

ROUTE GUARDS

1. In traditional server side applications the application would check permissions on the server and return a 403 error page if the user didn't have permissions, or perhaps redirect them to a login/register page if they were not signed up.

403 is a HTTP error code specifically this one means Permission Denied

We want to have the same functionality in our client side SPA, and with Router Guards we can.

With Router Guards we can prevent users from accessing areas that they're not allowed to access, or, we can ask them for confirmation when leaving a certain area.

2. Different types of guards based on specific use cases.
 - Maybe the user must login (authenticate) first.
 - Perhaps the user has logged in but is not authorized to navigate to the target component.
 - We might ask the user if it's OK to discard pending changes rather than save them.There are four different types of Guards:

- 2.1. **CanActivate**

Checks to see if a user can visit a route.

- 2.2. **CanActivateChild**

Checks to see if a user can visit a routes children.

- 2.3. **CanDeactivate**

Checks to see if a user can exit a route.

- 2.4. **Resolve**

Performs route data retrieval before route activation.

- 2.5. **CanLoad**

Checks to see if a user can route to a module that lazy loaded.

STEPS to implement guards:

1. Add angular material as follows and select the defaults when prompted with various options:

```
ng add @angular/material
```

2. Next in app module add below material related modules in imports[]:

```
MatButtonModule, MatCardModule, MatInputModule
```

3. Create a service within service folder as follows:

```
ng g s user
```

4. Update the userservice with following code:

```
url:string ="http://localhost:8080/";
constructor(private http:HttpClient) { }

loginUser(email:string, password:string){
  let emp ={email, password}
  return this.http.post<any>(this.url+"login", emp);
}

isAuthenticated():boolean{
  return !!localStorage.getItem("token");
}

logout(){
  localStorage.removeItem("token");
}
```

```
}
```

5. Create login component and update the ts , css and html as follows:

ng g c login

HTML

```
<mat-card>
  <mat-card-content>
    <form #loginForm="ngForm" (ngSubmit)="onSubmit()">
      <h2>Log In</h2>
      <mat-error *ngIf="!loginValid">
        The username or password were not recognized
      </mat-error>
      <mat-form-field>
        <input matInput placeholder="Email" [(ngModel)]="username" name="username" required>
      <mat-error>
        Please provide a valid username
      </mat-error>
      </mat-form-field>
      <mat-form-field>
        <input matInput type="password" placeholder="Password" [(ngModel)]="password"
name="password" required>
      <mat-error>
        Please provide a valid password
      </mat-error>
      </mat-form-field>
      <button mat-raised-button color="primary" [disabled]="!loginForm.form.valid">Login</button>
    </form>
  </mat-card-content>
</mat-card>
```

CSS

```
mat-card {
  max-width: 400px;
  margin: 2em auto;
  text-align: center;
}
```

```
mat-form-field {
  display: block;
}
```

Component ts file:

```
public loginValid = true;
public email = 'j@j.com';
public password = '1234';

constructor(private router:Router, private userserv:UserService) { }

/**
 * when user click on login button, get token for valid credentials and redirect to home page
 * or display an error message for invalid credentials and remain on login page
 */
public onSubmit(): void {
  this.loginValid = true;
```

```

console.log('login',this.email, this.password)
this.userserv.loginUser(this.email, this.password)
.subscribe({
  next: resp => {
    console.log(resp);
    if(resp !== undefined )
    {
      localStorage.setItem("token",resp.jwtToken);
      this.router.navigate([""]);
    }
  },
  error: err => this.loginValid = false
});
}

```

- Update the routes []. As follows:

```
{path:'login', component:LoginComponent}
```

- Update header component html to add login url:

```

<li class="nav-item">
  <a class="nav-link active" routerLinkActive="active" routerLink="/login">Login</a>
</li>

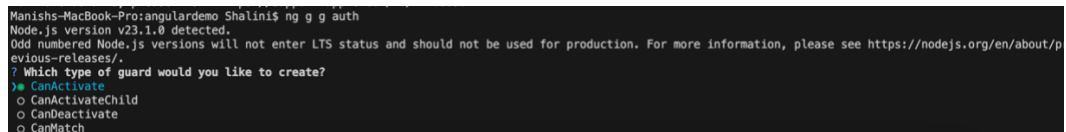
```

Check for success status and if token is generated in the console and stored in local storage

- To create a CanActivateGuard to protect , execute below command from with the **service folder** and select CanActivate as the default

```
ng g g auth
```

Select CanActivate and press enter



```

Manishs-MacBook-Pro:angulardemo Shalini$ ng g g auth
Node.js version v23.1.0 detected.
Odd numbered Node.js versions will not enter LTS status and should not be used for production. For more information, please see https://nodejs.org/en/about/p
revious-releases/.
? Which type of guard would you like to create?
> CanActivate
  CanActivateChild
  CanDeactivate
  CanMatch

```

- Update the auth guard ts file as follows that allows access to protected routes if it returns true else will display an alert and redirect to login page:

```

const us : UserService = inject(UserService);
const router:Router = inject(Router);

console.log('auth guard')

if(us.isAuthenticated()){
  return true;
}
alert('Please Login')
router.navigate(['login'])
return false;

```

- Lets guard the profile component from unauthorized access. Update routes as follows:

```

{path:'employees/:id', component:ProfileComponent, canActivate:[authGuard],
children:[
  {path:'info', component:ProfileinfoComponent}

```

```
}},
```

Executing the application, clicking on profile button should display an alert if not logged in. else should allow access to profile page

11. Still we get unauthorized access as backend application needs token in the header. To add token for protected routes create a service as follows within the service folder:

ng g s token

12. Add below code in token service to act as an interceptor and send the token as part of the header in request:

```
class TokenService implements HttpInterceptor{

  constructor(private us:UserService) { }
  intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
    console.log('interceptor')
    let reqclone = req;
    if(this.us.isAuthenticated()){
      console.log('yes')
      let token = localStorage.getItem('token')!
      console.log(token)
      reqclone = req.clone({
        headers: req.headers.set("Authorization", "" + token)
      })
    }
    return next.handle(reqclone);
  }
}
```

13. Inject this as a provider in the app.module.ts file as follows:

```
providers: [provideHttpClient(withInterceptorsFromDi()), provideAnimationsAsync(),
  { provide: HTTP_INTERCEPTORS, useClass: TokenService, multi: true }
],
```

14. Now running the code if user is logged in should allow access to the profile component and display the employee information on the page.

BUT THE DATA WILL NOT BE DISPLAYED: IDENTIFY THE ISSUE?

15. The login link in header should be conditionally displayed. Let's update the header.html and ts file for conditional rendering of links and the logout link

```
constructor(public us:UserService, private router: Router){ }
```

```
logout(){
  this.us.logout();
  this.router.navigate(['']);
}
```

```
<li class="nav-item" *ngIf="!us.isAuthenticated()">
  <a class="nav-link" routerLinkActive="active" routerLink="/login">Login</a>
</li>
<li class="nav-item">
  <a class="nav-link" routerLinkActive="active" routerLink="/employees">Employees List</a>
</li>
<li class="nav-item">
  <a class="nav-link" routerLinkActive="active" routerLink="/add">Add Employee</a>
```

```
</li>
<li class="nav-item" *ngIf="us.isAuthenticated()">
  <a class="nav-link" routerLinkActive="active" routerLink="/logout"
(click)="logout()">Logout</a>
</li>
```