AXON ACTIVE
Focusing on your agility

Axon Active Vietnam Co., Ltd.
Hai Au Building, 39B Truong Son St.
Ward 4, Tan Binh District
Ho Chi Minh City
Vietnam
www.axonactive.com
info@axonactive.com

Axon Active Schweiz AG
Schlössli Schönegg
Wilhelmshöhe
6003 Luzern
Switzerland
www.axonactive.ch
info@axonactive.ch

Other Branches
Da Nang
214 30/4 Street, Hai Chau District, Da Nang, Vietnam
Can Tho
57-59A CMT8 Street, Ninh Kieu District, Can Tho, Vietnam
San Francisco
281 Ellis Street, San Francisco, California 94102, USA

TESTING FOR ANGULAR

Author: *By VAMOS Team*
Date: January 23, 2024

I. Introduction

Angular uses Jasmine and Karma framework to help write unit tests for an application. These dependencies will be automatically installed if you create an Angular project with CLI.

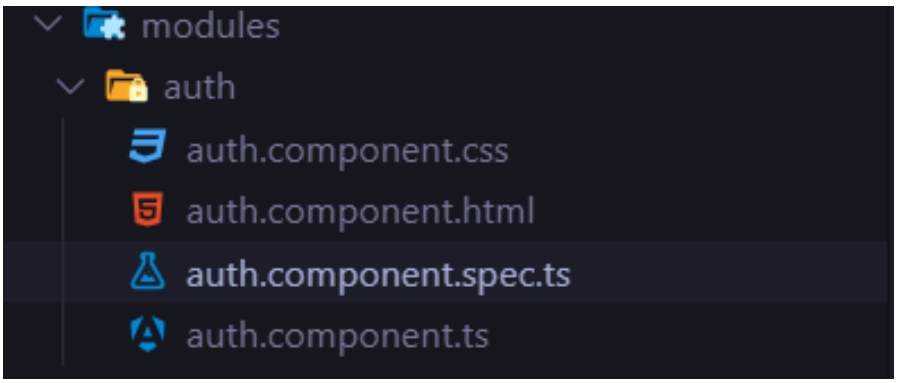
II. Configuration

- The Angular CLI takes care of Jasmine and Karma configuration for you. It constructs the full configuration in memory, based on options specified in the angular.json file.
- You can customize Karma through karma.conf.js. This file can be create by running the following command:

```
ng generate config karma
```

III. Test file convention

- The test file extension must be .spec.ts so that Angular can identify it as a file with tests (also known as a spec file). The test file should be named after the component or service you are testing on.
- Put unit test spec files in the same folder as the application source code files that they test



Axon Active Vietnam Co., Ltd.
Hai Au Building, 39B Truong Son St.
Ward 4, Tan Binh District
Ho Chi Minh City
Vietnam
www.axonactive.com
info@axonactive.com

Axon Active Schweiz AG
Schlössli Schönegg
Wilhelmshöhe
6003 Luzern
Switzerland
www.axonactive.ch
info@axonactive.ch

Other Branches
Da Nang
214 30/4 Street, Hai Chau District, Da Nang, Vietnam
Can Tho
57-59A CMT8 Street, Ninh Kieu District, Can Tho, Vietnam
San Francisco
281 Ellis Street, San Francisco, California 94102, USA

IV. Setup test

1/ Test suite

Each test must have at least one test suite. A suite is declared with **describe** block

```
describe('Suite description', () => {
  /* … */
});
```

describe is a function that takes two params:

- A string describing the name of the class under test
- A function containing the code for the unit test

Each suit consists of one or more specifications, or short, specs. A spec is declared with an **it** block:

```
describe('Suite description', () => {
  it('Spec description', () => {
    /* … */
  });
  /* … more specs …  */
});
```

it also takes two params, a string name and a function holding the spec code

2/ TestBed

Axon Active Vietnam Co., Ltd.
Hai Au Building, 39B Truong Son St.
Ward 4, Tan Binh District
Ho Chi Minh City
Vietnam
www.axonactive.com
info@axonactive.com

Axon Active Schweiz AG
Schlössli Schönegg
Wilhelmshöhe
6003 Luzern
Switzerland
www.axonactive.ch
info@axonactive.ch

Other Branches
Da Nang
214 30/4 Street, Hai Chau District, Da Nang, Vietnam
Can Tho
57-59A CMT8 Street, Ninh Kieu District, Can Tho, Vietnam
San Francisco
281 Ellis Street, San Francisco, California 94102, USA

The TestBed creates and configures an Angular environment so you can test particular application parts like Components and Services easily.
TestBed works like a testing module that you will need to declare components, directives; provide needed services as well as import necessary modules to run the test.
We can configure it using **configureTestingModule** static method

```
describe('auth component testing', () => {
  let component: AuthComponent;
  let fixture: ComponentFixture<AuthComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      imports: [FormsModule, HttpClientTestingModule],
      declarations: [AuthComponent],
      providers: [AuthService]
    })
    .compileComponents();
```

Finally we call the **compileComponents** method to compile all declared components into Javascript code.

3/ Rendering component

The component under test can be rendered using **createComponent** which returns a fixture holding the Component and provides a convenient interface to both the Component instance and the rendered DOM.

```
fixture = TestBed.createComponent(AuthComponent);
component = fixture.componentInstance;
fixture.detectChanges();
```

The fixture references the Component instance via the componentInstance property.

Axon Active Vietnam Co., Ltd.
Hai Au Building, 39B Truong Son St.
Ward 4, Tan Binh District
Ho Chi Minh City
Vietnam
www.axonactive.com
info@axonactive.com

Axon Active Schweiz AG
Schlössli Schönegg
Wilhelmshöhe
6003 Luzern
Switzerland
www.axonactive.ch
info@axonactive.ch

Other Branches
Da Nang
214 30/4 Street, Hai Chau District, Da Nang, Vietnam
Can Tho
57-59A CMT8 Street, Ninh Kieu District, Can Tho, Vietnam
San Francisco
281 Ellis Street, San Francisco, California 94102, USA

In our testing environment, there is no automatic change detection. Therefore, we have to trigger the change detection manually through the **detectChanges** method.

V. Some unit test examples

1/ Testing a simple component

In this test, we expect that there is a form for users to login after the component is rendered. If the 'form' element is defined, the test passes, otherwise it fails.

```
it('should render a form for user to login', () => {
  const { debugElement } = fixture;
  const form = debugElement.query(By.css('form'));
  expect(form).toBeDefined();
});
```

debugElement is a wrapper of the native DOM element which provides handy functions to work with the DOM element itself. Next, we will perform a css query using the query method of **debugElement** to get the form element and examine its existence.

2/ Testing a service with mock

Here we will write some unit tests for this AuthService.

```
export class AuthService {
  public token: string;

  constructor(private http: HttpClient) {}

  login(user: Omit<User, 'id'>): Observable<{ token: string }> {
    return this.http
      .post<{ token: string }>(`${environment.urlPath}/login`, user)
      .pipe();
  }

  getCurrentUser(): any {
    const token = localStorage.getItem('token');
    const user = JSON.parse(atob(token.split('.')[1]));
    return user;
  }
}
```

Axon Active Vietnam Co., Ltd.
Hai Au Building, 39B Truong Son St.
Ward 4, Tan Binh District
Ho Chi Minh City
Vietnam
www.axonactive.com
info@axonactive.com

Axon Active Schweiz AG
Schlössli Schönegg
Wilhelmshöhe
6003 Luzern
Switzerland
www.axonactive.ch
info@axonactive.ch

Other Branches
Da Nang
214 30/4 Street, Hai Chau District, Da Nang, Vietnam
Can Tho
57-59A CMT8 Street, Ninh Kieu District, Can Tho, Vietnam
San Francisco
281 Ellis Street, San Francisco, California 94102, USA

Notice that this service has a method **login** which sends a HTTP POST request to some URL. In unit test, we won't make a HTTP request directly but try to create a faking request instead.
Angular provides some useful library to help us simulate http requests

```
// Http testing module and mocking controller
import { HttpClientTestingModule, HttpTestingController, TestRequest } from '@angular/
common/http/testing';
import { HttpClient } from '@angular/common/http';
```

Import necessary module to the TestBed and get mock-up objects through the **inject** method

```
beforeEach(() => {
    TestBed.configureTestingModule({
        imports: [HttpClientTestingModule]
    });

    //Inject a mock http client and controller for each test
    httpClient = TestBed.inject(HttpClient);
    httpTestingController = TestBed.inject(HttpTestingController);
    authServVice = new AuthService(httpClient);

    user = {
      id: 1,
      email: 'example@axonactive.com',
      password: 'Example@123'};
```

We expect that the **login** method will make a POST request to any server having the /login endpoint

```
it('should make a POST request to /login endpoint when calling login method', () => {
    authServVice.login(user).subscribe(() => {
    });

    expect(httpTestingController.expectOne((req) => {
      return req.url.match('.*/login') && req.method === 'POST';
    })).toBeInstanceOf(TestRequest);

});
```

The **expectOne** method finds the pending request by the given criteria and returns a **TestRequest** instance if there is a match, otherwise it throws an exception.

Axon Active Vietnam Co., Ltd.
Hai Au Building, 39B Truong Son St.
Ward 4, Tan Binh District
Ho Chi Minh City
Vietnam
www.axonactive.com
info@axonactive.com

Axon Active Schweiz AG
Schlössli Schönegg
Wilhelmshöhe
6003 Luzern
Switzerland
www.axonactive.ch
info@axonactive.ch

Other Branches
Da Nang
214 30/4 Street, Hai Chau District, Da Nang, Vietnam
Can Tho
57-59A CMT8 Street, Ninh Kieu District, Can Tho, Vietnam
San Francisco
281 Ellis Street, San Francisco, California 94102, USA

For the **getCurrentUser** method, it should return an user when there is already a token stored in the localStorage

```
it('should return an user when parsing token stored in local storage', () => {
  const exampleToken = 'eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
  eyJpc3MiOiJBZ2lsZSBDb3Vyc2UiLCJpYXQiOjE3MDU3NDc5NjIsImV4cCI6MTcxMDg0NTU2MiwiYXVkIjoi
  d3d3LmV4YW1wbGUuY29tIiwic3ViIjoiZXhhbXBsZUBheG9uYWN0aXZlLmNvbSIsIkVtYWlsIjoiZXhhbXBs
  ZUBheG9uYWN0aXZlLmNvbSIsIlJvbGUiOiJ1c2VyIn0.
  wfOffX_xvxvx02IGju01dIJhzJA0ey07Z2tALrPuyNI';

  spyOn(localStorage, 'getItem').and.returnValue(exampleToken);
  expect(authService.getCurrentUser()).not.toBeUndefined();

})
```

We won't access directly to the localStorage in our browser but instead use our simulated localStorage that returns a given JWT token when calling **getItem** method.

VI. Run the test

We can execute our written unit test by simply running the command: **ng test** or **ng test --code-coverage** if you want to display the test coverage percentage.

```
✓ Browser application bundle generation complete.
24 01 2024 02:33:56.532:WARN [karma]: No captured browser, open http://localhost:9876/
24 01 2024 02:33:56.553:INFO [karma-server]: Karma v6.4.2 server started at http://localhost:9876/
24 01 2024 02:33:56.554:INFO [launcher]: Launching browsers Chrome with concurrency unlimited
24 01 2024 02:33:56.569:INFO [launcher]: Starting browser Chrome
24 01 2024 02:33:58.177:INFO [Chrome 120.0.0.0 (Windows 10)]: Connected on socket smpZmv3vzX9Z868CAAAB with id 8323273
1
Chrome 120.0.0.0 (Windows 10): Executed 5 of 5 SUCCESS (0.63 secs / 0.135 secs)
TOTAL: 5 SUCCESS

=============================== Coverage summary ===============================
Statements   : 75% ( 24/32 )
Branches     : 100% ( 0/0 )
Functions    : 58.82% ( 10/17 )
Lines        : 73.33% ( 22/30 )
================================================================================
```

Axon Active Vietnam Co., Ltd.
Hai Au Building, 39B Truong Son St.
Ward 4, Tan Binh District
Ho Chi Minh City
Vietnam
www.axonactive.com
info@axonactive.com

Axon Active Schweiz AG
Schlössli Schönegg
Wilhelmshöhe
6003 Luzern
Switzerland
www.axonactive.ch
info@axonactive.ch

Other Branches
Da Nang
214 30/4 Street, Hai Chau District, Da Nang, Vietnam
Can Tho
57-59A CMT8 Street, Ninh Kieu District, Can Tho, Vietnam
San Francisco
281 Ellis Street, San Francisco, California 94102, USA

It will open a browser to show the test result. The default browser is Chrome.