Software Quality and Management

# Lab 2: Implementing and Testing Web Application and API Service using Apache Maven and Spring Boot

Wali Mohammed

100865624

CRN: 75766
Group: 1
Date: 02/11/2025

Introduction:

This lab extends upon our experience with Apache Maven by implementing a web application and an API service using Spring Boot. The objectives of this lab include:

1. Building a web application using Spring Boot and Maven.
2. Implementing API endpoints for handling binary operations.
3. Adding unit tests for validating API and web application functionalities.

The primary focus is on implementing a web-based binary calculator and enhancing its capabilities by adding additional test cases and supporting more operations.

Source Code Design:

Binary Class

The Binary class represents binary numbers as strings and performs arithmetic and logical operations.

Key Methods

- Constructor:
    - Validates the binary string input.
    - Removes leading zeros.
    - Defaults invalid inputs to "0".
- add()
    - Implements binary addition using the carry method.
- multiply()
    - Implements binary multiplication using iterative shifting and addition.
- AND()
    - Implements bitwise AND operation.
- OR()
    - Implements bitwise OR operation.

Web Application (BinaryController.java)

The BinaryController class handles HTTP requests for the Binary Calculator Web Application.

- GET / → Loads the calculator view.

- POST / → Computes the result and displays the output.

When a user inputs binary numbers and an operator, the application processes the request and returns the result or an error message.

API Service (BinaryAPIController.java)

The API Controller handles API-based binary calculations.
It provides the following endpoints:

| Endpoint | Operation | Response Type |
|---|---|---|
| /add?operand1=101&operand2=110 | Binary Addition | String ("1011") |
| /add_json?operand1=101&operand2=110 | Binary Addition | JSON ({"result": "1011"}) |
| /multiply?operand1=101&operand2=11 | Binary Multiplication | String ("1111") |
| /multiply_json?operand1=101&operand2=11 | Binary Multiplication | JSON ({"result": "1111"}) |

---

Testing

Test Cases Summary

| Test Case | Target (Class/Function) | Purpose |
|---|---|---|
| Existing Test Cases | | |
| Test case 1 | BinaryController.getCalculator() | Ensures GET request loads calculator view. |

| | | |
|---|---|---|
| Test case 2 | BinaryController.getResult() | Checks valid binary addition in web app. |
| Test case 3 | BinaryAPIController.add() | Verifies API returns correct addition response. |
| Newly Added Test Cases | | |
| Test case 4 | BinaryController.getResult() | Tests multiplication (*) in the web application. |
| Test case 5 | BinaryAPIController.multiply() | Ensures API returns correct multiplication response. |
| Test case 6 | BinaryAPIController.multiply() | Validates API handling of invalid binary inputs. |

Conclusion

This lab provided hands-on experience with Spring Boot and Maven in web application and API development.
Key takeaways include:

- Understanding Spring Boot Controllers for web and API services.
- Implementing binary operations in both web and API versions.
- Using JUnit for automated testing in Maven projects.

All test cases were successfully executed, verifying that the implemented binary operations work correctly.