**MOCKITO MOCK DEPENDENCIES.**

**Exercise 1: Mocking a Service Dependency in a Controller Test**

Task: Write a unit test for a Spring controller that uses a service to fetch data. Mock the service dependency using Mockito.

**SOLUTION:**

**USERCONTROLLERTEST:**

import com.example.demo.controller.UserController;

import com.example.demo.model.User;

import com.example.demo.service.UserService;

import org.junit.jupiter.api.Test;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.springframework.http.ResponseEntity;

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

import static org.mockito.Mockito.\*;

import org.mockito.junit.jupiter.MockitoExtension;

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

@ExtendWith(MockitoExtension.class)

public class UserControllerTest {

@Mock

private UserService userService;

@InjectMocks

private UserController userController;

@Test

public void testGetUser() {

User mockUser = new User();

mockUser.setId(1L);

mockUser.setName("John Doe");

when(userService.getUserById(1L)).thenReturn(mockUser);

ResponseEntity<User> response = userController.getUser(1L);

assertEquals(200, response.getStatusCodeValue());

assertEquals("John Doe", response.getBody().getName());

}

}

**Exercise 2: Mocking a Repository in a Service Test**

Task: Write a unit test for a Spring service that uses a repository to fetch data. Mock the repository dependency using Mockito.

**SOLUTION:**

**USERSERVICETEST:**

import com.example.demo.model.User;

import com.example.demo.repository.UserRepository;

import com.example.demo.service.UserService;

import org.junit.jupiter.api.Test;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import java.util.Optional;

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

import static org.mockito.Mockito.\*;

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

import org.mockito.junit.jupiter.MockitoExtension;

@ExtendWith(MockitoExtension.class)

public class UserServiceTest {

@Mock

private UserRepository userRepository;

@InjectMocks

private UserService userService;

@Test

public void testGetUserById() {

User mockUser = new User();

mockUser.setId(2L);

mockUser.setName("Jane Doe");

when(userRepository.findById(2L)).thenReturn(Optional.of(mockUser));

User result = userService.getUserById(2L);

assertNotNull(result);

assertEquals("Jane Doe", result.getName());

}

}

**Exercise 3: Mocking a Service Dependency in an Integration Test**

Task:Write an integration test for a Spring Boot application that mocks a service dependency using Mockito.

**SOLUTION:**

**USERINTEGRATIONTEST:**

import com.example.demo.model.User;

import com.example.demo.service.UserService;

import org.junit.jupiter.api.Test;

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

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

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

import static org.mockito.Mockito.when;

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

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

import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;

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

@SpringBootTest

@AutoConfigureMockMvc

public class UserIntegrationTest {

@Autowired

private MockMvc mockMvc;

@MockBean

private UserService userService;

@Test

public void testGetUserEndpoint() throws Exception {

User mockUser = new User();

mockUser.setId(3L);

mockUser.setName("Test User");

when(userService.getUserById(3L)).thenReturn(mockUser);

mockMvc.perform(get("/users/3"))

.andExpect(status().isOk())

.andExpect(jsonPath("$.name").value("Test User"));

}

}