Tutoriel WPF

Interactions avec une liste Sharepoint

Lucas Girardin – Arnaud Derosin

22/06/2012

Tutoriel pour apprendre à ajouter des éléments et en récupérer depuis une liste Sharepoint via un projet WPF.

Sommaire

Introduction :

Ce tutoriel à pour but de vous apprendre à créer un projet WPF et de vous permettre de faire communiquer votre application avec un serveur SharePoint distant. Vous aurez à la fin de ce tutoriel les bases pour récupérer une les éléments d'une liste SharePoint et en ajouter de nouveau.

Ce tutoriel est destiné à des développeurs qui maitrisent déjà un minimum la programmation orientée objet peut importe le langage. Connaitre les bases du C# est un atout.

Cependant si vous avez déjà programmé en Java vous allez apprécier la syntaxe similaire.

Nous allons travailler avec des fichiers Xaml, la maitrise de ce langage n'est cependant pas nécessaire, nous ne toucherons qu'à des choses simples.

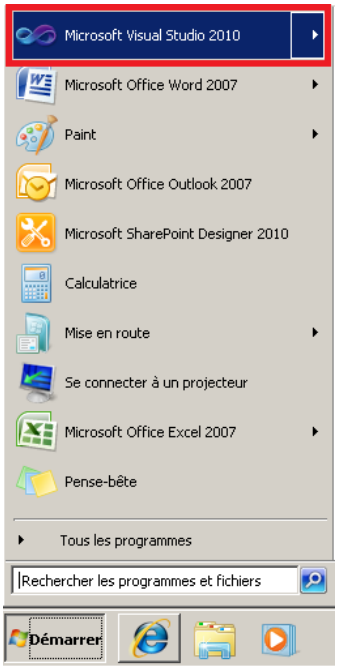
Les prés requis :

Avant de vous lancer dans ce tutoriel vous devez vérifier que :

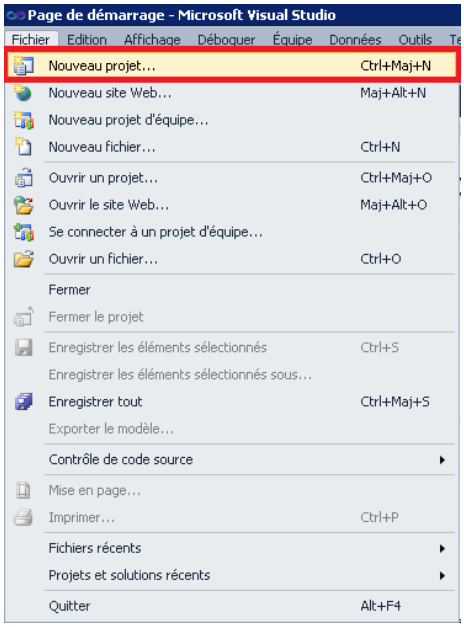
- Vous disposez d'une version de Visual Studio 2010 minimum.
- Vous avez accès depuis votre poste client à un serveur SharePoint depuis votre navigateur internet.
- Votre Framework est à jours. Nous travaillerons avec la version 4.0.

I - Création du projet

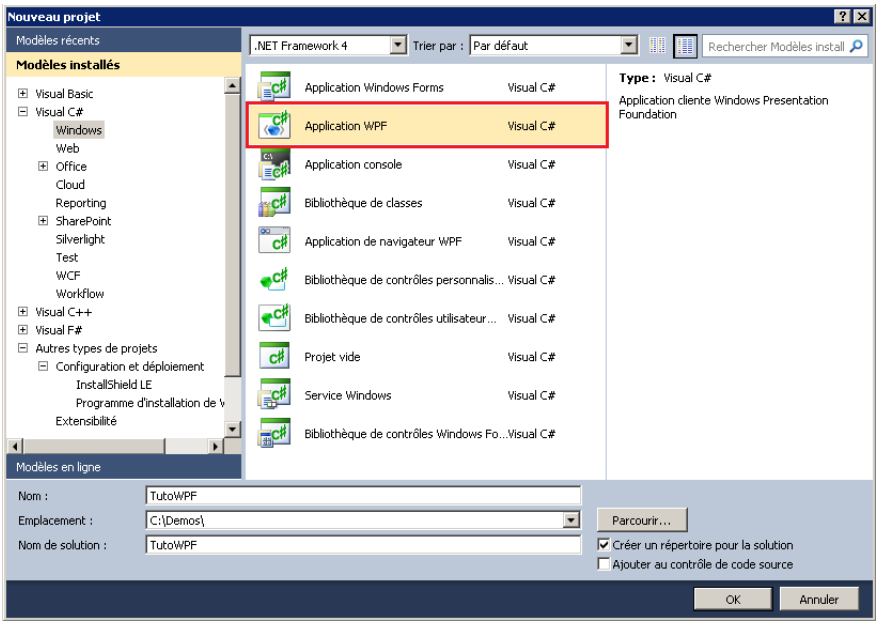
Tout d'abord vous allez ouvrir Visual Studio, si vous ne l'avez jamais utilisé auparavant, pas de panique nous allons vous guider. Ouvrez le donc



Une fois Visual Studio lancé nous allons créer un nouveau projet, pour cela faite Fichier->Nouveau projet

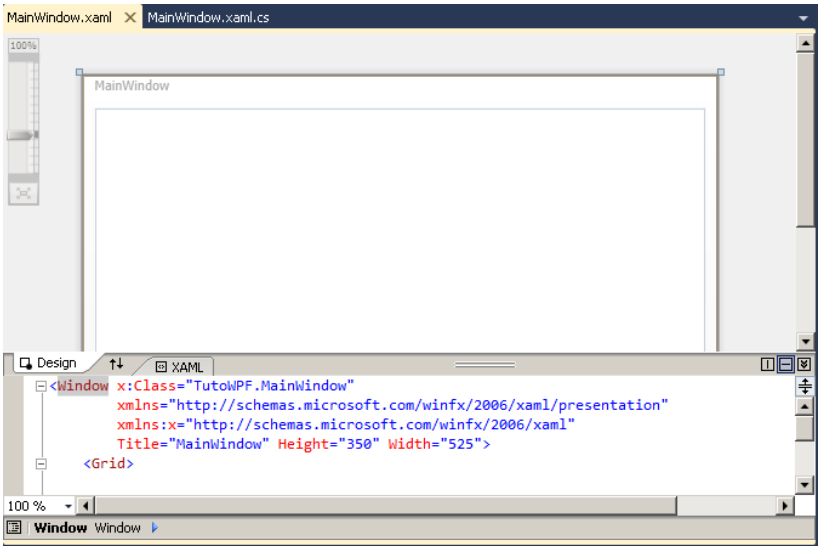


Pour illustrer notre tutoriel nous allons créer une fenêtre WPF, il faut donc choisir un projet Application WPF qui se trouve dans le menu Visual C#->Windows

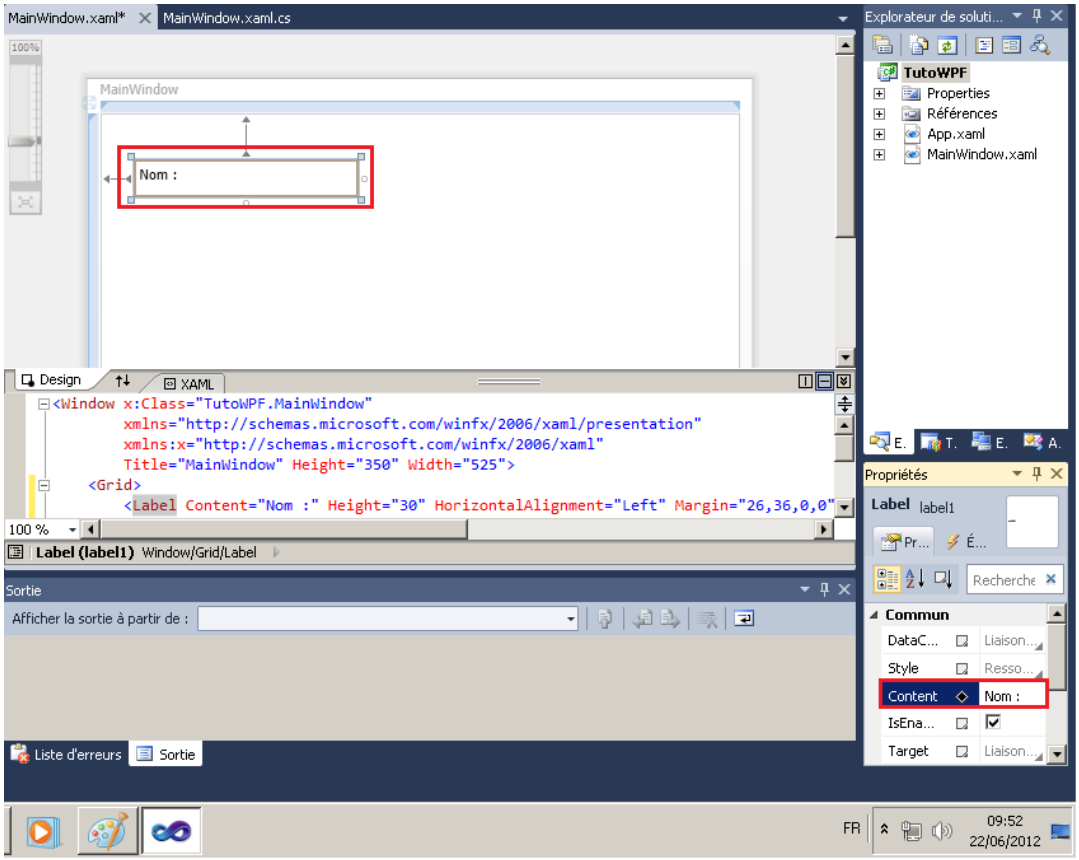


II - Création de la fenêtre

Vous avez maintenant devant vous votre première fenêtre. Ce n'est pas très beau pour le moment, mais vous pouvez y ajouter tous les éléments que vous le souhaitez, comme des boutons, des zones de saisies, des listes déroulantes et encore beaucoup de choses. Tous ces élément ce trouvent dans la boite à outil, faire Ctr+Alt+X pour la voir.

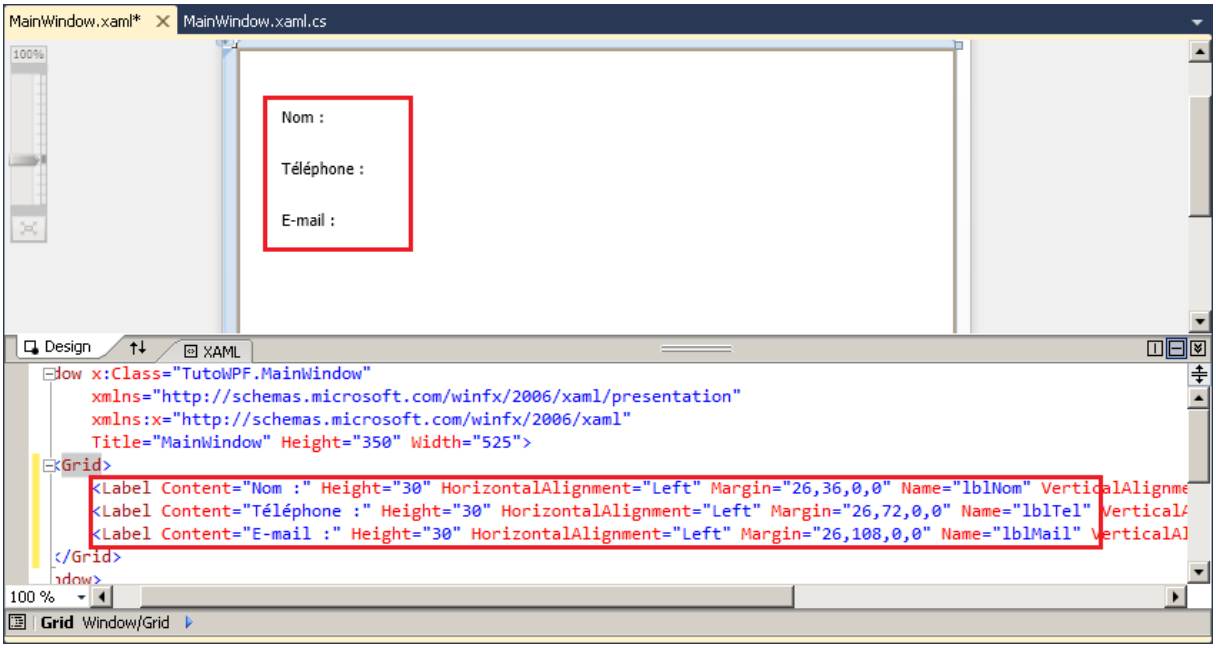


Pour commencer, nous allons mettre en forme notre fenêtre. Nous allons donc ajouter des labels qui sont des indications pour l'utilisateur. Pour changer le texte par défaut du label il suffit de faire F4 pour arriver dans ces propriétés. Vous allez vous rendre compte que plus vous avancez, plus vous aurez d'éléments. Il est donc important de leur donner un Name bien explicite pour pouvoir s'y retrouver.

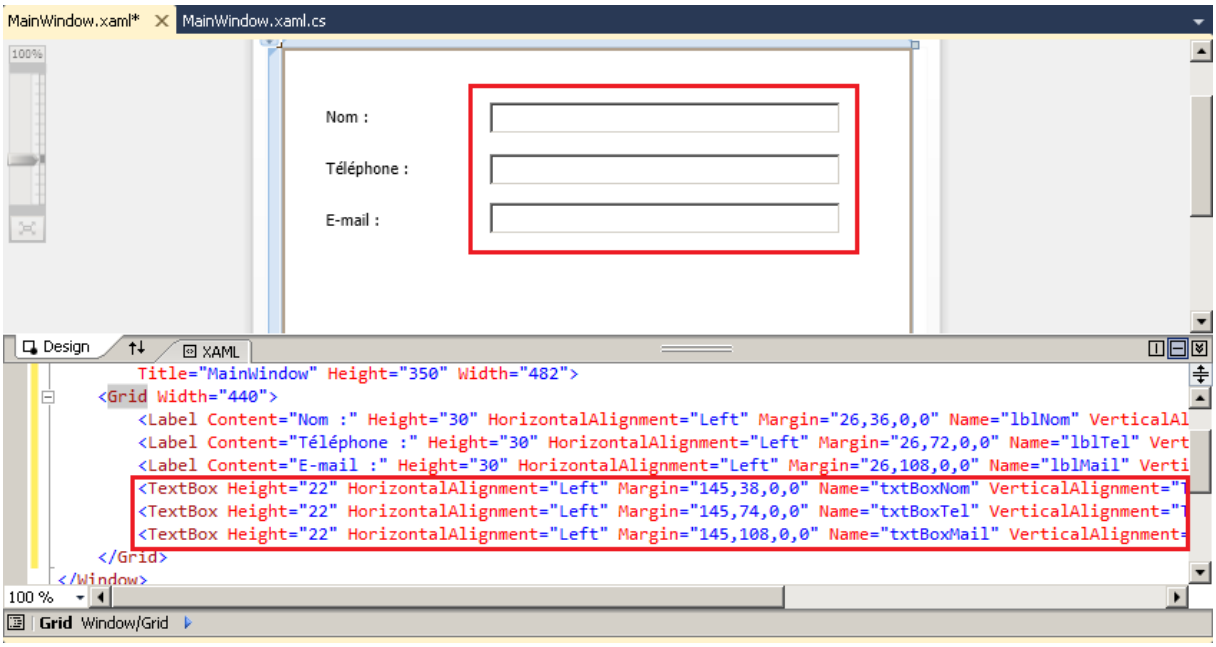



```
<Window x:Class="TutoWPF.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="350" Width="525">
    <Grid>
        <Label Content="Nom :" Height="30" HorizontalAlignment="Left" Margin="26,36,0,0" Name="lblNom" VerticalAlignment="Top"
    </Grid>
</Window>
```

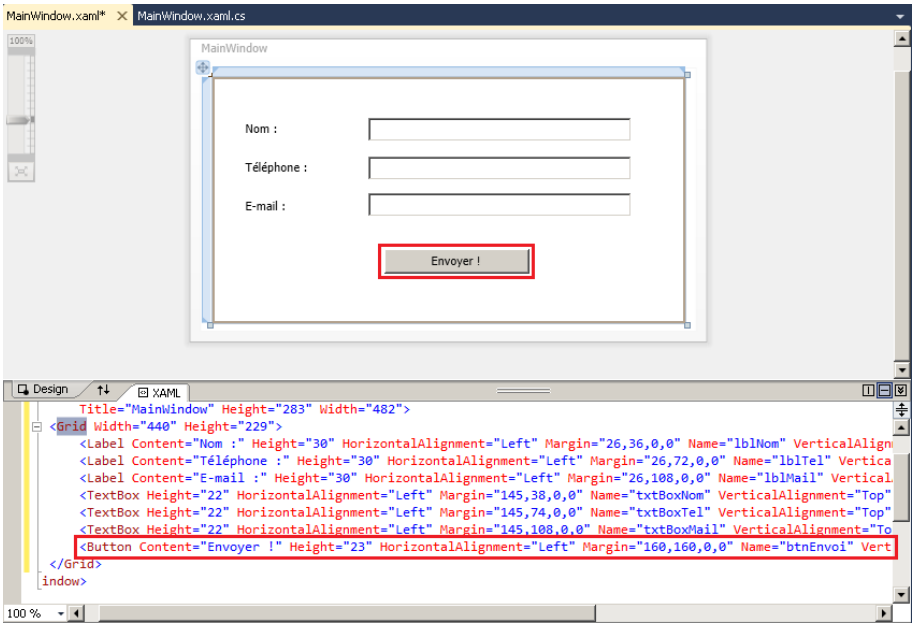
Nous avons pris dans notre exemple la création d'un client dans une liste « Clients » nous allons donc créer trois champ, donc trois label.



Nous allons maintenant ajouter trois « Textboxs » en face de ces champs. Une « Textbox » est une zone de saisie pour l'utilisateur. Cela va nous permettre de récupérer les informations entrées par l'utilisateur. Vous remarquerez les noms qui suivent les labels

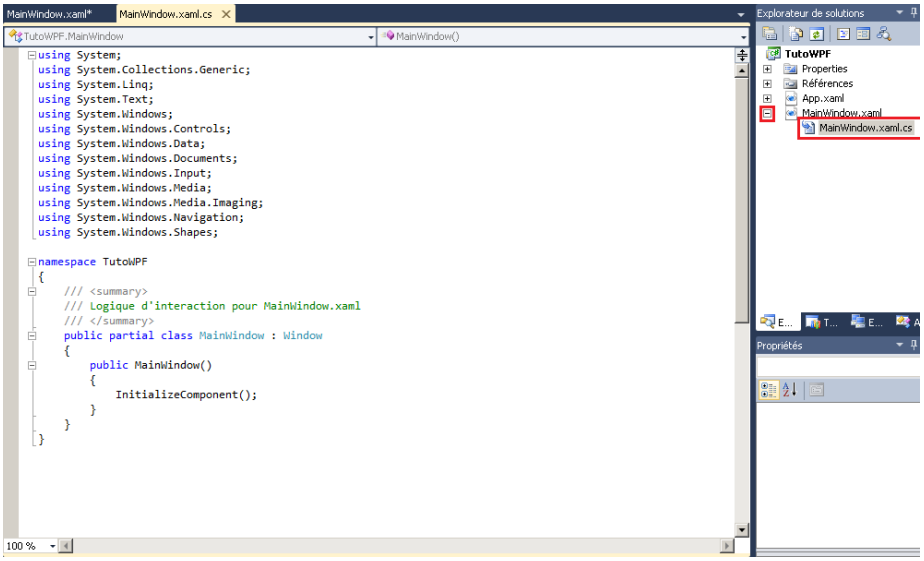


Enfin pour que l'utilisateur puisse nous dire qu'il à fini et qu'il souhaite envoyer les informations. Nous allons créer un bouton qui se nommera « Envoyer ! ». Voila la fenêtre commence déjà à ressembler à quelque chose de correct. Toute cette partie à écrit automatiquement du code en Xaml comme vous pouvez le voir, si vous ne comprenez pas tout ne vous en faites pas, je vous l'ai dit on touchera que très peu au Xaml.



Pour passer au code C# qui va nous permettre de donner des instructions, de lier notre interface à notre code nous allons directement double cliquer sur nos éléments pour qu'ils soient « créés » dans notre fichier .cs

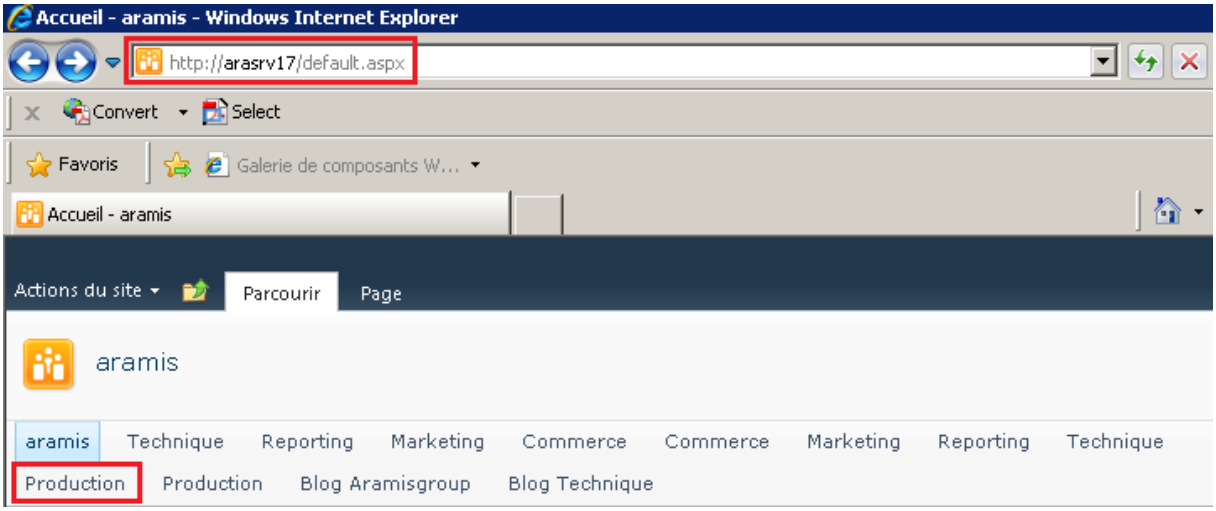
Votre fichier .cs ce trouve en déroulant avec le petit « + » devant votre fichier Xaml.

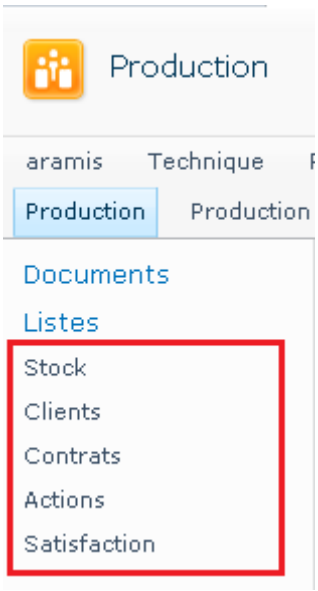


III - Lier votre fenêtre à votre code

1-Ajouter une référence de service

Avant toute chose allez sur votre intranet et récupérez l'url de l'endroit où sont vos listes.

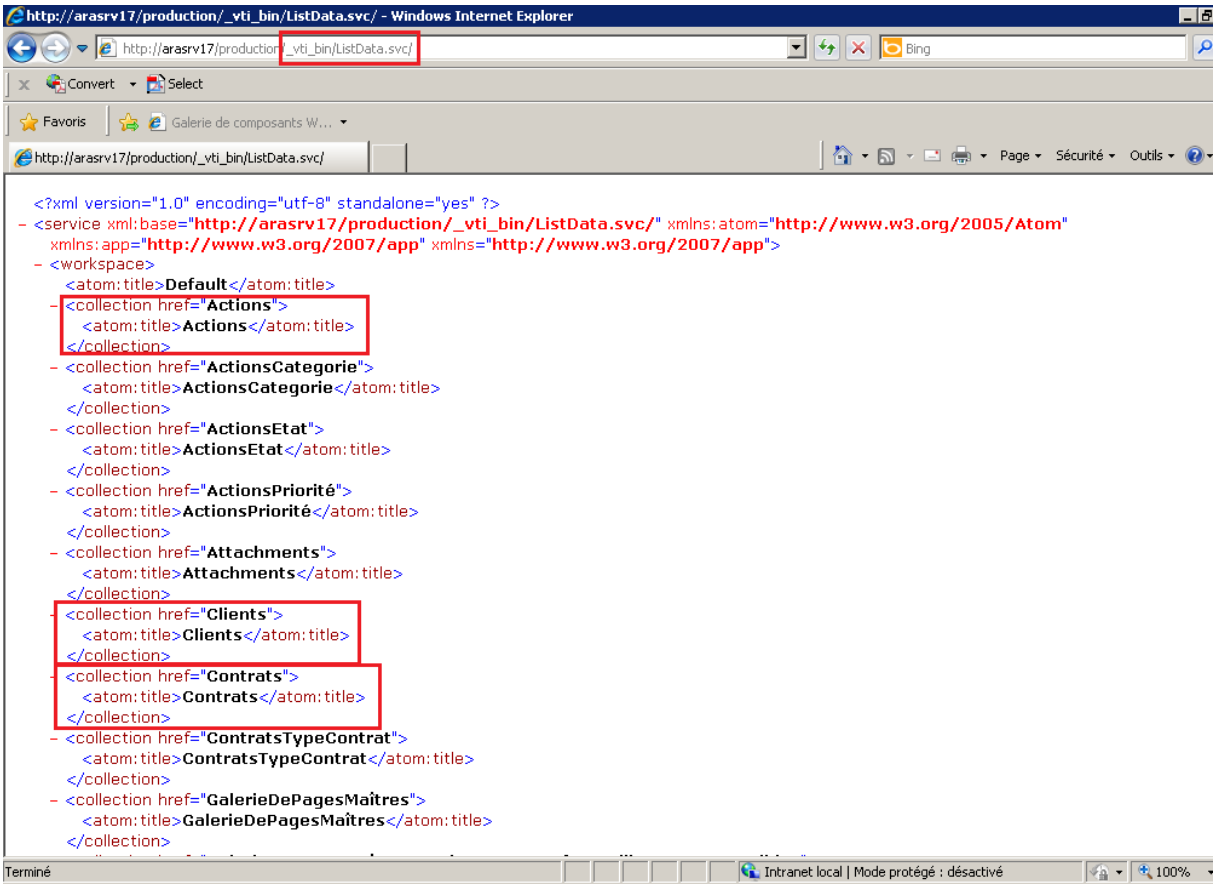




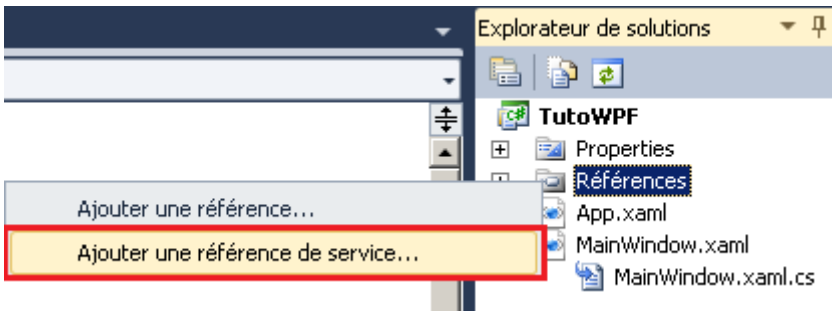
Puis ajouté ceci à l'url _vti_bin/ListData.svc/

Nous travaillons nous sur le sous site production, mais ce n'est peut être pas votre cas

Vous arriverez sur une page avec le nom de toutes vos listes

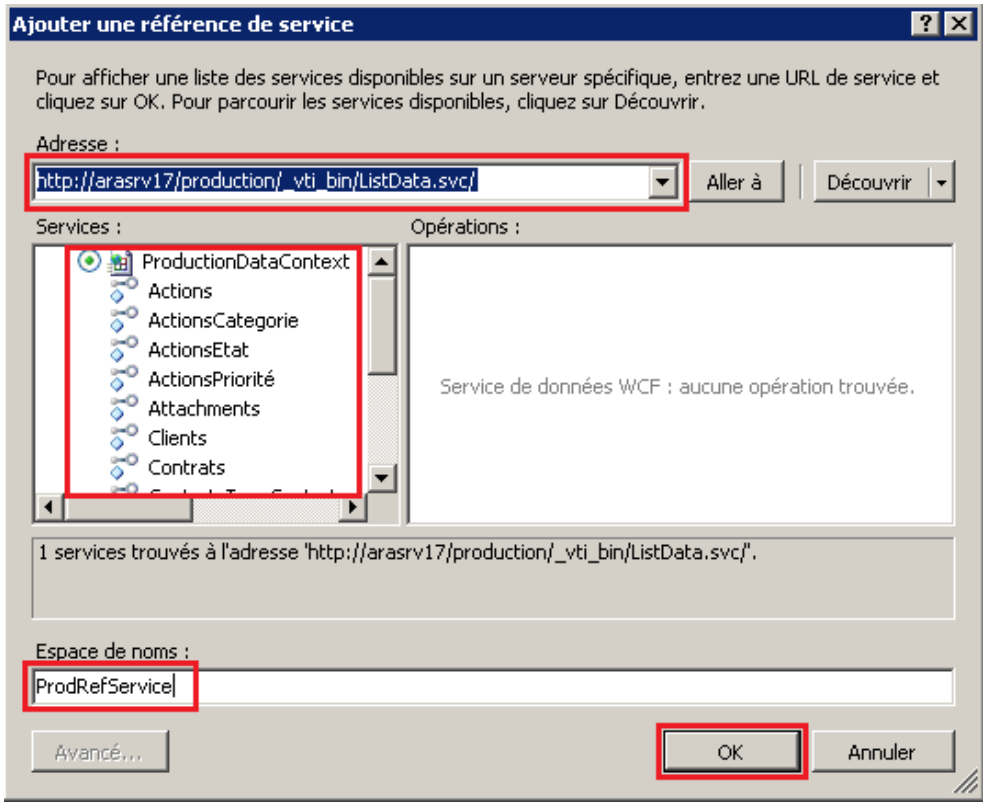


Retournez maintenant sur Visual Studio et faite clique droit sur référence, puis ajouter une référence de service.

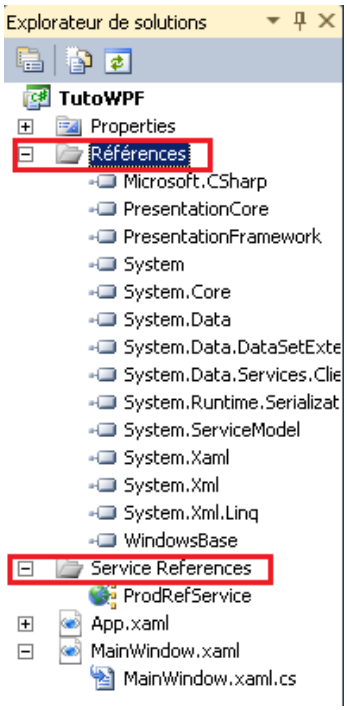


Dans la nouvelle fenêtre qui apparait ajoutez l'adresse où vous avez vu toutes vos listes. Puis cliquez sur « Allez à » Vous allez alors voir apparaitre un service que vous pouvez faire dérouler, vous verrez alors toutes vos listes.

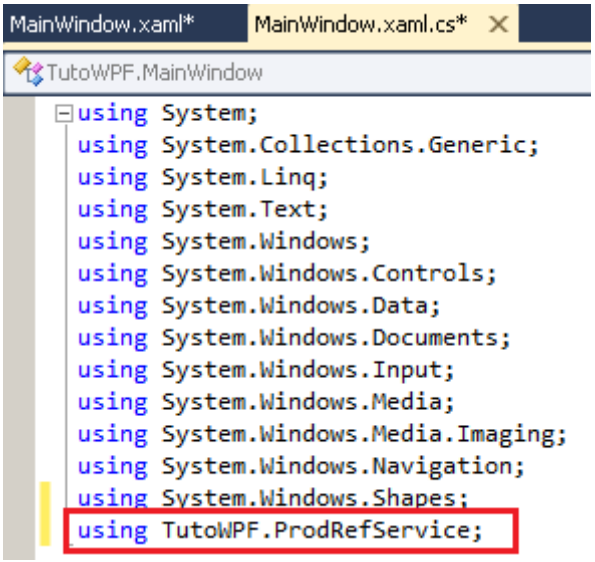
Ajoutez lui un nom et cliquez sur « OK ».



Vous pouvez vérifier que vos références se soient bien ajoutées

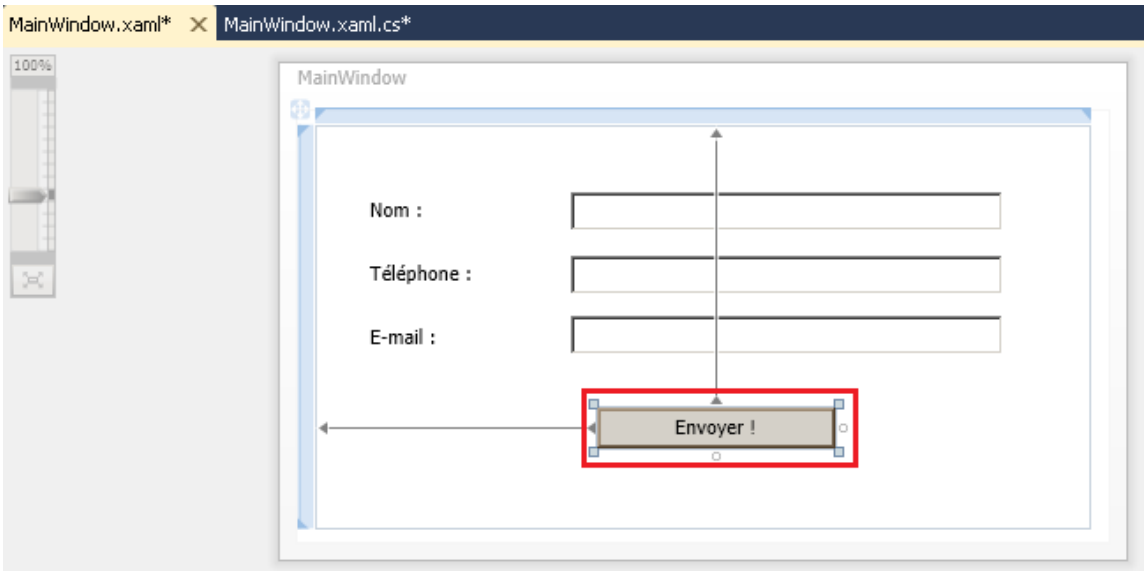


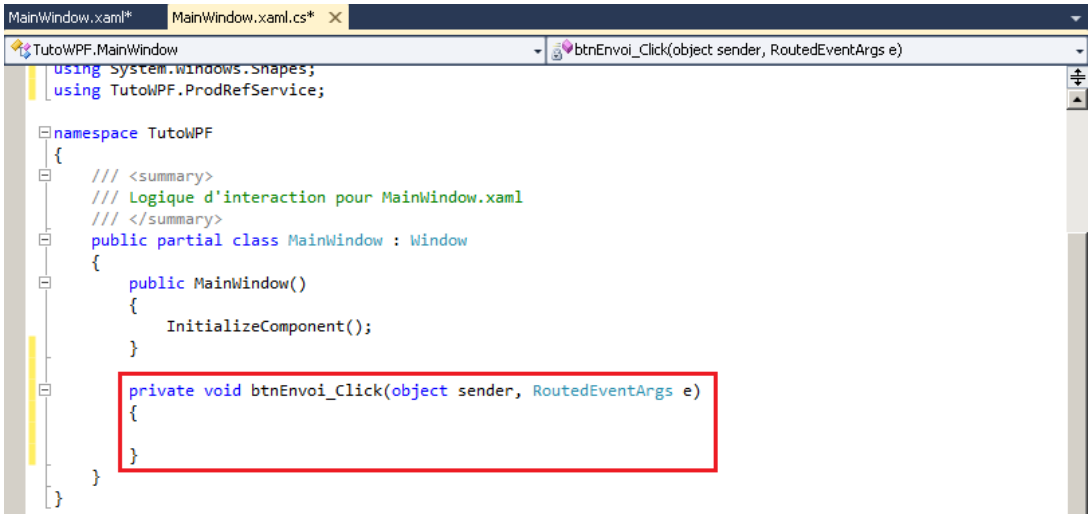
Maintenant il faut ajouter un « using » à votre projet qui correspond au nom du projet et de votre référence de service, ajouter le à la suite des using présent dans votre fichier .xaml.cs.



2-Etablir la connexion

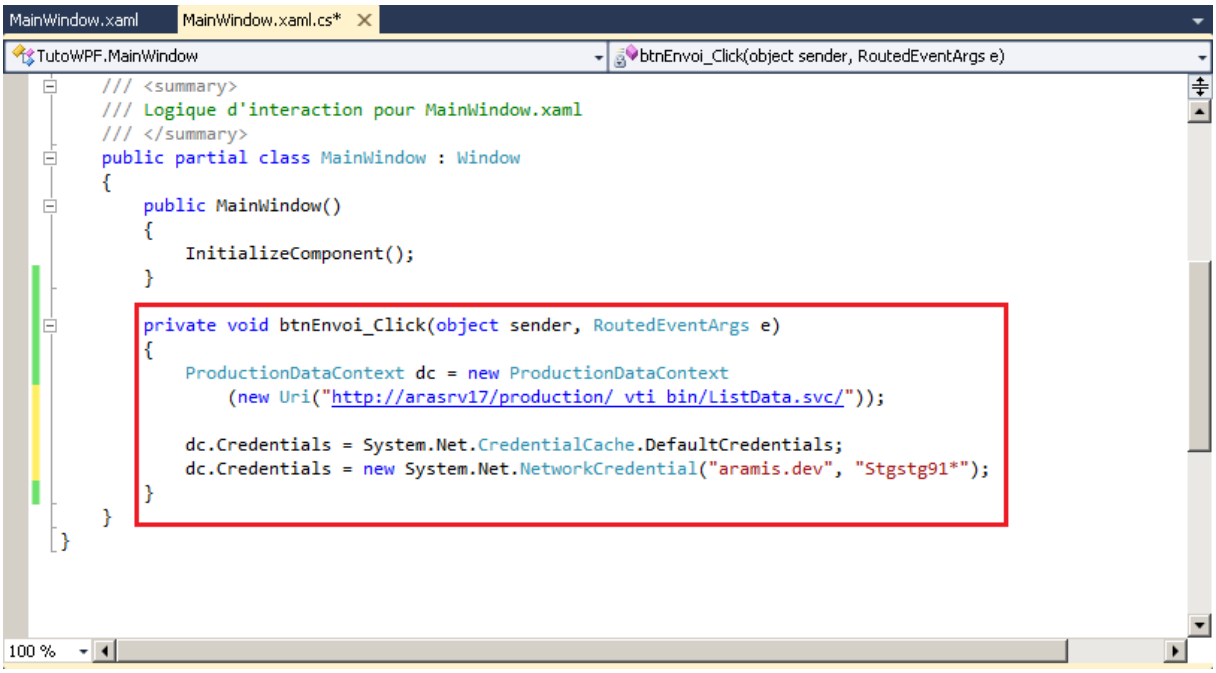
Vu que le code qui va s'exécuter ne se fera que quand on clique sur le bouton « Envoyer !» c'est sur lui que nous allons double-cliquer sur le bouton pour ajouter du code dans le fichier .cs.





La première chose à faire est de connecter votre application à l'intranet. L'instance de connexion sera contenue dans la variable « dc ». Pour l'instancier il faut donc écrire cela sans oublier en paramètre l'adresse de l'intranet que l'on a ajouté référence de service.

Cependant vous avez remarquez qu'il fallait se connecter avec des login. Notre application va donc aussi le faire avec en paramètre le login et le mot de passe :

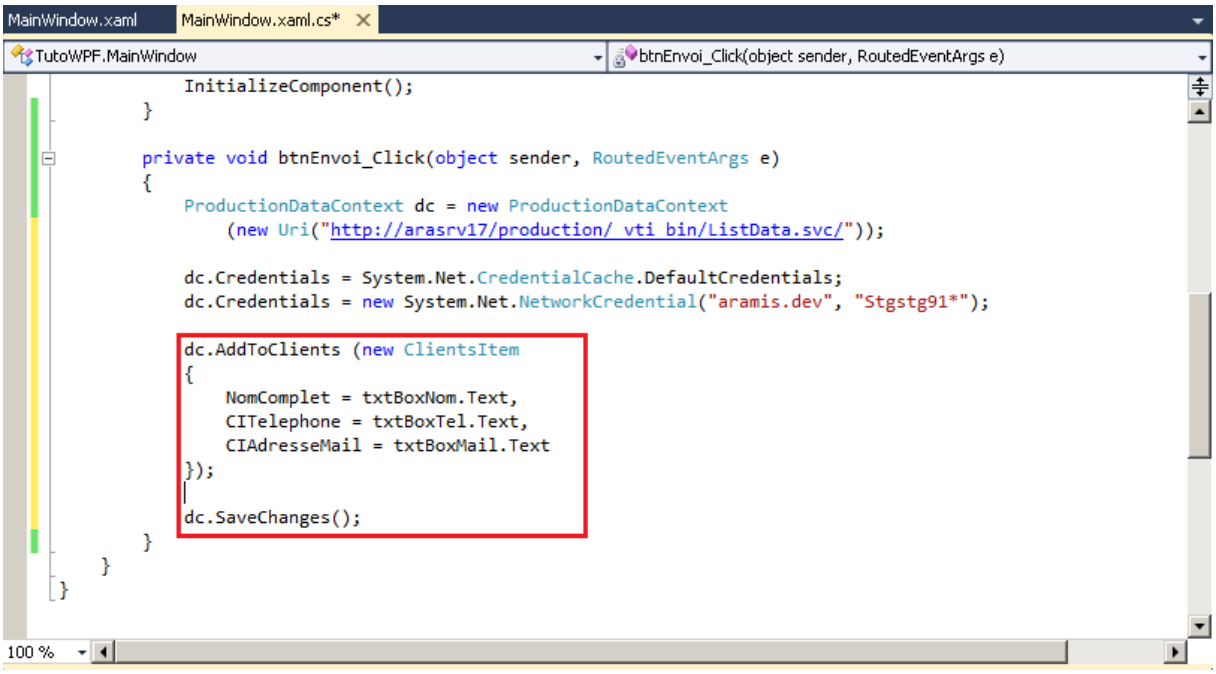


3-Ajouter un élément

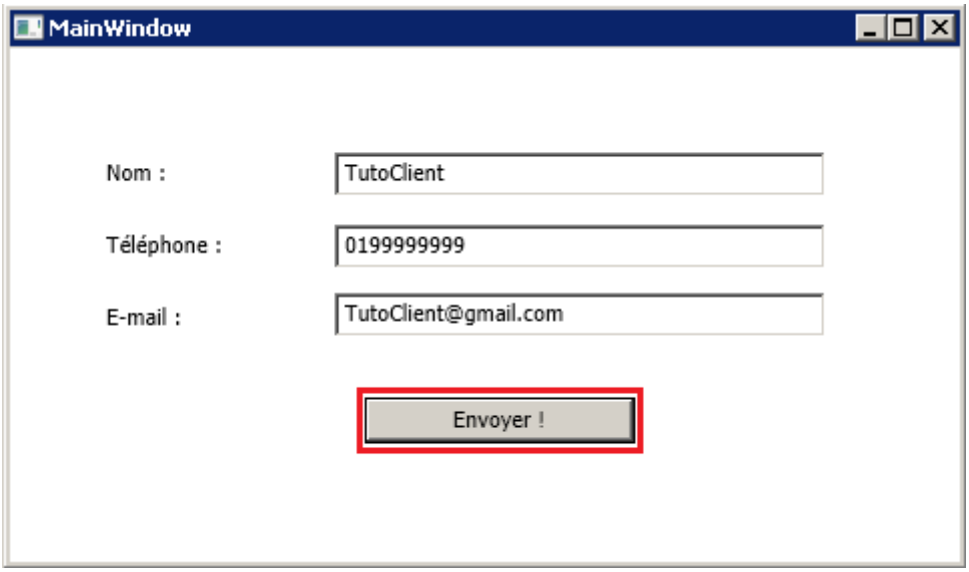
Il est temps maintenant d'ajouter les éléments que l'utilisateur à choisis. Vu que nous ajoutons un client dans la table « Clients » on va le préciser avec notre « AddTo »

Le « new ClientsItem » indique que l'on va ajouter un item, relatif à un client.

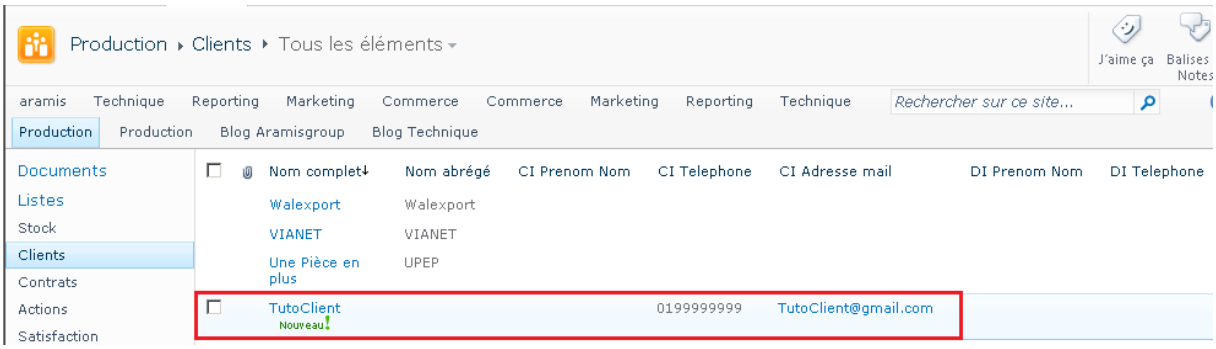
On peut maintenant affecter les informations contenues dans chaques Textbox dans nos colonnes SharePoint. « dc.SaveChanges() ; » permet de sauvegarder l'ajout dans la liste.



Une fois tout cela fait. Vous pouvez déjà lancer une première fois votre application avec F5. Remplissez les champs et faites « Envoyer !».



Voila votre nouveau client à été ajouté à la liste SharePoint, vous pouvez aller vérifier par vous-même.

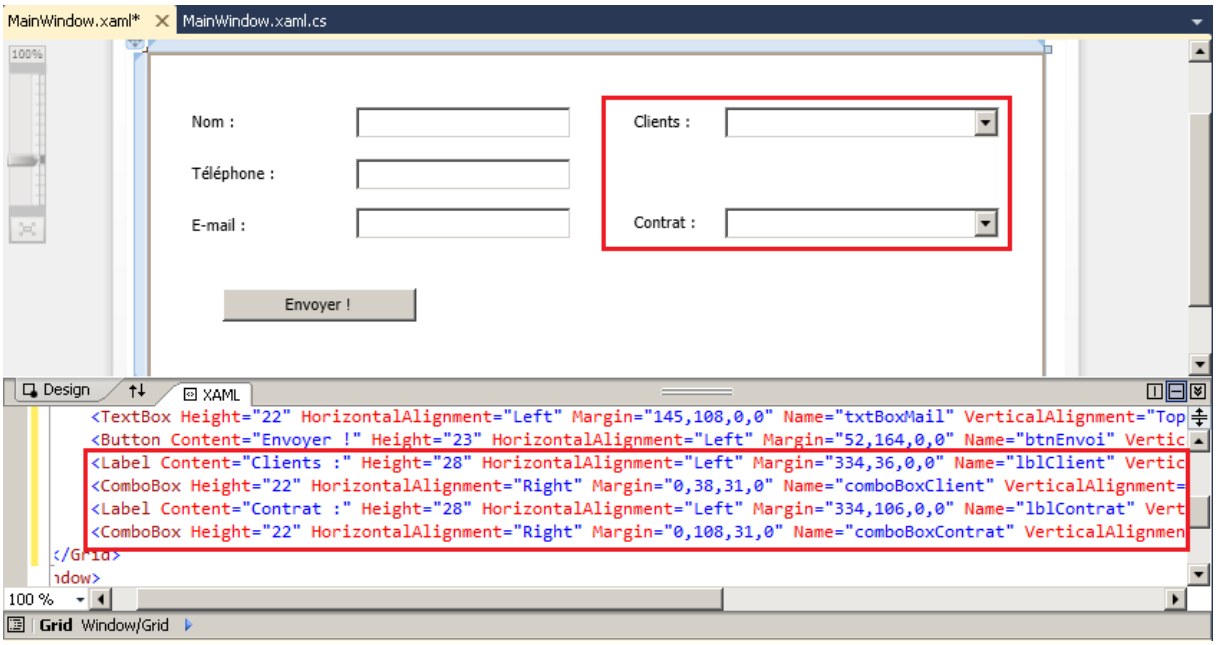


4 – Récupérer un élément

Pour récupérer un élément dans une liste SharePoint nous allons le faire sur la même fenêtre en l'agrandissant. Ceci dit vous pouvez très bien créer une autre fenêtre pour cela si vous vous en sentez capable.

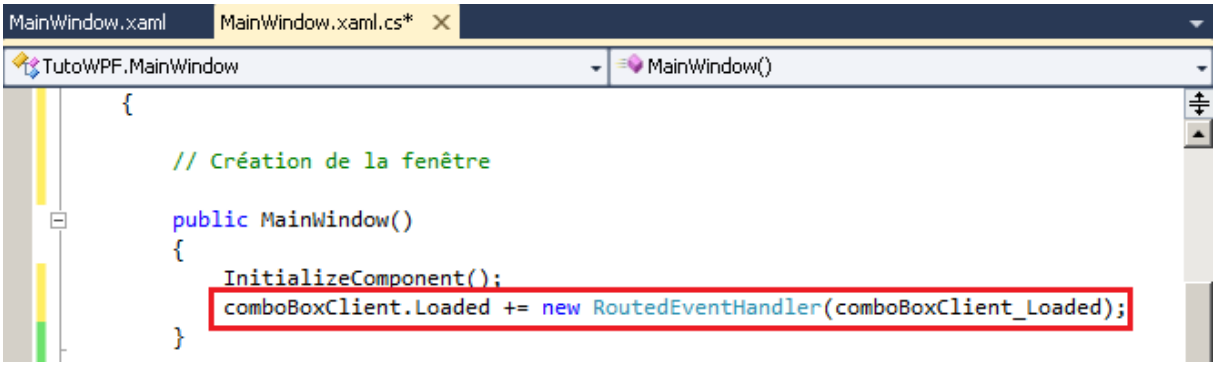
Les clients que nous allons récupérer serons afficher dans une liste déroulante appelé une « combobox ».

Pour la suite du tutoriel où nous allons voir les combobox en cascade nous allons aussi créer une autre combobox « Contrat » dès maintenant.



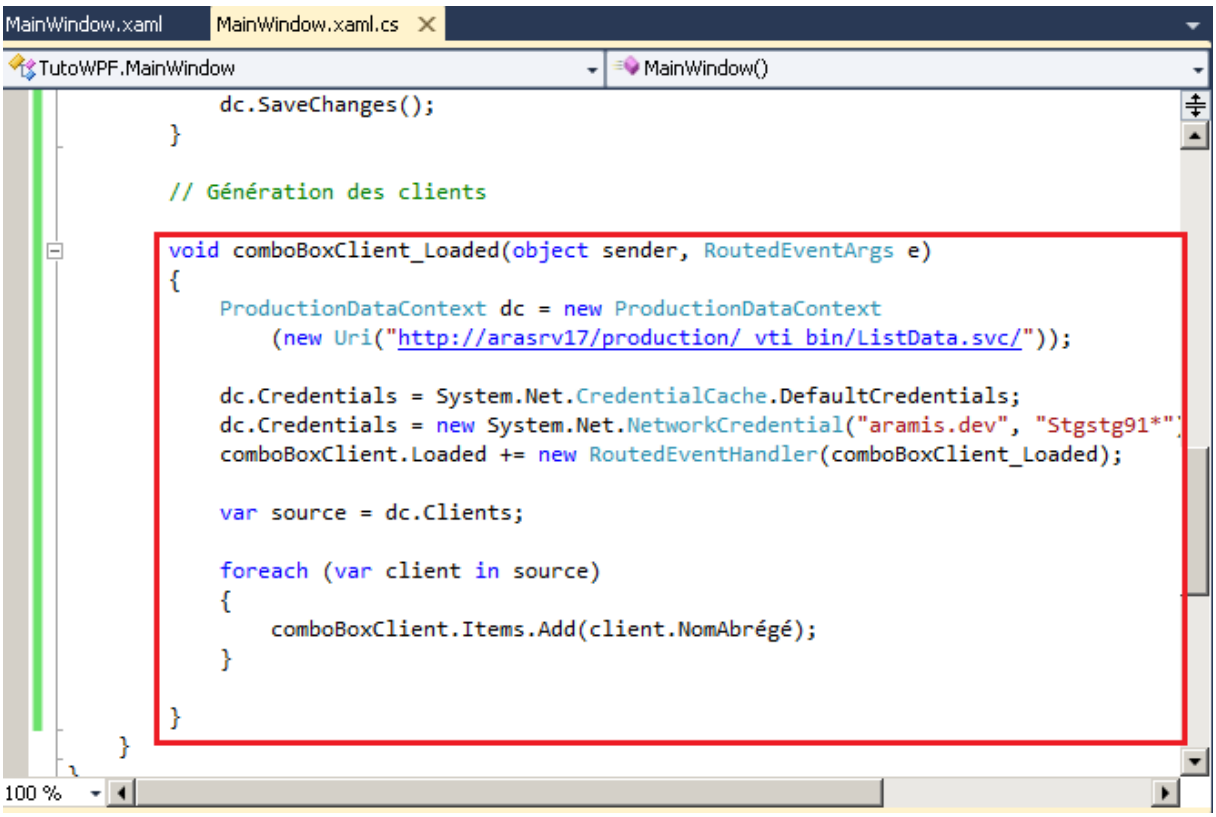
Pour « Client » nous allons créer dans le code deux combobox client, une qui à en paramètre l'événement du changement d'index, le code s'exécutera donc dès que la valeur de la combobox change. L'autre au chargement, donc dès l'arrivé de la page.

On va d'abord s'intéresser au chargement, on à ajouter une petite ligne au début du programme qui va permettre de charger « comboboxClient_Loaded ».



On recréer la connexion car on va de nouveau interroger SharePoint

Et enfin on rentre le code qui va récupérer tous les éléments de la liste « Clients » puis nous les ajoutons un à un avec le foreach dans notre combobox.



Maintenant passons à notre fameuse cascade. Avant cela, petit rappel. Les listes qui sont liés entre elles sur SharePoint comme « Clients » et « Contrats » on des colonnes qui s'appellent des « look up ». Pour retrouver donc le client attaché à ce contrat on va devoir avec SharePoint travailler avec des ID.

On va donc d'abord rajouter une collection en variable private « listeContrat », ainsi que « idClient »

On ajoute ensuite deux fonction, une pour avoir le l'id du client, et une pour avoir l'id du contrat. On retrouve la même structure pour récupérer la liste et la parcourir. On va par contre, faire des tests sur chaque élément pour ne garder que ceux qui nous intéresse.

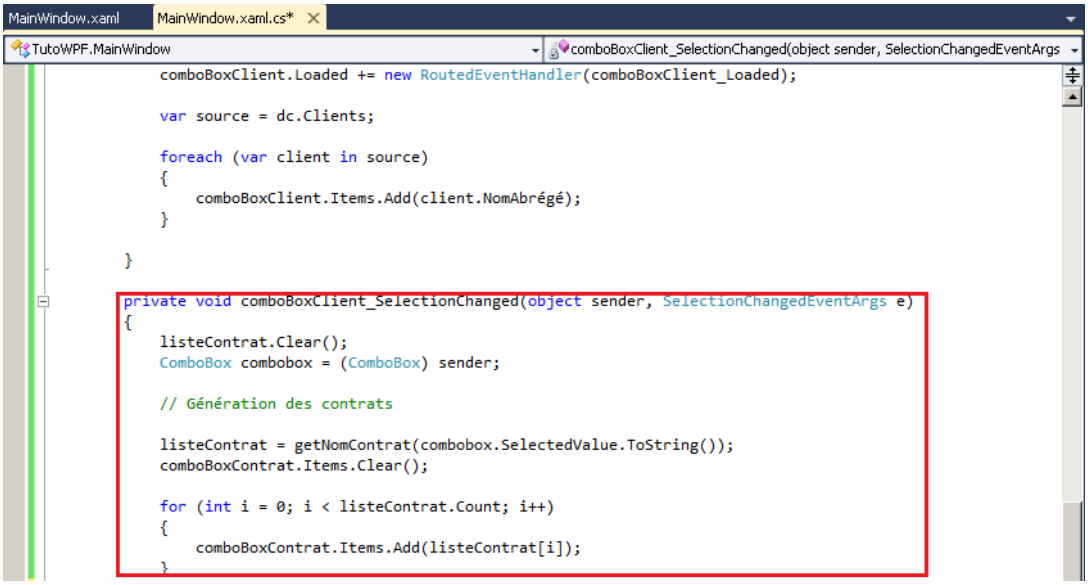
MainWindow.xaml.cs*

TutoWPF.MainWindow — getNomContrat(string unClient)

```
public int getIdClient(string unNomClient)
{
    ProductionDataContext dc = new ProductionDataContext
        (new Uri("http://arasrv17/production/_vti_bin/ListData.svc/"));

    dc.Credentials = System.Net.CredentialCache.DefaultCredentials;
    dc.Credentials = new System.Net.NetworkCredential("aramis.dev", "Stgstg91*");

    var source = dc.Clients;
    foreach (var client in source)
    {
        if (client.NomAbrégé == unNomClient || client.NomComplet == unNomClient)
        {
            idClient = client.ID;
        }
    }
    return idClient;
}

// Donne une liste de contrat lié à un nom de client

public List<string> getNomContrat(string unClient)
{
    ProductionDataContext dc = new ProductionDataContext
        (new Uri("http://arasrv17/production/_vti_bin/ListData.svc/"));

    dc.Credentials = System.Net.CredentialCache.DefaultCredentials;
    dc.Credentials = new System.Net.NetworkCredential("aramis.dev", "Stgstg91*");

    idClient = getIdClient(unClient);
    var source = dc.Contrats;

    foreach (var contrat in source)
    {
        if (contrat.ClientId == idClient)
        {
            listeContrat.Add(contrat.Contrat);
        }
    }
    return listeContrat;
}
```

Maintenant revenons à la « comboboxClient_SelectionChanged ». A l'intérieur, nous allons mettre le code qui permet de trouver quel contrat est associé à tel client. Vous voyez qu'encore une fois l'algorithme en lui-même n'est pas très différent de celui qui récupère tous les clients.

La seule différence se fait dans l'ajout du « if » qui permet un tri.



Voila maintenant les contrats affichés, sont bien ceux du client sélectionné dans la « comboboxClient ». Vous pouvez relancer votre programme et admirer le résultat.

Avant de finir ce tutoriel nous allons vous refaire un petit récapitulatif de ce que nous venons de voir.

IV – Récapitulatif

- Pour récupérer une liste on utilise :

```
var source = dc.[NomListe];
```

Exemple :

```
var source = dc.Clients;
```

- Pour parcourir une liste on se sert d'un foreach :

```
foreach (var src in source)
                    {
                        string Valeur = src.[ColonneSharePoint];
                    }
```

Exemple :

```
foreach (var client in source)
                    {
                        string actName = client.NomAbrégé;
                    }
```

- Pour ajouter un élément dans une liste SharePoint :

```
dc.AddTo[NomListe](
                new [NomListe]Item
                {
                    [ColonneSharePoint] = [Information] ;
                 });
            dc.SaveChanges();
```

Exemple :

```
            dc.AddToActions(
                new ActionsItem
                {
                    ClientId = getIdClient(comboBoxClient.Text),
                });
            dc.SaveChanges();
```

- Pour ajouter un élément dans une combobox :

```
comboBox[Name].Items.Add(src.[ColonneSharePoint]);
```

Exemple :

```
comboBoxClient.Items.Add(client.NomComplet);
```

Nous sommes arrivés à la fin de notre tutoriel, vous savez maintenant utiliser un projet WPF pour interagir avec un liste SharePoint, vous savez ajouter un élément, en extraire.

Vous pouvez bien évidement allez plus loin, et complexifier encore le programme. Vous pouvez aussi l'optimiser, tous ici à été fait pour que cela soit simple à comprendre.

Ceci n'est donc qu'une étape pour mieux appréhender les WPF mais il ne faut pas en rester là.

Enfin si à la fin de ce tutoriel votre programme ne marche pas, n'hésitez pas à le recommencer depuis le début et à bien lire chaque étape en la comprennent et non juste en recopiant bêtement le code. Si après tout ça vous n'y arriver pas. Il ne vous reste plus qu'a désinstaller Visual Studio ☺.