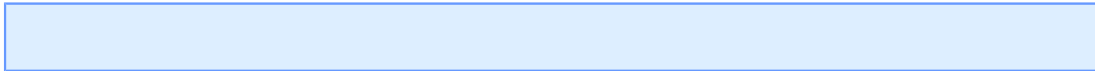


# Intégration du caml dans une WebPart

par Solczynski Ludovic (<http://lsolczynski.developpez.com>) (Blog)

Date de publication : 21 October 2010

Dernière mise à jour :



MCours.com

I - Introduction.....	3
II - Contexte.....	3
III - Mise en place de l'environnement.....	3
III-A - Création du projet.....	3
III-B - Création d'une liste SharePoint de référence.....	7
III-C - Activation des erreurs ASP.....	8
IV - Code de la WebPart.....	8
IV-A - La base.....	8
IV-A-1 - Les variables d'environnement.....	9
IV-A-2 - Le CAML.....	9
IV-A-3 - Exploitation de la requête.....	10
IV-B - Création de l'interface utilisateur :.....	10
IV-C - Activation de la feature.....	11
V - CAML Avancé.....	14
VI - Trucs et astuces.....	15
VI-A - Requête paramétrée rapide.....	15
VI-B - Bibliothèques de formulaires.....	16

## I - Introduction



La WebPart que nous allons créer ici, va permettre de se familiariser avec le langage CAML. Ce langage est très utilisé dans SharePoint 2007, il est au cœur même du système. On le trouve absolument partout, y compris dans les WebPart. Il permet d'effectuer des requêtes afin de récupérer différentes informations contenues dans votre site. Nous allons donc essayer de voir l'ensemble des fonctionnalités indispensables à connaître pour créer des WebParts permettant de faire des recherches. Ce n'est pas un cours sur le langage CAML qui est proposé ici, mais une introduction à son utilisation avec les WebParts, nous ne verrons donc que certains aspects de ce langage.

J'ai volontairement détaillé au maximum les différentes étapes, car toutes les personnes utilisant SharePoint ne sont pas des développeurs. Or tout utilisateur ou administrateur, peut être obligé de créer une WebPart pour un besoin spécifique à son environnement de travail.

## II - Contexte

Pour le bon fonctionnement de notre WebPart, nous avons besoin d'une custom list, ou d'une bibliothèque de formulaires ainsi que d'un projet Visual Studio de type WSPBuilder. Si vous avez déjà cette configuration ou si vous savez comment la mettre en place sans difficulté, je vous invite à passer directement au chapitre 4 sans plus attendre.

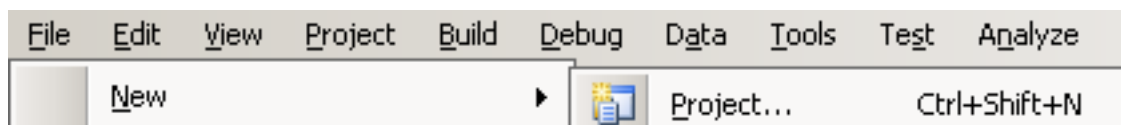
### Ce dont vous avez besoins pour suivre cet article :

- Windows Server 2003 R2 ou version ultérieure
- Microsoft Office SharePoint (  WSS ou  MOSS )
- Microsoft Visual Studio 2008 ou version ultérieure
- WSPBuilder (téléchargeable [ici](#))

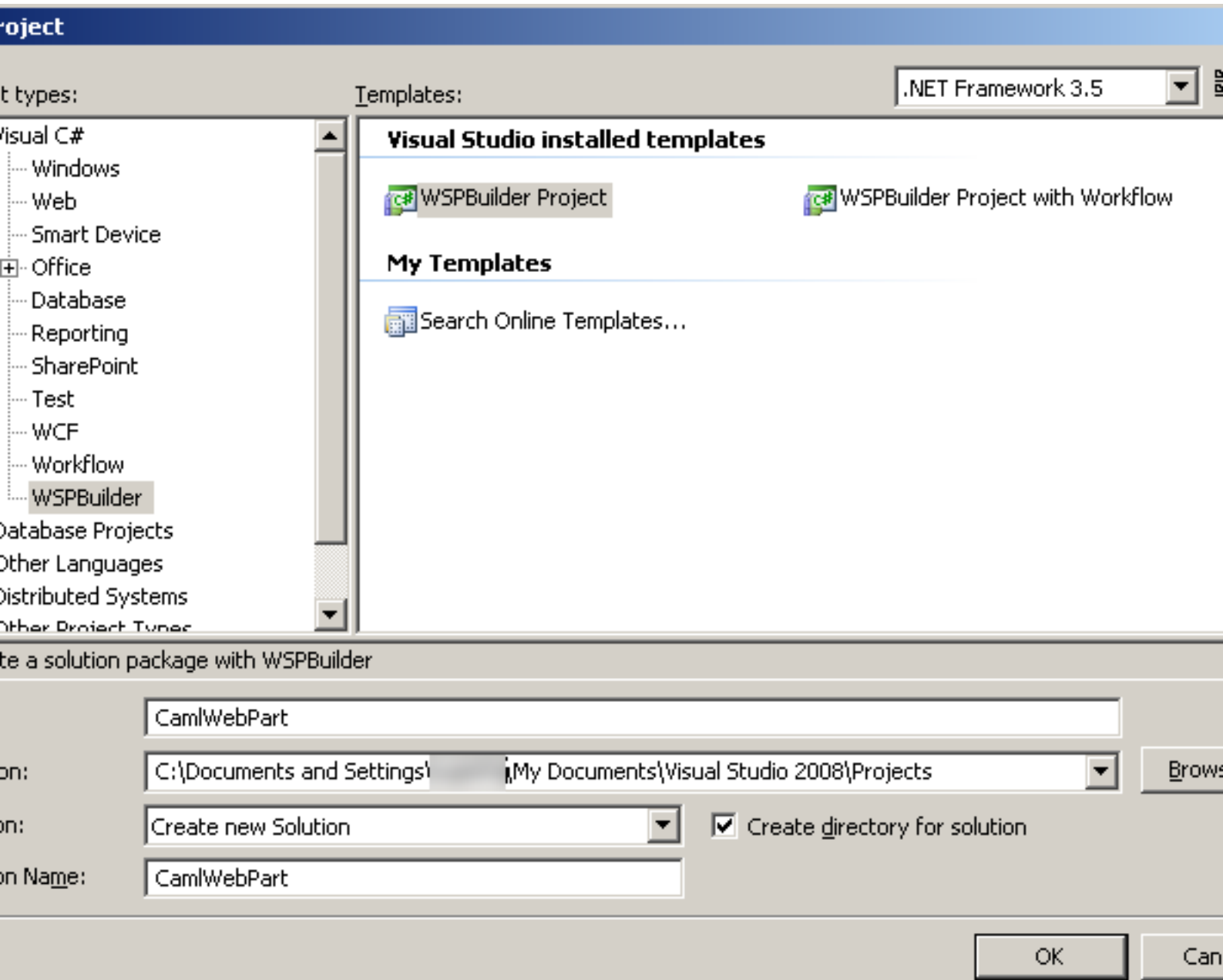
## III - Mise en place de l'environnement

### III-A - Création du projet

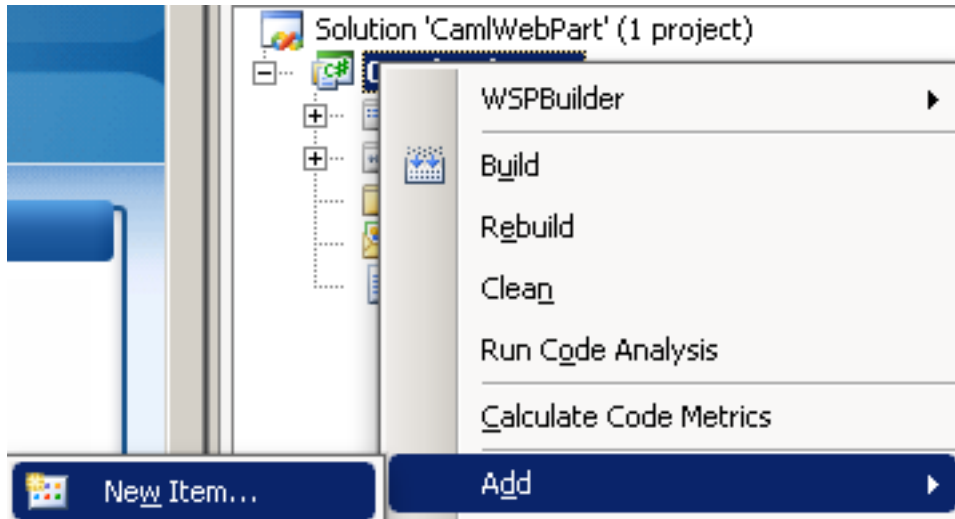
Pour commencer nous allons créer un nouveau projet dans Visual Studio, comme ceci :



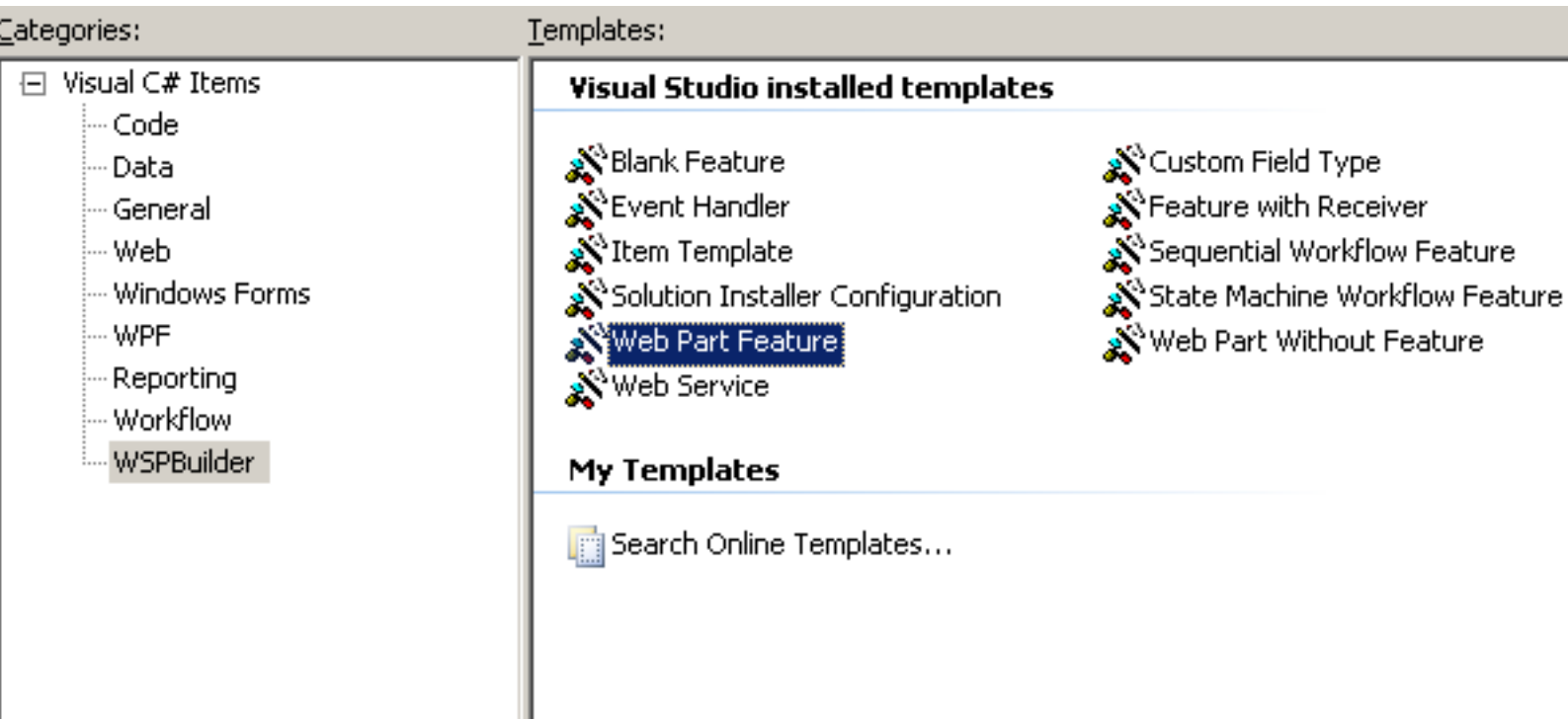
Nous choisissons un projet de type WSPBuilder, avec pour template "WSPBuilder Project". Nous nommerons ce projet "CamlWebPart".



Nous devons maintenant ajouter notre WebPart à notre solution. Pour cela, il faut se placer dans la fenêtre "Solution explorer", sélectionner notre projet, puis faire un click droit, "Add" puis "new item...".



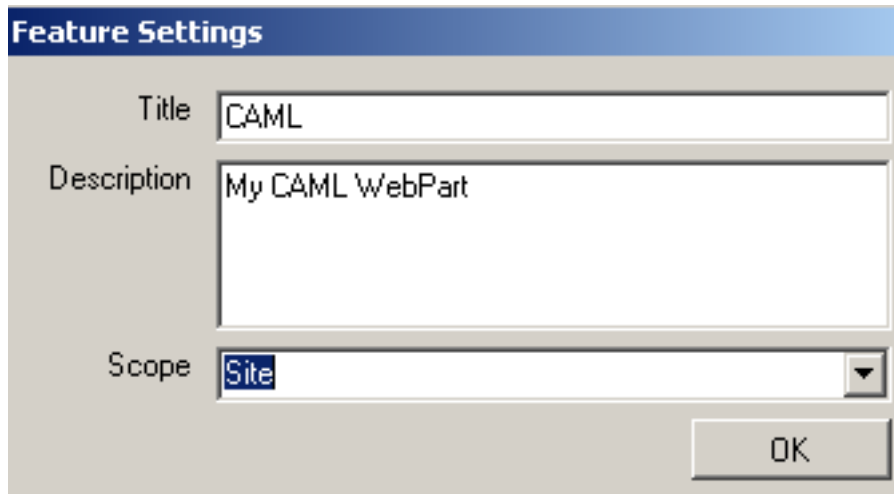
Maintenant nous allons ajouter la WebPart que l'on nommera "CAML" :



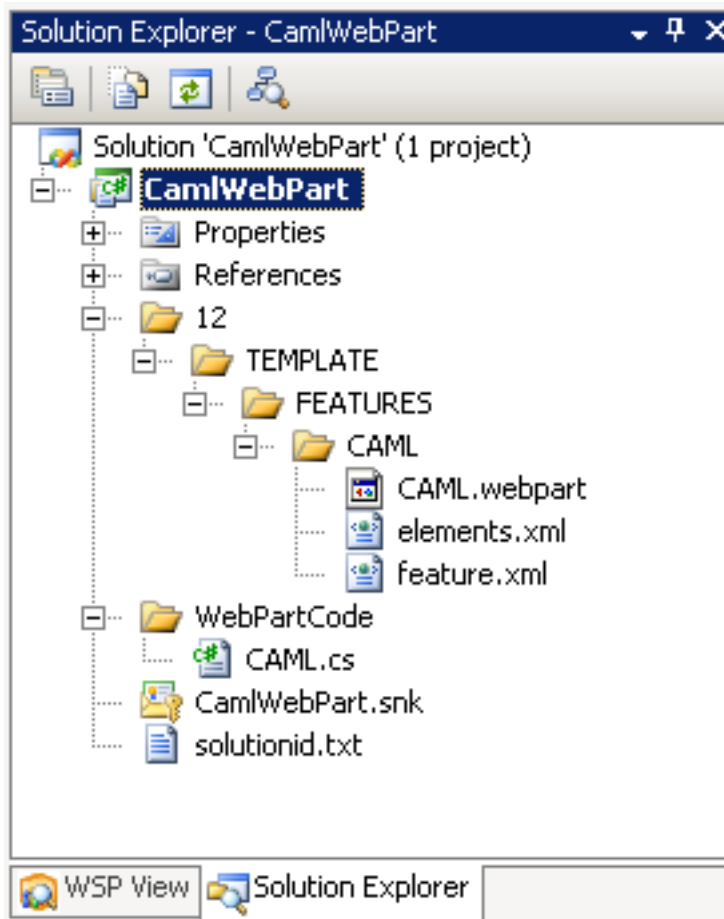
Add a new feature that deploys a web part

Name:

Nous choisirons la "Scope" Site. Libre à vous de placer ce que vous voulez dans le titre et dans la description de la WebPart.



Une fois cette étape terminée, nous allons pouvoir commencer à visualiser ce que les assistants ont fait pour nous. Tout d'abord, dans la fenêtre "Solution explorer", on constate qu'il existe plusieurs dossiers.



Ces différents dossiers correspondent à la hiérarchie de Microsoft Office Sharepoint. Plus particulièrement à la hiérarchie du dossier système où toutes les informations brutes de Sharepoint sont stockées, à savoir, tout ce qui n'est pas directement dans la base de données SQL Server.

Si vous êtes curieux et souhaitez en apprendre d'avantage, je vous invite à aller jeter un coup d'œil directement dans ce dossier qui se trouve dans "C:\Program Files\Common Files\Microsoft Shared\web server extensions\12".

**Dans notre dossier CAML, il existe trois fichiers :**

- CAML.webpart où nous retrouvons la définition de la WebPart, les informations basiques qui seront fournies à l'utilisateur lors de sa mise en place.

- elements.xml où nous retrouvons les données relatives à l'emplacement de la WebPart dans le menu de SharePoint, ici on constate notamment que la WebPart est par défaut dans un groupe appelé "MyGroup".
- feature.xml où nous retrouvons les données qui correspondent au versioning ainsi qu'à l'identification de la feature. On y trouve notamment l'Id de la feature mais aussi le titre, la description la version etc... Ces données correspondent aux informations que l'on retrouvera dans la liste des features présentent dans notre collection de site SharePoint.

Un autre dossier a également été créé, il s'agit du dossier "WebPartCode" qui contient le code C# de notre WebPart, il est donc pour le moment presque vide.

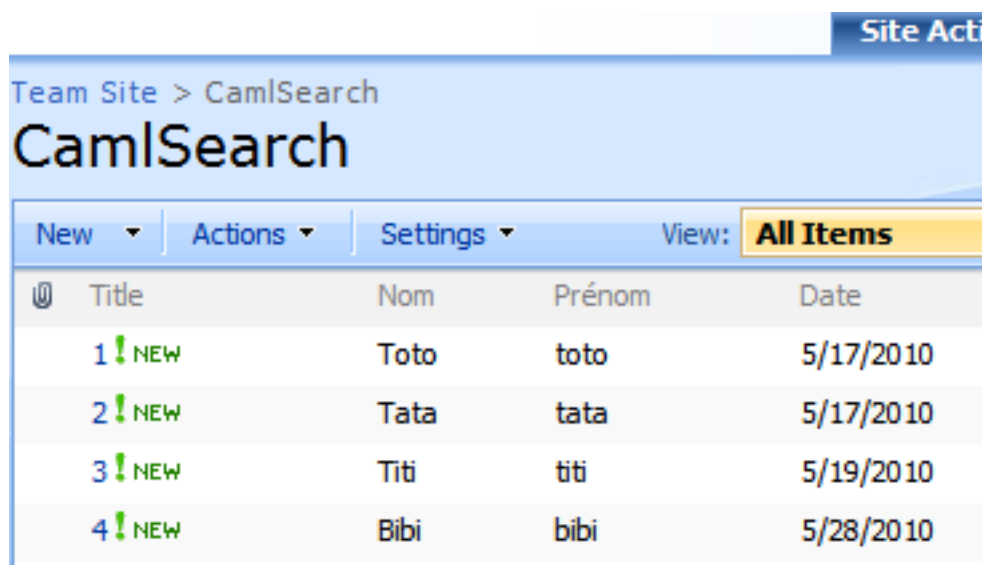
### III-B - Création d'une liste SharePoint de référence

Comme je vous l'ai précisé au début de cet article, le CAML est un langage intégré dans SharePoint 2007 qui nous permet de rechercher des informations contenues dans notre site.

Mais pour ce faire, faut-il encore avoir des données à aller chercher. Nous allons donc créer une liste dans SharePoint afin d'en avoir à aller chercher.

Je suppose que vous savez comment créer une liste, je ne vais donc pas détailler cette partie, je vais juste vous montrer la liste que je vais utiliser pour l'exemple afin de vous permettre de mieux comprendre la suite. Voici donc la liste que j'ai appelé "CamlSearch" :

Columns	
A column stores information about each item in the list. The following columns are available:	
Column (click to edit)	Type
Title	Single line of text
Nom	Single line of text
Prénom	Single line of text
Date	Date and Time
Created By	Person or Group
Modified By	Person or Group



Team Site > CamlSearch				
CamlSearch				
New ▾		Actions ▾		Settings ▾
				View: <b>All Items</b>
🔗	Title	Nom	Prénom	Date
1	! NEW	Toto	toto	5/17/2010
2	! NEW	Tata	tata	5/17/2010
3	! NEW	Titi	titi	5/19/2010
4	! NEW	Bibi	bibi	5/28/2010

Comme vous pouvez le constater c'est une liste très simple, avec uniquement trois champs ajoutés à savoir "Nom", "Prénom" et "Date".

### III-C - Activation des erreurs ASP

SharePoint est un outil destiné à un large panel d'utilisation et d'utilisateurs. Afin que votre site paraisse le mieux possible, SharePoint utilise ce que l'on appelle les "custom errors", ce qui veut dire que si une erreur intervient lors d'une manipulation, un gentil message apparaît afin de vous informer que l'opération désirée n'est pas possible et qu'il faut contacter votre administrateur afin d'obtenir d'avantages de renseignements. Étant des développeurs, comment faire pour savoir d'où provient notre erreur si nous ne pouvons pas visualiser son contenu? Pour éviter ce désagrément et avoir le vrai message d'erreur, il faut modifier le fichier "web.config" de votre site web. Celui-ci se trouve dans le dossier "C:\inetpub\wwwroot\wss\VirtualDirectories\Port\_Number", le "Port\_Number" correspond bien entendu au numéro du port que vous avez déterminé lors de la création de votre site. Il existe deux balises à modifier : (je vous invite à effectuer une recherche afin de les trouver)

- **customErrors mode="On"** : cette balise permet à SharePoint d'utiliser ces propres messages d'erreur, il faut donc la changer pour mettre "Off".
- **CallStack="false"** : cette balise va vous permettre d'afficher la pile d'appel de l'application, elle peut vous permettre d'obtenir d'avantage d'informations, il faut donc la modifier pour mettre "true".

Parfait nous allons pouvoir commencer à coder.

### IV - Code de la WebPart

#### IV-A - La base

Comme je viens de vous le préciser, le fichier contenant le code C# de notre WebPart n'est pas complètement vide. En effet WSPBuilder a déjà rempli ce fichier avec quelques fonctions, mais nous n'aurons pas besoin de tout cela, nous allons donc nettoyer ce fichier. Voici le contenu du fichier que vous devez obtenir :

Nous avons ici le strict minimum dont nous avons besoin, à savoir le constructeur et la méthode "CreateChildControls" qui n'est autre que la méthode de création de nos différents composants.

En premier lieu il est important de lister les différents éléments dont on va avoir besoin.

- Le nom de la liste : CamlSearch;
- Le nom de la colonne sur laquelle va se porter notre recherche : Nom;
- Le type de l'élément de la recherche : en fonction des tests;
- La valeur que l'on recherche : en fonction des tests;
- Afficher le résultat.

Nous pouvons désormais coder.

Nous allons commencer par créer un Label, celui-ci va nous permettre d'afficher le résultat de notre requête dans la WebPart.

```
public class CAML : Microsoft.SharePoint.WebPartPages.WebPart
{
    private Label lbl_Result;
```

Parfait! Maintenant nous allons créer une méthode que nous nommerons "ExecuteQuery", qui comme son nom l'indique va nous permettre d'exécuter la requête CAML.

Pour cela nous allons procéder en plusieurs étapes.



## IV-A-1 - Les variables d'environnement

```
private int ExecuteQuery()
{
    int nbRow = 0;
    SPWeb rootWeb = SPContext.Current.Web;
    SPList spList = rootWeb.Lists["CamlSearch"];
    SPQuery spQuery = new SPQuery();
```

Nous allons décrire l'utilité de ces variables :

rootWeb : permet de sélectionner le site web courant;

spList : permet de sélectionner la liste que nous avons créée précédemment. Pour cela nous utilisons la variable rootWeb, car la liste que nous requêtons est dans le même site que la WebPart (nous verrons plus loin dans cet article comment faire pour aller chercher une liste sur un site distant);

spQuery : il s'agit de la variable qui va contenir notre requête;

nbRow : il s'agit d'un entier que nous utiliserons comme compteur.

## IV-A-2 - Le CAML

Maintenant il faut que l'on construise la requête CAML. Voici donc un exemple de ce que vous pouvez faire :

```
<FieldRef Name='Nom' />
<Eq>
<Value Type='Text'>Toto</Value>
```

Dans cette requête nous allons chercher toutes les enregistrements où :

```
<FieldRef Name='Nom' />
```

Le "Nom" :

```
<Eq>
```

Est égal :

```
<Value Type='Text'>Toto</Value>
```

Au "Text" "Toto" :

De manière générale, on peut traduire cette requête par : `if(Nom == "Toto")`

Vous l'aurez donc certainement deviné, c'est dans la balise "Value Type" que nous pourrions spécifier si l'on cherche à comparer du texte, une date, un entier...

Maintenant il faut traduire cette requête en C#, et pour cela rien de plus facile : nous allons utiliser la propriété "query" de notre variable spQuery.

```
spQuery.Query = "<Where><Eq><FieldRef Name='Nom' /><Value Type='Text'>Toto</Value></Eq></Where>";
```

Voilà, je vous l'avais dit, cette étape est simple pour peu que l'on sache ce que l'on cherche...

### IV-A-3 - Exploitation de la requête

Bon ce n'est pas tout, mais il faut exploiter cette requête. Pour cela nous allons utiliser un "SPListItemCollection" qui va nous permettre de récupérer une collection d'items, qui va correspondre au résultat de notre requête. Il y a des centaines de raisons de vouloir récupérer ces données, pour nous le but sera d'incrémenter un simple compteur qui va nous permettre de savoir combien d'enregistrements dans ma liste correspondent à ma recherche. Pour cela nous allons donc utiliser un "foreach".

```
SPListItemCollection collectionListItems = spList.GetItems(spQuery);
foreach (SPListItem oListItem in collectionListItems)
    nbRow++;
```

Nous retrouvons ici la variable "nbRow" qui est incrémentée chaque fois qu'un enregistrement correspond à notre recherche.

Toute personne qui a déjà créée une WebPart s'est déjà confrontée à une page d'erreur de l'asp.net, et cela est très contraignant, car la page dans laquelle est placée la WebPart est inutilisable, on est obligé d'aller dans les propriétés et de supprimer la WebPart... Une manipulation pas très agréable et assez énervante quand cela est dû à une erreur de code toute bête. Je vais donc vous donner une petite astuce pour éviter cela : il faut placer notre requête dans un bloc "try catch". Comme cela nous allons pouvoir récupérer le message d'erreur et l'afficher directement dans notre WebPart et cela sans que la page soit cassée.

Pour ce faire, Visual Studio nous offre une solution très sympathique, il vous suffit de sélectionner le code que vous souhaitez placer dans votre bloc, ensuite faites un clic droit dessus, puis sélectionnez "Surround With" puis sélectionnez "try". Magique! Il a placé le bloc pour vous et il a même indenté le texte.

Bien. Maintenant il ne nous reste plus qu'à récupérer le message d'erreur et à afficher le résultat de notre requête. Voici donc le code que je vous propose pour la totalité de la fonction "ExecuteQuery" :

```
private int ExecuteQuery()
{
    int nbRow = 0;
    try
    {
        SPWeb rootWeb = SPContext.Current.Web;
        SPList spList = rootWeb.Lists["CamlSearch"];
        SPQuery spQuery = new SPQuery();
        spQuery.Query = "<Where><Eq><FieldRef Name='Nom'/><Value Type='Text'>Toto</Value></Eq></Where>";

        SPListItemCollection collectionListItems = spList.GetItems(spQuery);
        foreach (SPListItem oListItem in collectionListItems)
            nbRow++;
    }
    catch (Exception ex)
    {
        lbl_Result = new Label();
        lbl_Result.Text = ex.Message;
    }
    return nbRow;
}
```

### IV-B - Création de l'interface utilisateur :

L'interface utilisateur se gère dans la méthode "CreateChildControls", pour nous il s'agit ici d'afficher notre label créé précédemment, le code se résume donc à cela :

```
protected override void CreateChildControls()
{
    base.CreateChildControls();
    lbl_Result = new Label();
    lbl_Result.Text = ExecuteQuery().ToString();
}
```

```

this.Controls.Add(lbl_Result);
}

```

Ici nous n'avons rien de particulier, on instancie le label, ensuite on lui attribue le résultat de notre méthode "ExecuteQuery", sans oublier la méthode "ToString" car le label contient uniquement du texte or, notre méthode retourne un entier. Ensuite il reste à ajouter le label à la WebPart grâce à la méthode "Controls.add".

Parfait, nous pouvons maintenant compiler notre projet à l'aide de WSPBuilder. Pour cela, suivez ces étapes :

- placez-vous dans la fenêtre "Solution Explorer";
- sélectionnez votre projet, puis clic droit;
- sélectionnez "WSPBuilder";
- sélectionnez "Build WSP".



Maintenant, il ne nous reste qu'à déployer notre WebPart, pour cela il existe trois méthodes :

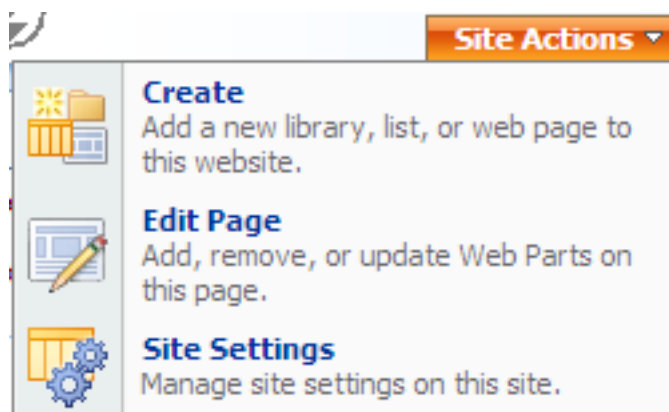
- WSPBuilder : pour cette méthode, c'est l'outil WSPBuilder qui va travailler pour nous. Il nous suffit, dans Visual Studio de sélectionner "Deploy" après avoir fait les mêmes manipulations que ci-dessus. Il va s'en dire que c'est la méthode la plus simple surtout lors de la phase de tests;



- SharePoint Solution Installer : cet outil est très pratique, il permet de déployer votre WebPart n'importe où. Il vous suffit de placer votre fichier WSP dans le dossier SharePoint Solution Installer puis de modifier le fichier "Setup.exe.config" directement dans Visual Studio. Pour plus d'informations je vous invite à vous rendre à cette adresse : <http://SharePointinstaller.codeplex.com>;
- Stsadm : il s'agit de l'outil de base proposé avec Visual Studio, celui-ci s'utilise directement dans l'invite de commande de Visual Studio. Le descriptif de cet outil est disponible à cette adresse : <http://technet.microsoft.com> (pour information, les autres outils font également appel à celui-ci, mais cela reste invisible pour l'utilisateur)

## IV-C - Activation de la feature

Nous allons maintenant pouvoir aller visualiser le résultat dans SharePoint. Pour cela nous devons accéder à la liste des features de notre site, dans le menu "Site Actions" de SharePoint et sélectionner "Site Settings".

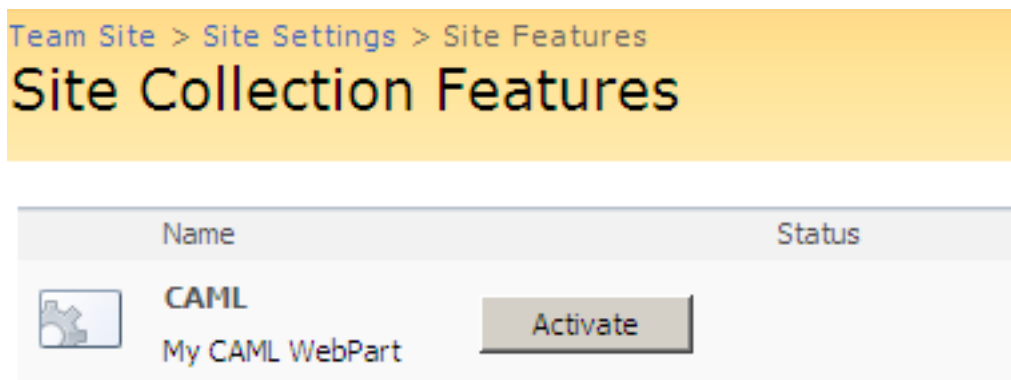


Ensuite dans le menu "Site Collection Administration" vous devez sélectionner "Site collection features".

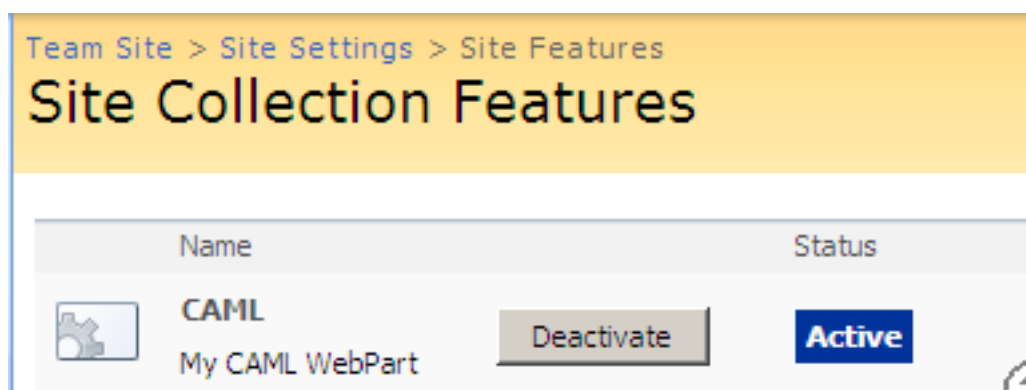
### Site Collection Administration

- ▣ Recycle bin
- ▣ Site collection features
- ▣ Site hierarchy
- ▣ Portal site connection

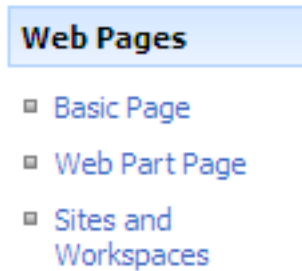
Nous pouvons visualiser notre feature dans la liste :



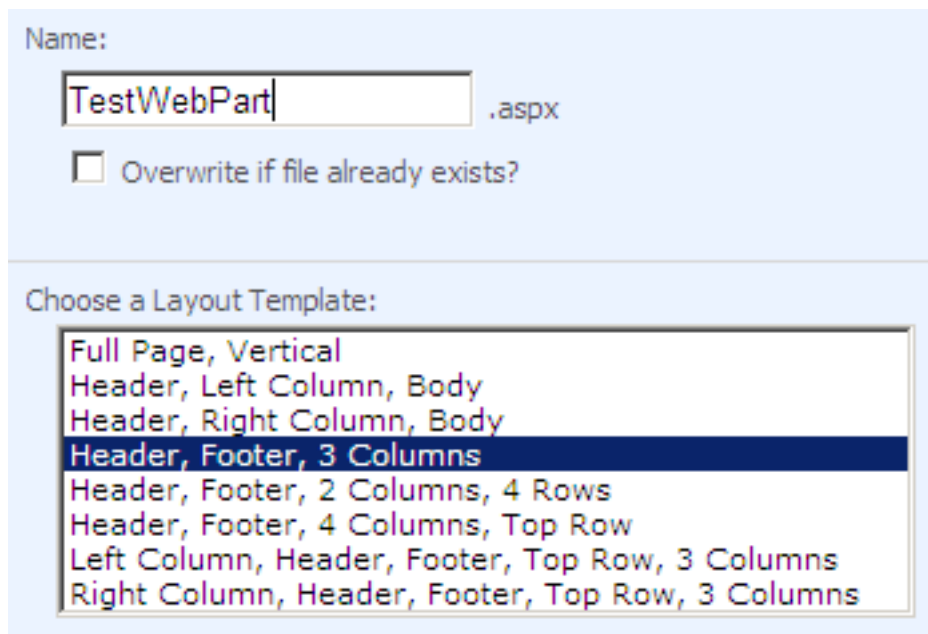
Il faut maintenant activer la feature, si l'opération se déroule correctement, vous devriez voir apparaitre un rectangle bleu sur la droite de la feature, comme ceci :



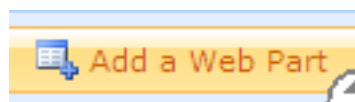
Parfait, nous pouvons maintenant aller ajouter notre WebPart à une page de composants WebPart. Pour créer cette page, il suffit de se rendre dans le menu "Site Actions" et de sélectionner "Create". Ensuite dans la section "Web Pages", sélectionnez "Web Part Page".



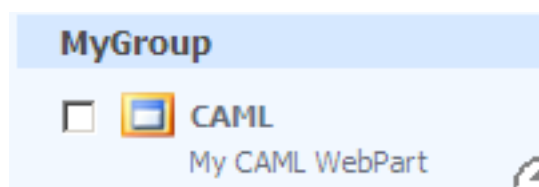
Nous nommerons notre page "TestWebPart".



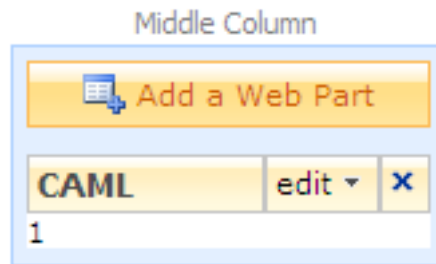
Maintenant il nous suffit d'ajouter notre WebPart dans la page, pour cela cliquer sur "Add a Web Part", puis sélectionner votre WebPart.



Si vous avez laissé les valeurs par défaut, vous devriez trouver votre WebPart dans la section du bas nommée "MyGroup". Cochez la case correspondante puis cliquez "Add" en bas de page.



Nous pouvons tout de suite visualiser le résultat de notre requête, ici "1". Félicitation! Vous venez de créer votre première WebPart contenant une requête CAML.



## V - CAML Avancé

Vous savez maintenant créer une WebPart qui permet de récupérer des données qui correspondent à une requête CAML. Le problème c'est que notre requête est figée dans notre code. Il est généralement plus intéressant de créer une WebPart qui permet à son utilisateur de rechercher les données qu'il souhaite, sans avoir à demander que l'on change le code de la WebPart, cela semble impensable. Il est donc utile de savoir comment faire pour créer une WebPart qui contienne une requête paramétrée. Il existe plusieurs solutions qui dépendent des besoins en production. En voici deux qui me semblent être les plus courantes:

- l'utilisation de champs texte dans la WebPart, afin de permettre à l'utilisateur de définir des critères de recherche;
- l'utilisation d'un éditeur personnalisé, afin que seul la personne ayant des droits suffisants puisse modifier les critères de recherche.

Il vous est maintenant aisé de créer une WebPart intégrant du CAML.

Mais il est bien souvent intéressant de pouvoir créer des requêtes un peu plus complexes, avec d'avantage de paramètres, afin de satisfaire une recherche bien spécifique.

Nous allons donc commencer par voir quels sont les éléments CAML à connaître pour pouvoir prétendre faire une requête plus importante, je vous donnerai ensuite des petits bouts de code afin que vous puissiez les intégrer rapidement.

En premier lieu, comment utiliser plusieurs critères.

Voici la construction d'une requête CAML avec deux critères :

```
<Where>
  <And>
    <Eq>
      <FieldRef Name='Nom' />
      <Value Type='Text'>Toto</Value>
    </Eq>
    <Eq>
      <FieldRef Name='Date' />
      <Value Type='DateTime'>5/17/2010</Value>
    </Eq>
  </And>
</Where>
```

Descriptif de la requête :

**Le champ "Nom" = Toto (champs de type Text)**

**Et**

**Le champ "Prénom" = toto (champs de type Text)**

Voici un autre exemple, toujours avec deux critères :

```
<Where>
  <Or>
    <Eq>
      <FieldRef Name='Nom' />
      <Value Type='Text'>Toto</Value>
    </Eq>
    <Eq>
```

```

        <FieldRef Name='Date' />
        <Value Type='DateTime'>5/17/2010</Value>
    </Eq>
</Or>
</Where>
    
```

Descriptif de la requête :

Le champ "Nom" = Toto (champs de type Text)

**OU**

Le champ "Date" = 5/17/2010 (champs de type date)

Comme nous pouvons le constater à l'aide de ces deux exemples, il existe de nombreuses possibilités. Ces exemples sont simples afin de bien comprendre la structure. Nous allons donc maintenant passer à des exemples plus complexes, mais toujours avec des éléments simples.

```

<Where>
  <Or>
    <And>
      <Eq>
        <FieldRef Name='Nom' />
        <Value Type='Text'>Toto</Value>
      </Eq>
      <Eq>
        <FieldRef Name='Prénom' />
        <Value Type='Text'>toto</Value>
      </Eq>
    </And>
    <Contains>
      <FieldRef Name='Nom' />
      <Value Type='Text'>Ti</Value>
    </Contains>
  </Or>
</Where>
    
```

Descriptif de la requête :

le champs "Nom" = Toto (champs de type Text)

et

le champs "Prénom" = toto (champs de type Text)

**OU**

le champs "Nom" contient Ti (champs de type Text)

Comme nous pouvons le voir, une fois que l'on met en place la structure, la requête est simple, mais cela prend du temps et est assez rébarbatif. Heureusement pour nous, il existe un logiciel, très léger et très pratique, qui propose de créer les requêtes pour nous. Il s'agit de CAML Query Builder. Vous le trouverez en libre téléchargement [ici](#).

## VI - Trucs et astuces

### VI-A - Requête paramétrée rapide

Maintenant, comme je vous l'ai promis, je vais vous donner quelques petits morceaux de code, afin de vous simplifier la création de requêtes CAML dans vos WebPart. Comme nous avons pu le voir, il existe quatre points indispensables dans une requête CAML :

- La liaison :
  - And;
  - Or.
- Le type de comparaison

CAML	Définition	Traduction
Eq	Equal	Egal
Neq	Not Equal	différent
Geq	Greater tan or Equal	Supérieur ou Egal
Lt	Lower Than	Plus petit
Leq	Lower than or Equal	Plus petit ou Egal
IsNull	Is Null	Est nul
Contains	Contains	Contient
Begins With	Begins With	Commence par

- Les champs où porte la recherche
- Le type de la recherche :
  - Text;
  - Number;
  - DateTime.

Ces différents éléments sont assez longs à mettre en place dans le code, c'est pourquoi je vous propose ces différents codes que vous pourrez aisément glisser dans votre WebPart afin de permettre à l'utilisateur de choisir les différents éléments de sa recherche. Il s'agit de "switch case" pour l'implémentation d'une DropDownList pour l'exemple (le code se divise en deux parties, la première contient le code pour alimenter la "DDL" dans l'éditeur personnalisé, la seconde contient un "switch case" à placer dans le code de la WebPart). Je pense que le nom des différentes variables est assez explicite pour ne pas avoir besoin de détailler le code.

- La Liaison;
- Le Type de comparaison;
- Le Type de la recherche.

C'est bien beau tout ce code, mais pas forcément parlant, voici donc un exemple de son utilisation :

```

SPQuery.Query = "<Where>"
    + linkOperator
      + camlOperator +
        "<FieldRef Name='"
        + column1
        + "'/><Value Type='"
        + camlValueType + "'>"
        + filter +
        "</Value>"
      + camlOperator_close
      + camlOperator2 +
        "<FieldRef Name='"
        + column2
        + "'/><Value Type='"
        + camlValueType2 + "'>"
        + filter2 +
        "</Value>"
      + camlOperator2_close
    + linkOperator_close +
    "</Where>";
    
```

Dans ce code, toutes les valeurs remplaçables par des variables, le sont, cela nous permet d'obtenir une requête qui peut être entièrement paramétrée par l'utilisateur.

## VI-B - Bibliothèques de formulaires

Dans l'utilisation des bibliothèques de formulaires avec InfoPath, il existe une subtilité qui peut rapidement vous faire tourner en rond et perdre beaucoup de temps. En effet dans un formulaire InfoPath il arrive très souvent, pour ne pas



dire chaque fois, que certain champs comportent un underscore. Or, dans la bibliothèque de formulaires associés dans SharePoint, les colonnes correspondantes ont des espaces à la place. Voici un exemple concret :

- dans InfoPath : Request\_Date (vue source de données);
- dans SharePoint : Request Date (en-tête de colonne).

Le problème est donc que dans votre WebPart vous n'aurez donc pas le bon nom. SharePoint n'a pas supprimé le underscore, mais il l'a remplacé par le code : "\_x0020\_".

Il existe donc une solution : utiliser la méthode "Replace" de la variable de type string. Pour reprendre l'exemple précédent, il suffit de modifier deux lignes :

- column1 -> column1.Replace("\_","\_x0020\_");
- column2 -> column2.Replace("\_","\_x0020\_");

Voilà, vous savez maintenant tout ce qu'il vous faut pour commencer à utiliser le CAML avec SharePoint. Bon courage à tous et bon code.

Un grand merci à **blade159** pour la relecture de cet article

# MCours.com