Silverlight 4 + RIA Services - Prêt pour les affaires

Consommer des données dans le client Silverlight

par Brad Abrams (Blog) Deepin Prayag (Traduction) (Home)

Date de publication : 14/02/2012

Dernière mise à jour : 20/02/2012

Cet article fait partie d'une série de traductions d'articles de Brad Abrams sur le développement d'applications métier avec Silverlight 4 et .NET RIA Services.

Cette série se concentre uniquement sur la base des applications métier : l'interrogation, la mise à jour, la validation et la sécurisation de vos données métier importantes.

Elle sera également utilisée pour mettre à jour certains billets de la **série Silverlight 3**.

Traduction

Cet article est la traduction la plus fidèle possible de l'article original de **Brad Abrams**, **Silverlight 4 + RIA Services - Ready for Business: Consuming Data in the Silverlight Client**

Prérequis

La procédure pas à pas requiert :

- **Visual Studio 2010** (ou la **version express gratuite**)
- **Silverlight 4 Tools** (inclut **RIA Services**)

Vous pouvez **télécharger l'application complète**.

J'ai implémenté cela avec Silverlight 4 RC, mais je m'attends à ce que cela fonctionne avec Silverlight 4 RTM.

Consommer des données dans le client Silverlight.

Pour **continuer notre série**, voyons d'où vient le plaisir quand je regarde la facilité avec laquelle on peut consommer du client. D'abord, juste pour vous aider à comprendre ce qui se passe derrière la porte, jetons un coup d'oeil à une solution code-behind. Dans View\Home.xaml mettez un simple DataGrid sur le formulaire.

```
<sdk:DataGrid Name="dataGrid1" Height="152" Width="692" />
```

Ensuite ajoutez ces lignes de code à Home.xaml.cs

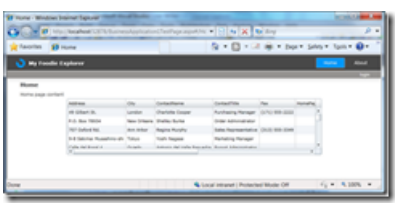
```
1: var context = new DishViewDomainContext();
2: this.dataGrid1.ItemsSource = context.Restaurants;
3:
4: context.Load(context.GetRestaurantsQuery());
5:
```

A la ligne 1, nous créons un DishViewDomainContext. Remarquez que cela est automatiquement généré (via une tache MSBuild) à partir du DishViewDomainService sur le serveur.

A la ligne 2, vous pouvez remarquer que nous avons une propriété Restaurants - nous fournissons cette propriété car il y a au moins une méthode de requête qui renvoie les restaurants. Remarquez l'effet magique de la liaison de données. Nous n'avons pas encore rempli ces données à partir du serveur, mais nous allons de l'avant et les lions. Cela évite toute sorte de logique de rappel complexe.

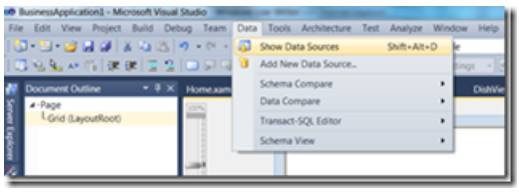
A la ligne 4, nous chargeons explicitement les données. C'est l'appel du réseau, donc nous voulons être sûrs que ce soit clair pour le développeur quand cela arrive. Comme argument, nous passons la méthode de requête exacte (et tous les arguments) à utiliser.

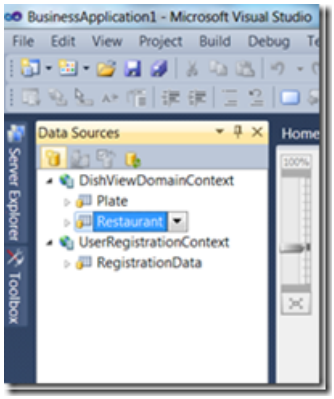
Maintenant nous l'exécutons et obtenons nos données.



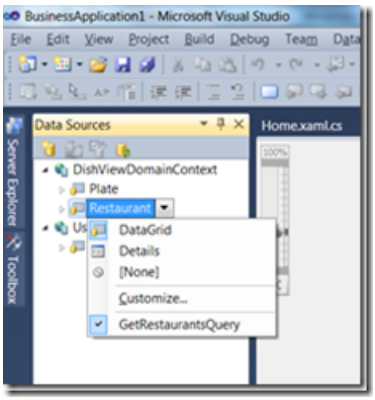
Sans gros efforts, mais cela devient encore plus facile ;-)

Allez-y et effacez ce code et le XAML. Ensuite prêtez attention à la fenêtre DataSources



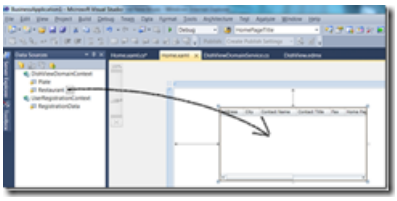


Remarquez que ceci est simplement une représentation visuelle exacte de ce que nous faisions dans le code.

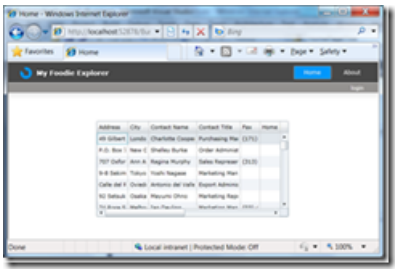


Remarquez que je peux modifier l'interface utilisateur par défaut à générer et quelle méthode de requête utiliser.

Faites glisser *Restaurants* sur le formulaire et bingo, nous avons nos données.



Exécutez-le et nous avons nos données. Qu'est ce qui pourrait être plus facile ?

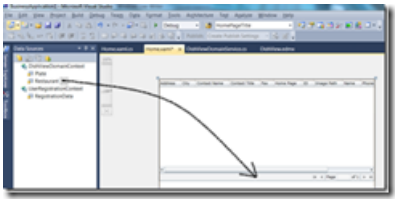


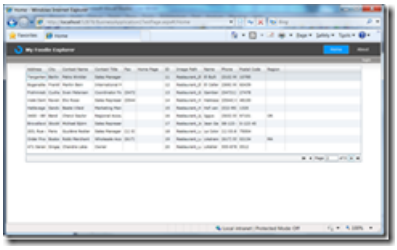
Cliquez sur les entêtes de colonne. Remarquez que le tri fonctionne ! Vous n'aviez pas à écrire de code sur le client ou le serveur pour l'activer. Cela survient gratuitement en retournant un IQueryable.

Maintenant ajoutons la pagination...

Tout d'abord nous glissons et déposons le contrôle DataPager sur le formulaire, ensuite nous faisons un peu de mise en page et nous nous retrouvons avec quelque chose qui semble pas mal.

Mais il nous faut encore relier le DataPager au même DataSource sous-jacent. Cela est assez facilement effectué en glissant et déposant le même élément *Restaurants* de la fenêtre DataSources sur le DataPager. Vous saurez que vous l'avez bien fait si le DataPager est activé.

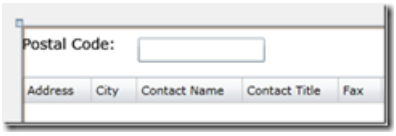




Remarquez que la pagination fonctionne à merveille. Encore une fois, pas de code nécessaire sur le client ou le serveur ; tout est fait en utilisant la simple requête LINQ que nous avons écrite sur le serveur. Mais qu'en est-il de la pagination ET du tri. Fonctionnent-ils bien ensemble ? Triera-t-il seulement la page de données qui est chargée localement ? La réponse, bien sur, est que le tri et la pagination fonctionne très bien ensemble et que le tri est en fait envoyé à la couche de données et exécuté là-bas d'une manière très efficace.

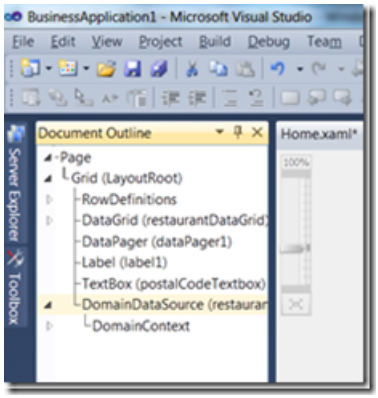
Maintenant ajoutons le filtrage. Avec un peu plus de travail dans l'interface utilisateur, nous pouvons ajouter un moyen de filtrer par code postal.

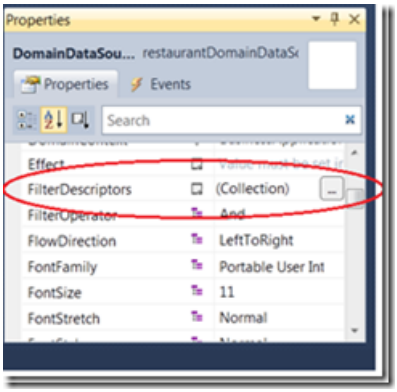
```
<sdk:Label Name="label1" Content="Postal Code:" FontSize="14" Margin="0,0,462,13" />
<TextBox Height="23" HorizontalAlignment="Left" Margin="110,12,0,0" Name="postalCodeText
```

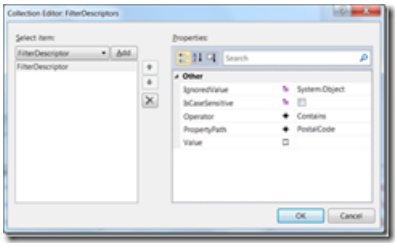


Ensuite nous devons relier cela à la requête. Bien sur, nous voulons que le filtre soit appliqué au plus près des données que possible. Nous ne voulons pas télécharger toutes les entités localement pour ensuite les filtrer. Nous ne voulons pas non plus toutes les tirer vers la couche intermédiaire et ensuite les filtrer. Ce que nous voulons, c'est que le filtre soit appliqué au niveau même de la base de données. Heureusement, cela est très facile à faire avec la composition LINQ.

D'abord nous sélectionnons le DomainDataSource dans la vue Document Outline.



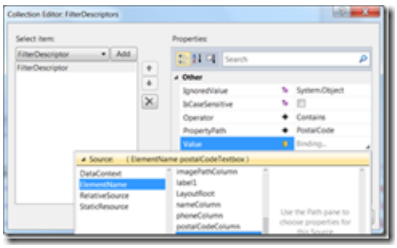




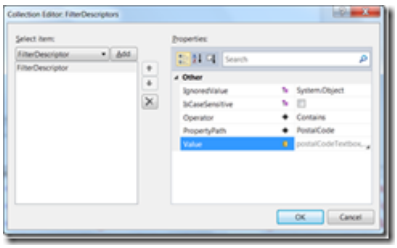
Operator - typiquement vous devriez définir cela à « Contains ». Si vous utilisez le « IsEqualTo » par défaut, le premier chargement (quand il n'y a pas de filtre) se traduira par aucun résultat renvoyé.

PropertyPath - ceci est la propriété de l'entité sur laquelle vous filtrez. Tapez seulement le nom simple.

Value - c'est là où on va obtenir la valeur avec laquelle comparer. C'est plus facile de faire une liaison Interface Utilisateur-à-Interface Utilisateur à la propriété Text du TextBox.

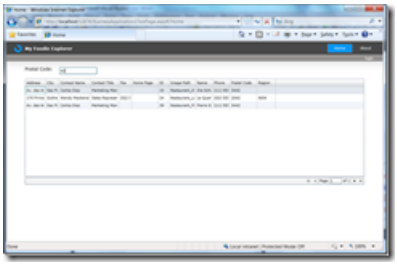


Voici la boîte de dialogue une fois que nous avons fini :



Et le XAML resultant :

```
<riaControls:DomainDataSource Name="restaurantDomainDataSource" AutoLoad="True"
                              d:DesignData="{d:DesignInstance my:Restaurant,
                                             CreateList=true}"
                              Height="0"  Width="0"
                              LoadedData="restaurantDomainDataSource_LoadedData_1"
                              QueryName="GetRestaurantsQuery">
        <riaControls:DomainDataSource.FilterDescriptors>
            <riaControls:FilterDescriptor Operator="Contains"
                                          PropertyPath="PostalCode"
                                          Value="{Binding ElementName=postalCodeTextbox, Path=Text}" />
        </riaControls:DomainDataSource.FilterDescriptors>
        <riaControls:DomainDataSource.DomainContext>
            <my:DishViewDomainContext />
        </riaControls:DomainDataSource.DomainContext>
</riaControls:DomainDataSource>
```



Remarquez qu'aucun changement particulier n'a été effectué, tant au niveau du code que de la logique métier dans le DomainService.

Conclusion

Ceci conclut la troisième partie de cette série. Dans la prochaine partie nous verrons comment mettre à jour les données dans le client Silverlight.

Liens

- **Visual Studio 2010**
- **Visual Studio 2010 Express**

- **Silverlight 4 Tools**
- **RIA Services**
- **Télécharger l'application complète**

Remerciements

Je tiens ici à remercier **Brad Abrams** pour son aimable autorisation de traduire l'article.
Je remercie **Jean-Michel Ormes** pour sa relecture technique et ses propositions.
Je remercie également **xyz** pour sa relecture orthographique et ses propositions.