Angular Guards

How to prevent unauthorized access?

Introduction

- You can defined so called **guards** for **Angular routes** to prevent unauthorized access
- There are many different kind of guards. The two most important are:
  1. **`CanActive`**
  2. **`CanDeactivate`**

Creating a guard

- You create a guard using the Angular CLI:

  ```
  ng generate guard core/Auth
  ```

- This generates a class which implements for example the **`CanActivate`** interface.
- The return value of the **`canActive()`** method defines if the page may be navigated to or not.

Example for a guard

- Let's say, we would like to use a **login page** before a certain page in our app can be accessed.
- We simulate this by storing some kind of **token** in the **sessionStorage**.

```
export class LoginComponent {
    login(): void {
        sessionStorage.setItem('token', 'abcdef');
    }
    logout(): void {
        sessionStorage.removeItem('token');
    }
}
```

Example for a guard (cont.)

- Implement the guard in **`auth.guards.ts`**

```
export class AuthGuard imlements CanActivate {
    constructor(private router: Router) {}

    canActivate(next: ActivatedRouteSnapshot, state:
    ↪  RouterStateSnapshot): boolean {
        const isOk = !!sessionStorage.getItem('token');
        if (!isOk) this.router.navigate(['/login']);
        return isOk;
    }
}
```

Example for a guard (cont.)

- Secure the protected routes:

```
...,
{ path: 'protected-page', component: ProtectedComponent, canActivate:
↪  [AuthGuard]},
...
```

Example for a guard (cont.)

- Automatically forward to the protected page.

```
canActivate(...): boolean {
    ...
    if (!isOk) this.router.navigate(['login', { originalUrl: next.url
    ↪  }]);
}
```

```
export class LoginComponent {
    constructor(private activatedRoute, private router: Router) {}

    login(): void {
        sessionStorage.setItem('token', 'abcdef');
        this.activatedRoute.paramMap.subscribe(x => {
            const url = x.get('originalUrl');
            this.router.navigate([url]);
        });
    }
}
```

CanDeactivate

- Use this guard to secure a page from **beeing left** (e.g. there are unsafed items).

```
export interface CheckSaveComponent {
    hasSaved(): () => boolean;
}

...
export class DeactivateGuard implements
↪   CanDeactivate<CheckSaveComponent> {
    canDeactivate(component: CheckSaveComponent, ...): boolean |
    ↪   UrlTree | Observable<boolean> {
        const isOk = component.hasSaved();
        return isOk;
    }
}
```

CanDeactivate (cont.)

- Component needs to implement the **CheckSaveComponent** interface.

```
export class PageOneComponent implements CheckSaveComponent {
    isSaved = false;
    save(): void {
```

```
        this.isSaved = true;
    }
    hasSaved(): boolean {
        return this.isSaved;
    }
}
```

- Secure the page as expected:

```
...,
{ path: 'page-one', component: PageOneComponent, canDeactivate:
↪  [DeactivateGuard] },
...
```