**Interceptor and Life cycle Hooks**

**Interceptor**

**https://alligator.io/angular/testing-http-interceptors/**

<https://medium.com/@er.surajnegi/what-are-angular-http-interceptors-and-how-to-create-them-82d1d6159c4e>

Why we need HTTP Interceptors:

According to Angular Team at Google “**When your application makes a request, interceptors transform it before sending it to the server, and the interceptors can transform the response on its way back before your application sees it**.”

How to Implement the Interceptors in Angular with code:

Below is the list of steps we need to do to implement the Interceptors in our Angular App:

1. We need to Import the **HttpClientModule** into our application in app.

2. We need to add a Class decorated with Injector Decorator, where we will write our logic for Interceptors.

3. Then we need to provide information to angular to use our class from 2nd step as interceptors.

Enough with the theory now, let’s do some coding now.

First of all, to make any http call we need to import **HttpClientModule**into our application in **app.module.ts**as follow:

|  |
| --- |
| import { NgModule } from '@angular/core'; |
|  | import { BrowserModule } from '@angular/platform-browser'; |
|  | import { HttpClientModule } from '@angular/common/http'; |
|  | import { AppComponent } from './app.component'; |
|  |  |
|  | @NgModule({ |
|  | imports: [BrowserModule, HttpClientModule], |
|  | declarations: [AppComponent], |
|  | bootstrap: [AppComponent] |
|  | }) |
|  | export class AppModule { } |

After this, we need to add a class which will work as an interceptor for us. Let’s create a file named as **my-interceptor.ts**and copy the below code in it save it.

|  |
| --- |
| import { Injectable } from "@angular/core"; |
|  | import { tap } from "rxjs/operators"; |
|  | import { |
|  | HttpRequest, |
|  | HttpHandler, |
|  | HttpEvent, |
|  | HttpInterceptor, |
|  | HttpResponse, |
|  | HttpErrorResponse |
|  | } from "@angular/common/http"; |
|  | import { Observable } from "rxjs/Observable"; |
|  |  |
|  | @Injectable() |
|  | export class MyInterceptor implements HttpInterceptor { |
|  | constructor() { } |
|  | //function which will be called for all http calls |
|  | intercept( |
|  | request: HttpRequest<any>, |
|  | next: HttpHandler |
|  | ): Observable<HttpEvent<any>> { |
|  | //how to update the request Parameters |
|  | const updatedRequest = request.clone({ |
|  | headers: request.headers.set("Authorization", "Some-dummyCode") |
|  | }); |
|  | //logging the updated Parameters to browser's console |
|  | console.log("Before making api call : ", updatedRequest); |
|  | return next.handle(request).pipe( |
|  | tap( |
|  | event => { |
|  | //logging the http response to browser's console in case of a success |
|  | if (event instanceof HttpResponse) { |
|  | console.log("api call success :", event); |
|  | } |
|  | }, |
|  | error => { |
|  | //logging the http response to browser's console in case of a failuer |
|  | if (event instanceof HttpResponse) { |
|  | console.log("api call error :", event); |
|  | } |
|  | } |
|  | ) |
|  | ); |
|  | } |
|  | } |

**Component Life Cycle Hooks**

[**https://codecraft.tv/courses/angular/components/lifecycle-hooks/**](https://codecraft.tv/courses/angular/components/lifecycle-hooks/)

**constructor**

This is invoked when Angular creates a component or directive by calling new on the class.

**ngOnChanges**

Invoked **every** time there is a change in one of th input properties of the component.

\*

https://www.concretepage.com/angular-2/angular-2-4-onchanges-simplechanges-example

<https://www.ryadel.com/en/angular-ngonchanges-onchanges-lifecycle-hook-changes-async/>

<https://medium.com/@christophkrautz/testing-ngonchanges-in-angular-components-bbb3b4650ee8>

**ngOnInit**

**https://toddmotto.com/angular-constructor-ngoninit-lifecycle-hook**

Invoked when given component has been initialized.  
This hook is only called **once** after the first ngOnChanges

hhttps://tttps://toddmotto.com/angular-constructor-ngoninit-lifecycle-hook}}

**ngDoCheck**

Invoked when the change detector of the given component is invoked. It allows us to implement our own change detection algorithm for the given component.

#### Important

ngDoCheck and ngOnChanges should not be implemented together on the same component.

**ngOnDestroy**

**https://www.concretepage.com/angular-2/angular-2-4-oninit-and-ondestroy-example**

This method will be invoked just before Angular destroys the component.  
Use this hook to unsubscribe observables and detach event handlers to avoid memory leaks.

### [Hooks for the components children](https://codecraft.tv/courses/angular/components/lifecycle-hooks/#_hooks_for_the_components_children)

These hooks are only called for components and not directives.

**ngAfterContentInit**

Invoked after Angular performs any content projection into the components view

**ngAfterContentChecked**

Invoked each time the content of the given component has been checked by the change detection mechanism of Angular.

**ngAfterViewInit**

Invoked when the component’s view has been fully initialized.

**ngAfterViewChecked**

Invoked each time the view of the given component has been checked by the change detection mechanism of Angular.