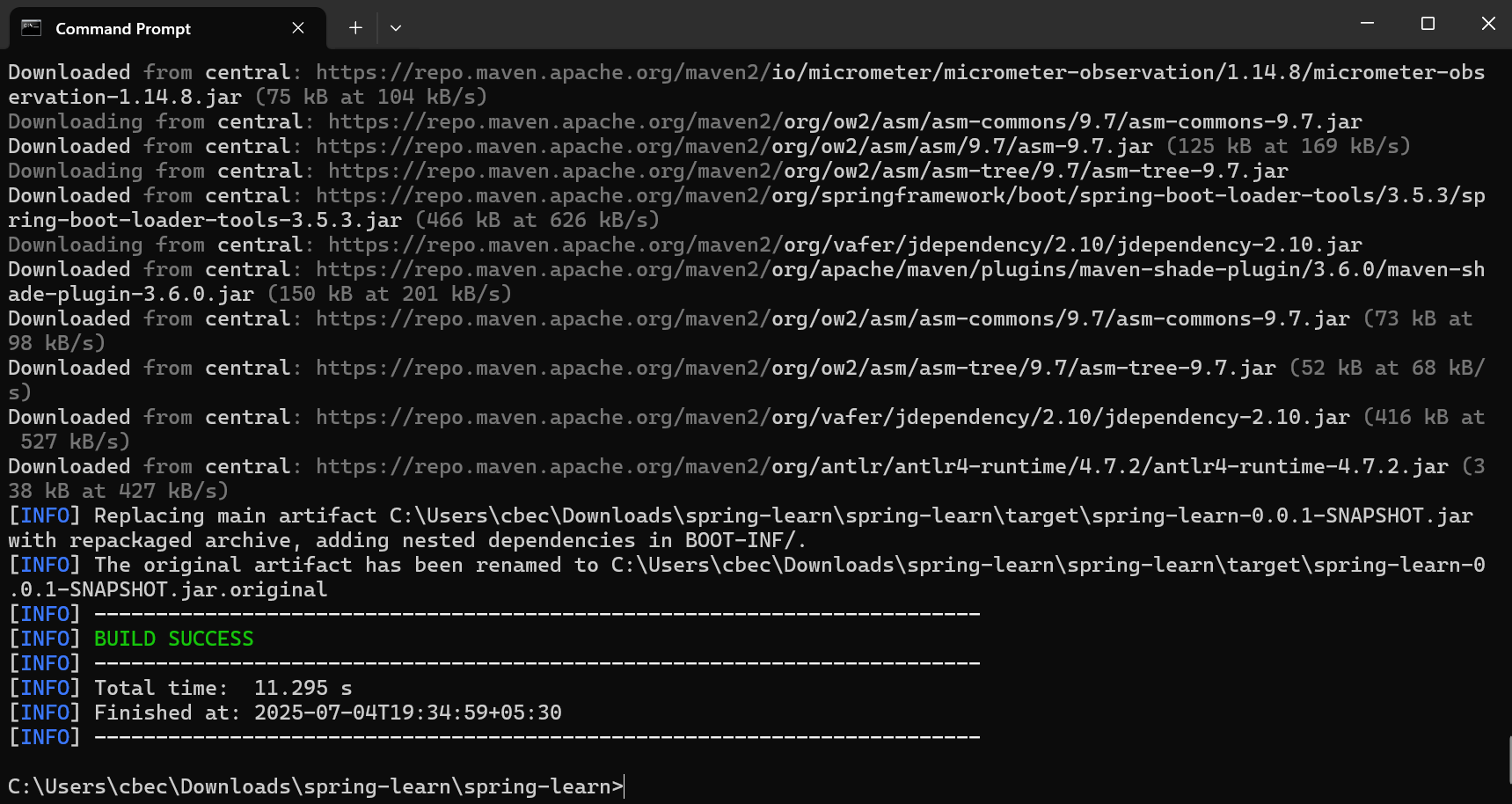
Spring REST

1) Create a Spring Web Project using Maven 

To create Spring web project using Maven first we have to go to Spring Initializr through the link : <https://start.spring.io> . Then under Project we choose Maven because it is told to build using Maven. Language will be Java. Under the Project Metadata part as mentioned in the assignment, we put the Group as com.cognizant, the Artifact as spring-learn, the Name as spring-learn. The package name will be com.cognizant.springlearn. Then adding the dependencies Spring Boot DevTools and Spring Web. Then click on generate to download the zip folder containing the project. Extacted the zip folder, then opened Elipse IDE and then File->Import->Existing maven project->Browse and find the correct folder then clicked on Finish. Then in command prompt ran the command : mvn clean package

In the command prompt the following was shown:  


This will build the project. Inside Eclipse IDE in SpringLearnApplication.java file present in src/main/java under com.cognizant.springlearn package added a printing statement.

**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) {

SpringApplication.*run*(SpringLearnApplication.**class**, args);

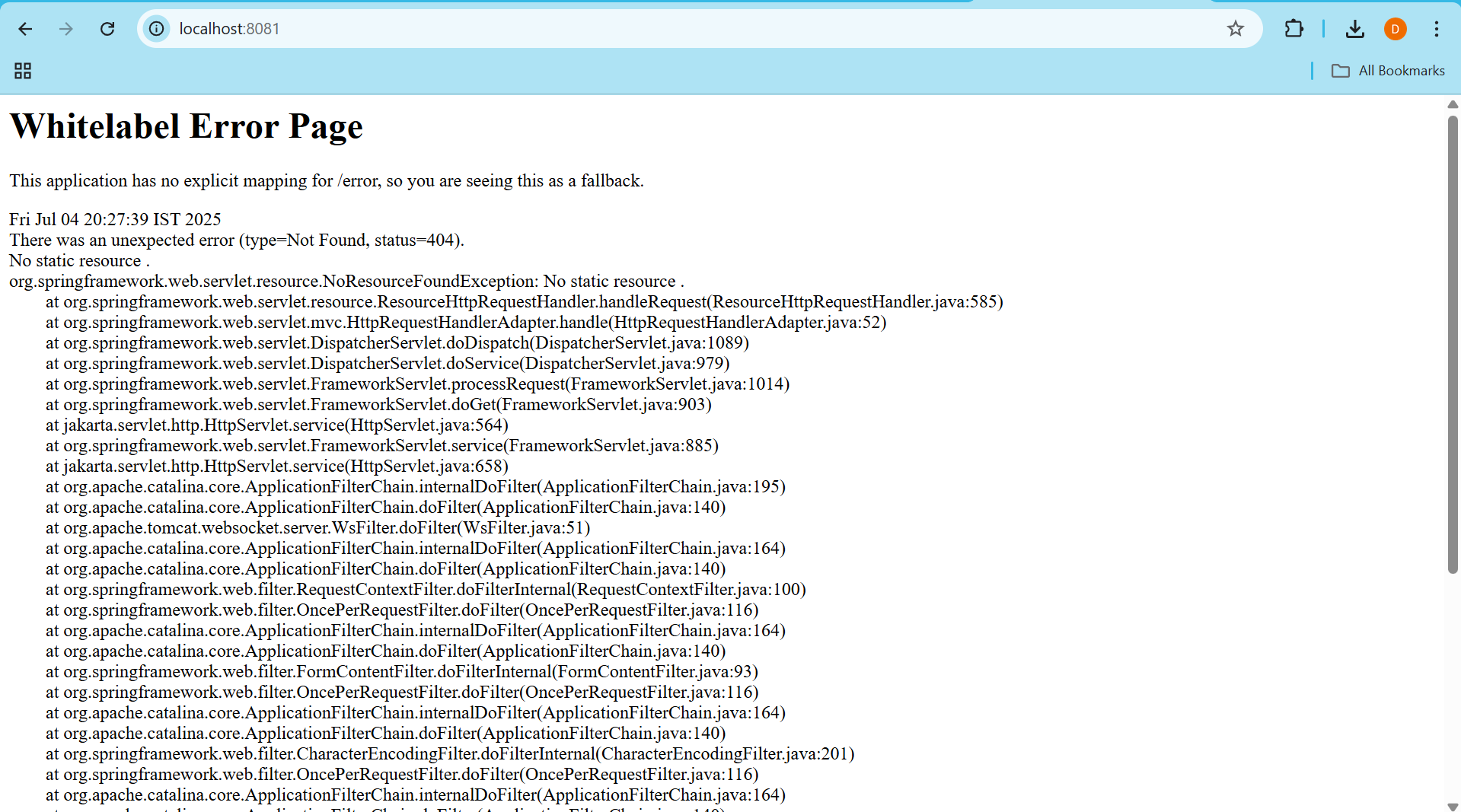
System.***out***.println("SpringLearnApplication started successfully!");

}

}

After this run this file and following output was generated.  
  
Output:



If the server is checked at port 8081 the following is being shown as nothing is mapped to route “/”.

2) Spring Core – Load Country from Spring Configuration XML

Inside com.cognizant.sprinlearn package of src/main/java folder created a class Country.java

**package** com.cognizant.springlearn;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**public** **class** Country {

**private** **static** **final** Logger ***LOGGER***= LoggerFactory.*getLogger*(Country.**class**);

**private** String code;

**private** String name;

**public** Country() {

***LOGGER***.debug("Inside Country Constructor");

}

**public** String getCode() {

***LOGGER***.debug("Inside getCode");

**return** code;

}

**public** **void** setCode(String code) {

***LOGGER***.debug("Inside setCode");

**this**.code= code;

}

**public** String getName() {

***LOGGER***.debug("Inside getName");

**return** name;

}

**public** **void** setName(String name) {

***LOGGER***.debug("Inside setName");

**this**.name= name;

}

@Override

**public** String toString() {

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

}

}

After that, inside src/main/resources, created country.xml

<?xmlversion="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

http://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="country" class="com.cognizant.springlearn.Country">

<property name="code" value="IN"/>

<property name="name" value="India"/>

</bean>

</beans>

Then modified the SpringLearnApplication.java main class.

**package** com.cognizant.springlearn;

**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) {

SpringApplication.*run*(SpringLearnApplication.**class**, args);

System.***out***.println("SpringLearnApplication started successfully!");

***LOGGER***.info("Start");

*displayCountry*();

***LOGGER***.info("End");

}

**public** **static** **void** displayCountry() {

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

Country country=context.getBean("country",Country.**class**);

***LOGGER***.debug("Country : {}", country.toString());

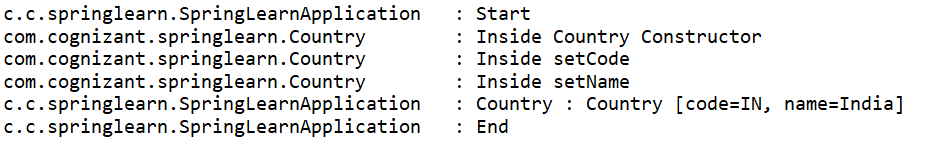
}

}

Added the displayCountry method and did the LOGGER.info(”Start”) and LOGGER.info(”End”).

Also inside application.properties added the line logging.level.com.cognizant=DEBUG.

Then ran the SpringLearnApplication.java as a Java Application.  
  
Output:



3) Hello World RESTful Web Service

In the spring-learn project created earlier, added a controller package com.cognizant.springlearn.controller and inside that package created a controller HelloController.java 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**);

//Get the Hello function

@GetMapping("/hello")

**public** String sayHello() {

***LOGGER***.info("START sayHello()");

String response="Hello World!!";

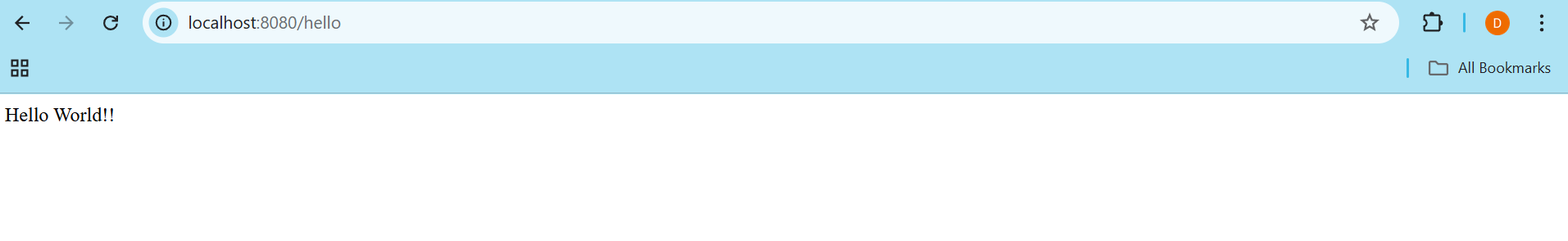
***LOGGER***.info("END sayHello()");

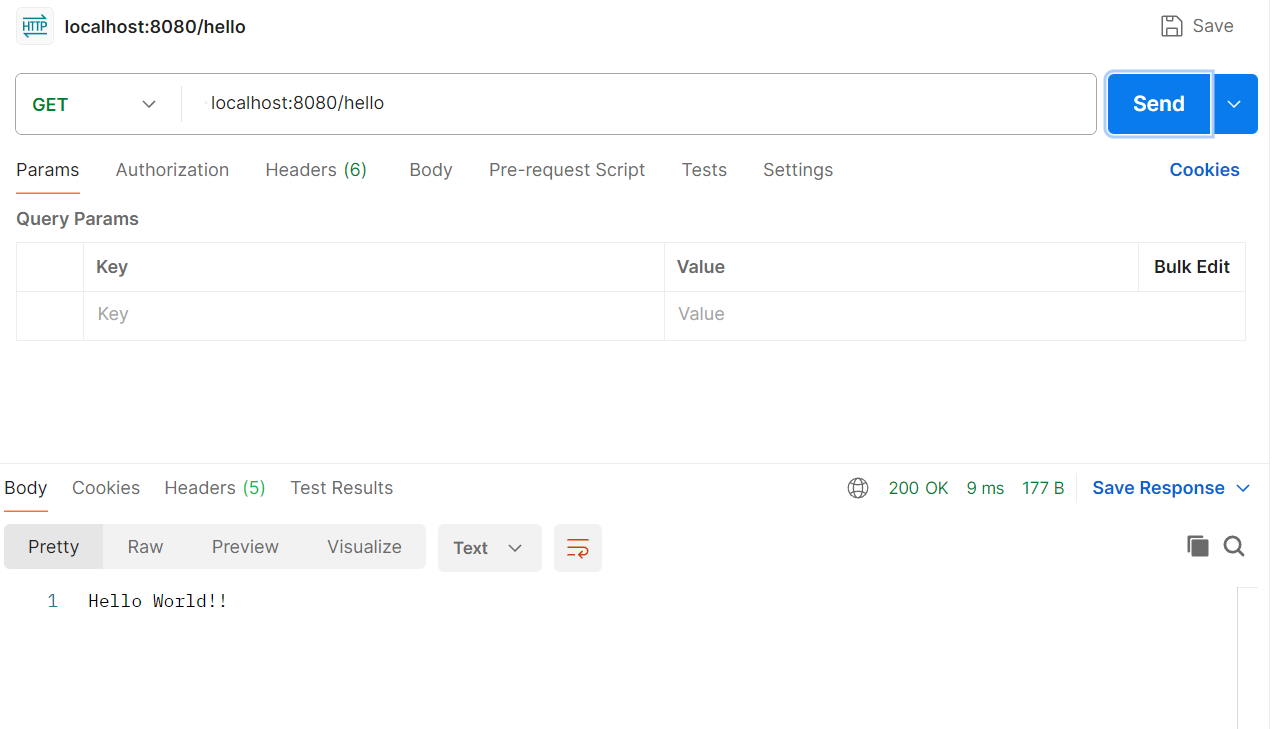
**return** response;

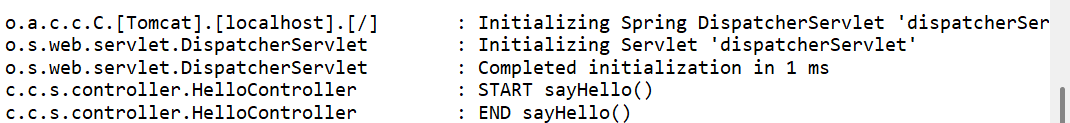
}

}

After that ran the SpringLearnApplication.java main class as a Java application. The tomcat server started running at default port 8080.

On typing http://localhost:8080/hello, on the web browser the following came on the screen:

On typing the same url in postman and sending the request with GET method, the following was observed:

The following was logged on the console:

4) REST - Country Web Service

Inside com.cognizant.springlearn.controller package, created a CountryController.java class, it is the controller class for the service.

**package** com.cognizant.springlearn.controller;

**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.RestController;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** com.cognizant.springlearn.Country;

@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("country",Country.**class**);

***LOGGER***.debug("Country: {}", country);

***LOGGER***.info("End getCountryIndia()");

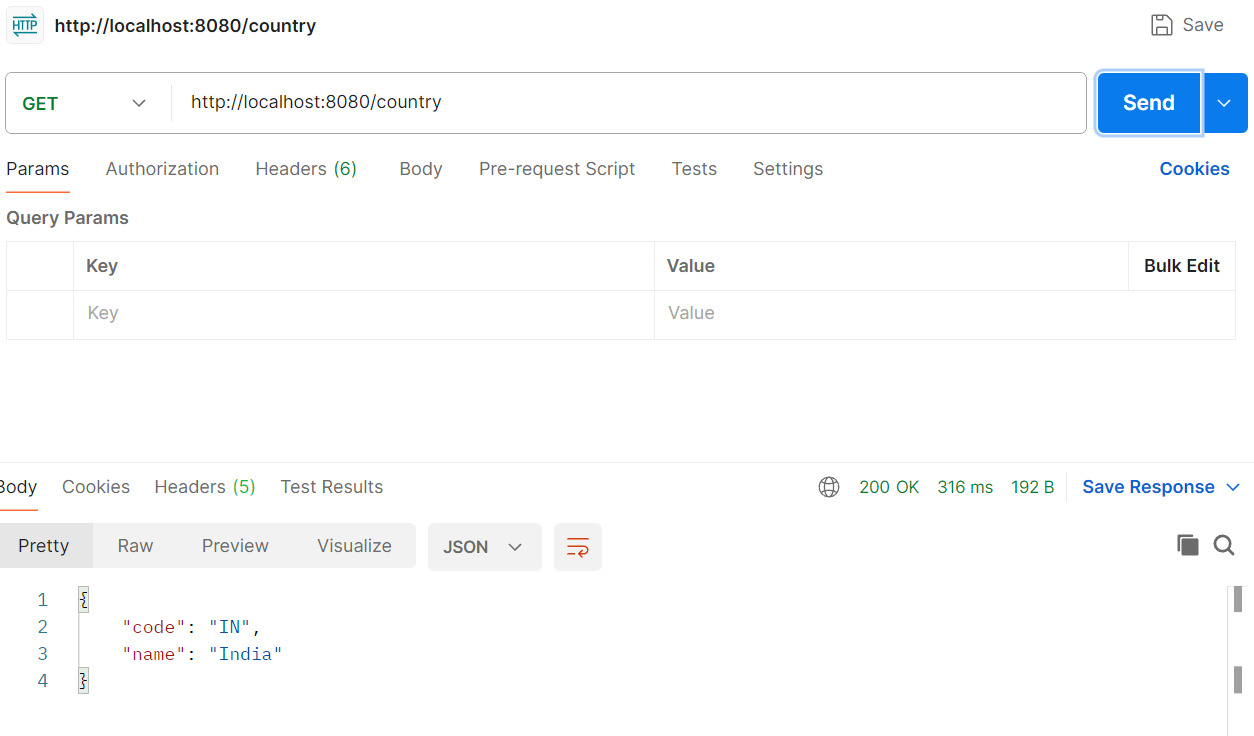
**return** country;

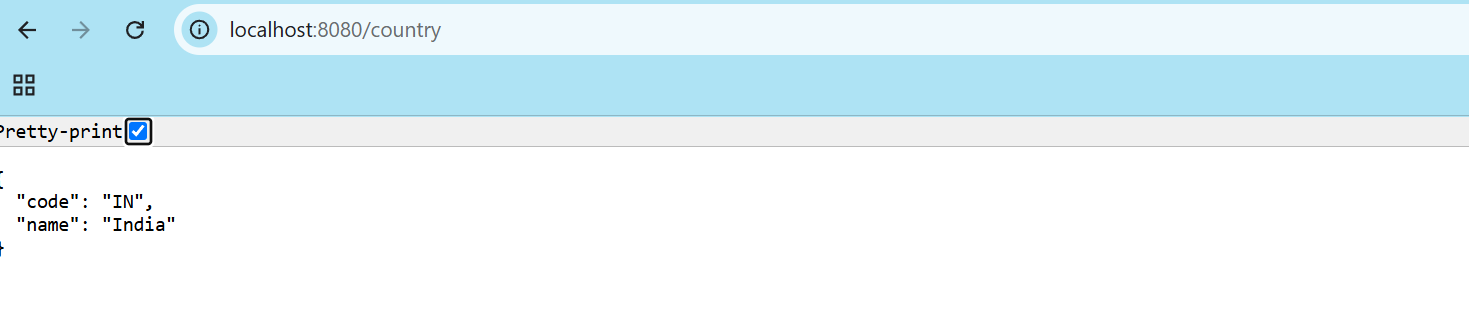
}

}

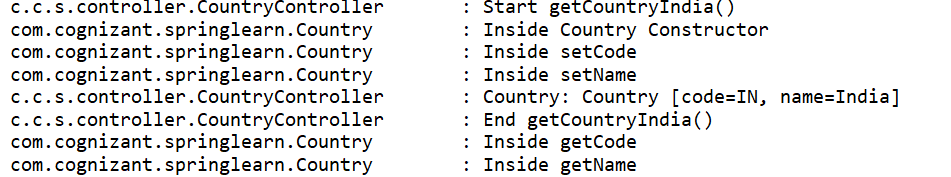
The remaining code remains the same as mentioned previously in Spring Core - Load Country from Spring Configuration Xml.

Now running the SpringLearnApplication.java file as Java application, the tomcat server starts at port 8080. On typing http://localhost:8080/country, the following came on postman and browser respectively.

Output:



In the console the following was logged:



5) REST - Get all countries

Inside src/main/resources created another xml file, countries.xml

<?xmlversion="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:util="http://www.springframework.org/schema/util"

xsi:schemaLocation="

http://www.springframework.org/schema/beans

https://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/util

https://www.springframework.org/schema/util/spring-util.xsd">

<util:list id="countryList">

<bean class="com.cognizant.springlearn.Country">

<property name="code" value="IN"/>

<property name="name" value="India"/>

</bean>

<bean class="com.cognizant.springlearn.Country">

<property name="code" value="US"/>

<property name="name" value="United States"/>

</bean>

<bean class="com.cognizant.springlearn.Country">

<property name="code" value="JP"/>

<property name="name" value="Japan"/>

</bean>

<bean class="com.cognizant.springlearn.Country">

<property name="code" value="DE"/>

<property name="name" value="Germany"/>

</bean>

<bean class="com.cognizant.springlearn.Country">

<property name="code" value="EN"/>

<property name="name" value="England"/>

</bean>

<bean class="com.cognizant.springlearn.Country">

<property name="code" value="FR"/>

<property name="name" value="France"/>

</bean>

<bean class="com.cognizant.springlearn.Country">

<property name="code" value="CN"/>

<property name="name" value="China"/>

</bean>

<bean class="com.cognizant.springlearn.Country">

<property name="code" value="AU"/>

<property name="name" value="Australia"/>

</bean>

<bean class="com.cognizant.springlearn.Country">

<property name="code" value="BR"/>

<property name="name" value="Brazil"/>

</bean>

<bean class="com.cognizant.springlearn.Country">

<property name="code" value="MX"/>

<property name="name" value="Mexico"/>

</bean>

</util:list>

</beans> 

Inside the CountryController added a method with GetMapping and added this code to the already existing controller class.

@GetMapping("/countries")

**public** List<Country> getAllCountries() {

ApplicationContext context= **new** ClassPathXmlApplicationContext("countries.xml");

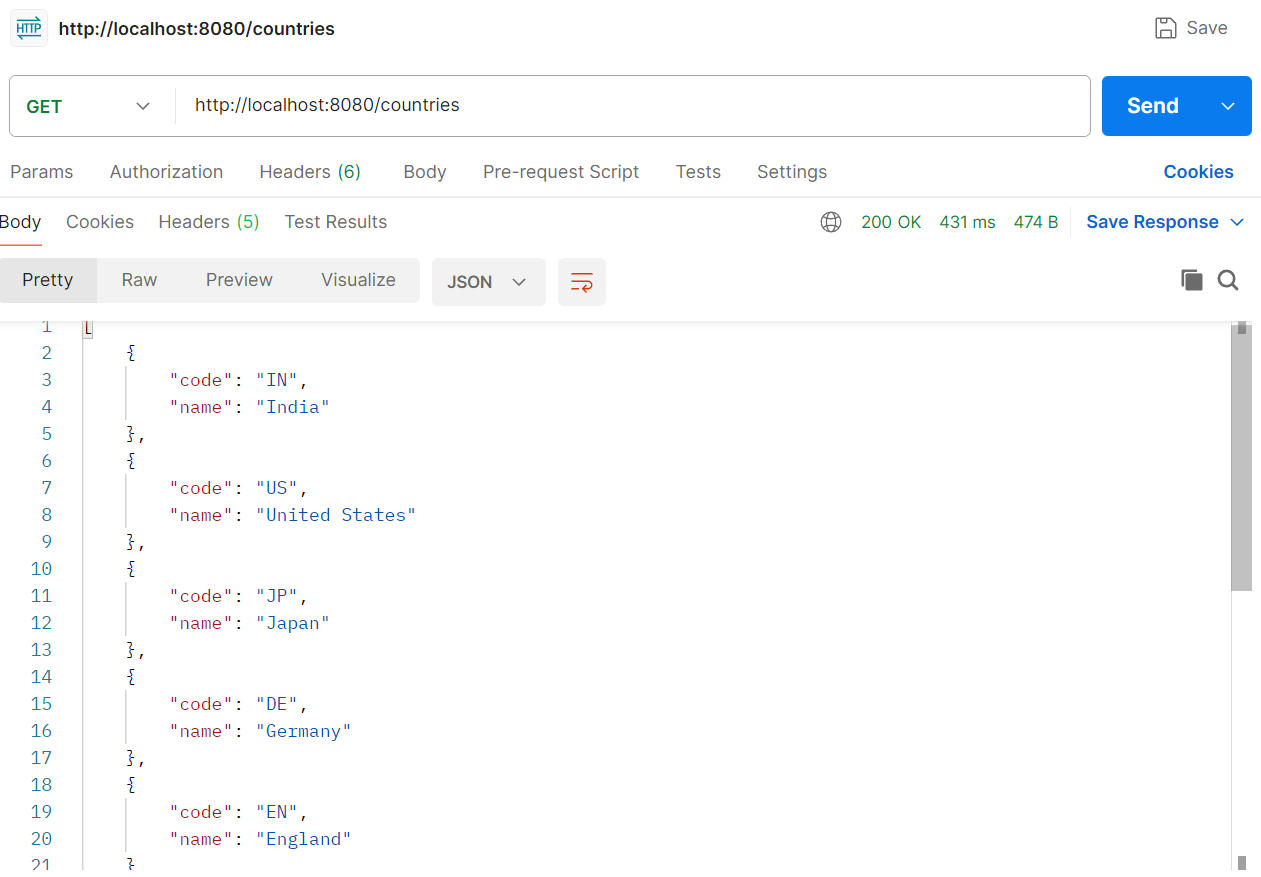
List<Country> countries= (List<Country>) context.getBean("countryList");

**return** countries;

}

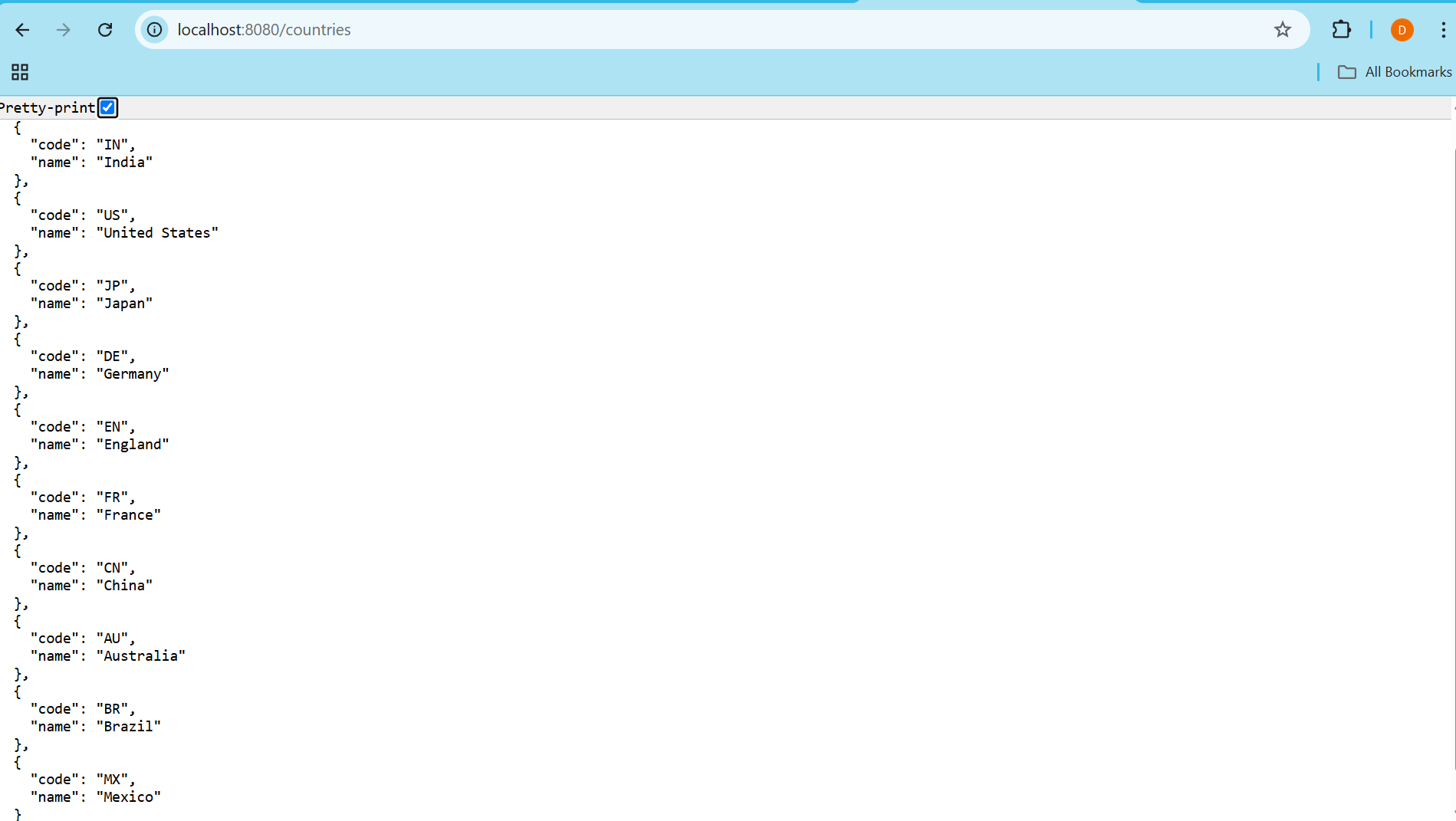
Then ran the SpringLearnApplication.java class as a Java application. On typing, http://localhost:8080/countries, in postman and browser the following output was obtained:

Output:





In the browser, the following was obtained:



6) REST - Get country based on country code

Created a service package inside com.cognizant.springlearn package. Inside it created a CountryService interface and a CountryServiceImpl implementation class.

Code:

Interface:

**package** com.cognizant.springlearn.service;

**import** com.cognizant.springlearn.Country;

**public** **interface** CountryService {

Country getCountry(String code);

}

Concrete class:  
**package** com.cognizant.springlearn.service;

**import** java.util.List;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**import** org.springframework.stereotype.Service;

**import** org.springframework.context.ApplicationContext;

**import** com.cognizant.springlearn.Country;

@Service

**public** **class** CountryServiceImpl **implements** CountryService{

@Override

**public** Country getCountry(String code) {

ApplicationContext context= **new** ClassPathXmlApplicationContext("countries.xml");

List<Country> countries= context.getBean("countryList", List.**class**);

**return** countries.stream()

.filter(country-> country.getCode().equalsIgnoreCase(code))

.findFirst()

.orElse(**null**);

}

}

Then in CountryController class added a new controller with GetMapping:  
  
@GetMapping("/countries/{code}")

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

**return** countryService.getCountry(code);

}

Inside the CountryController class added the autowired object of interface CountryService  
  
@Autowired

**private** CountryService countryService;

The following dependencies had to be imported to the CountryController class

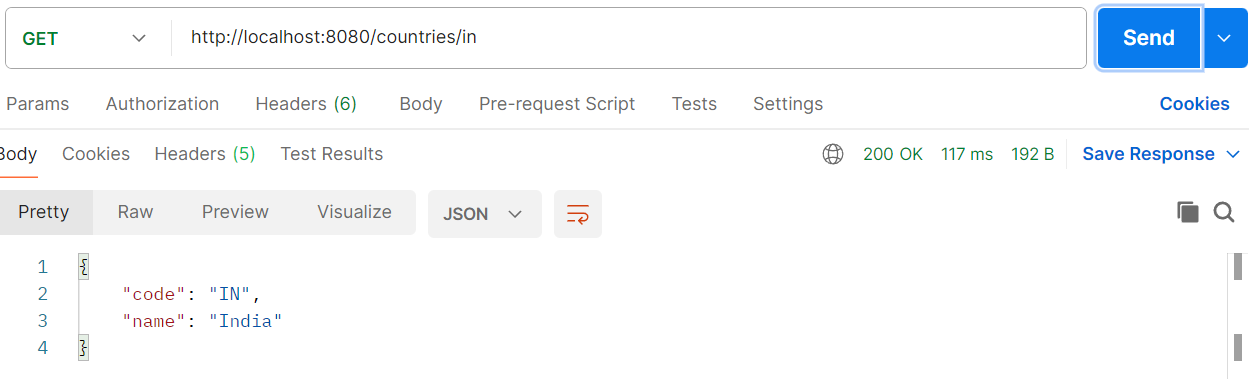
**import** org.springframework.web.bind.annotation.PathVariable;

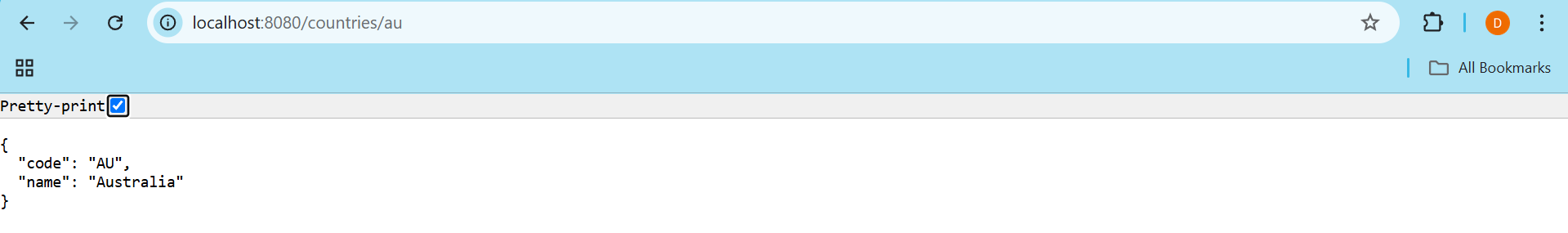
**import** org.springframework.beans.factory.annotation.Autowired;

The GetMapping dependency was already imported for previous assignment.

On running the SpringLearnApplication class as a Java application the following was observed on postman and the browser.

Output:





7) Create authentication service that returns JWT

Firstly, created a spring Boot project using Spring Boot Initializr, added dependencies Spring Web and Spring Security. After that downloaded the zip folder, imported the project and opened it in Eclipse IDE.

Inside pom.xml added a few more jwt dependencies. Added the following in pom.xml:

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-api</artifactId>

<version>0.11.5</version>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-impl</artifactId>

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-jackson</artifactId>

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

Then created 3 packages inside the base package com.example.jwt, the three packages are com.example.jwt.config, com.example.jwt.controller and com.example.jwt.util

Inside config package created SecurityConfig.java class.

**package** com.example.jwt.config;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

**import** org.springframework.security.config.annotation.web.builders.HttpSecurity;

**import** org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

**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;

@Configuration

@EnableWebSecurity

**public** **class** SecurityConfig {

@Bean

**public** UserDetailsService userDetailsService() {

UserDetails user= User.*withUsername*("user")

.password("{noop}pwd")

.roles("USER")

.build();

**return** **new** InMemoryUserDetailsManager(user);

}

@Bean

**public** SecurityFilterChain filterChain(HttpSecurity http) **throws**Exception {

http.csrf().disable()

.authorizeHttpRequests()

.requestMatchers("/authenticate").permitAll()

.anyRequest().authenticated()

.and()

.httpBasic();

**return** http.build();

}

}

Then inside com.example.jwt.controller package created AuthController class.

**package** com.example.jwt.controller;

**import** java.nio.charset.StandardCharsets;

**import** java.util.Base64;

**import** java.util.Collections;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

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

**import** org.springframework.web.bind.annotation.RequestHeader;

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

**import** com.example.jwt.util.JwtUtil;

@RestController

**public** **class**AuthController {

@Autowired

**private** JwtUtil jwtUtil;

@GetMapping("/authenticate")

**public** ResponseEntity<?> authenticate(@RequestHeader("Authorization") String authHeader){

String base64Credentials= authHeader.substring("Basic ".length());

**byte**[] credDecoded= Base64.*getDecoder*().decode(base64Credentials);

String credentials= **new**String(credDecoded,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");

}

}

}

Lastly inside util folder created JwtUtil class.

**package** com.example.jwt.util;

**import** java.sql.Date;

**import** javax.crypto.SecretKey;

**import** org.springframework.stereotype.Component;

**import** io.jsonwebtoken.Jwts;

**import** io.jsonwebtoken.SignatureAlgorithm;

**import** io.jsonwebtoken.security.Keys;

@Component

**public** **class** JwtUtil {

**private** **final** SecretKey SECRET\_KEY= Keys.*secretKeyFor*(SignatureAlgorithm.***HS256***);

**public** String generateToken(String username) {

**return** Jwts.*builder*()

.setSubject(username)

.setIssuedAt(**new**Date(System.*currentTimeMillis*()))

.setExpiration(**new**Date(System.*currentTimeMillis*()+ 1000\*60\*10))

.signWith(SECRET\_KEY,SignatureAlgorithm.***HS256***)

.compact();

}

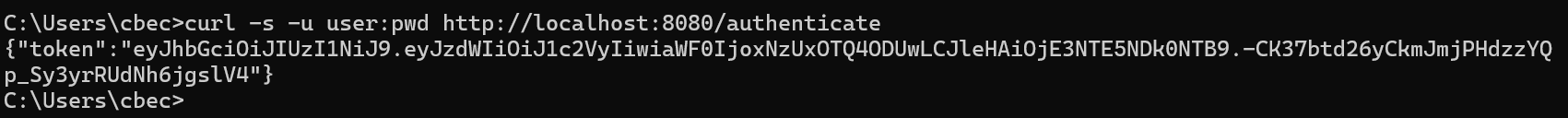
}

Then ran the main java class containing the main method. Then in command prompt wrote the following command:

curl -u user:pwd http://localhost:8080/authenticate

The following output was obtained:

Output:



Also on testing the route in postman, the following output came with a response code of 200 OK.

