**Script manager control in asp.net**

In ASP.NET, the **ScriptManager** control is part of the ASP.NET AJAX framework and is used to manage client-side scripts in web applications. It is typically used in combination with AJAX-enabled controls and provides functionality for managing and registering JavaScript files and scripts required for client-side functionality.

The primary purpose of the **ScriptManager** control is to:

1. Manage AJAX scripts: It helps in including the necessary JavaScript libraries for AJAX functionality, such as Microsoft's AJAX Control Toolkit or other custom AJAX scripts.
2. Enable partial-page rendering: It allows you to enable AJAX-based partial-page rendering, which means only specific parts of a web page are updated, rather than refreshing the entire page. This can lead to improved performance and a more seamless user experience.
3. Manage Script References: It allows you to register and manage client-side script references, ensuring that scripts are included on the page when they are needed. You can include both built-in ASP.NET AJAX scripts and custom scripts.

Here's an example of how to use the **ScriptManager** control in an ASP.NET web form:

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

**<Scripts>**

**<!-- Include built-in scripts -->**

**<asp:ScriptReference Name="MicrosoftAjax.js" Assembly="System.Web.Extensions" />**

**<asp:ScriptReference Name="MicrosoftAjaxWebForms.js" Assembly="System.Web.Extensions" />**

**<!-- Include custom scripts -->**

**<asp:ScriptReference Path="~/Scripts/CustomScript.js" />**

**</Scripts>**

**</asp:ScriptManager>**

**<!-- Other ASP.NET controls and content on your page -->**

n this example, the **ScriptManager** is defined with an ID of "ScriptManager1." It includes built-in scripts for Microsoft Ajax, and it also includes a custom script located in the "~/Scripts/CustomScript.js" file.

By using the **ScriptManager** control, you can ensure that your scripts are loaded and managed correctly, and it also provides features like script combining and minification for improved performance.

Keep in mind that starting with ASP.NET 4.0 and later versions, you can use the **<asp:ScriptManager>** control without needing to specify scripts explicitly. It can automatically manage scripts required for AJAX functionality.

Additionally, if you are working with ASP.NET Core, the **ScriptManager** control is not available as ASP.NET Core has a different approach to managing client-side resources. Instead, you typically use libraries like jQuery and manage your scripts using standard HTML and JavaScript techniques.

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.

However like other ASP.NET server controls, these AJAX server controls also 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'



The ScriptManager Control

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

It has the basic syntax:

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

</asp:ScriptManager>

If you create an 'Ajax Enabled site' or add an 'AJAX Web Form' from the 'Add Item' dialog box, the web form automatically contains the script manager control. The ScriptManager control takes care of the client-side script for all the server side controls.

The UpdatePanel Control

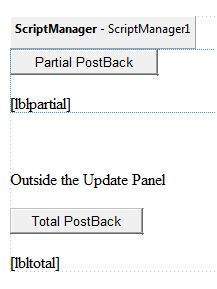
The UpdatePanel control is a container control and derives from the Control class. It acts as a container for the child controls within it and does not have its own interface. When a control inside it triggers a post back, the UpdatePanel intervenes to initiate the post asynchronously and update just 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.

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:

<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="lbltotal" runat="server"></asp:Label>

</form>

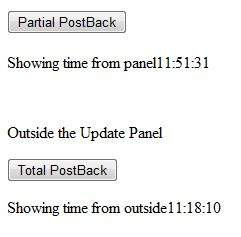
Both the button controls have same code for the event handler:

string time = DateTime.Now.ToLongTimeString();

lblpartial.Text = "Showing time from panel" + time;

lbltotal.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.



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.

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 object 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. |

The behavior 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. |

The 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.

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

The syntax for the UpdateProgress control is:

<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 postback unless it is assigned to a single update panel using the AssociatedUpdatePanelID property.

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. |

The Timer Control

The timer control is used to initiate the post back automatically. This could be done in two ways:

(1) Setting the Triggers property of the UpdatePanel control:

<Triggers>

<asp:AsyncPostBackTrigger ControlID="btnpanel2" EventName="Click" />

</Triggers>

(2) Placing a timer control directly inside the UpdatePanel to act as a child control trigger. A single timer can be the trigger for multiple UpdatePanels.

<asp:UpdatePanel ID="UpdatePanel1" runat="server" UpdateMode="Always">

<ContentTemplate>

<asp:Timer ID="Timer1" runat="server" Interval="1000">

</asp:Timer>

<asp:Label ID="Label1" runat="server" Height="101px" style="width:304px" >

</asp:Label>

</ContentTemplate>

</asp:UpdatePanel>