Programmation Web Transactionnelle

MS Identity

Autorisation

La sécurité: Autorisations

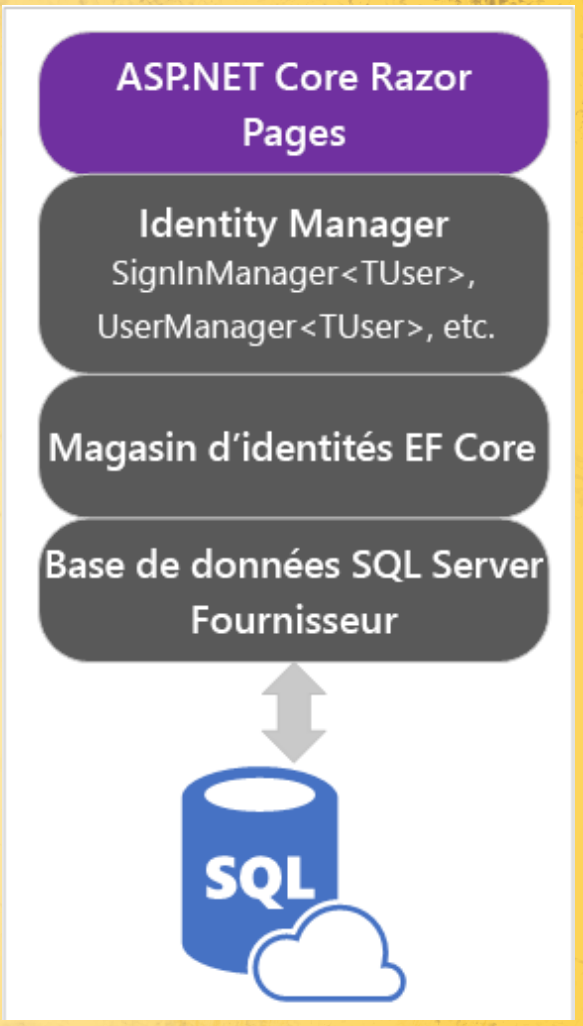
- Rendre accessible certaines parties de l'application/pages
- Basées sur des **rôles** et des **revendications** (*claims*) ou **caractéristiques** de l'utilisateur
- Les **privilèges** doivent être octroyés via une interface réservée aux administrateurs.

MS Identity: une librairie qui…

- Contient les informations et code requis pour la connexion
- Contient les modèles, views et appels => créer, modifier et identifier les utilisateurs
- Contient l'implantation de l'attribut [Authorize] permettant de sécuriser un contrôleur ou des actions
  - Valider l'utilisateur et les rôles associés

Architecture MS Identity

- gestionnaires (ou des **managers**): repositories pour les modèles contenu.
- **UserManager** : chercher, créer, modifier, supprimer des utilisateurs
- **SignInManager** : connexion, la création/suppression de cookie, etc.
- **RoleManager** chercher, créer, modifier, supprimer des rôles.




Autorisations
Rôles
Claims
policies

On évite les « chaines magiques hardcodées »

Dans projet type bibliothèque *Utility*

```
public static class AppConstants
{
    …
    // Rôles pour Autorisations
    public static string AdminRole = "Admin";
    public static string CustomerRole = "Customer";
    public static string AuthorRole = "Author";
}
```

Ajouter les rôles, les utilisateurs par défaut et sécuriser les actions

Dans votre Program.cs, section configure(…)

Ajouter les managers requis en paramètre :

UserManager<ApplicationUser> userManager,

RoleManager< ApplicationUser > roleManager

Créez vos rôles :

```
if (!_roleManager.RoleExistsAsync(AppConstants.AdminRole).GetAwaiter().GetResult())
{
    _roleManager.CreateAsync(new IdentityRole(AppConstants.AdminRole)).GetAwaiter().GetResult();
    _roleManager.CreateAsync(new IdentityRole(AppConstants.HunterRole)).GetAwaiter().GetResult();
    _roleManager.CreateAsync(new IdentityRole(AppConstants.PlayerRole)).GetAwaiter().GetResult();
}
```

Ajouter les rôles, les utilisateurs par défaut et sécuriser les actions

Créer vos utilisateurs :

```
// Créer un User pour le rôle Admin
_userManager.CreateAsync(new ApplicationUser {
    UserName = "VotreNom@ZombieParty.com",
    Email = "VotreNom@ZombieParty.com",
    NickName = "Votre surnom",
    PhoneNumber = "1111111111",
    EmailConfirmed = true,
}, "Admin123*").GetAwaiter().GetResult();

ApplicationUser user = _db.ApplicationUser.FirstOrDefault(u => u.Email ==
"VotreNom@ZombieParty.com");
_userManager.AddToRoleAsync(user, AppConstants.AdminRole).GetAwaiter().GetResult();
```

Ajouter les rôles, les utilisateurs par défaut et sécuriser les actions

Pour sécuriser une action on ajoute l'attribut Authorize, tout simplement.

Vous pouvez le mettre au niveau du contrôleur.

```
[Authorize(Roles = AppConstants.UtilisateurRole)]
public class HomeController : Controller
```

Vous pouvez le mettre au niveau de l'action (même chose mais sur la méthode!)

Vous pouvez mettre plusieurs rôles en utilisant la virgule : "Utilisateur,Administrateur,PeutEditer"

Sécuriser les actions que fait le [Authorize]

- Vérifier avant de faire l'appel si l'utilisateur est connecté à l'aide du cookie
- Si le cookie est invalide, il va vous rediriger vers la page de connexion.
- Dans la vue et le contrôleur, vous avez `this.User.Identity` qui permet d'avoir les informations contenues dans le cookie. À savoir, le username, les rôles, les claims et s'il est authentifié.

```
@if(this.User.IsInRole(''Administrateur''))
```

Limiter l'accès: *Controller*

Seuls Admin et Employee ont accès aux méthodes du *contrôleur*

```
[Authorize(Roles = SD.Role_Admin + "," + SD.Role_Employee)]
public class CompanyController : Controller
{
    private readonly IUnitOfWork _unitOfWork;
```

Sauf pour cette méthode: tout le monde a accès

```
[AllowAnonymous]
public IActionResult Index()
{
    return View();
}
```

Limiter l'accès: *Controller*

Seuls les Managers ont accès aux méthodes du *contrôleur*

```
[Authorize(Roles = SD.Role_Manager)]
public class CompanyController : Controller
{
    private readonly IUnitOfWork _unitOfWork;
```

Sauf pour cette méthode: doit être un admin

```
[Authorize(Roles = SD.Role_Admin)]
public IActionResult Index()
{
    return View();
}
```