Clase 11 @November 3, 2022

Esta clase veremos

- Interacción con una API REST a través de HTTP
- Observables
- Empezando a consumir la API
- Introducción a interceptors
- Introducción a guards

Interacción con una API REST a través de HTTP

1. Para empezar vamos a hacer un nuevo endpoint en nuestra API para retornar una lista de Homeworks.

```
using System;
namespace Domain;

public class Homework
{
    public int Id { get; set; }
    public string Description { get; set; }
    public DateTime DueDate  { get; set; }
    public int Score { get; set; }
    public int Rating { get; set; }
    public List<Exercise> exercises { get; set; }
}
```

```
using System;
namespace Domain;

public class Exercise
{
    public int Id { get; set; }
    public string Problem { get; set; }
    public int Score  { get; set; }
}
```

2. Vamos a restructurar un poco cambiando un poco la estructura de carpetas. Al cambiar empieza a fallar los imports por lo que configuramos los imports absolutos también

```
/* To learn more about this file see: https://angular.io/config/tsconfig. */
{
  "compileOnSave": false,
  "compilerOptions": {
    "baseUrl": "./", //Esta opción
    "outDir": "./dist/out-tsc",
    "forceConsistentCasingInFileNames": true,
    "strict": true,
    "noImplicitOverride": true,
    "noPropertyAccessFromIndexSignature": true,
    "noImplicitReturns": true,
    "noFallthroughCasesInSwitch": true,
    "sourceMap": true,
    "declaration": false,
    "downlevelIteration": true,
    "experimentalDecorators": true,
    "moduleResolution": "node",
    "importHelpers": true,
    "target": "es2020",
    "module": "es2020",
    "lib": [
      "es2020",
      "dom"
    ]
  },
  "angularCompilerOptions": {
    "enableI18nLegacyMessageIdFormat": false,
    "strictInjectionParameters": true,
    "strictInputAccessModifiers": true,
    "strictTemplates": true
  }
}
```

3. Cambiamos la base url para que esté en un .env
   Para esto es necesario ir a la carpeta enviroments y modificar cada ambiente

```
// This file can be replaced during build by using the `fileReplacements` array.
// `ng build` replaces `environment.ts` with `environment.prod.ts`.
// The list of file replacements can be found in `angular.json`.

export const environment = {
  production: false,
  BASE_URL: 'https://localhost:7012/api',
};

/*
 * For easier debugging in development mode, you can import the following file
```

```
 * to ignore zone related error stack frames such as `zone.run`, `zoneDelegate.invokeTask`.
 *
 * This import should be commented out in production mode because it will have a negative impact
 * on performance if an error is thrown.
 */
// import 'zone.js/plugins/zone-error';  // Included with Angular CLI.
```

```
import { Injectable } from '@angular/core';
import {
  HttpClient,
  HttpResponse,
  HttpRequest,
  HttpHeaders,
} from '@angular/common/http';
import { Observable, throwError } from 'rxjs';
import { map, tap, catchError } from 'rxjs/operators';
import { Homework } from 'src/app/models/Homework';
import { Exercise } from 'src/app/models/Exercise';
import { environment } from 'src/environments/environment';

@Injectable()
export class HomeworksService {
  private BASE_URL: string = environment.BASE_URL;

  constructor(private _httpService: HttpClient) {}

  getHomeworks(): Array<Homework> {
    return [
      new Homework(
        '1',
        'Una tarea',
        0,
        new Date(),
        [new Exercise('1', 'Un Problema', 0)],
        2
      ),
      new Homework('2', 'Otra tarea', 0, new Date(), [], 4),
    ];
  }
}
```

5. Agregamos los endpoints en un enum

```
export enum HomeworksEndpoints {
  GET_HOMEWORKS = '/homeworks',
}
```

6. Modificamos el servicio

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable, throwError } from 'rxjs';
import { Homework } from 'src/app/models/Homework';
import { environment } from 'src/environments/environment';
import { map, tap, catchError } from 'rxjs/operators';
import { HomeworksEndpoints } from '../endpoints';

@Injectable()
export class HomeworksService {
  private BASE_URL: string = environment.BASE_URL;

  constructor(private _httpService: HttpClient) {}

  getHomeworks(): Observable<Homework[]> {
    return this._httpService
      .get<Homework[]>(`${this.BASE_URL}${HomeworksEndpoints.GET_HOMEWORKS}`)
      .pipe(
        map((data) => <Homework[]>data),
        tap((data) =>
          console.log('Los datos que obtuvimos fueron: ' + JSON.stringify(data))
        ),
        catchError(this.handleError)
      );
  }

  private handleError(error: Response) {
    console.error(error);
    return throwError(error.json() || 'Server error'); //Cuidado!
  }
}
```

7. Modificamos donde se usa el servicio

```
//Archivo homework-list.component.ts
import { Component, OnInit } from '@angular/core';
import { Homework } from '../../models/Homework';
import { Exercise } from '../../models/Exercise';
import { HomeworksService } from 'src/app/core/http-services/homeworks/homeworks.service';
@Component({
  selector: 'app-homeworks-list',
  templateUrl: './homeworks-list.component.html',
  styleUrls: ['./homeworks-list.component.css'],
})
export class HomeworksListComponent implements OnInit {
  pageTitle: string = 'Homeworks List';
  listFilter: string = '';
  showExercises: boolean = false;
  homeworks: Homework[] = [];
  text: string = '';
  constructor(private serviceHomework: HomeworksService) {}
```

```
  ngOnInit() {
    this.serviceHomework.getHomeworks().subscribe(
      (data: Array<Homework>) => this.setHomeworks(data),
      (error: any) => console.log(error)
    );
  }

  private setHomeworks(data: Array<Homework>): void {
    this.homeworks = data;
  }

  onRatingClicked(message: string): void {
    this.pageTitle = 'Homeworks list ' + message;
  }

  toggleExercises(): void {
    this.showExercises = !this.showExercises;
  }
}
```

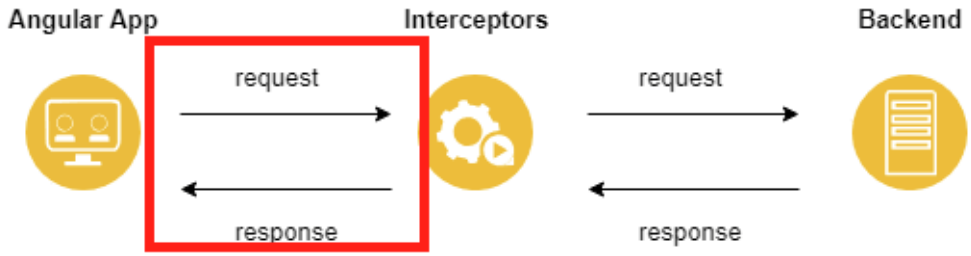
```
//Archivo homework-detail.component.ts
import { Component, OnInit } from '@angular/core';
import { Homework } from '../../models/Homework';
import { ActivatedRoute, Router } from '@angular/router';
import { HomeworksService } from 'src/app/core/http-services/homeworks/homeworks.service';

@Component({
  selector: 'app-homework-detail',
  templateUrl: './homework-detail.component.html',
  styleUrls: ['./homework-detail.component.css'],
})
export class HomeworkDetailComponent implements OnInit {
  pageTitle: string = '';
  aHomework: Homework | undefined;
  constructor(
    private _currentRoute: ActivatedRoute,
    private serviceHomework: HomeworksService,
    private _router: Router
  ) {}

  ngOnInit(): void {
    this.pageTitle = 'Homework detail!!';
    let id = this._currentRoute.snapshot.params['id'];
    this.serviceHomework.getHomeworks().subscribe(
      (data: Homework[]) => this.setHomework(data, id),
      (error: any) => console.log(error)
    );
  }

  private setHomework(data: Homework[], id: string): void {
    this.aHomework = data.find((x) => x.id === id);
  }
```

```
  onBack(): void {
    this._router.navigate(['/homeworks']);
  }
}
```

7. Tenemos funcional con la restructuración la pantalla que teníamos antes

Introducción a interceptors

Los interceptores podrían verse como análogos a lo que son los filters en .NET Core. Básicamente nos permiten interceptar requests o responses y hacer algún procesamiento extra o mutación sobre la data que se envía o se devuelve.



Nos prové de un método **intercept** que recibe un parámetro **req** con la request que se está enviando y otro parámetro **next** el cuál debemos ejecutar para que el próximo interceptor en la cadena de interceptors sea ejecutado.

Ejemplo BASE_URL

1. Vamos a ir a core/interceptos y creamos un api-interceptors
2. Creamos el interceptor

```
import {
  HttpEvent,
  HttpHandler,
  HttpInterceptor,
  HttpRequest,
} from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { environment } from 'src/environments/environment';

@Injectable()
export class APIInterceptor implements HttpInterceptor {
```

```
  intercept(
    req: HttpRequest<any>,
    next: HttpHandler
  ): Observable<HttpEvent<any>> {
    const apiReq = req.clone({ url: `${environment.BASE_URL}${req.url}` });
    return next.handle(apiReq);
  }
}
```

3. Lo agregamos en el app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';
import { HomeworksModule } from './homeworks/homeworks.module';
import { HTTP_INTERCEPTORS } from '@angular/common/http';
import { APIInterceptor } from './core/interceptors/api-interceptor';

@NgModule({
  declarations: [AppComponent],
  imports: [BrowserModule, FormsModule, HomeworksModule],
  providers: [
    {
      provide: HTTP_INTERCEPTORS,
      useClass: APIInterceptor,
      multi: true,
    },
  ],
  bootstrap: [AppComponent],
})
export class AppModule {}
```

4. Cambiamos el código de la request

```
get<Homework[]>(HomeworksEndpoints.GET_HOMEWORKS, options)
```

Otros ejemplos que después vamos a ver en detalle

1. Token Interceptor
2. Error Interceptor
3. RefreshInterceptor

Introducción a Guards

Las guards son una forma de verificar que una ruta puede de nuestra navegación puede ser accedida o no. En caso de que si continua con su comportamiento normal y en caso de que no podemos poner un fallback como navegar a una pantalla de no autorizado, al login o a un 404.

Provee de un método **canActivate** que recibe la **route** en la que está y otro parámetro que representa el **state** del router.

Ejemplo

1. Modificar schemantics en angular.json para que el nuevo servicio sea creado en la carpeta que nosotros queremos

```
"schematics": {
        "@schematics/angular:service": {
          "path": "src/app/core/http-services"
        }
      }
```

1. Crear service de login

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { UsersEndpoints } from '../endpoints';

@Injectable({
  providedIn: 'root',
})
export class UsersService {
  constructor(private _httpService: HttpClient) {}

  public login(): Observable<any> {
    return this._httpService.post<any>(UsersEndpoints.LOGIN, {
      Email: 'marcotest1@gmail.com',
      Password: '12345',
      Token: '1',
    });
  }
}
```

3. Crear screen de login
    a. ng generate module login
    b. ng generate component login

c. Modificamos el componente de menu

d. Agregamos la ruta en routes y agregamos el imports en app.module.ts

e. Agregamos un botón básico de login

```
<button (click)="isLoggedIn() ? logout() : login()" class="btn btn-primary">
  {{ isLoggedIn() ? "Logout" : "Login" }}
</button>
```

El component.ts queda así:

```
import { Component, OnInit } from '@angular/core';
import { UsersService } from '../core/http-services/users/users.service';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css'],
})
export class LoginComponent implements OnInit {
  logged: boolean = false;

  constructor(private _userService: UsersService) {}

  ngOnInit(): void {}

  login() {}

  logout() {}
}
```

4. Interactuar con la api

```
import { Component, OnInit } from '@angular/core';
import { UsersService } from '../core/http-services/users/users.service';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css'],
})
export class LoginComponent implements OnInit {
  constructor(private _userService: UsersService) {}

  ngOnInit(): void {}

  isLoggedIn(): boolean {
```

```
    return localStorage.getItem('userInfo') != null;
  }

  login() {
    this._userService.login().subscribe((userInfo) => {
      console.log('userInfo', userInfo);
      this.saveUserInfo(
        JSON.stringify({ email: userInfo.email, token: userInfo.token })
      );
    });
  }

  logout() {
    localStorage.removeItem('userInfo');
  }

  private saveUserInfo(userInfo: string): void {
    localStorage.setItem('userInfo', userInfo);
  }
}
```

5. Agregar Guard

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

@Injectable()
export class AuthGuard implements CanActivate {
  constructor(private _router: Router) {}

  public canActivate(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot
  ): boolean {
    if (localStorage.getItem('userInfo')) {
      // logged in so return true
      return true;
    }

    // not logged in so redirect to login page
    this._router.navigate([`/login`], {
      queryParams: { returnUrl: state.url },
    });
    return false;
  }
}
```

6. Hacer que la ruta de homeworks solo pueda ser accedida si el usuario está loggeado
    a. app.module.ts en providers
    b. homework.module.ts en routes con parámetro canActivate

```
const rootes: Routes = [
  {
    path: 'homeworks',
    component: HomeworksListComponent,
    canActivate: [AuthGuard],
  },
  {
    path: 'homeworks/:id',
    component: HomeworkDetailComponent,
    canActivate: [AuthGuard],
  },
];
```

7. Usar returnUrl

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute, Router } from '@angular/router';
import { UsersService } from '../core/http-services/users/users.service';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css'],
})
export class LoginComponent implements OnInit {
  constructor(
    private _userService: UsersService,
    private _route: ActivatedRoute,
    private _router: Router
  ) {}

  ngOnInit(): void {}

  isLoggedIn(): boolean {
    return localStorage.getItem('userInfo') != null;
  }

  login() {
    this._userService.login().subscribe((userInfo) => {
      console.log('userInfo', userInfo);
      this.saveUserInfo(
        JSON.stringify({ email: userInfo.email, token: userInfo.token })
      );
      let urlToGo;
      this._route?.queryParams.forEach((value: any) => {
```

```
        if (value?.returnUrl) {
          urlToGo = value?.returnUrl;
        }
      });
      if (urlToGo) {
        this._router.navigate([urlToGo]);
      }
    });
  }

  logout() {
    localStorage.removeItem('userInfo');
  }

  private saveUserInfo(userInfo: string): void {
    localStorage.setItem('userInfo', userInfo);
  }
}
```

Más interceptors

TokenInterceptor

```
import { Injectable } from '@angular/core';
import { HttpRequest, HttpHandler, HttpEvent, HttpInterceptor } from '@angular/common/http';
import { Observable } from 'rxjs';

@Injectable()
export class TokenInterceptor implements HttpInterceptor {
    intercept(request: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
        const userInfo = JSON.parse(localStorage.getItem('userInfo'));
        const token = userInfo?.token;
        if (token) {
            request = request.clone({
                setHeaders: {
                    Authorization: token
                }
            });
        }
        return next.handle(request);
    }
}
```