

Ecrire un webpart pour Sharepoint 2007 avec smartpart

par Ludovic Lefort ([Site web](#)) ([Blog](#))

Date de publication : 24/08/2007

Dernière mise à jour :

Ce document explique pas à pas comment créer et déployer un webpart custom avec Smartpart

- 0 - Introduction
- 1 - Installation de Smartpart
- 2 - création de notre premier web part
- 3 - Déploiement du user control dans Sharepoint
- 4 - Ajouter une propriété custom
- 5 - Implémentation de l'interface IAdvancedUserControl
 - 5.1 - Récupérer le contexte sharepoint du smartpart
 - 5.2 - Créer son propre toolpanel pour la configuration du webpart
- 6 - Faire communiquer deux smartparts
 - 6.1 - Le fournisseur
 - 6.2 - Le consommateur
- 7 - Conclusion
- 8 - Sources de l'exemple

0 - Introduction

Smartpart est une feature pour Sharepoint 2007 permettant d'intégrer des custom controls .net 2.0 directement dans une webpart page.

L'avantage est évident : plus besoin de créer ses contrôles à la main, on peut utiliser le designer de visual studio 2005.

Bien que vos contrôles peuvent être testés hors sharepoint il ne faut pas oublier qu'ils y seront déployés. Si vous utilisez par exemple des contrôles tels que le "SqlConnection" les ConnectionStrings devront être recopiés dans le web.config de l'application Sharepoint.

Pour plus d'infos je vous invite à jeter un oeil sur le blog de **Jan Tielens**.

1 - Installation de Smartpart

Pour cet article j'ai décidé d'utiliser la "sous-version" de SmartPart appelée "ReturnOfSmartPage" version 1.1, elle permet entre autres l'utilisation des contrôles AJAX.

Je vous invite à la télécharger [ici](#).

- 1 Décompressez le fichier zip dans le répertoire C:\ReturnOfSmartPartInstall de votre serveur Sharepoint
- 2 Ouvrez un command prompt (cmd.exe)
- 3 Déplacez vous dans le répertoire "C:\Program Files\Common Files\Microsoft Shared\web server extensions\12\BIN"
- 4 Exécutez : stsadm.exe -o addSolution -filename c:\returnofsmartpartinstall\ReturnOfSmartPartv1_1.wsp
- 5 Exécutez : stsadm.exe -o deploySolution -name ReturnOfSmartPartv1_1.wsp -allcontenturls -local -allowGacDeployment -force
- 6 Il faut maintenant activer la feature pour votre site collection. Pour cela, allez dans les site settings de votre application
- 7 Dans le menu Site Collection Administration > Site collection features (*figure 1*)
- 8 Repérer la ligne "Return of the SmartPart v1.1" dans la liste et cliquez sur le bouton "Activate" (*figure 2*)
- 9 Créons maintenant le répertoire qui contiendra nos users controls dans notre application sharepoint (par exemple : "C:\inetpub\wwwroot\wss\VirtualDirectories\80"). Il doit **impérativement** s'appeler : UserControls.
- 10 Si tout s'est bien passé vous devriez être capable d'ajouter un web part de type "SmartPart" sur une webpart page (*figure 3*)

Figure 1:

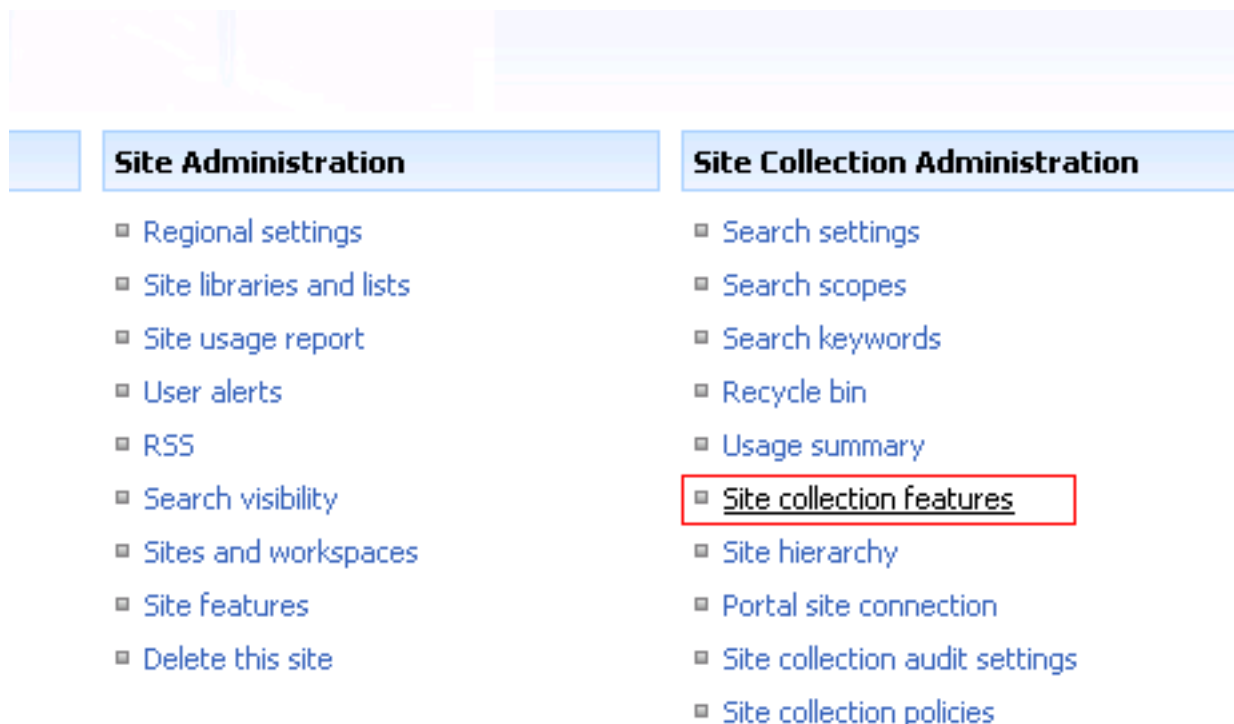


Figure 2:

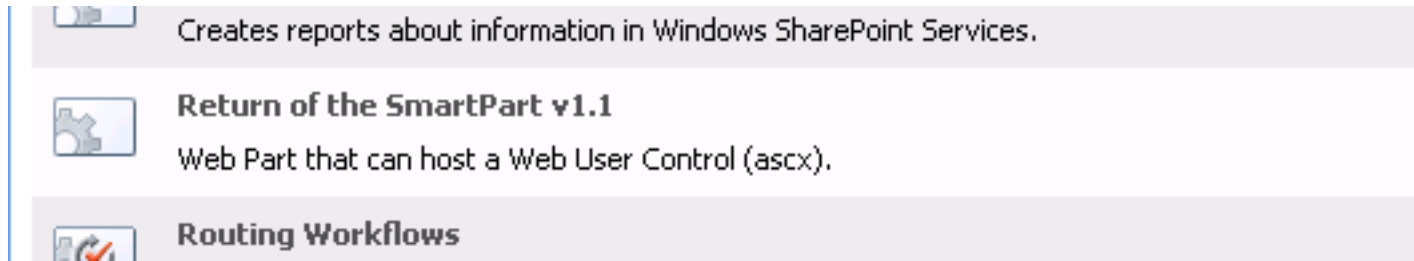



Figure 3:

Left









 Add a Web Part

Add Web Parts -- Web Page Dialog



LLESandBox

Add Web Parts to Left

-  **GepirSearch**
-  **Image Web Part**
Use to display pictures and photos.
-  **Page Viewer Web Part**
Use to display linked content, such as files, folders, or Web pages. The linked content on the Web Part Page.
-  **Relevant Documents**
Use this webpart to display documents that are relevant to the current user.
-  **SayHello**
-  **Site Users**
Use the Site Users Web Part to see a list of the site users and their online status.
-  **SmartPart**
A web part that can host any ASP.NET Web User Control (ascx).
-  **SmartPart with AJAX**
A web part that can host any ASP.NET Web User Control (ascx) and supports the ASP

2 - création de notre premier web part

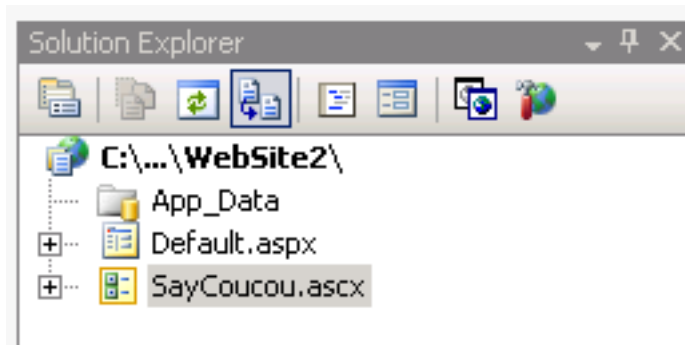
Maintenant que Smart est correctement installé sur notre serveur nous allons créer notre premier webpart custom.

Faisons simple pour commencer :

- un textbox
- un bouton
- un label qui affiche "coucou" et le texte du textbox

Ouvrons Visual studio 2005 et créons un nouveau site web.

Ajoutons un nouveau Web User Control que nous allons appeler SayCoucou.ascx :



Voici le code source de notre contrôle :

```
<%@ Control Language="C#" AutoEventWireup="true" CodeFile="SayCoucou.ascx.cs" Inherits="SayCoucou"
%>
<asp:TextBox ID="txtInput" runat="server"></asp:TextBox>
<asp:Button ID="btnAction" runat="server" Text="Clic !" /><br />
<asp:Label ID="lblResult" runat="server"></asp:Label>
```

Et le code behind sur le click du bouton :

```
protected void btnAction_Click(object sender, EventArgs e)
{
    this.lblResult.Text = "Coucou " + txtInput.Text;
}
```

Vous pouvez tester le contrôle hors sharepoint en le plaçant sur votre page default.aspx par exemple.

3 - Déploiement du user control dans Sharepoint

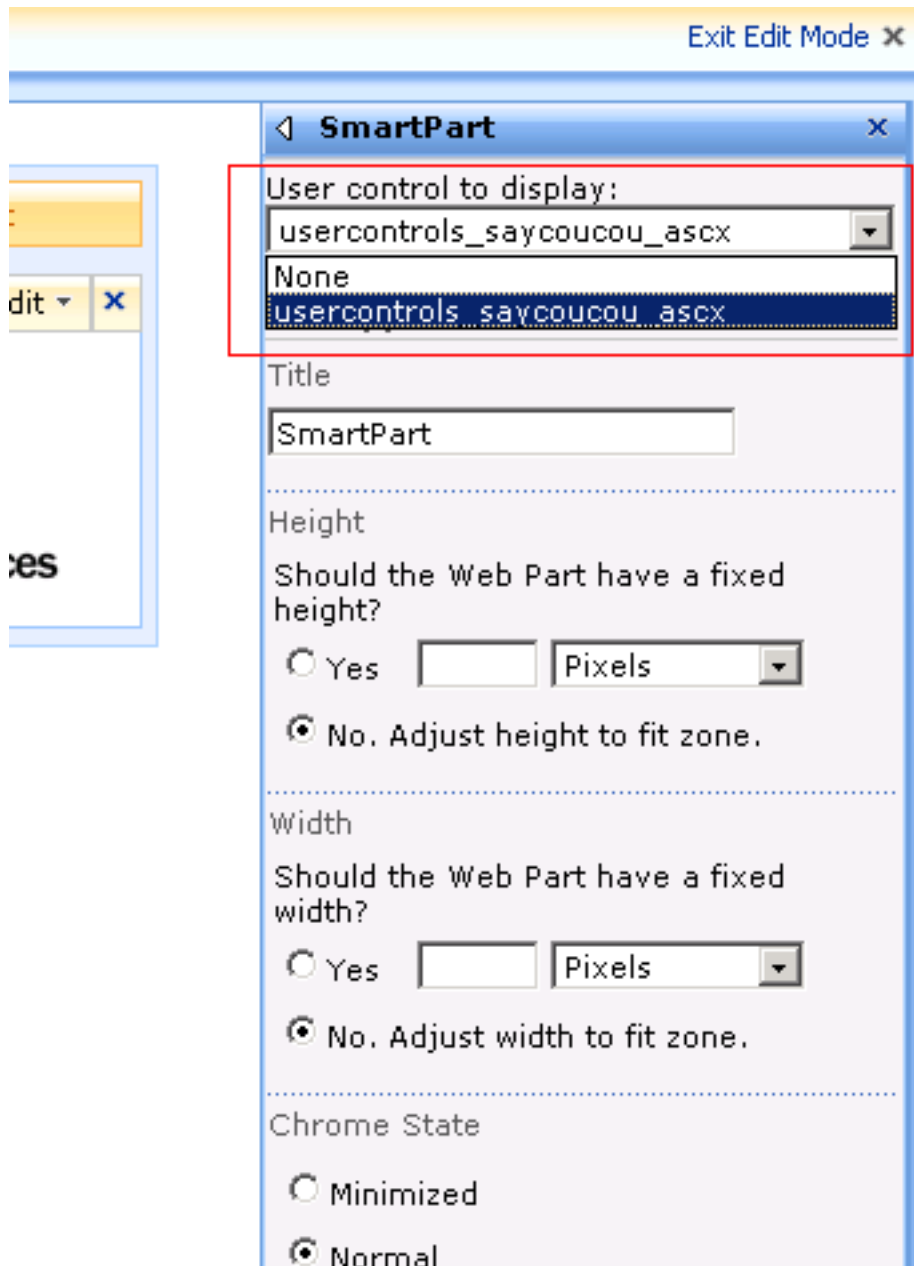
Voyons comment déployer ce user control sur notre serveur Sharepoint

Rien de plus facile ! il vous suffit de copier les fichiers : `sayCoucou.ascx` et `sayCoucou.ascx.cs` dans le répertoire UserControls que l'on a créé au premier chapitre.

Essayons d'ajouter notre contrôle sur une webpart page de sharepoint.

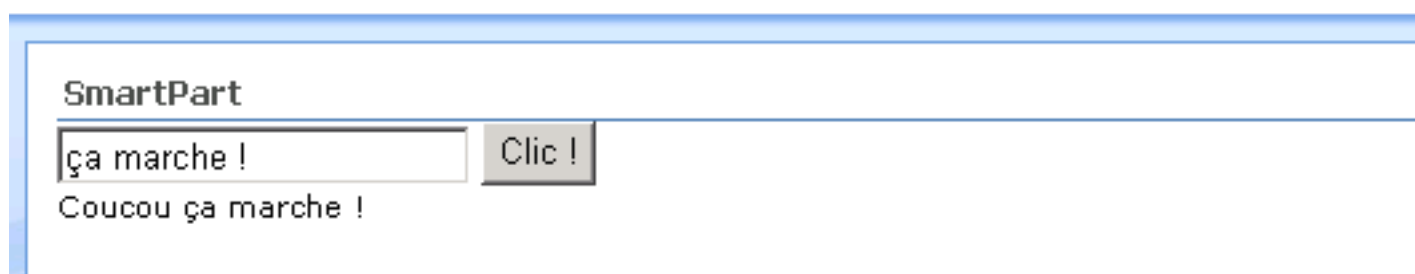
Pour ce faire la page doit contenir un SmartPart (il suffit de l'ajouter sur la page comme n'importe quel autre web part).

Ensuite dans le toolpanel du web part il faut spécifier le contrôle que l'on veut afficher. La liste affichée est la liste des contrôles se trouvant dans le répertoire UserControls :



Une fois le contrôle choisi dans la liste, cliquez sur "APPLY", "OK" et quittez le mode édition de la page

Voilà votre contrôle est intégré à Sharepoint et est customisable comme tout autre webpart.



4 - Ajouter une propriété custom

Les webparts de sharepoint offrent de base de nombreuses propriétés de customisation mais il est parfois nécessaire de pouvoir en définir d'autres.

Cela se fait grâce à des attributs dans le code du contrôle.

Pour notre exemple nous allons permettre à l'administrateur du site de définir un suffixe qui viendra se concaténer au texte du textbox.

Voici le nouveau code behind pour notre contrôle :

```
private string _suffix;

[Browsable(true)]
[Description("Suffix to display")]
[WebPartStorage(Storage.Shared)]
public string Suffix
{
    get
    {
        return _suffix;
    }
    set
    {
        _suffix = value;
    }
}

protected void btnAction_Click(object sender, EventArgs e)
{
    this.lblResult.Text = "Coucou " + txtInput.Text + " " + _suffix;
}
```

Trois espaces de nom ont du être ajoutés :

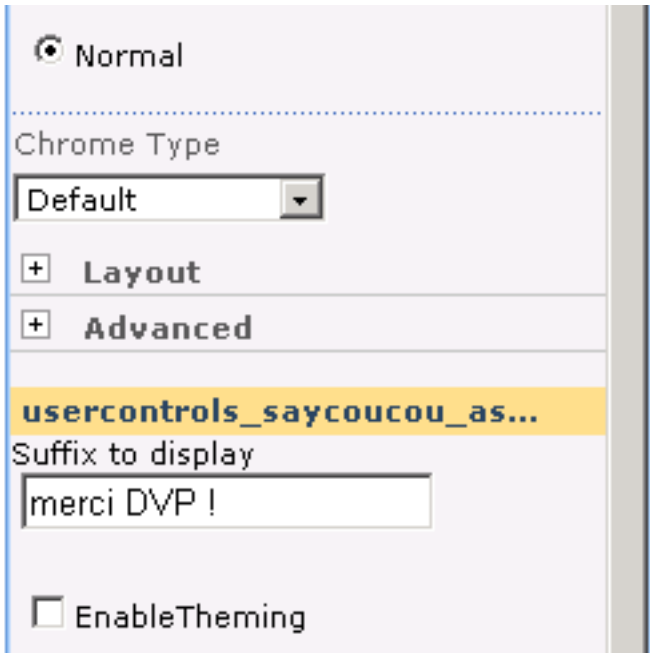
```
using System.Xml.Serialization;
using System.ComponentModel;
using Microsoft.SharePoint.WebPartPages;
```

Pour ajouter le namespace **Microsoft.SharePoint.WebPartPages**, vous devez ajouter une référence vers l'API de sharepoint : "C:\Program Files\Common Files\Microsoft Shared\web server extensions\12\ISAPI\Microsoft.SharePoint.dll" Comme vous pouvez le constater j'ai ajouté une variable string **_suffix** et une propriété **Suffix**

Pour qu'elle soit affichée dans le toolpanel de sharepoint je dois lui ajouter quelques attributs:

- [Browsable(true)] : spécifie que la propriété doit être affichée.
- [Description("Suffix to display")] : Texte du label dans le toolpanel.
- [WebPartStorage(Storage.Shared)] : la valeur du paramètre sera stockée pour tous les utilisateurs.

Vous pouvez voir votre propriété dans le toolpanel de sharepoint :



Les propriétés peuvent également être de type "Boolean" et "enum" (liste statique) qui afficheront respectivement un checkbox ou une combobox dans le toolpanel

5 - Implémentation de l'interface IAdvancedUserControl

Pour implémenter cette interface vous devez ajouter l'espace de nom "SmartPart":

```
using SmartPart;
```

Et ajouter une référence vers la librairie "ReturnOfSmartPart.dll" qui se trouve dans le répertoire d'installation de SmartPart (C:\returnOfSmartPartInstall)

Implémenter cette interface va nous permettre deux choses :

- Récupérer le contexte sharepoint du smartpart
- Créer son propre toolpanel pour la configuration du webpart

5.1 - Récupérer le contexte sharepoint du smartpart

Cela permet par exemple de récupérer le titre donné au webpart, des informations sur la page qui le contient, sur l'utilisateur, ...

Pour faire cela il est nécessaire d'implémenter la méthode **SetContext**

```
private SmartPart.SmartPart _webpart;  
void IAdvancedUserControl.SetContext(SmartPart.SmartPart webpart)  
{  
    _webpart = webpart;  
}
```


Dans ce code je définis un attribut pour mon contrôle et je lui affecte la valeur reçue par la méthode setContext().

De cette manière je peux à tout moment récupérer des infos de sharepoint au travers de l'objet _webpart.

5.2 - Créer son propre toolpanel pour la configuration du webpart

Implémenter la méthode **GetCustomToolParts()** permet de dessiner son propre toolpart pour sharepoint.

Je n'ai pas encore vraiment creusé cette piste, je ne vais donc pas me risquer à en parler dans cet article.

Si vous voulez en savoir plus je vous renvoie vers le blog de  **Ryan Bedell**.

6 - Faire communiquer deux smartparts

Il est possible d'utiliser le système de connexion entre web part de Sharepoint avec les smartPart.

Pour cela il nous faut un webpart "Fournisseur" et un autre "Consommateur".

6.1 - Le fournisseur

C'est ce smartpart qui va envoyer des infos vers l'extérieur, il doit implémenter l'interface `IConnectionProviderControl` du namespace `SmartPart`.

Dans cet exemple notre webpart va envoyer le texte de sa textbox.

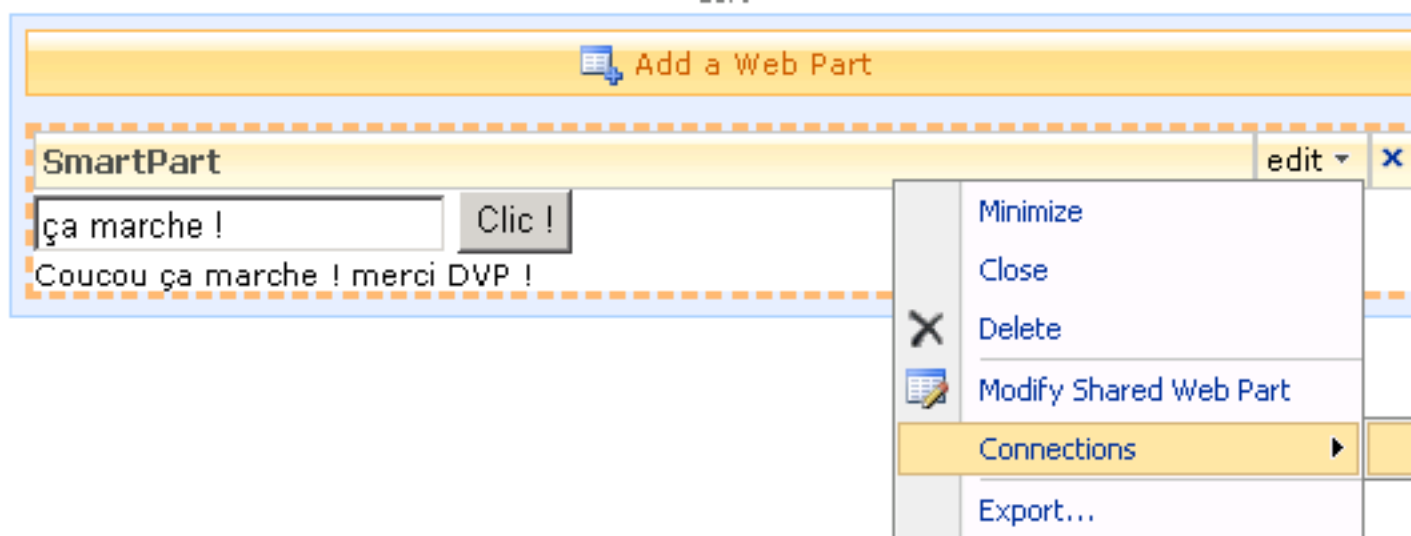
```
object IConnectionProviderControl.GetProviderData()  
{  
    return txtInput.Text;  
}  
  
string IConnectionProviderControl.ProviderMenuLabel  
{  
    get  
    {  
        return "TextBoxValue";  
    }  
}
```

Nous avons implémenté une méthode et une propriété.

La méthode `GetProviderData()` permet de définir les données sortantes dans ce cas il s'agit du contenu de la textbox.

La propriété `ProviderMenuLabel` défini le texte qui sera affiché dans le menu contextuel de Sharepoint.

Si nous déployons le contrôle sur notre serveur sharepoint nous pouvons voir dans le menu connections :



Le menu "TextBoxValue" apparait mais est grisé parcequ'il n'y a pas de contrôle sur la page pouvant recevoir sa valeur.

6.2 - Le consommateur

Nous allons créer un deuxième user control encore plus simple que l'on va appeler : **SayCoucouConsumer.ascx** avec juste un label

Il va se contenter d'afficher les données reçues par notre premier smartpart.

Pour écrire un consommateur il faut implémenter l'interface **IConnectionConsumerControl** du namespace SmartPart.

Voici le code de ce contrôle :

```
string IConnectionConsumerControl.ConsumerMenuLabel
{
    get
    {
        return "TextToDisplay";
    }
}

void IConnectionConsumerControl.SetConsumerData(object data)
{
    lblDisplay.Text = data.ToString();
}
```

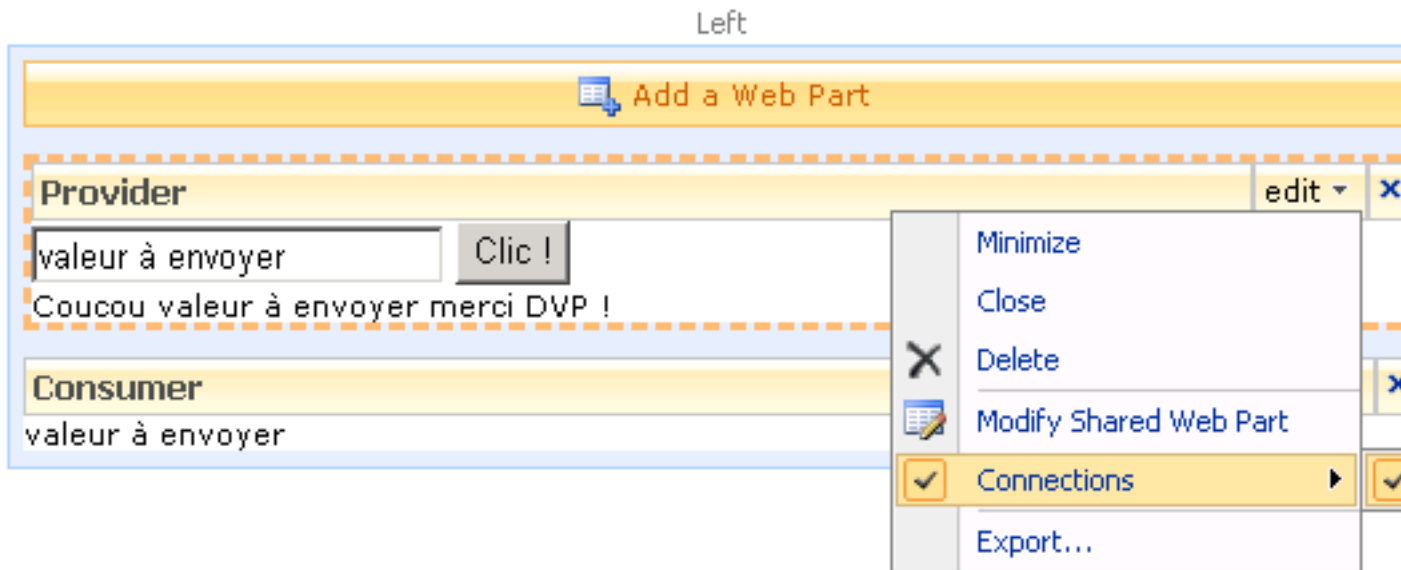
Comme pour le provider nous avons implémenté une propriété qui définit le label du menu contextuel de Sharepoint.

Mais également une méthode qui traite les données reçues de l'extérieur.

Nous pouvons maintenant compiler et déployer ce nouveau contrôle dans le répertoire **UserControls**.

Ajoutons maintenant un deuxième smartpart sur notre webpart page et choisissons SayCoucouConsumer comme contrôle.

Il nous reste à définir la connexion entre les deux smartpart :



Voilà vos smartpart sont connectés. Entrez un texte dans la textbox du premier et cliquez sur le bouton.

Normalement le texte devrait être recopié dans le deuxième smartpart.

7 - Conclusion

SmartPart est une feature gratuite qui vous permettra d'économiser énormément de temps lors de vos développements de webpart custom pour Sharepoint 2007.

8 - Sources de l'exemple

Téléchargez les **sources** de cet exemple

Merci à **Stephane Eyskens** pour sa relecture.

