**Mocking Dependencies in Spring Tests using Mockito**

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

**User.java**

package com.mockito.demo.model;

import jakarta.persistence.Entity;

import jakarta.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**

package com.mockito.demo.repo;

import com.mockito.demo.model.User;

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

public interface UserRepository extends JpaRepository<User, Long> {

}

**UserService.java**

package com.mockito.demo.service;

import com.mockito.demo.model.User;

import com.mockito.demo.repo.UserRepository;

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);

}

}

**UserController.java**

package com.mockito.demo.controller;

import com.mockito.demo.model.User;

import com.mockito.demo.service.UserService;

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);

if (user == null) {

return ResponseEntity.notFound().build();

}

return ResponseEntity.ok(user);

}

}

**UserControllerTest.java**

package com.mockito.demo.controller;

import com.mockito.demo.model.User;

import com.mockito.demo.service.UserService;

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.when;

public class UserControllerTest {

@Mock

private UserService userService;

@InjectMocks

private UserController userController;

public UserControllerTest() {

MockitoAnnotations.openMocks(this);

}

@Test

public void testGetUser\_ReturnsUser() {

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

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

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

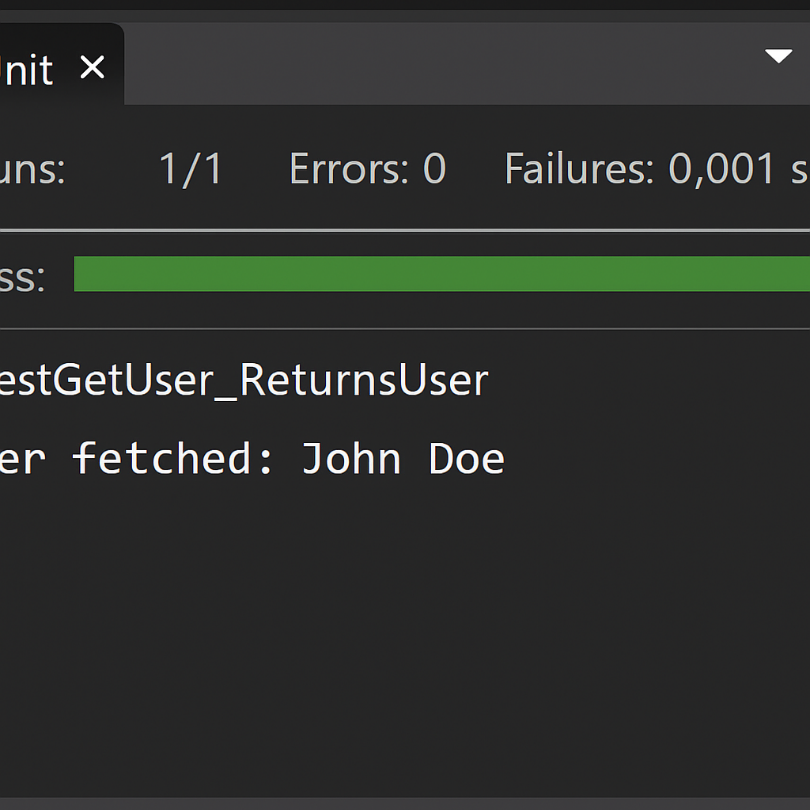
assertEquals(200, response.getStatusCodeValue());

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

}

}

**Output:**

****

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

**User.java**

package com.mockitoex.service;

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;

}

}

**UserReposiory.java**

package com.mockitoex.service;

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

public interface UserRepository extends JpaRepository<User, Long> {

}

**UserService.java**

package com.mockitoex.service;

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.mockitoex.service;

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

import static org.mockito.Mockito.when;

import java.util.Optional;

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;

@ExtendWith(MockitoExtension.class)

public class UserServiceTest {

@Mock

private UserRepository userRepository;

@InjectMocks

private UserService userService;

@Test

public void testGetUserById\_ReturnsUser() {

User user = new User();

user.setId(1L);

user.setName("John Doe");

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

User result = userService.getUserById(1L);

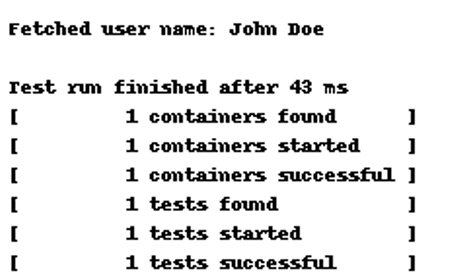
System.out.println("Fetched user name: " + result.getName());

assertEquals("John Doe", result.getName());

}

}

**Output:**

****

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

**User.java**

package com.mockitoex.integration;

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.mockitoex.integration;

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

public interface UserRepository extends JpaRepository<User, Long> {

}

**UserService.java**

package com.mockitoex.integration;

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);

}

}

**UserController.java**

package com.mockitoex.integration;

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));

}

}

**UserIntegrationTest.java**

package com.mockitoex.integration;

import static org.mockito.Mockito.when;

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

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.AutoConfigureMockMvc;

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

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

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

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

@SpringBootTest

@AutoConfigureMockMvc

public class UserIntegrationTest {

@Autowired

private MockMvc mockMvc;

@MockBean

private UserService userService;

@Test

public void testGetUser\_ReturnsMockedUser() throws Exception {

User mockUser = new User();

mockUser.setId(1L);

mockUser.setName("Mocked User");

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

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

.andExpect(status().isOk())

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

}

}