09. Spring Boot Test

## Inleiding Spring Boot Starter Test

Spring Boot Starter Test is een module binnen de Spring Boot framework-familie waarmee ontwikkelaars eenvoudig unit-, integratie- en functionele tests kunnen schrijven voor hun Spring Boot-applicaties. Het biedt een uitgebreide set hulpmiddelen om testen te schrijven die kunnen worden uitgevoerd op verschillende niveaus van de applicatiestapel, waaronder controllers, services en repositories. Deze module biedt ook ondersteuning voor het testen van RESTful-web services en integratietests met behulp van in-memory databases.

Het schrijven van tests is een belangrijk aspect van softwareontwikkeling en het gebruik van Spring Boot Starter Test kan het testproces aanzienlijk vereenvoudigen en versnellen. Deze module maakt het mogelijk om tests te schrijven met minimale configuratie en biedt ook uitgebreide rapportage- en assertiemogelijkheden. Het maakt het mogelijk om tests te schrijven die onafhankelijk zijn van externe bronnen, waardoor het testproces sneller en betrouwbaarder wordt.

In deze tutorial zullen we stap voor stap laten zien hoe je de Spring Boot Starter Test-module kunt gebruiken om tests te schrijven voor een repository in een Spring Boot-applicatie. We zullen voorbeelden geven van CRUD-operaties (create, read, update, delete) en integratietests met behulp van een in-memory database. We zullen ook geavanceerde testtechnieken behandelen die beschikbaar zijn in Spring Boot Starter Test, waaronder het gebruik van geautomatiseerde tests en het testen van RESTful-web services.

## Stappen

Hier zijn de stappen om Spring Boot Starter Test te gebruiken:

1. Voeg de Spring Boot Starter Test-afhankelijkheid toe aan uw Maven of Gradle-buildfile.
2. Schrijf tests voor uw applicatie met behulp van JUnit of een ander testframework zoals Mockito.
3. Gebruik de @RunWith-annotatie van JUnit om de SpringRunner-klasse te gebruiken.
4. Gebruik de @SpringBootTest-annotatie om de Spring Boot-context te laden voor uw test.
5. Gebruik de @Autowired-annotatie om de componenten te injecteren die u wilt testen.
6. Schrijf tests voor uw repository met behulp van Spring Boot-testhulpmiddelen zoals @DataJpaTest.
7. Voer uw tests uit en bekijk de rapportage.

## Voordelen

Een van de voordelen van Spring Boot Starter Test is dat het zorgt voor een snelle feedbackloop bij het ontwikkelen van software. Door het schrijven van tests kunnen ontwikkelaars problemen snel opsporen en oplossen voordat de code naar productie gaat. Dit zorgt voor een hogere kwaliteit van de code en een betere gebruikerservaring.

Een ander voordeel van Spring Boot Starter Test is dat het ontwikkelaars in staat stelt om tests te schrijven die onafhankelijk zijn van externe bronnen. Dit betekent dat tests kunnen worden uitgevoerd zonder dat er een database of andere services nodig zijn, wat het testproces sneller en betrouwbaarder maakt.

## Mockito

Mockito is een van de meest gebruikte testframeworks binnen Spring Boot Starter Test. Het stelt ontwikkelaars in staat om mock-objecten te maken en te gebruiken in hun tests, waardoor het mogelijk wordt om bepaalde delen van de code te isoleren en te testen zonder dat er externe bronnen nodig zijn. Dit maakt het testproces sneller en meer geautomatiseerd.

## JUnit VS Mockito

JUnit en Mockito zijn beide testframeworks die vaak worden gebruikt in combinatie met Spring Boot Starter Test. Het verschil tussen JUnit en Mockito is dat JUnit zich richt op het uitvoeren van testcases en het rapporteren van fouten, terwijl Mockito zich richt op het maken van mock-objecten om te gebruiken in tests.

Een voorbeeld van een JUnit-testcase zou kunnen zijn:

import static org.junit.Assert.assertEquals;  
import org.junit.Test;  
  
public class MyTest {  
   
 @Test  
 public void testSum() {  
 MyClass myClass = new MyClass();  
 int result = myClass.sum(2, 3);  
 assertEquals(5, result);  
 }  
}  
Copy

Deze testcase test de sum-methode van de MyClass-klasse en controleert of de verwachte uitkomst (5) wordt geretourneerd.

Een voorbeeld van het gebruik van Mockito in combinatie met JUnit zou kunnen zijn:

import static org.mockito.Mockito.\*;  
import static org.junit.Assert.\*;  
import org.junit.\*;  
   
public class MyTest {  
   
 @Test  
 public void test() {  
 // Maak een mock-object van MyClass  
 MyClass myClass = mock(MyClass.class);  
   
 // Definieer het verwachte gedrag van de mock  
 when(myClass.sum(2, 3)).thenReturn(5);  
   
 // Roep de methode aan die de mock gebruikt  
 Calculator calculator = new Calculator(myClass);  
 int result = calculator.calculate(2, 3);  
   
 // Controleer of het verwachte resultaat is geretourneerd  
 assertEquals(5, result);  
 }  
}  
Copy

Dit voorbeeld toont hoe Mockito kan worden gebruikt om een mock-object van MyClass te maken en het verwachte gedrag van de sum-methode te definiëren. Vervolgens wordt een instantie van de Calculator-klasse gemaakt die de mock-object gebruikt om de berekening uit te voeren. Ten slotte wordt gecontroleerd of de verwachte uitkomst wordt geretourneerd.

Kortom, JUnit wordt gebruikt om testcases uit te voeren en fouten te rapporteren, terwijl Mockito wordt gebruikt om mock-objecten te maken en het gedrag van die objecten te definiëren voor gebruik in tests.

## Bronnen

Roadmap en leerbronnen voor Spring Boot Starter Test Module:

1. Basisbeginselen van Spring Boot en Spring Boot Starter Test Module:
   * Officiële documentatie van Spring Boot: <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>
   * Baeldung-tutorial voor Spring Boot Starter Test Module: <https://www.baeldung.com/spring-boot-testing>
2. Testen van Repository in Spring Boot:
   * Officiële documentatie van Spring Data JPA: <https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#reference>
   * Baeldung-tutorial voor testen van Repository in Spring Boot: <https://www.baeldung.com/spring-data-jpa-test-in-memory-database>
3. Integratietests met Spring Boot:
   * Officiële documentatie van Spring Boot voor integratietests: <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#boot-features-testing>
   * Baeldung-tutorial voor integratietests met Spring Boot: <https://www.baeldung.com/spring-boot-testing>
4. Testen van RESTful Web Services in Spring Boot:
   * Officiële documentatie van Spring Boot voor het testen van RESTful Web Services: <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#boot-features-testing-spring-boot-applications-testing-web-apis>
   * Baeldung-tutorial voor het testen van RESTful Web Services in Spring Boot: <https://www.baeldung.com/spring-boot-testresttemplate>
5. Geavanceerde testtechnieken in Spring Boot:
   * Officiële documentatie van Spring Boot voor geavanceerde testtechnieken: <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#boot-features-testing-spring-boot-applications-testing-autoconfigured-tests>
   * Baeldung-tutorial voor geavanceerde testtechnieken in Spring Boot: <https://www.baeldung.com/spring-boot-testing-autoconfigured-tests>

Leerbronnen:

* Spring Boot Starters: <https://start.spring.io/>
* Spring Boot en Spring Boot Starter Test Module-video's van het officiële YouTube-kanaal van Spring: <https://www.youtube.com/user/SpringSourceDev/search?query=Spring+Boot+Testing>
* Spring Boot and Testing in Java-video's van Java Brains op YouTube: <https://www.youtube.com/playlist?list=PLqq-6Pq4lTTa9YGfyhyW2CqdtW9RtY-I3>

Om verder te leren over Spring Boot versie 3 kijk naar de YT-video van de volgende link:  
Java Pub House: <https://www.javapubhouse.com/2023/02/episode-102-oh-my-spring-boot-3-is-out-an-interview-with-dan-vega-from-the-pivotal-team.html>

Met deze roadmap en leerbronnen kun je aan de slag gaan met het testen van je Spring Boot-applicaties met behulp van de Spring Boot Starter Test Module.