# Angular Best Friends

## Module 8 Exercise 3 – Interceptors

## Goals

Interceptors are a great way to inspect HttpTraffic and manipulate requests and responses. In this exercise we’ll implement an interceptor.

## Steps

### Setting up the environment

1. Go to the **Module8Exercise3** folder. Open the “**initial**” folder in VS Code.
2. Open the terminal and run **npm install** to install the needed node modules.
3. This app is slightly different than all others because it also has a NodeJS server that serves the initial index.html file and that has API endpoints to play around with resources. So to build everything up run **npm start**.
4. Wait until you see the Angular specific bundles in the terminal.
5. Open a browser tab and navigate to **localhost:3000** . The NodeJs server listens to this port and this is how we run this Angular application

### Implement an interceptor

1. First we’ll have to create a new file for the interceptor. Navigate to the “**core**” folder and there create a new file named **add-header.interceptor.ts**.
2. In the newly created file paste the following code snippet:

import { Injectable } from '@angular/core';

import { HttpEvent, HttpInterceptor, HttpHandler, HttpRequest } from '@angular/common/http';

import { Observable } from 'rxjs';

@Injectable()

export class AddHeaderInterceptor implements HttpInterceptor {

intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {

console.log(`AddHeaderInterceptor - ${req.url}`);

let jsonReq: HttpRequest<any> = req.clone({

setHeaders: {'Content-Type': 'application/json'}

});

return next.handle(jsonReq);

}

}

1. Since the interceptor is a service, it needs to be provided in a module. We’ll add them to the core module. So navigate to the **core.module.ts** file. Add the following import:

import { HTTP\_INTERCEPTORS } from '@angular/common/http';

1. We also have to import the AddHeader interceptor class:

import { AddHeaderInterceptor } from './add-header.interceptor';

1. Add it to the provider using following syntax:

providers: [

LoggerService,

DataService,

{ provide: ErrorHandler, useClass: BookTrackerErrorHandlerService },

{ provide: HTTP\_INTERCEPTORS, useClass: AddHeaderInterceptor, multi: true }

]

1. Run the application and test it out. If you open the developer tools, we log the name of the interceptor there as the request passes through the interceptor. So this is a sign that it is working. If you want you can inspect each http call to the API and will see that each call will now have a content-type header.