ASP.NET AJAX

مقدمة

في هذا الدرس سوف أتحدث عن طريقة أستخدام أحدى أدوات الـ Ajax الموجودة في الـ Ajax Extension ، وأيضا سنتعرف بشكل مختصر وبسيط عن ماهي الـ Ajax وطريقة تعاملها في تبادل المعلومات المرسلة من قبل المستخدم الى الـ Server.

لنتحدث في البداية وبشكل مختصر عن ماهي الـ Ajax ومحاولة الوصول للفكرة التي سوف نستخدمها في تعملنا مع UpdatePanel ، بما أننا وصلنا الى مرحلة التعامل مع الـ Ajax أذا توجد لدينا فكرة عن طريقة تفاعل المتصفح (Browser) مع الـ Server، وهي أن يقوم المتصفح بأرسال معلومات بواسطة GET أو POST الى الـServer وفي دوره يقوم بمعالجة الطلب (Request) بالمعلومات المرسلة وأرجعها الى المستخدم بصيغةHTML وتحميل اي مصادر اخرى مربوطة بهذه الصفحة مثلا ملفات CSS الصور الموجودة والخ.

وهذه العملية تستخدم الى يومنا هذا ، لكن لنتوقف قليلا وندقق في العملية التي ذكرنها سابقا عن طريقة تفاعل المتصفح (Browser) مع الخادم (Server) ولنتصور وجود كمية كبيرة من البيانات بهذه الصفحة فعند أرسال عملية الى الخادم وبعد الأنتهاء من معالجة هذه العملية ، سوف يتم أرجاع كمية كبيرة من البيانات للمتصفح وبالأضافة إلى البيانات التي تمت معالجتها، ولنفرض أن العملية التي نريد معالجتها من الخادم كانت ذات حجم صغير؟ أذا نستطيع أستنتاج أن حتى لو كانت العملية المراد معالجتها في الخادم ذات حجم صغير ، لا بد من أرسال الصفحة بما فيها للخادم وأرجاعها مره أخرى للمتصفح.

وعلى هذا الأساس وعند ظهور تقنية الـ Ajax أتت لتساعد على عمل اتصال مع الخادم من دون الأضطرار إلى عمل العملية التي تم ذكرها سابقا، إذ هي تقوم فقط بأرسال الجزء المراد تحديثة للخادم ومعالجتة ومن ثم أرجاع هذا الجزء الى الصفحة وعمل تحديث للمكونات القديمة بما هو جديد.

أذا لنتعرف على أداة UpdatePanel و ماهي الخصائص المرتبطة بهذه الأداة، لكن قبل البدء لنتفق على مصطلح نستطيع به وصف العملية السابقة بطريقة تفاعل المتصفح مع الخادم بهذا المصطلح هو PostBack.

UpdatePanel Control

بعد العملية اتي تم شرحها سابقا في طريقة التفاعل بين المتصفح والخادم، تأتي هذه الأداة لتكون حل لتجنب PostBack الكامل للصفحة وذالك عن طريقة عمل تحديث فقط للجزء الموجود في هذه الأداة.

ولا بد وعند أستخدام الأدوات الموجودة في قائمة Ajax Extension أن يتم وضع أداة ScriptManager ، وذالك لأن هذه الأداة تكون بمثابة الجسر بين Client page و الـServer.

ومن بعض الخصائص المرتبطة بهذه الأداة كالتالي:

RenderMode

يمكن ضبط هذه الخاصية على block أو Inline لتعريف شكل الكود الذي يتم أرجاعة بوساطة الـ Updatepanel ، ففي الخاصية الأولى يتم أرجاع الكود على شكل و الخاصية الثانية يكون على شكل .

UpdateMode

يمكن ضبط هذه الخاصية على Always أو Conditional ، حيث تستخدم هذه الخاصية لتحديد الطريقة التي سوف يتم فيها عمل Update للـ Updatepanel (دائما ما تكون هذه الخاصية Always)، وفي المثال اللاحق سوف نقف عند هذه النقطة لنوضح الفرق بين الخاصيتين.

ContentTemplate

وتعتبر هذه الخاصية هي أهم الخواص عند أستخدام الـ Updatepanel حيث أن هذه الخاصة يكون بداخلها المحتويات أو بمسمى آخر الأدوات التي نريد أن نطبق عليها التحديث من دون عمل PostBack كامل للصفحة أو نستطيع تعريفة بـ AsyncPostBack، ويكون تعريفها بهذا الشكل:

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
        <%--The content of UpdatePanel--%>
    </ContentTemplate>
</asp:UpdatePanel>
```

Triggers

لنفرض أنه يوجد لدينا Updatepanel ونريد تحديث محتوياتها بواسطة Button خارج الـ UpdatePanel نفسها فعندها تأتي خاصية الـ Trigger لتساعدنا على تطبيق هذا العمل، وتحتوي خاصية الـ Trigger على PostBackTrigger و AsyncPostBacktrigger وهاتان الخاصيتان تحدد نوع الـ PostBack المراد تطبيقة عند الضغط على الـ Button.

عند أستخدام الخاصية الأولى الا وهي PostBackTrigger سوف يكون التحديث مثل الشكل الطبيعي أي سيكون عملية أرسال الصفحة الى الخادم والخ... ، بينما الخاصية الثانية تقوم بتحديث فقط ما يوجد بداخل الـ Updatepanel ، ويكون تعريف خاصية الـ Trigger بهذا الشكل:

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
        <%--The content of UpdatePanel--%>
    </ContentTemplate>
    <Triggers>
        <asp:AsyncPostBackTrigger ControlID="" EventName="" />
        <asp:PostBackTrigger ControlID="" />
    </Triggers>
</asp:UpdatePanel>
```

سنلاحظ اننا سوف نقوم بتعريف الـID الخاص بالأداة التي نريد منها أن تفعل عملية التحديث عندما يتم الضغط عليها على سبيل المثال في حالة Button ، والفرق الوحيد بين نوعي الـ Trigger هم أن الـ AsyncPostBack يحتاج الى تعريف الحدث (Event) الخاص بهذه الأداة وفي حالتنا و أن افترضنا أن الأداة هي Button أذا سوف يكون أسم الحدث Click.

ملاحظة: سوف نستخدم نوع واحد من الـ Triggers ، وتم وضع كلا النوعين لغرض التوضيح فقط.

لنعد للخلف قليلا ونتحدث عن الفرق في خيارات خاصية UpdateMode وأبسط طريقة لتوضيح هذا الفرق هو أخذ مثال، أذا لنبدا:

ملاحظة: هذا المثال سوف يتم التعديل عليه تقدما مع الحالات الجديدة.

كالعادة أفتح برنامج Visual Studio ومن قائمة File أختر New web Site وبعدها نختار ASP.NET Empty Web Site ، قم بتستمية بما تريد.

قم بأضافة صفحة جديدة الى المشروع ولنجعل أسمها Default.aspx، وبداخل هذه الصفحة قم بأضافة الأداة المهمة لأدارة عمليات الـ Ajax وهي ScriptManager. وفقط لتقصير الـ ID الخاص به لنقم بتغيرة إلى sm.

بعدها لنقم بإضافة عدد ٢ من أداة UpdatePanel الى الصفحة ووضع بداخل الـ ContentTemplate إلى الاداتين Label و Button. ليكون الشكل النهائي كالتالي:

```
<form id="form1" runat="server">
<asp:ScriptManager ID="sm" runat="server">
</asp:ScriptManager>
<div>
    <asp:UpdatePanel ID="UpdatePanel1" runat="server">
        <ContentTemplate>
            <asp:Label Text="" ID="lbl1" runat="server" /><br />
            <asp:Button ID="Button1" runat="server" Text="Refresh first label" />
        </ContentTemplate>
    </asp:UpdatePanel>
    <br />
    <asp:UpdatePanel ID="UpdatePanel2" runat="server">
        <ContentTemplate>
            <asp:Label Text="" ID="lbl2" runat="server" /><br />
            <asp:Button ID="Button2" runat="server" Text="Refresh second label" />
        </ContentTemplate>
    </asp:UpdatePanel>
</div>
</form>
```

الخطوة الثانية أفتح الـ Code Behind للصفحة وفي حدث الـ Page_Load أكتب كالتالي:

```
protected void Page_Load(object sender, EventArgs e)
{
    lbl1.Text = DateTime.Now.ToLongTimeString();
    lbl2.Text = DateTime.Now.ToLongTimeString();
}
```

أفتح الصفحة على المتصفح ، سنلاحظ وجود التاريخ مع الوقت وعند الضغط على أي زر من الأزرار سوف يتم تحديث الوقت، لكن لو نعد إلى الخلف نحن وضعنا عدد ٢ من أداة UpdatePanel أذا لماذا يتم تحدث الاثنتين معنا؟

وياتي سبب هذه المشكلة لان أداة الـ UpdatePanel تتعامل بشكل AsyncPostBack وبمساعدة الـ ScriptManager ، فعندما نعمل تحديث الى واحدة من الـ UpdatePanel سوف نرسل AsyncPostBack إلى الخادم وهذا ما يجعل الأداة الأخرى تعمل تحديث لمكوناتها.

أذا لتجنب وقوع هذه المشكلة هنا يأتي دور الـ UpdateMode حيث ذكرنا سابقا أنه دائما ما يكون Always ، لنعمل تعديل بسيط الى أداتي UpdatePanel ونضيف لهم كالتالي:

```
UpdateMode="Conditional"
```

ونعاود فتح الصفحة مرة آخرى ونلاحظ الفرق.

النتيجة سوف تكون تحديث كل أداة لوحدها من دون التدخل في تحديث الأخرى، وبهذا قد حللنا هذه المشكلة أذا لنتقدم قليلا ونعمل تعديلات بسيطة في ملف Code Behind كالتالي:

```
protected void Page_Load(object sender, EventArgs e)
{
    lbl1.Text = DateTime.Now.ToLongTimeString();
    System.Threading.Thread.Sleep(2000);
    lbl2.Text = DateTime.Now.ToLongTimeString();
}
```

ملاحظة: الكود الذي تمت أضافة يعمل تأخير مدة ٢ ثانية ، فأرجوا منك عدم أستخدام هذه الطريقة عند رفع الموقع على خادم، والهدف من أستخدام هذا الكود لأن الصفحة تعمل على خادم محلي (localhost) فسوف يكون تبادل البيانات بشكل سريع ولهذا السبب عملت هذا التأخير لنرى نتائج ما نريد فعله.

أفتح الصفحة على المتصفح سوف نلاحظ تأخير في بداية فتح الصفحة وبعدها لنقم بالضغط على الزر الثاني سنلاحظ أيضا تأخير في عملية تحديث البيانات. لربما يخطر في بالك أنه تم كتابة كود التأخير قبل تحديث الـ Label الثاني وأنه لن يوثر على الـ Label الأول ، أذا حاول أن تقوم بالضغط على الزر الأول وستلاحظ أنه أيضا تم التأخير في عملية تحديثة.

بالطبع هذه العملية ليست مرغوب فيها لو كانت الصفحة تحتوي على أدوات وكل وحده من هذه الأدوات تحتاج إلى وقت للتحديث وتصور أنها تؤثر على باقي الأدوات، لذالك نحن بحاجة إلى تحديد ماهي الأداة الموجودة في الـ UpdatePanel التي تساعد على عمل AsyncPostBack لها، وهنا يأتي دور الـ ScriptManager لمساعدتنا بوجود خصائص(Properties) تساعدنا على تحديد أذا كانت هذه الأداة تستخدم الـ AsyncPostBack. وفي مثالنا السابق نعلم أن الأداة التي تعمل على تحديث الـ UpdatePanel الثاني هي أداة Button2 ومنها لنقم بتعديل بسيط على الكود ونتعرف على الخصائص التي يقدمها لنا الـ ScriptManager كالتالي:

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!sm.IsInAsyncPostBack || sm.AsyncPostBackSourceElementID == "Button1")
    {
        lbl1.Text = DateTime.Now.ToLongTimeString();
    }
    if (!sm.IsInAsyncPostBack || sm.AsyncPostBackSourceElementID == "Button2")
    {
        System.Threading.Thread.Sleep(2000);
        lbl2.Text = DateTime.Now.ToLongTimeString();
    }
}
```

في الجزء الأول أستخدمت (!) للتحقق أذا أن ScriptManager ليس في حالة الـ AsyncPostBack وطبعا هذا التحقق صحيح لأن الكود يتم تنفيذة في حدث Page_Load فلن يكون في وضعية أرسال AsyncPostBack أو (||) تم تحديد الأداة التي سوف نستخدمها لتفعيل عملية AsyncPostBack ونطبق العمليات بعد التحقق، أفتح الصفحة من جديد ولأحظ الفرق.

سنلاحظ بأننا أستطعنا تحديد أي الأدوات وأي من الـ UpdatePanel سوف نقوم بعمل تحديث لها.

لننتقل الى خاصية الـ Triggers الموجودة في الـ UpdatePanel وكما تم ذكره سابقا أنها تساعد على عمل تحديث الى الـ UpdatePanel بواسطة أداة تكون خارجها، لنعمل تعديل على المثال السابق ونلاحظ كيفية أستخدام هذه الخاصية كالتالي:

```
<form id="form1" runat="server">
<asp:ScriptManager ID="sm" runat="server">
</asp:ScriptManager>
<div>
    <asp:UpdatePanel ID="UpdatePanel1" UpdateMode="Conditional" runat="server">
        <ContentTemplate>
            <asp:Label Text="" ID="lbl1" runat="server" /><br />
            <asp:Button ID="Button1" runat="server" Text="Refresh first label" />
        </ContentTemplate>
        <Triggers>
            <asp:AsyncPostBackTrigger ControlID="Button2" EventName="Click" />
        </Triggers>
    </asp:UpdatePanel>
    <br />
    <asp:Button ID="Button2" runat="server" Text="Refresh first label"
        onclick="Button2_Click" />
</div>
</form>
```

وأيضا أعمل التعديلات التالية لملف الـ Code Behind كالتالي:

```
    protected void Page_Load(object sender, EventArgs e)
    {
        lbl1.Text = DateTime.Now.ToLongTimeString();
    }
    protected void Button2_Click(object sender, EventArgs e)
    {
        lbl1.Text = DateTime.Now.ToLongTimeString();
    }
```

ستلاحظ النتيجة هي حتى لو قمت بالضغط على الزر الموجود داخل الـ UpdatePanel أو الموجود خارجها سوف يتم تحديث الـ Label.

قبل ختام هذا الدرس سوف نتحدث عن حالة تداخل (nested) أو بمسمى آخر وضع UpdatePanel بداخل UpdatePanel و مالذي نحتاجه للتعامل مع هذه الوضعية.

بكل بساطة كما تطرقنا سابقا أنه عند التعامل مع أكثر من UpdatePanel لابد من وضع خاصية UpdateMode إلى Conditional لتجنب عمل (refresh) إلى كل الـ UpdatePanels الموجودة في الصفحة، لكن سنلاحظ أن أي UpdatePanel تحوي على واحدة اخرى سوف يتم تحديثها عند عمل تحديث لهذه الـ UpdatePanel . أنظر الى المثال التالي:

```
    <form id="form1" runat="server">
    <asp:ScriptManager ID="sm" runat="server">
    </asp:ScriptManager>
    <div>
    <fieldset>
        <legend>The panel number 1</legend>
            <asp:UpdatePanel ID="UpdatePanel1" UpdateMode="Conditional"
runat="server">
                <ContentTemplate>
                    <asp:Label Text="" ID="lbl1" runat="server" /><br />
                    <asp:Button ID="Button1" runat="server" Text="Refresh first label"
/>
                    <br />
                    <fieldset>
                        <legend>The panel number 2</legend>
                            <asp:UpdatePanel ID="UpdatePanel2"
UpdateMode="Conditional" runat="server">
                                <ContentTemplate>
                                    <asp:Label ID="Lbl2"  runat="server" /><br />
                                    <asp:Button ID="Button2" runat="server"
Text="Refresh second label" />
                                    <br />
                                    <fieldset>
                                        <legend>The panel number 3</legend>
                                        <asp:UpdatePanel ID="UpdatePanel3"
UpdateMode="Conditional" runat="server">
                                            <ContentTemplate>
                                                    <asp:Label ID="Lbl3"
runat="server" /><br />
                                                    <asp:Button ID="Button3"
runat="server" Text="Refresh third label" />
                                            </ContentTemplate>
                                        </asp:UpdatePanel>
                                    </fieldset>
                                </ContentTemplate>
                            </asp:UpdatePanel>
                    </fieldset>
```

```
                </ContentTemplate>
            </asp:UpdatePanel>
    </fieldset>
</div>
</form>
```

ما كتب في الكود اعلى هو وجود عدد ٣ من UpdatePanels كل وحده منهم تحوي على الأخرى ، ولو نفتح الصفحة على المتصفح ونرى النتيجة سنلاحظ أنه عندما يتم على الزر الموجود في الأعلى يتم تحديث ما تحته من Labels ولو تم الضغط على آخر زر فسوف تكون النتيجة العكس. ولا ننسى أضافة هذا الجزء البسيط للـ Code Behind.

```
Lbl3.Text = DateTime.Now.ToLongTimeString();
```

في الختام تستطيع أستخدام أداة UpdatePanel بالعدد الذي تريدة في الصفحة ولكن يفضل أن يتم أستخدامها بحسب مانحتاجة، وكما يوجد Sys.WebForms.PageRequestManager Class وهو المساعد على أدارة الـ AsyncPostBack الذي يصدر بواسطة الـ UpdatePanel ولكن بأستخدام الـ Client Script.