**WEEK 4**

**Create a Spring Web Project using Maven**

**SpringLearnApplication.java**

package com.cognizant.springlearn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

public static void main(String[] args) {

System.out.println("SpringLearnApplication started...");

SpringApplication.run(SpringLearnApplication.class, args);

}}

Pom.xml

<dependencies>

<!-- Spring Boot Starter 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>

<optional>true</optional>

</dependency>

<!-- Testing -->

<dependency>

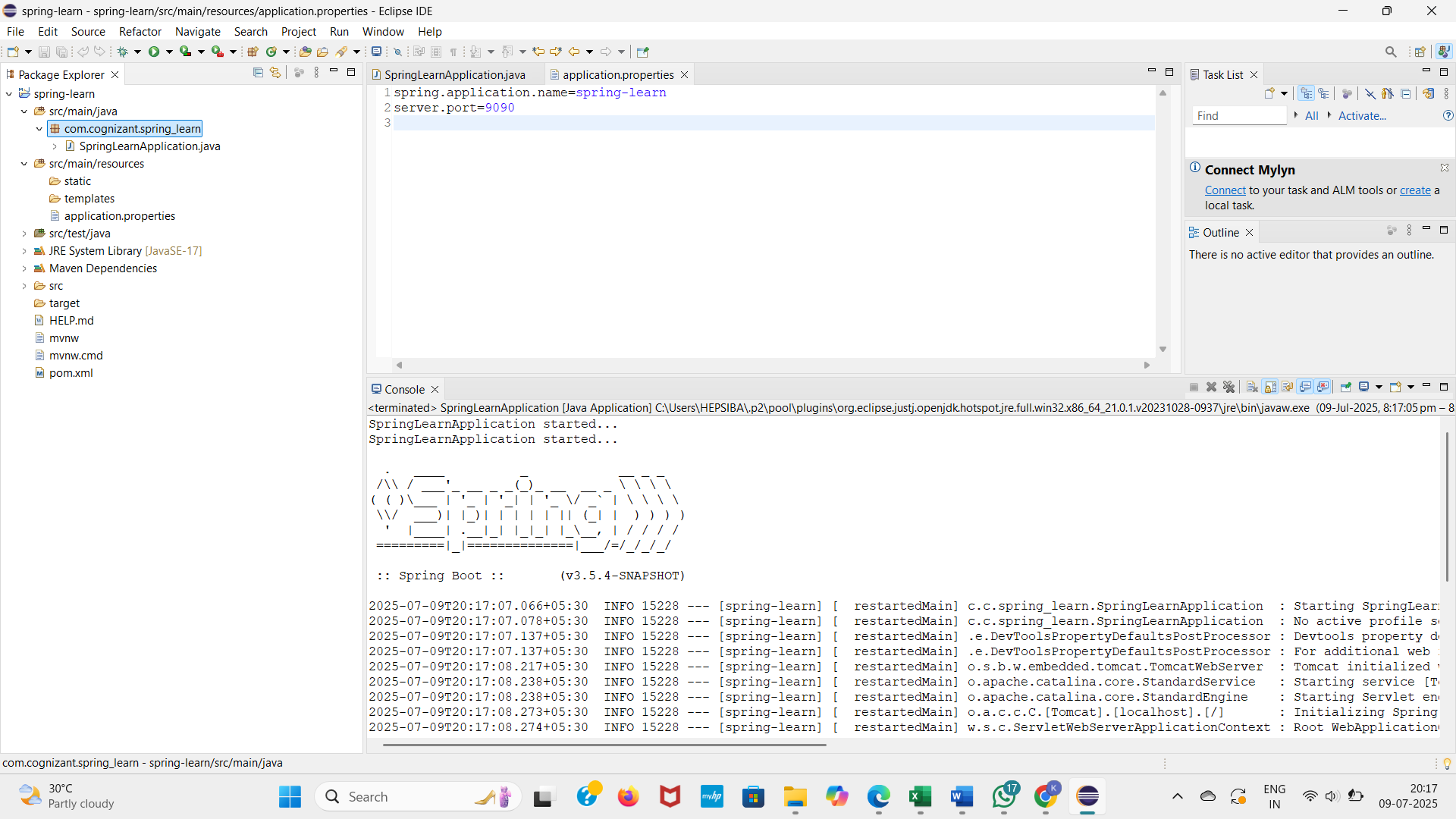
<groupId>org.springframework.boot</groupId>

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

<scope>test</scope>

</dependency>

</dependencies>

**OUTPUT**

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

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 Name

US United States

DE Germany

IN India

JP Japan

Above data has to be stored in spring configuration file. Write a program to read this configuration file and display the details.

**1. Country.java (with logging)**

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

return code;

}

public void setCode(String code) {

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

this.code = code;

}

public String getName() {

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

return name;

}

public void setName(String name) {

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

this.name = name;

}

@Override

public String toString() {

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

}

}

**2. SpringLearnApplication.java**

package com.cognizant.spring\_learn;

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.debug("Main started");

displayCountry();

LOGGER.debug("Main ended");

}

public static void displayCountry() {

LOGGER.debug("Inside displayCountry()");

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

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

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

}

}

**3. 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>

**4. pom.xml (Maven)**

<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.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-learn</name>

<description>Spring XML Config Example</description>

<dependencies>

<!-- Spring Context for XML Configuration -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.32</version>

</dependency>

<!-- Logging -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.36</version>

</dependency>

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>1.2.11</version>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Compiler plugin -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.1</version>

<configuration>

<source>17</source>

<target>17</target>

</configuration>

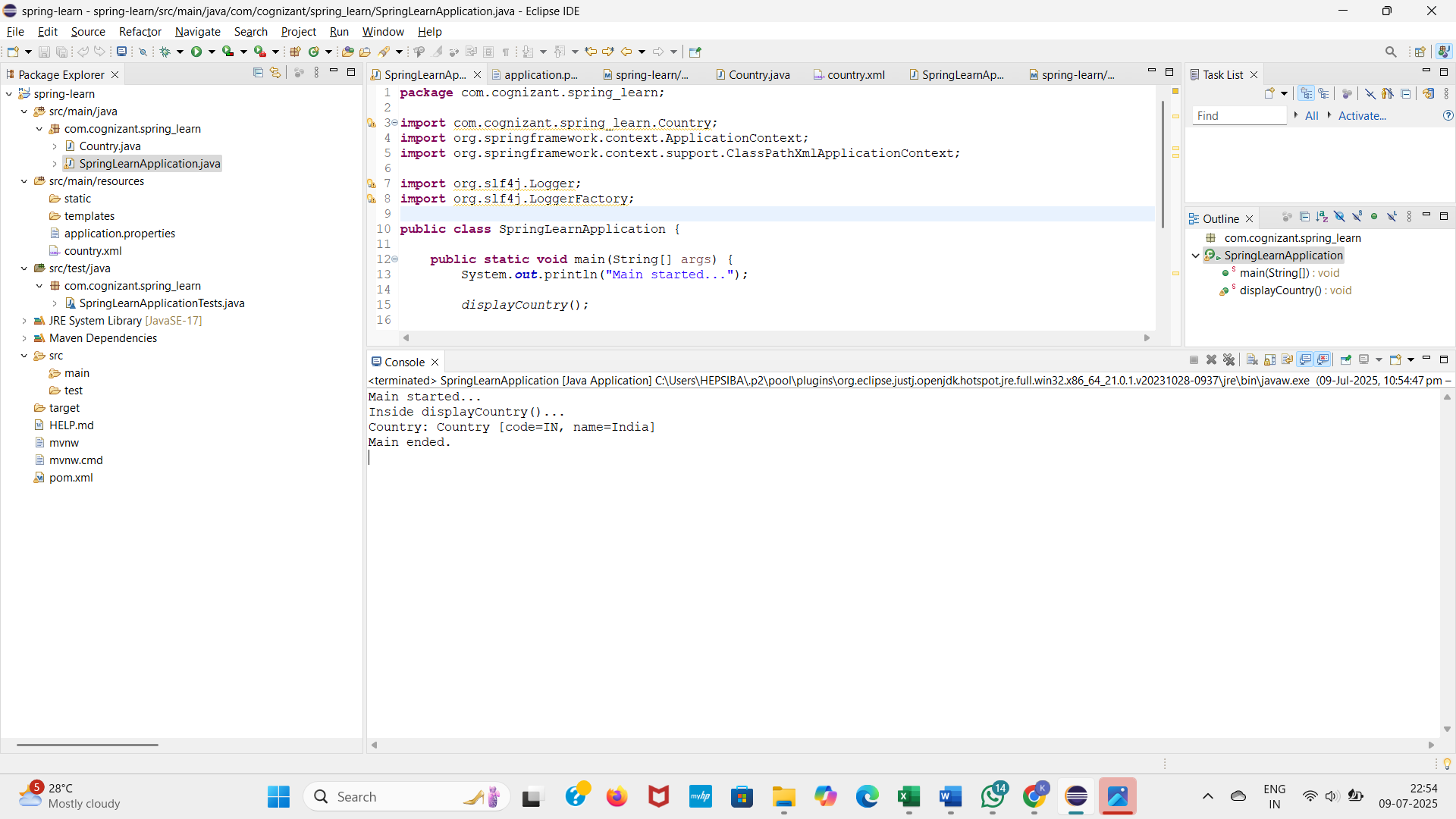
</plugin>

</plugins>

</build>

</project>

**OUTPUT**



**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:

**1. 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;

}

}

**2. Update application.properties**

server.port=8083

logging.level.root=debug

**3. Ensure your SpringLearnApplication.java looks like this:**

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);

}

}

**4. pom.xml dependencies (basic working Spring Boot setup)**

<dependencies>

<!-- Spring Boot Web -->

<dependency>

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

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

</dependency>

<!-- Logging (already comes with starter) -->

<dependency>

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

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

</dependency>

<!-- Testing (optional) -->

<dependency>

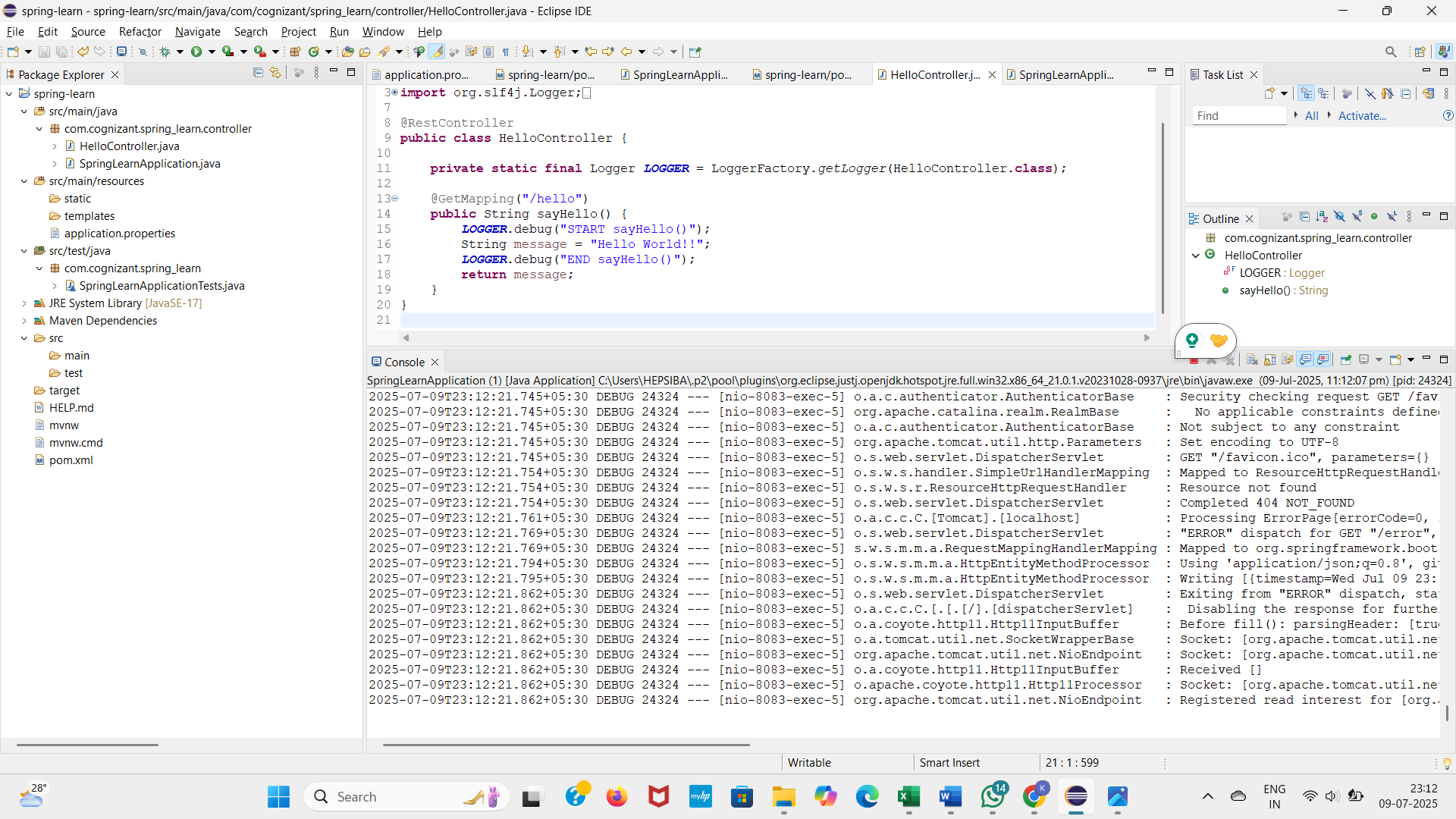
<groupId>org.springframework.boot</groupId>

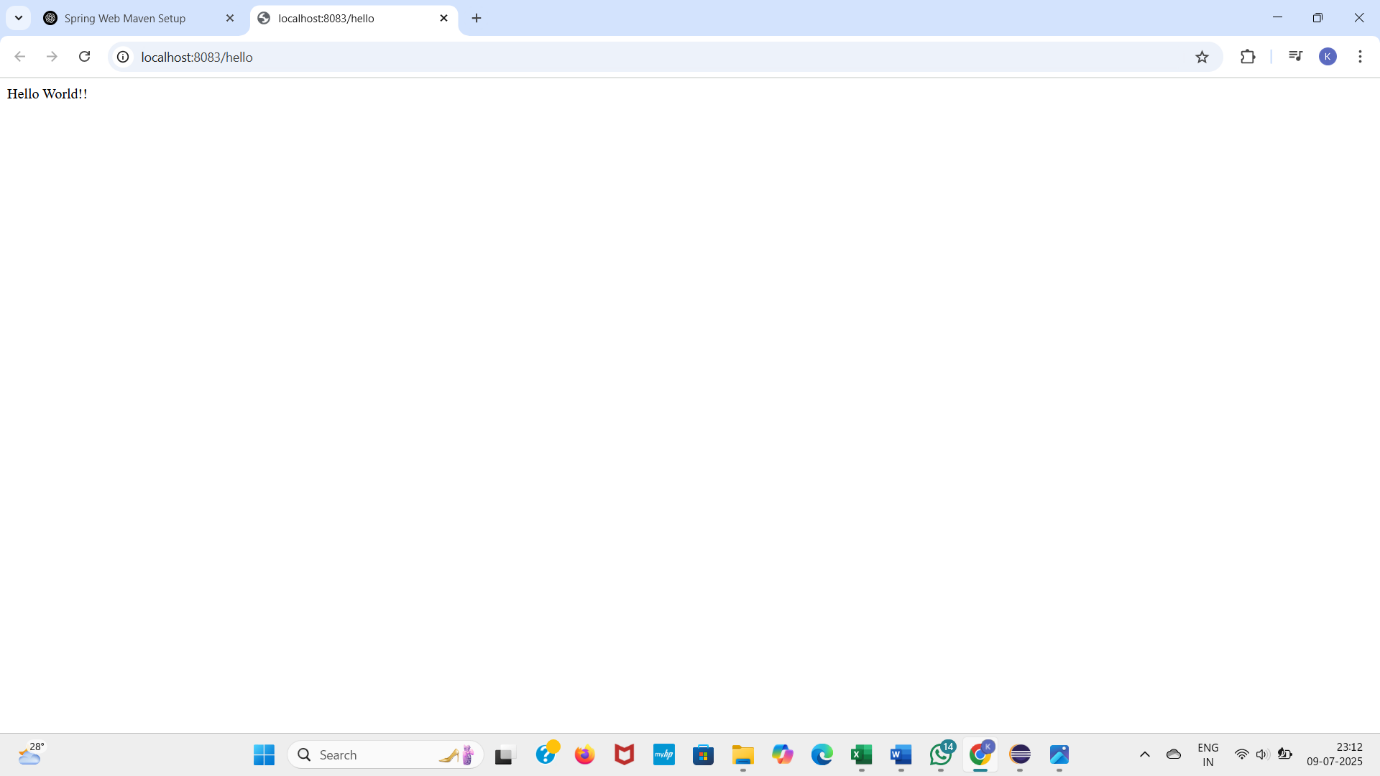
<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

**OUTPUT**



**REST - Country Web Service**   
  
Write a REST service that returns India country details in the earlier created spring learn application.  
**Step 1: 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

https://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="CN"/>

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

</bean>

</list>

</constructor-arg>

</bean>

</beans>

**Step 2: Model — 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;

}

}

**Step 3: Service — 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;

public class CountryService {

public Country getCountry(String code) {

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

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

// Case-insensitive search using Stream API

return countryList.stream()

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

.findFirst()

.orElse(null); // You can throw exception here if country not found

}

}

**Step 4: Controller — CountryController.java**

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.model.Country;

import com.cognizant.springlearn.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);

}

}

**Step 5: Sample Request**

http://localhost:8083/countries/in

**Sample Response**

json

CopyEdit

{

"code": "IN",

"name": "India"

}

**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

**Create Country Model Class**

**How to Create**

1. Right-click on model → New → Class
2. Name it Country
3. Paste this code:

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;

}

}

Save the file

**STEP 2: Create country.xml (Spring Config)**

<?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="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>

**STEP 3: Create CountryController Class**

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;

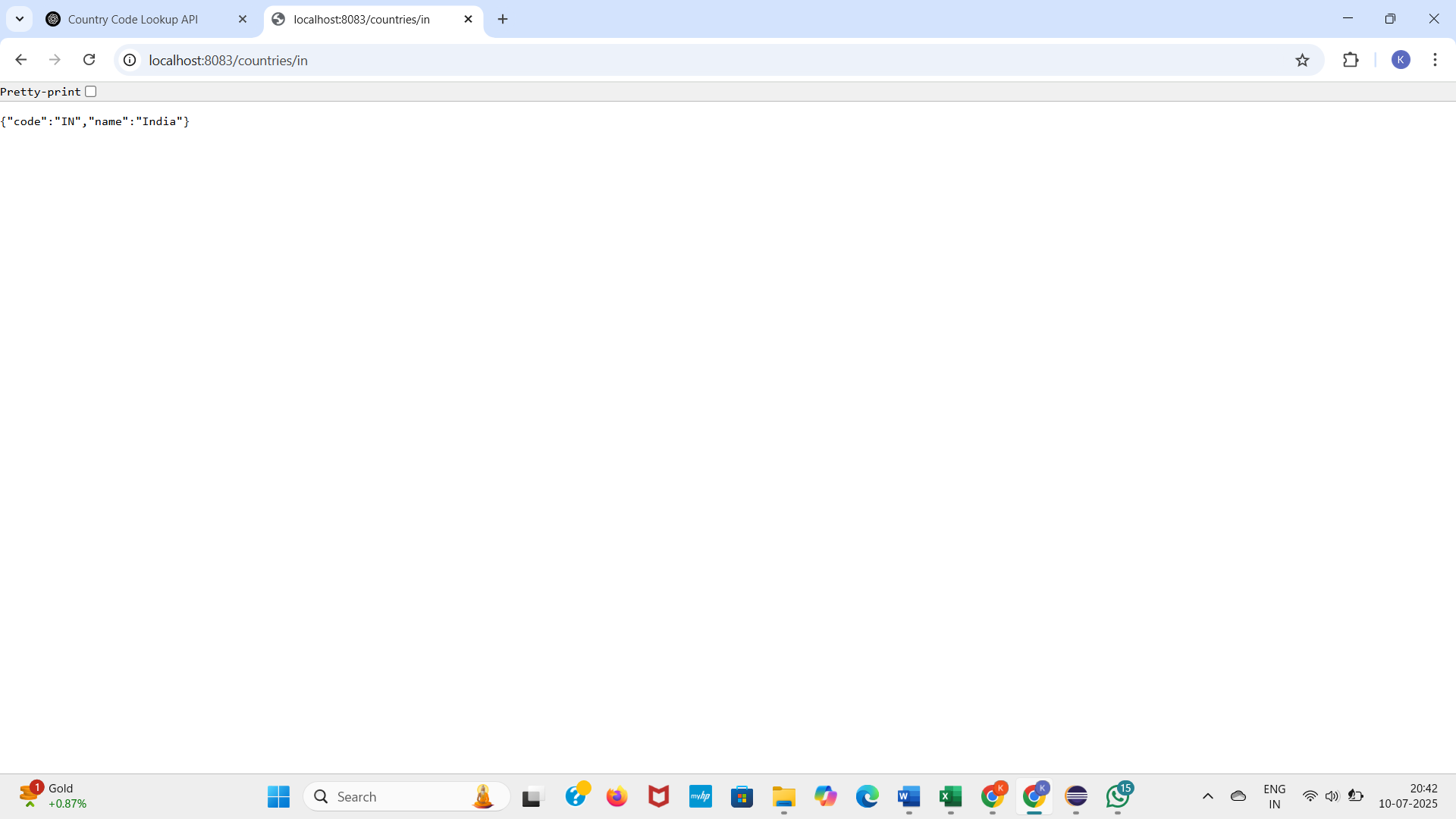
@GetMapping("/countries/{code}")

public Country getCountry(@PathVariable String code) {

return countryService.getCountry(code);

}

}



**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.

**1. JwtUtil.java**

package com.cognizant.spring\_learn.service;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

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

import org.springframework.stereotype.Service;

import java.util.Date;

@Service

public class JwtUtil {

private final String SECRET\_KEY = "mysecretkey";

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(SignatureAlgorithm.HS256, SECRET\_KEY)

.compact();

}

public String extractUsername(String token) {

return Jwts.parser()

.setSigningKey(SECRET\_KEY)

.parseClaimsJws(token)

.getBody()

.getSubject();

}

public boolean validateToken(String token, UserDetails userDetails) {

final String username = extractUsername(token);

return username.equals(userDetails.getUsername());

}

}

**2. AuthenticationController.java**

package com.cognizant.spring\_learn.controller;

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

import org.springframework.http.ResponseEntity;

import org.springframework.security.core.Authentication;

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

import java.util.HashMap;

import java.util.Map;

@RestController

public class AuthenticationController {

private final JwtUtil jwtUtil;

public AuthenticationController(JwtUtil jwtUtil) {

this.jwtUtil = jwtUtil;

}

@GetMapping("/authenticate")

public ResponseEntity<?> authenticate(Authentication authentication) {

String username = authentication.getName(); // From Basic Auth

String token = jwtUtil.generateToken(username);

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

response.put("token", token);

return ResponseEntity.ok(response);

}

}

**3. JwtRequestFilter.java**

package com.cognizant.spring\_learn.filter;

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

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.\*;

import org.springframework.stereotype.Component;

import org.springframework.web.filter.OncePerRequestFilter;

import java.io.IOException;

@Component

public class JwtRequestFilter extends OncePerRequestFilter {

@Autowired

private JwtUtil jwtUtil;

@Autowired

private UserDetailsService userDetailsService;

@Override

protected void doFilterInternal(HttpServletRequest request,

HttpServletResponse response,

FilterChain chain)

throws ServletException, IOException {

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

String username = null;

String jwt = null;

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

jwt = authHeader.substring(7);

try {

username = jwtUtil.extractUsername(jwt);

} catch (Exception e) {

System.out.println("Invalid JWT: " + e.getMessage());

}

}

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

UserDetails userDetails = userDetailsService.loadUserByUsername(username);

if (jwtUtil.validateToken(jwt, userDetails)) {

UsernamePasswordAuthenticationToken token =

new UsernamePasswordAuthenticationToken(userDetails, null, userDetails.getAuthorities());

SecurityContextHolder.getContext().setAuthentication(token);

}

}

chain.doFilter(request, response);

}

}

**4. SecurityConfig.java (Updated)**

package com.cognizant.spring\_learn.config;

import com.cognizant.spring\_learn.filter.JwtRequestFilter;

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.\*;

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

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

import org.springframework.security.web.SecurityFilterChain;

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

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

@Configuration

public class SecurityConfig {

@Autowired

private JwtRequestFilter jwtRequestFilter;

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http.csrf().disable()

.authorizeHttpRequests()

.requestMatchers("/authenticate").authenticated()

.anyRequest().authenticated()

.and()

.httpBasic();

http.addFilterBefore(jwtRequestFilter, UsernamePasswordAuthenticationFilter.class);

return http.build();

}

@Bean

public UserDetailsService userDetailsService() {

UserDetails user = User.withUsername("user")

.password("pwd")

.roles("USER")

.build();

return new InMemoryUserDetailsManager(user);

}

@Bean

public PasswordEncoder passwordEncoder() {

return NoOpPasswordEncoder.getInstance(); // For demo only

}

}

**5. HelloController.java (Test Endpoint)**

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("/hello")

public String hello() {

return "Hello, secure world!";

}

}

**OUTPUT**

