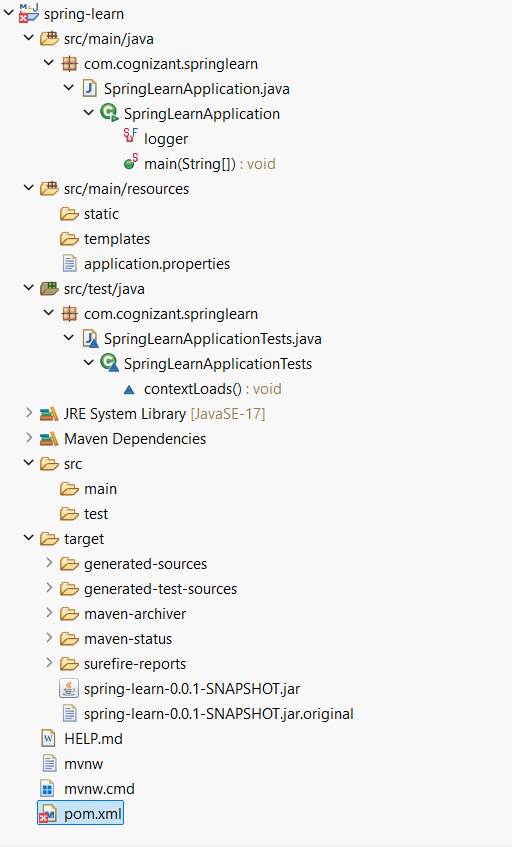
**1. Create a Spring Web Project using Maven**

**Project Name: Spring-learn**

**Project Structure:**

****

**Code:**

SpringLearnApplication.java

package com.cognizant.springlearn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

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

public static void main(String[] args) {

*logger*.info("Starting the Spring Learn application...");

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

*logger*.info("Spring Learn application started successfully.");

}

}

pom.xml:

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-learn</name>

<description>Spring Web Learning Project</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>

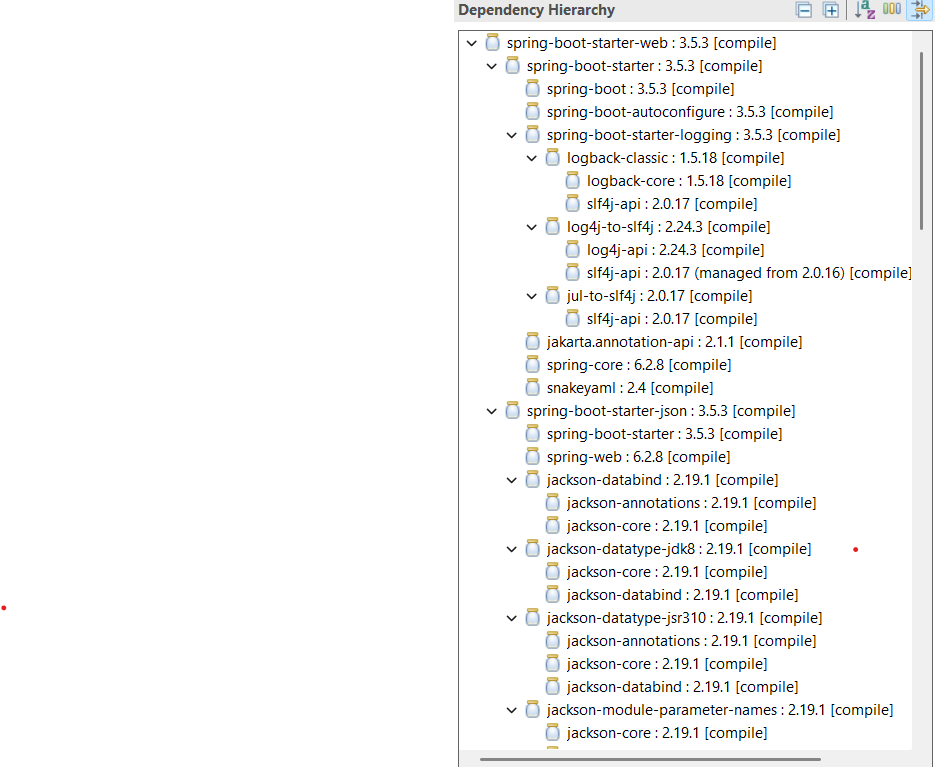
<groupId>org.springframework.boot</groupId>

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

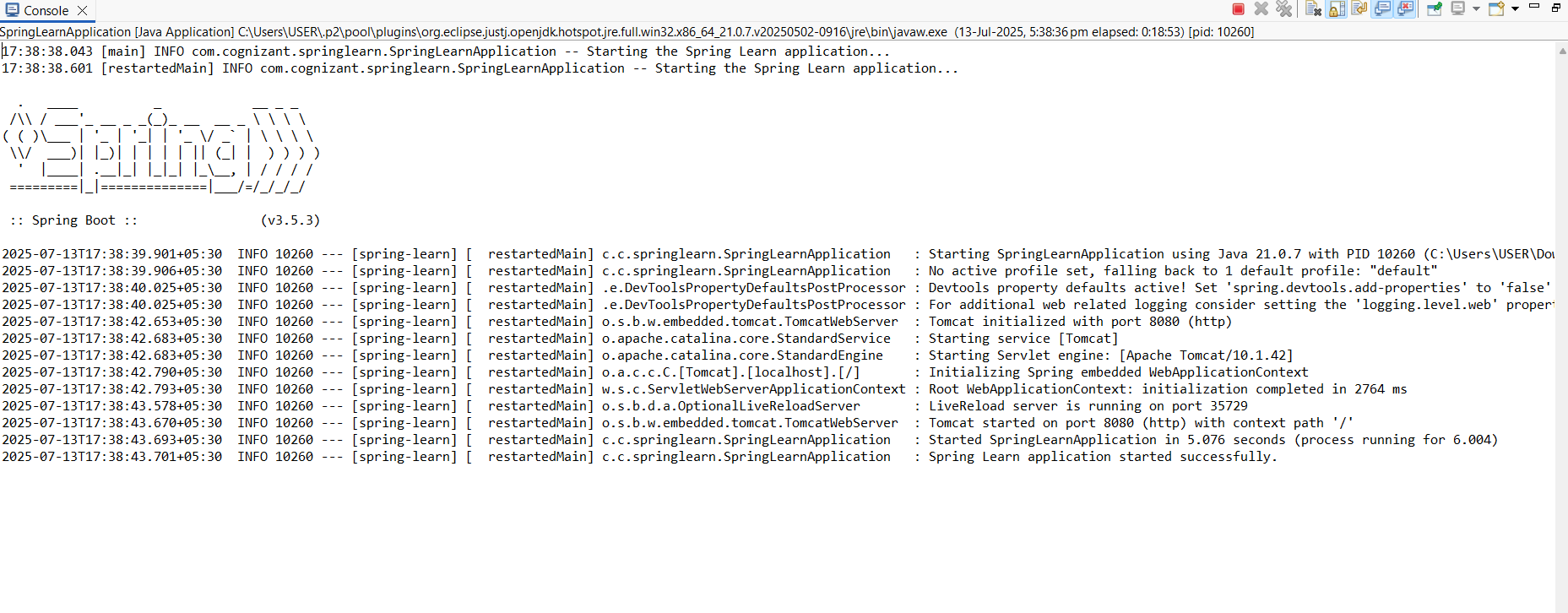
<scope>test</scope>

</dependency>

</dependencies>

**Dependency Hierarchy (from pom.xml):**

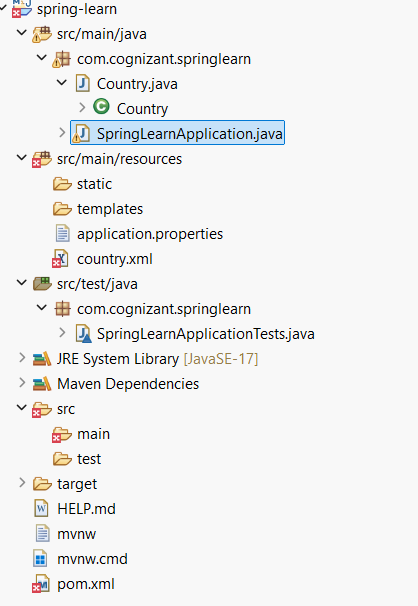
**Output:**

****

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

**Project Name: Spring-learn**

**Project Structure:**

****

**Code:**

Country.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("Getter for code called.");

return code;

}

public void setCode(String code) {

LOGGER.debug("Setter for code called.");

this.code = code;

}

public String getName() {

LOGGER.debug("Getter for name called.");

return name;

}

public void setName(String name) {

LOGGER.debug("Setter for name called.");

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="country" class="com.cognizant.springlearn.Country">

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

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

</bean>

</beans>

SpringLearnApplication.java

package com.cognizant.springlearn;

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) {

LOGGER.info("Starting the Spring Learn application...");

SpringApplication.run(SpringLearnApplication.class, args);

displayCountry();

LOGGER.info("Spring Learn application started successfully.");

}

public static void displayCountry() {

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

Country country = context.getBean("country", Country.class);

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

}

}

application.properties

logging.level.com.cognizant.springlearn=DEBUG

**Output:**



* **<bean> tag:** Declares a Spring-managed Java object.
* **id attribute:** Specifies the unique identifier for the bean. This ID is used to retrieve the bean from the container.
* **class attribute**: Indicates the fully qualified class name of the Java object that the Spring container should instantiate.
* **<property> tag**: Used to inject values into the bean's fields using setter methods.
* **name attribute (in property)**: Corresponds to the name of the property in the Java class (like setCode() and setName()).
* **value attribute (in property)**: Specifies the value to be injected into that property.
* ApplicationContext is the Spring container that manages beans.
* ClassPathXmlApplicationContext loads bean definitions from an XML file in the classpath.

In country.xml we load it using  
ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");

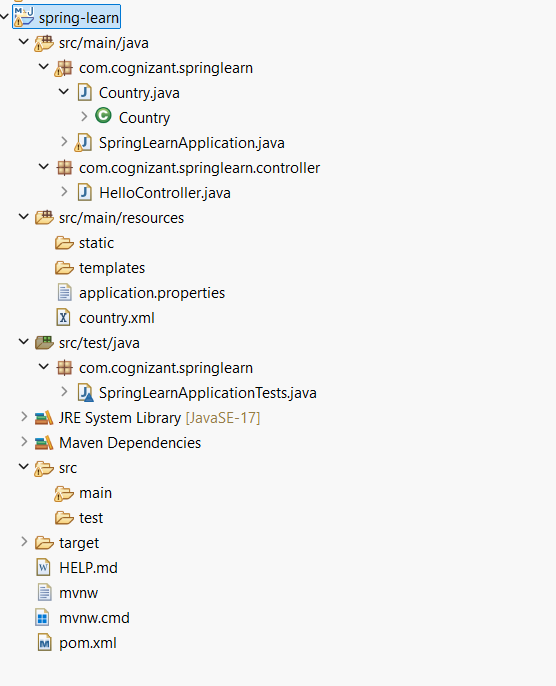
**when context.getBean() is invoked**

* Spring reads the country.xml file.
* Instantiates the Country object.
* Sets the values using setters.
* Returns the configured bean.

**3. Hello World RESTful Web Service**

**Project Name: Spring-learn**

**Project Structure:**

****

**Code:**

**HelloController.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()");

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

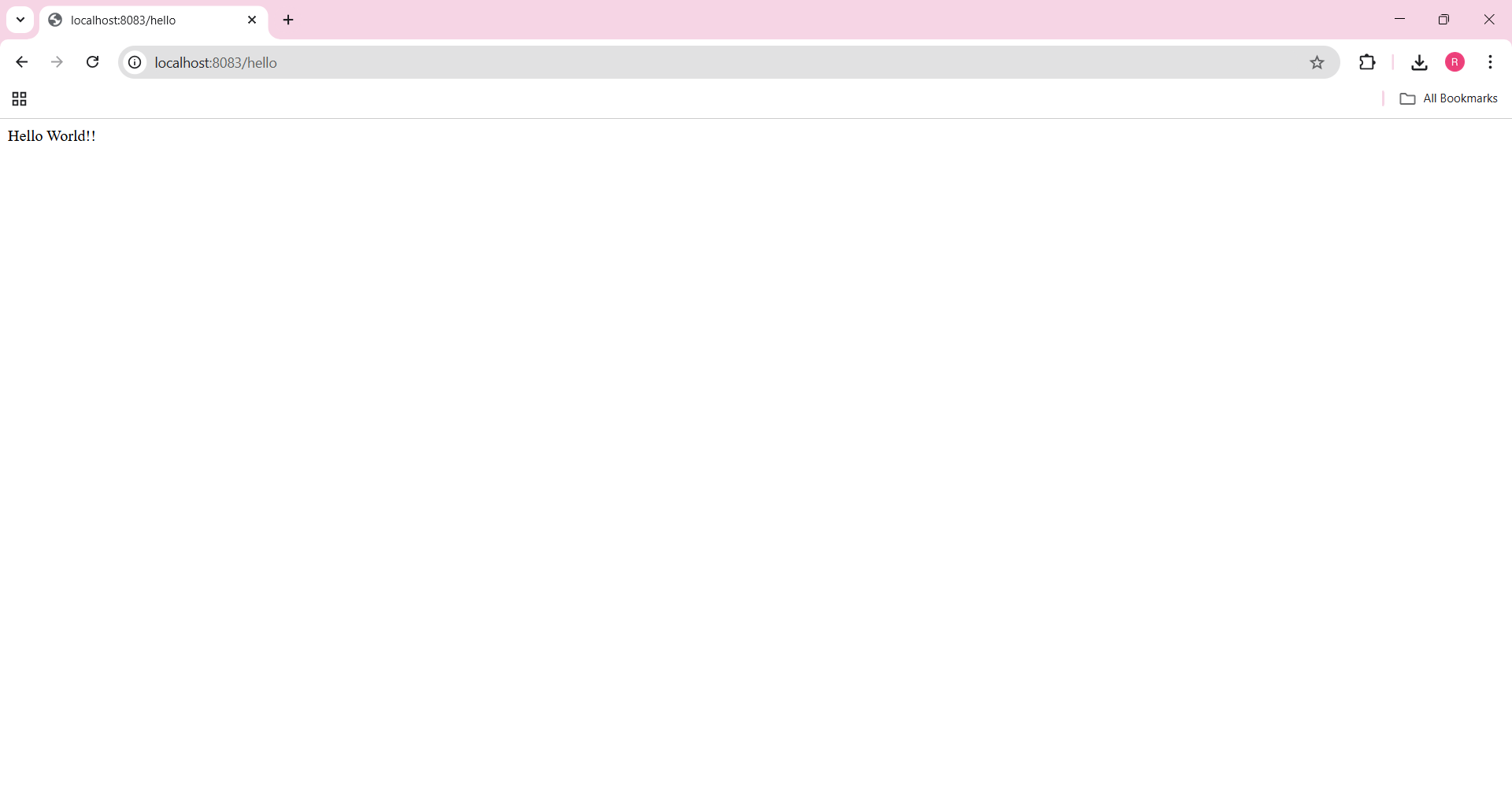
return "Hello World!!";

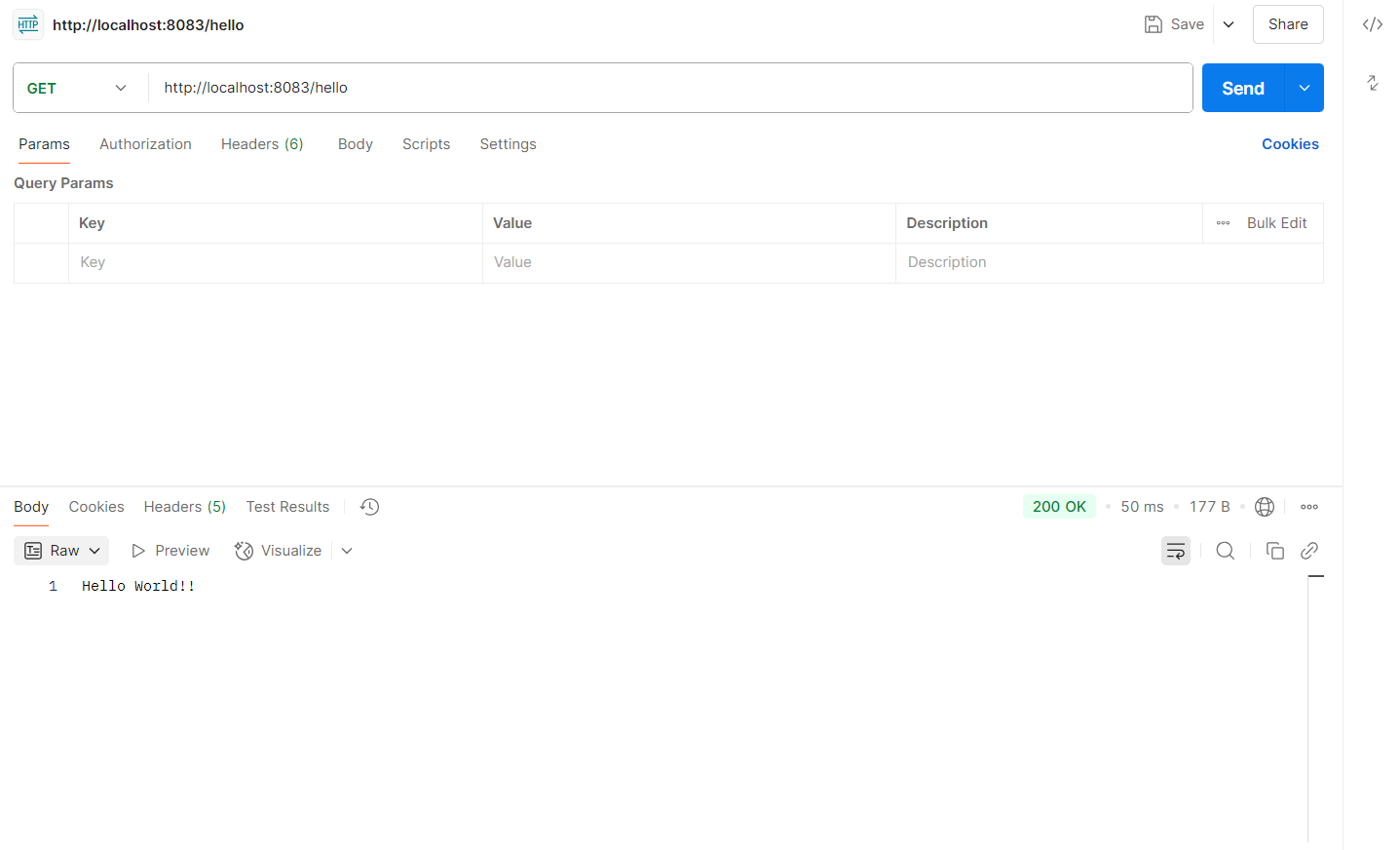
}

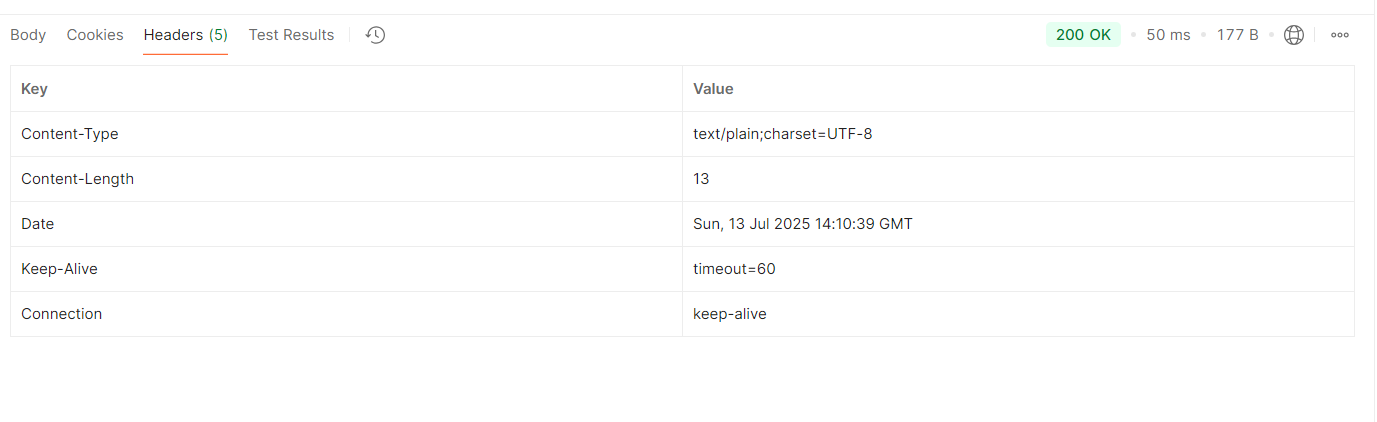
}

application.properties

server.port=8083

**Output: **



**Browser Network Tab:**

When we access http://localhost:8083/hello, the browser sends a GET request. The Network tab shows response headers like Content-Type, Status Code, and Content-Length, which confirm the server responded correctly.

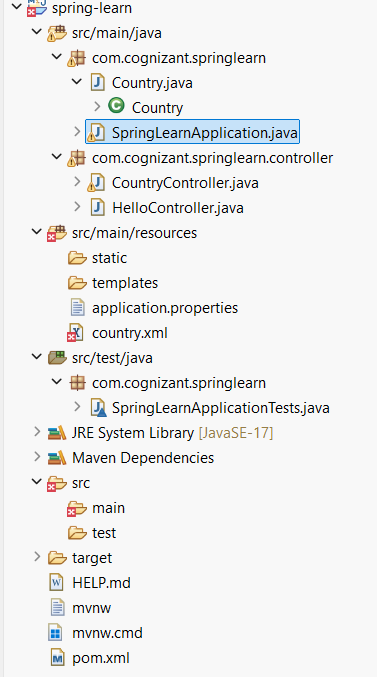
**Postman Headers Tab:**

In Postman, after sending a GET request to the same URL, the Headers tab shows metadata like Content-Type, Date, and Server, helping verify the response structure and format.

**4. REST - Country Web Service**

**Project Name: Spring-learn**

**Project Structure:**



**Code:**

**CountryController.java**

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.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 = context.getBean("country", Country.class);

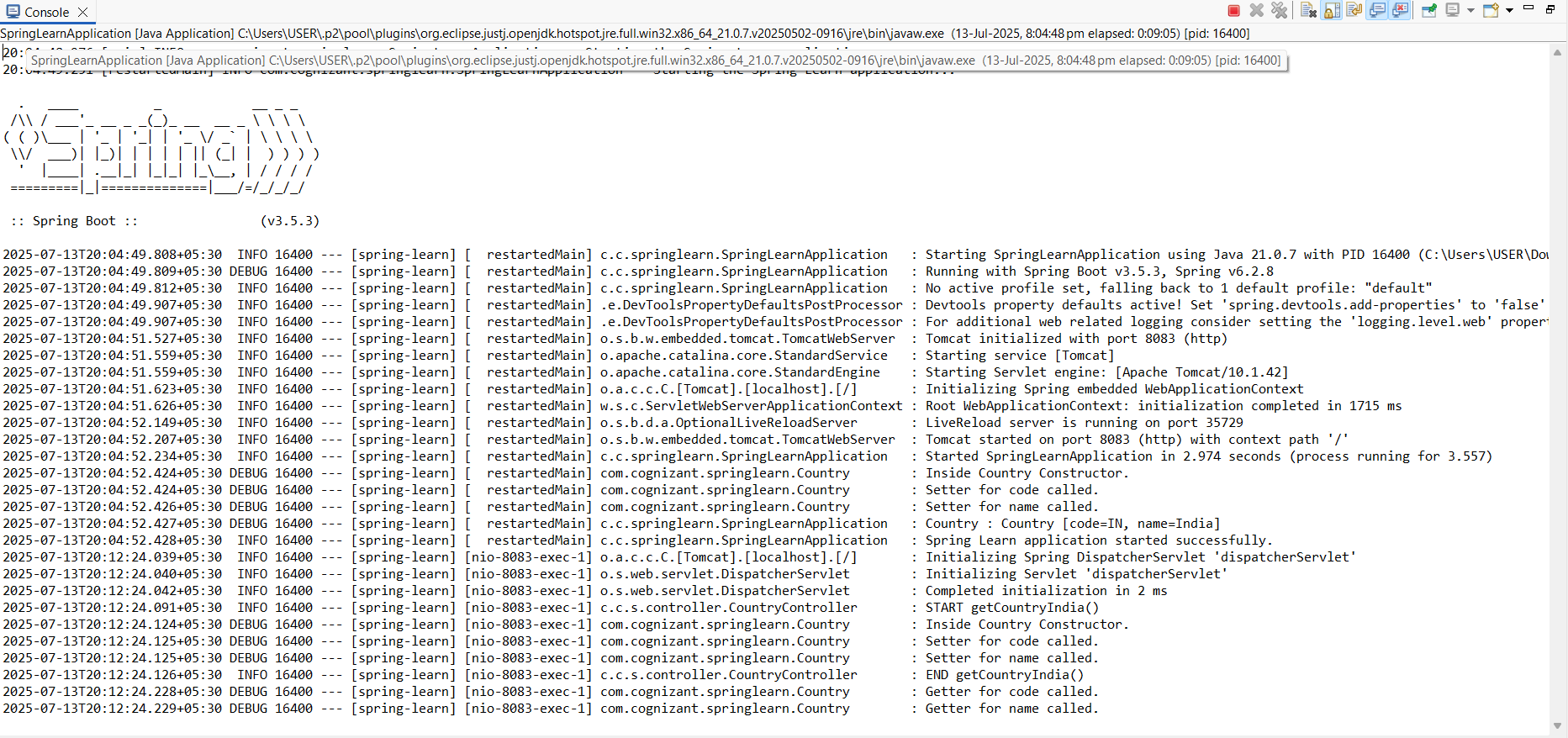
LOGGER.info("END getCountryIndia()");

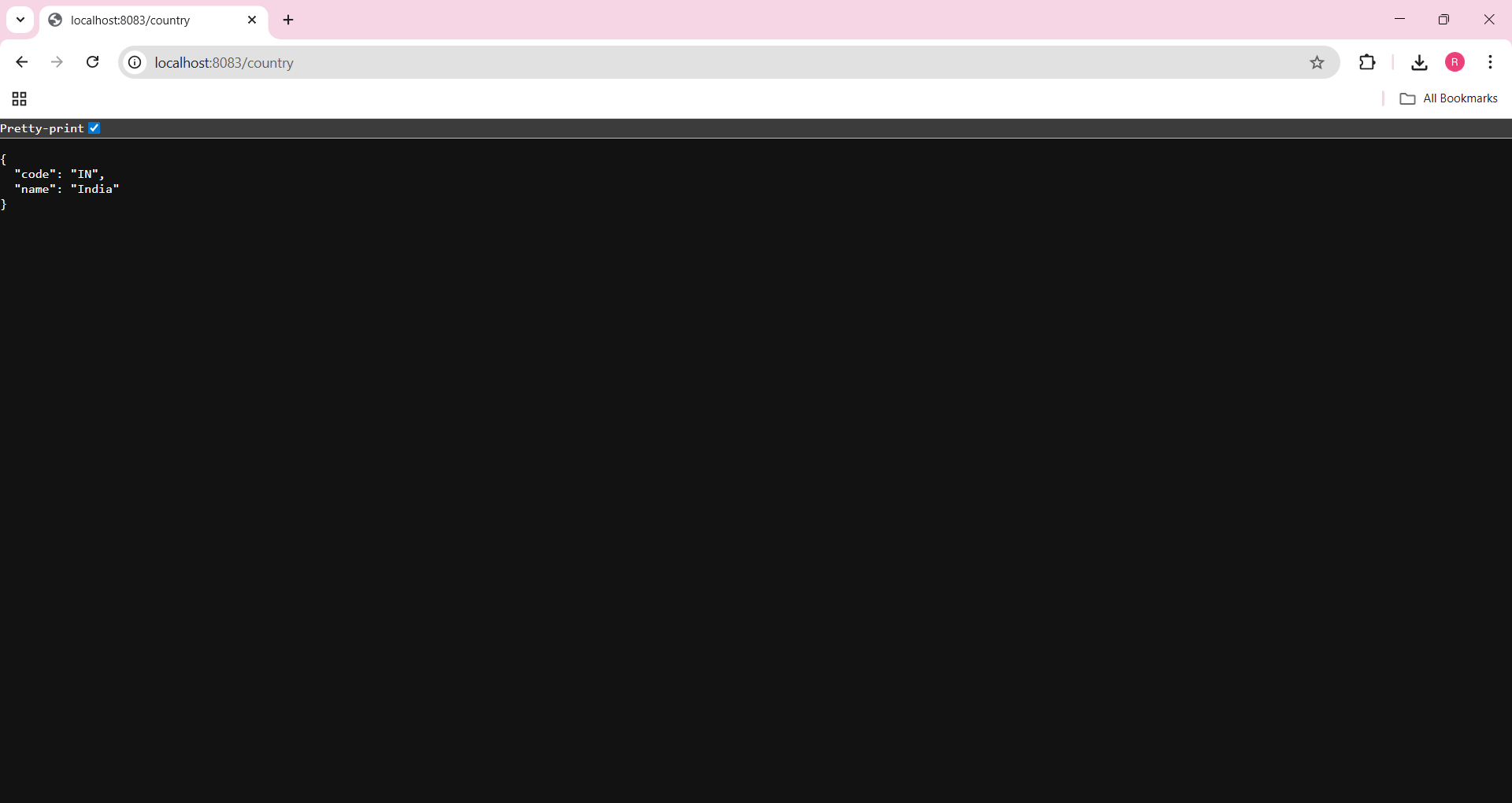
return country;

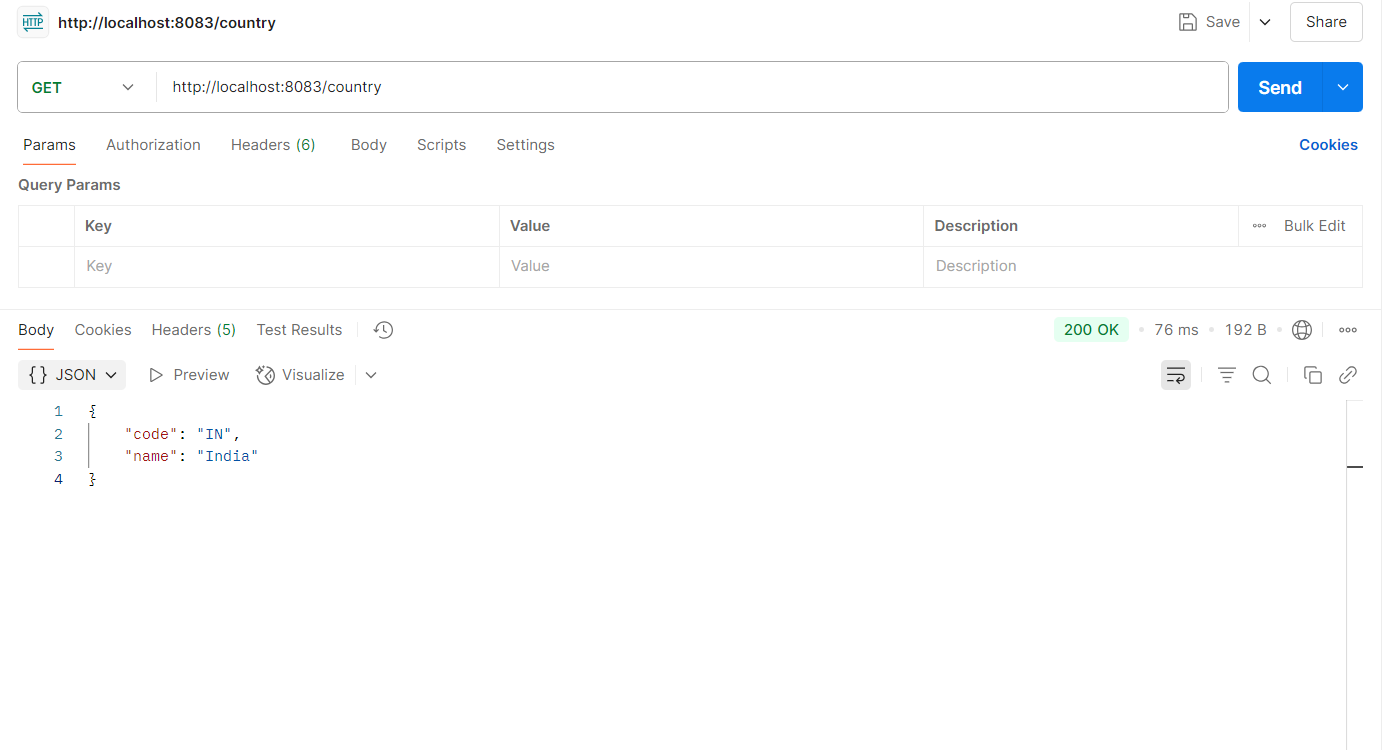
}

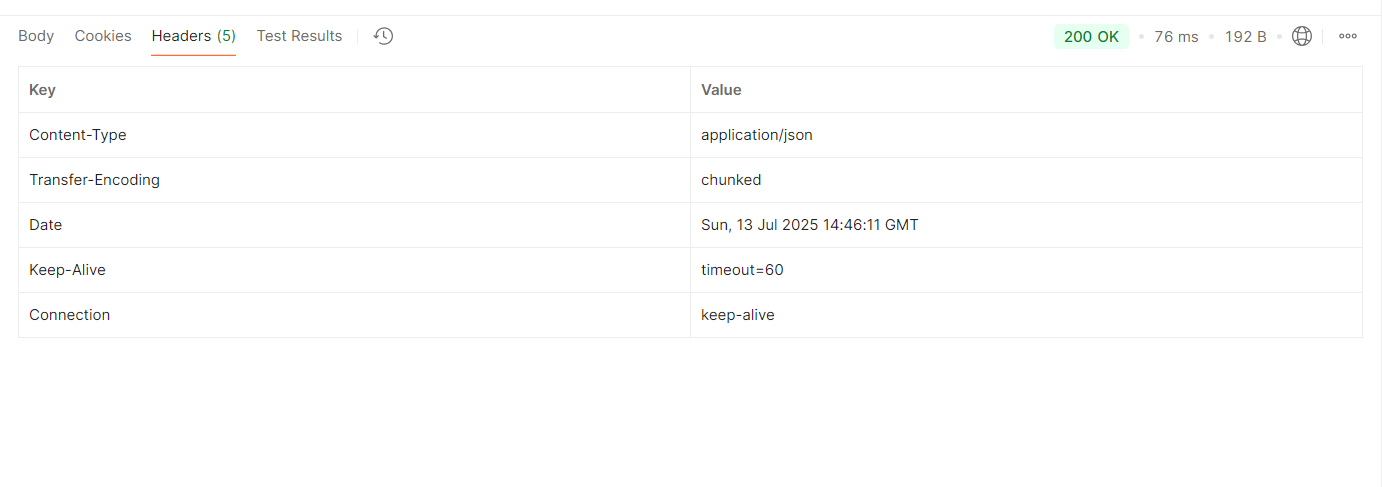
}

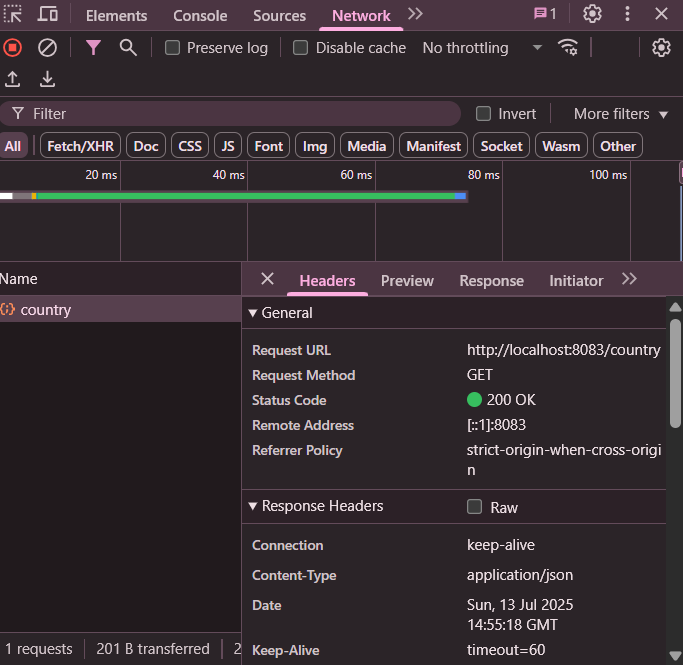
**Output:**

****







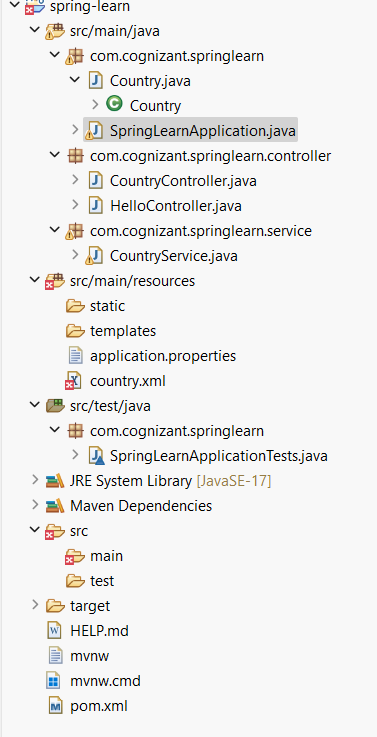


* **Controller Method**:  
  getCountryIndia() loads the country bean from country.xml using ApplicationContext and returns it.
* **JSON Conversion**:  
  Spring Boot uses Jackson to automatically convert the Country object into a JSON response.
* **Chrome Network Headers**:  
  Headers like Content-Type: application/json, Status: 200 OK, and others provide response metadata.
* **Postman Headers**:  
  After sending the request, Postman displays headers such as Content-Type, Content-Length, Server, and Date.

**5. REST - Get country based on country code**

**Project Name: Spring-learn**

**Project Structure:**

****

**Code:**

**CountryController.java**

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.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("/countries/{code}")

public Country getCountry(@PathVariable String code) {

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

Country country = countryService.getCountry(code);

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

return country;

}

}

**CountryService.java**

package com.cognizant.springlearn.service;

import com.cognizant.springlearn.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");

return countries.stream()

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

.findFirst()

.orElse(null);

}

}

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

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

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

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

</bean>

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

<constructor-arg>

<list>

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

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

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

</bean>

<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>

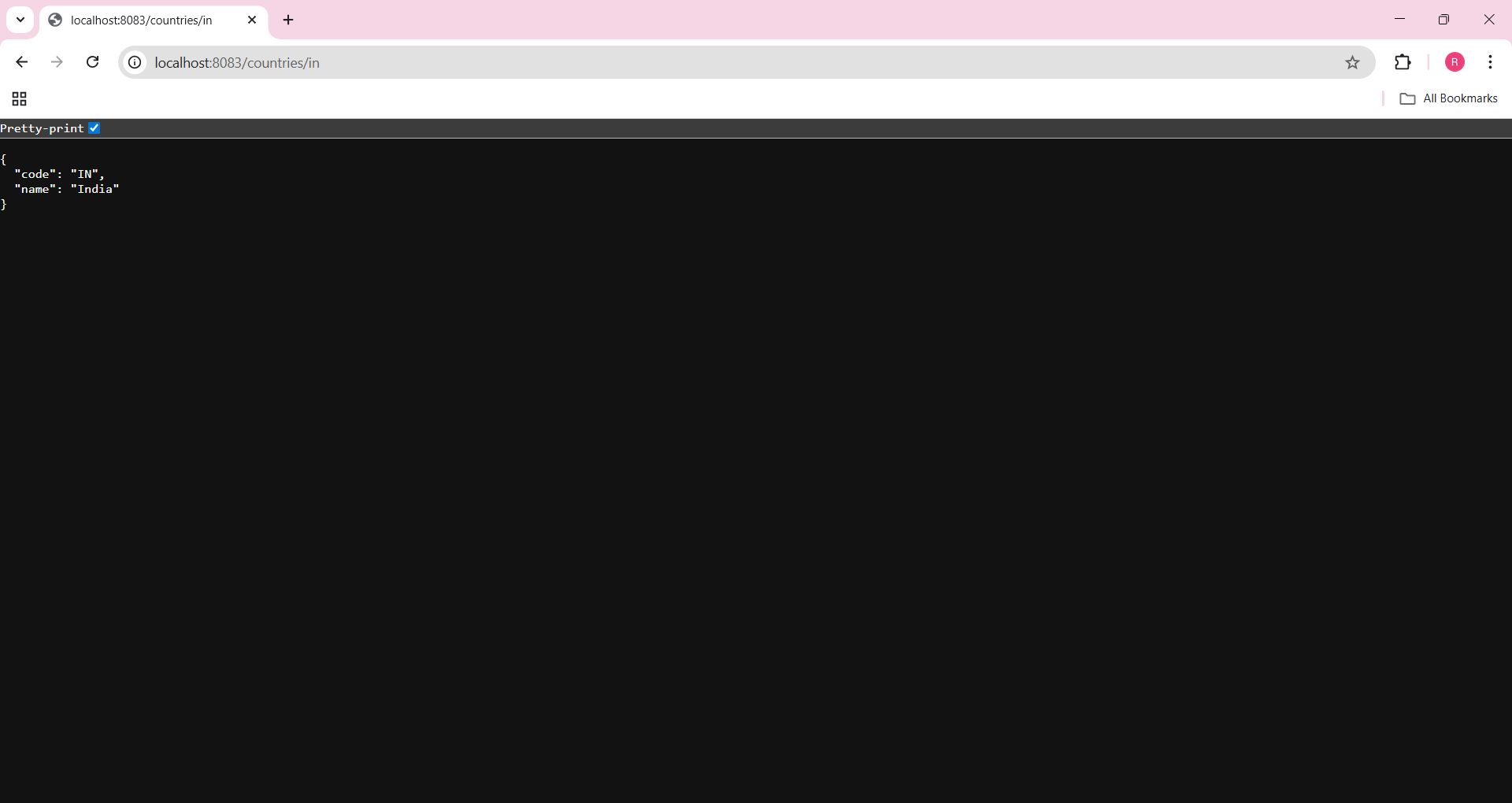
</list>

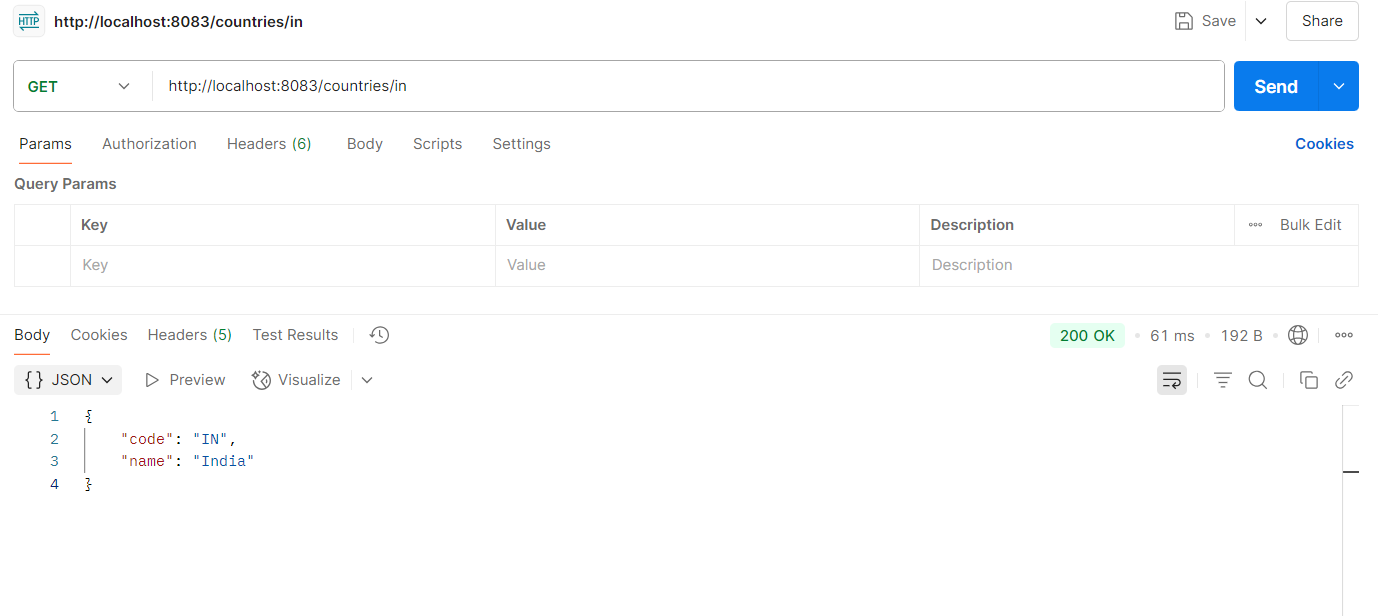
</constructor-arg>

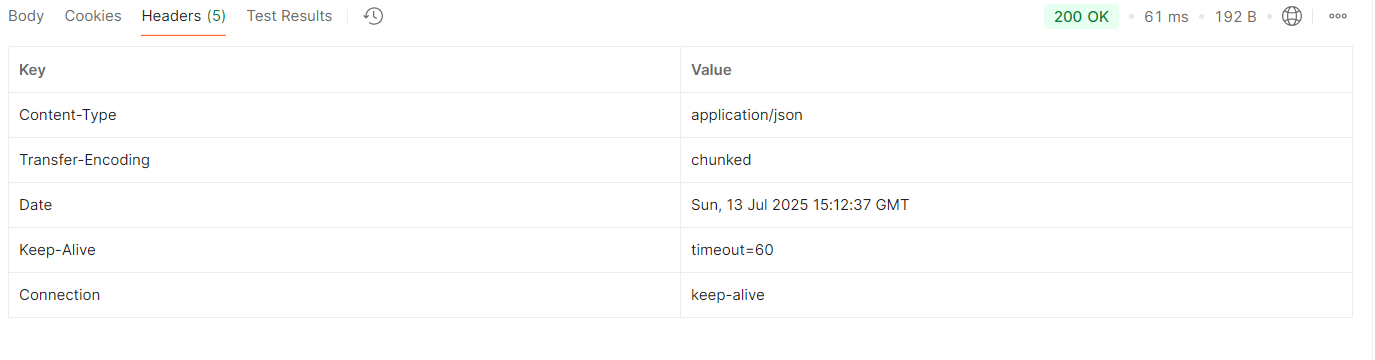
</bean>

</beans>

**Output:**

****





**Controller Method:** Extracts the country code from the URL and calls the getCountry() service method to return the matching Country object.

**JSON Conversion:** Spring Boot uses Jackson to auto-convert the Country object into a JSON response.

**Headers (Chrome & Postman):**

* Content-Type: application/json
* Status: 200 OK
* Date, Server, Content-Length, etc.