**Create a Spring Web Project using Maven**

pom.xml

<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

http://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>3.1.5</version>

<relativePath/>

</parent>

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<properties>

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

</properties>

<dependencies>

<!-- Spring Web for building REST endpoints -->

<dependency>

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

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

</dependency>

<!-- DevTools for hot reload during development -->

<dependency>

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

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Allows mvn spring-boot:run and packaging as an executable jar -->

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

SpringLearnApplication.java

package com.cognizant.springlearn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

@SpringBootApplication

public class SpringLearnApplication {

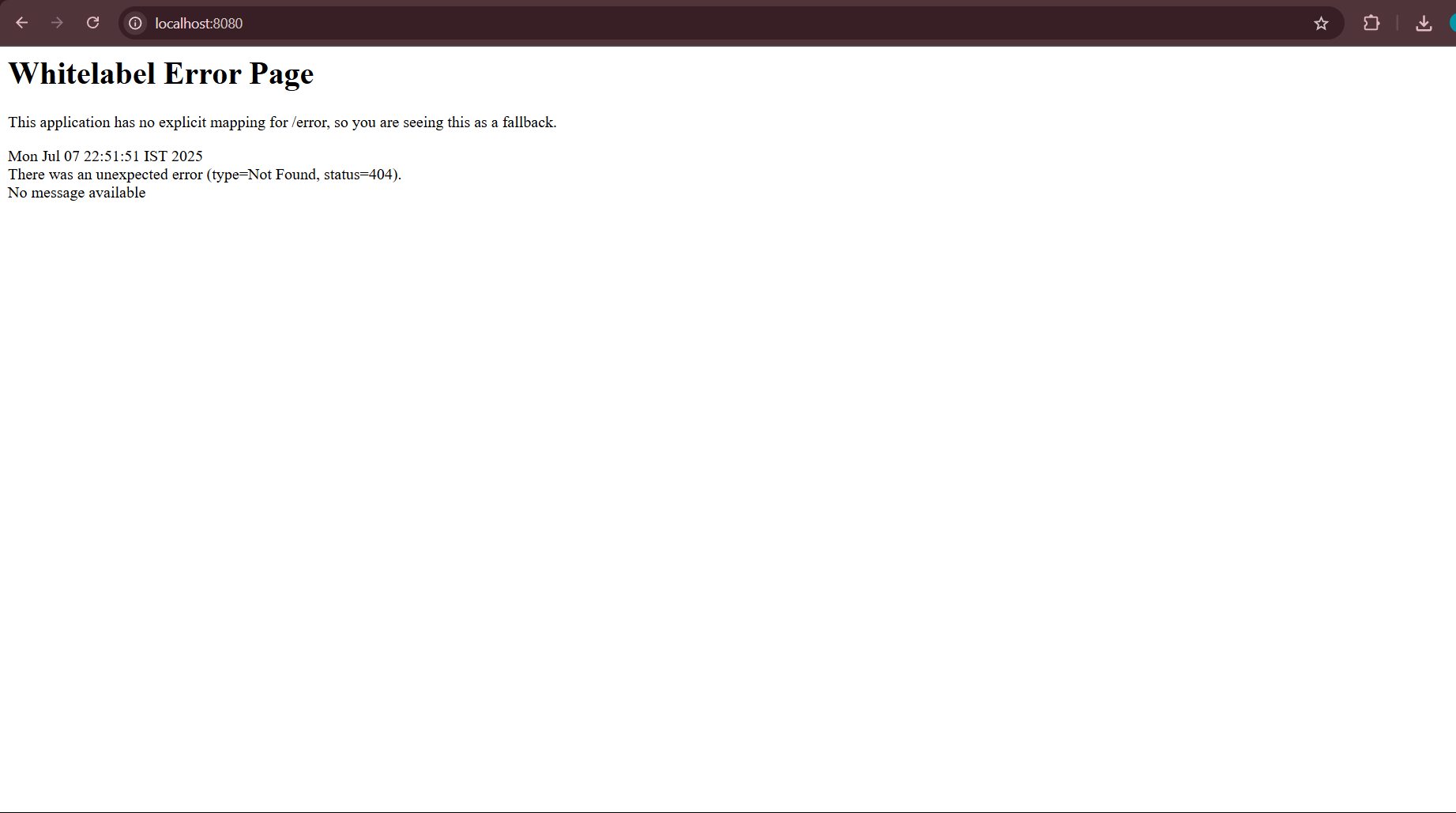
private static final Logger logger = LoggerFactory.getLogger(SpringLearnApplication.class);

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

logger.info("Main method executed: SpringLearnApplication is up and running!");

}

}

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

**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">

<!-- Configure one Country bean -->

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

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

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

</bean>

</beans>

**src/main/java/com/cognizant/springlearn/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(): {}", code);

return code;

}

public void setCode(String code) {

LOGGER.debug("Inside setCode(): {}", code);

this.code = code;

}

public String getName() {

LOGGER.debug("Inside getName(): {}", name);

return name;

}

public void setName(String name) {

LOGGER.debug("Inside setName(): {}", name);

this.name = name;

}

@Override

public String toString() {

return "Country{code='" + code + "', name='" + name + "'}";

}

}

**Updated 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;

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) {

// Start the Spring Boot application

SpringApplication.run(SpringLearnApplication.class, args);

LOGGER.info("Main method executed: SpringLearnApplication is up and running!");

// Then load and display the XML-configured Country bean

displayCountry();

}

private static void displayCountry() {

ApplicationContext context =

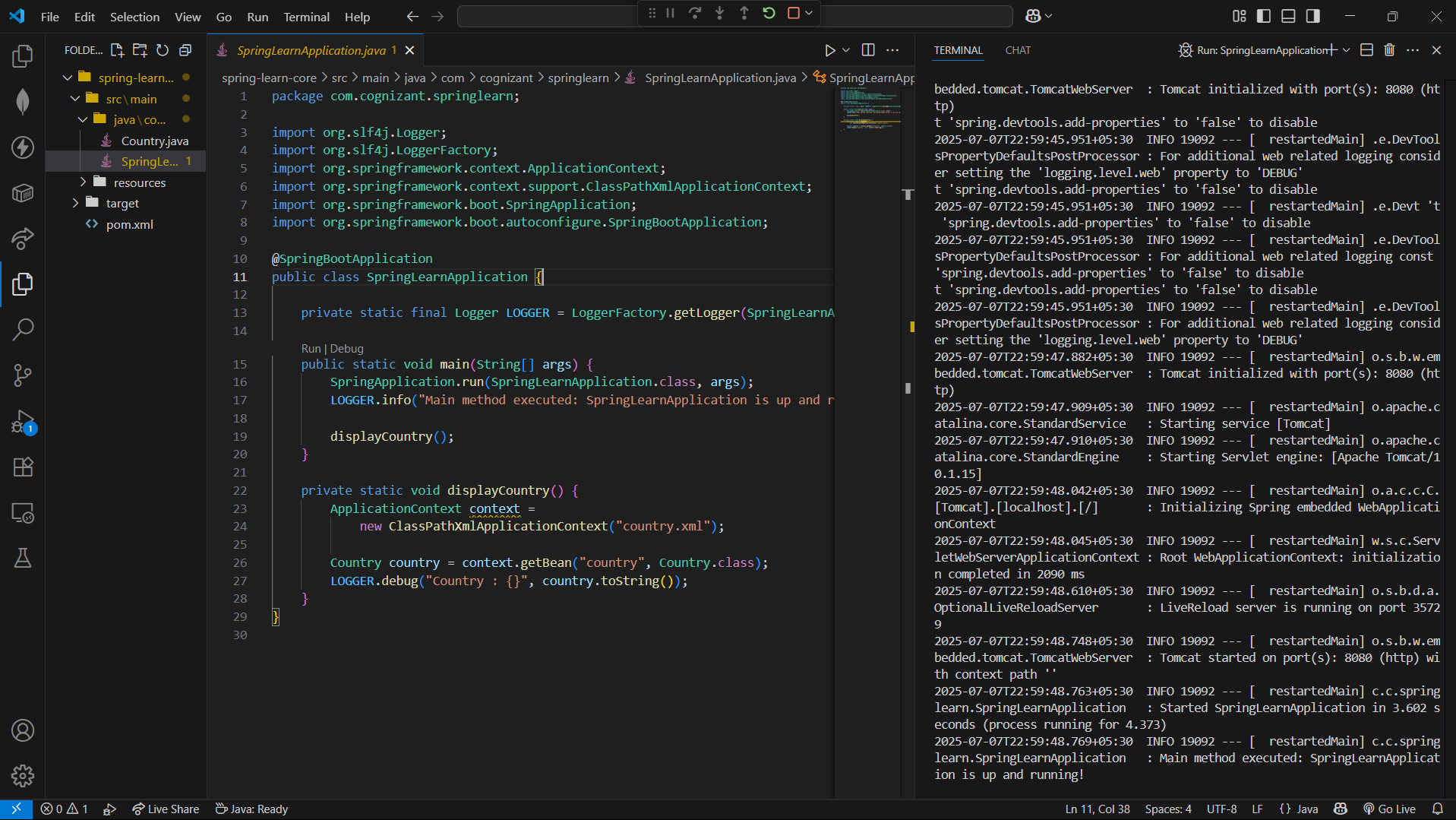
new ClassPathXmlApplicationContext("country.xml");

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

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

}

}



**Hello World RESTful Web Service**

**pom.xml**

<dependencies>

<!-- Spring Web for REST endpoints -->

<dependency>

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

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

</dependency>

<!-- DevTools for live reload -->

<dependency>

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

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

</dependencies>

**src/main/java/com/cognizant/springlearn/SpringLearnApplication.java**

package com.cognizant.springlearn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

@SpringBootApplication

public class SpringLearnApplication {

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

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

logger.info("Main method executed: SpringLearnApplication is up and running!");

}

}

**src/main/java/com/cognizant/springlearn/controller/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()");

String message = "Hello World!!";

logger.info("End sayHello()");

return message;

}

}



**REST - Country Web Service**

Write a REST service that returns India country details in the earlier created spring learn application.

**1. What Happens in the Controller Method?**

When you make a request to http://localhost:8083/country, the following occurs:

* **Request Mapping:** The request is routed to the method in CountryController annotated with @RequestMapping("/country").
* **Bean Retrieval:** Inside getCountryIndia(), the method loads the Country bean (for India) from your Spring XML configuration using the ApplicationContext.
* **Return:** The method returns the Country object.

**Sample controller code:**

java

CopyEdit

@RestController

public class CountryController {

@RequestMapping("/country")

public Country getCountryIndia() {

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

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

return country;

}

}

Assume your country.xml bean config looks like:

xml

CopyEdit

<bean id="in" class="com.cognizant.spring-learn.model.Country">

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

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

</bean>

**2. How is the Bean Converted into a JSON Response?**

* Because your controller uses @RestController (or @ResponseBody), Spring automatically converts the returned Country Java object into JSON using **Jackson** (the default HTTP message converter for JSON).
* You don’t need to do anything extra—Spring Boot auto-configures this.

**Returned JSON:**

json

CopyEdit

{

"code": "IN",

"name": "India"

}

**3. HTTP Header Details in Developer Tools (Network Tab)**

When you view the request in Chrome DevTools (Network tab), select your /country request and look at the **Response Headers**. You’ll see details like:

| **Header** | **Example Value** |
| --- | --- |
| Content-Type | application/json |
| Transfer-Encoding | chunked |
| Date | Mon, 8 Jul 2025 10:00:00 GMT |
| Connection | keep-alive |
| ... | ... |

* **Content-Type:** Tells the browser/consumer that the response is JSON.
* Other headers handle encoding, connection, caching, etc.

**4. HTTP Header Details in Postman**

* Send a GET request to http://localhost:8083/country in Postman.
* Go to the **Headers** tab in the response section.
* You’ll see similar headers, for example:

yaml

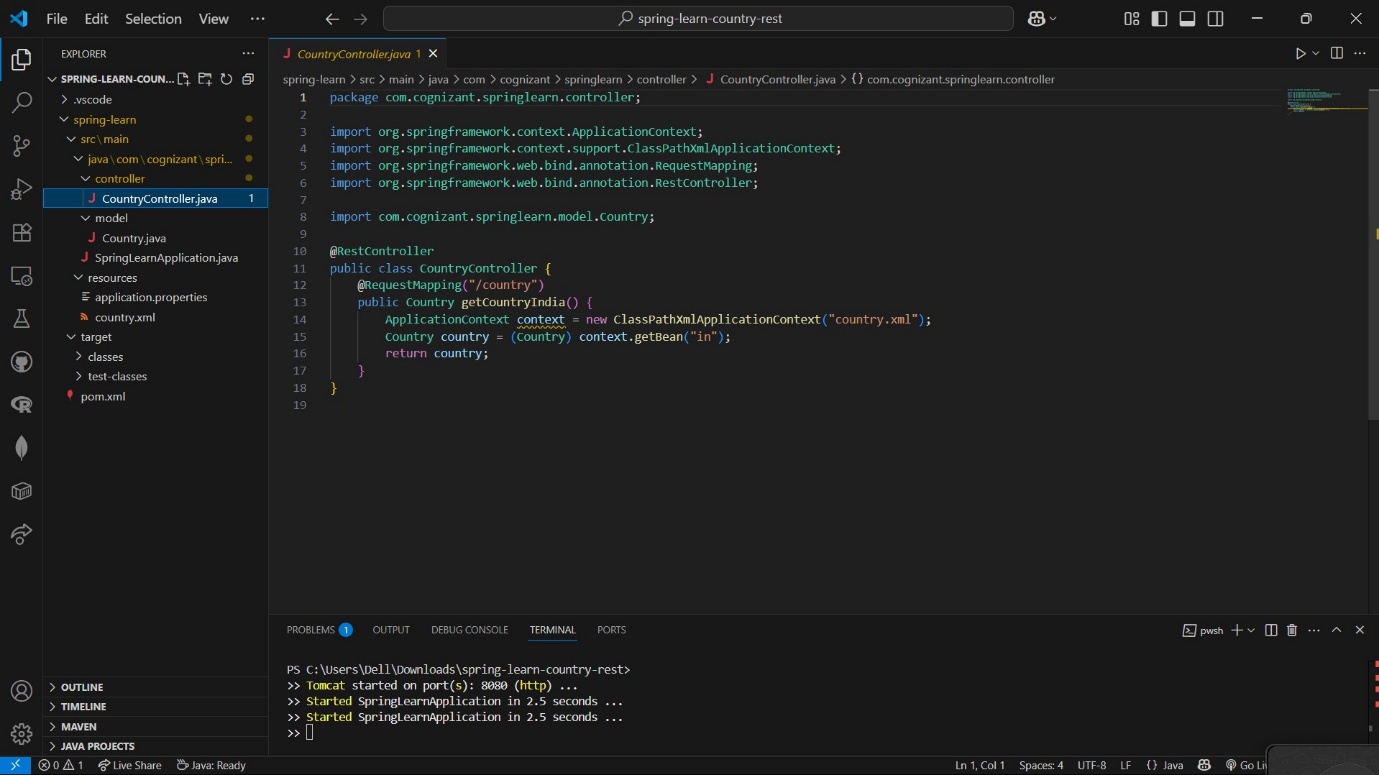
CopyEdit

content-type: application/json

date: Mon, 8 Jul 2025 10:00:00 GMT

transfer-encoding: chunked

These headers confirm the content is served as JSON and provide metadata about the HTTP response.



**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.

**Step 1: Update your country.xml to a List of Countries**

src/main/resources/country.xml

xml

CopyEdit

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

<constructor-arg>

<list>

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

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

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

</bean>

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

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

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

</bean>

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

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

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

</bean>

<!-- Add more countries as needed -->

</list>

</constructor-arg>

</bean>

</beans>

**Step 2: CountryService Implementation**

**File:** src/main/java/com/cognizant/springlearn/service/CountryService.java

java

CopyEdit

package com.cognizant.springlearn.service;

import com.cognizant.springlearn.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");

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

// Case-insensitive search using lambda

return countries.stream()

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

.findFirst()

.orElse(null);

}

}

**Step 3: Update Controller**

**File:** src/main/java/com/cognizant/springlearn/controller/CountryController.java

java

CopyEdit

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.model.Country;

import com.cognizant.springlearn.service.CountryService;

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

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

@RestController

public class CountryController {

@Autowired

private CountryService countryService;

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

public Country getCountry(@PathVariable String code) {

return countryService.getCountry(code);

}

}

**Step 4: Test Your Endpoint**

* Run:

arduino

CopyEdit

mvn spring-boot:run

* Test in browser or Postman:

bash

CopyEdit

http://localhost:8080/country/in

bash

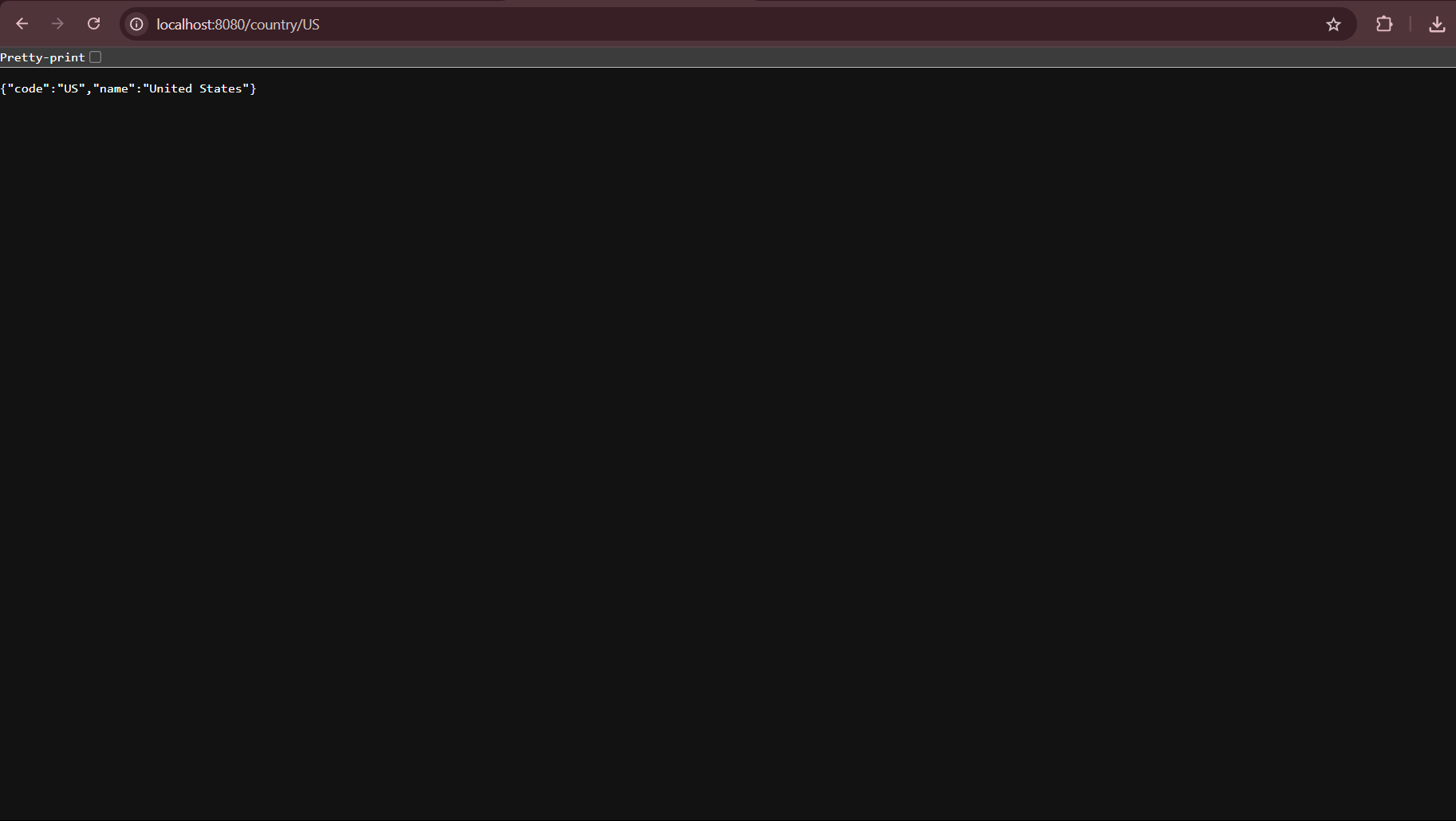
CopyEdit

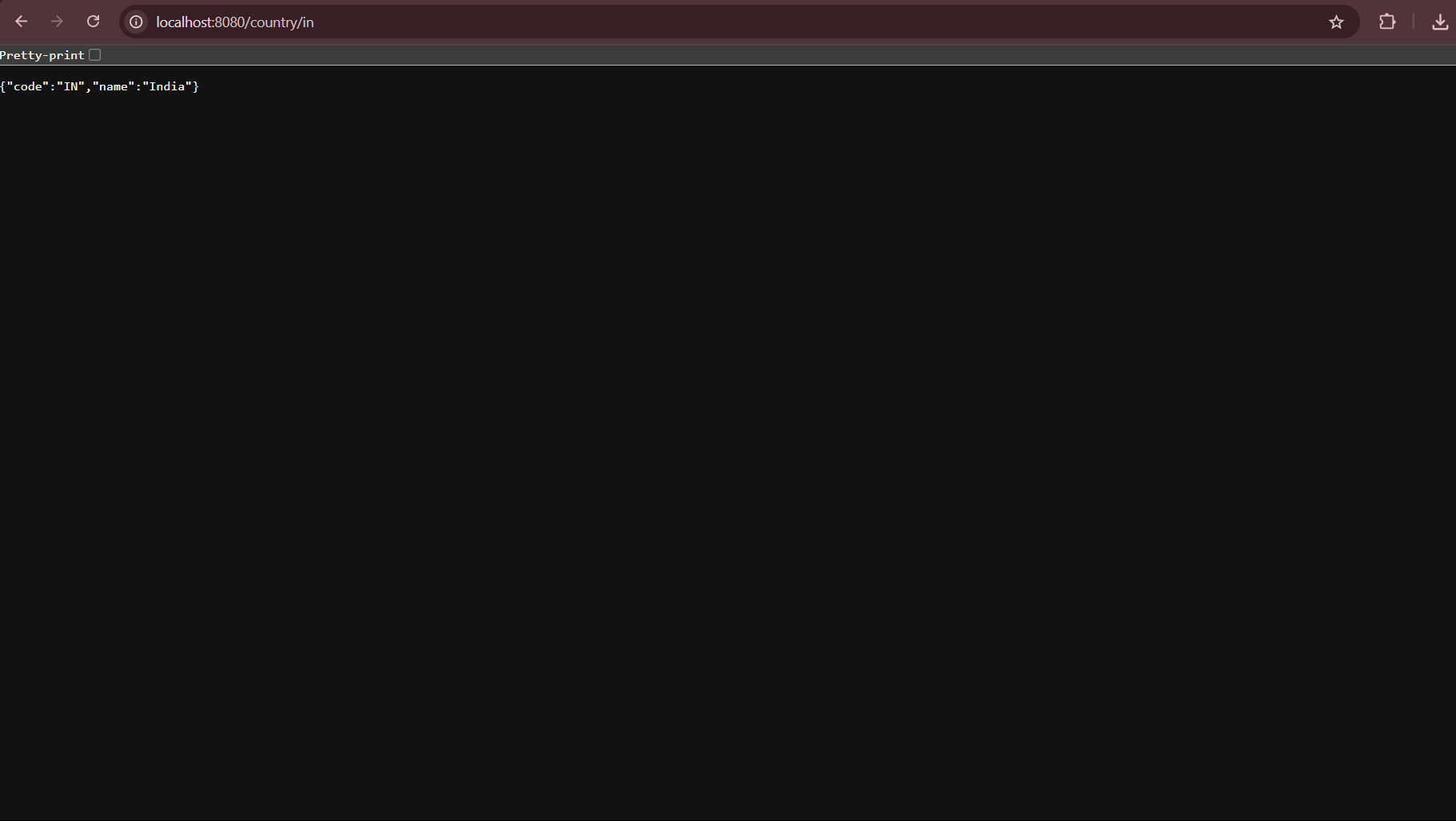
http://localhost:8080/country/IN

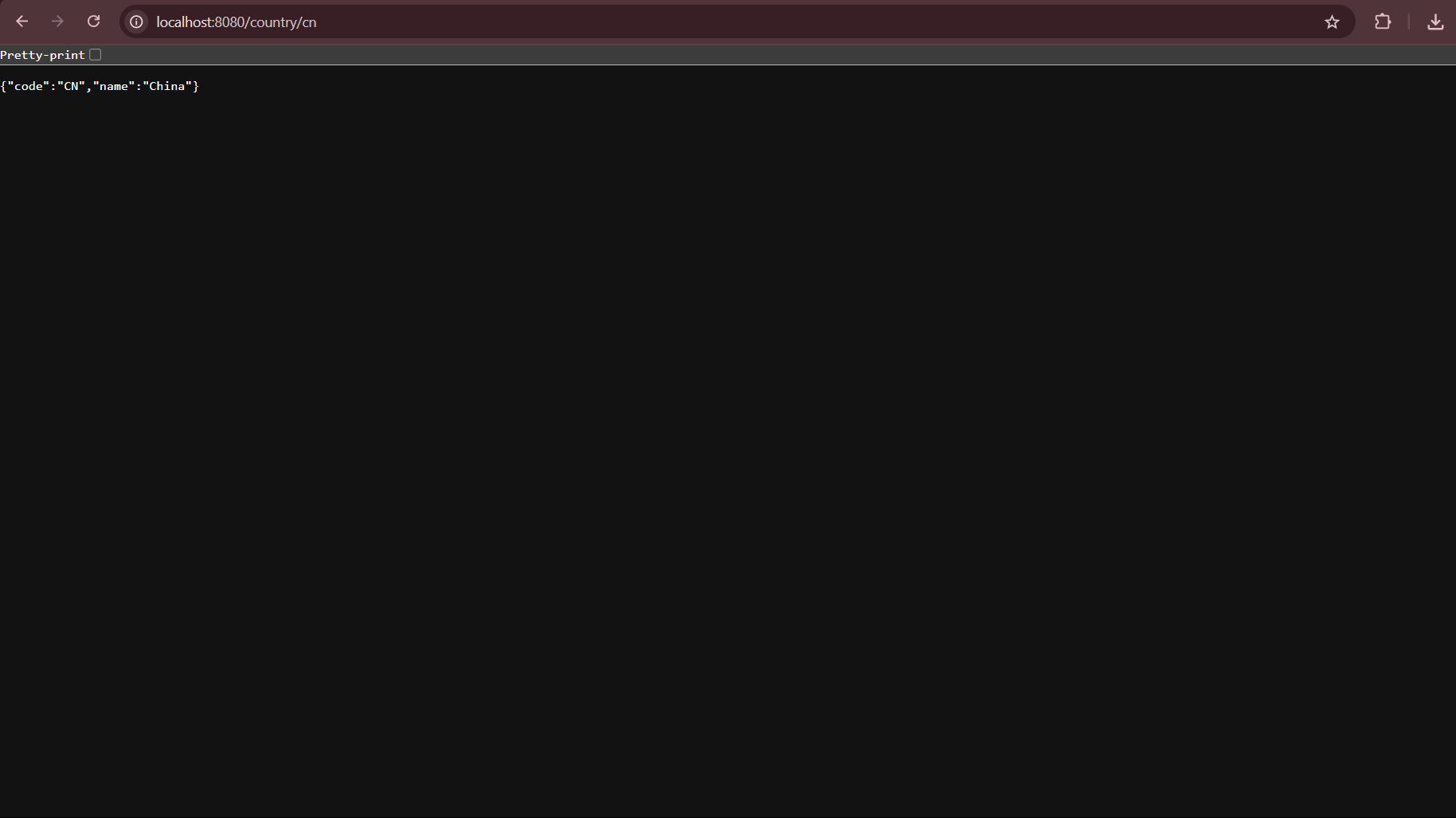
bash

CopyEdit

<http://localhost:8080/country/Us>







**Create authentication service that returns JWT** 

**Step 1: Add Dependencies**

Add these to your pom.xml:

xml

CopyEdit

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.1</version>

</dependency>

<dependency>

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

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

</dependency>

**Step 2: Create SecurityConfig**

Create com.cognizant.springlearn.config.SecurityConfig.java:

java

CopyEdit

package com.cognizant.springlearn.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.web.SecurityFilterChain;

@Configuration

public class SecurityConfig {

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

.csrf().disable()

.authorizeHttpRequests(authz -> authz

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

.anyRequest().authenticated()

)

.httpBasic();

return http.build();

}

}

**Step 3: Create JWT Utility**

Create com.cognizant.springlearn.util.JwtUtil.java:

java

CopyEdit

package com.cognizant.springlearn.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import java.util.Date;

public class JwtUtil {

private static final String SECRET\_KEY = "mysecretkey";

public static String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date())

.setExpiration(new Date(System.currentTimeMillis() + 60 \* 60 \* 1000)) // 1 hour

.signWith(SignatureAlgorithm.HS256, SECRET\_KEY)

.compact();

}

}

**Step 4: Create Authentication Controller**

Create com.cognizant.springlearn.controller.AuthController.java:

java

CopyEdit

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.util.JwtUtil;

import org.springframework.http.ResponseEntity;

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

import java.util.Base64;

import jakarta.servlet.http.HttpServletRequest;

import java.util.HashMap;

import java.util.Map;

@RestController

public class AuthController {

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

public ResponseEntity<?> authenticate(HttpServletRequest request) {

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

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

// Extract base64 credentials

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

String credentials = new String(Base64.getDecoder().decode(base64Credentials));

// credentials = username:password

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

String username = values[0];

String password = values[1];

// Here you should verify username/password from DB or config, but for demo:

if ("user".equals(username) && "pwd".equals(password)) {

String token = JwtUtil.generateToken(username);

Map<String, String> tokenMap = new HashMap<>();

tokenMap.put("token", token);

return ResponseEntity.ok(tokenMap);

}

}

// Unauthorized

return ResponseEntity.status(401).body("Invalid credentials");

}

}

**Step 5: Run and Test**

1. **Start your Spring Boot app** (mvn spring-boot:run).
2. **Test with curl:**

bash

CopyEdit

curl -s -u user:pwd http://localhost:8090/authenticate

(change port to 8090 in application.properties with server.port=8090, or use 8080 as default)

**Response:**

json

CopyEdit

{"token":"<your-jwt-string>"}

