**WEEK 4**

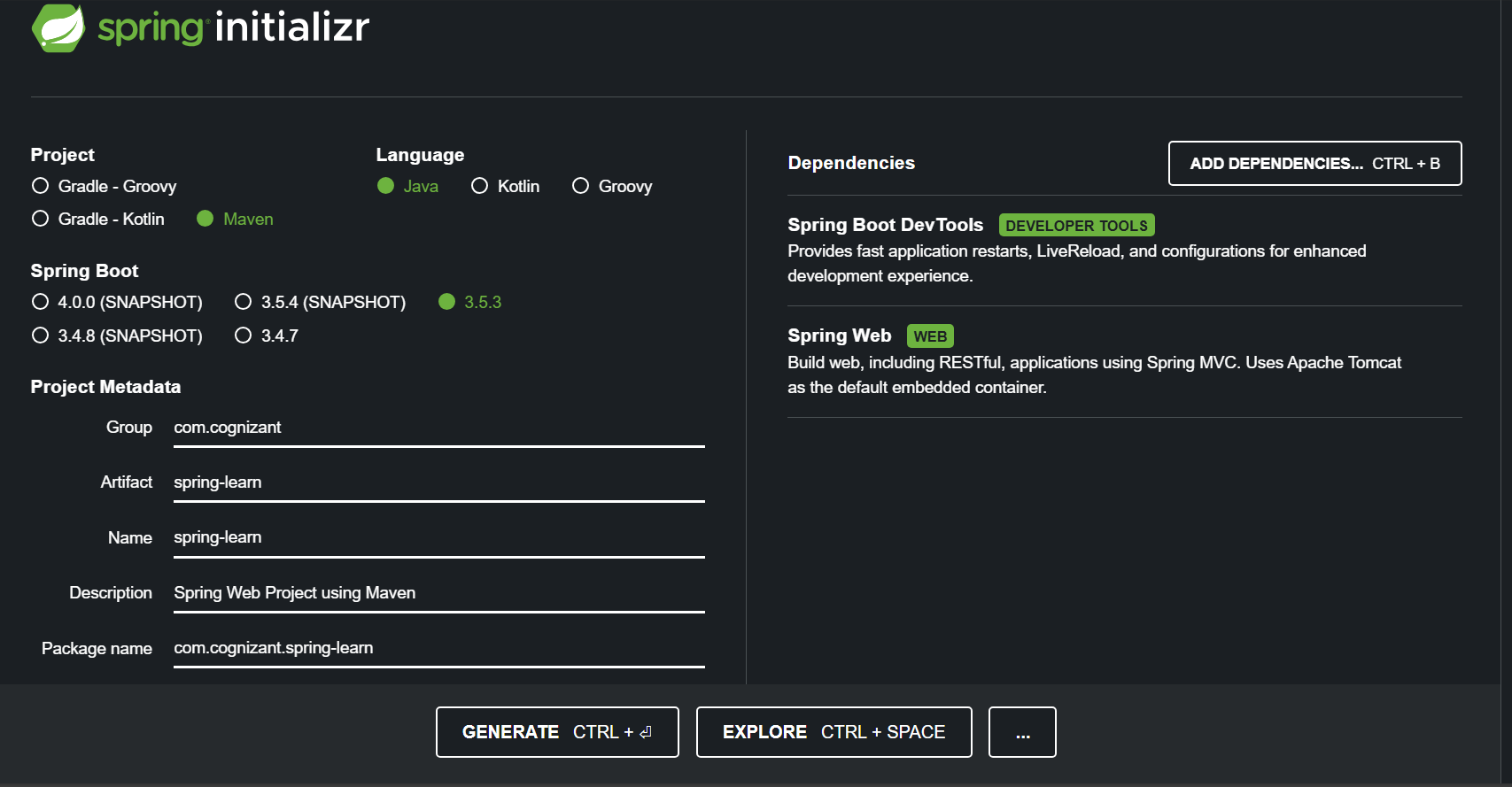
**SPRING REST USING SPRING BOOT**

**1. Spring Rest HandsOn.docx**

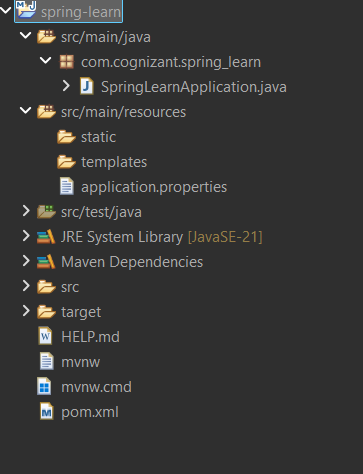
**Hands on 1**

**Create a Spring Web Project using Maven**

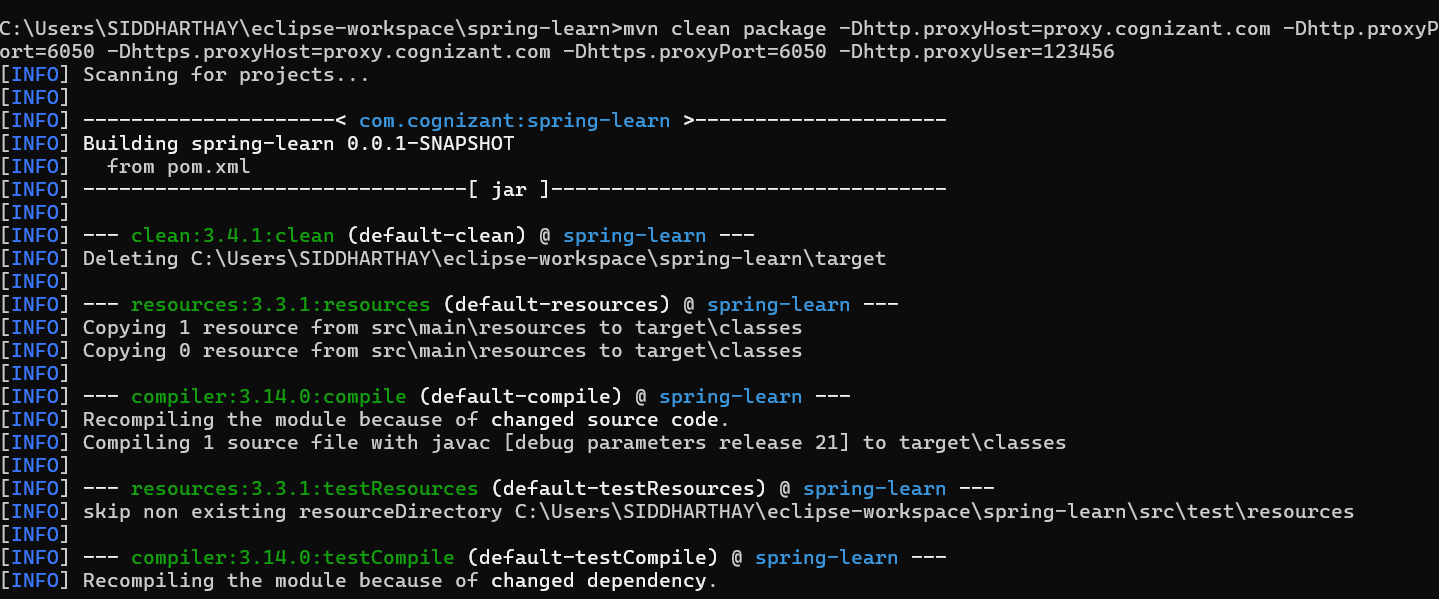
**1. Creating the Spring Web Project using Spring Initializr:**

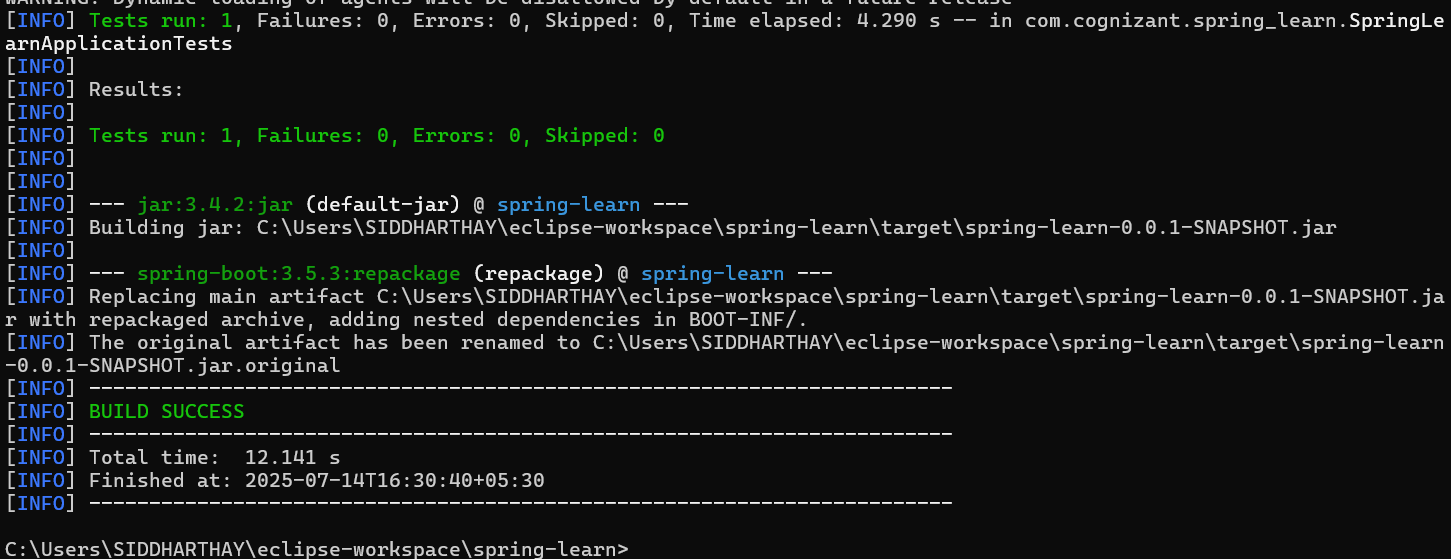


**2. Project Structure after Importing it into Eclipse:**

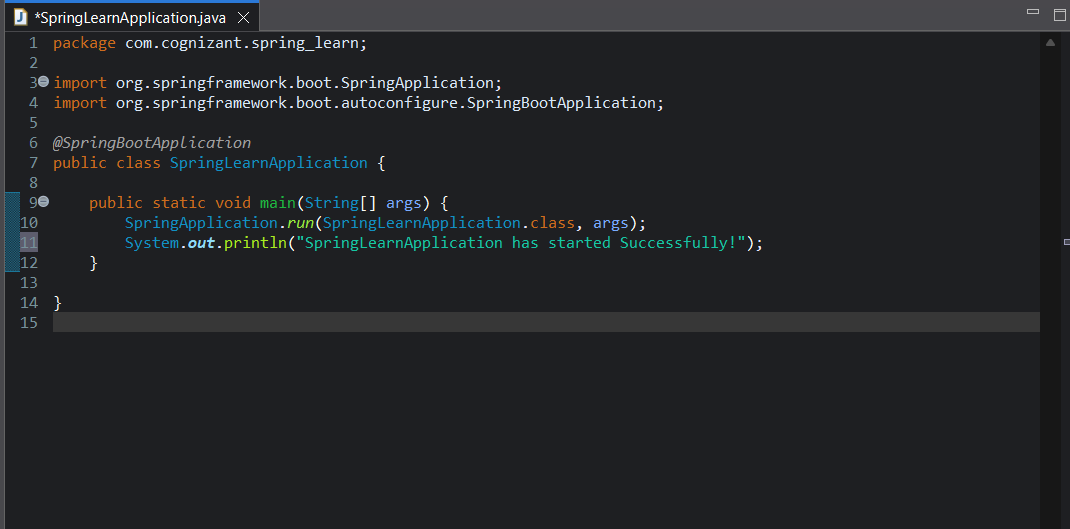


**3. Building the Project using Maven(proxy):**

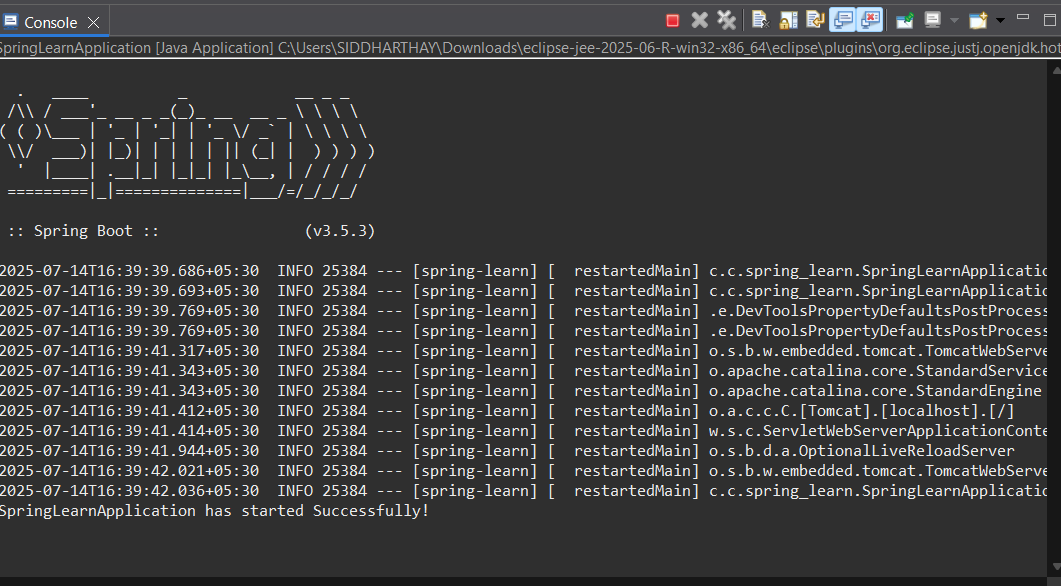




**4. Adding Logging in Main() method:**



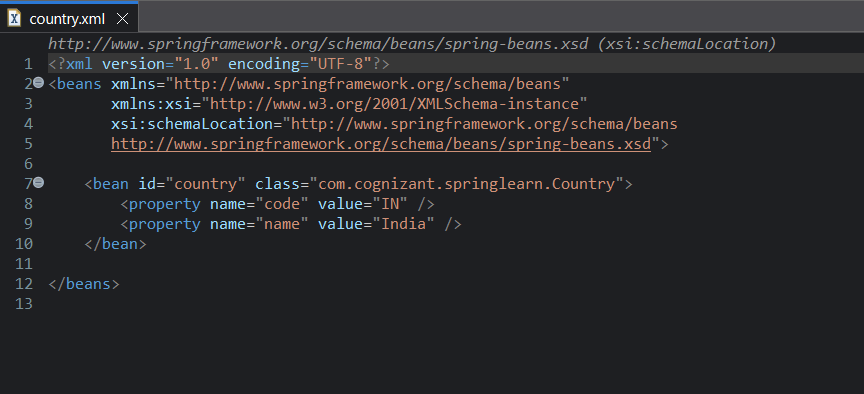
**5. Output:**



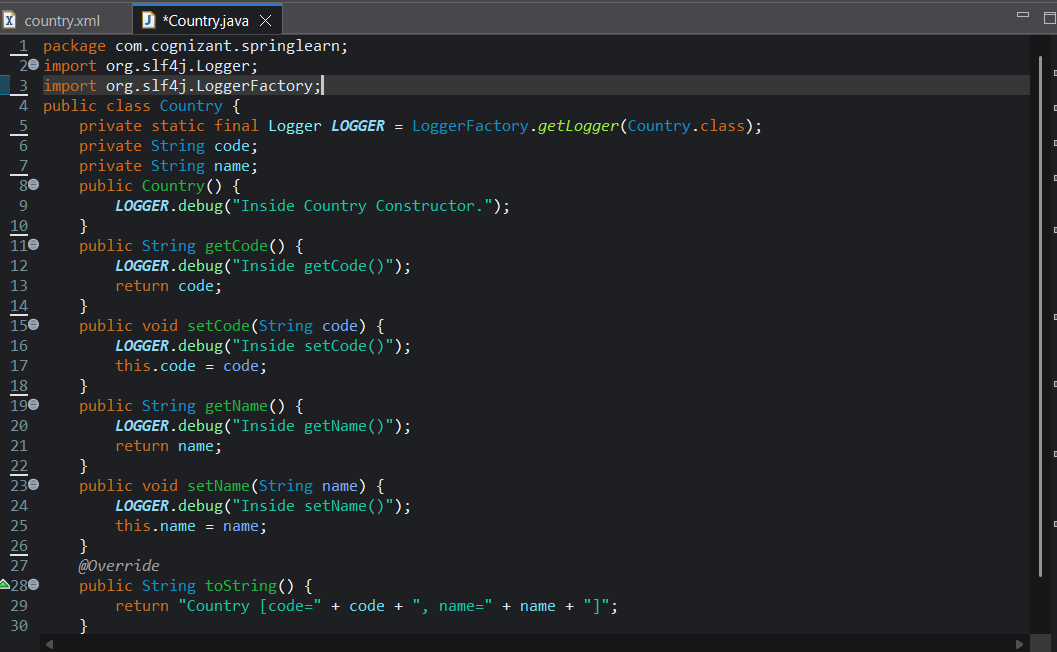
**Hands on 4**

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

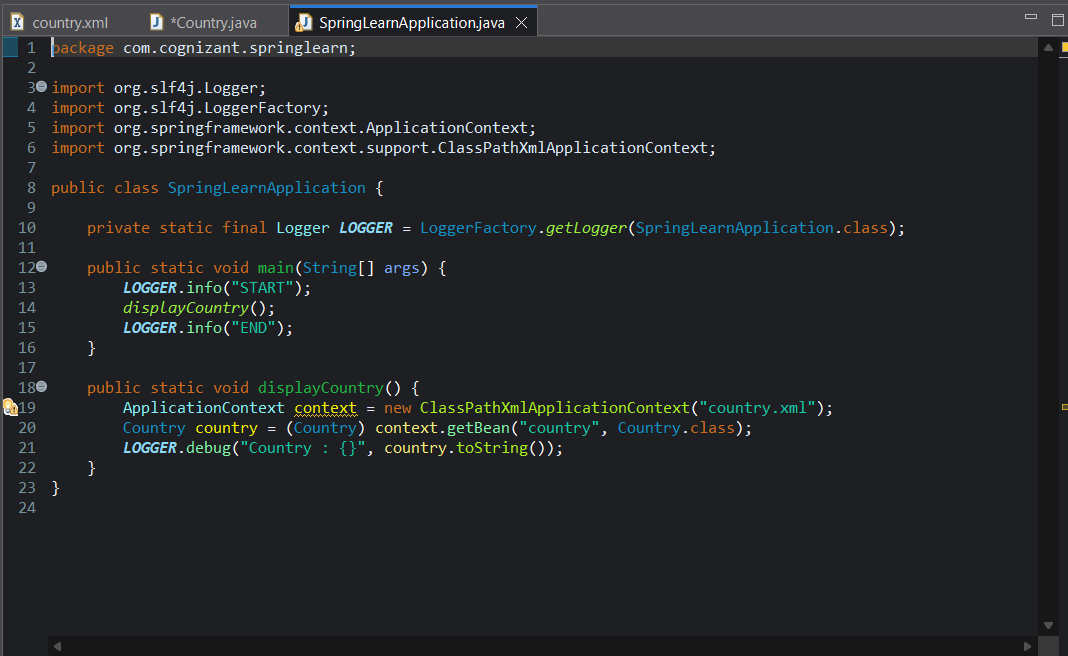
**1. country.xml config file bean definition of country class:**



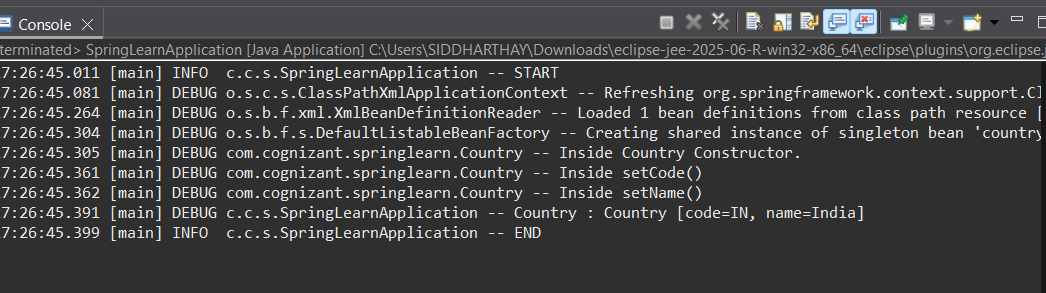
**2. Country.java shows all getter and setter methods**



**3. Main class shows displayCountry() method and how bean is loaded:**



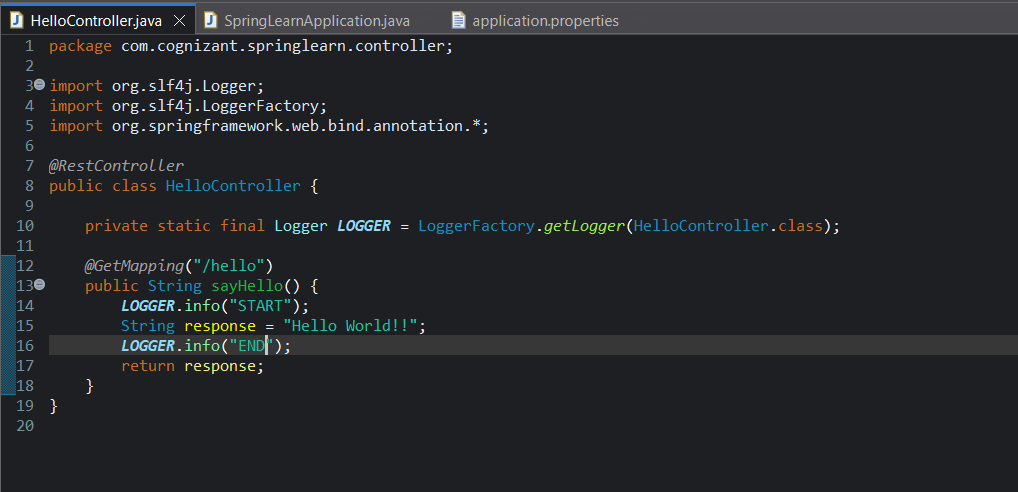
**4. Console showing debug logs:**



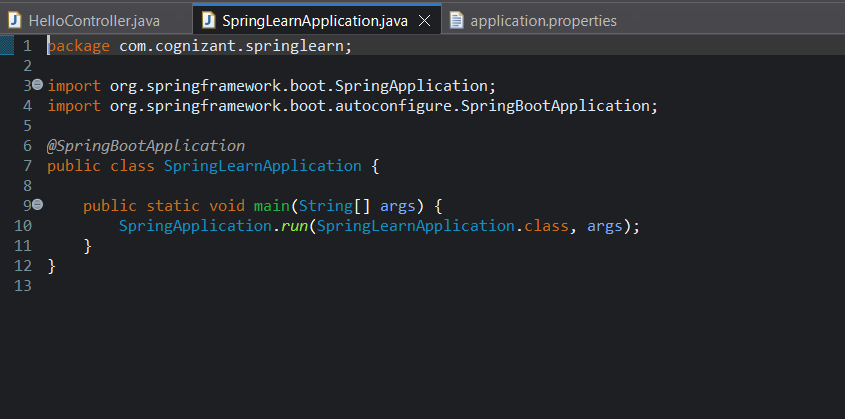
**2. Spring Rest HandsOn.docx**

**Hello World RESTful Web Service**   
  
Write a REST service in the spring learn application created earlier, that returns the text "Hello World!!" using Spring Web Framework. Refer details below:  
  
**Method:** GET  
**URL:** /hello  
**Controller:** com.cognizant.spring-learn.controller.HelloController  
**Method Signature:** public String sayHello()  
**Method Implementation:** return hard coded string "Hello World!!"  
**Sample Request**: http://localhost:8083/hello  
**Sample Response:** Hello World!!

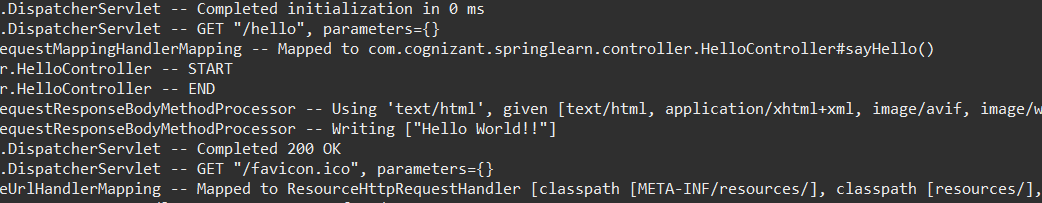
**1. HelloController.java Rest Controller:**



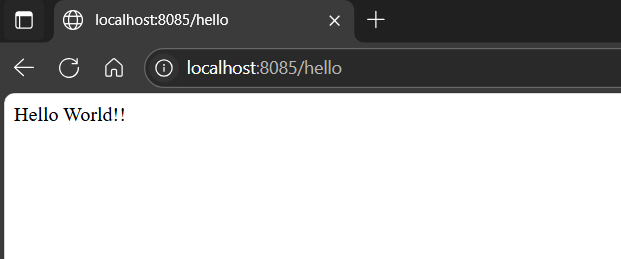
**2. Main() for running the application:**



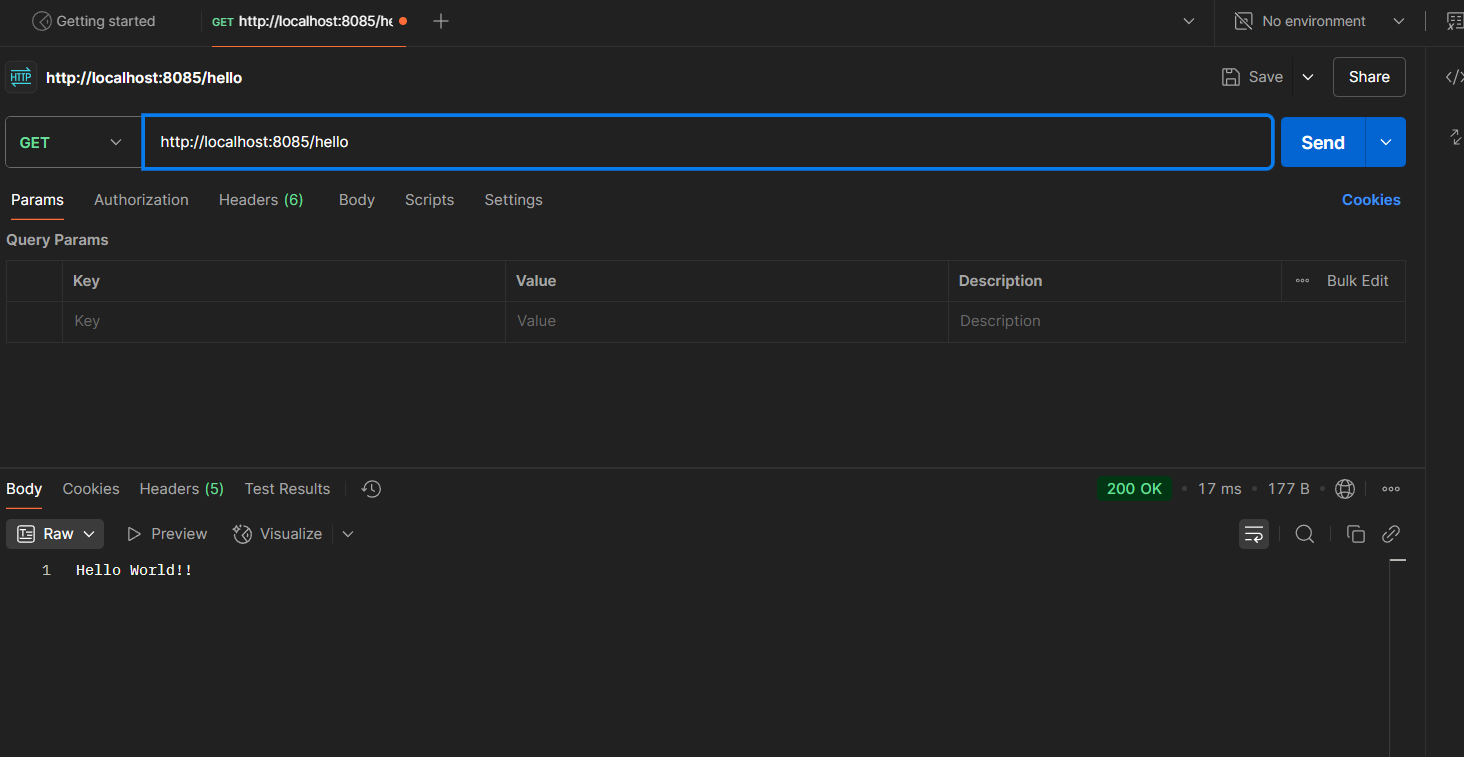
**3. Console:**



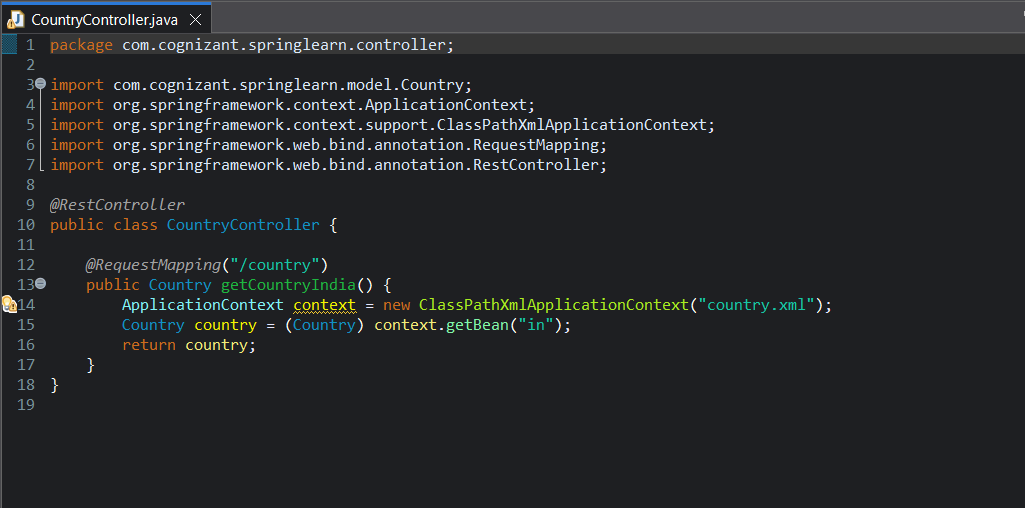
**4. Response through web browser:**



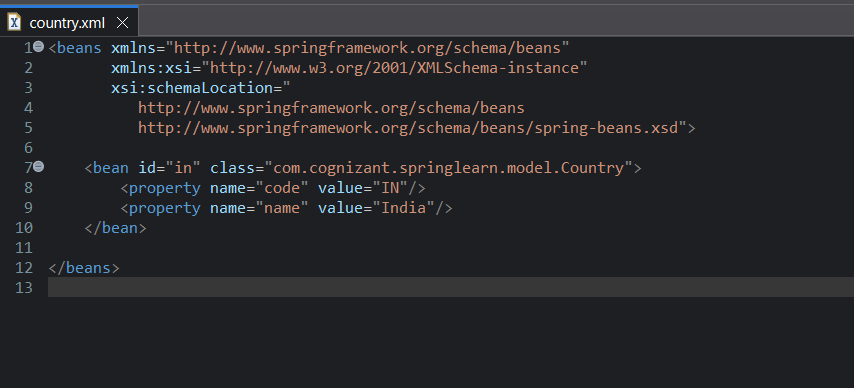
**5. Response through Postman:**



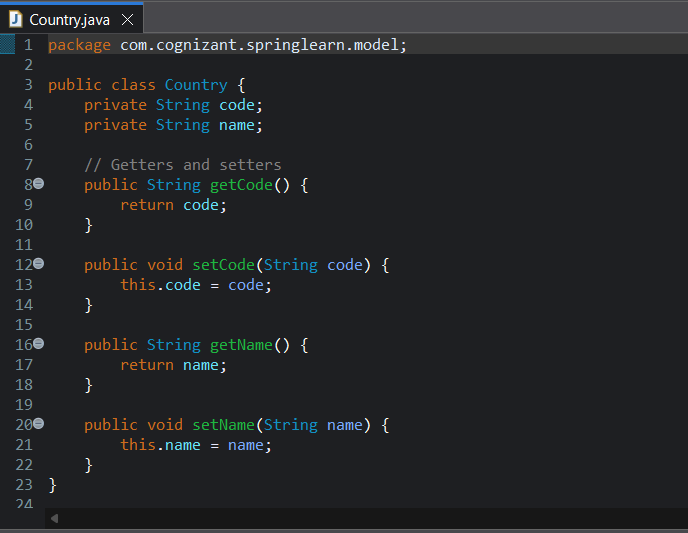
**REST - Country Web Service**   
  
Write a REST service that returns India country details in the earlier created spring learn application.

**1. CountryController.java handles REST API requests** 

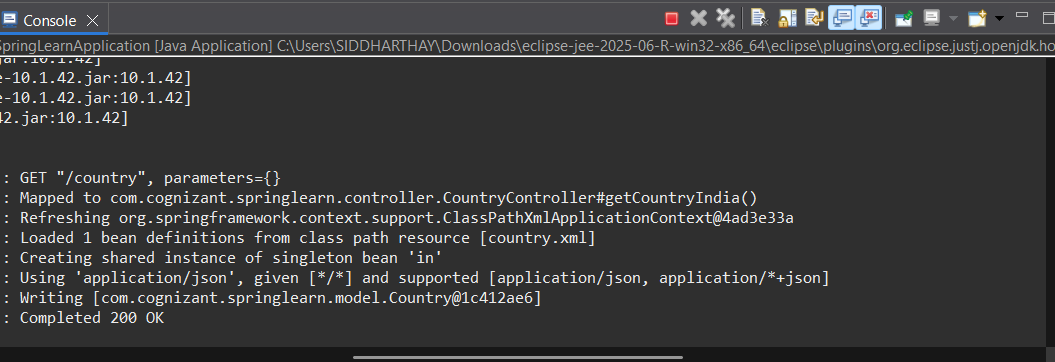
**2. country.xml configures country bean**



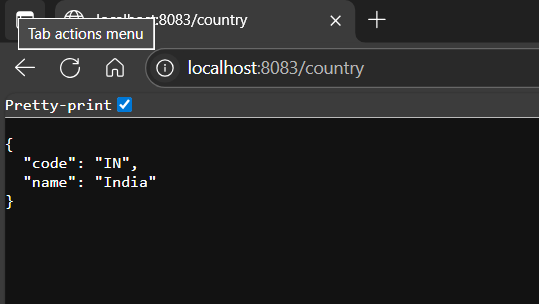
**3. Country.java model class contains getter and setter methods**



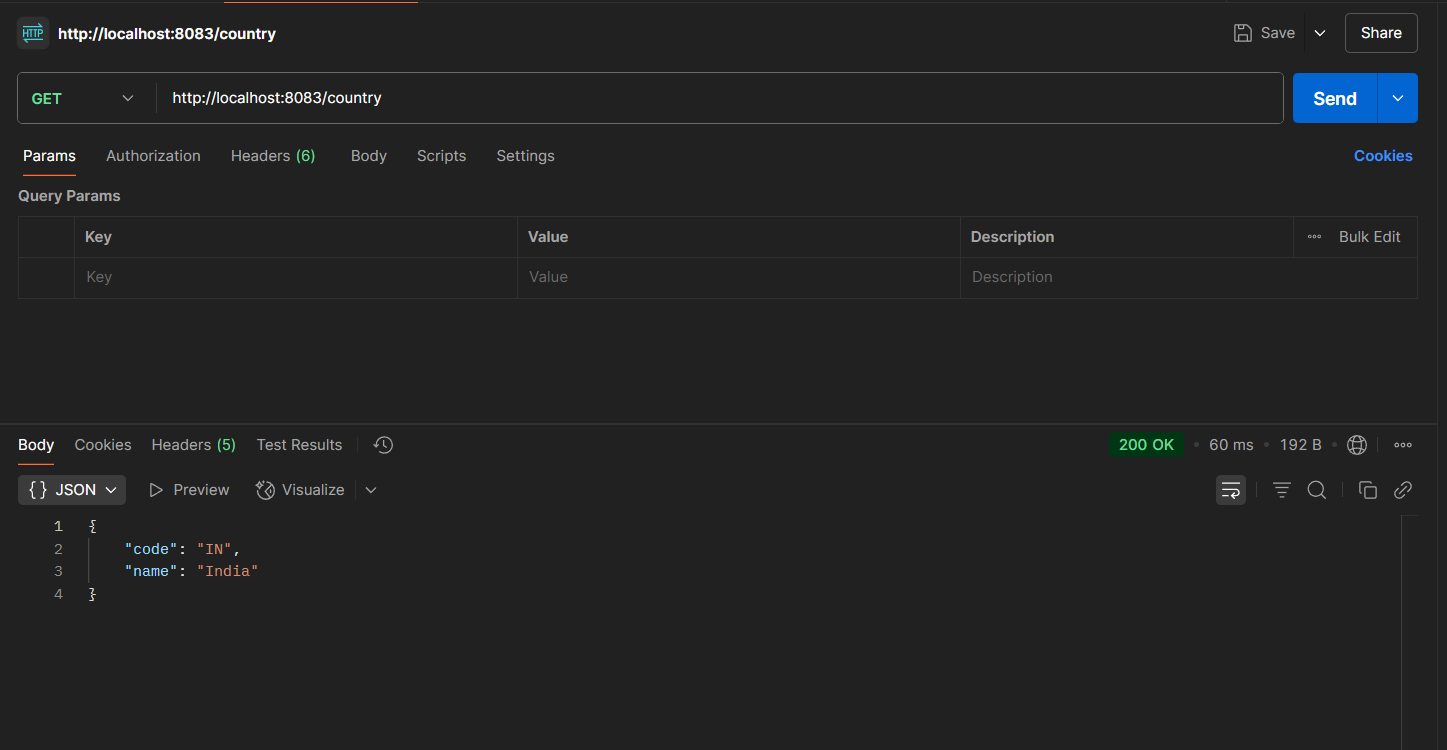
**4. Console output:**



**5. Response through web browser:**

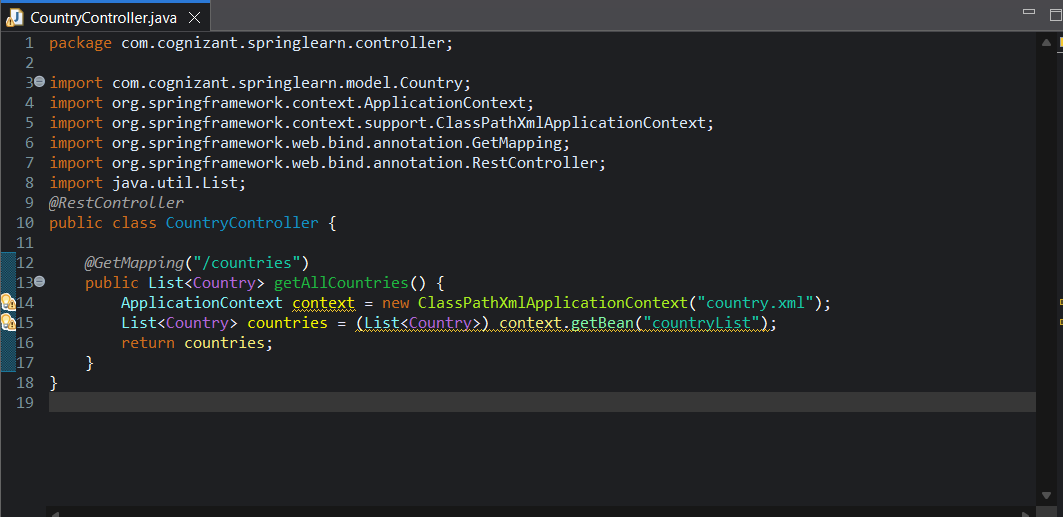


**6. Response through Postman:**

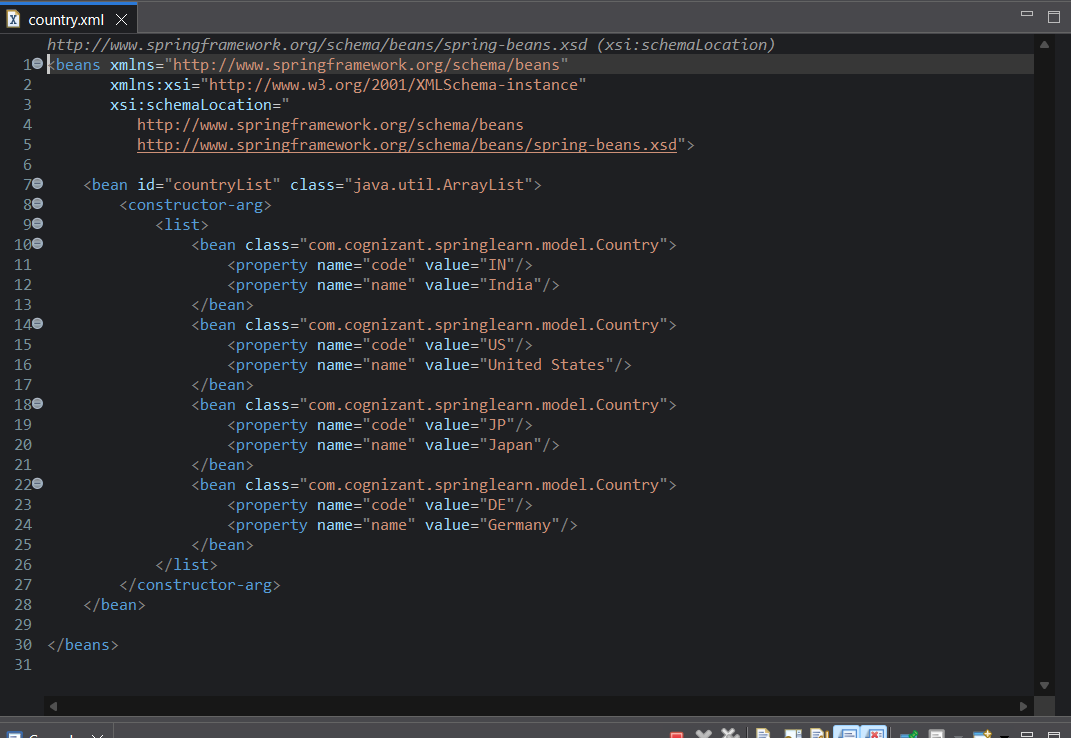


**REST - Get all countries**   
Write a REST service that returns all the countries.

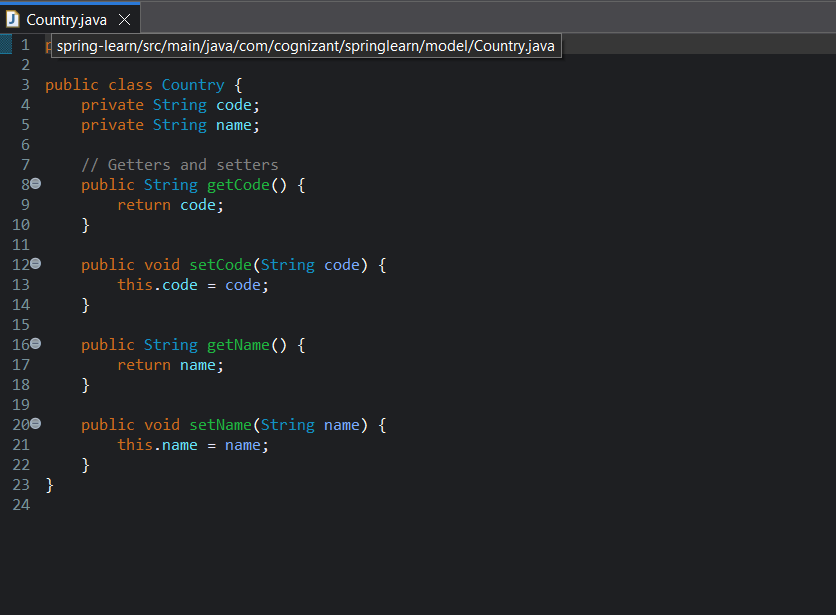
**1. CountryController.java to handles REST API requests:**



**2. country.xml configures all country bean**



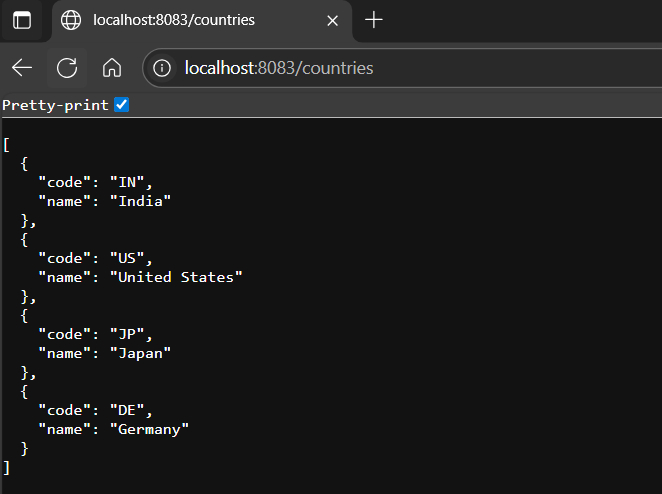
**3. Country.java model class contains getter and setter methods**



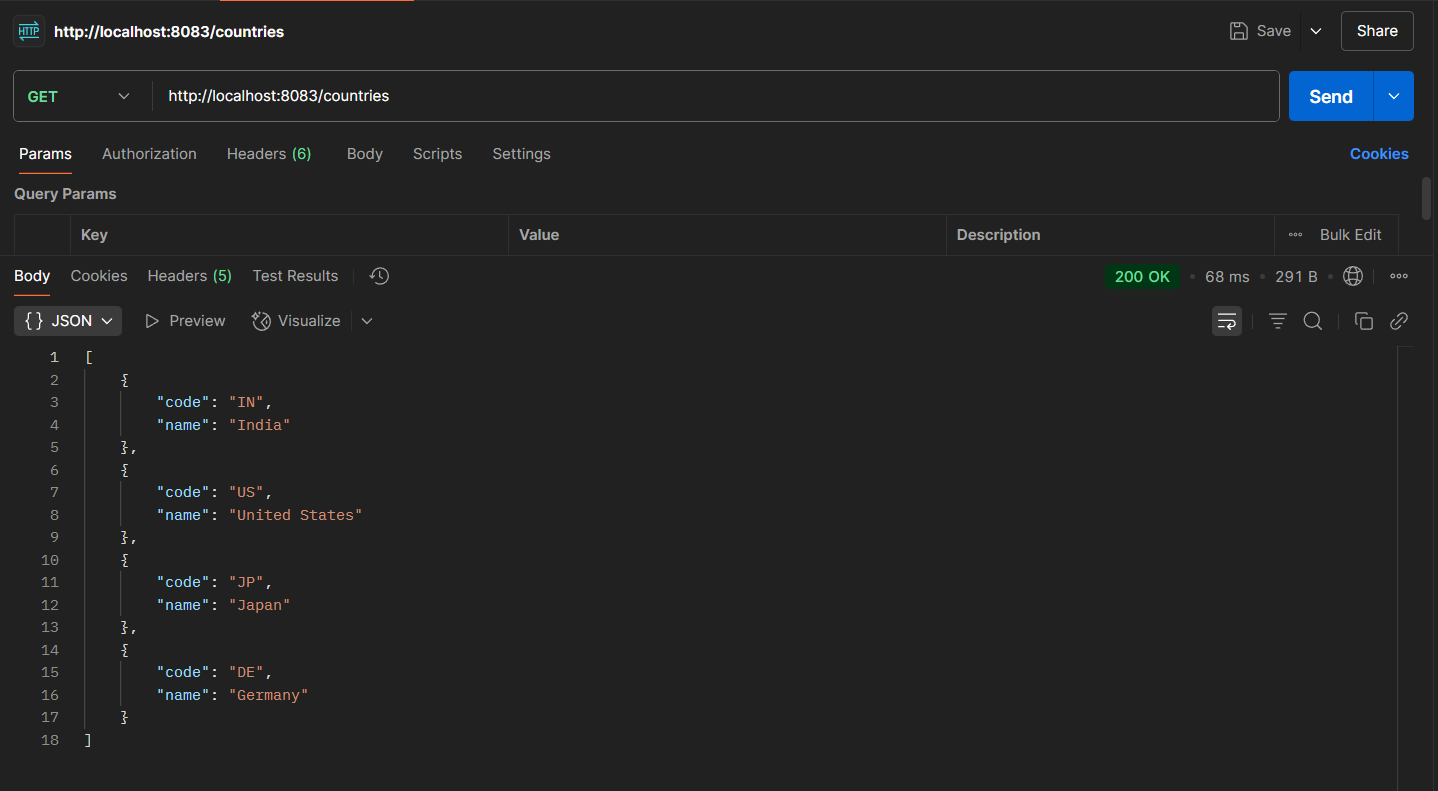
**4. Console Output:**



**5. Response through Web Browser:**

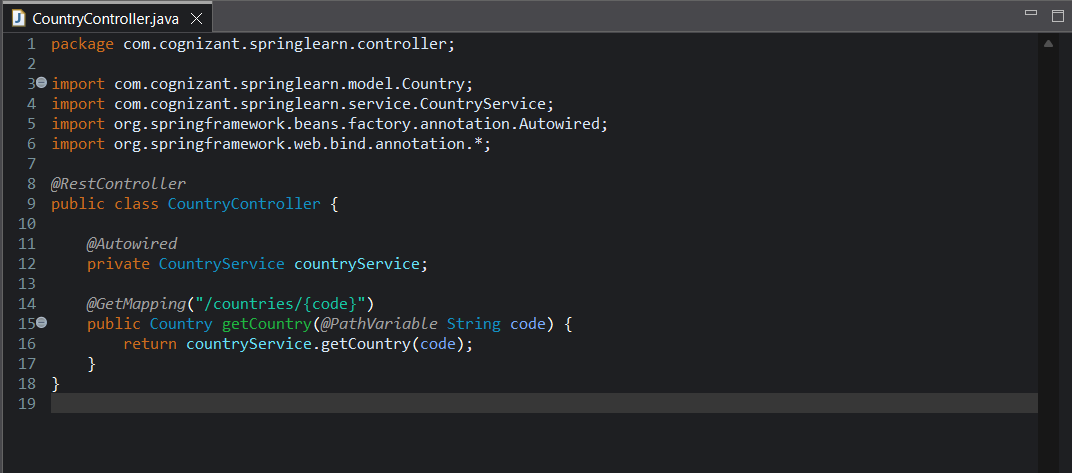


**5. Response through Postman:**

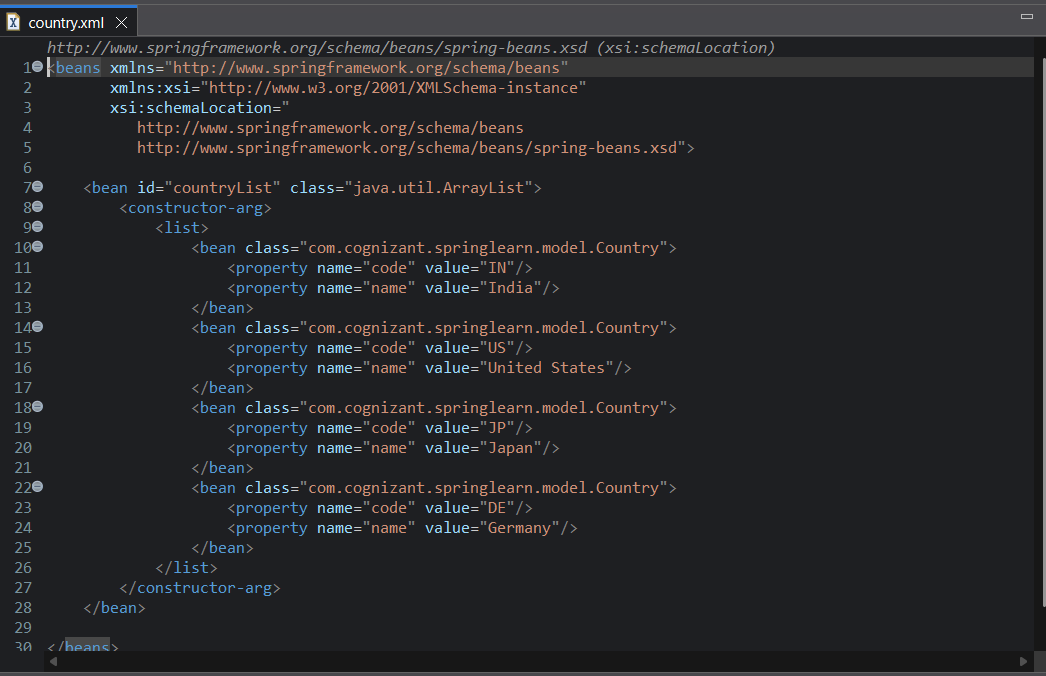


**REST - Get country based on country code**   
  
Write a REST service that returns a specific country based on country code. The country code should be case insensitive.

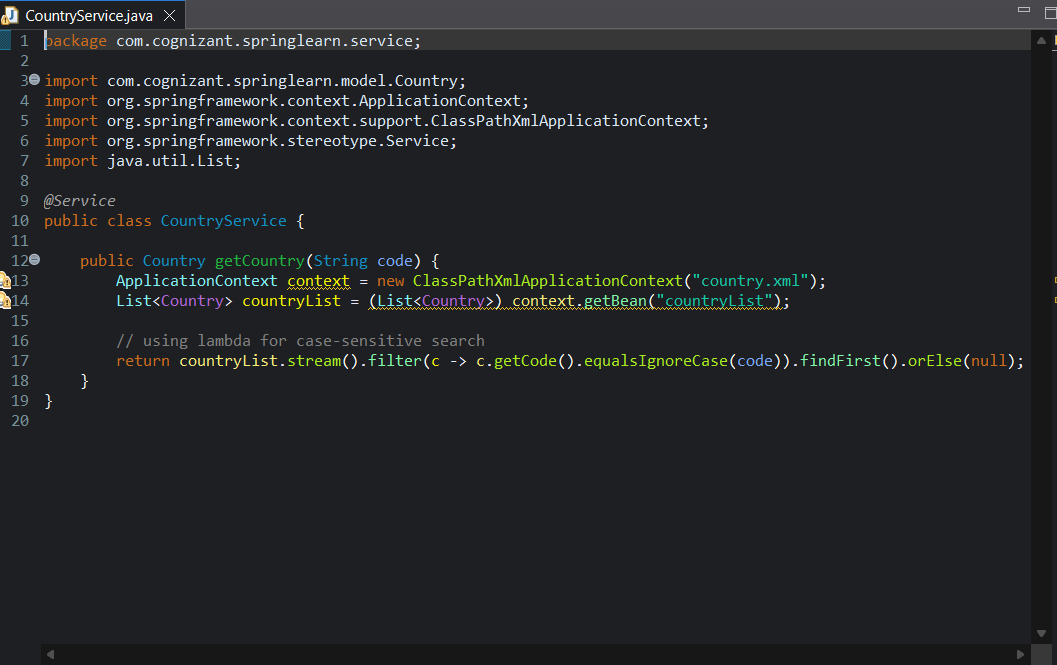
**1. CountryController.java to handles REST API requests:**



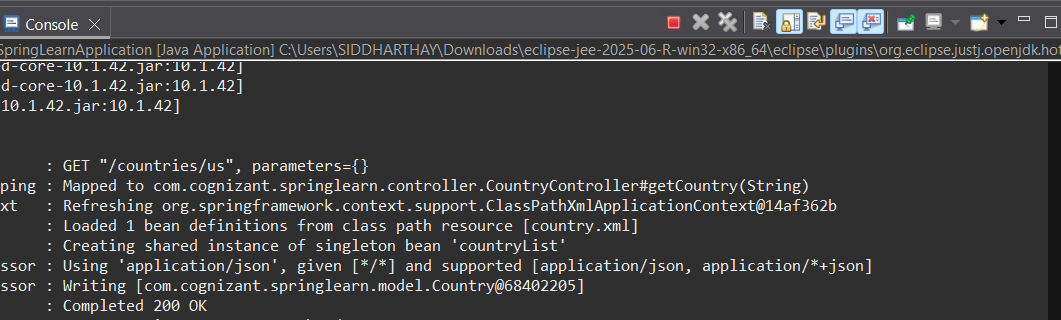
**2. country.xml configures all country bean**



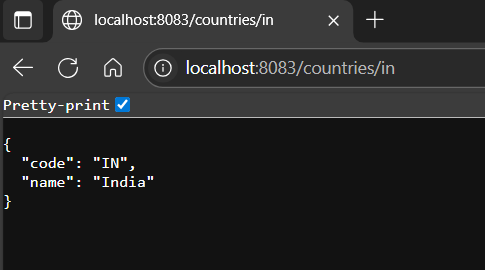
**3. CountryService.java contains the business logic:**

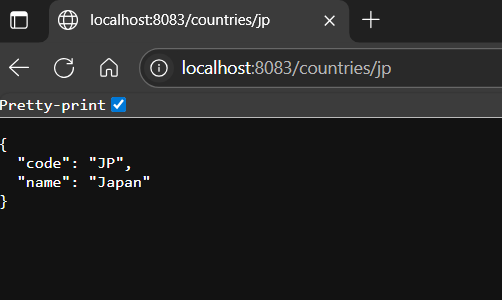


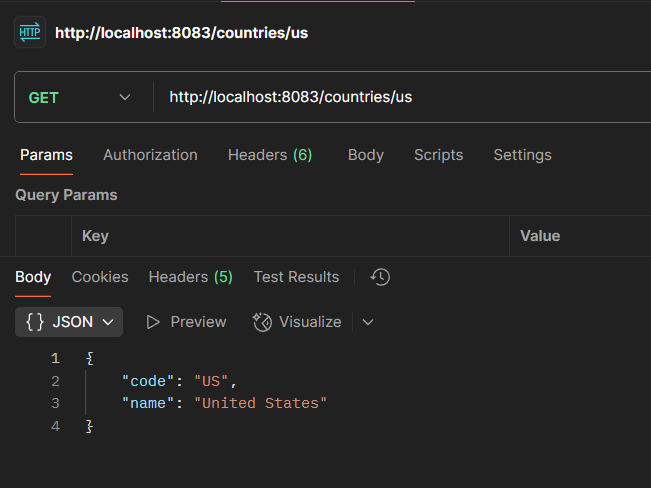
**4. Console Output:**



**5. Responses through Web Browser and Postman:**



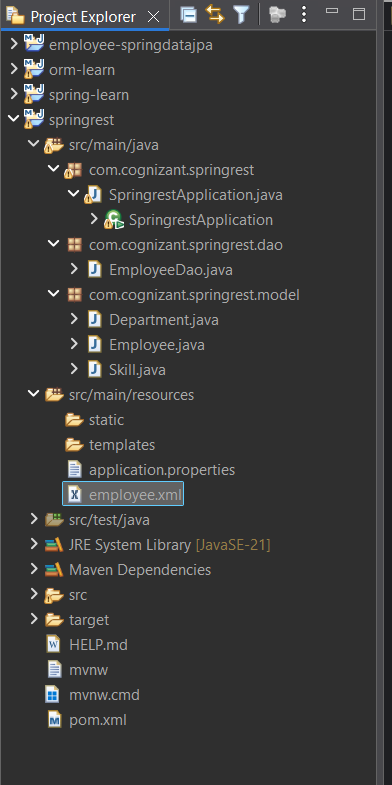




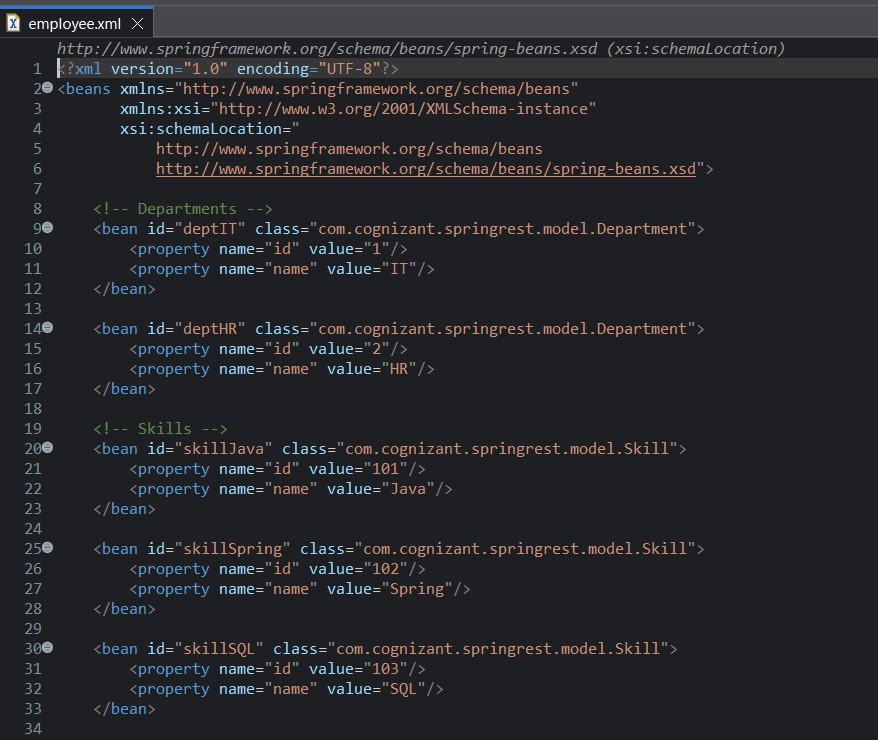
**3. Spring Rest HandsOn.docx**

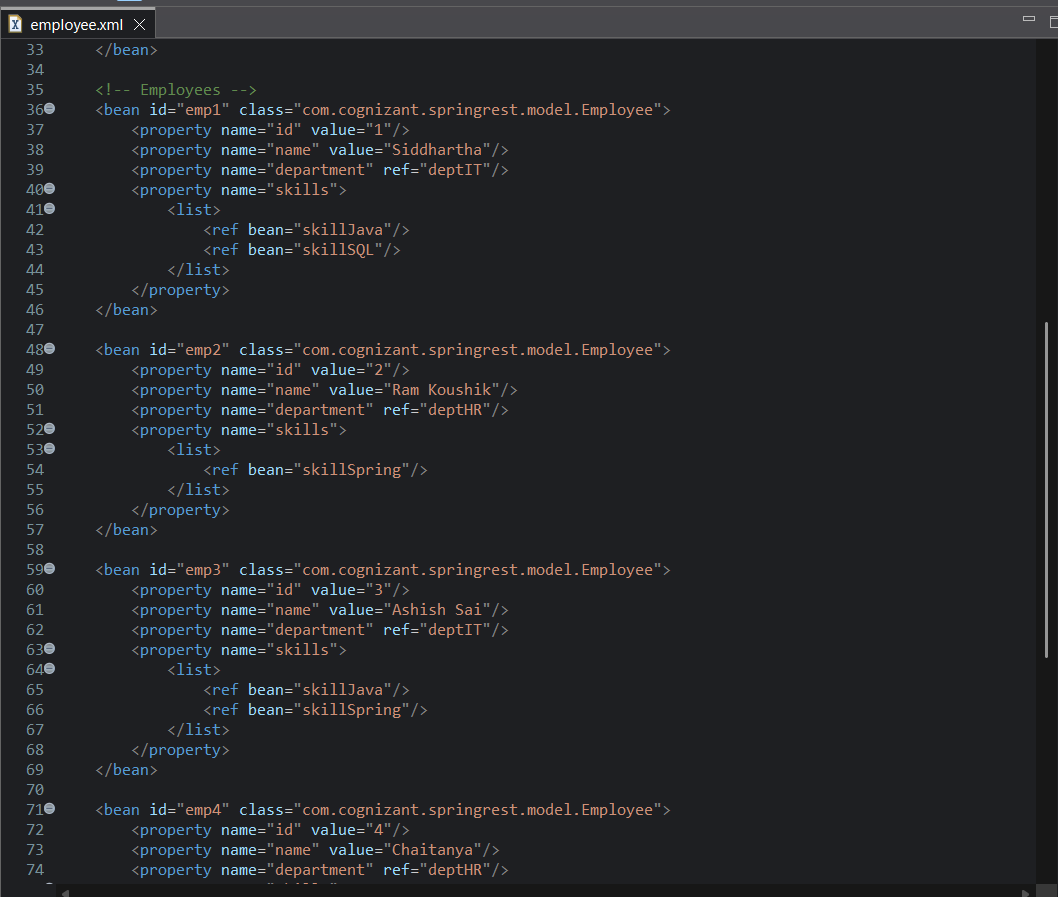
**Create static employee list data using spring xml configuration**

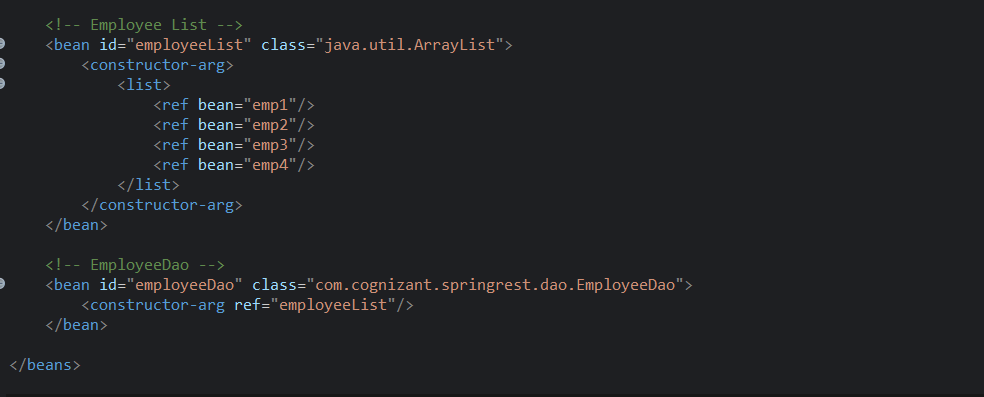
**1. Project Structure**



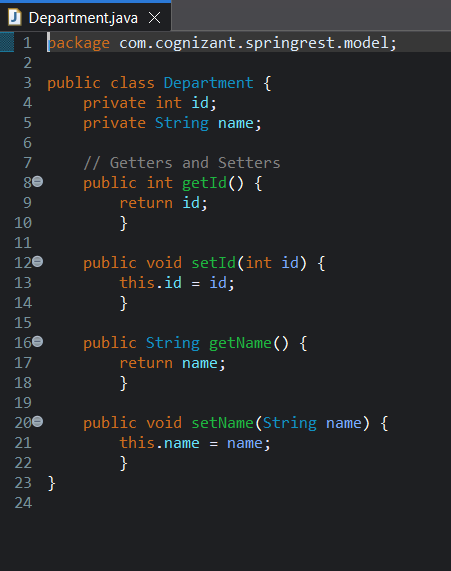
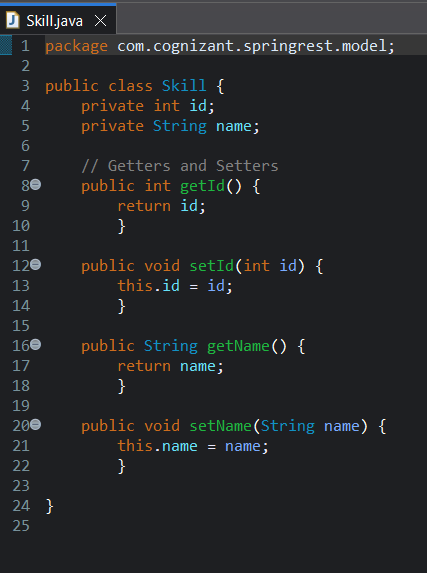
**2. employee.xml for bean configuration**







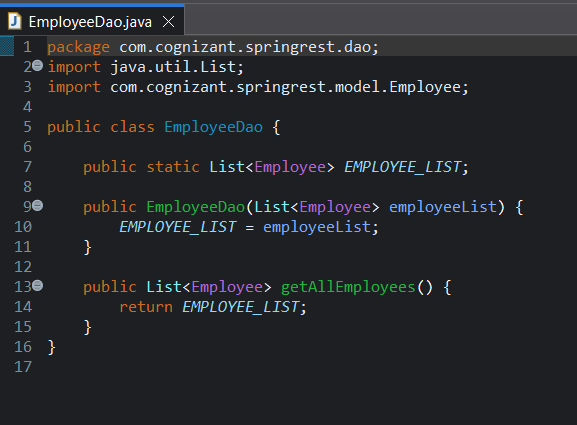
**3,4 Department,Skill.java contain getter and setter methods for departments and Skills**

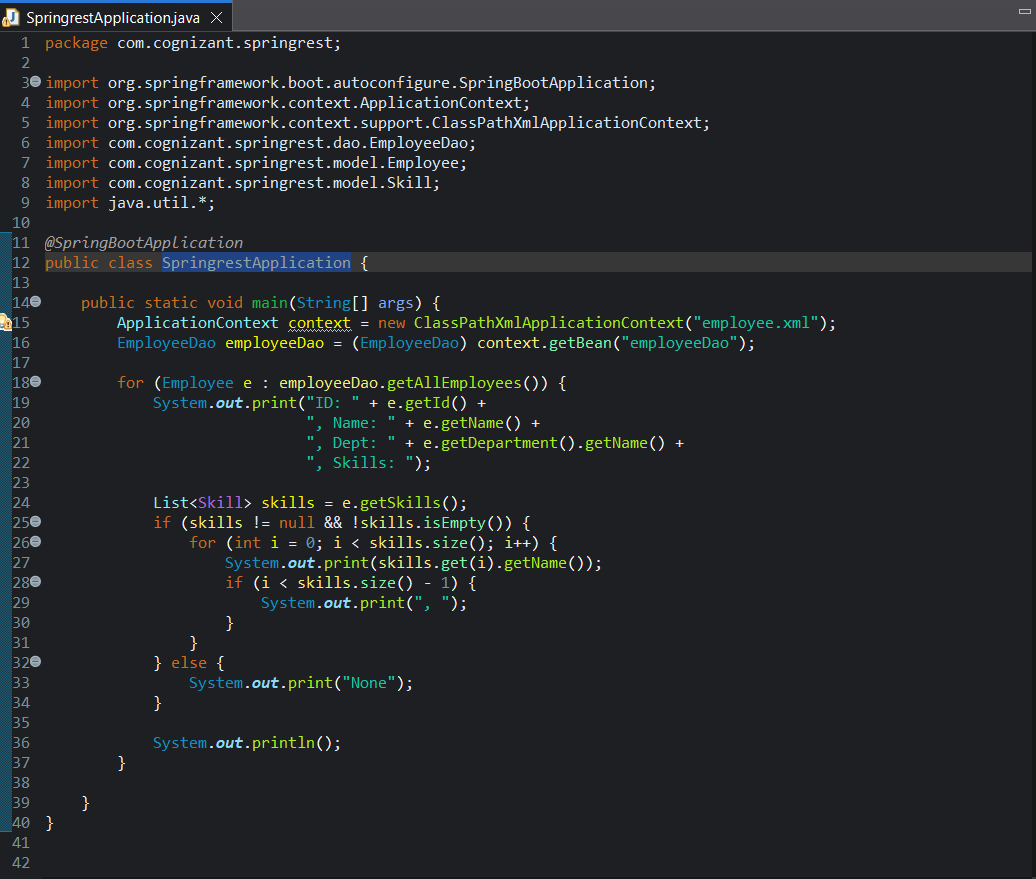
**5. Employee.java contains employee getter and setter methods**



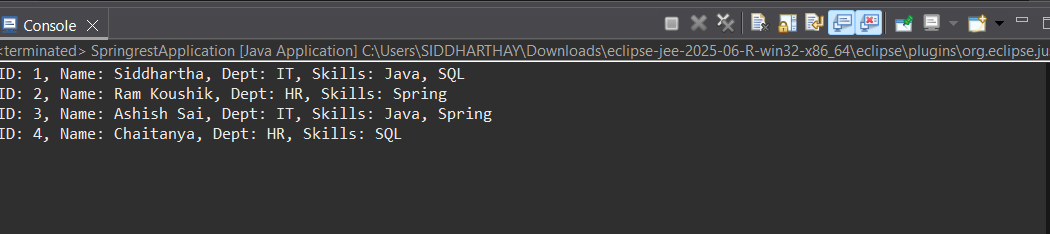
**6. EmployeeDao.java returns Employee list**



**7. SpringrestApplication Main Class prints the employee details**



**8. Console Output:**



**Create REST service to gets all employees**

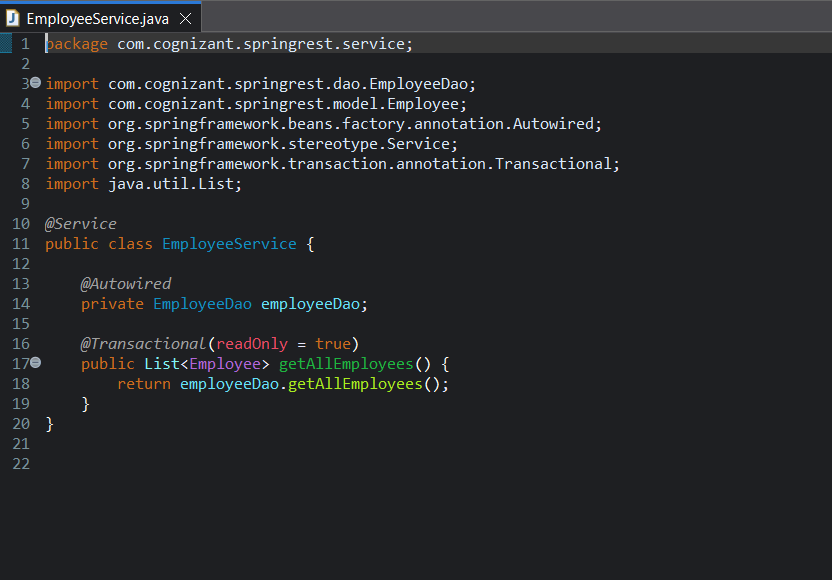
* In EmployeeService, incorporate the following:
  + Change the annotation for this class from @Component to @Service
  + Create method getAllEmployees() that invokes employeeDao.getAllEmployees() and return the employee list
  + Define @Transactional annotation for this method.

* In EmployeeController, incorporate the following:
  + Include a new get method with name getAllEmployees() that returns the employee list
  + Mark this method as GetMapping annotation with the URL as '/employees'
  + Within this method invoke employeeService.getAllEmployees() and return the same.

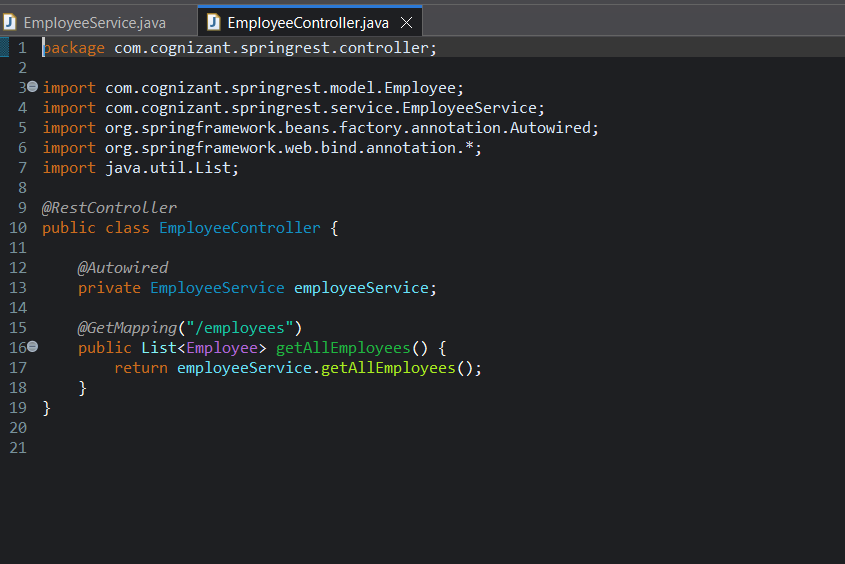
​

* Test ​the service using postman.

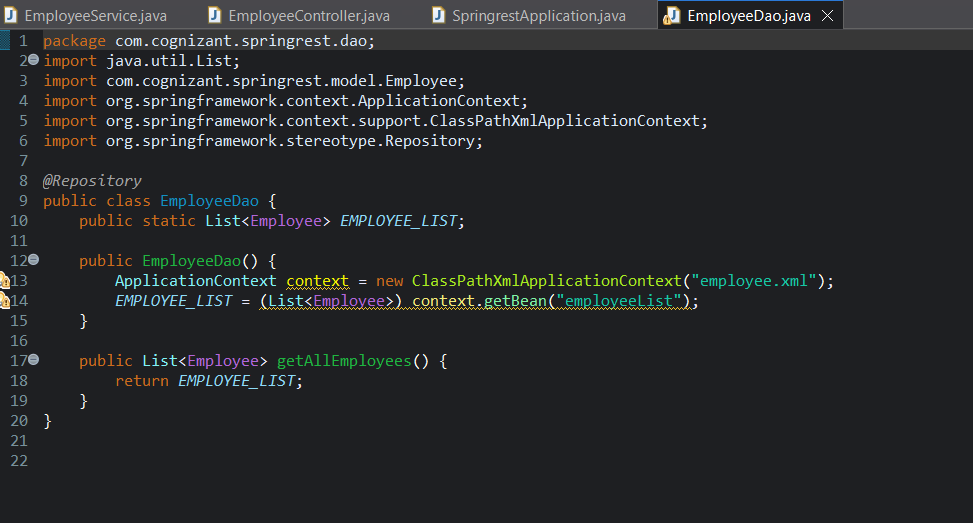
**1. EmployeeService.java:**



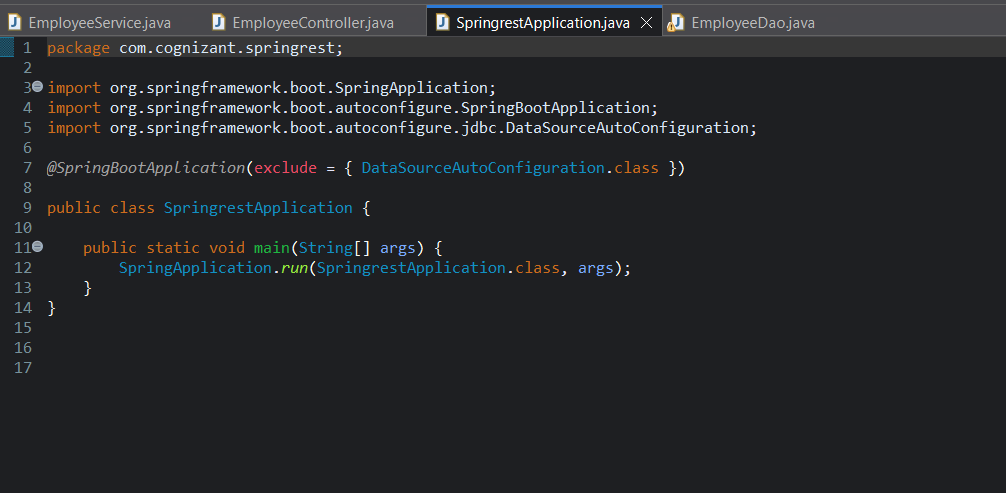
**2. EmployeeController.java:**



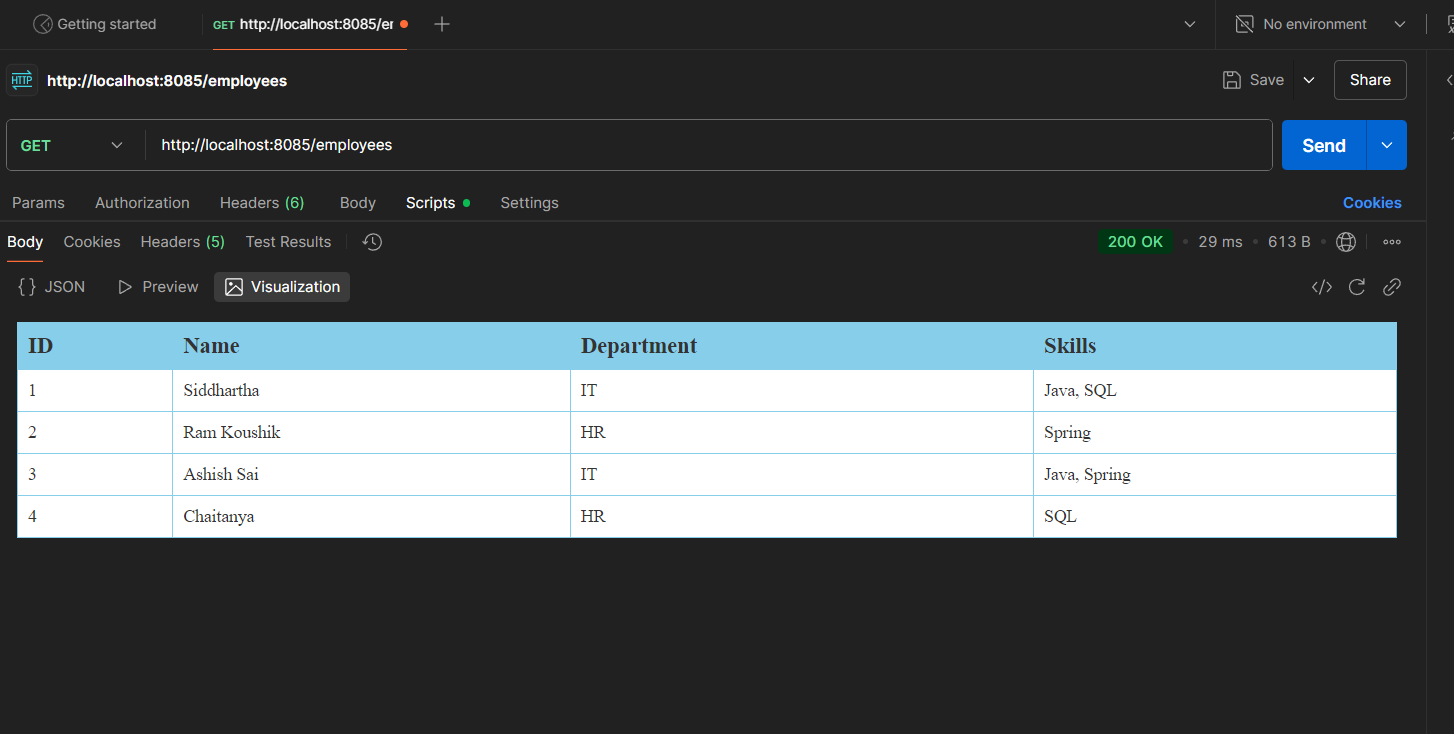
**3. EmployeeDao.java:**



**4. Main Class SpringrestApplication.java:**



**5. Testing in Postman:**

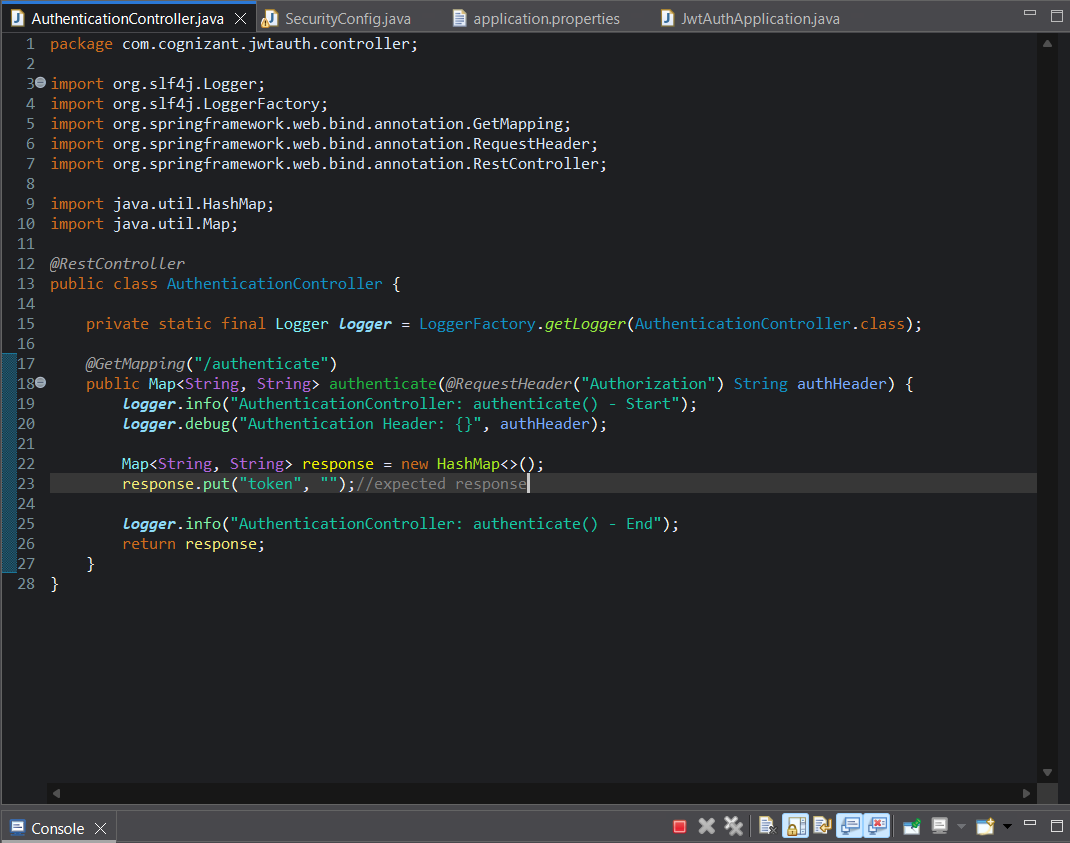


**5. JWT – HandsOn**

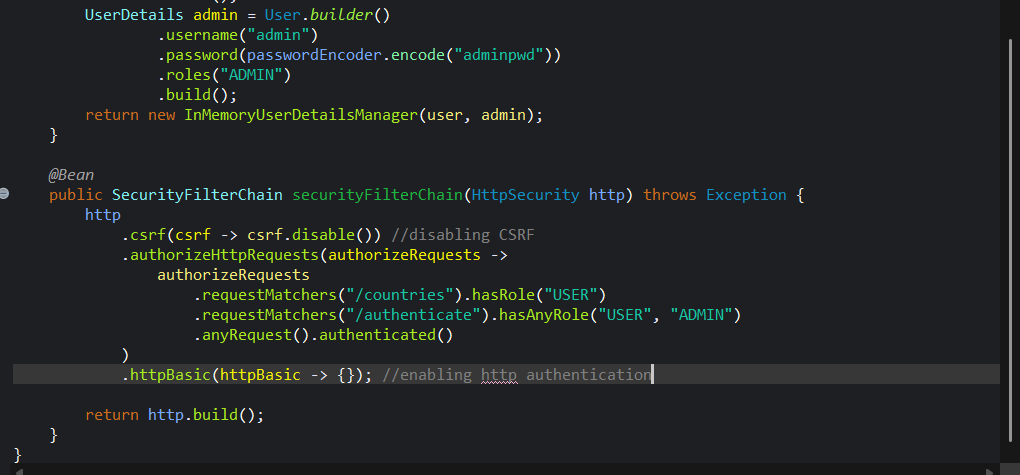
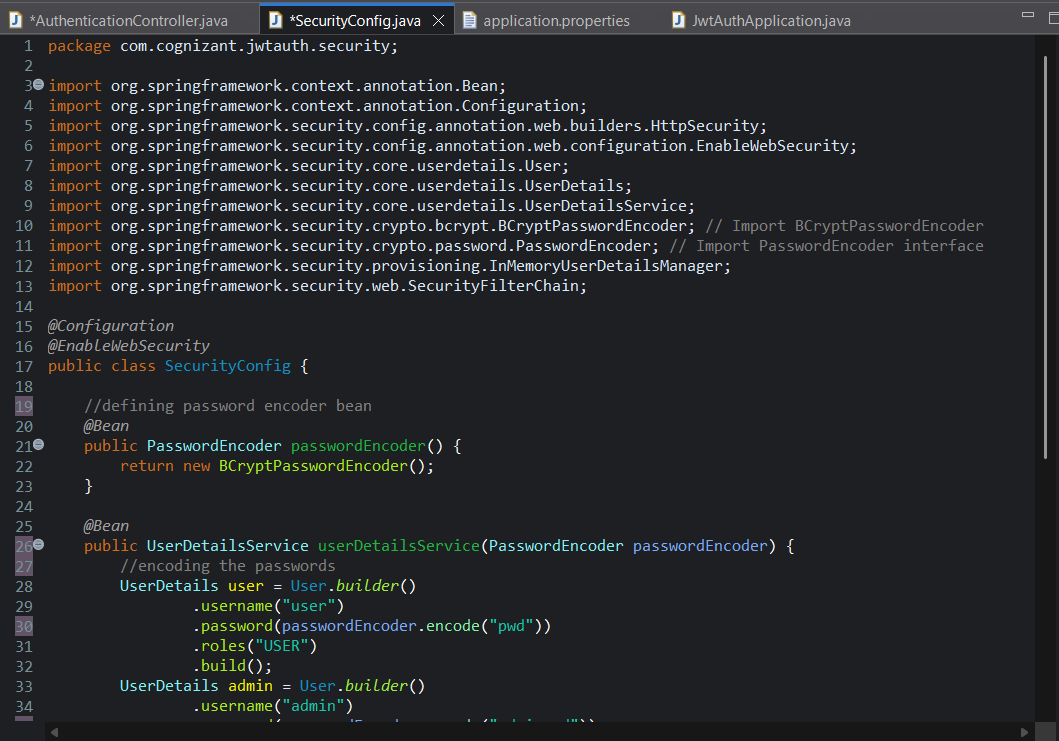
**Create authentication service that returns JWT**

* **Create authentication controller and configure it in SecurityConfig**

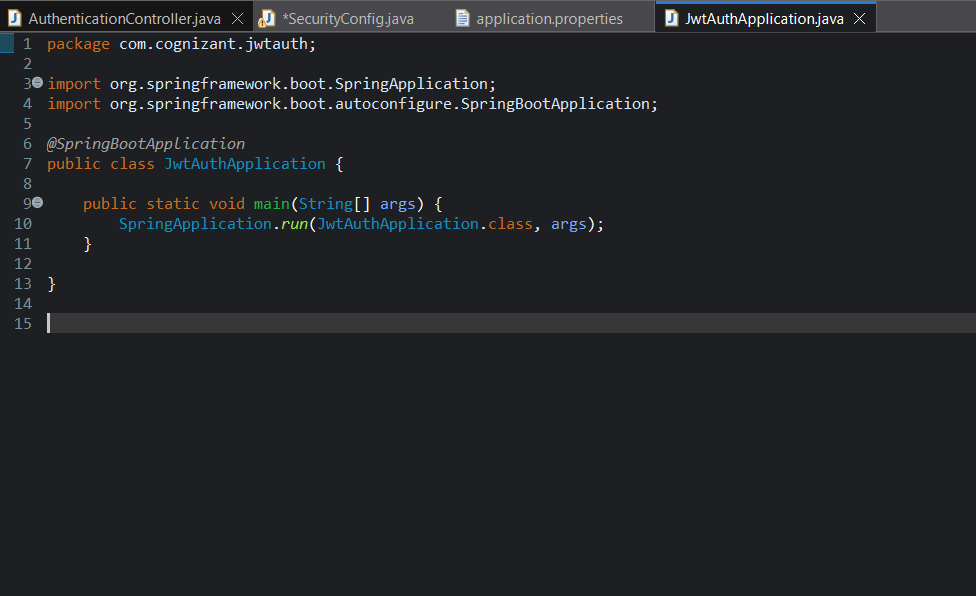
**1. AuthenticationController.java:**

****

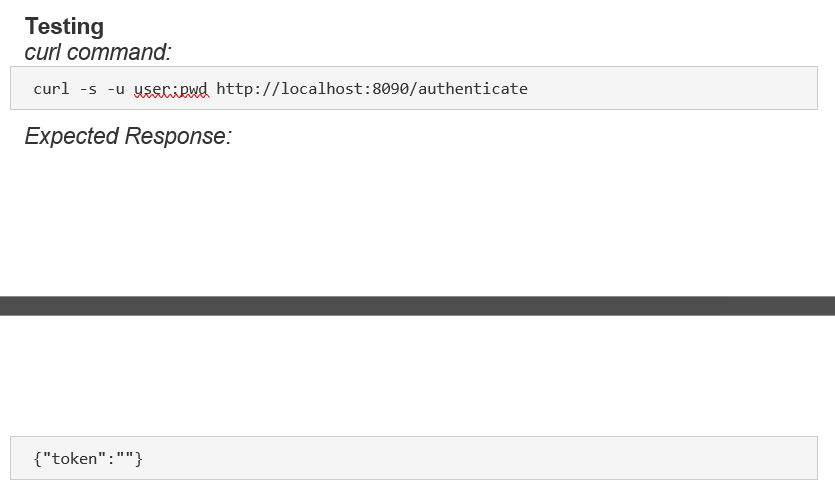
**2. SecurityConfig.java:**



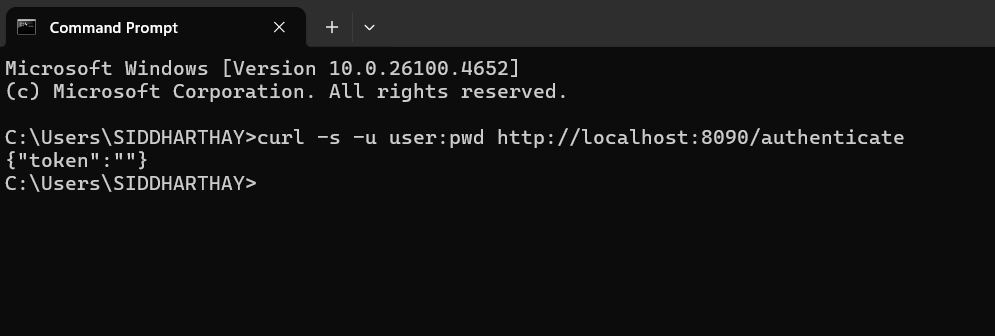
**3. Main SpringBootApplication Class:**

****

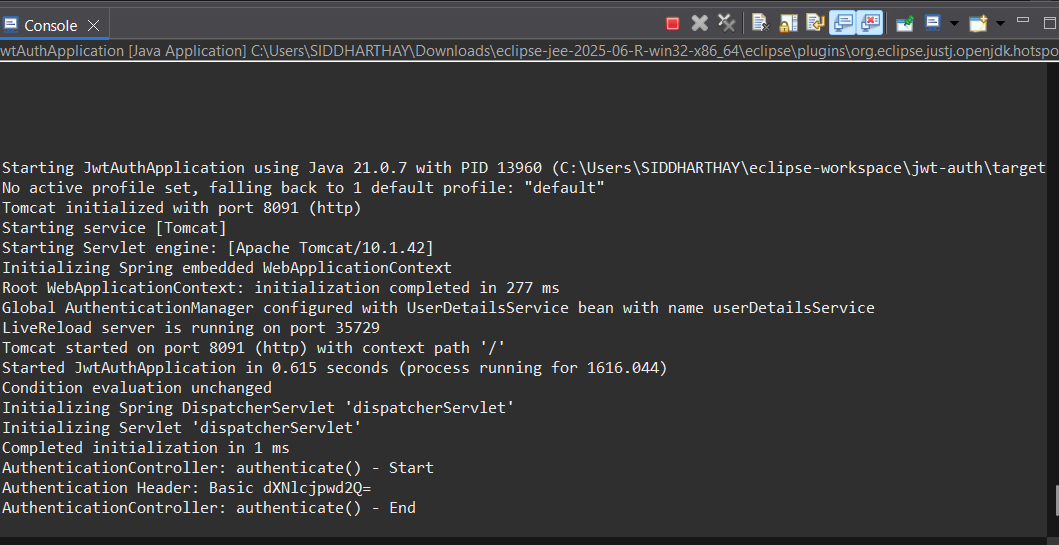
**4. Expected Response:**

****

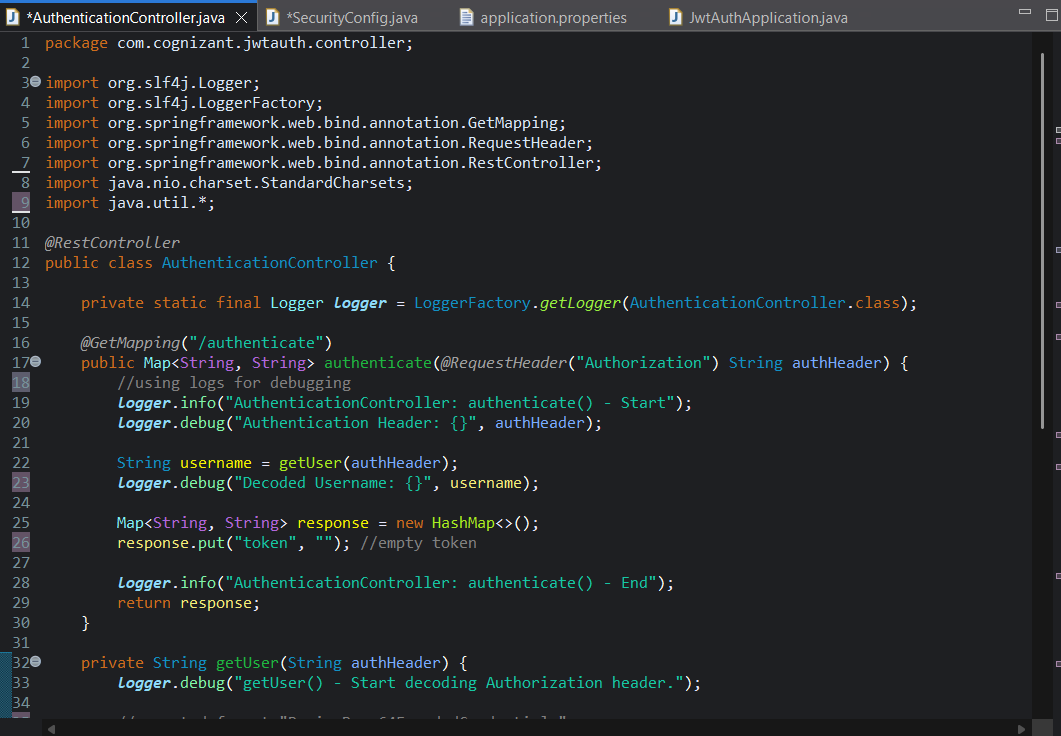
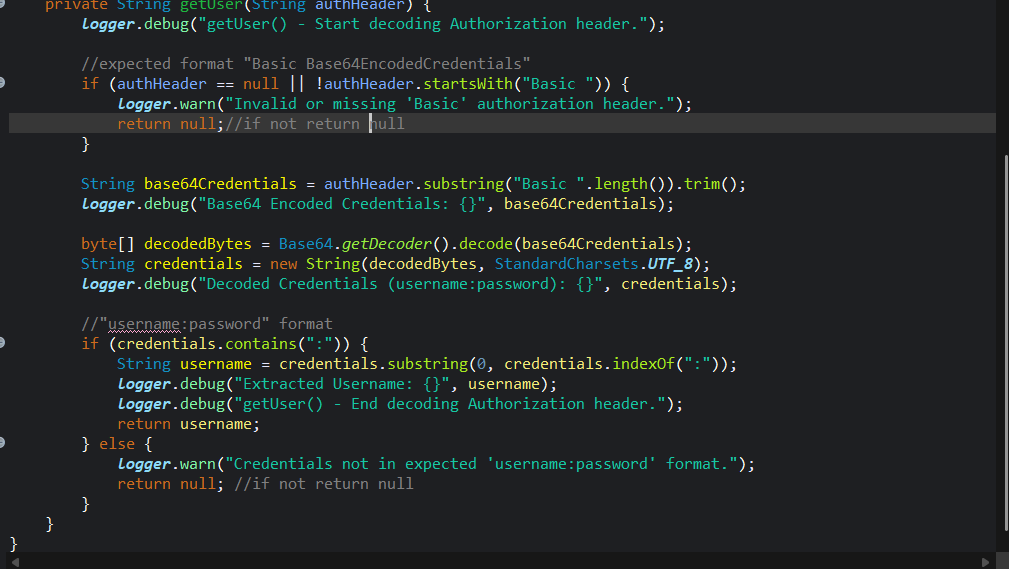
**5. Actual Response:**

****

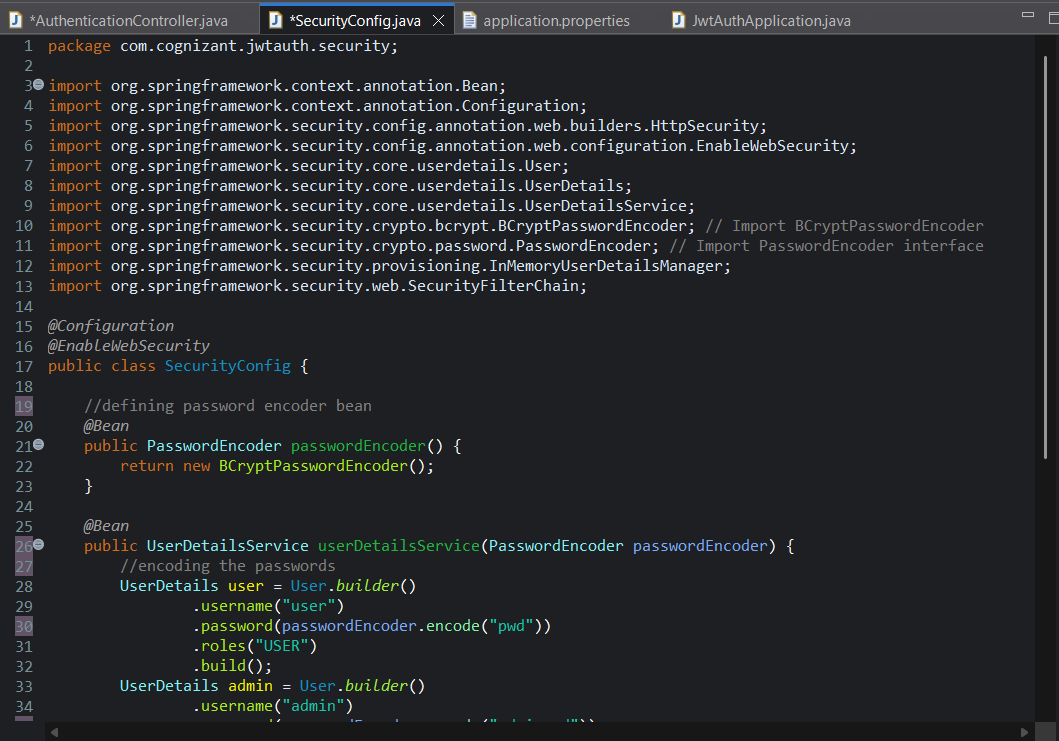
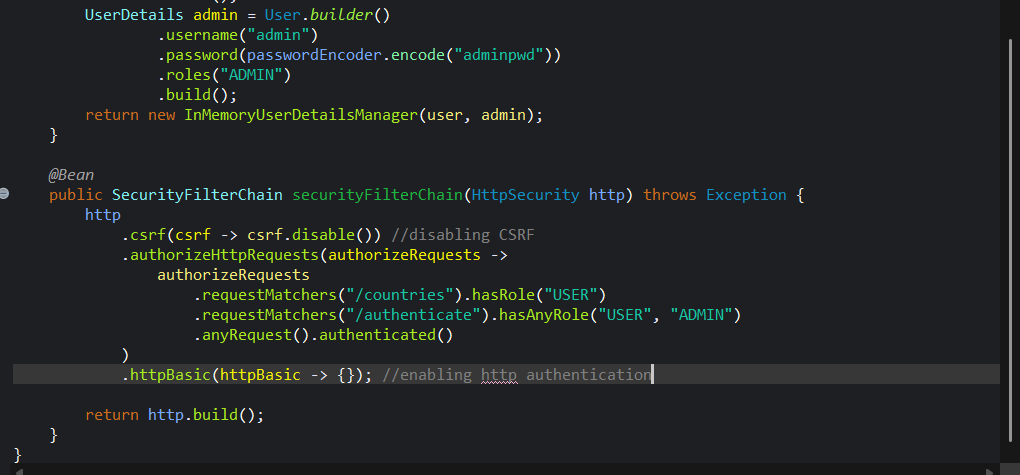
**6. Console Log Verification:**

****

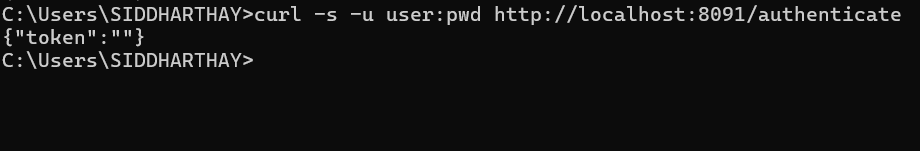
* **Read Authorization header and decode the username and password**

**1. AuthenticationController.java:**  
 

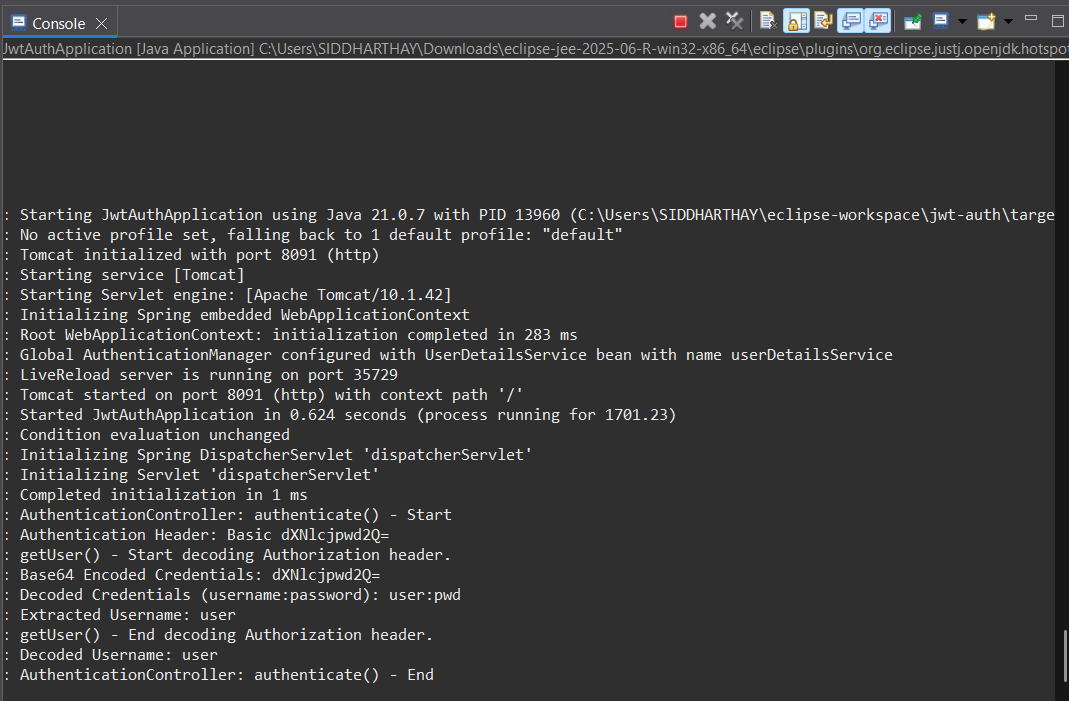
**2. SecurityConfig.java:**

**3. Response**

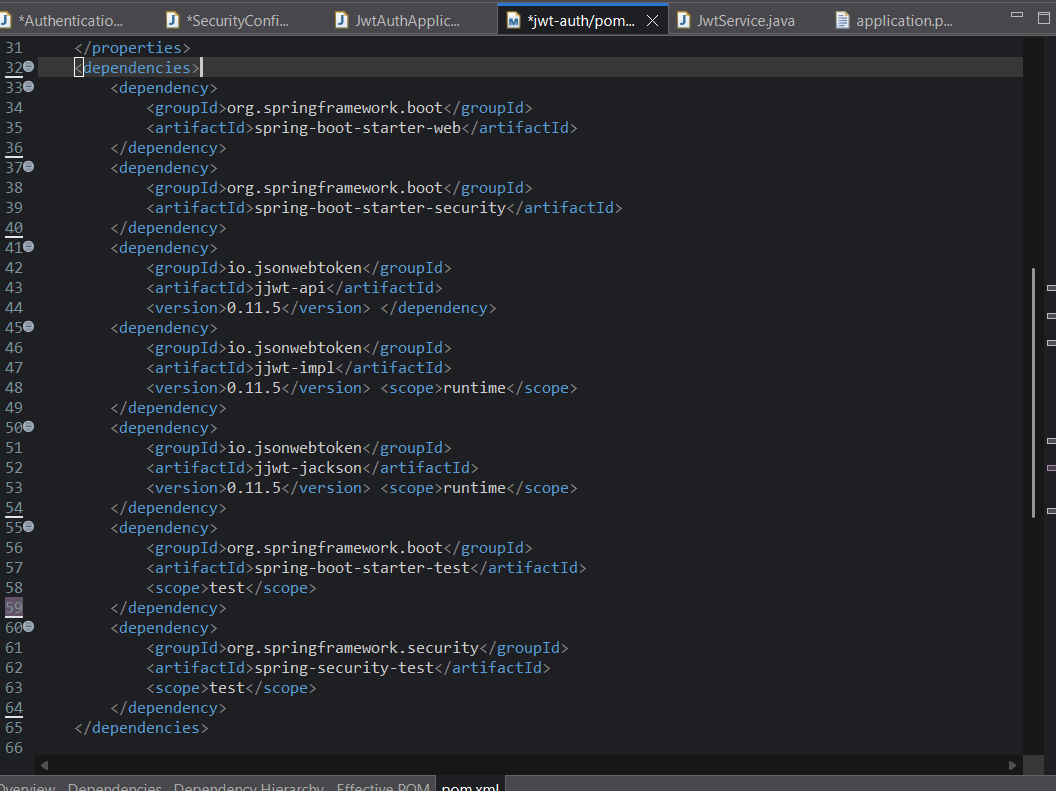


**4. Console Log Verification:**

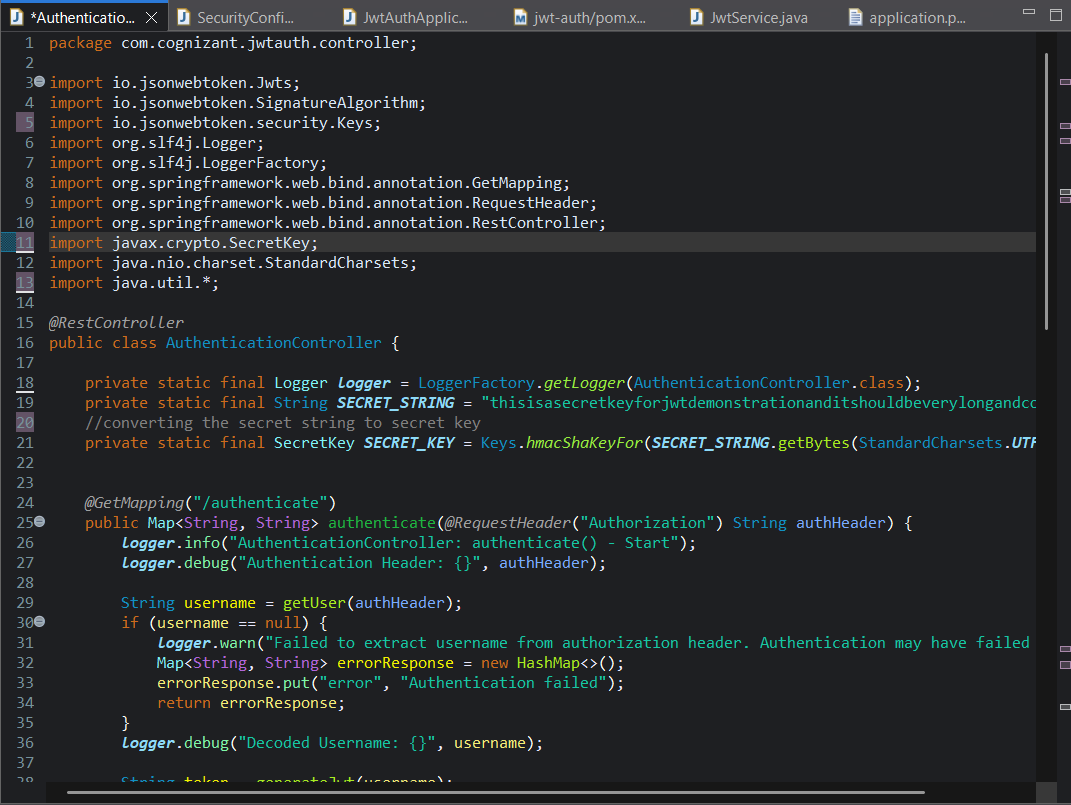


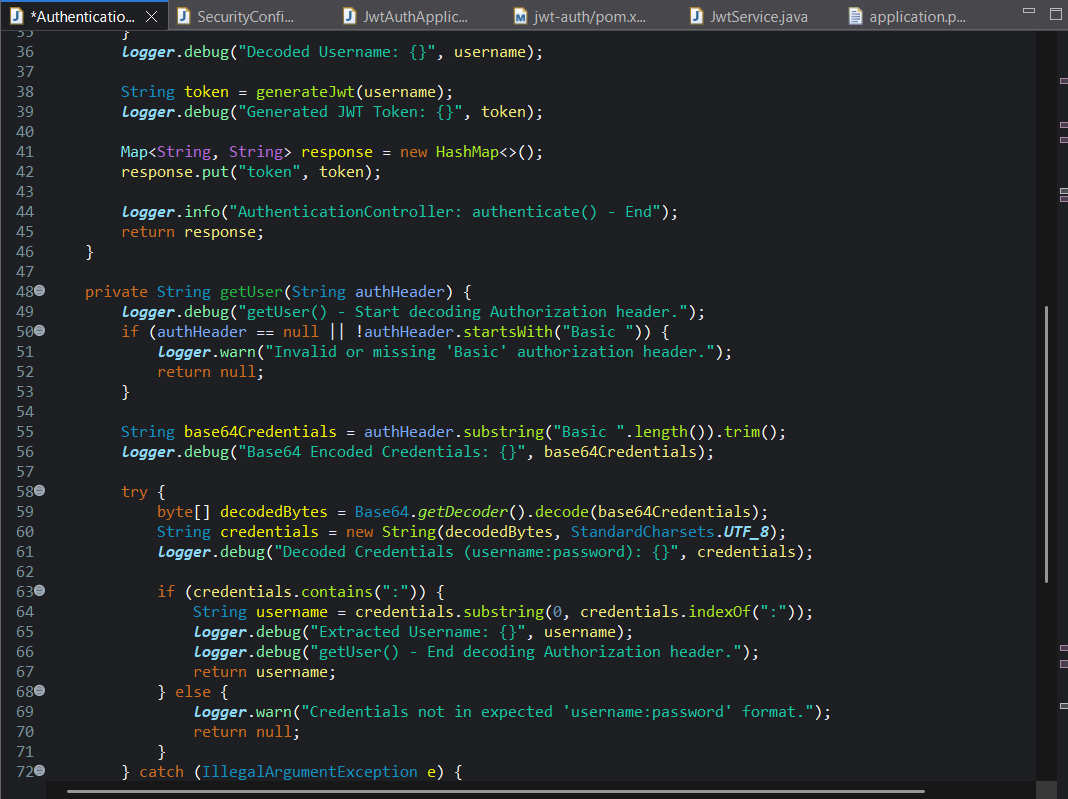
* **Generate token based on the user**

**1. adding the dependencies:**

****

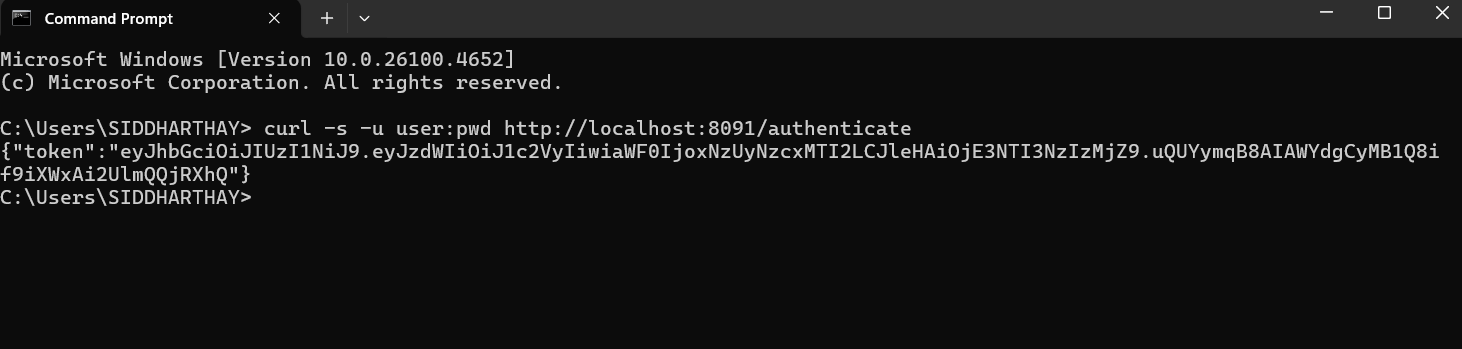
**2. AuthenticationController.java:**

****

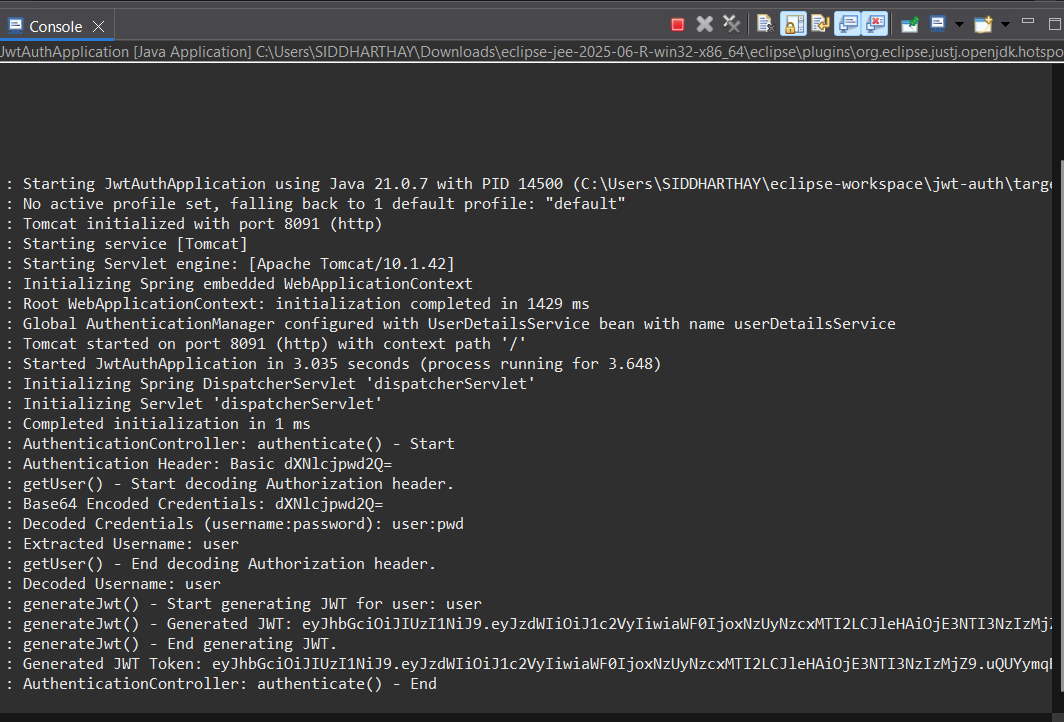
****

****

**3. curl command response:**

****

**4. Console Log verification:**

****