**PAKHI SHARMA**

Mocking Dependencies in Spring Tests

using Mockito

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

\*\*Task:\*\* Write a unit test for a Spring controller that uses a service to fetch data. Mock the service dependency using Mockito.

### Step-by-Step Solution:

1. \*\*Create the User Entity:\*\* @Entity

public class User { @Id

private Long id; private String name;

// getters and setters

}

1. \*\*Create the UserService:\*\* @Service

public class UserService { @Autowired

private UserRepository userRepository; public User getUserById(Long id) {

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

}

}

1. \*\*Create the UserController:\*\* @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));

}

}

1. \*\*Create the UserControllerTest:\*\* Write code for this.

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 testGetUserById() throws Exception {

User user = new User();

user.setId(1L);

user.setName("John");

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

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

.andExpect(status().isOk())

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

}

}

# Exercise 2: Mocking a Repository in a Service Test

\*\*Task:\*\* Write a unit test for a Spring service that uses a repository to fetch data. Mock the repository dependency using Mockito.

### Step-by-Step Solution:

1. \*\*Create the User Entity:\*\* @Entity

public class User { @Id

private Long id; private String name;

// getters and setters

}

1. \*\*Create the UserRepository:\*\*

public interface UserRepository extends JpaRepository<User, Long> {

}

1. \*\*Create the UserService:\*\* @Service

public class UserService { @Autowired

private UserRepository userRepository; public User getUserById(Long id) {

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

}

}

1. \*\*Create the UserServiceTest:\*\* Write code for this.

import org.junit.jupiter.api.Test;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.junit.jupiter.MockitoExtension;

import org.junit.jupiter.api.extension.ExtendWith;

import java.util.Optional;

import static org.mockito.Mockito.when;

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

@ExtendWith(MockitoExtension.class)

public class UserServiceTest {

@Mock

private UserRepository userRepository;

@InjectMocks

private UserService userService;

@Test

void testGetUserById() {

User user = new User();

user.setId(1L);

user.setName("Alice");

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

User result = userService.getUserById(1L);

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

}

}

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

\*\*Task:\*\* Write an integration test for a Spring Boot application that mocks a service dependency using Mockito.

### Step-by-Step Solution:

1. \*\*Create the User Entity:\*\* @Entity

public class User { @Id

private Long id; private String name;

// getters and setters

}

1. \*\*Create the UserService:\*\* @Service

public class UserService {

@Autowired

private UserRepository userRepository; public User getUserById(Long id) {

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

}

}

1. \*\*Create the UserController:\*\* @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));

}

}

1. \*\*Create the UserIntegrationTest:\*\* Write code for this.

Hint: Use ‘@SpringBootTest’, ‘@AutoConfigureMockMvc’

import org.junit.jupiter.api.Test;

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

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

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

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.\*;

@SpringBootTest

@AutoConfigureMockMvc

public class UserIntegrationTest {

@Autowired

private MockMvc mockMvc;

@MockBean

private UserService userService;

@Test

void testGetUserIntegration() throws Exception {

User user = new User();

user.setId(1L);

user.setName("David");

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

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

.andExpect(status().isOk())

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

}

}