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.

**Code:**

**Pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>mocking-demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>MockingDemo</name>

<description>Spring Boot Mockito Mock Test</description>

<packaging>jar</packaging>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.2.5</version>

<relativePath/>

</parent>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<!-- Web -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<!-- JPA -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<!-- H2 DB -->

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<!-- Test (Mockito, JUnit) -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**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;

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.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);

}

}

**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));

}

}

**MockingDemoApplication.java**

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@SpringBootApplication*

public class MockingDemoApplication {

public static void main(String[] args) {

SpringApplication.*run*(MockingDemoApplication.class, args);

}

}

**UserControllerTest.java**

package com.example.demo;

import org.junit.jupiter.api.Test;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.MockitoAnnotations;

import org.springframework.http.ResponseEntity;

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

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

import static org.mockito.Mockito.\*;

*@SpringBootTest*

public class UserControllerTest {

*@Mock*

private UserService userService;

*@InjectMocks*

private UserController userController;

public UserControllerTest() {

MockitoAnnotations.*openMocks*(this);

}

*@Test*

public void testGetUser() {

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());

}

}

**application.properties**

# H2 Database Configuration

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

# Hibernate settings

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.jpa.hibernate.ddl-auto=update

# Show SQL in console

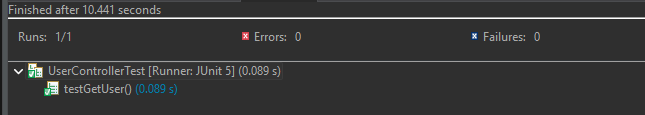
spring.jpa.show-sql=true

# H2 Console

spring.h2.console.enabled=true

spring.h2.console.path=/h2-console

**Output:**

****

# 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.

**Code:**

**Pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>MockingRepo\_In\_ServiceTest</artifactId>

<version>1.0</version>

<name>MockingRepo\_In\_ServiceTest</name>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.2.5</version>

<relativePath/>

</parent>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<!-- Web -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<!-- JPA -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<!-- H2 DB -->

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<!-- Test (Mockito, JUnit) -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**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;

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.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.*assertEquals*;

import static org.mockito.Mockito.*when*;

import java.util.Optional;

import org.junit.jupiter.api.BeforeEach;

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;

*@BeforeEach*

public void setUp() {

MockitoAnnotations.*openMocks*(this);

}

*@Test*

public void testGetUserById\_Found() {

// Arrange

Long userId = 1L;

User mockUser = new User(userId, "John Doe");

*when*(userRepository.findById(userId)).thenReturn(Optional.*of*(mockUser));

// Act

User result = userService.getUserById(userId);

// Assert

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

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

}

*@Test*

public void testGetUserById\_NotFound() {

// Arrange

Long userId = 2L;

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

// Act

User result = userService.getUserById(userId);

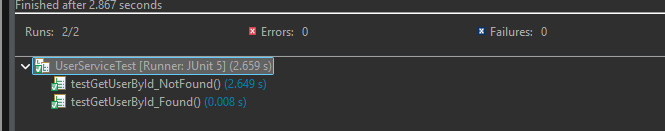
// Assert

*assertEquals*(null, result);

}

}

**Output:**

****

# 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’

**Code:**

**Pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>IntegrationTest\_Using\_Mockito</artifactId>

<version>1.0</version>

<name>IntegrationTest\_Using\_Mockito</name>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.2.5</version>

<relativePath/>

</parent>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<!-- Web -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<!-- JPA -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<!-- H2 DB -->

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<!-- Test (Mockito, JUnit) -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**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;

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.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);

}

}

**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));

}

}

**UserIntegrationTest.java**

package com.example.demo;

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 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.AutoConfigureMockMvc;

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

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

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

*@SpringBootTest*

*@AutoConfigureMockMvc*

public class UserIntegrationTest {

*@Autowired*

private MockMvc mockMvc;

*@MockBean*

private UserService userService; // Mocking the service

*@Test*

public void testGetUserById() throws Exception {

// Arrange

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

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

// Act & Assert

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

.andExpect(*status*().isOk())

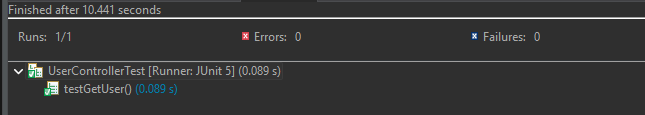
.andExpect(*jsonPath*("$.id").value(1))

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

}

}

**Output:**

****