**T11-HOL001\_TestCase [Testing of Angular Service with HttpClient]**

In this hands on we will perform of unit testing of angular service with HttpClient.

**Tasks:**

1. Create an interface called Post [**use the command: ng g interface post model**]. Under the interface add the following fields: id: number,userId: string,body: string,title: string.

**Sample code is given for your reference:**

export interface Post {

    id: number;

    userId: string;

    body: string;

    title: string;

}

1. Create a service called **Data** [**use the command: ng g s data**]. Under the service class do the following:
2. Create a public variable called ROOT\_URl and under it call the api : **'http://jsonplaceholder.typicode.com'**

***[Here we are using fake API as on this hands on, we are mainly concerned on how to perform unit testing of angular service using http client and not at how to create an Api].***

1. Inject the **HttpClient** under the constructor of the service class.
2. Create a method called **getPostData()** which will return the data from the Api.

**Sample Code is given for your reference:**

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

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

import {Post} from './post.model';

@Injectable({

  providedIn: 'root'

})

export class DataService {

  public ROOT\_URl = 'http://jsonplaceholder.typicode.com';

    constructor(private  http: HttpClient) {

    }

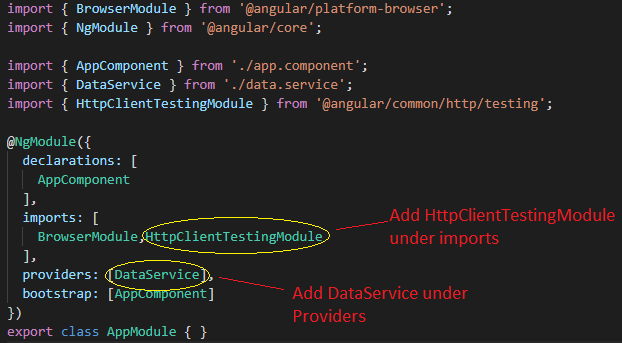
    getPostData() {

        return this.http.get<Post[]>(`${this.ROOT\_URl}/posts`);

    }

}

1. Under app.module.ts do the following:



1. Check if the describe () block for DataService is present under data.service.spec.ts.
2. Under  beforeEach() block do the following:
3. Create an instance of DataService
4. Create an instance of HttpTestingController to mock the http request
5. Also under TestBed.configureTestingModule() add the following:

imports: [HttpClientTestingModule],

providers: [DataService]

1. Add an afterEach() method, a method in which we run http.verify() and test our assertion mock like http[ basically it ensures that no request is outstanding].
2. Create a test case to check if the DataService is created.
3. Create a test case to mock the http request and test the expected method ‘GET’.

**Sample Test Code is given for your reference:**

import { TestBed } from '@angular/core/testing';

import { DataService } from './data.service';

import { Post } from './post.model';

import {HttpTestingController,HttpClientTestingModule} from '@angular/common/http/testing';

describe('DataService', () => {

  let service: DataService;

  let httpMock: HttpTestingController;

  beforeEach(() => {

      TestBed.configureTestingModule({

          imports: [HttpClientTestingModule],

          providers: [DataService]

      });

      service = TestBed.get(DataService);

      httpMock = TestBed.get(HttpTestingController);

  });

  afterEach(() => {

    httpMock.verify();

});

  it('should be created Data Service', () => {

    const service: DataService = TestBed.get(DataService);

    expect(service).toBeTruthy();

  });

  it('be able to retrieve posts from the API bia GET', () => {

    const dummyPosts: Post[] = [

      {

        userId: '1',

        id: 1,

        body: 'Welcome User 1',

        title: 'This is User 1'

        },

        {

        userId: '2',

        id: 2,

        body: 'Welcome User 2',

        title: 'esting AngulTar ServThis is User 2ices'

    }];

    service.getPostData().subscribe(posts => {

        expect(posts.length).toBe(2);

        expect(posts).toEqual(dummyPosts);

    });

const request = httpMock.expectOne( `${service.ROOT\_URl}/posts`);

expect(request.request.method).toBe('GET');

request.flush(dummyPosts);

});

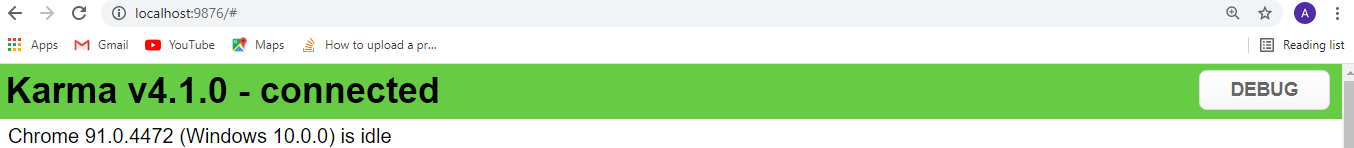
});

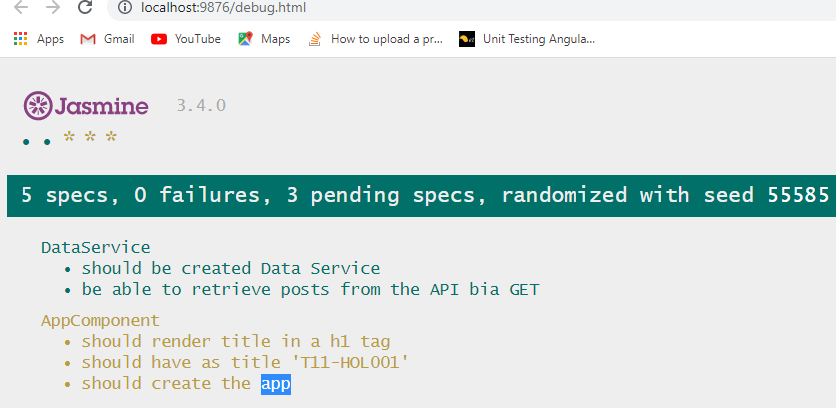
***Note:*** Here, we subscribe on the method of service getPostData() and first we expect that the length of the posts is 2 or not and in another expect we ensure that the post we get from the route is equal to the dummy posts.

At this point we don’t want to hit the actual HTTP request for the data, So for that, we need to mock the HTTP request and import HttpTestingController which comes from the @angular/common/http/testing.

1. Run the command ng test in cli and click the debug button of Karma window in browser to see the jasmine test results.

**Sample output is given for your reference:**

****

****