**WEEK-4**

**Spring REST using Spring Boot 3**

**Create a Spring Web Project using Maven** 

**CODE:**

* **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**);

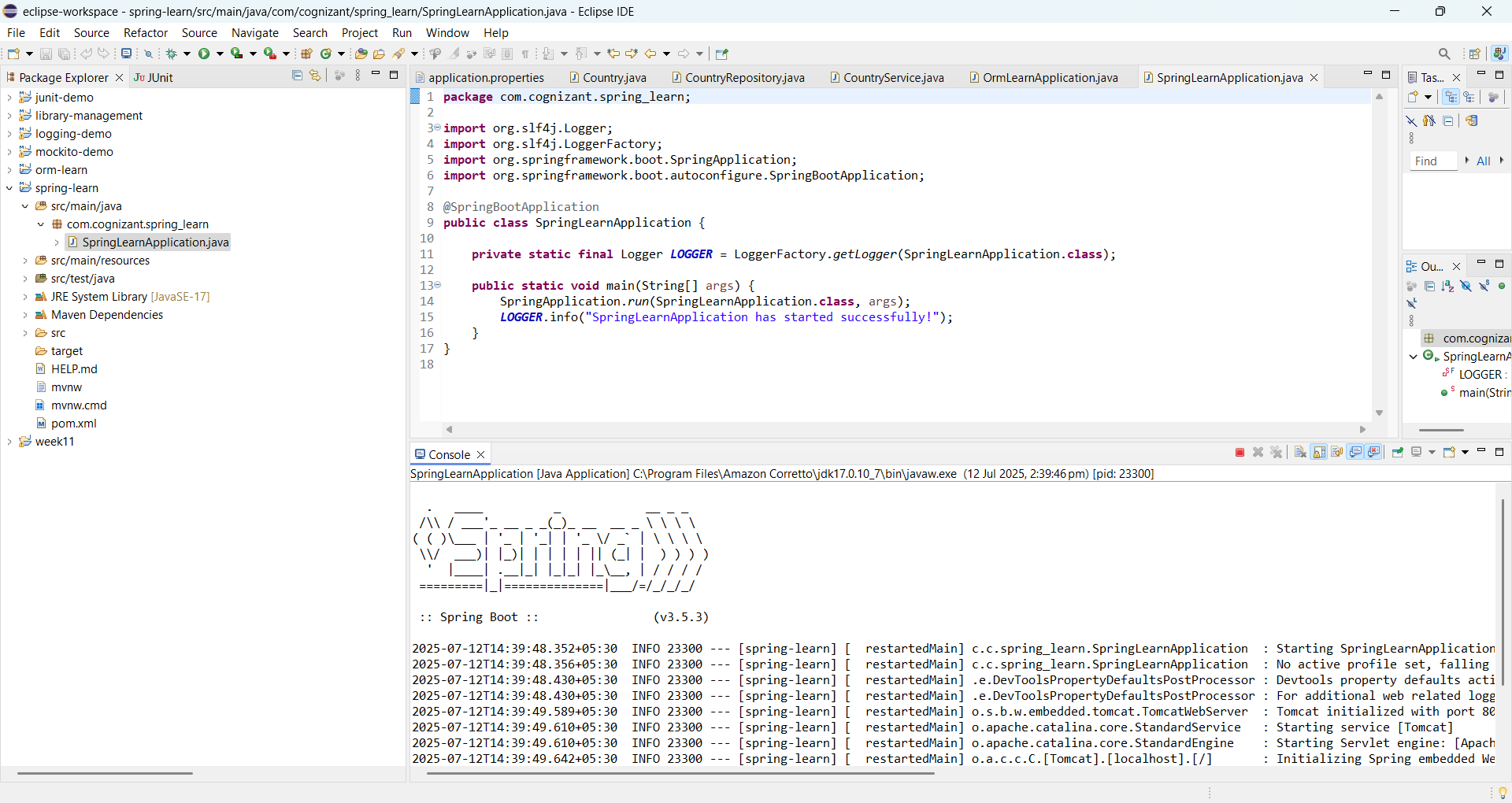
**public** **static** **void** main(String[] args) {

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

***LOGGER***.info("SpringLearnApplication has started successfully!");

}

**OUTPUT:**

}****

**Exercise-2**

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

**Scenario:**

An airlines website is going to support booking on four countries. There will be a drop down on the home page of this website to select the respective country. It is also important to store the two-character ISO code of each country.

**Code:**

**Country.java**

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.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.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);

displayCountry();

}

public static void displayCountry() {

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

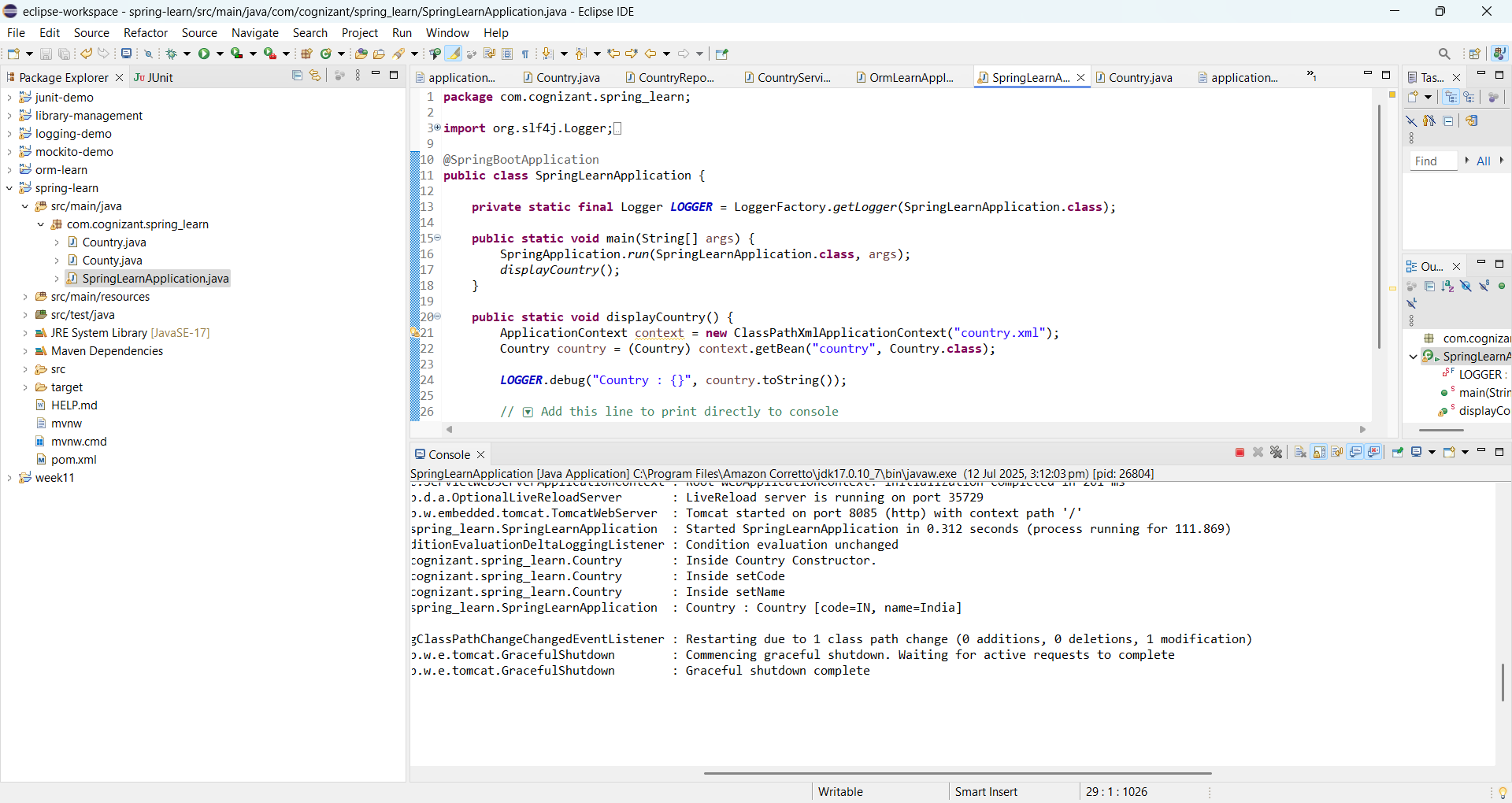
Country country = (Country) context.getBean("country", Country.class);

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

}

}

**Output:**

****

**SME:**

1. **What is ApplicationContext?**

* It's the central interface for accessing Spring beans.
* It manages lifecycle, instantiation, configuration, and wiring of beans.

1. **What is ClassPathXmlApplicationContext?**

* It loads the XML config file from the **classpath** (src/main/resources).
* Example:

new ClassPathXmlApplicationContext("country.xml");

1. **What Happens During context.getBean("country")?**
2. Spring reads country.xml.
3. It finds the bean with id country.
4. It creates an instance of Country.
5. It sets the properties using setters.
6. The bean is returned ready to use.

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

**Scenario:**

Write a REST service in the spring learn application created earlier, that returns the text "Hello World!!" using Spring Web Framework. Refer details below:

**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.debug("START: sayHello()");

String message = "Hello World!!";

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

return message;

}

}

**SpringLearnApplication.java**

@SpringBootApplication

public class SpringLearnApplication {

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

}

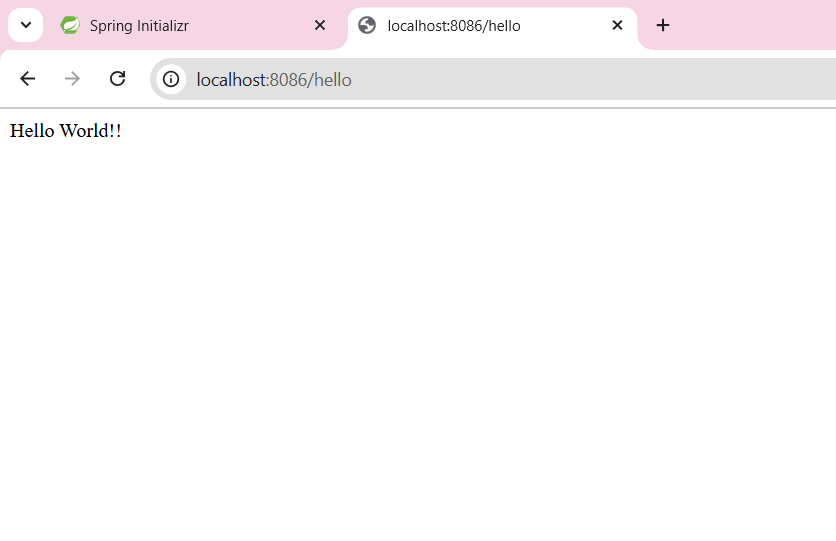
}

**application.properties**

server.port=8086

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

**Output:**

****

**Exercise-4**

**Scenario:**

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

**Code:**

**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

https://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.debug("START: getCountryIndia()");

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

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

LOGGER.debug("END: getCountryIndia()");

return country;

}

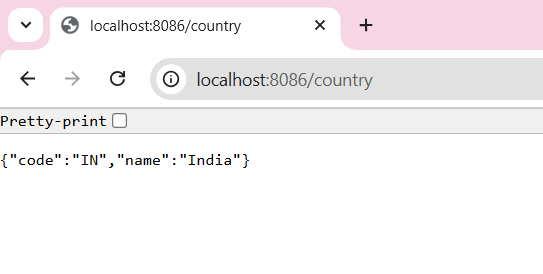
}

**application.properties**

server.port=8086

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

**Output:**



**Ecercise-5 REST - Get country based on country code** 

**Scenario:**

Write a REST service that returns a specific country based on country code. The country code should be case insensitive.

**Code:**

**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

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

<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>

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

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

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

</bean>

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

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

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

</bean>

</list>

</constructor-arg>

</bean></beans>

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

.findFirst()

.orElse(null);

}

}

**CountryController.java**

package com.cognizant.spring\_learn.controller;

import com.cognizant.spring\_learn.Country;

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

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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

@RestController

public class CountryController {

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

@Autowired

private CountryService countryService;

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

public Country getCountry(@PathVariable String code) {

LOGGER.debug("START: getCountry() with code = {}", code);

Country country = countryService.getCountry(code);

LOGGER.debug("END: getCountry() -> {}", country);

return country;

}

}

**Output:**



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

**Scenario:**

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.

**Code:**

**AuthController.java**

package com.cognizant.spring\_learn.controller;

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

import org.springframework.http.ResponseEntity;

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

import java.util.Base64;

@RestController

public class AuthController {

@GetMapping("/authenticate")

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

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 credentials = new String(decodedBytes);

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

String username = parts[0];

String password = parts[1];

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");

}

}}

**JwtUtil.java**

package com.cognizant.spring\_learn.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

import java.security.Key;

import java.util.Date;

public class JwtUtil {

private static final Key key = Keys.secretKeyFor(SignatureAlgorithm.HS256);

public static String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

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

.setExpiration(new Date(System.currentTimeMillis() + 600000)) // 10 mins

.signWith(key)

.compact();

}

}

**SecurityConfig.java**

package com.cognizant.spring\_learn.security;

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.builders.HttpSecurity;

import org.springframework.security.config.Customizer;

import org.springframework.security.core.userdetails.User;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.security.web.SecurityFilterChain;

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

@Configuration

public class SecurityConfig {

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

.csrf(csrf -> csrf.disable())

.authorizeHttpRequests(auth ->

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

.anyRequest().authenticated()

)

.httpBasic(Customizer.withDefaults());

return http.build();

}

@Bean

public UserDetailsService userDetailsService() {

var user = User.withUsername("user")

.password("pwd")

.roles("USER")

.build();

return new InMemoryUserDetailsManager(user);

}

@SuppressWarnings("deprecation")

@Bean

public static NoOpPasswordEncoder passwordEncoder() {

return (NoOpPasswordEncoder) NoOpPasswordEncoder.getInstance();

}

@Bean

public AuthenticationManager authManager(HttpSecurity http, UserDetailsService userDetailsService)

throws Exception {

return http.getSharedObject(AuthenticationManagerBuilder.class)

.userDetailsService(userDetailsService)

.passwordEncoder(passwordEncoder())

.and()

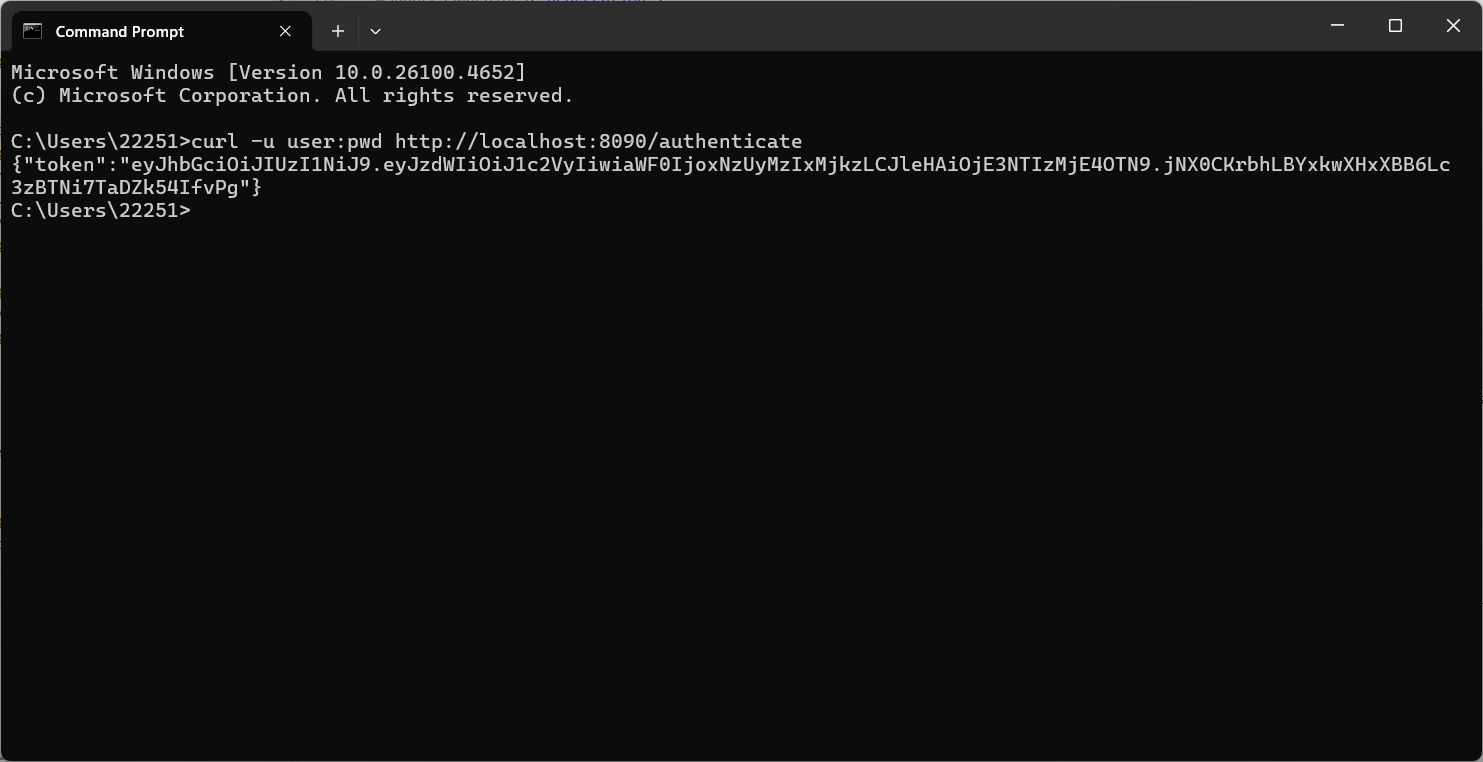
.build();

}}

**application.properties**

server.port=8090

**Output:**

****