

# Consommation avec Angular des api sécurisés JWT

L'objectif général de cet atelier est d'apprendre comment consommer à partir de Angular les api sécurisés qu'on avait développés dans les ateliers précédents.

A la fin de l'atelier on aura un projet Angular qui utilise l'authentification JWT et consomme les api REST sécurisés.

## Objectifs :

1. Récupérer le token JWT à la suite d'un login,
2. Inclure le token JWT dans l'appel de l'api /api/all,
3. Afficher la page login seulement lorsque l'utilisateur n'est pas connecté,
4. Ajouter l'authentification JWT aux autres opérations CRUD.

## Récupérer le token JWT à la suite d'un login

Ici on va apporter les modifications nécessaires au projet Angular de départ pour réaliser l'authentification par l'obtention du token JWT en utilisant l'api REST :

<http://localhost:8081/users/login>

1. Installer le module *auth0/angular-jwt* qui permettra de decoder le token JWT

```
npm install @auth0/angular-jwt
```

2. Modifier la classe **AuthService** en ajoutant les méthodes suivantes :

```
apiURL: string = 'http://localhost:8081/users';
token!:string;

constructor(private router: Router,
             private http : HttpClient) { }

login(user : User)
{
    return this.http.post<User>(this.apiURL+'/login', user , {observe:'response'});
}

saveToken(jwt:string){
    localStorage.setItem('jwt',jwt);
    this.token = jwt;
    this.isloggedIn = true;
}
```

```
loadToken() {
  this.token = localStorage.getItem('jwt');
}
getToken():string {
  return this.token;
}
```

3. Modifier la méthode *onLoggedIn ()* de la classe *LoginComponent* :

```
err:number = 0;
```

La version avec la méthode *subscribe deprecated* :

```
onLoggedIn()
{
  this.authService.login(this.user).subscribe((data)=> {
    let jwtToken = data.headers.get('Authorization');
    this.authService.saveToken(jwtToken);
    this.router.navigate(['/']);
  },(erreur)=>{ this.err = 1;
  });
}
```

La version avec la méthode *subscribe modifiée*

```
onLoggedIn()
{
  this.authService.login(this.user).subscribe({
    next: (data) => {
      let jwtToken = data.headers.get('Authorization')!;
      this.authService.saveToken(jwtToken);
      this.router.navigate(['/']);
    },
    error: (err: any) => {
      this.err = 1;
    }
  });
}
```

#### 4. Ajouter ces lignes au fichier login.component.html

```
<div class="row justify-content-md-center">
  <div class="col-md-4">
    <div class="alert alert-danger" *ngIf="err==1">
      <strong>login ou mot de passe erronés..</strong>
    </div>
    <form >
```

## Modification à faire au niveau du projet Spring boot users-microservice

Si jamais vous avez rencontré le problème CORS, voici un rappel des modifications à faire au niveau du projet spring boot.

Dans la classe SecurityConfig, Modifiez la méthode `filterChain(HttpSecurity http)` comme suit :

```
.sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS).and()

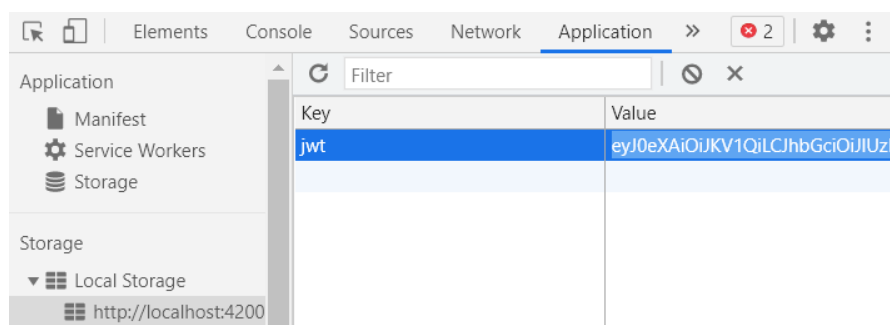
        .cors().configurationSource(new CorsConfigurationSource() {
            @Override
            public CorsConfiguration getCorsConfiguration(HttpServletRequest
request) {

                CorsConfiguration config = new CorsConfiguration();

                config.setAllowedOrigins(Collections.singletonList("http://localhost:4200"));
                config.setAllowedMethods(Collections.singletonList("*"));
                config.setAllowCredentials(true);
                config.setAllowedHeaders(Collections.singletonList("*"));
                config.setExposedHeaders(Arrays.asList("Authorization"));
                config.setMaxAge(3600L);
                return config;
            }
        }).and()
```

...

5. Tester la recuperation du token JWT à partir du microservice
6. Vérifier la sauvegarde du jwt dans LocalStorage (F12/Application)



## Inclure le jwt dans l'appel de l'api /api/all

7. Modifier la classe ProduitService pour inclure le token jwt dans les appels des api émis vers le microservice produits-api

```
apiURL: string = 'http://localhost:8080/produits/api';

listeProduit(): Observable<Produit[]>{
    let jwt = this.authService.getToken();
    jwt = "Bearer "+jwt;
    let httpHeaders = new HttpHeaders({"Authorization":jwt})
    return this.http.get<Produit[]>(this.apiURL+"/all",{headers:httpHeaders});
}
```

8. Modifier la méthode saveToken () de la classe AuthService pour extraire le nom utilisateur et ses rôles du token JWT

```
import { JwtHelperService } from '@auth0/angular-jwt';

private helper = new JwtHelperService();

saveToken(jwt:string){
    localStorage.setItem('jwt',jwt);
    this.token = jwt;
    this.isloggedIn = true;
    this.decodeJWT();
}

decodeJWT()
{
    if (this.token == undefined)
        return;
    const decodedToken = this.helper.decodeToken(this.token);
    this.roles = decodedToken.roles;
    this.loggedUser = decodedToken.sub;
}

loadToken() {
    this.token = localStorage.getItem('jwt')!;
    this.decodeJWT();
}
```

9. Modifier la méthode `isAdmin ()` de la classe `AuthService`

```
isAdmin():Boolean{
    if (!this.roles)
        return false;
    return this.roles.indexOf('ADMIN') >=0;
}
```

10. Modifier la méthode `logout ()` de la classe `AuthService`

```
logout() {
    this.loggedUser = undefined!;
    this.roles = undefined!;
    this.token= undefined!;
    this.isloggedIn = false;
    localStorage.removeItem('jwt');
    this.router.navigate(['/login']);
}
```

### Afficher la page login seulement lorsque l'utilisateur n'est pas connecté

11. Modifier la méthode `ngOnInit ()` de la classe `AppComponent`

```
ngOnInit () {
    this.authService.loadToken();
    if (this.authService.getToken()==null ||
        this.authService.isTokenExpired())
        this.router.navigate(['/login']);
}
```

12. Ajouter la méthode `isTokenExpired ()` de la classe `AuthService`

```
isTokenExpired(): Boolean
{
    return this.helper.isTokenExpired(this.token);
}
```

13. Tester un login puis affichez la liste de tous les produits

## Ajouter l'authentification JWT aux autres operations CRUD

14. Modifier la classe ProduitService pour inclure le token jwt dans les appels de toutes les api émis vers le microservice produits :

```
listeProduit(): Observable<Produit[]>{
    let jwt = this.authService.getToken();
    jwt = "Bearer "+jwt;
    let httpHeaders = new HttpHeaders({"Authorization":jwt})
    return this.http.get<Produit[]>(apiURL+"/all",{headers:httpHeaders});
}

ajouterProduit( prod: Produit):Observable<Produit>{
    let jwt = this.authService.getToken();
    jwt = "Bearer "+jwt;
    let httpHeaders = new HttpHeaders({"Authorization":jwt})
    return this.http.post<Produit>(apiURL+"/addprod", prod, {headers:httpHeaders});
}

supprimerProduit(id : number) {
    const url = `${apiURL}/delprod/${id}`;
    let jwt = this.authService.getToken();
    jwt = "Bearer "+jwt;
    let httpHeaders = new HttpHeaders({"Authorization":jwt})
    return this.http.delete(url, {headers:httpHeaders});
}

consulterProduit(id: number): Observable<Produit> {
    const url = `${apiURL}/getbyid/${id}`;
    let jwt = this.authService.getToken();
    jwt = "Bearer "+jwt;
    let httpHeaders = new HttpHeaders({"Authorization":jwt})
    return this.http.get<Produit>(url,{headers:httpHeaders});
}

updateProduit(prod :Produit) : Observable<Produit>    {
    let jwt = this.authService.getToken();
    jwt = "Bearer "+jwt;
    let httpHeaders = new HttpHeaders({"Authorization":jwt})
    return this.http.put<Produit>(apiURL+"/updateprod", prod, {headers:httpHeaders});
}
```

```

listeCategories():Observable<CategorieWrapper>{
    let jwt = this.authService.getToken();
    jwt = "Bearer "+jwt;
    let httpHeaders = new HttpHeaders({"Authorization":jwt})
    return this.http.get<CategorieWrapper>(this.apiUrlCat,{headers:httpHeaders}
);
    }

    rechercherParCategorie(idCat: number): Observable<Produit[]> {
        const url = `${apiURL}/prodscat/${idCat}`;
        return this.http.get<Produit[]>(url);
    }

    rechercherParNom(nom: string):Observable< Produit[]> {
        const url = `${apiURL}/prodsByName/${nom}`;
        return this.http.get<Produit[]>(url);
    }

    ajouterCategorie( cat: Categorie):Observable<Categorie>{
        return this.http.post<Categorie>(this.apiUrlCat, cat, httpOptions);
    }

```