ASP.NET

Lecture 14

CS 638 Web Programming

Review

- ASP = Active Server Pages
  - ASP here refers to ASP.NET, not original ASP
- ASP pages have HTML and ASP controls
- ASP controls are members of a class that is built for you – can access them from program code that is associated with the ASP page
- Controls can raise events / cause postbacks
- Processing of a page has many steps in its lifecycle *on the server*

ASP.NET Project Creation

- Start VS 2005
- New->Web Site
- Select ASP.NET Web Site
- Fill in location
- Hit OK
- An "empty" default.aspx page is created and shown.

Adding an ASP.NET control

- Hover or click on Toolbox
- Click and hold on "Label"
- Drag to desired location in aspx file (between the "div" statements in this case)
- Change ID of control to "lblTime"
- Debug / Run web page (F5)
- Say yes to creating a web config setting
- Browser appears with page loaded

If IE is not Default Browser

- If a browser other than Internet Explorer comes up, do this:
  - Switch back to VS2005
  - Hit Shift-F5 (kills currently running debug session)
  - Click or hover over Solution Explorer
  - Right click on default.aspx
  - Click on "Browse With"
  - Select Internet Explorer and set it to default
  - Hit OK and rerun page (F5) IE should come up

Debugging an ASP.NET page

- While still viewing page, hit reload (Ctrl-R)
- Page reloads
- Switch to VS2005
- Right click to bring up context menu
- Select "View code"
- default.aspx.cs comes up
- Put breakpoint (F9) on closing brace of Page_Load

Debugging an ASP.NET page

- Switch back to browser
- Reload page
- Execution of the page halts at the breakpoint moving VS2005 to front
- Continue execution with F5
- Browser comes to front
- Close browser window – VS2005 comes back – debugging is terminated
- Remove breakpoint (F9 again when on line)

Adding Another Control

- Looking at default.aspx, place (click) cursor before opening "<" of the ASP label
- Click or hover over Toolbox
- Double click on Button
- Button appears where cursor was positioned
- Change ID to "btnSubmit" and Text to "Get Time"
- Run page – should look like this…

Page so far

Add Event Handler

- Terminate debugging of page
- View default.aspx
- At bottom click "Design"
- A WYSIWYG preview of your page comes up
- Double click on the button
- View switches to default.aspx.cs
- VS2005 adds an event handling stub for you

Add Event Handler

- Enter:
  lblTime.Text = DateTime.Now.ToShortTimeString();
- Run page (F5)
- Click on button
- Should see:

Modifying Control's Look

- ASP.NET control offer many options for modifying their look
- Modifications can be declared right along with the control's declaration
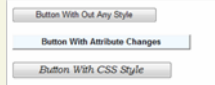- Controls can also reference CSS styles

Modifying a Control's Look

```
<asp:Button ID="Button1" runat="server" Text="" />
<asp:Button ID="Button2" runat="server" Text=""
   BackColor="AliceBlue"
   Font-Bold="True"  />
<asp:Button CssClass="bstyle" ID="Button3"
   runat="server" Text="" />
```

```
input.bstyle
{
    font-size: 16px;
    font-style:italic;
    font-family: Bookman Old Style, Garamond, Serif;
}
```

Button With Out Any Style
Button With Attribute Changes
Button With CSS Style

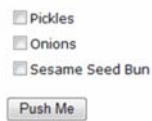## Working with Controls

```
<asp:CheckBoxList ID="CheckBoxList1" runat="server">
   <asp:ListItem Text="Pickles" Value="1"></asp:ListItem>
   <asp:ListItem Value="2">Onions</asp:ListItem>
   <asp:ListItem>Sesame Seed Bun</asp:ListItem>
</asp:CheckBoxList>
<br />
<asp:Button ID="Button1" runat="server" Text="Push Me" />
```

Working with Controls

☐ Pickles
☐ Onions
☐ Sesame Seed Bun

Push Me

Working with Tables

From the aspx file:
CheckBoxList1 states:

```
<asp:Table runat="server" ID="tblCBL1" Width="400">
</asp:Table>
```

CheckBoxList1 states:

Index	Text	Value	Checked
0	Pickles	1	False
1	Onions	2	False
2	Sesame Seed Bun	Sesame Seed Bun	False

Working with Tables

```
tbl.Rows.Clear();
TableRow row = new TableRow();
TableCell cell;
string[] headings = { "Index", "Text", "Value", "Checked" };
foreach (string s in headings)
{
    cell = new TableCell();
    cell.Text = s;
    row.Cells.Add(cell);
}
tbl.Rows.Add(row);
```

Initializing Controls via Code

❑ Initializing controls via code is often required
   ❑ e.g.: loading control contents from database
❑ From aspx file:

```
<asp:CheckBoxList ID="CheckBoxList1" runat="server">
</asp:CheckBoxList>
```

Initializing Controls via Code

```
ListItem[] cbrecords = { new ListItem("Pickles", "1"), new ListItem("Onions", "
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        InitializeCheckBoxList(cbrecords, CheckBoxList1);
    }
    DumpCheckBoxList(CheckBoxList1, tblCBL1);
}

private void InitializeCheckBoxList(ListItem[] records, CheckBoxList cbl)
{
    cbl.Items.Clear();

    foreach (ListItem r in records)
    {
        cbl.Items.Add(r);
    }
}
```

Questions…

```
ListItem[] cbrecords = { new ListItem("Pickles", "1"), new ListItem("Onions", "
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        InitializeCheckBoxList(cbrecords, CheckBoxList1);
    }
    DumpCheckBoxList(CheckBoxList1, tblCBL1);
}

private void InitializeCheckBoxList(ListItem[] records, CheckBoxList cbl)
{
    cbl.Items.Clear();

    foreach (ListItem r in records)
    {
        cbl.Items.Add(r);
    }
}
```

A Server-Based View Control

```
This Is View 1
[ Previous View ] [ Next View ]
Trips to the server this session: 0
```

□ Common in today's web pages are view or paging controls

□ ASP.NET provides a server-based view control

Control Container and its Items

```
<asp:MultiView runat="Server" ID="MultiView1" ActiveViewIndex="0">
    <asp:View ID="View1" runat="server">
        Mark up
    </asp:View>
    <asp:View ID="View2" runat="server">
        More mark up
    </asp:View>
    <asp:View ID="View3" runat="server">
        Even more mark up
    </asp:View>
</asp:MultiView>
```

ASP:Buttons Have New "Command" Semantics

□ You have seen:
`<asp:Button ID= "id" runat="server" Text="text" OnClick= "EventHandler" />`

□ There is also a "command based" event handler that permits "reserved" actions such as "PrevView" and "NextView" for View controls

□ What's the benefit?
You can better encapsulate implementation when creating controls for others to use

Master Pages

□ To create pages with a common look, use Master Pages

□ The master page aspx file specifies the markup of the overall look of the page

□ A placeholder is included for the markup of each specific or subordinate page

□ Both the master and the subordinate have implementation (code) files too and go through their own page lifecycles

Master Pages

□ Master page has different Page directive

```
<%@ Master Language="C#" AutoEventWireup="true"
    CodeFile="MasterPage.master.cs" Inherits="MasterPage" %>
```

□ Subordinate page's Page directive also changed:

… MasterPageFile="~/MasterPage.master" …

Master Pages

□ Master page includes:

```
<asp:contentplaceholder id="ContentPlaceHolder1" runat="server">
</asp:contentplaceholder>
```

□ Subordinate pages include:

```
<asp:Content ID="Content1"
    ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
… markup …
</asp:Content>
```