# Angular 7 Unit Testing

selvaraj chinnasamy Follow
Oct 25, 2018 · 4 min read

These type of testing sometimes called **Isolated Testing,** because this testing is used to test a **small isolated** piece of code.

If your test is using some external resources like a network or database then that is not unit testing.

We can test our angular application simply by using normal javascript, get the instance of the class and use that instance, test your functions and we can check expected return value to actual result.

Since testing is the common thing, there are a number of libraries and tools such as JSUnit, Unit.js, QUnit, Jasmine, Karma, Mocha etc…
Here we are going to use **Jasmine** and **Karma** to test our **Angular 7 Application**.

**Jasmine**
Jasmine is an open source testing framework for JavaScript.
Here we are not going to discuss much on **Jasmine.** If you want to know more about Jasmine you can make use this link: https://jasmine.github.io/.
Before starting you need to understand the basics of **Jasmine.**

**describe**- is the function that has the collection of individual test spec
now, what is the test spec?
it just has one or more test expectation
Example:

```
// describe
describe("Hello World", function() {

// test spce
 it("to be equal to hello world", function() {
```

```
// expectations
 expect(foo()).toEqual('hello world');
 });
});
```

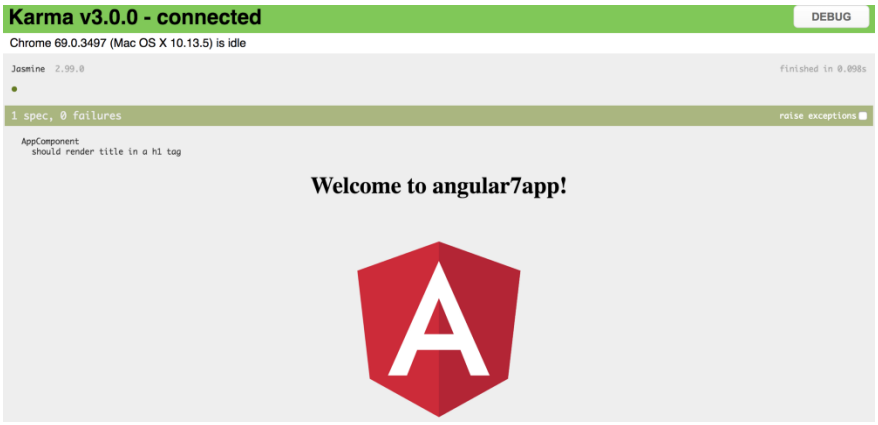
## Setup and teardown

Sometimes in the real-world application before performing or after performed our test case we need to insert some mock data or we need to do some cleaning activity, for these purposes we have
**beforeAll**- This function is called **once** before all the specs in the test suite are run.
**afterAll**- This function is called **once** after all the specs in a test suite are finished.
**beforeEach**- This function is called before each test specs.
**afterEach**- This function is called after each test specs.

```
describe('Hello world', () => {
  let expected = "";
  beforeEach(() => {
    // some cleaning or insert operation
  });
  afterEach(() => {
    // some cleaning operetions
  });
  it('says hello', () => {
    expect(helloWorld())
        .toEqual(expected);
  });
});
```

**Karma**

It is just a test runner.
It is a tool which lets us spawn browsers and run jasmine tests inside of them all from the command line. The results of the tests are also displayed on the command line.




**Let's start into Angular 7**

**pre-request**

you should have your angular 7 application running if not do this

```
npm install -g @angular/cli
ng new <project name>
```

if run these comments angular will create a project for you with some default test specs like this

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

import { RouterTestingModule } from
'@angular/router/testing';

import { AppComponent } from './app.component';

describe('AppComponent', () => {

beforeEach(async(() => {

TestBed.configureTestingModule({

imports: [

RouterTestingModule

],

declarations: [

AppComponent
```

```
  ],

  }).compileComponents();

  }));

  it('should create the app', () => {

  const fixture = TestBed.createComponent(AppComponent);

  const app = fixture.debugElement.componentInstance;

  expect(app).toBeTruthy();

  });

  it(`should have as title 'angular7app'`, () => {

  const fixture = TestBed.createComponent(AppComponent);

  const app = fixture.debugElement.componentInstance;

  expect(app.title).toEqual('angular7app');

  });

  it('should render title in a h1 tag', () => {

  const fixture = TestBed.createComponent(AppComponent);

  fixture.detectChanges();

  const compiled = fixture.debugElement.nativeElement;

  expect(compiled.querySelector('h1').textContent).toContain('
  Welcome to angular7app!');

  });

  });
```

here you can see every jasmine functions that we discussed. If you run `npm test`

karma will open your browser where you can see test results

### Let’s test class

Everything in Angular is an instance of a class, be it a Component, Directive, Pipe and so on. So once you know how to test a basic class you can test everything. let say we have an app component `app.component.ts`

```
export class AppComponent {
title = 'angular7app';
}
```

let's start writing test cases

```
import { AppComponent } from './app.component'; // step 1
import { TestBed, async } from '@angular/core/testing';
import { RouterTestingModule } from
'@angular/router/testing';
describe('AppComponent', () => {
beforeEach(async(() => {
TestBed.configureTestingModule({
imports: [
RouterTestingModule // modules
],
declarations: [
AppComponent // components
],
}).compileComponents();
}));
it('should render title in a h1 tag', () => {
const fixture = TestBed.createComponent(AppComponent);
```

```
fixture.detectChanges();

const compiled = fixture.debugElement.nativeElement;

expect(compiled.querySelector('h1').textContent).toContain('
Welcome to angular7app!');

});

});
```

**Angular Testbed (ATB)**

The Angular Test Bed (ATB) is a higher level *Angular Only* testing framework that allows us to easily test behaviors that depend on the Angular Framework.

```
TestBed.configureTestingModule({

imports: [RouterTestingModule],

declarations: [AppComponent],

}).compileComponents();

}));
```

we need to configure testbed, here we need to declare components that we going to test.

**ATB features**

- It allows us to test the interaction of a directive or component with its template.
- It allows us to easily test change detection.
- It allows us to test and use Angulars DI framework.
- It allows us to test using the `NgModule` configuration we use in our application.
- It allows us to test user interaction via clicks & input fields.

Hope that you got some basic understanding in Angular Unit testing. this is just a basic, In the next story, I will explain how to test components in in-depth.