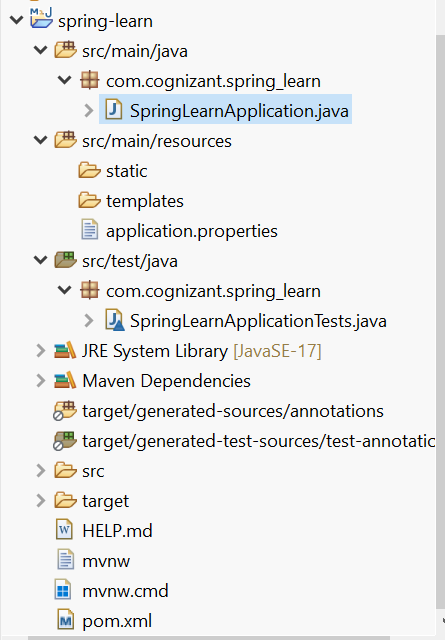
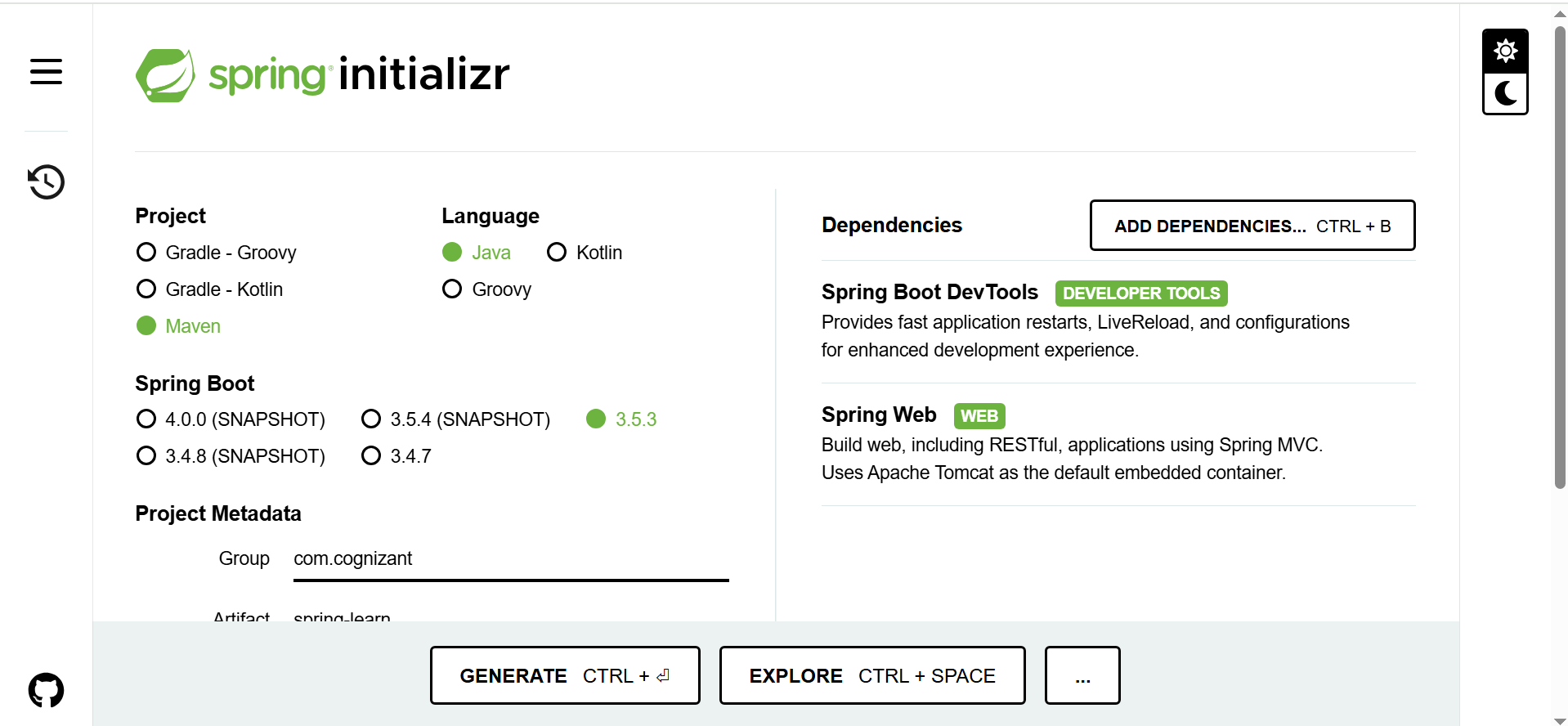
**1) Create a Spring Web Project using Maven**   
  
Follow steps below to create a project: 

1. Go to <https://start.spring.io/>
2. Change Group as “com.cognizant”
3. Change Artifact Id as “spring-learn”
4. Select Spring Boot DevTools and Spring Web
5. Create and download the project as zip
6. Extract the zip in root folder to Eclipse Workspace
7. Build the project using ‘mvn clean package -Dhttp.proxyHost=proxy.cognizant.com -Dhttp.proxyPort=6050 -Dhttps.proxyHost=proxy.cognizant.com -Dhttps.proxyPort=6050 -Dhttp.proxyUser=123456’ command in command line
8. Import the project in Eclipse "File > Import > Maven > Existing Maven Projects > Click Browse and select extracted folder > Finish"
9. Include logs to verify if main() method of SpringLearnApplication.
10. Run the SpringLearnApplication class.

**Project Structure:**

****

**Spring initializr:**

****

**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>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

**SpringLearnApplication.java:**

**package** com.cognizant.spring\_learn;

**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) {

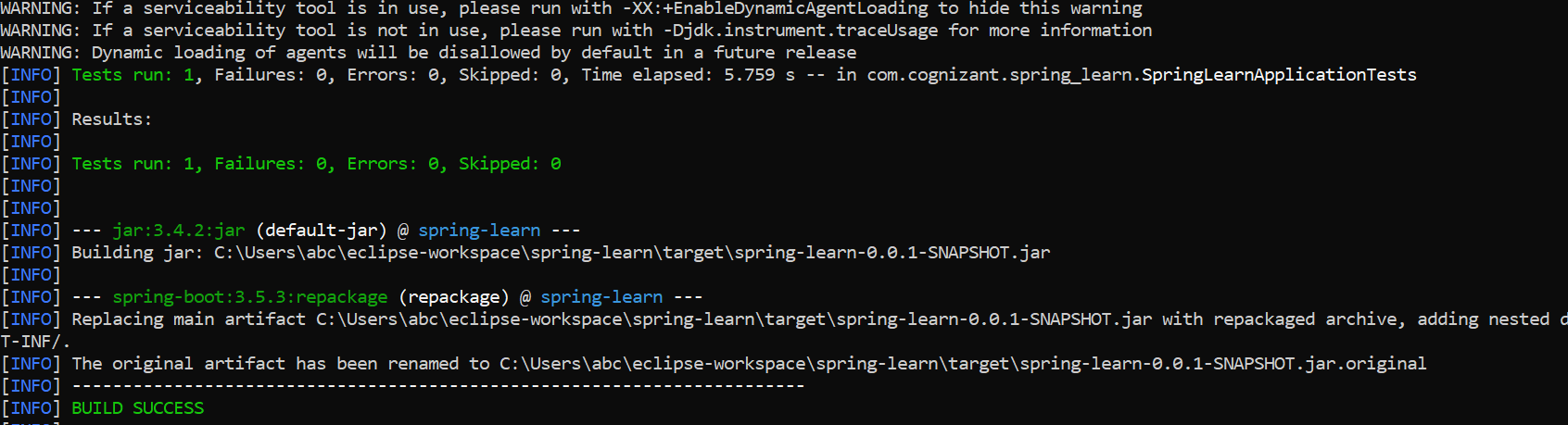
***LOGGER***.info("currently in main() method");

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

}

}

**Output:**

****



**application.properties:**

spring.application.name=spring-learn

**Purpose of @SpringBootApplication annotation**:

@SpringBootApplication is a shortcut annotation that combines @Configuration, @EnableAutoConfiguration, and @ComponentScan.  
It enables Spring Boot to auto-configure the application based on dependencies and scan for components.  
This annotation is typically placed on the main class to bootstrap the app.

**2) Spring Core – Load Country from Spring Configuration XML**   
  
An airlines website is going to support booking on four countries. There will be a drop down on the home page of this website to select the respective country. It is also important to store the two-character ISO code of each country. 

|  |  |
| --- | --- |
| **Code** | **Name** |
| US | United States |
| DE | Germany |
| IN | India |
| JP | Japan |

Above data has to be stored in spring configuration file. Write a program to read this configuration file and display the details.  
  
Steps to implement

* Pick any one of your choice country to configure in Spring XML configuration named country.xml.
* Create a bean tag in spring configuration for country and set the property and values

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

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

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

    </bean>

* Create Country class with following aspects:
  + Instance variables for code and name
  + Implement empty parameter constructor with inclusion of debug log within the constructor with log message as “Inside Country Constructor.”
  + Generate getters and setters with inclusion of debug with relevant message within each setter and getter method.
  + Generate toString() method
* Create a method displayCountry() in SpringLearnApplication.java, which will read the country bean from spring configuration file and display the country details. ClassPathXmlApplicationContext, ApplicationContext and context.getBean(“beanId”, Country.class). Refer sample code for displayCountry() method below.

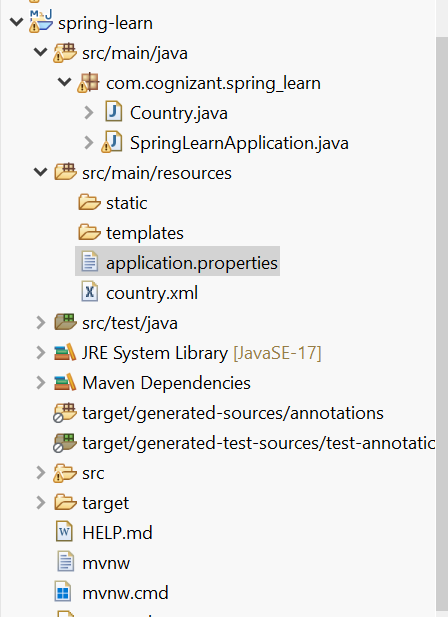
ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");

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

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

* Invoke displayCountry() method in main() method of SpringLearnApplication.java.
* Execute main() method and check the logs to find out which constructors and methods were invoked.

**Project Structure:**

****

**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.spring\_learn.Country"*>

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

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

</bean>

</beans>

**Code:**

**Country.java**

**package** com.cognizant.spring\_learn;

**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 Constructor.");

}

**public** String getCode() {

***LOGGER***.debug("getCode method called");

**return** code;

}

**public** **void** setCode(String code) {

***LOGGER***.debug("setCode method called with {}", code);

**this**.code = code;

}

**public** String getName() {

***LOGGER***.debug("getName method called");

**return** name;

}

**public** **void** setName(String name) {

***LOGGER***.debug("setName method called with {}", name);

**this**.name = name;

}

@Override

**public** String toString() {

**return** String.*format*("Country [code=%s, name=%s]", code, name);

}

}

**SpringLearnApplication.java:**

**package** com.cognizant.spring\_learn;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

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

@SpringBootApplication

**public** **class** SpringLearnApplication {

**private** **static** **final** Logger ***LOGGER*** =

LoggerFactory.*getLogger*(SpringLearnApplication.**class**);

**public** **static** **void** main(String[] args) {

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

*displayCountry*();

}

**private** **static** **void** displayCountry() {

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

ApplicationContext context =

**new** ClassPathXmlApplicationContext("country.xml");

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

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

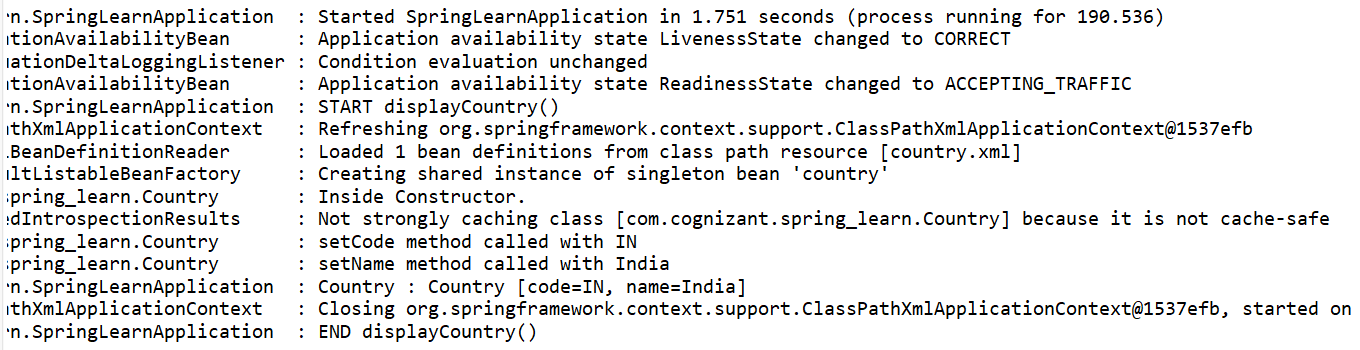
((ClassPathXmlApplicationContext) context).close();

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

}

}

**Output:**



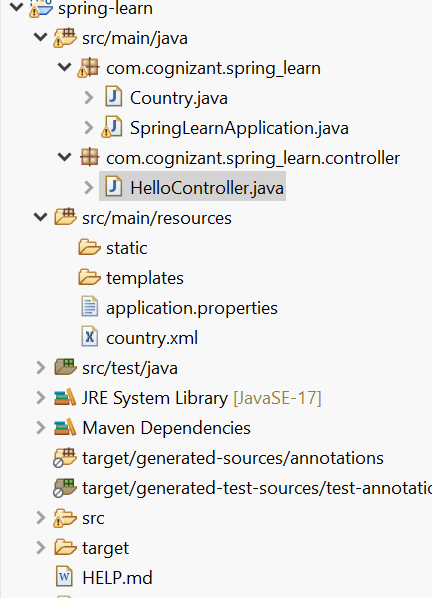
When context.getBean() is invoked, the Spring container looks up the bean definition using the provided ID or class. If the bean is a singleton and hasn't been created yet, Spring instantiates it, injects any required dependencies, and performs any initialization steps defined (e.g., constructor, setters, init-method). Finally, the fully initialized bean is returned for use in the application.

**3) Hello World RESTful Web Service**

Write a REST service in the spring learn application created earlier, that returns the text "Hello World!!" using Spring Web Framework.

Refer details below:  
  
**Method:** GET  
**URL:** /hello  
**Controller:** com.cognizant.spring-learn.controller.HelloController  
**Method Signature:** public String sayHello()  
**Method Implementation:** return hard coded string "Hello World!!"  
**Sample Request**: http://localhost:8083/hello  
**Sample Response:** Hello World!!   
  
**IMPORTANT NOTE**: Don't forget to include start and end log in the sayHello() method.  
  
Try the URL http://localhost:8083/hello in both chrome browser and postman.

**Project Structure**:



**Code:**

**HelloController.java**:

**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()");

String response = "Hello World!!";

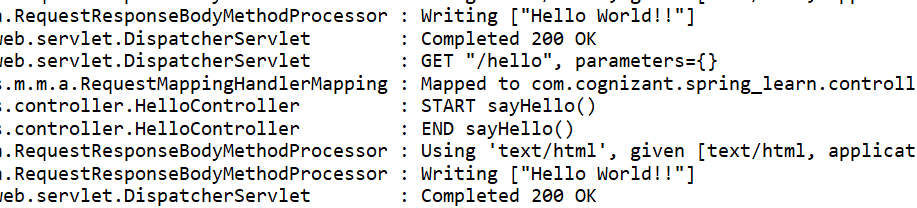
***LOGGER***.debug("END sayHello()");

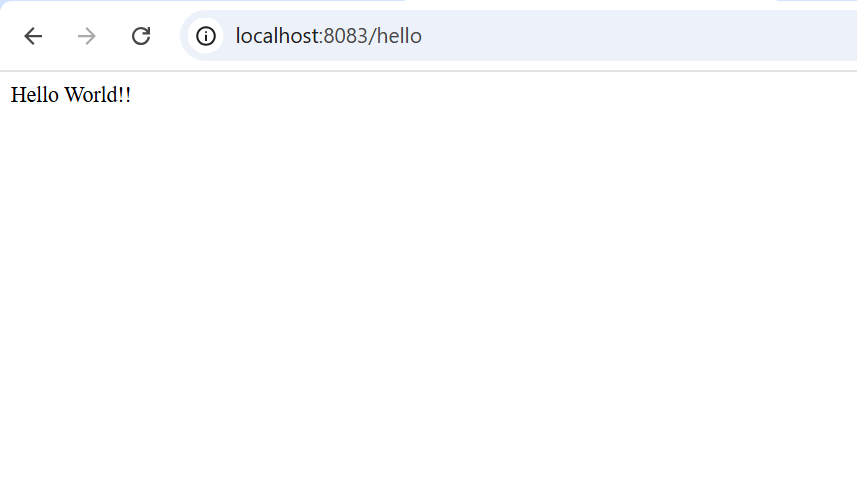
**return** response;

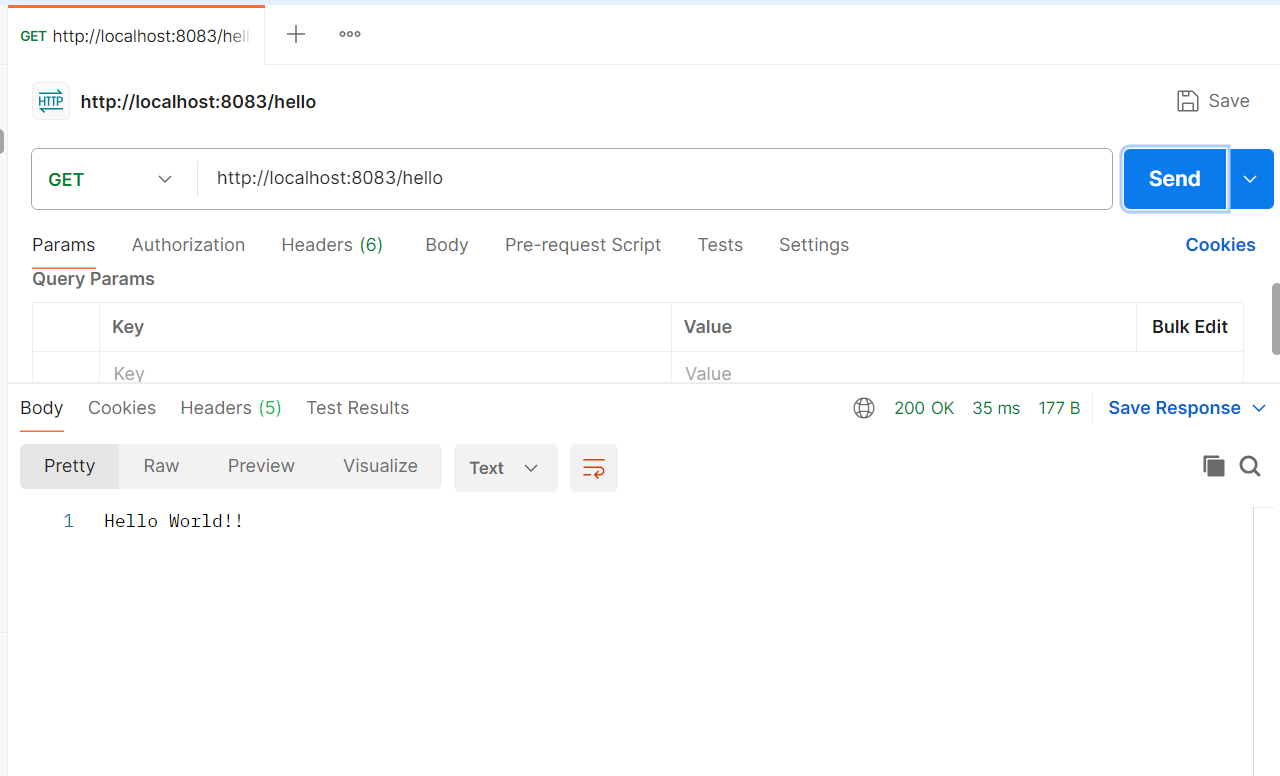
}

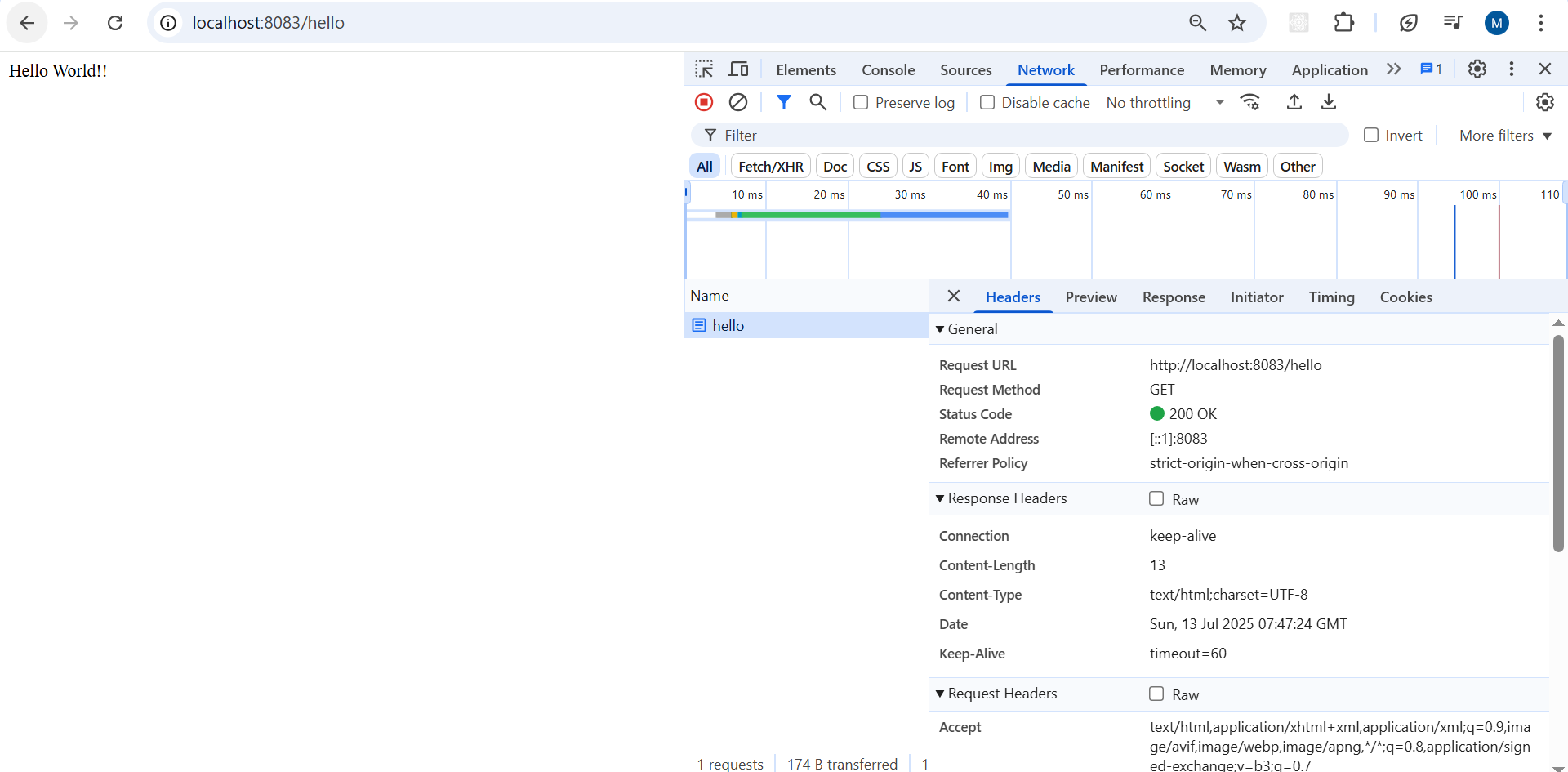
}

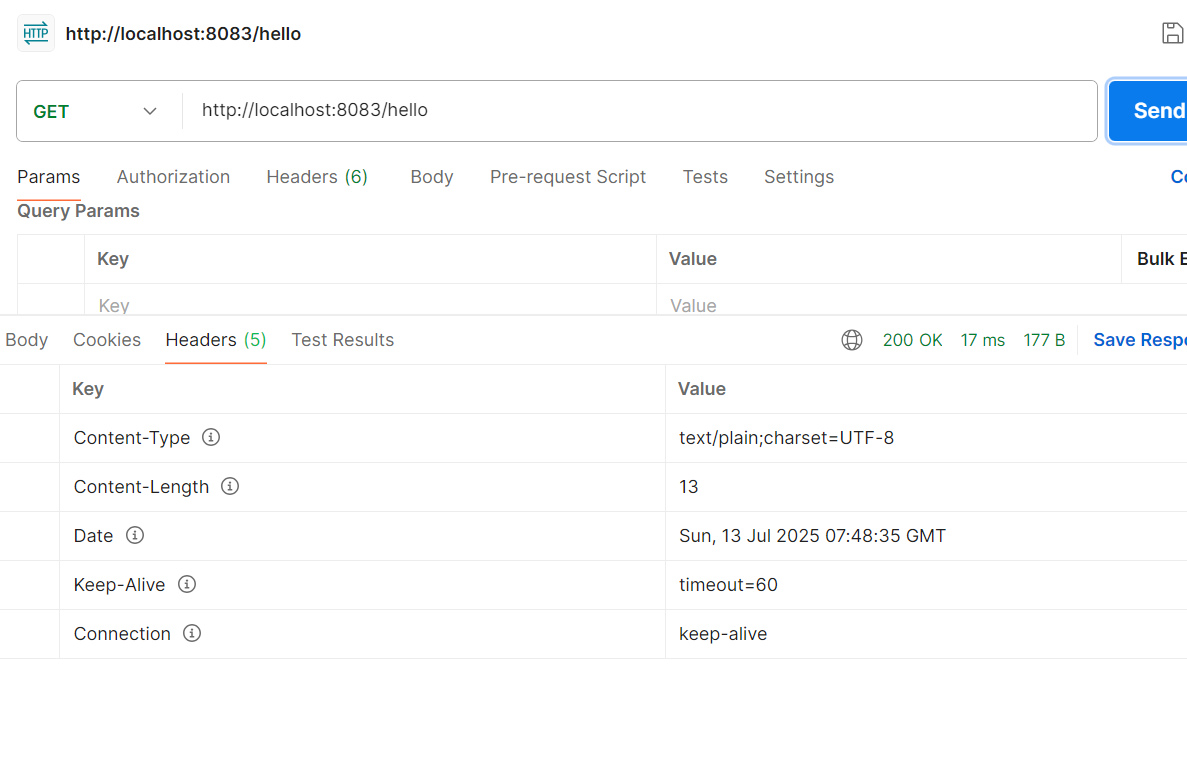
**Output:**

****

****



****

****

**4) REST - Country Web Service**

Write a REST service that returns India country details in the earlier created spring learn application.  
**URL**: /country  
**Controller**: com.cognizant.spring-learn.controller.CountryController  
**Method Annotation**: @RequestMapping  
**Method Name**: getCountryIndia()  
**Method Implementation**: Load India bean from spring xml configuration and return  
**Sample Request**: http://localhost:8083/country  
**Sample Response**:

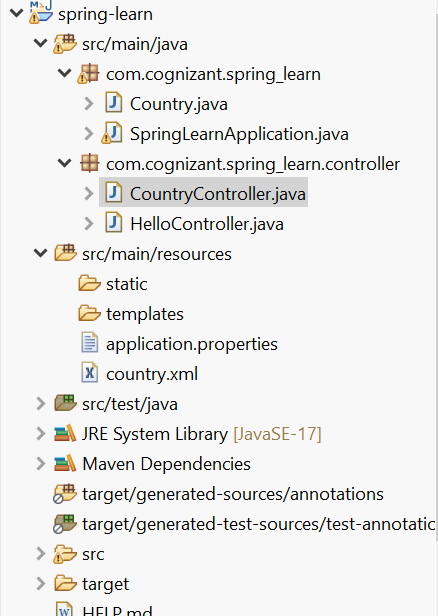
{

  "code": "IN",

  "name": "India"

}

**Project Structure**:



**Code:**

**CountryController.java:**

**package** com.cognizant.spring\_learn.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.spring\_learn.Country;

@RestController

**public** **class** CountryController {

**private** **static** **final** Logger ***LOGGER*** =

LoggerFactory.*getLogger*(CountryController.**class**);

@RequestMapping(value = "/country",

method = RequestMethod.***GET***,

produces = "application/json")

**public** Country getCountryIndia() {

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

// loading the XML, fetching the bean

ApplicationContext context =

**new** ClassPathXmlApplicationContext("country.xml");

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

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

((ClassPathXmlApplicationContext) context).close();

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

**return** country;

}

}

**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.spring\_learn.Country"*>

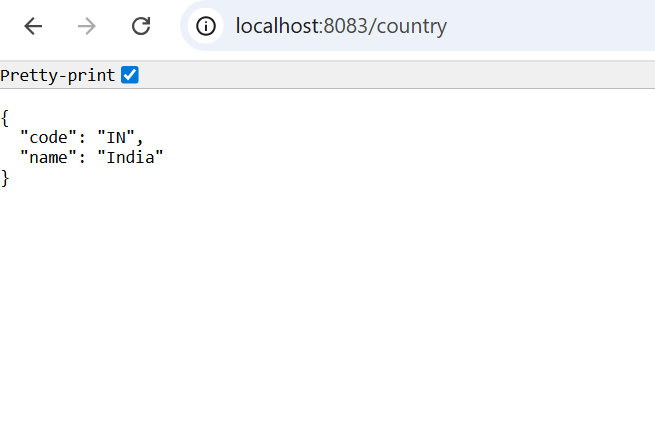
<property name=*"code"* value=*"IN"*/>

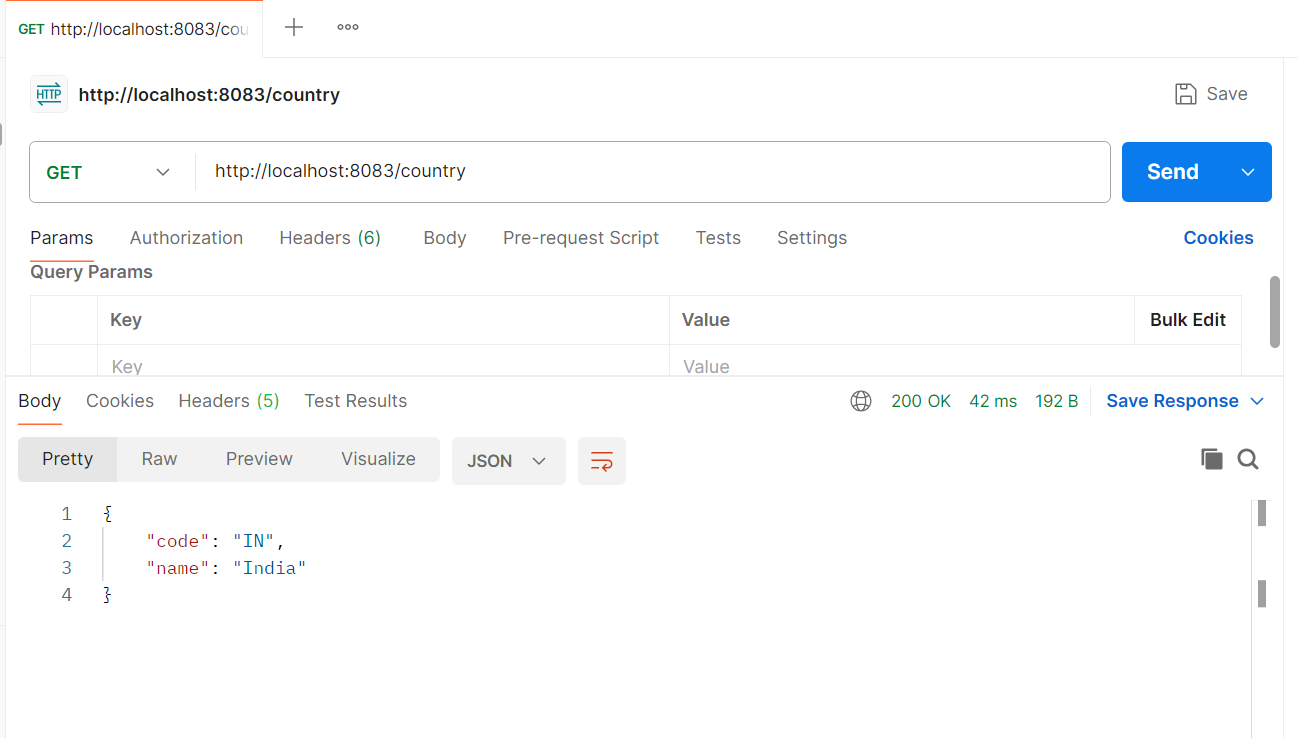
<property name=*"name"* value=*"India"*/>

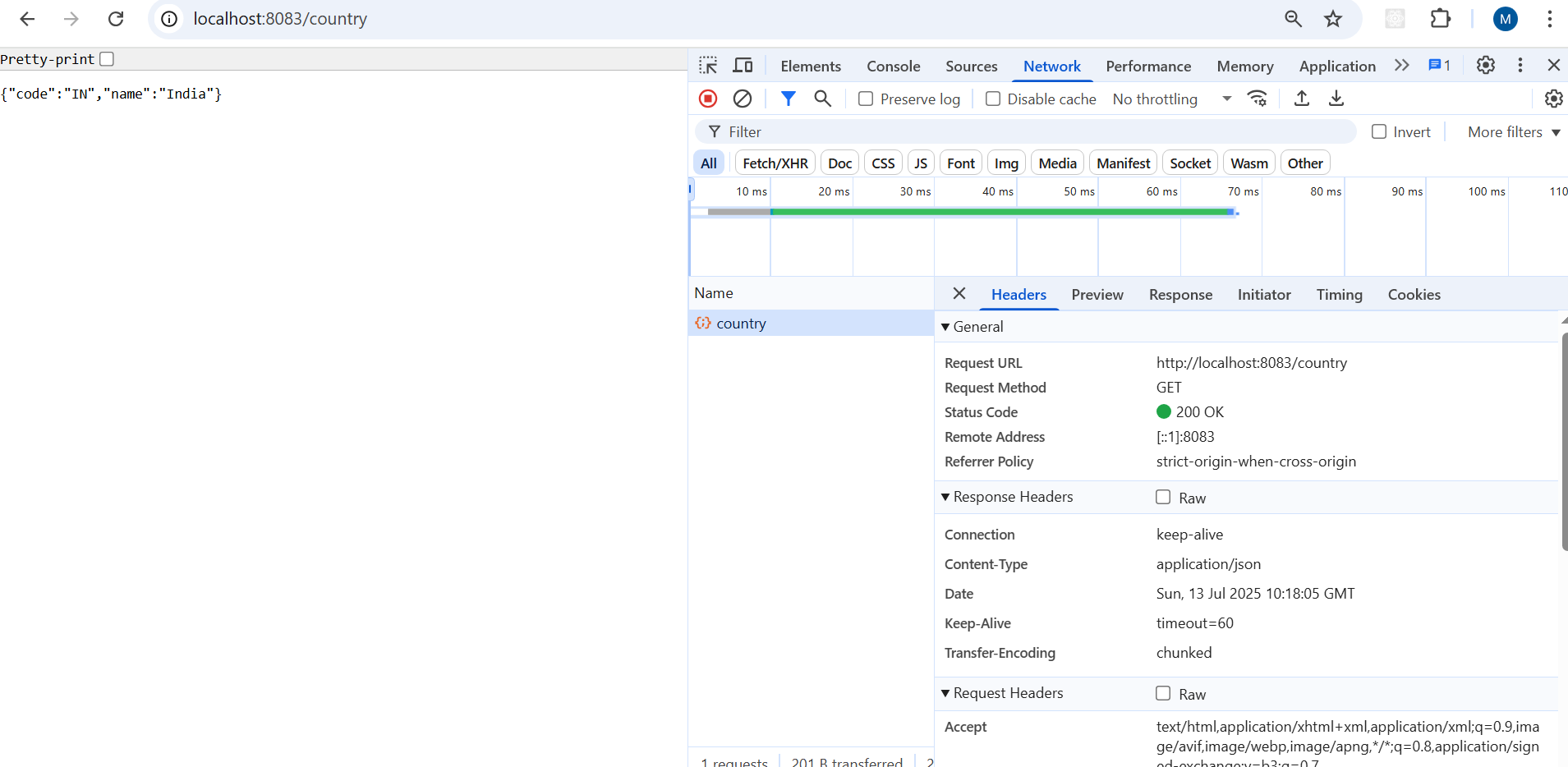
</bean>

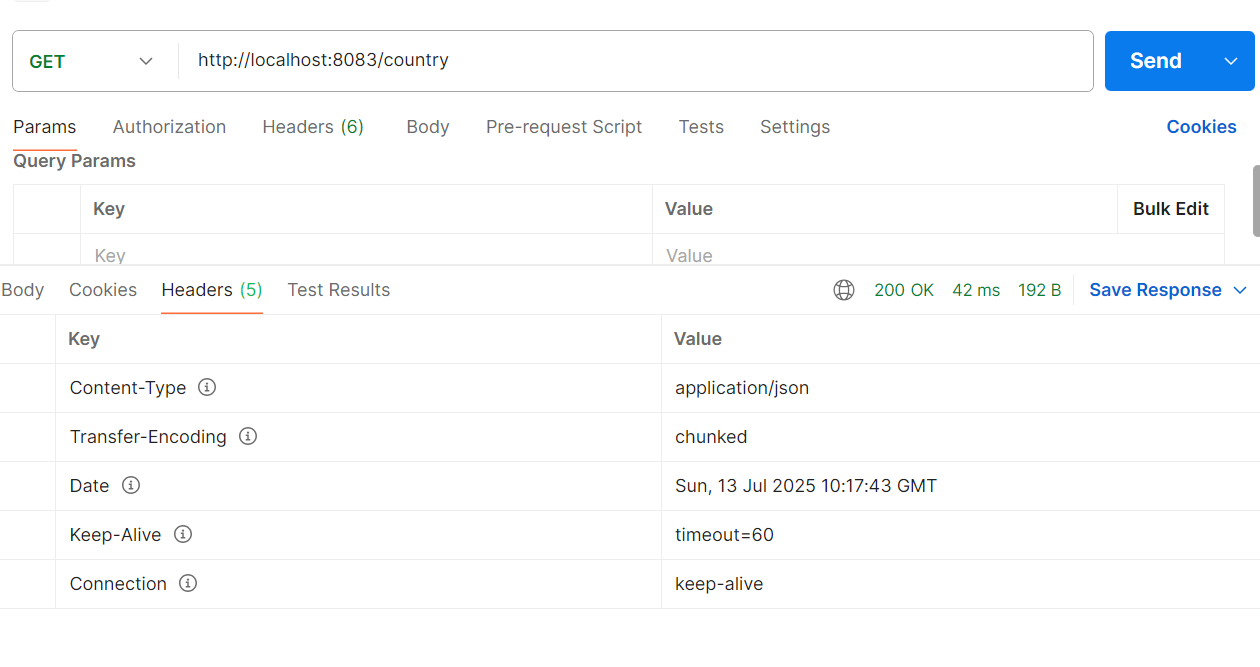
</beans>

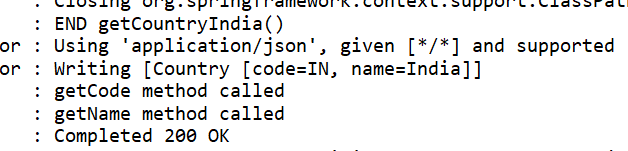
**Output:**











When the controller method getCountryIndia() is called, it logs the start of the method and then loads the Spring XML configuration file (country.xml) using ClassPathXmlApplicationContext. It retrieves the country bean defined in the XML, which contains India's details. The method logs the retrieved bean and returns it, after which the context is closed and the method ends.

Spring Boot uses **Jackson**, a built-in JSON converter in the spring-boot-starter-web dependency. When a Java object like Country is returned from a controller method annotated with @RestController, Spring automatically uses Jackson to serialize the object into JSON format. The resulting JSON is sent back as the HTTP response body with Content-Type as application/json. This conversion happens seamlessly with no manual code needed.

**5) REST - Get country based on country code**   
  
Write a REST service that returns a specific country based on country code. The country code should be case insensitive.  
  
**Controller**: com.cognizant.spring-learn.controller.CountryController  
**Method Annotation:** @GetMapping("/countries/{code}")  
**Method Name**: getCountry(String code)  
**Method Implemetation**: Invoke countryService.getCountry(code)   
**Service Method:**com.cognizant.spring-learn.service.CountryService.getCountry(String code)  
  
**Service Method Implementation**:

* Get the country code using @PathVariable
* Get country list from country.xml
* Iterate through the country list
* Make a case insensitive matching of country code and return the country.
* Lambda expression can also be used instead of iterating the country list

**Sample Request**: http://localhost:8083/country/in  
  
**Sample Response**:

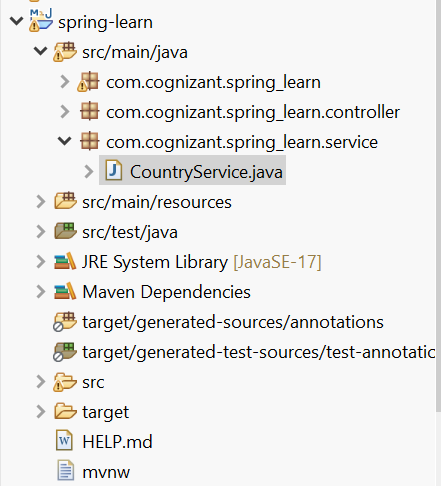
{

  "code": "IN",

  "name": "India"

}

**Project Structure:**

****

**Code:**

**CountryService.java:**

**package** com.cognizant.spring\_learn.service;

**import** java.util.List;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**import** org.springframework.stereotype.Service;

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

@Service

**public** **class** CountryService {

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

**private** **static** **final** ApplicationContext ***CONTEXT*** =

**new** ClassPathXmlApplicationContext("country.xml");

@SuppressWarnings("unchecked")

**public** Country getCountry(String code) {

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

List<Country> list = (List<Country>) ***CONTEXT***.getBean("countryList");

Country result = list.stream()

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

.findFirst()

.orElse(**null**);

***LOGGER***.debug("END getCountry() → {}", result);

**return** result;

}

}

**CountryController.java:**

**package** com.cognizant.spring\_learn.controller;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

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

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

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

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

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

**import** com.cognizant.spring\_learn.service.CountryService;

@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);

Country country = countryService.getCountry(code);

***LOGGER***.debug("END getCountry() → {}", country);

**return** country;

}

}

**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=*"countryUS"* class=*"com.cognizant.spring\_learn.Country"*>

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

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

</bean>

<bean id=*"countryDE"* class=*"com.cognizant.spring\_learn.Country"*>

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

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

</bean>

<bean id=*"countryIN"* class=*"com.cognizant.spring\_learn.Country"*>

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

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

</bean>

<bean id=*"countryJP"* class=*"com.cognizant.spring\_learn.Country"*>

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

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

</bean>

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

<constructor-arg>

<list>

<ref bean=*"countryUS"*/>

<ref bean=*"countryDE"*/>

<ref bean=*"countryIN"*/>

<ref bean=*"countryJP"*/>

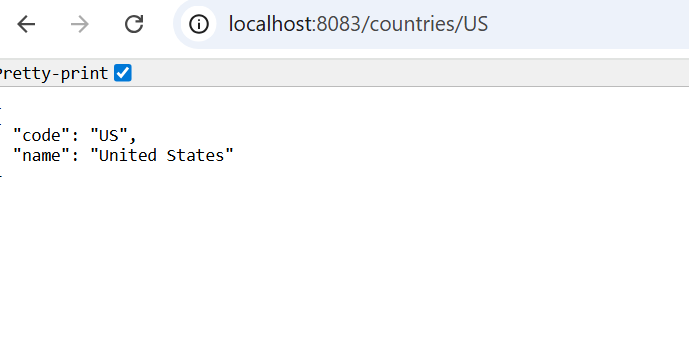
</list>

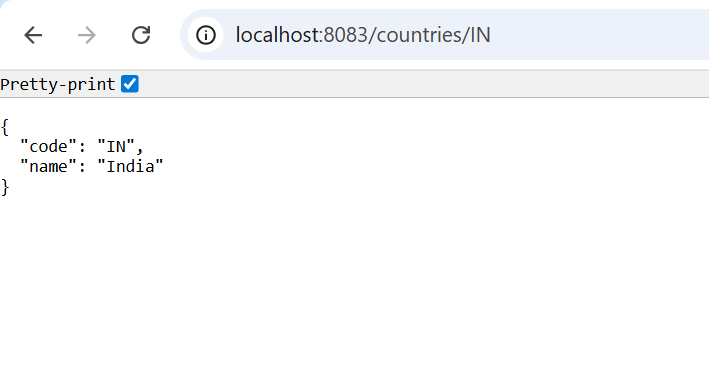
</constructor-arg>

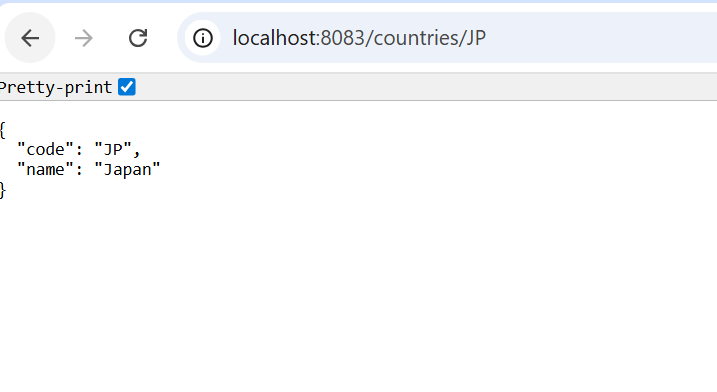
</bean>

</beans>

**Output:**

****

****

****