Perhaps the most visible feature of the ASP.NET AJAX Extensions is the ability to do a partial or incremental page updates without doing a full postback to the server, with no code changes and minimal markup changes. The advantages are extensive – the state of your multimedia (such as Adobe Flash or Windows Media) is unchanged, bandwidth costs are reduced, and the client does not experience the flicker usually associated with a postback.

Introduction

Microsoft's ASP.NET technology brings an object-oriented and event-driven programming model and unites it with the benefits of compiled code. However, its server-side processing model has several drawbacks inherent in the technology:

1. Page updates require a round-trip to the server, which requires a page refresh.
2. Round-trips do not persist any effects generated by Javascript or other client-side technology (such as Adobe Flash)
3. During postback, browsers other than Microsoft Internet Explorer do not support automatically restoring the scroll position. And even in Internet Explorer, there is still a flicker as the page is refreshed.
4. Postbacks may involve a high amount of bandwidth as the \_\_VIEWSTATE form field may grow, especially when dealing with controls such as the GridView control or repeaters.
5. There is no unified model for accessing Web Services through JavaScript or other client-side technology.

Enter Microsoft's ASP.NET AJAX extensions. AJAX, which stands for Asynchronous JavaScript And XML, is an integrated framework for providing incremental page updates via cross-platform JavaScript, composed of server-side code comprising the Microsoft AJAX Framework , and a script component called the Microsoft AJAX Script Library. The ASP.NET AJAX extensions also provide cross-platform support for accessing ASP.NET Web Services via JavaScript.

Partial Page Updates

Perhaps the most visible feature of the ASP.NET AJAX Extensions is the ability to do a partial or incremental page updates without doing a full postback to the server, with no code changes and minimal markup changes. The advantages are extensive - the state of your multimedia (such as Adobe Flash or Windows Media) is unchanged, bandwidth costs are reduced, and the client does not experience the flicker usually associated with a postback.

ScriptManager

Any Ajax control requires the existence of ScriptManager to run Ajaxified Forms . The ScriptManager control is the essential core for the ASP.NET AJAX Extensions. It provides access to the script library (including the extensive client-side script type system), supports partial rendering, and provides extensive support for additional ASP.NET services (such as authentication and profiling, but also other Web Services). The ScriptManager control also provides globalization and localization support for the client scripts.

AJAX UpdatePanel Triggers

When working in the markup editor in Visual Studio, you may notice (from IntelliSense) that there are two child elements of an UpdatePanel control. One of which is the Triggers element, which specifies the controls on the page (or the user control, if you are using one) that will trigger a partial render of the UpdatePanel control in which the element resides.

Triggers for a given UpdatePanel, by default, automatically include any child controls that invoke a postback, including (for example) TextBox controls that have their AutoPostBack property set to true. However, triggers can also be included declaratively using markup; this is done within the <triggers> section of the UpdatePanel control declaration. Although triggers can be accessed via the Triggers collection property, it is recommended that you register any partial render triggers at run-time (for instance, if a control is not available at design time) by using the RegisterAsyncPostBackControl(Control) method of the ScriptManager object for your page, within the Page\_Load event. Remember that Pages are stateless, and so you should re-register these controls every time they are created.

Automatic child trigger inclusion can also be disabled (so that child controls that create postbacks do not automatically trigger partial renders) by setting the ChildrenAsTriggers property to false. This allows you the greatest flexibility in assigning which specific controls may invoke a page render, and is recommended, so that a developer will opt-in to respond to an event, rather than handling any events that may arise.

Note that when UpdatePanel controls are nested, when the UpdateMode is set to Conditional, if the child UpdatePanel is triggered, but the parent is not, then only the child UpdatePanel will refresh. However, if the parent UpdatePanel is refreshed, then the child UpdatePanel will also be refreshed.

## <asp:AsyncPostBackTrigger>

Specifies a control and event that will cause a partial page update for the UpdatePanel that contains this trigger reference.

## <asp:PostBackTrigger>

Specifies a control and event that will cause a full page update (a full page refresh). This tag can be used to force a full refresh when a control would otherwise trigger partial rendering.

HoverMenu

HoverMenu is an ASP.NET AJAX extender that can be attached to any ASP.NET WebControl, and will associate that control with a popup panel do display additional content. When the user moves the mouse cursor over the main control two things happen:

1. The popup panel is displayed at a position specified by the page developer (at the left, right, top, or bottom of the main control)
2. Optionally, a CSS style is applied to the control to specify it as "hot"

**HoverMenu Properties**

**<ajaxToolkit:HoverMenuExtender ID="hme2" runat="Server"**

 **TargetControlID="Panel9"**

 **PopupControlID="PopupMenu"**

 **HoverCssClass="popupHover"**

 **PopupPosition="Left"**

 **OffsetX="0"**

 **OffsetY="0"**

 **PopDelay="50" />**

ConfirmButton

ConfirmButton is a simple extender that catches clicks on a button (or any instance of a type derived from Button) and displays a message to the user. If the "OK" button is clicked, the button or link functions normally. If not, the click is trapped and the button will not perform its default submit behavior; optionally, a client script is executed if the OnClientCancel property is set. This is useful for delete links or anything else that requires confirmation from the user.

**ConfirmButton Properties**

**<ajaxToolkit:ConfirmButtonExtender ID="cbe" runat="server"**

 **TargetControlID="LinkButton1"**

 **ConfirmText="Are you sure you want to click this?"**

 **OnClientCancel="CancelClick" />**

ModalPopup

The ModalPopup extender allows a page to display content to the user in a "modal" manner which prevents the user from interacting with the rest of the page. The modal content can be any hierarchy of controls and is displayed above a background that can have a custom style applied to it.

When displayed, only the modal content can be interacted with; clicking on the rest of the page does nothing. When the user is done interacting with the modal content, a click of an OK/Cancel control dismisses the modal content and optionally runs custom script. The custom script will typically be used to apply whatever changes were made while the modal mode was active. If a postback is required, simply allow the OK/Cancel control to postback and the page to re-render. You can also absolutely position a modal popup by setting the X and Y properties. By default it is centered on the page, however if just X or Y is specified then it is centered vertically or horizontally.

You can provide OnShowing/OnShown/OnHiding/OnHidden animations which are played when the modal content is shown and hidden. For example, you can use these animations to fade-in and fade-out modal content.

**ModalPopup Properties**

**<ajaxToolkit:ModalPopupExtender ID="MPE" runat="server"**

 **TargetControlID="LinkButton1"**

 **PopupControlID="Panel1"**

 **BackgroundCssClass="modalBackground"**

 **DropShadow="true"**

 **OkControlID="OkButton"**

 **OnOkScript="onOk()"**

 **CancelControlID="CancelButton"**

 **PopupDragHandleControlID="Panel3" >**

 **<Animations>**

 **<OnShowing> .. </OnShowing>**

 **<OnShown> .. </OnShown>**

 **<OnHiding> .. </OnHiding>**

 **<OnHidden> .. </OnHidden>**

 **</Animations>**

 **</ajaxToolkit:ModalPopupExtender>**