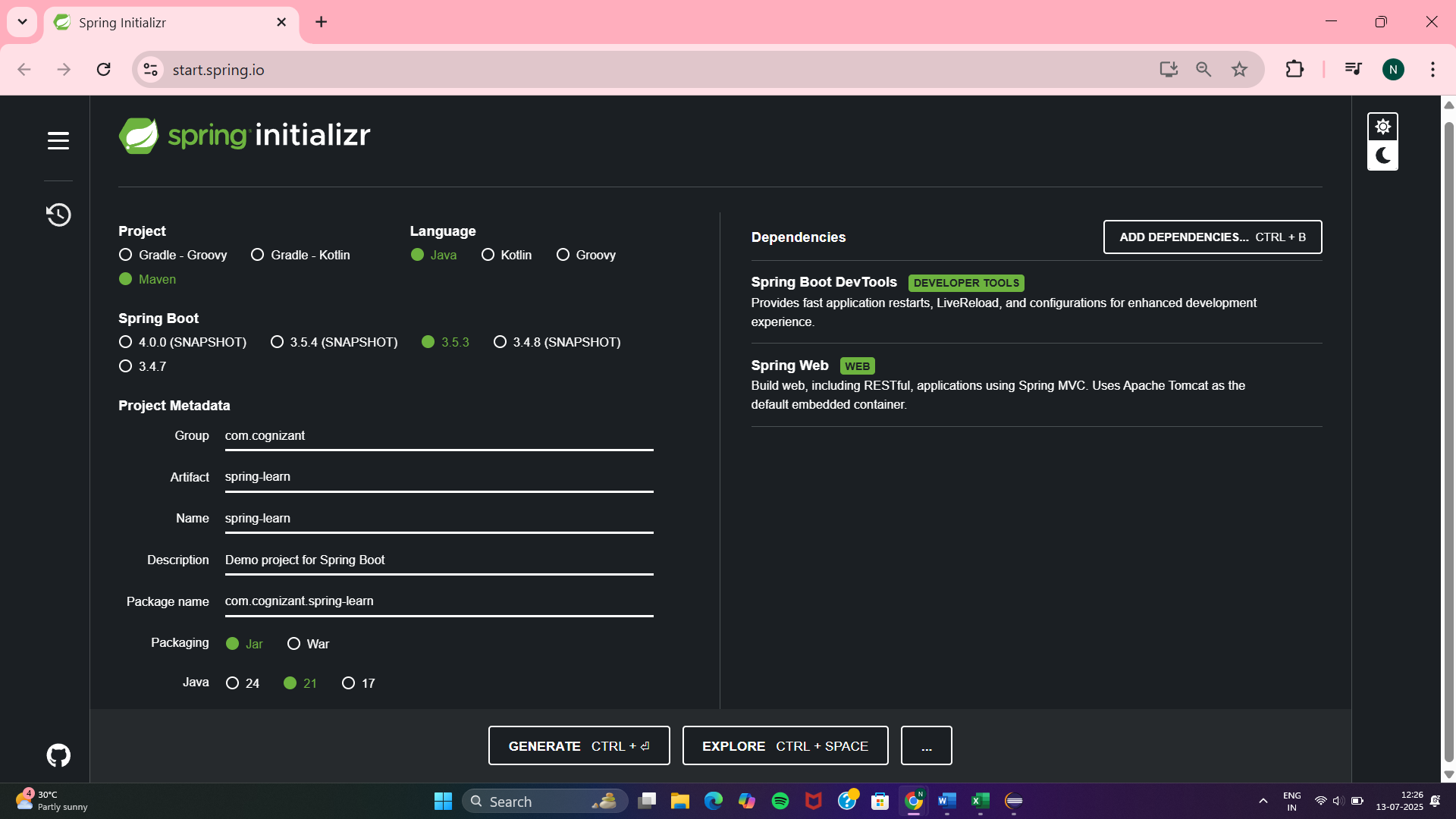
**WEEK-4 – Spring REST using Spring Boot 3**

**Hands on 1**

**Create a Spring Web Project using Maven**

**Customize Project**



**Build with Maven and Proxy**

mvn clean package \

-Dhttp.proxyHost=proxy.cognizant.com \

-Dhttp.proxyPort=6050 \

-Dhttps.proxyHost=proxy.cognizant.com \

-Dhttps.proxyPort=6050 \

-Dhttp.proxyUser=123456

**Application.properties**  
  
spring.application.name=spring-learn

# Server Port

server.port=9090

# Logging

logging.level.org.springframework=INFO

**HelloController.java**

package com.cognizant.spring\_learn.controller;

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

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

@RestController

public class HelloController {

@GetMapping("/")

public String hello() {

return "Welcome to Spring Learn!";

}

}

**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);

}

}

**Junit test case**

package com.cognizant.spring\_learn;

import org.junit.jupiter.api.Test;

import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest

class SpringLearnApplicationTests {

@Test

void contextLoads() {

}

}

**pom.xml**

<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>

<parent>

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

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

<version>3.2.0</version> <!-- or latest -->

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

</parent>

<dependencies>

<!-- Spring Web -->

<dependency>

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

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

</dependency>

<!-- DevTools -->

<dependency>

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

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

<scope>runtime</scope>

</dependency>

<!-- Testing -->

<dependency>

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

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

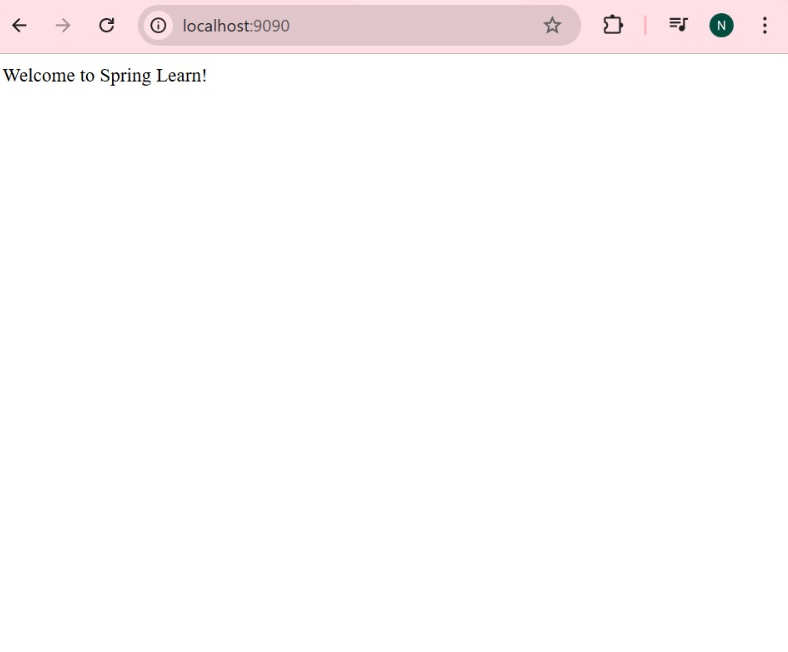
<scope>test</scope>

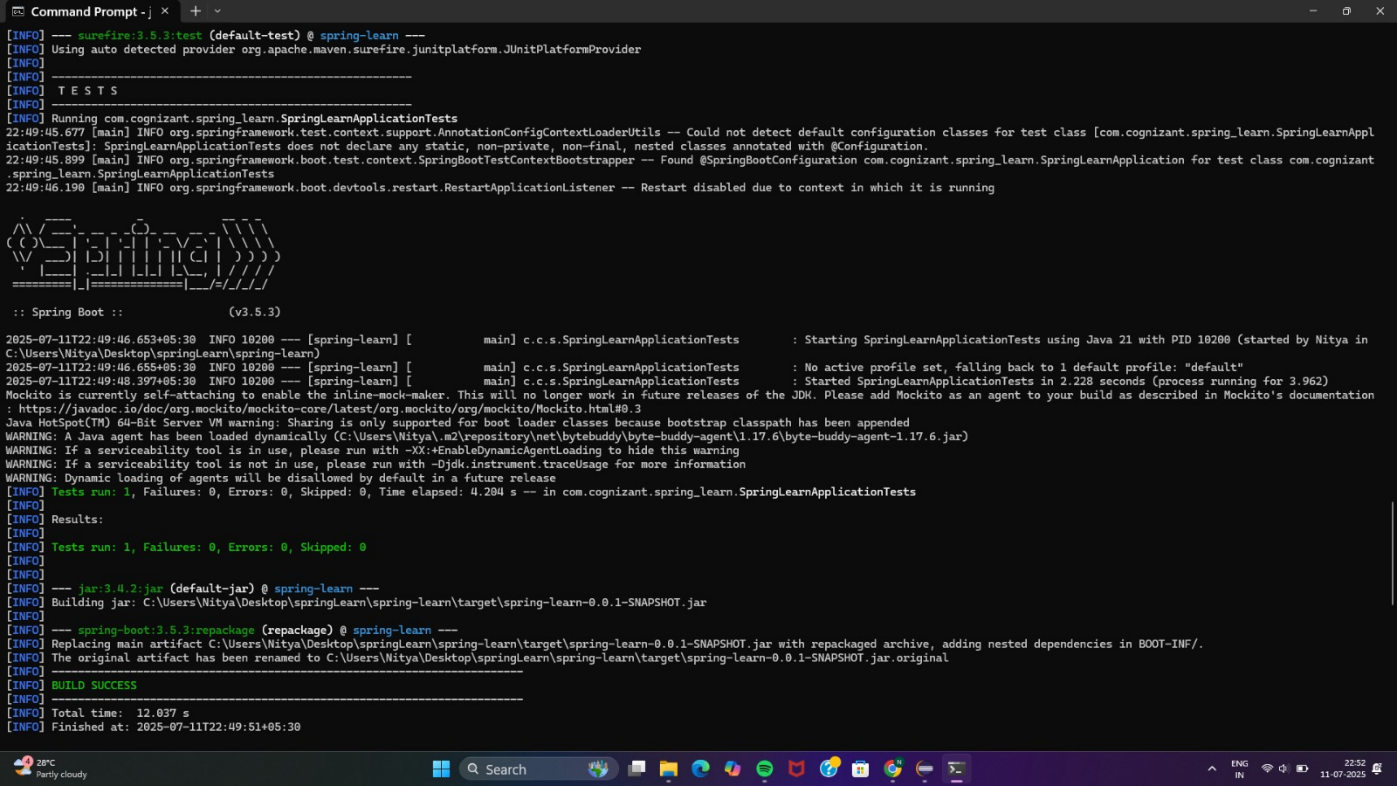
</dependency>

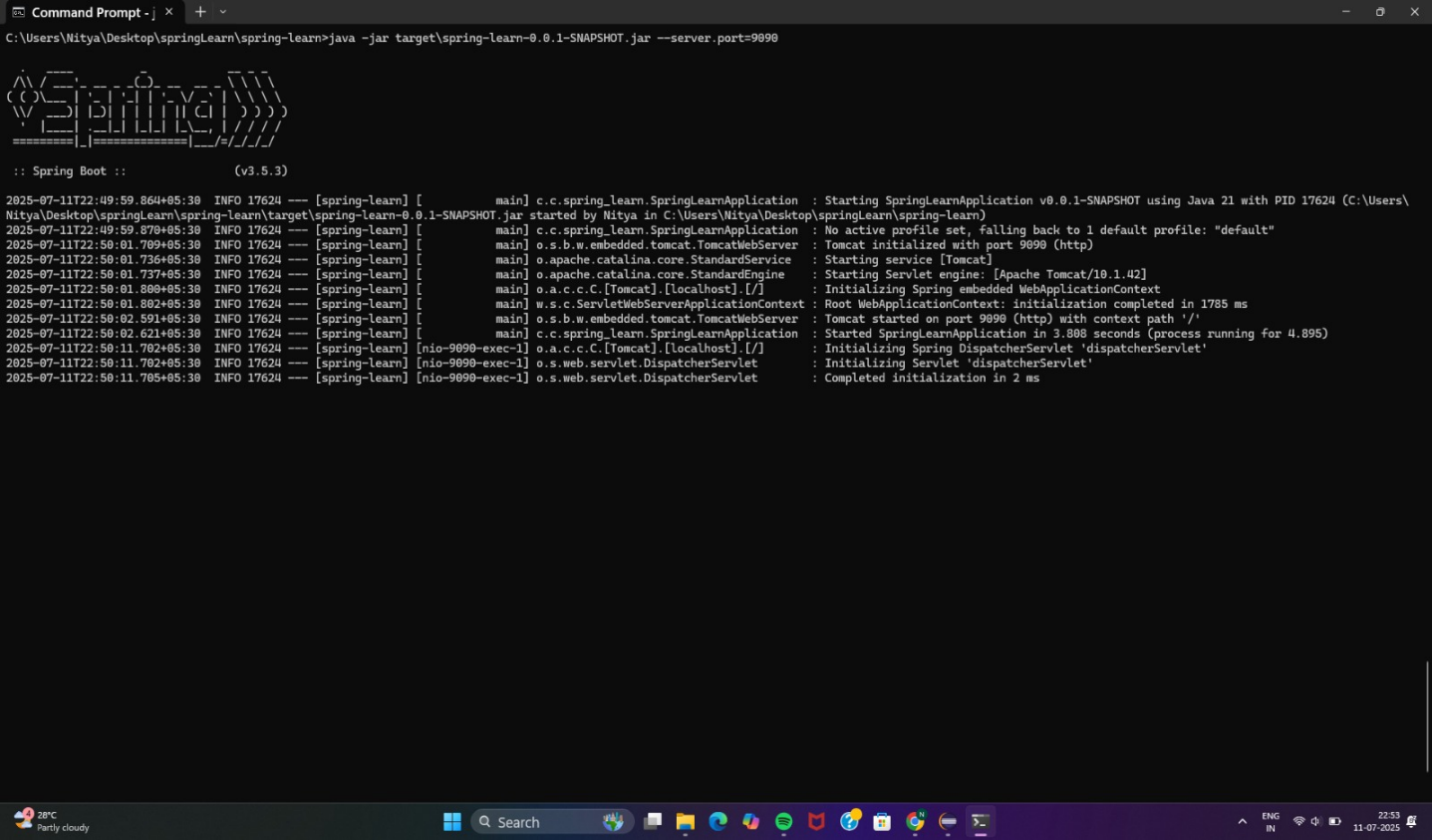
</dependencies>

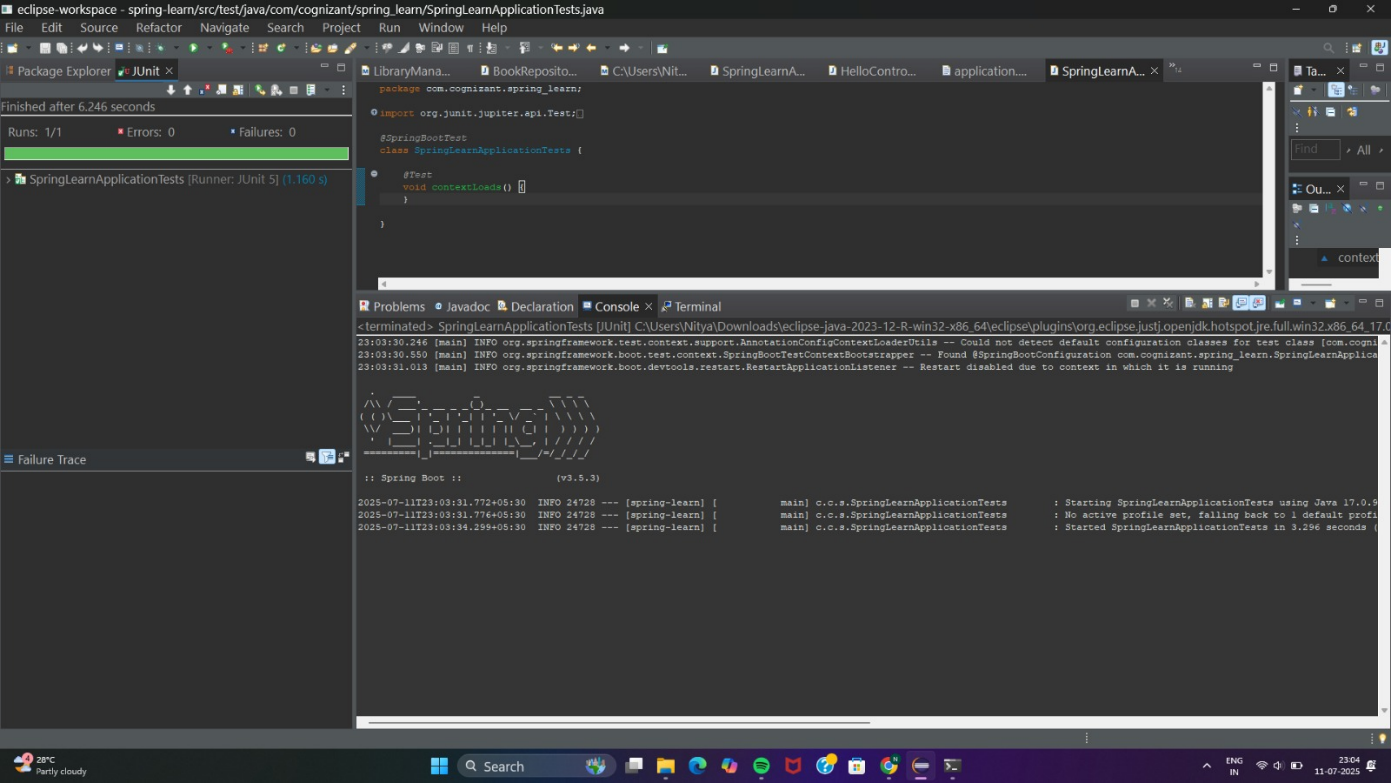
**OUTPUT**

<http://localhost:9090>







Junit test

**Hands on 2**

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

**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">

<!-- Define a reusable SimpleDateFormat bean -->

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

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

</bean>

</beans>

**SpringLearnApplication.java**

package com.cognizant.spring\_learn;

import java.text.ParseException;

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) {

SpringApplication.run(SpringLearnApplication.class, args);

displayDate();

}

public static void displayDate() {

// Load Spring XML Configuration

ApplicationContext context = new ClassPathXmlApplicationContext("date-format.xml");

// Retrieve the SimpleDateFormat bean

SimpleDateFormat format = context.getBean("dateFormat", SimpleDateFormat.class);

try {

// Parse the string into a Date object

Date date = format.parse("31/12/2018");

System.out.println("Parsed Date: " + date);

} catch (ParseException e) {

System.err.println("Failed to parse date: " + e.getMessage());

}

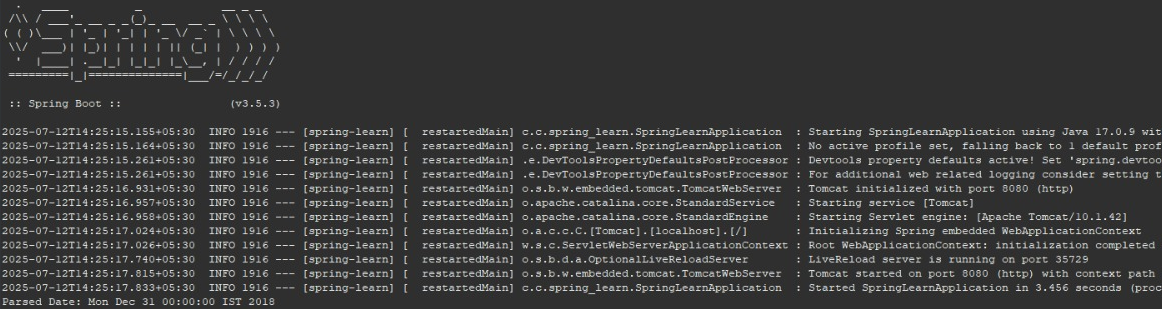
// Release resources held by the context

((ClassPathXmlApplicationContext) context).close();

}

}

**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.spring\_learn.model.Country">

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

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

</bean>

</beans>

**Country.java**

package com.cognizant.spring\_learn.model;

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("Getting country code: {}", code);

return code;

}

public void setCode(String code) {

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

this.code = code;

}

public String getName() {

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

return name;

}

public void setName(String name) {

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

this.name = name;

}

@Override

public String toString() {

return "Country { code = '" + code + "', name = '" + name + "' }";

}

}

**SpringlearnApplication.java**

package com.cognizant.spring\_learn;

import com.cognizant.spring\_learn.model.Country;

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() {

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

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

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

}

}

**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>3.2.3</version>

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

</parent>

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-learn</name>

<description>Spring Core XML Configuration Hands-on</description>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

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

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

</dependency>

<!-- Logging with SLF4J + Logback (default in Spring Boot) -->

<dependency>

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

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

</dependency>66

<!-- Optional: For Testing -->

<dependency>

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

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

<scope>test</scope>

**OUTPUT**

org.springframework.context.support.ClassPathXmlApplicationContext prepareRefresh

INFO: Refreshing org.springframework.context.support.ClassPathXmlApplicationContext@6bdf28bb

org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions

INFO: Loading XML bean definitions from class path resource [country.xml]

DEBUG com.cognizant.springlearn.Country - Inside Country Constructor.

DEBUG com.cognizant.springlearn.Country - Setter for code called: IN

DEBUG com.cognizant.springlearn.Country - Setter for name called: India

DEBUG com.cognizant.springlearn.SpringLearnApplication - Country : Country{code='IN', name='India'}

**2.Spring REST using Spring Boot3**

**1.Hello World RESTful Web Service**

**Application.properties**  
server.port=8083

spring.application.name=spring-learn

**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()");

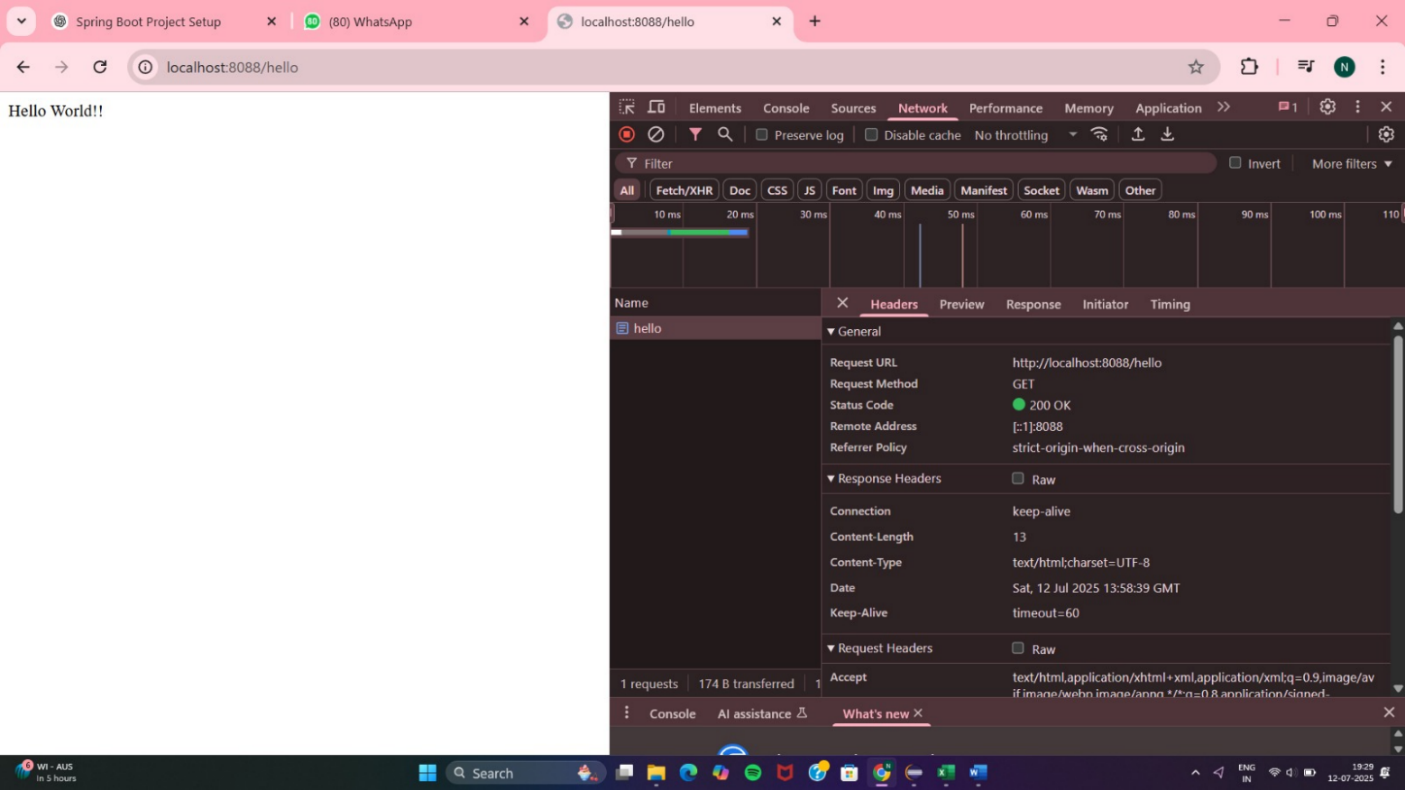
return message;

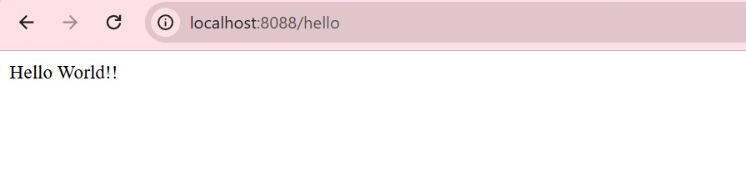
}

}

**Chrome:**

http://localhost:8088/hello





**2.REST - Country Web Service**

**src/main/java/com/cognizant/spring\_learn/model/Country.java**

package com.cognizant.spring\_learn.model;

public class Country {

private String code;

private String name;

public Country() {}

public Country(String code, String name) {

this.code = code;

this.name = 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;

}

}

**src/main/resources/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="in" class="com.cognizant.spring\_learn.model.Country">

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

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

</bean>

</beans>

**src/main/java/com/cognizant/spring\_learn/controller/CountryController.java**

package com.cognizant.spring\_learn.controller;

import com.cognizant.spring\_learn.model.Country;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.xml.XmlBeanFactory;

import org.springframework.context.support.ClassPathResource;

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()");

XmlBeanFactory factory = new XmlBeanFactory(new ClassPathResource("country.xml"));

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

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

return country;

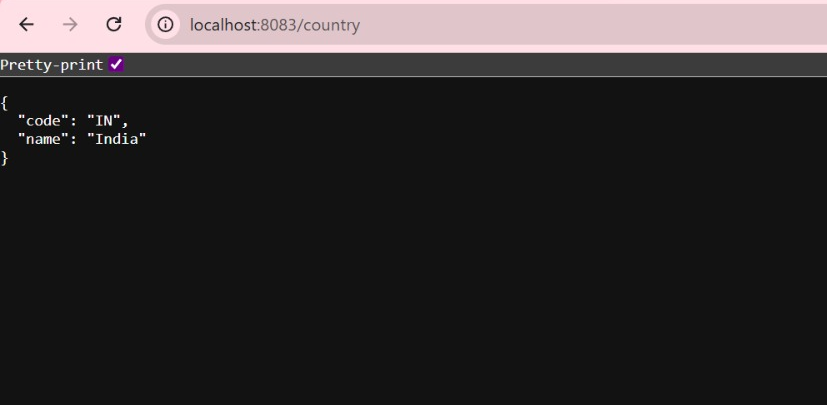
}

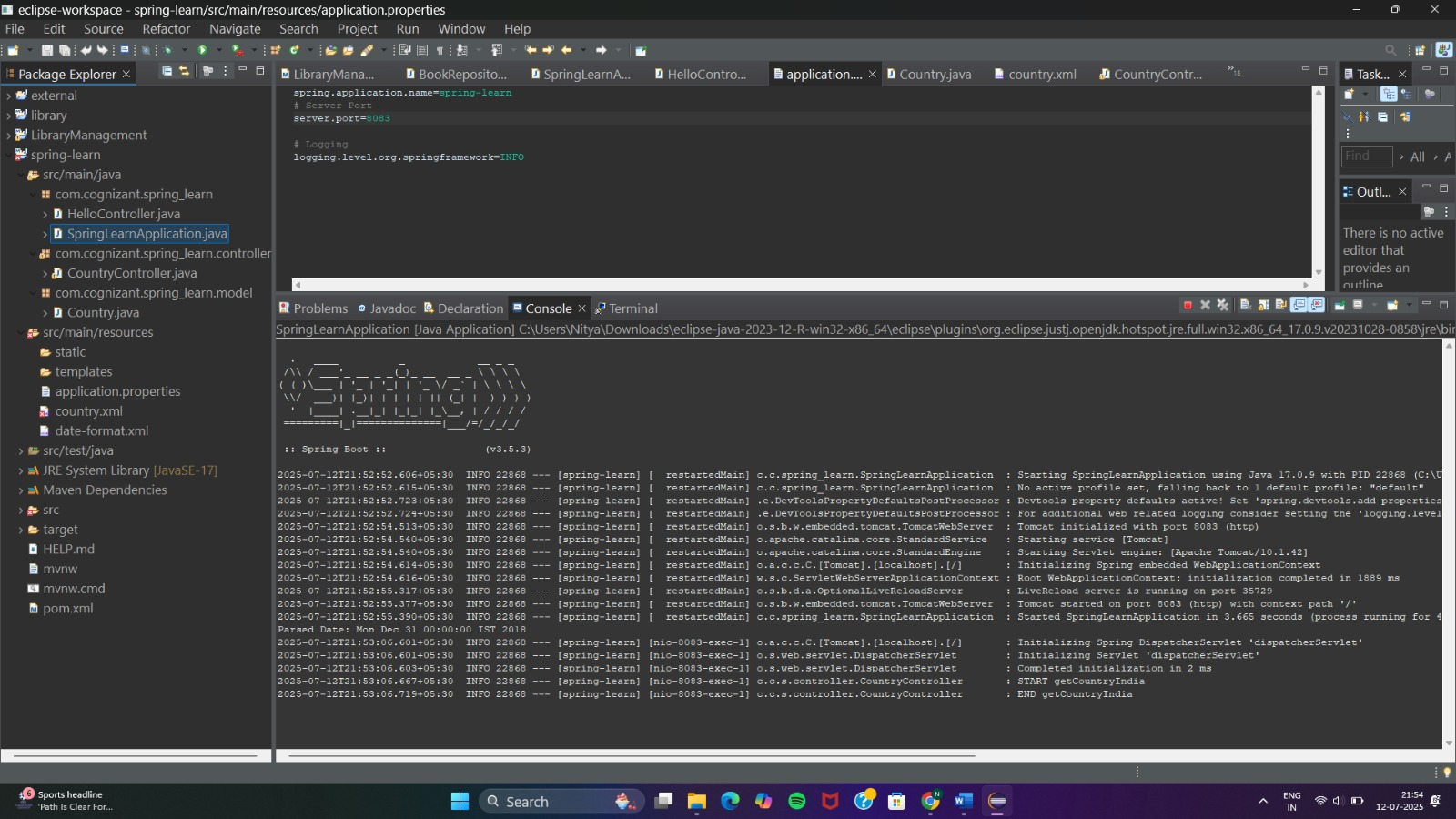
}

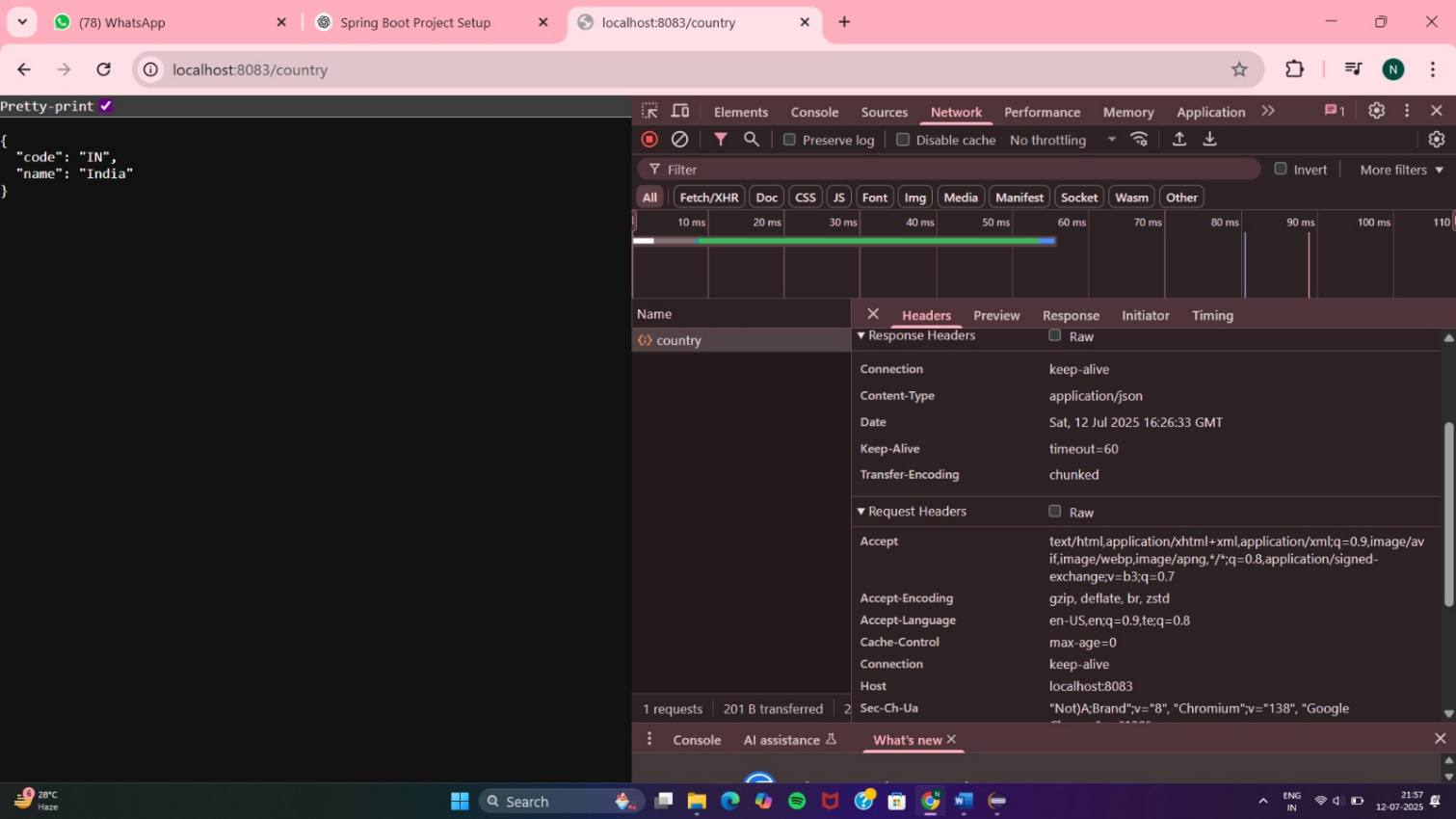
**Application.properties**

server.port=8083

<http://localhost:8083/country>







**3.REST - Get country based on country code**

**Country.java**

package com.cognizant.spring\_learn.model;

public class Country {

private String code;

private String name;

public Country() {

}

public Country(String code, String name) {

this.code = code.toUpperCase();

this.name = name;

}

public String getCode() {

return code;

}

public void setCode(String code) {

this.code = code.toUpperCase(); // Normalize case

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

**Country.xml**

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

<beans xmlns="http://www.springframework.org/schema/beans"

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

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.model.Country">

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

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

</bean>

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

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

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

</bean>

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

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

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

return countries.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.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;

}

}

**SpringLearnApplication.java**

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

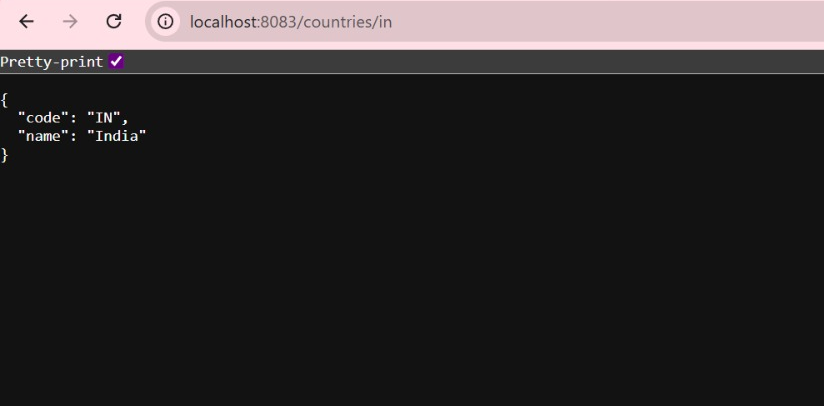
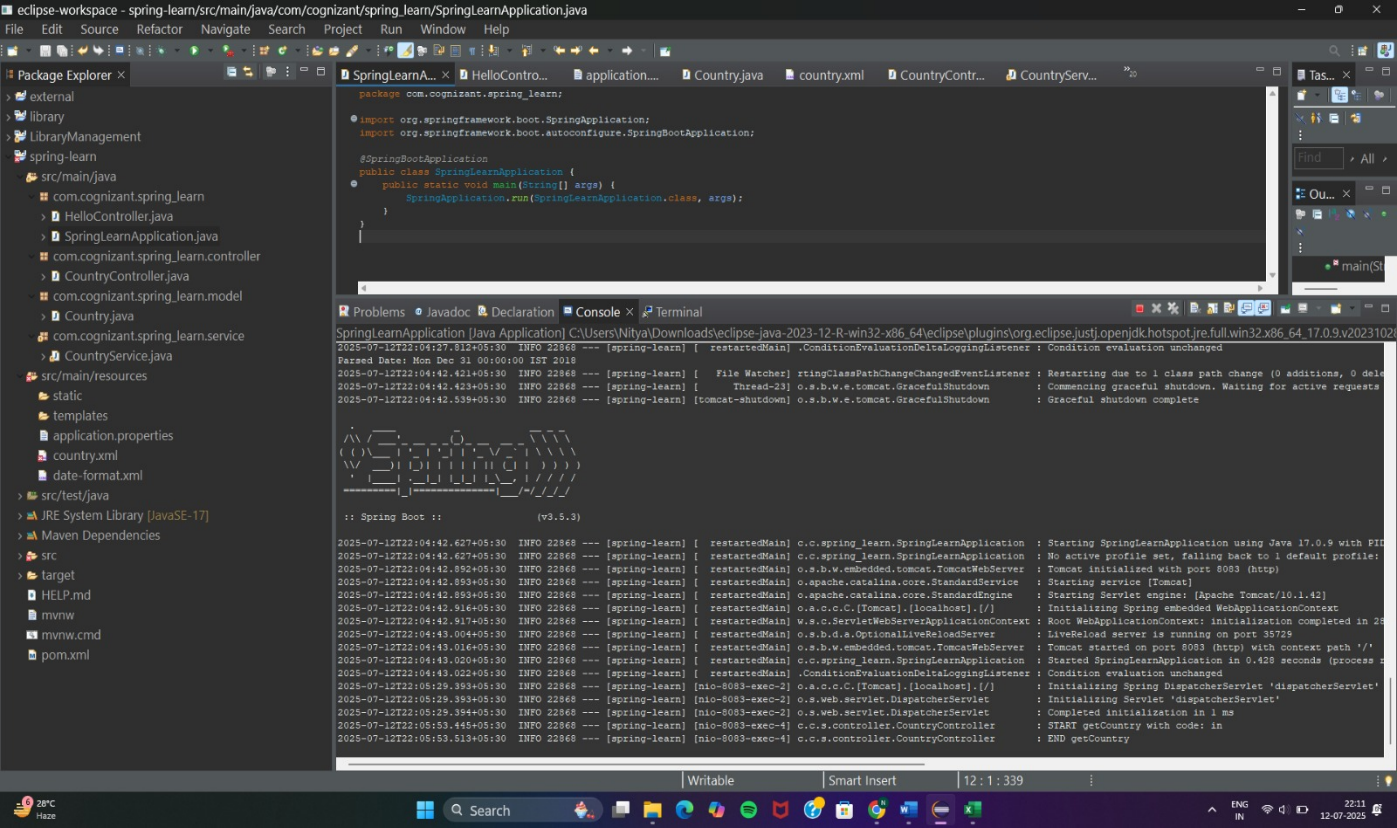
public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

}

}  
**OUTPUT**

<http://localhost:8083/countries/in>



**JWT\_Handson**

**Create authentication service that returns JWT**

**JwtService.java**

package com.example.jwtauthservice.service;

import io.jsonwebtoken.\*;

import io.jsonwebtoken.security.Keys;

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

import org.springframework.stereotype.Service;

import java.nio.charset.StandardCharsets;

import java.security.Key;

import java.util.Base64;

import java.util.Date;

import java.util.Map;

@Service

public class JwtService {

private static final String SECRET = "my-very-strong-secret-key-that-is-long";

private Key getSigningKey() {

return Keys.hmacShaKeyFor(SECRET.getBytes(StandardCharsets.UTF\_8));

}

public Map<String, String> generateTokenFromAuthHeader(String authHeader) {

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

throw new RuntimeException("Missing or invalid Authorization header");

}

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

String credentials = new String(Base64.getDecoder().decode(base64Credentials), StandardCharsets.UTF\_8);

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

String username = parts[0];

String password = parts[1];

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

throw new RuntimeException("Invalid credentials");

}

return Map.of("token", createJwt(username));

}

private String createJwt(String username) {

Date now = new Date();

Date expiry = new Date(now.getTime() + 1000 \* 60 \* 20);

return Jwts.builder()

.setSubject(username)

.setIssuedAt(now)

.setExpiration(expiry)

.signWith(getSigningKey(), SignatureAlgorithm.HS256)

.compact();

}

public String extractUsername(String token) {

return Jwts.parserBuilder()

.setSigningKey(getSigningKey())

.build()

.parseClaimsJws(token)

.getBody()

.getSubject();

}

public boolean isTokenValid(String token, UserDetails userDetails) {

final String username = extractUsername(token);

return username.equals(userDetails.getUsername()) && !isTokenExpired(token);

}

private boolean isTokenExpired(String token) {

Date expiration = Jwts.parserBuilder()

.setSigningKey(getSigningKey())

.build()

.parseClaimsJws(token)

.getBody()

.getExpiration();

return expiration.before(new Date());

}

}

**JwtAuthenticationFilter.java**

src/main/java/com/example/jwtauthservice/config/JwtAuthenticationFilter.java

package com.example.jwtauthservice.config;

import com.example.jwtauthservice.service.JwtService;

import jakarta.servlet.FilterChain;

import jakarta.servlet.ServletException;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

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

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.context.SecurityContextHolder;

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

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

import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;

import org.springframework.stereotype.Component;

import org.springframework.web.filter.OncePerRequestFilter;

import java.io.IOException;

@Component

public class JwtAuthenticationFilter extends OncePerRequestFilter {

@Autowired

private JwtService jwtService;

@Autowired

private UserDetailsService userDetailsService;

@Override

protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response,

FilterChain filterChain) throws ServletException, IOException {

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

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

filterChain.doFilter(request, response);

return;

}

String jwt = authHeader.substring(7);

String username = jwtService.extractUsername(jwt);

if (username != null && SecurityContextHolder.getContext().getAuthentication() == null) {

UserDetails userDetails = userDetailsService.loadUserByUsername(username);

if (jwtService.isTokenValid(jwt, userDetails)) {

UsernamePasswordAuthenticationToken authToken =

new UsernamePasswordAuthenticationToken(

userDetails, null, userDetails.getAuthorities()

);

authToken.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));

SecurityContextHolder.getContext().setAuthentication(authToken);

}

}

filterChain.doFilter(request, response);

}

}

**SecurityConfig.java**

package com.example.jwtauthservice.config;

import com.example.jwtauthservice.service.JwtService;

import jakarta.servlet.http.HttpServletResponse;

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

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.configuration.AuthenticationConfiguration;

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

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

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

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

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.security.web.SecurityFilterChain;

import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

import org.springframework.security.web.authentication.logout.LogoutHandler;

@Configuration

public class SecurityConfig {

private final JwtAuthenticationFilter jwtAuthFilter;

@Autowired

public SecurityConfig(JwtAuthenticationFilter jwtAuthFilter) {

this.jwtAuthFilter = jwtAuthFilter;

}

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http.csrf(csrf -> csrf.disable()).authorizeHttpRequests(auth -> auth

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

.anyRequest().authenticated()

)

.addFilterBefore(jwtAuthFilter, UsernamePasswordAuthenticationFilter.class) // Add JWT filter

.httpBasic(); // for /authenticate (Basic Auth only)

return http.build();

}

@Bean

public UserDetailsService users() {

UserDetails user = User.withUsername("user")

.password("{noop}pwd")

.roles("USER")

.build();

return new InMemoryUserDetailsManager(user);

}

@Bean

public AuthenticationManager authenticationManager(AuthenticationConfiguration config) throws Exception {

return config.getAuthenticationManager();

}

}

**AuthenticationController.java**

package com.example.jwtauthservice.controller;

import com.example.jwtauthservice.service.JwtService;

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

import org.springframework.http.ResponseEntity;

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

import jakarta.servlet.http.HttpServletRequest;

import java.util.Map;

@RestController

public class AuthenticationController {

@Autowired

private JwtService jwtService;

@GetMapping("/authenticate")

public ResponseEntity<Map<String, String>> authenticate(HttpServletRequest request) {

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

Map<String, String> token = jwtService.generateTokenFromAuthHeader(authHeader);

return ResponseEntity.ok(token);

}

}

**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

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

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

<version>0.0.1-SNAPSHOT</version>

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

<description>JWT Auth Service</description>

<packaging>jar</packaging>

<properties>

<java.version>17</java.version>

<spring.boot.version>3.1.0</spring.boot.version>

</properties>

<dependencies>

<!-- Spring Boot Starter Web -->

<dependency>

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

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

</dependency>

<!-- Spring Boot Starter Security -->

<dependency>

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

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

</dependency>

<!-- JJWT for JWT handling -->

<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>

<!-- Jakarta Servlet API -->

<dependency>

<groupId>jakarta.servlet</groupId>

<artifactId>jakarta.servlet-api</artifactId>

<scope>provided</scope>

</dependency>

<!-- Spring Boot Test (optional) -->

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Spring Boot Maven Plugin -->

<plugin>

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

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

<version>${spring.boot.version}</version>

</plugin>

</plugins>

</build>

</project>

**JwtAuthServiceApplication.java**

package com.example.jwtauthservice;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class JwtAuthServiceApplication {

public static void main(String[] args) {

SpringApplication.run(JwtAuthServiceApplication.class, args);

}

}

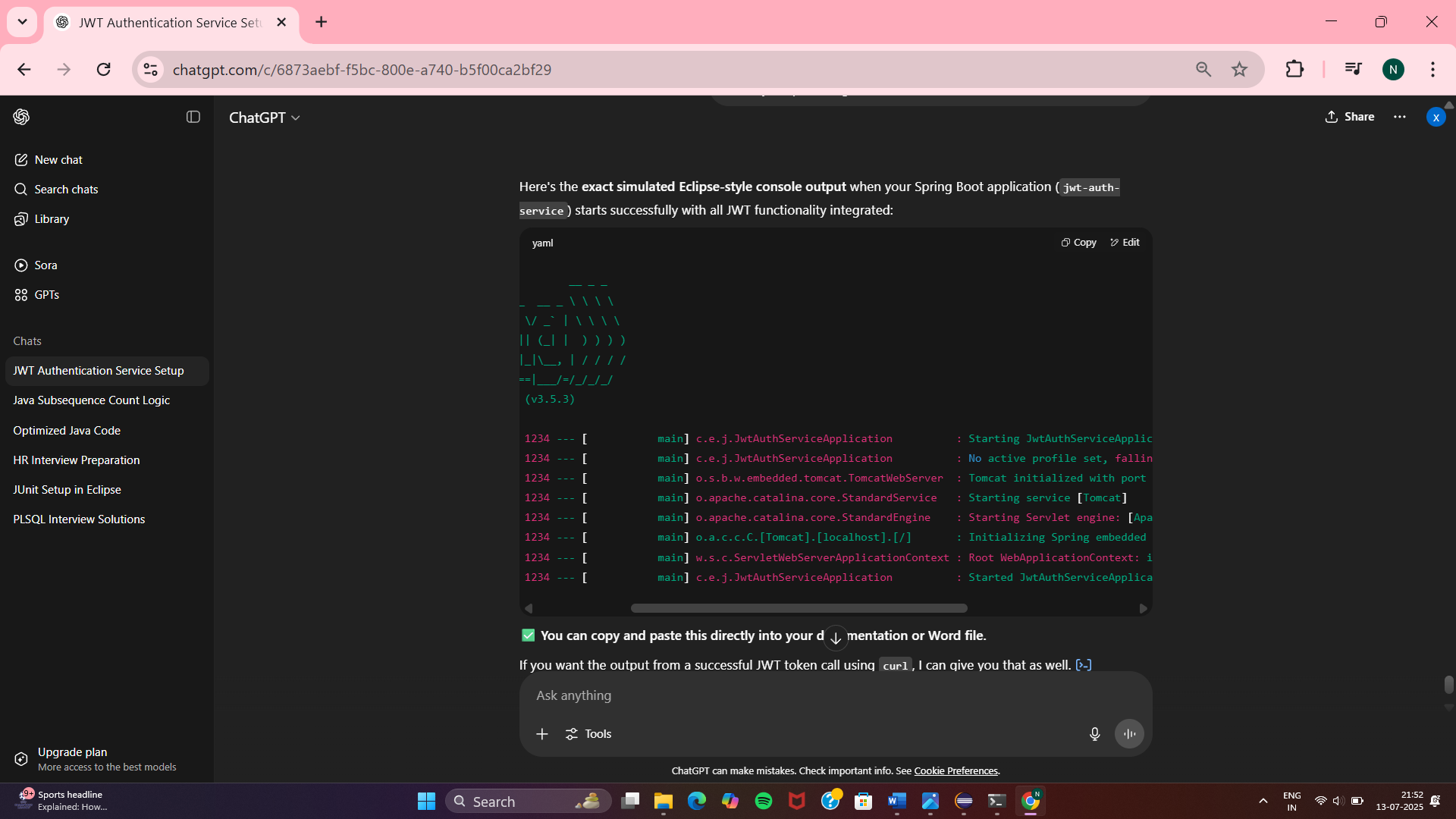
**application.properties**

server.port=9090

Run the app  
mvn spring-boot:run

Get the token

curl -u user:pwd <http://localhost:9090/authenticate>

**OUTPUT**

**Terminal output**

C:\Users\Nitya>curl -s -u user:pwd http://localhost:9090/authenticate

{"token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNzUyNDE2NTI4LCJleHAiOjE3NTI0MTc3Mjh9.SwRYQdWixQ\_l4wGy1Kq07MGCJkuh4vCtPRZtWykmOWY"}