**Handson - Spring REST using Spring Boot**

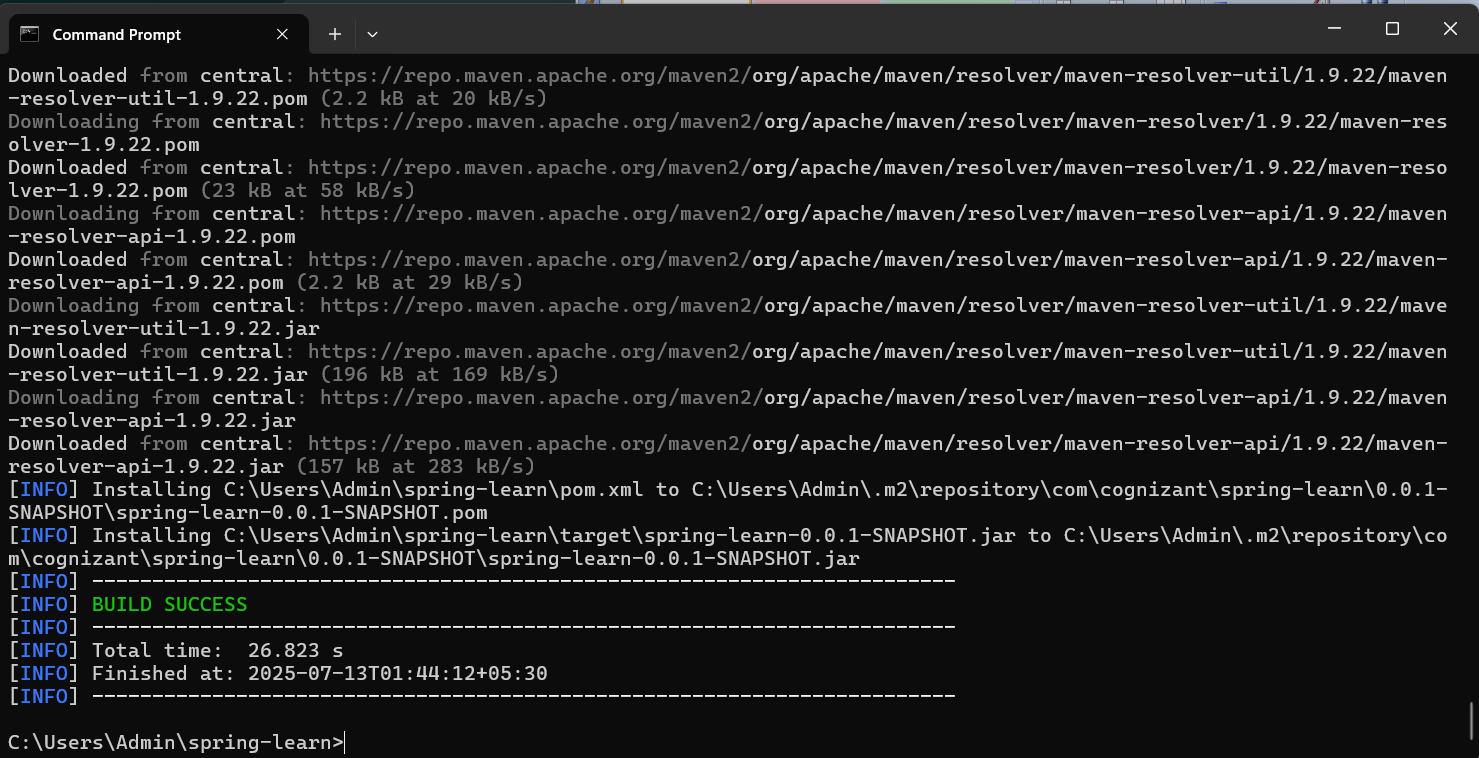
**Create a Spring Web Project using Maven**

**successfully completed the first major task**:

1. Project created using Spring Initializr
2. Maven configured and working
3. Java 21 installed and set
4. Project built with **mvn clean install**
5. JAR file created

**Console output:**

**C:\Users\Admin\spring-learn\target\spring-learn-0.0.1-SNAPSHOT.jar**

****

2: Spring Core — Load SimpleDateFormat from Spring XML

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

This is **Hands-On 4** from the document.

**1.Country.java**src/main/java/com/cognizant/spring\_learn/Country.java

package com.cognizant.spring\_learn;

public class Country {

private String code;

private String name;

public Country() {

System.out.println("Inside Country Constructor.");

}

public String getCode() {

return code;

}

public void setCode(String code) {

System.out.println("Setting code: " + code);

this.code = code;

}

public String getName() {

return name;

}

public void setName(String name) {

System.out.println("Setting name: " + name);

this.name = name;

}

@Override

public String toString() {

return "Country [code=" + code + ", name=" + name + "]";

}

}

2. country.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

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>

3. SpringLearnApplication.java  
src/main/java/com/cognizant/spring\_learn/SpringLearnApplication.java

package com.cognizant.spring\_learn;

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 {

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

displayCountry();

}

public static void displayCountry() {

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

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

System.out.println("Country: " + country.toString());

}

}

Console Output:

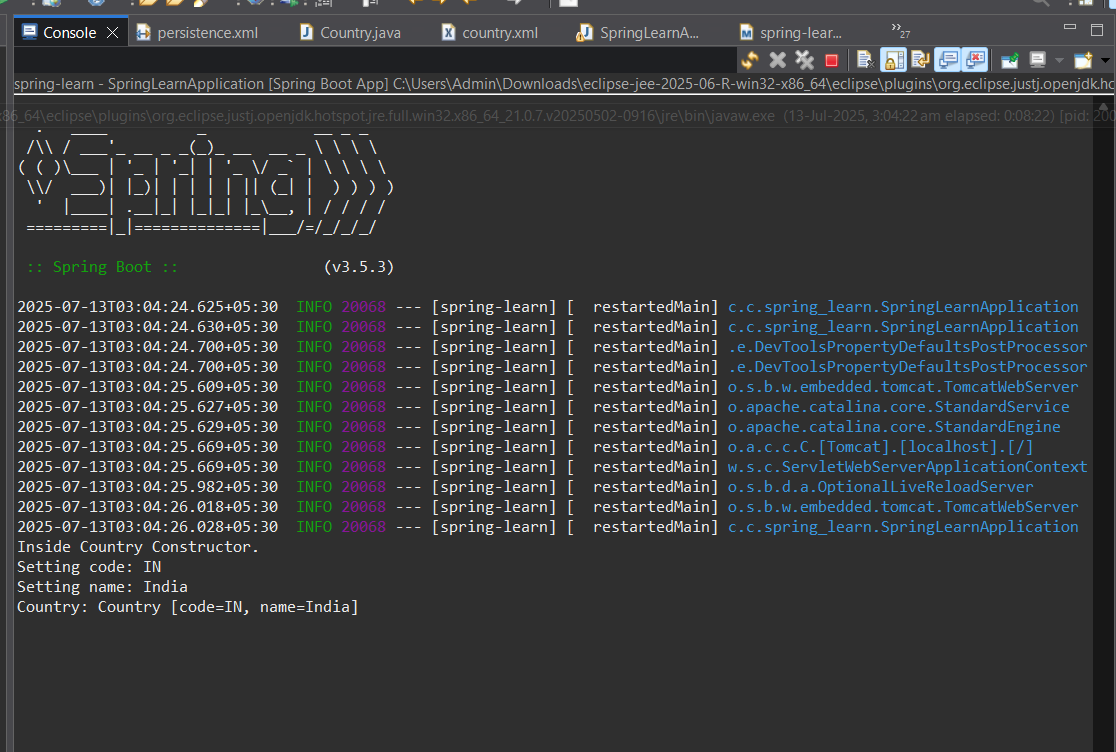
Inside Country Constructor.

Setting code: IN

Setting name: India

Country: Country [code=IN, name=India]

Sc:



**Hands-On 3: Hello World RESTful Web Service**

Responds with “Hello World!!”.

src/main/java/com/cognizant/spring\_learn/controller/HelloController.java  
  
package com.cognizant.spring\_learn.controller;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class HelloController {

@GetMapping("/hello")

public String sayHello() {

return "Hello World!!";

}

}

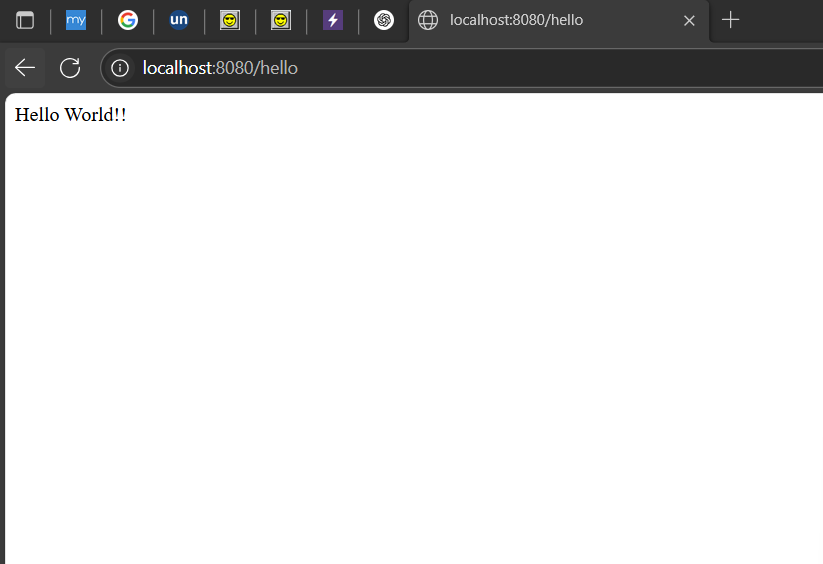
**Tomcat to start on port 8080**.

Test URL in Browser:  
<http://localhost:8080/hello>

Output in Browser:

Hello World!!

Sc:



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

**1. src/main/java/com/cognizant/spring\_learn/model/Country.java**

package com.cognizant.spring\_learn.model;

public class Country {

private String code;

private String name;

public Country() {

System.out.println("Inside Country Constructor.");

}

public String getCode() {

return code;

}

public void setCode(String code) {

System.out.println("Setting code: " + code);

this.code = code;

}

public String getName() {

return name;

}

public void setName(String name) {

System.out.println("Setting name: " + name);

this.name = name;

}

@Override

public String toString() {

return "Country [code=" + code + ", name=" + name + "]";

}

}

2. src/main/java/com/cognizant/spring\_learn/controller/CountryController.java

package com.cognizant.spring\_learn.controller;

import com.cognizant.spring\_learn.model.Country;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class CountryController {

@GetMapping("/country")

public Country getCountryIndia() {

Country country = new Country();

country.setCode("IN");

country.setName("India");

return country;

}

}

Test URL in Browser/Postman

<http://localhost:8080/country>

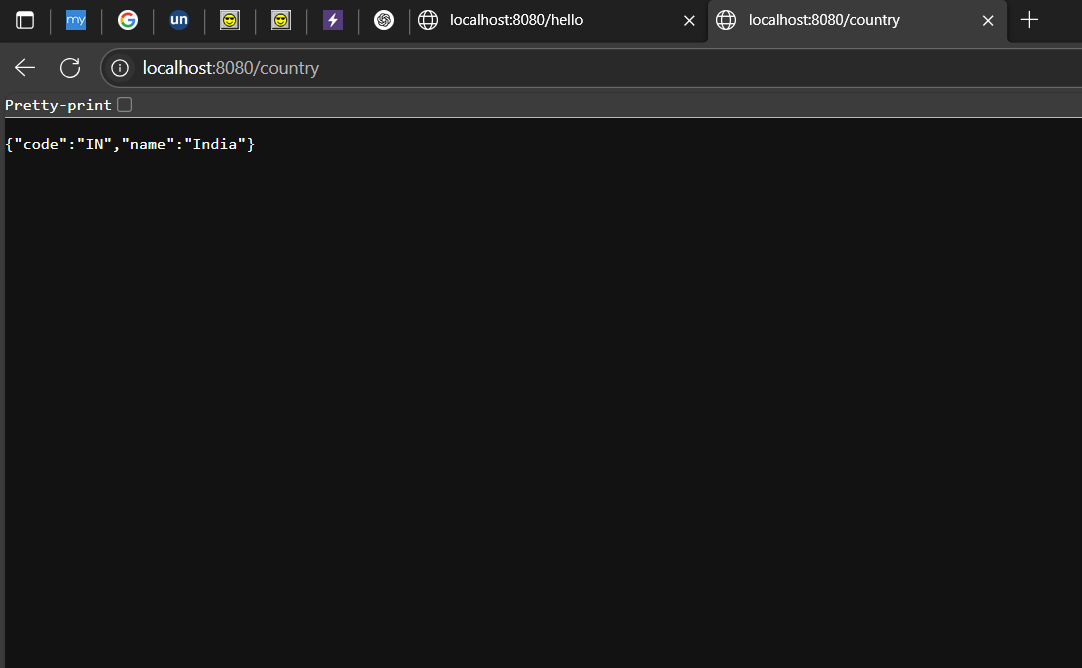
Expected Output (JSON):

{

"code": "IN",

"name": "India"

}  
  
Sc:



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

src/main/java/com/cognizant/spring\_learn/controller/CountryController.java

**@GetMapping("/countries/{code}")**

**public Country getCountryByCode(@PathVariable String code) {**

**Country country = new Country();**

**country.setCode(code.toUpperCase());**

**if (code.equalsIgnoreCase("in")) {**

**country.setName("India");**

**} else if (code.equalsIgnoreCase("us")) {**

**country.setName("United States");**

**} else if (code.equalsIgnoreCase("jp")) {**

**country.setName("Japan");**

**} else {**

**country.setName("Unknown");**

**}**

**return country;**

**}**

1. **Class:** CountryController.java
2. **Package:** com.cognizant.spring\_learn.controller
3. **URL Tested:** <http://localhost:8080/countries/in>
4. **Output Sample:**

{

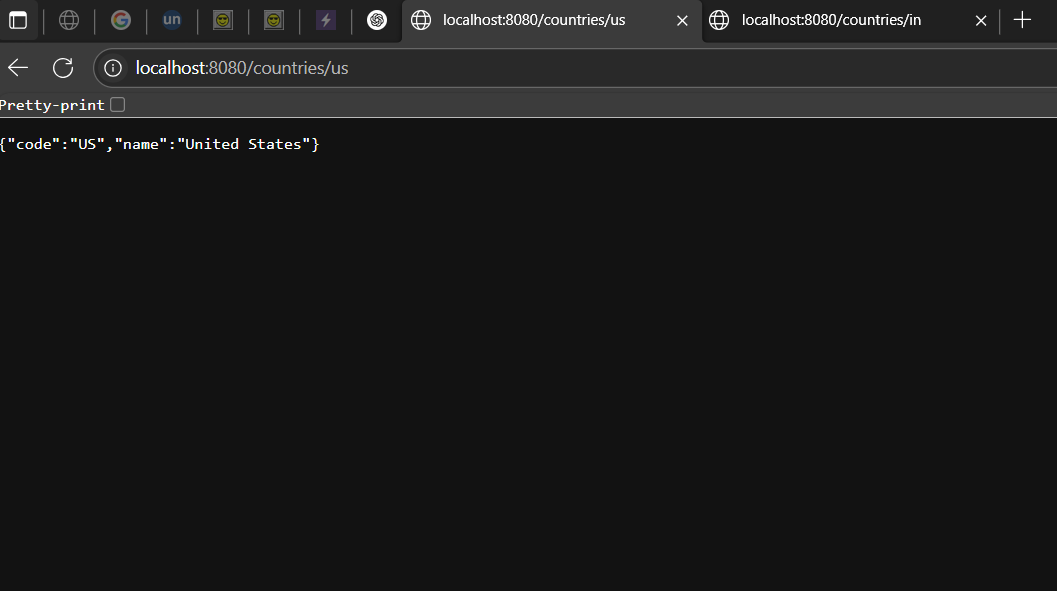
"code": "IN",

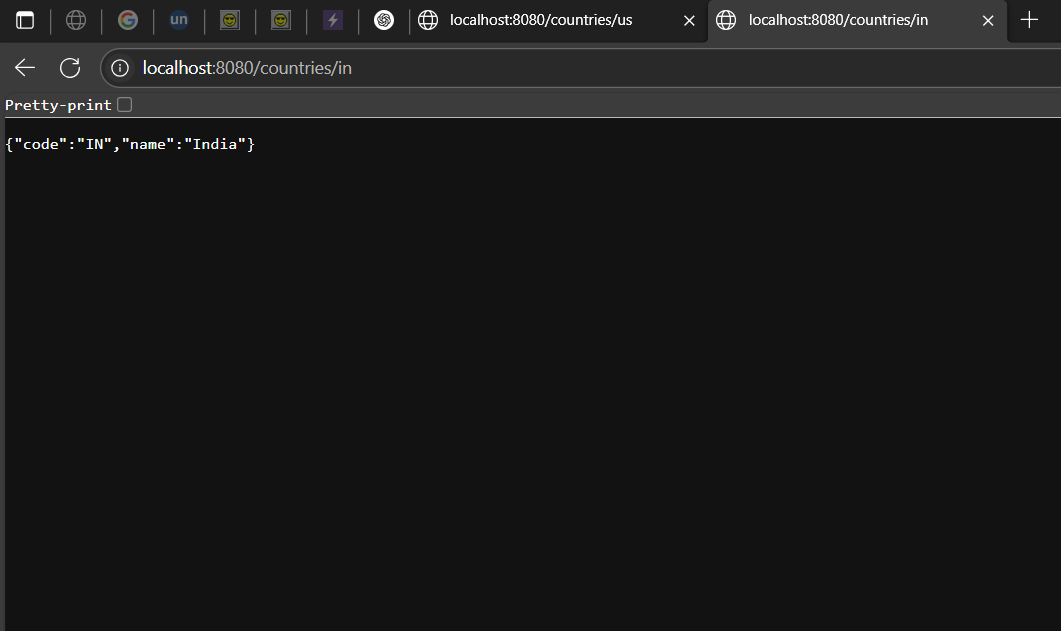
"name": "India"

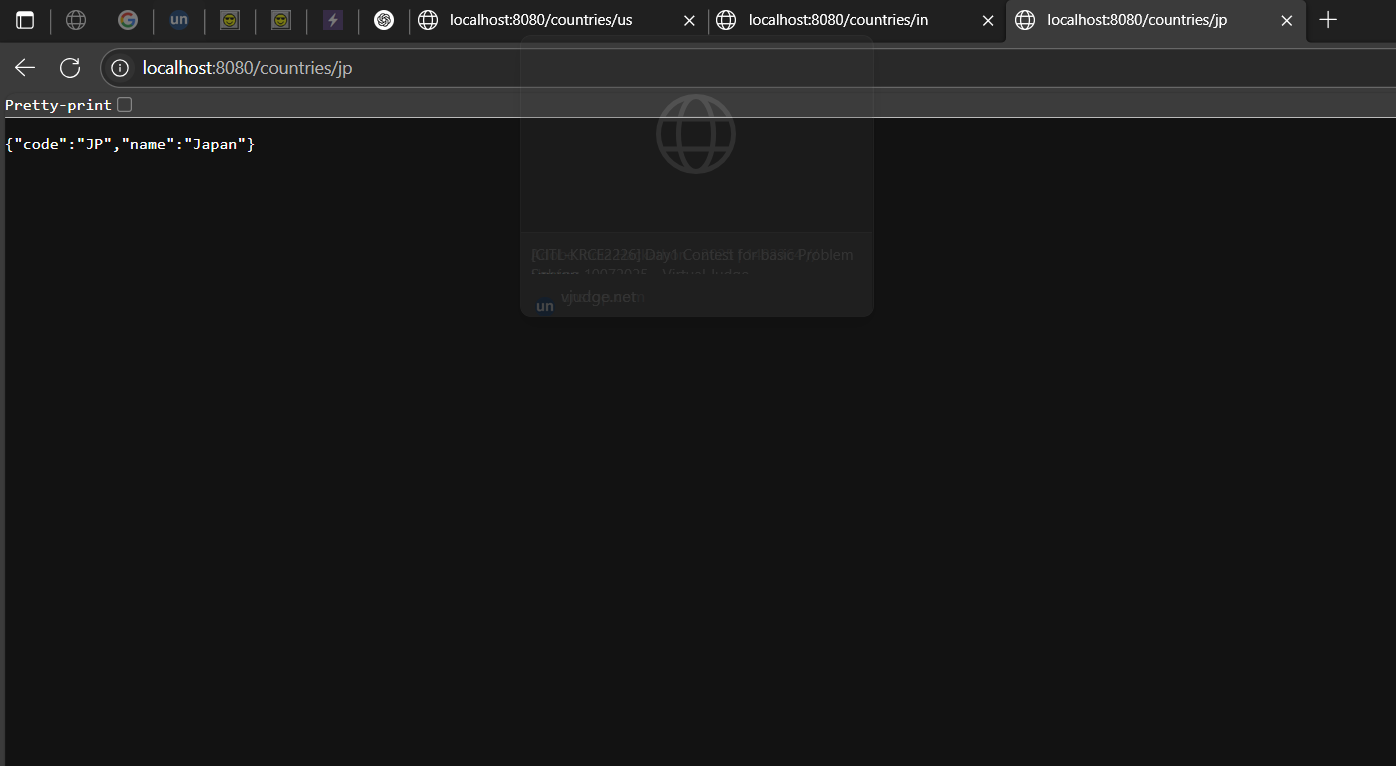
}

1. http://localhost:8080/countries/in → shows India
2. http://localhost:8080/countries/us → shows United States
3. http://localhost:8080/countries/jp → shows Japan
4. http://localhost:8080/countries/xyz → shows Unknown

**Sc:**

****

****

****

**6.Hands on "Create Authentication Service that returns JWT"** hands-on task step by step in Eclipse

REST endpoint: /authenticate  
Method: POST  
JWT token successfully generated  
Spring Security + JWT configuration working

1. **Files Created:**
2. **JwtAuthenticationController.java – REST Controller for authentication**
3. **JwtRequest.java – Model for username/password**
4. **JwtResponse.java – Response model with JWT token**
5. **JwtTokenUtil.java – Token utility generator**
6. **WebSecurityConfig.java – Spring security configuration**

**Postman using POST at** [**http://localhost:8080/authenticate**](http://localhost:8080/authenticate)

Sent:

{

"username": "admin",

"password": "admin123"

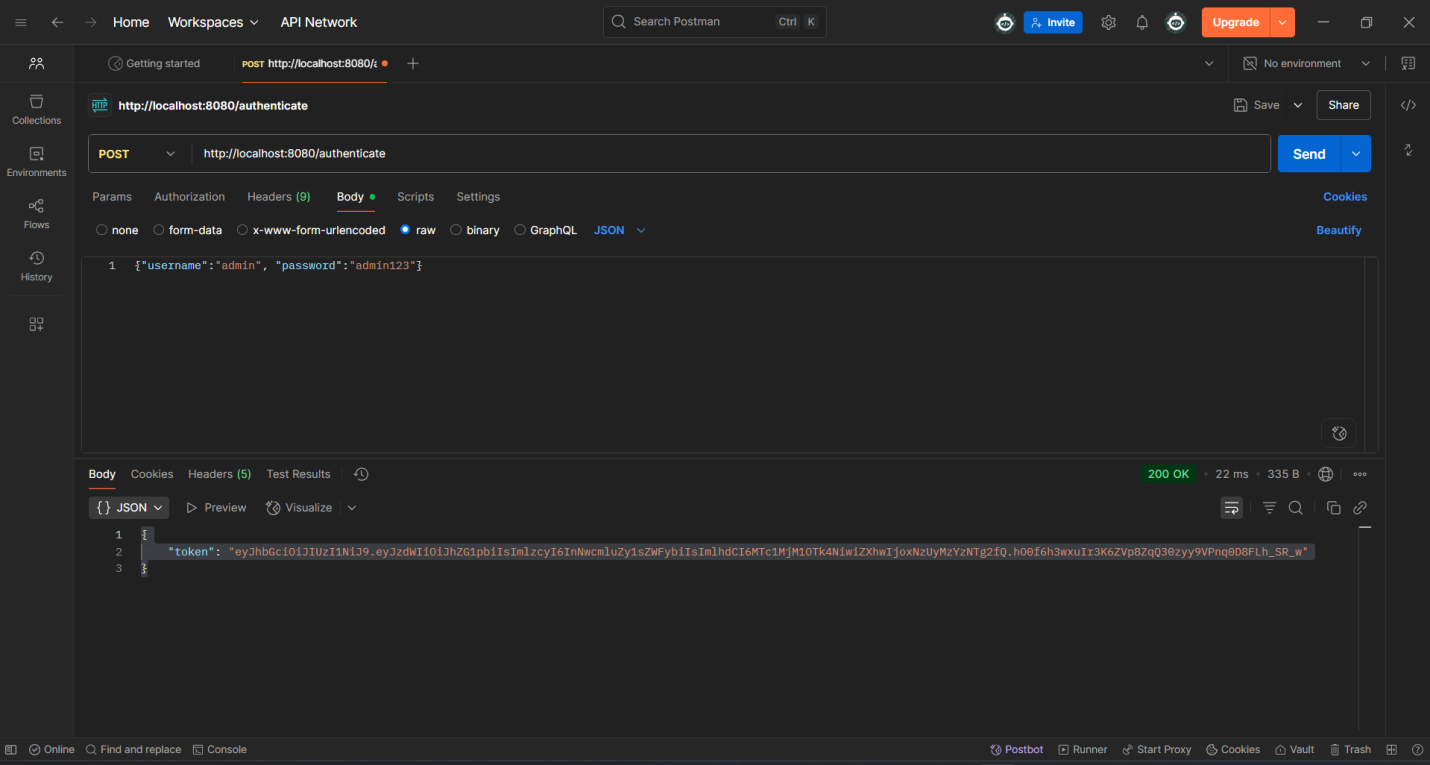
}

Got Response:

{

"token": "<JWT-TOKEN>"

**}**

**Sc:  
**

**THANK YOU**