

Résolution du problème de **BANKINGSYSTEM**

Nous utilisons une version classique de **.net core (4.8)** or **Microsoft.EntityFrameworkCore** fonctionne à partir de la version 5 et plus de dotnet. **Dotnet framework (4.8)** lui fonctionne avec le **System.Data.Entity** et la configuration de son instance **SQLSERVER** se fait dans le fichier **web.config**

Ce Document de quelques pages que j'ai rédigé sert de roadmap pour pouvoir effectuer une connection à l'instance SQLSERVER via EntityFrameWork 6.5.1 qui fonctionne bien avec le .NET 4.8.

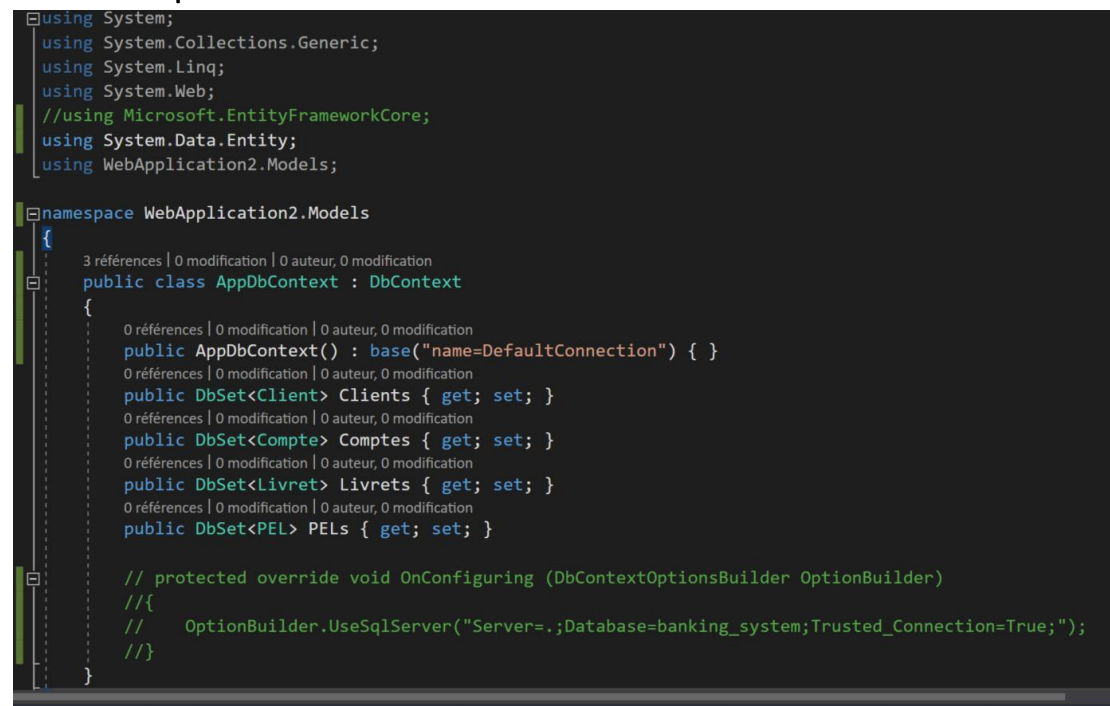
Objectif

- Connecter le projet **.NET 4.8** à l'instance SQLSERVER Via **EntityFramework 6.5.1**

Etapes

- Mettre en place le fichier context
- Configurer la chaîne de connection
- Installer Entity Framework via Nuget
- Activer la migration Code First
- Créer la migration initial
- Mettre à jour la base de données

A. Mettre en place le fichier context



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
//using Microsoft.EntityFrameworkCore;
using System.Data.Entity;
using WebApplication2.Models;

namespace WebApplication2.Models
{
    3 références | 0 modification | 0 auteur, 0 modification
    public class AppDbContext : DbContext
    {
        0 références | 0 modification | 0 auteur, 0 modification
        public AppDbContext() : base("name=DefaultConnection") { }
        0 références | 0 modification | 0 auteur, 0 modification
        public DbSet<Client> Clients { get; set; }
        0 références | 0 modification | 0 auteur, 0 modification
        public DbSet<Compte> Comptes { get; set; }
        0 références | 0 modification | 0 auteur, 0 modification
        public DbSet<Livret> Livrets { get; set; }
        0 références | 0 modification | 0 auteur, 0 modification
        public DbSet<PEL> PELs { get; set; }

        // protected override void OnConfiguring (DbContextOptionsBuilder OptionBuilder)
        //{
        //    OptionBuilder.UseSqlServer("Server=.;Database=banking_system;Trusted_Connection=True;");
        //}
    }
}
```

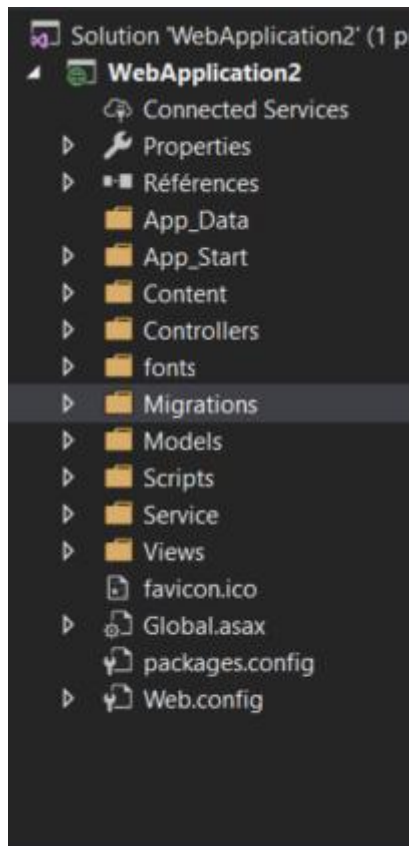
Vous remarquerez que j'ai mis en commentaire le **Microsoft.EntityFrameworkCore** et en rajoutant à sa place **System.Data.Entity** comme explique dès le début du document. Cela nous permettra de pouvoir utiliser le **DbContext** propre à dotnet 4.8 .

N'oublier pas de mettre l'initialiseur de base.

INFO : "name=DefaultConnection" signifie que DbContext va chercher dans le Web.config une chaîne de connexion nommée DefaultConnection.

B. Configurer la chaîne de connexion

Vérifier dans votre explorateur de fichier dotnet.



Vous verrez un fichier nommé Web.config cliquez dessus. Dedans, vous verrez :

```
<?xml version="1.0" encoding="utf-8" ?>
<!--
  Pour plus d'informations sur la configuration de votre application ASP.NET, visitez
  https://go.microsoft.com/fwlink/?LinkId=301880
-->
<configuration>
  <configSections>
    <!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/fwlink/?LinkId=2374 -->
    <section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework" />
  </configSections>

  <!-- Cette ligne permet de lier votre instance SQLSERVER à votre projet -->
  <connectionStrings>
    <add name="DefaultConnection"
        connectionString="Data Source=_I_B_M_\DEV_SQLSERVER;Initial Catalog=banking_system;Integrated Security=True"
        providerName="System.Data.SqlClient" />
  </connectionStrings>

  <appSettings>
    <add key="webpages:Version" value="3.0.0.0" />
    <add key="webpages:Enabled" value="false" />
    <add key="ClientValidationEnabled" value="true" />
    <add key="UnobtrusiveJavaScriptEnabled" value="true" />
  </appSettings>
  <!--
    Pour obtenir une description des modifications de web.config, voir http://go.microsoft.com/fwlink/?LinkId=235367.

    Les attributs suivants peuvent être définis dans la balise <httpRuntime>.
    <system.Web>
  </-->
</configuration>
```

Ce qui nous intéresse ici c'est la balise `<connectionStrings></connectionStrings>` C'est lui qui nous

permettra de pouvoir lier notre projet .net à l'instance SQLSERVER.

Explication des attributs de la balise (name, connectionString, providerName)

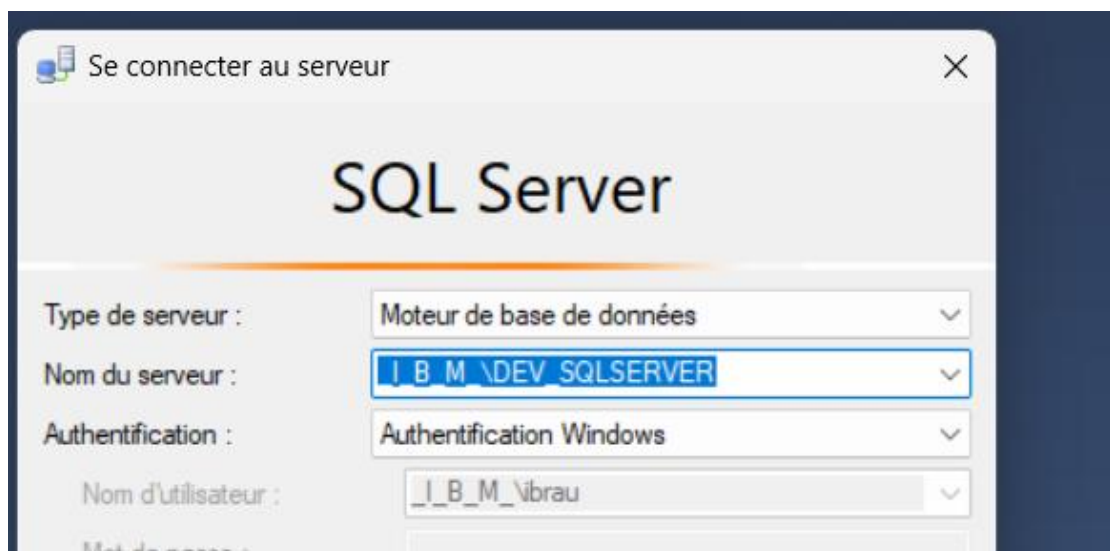
name : Elle nous permet de pouvoir définir un nom à notre balise connectionString et ce nom sera utilisé dans tout le projet lorsqu'on aura besoin de sa valeur. C'est un peu comme l'appel clé-valeur en python.

connectionString : C'est la valeur que retournera name lorsqu'on le fait appel dans le projet et comme son nom l'indique, il contient les identifiants de connexion à la base de données (C'est à dire le nom du serveur en question, la base de données, l'activation de la sécurité).

providerName : Il permet d'indiquer le pilote à utiliser pour la connexion à la base de données.

Copier ce bout de code XML et coller le dans le web.config à la même emplacement comme indiqué sur la figure ci-dessus.

```
<connectionStrings>
  <add name="DefaultConnection"
        connectionString="Data Source=NomDeVotreServeurSQLSERVER;Initial
        Catalog=banking_system;Integrated Security=True"
        providerName="System.Data.SqlClient" />
</connectionStrings>
```



C. Installation d'EntityFramework via NuGet

Ouvrir la console de gestion de package et exécuter

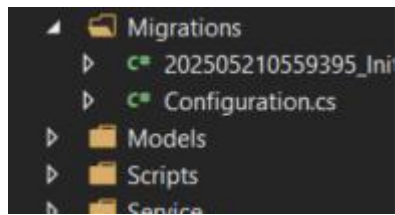
```
Install-Package EntityFramework
```

D. Activer la migration Code First

Ouvrir la console de gestion de package et exécuter

```
Enable-Migrations
```

Cela génère un dossier Migrations avec un fichier **Configuration.cs**.



E. Créer la migration initial

Ouvrir la console de gestion de package et exécuter

Add-Migration InitialCreate

Cette commande génère les scripts SQL nécessaires pour créer les tables.

F. Mettre à jour la base de donnée

Ouvrir la console de gestion de package et exécuter

Update-Database

Vérifier dans SQL Server Management Studio que la base et les tables sont créées.

