Sommario

[Angular HTTP Interceptor 2](#_Toc22549374)

[Come intercettare richieste e risposte HTTP 2](#_Toc22549375)

[Creare un Interceptor 2](#_Toc22549376)

[Il metodo intercept 3](#_Toc22549377)

[Providing di un interceptor 3](#_Toc22549378)

[Ordine di esecuzione 4](#_Toc22549379)

[Il mio esempio 5](#_Toc22549380)

# Angular HTTP Interceptor

## Come intercettare richieste e risposte HTTP

Angular è un framework in costante evoluzione e, con ogni versione rilasciata, nuove feature sono messe a disposizione degli sviluppatori.

Sin dagli albori, come ogni web framework, Angular ha un package per la gestione delle richieste Http. La versione 4.3 ha introdotto il package *HttpClient*, deprecando *Http*. Con *HttpClient* sono stati introdotti gli **HTTP Interceptor,**che consentonodi intercettare le *response* e le *request*delle chiamate HTTP effettuate dalla *Single Page Application*. Il funzionamento di un interceptor è del tutto comparabile ad un route middleware di un qualsiasi web framework, come [Slim](http://www.slimframework.com/" \t "_blank) oppure [Express](https://expressjs.com/).

## Creare un Interceptor

Per creare un interceptor in Angular è necessario definire un servizio che implementa l’interfaccia *HttpInterceptor*, importandola da *@angular/common/http.*La classe implementerà il metodo *intercept*, che verrà invocato per intercettare una richiesta o una risposta HTTP.

import {   
 HttpEvent,   
 HttpInterceptor,   
 HttpHandler,   
 HttpRequest   
} from '@angular/common/[http](https://angular.io/api/common/http" \t "_blank)';import { Observable } from 'rxjs';export class FakeInterceptor implements HttpInterceptor { intercept(req: HttpRequest<any>, next: HttpHandler)  
 :Observable<HttpEvent<any>> { console.log(new Date());  
 return next.handle(req);   
 }  
}

L’interceptor definito nell’esempio è banale e si occuperà di stampare a console la data della intercettazione.

All’interno di una applicazione è possibile definire molteplici interceptor, che verranno chiamati in successione.

## Il metodo intercept

Il metodo *intercept* ha due parametri in ingresso.

Il primo parametro è un oggetto che modella la richiesta HTTP e implementa l’interfaccia *HttpRequest*. Tramite questo oggetto è possibile effettuare manipolazioni della richiesta, ma attenzione! L’oggetto della richiesta è immutabile. Manipolare la richiesta significa, sostanzialmente, creare un clone e apportare le modifiche del caso su di esso.

Il secondo parametro implementa l’interfaccia *HttpHandler,* la quale definisce il metodo *handle*. Esso ha in input una *HttpRequest* e restituisce un observable. La chiamata ad *handle* continua la catena di interceptor, passando il controllo al prossimo.

## Providing di un interceptor

Come per ogni altro servizio in Angular, é necessario definire un provider per gli interceptor. Il developer deve effettuare il providing degli interceptor nello stesso injector (o in un padre dell’injector) che fa da provider per *HttpClient*.

Un esempio di providing è il seguente:

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

L’oggetto che definisce il provider ha tre proprietà:

* *provide*è avvalorato con*HTTP\_INTERCEPTORS,*importato da *@angular/common/http*.
* *useClass* indica il nome della classe da istanziare.
* *multi,*quando ha valore*true*, indica al sistema di Dependency Injection che il token *HTTP\_INTERCEPTORS* è un *multiprovider,*ovvero effettua l’injection di un array di istanze.

## Ordine di esecuzione

L’ordine in cui vengono chiamati gli interceptor corrisponde all’ordine in cui viene effettuato il providing dei relativi servizi.

Ringrazio [Francesco Dammacco](https://medium.com/u/46807b521ccb?source=post_page-----417ffda5f2e6----------------------) per la collaborazione alla stesura di questo blog post.

## Il mio esempio

* Creare una cartella http-interceptor
* Creare un file ErrorInterceptor.ts
* import { HttpErrorResponse, HttpEvent, HttpHandler, HttpInterceptor, HttpRequest } from '@angular/common/http';
* import { Injectable } from '@angular/core';
* import { Observable, throwError } from 'rxjs';
* import { catchError } from 'rxjs/operators';
* // import { LoaderService } from '../services/loader.service';
* /\*\* Pass untouched request through to the next request handler. \*/
* @Injectable()
* export class ErrorInterceptor implements HttpInterceptor {
* constructor(
* // private \_loaderSVC: LoaderService
* ) { }
* showError: boolean;
* intercept( req: HttpRequest<any>, next: HttpHandler ):
* Observable<HttpEvent<any>> {
* console.debug( `Http Request, method: ${req.method} - url: ${req.url} - params: ${req.params}` );
* //  setTimeout(() => {
* //  this.\_loaderSVC.displayLoader(false);
* //  }, 1000);
* return next.handle( req ).pipe(
* catchError( this.handleError )
* );
* }
* /\*\*
* \*
* \* @param error
* \*/
* private handleError( error: HttpErrorResponse ) {
* if ( error.error instanceof ErrorEvent ) {
* // A client-side or network error occurred. Handle it accordingly.
* console.error( 'An error occurred:', error.error.message );
* } else {
* // The backend returned an unsuccessful response code.
* // The response body may contain clues as to what went wrong,
* console.error(
* `Backend returned code ${error.status}, ` +
* `body was: ${error.error}` );
* }
* // return an ErrorObservable with a user-facing error message
* return throwError(new Error(
* 'Something bad happened; please try again later.' ));
* };
* }
* App.module.ts

Prima di @NgModule inserire le seguenti righe :

export const httpInterceptorProviders = [

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

  ];

Nella sezione providers contenuta in @NgModule aggiungere :

    providers: [

        httpInterceptorProviders,