderStyle font-bold="True" forecolor="white" backcol
            <ItemStyle backcolor="#DEDFDE"></ItemStyle>
        </asp:datagrid>
    </form>
</body>
</html>
```

My ASP.NET Contacts

You can use the contact list feature to sign-in into Windows Messenger service and interact with your contacts in the 'My ASP.NET Contacts' group.

Click here to sign in now.

Programming Basics

- First ASP.NET Example
- Page Syntax
- Server Controls
- Code Blocks
- Data Bind Expressions
- Render Code

Salam.aspx

```
<%@ Page Language="c#" %>
<script runat="server">
    public void B_Click (object sender, System.EventArgs e) {
       Label1.Text = "Salam, the time is " + DateTime.Now;
    }
</script>
<html>
<head>
</head>
<body>
    <form method="post" runat="server">
      <asp:Button id="Button1" onclick="B_Click" runat="server“
            Text="Push Me" name="Button1"></asp:Button>
      <p>
         <asp:Label id="Label1" runat="server"></asp:Label>
       </p>
    </form>
</body>
</html>
```

Page Syntax

- The most basic page is just static text
  - Any HTML page can be renamed .aspx
- Pages may contain:
  - Directives: `<%@ Page Language="C#" %>`
  - Server side comments: `<%-- --%>`
  - Server (web) controls: `<asp:Button runat="server" >`
  - Code blocks:

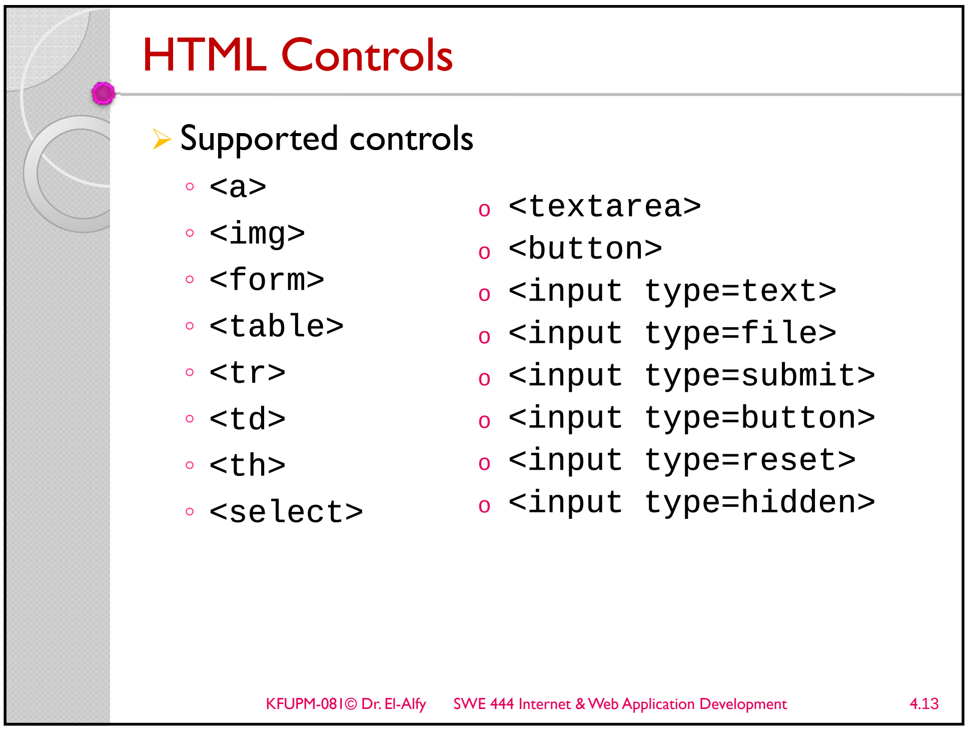
    `<script runat="server">…</script>`
  - Data bind expressions: `<%# %>`
  - Render code: `<%= %>` and `<% %>`
    - Use is discouraged; use `<script runat="server">` with code in event handlers instead

The Page Directive

- Directives are commands used by the compiler when the page is compiled
  - The `Page` directive is most frequently used directive
    - <%@ Page Language="C#" %>
- Only one `Page` directive per .aspx file
- Lets you specify page-specific attributes, e.g.
  - `Language`: Programming language
  - `Inherits`: Base class of Page object
  - `AspCompat`: Compatibility with ASP
  - `CodePage`: Code page for this .aspx page
  - `Trace`: Enables tracing for this page

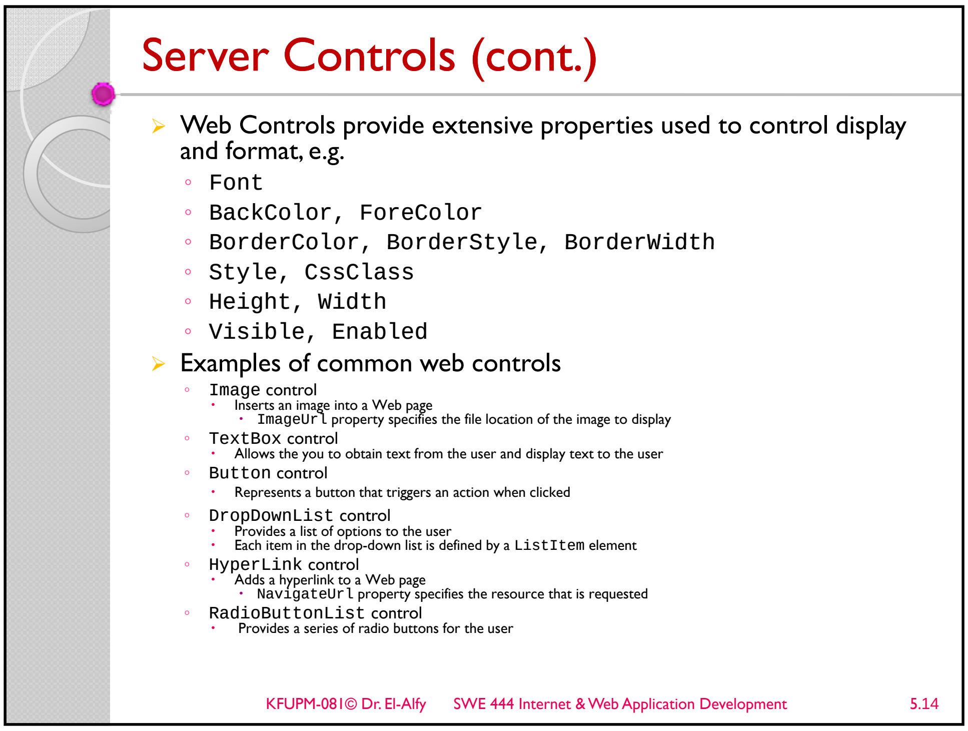
Server Controls

- With Classic ASP it is impossible to separate executable code from the HTML itself.
  - makes the page difficult to read, and difficult to maintain.
- ASP.NET has solved this "spaghetti-code" problem with server controls
- There are three kinds of server controls:
  - HTML Controls - Traditional HTML tags
  - Web Controls - New ASP.NET tags; richer functionality and more consistent object model
  - Validation Controls - For input validation
- ASP.NET contains a large set of HTML controls.
  - Almost all HTML elements on a page can be defined as ASP.NET control objects that can be controlled by scripts.
- ASP.NET also contains a new set of object oriented input controls, like programmable list boxes and validation controls.
- A new data grid control supports sorting, data paging, and everything you expect from a dataset control.
- All ASP.NET objects on a Web page can expose events that can be processed by ASP.NET code.
- Use Server controls when
  - You require a richer set of functionality to perform complicated page requirements
  - Developing pages for multiple browser types



HTML Controls

- Supported controls
  - `<a>`
  - `<img>`
  - `<form>`
  - `<table>`
  - `<tr>`
  - `<td>`
  - `<th>`
  - `<select>`
  - `<textarea>`
  - `<button>`
  - `<input type=text>`
  - `<input type=file>`
  - `<input type=submit>`
  - `<input type=button>`
  - `<input type=reset>`
  - `<input type=hidden>`



Server Controls (cont.)

- Web Controls provide extensive properties used to control display and format, e.g.
  - `Font`
  - `BackColor, ForeColor`
  - `BorderColor, BorderStyle, BorderWidth`
  - `Style, CssClass`
  - `Height, Width`
  - `Visible, Enabled`
- Examples of common web controls
  - `Image` control
    - Inserts an image into a Web page
      - `ImageUrl` property specifies the file location of the image to display
  - `TextBox` control
    - Allows the you to obtain text from the user and display text to the user
  - `Button` control
    - Represents a button that triggers an action when clicked
  - `DropDownList` control
    - Provides a list of options to the user
    - Each item in the drop-down list is defined by a `ListItem` element
  - `HyperLink` control
    - Adds a hyperlink to a Web page
      - `NavigateUrl` property specifies the resource that is requested
  - `RadioButtonList` control
    - Provides a series of radio buttons for the user

Control Syntax

- All server controls must appear within a <form> tag, and the <form> tag must contain the runat="server" attribute
  - The runat="server" attribute indicates that the form should be processed on the server
- Server controls use the runat="server" attribute and id attribute
- Id attribute provides programmatic identifier
  - It names the instance available during postback

```
<input type="text" id="text2" runat="server" />
<asp:calendar id="myCal" runat="server" />
```

- Tag identifies which type of control to create
  - Control is implemented as an ASP.NET class
  - Controls are derived from `System.Web.UI.Control`

Server Control Properties

- Tag attributes map to control properties

```
<asp:button id= "c1" Text="Foo" runat= "server">
<asp:ListBox id="c2" Rows="5" runat="server">
```

- Tags and attributes are case-insensitive
- Control properties can be set programmatically

```
c1.Text = "Foo";
c2.Rows = 5;
```

Maintaining State

- By default, controls maintain their state across multiple postback requests
  - A postback occurs when a page generates an HTML form whose values are posted back to the same page
  - Implemented using a hidden HTML field: `__VIEWSTATE`
  - Works for controls with input data (e.g. `TextBox, CheckBox`), non-input controls (e.g. `Label, DataGrid`), and hybrids (e.g. `DropDownList, ListBox`)
- Can be disabled per control or entire page
  - Set `EnableViewState="false"`
  - Lets you minimize size of `__VIEWSTATE`

Server Code Blocks

- Server code lives in a script block marked `runat="server"`

```
<script language="C#" runat="server">
<script language="VB" runat="server">
<script language="JScript" runat="server">
```

- Script blocks can contain
  - Variables, methods, event handlers, properties
  - They become members of a custom Page object

Page Events

- Pages are structured using events
  - Enables clean code organization
  - Avoids the "Monster IF" statement
  - Less complex than ASP pages
- Code can respond to page events
  - e.g. `Page_Load, Page_Unload`
- Code can respond to control events
  - `Button1_Click`
  - `Textbox1_Changed`

Page Event Lifecycle

Initialize	`Page_Init`
Restore Control State	
Load Page	`Page_Load`
Control Events	
1. Change Events	`Textbox1_Changed`
2. Action Events	`Button1_Click`
Save Control State	
Render	
Unload Page	`Page_Unload`

Page Loading

- Page_Load fires at beginning of request after controls are initialized
  - Input control values already populated

```
protected void Page_Load(Object s, EventArgs e) {
  message.Text = textbox1.Text;
}
```

Page Loading

- Page_Load fires on every request
  - Use Page.IsPostBack to execute conditional logic

```
protected void Page_Load(Object s, EventArgs e) {
  if (! Page.IsPostBack) {
    // Executes only on initial page load
    Message.Text = "initial value";
  }
  // Rest of procedure executes on every request
}
```

Server Control Events

- Action Events
  - Cause an immediate postback to server
  - E.g. `OnClick`
- Change Events
  - By default, these execute only on next action event
    - Use `autopostback="true"` atribute to make them respond directly without waiting for an action event
  - E.g. `OnTextChanged, OnCheckedChanged`
  - Change events fire in random order

Wiring Up Control Events

- Control event handlers are identified on the tag

```
<asp:button onclick="btn1_click" runat="server">
<asp:textbox onchanged="text1_changed" runat="server">
```

- Event handler code

```
protected void btn1_Click(Object s, EventArgs e) {
  Message.Text = "Button1 clicked";
}
```

Event Arguments

- Events pass two arguments:
  - The sender, declared as type object
    - Usually the object representing the control that generated the event
    - Allows you to use the same event handler for multiple controls
  - Arguments, declared as type EventArgs
    - Provides additional data specific to the event
    - `EventArgs` itself contains no data; a class derived from `EventArgs` will be passed

Page Unloading

- `Page_Unload` fires after the page is rendered
  - Don't try to add to output
- Useful for logging and clean up

```
protected void Page_Unload(Object s, EventArgs e) {
  MyApp.LogPageComplete();
}
```

Import Directive

- Adds code namespace reference to page
  - Avoids having to fully qualify .NET types and class names
  - Equivalent to the using directive of C#

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Net" %>
<%@ Import Namespace="System.IO" %>
```

Page Class

- The Page object is always available when handling server-side events
- Provides a large set of useful properties and methods, including:
  - `Application, Cache, Controls, EnableViewState, EnableViewStateMac, ErrorPage, IsPostBack, IsValid, Request, Response, Server, Session, Trace, User, Validators`
  - `DataBind(), LoadControl(), MapPath(), Validate()`

Q & A

?

References

- H. M. Deitel, P. J. Deitel, and A. B. Goldberg, *Internet and World Wide Web How to Program*, 4/e, Pearson Education Inc., 2008.
- Some useful links with examples and other resources:
  - The Official Microsoft ASP.NET Site
    - www.asp.net
  - ASP.NET QuickStart Tutorial
    - http://quickstarts.asp.net/QuickstartV20/aspnet/
  - W3School ASP.NET Tutorial
    - http://www.w3schools.com/ASPNET/default.asp
  - ASP.NET at wikipedia
    - http://en.wikipedia.org/wiki/ASP.NET