**MOCKITO MOCK DEPENDENCIES EXERCISES:**

**PROJECT STRUCTURE:**

SpringMockingTests/

├── pom.xml

├── src/

│ ├── main/

│ │ ├── java/

│ │ │ └── com/

│ │ │ └── example/

│ │ │ ├── SpringMockingTestsApplication.java ← ✅ Main class

│ │ │ ├── controller/

│ │ │ │ └── UserController.java ← ✅ REST controller

│ │ │ ├── entity/

│ │ │ │ └── User.java ← ✅ JPA entity

│ │ │ ├── repository/

│ │ │ │ └── UserRepository.java ← ✅ JPA repository

│ │ │ └── service/

│ │ │ └── UserService.java ← ✅ Service with constructor injection

│ └── resources/

│ └── application.properties ← (optional)

│

├── src/

│ └── test/

│ └── java/

│ └── com/

│ └── example/

│ ├── UserIntegrationTest.java ← ✅ Integration test using @SpringBootTest + MockMvc

│ ├── controller/

│ │ └── UserControllerTest.java ← ✅ Unit test using mock UserService

│ └── service/

│ └── UserServiceTest.java ← ✅ Unit test using mock UserRepository

**UserController.java**

package com.example.controller;

import com.example.entity.User;

import com.example.service.UserService;

import org.springframework.http.ResponseEntity;

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

@RestController

@RequestMapping("/users")

public class UserController {

    private final UserService userService;

    // ✅ Constructor injection

    public UserController(UserService userService) {

        this.userService = userService;

    }

    @GetMapping("/{id}")

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

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

    }

}

**User.java**

package com.example.entity;

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; }

}

**UserRepository.java**

package com.example.repository;

import com.example.entity.User;

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

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

**UserService.java**

package com.example.service;

import com.example.entity.User;

import com.example.repository.UserRepository;

import org.springframework.stereotype.Service;

@Service

public class UserService {

    private final UserRepository userRepository;

    // ✅ Constructor Injection

    public UserService(UserRepository userRepository) {

        this.userRepository = userRepository;

    }

    public User getUserById(Long id) {

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

    }

}

**SpringMockingTestsApplication.java**

package com.example;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringMockingTestsApplication {

    public static void main(String[] args) {

        SpringApplication.run(SpringMockingTestsApplication.class, args);

    }

}

**UserControllerTest.java**

package com.example.controller;

import com.example.entity.User;

import com.example.service.UserService;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

import org.springframework.http.ResponseEntity;

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

public class UserControllerTest {

    @Test

    public void testGetUser() {

        // ✅ Create mock

        UserService mockService = Mockito.mock(UserService.class);

        // ✅ Inject mock via constructor

        UserController controller = new UserController(mockService);

        User mockUser = new User();

        mockUser.setId(1L);

        mockUser.setName("Test User");

        Mockito.when(mockService.getUserById(1L)).thenReturn(mockUser);

        ResponseEntity<User> response = controller.getUser(1L);

        assertEquals(200, response.getStatusCodeValue());

        assertEquals("Test User", response.getBody().getName());

    }

}

**UserServiceTest.java**

package com.example.service;

import com.example.entity.User;

import com.example.repository.UserRepository;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

import java.util.Optional;

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

public class UserServiceTest {

    @Test

    public void testGetUserById() {

        // Create mock repository

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

        // ✅ FIX: Inject mock via constructor

        UserService service = new UserService(mockRepo);

        // Sample user data

        User user = new User();

        user.setId(1L);

        user.setName("Test");

        // Stubbing

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

        // Execute

        User result = service.getUserById(1L);

        // Assert

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

    }

}

**UserIntegrationTest.java**

package com.example;

import com.example.entity.User;

import com.example.service.UserService;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

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.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.\*;

@SpringBootTest(classes = SpringMockingTestsApplication.class)

@AutoConfigureMockMvc

public class UserIntegrationTest {

    @Autowired

    private MockMvc mockMvc;

    @MockBean

    private UserService userService;

    @Test

    public void testGetUserEndpoint() throws Exception {

        User user = new User();

        user.setId(1L);

        user.setName("Mocked User");

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

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

                .andExpect(status().isOk())

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

    }

}

**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>SpringMockingTests</artifactId>

    <version>1.0-SNAPSHOT</version>

    <packaging>jar</packaging>

    <!-- ✅ Spring Boot Parent BOM -->

    <parent>

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

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

        <version>3.2.5</version>

        <relativePath/> <!-- Lookup parent from repository -->

    </parent>

    <properties>

        <java.version>17</java.version> <!-- or change to 23 if you're targeting Java 23 -->

    </properties>

    <dependencies>

        <!-- ✅ Spring Boot Starter Web -->

        <dependency>

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

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

        </dependency>

        <!-- ✅ Spring Data JPA -->

        <dependency>

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

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

        </dependency>

        <!-- ✅ In-Memory DB for Testing -->

        <dependency>

            <groupId>com.h2database</groupId>

            <artifactId>h2</artifactId>

            <scope>runtime</scope>

        </dependency>

        <!-- ✅ Spring Boot Test Starter -->

        <dependency>

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

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

            <scope>test</scope>

        </dependency>

        <!-- ✅ Mockito Inline (required for Java 17–23) -->

        <dependency>

            <groupId>org.mockito</groupId>

            <artifactId>mockito-inline</artifactId>

            <version>5.2.0</version>

            <scope>test</scope>

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <!-- ✅ Compiler Plugin -->

            <plugin>

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

                <artifactId>maven-compiler-plugin</artifactId>

                <version>3.11.0</version>

                <configuration>

                    <source>${java.version}</source>

                    <target>${java.version}</target>

                </configuration>

            </plugin>

        </plugins>

    </build>

</project>

OUTPUT:

