

LINQ, Introduction

LINQ (Language Integrated Query) est une innovation dans Visual Studio 2008 et la version 3.5 du .NET Framework qui permet de rapprocher le monde des objets et le monde des données.

Avec LINQ, toute requête prend la forme d'une construction de langage de premier ordre en C# et Visual Basic. Vous écrivez des requêtes sur des collections d'objets fortement typées en utilisant des mots clés du langage et des opérateurs familiers

LINQ simplifie cette situation en proposant un modèle cohérent qui permet d'utiliser des données de types de sources et de formats divers. Dans une requête LINQ, vous travaillez toujours avec des objets. Vous utilisez les mêmes modèles de codage de base pour interroger et transformer des données en documents XML, en bases de données SQL, en groupes de données ADO.NET, en collections .NET et en tout autre format pour lesquels un fournisseur LINQ est disponible.

Toutes les opérations de requête LINQ comportent trois actions distinctes :

1. Se connecter à la base de données avec le DataContext

Le DataContext est le conduit principal par lequel vous vous connectez à une base de données, récupérez des objets de celle-ci et soumettez des modifications. Le DataContext fonctionne comme un SqlConnection ADO.NET. En fait, le DataContext est initialisé avec une connexion ou une chaîne de connexion que vous fournissez.

Exemple, pour un serveur SQL Server :

```
const string chaine = "data source= PYACOUBS\MSSQL2012; Initial  
Catalog = PatocheBd; User Id = Patoche; password =remi2002";
```

```
private DataContext sqlBd = new DataContext(chaine);
```

1. Obtenir les sources de données :

La source de données peut être un tableau de données ou une partie des tables de la base de données.

Lorsque la source de données utilise des tables de la base de données, celle-ci est obtenue comme suit :

```
Table <NomTableBD>lesdepartements = sqlBd.GetTable <NomTableBD >();
```

Exemple

```
Table <departements> lesdepartements = sqlBd.GetTable<departements>();
```

La source de données est `lesdepartements` qui correspond à la table `Departements` de la base de données.

2. Créer la requête.

La requête spécifie les informations à récupérer de la source ou des sources de données. Elle peut également spécifier la manière dont ces informations doivent être triées, regroupées et mises en forme avant d'être retournées. Une requête est stockée dans une variable et initialisée avec une expression de requête. C# a introduit une nouvelle syntaxe pour simplifier l'écriture des requêtes.

L'expression de requête contient trois clauses : `from`, `where` et `select`. Remarquez que l'ordre des clauses est inversé par rapport à l'ordre dans SQL. La clause `from` spécifie la source de données, la clause `where` applique le filtre et la clause `select` spécifie le type des éléments retournés.

Exemple1

```
var requ = from de in lesdepartements
select de.nomdepartement;
```

Exemple 2 :

```
lesdepartements = sqlBd.GetTable<departements>();
leemployes = sqlBd.GetTable<Employes>();
var jointure = from emp in leemployes
               join de in lesdepartements
               on emp.deptno equals de.deptno
               where de.nomdepartement == "inf"
               orderby emp.nom
               select emp.nom;
```

3. Exécuter la requête

Comme indiqué précédemment, la variable de requête elle-même stocke simplement les commandes de requête. L'exécution réelle de la requête est différée jusqu'à ce que

vous itérez la variable de requête dans une instruction foreach. Ce concept, connu sous le nom d'exécution différée.

```
foreach (var res in jointure)
{
    Console.WriteLine(res);
}
```

Pour les requêtes de mise à jour (INSERT, UPDATE et DELETE) elles se font au niveau programmation avec les méthodes :

InsertOnSubmit() de la source de données; Uniquement pour l'insertion. Suivi de SubmitChanges();

SubmitChanges(); de la connexion (DataContext), pour les UPDATE

DeleteAllOnSubmit(), pour la suppression. Suivi de SubmitChanges();

4. Créer un projet LINQ

1. Démarrer un projet windows forms ;
2. Établir une connexion au serveur;

Ajouter une connexion

Entrez les informations pour vous connecter à la source de données sélectionnée ou cliquez sur "Modifier" pour sélectionner une autre source de données et/ou un autre fournisseur.

Source de données :
Microsoft SQL Server (SqlClient) [Modifier...]

Nom du serveur :
PYACOUBS\MSSQL2012 [Actualiser]

Connexion au serveur

Utiliser l'authentification Windows

Utiliser l'authentification SQL Server

Nom d'utilisateur : Patoche

Mot de passe : ●●●●●●

Enregistrer mon mot de passe

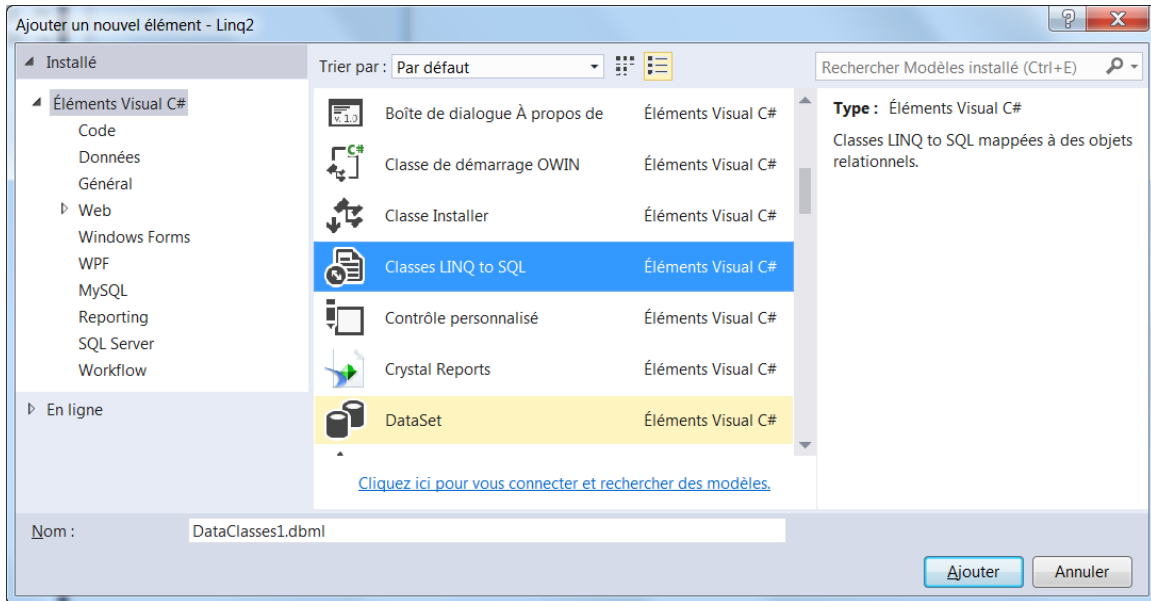
Connexion à la base de données

Sélectionner ou entrer un nom de base de données :

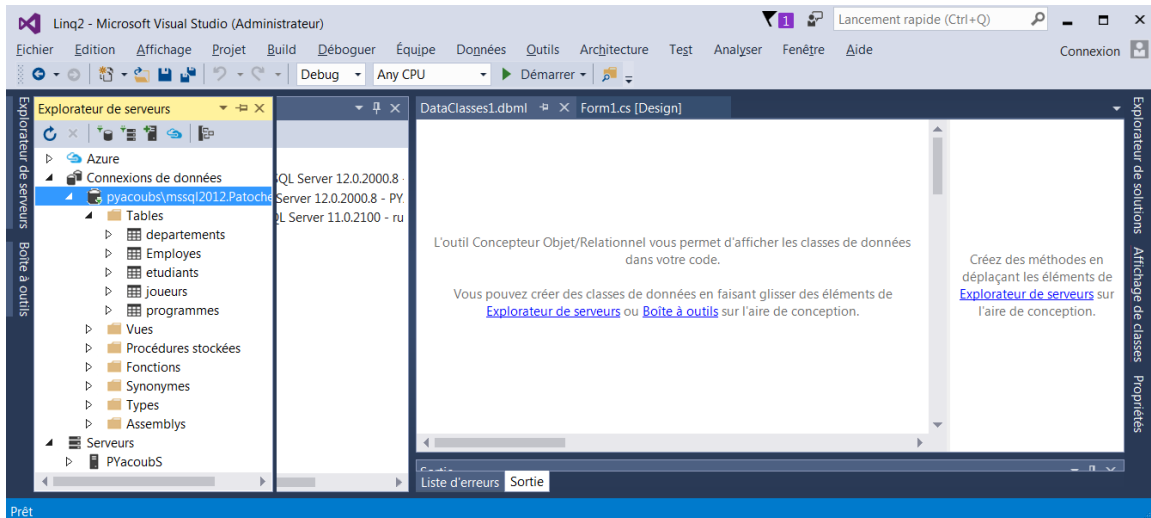
- bdRuby
- ImportOracle
- MaBd
- master
- model
- msdb
- nouvelleBd
- PatocheBd
- tempdb
- UneBd
- User1Bd

[Tester la connexion] [OK] [Annuler]

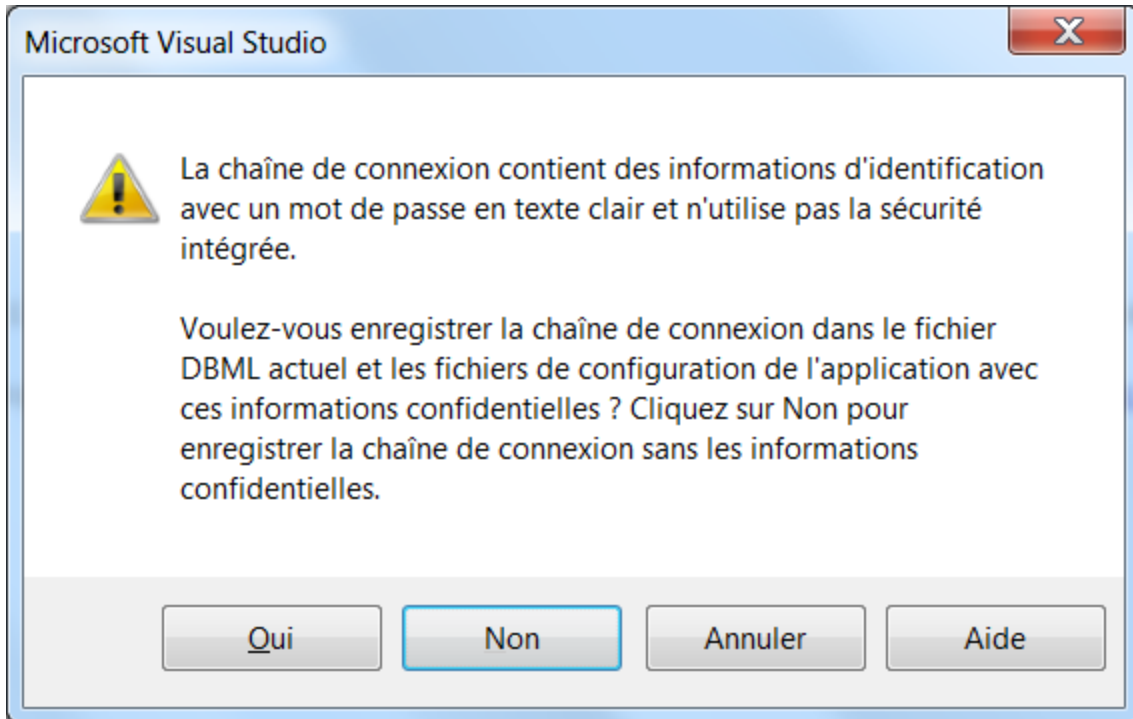
3. Ajouter une classes LINQ to SQL : Explorateur de solution, ajouter une classe, choisir



4. Ajouter les sources de données au projet : Dérouler votre serveur, puis glisser vos tables sur Explorateur de serveurs.



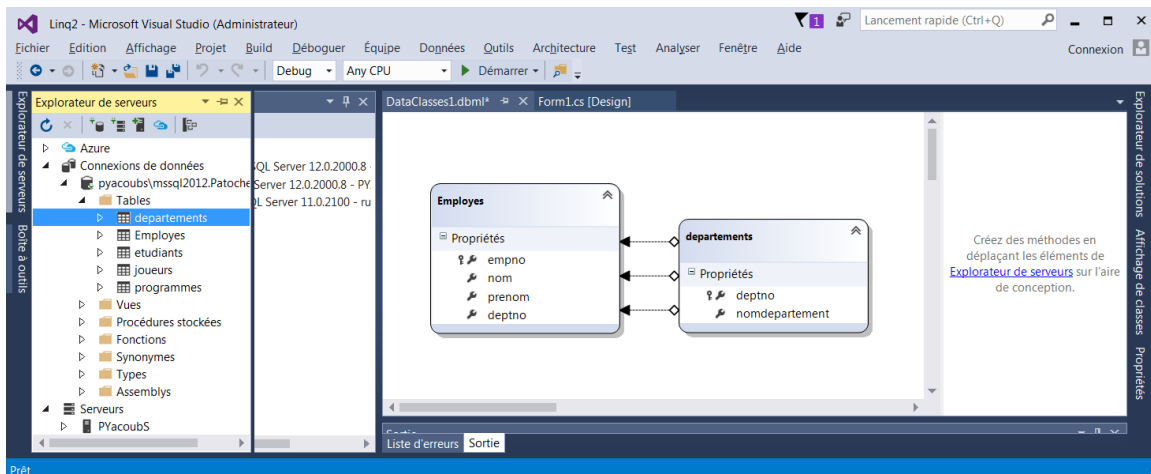
Le système va vous demander d'enregistrer la chaîne de connexion dans App.config. Répondez par OUI.



5. Vérifier que tout est OK

Vérifier que dans votre App.config vous avez votre chaîne de connexion

Voici ce que vous allez avoir une fois que les deux tables Employés et départements sont sélectionnés.



6. Environnement SQL Server.

Vous pouvez créer vos requêtes à la base de données directement dans cet environnement (plus besoin de démarrer SQL server)

5 Exemple d'application avec LINQ

Voici ce que nous souhaitons avoir comme application Windows Forms avec LINQ to SQL.

Notez que le code n'est pas optimal. Il peut être simplifié. Et il faudra faire une capture des exceptions :

Form1

Liste des employes selon le département

Bien
bla
Fafar
Hugo
Lemieux
Patoche

informatique

Liste departements

Nom et prenom	
Patoche	Alain
Fafar	Gavroche
Bien	Thierry
Lemieux	Lebon

Liste employes

Insertion

Modifier

Supprimer

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.Linq;

```

```

namespace WindowsFormsLinqtoSQL
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }

```

```

//Declarations et connexion à la BD
    const string chaine = "data source= PYACOUBS\\MSSQL2012;
Initial Catalog = PatocheBd; User Id = Patoche; password =remi2002";
    private DataContext sqlBd = new DataContext(chaine);
    private Table<Employes> lesemployes;
    private Table<departements> lesdepartements;

```

```

// Code du bouton qui affiche dans le DGV. Le DGV a une seule colonne.
    private void liste_Click(object sender, EventArgs e)
    {
        lesemployes = sqlBd.GetTable<Employes>();
        var requete = from emp in lesemployes
        select (emp.nom + " " + emp.prenom);

        foreach (var nomp in requete)
        {
            dgvEmp.Rows.Add(nomp);
        }
    }

```

```

//Bouton qui appelle la fonction qui liste les départements
    private void listDept_Click(object sender, EventArgs e)
    {
        listedepartement();
    }

```



```
//fonction qui liste les départements et les met dans la comboBox
// Cette fonction est appelée par le bouton Liste départements. (nom du
bouton //listDept.
```

```
private void listedeptement()
{
    Table <departements> lesdepartements =
    sqlBd.GetTable<departements>();

    var req = from de in lesdepartements
    select de.nomdepartement;

    foreach (var nomd in req)
    {
        comboBox1.Items.Add(nomd);
        comboBox1.SelectedIndex = 0;
    }
}
```

```
//Lorsque la sélection change dans le comboBox, la liste des employés
change. C'est le code du comboBox
```

```
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    listEmp.Items.Clear();
    empDept();
}
```

```
//Fonction qui affiche les employés selon le département. Il y a une
jointure entre employés et départements.
// la liste des employés change lorsque le contenu du comboBox change
```

```
private void empDept()
{
    lesdepartements = sqlBd.GetTable<departements>();
    lesemployes = sqlBd.GetTable<Employes>();
    var jointure = from emp in lesemployes
    join de in lesdepartements
    on emp.deptno equals de.deptno
    where de.nomdepartement == comboBox1.Text
    orderby emp.nom
    select emp.nom;

    foreach (var nomp in jointure)
    {
        listEmp.Items.Add(nomp);
    }
}
```

//Insertion: On déclare un objet de Type Employes, puis on applique
//les méthode InsertOnSubmit() et SubmitChanges();

```
private void insertion_Click(object sender, EventArgs e)
{
    lesemployes = sqlBd.GetTable<Employes>();
    Employes emp1 = new Employes
    {
        empno = 222,
        nom = "Linq",
        prenom = "SQL"
    };

    lesemployes.InsertOnSubmit(emp1);
    sqlBd.SubmitChanges();
}
```

//Modification

```
private void modifier_Click(object sender, EventArgs e)
{
    lesemployes = sqlBd.GetTable<Employes>();

    var requete = from emp1 in lesemployes
                  where emp1.empno == 12
                  select emp1;

    foreach (Employes emp1 in requete)
    {
        emp1.nom = "NouveauNom";
        sqlBd.SubmitChanges();
    }
}
```

//Suppression:

```
private void supprimer_Click(object sender, EventArgs e)
{
    lesemployes = sqlBd.GetTable<Employes>();
    var requetesup = from empsupp in lesemployes
                     where empsupp.empno == 222
                     select empsupp;
    foreach (var emp1 in requetesup)
    {
        lesemployes.DeleteAllOnSubmit(requetesup);
        sqlBd.SubmitChanges();
    }
}
}
```

Sources :

<https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/sql/linq/getting-started>

[https://msdn.microsoft.com/fr-fr/library/system.data.linq.datacontext\(v=vs.110\).aspx](https://msdn.microsoft.com/fr-fr/library/system.data.linq.datacontext(v=vs.110).aspx)