420-KBA-LG, programmation de bases de données

Saliha Yacoub

ADO.NET

- Rappels:
  - L'objet SqlConnection
    - Propriétés
    - méthoedes
  - L'objet SqlCommand
    - Propriétés
    - Méthodes
  - L'objet SqlDataReader
    - Propriétés
    - Méthodes.
- L'objet sqlParameter
  - Retour sur SqlCommand
  - Cas de paramètres en IN
  - Cas d'une fonction stockée.

ADO.NET

Introduction:

- Il est rare que l'exploitation des bases de données se fasse directement par l'interface client des SGBD (Sqldeveloper, SSMS, MySQL Workbench).
- Les développeurs conçoivent des applications afin de rendre l'accès au données facile et convivial.
- Il existe au moins deux solutions pour d'accéder aux données d'une bases de données :

1 - Une solution offerte par les SGBDs: les objets permettant l'accès aux données est du côté du SGBD (Exemple: Mysqli, Oracle Call Interface). Dans ce cas les programme C# qui va accéder à un BD SQL Server et à une BD Oracle sont très différents. Les solutions ne sont pas portables d'un SGBD à l'autre. Par contre les accès aux données sont très rapides.

ADO.NET

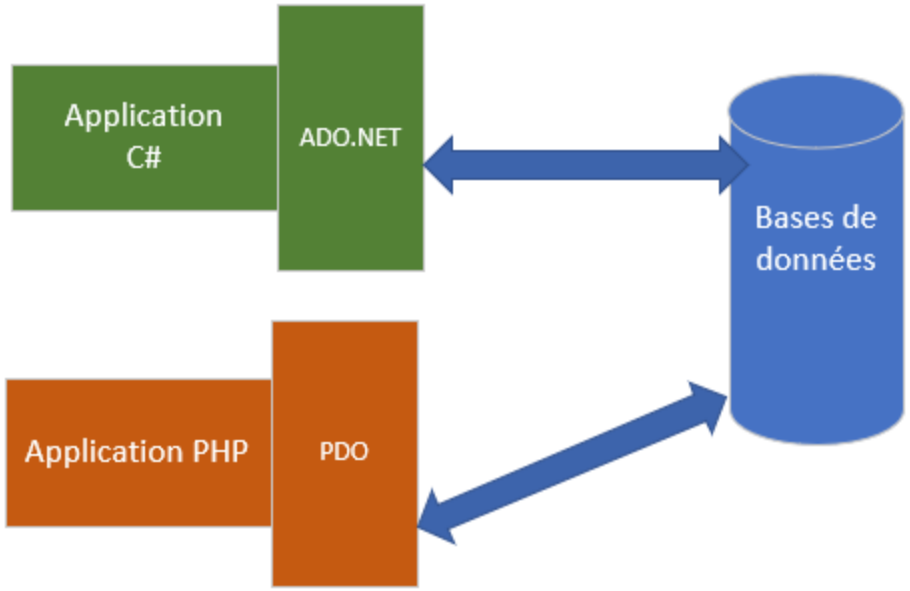
2- Une solution offerte par les langages de programmation. (ADO.NET,PDO,JDBC).

Dans ce cas , la solution est portable d'un SGBD à un autre. L'accès par C# à une BD SQL Server ou une BD MySQL est presque le même.

La figure ci-après montre les accès côté application à une BD.

Une application C#, par ADO.NET

Une application PHP par PDO.



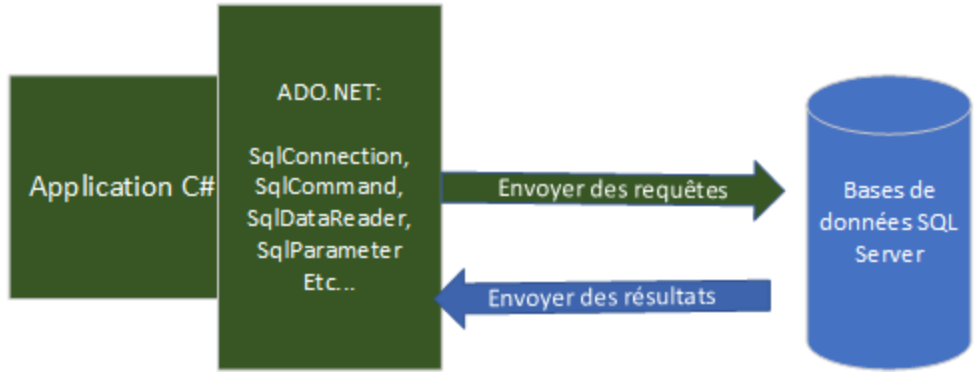
ADO.NET

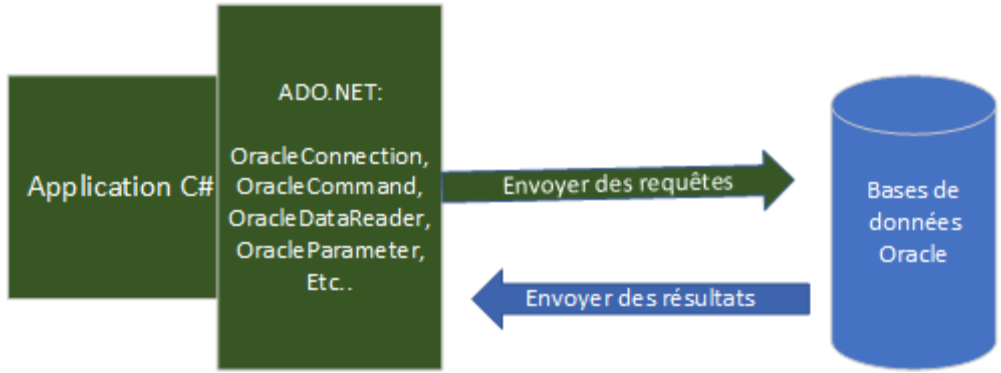
Définition

ADO.NET est un ensemble de classes qui exposent des services standardisés d'accès aux données.

Ces classes permettent donc aux programmeurs de concevoir des applications permettant de se connecter à des sources de données variées et, d'extraire, de manipuler et de mettre à jour ces données.

Dans une application ADO.NET, pour accéder à SQL server ou à Oracle, seul le nom des classes change. Les propriétés et les méthodes restent les même

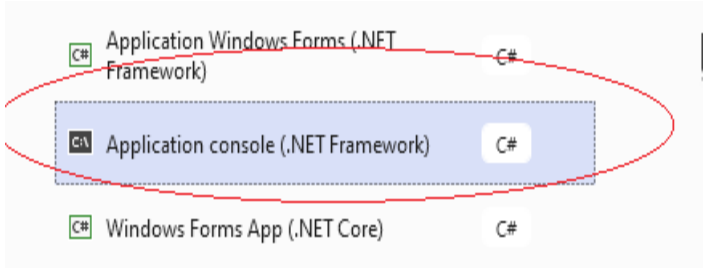




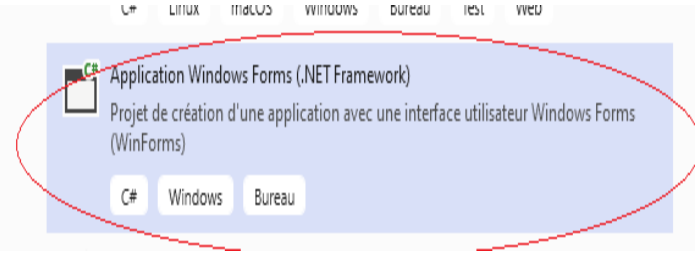
ADO.NET

Type d'application C#:

En mode consoles



En mode graphique



En tout temps, utiliser le
using System.Data.SqlClient

```
4   using System.Text;
5   using System.Threading.Tasks;
6   using System.Data.SqlClient;
7
```

ADO.NET

La classe SqlConnection: Définition

- Un objet SqlConnection représente une connexion à la base de données SQL Server. Il a donc pour rôle d'établir une connexion à la base de données.
- Les connexions sont utilisées pour « parler » aux bases de données et sont présentées par la classe SqlConnection.
- On crée un objet SqlConnection avec la méthode **new()**

```
private SqlConnection connSql = new SqlConnection();
```

Pour se connecter à la base de données, nous avons besoin:

- De fournir les information de connexion (nom du serveur, nom de la bd, nom du user, le mot de passe). Ces informations de connexion formeront une chaine de connexion.
- Ouvrir une connexion.

ADO.NET

La classe SqlConnection: Propriétés importantes:

ConnectionString

Il s'agit de la chaîne de connexion qui comprend des paramètres requis pour établir la connexion initiale. La valeur par défaut de cette propriété est une chaîne vide ("").

Les principaux paramètres pouvant être inclus dans la chaîne sont les suivants : (séparés par un point virgule)

Data source : le nom de serveur ou de la source de données;

Initial Catalog = nom de la base de données

User Id : Compte de connexion SQL Server ;

Password : Mot de passe de la session du compte SQL server

```
string chaineConnexion = "Data source =M-INFO-SY\\SQLEXPRESS; Initial Catalog =EmpClgDB; User Id= Patoche;password =123456";
```

ADO.NET

La classe SqlConnection: Propriétés importantes:

À ce stade nous avons un objet SqlConnection: connSql

Nous avons une chaine de connexion: chaineConexion.

La propriété ConnectionString va permettre de passer la chaine de connexion.

```
connSql.ConnectionString = ChaineConnexion;
```

ADO.NET

La classe SqlConnection: Propriétés importantes:

State:

Indique l'état actuel de de la connexion. Nous avons deux valeurs pour cette propriété. Closed ou Open Par défaut la valeur est Closed

```
connSql.State;
```

La classe SqlConnection: méthodes importantes:

Il existe deux méthodes importantes pour cette classe:

Open() → permet d'ouvrir une connexion

```
connSql.Open();
```

Close() → permet de fermer une connexion ouverte.

```
connSql.Close();
```

ADO.NET

```
namespace AdoSqlServer
{
   class Program
   {
      static void Main(string[] args)
      {
         SqlConnection connSql = new SqlConnection();
          try  {
              string chaineConnexion = "Data source =M-INFO-SY\\SQLEXPRESS;  Initial Catalog =EmpClgDB; User Id= Patoche;password =2002";
             connSql.ConnectionString  = chaineConnexion;
              connSql.Open();
              Console.WriteLine("statut de la connexion"  + " " + connSql.State);
                }
         catch (Exception ex)
              { Console.WriteLine(ex.Message) }
      connSql.Close();
       Console.WriteLine("statut de la connexion" + " " + connSql.State);
    }
   }
```

ADO.NET

La classe SqlCommand: Propriétés importantes:

L'objet SqlCommand contient les commandes envoyées aux SGBD. Ces commandes sont envoyées soit en utilisant des requêtes simples, soit en utilisant des procédures stockées. Lorsque la requête SQL ou procédure retourne un résultat, il est retourné dans un SqlDataReader ou autre (voir **plus loin).**

SqlCommand possède plusieurs constructeurs, pour commencer, celui que nous allons utiliser est le constructeur suivant :

Un objet SqlCommand se crée avec la méthode new();

```
SqlCommand(string SQL , SqlConnection connSql)
```

```
SqlCommand sqlComd = new SqlCommand(sql, connSql);
```

sql est une requête SQL

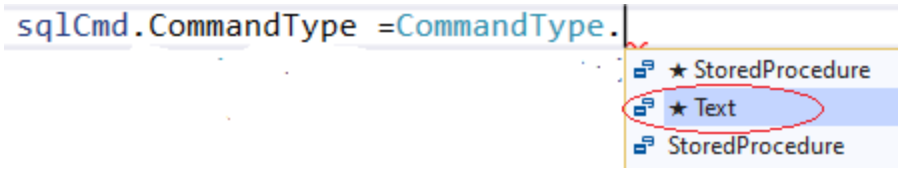
connSql est un objet SqlConnection

ADO.NET

La classe SqlCommand: Propriétés importantes:

Propriétés	Signification
CommandText	Obtient ou définit l'instruction SQL ou la procédure stockée à exécuter sur la base de données
CommandType	Obtient ou définit une valeur indiquant la manière dont la propriété CommandText doit être interprétée (instruction SQL ou procédure stockée).
Connection	Obtient ou définit l'objet SqlConnection utilisé par cette instance de SqlCommand.
Parameters	Spécifie les paramètres de la requête SQL ou de la procédure stockée

› Par défaut la propriété CommandType a la valeur Text (pour une requête SQL qui n'est pas une procédure



ADO.NET

La classe SqlCommand: méthodes importantes

Méthodes	Significations
ExecuteNonQuery()	Exécute une instruction SQL sur Connection et retourne le nombre de lignes affectées. Pour toutes les requêtes DML
ExecuteReader()	Surchargé. Envoie CommandText à Connection et génère SqlDataReader. Pour les requêtes SELECT
ExecuteScalar()	Exécute la requête et retourne la première colonne de la première ligne. (en général pour les SELECT COUNT, SELECT MIN …

- **La méthode ExecuteNonQuery():** Cette méthode permet d'exécuter une requête DML (INSERT, UPDATE et DELETE). Cette méthode retourne un **int**, indiquant le nombre de lignes affectées par la requête.
- **La méthode ExecuteReader():** Cette méthode permet d'exécuter une requête SELECT et retourne un **SqlDataReadr** contenant le résultat de la requête.
- **La méthode ExecuteScalar():** Exécute la requête et retourne la première colonne de la première ligne. Elle est utilisée pour des requêtes comme SELECT COUNT, SELECT MIN etc…

ADO.NET

nous sommes déjà connectés: Le nom de l'objet SqlConnection est: connSql

Exemple1: Cas d'une instruction DML (INSERT)

```
string sql = "insert into employesClg(nom, prenom, typeEmploye) values('ADO', 'NET','P')";
SqlCommand sqlCommd = new SqlCommand(sql, connSql);
int nb_lignes = sqlCommd.ExecuteNonQuery();
Console.WriteLine("nombre de ligne" + nb_lignes);
Console.Read();
```

Exemple 2: Cas d'un SELECT qui retourne UNE SEULE valeur (ici un INT)

```
string sql3 = "select count(*) from employesClg";
SqlCommand sqlCommdCount = new SqlCommand(sql3, connSql);
int totalEmp = (int)sqlCommdCount.ExecuteScalar();
Console.WriteLine(totalEmp);
Console.Read();
```

ADO.NET

La classe SqlDataReader: définition :

un objet **DataReader** sert à extraire d'une base de données un flux de données en lecture seule et dont le défilement se fera par en avant uniquement (read-only, forward-only,). Le contenu d'un SqlDataReader correspond à une requête SELECT (plusieurs lignes, plusieurs colonnes. En ce sens c'est comme un curseur).

L'objet SqlDataReader NE se crée PAS avec la Méthone new()

L'objet SqlDataReader se crée avec la méthode ExecuteReader() de l'objet SqlCommand

```
SqlCommand sqlComd = new SqlCommand(sql, connSql);
SqlDataReader sqlRead = sqlComd.ExecuteReader();
```

Par défaut, un **DataReader** charge une ligne entière en mémoire à chaque appel de la méthode **Read()**. Il est possible d'accéder aux valeurs de colonnes soit par leurs noms soit par leurs références ordinales.

Une solution plus performante est proposée permettant d'accéder aux valeurs dans leurs types de données natifs (**GetInt64**, **GetDouble**, **GetString**). Par exemple si la première colonne de la ligne indicée par 0 est de type **int**, alors il est possible de la récupérer à l'aide de la méthode **GetInt64(0)** de l'objet **DataReader.**

ADO.NET

La classe SqlDataReader: méthodes importantes

Méthodes	Signification
Colse()	Ferme l'objet SqlDataReader
Dispose ()	Libère toutes les ressources occupées par l'objet SqlDataReader
Read ()	Permet d'avancer l'objet SqlDataReader jusqu'à l'enregistrement suivant.
GetDateTime(indicolonne)	Obtient la valeur de la colonne spécifiée sous la forme d'un objet DateTime.
GetDecimal(indicolonne)	Obtient la valeur de la colonne spécifiée sous la forme d'un objet Decimal.
GetDouble(indicolonne)	Obtient la valeur de la colonne spécifiée sous la forme d'un nombre de type Double.
GetFloat(indicolonne)	Obtient la valeur de la colonne spécifiée sous la forme d'un nombre de type Float.
GetInt16(indicolonne)	Obtient la valeur de la colonne spécifiée sous la forme d'un entier signé 16 bits.
GetInt32(indicolonne)	Obtient la valeur de la colonne spécifiée sous la forme d'un entier signé 32 bits.
GetInt64(indicolonne)	Obtient la valeur de la colonne spécifiée sous la forme d'un entier signé 64 bits.
GetLifetimeService()	Extrait l'objet de service de durée de vie en cours qui contrôle la stratégie de durée de vie de cette instance.
GetName(indicolonne)	Obtient le nom de la colonne spécifiée.
GetString(indicolonne)	Obtient la valeur de la colonne spécifiée sous la forme d'une chaîne.

ADO.NET

La classe SqlDataReader: Exemple

```
string sql2 = "select  nom, prenom from employesClg where typeEmploye = ‘P' ";
SqlCommand sqlCommdSelect = new SqlCommand(sql2, connSql);
 SqlDataReader sqlRead = sqlCommdSelect.ExecuteReader();
          while (sqlRead.Read())
              {
             Console.WriteLine("{0} - {1}",
             sqlRead.GetString(0),sqlRead.GetString(1));
              }
  sqlRead.Close();
  sqlRead.Dispose();
```

ADO.NET

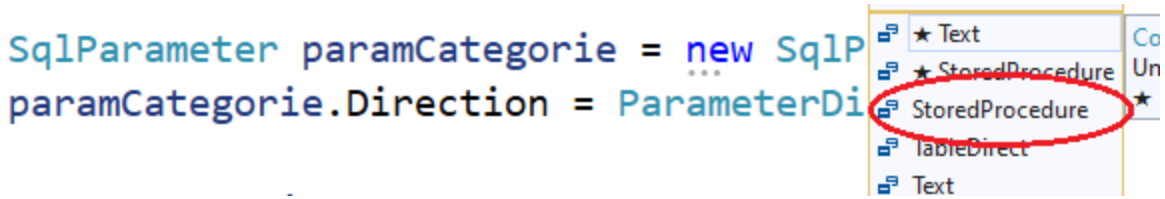
- SqlParameter
  - Retour sur SqlCommand
  - Cas de paramètres en IN
  - Cas d’une fonctions stockée.

ADO.NET

La classe SqlCommand: Propriétés importantes:

Propriétés	Signification
CommandText	Obtient ou définit l'instruction SQL ou la procédure stockée à exécuter sur la base de données
CommandType	Obtient ou définit une valeur indiquant la manière dont la propriété CommandText doit être interprétée (instruction SQL ou procédure stockée).
Connection	Obtient ou définit l'objet SqlConnection utilisé par cette instance de SqlCommand.
Parameters	**Spécifie les paramètres de la requête SQL ou de la procédure stockée**

› Par défaut la propriété CommandType a la valeur Text (pour une requête SQL qui n'est pas une procédure



ADO.NET

La classe SqlParameter: Constructeurs:

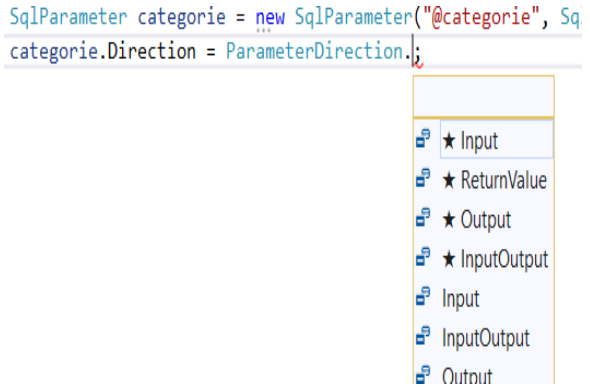
L'objet SqlParameter représente un paramètre pour l'SqlCommand ou une colonne du DataSet

SqlParameter()	Instancie un SqlParameter
SqlParameter (string, SqlDbType)	String désigne le nom du paramètre, le SqlDbtype désigne le type de données du Pramètre (un SqlType.) : public SqlParameter(string parameterName, SqlDbType)
SqlParameter(string, SqlDbType, int)	Même que le précédent, sauf qu'on indique la taille du paramètre
SqlParameter(string, SqlDbType, int, string)	Même que le précédent, sauf qu'on indique la taille du paramètre et le nom de la colonne source : à voir avec le DataSet
SqlParameter(string,SqlDbType,ParameterDirection)	Le ParameterDirection indique si le paramètre est en IN ou Out. Utilisé lorsque nous avons une procédure stockée

ADO.NET

Direction du paramètre:

Paramètre de la procédure	Direction du SqlParameter
IN(par défaut)	Input (par défaut)
OUT	Output
Une fonction	ReturnValue



ADO.NET

La classe SqlParameter: Propriétés importantes

Direction	Obtient ou définit une valeur qui indique si le paramètre est un paramètre d'entrée uniquement, de sortie uniquement, bidirectionnel ou de valeur de retour d'une procédure stockée.
ParameterName	Obtient ou définit le nom de SqlParameter
SqlDbType	Spécifie le type de données Sql
Size	Obtient ou définit la taille maximale, en octets, des données figurant dans la colonne.
Value	Obtient ou définit la valeur du paramètre

ADO.NET

- Comment passer une procédure et ses paramètres ?
  - Étape 0: Le SqlCommad va permettre d'indiquer clairement qu'il s'agit d'une procédure par ses propriétés CommandText et CommandType:

```
SqlCommand cmdInsert = new SqlCommand("ajouterJoueur", sqlconn);
cmdInsert.CommandText = "ajouterJoueur";
cmdInsert.CommandType = CommandType.StoredProcedure;
```

ajouterJoueur est le nom de la procédure stockée

ADO.NET

- Comment passer une procédure et ses paramètres ?
  - Étape 1: définir un SqlParameter pour chaque paramètre de la procédure

```
SqlParameter parmNom = new SqlParameter("@nom", SqlDbType.VarChar, 30);
parmNom.Direction = ParameterDirection.Input;

SqlParameter parmPrenom = new SqlParameter("@prenom", SqlDbType.VarChar, 30);
parmPrenom.Direction = ParameterDirection.Input;

SqlParameter parmSalaire = new SqlParameter("@salaire", SqlDbType.Decimal);
parmSalaire.Direction = ParameterDirection.Input;

SqlParameter parmEquipe = new SqlParameter("@codeEquipe", SqlDbType.Char, 3);
parmEquipe.Direction = ParameterDirection.Input;
```

La procédure ajouterJoueur a 4 paramètres. Ils sont tous en IN

ADO.NET

- Comment passer une procédure et ses paramètres ?
  - Étape 2: donner des valeurs aux paramètres en utilisant le propriété Value de SqlParameter

```
parmNom.Value = 'Patoche';
parmPrenom.Value = 'Alain';
parmSalaire.Value = 100000;
parmEquipe.Value = 'MTL';
```

Les valeurs des paramètres peuvent être fournis par des zones de texte.

ADO.NET

- Comment passer une procédure et ses paramètres ?
  - Étape 3: Ajouter les paramètres avec leurs valeurs à SqlCommand avec la propriété Parameters de SqlCommand.

```
cmdInsert.Parameters.Add(parmNom);
cmdInsert.Parameters.Add(parmPrenom);
cmdInsert.Parameters.Add(parmSalaire);
cmdInsert.Parameters.Add(parmEquipe);
```

Les valeurs des paramètres peuvent être fournis par des zones de texte.

ADO.NET

- Comment passer une procédure et ses paramètres ?
  - Étape 4: Exécuter la procédure Stockée. Il faudra utiliser la méthode appropriée de SqlCommand.

```
cmdInsert.ExecuteNonQuery();
```

ajouterJoueur est le nom de la procédure stockée qui fait une insertion . Donc la méthode appropriée de sqlCommnad est ExecuteNonQuery();

ADO.NET

Exemple de fonction sans paramètre qui retourne une valeur:

```
private void compter()
    {
          SqlCommand sqlcmdTotal = new SqlCommand("totalJoueur", sqlconn);
         sqlcmdTotal l.CommandText = "totalJoueur";
         sqlcmdTotal.CommandType = CommandType.StoredProcedure;
//On déclare le paramètre de retour
         SqlParameter resultat = new SqlParameter("@totaljoueur", SqlDbType.Int);
         resultat.Direction = ParameterDirection.ReturnValue;
//ajouter les paramètres au SqlCommand
         sqlcmdTotal .Parameters.Add(resultat);
//On exécute
        sqlcmdTotal.ExecuteScalar();
        total.Text = resultat.Value.ToString();
    }
```

ADO.NET

Exemple de fonction avec paramètre qui retourne une valeur:

```
private void nbJoueursEquipe() {
        SqlCommand cmdTotal = new SqlCommand("totalJoueurEquipe", sqlconn);
        cmdTotal.CommandText = "totalJoueurEquipe";
        cmdTotal.CommandType = CommandType.StoredProcedure;
//Paramètre en IN
        SqlParameter paramEquipe = new SqlParameter("@nomEquipe", SqlDbType.VarChar, 30);
        paramEquipe.Direction = ParameterDirection.Input;
        paramEquipe.Value = nomEquip; //'Canadiens'
//paramètre de retour
        SqlParameter resultat = new SqlParameter("@totaEtudiant", SqlDbType.Int);
        resultat.Direction = ParameterDirection.ReturnValue;
//ajouter les paramètres au SqlCommand
        cmdTotal.Parameters.Add(resultat);
        cmdTotal.Parameters.Add(paramEquipe);
//On execute
        cmdTotal.ExecuteScalar();
        totalParEquipe.Text = resultat.Value.ToString();
```

ADO.NET

- Cas d'une fonction table:

Le cas d'une fonction table est traité comme une requête SQL SELECT quelconque.

Pour exécuter une fonction table dans Managment Studio on fait : select * from nomfonction().

CONCLUSION

QUESTIONS ??