Exemple d'applications métier avec Silverlight 3 RTM et .NET RIA Services mise à jour de juillet

Partie 12 : DataSet

par Brad Adams Jérôme Lambert (Traduction) (Espace perso) (Blog)

Date de publication : 08 janvier 2011

Dernière mise à jour :

Cet article est une traduction d'un des articles de la série "**Business Apps Example for Silverlight 3 RTM and .NET RIA Services July Update**" de Brad Adams.

I - DataSet.....3

Copyright 2011 - Brad Abrams. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts. Droits de diffusion permanents accordés à developpez LLC.

I - DataSet

Lors de ma conférence "**building business applications with Silverlight 3**" lors du **Mix09**, beaucoup de visiteurs m'ont dit qu'ils adoraient **Entity Framework** et **LinqToSql**, mais ils ne sont malheureusement pas toujours capables de les utiliser dans leurs projets. En fait, le nombre de personnes qui utilisent ADO.NET DataSet, DataReader, etc est très élevé. J'ai donc voulu montré l'utilisation du modèle de données standard d'ADO.NET pour l'accès aux données en prenant ma démo du Mix et la modifiant.

Ceci vous permettra d'utiliser des DataSet avec Silverlight ET prendre l'avantage des excellentes fonctionnalités qu'offre RIA Services concernant la validation de données, la pagination, etc.

Pour cet article, vous pouvez **la vidéo complète de la session**.

Cette démo nécessite (tout est 100% gratuit et pour toujours) :

1. **VS2008 SP1**
2. **Silverlight 3 RTM**
3. **.NET RIA Services July '09 Preview** <--- N'est en réalité pas nécessaire pour cette démo ! Mais c'est néanmoins bien de le savoir ;-)

Téléchargez ensuite l'ensemble des fichiers des démos (**lien sur le site original** - **lien sur developpez.com**).

Tout d'abord, nous pouvons supprimer le modèle Entity Framework de notre projet... nous allons utiliser des DataSet comme notre modèle d'accès aux données dans cette démo. Notez que ce modèle a probablement plus de send si vous avez déjà construit beaucoup d'infrastructures autour des DataSet... Si ce n'est pas le cas, alors l'utilisation des DataReader/Writer devrait être un bon choix.

Tout d'abord, nous avons besoin de créer un type que nous retournons au client.

```
public class SuperEmployee
{

    [ReadOnly(true)]
    [Key]
    public int EmployeeID { get; set; }


    [RegularExpression("^(?:m|M|male|Male|f|F|female|Female)$",
        ErrorMessage = "Gender must be 'Male' or 'Female'")]
    public string Gender { get; set; }

    [Range(0, 10000,
        ErrorMessage = "Issues must be between 0 and 1000")]
    public Nullable<int> Issues { get; set; }

    public Nullable<DateTime> LastEdit { get; set; }

    [Required]
    [StringLength(100)]
    public string Name { get; set; }

    public string Origin { get; set; }

    public string Publishers { get; set; }

    public string Sites { get; set; }
}
```

Notez ici que nous pouvons placer les métadonnées de validation directement sur le type que nous retournons. Maintenant, nous avons juste besoin de remplir ce type à partir de la base de données..

Copyright 2011 - Brad Abrams. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts. Droits de diffusion permanents accordés à developpez LLC.

Commençons par définir un DomainService

```
[EnableClientAccess()]
 public class SuperEmployeeDomainService : DomainService
 {
     DataSet Context = new DataSet();

     const int PageSize = 20;
```

Notez ici que nous sommes dirigés directement à partir de DomainService... Il n'y a aucune besoin d'utiliser l'Entity Framework ou LinqToSqlDomainService.. Nous configurons ensuite le contexte en tant que DataSet.. Nous alimenterons ce DataSet dans les méthodes du DomainService. Enfin, nous définisons une taille de page pour nos données.. Cela nous donnera un bloc standard pour accéder à la base de données.

Ensuite, j'ai écrit quelques lignes de codes permettant d'alimenter le DataSet... Je suppose qu'il serait facile de changer ceci pour fonctionner avec n'importe quel autre modèle que vous utilisez avec vos DataSets.

```
void FillSuperEmployees(DataSet ds, int page, int employeeID)
{
    var conn = new SqlConnection();
    conn.ConnectionString = ConfigurationManager.ConnectionStrings["MainConnStr"].ConnectionString;

    SqlDataAdapter da;
    if (employeeID == -1)
    {
        da = new SqlDataAdapter(
            "SELECT * " +
            "FROM SuperEmployees",
            conn);
    }
    else
    {
         da = new SqlDataAdapter(
            "SELECT * " +
            "FROM SuperEmployees " +
            "WHERE EmployeeID=" + employeeID,
            conn);
    }
    if (page == -1) da.Fill(ds, "SuperEmployees");
    else            da.Fill(ds, page * PageSize, PageSize, "SuperEmployees");
}
```

Nous écrivons ensuite une méthode qui va retourner une requête..

```
public IQueryable<SuperEmployee> GetSuperEmployees(int pageNumber)
{
 Context = new DataSet();
 FillSuperEmployees(Context, pageNumber,-1);
 DataTable superEmployees = Context.Tables["SuperEmployees"];

 var query =  from  row in
   superEmployees.AsEnumerable()
  select new SuperEmployee
  {
   EmployeeID = row.Field<int>("EmployeeID"),
   Name = row.Field<string>("Name"),
   Gender = row.Field<string>("Gender"),
   Issues = row.Field<int?>("Issues"),
   LastEdit = row.Field<DateTime>("LastEdit"),
   Origin = row.Field<string>("Origin"),
   Publishers = row.Field<string>("Publishers"),
   Sites = row.Field<string>("Sites"),
  };
 return query.AsQueryable();
}
```


Copyright 2011 - Brad Abrams. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts. Droits de diffusion permanents accordés à developpez LLC.

A la ligne 4, nous remplissons le DataSet. Ensuite des lignes 8 à 20, nous utilisons un peu **LinqToDataSet** pour facilité la création d'une projection de notre DataSet. Si vous ne préférez pas utiliser Linq ici, il n'y a aucun problème car vous pouvez simplement une copie de la méthode qui récupère des données du DataSet afin d'initialiser notre type SuperEmployee. N'importe quelle collection peut être retournée comme une IQueryable. Notez que nous prenons en compte le numéro de page ici... Nous allons suivre le même modèle de pagination explicite que j'ai introduit dans **l'exemple WCF**.

Jettons ensuite un oeil sur la mise à jour... Cette méthode est appelée lorsqu'il y a un changement d'un des champs dans notre instance de SuperEmployee...

```
public void UpdateSuperEmployee(SuperEmployee currentSuperEmployee)
{

 GetSuperEmployee(currentSuperEmployee.EmployeeID);

 DataRow updateRow = null;
 foreach (DataRow row in Context.Tables["SuperEmployees"].Rows) {
  if (row.Field<int>("EmployeeID") == currentSuperEmployee.EmployeeID) {
  updateRow = row;
  }
 }

 var orgEmp = this.ChangeSet.GetOriginal(currentSuperEmployee);

 if (orgEmp.Gender != currentSuperEmployee.Gender)
  updateRow.SetField("Gender", currentSuperEmployee.Gender);
 if (orgEmp.Issues != currentSuperEmployee.Issues)
  updateRow.SetField("Issues", currentSuperEmployee.Issues);
 if (orgEmp.LastEdit != currentSuperEmployee.LastEdit)
  updateRow.SetField("LastEdit", currentSuperEmployee.LastEdit);
 if (orgEmp.Name != currentSuperEmployee.Name)
  updateRow.SetField("Name", currentSuperEmployee.Name);
 if (orgEmp.Origin != currentSuperEmployee.Origin)
  updateRow.SetField("Origin", currentSuperEmployee.Origin);
 if (orgEmp.Publishers != currentSuperEmployee.Publishers)
  updateRow.SetField("Publishers", currentSuperEmployee.Publishers);
 if (orgEmp.Sites != currentSuperEmployee.Sites)
  updateRow.SetField("Sites", currentSuperEmployee.Sites);

}
```

Tout d'abord, nous avons besoin de récupérer la DataRow à mettre à jour. A la ligne 4, nous la chargeons à partir de la base de données, ensuite aux lignes 6 à 11, nous la retrouvrons dans le DataSet courant (rappelez-vous, nous faisons le traitement par lots, il pourrait donc y avoir de nombreuses mises à jour déjà faites dans le DataSet).

Notez le comportement général utilisé et qui est de comparer les valeurs orginales visible par le client (à partir de la ligne 13) aux valeurs renvoyées par le client. Ceci assure que nous changerons uniquement les champs qui ont été vraiment mis à jour. Sinon, nous pourrions écraser des changements provenant d'autres clients. Ceci est beaucoup plus comme le code que nous avons fait dans l'**exemple DTO**.

Au final, dans Submit, nous avons simplement besoin de valider ces changements en base de données.

```
public override void Submit(ChangeSet changeSet)
{
 base.Submit(changeSet);
 var conn = new SqlConnection();
 conn.ConnectionString = ConfigurationManager.ConnectionStrings["MainConnStr"].ConnectionString;


 SqlDataAdapter da = new SqlDataAdapter(
  "SELECT * " +
  "FROM SuperEmployees ",
  conn);
 SqlCommandBuilder com = new SqlCommandBuilder(da);
 da.Update(Context, "SuperEmployees");
```


Copyright 2011 - Brad Abrams. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts. Droits de diffusion permanents accordés à developpez LLC.

```
}
```

En regardant cette réimplémentation de Submit, nous avons donne quelques bons aperçus sur la façon dont RIA Services fonctionne vraiment. Submit est appelée lorsqu'une première requête arrive en provenance du client. Cette requête peut contenir de nombreux ajouts, suppressions ou mises à jour dans l'ensemble de modifications. L'appel de base.Submit() prend l'ensemble de modifications et appel la méthode update/add/delete appropriée pour chaque changement. Ces changements devraient laisser le DataSet populé avec les changements que nous avons besoin de valider en base de données. La ligne 13 veille à ça. Notez au passage que c'est un très bon endroit pour placer une point d'arrêt lorsque vous êtes en train de débogguer votre DomainService.

Les seuls réels changements sur le client sont d'adapter le modèle de panignation explicite que nous avons vu dans **l'exemple WCF**... Ce qui est excellent... Ce qui signifie que nous pouvons passer de ce modèle DataSet à Entity Framework avec très peu de changements au niveau du client.

Cet exemple vous a présenté comment utiliser du code existant fonctionnant avec des DataSet et l'exposer à des clients Silverlight au travers de .NET RIA Services.

Copyright 2011 - Brad Abrams. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts. Droits de diffusion permanents accordés à developpez LLC.