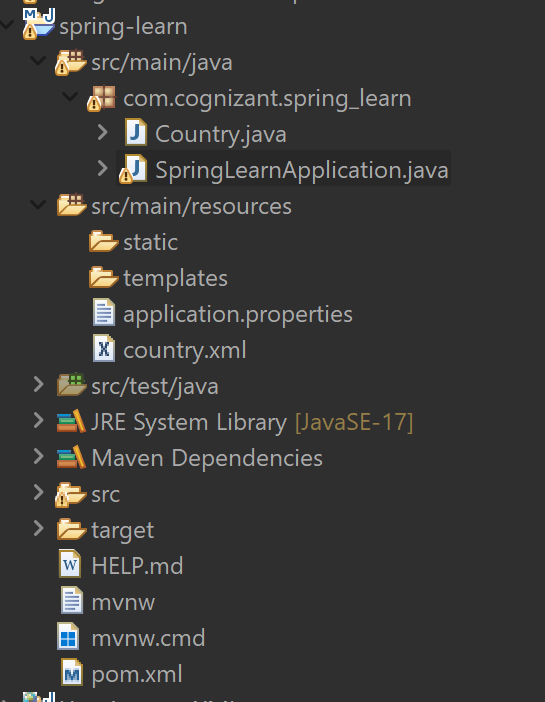
**Exercise 1: Create a Spring Web Project using Maven**

**Steps:**

1. **Set Up a Spring Project:**
   * Create a Maven project named **spring-learn**.



1. **Add the dependencies:**
   * Add Spring Core dependencies in the **pom.xml** file.

**pom.xml:**

<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>

1. **Create SpringLearnapplication Context:**
   * Define the springlearnapplication class.

**SpringLearnapplication.java:**

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

@SpringBootApplication // Important to enable component scan and Spring Boot features

public class SpringLearnApplication {

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

public static void main(String[] args) {

LOGGER.info("START");

//This starts the Spring Boot web application on the configured port (like 8081)

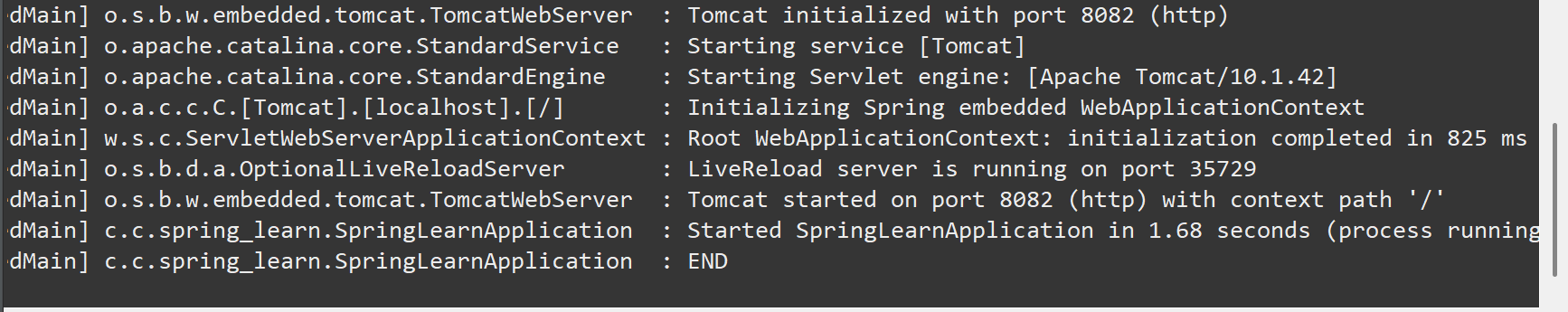
SpringApplication.run(SpringLearnApplication.class, args);

LOGGER.info("END");

    }

}

**OUTPUT:**

****

**Exercise 2: Spring Core – Load Country from Spring Configuration XML**

**Steps:**

1. **Add the XML file:**

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

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

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

</bean>

1. **Add the Country.java Class:**

**BookService.java:**

public class Country {

private String code;

private String name;

public Country() {

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

}

public void setCode(String code) {

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

this.code = code;

}

public void setName(String name) {

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

this.name = name;

}

public String toString() {

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

}

}

1. **Modify SpringLearnApplication:**
   * Add the display country method.

**SpringLearnapplication.java**

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

//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) {

LOGGER.info("START");

displayCountry();

LOGGER.info("END");

}

public static void displayCountry() {

System.out.println("Before loading context");

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

System.out.println("After loading context");

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

System.out.println("After getBean");

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

}

}

1. **Configure applications.properties:**

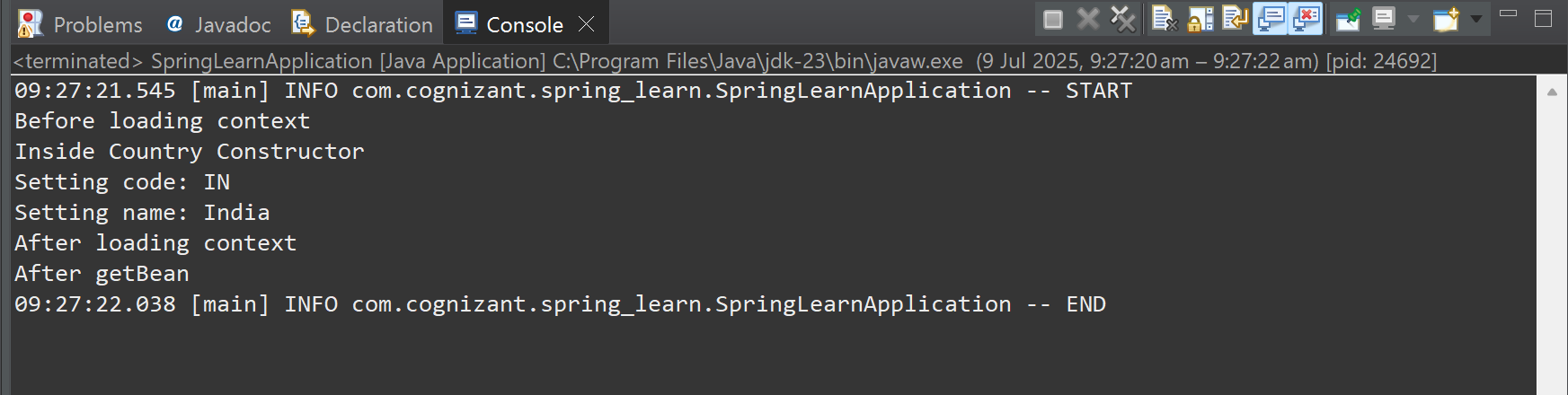
**application.properties**

spring.application.name=spring-learn

server.port=8081

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

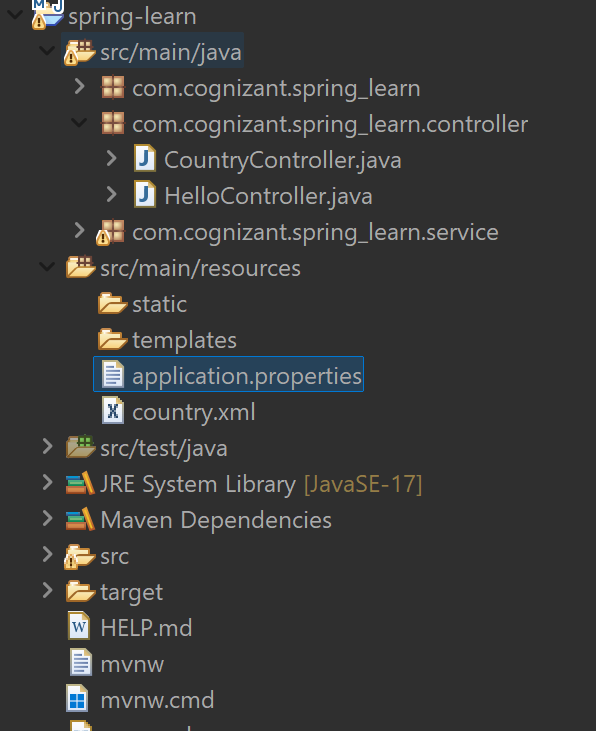
**OUTPUT:**



**Exercise 3: Hello World RESTful Web Service**

**Steps:**

1. **Modify the Maven Project:**
   * Create a controller package.



1. **Add HelloController.java class:**
   * Create the controller class:

**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: sayHello()");

String message = "Hello World!!";

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

return message;

}

}

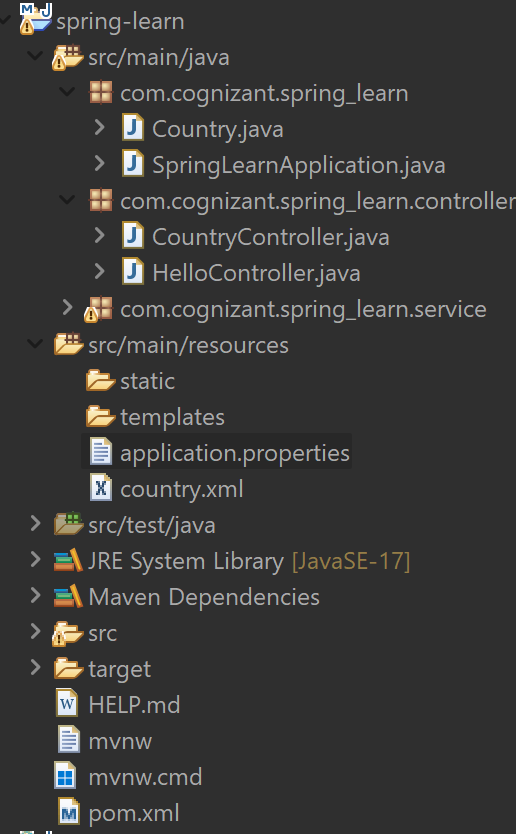
**OUTPUT:**



**Exercise 4: REST - Country Web Service**

**Steps:**

1. **Ensure we have the previous file country.java and country.xml**



1. **Add CountryController.java class:**
   * Create the controller class:

**CountryController.java**

package com.cognizant.spring\_learn.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.RequestMapping;

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

import com.cognizant.spring\_learn.model.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.info("END: getCountryIndia");

return country;

}

}

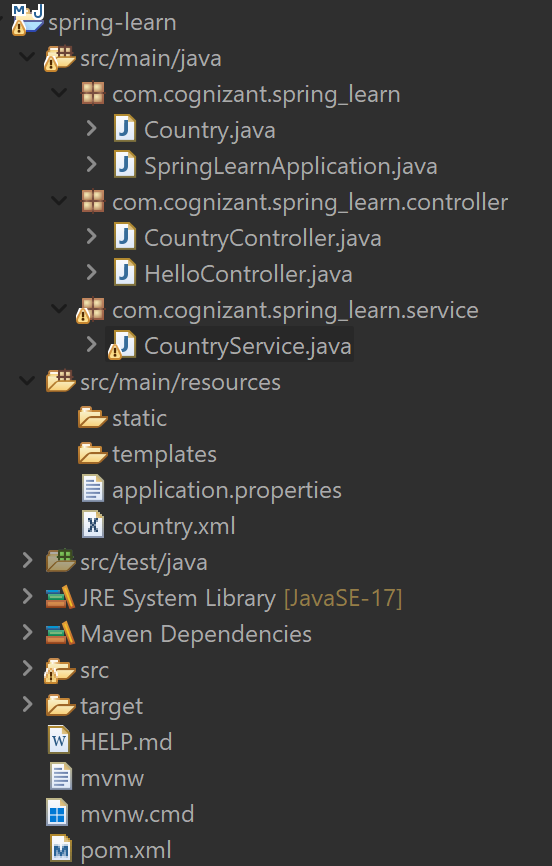
**OUTPUT:**



**Exercise 5: REST - Get country based on country code**

**Steps:**

1. **Modify the previous file country.java and country.xml**



**Country.xml**

<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.spring\_learn.Country">

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

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

</bean>

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

<constructor-arg>

<list>

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

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

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

</bean>

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

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

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

</bean>

</beans>

1. **Modify CountryController.java class:**

**CountryController.java**

package com.cognizant.spring\_learn.controller;

import com.cognizant.spring\_learn.Country;

import com.cognizant.spring\_learn.service.CountryService;

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

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

@RestController

public class CountryController {

@Autowired

private CountryService countryService;

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

public Country getCountry(@PathVariable String code) {

return countryService.getCountry(code);

}

}

1. **Add CountryService.java class**

**CountryService.java**

package com.cognizant.spring\_learn.service;

import com.cognizant.spring\_learn.Country;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class CountryService {

public Country getCountry(String code) {

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

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

return countryList.stream()

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

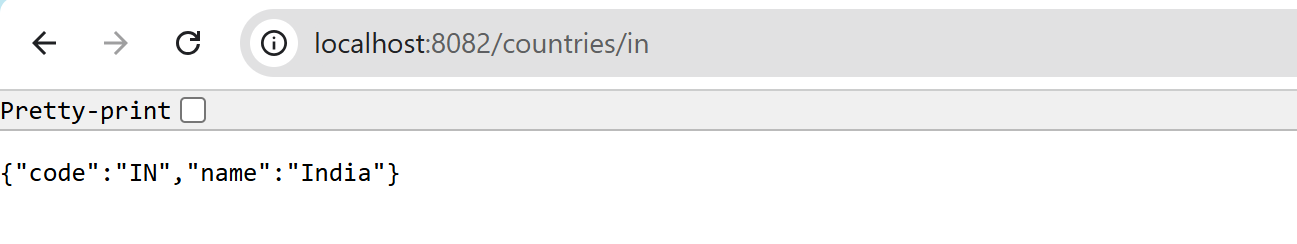
.findFirst()

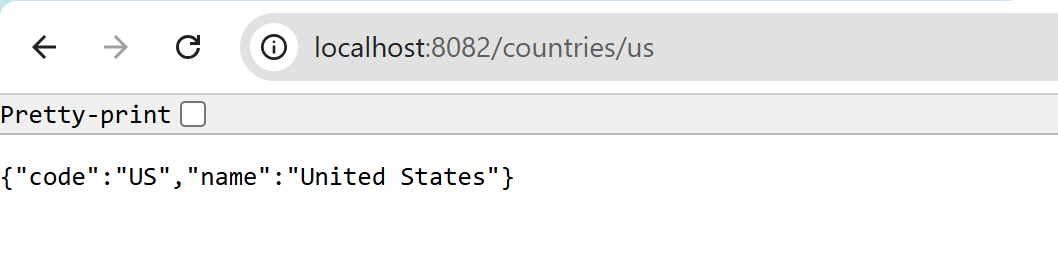
.orElse(null);

}

}

**OUTPUT:**

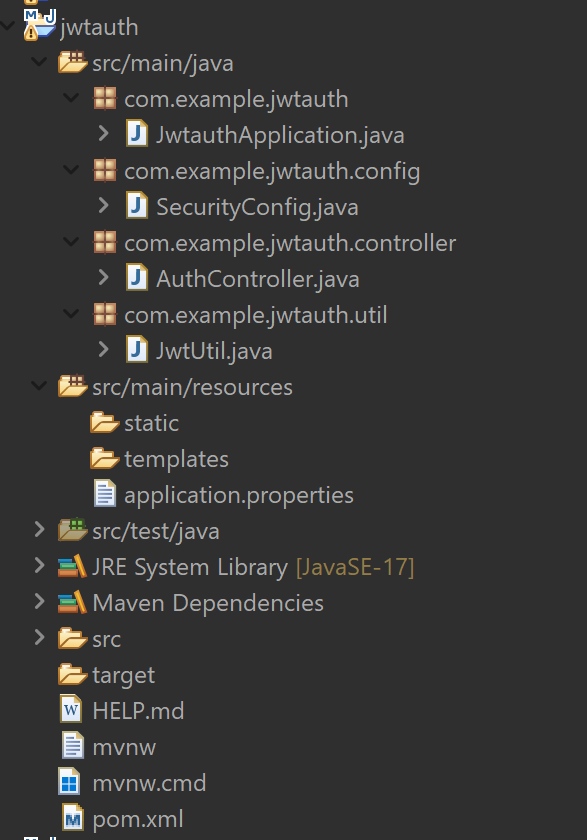




**Exercise 6: Create authentication service that returns JWT**

**Steps:**

1. **Create a Maven Project named jwtauth**



1. **Configure the pom.xml:**

* Add the jwt dependencies.

**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> <!-- instead of jjwt-gson or jjwt-orgjson -->

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

1. **Add the required classes:**

* Created a Spring Boot app (JwtauthApplication) with security config (SecurityConfig) to allow basic auth, used AuthController to expose /authenticate, generated JWT using JwtUtil.

**JwtUtil.java**

package com.example.jwtauth.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import java.util.Date;

import org.springframework.stereotype.Component;

@Component

public class JwtUtil {

private final String SECRET\_KEY = "mykey";

private final long EXPIRATION\_TIME = 1000 \* 60 \* 10;

public String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date())

.setExpiration(new Date(System.currentTimeMillis() + EXPIRATION\_TIME))

.signWith(SignatureAlgorithm.HS256, SECRET\_KEY)

.compact();

}

}

**AuthController.java**

package com.example.jwtauth.controller;

import com.example.jwtauth.util.JwtUtil;

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

import org.springframework.http.ResponseEntity;

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

import java.util.Base64;

import javax.servlet.http.HttpServletRequest;

@RestController

public class AuthController {

@Autowired

private JwtUtil jwtUtil;

@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()).trim();

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

String credentials = new String(credDecoded);

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

String username = values[0];

String password = values[1];

// Basic hardcoded authentication

if ("user".equals(username) && "pwd".equals(password)) {

String token = jwtUtil.generateToken(username);

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

} else {

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

}

}

}

**SecurityConfig.java**

package com.example.jwtauth.config;

import org.springframework.context.annotation.Bean;

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

import org.springframework.security.web.SecurityFilterChain;

import org.springframework.context.annotation.Configuration;

@Configuration

public class SecurityConfig {

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

.csrf().disable()

.authorizeHttpRequests()

.anyRequest().permitAll()

.and()

.httpBasic(); // to allow curl -u username:password

return http.build();

}

}

**JwtAuthApplication.java**

package com.example.jwtauth;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class JwtAuthApplication {

public static void main(String[] args) {

SpringApplication.run(JwtAuthApplication.class, args);

}

}

**OUTPUT:**

