

Silverlight : Se connecter à une base de données grâce à Linq et WCF

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

Dernière mise à jour :

Ce document décrit pas à pas comment accéder à une base de données dans une application Silverlight.

0 - Introduction.....	3
1 - Pré-requis.....	3
2 - Création de l'application Silverlight.....	3
3 - Linq To SQL.....	4
4 - Le Service WCF.....	7
5 - L'Application Silverlight.....	9
6 - Conclusion.....	14
7 - Liens.....	14
8 - Sources du projet.....	14

0 - Introduction

Dans une application Silverlight il est actuellement impossible d'accéder directement à une base de données. Cela n'empêche qu'il est parfois indispensable de pouvoir le faire.

Dans cet article pour contourner ce problème nous utiliserons un service WCF et une query Linq.

Pour illustrer tout ça nous allons créer une application Silverlight avec un datagrid qui lors du chargement affichera la liste des clients de la base de données Nothwind.

Remarque : Nothwind est une base de données de démonstration pouvant être installée sur SQL Server 2005. Si vous ne l'avez pas vous pouvez la télécharger ici : <http://www.microsoft.com/downloads/details.aspx?FamilyID=06616212-0356-46A0-8DA2-EEBC53A68034&displaylang=en>

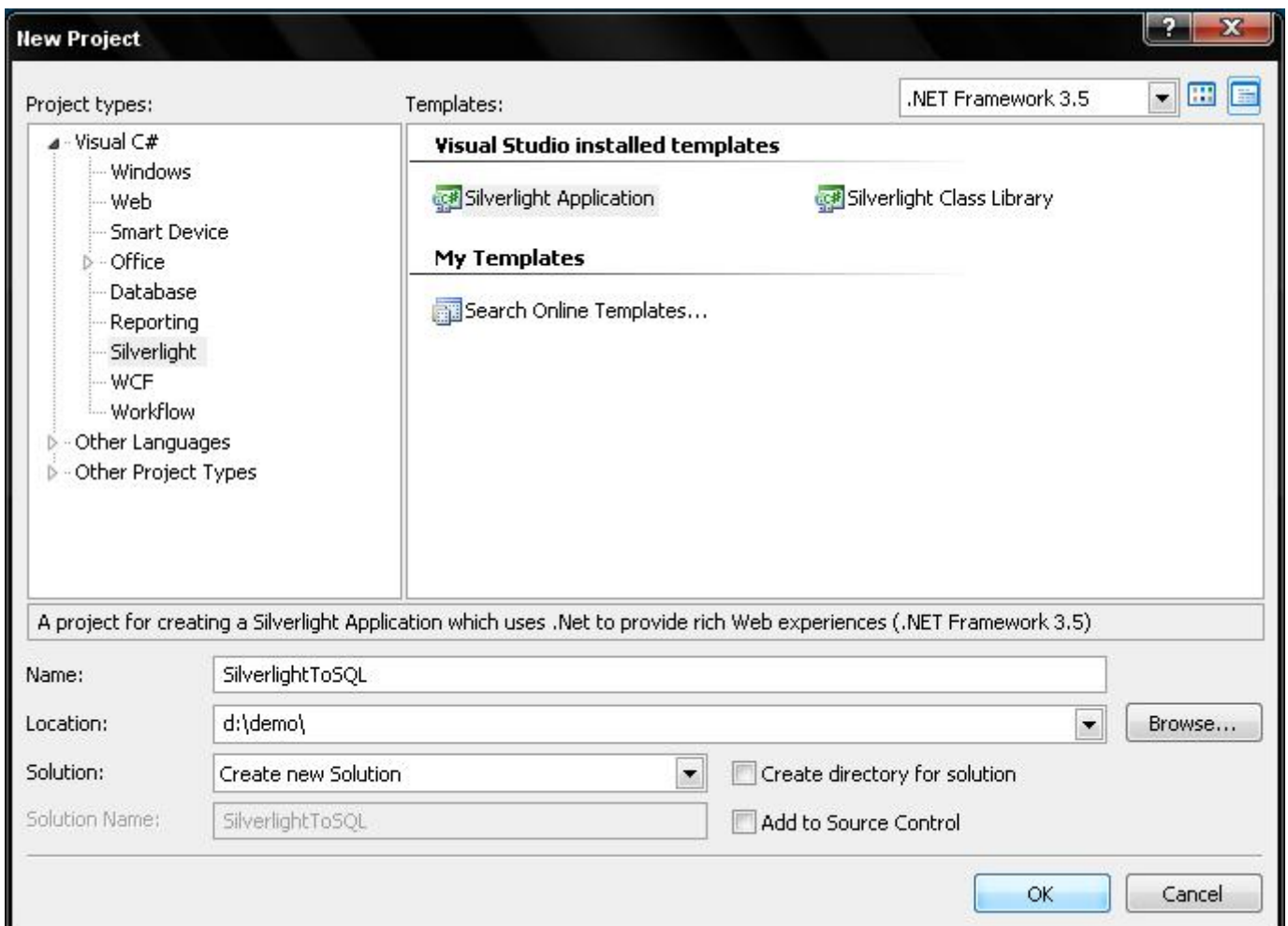
1 - Pré-requis

- Microsoft Visual Studio 2008 (**Virtual pc**)
- **Microsoft Silverlight Tools Beta 1 for Visual Studio 2008**
- .Net Framework 3.5

2 - Création de l'application Silverlight

Notre première étape sera de créer notre projet Silverlight ainsi qu'une application web qui contiendra notre service WCF.

Créez une nouvelle solution dans Visual Studio 2008 et choisissez le type de projet suivant :

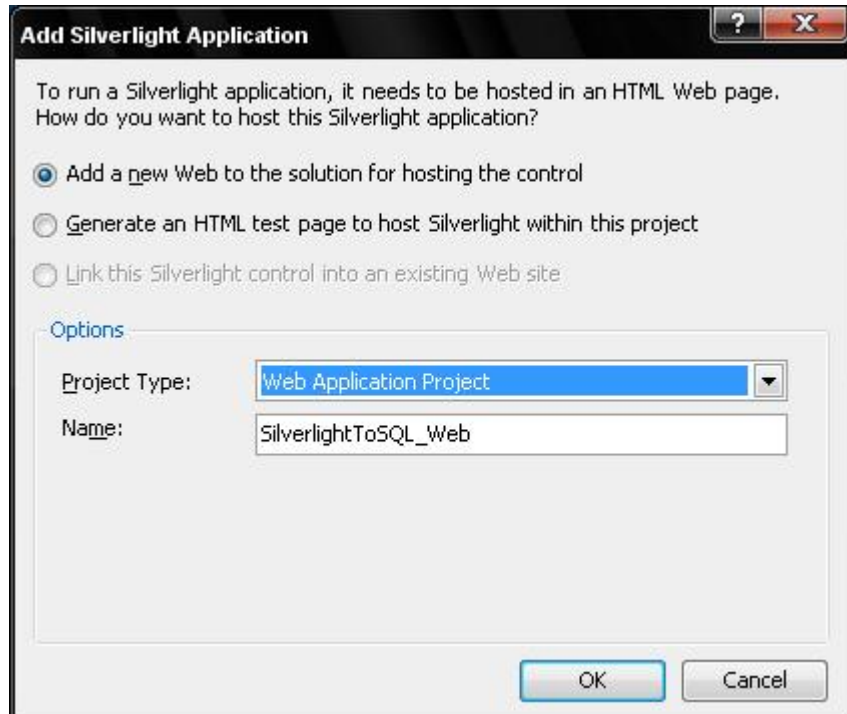


C# > Silverlight > Silverlight Application

Nommez votre projet **SilverlightToSQL** et cliquez sur OK.

Visual Studio va ensuite vous demander si vous souhaitez lier votre projet à une application web.

Nous en avons besoin afin d'écrire notre service WCF, choisissez donc **Web Application Project** dans la liste et cliquez sur OK.



Votre solution contient donc maintenant deux projets :

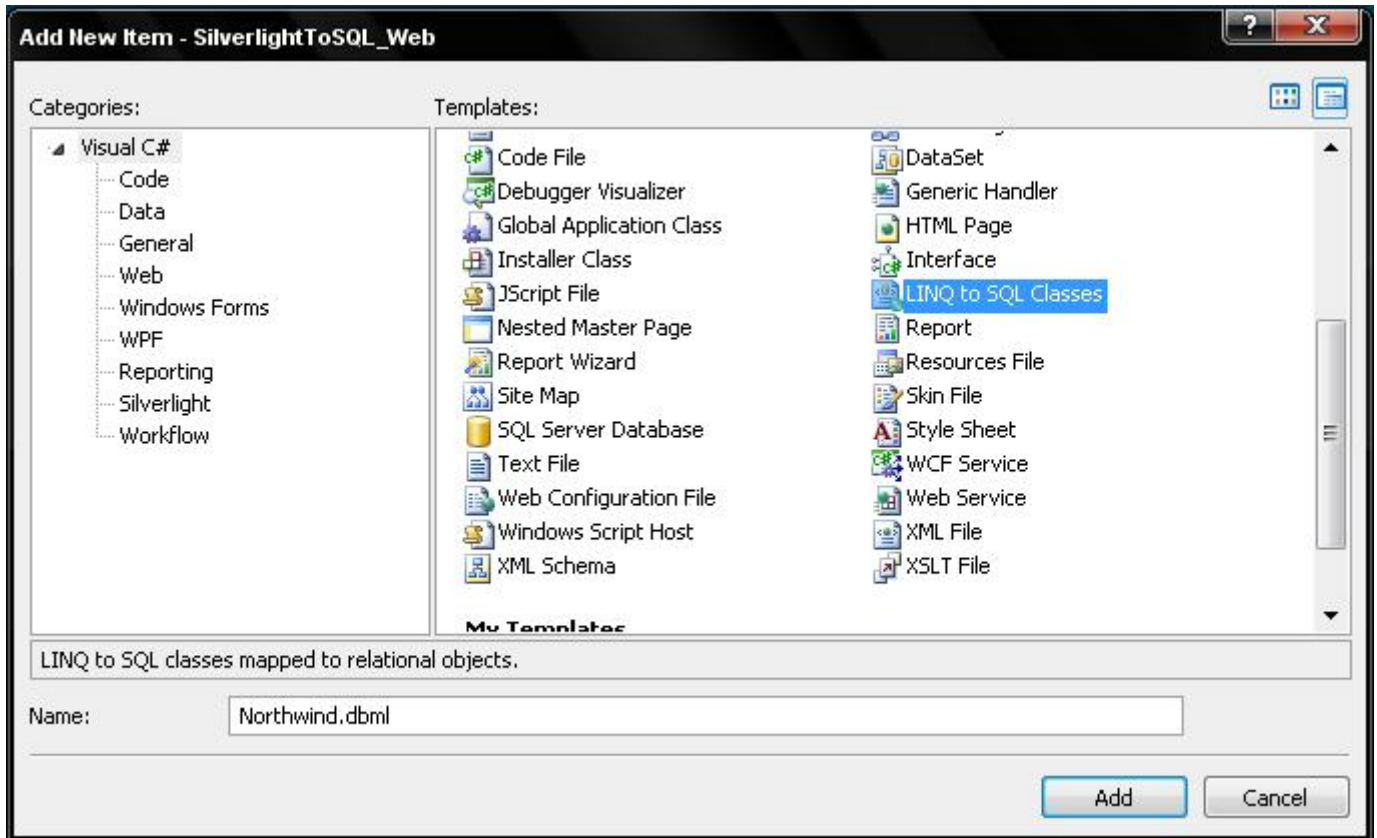
- SilverlightToSQL : L'application Silverlight
- SilverlightToSQL_Web : L'application web qui contiendra le service WCF

3 - Linq To SQL

Linq est une nouveauté de la version 3 de C#, il prétend à être la technique recommandée pour accéder aux bases de données depuis Silverlight.

Si vous souhaitez apprendre les bases de cette technologie je vous invite à lire cet article : <http://morpheus.developpez.com/linq/>.

Revenons à notre exemple. Cliquez droit sur le projet **Web Application** et choisissez **Add new items**



Nous allons ajouter un élément de type **Linq to SQL Classes** et le nommer **Northwind**. Cette classe fera le mapping entre notre objet .Net et la table **Customers** dans la base de données Northwind. Commençons par créer une connexion vers la base de données dans le **Server Explorer** de Visual Studio. Dans mon cas le serveur est en local sur ma machine.

Add Connection

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source:
Microsoft SQL Server (SqlClient) Change...

Server name:
LLE-D830 Refresh

Log on to the server

Use Windows Authentication
 Use SQL Server Authentication

User name:
Password:
 Save my password

Connect to a database

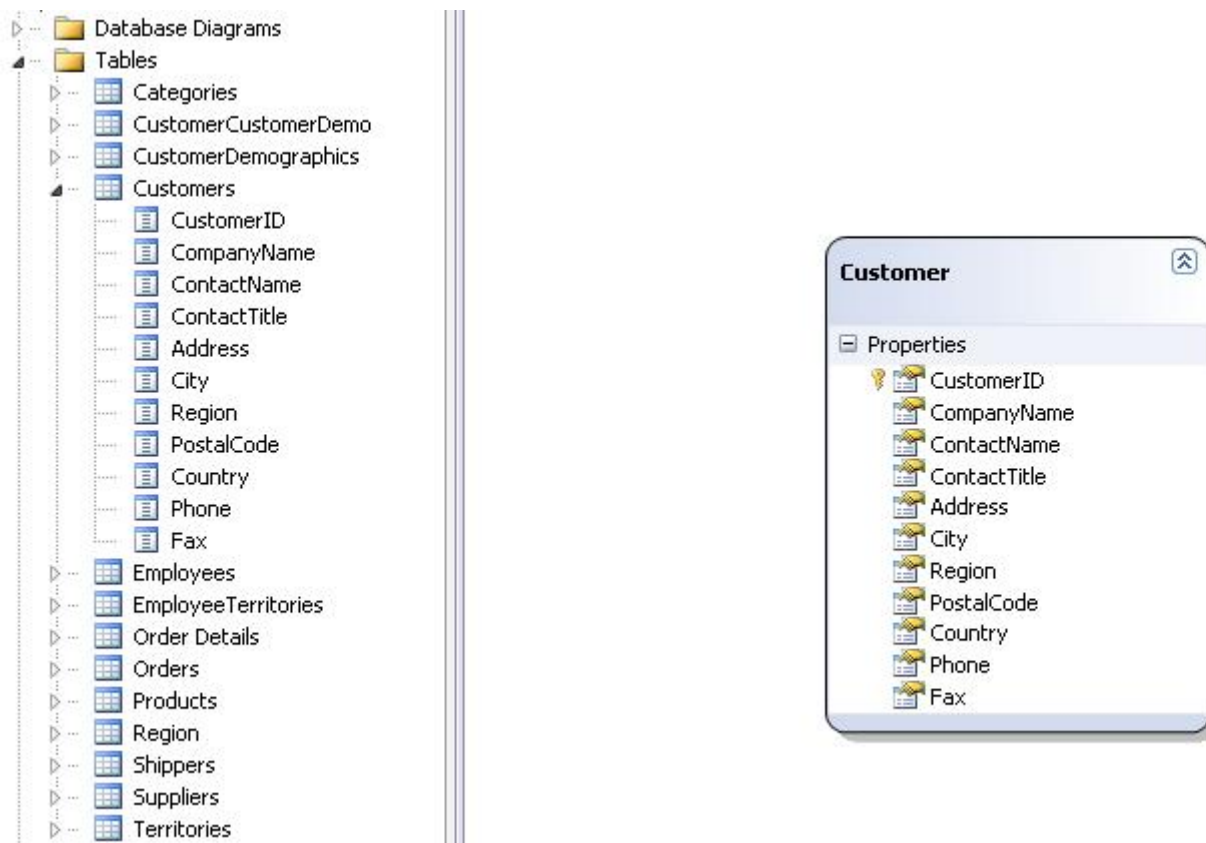
Select or enter a database name:
Northwind

Attach a database file:
 Browse...
Logical name:

Advanced...

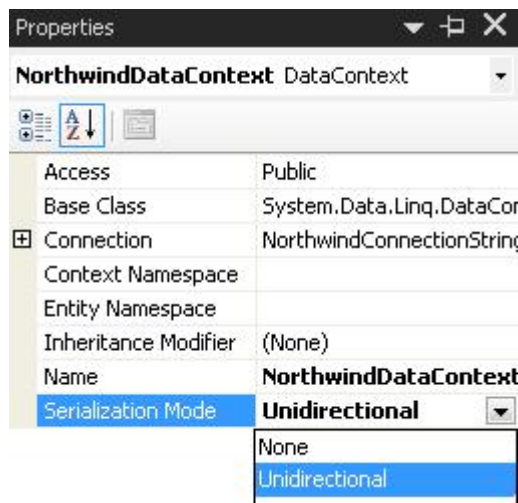
Test Connection OK Cancel

Prochaine étape : ajouter la table **Customers** dans notre schéma pour cela il suffit de "drag and dropper" notre table.



Par défaut cette classe n'est pas sérialisable, cela va poser problème lorsque l'on voudra l'utiliser au travers de notre service WCF.

Pour la rendre sérialisable rien de plus simple il suffit de modifier sa propriété **Serialization Mode** en mettant la valeur **Unidirectional**.

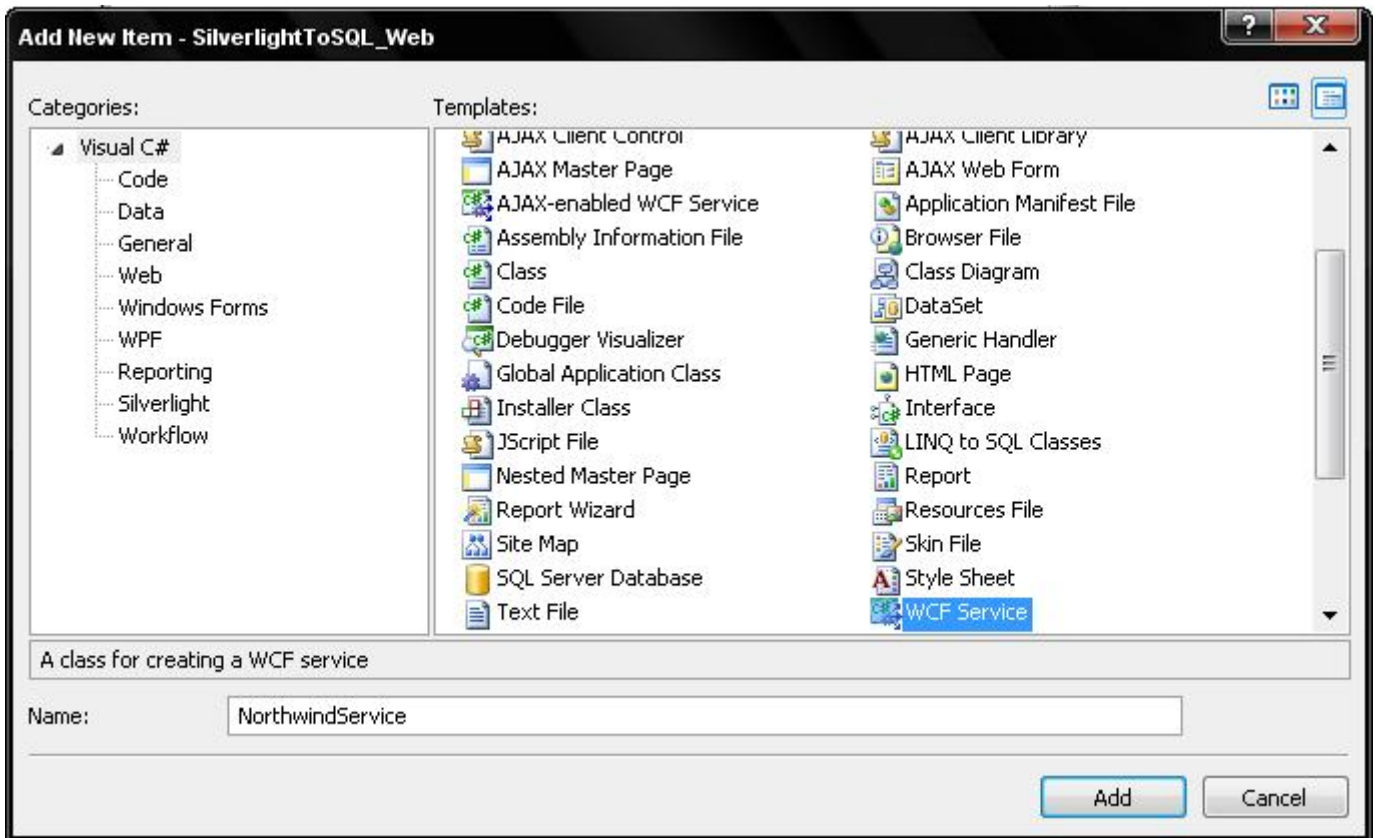


4 - Le Service WCF

Maintenant que notre classe Linq est créée et que donc notre classe **Customer** est connue par le système il nous reste à écrire les méthodes permettant de travailler avec cet objet.

Pour cela nous allons faire appel à un service WCF.

Faites un clic droit sur notre **Web Application** et choisissez **Add > New Item**. Nous allons ajouter un élément de type **WCF Service** qui nous appellerons **NorthwindService**.



Comme vous l'avez certainement remarqué notre service est composé de deux fichiers sources :

- **INorthwindService.cs** : Interface
- **NorthwindService.scv.cs** : Implémentation de l'interface

Commençons par définir deux méthodes dans notre interface :

INorthwindService.cs

```

[ServiceContract]
public interface INorthwindService
{
    [OperationContract]
    List<String> getCountries();

    [OperationContract]
    List<Customer> getCustomers(string countryName);
}
    
```

Notre service relativement simpliste contiendra deux méthodes : une première retournant la liste des pays se trouvant dans la table Customers et une deuxième pour obtenir tout les clients liés au pays sélectionné.

Les attributs **[ServiceContract]** et **[OperationContract]** sont indispensables au bon fonctionnement du service.

Voyons à présent l'implémentation de ces méthodes :

INorthwindService.cs

```

public class NorthwindService : INorthwindService
{
    public List<string> getCountries()
    {
        NorthwindDataContext dc = new NorthwindDataContext();
    }
}
    
```

INorthwindService.cs

```
var AllCountries = (from countries in dc.Customers orderby countries.Country select
countries.Country).Distinct();
return AllCountries.ToList();
}

public List<Customer> getCustomers(string countryname)
{
NorthwindDataContext dc = new NorthwindDataContext();
var CustomerForCountry = from customers in dc.Customers orderby customers.ContactName where
customers.Country == countryname select customers;
return CustomerForCountry.ToList() ;
}
}
```

Deux concepts nouveaux depuis C# 3.0 sont utilisés dans ce code : l'inférence de type et Linq.

Linq permet d'accéder une base de données depuis votre code c# sans écrire la moindre ligne de SQL, une vraie révolution pour les développements futurs !

Afin de rendre ce code fonctionnel depuis Silverlight une petite modification du fichier web.config est obligatoire.

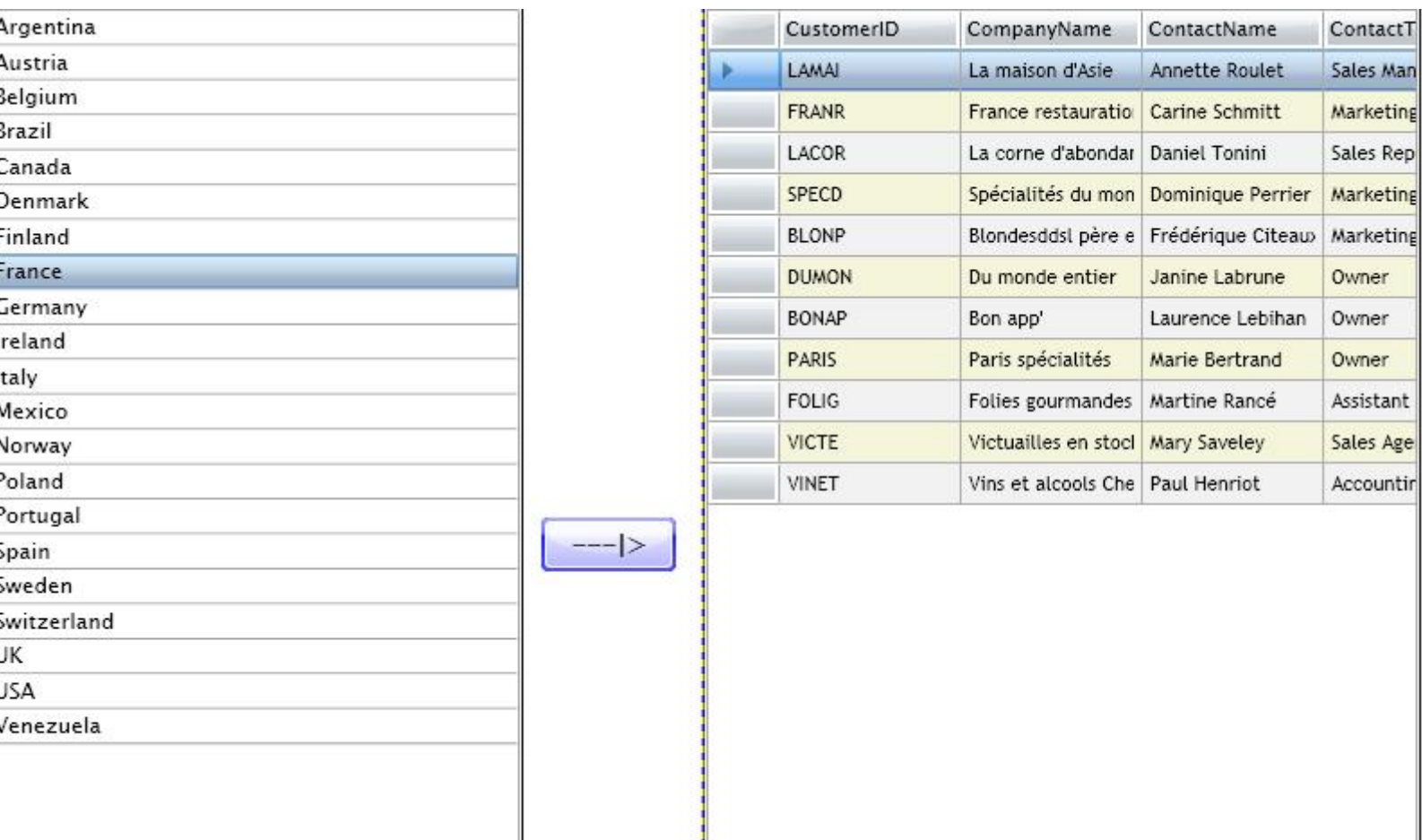
En effet le binding par défaut est **wsHttpBinding**, pour fonctionner avec Silverlight il faut utiliser du **basicHttpBinding**

web.config

```
<endpoint address="" binding="basicHttpBinding" contract="SilverlightToSQL_Web.INorthwindService">
<identity>
<dns value="localhost"/>
</identity>
</endpoint>
```

5 - L'Application Silverlight

Passons maintenant à la création de notre application Silverlight proprement dite, voici ce à quoi elle devra ressembler :



L'interface devra contenir une liste avec tout les pays à gauche. Lorsque l'utilisateur en sélectionne un et clique sur la flèche la datagrid de droite se remplit avec tout les clients liés au pays sélectionné.

Voici le code de l'interface :

```

Page.xaml
<UserControl xmlns:my="clr-
namespace:System.Windows.Controls;assembly=System.Windows.Controls.Data" x:Class="SilverlightToSQL.Page"
xmlns="http://schemas.microsoft.com/client/2007"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
Width="800" Height="600">
<Grid x:Name="LayoutRoot" Background="White" ShowGridLines="True">

    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="400"/>
        <ColumnDefinition Width="400"/>
    </Grid.ColumnDefinitions>

    <StackPanel Grid.Row="1" Grid.Column="0" Orientation="Horizontal">

        <ListBox x:Name="lstCountries" Width="300" ></ListBox>
        <Button x:Name="Search" Width="75" Height="30" Margin="10,0,0,,4" Content="---|
>" VerticalAlignment="Center" Background="Blue" FontWeight="Bold" FontSize="14"/>
    </StackPanel>

    <my:DataGrid x:Name="theDataGrid" AlternatingRowBackground="Beige" AutoGenerateColumns="True" Width="400" Height="
    </Grid>
</UserControl>

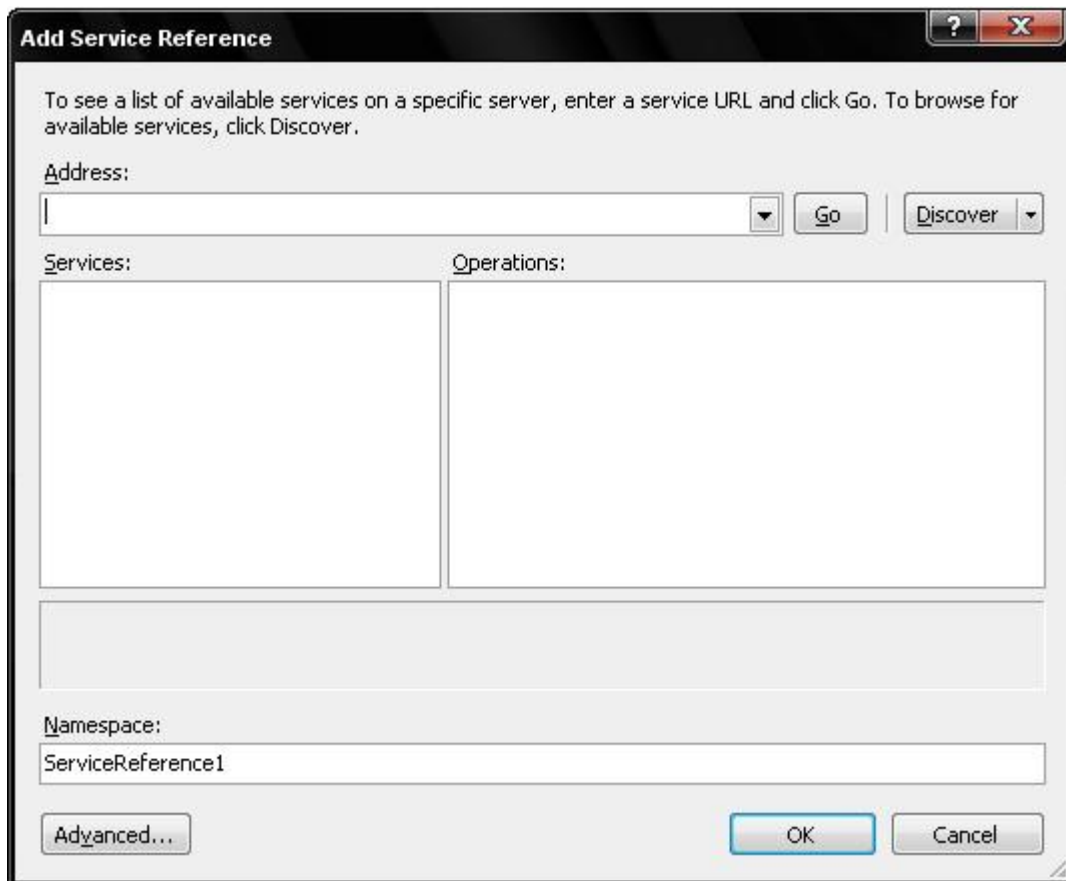
```

Nous avons créé une application Silverlight et un service WCF, reste maintenant à lier les deux. Pour cela : clic droit sur le projet silverlight --> add Service Reference.

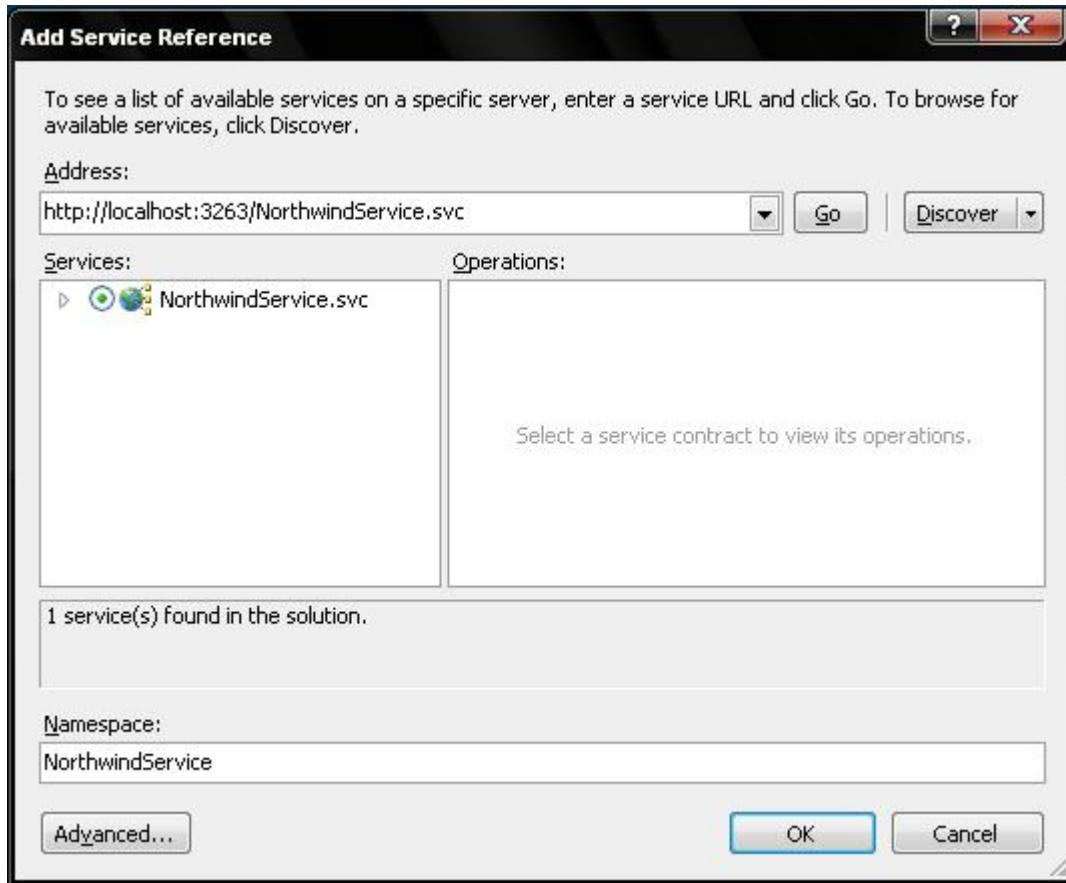
Vous obtenez alors une fenêtre permettant de sélectionner un service :



Ouvrez ensuite le menu discover et choisissez "Service in solution" pour ajouter un service présent dans la solution :



Choisissez NortwindService.svc dans la liste et appelez le **NorthwindService** :



Reste ensuite à utiliser le proxy que nous venons de créer. Modifions donc notre code dans le fichier page.xaml.cs :

Page.xaml.cs

```
public partial class Page : UserControl
{
    public Page()
    {
        InitializeComponent();
        Loaded += new RoutedEventHandler(Page_Loaded);
    }

    void Page_Loaded(object sender, RoutedEventArgs e)
    {
        NorthwindService.NorthwindServiceClient client = new
SilverlightToSQL.NorthwindService.NorthwindServiceClient();
        client.getCountriesCompleted += new
EventHandler<SilverlightToSQL.NorthwindService.getCountriesCompletedEventArgs>(client_getCountriesCompleted);
        client.getCountriesAsync();
        Search.Click += new RoutedEventHandler(Search_Click);
    }

    void client_getCountriesCompleted(object sender,
SilverlightToSQL.NorthwindService.getCountriesCompletedEventArgs e)
    {
        lstCountries.ItemsSource = e.Result;
    }

    void Search_Click(object sender, RoutedEventArgs e)
    {
        NorthwindService.NorthwindServiceClient client = new
SilverlightToSQL.NorthwindService.NorthwindServiceClient();
        client.getCustomersCompleted += new
EventHandler<SilverlightToSQL.NorthwindService.getCustomersCompletedEventArgs>(client_getCustomersCompleted);
        client.getCustomersAsync(lstCountries.SelectedItem.ToString());
    }
}
```

Page.xaml.cs

```

    }

    void client_getCustomersCompleted(object sender,
SilverlightToSQL.NorthwindService.getCustomersCompletedEventArgs e)
    {
        theDataGrid.ItemsSource = e.Result;
    }
}

```

Décrivons ce code :

Dans le constructeur nous avons ajouté un eventhandler sur le chargement de l'application :

```
Loaded += new RoutedEventHandler(Page_Loaded);
```

Voyons donc le code qui sera déclenché lors du chargement de l'application :

Page_Loaded

```

void Page_Loaded(object sender, RoutedEventArgs e)
{
    NorthwindService.NorthwindServiceClient client = new
SilverlightToSQL.NorthwindService.NorthwindServiceClient();
    client.getCountriesCompleted += new
EventHandler<SilverlightToSQL.NorthwindService.getCountriesCompletedEventArgs>(client_getCountriesCompleted);
    client.getCountriesAsync();
    Search.Click += new RoutedEventHandler(Search_Click);
}

```

Après avoir instancié notre service, nous ajoutons un eventhandler sur l'event **getCountriesCompleted** et appelons la méthode **getCustomersAsync** de manière asynchrone.

Pour finir ajoutons un eventhandler sur le clic du bouton "flèche" qui chargera notre DataGrid.

Remarque : Dans Silverlight les services ne peuvent être appelés que de manière asynchrone

L'étape suivant consiste à récupérer l'event lancé lors de la réponse du Service avec la liste des pays :

client_getCountriesCompleted

```

void client_getCountriesCompleted(object sender,
SilverlightToSQL.NorthwindService.getCountriesCompletedEventArgs e)
{
    lstCountries.ItemsSource = e.Result;
}

```

D'une simplicité déconcertante ce code va remplir notre liste de droite avec les pays retournés par le service WCF. Occupons nous maintenant de l'appui sur le bouton de recherche (-->) :

Search_Click

```

void Search_Click(object sender, RoutedEventArgs e)
{
    NorthwindService.NorthwindServiceClient client = new
SilverlightToSQL.NorthwindService.NorthwindServiceClient();
    client.getCustomersCompleted += new
EventHandler<SilverlightToSQL.NorthwindService.getCustomersCompletedEventArgs>(client_getCustomersCompleted);
    client.getCustomersAsync(lstCountries.SelectedItem.ToString());
}

```

Exactement comme précédemment nous allons créer une nouvelle instance de notre service, attacher une méthode à l'event **getCustomersCompleted** et exécuter la méthode.

Dernière étape : afficher le résultat dans le datagrid :

Search_Click

```
void client_getCustomersCompleted(object sender,
    SilverlightToSQL.NorthwindService.getCustomersCompletedEventArgs e)
{
    theDataGrid.ItemsSource = e.Result;
}
```

Voilà il ne vous reste plus maintenant qu'à exécuter l'application et observer le résultat !

6 - Conclusion

Au travers de cet article j'ai pu vous démontrer la simplicité avec laquelle il est possible de se connecter à une base de données à partir d'une application Silverlight.

Bien entendu comme dans toutes "interface riche" le plus gros du travail reste le design ...

7 - Liens

Mes articles : <http://lefortludovic.developpez.com>

Mon blog : <http://blogs.ezos.com/blog/le>

8 - Sources du projet

Cliquez [ici](#) pour télécharger les sources de ce projet.

Je tiens à remercier [RideKick](#) pour la correction orthographique de cet article.