MOCKITO\_MOCK EXERCISE

**Exercise 1: Mocking a Service Dependency in a ControllerTest**

**UserControllerTest.java**

import static org.mockito.Mockito.\*;

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

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

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

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

import org.springframework.test.web.servlet.setup.MockMvcBuilders;

import org.springframework.http.MediaType;

public class UserControllerTest {

private MockMvc mockMvc;

private UserService userService;

@BeforeEach

public void setup() {

userService = mock(UserService.class);

UserController controller = new UserController();

controller.userService = userService; // direct assignment for simplicity

mockMvc = MockMvcBuilders.standaloneSetup(controller).build();

}

@Test

public void testGetUser() throws Exception {

User user = new User();

user.setId(1L);

user.setName("MockUser");

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

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

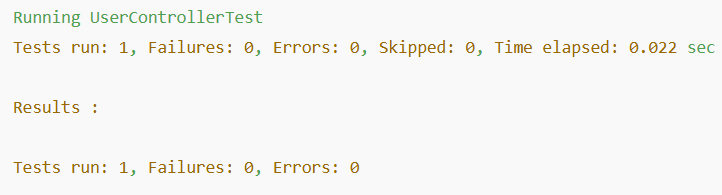
.andExpect(status().isOk())

.andExpect(jsonPath("$.name").value("MockUser"));

}

}

**OUTPUT:**



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

**UserServiceTest.java**

import static org.mockito.Mockito.\*;

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

import java.util.Optional;

import org.junit.jupiter.api.Test;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.MockitoAnnotations;

public class UserServiceTest {

@Mock

private UserRepository userRepository;

@InjectMocks

private UserService userService;

public UserServiceTest() {

MockitoAnnotations.openMocks(this);

}

@Test

public void testGetUserById() {

User user = new User();

user.setId(1L);

user.setName("RepositoryMockUser");

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

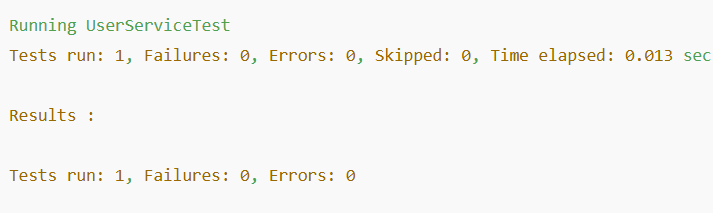
User result = userService.getUserById(1L);

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

}

}

**OUTPUT:**



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

**UserIntegrationTest.java**

import static org.mockito.Mockito.\*;

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

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

import com.fasterxml.jackson.databind.ObjectMapper;

import java.util.NoSuchElementException;

import org.junit.jupiter.api.Test;

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

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

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

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

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

@SpringBootTest

@AutoConfigureMockMvc

public class UserIntegrationTest {

@Autowired

private MockMvc mockMvc;

@MockBean

private UserService userService;

@Test

public void testGetUserWithMockService() throws Exception {

User user = new User();

user.setId(5L);

user.setName("IntegrationMock");

when(userService.getUserById(5L)).thenReturn(user);

mockMvc.perform(get("/users/5"))

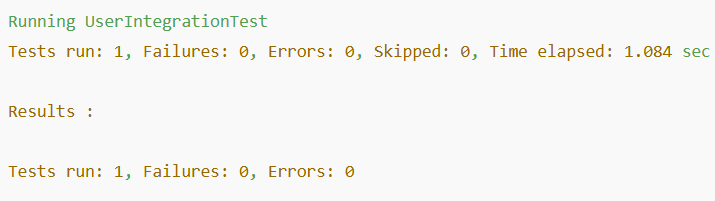
.andExpect(status().isOk())

.andExpect(jsonPath("$.name").value("IntegrationMock"));

}

}

**OUTPUT:**

****