**REST - Country Web Service**

**Code –**

*CountryController.java –*

package com.cognizant.springlearn.controller;

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.RequestMethod;

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

import com.cognizant.springlearn.Country;

*@RestController*

public class CountryController {

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

*@RequestMapping*(value = "/country", method = *RequestMethod*.***GET***)

public Country getCountryIndia() {

***LOGGER***.debug("Start - getCountryIndia()");

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

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

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

return country;

}

}

*SpringlearnApplication.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);

}

}

*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("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 + "]";

}

}

*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**>

*Logback.xml –*

<**configuration**>

<**appender** name=*"STDOUT"* class=*"ch.qos.logback.core.ConsoleAppender"*>

<**encoder**>

<**pattern**>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</**pattern**>

</**encoder**>

</**appender**>

<**root** level=*"debug"*>

<**appender-ref** ref=*"STDOUT"*/>

</**root**>

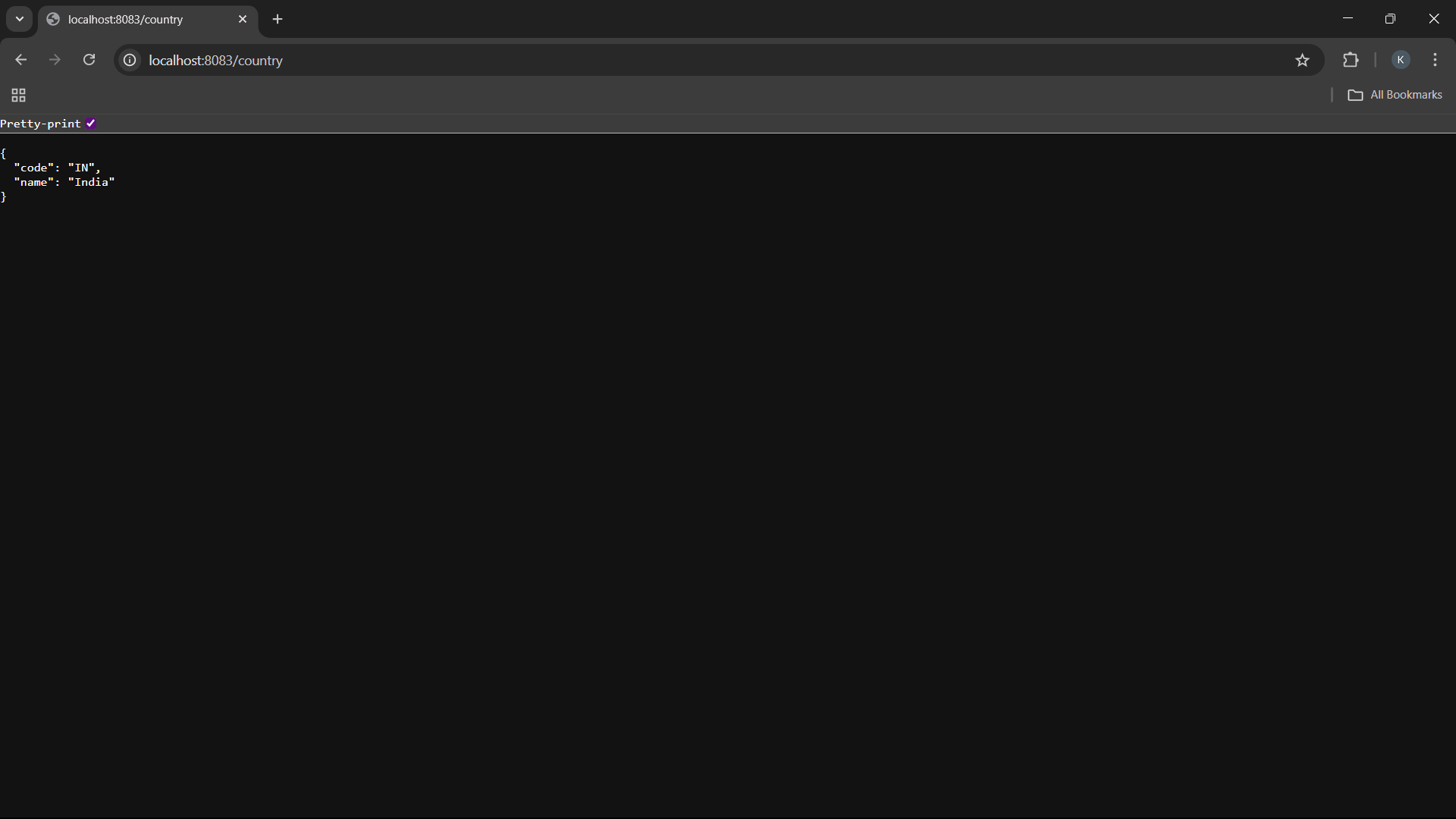
</**configuration**>

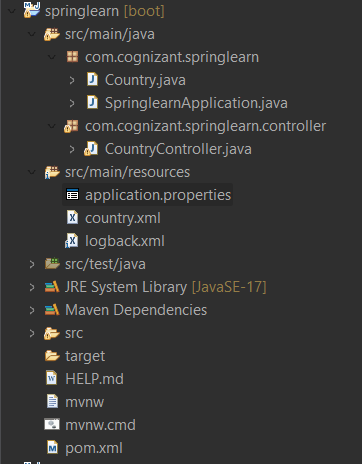
*Application.properties –*

spring.application.name=springlearn

server.port=8083

**Output – (Screenshots)**



****

**SME –**

* **What happens in the controller method?**

**Answer -** When the user hits the URL http://localhost:8083/country, the following happens:

Lifecycle:

1. Spring Boot handles the incoming HTTP GET request to /country.
2. It matches the request to this controller method:

@RequestMapping(value = "/country", method = RequestMethod.GET)

public Country getCountryIndia() { ... }

1. The Country bean (either autowired or loaded from XML) is returned from the method.
2. Spring uses **HttpMessageConverter** (Jackson by default) to convert the Country object into **JSON**.
3. The JSON is sent back in the **HTTP Response Body**.

* **How the bean is converted into JSON reponse?**

**Answer -** Spring Boot uses Jackson (a JSON library) which is included automatically via:

<dependency>

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

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

</dependency>

**Behind the scenes:**

* @RestController tells Spring to skip view rendering and instead return **data** (usually as JSON).
* Jackson looks at your Country class (using getters) and serializes it into JSON.
* For example: Country country = new Country("IN", "India");

is converted to:

{

"code": "IN",

"name": "India"

}

This happens automatically because of **Spring's HttpMessageConverter chain**.

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

**Answer** - You can inspect the response headers using:

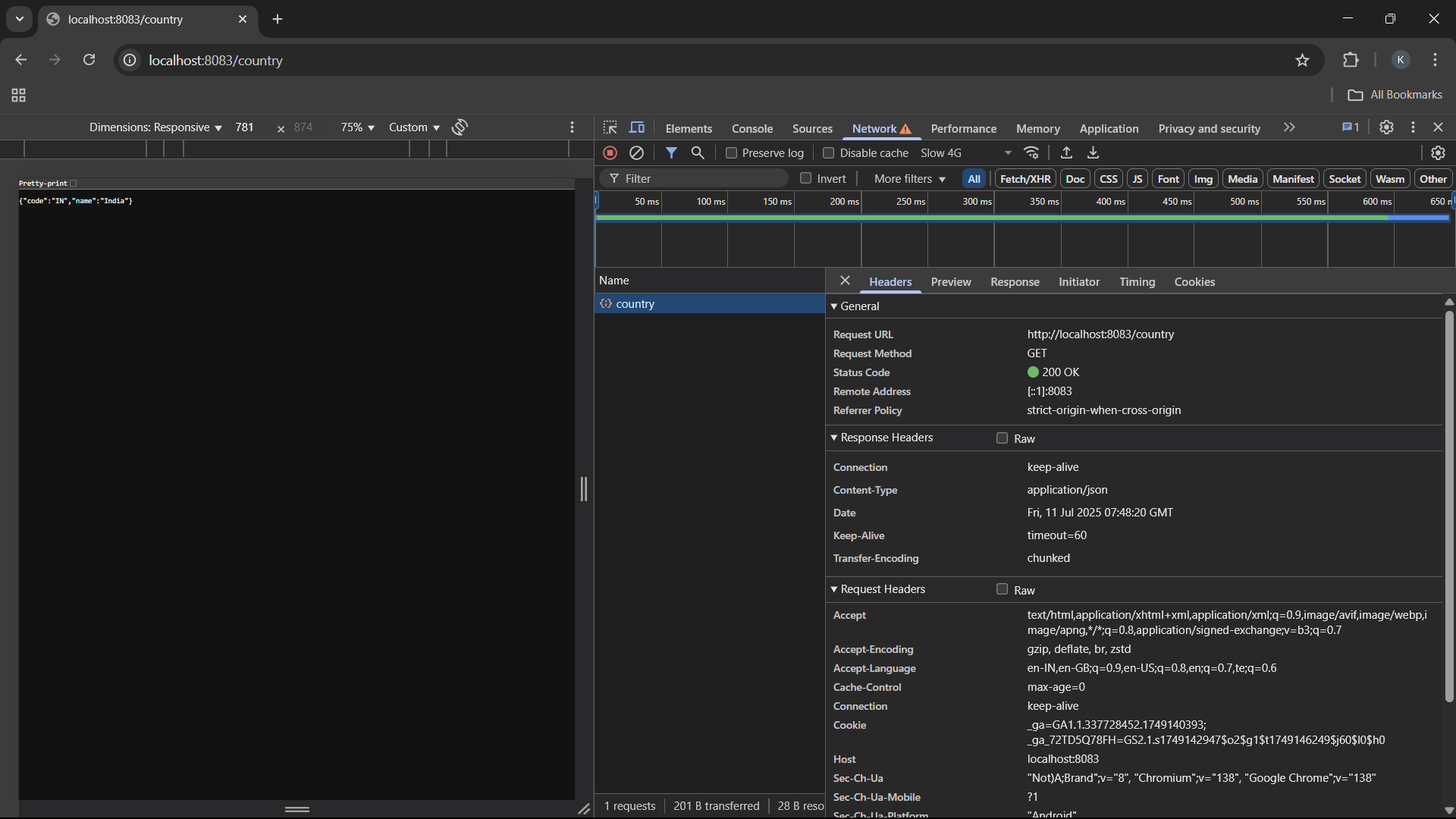
**Steps (in Chrome):**

1. Right-click > Inspect → Open Developer Tools.
2. Go to the Network tab.
3. Hit your endpoint in browser: http://localhost:8083/country
4. Click on the network request (usually labeled country).
5. View the Headers tab:
   * Request Headers: Info sent from browser (like User-Agent, Accept).
   * Response Headers: Info sent from your Spring Boot app, such as:

Content-Type: application/json

Content-Length: 34

Date: Thu, 11 Jul 2025 07:35:12 GMT



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

**Answer –**

**Steps:**

1. Open Postman.
2. Send a GET request to: http://localhost:8083/country
3. Go to the Headers tab (right next to "Body" and "Cookies").
4. You'll see:
   * Content-Type: application/json (returned from Spring)
   * Content-Length
   * Date
   * Server: Apache Tomcat

These headers confirm that the response is valid JSON, sent over HTTP correctly.

