Exemple d'application d'entreprises pour Silverlight 3 et .NET RIA Services. - Partie 3: L'authentification

Cet exercice va nous faire découvrir l'accès aux données avec Silverlight.

Cet exercice requière (tout est gratuit et le restera) :
• **VS2008SP1 (qui comprend Sql Express 2008)**
• **Silverlight 3.0**
• **.NET RIA Service**

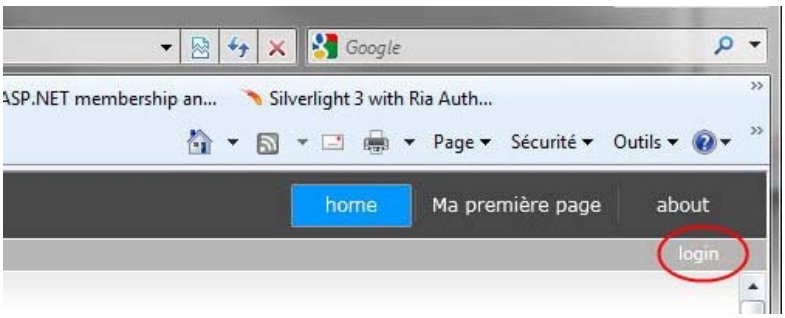
Aujourd'hui, nous allons parler de l'authentification.

NdT : pour cette session, il ne s'agit que d'une présentation basique des possibilités offerte par .NET RIA Services et Silverlight. Une session plus approfondie fera l'objet d'un article futur.

Les applications professionnelles ont souvent accès à des données sensibles. Il est important que vous puissiez auditer, restreindre et contrôler l'accès à vos données. Voyons comment .NET RIA Services et Silverlight opèrent pour réaliser cela.

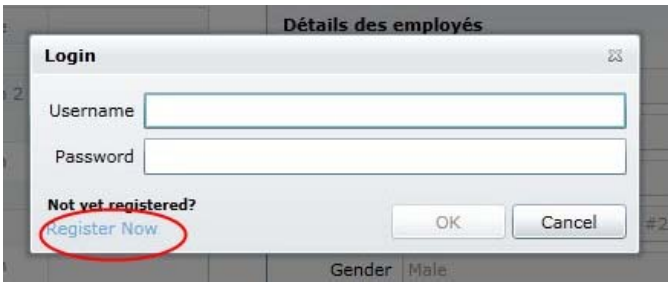
Utiliser le modèle *Silverlight Business Application* est super facile… Par défaut, il est connecté au système d'authentification d'ASP.NET qui offre un système de gestion utilisateur personnalisable.

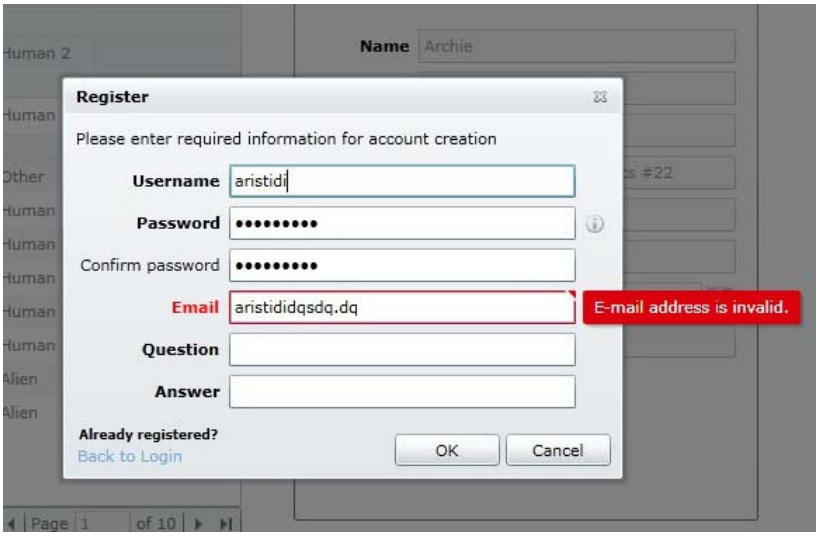
Je vais faire la démonstration en utilisant l'authentification par formulaire, vous pouvez bien sûr utiliser l'authentification Windows en faisant une légère modification du modèle.



Pour cette démo, je vais faire la démonstration de la création d'un nouvel utilisateur, mais si vous disposer déjà d'une base utilisateur, vous pouvez l'utiliser.

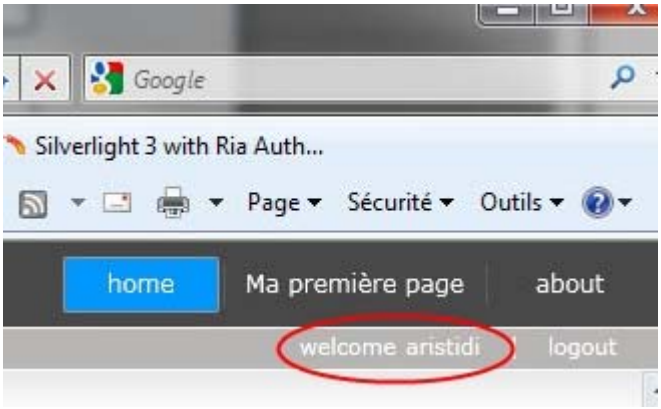
NdT : pour cette session, pensez à lancer l'application avec CTRL+F5.





Notez que nous avons des règles de validations fonctionnelles.

Maintenant, l'application sait qui vous êtes…



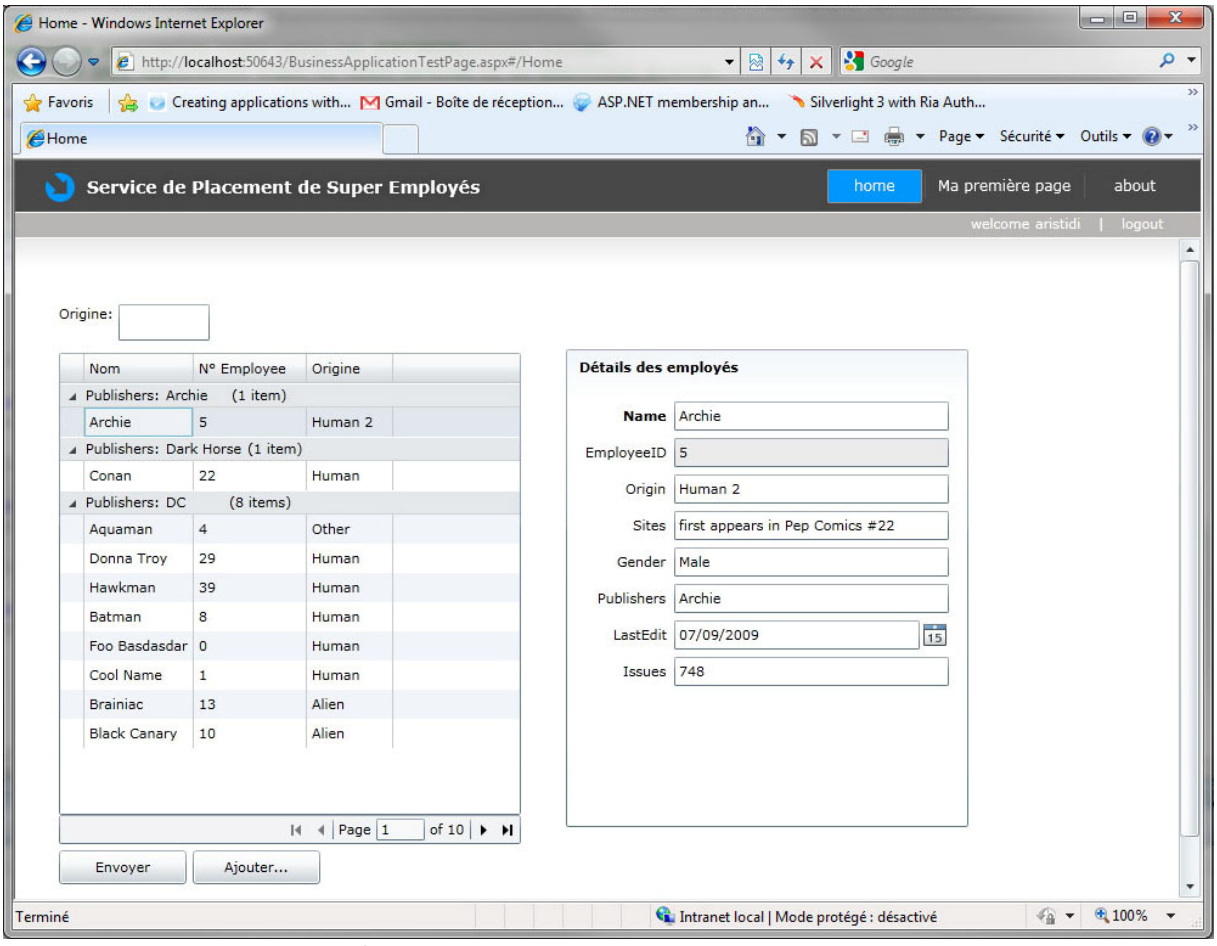
Notez que tout ce qui concerne l'expérience utilisateur sur le client est complètement personnalisable étant donné que tout le code source est inclus dans le projet. Mais le modèle par défaut est pas mal pour de nombreuses applications.

Maintenant que nous sommes connectés, faisons quelque chose de ces données utilisateur. Par exemple, faisons en sorte que seuls les utilisateurs enregistrés puissent accéder aux données des supers employés. Editons la classe *SuperEmployeeDomainService (SuperEmployeeDomainService.cs)* sur le serveur afin d'ajouter l'attribut *RequiresAuthentication*. Il y a d'autres attributs comme les rôles et la possibilité de les gérer dans votre code si vous le voulez.

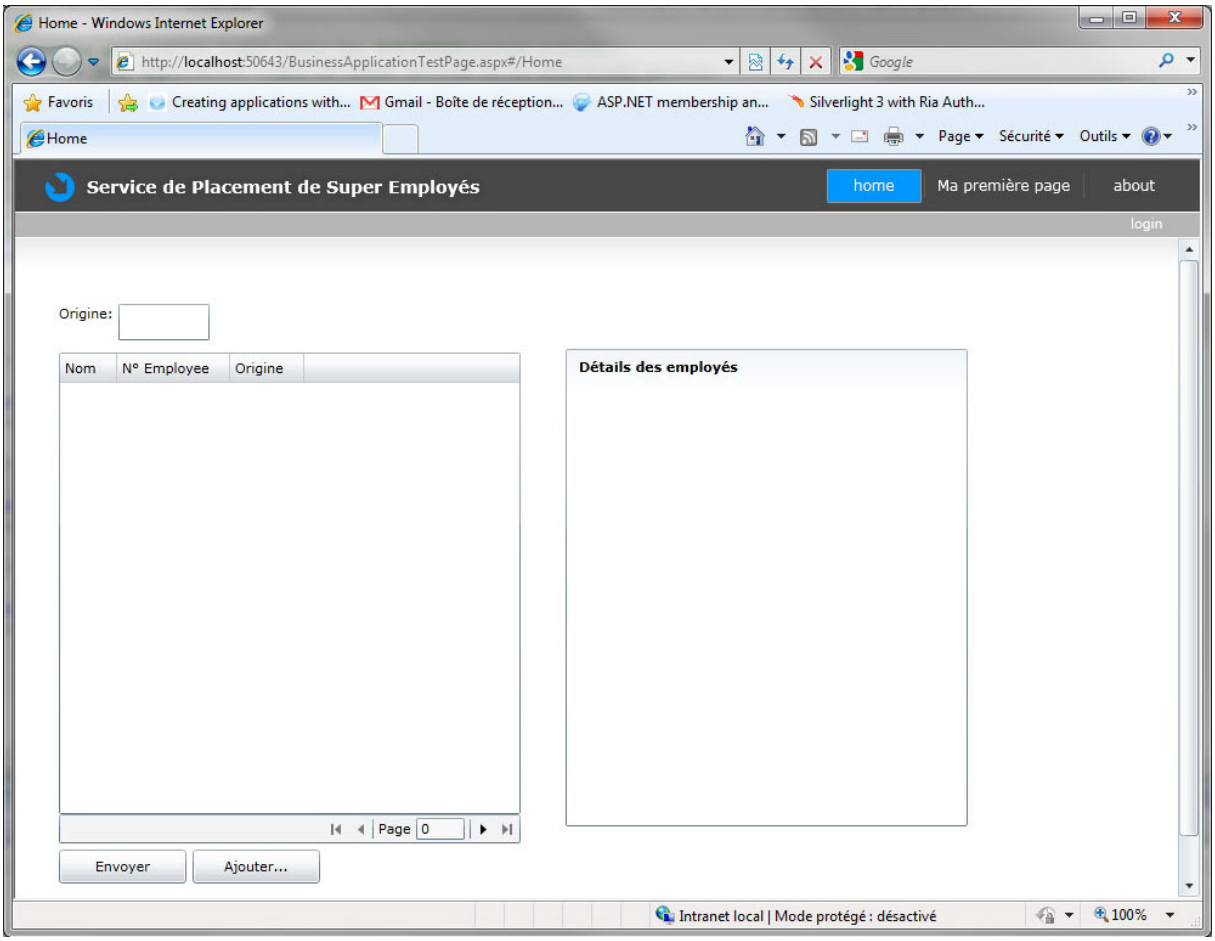
```
public class SuperEmployeeDomainService : LinqToEntitiesDomainService<NORTHWNDEntities>
{

    // TODO: Consider
    // 1. Adding parameters to this method and constraining returned results, and/or
    // 2. Adding query methods taking different parameters.
    [RequiresAuthentication]
    public IQueryable<SuperEmployees> GetSuperEmployees()
    {
        return this.Context.SuperEmployees
            .Where(emp => emp.Issues > 100)
            .OrderBy(emp => emp.EmployeeID);
    }
```

Maintenant, si nous lançons l'application et que nous ne nous connectons pas, nous n'aurons aucune donnée. Notez que cette validation est faite coté client pour une expérience utilisateur riche et une fois encore sur le serveur pour s'assurer de la sécurité.



Lorsque nous somme connectés…



… et lorsque nous le sommes pas.