**Junit and Mockito**

## **Maven Dependencies**

The spring-boot-starter-test dependency includes [JUnit 5 dependencies, and Mockito. So we only need to include this dependency](https://howtodoinjava.com/junit5/junit5-maven-dependency/).

**pom.xml**

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope></dependency>

## **Setup**

In this example, we are unit testing primarily two classes PolicyServiceImpl and PolicyRepository. As the name implies, the manager class represents the service layer, and dao class interacts with the database.

PolicyServiceImpl class has a dependency on PolicyRepositoryand delegate method calls to get the data which is finally returned to controller classes.

To test the methods in PolicyServiceImpl , we can create a JUnit test class PolicyServiceImplTest in below given way:

@RunWith(MockitoJUnitRunner.**class**)

@SpringBootTest

**public** **class** PolicyServiceImplTest {

@InjectMocks

PolicyServiceImpl policyService;

@Mock

PolicyRepository policyRepository;

## **2.1 @Mock vs @InjectMocks**

* The @Mock annotation creates a mock implementation for the class it is annotated with.
* @InjectMocks also creates the mock implementation, additionally injects the dependent mocks that are marked with the annotations @Mock into it.

In the above example, we have annotated PolicyServiceImpl class with @InjectMocks, so mockito will create the mock object for PolicyServiceImpl class and inject the mock dependency of PolicyRepository into it.

## **JUnit Tests using Mockito**

* Let’s see a few examples of writing the junit tests to unit test the service layer methods using mock objects created with mockito.

package com.cybage.genworth.insurance;

import static org.junit.jupiter.api.Assertions.assertEquals;

import static org.mockito.Mockito.when;

import java.util.ArrayList;

import java.util.List;

import java.util.Optional;

import org.junit.Test;

import org.junit.runner.RunWith;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.internal.verification.Times;

import org.mockito.junit.MockitoJUnitRunner;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.boot.test.mock.mockito.MockBean;

import org.springframework.test.context.junit4.SpringRunner;

import com.cybage.genworth.insurance.model.Policy;

import com.cybage.genworth.insurance.repository.PolicyRepository;

import com.cybage.genworth.insurance.service.PolicyService;

import com.cybage.genworth.insurance.serviceImpl.PolicyServiceImpl;

@RunWith(MockitoJUnitRunner.class)

@SpringBootTest

public class PolicyServiceImplTest {

@InjectMocks

PolicyServiceImpl policyService;

@Mock

PolicyRepository policyRepository;

@Test

public void savePolicyTest() {

Policy policy = new Policy(3, "1255", "500", "ram", "ram@gmail.com", "5012", "active");

when(policyRepository.save(policy)).thenReturn(policy);

assertEquals(policy, policyService.savePolicy(policy));

}

@Test

public void getAllPolicyTest() {

List<Policy> list = new ArrayList<>();

Policy policy1 = new Policy(1, "101", "1000", "Ram", "ram@gmail.com", "2020", "active");

Policy policy2 = new Policy(2, "102", "2000", "Shyam", "shyam@gmail.com", "2021", "active");

Policy policy3 = new Policy(3, "103", "3000", "Rahul", "rahul@gmail.com", "2022", "active");

list.add(policy1);

list.add(policy2);

list.add(policy3);

when(policyRepository.findAll()).thenReturn(list);

Iterable<Policy> policyList = policyService.getAllPolicy();

assertEquals(list, policyList);

}

@Test

public void getPolicyByIdTest() {

when(policyRepository.findById(3))

.thenReturn(new Policy(3, "103", "1000", "Shyam", "shyam@gmail.com", "2022", "active"));

Policy policy = policyService.getPolicyById(3);

assertEquals("103", policy.getPolicyNumber());

assertEquals("Shyam", policy.getName());

assertEquals("active", policy.getStatus());

}

@Test

public void deletePolicyTest() {

Policy policy = new Policy(1, "101", "1000", "Ram", "ram@gmail.com", "2020", "active");

policyRepository.save(policy);

policyRepository.deleteById(policy.getId());

Policy exp = policyRepository.findById(policy.getId());

assertEquals(null, exp);

}

}

**The unit test results will be like this.**

