Mocking Dependencies in Spring Tests using Mockito

Exercise 1: UserControllerTest

import org.junit.jupiter.api.Test;

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

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.junit.jupiter.MockitoExtension;

import org.springframework.http.ResponseEntity;

import static org.mockito.Mockito.\*;

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

@ExtendWith(MockitoExtension.class)

public class UserControllerTest {

@Mock

private UserService userService;

@InjectMocks

private UserController userController;

@Test

public void testGetUserById\_Success() {

*// Arrange*

Long userId = 1L;

User mockUser = new User();

mockUser.setId(userId);

mockUser.setName("John Doe");

when(userService.getUserById(userId)).thenReturn(mockUser);

*// Act*

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

*// Assert*

assertEquals(200, response.getStatusCodeValue());

assertNotNull(response.getBody());

assertEquals(userId, response.getBody().getId());

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

verify(userService, times(1)).getUserById(userId);

}

@Test

public void testGetUserById\_NotFound() {

*// Arrange*

Long userId = 99L;

when(userService.getUserById(userId)).thenReturn(null);

*// Act*

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

*// Assert*

assertEquals(200, response.getStatusCodeValue()); *// Note: Controller returns 200 even for null*

assertNull(response.getBody());

verify(userService, times(1)).getUserById(userId);

}

}

Exercise 2: UserServiceTest

import org.junit.jupiter.api.Test;

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

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.junit.jupiter.MockitoExtension;

import java.util.Optional;

import static org.mockito.Mockito.\*;

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

@ExtendWith(MockitoExtension.class)

public class UserServiceTest {

@Mock

private UserRepository userRepository;

@InjectMocks

private UserService userService;

@Test

public void testGetUserById\_Success() {

*// Arrange*

Long userId = 1L;

User mockUser = new User();

mockUser.setId(userId);

mockUser.setName("Jane Smith");

when(userRepository.findById(userId)).thenReturn(Optional.of(mockUser));

*// Act*

User result = userService.getUserById(userId);

*// Assert*

assertNotNull(result);

assertEquals(userId, result.getId());

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

verify(userRepository, times(1)).findById(userId);

}

@Test

public void testGetUserById\_NotFound() {

*// Arrange*

Long userId = 99L;

when(userRepository.findById(userId)).thenReturn(Optional.empty());

*// Act*

User result = userService.getUserById(userId);

*// Assert*

assertNull(result);

verify(userRepository, times(1)).findById(userId);

}

}

Exercise 3: UserIntegrationTest

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

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

import static org.mockito.Mockito.\*;

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

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\_Success() throws Exception {

*// Arrange*

Long userId = 1L;

User mockUser = new User();

mockUser.setId(userId);

mockUser.setName("Test User");

when(userService.getUserById(userId)).thenReturn(mockUser);

*// Act & Assert*

mockMvc.perform(get("/users/{id}", userId))

.andExpect(status().isOk())

.andExpect(jsonPath("$.id").value(userId))

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

verify(userService, times(1)).getUserById(userId);

}

@Test

public void testGetUserEndpoint\_NotFound() throws Exception {

*// Arrange*

Long userId = 99L;

when(userService.getUserById(userId)).thenReturn(null);

*// Act & Assert*

mockMvc.perform(get("/users/{id}", userId))

.andExpect(status().isOk())

.andExpect(content().string(""));

verify(userService, times(1)).getUserById(userId);

}

}