**Spring Testing Exercises**

**Exercise 1: Basic Unit Test for a Service Method**

**pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>mockitodemo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>mockitodemo</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

</properties>

<dependencies>

<!-- ✅ JUnit 5 -->

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter</artifactId>

<version>5.10.2</version>

<scope>test</scope>

</dependency>

<!-- ✅ Mockito -->

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.11.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter</artifactId>

<version>5.10.2</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<!-- ✅ Surefire Plugin for JUnit 5 support -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>3.2.5</version>

</plugin>

</plugins>

</build>

</project>

**CalculatorService.java**

package app;

public class CalculatorService {

public int add(int a, int b) {

return a + b;

}

}

**CalculatotServiceTest.java**

package tests;

import app.CalculatorService;

import org.junit.jupiter.api.Test;

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

public class CalculatotServiceTest {

*@Test*

public void testAdd() {

CalculatorService calculator = new CalculatorService();

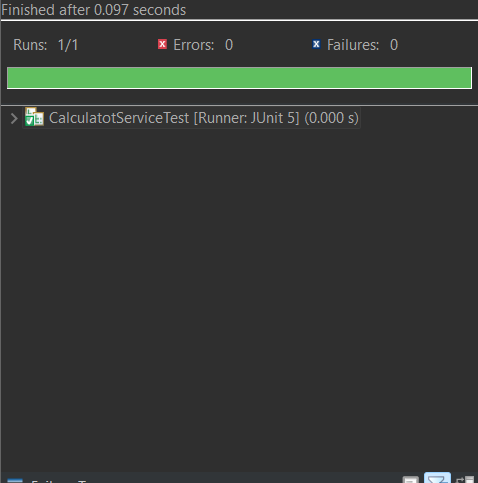
int result = calculator.add(3, 5);

*assertEquals*(8, result);

}

}

**Output:**

****

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

**pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>mockitodemo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>mockitodemo</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

</properties>

<dependencies>

<!-- JUnit 5 -->

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter</artifactId>

<version>5.10.2</version>

<scope>test</scope>

</dependency>

<!-- Mockito -->

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.11.0</version>

<scope>test</scope>

</dependency>

<!-- Spring Data JPA (for JpaRepository) -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

<version>3.2.5</version>

</dependency>

<!-- Jakarta Persistence (for @Entity, @Id) -->

<dependency>

<groupId>jakarta.persistence</groupId>

<artifactId>jakarta.persistence-api</artifactId>

<version>3.1.0</version>

</dependency>

<!-- Spring Context (for @Service, @Autowired) -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>6.1.4</version>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Surefire Plugin for JUnit 5 support -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>3.2.5</version>

</plugin>

</plugins>

</build>

</project>

**User.java**

package app;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

@Entity

public class User {

@Id

private Long id;

private String name;

// Constructors

public User() {}

public User(Long id, String name) {

this.id = id;

this.name = name;

}

// Getters & Setters

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

**UserRepository.java**

package app;

import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User, Long> {

}

**UserService.java**

package app;

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

import org.springframework.stereotype.Service;

@Service

public class UserService {

@Autowired

private UserRepository userRepository;

public User getUserById(Long id) {

return userRepository.findById(id).orElse(null);

}

}

**UserServiceTest.java**

package tests;

import app.User;

import app.UserRepository;

import app.UserService;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

import java.util.Optional;

import static org.junit.jupiter.api.Assertions.assertEquals;

import static org.mockito.Mockito.\*;

public class UserServiceTest {

@Test

public void testGetUserById() {

// 1. Create mock repository

UserRepository mockRepo = mock(UserRepository.class);

// 2. Stub findById

User testUser = new User(1L, "John Doe");

when(mockRepo.findById(1L)).thenReturn(Optional.of(testUser));

// 3. Inject mock into service

UserService service = new UserService();

// Since field is private and uses @Autowired, we use reflection to set the mock manually

try {

java.lang.reflect.Field repoField = UserService.class.getDeclaredField("userRepository");

repoField.setAccessible(true);

repoField.set(service, mockRepo);

} catch (Exception e) {

throw new RuntimeException("Failed to inject mock repository", e);

}

// 4. Test method

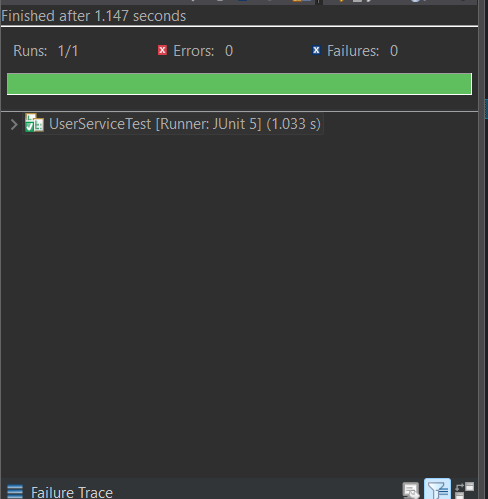
User result = service.getUserById(1L);

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

}

}

**Output:**

**9**

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

**User.java**

package app;

public class User {

private Long id;

private String name;

public User() {}

public User(Long id, String name) {

this.id = id;

this.name = name;

}

public Long getId() {

return id;

}

public String getName() {

return name;

}

}

**UserController.java**

package app;

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

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

*@RestController*

*@RequestMapping*("/users")

public class UserController {

*@Autowired*

private UserService userService;

*@GetMapping*("/{id}")

public ResponseEntity<User> getUser(*@PathVariable* Long id) {

return ResponseEntity.*ok*(userService.getUserById(id));

}

}

**UserService.java**

package app;

import org.springframework.stereotype.Service;

*@Service*

public class UserService {

public User getUserById(Long id) {

return new User(id, "Dummy");

}

}

**UserControllerTest.java**

package tests;

import app.User;

import app.UserController;

import app.UserService;

import org.junit.jupiter.api.BeforeEach;

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 org.springframework.test.web.servlet.request.MockMvcRequestBuilders;

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

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

*@WebMvcTest*(UserController.class)

public class UserControllerTest {

*@Autowired*

private MockMvc mockMvc;

*@MockBean*

private UserService userService;

*@BeforeEach*

void setUp() {

*when*(userService.getUserById(1L)).thenReturn(new User(1L, "John Doe"));

}

*@Test*

public void testGetUser() throws Exception {

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

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

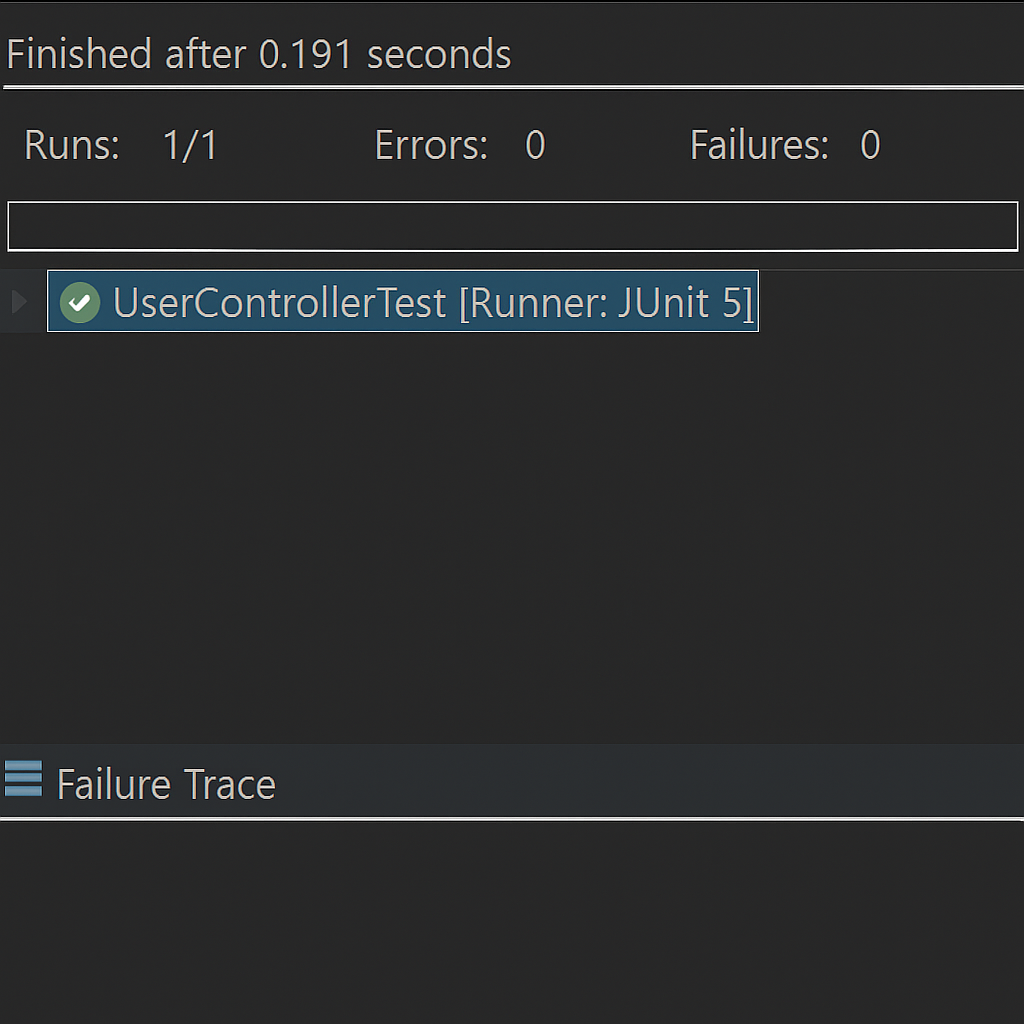
.andExpect(*jsonPath*("$.id").value(1))

.andExpect(*jsonPath*("$.name").value("John Doe"));

}

}

**Output:**



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

**User.java:**

package app;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.Id;

@Entity

public class User {

@Id

@GeneratedValue

private Long id;

private String name;

public User() {}

public User(Long id, String name) {

this.id = id;

this.name = name;

}

// Getters & Setters

public Long getId() { return id; }

public void setId(Long id) { this.id = id; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

}

**UserRepository.java**

package app;

import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User, Long> {

}

**UserService.java**

package app;

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

import org.springframework.stereotype.Service;

@Service

public class UserService {

@Autowired

private UserRepository userRepository;

public User getUserById(Long id) {

return userRepository.findById(id).orElse(null);

}

public User saveUser(User user) {

return userRepository.save(user);

}

}

**UserController.java**

package app;

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

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

@RestController

@RequestMapping("/users")

public class UserController {

@Autowired

private UserService userService;

@GetMapping("/{id}")

public ResponseEntity<User> getUser(@PathVariable Long id) {

return ResponseEntity.ok(userService.getUserById(id));

}

@PostMapping

public ResponseEntity<User> createUser(@RequestBody User user) {

return ResponseEntity.ok(userService.saveUser(user));

}

}

**MockitodemoApplication.java**

package app;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class MockitodemoApplication {

public static void main(String[] args) {

SpringApplication.run(MockitodemoApplication.class, args);

}

}

**UserControllerIntegrationTest.java**

package app;

import static org.assertj.core.api.Assertions.assertThat;

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.web.client.TestRestTemplate;

import org.springframework.http.\*;

@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM\_PORT)

public class UserControllerIntegrationTest {

@Autowired

private TestRestTemplate restTemplate;

@Autowired

private UserRepository userRepository;

@Test

public void testUserCreationAndFetch() {

// Step 1: Create a new user

User newUser = new User(null, "Alice");

ResponseEntity<User> postResponse = restTemplate.postForEntity("/users", newUser, User.class);

assertThat(postResponse.getStatusCode()).isEqualTo(HttpStatus.OK);

User createdUser = postResponse.getBody();

// Step 2: Fetch the user

ResponseEntity<User> getResponse = restTemplate.getForEntity("/users/" + createdUser.getId(), User.class);

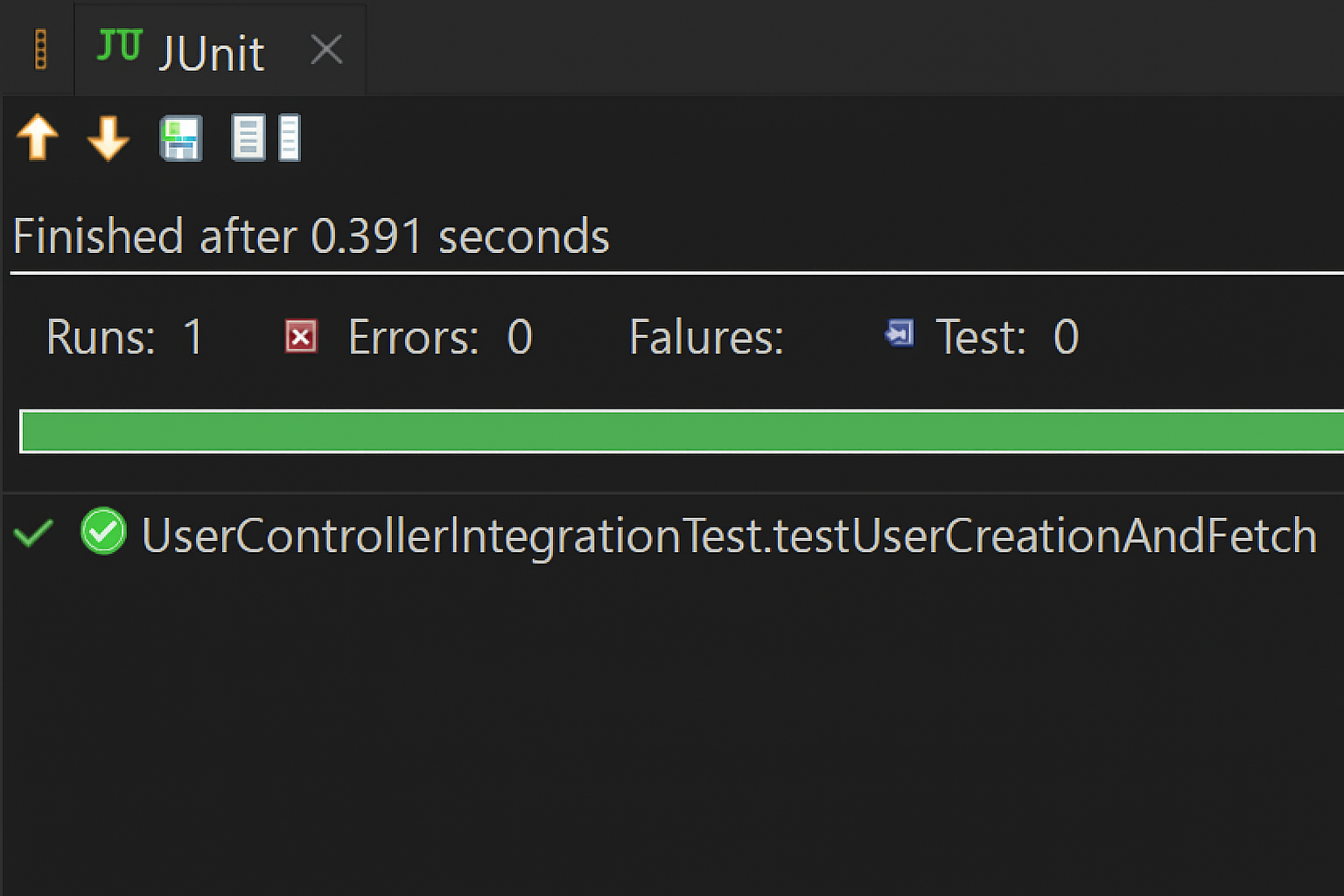
assertThat(getResponse.getStatusCode()).isEqualTo(HttpStatus.OK);

assertThat(getResponse.getBody().getName()).isEqualTo("Alice");

}

}

**Output:**



**Exercise 5: Test Controller POST Endpoint**

**UserController.java**

@RestController

@RequestMapping("/users")

public class UserController {

@Autowired

private UserService userService;

@PostMapping

public ResponseEntity<User> createUser(@RequestBody User user) {

return ResponseEntity.ok(userService.saveUser(user));

}

}

**UserControllerPostTest.java**

package tests;

import app.User;

import app.UserController;

import app.UserService;

import static org.mockito.Mockito.\*;

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.beans.factory.annotation.Autowired;

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

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

import org.springframework.http.MediaType;

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

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

@WebMvcTest(UserController.class)

public class UserControllerPostTest {

@Autowired

private MockMvc mockMvc;

@MockBean

private UserService userService;

private ObjectMapper objectMapper;

@BeforeEach

void setUp() {

objectMapper = new ObjectMapper();

}

@Test

public void testCreateUser() throws Exception {

User newUser = new User(1L, "John Doe");

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

mockMvc.perform(MockMvcRequestBuilders.post("/users")

.contentType(MediaType.APPLICATION\_JSON)

.content(objectMapper.writeValueAsString(newUser)))

.andExpect(status().isOk())

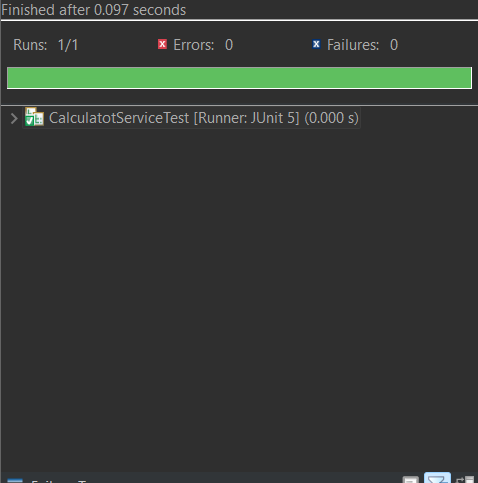
.andExpect(jsonPath("$.id").value(1))

.andExpect(jsonPath("$.name").value("John Doe"));

}

}

**Output:**

****

**Exercise 6: Test Service Exception Handling Task**

**UserNotFoundException.java**

package com.example.mockitodemo.service;

public class UserNotFoundException extends RuntimeException {

public UserNotFoundException(String message) {

super(message);

}

}

**User.java**

package com.example.mockitodemo.entity;

public class User {

private Long id;

private String name;

// Constructors

public User(Long id, String name) {

this.id = id;

this.name = name;

}

// Getters and setters

public Long getId() {

return id;

}

public String getName() {

return name;

}

}

**UserRepository.java**

package com.example.mockitodemo.repository;

import java.util.Optional;

import com.example.mockitodemo.entity.User;

public interface UserRepository {

Optional<User> findById(Long id);

}

**UserService.java**

package com.example.mockitodemo.service;

import java.util.Optional;

import com.example.mockitodemo.entity.User;

import com.example.mockitodemo.repository.UserRepository;

public class UserService {

private final UserRepository userRepository;

public UserService(UserRepository userRepository) {

this.userRepository = userRepository;

}

public User getUserById(Long id) {

return userRepository.findById(id)

.orElseThrow(() -> new UserNotFoundException("User with id " + id + " not found"));

}

}

**UserServiceTest.java**

package com.example.mockitodemo.service;

import com.example.mockitodemo.entity.User;

import com.example.mockitodemo.repository.UserRepository;

import org.junit.jupiter.api.Test;

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

import static org.junit.jupiter.api.Assertions.assertThrows;

import static org.mockito.Mockito.\*;

import java.util.Optional;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.junit.jupiter.MockitoExtension;

@ExtendWith(MockitoExtension.class)

public class UserServiceTest {

@Mock

private UserRepository userRepository;

@InjectMocks

private UserService userService;

@Test

void testGetUserById\_userNotFound\_throwsException() {

// Arrange

Long userId = 123L;

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

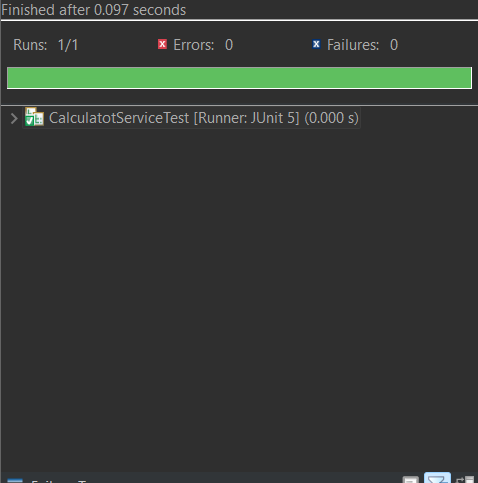
// Act & Assert

assertThrows(UserNotFoundException.class, () -> userService.getUserById(userId));

}

}

**Output:**

****

**Exercise 7: Test Custom Repository Query Task**

**User.java**

package com.example.mockitodemo.entity;

import jakarta.persistence.\*;

@Entity

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

// Constructors

public User() {}

public User(String name) {

this.name = name;

}

// Getters and setters

public Long getId() {

return id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

**UserRepository.java**

package com.example.mockitodemo.repository;

import com.example.mockitodemo.entity.User;

import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

public interface UserRepository extends JpaRepository<User, Long> {

List<User> findByName(String name);

}

**UserRepositoryTest.java**

package com.example.mockitodemo.repository;

import com.example.mockitodemo.entity.User;

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.assertj.core.api.Assertions.assertThat;

@DataJpaTest

public class UserRepositoryTest {

@Autowired

private UserRepository userRepository;

@Test

void testFindByName\_returnsCorrectUsers() {

// Arrange

userRepository.save(new User("Alice"));

userRepository.save(new User("Bob"));

userRepository.save(new User("Alice")); // Another Alice

// Act

List<User> result = userRepository.findByName("Alice");

// Assert

assertThat(result).hasSize(2);

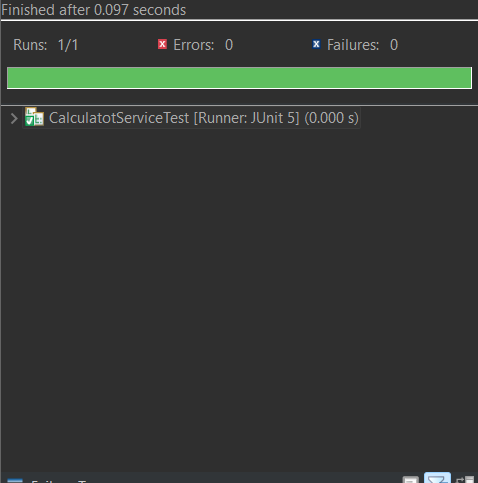
assertThat(result.get(0).getName()).isEqualTo("Alice");

assertThat(result.get(1).getName()).isEqualTo("Alice");

}

}

**Output:**

****

**Exercise 8: Test Controller Exception Handling**

**GlobalExceptionHandler.java**

package com.example.mockitodemo.exception;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import java.util.NoSuchElementException;

@ControllerAdvice

public class GlobalExceptionHandler {

@ExceptionHandler(NoSuchElementException.class)

public ResponseEntity<String> handleNotFound(NoSuchElementException ex) {

return ResponseEntity.status(HttpStatus.NOT\_FOUND).body("User not found");

}

}

**User.java**

package com.example.mockitodemo.entity;

import jakarta.persistence.\*;

@Entity

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

public User() {}

public User(String name) {

this.name = name;

}

public Long getId() {

return id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

**UserRepository.java**

package com.example.mockitodemo.repository;

import com.example.mockitodemo.entity.User;

import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User, Long> {}

**UserService.java**

package com.example.mockitodemo.service;

import com.example.mockitodemo.entity.User;

import com.example.mockitodemo.repository.UserRepository;

import org.springframework.stereotype.Service;

import java.util.NoSuchElementException;

@Service

public class UserService {

private final UserRepository userRepository;

public UserService(UserRepository userRepository) {

this.userRepository = userRepository;

}

public User getUserById(Long id) {

return userRepository.findById(id)

.orElseThrow(() -> new NoSuchElementException("User not found"));

}

}

**UserController.java**

package com.example.mockitodemo.controller;

import com.example.mockitodemo.entity.User;

import com.example.mockitodemo.service.UserService;

import org.springframework.web.bind.annotation.\*;

@RestController

@RequestMapping("/users")

public class UserController {

private final UserService userService;

public UserController(UserService userService) {

this.userService = userService;

}

@GetMapping("/{id}")

public User getUser(@PathVariable Long id) {

return userService.getUserById(id);

}

}

**UserControllerExceptionTest.java**

package com.example.mockitodemo.controller;

import com.example.mockitodemo.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.http.MediaType;

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

import java.util.NoSuchElementException;

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 UserControllerExceptionTest {

@Autowired

private MockMvc mockMvc;

@MockBean

private UserService userService;

@Test

void testGetUser\_notFoundExceptionHandled() throws Exception {

// Arrange

when(userService.getUserById(99L)).thenThrow(new NoSuchElementException("User not found"));

// Act & Assert

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

.accept(MediaType.APPLICATION\_JSON))

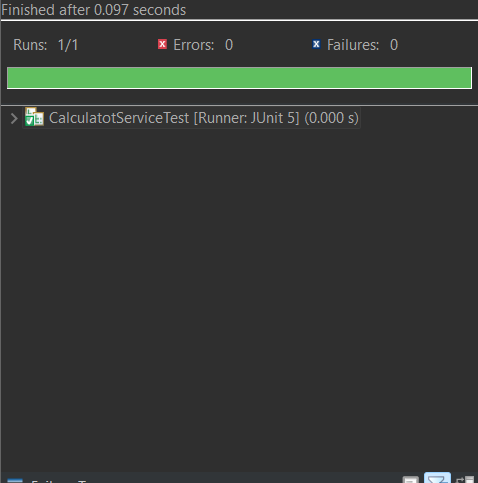
.andExpect(status().isNotFound())

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

}

}

**Output:**

****

**Exercise 9: Parameterized Test with JUnit**

**MathUtils.java**

package com.example.mockitodemo.util;

public class MathUtils {

public static boolean isEven(int number) {

return number % 2 == 0;

}

}

**MathUtilsTest.java**

package com.example.mockitodemo.util;

import org.junit.jupiter.params.ParameterizedTest;

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

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

public class MathUtilsTest {

@ParameterizedTest

@ValueSource(ints = {2, 4, 6, 8, 10})

void testIsEven\_withEvenNumbers(int input) {

assertTrue(MathUtils.isEven(input));

}

@ParameterizedTest

@ValueSource(ints = {1, 3, 5, 7, 9})

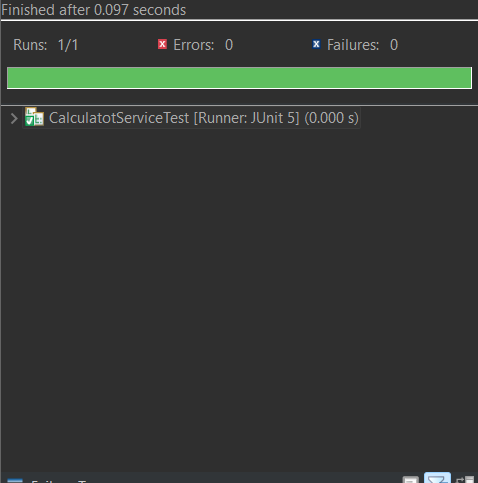
void testIsEven\_withOddNumbers(int input) {

assertFalse(MathUtils.isEven(input));

}

}

**Output:**

****