



Autorisations et rôles

420-KYA-JQ
Interfaces et base de données

Gestion des accès

- 
- Afin de gérer les accès aux pages ainsi que l'authentification des utilisateurs, nous allons créer :
 - Des rôles ;
 - Un fournisseur d'autorisations personnalisé ;

Qu'est-ce qu'un fournisseur d'autorisation ?



C'est un service (pas un contrôleur) intégré à Blazor qui utilise le `HttpContext.User` de ASP.NET Core ;



Ce service permet de créer et gérer des rôles ainsi que de donner des accès à certains rôles spécifiquement.

Comment créer un fournisseur d'accès ?

- Dans votre dossier « **Authentication** », créer une classe « **CustomAuthenticationStateProvider** » qui hérite de « **AuthenticationStateProvider** » ;
- Dans votre program.cs, ajouter un service « **AddScoped** », le type de ce service sera « **ProtectedSessionStorage** » ;
- Toujours dans votre program.cs, ajouter un second service « **AddScoped** ». Le type de ce service sera « **AuthenticationStateProvider**, **CustomAuthenticationStateProvider** » ;
- Encore une fois dans program.cs, ajouter un service « **AddAuthenticationCore** ».

Comment créer un fournisseur d'accès ? (suite)

- Dans le dossier « Authentication », créer une classe « UserSession » ;
 - Cette classe correspond à un modèle, elle va contenir TOUTES les informations que vous voulez garder sur l'utilisateur connecté ;
 - Cette classe devrait contenir AU MOINS le nom d'utilisateur et le rôle de la personne connectée ;
 - N'hésitez pas à utiliser de l'héritage si nécessaire



Comment créer un fournisseur d'accès ? (suite)

- De retour à votre classe « [*CustomAuthenticationStateProvider*](#) » ;
 - Créer un attribut **privé** et **readonly** de type « [*ProtectedSessionStorage*](#) » (ça permet d'accéder à la session, un peu comme en PHP);
 - Créer un attribut privé de type « [*ClaimsPrincipal*](#) » et donner lui tout de suite comme valeur un nouveau « [*ClaimsPrincipal*](#) » avec comme paramètre un nouveau « [*ClaimsIdentity*](#) » (ce sera la valeur par défaut d'un utilisateur pas encore connecté)
 - Créer un constructeur qui reçoit en paramètre un « [*ProtectedSessionStorage*](#) » et qui l'assigne à notre attribut.

Comment créer un fournisseur d'accès ? (suite)

- ClaimsPrincipal : Permet de gérer une collection d'identité (ClaimsIdentity)
- Le fournisseur d'accès aura besoin de deux méthodes :

```
public override async Task<AuthenticationState> GetAuthenticationStateAsync()...
```

0 références

```
public async Task UpdateAuthenticationState(UserSession userSession)...
```

GetAuthenticationStateAsync

```
public override async Task<AuthenticationState> GetAuthenticationStateAsync()
{
    var claimsPrincipal = _anonymous;
    try
    {
        var userSessionStorageResult =
            await _sessionStorage.GetAsync<UserSession>("UserSession");
        var userSession =
            userSessionStorageResult.Success ? userSessionStorageResult.Value : null;
        if (userSession != null)
        {
            claimsPrincipal =
                new ClaimsPrincipal(new ClaimsIdentity(new List<Claim>
                {
                    new Claim(ClaimTypes.Name, userSession.UserName),
                    new Claim(ClaimTypes.Role, userSession.Role)
                }, "CustomAuth"));
        }
    }
    catch
    {
        claimsPrincipal = _anonymous;
    }

    return await Task.FromResult(new AuthenticationState(claimsPrincipal));
}
```

UpdateAuthenticationState

```
public async Task UpdateAuthenticationState(UserSession userSession)
{
    ClaimsPrincipal claimsPrincipal;

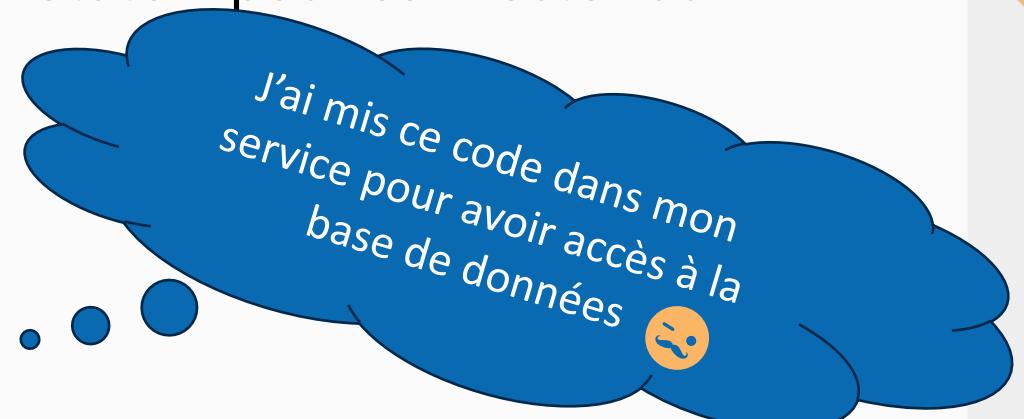
    if(userSession != null)
    {
        await _sessionStorage.SetAsync("UserSession", userSession);
        claimsPrincipal =
            new ClaimsPrincipal(new ClaimsIdentity(new List<Claim>
            {
                new Claim(ClaimTypes.Name, userSession.UserName),
                new Claim(ClaimTypes.Role, userSession.Role)
            }, "CustomAuth"));
    }
    else
    {
        await _sessionStorage.DeleteAsync("UserSession");
        claimsPrincipal = _anonymous;
    }

    NotifyAuthenticationStateChanged(Task.FromResult(new AuthenticationState(claimsPrincipal)));
}
```

Comment utiliser le fournisseur d'accès ?

- Dans votre view qui gère la connexion :
 - Injecter le service approprié ;
 - Injecter votre fournisseur d'accès ;
 - Utiliser la méthode « UpdateAuthenticationState » pour connecter ou déconnecter l'utilisateur ;

```
await authProvider.UpdateAuthenticationState(new UserSession
{
    UserName = loggedUser.Email,
    Role = loggedUser.Role
});
```



PSSST : si vous injecter un NavigationManager dans votre page, vous pouvez l'utiliser pour faire des redirections (ex: `navManager.NavigateTo("/", true);`)

Comment gérer mes rôles ?

- Dans votre Routes.razor, on va ajouter une balise « CascadingAuthenticationState » qui va englober tout et modifier le « RouteView »

```
<CascadingAuthenticationState> ←
  <Router AppAssembly="@typeof(App).Assembly">
    <Found Context="routeData">
      → <AuthorizeRouteView RouteData="@routeData" DefaultLayout="@typeof(MainLayout)" />
        <FocusOnNavigate RouteData="@routeData" Selector="h1" />
    </Found>
    <NotFound>
      <PageTitle>Not found</PageTitle>
      <LayoutView Layout="@typeof(MainLayout)">
        <p role="alert">Sorry, there's nothing at this address.</p>
      </LayoutView>
    </NotFound>
  </Router>
</CascadingAuthenticationState>
```

- Ajouter `@using Microsoft.AspNetCore.Components.Authorization`

Comment gérer mes rôles ? (suite)

- Dans vos views, dans la section @code, il faut ajouter un paramètre :

```
[CascadingParameter]
private Task<AuthenticationState> authenticationState { get; set; }
```

- On peut ensuite l'utiliser comme suit :

```
var authState = await authenticationState;
var name = authState.User.Identity.Name;
```

- Dans vos views, les balises qui varient selon les rôles devront être mises dans une balise AuthorizeView :

```
<AuthorizeView></AuthorizeView>
```

AuthorizedView

- Cette balise peut recevoir le paramètre Roles avec comme valeur, en chaîne de caractères, chacun des rôles permis séparé par une virgule :

```
<AuthorizeView></AuthorizeView>
<AuthorizeView Roles="Etu,Prof"></AuthorizeView>
<AuthorizeView Roles="Admin"></AuthorizeView>
```

AuthorizeView (suite)

- Cette balise vous donne accès à deux balises spéciales :

```
<Authorized></Authorized>  
<NotAuthorized></NotAuthorized>
```

- Authorized : ce que ça affiche si l'utilisateur est connecté ET respecte les rôles spécifiés ;
- NotAuthorized : ce que ça afficher si l'utilisateur n'est pas connecté OU ne respecte pas les rôles spécifiés.

Exemple

```
<div class="@NavControllerCssClass" @onClick="ToggleNavMenu">
    <nav class="flex-column">
        <AuthorizeView Roles="Prof, Etu">
            <Authorized>
                <div class="nav-item px-3">
                    <NavLink class="nav-link" href="Horaire">
                        <span class="oi oi-plus" aria-hidden="true"></span> Mon horaire
                    </NavLink>
                </div>
                <div class="nav-item px-3">
                    <NavLink class="nav-link" href="Cours">
                        <span class="oi oi-plus" aria-hidden="true"></span> Mes cours
                    </NavLink>
                </div>
            </Authorized>
        </AuthorizeView>
        <AuthorizeView Roles="Admin">
            <Authorized>
                <div class="nav-item px-3">
                    <NavLink class="nav-link" href="GestUser">
                        <span class="oi oi-plus" aria-hidden="true"></span> Gérer les utilisateurs
                    </NavLink>
                </div>
                <div class="nav-item px-3">
                    <NavLink class="nav-link" href="GestCours">
                        <span class="oi oi-plus" aria-hidden="true"></span> Gérer les cours
                    </NavLink>
                </div>
            </Authorized>
        </AuthorizeView>
    </nav>
</div>
```

Toujours amener à la connexion

```
<CascadingAuthenticationState>
    <Router AppAssembly="@typeof(App).Assembly">
        <Found Context="routeData">
            <AuthorizeRouteView RouteData="@routeData" DefaultLayout="@typeof(MainLayout)" >
                <NotAuthorized><LoginComponent /></NotAuthorized>
            </AuthorizeRouteView>
            <FocusOnNavigate RouteData="@routeData" Selector="h1" />
        </Found>
        <NotFound>
            <PageTitle>Not found</PageTitle>
            <LayoutView Layout="@typeof(MainLayout)">
                <p role="alert">Sorry, there's nothing at this address.</p>
            </LayoutView>
        </NotFound>
    </Router>
</CascadingAuthenticationState>
```

Vidéo

- Si jamais vous voulez plus d'information ou une démonstration :
 - [Vidéo](#)