



Dotnet France
Technologies Sharepoint, SQL Server & .NET

Association Dotnet France

LINQ to Entities

Version 1.0

Sommaire

1	Introduction à LINQ to Entities.....	3
2	Créer un projet LINQ to Entities avec Visual Studio	3
2.1	Utiliser le Framework 3.5	3
2.2	Créer et référencer votre source de données.....	3
3	Le modèle de données d'entité.....	5
4	Exemple	6
4.1	Pré-requis	6
4.1.1	La table « Personnel »	6
4.1.2	Le modèle de données d'entité.....	7
4.2	Exemple de code	8
5	Conclusion	10

1 Introduction à LINQ to Entities

LINQ to Entities est à peu de choses près similaires à LINQ to SQL. En effet, la seule différence réside dans le fait que LINQ to Entities n'utilise pas la base de données physique, mais plutôt un modèle conceptuel de cette base de données. Cela se nomme aussi « Modèle de données d'entité » (EDM : Entity Data Model). LINQ to Entities a été imaginé afin de permettre aux applications d'interagir avec des données représentées sous forme relationnelle. Ceci est dû au fait que, de manière générale, les schémas des bases de données ne sont pas toujours idéaux pour faire des manipulations de ces informations. Grâce à LINQ to Entities, les applications vont pouvoir fonctionner avec des données sous forme d'entité ou d'objets dans l'environnement .NET, ce qui rend leur utilisation plus aisée.

Pour mieux comprendre ce cours, il vous est conseillé de voir les chapitres « Introduction au LINQ », « LINQ to Objects » ainsi que « LINQ to SQL ».

2 Créer un projet LINQ to Entities avec Visual Studio

Pour programmer avec LINQ to Entities, il y a quelques manipulations supplémentaires à faire par rapport à un projet courant. Vous pouvez trouver ci-dessous la marche à suivre afin de paramétrer ledit projet.

2.1 Utiliser le Framework 3.5

- 1) Tout d'abord, créez votre projet C# ou VB.NET en fonction de vos besoins. Vous pouvez également utiliser un projet déjà existant.
- 2) Faites un clic droit sur votre projet maintenant ouvert puis choisissez *Propriétés*.
 - a. Pour un projet C#, dans la page de propriétés *Application* et dans l'onglet *.NET*, sélectionnez la valeur *.NET Framework 3.5* pour la liste déroulante *Framework cible*.
 - b. Pour un projet VB.NET, dans la page de propriétés *Compiler* cliquez sur le bouton *Options avancées de compilation...* . Dans la fenêtre qui s'ouvre, sélectionnez la valeur *.NET Framework 3.5* pour la liste déroulante *Framework cible (toutes les configurations)*.
- 3) Ajoutez les espaces de nom requis pour utiliser LINQ to Entities :
 - a. Si votre projet est en C#, vérifiez la présence de l'espace de nom *System.Linq* et ajoutez *System.Data.Objects* à votre fichier source ou votre projet grâce au mot-clé *using*.
 - b. Si votre projet est en VB.NET ajoutez les espaces de nom *System.Linq* et *System.Data.Objects* à votre fichier source ou votre projet en utilisant le mot-clé *Imports*.

2.2 Créer et référencer votre source de données

Dans nos exemples, nous utilisons *SQL Server* qui est inclus lors de l'installation de Visual Studio. Vous pouvez trouver ci-dessous la démarche pour créer et référencer votre source d'informations depuis une base de données existante.

- 1) Pour commencer, réalisez un clic droit sur votre projet puis choisissez *Ajouter* puis *Nouvel élément...* .
- 2) Dans la fenêtre qui s'affiche, sélectionnez le modèle *ADO.NET Entity Data Model* et nommez ce fichier à votre convenance avant de valider en cliquant sur *Ajouter*.

Remarque : Pour nos exemples, nous avons choisit de nommer ce fichier « *LINQtoEntities.edmx* ».

- 3) Une nouvelle fenêtre s'affiche vous proposant de *Générer à partir de la base de données* votre source d'informations pour le projet. Gardez ce choix et cliquez sur *Suivant*.
- 4) A l'écran *Choisir* votre connexion de données, sélectionnez votre connexion déjà existante ou créez en une à partir du bouton *Nouvelle connexion*.
- 5) Avant de continuer, nous vous conseillons de nommer l'enregistrement des paramètres de connexion d'entité d'App.Config avec le même nom que votre fichier « .edmx » suivi du mot « Entities » pour des raisons de lisibilité.

Remarque : Nous avons nommé ce fichier « LINQtoEntitiesEntities ».

- 6) Après avoir validé vos choix, une étape *Choisir vos objets de base de données* apparaît. Développez l'arborescence *Tables* afin de sélectionner les tables qui seront utiles pour le projet.
- 7) Nous vous conseillons ensuite de nommer l'espace de nom du modèle avec le nom de votre fichier « .edmx » suivi du mot « Model ». Fermez cette fenêtre en cliquant sur *Terminer*.

Remarque : Nous avons sélectionné une table « Personnel » contenant des colonnes « PersonnelID », « Nom », « Prenom » et « Date_de_naissance » puis nommé le fichier de sauvegarde « LINQtoEntitiesModel ».

- 8) Ajoutez vos paramètres de connexion nouvellement créés à votre fichier source :
 - a. Pour un projet en C#, utilisez l'instruction *using* suivit de l'instanciation de votre conteneur d'entités (nom avec le suffixe « Entities ») dans la méthode principale *Main()*.
 - b. Pour un projet en VB.NET, instanciez votre conteneur d'entités (nom avec le suffixe « Entities ») dans la méthode principale *Main()*.

```
//C#
using (LINQtoEntitiesEntities MyEntities = new LINQtoEntitiesEntities())
{
    //Code utilisant la base de données
}

'VB.NET
Dim MyEntities As LINQtoEntitiesEntities = New LINQtoEntitiesEntities()
'Code utilisant la base de données
```

Votre projet devrait maintenant être bien configuré pour utiliser LINQ to Entities.

3 Le modèle de données d'entité

Le modèle de données d'entité est représenté par le fichier « .edmx » dans votre projet. Celui-ci représente de manière graphique toutes les tables, vues, procédures stockées et les éventuels liens existant entre les tables (association, héritage, etc.) aussi appelé mappage. Dans notre exemple, nous n'avons qu'une seule table nommée « Personnel ».

Voici l'affichage de notre fichier « LINQtoEntities.edmx » :



L'affichage du modèle de données d'entité est le même pour un projet en C# ou un projet en VB.NET.

Grâce à cette vue, il est possible de renommer les champs (propriétés scalaires) qui sont liées à notre table « Personnel » dans la base de données. Dans ce cas, il sera possible d'utiliser ce nouveau nom de champ dans une requête à la place de celui inscrit de la base de données pour le même résultat.

Remarque : Renommer une colonne dans votre fichier « .edmx » ne modifie pas le nom de la colonne dans la base de données.

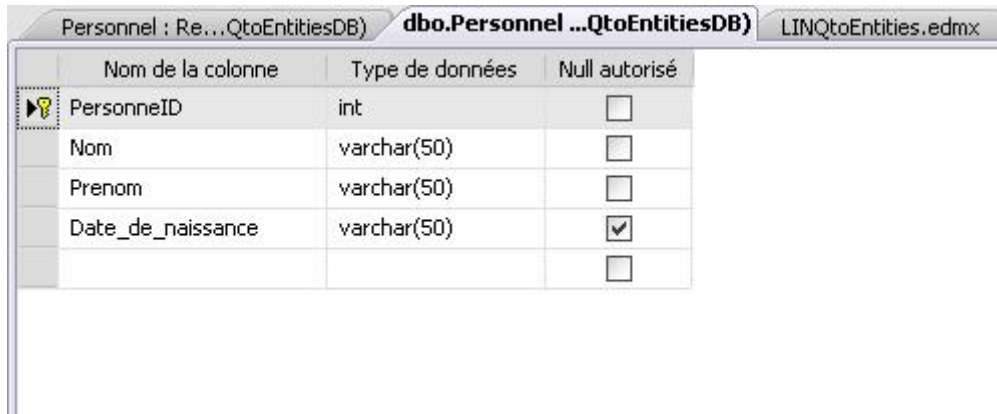
Il est également possible via cette vue d'éditer, de créer ou de supprimer des mappages entre des tables.

4 Exemple

4.1 Pré-requis

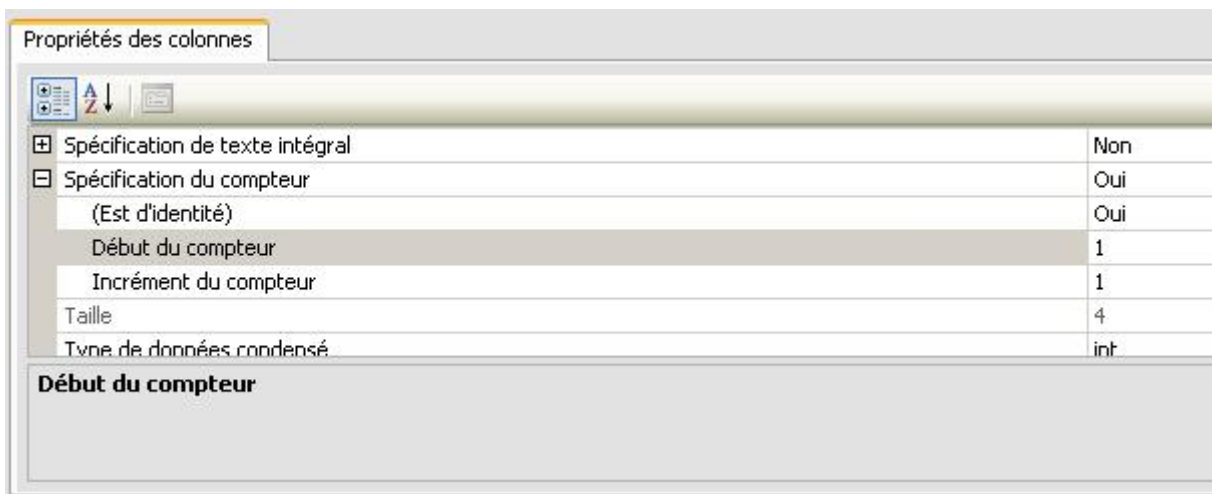
4.1.1 La table « Personnel »

Par soucis de compréhension, voici ci-dessous les détails de la table « Personnel » que nous utilisons dans l'exemple :



Nom de la colonne	Type de données	Null autorisé
PersonneID	int	<input type="checkbox"/>
Nom	varchar(50)	<input type="checkbox"/>
Prenom	varchar(50)	<input type="checkbox"/>
Date_de_naissance	varchar(50)	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

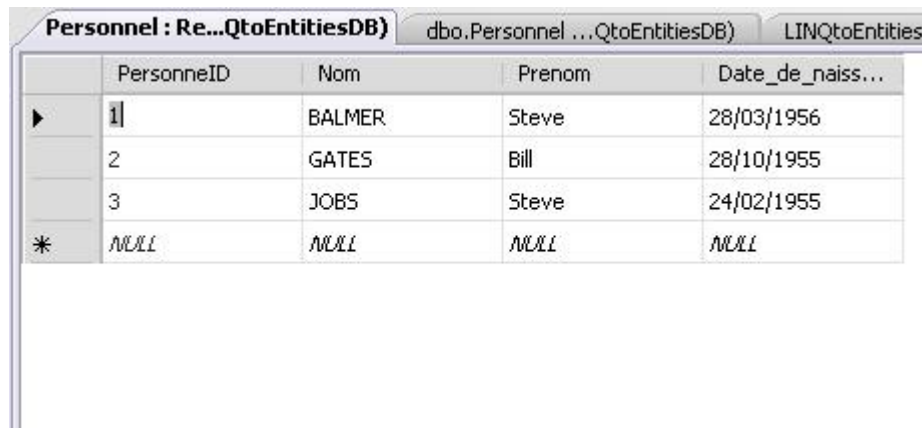
La colonne « PersonneID » est quelque peu particulière car en plus d'être la clé primaire, cette colonne procède à une auto-incrémentation. Pour paramétrer cette fonction, sélectionner la colonne puis dans le cadre inférieur *Propriétés des colonnes*, développer *Spécification du compteur* et enfin changer la valeur « (Est d'identité) » à « Oui ». Garder la valeur par défaut des champs « Début du compteur » et « Incrément du compteur » :



Propriété	Valeur
Spécification de texte intégral	Non
Spécification du compteur	Oui
(Est d'identité)	Oui
Début du compteur	1
Incrément du compteur	1
Taille	4
Type de données condensé	int

Début du compteur

Pour plus de simplicité, le contenu de la table a été inséré manuellement à partir de l’affichage de la table. Voici le contenu de la table :



	PersonneID	Nom	Prenom	Date_de_naiss...
▶	1	BALMER	Steve	28/03/1956
	2	GATES	Bill	28/10/1955
	3	JOBS	Steve	24/02/1955
*	NULL	NULL	NULL	NULL

4.1.2 Le modèle de données d’entité

Dans la partie « Le modèle de données d’entités », il a été spécifié la possibilité de renommer une colonne dans notre fichier « .edmx ». Dans notre cas, nous appliquons cela au champ « Date_de_naissance » qui est trop long à écrire. Son nouveau nom au sein de notre projet est « Naissance ».



4.2 Exemple de code

Les requêtes LINQ to Entities s'écrivent de la même manière qu'avec les autres LINQ.

L'extrait de code suivant permet de récupérer l'ID, le nom, le prénom et la date de naissance d'une personne grâce à son nom :

```
//C#
using (LINQtoEntitiesEntities MyEntities = new LINQtoEntitiesEntities())
{
    ObjectQuery<Personnel> Personnel = MyEntities.Personnel;

    //Nous utilisons la propriété Naissance à la place de
    //Date_de_naissance
    var requete = from p in Personnel
                  where p.Nom == "BALMER"
                  select new {p.PersonneID, p.Nom, p.Prenom,
p.Naissance};

    foreach (var ligne in requete)
        Console.WriteLine(ligne.PersonneID + " " + ligne.Nom + " " +
ligne.Prenom + " " + ligne.Naissance);
}

Console.Read();

'VB.NET
Dim MyEntities As LINQtoEntitiesEntities = New LINQtoEntitiesEntities()

Dim Personnel As ObjectQuery(Of Personnel) = MyEntities.Personnel

'Nous utilisons la propriété Naissance à la place de Date_de_naissance
Dim requete = From p In Personnel
              Where p.Nom = "GATES"
              Select New With {p.PersonneID, p.Nom, p.Prenom,
p.Naissance}

For Each ligne In requete
    Console.WriteLine(ligne.PersonneID.ToString() + " " + ligne.Nom + " "
+ ligne.Prenom + " " + ligne.Naissance)
Next

Console.Read()
```

Dans un premier temps, il est utile de créer un *ObjectQuery* qui représentera la table « Personnel » de notre conteneur d'entités. Il aurait été possible d'omettre cette instantiation. Auquel cas, lors de notre requête, il aurait fallu sélectionner *MyEntities.Personnel* comme source à la place de l'objet « Personnel » créé.

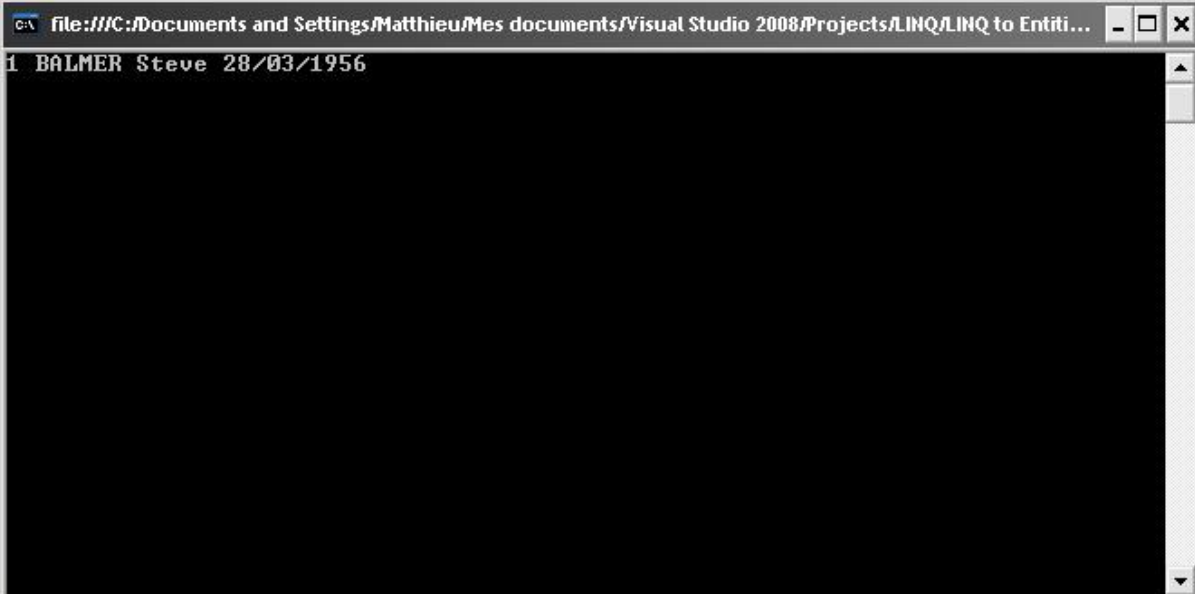
La requête sélectionne les personnes incluses dans notre entité « Personnel » (correspondant à notre table) dont le nom correspond à « BALMER » dans le cas du C# et « GATES » dans celui du VB.NET. Les informations de cette personne (id, nom, prénom et date de naissance) sont incluses dans un nouvel objet anonyme.

Enfin, nous affichons les éventuels objets sélectionnés par la requête grâce à une boucle *foreach*. Normalement, un seul objet est retourné ce qui rend la boucle inutile, mais nous conseillons de l'utiliser dans les projets pour les cas où plusieurs objets seraient retournés.

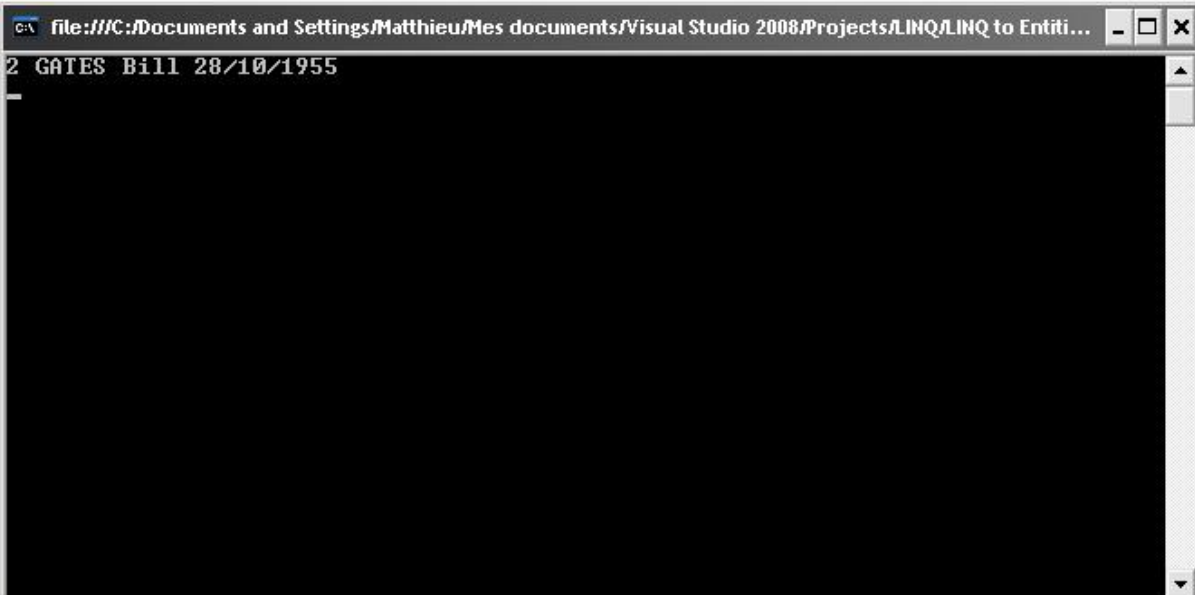
Remarque : Dans le langage VB.NET, la méthode ToString() n'est pas automatiquement appelée lors de l'affichage de la valeur « PersonneID ».

Remarque : L'utilisation du Console.Read() permet simplement de faire une pause dans le programme et donc de voir les résultats. Sans celui-ci, l'application se serait fermée immédiatement après l'affichage du résultat.

Voici ci-dessous les affichages des résultats :



```
c:\ file:///C:/Documents and Settings/Matthieu/Mes documents/Visual Studio 2008/Projects/LINQ/LINQ to Entiti... - □ X
1 BALMER Steve 28/03/1956
```



```
c:\ file:///C:/Documents and Settings/Matthieu/Mes documents/Visual Studio 2008/Projects/LINQ/LINQ to Entiti... - □ X
2 GATES Bill 28/10/1955
```

5 Conclusion

Dans ce chapitre nous avons abordé les spécificités de l'utilisation de LINQ to Entities. L'atout majeur de LINQ étant de garder une même requête adaptable à plusieurs sources de données, il n'a pas été traité les manières de créer, modifier ou supprimer des valeurs d'une table. Si vous souhaitez en apprendre plus sur la manipulation de données avec LINQ, les chapitres « Introduction au LINQ », « LINQ to Objects » et « LINQ to SQL » sont à disposition sur notre site Internet.

Plus d'informations sur le MSDN : <http://msdn.microsoft.com/fr-fr/library/bb386992.aspx>