

# Workbook for Ajax

---

**A guide to AJAX Programming**

## Why This Module

ASP.NET AJAX is the free Microsoft AJAX framework for building highly interactive and responsive web applications that work across all popular browsers. The ASP.NET AJAX framework includes Server-Side ASP.NET AJAX, Client-Side ASP.NET AJAX, the AJAX Control Toolkit, and the jQuery library. ASP.NET AJAX enables developers to choose their preferred method of AJAX development, whether it is server-side programming, client-side programming, or a combination of both.

Developers familiar with the ASP.NET server-side programming model can take advantage of the ASP.NET AJAX server-side controls such as the ScriptManager, UpdatePanel, and the UpdateProgress control to add AJAX functionality to an ASP.NET application without writing any JavaScript.

If you prefer to work directly with JavaScript then you can take advantage of client-side ASP.NET AJAX. The client-side ASP.NET AJAX Library provides a foundation for building rich client-side applications. The library simplifies cross-browser development of client-side applications. For example, the library enables you to call web services and create components and controls -- all through pure client-side JavaScript code.

<b>Contents</b>	<b>Page No.</b>
<b>Why This Module</b>	<b>2</b>
<b>1.0 Introduction to AJAX</b>	<b>8-12</b>
1.1 Overview of AJAX	09
1.2 Communication through AJAX	10
1.3 Crossword	13
<b>2.0 Updating the page</b>	<b>14-27</b>
2.1 Using the UpdatePanel Control	15
2.2 Using the UpdateProgress Control	27
2.3 Crossword	33
<b>3.0 Other AJAX Extensions</b>	<b>34-45</b>
3.1 Using Timer Control	35
3.2 Using ScriptManagerProxy	42
<b>4.0 Web Services</b>	<b>46-63</b>
4.1 Introduction to Web Services	47
4.1 Error Handling	50
4.2 Page Methods	52
4.3 Maintaining Session State	54
4.4 Exchanging Complex Data with the Server	55
4.5 Consuming Web Services with JavaScript	59
4.6 Crossword	64
<b>5.0 AJAX Authentication Service</b>	<b>66-72</b>
5.1 Introduction to AJAX Authentication Service	67
5.2 Preparing the Application	68

<b>5.3 Login and Logout</b>	<b>69</b>
<b>5.4 Crossword</b>	<b>73</b>

## **6.0 AJAX Profile Service 74-85**

<b>6.1 Introduction to AJAX Profile Service</b>	<b>75</b>
<b>6.1 Preparing Web Site</b>	<b>76</b>
<b>6.2 Accessing Profile Data</b>	<b>80</b>
<b>6.3 Accessing Profile Group Data</b>	<b>83</b>
<b>6.4 Crossword</b>	<b>86</b>

## **7.0 Localizing and Globalizing Applications 87-92**

<b>7.1 Globalizing the Applications</b>	<b>88</b>
<b>7.2 Localizing the Applications</b>	<b>91</b>

## **8.0 Built-in support for AJAX – Server and Client 93-100**

<b>8.1 AJAX Client Architecture</b>	<b>94</b>
<b>8.2 AJAX Server Architecture</b>	<b>98</b>
<b>8.3 Crossword [Chap 7 &amp; 8 Combined]</b>	<b>101</b>

## **9.0 Using the ASP.NET AJAX Control Toolkit 102-130**

<b>9.1 Accordion and Accordion Pane Control</b>	<b>104</b>
<b>9.2 AlwaysVisible Control</b>	<b>108</b>
<b>9.3 AutoComplete Control</b>	<b>110</b>
<b>9.4 CollapsiblePanel</b>	<b>113</b>
<b>9.5 DragPanel Control</b>	<b>115</b>
<b>9.6 DropShadow Extender</b>	<b>119</b>
<b>9.7 PasswordStrength</b>	<b>122</b>
<b>9.8 RoundedCorners Extender</b>	<b>126</b>
<b>9.9 Crossword</b>	<b>131</b>










## **10.0 Advantages and Disadvantages of using AJAX in Application 133-174**




---

<b>10.1 Advantages and Disadvantages of using AJAX</b>	<b>134</b>
<b>10.2 Disadvantages of using AJAX</b>	<b>135</b>
<b>**Answers for Crosswords</b>	<b>137</b>
<b>**Contributors</b>	<b>138</b>

## Guide to Use this Workbook

### Conventions Used

Convention	Description
 <b>Topic</b>	Indicates the Topic to be discussed.
<b>Estimated Time</b>	Overview of estimated time needed to understand the Topic and complete the Practice session.
 <b>Presentation</b>	A brief introduction about the Topic.
 <b>Scenario</b>	An idea on real time situation where topics are used.
 <b>Demonstration/Code Snippet</b>	Topics implemented in real time code along with screenshots
<i>Code in Italic</i>	Italic framed lines are generated by the System related to that particular event.
// OR	Topic is being described from complete program as code snippet
 <b>Context</b>	When the Topic should be used in an application.
 <b>Practice Session</b>	Participant should be able to implement the Topic to develop an Application in practice sessions.
 <b>Check list</b>	List of brief content of the Topic.
 <b>Common Errors</b>	List of common errors occurred while developing an Application.
 <b>Exceptions</b>	List of exceptions resulted from the execution of an Application.

 <b>Lessons Learnt</b>	Lessons learnt from the article of the workbook.
 <b>Best Practices</b>	Efficient development of the an Application through best ways
 <b>Notes</b>	Important information related to the Topic

## 1.0 Introduction to AJAX

### Topics

 **1.1 Overview of AJAX**

 **1.2 Communication through AJAX**

 **1.3 Crossword**





---

**Topic: Overview of AJAX****Estimated Time: 10 Minutes**

---

**Objectives :** This module will familiarize the participant with

Basic idea of AJAX.

**Presentation :**

- AJAX stands for **Asynchronous JavaScript And XML**.
- It is an approach or pattern to web development that uses client-side scripting to exchange data with a web server. This approach enables pages to be updated dynamically without causing a full page refresh to occur (the dream, we presume, of every web developer).
- As a result, the interaction between the user and the application is uninterrupted and remains continuous and fluid.
- While some consider this approach to be a technology rather than a pattern, it's actually just a combination of related technologies, used together in a creative, new way.

**Topic: Communication through AJAX****Estimated Time: 20 Minutes****Objectives :** This module will familiarize the participant with

- **Communication model of AJAX.**
- **How AJAX works with Classical Web Applications.**

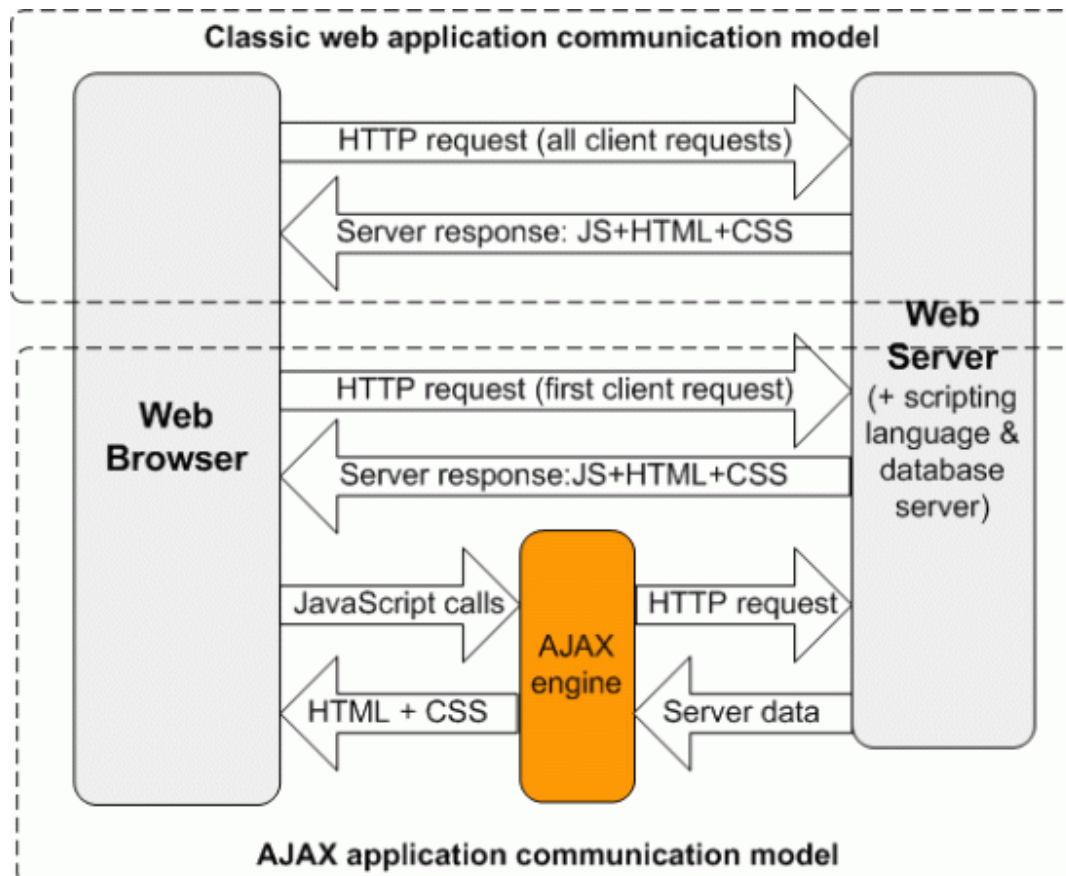
**Presentation :**

- When an application uses AJAX, a new layer is added to the communication model. In the classic web application, communication between the client (the browser) and the web server were performed directly, using HTTP requests.
- In Classic web application, when the visitor requests a page, the server will send the full HTML and CSS code at once. After the visitor fills in a form and submits it, the server processes the information and rebuilds the page. It then sends the full page back to the client. And so on.
- Whereas when using AJAX, the page is loaded entirely only once, the first time it is requested. Besides the **HTML** and **CSS** code that make up the page, some **JavaScript** files are also downloaded: i.e., the **AJAX engine**.
- All requests for data to the sever will then be sent as JavaScript calls to this engine. The AJAX engine then requests information from the web server **asynchronously**.
- Thus, only small page bits are requested and sent to the browser, as they are needed by the user. The engine then displays the information without reloading the entire page.
- This leads to a much more responsive interface, because only the necessary information is passed between the client and server, not the whole page. This produces the feeling that information is displayed immediately, which brings web applications closer to their desktop relatives.



### Demonstration/Code Snippet :

To better illustrate the communication between the client (browser) and the remote server, as well as the differences between the classic and the AJAX-powered applications, take a look at the diagram below:



*Illustration 1: Classic and AJAX communication models*

- At the heart of the AJAX method of communicating with the server lies the AJAX engine. This is nothing more than some JavaScript code that instantiates and uses the XMLHttpRequest object.
- This is a JavaScript object that allows sending, receiving and processing HTTP requests to and from the server without refreshing the entire page.
- In AJAX-powered applications, HTTP requests for data can be made completely in the background, without the user experiencing any interruptions. This means the user can

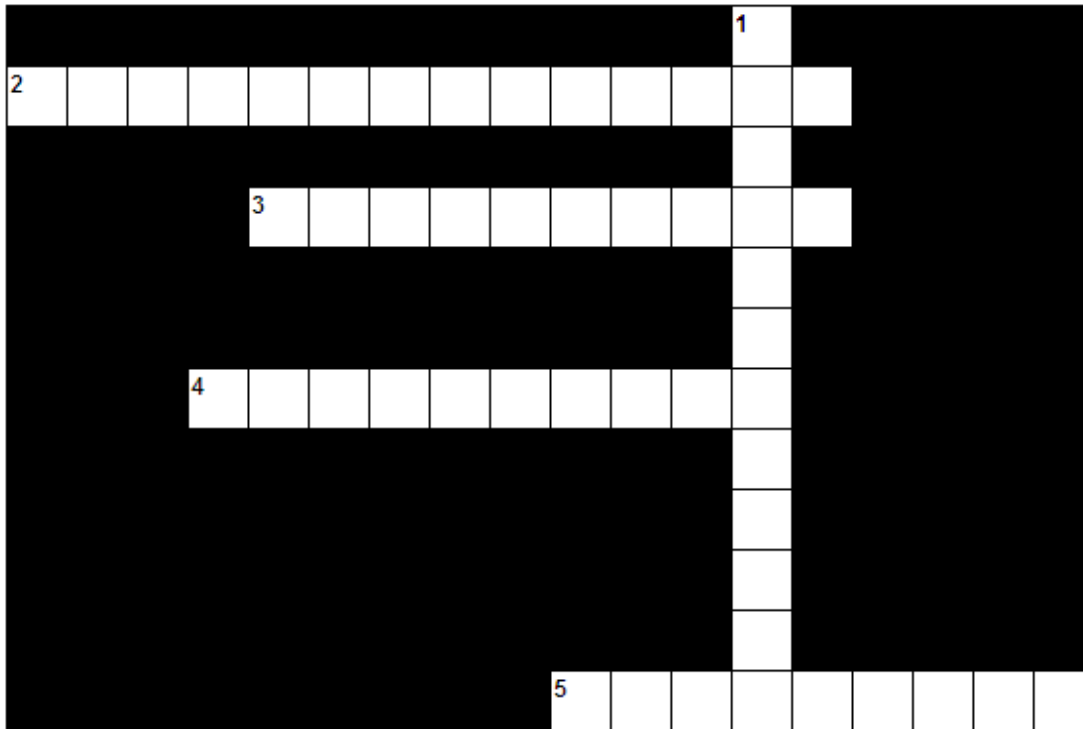
---

continue working and using the application, while the necessary page sections are received from the server.

- The **XMLHttpRequest** object was implemented as an **ActiveX object** in Internet Explorer, and has later become a native JavaScript object in most modern browsers (FireFox, Safari).
- Although adding an extra layer to any kind of model should add to the response time, this is an exception. Through the use of this new layer – the AJAX engine – response time shortens and the user interface seems much more connected to the application logic. Moreover, the user no longer has to wait around for the page to load.



HTTP is the acronym for Hyper Text Transfer Protocol, the communication protocol used throughout the Internet.

**Across:**

2. With AJAX, Java Script communicate directly with the server using this object.[XMLHTTPREQUEST]
3. When using AJAX, first time when a page is requested, the JavaScript files that are downloaded, is known as\_\_\_\_\_.[AJAXENGINE]
4. First time, the HTTP request is send by \_\_\_\_\_. [WEBBROWSER]
5. Applying AJAX, load is being reduced from\_\_\_\_\_. [WEBSERVER]

**Down:**

1. Through AJAX, data transfer is being done \_\_\_\_\_ly.[ASYNCHRONOUS]

## 2.0 Updating the Page

### Topics

 **2.1 Using the UpdatePanel Control**

 **2.2 Using the UpdateProgress Control**

 **2.3 Crossword**



**Topic: Using the UpdatePanel Control****Estimated Time: 20 Minutes****Objectives :** This module will familiarize the participant with

- Concept of Partial Page Update.
- How to use Update Panel Control step by step.

**Presentation :**

- Here we will build an application that displays pages of data from a sample database.
- The application uses the UpdatePanel control to refresh only the part of the page that has changed, without the page flash that occurs with a postback. This is referred to as a partial-page update.

To implement the procedures in your own development environment you need at least:

- i) Microsoft Visual Studio 2005 or Microsoft Visual Web Developer Express Edition.
- ii) The latest release of Microsoft ASP.NET AJAX installed and configured. For more information, see Installing ASP.NET AJAX.
- iii) An ASP.NET AJAX Web site.
- iv) A sample data base named “**Test**” which has a table named ‘**MyTable**’ consists of at least 25 rows(preferably).It is expected to be created by the user.

**Demonstration/Code Snippet :****Creating an ASP.NET AJAX-Enabled Web Site**

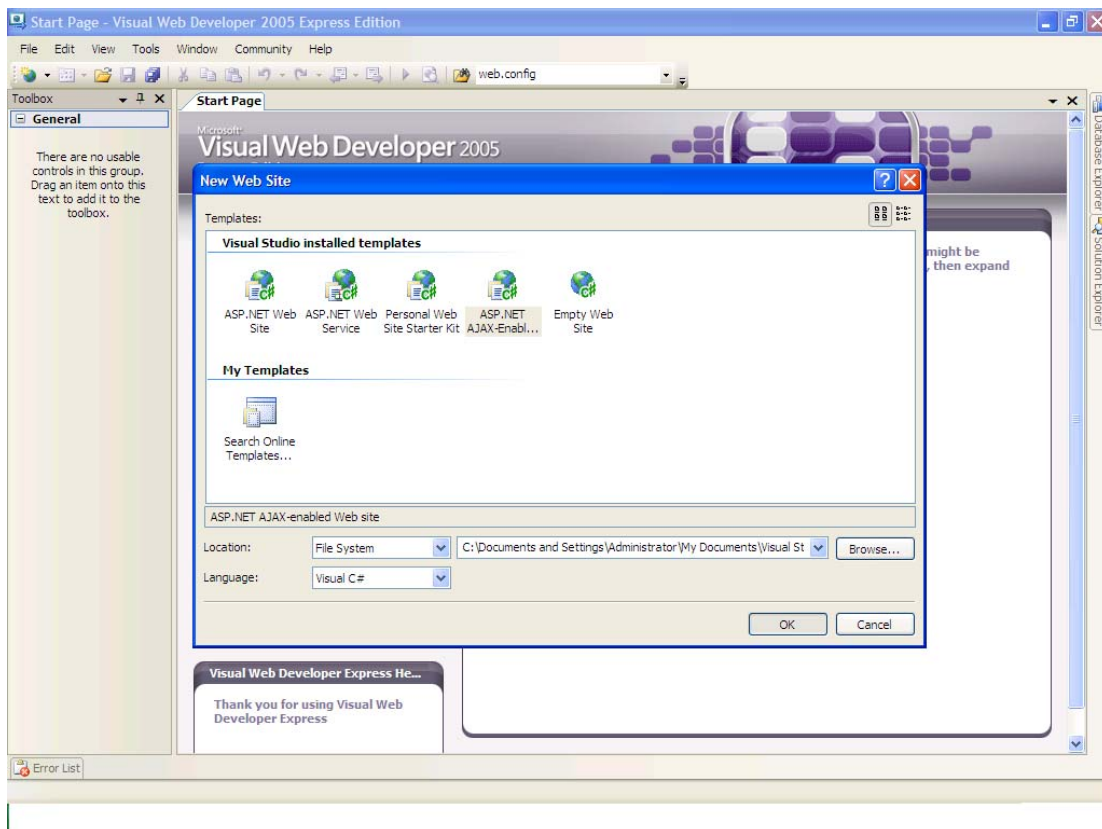
You can create ASP.NET AJAX-enabled Web sites in Visual Studio by using the template installed with ASP.NET AJAX. To create an ASP.NET AJAX-enabled web-site:

1. Start Visual Studio.

2. In the File menu, click New Web Site.

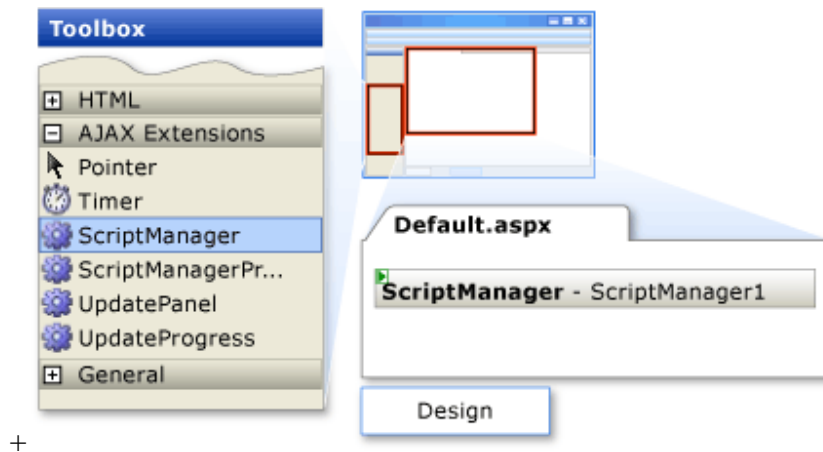
The New Web Site dialog box is displayed.

3. Under Visual Studio installed templates, select ASP.NET AJAX-Enabled Web Site.
4. Enter a location and a language, and then click OK.

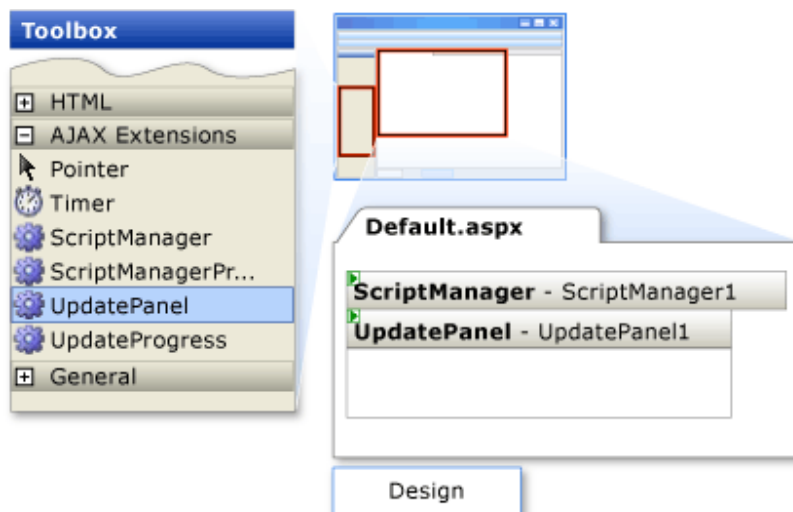


- Now comes adding an UpdatePanel Control to the ASP.NET Web Page
- Before you add an **UpdatePanel** control to the page, you must add a **Script Manager** control. The **UpdatePanel** control relies on the **Script Manager** control to manage partial-page updates. To do that, follow the below steps:
  1. Switch to Design view.
  2. In the AJAX Extensions tab of the toolbox, double-click the **ScriptManager** control to add it to the page.





3. Drag an **UpdatePanel** control from the toolbox and drop it underneath the **ScriptManager** control.



### Adding Content to an UpdatePanel Control

The **UpdatePanel** control performs partial-page updates and identifies content that is updated independently of the rest of the page. In this part of the tutorial, you will add a data-bound control that displays data from your test database.

#### To add content to an UpdatePanel control

1. From the Data tab of the toolbox, drag a **GridView** control into the editable area of the **UpdatePanel** control.

2. In the GridView Tasks menu, click Auto Format.
3. In the Auto Format panel, under Select a scheme, select Colorful and then click OK.
4. In the GridView Tasks menu, select <New data source> from the Choose Data Source list.

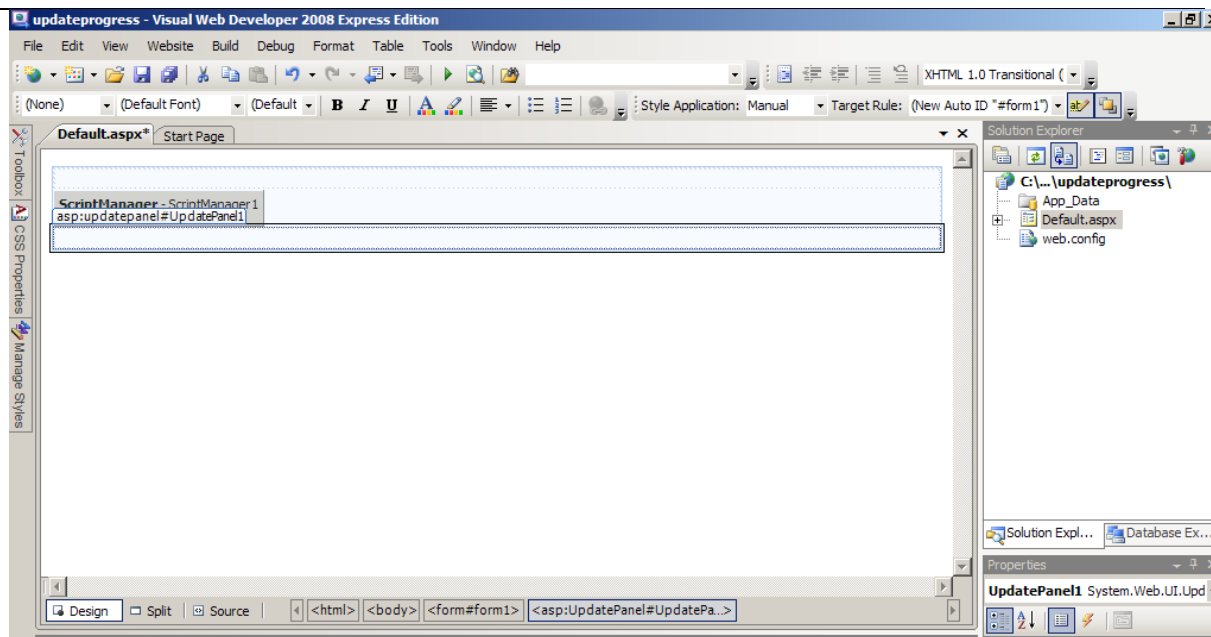
The Data Source Configuration wizard is displayed.

5. Under Where will the application get data from, select your Sample **Database** and then click OK.
6. In the Configure Data Source wizard, for the Choose Your Data Connection step, configure a connection to your “TEST” database and then click Next.
7. For the Configure the Select Statement step, select Specify a custom SQL statement or stored procedure and then click Next.
8. In the SELECT tab of the Define Custom Statement or Stored Procedures step, enter the following SQL statement:

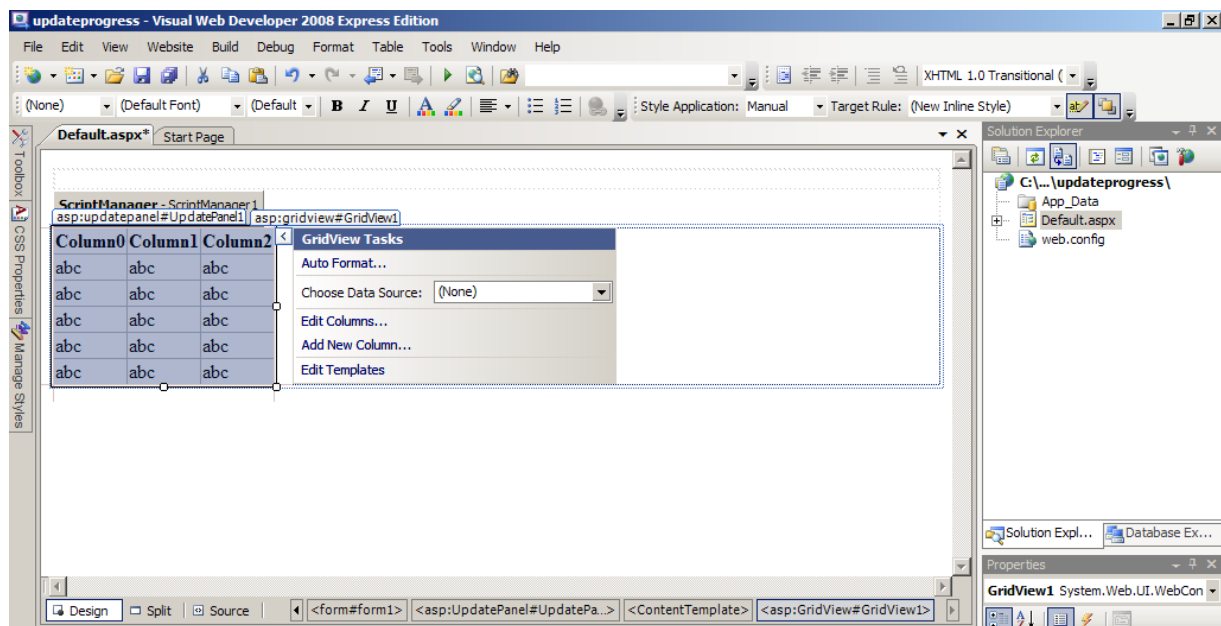
Select \* from tablename;

9. Click Next.
10. Click Finish.
11. In the GridView Tasks menu, select the Enable paging check box.
12. Save your changes, and then press CTRL+F5 to view the page in a browser.

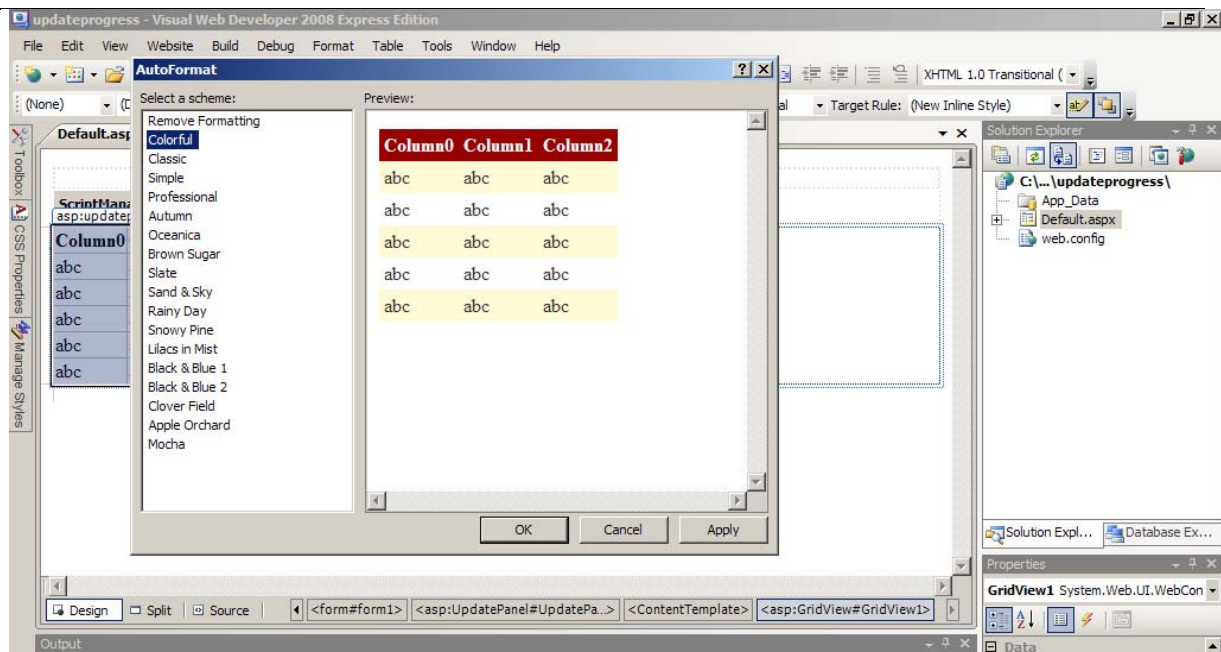
Notice that there is no page flash when you select different pages of data. This is because the page is not performing a postback and updating the whole page every time.



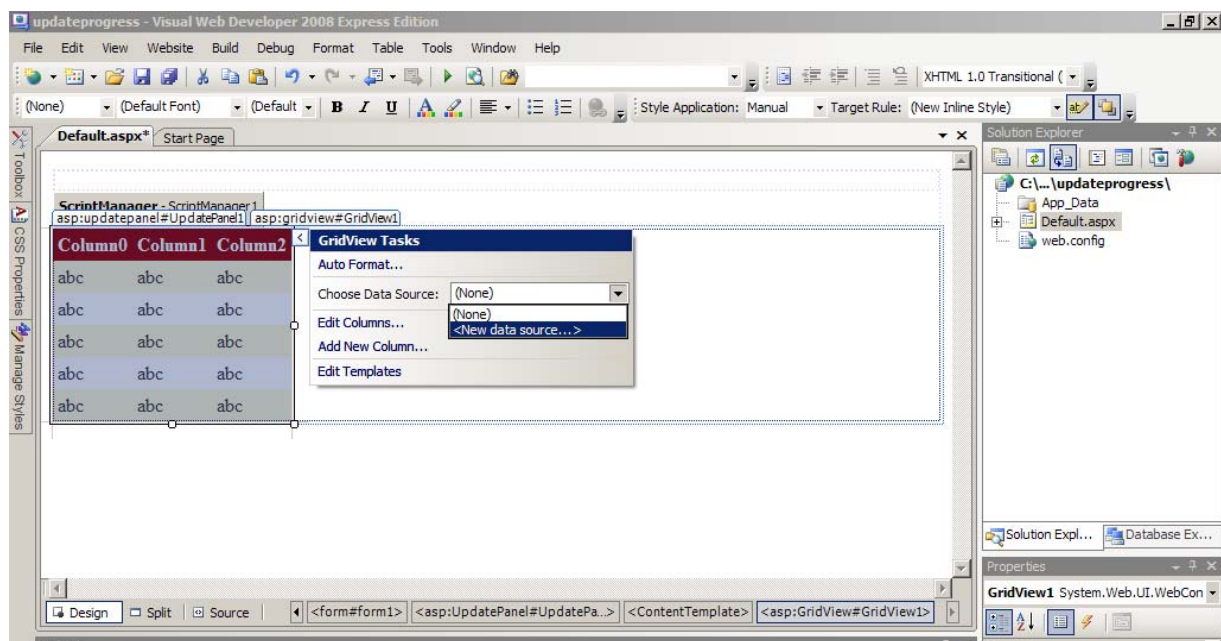
## 1.Designing



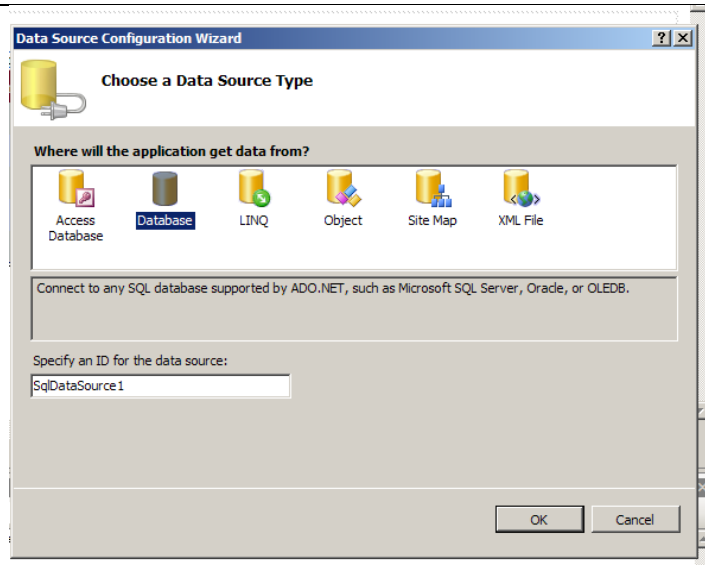
2. GridView control is added.



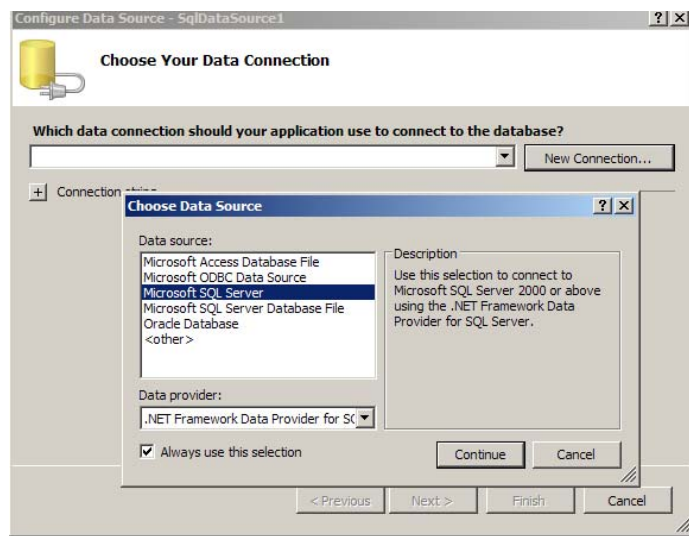
#### 4. Changing properties of GridView Control.



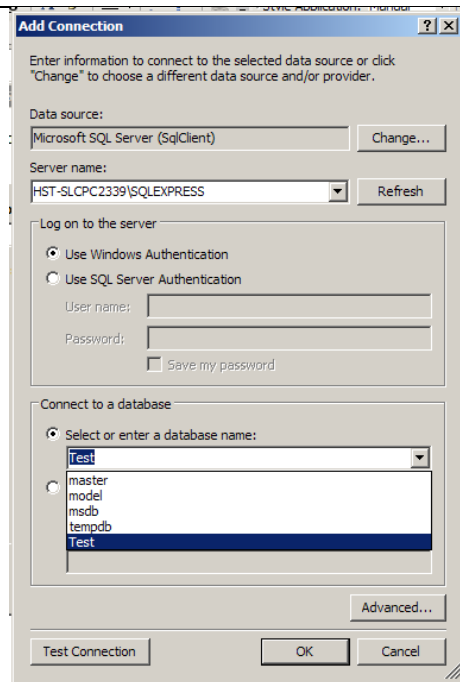
#### 5. Relating the GridView with our sample Test database.



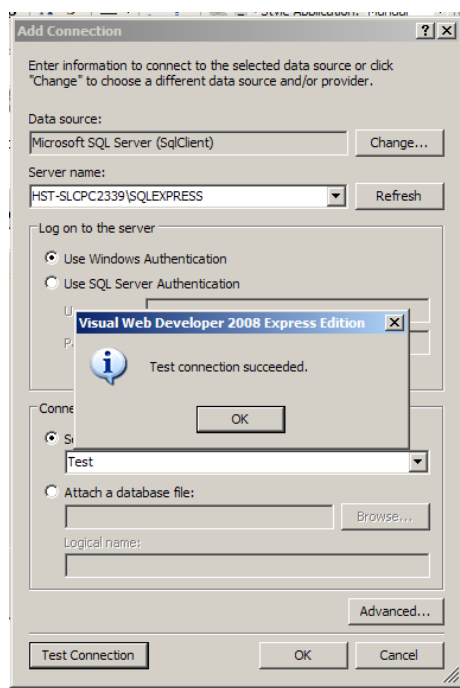
6. Relating the GridView with our sample Test database.



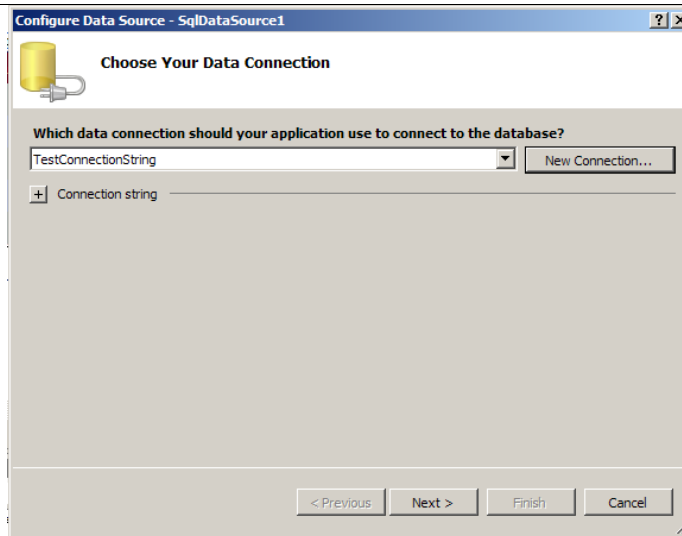
7. Relating the GridView with our sample Test database.



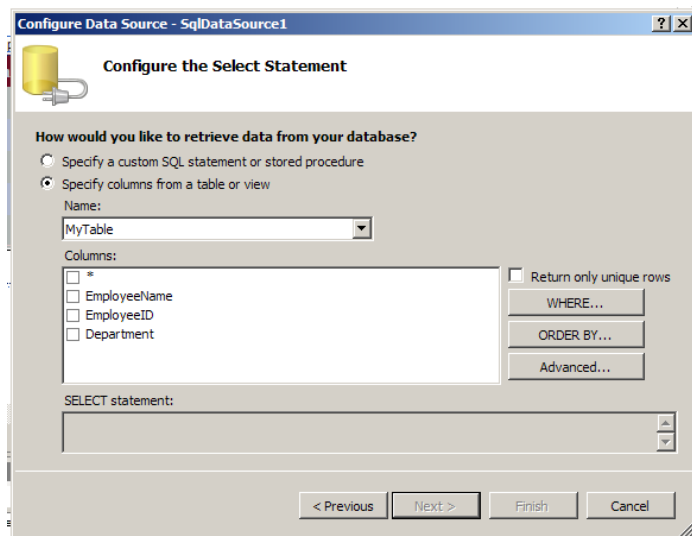
8. Relating the GridView with our sample Test database.



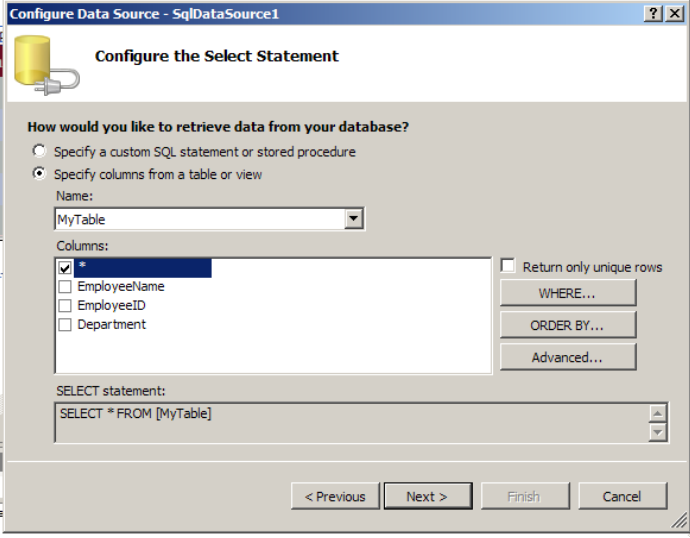
9. Relating the GridView with our sample Test database.



10. Relating the GridView with our sample Test database.



11. Choosing the table MyTable from Test database.



**Configure Data Source - SqlDataSource1**

**Configure the Select Statement**

How would you like to retrieve data from your database?

- ☐ Specify a custom SQL statement or stored procedure
- ☒ Specify columns from a table or view

Name:

Columns:

- ☒ \*
- ☐ EmployeeName
- ☐ EmployeeID
- ☐ Department

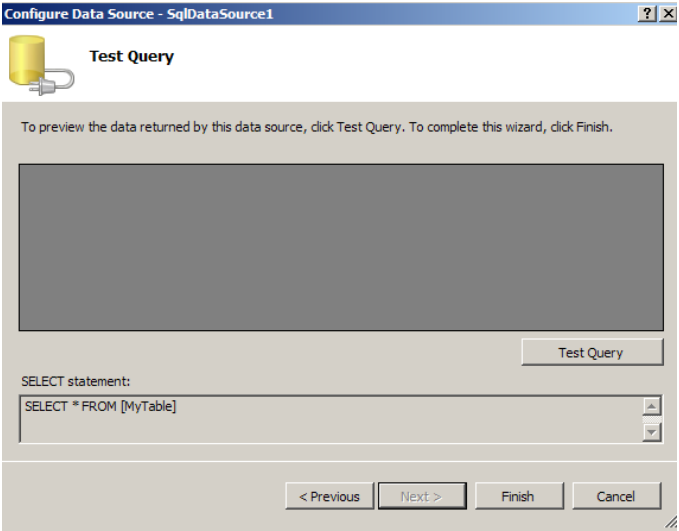
☐ Return only unique rows

WHERE...  
ORDER BY...  
Advanced...

SELECT statement:

< Previous   Next >   Finish   Cancel

12. Setting the query on MyTable on Test database.



**Configure Data Source - SqlDataSource1**

**Test Query**

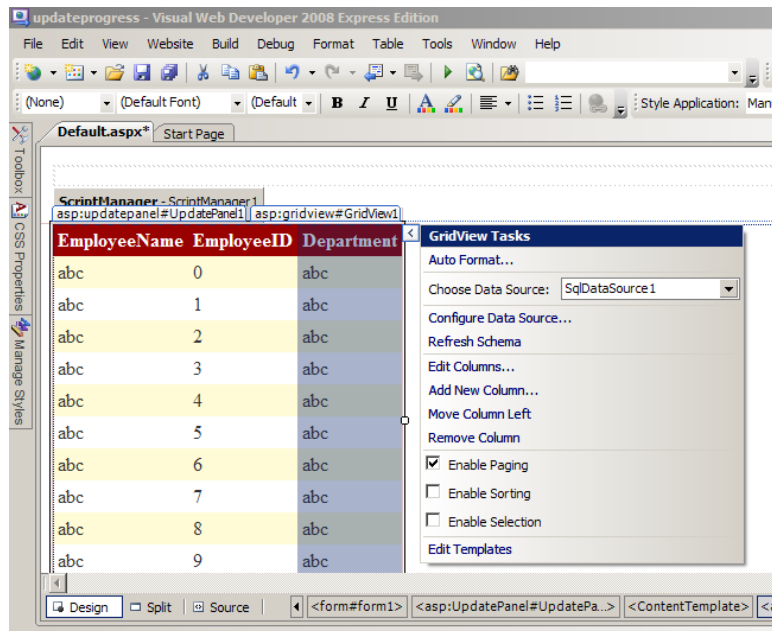
To preview the data returned by this data source, click Test Query. To complete this wizard, click Finish.

SELECT statement:

< Previous   Next >   Finish   Cancel

13. Click Finish or you can test your query here!





#### 14. Enable Paging in GridView.



### 15. Running the application.[first page]

Untitled Page - Windows Internet Explorer

http://localhost:1239/updateprogress/Default.aspx

File Edit View Favorites Tools Help

Untitled Page

EmployeeName	EmployeeID	Department
Sujoy	11	IT
Avijit	12	N&S
Pratik	13	Manager
Devika	14	Sales
Onnesa	15	TechnicalWriter
Chandra	16	R&D
Minoti	17	Sales
Pamela	18	Creative Writer
Madhu	19	Tester
Madhulina	20	Developer

1 2 3

### 16. Second page.

Untitled Page - Windows Internet Explorer

http://localhost:1239/updateprogress/Default.aspx

File Edit View Favorites Tools Help

Untitled Page

EmployeeName	EmployeeID	Department
Madhurima	21	Sales
Aparna	22	Marketting
Antara	23	Marketting
Arghya	24	N&S
Ananda	25	TechSupport

1 2 3

### 17. Third Page.

**Topic: Using the UpdateProgress Control****Estimated Time: 20 Minutes****Objectives :** This module will familiarize the participant with

- The basic idea of UpdateProgress Control and how to use it.
- How to add delay in the application to see the text of the UpdateProgress Control properly.

**Presentation :**

- The UpdateProgress control displays a status message while new content for an UpdatePanel control is being requested. Please note that here the UpdateProgress Control is used in addition with UpdatePanel Control example stated above.

**Scenario :**

Mr Rudlof Gatting, wants to know his companies employee details, but while doing so the fact this information is distributed among several pages makes him irritated as it takes some seconds to go to the next page. So he was advised to use UpdateProgress control in the page so that at least he gets a message that the information's are being fetched.

**Demonstration/Code Snippet :****To Add an UpdateProgress Control to the Page:**

1. From the AJAX Extensions tab of the toolbox, drag an UpdateProgress control onto the page and drop it underneath the UpdatePanel control.
2. Select the UpdateProgress control, and in the Properties window, set the **AssociatedUpdatePanelID** property to UpdatePanel1 . This associates the UpdateProgress control with the UpdatePanel control that you added previously.

3. In the editable area of the UpdateProgress control, type “Fetching Informations . . . . .”
4. Save your changes, and then press CTRL+F5 to view the page in a browser.

If there is a delay while the page runs the SQL query and returns the data, the UpdateProgress control displays the message that you entered into the UpdateProgress control.

### Adding a Delay to the Sample Application

If your application updates each page of data quickly, you might not see the content of the Update Progress control on the page. The Update Progress control supports a Display After property that enables you to set a delay before the control is displayed. This prevents the control from flashing in the browser if the update occurs very fast. By default, the delay is set to 500 milliseconds (.5 second), meaning that the Update Progress control will not be displayed if the update takes less than half a second.

In a development environment, you can add an artificial delay to your application to make sure that the Update Progress control is functioning as intended. This is an optional step and is only for testing your application.

To add delay to the sample application:

1. Inside the Update Panel control, select the Grid View control.
2. In the Properties window, click the Events button.
3. Double-click the PageIndexChanged event to create an event handler.
4. Add the following code to the PageIndexChanged event handler to artificially create a three-second delay:

CS

```
Include three second delay for example only.System.Threading.Thread.Sleep (3000);
```

VB

Include three second delay for example only.

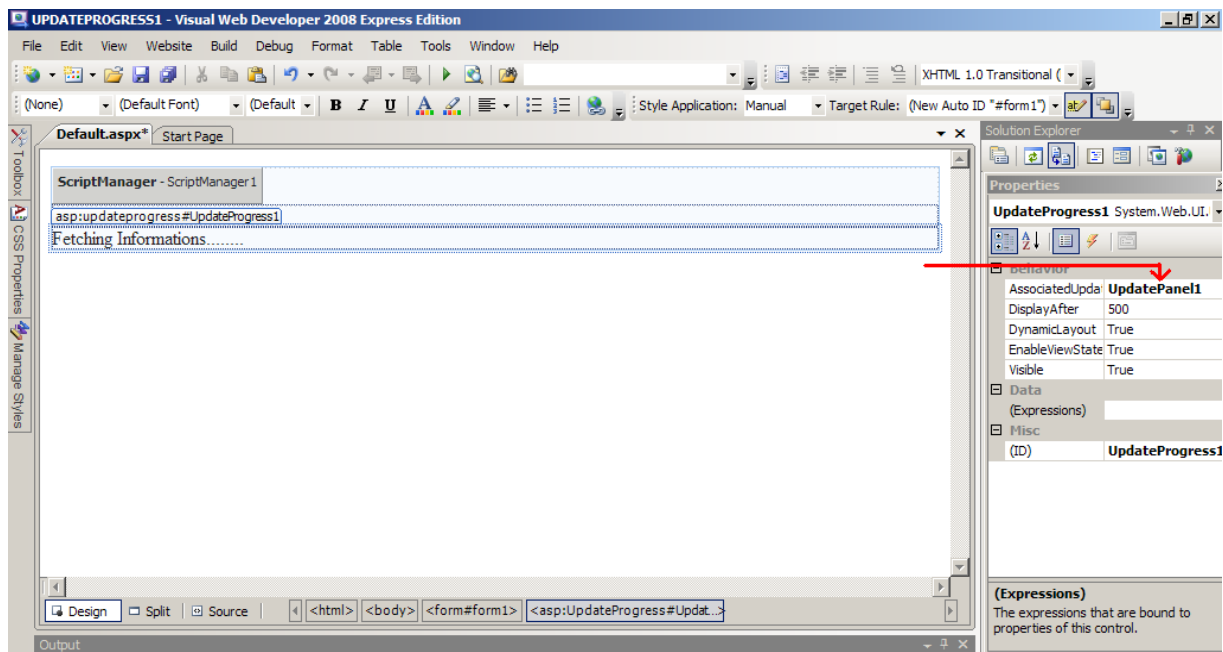
```
System.Threading.Thread.Sleep (3000);
```

5. Save your changes, and then press CTRL+F5 to view the page in a browser. Because there is now a three-second delay every time that you move to a new page of data, you will be able to see the text of UpdateProgress control.

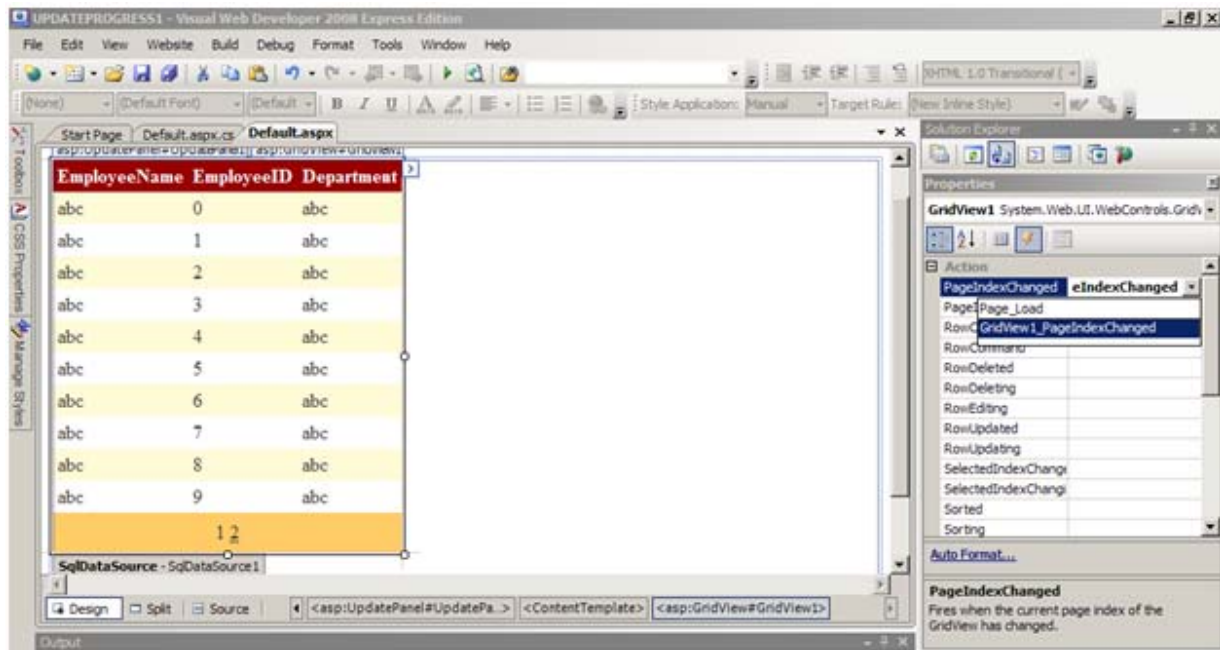


The handler for the PageIndexChanged event intentionally introduces a delay for this tutorial. In practice, you would not introduce a delay. Instead, the delay would be the result of server traffic or of server code that takes a long time to process, such as a long-running database query.

## Screenshots:



1. After designing the page.



2. Creating the GridView1\_PageIndexChanged event.



2. Running the application. Page is loaded for the first time.

Untitled Page - Windows Internet Explorer

http://localhost:1777/UPDATEPROGRESS1/Default.aspx

File Edit View Favorites Tools Help

Untitled Page

EmployeeName	EmployeeID	Department
Avishek	1	IT
Anindita	2	Developer
Ashwani	3	Manager
Nehera	4	Sales
Asin	5	Marketing
Asit	6	Developer
Ankit	7	Tester
Arijit	8	IT
Arnab	9	Manager
Santana	10	IT

1 2 3

Fetching Informations.....

3. Clicking on page Index 2 gives this output; the content of UpdateProgress is visible until the next page is appeared.

Untitled Page - Windows Internet Explorer

http://localhost:1777/UPDATEPROGRESS1/Default.aspx

File Edit View Favorites Tools Help

Untitled Page

EmployeeName	EmployeeID	Department
Sujoy	11	IT
Avijit	12	N&S
Pratik	13	Manager
Devika	14	Sales
Onnesa	15	TechnicalWriter
Chandra	16	R&D
Minoti	17	Sales
Pamela	18	Creative Writer
Madhu	19	Tester
Madhulina	20	Developer

1 2 3

4. Page2 has come, content of UpdateProgress is gone.

Untitled Page - Windows Internet Explorer

http://localhost:1777/UPDATEPROGRESS1/Default.aspx

File Edit View Favorites Tools Help

Untitled Page

EmployeeName	EmployeeID	Department
Sujoy	11	IT
Avijit	12	N&S
Pratik	13	Manager
Devika	14	Sales
Onnesa	15	TechnicalWriter
Chandra	16	R&D
Minoti	17	Sales
Pamela	18	Creative Writer
Madhu	19	Tester
Madhurima	20	Developer

1 2 3

Fetching Informations.....

5. Similarly for picture 5 and 6.

Untitled Page - Windows Internet Explorer

http://localhost:1777/UPDATEPROGRESS1/Default.aspx

File Edit View Favorites Tools Help

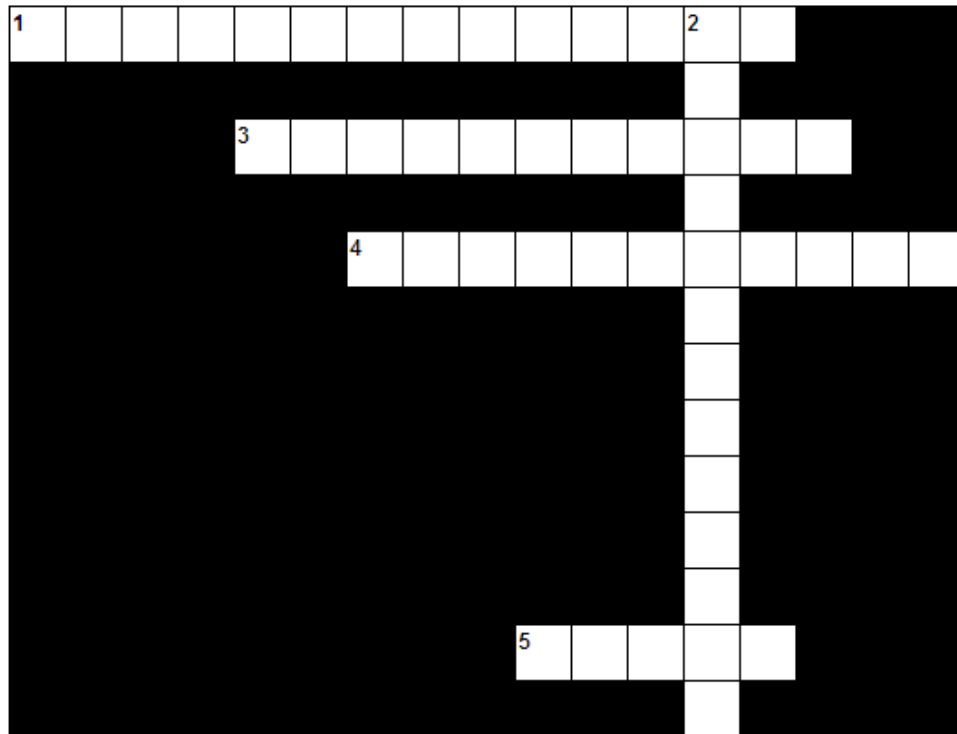
Untitled Page

EmployeeName	EmployeeID	Department
Madhurima	21	Sales
Aparna	22	Marketing
Antara	23	Marketing
Arghya	24	N&S
Ananda	25	TechSupport

1 2 3

6. In continuation with previous pics.



**Crossword: Unit-2****Estimated Time:10 Minutes****Across:**

1. We use this control in addition with UpdatePanel to display status message\_\_\_\_\_.[UpdateProgress]
3. By default, in UpdateProgress Control,the delay is set to \_\_\_\_\_ milliseconds.[FIVEHUNDRED]
4. To avoid full page postback, we use this control \_\_\_\_\_.[UPDATEPANEL]
5. In C#, the code for adding delay of 3 seconds is System.Threading.Thread.\_\_\_\_\_  
(3000);[SLEEP]

**DOWN:**

3. The control you must add at first to run an AJAX application. [SCRIPTMANAGER].

## 3.0 Other AJAX Extensions

### Topics

 **3.1 Using Timer Control**

 **3.2 Using ScriptManagerProxy**





**Objectives :** This module will familiarize the participant with

- Basic idea of Timer Control and how to use it for setting intended interval.



**Presentation :**

To refresh an UpdatePanel Control at a timed interval:

Here we will update part of a Web page at a timed interval by using three Microsoft ASP.NET 2.0 AJAX Extensions server controls: the ScriptManager control, the UpdatePanel control, and the Timer control. Adding these controls to a page eliminates the need to refresh the whole page with each postback. Only the contents of the UpdatePanel control will be updated.



**Scenario :**

Mr Ketan Saxena, who is a share broker, wants to build his own share trading site. So he was advised by the experts that he can go for an AJAX enabled web site application where he can update/refresh his page informations, i.e., share prices after certain intended intervals.

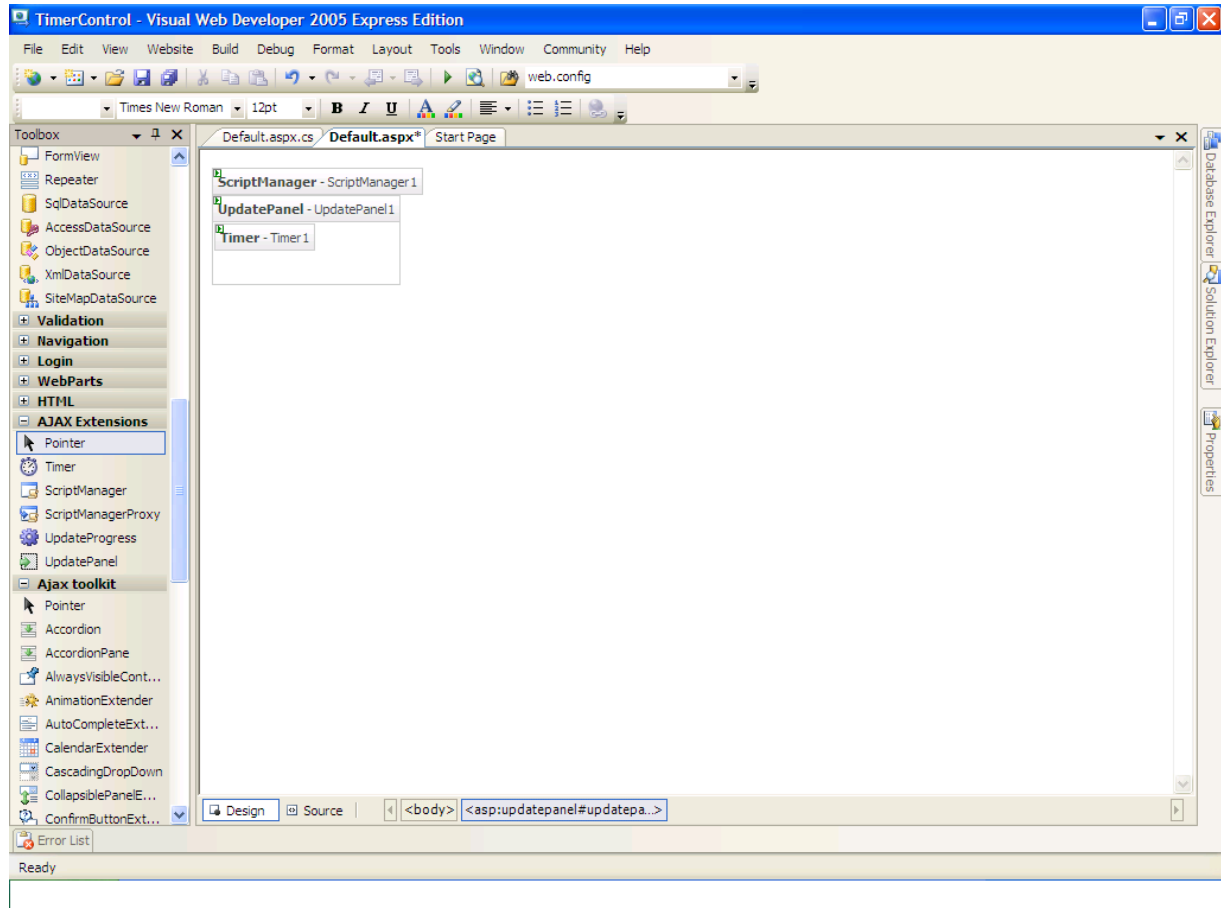


**Demonstration/Code Snippet :**

So let's start our journey with Timer Control:

1. Create a new page and switch to Design view.
2. If the page does not already contain a **ScriptManager** control, in the AJAX Extensions tab of the toolbox, double-click the **ScriptManager** control to add it to the page.
3. In the AJAX Extensions, double-click the **UpdatePanel** control to add it to the page.

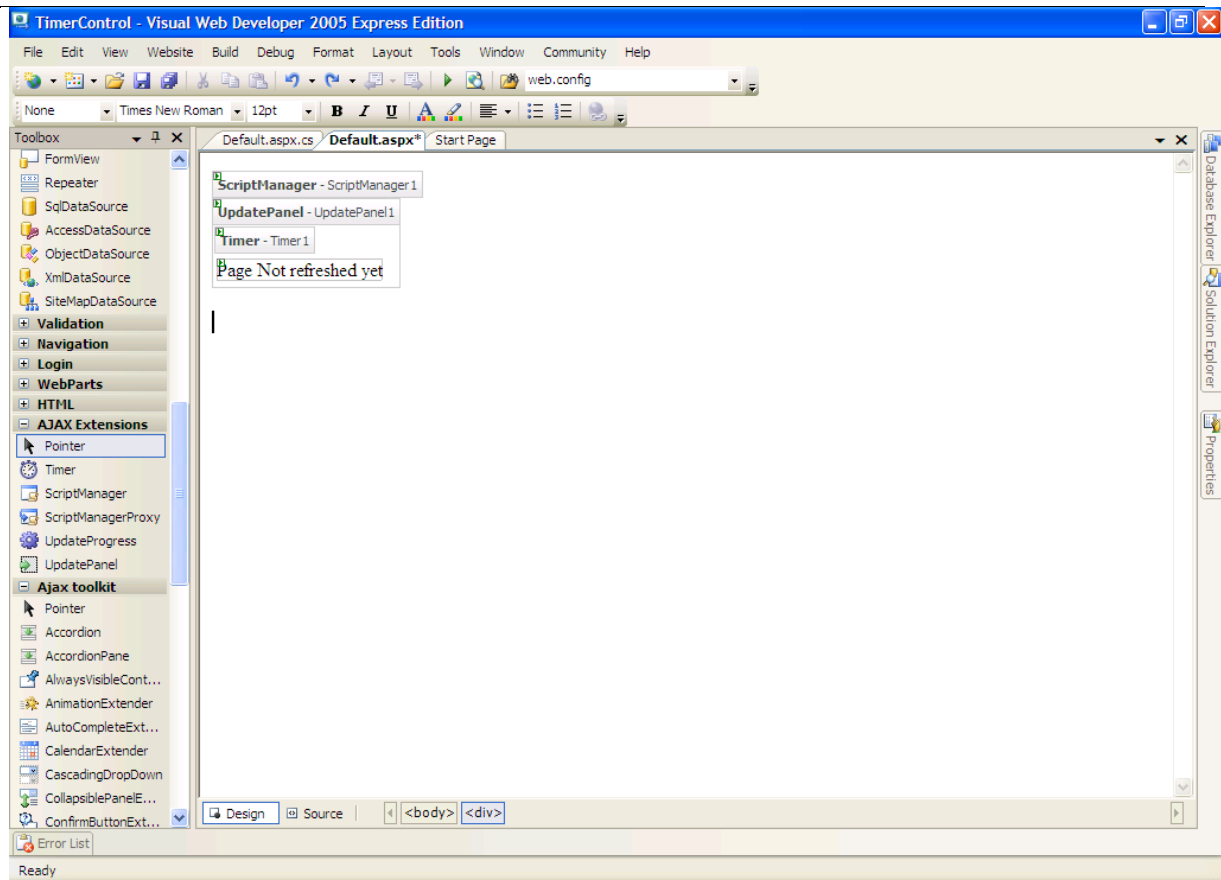
4. Click inside the **UpdatePanel** control and then double-click the **Timer** control to add it to the **UpdatePanel** control.



5. Set the **Interval** property of the **Timer** control to 10000.

The **Interval** property is defined in milliseconds, so that setting the **Interval** property to 10,000 milliseconds will refresh the **UpdatePanel** control every 10 seconds.

6. Click inside the **UpdatePanel** control and then in the Standard tab of the toolbox, double-click the **Label** control to add it to the **UpdatePanel** control.
7. click the **Label** control to add it to the **UpdatePanel** control.
8. Set the label's Text property to Panel not refreshed yet.



9. Click outside the **UpdatePanel** control and double-click the **Label** control to add a second label outside the **UpdatePanel** control.

#### note

Make sure that you add the second **Label** control outside the **UpdatePanel** control.

10. Double-click the **Timer** control to create a handler for the **Tick** event.
11. Add code that sets the Text property of the Label1 control to the current time.
12. Create a Page\_Load handler and add code that sets the Text property of the Label2 control to the time that the page is created.
13. Switch to Source view.

---

Make sure that the markup for the page resembles the following:

*cs*

```
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        Label2.Text = "Page created at: " +
            DateTime.Now.ToLongTimeString();
    }
    protected void Timer1_Tick(object sender, EventArgs e)
    {
        Label1.Text = "Panel refreshed at: " +
            DateTime.Now.ToLongTimeString();
    }
}
```

*vb*

```
Partial Class _Default
    Inherits System.Web.UI.Page

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
        Handles Me.Load
        Label2.Text = "Page created at: " & _
```

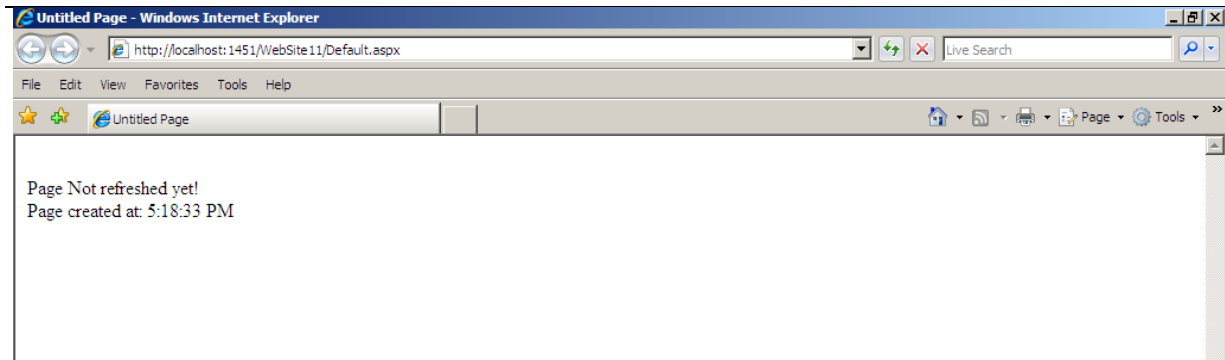
```
DateTime.Now.ToLongTimeString()  
  
End Sub  
  
Protected Sub Timer1_Tick(ByVal sender As Object, ByVal e As System.EventArgs)  
    Label1.Text = "Panel refreshed at: " & _  
        DateTime.Now.ToLongTimeString()  
  
End Sub  
  
End Class
```

14. Save your changes and press CTRL+F5 to view the page in a browser.

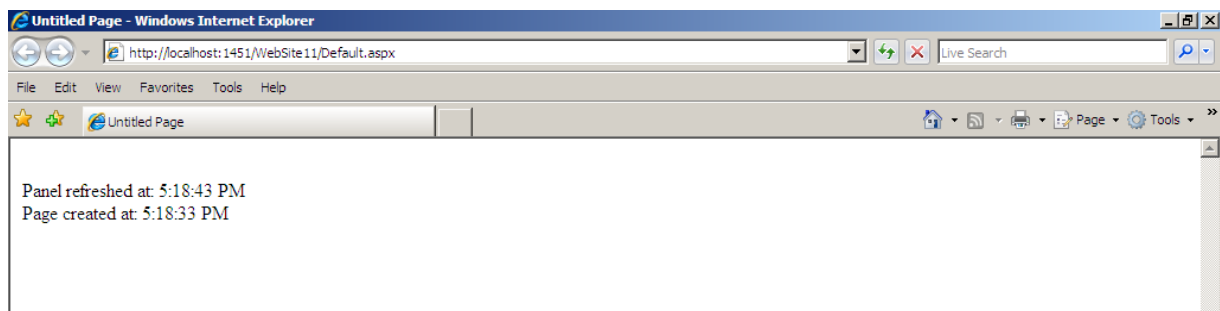
15. Wait at least 10 seconds for panel to refresh.

The text inside the panel changes to display the last time that the panel's content was refreshed. However, the text outside the panel is not refreshed.

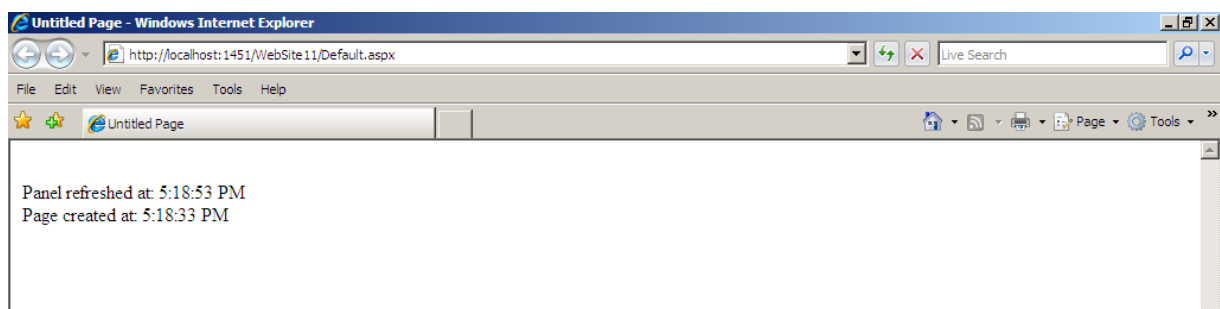
#### **ScreenShots:**



Pic1:Initially page is just created, not refreshed!



Pic2:Page is refreshed for the first time!



Pic 3:Page is again refreshed,notethe interval with the last pic!



1. In this example, the timer interval is set to 10 seconds. That way, when you run the example, you do not have to wait a long time to see the results. However, each timer interval



causes a post-back to the server and causes network traffic. Therefore, in production applications, you should set the interval to the longest time that is still practical for your application.

2. Make sure that you add the Label control inside the UpdatePanel control.
3. Make sure that you add the Label control inside the UpdatePanel control.

**Topic: Using the ScriptManagerProxy Control****Estimated Time: 20 Mins.****Objectives :** This module will familiarize the participant with

- Basic idea of ScriptManagerProxy Control and how to use it content pages.

**Presentation :****The ScriptManagerProxy Class**

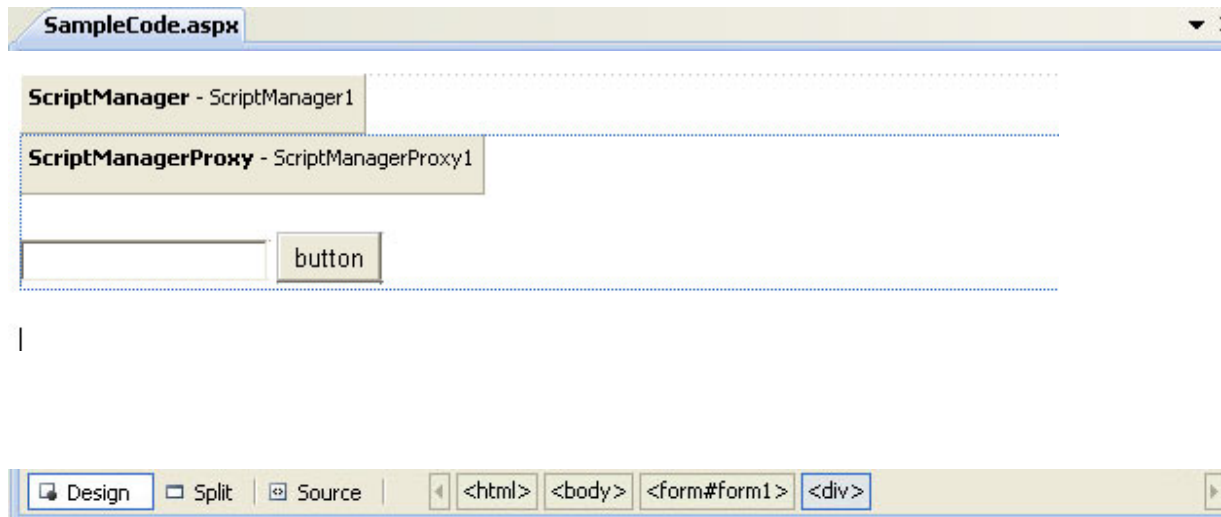
Only one instance of the ScriptManager control can be added to the page. The page can include the control directly or indirectly inside a nested component such as a user control, content page for a master page, or nested master page. **In cases where a ScriptManager control is already on the page but a nested or parent component needs additional features of the ScriptManager control, the component can include a ScriptManagerProxy control.** For example, the ScriptManagerProxy control enables you to add scripts and services that are specific to nested components.

**Class Reference**

<b>Class</b>	<b>Description</b>
<a href="#">ScriptManager</a>	A server control that makes script resources available to the browser, including the Microsoft AJAX Library and the functionality that enables partial-page rendering.
<a href="#">ScriptManagerProxy</a>	A server control that enables nested components to add script and service references if the page already contains a <a href="#">ScriptManager</a> control.

Learn how a ScriptManagerProxy enables a content page to pass references to the ScriptManager placed on its ASP.NET, allowing each content page to define its own AJAX behavior.

## 1. How-to-use-ScriptManagerProxy-c.aspx (Design Page)



### Scenario :

Mr Ketan Saxena, who is a share broker, wants to build his own share trading site. He had masterpage which contain ScriptManager in it and wants to invoke a Webservice in the descendent webpage of master page. So he was advised by the experts that he can go for an ScriptManagerProxy to incorporate the same.



### Demonstration/Code Snippet :

## 2. How-to-use-ScriptManagerProxy-c.aspx (Source Page)

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="how-to-use-ScriptManagerProxy-c.aspx.cs" Inherits="how_to_use_ScriptManagerProxy_c" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>How to Use ScriptManagerProxy</title>
</head>
<script language="javascript" type="text/javascript">
<!--

function Button1_onclick() {
reqviaprovy = SimpleService.SayHello(
  document.getElementById("Text1").value,
  OnComplete,
  OnTimeout);
return false;
}

function OnComplete(results)
{
alert(results);
}
function OnTimeout(results)
{
alert("Timout");
}
// -->
</script>
<body>
  <form id="form1" runat="server">
    <asp:ScriptManager ID="ScriptManager1" runat="server" />
    <asp:UpdatePanel runat="server" id="UpdatePanel1">
      <ContentTemplate>
        <asp:ScriptManagerProxy id="ScriptManagerProxy1" runat="server">
          <Services>
            <asp:ServiceReference Path="SimpleService.asmx" />
          </Services>
        </asp:ScriptManagerProxy>
      <br />
      <input id="Text1" type="text" />
      <input id="Button1" type="button" value="button" onclick="return Button1_onclick()" />
    </ContentTemplate>
  </asp:UpdatePanel>
</form>
</body>
</html>
```

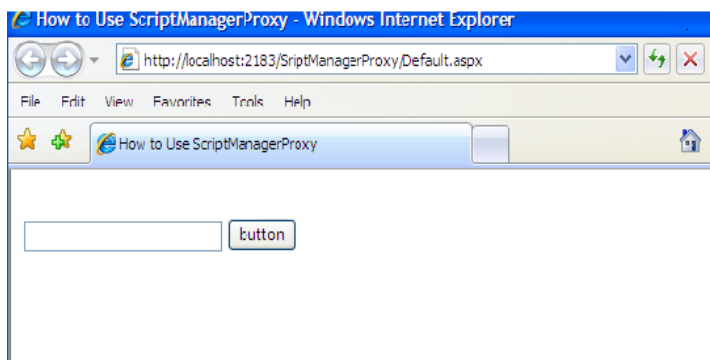
### 3. How-to-use-ScriptManagerProxy-c.aspx.cs (Code Behind C# Language)

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;

public partial class how_to_use_ScriptManagerProxy_c : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {



    }
}
```

### 4. How-to-use-ScriptManagerProxy-c.aspx (View in Browser)



## 4.0 Web Services

### Topics

-  **4.1 Introduction**
-  **4.2 Error Handling**
-  **4.3 Page Methods**
-  **4.4 Maintaining Session State**
-  **4.5 Exchanging Complex Data with Server**
-  **4.6 Consuming WebServices with Java Script**
-  **4.7 Crossword[Chap 3 & 4 Combined]**



**Objectives :** This module will familiarize the participant with

Basic idea of Web Services and creating AJAX enabled WebServices.

**Presentation :**

Web Services are an integral part of the .NET framework that provides a cross-platform solution for exchanging data between distributed systems. Although Web Services are normally used to allow different operating systems, object models and programming languages to send and receive data, they can also be used to dynamically inject data into an ASP.NET AJAX page or send data from a page to a back-end system. All of this can be done without resorting to postback operations.

While the ASP.NET AJAX UpdatePanel control provides a simple way to AJAX enable any ASP.NET page, there may be times when you need to dynamically access data on the server without using an UpdatePanel. Here we'll see how to accomplish this by creating and consuming Web Services within ASP.NET AJAX pages.

**Creating Ajax-Enabled Web Services**

The ASP.NET AJAX framework provides several different ways to call Web Services. We can use the AutoCompleteExtender control (available in the ASP.NET AJAX Toolkit) or JavaScript. However, before calling a service you have to AJAX-enable it so that it can be called by client-script code.

Visual Studio .NET 2008 has built-in support for creating .asmx Web Service files and automatically derives associated code beside classes from the System.Web.Services.WebService class. As you add methods into the class you must apply the WebMethod attribute in order for them to be called by Web Service consumers.

Here is an example of applying the WebMethod attribute to a method named GetCustomersByCountry ().

```
[WebMethod]
public Customer [] GetCustomersByCountry(string country)
{
    return Biz.BAL.GetCustomersByCountry (country);
}
```

The GetCustomersByCountry() method accepts a country parameter and returns a Customer object array. The country value passed into the method is forwarded to a business layer class which in turn calls a data layer class to retrieve the data from the database, fill the Customer object properties with data and return the array.

### Web Services Configuration

When a new Web Site project is created with Visual Studio 2008, the web.config file has a number of new additions that may be unfamiliar to users of previous versions of Visual Studio. Some of these modifications map HttpHandlers and HttpModules. Below are shown modifications made to the <httpHandlers> element in web.config that affects Web Service calls. The default HttpHandler used to process .asmx calls is removed and replaced with a ScriptHandlerFactory class located in the System.Web.Extensions.dll assembly. System.Web.Extensions.dll contains all of the core functionality used by ASP.NET AJAX.

#### ASP.NET AJAX Web Service Handler Configuration

```
<httpHandlers>
<remove verb="*" path="*.asmx"/> <add verb="*" path="*.asmx" validate="false"
type="System.Web.Script.Services.ScriptHandlerFactory,
System.Web.Extensions, Version=1.0.61025.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35"/>
</httpHandlers>
```

This HttpHandler replacement is made in order to allow JavaScript Object Notation (JSON) calls to be made from ASP.NET AJAX pages to .NET Web Services using a JavaScript Web Service proxy.

### Using the ScriptService Attribute

While adding the WebMethod attribute allows the GetCustomersByCountry() method to be called by clients that send standard SOAP messages to the Web Service, it doesn't allow JSON calls to be made from ASP.NET AJAX applications out of the box. To allow JSON calls to be made you have to apply the ASP.NET AJAX Extension's ScriptService attribute to the Web



Service class. This enables a Web Service to send response messages formatted using JSON and allows client-side script to call a service by sending JSON messages.

#### Using the ScriptService Attribute to AJAX-enable a Web Service

```
[System.Web.Script.Services.ScriptService]
[WebService(Namespace = "http://xmlforasp.net")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
public class CustomersService : System.Web.Services.WebService
{
    [WebMethod]
    public Customer[] GetCustomersByCountry(string country)
    {
        return Biz.BAL.GetCustomersByCountry(country);
    }
}
```

The ScriptService attribute acts as a marker that indicates it can be called from AJAX script code.

#### Using the ScriptMethod Attribute

The ScriptService attribute is the only ASP.NET AJAX attribute that has to be defined in a .NET Web Service in order for it to be used by ASP.NET AJAX pages. However, another attribute named ScriptMethod can also be applied directly to Web Methods in a service. ScriptMethod defines three properties including UseHttpGet, ResponseFormat and XmlSerializeString. Changing the values of these properties can be useful in cases where the type of request accepted by a Web Method needs to be changed to GET, when a Web Method needs to return raw XML data in the form of an XmlDocument or XmlElement object or when data returned from a service should always be serialized as XML instead of JSON.

Using the ScriptMethod attribute with the UseHttpGet property.

```
[WebMethod]
[ScriptMethod(UseHttpGet = true)]
public string HttpGetEcho(string input)
{
    return input;
}
```



**Objectives :** This module will familiarize the participant with

How we can handle errors through WebServices.



**Presentation :**

Asynchronous callbacks to Web Services can encounter different types of errors such as the network being down, the Web Service being unavailable or an exception being returned. Fortunately, JavaScript proxy objects generated by the ScriptManager allow multiple callbacks to be defined to handle errors and failures in addition to the success callback shown earlier. An error callback function can be defined immediately after the standard callback function in the call to the Web Method.



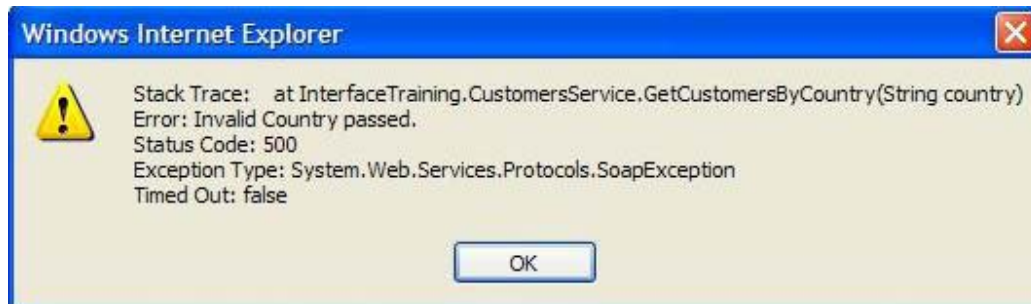
**Demonstration/Code Snippet :**

Defining an error callback function and displaying errors.

```
function GetCustomersByCountry()
{
    var country = $get("txtCountry").value;
    InterfaceTraining.CustomersService.GetCustomersByCountry(country,
    OnWSRequestComplete, OnWSRequestFailed);
}
```

```
function OnWSRequestFailed(error)
{
    alert("Stack Trace: " + error.get_stackTrace() + "/r/n" + "Error: " +
    error.get_message() + "/r/n" + "Status Code: " + error.get_statusCode() + "/r/n" +
    "Exception Type: " + error.get_exceptionType() + "/r/n" + "Timed Out: " +
    error.get_timedOut());
}
```

Any errors that occur when the Web Service is called will trigger the OnWSRequestFailed() callback function to be called which accepts an object representing the error as a parameter. The error object exposes several different functions to determine the cause of the error as well as whether or not the call timed out.



Output generated by calling ASP.NET AJAX error functions.

**Topic: Page Methods****Estimated Time:20 Minutes****Objectives :** This module will familiarize the participant with

How to expose re-usable services to ASP.Net AJAX pages.

**Demonstration/Code Snippet :**

Web Services provide an excellent way to expose re-useable services to a variety of clients including ASP.NET AJAX pages. However, there may be cases where a page needs to retrieve data that won't ever be used or shared by other pages. In this case, making an .asmx file to allow the page to access the data may seem like overkill since the service is only used by a single page.

ASP.NET AJAX provides another mechanism for making Web Service-like calls without creating standalone .asmx files. This is done by using a technique referred to as "page methods". Page methods are static (shared in VB.NET) methods embedded directly in a page or code-behind file that have the WebMethod attribute applied to them. By applying the WebMethod attribute they can be called using a special JavaScript object named PageMethods that gets dynamically created at runtime. The PageMethods object acts as a proxy that shields you from the JSON serialization/deserialization process. Note that in order to use the PageMethods object you must set the ScriptManager's EnablePageMethods property to true.

```
<asp:ScriptManager ID="ScriptManager1" runat="server" EnablePageMethods="true">
</asp:ScriptManager>
```

Below is an example of defining two page methods in an ASP.NET code-behind class. These methods retrieve data from a business layer class located in the App\_Code folder of the Website.

**Defining page methods:**

```
[WebMethod]
public static Customer[] GetCustomersByCountry(string country)
{
    return Biz.BAL.GetCustomersByCountry(country);
}

[WebMethod]
```

```
public static Customer[] GetCustomersByID(string id)
{
    return Biz.BAL.GetCustomersByID(id);
}
```

When ScriptManager detects the presence of Web Methods in the page it generates a dynamic reference to the PageMethods object mentioned earlier. Calling a Web Method is accomplished by referencing the PageMethods class followed by the name of the method and any necessary parameter data that should be passed.

Calling page methods with the PageMethods JavaScript object

```
function GetCustomerByCountry()
{
    var country = $get("txtCountry").value;
    PageMethods.GetCustomersByCountry(country, OnWSRequestComplete);
}

function GetCustomerByID()
{
    var custID = $get("txtCustomerID").value;
    PageMethods.GetCustomersByID(custID, OnWSRequestComplete);
}

function OnWSRequestComplete(results)
{
    var searchResults = $get("searchResults");
    searchResults.control.set_data(results);
    if (results != null) GetMap(results[0].Country,results);
}
```

Using the PageMethods object is very similar to using a JavaScript proxy object. You first specify all of the parameter data that should be passed to the page method and then define the callback function that should be called when the asynchronous call returns.

**Topic: Maintaining Session State****Estimated Time:20 Minutes****Objectives :** This module will familiarize the participant with

How to maintain session state

**Demonstration/Code Snippet :**

In .Net Web Services, session state can be maintained. And if Ajax is used, the session state is still available from ASP.Net Ajax application. ASP.Net Ajax would enable different Ajax applications on the same server share information from the same user.

The EnableSession property of the [WebMethod] helps in maintaining the Session State in ASP.Net Ajax-Enabled Web Services. ASP.Net Session objects can be directly accessed and data can be written to and read from it. Since the Web methods need to be static, HttpContext.Current.Session needs to be used, not only Session, which is available only to actual instances of the Page class.

The Code snippet given below illustrates two functions: one stores the current time in a Session, the other determines the difference between the current time and the time stamp in the Session. If there is no time stamp in the session, -1 is returned.

```
[WebMethod(EnableSession=true)]
[System.Web.Script.Services.ScriptMethod]
public static bool savetime()
{
    HttpContext.Current.Session["PageLoaded"] = DateTime.Now;
    return true;
}
[WebMethod(EnableSession = true)]
[System.Web.Script.Services.ScriptMethod]
public static double calculatedifference()
{
    if (HttpContext.Current.Session["PageLoaded"] == null)
        return -1;
    else
    {
        DateTime then=(DateTime)HttpContext.Current.Session["PageLoaded"];
        TimeSpan diff = DateTime.Now.Subtract(then);
        return diff.TotalSeconds;
    }
}
```

---

**Topic: Exchanging Complex Data with the Server    Estimated Time:20 Mins**

---



**Objectives :** This module will familiarize the participant with

How to exchange complex data with the server.



**Demonstration/Code Snippet :**

Not only primitive types (strings, Booleans, numbers) but complex data can also be exchanged between client and server. JSON supports for exchanging arrays and objects between client and server. ASP.Net Ajax comes with JSON serialization and deserialization features. To understand this, a new Web Method will be created that will return two pieces of information in one, the result of the division and the timestamp from the server. For this to work, a new class MathService.asmx will be created, that will provide an object that later has to be returned.

```
public class DivisionData
{
    public float result;
    public string calculationtime;
}
```

The method illustrated below then creates and returns the object:

```
[WebMethod]
public DivisionData ExtendedDivideNumbers(int a, int b)
{
    float res;
    if (b == 0)
        throw new DivideByZeroException();
    else
        res=(float)a/b;
    string stamp=DateTime.Now.ToLongTimeString();
    DivisionData d=new DivisionData();
    d.result=res;
    d.calculationTime=stamp;
    return d;
}
```

In order to make the returned object accessible for javascript, ASP.Net Ajax must serialize the data into proper JSON. The `GenerateScriptType` attribute (defined in `Script.Web.Script.Services` [where the `ScriptService` and `ScriptMethod` attributes are also located]) instructs the ASP.Net Ajax to pick the correct object definition.

Below is an example with updated `MathService.asmx` code:

```
using System;
using System.Linq;
using System.Web;
using System.Web.Services;
using System.Web.Services.Protocols;
using System.Xml.Linq;
public class DivisionData
{
    public float result;
    public string calculationTime;
}

[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
[System.Web.Script.Services.ScriptService]
[System.Web.Script.Services.GenerateScriptType(typeof(DivisionData))]
public class Service : System.Web.Services.WebService
{
    [WebMethod]
    public float Dividenumbers(int a, int b)
    {
        if (b == 0)
            throw new DivideByZeroException();
        else
            return (float)a/b;
    }
    [WebMethod]
    public DivisionData ExtendedDivideNumbers(int a, int b)
    {
        float res;
        if (b == 0)
            throw new DivideByZeroException();
        else
            res=(float)a/b;
    }
}
```



```
string stamp=DateTime.Now.ToLongTimeString();
DivisionData d=new DivisionData();
d.result=res;
d.calculationTime=stamp;
return d;

}
}
```

On the client, the desialization of the DivisionData object is entirely done automatically. The result from the web services call has the same properties, result and calculationTime as the DivisionData object.

Example below shows the required JavaScript code to call the extended web service:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Complex.aspx.cs"
Inherits="Complex" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>ASP.Net Ajax</title>
  <script language="javascript" type="text/javascript">
function callservice(f)
{
  document.getElementById("c").innerHTML="";

  webService.useService("MathService.asmx?WSDL","MathService");
  webService.MathService.callservice(
  callComplete,
  "Dividenumbers",
  f.elements["a"].value,f.elements["b"].value);

}
function callComplete(result)
{
  document.getElementById("c").innerHTML=result.value;

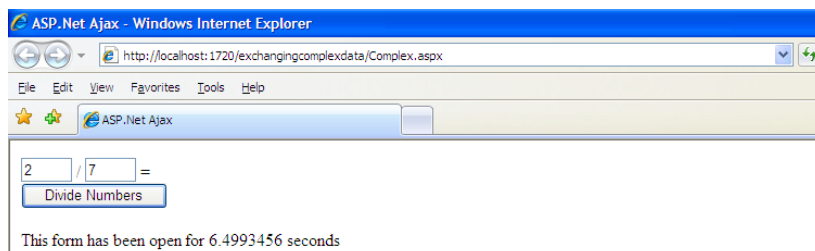
}

</script>
</head>
<body>
```

```
<div id="webService" style="behaviour:url(webservice.htc);">
</div>
<form method="post" action="Service.aspx" onsubmit="return false;">
<div>

<input type="text" id="a" size="2" />
/
<input type="text" id="b" size="2" />
=
<span id="c" style="width: 50px";></span>

<br />
<input type="button" value="Divide Numbers" onclick="callservice(this.form);" />
</div>
</form>
</body>
</html>
```



**Topic: Consuming Web Services with JavaScript Estimated Time: 20 Mins.****Objectives :** This module will familiarize the participant with

How to consume Web Services with Java Script.

**Scenario :****A Zip Code Lookup**

The scenario that we are going to work on is the need for a product website to inform the user of the nearest dealer to them based on the zip code that they enter. While it is true that you can just post back to the server, recreate the entire page, add the desired content to the page and maybe scroll their browser down to see the new results, that is doing way more than it needs to. On top of it all, our 'Zip Code Lookup' functionality will need to be publicly exposed to some other affiliate websites, so the functionality needs to be publicly accessible.

The solution is very simple. We are going to build a Web Service that does the zip code lookup for us. This will allow us to use it in our own web application, and it will also expose the functionality to the public. Let see how easy it is to consume that web service with ASP.NET AJAX.

**Demonstration/Code Snippet :**

First let's take a quick look at the Web Service code so that we can get an understanding of what we will be working with. This code is very simple, and I intentionally am just returning a hard coded result for the purpose of this demo. Here is the code:

```
using System;
using System.Web.Services;
using System.Web.Script.Services;

[ScriptService]
public class VendorLocator : WebService
{
    [WebMethod, ScriptMethod]
    public SearchResult GetNearestDealer()
    {
        // We are intentionally returning a hard coded result for this example.
        return new SearchResult(10, "Bob's Widget Store", "(555) 567-8901");
    }
}
```

```
// This is our "SearchResult" class which will automatically be JSON serialized  
// by ASP.NET AJAX and passed into JavaScript as a parameter.
```

```
public class SearchResult  
{  
    public SearchResult(int distance, string name, string phoneNumber)  
    {  
        this.distance = distance;  
  
        this.name = name;  
  
        this.phoneNumber = phoneNumber;  
    }  
  
    private int distance;  
    public int Distance  
    {  
        get { return this.distance; }  
        set { this.distance = value; }  
    }  
  
    private string name;  
    public string Name  
    {  
        get { return this.name; }  
        set { this.name = value; }  
    }  
  
    private string phoneNumber;  
    public string PhoneNumber  
    {  
        get { return this.phoneNumber; }  
        set { this.phoneNumber = value; }  
    }  
}
```

There are a few things to notice here. First of all, you'll notice that we had to add a few attributes to our class and method so that ASP.NET will know that we plan on exposing these abilities to ASP.NET AJAX. This does not hurt the application in any way and will not change the way anyone else uses the web service, but rather it adds functionality. The two attributes that we needed was the "ScriptService" attribute on the class and the "ScriptMethod" attribute on the method. Very simple so far.

Another thing to note is that we made our own custom class to hold our search data. ASP.NET

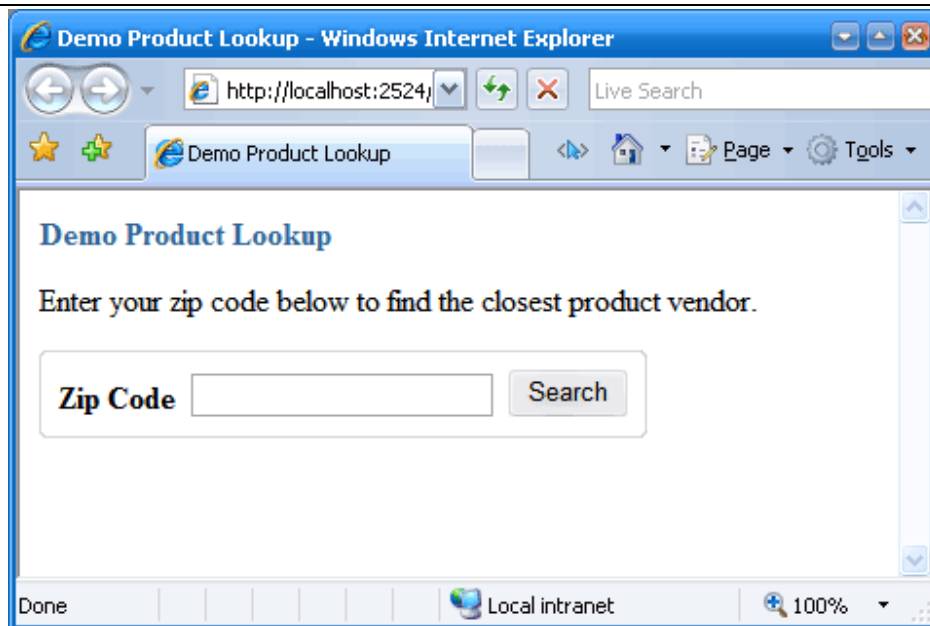
AJAX will automatically JSON (JavaScript Object Notation) serialize all of the public properties (in this case "Distance", "Name" and "PhoneNumber") and pass them as a parameter back to JavaScript.

### The ASP.NET Code

Now let's look at our ASP.NET code for our web application that is going to consume this Web Service:

```
<form runat="server">
  <asp:ScriptManager ID="ScriptManager" runat="server" />
  <h1>
    Demo Product Lookup</h1>
  <p>
    Enter your zip code below to find the closest product vendor.</p>
  <fieldset>
    <label>
      Zip Code<input type="text" ID="ZipCode" /></label>
    <button onclick="findVendor();">
      Search
    </button>
  </fieldset>
  <dl id="dealerResults" style="display: none;">
    <dt>Distance:</dt>
    <dd id="distance" />
    <dt>Name:</dt>
    <dd id="name" />
    <dt>Phone:</dt>
    <dd id="phoneNumber" />
  </dl>
</form>
```

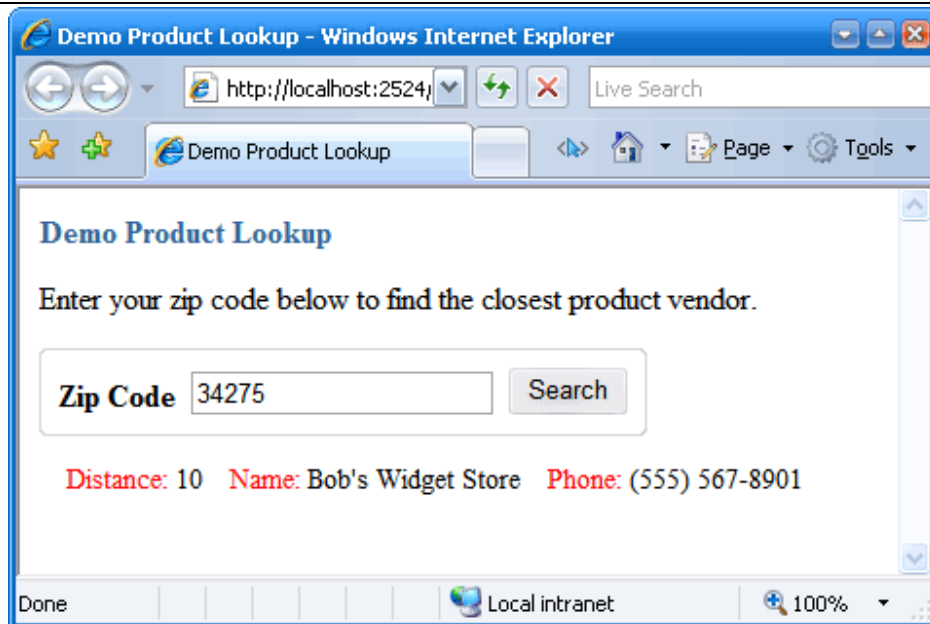
With a little bit of styling, the above code looks like this:



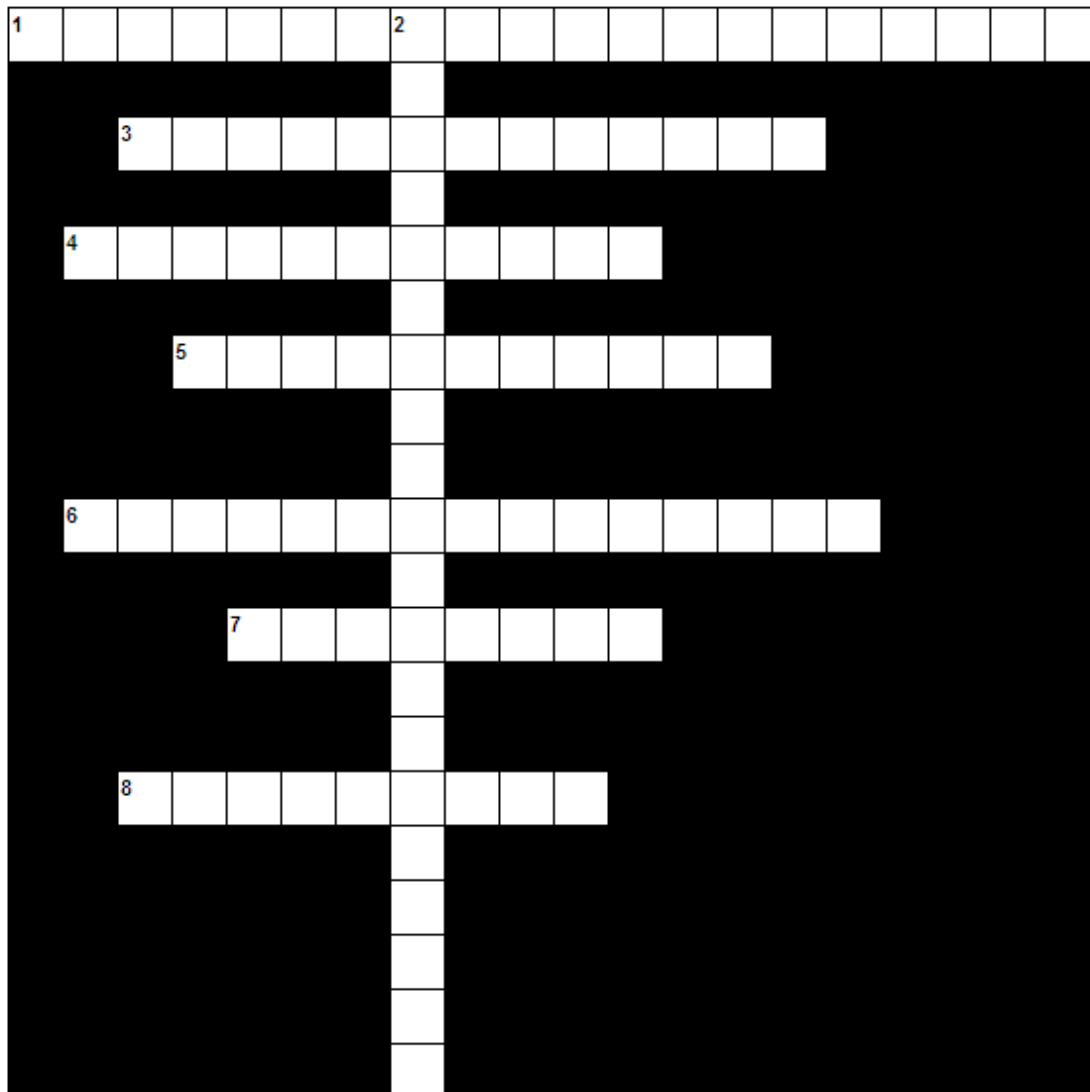
The JavaScript Code :Our button's "onclick" property is set to call the function 'findVendor'. That JavaScript code will look like this:

```
function findVendor() {  
    var zipCode = $get("ZipCode").value;  
  
    VendorLocator.GetNearestDealer(function(result) {  
        // Assign the Distance  
        $get("distance").innerHTML = result.Distance;  
  
        // Assign the Name  
        $get("name").innerHTML = result.Name;  
  
        // Assign the Phone Number  
        $get("phoneNumber").innerHTML = result.PhoneNumber;  
  
        // Make the dealer results visible  
        $get("dealerResults").style.display = "";  
    });  
}
```

When we enter in our zip code and press the search button we get the following results:



Notes: ASP.NET AJAX does all of the hard work for us, and beautifully returns our "SearchResult" object which we built in C#. What they have done is truly amazing. The "result" object that comes back from our function call will have all properties populated and accessible in our JavaScript code. The communication to the server, the creation of the client-side function, the JSON serialization - it's all handled by ASP.NET AJAX. So get excited about the web again, start to think of web applications as being 'connected', and for goodness' sake, mash it up with ASP.NET AJAX.

**Crossword: Chapter 3 & 4**
**Estimated Time:10 Mins**

**ACROSS:**

1. In Visual Studio 2008, the default `HttpHandler` used to process `asmx` calls is replaced with this class. [`ScriptHandlerFactory`].

3. To allow JSON calls, you have to apply this attribute to the Web Service class. [`ScriptService`]

4. By default, the interval in timer control is set to 60000 \_\_\_\_\_ [Milliseconds].

5. The technique by which AJAX provides WebService like calls without creating



---

standalone .asmx files.[PageMethods].





6. Any error that occur when the WebService is called will trigger this callback function.[OnRequestFailed].
7. This property in the timer control sets the time gap after which the page is going to be refreshed.[Interval].
8. The methods in the webservises are added under this attribute.[WebMethod].

**DOWN:**

2. This control, available in AJAX toolkit, enable us to call webservice.  
[AutoCompleteExrender]

## 5.0 AJAX Authentication Service

### Topics

-  **5.1 Introduction**
-  **5.2 Preparing the Application**
-  **5.3 Login and Logout**
-  **5.4 Crossword**





**Objectives :** This module will familiarize the participant with

Basic idea of AJAX Authentication Service.

### **Ajax Authentication Service**

Authentication is one of the important modules in any website, unless and until there is nothing private on the site or something. To provide Forms Authentication in ASP.Net AJAX, there is a service called Authentication Service is packed with the ASP.Net AJAX framework. It is a web service which helps in authenticating users against the authentication database in ASP.Net AJAX applications. This web service is called through a JavaScript proxy, which gets exposed to the client by the ASP.Net AJAX framework when forms authentication is enabled.

This proxy scripts are packed inside Sys.Services namespace which can be used to access the web service from client side to verify the user and to set the authentication cookie.

By default, when the form authentication is enabled, it authenticates the user against the server with a default web service packed with ASP.Net AJAX. This default web service will authenticate the user using default provider and the default database aspnetdb. This means, by default, it is not required to write any code for authentication. Obviously, for a successful authentication the user should be added as a member in provide database.

The authentication service uses the existing provider model. Hence, it is easy to migrate an existing asp.net application that uses login controls and providers. By default, this authentication service will also create an ASPXAUTH cookie on successful authentication, like the existing asp.net authentication infrastructure. So, whatever is possible with the existing asp.net authentication infrastructure is also possible with ASP.Net AJAX infrastructure with the support of client side proxy classes in Sys.Service namespace. For example, denying access to folders using location tag in web.config, etc.

Design Consideration for Forms Authentications in AJAX application:

The main aim of having AJAX application is to prevent the page refresh for server communication which provides better user experience when compared to classic asp.net applications. In classic asp.net, we will normally have a separate login.aspx page to authenticate users. Means, whenever the user gives a request to access the private resource he/she will be redirected to login.aspx or the login page we provide in <Forms> tag in web.config.

**Topic: Preparing the Application****Estimated Time:20 Minutes****Objectives :** This module will familiarize the participant with

How to prepare the application for AJAX authentication service.

**Presentation :**

Let us follow these steps for preparing the application:

1. Steps

- i) Open Visual studio 2005. Click File> New Website.
- ii) Select ASP.Net AJAX-Enabled Web Site. Type your website name.

2. Enabling Authentication Services

The below setting in Web.Config will enable authentication service for the AJAX application.

```
<system.web.extensions>
  <scripting>
    <webServices>
      <authenticationService enabled="true" />
    </webServices>
  </scripting>
</system.web.extensions>
```

When we include the above setting in Web.Config file, the client side proxy class for calling webservice will be automatically exposed in the client side by Asp.net Ajax.

Like classic asp.net application, we need to set the authentication mode to “Forms” for the forms authentication to work.

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



**Objectives :** This module will familiarize the participant with

How to implement authentication through Login and Logout.



**Presentation :**

**Constructing the Login:**

After you have prepared the application now you need to do the following

Drag 2 textbox for Username and Password with a Button control for login click.

```
<table>
  <tr>
    <td style="width: 100px">
      UserName
    </td>
    <td style="width: 100px">
      <asp:TextBox ID="txtUserName"
runat="server"></asp:TextBox></td>
    </tr>
    <tr>
    <td style="width: 100px">
      Password
    </td>
    <td style="width: 100px">
      <asp:TextBox ID="txtPass" runat="server"></asp:TextBox></td>
    </tr>
    <tr>
    <td style="width: 100px">
    </td>
    <td style="width: 100px">
      <asp:Button ID="btnLogin" runat="server" OnClientClick="return
OnLogin();" Text="Login" /></td>
    </tr>
</table>
```

From JavaScript, it is required to call the authentication web service for authentication. It is done by calling the webservice through the proxy method called login () in

---

AuthenticationService class which is packed in Sys.Service namespace using JavaScript.

The syntax of the login method is,

```
Sys.Services.AuthenticationService.login(userName, password, isPersistent, customInfo, redirectUrl, loginCompletedCallback, failedCallback, userContext);
```

*isPersistent* – Specifies the cookie is persistent or non persistent.

*customInfo* - Reserved for future use. The default is null.

*redirectUrl* – Redirect URL for successful authentication. Null specifies no redirection.

*loginCompletedCallback* – This callback JavaScript function will be called once the authentication service returns to client. It is called for both authentication failure and success.

*failedCallback* – This callback function will be called if there is any exception happened on the server while authenticating. Authentication failure will not call this callback method.

*userContext* – User context info we are passing for callback function.

When the method is called with argument values, it calls the webservice login method which will verify the credentials in the server against the database and set the auth cookie. This webservice will in turn use the membership provider and the user is checked against the default database in aspnetdb in AppData folder.

```
function OnLogin()
{
    var username = $get("txtUserName");
    var password = $get("txtPass");
    Sys.Services.AuthenticationService.login(username.value,
    password.value, false,null,null,onLoginCompleted,onLoginFailed,"User
Context");
    return false;
}

function onLoginFailed()
{
    alert("Login failed due to some exception");
}
```

```
}  
  
function onLoginCompleted()  
{  
    loggedIn = Sys.Services.AuthenticationService.get_isLoggedIn();  
    if(loggedIn)  
    {  
        $get("CommentsDIV").style.display = "block";  
        $get("UserDetails").style.display = "block";  
        $get("UserName").innerText = $get("txtUserName").value;  
        $get("LoginDIV").style.display = "none";  
    }  
    else  
    {  
        alert("Login failed due to invalid credentials!!");  
    }  
}
```

As it is seen from the above code, we can use the login completed callback function to enable/disable protected resources on the page. Since, there is an enabling/disabling control in JavaScript it is advisable to check whether the user is authenticated before performing any protected server side operation which requires authentication.

### **Constructing Logout:**

The syntax of this method is,

```
Sys.Services.AuthenticationService.logout(redirectUrl, logoutCompletedCallback,  
failedCallback, userContext);
```

logoutCompletedCallback & failedCallback are callback methods for respective operations, as the name suggests.

There can be a hyperlink to call the logout method similar to below.

```
function SignOut()  
{  
    Sys.Services.AuthenticationService.logout(null,onLogoutCompleted,null,null);  
    return false;  
}
```

On Logout completed callback, The elements can be disabled that requires authentication.

Before authenticating, It needs to register the user or create users in aspnetdb database. Use CreateUserWizard from login controls to create new users easily. The aspnetdb database in App\_Data folder will be automatically created for the first request given to the provider for creating new user. Once a new user is created, the above authentication service can be used for successful authentication.

#### Callback function

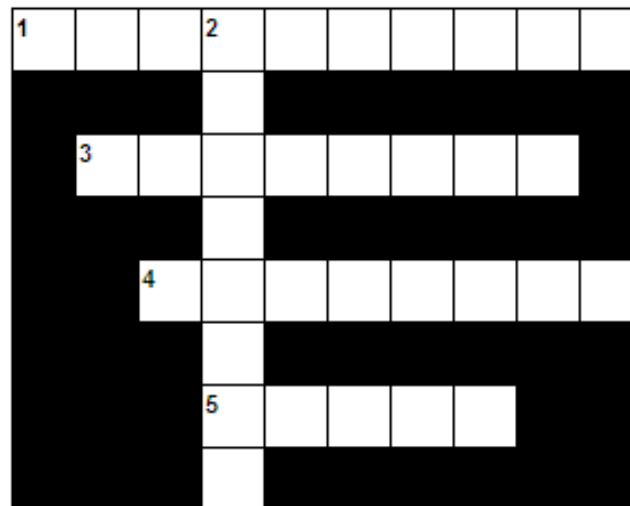
The callback function that is used in login and logout method can have 3 arguments.

1. *Result* which gives the client side exception object that substitutes the server exception. To get exception message, etc.
2. *userContext* Usercontext info we passed during server method call.
3. *Method* Method that made the server calls.

Hence the above callback method can be written as,

```
function onLoginFailed(Result, userContext, Method)
{
    alert("Login failed due to exception: "+Result.get_message()+" on method "+
Method);
}
```



**Crossword: Chapter 5****Estimated Time:10 Mins****ACROSS:**






- 1.This wizard from login controls create new users easily.[CreateUser].
- 3.Default ASP.Net database.[ASPNETDB].
- 4.This types of functions enable/disable protected resources on the page.[Callback].
- 5.The callback function used in login and logout method can have \_\_\_\_\_ arguments.[Three]

**DOWN:**

- 2.This cookie is created on successful authentication[AXPXAUTH]

## 6.0 AJAX Profile Service

### Topics

-  **6.1 Introduction**
-  **6.2 Preparing Web Site**
-  **6.3 Accessing Profile Data**
-  **6.4 Accessing Profile Group Data**
-  **6.5 Crossword**



---

**Topic: Introduction to AJAX Profile Service      Estimated Time:20 Minutes**

---

ASP.NET 2.0 AJAX Extensions profile service enables you to use the ASP.NET profile application service in applications. You can access profile data by using ECMAScript (JavaScript) code that accesses a client representation of the profile.

User profiles are storage mechanisms that can store data for both known (authenticated) and unknown (anonymous) users on the server. User profiles are neither a new idea nor something particularly special. HTTP cookies, for example, provide a means to maintain data on the client. For performance reasons though, most profile schemes do not store the full complement of user information on the client, but only use a unique identifier (ID). This ID is then sent back and forth between client and server.

ASP.Net AJAX comes with a JavaScript API that allows client-side code to access server-side profile data. As a result, JavaScript can read and write profile data, enabling it to create dynamic, profile-driven web pages while using a few page refresh cycles as possible. In the background, the JavaScript code uses the web services support of ASP.Net AJAX to call a server component, which then accesses the profile data.

**Topic: Preparing Web Site****Estimated Time:20 Minutes****Objectives :** This module will familiarize the participant with

Basic idea about Profile service

**Presentation :**

**Enabling the Profile Service :**To use the profile service from script, you must enable the profile service by using the following element in the application's Web.config file.

```
<system.web.extensions>
  <scripting>
    <webServices
      < profileService enabled="true" />
    </webServices>
  </scripting>
</system.web.extensions>
```

By default, no profile properties are available. For each profile property that you want to make available in a client application, add the property name to the **readAccessProperties** attribute of the **<profileService>** element. Similarly, for each server profile property that can be set based on data sent from a client application, add the property name to the **writeAccessProperties** attribute.

The following example shows how to expose individual properties.

```
<profileService enabled="true"
  readAccessProperties="BackgroundColor,ForegroundColor"
  writeAccessProperties=" Backgroundcolor,ForegroundColor"/>
```

To define the profile properties in the **<profile>** section by using syntax like that in the following example. For grouped properties, use the **<group>** element.

```
<profile enabled="true">
  <add name=" Backgroundcolor" type="System.String"
    defaultValue="white" />
  <add name=" Foregroundcolor" type="System.String"
    defaultValue="black" />
  <properties>
    <group name="Address">
      <add name="Street" type="System.String" />
      <add name="City" type="System.String"/>
      <add name="PostalCode" type="System.String" />
    </group>
  </properties>
</profile>
```

**Example:** The following example shows how to use the profile service from client script. The example consists of an ASP.NET Web page that contains an ScriptManager control. When this control is included on the page, the AuthenticationService object is automatically available to any client script in the page.

The page contains a login button with an associated event handler named OnClickLogin. Code in the method calls the login method of the AuthenticationService object.

After you are logged in, the loginComplete callback function is called, which calls the load method of the ProfileService object to load the profile for the current user.

The profile information consists of background and foreground colors that you can set after you are logged in. The background and foreground colors are profile properties you must set in the Web.config file. To define these properties, add the following elements to the application's Web.config file:

```
<profile enabled="true">
```

```
<properties>
  <add name="Backgroundcolor" type="System.String"
    defaultValue="white"/>
  <add name="Foregroundcolor" type="System.String"
    defaultValue="black"/>
</properties>
</profile>
```

### Preparing The Web Site:

To take advantage of ASP.Net AJAX profile support, it must first be enabled. To enable the profile service, the Web.config file must have additional elements defined. Within the <system.web> node, the <profile> element is used to define the properties in an application.

By default, profile support in ASP.net works only for authenticated users. However, by using the <anonymousIdentification> element, a unique token can be generated for unauthenticated users to identify their server-side profile data:

```
<configuration>
[... ]
<system.web>
  <anonymousIdentification enabled="true"/>
```

For actual profile data, properties can be defined in two different ways, individually or as groups. The below code snippet will show both an individual username property and a grouped UserData property. The UserData property consists of myUserName and myPassword.

```
<system.web>
  <profile>
    <properties>
      <add name="UserName" allowAnonymous="true"/>
      <group name="UserData">
        <add name="myUserName" allowAnonymous="true"/>
        <add name="myPassword" allowAnonymous="true"/>
      </group>
    </properties>
  </profile>
</system.web>
```

Finally, to enable the component that grants JavaScript access to the profile information. Add the <system.web.extensions> element to the end of the Web.config file, just above the closing </configuration> tag, and configure it as shown below:

```
<system.web.extensions>
  <scripting>
    <webServices>
      <profileService enabled="true"

        readAccessProperties="userName,UserData.myUserName,UserData.myPassword"

        writeAccessProperties="userName,UserData.myUserName,UserData.myPassword"/>
    </webServices>
  </scripting>
</system.web.extensions>
</configuration>
```

In the <profileService> property, a list must be provided defining all properties from which the application can read and to which it can write. The properties that have to be exposed to the client script can be a subset of all the profile properties defined in the application. In addition, for client-script access, ASP.Net distinguishes between read and write access.

**Topic: Accessing Profile Data****Estimated Time:20 Minutes****Objectives :** This module will familiarize the participant with

Basic idea about Profile service

**Presentation :**

Profile-Related ASP.Net AJAX functionality is defined in the client Sys.Services.ProfileService class. The first step for every application using profile support is to load profile information by using the Sys.Services.ProfileService.load() method, which expects four parameters:

**propertyNames**

The properties to be loaded.If set to null or to an empty string, all exposed profile properties are retrieved from the server.

**loadCompletedCallback**

The method to call when the profile loading succeeds.

**failedCallback**

The method to call if profile loading fails.

**Usercontext**

Optional information that is submitted as an argument to the callback functions.

After loading the profile information, accessing properties is easy. Using Sys.Services.ProfileService.properties.<property name>, there is read and write access to the property-as long as the web.config configuration allows it.Though, setting a profile property does not actually save the information on the server; it only makes it available to the current



script. In order to persist this information, the save() method must be called. It expects four arguments, just as load() does:

### **propertyNames**

The properties to be saved. If set to null or to an empty string, all properties available to client script are sent to the server.

### **saveCompletedCallback**

The method to call when the profile saving succeeds.

### **failedCallback**

The method to call if profile saving fails.

### **Usercontext**

Optional information that is submitted as an argument to the callback functions.

The below example gives a full view of how profile data is to be written and read:

```
<%@ Page Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Ajax Profile Service</title>
    <script type="text/javascript">
        function pageload()
        {
            $addHandler($get("txtUsername"), "change", saveProfile);
Sys.Services.ProfileService.load(null, profileLoaded, profileError, "load");
        }
        function profileLoaded()
        {
            $get("statusText").firstChild.nodeValue="Profile data Loaded";
            if(Sys.Services.ProfileService.properties.username!=null)
            {
                $get("txtUsername").value=Sys.Services.ProfileService.properties.userName;
            }
        }
    </script>
</head>
</html>
```

```
}
function profileError(result,context)
{
    $get().firstChild.nodeValue="Could not" + context +"profile
("+result.get_message()+").Check the configuration in web.config!";
}
function saveProfile()
{
Sys.Services.ProfileService.properties.userName=$get("txtUsername").value;
Sys.Services.ProfileService.save(null,profilesaved,profileError,"save");
}
function profileSaved()
{
    $get("statusText").firstChild.nodeValue="Profile data saved";
}
</script>
</head>
<body>
<form id="form1" runat="server">
<div>
Username:<input type="text" id="txtUsername" runat="server" /><br/>

Password:<input type="password" id="txtPassword" runat="server" /><br/>
<input type="button" id="Button1" runat="server" value="Login"
onclick="alert('not implemented');"/><br />
<span id="statusText" runat="server">&nbsp;</span>
</div>
<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>
</form>
</body>
</html>
```

When this example is run, the login form is initially empty. When a user name is entered and tab is pressed, the name entered is saved. From then on, any time the page is loaded the user name entered is preloaded into the user name textbox.

**Topic: Accessing Profile Group Data****Estimated Time: 20 Minutes****Objectives :** This module will familiarize the participant with

Basic idea about Profile service

**Presentation :**

While using the grouped profile data, accessing the information differs only a little from individual profile properties. Use a dot to separate the group name from the property name:

`Sys.Services.ProfileService.properties.<group name>.<property name>`

Now to include two fields like saving the password along with the username in the profile. It requires only a little extra code-specifically two save functions, one for each property.

```
function saveProfile1()
{
    Sys.Services.ProfileService.properties.UserData.myuserName=$get("txtUsername").value;

    Sys.Services.ProfileService.save(null, profilesaved, profileError, { "operation": "save", "property": "username" });
}
function saveProfile2()
{
    Sys.Services.ProfileService.properties.UserData.myPassword=$get("txtPassword").value;

    Sys.Services.ProfileService.save(null, profilesaved, profileError, { "operation": "save", "property": "password" });
}
```

It can be seen, that the context is used once again, but this time an object is used instead of a simple string. The object's operation property contains either "save" or "load", and the property member is used to transmit which profile information has been saved. This serves two purposes:

only one error handling function is required, and also only one handling function is needed to respond to a successful save.

Below is the code for reading and writing profile group data:

```
<%@ Page Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Ajax Profile Service</title>
    <script type="text/javascript">
        function pageload()
        {
            $addHandler($get("txtUsername"), "change", saveProfile1);
            $addHandler($get("txtPassword"), "change", saveProfile2);

Sys.Services.ProfileService.load(null, profileLoaded, profileError, {"operation"
:"load"});
        }
        function profileLoaded()
        {

            if (Sys.Services.ProfileService.properties.UserData != null)
            {
                $get("statusText").firstChild.nodeValue = "Profile data Loaded";

$get("txtUsername").value = Sys.Services.ProfileService.properties.UserData.myU
serName;

$get("txtPassword").value = Sys.Services.ProfileService.properties.UserData.myP
assword;
            }
            else
            {
                $get("statusText").firstChild.nodeValue = "No data available";
            }
        }
        function profileError(result, context)
        {
            $get("statusText").firstChild.nodeValue = "Could not" +
context.operation + "profile (" + result.get_message() + "). Check the
configuration in web.config!";
        }
        function saveProfile1()
        {

Sys.Services.ProfileService.properties.UserData.myuserName = $get("txtUsername"
).value;
```

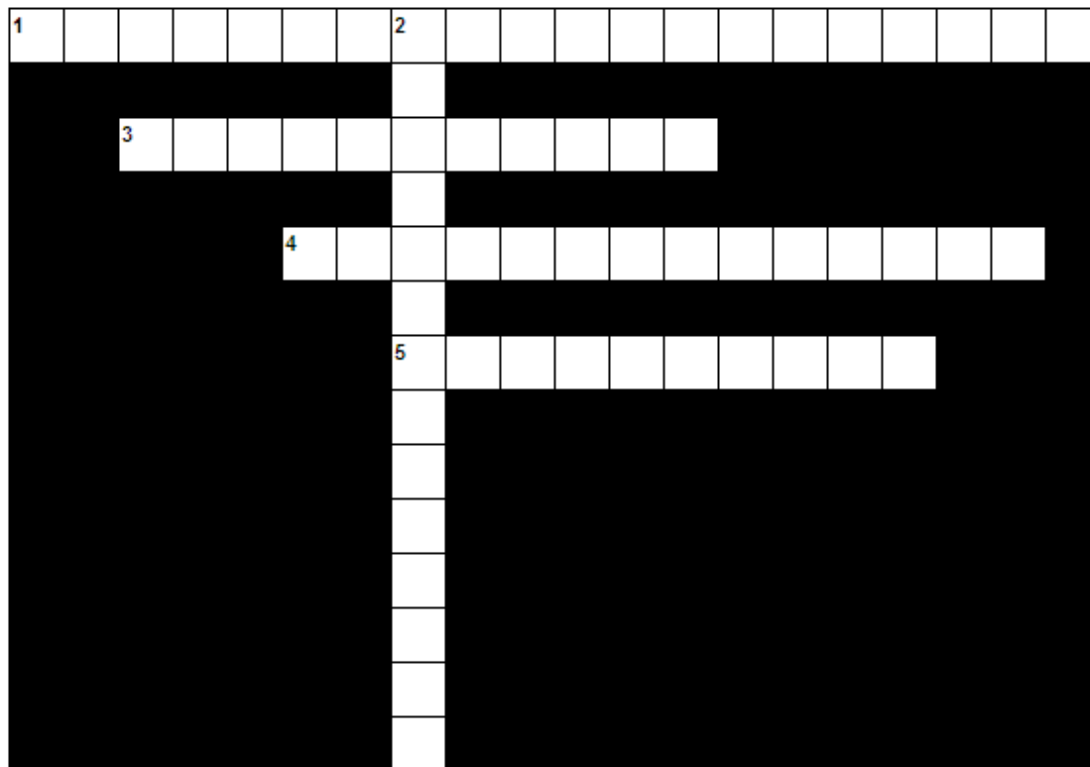
```

Sys.Services.ProfileService.save(null,profilesaved,profileError,{"operation":
"save","property":"username"});
    }
    function saveProfile2()
    {

Sys.Services.ProfileService.properties.UserData.myPassword=$get("txtPassword"
).value;

Sys.Services.ProfileService.save(null,profilesaved,profileError,{"operation":
"save","property":"password"});
    }
    function profileSaved(success,context)
    {
        $get("statusText").firstChild.nodeValue="Profile data
("+context.property+") saved.";
    }
</script>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            Username:<input type="text" id="txtUsername" runat="server" /><br/>
            Password:<input type="password" id="txtPassword" runat="server" /><br/>
            <input type="button" id="Button1" runat="server" value="Login"
onclick="alert('not implemented');"/><br />
            <span id="statusText" runat="server">&nbsp;</span>
        </div>
        <asp:ScriptManager ID="ScriptManager1" runat="server">
        </asp:ScriptManager>
    </form>
</body>
</html>

```

**Crossword: Chapter 6****Estimated Time:10 Mins****ACROSS:**


- 1.The method to be called when the profile saving succeeds.[SAVECOMPLETECALLBACK]
3. Optimal Information that is submitted as an argument to the callback.[USERCONTEXT].
- 4.The method to call if profile saving fails.[failedcallback]
- 5.You can access profile data by using this javascript code.[ECMAScript].


**DOWN:**

- 2.Profile related ASP.Net AJAX functionality is defined in the Sys.Services.\_\_\_\_\_ class.[ProfileService]

## 7.0 Localizing and Globalizing Applications

### Topics

 **7.1 Globalizing the Applications**

 **7.2 Localizing the Applications**



**Topic: 7.1 Globalizing the Applications****Estimated Time: 20 Minutes****Objectives :** This module will familiarize the participant with

How to Globalize Web Application by using AJAX.

**Presentation :**

Globalization and localization are often overlooked features in ASP.NET. ASP.NET AJAX has refreshed the face of these two beauties now by bringing them to the client-side world. This topic consists of what these two topics cover as well as how easy it is to utilize them in JavaScript with ASP.NET AJAX.

**Globalization**

A process of creating a product or service that is successful in many countries without modification. Basically, as a web developer building a multi-national web application, you can utilize ASP.NET (and in this case ASP.NET AJAX) to automatically format dates, currency and other things according to the preferences of your visitors.

**Formatting Dates and Numbers with Globalization**

The following code and screen shots will show the how easy it is to change the formatting of dates and numbers based on the culture of someone visiting your site. We are going to build a page which simply displays the current date and time, and also a number formatted as currency. Then we'll see how they automatically change as we switch to a different culture.

Here is some basic HTML that we are going to use to display the current date and a currency field.

```
<p>
  Currency for the culture selected: <span id="currencySpan" />
</p>
<p>
```

A date for the culture selected: <span id="dateSpan" />

Now we'll look at a simple JavaScript function that will display values into the above spans.

```
window.onload = function() {
  $get("currencySpan").innerHTML = String.localeFormat("{0:c}", 12345.67);
  $get("dateSpan").innerHTML = String.localeFormat("{0:dddd, dd MMMM yyyy hh:mm:ss tt}", new
```



```
Date());  
}
```

“\$get” and “String.localeFormat” are functions that are added into JavaScript just by having the ASP.NET AJAX ScriptManager object on your ASP.NET page. The “\$get” function is similar to the “document.getElementById” function in JavaScript.

The String.localeFormat function converts a value into a string in the format that is expected from the locale (culture or language).

Below are the results of the two conversions above:



Now, notice what happens to the values as we change to a different culture. Keep in mind; There is no change in the code at all.



**Topic: 7.2 Localizing the Applications****Estimated Time: 20 Minutes****Objectives :** This module will familiarize the participant with

How to Localize Web Application by using AJAX.

**Presentation :**

Localization is similar to globalization, but it focuses on translating parts of the site (primarily the UI) based on the user's culture (or language). Unlike globalization, with localization the translation itself is not done by ASP.NET. It needs to create "resource" files for each culture that should be supported. The job of ASP.NET AJAX is to detect the culture of the user coming to your site, and to show him the right text.

Keep in mind that both globalization and localization are not new concepts, and they are not new features to ASP.NET. The part of this that is new is the tie-in to JavaScript brought about by ASP.NET AJAX.

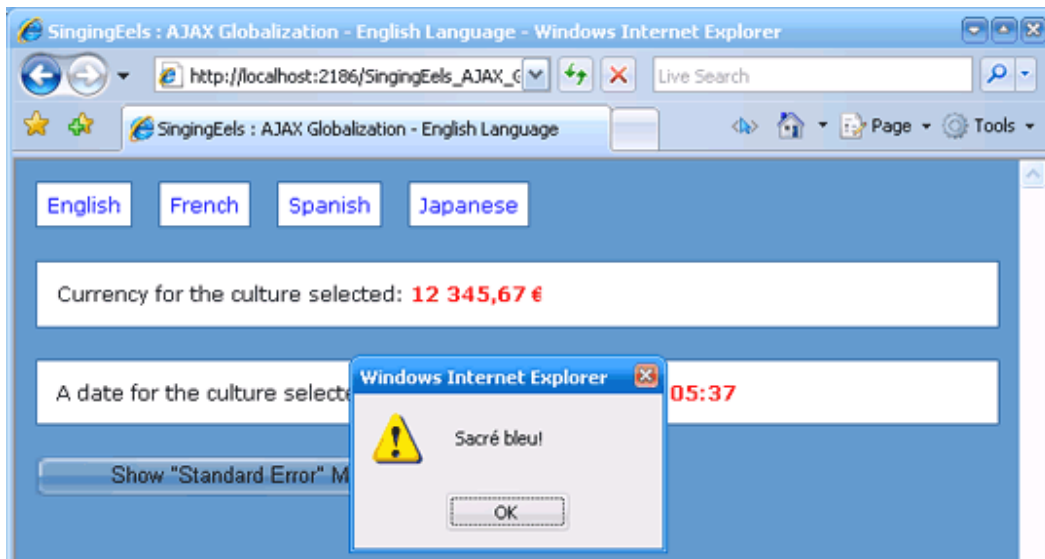
**Demonstration/Code Snippet :****Multi-Lingual UI with Localization**

Localization gives the power to support different languages in the web application. It can be chosen to translate the UI (the text on buttons, labels etc) or it can change other things such as error messages. It is important to note (again) that the localization features will not translate your text for you (as globalization does), but will simply use the translation you provide for the right culture.

Just add the following HTML to the above example:

```
<button onclick="DisplayStandardError();">
  Show "Standard Error" Message</button>
And here is the JavaScript code that contains the function "DisplayStandardError":
function DisplayStandardError()
{
  alert(ErrorCodes.StandardError);
}
```

Notice that it is alerting "ErrorCodes.StandardError", but it is not defined that value anywhere. This is where localization comes in. We are going to tell ASP.NET AJAX that we have some resources in another DLL that will contain the "ErrorCodes" object which has a "StandardError" string value in there. Also, we are going to provide a translation for French, Spanish and Japanese.



Localized Application according to language!

## 8.0 Built-in support for AJAX – Server and Client

### Topics

-  **8.1 AJAX Client Architecture**
-  **8.2 Ajax Server Architecture**
-  **8.3 Crossword[Chap 7 & 8 Combined]**





**Objectives :** This module will familiarize the participant with

The basic idea of AJAX Client architecture.



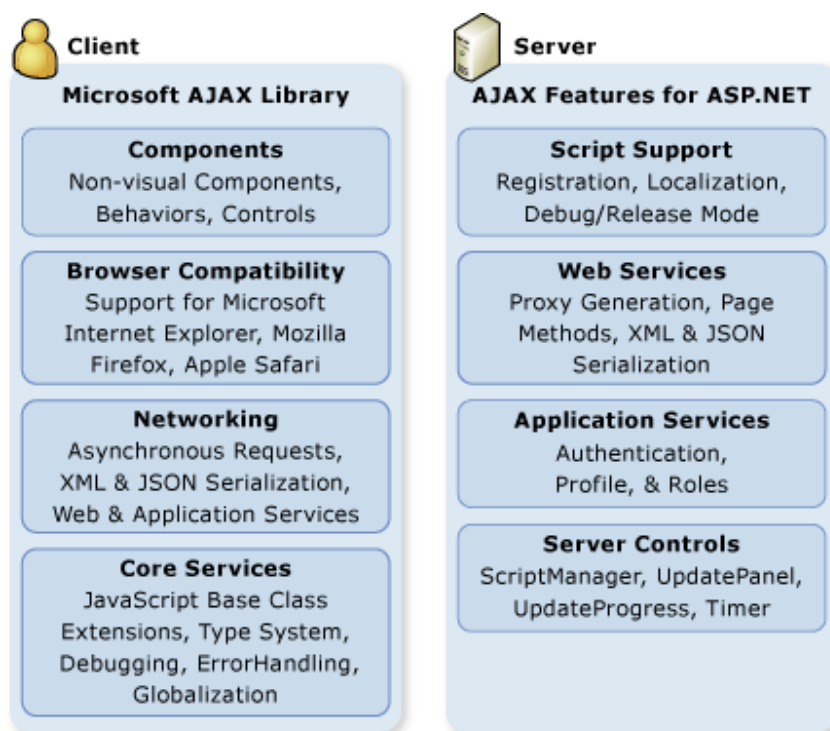
**Presentation :**

The architecture of AJAX features in ASP.NET consists of two pieces: client-script libraries and server components. These pieces are integrated to provide a robust development framework.



**Demonstration/Code Snippet :**

### ASP.NET AJAX client and server architecture



The illustration shows the functionality of the client-based Microsoft AJAX Library, which includes support for creating client components, browser compatibility, and networking

and core services. The illustration also shows the functionality of server-based AJAX features, which include script support, Web services, application services, and server controls. The following sections describe the illustration in more detail.

**Ajax Client Architecture:** The client architecture includes libraries for component support, browser compatibility, networking, and core services.

### 1. Components

Client components enable rich behaviors in the browser without postbacks. Components fall into three categories:

- Components, which are non-visual objects that encapsulate code, such as a timer object.
- Behaviors, which extend the basic behavior of existing DOM elements.
- Controls, which represent a new DOM element that has custom behavior.

The type of component that you use depends on the type of client behavior you want. For example, a watermark for an existing text box can be created by using a behavior that is attached to the text box.

### Creating Custom AJAX Client Controls

This overview shows you how to create a custom ASP.NET AJAX client control and use it in a page. In this overview you will learn how to do the following:

- Use the prototype design pattern in ECMAScript (JavaScript) to define a control class.
- Register a control as a class that derives from the `Sys.UI.Control` base class.
- Initialize the **Control** base class and invoke its methods.
- Create custom events that a page developer can bind to and handle.
- Use the client control in a page and bind to the control's events.

### 2. Browser Compatibility

The browser compatibility layer provides AJAX scripting compatibility for the most frequently used browsers (including Microsoft Internet Explorer, Mozilla Firefox, and Apple Safari). This enables you to write the same script regardless of which supported browser you are targeting.

---

### **3. Networking**

The networking layer handles communication between script in the browser and Web-based services and applications. It also manages asynchronous remote method calls. In many common scenarios, such as partial-page updates that use the UpdatePanel control, the networking layer is used automatically and does not require that you write any code.

The networking layer also provides support for accessing server-based forms authentication, role information, and profile information in client script. This support is also available to Web applications that are not created by using ASP.NET, as long as the application has access to the Microsoft AJAX Library.

### **4. Core Services**

The AJAX client-script libraries in ASP.NET consist of JavaScript (.js) files that provide features for object-oriented development. The object-oriented features included in the ASP.NET AJAX client-script libraries enable a high level of consistency and modularity in client scripting. The following core services are part of the client architecture:

- Object-oriented extensions to JavaScript, such as classes, namespaces, event handling, inheritance, data types, and object serialization.
- A base class library, which includes components such as string builders and extended error handling.
- Support for JavaScript libraries that are either embedded in an assembly or are provided as standalone JavaScript (.js) files. Embedding JavaScript libraries in an assembly can make it easier to deploy applications and can help solve versioning issues.

### **5. Debugging and Error Handling**

The core services include the Sys.Debug class, which provides methods for displaying objects in readable form at the end of a Web page. The class also shows trace messages, enables you to use assertions, and lets you break into the debugger. An extended Error object API provides helpful exception details with support for release and debug modes.

### **6. Globalization**

The AJAX server and client architecture in ASP.NET provides a model for localizing and globalizing client script. This enables you to design applications that use a single code base to provide UI for many locales (languages and cultures). For example, the AJAX architecture





---

enables JavaScript code to format **Date** or **Number** objects automatically according to culture settings of the user's browser, without requiring a postback to the server.

**Topic: AJAX Server Architecture****Estimated Time: 20 Minutes****Objectives :** This module will familiarize the participant with

The basic idea of AJAX Client architecture.

**Presentation :**

The server pieces that support AJAX development consist of ASP.NET Web server controls and components that manage the UI and flow of an application. The server pieces also manage serialization, validation, control extensibility, and so on. There are also ASP.NET Web services that enable you to access ASP.NET application services for forms authentication, roles, and user profiles.

**Demonstration/Code Snippet :**

AJAX server architecture consists of the following:

**1. Script Support**

AJAX features in ASP.NET are implemented by using supporting scripts that are sent from the server to the client. Depending on what AJAX features that you enable, different scripts are sent to the browser.

You can also create custom client script for your ASP.NET applications. In that case, you can also use AJAX features to manage your custom script as static .js files (on disk) or as .js files embedded as resources in an assembly.

ASP.NET AJAX features include a model for release and debug modes. Release mode provides error checking and exception handling that is optimized for performance, with minimized script

size. Debug mode provides more robust debugging features, such as type and argument checking. ASP.NET runs the debug versions when the application is in debug mode. This enables you to throw exceptions in debug scripts while minimizing the size of release code.

Script support for AJAX in ASP.NET is used to provide two important features:

- The Microsoft AJAX Library, which is a type system and a set of JavaScript extensions that provide namespaces, inheritance, interfaces, enumerations, reflection, and additional featuresPartial-page rendering, which updates regions of the page by using an asynchronous postback.

## **2. Localization**

The ASP.NET AJAX architecture builds on the foundation of the ASP.NET 2.0 localization model. It provides additional support for localized .js files that are embedded in an assembly or that are provided on disk. ASP.NET can serve localized client scripts and resources automatically for specific languages and regions.

## **3. Web Services**

With AJAX functionality in an ASP.NET Web page, you can use client script to call both ASP.NET Web services (.asmx) and Windows Communication Foundation (WCF) services (.svc). The required script references are automatically added to the page, and they in turn automatically generate the Web service proxy classes that you use from client script to call the Web service.

You can also access ASP.NET Web services without using ASP.NET AJAX server controls (for example, if you are using a different Web development environment). To do so, in the page you can manually include references to the Microsoft AJAX Library, to script files, and to the Web service itself. At run time, ASP.NET generates the proxy classes that you can use to call the services.

---

#### 4. Application Services

Application services in ASP.NET are built-in Web services that are based on ASP.NET forms authentication, roles, and user profiles. These services can be called by client script in an AJAX-enabled Web page, by a Windows client application, or by a WCF-compatible client.

#### 5. Server Controls

ASP.NET AJAX server controls consist of server and client code that integrate to produce rich client behavior. When you add an AJAX control to an ASP.NET Web page, the page automatically sends supporting client script to the browser for AJAX functionality. You can provide additional client code to customize the functionality of a control, but this is not required.

The following list describes the most frequently used ASP.NET AJAX server controls.

##### ScriptManager

Manages script resources for client components, partial-page rendering, localization, globalization, and custom user scripts. The ScriptManager control is required in order to use the UpdatePanel, UpdateProgress, and Timer controls.

##### UpdatePanel

Enables you to refresh selected parts of the page, instead of refreshing the whole page by using a synchronous postback.

##### UpdateProgress

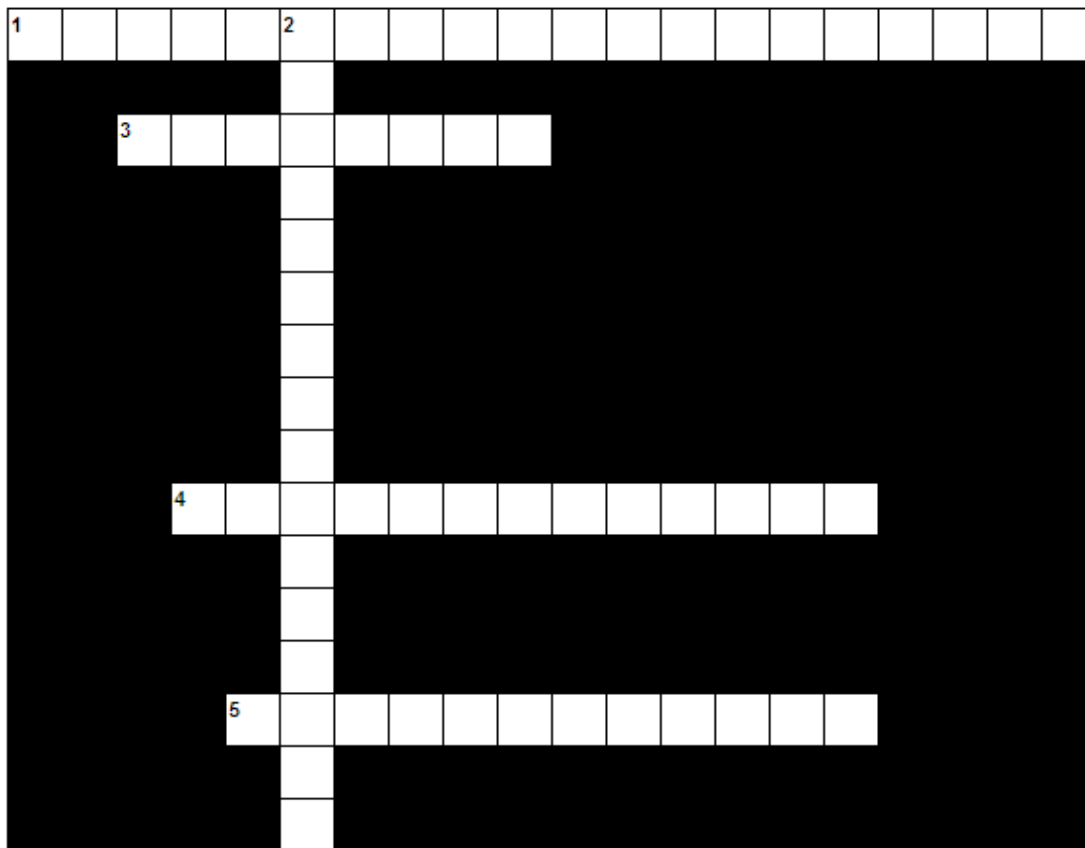
Provides status information about partial-page updates in UpdatePanel controls.

##### Timer

Performs postbacks at defined intervals. You can use the Timer control to post the whole page, or use it together with the UpdatePanel control to perform partial-page updates at a defined interval.

You can also create custom ASP.NET server controls that include AJAX client behaviors. Custom controls that enhance the capabilities of other ASP.NET Web controls are referred to as *extender controls*.

**Crossword:**  
**Chapter 7 & 8**  
**Estimated Time :10 Mins**



**ACROSS:**










1. This layer in the AJAX Client architecture provides AJAX scripting compatibility for the most frequently used browsers.[BrowserCompatibility].
3. AJAX Client components fall into three categories-Components,Behaviours and \_\_\_\_\_.[Controls]
4. A process of creating a product or service that is successful in many countries without modifications. [Globalization]
5. It gives the power to support different languages in the web application. [Localization]

**DOWN:**

3. Custom controls that enhance the capabilities of other ASP.Net Web Controls are referred to as \_\_\_\_\_.[ExtenderControls]

## 9.0 Using the ASP.net AJAX Control Toolkit

### Topics

-  **9.1 Accordion Control**
-  **9.2 AlwaysVisibleControl Extender**
-  **9.3 AutoCompleteExtender**
-  **9.4 CollapsiblePanelExtender**
-  **9.5 DragPanelExtender**
-  **9.6 DropShadowExtender**
-  **9.7 PasswordStrength Control**
-  **9.8 RoundedCornersExtender**
-  **9.9 Crossword**



**The AJAX Control Toolkit** is a joint project between the community and Microsoft. Built upon the ASP.NET 3.5 AJAX Extensions, the Toolkit aims to be the biggest and best collection of web-client components available.

The Toolkit addresses three needs:

1. First it gives website developers a place to get components to make their web applications spring to life.
2. Second it gives a set of great examples for those wishing to write client-side code,
3. Third it is a place for the best script developers to get their work highlighted.



**Objectives :** This module will familiarize the participant with

- Basic idea of Accordion Control and how to use it to create Collapsible Panels.



**Presentation :**

**Introduction:** An accordion menu is a collection of multiple, stacked panels. Only one of them is visible and another can be opened by clicking the header of a closed one using a smooth transition of the heights of the content area of the panels. It is like having several Collapsible Panels where only one can be expanded at a time. The Accordion is implemented as a web control that contains AccordionPane web controls. Each AccordionPane control has a template for its Header and its Content. We keep track of the selected pane so it stays visible across postbacks.



**Demonstration/Code Snippet :**

The Aspx Code for Accordion is:

```
<% @ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs"
Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>AJAX CONTROL: Accordion Pane Show</title>
    <link href="MyStyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
        <table cellpadding="0" cellspacing="0" width="50%" align="center">
            <tr>
                <td>
                    <ajaxtoolkit:toolkitscriptmanager id="ScriptManager1" runat="server" />
                    <div>
                        <ajaxtoolkit:accordion id="Accordion1" runat="server" fadetransitions="True"
selectedindex="0"
transitionduration="300" headercssclass="accordionHeader"
contentcssclass="accordionContent">
                            <Panes>
```

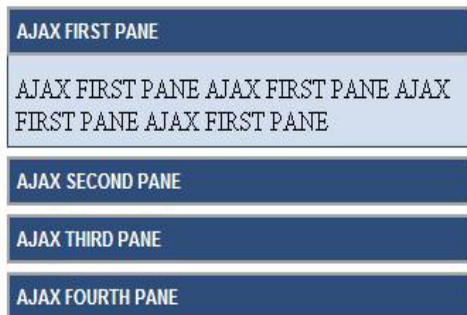


```

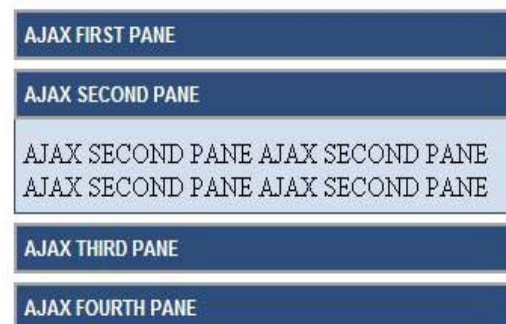
<ajaxToolkit:AccordionPane ID="AccordionPane1" runat="server">
  <Header>
    AJAX FIRST PANE</Header>
  <Content>
    AJAX FIRST PANE AJAX FIRST PANE AJAX FIRST PANE AJAX
FIRST PANE
  </Content>
</ajaxToolkit:AccordionPane>
<ajaxToolkit:AccordionPane ID="AccordionPane2" runat="server">
  <Header>
    AJAX SECOND PANE
  </Header>
  <Content>
    AJAX SECOND PANE AJAX SECOND PANE AJAX SECOND
PANE AJAX SECOND PANE
  </Content>
</ajaxToolkit:AccordionPane>
<ajaxToolkit:AccordionPane ID="AccordionPane3" runat="server">
  <Header>
    AJAX THIRD PANE
  </Header>
  <Content>
    AJAX THIRD PANE AJAX THIRD PANE AJAX THIRD PANE
AJAX THIRD PANE
  </Content>
</ajaxToolkit:AccordionPane>
<ajaxToolkit:AccordionPane ID="AccordionPane4" runat="server">
  <Header>
    AJAX FOURTH PANE
  </Header>
  <Content>
    AJAX FOURTH PANE AJAX FOURTH PANE AJAX FOURTH
PANE AJAX FOURTH PANE
  </Content>
</ajaxToolkit:AccordionPane>
</Panes>
</ajaxtoolkit:accordion>
</div>
</td>
</tr>
</table>
</form>
</body>
</html>

```

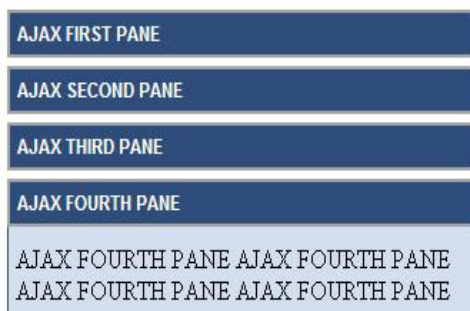
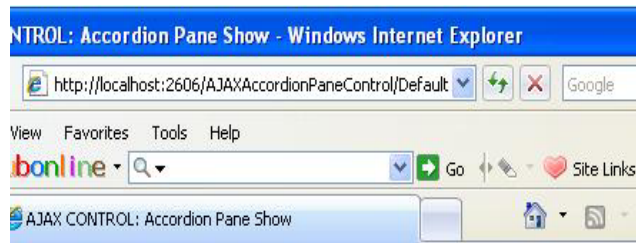
The Running Mode of this is:



When I click on Second Menu then second pane will visible while first become hide.



Finally



**Objectives :** This module will familiarize the participant with

- Basic idea of AlwaysVisible Control and how to position it at a particular location.

**Presentation :**

The AlwaysVisibleControl is a simple extender allowing you to pin controls to the page so that they appear to float over the background body content when it is scrolled or resized. It targets any ASP.NET control and always keeps the position a specified distance from the desired horizontal and vertical sides.

To avoid having the control flash and move when the page loads, it is recommended that you absolutely position the control in the desired location in addition to attaching the extender.

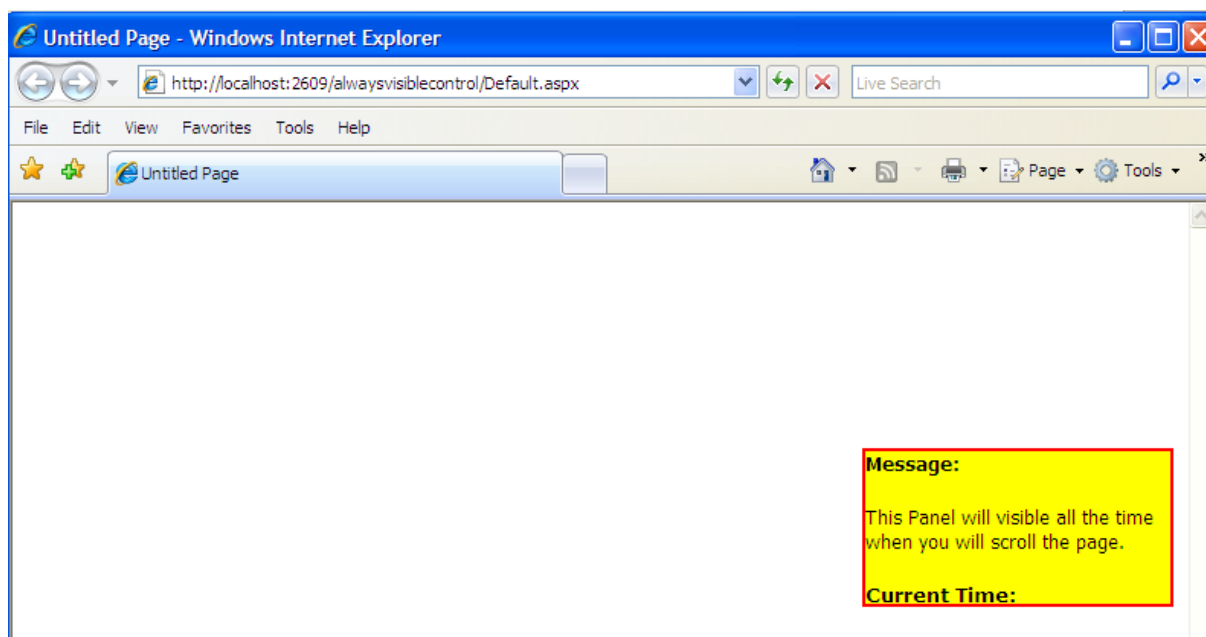
**Demonstration/Code Snippet :**

Here is the code for .aspx page:

```
<asp:ScriptManager ID="ScriptManager1" runat="server" />
<asp:Panel ID="Panel1" BorderStyle=solid BorderWidth=2 runat="server" Height="100px"
Width="200px" BackColor=yellow BorderColor=red>
    <p>
        <span style="font-size: 10pt; font-family: Tahoma">
            <strong>Message:</strong>
        </span>
    </p>
    <div>
        <span style="font-size: 10pt">
            <span style="font-family: Tahoma">
                This Panel will visible all the time when you will scroll the page.
            </span>
        </span>
    </div>
    <p>
        <span style="font-size: 10pt">
            <strong>
                <span style="font-family: Verdana">Current Time:
            </span>
        </strong>
    </p>
```

```
</span>
<asp:Label ID="Label1" runat="server" Font-Bold="True" Font-Names="Verdana"
Font-Size="10pt">
</asp:Label>
</p>
</asp:Panel>
<div>
<cc1:alwaysvisiblecontrolextender ScrolleEffectDuration=".1" HorizontalSide=Right
HorizontalOffset=10
VerticalSide="Middle" VerticalOffset="10" TargetControlID="Panel1"
id="AlwaysVisibleControlExtender1" runat="server">
</cc1:alwaysvisiblecontrolextender>
</div>
```

#### Screenshot:



**Topic: AutoCompleteExtender      Estimated Time: 20 Minutes**



**Objectives :** This module will familiarize the participant with

- Basic idea of AutoCompleteExtender and how to use it to create Collapsible Panels.



**Presentation :**

It is an ASP.NET AJAX extender that can be attached to any TextBox control. When the user types some letters in the Textbox, a popup panel will come to action and displayed the related words. So that the user can choose exact word from the popup panel. Here I tried to explain how this AutoComplete fetches data from the database through a Webservice.

Open Microsoft Visual Studio, click on New Website. Then choose ASP.NET Ajax Enabled Website.

Now drag and drop a Textbox from your Toolbox. Then drag and drop a ScriptManager and AutoCompleteExtender to your Default.aspx page. Then add a webservice to your project as WebService.asmx. First thing you have to do is to add the ScriptService reference to the webserive as follows.

```
[System.Web.Script.Services.ScriptService]
```



**Scenario :**

Mr. Sudhin Acharya wants to develop an online dictionary which will be freely downloadable for students use. In this dictionary he wants to implement a feature, where after pressing any letter, the words starting with that letter will get popped up. So, for this purpose, he was suggested to use AutoCompleExtender Control in his application.



**Demonstration/Code Snippet :**

Now, write a webmethod 'GetCountryInfo' to fetch the data from the country table as follows

```
[WebMethod]
public string[] GetCountryInfo(string prefixText)
{
    int count = 10;
```

```
string sql = "Select * from Country Where Country_Name like @prefixText";
SqlDataAdapter da = new SqlDataAdapter(sql,"Your Connection String Comes Here");
da.SelectCommand.Parameters.Add("@prefixText", SqlDbType.VarChar, 50).Value =
prefixText+ "%";
DataTable dt = new DataTable();
da.Fill(dt);
string[] items = new string[dt.Rows.Count];
int i = 0;
foreach (DataRow dr in dt.Rows)
{
    items.SetValue(dr["Country_Name"].ToString(),i);
    i++;
}
return items;
}
```

The above webmethod takes prefixText as argument, sends it to the query to fetch only the related words that starts with the prefixText values. Then it returns the result as an array of strings.

Next, in the Default.aspx page, set the AutoCompleteExtender's TargetControlID property to the TextBox Id. Now you can see a new Extenders Tab is added in the Textbox's Property window. Set ServicePath as WebService.asmx, ServiceMethod as GetCountryInfo and MinimimPrefixLength as 1.

```
<cc1:AutoCompleteExtender ID="AutoCompleteExtender1" runat="server"
MinimumPrefixLength="1" ServiceMethod="GetCountryInfo"
ServicePath="WebService.asmx" TargetControlID="TextBox1">
</cc1:AutoCompleteExtender>
```

Now, its time to run the project. Select the Default.aspx and click on View in Browser. You can see the excellent application starts to run. Type your country's first letter. See all the countries starts with that letter will appear in the popup.

See the example below..

Enter Your Country :

Enter Your Country :

Enter Your Country :

a
Andorra
Afghanistan
Antigua and Barbuda
Anguilla
Albania
Armenia
Angola
Antarctica
Argentina
American Samoa
Austria
Australia
Aruba
Azerbaijan
Algeria

This is the result after you typed “a” in the textbox.



**Objectives :** This module will familiarize the participant with

- Basic idea of CollapsiblePanel Extender and how to use it to create Collapsible Panels.

**Presentation :**

The CollapsiblePanel is a very flexible extender that allows you to easily add collapsible sections to your web page. This extender targets any ASP.NET Panel control. The page developer specifies which control(s) on the page should be the open/close controller for the panel, or the panel can be set to automatically expand and/or collapse when the mouse cursor moves in or out of it, respectively.

The panel is also post-back aware. On a client postback, it automatically remembers and restores its client state. This demonstrates the ability of these extenders to have some communication between the client and the server code.

**Demonstration/Code Snippet :**

Here is the code for .aspx page

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs"
Inherits="_Default" %>

<%@ Register Assembly="AjaxControlToolkit" Namespace="AjaxControlToolkit"
TagPrefix="cc1" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>Untitled Page</title>
  <link href="ajax.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <form id="form1" runat="server">
    <div>

      <asp:ScriptManager ID="ScriptManager1" runat="server">
```

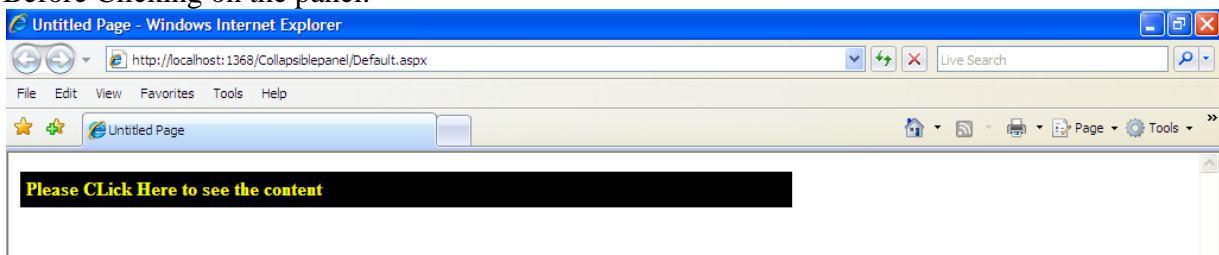
```
</asp:ScriptManager>
<asp:Panel ID="PnlTitle" runat="server" CssClass="collapsePanelHeader">
Please CLick Here to see the content</asp:Panel>

<asp:Panel ID="PnlContent" runat="server" CssClass="collapsePanel">
After Clicking above you will be able to see this text</asp:Panel>

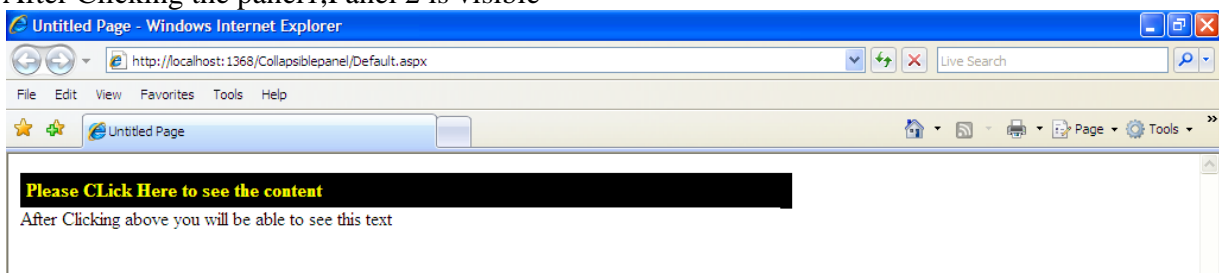
<cc1:collapsiblepaneextender ID="CollapsiblePanelExtender1"
    runat="server" TargetControlID="PnlContent" ExpandControlID="PnlTitle"
    CollapseControlID="PnlTitle"
    Collapsed="True" SuppressPostBack="true">
</cc1:collapsiblepaneextender>
</div>
</form>
</body>
</html>
```

### Screenshots:

Before Clicking on the panel.



After Clicking the panel1, Panel 2 is visible



**Topic: DragPanel Extender      Estimated Time: 20 Minutes**



**Objectives :** This module will familiarize the participant with

- Basic idea of DragPanel Extender and how to use it with mouse based drag drop events.



**Presentation :**

A DragPanel is used mainly to allow you to implement mouse based drag and drop movable content in your ASP.NET web page, where the end user of your website is allowed to move the content and place it anywhere on the page.

In this sample application we will use the Visual Basic or C# language for the sample application. Open Microsoft Visual Studio, click on New Website. Then choose ASP.NET Ajax Enabled Website. Open Default.aspx page in design view.

1. Listing 1 contains the style sheet content that should be placed inside the file. You can copy and paste directly.



**Demonstration/Code Snippet :**

**Listing 1: Style Sheet Content**

```
.dragContainer{
    background-color: #FFC0FF;
    height: 282px;
    width: 357px;
    border-bottom-color: black;
}
.dragHeader{
    background-color: #8080FF;
    height: 48px;
    width: 358px;
}
.dragDetail{
    background-color: #FFC0FF;
    height: 213px;
    width: 357px;
```

```
}
```

2. Go to the source mode, click and drop the style sheet file into the header section of the default web form in order to be able to reference it.

3. Now drag one panel from the standard Toolbox, go to source mode and change the ID tag to **PnlContainer**, delete the height and width tags and insert a cssclass tag with a value **dragContainer** as selected from the style sheet like listing 2.

#### Listing 2: Container Panel Content

```
<asp:Panel ID="PnlContainer" runat="server" cssclass="dragContainer">
```

4. Now, drag two panels from the standard toolbox and drop them inside the container panel.

6. Name the first panel **PnlHeader** and delete the height and width specs. Then create a CssClass tag with value **dragHeader** like in Listing 3 and add the content specified.

#### Listing 3 - Header Panel Content

```
<asp:Panel ID="PnlHeader" runat="server" CssClass="dragHeader">
```

Click and Drag Here To Move ME.....around the page</asp:Panel>

7. Name the second panel **PnlDetail** and delete the height and width specs. Create a CssClass tag with value **dragDetail** like in listing 4.

#### Listing 4 - Detail Panel Content

```
<asp:Panel ID="PnlDetail" runat="server" CssClass="dragDetail">
```

Here is just some content where you can add as much text as you like

```
<br />
line 1<br />
line 2<br />
```

Remember that you can add any html text here

```
<br /><br />
</asp:Panel>
```

8. Place the cursor after the script manager html end tag. From the Toolbox of the Toolkit drag a DragPanel control and set the properties as specified in Listing 5.

---

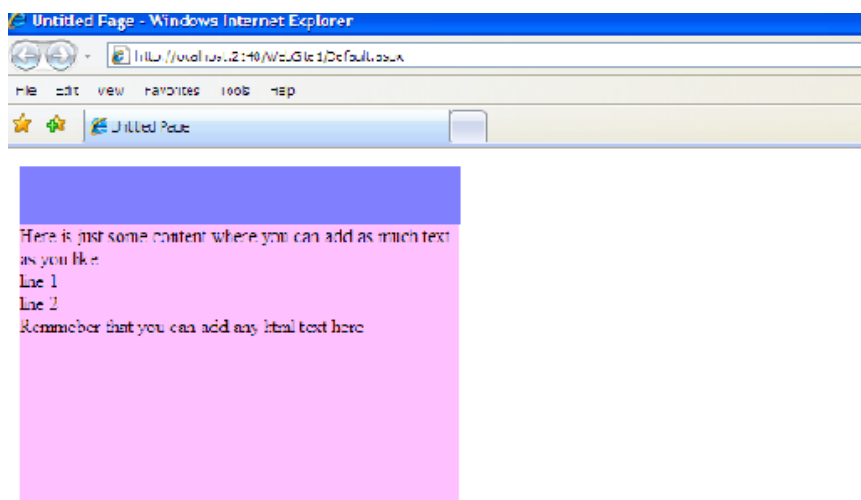
**Listing 5 - DragPanel Extender Format**

```
<ajaxToolkit:DragPanelExtender ID="DragPanelExtender1" runat="server"
    TargetControlID="PnlContainer" DragHandleID="PnlHeader">
</ajaxToolkit:DragPanelExtender>
```

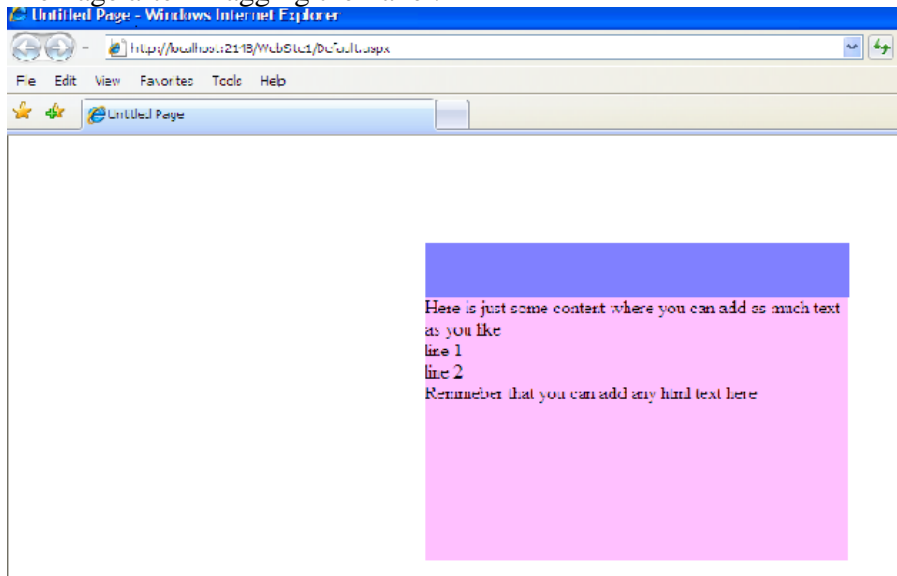
9. Please note that some of these properties cannot be specified using the normal properties window.
10. The target control ID specifies the control to drag and move (the container panel in our case).
11. The drag handle specifies the control that the user is able to click and drag from (the header panel in our case).
12. All tasks performed so far allow you to move the panel around without fixing it in place. You can try this by running your code and testing the panel.

**Screenshots:**

1. Actual Page before Dragging the Panel.



## 2. The Page after Dragging the Panel.



**Objectives :** This module will familiarize the participant with

- Basic idea of DropShadow Extender and how to use it to create shadow to a panel.

**Presentation :****DropShadow Extender**

DropShadow is an extender which applies a "Drop Shadow" to a Panel. It allows you to specify how wide the shadow is as well as how opaque it is, or if you would like rounded corners. For pages that allow the user to move or resize the panel, the DropShadow has a mode that will resize/reposition it to match that of the target panel at run time.

**Demonstration/Code Snippet :**

The code for aspx page is:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs"
Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Ajax :: DropShadowExtender</title>
</head>
<body>
    <form id="form1" runat="server">
        <asp:ScriptManager ID="ScriptManager1" runat="server">
        </asp:ScriptManager>
        <div>
            <table cellpadding="0" cellspacing="0" width="50%" align="center">
                <tr>
                    <td>
                        <asp:Panel ID="PanelLoginControl" runat="server" Height="190px"
Width="395px" BackColor="#00AEEF">
                            <div>
                                <table cellpadding="3" width="98%" align="center" style="margin: 0px 5px
0px 5px;">
```

```

        <tr>
        <td align="center" style="font-weight: bold;">
            AJAX: A Good Control To Make Shadow around a panel
        </td>
        </tr>
    </table>
</div>
</asp:Panel>
</td>
</tr>
</table>
</div>
<div>
    <ajaxToolkit:DropShadowExtender ID="DSExtender" runat="server" Opacity="0.30"
Radius="4"
    Rounded="true" TargetControlID="PanelLoginControl">
    </ajaxToolkit:DropShadowExtender>
</div>
</form>
</body>
</html>

```

When the application runs.

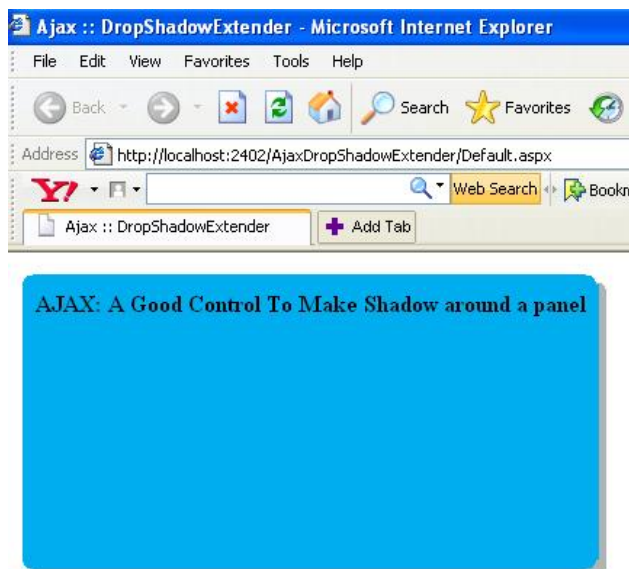


Figure 1.



Here If the Property of ShadowExtender control is changed like  
Rounded="false" Then Result

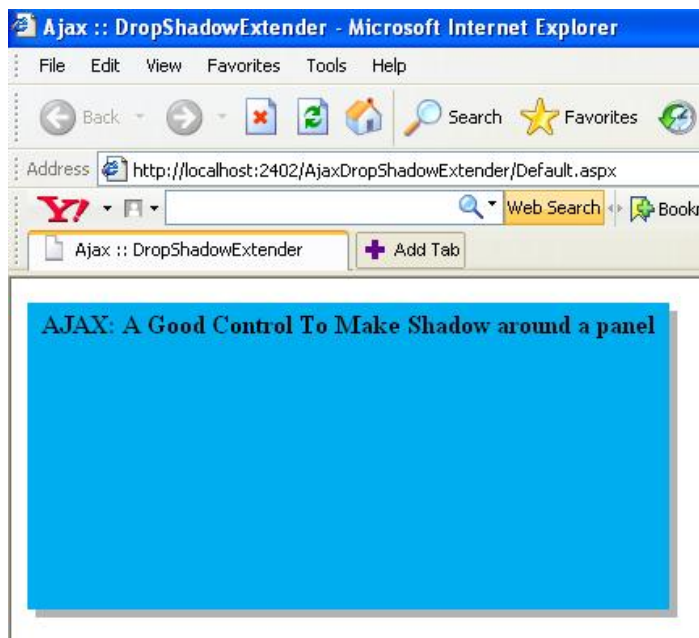
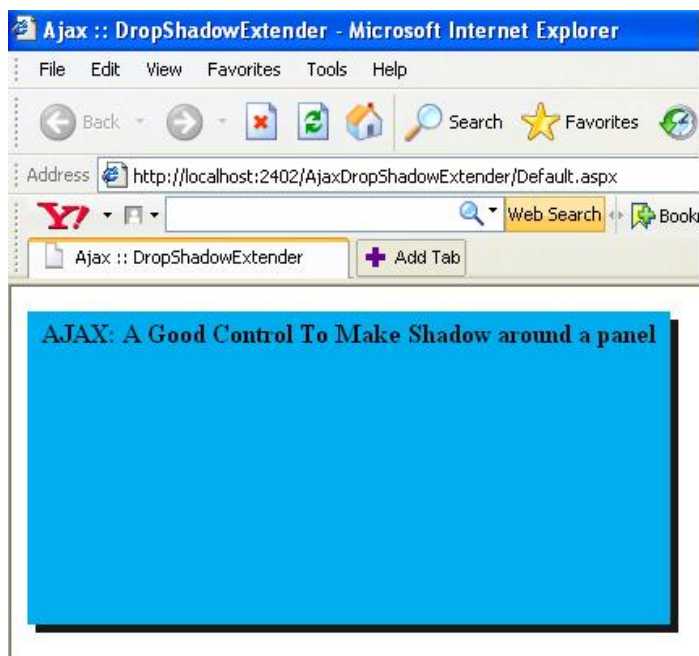


Figure 2.  
If Opacity="0.90" is set. Then Result is



**Objectives :** This module will familiarize the participant with

- Basic idea of Password Strength Control and how to use it to check the strength of user typed password.

**Presentation :**

PasswordStrength is an ASP.NET AJAX extender that can be attached to an ASP.NET TextBox control used for the entry of passwords. The PasswordStrength extender shows the strength of the password in the TextBox and updates itself as the user types the password. The indicator can display the strength of the password as a text message or with a progress bar indicator. The styling and position of both types of indicators is configurable. The required strength of the password is also configurable, allowing the page author to tailor the password strength requirements to their needs. The text messages that describe the current strength of the password can also be configured and their default values have localization support built-in. A help indicator can be used to provide explicit instructions about what changes are required to achieve a strong password. The indicator is displayed when the user begins typing into the TextBox and is hidden from view once the TextBox loses focus.

PasswordStrength control holds the following **properties**:

- TextCssClass
- HelpStatusLabelID
- MinimumNumericCharacters
- MinimumSymbolCharacters
- DisplayPosition
- PreferredPasswordLength
- RequiresUpperAndLowerCaseCharacters
- StrengthIndicatorType
- TargetControlID
- TextStrengthDescriptions

**Scenario :**

Mr. Devenport, who is building an website where the user is suppose to create an account for registering.He wants that the user should be informed their password strength while registering.So, he researched on the net and found that he can take the help of PasswordStrengthControl tool in AJAX toolkit.

**Demonstration/Code Snippet :****Sample:**

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs"
Inherits="_Default" %>

<%@ Register assembly="AjaxControlToolkit" namespace="AjaxControlToolkit"
tagprefix="cc1" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Password Strength</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Panel ID="Panel1" runat="server" BackColor="#cccc66" Height="100px"
Width="500px"><br />&nbsp;

                <asp:Label ID="LabelName" runat="server" Text="Enter Password" Font-
Bold="true"></asp:Label>

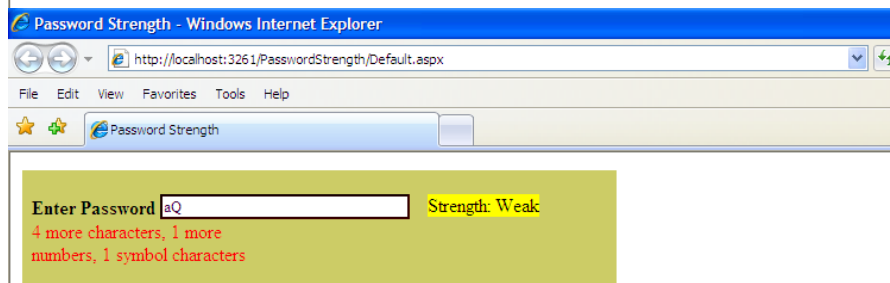
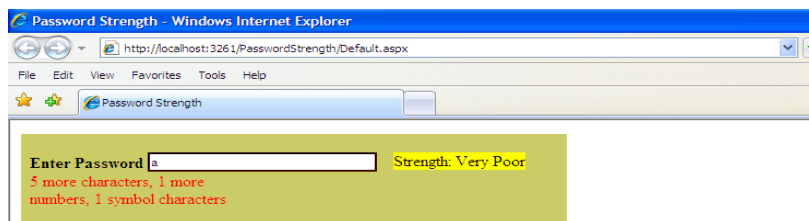
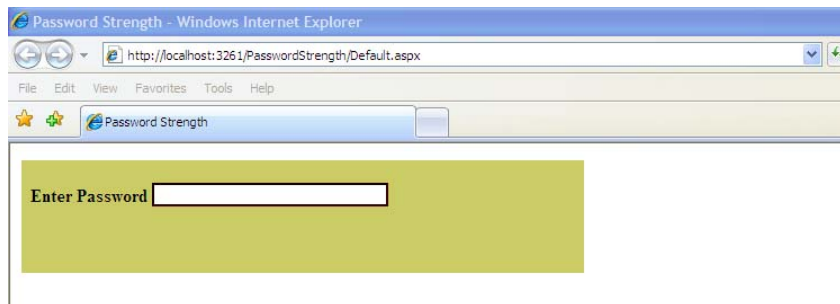
                <asp:TextBox ID="TextBox1" runat="server" Width="204px" BorderColor="#400000"
Font-Names="Times New Roman" ForeColor="#400040"></asp:TextBox><br />&nbsp;

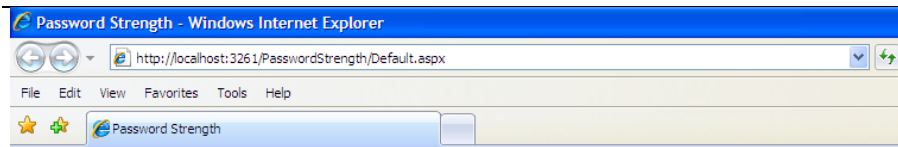
                <asp:Label ID="Label1" runat="server" Width="204px"
ForeColor="red"></asp:Label><br /><br />

            </asp:Panel>
        </div>
        <asp:ScriptManager ID="ScriptManager1" runat="server">
            </asp:ScriptManager>
            <cc1:PasswordStrength ID="PasswordStrength1" runat="server"
HelpStatusLabelID="Label1" MinimumNumericCharacters="1"
MinimumSymbolCharacters="1" DisplayPosition="RightSide" PreferredPasswordLength="6"
RequiresUpperAndLowerCaseCharacters="True" StrengthIndicatorType="Text"
TargetControlID="TextBox1" TextStrengthDescriptions="Very
Poor;Weak;Average;Strong;Excellent">
                </cc1:PasswordStrength>
            </form>
        </body>
```

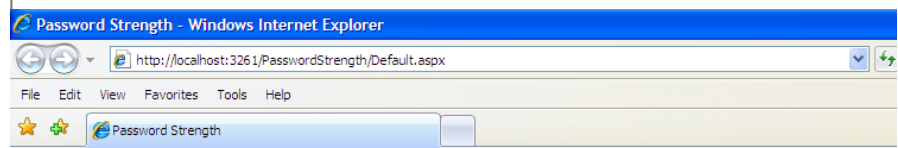
```
</html>
```

## ScreenShots:





Enter Password  Strength: Average  
3 more characters, 1 symbol  
characters



Enter Password  Strength: Strong  
2 more characters

**Objectives :** This module will familiarize the participant with

- Basic idea of RoundedCorners Extender and how to use it to round corners of its target control.

**Presentation :**

The RoundedCorners extender applies rounded corners to existing elements. To accomplish this it inserts elements before and after the element that is selected, so the overall height of the element will change slightly. You can choose the corners of the target panel that should be rounded by setting the Corners property on the extender to None, TopLeft, TopRight, BottomRight, BottomLeft, Top, Right, Bottom, Left, or All.

**Scenario :**

Mr. Pandey, CEO of Meteorological Department of Andhra Pradesh wants to make a website for maintaining weather information, forecasts. His criteria's for the website are that it should look attractive, it should be able to give updates periodically, publish satellite images. So he approaches to Mr. Avishek Chowdhury, Manager, IT Department, who advises him to go for a Ajax enabled Web-Site, where he can meet his requirements using various tools of Ajax Toolkit.

**Demonstration/Code Snippet :**

The code for ASPX Page is.

```
<% @ Page Language="C#" AutoEventWireup="true"
CodeFile="Default.aspx.cs" Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>AJAX: RoundedCorners</title>
</head>
<body>
    <form id="form1" runat="server">
```

```

<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>
<div>
    <table cellpadding="0" cellspacing="0" width="50%" align="center">
        <tr>
            <td>
                <asp:Panel ID="PanelLogin" runat="server" Height="190px" Width="395px"
                BackColor="#00AEEF">
                    <div>
                        <table cellspacing="3" width="98%" align="center" style="margin: 0px 5px
                        0px 5px;">
                            <tr>
                                <td align="center" style="font-weight: bold;">
                                    AJAX: A Good Control To Make rounded Corner
                                </td>
                            </tr>
                        </table>
                    </div>
                </asp:Panel>
            </td>
        </tr>
    </table>
</div>
<div>
    <ajaxToolkit:RoundedCornersExtender ID="rce" runat="server"
    TargetControlID="PanelLogin"
    Corners="Top" Radius="6" />
</div>
</form>
</body>
</html>

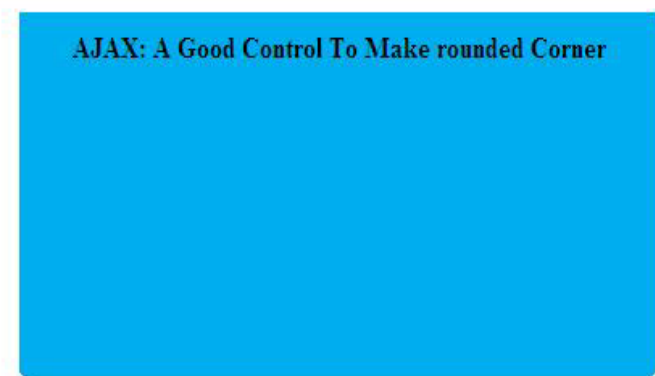
```

### Screenshots:

When the Application is run and Corners="None" is set Then



When the Application is run and Corners=" Bottom "is set Then



When the Application is run and Corners=" BottomLeft "is set Then





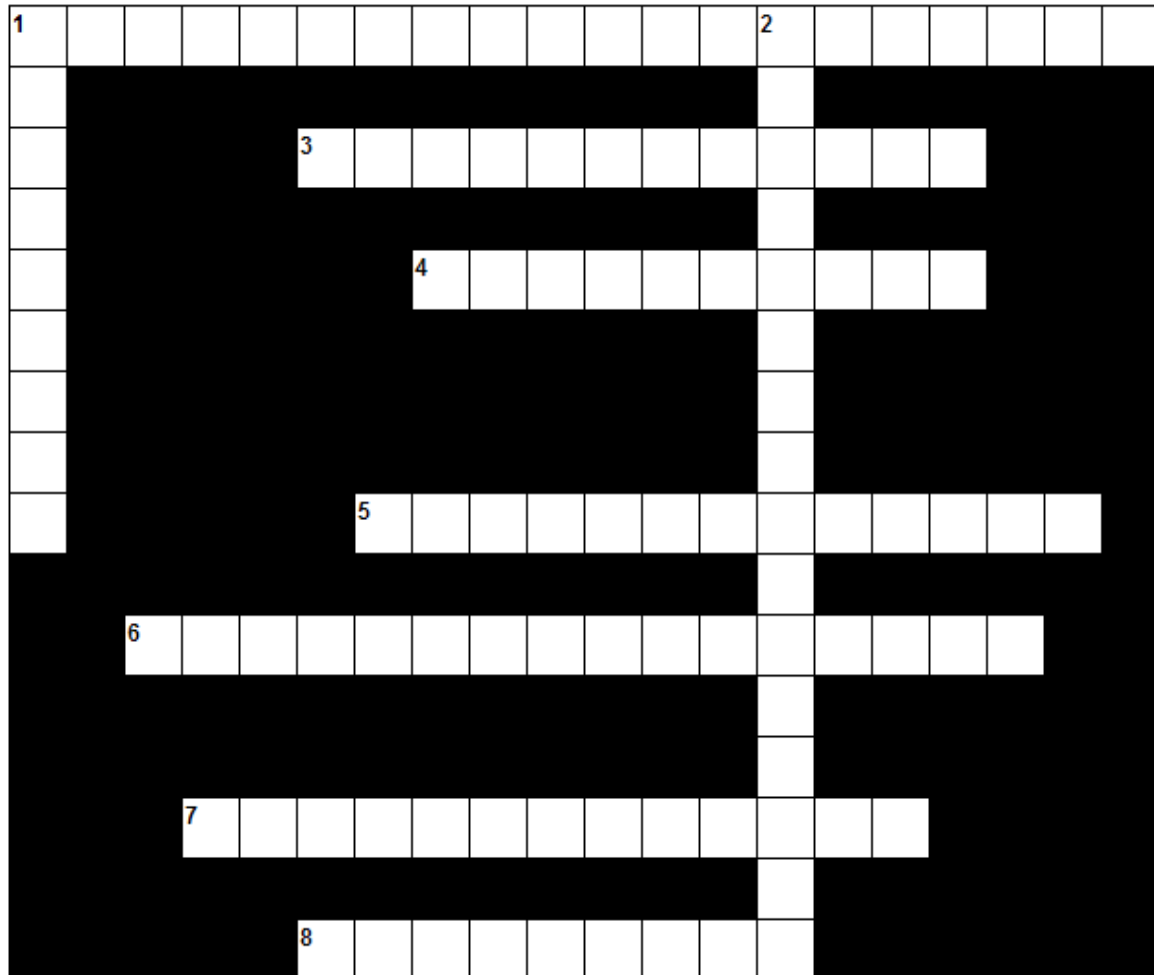
When the Application is run and Corners=" Right "is set Then



When the Application is run and Corners=" Top "is set Then



**AJAX: A Good Control To Make rounded Corner**

**Crossword: Chapter 9**
**Estimated Time: 10 Mins**

**Across:**

1. It keeps controls at a particular position in the page when it is scrolled or resized. [AlwaysVisibleControl].
3. When a user types some letters in a text box, the related words starting with the same letter gets popped up through this extender. [AutoComplete]
4. It allows you to specify how wide the shadow of a panel is. [DropShadow]

---

5.This extender catches clicks on a button or any instance of a type derived from button.[ConfirmButton].

6.The indicator displays how strong the typed password is.[PasswordStrength]

7.It applies rounded corners to existing elements .[RoundedCorner]

8.It is used mainly to implement mouse based drag drop events.[ConfirmButton]

**Down:**

1.A \_\_\_\_\_ menu is a collection of multiple, stacked panels.[Accordion]

2.This extender allows you to easily add collapsible sections to your webpage.[Collapsiblepanel].

## 10.0 Advantages and Disadvantages of using AJAX in application

### Topics

 **10.1 Advantages of using AJAX**

 **10.2 Disadvantages of using AJAX**





**Objectives :** This module will familiarize the participant with

- **Advantages of using AJAX technology in building web-sites.**



**Presentation :**

To help you decide whether AJAX is for you or not, here are some of the advantages it has over classic web development techniques:

1. The interface is much more responsive, because only a small part of the page is transferred at a time. The user has the feeling that changes are instantaneous.
2. In a classic web application, when the web server sends a web page to the browser, it can use multiple connection threads to speed up delivery. However, this happens for content only – what is between the <body> tags. All script and CSS files linked in the page's <head> section are transferred using only one connection thread, which diminishes performance. With AJAX, you only have to load the basic scripts and CSS files, and request the rest as content, through multiple connections. In traditional web applications, multiple connection threads are used to speed up deliveries between client and server. But this only affects content. All scripts and format information are communicated through a single connection thread, which negatively impacts performance. Ajax uses multiple connections to load basic scripts and CSS files; the rest is requested as content.
3. Waiting time is reduced – when the visitor submits a form, they no longer have to wait for the entire page to be rebuilt and re-transmitted by the server. Instead, only the relevant content changes, and in non-critical cases, the visitor can still work while the data is being submitted.
4. If a page section encounters an error, other sections are not affected (if not logically linked) and the data already entered by the user is not lost. This way, the application fails gracefully, without causing head-aches for the users.
5. Traffic to and from the server is reduced considerably – because you do not have to send the entire page content (CSS, images, static sections), the bandwidth usage is most likely to decrease.



**Objectives :** This module will familiarize the participant with

- **Disadvantages of using AJAX technology in building web-sites.**



**Presentation :**

As with most new web development techniques, AJAX has its critics. There are some disadvantages that need to be considered before implementing an AJAX based solution:

1. Building an AJAX-powered application can increase development time and costs. It is usually considered more difficult than building a classic web application, because of the mixture of technologies and the special concern about everything going smoothly. However, because it is dealing with relatively known technologies, AJAX is not rocket science.
2. Although AJAX relies on existing and mature technologies like Javascript, HTML and CSS, the available frameworks and components still need to completely mature. Tools like the [Dojo toolkit](#) or the [Rico framework](#) are just a milestone on this long road.
3. Not all concerns regarding security and user privacy have been answered. This fueled a wave of criticism about how safe is the AJAX way of building applications.
4. Using AJAX to asynchronously load bits of content into an existing page conflicts with the way we are used to navigate and create bookmarks in modern browsers. Because viewing some part of the page information no longer corresponds to a newly loaded page, the browser history and bookmarks will not be able to restore the exact page state. Also, clicking the Back button in a browser might not have any effect, since the URL was unchanged (even if parts of the page were changed). To overcome these problems you must implement a way to save the current page state so that it can be restored later on, when called from a bookmark or from the browser history.
5. AJAX is not meant to be used in every application. One of the main reasons for this stays in the fact that Google cannot index it. For example, suppose you are building an e-commerce site. A complete AJAX application would be a mistake because search engines won't be able to index it. And without a search engine, a site won't be able to sell products. Keeping this in mind, a much better idea than creating complete AJAX applications would be to scatter AJAX features within the application.
6. The biggest concern with AJAX is accessibility. This is because not all browsers (especially older ones) have complete support for JavaScript or the XMLHttpRequest object. Some of the visitors do have browsers with the required features, but they choose or have to turn off JavaScript support. When you design the application you must make sure that a fail-safe solution exists for those users, so it can be accessed by anyone. Further more, the way to access and use the XMLHttpRequest object in Internet Explorer and other browsers is different, which leads to a fork in the code and the need to treat each case separately.

- 
7. The last disadvantage lies in the actual XMLHttpRequest object itself. Due to security constraints, you can only use it to access information from the host that served the initial page. If you need to display information from another server, it is not possible within the AJAX paradigm.



## **\*\*Answers for Crosswords**

### **Chapter-1**

<b>Across</b>	2) XMLHTTPREQUEST 3) AJAXENGINE 4) WEBBROWSER 5) WEBSERVER
<b>Down</b>	1) ASYNCHRONOUS

### **Chapter-2**

<b>Across</b>	1) UpdateProgress3) FIVEHUNDRED4) UPDATEPANEL5) SLEEP
<b>Down</b>	3) SCRIPTMANAGER

### **Chapter-3&4**

<b>Across</b>	1) ScriptHandlerFactory3) ScriptService4) Miliseconds5) PageMethods6) OnRequestFailed7) Interval8) WebMethod
<b>Down</b>	2) AutoCompleteExrender

### **Chapter-5**

<b>Across</b>	1) CreateUser 3) ASPNETDB 4) Callback 5) Three
<b>Down</b>	2) AXPXAUTH

### **Chapter-6**

<b>Across</b>	1) SAVECOMPLETECALLBACK 3) USERCONTEXT 4) failedcallback 5) ECMAScript
<b>Down</b>	2) ProfileService

### **Chapter-7&8**

<b>Across</b>	1) BrowserCompatibility 3) Controls 4) Globalization 5) Localization
<b>Down</b>	2) ExtenderControls

### **Chapter-9**

<b>Across</b>	1) AlwaysVisibleControl 3) AutoComplete 4) DropShadow 5) ConfirmButton 6) PasswordStrength 7) RoundedCorner 8) ConfirmButton
<b>Down</b>	1) Accordion 2) Collapsiblepanel

### Team Members



**Srikanth Nivarthi**  
47275



**Sreenivas Ram**  
66223



**Seshu Babu Barma**  
56150



**Radhika Shashidhar jaji**  
75217



**Ashok Sharma**  
62751



**Veerendra Kumar Ankem**  
77964



**Teena Arora**  
74572

### Contributors



**Mohit Kumar Sharma**   **Avishek Roy Chowdhury**  
73127                                      73001