**5. Mockito Mock dependencies exercises**

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

**UserControllerTest:**

package com.example.controller;

import com.example.model.User;

import com.example.service.UserService;

import org.junit.jupiter.api.Test;

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

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

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

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

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

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

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

@WebMvcTest(UserController.class)

public class UserControllerTest {

@Autowired

private MockMvc mockMvc;

@MockBean

private UserService userService;

@Test

public void testGetUserById() throws Exception {

User mockUser = new User();

mockUser.setId(1L);

mockUser.setName("Alice");

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

mockMvc.perform(*get*("/users/1"))

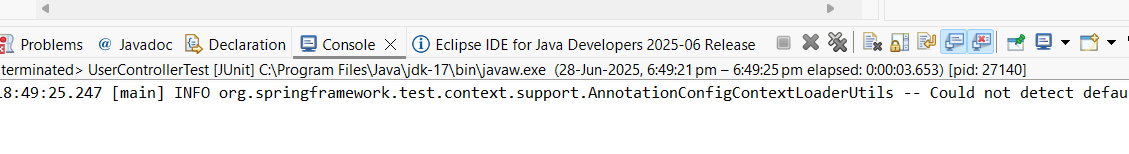
.andExpect(*status*().isOk())

.andExpect(*jsonPath*("$.name").value("Alice"));

}

}

**OUTPUT**

****

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

UserServiceTest

package com.example.service;

import com.example.model.User;

import com.example.repository.UserRepository;

import org.junit.jupiter.api.Test;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.MockitoAnnotations;

import org.junit.jupiter.api.BeforeEach;

import java.util.Optional;

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

import static org.mockito.Mockito.\*;

public class UserServiceTest {

@Mock

private UserRepository userRepository;

@InjectMocks

private UserService userService;

@BeforeEach

public void setup() {

MockitoAnnotations.openMocks(this);

}

@Test

public void testGetUserById\_found() {

User user = new User();

user.setId(1L);

user.setName("Bob");

when(userRepository.findById(1L)).thenReturn(Optional.of(user));

User result = userService.getUserById(1L);

assertNotNull(result);

assertEquals("Bob", result.getName());

}

@Test

public void testGetUserById\_notFound() {

when(userRepository.findById(2L)).thenReturn(Optional.empty());

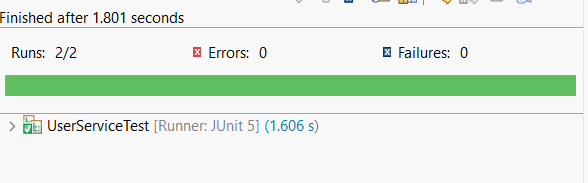
User result = userService.getUserById(2L);

assertNull(result);

}

}

**OUTPUT**



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

**CODE**

**UserIntegrationTest:**

package com.example.integration;

import com.example.model.User;

import com.example.service.UserService;

import com.example.app.SpringTestingApplication;

import org.junit.jupiter.api.Test;

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

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

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

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

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

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

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

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

@SpringBootTest(classes = SpringTestingApplication.class)

@AutoConfigureMockMvc

public class UserIntegrationTest {

@Autowired

private MockMvc mockMvc;

@MockBean

private UserService userService;

@Test

public void testGetUserMockedService() throws Exception {

User user = new User();

user.setId(1L);

user.setName("Charlie");

*when*(userService.getUserById(1L)).thenReturn(user);

mockMvc.perform(*get*("/users/1"))

.andExpect(*status*().isOk())

.andExpect(*jsonPath*("$.name").value("Charlie"));

}

}

**OUTPUT**

