**WEEK 04**

**Spring-rest-HandsOn**

1. **Create a Spring Web Project using Maven**

Solution:

**SME Walkthrough**

**src/main/java:**

* The main class SpringLearnApplication.java is inside the package: com.cognizant.springlearn
* The src/main/java folder contains the main source code of the Spring Boot application, including the entry point class and all functional components.

**src/main/resources:**

* application.properties – to define settings like server port, logging level, database URL, etc.
* The src/main/resources folder is used to store configuration and property files required for application setup.

**src/test/java:**

* It contains package : com.cognizant.springlearn
* Uses JUnit testing libraries
* The src/test/java folder contains test cases to validate the application logic and ensure code reliability.

**SpringLearnApplication.java:**

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

public static void main(String[] args) {

System.*out*.println("SpringLearnApplication main method started...");

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

}

}

**SpringLearnApplicationTests.java:**

package com.cognizant.spring\_learn;

import org.junit.jupiter.api.Test;

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

@SpringBootTest

class SpringLearnApplicationTests {

@Test

void contextLoads() {

}

}

**Dependencies on pom.xml:**

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

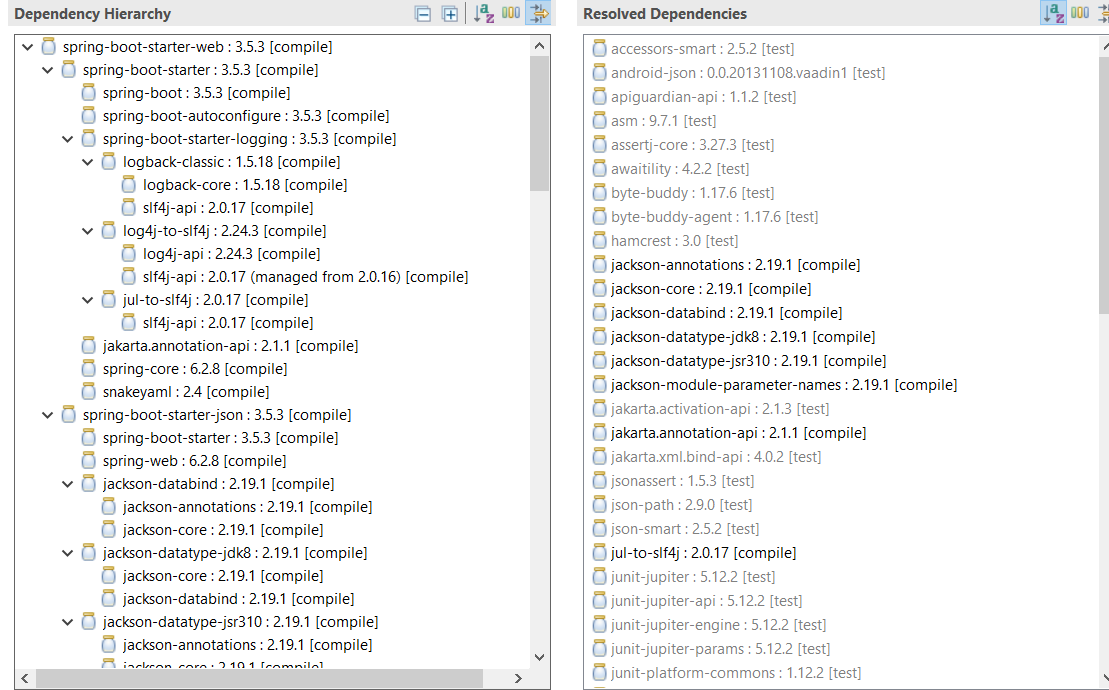
<artifactId>spring-boot-devtools</artifactId>

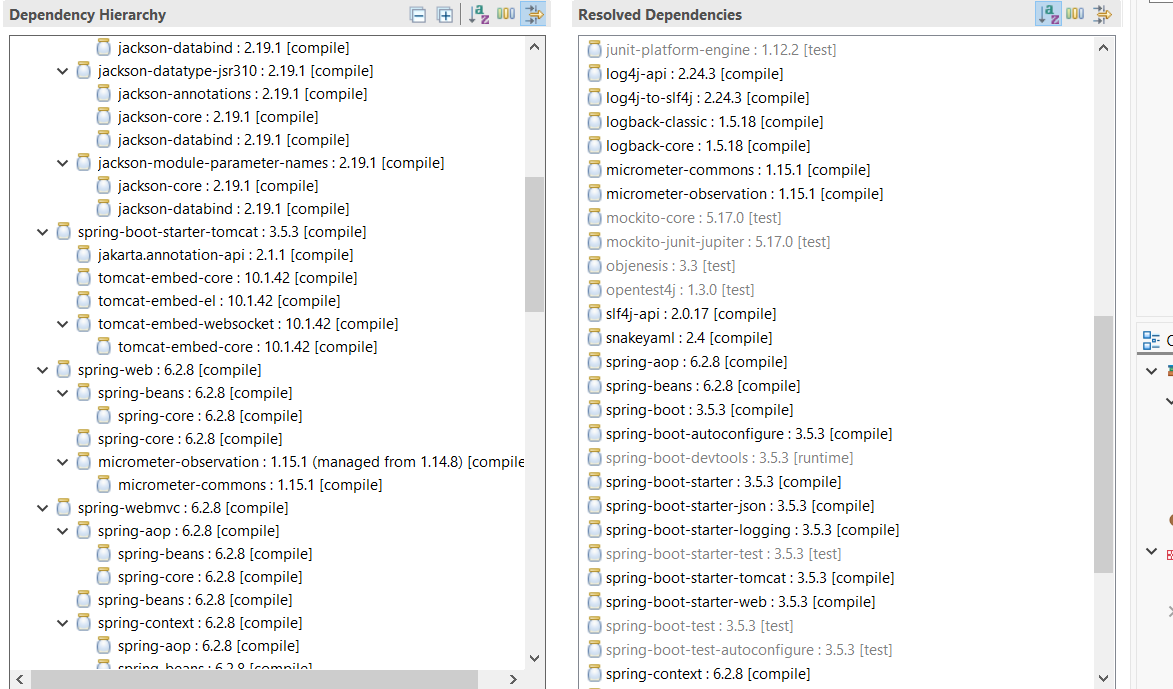
<scope>runtime</scope>

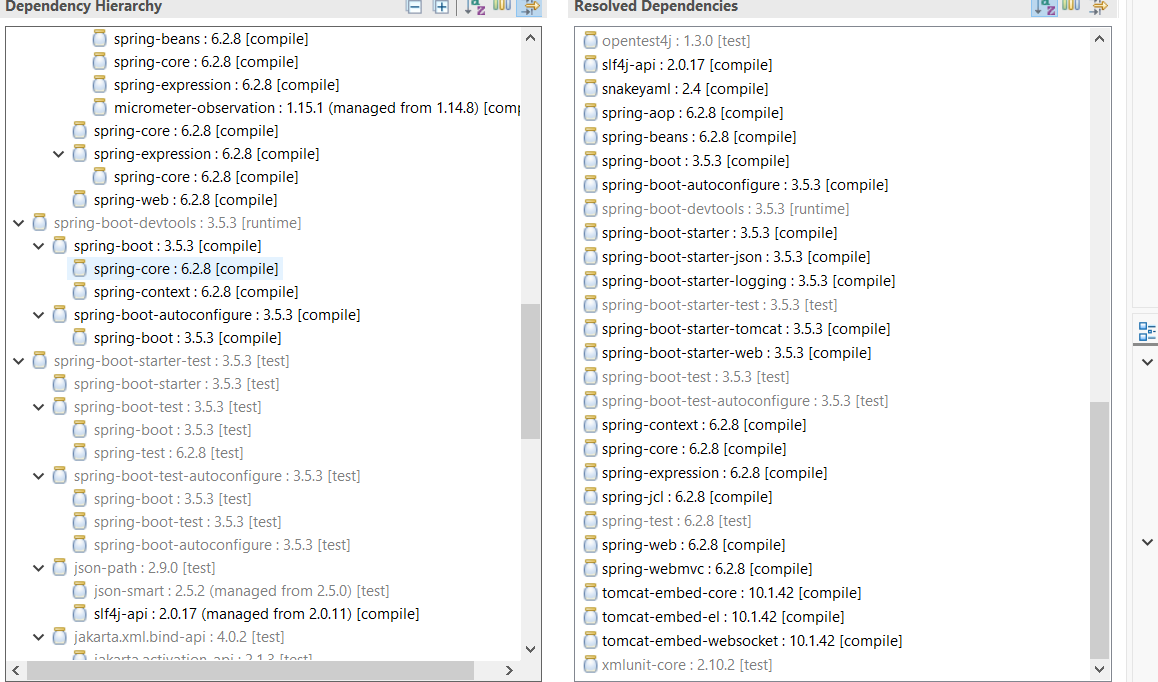
</dependency>

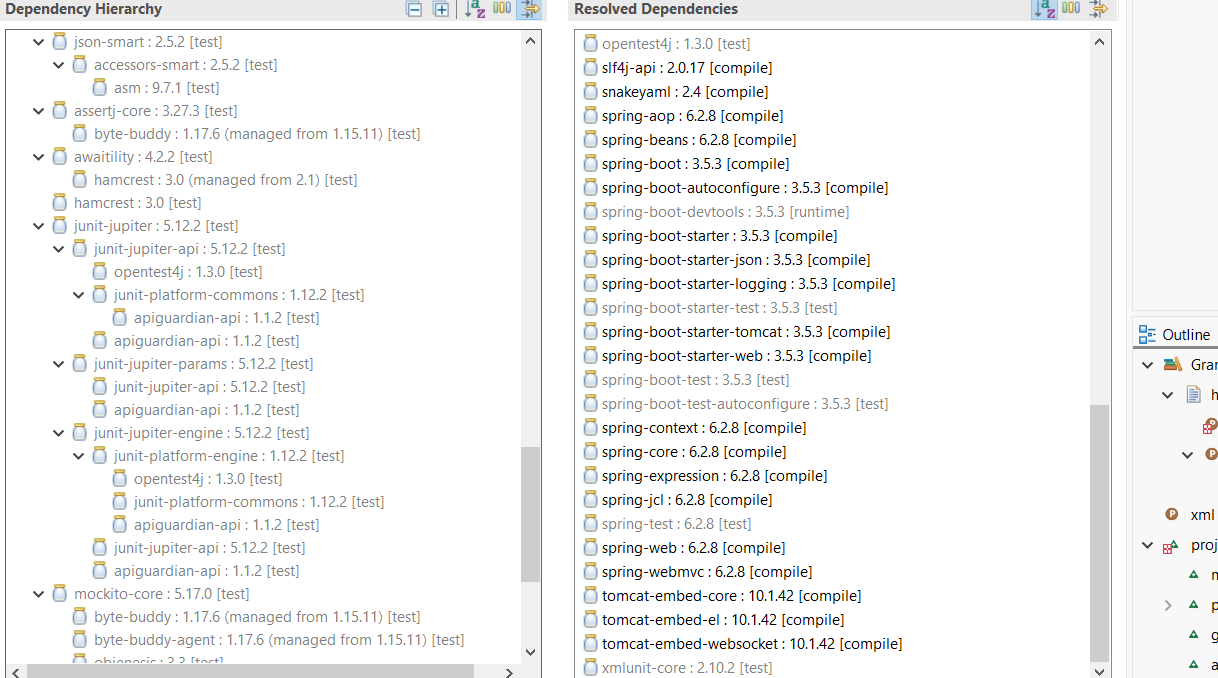
</dependencies>

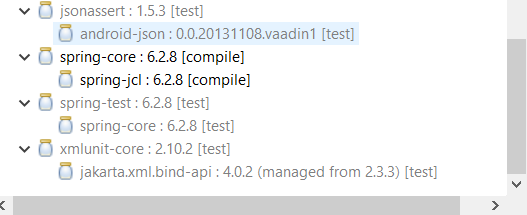
**Dependency Hierarchy:**

****

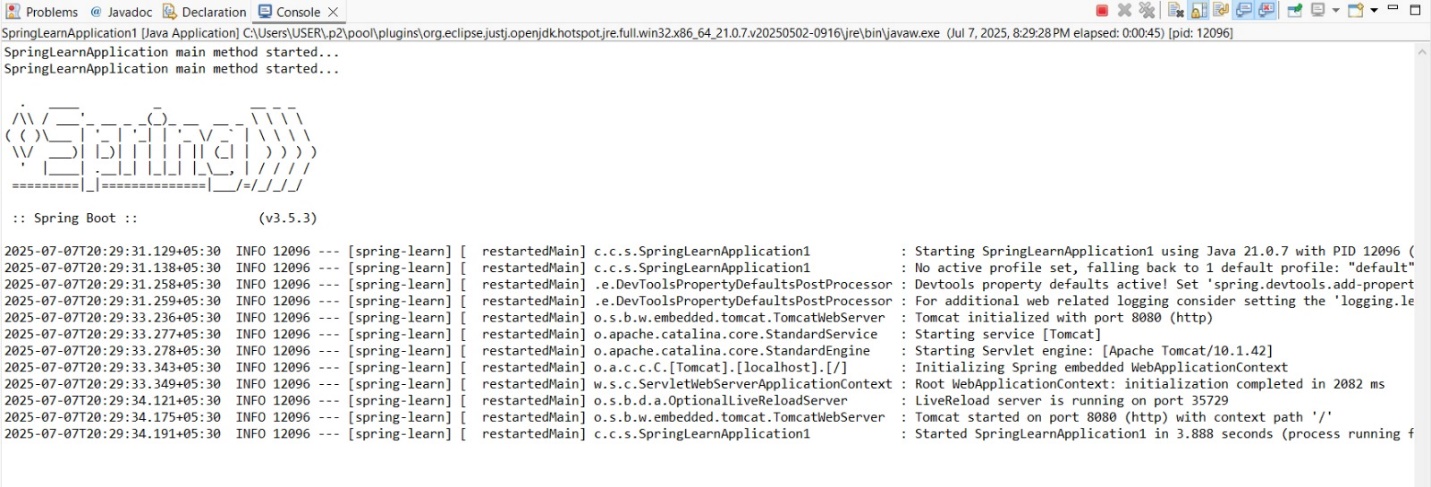
****

****

****

****

**Output:**

****

**Explanation:**

* Created a Spring Boot project named spring-learn using [start.spring.io](https://start.spring.io/). The project was configured with:
* Group: com.cognizant
* Dependencies: Spring Boot DevTools and Spring Web
* The downloaded ZIP was extracted into the Eclipse workspace. The project was built using the mvn clean package command with proxy settings and imported into Eclipse via File > Import > Maven > Existing Maven Projects.
* To verify, I have added logs inside the main() method of SpringLearnApplication.java and confirmed the application starts successfully.

**02. Spring Core – Load Country from Spring Configuration XML**

**Solution:**

**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.cognizant</groupId>

<artifactId>spring-core-xml</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-core-xml</name>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.32</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>5.3.32</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>5.3.32</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.36</version>

</dependency>

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>1.2.11</version>

</dependency>

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-core</artifactId>

<version>1.2.11</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>exec-maven-plugin</artifactId>

<version>3.1.0</version>

<executions>

<execution>

<goals>

<goal>java</goal>

</goals>

</execution>

</executions>

<configuration> <mainClass>com.cognizant.springlearn.SpringLearnApplication</mainClass>

</configuration>

</plugin>

</plugins>

</build>

</project>

**country.xml (File created in src/main/resources):**

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

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

xsi:schemaLocation="

http://www.springframework.org/schema/beans

https://www.springframework.org/schema/beans/spring-beans.xsd">

<!-- Single Country Bean -->

<bean id="country" class="com.cognizant.springlearn.Country">

<property name="code" value="IN"/>

<property name="name" value="India"/>

</bean>

<!-- List of Country Beans -->

<bean id="countryList" class="java.util.ArrayList">

<constructor-arg>

<list>

<ref bean="country"/>

<bean class="com.cognizant.springlearn.Country">

<property name="code" value="US"/>

<property name="name" value="United States"/>

</bean>

<bean class="com.cognizant.springlearn.Country">

<property name="code" value="DE"/>

<property name="name" value="Germany"/>

</bean>

<bean class="com.cognizant.springlearn.Country">

<property name="code" value="JP"/>

<property name="name" value="Japan"/>

</bean>

</list>

</constructor-arg>

</bean>

</beans>

**Country.java (Class in src/main/java):**

package com.cognizant.springlearn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class Country {

private static final Logger *LOGGER* = LoggerFactory.*getLogger*(Country.class);

private String code;

private String name;

public Country() {

*LOGGER*.debug("Inside Country Constructor.");

}

public String getCode() {

*LOGGER*.debug("Inside getCode()");

return code;

}

public void setCode(String code) {

*LOGGER*.debug("Inside setCode()");

this.code = code;

}

public String getName() {

*LOGGER*.debug("Inside getName()");

return name;

}

public void setName(String name) {

*LOGGER*.debug("Inside setName()");

this.name = name;

}

@Override

public String toString() {

return "Country [code=" + code + ", name=" + name + "]";

}

}

**SpringLearnApplication.java(Main class created in src/main/java):**

package com.cognizant.springlearn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import java.util.List;

public class SpringLearnApplication {

private static final Logger *LOGGER* = LoggerFactory.*getLogger*(SpringLearnApplication.class);

public static void main(String[] args) {

*LOGGER*.debug("START");

*displayCountries*();

*LOGGER*.debug("END");

}

public static void displayCountries() {

ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");

List<Country> countryList = (List<Country>) context.getBean("countryList");

for (Country country : countryList) {

*LOGGER*.debug("Country: {}", country);

}

}

}

**Output:**

****

**Explanation:**

* Created a Spring Core Maven project that loads a Country bean using XML configuration.
* The country data (code and name) is defined in country.xml using the <bean> tag. The Country class includes fields, constructor, getters, setters, and logs to track method calls.
* In SpringLearnApplication.java, I have used ApplicationContext and ClassPathXmlApplicationContext to load the Spring context and retrieve the bean using context.getBean().
* Logs confirmed successful instantiation and injection of properties.

**2.spring-rest-handson**

**03. Hello World RESTful Web Service**

**Solution:**

**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.cognizant.spring-learn</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>Spring Learn</name>

<parent>

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

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

<version>2.7.18</version> <!-- You can use the latest stable version -->

<relativePath/> <!-- Lookup from Maven Central -->

</parent>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

</dependency>

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

**SpringLearnApplication.java (Class created in src/main/java):**

package com.cognizant.springlearn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

public static void main(String[] args) {

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

}

}

**HelloController.java (Class in src/main/java):**

package com.cognizant.springlearn.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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

@RestController

public class HelloController {

private static final Logger *LOGGER* = LoggerFactory.*getLogger*(HelloController.class);

@GetMapping("/hello")

public String sayHello() {

*LOGGER*.info("START: sayHello()");

String message = "Hello World!!";

*LOGGER*.info("END: sayHello()");

return message;

}

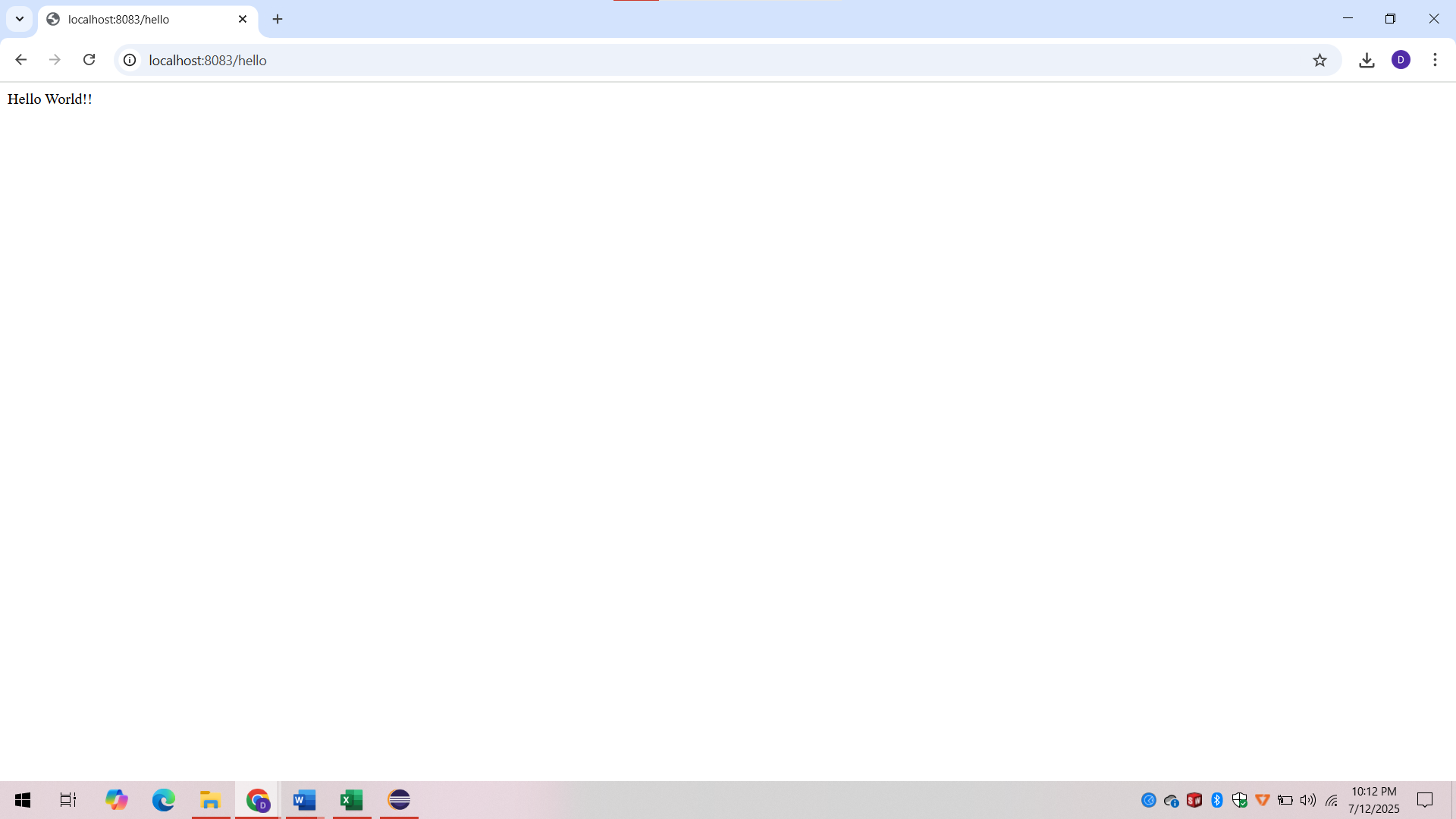
}

**Output:**

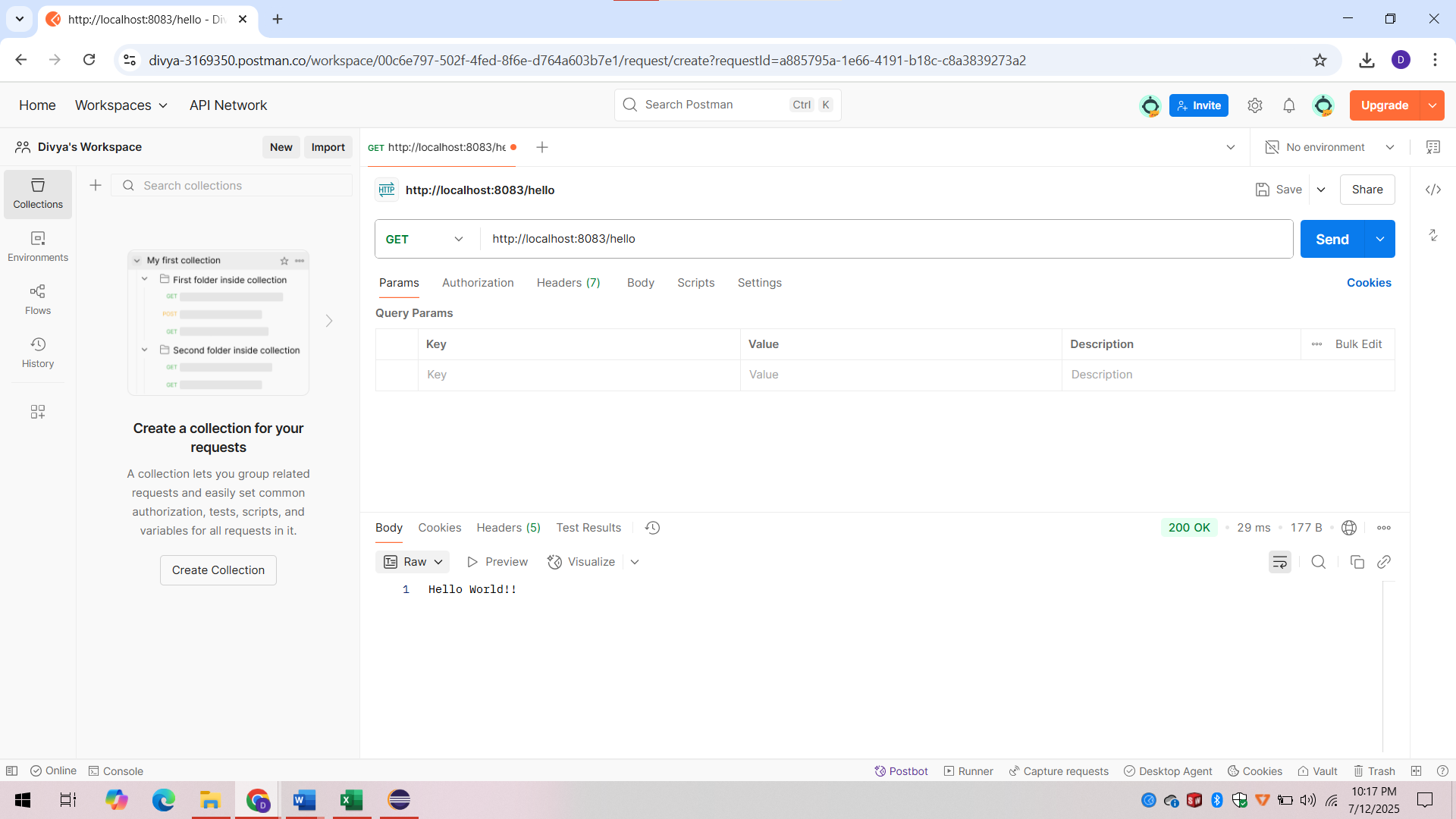
* Console output in eclipse

****

* **URL in Chrome browser**

****

* **URL in Postman**

****

**4. REST - Country Web Service**

**Solution:**

**Country1.java (Class created in src/main/java):**

package com.cognizant.springlearn.model;

public class Country1 {

private String code;

private String name;

public Country1() { }

public Country1(String code, String name) {

this.code = code;

this.name = name;

}

public String getCode() {

return code;

}

public void setCode(String code) {

this.code = code;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

@Override

public String toString() {

return "Country [code=" + code + ", name=" + name + "]";

}

}

**country.xml:**

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

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

xsi:schemaLocation="

http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="in" class="com.cognizant.springlearn.model.Country">

<property name="code" value="IN"/>

<property name="name" value="India"/>

</bean>

</beans>

**CountryController.java (Class created in src/main/java):**

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.model.Country;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

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

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

@RestController

public class CountryController {

private static final Logger LOGGER = LoggerFactory.getLogger(CountryController.class);

@RequestMapping("/country")

public Country getCountryIndia() {

LOGGER.info("START: getCountryIndia()");

ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");

Country country = (Country) context.getBean("in");

LOGGER.info("END: getCountryIndia()");

return country;

}

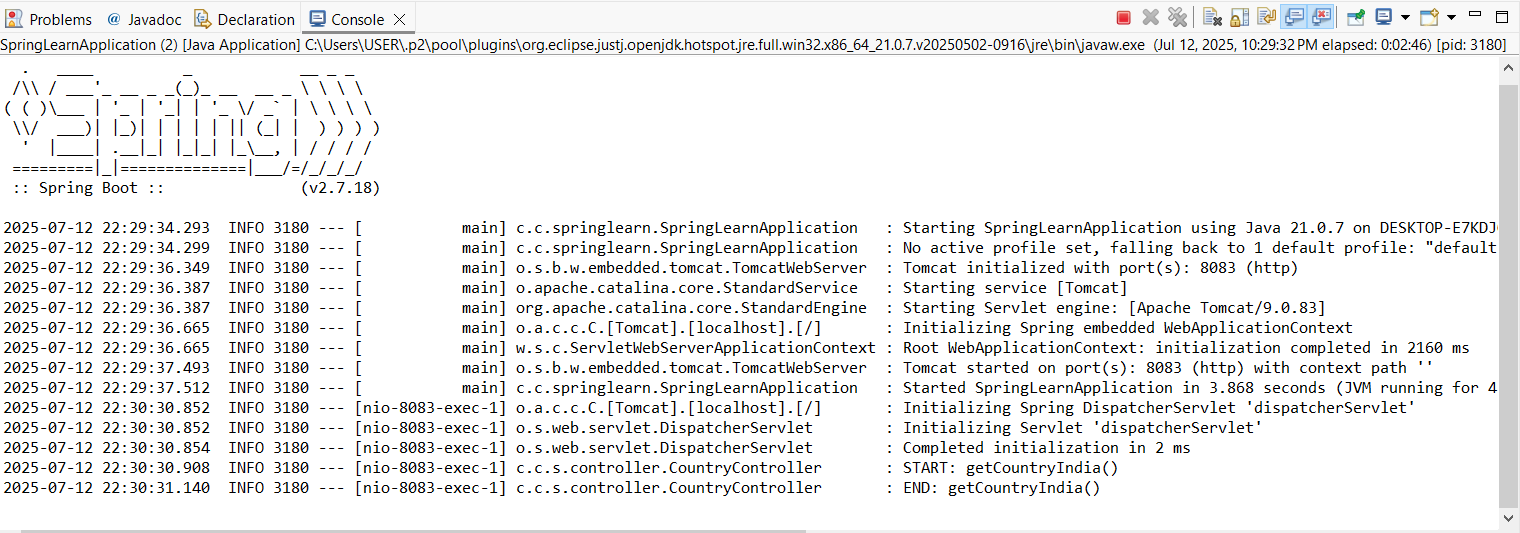
}

**application.properties (File created in src/main/resources):**

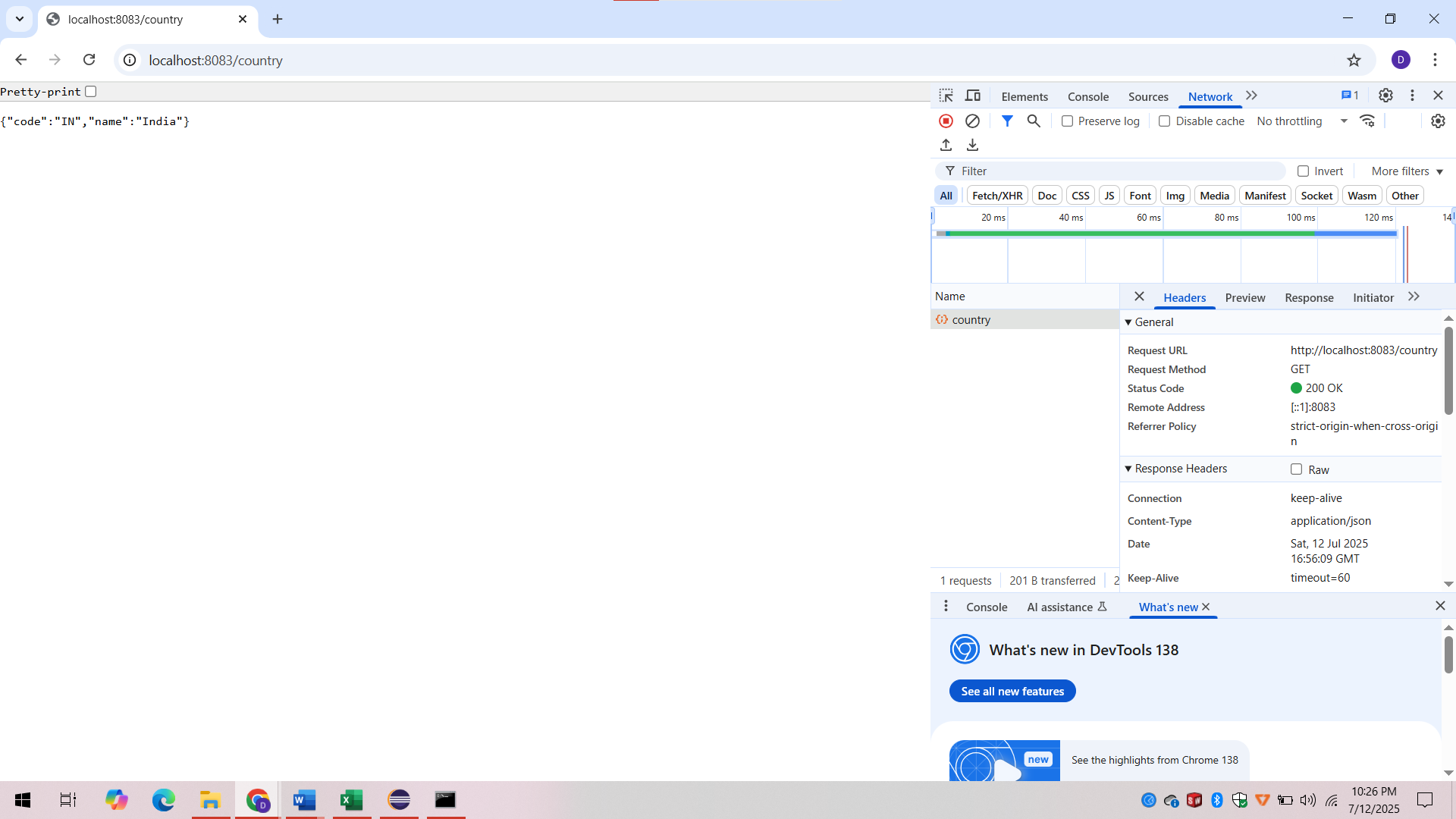
server.port=8083

**Output:**

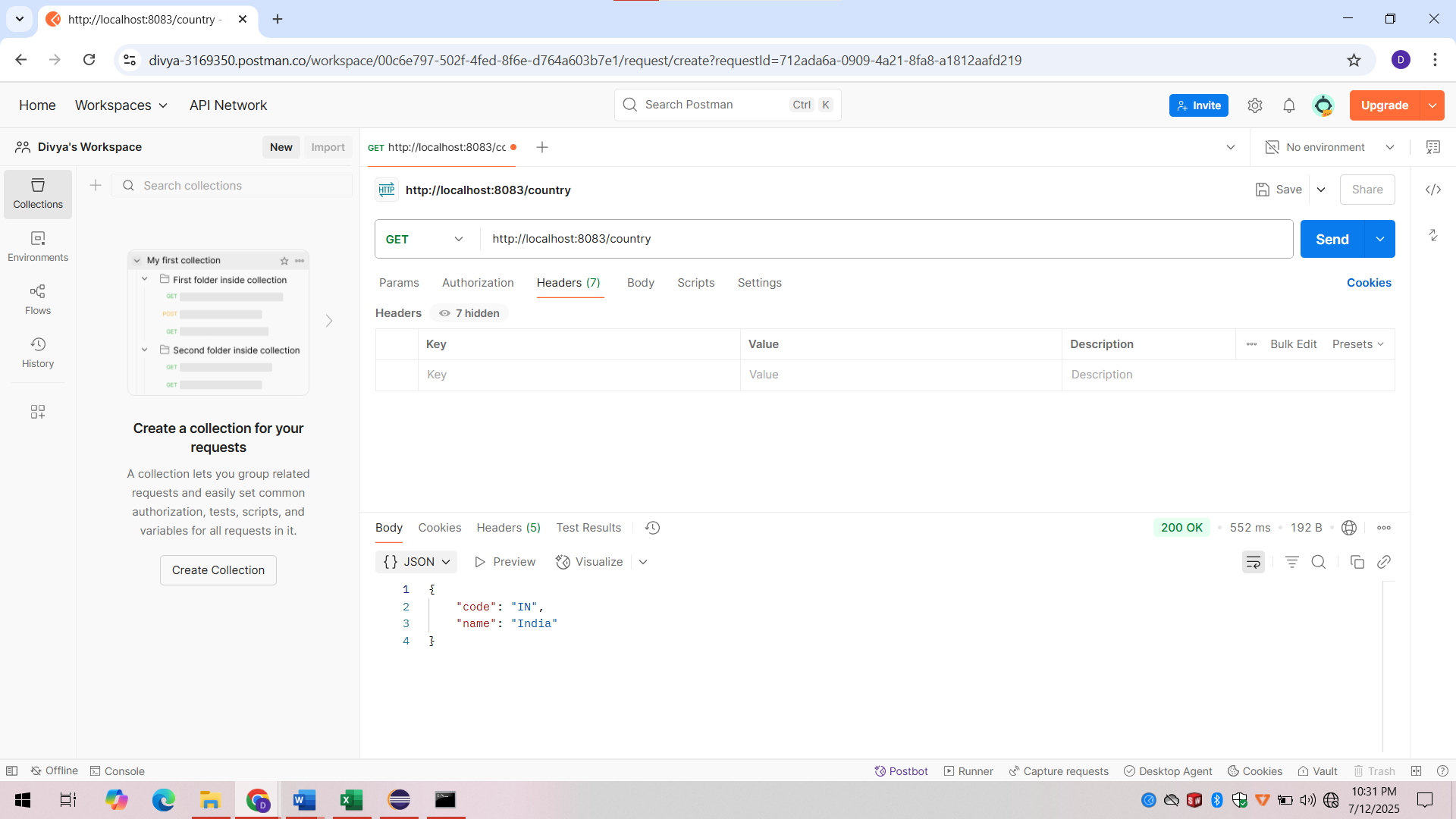
**Console output in eclipse**

****

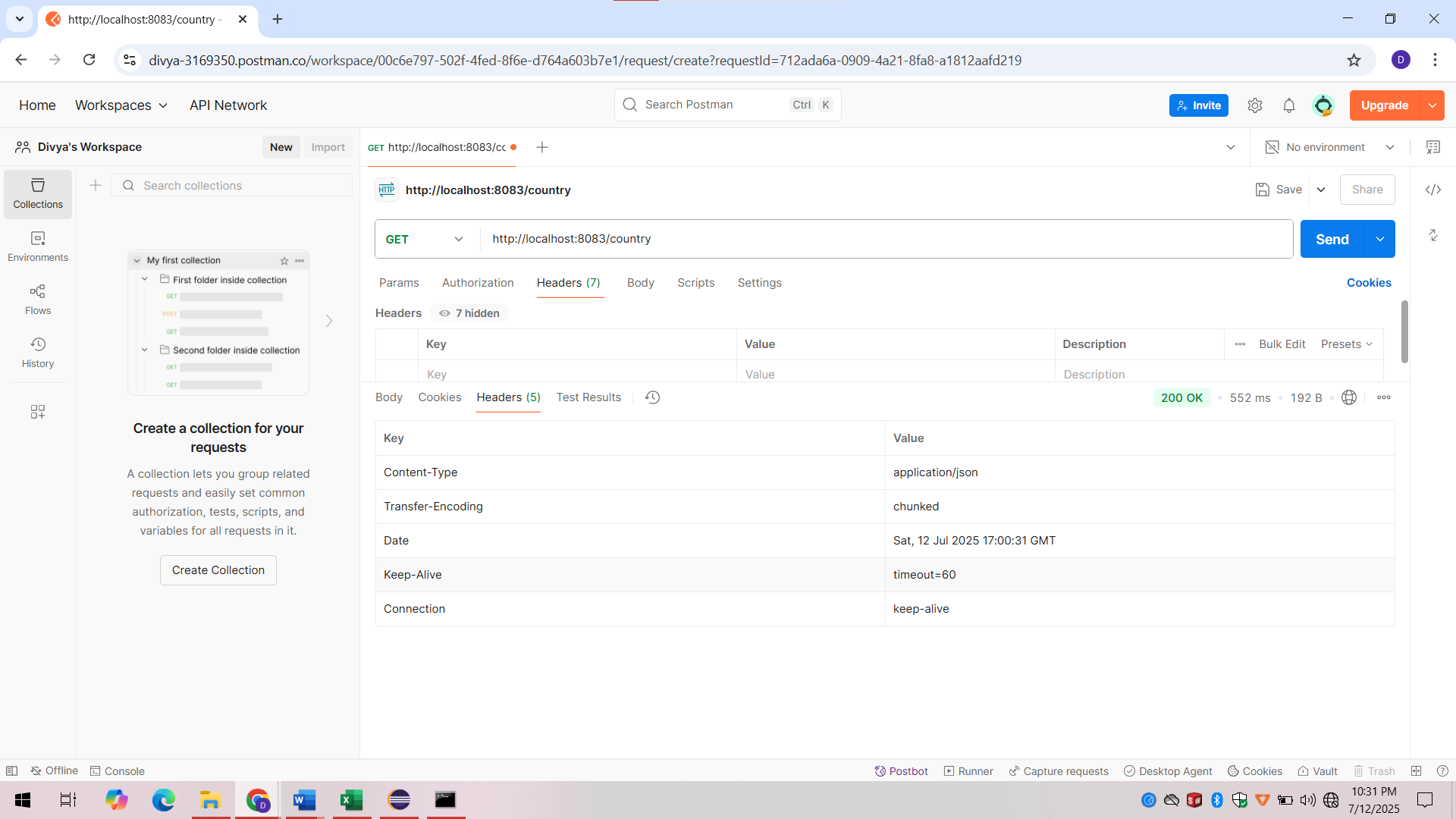
**Header details in chrome browser:**

****

**Output in postman:**

****

**Header details in postman:**

****

**Explanation:**

* In the Spring Boot springlearn application, a RESTful web service was implemented to return the details of India through the /country endpoint.
* A Country model class was created with fields code and name, and a bean representing India was defined in an external Spring XML configuration file named country.xml.
* The CountryController class loads this bean using ApplicationContext and returns it in the getCountryIndia() method, which is mapped to /country using the @RequestMapping annotation.
* When the URL http://localhost:8083/country is accessed in a browser or Postman, the application returns a JSON response containing India’s details: { "code": "IN", "name": "India" }.
* Spring Boot internally uses the Jackson library to automatically convert the Java object into JSON format.
* The request and response header details can be observed using Chrome Developer Tools (under the Network tab) or in Postman by viewing the Headers tab after sending the request, verifying correct transmission and response structure.

**05. REST - Get country based on country code**   
Solution:

**Country.java:**

package com.cognizant.springlearn.model;

public class Country {

private String code;

private String name;

public String getCode() {

return code;

}

public void setCode(String code) {

this.code = code;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

**country.xml (File created in src/main/resources):**

<beans xmlns="http://www.springframework.org/schema/beans"

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

xsi:schemaLocation="

http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="countryList" class="java.util.ArrayList">

<constructor-arg>

<list>

<bean class="com.cognizant.springlearn.model.Country">

<property name="code" value="IN"/>

<property name="name" value="India"/>

</bean>

<bean class="com.cognizant.springlearn.model.Country">

<property name="code" value="US"/>

<property name="name" value="United States"/>

</bean>

<bean class="com.cognizant.springlearn.model.Country">

<property name="code" value="FR"/>

<property name="name" value="France"/>

</bean>

</list>

</constructor-arg>

</bean>

</beans>

**CountryService.java:**

package com.cognizant.springlearn.service;

import com.cognizant.springlearn.model.Country;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class CountryService {

public Country getCountry(String code) {

ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");

List<Country> countries = (List<Country>) context.getBean("countryList");

// Case-insensitive match using lambda

return countries.stream()

.filter(c -> c.getCode().equalsIgnoreCase(code))

.findFirst()

.orElse(null); // Or throw custom exception

}

}

**CountryController.java (class in src/main/java):**

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.model.Country;

import com.cognizant.springlearn.service.CountryService;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

@RestController

public class CountryController {

private static final Logger LOGGER = LoggerFactory.getLogger(CountryController.class);

@Autowired

private CountryService countryService;

@GetMapping("/country/{code}")

public Country getCountry(@PathVariable String code) {

LOGGER.info("START: getCountry()");

Country country = countryService.getCountry(code);

LOGGER.info("END: getCountry()");

return country;

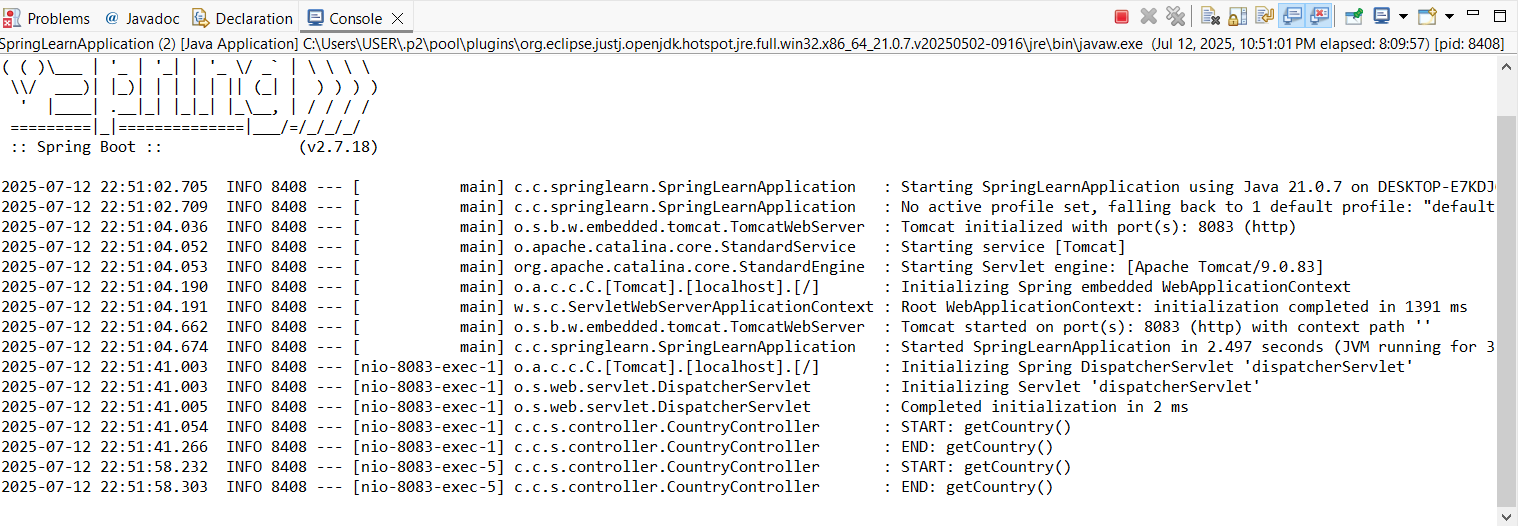
}

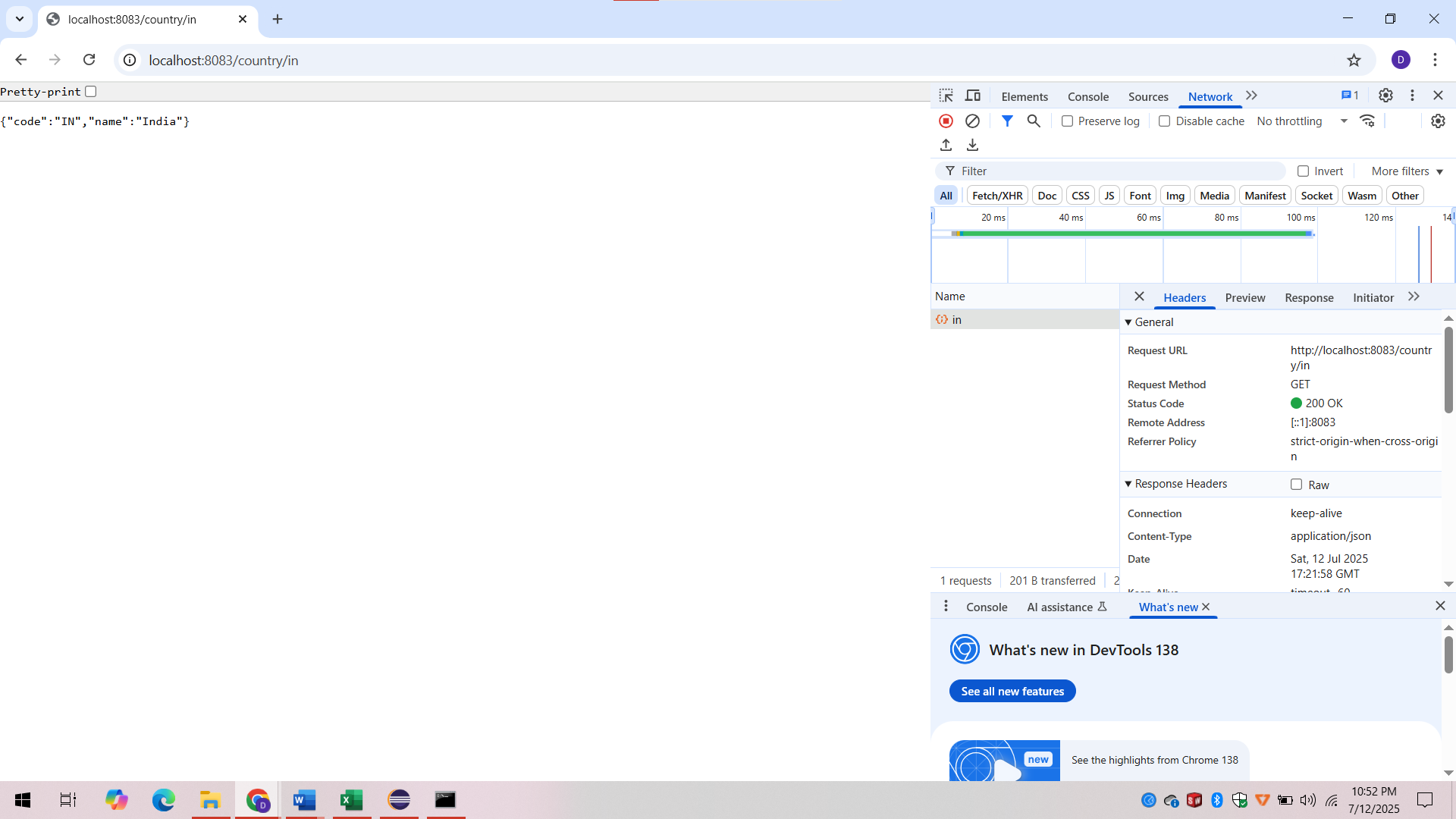
}

**application.properties (File created in src/main/resources):**

server.port=8083

**Output:**

****



**Explanation:**

* In the spring-learn application, a RESTful web service was developed to return details of a specific country based on its country code through the /country/{code} endpoint.
* The controller method receives the code as a path variable and calls the getCountry(code) method in the CountryService class.
* The service loads a list of countries from the Spring XML configuration (country.xml) and uses a case-insensitive search to find and return the matching Country object.
* The response is automatically converted to JSON using Spring Boot’s built-in Jackson library.
* This functionality was tested using a browser, where request and response headers were also verified using Developer Tools and the Headers tab respectively.

**JWT Handson**

**06.Create authentication service that returns JWT**

Solution:

**JwtAuthServiceApplication.java:**

package com.cognizant.jwtauth;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class JwtAuthServiceApplication {

public static void main(String[] args) {

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

}

}

**SecurityConfig.java:**

package com.cognizant.jwtauth.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.core.userdetails.User;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.security.crypto.password.NoOpPasswordEncoder;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import java.util.Base64;

import java.util.Collections;

@Configuration

@EnableWebSecurity

public class SecurityConfig {

@Bean

public InMemoryUserDetailsManager userDetailsService() {

UserDetails user = User.*withUsername*("user")

.password("{noop}pwd") // <-- {noop} required if NoOp encoder

.roles("USER")

.build();

return new InMemoryUserDetailsManager(user);

}

@Bean

public AuthenticationManager authenticationManager(AuthenticationConfiguration config) throws Exception {

return config.getAuthenticationManager();

}

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http.csrf().disable()

.authorizeRequests()

.antMatchers("/authenticate").permitAll()

.anyRequest().authenticated()

.and()

.httpBasic(); // Basic Auth

return http.build();

}

}

@Bean

public static PasswordEncoder passwordEncoder() {

return NoOpPasswordEncoder.getInstance();

}

}

**AuthenticationController.java:**

**package** com.cognizant.jwtauth.config;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

**import** org.springframework.security.core.userdetails.User;

**import** org.springframework.security.core.userdetails.UserDetails;

**import** org.springframework.security.provisioning.InMemoryUserDetailsManager;

**import** org.springframework.security.crypto.password.PasswordEncoder;

**import** org.springframework.security.crypto.password.NoOpPasswordEncoder;

**import** org.springframework.security.authentication.AuthenticationManager;

**import** org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;

**import** org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

**import** java.util.Base64;

**import** java.util.Collections;

@Configuration

@EnableWebSecurity

**public** **class** SecurityConfig {

@Bean

**public** InMemoryUserDetailsManager userDetailsService() {

UserDetails user = User.*withUsername*("user")

.password("{noop}pwd") // <-- {noop} required if NoOp encoder

.roles("USER")

.build();

**return** **new** InMemoryUserDetailsManager(user);

}

@Bean

**public** AuthenticationManager authenticationManager(AuthenticationConfiguration config) **throws** Exception {

**return** config.getAuthenticationManager();

}

@Bean

**public** SecurityFilterChain filterChain(HttpSecurity http) **throws** Exception {

http.csrf().disable()

.authorizeRequests()

.antMatchers("/authenticate").permitAll()

.anyRequest().authenticated()

.and()

.httpBasic(); // Basic Auth

**return** http.build();

}

}

@Bean

**public** **static** PasswordEncoder passwordEncoder() {

**return** NoOpPasswordEncoder.getInstance();

}

}

**AuthRequest.java:**

package com.cognizant.jwtauth.model;

public class AuthRequest {

private String username;

private String password;

// Default constructor

public AuthRequest() {}

// Parameterized constructor

public AuthRequest(String username, String password) {

this.username = username;

this.password = password;

}

// Getters and Setters

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

}

**JwtUtil.java:**

package com.cognizant.jwtauth.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import org.springframework.stereotype.Component;

import java.util.Date;

@Component

public class JwtUtil {

private final String SECRET\_KEY = "mysecretkey";

public String generateToken(String username) {

return Jwts.*builder*()

.setSubject(username)

.setIssuedAt(new Date())

.setExpiration(new Date(System.*currentTimeMillis*() + 1000 \* 60 \* 60 \* 10))

.signWith(SignatureAlgorithm.*HS256*, SECRET\_KEY)

.compact();

}

}

**application.properties (File created src/main/resources):**

server.port=8090

**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.cognizant</groupId>

<artifactId>jwt-auth-service</artifactId>

<version>0.0.1-SNAPSHOT</version>

<parent>

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

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

<version>2.7.12</version>

<relativePath/>

</parent>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<!-- Spring Boot Web -->

<dependency>

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

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

</dependency>

<!-- Spring Boot Security -->

<dependency>

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

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

</dependency>

<!-- JWT -->

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.1</version>

</dependency>

<!-- Lombok (Optional) -->

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<version>1.18.30</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-api</artifactId>

<version>0.11.5</version>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-impl</artifactId>

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-jackson</artifactId> <!-- or jjwt-gson if using Gson -->

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

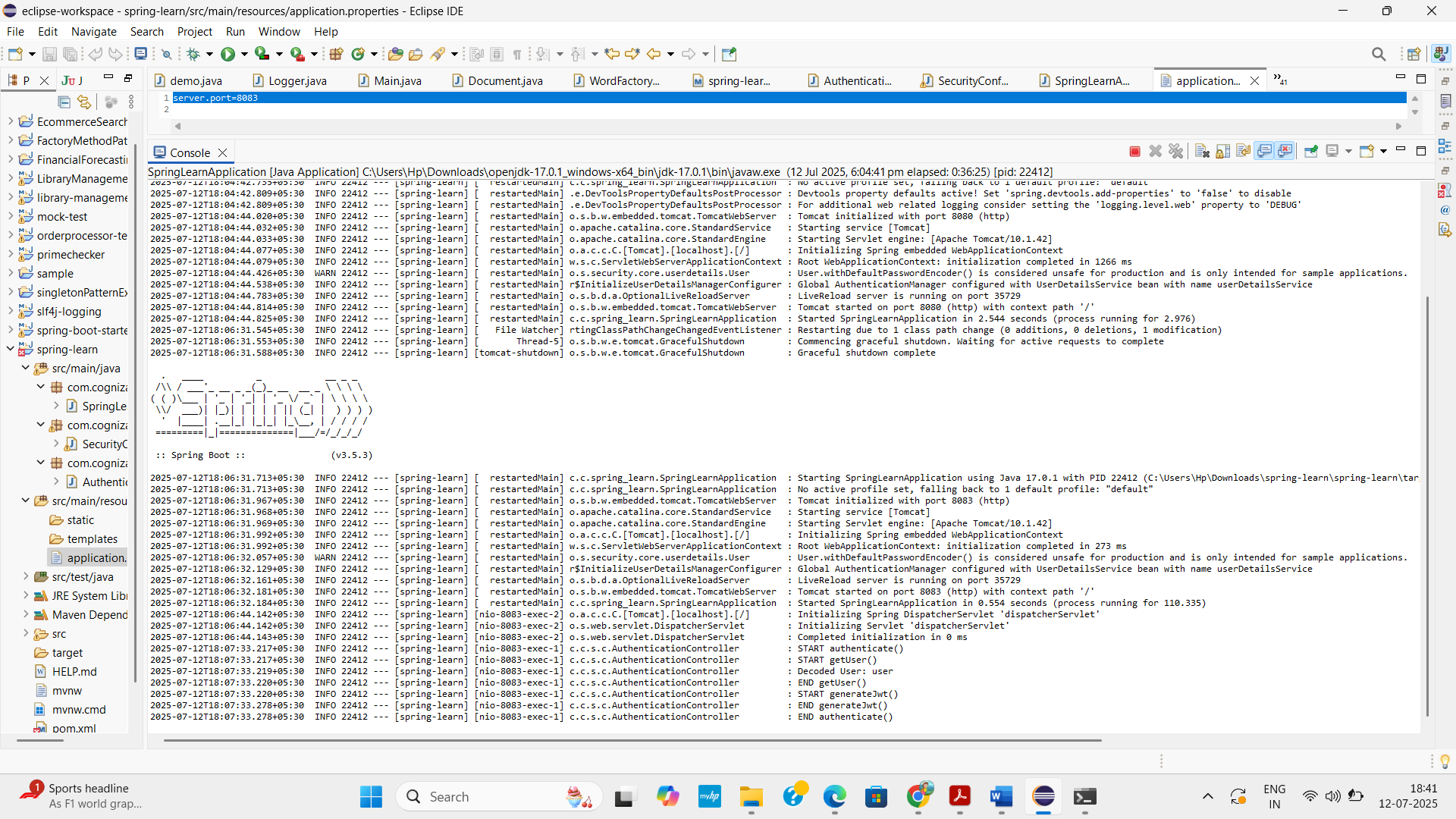
</plugin>

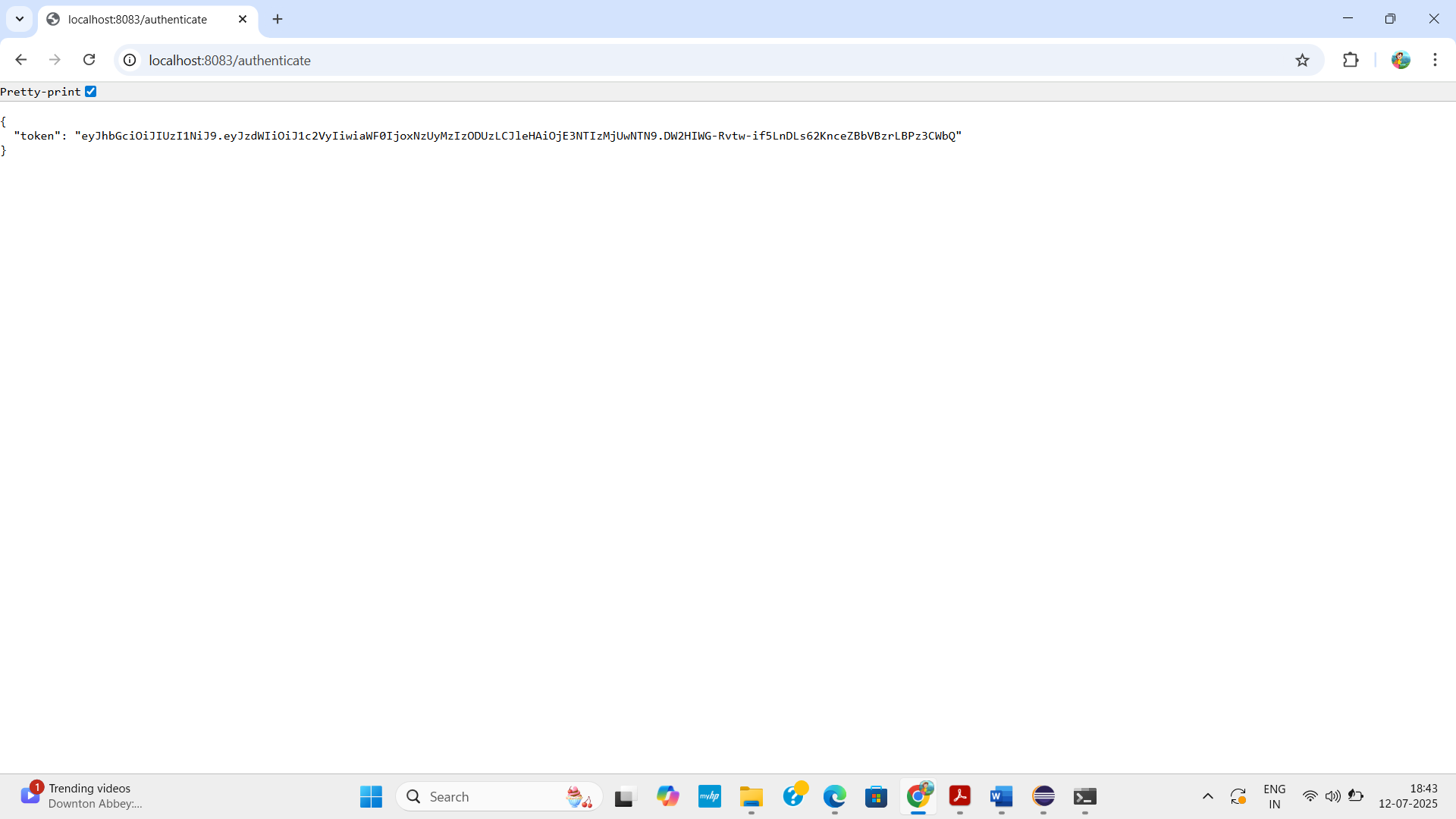
</plugins>

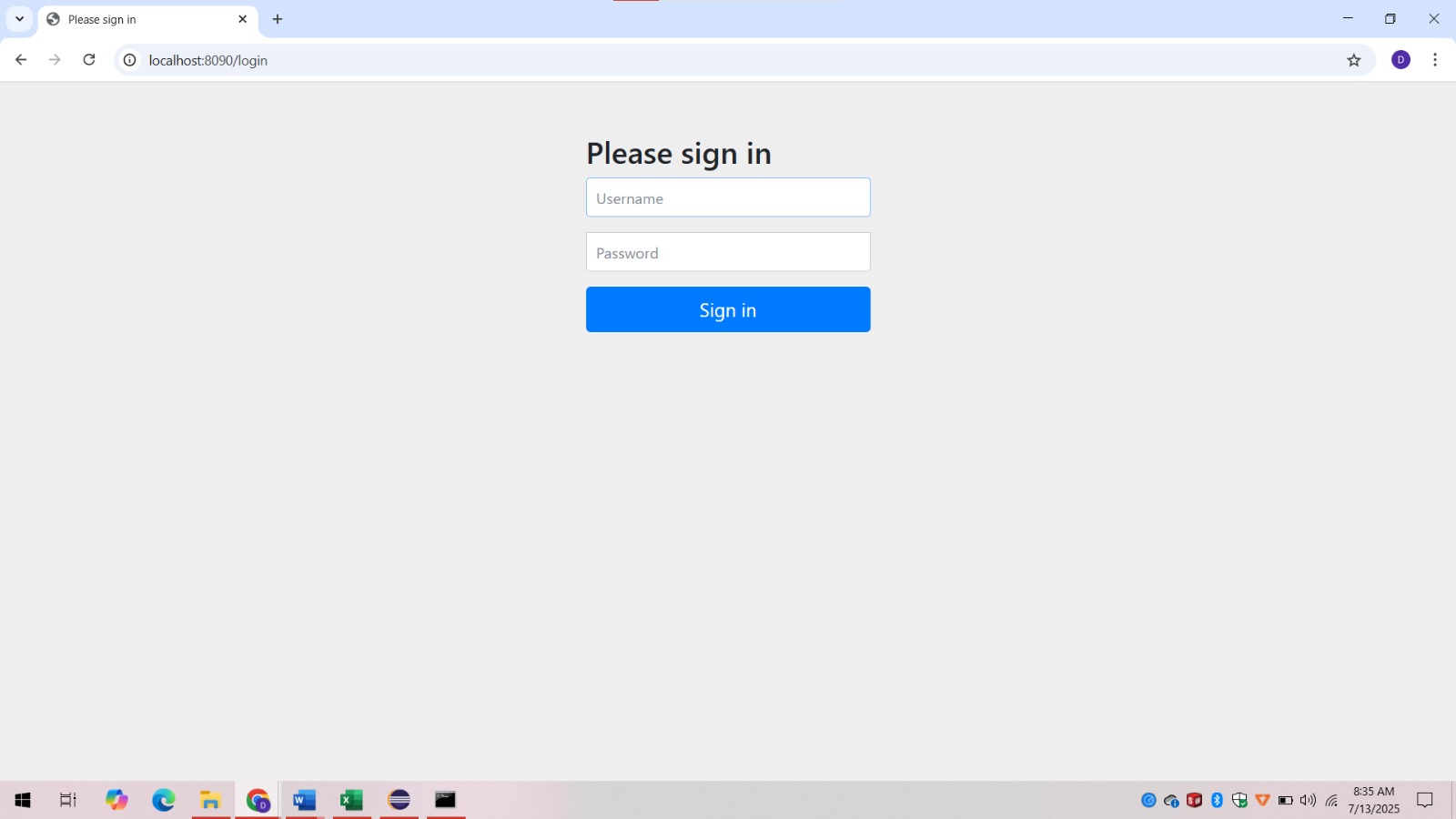
</build>

</project>

**Output:**







**Explanation:**

* In this JWT Authentication Service, user credentials are validated using Spring Security.
* When a client sends a request with a valid username and password (via Basic Auth), the /authenticate endpoint generates a JWT token.
* This token is created using the io.jsonwebtoken (JJWT) library and contains user-related claims like subject, issued time, and expiration. The generated JWT is digitally signed and sent back to the client.
* This token can then be used to securely access other protected endpoints in subsequent requests.