SPRING TESTING EXERCISE

Exercise 1: Basic Unit Test for a Service Method

**CalculatorServiceTest.java**

import org.junit.jupiter.api.Test;

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

public class CalculatorServiceTest {

@Test

public void testAdd() {

CalculatorService service = new CalculatorService();

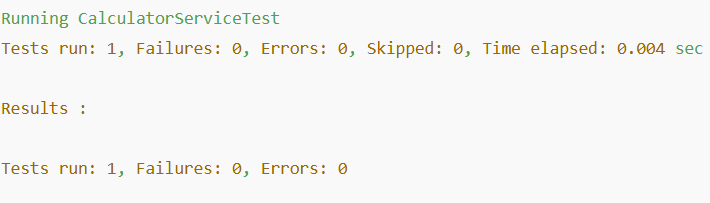
int result = service.add(10, 5);

assertEquals(15, result);

}

}

**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("John");

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

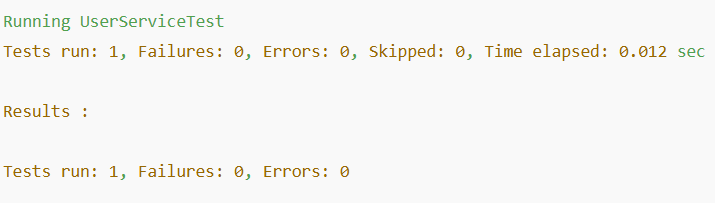
User result = userService.getUserById(1L);

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

}

}

**OUTPUT:**

**** **Exercise 3: Testing a REST Controller with MockMvc**

**UserControllerTest.java**

import static org.mockito.Mockito.\*;

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

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

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 = mock(UserService.class);

@BeforeEach

void setup() {

UserController controller = new UserController();

controller.userService = userService;

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

}

@Test

public void testGetUser() throws Exception {

User user = new User();

user.setId(1L);

user.setName("Alice");

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

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

.andExpect(status().isOk())

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

}

} import static org.mockito.Mockito.\*;

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

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

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 = mock(UserService.class);

@BeforeEach

void setup() {

UserController controller = new UserController();

controller.userService = userService;

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

}

@Test

public void testGetUser() throws Exception {

User user = new User();

user.setId(1L);

user.setName("Alice");

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

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

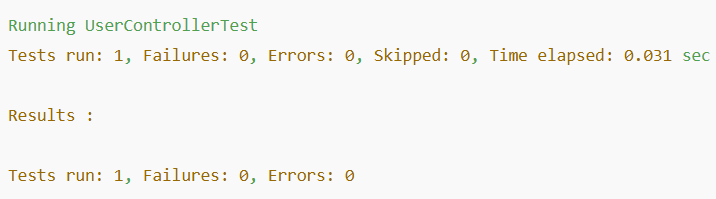
.andExpect(status().isOk())

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

}

}

**OUTPUT:**

****

**Exercise 4: Integration Test with Spring Boot**

**UserIntegrationTest.java**

import org.junit.jupiter.api.Test;

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

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

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

@SpringBootTest

public class UserIntegrationTest {

@Autowired

private UserService userService;

@Test

public void testFullFlow() {

User user = new User();

user.setId(1L);

user.setName("Bob");

userService.saveUser(user);

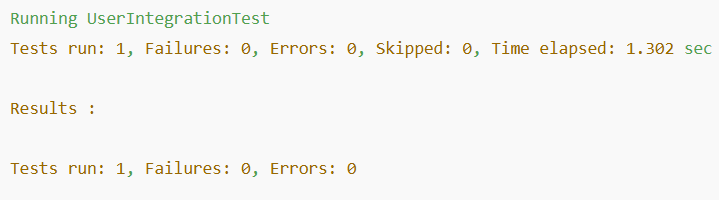
User fetched = userService.getUserById(1L);

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

}

}

**OUTPUT:**

**** **Exercise 5: Test Controller POST Endpoint**

**UserPostControllerTest.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 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;

public class UserPostControllerTest {

private MockMvc mockMvc;

private UserService userService = mock(UserService.class);

private ObjectMapper mapper = new ObjectMapper();

@BeforeEach

public void setup() {

UserController controller = new UserController();

controller.userService = userService;

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

}

@Test

public void testCreateUser() throws Exception {

User user = new User();

user.setId(2L);

user.setName("Jane");

when(userService.saveUser(any(User.class))).thenReturn(user);

mockMvc.perform(post("/users")

.contentType("application/json")

.content(mapper.writeValueAsString(user)))

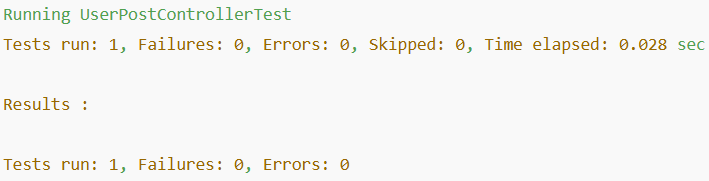
.andExpect(status().isOk())

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

}

}

**OUTPUT:**

****

**Exercise 6: Test Service Exception Handling**

**UserServiceExceptionTest.java**

import static org.mockito.Mockito.\*;

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

import java.util.Optional;

import java.util.NoSuchElementException;

import org.junit.jupiter.api.Test;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.MockitoAnnotations;

public class UserServiceExceptionTest {

@Mock

private UserRepository userRepository;

@InjectMocks

private UserService userService;

public UserServiceExceptionTest() {

MockitoAnnotations.openMocks(this);

}

@Test

public void testMissingUserThrowsException() {

when(userRepository.findById(99L)).thenReturn(Optional.empty());

assertThrows(NoSuchElementException.class, () -> {

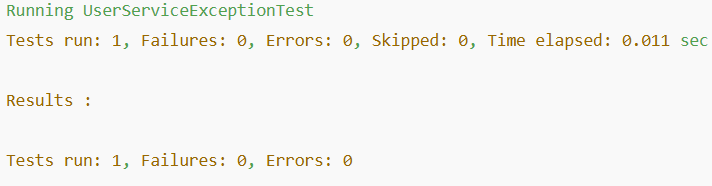
userService.getUserById(99L).getName();

});

}

}

**OUTPUT:**

****

**Exercise 7: Test Custom Repository Query**

**UserRepositoryTest.java**

import org.junit.jupiter.api.Test;

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

import org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;

import java.util.List;

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

@DataJpaTest

public class UserRepositoryTest {

@Autowired

private UserRepository userRepository;

@Test

public void testFindByName() {

User user = new User();

user.setId(3L);

user.setName("David");

userRepository.save(user);

List<User> users = userRepository.findByName("David");

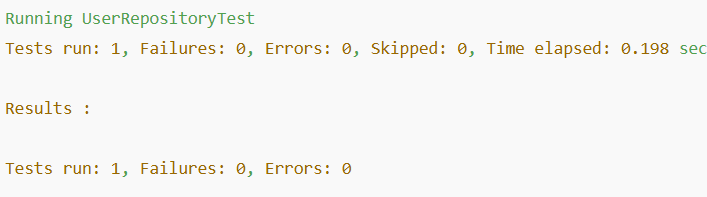
assertEquals(1, users.size());

assertEquals("David", users.get(0).getName());

}

}

**OUTPUT:**



**Exercise 8: Test Controller Exception Handling**

**GlobalExceptionHandlerTest.java**

import static org.mockito.Mockito.\*;

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

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

import java.util.NoSuchElementException;

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;

public class GlobalExceptionHandlerTest {

private MockMvc mockMvc;

private UserService userService = mock(UserService.class);

@BeforeEach

void setup() {

UserController controller = new UserController();

controller.userService = userService;

mockMvc = MockMvcBuilders

.standaloneSetup(controller)

.setControllerAdvice(new GlobalExceptionHandler())

.build();

}

@Test

void testNoSuchElementHandled() throws Exception {

when(userService.getUserById(99L)).thenThrow(new NoSuchElementException());

mockMvc.perform(get("/users/99"))

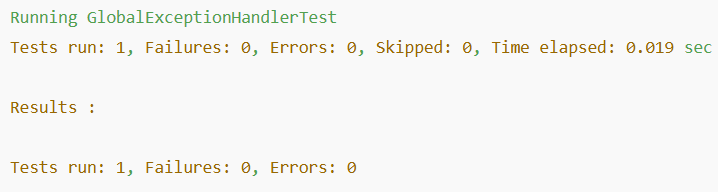
.andExpect(status().isNotFound())

.andExpect(content().string("User not found"));

}

}

**OUTPUT:**



**Exercise 9: Parameterized Test with Junit**

**ParameterizedTestExample.java**

import org.junit.jupiter.params.ParameterizedTest;

import org.junit.jupiter.params.provider.ValueSource;

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

public class ParameterizedTestExample {

@ParameterizedTest

@ValueSource(strings = {"Alice", "Bob", "Charlie"})

public void testNameNotEmpty(String name) {

assertNotNull(name);

assertFalse(name.isEmpty());

}

}

**OUTPUT:**

