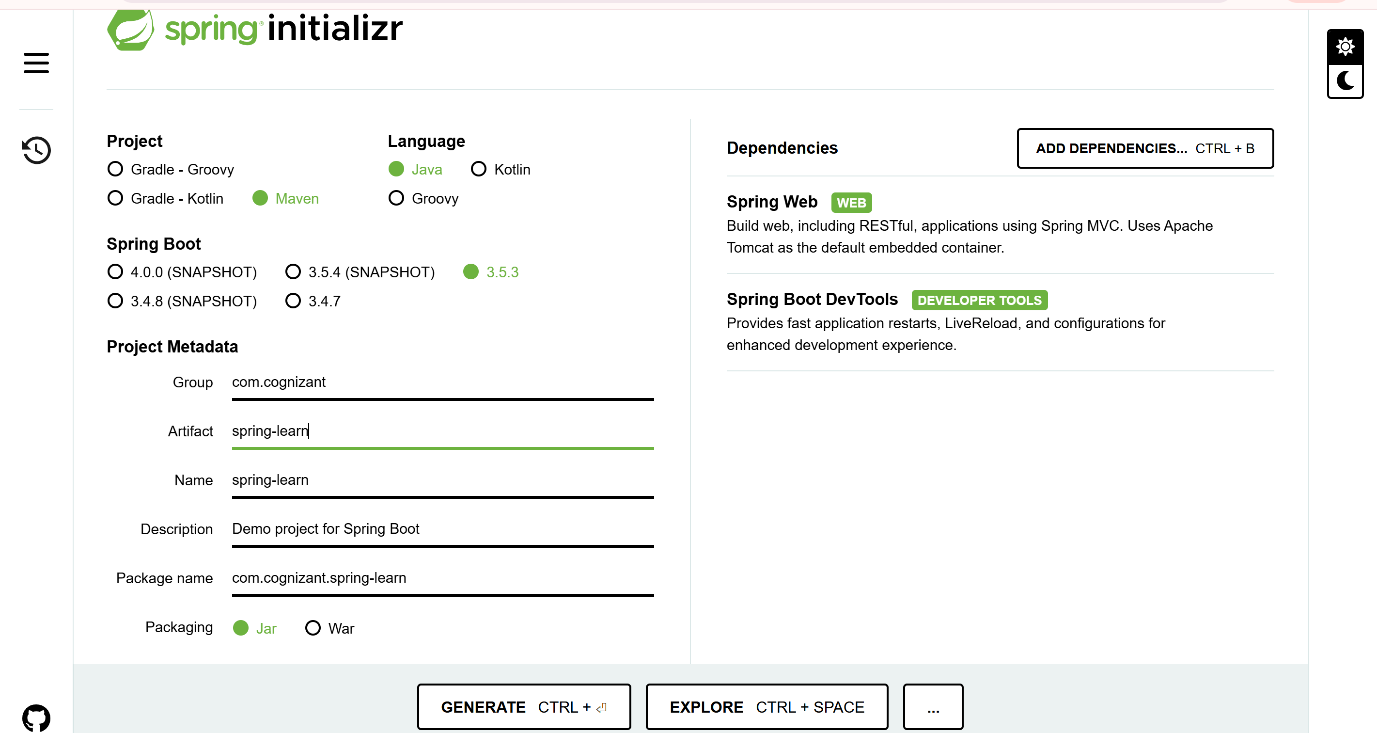
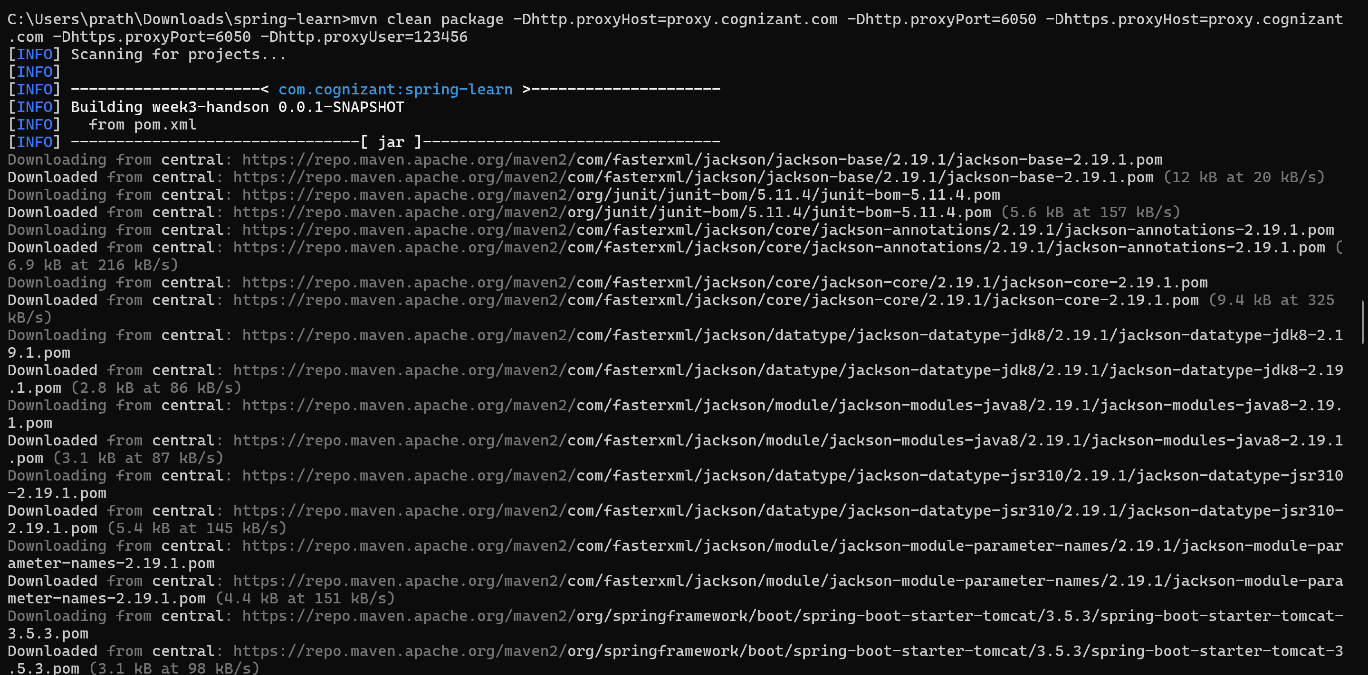
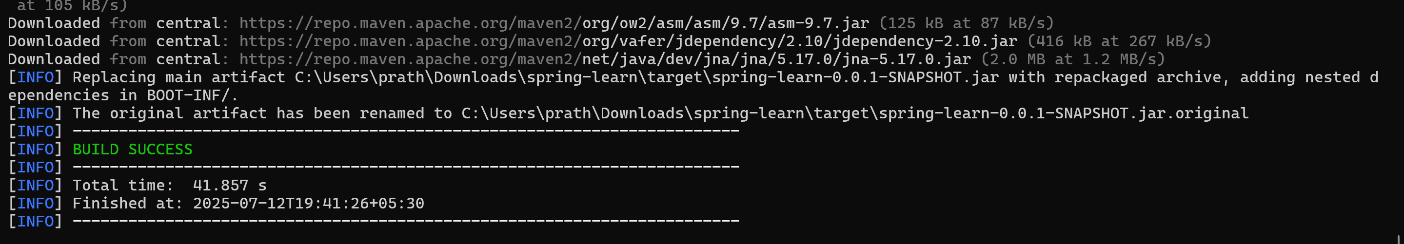
**SPRING REST USING SPRING BOOT 3**

**1. Create a Spring Web Project using Maven:  
  
Step 1:**

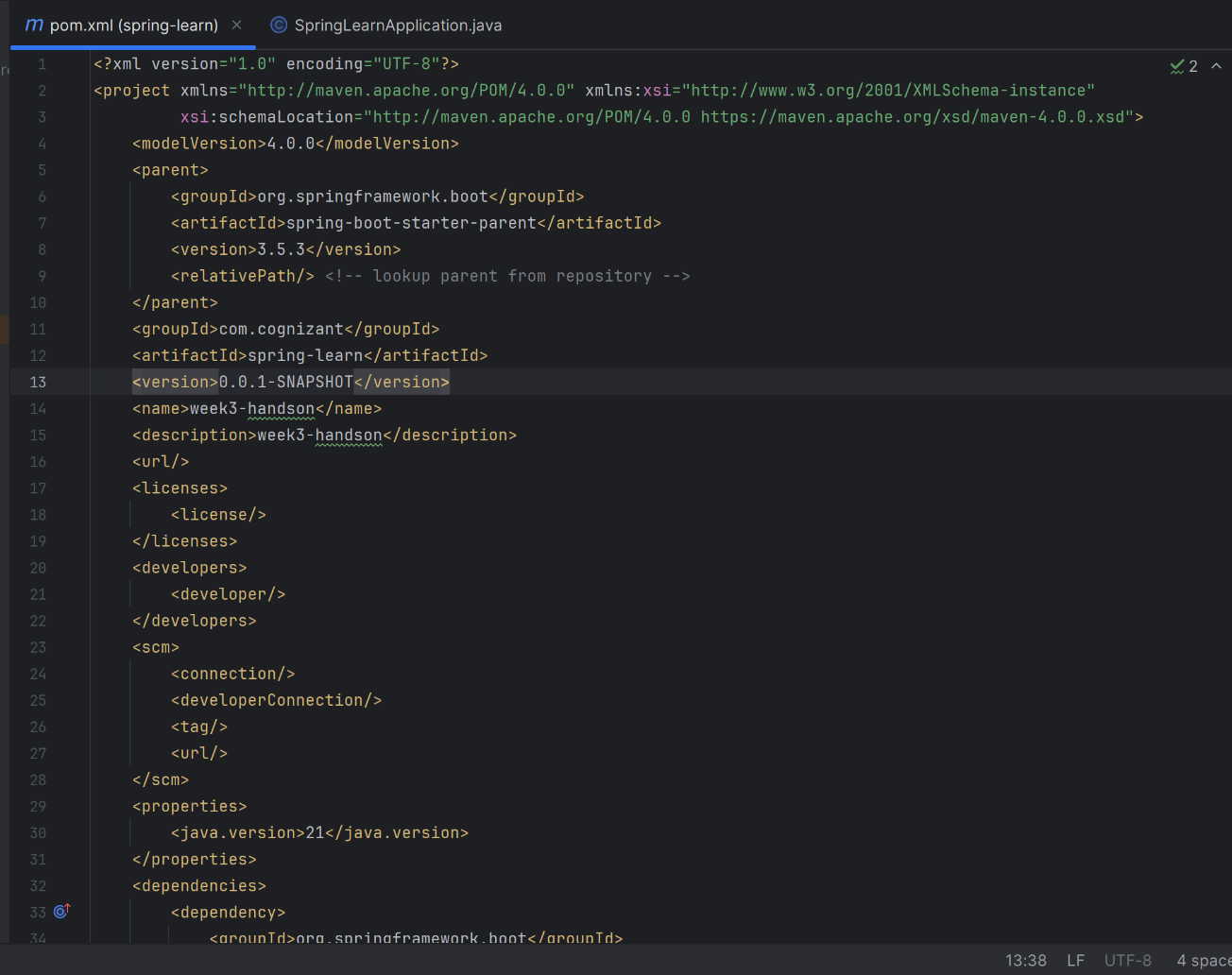
****

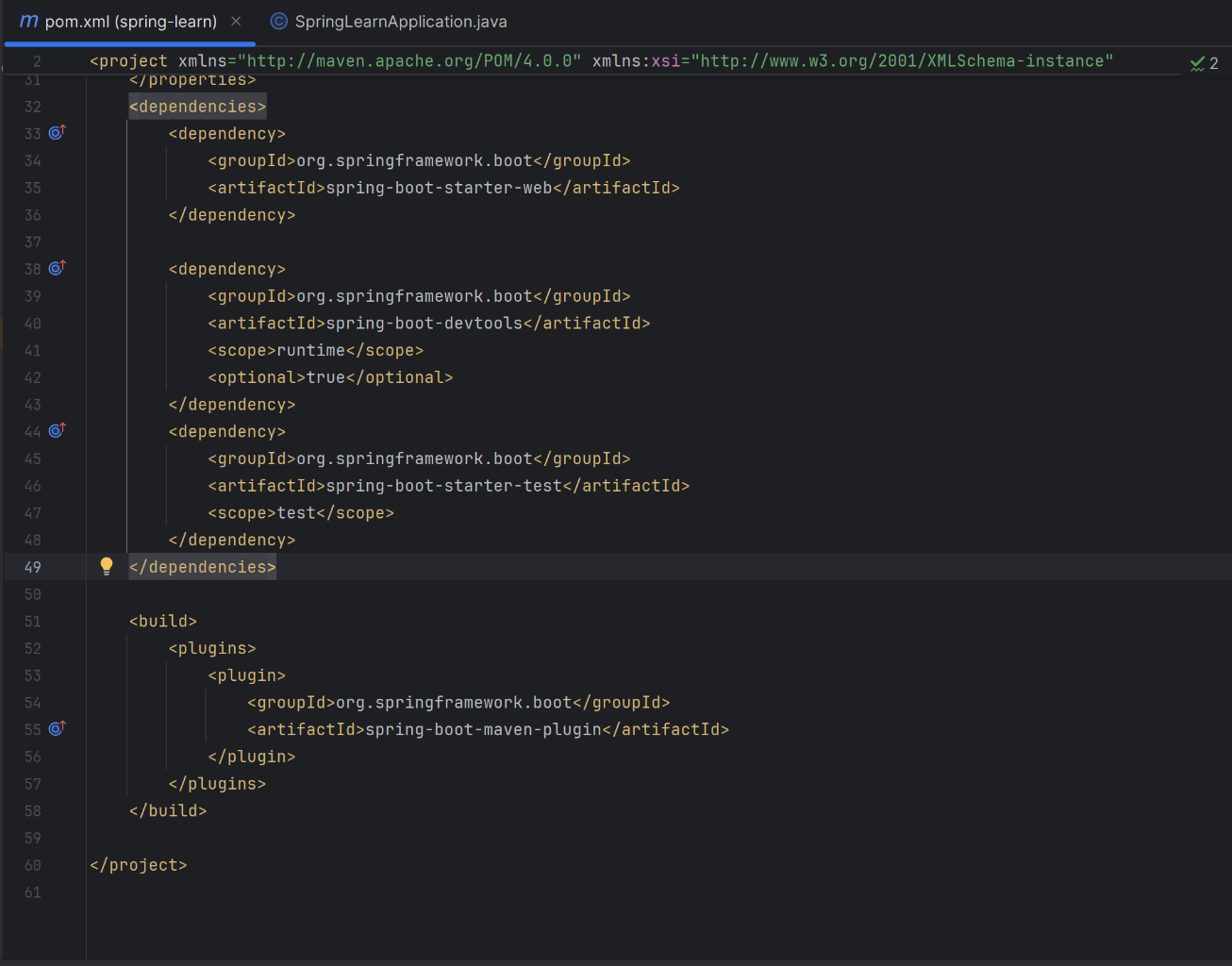
**Step 2 :**

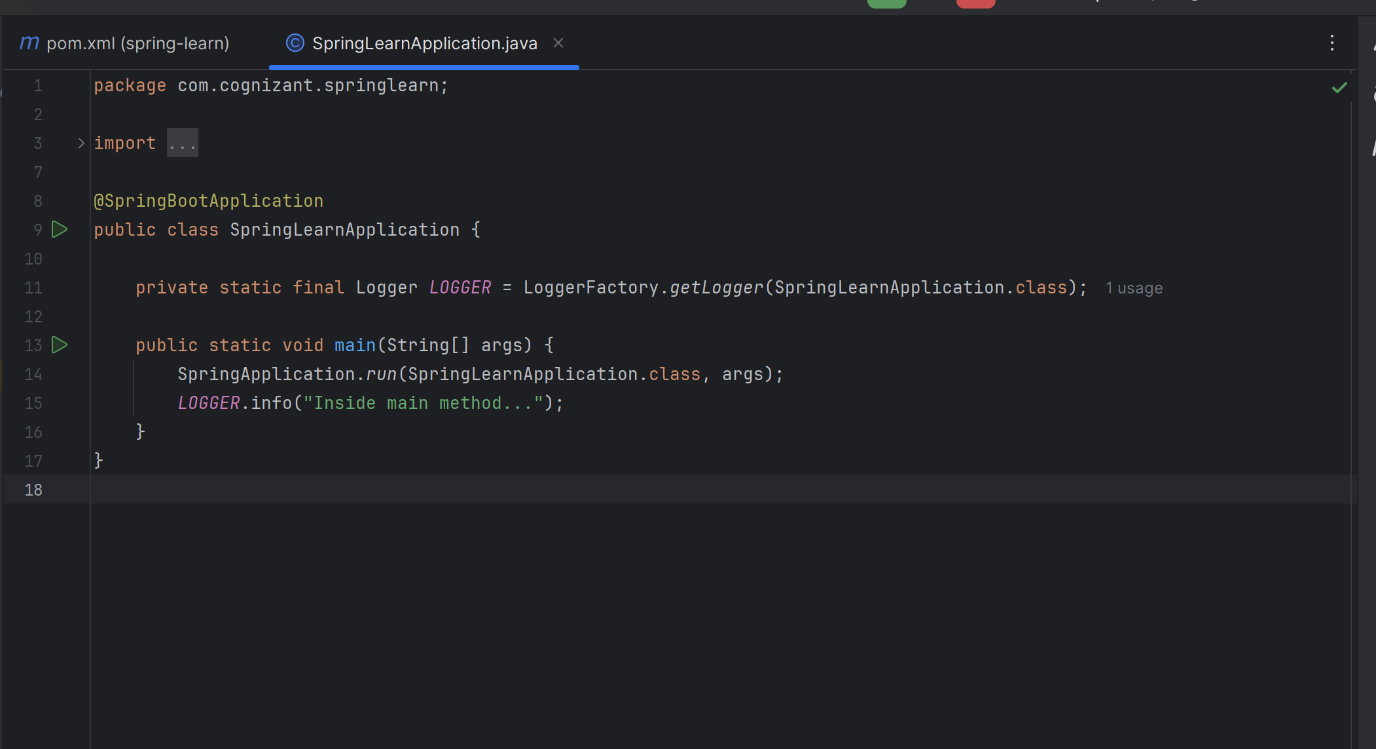
****

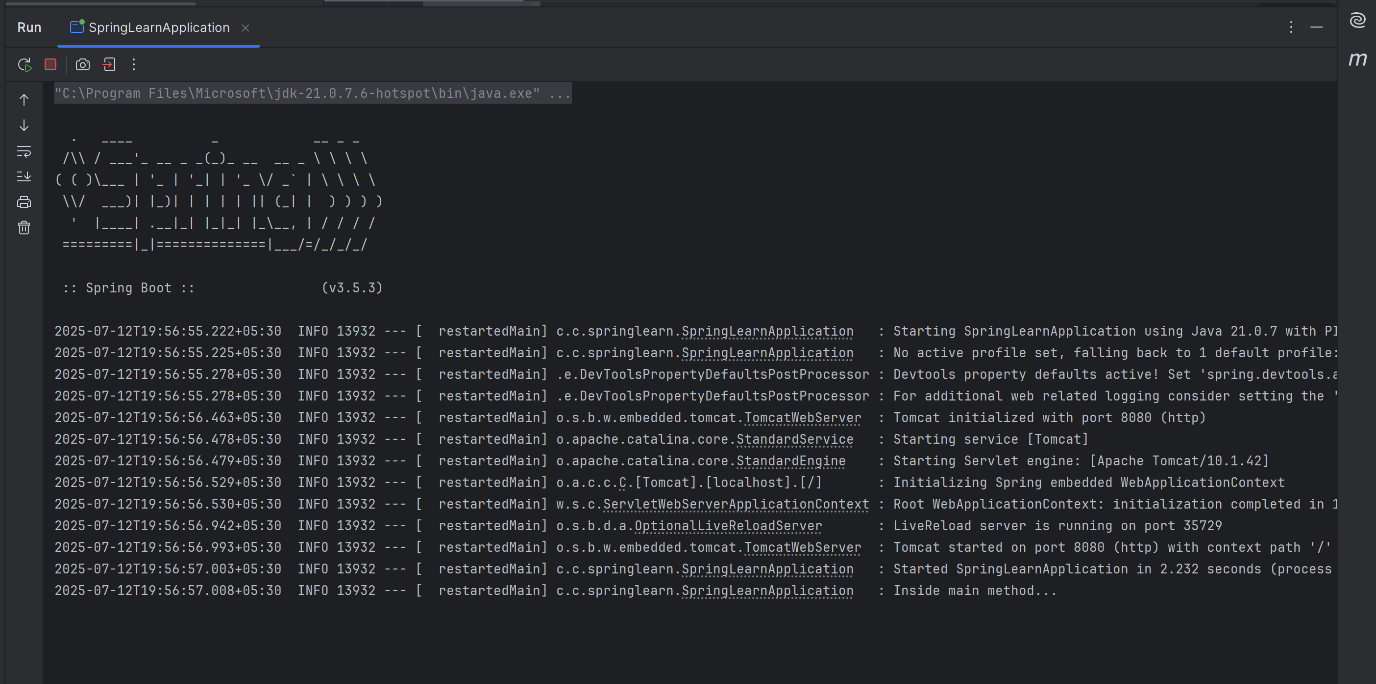
****

**Pom.xml:**

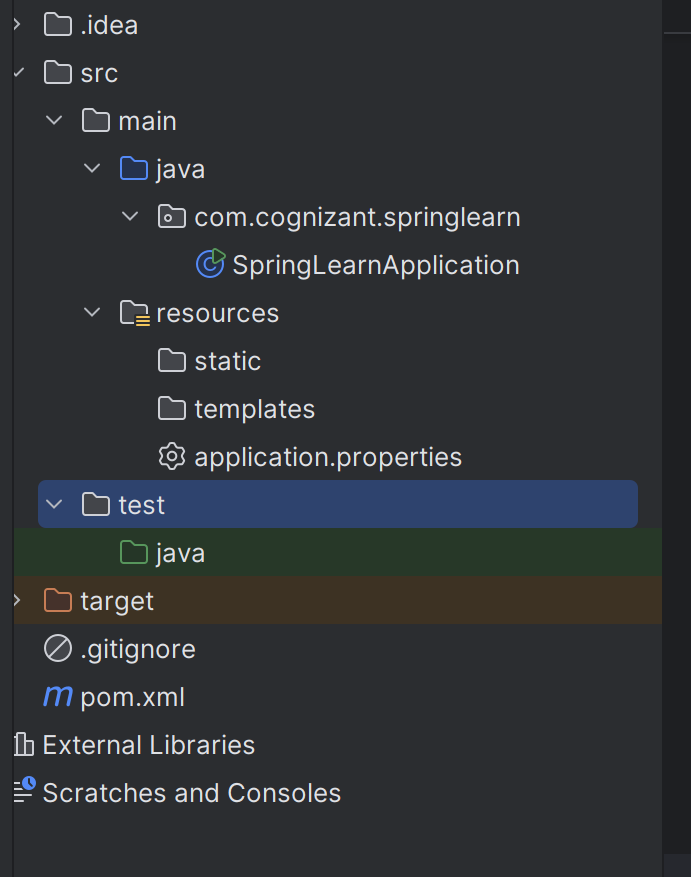
****



**SpringApplication.java:**  


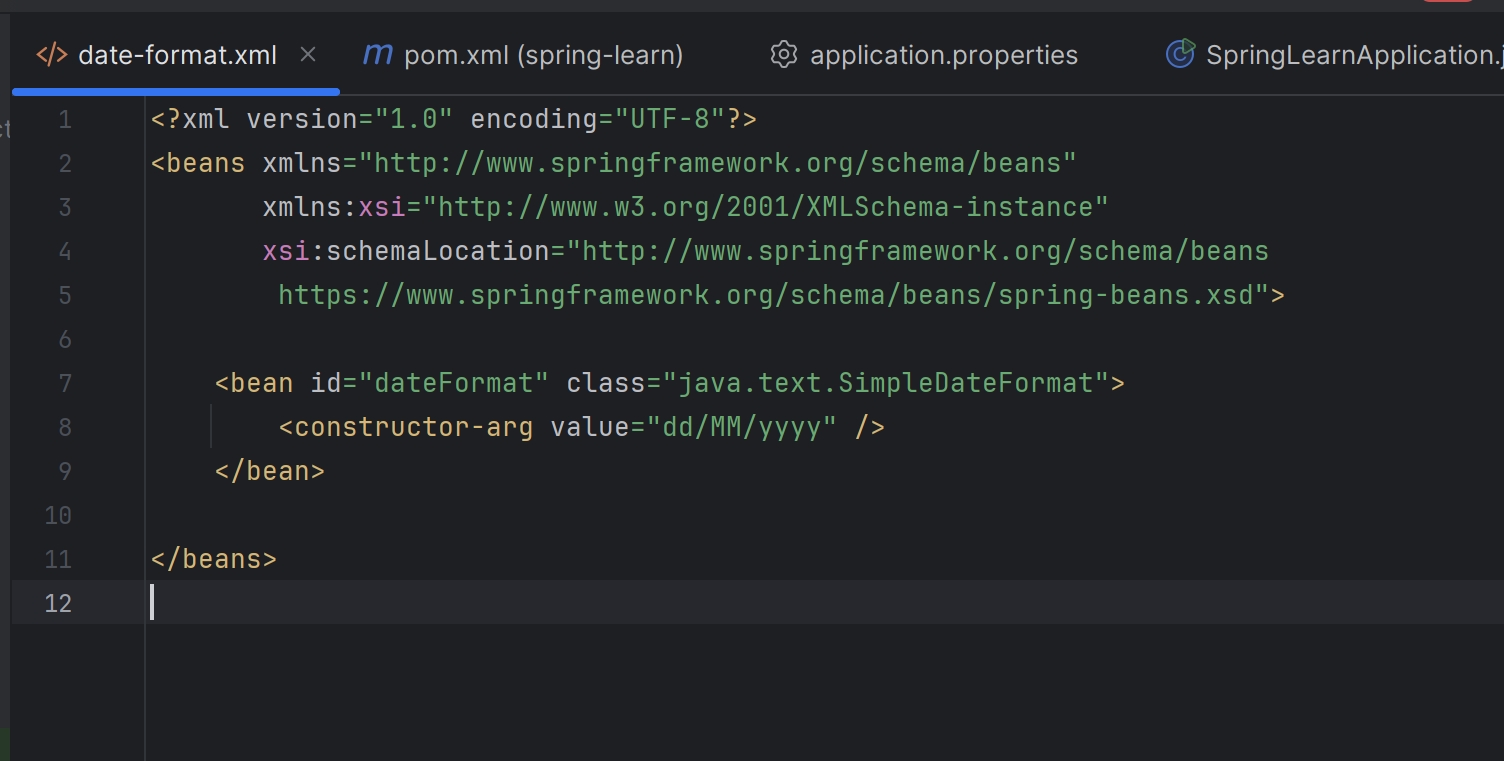
**OUTPUT :  
  
**

**Project Structure:**



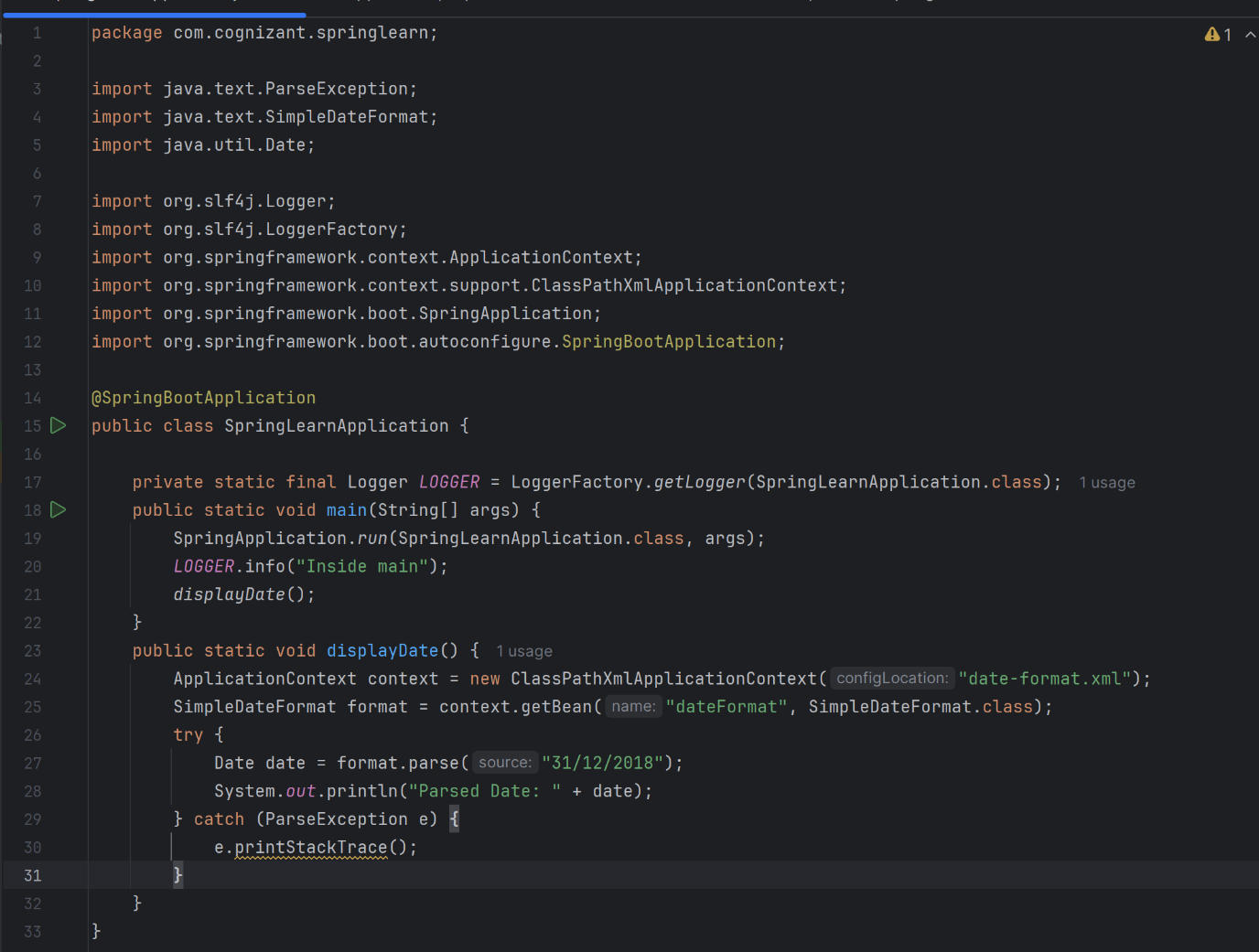
1. src/main/java: Contains all Java application source code.
2. src/main/resources: Contains configuration files like application.properties.
3. src/test/java: Reserved for unit and integration testing.
4. SpringLearnApplication.java:
   * This is the main entry point.
   * Annotated with @SpringBootApplication, it initializes the Spring context.
   * Log message added to confirm execution.
5. @SpringBootApplication Annotation:
   * Combines @Configuration, @EnableAutoConfiguration, and @ComponentScan
   * Enables Spring Boot’s auto-configuration and component scanning.
6. pom.xml Walkthrough:
   * Contains all Maven configurations and dependencies.
   * Defined spring-boot-starter-web and spring-boot-devtools
   * Shows project metadata (groupId, artifactId, version)
7. Dependency Hierarchy:
   * Opened in IntelliJ to view all transitive dependencies.
   * Noted that spring-boot-starter-web brings in:
     + Tomcat (embedded server)
     + Jackson (for JSON)
     + Spring MVC modules

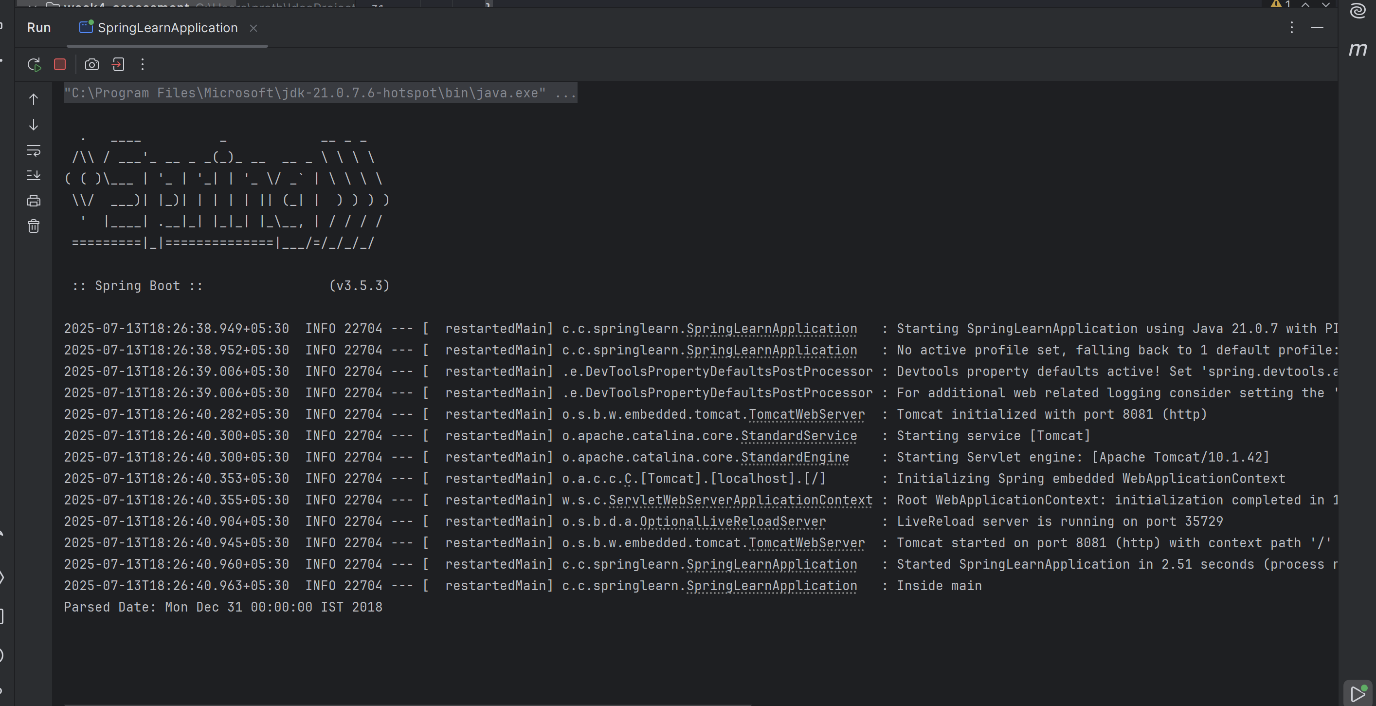
**2.Spring Core – Load Country from Spring Configuration XML**

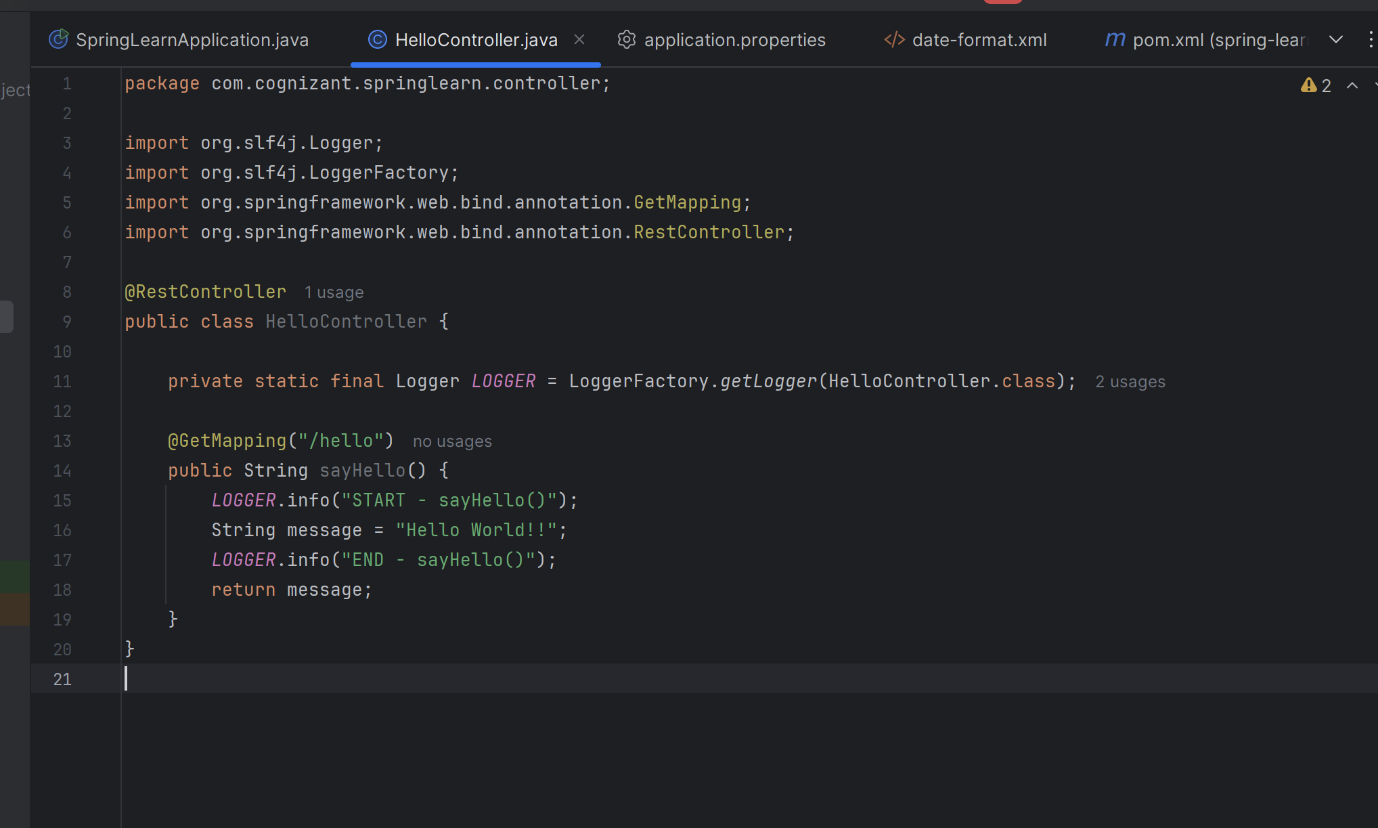
**Date-format.xml:  
**

**application.properties:  
**

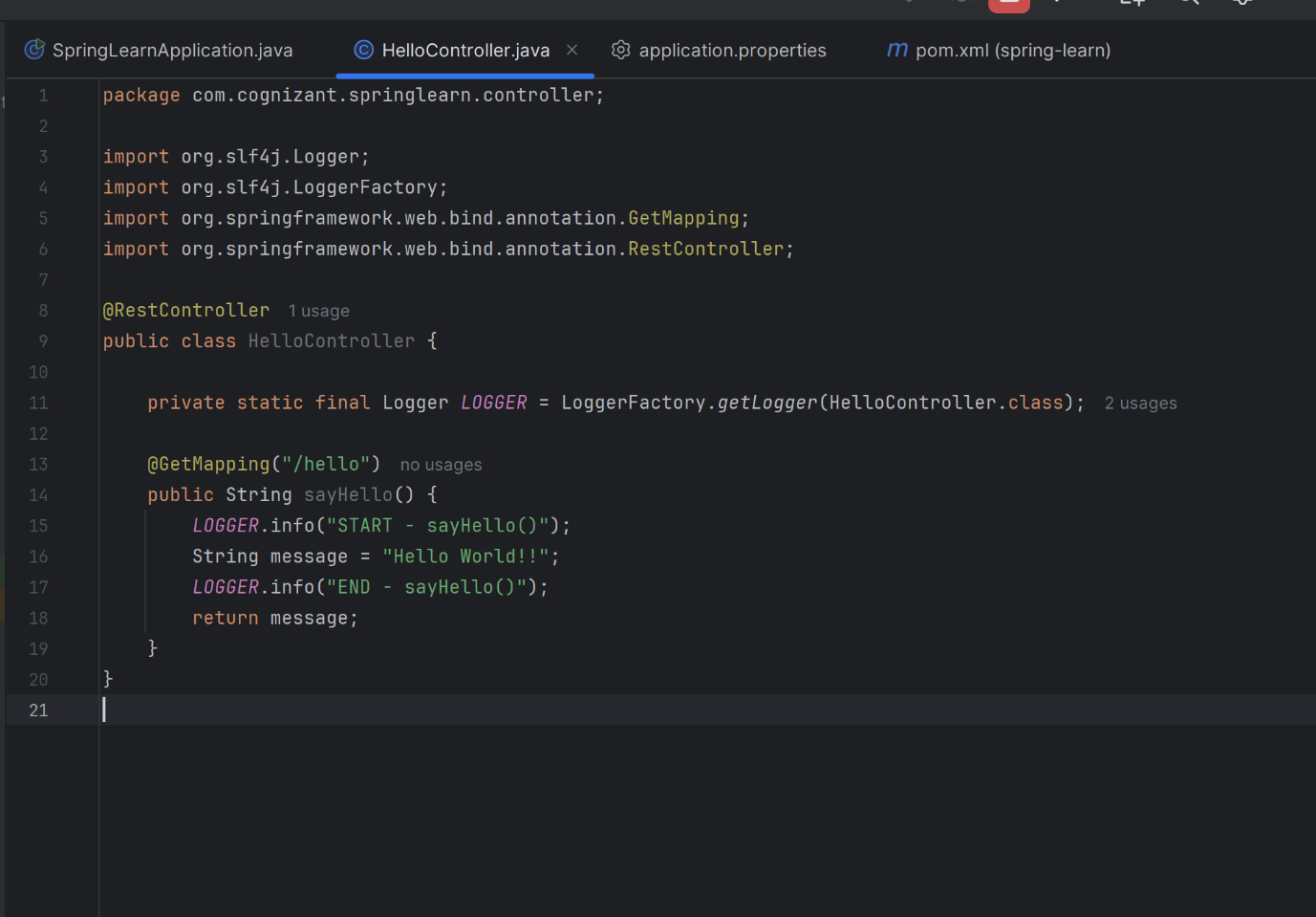
**SpringApplication.java :**

****

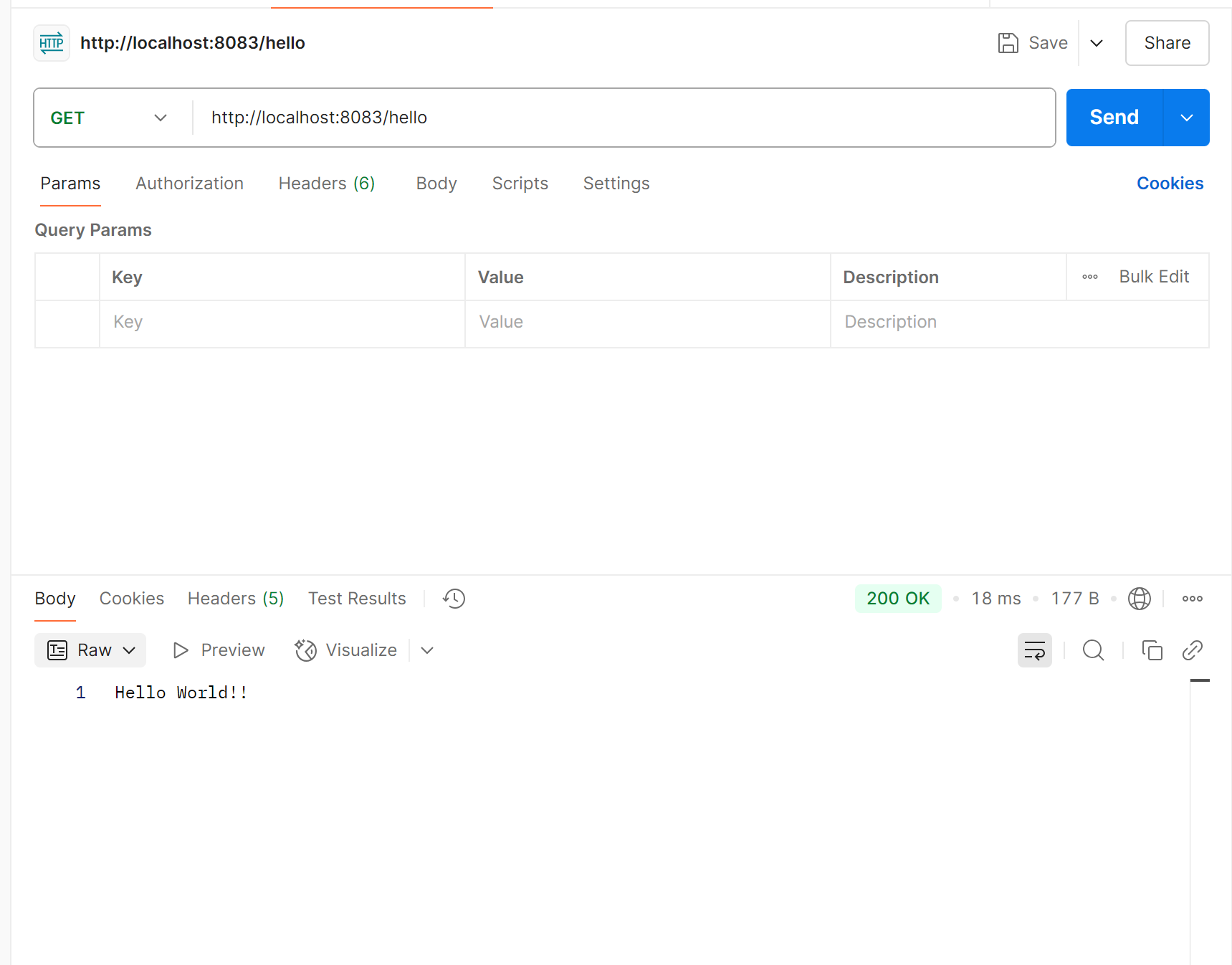
**Output:  
**

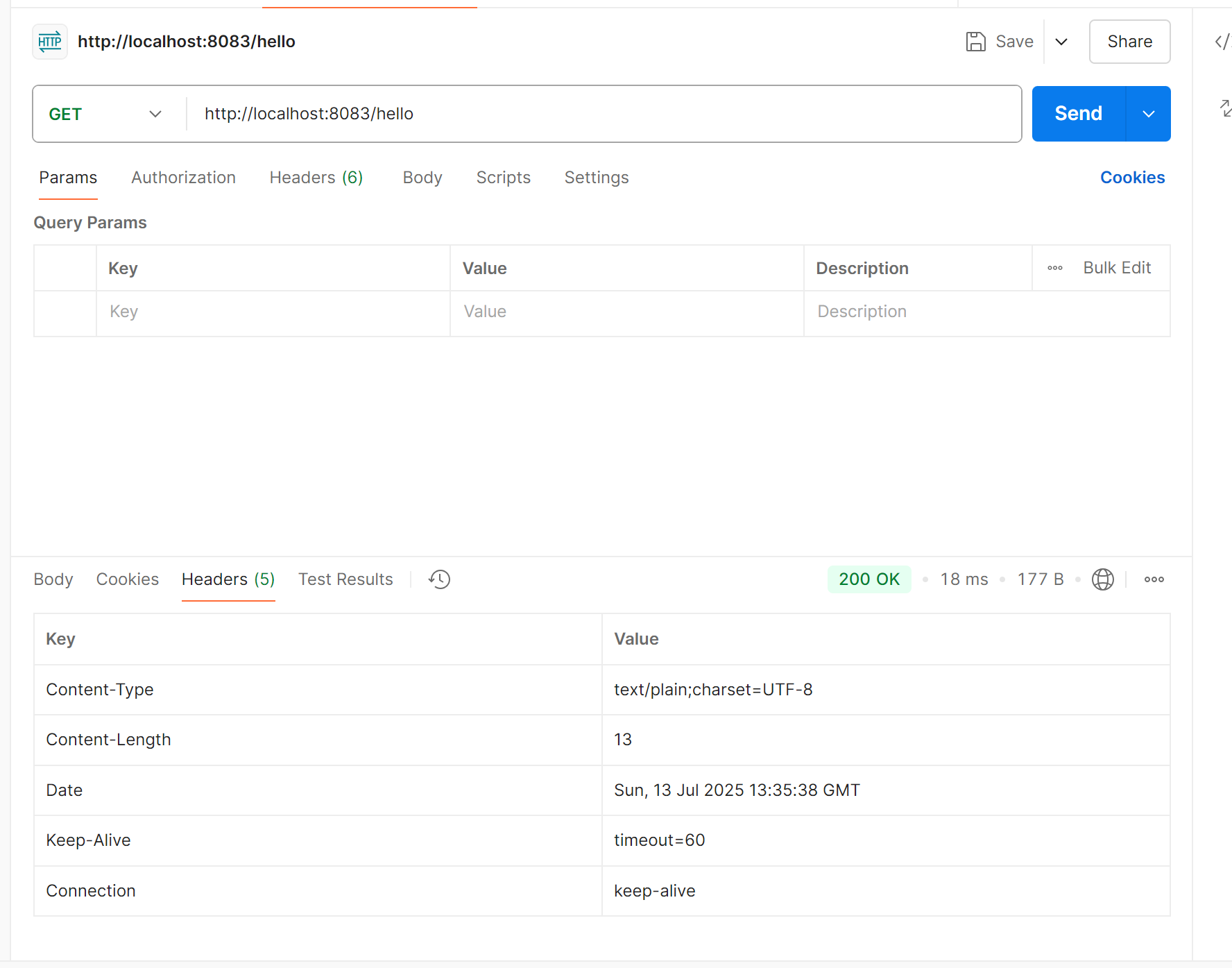
**3.Hello World RESTful Web Service :  
  
HelloController:  
  
**

**Application.properties:  
**

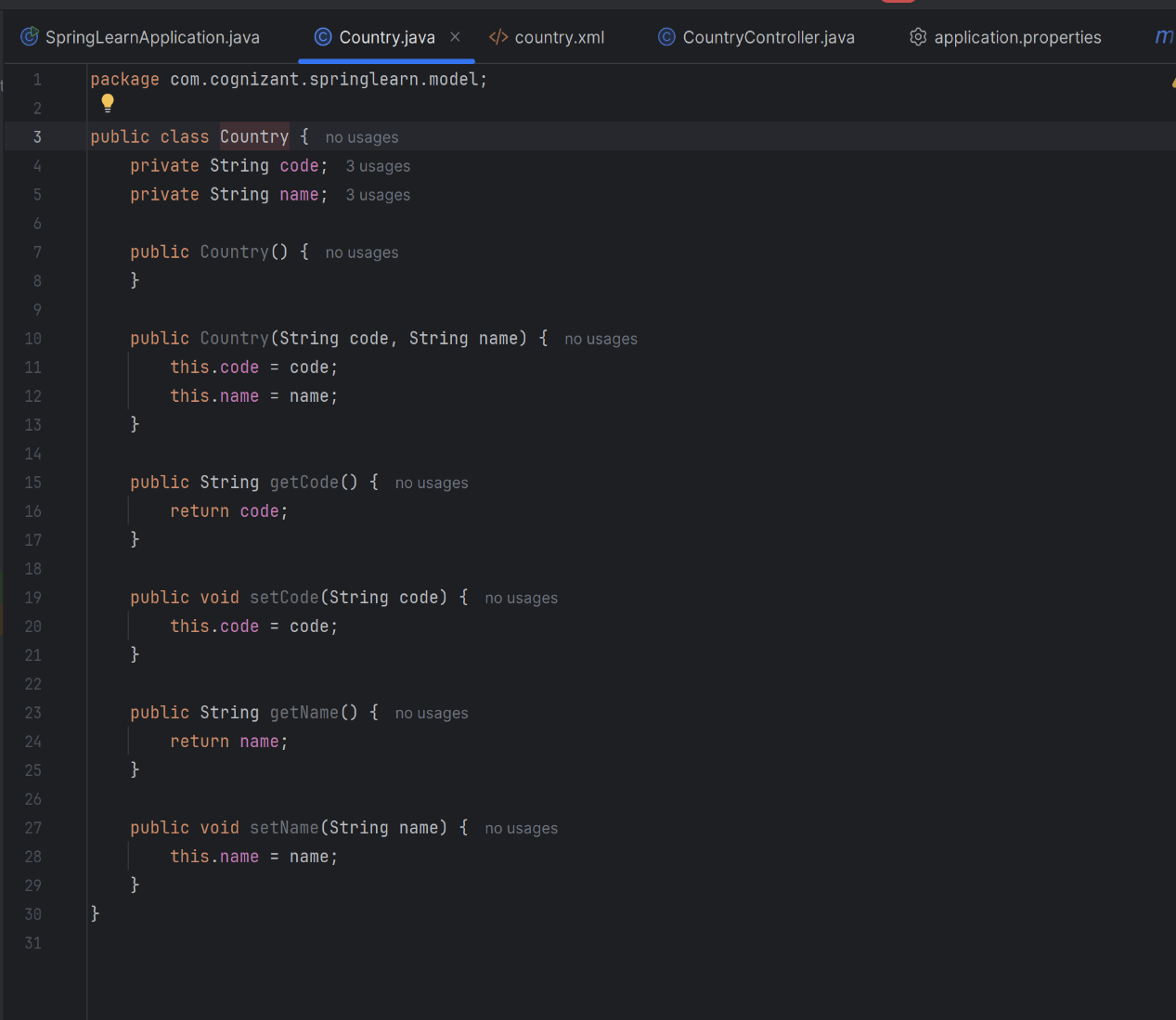
**SpringLearnApplication.java :  
**

**Output:  
**

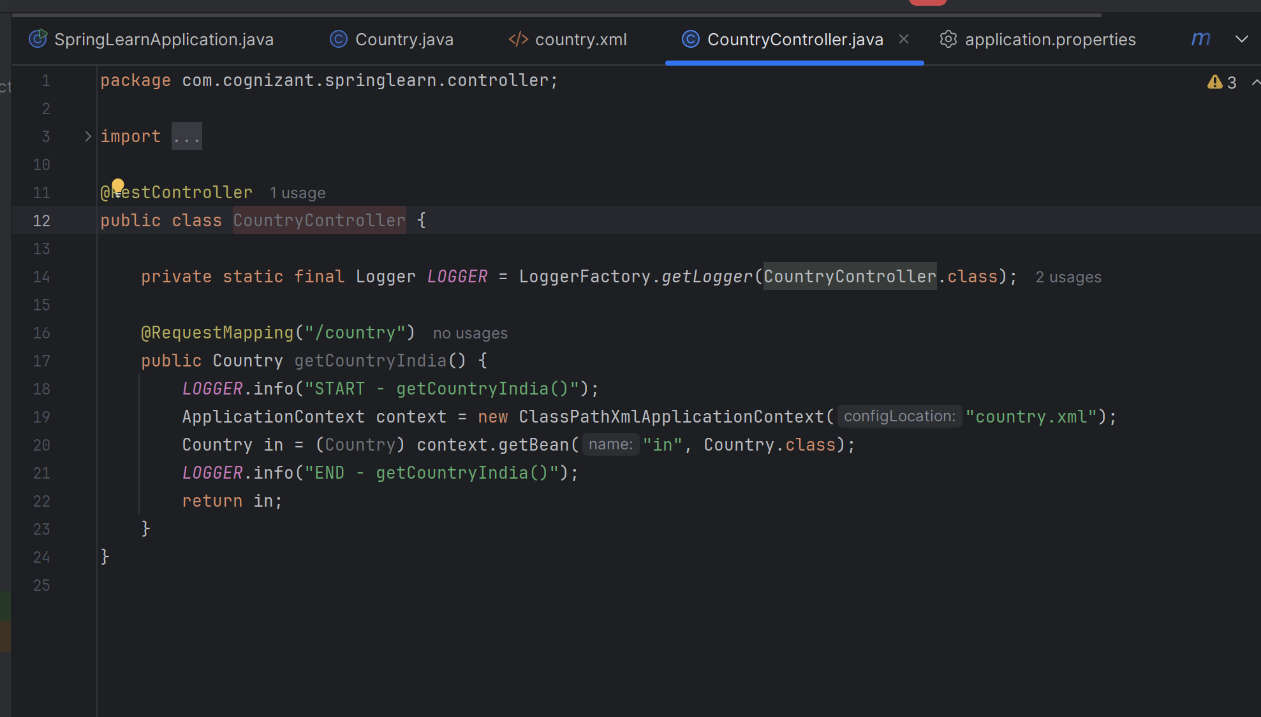
**Checking using postman:  
  
**

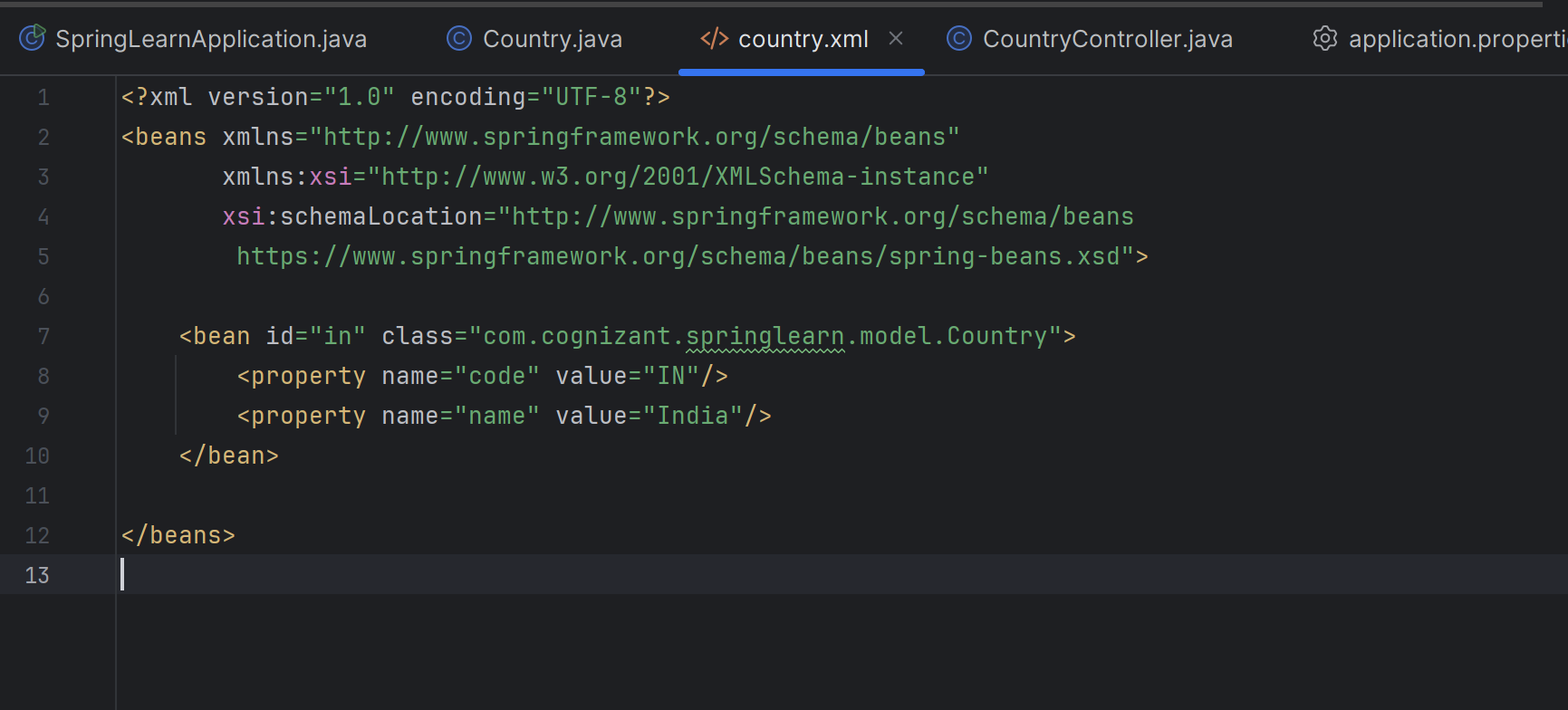
**Header:  
  
**

**4.REST-Country Web Service:**

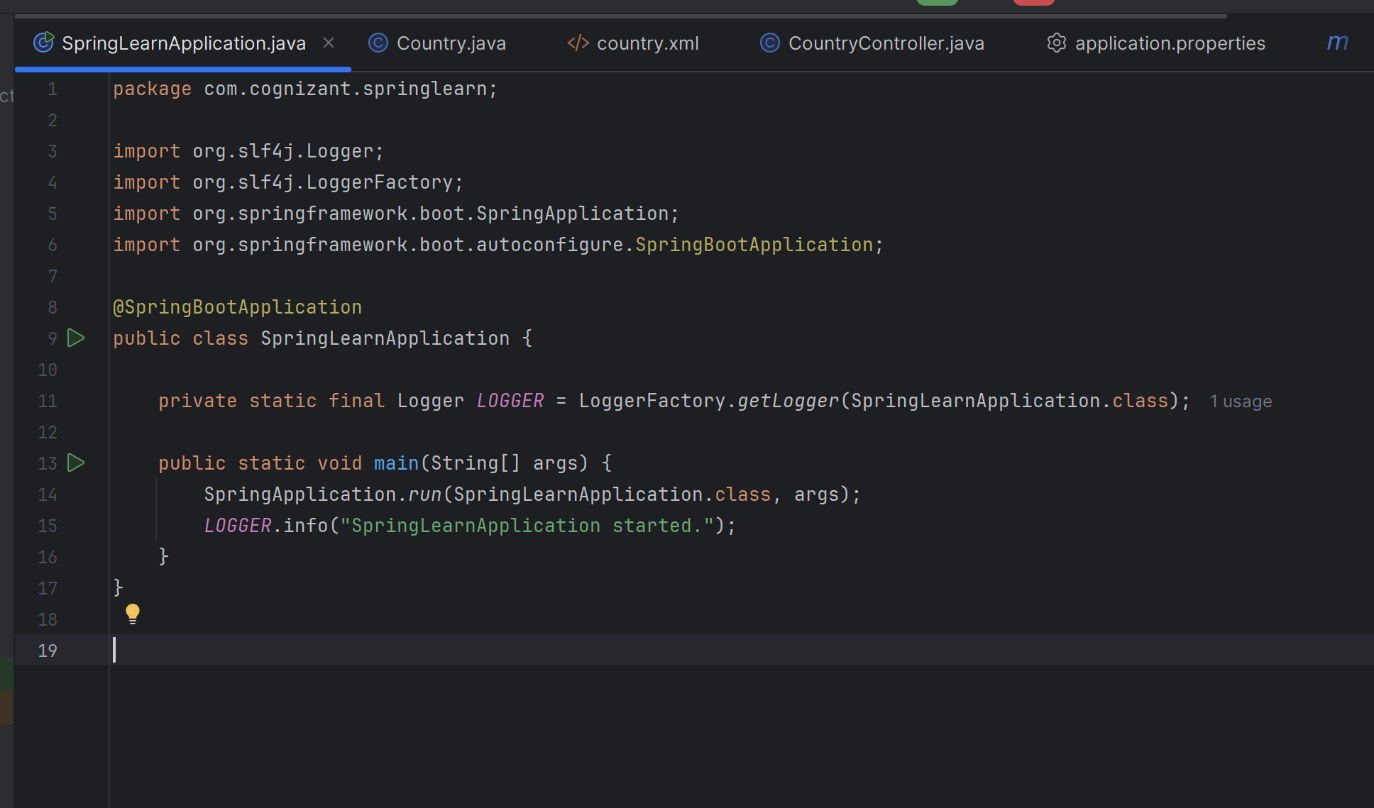
**Model/country :  
**

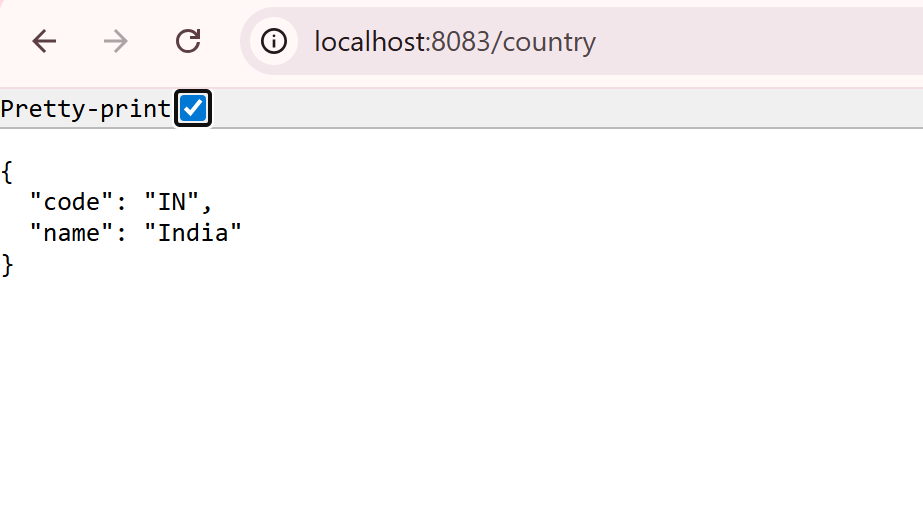
**CountryController.java :**

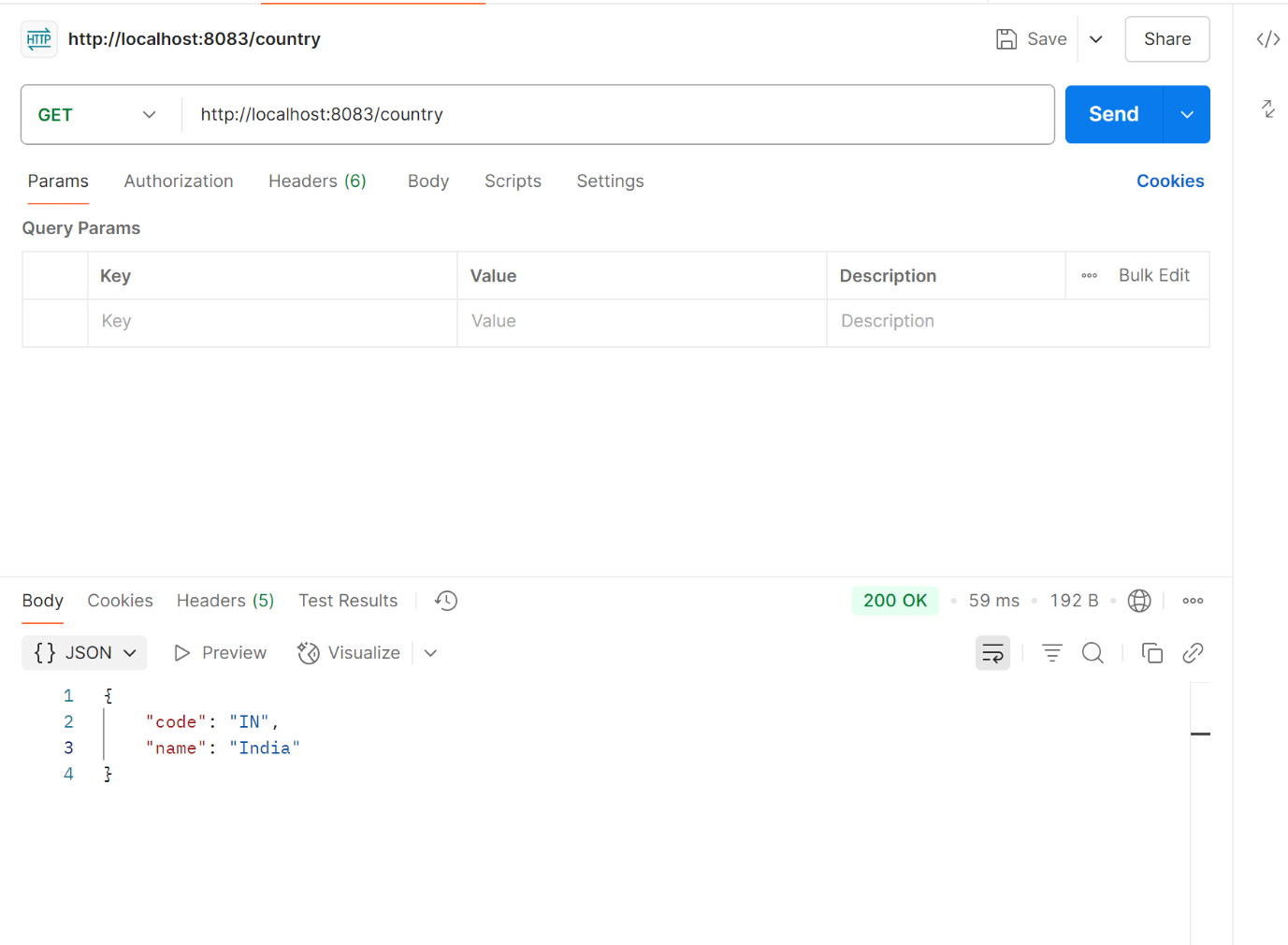
****

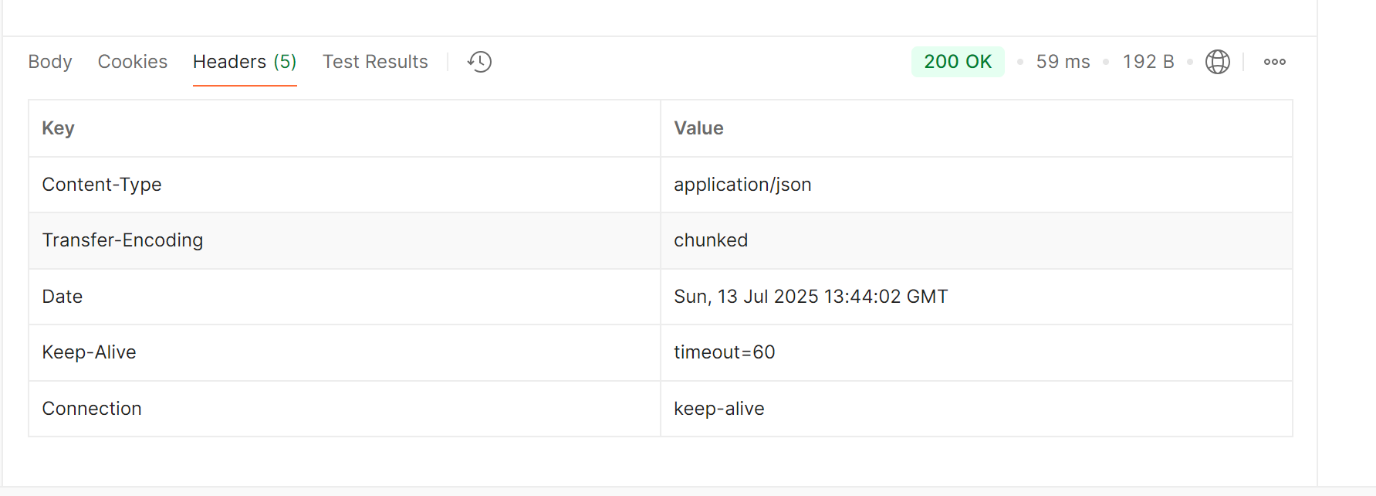
**Country.xml:  
**

**SpringLearnApplication.java:**

****

**Output:  
**

**Postman output:  
**

**Header:  
**Controller Method :

* @RequestMapping("/country") handles GET requests to /country
* getCountryIndia() loads the Spring XML context
* Gets the in bean of type Country
* Returns it to Spring, which serializes it to JSON using Jackson

Bean Converted to JSON :

Spring Boot includes Jackson (via spring-boot-starter-web) which:

* Converts any object returned from a controller into JSON
* This is done by the built-in MappingJackson2HttpMessageConverter

So, returning a Country object automatically gives:

{

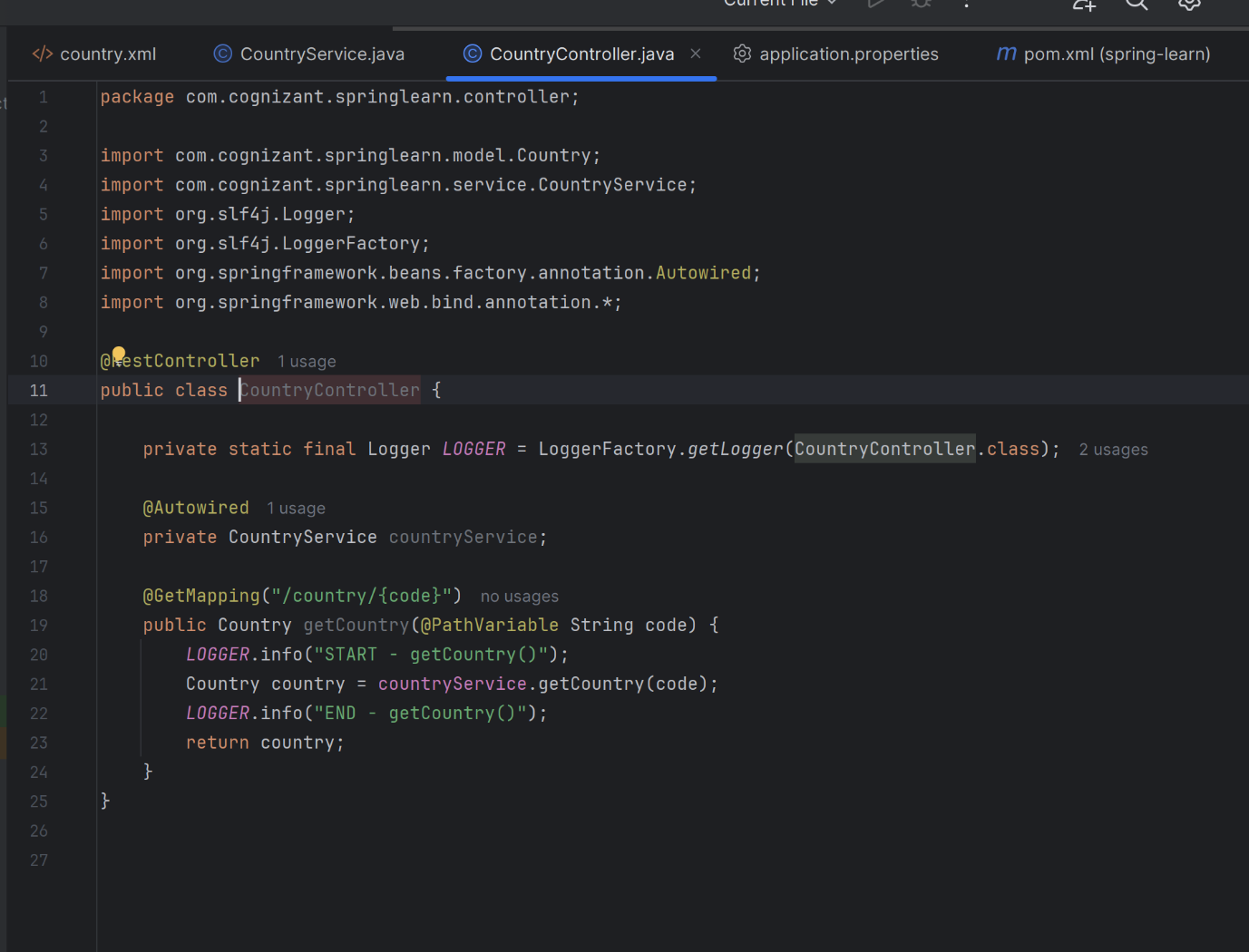
"code": "IN",

"name": "India"

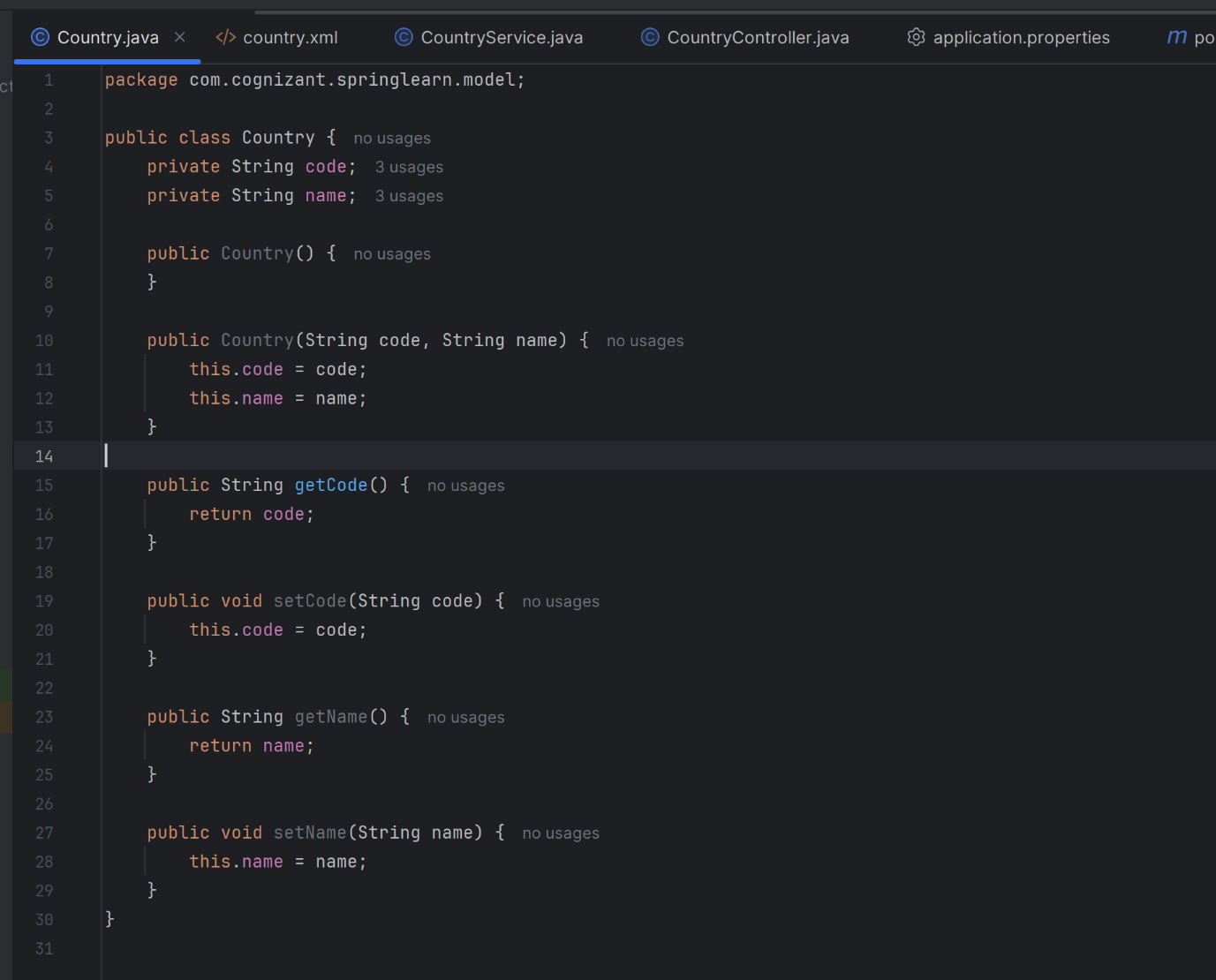
}

**5.REST-Get Country based on country code:**

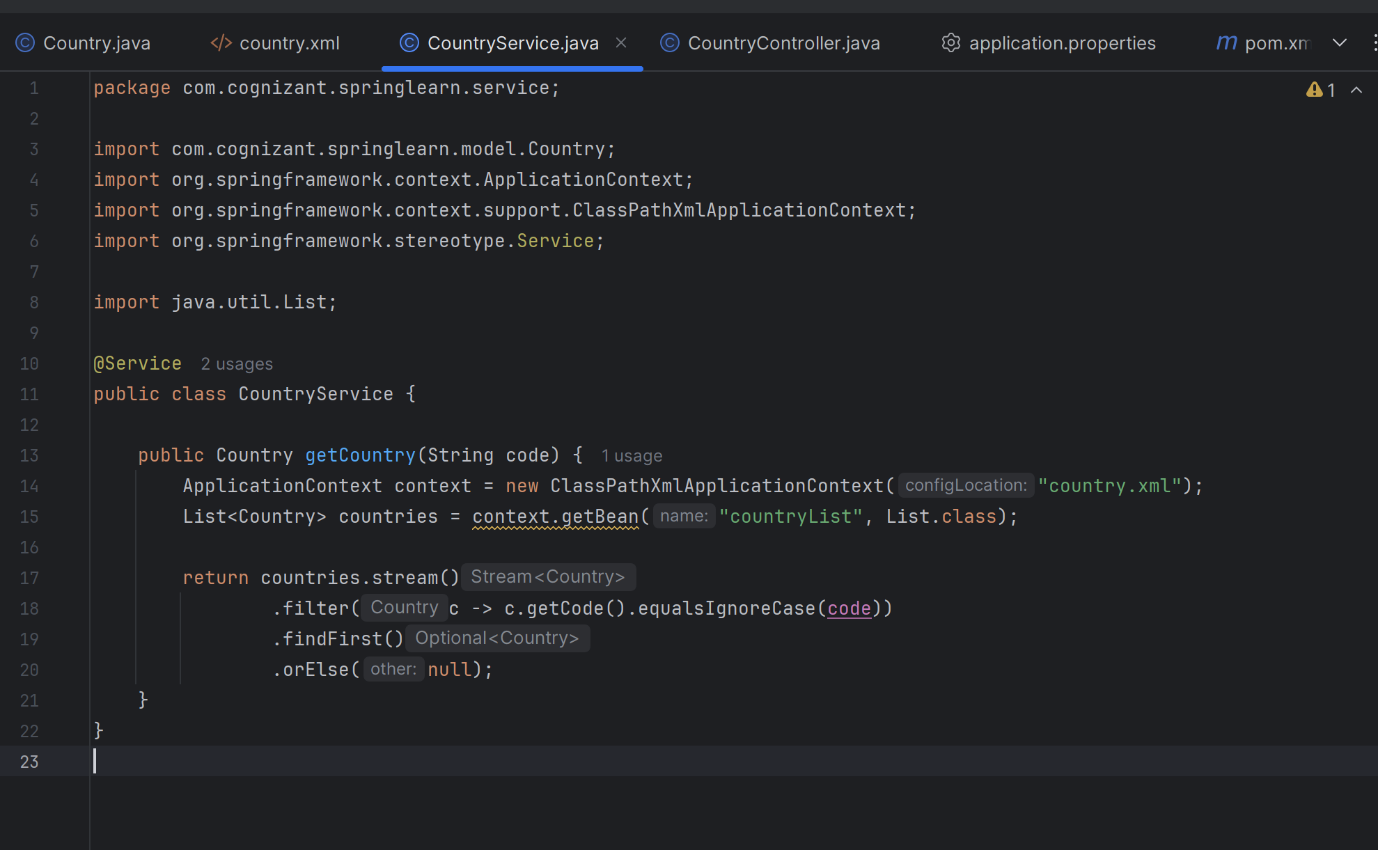
**CountryController :**

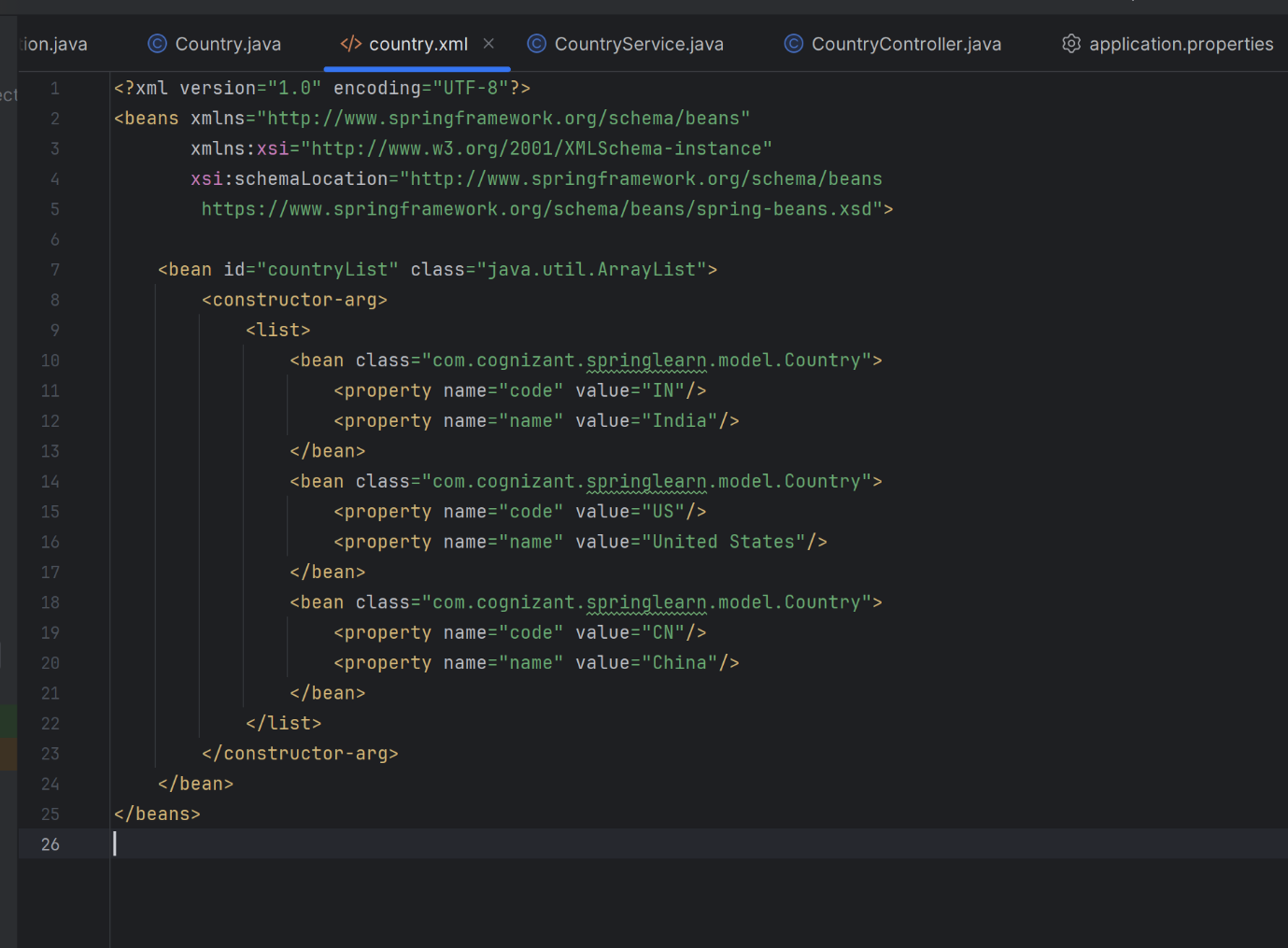
****

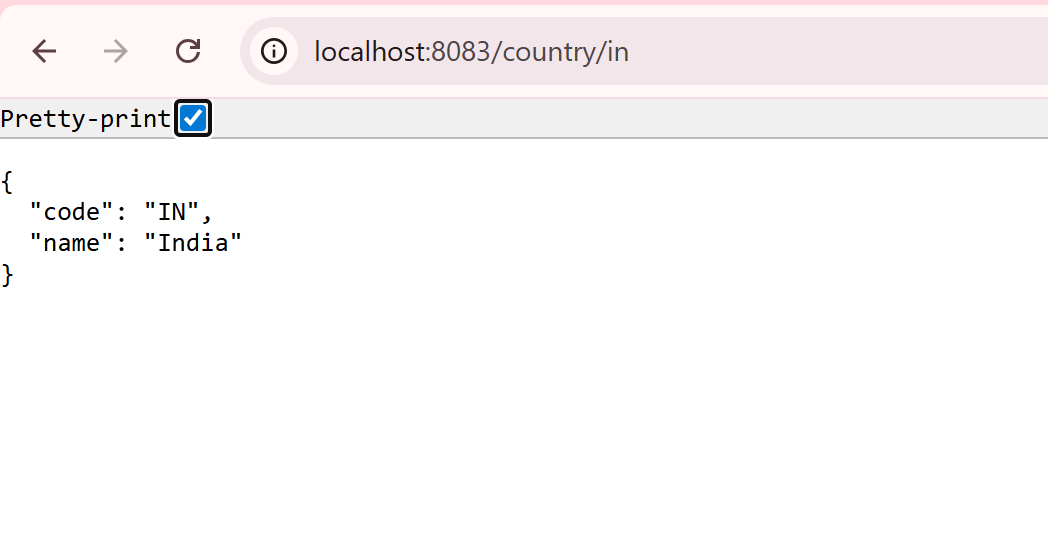
**Model/Country :**

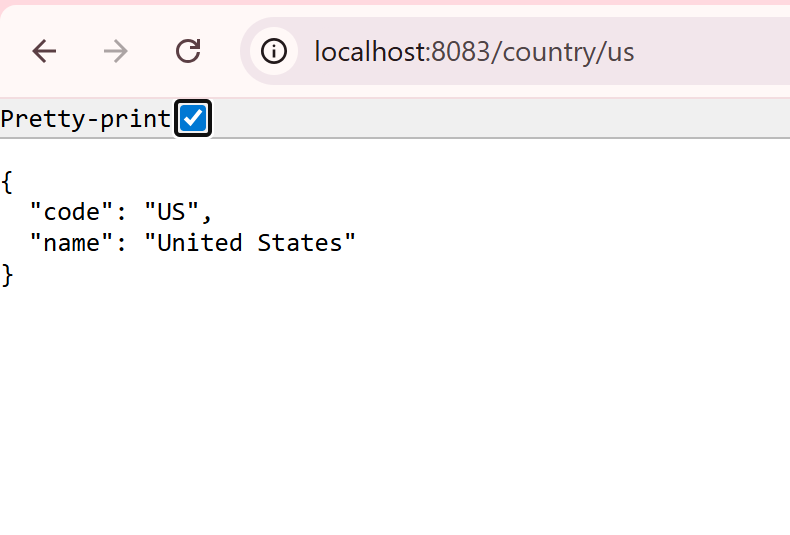
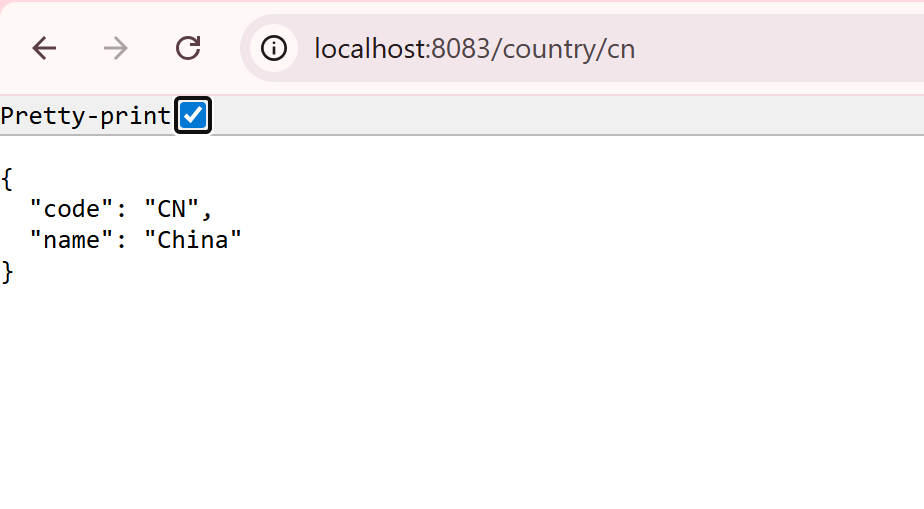
****

**CountryService:**

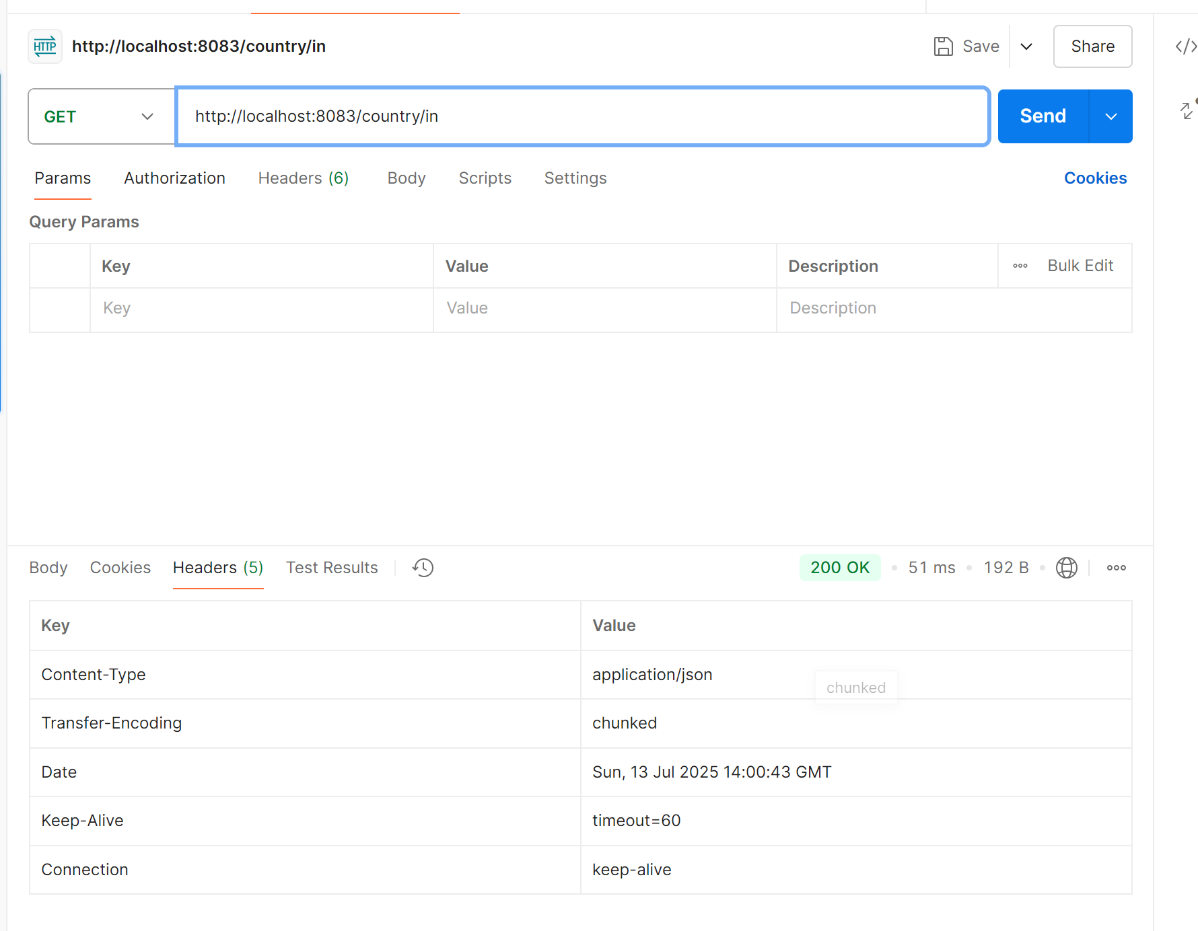
****

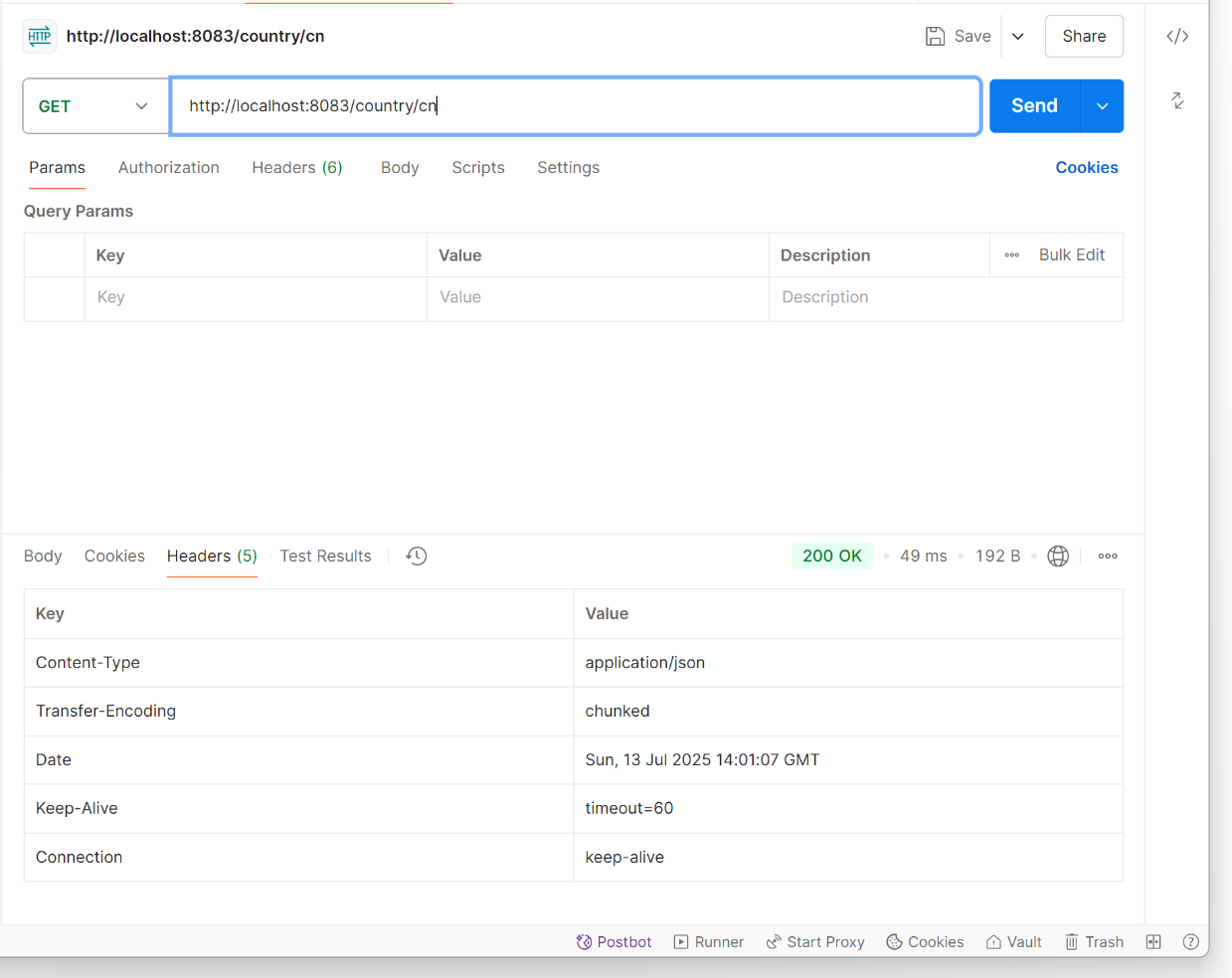
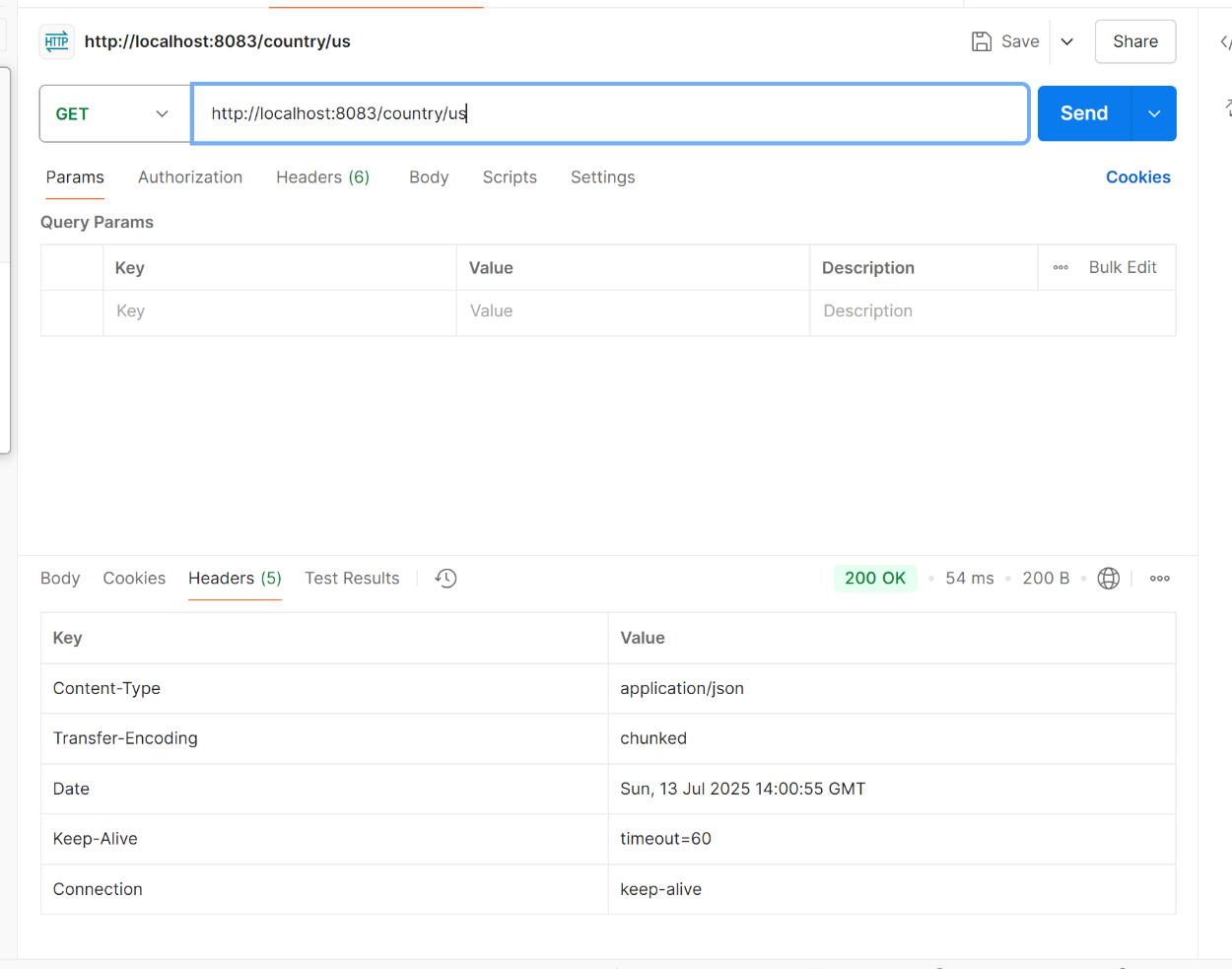
**Country.xml:  
**

**Output:  
  
**

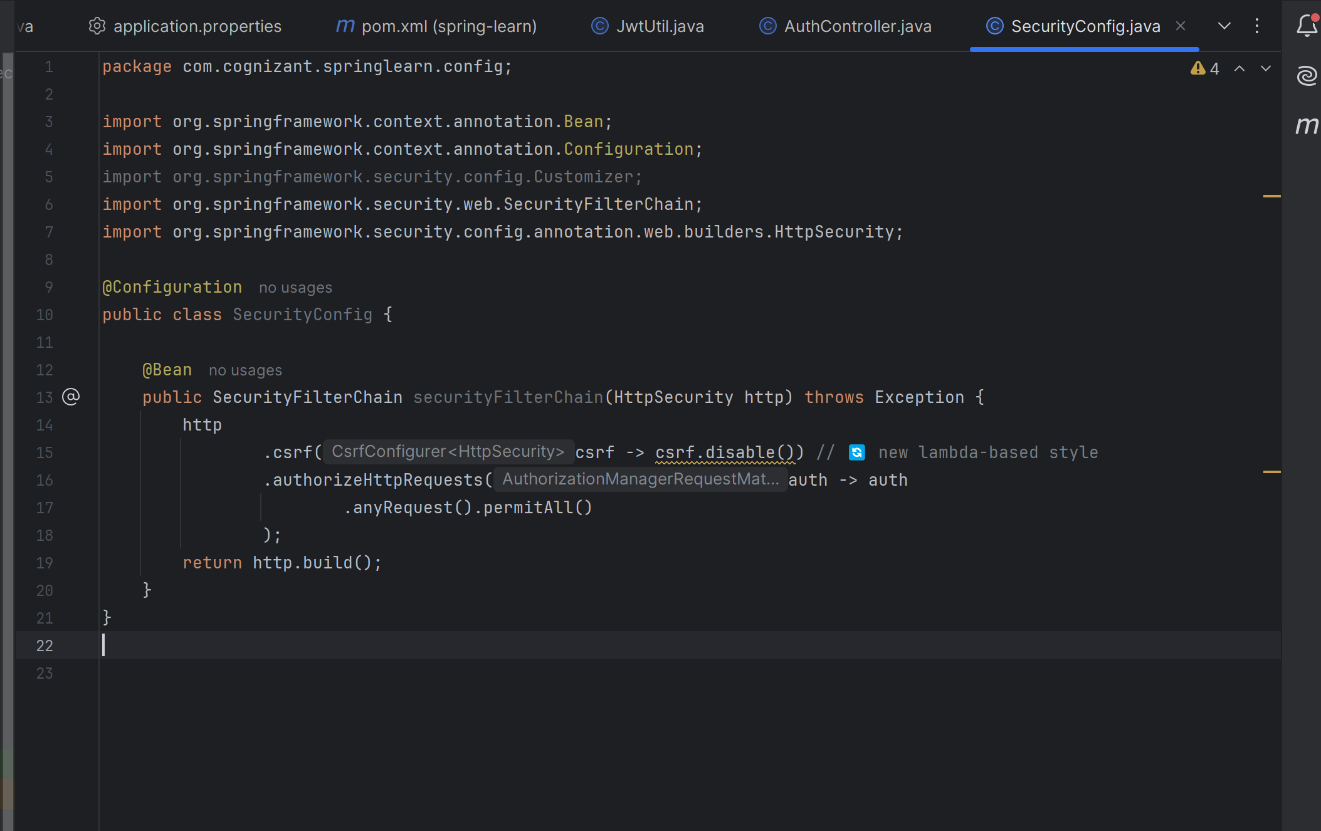
****

**Headers:**

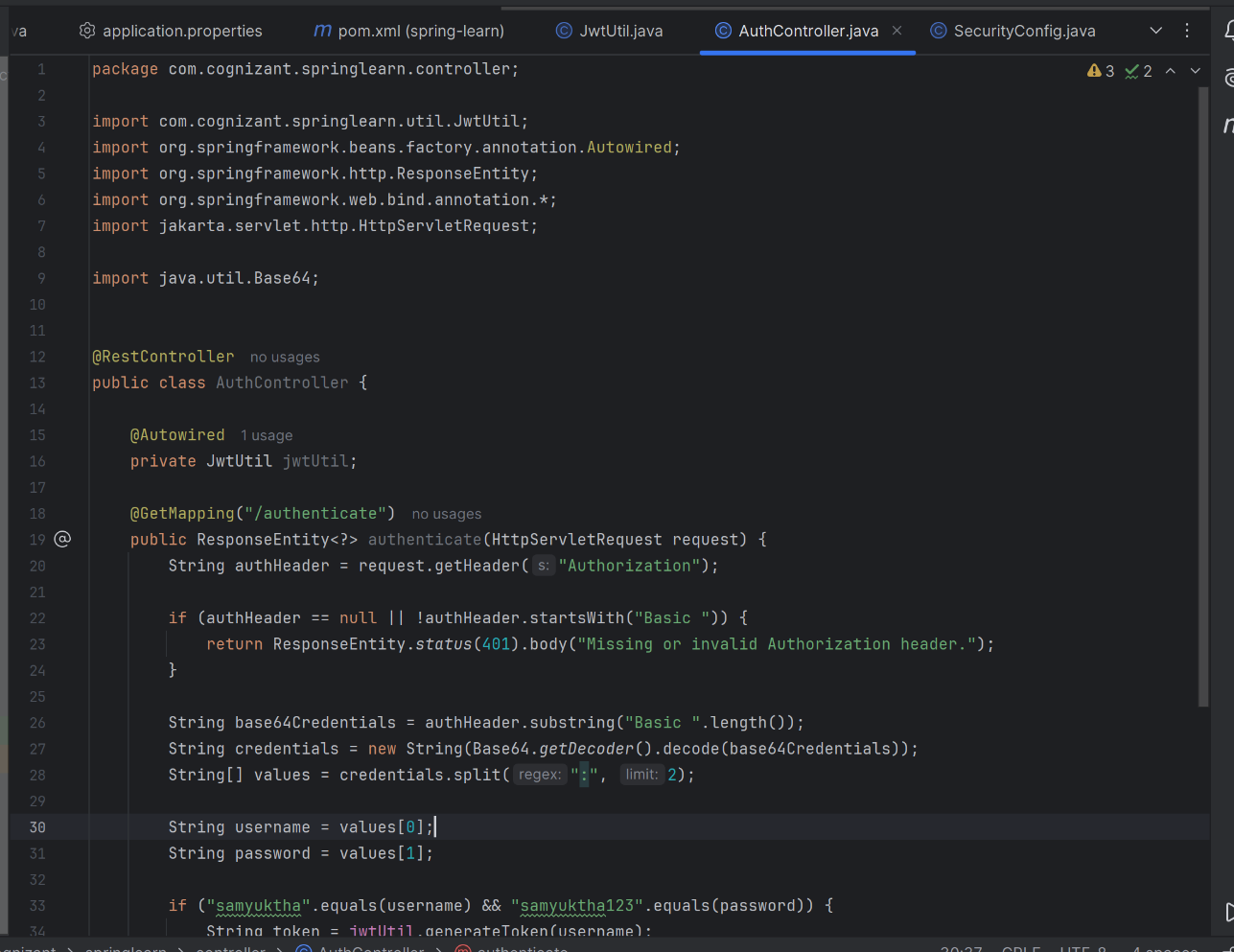
****

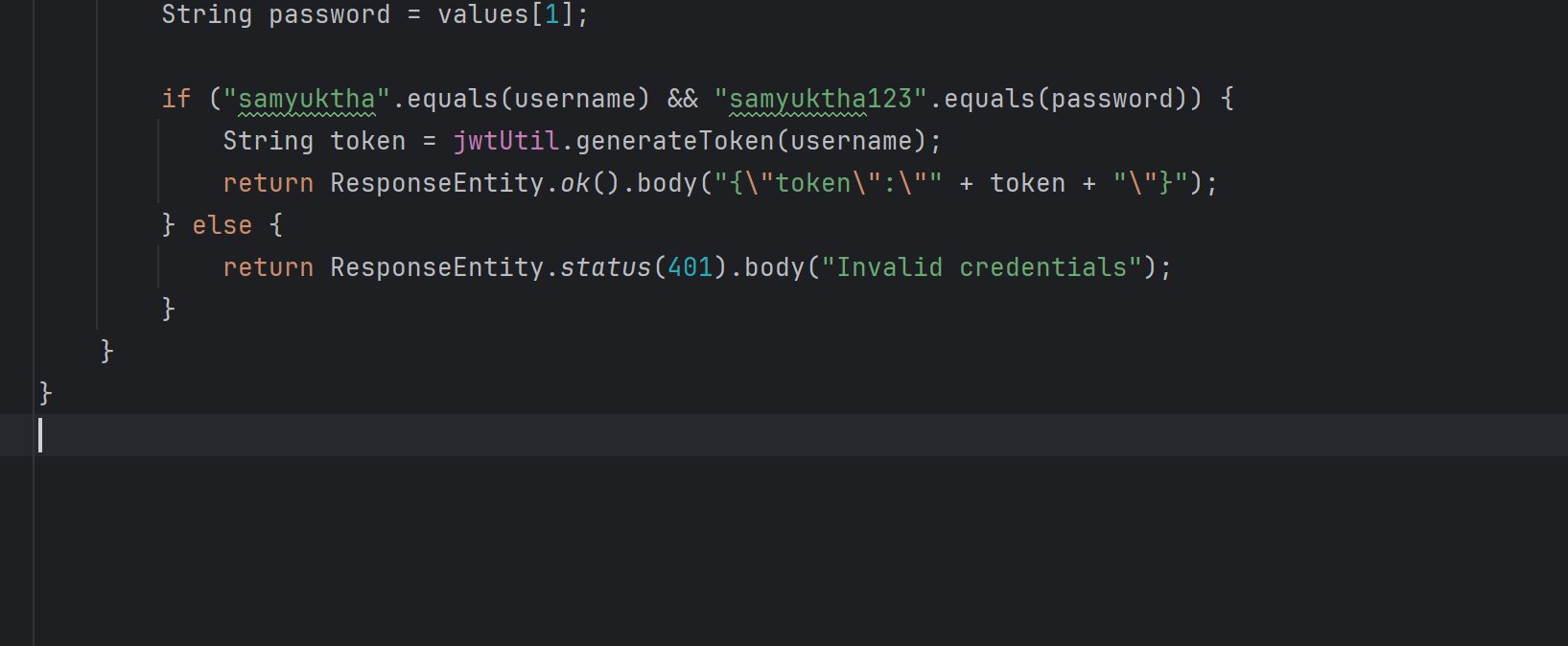
****

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

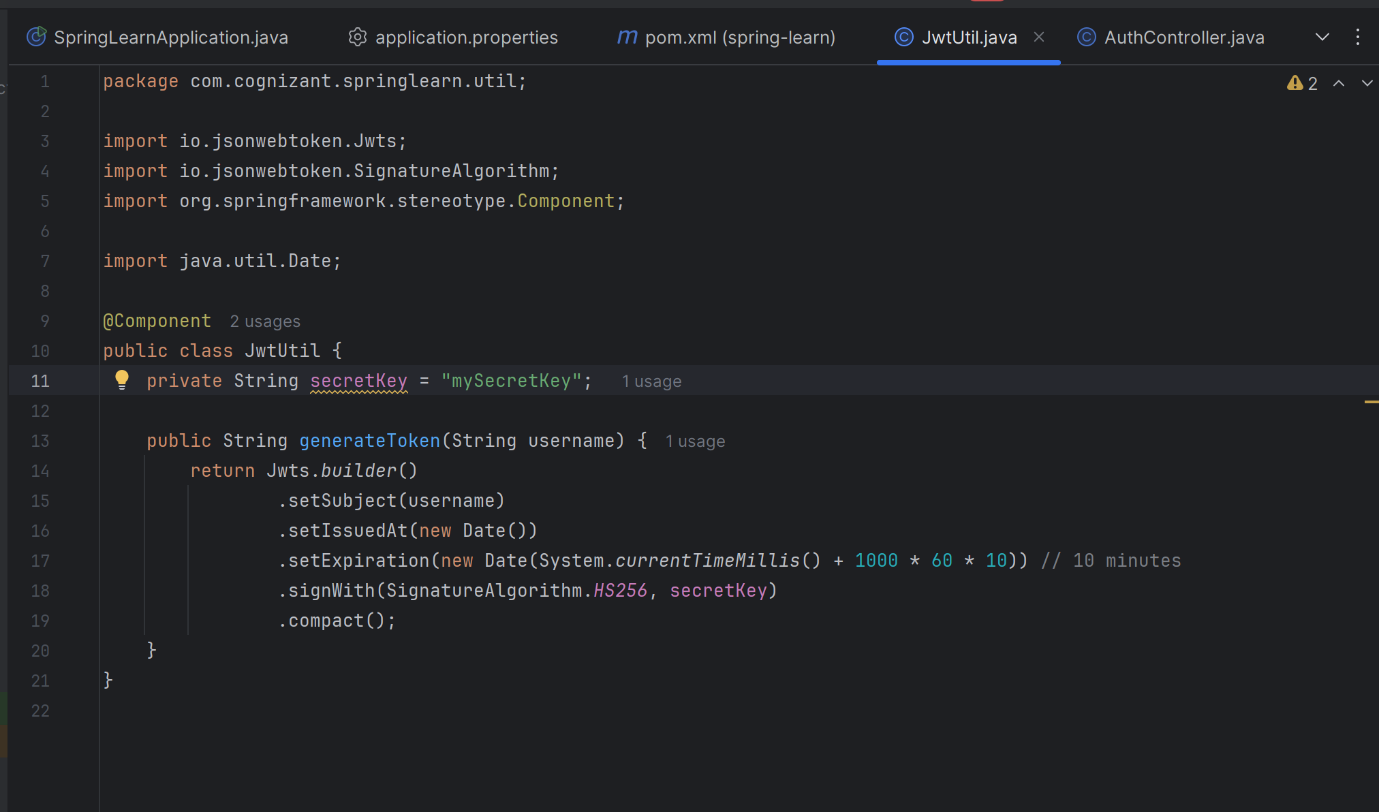
**SecurityConfig:  
**

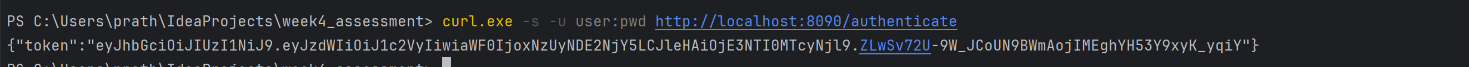
**AuthController.java:**

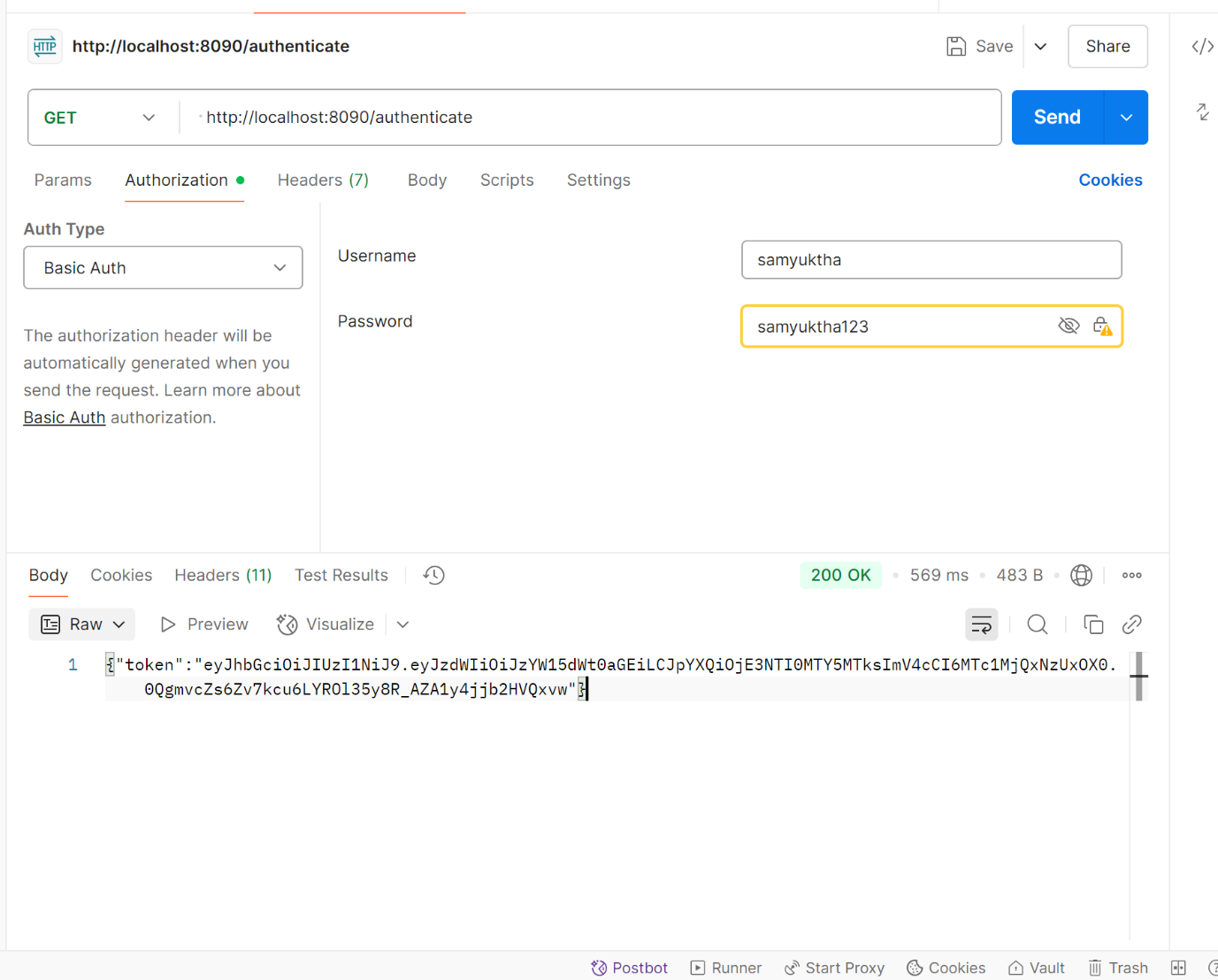
****

****

**JwtUtil.java:**

****

**Output:  
**

**Postman output:  
**