

ASP.NET Core et Entity Framework core

MOHAMED AIROUCHE

AUTOMNE 2020

Entity Framework Core

- Entity Framework Core est la nouvelle version d'Entity Framework après EF 6.x. Il s'agit d'une version open source, légère, extensible et multiplateforme de la technologie d'accès aux données Entity Framework.
- Entity Framework est un cadre de mappage objet/relationnel (ORM). Il s'agit d'une amélioration d'ADO.NET qui offre aux développeurs un mécanisme automatisé pour accéder et stocker les données dans la base de données.
- EF Core est destiné à être utilisé avec les applications .NET Core. Cependant, il peut également être utilisé avec des applications basées sur le framework .NET 4.5+ standard.

Historique des versions d'EF Core

EF Core Version	Release Date
EF Core 2.0	August 2017
EF Core 1.1	November 2016
EF Core 1.0	June 2016

<https://www.entityframeworktutorial.net/efcore/entity-framework-core.aspx>

Exemple

Développement d'une application qui permet de gérer des employés et des départements :

On souhaite créer une application qui permet de gérer des employés et les départements de ces employés. Un département est défini par son identifiant et son nom. Chaque employé est défini par son identifiant, son nom, son prénom, son sexe, son identifiant de département et sa ville.

L'application doit permettre de :

➤ Pour les employés :

- Afficher tous les employés.
- Afficher les détails d'un employé.
- Saisir et ajouter un nouvel employé.
- Éditer et modifier un employé.
- Supprimer un employé.

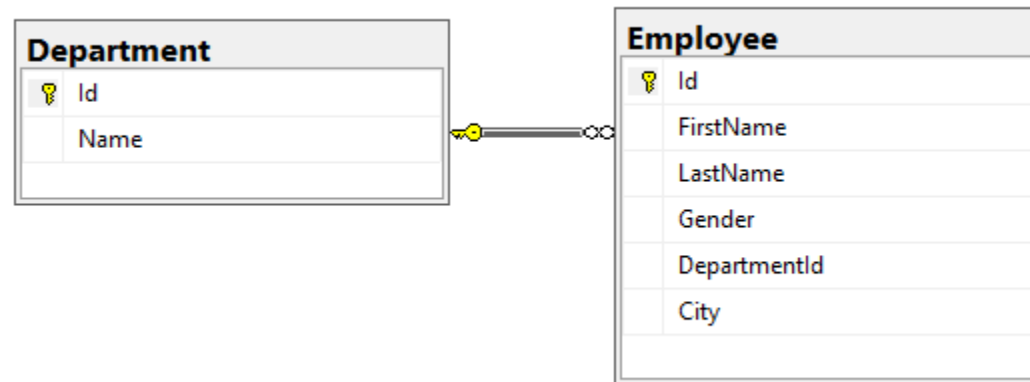
➤ Pour les départements :

- Afficher tous les départements.
- Afficher les détails d'un département.
- Saisir et ajouter un nouveau département.
- Éditer et modifier un département.
- Supprimer un département.

Exemple

Développement d'une application qui permet de gérer des employés et des départements :

Les employés et les départements seront stockés dans une base de données locale SQL Server Express. Cette base de données doit contenir deux tables (table Employee et table Department).

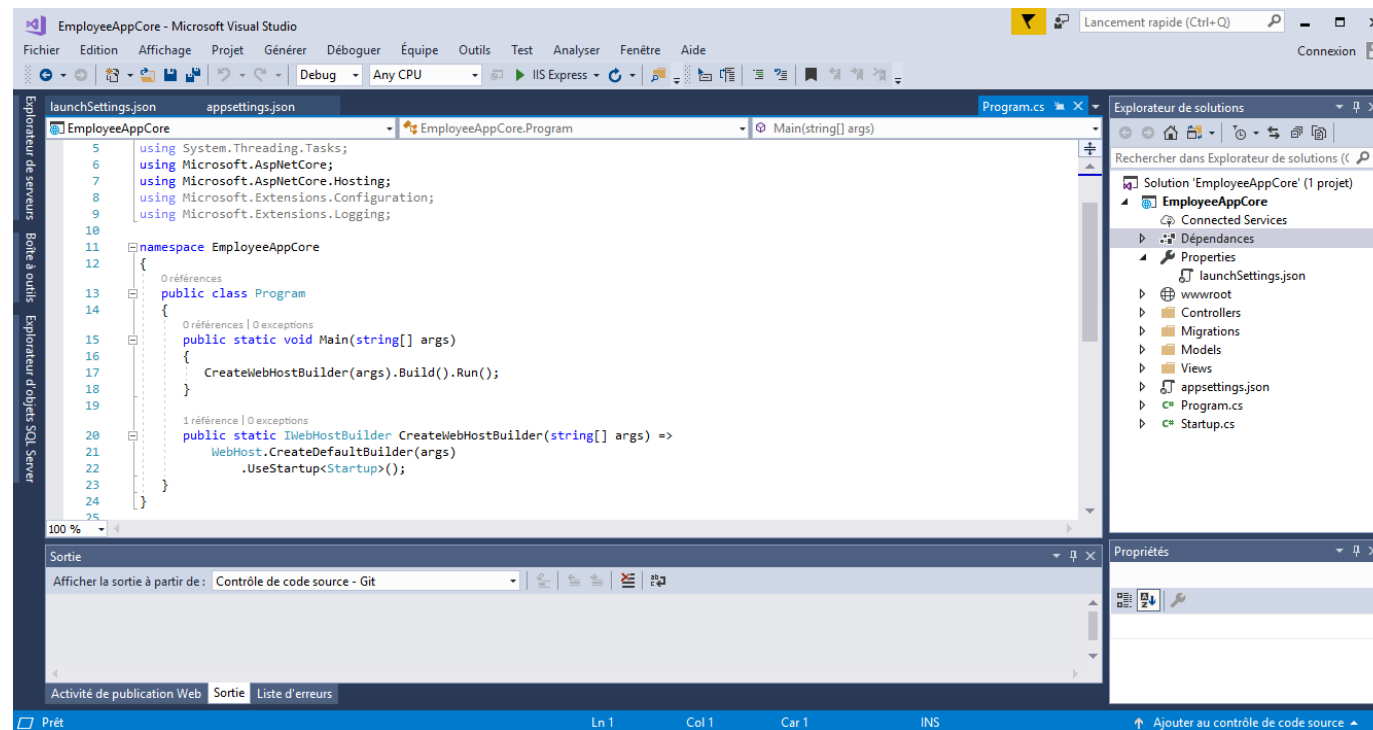


Dans cet exemple, l'application utilisera l'approche Code First de l'ORM Entity Framework Core pour l'accès aux données.

Exemple

Création d'une application ASP.NET Core MVC :

- Créez une application ASP.NET Core MVC : **EmployeeAppCore**.
- Dans cet exemple nous utiliserons l'approche Code First de l'ORM Entity Framework Core.



Exemple

Ajout de modèle à utiliser par Entity Framework :

- ✓ Pour utiliser l'approche Code First d'ORM Entity Framework Core, il faut ajouter les classes des entités et la classe de contexte qui hérite de la classe DbContext.
- ✓ La classe de contexte de la base de données : [EmployeeDbContext](#)

```
using Microsoft.EntityFrameworkCore;

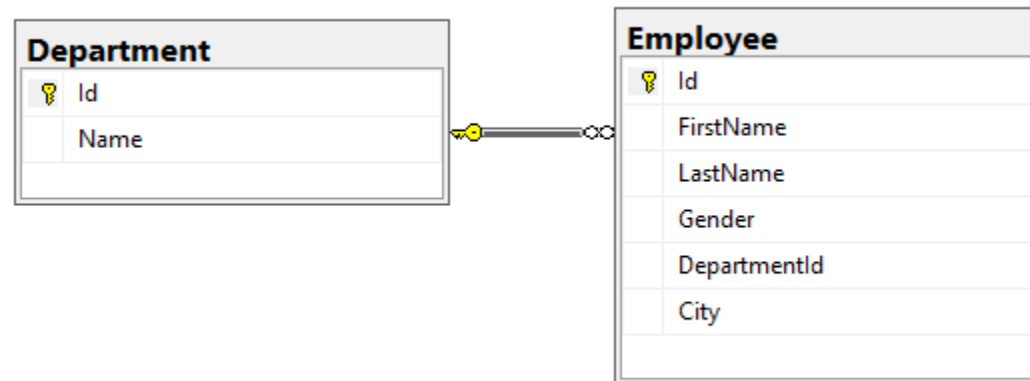
namespace EmployeeAppCore.Models
{
    public class EmployeeDbContext : DbContext
    {
        public EmployeeDbContext(DbContextOptions<EmployeeDbContext> options)
            : base()
        {
        }
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseSqlServer("Server =(localdb)\\mssqllocaldb;Database=EmployeeDbMVC;Trusted_Connection=True;");
        }
        public DbSet<Employee> Employees { get; set; }

        public DbSet<Department> Departments { get; set; }
    }
}
```

Exemple

L'association : One To Many ----- ManyToOne :

La base de données doit contenir deux tables reliées (table Employee et table Department).



- ✓ Prenons le cas où chaque employé appartient à un département et un département contient plusieurs employés.
- ✓ Dans ce cas, nous avons deux entités : Employee et Department.
- ✓ La relation qui lie ces deux entités est de type OneToMany \longleftrightarrow ManyToOne

Exemple

✓ La classe d'entité : **Employee**

```
using System.ComponentModel.DataAnnotations;

namespace _09_DemoEmployeeDepartementEF.Models
{
    public class Employee
    {
        public int Id { get; set; }
        [Required]
        public string FirstName { get; set; }
        [Required]
        public string LastName { get; set; }
        [Required]
        public string Gender { get; set; }
        [Required]
        public int DepartmentId { get; set; }
        [Required]
        public string City { get; set; }
        public virtual Department Department { get; set; }
    }
}
```

✓ La classe d'entité : **Department**

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace _09_DemoEmployeeDepartementEF.Models
{
    public class Department
    {
        public int Id { get; set; }
        [Required]
        public string Name { get; set; }
        public virtual ICollection<Employee> Employees { get; set; }
    }
}
```

Propriété de clé

Propriétés de navigation

Exemple

Lazy Loading


- Dans le monde des ORMs, une technique appelée "le lazy loading" est utilisée pour éviter de charger les entités liées en mémoire sans en avoir besoin. Cela peut être intéressant de faire si vous devez précharger en une seule requête un nombre important de données.
- Dans notre exemple, quand on demande à Entity Framework de charger un Objet employé, il ne va pas charger le Département lié à cet employé.
- Pour activer le lazy loading, il faut ajouter le mot clé **Virtual** devant les propriétés de navigation.
- Pour charger les entités liées, on peut utiliser la méthode **Include** dans les requêtes LINQ.
- L'exemple suivant permet de retourner la liste des employés et pour chaque employé, il charge avec son département: **return employees = db.Employees.Include(e => e.Department)**
- Si la méthode Include n'est pas utilisée, il ne va charger que la liste des employés.

Exemple


Inscrire le contexte de base de données

ASP.NET Core comprend l'injection de dépendances. Les services (tels que le contexte de base de données EF Core) doivent être inscrits auprès de l'injection de dépendances au démarrage de l'application. Ces services sont fournis par les composants qui requièrent ces services par le biais de paramètres de constructeur. Vous allez inscrire le contexte de base de données auprès du conteneur d'injection de dépendances.

- ✓ En tête du fichier *Startup.cs*, ajoutez l'instruction using suivante :

 `using Microsoft.EntityFrameworkCore;`

- ✓ Ajoutez le code en surbrillance suivant dans la méthode `Startup.ConfigureServices` :

```
public void ConfigureServices(IServiceCollection services)
{
     services.AddDbContext<EmployeeDbContext>(options =>
    options.UseSqlServer(Configuration.GetConnectionString("EmployeeConnection")));
    services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_1);
}
```


<https://docs.microsoft.com/fr-ca/aspnet/core/tutorials/first-mvc-app/adding-model?view=aspnetcore-3.1&tabs=visual-studio>

Exemple

Ajouter une chaîne de connexion de base de données

Ajoutez une chaîne de connexion dans le fichier *appsettings.json* :

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "EmployeeConnection":
      "Server=(localdb)\\mssqllocaldb;Database=EmployeeDbMVC;Trusted_Connection=True;MultipleActiveResultSets=true"
  }
}
```



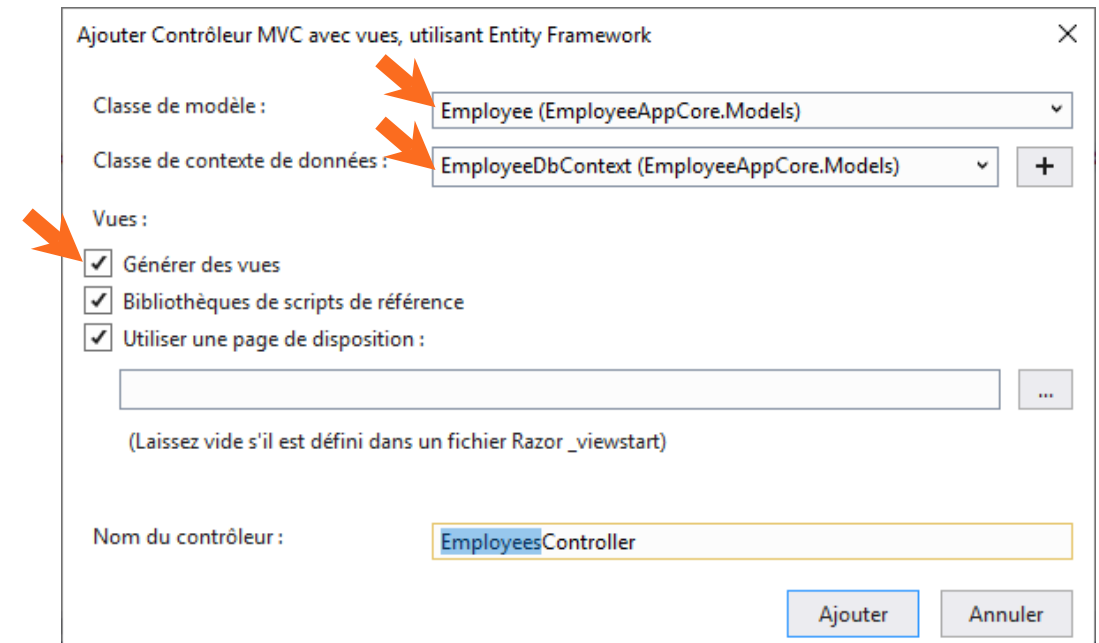
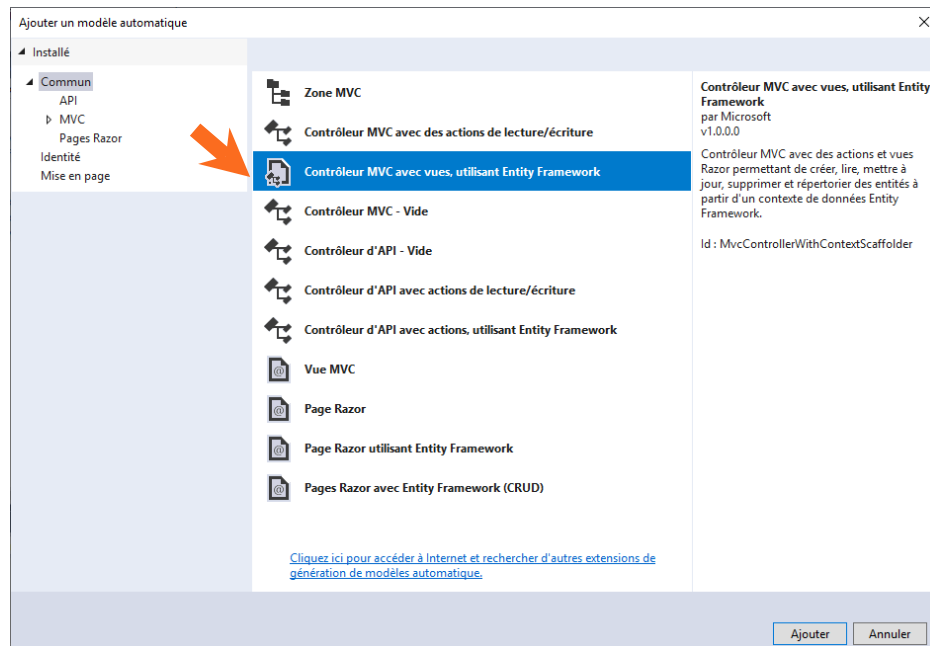
Générez le projet pour la vérification des erreurs du compilateur.

<https://docs.microsoft.com/fr-ca/aspnet/core/tutorials/first-mvc-app/adding-model?view=aspnetcore-3.1&tabs=visual-studio>

Exemple

Générer automatiquement du contrôleur Employee des pages d'employés

Utilisez l'outil de génération de modèles automatique pour créer, lire, mettre à jour et supprimer des pages du modèle **Employee**.



<https://docs.microsoft.com/fr-ca/aspnet/core/tutorials/first-mvc-app/adding-model?view=aspnetcore-3.1&tabs=visual-studio>

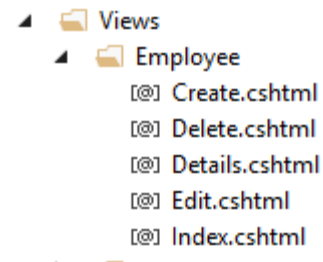
Exemple

Visual studio vas créer automatiquement dans le contrôleur **Employee** les méthodes d'actions suivantes :

- Index
- Create (GET, POST)
- Edit (GET, POST)
- Delete (GET, POST)
- Details

De plus, pour chaque méthode d'action de contrôleur Visual studio ajoute une vue. Au final, on doit trouver dans le dossier **Views/Employee** les vues suivantes :

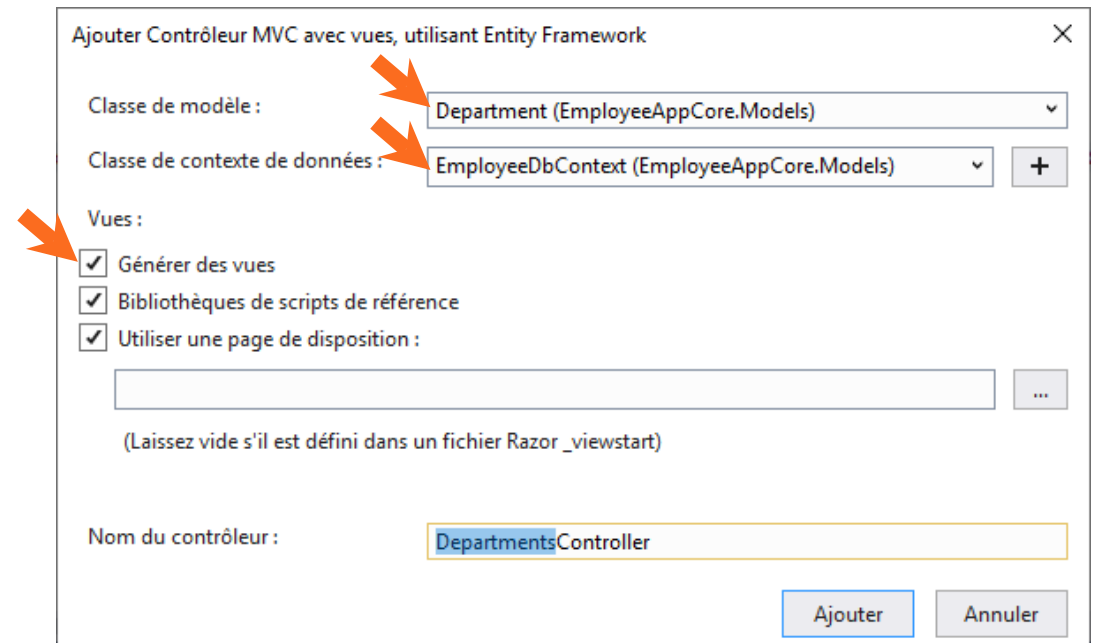
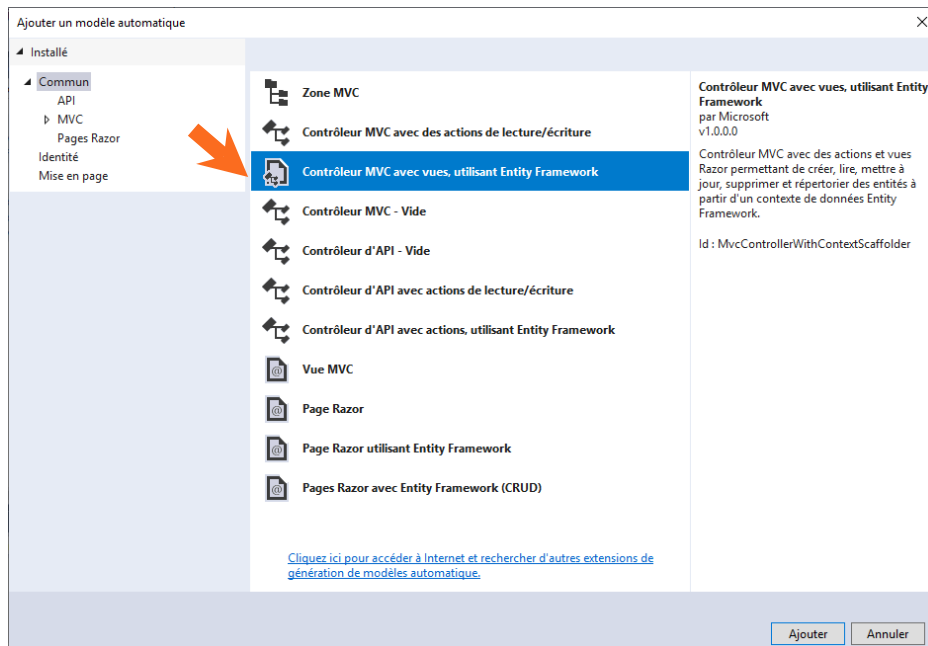
- Index.cshtml
- Create.cshtml
- Edit.cshtml
- Delete.cshtml
- Details.cshtml



Exemple

Générer automatiquement du contrôleur Department des pages de départements

Utilisez l'outil de génération de modèles automatique pour créer, lire, mettre à jour et supprimer des pages du modèle **Department**.



<https://docs.microsoft.com/fr-ca/aspnet/core/tutorials/first-mvc-app/adding-model?view=aspnetcore-3.1&tabs=visual-studio>

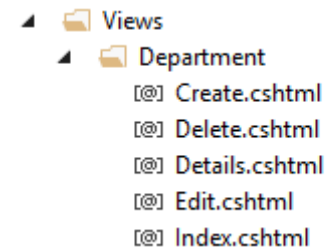
Exemple

Visual studio vas créer automatiquement dans le contrôleur **Department** les méthodes d'actions suivantes :

- Index
- Create (GET, POST)
- Edit (GET, POST)
- Delete (GET, POST)
- Details

De plus, pour chaque méthode d'action de contrôleur Visual studio ajoute une vue. Au final, on doit trouver dans le dossier **Views/Department** les vues suivantes :

- Index.cshtml
- Create.cshtml
- Edit.cshtml
- Delete.cshtml
- Details.cshtml



Exemple

Vous ne pouvez pas encore utiliser les pages générées automatiquement, car la base de données n'existe pas. Si vous exécutez l'application et cliquez sur le lien de l' **application de film** , vous recevez un message d'erreur Impossible d' *ouvrir la base de données* ou une *table de ce type* : message d'erreur.

Migration initiale

Utilisez la fonctionnalité Migrations d'EF Core pour créer la base de données. La fonctionnalité Migrations est un ensemble d'outils qui vous permettent de créer et de mettre à jour une base de données pour qu'elle corresponde à votre modèle de données.

Dans le menu **Outils**, sélectionnez **Gestionnaire de package NuGet > Console du gestionnaire de package** (PMC).

Dans la console du gestionnaire de package, entrez les commandes suivantes :

 **Add-Migration** InitialCreate
Update-Database

<https://docs.microsoft.com/fr-ca/aspnet/core/tutorials/first-mvc-app/adding-model?view=aspnetcore-3.1&tabs=visual-studio>

Exemple

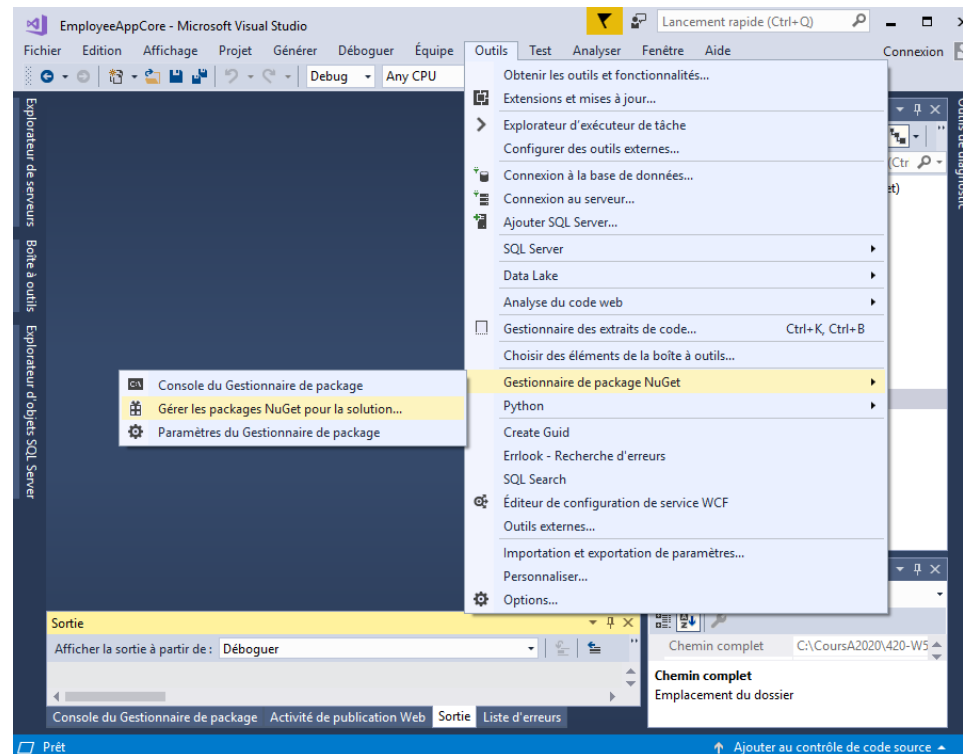
- **Add-Migration InitialCreate** : Génère un fichier de migration *_InitialCreate.cs* *migrations/{timestamp}*. L'argument **InitialCreate** est le nom de la migration. Vous pouvez utiliser n'importe quel nom, mais par convention, un nom décrivant la migration est sélectionné. Étant donné qu'il s'agit de la première migration, la classe générée contient du code permettant de créer le schéma de la base de données. Le schéma de base de données est basé sur le modèle spécifié dans la classe `EmployeeDbContext`.
- **Update-Database** : Met à jour la base de données avec la dernière migration, créée par la commande précédente. La commande exécute la méthode `Up` dans le fichier *Migrations/{horodatage}_InitialCreate.cs*, ce qui entraîne la création de la base de données.

<https://docs.microsoft.com/fr-ca/aspnet/core/tutorials/first-mvc-app/adding-model?view=aspnetcore-3.1&tabs=visual-studio>

Exemple

Ajout d'Entity Framework Core au projet :

- ✓ Pour ajouter une dépendance à un projet, utilisez l'outil NuGet en cas où votre projet n'arrive pas à ajouter la dépendance EF.



Exemple

Ajout d'Entity Framework Core au projet :

- ✓ Pour ajouter Entity Framework Core au projet, utilisez l'outil NuGet.
- ✓ Chercher Entity Framework Core puis cocher la case à cocher pour l'ajouter au projet.

