# ASP.Net – Part II

Jim Fawcett

CSE686 – Internet Programming

Spring 2011

## References

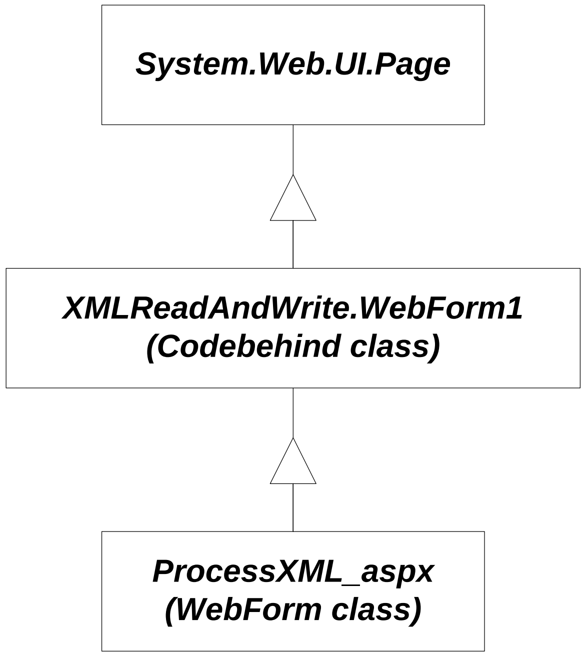
- Pro ASP.Net 4.0 in C# 2010, MacDonald, Freeman, & Szpuszta, Apress, 2010
- Programming Microsoft .Net, Jeff Prosise, Microsoft Press, 2002, Chapters 5 and 6.
- Essential ASP.NET with Examples in C#, Fritz Onion, Addison-Wesley, 2003
  - Several of the examples used here for state management were used with only minor modifications from this reference.

## Topics

- Architecture
- Controls
- Data Binding
- State Management

# Architecture

- ASP application
  - ProcessXML.aspx
  - ProcessXML.aspx.cs
  - Web.config
- Page Class
  - MapPath()
  - Application
  - ContentType
  - Context
  - IsPostBack
  - Request
  - Response
  - Server
  - Session
  - Trace
  - User
  - …
- ProcessXML_aspx
  - Page_Load(Object, System.EventArgs)
  - Button1_Click(Object, System.EventArgs)
  - InitializeComponent()
  - …


System.Web.UI.Page
XMLReadAndWrite.WebForm1 (Codebehind class)
ProcessXML_aspx (WebForm class)

## Page Events

- public event EventHandler Init;
  Page_Init(object,EventArgs)
- public event EventHandler Load;
  Page_Load(object,EventArgs)
- public event EventHandler PreRender;
  Page_PreRender(object,EventArgs)
- public event EventHandler Unload;
  Page_Unload(object,Eventargs)

- protected virtual void
  OnInit(EventArgs e);
- protected virtual void
  OnLoad(EventArgs e);
- protected virtual void
  OnPreRender(EventArgs e);
- protected virtual void
  OnUnload(EventArgs e);

# ASP.Net Directives

- @Page
  - Defines Language and Code-Behind file
- @Import Namespaces
  - Equivalent to using directives
- @Register
  - Registers user controls with page.  Page will call render on each of its registered controls.
- @Implements
  - Declares an interface this page implements
- @Reference
  - Specifies a page or user control that will be compiled and linked at run-time
- @Assembly
  - Links an assembly to the current page during compilation
- Plus more – see help documentation

## Page Attribures

- CodeFile
  - Specifies a path to a code-behind file for the page.  Used with Inherits attribute.
- Inherits
  - Defines a code-behind class for the page to inherit.
- AutoEventWireup
  - If true, the default, simple event handlers like Page_Load(…) are wired up automatically.
- Debug
  - If true, code behind is compiled with debug symbols.

## ASP Components

- You can create library assemblies that are available to every aspx page in your application.
  - Compile the library dll assembly
  - Place it in a bin directory under the application virtual directory
  - It will then be implicitly referenced by any page that loads from the application directory
  - You can copy over the dll with an update without stopping IIS.
    - If you do this, the new version becomes available on the next page load.

# Controls

- ***HTML Controls***
  - HTML syntax
  - runat=server attribute
  - Derives from HtmlControl
  - Instance created at server when page is constructed
- Examples:
  - <form runat=server>
  - <img runat=server>
  - <input type=file runat=server>
  - <input type=radio runat=server>

- ***Web Controls***
  - asp: prefix
  - runat=server attribute
  - Derives from WebControl
  - Instance created at server when page is constructed
  - Richer set of methods, properties, and events than HTML Controls
- Examples:
  - <asp:TextBox id=tb1 runat=server>
  - <asp:Button Text="Submit" runat=server>

## Web Control Catalog

- TextBox
- Label
- HyperLink
- Image
- CheckBox
- RadioButton
- Table – matrix addresses
- Panel
- Button
- ListBox
- DropDownList
- CheckBoxList
- RadioButtonList
- Repeater – HTML template
- DataList – HTML template
- DataGrid – no longer in toolbox by default, but can be added
- Calendar
- Validation Controls
  - RequiredField
  - RegularExpression
  - Range
  - Compare
  - Custom

# Data Related Controls

- ***Data Controls***
  - GridView
  - DataList
  - DataSet
  - DetailsView
  - FormView
  - Repeater
  - SqlDataSource
  - ObjectDataSource
  - XmlDataSource
  - SiteMapDataSource
- ***Validation Controls***
  - RequiredFieldValidator
  - RangeValidator
  - RegularExpressionValidator
  - CompareValidator
  - CustomValidator

# More Controls

- ***Navigation Controls***
  - SiteMapPath
  - Menu
  - TreeView
- ***Login Controls***
  - Login
  - LoginView
  - PasswordRecovery
  - LoginStatus
  - LoginName
  - ChangePassword
- ***Webparts***
  - WebPartManager
  - ProxyWebPartManager
  - WebPartZone
  - CatalogZone
  - DeclarativeCatalogPart
  - PageCatalogPart
  - ImportCatalogPart
  - EditorZone
  - AppearanceEditorPart
  - BehaviorEditorPart
  - LayoutEditorPart
  - PropertyGrideEditorPart
  - ConnectionsZone

# User Defined Controls

- User controls are stored in ascx files.
- They contain an @control directive that plays the same role as the @Page directive for WebForms.
  - <%@ Control classname="UserControlCS" %>
- In an aspx file that uses the control:
  - <%@ Register
        TagPrefix="cse686" TagName="IP" Src="MyControl.ascx"
    %>
  - <cse686:IP id="myControl1" runat="server" />
- A user control may contain HTML and codebehind with methods, properties, and events.
- Events are declared as delegates with the event qualifier

## Custom Server Controls

- Custom Server Controls are stored in C# files.
- A Server Control contains a C# class that defines the attributes:
  - [Bindable(true)]
  - [Category("Appearance")]
  - [ToolboxData("<{0}:NavBar runat=server></{0}:NavBar>")]
- And a class NavBar : System.Web.UI.WebControls.WebControl
- In an aspx file that uses the control:

  -
    ```
    <%@ Register
        TagPrefix="cse686" assembly="NavControl"
        namespace="NavControl
    %>
    ```
  - <cse686:NavBar id="NavBar1" runat="server" />

## Data Binding

- Data Binding provides an abstraction for loading a control with data provided by some collection.
- The data is cached in the control until it is rendered on the client's page by putting it onto the response buffer, formatted according to the control's policy.
- We have already seen an example of binding an HTML table to an XML file, in Lecture #2.
- Binding is often used when an ASP application connects to a database through a DataReader or DataSet.

## Data Binding

- Controls that Support Data Binding must expose:
  - a property called DataSource
  - a method called DataBind()

- The data source must provide:
  - IEnumerable interface

- Example:

```
DataSet ds = new DataSet();
ds.ReadXML(Server.MapPath("test.xml");
ListBox1.DataSource = ds;
ListBox1.DataTextField = "file";  // omit if flat
ListBox1.DataBind();
```

# Data Binding

- ***Data Binding Controls***
  - HtmlSelect
  - CheckBoxList
  - DataGrid
  - DataList
  - Repeater
  - DropDownList
  - ListBox
  - RadioButtonList
- ***Data Sources***
  - Array
  - ArrayList
  - HashTable
  - Queue
  - SortedList
  - Stack
  - StringCollection
  - DataView
  - DataTable
  - DataSet
  - IDataReader
  - Classes that implement IEnumerable

## State Management

- Adding user state inherently reduces scalability.
  - So if you are trying to provide a resource that handles a large volume of traffic, you will want to minimize use of state.

- Types of state
  - Application:
    Shared across all clients of this application
  - Session:
    Per client state persistent over page boundaries. Requires cookies or URL mangling to manage client association.
  - Cookie:
    Per client state stored on client. Clients can disable cookies.
  - ViewState:
    Shared across post requests to the same page. Sent back and forth with each request.

# Application State

- In Global.asax: (add new item/Global Application Class)

```
void Application_Start(object src, EventArgs e)
{
  DataSet ds = new DataSet();  // populated by clients
  Application[“SharedDataSet”] = ds;
}
```

- In Application Page:

```
private void Page_Load(object src, EventArgs e)
{
  DataSet ds = (DataSet)(Application[“SharedDataSet”]);
  // client interacts with DataSet
}
```

## Session State

- By default session state is managed in the same process and application domain as the application so you can store any data in session state directly.
- Session state is available as a property of both Page and HttpContext classes.
- It is:
  - Initialized in Global.asax
  - Accessed in any member function of the Page.
- You specify whether you want session ids managed as cookies or URL mangling in the web.config file:

```
<configuration>
  <system.web>
    <sessionState cookieless=“true” />
  </system.web>
</configuration>
```

## Session State

- In Global.asax:

```
void Session_Start(object src, EventArgs e)
{
  DataSet ds = new DataSet();  // populated by clients
  Session[“myDataSet”] = ds;
}
```

- In Application Page:

```
private void Page_Load(object src, EventArgs e)
{
  DataSet ds = (DataSet)(Session[“myDataSet”]);
  // client interacts with DataSet
}
```

# Cookies

-
  ```
  Protected void Page_Load(Object sender, EventArgs e)
  {
    int age = 0;
    if(Request.Cookies[“Age”] == null)
      HttpCookie ac = new HttpCookie(“Age”);
      ac.Value = ageTextBox.Text;
      Response.Cookies.Add(ac);
      age = Convert.ToInt32(ageTextBox.Text);
    }
    else
    {
      age = Convert.ToInt32(Request.Cookies[“Age”].Value);
    }
      // use age
  }
  ```

# ViewState

- ViewState is used by ASP controls to transfer control state back and forth between server and client.
- You also can use ViewState to transfer application state:

```
private void Page_Load(Object sender, EventArgs e)
{
  ArrayLIst cart = (ArrayList)ViewState[“Cart”];
  if(cart == null)
  {
    cart = new ArrayList();
    ViewState[“Cart”] = cart;
  }
}
// use cart with:
   ArrayList cart = (ArrayList)ViewState[“Cart”];
   cart… yada, yada, yada
```

## End of Presentation