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

1. Qu'est-ce qu'AJAX ?

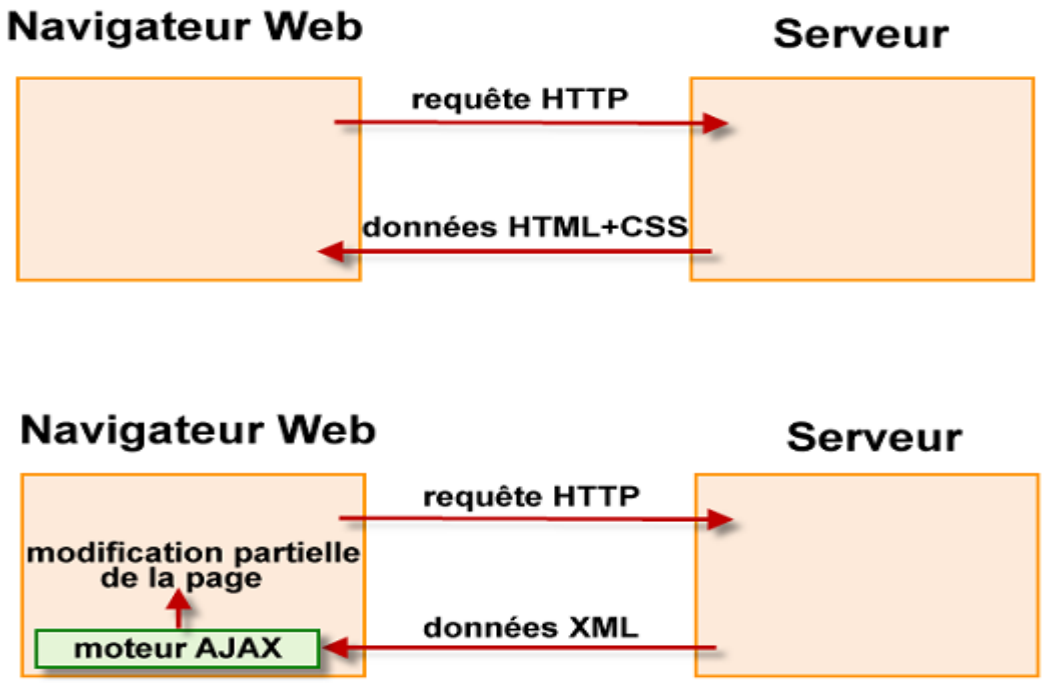
Lorsque vous naviguez de page en page sur un site Web traditionnel (entendez par là non-AJAX), les actions de l'internaute se traduisent par les actions suivantes :

1. Envoi d'une requête au serveur afin d'obtenir une nouvelle page.
2. Calcul de la nouvelle page par le serveur et envoi des données HTML/CSS correspondantes.
3. Affichage de ces données dans le navigateur.

Cette technique fonctionne très bien dans la plupart des cas, mais parfois seule une partie de la page nécessite d'être mise à jour. C'est là qu'intervient AJAX :

1. Dans un premier temps, envoi d'une requête au serveur afin d'obtenir les données qui seront affichées dans une partie bien précise de la page actuelle.
2. Calcul des données demandées par le serveur et envoi de ces données au navigateur au format XML.
3. Réception des données envoyées par le programme (on dit aussi moteur) AJAX qui les a demandées et affichage dans un endroit bien précis de la page actuelle sans toucher au reste de la page.

La figure suivante résume ces deux modes de fonctionnement.



Les deux modes de fonctionnement d'un site Web : client-serveur et AJAX

Si, dans la plupart des cas, un fonctionnement traditionnel est entièrement satisfaisant, les performances d'affichage peuvent être grandement améliorées dans certains cas particuliers, comme par exemple l'affichage de données mises à jour à intervalles réguliers (cours d'actions en bourse par exemple), la sauvegarde des données pendant la saisie dans un formulaire, la mise à jour et/ou la vérification dynamique des champs d'un formulaire en fonction des

données saisies par l'utilisateur, la saisie prédictive (comme le fait Google en proposant des réponses lorsque vous commencez à taper quelques caractères dans la case de recherche), etc.

AJAX est l'acronyme d'*Asynchronous JavaScript and XML*. Le langage JavaScript est utilisé pour demander des données au serveur. Ces données lui sont retournées de façon asynchrone sous une forme XML.

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 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-transitional.<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>
```

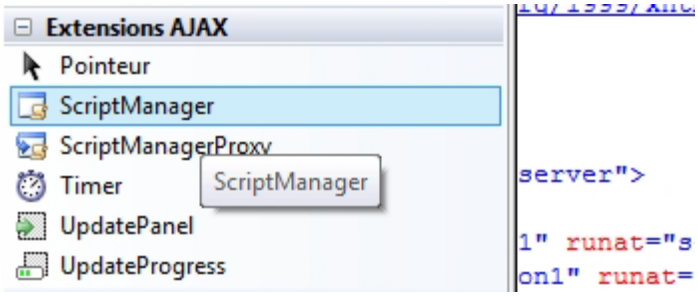
```
</body>
</html>
```

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. 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/**

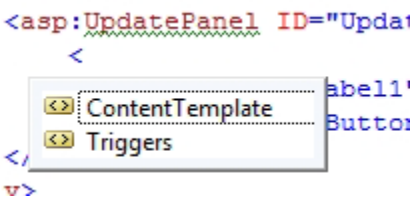


a. Problème connu : Si vous installez Visual Studio 2008 Beta 2 sur un ordinateur où Visual Studio 2005 est déjà installé avec les extensions AJAX pour ASP.NET 2.0, Visual Studio 2008 va importer les éléments de la section "Extensions AJAX" dans la boîte à outils. Vous pouvez déterminer si tel est le 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.

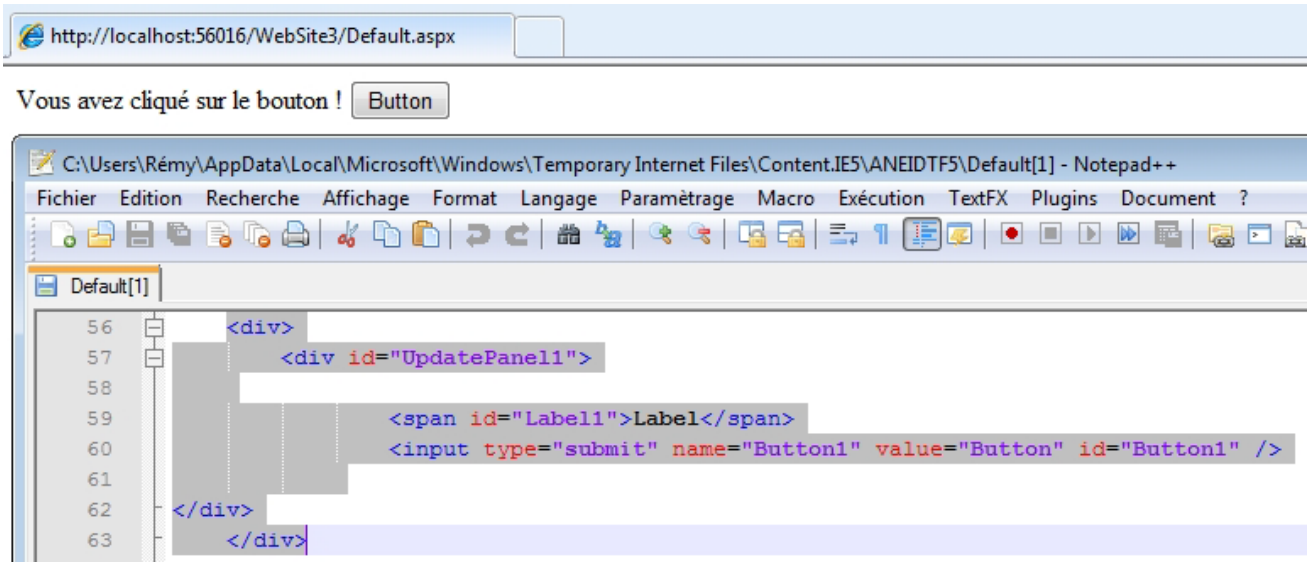


• 9) N'importe où dans l'élément ***<form>***, insérez un contrôle ScriptManager en double cliquant sur l'élément ScriptManager dans la boîte à outils.
• 10) Editez la balise ***<asp:ScriptManager>*** de façon à y préciser l'attribut EnablePartialRendering="true"

```
<%@ 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-transitional.<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">

</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.



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

Le contrôle ScriptManager est le cœur 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.

5 - Le contrôle UpdatePanel

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 dû 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 - Le contrôle UpdateProgress

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 luimême 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.

7 - 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.