**Spring REST using Spring Boot 3**

**EXERCISE: Create a Spring Web Project using Maven**

**Introduction:**

This hands-on exercise demonstrates the creation of a simple RESTful web service using the Spring Boot 3 framework with Maven. The purpose of this activity is to explore the capabilities of Spring Web for building Java-based backend APIs. The application created is a minimal Spring Boot REST service that responds with a greeting message when accessed through a web browser or HTTP client.

**Tools and Technologies Used**

Java– Version 17 (or 21)

Spring Boot – Version 3.5.3

Maven – For build and dependency management

Eclipse IDE – Integrated development environment

Spring Initializr – For scaffolding the Spring Boot structure

Web Browser – For testing the REST API

**Setup and Configuration**

The application structure was generated using \*\*Spring Initializr\*\*, a web-based utility that allows users to select dependencies and project metadata. The application was configured with the following parameters:

\* Group ID: `com.example`

\* Artifact ID: `springwebdemo`

\* Name: `springwebdemo`

\* Package Name: `com.example.springwebdemo`

\* Project Type: Maven

\* Packaging: Jar

\* Java Version: 17

\* Dependency Added: Spring Web

The generated `.zip` file was extracted and imported into Eclipse as an Existing Maven Project. Maven resolved all dependencies automatically, and the project structure was prepared for development.

**Implementation Details**

The entry point of the application is the `SpringwebdemoApplication` class, which is annotated with `@SpringBootApplication`. A controller class named `HelloController` was added to the same package to expose a REST endpoint.

**\*HelloController.java**

package com.example.springwebdemo;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class HelloController {

@GetMapping("/hello")

public String hello() {

return "Hello from Spring Boot!";

}

}

This controller handles HTTP GET requests to `/hello` and returns a plain text response.

**Server Configuration**

By default, Spring Boot runs on port 8080. However, in some cases, this port may already be occupied by another application. In this hands-on, a port conflict was encountered. To resolve it, the default port was changed to 8081 by adding the following configuration in the `application.properties` file:

server.port=8081

This configuration was placed under:

src/main/resources/application.properties

**Output Verification**

The application was run from within Eclipse, and the embedded Tomcat server started successfully on port 8081. The `/hello` endpoint was tested by navigating to:

<http://localhost:8081/hello>

The browser displayed the expected output:

Hello from Spring Boot!

This verified that the REST API was functional and responding as designed.

OUTPUT:

