**WEEK-2**

**Mocking Dependencies in Spring Tests using Mockito**

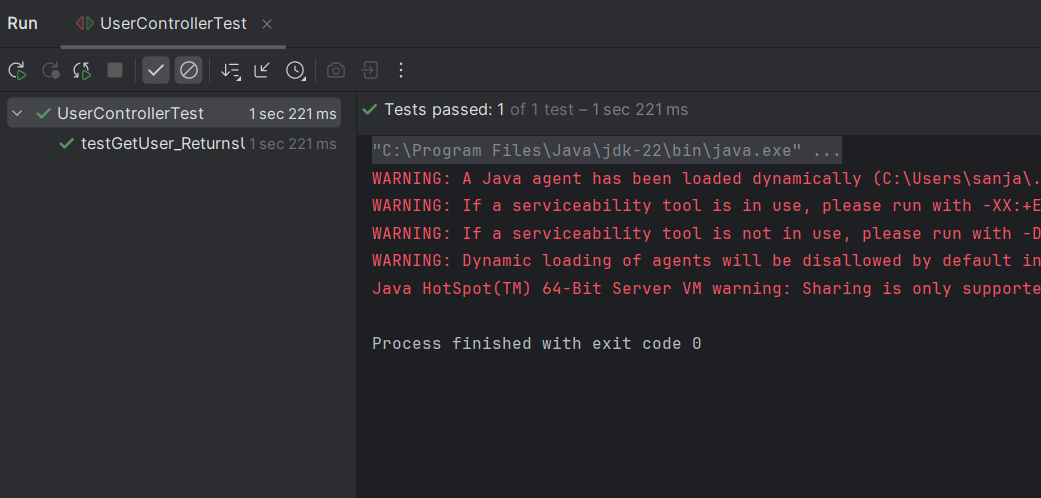
**Exercise 1: Mocking a Service Dependency in a Controller Test**

**Code:**

**UserControllerTest.java**

import com.example.\*;  
  
  
import org.junit.jupiter.api.BeforeEach;  
import org.junit.jupiter.api.Test;  
import org.mockito.InjectMocks;  
import org.mockito.Mock;  
  
import org.mockito.MockitoAnnotations;  
import org.springframework.http.ResponseEntity;  
  
import static org.junit.jupiter.api.Assertions.*assertEquals*;  
import static org.mockito.Mockito.\*;  
  
public class UserControllerTest {  
  
 @Mock  
 private UserService userService;  
  
 @InjectMocks  
 private UserController userController;  
  
 @BeforeEach  
 void setUp() {  
 MockitoAnnotations.*openMocks*(this); // Initializes @Mock and @InjectMocks  
 }  
  
 @Test  
 void testGetUser\_ReturnsUser() {  
 // Arrange  
 Long userId = 1L;  
 User mockUser = new User();  
 mockUser.setId(userId);  
 mockUser.setName("Thilagar");  
  
 *when*(userService.getUserById(userId)).thenReturn(mockUser);  
  
 // Act  
 ResponseEntity<User> response = userController.getUser(userId);  
  
 // Assert  
 *assertEquals*(200, response.getStatusCodeValue());  
 *assertEquals*("Sanjai", response.getBody().getName());  
 *assertEquals*(userId, response.getBody().getId());  
  
 *verify*(userService, *times*(1)).getUserById(userId); // Ensure the mock was called  
 }  
}

**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;  
  
 // 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;  
 }  
}

**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);  
 }  
}

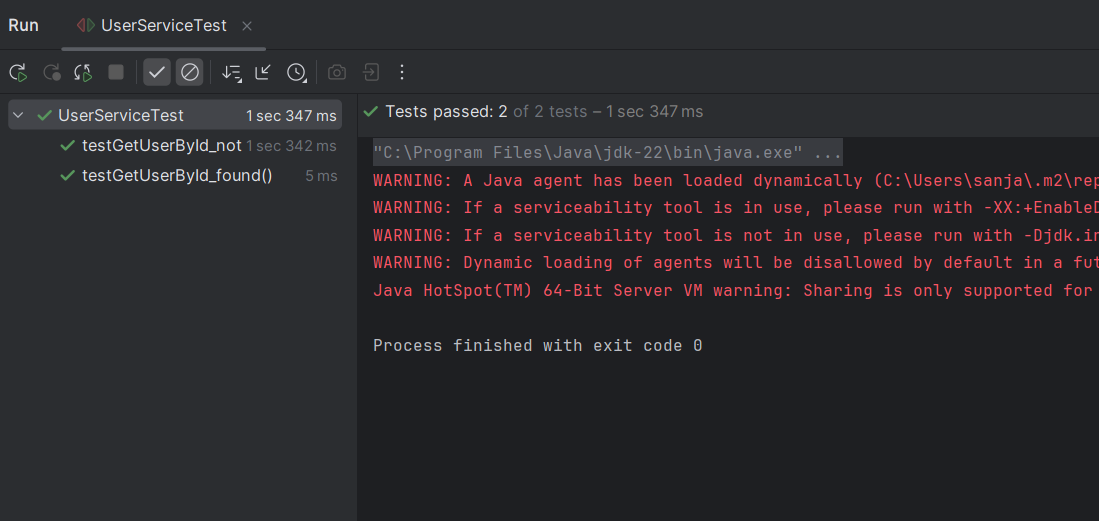
**UserRepository.java**

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

**UserServiceTest.java**

import com.example.\*;  
  
import org.junit.jupiter.api.BeforeEach;  
import org.junit.jupiter.api.Test;  
import org.mockito.InjectMocks;  
import org.mockito.Mock;  
  
import static org.mockito.Mockito.*when*;  
import static org.junit.jupiter.api.Assertions.\*;  
  
import org.mockito.MockitoAnnotations;  
  
import java.util.Optional;  
  
public class UserServiceTest {  
  
 @Mock  
 private UserRepository userRepository;  
  
 @InjectMocks  
 private UserService userService;  
  
 @BeforeEach  
 void setup() {  
 MockitoAnnotations.*openMocks*(this); // Initialize mocks  
 }  
  
 @Test  
 void testGetUserById\_found() {  
 User user = new User();  
 user.setId(1L);  
 user.setName("Thilagar");  
  
 *when*(userRepository.findById(1L)).thenReturn(Optional.*of*(user));  
  
 User result = userService.getUserById(1L);  
 *assertNotNull*(result);  
 *assertEquals*("Sanjai", result.getName());  
 }  
  
 @Test  
 void testGetUserById\_notFound() {  
 *when*(userRepository.findById(2L)).thenReturn(Optional.*empty*());  
  
 User result = userService.getUserById(2L);  
 *assertNull*(result);  
 }  
}

**Output:**

****

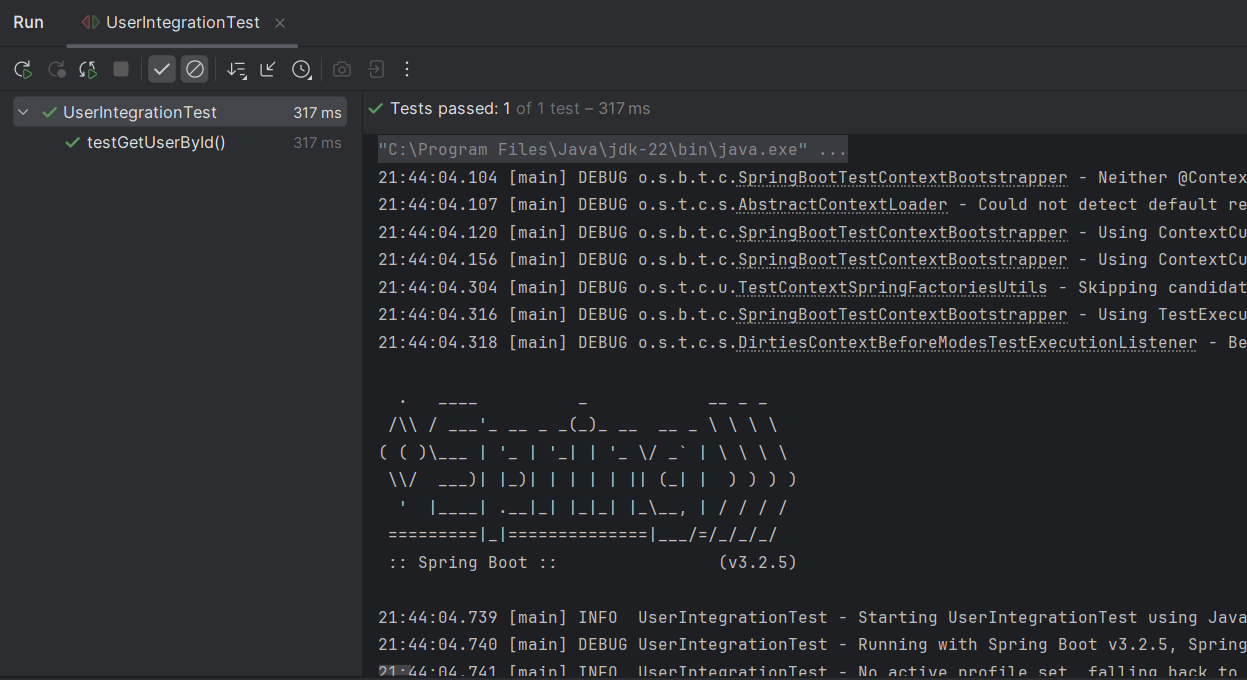
**Exercise 3: Mocking a Service Dependency in an Integration Test**

**Code:**

**UserIntegretionTest.java**

import com.example.\*;  
  
  
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.mock.mockito.MockBean;  
import org.springframework.boot.test.context.SpringBootTest;  
  
import org.springframework.http.MediaType;  
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.\*;  
  
@SpringBootTest(classes = Slf4jApplication.class) // 👈 Add your main class here  
@AutoConfigureMockMvc  
public class UserIntegrationTest {  
  
 @Autowired  
 private MockMvc mockMvc;  
  
 @MockBean  
 private UserService userService;  
  
 @Test  
 void testGetUserById() throws Exception {  
 User user = new User();  
 user.setId(1L);  
 user.setName("Thilagar");  
  
 *when*(userService.getUserById(1L)).thenReturn(user);  
  
 mockMvc.perform(*get*("/users/1")  
 .contentType(MediaType.*APPLICATION\_JSON*))  
 .andExpect(*status*().isOk())  
 .andExpect(*jsonPath*("$.id").value(1L))  
 .andExpect(*jsonPath*("$.name").value("Thilagar"));  
 }  
}

**Output:**

****