**Hands-On 1**

**Creating a Spring Web Project using Maven**

**Objective:** To create and run a basic Spring Boot Web project using Maven in Eclipse IDE.

**Tools Used**

* Spring Initializer (<https://start.spring.io/>)
* Eclipse IDE
* Apache Maven
* Java 17
* Spring Boot 3.5.3

**Step 1: Generate Project Using Spring Initializer**

1. Go to <https://start.spring.io/>
2. Set:
   * **Group**: com.cognizant
   * **Artifact**: spring-learn
3. Select Dependencies:
   * Spring Boot DevTools
   * Spring Web
4. Click **Generate** to download the ZIP file.

**Step 2: Extract Project ZIP**

* Extract the downloaded spring-learn.zip to eclipse-workspace

**Step 3: Build Using Maven**

Open Command Prompt and run:

cd C:\Users\User\eclipse-workspace\spring-learn

mvn clean package -Dhttp.proxyHost=proxy.cognizant.com -Dhttp.proxyPort=6050 -Dhttps.proxyHost=proxy.cognizant.com -Dhttps.proxyPort=6050 -Dhttp.proxyUser=123456

**Step 4: Import the Project into Eclipse**

1. Open Eclipse.
2. Go to:

File > Import > Maven > Existing Maven Projects

1. Select:

C:\Users\User \eclipse-workspace\spring-learn

Click Finish

**Step 5: Run the Application**

1. Open:

src/main/java/com/cognizant/spring\_learn/SpringLearnApplication.java

1. Right-click → **Run As > Java Application**
2. Check console output:

Tomcat started on port 8080

**Step 6: Add a Simple Controller**

Create a file HelloController.java under the same package:

package com.cognizant.spring\_learn;

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

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

@RestController

public class HelloController {

@GetMapping("/hello")

public String sayHello() {

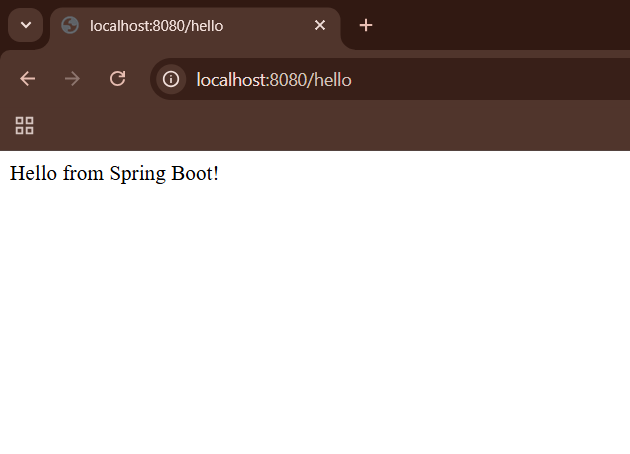
return "Hello from Spring Boot!";

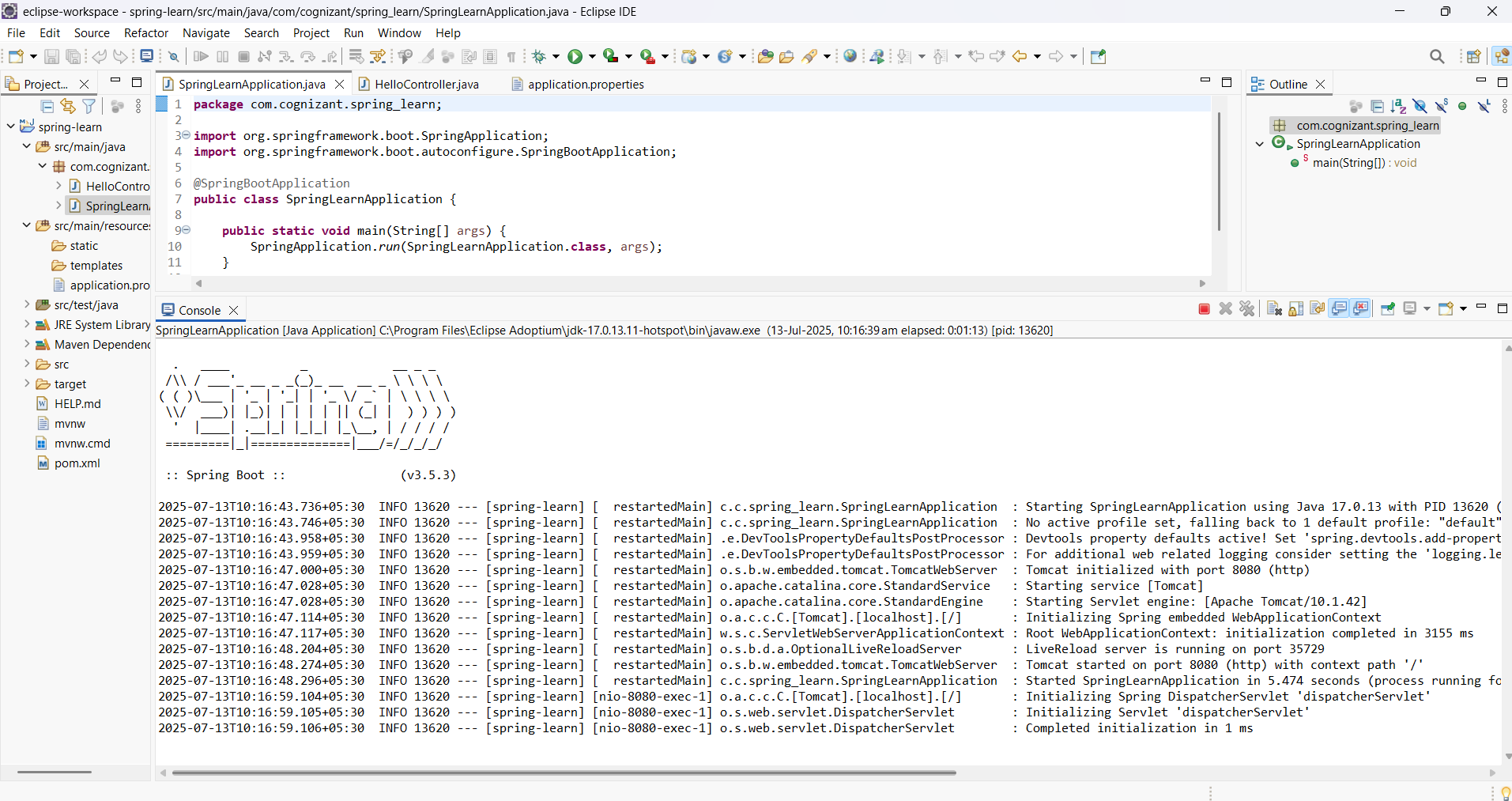
}

}

**Step 7: Test in Browser:** [*http://localhost:8080/hello*](http://localhost:8080/hello)

**Output:**

****



**Hands-on 4:**

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

**Objective:**

An airline’s website supports booking for four countries. Each country has a name and a two-character ISO code. This information is stored in a Spring XML configuration file. The goal is to read the configuration and display the country details.

**Technologies Used:**

* Java
* Spring Core (IOC container)
* Maven
* Eclipse IDE
* SLF4J + Logback for logging

1. 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 + "]";

}

}

1. SpringLearnApplication.java

package com.cognizant.springlearn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class SpringLearnApplication {

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

public static void main(String[] args) {

LOGGER.info("START");

displayCountry();

LOGGER.info("END");

}

public static void displayCountry() {

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

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

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

}

}

1. 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>

1. logback.xml

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

<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>

1. pom.xml Dependencies

<dependencies>

<!-- Spring Core -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.10</version>

</dependency>

<!-- SLF4J Logging -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.30</version>

</dependency>

<dependency>

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

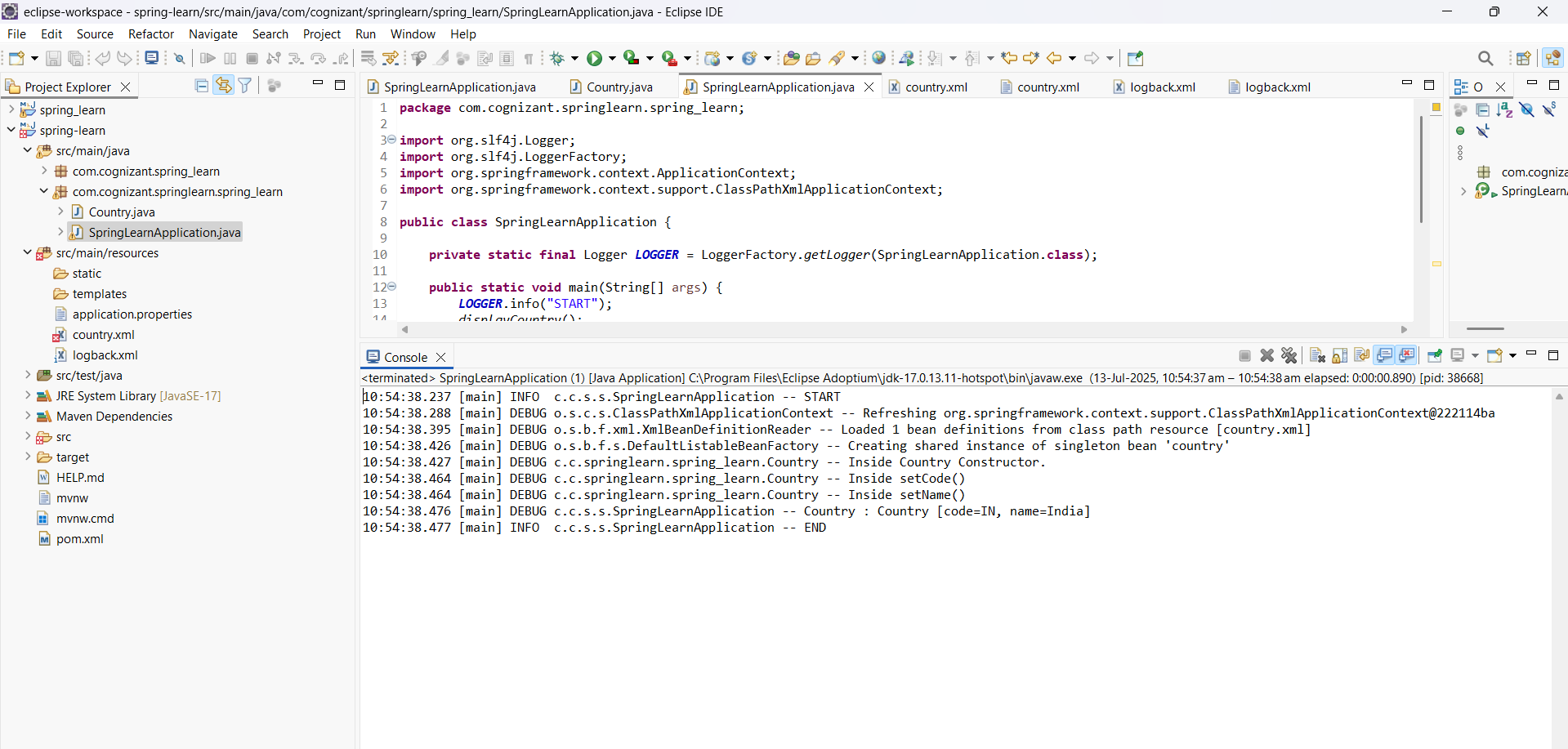
<artifactId>logback-classic</artifactId>

<version>1.2.3</version>

</dependency>

</dependencies>

**Console Output:**

****

**Hello World RESTful Web Service using Spring Boot:**

**Objective:**

Create a RESTful web service that returns "Hello World!!" using the Spring Web Framework.

**Step 1: Create the Controller Class**

**Location:**  
src/main/java/com/cognizant/spring\_learn/controller/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.info("START - sayHello()");

String message = "Hello World!!";

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

return message;

}

}

**Step 2: Configure Port in application.properties**

Location:  
src/main/resources/application.properties

**server.port=8083**

**Step 3: Run the Application**

In Eclipse:  
Right-click on SpringLearnApplication.java → Run As → Java Application

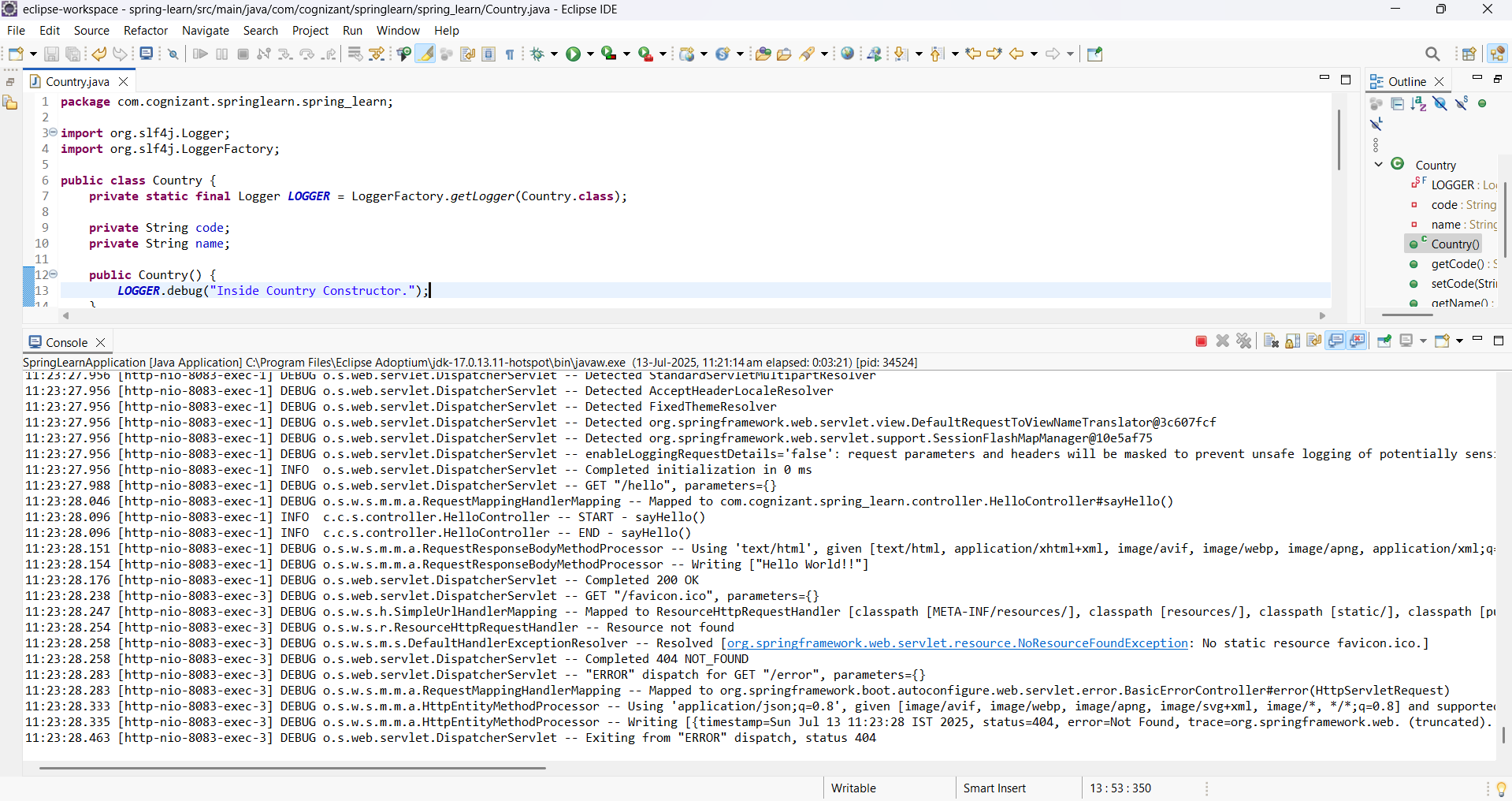
**Step 4: Test the Endpoint**

In Chrome Browser: <http://localhost:8083/hello>

**B. In Postman:**

* Method: GET
* URL: http://localhost:8083/hello
* Click **Send**



****

**Spring Boot REST Service - Country API**

**Objective**

To build a Spring Boot RESTful web service that returns details of the country India using Spring XML configuration.

**1. Country.java**

Location: src/main/java/com/cognizant/spring\_learn/model/Country.java

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 + "]";

}

}

**2. country.xml**

Location: src/main/resources/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.spring\_learn.model.Country">

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

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

</bean>

</beans>

1. **CountryController.java**

src/main/java/com/cognizant/spring\_learn/controller/CountryController.java

package com.cognizant.spring\_learn.controller;

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

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 {

@RequestMapping("/country")

public Country getCountryIndia() {

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

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

return country;

}

}

1. **SpringLearnApplication.java**

Location: src/main/java/com/cognizant/spring\_learn/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) {

SpringApplication.run(SpringLearnApplication.class, args);

}

}

**Testing the API**

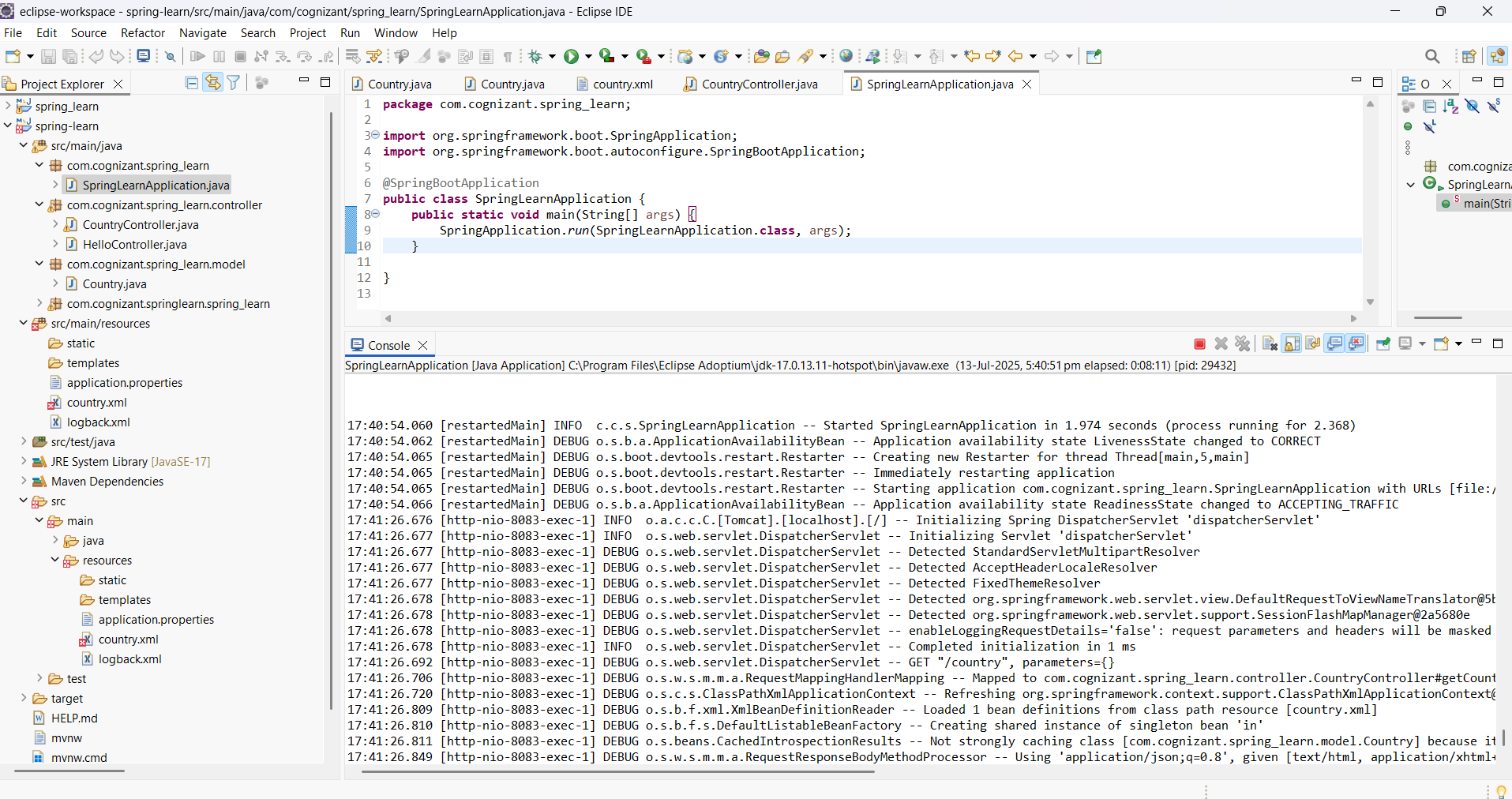
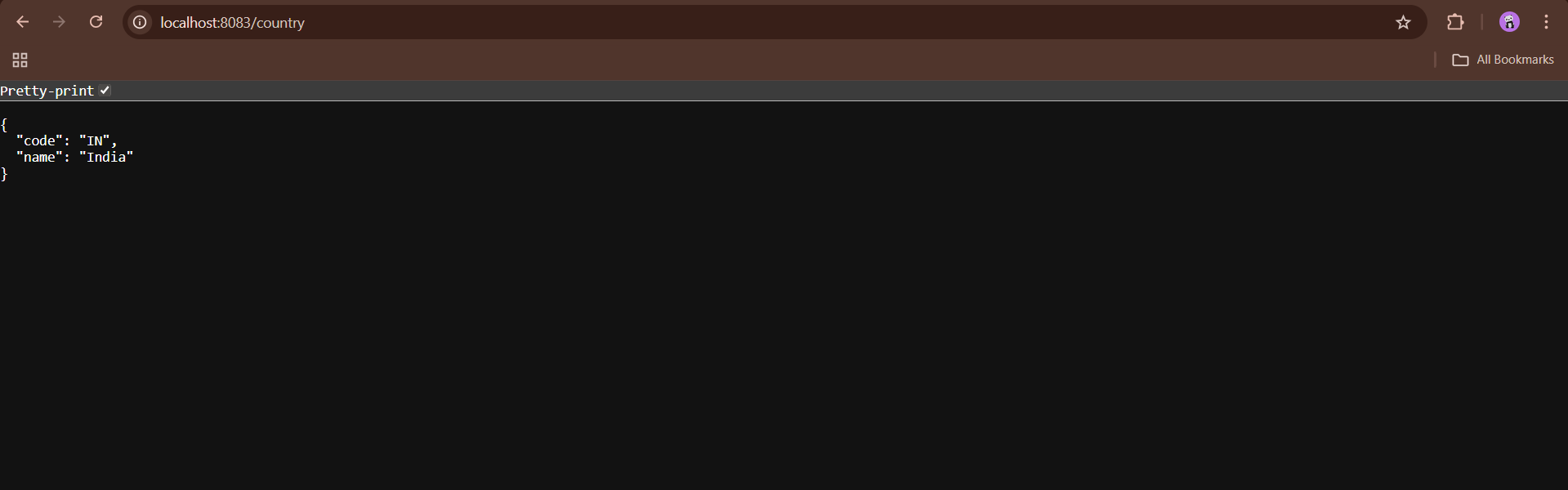
Run the Application:

1. Right-click on SpringLearnApplication.java
2. Click Run As → Java Application
3. Wait for log: Tomcat started on port(s): 8083

🌐 Test in Browser:

Go to: <http://localhost:8083/country>

**Output:**



**Spring Boot REST Service - Get Country by Code**

**Objective**

Build a REST service using Spring Boot that returns a country based on its **code** using Spring XML configuration. The country code match is **case-insensitive**.

**1. Model Class: Country.java**

**Location:** src/main/java/com/cognizant/spring\_learn/model/Country.java

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 + "]";

}

}

**2. Spring Bean XML: 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.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="countryList" class="java.util.ArrayList">

<constructor-arg>

<list>

<ref bean="in"/>

<ref bean="us"/>

</list>

</constructor-arg>

</bean>

</beans>

1. **Service Interface: CountryService.java**

package com.cognizant.spring\_learn.service;

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

public interface CountryService {

Country getCountry(String code);

}

1. **Service Implementation: CountryServiceImpl.java**

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 CountryServiceImpl implements CountryService {

@Override

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);

}

}

1. **Controller: CountryController.java**

package com.cognizant.spring\_learn.controller;

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

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

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

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

@RestController

public class CountryController {

@Autowired

private CountryService countryService;

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

public Country getCountry(@PathVariable String code) {

return countryService.getCountry(code);

}

}

**Testing the API**

**Start the Application**

* Right-click on SpringLearnApplication.java
* Select: Run As → Java Application

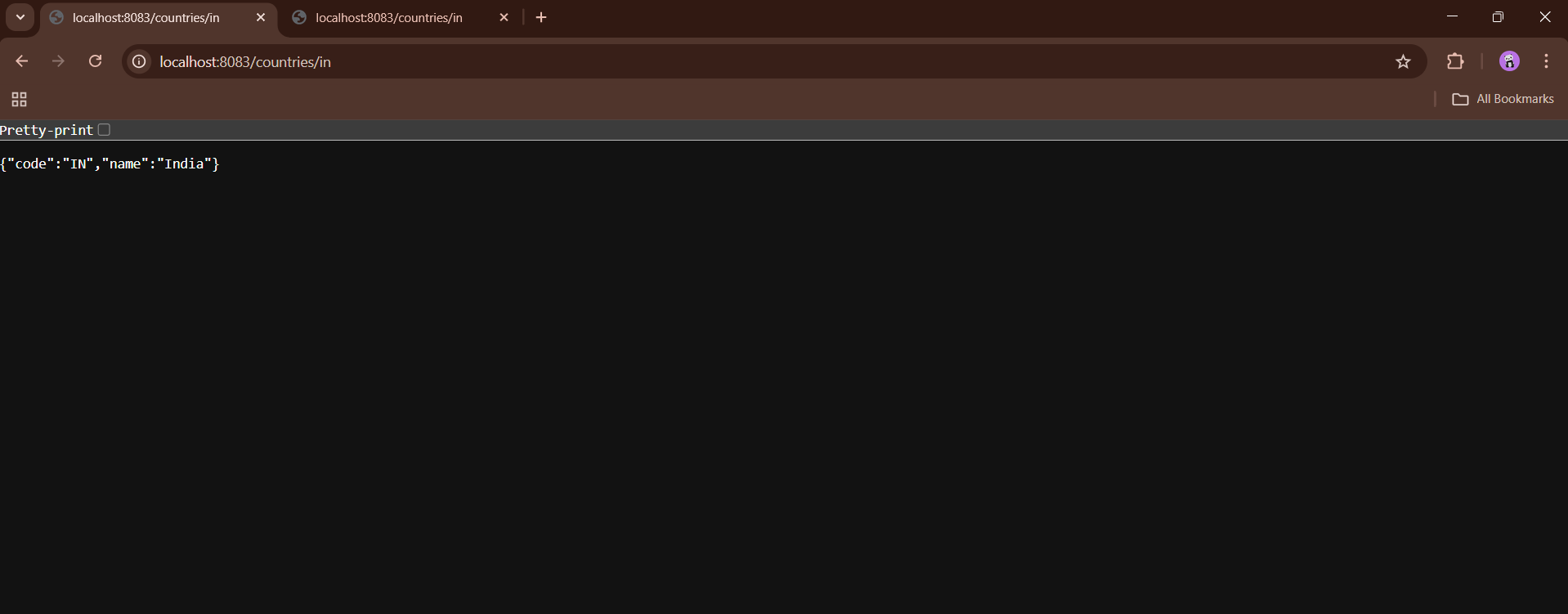
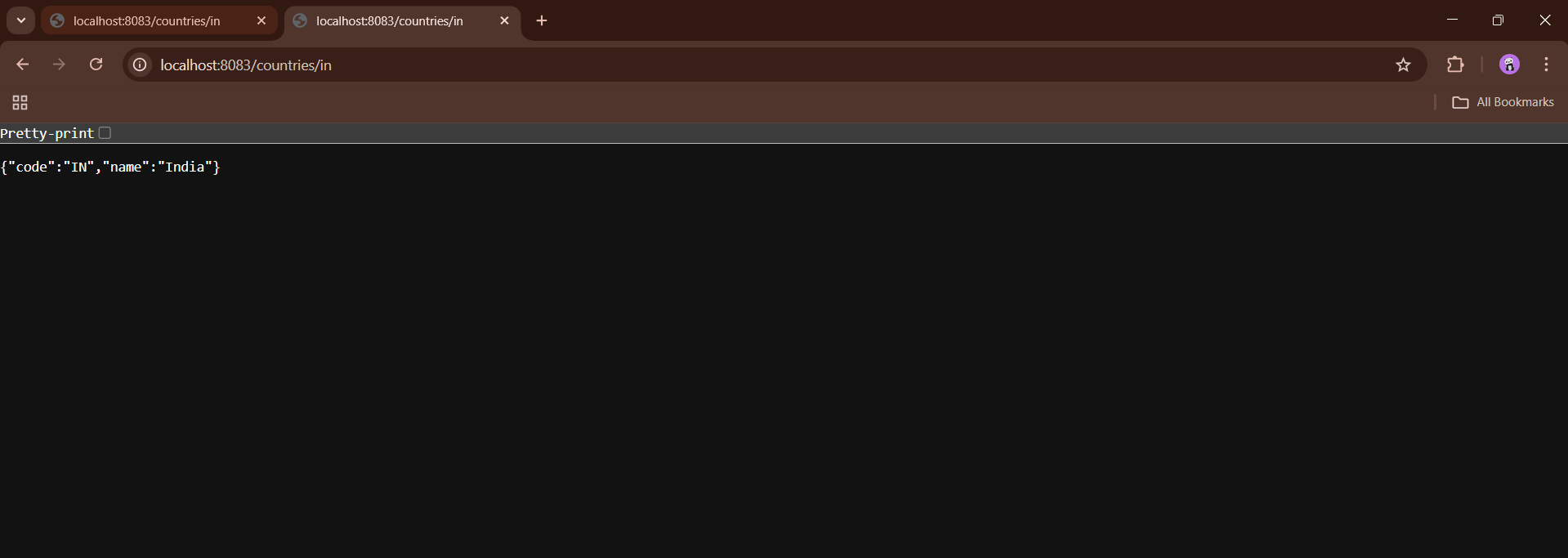
Console should say:   
Tomcat started on port(s): 8083

**Test via Browser/Postman**

**URL:**

GET <http://localhost:8083/countries/in>

**Output:**



**Spring Boot – JWT Authentication Service**

**Objective**

To build a RESTful authentication service using Spring Boot that returns a **JWT token** when valid user credentials are passed using **Basic Authentication**.

**1.application.properties**

server.port=8080

spring.security.user.name=user

spring.security.user.password=pwd

**2.pom.xml Dependencies**

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

</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>

</dependencies>

**3.JwtAuthApplication.java**

package com.cognizant.jwt;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class JwtAuthApplication {

public static void main(String[] args) {

SpringApplication.run(JwtAuthApplication.class, args);

}

}

**4.JwtUtil.java**

package com.cognizant.jwt.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import org.springframework.stereotype.Component;

import java.util.Date;

@Component

public class JwtUtil {

private String secretKey = "secret";

public String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date())

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

.signWith(SignatureAlgorithm.HS256, secretKey)

.compact();

}

}

**5.AuthenticationController.java**

package com.cognizant.jwt.controller;

import com.cognizant.jwt.util.JwtUtil;

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

import org.springframework.http.ResponseEntity;

import org.springframework.security.core.Authentication;

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

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

@RestController

public class AuthenticationController {

@Autowired

private JwtUtil jwtUtil;

@GetMapping("/authenticate")

public ResponseEntity<?> authenticate(Authentication authentication) {

if (authentication == null) {

return ResponseEntity.status(401).body("Authentication failed: No user found");

}

String token = jwtUtil.generateToken(authentication.getName());

return ResponseEntity.ok().body("{\"token\":\"" + token + "\"}");

}

}

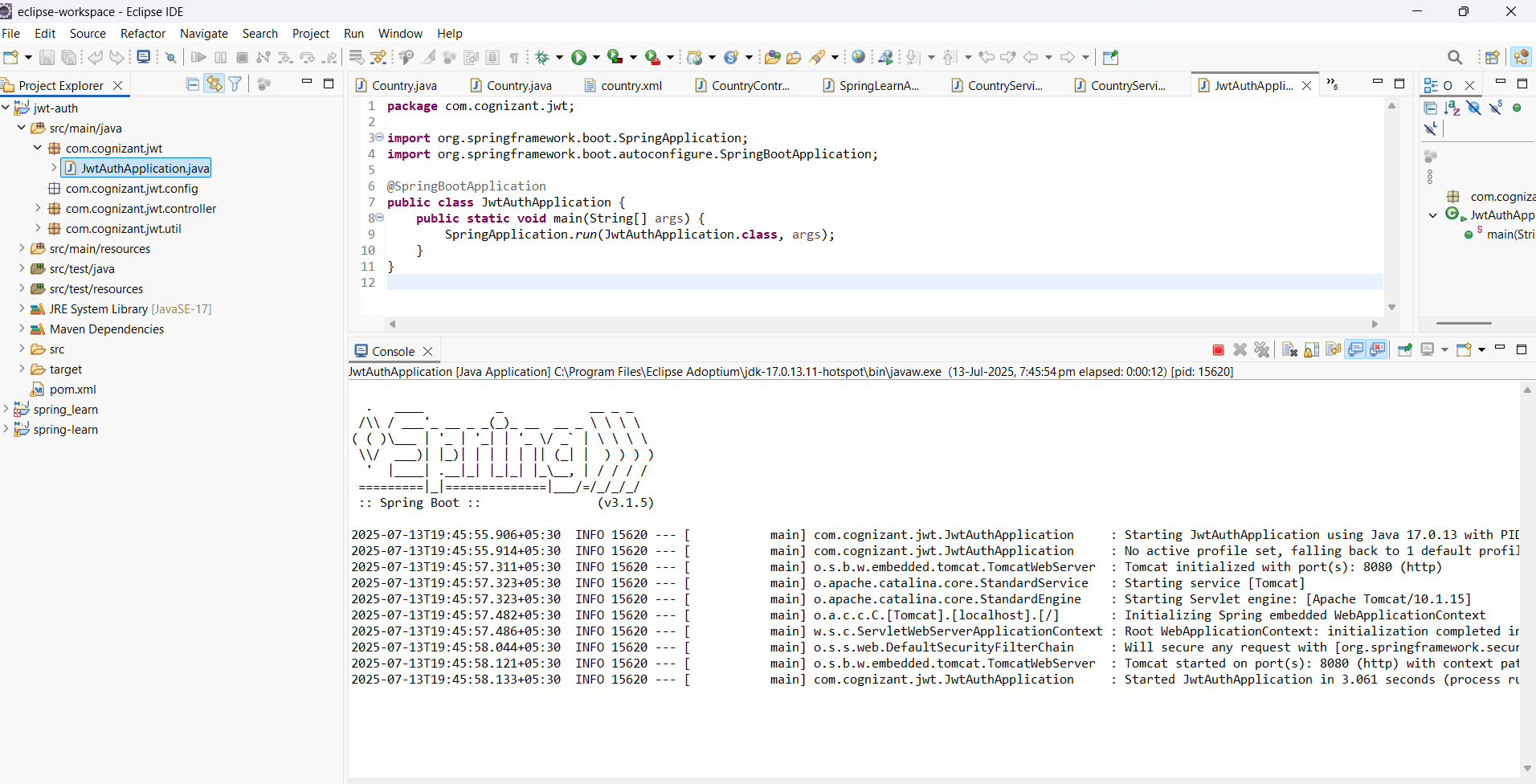
**6.Test Using PowerShell**

$pair = "user:pwd"

$encoded = [Convert]::ToBase64String([Text.Encoding]::ASCII.GetBytes($pair))

Invoke-RestMethod -Uri http://localhost:8080/authenticate -Headers @{Authorization = "Basic $encoded"}

**Output :**

****