**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. Refer details below:**

Method: GET

URL: /hello

Controller: com.cognizant.spring-learn.controller.HelloController

Method Signature: public String sayHello()

Method Implementation: return hard coded string "Hello World!!"

Sample Request: http://localhost:8083/hello

Sample Response: Hello World!!

IMPORTANT NOTE: Don't forget to include start and end log in the sayHello() method.

Try the URL http://localhost:8083/hello in both chrome browser and postman.

SME to explain the following aspects:

In network tab of developer tools show the HTTP header details received

In postman click on "Headers" tab to view the HTTP header details received

**Answer:**

**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()");

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

return "Hello World!!";

}

}

**SpringLearnApplication.java**

package com.cognizant.spring\_learn;

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);

}

}

**application.properties**

server.port=8083

**pom.xml**

<dependencies>

<!-- Spring Boot Starter Web -->

<dependency>

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

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

</dependency>

<!-- Logging with SLF4J -->

<dependency>

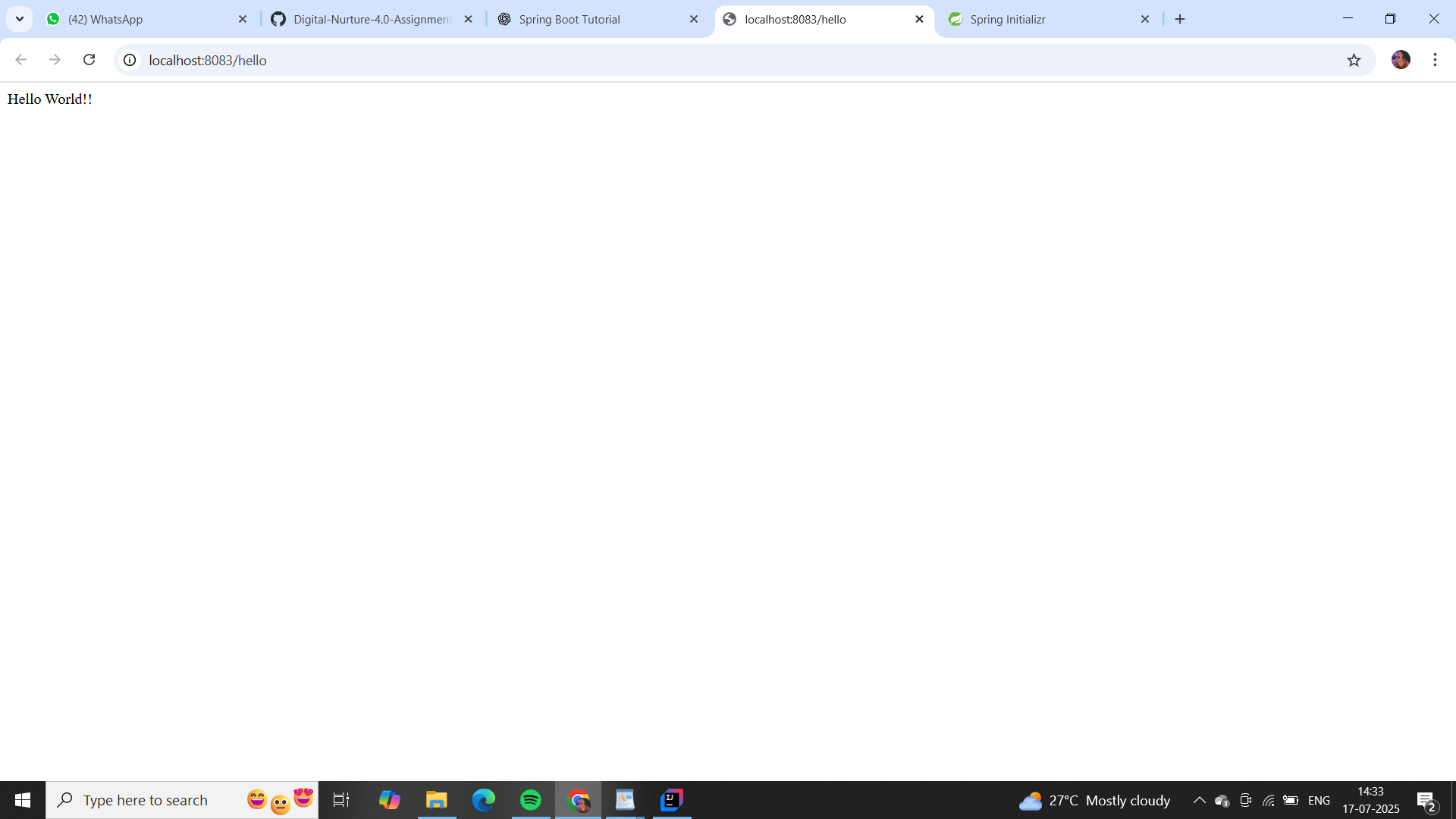
<groupId>org.springframework.boot</groupId>

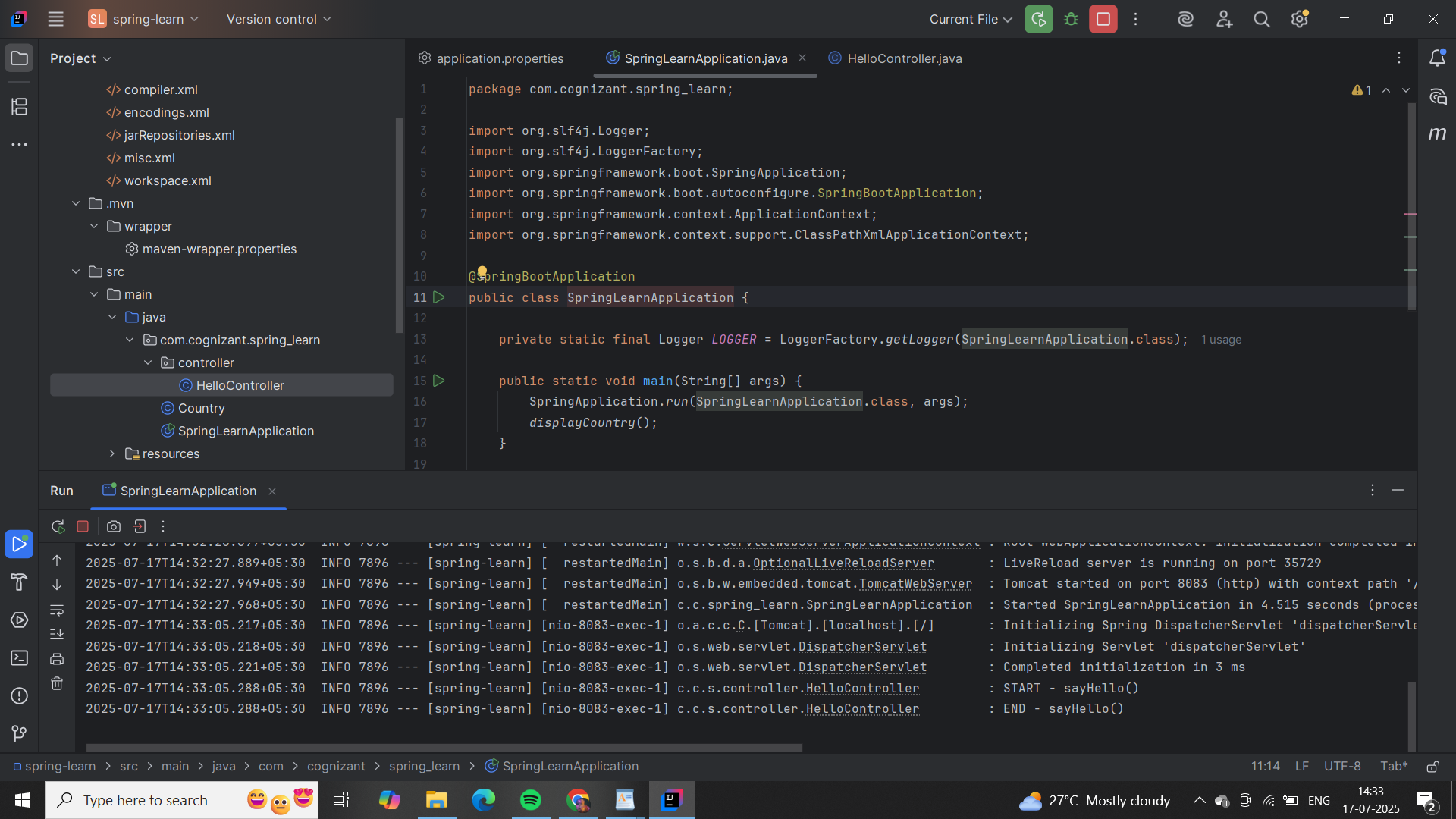
<artifactId>spring-boot-starter-logging</artifactId>

</dependency>

</dependencies>

**Output:**

****

****

**REST - Country Web Service**

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

URL: /country

Controller: com.cognizant.spring-learn.controller.CountryController

Method Annotation: @RequestMapping

Method Name: getCountryIndia()

Method Implementation: Load India bean from spring xml configuration and return

Sample Request: http://localhost:8083/country

Sample Response:

{

"code": "IN",

"name": "India"

}

SME to explain the following aspects:

What happens in the controller method?

How the bean is converted into JSON reponse?

In network tab of developer tools show the HTTP header details received

In postman click on "Headers" tab to view the HTTP header details received.

**Answer:**

**Country.java**

package com.cognizant.spring\_learn;

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 + '\'' +

'}';

}

}

**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.spring\_learn.Country">

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

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

</bean>

</beans>

**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 getCountryIndia");

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

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

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

LOGGER.info("END getCountryIndia");

return country;

}

}

**application.properties**

server.port=8083

**SpringLearnApplication.java**

package com.cognizant.spring\_learn;

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);

}

}

**pom.xml**

<dependencies>

<!-- Spring Boot Web -->

<dependency>

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

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

</dependency>

<!-- Spring Context (for XML config support) -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

</dependency>

<!-- Logging -->

<dependency>

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

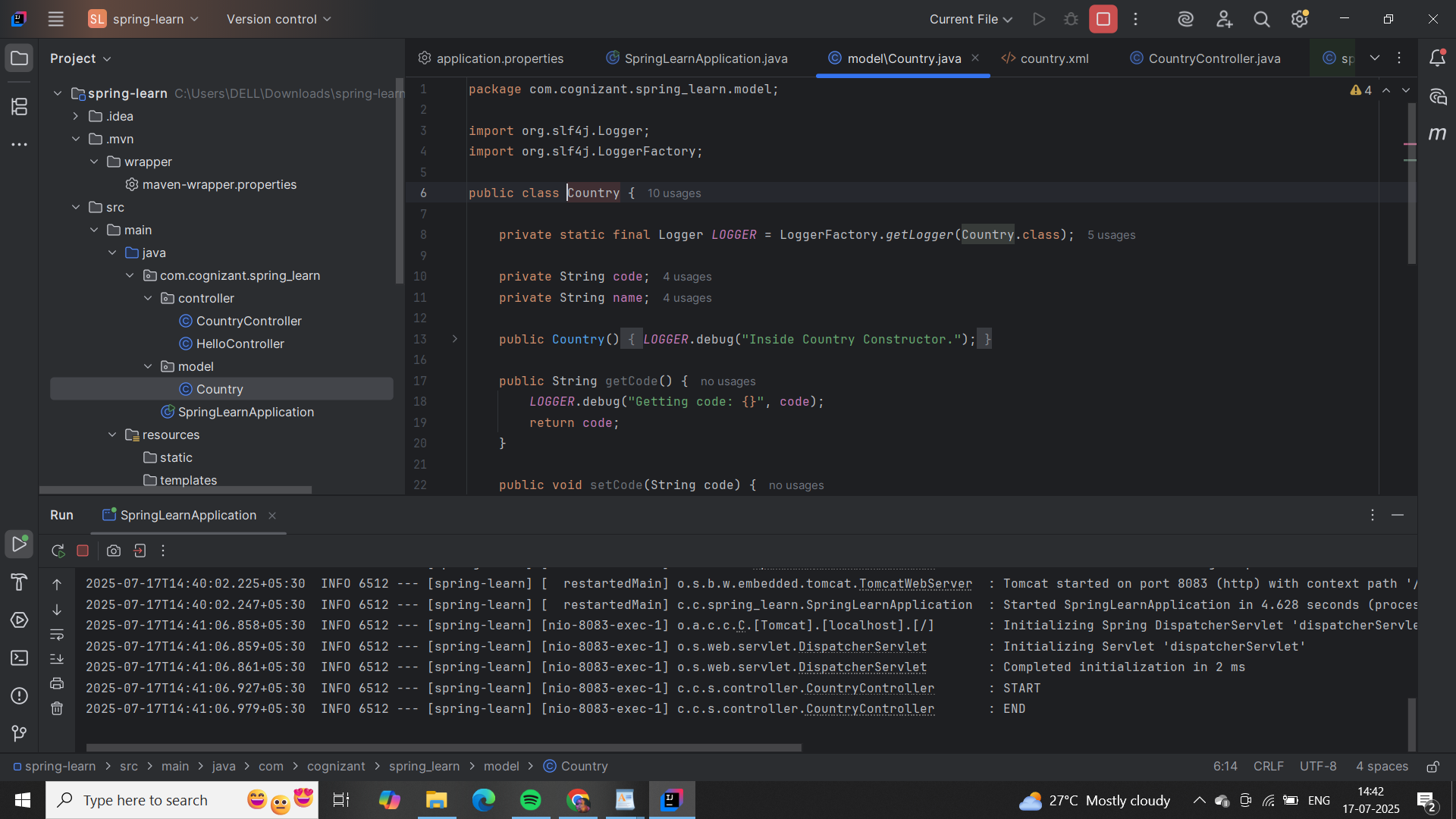
<artifactId>spring-boot-starter-logging</artifactId>

</dependency>

</dependencies>

**Output:**

****

****

**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.**

Controller: com.cognizant.spring-learn.controller.CountryController

Method Annotation: @GetMapping("/countries/{code}")

Method Name: getCountry(String code)

Method Implemetation: Invoke countryService.getCountry(code)

Service Method: com.cognizant.spring-learn.service.CountryService.getCountry(String code)

Service Method Implementation:

Get the country code using @PathVariable

Get country list from country.xml

Iterate through the country list

Make a case insensitive matching of country code and return the country.

Lambda expression can also be used instead of iterating the country list

Sample Request: http://localhost:8083/country/in

Sample Response:

{

"code": "IN",

"name": "India"

}

**Answer:**

**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"

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

xsi:schemaLocation="

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

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

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

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

<!-- Individual Country Beans -->

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

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

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

</bean>

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

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

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

</bean>

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

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

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

</bean>

<!-- List of Country Beans -->

<util:list id="countryList" value-type="com.cognizant.spring\_learn.model.Country">

<ref bean="in"/>

<ref bean="us"/>

<ref bean="fr"/>

</util:list>

</beans>

**CountryService.java**

package com.cognizant.spring\_learn.service;

import com.cognizant.spring\_learn.model.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 = (List<Country>) context.getBean("countryList");

return countryList.stream()

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

.findFirst()

.orElse(null);

}

}

**CountryController.java**

package com.cognizant.spring\_learn.controller;

import com.cognizant.spring\_learn.model.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) {

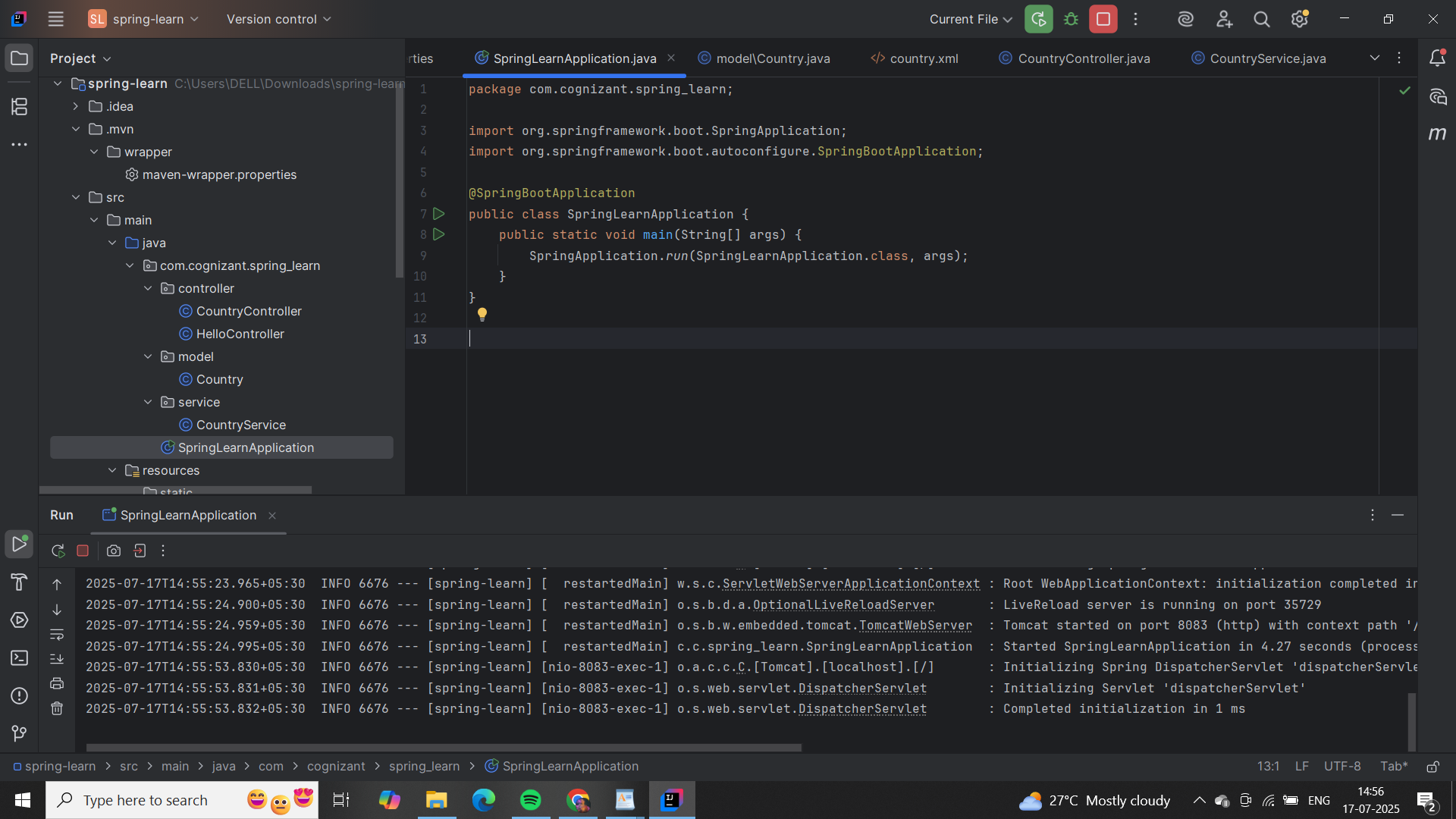
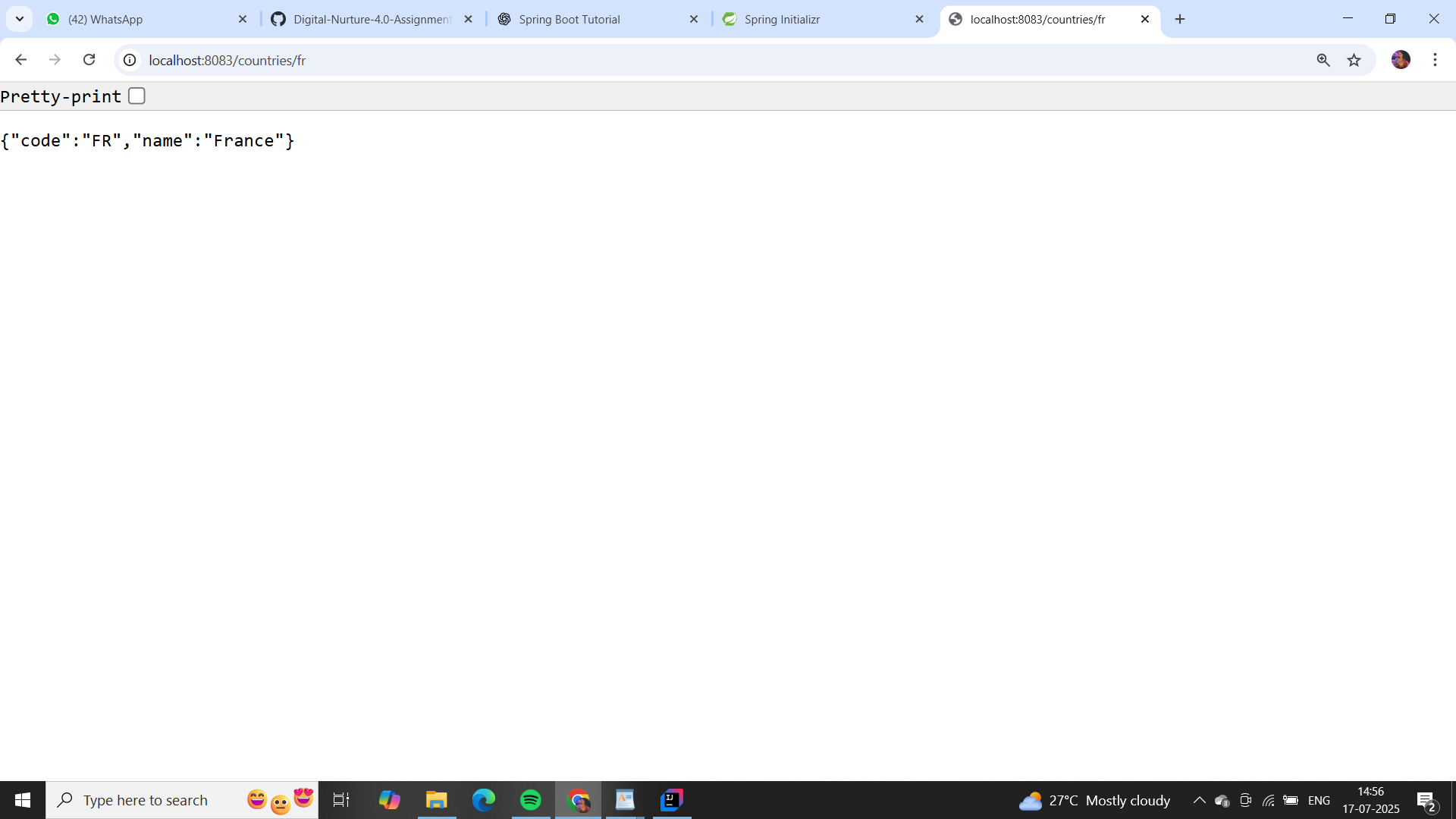
return countryService.getCountry(code);

}

}

**URL:** <http://localhost:8083/countries/fr>

**Output:**

****

**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.**

Ideally when the below curl command is executed that calls the new authentication service, the token should be responded. Kindly note that the credentials are passed using -u option.

Request

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

Response

{"token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNTcwMzc5NDc0LCJleHAiOjE1NzAzODA2NzR9.t3LRvlCV-hwKfoqZYlaVQqEUiBloWcWn0ft3tgv0dL0"}

This can be incorporated as three major steps:

Create authentication controller and configure it in SecurityConfig

Read Authorization header and decode the username and password

Generate token based on the user retrieved in the previous step

Let incorporate the above as separate hands on exercises.

**Answer:**

AuthenticationController.java

package com.cognizant.spring\_learn.controller;

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

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.Authentication;

import org.springframework.security.core.AuthenticationException;

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

import com.cognizant.spring\_learn.util.JwtUtil;

import java.util.Base64;

import java.util.Map;

import java.util.HashMap;

@RestController

public class AuthenticationController {

@Autowired

private AuthenticationManager authenticationManager;

@Autowired

private JwtUtil jwtUtil;

@RequestMapping(value = "/authenticate", method = RequestMethod.GET)

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

try {

// Extract credentials

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

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

String credentials = new String(decoded);

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

String username = parts[0];

String password = parts[1];

// Authenticate user

Authentication authentication = authenticationManager.authenticate(

new UsernamePasswordAuthenticationToken(username, password)

);

// Generate JWT

String token = jwtUtil.generateToken(authentication);

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

response.put("token", token);

return response;

} catch (AuthenticationException e) {

throw new RuntimeException("Invalid credentials");

}

}

}

**JwtUtil.java**

package com.cognizant.spring\_learn.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import org.springframework.security.core.Authentication;

import org.springframework.stereotype.Component;

import java.util.Date;

@Component

public class JwtUtil {

private final String SECRET\_KEY = "secret";

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

public String generateToken(Authentication auth) {

return Jwts.builder()

.setSubject(auth.getName())

.setIssuedAt(new Date())

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

.signWith(SignatureAlgorithm.HS256, SECRET\_KEY)

.compact();

}

}

**SecurityConfig.java**

package com.cognizant.spring\_learn.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

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

import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

@Configuration

@EnableWebSecurity

public class SecurityConfig extends WebSecurityConfigurerAdapter {

@Override

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.inMemoryAuthentication()

.withUser("user").password("{noop}pwd").roles("USER");

}

@Override

@Bean

public AuthenticationManager authenticationManagerBean() throws Exception {

return super.authenticationManagerBean();

}

}

**Output:**

