Programmation Orientée Objet C#:

M.WANE
Ingénieur Informaticien
Analyste, Concepteur
Développeur d'Applications

Email : douvewane85@gmail.com

Programme

- **ADO.NET : Accès aux bases de données avec C#**
  - Mode déconnecté
  - DataSet, DataAdapter
  - Gestion des transactions
  - Applications WindowsForms
  - - Utilisation des assistants Visual Studio

❑ ADO.NET : Accès aux bases de données avec C#

❖ Présentation

- ADO.NET est un ensemble de classes qui exposent les services d'accès aux données pour les programmeurs .NET Framework. ADO.NET propose un large ensemble de composants pour la création d'applications distribuées avec partage de données.

Partie intégrante du .NET Framework, il permet d'accéder à des données:

- **relationnelles,**
- **XML**
- **d'application. ADO.NET**

- ADO.NET répond à divers besoins en matière de développement, en permettant notamment:
  - **de créer des clients de bases de données frontaux**
  - **des objets métier de couche intermédiaire utilisés par**
    - **des applications,**
    - **outils, langages**
    - **ou navigateurs Internet**.
- ADO.NET sépare l'accès aux données de leur manipulation en composants distincts qui peuvent être utilisés individuellement ou en tandem. ADO.NET comprend des fournisseurs de données .NET Framework pour la connexion à une base de données, l'exécution de commandes et l'extraction de résultats. fournisseurs de données .NET Framework les utilisés sont:
  - **System.Data.SqlClient** Pour les sources de données SQL SERVER
  - **System.Data.OleDb** Pour les sources de données exposées à l'aide de OLE DB.
  - **System.Data.Odbc** Pour les sources de données exposées à l'aide de ODBC.
  - Pour les sources de données Oracle et utilise l'espace de noms **System.Data.OracleClient**.
  - Pour les sources de données Mysql et utilise l'espace de noms **System.Data.MysqlClient**.
  - Fournit un accès aux données pour les applications EDM (Entity Data Model) et utilise l'espace de noms **System.Data.EntityClient**.

➢ Objets principaux des fournisseurs de données .NET Framework

object	Description
Connection	Établit une connexion à une source de données spécifique. La classe de base pour tous les objets **Connection** est la classe DbConnection.
Command	Exécute une commande sur une source de données. Expose **Parameters** et peut exécuter dans la portée d'une **Transaction** à partir d'un **Connection**. La classe de base pour tous les objets **Command** est la classe DbCommand.
DataReader	Lit un flux de données avant uniquement (forward only) et en lecture seule à partir d'une source de données. La classe de base pour tous les objets **DataReader** est la classe DbDataReader.
DataAdapter	Remplit un **DataSet** et répercute les mises à jour dans la source de données. La classe de base pour tous les objets **DataAdapter** est la classe DbDataAdapter.

- Un objet **Connection** est utilisé conjointement à la classe DataAdapter .

NB : Dans ce cours nous utiliserons le SGBD SQLSERVER donc :

- **L'espace de Nom sera System.Data.SqlClient**
- L' objet **Connection** va correspondre à **SqlConnection**
- L' objet **Command** va correspondre à **SqlCommand**
- L' objet **DataAdapter** va correspondre à **SqlDataAdapter**
- L' objet **DataReader** va correspondre à **SqlDataReader**

➢ Méthode d'utilisation de ADO.NET en Mode Connecté

1. **Importation des Namespace:**
   - **using System.Data;**
   - **using System.Data.SqlClient;**

2. **Etablir une Connection a la Source de Donnée**

   A) Récupérer la Chaine de Connexion

   String ChaineConnection = «

   "Data Source= **seveu**r \SQLEXPRESS; => **Instance**

   Initial Catalog= **NomBaseDonnee**; => **Base de Donnée**

   Integrated Security= True " => **Type d'authentification par defaut Windows**

   B) Créer un objet de Type SqlConnection

   SqlConnection Conn = new SqlConnection(ChaineConnection);

   C) Ouvrir la Connexion

   Conn.open();

3. **Extraire les données de la source de données.**

   A) Ecrire la requête avec
   - **string.Format**
   - **Requete paramétrée et le paramètre est représenté par @nomParametre**

   B) Créer Un Objet SqlCommand

   SqlCommand cmd = new SqlCommand("requete", Conn);

   NB : Si la Requete paramétrée donc il faut remplacer les valeurs par leur paramètre dans la requete. On créera un Objet de la Classe SqlParameter comme suit:

   SqlParameter param=new SqlParameter("@nomParametre",valeur);

   Ajouter le Paramètre cmd.Parameters.Add(param);

   C) Exécution de la requete

   Requete Select: SqlDataReader dr = cmd.ExecuteReader();

   Requete Mis a Jour : int nbreLigneAffecte=cmd. cmd.ExecuteNonQuery();

   D) Parcourir le SqlDataReader avec la méthode dr.Read().A la fin fermer le SqlDataReader avec dr.Close().

4. **Fermer la Connection avec Conn.close();**

Remarque

On peut exécuter les requetes a l'aide des transaction .

1) Création d'un Objet de Type **SqlTransaction**

```
SqlTransaction tx = conn.BeginTransaction();
```

2) Ajouter la Transaction dans la Commande

```
cmd.Transaction = tx;
```

En cas de Succès on fait un **tx.Commit()**

En cas de d' échec on fait un **tx.Rollback();**

➢ Mode déconnecté

▪ Mode déconnecté :

- L'objet DataSet ADO.NET est une **représentation de données résidente en mémoire** qui propose un modèle de programmation relationnel cohérent, quelle que soit la source des données qu'il contient .
- Un objet **DataSet** représente un jeu de données complet, y compris les tables qui contiennent et organisent les données et y appliquent des contraintes, ainsi que les relations entre les tables.
- L'utilisation d'un objet **DataSet** peut se faire via différentes méthodes qui peuvent être appliquées indépendamment les unes des autres ou combinées. Plusieurs possibilités s'offrent à vous :

  – Créer par programmation un objet
    - » **DataTable,**
    - » **DataRelation**
    - » **Constraint**

    dans un objet **DataSet**, puis remplir les tables de données.

  – Remplir l'objet **DataSet** de tables de données provenant d'une source de données relationnelles existante à l'aide d'un objet **DataAdapter**.

  – Charger et rendre persistent le contenu de l'objet **DataSet** à l'aide de XML.

- Un objet **DataSet** fortement typé peut aussi être transporté au moyen d'un service Web XML. Le design de l'objet **DataSet** le rend idéal pour le transport de données à l'aide des services Web XML. Pour une vue d'ensemble des services Web XML, voir Vue d'ensemble des services Web XML.

➢ DataSet, DataAdapter

▪ DataSet

- ✓ L'objet **DataSet** est une représentation relationnelle de données résidente en mémoire, indépendante de toute source de données.
- ✓ Toutefois, le **DataSet** peut être utilisé avec les données existantes d'une source de données par l'intermédiaire d'un **fournisseur de données .NET Framework**.
- ✓ Un fournisseur de données .NET Framework utilise un **DataAdapter** pour remplir le DataSet de données et d'informations de schéma, ainsi que pour répercuter dans la source de données les modifications apportées aux données.

▪ DataAdapter

- ✓ Le DataAdapter contrôle l'interaction entre **DataSet** avec les sources de données existantes
- ✓ **La propriété SelectCommand** du **DataAdapter** est un objet **Command** qui extrait les données de la source de données.
- ✓ **Les propriétés InsertCommand, UpdateCommand et DeleteCommand** du **DataAdapter** sont des objets **Command** qui gèrent les mises à jour dans la source de données conformément aux modifications faites dans le **DataSet.**
- ✓ **La méthode Fill** du **DataAdapter** est utilisée pour remplir un **DataSet** avec les résultats de **SelectCommand** du **DataAdapter**. **Fill** prend comme arguments un **DataSet** à remplir et un objet **DataTable** ou le nom du **DataTable** à remplir avec les lignes retournées par **SelectCommand**.

➢ Méthode d'utilisation de ADO.NET en Mode Déconnecté

1. **Importation des Namespace:**
   - **using System.Data;**
   - **using System.Data.SqlClient;**

2. **Etablir une Connection a la Source de Donnée**

   A) Récupérer la Chaine de Connexion

```
String ChaineConnection = «
        "Data Source= seveur \SQLEXPRESS;    => Instance
         Initial Catalog= NomBaseDonnee;     => Base de Donnée
         Integrated Security= True "         => Type d'authentification par defaut Windows
```

   B) Créer un objet de Type IDbConnection

```
IDbConnection Conn = new SqlConnection( ChaineConnection);
```

   C) Ouvrir la Connexion

```
Conn.open();
```

3. **Extraire les données de la source de données.**

   A) Ecrire la requête avec
   - **string.Format**
   - **Requete paramétrée et le paramètre est représenté par @nomParametre**

   B) Créer Un Objet IDbCommand

```
IDbCommand cmd = conn.CreateCommand();
```

   C) Passer la Requete a l'objet IDbCommand

```
cmd.CommandText = Requete;
```

   D) Préciser le Type de la Requete
   - **cmd.CommandType = CommandType.Text; Dans le cas d'une Requete SQL**
   - **cmd.CommandType = CommandType.StoredProceduret; Dans le cas d'une Procédure stockée**

NB : Si la Requete paramétrée donc il faut remplacer les valeurs par leur paramètre dans la requete. On créera un Objet de la Classe SqlParameter comme suit:

```
SqlParameter param=new SqlParameter("@nomParametre",valeur);
```

Ajouter le Paramètre `cmd.Parameters.Add(param);`

E) Création du DataSet

- **DataSet ds = new DataSet();//**Contenir les données en Mémoire
- **IDbDataAdapter da = new SqlDataAdapter();** //Mappage entre la Base de Donnée et le DataSet

F) Exécuter la Requete

da.Fill(ds);//Remplir le DataSet

G)Parcourir le DataSet

```
foreach (DataColumn dc in ds.Tables[0].Columns)
        {
           Console.Write(dc.ColumnName+"\t");
        }
        Console.WriteLine("");
        foreach (DataRow dr in ds.Tables[0].Rows)
        {
           foreach (DataColumn dc in ds.Tables[0].Columns)
           {
              Console.Write(dr[dc.ColumnName] + "\t");
           }
           Console.WriteLine("");
        }
```

4. Fermer la Connection avec Conn.close();