**JUNIT TEST SPRING EXERCISES**

**PROJECT STRUCTURE:**

SpringTestExercises/

├── pom.xml

├── src/

│ ├── main/

│ │ └── java/

│ │ └── com/

│ │ └── example/

│ │ ├── SpringTestingExercisesApplication.java

│ │ ├── controller/

│ │ │ └── UserController.java

│ │ ├── exception/

│ │ │ └── GlobalExceptionHandler.java

│ │ ├── model/

│ │ │ └── User.java

│ │ ├── repository/

│ │ │ └── UserRepository.java

│ │ └── service/

│ │ ├── CalculatorService.java

│ │ └── UserService.java

│ └── test/

│ └── java/

│ └── com/

│ └── example/

│ ├── CalculatorServiceTest.java

│ ├── CustomQueryTest.java

│ ├── CreateUserControllerTest.java

│ ├── ControllerExceptionTest.java

│ ├── ParameterizedTestExample.java

│ ├── UserControllerTest.java

│ ├── UserIntegrationTest.java

│ └── UserServiceTest.java

**CalculatorService.java**

package com.example;

import org.springframework.stereotype.Service;

@Service

public class CalculatorService {

    public int add(int a, int b) {

        return a + b;

    }

}

**GlobalExceptionHandler.java**

package com.example;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.ControllerAdvice;

import org.springframework.web.bind.annotation.ExceptionHandler;

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");

    }

}

**SpringTestingExercisesApplication.java**

package com.example;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringTestingExercisesApplication {

    public static void main(String[] args) {

        SpringApplication.run(SpringTestingExercisesApplication.class, args);

    }

}

**User.java**

package com.example;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

@Entity

public class User {

    @Id

    private Long id;

    private String 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;

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));

    }

}

**UserRepository.java**

package com.example;

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

import java.util.List;

public interface UserRepository extends JpaRepository<User, Long> {

    List<User> findByName(String name);

}

**UserService.java**

package com.example;

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

import org.springframework.stereotype.Service;

import java.util.NoSuchElementException;

@Service

public class UserService {

    @Autowired

    private UserRepository userRepository;

    public User getUserById(Long id) {

        return userRepository.findById(id).orElseThrow(() -> new NoSuchElementException("User not found"));

    }

    public User saveUser(User user) {

        return userRepository.save(user);

    }

}

**CalculatorServiceTest.java**

package com.example;

import org.junit.jupiter.api.Test;

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

public class CalculatorServiceTest {

    CalculatorService calculatorService = new CalculatorService();

    @Test

    public void testAdd() {

        assertEquals(5, calculatorService.add(2, 3));

    }

}

**ControllerExceptionTest.java**

package com.example;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

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

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

import java.util.NoSuchElementException;

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)  // ✅ Only load UserController

public class ControllerExceptionTest {

    @Autowired

    private MockMvc mockMvc;

    @MockBean

    private UserService userService;  // ✅ mock the dependency

    @Test

    public void testNotFoundException() throws Exception {

        Long userId = 100L;

        Mockito.when(userService.getUserById(userId))

               .thenThrow(new NoSuchElementException("User not found"));

        mockMvc.perform(get("/users/" + userId))

               .andExpect(status().isNotFound())

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

    }

}

**CreateUserControllerTest.java**

package com.example;

import com.fasterxml.jackson.databind.ObjectMapper;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

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

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;

@WebMvcTest(UserController.class) // ✅ Only load UserController

public class CreateUserControllerTest {

    @Autowired

    private MockMvc mockMvc;

    @MockBean

    private UserService userService; // ✅ mock the dependency

    @Autowired

    private ObjectMapper objectMapper;

    @Test

    public void testCreateUser() throws Exception {

        User user = new User();

        user.setId(1L);

        user.setName("John Doe");

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

        mockMvc.perform(post("/users")

                .contentType(MediaType.APPLICATION\_JSON)

                .content(objectMapper.writeValueAsString(user)))

                .andExpect(status().isOk())

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

    }

}

**CustomQueryTest.java**

package com.example;

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

import static org.mockito.Mockito.\*;

import java.util.Arrays;

import java.util.List;

import org.junit.jupiter.api.Test;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.MockitoAnnotations;

public class CustomQueryTest {

    @Mock

    UserRepository userRepository;

    @InjectMocks

    UserService userService;

    public CustomQueryTest() {

        MockitoAnnotations.openMocks(this);

    }

    @Test

    public void testFindByName() {

        List<User> users = Arrays.asList(new User());

        when(userRepository.findByName("Alice")).thenReturn(users);

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

        assertEquals(1, result.size());

    }

}

**ParameterizedAdditionTest.java**

package com.example;

import org.junit.jupiter.params.ParameterizedTest;

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

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

public class ParameterizedAdditionTest {

    CalculatorService calculatorService = new CalculatorService();

    @ParameterizedTest

    @CsvSource({

        "1, 2, 3",

        "3, 4, 7",

        "10, 5, 15"

    })

    void testAdd(int a, int b, int expected) {

        assertEquals(expected, calculatorService.add(a, b));

    }

}

**ServiceExceptionTest.java**

package com.example;

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

import static org.mockito.Mockito.\*;

import java.util.Optional;

import org.junit.jupiter.api.Test;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.MockitoAnnotations;

import java.util.NoSuchElementException;

public class ServiceExceptionTest {

    @Mock

    UserRepository userRepository;

    @InjectMocks

    UserService userService;

    public ServiceExceptionTest() {

        MockitoAnnotations.openMocks(this);

    }

    @Test

    public void testUserNotFound() {

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

        assertThrows(NoSuchElementException.class, () -> userService.getUserById(99L));

    }

}

**UserControllerTest.java**

package com.example;

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

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

import org.mockito.Mockito;

@WebMvcTest(UserController.class)

class UserControllerTest {

    @Autowired

    private MockMvc mockMvc;

    @MockBean

    private UserService userService; // must be mocked if used in controller

    @Test

    void testGetUser() throws Exception {

        User user = new User();

        user.setId(1L);

        user.setName("Test User");

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

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

                .andExpect(status().isOk())

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

    }

}

**UserIntegrationTest.java**

package com.example;

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.web.server.LocalServerPort;

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

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

class UserIntegrationTest {

    @LocalServerPort

    private int port;

    @Autowired

    private TestRestTemplate restTemplate;

    @Test

    void testCreateAndGetUser() {

        User user = new User();

        user.setId(1L);

        user.setName("Integration User");

        // POST the user

        restTemplate.postForObject("http://localhost:" + port + "/users", user, User.class);

        // GET the user back

        User result = restTemplate.getForObject("http://localhost:" + port + "/users/1", User.class);

        assertThat(result).isNotNull();

        assertThat(result.getName()).isEqualTo("Integration User");

    }

}

**UserServiceTest.java**

package com.example;

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

    UserRepository userRepository;

    @InjectMocks

    UserService userService;

    public UserServiceTest() {

        MockitoAnnotations.openMocks(this);

    }

    @Test

    public void testGetUserById() {

        User user = new User();

        user.setId(1L);

        user.setName("Test User");

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

        User result = userService.getUserById(1L);

        assertEquals("Test User", result.getName());

    }

}

**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>

    <parent>

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

        <artifactId>spring-boot-starter-parent</artifactId>

        <version>3.1.2</version>

        <relativePath/>

    </parent>

    <groupId>com.example</groupId>

    <artifactId>SpringTestingExercises</artifactId>

    <version>1.0-SNAPSHOT</version>

    <packaging>jar</packaging>

    <name>Spring Testing Exercises</name>

    <properties>

        <java.version>17</java.version>

    </properties>

    <dependencies>

        <!-- Core Spring Boot Dependencies -->

        <dependency>

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

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

        </dependency>

        <dependency>

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

            <artifactId>spring-boot-starter-web</artifactId>

        </dependency>

        <dependency>

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

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

        </dependency>

        <!-- Testing Dependencies -->

        <dependency>

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

            <artifactId>spring-boot-starter-test</artifactId>

            <scope>test</scope>

        </dependency>

        <!-- JSON support -->

        <dependency>

            <groupId>com.fasterxml.jackson.core</groupId>

            <artifactId>jackson-databind</artifactId>

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <!-- Spring Boot Maven Plugin -->

            <plugin>

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

                <artifactId>spring-boot-maven-plugin</artifactId>

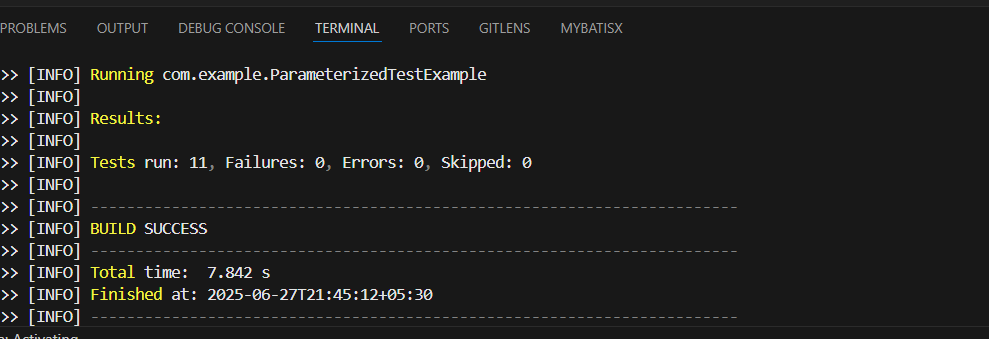
            </plugin>

        </plugins>

    </build>

</project>

**OUTPUT:**

****