The World Wide Web: Web Applications and Web Forms, continued

Web Application Programming

- Simple HTML is ok for “static pages”
  - When there’s no user input and no processing
- But “real” web apps may do many other things
  - May receive input from users on the client side
  - May need to retrieve data from a database
  - May perform computations
  - The HTML they return to browsers will change depending on input and results
  - Client side is easy, but restricted
    - Just use ordinary HTML
    - Or scripts that run on the client side
      - e.g., JavaScript or VBscript
  - What about server side?

Form Tags and the Server Side

- The heart of most real web apps that accept input is the HTML Form tag: <form> … </form>
  - Some fields (lots of them):

    <input type="text" … />
    - Browser renders this tag as a textbox input field

    <input type="submit" … />
    - Browser renders this as a push button
    - When clicked, Form is submitted to the Web Server
    - Textbox input values are also submitted to the Server
    - If there's no method attribute or if form contains a method="get" tag:
      - » Browser sends an HTTP GET command to server with user input appended (e.g., user enters 2 & 3 for text fields named op1 & op2):

        GET /calc.html?op1=2&op2=3 HTTP/1.1
    - If the form has a method="post" attribute:
      - » Form is submitted with an HTTP POST command with user input in the body of the HTTP request

Postback

- When user input from an HTML Form is submitted back to the server, a "postback" has occurred
  - Look at http://localhost/calc.html
    - "View" | "Source" to see html
    - Click "=" & look in browser address bar to see GET postback data
- The Server should respond to the postback by extracting the user input and generating html to display the data and the results
- An important reality: HTML is "stateless"
  - A page stores no information about its contents from one invocation to another
  - So server side code must be running to extract the user input and generate a new web page that displays the desired result
    - … and restore the original data if needed and if it is to be visible

Server Response in calc.html Form

- Calc Form allows user to enter two numbers to be added
- Pressing “=” button submits numbers to server
- Original numbers and sum should be returned to browser
- Server should generate something like the following HTML in response to user entering 2 and 3 and clicking the “=” button:

```
<html>
 <body>
  <form>
   <input type="text" name="op1" value="2" />
   +
   <input type="text" name="op2" value="3" />
   <input type="submit" value="  =  " />
   5
  </form>
 </body>
</html>
```

- Note: generating repeat input values gives illusion user is seeing one Web page when really we’re seeing two pages in succession

Generating the Server Response

- One way:
  - Use the Common Gateway Interface (CGI)
    - A low-level programmatic interface between web servers and applications that run on Web servers
    - A server-side program script that reads inputs, computes, and writes http responses back to browser
    - Usually written in Perl, but can be done in other languages (C)
    - Hard to use, slow, and has security issues
    - Not used much any more except on UNIX-based Web servers

Programming Web Apps Using ASP

- Mix HTML & server-side scripts in a single .asp file
  - Scripts usually written in VBScript or JavaScript
    - Interpreted, so it can be slow
  - Make it easy to write code that generates html content dynamically
- When an ASP page is requested, the page is parsed by the Server and any scripts it contains are executed
  - ASP Request object accesses input
  - ASP Response object writes HTML to the HTTP response
  - The following example uses VBScript code between <% and %> tags to check incoming requests for inputs named op1 & op2
    - VB script converts them to integers, adds them, converts result to a string, and writes string to http response using Response.Write( )
  - Note the HTML returned ("View" | "Source")
- ASP provides an easy way to dynamically generate HTML on Web servers

Calc.asp Example Program

```
<%@ Language = "VBScript" %>

<html>
  <body>
    <form>
      <input type="text" name="op1" value="<%= Request ("op1") %>"/>
      +
      <input type="text" name="op2" value="<%= Request ("op2") %>"/>
      <input type = "submit" value="  =  " />
      <%
        If Request ("op1") < > "" And Request ("op2") < > "" Then
          a = CInt (Request("op1"))
          b = CInt (Request("op2"))
          Response.Write (CStr (a + b))
        End If
      %>
     </form>
  </body>
</html>
```

Problems with ASP

- ASP is a good solution for doing server-side processing of HTML Form input and dynamically generating HTML
  - Higher level of abstraction than CGI
  - Also integrates seamlessly with ADO data bases
- But it has some problems
  - Interpreted scripts means slow execution
  - ASP has no true encapsulation model
    - Can't build reusable controls that encapsulate complex rendering and behavior logic
    - No event model as for Windows Forms
- ASP.NET – the new Microsoft solution

ASP.NET and Web Forms

- Web Forms are built from a combination of HTML, scripts, and server controls
  - Some examples: Button, TextBox, Label, DropdownList
  - Defined in classes in System.Web.UI.WebControls
- Object oriented
  - Whenever a Web page with server control objects is requested:
    - ASP.NET tells each object to render itself into HTML
    - HTML returned by controls is included in the HTTP response
    - Scripts (with event handlers) are executed as in ASP

ASP.NET Controls

- Server Control tag HTML general format:

  <asp:Control properties event-handler="handler-ftn" RunAt="server" />

  - Button Control example:

    <asp:Button Text=" = " OnClick="OnAdd" RunAt="server" />

    - Text= " = " Text property (= sign is displayed on button)
    - OnClick="OnAdd" Wires OnAdd event handler to button's Click event
    - RunAt="server" attribute
      - Signals ASP.NET is to execute the tag rather than treat as static HTML
      - Must be used with every tag that ASP.NET is to process
- Server Control script
  - Specify language and content of event handlers
  - Following example script's Click handler reads Text properties of TextBoxes ("op1" & "op2") and converts them to integers
  - Result is converted to a string & put into "Sum" Label's Text property
- Example calc.aspx (coded manually)
  - Put it into \Inetpub\wwwroot IIS root virtual directory
    - IIS must be running on the computer
  - Run locally by typing http://localhost/Calc.aspx

Calc.aspx

```
<html>
  <body>
    <form runat="server">
      <asp:TextBox ID="op1" RunAt="server" />
      +
      <asp:TextBox ID="op2" RunAt="server" />
      <asp:Button Text=" = " OnClick="OnAdd" RunAt="server" />
      <asp:Label ID="Sum" RunAt="server" />
    </form>
  </body>
</html>

<script language="C#" RunAt="server">
 void OnAdd (Object sender, EventArgs e)
 {
   int a = int.Parse(op1.Text);
   int b = int.Parse(op2.Text);
   Sum.Text = (a + b).ToString();
 }
</script>
```

HTML Returned by calc.aspx

```
<html>
 <body>
  <form name="_ctl0" method="post" action="calc.aspx" id="_ctl0">
   <input type="hidden" name="__VIEWSTATE"
    value="dDwxOTE0NDY4ODE2O3Q8O2w8aTwxPjs+O2w8dDw7bD
    xpPDc+Oz47bDx0PHA8cDxsPFRleHQ7PjtsPDU7Pj47Pjs7Pjs+Pjs+Pj
    s+dBISuMwmkqCcHr+yOOms28nF+A0=" />

   <input name="op1" type="text" value="2" id="op1" />
   +
   <input name="op2" type="text" value="3" id="op2" />
   <input type="submit" name="_ctl1" value="  =  " />
   <span id="Sum">5</span>
  </form>
 </body>
</html>
```

What Has ASP.NET Done?

- (Look at “View” | “Source” after entering data and clicking “=”)
- ASP.NET has:
  - Turned TextBox controls into <input type=“text”> form tags
  - Turned Button control into <input type=“submit”> form tag
  - Turned Label control into <span> (formatting) form tag
- The controls project a user interface by rendering themselves into HTML

What Happened?

- User clicks "=" button
  - Form is posted back to the server
  - ASP.NET notifies the Button object and the Button responds by firing a Click event on the Server
  - ASP.NET then calls the OnAdd() handler
    - (We write this)
  - In calc.aspx, it computes the sum, converts it to a string, and puts it in the Sum label's Text property
  - ASP.NET then renders the result into an HTML page
  - Since Sum.Text is now a non-null string, output of the Sum label control includes that string inserted between <span> tags
  - Page is returned to the browser

__VIEWSTATE Tag

- Mechanism ASP.NET uses to round-trip data from client to server back to client
- Recall HTML is stateless
  - Nothing is remembered when a new page replaces the old one
  - So how do we determine if the state changed?
  - View State
    - A place where controls can store their state in a way that it remains valid from one request to the next
    - Especially useful for controls that fire change events
  - View State is transmitted to the client in a hidden control and transmitted back to the server with the Form's postback data
    - VIEWSTATE tag contains all data encoded so that ASP.NET can detect changes to the page and fire change events

Using VS Designer to Create Web Forms

- "File" | "New" | "Web Site" | "ASP.NET WebSite"
  - Language: Visual C#
  - Location posibilities:
    - File System for a local Web Form that cannot be served to other machines
      - Browse to a parent directory and enter a Directory name
      - Physical directory of project files: C:\ParentDirectory\DirectoryName
        - » URL will be: http://localhost:PortAddress/ParentDirectory /DirectoryName
        - » PortAddress will be assigned by Visual Web Developer
    - HTTP means IIS will be used to serve the page from a virtual directory
      - Project files will be in physical directory C:\Inetpub\wwwroot\DirectoryName
        - » The URL will be http://localhost/DirectoryName/FormName (from same machine)
        - » http://HostMachineName/DirectoryName/FormName (from another machine)
  - In either case .sln solution file will be in a separate new directory:
    - C:\My Documents\Visual Studio 2005\Projects\DirectoryName
- Two source files generated: default.aspx and default.aspx.cs
    - Separates the HTML from the C# code script
    - Called code-behind programming

- In Solution Explorer, right click on the default.axpx file to rename it
  - Also change the class name in the .aspx.cs file
  - Also change name after Inherits= in first line of the .aspx file so it is the same as the new name (without file extensions)
- Drag and drop server controls from the tool box
- Set properties, add event handlers as for Windows Forms applications
- Edit the skeleton code generated by the Designer
- Start the Web application
  - Debug | Start Without debugging
  - In a "File System" project the Visual Web Developer Web server is started
    - There's an icon on the task bar
      - When double clicked a bubble msg appears giving the physical and logical address of the Web server and the port the server is listening to
- Examples: WebSite2006HTTP, WebPageFile-2006

Validation Web Controls

- Determine whether data in another control is in correct format
  - Provide a mechanism for validating user input on the client
- Validation is performed on the server side
- To use: attach validation control to an input control and set an error message
  - At run time, when user inputs data, the error message is displayed if the validation rule is violated

Some Validation Controls

Control	Purpose	Properties to Set
RequiredFieldValidator	User must enter something in field	ControlToValidate ErrorMessage
CompareValidator	Compare value in field to another value	ControlToValidate ControlToCompare or ValueToCompare Type, ErrorMessage
RangeValidator	Makes sure input value falls in specified range	ControlToValidate Minimum Value Maximum Value Type, ErrorMessage
RegularExpressionValidator	Validates against a regular expression such as required number of digits or a formatted value (e.g., SS #) Use Regular Expression Editor	ControlToValidate ValidationExpression ErrorMessage
ValidationSummary	Displays a summary of all messages from other validation controls	Display Mode

- These controls are in the Designer tool box and can be dragged over
- TestWeb Example: Using a RequiredFieldValidator

Using a Virtual Directory

- If you don’t want to clutter up the Inetpub\wwwroot directory on the target computer, put your program into a virtual directory
- Creating a Virtual Directory on target machine
  - “Start” | “Control Panel” | “Administrative Tools”
  - Start “Internet Information Services”
    - In left pane expand the Local Computer\Web Sites folder
    - Select Default Web Site
    - From “Action” menu item
      - “New” | “Virtual Directory”
      - Starts the Virtual Directory Creation Wizard
        - » Type in an alias name for the directory
        - » Enter or browse to the path of the desired directory
        - » Click “Next” and “Finish”
- To run the app, type URL address into browser:

  http://machine-name/alias-name/pgm-folder/pgm-name.aspx