SJSU - CMPE-272

Enterprise Software Platform

Lab Assignment 1 – Spring Boot and RESTfull Web Services

Prof. M. A. Rayes

**Objective**

To create a RESTful web service using Spring Boot. This assignment will introduce you to the basics of Spring Boot, including project setup, creating REST endpoints, and testing the service. This assignment provides a basic introduction to Spring Boot and RESTful web services, laying the foundation for more complex assignments in the future.

Spring Boot, <https://spring.io/projects/spring-boot>, is an open-source, microservice-based Java web framework offered by Spring, particularly useful for software engineers developing web apps and microservices**.**

## Features

* Create stand-alone Spring applications
* Embed Tomcat, Jetty or Undertow directly (no need to deploy WAR files)
* Provide opinionated 'starter' dependencies to simplify your build configuration
* Automatically configure Spring and 3rd party libraries whenever possible
* Provide production-ready features such as metrics, health checks, and externalized configuration
* Absolutely no code generation and no requirement for XML configuration

## Getting Started

* Super quick — [Quickstart Guide](https://spring.io/quickstart).
* More general — [Building an Application with Spring Boot](https://spring.io/guides/gs/spring-boot/)
* More specific — [Building a RESTful Web Service](https://spring.io/guides/gs/rest-service/).
* Or search through all our guides on the [Guides](https://spring.io/guides) homepage.

## Talks and videos

* [Mind the Gap: Jumping from Spring Boot 2.x to 3.x](https://www.youtube.com/watch?v=HrRQExD3xow)
* [Demystifying Spring Internals](https://www.youtube.com/watch?v=LeoCh7VK9cg)
* [Ahead Of Time and Native in Spring Boot 3.0](https://www.youtube.com/watch?v=TS4DpYSmfXk)
* [Improve Your Developer Experience with Spring Boot Dev Services](https://www.youtube.com/watch?v=Yqss7tYP890)

**Tools and Technologies:**

* Java (preferably version 11 or above)
* Spring Boot
* MAVEN or Gradle
* Postman (or any other REST client) for testing

**Assignment Details:**

**Part 1: Project Initialization**

1. **Project Initialization:**
   * Go to [Spring Initializr](https://start.spring.io/): <https://start.spring.io/>
   * Set the following configurations:
     + Project: MAVEN Project
     + Language: Java
     + Spring Boot: 2.7.0 (or the latest stable version)
     + Group: com.example
     + Artifact: simple-rest-service
     + Name: SimpleRestService
     + Package name: com.example.simplerestservice
     + Packaging: Jar
     + Java: 11 (or the version you have installed)
   * Add Dependencies:
     + Spring Web
   * Click on "Generate" to download the project, then unzip it.
2. **Import Project:**
   * Open your IDE (IntelliJ IDEA, Eclipse, etc.).
   * Import the unzipped project as a MAVEN project.

**Part 2: Creating the RESTful Web Service**

* Create a Model Class:
  + Create a package named model under src/main/java/com/example/simplerestservice.
  + Create a Java class named Greeting inside the model package.

package com.example.simplerestservice.model;

public class Greeting {

private final long id;

private final String content;

public Greeting(long id, String content) {

this.id = id;

this.content = content;

}

public long getId() {

return id;

}

public String getContent() {

return content;

}

}

* **Create a Controller:**
* Create a package named controller under src/main/java/com/example/simplerestservice.
* Create a Java class named GreetingController inside the controller package

package com.example.simplerestservice.controller;

import com.example.simplerestservice.model.Greeting;

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

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

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

import java.util.concurrent.atomic.AtomicLong;

@RestController

public class GreetingController {

private static final String template = "Hello, %s!";

private final AtomicLong counter = new AtomicLong();

@GetMapping("/greeting")

public Greeting greeting(@RequestParam(value = "name", defaultValue = "World") String name) {

return new Greeting(counter.incrementAndGet(), String.format(template, name));

}

}

**3. Main Application Class:**

* Ensure that the SimpleRestServiceApplication class is located in the src/main/java/com/example/simplerestservice package.

package com.example.simplerestservice;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SimpleRestServiceApplication {

public static void main(String[] args) {

SpringApplication.run(SimpleRestServiceApplication.class, args);

}

}

### **Part 3: Running and Testing the Application**

1. **Run the Application:**
   * Run the SimpleRestServiceApplication class.
   * The application will start on the default port 8080.
2. **Test the RESTful Web Service:**
   * Open Postman (or any other REST client).
   * Make a GET request to http://localhost:8080/greeting.
   * You should see a JSON response similar to:

**{**

**"id": 1,**

**"content": "Hello, World!"**

**}**

* Test with a query parameter: http://localhost:8080/greeting?name=John.
* You should see a JSON response similar to:

**{**

**"id": 2,**

**"content": "Hello, John!"**

**}**

### Deliverables:

1. Source code of the project.
2. Screenshot of the Postman request and response for both the default and custom name cases.
3. A README file with instructions on how to run the application.