

Page Request Manager

مقدمة

لقد تحدثنا في أحد الدروس السابقة عن طريقة تجنب التحديث الكامل وعرفناه بـ `AsyncPostBack`، وأستطعنا عمل هذا بواسطة أداة الـ `UpdatePanel`، وأيضا واصلنا الحديث عند أداة أخرى هي أداة الـ `UpdateProgress` حيث أنها تكون مساعده للأداة الأولى التي تم ذكرها. وكان عمل الأداة الثانيه هو أظهار رسالة تنبيه للمستخدم تفيد به بأن الطلب الذي يرسله للخادم (وعلى سبيل المثال ضغط زر لتحديث الوقت كما في المثال الذي تم شرحه في الدرس) يتم معالجته وأظهار كلمة تخبره بالانتظار. أذاً لنتصور بأن هناك إمكانية بالتحدثكم بهذا الطلب والقصد هنا هو الـ `AsyncPostBack` عن طريقة استخدام الـ `JavaScript` وأيضا بأستطاعتنا جعل المستخدم يلغي إرسال هذا الطلب بواسطته و أيضاً منع المستخدم من إرسال الطلب الجديد الأ بعد معالجة الطلب السابق، فهنا يأتي دور ما سوف نشرحه في هذا الدرس وهو استخدام الـ `PageRequestManager` الموجود في `System.WebForms` Namespace.

PageRequestManager

Manages partial-page updates of server `UpdatePanel` controls in the browser, and defines properties, events, and methods that can be used to customize a Web page by using client script.

أذاً من التعريف الذي في الأعلى نستطيع أن نستنتج بأن الـ `PageRequestManager` يساعد على إدارة التحديث الجزئي للصفحة أو مانعرفه بـ `AsyncPostBack` وبعدها يمكننا تعريف خصائص، أحداث و دوال مساعده وذلك بأستخدام الـ `Client Script`.

ملاحظة: الـ `PageRequestManager` يكون موجود على شكل `Class` يتم تحميله إلى الصفحة بعدما نضع أداة الـ `Script Manager` على شكل ملفات `JavaScript` وهذا ما يعرف بـ `Microsoft Ajax Library` ولتستطيع رؤية هذه الملفات افتح الصفحة وبعدها `View Page Source`.

ملاحظة: أنا أفضل استخدام الأداة التي تأتي كأضافة إلى متصفح الـ `Firefox` وهي `FireBug` أو أي أداة مشابهة لعملها في المتصفحات الأخرى وذلك لرؤية الملفات التي تم ذكرها سابقاً بشكل أفضل.

ومن الأشياء التي نستطيع أن نفعلها بأستخدام الـ `PageRequestManager` هي كالتالي:

Name	Description
<code>getInstance Method</code>	Returns the instance of the <code>PageRequestManager</code> class for the page. Note: You do not create an instance of the <code>PageRequestManager</code> directly. When partial-page rendering is enabled, an instance of the <code>PageRequestManager</code> class is automatically available. You can access it through the <code>getInstance</code> method.
<code>isInAsyncPostBack Property</code>	Returns a value that indicates whether the <code>PageRequestManager</code> object is processing a postback.
<code>abortPostBack Method</code>	Stops all updates that would occur as a result of an asynchronous postback.

قبل البدء في تطبيق المثال لابد من فهم العملية التي سوف نطبقها في المثال، تبدأ العملية عندما يتم عمل تحديث بواسطة الـ `UpdatePanel` حيث سوف يتم إرسال `AsyncPostBack` إلى الخادم وفي هذه اللحظة سوف يكون الـ `PageRequestManager` موجود بسبب تحميل أداة الـ `Script Manager` لملفات الـ `Microsoft Ajax Library` والموجود بداخلها `Class` الـ `PageRequestManager` فيأتي دورنا بعد هذه الخطوات و نقوم بأخذ `Instance` من هذا الـ `PageRequestManager` وتطبيق عليها عمليتنا التي نريدها.

وأيضا يوجد هناك أحداث يتم تعريفها بواسطة الـ `PageRequestManager` والتي نستطيع أن نستخدمها لنحدد أي مرحلة يتم تطبيق الكود الذي نريده، على سبيل المثال حدث `initializeRequest` حيث يكون هذا الحدث قبل بدأ عملية معالجة الـ `AsyncPostBack` ومن الأشياء التي نستطيع أن نفعلها في هذا الحدث على سبيل المثال هو إيقاف عملية التحديث وهكذا، والجدول يوضح الأحداث كالتالي:

PageRequestManager event	Description
initializeRequest	Raised before processing of the asynchronous request starts. You can use this event to cancel a postback.
beginRequest	Raised before processing of an asynchronous postback starts and the postback is sent to the server. You can use this event to set request headers or to begin an animation that indicates that the page is processing.
pageLoading	Raised before processing of an asynchronous postback starts and the postback is sent to the server. You can use this event to set request headers or to begin an animation that indicates that the page is processing.
pageLoaded	Raised after all content on the page is refreshed, as a result of either a synchronous or an asynchronous postback. You can use this event to provide a custom transition effect for updated content.
endRequest	Raised after an asynchronous postback is finished and control has been returned to the browser. You can use this event to provide a notification to users or to log errors.

والطريقة لتعريف الأحداث السابقة هي كالتالي:

```
Sys.WebForms.PageRequestManager.getInstance().add_initializeRequest(FunctionName);
```

والدالة التي سوف نكتب بها الكود الخاص بهذا الحدث كالتالي:

```
function FunctionName(sender, args) {
```

```
}
```

وهكذا مع بقية الأحداث يتم أولاً تعريفه وبعدها كتابة دالة تحتوي على العمليات التي يتم تطبيقها في هذا الحدث. أذاً لنبدأ بتطبيق مثال وتطبيق ما تم شرحه. في البداية نفتح برنامج الـ Visual Studio وبعدها نختار File وبعدها New Web Site نختار ASP.NET Empty Web Site

ملاحظة: قبل البدء في المثال سوف نقوم بعمل مثال الناتج منه سوف يكون مشابه لأداة الـ UpdateProgress ولكن باستخدام الـ JavaScript وبعدها سوف نطبق مباشرة على أداة الـ UpdateProgress.

في البداية سوف نقوم بإضافة الـ Script Manager للصفحة وبعدها سوف نضع أداة الـ UpdatePanel ونعمل التغييرات كالتالي:

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
  <ContentTemplate>
    <asp:Label ID="lbl" runat="server"></asp:Label>
    <br />
    <asp:Button ID="Button1" runat="server" Text="Refresh" onclick="Button1_Click" />
  </ContentTemplate>
</asp:UpdatePanel>
<br />
<div id="Message" style="color:Red;"></div>
```

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        lbl.Text = DateTime.Now.ToLongTimeString();
    }
}

protected void Button1_Click(object sender, EventArgs e)
{
    System.Threading.Thread.Sleep(3000);
    lbl.Text = DateTime.Now.ToLongTimeString();
}
```

في الخطوة التالية نقوم بإنشاء ملف JavaScript ونُعرف هذا الملف في أداة الـ ScriptManager كالتالي:

```
<asp:ScriptManager ID="sm" runat="server">
    <Scripts>
        <asp:ScriptReference Path="~/PageRequest.js" />
    </Scripts>
</asp:ScriptManager>
```

ونكتب الكود التالي في ملف الـ JavaScript

```
Sys.WebForms.PageRequestManager.getInstance().add_initializeRequest(CheckStatus);
Sys.WebForms.PageRequestManager.getInstance().add_endRequest(endRequest);
function CheckStatus(sender, args) {
    var prm = Sys.WebForms.PageRequestManager.getInstance();
    if (!prm.get_isInAsyncPostBack() & args.get_postBackElement().id == "Button1") {
        ShowProgressMessage('Please Wait...');
    }
}
function endRequest(sender, args) {
    ShowProgressMessage("");
}
function ShowProgressMessage(msg) {
    $get('Message').innerHTML = msg;
}
```

في بداية الكود تم تعريف الأحداث التي نريد أن نعمل عليها وهي initializeRequest و endRequest وأيضاً تم تعريف دالة أخرى لتحديث محتويات الـ <div> الذي تم تعريفه أسفل الـ UpdatePanel، و الحدث initializeRequest ينفذ الكود الموجود في دالة الـ CheckStatus

حيث أن في هذه الدالة بدأنا بأخذ Instance من الـ PageRequestManager وبعدها التحقق من هذا الـ Instance والذي أسمه prm ليس في مرحلة AsyncPostBack وأن الأداة التي تعمل هذا الـ AsyncPostBack هي Button1 كما في مثالنا، أذا أظهر رسالة تفيد المستخدم بأن طلبية يتم تحديثه. وفي حدث الـ endRequest تم إزالة الرسالة الموجودة في الـ <div> بعد ما تم أظهارها للمستخدم في هذه العملية وذلك باعتبار بأن الطلب أنتهى. أفتح الصفحة في المتصفح ولاحظ النتيجة.

لننتقل إلى الخطوة التالية ونعمل تحديث الى الأكواد لكي نجعل المستخدم يقوم بإلغاء هذا الطلب قبل معالجته، في البداية سنحتاج إلى أضافة زر جديد وظيفته عند الضغط عليه يلغي عملية التحديث والشكل النهائي لهذه الأضافة كالتالي:

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
        <asp:Label ID="lbl" runat="server"></asp:Label>
        <br />
        <asp:Button ID="Button1" runat="server" Text="Refresh" onclick="Button1_Click" />
        <asp:Button ID="Button2" runat="server" Text="Cancel" />
    </ContentTemplate>
</asp:UpdatePanel>
<br />
<div id="Message" style="color:Red;"></div>
```

بعدها نقوم بتحديث الكود لدالة الـ CheckStatus كالتالي:

```
function CheckStatus(sender, args) {
    var prm = Sys.WebForms.PageRequestManager.getInstance();
    if (prm.get_isInAsyncPostBack() & args.get_postBackElement().id == "Button2") {
        prm.abortPostBack();
    }
    else if (!prm.get_isInAsyncPostBack() & args.get_postBackElement().id == "Button1") {
        ShowProgressMessage('Please Wait...');
    }
}
```

أفتح الصفحة ولاحظ النتيجة، بعدما تضغط على زر التحديث مباشرة أضغط على زر الإلغاء وستلاحظ أنه لن يتم تحديث الوقت.

لننتقل إلى آخر خطوة وهي منع المستخدم من إرسال طلب جديد إلا عند الانتهاء من الطالب السابق، وفي هذه المرحلة لن نضيف أي شيء جديد بل سوف نعمل تحديث لدالة الـ CheckStatus كالتالي:

```
function CheckStatus(sender, args) {
    var prm = Sys.WebForms.PageRequestManager.getInstance();
    if (prm.get_isInAsyncPostBack() & args.get_postBackElement().id == "Button2") {
        prm.abortPostBack();
    }
    else if (prm.get_isInAsyncPostBack() & args.get_postBackElement().id == "Button1") {
        args.set_cancel(true);
        ShowProgressMessage('Still working on previous request!');
    }
    else if (!prm.get_isInAsyncPostBack() & args.get_postBackElement().id == "Button1") {
        ShowProgressMessage('Please Wait...');
    }
}
```

الكود المضاف في هذه الحالة هو للتحقق إذا كان الـ Instance في حالة AsyncPostBack والزر الذي تم الضغط عليه هو مثل الزر الذي أنشأ هذا الـ AsyncPostBack إذا الناتج هو أظهار رسالة تفيد المستخدم بأنه الطلب الأول في مرحلة المعالجة، ولمشاهدة الناتج أفتح الصفحة وأضغط على الزر الذي يعمل تحديث وفي نفس الوقت أضغط عليه مرة أخرى حيث ستلاحظ ظهور رسالة آخرة تفيد المستخدم بالانتظار للانتهاء من تنفيذ الطلب الأول.

من الملاحظ عند تطبيق المثال السابق لقد تم عمل شيء مثل لأداة الـ UpdateProgress وهو أظهار الرسائل تفيد بالانتظار، أذا يمكننا بدل كتابة الكود بهذا الحجم بأن نعمل له اختصار باستخدام أداة الـ UpdateProgress. أذا لنقم بهذه التعديلات وصولاً للناتج.

في البداية لنقم بإنشاء ملف JavaScript جديد وربطه مع الـ Script Manager كالتالي:

```
<asp:ScriptManager ID="sm" runat="server">
    <Scripts>
        <asp:ScriptReference Path="~/PageRequestUpdateProgress.js" />
    </Scripts>
</asp:ScriptManager>
```

وبعدها نعمل تحديث للصفحة بإضافة أداة الـ UpdateProgress أسفل أداة الـ UpdatePanel كالتالي:

```
<asp:UpdateProgress ID="UpdateProgress1" AssociatedUpdatePanelID="UpdatePanel1" runat="server">
    <ProgressTemplate>
        Please Wait...
    <br />
    <div id="Message" style="color:Red;"></div>
    </ProgressTemplate>
</asp:UpdateProgress>
```

وفي ملف الـ JavaScript نكتب الأكواد كالتالي:

```
Sys.WebForms.PageRequestManager.getInstance().add_initializeRequest(CheckStatus);
function CheckStatus(sender, args) {
    var prm = Sys.WebForms.PageRequestManager.getInstance();
    if (prm.get_isInAsyncPostBack() & args.get_postBackElement().id == "Button2") {
        prm.abortPostBack();
    }
    if (prm.get_isInAsyncPostBack()) {
        args.set_cancel(true);
        ShowProgressMessage('Still working on previous request!');
    }
    else {
        ShowProgressMessage("");
    }
}
function ShowProgressMessage(msg) {
    $get('Message').innerHTML = msg;
}
```

وبعدها نفتح الصفحة في المتصفح ونلاحظ النتيجة. حيث أن ما فعلناه في هذا الكود هو إمكانية المستخدم من إيقاف الطلب كما فعلنا سابقاً وأخبار المستخدم بأن الطلب في المعالجة وعدم السماح له بتنفيذ طلب آخر إلا بعد الانتهاء من الطلب الأول.

وفي الختام حاولت تطبيق الفكرة بطريقتين ورأينا أن الطريقة الثانية تقلل من الجهد وذلك بمساعدة أداة الـ UpdateProgress.