**Cognizant Deep Skilling - Digital Nurture 4.0**

**Spring REST using Spring Boot 3**

**Filename - 1. spring-rest-handson**

**Hands on 1**

**Create a Spring Web Project using Maven**

**SpringLearnApplication.java**

package com.cognizant.spring\_learn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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);

LOGGER.info("END");

}

}

**Pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<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.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-learn</name>

<description>Demo project for Spring Boot</description>

<packaging>jar</packaging>

<properties>

<java.version>21</java.version>

</properties>

<parent>

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

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

<version>3.5.3</version>

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

</parent>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

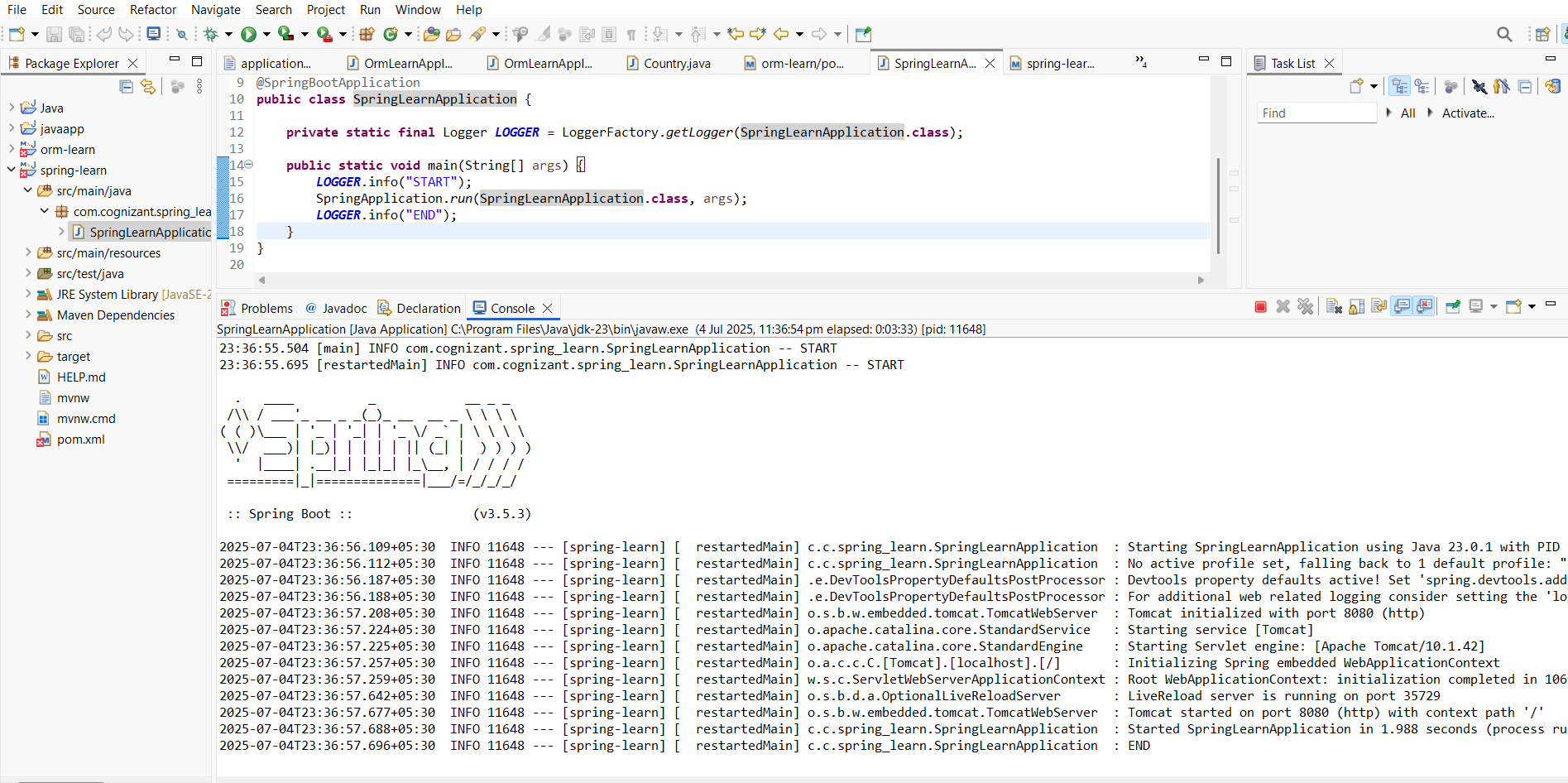
</plugin>

</plugins>

</build>

</project>

**Output:**

****

**Hands on 4**

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

**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

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>

**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=" + getCode() + ", name=" + getName() + "]";

}

}

**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;

public class SpringLearnApplication {

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

public static void main(String[] args) {

*displayCountry*();

}

public static void displayCountry() {

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

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

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

}

}

**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.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>1.0.0</version>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.38</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>2.0.12</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-simple</artifactId>

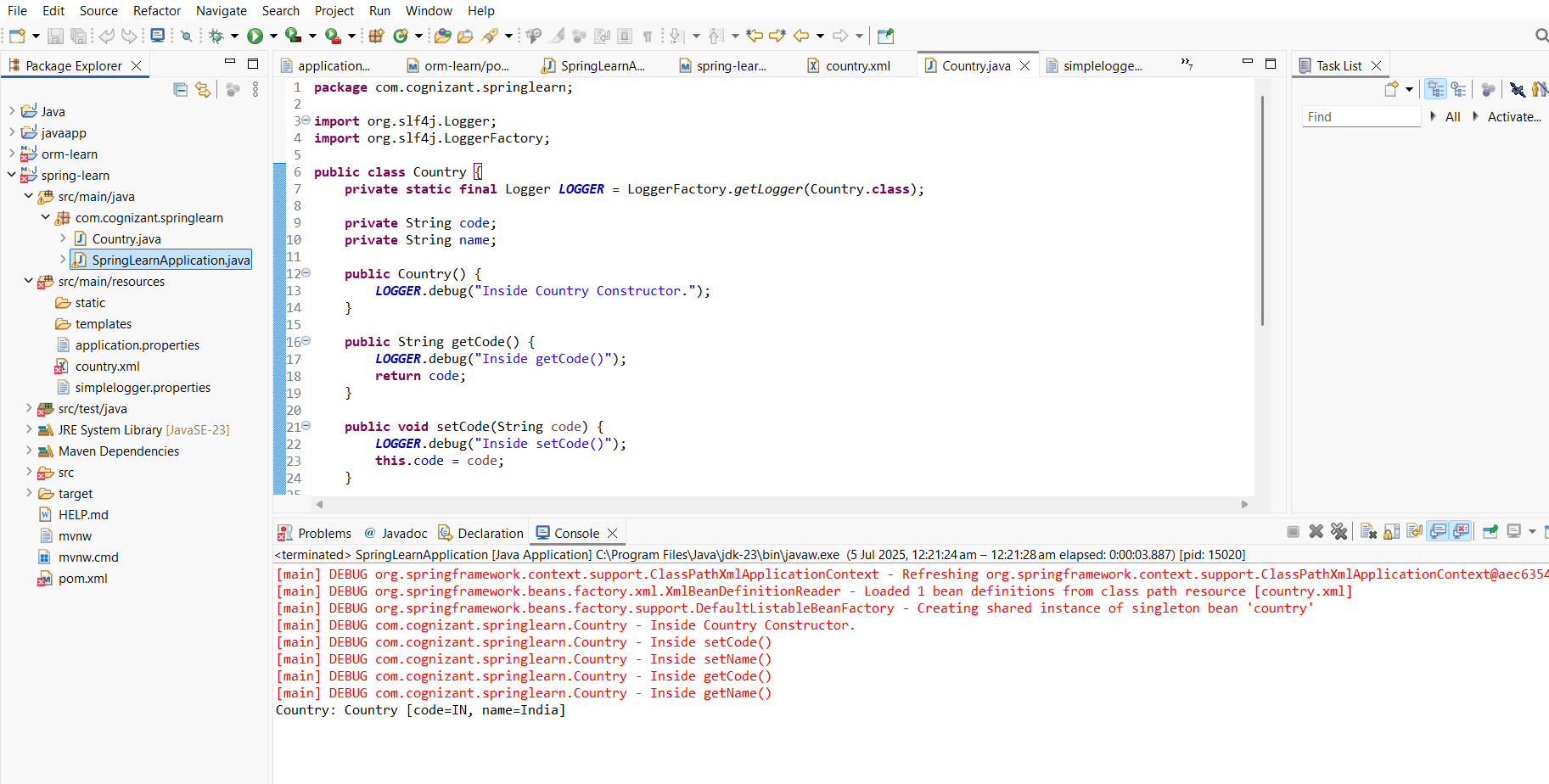
<version>2.0.12</version>

</dependency>

</dependencies>

</project>

**Output:**

****

**Filename - 2. spring-rest-handson**

**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.**

**HelloController.java**

package com.example.demo;

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 - sayHello()");

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

return "Hello World!!";

}

}

**DemoApplication.java**

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class DemoApplication {

public static void main(String[] args) {

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

}

}

**application.properties**

server.port=8083

**pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<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>

<parent>

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

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

<version>3.5.3</version>

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

</parent>

<groupId>com.example</groupId>

<artifactId>demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>demo</name>

<description>Demo project for Spring Boot</description>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

<java.version>21</java.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

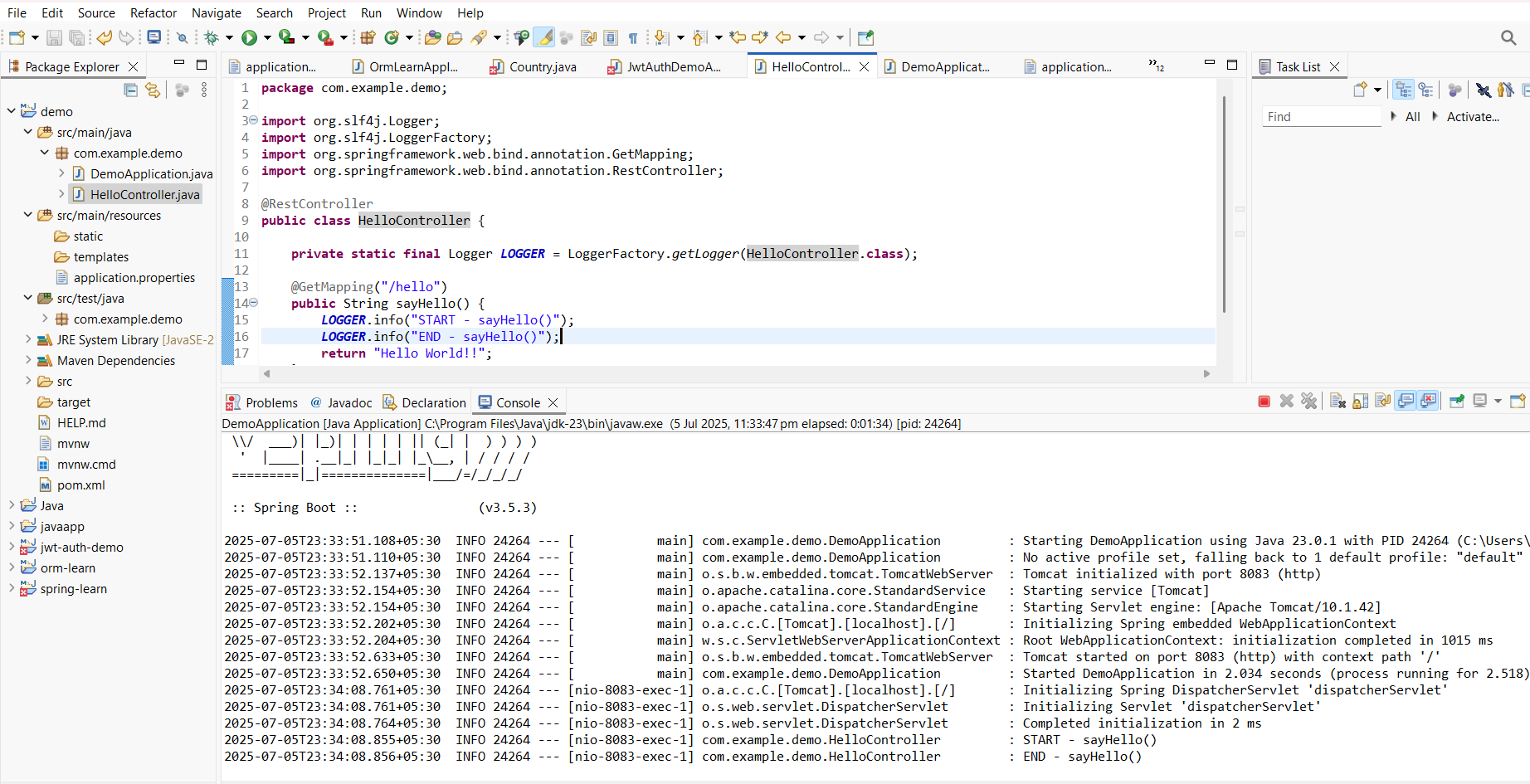
</plugin>

</plugins>

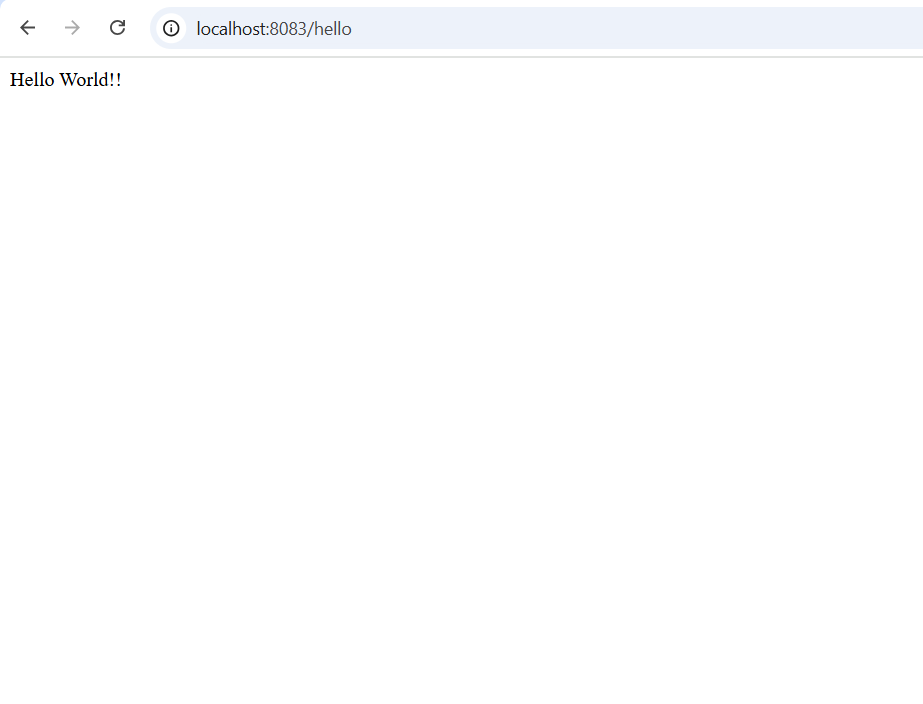
</build>

</project>

**Output:**

****

**Request**: <http://localhost:8083/hello>

****

**REST - Country Web Service**

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

**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

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

<bean id="country" class="com.example.demo.model.Country">

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

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

</bean>

</beans>

**Country.java**

package com.example.demo.model;

public class Country {

private String code;

private String name;

public String getCode() {

return code;

}

public void setCode(String code) {

this.code = code;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

**CountryController.java**

package com.example.demo.controller;

import com.example.demo.model.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 - getCountryIndia()");

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

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

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

return country;

}

}

**DemoApplication.java**

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class DemoApplication {

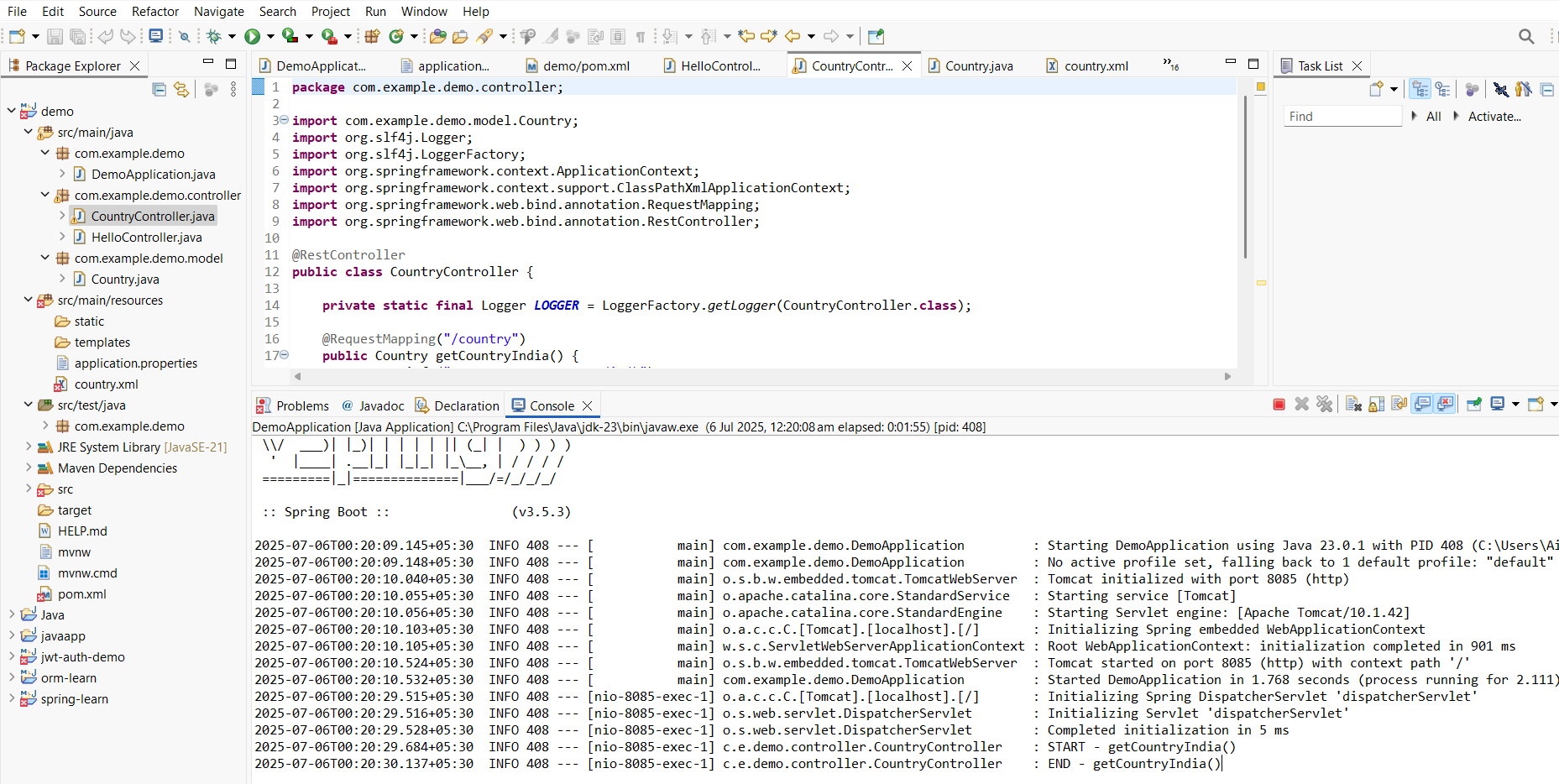
public static void main(String[] args) {

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

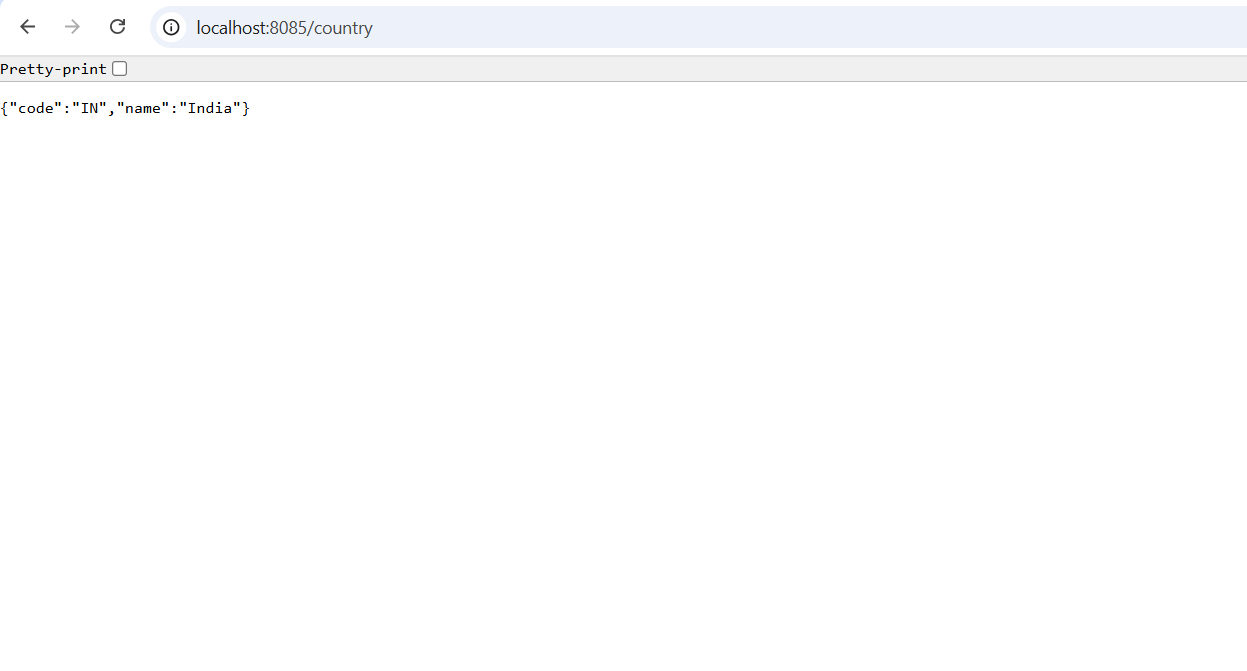
}

}

**Output:**

****

**Request**: <http://localhost:8085/country>

****

**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.**

**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

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

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

<constructor-arg>

<list>

<bean class="com.example.demo.model.Country">

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

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

</bean>

<bean class="com.example.demo.model.Country">

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

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

</bean>

</list>

</constructor-arg>

</bean>

</beans>

**Country.java**

package com.example.demo.model;

public class Country {

private String code;

private String name;

public String getCode() {

return code;

}

public void setCode(String code) {

this.code = code;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Country() {}

public Country(String code, String name) {

this.code = code;

this.name = name;

}

}

**CountryService.java**

package com.example.demo.service;

import java.util.List;

import java.util.Optional;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.stereotype.Service;

import com.example.demo.model.Country;

@Service

public class CountryService {

public Country getCountry(String code) {

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

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

Optional<Country> country = countryList.stream()

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

.findFirst();

if (country.isPresent()) {

return country.get();

} else {

throw new RuntimeException("Country not found for code: " + code);

}

}

}

**CountryController.java**

package com.example.demo.controller;

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

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

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

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

import com.example.demo.model.Country;

import com.example.demo.service.CountryService;

@RestController

public class CountryController {

@Autowired

private CountryService countryService;

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

public Country getCountry(@PathVariable String code) {

return countryService.getCountry(code);

}

}

**application.properties**

server.port=8082

**DemoApplication.java**

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class DemoApplication {

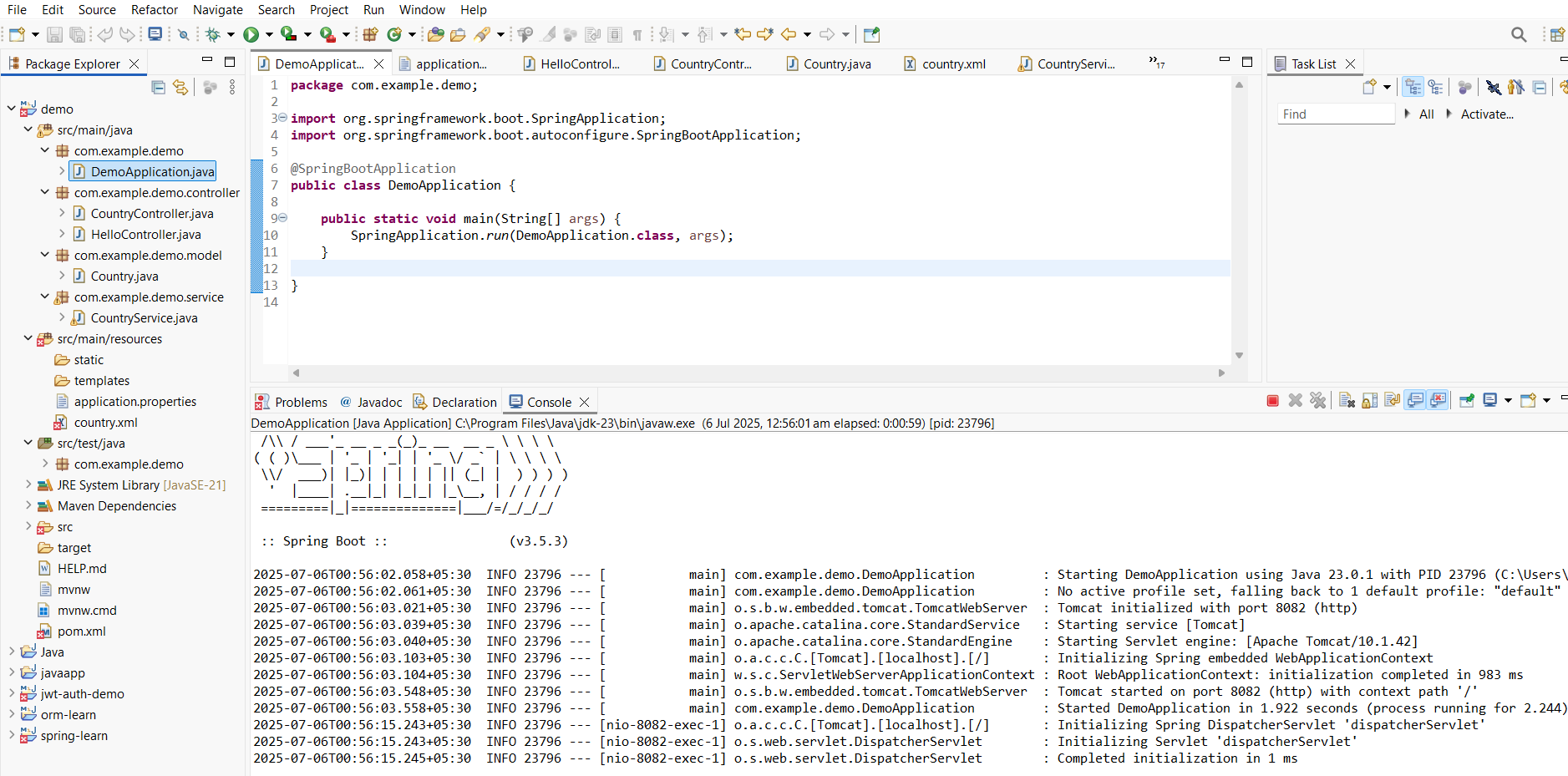
public static void main(String[] args) {

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

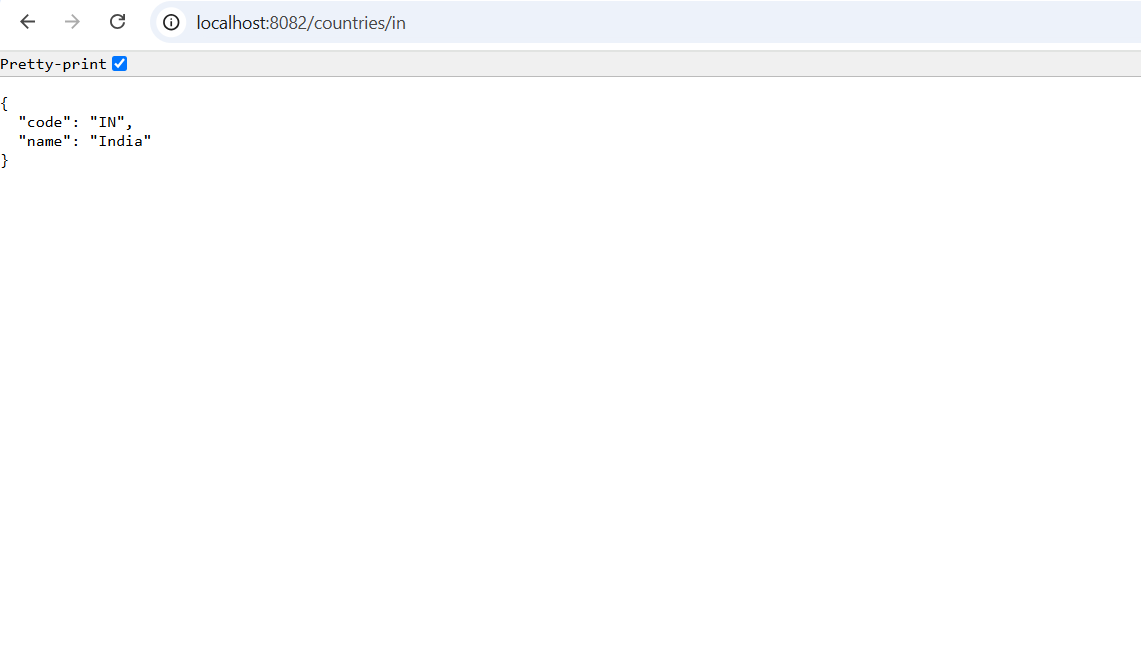
}

}

**Output:**

****

**Request**: <http://localhost:8082/countries/in>



**Filename - 5. JWT-handson**

**Create authentication service that returns JWT**

**As part of first step of JWT process, the user credentials needs to be sent to authentication service request that generates and returns the JWT.**

**Pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<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>

<parent>

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

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

<version>3.5.3</version>

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

</parent>

<groupId>com.example</groupId>

<artifactId>jwt-auth-service</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>jwt-auth-service</name>

<description>Demo project for Spring Boot</description>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

<java.version>21</java.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-test</artifactId>

<scope>test</scope>

</dependency>

<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>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<configuration>

<annotationProcessorPaths>

<path>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</path>

</annotationProcessorPaths>

</configuration>

</plugin>

<plugin>

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

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

<configuration>

<excludes>

<exclude>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</exclude>

</excludes>

</configuration>

</plugin>

</plugins>

</build>

</project>

**SecurityConfig.java**

package com.example.jwtauthservice.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.web.SecurityFilterChain;

@Configuration

public class SecurityConfig {

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

.csrf().disable()

.authorizeHttpRequests(auth -> auth

.anyRequest().permitAll()

);

return http.build();

}

}

**AuthenticationController.java**

package com.example.jwtauthservice.controller;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

import org.springframework.http.ResponseEntity;

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

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

import jakarta.servlet.http.HttpServletRequest;

import javax.crypto.SecretKey;

import java.util.Base64;

import java.util.Date;

@RestController

public class AuthenticationController {

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

@GetMapping("/authenticate")

public ResponseEntity<?> authenticate(HttpServletRequest request) {

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

if (authHeader == null || !authHeader.startsWith("Basic ")) {

return ResponseEntity.*status*(401).body("Missing or invalid Authorization header.");

}

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

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

String decodedCredentials = new String(decodedBytes);

String[] credentials = decodedCredentials.split(":", 2);

if (credentials.length != 2) {

return ResponseEntity.*status*(400).body("Invalid Basic authentication format.");

}

String username = credentials[0];

String password = credentials[1];

if (!"user".equals(username) || !"pwd".equals(password)) {

return ResponseEntity.*status*(401).body("Invalid credentials.");

}

String token = Jwts.*builder*()

.setSubject(username)

.setIssuedAt(new Date())

.setExpiration(new Date(System.*currentTimeMillis*() + 3600 \* 1000))

.signWith(*SECRET\_KEY*)

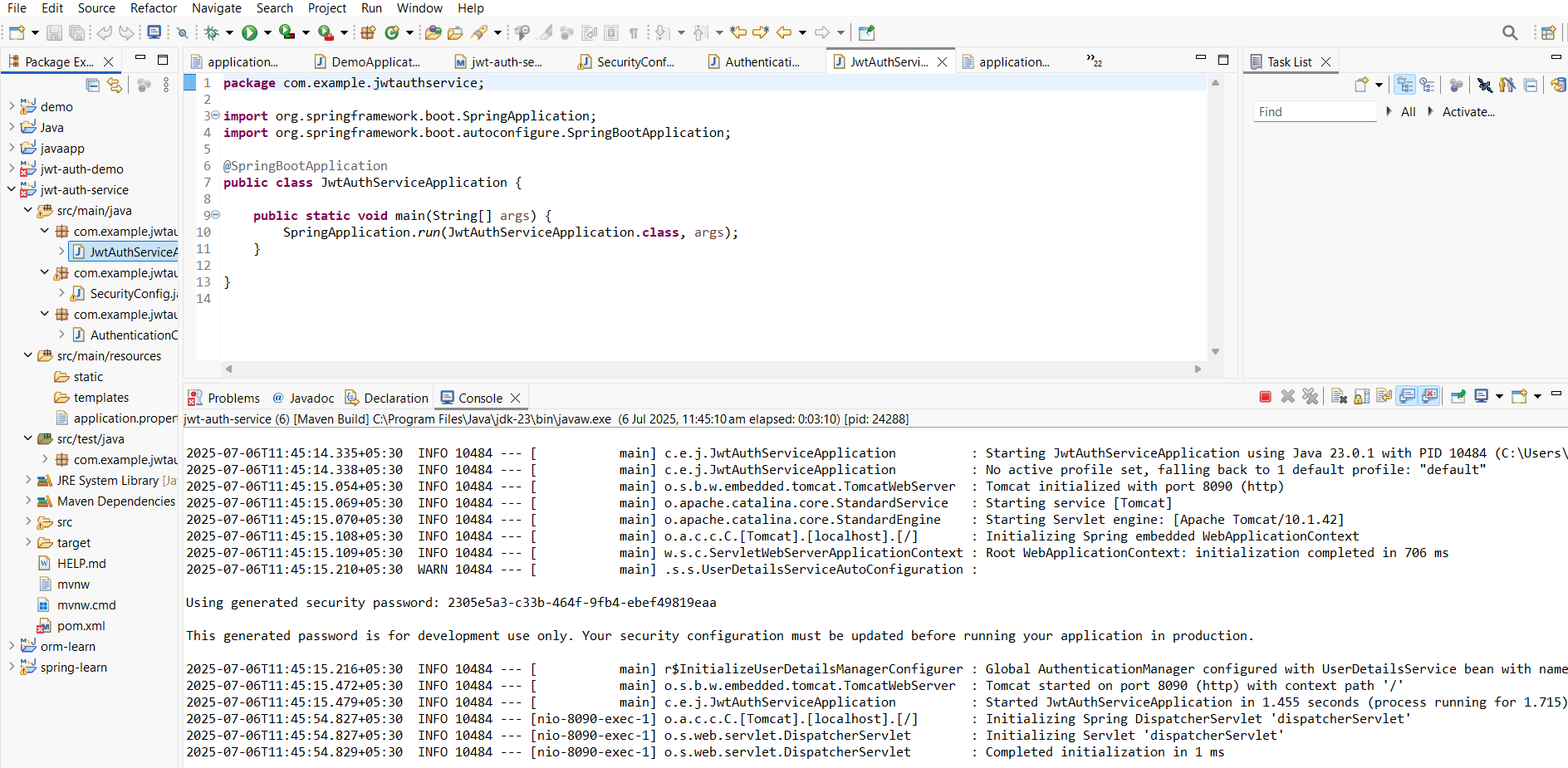
.compact();

return ResponseEntity.*ok*().body("{\"token\":\"" + token + "\"}");

}

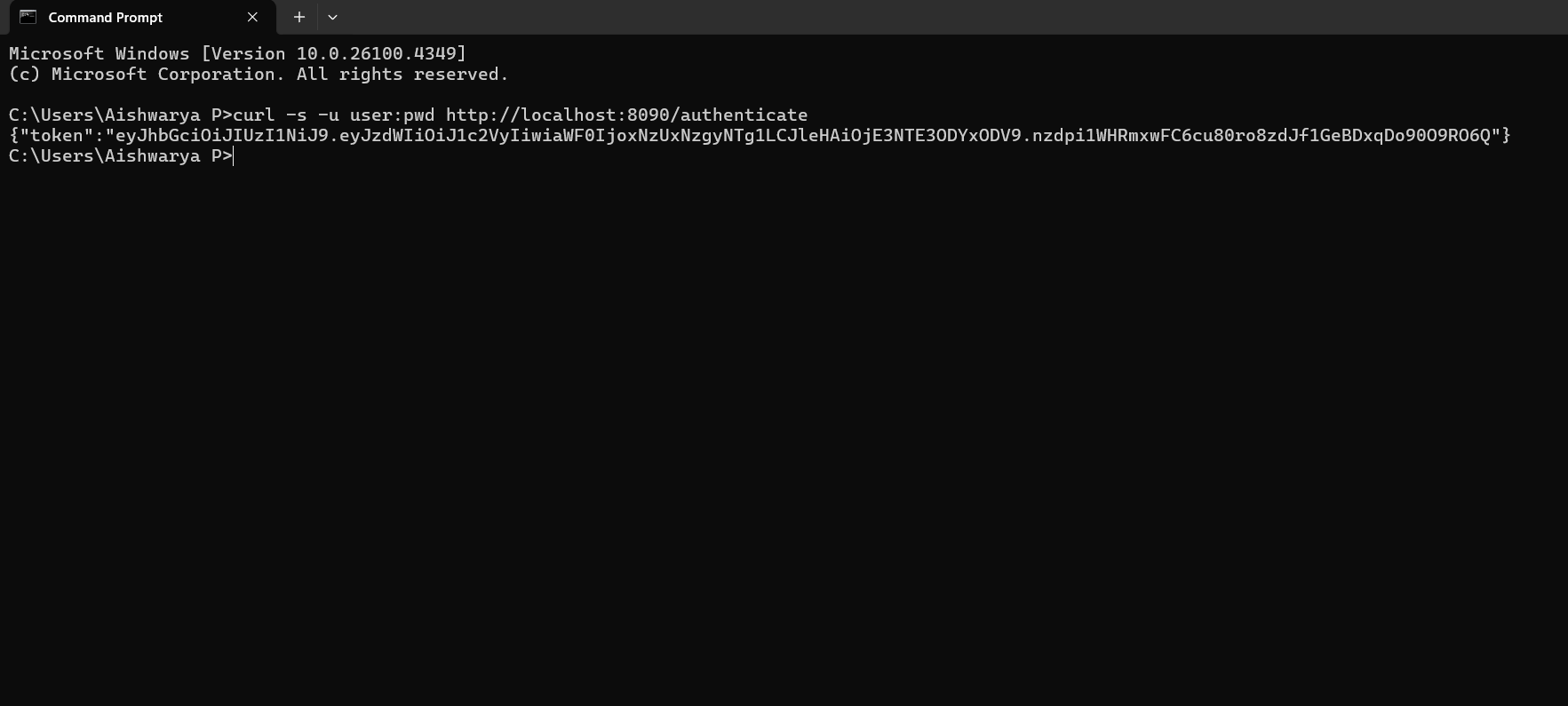
}

**Output:**

****

**Request**

curl -s -u user:pwd http://localhost:8090/authenticate

****