Silverlight 4 + RIA Services - Prêt pour les affaires

Authentification et personnalisation

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: Authentication and Personalization**

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**.

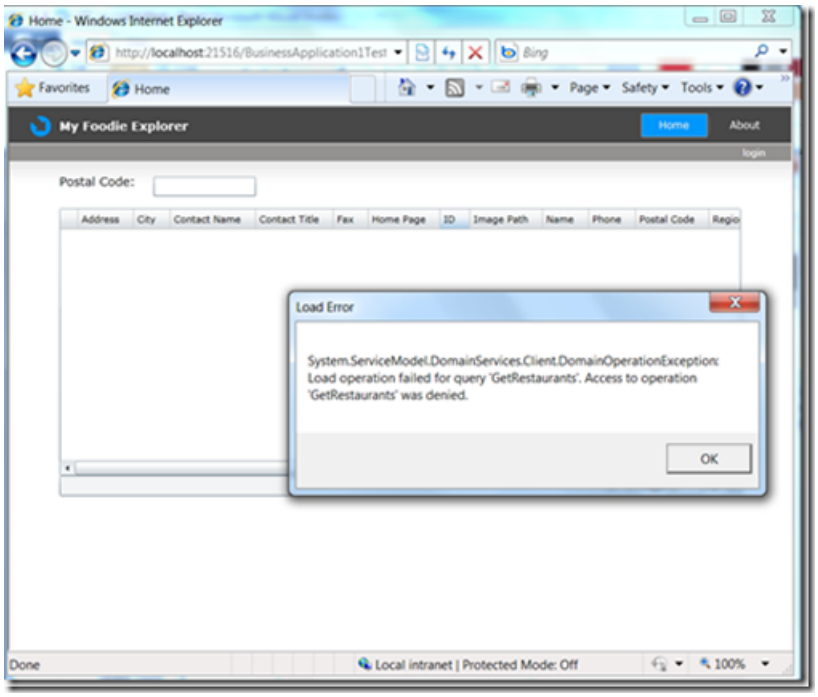
J'ai implémenté cela avec Silverlight 4 RC, mais je m'attends à ce que cela fonctionne avec Silverlight 4 RTM.

Authentification et Personnalisation

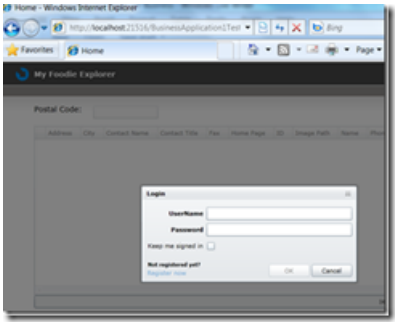
Pour **continuer notre série**, dans les applications métier réelles nos données sont souvent très précieuses et donc, nous devons savoir qui accède a quelles données et nous devons donner accès à certaines données uniquement aux utilisateurs privilégiés. Heureusement cela est très facile à faire avec RIA Services. Par exemple, disons que nous voulons que seuls les utilisateurs authentifiés aient accès à nos données dans cet exemple. C'est aussi facile à réaliser que l'ajout d'un attribut, voir la ligne 2 ci-dessous.

```
1:      [EnableClientAccess]
2:      [RequiresAuthentication]
3:      public class DishViewDomainService : LinqToEntitiesDomainService<DishViewEntities>
4:      {
5:
```

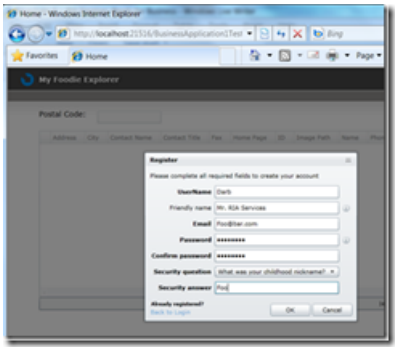
Quand on exécute l'application, nous obtenons maintenant une erreur. Vous pouvez clairement faire un peu mieux d'un point de vue utilisateur. Mais le message est assez clair.



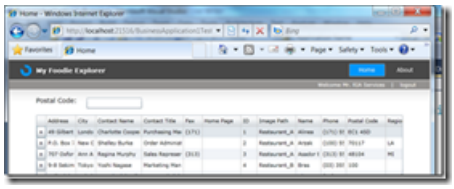
Remarquez qu'il y a une option de login, afin que nous puissions nous connecter.



et même créer un nouvel utilisateur.



Et avec une actualisation, nous obtenons désormais nos données.



Et l'application sait qui je suis sur le client et me donne une façon de me déconnecter.

Maintenant vous pouvez également interagir facilement avec l'utilisateur actuel sur le serveur. Donc, par exemple, renvoyez uniquement les dossiers qu'ils ont édités, ou dans ce cas présent, enregistrez tous les accès :

```
1:          public IQueryable<Restaurant> GetRestaurants()
2:          {
3:              File.AppendAllLines(@"C:\Users\brada\Desktop\log.txt", new string[] {
4:                  String.Format("{0}:{1}", DateTime.Now,
5:                  this.ServiceContext.User.Identity.Name)});
6:              return this.ObjectContext.Restaurants
7:                  .Where (r=>r.Region != "NC")
8:                  .OrderBy(r=>r.ID);
9:          }
10:
```

La ligne 5 est la plus importante. nous sommes en train d'accéder aux utilisateurs actuels sur le serveur. Cela nous donne un journal simple et agréable.

3/7/2010 9:42:57 PM:darb
3/7/2010 9:43:05 PM:darb

Maintenant nous pouvons également personnaliser un peu tout cela. Disons que nous voulons que nos utilisateurs soient capables de nous donner une couleur préférée et nous gardons une trace de cela sur le serveur et le client, de sorte que cela fonctionne en toute transparence depuis n'importe quelle machine.

Tout d'abord, nous devons ajouter *BackgroundColor* à notre emplacement de sauvegarde. Dans ce cas j'utilise le stockage de profil ASP.NET, donc j'ajoute les trucs appropriés à *web.config*

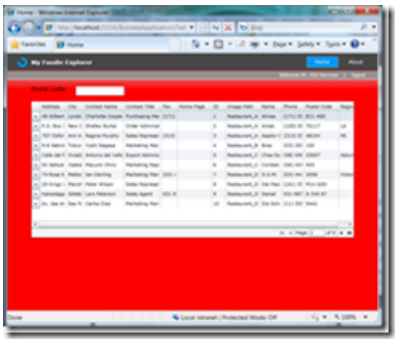
Ensuite je dois y accéder depuis le client Silverlight, donc j'ajoute une propriété à l'instance User dans le Models\User.cs

```
public partial class User : UserBase
    {
        public string FriendlyName { get; set; }
        public string BackgroundColor { get; set; }
    }
```

Finalement, nous devons y accéder sur le client. Dans *main.xaml* ajoutez les lignes 2 et 3 :

```
1:    <Grid x:Name="LayoutRoot" Style="{StaticResource LayoutRootGridStyle}"
2:          Background="{Binding Path=User.BackgroundColor}"
3:          DataContext="{StaticResource WebContext}">
4:
5:
```

Exécutez-le et nous obtenons notre belle couleur de fond par défaut !



Maintenant, ça c'est bien, mais ce serait encore mieux de donner à l'utilisateur la possibilité d'éditer ses paramètres. Donc, dans About.xaml, nous utilisons un modèle très similaire à celui ci-dessus.

```
<Grid x:Name="LayoutRoot"
      DataContext="{StaticResource WebContext}">
```

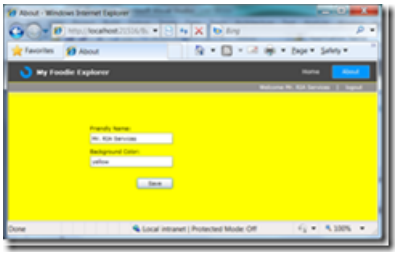
et

```
<sdk:Label Content="Background Color:" />
<TextBox Text="{Binding Path=User.BackgroundColor, Mode=TwoWay}" Height="23" />
```

Ensuite reliez un bouton « Save »

```
        private void button1_Click(object sender, System.Windows.RoutedEventArgs e)
        {
            WebContext.Current.Authentication.SaveUser(false);
        }
```

Et ça fonctionne !



Et ce qui est mieux c'est que si vous l'exécutez à partir d'un autre navigateur, sur une autre machine, une fois que vous vous connectez vous obtenez exactement les mêmes préférences !

Conclusion

Ceci conclut la sixième partie de cette série. Dans la prochaine tranche nous aborderons l'optimisation pour les moteurs de recherche (SEO).

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 **tomlev** pour sa relecture technique et ses propositions.
Je remercie également **xyz** pour sa relecture orthographique et ses propositions.