***WEEK 2***

***4. Spring Testing Exercises***

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

**CalculatorService.java**

package com.example.service;

import org.springframework.stereotype.Service;

*@Service*

public class CalculatorService {

public int add(int a, int b) {

return a + b;

}

}

**CalculatorServiceTest.java**

package com.example.service;

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

import org.junit.jupiter.api.Test;

public class CalculatorServiceTest {

*@Test*

public void testAdd() {

CalculatorService calculatorService = new CalculatorService();

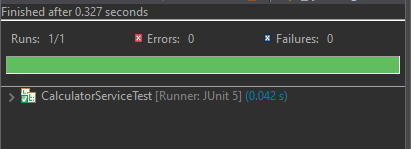
int result = calculatorService.add(10, 20);

*assertEquals*(30, result, "10 + 20 should be 30");

}

}

**OUTPUT**



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

**User.java**

package com.example.demo;

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 and 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 com.example.demo;

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

public interface UserRepository extends JpaRepository<User, Long> {

}

**UserService.java**

package com.example.demo;

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 com.example.demo;

import static org.mockito.Mockito.\*;

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

import java.util.Optional;

import org.junit.jupiter.api.Test;

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

import org.mockito.InjectMocks;

import org.mockito.Mock;

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

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

import org.springframework.test.context.junit.jupiter.SpringExtension;

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

*@ExtendWith*(SpringExtension.class)

public class UserServiceTest {

*@Mock*

private UserRepository userRepository;

*@InjectMocks*

private UserService userService;

*@Test*

public void testGetUserById() {

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

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

User result = userService.getUserById(1L);

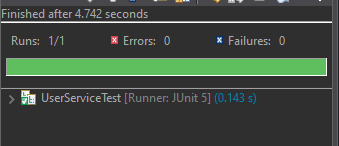
*assertNotNull*(result);

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

}

}

**OUTPUT**

****

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

**User.java**

package com.example.demo;

public class User {

private Long id;

private String name;

public User() {}

public User(Long id, String name) {

this.id = id;

this.name = name;

}

// Getters and 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; }

}

**UserController.java**

package com.example.demo;

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 com.example.demo;

public interface UserService {

User getUserById(Long id);

}

**UserControllerTest.java**

package com.example.demo;

import static org.mockito.Mockito.\*;

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

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

import com.fasterxml.jackson.databind.ObjectMapper;

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;

*@WebMvcTest*(UserController.class)

public class UserControllerTest {

*@Autowired*

private MockMvc mockMvc;

*@*~~MockBean~~

private UserService userService;

*@Test*

public void testGetUser() throws Exception {

// Given

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

*when*(userService.getUserById(1L)).thenReturn(mockUser);

// When & Then

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

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

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

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

}

}

**DemoApplication.java**

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@SpringBootApplication*

public class DemoApplication {

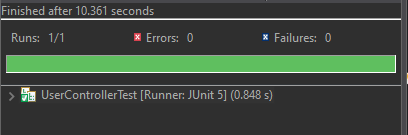
public static void main(String[] args) {

SpringApplication.*run*(DemoApplication.class, args);

}

}

**OUTPUT**

****

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

**AppUser.java**

package com.example.demo;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

*@Entity*

public class AppUser {

*@Id*

private Long id;

private String name;

public AppUser() {}

public AppUser(Long id, String name) {

this.id = id;

this.name = name;

}

// Getters and 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; }

}

**AppUserController.java**

package com.example.demo;

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

import org.springframework.http.ResponseEntity;

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

*@RestController*

*@RequestMapping*("/users")

public class AppUserController {

*@Autowired*

private AppUserService userService;

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

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

AppUser user = userService.getUserById(id);

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

}

}

**AppUserRepository.java**

package com.example.demo;

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

public interface AppUserRepository extends JpaRepository<AppUser, Long> {

}

**AppUserService.java**

package com.example.demo;

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

import org.springframework.stereotype.Service;

*@Service*

public class AppUserService {

*@Autowired*

private AppUserRepository userRepository;

public AppUser getUserById(Long id) {

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

}

}

**DemoApplication.java**

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@SpringBootApplication*

public class DemoApplication {

public static void main(String[] args) {

SpringApplication.*run*(DemoApplication.class, args);

}

}

**UserIntegrationTest.java**

package com.example.demo;

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

import org.junit.jupiter.api.BeforeEach;

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.boot.test.autoconfigure.jdbc.AutoConfigureTestDatabase;

import org.springframework.boot.test.web.server.LocalServerPort; // ✅ correct in Spring Boot 2.6+

import org.springframework.http.ResponseEntity;

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

*@AutoConfigureTestDatabase*

public class UserIntegrationTest {

*@LocalServerPort*

private int port;

*@Autowired*

private AppUserRepository userRepository;

*@Autowired*

private TestRestTemplate restTemplate;

*@BeforeEach*

public void setup() {

userRepository.deleteAll(); // clean DB

userRepository.save(new AppUser(1L, "John Doe")); // test data

}

*@Test*

public void testGetUserById() {

String url = "http://localhost:" + port + "/users/1";

ResponseEntity<AppUser> response = restTemplate.getForEntity(url, AppUser.class);

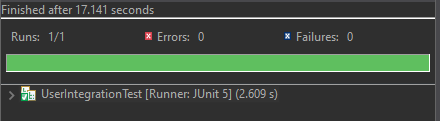
*assertThat*(response.getStatusCode().value()).isEqualTo(200);

*assertThat*(response.getBody().getName()).isEqualTo("John Doe");

}

}

**OUTPUT**

****

**Exercise 5: Test Controller POST Endpoint**

**DemoApplication.java**

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@SpringBootApplication*

public class DemoApplication {

public static void main(String[] args) {

SpringApplication.*run*(DemoApplication.class, args);

}

}

**User.java**

package com.example.demo;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

*@Entity*

*@Table*(name = "`user`") // Avoid SQL keyword conflict

public class User {

*@Id*

private Long id;

private String name;

public User() {}

public User(Long id, String name) {

this.id = id;

this.name = name;

}

// Getters and 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; }

}

**UserController.java**

package com.example.demo;

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;

*@PostMapping*

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

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

}

}

package com.example.demo;

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

public interface UserRepository extends JpaRepository<User, Long> {

}

**UserService.java**

package com.example.demo;

public interface UserService {

User saveUser(User user);

}

**UserServiceImpl.java**

package com.example.demo;

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

import org.springframework.stereotype.Service;

*@Service*

public class UserServiceImpl implements UserService {

*@Autowired*

private UserRepository userRepository;

*@Override*

public User saveUser(User user) {

return userRepository.save(user);

}

}

**UserControllerPostTest.java**

package com.example.demo;

import static org.mockito.Mockito.\*;

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

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

import com.fasterxml.jackson.databind.ObjectMapper;

import org.junit.jupiter.api.Test;

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

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

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

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

import org.springframework.http.MediaType;

*@WebMvcTest*(UserController.class)

public class UserControllerPostTest {

*@Autowired*

private MockMvc mockMvc;

*@*~~MockBean~~

private UserService userService;

private static ObjectMapper *mapper* = new ObjectMapper();

*@Test*

public void testCreateUser() throws Exception {

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

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

String json = *mapper*.writeValueAsString(user);

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

.contentType(MediaType.***APPLICATION\_JSON***)

.content(json))

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

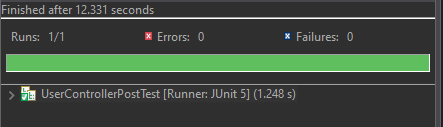
.andExpect(*jsonPath*("$.id").value(1L))

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

}

}

**OUTPUT**

****

**Exercise 6: Test Service Exception Handling**

**DemoApplication.java**

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@SpringBootApplication*

public class DemoApplication {

public static void main(String[] args) {

SpringApplication.*run*(DemoApplication.class, args);

}

}

**User.java**

package com.example.demo;

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;

}

// Getters and 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; }

}

**UserNotFoundException.java**

package com.example.demo;

public class UserNotFoundException extends RuntimeException {

public UserNotFoundException(String message) {

super(message);

}

}

**UserRepository.java**

package com.example.demo;

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

public interface UserRepository extends JpaRepository<User, Long> {

}

**UserService.java**

package com.example.demo;

public interface UserService {

User getUserById(Long id);

}

**UserServiceImpl.java**

package com.example.demo;

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

import org.springframework.stereotype.Service;

*@Service*

public class UserServiceImpl implements UserService {

*@Autowired*

private UserRepository userRepository;

*@Override*

public User getUserById(Long id) {

return getUserRepository().findById(id)

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

}

public UserRepository getUserRepository() {

return userRepository;

}

public void setUserRepository(UserRepository userRepository) {

this.userRepository = userRepository;

}

}

**UserServiceExceptionTest.java**

package com.example.demo;

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

import static org.mockito.Mockito.\*;

import java.util.Optional;

import org.junit.jupiter.api.Test;

public class UserServiceExceptionTest {

*@Test*

public void testGetUserById\_UserNotFound\_ThrowsException() {

// Step 1: Mock repository

UserRepository mockRepo = *mock*(UserRepository.class);

*when*(mockRepo.findById(999L)).thenReturn(Optional.*empty*());

// Step 2: Inject mock into service

UserServiceImpl service = new UserServiceImpl();

service.setUserRepository(mockRepo); // make field public OR use constructor

// Step 3: Assert exception

UserNotFoundException thrown = *assertThrows*(UserNotFoundException.class, () -> {

service.getUserById(999L);

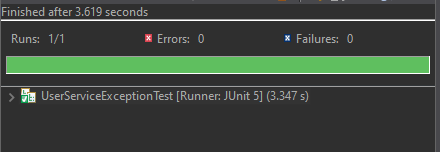
});

*assertEquals*("User not found with id: 999", thrown.getMessage());

}

}

**OUTPUT**

****

**Exercise 7: Test Custom Repository Query**

**DemoApplication.java**

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@SpringBootApplication*

public class DemoApplication {

public static void main(String[] args) {

SpringApplication.*run*(DemoApplication.class, args);

}

}

**User.java**

package com.example.demo;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

*@Entity*

*@Table*(name = "`user`") // quoted to avoid SQL keyword conflicts

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; }

}

package com.example.demo;

import java.util.List;

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

public interface UserRepository extends JpaRepository<User, Long> {

// custom finder

List<User> findByName(String name);

}

**UserRepositoryTest.java**

package com.example.demo;

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

import java.util.List;

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;

*@DataJpaTest*

public 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"));

}

*@Test*

void whenFindByName\_thenReturnMatchingUsers() {

// Act

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

// Assert

*assertThat*(alices)

.hasSize(2)

.extracting(User::getId)

.containsExactlyInAnyOrder(1L, 3L);

}

*@Test*

void whenFindByName\_notFound\_thenReturnEmptyList() {

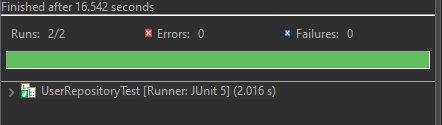
List<User> list = userRepository.findByName("Charlie");

*assertThat*(list).isEmpty();

}

}

**OUTPUT**

****

**Exercise 8: Test Controller Exception Handling**

**DemoApplication.java**

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@SpringBootApplication*

public class DemoApplication {

public static void main(String[] args) {

SpringApplication.*run*(DemoApplication.class, args);

}

}

**GlobalExceptionHandler.java**

package com.example.demo;

import java.util.NoSuchElementException;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

*@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.demo;

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

public String getName() { return name; }

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

}

**UserController.java**

package com.example.demo;

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 com.example.demo;

import java.util.NoSuchElementException;

import org.springframework.stereotype.Service;

*@Service*

public class UserService {

public User getUserById(Long id) {

throw new NoSuchElementException("User not found with id: " + id);

}

}

**UserControllerExceptionTest.java**

package com.example.demo;

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

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

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.\*;

*@WebMvcTest*(UserController.class)

public class UserControllerExceptionTest {

*@Autowired*

private MockMvc mockMvc;

*@*~~MockBean~~

private UserService userService;

*@Test*

public void testNotFoundException() throws Exception {

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

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

.contentType(MediaType.***APPLICATION\_JSON***))

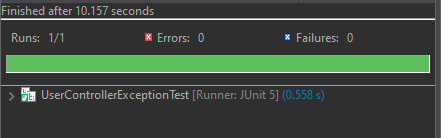
.andExpect(*status*().isNotFound())

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

}

}

**OUTPUT**

****

**Exercise 9: Parameterized Test with Junit**

**Calculator.java**

package com.example.demo;

public class Calculator {

public boolean isEven(int number) {

return number % 2 == 0;

}

}

**CalculatorTest.java**

package com.example.demo;

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

import org.junit.jupiter.params.ParameterizedTest;

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

public class CalculatorTest {

private final Calculator calculator = new Calculator();

*@ParameterizedTest*

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

public void testIsEven\_WithEvenNumbers\_ReturnsTrue(int number) {

*assertTrue*(calculator.isEven(number));

}

*@ParameterizedTest*

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

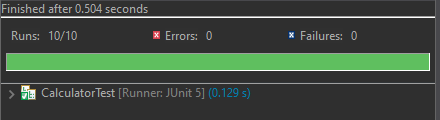
public void testIsEven\_WithOddNumbers\_ReturnsFalse(int number) {

*assertFalse*(calculator.isEven(number));

}

}

**OUTPUT**

****