Rafraichissement conditionné d'une page en .NET

Test avec AJAX pour rafraichissement

En utilisant AJAX, voici une possibilité de faire un rafraichissement conditionné. Nous verrons dans cet article une méthode très simple pour effectuer un test qui conditionne le rafraichissement d'une page, ceci pour ne pas surcharger le serveur.

Introduction

Avec l'augmentation des débits pour les connexions Internet, nous pouvons maintenant créer des pages web de plus en plus riches. Ceci nous entraine vers un poids des pages de plus en plus importants. On remarque aussi pour les développements que nous utilisons tous les users controls, les servers controls ou les webparts. Ceci permet de découper nos développement Web en composants (ou briques) de bases.

Une fois toutes ces briques assemblées dans un "portail", nous constatons que pour certains composants nous avons besoin d'une interaction sans attendre que l'utilisateur effectue une action. Il faut donc voir comment ajouter cet automatisme.

Présentation

Notre cas est simple, nous voulons pouvoir rafraichir notre page uniquement dans le cas où un test sur une base de données est positif. Nous aurons donc une couche métier qui lancera une procédure stockée. Sur cette couche métier, nous aurons une page web extrêmement simple qui renvoi True ou False et enfin un code AJAX qui testera cette page pour effectuer ou non ce refresh.

Nous découperons donc cet article en 2 parties :

- Script coté serveur (C#)
- Script coté client (AJAX)

Script côté serveur

Nous devons créer une procédure stockée (ou l'équivalent) qui fera le test sur la base. Ce test est à adapter suivant votre besoin fonctionnel, mais le principe est de tester si l'utilisateur courant à un nouvel élément ou non.

Nous devons par la suite créer la couche de présentation qui exécutera ce test. Pour cela, nous pouvons passer par différentes techniques :

- Un WebService : ASMX
- Un Handler : ASHX
- Une page : ASPX

Chacune de ces possibilités a ses avantages, nous pouvons dans notre cas présent supprimer le WebService, à cause du traitement des enveloppes SOAP. Ceci est difficilement utilisable avec les appels AJAX.

Il nous reste donc le passage par le Handler ou la page web. Nous avons un très bon article sur l'utilisation du Handler écrit par Aurélien :

- **Ajax.ActiveUsersList (FR)**

De ce fait, nous allons simplement créer une page ASPX qui renvoi uniquement "True" ou "False", suivant le cas. Notre exemple sera en C#, mais facilement traduisible en VB.NET.

Dans notre page ASPX, nous aurons uniquement le référencement vers le fichier de code behind (xxxx.aspx.cs). Il faut enlever tout ce qui est ajouté à la création de celle-ci.

Dans le fichier de code, nous aurons l'appel de la couche métier pour la base de données.

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace ASP-PHP.AjaxProject
{
  /// <summary>
  /// Description résumée de AjaxProject.
  /// </summary>
  public class AjaxProjectASPX : System.Web.UI.Page
  {
    private void Page_Load(object sender, System.EventArgs e)
    {
      HttpRequest Request = System.Web.HttpContext.Current.Request;
      HttpResponse Response = System.Web.HttpContext.Current.Response;
      bool retour = false;
      string LeLogin = string.Empty;
      try
      {
        LeLogin = System.Web.HttpContext.Current.User.Identity.Name.ToString();
        //Notre appel à la couche métier qui renvoie vrai ou faux
        retour = CheckNewInDB(LeLogin);
      }
      catch
      {
        retour = false;
      }
      finally
      {
        LeLogin = null;
      }
      Response.Write(retour);
    }

    #region Code généré par le Concepteur Web Form
    override protected void OnInit(EventArgs e)
    {
      //
      // CODEGEN : Cet appel est requis par le Concepteur Web Form ASP.NET.
      //
      InitializeComponent();
      base.OnInit(e);
    }

    /// <summary>
    /// Méthode requise pour la prise en charge du concepteur - ne modifiez pas
    /// le contenu de cette méthode avec l'éditeur de code.
    /// </summary>
    private void InitializeComponent()
    {
      this.Load += new System.EventHandler(this.Page_Load);
    }
    #endregion
  }
}
```

Nous voyons donc que cette page est vraiment très simple, la seule complication est la couche métier permettant de faire ce test, il faut donc adapter cette partie à votre besoin. Dans mon cas d'origine, il me fallait tester tous les nouveaux éléments affectés à l'utilisateur courant.

Nous allons maintenant regarder la couche à ajouter dans le client WEB.

Script côté client

Cette partie est celle préférée par Aurélien. En effet, nous allons utiliser le concept de base de l'AJAX. En effet, notre but est que ce script JS effectue ce test directement sur la page web que nous venons de créer. Le fait de passer par cet appel Javascript est que cet appel utilise le context courant et donc transmet le login de l'utilisateur courant.

```
<script language="javascript">
var req;
var sURL=unescape(window.location.pathname);
var url='LapageDeTest.aspx';

function refresh()
{
  window.location.href=sURL;
}
function doLoad()
{
  setTimeout( "refresh()", 1000 );
}
function doReloadJS()
{
  setTimeout( "loadXMLDoc(url)", 600000);
}

function processReqChange()
{
   if (req.readyState==4) {
      // only if "OK"
      if (req.status==200) {
         if (req.responseText=='True') {
            doLoad();
         }
         else
         {
            doReloadJS();
         }
      }
   }
}

function loadXMLDoc(url)
{
   if (window.XMLHttpRequest) {
      req = new XMLHttpRequest();
      req.onreadystatechange = processReqChange;
      req.open("GET", url, true);
      req.send(null);
   } else if (window.ActiveXObject) {
      req = new ActiveXObject("Microsoft.XMLHTTP");
      if (req) {
         req.onreadystatechange = processReqChange;
         req.open("GET", url, true);
         req.send();
      }
   }
}
loadXMLDoc(url);
</script>
```

Dans cet exemple, nous voyons que si la valeur est True, nous rechargeons la page, sinon on place un timer de 10 minutes (600 000 Millisecondes) pour exécuter ce test à nouveau.

Conclusion

Nous avons à partir de cet article faire une approche très simple de ce que la technologie AJAX peut apporter dans un développement de site web. Il n'est pas du tout indispensable d'utiliser celle-ci, pour plusieurs raisons telle que l'obligation que le poste client accepte l'exécution des scripts JS ou encore l'augmentation de la complexité de son code, ...

Quoi qu'il en soit, cette technologie a le vent en poupe et la mise en place de sites comme **Windows Live** ou encore de la nouvelle version en développement du site Hotmail nous impose de comprendre les rouages de cette technologie.

Si vous souhaitez aller bien plus loin dans cette apprentissage, je vous invite à lire l'article de notre ami Aurelien très complet sur le sujet.

Voici quelques liens utiles si cet article vous a intéressé :

- **AJAX + ASP ou PHP (FR)**
- **Ajax.ActiveUsersList - Article d'Aurelien (FR)**
- **Very Dynamic Web Interfaces (US)**
- **Tutorial sur les requêtes serveur en JavaScript (FR)**
- **Ajax.NET - A free library for the Microsoft .NET Framework (US)**
- **IServerXMLHTTPRequest/ServerXMLHTTP Members (US)**
- **Atlas Project (US)**

En vous souhaitant de bons projets de développement.

Romelard Fabrice (alias F___)
Consultant Technique **ILEM**