Comprendre le rafraichissement partiel d'une page avec ASP.NET AJAX.

par Scott Cate (Traduction par Rémy Mainil)

La fonctionnalité la plus spectaculaire des extensions ASP.NET AJAX est probablement la possibilité d'effectuer un rafraichissement partiel ou incrémental d'une page sans effectuer un postback complet vers le serveur, sans changement de code et avec peu de changements de balises. Les avantages sont nombreux : l'état de vos objets multimédias (tels que Adobe Flash ou Windows Media) est inchangé, les coûts de bande passante sont réduits et le client ne subit pas le clignotement habituellement associé au postback.

Traduction

Cet article est la traduction la plus fidèle possible de l'article original : **Understanding Partial Page Updates with ASP.NET AJAX**

1 - Introduction

La technologie ASP.NET de Microsoft apporte un modèle de programmation orienté-objet et axé sur les événements, couplé au bénéfice du code compilé. Cependant, son modèle de traitement côté serveur souffre de quelques inconvénients inhérents à la technologie :

- Actualiser une page nécessite un aller-retour vers le serveur, ce qui nécessite un rafraichissement de la page
- Les aller-retour serveurs ne maintiennent pas les modifications apportées par le JavaScript ou toute autre technologie "côté-client" (telle que Adobe Flash) et, même dans Internet Explorer, il y a toujours un clignotement lorsque la page est actualisée.
- Les postbacks entraînent une consommation gourmande de la bande passante au fur et à mesure que le __VIEWSTATE grandit, spécialement lorsqu'on travaille avec des contrôles tels que la GridView ou les Repeaters.
- Il n'y a pas de modèle unifié pour utiliser des Web Services via JavaScript ou une autre technologie "côté-client"

Voyons maintenant du côté de Microsoft ASP.NET AJAX. AJAX, qui signifie **A**synchronous **J**avaScript **A**nd **X**ML, est un Framework intégré permettant des actualisations progressives des pages via du JavaScript multiplateformes, composé de code côté serveur comprenant le Framework AJAX, ainsi que d'un script qui constitue la librairie Microsoft AJAX Script. Les extensions ASP.NET AJAX fournissent également une méthode, multiplateformes, d'accès à des services Web ASP.NET via JavaScript.
Cet article explique la fonctionnalité de rafraichissement partiel des extensions ASP.NET AJAX, ce qui inclus le composant ScriptManager, les contrôles UpdatePanel et UpdateProgress, et explore différents scénarios dans lesquels ils devraient, ou non, être utilisés.
Cet article (NdT : au moment de sa rédaction) est basé sur la version Beta 2 de Visual Studio 2008 et sur le Framework .NET 3.5, qui intègre les extensions ASP.NET AJAX dans les classes de base (là où elles étaient précédemment disponibles sous forme d'add-on pour ASP.NET 2.0). Cet article présume également que vous utilisez Visual Studio 2008 et non Visual Web Developer Express Edition; certains modèles de projet qui sont référencés ne sont éventuellement pas disponibles pour les utilisateurs de Visual Web Developer Express.

2 - Rafraichissement partiel d'une page

La fonctionnalité la plus spectaculaire des extensions ASP.NET AJAX est probablement la possibilité d'effectuer un rafraichissement partiel ou incrémental d'une page sans effectuer un postback complet vers le serveur, sans changement de code et avec peu de changements de balises. Les avantages sont nombreux : l'état de vos objets multimédias (tels que Adobe Flash ou Windows Media) est inchangé, les coûts de bande passante sont réduits et le client ne subit pas le clignotement habituellement associé au postback.
Peu de changements sont nécessaires dans votre projet pour y intégrer le rafraichissement partiel des pages, cette technologie s'intégrant dans ASP.NET.

3 - Mode d'emploi : Intégrer le rafraichissement partiel dans un projet existant

- 1) Dans Microsoft Visual Studio 2008, créez un nouveau Site Web ASP.NET en allant dans le menu Fichier -> Nouveau -> Site Web et en choisissant Site Web ASP.NET. Vous pouvez le nommer comme bon vous semble et vous pouvez l'installer soit dans le système de fichiers soit dans Internet Information Services (IIS).
- 2) Vous vous retrouverez devant la classique page par défaut avec les balises ASP.NET standard (un form avec runat="server" et la directive @Page). Déposez-y un label appelé **Label1** et un bouton appelé **Button1** entre les balises du formulaire. Vous pouvez attribuer ce que vous voulez à leur propriété Text.
- 3) Dans la vue Design, double-cliquez sur le contrôle Button1 pour générer le gestionnaire d'événement côté code. Dans ce gestionnaire d'événement, changez la propriété Text de Label1 en "Vous avez cliqué sur le bouton !".

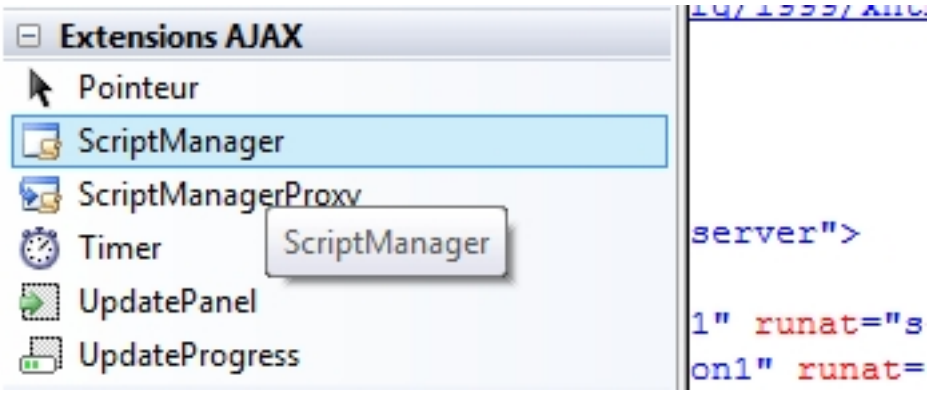
Listing 1: Contenu de default.aspx avant que le rafraichissement partiel ne soit mis en place

```
<%@ Page Language="C#" AutoEventWireup="true"  CodeFile="Default.aspx.cs" Inherits="_Default" %>

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

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
        <asp:Button ID="Button1" runat="server" Text="Button" onclick="Button1_Click" />
    </div>
    </form>
</body>
</html>
```

Listing 2: Codebehind (allégé) de default.aspx.cs

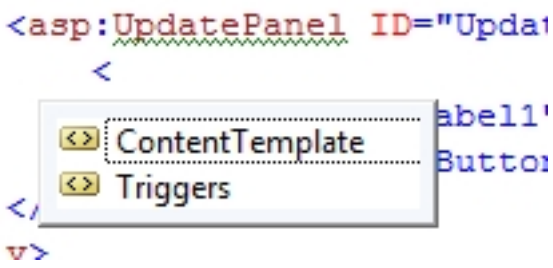
```
public partial class _Default : System.Web.UI.Page
{
    protected void Button1_Click(object sender, EventArgs e)
    {
        Label1.Text = "Vous avez cliquÃ© sur le bouton !";
    }
}
```

- 4) Appuyez sur F5 pour lancer le site web. Visual Studio vous proposera de modifier le fichier web.config pour activer le débogage; faites-le. Quand vous cliquez sur le bouton, un rafraichissement de la page survient pour changer le contenu du label et il y a un léger scintillement lorsque la page est redessinée.
- 5) Après avoir fermé la fenêtre de votre navigateur, retournez dans Visual Studio, sur l'affichage des balises de la page. Cherchez dans la boîte à outils de Visual Studio et trouvez la section "Extensions AJAX". (Si vous n'avez pas cette section du fait que vous utilisez une ancienne version des extensions AJAX ou Atlas, référez-vous au mode d'emploi pour enregistrer la section des Extensions AJAX dans la boîte à outils, plus loin dans cet article, ou installez la version actuelle avec l'installeur téléchargeable depuis le site **http://www.asp.net/ajax/**



- a. Problème connu : Si vous installez Visual Studio 2008 Beta 2 sur un ordinateur où Visual Studio 2005 est déjà installé avec les extensions AJAX pour ASP.NET 2.0, Visual Studio 2008 va importer les éléments de la section "Extensions AJAX" dans la boîte à outils. Vous pouvez déterminer si tel est le cas en examinant l'info bulle au survol des composants; ils devraient indiquer "Version 3.5.0.0". S'ils indiquent "Version 2.0.0.0", alors vous avez importé les anciens éléments de la boîte à outils et vous devrez importer les bons manuellement en choisissant "Choisir les éléments" (NdT : via clic droit dans la boîte à outils) dans Visual Studio. Il ne vous sera pas possible d'ajouter les contrôles en version 2 via le designer.

- 6) Avant la balise ouvrante ***<asp:Label>***, ajoutez une ligne blanche puis double-cliquez sur le contrôle "UpdatePanel" dans la boîte à outils. Notez qu'une nouvelle directive @Register est inclue en début de page, indiquant que les contrôles dans l'espace de noms System.Web.UI doivent être insérés en utilisant le préfixe ***asp:***
- 7) Déplacez la balise fermante ***</asp:UpdatePanel>*** après la balise fermante de l'élément Button de telle façon que l'élément soit bien formé et qu'il comprenne le Label et le Bouton.
- 8) Après la balise ouvrante ***<asp:UpdatePanel>***, ouvrez une nouvelle balise. Notez que l'IntelliSense vous propose 2 options. Dans ce cas, créez une balise ***ContentTemplate***. Soyez sur que ces balises englobent le Label et le Bouton de façon à ce que la balise soit bien formée.



- 9) N'importe où dans l'élément ***<form>***, insérez un contrôle ScriptManager en double cliquant sur l'élément ScriptManager dans la boîte à outils.
- 10) Editez la balise ***<asp:ScriptManager>*** de façon à y préciser l'attribut EnablePartialRendering="true"

Listing 3: Détail des balises pour default.aspx avec le rafraichissement partiel activé

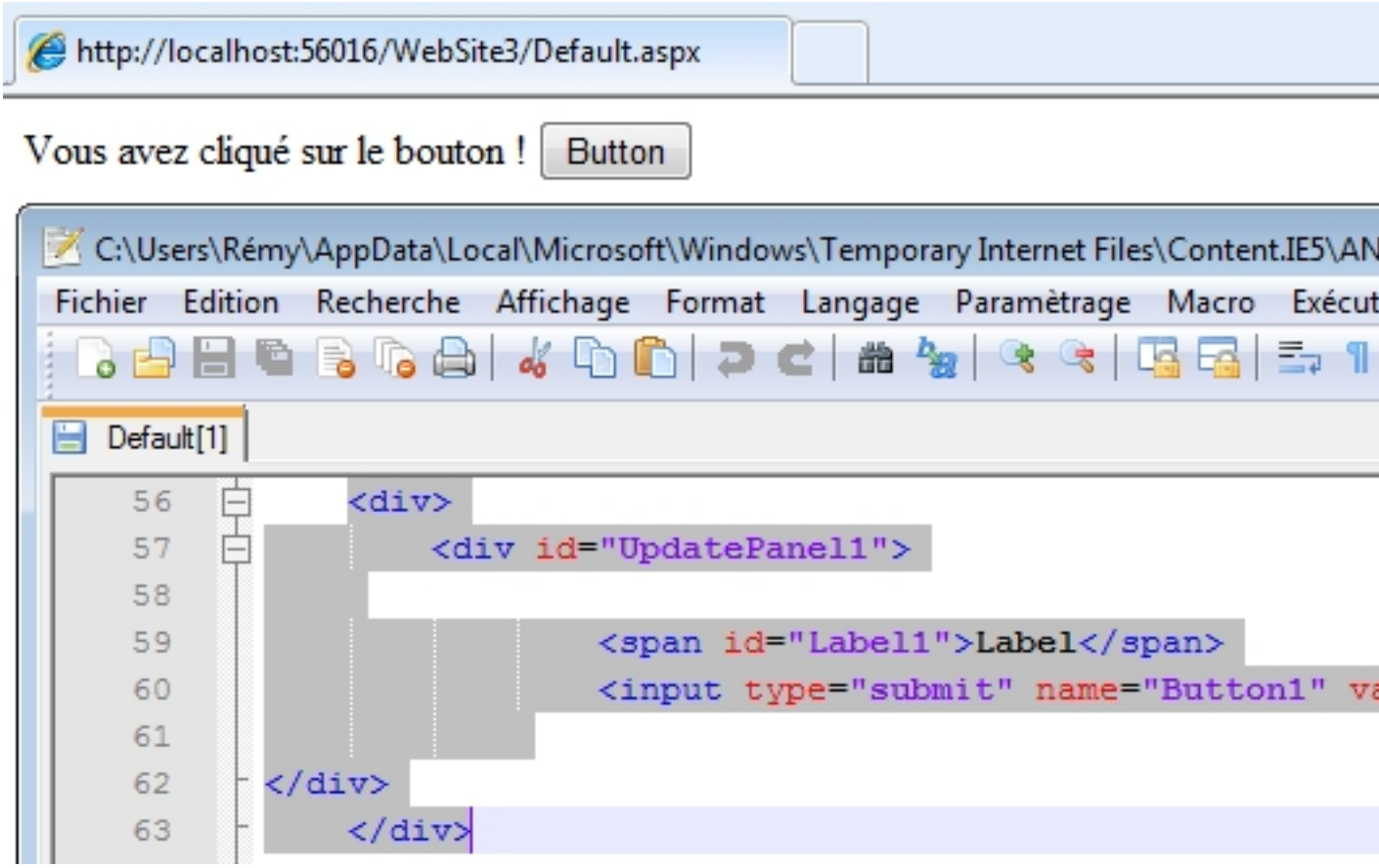
```
<%@ Page Language="C#" AutoEventWireup="true"  CodeFile="Default.aspx.cs" Inherits="_Default" %>
<%@ Register
 Assembly="System.Web.Extensions, Version=1.0.61025.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35"
 Namespace="System.Web.UI" TagPrefix="asp" %>

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

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

Listing 3: Détail des balises pour default.aspx avec le rafraichissement partiel activé

```
    </asp:ScriptManager>
    <div>
        <asp:UpdatePanel ID="UpdatePanel1" runat="server">
            <ContentTemplate>
                <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
                <asp:Button ID="Button1" runat="server" Text="Button" onclick="Button1_Click" />
            </ContentTemplate>
        </asp:UpdatePanel>
    </div>
    </form>
</body>
</html>
```

- 11) Ouvrez votre web.config. Notez que Visual Studio a automatiquement ajouté une série de références vers System.Web.Extensions.dll

- a. Ce qu'il y a de nouveau dans Visual Studio 2008 : Le fichier web.config ajouté avec un projet de site web ASP.NET inclus automatiquement toutes les références nécessaires vers les extensions AJAX ASP.NET, et inclus des sections de configuration commentées qui peuvent être décomentées pour activer des fonctionnalités supplémentaires. Visual Studio 2005 dispose du même modèle de projet une fois les extensions AJAX pour ASP.NET 2.0 installées. Cependant, dans Visual Studio 2008, les extensions AJAX sont opt-out par défaut (ce qui signifie qu'elles sont référencées par défaut mais peuvent être retirées comme références).



- 12) Appuyez sur F5 pour lancer le site web. Notez qu'aucune modification du code source de la page n'a été nécessaire pour activer le rafraichissement partiel. Seules les balises ont été modifiées.

Quand vous lancez le site web, vous devriez remarquer que le rafraichissement partiel est maintenant activé, parce que quand vous cliquez sur le bouton il n'y a plus de clignotement ni de changement de position dans le défilement de la page (bien que non visible dans l'exemple actuel). Si vous regardez le code source de la page affichée après

avoir cliqué sur le bouton, cela confirmera qu'aucun postback n'a eu lieu - le texte original du label fait toujours partie de la source, et le label a été modifié via JavaScript.
Visual Studio 2008 Beta 2 ne semble pas venir avec un modèle prédéfini de site web ASP.NET avec AJAX activé. Cependant, un tel modèle était disponible dans Visual Studio 2005 si les Extensions AJAX d'ASP.NET 2.0 pour Visual Studio 2005 étaient installées. En conséquence, configurer un site web et démarrer avec le modèle de site web avec AJAX activé sera probablement encore plus facile, car le modèle comprend un fichier web.config entièrement configuré (supportant toutes les Extensions AJAX pour ASP.NET, incluant l'accès aux Services WEB et la sérialisation JSON - JavaScript Object Notation) et inclus un UpdatePanel et un ContentTemplate dans le Form de la page principale par défaut. Activer le rafraichissement partiel pour cette page par défaut est aussi simple que ré exécuter le point 10 de cet article et de déposer le contrôle sur la page.

4 - Le contrôle ScriptManager

4.1 - Références du contrôle ScriptManager

Propriétés accessibles depuis la balise :

Nom	Type	Description
AllowCustomErrors-Redirect	Bool	Détermine s'il faut ou non utiliser la section des erreurs personnalisées du web.config pour la gestion des erreurs
AsyncPostBackError-Message	String	Obtient ou définit le message d'erreur envoyé au client si une erreur survient.
AsyncPostBack-Timeout	Int32	Obtient ou définit le délai d'attente client maximum pour qu'une requête asynchrone se termine.
EnableScript-Globalization	Bool	Obtient ou définit si l'internationalisation du script est activée.
EnableScript-Localization	Bool	Obtient ou définit si la localisation du script est activée.
ScriptLoadTimeout	Int32	Obtient ou définit le temps maximum impartit (en secondes) pour le chargement des scripts côté client.
ScriptMode	Enum (Auto, Debug, Release, Inherit)	Obtient ou définit le mode de rendu des scripts.
ScriptPath	String	Obtient ou définit l'emplacement où sont stockés les fichiers de script à envoyer au client.

Propriétés accessibles par code uniquement :

Nom	Type	Description
AuthenticationService	AuthenticationService-Manager	Obtient l'objet AuthenticationService-Manager associé à l'instance du ScriptManager en cours.
IsDebuggingEnabled	Bool	Obtient une valeur qui indique si les versions debug de bibliothèques de scripts clients seront restituées.
IsInAsyncPostback	Bool	Obtient une valeur qui indique si le postback en

		cours est exécutée en mode de rendu partiel.
ProfileService	ProfileService-Manager	Obtient l'objet ProfileService-Manager associé à l'instance du ScriptManager en cours.
Scripts	Collection<Script-Reference>	Obtient un objet ScriptReferenceCollection qui contient les objets ScriptReference, chacun représentant un fichier de script restitué au client.
Services	Collection<Service-Reference>	Obtient un objet ServiceReferenceCollection qui contient un objet ServiceReference pour chaque service Web qu'ASP.NET expose sur le client pour les fonctionnalités AJAX.
SupportsPartialRendering	Bool	Obtient une valeur qui indique si le client prend en charge le rendu de page partiel. Si la propriété renvoie false, alors toutes les requêtes seront des postbacks standards.

Méthode publique :

Nom	Type	Description
SetFocus(string)	Void	Attribue le focus au contrôle précisé lorsque la requête est terminée.

Balises enfants :

Tag	Description
<AuthenticationService>	Fournit des détails à propos du proxy vers le service d'authentification d'ASP.NET.
<ProfileService>	Fournit des détails à propos du proxy vers le service de profilage d'ASP.NET.
<Scripts>	Fournit des références vers des scripts additionnels.
<asp:ScriptReference>	Décrit une référence à un script
<Service>	Fournit des références supplémentaires vers des Services Web pour lesquels un proxy sera généré.
<asp:ServiceReference>	Décrit une référence à un service Web

Le contrôle ScriptManager est le c□ur des extensions ASP.NET AJAX. Il fournit l'accès à la bibliothèque des scripts, le support du rafraichissement partiel et fournit un support complet pour des services ASP.NET supplémentaires (tels que l'authentification et le profilage, mais également d'autres Services Web). Le contrôle ScriptManager fournit également le support de l'internationalisation et localisation pour les scripts clients.

4.2 - Fournir des scripts alternatifs et / ou supplémentaires

Alors que les Extensions AJAX pour ASP.NET 2.0 comprennent l'intégralité du code du script, tant en mode debug que production, comme ressources incorporées dans les assemblies référencées, les développeurs sont libres de rediriger le ScriptManager vers des scripts personnalisés ou encore d'ajouter des scripts supplémentaires nécessaires.
Pour écraser le lien par défaut vers les scripts habituellement inclus (tels que ceux qui supportent l'espace de noms Sys.WebForms et le système de typage personnalisé), vous devez vous abonner à l'événement ResolveScriptReference de la classe ScriptManager. Quand cette méthode est invoquée, le gestionnaire d'événements à l'opportunité de changer le chemin vers les scripts en question; le ScriptManager enverra alors une version différente ou modifiée des scripts au client.
De plus, les références aux scripts (représentées par la classe ScriptReference) peuvent être ajoutées via le code ou la balise. Pour faire cela, modifiez soit par code la collection ScriptManager.Scripts, ou ajoutez un tag <asp:ScriptReference> en dessous du tag <Scripts> qui est un enfant direct du contrôle ScriptManager.

4.3 - Gestion d'erreurs personnalisée pour les UpdatePanels

Même si les rafraichissements sont gérés à travers des déclencheurs définis par des UpdatePanels, la gestion des erreurs et des messages d'erreur personnalisés s'effectue via l'instance du contrôle ScriptManager de la page. Cela s'effectue en exposant un événement, AsyncPostBackError, à laquelle on peut abonner la page pour fournir une gestion d'erreurs personnalisée.
En consommant l'événement AsyncPostBackError, vous pouvez préciser la propriété AsyncPostBackErrorMessage, ce qui provoquera l'apparition d'une boîte d'alerte à la fin du callback.
La personnalisation côté client est également possible plutôt que d'utiliser la boite d'alerte par défaut. Vous pourriez, par exemple, afficher une personnalisée plutôt que la boîte de dialogue par défaut. Dans ce cas, vous pouvez gérer l'erreur dans un script client :

```
<script type="text/javascript">
    <!--
        Sys.WebForms.PageRequestManager.getInstance().add_EndRequest(Request_End);
        function Request_End(sender, args){
            if (args.get_error() != undefined){
                var errorMessage = "";
                if (args.get_response().get_statusCode() == "200"){
                    errorMessage = args.get_error().message;
                }else{
                    // Le serveur n'était pas le problème
                    errorMessage = "Une erreur inconnue est survenue...";
                }
                // Faites ce que vous voulez avec le message d'erreur ici.
                // Maintenant, il faut signaler qu'on a géré l'erreur.
                args.set_errorHandled(true);
            }
        }
    // -->
    </script>
```

De façon très simple, le script ci-dessus définit une fonction de rappel (callback), via le noyau AJAX côté client, pour lorsque la requête asynchrone sera terminée. Il vérifie ensuite si un message d'erreur a été rapporté, et si c'est le cas, le traite pour finalement indiquer au noyau que l'erreur a été gérée dans un script personnalisé.

4.4 - Support de l'Internationalisation et de la Localisation

Le contrôle ScriptManager fournit un support étendu pour la localisation des chaînes des scripts et les composants de l'interface utilisateur; cependant, ce sujet dépasse le cadre de cet article. Pour en savoir plus, lisez l'article **Comprendre l'internationalisation avec ASP.NET AJAX**.

6 - Le contrôle UpdatePanel

6.1 - Références du contrôle UpdatePanel

Propriétés accessibles depuis la balise :

Nom	Type	Description
ChildrenAsTriggers	bool	Précise si les contrôles enfants provoquent automatiquement un rafraichissement lorsqu'ils déclenchent un postback.
RenderMode	enum (Block, Inline)	Précise la façon dont le contenu sera présenté (dans un ou dans un
UpdateMode	enum (Always, Conditional)	Précise si l'UpdatePanel est actualisé à chaque rafraichissement partiel ou uniquement si un déclencheur est activé.

Propriétés accessibles par code uniquement :

Nom	Type	Description
IsInPartialRendering	bool	Obtient une valeur qui indique si le contrôle UpdatePanel est mis à jour par suite d'une publication asynchrone.
ContentTemplate	ITemplate	Obtient ou définit le modèle qui définit le contenu du contrôle UpdatePanel.
ContentTemplateContainer	Control	Obtient un objet de contrôle auquel vous pouvez ajouter des contrôles enfants par programme.
Triggers	UpdatePanel-TriggerCollection	Obtient un objet UpdatePanelTriggerCollection qui contient des objets AsyncPostBackTrigger et PostBackTrigger enregistrés de manière déclarative pour le contrôle UpdatePanel.

Méthode publique :

Nom	Type	Description
Update()	vois	Entraîne une mise à jour du contenu d'un contrôle UpdatePanel.

Balises enfants :

Tag	Description
<ContentTemplate>	Précise les balises qui serviront à l'affichage du résultat du rafraichissement partiel. Enfant de <asp:UpdatePanel>.
<Triggers>	Définit une collection de contrôles associés au rafraichissement de l'UpdatePanel. Enfant de <asp:UpdatePanel>.
<asp:AsyncPostBackTrigger>	Définit un déclencheur qui invoque en rafraichissement partiel de la page pour l'UpdatePanel concerné. Il peut s'agir ou non d'un contrôle descendant de l'UpdatePanel en question. Se rapporte à un nom d'événement. Enfant de <Triggers>
<asp:PostBackTrigger>	Définit un contrôle qui déclenche un rafraichissement complet de la page. Il peut s'agir ou non d'un contrôle descendant de l'UpdatePanel en question. Se rapporte à l'objet. Enfant de >Triggers<.

Le contrôle UpdatePanel est le contrôle qui délimite le contenu côté serveur qui prendra part dans la fonctionnalité de rafraichissement partiel des extensions AJAX. Il n'y a pas de restriction quant au nombre de contrôles UpdatePanel qui peut être placé sur une page, et ils peuvent être imbriqués. Chaque UpdatePanel est isolé de façon à ce que chacun travaille indépendamment (vous pouvez avoir deux UpdatePanels qui s'exécutent en même temps, actualisant deux parties différentes de la page, indépendamment du postback de la page).
Le contrôle UpdatePanel travaille principalement avec des déclencheurs - par défaut, n'importe quel contrôle contenu dans le **ContentTemplate** d'un UpdatePanel qui provoque un postback est enregistré comme déclencheur pour l'UpdatePanel concerné. Cela signifie que l'UpdatePanel est capable de fonctionner avec les contrôles à liaison de données (tel que le contrôle GridView), avec les contrôles utilisateurs, et ils peuvent être programmés dans le script. Par défaut, quand un rafraichissement partiel est déclenché, tous les UpdatePanels de la page seront actualisés, qu'ils aient, ou non, définit un déclencheur pour une telle action. Par exemple, si un UpdatePanel contient un Bouton, et que le Bouton est cliqué, tous les UpdatePanels de la page seront actualisés par défaut. Cela est du au fait que, par défaut, la propriété **UpdateMode** d'un UpdatePanel est définie à **Always**. Alternativement, vous pouvez définir cette propriété à **Conditional**, ce qui signifie que l'UpdatePanel ne sera actualisé que si un déclencheur spécifique est activé.

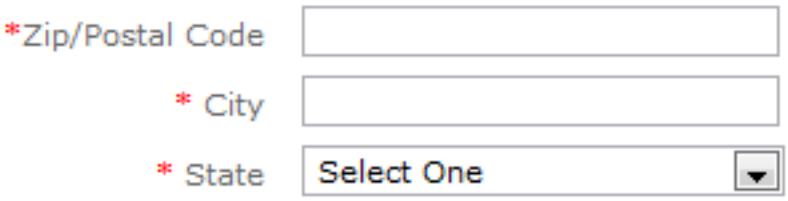
6.2 - Note pour les contrôles personnalisés

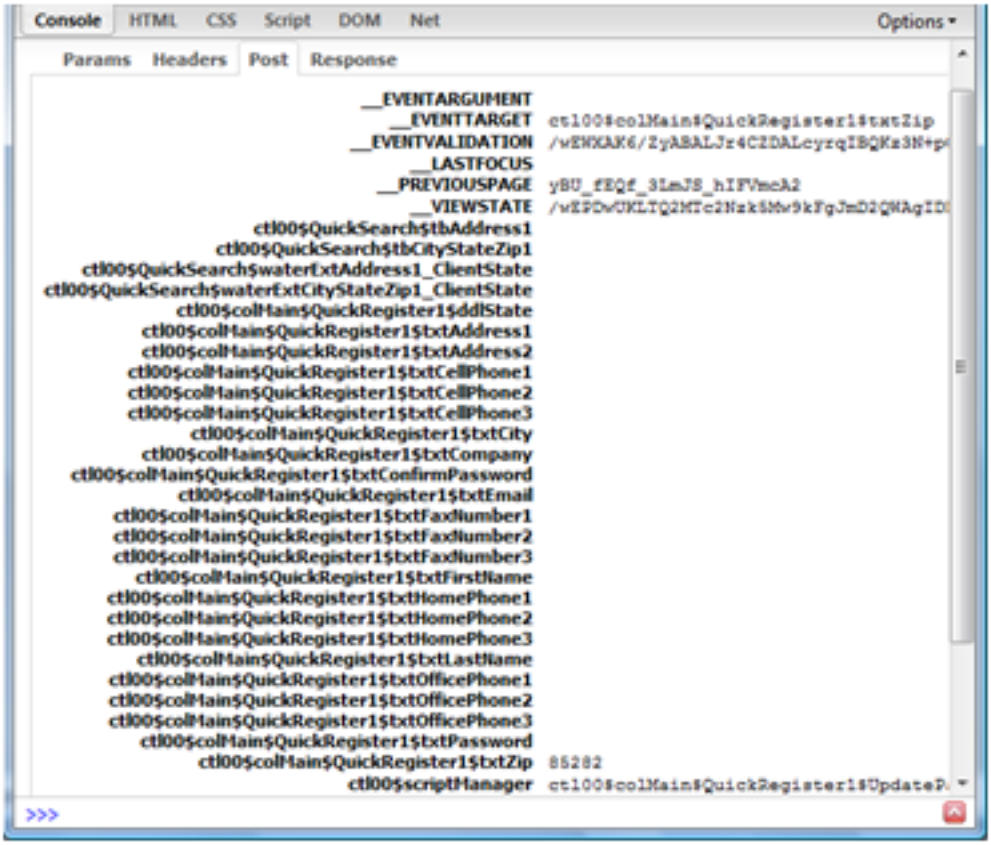
Un UpdatePanel peut être ajouté à n'importe quel contrôle utilisateur ou contrôle personnalisé; cependant, la page sur laquelle ces contrôles sont inclus doit également comprendre un contrôle ScriptManager pour lequel la propriété EnablePartialRendering est définie à **true**. Une méthode pour vous affranchir de ce problème lorsque vous utilisez des contrôles Web personnalisés et de surcharger la méthode protected **CreateChildControls()** de la classe **CompositeControl**. En procédant de la sorte, vous pouvez injecter un UpdatePanel entre les enfants du contrôle et le reste si vous déterminez que la page supporte le rafraichissement partiel; autrement, vous pouvez simplement placer les contrôles enfants dans une instance de contrôle conteneur.

6.3 - Considérations sur l'UpdatePanel

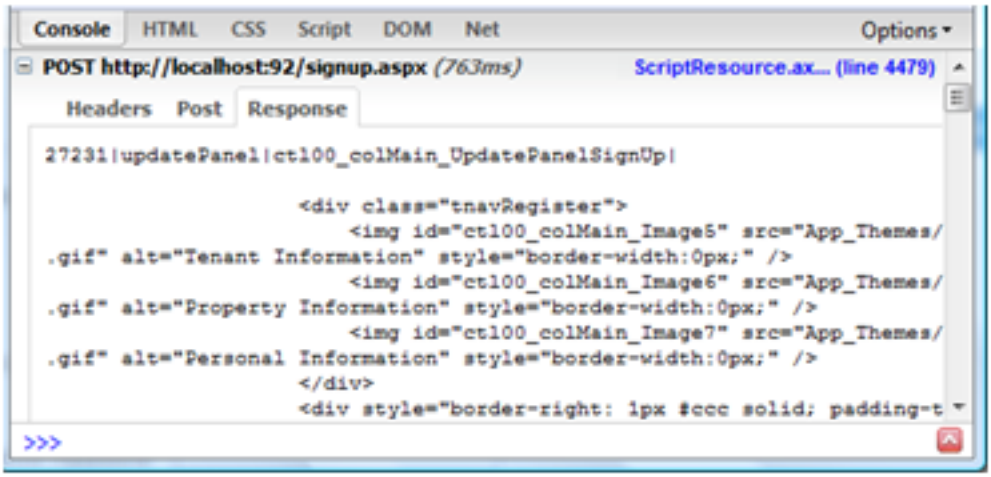
Le fonctionnement du contrôle UpdatePanel ressemble à celui d'une boîte noire, encapsulant les postbacks ASP.NET dans le contexte de l'objet JavaScript XMLHttpRequest. Cependant, il y a un impact significatif des performances à prendre en compte, tant en terme de comportement que de vitesse. Pour comprendre comment l'UpdatePanel fonctionne, de façon à pouvoir décider quand son utilisation est perspicace, vous devriez examiner les échanges AJAX. L'exemple suivant repose sur un site existant et Mozilla Firefox avec l'extension Firebug (Firebug permet d'observer les données de l'objet XMLHttpRequest).
Imaginez un formulaire qui, entre autres choses, contient un contrôle textbox pour saisir un code postal. Ce contrôle étant supposé remplir les champs ville et pays du formulaire. Ce formulaire collecte finalement des informations

d'identification telles qu'un nom d'utilisateur, une adresse et des informations de contact. Il y a de nombreuses exigences de conception à prendre en compte, basées sur des pré-requis du projet concerné.





Lors du premier chargement de cette application, un contrôle est créé qui contient l'entièreté des informations d'identification de l'utilisateur, y compris le code postal, la ville et le pays. Le contrôle entier a été inclus dans un UpdatePanel et placé dans un Web Form. Lorsque le code postal est saisi par l'utilisateur, l'UpdatePanel détecte l'événement (l'événement TextChanged en l'occurrence, soit en spécifiant un déclencheur ou en définissant la propriété ChildrenAsTriggers à **true**. Ajax envoie tous les champs contenus dans l'UpdatePanel, comme montré dans la capture de FireBug (voir le diagramme sur la droite).
Comme illustré sur la capture d'écran, les valeurs de tous les contrôles contenus dans l'UpdatePanel sont transmises (dans ce cas, elles sont toutes vides), ainsi que le ViewState. Tout compris, c'est environ 9kb de données qui sont envoyés là où seulement 5 bytes étaient nécessaires pour effectuer cette requête. La réponse est encore plus étonnante : 57kb sont envoyés au client pour simplement actualiser la valeur d'un champ texte et d'une liste déroulante.
Il est également intéressant de voir comment ASP.NET AJAX actualise la présentation des contrôles. La portion de réponse à la requête d'actualisation de l'UpdatePanel est montrée dans la console d'affichage de FireBug, sur la gauche; il s'agit d'une chaîne formatée, délimitée qui est analysée par le script client et ensuite réassemblée sur la page. Plus précisément, ASP.NET AJAX définit la propriété innerHTML de l'élément HTML qui représente l'UpdatePanel côté client. Pendant que le navigateur régénère le DOM, il peut survenir un certain délai, dont la durée dépend de la quantité d'information qui doit être traitée.
La régénération du DOM déclenche un certain nombre de problèmes supplémentaires :



- Si l'élément HTML qui a le focus se trouve dans l'UpdatePanel, il perdra le focus. Ainsi, pour les utilisateurs qui ont appuyé sur la touche Tab pour quitter la zone de saisie du code postal, leur prochaine destination aurait du être la zone de saisie de la Ville. Cependant, une fois que l'UpdatePanel actualise son contenu, le formulaire qu'il contient perd le focus, et appuyer sur la touche Tab donnera le focus au premier élément non contenu dans l'UpdatePanel.
- Si un script client personnalisé est en train d'utiliser des éléments du DOM, les références éventuellement maintenues dans des fonctions risquent de devenir invalides lors de postback partiel.

Les UpdatePanels ne sont pas destinés à être des "solutions miracles". Au contraire, ils fournissent une solution rapide à certaines situations, telles que du prototypage, des mises à jour de petits contrôles, et fournir une interface familière aux développeurs ASP.NET qui sont peut-être familiers avec le model objet .NET mais moins à l'aise avec le DOM. Il existe de nombreuses alternatives qui peuvent donner de meilleures performances, selon le scénario de l'application :

- Utiliser les Méthodes de Page et JSON (JavaScript Object Notation) permet au développeur d'invoquer une méthode statique d'une page comme s'il s'agissait d'un appel à un service Web. Comme les méthodes sont statiques, aucun état n'est nécessaire; le script appelant fournit les paramètres et le résultat est retourné de façon asynchrone.
- Pensez à utiliser un service Web et JSON si vous avez un contrôle qui doit être utilisé à plusieurs endroits de votre application. De nouveau, cela nécessite peu de travail et fonctionne de façon asynchrone.

Ajouter des fonctionnalités à travers un Service Web ou des méthodes de page a également des inconvénients. D'abord et avant tout, les développeurs ASP.NET tendent habituellement à créer des petits composants regroupant certaines fonctionnalités au sein de contrôles utilisateurs (fichiers .ascx). Les méthodes de Page ne peuvent pas être intégrées à ces contrôles; elles doivent être intégrées directement dans le code-behind de la page .aspx. Les services Web, quant à eux, doivent être compris dans des fichiers .asmx. En fonction de l'application, cette architecture peut ne pas respecter le principe de responsabilité unique en ce sens que la fonctionnalité d'un composant unique est maintenant répartie à travers plusieurs composants physiques qui peuvent avoir peu de cohésion entre eux.
Pour finir, si une application nécessite d'utiliser des UpdatePanels, les règles suivantes devraient vous aider gérer les problèmes et la maintenance.

- **Evitez autant que possible d'emboîter les UpdatePanels, pas seulement au sein du même composant, mais également à travers différents composants.** Par exemple, un contrôle sur une page qui contient lui-même un contrôle contenant un UpdatePanel dans lequel se trouve également un contrôle contenant un UpdatePanel est de l'emboîtement inter composants. Procéder sans emboîtement permet de savoir facilement quel élément va être rafraichi et évite des actualisations involontaires d'UpdatePanels enfants.
- **Maintenez la propriété *ChilrdrenAsTriggers* à false et définissez explicitement les événements déclencheurs.** Utiliser la collection <Triggers> est la méthode la plus propre pour gérer les événements et peut éviter des comportements inattendus, facilitant la maintenance du composant et forçant les développeurs à effectuer des opt-in pour les événements.

- **Utilisez le composant le plus petit possible pour implémenter une fonctionnalité.** Comme souligné dans la discussion sur le service du code postal, encapsuler le strict minimum réduit la charge sur le serveur, le temps de traitement, la taille des informations nécessaires aux échanges client-serveur, ce qui améliore grandement les performances.

7 - Le contrôle UpdateProgress

7.1 - Références du contrôle UpdateProgress

Propriétés accessibles depuis la balise :

Nom	Type	Description
AssociatedUpdate-PanelID	String	Définit l'ID de l'UpdatePanel auquel cet UpdateProgress doit se référer.
DisplayAfter	Int	Définit le délai en millisecondes qui sépare le début de la requête asynchrone de l'affichage de l'UpdateProgress.
DynamicLayout	Bool	Définit si le modèle de progression est restitué dynamiquement.

Balise enfant :

Tag	Description
<ProgressTemplate>	Obtient ou définit le modèle qui définit le contenu du contrôle.

Le contrôle UpdateProgress fournit un moyen de tenir l'utilisateur informé de la progression du traitement pendant le temps nécessaire à la requête. Cela permet de faire savoir à vos utilisateurs que vous êtes en train d'effectuer un traitement même si cela n'est pas apparent et cela est d'autant plus important que de plus en plus d'entre eux sont habitués à la barre de progression dans la barre de statut du navigateur ainsi qu'au bouton "actualiser".
Il est intéressant de noter que le contrôle UpdateProgress peut apparaître n'importe où dans la hiérarchie de la page. Cependant, au cas où le postback partiel serait provoqué par un UpdatePanel enfant (un UpdatePanel lui-même compris dans un autre UpdatePanel), le postback déclenché par l'UpdatePanel enfant provoquera l'affichage de l'UpdateProgress pour son UpdatePanel mais également pour celui du parent. Par contre, si le déclencheur est un enfant direct de l'UpdatePanel parent, seul l'UpdateProgress qui lui est associé sera affiché.

8 - Résumé

Les extensions Microsoft ASP.NET AJAX sont des éléments sophistiqués destinés à aider à rendre le contenu de votre site web plus accessible et à fournir à vos utilisateurs une expérience utilisateur plus riche. Inclus dans les extensions AJAX pour ASP.NET, les contrôles de restitution partielle, comprenant le ScriptManager, l'UpdatePanel et l'UpdateProgress, constituent la partie la plus visible du toolkit.
Le composant ScriptManager fournit la collection des scripts JavaScript côté client et permet le fonctionnement des différents composants, côté client comme côté serveur, avec un minimum de développement.
Le contrôle UpdatePanel est la "boîte magique" visuelle - les balises placées à l'intérieur peuvent avoir du code côté serveur et ne pas déclencher de rafraichissement de la page. Les contrôles UpdatePanel peuvent être imbriqués et peuvent dépendre de contrôles placés dans d'autres UpdatePanels. Par défaut, les UpdatePanels gèrent tous les postbacks provoqués par leurs contrôles enfants, même si cette fonctionnalité peut-être configurée plus précisément, tant de façon déclarative que via le code.
Lorsqu'ils utilisent le contrôle UpdatePanel, les développeurs doivent être conscients de l'impact sur les performances qu'ils peuvent avoir. Les alternatives éventuelles comprennent les services Web et les méthodes de page, bien que leur utilisation potentielle dépend de la conception de l'application.
Le contrôle UpdateProgress permet à l'utilisateur de savoir qu'il ou elle n'est pas ignoré, et que la requête en arrière plan se poursuit pendant que la page ne donne aucun retour à son action. Il offre également la possibilité d'annuler un rafraichissement partiel en cours.
Utilisés conjointement, ces outils aident à créer une expérience utilisateur riche et sans accros en faisant travailler le serveur de façon plus transparente pour l'utilisateur et en interrompant moins l'enchaînement des actions.

Biographie

Scott Cate travaille avec les technologies Microsoft Web depuis 1997 et est le président de myKB.com (**www.myKb.com**) où il s'est spécialisé dans l'écriture d'applications basées sur ASP.NET et orientées vers la gestion informatique des Bases de Connaissances. Scott peut-être contacté via email **scott.cate@myKb.com** ou via son blog : **http://weblogs.asp.net/scottcate**.