**Handle Angular 13 HTTP Errors with HttpInterceptor & RxJS**

In this chapter

will learn how to handle HTTP errors in Angular using HttpInterceptor and RxJS (reactive programming) error operators.

Error handling is a process of handling errors or exceptions manifested in software. Errors may occur in an application due to some internal or external problems.

You always have a proper structure to manage exceptions; exception handling helps users find out the cause and propel them to rectify the problem.

Consequently, every developer’s notion that error handling is considered significant from the user experience perspective.

**Angular 13 Http Error Handling Examples**

In this angular error handling example, we will learn to handle the client-side error while making or processing the HTTP requests in the angular app.

You will also see how to receive the error response with the proper error message using HttpInterceptor and RxJS catchError operator.

Generically, we have heard about two kind of exceptions server-side and client-side:

Server-side errors come with a 5xxx status code with the conventional reason for the occurrence.

Whereas, client-side errors occur due to some internal problems. To handle angular client-side errors, we are explaining everything bit by by:

To get along with this guide, you need to set up angular CLI and an angular application:

npm install -g @angular/cli@latest

Bash

COPY

Install new angular application:

ng new project-name

Bash

COPY

**Disable Strict Angular TypeStrict Errors**

The latest version of Angular comes with strict mode, you have to manually disable the strict mode you can set “strict”: false, "noImplicitReturns": false and "strictTemplates": false inside the compilerOptions and angularCompilerOptions in **tsconfig.json** file.

**Catch Errors with HttpClient and RxJS**

In this step, we will share with you angular error handling best practice that requires you to import retry, catchError, Observable, throwError, and equally important HttpClient APIs.

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

import { HttpClient } from '@angular/common/http';

import { retry, catchError } from 'rxjs/operators';

import { Observable, throwError } from 'rxjs';

class User {

constructor(

public id: string,

public name: string,

public email: string,

public phone: string

) { }

}

@Injectable({

providedIn: 'root'

})

export class UserService {

private API = 'API Goes Here';

constructor(private httpClient: HttpClient) { }

fetchBlogList(): Observable<User[]> {

return this.httpClient.get<User[]>(this.API)

.pipe(

retry(2),

catchError(this.getError)

);

}

getError(error) {

let message = '';

if (error.error instanceof ErrorEvent) {

// handle client-side errors

message = `Error: ${error.error.message}`;

} else {

// handle server-side errors

message = `Error Code: ${error.status}\nMessage: ${error.message}`;

}

console.log(message);

return throwError(message);

}

}

TypeScript

COPY

**Error Handling with Angular HttpInterceptor**

In this step, you will see a profound way of handling errors in angular using HttpInterceptor API. It is an interface; it Intercepts and handles an HttpRequest or HttpResponse.

Create a new **exception.interceptor.ts** file, update the below code within:

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

import { catchError, retry } from 'rxjs/operators';

import { Observable, throwError } from 'rxjs';

export class ExceptionIntercept implements HttpInterceptor {

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

return next.handle(request)

.pipe(

retry(1),

catchError((error: HttpErrorResponse) => {

let message = '';

if (error.error instanceof ErrorEvent) {

// handle client-side error

message = `Error: ${error.error.message}`;

} else {

// handle server-side error

message = `Error Status: ${error.status}\nMessage: ${error.message}`;

}

console.log(message);

return throwError(message);

})

)

}

}

TypeScript

COPY

Import HttpEvent, HttpHandler, HttpRequest, HttpErrorResponse, HttpInterceptor from ‘@angular/common/http’ library at the top section of your interceptor template.

Also, import catchError, retry, Observable, throwError RxJS operator, and methods.

The intercept() method takes the request and next parameter, followed by HttpRequest and HttpHandler.

We used the pipe() method to run and execute the various RxJS operator simultaneously.

The catchError() handles the client-side similarly server-side errors and returns an error response with the conventional message.

**Conclusion**

This tutorial is completed; we reckon it will enhance your understanding of handling HTTP client-side errors in the angular app.