**Jasmine test framework** : Default unit testing framework in Angular

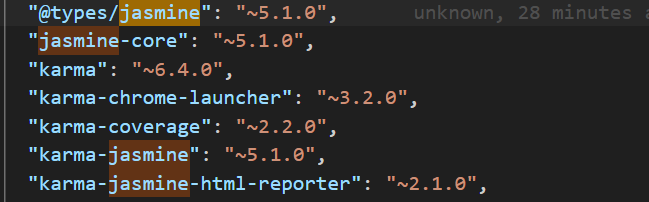
It checks .spec.ts files

<https://angular.dev/guide/testing>

<https://angular.dev/cli/test>

<https://angular.dev/guide/testing/components-scenarios#testing-with-a-spy>

In package.json



**Key Concepts**:

**TestBed**: Angular's testing utility that helps create a test module (like **NgModule** but for testing).

**ComponentFixture**<T>: A wrapper around the component that allows access to its instance and the DOM.

**describe**(): A Jasmine function used to define a **test suite** (a group of related tests).

**FirstComponentComponent**: A label for this test suite, typically the name of the component being tested.

**component**: A reference to the actual component instance.

**fixture**: A testing wrapper that helps control and inspect the component.

**it()** is a **Jasmine function** for defining a test case.

**Test Description**: 'should create the component' (explains what the test does).

expect(component).toBeTruthy();:

* expect() is a **Jasmine assertion function**.
* .toBeTruthy() checks if the component is successfully created (true if it exists)

**Matchers** are functions in Jasmine that compare the actual test result with an expected value. Types:

Truthy & Falsy Matchers :toBeTruthy(), toBeFalsy()

Equality Matchers: toBe(), toEqual()

Comparision Matchers:toBeGreaterThan()

String & Array Matchers:toContain(x)

Spy Matchers: toHaveBeenCalled()

**beforeEach()**: Runs before each test (it() block).

TestBed.configureTestingModule({...}):

* Creates a test module that declares FirstComponentComponent.
* Mocks dependencies and sets up the testing environment.

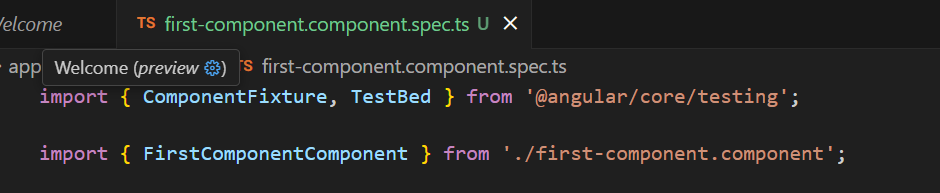
.compileComponents(): Ensures the component's template and styles are compiled **before running the test**.

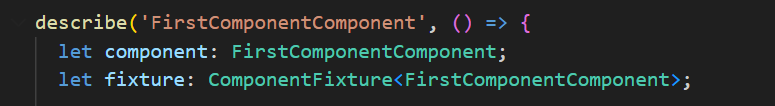
**mock** is a fake version of a real object (service, API, or dependency) used in testing. Instead of calling actual implementations, we provide **dummy data** to control test behavior.

**spy** is a Jasmine feature that tracks function calls (how many times they were called, with what arguments, etc.).  
Instead of replacing the whole service like a mock, a **spy just observes** a method.

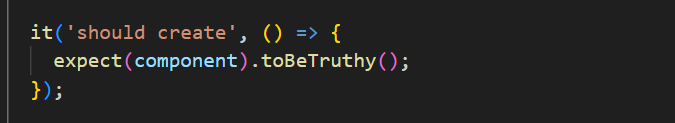
spy object is considered a "partial mock," it wraps around a real object and allows you to monitor specific interactions with it,

**spyOn()** allows us to **track and mock method calls** on services, components, or objects.

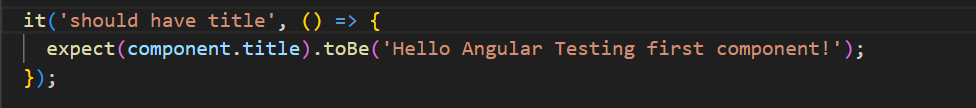




First test case:



Can add more test case

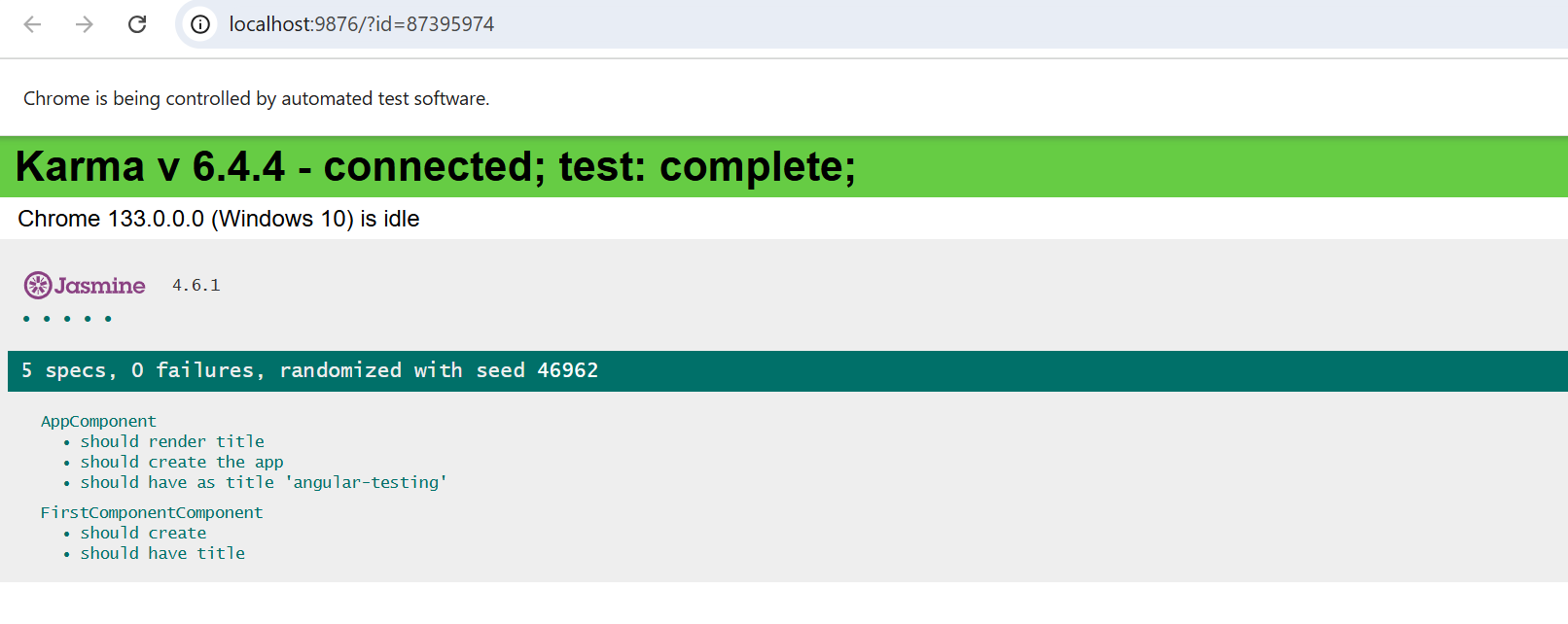


Run the test:

ng test

This will:

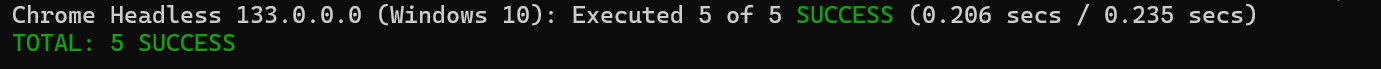
* Start **Karma**, which opens a browser.
* Run all tests inside \*.spec.ts files.
* Display test results in the console.



**(Optional) Run Tests in Headless Mode**

If you don’t want the browser to open, use:

ng test --watch=false --browsers=ChromeHeadless



Generate a **test coverage report**

ng test --no-watch --code-coverage

**ng test**

* Runs the unit tests in your Angular application.

**--no-watch**

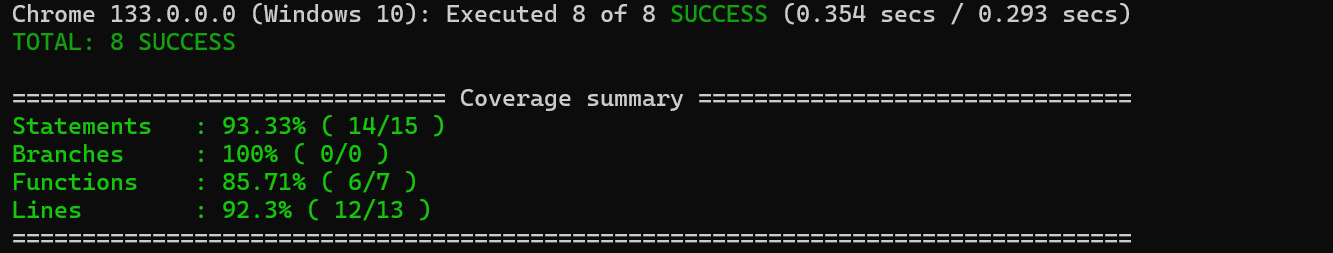
* Normally, ng test **watches for file changes** and re-runs tests automatically.
* Adding --no-watch **disables the watch mode**, meaning:
  + The tests **run once** and then **exit**.

**--code-coverage**

* Generates a **test coverage report**.
* Shows **how much of your code is covered** by tests.
* The report is stored in:

coverage/index.html

* You can open this file in a browser to see:
  + **Statements** covered
  + **Branches** tested
  + **Functions** tested
  + **Lines** covered



**Using --include to Run a Single Component's Tests**

ng test --include=src/app/first-component/first-component.component.spec.ts

ng test --include=src/app/post.service.spec.ts

**Testing in Service:**

Use HttpTestingController to **mock API responses in tests**.

