University of Salford MANCHESTER

4. Web Applications and ASP.NET

Introduction

- Web Applications are server side applications consisting of one or more Web Forms.
  - Microsoft's previous technology for generating Web Applications was Active Server Pages (now often referred to as ASP Classic)
- ASP.NET is a complete overhaul of traditional Active Server Pages.
- It offers a very different methodology for building Web applications. **ASP.NET is much more powerful**, offering developers a more efficient way to build Web applications.

Introduction

Fundamental Changes from ASP

- Classic ASP was built on top of the Windows operating system and IIS.
- It was always a **separate entity**, and therefore its functionality was limited.
- ASP.NET, on the other hand, is an **integral part of the operating system** under the .NET Framework.
  - It shares many of the same objects that traditional applications would use, and all .NET objects are available for ASP.NET's use.
- ASP made it very clear that client and server were two separate entities.
- Once ASP was finished with its work on the server, it passed the HTML to the client and forgot about it.

Introduction

Fundamental Changes from ASP

- **ASP.NET ties together the client and the server** through clever use of server-side and client-side code, all invisible to the developer.
- **Web development is now much more like traditional application development** than the disconnected request/response model that ASP development typified.
- Furthermore, ASP.NET code is **compiled**, whereas classic ASP used interpreted scripting languages.
  - Using compiled code means an automatic boost in performance over ASP applications.

Introduction

Fundamental Changes from ASP

- **Session state** was a very important concept in classic ASP.
  - It's the ability to automatically determine if a sequence of requests is coming from the same client.
  - Mainly through the use of cookies.
- This session management made it easy to keep track of users and their actions.
  - Easy-to-build shopping carts and data scrolling were born.
- As Web sites began moving to server farms developers began to see the limitation of ASP-based session management.
  - Sessions weren't transferable across servers.
- Session management has become much easier and more powerful with ASP.NET.
- ASP.NET addresses this issue by providing built-in session support that's **scalable across Web farms**.
- It also provides reliability that can even survive server crashes, and it can work with browsers that don't support cookies.

Introduction

Fundamental Changes from ASP

- ASP.NET is now **completely object-oriented**.
- ASP and other traditional dynamic Web page technologies are very different from ASP.NET because they're built using **interpreted** languages, such as VBScript and JavaScript
- **ASP.NET is built using compiled languages**
  - such as Visual Basic and C#
- Using interpreted languages produces Web pages in which the program code and content code are joined.
- **Compiled languages, on the other hand, keep program code and HTML separate.**

Introduction

Fundamental Changes from ASP

- To understand this difference, first you've got to make sure you've got a solid grasp of how traditional technologies like ASP work.
  - Dynamic Web pages that are built using traditional technologies, ASP or PHP, usually contain a few lines of code followed by a few lines of HTML, followed by still more lines of code.
- When code is intermingled with the content in this way. the Web server is forced to do what is called **context switching**.
- For each line in the page, the server must interpret whether it's code or content.
- Then it must compile and run the line if it's code, or output the line to the client's browser if it's content.
- This form of compilation forces developers to write code that's not easily structured and is difficult to reuse.

Introduction

Fundamental Changes from ASP

- In contrast, with ASP.NET, developers produce program code that's kept **separate from the content** on the page.
- When code is separated in this way, a number of advantages emerge:
  - The Web server knows exactly which portion is the code and which is the content, so it can **compile the code in its entirety once**, rather than one line at a time, significantly reducing execution time.
  - The designer doesn't have to look at all the developer's code while laying out the page.
  - The developer won't accidentally alter the design when writing the code.
  - Development and design tools like Visual Studio, Expression Web and Dreamweaver can better serve their users by focusing on the two components of a dynamic page, program code and HTML, separately.

Introduction

Fundamental Changes from ASP

- You can even place the code in a separate file called a **code-behind** file.
- By relocating program code to the code-behind files, your Web page can comprise only content.
- This **full separation** of HTML and program code makes it easier to reuse both program code and the HTML.

Features in ASP.NET

- Besides the complete change in separation between code and content and the resulting way in which the code is compiled, ASP.NET has some significant new features that aren't a part of standard ASP or other traditional Web development environments.

Features in ASP.NET

ASP.NET Controls

- **Server Controls**
  - ASP.NET server controls comprise programmable, **pre-packaged server-side program code** written to perform dynamic functions.
  - These server controls are referenced by tags using a special syntax, <asp:tagname...>, and are then placed within your Web pages, where they execute.
  - When subsequently executed by ASP.NET, **these tags are converted into HTML** and content to be rendered by the user's browser.
  - For example, a simple control "asp:label" represents a server control that displays text using HTML <span...> tags.
  - Server controls vary in the functionality they provide.
  - Standard ASP.NET server controls such as "asp:button" and "asp:textbox" are designed to be used in place of their more traditional counterparts (the HTML <input> form elements of type button and text).

Features in ASP.NET

ASP.NET Controls

- They look the same in Visual Studio and Dreamweaver's Design view and will appear the same in the browser.
- However, although they look the same in Design view, the markup is vastly different.
- Using server controls in place of the traditional HTML controls lets you take advantage of ASP.NET features, such as being able to **dynamically set the control's attributes at run time**.

Validation Controls

- Validation controls are used to **validate the data that an end user enters**, or possibly fails to enter, into form elements.
- For example, a validation control can be used to make a field in an entry form a required field.
- In addition to requiring form fields to be filled, validation controls in ASP.NET can be used to validate the user's input against a range of values or to compare two values

Features in ASP.NET

ASP.NET Controls

- **List Controls**
  - List controls are used to **iterate, process, and display dynamic data**.
  - To associate a list control with data, we **bind (link) dynamic data, such as database query results, to a list control**.
  - Performing the binding operation automatically populates a List control with data from a data source such as an array or a database.
- **Data Controls**
  - **Data access** in ASP.NET can be accomplished completely declaratively (no code) using the new data-bound and data source controls.
  - There are data source controls to represent different data backends such as SQL database, Access database, and XML, and there are data-bound controls for displaying your data, such as gridview, detailsview, listview and formview.

Features in ASP.NET

ASP.NET Controls

- **Rich controls**
  - These are **complex components** that can be placed directly into an ASP.NET Web page.
  - Examples of rich controls include the Calendar and Ad Rotator controls, which display a calendar and a rotating advertisement respectively.
- **Navigation Controls**
  - The navigation controls provide a **common user interface for navigating between pages** in your site, such as treeview, menu, and sitemappath.
- **Login Controls**
  - The new login controls provide the building blocks to add **authentication** and **authorization** to your site, such as login forms, create user forms, password retrieval, and a custom user interface for logged in users or roles.

Features in ASP.NET

ASP.NET Controls

- **Web Part Controls**
  - Web parts are a new family of controls that enable you to **add rich, personalized content and layout** to your site, as well as the ability to **edit that content and layout directly from your application pages**.
- **Master Pages**
  - This feature provides the ability to define **common structure and interface elements for your site**, such as a page header, footer, or navigation bar, in a common location called a “master page”, to be shared by many pages in your site.
- **Themes and Skins**
  - Allows for **easy customization of your site's look-and-feel**.
  - You can define style information in a common location called a "theme", and apply that style information globally to pages or controls in your site.
- **Integrated AJAX Support**
  - ASP.NET now contains AJAX support

Features in ASP.NET

Web Services Integration

- Another new feature of ASP.NET is the **complete integration of Web services**.
- A Web service is, in a sense, a **small program available over the Internet** to which you can call functions and get results.
  - For example, the Postal Office might write a Web service that provides a list of all post codes.
  - From within your Web page, you could call this Web service to get the list of post codes and add that data to a drop-down list box.
  - Visitors to your Web site would then be able to select from that list.
  - If the Postal Office updated the list, your page would automatically be updated as well, since the information isn't being stored locally.
- Web services work by transferring data encoded in XML over a transportation technology, such as HTTP.

Features in ASP.NET

Handling Post Back

- One of the biggest differences between ASP and ASP.NET is that in **ASP.NET a Web Form must post back to itself** rather than post to a different page.
- Historically, developers posted to a different page by setting the form's action attribute.
- Posting to a separate page was a good idea because it made for a cleaner separation of code from HTML.
- Since ASP.NET handles events in the same Web Form in which they're raised, the form must post back to the same page.
  - Even if you set the action attribute of the form to a different page, the Web server finds the runat="server" attribute setting and overrides your action value.
- One of the major advantages in posting back to the same page is that ASP.NET maintains a hidden variable called _VIEWSTATE which holds all the values on your form.
  - This means that when you process a form using post back you do have to worry about maintaining the state of your variables or any data entered on a form, ASP.NET will do this for you.

Features in ASP.NET

Moving Between Pages

- How is it possible to navigate to other pages on your site if you're always posting back to the same Web Form?
- The answer is the **Response.Redirect** command.
  - First, handle the post back in your Web Form.
  - Then give the Response.Redirect command the URL of the next page you want the visitor to go to, like this:
    - Response.Redirect("NextPage.aspx");
  - To increase performance there's a second, optional parameter you can add to the command. It determines whether the server should halt processing the current page and transfer immediately or whether it should finish the page first.
    - Response.Redirect("NextPage.aspx", true);
    - true to halt and redirect immediately, false to wait
- Transferring to a new page is when we really have to start worrying about maintaining state.
  - The __VIEWSTATE hidden form element is not available when transferring between pages using the Response.Redirect command.
  - Any values that need to be passed to another web page should use Session Variables.