**Hands-on 1:**

**Create a Spring Web Project using Maven** 

**Step 1:Generate a Spring Boot Project**

1. Open your browser and go to: https://start.spring.io  
2. Set the following options:  
 - Group: com.cognizant  
 - Artifact: spring-learn  
3. Add the following dependencies:  
 - Spring Web  
 - Spring Boot DevTools  
4. Click Generate, and a zip file will be downloaded.  
5. Extract the zip file to a known location, like your Eclipse workspace:  
 C:\Users\THANUSHYA T S\Downloads\spring-learn

# Step 2: Import into Eclipse

1. Open Eclipse.  
2. Go to: File > Import > Maven > Existing Maven Projects  
3. Browse to the folder where the project was extracted.  
4. Click Finish to import the project.

# Step 3: Build the Project Using Maven

Open Command Prompt and run: **mvn clean package -Dhttp.proxyHost=proxy.cognizant.com -Dhttp.proxyPort=6050 -Dhttps.proxyHost=proxy.cognizant.com -Dhttps.proxyPort=6050 -Dhttp.proxyUser=123456**  
Ensure you are in the directory where the pom.xml file is located.

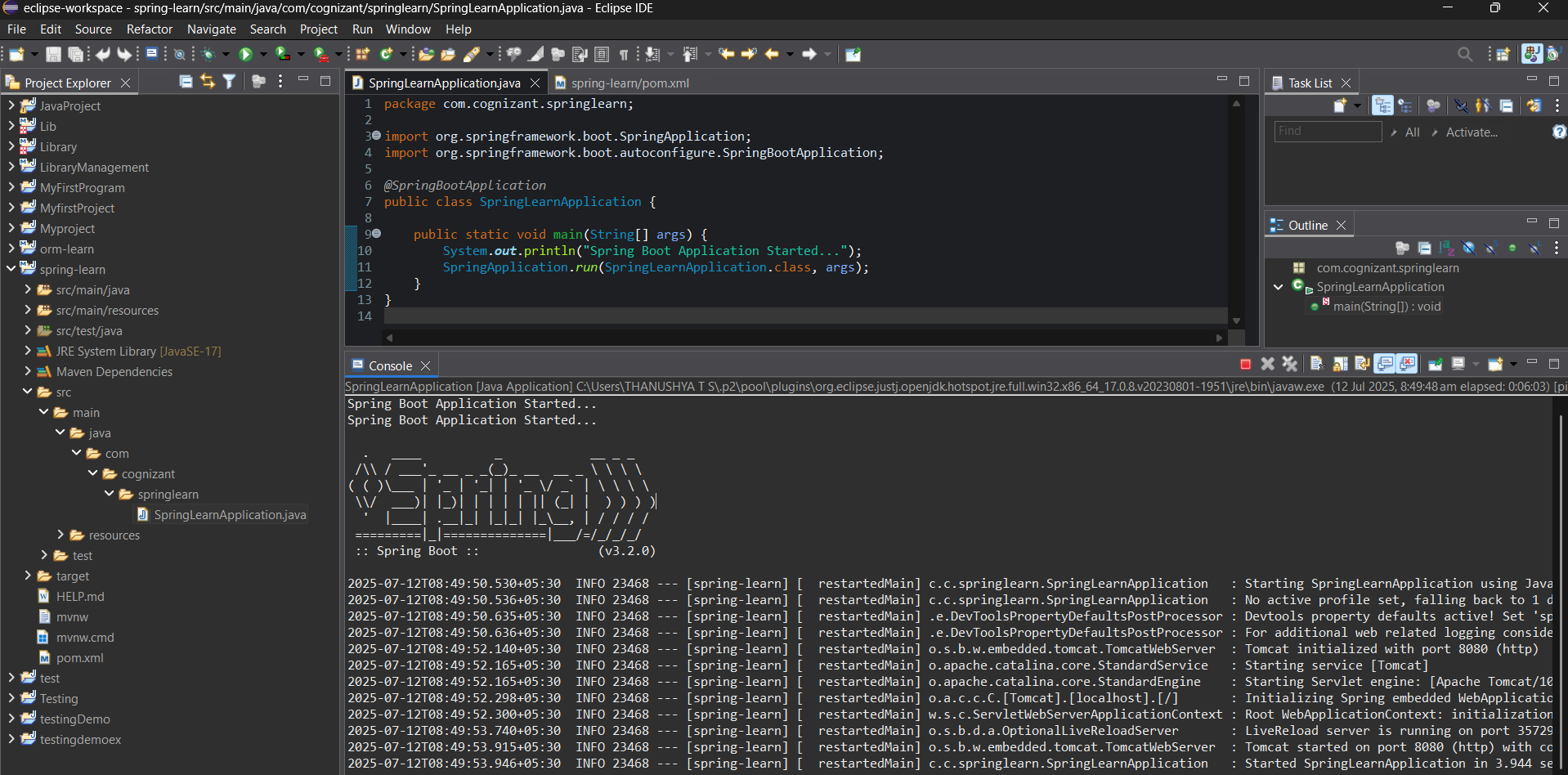
# Step 4: Run the Spring Boot Application

1. In Eclipse, open:  
 src/main/java/com/cognizant/springlearn/SpringLearnApplication.java  
  
2. The file should contain:

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) {  
 System.out.println("Spring Boot Application Started...");  
 SpringApplication.run(SpringLearnApplication.class, args);  
 }  
}

3. Right-click the class and select Run As > Java Application  
4. You should see logs in the Eclipse console that include:  
 - Spring Boot Application Started...  
 - Tomcat started on port 8080  
  
Your application is now running!

**OUTPUT:**



**Hands-On 2:**

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

**Step 1: Create Spring Configuration XML**

1. In Eclipse, go to: src/main/resources
2. 2. Create a new XML file: date-format.xml

**Step 2: Add Bean Definition**

Paste the following into date-format.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="dateFormat" class="java.text.SimpleDateFormat">  
 <constructor-arg value="dd/MM/yyyy" />  
 </bean>  
  
</beans>

# Step 3: Modify SpringLearnApplication.java

Ensure the following imports are present at the top:

import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
import java.text.SimpleDateFormat;  
import java.text.ParseException;  
import java.util.Date;

Add the method below:

public static void displayDate() {  
 ApplicationContext context = new ClassPathXmlApplicationContext("date-format.xml");  
 SimpleDateFormat format = context.getBean("dateFormat", SimpleDateFormat.class);  
 try {  
 Date date = format.parse("31/12/2018");  
 System.out.println("Parsed Date: " + date);  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
}

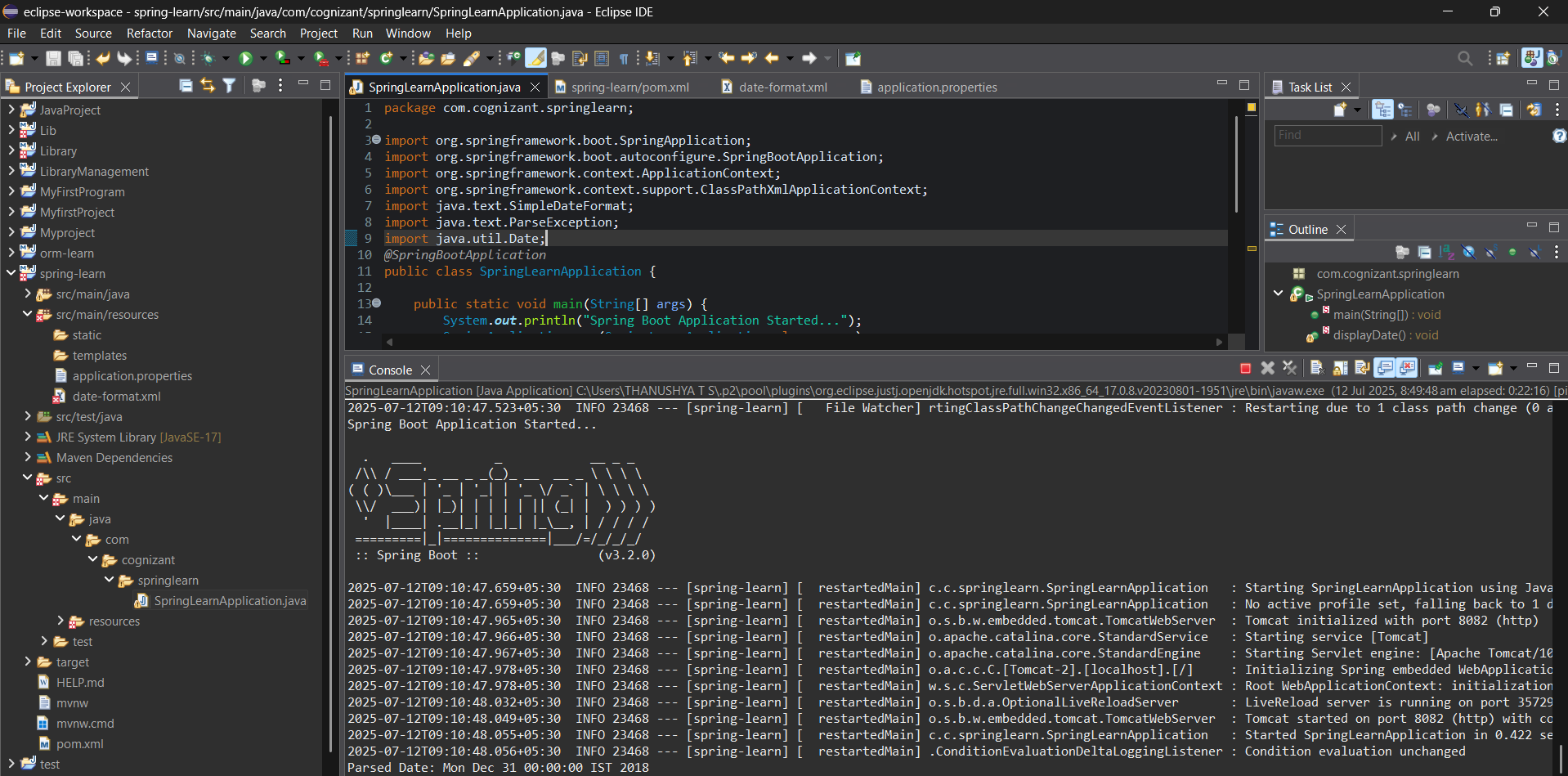
Update your main() method as:

public static void main(String[] args) {  
 System.out.println("Spring Boot Application Started...");  
 SpringApplication.run(SpringLearnApplication.class, args);  
 displayDate();  
}

# Step 4: Run the Application

1. Right-click on SpringLearnApplication.java
2. 2. Select Run As > Java Application
3. 3. Console should show: Parsed Date: Mon Dec 31 00:00:00 IST 2018

**OUTPUT:**



**Hands-on 3:**

**Hello World RESTful Web Service**

**Step 1:Create a Controller Class**

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;

}

}

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

server.port=8085

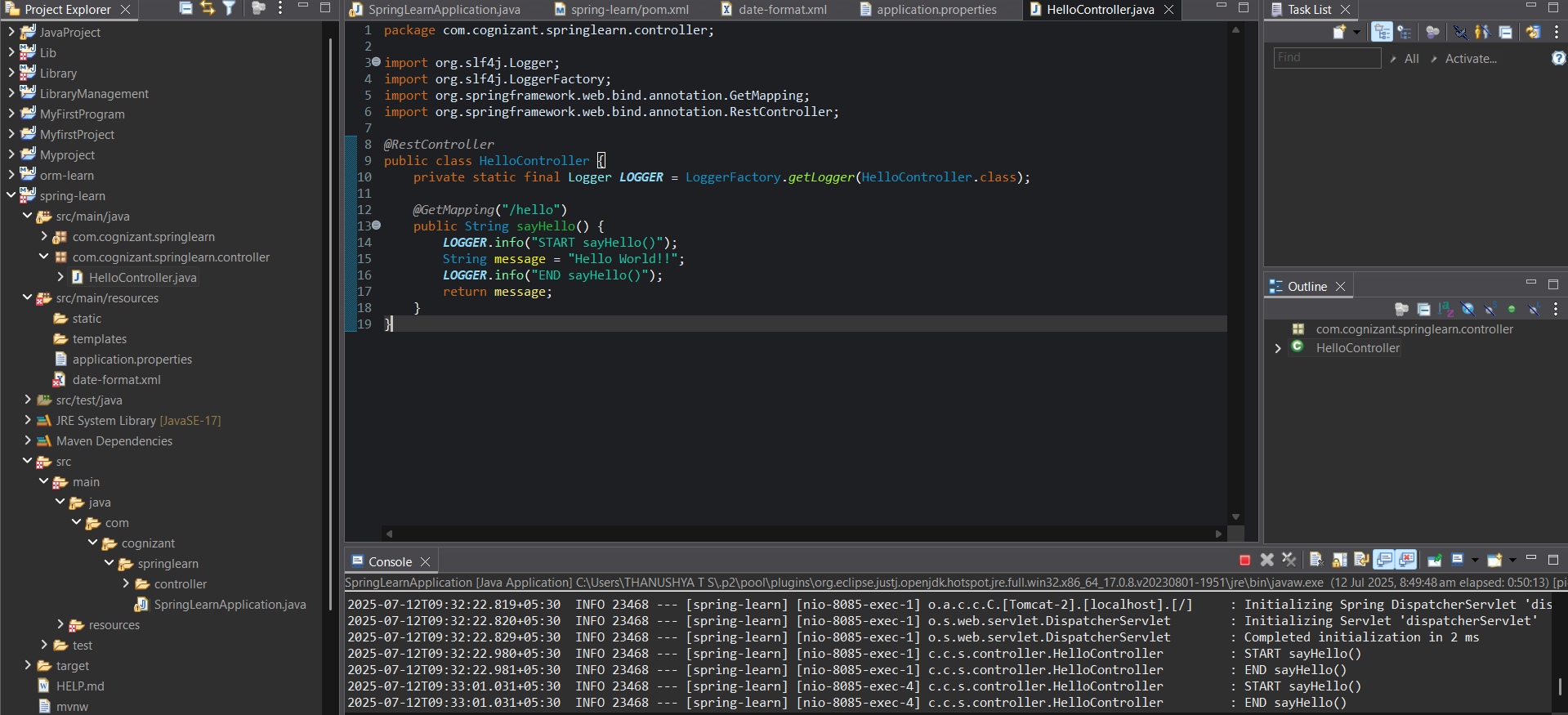
**Step 3:Run the Application**

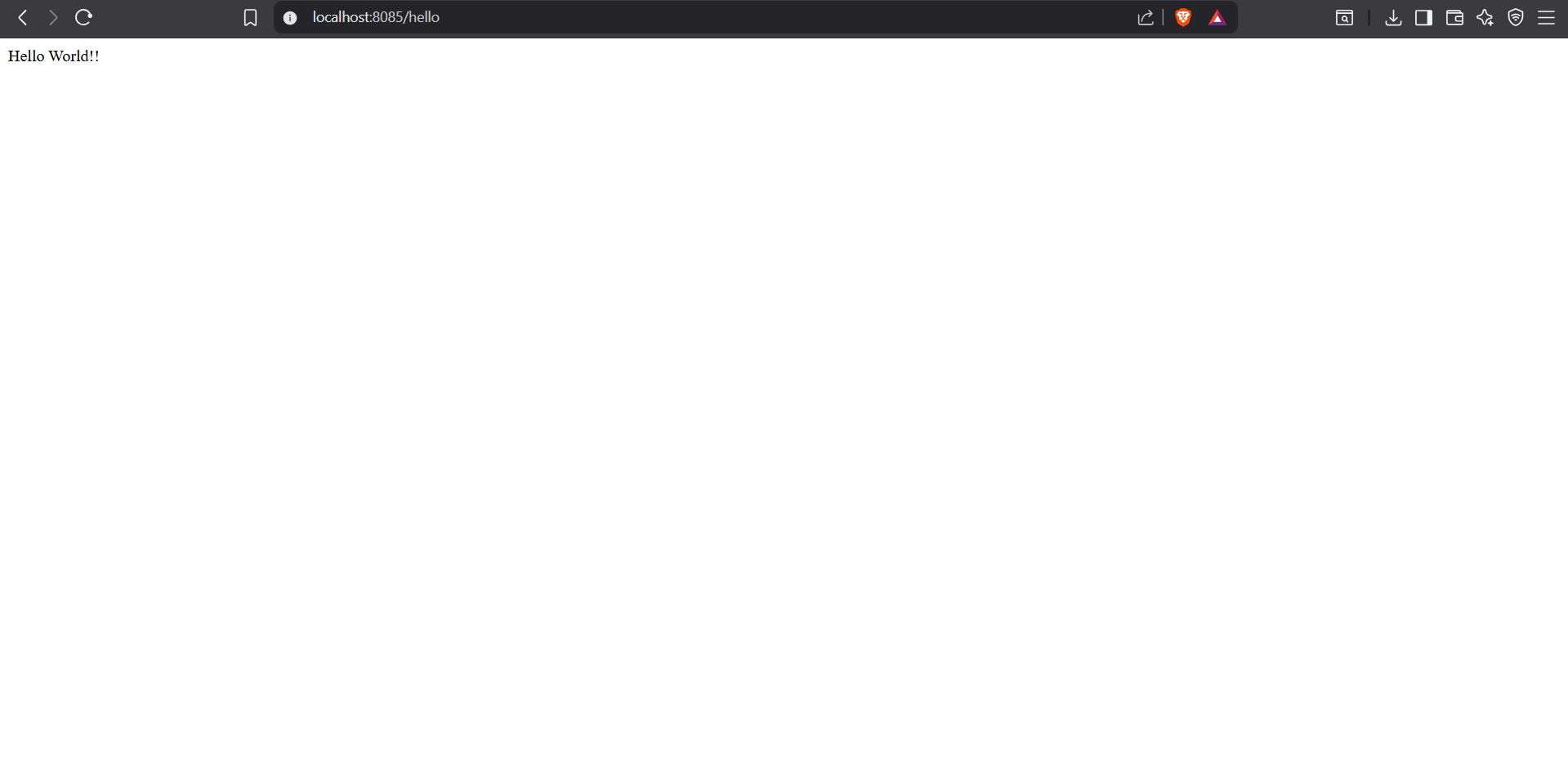
* Right-click SpringLearnApplication.java > Run As > Java Application
* Open browser and go to: <http://localhost:8085/hello>

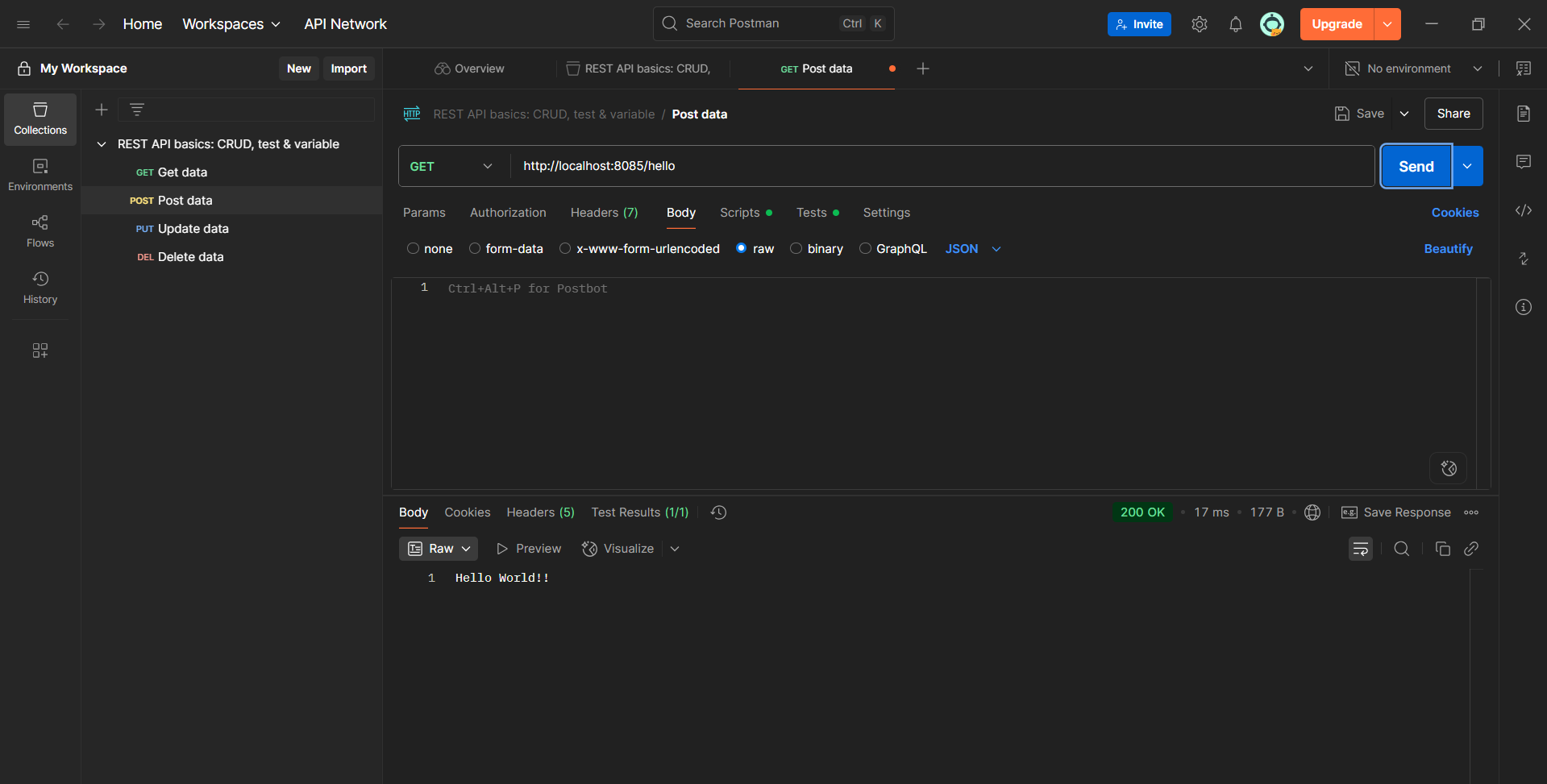
**Step 4:Test in Postman**

* Method: GET
* URL: http://localhost:8085/hello
* Expected Response: Hello World!!

**OUTPUT:**

****

****



**Hands-On 4: REST - Country Web Service**

### 1.Create Country Bean Definition in 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  
 https://www.springframework.org/schema/beans/spring-beans.xsd">  
  
 <bean id="in" class="com.cognizant.springlearn.model.Country">  
 <property name="code" value="IN" />  
 <property name="name" value="India" />  
 </bean>  
  
</beans>

### 2. Create Country Model

Location: src/main/java/com/cognizant/springlearn/model/Country.java

package com.cognizant.springlearn.model;  
  
public class Country {  
 private String code;  
 private String 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;  
 }  
}

### 3. Create REST Controller

Location: src/main/java/com/cognizant/springlearn/controller/CountryController.java

package com.cognizant.springlearn.controller;  
  
import com.cognizant.springlearn.model.Country;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
@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 = context.getBean("in", Country.class);  
 LOGGER.info("END getCountryIndia()");  
 return country;  
 }  
}

### 4. Add Port in application.properties

If port 8083 is already used, change it to a free one:

server.port=8085

### 5. Run the Application

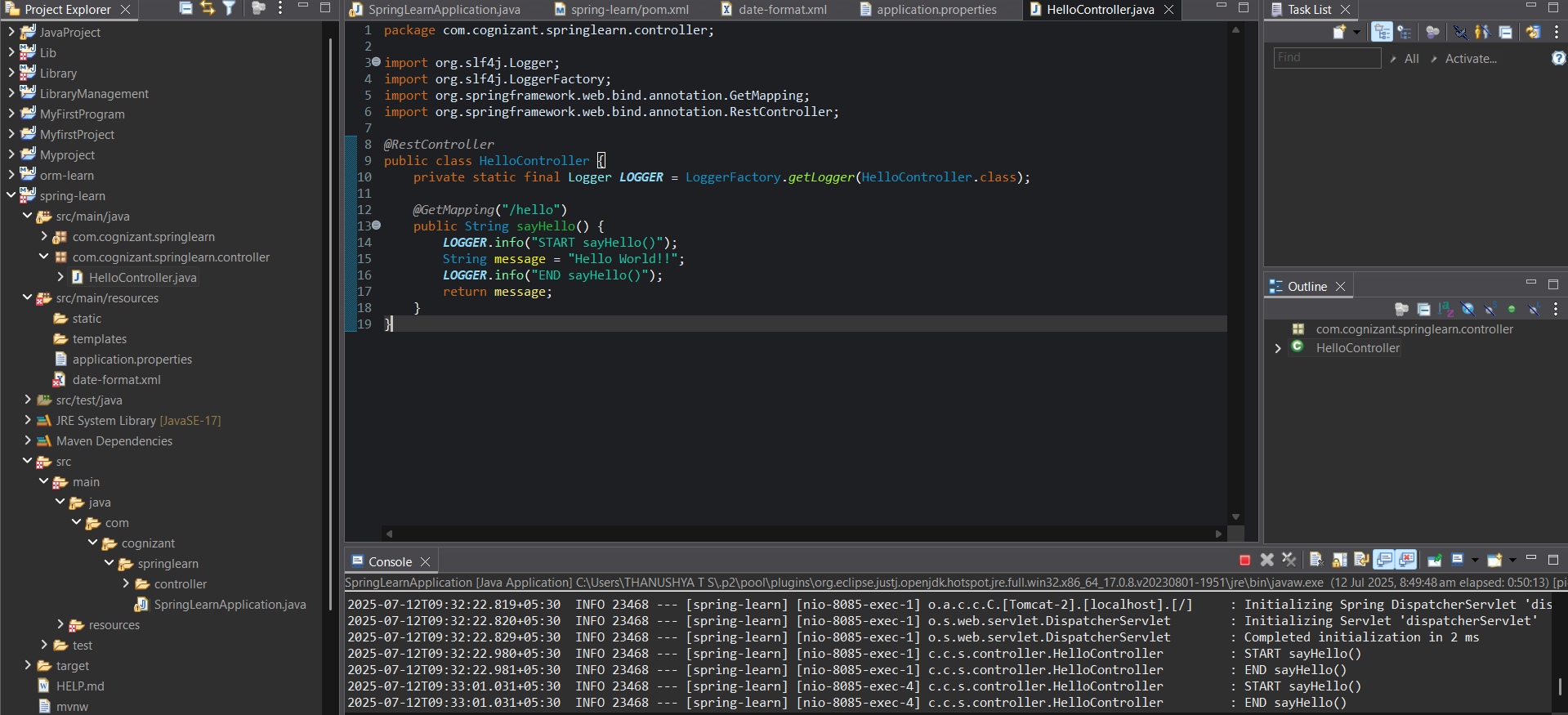
Right-click on SpringLearnApplication.java > Run As > Java Application

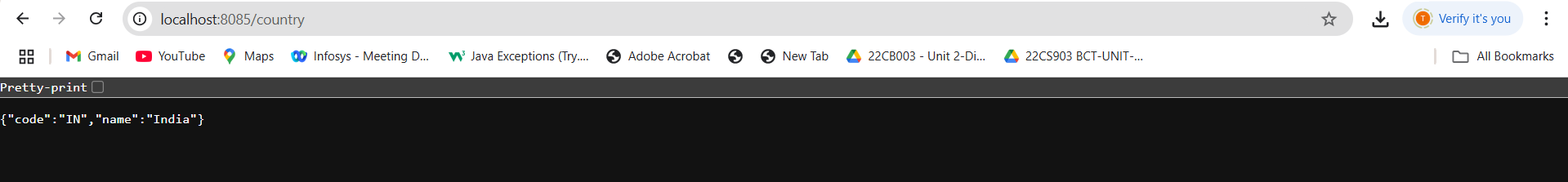
Open browser or Postman and go to: http://localhost:8085/country

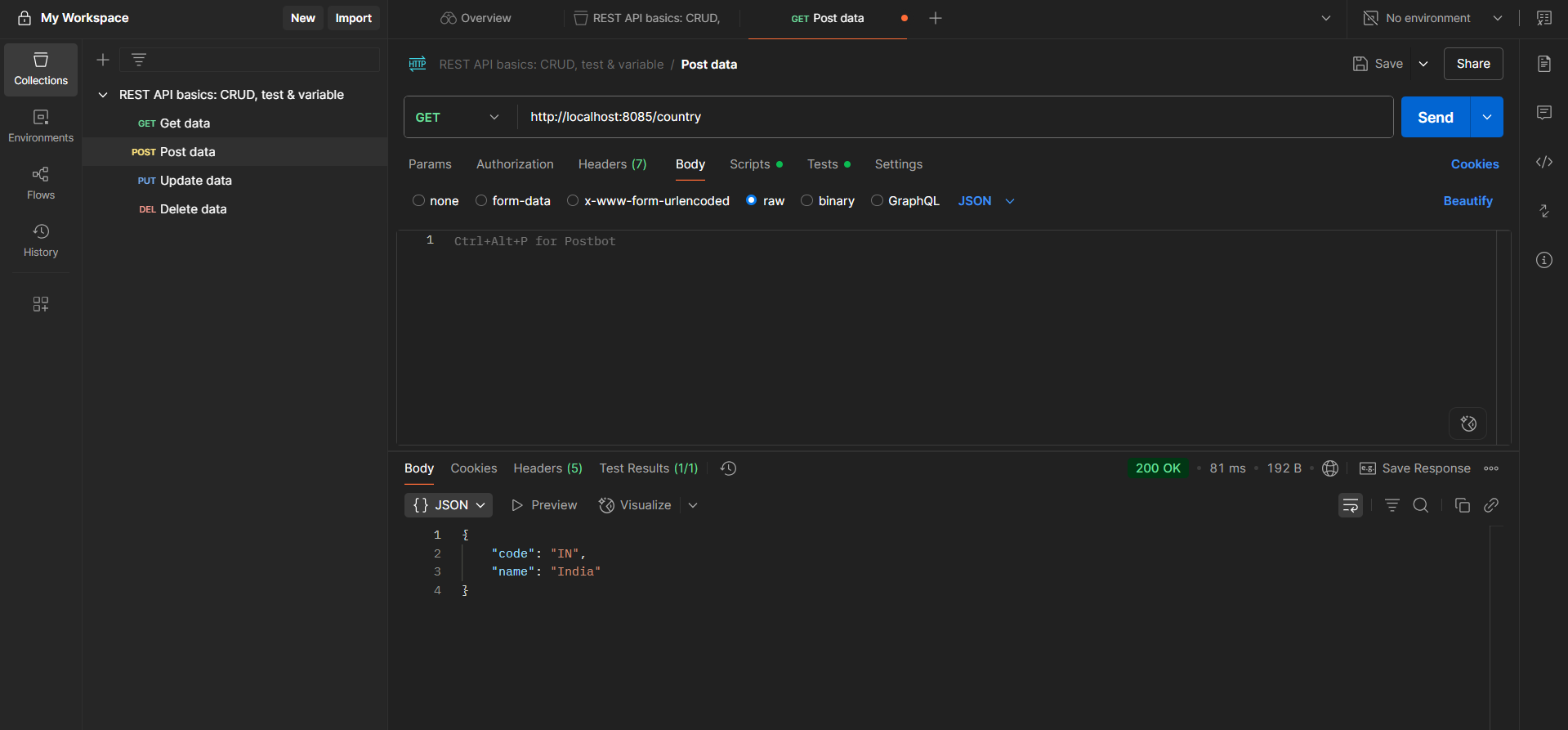
## 📄 Sample Response:

{  
 "code": "IN",  
 "name": "India"  
}

**OUTPUT:**

****

****

****

**Hands-On 5:** **REST - Get country based on country code**

### 1.Update country.xml to Include Multiple Countries

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  
 https://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>  
 </list>  
 </constructor-arg>  
 </bean>  
</beans>**

### 2. Create CountryService Class

Location: src/main/java/com/cognizant/springlearn/service/CountryService.java

**package com.cognizant.springlearn.service;  
  
import com.cognizant.springlearn.model.Country;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
import java.util.List;  
  
public class CountryService {  
  
 public Country getCountry(String code) {  
 ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");  
 List<Country> countries = context.getBean("countryList", List.class);  
  
 return countries.stream()  
 .filter(c -> c.getCode().equalsIgnoreCase(code))  
 .findFirst()  
 .orElse(null);  
 }  
}**

### 3. Update REST Controller

Location: src/main/java/com/cognizant/springlearn/controller/CountryController.java

**package com.cognizant.springlearn.controller;  
  
import com.cognizant.springlearn.model.Country;  
import com.cognizant.springlearn.service.CountryService;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.RestController;  
  
@RestController  
public class CountryController {  
  
 private static final Logger LOGGER = LoggerFactory.getLogger(CountryController.class);  
 private final CountryService service = new CountryService();  
  
 @GetMapping("/countries/{code}")  
 public Country getCountry(@PathVariable String code) {  
 LOGGER.info("START getCountry()");  
 Country country = service.getCountry(code);  
 LOGGER.info("END getCountry()");  
 return country;  
 }  
}**

### 4. Add Port in application.properties

Location: src/main/resources/application.properties

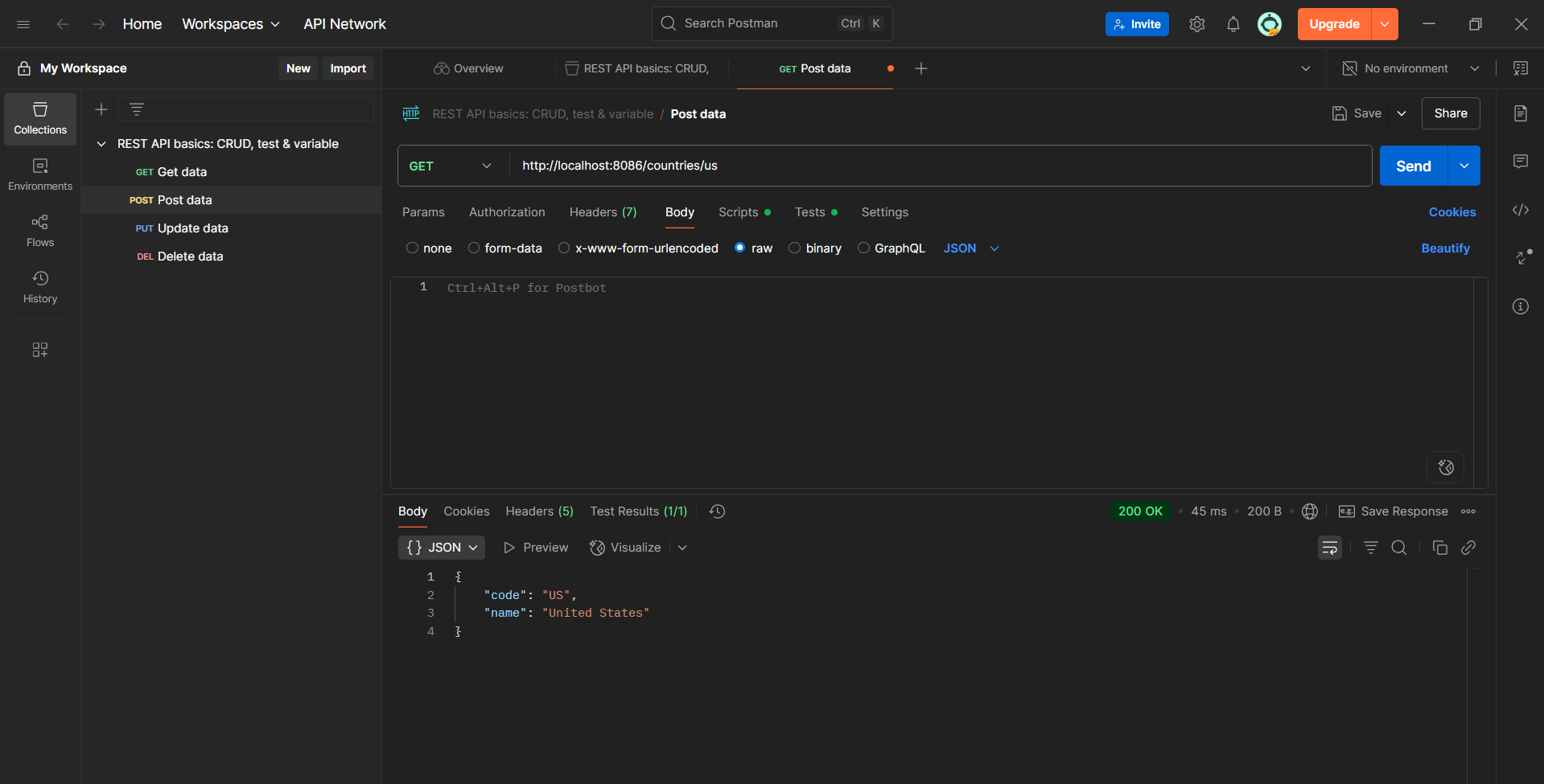
**server.port=8085**

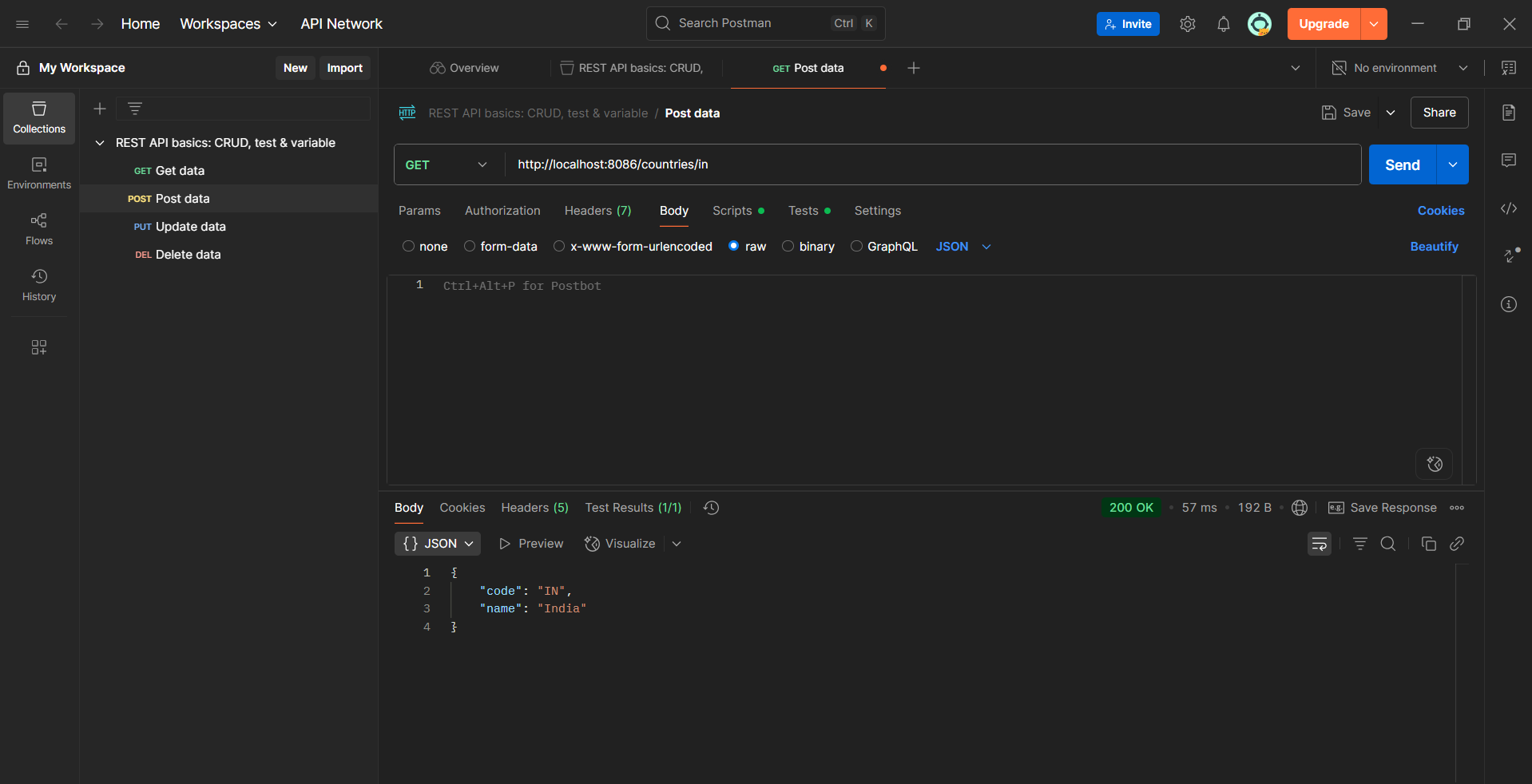
### 5. Run the Application

Right-click on SpringLearnApplication.java > Run As > Java Application

Open browser or Postman and go to: http://localhost:8085/countries/in

**OUTPUT:**

****



**HandsOn 6: Create authentication service that returns JWT**

**1. Project Setup**

Create a Maven Spring Boot project.  
Add the following dependencies in pom.xml:

<dependencies>  
 <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>

**2. Create JwtUtil Class**

package com.example.jwt.util;  
  
import io.jsonwebtoken.Jwts;  
import io.jsonwebtoken.SignatureAlgorithm;  
import org.springframework.stereotype.Component;  
  
import java.util.Date;  
  
@Component  
public class JwtUtil {  
  
 private final String SECRET\_KEY = "mysecretkey";  
  
 public String generateToken(String username) {  
 return Jwts.builder()  
 .setSubject(username)  
 .setIssuedAt(new Date(System.currentTimeMillis()))  
 .setExpiration(new Date(System.currentTimeMillis() + 1000 \* 60 \* 10)) // 10 min  
 .signWith(SignatureAlgorithm.HS256, SECRET\_KEY)  
 .compact();  
 }  
}

**3. Create AuthenticationController**

package com.example.jwt.controller;  
  
import com.example.jwt.util.JwtUtil;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.\*;  
import org.springframework.web.bind.annotation.\*;  
  
import java.nio.charset.StandardCharsets;  
import java.util.Base64;  
import java.util.Collections;  
  
@RestController  
public class AuthenticationController {  
  
 @Autowired  
 private JwtUtil jwtUtil;  
  
 @GetMapping("/authenticate")  
 public ResponseEntity<?> authenticate(@RequestHeader(HttpHeaders.AUTHORIZATION) String authHeader) {  
 if (authHeader != null && authHeader.startsWith("Basic ")) {  
 String base64Credentials = authHeader.substring("Basic ".length());  
 byte[] decodedBytes = Base64.getDecoder().decode(base64Credentials);  
 String credentials = new String(decodedBytes, StandardCharsets.UTF\_8);  
  
 String[] values = credentials.split(":", 2);  
 String username = values[0];  
 String password = values[1];  
  
 if ("user".equals(username) && "pwd".equals(password)) {  
 String token = jwtUtil.generateToken(username);  
 return ResponseEntity.ok(Collections.singletonMap("token", token));  
 } else {  
 return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body("Invalid credentials");  
 }  
 }  
 return ResponseEntity.status(HttpStatus.BAD\_REQUEST).body("Missing or invalid Authorization header");  
 }  
}

**4. Security Configuration**

package com.example.jwt.config;  
  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.core.userdetails.User;  
import org.springframework.security.core.userdetails.UserDetails;  
import org.springframework.security.core.userdetails.UserDetailsService;  
import org.springframework.security.provisioning.InMemoryUserDetailsManager;  
import org.springframework.security.web.SecurityFilterChain;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
  
@Configuration  
public class SecurityConfig {  
  
 @Bean  
 public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {  
 http.csrf().disable()  
 .authorizeHttpRequests()  
 .requestMatchers("/authenticate").permitAll()  
 .anyRequest().authenticated()  
 .and()  
 .httpBasic();  
 return http.build();  
 }  
  
 @Bean  
 public UserDetailsService userDetailsService() {  
 UserDetails user = User.withUsername("user")  
 .password("{noop}pwd") // {noop} means plain text password  
 .roles("USER")  
 .build();  
 return new InMemoryUserDetailsManager(user);  
 }  
}

**OUTPUT:**

