**Créer une solution mvc – Razor avec Identity**

# MVC Login and Registration with Identity - Identity Customization - .NET Core 6 - Part 1

https://www.youtube.com/watch?v=I-ZzFLruiuo&t=592s

# Tutoriel Étape par Étape: Application Complète CRUD avec ASP.NET Core MVC & Entity Framework Core

https://www.youtube.com/watch?v=uBFg1azlG-8

Nous utiliserons DotNet 6.0 car c’est un Framework modulaire crossplateforme.

1. **Créer un nouveau projet**

Choisir le modèle Application web ASP.Net Core (modèle-vue-contrôleur)

Donner un nom à la solution (appelé Solution.Name dans ce document)

Choisir l’emplacement

Infrastructure : .NET 6.0

Type d’authentification : Comptes individuels

1. **Packages et Json**

Vérifier que les packages NuGet suivants sont bien installés (en prenant la dernière version 6)  :

Microsoft.EntityFrameworkCore (++)

Microsoft.VisualStudio.Web.Codegeneration.Design (++)

Microsoft.EntityFrameworkCore.Design

Microsoft.EntityFrameworkCore.SqlServer

Microsoft.EntityFrameworkCore.Tools

Puis ajouter un nouvel élément généré automatiquement : Identité

Cochez « Remplacer tous les fichiers » ce qui va en fait cocher toutes les cases

Choisissez la classe de contexte de données : ApplicationDbContext(Solution.Name.Data)

Ouvrir L’explorateur de serveur pour noter le nom de votre machine.

Dans appsettings.json, Remplacer le ligne « DefaultConnection » par :

"DefaultConnection": "Server=(Nom de votre machine trouvé dans l’explorateur de serveur);Database=(Nom de la base de données à créer);Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False"

1. **1ere Migration**

Lancer le programme et cliquer sur « Register »

Créer un compte jusqu’à arriver sur une page proposant : Apply Migrations

Fermer le programme

Dans « Outils » / Gestionnaire de packages NuGet / Console du gestionnaire de package, entrer :

Update-Database

1. **Program.cs**

Dans Program.cs, remplacer :

builder.Services.AddDefaultIdentity<IdentityUser>(options => options.SignIn.RequireConfirmedAccount = true)

.AddEntityFrameworkStores<ApplicationDbContext>();

Par :

builder.Services.AddDefaultIdentity<IdentityUser>(options => options.SignIn.RequireConfirmedAccount = false)

.AddRoles<IdentityRole>()

.AddEntityFrameworkStores<ApplicationDbContext>();

1. **Classe User**

Parmi les Models, créer une nouvelle classe « User », et ajouter ces attributs :

public string Id { get; set; }

public string UserName { get; set; }

public string NormalizedUserName { get; set; }

public string Email { get; set; }

public string NormalizedEmail { get; set; }

public bool EmailConfirmed { get; set; }

public string PasswordHash { get; set; }

public string SecurityStamp { get; set; }

public string ConcurrencyStamp { get; set; }

public string PhoneNumber { get; set; }

public bool PhoneNumberConfirmed { get; set; }

public bool TwoFactorEnabled { get; set; }

public DateTimeOffset LockoutEnd { get; set; }

public bool LockoutEnabled { get; set; }

public int AccessFailedCount { get; set; }

Puis, dans le context, ajoutez le modèle dans la classe (à la fin) :

public DbSet< Solution.Name.Models.User> User { get; set; }

1. **Controller ManageUser**

Parmi les contrôleurs, a jouter un contrôleur (utilisant Entity Framework).

Choisir la classe « User » et la classe de contexte de données qui est proposé.

Donner un nom qui finit par « Controller » au controlleur (ici : ManageUserController)

Ajouter cet attribut parmi les readonly :

private readonly ILogger<ManageUserController> \_logger;

Puis modifier le constructeur pour y ajouter la variable :

public ManageUserController(ApplicationDbContext context, ILogger<ManageUserController> logger)

{

\_context = context;

\_logger = logger;

}

Enfin, remplacer la fonction « Index » par :

//[Authorize(Roles = "Admin")]

public ActionResult Index()

{

var uer = \_context.Users.ToList();

var list = \_context.Users.Select(user =>

new User

{

Id= user.Id,

UserName=user.UserName,

NormalizedUserName=user.NormalizedUserName,

Email=user.Email,

NormalizedEmail=user.NormalizedEmail,

EmailConfirmed=user.EmailConfirmed,

PasswordHash=user.PasswordHash,

SecurityStamp=user.SecurityStamp,

ConcurrencyStamp=user.ConcurrencyStamp,

PhoneNumber=user.PhoneNumber,

PhoneNumberConfirmed=user.PhoneNumberConfirmed,

TwoFactorEnabled=user.TwoFactorEnabled,

LockoutEnabled=user.LockoutEnabled,

AccessFailedCount=user.AccessFailedCount,

}).ToList();

//Content("Admin");

return View(list);

}

Il est possible de limiter l’accès aux admins seulement en décommentant la première et l’avant dernière ligne.

1. **Acces admin via \_layout**

Dans le \_Layout, ajouter cette balise « li » dans la navbarre :

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-area**="" **asp-controller**="ManageUser" **asp-action**="Index">Admin</**a**>

</li>

Il est possible de rendre cet item visible uniquement pour les admins en le mettant dans une balise razor :

@if (User.IsInRole( "Admin"))

{

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-area**="" **asp-controller**="ManageUser" **asp-action**="Index">Admin</**a**>

</li>

}

1. **Base de donnée**

Dans sql server manager, retrouver votre base de donnée.

Tenter d’afficher le diagramme, un message à accepter vous proposera d’ajouter des éléments manquants pour créer le diagramme.

Faire un clic droit pour créer un nouveau schéma, ajouter l’ensemble des tables.

Les relations vont se générer automatiquement.

1. **2eme Migration**

Dans « Outils » / Gestionnaire de packages NuGet / Console du gestionnaire de package, entrer :

**Créer une migration**

PM> Add-Migration InitialCreate -Context (nom du contexte) -OutputDir (route de la migration)

Exemple :

PM> Add-Migration InitialCreate -Context ApplicationDbContext -OutputDir Data\Migrations\

**appliquer la migration :**

PM> Update-Database -Context (nom du contexte)

1. **MCD final**

Voir le doc 2 comme modèle conceptuel complet pour un site marchand.

**Ce document n’existe pas encore**

Créer toutes les tables qui semble nécessaires, ajouter des colonnes au besoin et faites les relations

1. **Reverse engineering**

Dans « Outils » / Gestionnaire de packages NuGet / Console du gestionnaire de package, entrer :

Scaffold-DbContext "Data Source=(Nom de votre machine trouvé dans l’explorateur de serveur);Initial Catalog=(Nom de la base de données);Integrated Security=True;" Microsoft.EntityFrameworkCore.SqlServer -ContextDir Data -OutputDir Models

Les classes vont se générer automatiquement dans le programme.

1. **Bootstrap**

Bootstrap est un Framework full-responsive.

Choisir un modèle sur internet.

Plutôt que de copier les bibliothèques, faire un clic droit sur le dossier « lib » (dans root) pour ajouter des bibliothèques coté client et rechercher les librairies

Ainsi elles seront géré par libman.

Vérifier bien le chemin relatif dans le fichier libman.json

Copier aussi dans root les images et fonts (en créant des dossier si nécessaire)

Pour les bibliothèques « maison » (js, css), les copier directement dans le projet soit dans le dossier js, soit dans css.

Pour les bibliothèques « maison » qui contiennent et du js et du css, créer un autre dossier « biblio » dans root.

Ainsi tout ce qui est dans « lib » est géré et maintenu par libman, tout ce qui n’y est pas sont des bibliothèques « maison ».

**13) SQL Server et SSMS**

Instance par défaut : MSSQLSERVER

Utilisateurs :

Pierre-Yves

peter

loj

MDP standard PY

Nom de l'Ordinateur : DESKTOP-L4KCR01

**ANNEXES**

**Références**

**Création d’une appli MVC avec ses propres utilisateurs et utilisant Identity**

https://www.youtube.com/watch?v=I-ZzFLruiuo&t=592s

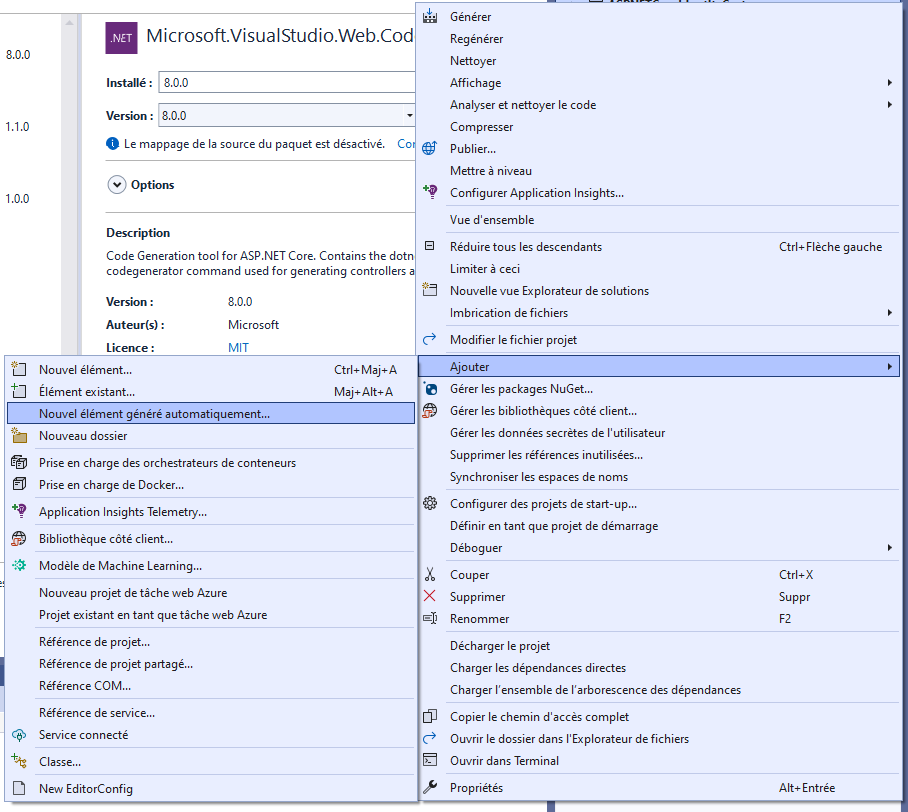
**Commencer par créer un projet MVC sans authentification**

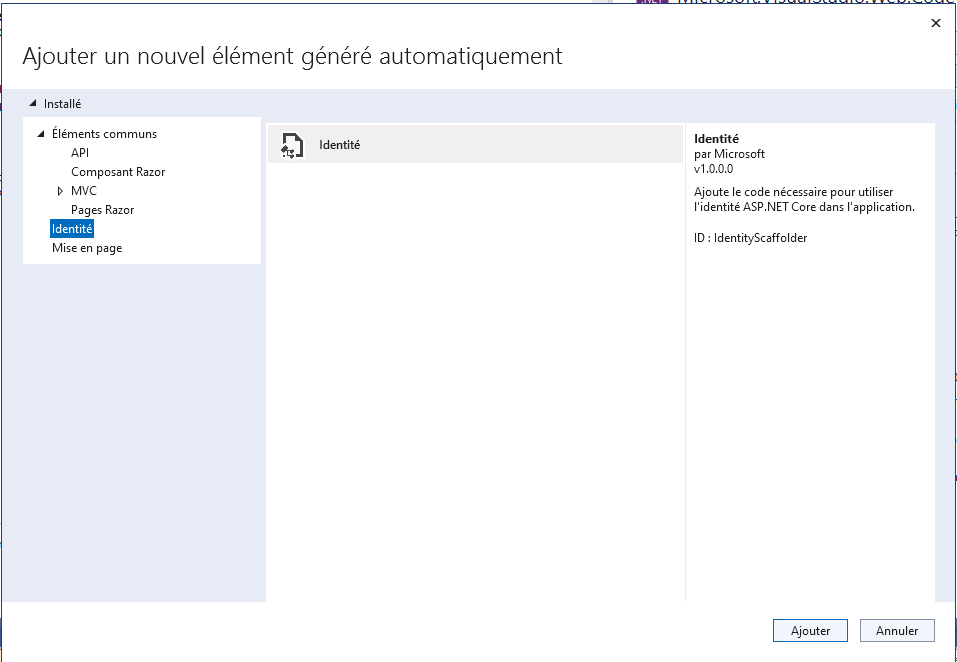
**Ajouter le paquet Nuget Microsoft.VisualStudio.Web.CodeGeneration.Design**

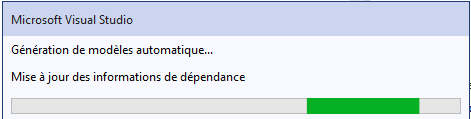
Clic droit sur le projet

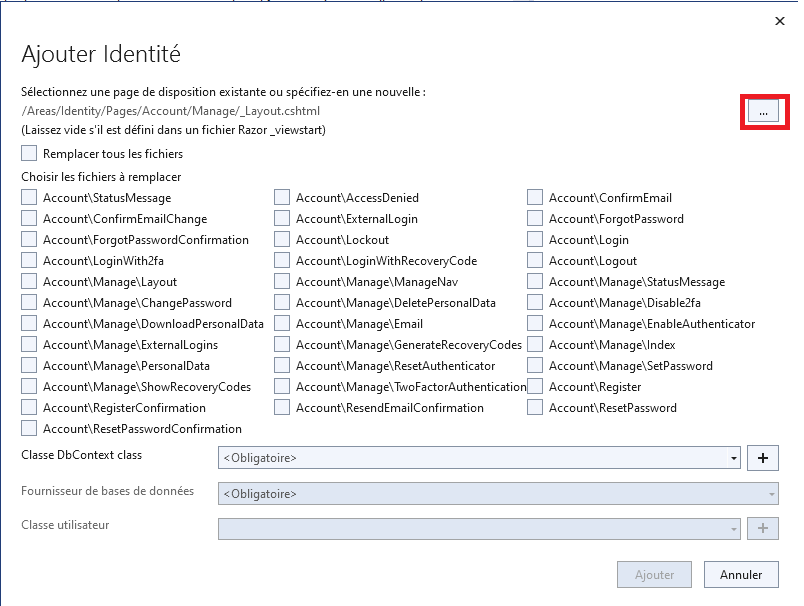
Ajouter

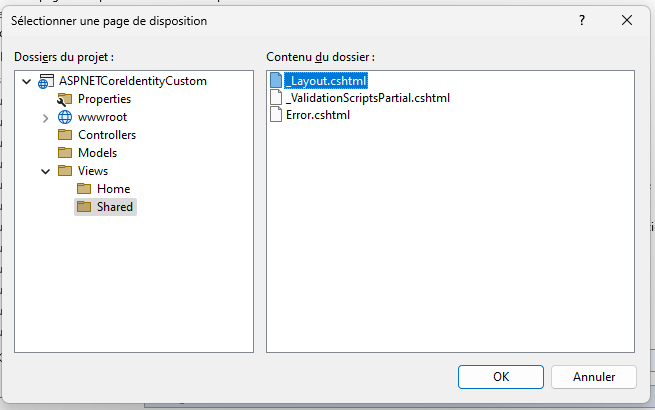
Nouvel élément généré automatiquement











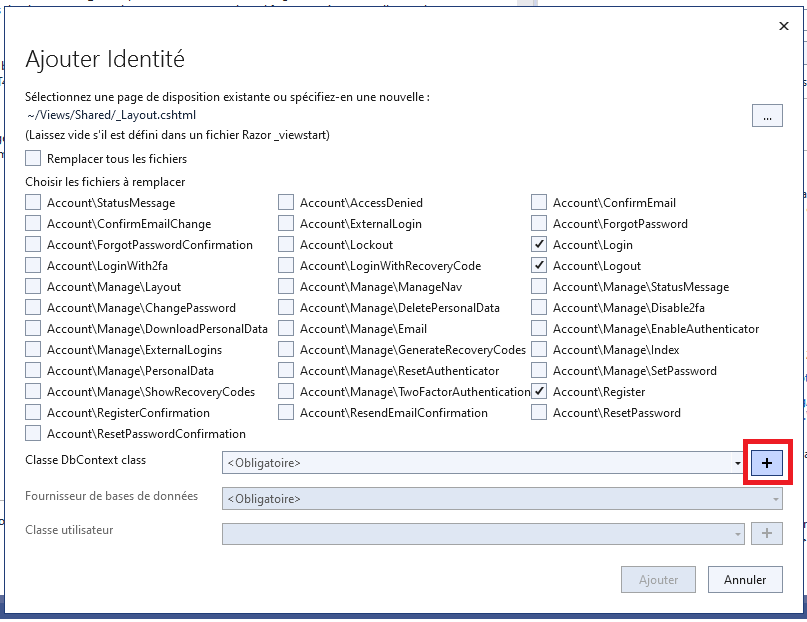
On revient sur la fenêtre précédente. On coche :

Account\Login

Account\Login

Account\Register

Puis on clique sur le signe + pour ajouter un DbContext



On obtient une fenêtre avec un nom par défaut assez long.

On choisit un nom plus simple et plus court : ApplicationDbContext (mais on peut garder le nom par défaut ou mettre tout nom à notre convenance

On clique sur « Ajouter «  pour terminer »



On sélectionne ensuite une base de données

Dans cet exemple, on prendra SQL Server

Enfin, de la même façon que l’on a ajouté un DbContext, on choisira un nom classe pour l’utilisateur. Là encore, plutôt que le nom par défaut, on choisira ApplicationUser

**Ajouter Identity**

https://dotnet.developpez.com/actu/307454/IdentityServer4-utiliser-ASP-NET-Core-Identity-pour-l-authentification-un-billet-de-blog-d-Hinault-Romaric/

**Chaine de connexion en .NET 8 ?**

"DefaultConnection": "Server=DESKTOP-L4KCR01;Database=MVC03;Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False"

**Chaine de connexion en .NET 6 ? (marche avec le projet MVC 03)**

{

"ConnectionStrings": {

"DefaultConnection": "Server=DESKTOP-L4KCR01;Database=MVC03;Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False"

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*"

}

Passage de .NET6 en .NET 8, selon Bard

To convert the provided connection string from .Net 6 to .Net 8, you'll need to make the following modifications:

1. **Replace Data Source with Server:**

DefaultConnection": "Server=DESKTOP-L4KCR01;Database=MVC03;Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False"

1. **Replace Initial Catalog with Database:**

DefaultConnection": "Server=DESKTOP-L4KCR01;Database=MVC03;Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False"

1. **Replace Integrated Security with User ID:**

DefaultConnection": "Server=DESKTOP-L4KCR01;Database=MVC03;User ID=;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False"

The updated connection string will now be compatible with .Net 8:

DefaultConnection": "Server=DESKTOP-L4KCR01;Database=MVC03;User ID=;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False"

You can also use the DbConnectionStringBuilder class to simplify the conversion. Here's an example of how to do this:

C#

var connectionString = new DbConnectionStringBuilder("DefaultConnection": "Server=DESKTOP-L4KCR01;Database=MVC03;Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False");

connectionString.DataSource = "Server";

connectionString.Database = "Database";

connectionString.UserID = "";

var connection = new MyDbContext(connectionString);

Utilisez le code avec précaution. [En savoir plus](https://bard.google.com/faq#coding)

This code will produce the same updated connection string as the previous one.

**Sources**

1. [blazor.syncfusion.com/documentation/common/data-binding/restful-service-binding](https://blazor.syncfusion.com/documentation/common/data-binding/restful-service-binding)
2. [www.codeproject.com/Articles/5348568/Entity-Framework-Fundamentals](https://www.codeproject.com/Articles/5348568/Entity-Framework-Fundamentals)
3. [github.com/caboosecodes/TTA\_Basic\_C\_Sharp\_projects](https://github.com/caboosecodes/TTA_Basic_C_Sharp_projects)
4. [stackoverflow.com/questions/53573209/cant-connect-to-local-database-when-using-connection-string](https://stackoverflow.com/questions/53573209/cant-connect-to-local-database-when-using-connection-string)