**Week-4: Spring REST Using Spring Boot 3**

| **Hands-on-1: Create a Spring Web Project using Maven**  **Project Structure:**  **Code:**  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);  System.out.println("Application Started");  }  }  **Output**    **Hands-on 2: Spring Core – Load SimpleDateFormat from Spring Configuration XML**  **Project Structure:**    **Code:**  SpringLearnApplication.java  package com.cognizant.spring\_learn;  import java.text.SimpleDateFormat;  import java.util.Date;  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 {  public static void main(String[] args) throws Exception {  SpringApplication.run(SpringLearnApplication.class, args);  System.out.println("Application Started");  displayDate(); // Call the new method  }  public static void displayDate() throws Exception {  ApplicationContext context = new ClassPathXmlApplicationContext("date-format.xml");  SimpleDateFormat format = context.getBean("dateFormat", SimpleDateFormat.class);  Date date = format.parse("31/12/2018");  System.out.println("Parsed Date: " + date);  }  }  Date-format.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">

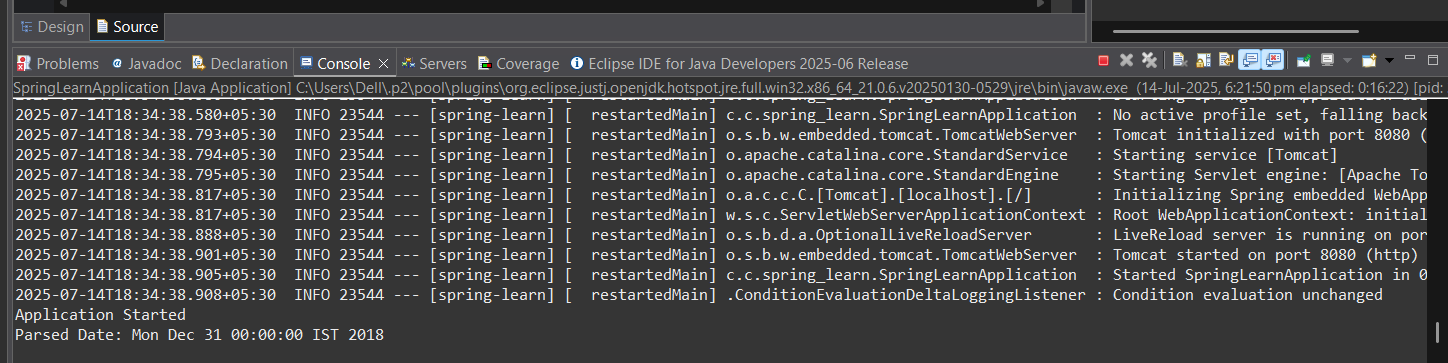
<bean id="dateFormat" class="java.text.SimpleDateFormat">

<constructor-arg value="dd/MM/yyyy" />

</bean>

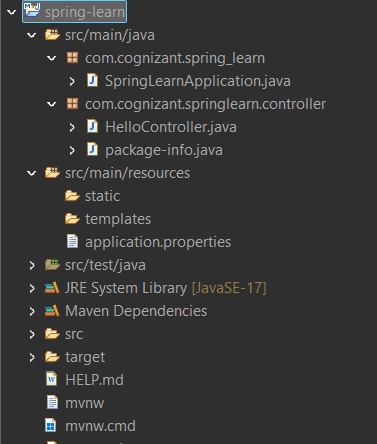
</beans>

**Output**

****

**Hands-on : Hello World RESTful Web Service**

**Project Structure:**

****

**Code:**

SpringLearnApplication.java

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication(scanBasePackages = "com.cognizant.springlearn")

public class SpringLearnApplication {

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

}

}

HelloController.java

package com.cognizant.springlearn.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 response = "Hello World!!";

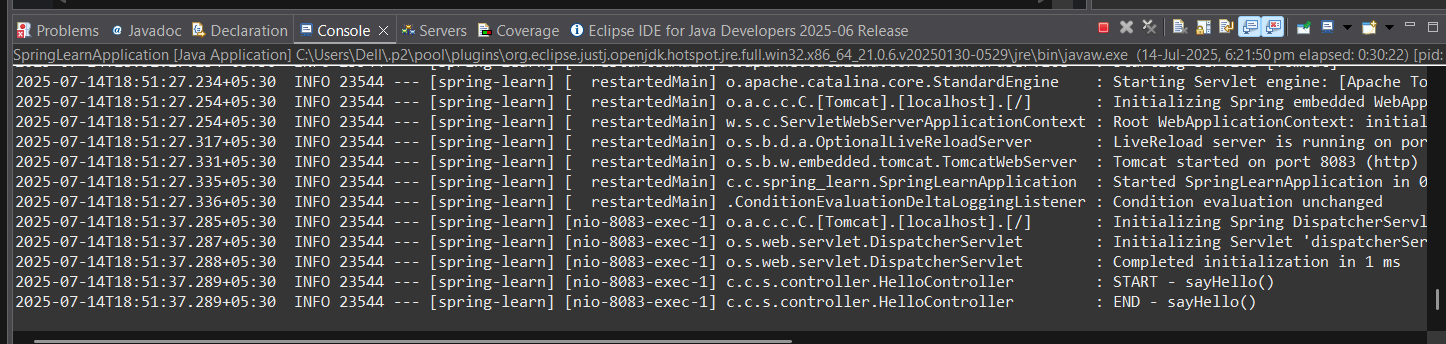
LOGGER.info("END - sayHello()");

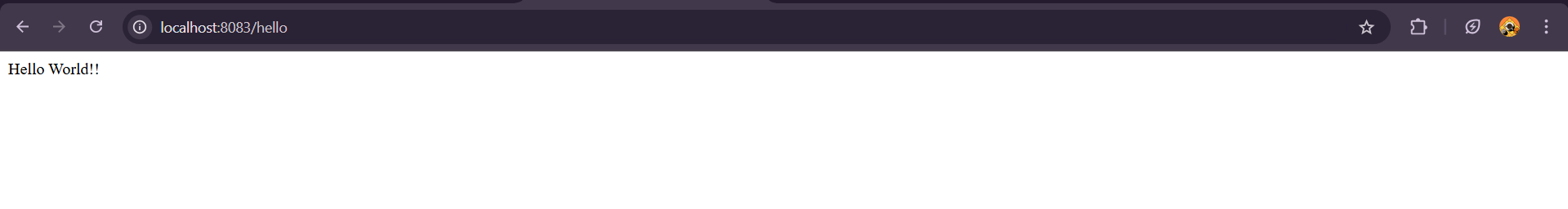
return response;

}

}

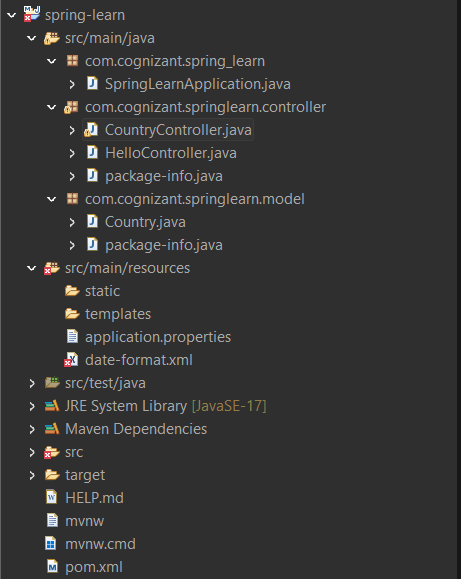
**Output**

****

****

**Hands-on : REST - Country Web Service**

**Project Structure:**

****

**Code:**

CountryController.java

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.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("date-format.xml");

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

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

return country;

}

}

Country.java

package com.cognizant.springlearn.model;

public class Country {

private String code;

private String name;

// Getters and setters

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;

}

}

Date-format.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="in" class="com.cognizant.springlearn.model.Country">

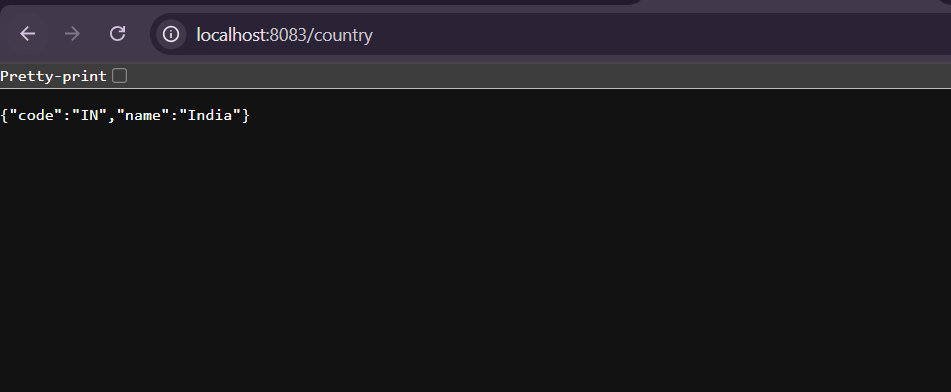
<property name="code" value="IN"/>

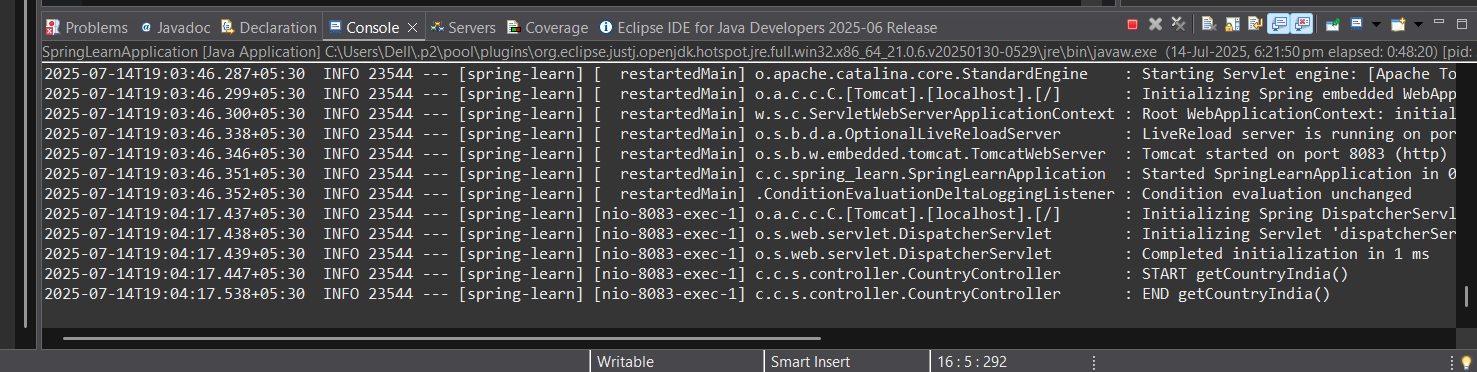
<property name="name" value="India"/>

</bean>

</beans>

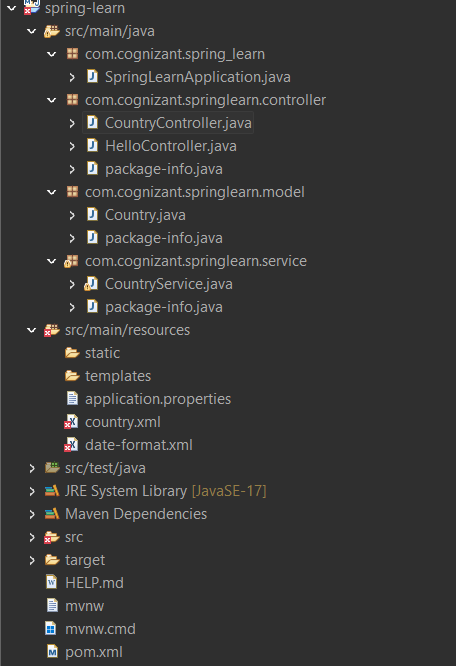
**Output**





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

**Project Structure:**



**Code:**

CountryController.java

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.model.Country;

import com.cognizant.springlearn.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.info("START getCountry() with code: {}", code);

Country country = countryService.getCountry(code);

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

return country;

}

}

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.cognizant.springlearn.model.Country">

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

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

</bean>

<bean class="com.cognizant.springlearn.model.Country">

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

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

</bean>

<bean class="com.cognizant.springlearn.model.Country">

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

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

</bean>

</list>

</constructor-arg>

</bean>

</beans>

CountryService.java

package com.cognizant.springlearn.service;

import com.cognizant.springlearn.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 = context.getBean("countryList", List.class);

return countryList.stream()

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

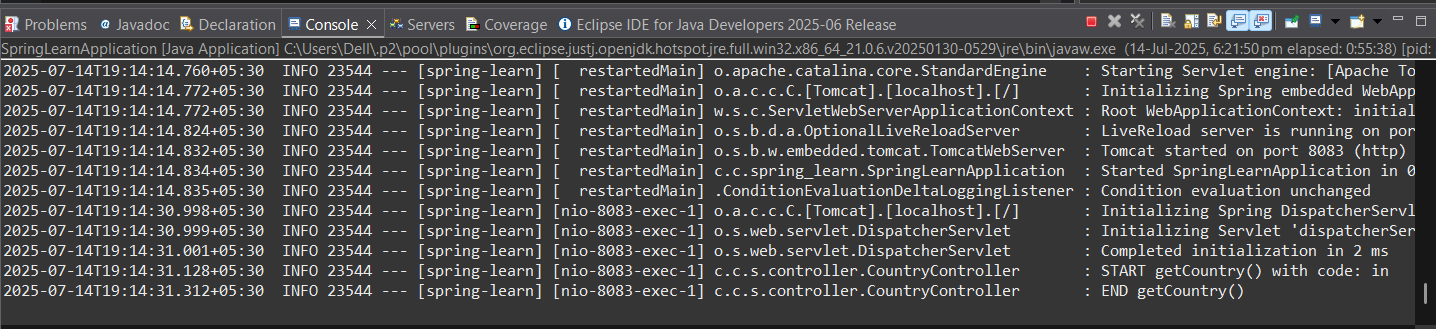
.findFirst()

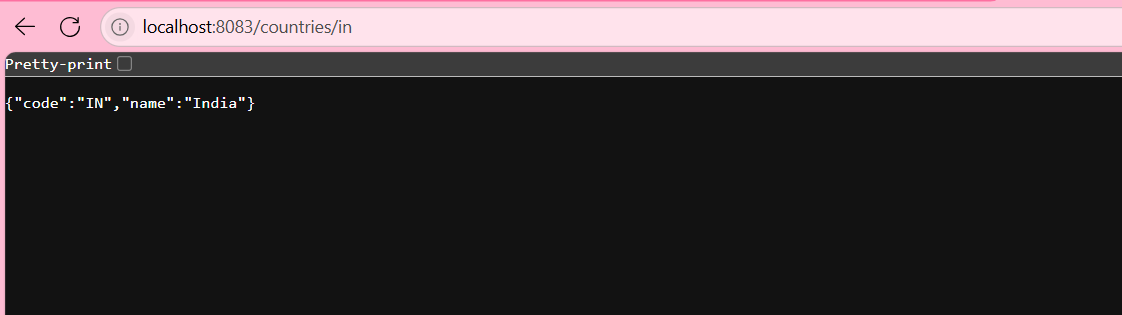
.orElse(null);

}

}

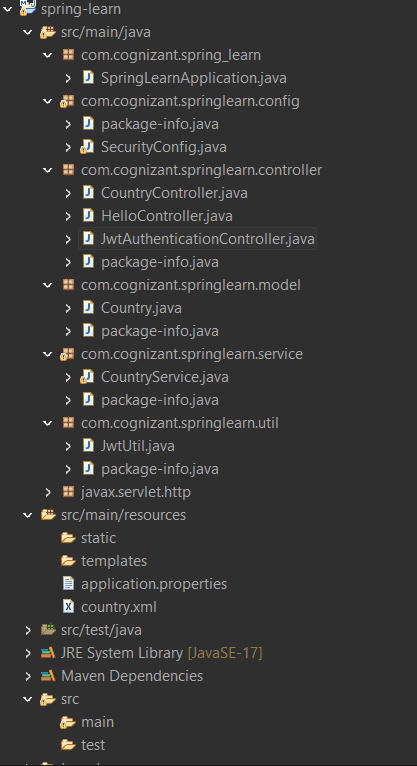
**Output**

****

****

**Hands-on : Create authentication service that returns JWT**

**Project Structure:**

****

**Code:**

JwtAuthenticationController.java

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.util.JwtUtil;

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

import org.springframework.security.core.Authentication;

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

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

@RestController

public class JwtAuthenticationController {

@Autowired

private JwtUtil jwtUtil;

@GetMapping("/authenticate")

public String generateToken(Authentication authentication) {

String username = authentication.getName(); // authenticated user

String token = jwtUtil.generateToken(username);

return "{\"token\":\"" + token + "\"}";

}

}

SecurityConfig.java

package com.cognizant.springlearn.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.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.bcrypt.BCryptPasswordEncoder;

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

@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(); // ENABLE Spring Basic Auth

return http.build();

}

@Bean

public UserDetailsService userDetailsService() {

return new InMemoryUserDetailsManager(

User.withUsername("user")

.password(passwordEncoder().encode("pwd"))

.roles("USER")

.build()

);

}

@Bean

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

}

JwtUtil.java

package com.cognizant.springlearn.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

import org.springframework.stereotype.Component;

import java.security.Key;

import java.util.Date;

@Component

public class JwtUtil {

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

private final long EXPIRATION\_TIME = 1000 \* 60 \* 60; // 1 hour

public String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date())

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

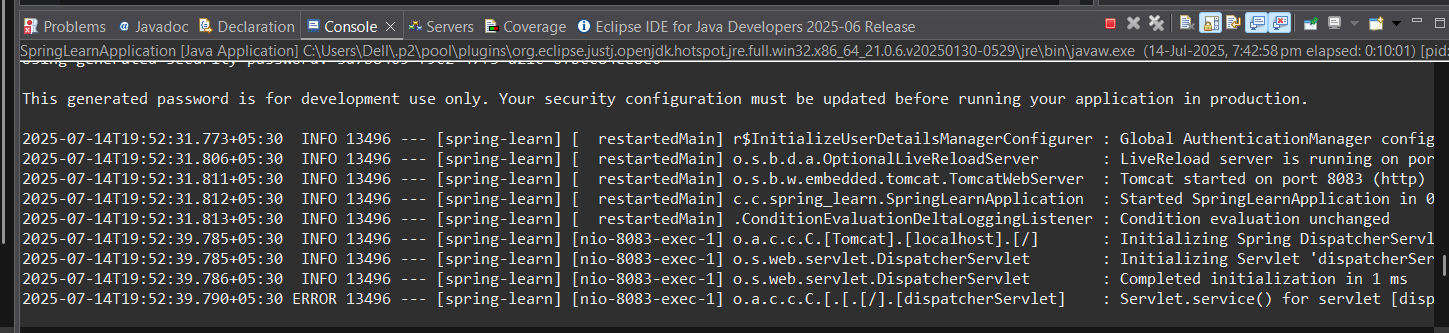
.signWith(key)

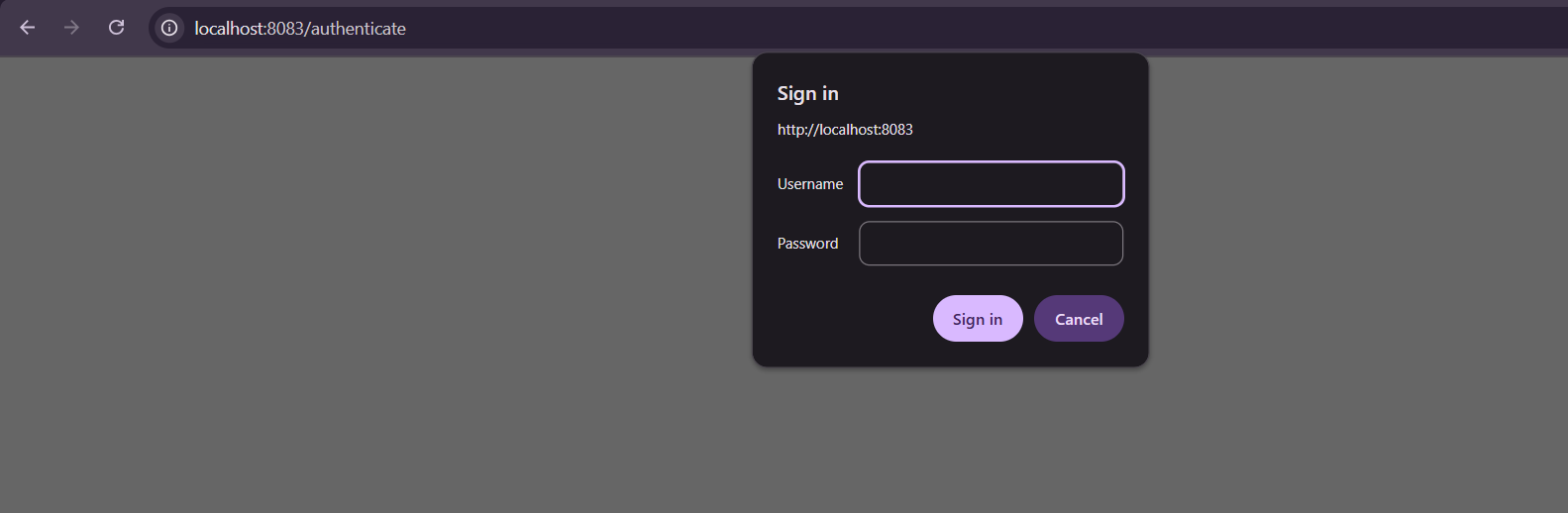
.compact();

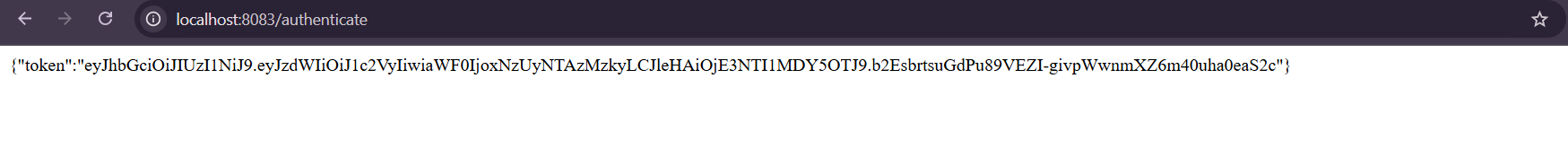
}

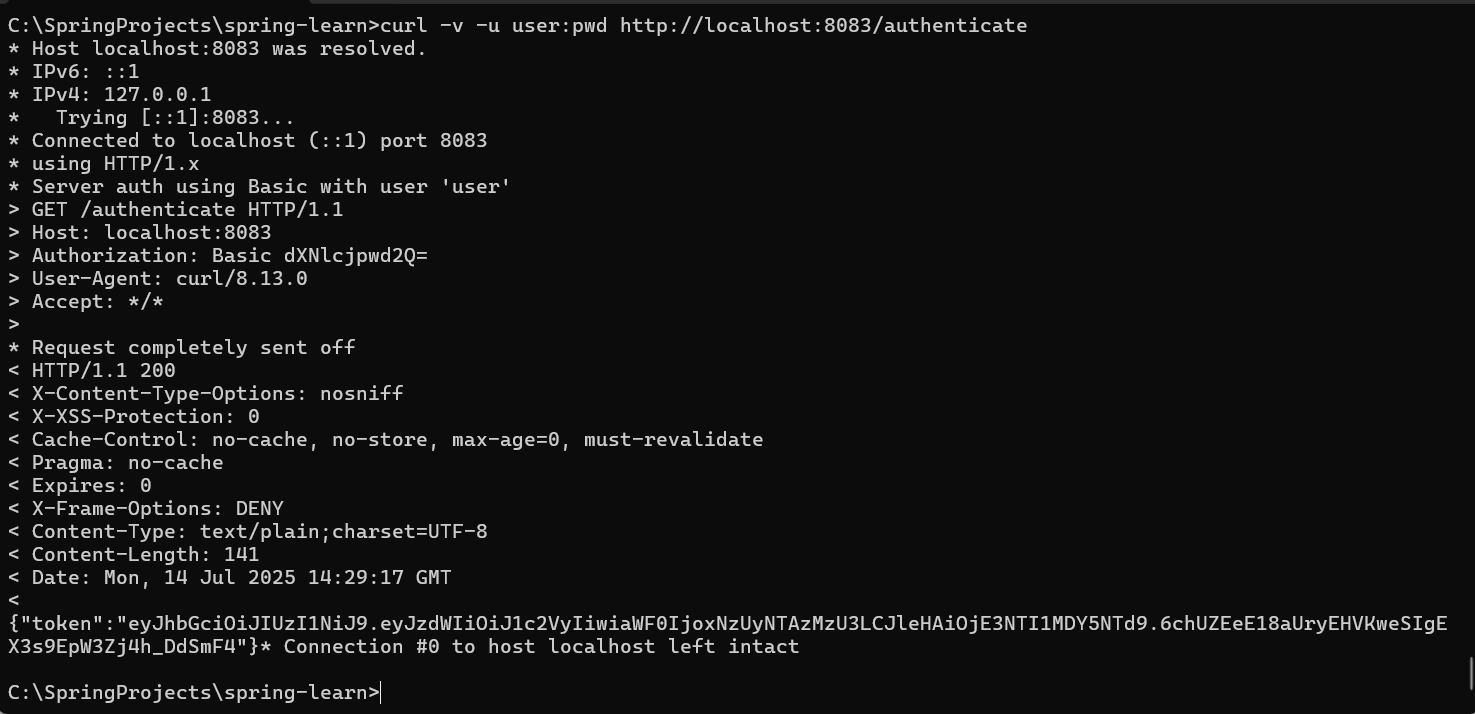
}

**Output**

****

****

****

****