**WEEK 4**

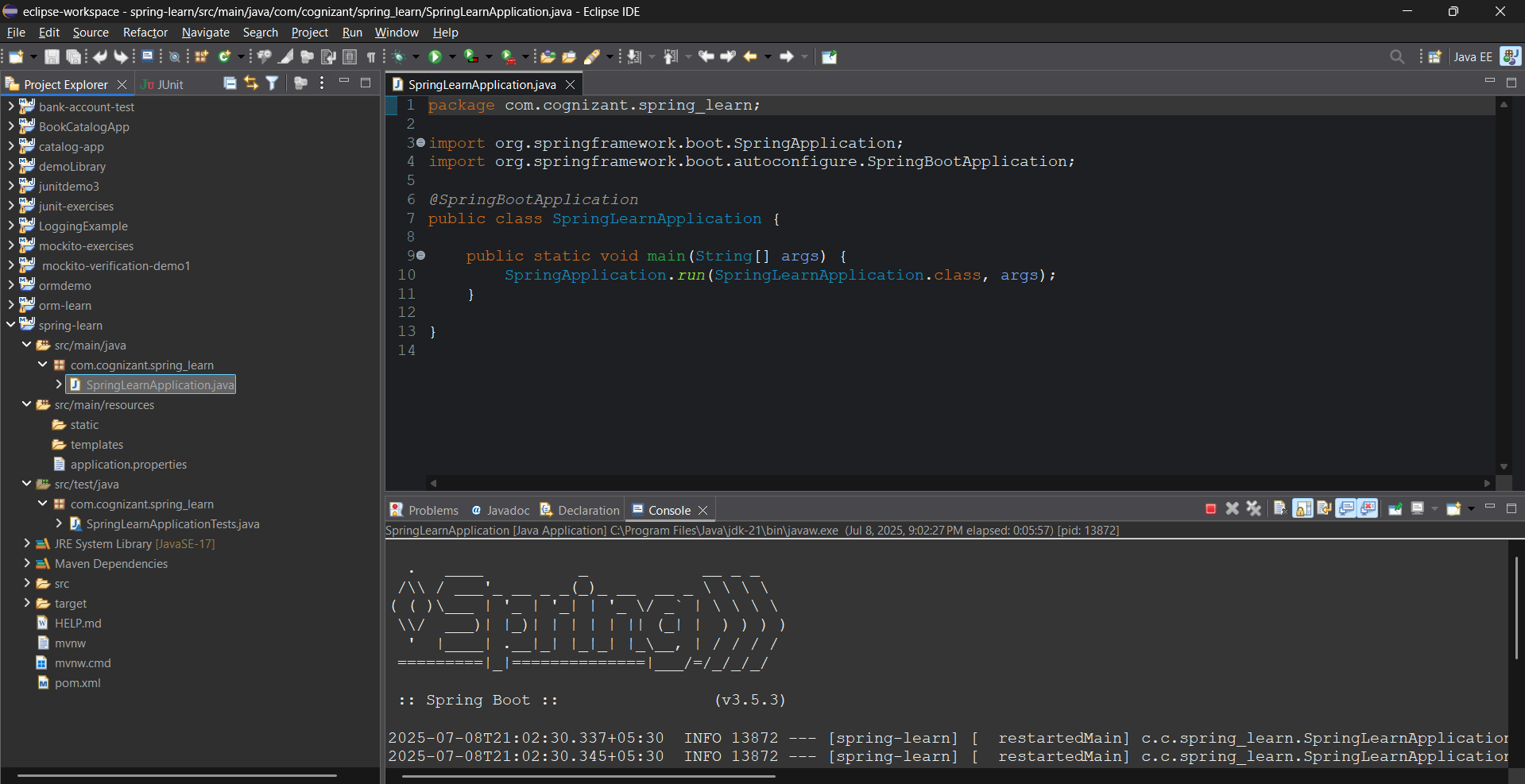
**SPRING REST USING SPRING BOOT 3**

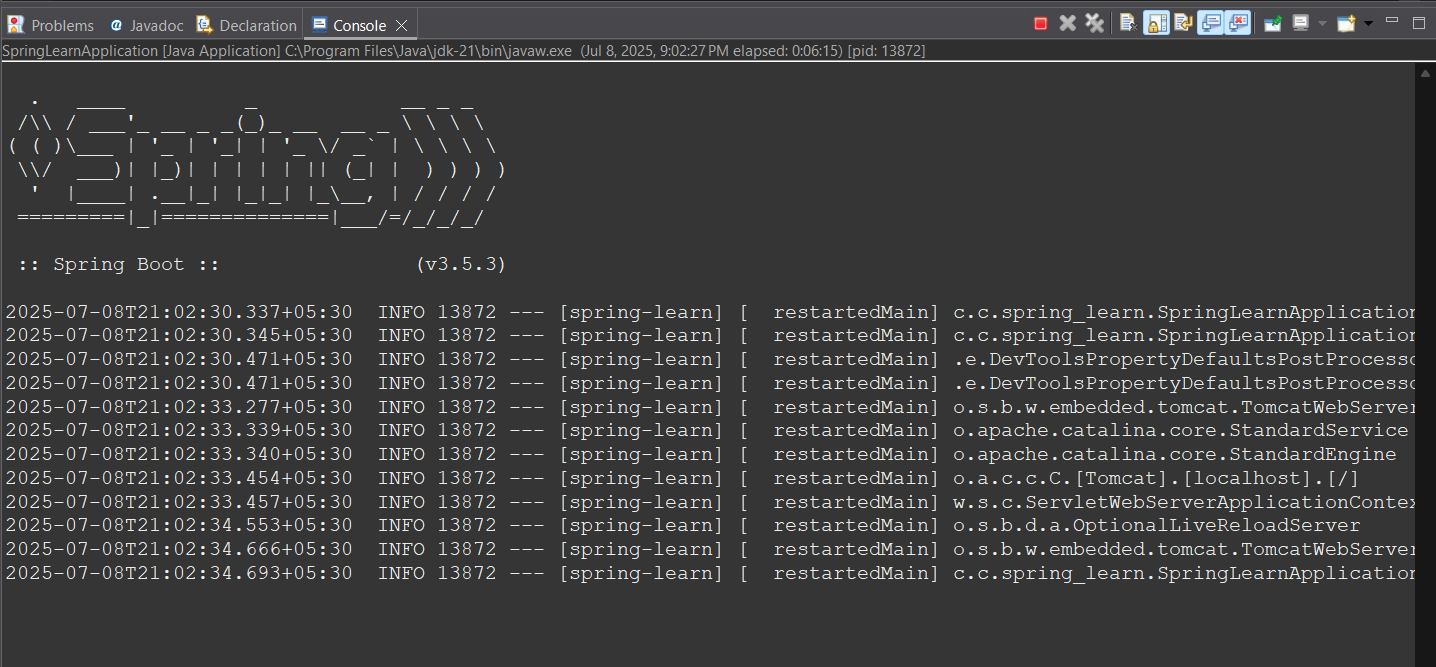
**EXERCISE 1:**

**Create a Spring Web Project using Maven:**

**EXPLANATION:**

I implemented a Spring Web project using Maven via Spring Initializr, added required dependencies, and imported it into Eclipse.  
I ran the SpringLearnApplication class to verify successful setup and application startup using Spring Boot.

**OUTPUT:** ****



**EXERCISE 2:**

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

**EXPLANATION:**

I created a Country class with code and name properties and configured it as a Spring bean in country.xml.  
I loaded the XML using ClassPathXmlApplicationContext in the displayCountry() method of SpringLearnApplication.  
I used SLF4J logging to confirm that the Country bean was created and its values were loaded successfully.

**CODE:**

**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("Getting country code.");

return code;

}

public void setCode(String code) {

LOGGER.debug("Setting country code.");

this.code = code;

}

public String getName() {

LOGGER.debug("Getting country name.");

return name;

}

public void setName(String name) {

LOGGER.debug("Setting country name.");

this.name = name;

}

@Override

public String toString() {

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

}

}

**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.springlearn.Country">

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

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

</bean>

</beans>  
  
**SpringLearnApplication.java:**

package com.cognizant.springlearn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

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

public static void main(String[] args) {

LOGGER.info("START");

SpringApplication.run(SpringLearnApplication.class, args);

displayCountry();

LOGGER.info("END");

}

public static void displayCountry() {

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

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

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

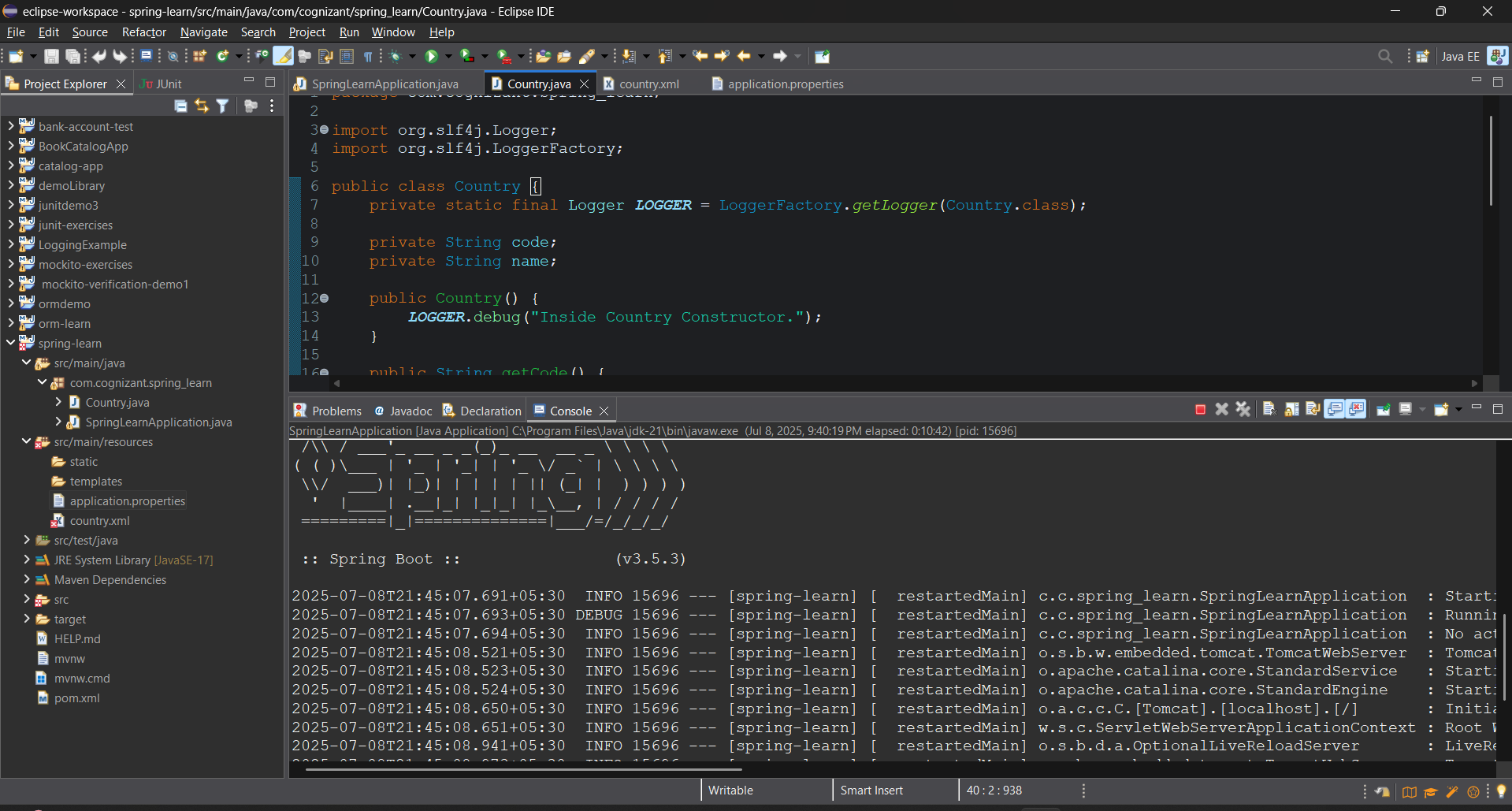
}

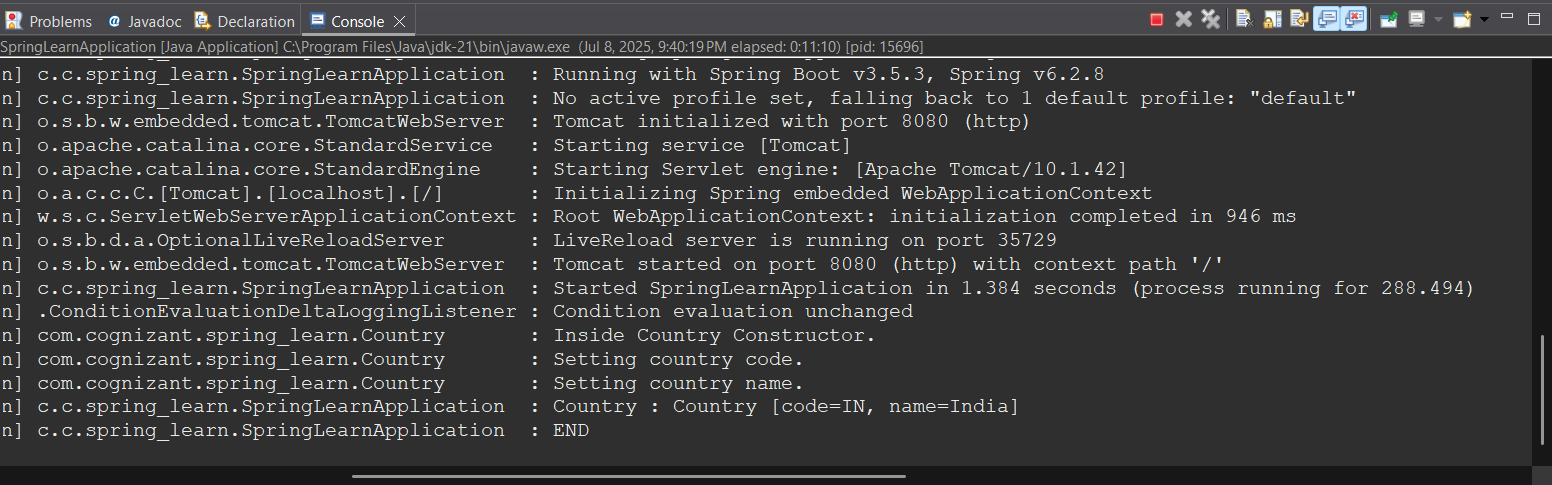
}

**application.properties:**

spring.application.name=spring-learn

logging.level.com.cognizant.spring\_learn=DEBUG

**OUTPUT:** ****

****

**EXERCISE 3:**

**Hello World RESTful Web Service**

**EXPLANATION :**

I created a REST controller named HelloController with a /hello GET endpoint using @RestController and @GetMapping.  
I returned the hardcoded string "Hello World!!" and added logging at start and end of the method.  
I tested the endpoint using a browser and Postman to verify the correct response.

**CODE:**

**HelloController.java:**

package com.cognizant.spring\_learn.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");

LOGGER.info("END");

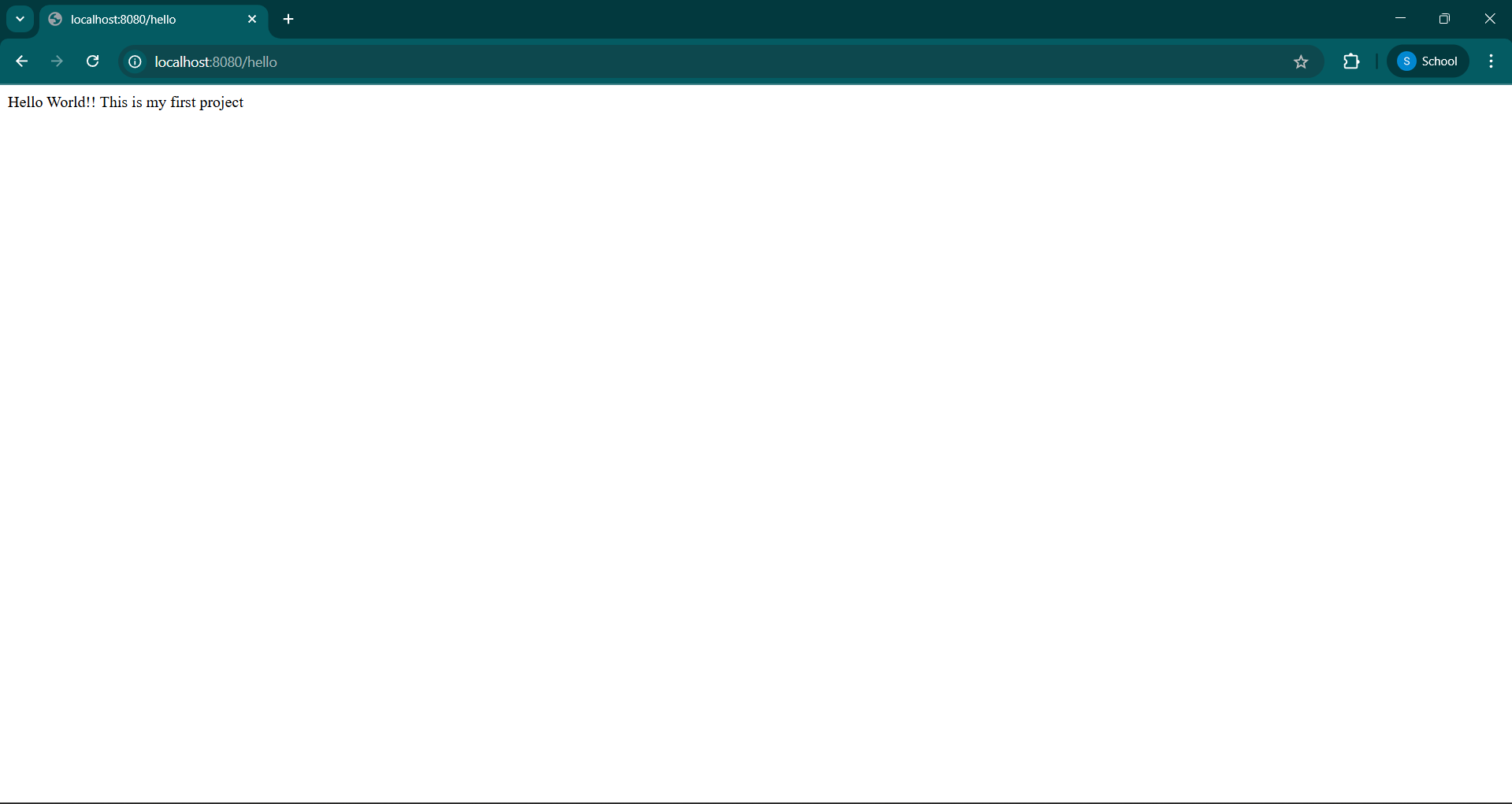
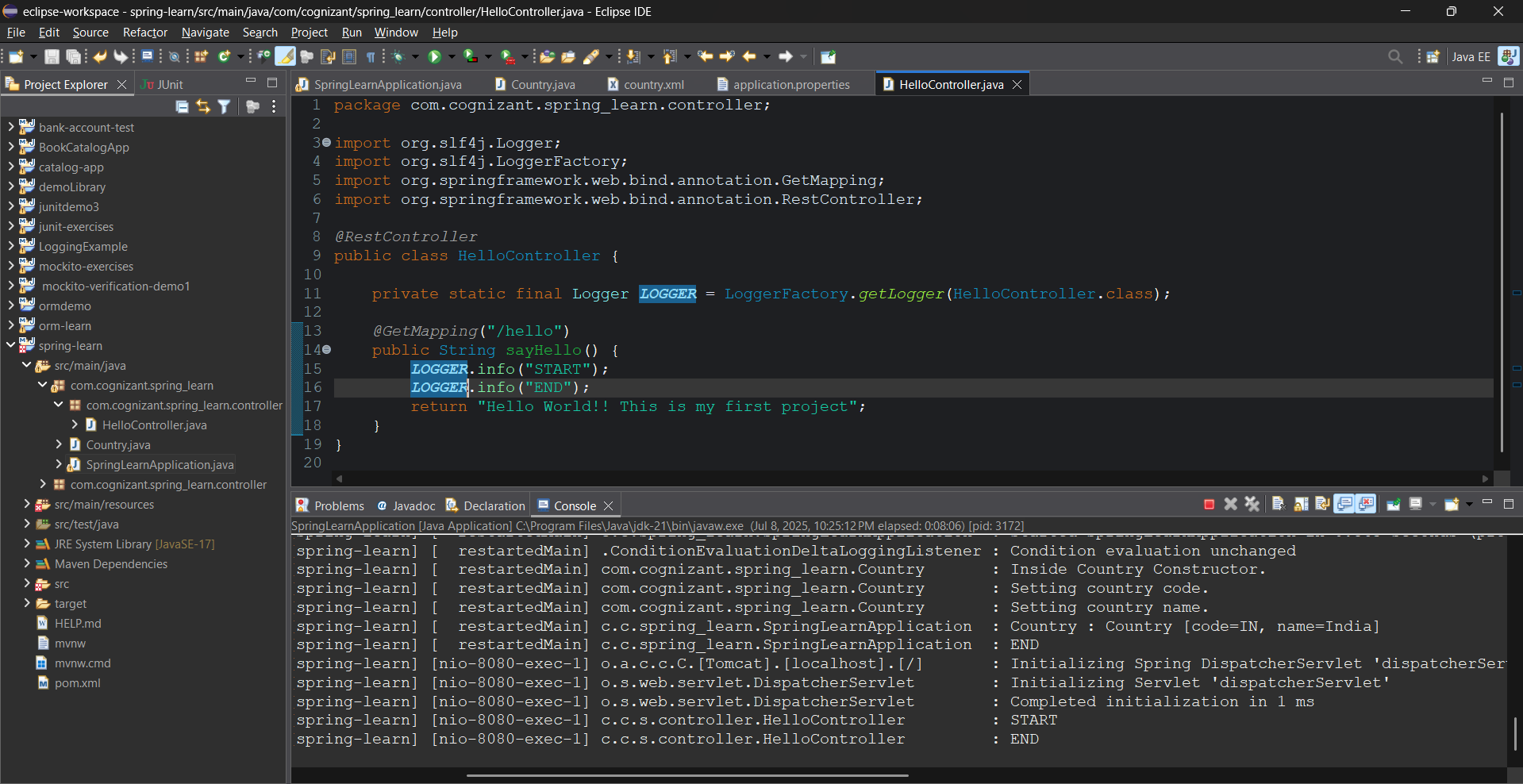
return "Hello World!!";

}

}

**LOCALHOST:**

<http://localhost:8080/hello>

**OUTPUT:** ****

**EXERCISE 4:**

**REST - Country Web Service:**

**EXPLANATION:**

I created a CountryController class with /country REST endpoint using @RestController and @RequestMapping.  
I loaded the Country bean from country.xml and returned it, which got converted to JSON automatically.  
I tested the endpoint using a browser and Postman to verify the JSON response.

**CODE:**

**CountryController.java:**

package com.cognizant.spring\_learn.controller;

import com.cognizant.spring\_learn.Country;

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.RequestMapping;

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

*@RestController*

public class CountryController {

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

*@RequestMapping*("/country")

public Country getCountryIndia() {

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

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

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

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

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

return country;

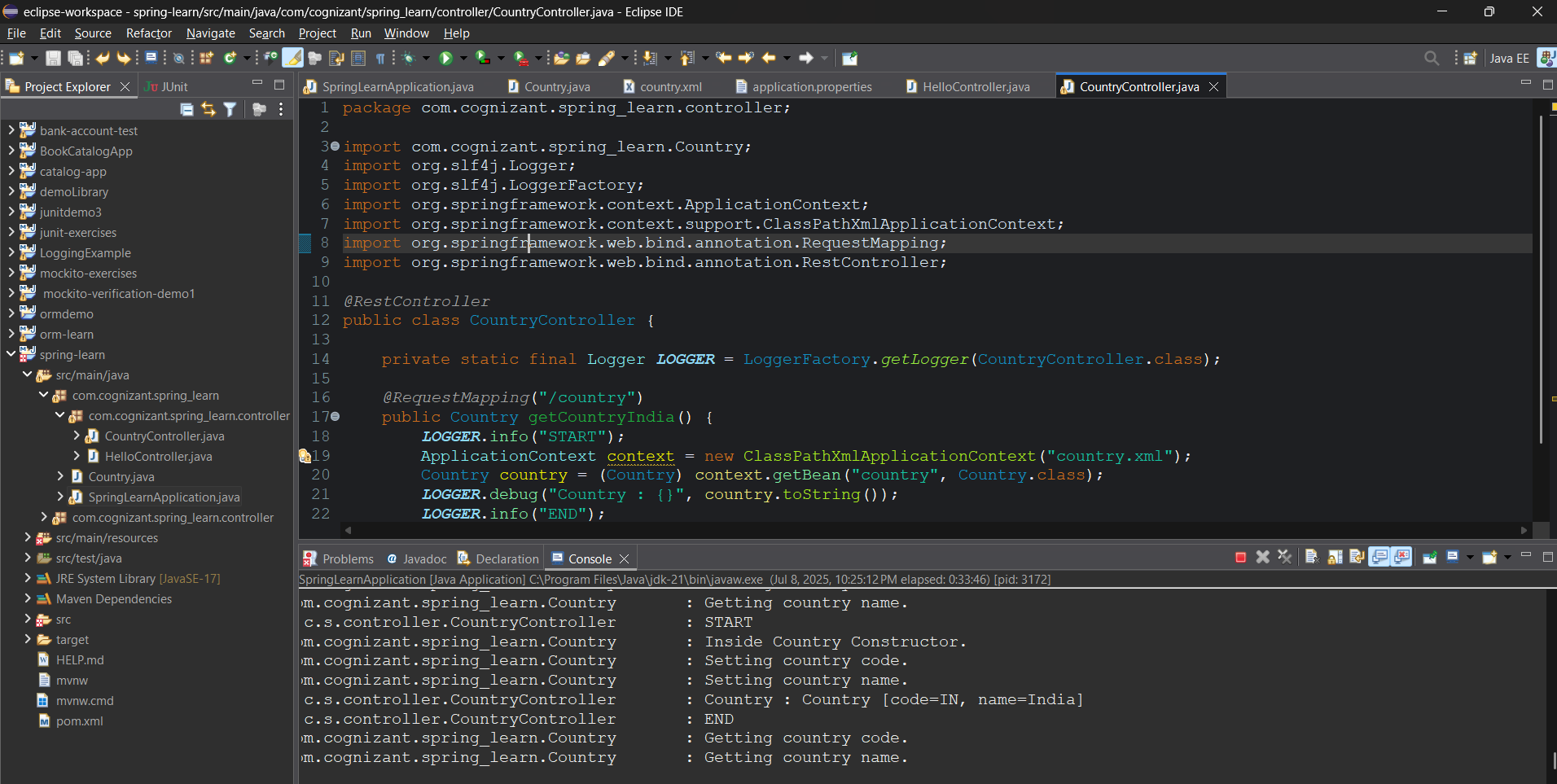
}

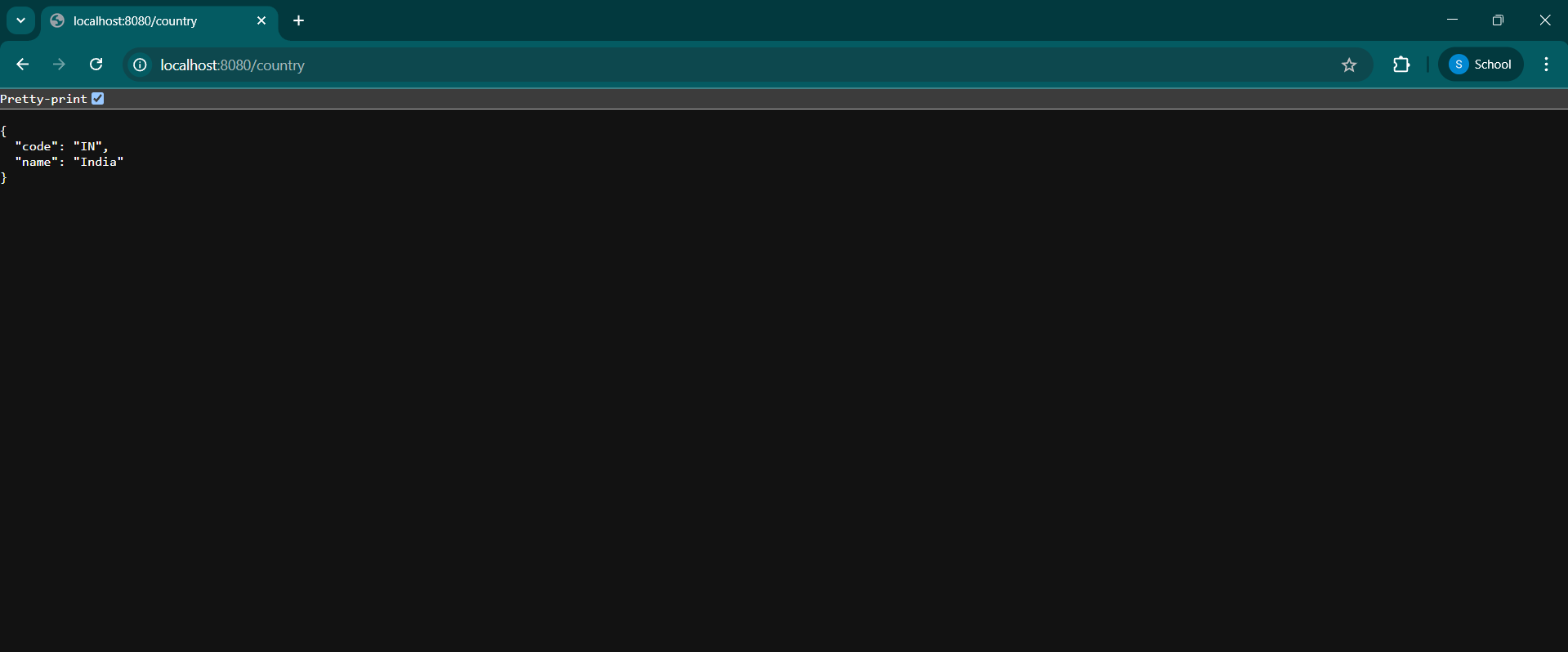
}

**Test in Browser/Postman**

<http://localhost:8080/country>

**OUTPUT:**

****

****

**EXERCISE 5:**

**REST - Get country based on country code:**

**EXPLANATION:**

I added a /countries/{code} endpoint using @PathVariable to search for a country in the XML-configured list.  
If the code matches (case-insensitive), the country is returned as JSON; otherwise, an exception is thrown.  
I tested both valid and invalid country codes using Postman and browser.

**CODE:**

**CountryController.java:**

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

public Country getCountry(@PathVariable String code) throws Exception {

LOGGER.info("START - getCountry({})", code);

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

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

for (Country country : countries) {

if (country.getCode().equalsIgnoreCase(code)) {

LOGGER.info("Country found: {}", country.toString());

LOGGER.info("END");

return country;

}

}

LOGGER.error("Country code {} not found", code);

throw new Exception("Country not found");

}

**country.xml:**

<bean id="in" class="com.cognizant.spring\_learn.Country">

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

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

</bean>

<bean id="us" class="com.cognizant.spring\_learn.Country">

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

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

</bean>

<bean id="jp" class="com.cognizant.spring\_learn.Country">

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

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

</bean>

<bean id="de" class="com.cognizant.spring\_learn.Country">

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

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

</bean>

<bean id="countryList" class="java.util.ArrayList">

<constructor-arg>

<list>

<ref bean="in"/>

<ref bean="us"/>

<ref bean="jp"/>

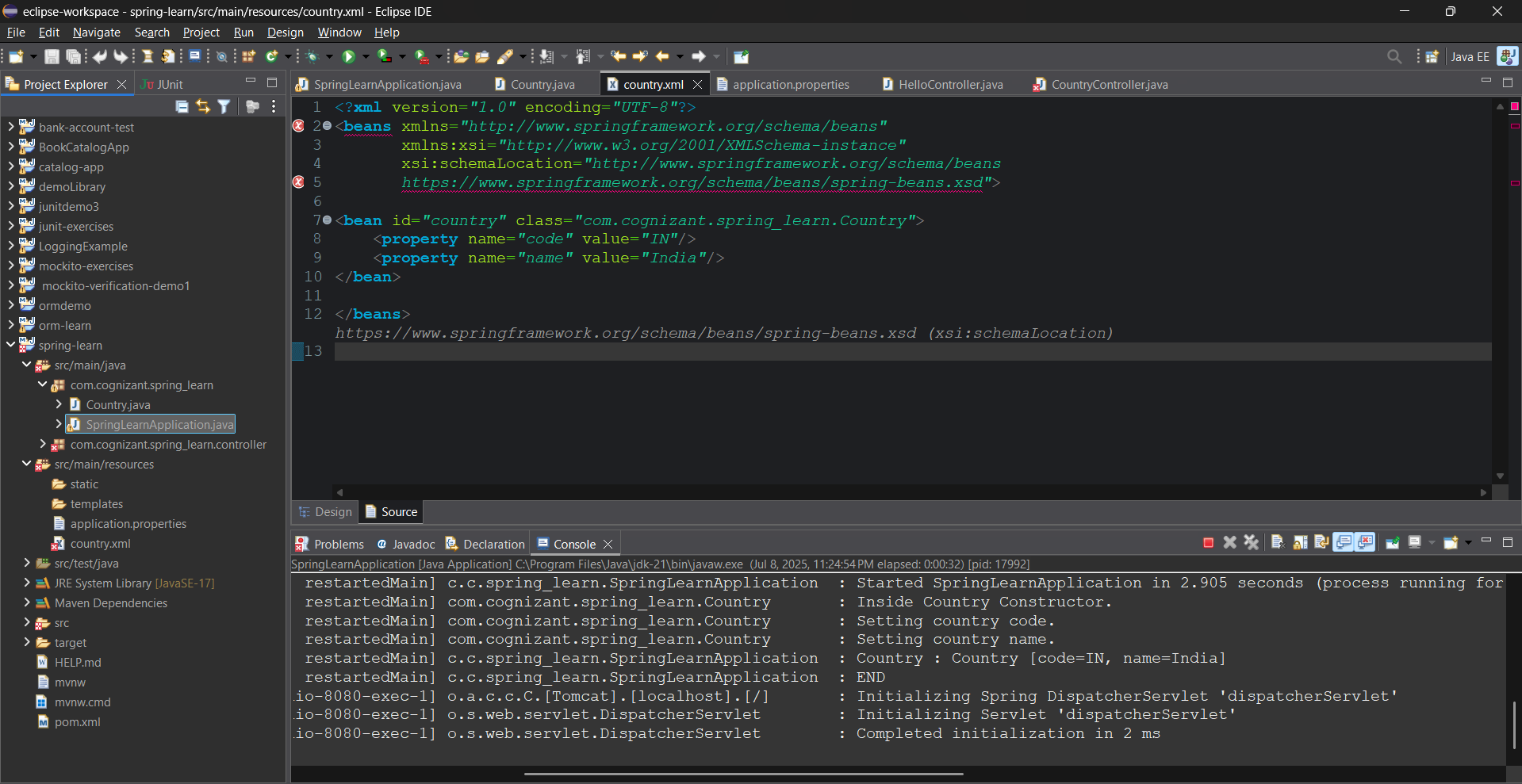
<ref bean="de"/>

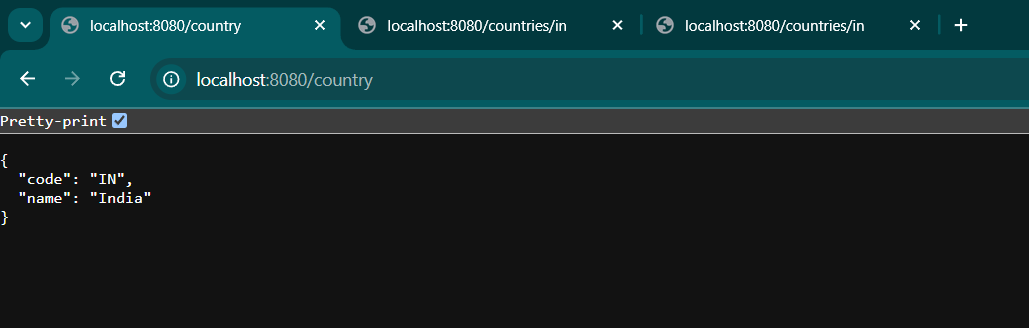
</list>

</constructor-arg>

</bean>

**OUTPUT:**





**EXERCISE 6:**

**Create authentication service that returns JWT:**

**EXPLANATION:**

I created a Spring Boot project named springjwt1 using Maven in Eclipse. I implemented an /authenticate API that returns a JWT token after successful Basic login using a username and password. Spring Security was configured to secure all endpoints and validate JWTs using a custom filter. I added required dependencies such as Spring Web, Spring Security, and JJWT in the pom.xml. The project was tested using the curl command to verify token generation. The setup successfully returned a valid JWT token for authorized users.

**CODE:**

**SpringJwtApplication.java:**

package com.example;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringJwtApplication {

public static void main(String[] args) {

SpringApplication.run(SpringJwtApplication.class, args);

}

}

**SecurityConfig.java:**

package com.example.config;

import com.example.filter.JwtAuthorizationFilter;

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

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.config.annotation.authentication.builders.\*;

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

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

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

@EnableWebSecurity

@Configuration

public class SecurityConfig extends WebSecurityConfigurerAdapter {

@Override

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.inMemoryAuthentication()

.withUser("user").password(passwordEncoder().encode("pwd")).roles("USER")

.and()

.withUser("admin").password(passwordEncoder().encode("pwd")).roles("ADMIN");

}

@Override

protected void configure(HttpSecurity http) throws Exception {

http.csrf().disable().httpBasic().and()

.authorizeRequests()

.antMatchers("/authenticate").hasAnyRole("USER", "ADMIN")

.anyRequest().authenticated()

.and()

.addFilter(new JwtAuthorizationFilter(authenticationManager()));

}

@Bean

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

@Bean

public AuthenticationManager authenticationManagerBean() throws Exception {

return super.authenticationManagerBean();

}

}

**AuthenticationController.java:**

package com.example.controller;

import io.jsonwebtoken.\*;

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

import java.util.\*;

@RestController

public class AuthenticationController {

@GetMapping("/authenticate")

public Map<String, String> authenticate(@RequestHeader("Authorization") String authHeader) {

System.out.println(">>> authHeader = " + authHeader); // debug log

Map<String, String> map = new HashMap<>();

try {

String user = getUser(authHeader);

String token = generateJwt(user);

map.put("token", token);

} catch (Exception e) {

e.printStackTrace();

map.put("error", "Failed to generate token");

}

return map;

}

private String getUser(String authHeader) {

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

byte[] decodedBytes = Base64.getDecoder().decode(encoded);

return new String(decodedBytes).split(":")[0];

}

private String generateJwt(String user) {

JwtBuilder builder = Jwts.builder();

builder.setSubject(user);

builder.setIssuedAt(new Date());

builder.setExpiration(new Date(System.currentTimeMillis() + 1200000)); // 20 min

builder.signWith(SignatureAlgorithm.HS256, "secretkey");

return builder.compact();

}

}

**JwtAuthorizationFilter.java:**

package com.example.filter;

import io.jsonwebtoken.\*;

import org.springframework.security.authentication.\*;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.security.web.authentication.www.BasicAuthenticationFilter;

import javax.servlet.\*;

import javax.servlet.http.\*;

import java.io.IOException;

import java.util.\*;

public class JwtAuthorizationFilter extends BasicAuthenticationFilter {

public JwtAuthorizationFilter(AuthenticationManager authManager) {

super(authManager);

}

@Override

protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response,

FilterChain chain) throws IOException, ServletException {

String header = request.getHeader("Authorization");

if (header == null || !header.startsWith("Bearer ")) {

chain.doFilter(request, response);

return;

}

UsernamePasswordAuthenticationToken authentication = getAuthentication(header);

SecurityContextHolder.getContext().setAuthentication(authentication);

chain.doFilter(request, response);

}

private UsernamePasswordAuthenticationToken getAuthentication(String header) {

try {

String token = header.replace("Bearer ", "");

String user = Jwts.parser()

.setSigningKey("secretkey")

.parseClaimsJws(token)

.getBody()

.getSubject();

if (user != null) {

return new UsernamePasswordAuthenticationToken(user, null, new ArrayList<>());

}

} catch (JwtException e) {

return null;

}

return null;

}

}

**pom.xml:**

<project xmlns="http://maven.apache.org/POM/4.0.0"

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

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>springjwt1</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<!-- ✅ Add Spring Boot parent -->

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.7.10</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<!-- ✅ Wrap all dependencies inside this block -->

<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>0.9.0</version>

</dependency>

<dependency>

<groupId>javax.xml.bind</groupId>

<artifactId>jaxb-api</artifactId>

<version>2.3.1</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

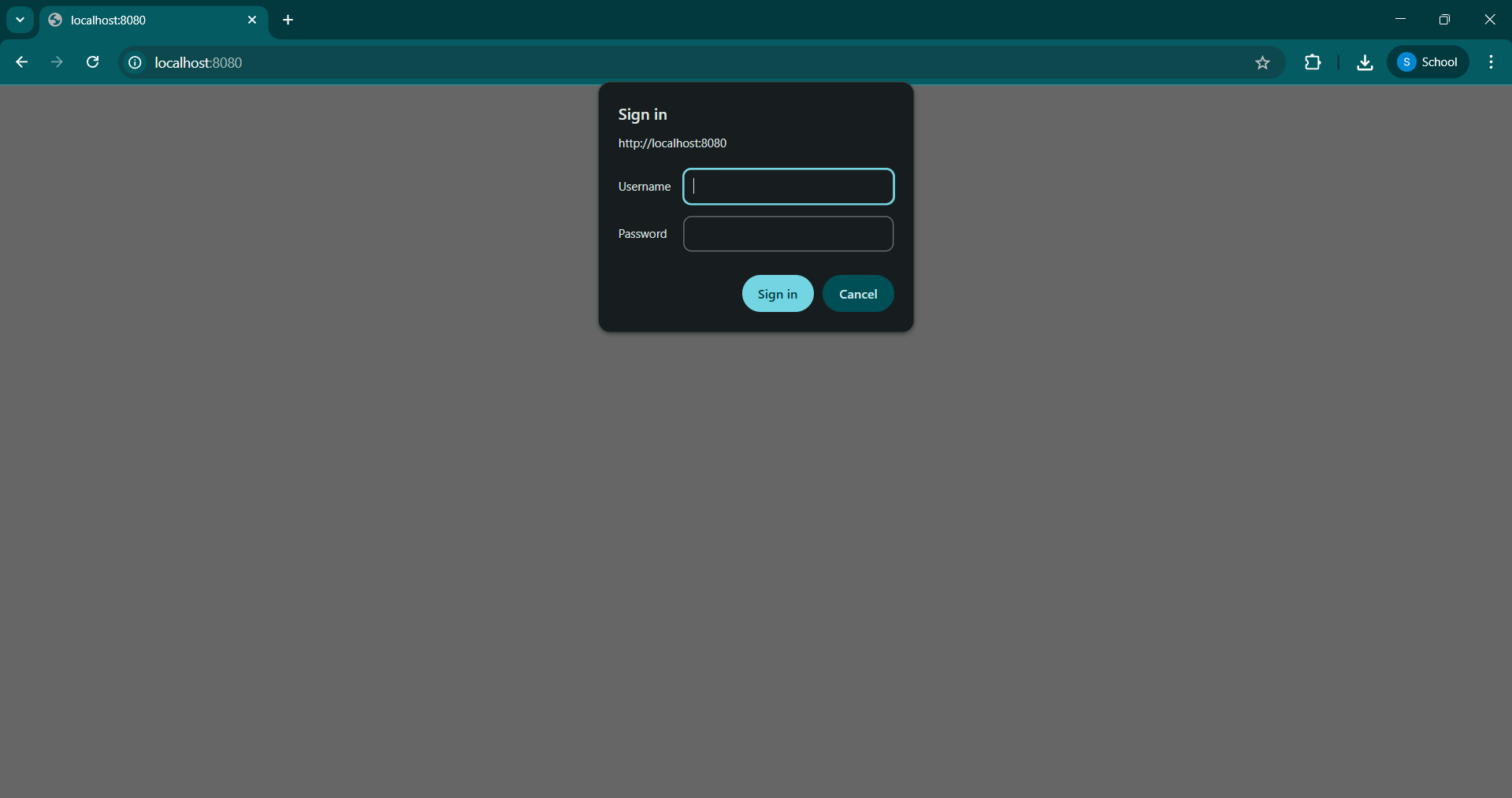
<artifactId>spring-boot-maven-plugin</artifactId>

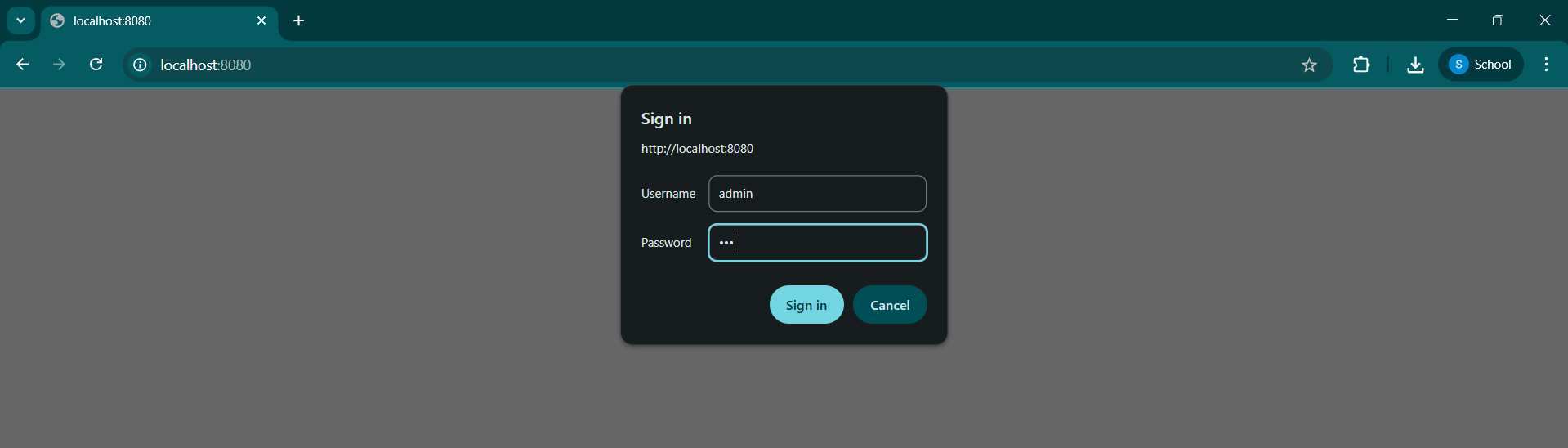
</plugin>

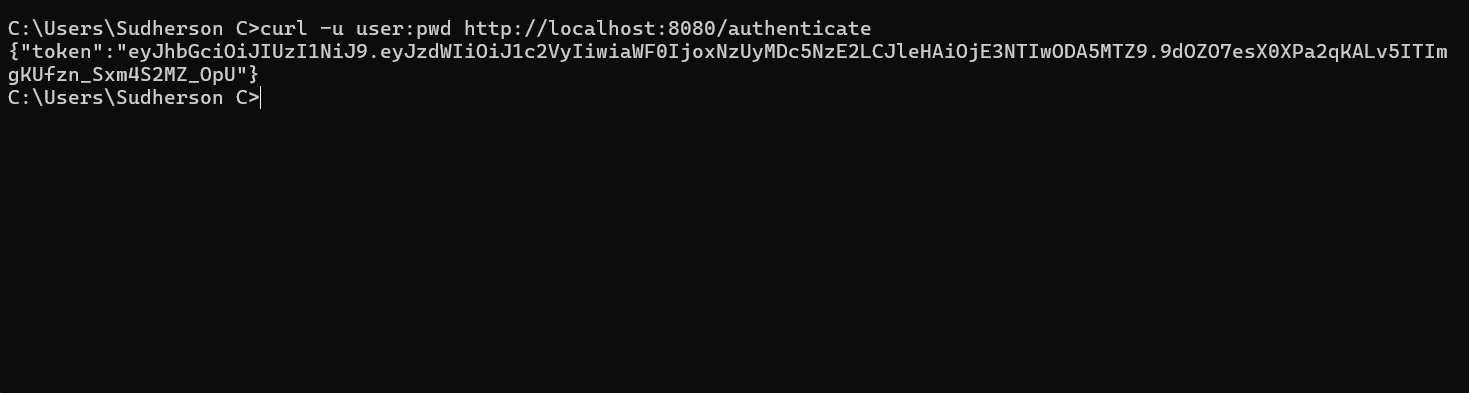
</plugins>

</build>

</project>

**OUTPUT:** ****

****

**COMMAND PROMPT OUTPUT:** ****