ADO.NET

Le mode connecté

Introduction

- ADO .NET (ActiveX Data Object) supporte l'accès à différentes sources de données.
- Espaces de noms disponibles dans le Framework .NET
  - System.Data: contient des classes indépendantes des sources de données.
  - System.Data.OleDb: accès aux sources de données qui utilisent un fournisseur OleDb (Exemple MS Access)
  - System.Data.Odbc: accès à une source de données via un pilote ODBC.
  - System.Data.SqlClient: accès au bases de données Sql Server.
- D'autres bibliothèques (fournisseurs ou connecteurs) spécifique à un SGBDR sont développées par des éditeurs tiers:
  - Oracle: System.Data.Oracle
  - Mysql: MySql.Data.MySqlClient

- Etape 1 : importer l'espace de noms du connecteur d'accès à la source de données et l'espace de noms System.Data
- Etape2: création et ouverture de la connexion.
  - Chaînes de connexion: contient les informations nécessaires pour établir la connexion avec la source de données:
    - MS ACCESS:

```
String sCnx= "provider=Microsoft.Jet.OleDb.4.0; Data Source='C:/service.mdb'";
```

    - SQL SERVER
      - Mode authentification windows:

```
string sCnx ="Data Source= nom_machine\\nom_instance; Initial
   Catalog=base;Integrated Security=true;"
```

      - Mode Authtentification SQL Server:

```
string sCnx=" Data Source= nom_machine\\nom_instance ;Initial Catalog=base;User
Id=nom_utilisateur; Password=pass;";
```

  - Création de la Connexion: Un objet de type XConnection où X désigne un préfixe spécifique au connecteur utilisé permet de gérer une connexion.
    - Exemple
      - SQL Server: `SqlConnection cnx = new SqlConnection(sCnx);`
      - MS ACCESS: `OleDbConnection cnx = new OleDbConnection(sCnx);`
  - ouverture de la connexion:

```
cnx.Open();
```

- Remarques
  - la méthode Open() déclenche deux types d'exceptions:
  - InvalidOperationException : si la connexion est déjà ouverte.
  - OleDbException : s'il y a une erreur d'accès à la base de données.
  - La connexion peut être aussi créée à l'aide du constructeur par défaut:

```
SqlConnection cnx = new SqlConnection(();
cnx.ConnectionString=sCnx;
```

- Etape 3: Création de la commande SQL

Un Objet de type XCommand (X est un préfixe spécifique au fournisseur d'accès aux données) peut contenir les informations d'une requête SQL:

Instruction SQL: `String sql = "Select desgination from plat";`

`SqlCommand cmd = new SqlCommand(sql, cnx);`

- Remarques:
  - l'objet **cmd** peut être aussi créé à l'aide du constructeur par défaut:

```
SqlCommand cmd = new SqlCommand();
cmd.Connection = cnx ;
cmd.CommandText = sql ;
```

  - La propriété CommandType permet de détérminer la nature du contenu de la propriété CommandText, et peut prendre les valeurs suivantes :
    - cmd.CommandType = CommandType.Text : instruction Sql (valeur par défaut).
    - cmd.CommandType = CommandType.Stored Procedure : Nom d'une procédure stockée.
    - cmd.CommandType = CommandType.Table Direct : Nom d'une table.

- Etape 4 exécution de la commande

L'objet OleDbCommand possede 3 méthodes pour exécuter une instruction SQL selon le type de résultat qu'elle retourne.

- cmd.ExecuteNonQuery() : à utiliser dans le cas d'une instructions SQL de type Insert,Update ou Delete . Cette méthode retour le nombre de lignes affectées par l'instruction SQL.

- cmd.ExecuteScalar() : Exécute une instruction SQL qui retourne une valeur, exemple : select count(*) from plat. Le type de retour de cette méthode est Object.
- cmd.ExecuteReader() : Exécute une instruction SQL qui retourne un jeu d'enregistrement. La méthode retourne les résultats dans un curseur (de type SqlDataReader).

- Etape 5: Récupérer le résultat de la requête

un objet de type XDataReader (où X désigne un préfixe spécifique au connecteur utilisé), représente un curseur en avant uniquement (Forward Only).

```
SqlDataReader dr = cmd.ExecuteReader();
```

- Lors de sa création le curseur est positionné avant le premier enregistrement, l'appel de la méthode dr.Read() déplace le curseur vers l'enregistrement suivant et retourne true si cet enregistrement existe et false sinon.

- Remarque:
  - Une les données récupérées le curseur et la connexion doivent être fermés:
    - dr.Close();
    - cnx.Close();