Silverlight 4 + RIA Services - Prêt pour les affaires

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

Date de publication : 11/25/2011

Dernière mise à jour : 11/25/2011

Série d'articles sur les applications métier avec Silverlight 4 et RIA Services.

Introduction générale

Cette série d'articles est la traduction de 'Silverlight 4 + RIA Services - Ready for Business' de Brad Abrams.

I - Traduction

Cet article est la traduction la plus fidèle possible de l'article original de **Brad Abrams**, **Silverlight 4 + RIA Services - Ready for Business: Starting a New Project with the Business Application Template**

II - Prérequis

La procédure pas à pas requiert :

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

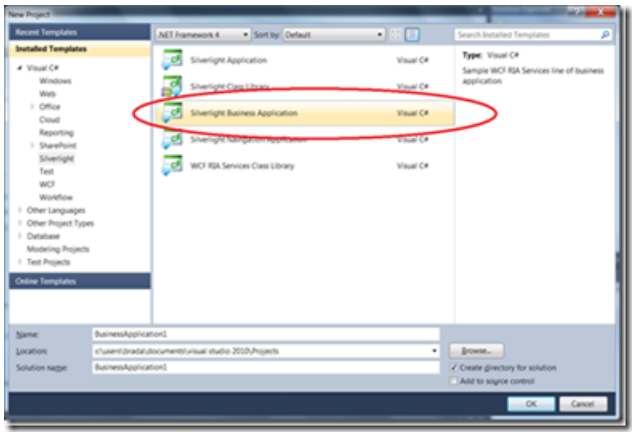
Vous pouvez **télécharger l'application complète**.

J'ai écrit cela avec Silverlight 4 RC, mais je m'attends à ce qu'elle fonctionne avec Silverlight 4 RTM.

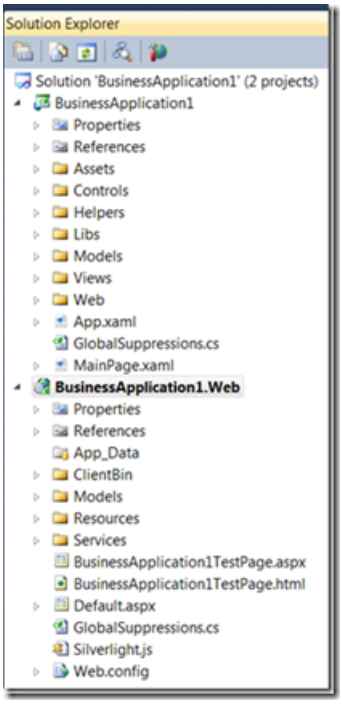
III - Démarrer un nouveau projet avec le modèle d'application métier.

Pour **lancer notre série**, je voulais me concentrer sur notre objectif de vous aider à vous concentrer sur votre métier, et non pas plomber du code. Le premier endroit où vous verrez cela c'est dans les composants préconstruits dans le modèle d'application métier. Il décrit une structure d'application normative, est visuellement agréable et est facilement personnalisable.

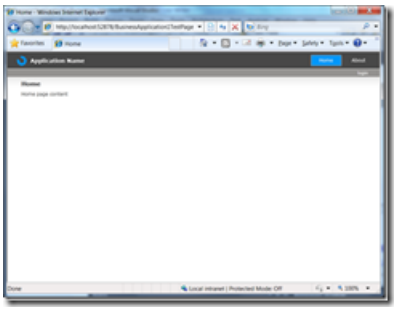
Après avoir correctement installé Silverlight 4 pour les développeurs (qui comprend RIA Services), vous aurez quelques nouveaux projets dans la section Silverlight. Nous allons nous concentrer sur le modèle d'application métier (Business Application Template).



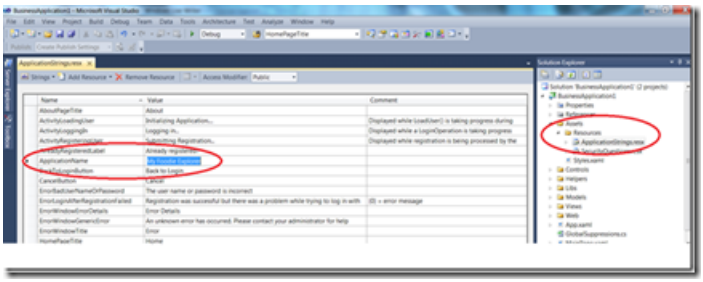
Remarquez que cela crée une solution unique avec deux projets. BusinessApplication1 est la partie cliente de l'application et BusinessApplication2 est la partie serveur. Ce sont des projets connexes qui se connaissent l'un et l'autre.

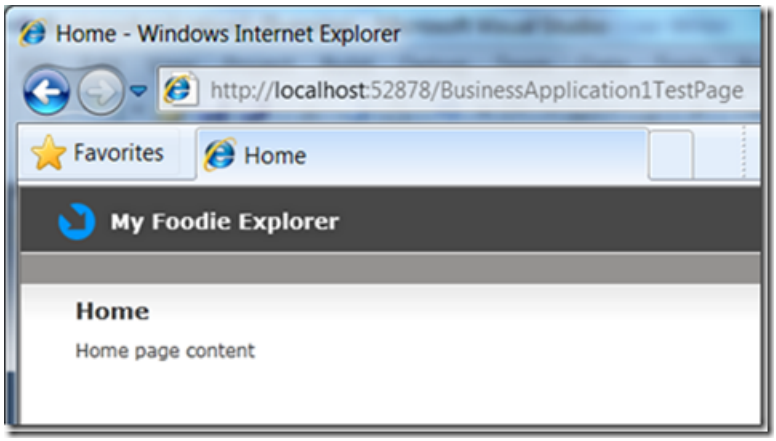


Et par défaut, vous obtenez le cadre pour une application visuellement agréable, et professionnelle.

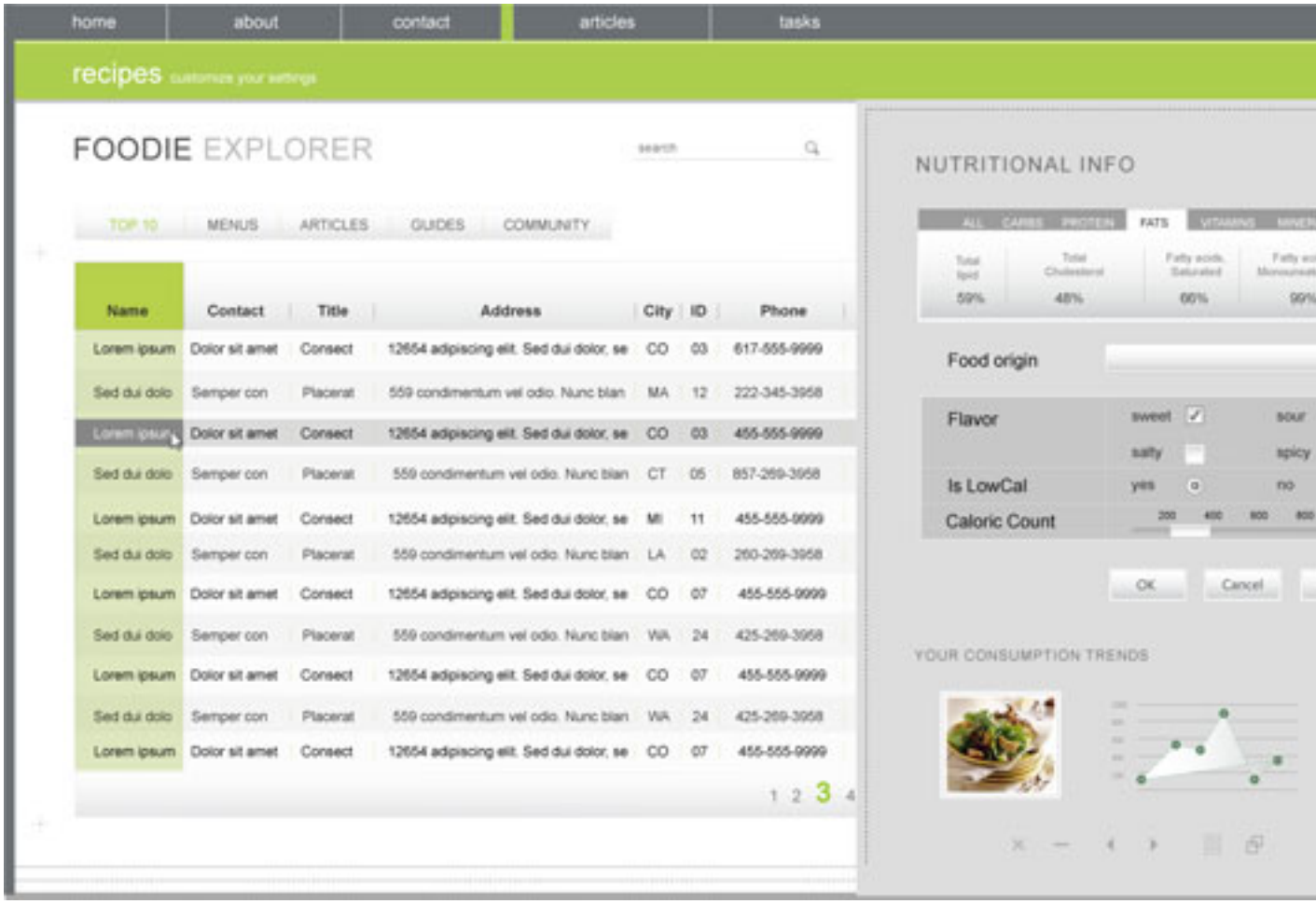


L'application de démarrage est entièrement localisable et personnalisable. Par exemple, je veux changer le nom de l'application de «Application Name» à «My Foodie Explorer». Cela se fait facilement dans le répertoire *Assets \Resources* en éditant le fichier ApplicationStrings.resx.





Jetez un d'oeil **à la procédure pas à pas plus approfondie**, elle est un peu ancienne, mais toujours valide et Tim a récemment posté **APERÇU : Nouveaux thèmes d'application Silverlight** tels que :



Appréciez !

IV - Conclusion

Ceci conclut la première partie de cette série. Dans la deuxième partie nous verrons comment exposer vos données du coté serveur de votre application.

V - Liens

- **Visual Studio 2010**
- **Visual Studio 2010 Express**
- **Silverlight 4 Tools**
- **RIA Services**
- **Télécharger l'application complète**

VI - Traduction

Cet article est la traduction la plus fidèle possible de l'article original de **Brad Abrams**, **Silverlight 4 + RIA Services - Ready for Business: Exposing Data from Entity Framework**

VII - 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 écrit cela avec Silverlight 4 RC, mais je m'attends à ce qu'elle fonctionne avec Silverlight 4 RTM.

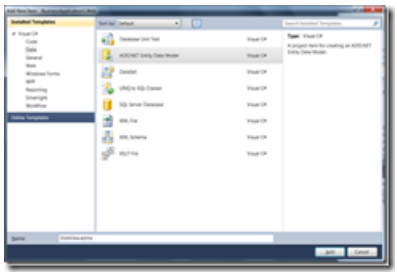
VIII - Exposer les données d'Entity Framework

Pour **continuer notre série**, j'ai voulu ensuite regarder la façon d'exposer vos données du coté serveur de votre application.

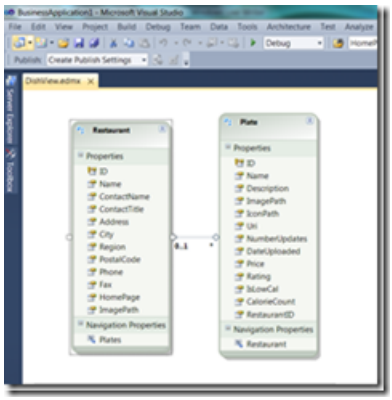
Les données intéressantes dans vos applications métier proviennent d'une grande variété de sources de données. D'une base de données SQL, d'Oracle DB, de SQL Azure, de SharePoint, d'un mainframe et vous avez probablement déjà choisi un modèle de données tel que NHibernate, Linq2Sql, Entity Framework, une procédure stockée, un service. Le but de RIA Services dans ce release est de faciliter le travail avec les données d'une ou de plusieurs sources, d'une manière transparente, à partir d'une application Silverlight. Cette procédure pas à pas utilisera Entity Framework pour accéder à la base de données SQL Express, mais le concept de base s'applique à n'importe quelle source de données.

Ajoutez DishView.mdf à BusinessApplication1.Web\App_Data - Bien entendu, dans un exemple réel vous devriez juste avoir une chaîne de connexion à une base de données existante.

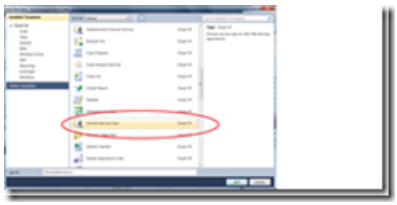
Ensuite, créez un modèle Entity Framework au dessus de celui-ci.



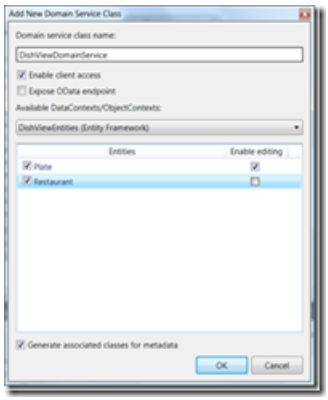
Comme vous pouvez le voir, nous avons un jeu d'entités très simple. Chaque Restaurant peut avoir n'importe quel nombre de Plats.



Ensuite, nous avons besoin d'écrire une certaine logique métier qui contrôle et façonne les données telles qu'elles sont perçues par le client Silverlight. Pour ce faire, nous ajoutons un nouveau DomainService. Un DomainService est simplement un type particulier de service WCF qui rend BEAUCOUP PLUS facile d'interroger, de mettre à jour, de sécuriser et de valider vos données. Mais si vous êtes un expert WCF et connaissez tous les détails pour la configuration d'un service WCF, vous pouvez certainement personnaliser ce service, d'une façon à ce qu'il corresponde exactement à vos besoins. Bien sûr, dans 90% des cas, nous espérons que vous n'aurez pas besoin d'y avoir recours.



Ensuite, il nous est donné la possibilité de pré-remplir le DomainService.



Dans ce cas, nous allons exposer Plat et Restaurant, mais seul Plat pourra être mis à jour. Nous allons également générer une classe de métadonnées pour accrocher les attributs de validation afin que vous puissiez régénérer le modèle EF sans perdre aucune personnalisation. Nous obtenons une classe de démarrage DishViewDomainService. Je l'ai mis à jour aux lignes 8-9 avec un peu de logique métier. Examinons chaque ligne pas à pas...

```
 1:     [EnableClientAccess]
 2:     public class DishViewDomainService : LinqToEntitiesDomainService<DishViewEntities>
 3:     {
 4:
 5:         public IQueryable<Restaurant> GetRestaurants()
 6:         {
 7:             return this.ObjectContext.Restaurants
 8:                 .Where (r=>r.Region != "NC")
 9:                 .OrderBy(r=>r.ID);
10:         }
```

```
11:
```

Ligne 1 : Cet attribut marque le DomainService comme étant accessible via http. Sans lui le service ne peut être appelé que du processus. Ceci est utile dans le scenario ASP.NET Webforms / Données dynamiques.

Ligne 2 : Nous dérivons cette classe de LinqToEntitiesDomainService. C'est une classe utilitaire qui fournit quelques helpers pour travailler avec EF. Le vrai travail est effectué par la classe de base DomainService, celle-ci étant la classe à partir de laquelle vous dérivez pour POCO et d'autres scenarios personnalisés.

Ligne 5 : Remarquez que nous renvoyons un IQueryable de notre type DAL Restaurant. IQueryable est l'interface sur laquelle est construite LINQ. Cela permet des fonctions d'interrogation depuis le client de sorte que vous puissiez faire des choses comme le tri, la pagination, le filtrage depuis le client et l'avoir à composer avec votre logique métier personnalisée et exécuter sur la couche de données. Cela signifie qu'aucune donnée supplémentaire n'est aspirée dans la couche du milieu et le client, mais vous n'avez pas à « crasser » votre logique métier pour faire face à cela.

Lignes 8-9 : Nous avons une logique métier qui élimine tout Restaurant en Caroline du Nord et met les résultats dans un ordre prédéfini. Vous pouvez bien sur imaginer une logique métier plus intéressante.

IX - Conclusion

Ceci conclut la deuxième partie de cette série. Dans la prochaine tranche nous verrons comment consommer les données dans le client Silverlight.

X - 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 également **xxxx** pour sa relecture et ses propositions.