Dynamic Web Programming

BUILDING WEB APPLICATIONS USING ASP.NET, AJAX AND JAVASCRIPT
AGENDA

11. ASP.NET AJAX
11.1 What is AJAX
11.2 ASP.NET AJAX Server Controls
11.3 ASP.NET AJAX Control Toolkit

12. Advanced Topics
12.1 User Controls
12.2 ASP.NET Security
12.3 Forms Authentication
12.4 Web Services
12.5 Microsoft Silverlight
12.6 Microsoft Azure
Building Web Applications Using ASP.NET, AJAX And JavaScript

11. ASP.NET AJAX
11.1 WHAT IS AJAX

- ASP.NET is heavily dependent on server-side processing.
- Use JavaScript to perform client-side processing.
- One more option, AJAX (Asynchronous JavaScript and XML).
- AJAX constitute a partial page update allowing the developer send only parts of a page back to the server.

AJAX technologies:

- XMLHttpRequest: Powerful object that uses JavaScript to make HTTP GET and POST requests without submitting the whole page.
- DOM scripting: Using JavaScript, the raw data returned from the request is parsed and integrated into the page.
- Standards-based HTML and CSS: HTML provides the structure/content, CSS controls presentation of interface.
11.1 WHAT IS AJAX

- Partial page update makes round trip much faster.
- Page itself is not refreshed in the browser. You will notice that the scrollbar position remains constant.

Implementation of AJAX in ASP.NET:

- Set of client-side libraries, known as Microsoft AJAX library, providing the framework of cross-browser functionality that is used by ASP.NET AJAX server-side components.

Three core JavaScript files:

- MicrosoftAjax.js
- MicrosoftAjaxWebForms.js
- MicrosoftAjaxTimer.js

- HTTP handlers and modules on the server-side to wrap standard XML-based web services to respond to AJAX requests using JSON.

- Set of server-side controls that make “Ajaxifying” a page incredibly easy. Most important one are ScriptManager and UpdatePanel controls.
### 11.2 ASP.NET AJAX Server Controls

- **Toolbox contains AJAX Extensions section:**

<table>
<thead>
<tr>
<th>AJAX Server Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ScriptManager</td>
<td>The ScriptManager is the brains of the server-side ASP.NET AJAX model. It is a web control that does not have any visual appearance on the page. However, it performs a key task—it renders the links to the ASP.NET AJAX JavaScript libraries.</td>
</tr>
<tr>
<td>ScriptManagerProxy</td>
<td>When using master pages, it makes sense to place a ScriptManager object in the master page. However, this can occasionally cause problems, because different content pages may want to configure the properties of the ScriptManager differently (for example, adding new scripts and web service references). In this scenario, the solution is to use the ScriptManager in the master page and the ScriptManagerProxy in your content page. Each content page can configure the ScriptManagerProxy control in the same way it would configure the ScriptManager.</td>
</tr>
<tr>
<td>Timer</td>
<td>The Timer control is refreshingly straightforward. You simply add it to a page and set its Interval property to the maximum number of milliseconds that should elapse before an update. For example, if you set Interval to 60000, the timer will force a postback after one minute elapses.</td>
</tr>
<tr>
<td>UpdatePanel</td>
<td>The UpdatePanel is a handy control that lets you take an ordinary page with server-side logic and make sure it refreshes itself in flicker-free Ajax style. The basic idea is that you divide your web page into one or more distinct regions, each of which is wrapped inside an invisible UpdatePanel. When an event occurs in an UpdatePanel that would normally trigger a postback, the UpdatePanel intercepts the event and performs an asynchronous callback instead.</td>
</tr>
<tr>
<td>UpdateProgress</td>
<td>ASP.NET AJAX also includes an UpdateProgress control that works in conjunction with partial rendering at the UpdatePanel. Essentially, the UpdateProgress control allows you to show a message while a time-consuming update is under way.</td>
</tr>
</tbody>
</table>

---

**Example 10-1: Using AJAX server controls**

1. Use the AJAX web site provided by the instructor.
2. The page Example10_1.aspx (and the corresponding cs page) are already set up in this project.
3. First, test the web site by running viewing the page in your browser. It should display book records in a GridView control. This page is a copy of Example7_3.
4. Size your browser window to a smaller size vertically so that the vertical scrollbar is displayed.
5. Scroll down to the bottom and click on an ISBN number in the GridView control. Notice that the page is posted back and the page is positioned back at the top (scrollbar up).
11.2 ASP.NET AJAX Server Controls

- The UpdatePanel wraps controls that are to be updated asynchronously using AJAX.
- Divide your page in distinct regions using UpdatePanels to initiate AJAX requests.
- When an event occurs based on controls within the UpdatePanel that normally trigger a postback, it is intercepted and asynchronously processed.

Example 10.1 (continued): Using AJAX server controls

7. From the toolbox, AJAX extension, drag an update panel below the form’s div tag but above the ObjectDataSource. Place the end tag at the bottom of the page before the end div tag.
8. Inside the UpdatePanel type <Con> and select ContentTemplate from the Intellisense list. Place the end tag before the UpdatePanel end tag.
9. Save the project and view the page in your browser. You should see the following error message:

Server Error in /AJAX Application.
The control with ID 'UpdatePanel1' requires a ScriptManager on the page. The ScriptManager must appear before any controls that need it.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.Web.HttpException: The control with ID 'UpdatePanel1' requires a ScriptManager on the page. The ScriptManager must appear before any controls that need it.

Source Error:
An unhandled exception was generated during the execution of the current web request. Information regarding the origin and location of the exception can be identified using the exception stack trace below.

Stack Trace:

[...]
10. Navigate back in source view of the aspx page and add a ScriptManager object from the AJAX section of the toolbox above the UpdatePanel.
11. Save the project and view the page in your browser. Repeat steps 4 and 5 and notice that the page itself is not posted back, yet the book record was selected. Also the page position was not changed.
Example 10.2: Using multiple AJAX Update Panels

1. Right-click on the project in Solution Explorer of the AJAX project and select Add New Item.
2. Select Web Form and type Example10_2.aspx as the file name. Click on OK.
3. Type the following aspx markup code inside the div tags:

```html
<h3>AJAX calls using two separate update panels</h3>
<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
        <asp:Label ID="lblDateTime1" runat="server" Text="Label"></asp:Label>
        <asp:Button ID="btnDateTime1" runat="server" Text="Date/Time 1" />
    </ContentTemplate>
</asp:UpdatePanel>
<br />
<br />
<asp:UpdatePanel ID="UpdatePanel2" runat="server">
    <ContentTemplate>
        <asp:Label ID="lblDateTime2" runat="server" Text="Label"></asp:Label>
        <asp:Button ID="btnDateTime2" runat="server" Text="Date/Time 2" />
    </ContentTemplate>
</asp:UpdatePanel>
<br />
<br />
<h4>The following button lies outside any update panel</h4>
<asp:Button ID="btnDateTime3" runat="server" Text="Date/Time 3" />
```

4. Type the following C# code in the page load:

```csharp
protected void Page_Load(object sender, EventArgs e)
{
    lblDateTime1.Text = DateTime.Now.ToLongTimeString();
    lblDateTime2.Text = DateTime.Now.ToLongTimeString();
}
```

5. Save the project and view the page in your browser. Notice that when you click the Date/Time 1 or the Date/Time 2 button, it updates both regions in the separate Update panels at the same time. When you click on the Date/Time 3 button, it performs a full postback as this button is not wrapped inside an Update panel.

6. Navigate back to the source view of the aspx page. For each Update panel, set the UpdateMode property using the property sheet to Conditional.

7. Save the project and view the page in your browser. Now click on each individual button and you notice that only the corresponding panel is updated.
With conditional update mode, you have other options for triggering an update.

By default, all controls within the UpdatePanel trigger AJAX.

Use triggers to initiate AJAX for specific controls only within the UpdatePanel.

You can also use triggers to refer to controls outside the UpdatePanel to trigger an AJAX request.

Finally, you can use triggers to actually specify a “normal” postback for a specific control.

Example 10-3: Using AJAX Update Panels events

1. Right-click on page Example10_2 and select copy. Paste the page into the current project and rename it Example10_3.
2. Add the following trigger to the UpdatePanel1:

   `<Triggers>
   <asp:AsyncPostBackTrigger ControlID="btnDateTime3" EventName="Click" />
   </Triggers>`

3. Save the project and view the page in your browser. Click on the Date/Time 3 button and notice that it triggers the asynchronous update of UpdatePanel1.
4. Add another trigger to the UpdatePanel2 as shown below:

   `<Triggers>
   <asp:PostBackTrigger ControlID="btnDateTime2" />
   </Triggers>`

5. Save the project and view the page in your browser. Notice that when you click the button Date/Time 2 that this button triggers a full postback and both UpdatePanels are updated.
Collaborative, open-source model AJAX toolkit containing many more AJAX controls.

Download it from http://ajax.asp.net/ajaxtoolkit

Extract files to any folder on your computer.

Zip file contains sample web site in addition to the toolkit demonstrating each individual control.

Create new tab in toolbox, name it AJAX Toolkit.

Right-click in the new tab, select Choose Items.

Click on browse and navigate to the folder where you extracted the files. Under SampleWebSite/bin folder, select all controls.
Example 10-4: Installing the AJAX Control Toolkit

1. Copy the AjaxControlToolkit.dll into C:\Temp\Webapps.
2. Open the Ajax web site in Visual Web Developer.
3. Add a new item to the project using the Web Form template and name it Example10_4.
4. In the toolbox right-click and select Add Tab and type AJAX Control Toolkit.
5. Right-click in a blank area of that tab and select Choose Items.
6. Under .NET Framework Components, click on the Browse button. Navigate to C:\Temp\Webapps.
7. Select the file AjaxControlToolkit.dll and click on OK.
8. Back in the dialog Choose Toolbox Item, click on OK.
9. You should see now controls in the newly added tab of your toolbox.
10. Drag the ToolscriptManager onto the page inside the div tags.
11. At the top of the page, you notice a new page directive, the Register Assembly directive. Change the TagPrefix to ajax.
12. Now change the tag prefix for the ToolscriptManager to ajax. This is simply to differentiate the ajax control toolkit controls from the standard web controls or any other controls.
13. Type the following aspx page:

```html
<div>
<ajax:ToolkitsScriptManager runat="Server" EnableScriptGlobalization="true"
  EnableScriptLocalization="true" ID="ScriptManager1" />
<h2>Calendar Demonstration</h2>
<br />
<samp>Default calendar: </samp>
<br />
<asp:TextBox runat="server" ID="txtDate" autocomplete="off" />
<asp:CalendarExtender ID="defaultCalendarExtender" runat="server" TargetControlID="txtDate" />
</div>
```

14. Save the project and view the page in your browser.
15. As you can see, the markup code is pretty minimal, but the functionality is pretty powerful.
12. ADVANCED TOPICS
12.1 USER CONTROLS

- User controls are composed of user interface portion with control tags (ascx) and code behind file (cs).

- Key differences between web form and user control:
  - User controls begin with a Control directive instead of a Page directive.
  - User controls use the file extension .ascx instead of .aspx, and their code-behind files inherit from the System.Web.UI.UserControl class.
  - User controls cannot be requested directly by a client browser. (ASP.NET will give a generic “that file type is not served” error message to anyone who tries.) Instead, user controls are embedded inside other web pages.

**Example 10-5:** Creating a Simple User Control

1. Right-click on the root of the AJAX project in solution explorer and select Add New Item.
2. Select the template Web User Control and name it Header.ascx. Click on Add.
12.1 USER CONTROLS

Example 10-5 (continued): Creating a Simple User Control

3. Type the following markup code:

```csharp
Control Language="C#" AutoEventWireup="true" CodeFile="Header.ascx.cs" Inherits="Header"

<style type="text/css">
.header
{
    font-size: x-large;
    font-weight: bold;
}
</style>
<table width="100%" border="0" style="background-color:Scrollbar">
    <tr>
        <td colspan="2" align="center">
            <span class="header">AJAX Web Site</span>
        </td>
    </tr>
    <tr>
        <td align="left">
            <asp:Label ID="lblDateTime" runat="server" Text="Label"></asp:Label>
        </td>
    </tr>
    <tr>
        <td align="right">
            <b>Web App Class © 2011</b>
        </td>
    </tr>
</table>
```

4. Type the following C# code in the control’s load event:

```csharp
protected void Page_Load(object sender, EventArgs e)
{
    lblDateTime.Text = DateTime.Now.ToString();
}
```

5. Save the user control page. To integrate this control into another page, you need to include the `Register` directive at the top of web form page.

6. Copy page Example10_3.aspx and name it Example10_6.aspx. Add the following directive:

```csharp
<%@ Register TagPrefix="hdr" TagName="Header" Src="~/Header.ascx" %>
```

7. Add the user control below the form’s div tag as shown below:

```html
<form id="form1" runat="server">
    <div>
        <hdr:Header ID="Header" runat="server" />
        <h3>AJAX calls using two separate update panels</h3>
    </div>
</form>
```

8. Save the project and view the page in your browser.
Security is all about protecting your assets from unauthorized actions.

You use several mechanisms to this end, including identifying users, granting or denying access to sensitive resources, and protecting the data that is stored on the server and transmitted over the wire.

ASP.NET contains classes for authenticating and authorizing users as well as dealing with authenticated users.

ASP.NET provides base classes for implementing confidentiality and integrity through encryption and digital signatures.

All these mechanisms are only part of a security model. You must write secure code.
Keep the following guidelines in mind when writing code:

- Never trust user input.
- Never use string concatenation for creating SQL statements.
- Never output data entered by a user directly on your web page before validating and encoding it.
- Never store sensitive data, business-critical data, or data that affects internal business rule decisions made by your application in hidden fields on your web page.
- Never store sensitive data or business-critical data in view state.
- Enable SSL when using Basic authentication or ASP.NET forms authentication.
- Protect your cookies.
- Use SSL.
12.2 ASP.NET SECURITY

- Basically, for mainstream web applications, the fundamental tasks for implementing security are always the same:
  - Authentication: Who is accessing the application.
  - Authorization: Which resources may this user access.
  - Confidentiality: Nobody else is able to view sensitive data processed by that user. (Data encryption)
  - Integrity: Ensure that data transmitted between client and server is not changed by unauthorized users. Use digital signatures to mitigate that threat.
  - Use System.Security namespace for encrypting and signing data.
  - Use SSL as a standardized way for ensuring confidentiality and integrity of data transmitted between client and server.
12.3 FORMS AUTHENTICATION

- Forms authentication is a ticket-based system.
- Forms authentication is a great option for developers:
  - You have full control over the authentication code.
  - You have full control over the appearance of the login form.
  - It works with any browser.
  - It allows you to decide how to store user information.

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FormsAuthentication</td>
<td>This is the primary class for interacting with the forms authentication infrastructure. It provides basic information about the configuration and allows you to create the ticket, set the cookie, and redirect from the login page to the originally requested page if the validation of credentials was successful.</td>
</tr>
<tr>
<td>FormsAuthenticationEventArgs</td>
<td>The FormsAuthenticationModule raises an Authenticate event that you can catch. The event arguments passed are encapsulated in an instance of this class. It contains basic information about the authenticated user.</td>
</tr>
<tr>
<td>FormsAuthenticationTicket</td>
<td>This class represents the user information that will be encrypted and stored in the authentication cookie.</td>
</tr>
<tr>
<td>FormsIdentity</td>
<td>This class is an implementation of IIdentity that is specific to forms authentication. The key addition to the FormsIdentity class, in addition to the members required when implementing the IIdentity interface, is the Ticket property, which exposes the forms authentication ticket. This allows you to store and retrieve additional information in the ticket, such as caching role information for simple scenarios.</td>
</tr>
<tr>
<td>FormsAuthenticationModule</td>
<td>This is the core of the forms authentication infrastructure that establishes the security context and performs the automatic page redirects to the login page if necessary.</td>
</tr>
</tbody>
</table>
12.3 FORMS AUTHENTICATION

- Complete the following steps to user forms authentication:
  - Configure forms authentication in the web.config file.
  - Configure IIS to allow anonymous access to the virtual directory, and configure ASP.NET to restrict anonymous access to the web application.
  - Create a custom login page that collects and validates a user name and password and then interacts with the forms authentication infrastructure for creating the ticket.
- Use Login controls in toolbox:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login</td>
<td>This control displays a login form that contains a User Name text box, a Password text box, a Remember me next time checkbox, and a Log In button. It is integrated with the membership API, and performs the login functionality without requiring you to write any code. The layout is customizable through templates and multiple properties.</td>
</tr>
<tr>
<td>LoginView</td>
<td>This control contains templates that display different data depending on whether or not the user is logged in. It can also display different templates for authenticated users depending on their roles.</td>
</tr>
<tr>
<td>PasswordRecovery</td>
<td>If the user has provided an email address and a secret question and answer during registration, this control will use them to recover the user's password.</td>
</tr>
<tr>
<td>LoginStatus</td>
<td>This is a simple yet useful control that displays a Login link if the user is not logged in; otherwise, it displays a Logout link. Again, this control requires no additional coding in order to work with your application’s membership data.</td>
</tr>
<tr>
<td>LoginName</td>
<td>This control displays the name of the logged-in user.</td>
</tr>
<tr>
<td>CreateUserWizard</td>
<td>This control displays a wizard for creating a new user account.</td>
</tr>
<tr>
<td>ChangePassword</td>
<td>This control displays a form that requests the user’s existing password and a new password, and includes the functionality to change the user’s password automatically, without requiring you to write additional code.</td>
</tr>
</tbody>
</table>
Example 10-5: Simple Forms Authentication

1. Create a new web form page in the AJAX project and name it Login.aspx.
2. Simply drag the Login control from the toolbox onto the page below the div tag. Name the control logAjax as shown below:

```html
<div>
    <asp:Login ID="logAjax" runat="server" />
</asp:Login>
</div>
```

3. Write the following C# code in the CS page. Make sure to add using System.Web.Security:

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class Login : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
    protected void Authenticate(object sender, AuthenticateEventArgs e)
    {
        e.Authenticated = FormsAuthentication.Authenticate(logAjax.UserName, logAjax.Password);
    }
}
```

4. Navigate to the web.config file and modify the authentication section:

```xml
<authentication mode="Forms">
    <forms />
</authentication>
```

The <authentication> section enables configuration of the security authentication mode used by ASP.NET to identify an incoming user. Password is password
5. Navigate back to the Login.aspx page in design view. Select the Login control and set the Authenticated event to the method Authenticate we have written above.

```
<tr>
  <td align="left">
    <asp:LoginStatus ID="LoginStatus1" runat="server" />
  </td>
  <td align="right">
    <asp:LoginName ID="LoginName1" runat="server" />
  </td>
</tr>
```

6. Open the Header.ascx file and another table row to include the following controls:

7. Save the project and try to run page Example10_5.aspx. You should be presented with the Login page rather than Example10_5. Log in as user with password password. If successful, you should see now the page Example10_5.

Also note the Logout link and the user name user.
Web Services
- One or more server-side methods that can be called by remote clients.
- Client sends HTTP request message, similar to web-page postback, except body of request contains arguments that are being passed to methods.
- ASP.NET creates web service object, runs the code, returns result and destroys the web service object.

Microsoft Silverlight
- Direct competitor to Flash.
- Create interactive content that runs on client, with support for dynamic graphics, media, and animation far beyond ordinary HTML.
12.6 MICROSOFT AZURE

- Application platform in the cloud.
- It provides cloud operating system called Windows Azure that serves as a runtime for applications.
- It provides a set of services to develop, manage and host applications off-premises.
- All Azure services and applications built using them run on top of Windows Azure.

Windows Azure has three components:

- **Compute:** Computation environment with Web Role, Worker Role and VM Role.
- **Storage:** SQL Azure, scalable version of SQL Server that runs on the Azure platform.
- **Fabric:** Network of interconnected nodes consisting of servers, high-speed connections and switches.