

# SharePoint 2007 et Silverlight 2.0 : Premiers pas

---

## Introduction

Avec l'arrivée à maturité de Silverlight avec le lancement de la v2.0, il devient intéressant de l'utiliser pour proposer une interface plus souple pour l'affichage et la manipulation de données.

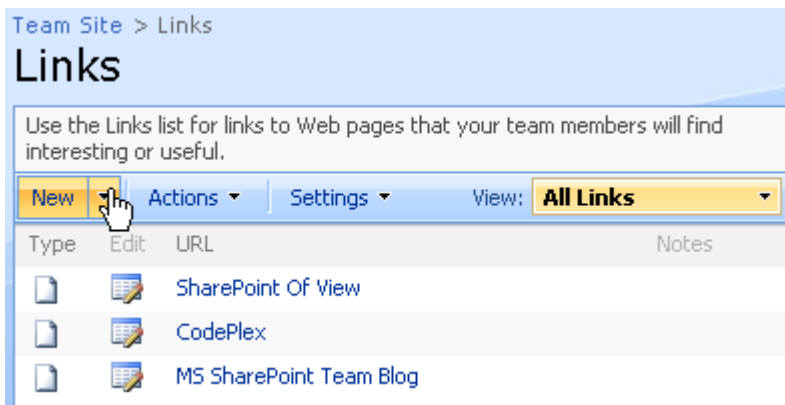
Le projet "[Silverlight Blueprint for SharePoint](#)" passé depuis quelques temps maintenant sur [CodePlex](#) est un bon point de départ pour tester et comprendre certains éléments d'intégration de Silverlight au sein de SharePoint. Mais nous allons adopter ici la démarche pas à pas pour utiliser Silverlight en combinaison avec SharePoint qui tiendra lieu de source d'information.

L'objectif de ce tutoriel est d'afficher le contenu d'une liste de liens standard dans une application Silverlight en se basant sur le protocole « Remote Procedure Call » (requête HTTP) proposé par SharePoint. Le code source du projet est fourni en téléchargement à la fin de l'article.

## Récupération des informations via RPC

Nous avons tout d'abord besoin d'interroger à distance SharePoint pour obtenir la liste des éléments (liens) de notre liste. En effet, étant donné que Silverlight tourne en tant que plugin dans le navigateur, c'est bien un accès distant qui est effectué et non un appel direct à l'API SharePoint.

Voici notre liste de liens source :

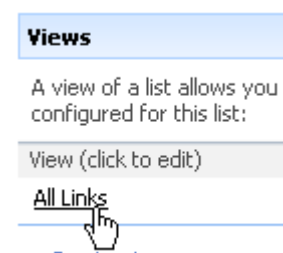


Nous allons utiliser pour cela le protocole « Remote Procedure Call » qui est un des moyens d'interroger SharePoint à distance, et qui est relativement simple vu qu'il consiste en une simple requête HTTP. Je pourrai ainsi utiliser la classe [WebClient](#) pour l'appeler. [Renaud Comte en avait parlé il y a déjà quelques temps sur son blog](#) et proposait une technique pour obtenir les identifiants nécessaires à la requête, je vous propose ici une autre alternative pour la fabriquer.

Cette URL d'affichage de liste a le format suivant:

[http://<url site>/\\_vti\\_bin/owssvr.dll?cmd=Display&List={Guid}&View={Guid}&XMLDATA=TRUE](http://<url site>/_vti_bin/owssvr.dll?cmd=Display&List={Guid}&View={Guid}&XMLDATA=TRUE)

Où les 2 « {Guid} » correspondent aux identifiants de la liste et de la vue. Rendez-vous dans les paramètres de la liste et cliquez sur la vue qui vous intéresse :



A partir de là, vous aurez toutes les informations directement dans l'url du navigateur (pensez juste à nettoyer un peu pour retirer le paramètre "Source" qui n'est rien d'autre que l'url de la page précédente :

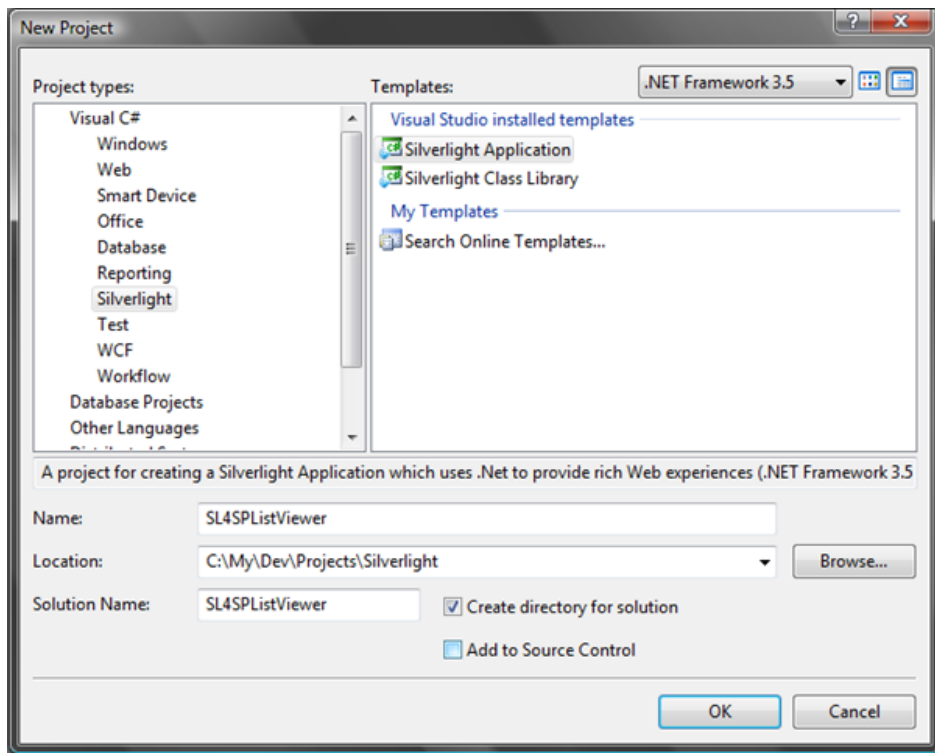
```
ViewEdit.aspx?List=%7B5F48DF89%2D51CE%2D4B2C%2DB19D%2D36C5D30D2AA6%7D&View=%7BDC3D5D6F%2DA351%
```

Nous voilà donc avec le nécessaire. Si vous entrez l'url ainsi générée dans votre navigateur, voici ce que vous obtenez :

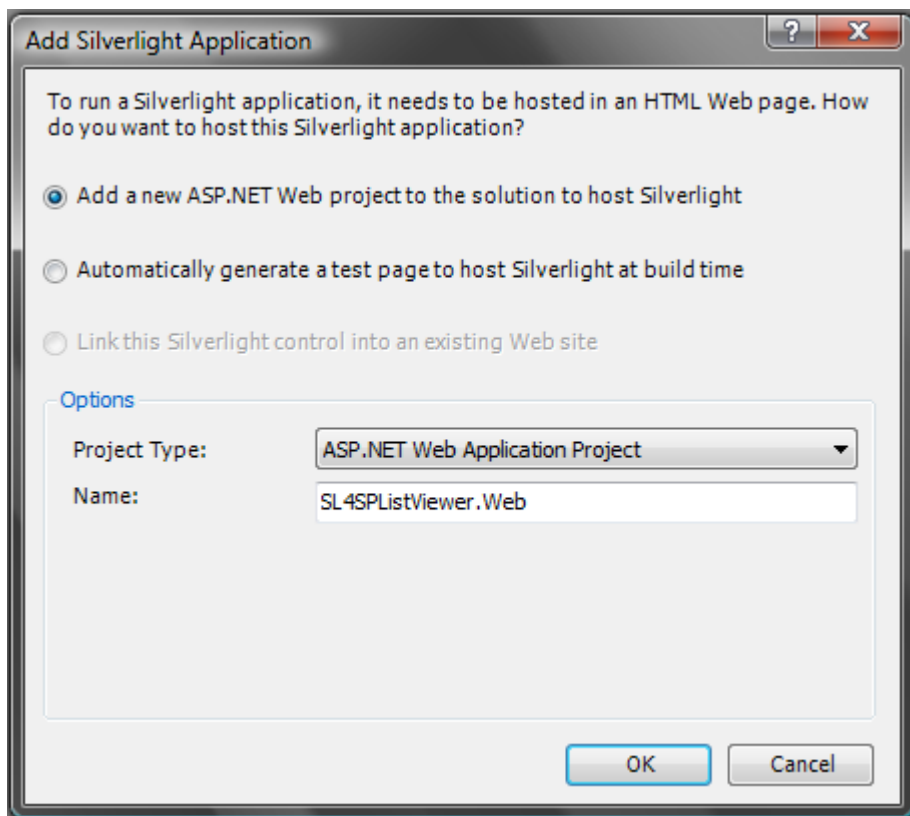
```
<xml xmlns:s="uuid:BDC6E3F0-6DA3-11d1-A2A3-00AA00C14882"
  xmlns:dt="uuid:C2F41010-65B3-11d1-A29F-00AA00C14882"
  xmlns:rs="urn:schemas-microsoft-com:rowset" xmlns:z="#"RowsetSchema">
- <s:Schema id="RowsetSchema">
- <s:ElementType name="row" content="eltOnly" rs:CommandTimeout="30">
- <s:AttributeType name="ows_DocIcon" rs:name="Type" rs:number="1">
  <s:datatype dt:type="string" dt:maxLength="512" />
  </s:AttributeType>
- <s:AttributeType name="ows_URLwMenu" rs:name="URL" rs:number="2">
  <s:datatype dt:type="string" dt:maxLength="512" />
  </s:AttributeType>
- <s:AttributeType name="ows_Comments" rs:name="Notes" rs:number="3">
  <s:datatype dt:type="string" dt:maxLength="1073741823" />
  </s:AttributeType>
  </s:ElementType>
</s:Schema>
- <rs:data>
  <z:row ows_URLwMenu="http://www.sharepointofview.fr/, SharePoint Of
    View" />
  <z:row ows_URLwMenu="http://www.codeplex.com, CodePlex" />
  <z:row ows_URLwMenu="http://blogs.msdn.com/sharepoint, MS SharePoint
    Team Blog" />
</rs:data>
</xml>
```

## Création du projet Silverlight sous Visual Studio 2008

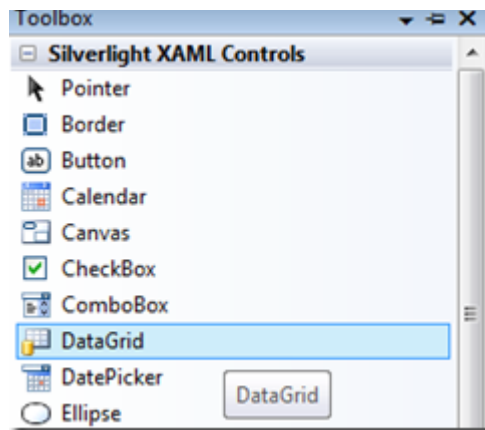
Créons maintenant un nouveau projet et choisissons « Silverlight Application » dans la section « Silverlight » :



Pensez à prendre l'option d'ajout d'un site ASP.Net :



Rajoutez dans le fichier *Page.xaml* un contrôle *DataGrid* (appelez le *ListItems*). Il nous servira à afficher les liens :



Définissez aussi 2 lignes dans le Grid ainsi qu'un *TextBlock* pour afficher des messages d'information en bas de l'application.

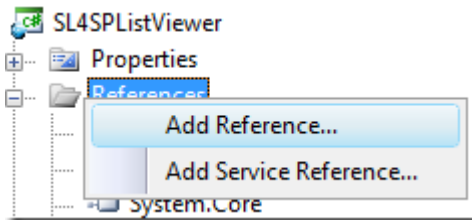
```
<UserControl x:Class="SL4SPListViewer.Page"
  xmlns:data="clr-namespace:System.Windows.Controls;assembly=System.Windows.Controls.Data"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Width="400" Height="300">
  <Grid x:Name="LayoutRoot" Background="White">
    <Grid.RowDefinitions>
      <RowDefinition Height="*"></RowDefinition>
      <RowDefinition Height="25"></RowDefinition>
    </Grid.RowDefinitions>
    <data:DataGrid x:Name="ListItems" Grid.Row="0" Width="Auto" Height="Auto">
    </data:DataGrid>
    <TextBlock x:Name="Message" Grid.Row="1" />
  </Grid>
</UserControl>
```

Passons maintenant dans le Code Behind pour coder l'appel RPC.

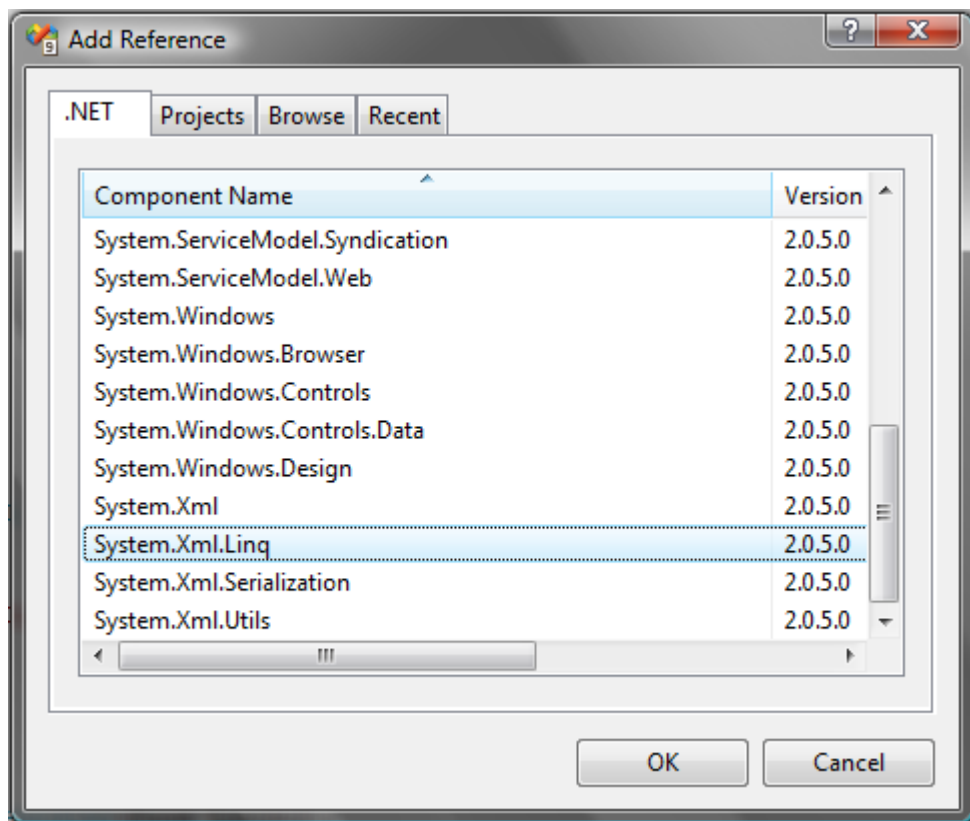
## Passage en Code Behind : requête RPC

Ajoutons d'abord une référence à *System.Xml.Linq* car nous allons récupérer les informations via LINQtoXML (le RPC fournissant un résultat sous forme XML, c'est quand même le plus adapté).

Clic droit sur « References », « Add Reference » :



Puis sélection du namespace :



Ajoutons le code suivant pour l'appel (utilisation de la classe WebClient) sans en mettre plus pour l'instant dans l'événement : nous allons d'abord vérifier que le contenu est correctement récupéré et nous afficherons un message en cas de succès ou d'échec.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;
using System.IO;
using System.Xml.Linq;

namespace SL4SPListViewer
{
    public partial class Page : UserControl
    {
        private const string baseRPCUrl =
            @"{0}/_vti_bin/owssvr.dll?cmd=Display&List={1}&View={2}&XMLDATA=TRUE";

        public Page()
        {
            InitializeComponent();
            this.Loaded += new RoutedEventHandler(Page_Loaded);
        }

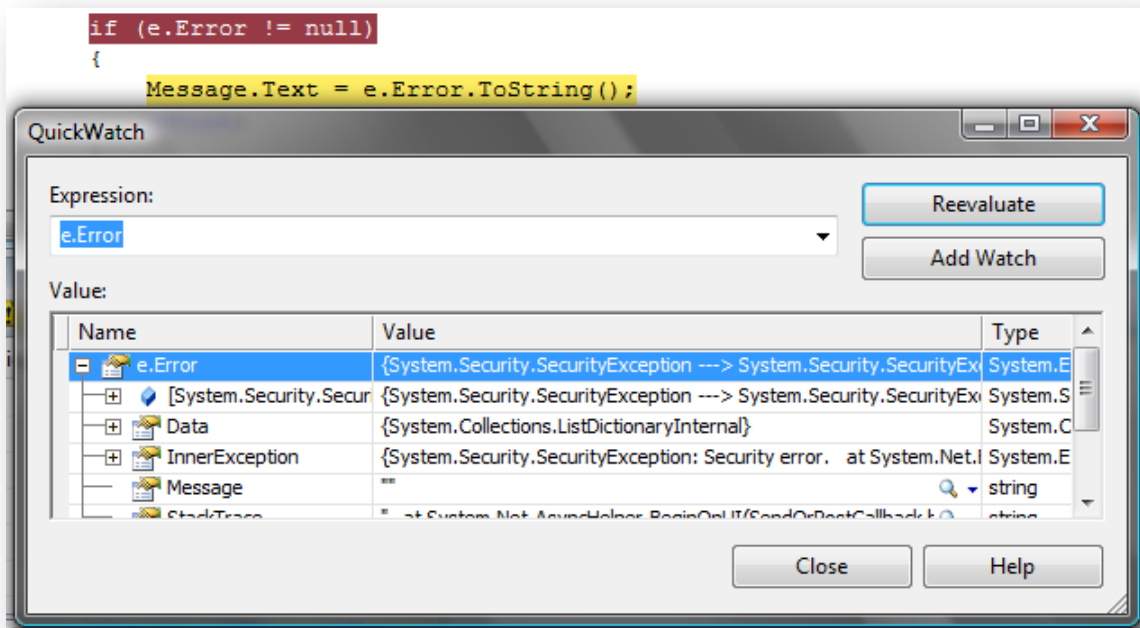
        void Page_Loaded(object sender, RoutedEventArgs e)
        {
            string webUrl = "http://dev.local";
            string listId = "%7B5F48DF89%2D51CE%2D4B2C%2DB19D%2D36C5D30D2AA6%7D";
            string viewId = "%7BDC3D5D6F%2DA351%2D4600%2D9B7B%2DFEBEEB825937%7D";
            // Generate my RPC url
            string myListRPC = string.Format(baseRPCUrl, webUrl, listId, viewId);

            WebClient wc = new WebClient();
            // Bind to event
            wc.OpenReadCompleted += new OpenReadCompletedEventHandler(wc_OpenReadCompleted);
            // Async call
            wc.OpenReadAsync(new Uri(myListRPC));
        }

        void wc_OpenReadCompleted(object sender, OpenReadCompletedEventArgs e)
        {
            if (e.Error != null)
            {
                Message.Text = e.Error.ToString();
            }
            else
            {
                Message.Text = "List successfully retrieved";
            }
        }
    }
}

```

En lançant en mode Debug, on remarque qu'une erreur se produit lors de la récupération de la page.



En effet, il n'est pas possible d'appeler la page, une exception liée à la sécurité est levée. C'est en fait une erreur classique et normale: mon site SharePoint n'est pas sur le même domaine que ma page de test (*dev.local* alors que ma page de test fonctionne en *localhost*). Et il n'est pas possible d'appeler de ressources distantes si le site les hébergeant ne le permet pas. Il va falloir définir un fichier d'autorisation pour réussir mon appel côté SharePoint.

## Faire fonctionner mon appel distant

Pour voir les appels HTTP effectués et ce qui peut poser problème, vous pouvez utiliser [Fiddler](#), un petit utilitaire permettant de sniffer les échanges réseaux entre votre poste client et les sites distants.

On remarque grâce à ce biais que l'application Silverlight a tenté d'accéder à 2 fichiers XML sur le site SharePoint : "clientaccesspolicy.xml" et "crossdomain.xml" mais sans succès (erreur 404):

The image shows the Fiddler - HTTP Debugging Proxy interface. The 'Web Sessions' table contains two entries, both with a red triangle icon indicating an error (404).

#	Result	Protocol	Host	URL	Body	Caching
0	404	HTTP	dev.local	/clientaccesspolicy.xml	13	private,m...
1	404	HTTP	dev.local	/crossdomain.xml	13	private,m...

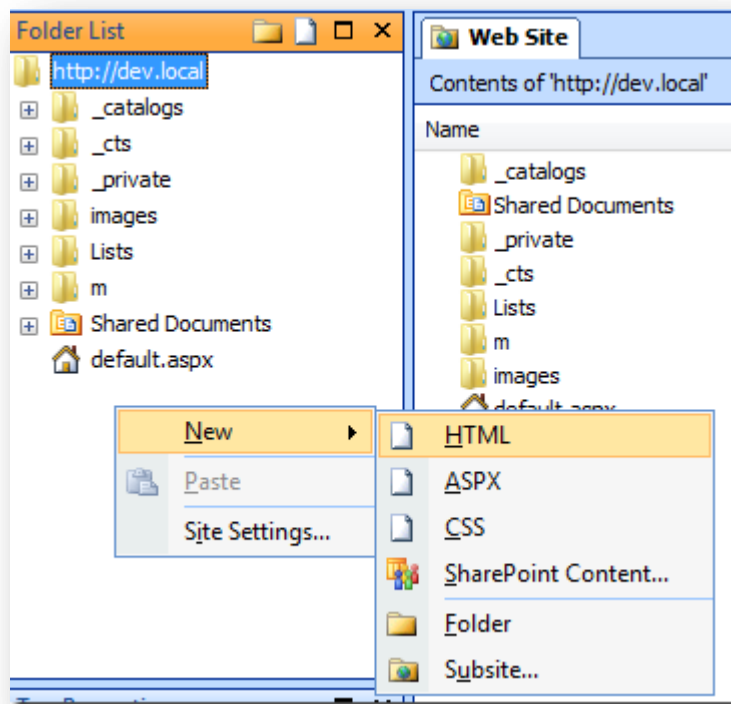
Le fichier "ClientAccessPolicy.xml" permet de définir les services disponibles et qui peut les consommer. Donc point de salut si vous désirez consommer des services sur des sites externes, il faudra qu'ils définissent de leur côté ce fichier.

Vous trouverez toutes les informations à ce sujet sur le MSDN :

- [Network Security Access Restrictions in Silverlight 2](#)
- [Making a Service Available Across Domain Boundaries](#)

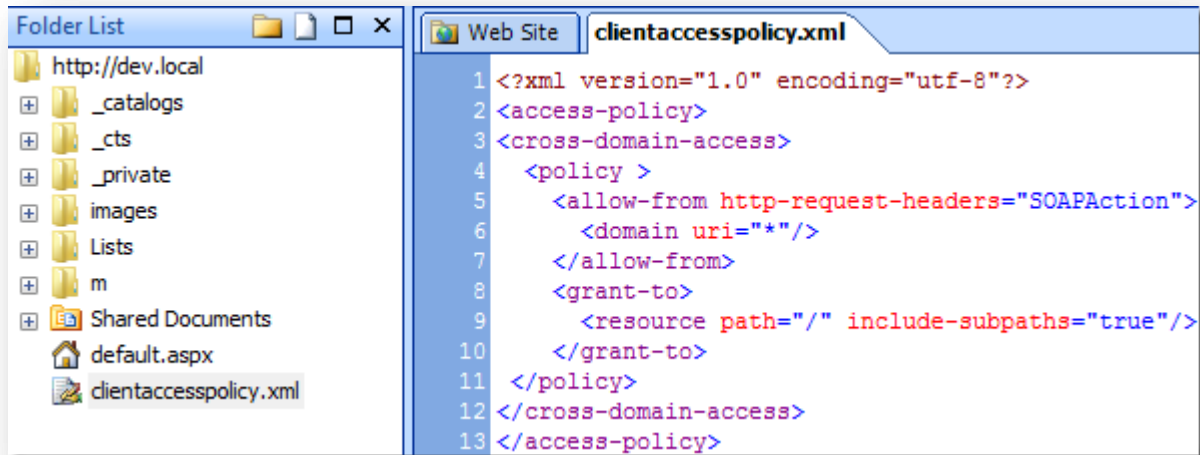
Faisons simple et créons ce fichier XML avec SharePoint Designer 2007 dans le site situé à la racine de votre application (par exemple ici *http://dev.local*).

Connectez-vous au site SharePoint, et rajoutez un fichier HTML à la racine :



Renommez-le fichier en "**ClientAccessPolicy.xml**" et remplacez son contenu par le code XML suivant :

```
<?xml version="1.0" encoding="utf-8"?>
<access-policy>
<cross-domain-access>
  <policy >
    <allow-from http-request-headers="SOAPAction">
      <domain uri="*" />
    </allow-from>
    <grant-to>
      <resource path="/" include-subpaths="true" />
    </grant-to>
  </policy>
</cross-domain-access>
</access-policy>
```



J'ai volontairement fait simple avec une définition de la sécurité assez proche de l'Open Bar : accès en mode SOAP à n'importe quel service/url depuis n'importe quel domaine.

Sauvegardez le tout et relancez votre application. Ce coup-ci, ça marche !

List successfully retrieved

Reste maintenant à binder nos données XML au DataGrid.

## Binding des données au DataGrid

Pour cela nous allons créer une classe "Link" représentant le lien et générons en une collection via une belle requête LINQ.

Ajoutez une nouvelle classe dans le projet (fichier *Link.cs*) :

```
namespace SL4SPListViewer
{
    public class Link
    {
        public string Title { get; set; }
        public string Url { get; set; }
        public string Comments { get; set; }
    }
}
```

Dans le code behind de la page, modifiez le code de l'événement "OpenReadCompleted" pour requêter le contenu depuis un *XDocument* rempli grâce à la réponse à l'appel RPC.

```

void wc_OpenReadCompleted(object sender, OpenReadCompletedEventArgs e)
{
    if (e.Error != null)
    {
        Message.Text = e.Error.ToString();
    }
    else
    {
        Message.Text = "List successfully retrieved";

        using (Stream s = e.Result)
        {
            XDocument rpcXmlDoc = XDocument.Load(s);
            if (rpcXmlDoc.Root != null)
            {
                XNamespace nrs = "urn:schemas-microsoft-com:rowset";
                XNamespace nz = "#RowsetSchema";

                var links =
                from row in rpcXmlDoc.Root.Descendants(nrs + "data").Descendants(nz + "row")
                select new Link
                {
                    Title = GetLinkTitle((string)row.Attribute("ows_URLwMenu")),
                    Url = GetLinkUrl((string)row.Attribute("ows_URLwMenu")),
                    Comments = (string)row.Attribute("ows_Comments")
                };

                ListItem.SelectedIndex = -1;
                ListItem.ItemsSource = links;

                Message.Text +=
                    string.Format("({0} link{1})", links.Count(), links.Count() > 1 ? "s" : "");
            }
            else
            {
                Message.Text += " (Empty list)";
            }
        }
    }
}

string GetLinkUrl(string linkUrl)
{
    if (!string.IsNullOrEmpty(linkUrl))
    {
        int i = linkUrl.IndexOf(", ");
        if (i > 0) linkUrl = linkUrl.Substring(0, i);
    }
    return linkUrl;
}

string GetLinkTitle(string linkUrl)
{
    if (!string.IsNullOrEmpty(linkUrl))
    {
        int i = linkUrl.IndexOf(", ");
        if (i > 0) linkUrl = linkUrl.Substring(i + 2);
    }
    return linkUrl;
}

```

1. Load XML

2. Define Namespaces

3. LINQ Query

4. Bind to DataGrid

Cleanup Title and URL

Dans le détail :

1. Chargement du flux XML dans le XDocument
2. Définition des namespaces XML afin d'accéder aux noeuds <z:row />
3. Création de la requête LINQ en générant les liens depuis les attributs des noeuds récupérés (j'ai ajouté 2 petites méthodes Helper pour découper proprement l'url puisque SharePoint stocke les valeurs sous le format « *http://url, titre* »).
4. Pour finir, binding de nos liens avec le DataGrid

Et voici le résultat obtenu :

Title	Url	Comme
SharePoint Of View	<a href="http://www.sharepointofview.fr/">http://www.sharepointofview.fr/</a>	
CodePlex	<a href="http://www.codeplex.com">http://www.codeplex.com</a>	
MS SharePoint Team Blog	<a href="http://blogs.msdn.com/sharepoint">http://blogs.msdn.com/sharepoint</a>	

List successfully retrieved (3 links)

Pas super sexy, mais fonctionnel !

En modifiant un peu le XAML (Page.Xaml) pour remplacer le DataGrid par une ListBox contenant un lien hypertexte et un champs de commentaires :

```
<ListBox x:Name="ListItems" Grid.Row="0">
  <ListBox.ItemTemplate>
    <DataTemplate>
      <StackPanel Orientation="Vertical">
        <HyperlinkButton x:Name="Link" NavigateUri="{Binding Url}"
          Content="{Binding Title}" TargetName="_blank" />
        <TextBlock x:Name="Comments" Text="{Binding Comments}" />
      </StackPanel>
    </DataTemplate>
  </ListBox.ItemTemplate>
</ListBox>
```

On obtient un résultat directement exploitable qui permet d'avoir les liens cliquables:

<a href="#">SharePoint Of View</a>
<a href="#">CodePlex</a>
<a href="#">MS SharePoint Team Blog</a>

List successfully retrieved (3 links)

## Affichage du contrôle dans SharePoint

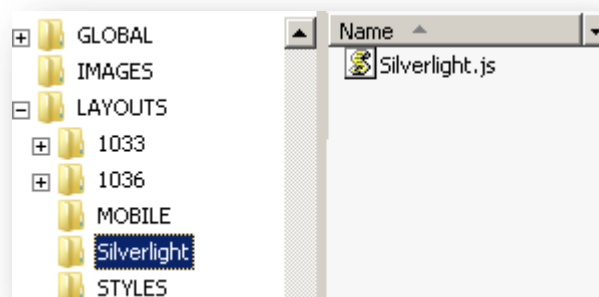
Finissons ce projet en incluant le contrôle Silverlight au sein d'un site SharePoint.

Cela va nous permettre de voir les quelques modifications à apporter pour que cela fonctionne correctement.

### Ajout du fichier Silverlight.js

Pour initialiser correctement le contrôle Silverlight, il nous faut le fichier Javascript fourni par le Framework (celui-ci contient le nécessaire). Etant donné que ce fichier va être réutilisé par d'autres contrôles, autant le centraliser en le plaçant dans un sous-répertoire « Silverlight » dans « 12\TEMPLATE\LAYOUTS ».

**Remarque :** Bien sûr, je zappe ici le packaging sous forme de solution, mais c'est le passage **obligé** (et propre) pour déployer le fichier proprement sur vos plateformes ! Vous trouverez à la fin de l'article le lien vers la solution déployant le fichier *Silverlight.js*.



### Mise à disposition du XAP

Plaçons ensuite le fichier XAP (*SL4SPListViewer.xap*) situé dans le répertoire « bin/debug » de notre projet dans une bibliothèque de documents (ici appelée *Silverlight* mais le nom importe peu) :

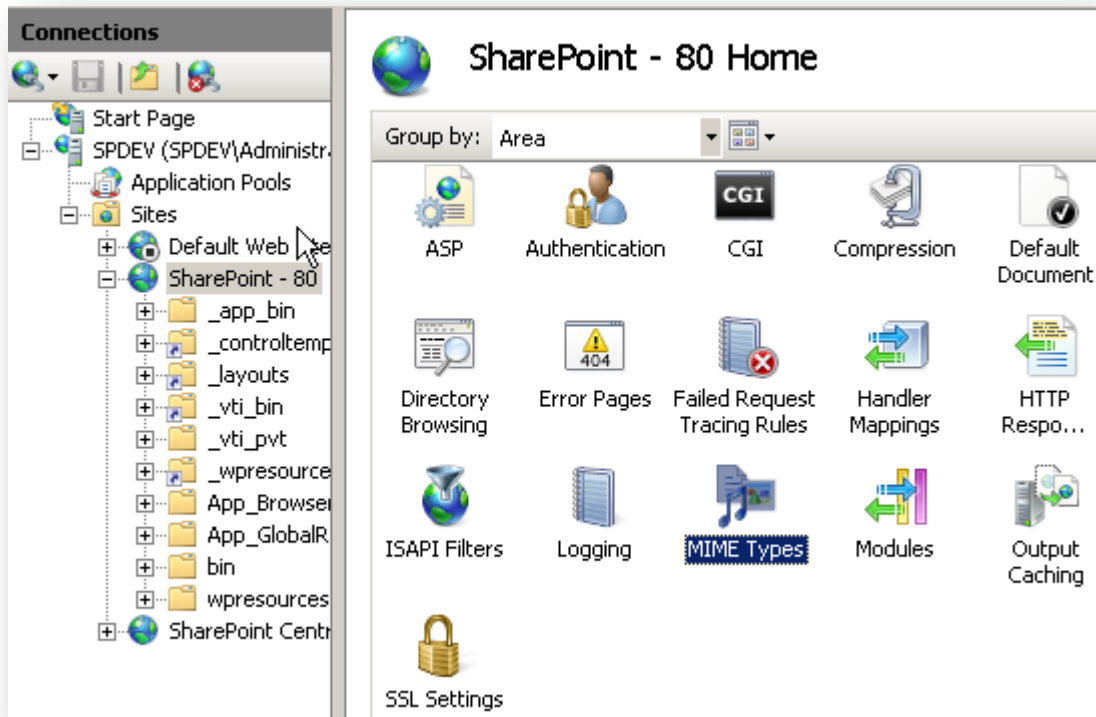


**Remarque :** vous n'êtes pas du tout obligé de mettre le XAP dans une bibliothèque de documents, cela dépendra pour beaucoup des facilités de déploiement et de la structure de votre projet, mais la bibliothèque vous permet de gérer le versioning et propose une interface de gestion facilitée.

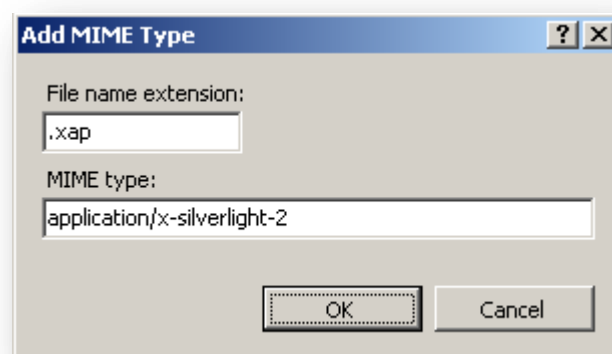
## Ajout du type MIME « .xap »

Mais ce n'est pas suffisant car il faut que votre serveur IIS reconnaisse le nouveau type de fichier et il faut ajouter le type MIME « .xap » à notre application IIS. Je vous donne ici l'explication pour Windows Server 2008 mais c'est la même logique pour Windows Server 2003.

Rendez-vous sur l'application web hébergeant votre site SharePoint, dans la vue « Feature », et sélectionnez « MIME Types »



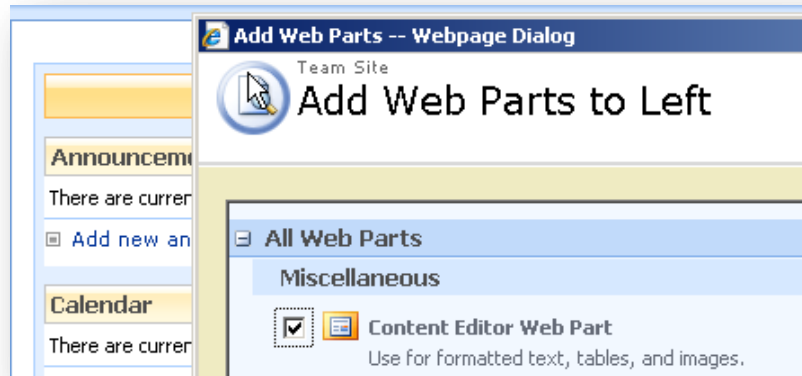
Cliquez sur « Add » et entrez les informations suivantes :



**Remarque :** attention, si vous l'aviez fait pour la version bêta, le type a changé de nom, pensez à le mettre à jour !

## Affichage du contrôle dans une WebPart

Nous allons faire simple et utiliser la WebPart d'édition de contenu (**Content Editor WebPart**) pour appeler notre application Silverlight. Rendez-vous sur la page d'accueil du site en mode édition et ajoutez la WebPart:



Dans les propriétés de celle-ci, cliquez sur le bouton permettant l'édition du code source et collez le contenu suivant :

```
<script type="text/javascript" src="/_layouts/silverlight/Silverlight.js"></script>
<script type="text/javascript">
    function onSilverlightError(sender, args) {
        var appSource = "";
        if (sender != null && sender != 0) {
            appSource = sender.getHost().Source;
        }
        var errorType = args.ErrorType;
        var iErrorCode = args.ErrorCode;

        var errMsg = "Unhandled Error in Silverlight 2 Application " + appSource + "\n" ;
        errMsg += "Code: " + iErrorCode + " \n";
        errMsg += "Category: " + errorType + " \n";
        errMsg += "Message: " + args.ErrorMessage + " \n";
        if (errorType == "ParserError")
        {
            errMsg += "File: " + args.xamlFile + " \n";
            errMsg += "Line: " + args.lineNumber + " \n";
            errMsg += "Position: " + args.charPosition + " \n";
        }
        else if (errorType == "RuntimeError")
        {
            if (args.lineNumber != 0)
            {
                errMsg += "Line: " + args.lineNumber + " \n";
            }
        }
    }
}
```

```

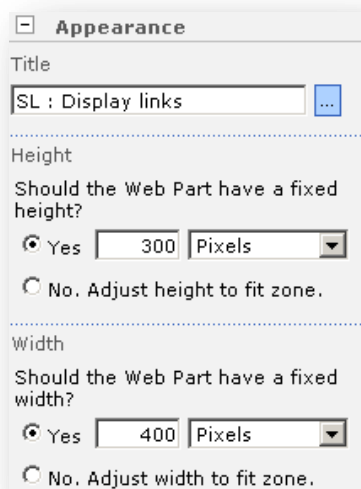
errMsg += "Position: " + args.charPosition + " \n";
}
errMsg += "MethodName: " + args.methodName + " \n";
}
throw new Error(errMsg);
}
</script>
<div id="silverlightControlHost">
  <object data="data:application/x-silverlight-2," type="application/x-silverlight-2"
width="100%" height="100%">
    <param name="source" value="/Silverlight/SL4SPListViewer.xap"/>
    <param name="onerror" value="onSilverlightError" />
    <param name="background" value="white" />
    <param name="minRuntimeVersion" value="2.0.31005.0" />
    <param name="autoUpgrade" value="true" />
    <a href="http://go.microsoft.com/fwlink/?LinkID=124807" style="text-
decoration: none;">

</a>
  </object>
  <iframe style='visibility:hidden;height:0;width:0;border:0px'></iframe>
</div>

```

Ce code HTML correspond à des balises classiques HTML. Il vous faudra l'adapter selon l'emplacement de votre fichier Javascript et du fichier XAP.

Malgré cela, notre WebPart semble être vide. Il s'agit juste qu'un petit problème de style et d'encapsulation dans les balises générées par notre WebPart. Modifiez-la et changez la largeur et la hauteur dans la section « Apparence » pour qu'elles correspondent à celles de notre contrôle :



Et voilà !

The screenshot shows a SharePoint Team Site interface. At the top left, there is a logo with three colored circles (red, green, blue) followed by the text "Team Site". Below this is a "Home" button. A vertical navigation pane on the left contains several sections: "View All Site Content", "Documents" (with sub-items "Shared Documents" and "Silverlight"), "Lists" (with sub-items "Calendar" and "Tasks"), "Discussions" (with sub-item "Team Discussion"), "Sites", "People and Groups", and "Recycle Bin" (with a trash icon). The main content area is titled "SL : Display links" and contains a list of three links: "SharePoint Of View", "CodePlex", and "MS SharePoint Team Blog". A mouse cursor is visible over the list. At the bottom of the main content area, a status message reads "List successfully retrieved (3 links)".

## Conclusion

J'espère qu'au travers de ce tutoriel vous aurez suivi l'ensemble des petites problématiques de connexion entre Silverlight et SharePoint afin d'en lire le contenu, ainsi que l'intégration de vos applications Silverlight au sein de vos sites SharePoint.

Deux petites remarques pour finir :

- Pour utiliser les requêtes RPC, l'utilisateur doit être authentifié et reconnu (au moins lecteur sur le site) sinon le document XML récupéré via les RPC sera vide. Ainsi, dans le cadre d'accès anonyme cela pourra vous poser quelques problèmes.
- En bêta, le contrôle **HyperLinkButton** demandait obligatoirement une Uri pour le binding sur sa propriété *NavigateUri*, il semble que depuis la RTM une chaîne de caractères suffise. Dans le cas contraire, il vous faudra passer par un [Converter](#).

### Liens utiles :

- [Tutoriel Silverlight de Scott Guthrie : développement d'un client Digg](#)
- [Téléchargement du Service Pack 1 pour Visual Studio 2008](#)
- [Microsoft Silverlight Tools for Visual Studio 2008 SP1](#)
- Guide de démarrage Silverlight <http://silverlight.net/GetStarted/>
- [Silverlight Toolkit](#) sur CodePlex
- [Windows SharePoint Services RPC Methods](#)
- [Office Online](#)
- [Club SPS MOSS FRANCE\(FR\)](#)

### Téléchargements :

- [Solution SharePoint installant le fichier silverlight.js](#)
- [Code source du projet](#)

En vous souhaitant de bons projets autour de SharePoint et de Silverlight !

**Gaëtan Bouveret** (alias Gat)

Expert Consultant et Formateur SharePoint chez [Winwise](#)

Blog : [myPointOfView](#)

Membre de la [SoV Team](#)