ADO .NET and Entity Framework C#

Mme Ben bouzid NOURHENE

ADO .NET

Introduction ADO.NET

- ADO.NET : ActiveX Data Objects
- Un ensemble de classes qui exposent les services d'accès aux données pour les programmeurs.
- ADO.NET propose un large ensemble de composants pour la création d'applications distribuées avec partage de données.
- Des applications grand public de partage de données peuvent utiliser ADO.NET pour se connecter à des sources de données et extraire, manipuler et mettre à jour les données qu'elles contiennent.
- 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.

Modes de fonctionnement ADO.NET

- **Mode connecté :**

  le programme se connecte à la base de données et effectue des opérations comme SELECT, INSERT, UPDATE... La connexion est ensuite fermée et aucune donnée n'est donc stockée en mémoire sur le client.

- **Mode déconnecté :**

  le programme se connecte à la base de données, récupère les données et les stocker en mémoire et referme immédiatement la connexion. Les opérations sur les données (affichage...) ne se font qu'une fois la connexion fermée.

Les Managed Providers

- Pour pouvoir faire appel aux classes proposées par ADO.Net il est nécessaire d'inclure une référence à l'espace de noms correspondant.
- Quelque exemples:

Espace de nom	Description
System.Data.SqlClient	Fournisseur de données spécifiques pour SQL Server
System.Data.Odbc	Propose la gestion de sources de données accédées via un driver Odbc
System.Data.OracleClient	Propose un accès à des sources de données Oracle

Etapes:

- Les étapes pour faire le CRUD
  - Construire la chaine de connexion
  - Etablir la connexion
  - Préparer la commande
  - Exécuter la commande et récupérer le résultat

Quelques classes de ADO.NET

- **Connection :** permet d'assurer la connectivité avec une source de données
- **Command :** permet de définir des requêtes SQL afin de lire ou écrire des données, ou d'éxecuter des procédures stockées...
- **DataReader :** fournit un flux très performant de données en provenance de la source de données.
- **DataAdapter :** utilise la requête définie dans un objet Command afin de charger le DataSet avec des données.
- **DataSet :** est un tableau conçu pour accueillir des données venant d'une source de données relationnelle.

DataSet vs DataReader

- **DataReader :**

  fonctionne en mode connecté / plus rapide

- **DataSet :**

  fonctionne en mode déconnecté.

  explicitement conçu pour un accès aux données

  indépendamment de toute source de données (XML, Base de données...)

ADO.NET

```
CREATE DATABASE StudentDB;
GO

USE StudentDB;
GO

CREATE TABLE Student(
 Id INT PRIMARY KEY,
 Name VARCHAR(100),
 Email VARCHAR(50),
 Mobile VARCHAR(50)
)
GO

INSERT INTO Student VALUES (101, 'Anurag', 'Anurag@dotnettutorial.net', '1234567890')
INSERT INTO Student VALUES (102, 'Priyanka', 'Priyanka@dotnettutorial.net',
'2233445566')
INSERT INTO Student VALUES (103, 'Preety', 'Preety@dotnettutorial.net', '6655443322')
INSERT INTO Student VALUES (104, 'Sambit', 'Sambit@dotnettutorial.net', '9876543210')
GO
```

ADO.NET

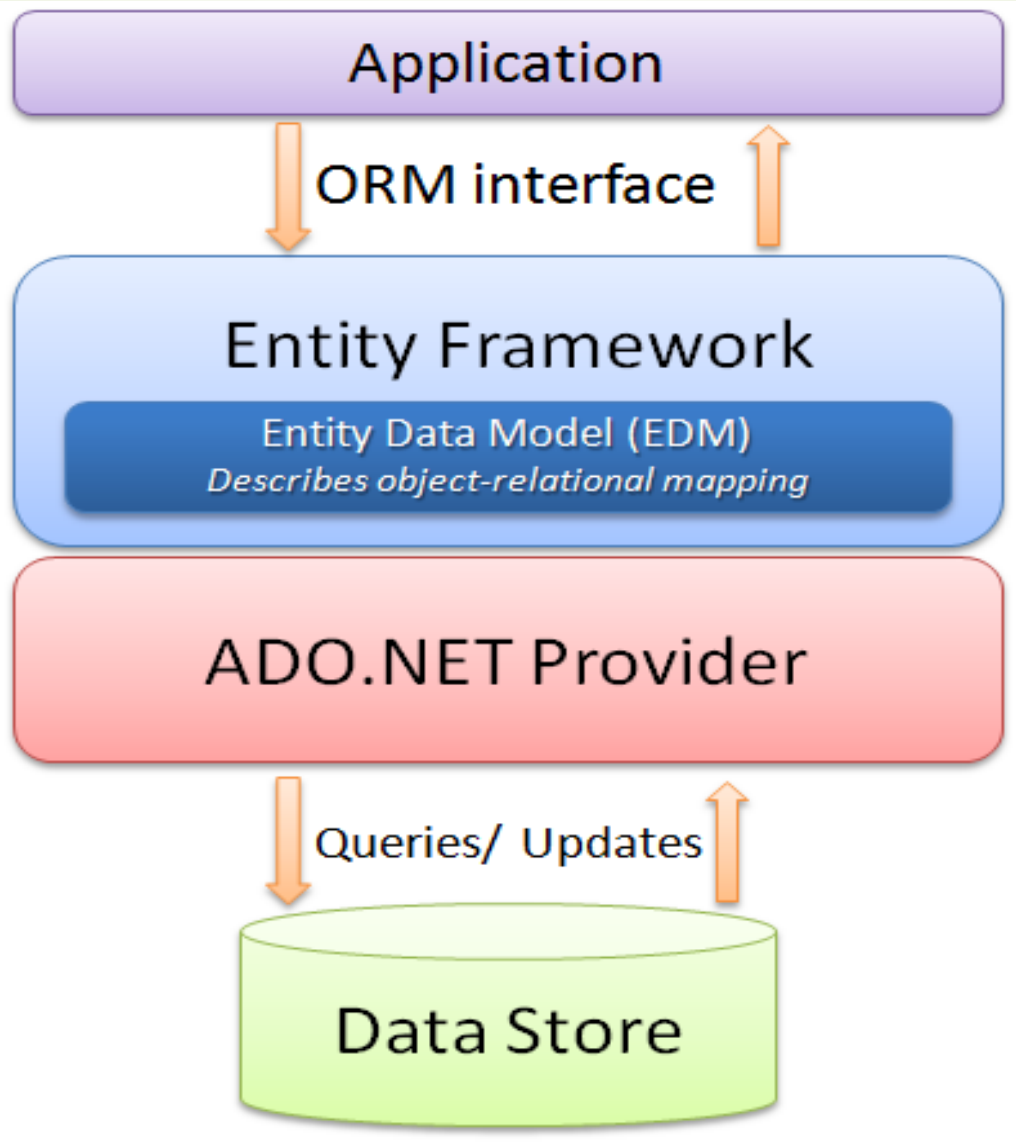
```
using System;
using System.Data.SqlClient;
namespace AdoNetConsoleApplication
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                string ConString = "data source=.; database=StudentDB; integrated
security=SSPI";
                using (SqlConnection connection = new SqlConnection(ConString))
                {
                    // Creating SqlCommand objcet
                    SqlCommand cm = new SqlCommand("select * from student", connection);

                    // Opening Connection
                    connection.Open();

                    // Executing the SQL query
                    SqlDataReader sdr = cm.ExecuteReader();
                    while (sdr.Read())
                    {
                        Console.WriteLine(sdr["Name"] + ",  " + sdr["Email"] + ",  " +
sdr["Mobile"]);
                    }
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("OOPs, something went wrong.\n" + e);
            }
            Console.ReadKey();
        }
    }
```

Entity Framework

Introduction EF:

- c’est le nom de l’ORM officiel de .NET. Développé par Microsoft.
- le Entity Framework est un ensemble de technologies ADO.NET qui prennent en charge le développement d’applications logicielles orientées données.
- Avec Entity Framework, les développeurs peuvent travailler à un niveau supérieur d’abstraction lorsqu’ils traitent les données, et peuvent créer et maintenir des applications orientées données avec moins de code que dans les applications traditionnelles.

Introduction EF:



Les différents approches de Entity Framework:

- **Database First:**

une DB existe, on souhaite l'utiliser pour générer un modèle de données.

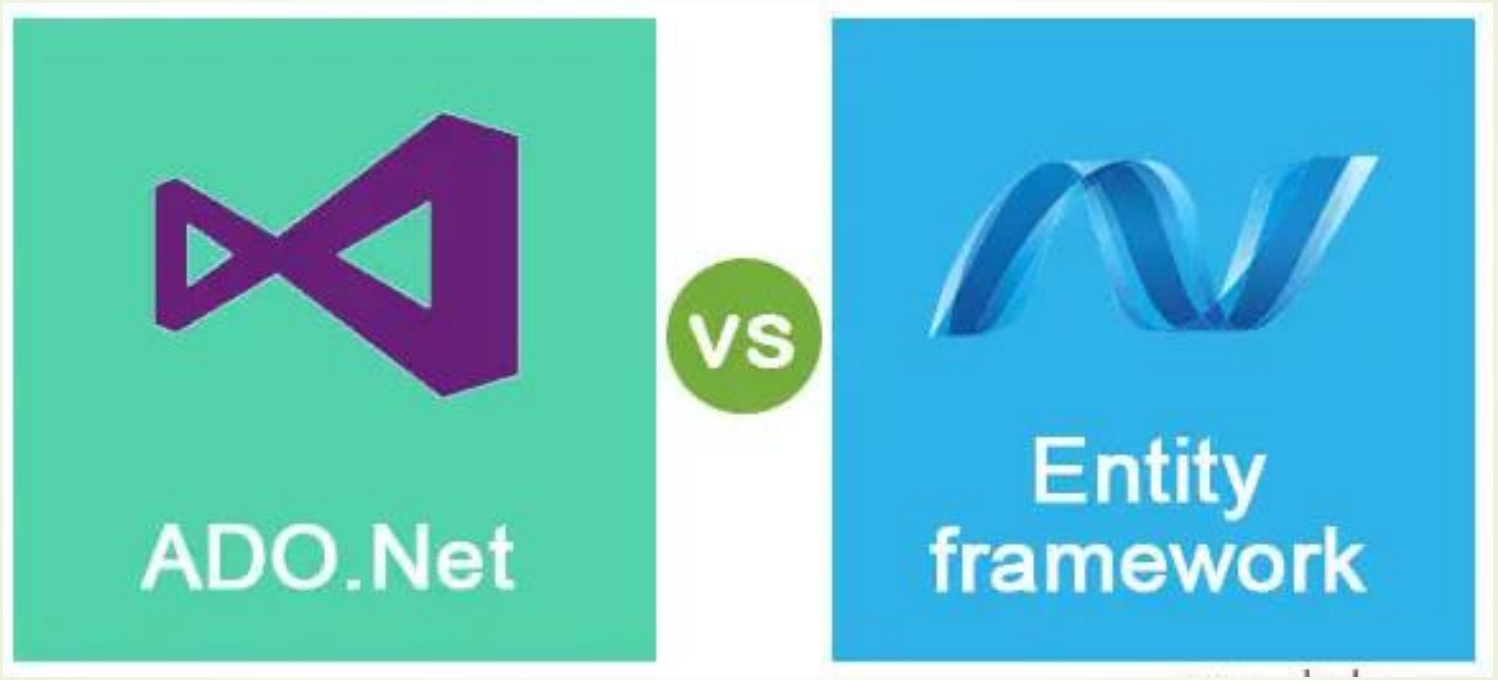
- **Model First:**

aucune DB n'existe, on souhaite en générer une depuis un modèle de données.

- **Code First:**

aucune DB n'existe, on souhaite en générer une depuis du code C# (Domain classes)

ADO.NET **VS** Entity Framework:



ADO.NET **VS** Entity Framework:

- Performance:

  ADO.NET offre de meilleures performances car il est directement connecté à la source de données, ce qui rend le traitement plus rapide qu'Entity Framework car il traduit d'abord les requêtes LINQ en SQL, puis traite la requête.

- Vitesse de développement

  Entity Framework génère automatiquement des modèles et des classes de contexte de base de données pour gérer les opérations de base de données et nécessite donc moins d'efforts et de temps.

ADO.NET **VS** Entity Framework:

- Maintenabilité de code :

  Dans ADO.NET, le débogage est fastidieux lorsque vous passez de la couche application à la couche base de données une par une pour trouver ce qui se passe dans une application alors que Entity Framework donne des relations clairement modélisées d'entités et de niveaux dépendants.

- Flexibilité

  ADO.NET offre une grande flexibilité en termes de requêtes et de procédures SQL brutes par rapport à Entity Framework en offrant un contrôle total sur une base de données.

Merci pour votre attention