**WEEK-2**

**Spring Testing Exercises**

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

**Code:**

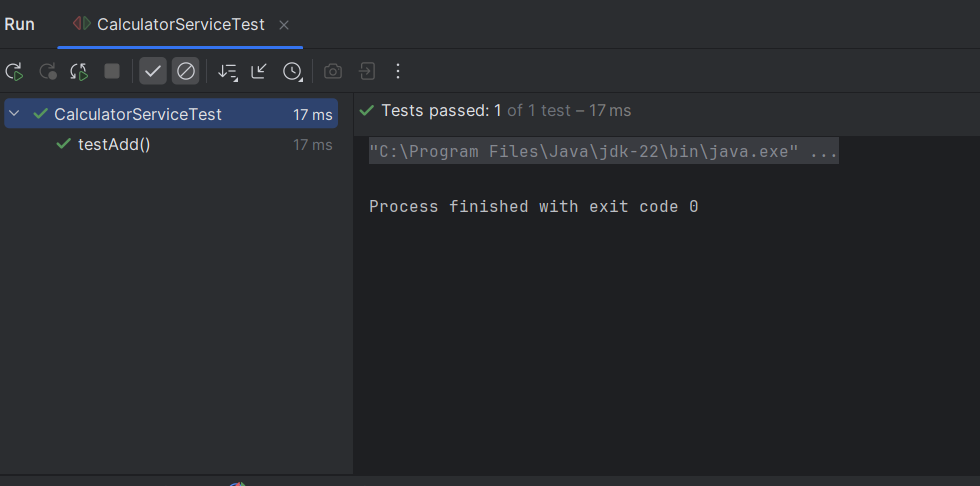
**CalculatorService.java**

package com.example;  
  
import org.springframework.stereotype.Service;  
  
@Service  
public class CalculatorService {  
 public int add(int a, int b) {  
 return a + b;  
 }  
}

**CalculatorServiceTest.java**

import com.example.CalculatorService;  
import org.junit.jupiter.api.Test;  
import static org.junit.jupiter.api.Assertions.\*;  
  
public class CalculatorServiceTest {  
  
 private final CalculatorService calculatorService = new CalculatorService();  
  
 @Test  
 public void testAdd() {  
 int result = calculatorService.add(2, 3);  
 *assertEquals*(5, result, "Addition should return the correct sum");  
 }  
}

**Output:**

****

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

**Code:**

**User.java**

package com.example;  
  
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  
 public Long getId() { return id; }  
 public String getName() { return name; }  
  
 // Setters  
 public void setId(Long id) { this.id = id; }  
 public void setName(String name) { this.name = name; }  
}

**UserRepository.java**

package com.example;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface UserRepository extends JpaRepository<User, Long> {  
}

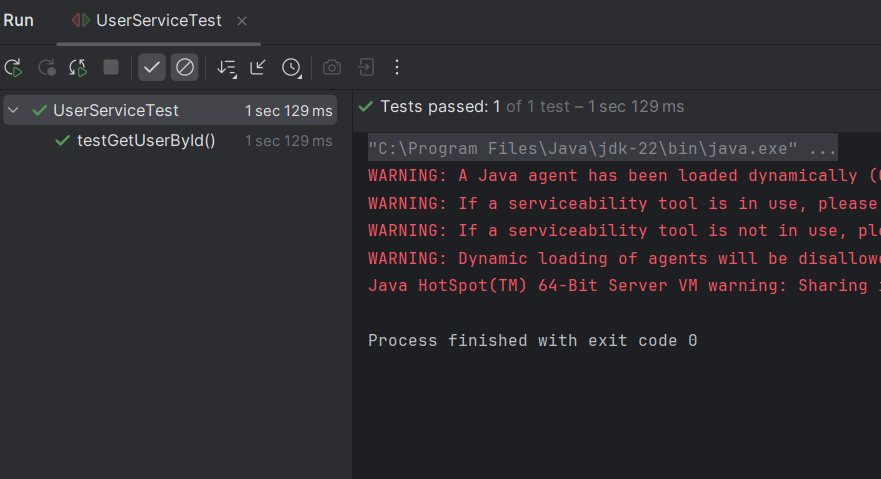
**UserService.java**

package com.example;  
  
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**

import com.example.\*;  
  
import org.junit.jupiter.api.Test;  
import static org.junit.jupiter.api.Assertions.\*;  
import static org.mockito.Mockito.\*;  
  
import java.util.Optional;  
  
import org.mockito.InjectMocks;  
import org.mockito.Mock;  
import org.mockito.junit.jupiter.MockitoExtension;  
import org.junit.jupiter.api.extension.ExtendWith;  
  
@ExtendWith(MockitoExtension.class)  
public class UserServiceTest {  
  
 @Mock  
 private UserRepository userRepository;  
  
 @InjectMocks  
 private UserService userService;  
  
 @Test  
 public void testGetUserById() {  
 // Arrange  
 User user = new User(1L, "Sanjai");  
 *when*(userRepository.findById(1L)).thenReturn(Optional.*of*(user));  
  
 // Act  
 User result = userService.getUserById(1L);  
  
 // Assert  
 *assertNotNull*(result);  
 *assertEquals*("Sanjai", result.getName());  
 *verify*(userRepository).findById(1L);  
 }  
}

**Output:**

****

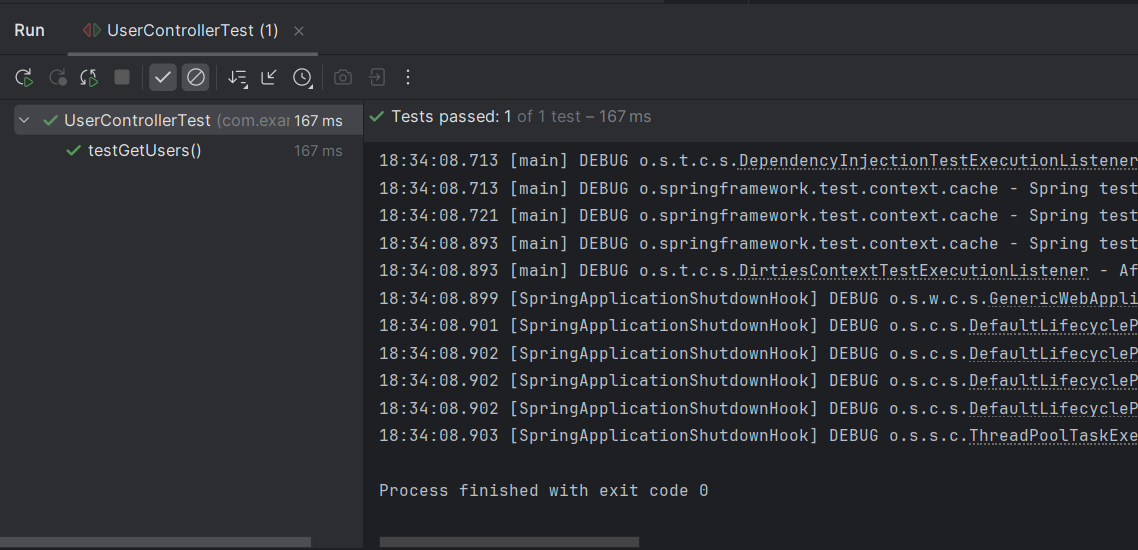
**Exercise 3: Testing a REST Controller with MockMvc**

**Code:**

**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.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 UserControllerTest {  
  
 @Autowired  
 private MockMvc mockMvc;  
  
 @MockBean  
 private UserService userService;  
  
 @Test  
 void testGetUsers() throws Exception {  
 *when*(userService.getAllUsers()).thenReturn("Mocked User List");  
  
 mockMvc.perform(*get*("/users"))  
 .andExpect(*status*().isOk())  
 .andExpect(*content*().string("Mocked User List"));  
 }  
}

**Output:**

****

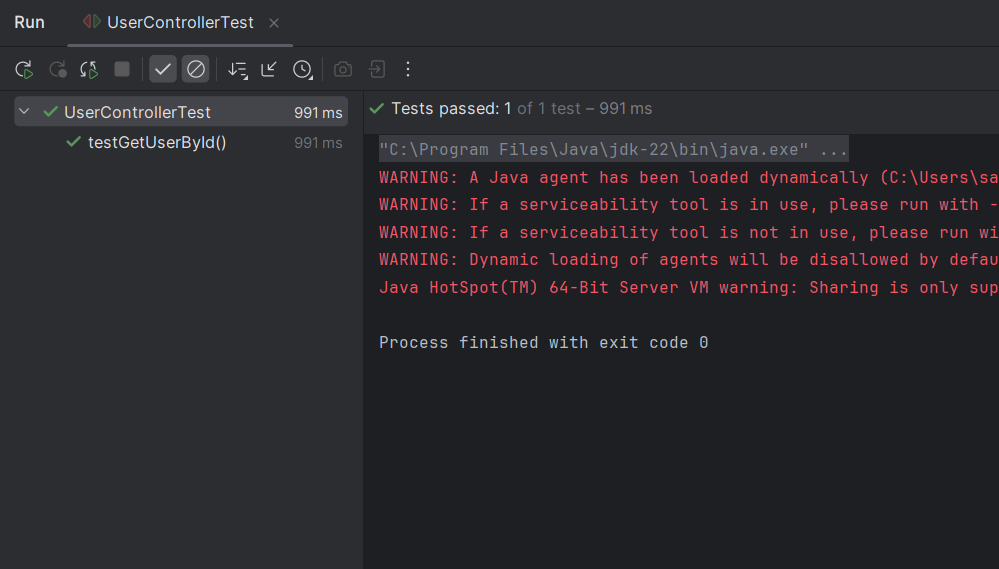
**Exercise 4: Integration Test with Spring Boot**

**Code:**

**UserControllerTest.java**

import com.example.UserService;  
import com.example.UserController;  
import com.example.User;  
  
import org.junit.jupiter.api.Test;  
import org.junit.jupiter.api.extension.ExtendWith;  
import org.mockito.InjectMocks;  
import org.mockito.Mock;  
import org.mockito.junit.jupiter.MockitoExtension;  
import org.springframework.http.ResponseEntity;  
import static org.junit.jupiter.api.Assertions.*assertEquals*;  
import static org.mockito.Mockito.*when*;  
  
@ExtendWith(MockitoExtension.class)  
public class UserControllerTest {  
  
 @Mock  
 private UserService userService;  
  
 @InjectMocks  
 private UserController userController;  
  
 @Test  
 public void testGetUserById() {  
 // Arrange: Set up the mock data  
 Long userId = 1L;  
 User mockUser = new User();  
 mockUser.setId(userId);  
 mockUser.setName("Test User");  
  
 // Tell Mockito what to do when userService.getUserById is called  
 *when*(userService.getUserById(userId)).thenReturn(mockUser);  
  
 // Act: Call the controller method  
 ResponseEntity<User> response = userController.getUser(userId);  
  
 // Assert: Check the results  
 *assertEquals*(200, response.getStatusCode().value()); // Updated to fix deprecation  
 *assertEquals*(mockUser, response.getBody());  
 *assertEquals*("Test User", response.getBody().getName());  
 }  
}

**Output:**

****

**Exercise 5: Test Controller POST Endpoint**

**Code:**

**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; }  
}

**UserService.java**

package com.example;  
  
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);  
 }  
}

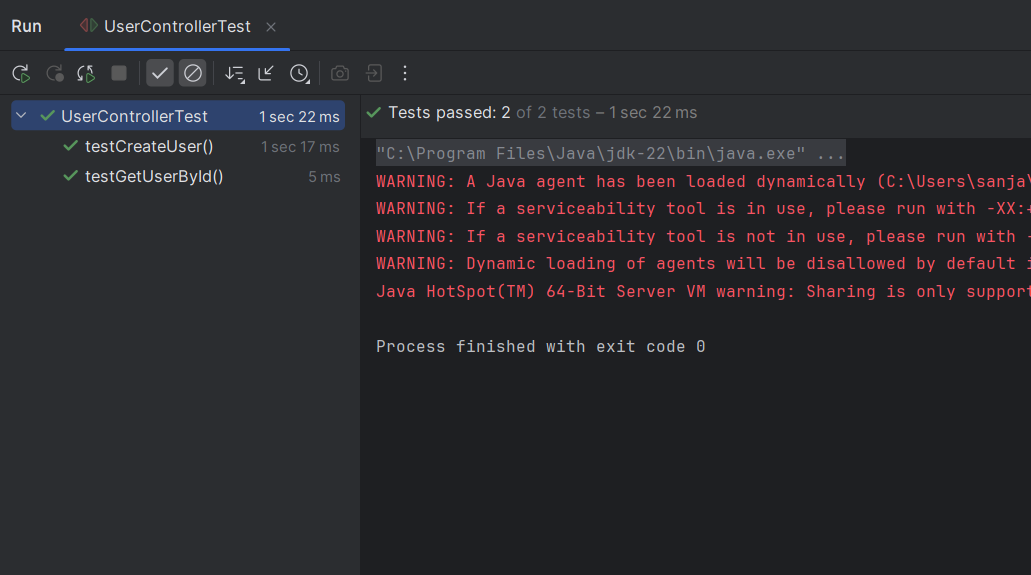
**UserController.java**

package com.example;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
@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));  
 }  
}

**UsercontrollerTest.java**

import com.example.\*;  
  
import com.example.User;  
import com.example.UserService;  
import com.example.UserController;  
import org.junit.jupiter.api.Test;  
import org.junit.jupiter.api.extension.ExtendWith;  
import org.mockito.InjectMocks;  
import org.mockito.Mock;  
import org.mockito.junit.jupiter.MockitoExtension;  
import org.springframework.http.ResponseEntity;  
import static org.junit.jupiter.api.Assertions.*assertEquals*;  
import static org.mockito.Mockito.*when*;  
  
@ExtendWith(MockitoExtension.class)  
public class UserControllerTest {  
  
 @Mock  
 private UserService userService;  
  
 @InjectMocks  
 private UserController userController;  
  
 @Test  
 public void testGetUserById() {  
 // Arrange: Set up the mock data  
 Long userId = 1L;  
 User mockUser = new User();  
 mockUser.setId(userId);  
 mockUser.setName("Test User");  
  
 // Tell Mockito what to do when userService.getUserById is called  
 *when*(userService.getUserById(userId)).thenReturn(mockUser);  
  
 // Act: Call the controller method  
 ResponseEntity<User> response = userController.getUser(userId);  
  
 // Assert: Check the results  
 *assertEquals*(200, response.getStatusCode().value());  
 *assertEquals*(mockUser, response.getBody());  
 *assertEquals*("Test User", response.getBody().getName());  
 }  
  
 @Test  
 public void testCreateUser() {  
 // Arrange: Set up the mock data  
 User inputUser = new User();  
 inputUser.setId(1L);  
 inputUser.setName("New User");  
  
 User savedUser = new User();  
 savedUser.setId(1L);  
 savedUser.setName("New User");  
  
 // Tell Mockito what to do when userService.saveUser is called  
 *when*(userService.saveUser(inputUser)).thenReturn(savedUser);  
  
 // Act: Call the controller method  
 ResponseEntity<User> response = userController.createUser(inputUser);  
  
 // Assert: Check the results  
 *assertEquals*(200, response.getStatusCode().value());  
 *assertEquals*(savedUser, response.getBody());  
 *assertEquals*("New User", response.getBody().getName());  
 }  
}

**Output:**

****

**Exercise 6: Test Service Exception Handling**

**Code:**

**UserNotFoundException.java**

package com.example;  
  
public class UserNotFoundException extends RuntimeException {  
 public UserNotFoundException(String message) {  
 super(message);  
 }  
}

**UserService.java**

package com.example;  
  
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));  
 }  
  
 public User saveUser(User user) {  
 return userRepository.save(user);  
 }  
}

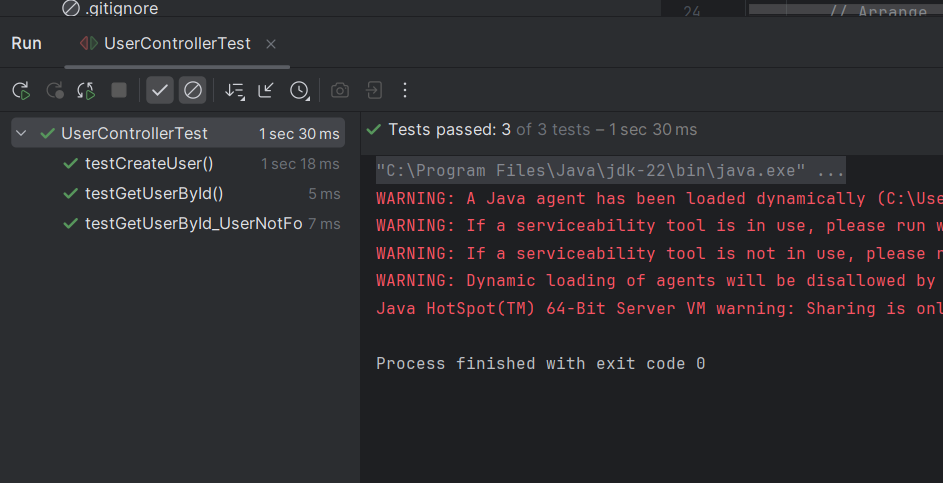
**UserController.java**

package com.example;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
@RestController  
@RequestMapping("/users")  
public class UserController {  
 @Autowired  
 private UserService userService;  
  
 @GetMapping("/{id}")  
 public ResponseEntity<User> getUser(@PathVariable Long id) {  
 try {  
 User user = userService.getUserById(id);  
 return ResponseEntity.*ok*(user);  
 } catch (UserNotFoundException e) {  
 return ResponseEntity.*notFound*().build();  
 }  
 }  
  
 @PostMapping  
 public ResponseEntity<User> createUser(@RequestBody User user) {  
 return ResponseEntity.*ok*(userService.saveUser(user));  
 }  
}

**UserControllerTest.java**

import com.example.\*;  
  
import org.junit.jupiter.api.Test;  
import org.junit.jupiter.api.extension.ExtendWith;  
import org.mockito.InjectMocks;  
import org.mockito.Mock;  
import org.mockito.junit.jupiter.MockitoExtension;  
import org.springframework.http.ResponseEntity;  
import static org.junit.jupiter.api.Assertions.*assertEquals*;  
import static org.junit.jupiter.api.Assertions.*assertNull*;  
import static org.mockito.Mockito.*when*;  
  
@ExtendWith(MockitoExtension.class)  
public class UserControllerTest {  
  
 @Mock  
 private UserService userService;  
  
 @InjectMocks  
 private UserController userController;  
  
 @Test  
 public void testGetUserById() {  
 // Arrange  
 Long userId = 1L;  
 User mockUser = new User();  
 mockUser.setId(userId);  
 mockUser.setName("Test User");  
  
 *when*(userService.getUserById(userId)).thenReturn(mockUser);  
  
 // Act  
 ResponseEntity<User> response = userController.getUser(userId);  
  
 // Assert  
 *assertEquals*(200, response.getStatusCode().value());  
 *assertEquals*(mockUser, response.getBody());  
 *assertEquals*("Test User", response.getBody().getName());  
 }  
  
 @Test  
 public void testCreateUser() {  
 // Arrange  
 User inputUser = new User();  
 inputUser.setId(1L);  
 inputUser.setName("New User");  
  
 User savedUser = new User();  
 savedUser.setId(1L);  
 savedUser.setName("New User");  
  
 *when*(userService.saveUser(inputUser)).thenReturn(savedUser);  
  
 // Act  
 ResponseEntity<User> response = userController.createUser(inputUser);  
  
 // Assert  
 *assertEquals*(200, response.getStatusCode().value());  
 *assertEquals*(savedUser, response.getBody());  
 *assertEquals*("New User", response.getBody().getName());  
 }  
  
 @Test  
 public void testGetUserById\_UserNotFound() {  
 // Arrange  
 Long userId = 2L;  
 *when*(userService.getUserById(userId)).thenThrow(new UserNotFoundException("User not found with id: " + userId));  
  
 // Act  
 ResponseEntity<User> response = userController.getUser(userId);  
  
 // Assert  
 *assertEquals*(404, response.getStatusCode().value());  
 *assertNull*(response.getBody());  
 }  
}

**Output:**

****

**Exercise 7: Test Custom Repository Query**

**Code:**