

# ASP.NET

CS 640, Lecture 12



---

---

---

---

---

---

---

---

## Lecture outline

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



---

---

---

---

---

---

---

---

## Short history of ASP.NET

- Server side programming initially relied on external programs (CGI) -- inefficient
- Microsoft introduced **A**ctive **S**erver **P**ages 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)



---

---

---

---

---

---

---

---

ASP.NET hello world page - Mozilla Firefox

http://localhost:1441/SimpleSite/HelloWorld.aspx

HelloWorld.aspx at server

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="HelloWorld.aspx.cs" Inherits="HelloWorld" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/...">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server"><title>ASP.NET hello world page</title></head>
<body id="theBody" runat="server"></body>
</html>
```

HelloWorld.aspx.cs at server

```
using System;

public partial class HelloWorld : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        theBody.InnerHtml = "<h1>Hello world from ASP.NET!</h1>";
    }
}
```

View source at client

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/...">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head><title> ASP.NET hello world page</title></head>
<body id="theBody"><h1>Hello world from ASP.NET!</h1></body>
</html>
```

---

---

---

---

---

---

---

---

## How it works – basics

- ASP.NET pages with .aspx extension and contain
  - Plain HTML – typically makes it to the client unmodified
  - The <% Page ...%> directive – for web server only
  - Various server controls (discussed later)
- Code-behind file holds the code for the page
  - Parts of the class are defined in .aspx file – "partial class"
    - Tags with "runat=server" become members of page object
- Page lifecycle in a nutshell: server parses .aspx file, creates page object, invokes event handlers, page renders itself (into HTML), page object disposed

---

---

---

---

---

---

---

---

## Server Controls

- HTML server controls are just HTML tags accessible to the page object (not used often)
- **Web server controls** are objects controlling properties of web page elements
  - Fundamental building blocks of ASP.NET pages
  - Tag names for ASP.NET controls start with <asp:
  - Some controls hold input from user
  - Validation controls check correctness of user input
  - Can bind data sources to HTML elements
  - Examples: drop-down lists, text boxes, tables, buttons

---

---

---

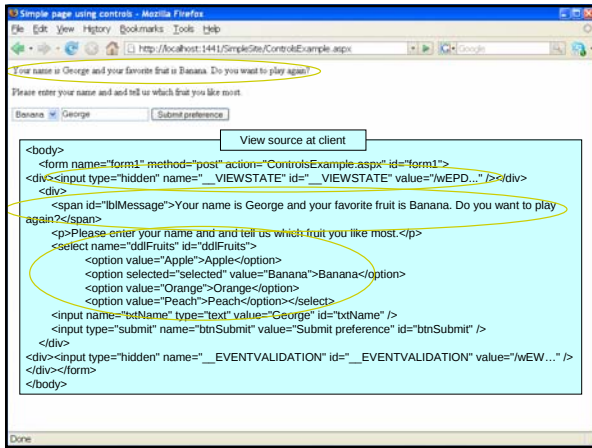
---

---

---

---

---




---

---

---

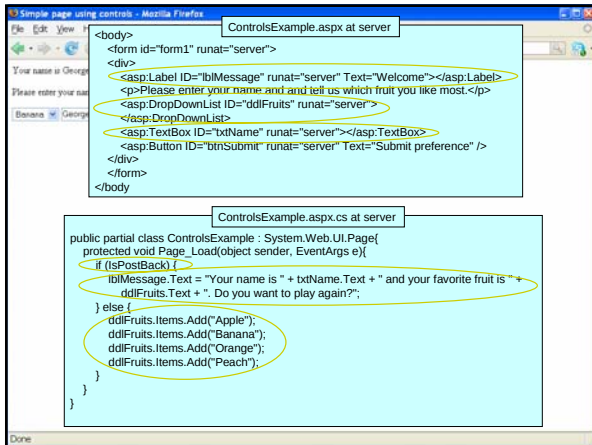
---

---

---

---

---




---

---

---

---

---

---

---

---

## 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

---

---

---

---

---

---

---

---

ASP.NET Hello world page - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://localhost:1441/SimpleSite/HelloWorld.aspx

**Hello world from ASP.NET!**

Request Details

Session ID: mcdmccafgnsl3pckhemmg Request Type: GET  
Time of Request: 2/20/2007 4:48:11 PM Status Code: 200  
Request Encoding: Unicode (UTF-8) Response Encoding: Unicode (UTF-8)

Trace Information

| Category  | Message                 | From First(s)       | From Last(s) |
|-----------|-------------------------|---------------------|--------------|
| aspn_page | Begin Print             | 0.00290289290830259 | 0.002903     |
| aspn_page | End Print               | 0.00299073066548981 | 0.001088     |
| aspn_page | Begin Init              | 0.00408956718593129 | 0.000298     |
| aspn_page | End Init                | 0.0041281793019909  | 0.000039     |
| aspn_page | Begin InitComplete      | 0.00416505449714978 | 0.000037     |
| aspn_page | End InitComplete        | 0.00419997513650478 | 0.000035     |
| aspn_page | Begin PreLoad           | 0.00423461641074494 | 0.000035     |
| aspn_page | End PreLoad             | 0.00427400689193738 | 0.000039     |
| aspn_page | Begin Load              | 0.00556495308761309 | 0.001291     |
| aspn_page | End Load                | 0.00561970855012173 | 0.000055     |
| aspn_page | Begin LoadComplete      | 0.00565770230577998 | 0.000038     |
| aspn_page | End LoadComplete        | 0.0056937404055434  | 0.000036     |
| aspn_page | Begin PreRender         | 0.00574449596960208 | 0.000055     |
| aspn_page | End PreRender           | 0.00578648962368122 | 0.000038     |
| aspn_page | Begin PreRenderComplete | 0.00582168962815106 | 0.000035     |
| aspn_page | End PreRenderComplete   | 0.015814287848433   | 0.045860     |
| aspn_page | Begin SaveState         | 0.0517384192683707  | 0.000057     |
| aspn_page | End SaveState           | 0.0517761335588741  | 0.000038     |
| aspn_page | Begin SaveStateComplete | 0.0518118922935736  | 0.000036     |
| aspn_page | End SaveStateComplete   | 0.0518501653143067  | 0.000038     |
| aspn_page | Begin Render            | 0.051753127841667   | 0.020963     |
| aspn_page | End Render              |                     |              |

Done

---

---

---

---

---

---

---

---

## 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/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

```
<%@ Master Language="C#" AutoEventWireup="true" CodeFile="MasterPg.master.cs" Inherits="MasterPg" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/...">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server"><title>Untitled Page</title></head>
<body><table><tr>
<td width="100px" style="background-color:Yellow" valign="top">
<a href="HelloWorld.aspx">Hello</a><br />
<a href="ControlsExample.aspx">Controls</a></td>
<td><asp:contentplaceholder id="ContentPlaceholder1" runat="server">
</asp:contentplaceholder></td></tr></table>
</body></html>

ContentPage.aspx at server
<%@ Page Language="C#" MasterPageFile="~/MasterPg.master" AutoEventWireup="true"
CodeFile="ContentPage.aspx.cs" Inherits="ContentPage" Title="Using master page" %>
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceholder1" Runat="Server">
This is the text of the individual content page that is combined with the master page. <br /><br />
</asp:Content>

View source at client
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/...">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head><title>Using master page</title></head>
<body><table><tr>
<td width="100px" style="background-color:Yellow" valign="top">
<a href="HelloWorld.aspx">Hello</a><br />
<a href="ControlsExample.aspx">Controls</a></td>
<td>This is the text of ...<br /><br /></td></tr></table>
</body></html>
```