Short history of ASP.NET

- Server side programming initially relied on external programs (CGI) -- inefficient
- Microsoft introduced Active Server Pages in 1996
  - Pages include code executed inside web server process
- Latest framework ASP.NET 2.0 released in 2005
  - Better developer productivity and site maintainability
  - Some concepts inherited from desktop programming
  - Separation between HTML and code
  - Can use various languages for the code (Visual Basic, C#)
  - Also uses client-side programs (JavaScript)
Hello world from ASP.NET!
<form name="form1" method="post" action="ControlsExample.aspx" id="form1">
  <div><input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE" value="/wEPD..." /></div>
  <div>
    <span id="lblMessage">Your name is George and your favorite fruit is Banana. Do you want to play again?</span>
    <p>Please enter your name and tell us which fruit you like most.</p>
    <select name="ddlFruits" id="ddlFruits">
      <option value="Apple">Apple</option>
      <option selected="selected" value="Banana">Banana</option>
      <option value="Orange">Orange</option>
      <option value="Peach">Peach</option>
    </select>
    <input name="txtName" type="text" value="George" id="txtName" />
    <input type="submit" name="btnSubmit" value="Submit preference" id="btnSubmit" />
  </div>
  <div><input type="hidden" name="__EVENTVALIDATION" id="__EVENTVALIDATION" value="/wEW..." /></div></form>

```csharp
public partial class ControlsExample : System.Web.UI.Page{
    protected void Page_Load(object sender, EventArgs e){
        if (IsPostBack) {
            lblMessage.Text = "Your name is " + txtName.Text + " and your favorite fruit is " + ddlFruits.Text + ". Do you want to play again?";
        }
        else {
            ddlFruits.Items.Add("Apple");
            ddlFruits.Items.Add("Banana");
            ddlFruits.Items.Add("Orange");
            ddlFruits.Items.Add("Peach");
        }
    }
}
```

Conventional IDs of controls

- Label – lbl...
- TextBox – txt...
- Button – btn...
- LinkButton – lbtn...
- ImageButton – ibtn...
- Hyperlink – hlnk...
- Table – tbl...
- DropDownList – ddl...
- ListBox – lst...
- CheckBox – chk...
- CheckBoxList – cbl...
- RadioButton – rdo...
- RadioButtonList – rbl...
- Image – img...
- ImageMap – imap...
- BulletedList – blst...
- Calendar – cln...
- FileUpload – upl...
Debugging

- Can use VS debugging features for C# code
  - Browser will be waiting for reply from server while you stop C# code
- Can set `Trace="True"` in `Page` directive
  - Outputs generous details at end of web page
    - Profiling information for important steps
    - Hierarchy of elements (control tree) with rendered sizes
    - Request and response details and headers
    - Details about server
    - Much more

Lecture outline

- ASP.NET overview
- Keeping state
- ASP.NET page lifecycle
- Multi-page applications
Keeping state between pages

- http is stateless protocol – web server may reboot between two requests from client
- Need to pass information between pages
- Also between postbacks of same page
- ASP.NET uses multiple mechanisms
  - ViewState
  - Session state
  - Application state

ViewState

- Preserves values of server control properties that are set from code
  - E.g. preserving selections in drop-down list
- Encoded into a hidden input field
  - If it gets too large can be a performance problem
  - Can be disabled setting to False control’s EnableViewState property
  - Trace also shows how many bytes of view state each control uses

Session state object

- The session state object is used to store key-value pairs the programmer needs
  - Accessible through the Session property of page object (e.g. Session["Email"]="foo@bar")
  - Accessible to all pages of web site
- Separate session for each visitor
  - Session ID identifies individual session
  - Typically stored as a cookie at the client
  - If cookies disabled, encoded in URLs
  - New visitors start with empty session object
Session configuration
- In site’s web.config file
- Where is session data stored?
  - By default inside the web server’s memory
  - Can be at separate “state server”
    - Web servers in server farm get consistent view
    - At database server
- Timeout specifies after how much inactivity the session data gets discarded
  - Defaults to 20 minutes

Lecture outline
- ASP.NET overview
- Keeping state
- ASP.NET page lifecycle
- Multi-page applications

When is a page submitted?
- Explicit postback by the user – pushing the submit button
- Implicitly by changing the value of an input control with the AutoPostBack property
  - Works for check boxes, drop-down lists, text fields, radio buttons, etc.
  - Not needed for buttons – they always cause a postback
How is a page processed?

- The web server builds the page through a well-defined succession of events
  - Events associated with page or individual control
  - Handlers are methods of the page object
- How to specify/override handlers for events
  - `<asp:Button ... OnClick="btnClick" />` means `btnClick()` runs when user clicks button
  - Overriding say `OnLoad()` handler of page
  - `Page_Load()` method invoked on page load
    - The `OnLoad()` method of base class called implicitly

Page lifecycle (1)

1. Server receives request
   - If page dynamic, server parses it and compiles code behind file if source newer than the .dll
2. Object initialization
   - Page's `Init_Page()`/`OnInit()` and controls’ `OnInit` handlers called
   - Should not reference other controls, they may be uninitialized
   - Suggested use: read/initialize control properties
3. ViewState data loaded into controls

Page lifecycle (2)

4. Postback data loaded into controls
5. Load event for page and controls
   - Suggested use: open DB connections, set / change properties of page/control
6. Postback change events for controls
   - E.g. `OnTextChanged()` handler for a text box
7. `PreRender` for page and controls
   - Suggested use: final changes to page/controls
Page lifecycle (3)

8. ViewState data saved
9. Rendering: the page asks the controls to produce the HTML code they contribute and the document is built
10. Unload event for page and controls
   - Suggested use: close DB connections, close files, other cleanup operations

Lecture outline

- ASP.NET overview
- Keeping state
- ASP.NET page lifecycle
- Multi-page applications

Moving to another page

- Links work just as for normal web pages
- The PostBackURL attribute of a button may specify a different page
  - PreviousPage refers to page generating post
  - Use its FindControl(id) to access values user filled in
  - Request.Form[id] has values submitted to server
- Within code-behind use Server.Transfer(URL) or Response.Redirect(URL) to go to other page
  - Redirect involves an extra roundtrip time, but it updates the URL displayed by browser – more user friendly
A consistent look and feel

- The problem
  - The pages of your site/application should incorporate some common elements
    - E.g. navigation menu, logo, footer
  - When common element changes, manual update of pages can lead to inconsistencies
- The solutions (centralize common elements)
  - The old solution: using server side includes (SSI)
  - ASP.NET: using master pages

ASP.NET master pages

- The master page contains all elements common among pages of a site
- Individual pages replace the `<asp:contentplaceholder>` element
- Master page may contain controls, has code behind file
  - MasterPage object accessible to the code of the individual page as the Master property
- Can have multiple Master pages per site