Gestion des données entre C# et MySQL

Contexte

Ce tutoriel permet de se connecter à une base de données MySQL via du code C#, et d'y exécuter des requêtes SQL.

Sommaire

I. Pré requis

Il est nécessaire d'avoir créé une base de données sur le serveur SQL Server.
Il est préférable de créer certaines classes spéciales où seront rangées toutes les méthodes d'accès à la base de données. On nomme ce type de classe « une classe DAO ».

Ce documentation s'applique à un projet réalisé sous Visual Studio 2010.

a. Lancer un projet

Ce tutoriel s'applique à un projet réalisé sous Visual Studio 2010.
Ouvrir ou créer un projet.

b. Directives C#

Il est nécessaire d'ajout cette directives dans chaque fichier en lien avec la base de données :

```
using System.Data.Sql;
using System.Data.SqlClient;
```

II. Connexion à la base de données

Procédure de connexion à la base de données :

a. Créer la chaine de connexion

Le format de la chaine de connexion SQL Server est :

```
String chaineConnexion = @"Data Source= IP_Server; Initial Catalog= Database_Name;
User ID=User_Name; Password = User_Password";
```

b. Créer une variable SQLConnection

Cette variable permettra de stocker la connexion à la base de données.

```
SqlConnection connexion = new SqlConnection(chaineConnexion);
```

c. Ouverture et fermeture de la connexion

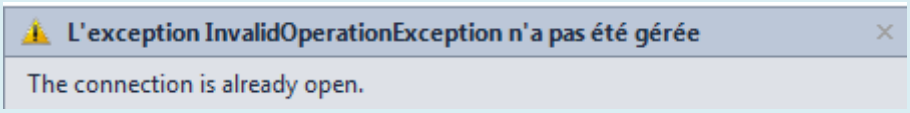
Avant chaque requête il faut ouvrir la connexion pour avoir accès à la base de données grâce à la commande Open() .

```
connexion.Open();
```

Il est important de fermer chaque connexion après chaque ouverture avec la commande Close() .

```
connexion.Close();
```

Fermer une connexion après utilisation permet d'économiser en ressources système et permet aussi d'éviter les erreurs :
Si une connexion tente d'être ouverte alors qu'elle l'est déjà, une exception MySQL empêcheras l'exécution du code.



III. Création d'un commande

a. Créer une variable SQLCommand

Cette variable permet de lier la requête à la base de données en lui donnant les informations de connexion.
Pour créer une variable SqlCommand, il faut lui passer en paramètre la requête SQL et la variable de connexion (SqlConnection) :

```
string requete = "Select * from commercial";
SqlCommand requete = new SqlCommand(requete, connexion);
```

IV. Sélection d'information

a. Select

Dans le cas d'une requête SQL de sélection, il faut récupérer le résultat de la commande dans une variable de type `SqlDataReader`.

Le `SqlDataReader` est un objet utilisé pour récupérer des données à partir de la base de données. Il fournit un accès rapide en lecture uniquement permettant d'interroger les résultats. Il est le moyen le plus efficace pour récupérer des données à partir des tables.

```
connexion.Open();
SqlCommand requete = new SqlCommand("Select * from commercial", this.connex);

       using (SqlDataReader reader = requete.ExecuteReader())
       {
           while (reader.Read())
           {
              //recupération des données…
           }
       }
       catch (SqlException ex)
       {
           MessageBox.Show(ex.Message);
       }
       connexion.Close();
```

b. Les méthodes de SqlCommand et SqlDataReader

La méthode ExecuteReader()

Permet d'exécuter la requête et de récupérer les données dans la variable (`MySqlDataReader`) sous forme de ligne.

La méthode HasRows()

Vérifie si des données ont été entrées dans la variable (`MySqlDataReader`), afin de lancer le traitement.
Renvoi la valeur 0 si aucune ligne n'est présente, sinon retourne la valeur 1.

La méthode Read()

Fait avancer le lecteur des données ligne par ligne.
Il retourne vrai s'il existe encore des lignes; sinon false.

c. Récupération des données

La récupération des données se fait via l'exécution de commande sur chaque ligne du lecteur de commande (SqlDataReader) et dépendant du type de données à récupérer :

Il est possible de retrouver toutes les méthodes C# permettant ce traitement via ce lien MSDN.

V. Envoi de requête SQL

Mise à part les requêtes de sélection qui récupère un certains nombre d'informations à traiter, les autres requêtes SQL sont plus simple à traiter.

```
MySqlCommand commande = new MySqlCommand(requete, connex);
            try
            {
                connexion.Open();
                commande.ExecuteNonQuery();
            }
            catch (MySqlException ex)
            {
                 MessageBox.Show(ex.Message);
            }
            finally
            {
                connexion.Close();
            }
```

a. Particularité de SQL Server

b. La méthode ExecuteNonQuery()

Cette méthode permet d'envoyer la requête dans la base de données.

Si un problème survient, la commande ne sera pas traitée par la base de données. L'objet de type `MySqlException` sera implémenté avec les éléments liés à l'erreur survenue.

Cependant si aucune exception n'est relevée, alors la requête à était exécuté avec succès.

Conclusion

Ainsi il est possible de communiquer via des classes DAO avec une base de données SQL Server depuis un programme codé en C#.
Une documentation annexe fera l'objet des procédures stockées, afin de sécurisé le code de l'application et limité les erreurs.