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

**@class**

**CalculatorService.java:**

public class CalculatorService {

public int add(int a, int b) {

return a + b;

}

}

**CalculatorServiceTest.java:**

import org.junit.jupiter.api.Test;

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

public class CalculatorServiceTest {

@Test

public void testAdd() {

CalculatorService calculatorService = new CalculatorService();

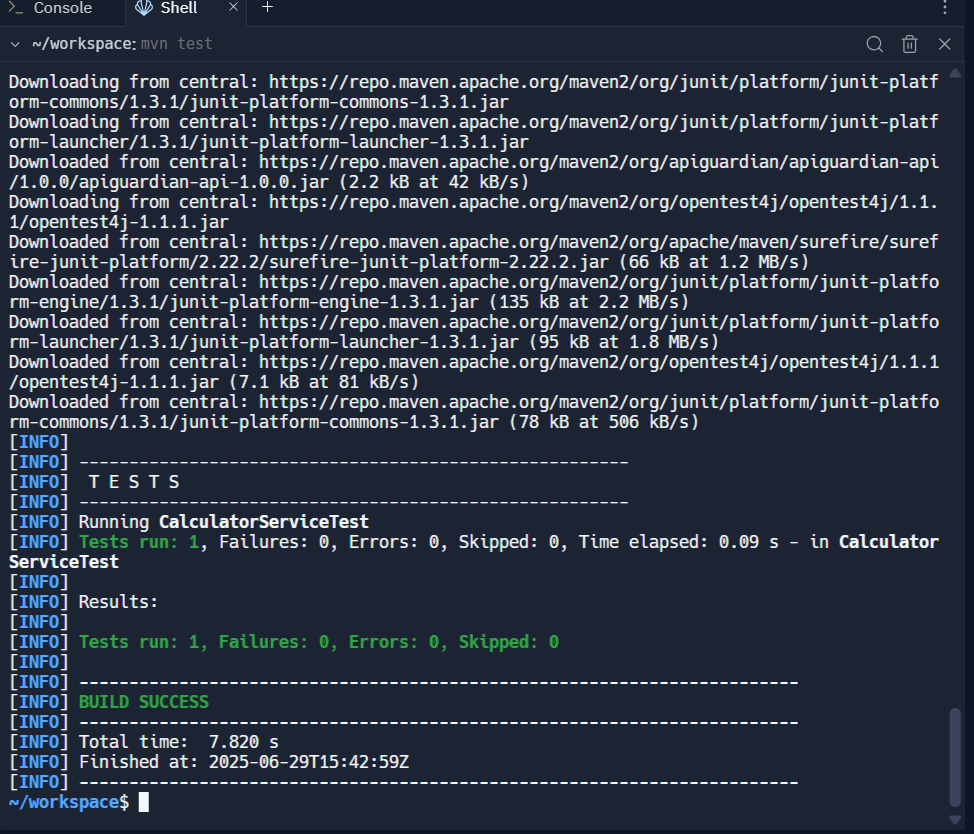
int result = calculatorService.add(3, 4);

assertEquals(7, result);

}

}

**OUTPUT:**

****

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

**@class**

**User.java:**

import javax.persistence.Entity;

import javax.persistence.Id;

@Entity

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; }

}

**UserRepository.java:**

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

public interface UserRepository extends JpaRepository<User, Long> {

}

**UserService.java:**

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 org.junit.jupiter.api.Test;

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

import static org.mockito.Mockito.\*;

import java.util.Optional;

public class UserServiceTest {

@Test

public void testGetUserById\_ReturnsUser() {

UserRepository mockRepository = mock(UserRepository.class);

UserService userService = new UserService();

User expectedUser = new User(1L, "Alice");

java.lang.reflect.Field field;

try {

field = UserService.class.getDeclaredField("userRepository");

field.setAccessible(true);

field.set(userService, mockRepository);

} catch (Exception e) {

fail("Failed to inject mock repository");

}

when(mockRepository.findById(1L)).thenReturn(Optional.of(expectedUser));

User actualUser = userService.getUserById(1L);

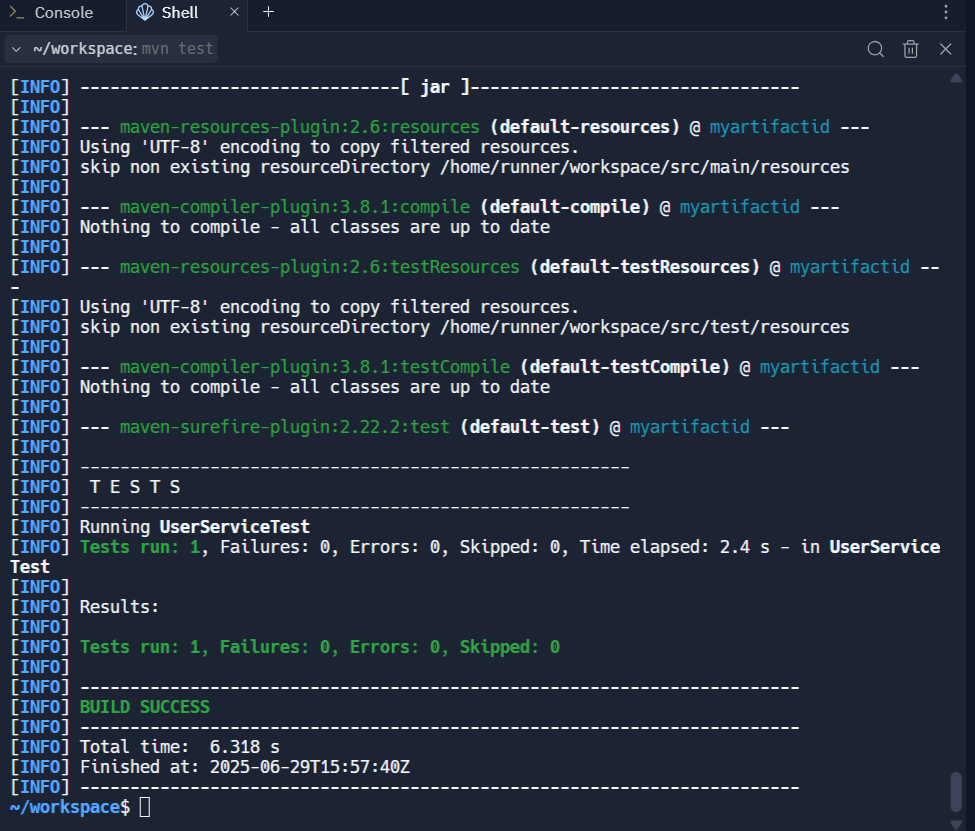
assertNotNull(actualUser);

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

}

}

**OUTPUT:**



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

**@class**

**User.java:**

public class User {

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;

}

}

**UserController.java:**

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);

}

}

**UserService.java:**

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);

}

}

**UserControllerTest.java:**

import com.fasterxml.jackson.databind.ObjectMapper;

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.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.ArgumentMatchers.anyLong;

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;

@Autowired

private ObjectMapper objectMapper;

@Test

public void testGetUserById() throws Exception {

User mockUser = new User(1L, "John Doe");

Mockito.when(userService.getUserById(anyLong())).thenReturn(mockUser);

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

.contentType(MediaType.APPLICATION\_JSON))

.andExpect(status().isOk())

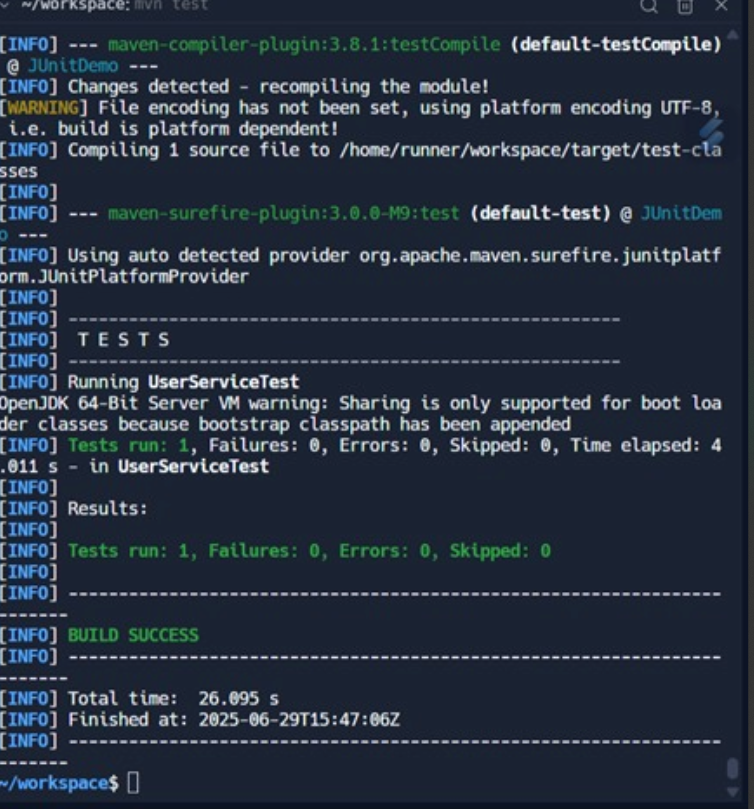
.andExpect(jsonPath("$.id").value(1))

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

}

}

**OUTPUT:**

****

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

**@class**

**UserIntegrationTest.java:**

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.autoconfigure.jdbc.AutoConfigureTestDatabase;

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

import org.springframework.boot.web.server.LocalServerPort;

import org.springframework.http.ResponseEntity;

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

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

@AutoConfigureTestDatabase(replace = Replace.ANY)

public class UserIntegrationTest {

@LocalServerPort

private int port;

@Autowired

private TestRestTemplate restTemplate;

@Autowired

private UserRepository userRepository;

@Test

public void testGetUserById() {

User user = new User();

user.setId(1L);

user.setName("John Doe");

userRepository.save(user);

String url = "http://localhost:" + port + "/users/1";

ResponseEntity<User> response = restTemplate.getForEntity(url, User.class);

assertEquals(200, response.getStatusCodeValue());

assertNotNull(response.getBody());

assertEquals("John Doe", response.getBody().getName());

}

}

**OUTPUT:**

