**JUNIT - SPRING TESTING EXERCISES**

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

**// CalculatorService.java** (File: src/main/java/com/example/calculator/CalculatorService.java)

package com.example.calculator;

import org.springframework.stereotype.Service;

@Service

public class CalculatorService {

public int add(int a, int b) {

return a + b;

}

}

**// CalculatorServiceTest.java** (File: src/test/java/com/example/calculator/CalculatorServiceTest.java)

package com.example.calculator;

import org.junit.jupiter.api.Test;

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

public class CalculatorServiceTest {

CalculatorService calculatorService = new CalculatorService();

@Test

void testAdd() {

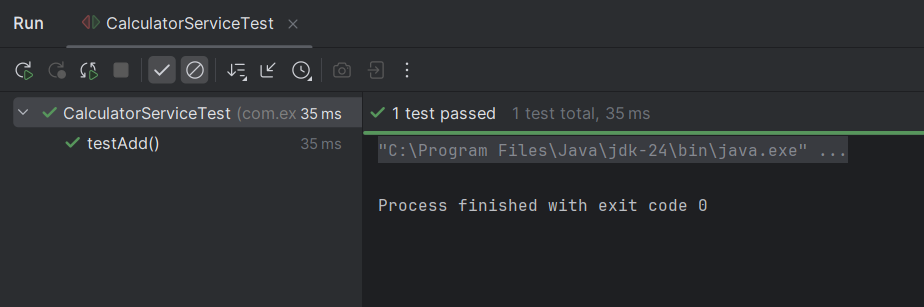
int result = calculatorService.add(2, 3);

assertEquals(5, result, "Addition result should be 5");

}

}

**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;

// 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.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;

public class UserServiceTest {

@Mock

private UserRepository userRepository;

@InjectMocks

private UserService userService;

public UserServiceTest() {

MockitoAnnotations.openMocks(this); // Initialize @Mock and @InjectMocks

}

@Test

void testGetUserById() {

// Arrange

User mockUser = new User();

mockUser.setId(1L);

mockUser.setName("Alice");

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

// Act

User result = userService.getUserById(1L);

// Assert

assertNotNull(result);

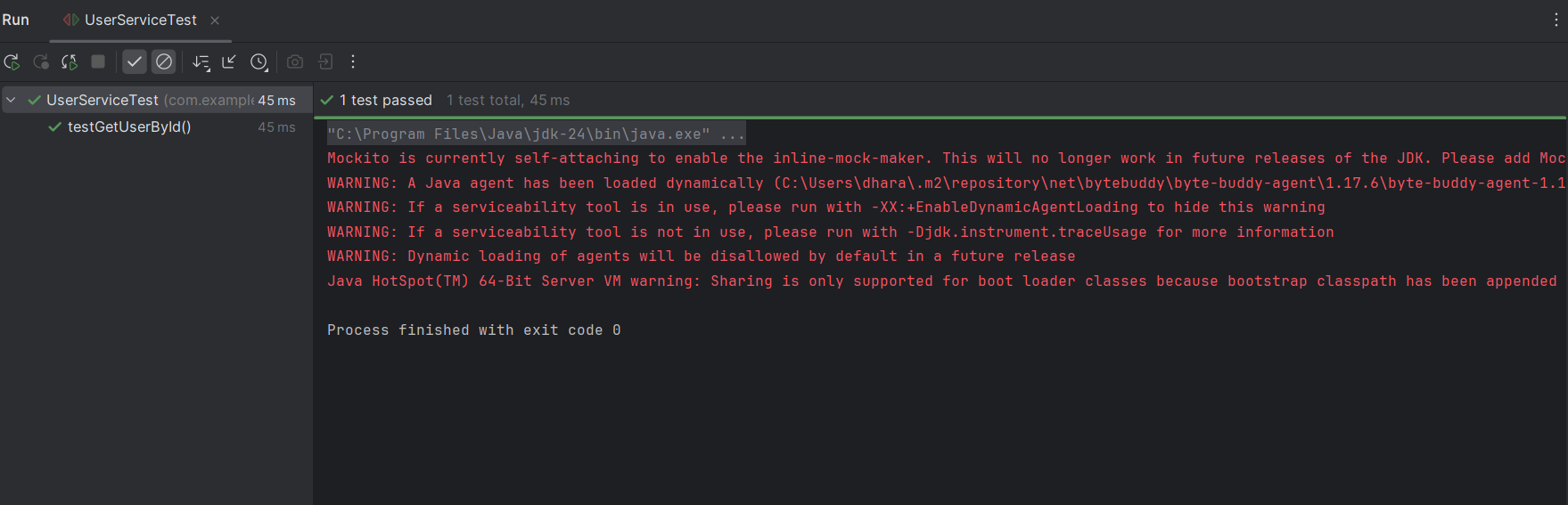
assertEquals(1L, result.getId());

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

}

}

**Output:**



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

**// 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 org.springframework.stereotype.Service;

@Service

public class UserService {

public User getUserById(Long id) {

return null; // mocked in test

}

}

**// User.java**

package com.example.demo;

public class User {

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; }

}

**// UserControllerTest.java**

package com.example.demo;

import org.junit.jupiter.api.Test;

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

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

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.\*;

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

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

import org.springframework.http.MediaType;

@WebMvcTest(UserController.class)

public class UserControllerTest {

@Autowired

private MockMvc mockMvc;

@MockBean

private UserService userService;

@Test

public void testGetUserById() throws Exception {

User mockUser = new User();

mockUser.setId(1L);

mockUser.setName("Alice");

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

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

.accept(MediaType.APPLICATION\_JSON))

.andExpect(status().isOk())

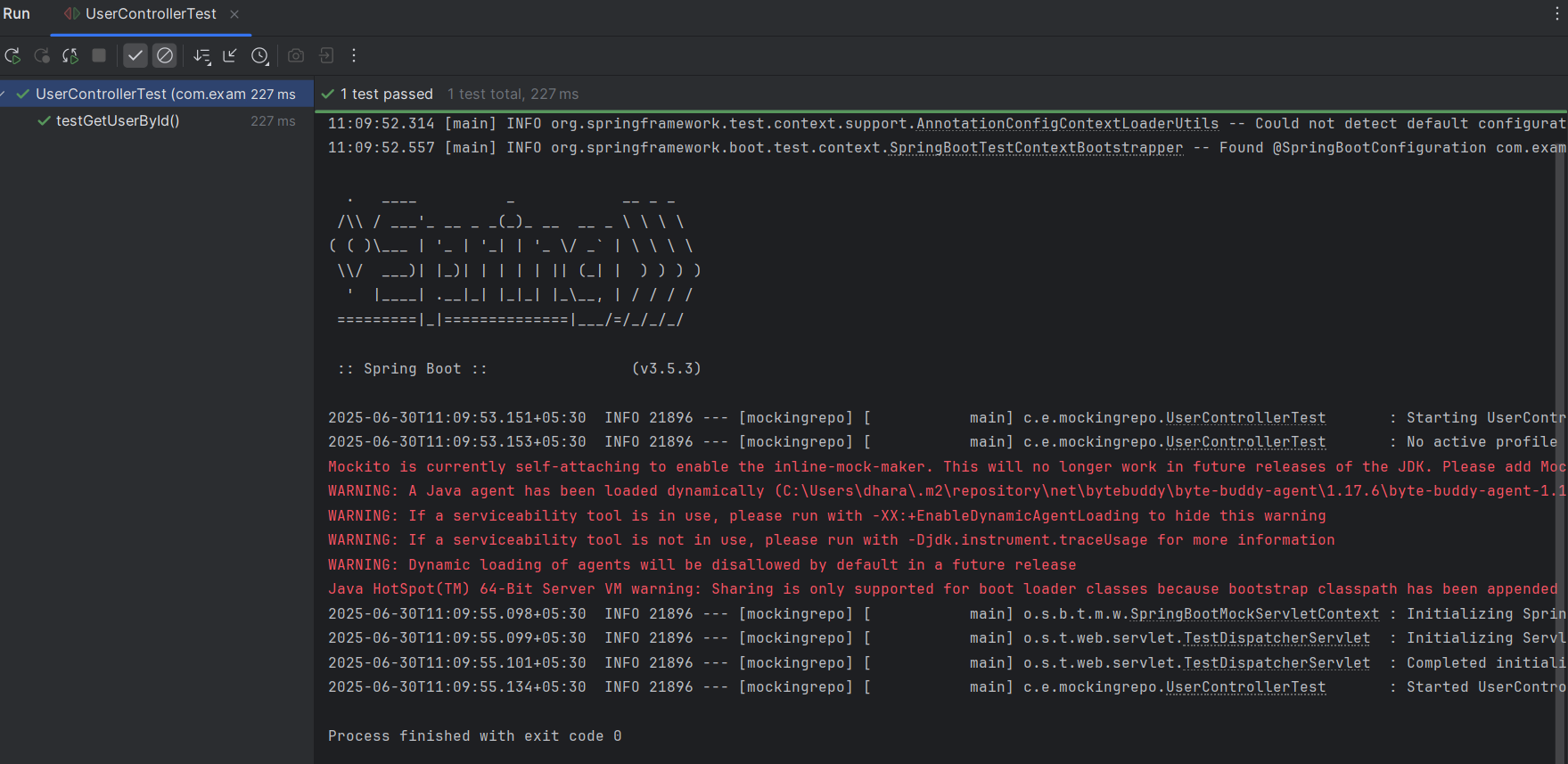
.andExpect(jsonPath("$.id").value(1L))

.andExpect(jsonPath("$.name").value("Alice"));

}

}

**Output:**



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

**// User.java**

package com.example.demo;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

@Entity

@Table(name = "users")

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

public User() {}

// 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 saveUser(User user) {

return userRepository.save(user);

}

public User getUser(Long id) {

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

}

}

**// 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));

}

@GetMapping("/{id}")

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

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

}

}

**// UserIntegrationTest.java**

package com.example.demo;

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

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 org.springframework.http.\*;

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

public class UserIntegrationTest {

@LocalServerPort

private int port;

@Autowired

private TestRestTemplate restTemplate;

@Test

void testCreateAndGetUser() {

// Prepare request body

User user = new User();

user.setName("Test User");

// POST /users

ResponseEntity<User> postResponse = restTemplate.postForEntity("http://localhost:" + port + "/users", user, User.class);

assertThat(postResponse.getStatusCode()).isEqualTo(HttpStatus.OK);

assertThat(postResponse.getBody()).isNotNull();

assertThat(postResponse.getBody().getId()).isNotNull();

Long userId = postResponse.getBody().getId();

// GET /users/{id}

ResponseEntity<User> getResponse = restTemplate.getForEntity("http://localhost:" + port + "/users/" + userId, User.class);

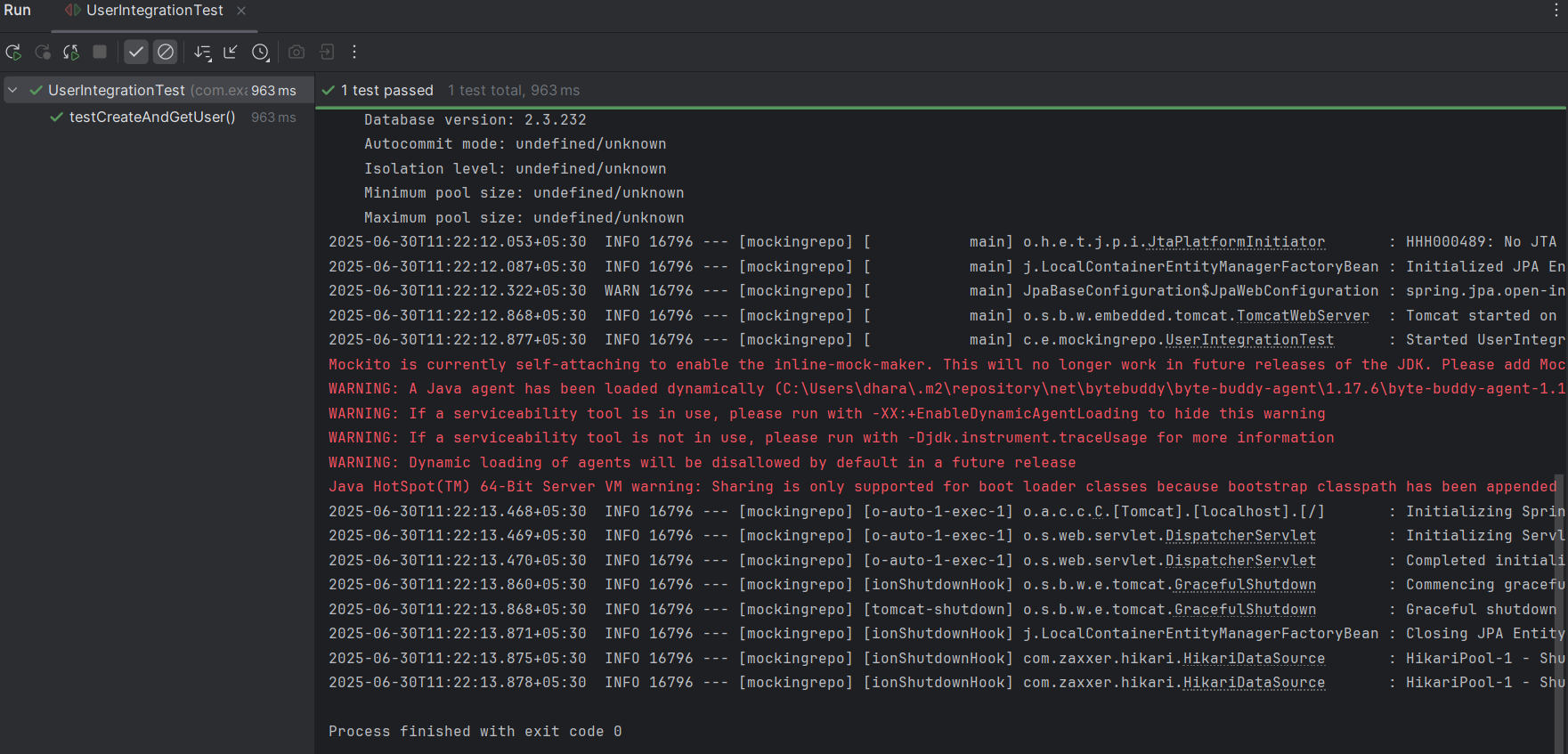
assertThat(getResponse.getStatusCode()).isEqualTo(HttpStatus.OK);

assertThat(getResponse.getBody().getName()).isEqualTo("Test User");

}

}

**Output:**



**Exercise 5: Test Controller POST Endpoint**

**// User.java**

package com.example.demo;

public class User {

private Long id;

private String name;

public User() {}

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.demo;

import org.springframework.stereotype.Service;

@Service

public class UserService {

public User saveUser(User user) {

// Normally you'd save to the DB; here we mock this

user.setId(1L); // Simulate DB-generated ID

return user;

}

}

**// 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) {

User savedUser = userService.saveUser(user);

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

}

}

**// UserControllerPostTest.java**

package com.example.demo;

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.http.MediaType;

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

import static org.mockito.Mockito.when;

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

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

@WebMvcTest(UserController.class)

public class UserControllerPostTest {

@Autowired

private MockMvc mockMvc;

@MockBean

private UserService userService;

private final ObjectMapper objectMapper = new ObjectMapper();

@Test

void testCreateUser() throws Exception {

// Arrange

User inputUser = new User();

inputUser.setName("Test User");

User savedUser = new User();

savedUser.setId(1L);

savedUser.setName("Test User");

when(userService.saveUser(inputUser)).thenReturn(savedUser);

// Act & Assert

mockMvc.perform(post("/users")

.contentType(MediaType.APPLICATION\_JSON)

.content(objectMapper.writeValueAsString(inputUser)))

.andExpect(status().isOk())

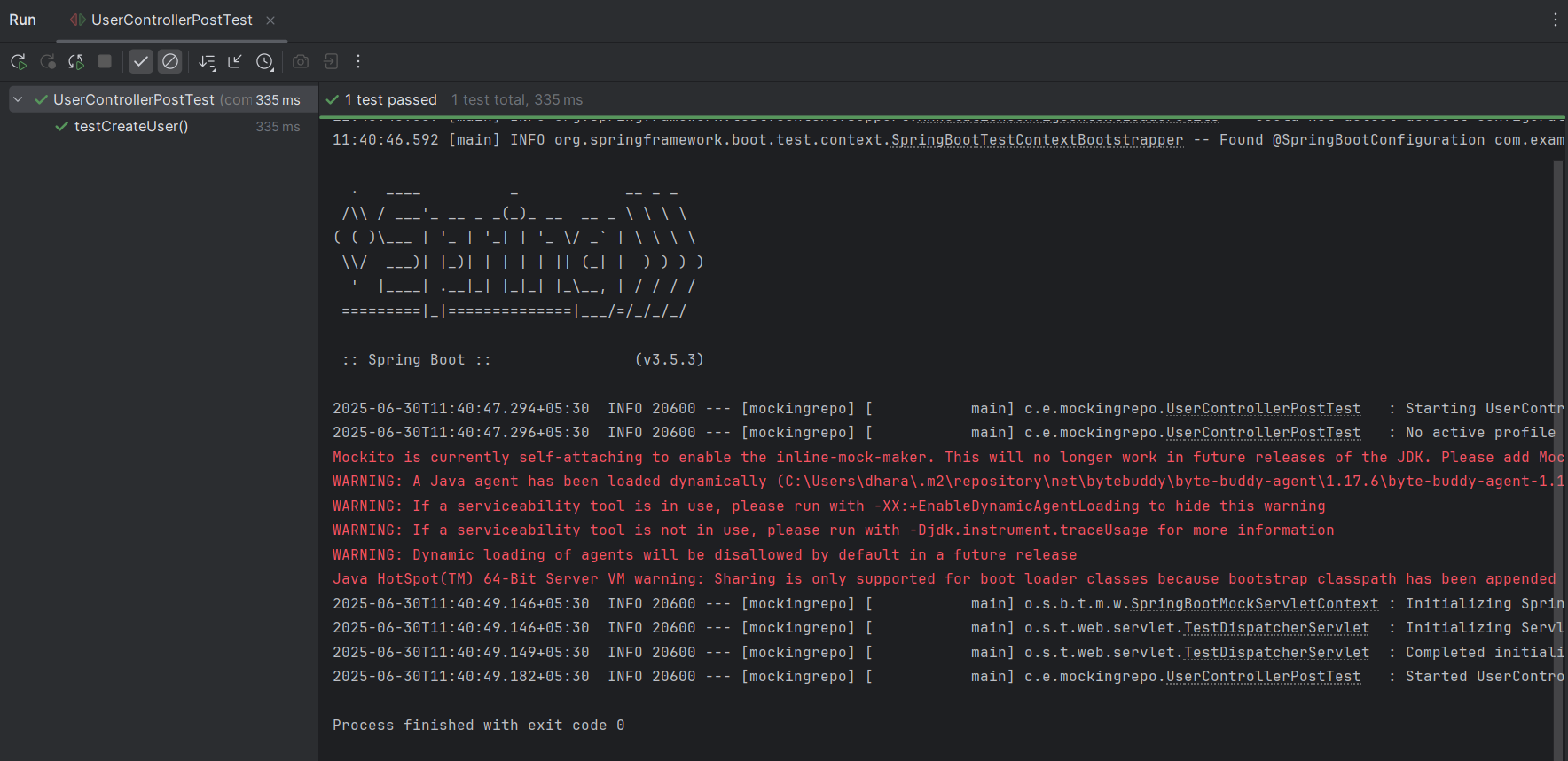
.andExpect(jsonPath("$.id").value(1L))

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

}

}

**Output:**



**Exercise 6: Test Service Exception Handling**

**// User.java**

package com.example.demo.model;

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; }

}

**// UserRepository.java**

package com.example.demo.repository;

import com.example.demo.model.User;

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

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

**// UserService.java**

package com.example.demo.service;

import com.example.demo.model.User;

import com.example.demo.repository.UserRepository;

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 with id: " + id));

}

public User saveUser(User user) {

return userRepository.save(user);

}

}

**// 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) {

User savedUser = userService.saveUser(user);

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

}

}

**// UserServiceTest.java**

package com.example.demo;

import com.example.demo.model.User;

import com.example.demo.repository.UserRepository;

import com.example.demo.service.UserService;

import org.junit.jupiter.api.Test;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import java.util.Optional;

import java.util.NoSuchElementException;

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

import static org.mockito.Mockito.\*;

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

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

import org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;

import org.springframework.boot.test.autoconfigure.jdbc.AutoConfigureTestDatabase;

@SpringBootTest

public class UserServiceTest {

@Mock

private UserRepository userRepository;

@InjectMocks

private UserService userService;

@Test

void testGetUserById\_NotFound\_ShouldThrowException() {

// Arrange

Long userId = 1L;

when(userRepository.findById(userId)).thenReturn(Optional.empty());

// Act & Assert

NoSuchElementException exception = assertThrows(NoSuchElementException.class,

() -> userService.getUserById(userId));

assertEquals("User not found with id: 1", exception.getMessage());

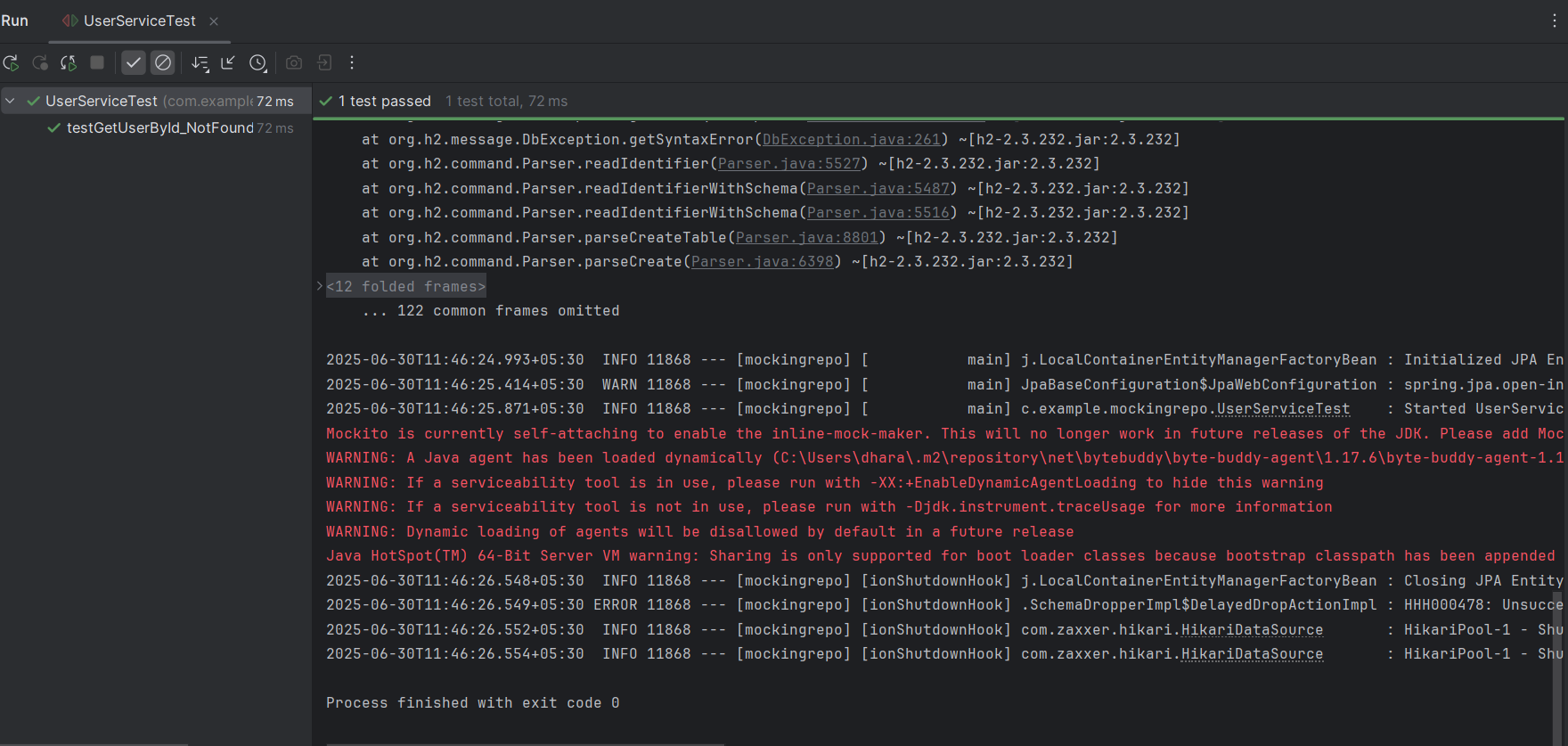
// Verify interaction

verify(userRepository).findById(userId);

}

}

**Output:**



**Exercise 7: Test Custom Repository Query**

**// User.java**

package com.example.mockingrepo;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

@Entity

@Table(name = "users")

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.mockingrepo;

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

import java.util.List;

public interface UserRepository extends JpaRepository<User, Long> {

List<User> findByName(String name); // Custom query method

}

**// UserRepositoryTest.java**

package com.example.mockingrepo;

import org.junit.jupiter.api.Test;

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

import org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;

import java.util.List;

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

@DataJpaTest

public class UserRepositoryTest {

@Autowired

private UserRepository userRepository;

@Test

public void testFindByName() {

// Arrange - save users with different names

userRepository.save(new User(1L, "Alice"));

userRepository.save(new User(2L, "Bob"));

userRepository.save(new User(3L, "Alice"));

// Act - query by name

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

// Assert - should find 2 users named Alice

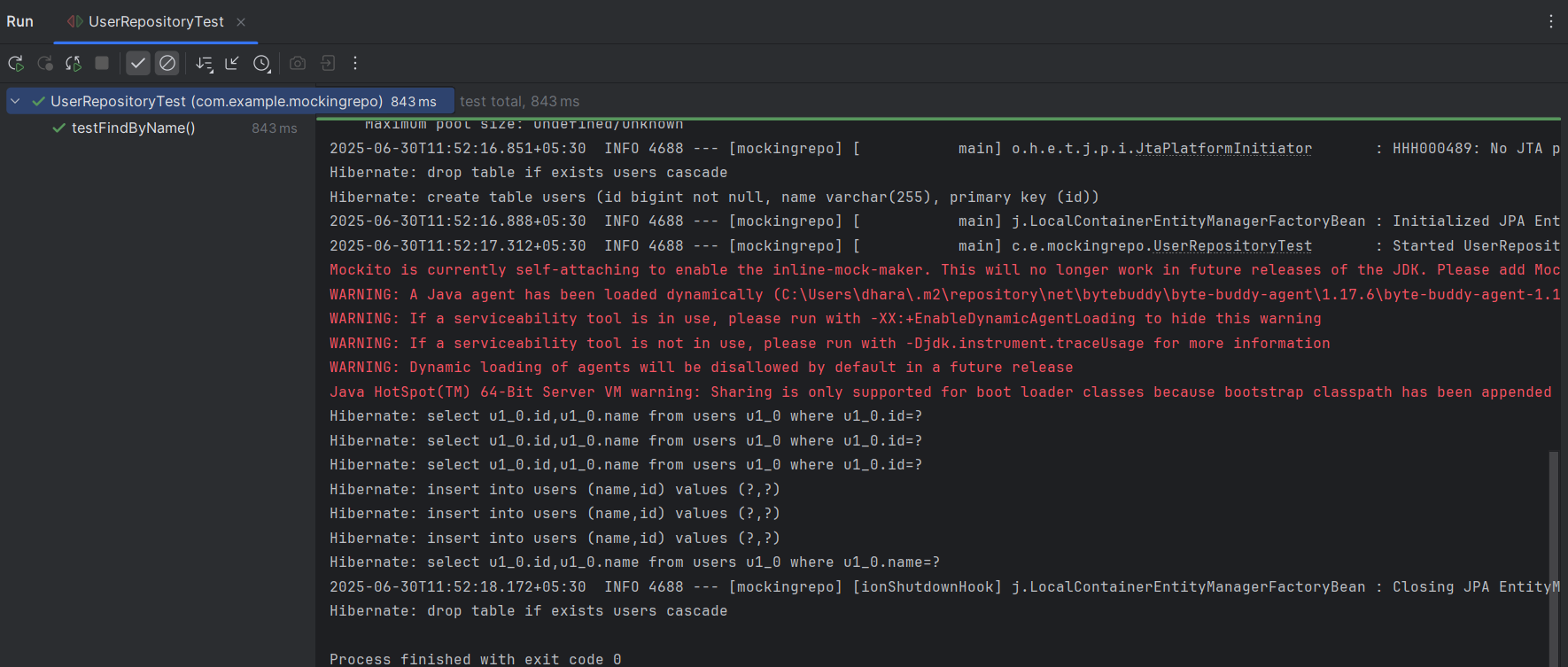
assertThat(usersNamedAlice).hasSize(2);

assertThat(usersNamedAlice).allMatch(user -> user.getName().equals("Alice"));

}

}

**Output:**



**Exercise 8: Test Controller Exception Handling**

**// User.java**

package com.example.mockingrepo;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@Entity

@Table(name = "users")

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; }

}

**// UserService.java**

package com.example.mockingrepo;

import org.springframework.stereotype.Service;

import java.util.NoSuchElementException;

@Service

public class UserService {

public User getUserById(Long id) {

// Simulate user not found

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

}

}

**// UserController.java**

package com.example.mockingrepo;

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) {

User user = userService.getUserById(id);

return ResponseEntity.ok(user);

}

}

**// GlobalExceptionHandler.java**

package com.example.mockingrepo;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

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");

}

}

**// UserControllerExceptionTest.java**

package com.example.mockingrepo;

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 java.util.NoSuchElementException;

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 UserControllerExceptionTest {

@Autowired

private MockMvc mockMvc;

@MockBean

private UserService userService;

@Test

void testUserNotFoundExceptionHandling() throws Exception {

Long userId = 1L;

when(userService.getUserById(userId))

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

mockMvc.perform(get("/users/{id}", userId))

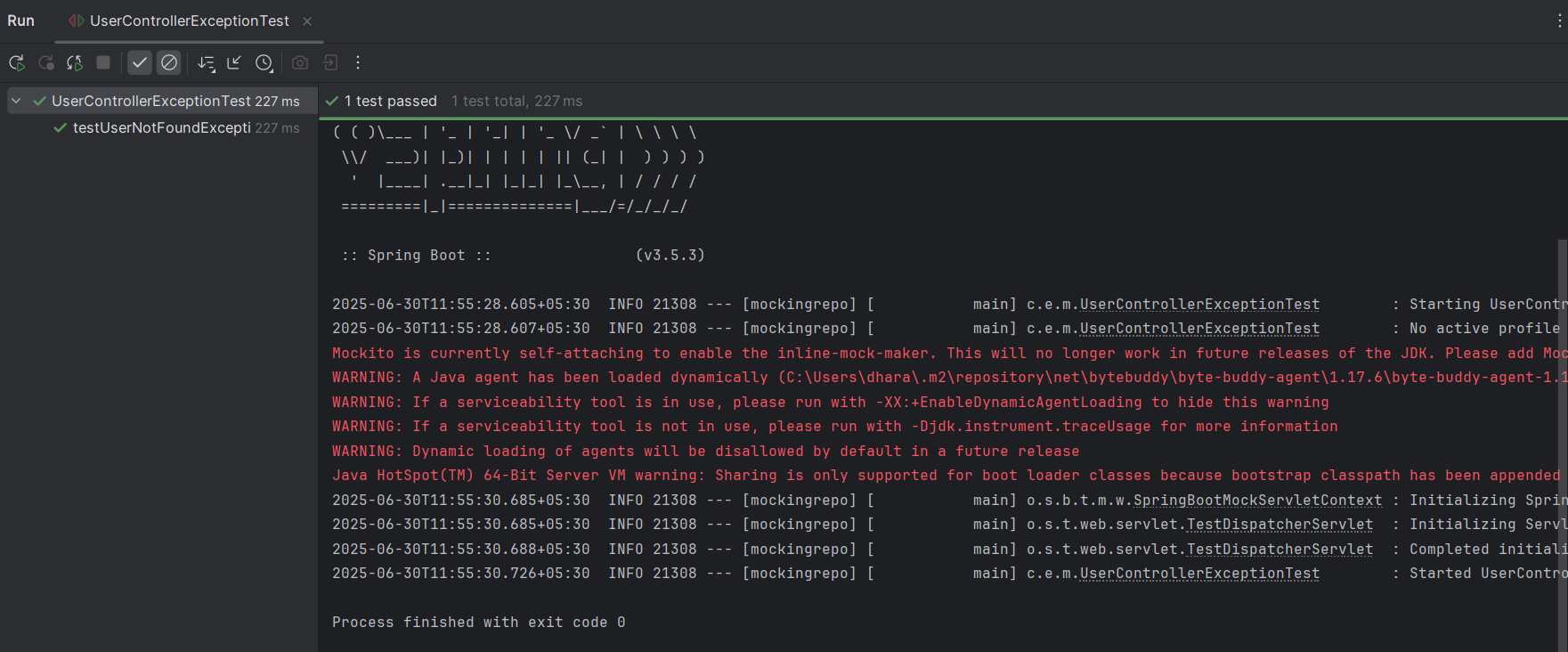
.andExpect(status().isNotFound())

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

}

}

**Output:**



**Exercise 9: Parameterized Test with JUnit**

**// UserUtils.java**

package com.example.mockingrepo;

public class UserUtils {

public static boolean isValidName(String name) {

return name != null && !name.trim().isEmpty() && name.matches("^[A-Za-z ]+$");

}

}

**// UserUtilsTest.java**

package com.example.mockingrepo;

import org.junit.jupiter.params.ParameterizedTest;

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

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

public class UserUtilsTest {

@ParameterizedTest

@CsvSource({

"'John', true",

"' ', false",

"'123', false",

"'Alice Smith', true",

"'', false",

"'@dmin', false"

})

void testIsValidName(String input, boolean expected) {

boolean actual = UserUtils.isValidName(input);

assertEquals(expected, actual, "Name validation failed for input: " + input);

}

}

**Output:**

