**Error Interceptor**

An error interceptor is a special type of interceptor which is used for handling errors that arise while making an HTTP request. The error may come in either the client side (browser) or it may be an error from the server side when the request fails due to any reason. Error handling is very important for any modern day application as it is very imperative to do the next course of actions if some request response cycle fails. This is where the mechanism of error interceptor comes into play.

Now let’s explore the main part of this article i.e the error interceptor. Let’s create a new service inside the services folder and name it as httperrorinterceptor.service.ts. This is the file where we will be writing the error interceptor logic. In the file we will have a very basic structure for error handling and we will see how to take care of different kinds of errors. Let’s have the following content in the httperrorinterceptor.service.ts file.

import { HttpEvent, HttpInterceptor, HttpHandler, HttpRequest, HttpErrorResponse } from '[@angular/common](http://twitter.com/angular/common)/http';  
import { Observable, throwError } from 'rxjs';  
import { catchError } from 'rxjs/operators';export class HttpErrorInterceptor implements HttpInterceptor {  
 intercept(request: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {  
 return next.handle(request)  
 .pipe(  
 catchError((error: HttpErrorResponse) => {  
 let errorMsg = '';  
 if (error.error instanceof ErrorEvent) {  
 console.log('this is client side error');  
 errorMsg = `Error: ${error.error.message}`;  
 }  
 else {  
 console.log('this is server side error');  
 errorMsg = `Error Code: ${error.status}, Message: ${error.message}`;  
 }  
 console.log(errorMsg);  
 return throwError(errorMsg);  
 })  
 )  
 }  
}

1. the HttpErrorInterceptor class implements the HttpInterceptor interface which is imported as part of @angular/common/http npm package.This is basically to implement the **intercept()** method.
2. the intercept method takes two parameters namely request which is of type HttpRequest and next which is of type HttpHandler.
3. the intercept method returns an Observable which of type HttpEvent with a signature any. You can have the type set to any custom type.
4. inside the rxjs pipe() operator we are checking if the error is an instance of ErrorEvent. This check is done to verify if the error is arising out of the client side.
5. else we take the error as a server side generated error.
6. for the sake of simplicity of demonstration we are just forming an error message with ES6 interpolation operator and returning the error through the throwError() operator of rxjs. You can also show some sort of alert message with toastr with the return value.

Now heading on to the final step, we will register the above file with out app module. So let’s move to the app.module.ts file and make some minor changes.

app.module.ts

import { BrowserModule } from '[@angular/platform-browser](http://twitter.com/angular/platform-browser)';  
import { NgModule } from '[@angular/core](http://twitter.com/angular/core)';import { AppRoutingModule } from './app-routing.module';  
import { AppComponent } from './app.component';  
import { HttpClientModule, HTTP\_INTERCEPTORS } from '[@angular/common](http://twitter.com/angular/common)/http';  
import { HttpErrorInterceptor } from './services/httpinterceptor.service';[@NgModule](http://twitter.com/NgModule)({  
 declarations: [  
 AppComponent,  
 ],  
 imports: [  
 BrowserModule,  
 AppRoutingModule,  
 HttpClientModule  
 ],  
 providers: [  
 {  
 provide: HTTP\_INTERCEPTORS,  
 useClass: HttpErrorInterceptor,  
 multi: true  
 }  
 ],  
 bootstrap: [AppComponent]  
})  
export class AppModule { }

Here we simply import the HttpErrorInterceptor class from the service file and register it in the providers array.One thing to note here is that the interceptor file is provided as HTTP\_INTERCEPTORS and we have kept multi as true to facilitate the provision for multiple interceptors.