Interceptor

TABLE OF CONTENTS

Introduction 3

Interceptor 3

What is Interceptor? 3

How Interceptor works? 3

Uses of Interceptor in Angular 4

Headers 4

Loaders 4

Caching 4

Response Formatting 4

Implementation 5

Step 1: Create a Token Interceptor to modify header 5

Step 2: Implement intercept method 5

Step 3: Add Interceptors to provider 6

Conclusion 6

Anandhakumar R

# Introduction

This documentation will provide a detailed explanation of Interceptor as well as a step-by-step process for better implementing them in a sample Angular application.

# Interceptor

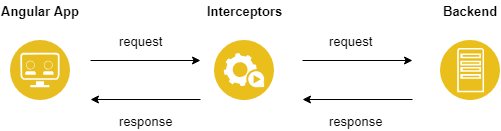
## What is Interceptor?

Interceptor is a special Angular Service that can be used to intercept all the request and response calls and modify them to our requirement.

Most interceptors transform the outgoing request before passing it to the next interceptor in the chain(if there are multiple interceptors), by calling the next.handle(transformedReq).

We can transform the response stream by applying additional RxJS operators on the stream returned by the next.handle().

## How Interceptor works?



As you see in the above diagram, every request that comes from the Angular app will pass through the interceptor, and then it reaches the backend. Likewise, responses coming from the backend also pass through the interceptor before reaching the Angular app. So, in the interceptor, we can modify request and response data.

# Uses of interceptor in angular

We can use an interceptor for many purposes. We'll look at some of the most common interceptor applications in Angular.

## Headers

Angular interceptors manipulate the headers as they provide features such as authentication and authorization. For example, we can intercept the request and add the authentication token in the Authorization header. By doing this, we can add tokens to all the requests that have been sent from the front end.

## Loaders

Interceptors can be used as loaders whenever there are different active requests. A loader function with both hide and show features is used to handle the requests. And we know that request has been sent, and we are waiting for the response by seeing the loading animation.

## Caching

We’ll create our HTTP interceptor so that whenever we place a GET request, the request will pass through the interceptors in the chain. Our interceptor will check the request to determine whether it has been cached. If yes, it will return the cached response. If not, it will pass the request along to the remainder of the chain to eventually make an actual server request. The interceptor will watch for the response when it receives it and cache it so that any other request will return the cached response.

## Response Formating

Interceptor can even convert the format of the API response we receive. For example, we can change the XML data from the API to JSON before it reaches our Angular application.

# Implementation

## Step 1: Create a Token Interceptor to modify header

Open the CLI inside your Angular project, move to the project directory, and enter the below command to create a TokenInterceptorService. This command will create a token interceptor inside interceptors folder in the core module.

****

After creating the class, implement the HttpInterceptor as shown in the below image. This interface has a method called intercept that will intercept and handle an HTTP request or HTTP response.

Graphical user interface, text, application, email

Description automatically generated

**req** in intercept method is the outgoing request object to handle.

**next in the intercept method is t**he next interceptor in the chain, or the backend if no interceptors remain in the chain.

Intercept method returns an observable of event stream.

## Step 2: Implement Intercept method

Inside intercept method we copied the outgoing request. We get the token of the current user by calling getLoggedInUser method in the userService class. We then include that token in the header. And then we pass this cloned request on to the next. handle() function. So the next interceptor in the chain or backend will receive the modified header. As a result, before reaching the backend, every request sent from Angular will include the token.

Graphical user interface, text, application, email

Description automatically generated

## Step 3: Add Interceptors to providers

The interceptor needs to be added to the HTTP\_INTERCEPTORS array. This is done by adding the new class to the providers array of our application root module, as shown in the below image.

Graphical user interface

Description automatically generated

# Conclusion

In this documentation, we have created the sample Token Interceptor that will intercept all the outgoing request from the Angular application and add the token of the current user to the request. So that we don’t need to add token manually for every request. And we also discussed about the uses of the interceptor.