

TP n°2ASP.NET Core MVC (.NET 8.0)
DataBase First ApproachActions des contrôleurs et Entity Framework

Dans ce TP, nous allons travailler sur une solution ASP.NET Core MVC et nous allons nous baser sur l'approche de développement « DataBase First ».

Lancez Visual Studio Community 2022 et créez un nouveau projet.

Dans la première boîte qui apparaît, choisissez "**C#**", "**Toutes les plateformes**" et "**Web**". Sélectionnez "**Application Web ASP.NET Core (modèle-vue-contrôleur)**" et cliquez sur "**Suivant**".

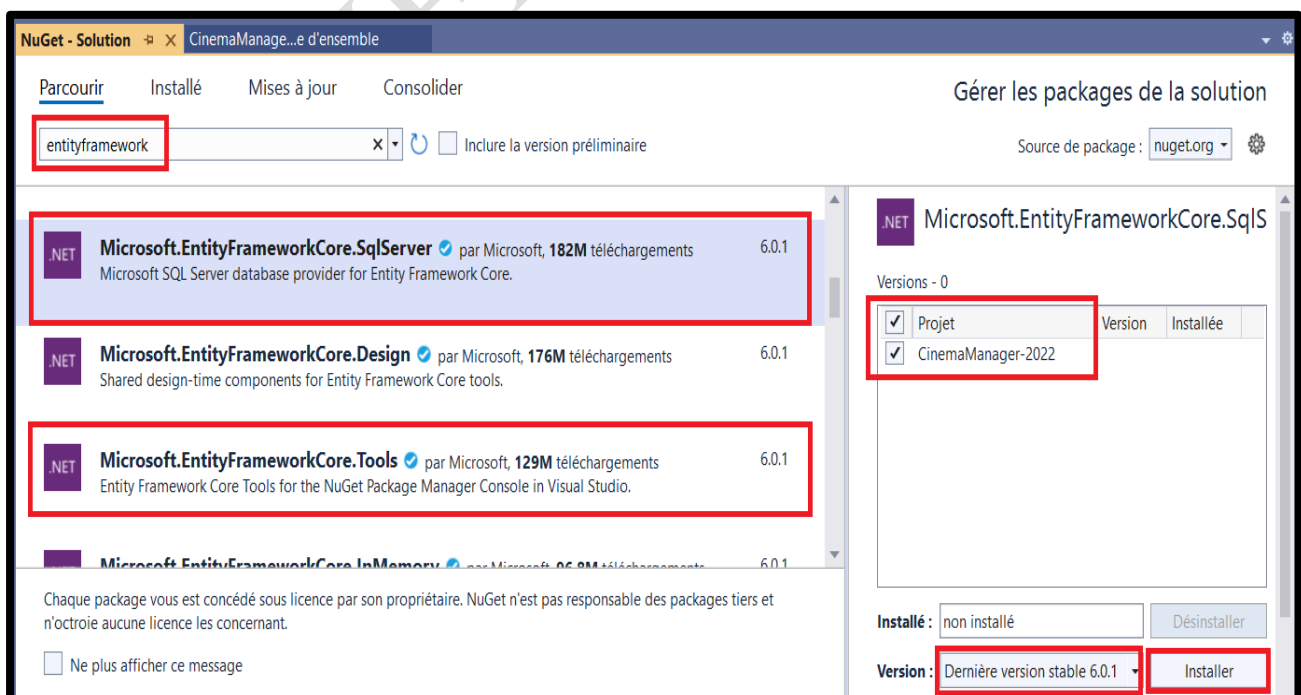
Dans la deuxième boîte, attribuer le nom **CinemaManager-X** au projet et à la solution (X est votre nom), puis cliquez sur "**Suivant**".

Dans la dernière boîte, choisissez "**.NET 8.0**" comme Framework, laissez les autres paramètres par défaut et cliquez sur "**Créer**".

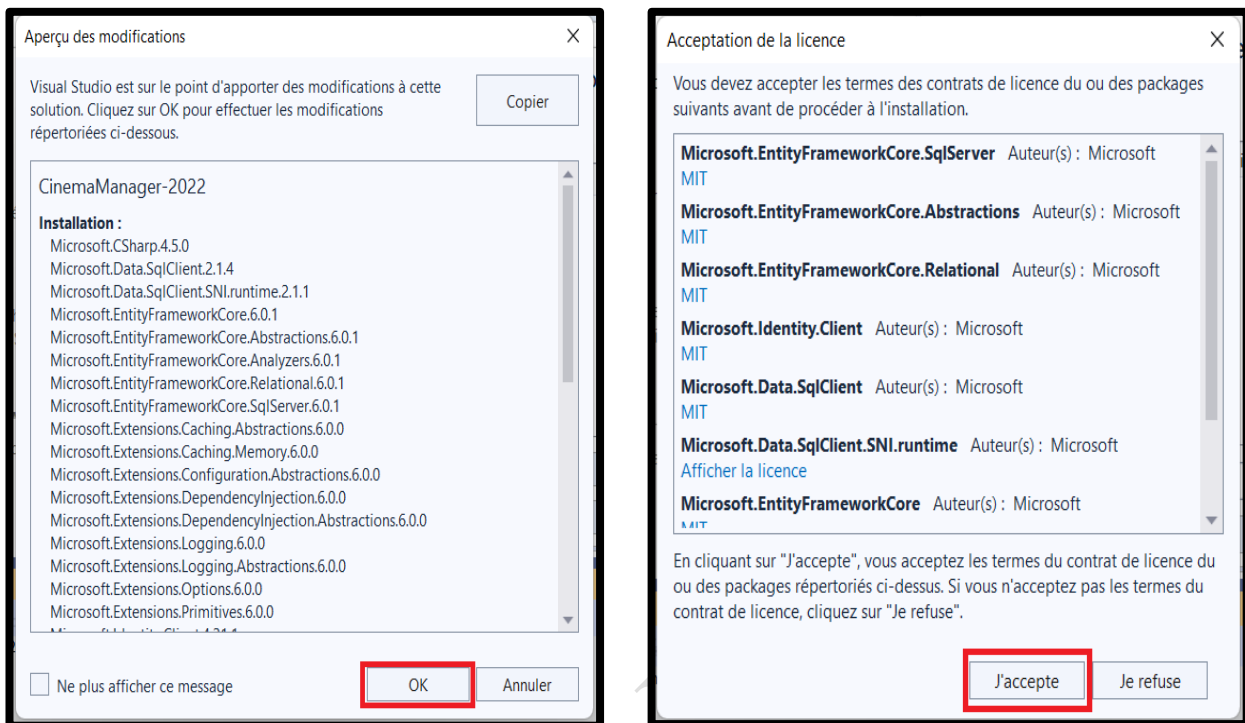
I. Installation d'EntityFrameworkCore (SqlServer et Tools)

A la différence des solutions ASP.NET MVC, Entity Framework n'est pas ajouté par défaut aux solutions ASP.NET Core. Pour cela, procédez à l'installation de ce Framework à partir de Nuget : Menu **Outils – Gestionnaire de package Nuget – Gérer les packages Nuget pour la solution...**

Passez à l'onglet « Parcourir », tapez dans la barre de recherche « **EntityFramework** », choisissez « **Microsoft.EntityFrameworkCore.SqlServer** », demandez son installation pour le projet en cours (voir figure ci-dessous), gardez la dernière version stable puis cliquez sur « Installer ».



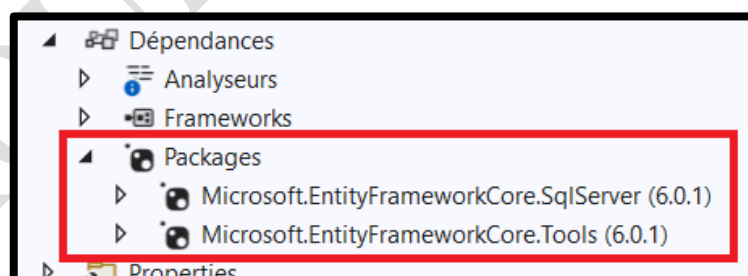
Visual Studio affiche qu'il procédera à l'installation d'un ensemble de packages liés à EFCore, cliquez sur « OK ». Puis acceptez les termes de licences (voir figures ci-dessous).



Une fois terminé, c'est bon, tous les sous-packages EFCore pour SQL Server sont maintenant disponibles pour le projet « CinemaManager » et sont prêts à être utilisés.

NB : Il faut refaire la même chose pour le package « *Microsoft.EntityFrameworkCore.Tools* ».

Une fois terminé, vérifiez que vous avez bel et bien ces packages dans l'explorateur de solutions (sous dépendances).

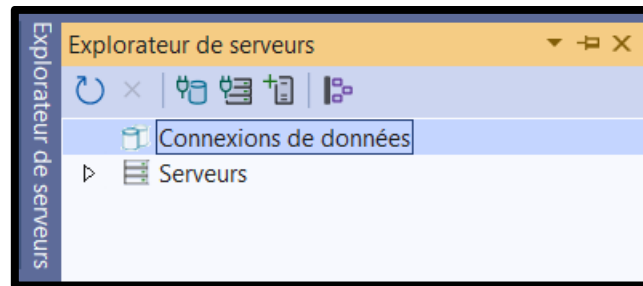


II. Création du modèle

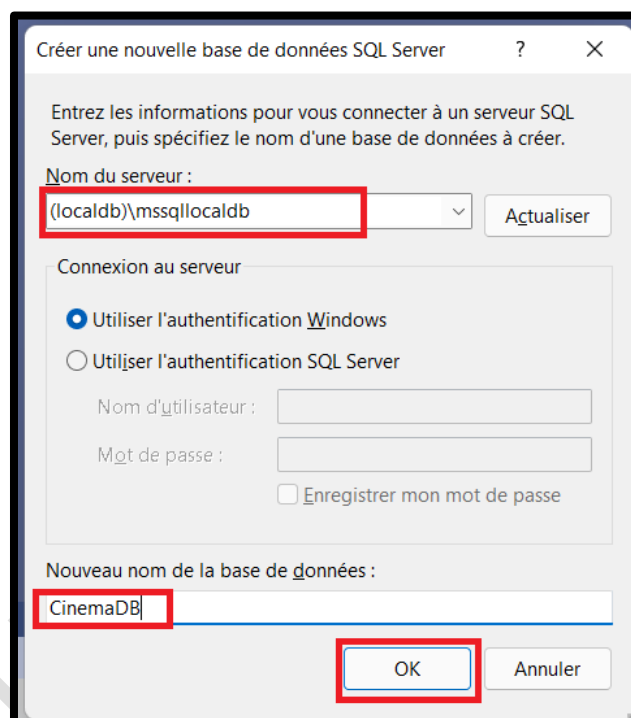
A. Création de la base de données

L'application **CinemaManager** utilise une base de données pour stocker les informations sur des films et leurs producteurs.

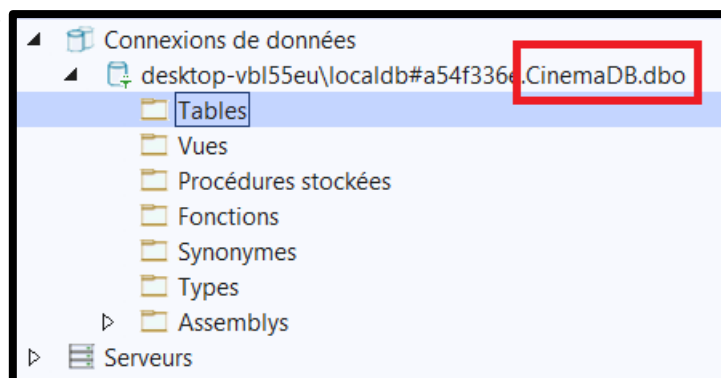
Pour créer une nouvelle base de données, ouvrez l'explorateur de serveurs (à gauche ou Affichage – Explorateur de Serveurs, voir figure ci-dessous).



Faites un clic droit sur « *Connexions de données* » et choisissez **Créer une nouvelle base de données SQL Server...** Dans la boîte de dialogue qui s'ouvre saisissez « **(localdb)\mssqllocaldb** » comme nom du serveur (*c'est le nom par défaut, attention les parenthèses font partie du nom et il y a 2 's' et 2 'l' qui se suivent*) puis attribuez le nom **CinemaDB** à la base puis cliquez sur OK (voir figure ci-dessous).



C'est bon, la base a été créée (elle est vide) et fait son apparition dans l'explorateur de serveurs.



B. Création de la table « Producer »

Faites un clic droit sur « Tables », puis « Ajouter une nouvelle table ».

Dans l'interface qui apparaît, créez les colonnes suivantes pour la table **Producer**.

Nom de colonne	Type de donnée	Indications
<i>Id</i>	<i>int</i>	<i>Clé primaire (ajoutée par défaut), auto-incrémentale, ne peut pas être NULL.</i>
<i>Name</i>	<i>nvarchar(30)</i>	<i>Ne peut pas être NULL</i>
<i>Nationality</i>	<i>nvarchar(30)</i>	<i>Ne peut pas être NULL</i>
<i>Email</i>	<i>nvarchar(30)</i>	<i>Ne peut pas être NULL</i>

NB :

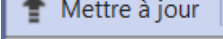
- Pour rendre l'Id auto-incrémental, il suffit soit d'ajouter la propriété **IDENTITY** dans le code T-SQL ou de modifier la propriété « Spécification du compteur » comme suit :

Spécification du compteur	False	=>	Spécification du compteur	True
(Est d'identité)	False		(Est d'identité)	True
Incrément d'identité			Incrément d'identité	1
Seed d'identité			Seed d'identité	1

- Pour attribuer le nom « Producer » à la table, écrivez ceci dans le code T-SQL qui devrait apparaître finalement comme suit :

```

1 CREATE TABLE [dbo].[Producer]
2 (
3     [Id] INT NOT NULL PRIMARY KEY IDENTITY,
4     [Name] NVARCHAR(30) NOT NULL,
5     [Nationality] NVARCHAR(30) NOT NULL,
6     [Email] NVARCHAR(30) NOT NULL
7 )
  
```

- Cliquez sur « Mettre à jour »  (au-dessus de l'espace de conception de la BD) puis encore « Mettre à jour la base de données » afin d'ajouter la table « Producer » à la BD.
- Vérifiez que la table a été ajoutée avec succès.



- Remplissez la table avec quelques enregistrements qui vont servir pour les prochains tests (clic-droit sur Producer, puis Afficher les données de la table).

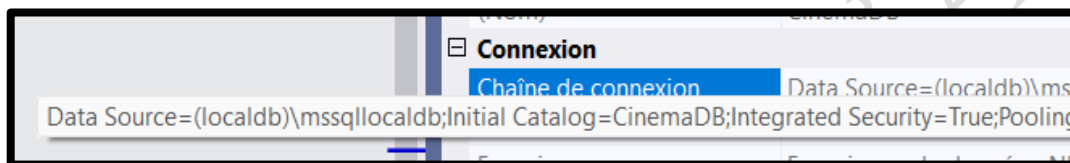
C. Création du modèle de données

Afin de créer le modèle de données correspondant à la base précédente, il faut suivre les étapes suivantes :

- Ajoutez la chaîne de connexion relative à la BD au fichier **appsettings.json**.

```
{
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "CinemaCS": "Data Source=(localdb)\\mssqllocaldb;Initial Catalog=CinemaDB;Integrated Security=True;Pooling=False"
  }
}
```

- o Pour récupérer cette chaîne de connexion, sélectionnez le nom de la base dans l'explorateur de serveurs, puis dans la fenêtre des propriétés copiez la valeur qui lui correspond et collez-la dans le fichier.



- Lancez la commande suivante dans la console du gestionnaire du package (Menu **Outils – Gestionnaire de package Nuget – Console du Gestionnaire du package**).

Scaffold-DbContext Name=**CS_Name** Microsoft.EntityFrameworkCore.SqlServer -OutputDir "**Target_Dir**"

NB :

- o **CS_Name** est le nom de la chaîne de connexion déclaré dans appsettings.json. Elle s'écrit *ConnexionStrings* :... car on va la lire à partir de l'environnement (variable d'environnement).
- o Le dossier de génération du modèle est « **Models\Cinema** » (voir figure ci-dessous).

```
PM> Scaffold-DbContext Name=ConnectionStrings:CinemaCS Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models\Cinema
Build started...
Build succeeded.
PM>
```

Cette commande créera un dossier « **Cinema** » sous « **Models** » contenant deux classes :

- o Une classe de modèle qui correspond à la table **Producer**.
- o Une classe de contexte « **CinemaDbContext** » qui hérite de **DbContext** et qui joue le rôle d'un pont entre les classes du modèle et la BD. Elle permettra d'unir tous les éléments du modèle dans un même contexte et de les manipuler avec des opérations de type CRUD.
 - Cette classe déclare une référence de type **DbSet** vers la classe **Producer**.
 - C'est cette référence « **Producers** » qui servira d'accéder aux données des producteurs et de les manipuler.

```
public virtual DbSet<Producer> Producers { get; set; } = null!;
```

- Ajoutez la classe de contexte et la chaîne de connexion à sa base dans le fichier « Program.cs » afin que l'application ouvre la connexion à la base lors de son démarrage. Cet ajout se fera sous forme d'une injection de dépendance d'un service (voir l'instruction encadrée dans la figure ci-dessous).
 - *Des usings seront nécessaires.*

```
// Add services to the container.
builder.Services.AddControllersWithViews();

builder.Services.AddDbContext<CinemaDbContext>(
    options => options.UseSqlServer(
        builder.Configuration.GetConnectionString("CinemaCS")
    ));

var app = builder.Build();
```

A ce niveau, c'est bon, le modèle est prêt à être utilisé par l'application pour interroger la base de données.

III. Contrôleurs et Vues

Commencez par générer la solution (compiler le tout) : Menu **Générer** puis **Générer la solution**. Ensuite, créez le contrôleur **ProducersController** en faisant un clic-droit sur le répertoire *Controllers* et en choisissant **Ajouter, Contrôleur...** Pour le template, choisissez **Contrôleur MVC avec des actions de lecture/écriture**.

Le nouveau contrôleur ajouté contient un ensemble de méthodes (*Index, Details, Create, Edit et Delete*) sur lesquelles vous allez agir afin de développer les CRUD des producteurs.

NB : Pour pouvoir tester l'application, il faut faire le nécessaire afin que le contrôleur *ProducersController* soit celui sur lequel démarre l'application à la place de *HomeController*.

A. Lister les producteurs

Permettez à l'action **Index** de ce contrôleur de lister les producteurs depuis la table **Producer** de la base de données. Bien évidemment, le code de l'action **Index** doit être modifié et la vue **Index** correspondante doit être créée.

NB :

- *Pour pouvoir accéder aux producteurs, il faut créer une instance de la classe **CinemaDbContext**. Ceci se fera par injection de dépendance au niveau du constructeur du contrôleur.*

```
CinemaDbContext _context;
public ProducersController(CinemaDbContext context)
{
    _context = context;
}
```

- Pour créer la vue *Index*, faites un clic droit dans le code de l'action *Index*, Ajouter une vue... puis choisissez *Vue Razor*.
- Dans la fenêtre **Ajouter Vue Razor**, gardez *Index* comme nom, choisissez l'option **List** pour le **template (Modèle)**, sélectionnez la classe de modèle **Producer (CinemaManager-X.Models.Cinema)** et la classe **CinemaDbContext (CinemaManager-X.Models.Cinema)** comme classe de contexte.

B. Créer de nouveaux producteurs

Les deux méthodes **Create()** du contrôleur agissent différemment. La première concerne l'action de type GET qui retourne un formulaire HTML pour la saisie des informations du nouveau producteur. La 2^{ème} concerne l'action de type POST et s'occupe d'insérer les informations saisies dans la base de données.

Permettez à cette action d'ajouter un nouveau producteur en modifiant l'argument de la 2^{ème} méthode (POST) et en utilisant la méthode **Add()** d'un objet producteur et **SaveChanges()** de l'objet **_context**.
*** **Garder la méthode GET telle quelle.**

Afin de tester l'action **Create**, il faut ajouter la vue correspondante (choisissez l'option **Create** pour le **template**).

C. Editer les producteurs

La première méthode **Edit()** accepte un paramètre **id** représentant l'identifiant de l'enregistrement devant être édité. Vous devez alors permettre à cette méthode de trouver le producteur correspondant à cet identifiant. Puis, une vue contenant un formulaire HTML pour éditer cet enregistrement doit être retournée.

NB :

- La recherche d'un producteur dans un modèle peut se faire grâce à la méthode **Find()**.
- Créez la vue correspondante avec le template **Edit**.

La deuxième méthode s'occupe de la mise à jour des données sur la base. Elle doit accepter une instance de la classe **Producer** comme paramètre.

NB : Utilisez les méthodes **Update()** et **SaveChanges()** pour la persistance des modifications.

D. Supprimer des producteurs

La première méthode de l'action **Delete** affiche un formulaire de confirmation de suppression. La 2^{ème} méthode s'occupe de la suppression réelle de l'enregistrement.

Utilisez la méthode **Remove()** et bien évidemment, il ne faut pas oublier **SaveChanges()**.

NB : Ajoutez la vue correspondante avec le template **Delete**.

E. Détailler un producteur

Développez l'action **Details** afin d'afficher les détails d'un producteur dans une vue appropriée.

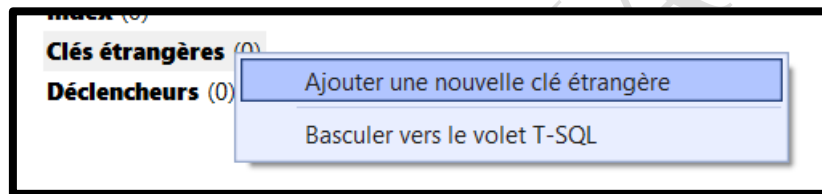
IV. (MAJ BD + MAJ Modèle + Nouveau Contrôleur + N^{lles} Vues)

1. Ajoutez à la BD, une table « **Movie** » ayant les colonnes suivantes :

Nom de colonne	Type de donnée	Indications
<i>Id</i>	<i>int</i>	Clé primaire (ajoutée par défaut), auto-incrémentale, ne peut pas être NULL.
<i>Title</i>	<i>nvarchar(30)</i>	Ne peut pas être NULL
<i>Genre</i>	<i>nvarchar(20)</i>	Ne peut pas être NULL
<i>ProducerId</i>	<i>int</i>	Clé étrangère, ne peut pas être NULL

2. Créez la relation entre les deux tables **Movie** et **Producer** basée sur les deux colonnes **ProducerId** du côté Movie et **Id** du côté Producer.

Faites un clic-droit sur « Clés étrangères », puis « Ajouter une nouvelle clé étrangère ». Choisissez un nom pour la relation, par exemple « FK_Movie_Prod » (FK pour dire Foreign Key).



Une nouvelle instruction apparaîtra au niveau du code T-SQL de la table **Movie**. Cette ligne est à modifier avec les bonnes informations.

3. Remplissez la table **Movie** avec des données de test. Vérifiez que l'association est bel et bien fonctionnelle (par exemple, elle doit refuser d'ajouter un film ayant un id d'un producteur inexistant).
4. Mettez à jour le modèle EntityFramework créé dans la Partie II en utilisant la même commande **Scaffold-DbContext** à laquelle ajoutez l'option **-Force** afin de forcer la régénération des classes existantes.
5. Créez un contrôleur pour les films (**MoviesController**) en choisissant l'option « **Contrôleur MVC avec vues, utilisant Entity Framework** ».

NB : La classe de modèle doit être « **Movie** »

6. Parcourez les différentes actions de **MoviesController** et testez-les afin de vous assurer que tout marche correctement.