Utilisation d'ADO .NET

Introduction :

Voyons comment .NET peut être utilisé pour tirer profit, au maximum, des performances et des possibilités, de cette base de données.

Connexion aux bases de données :

Comme toutes les autres bases de données que vous pouvez utiliser sous .NET, vous avez plusieurs moyens pour les joindre :

- Via un provider intégré au Framework .NET (**SQL** pour SQL Server et **OLEDB** pour plusieurs SGBD, dont Access): bien qu'efficace, cette technique peut ne pas vous satisfaire si vous cherchez à avoir des performances optimales.

- Via un provider dédié: qu'il soit créé par la société qui a créé la base de données (**ODP** pour Oracle par exemple), ou par une autre société, préférez toujours ce type de provider. En effet, conçus exclusivement pour cette base, ils vous permettront d'avoir les meilleurs résultats possibles.

- Via **ODBC**: méthode "universelle" de connexion à une base de données. Cette technique fonctionne toujours, mais ne vous garantit pas les meilleurs résultats.

Ado.net propose deux modes d'accès, le **mode connecté** et le **mode déconnecté**.

Le mode connecté *:*

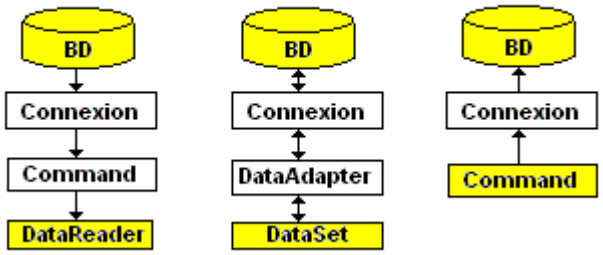
Ce mode classique maintient la connexion à la base de données entre l'ouverture et la fermeture, il permet de créer un mécanisme de "**curseur**" permettant de parcourir les données ligne à ligne. Ado.net ne propose qu'un **accès en lecture -en avant seulement-** avec ce mode. Il est approprié pour parcourir des **tables volumineuses rapidement**.

Le mode déconnecté *:*

C'est la grande nouveauté de l'architecture .net. **Après une connexion et le chargement de données, tout se passe en mémoire**, le client est alors déconnecté du serveur. Ceci libère les ressources du serveur de données et rend plus facile la manipulation. En revanche le mécanisme de cohérence et d'intégrité des données est exigeant, dont cela ralentit l'application par rapport au mode connecté.

Les objets ADO.NET

Résumons les différents objets nécessaires pour travailler sur une BD:



Noter bien le sens des flèches: le DataReader est en lecture seule, le DataSet peut lire et écrire, l'objet Command peut modifier la BD.
Ce schéma souligne aussi les objets intermédiaire nécessaire: un objet connexion dans tous les cas, un objet Command pour le DataReader, un objet DataAdapter pour le DataSet.
Enfin certains contrôles comme le DataGrid par exemple peuvent afficher des données à partir d'un DatSet.
On peut aussi créer une DataTable à partir d'un DataSet et alimenter une listbox ou une ComboBox.

Objet	**Description**
Connection	Ouvre une connexion vers une source de données spécifiques
Command	Exécute une commande sur une source de données
DataReader	Lit un flux de données à partir d'une source de données en mode **connecté**. Le mode d'accès est en lecture seule avec un curseur en avant seulement.

L'objet connexion

Le framework .Net propose ainsi des objets de connexion différents en fonction du type de fournisseur de données choisies. Par exemple vous devrez utiliser l'objet **OleDbConnection** si votre fournisseur est un fournisseur **OleDb ou MySqlConnection si le** fournisseur est un fournisseur Mysql.

L'ouverture d'une connexion est réalisée par la méthode **Open** et la fermeture par la méthode **Close**.

Exemple de gestion d'une connexion

```
using MySql.Data.MySqlClient;

private ConnexionSql(string unProvider, string uneDataBase, string unUid, string unMdp) {


        try
        {
        string connString;
            connString = "SERVER=" + unProvider + ";" + "DATABASE=" +
            uneDataBase + ";" + "UID=" + unUid + ";" + "PASSWORD=" + unMdp + ";";
            try
            {
                mySqlCn = new MySqlConnection(connString);
            }
            catch (Exception emp)
            {
                throw (emp);
            }
        }
        catch (Exception emp)
        {
            throw (emp);
        }



    }
```

L'objet Command

Une fois la connexion vers une base de données effectuée, vous pouvez exécuter une requête et récupérer son résultat en utilisant l'objet Command. La création d'un objet Command nécessite l'instanciation d'un objet **SqlCommand**. Cet objet expose différentes méthodes **Execute** à utiliser selon le résultat attendu :

- La méthode **ExecuteReader** peut être utilisée pour récupérer un jeu d'enregistrement et retourne un objet **DataReader** ;
- La méthode **ExecuteScalar** récupère une valeur unitaire ;
- La méthode **ExecuteNonQuery** exécute une commande ne retournant pas de lignes.

Avec un objet **Command** on peut manipuler directement la BD (Update, Insert, Delete)

Exemple d'un objet Command ne retournant pas de lignes:

```
MySqlCommand mysqlCom = new MySqlCommand("delete from employe where idEmp = 10", mySqlCn);

mysqlCom.ExecuteNonQuery() ;
```

Exemple d'un objet Command retournant des lignes:

```
MySqlCommand mysqlCom = new MySqlCommand("select * from employe, mySqlCn);

MySqlDataReader  reader = mysqlCom.ExecuteReader();
```

L'objet DataReader

Avec un objet **DataReader (mode connecté) on extrait les données en lecture seule**: c'est rapide; on peut lire uniquement les données et aller à l'enregistrement suivant. Pour gérer un DataReader on a besoin d'un objet **Command** (qui correspond à une requête SQL).

l'objet DataReader permet de récupérer d'une source de données un flux en lecture seule en avant seulement (read only, forward only). Il résulte de l'exécution de la méthode **ExecuteReader** sur un objet Command.

L'objet DataReader ne stocke en mémoire qu'une seule ligne à la fois, permettant ainsi d'augmenter les performances d'une application et d'en réduire la charge.

Il est recommandé d'utiliser cet objet si :

- Vous n'avez pas besoin de réaliser un cache des données ;
- Vous traitez un jeu d'enregistrements trop importants pour être stocké en mémoire ;
- Vous souhaitez accéder à des données rapidement en lecture seule en avant seulement.

Exemple d'un objet DataReader :

MySqlDataReader reader = mysqlCom.ExecuteReader();

```
while (reader.Read())
                    {

                        int numero = (int)reader.GetValue(0);
                        string nom = (string)reader.GetValue(1);
                        string prenom = (string)reader.GetValue(2);
                        string login = (string)reader.GetValue(3);
                        double salaire = (double)reader.GetValue(4);

                        //Instanciation d'un Emplye
                e = new Employe(numero, nom, prenom, login,salaire);

                        // Ajout de cet employe à la liste
                        lc.Add(e);


                    }



                    reader.Close();
```