

# ***Chapter 7***

## ***Introducing AJAX Controls***



***Chapter 7***

***Introducing AJAX Controls***

***eduplurash***

***Just Another Way To Learn***

# AJAX

- AJAX stands for ***Asynchronous Java and XML***.
- Microsoft has both a server-side Ajax framework and a client-side Ajax framework.
- The advantage of the server-side framework is that it provides existing ASP.NET developers with a painless method of doing Ajax.
- The disadvantage of the server-side framework is that it doesn't escape all the problems associated with a server-side framework. You still have to run back to the server whenever you perform any client-side action.
- To build applications using client side AJAX you must use JavaScript.
- The advantage of building applications with the client-side framework is that you can build rich and responsive web applications.

# AJAX

## Debugging the AJAX applications

- You can use a tool called fiddler. You can download this tool (for free) at <http://www.fiddlertool.com>.
- The other critical Ajax debugging tool is Firebug, which is a free Firefox extension.
- You can download Firebug by launching Firefox and selecting Tools, Add-ons.



# ASP .Net AJAX Controls

- ScriptManager
- ScriptManagerProxy
- UpdatePanel
- Timer
- UpdateProgress



educrash  
Just Another Way To Learn

# ScriptManager Control

You must use a [ScriptManager](#) control on a page to enable the following features of ASP.NET AJAX:

- *Client-script functionality of the Microsoft AJAX Library* can be accessed only if we use ScriptManager.
- ScriptManager enables us to use Partial-page rendering. *Partial-page rendering*, enables regions on the page to be independently refreshed without a postback.
- The ASP.NET AJAX [UpdatePanel](#), [UpdateProgress](#), and [Timer](#) controls require a [ScriptManager](#) control to support partial-page rendering else they will not work.
- It is used to access web-services through JavaScript proxy classes.
- It is used by JavaScript classes to access ASP.NET authentication and profile application services.

# ScriptManagerProxy Control

- We can have *only one ScriptManager Control on a web 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,***

- Consider a scenario in which we have a ScriptManager control on the Master Page. Let a web page is using this Master Page. So this web page can not have its own ScriptManager control.
- Hence in order to access a ScriptManager control that is defined in a master page from a content page, you can use the ScriptManagerProxy.

# ScriptManagerProxy Control

That is in short

If a page already contains a ScriptManager control, but a nested or parent component needs additional features of the ScriptManager control, the component can include a ScriptManagerProxy control.



# Using the UpdatePanel Control

- Microsoft's server-side AJAX framework consists of one main control: UpdatePanel.
- The UpdatePanel control enables you to update a portion of a page without updating the entire page.
- In other words, it enables you to perform partial-page rendering.
- You can do nesting of Update Panels to any levels.

- **Example:**

Let's start with an example of a page that uses the UpdatePanel control.

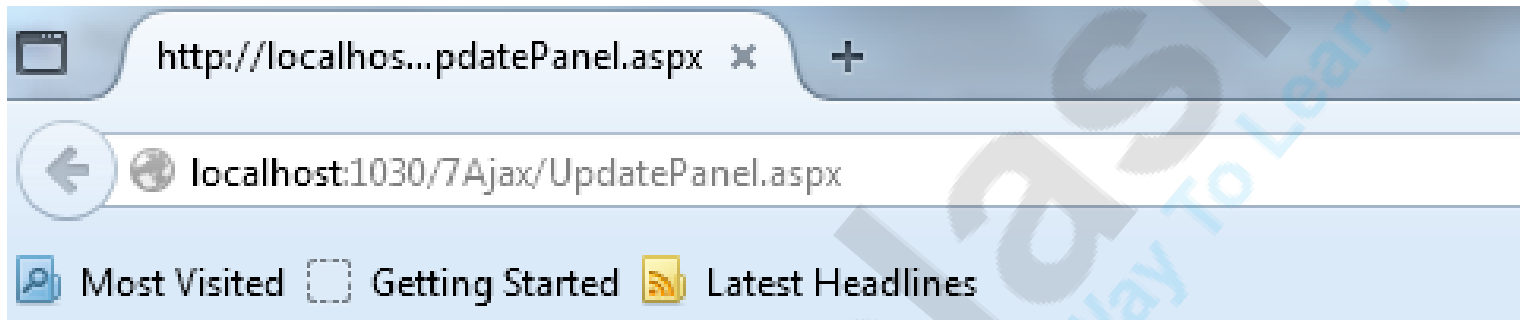
The example contains a ScriptManager control and an UpdatePanel control.

The UpdatePanel control contains a single Button control.

When you click the button, only the content contained in the UpdatePanel control is refreshed



# UpdatePanel Control



This is an Example for Update Panel Control

The Page Time is: AM 03:36:06

The Update Panel controls time is: AM 03:37:57

Get Time

# UpdatePanel Control

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

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

```
<script runat="server">
```

```
protected void Page_Init(object sender, EventArgs e)
```

```
{
```

```
    lblPage.Text = DateTime.Now.ToString("T");
```

```
}
```

```
protected void btnUpdate_Click(object sender, EventArgs e)
```

```
{
```

```
    lblPanel.Text = DateTime.Now.ToString("T");
```

```
}
```

```
</script>
```

# UpdatePanel Control

```
<html xmlns="http://www.w3.org/1999/xhtml">  
<head runat="server">  
  <title></title>  
</head>
```



educclash  
Just Another Way To Learn

# UpdatePanel Control

```
<body>
  <form id="form1" runat="server">
    <div>
      <asp:ScriptManager ID="ScriptManager1" runat="server"> </asp:ScriptManager>
```

This is an Example for Update Panel Control <br /> <br />

The Page Time is:

```
<asp:Label ID="lblPage" runat="server" Text=""></asp:Label>
```

```
<br /> <br />
```

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
```

```
  <ContentTemplate>
```

The Update Panel controls time is:

```
  <asp:Label ID="lblPanel" runat="server" Text=""></asp:Label> <br />
```

```
  <asp:Button ID="btnUpdate" runat="server" Text="Get Time" OnClick="btnUpdate_Click" />
```

```
  </ContentTemplate>
```

```
</asp:UpdatePanel>
```

```
</div>
```

```
</form>
```

```
</body>
```

```
</html>
```

# UpdatePanel Control

- The above example displays the current time both inside and outside the UpdatePanel control.
- When you click the button, only the time within the UpdatePanel control is refreshed.
- The UpdatePanel hijacks the normal postback and performs a “partial” postback to grab the new content in the background.
- The **ScriptManager** control in above example adds the necessary JavaScript scripts to enable Ajax.
- Anytime you create a page that uses Ajax, regardless of whether you are doing server-side or client-side Ajax, you’ll add a ScriptManager control to the page.

# UpdatePanel Control

## *Major properties of UpdatePanel Control*

- **ChildrenAsTriggers**—Gets or sets a Boolean value that indicates whether child controls should trigger an asynchronous postback automatically.
- **ContentTemplateContainer**—Gets the container for the UpdatePanel control's ContentTemplate. You can add controls to the ContentTemplate programmatically using this property.
- **IsInPartialRendering**—Gets a Boolean value indicating whether the UpdatePanel is rendered in response to an asynchronous postback.
- **Triggers**—Gets a list of controls that trigger the UpdatePanel to perform either an asynchronous or synchronous postback.

# Specifying UpdatePanel Triggers

- By default, an UpdatePanel hijacks any postbacks that any of its child controls performs.
- For example, if a Button control is contained in an UpdatePanel, the UpdatePanel hijacks the button Click event and performs an Ajax call instead of the normal postback.
- You can cause an UpdatePanel to refresh its contents from a control located outside of the UpdatePanel by specifying a trigger.
- For example, the below example contains a Button control outside of an UpdatePanel that causes the UpdatePanel to refresh its content.

# UpdatePanel Trigger

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

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

```
<script runat="server">
```

```
</script>
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head runat="server">
```

```
<title>Trigger UpdatePanel</title>
```

```
</head>
```

```
<body>
```

```
<form id="form1" runat="server">
```

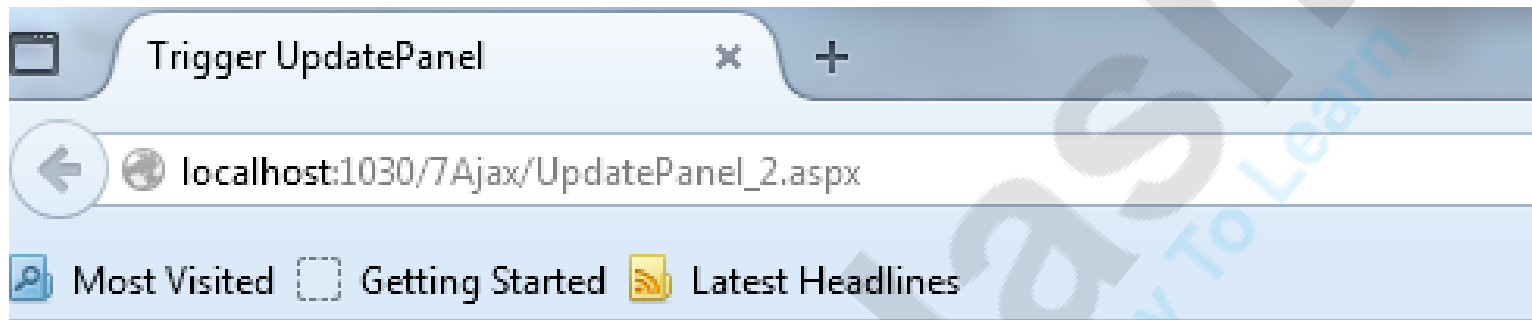
```
<div>
```



# UpdatePanel Trigger

```
<asp:ScriptManager id="sm1" Runat="server" />
  Page Time: <%= DateTime.Now.ToString("T") %>
  <br />
  <asp:Button id="btnUpdate" Text="Update" Runat="server" />
    <br />
  <asp:UpdatePanel id="up1" Runat="server">
    <Triggers>
      <asp:AsyncPostBackTrigger ControlID="btnUpdate"
        EventName="Click" />
    </Triggers>
    <ContentTemplate>
      Update Panel Time: <%= DateTime.Now.ToString("T") %>
    </ContentTemplate>
  </asp:UpdatePanel>
</div> </form> </body> </html>
```

# UpdatePanel Trigger



Page Time: AM 04:23:13

Update Panel Time: AM 04:23:33

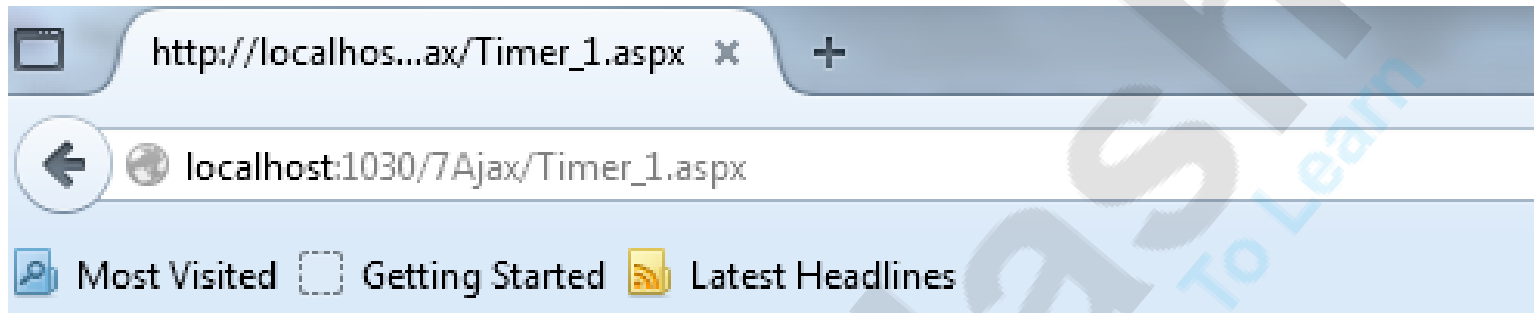
# UpdatePanel Triggers

- If you want, you can prevent the UpdatePanel from refreshing its contents unless you have explicitly created a trigger.
- If you set the UpdatePanel control's ChildrenAsTriggers property to the value false, you must explicitly create a trigger to update the contents of the UpdatePanel.
- The UpdatePanel supports two types of triggers:
  - AsyncPostBackTrigger and PostBackTrigger.
  - The AsyncPostBackTrigger causes an asynchronous (Ajax) postback.
  - The PostBackTrigger causes a normal entire-page postback.

# Timer Control

- *The ASP.NET AJAX Timer control enables you to refresh an UpdatePanel (or the entire page) on a timed basis.*
- The Timer control has one important property:  
***Interval***—The amount of time, in milliseconds, between Tick events. The default value is 60,000 (1 minute).
- The Timer control raises a ***Tick event***, depending on the value of its Interval property.
- *If you don't associate the Timer control with an UpdatePanel, the Timer posts the entire page back to the server performing a normal postback.*
- ***For example, in the below example,*** the page posts the entire page back to the server every 2 seconds.

# Timer Control



The time is AM 04:45:20

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

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

```
<script runat="server">  
</script>
```

```
<html xmlns="http://www.w3.org/1999/xhtml">  
<head runat="server">  
  <title></title>  
</head>
```

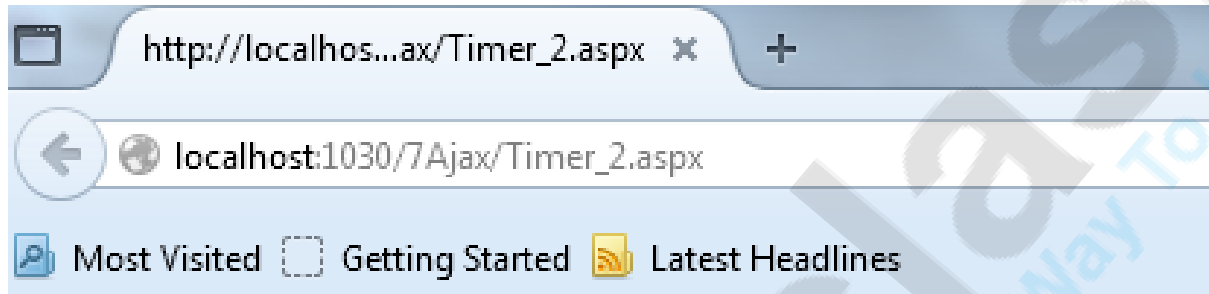
# Timer Control

```
<body>  
  <form id="form1" runat="server">  
    <div>  
      <asp:ScriptManager ID="ScriptManager1" runat="server" />  
      <asp:Timer ID="Timer1" Interval="2000" runat="server" />  
      The time is <%= DateTime.Now.ToString("T") %>  
    </div>  
  </form>  
</body>  
</html>
```

# Timer Control

## **Another Example:**

*Refresh an UpdatePanel control's content on a timed basis.*



The Page time is AM 04:59:54

The Update Panel time is: AM 05:00:13

```
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<script runat="server">
</script>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title></title>
</head>
```

# Timer Control

```
<body>
  <form id="form1" runat="server">
    <div>
      <asp:ScriptManager ID="ScriptManager1" runat="server" />
      <asp:Timer ID="Timer1" Interval="2000" runat="server" />
      The Page time is <%= DateTime.Now.ToString("T") %> <br /> <br />
      <asp:UpdatePanel ID="UpdatePanel1" runat="server">
        <ContentTemplate>
          The Update Panel time is: <%= DateTime.Now.ToString("T") %>
        </ContentTemplate>
        <Triggers>
          <asp:AsyncPostBackTrigger ControlID="Timer1" EventName="tick" />
        </Triggers>
      </asp:UpdatePanel>
    </div>
  </form>
</body>
</html>
```



# UpdateProgress Control

- This control enables you to display a progress indicator while an UpdatePanel is updating its content.
- During a normal postback, the browser displays its progress in downloading new content by spinning an icon or displaying a progress bar.
- During an asynchronous postback, on the other hand, there is no visual indication of progress.
- You can use the UpdateProgress control to give the users some sense that something is happening during an asynchronous postback.

# UpdateProgress Control

- The following example illustrates how to use the UpdateProgress control.
- If you click the button, an animation spins while the asynchronous postback is performed

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

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

```
<script runat="server">
```

```
protected void btnGetTime_Click(object sender, EventArgs e)
```

```
{
```

```
    System.Threading.Thread.Sleep(5000);
```

```
}
```

```
</script>
```

# UpdateProgress Control

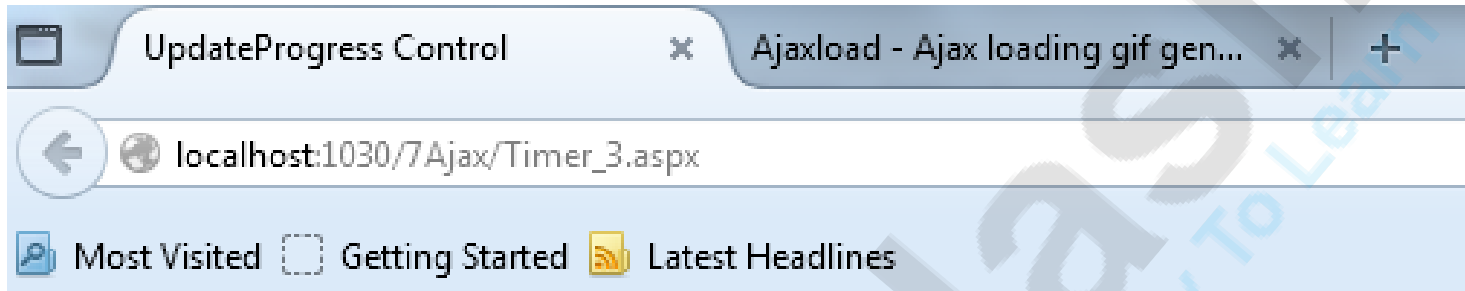
```
<html xmlns="http://www.w3.org/1999/xhtml">  
<head runat="server">  
  <title></title>  
</head>  
<body>  
  <form id="form1" runat="server">  
    <div>  
      <asp:ScriptManager ID="ScriptManager1" runat="server"  
></asp:ScriptManager>
```

# UpdateProgress Control

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server">  
  <ContentTemplate>  
    <asp:Button ID="btnGetTime" runat="server" Text="Get Time"  
      OnClick="btnGetTime_Click" /> <br />  
    The time is:  
    <%= DateTime.Now.ToString("T") %>  
  </ContentTemplate>  
</asp:UpdatePanel>
```

```
<asp:UpdateProgress ID="UpdateProgress1" runat="server"  
  AssociatedUpdatePanelID="UpdatePanel1" >  
  <ProgressTemplate>  
    <asp:Image ID="Loader" ImageUrl="~/Images/ajax-  
      loader.gif" runat="server" />  
  </ProgressTemplate>  
</asp:UpdateProgress>  
</div> </form> </body> </html>
```

# UpdateProgress Control



Get Time

The time is: AM 11:42:27



Several websites enable you to generate fancy animator progress indicator icons. One of such website is:

<http://www.ajaxload.info>

Thanks !!!



Just Another Way To Learn