Junit\_Spring Test Exercises

Exercise 1: Basic Unit Test for a Service Method

CalculatorService.java:

package com.example.demo.service;

import org.springframework.stereotype.Service;

@Service

public class CalculatorService { public int add(int a, int b) { return a + b;

}

}

# Exercise 2: Mocking a Repository in a Service Test

UserServiceTest.java:

package com.example.demo.service;

import com.example.demo.model.User;

import com.example.demo.repository.UserRepository; import org.junit.jupiter.api.Test;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.MockitoAnnotations;

import java.util.Optional;

import static org.mockito.Mockito.\*;

import static org.junit.jupiter.api.Assertions.\*; class UserServiceTest {

@Mock

private UserRepository userRepository; @InjectMocks

private UserService userService;

public UserServiceTest() {

MockitoAnnotations.openMocks(this); // Initialize mocks

}

@Test

void testGetUserById\_Found() { Long userId = 1L;

User mockUser = new User(userId, "Alice"); when(userRepository.findById(userId)).thenReturn(Optional.of(mockUser)); User result = userService.getUserById(userId);

assertNotNull(result);

assertEquals("Alice", result.getName()); assertEquals(1L, result.getId());

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

}

@Test

void testGetUserById\_NotFound() { Long userId = 2L;

when(userRepository.findById(userId)).thenReturn(Optional.empty()); User result = userService.getUserById(userId);

assertNull(result);

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

}

}

# Exercise 3: Testing a REST Controller with MockMvc

UserControllerTest.java:

package com.example.demo.controller; import com.example.demo.model.User;

import com.example.demo.service.UserService;

import org.junit.jupiter.api.Test; import static org.mockito.Mockito.\*;

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

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 com.fasterxml.jackson.databind.ObjectMapper;

@WebMvcTest(UserController.class)

class UserControllerTest {

@Autowired

private MockMvc mockMvc; @MockBean

private UserService userService; @Autowired

private ObjectMapper objectMapper;

@Test

void testGetUser\_ReturnsUser() throws Exception { Long userId = 1L;

User mockUser = new User(userId, "Alice"); when(userService.getUserById(userId)).thenReturn(mockUser); mockMvc.perform(MockMvcRequestBuilders.get("/users/{id}", userId))

.andExpect(status().isOk())

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

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

}

}

# Exercise 4: Integration Test with Spring Boot

Entity: User.java

package com.example.demo.model;

import jakarta.persistence.Entity; import jakarta.persistence.Id;

@Entity

public class User { @Id

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

public String getName() { return name; }

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

}

Repository: UserRepository.java

package com.example.demo.repository;

import com.example.demo.model.User;

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

public interface UserRepository extends JpaRepository<User, Long> {

}

Service: UserService.java

package com.example.demo.service;

import com.example.demo.model.User;

import com.example.demo.repository.UserRepository;

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);

}

}

Controller: UserController.java

package com.example.demo.controller;

import com.example.demo.model.User;

import com.example.demo.service.UserService;

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) { User user = userService.getUserById(id);

return user != null ? ResponseEntity.ok(user) : ResponseEntity.notFound().build();

}

}

Integration Test: UserIntegrationTest.java

package com.example.demo;

import com.example.demo.model.User;

import com.example.demo.repository.UserRepository; import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

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

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.test.web.servlet.MockMvc;

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

@SpringBootTest

@AutoConfigureMockMvc class UserIntegrationTest {

@Autowired

private MockMvc mockMvc;

@Autowired

private UserRepository userRepository;

@BeforeEach void setUp() {

userRepository.deleteAll();

userRepository.save(new User(1L, "Alice"));

}

@Test

void testGetUserById\_Integration() throws Exception { mockMvc.perform(MockMvcRequestBuilders.get("/users/1"))

.andExpect(status().isOk())

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

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

}

@Test

void testGetUserById\_NotFound() throws Exception { mockMvc.perform(MockMvcRequestBuilders.get("/users/999"))

.andExpect(status().isNotFound());

}

}

# Exercise 5: Test Controller POST Endpoint

## Test:

UserControllerPostTest.java

package com.example.demo.controller; import com.example.demo.model.User;

import com.example.demo.service.UserService; import com.fasterxml.jackson.databind.ObjectMapper; import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

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

class UserControllerPostTest {

@Autowired

private MockMvc mockMvc; @MockBean

private UserService userService; @Autowired

private ObjectMapper objectMapper; @Test

void testCreateUser() throws Exception {

// Arrange

User inputUser = new User(1L, "Bob");

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

// Act & Assert

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

.contentType(MediaType.APPLICATION\_JSON)

.content(objectMapper.writeValueAsString(inputUser)))

.andExpect(status().isOk())

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

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

verify(userService, times(1)).saveUser(any(User.class));

}

}

private MockMvc mockMvc;

@MockBean

private UserService userService;

@Autowired

private ObjectMapper objectMapper;

@Test

void testCreateUser() throws Exception {

// Arrange

User inputUser = new User(1L, "Bob");

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

// Act & Assert

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

.contentType(MediaType.APPLICATION\_JSON)

.content(objectMapper.writeValueAsString(inputUser)))

.andExpect(status().isOk())

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

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

verify(userService, times(1)).saveUser(any(User.class));

}

}

# Exercise 6: Test Service Exception Handling

Create Custom Exception: UserNotFoundException.java

package com.example.demo.exception;

public class UserNotFoundException extends RuntimeException { public UserNotFoundException(String message) {

super(message);

}

}

Update Service: UserService.java

package com.example.demo.service;

import com.example.demo.exception.UserNotFoundException; import com.example.demo.model.User;

import com.example.demo.repository.UserRepository;

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)

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

}

}

Unit Test for Exception: UserServiceExceptionTest.java

package com.example.demo.service;

import com.example.demo.exception.UserNotFoundException; import com.example.demo.repository.UserRepository;

import com.example.demo.model.User;

import org.junit.jupiter.api.BeforeEach; import org.junit.jupiter.api.Test;

import java.util.Optional;

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

class UserServiceExceptionTest { private UserRepository userRepository; private UserService userService;

@BeforeEach void setUp() {

userRepository = mock(UserRepository.class); userService = new UserService(

userService.userRepository = userRepository; // manually inject mock

}

@Test

void testGetUserById\_UserNotFound\_ThrowsException() {

// Arrange

Long id = 999L; when(userRepository.findById(id)).thenReturn(Optional.empty());

// Act & Assert

UserNotFoundException ex = assertThrows(UserNotFoundException.class, () -> { userService.getUserById(id);

});

assertEquals("User not found with ID: 999", ex.getMessage()); verify(userRepository, times(1)).findById(id);

}

}

Global Exception Handler: GlobalExceptionHandler.java

package com.example.demo.exception;

import org.springframework.http.ResponseEntity; import org.springframework.web.bind.annotation.\*; @ControllerAdvice

public class GlobalExceptionHandler {

@ExceptionHandler(UserNotFoundException.class)

public ResponseEntity<String> handleUserNotFound(UserNotFoundException ex) { return ResponseEntity.status(404).body(ex.getMessage());

}

}

# Exercise 7: Test Custom Repository Query

## Test:

Integration Test: UserRepositoryTest.java package com.example.demo.repository; import com.example.demo.model.User;

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.orm.jpa.DataJpaTest; import java.util.List;

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

class UserRepositoryTest {

@Autowired

private UserRepository userRepository; @BeforeEach

void setUp() { userRepository.deleteAll();

userRepository.save(new User(1L, "Alice")); userRepository.save(new User(2L, "Bob"));

userRepository.save(new User(3L, "Alice")); // Duplicate name @Test

void testFindByName\_ReturnsCorrectUsers() {

List<User> users = userRepository.findByName("Alice"); assertEquals(2, users.size());

for (User user : users) {

assertEquals("Alice", user.getName());

}

}

@Test

void testFindByName\_NoMatch() {

List<User> users = userRepository.findByName("Charlie"); assertTrue(users.isEmpty());

}

}

# Exercise 8: Test Controller Exception Handling

## Test:

UserControllerExceptionTest.java package com.example.demo.controller; import com.example.demo.model.User;

import com.example.demo.service.UserService;

import org.junit.jupiter.api.Test; import static org.mockito.Mockito.\*;

import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.\*; 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; @WebMvcTest(UserController.class)

class UserControllerExceptionTest { @Autowired

private MockMvc mockMvc;

@MockBean

private UserService userService; @Test

void testGetUser\_NotFound\_TriggersExceptionHandler() throws Exception {

// Arrange

Long userId = 999L; when(userService.getUserById(userId)).thenReturn(null);

// Act & Assert

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

.andExpect(status().isNotFound())

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

}

}

# Exercise 9: Parameterized Test with Junit

## Service Class: MathService.java

package com.example.demo.service;

import org.springframework.stereotype.Service;

@Service

public class MathService {

public int doubleValue(int input) { return input \* 2;

}

}

## Parameterized Test: MathServiceTest.java

package com.example.demo.service;

import org.junit.jupiter.params.ParameterizedTest; import org.junit.jupiter.params.provider.CsvSource; import static org.junit.jupiter.api.Assertions.assertEquals; class MathServiceTest {

private final MathService mathService = new MathService(); @ParameterizedTest

@CsvSource({ "1, 2",

"2, 4",

"0, 0",

"-3, -6",

"5, 10"

})

void testDoubleValue(int input, int expected) { int result = mathService.doubleValue(input); assertEquals(expected, result);

}

}