**Junit and Mockito**

**Step-1- Adding 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/).

<dependency>

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

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

<scope>test</scope>

</dependency>

**Step-2- Tested Save Policy Operation.**

In this example, need to test the below classes - PolicyServiceImplTest and PolicyRepository.

**Note-**

1. Here PolicyServiceImplTest class represents the service layer and dao class represents repository layer means interacts with the database.
2. PolicyServiceImplTest class has a dependency on PolicyRepository and delegate method calls to get the data which is finally returned to controller classes.
3. Pre-requisites things are

-Create Spring Boot Project

-Create Model Class

-Create Service and its implementation

-Create Repository

**Step-3- For testing purpose, need to create the Junit test class**

As PolicyServiceImplTest.Java inject the policyService and PolicyRepository object into that class as per below

@RunWith(MockitoJUnitRunner.class)

@SpringBootTest

public class PolicyServiceImplTest {

@InjectMocks

PolicyServiceImpl policyService;

@Mock

PolicyRepository policyRepository;

**Note-**

1. The @Mock annotation creates a mock implementation for the class it is annotated with.
2. @InjectMocks also creates the mock implementation, additionally injects the dependent mocks that are marked with the annotations @Mock into it.
3. In the above example, we have annotated PolicyServiceImpl class with @InjectMocks, so mockito will create the mock object for PolicyServiceTestImpl class and inject the mock dependency of PolicyRepository into it.

**Step-4- Junit Implementation**

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.PolicyServiceTestImpl;

@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);

}

}

**Step-5- Right click on that class then Run as Junit Test**

**Controller Test**

**package** com.example.demo.controller;

**import** **static** org.junit.jupiter.api.Assertions.\*;

**import** **static** org.mockito.Mockito.*when*;

**import** **static** org.springframework.test.web.servlet.request.MockMvcRequestBuilders.*post*;

**import** **static** org.springframework.test.web.servlet.result.MockMvcResultHandlers.*print*;

**import** **static** org.springframework.test.web.servlet.result.MockMvcResultMatchers.*status*;

**import** org.junit.jupiter.api.Test;

**import** org.junit.jupiter.api.extension.ExtendWith;

**import** org.mockito.InjectMocks;

**import** org.mockito.Mock;

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

**import** org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;

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

**import** org.springframework.http.MediaType;

**import** org.springframework.test.context.junit.jupiter.SpringExtension;

**import** org.springframework.test.web.servlet.MockMvc;

**import** com.example.demo.model.Employee;

**import** com.example.demo.service.EmployeeService;

**import** com.fasterxml.jackson.databind.ObjectMapper;

**import** com.fasterxml.jackson.databind.SerializationFeature;

**import** com.fasterxml.jackson.datatype.jsr310.JavaTimeModule;

@ExtendWith(SpringExtension.**class**)

@AutoConfigureMockMvc

@SpringBootTest

**class** EmployeeControllerTest {

@Autowired

MockMvc mockMvc;

@InjectMocks

EmployeeController employeeController;

@Mock

EmployeeService employeeService;

@Test

**void** test() **throws** Exception {

ObjectMapper objectMapper = **new** ObjectMapper();

objectMapper.registerModule(**new** JavaTimeModule());

objectMapper.configure(SerializationFeature.***WRITE\_DATES\_AS\_TIMESTAMPS***, **false**);

Employee employee = **new** Employee();

employee.setId(1);

employee.setName("ram");

employee.setCity("pune");

String employeeObject = objectMapper.writeValueAsString(employee);

*when*(employeeService.saveEmployee(employee)).thenReturn(employee);

**this**.mockMvc.perform(*post*("/employee/save").contentType(MediaType.***APPLICATION\_JSON***).content(employeeObject))

.andDo(*print*()).andExpect(*status*().isOk());

}

}

**Service Test**

package com.example.demo.service;

import static org.junit.jupiter.api.Assertions.\*;

import static org.mockito.Mockito.when;

import java.util.ArrayList;

import java.util.List;

import org.junit.jupiter.api.Test;

import org.junit.jupiter.api.extension.ExtendWith;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;

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

import org.springframework.test.context.junit.jupiter.SpringExtension;

import com.example.demo.model.Employee;

import com.example.demo.repository.EmployeeRepository;

@ExtendWith(SpringExtension.class)

@AutoConfigureMockMvc

@SpringBootTest

class EmployeeServiceTest {

@InjectMocks

EmployeeServiceImpl employeeServiceImpl;

@Mock

EmployeeRepository employeeRepository;

@Test

void test() {

Employee employee = new Employee();

employee.setId(1);

employee.setName("ram");

employee.setCity("pune");

when(employeeRepository.save(employee)).thenReturn(employee);

assertEquals(employee, employeeServiceImpl.saveEmployee(employee));

}

@Test

void test\_zero() {

Employee employee = new Employee();

employee.setId(0);

employee.setName("ram");

employee.setCity("pune");

when(employeeRepository.save(employee)).thenReturn(employee);

assertEquals(employee, employeeServiceImpl.saveEmployee(employee));

}

@Test

void test\_employeeList() {

Employee employee1 = new Employee();

employee1.setId(1);

employee1.setName("ram");

employee1.setCity("pune");

Employee employee2 = new Employee();

employee2.setId(2);

employee2.setName("ashok");

employee2.setCity("mumbai");

List<Employee> empList = new ArrayList<>();

empList.add(employee1);

empList.add(employee2);

when(employeeRepository.findAll()).thenReturn(empList);

assertEquals(empList, employeeServiceImpl.getAllEmployee());

}

}