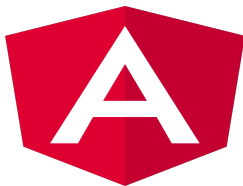


Angular : guard

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en programmation par contrainte (IA)
Ingénieur en génie logiciel

`elmouelhi.achref@gmail.com`



Plan

- 1 Introduction
- 2 Création
- 3 Exemple avec `CanActivate`
- 4 Exemple avec `CanDeactivate`

Angular

Guard ?

- Un service **Angular** (donc décoré par `@Injectable`) qui implémente une des interfaces suivantes
 - `CanActivate` : vérifie si un utilisateur peut visiter une route.
 - `CanDeactivate` : vérifie si un utilisateur peut quitter une route.
 - `CanActivateChild` : vérifie si un utilisateur peut visiter les routes enfants.
 - `CanLoad` : vérifie si un utilisateur peut aller sur une route d'un module défini avec un lazy loading

Angular

Exemple

- On veut que l'accès à la route `/adresse` soit seulement autorisé aux utilisateurs authentifiés
- On va créer une guard `auth` et l'associer à la route `adresse`

Angular

Pour créer une guard

```
ng generate guard guard-name
```

© Achref EL MOU

Angular

Pour créer une guard

```
ng generate guard guard-name
```

Ou le raccourci

```
ng g g guard-name
```

Angular

Pour notre exemple, on va créer une garde qui vérifie si un utilisateur est authentifié avant de charger certaines routes

```
ng g g guards/auth
```

© Achref EL MOUËZ

Angular

Pour notre exemple, on va créer une garde qui vérifie si un utilisateur est authentifié avant de charger certaines routes

```
ng g g guards/auth
```

Dans le menu qui s'affiche

- Pointer sur `CanActivate`
- Puis cliquer une première fois sur `espace` et une deuxième sur `entrée`

Angular

On peut aussi préciser dans la commande l'interface à implémenter

```
ng g g guards/auth --implements CanActivate
```

© Achref EL M...

Angular

On peut aussi préciser dans la commande l'interface à implémenter

```
ng g g guards/auth --implements CanActivate
```

Résultat

```
CREATE src/app/guards/auth.guard.spec.ts (346 bytes)  
CREATE src/app/guards/auth.guard.ts (456 bytes)
```

Contenu du auth.guard.ts

```
import { Injectable } from '@angular/core';
import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot,
  UrlTree } from '@angular/router';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class AuthGuard implements CanActivate {
  canActivate(
    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> |
    Promise<boolean | UrlTree> | boolean | UrlTree {
    return true;
  }
}
```

Contenu du `auth.guard.ts`

```
import { Injectable } from '@angular/core';
import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot,
  UrlTree } from '@angular/router';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class AuthGuard implements CanActivate {
  canActivate(
    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> |
    Promise<boolean | UrlTree> | boolean | UrlTree {
    return true;
  }
}
```

- `ActivatedRouteSnapshot` : contient des informations comme les paramètres envoyés pour la route demandée...
- `RouterStateSnapshot` : contient des informations comme l'URL de la route demandée
- La méthode `canActivate` ne fait aucun contrôle car elle retourne toujours `true`

Angular

Associons AuthGuard à la route /adresse dans
app-routing.module.ts

```
const routes: Routes = [  
  // les autres routes  
  {  
    path: 'adresse',  
    component: AdresseComponent,  
    canActivate: [AuthGuard]  
  },  
  { path: 'auth', component: AuthComponent },  
  // le reste des routes  
];
```

Angular

Associons `AuthGuard` à la route `/adresse` dans `app-routing.module.ts`

```
const routes: Routes = [  
  // les autres routes  
  {  
    path: 'adresse',  
    component: AdresseComponent,  
    canActivate: [AuthGuard]  
  },  
  { path: 'auth', component: AuthComponent },  
  // le reste des routes  
];
```

La route `/auth` va permettre d'exécuter le composant `AuthComponent` et afficher et gérer l'authentification

Angular

Contenu de auth.component.html

```
<h1>Page d'authentification</h1>
<form (ngSubmit)="isAuthenticated()">
  <div>
    Nom d'utilisateur :
    <input type="text" [(ngModel)]="login" name="login">
  </div>
  <div>
    Mot de passe :
    <input type="text" [(ngModel)]="password" name="password">
    <button type="submit">Se connecter</button>
  </div>
  <div [hidden]='erreur'>Identifiants incorrects</div>
</form>
```

Contenu de auth.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';

@Component({
  selector: 'app-auth',
  templateUrl: './auth.component.html',
  styleUrls: ['./auth.component.css']
})
export class AuthComponent implements OnInit {
  erreur = true;
  password = '';
  login = '';
  constructor(private router: Router) { }
  ngOnInit() { }

  isAuthenticated() {
    if (this.login === 'wick' && this.password === 'john') {
      localStorage.setItem('isConnected', 'true');
      this.router.navigateByUrl('/personne');
    } else {
      this.erreur = false;
    }
  }
}
```


Mettons à jour le contenu de `auth.guard.ts`

```
import { Injectable } from '@angular/core';
import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot,
  Router } from '@angular/router';

@Injectable({
  providedIn: 'root'
})
export class AuthGuard implements CanActivate {

  constructor(private router: Router) { }

  canActivate(
    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): boolean {
    if (Boolean(localStorage.getItem('isConnected'))) {
      return true;
    } else {
      this.router.navigateByUrl('/auth');
      return false;
    }
  }
}
```

Angular

Pour tester

- Essayez d'accéder à la route `/adresse` sans authentification
- Authentifiez-vous avec les identifiants `wick` et `john` et réessayez

Angular

Remarque

- Préciser l'interface à implémenter pendant la génération existe depuis **Angular 8**
- **Angular-cli** ne propose pas l'interface **CanDeactivate**
- On va spécifier l'interface **CanActivate** puis on modifie

© Achref EL MOUADJID

Angular

Remarque

- Préciser l'interface à implémenter pendant la génération existe depuis **Angular 8**
- **Angular-cli** ne propose pas l'interface **CanDeactivate**
- On va spécifier l'interface **CanActivate** puis on modifie

Exécutons la commande suivante

```
ng g g guards/leave
```

Angular

Remarque

- Préciser l'interface à implémenter pendant la génération existe depuis **Angular 8**
- **Angular-cli** ne propose pas l'interface **CanDeactivate**
- On va spécifier l'interface **CanActivate** puis on modifie

Exécutons la commande suivante

```
ng g g guards/leave
```

Le résultat

```
CREATE src/app/guards/leave.guard.spec.ts (352 bytes) CREATE  
src/app/guards/leave.guard.ts (472 bytes)
```

Code généré

```
@Injectable({
  providedIn: 'root'
})
export class LeaveGuard implements CanActivate {
  canActivate(
    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree>
    | Promise<boolean | UrlTree> | boolean | UrlTree {
    return true;
  }
}
```

© Achille

Code généré

```

@Injectable({
  providedIn: 'root'
})
export class LeaveGuard implements CanActivate {
  canActivate(
    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree>
    | Promise<boolean | UrlTree> | boolean | UrlTree {
    return true;
  }
}

```

Ensuite

- Remplacez `CanActivate` par `CanDeactivate<FormulaireComponent>`
- Faites l'import pour **CanDeactivate**
- Supprimer la méthode générée pour `CanActivate` et ajouter la méthode de l'interface qu'il faut implémenter

Nouveau code

```
@Injectable({
  providedIn: 'root'
})
export class LeaveGuard implements CanDeactivate<FormulaireComponent> {
  canDeactivate(component: FormulaireComponent,
    currentRoute: ActivatedRouteSnapshot,
    currentState: RouterStateSnapshot,
    nextState?: RouterStateSnapshot): boolean {
    throw new Error("Method not implemented.");
  }
}
```

© Achref EL

Nouveau code

```

@Injectable({
  providedIn: 'root'
})
export class LeaveGuard implements CanDeactivate<FormulaireComponent> {
  canDeactivate(component: FormulaireComponent,
    currentRoute: ActivatedRouteSnapshot,
    currentState: RouterStateSnapshot,
    nextState?: RouterStateSnapshot): boolean {
    throw new Error("Method not implemented.");
  }
}

```

Objectif

- On ajoute le lien suivant dans `formulaire.component.html` :

```
<a routerLink='/personne'>aller ailleurs</a>
```

- Si l'utilisateur remplit les deux champs `nom` et `prenom` dans `formulaire.component.html` et clique sur le lien pour aller sur le composant `personne`, on lui demande une confirmation.

Angular

Modifions la garde

```
@Injectable({
  providedIn: 'root'
})
export class LeaveGuard implements CanDeactivate<FormulaireComponent> {
  canDeactivate(component: FormulaireComponent,
    currentRoute: ActivatedRouteSnapshot,
    currentState: RouterStateSnapshot,
    nextState?: RouterStateSnapshot): boolean {
    return component.personne.nom === undefined ||
      component.personne.prenom === undefined ||
      component.personne.nom.length === 0 ||
      component.personne.prenom.length === 0 ||
      confirm('voulez-vous vraiment quitter ?');
  }
}
```

Angular

Associations LeaveGuard à la route /formulaire dans
app-routing.module.ts

```
const routes: Routes = [  
  // les autres routes  
  {  
    path: 'adresse',  
    component: AdresseComponent,  
    canActivate: [AuthGuard]  
  },  
  {  
    path: 'formulaire',  
    component: FormulaireComponent,  
    canDeactivate: [LeaveGuard]  
  },  
  { path: 'auth', component: AuthComponent },  
  // le reste des routes  
];
```

Angular

Il est possible d'associer plusieurs guards à une route

```
const routes: Routes = [  
  // les autres routes  
  {  
    path: 'adresse',  
    component: AdresseComponent,  
    canActivate: [AuthGuard]  
  },  
  {  
    path: 'formulaire',  
    component: FormulaireComponent,  
    canActivate: [AuthGuard],  
    canDeactivate: [LeaveGuard],  
  },  
  { path: 'auth', component: AuthComponent },  
  // le reste des routes  
];
```