

# **Alfido Tech Internship – Task 3**

Unit Testing & Logging in Spring Boot

Name: Revathi S

Role: Java Developer Intern

Task Number: Task 3

Organization: Alfido Tech

Technology: Java, Spring Boot

## **1. Task Overview**

The objective of Task 3 is to implement unit testing for service layer classes and configure logging in a Spring Boot application. This task focuses on improving code quality, validating business logic using test cases, and ensuring proper logging of important application events.

## **2. Technologies and Tools Used**

The following technologies and tools were used to complete this task:

Java

Spring Boot

JUnit 5

Mockito

Maven

SLF4J

Logback

Git & GitHub

Visual Studio Code

## **3. Project Description**

This Spring Boot project is based on an Employee Management system. The service layer handles business logic related to employee operations. Unit tests were written to validate service methods independently by mocking repository dependencies.

Flyway migrations were used to manage database schema creation and sample data insertion.

## **4. Unit Testing Implementation**

Unit testing was implemented for the service layer using JUnit 5 and Mockito.

Key Highlights:

Mockito is used to mock repository dependencies

JUnit annotations such as @Test and @ExtendWith are used

Service logic is tested independently without database interaction

```
EmployeeServiceImpl.java 3 EmployeeServiceImplTest.java X EmployeeController.java app  
employeeapi > src > test > java > com > example > employeeapi > service > EmployeeServiceImplTest.java  
1 package com.example.employeeapi.service;  
2  
3 import static org.junit.jupiter.api.Assertions.assertEquals;  
4 import static org.mockito.Mockito.verify;  
5 import static org.mockito.Mockito.when;  
6  
7 import java.util.Arrays;  
8 import java.util.List;  
9 import java.util.Optional;  
10  
11 import com.example.employeeapi.entity.Employee;  
12 import com.example.employeeapi.repository.EmployeeRepository;  
13  
14 import org.junit.jupiter.api.Test;  
15 import org.mockito.InjectMocks;  
16 import org.mockito.Mock;  
17 import org.springframework.boot.test.context.SpringBootTest;  
18  
19 @SpringBootTest  
D 20 public class EmployeeServiceImplTest {  
21  
22     @Mock  
23     private EmployeeRepository repository;  
24  
25     @InjectMocks  
26     private EmployeeServiceImpl employeeService;  
27  
28     @Test  
D 29     void testGetAllEmployees() {  
30         Employee emp1 = new Employee();  
31         emp1.setId(id: 1L);  
32         emp1.setName(name: "Revathi");  
33         emp1.setRole(role: "Developer");  
34  
35         Employee emp2 = new Employee();  
X  employeepi  main*  0  3  Java: Ready
```

```
EmployeeServiceImpl.java 3 EmployeeServiceImplTest.java X EmployeeController.java appl  
employeeapi > src > test > java > com > example > employeeapi > service > EmployeeServiceImplTest.java  
20 public class EmployeeServiceImplTest {  
21     void testGetAllEmployees() {  
22         when(repository.findAll()).thenReturn(Arrays.asList(emp1, emp2));  
23         List<Employee> result = employeeService.getAllEmployees();  
24         assertEquals(2, result.size());  
25     }  
26  
27     @Test  
28     void testGetEmployeeById() {  
29         Employee emp = new Employee();  
30         emp.setId(id: 1L);  
31         emp.setName(name: "Revathi");  
32         emp.setRole(role: "Developer");  
33  
34         when(repository.findById(1L)).thenReturn(Optional.of(emp));  
35  
36         Employee result = employeeService.getEmployeeById(id: 1L);  
37  
38         assertEquals("Revathi", result.getName());  
39     }  
40  
41     @Test  
42     void testCreateEmployee() {  
43         Employee emp = new Employee();  
44         emp.setName(name: "Revathi");  
45         emp.setRole(role: "Developer");  
46  
47         when(repository.save(emp)).thenReturn(emp);  
48  
49         Employee saved = employeeService.createEmployee(emp);  
50  
51         assertEquals("Developer", saved.getRole());  
52     }  
53  
54     @Test  
55     void testDeleteEmployee() {  
56         employeeService.deleteEmployee(id: 1L);  
57         verify(repository).deleteById(1L);  
58     }  
59 }]
```

employeeapi 96 main\* ① ② ③ Java: Ready

## 5. Logging Configuration

Logging was configured using SLF4J with Logback to capture important application events.

Logging Features:

Informational logs for service operations

Error logs for exception handling

Console logging configured via Logback

Sample Logback Configuration:

Xml

```
<root level="INFO">
    <appender-ref ref="CONSOLE"/>
</root>
```

This ensures better monitoring and debugging of application behavior.

## 6. Database Migration

Flyway was used for database migration and version control.

Migration Files:

V1\_CreateEmployeeTable.sql – Creates the employee table

V2\_InsertSampleData.sql – Inserts sample employee data

## 7. Test Execution and Results

Unit tests were executed using Maven.

Command Used:

mvn test

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS cmd + ⌘ ⌂ ⌄ ⌅

```
C:\Projects\employeeapi\employeeapi>mvnw test
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.example:employeeapi >-----
[INFO] Building employeeapi 0.0.1-SNAPSHOT
[INFO]   from pom.xml
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ employeeapi ---
[INFO] Copying 1 resource from src\main\resources to target\classes
[INFO] Copying 2 resources from src\main\resources to target\classes
[INFO]
[INFO] --- compiler:3.11.0:compile (default-compile) @ employeeapi ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- resources:3.3.1:testResources (default-testResources) @
employeeapi ---
[INFO] skip non existing resourceDirectory C:\Projects\employeeapi\employeeapi\src\test\resources
[INFO]
[INFO] --- compiler:3.11.0:testCompile (default-testCompile) @ employeeapi ---
[INFO] Changes detected - recompiling the module! :input tree
> employeeapi main* ⌂ ⑧ 0 △ 3 Java: Ready Ln 79, Col 2 Spaces: 4 UTR
```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS cmd + ⌘ ⌂ ⌄ ⌅

```
HHH000026: Second-level cache disabled
2026-01-14T16:12:54.390+05:30 INFO 32272 --- [employeeapi] [           main] o.s.o.j.p.SpringPersistenceUnitInfo      :
No LoadTimeWeaver setup: ignoring JPA class transformer
2026-01-14T16:12:56.484+05:30 INFO 32272 --- [employeeapi] [           main] o.h.e.t.j.p.i.JtaPlatformInitiator      :
HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable JTA platform integration)
2026-01-14T16:12:56.491+05:30 INFO 32272 --- [employeeapi] [           main] j.LocalContainerEntityManagerFactoryBean :
Initialized JPA EntityManagerFactory for persistence unit 'default'
2026-01-14T16:12:57.505+05:30 WARN 32272 --- [employeeapi] [           main] JpaBaseConfiguration$JpaWebConfiguration :
spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2026-01-14T16:12:58.653+05:30 INFO 32272 --- [employeeapi] [           main] c.e.e.service.EmployeeServiceImplTest      :
Started EmployeeServiceImplTest in 11.252 seconds (process running for 14.339)
Java HotSpot(TM) 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been appended
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 14.86 s -- in com.example.employeeapi.service.EmployeeServiceImplTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 24.856 s
[INFO] Finished at: 2026-01-14T16:13:01+05:30
[INFO] -----
```

> employeeapi main\* ⌂ ⑧ 0 △ 3 Java: Ready Ln 79, Col 2 Spaces: 4 UTR

Terminal output showing successful test execution.

## 8. GitHub Repository Link

The complete source code for Task 3 is available at the following GitHub repository:

 GitHub Link:

<https://github.com/Revathi2006-op/Alfido-Tech-Internship-Task-3>

## 9. Conclusion

Through Task 3, I gained hands-on experience in:

Writing unit tests using JUnit and Mockito

Mocking dependencies for isolated testing

Configuring logging using SLF4J and Logback

Managing database migrations using Flyway

Improving application reliability and maintainability

This task strengthened my understanding of professional Spring Boot development practices.