Automne

Introduction à LINQ to SQL

Saliha Yacoub
COLLÈGE LIONEL-GROULX

Table des matières

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.

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.

Dans LINQ to SQL le modèle relationnel d'une base de données est mappé en un modèle objet exprimé dans le langage de programmation (C#). Lorsque l'application est exécutée, LINQ to SQL convertit en SQL les requêtes intégrées au langage dans le modèle objet et les envoie à la base de données pour exécution. Lorsque la base de données renvoie les résultats, LINQ to SQL les convertit en objets que vous pouvez utiliser dans votre propre langage de programmation.

Toutes les opérations de requête LINQ comportent trois actions distinctes :

- Obtenir la source de données :
- Créer la requête.
- Exécuter la requête.

1. Obtenir la source de données :

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.

Quelques constructeurs

Constructeurs	**Explications**

DataContext(IDbConnection)	Initialise une nouvelle instance de la classe DataContext en référençant la connexion utilisée par le .NET Framework. La connexion est obtenue comme suit
	`SqlConnection sqlconn= new SqlConnection();`
	`sqlconn.ConnectionString = unechainedeconnexion;`

DataContext(String)	Initialise une nouvelle instance de la classe DataContext en fournissant une chaine de connexion ou l'emplacement du fichier .mdf de la BD La chaine de connexion est sous la forme : `string chaineConenxion = "Data Source=M-INFO-SY\\SQLEXPRESS2017;Initial Catalog=BdJeu;User ID=patoche;Password=123456";` le Data Source est le nom du serveur. Ce qui représente le nom de l'ordinateur suivi du nom de l'instance du serveur. Initial Catalog représente le nom de la base de données. Le User Id, représente le nom de la connexion sql Server. (user). Évidemment la chaine de connexion, change d'un SGBD à l'autre

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);
```

ou

```
private DataClasses1DataContext dcBd = new DataClasses1DataContext(chaine);

DataClasses1 est le nom de la classe LINQ.
```

Quelques méthodes importantes :

Méthodes	**Rôles**
CreateDatabase()	Créer une base de données sur le serveur. Le nom de la base de données est celui de chaine de connexion (si la BD n'existe pas)

ExecuteCommand(String, Object[])	Exécute une commande SQL directement sur la base de données :
	`dcBd.ExecuteCommand("update questions set enonce ='qui suis-je ?' where idQuestion =12 ");`
	`dcBd.ExecuteCommand("create table joueurs(id int identity primary key, nom varchar(30))");`

GetTable<TEntity>()	Retourne une collection d'objets d'un type particulier, où le type est défini par le paramètre TEntity. `Table<categories> tablecategorie = dcBd.GetTable<categories>();`

SubmitChanges()	Permet de mettre à jour la base de données.
	`Table<Questions> qcm = dcBd.GetTable<Questions>();`
	`Questions qcm = new Questions();`
	`{`
	`qcm.enonce = "LINQ to SQL";`
	`qcm.flag = 1;`
	`qcm.difficulte = "facile";`
	`qcm.code_categorie = 1;`
	`};`
	`dcBd.Questions.InsertOnSubmit(qt);`
	`dcBd.SubmitChanges();`

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>();
        lesemployes = sqlBd.GetTable<Employes>();
        var jointure = from emp in lesemployes
                       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 itériez 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 :

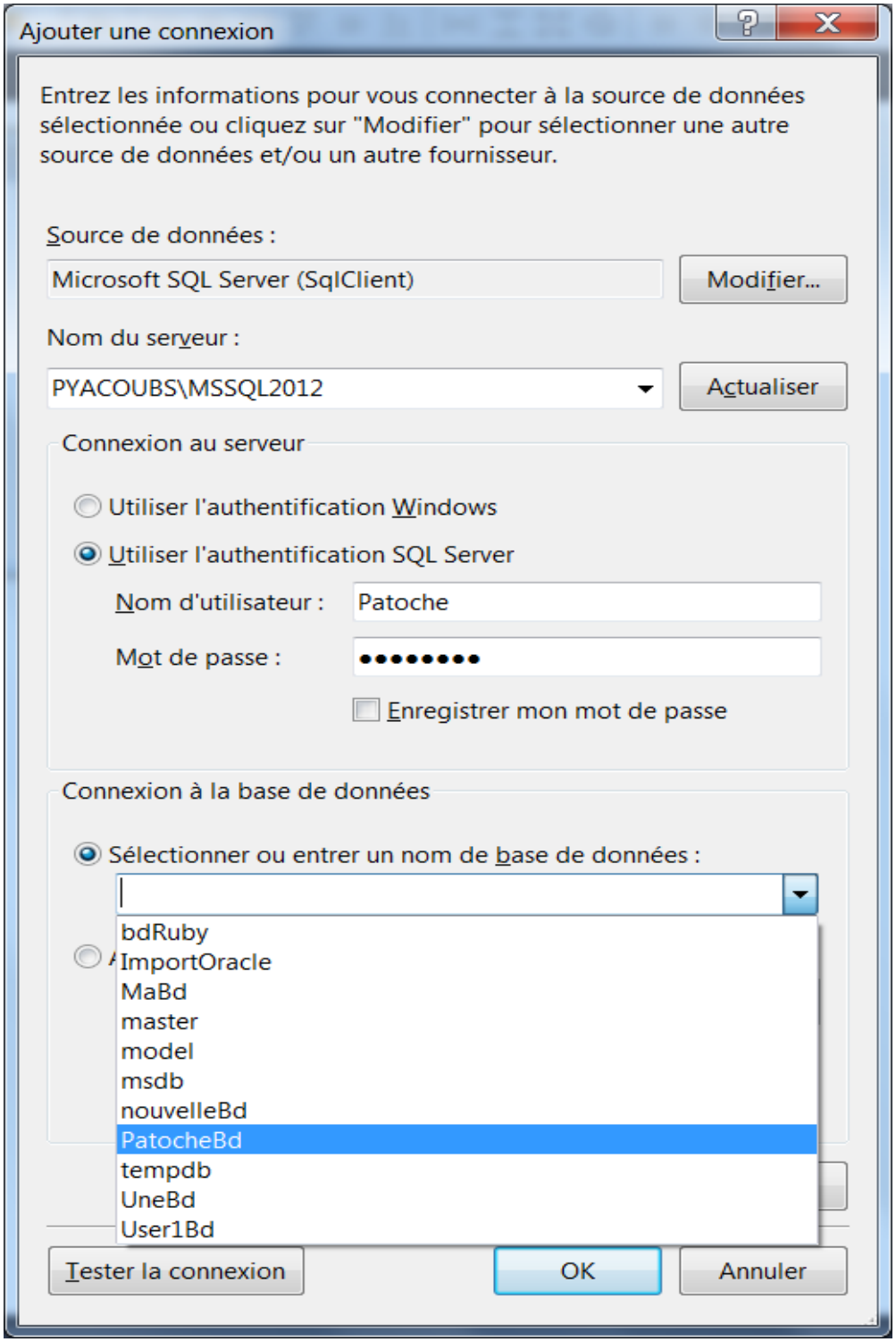
`InsertOnSubmit()` de la source de données; Uniquement pour l'insertion. Suivi de `SubmitChanges();`

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

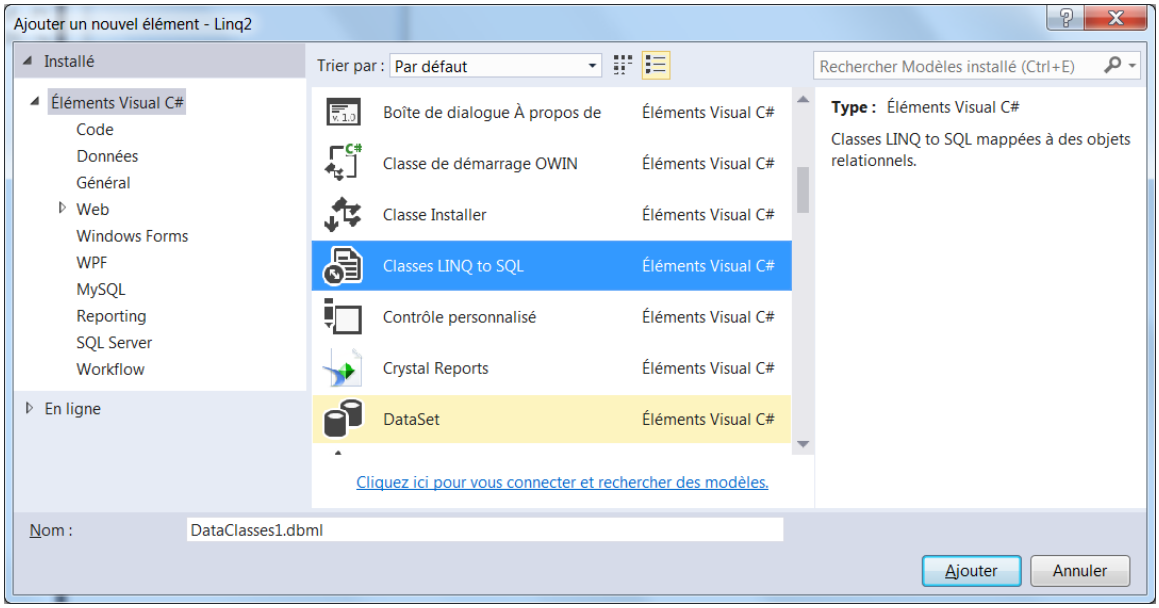
`DeleteAllOnSubmit(),` pour la suppression . Suivi de `SubmitChanges();`

4. Créer un projet LINQ

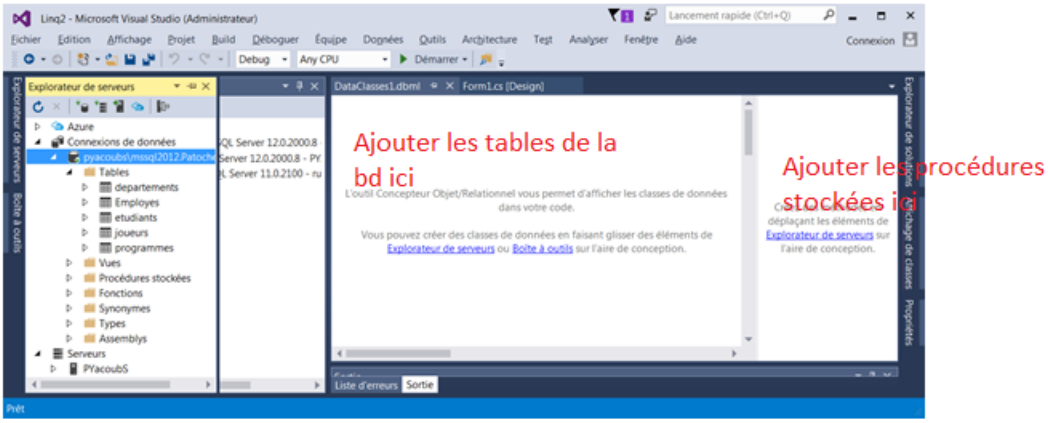
1. Démarrer un projet Windows Forms ;
2. Établir une connexion au serveur;



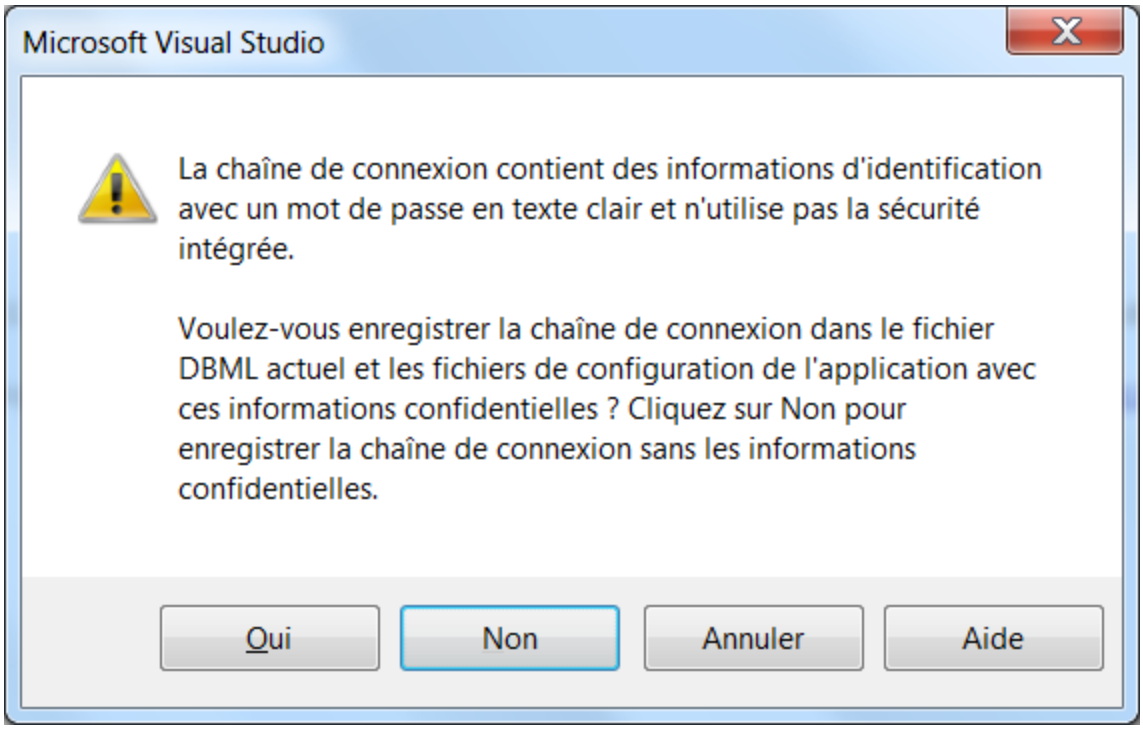
3. Ajouter une classes LINQ to SQL : Explorateur de solution, ajouter une classe, choisir



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



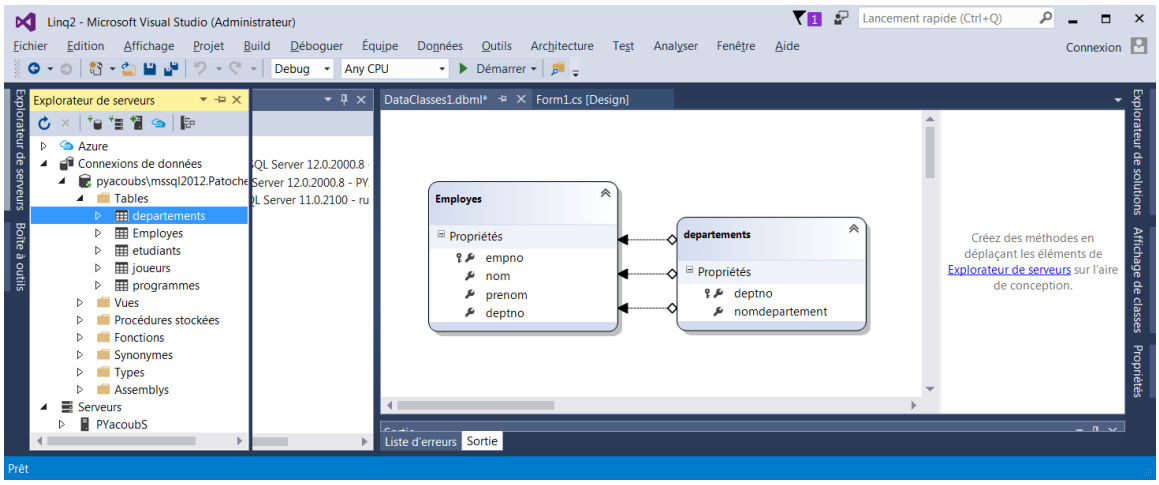
Le système va vous demander d'enregistrer la chaine de connexion dans App.config. Répondez par OUI.



5. Vérifier que tout est OK

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

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

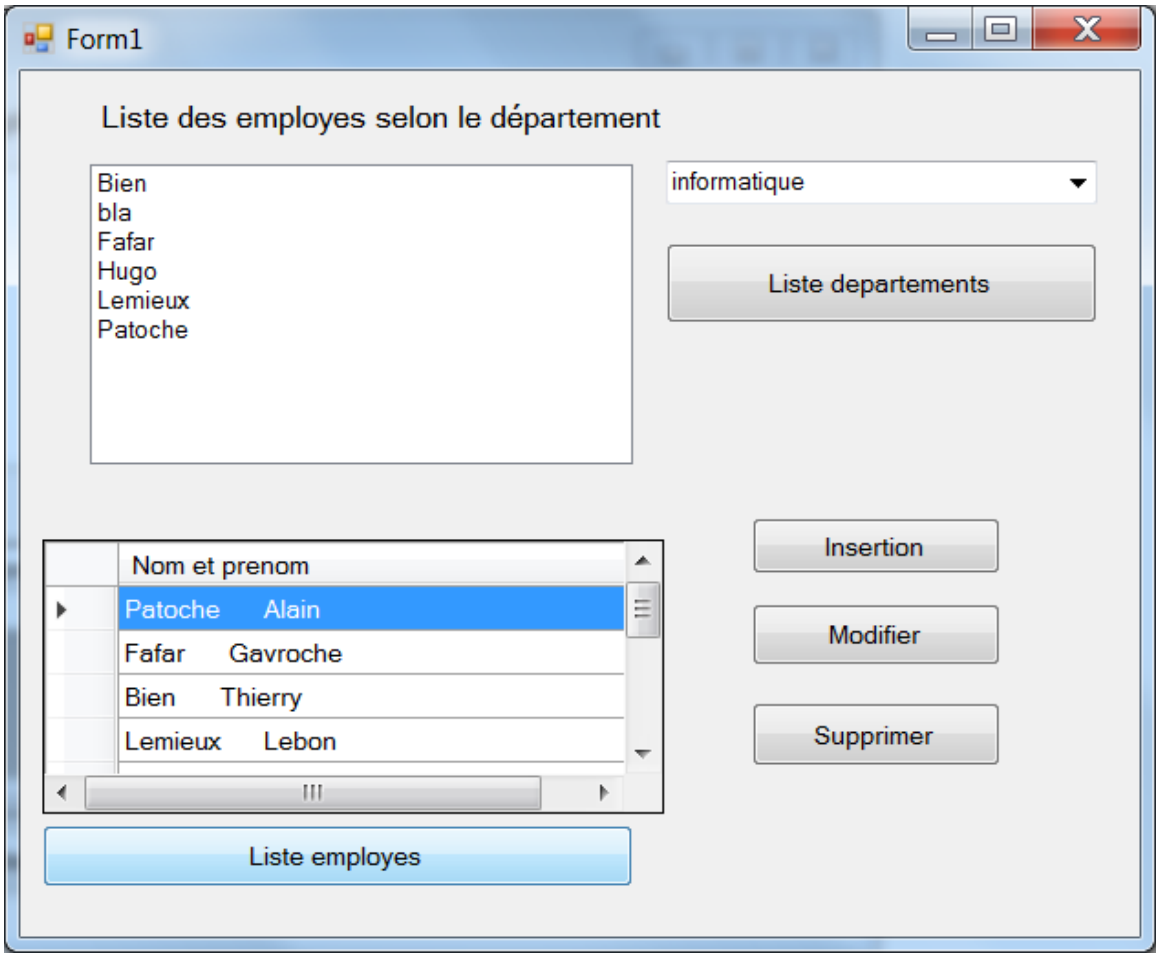


Environnement SQL Server.
Vous pouvez créer vos requêtes à la base de données directement dans cet environnement (plus besoin de démarrer SQL server)

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 :



```
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 departements. (nom du
bouton //listDept.

private void listedepartement()
        {

       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 employes selon le département. Il y a une
jointure entre employes et departements.
// la liste des employes 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);
            }
        }
```

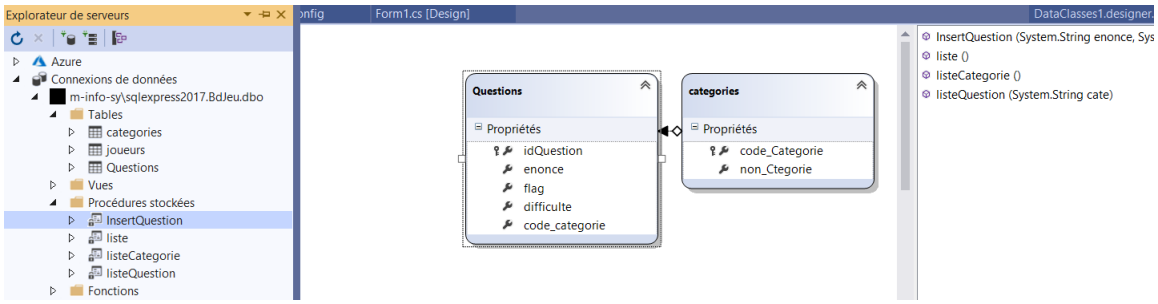
```
//Insertion: On déclare un objet de Type Employes, puis on applique
//les méthode InsertOnSubmit() et SubmitChanges();
       private void insertion_Click(object sender, EventArgs e)
        {
            lesemployes = sqlBd.GetTable<Employes>();
             Employes emp1 = new Employes
            {
                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();
            }
        }
    }
}
```

6. Utilisation de procédures stockées :

Pour utiliser les procédures stockées, il faut tout simplement les appeler en fournissant les valeurs des paramètres

Si vous n'avez pas ajouter vos procédures stockées à votre projet, alors c'est le moment de le faire :

Il suffit de glisser les procédures stockées au bon endroit.

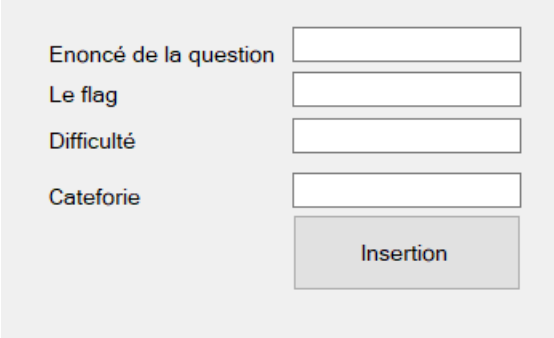


Exemple : appel de la procédure : InsertQuestion

Cette procédure est définie comme suit : Elle a quatre (4) paramètres.

```
create procedure InsertQuestion
(@enonce varchar(200), @flag int,@difficulte varchar(20),@categ int) as
begin
insert into Questions values(@enonce,@flag,@difficulte,@categ);
end;
```

La table Question a une clé primaire IDENTITY



```
private void BtnAajouter_Click(object sender, EventArgs e)
        {
   int leFlag = int.Parse(textFlag.Text.ToString());
   int laCategorie = int.Parse(textCateg.Text.ToString());
   dcBd.InsertQuestion(textEnonce.Text, leFlag, textDifficulte.Text,laCategorie);
        }
```

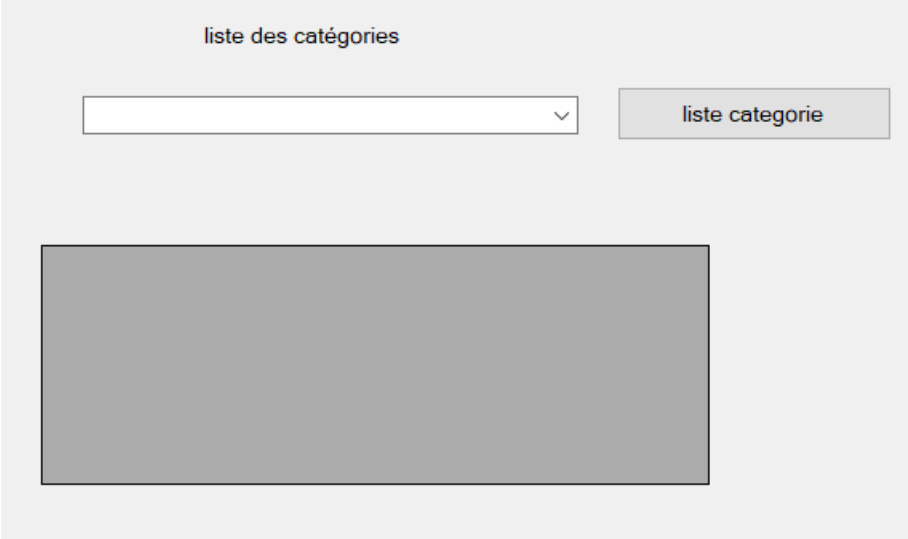
Les paramètres de la procédure correspondent aux valeurs rentrées par les zones de texte.

Le code complet plus bas avec la declaration du DataContext, et de la chaine de connexion.

```
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 LinqtoSQL
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

      //declaration du DataContext avec la chaine de connexion

         const string chaineConenxion = "Data Source=M-INFO-
SY\\SQLEXPRESS2017;Initial Catalog=BdJeu;User ID=patoche;Password=123456";
       private DataClasses1DataContext dcBd = new
       DataClasses1DataContext(chaineConenxion);



        private void BtnAajouter_Click(object sender, EventArgs e)
        {
    int leFlag = int.Parse(textFlag.Text.ToString());
    int laCategorie = int.Parse(textCateg.Text.ToString());
    dcBd.InsertQuestion(textEnonce.Text, leFlag, textDifficulte.Text,laCategorie);
        }
}

}
```

Autre exemple :

Afficher la liste des questions selon la catégorie



Le bouton, liste categorie permet d'afficher les noms de catégorie dans un comboBox. Cette opération fait appel à une procédure stockée `listeCategorie()` pour liste des catégorie.

Le code de la procédure est le suivant :

```
CREATE  procedure  listeCategorie as
BEGIN
select non_ctegorie from categories;
END;
```

Voici le code C# du bouton :

```
        private void BtnCategorie_Click(object sender, EventArgs e)
        {
            foreach (var listeCa in dcBd.listeCategorie())
            comboCategorie.Items.Add(listeCa.non_ctegorie);
        }
```

L'évènement `SelectedIndexChanged` du comboBox permettra d'aller chercher les question en fonction de la catégorie, et donc d'appeler la procédure `listeQuestion(nomCategorie)` . On utilise un DataGridView pour afficher la liste des Questions.

Le code de la procédure `listeQuestion(nomCategorie)` . est définie comme suit :

```
CREATE procedure listeQuestion(@cate varchar(30)) AS
BEGIN
select enonce,difficulte from Questions inner join categories
on Questions.code_categorie = categories.code_Categorie where
non_Ctegorie =@cate;
END;
```

Et voici le code C#

```
  private void ComboCategorie_SelectedIndexChanged(object sender, EventArgs e)
        {
            string nomCat = comboCategorie.SelectedItem.ToString();
            dgvQuestions.DataSource = dcBd.listeQuestion(nomCat);
        }
```

Sources :

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

https://docs.microsoft.com/en-us/dotnet/api/system.data.linq.datacontext?view=netframework-4.8