**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 AccountServiceImpl and AccountRepository. As the name implies, the manager class represents the service layer, and dao class interacts with the database.

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

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

@RunWith(MockitoJUnitRunner.**class**)

@SpringBootTest

**public** **class** AccountServiceImplTest {

@InjectMocks

AccountServiceImpl accountService;

@Mock

AccountRepository accountRepository;

**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 AccountServiceImpl class with @InjectMocks, so mockito will create the mock object for AccountServiceImpl class and inject the mock dependency of AccountRepository 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.mahindra.idbi.banking.ServiceImpl;
* **import** **static** org.junit.jupiter.api.Assertions.*assertEquals*;
* **import** **static** org.mockito.Mockito.*when*;
* **import** java.util.ArrayList;
* **import** java.util.List;
* **import** org.junit.Test;
* **import** org.junit.runner.RunWith;
* **import** org.mockito.InjectMocks;
* **import** org.mockito.Mock;
* **import** org.mockito.Mockito;
* **import** org.mockito.junit.MockitoJUnitRunner;
* **import** org.springframework.boot.test.context.SpringBootTest;
* **import** com.mahindra.idbi.banking.model.Account;
* **import** com.mahindra.idbi.banking.repository.AccountRepository;
* **import** com.mahindra.idbi.banking.serviceImpl.AccountServiceImpl;
* @RunWith(MockitoJUnitRunner.**class**)
* @SpringBootTest
* **public** **class** AccountServiceImplTest {
* @InjectMocks
* AccountServiceImpl accountService;
* @Mock
* AccountRepository accountRepository;
* @Test
* **public** **void** saveAccount() {
* Account account=**new** Account(101, "AC101", "5000", "TRN55541", "Active", "Rahul", "rahul@gmail.com");
* *when*(accountRepository.save(account)).thenReturn(account);
* *assertEquals*(account, accountService.saveAccount(account));
* }
* @Test
* **public** **void** getAllAccountTest() {
* List<Account> list=**new** ArrayList<>();
* list.add(**new** Account(101, "AC101", "5000", "TRN55541", "Active", "Rahul", "rahul@gmail.com"));
* list.add(**new** Account(102, "AC102", "6000", "TRN55555", "Active", "Sachin", "Sachin@gmail.com"));
* list.add(**new** Account(103, "AC103", "7000", "TRN55621", "Active", "Parth", "Parth@gmail.com"));
* *when*(accountRepository.findAll()).thenReturn(list);
* *assertEquals*(list, accountService.getAllAccount());
* }
* @Test
* **public** **void** getByIdTest() {
* *when*(accountRepository.findById(103)).thenReturn(**new** Account(103, "AC103", "7000", "TRN55621", "Active", "Parth", "Parth@gmail.com"));
* Account account=accountService.getAccountById(103);
* *assertEquals*(103, account.getId());
* *assertEquals*("AC103", account.getAccountNumber());
* *assertEquals*("7000", account.getAccountAmount());
* *assertEquals*("TRN55621", account.getTransactionNumber());
* *assertEquals*("Active", account.getStatus());
* *assertEquals*("Parth", account.getName());
* *assertEquals*("Parth@gmail.com", account.getEmail());
* }
* @Test
* **public** **void** deleteAccountTest() {
* Account account=**new** Account(101, "AC101", "5000", "TRN55541", "Active", "Rahul", "rahul@gmail.com");
* accountRepository.save(account);
* accountRepository.deleteById(account.getId());
* Account account1=accountRepository.findById(account.getId());
* *assertEquals*(**null**, account1);
* }
* }

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

