Angular :

Utiliser Oauth2 et Keycloak

Objectifs :

1. Télécharger le projet de départ
2. Installer le module keycloak-angular
3. Modifier le composant app.component
4. Configurer le provider *provideKeycloakAngular*
5. Protection par les rôles

Télécharger le projet de départ

https://github.com/nadhemBelHadj/Angular19_keycloack_depart

Installer le module keycloak-angular

https://www.npmjs.com/package/keycloak-angular

npm install keycloak-angular keycloak-js

Modifier le composant app.component

Le fichier app.component.html

```
 <ul class="navbar-nav ms-auto m-2" >
        <li class="nav-item dropdown">
          <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown"
role="button" data-bs-toggle="dropdown" aria-expanded="false">
            Connexion
          </a>
          <ul class="dropdown-menu" aria-labelledby="navbarDropdown">
            @if (!authenticated) {
               <button class="action-item" (click)="login()">Login</button>
                } @else {
                 <button class="action-item" (click)="logout()">Logout</button>
                 }

          </ul>
        </li>
      </ul>
```

Le fichier app.component.ts

```
import { RouterLink, RouterOutlet } from '@angular/router';
import { Component, effect, inject } from '@angular/core';
import { RouterModule } from '@angular/router';
import Keycloak, { KeycloakProfile } from 'keycloak-js';

import {
  HasRolesDirective,
  KEYCLOAK_EVENT_SIGNAL,
  KeycloakEventType,
  typeEventArgs,
  ReadyArgs
} from 'keycloak-angular';


@Component({
  selector: 'app-root',
  imports: [RouterOutlet,RouterLink],
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
export class AppComponent {
  public profile? : KeycloakProfile;
  authenticated = false;
  keycloakStatus: string | undefined;
```

```
  private readonly keycloak = inject(Keycloak);
  private readonly keycloakSignal = inject(KEYCLOAK_EVENT_SIGNAL);

  constructor() {
    effect(() => {
      const keycloakEvent = this.keycloakSignal();

      this.keycloakStatus = keycloakEvent.type;

      if (keycloakEvent.type === KeycloakEventType.Ready) {
        this.authenticated = typeEventArgs<ReadyArgs>(keycloakEvent.args);
      }

      if (keycloakEvent.type === KeycloakEventType.AuthLogout) {
        this.authenticated = false;
      }
    });
  }

  login() {
    this.keycloak.login();
  }

  logout() {
    this.keycloak.logout();
  }
}
```

Le fichier app.config.ts

```
import { ApplicationConfig, provideZoneChangeDetection } from '@angular/core';
import { provideRouter } from '@angular/router';
import { routes } from './app.routes';
import { provideHttpClient, withInterceptors } from '@angular/common/http';
import { includeBearerTokenInterceptor, provideKeycloak } from 'keycloak-
angular';
import { provideKeycloakAngular } from './keycloak.config';

export const appConfig: ApplicationConfig = {
  providers: [
    provideKeycloakAngular(),
    provideZoneChangeDetection({ eventCoalescing: true }),
    provideRouter(routes),
    provideHttpClient(withInterceptors([includeBearerTokenInterceptor])
),
]
```

```
};
```

Configurer le provider *provideKeycloakAngular*

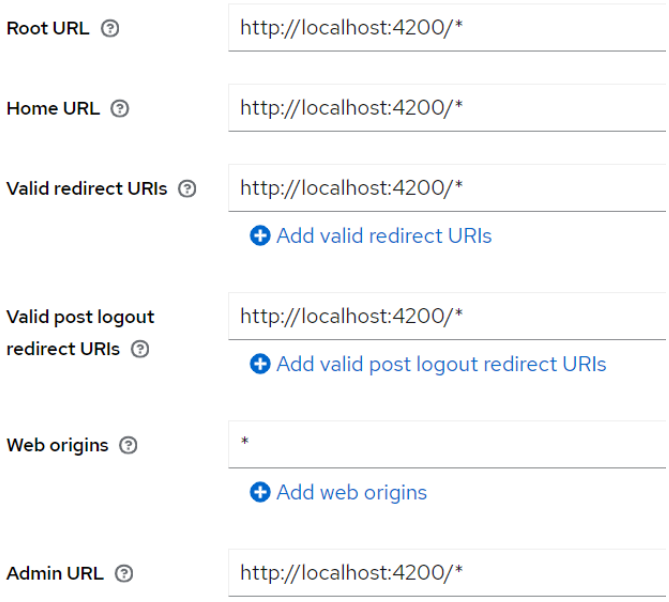
Le fichier keycloak.config.ts

```
import {
  provideKeycloak,
  createInterceptorCondition,
  IncludeBearerTokenCondition,
  INCLUDE_BEARER_TOKEN_INTERCEPTOR_CONFIG,
  withAutoRefreshToken,
  AutoRefreshTokenService,
  UserActivityService
} from 'keycloak-angular';

const localhostCondition =
createInterceptorCondition<IncludeBearerTokenCondition>({
  urlPattern: /^(http:\/\/localhost:8080)(\/.*)?$/i
});

export const provideKeycloakAngular = () =>
  provideKeycloak({
    config: {
        url: 'http://localhost:8090',
        realm: 'nadhem-realm',
        clientId: 'prod-app'
    },
    initOptions: {

         onLoad: 'login-required',

      /*onLoad: 'check-sso',
      silentCheckSsoRedirectUri: window.location.origin + '/silent-check-
sso.html',
      redirectUri: window.location.origin + '/'
      */
    },
    features: [
      withAutoRefreshToken({
        onInactivityTimeout: 'logout',
        sessionTimeout: 60000
      })
    ],
    providers: [
```

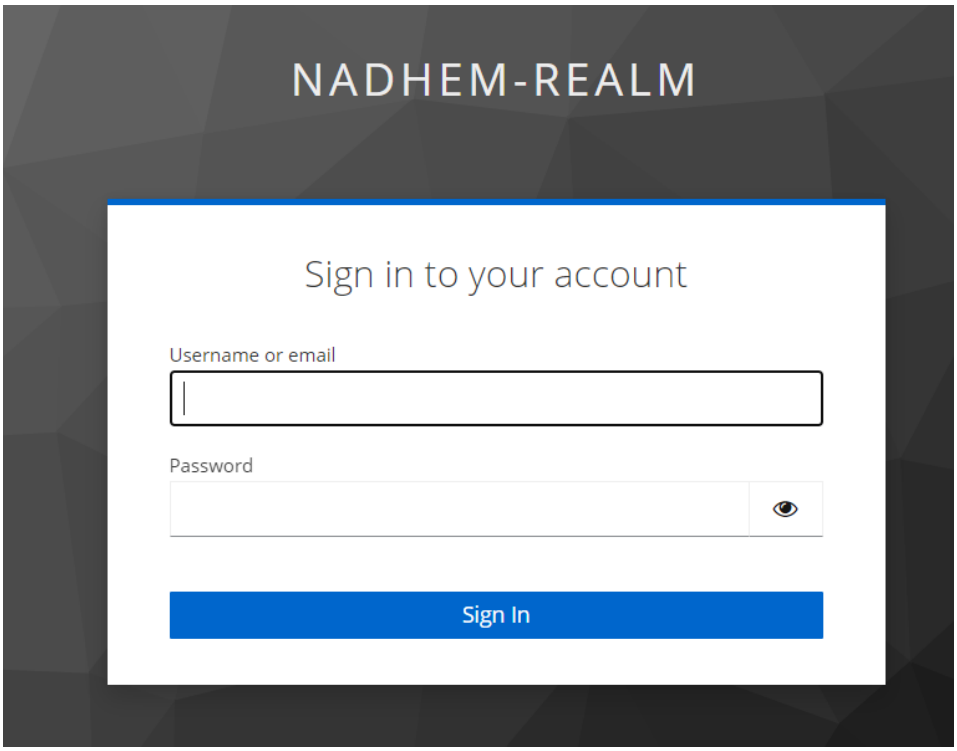
```
      AutoRefreshTokenService,
      UserActivityService,
      {
        provide: INCLUDE_BEARER_TOKEN_INTERCEPTOR_CONFIG,
        useValue: [localhostCondition]
      }
    ]
  });
```

Configuration de Keycloack

Client : prod-app /Access settings



Démarrer et tester



Création du comopsant user-profile :

ng g c user-profile

user.model.ts

```
export interface User {
  name: string;
  email?: string;
  username?: string;
}
```

user-profile.component.html :

```
<div class="section">
  <header>
    <h1>Welcome, {{ user?.name }}</h1>

  </header>

  <main>
    <section class="profile-card">
      <h2>Profile Details</h2>
      <p><strong>Name:</strong> {{ user?.name }}</p>
      <p><strong>Email:</strong> {{ user?.email }}</p>
```

```
      <p><strong>Username:</strong> {{ user?.username }}</p>
    </section>
  </main>
</div>
```

user-profile.component.ts

```
import { Component, OnInit } from '@angular/core';
import Keycloak from 'keycloak-js';
import { User } from '../model/user.model';

@Component({
  selector: 'app-user-profile',
  templateUrl: 'user-profile.component.html',
  styleUrls: [`user-profile.component.css`]
})
export class UserProfileComponent implements OnInit {
  user: User | undefined;

  constructor(private readonly keycloak: Keycloak) {}

  async ngOnInit() {
    if (this.keycloak?.authenticated) {
      const profile = await this.keycloak.loadUserProfile();

      this.user = {
        name: `${profile?.firstName} ${profile.lastName}`,
        email: profile?.email,
        username: profile?.username
      };

    }
  }
}
```

```
<ul class="navbar-nav ms-auto m-2" >
       <li> <a routerLink="/profile" class="nav-link dropdown-toggle">My
Profile</a>   </li>
```

```
  { path: 'profile', component: UserProfileComponent },
```

Protection par les rôles

Ne pas se connecter au démarrage :

Décommentez ce code

```
 /*onLoad: 'check-sso',
      silentCheckSsoRedirectUri: window.location.origin + '/silent-check-
sso.html',
      redirectUri: window.location.origin + '/'
      */
```

Créer le fichier silent-check-sso.html dans le dossier *public*

```
<html>
  <body>
    <script>
      parent.postMessage(location.href, location.origin);
    </script>
  </body>
</html>
```

Créer le composant forbidden :

```
<div class="section">
  <header>
    <h1>403 - Forbidden</h1>
  </header>

  <main>
    <p class="error-message">You do not have permission to access this
page.</p>
    <a class="home-link" routerLink="/">Go to Home</a>
  </main>
</div>
```

Créer le fichier app/guards/ auth-role.guard.ts :

```
import { AuthGuardData, createAuthGuard } from 'keycloak-angular';
import { ActivatedRouteSnapshot, CanActivateFn, Router, RouterStateSnapshot,
UrlTree } from '@angular/router';
import { inject } from '@angular/core';


const isAccessAllowed = async (
  route: ActivatedRouteSnapshot,
  __: RouterStateSnapshot,
  authData: AuthGuardData
```

```
): Promise<boolean | UrlTree> => {
  const { authenticated, grantedRoles } = authData;

  const requiredRole = route.data['role'];
  if (!requiredRole) {
    return false;
  }

  const hasRequiredRole = (role: string): boolean =>
    Object.values(grantedRoles.realmRoles).some((roles) =>
roles.includes(role));

  if (authenticated && hasRequiredRole(requiredRole)) {
    return true;
  }

  const router = inject(Router);
  return router.parseUrl('/forbidden');
};

export const canActivateAuthRole =
createAuthGuard<CanActivateFn>(isAccessAllowed);
```

Modifier le fichier app.routes.ts

```
import { Routes } from '@angular/router';
import { ProduitsComponent } from './produits/produits.component';
import { canActivateAuthRole } from './guards/auth-role.guard';
import { ForbiddenComponent } from './forbidden/forbidden.component';
import { UserProfileComponent } from './user-profile/user-profile.component';

export const routes: Routes = [

    {path: "produits",
    component : ProduitsComponent,
    canActivate: [canActivateAuthRole],
    data: { role: 'ADMIN' }
    },
     {
    path: 'profile',
    component: UserProfileComponent
   },
    { path: 'forbidden', component: ForbiddenComponent }

];
```

Testez l'accès à la liste des produits

Cacher la commande Lister pour les utilisateurs qui ne possèdent pas le rôle ADMIN :

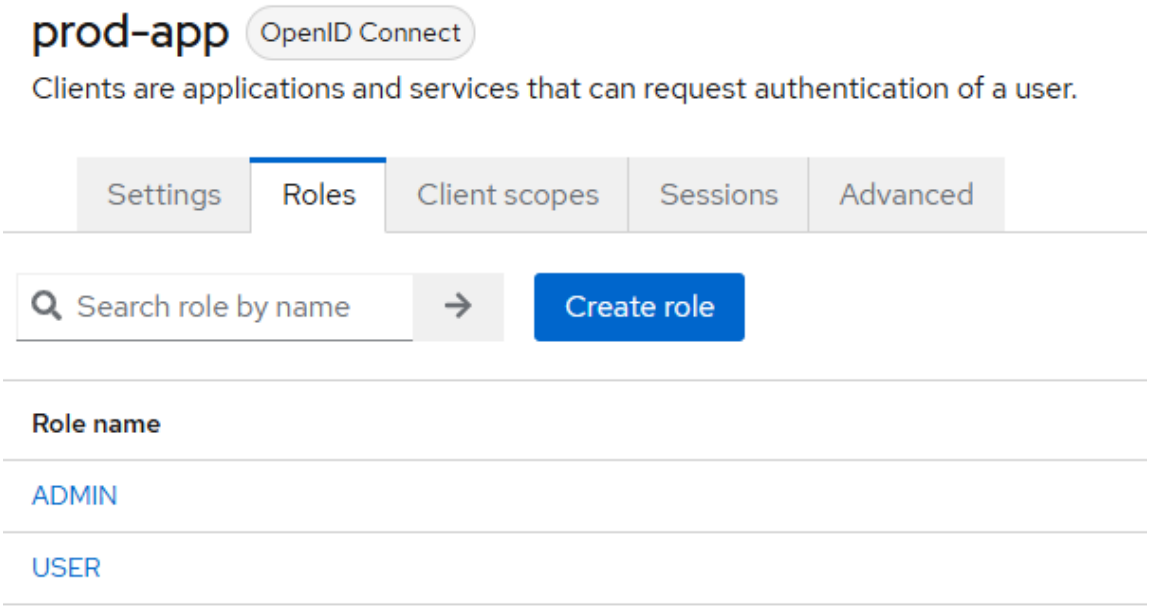
```
  <li><a class="dropdown-item"  routerLink="/produits"
*kaHasRoles="['ADMIN']; checkRealm:true " >Lister</a></li>
```

```
  imports: [RouterOutlet,RouterLink,HasRolesDirective],
```

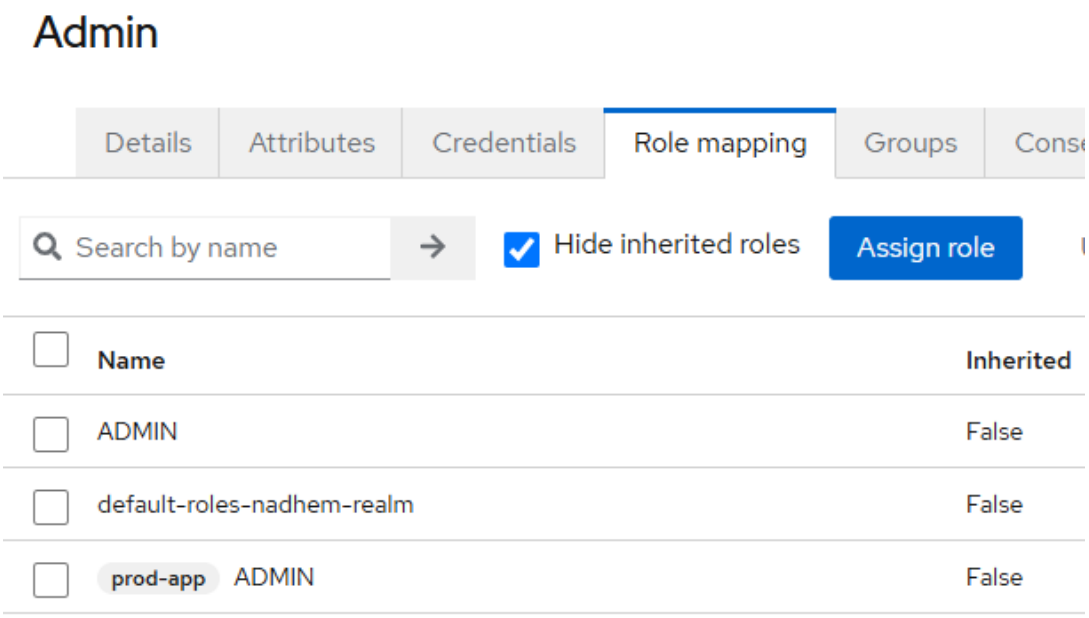
Les rôles du Realm Vs les rôles du Client

Créer les rôles du client 'prod-app'

Créer au niveau de Keycloak les rôles du client 'prod-app' : ADMIN, USER



Affecter ADMIN à l'utilisateur admin



Tester sans login est-il possible d'afficher la liste des produits ?

auth-role.guard.ts

```
  const hasRequiredRole = (role: string): boolean =>
    Object.values(grantedRoles.resourceRoles).some((roles) =>
roles.includes(role));
```

```
  <li><a class="dropdown-item"  routerLink="/produits"
 *kaHasRoles="['ADMIN']" >Lister</a></li>
```