502 SreeNilayam, Lanco Hills Road,
Manikonda, Hyderabad – 500089
08413485772 (landline) / 9701377070
Email : hr@syllogistek.com

Contents

AJAX

AJAX stands for Asynchronous JavaScript and XML. This is a cross platform technology which speeds up response time. The AJAX server controls add script to the page which is executed and processed by the browser.

Ajax is a set of web development techniques using many web technologies on the client side to create asynchronous web applications. With Ajax, web applications can send and retrieve data from a server asynchronously (in the background) without interfering with the display and behaviour of the existing page. By decoupling the data interchange layer from the presentation layer, Ajax allows web pages and, by extension, web applications, to change content dynamically without the need to reload the entire page. In practice, modern implementations commonly utilize JSON instead of XML.

AJAX is not a single technology, but rather a group of technologies. HTML and CSS can be used in combination to mark up and style information. The webpage can then be modified by JavaScript to dynamically display—and allow the user to interact with—the new information. The built-in **XMLHttpRequest** object, or since 2017 the new "fetch()" function within JavaScript is commonly used to execute Ajax on web pages allowing websites to load content onto the screen without refreshing the page. Ajax is not a new technology, or different language, just existing technologies used in new ways.

- Update a web page without reloading the page
- Request data from a server - after the page has loaded
- Receive data from a server - after the page has loaded
- Send data to a server - in the background

Advantages

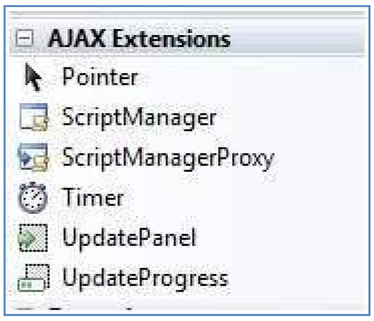
- Reduces the traffic travels between the client and the server.
- No cross browser pain.
- Better interactivity and responsiveness.
- With AJAX, several multipurpose applications and features can be handled using a single web page(SPA).
- API's are good because those work with HTTP method and JavaScrtipt.

Disadvantages

- Search engines like Google would not be able to index an AJAX application.
- It is totally built-in JavaScript code. If any user disables JS in the browser, it won't work.
- The server information cannot be accessed within AJAX.
- Security is less in AJAX applications as all the files are downloaded at client side.
- The data of all requests is URL-encoded, which increases the size of the request.

Like other ASP.NET server controls, Microsoft provides server controls for AJAX which can have methods and event handlers associated with them, which are processed on the server side.

The control toolbox in the Visual Studio IDE contains a group of controls called the 'AJAX Extensions'



ScriptManager Control

The ScriptManager control is the most important control and must be present on the page for other controls to work.

ScriptManager control registers the script for the Microsoft AJAX Library with the page. This enables client script support features such as partial-page rendering and Web-service calls.

You must use a ScriptManager control on a page to enable the following features of ASP.NET AJAX:

1. Client-script functionality of the Microsoft AJAX Library, and any custom script that you want to send to the browser.
2. Partial-page rendering, which enables regions on the page to be independently refreshed without a post back. The ASP.NET AJAX UpdatePanel, UpdateProgress, and Timer controls require a ScriptManager control to support partial-page rendering.
3. JavaScript proxy classes for Web services, which enable you to use client script to access Web services by exposing Web services as strongly typed objects.
4. JavaScript classes to access ASP.NET authentication and profile application services.

Example

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

</asp:ScriptManager>
```

UpdatePanel

An **UpdatePanel** is a set of components that you want to be affected by AJAX updates in your web application. The triggers are what cause the panel of components to update. The UpdatePanel control is a container control and derives from the Control class. It acts as a container for the child controls within it and does not have its own interface. When a control inside it triggers a post back, the UpdatePanel intervenes to initiate the post asynchronously and update just that portion of the page.

For example, if a button control is inside the update panel and it is clicked, only the controls within the update panel will be affected, the controls on the other parts of the page will not be affected. This is called the partial post back or the asynchronous post back.

Properties of the UpdatePanel Control

The following table shows the properties of the update panel control:

Properties	**Description**
ChildrenAsTriggers	This property indicates whether the post backs are coming from the child controls, which cause the update panel to refresh.
ContentTemplate	It is the content template and defines what appears in the update panel when it is rendered.
ContentTemplateContainer	Retrieves the dynamically created template container objects and used for adding child controls programmatically.
IsInPartialRendering	Indicates whether the panel is being updated as part of the partial post back.
RenderMode	Shows the render modes. The available modes are Block and Inline.
UpdateMode	Gets or sets the rendering mode by determining some conditions.
Triggers	Defines the collection trigger objects each corresponding to an event causing the panel to refresh automatically.

Methods of the UpdatePanel Control

The following table shows the methods of the update panel control:

Methods	**Description**
CreateContentTemplateContainer	Creates a Control object that acts as a container for child controls that define the UpdatePanel control's content.
CreateControlCollection	Returns the collection of all controls that are contained in the UpdatePanel control.
Initialize	Initializes the UpdatePanel control trigger collection if partial-page rendering is enabled.
Update	Causes an update of the content of an UpdatePanel control.

Behaviour of UpdatePanelControl

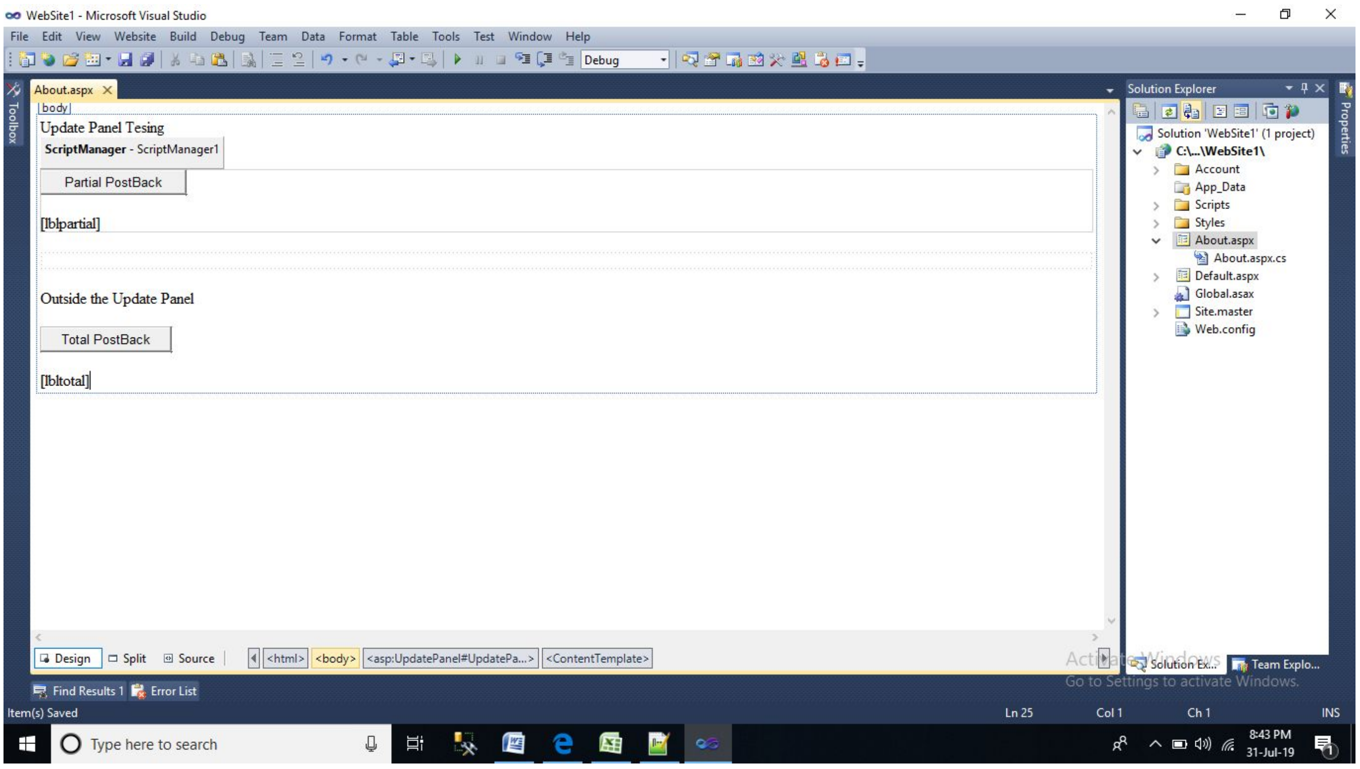
The behaviour of the update panel depends upon the values of the UpdateMode property and ChildrenAsTriggers property.

UpdateMode	ChildrenAsTriggers	Effect
Always	False	Illegal parameters.
Always	True	UpdatePanel refreshes if whole page refreshes or a child control on it posts back.
Conditional	False	UpdatePanel refreshes if whole page refreshes or a triggering control outside it initiates a refresh.
Conditional	True	UpdatePanel refreshes if whole page refreshes or a child control on it posts back or a triggering control outside it initiates a refresh.

Example

Add an AJAX web form in your application. It contains the script manager control by default. Insert an update panel. Place a button control along with a label control within the update panel control. Place another set of button and label outside the panel.

The design view looks as follows:



The source file is as follows:

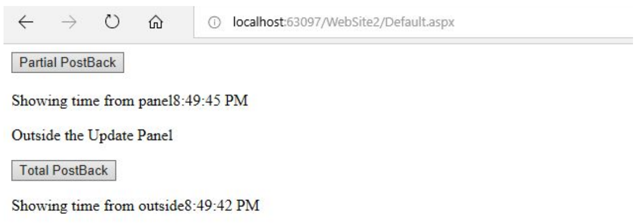
```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default" %>
```

```
<!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>Update Panel Testing</title>
</head>
<body>
  <form id="form1" runat="server">
  <div>
    <asp:ScriptManager ID="ScriptManager1" runat="server" />
  </div>
  <asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
      <asp:Button ID="btnpartial" runat="server" OnClick="btnpartial_Click" Text="Partial PostBack" />
      <br />
      <br />
      <asp:Label ID="lblpartial" runat="server"></asp:Label>
    </ContentTemplate>
  </asp:UpdatePanel>
  <p>
  </p>
  <p>
    Outside the Update Panel</p>
  <p>
    <asp:Button ID="btntotal" runat="server" OnClick="btntotal_Click" Text="Total PostBack" />
  </p>
  <asp:Label ID="lbltime" runat="server"></asp:Label>
  </form>
</body>
</html>
```

Default.aspx.cs code

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

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

  }

  protected void btnpartial_Click(object sender, EventArgs e)
  {
    string time = DateTime.Now.ToLongTimeString();
    lblpartial.Text = "Showing time from panel" + time;
  lbltime.Text = "Showing time from outside" + time;
  }

  protected void btntotal_Click(object sender, EventArgs e)
  {
    string time = DateTime.Now.ToLongTimeString();
    lblpartial.Text = "Showing time from panel" + time;
  lbltime.Text = "Showing time from outside" + time;
  }
}
```

Observe that when the page is executed, if the total post back button is clicked, it updates time in both the labels but if the partial post back button is clicked, it only updates the label within the update panel.



```
Showing time from panel8:49:45 PM

Outside the Update Panel
```

```
Showing time from outside8:49:42 PM
```

A page can contain multiple update panels with each panel containing other controls like a grid and displaying different part of data.

When a total post back occurs, the update panel content is updated by default. This default mode could be changed by changing the UpdateMode property of the control. Let us look at other properties of the update panel.

UpdateProgress Control

The UpdateProgress control provides a sort of feedback on the browser while one or more update panel controls are being updated. For example, while a user logs in or waits for server response while performing some database oriented job.

When a postback event originates from inside an UpdatePanel control, any associated **UpdateProgress** controls are displayed.

It provides a visual acknowledgement like "Loading page...", indicating the work is in progress.

Properties of the UpdateProgress Control

The following table shows the properties of the update progress control:

Properties	Description
AssociatedUpdatePanelID	Gets and sets the ID of the update panel with which this control is associated.
Attributes	Gets or sets the cascading style sheet (CSS) attributes of the UpdateProgress control.
DisplayAfter	Gets and sets the time in milliseconds after which the progress template is displayed. The default is 500.
DynamicLayout	Indicates whether the progress template is dynamically rendered.
ProgressTemplate	Indicates the template displayed during an asynchronous post back which takes more time than the DisplayAfter time.

Methods of the UpdateProgress Control

The following table shows the methods of the update progress control:

Methods	Description
GetScriptDescriptors	Returns a list of components, behaviors, and client controls that are required for the UpdateProgress control's client functionality.
GetScriptReferences	Returns a list of client script library dependencies for the UpdateProgress control.

Example

```
<asp:UpdateProgress ID="UpdateProgress1" runat="server" DynamicLayout="true"
AssociatedUpdatePanelID="UpdatePanel1" >

   <ProgressTemplate>
      Loading...
   </ProgressTemplate>

</asp:UpdateProgress>
```

The above snippet shows a simple message within the ProgressTemplate tag. However, it could be an image or other relevant controls. The UpdateProgress control displays for every asynchronous post back unless it is assigned to a single update panel using the AssociatedUpdatePanelID property.

Trigger:

When a button is outside of the update panel it can be used to associate with a update panel using trigger

Ajax UpdatePanel contains property called UpdateMode this property is used to specify whether UpdatePanel is always refreshed during a partial render or if it refresh only when a particular trigger hit. By default UpdatePanel contains UpdateMode="Always" if we want to set it conditionally we need to change this property UpdateMode="Conditional"

Ajax UpdatePanel control contains two child tags those are **ContentTemplate** and **Triggers**.

ContentTemplate is used to hold the content of the page means suppose we designed page with some controls we will place controls inside of the ContentTemplate

Triggers we used in a situation like need to refresh UpdatePanel only whenever I click some button control in that situation I will define those controls with this Triggers child tag.

Example

```
<asp:UpdatePanel ID="UpdatePanel2" runat="server"
            UpdateMode="Conditional">
            <ContentTemplate>
                <asp:Label ID="lblTrigger" runat="server"
                    ForeColor="red" />
            </ContentTemplate>
            <Triggers>
                <asp:AsyncPostBackTrigger ControlID="btnTrigger"
                    EventName="Click" />
            </Triggers>
        </asp:UpdatePanel>
        <p>
            <asp:Button ID="btnTrigger" runat="server" Text="Using Trigger" OnClick="btnTrigger_Click" />
        </p>
```

```
protected void btnTrigger_Click(object sender, EventArgs e)
        {
            string time = DateTime.Now.ToLongTimeString();
            lblTrigger.Text = "Showing time using trigger" + time;
          }
```