**2. Spring-Rest-HandsOn**

**1. Hello World RESTful Web Service**

### Overview

This REST service returns the hardcoded string "Hello World!!" using the Spring Web Framework.

### Endpoint Details

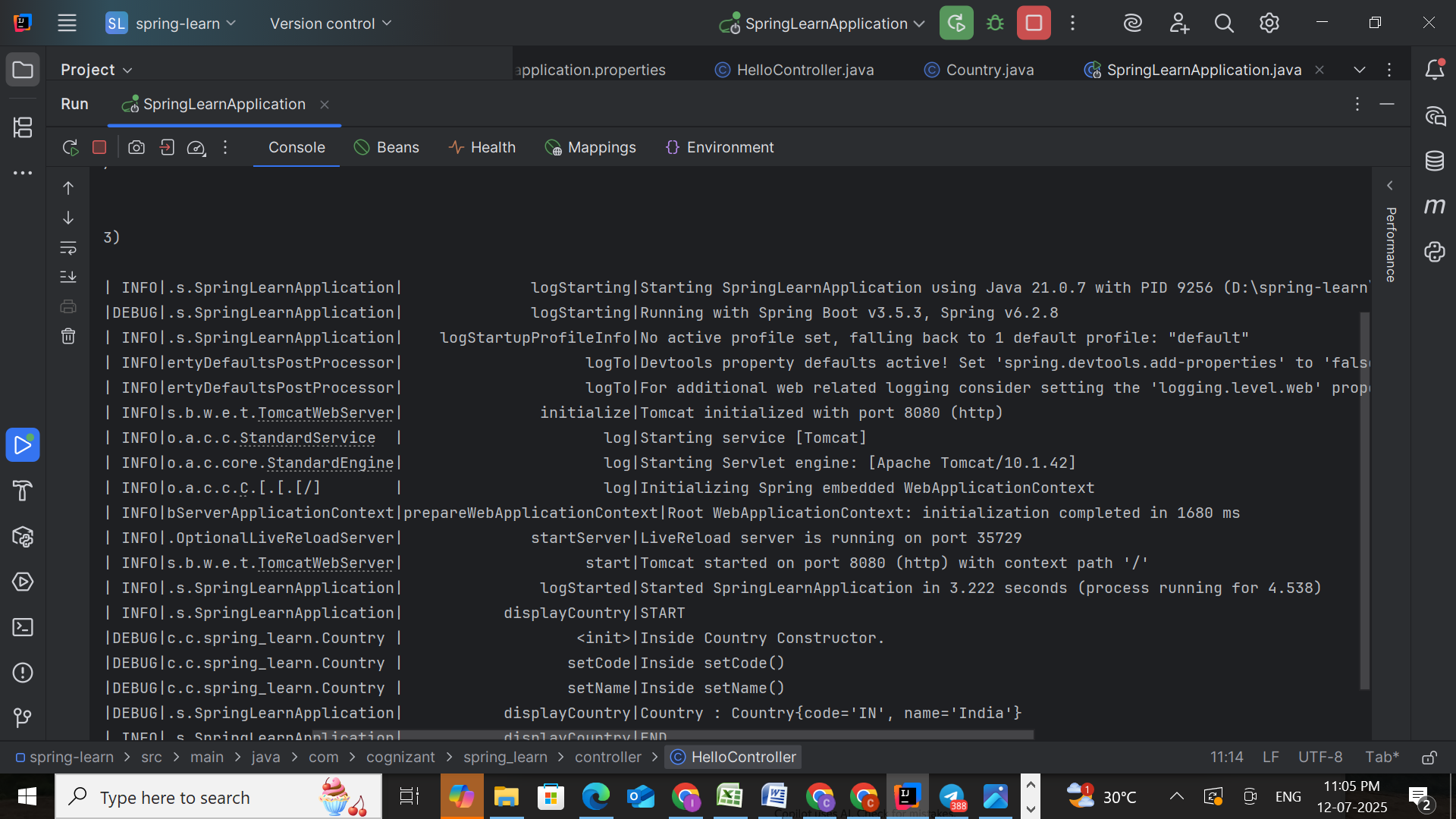
* **Method:** GET
* **URL:** /hello
* **Controller:** com.cognizant.spring\_learn.controller.HelloController
* **Method Signature:** public String sayHello()

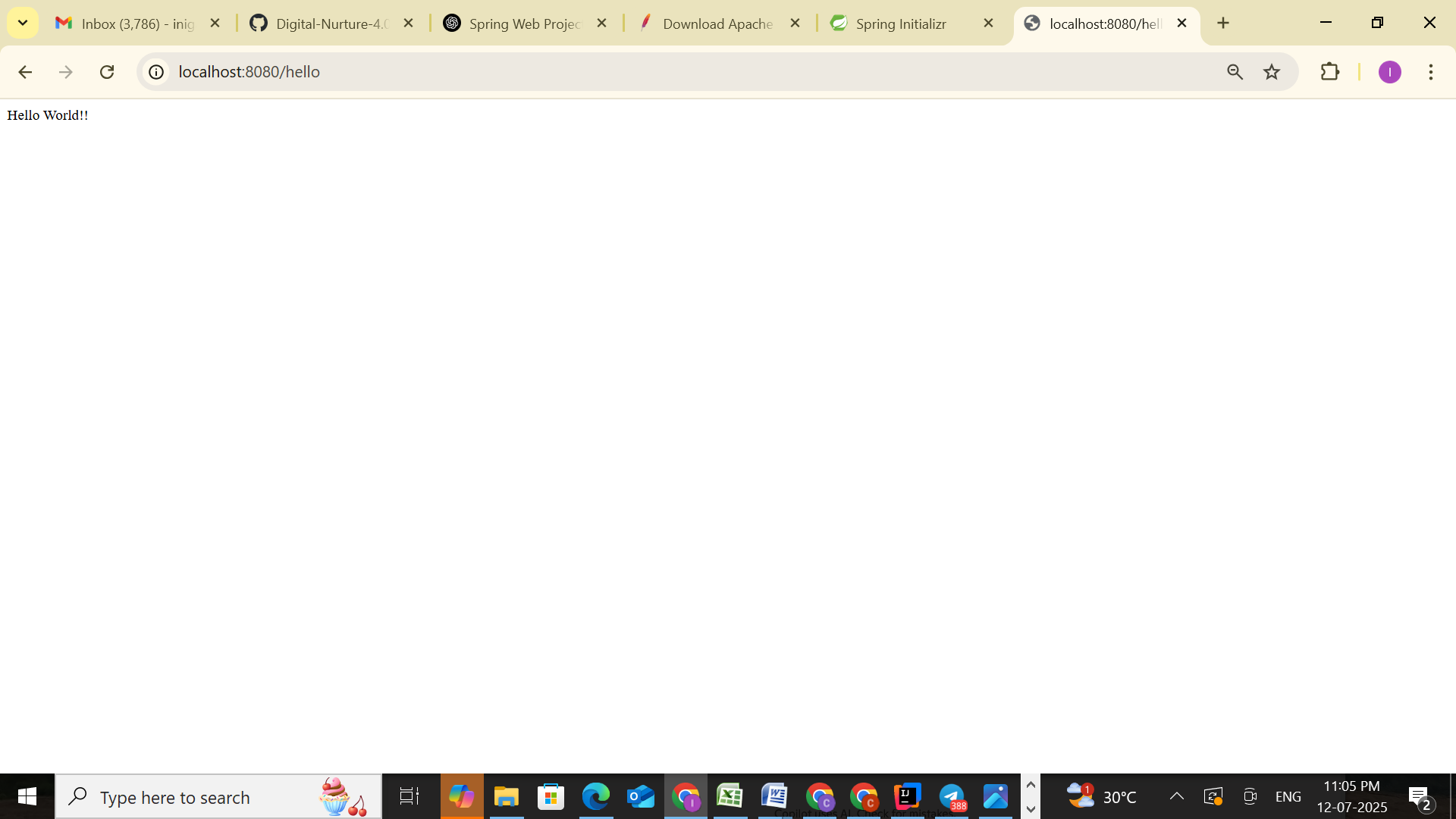
**Step-by-step:**

1. **Create a Controller Class**
   * File: HelloController.java
   * Location: src/main/java/com/cognizant/spring\_learn/controller/

package com.cognizant.spring\_learn;  
  
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("START");  
 SpringApplication.*run*(SpringLearnApplication.class, args);  
 *displayCountry*();  
 *LOGGER*.info("END");  
 }  
  
 public static void displayCountry() {  
 *LOGGER*.info("START");  
 ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");  
 Country country = context.getBean("country", Country.class);  
 *LOGGER*.debug("Country : {}", country.toString());  
 *LOGGER*.info("END");  
 }  
}

1. **Run the application**
2. **Open browser/postman:**
   * **Test URL:** http://localhost:8083/hello
   * **Output:** Hello World!!



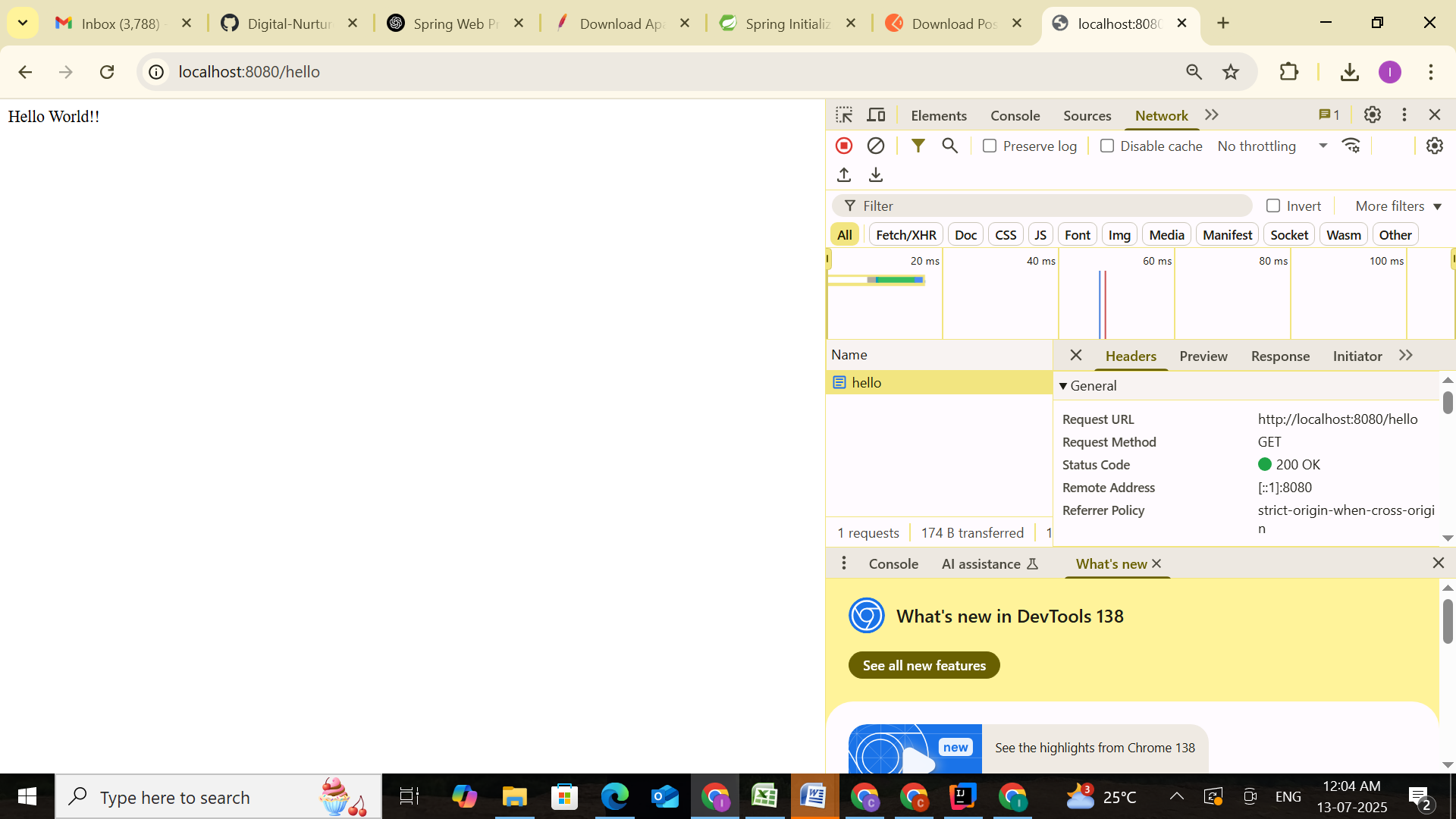


### SME Explanation for HTTP Header Inspection

### 1. ****In Browser – Developer Tools → Network Tab****

When you test a REST API like http://localhost:8080/hello in Chrome:

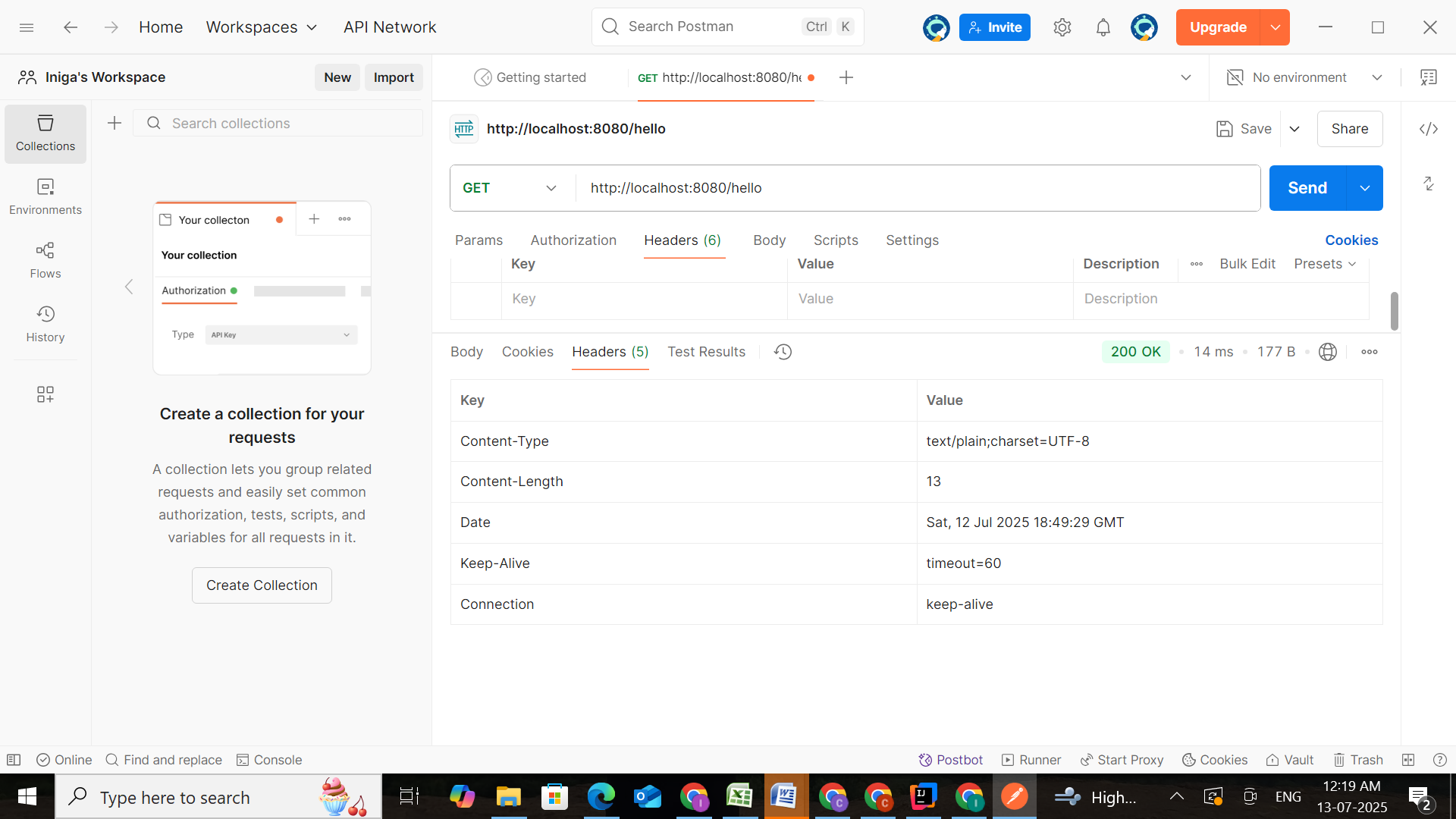
1. Open **Developer Tools**.
2. Go to the **Network** tab.
3. Refresh the page after opening this tab.
4. Click on the /hello request.
5. Select the **Headers** section.



### 2. ****In Postman – Headers Tab****

After hitting the request (e.g., GET http://localhost:8080/hello) in Postman:

1. Switch to the **Headers** tab in the response section.
2. It shows the response headers sent by the Spring Boot application.



**2. REST - Country Web Service**

### Overview

This REST endpoint returns a country bean (India) by loading it from the Spring XML configuration file.

### Endpoint Details

* **Method:** GET
* **URL:** /country
* **Controller:** com.cognizant.spring\_learn.controller.CountryController
* **Method Name:** getCountryIndia()

**Step-by-step:**

1. **Create XML File**
   * **File:** country.xml
   * **Location:** src/main/resources/

<?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.spring\_learn.Country">  
 <property name="code" value="IN" />  
 <property name="name" value="India" />  
 </bean>  
</beans>

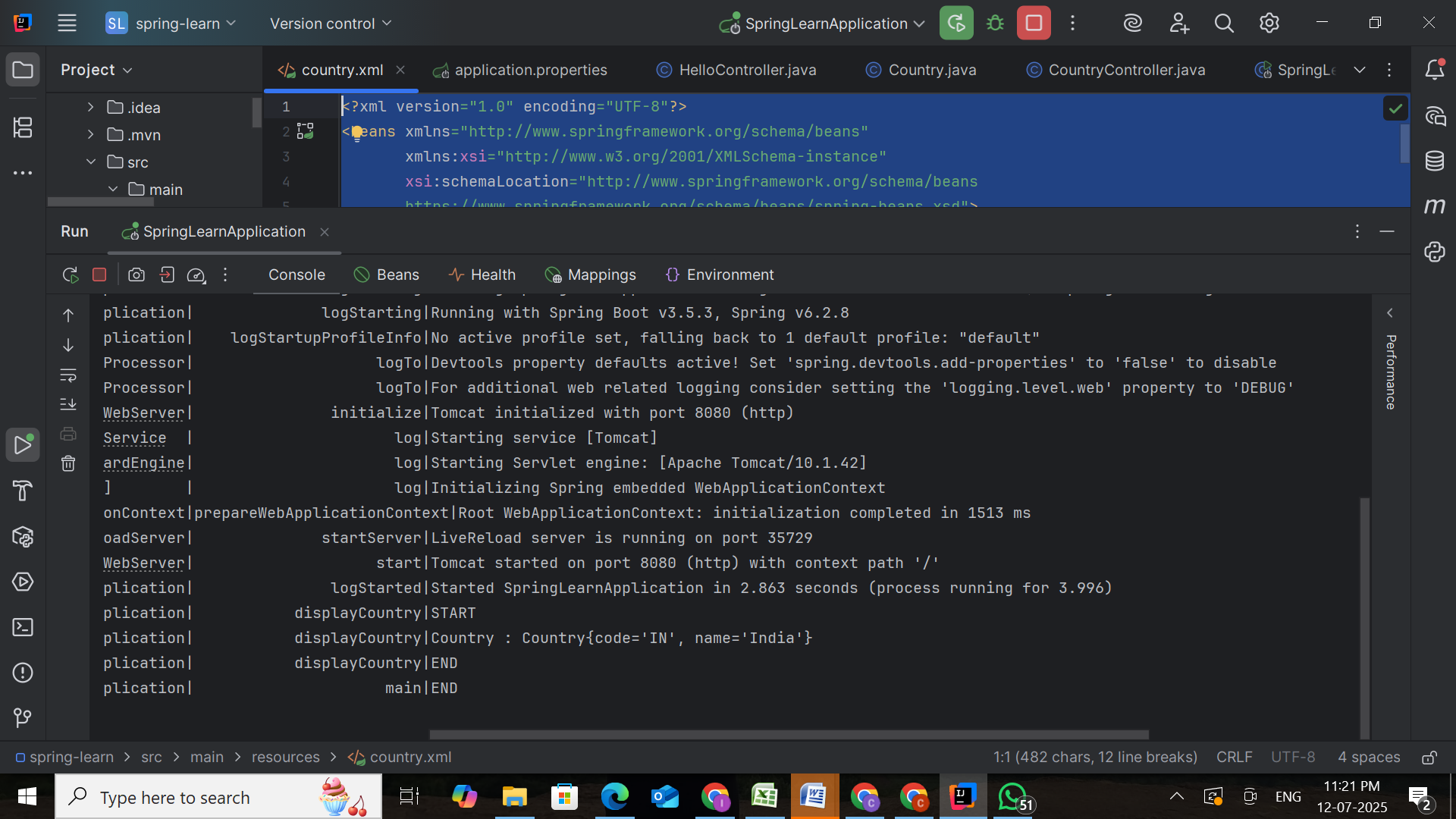
1. **Create Country class**
   * **File:** Country.java
   * **Location:** src/main/java/com/cognizant/spring\_learn/

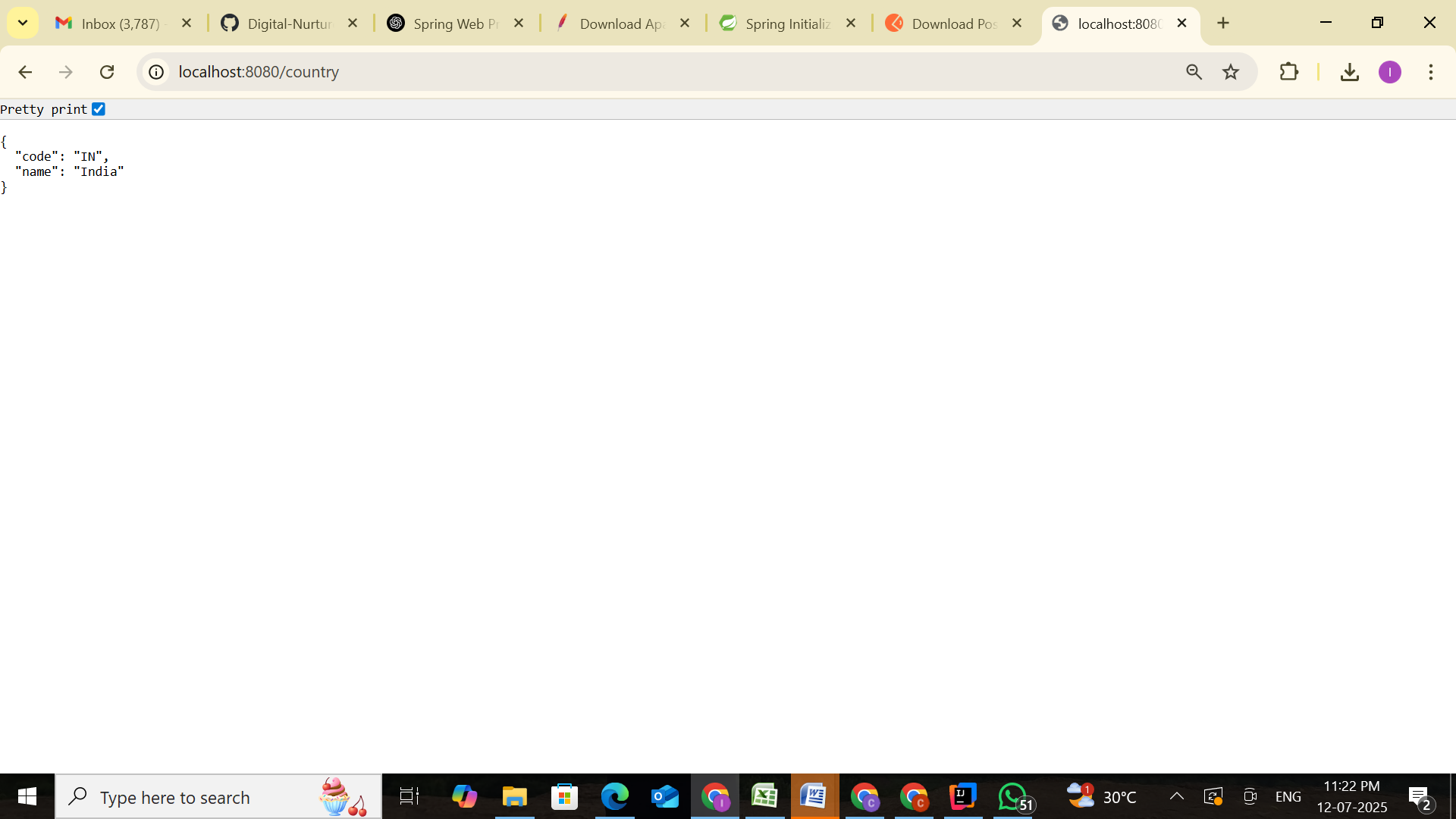
package com.cognizant.spring\_learn;  
  
public class Country {  
 private String code;  
 private String name;  
  
 public Country() {}  
  
 public String getCode() {  
 return code;  
 }  
  
 public void setCode(String code) {  
 this.code = code;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 @Override  
 public String toString() {  
 return "Country{code='" + code + "', name='" + name + "'}";  
 }  
}

1. **Create Controller**
   * **File:** CountryController.java
   * **Location:** controller package

package com.cognizant.spring\_learn.controller;  
  
import com.cognizant.spring\_learn.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 = (Country) context.getBean("country");  
 *LOGGER*.info("End - getCountryIndia()");  
 return country;  
 }  
}

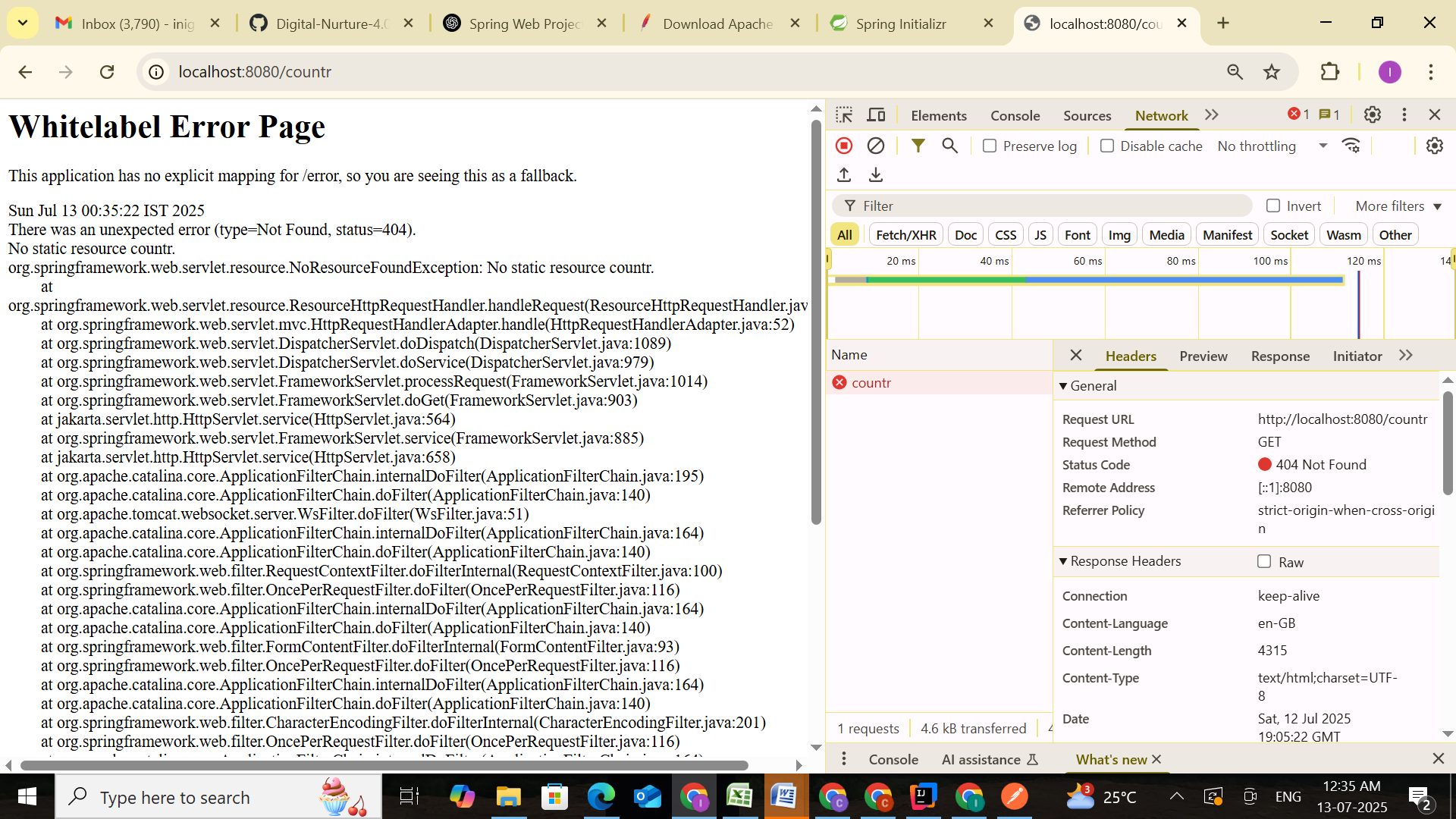
1. **Run the application**
2. **Test using browser or Postman**
   * **URL:** http://localhost:8080/country
   * **Output:** {"code":"IN","name":"India"}





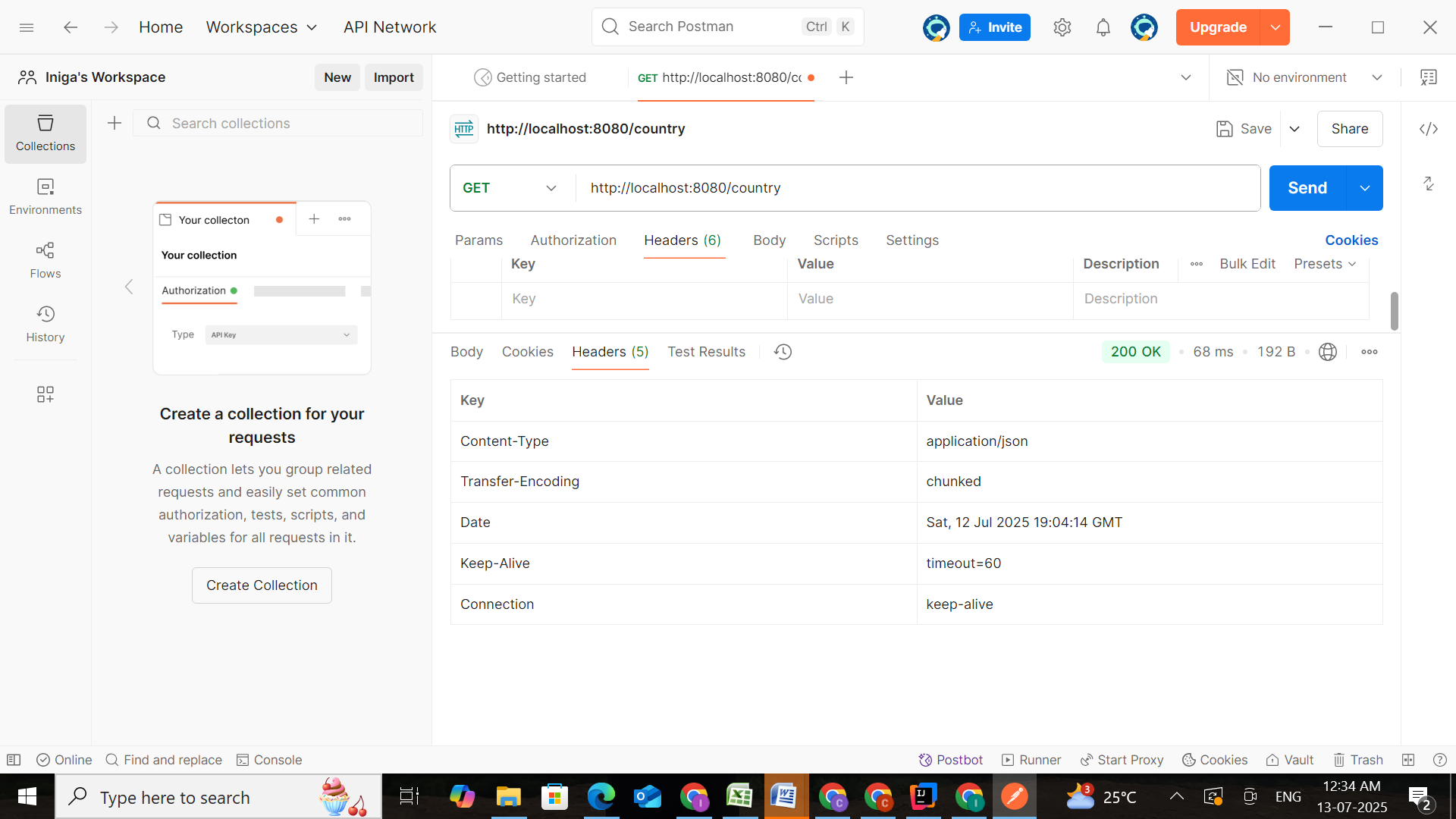
### SME Explanation

* **What happens in the controller method?**
  + The method loads country.xml from classpath.
  + Fetches the country bean using context.getBean().
  + Logs start and end.
  + Returns the country object.
* **How the bean is converted into JSON?**
  + Spring Boot includes Jackson (JSON library).
  + Jackson automatically serializes Java objects (POJOs) returned from controller to JSON.
* **HTTP Headers in Browser (Developer Tools):**
  + Content-Type: application/json
  + Status: 200 OK or 404 Not Found

****

**Example for NOT FOUND**

* **Postman Headers Tab:**
  + JSON content with country data.
  + Or 404 error with reason: Country not found.



Example for FOUND

**3. REST - Get Country Based on Country Code**

### Overview

This REST service returns a country object based on the given country code. It supports case-insensitive matching and throws an exception if not found.

### Endpoint Details

* **Method:** GET
* **URL:** /countries/{code}
* **Controller:** com.cognizant.spring\_learn.controller.CountryController
* **Method Name:** getCountry(String code)
* **Service:** com.cognizant.spring\_learn.service.CountryService

**Step-by-step:**

1. **Update 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="countryList" class="java.util.ArrayList">  
 <constructor-arg>  
 <list>  
 <bean class="com.cognizant.spring\_learn.Country">  
 <property name="code" value="IN"/>  
 <property name="name" value="India"/>  
 </bean>  
 <bean class="com.cognizant.spring\_learn.Country">  
 <property name="code" value="US"/>  
 <property name="name" value="United States"/>  
 </bean>  
 <bean class="com.cognizant.spring\_learn.Country">  
 <property name="code" value="DE"/>  
 <property name="name" value="Germany"/>  
 </bean>  
 <bean class="com.cognizant.spring\_learn.Country">  
 <property name="code" value="JP"/>  
 <property name="name" value="Japan"/>  
 </bean>  
 </list>  
 </constructor-arg>  
 </bean>  
</beans>

1. **Create CountryNotFoundException**
   * Location: com.cognizant.spring\_learn.service.exception

package com.cognizant.spring\_learn.service.exception;  
  
import org.springframework.http.HttpStatus;  
import org.springframework.web.bind.annotation.ResponseStatus;  
  
@ResponseStatus(value = HttpStatus.*NOT\_FOUND*, reason = "Country not found")  
public class CountryNotFoundException extends Exception {  
 public CountryNotFoundException(String message) {  
 super(message);  
 }  
}

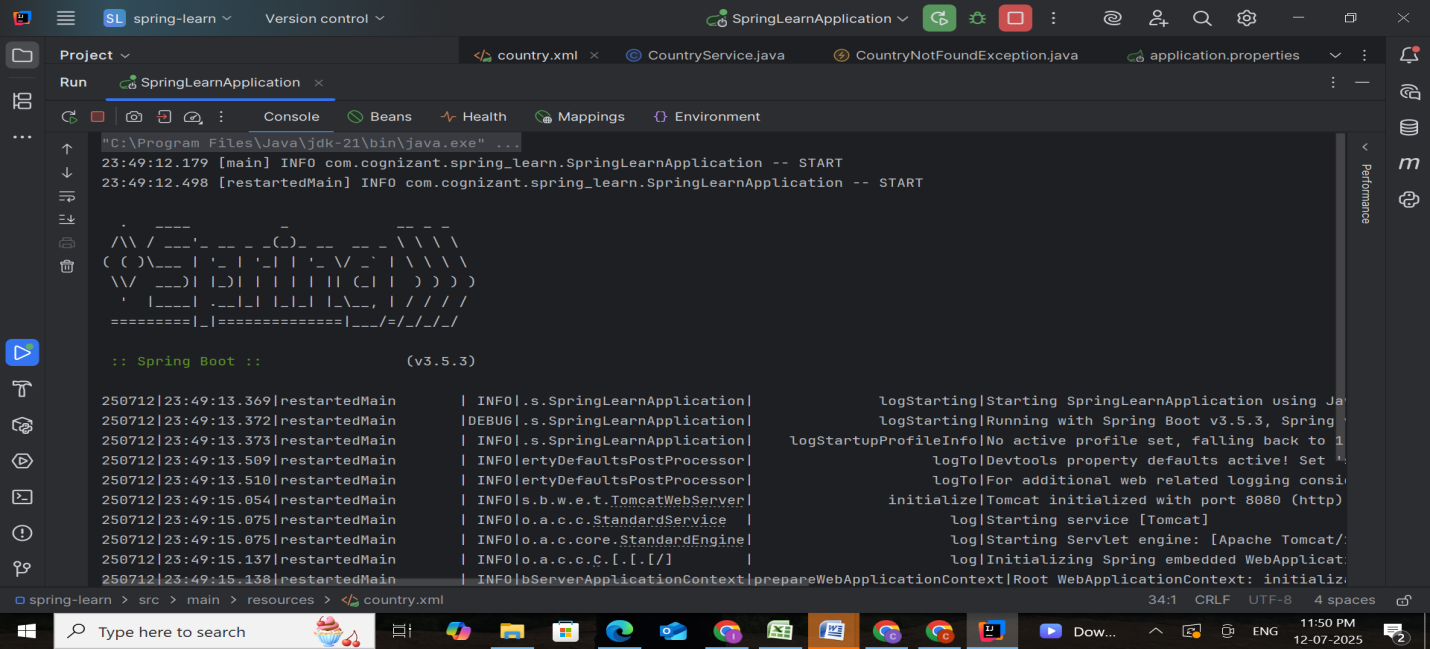
1. **Create CountryService**
   * Location: com.cognizant.spring\_learn.service

package com.cognizant.spring\_learn.service;  
  
import com.cognizant.spring\_learn.Country;  
import com.cognizant.spring\_learn.service.exception.CountryNotFoundException;  
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) throws CountryNotFoundException {  
 ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");  
 List<Country> countryList = context.getBean("countryList", List.class);  
 return countryList.stream()  
 .filter(c -> c.getCode().equalsIgnoreCase(code))  
 .findFirst()  
 .orElseThrow(() -> new CountryNotFoundException("Country Not Found"));  
 }  
}

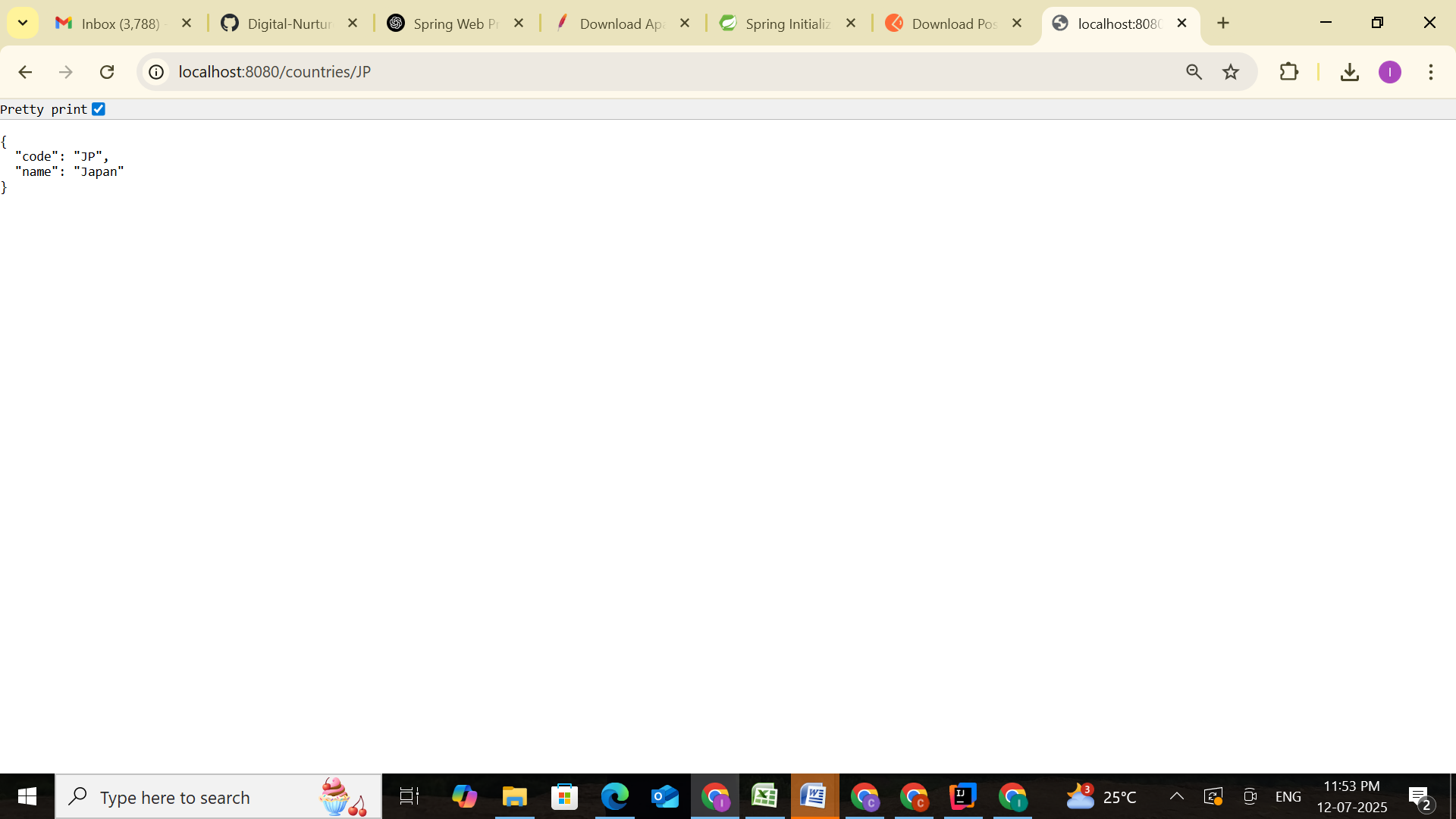
1. **Update CountryController**
   * Inject the CountryService and add method:

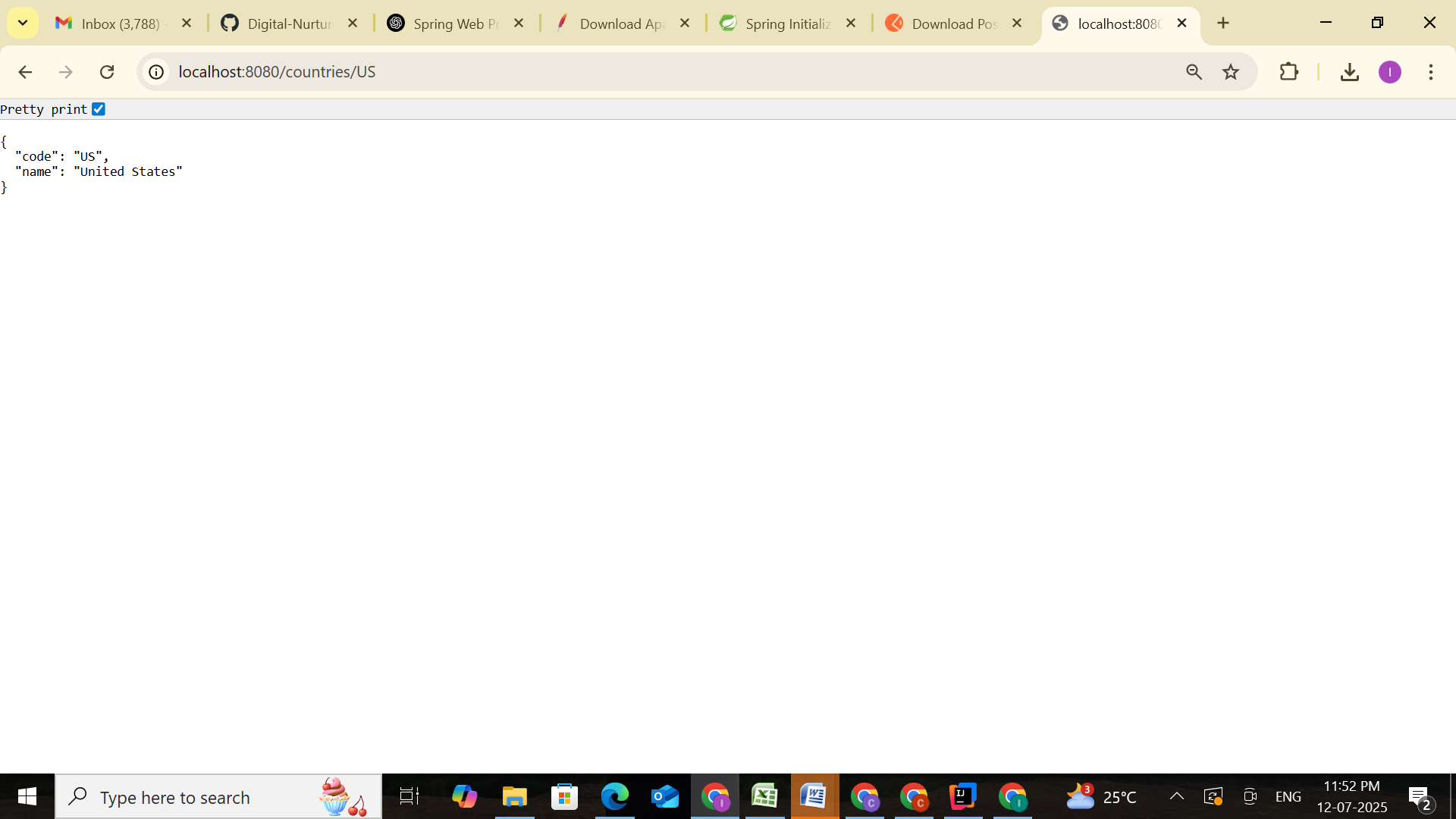
package com.cognizant.spring\_learn.controller;  
  
import com.cognizant.spring\_learn.Country;  
import com.cognizant.spring\_learn.service.CountryService;  
import com.cognizant.spring\_learn.service.exception.CountryNotFoundException;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
@RestController  
public class CountryController {  
  
 @Autowired  
 CountryService countryService;  
  
 @GetMapping("/countries/{code}")  
 public Country getCountry(@PathVariable String code) throws CountryNotFoundException {  
 return countryService.getCountry(code);  
 }  
}

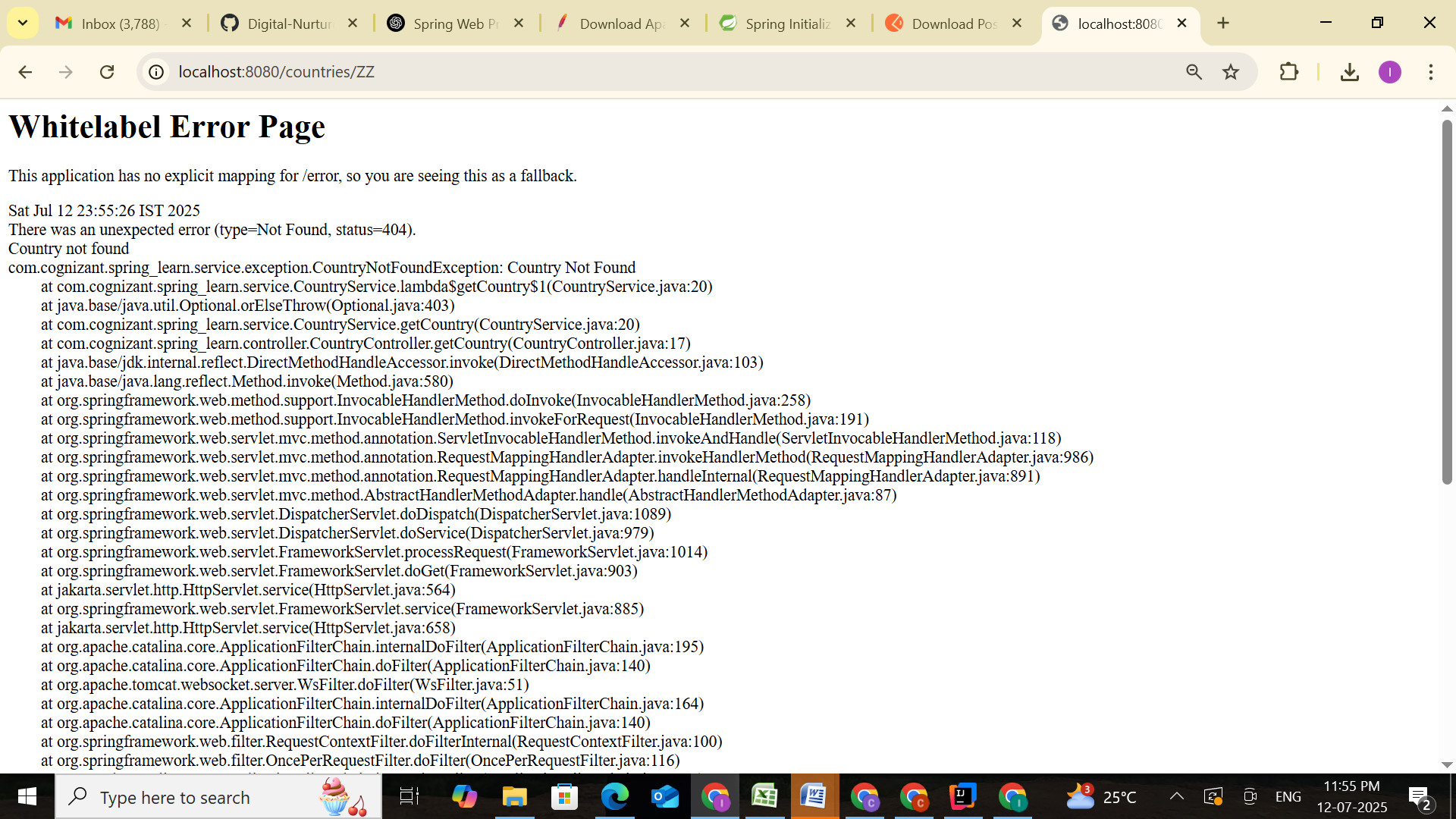
1. **Run the application**



1. **Test with Postman**
   * URL: http://localhost:8080/countries/US → Should return United States
   * URL: http://localhost:8080/countries/ZZ → Should return 404

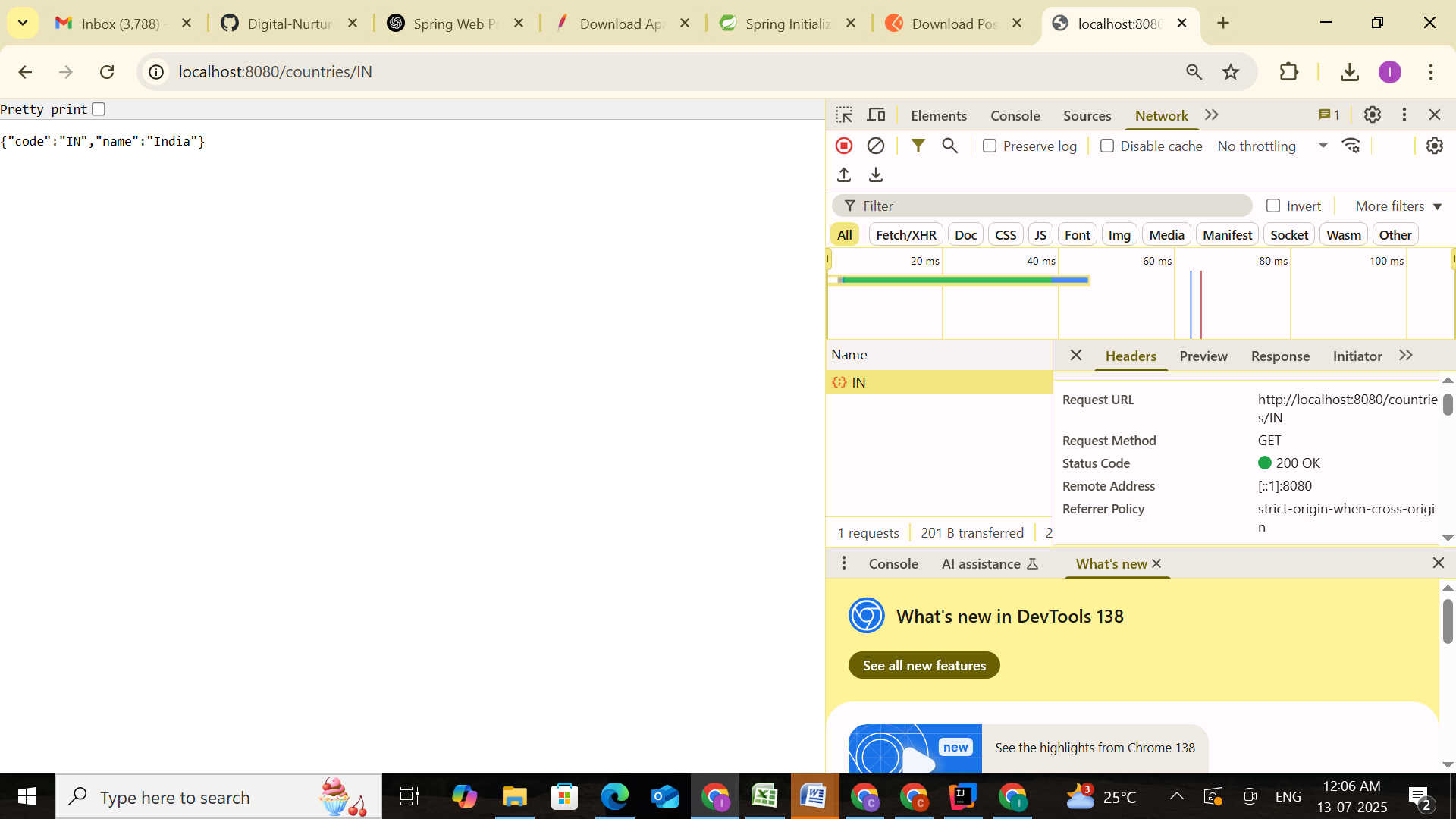






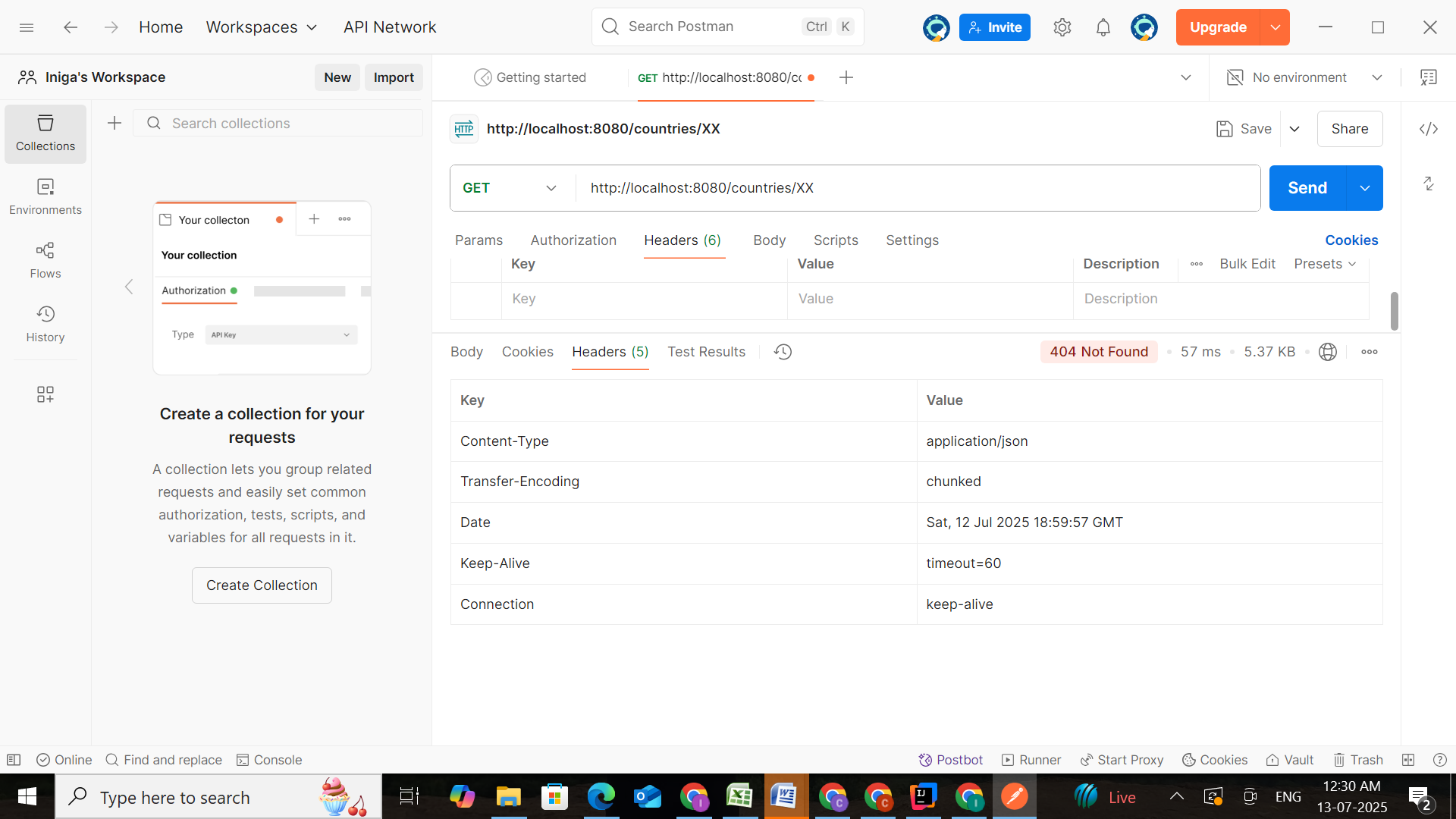
### SME Explanation

* **How the controller works:**
  + Accepts the path variable code.
  + Passes it to countryService.getCountry(code).
* **How bean is fetched:**
  + Loads countryList bean from country.xml.
  + Performs case-insensitive match using Java 8 Stream API.
* **When no match found:**
  + CountryNotFoundException is thrown.
  + @ResponseStatus(HttpStatus.NOT\_FOUND) returns 404.
* **Browser Headers (Developer Tools):**
  + Content-Type: application/json
  + Status: 200 OK or 404 Not Found

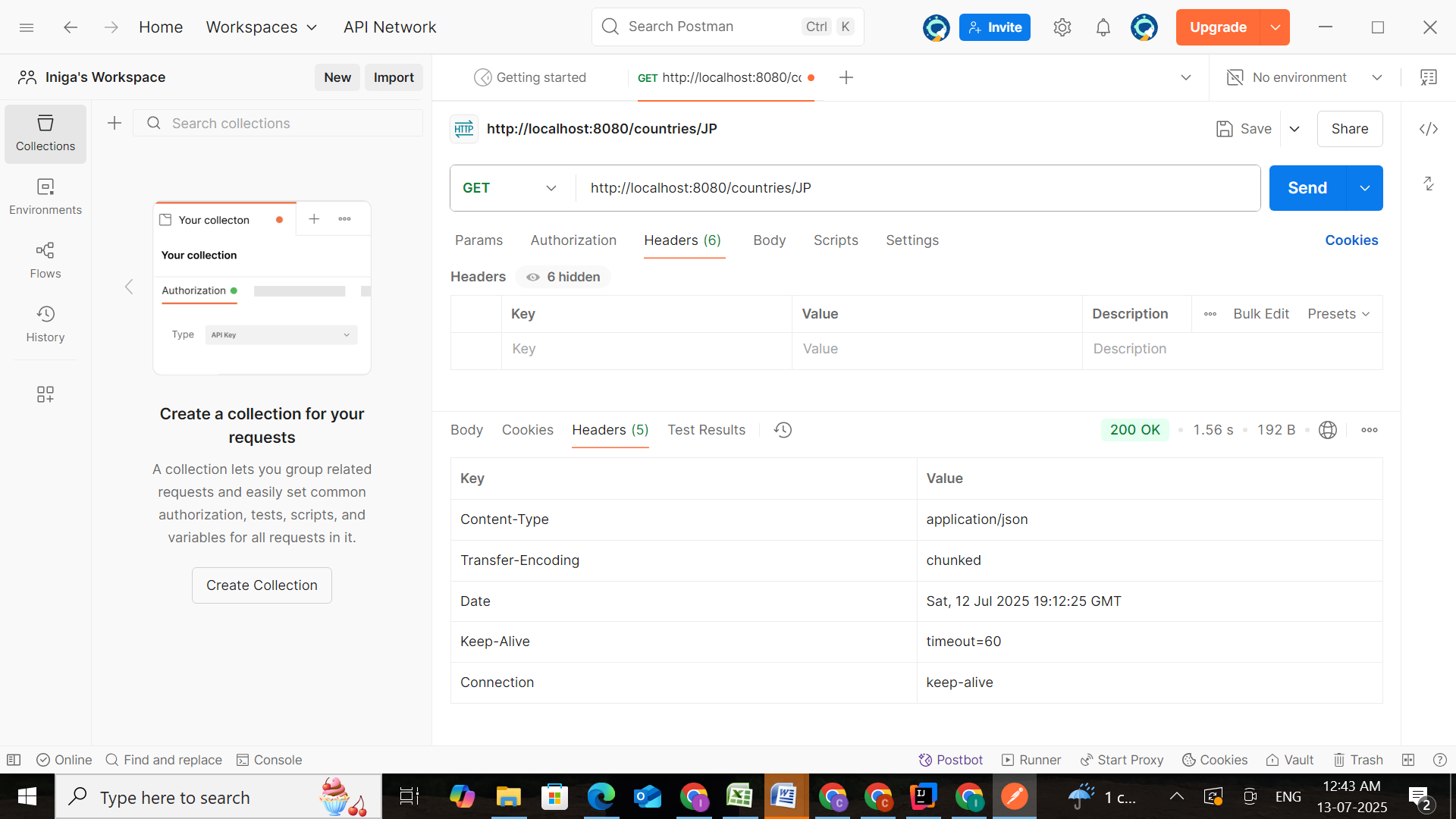


Example for FOUND

* **Postman Headers:**
  + JSON content with country data.
  + Or 404 error with reason: Country not found.



Example for NOT FOUND



Example for FOUND