Name : Arshiya Tabassum A

Superset ID : 6424209

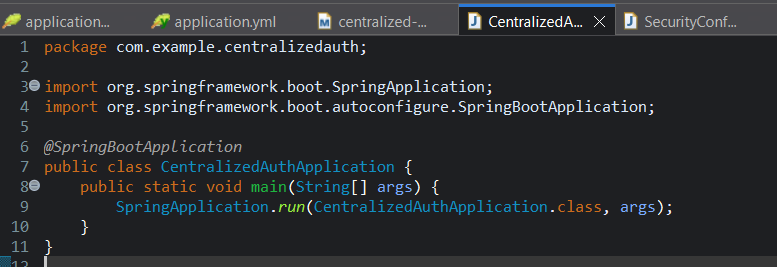
**WEEK 5**

**Sample exercises on Centralized Authentication and SSO with Spring Boot 3 and Spring Cloud**

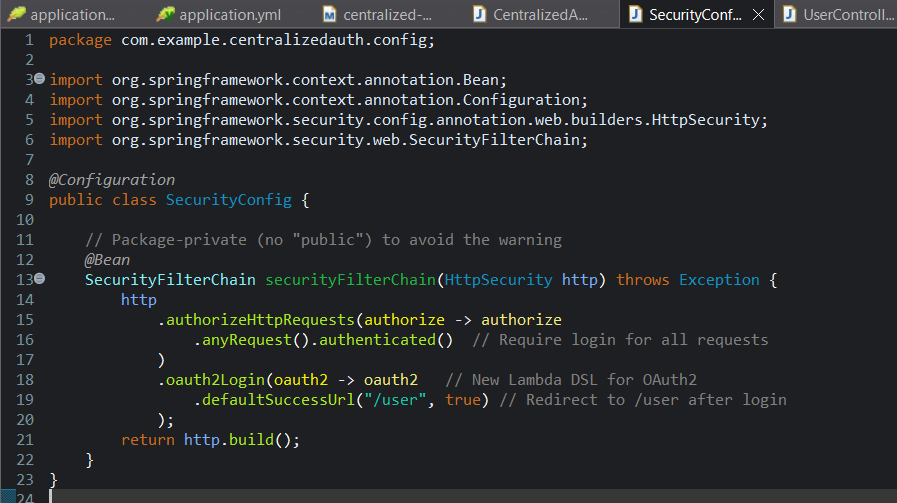
**Exercise 1: Implementing Centralized Authentication with OAuth 2.1/OIDC Task: Implement centralized authentication using OAuth 2.1/OIDC in a Spring Boot application.**

**Code :**

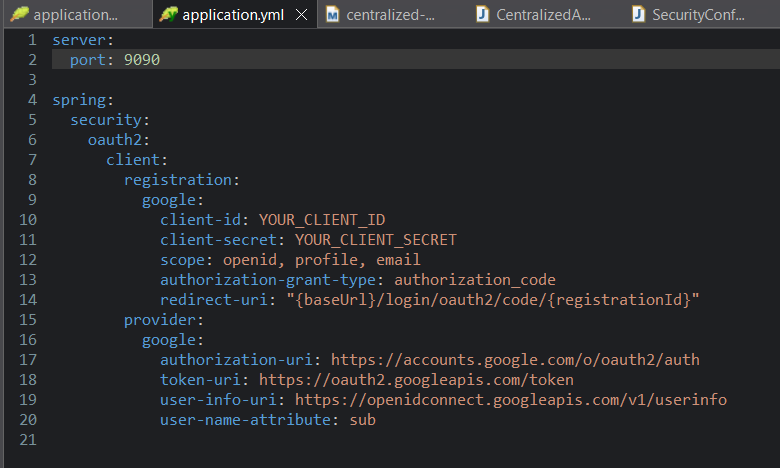
**CentralizedAuthApplication.java**

****

**SecurityConfig.java**

****

**Application.yml**

****

**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>3.5.3</version>

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

</parent>

<groupId>com.example</groupId>

<artifactId>centralized-auth</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>centralized-auth</name>

<description>Demo project for Spring Boot</description>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

<java.version>21</java.version>

</properties>

<dependencies>

<dependency>

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

<artifactId>spring-boot-starter-oauth2-client</artifactId>

</dependency>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

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

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

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

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

<artifactId>spring-boot-configuration-processor</artifactId>

<optional>true</optional>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

<configuration>

<annotationProcessorPaths>

<path>

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

<artifactId>spring-boot-configuration-processor</artifactId>

</path>

</annotationProcessorPaths>

</configuration>

</plugin>

<plugin>

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

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

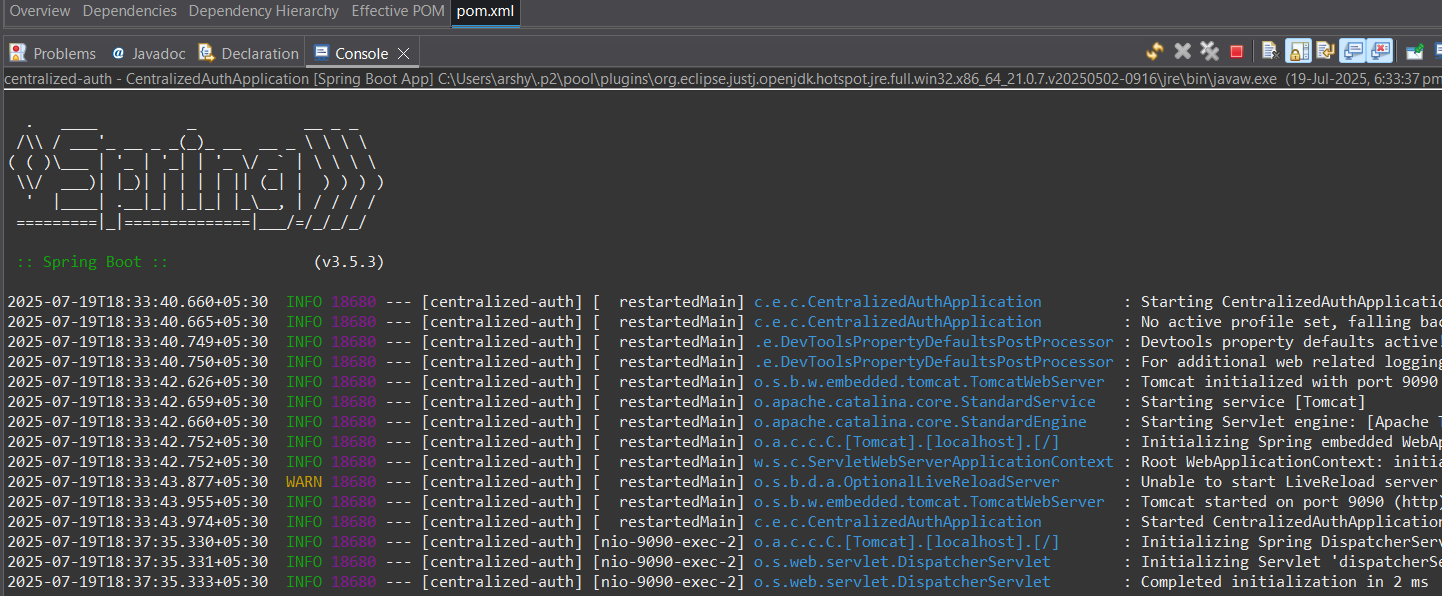
</plugin>

</plugins>

</build>

</project>

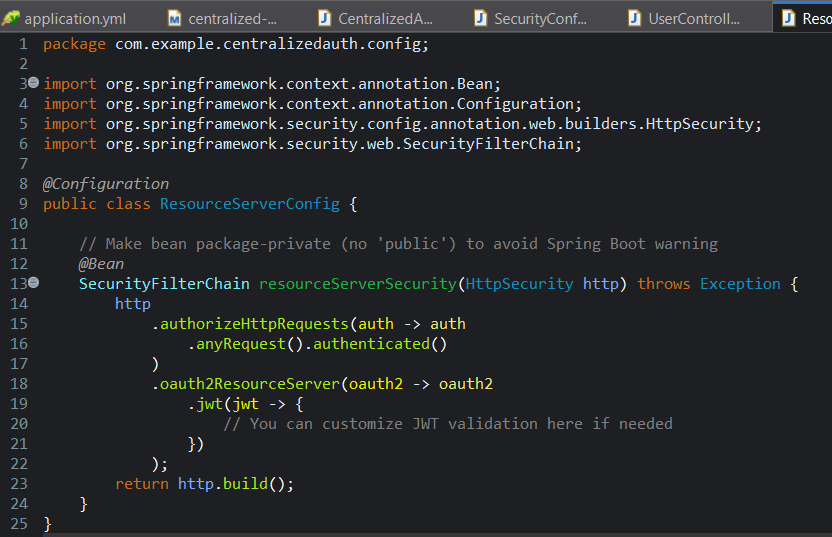
**Output :**

****

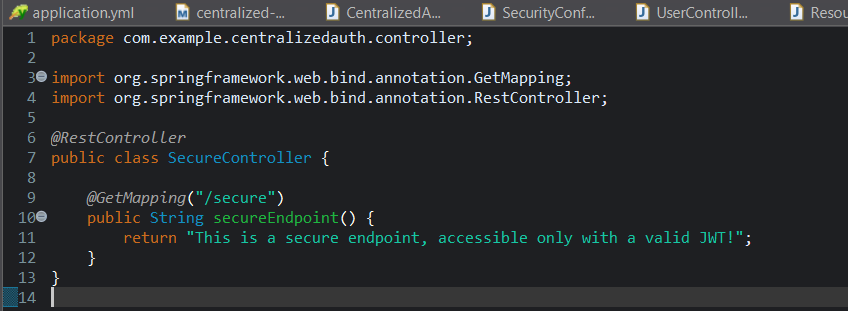
**Exercise 2: Configuring Authorization Servers and Resource Servers Task: Configure Authorization Servers and Resource Servers in a Spring Boot application.**

**Code :**

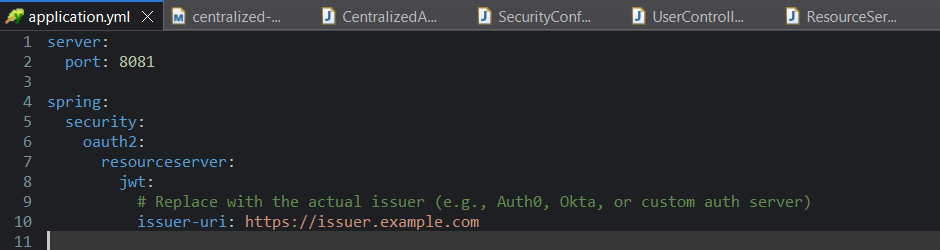
**REsourceServerConfig.java**

****

**SecureController.java**

****

**Application.yml**

****

**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>3.5.3</version>

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

</parent>

<groupId>com.example</groupId>

<artifactId>centralized-auth</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>centralized-auth</name>

<description>Demo project for Spring Boot</description>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

<java.version>21</java.version>

</properties>

<dependencies>

<dependency>

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

<artifactId>spring-boot-starter-oauth2-client</artifactId>

</dependency>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

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

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

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

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

<artifactId>spring-boot-configuration-processor</artifactId>

<optional>true</optional>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

<configuration>

<annotationProcessorPaths>

<path>

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

<artifactId>spring-boot-configuration-processor</artifactId>

</path>

</annotationProcessorPaths>

</configuration>

</plugin>

<plugin>

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

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

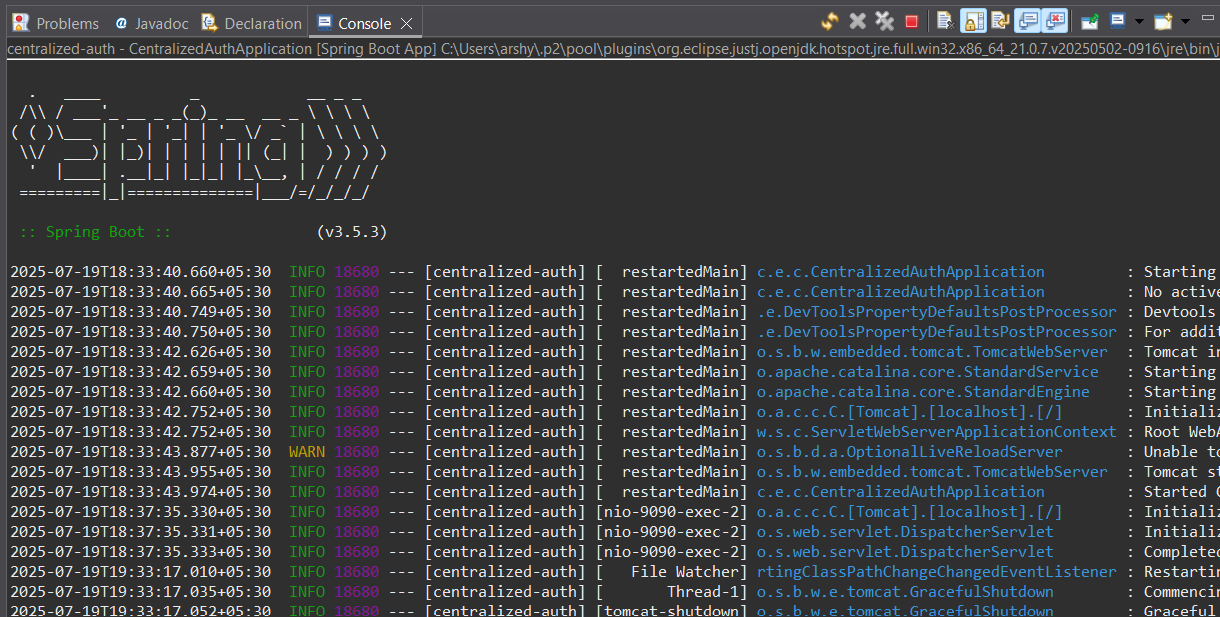
</plugin>

</plugins>

</build>

</project>

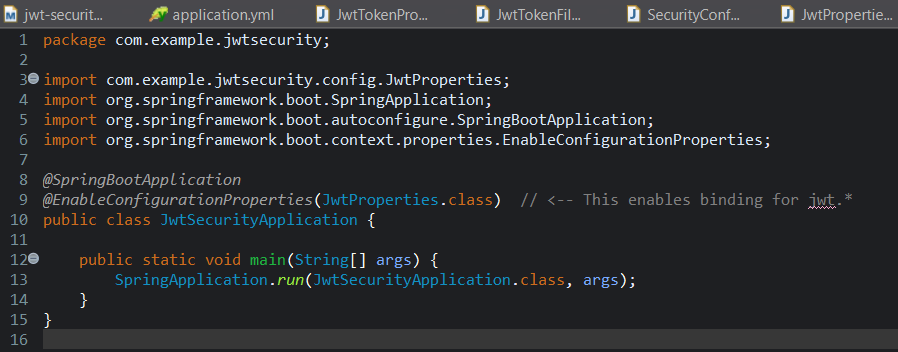
**Output :**

****

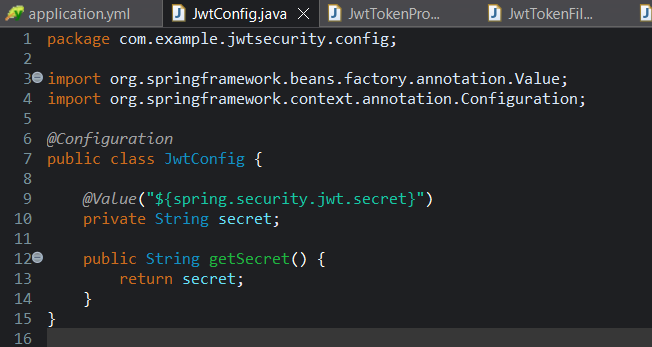
**Exercise 3: Using JSON Web Tokens (JWT) for Secure Communication Task: Use JSON Web Tokens (JWT) for secure communication in a Spring Boot application.**

**Code :**

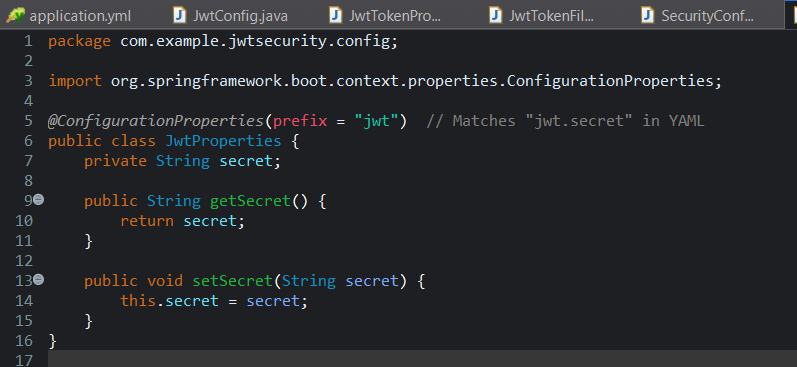
**JwtSecurityApplication :**

****

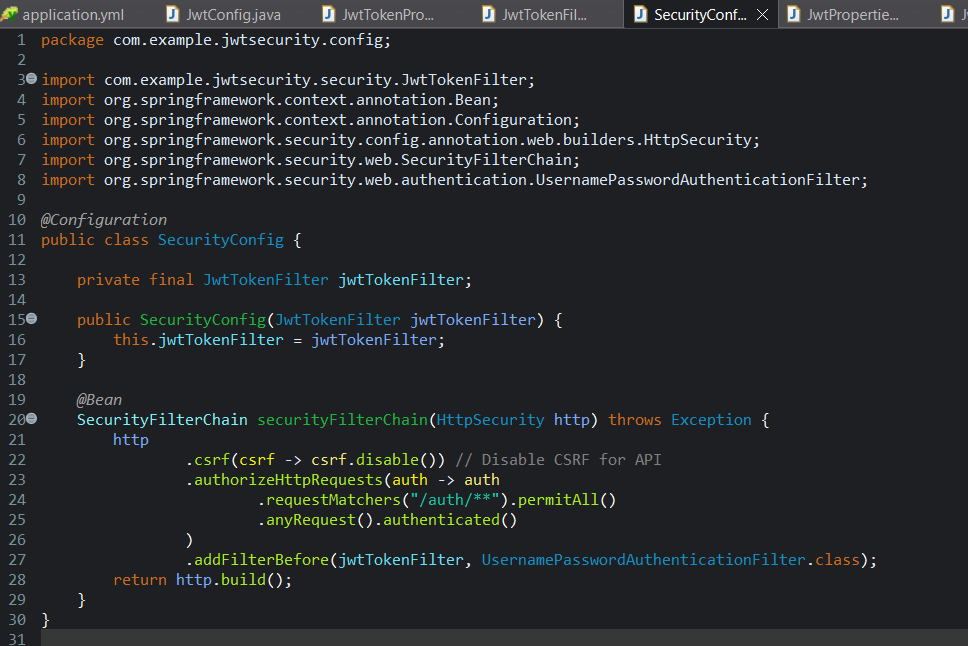
**JwtConfig.java**

****

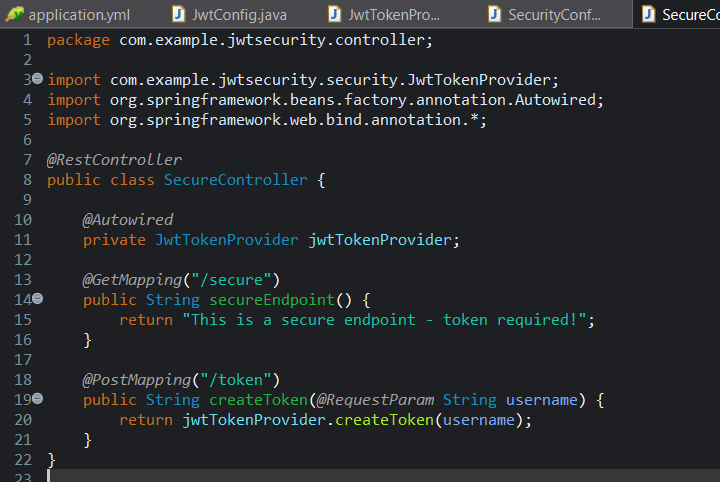
**JwtProperties.java**

****

**SecurityConfig.java**

****

**SecureController.java**

****

**JwtTokenFilter.java**

package com.example.jwtsecurity.security;

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.User;

import org.springframework.stereotype.Component;

import org.springframework.web.filter.OncePerRequestFilter;

import java.io.IOException;

import java.util.Collections;

*@Component*

public class JwtTokenFilter extends OncePerRequestFilter {

*@Autowired*

private JwtTokenProvider jwtTokenProvider;

*@Override*

protected void doFilterInternal(HttpServletRequest request,

HttpServletResponse response,

FilterChain filterChain) throws ServletException, IOException {

String token = resolveToken(request);

if (token != null && jwtTokenProvider.validateToken(token)) {

// Extract username from the token

String username = jwtTokenProvider.getUsername(token);

// Create an Authentication object (Spring Security compatible)

UsernamePasswordAuthenticationToken auth =

new UsernamePasswordAuthenticationToken(

new User(username, "", Collections.*emptyList*()),

null,

Collections.*emptyList*()

);

// Set authentication into the SecurityContext

SecurityContextHolder.*getContext*().setAuthentication(auth);

}

filterChain.doFilter(request, response);

}

private String resolveToken(HttpServletRequest request) {

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

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

return bearerToken.substring(7);

}

return null;

}

}

**JwtTokenProvider.java**

package com.example.jwtsecurity.security;

import com.example.jwtsecurity.config.JwtProperties;

import io.jsonwebtoken.Claims;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

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

import org.springframework.stereotype.Component;

import java.util.Date;

*@Component*

public class JwtTokenProvider {

*@Autowired*

private JwtProperties jwtProperties;

// Generate a token for a username

public String createToken(String username) {

Date now = new Date();

Date validity = new Date(now.getTime() + 3600000); // 1 hour

return Jwts.*builder*()

.setSubject(username)

.setIssuedAt(now)

.setExpiration(validity)

.signWith(*SignatureAlgorithm*.***HS256***, jwtProperties.getSecret())

.compact();

}

// Validate a token (check signature and expiration)

public boolean validateToken(String token) {

try {

Jwts.*parser*()

.setSigningKey(jwtProperties.getSecret())

.parseClaimsJws(token);

return true;

} catch (Exception e) {

return false;

}

}

// Extract username from token

public String getUsername(String token) {

Claims claims = Jwts.*parser*()

.setSigningKey(jwtProperties.getSecret())

.parseClaimsJws(token)

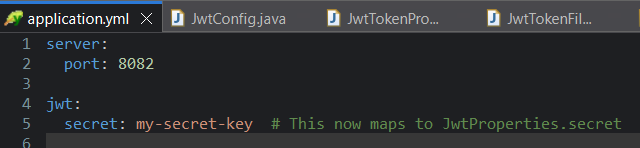
.getBody();

return claims.getSubject();

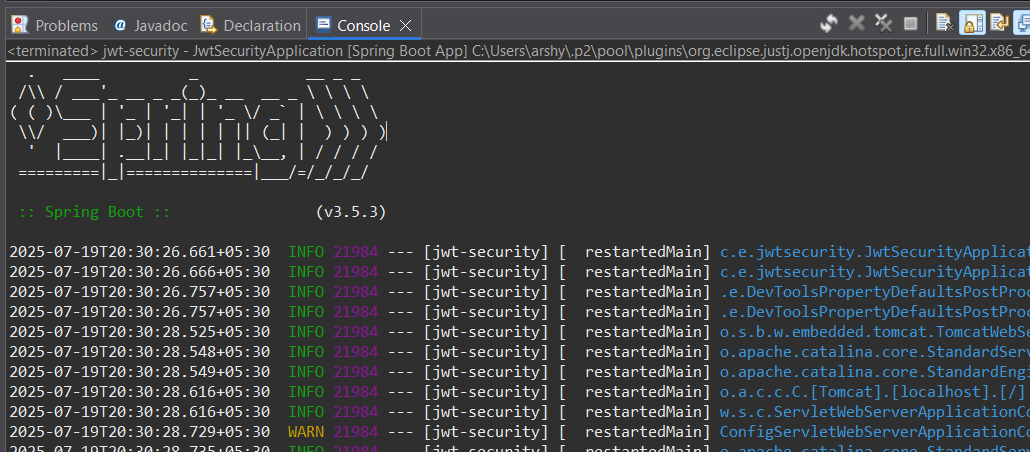
}

}

**Application.yml**

****

**Output :**

****