

ASP.NET CORE Identity

Mme. Fatima-Ezzahra AIT BENNACER

f.aitbennacer@emsi.ma

2025 - 2026

Plan du cours

- ① Introduction
- ② Authentification Vs Autorisation
- ③ L'architecture d'ASP.NET Core Identity
- ④ Les étapes de création



ASP.NET Core Identity

Introduction



ASP.NET Core Identity

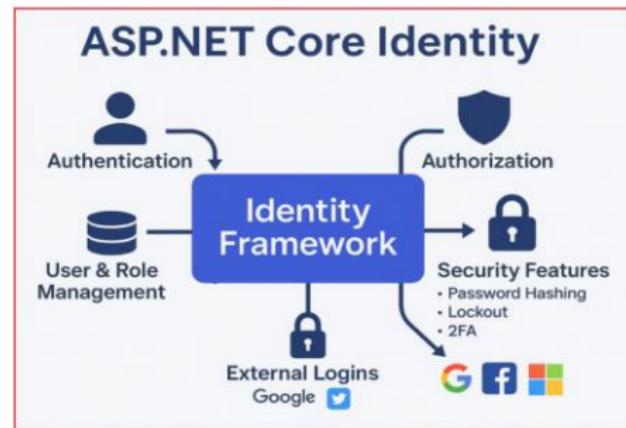
■ ASP.NET Core Identity:

- Un système de gestion d'adhésion qui permet d'ajouter des fonctionnalités de connexion à votre application. Il gère l'authentification (qui êtes-vous ?) et l'autorisation (que pouvez-vous faire ?).
- Il est indispensable de pouvoir :
 - Identifier les utilisateurs qui accèdent à l'application,
 - Contrôler ce qu'ils ont le droit de faire.
- ASP.NET Core fournit un framework robuste et extensible appelé **ASP.NET Core Identity**, destiné à gérer l'authentification et l'autorisation des utilisateurs de manière sécurisée et standardisée.

ASP.NET Core Identity

■ ASP.NET Core Identity repose sur:

- Entity Framework Core,
- Une base de données relationnelle,
- Des mécanismes intégrés de sécurité (hashage des mots de passe, cookies sécurisés, tokens, etc.).



ASP.NET Core Identity

■ Authentification vs Autorisation

- **Authentification** : Processus de vérification de l'identité d'un utilisateur. Exemple : Connexion avec email et mot de passe
- **Autorisation** : Processus de détermination des droits d'accès
Exemple : Vérifier si l'utilisateur peut accéder à une page admin



ASP.NET Core Identity Architecture

Architecture d'ASP.NET Core Identity

■ ASP.NET Core Identity est organisé en trois couches principales :

1. **Services Identity de Base (Core Identity Services)** : Les trois services principaux qui orchestrent toutes les opérations Identity

➤ **UserManager<TUser> - Gestionnaire des utilisateurs:**

- Création, modification, suppression d'utilisateurs
- Gestion des mots de passe (hachage, validation, réinitialisation)
- Attribution et retrait de rôles
- Gestion des claims (revendications) des utilisateurs
- Verrouillage de compte après tentatives échouées



Architecture d'ASP.NET Core Identity

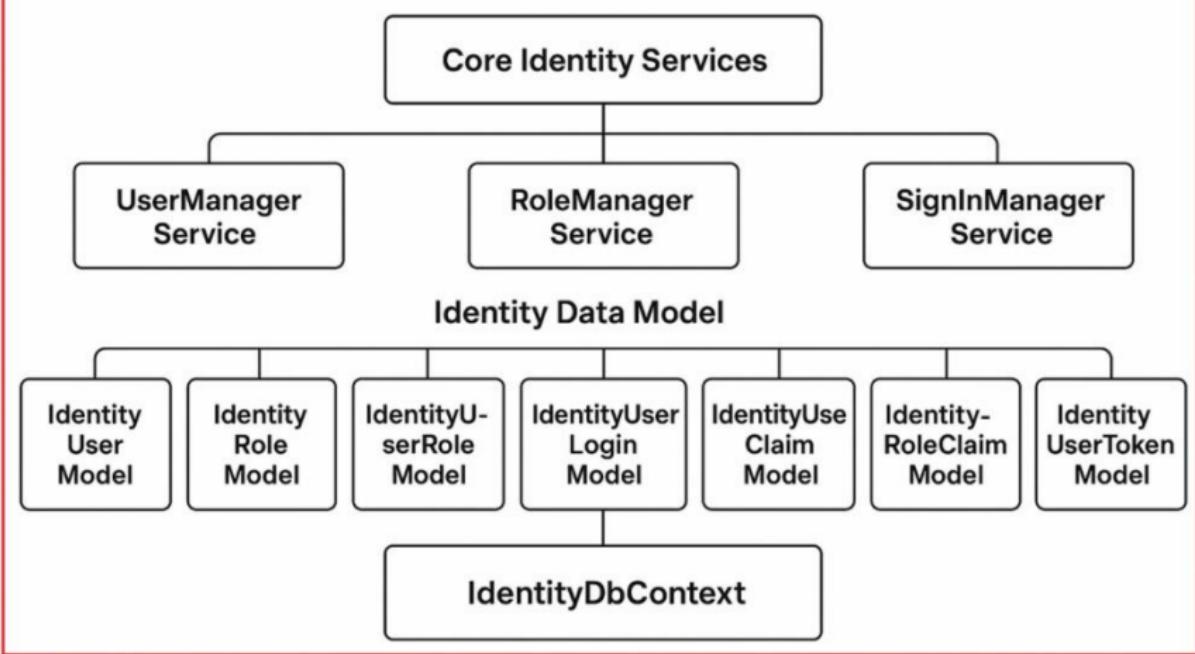
■ ASP.NET Core Identity est organisé en trois couches principales :

- **RoleManager<TRole> - Gestionnaire des rôles:**
 - Création, modification, suppression de rôles
 - Vérification de l'existence des rôles
 - Gestion des claims associés aux rôles
- **SignInManager<TUser> - Gestionnaire de connexion:**
 - Connexion et déconnexion des utilisateurs
 - Validation des identifiants (email/mot de passe)
 - Gestion de l'authentification à deux facteurs (2FA)
 - Gestion de la persistance des sessions (cookies)



Architecture d'ASP.NET Core Identity

Key Components of ASP.NET Core Identity





ASP.NET Core Identity

Étapes



ASP.NET Core Identity : étapes

1. Installation et Configuration:

Packages NuGet :

- **Microsoft.AspNetCore.Identity.EntityFrameworkCore**
- **Microsoft.AspNetCore.Identity.UI**
- Microsoft.EntityFrameworkCore.SqlServer
- Microsoft.EntityFrameworkCore.Tools

2. La création de Identity Database Context

- IdentityDbContext est une classe fournie par ASP.NET Core Identity qui :
 - hérite de **DbContext**, contient tous les DbSet nécessaires à Identity:
 - AspNetUsers, AspNetRoles, AspNetUserRoles, AspNetUserClaims, etc.



ASP.NET Core Identity : étapes

3. Créer la classe AuthDbContext.cs:

```
namespace Web_App_Migration.Data
{
    public class AuthDbContext : IdentityDbContext
    {
        public AuthDbContext(DbContextOptions<AuthDbContext>
options)
            : base(options)
        {
        }
    }
}
```

- Ce contexte sera responsable uniquement des données d'authentification.



ASP.NET Core Identity : étapes

4. Configuration des chaînes de connexion

Principe : séparation des bases

- BD_VENTE_MIG : Données Métier
- BD_VENTE_AUTH : Utilisateurs, rôles, sécurité

➤ Fichier AppSettings.json

```
"ConnectionStrings": {  
    "VenteDb": "Data Source=YOUR_SERVER;Initial  
Catalog=DB_VENTE_MIG;Integrated Security=True;TrustServerCertificate=True",  
  
    "AuthDb": "Data Source=YOUR_SERVER;Initial  
Catalog=BD_VENTE_AUTH;Integrated Security=True;TrustServerCertificate=True"  
}
```



ASP.NET Core Identity : étapes

5. Configuration d'Identity et EF Core dans Program.cs

- Enregistrement des DbContext

```
// DB Métier  
builder.Services.AddDbContext<VenteContext>(options =>  
  
    options.UseSqlServer(builder.Configuration.GetConnectionString("VenteDb"))  
);  
  
// DB Auth  
builder.Services.AddDbContext<AuthDbContext>(options =>  
  
    options.UseSqlServer(builder.Configuration.GetConnectionString("AuthDb")));
```



ASP.NET Core Identity : étapes

5. Configuration d'Identity et EF Core dans Program.cs

- Enregistrement des services Identity

```
builder.Services.AddIdentity<IdentityUser, IdentityRole>()
    .AddEntityFrameworkStores<AuthDbContext>()
    .AddDefaultTokenProviders();
```

Pourquoi **<IdentityUser, IdentityRole>**:

- IdentityUser : classe utilisateur par défaut
- IdentityRole : classe rôle par défaut



ASP.NET Core Identity : étapes

5. Configuration d'Identity et EF Core dans Program.cs

- Activation des Razor Pages Identity: Les pages Login/Register sont des **Razor Pages**.

```
builder.Services.AddRazorPages();
```

```
app.MapRazorPages();
```

Sans ces lignes :

/Identity/Account/Login ne fonctionne pas

/Identity/Account/Register retourne 404



ASP.NET Core Identity : étapes

5. Configuration d'Identity et EF Core dans Program.cs

- Middleware d'authentification et autorisation

```
app.UseAuthentication(); // identification  
app.UseAuthorization(); // contrôle d'accès
```

NB: L'ordre est obligatoire:

```
app.UseAuthentication();  
app.UseAuthorization();  
app.MapControllerRoute();
```



ASP.NET Core Identity : étapes

6. Création de la base Identity (Migration)

- Génération de la migration: Dans la console PM

```
Add-Migration IdentityInitial -Context AuthDbContext
```

- Application de la migration

```
Update-Database -Context AuthDbContext
```

BD_VENTE_AUTH
Diagrammes de base de données
Tables
Tables système
FileTables
Tables externes
Tables de graphe
dbo__EFMigrationsHistory
dbo.AspNetRoleClaims
dbo.AspNetRoles
dbo.AspNetUserClaims
dbo.AspNetUserLogins
dbo.AspNetUserRoles
dbo.AspNetUsers
dbo.AspNetUserTokens
Vues
Ressources externes
Synonymes
Programmabilité



ASP.NET Core Identity : étapes

7. Intégration du *Login / Logout* dans l'interface

The screenshot shows a web browser window with the title "Log in - Web_App_Migration". The address bar displays "localhost:7154/Identity/Account/Login?ReturnUrl=%2F". The page content is the "Log in" view of the ASP.NET Core Identity system. It features two main sections: "Use a local account to log in." and "Use another service to log in.". The "Use a local account to log in." section contains input fields for "Email" (containing "f.aitbennacer@emsi.ma") and "Password" (containing masked text). There is also a "Remember me?" checkbox and a large blue "Log in" button. Below the "Log in" button are links for "Forgot your password?", "Register as a new user", and "Resend email confirmation". The "Use another service to log in." section contains a message stating there are no external authentication services configured, with a link to an article about setting up external services.

TP N° 11