**Hands on 1**

**Create a Spring Web Project using Maven** 

1. src/main/java - Folder with application code

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("Application Started...");

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

}

}

1. src/main/resources - Folder for application configuration

spring.application.name=spring-learn

1. src/test/java - Folder with code for testing the application

package com.cognizant.spring\_learn;

import org.junit.jupiter.api.Test;

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

@SpringBootTest

class SpringLearnApplicationTests {

@Test

void contextLoads() {

}

}

1. SpringLearnApplication.java - Walkthrough the main() method.

public static void main(String[] args) {

System.out.println("Application Started...");

SpringApplication.run(SpringLearnApplication.class, args);

}

1. Purpose of @SpringBootApplication annotation

Combines:

* + @Configuration: Marks class as source of Spring configuration
  + @EnableAutoConfiguration: Enables Spring Boot’s auto-configuration
  + @ComponentScan: Scans current and sub-packages for Spring components

1. pom.xml
   1. Walkthrough all the configuration defined in XML file

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

<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 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

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

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

<version>3.5.3</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-learn</name>

<description>Demo project for Spring Boot</description>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

<java.version>17</java.version>

</properties>

<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>

<optional>true</optional>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

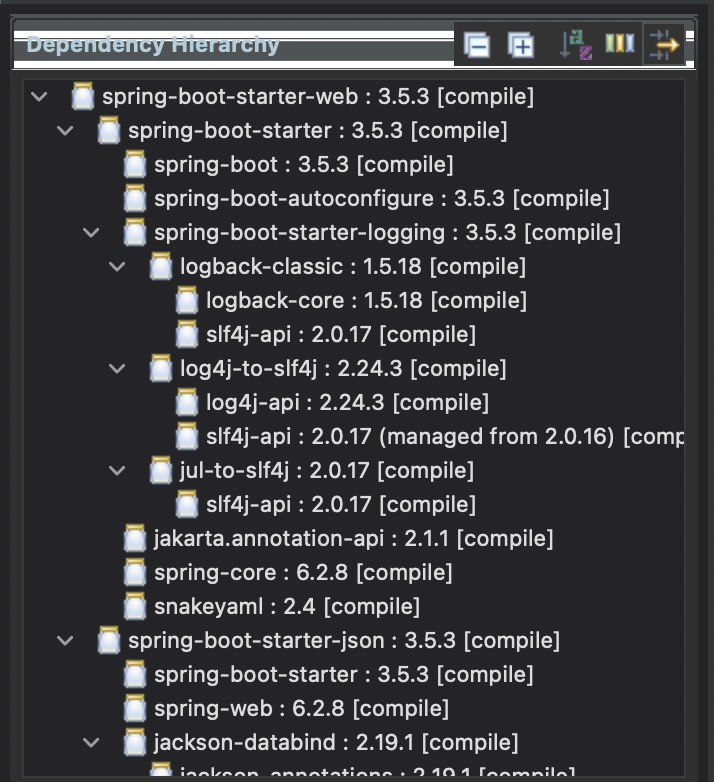
</plugin>

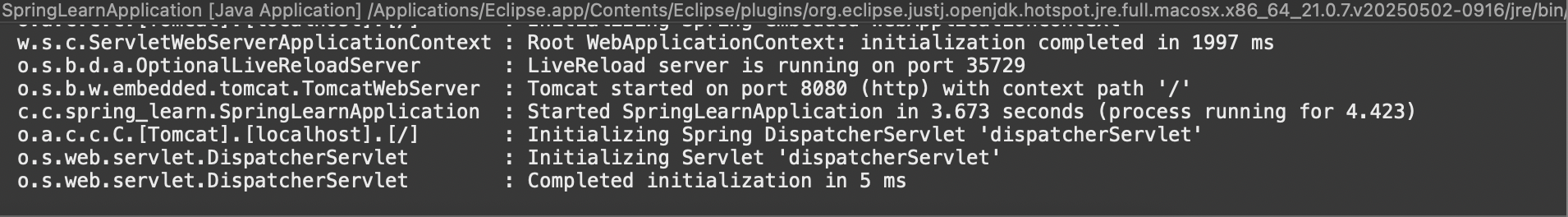
</plugins>

</build>

</project>

* 1. Open 'Dependency Hierarchy' and show the dependency tree.



1. Output

**Hands on 6**

**Spring Core – Load list of countries from Spring Configuration XML** 

package com.cognizant.spring\_learn.model;

public class Country {

private String code;

private String name;

public Country() {}

public Country(String code, String name) {

this.code = code;

this.name = name;

}

// Getters and Setters

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 + "]";

}

}

<?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">

<!-- Individual Country Beans -->

<bean id="in" class="com.cognizant.spring\_learn.model.Country">

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

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

</bean>

<bean id="us" class="com.cognizant.spring\_learn.model.Country">

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

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

</bean>

<bean id="de" class="com.cognizant.spring\_learn.model.Country">

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

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

</bean>

<bean id="jp" class="com.cognizant.spring\_learn.model.Country">

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

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

</bean>

<!-- List of Country Beans -->

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

<constructor-arg>

<list>

<ref bean="in" />

<ref bean="us" />

<ref bean="de" />

<ref bean="jp" />

</list>

</constructor-arg>

</bean>

</beans>

package com.cognizant.spring\_learn;

import java.util.List;

import com.cognizant.spring\_learn.model.Country;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

@SpringBootApplication

public class SpringLearnApplication {

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

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

displayCountries(); // Call your method here

}

public static void displayCountries() {

LOGGER.debug("START displayCountries");

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

@SuppressWarnings("unchecked")

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

for (Country country : countryList) {

LOGGER.debug(country.toString());

}

LOGGER.debug("END displayCountries");

}

}

spring.application.name=spring-learn

logging.level.root=DEBUG

**SME Explanation of XML Tags**

<list>

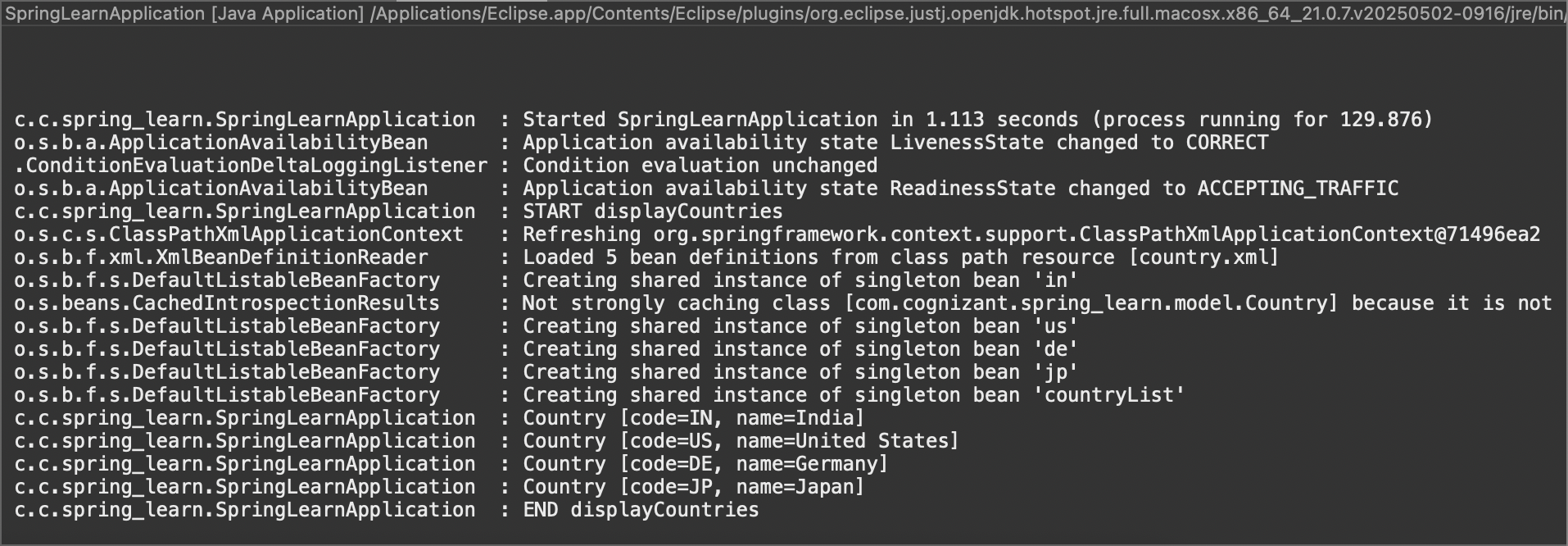
* Creates a Java List (like ArrayList)
* Allows adding multiple elements (bean references)

<ref>

* Points to another bean by ID
* For example: <ref bean="in" /> refers to the bean with id=“in"

bean attribute

* Used inside <ref> to specify which bean to link to
* It injects that bean into a list or another bean

Output

**Hello World RESTful Web Service**   
  
package com.cognizant.spring\_learn.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.debug("START sayHello");

LOGGER.debug("END sayHello");

return "Hello World!!";

}

}

server.port=8083

logging.level.root=DEBUG

**SME to explain the following aspects:**

In network tab of developer tools show the HTTP header details received

HTTP/1.1 200

Content-Type: text/html;charset=UTF-8

Content-Length: 13

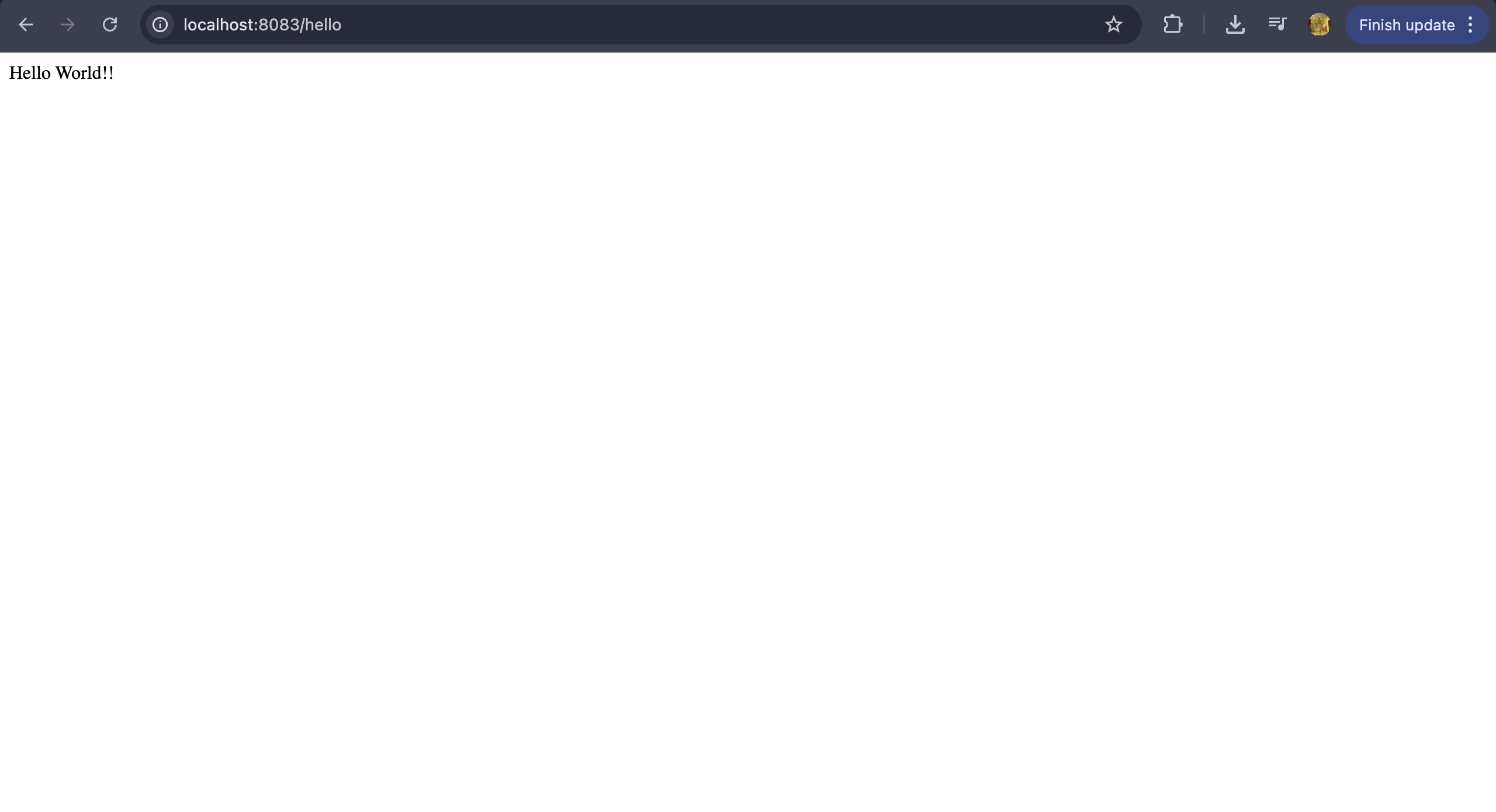
Date: Sat, 12 Jul 2025 12:57:03 GMT

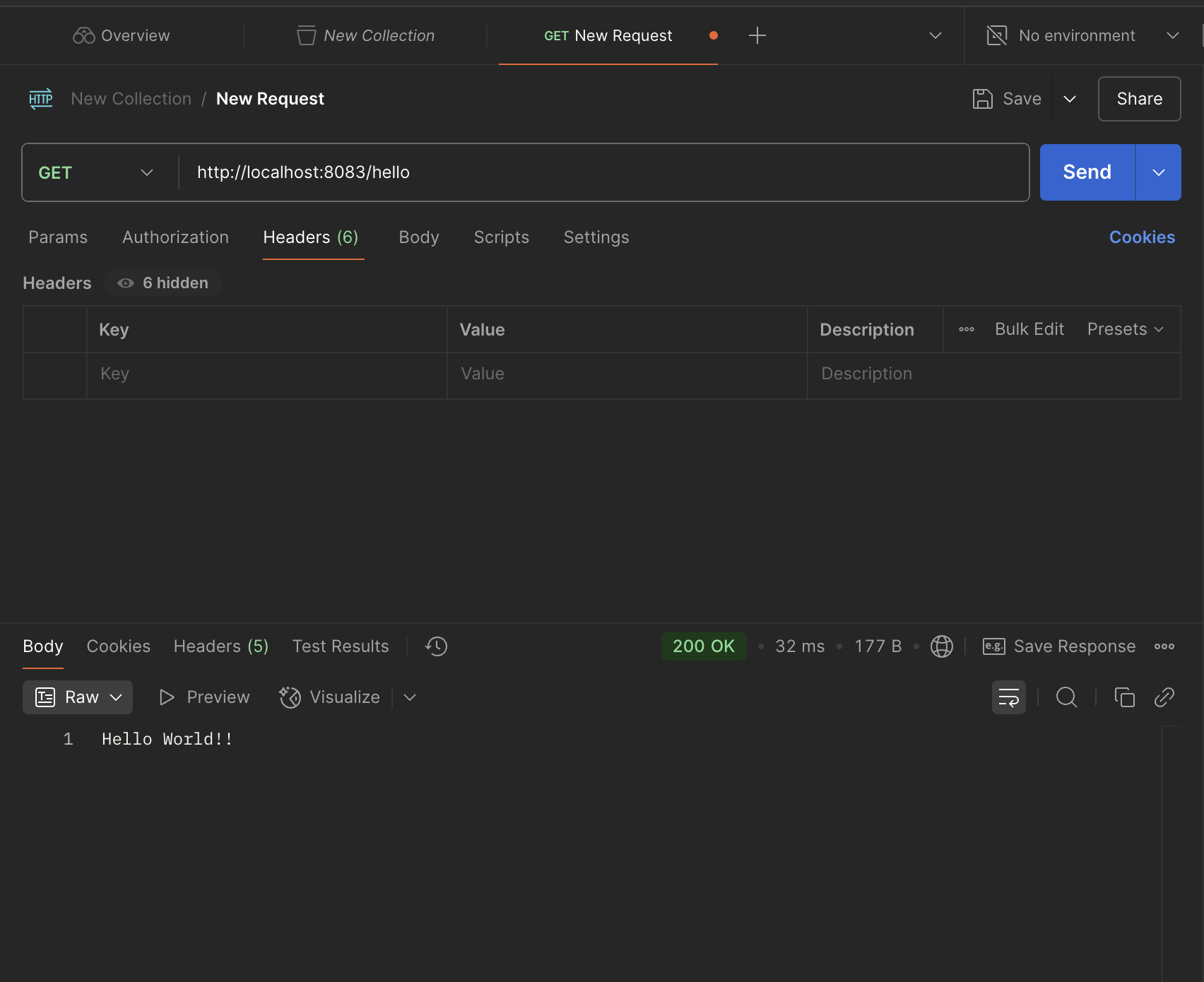
Keep-Alive: timeout=60

Connection: keep-alive

In postman click on "Headers" tab to view the HTTP header details received

|  |  |
| --- | --- |
| Content-Type | text/plain;charset=UTF-8 |
| Content-Length | 13 |
| Date | Sat, 12 Jul 2025 13:00:05 GMT |
| Keep-Alive | timeout=60 |
| Connection | keep-alive |

**OUTPUT**

****

**REST - Country Web Service**   
  
package com.cognizant.spring\_learn.controller;

import com.cognizant.spring\_learn.model.Country;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

@RestController

public class CountryController {

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

@RequestMapping("/country")

public Country getCountryIndia() {

LOGGER.debug("START getCountryIndia");

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

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

LOGGER.debug("END getCountryIndia");

return country;

}

}

package com.cognizant.spring\_learn.model;

public class Country {

private String code;

private String name;

public Country() {

}

public Country(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 + "]";

}

}

<?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">

<!-- Individual Country Beans -->

<bean id="in" class="com.cognizant.spring\_learn.model.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="in" />

</list>

</constructor-arg>

</bean>

</beans>

**SME to explain the following aspects:**

* What happens in the controller method?
  + Spring maps the URL /country to the getCountryIndia() method.
  + It loads the Spring context from country.xml.
  + It fetches the in bean (a Country object).
  + It returns the object, and Spring automatically converts it to JSON using Jackson (built into Spring Boot).
* How the bean is converted into JSON reponse?
  + Spring Boot includes Jackson, a JSON library.
  + Any POJO with getters/setters returned from a @RestController is automatically serialized into JSON.
  + This is done behind the scenes by the MappingJackson2HttpMessageConverter.
* In network tab of developer tools show the HTTP header details received

Connection keep-alive

Content-Type application/json

Date Sat, 12 Jul 2025 13:14:02 GMT

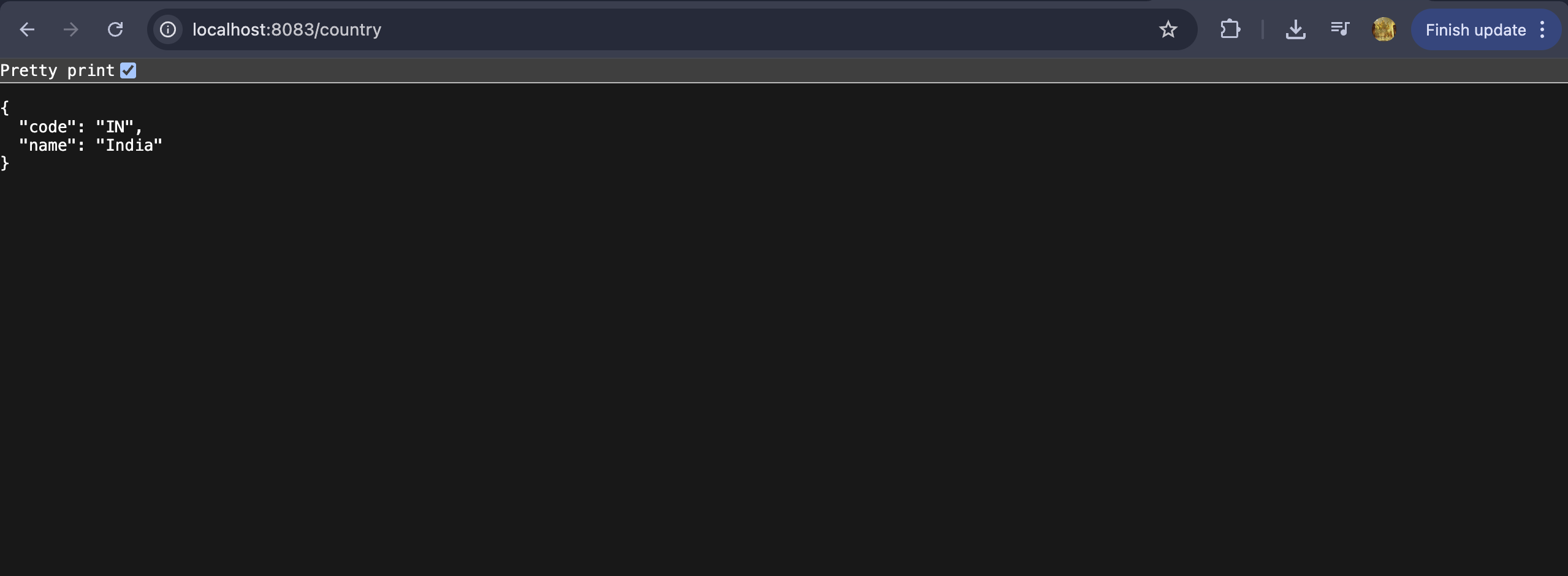
Keep-Alive timeout=60

Transfer-Encoding chunked

* In postman click on "Headers" tab to view the HTTP header details received

|  |  |
| --- | --- |
| Content-Type | application/json |
| Transfer-Encoding | chunked |
| Date | Sat, 12 Jul 2025 13:16:27 GMT |
| Keep-Alive | timeout=60 |
| Connection | keep-alive |

**OUTPUT**



**REST - Get country based on country code**   
  
package com.cognizant.spring\_learn.service;

import com.cognizant.spring\_learn.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");

@SuppressWarnings("unchecked")

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

return countryList.stream()

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

.findFirst()

.orElse(null); // You can throw a custom exception here if needed

}

}

package com.cognizant.spring\_learn.controller;

import com.cognizant.spring\_learn.model.Country;

import com.cognizant.spring\_learn.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("/countries/{code}")

public Country getCountry(@PathVariable String code) {

LOGGER.debug("START getCountry - code: {}", code);

Country country = countryService.getCountry(code);

LOGGER.debug("END getCountry");

return country;

}

}

<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.spring\_learn.model.Country">

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

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

</bean>

<bean id="us" class="com.cognizant.spring\_learn.model.Country">

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

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

</bean>

<bean id="de" class="com.cognizant.spring\_learn.model.Country">

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

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

</bean>

<bean id="jp" class="com.cognizant.spring\_learn.model.Country">

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

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

</bean>

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

<constructor-arg>

<list>

<ref bean="in"/>

<ref bean="us"/>

<ref bean="de"/>

<ref bean="jp"/>

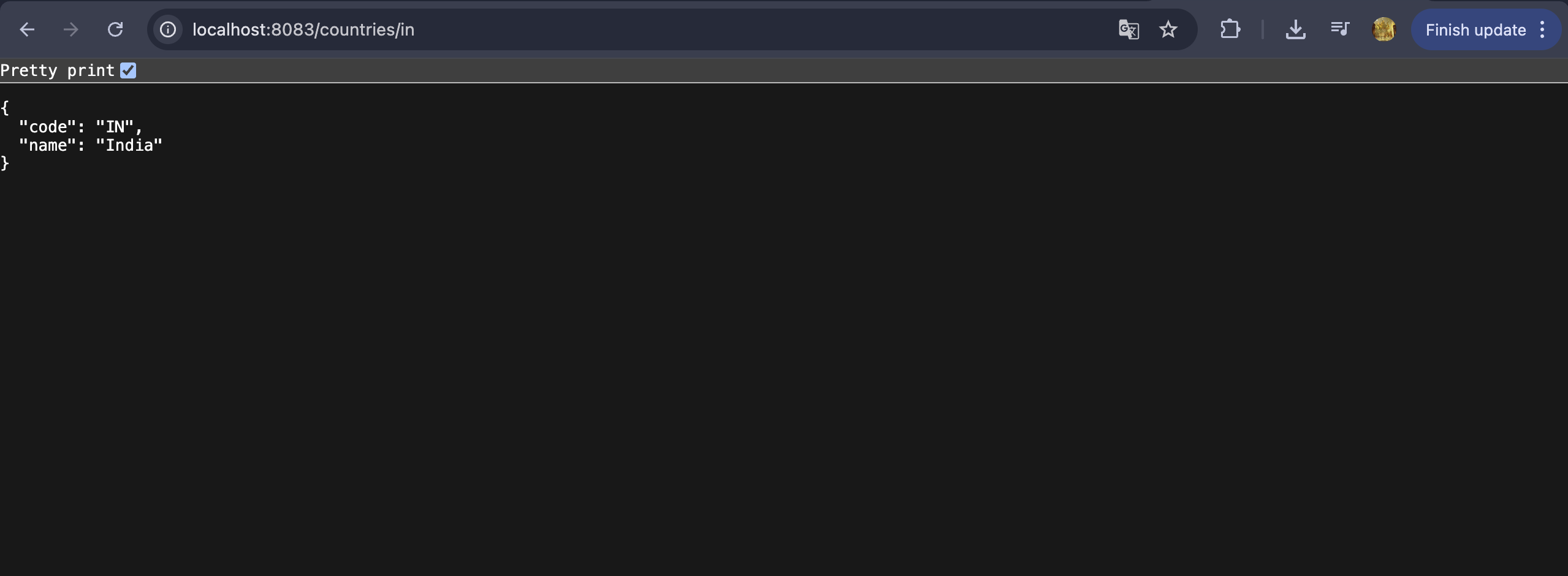
</list>

</constructor-arg>

</bean>

</beans>

**OUTPUT**

****

**Create authentication service that returns JWT**

**// File : SecurityConfig.java**

package com.cognizant.spring\_learn.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

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

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

import org.springframework.security.web.SecurityFilterChain;

*@Configuration*

public class SecurityConfig {

*@Bean*

public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

http

.csrf(csrf -> csrf.disable())

.authorizeHttpRequests(auth -> auth

.requestMatchers("/authenticate").permitAll()

.anyRequest().authenticated())

.httpBasic()

;

return http.build();

}

*@Bean*

public UserDetailsService userDetailsService() {

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

.password("{noop}pwd") // password is 'pwd', no encoding

.roles("USER")

.build();

return new InMemoryUserDetailsManager(user);

}

}

**// File : application.properties**

spring.application.name=spring-learn

server.port=8091

**// File : AuthenticationController.java**

package com.cognizant.spring\_learn.controller;

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

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

//import javax.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletRequest;

import org.apache.tomcat.util.codec.binary.Base64;

import com.cognizant.spring\_learn.util.JwtUtil;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

*@RestController*

public class AuthenticationController {

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

*@GetMapping*("/authenticate")

public String authenticate(HttpServletRequest request) {

***LOGGER***.debug("START authenticate()");

String header = request.getHeader("Authorization");

String username = "";

String password = "";

if (header != null && header.startsWith("Basic ")) {

String base64Credentials = header.substring("Basic ".length());

byte[] credDecoded = Base64.decodeBase64(base64Credentials);

String credentials = new String(credDecoded);

String[] values = credentials.split(":", 2);

username = values[0];

password = values[1];

}

***LOGGER***.debug("Decoded username: {}", username);

// Normally you should validate the username & password

String token = JwtUtil.*generateToken*(username);

***LOGGER***.debug("END authenticate()");

return "{\"token\":\"" + token + "\"}";

}

}

**// File : JwtUtil.java**

package com.cognizant.spring\_learn.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import java.util.Date;

public class JwtUtil {

private static final String ***SECRET\_KEY*** = "secret"; // Use a secure secret in real projects

public static String generateToken(String username) {

long currentTimeMillis = System.*currentTimeMillis*();

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date(currentTimeMillis))

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

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

.compact();

}

}

**// File : pom.xml**

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

<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 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

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

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

<version>3.5.3</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-learn</name>

<description>Demo project for Spring Boot</description>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

<java.version>17</java.version>

</properties>

<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>

<optional>true</optional>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

<dependency>

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

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

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.1</version>

</dependency>

<dependency>

<groupId>javax.xml.bind</groupId>

<artifactId>jaxb-api</artifactId>

<version>2.3.1</version>

</dependency>

<dependency>

<groupId>org.glassfish.jaxb</groupId>

<artifactId>jaxb-runtime</artifactId>

<version>2.3.1</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

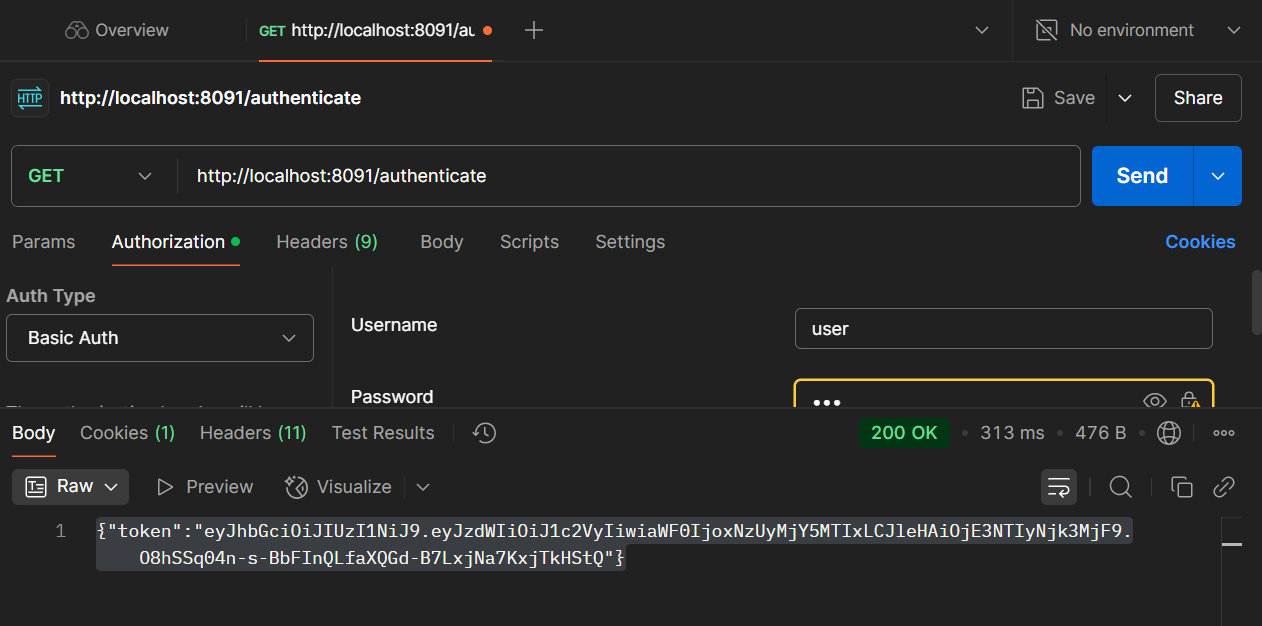
</plugin>

</plugins>

</build>

</project>

**OUTPUT**

****