**Unit testing in Angular**

Inhoud

[Introduction ( 1 - 2) 2](#_Toc46781733)

[AAA 2](#_Toc46781734)

[Testing principals 2](#_Toc46781735)

[Isolated Unit tests 2](#_Toc46781736)

[Mocking a service 2](#_Toc46781737)

# Introduction ( 1 - 2)

## AAA

Unit test should have the following structure:

* Arrange: all necessary preconditions and inputs
* Act: on the object or class under test
* **A**ssert: that the expected results have occurred.

This is anolog to given, when, then

## Testing principals

DRY (don’t repeat yourself) vs DAMP (repeat yourself if necessary)

Repeat if necessary applies to situations where, for instance the setup is very critical to the story. In that case the setup is not done in the beforeEach but in the it(‘shoul…’, () => {}

# Isolated Unit tests

You can look at isolated test as if it’s not an angular class but just a piece of javascript. (in your mind) remove all the things like @Component, @Pipe, @Input etc.

## Mocking a service

Test delete method of the HeroComponent (depending on a service)

See: the Herocomponent.

* The HeroComponent has a heroes property. To test the delete function we have to populate heroes first.
* The HeroComponent is depending on the HeroService which is injected into the contructor.
* To be independent of the HeroService we have to Mock it. Jasmine provides craeateSpyObj()  
  which takes an array of methodnames of the HeroService we want to use.
* As the heroService.deleteHero(…) returns an observable so should our mockHeroService. We can achieve that by returning an subject when calling deleteHero using the “of” method, like so:  
  mockHeroService.deleteHero.and.returnValue(of(true));
* The ‘should delete …..’ test is setup following the AAA pricipel.

The code for this test with some highlights:

import { Hero } from "../hero";

import { HeroesComponent } from "./heroes.component";

import { of } from "rxjs";

describe('HeroesComponent', () => {

let heroesComponent: HeroesComponent;

let HEROES;

let mockHeroService;

beforeEach(() => {

HEROES = [

{ id: 1, name: 'pietje', strength: 25 },

{ id: 2, name: 'jantje', strength: 75 },

{ id: 3, name: 'klaasje', strength: 10 }

]

mockHeroService = jasmine.createSpyObj(['addHero', 'getHeroes', 'deleteHero'])

heroesComponent = new HeroesComponent(mockHeroService);

});

it('should delete a hero', () => {

// **Arrange**: init HEROES: is already done by beforeEach

heroesComponent.heroes = HEROES;

// make the mockHeroService.delete retrurn a observable

mockHeroService.deleteHero.**and.returnValue**(**of**(true));

// **Act**: delete a hero

heroesComponent.delete(HEROES[2]);

// **Assert**: HEROES contains two elements now

expect(heroesComponent.heroes.length).toBe(2);

expect(heroesComponent.heroes[0].id).toBe(1);

expect(heroesComponent.heroes[1].id).toBe(2);

})

})

# Appendix

Code source: <https://github.com/joeeames/PSAngularUnitTestingCourse>

Course: <https://app.pluralsight.com/library/courses/unit-testing-angular/table-of-contents>