**CTS DIGITAL NURTURE - 4.0 JAVA FSE**

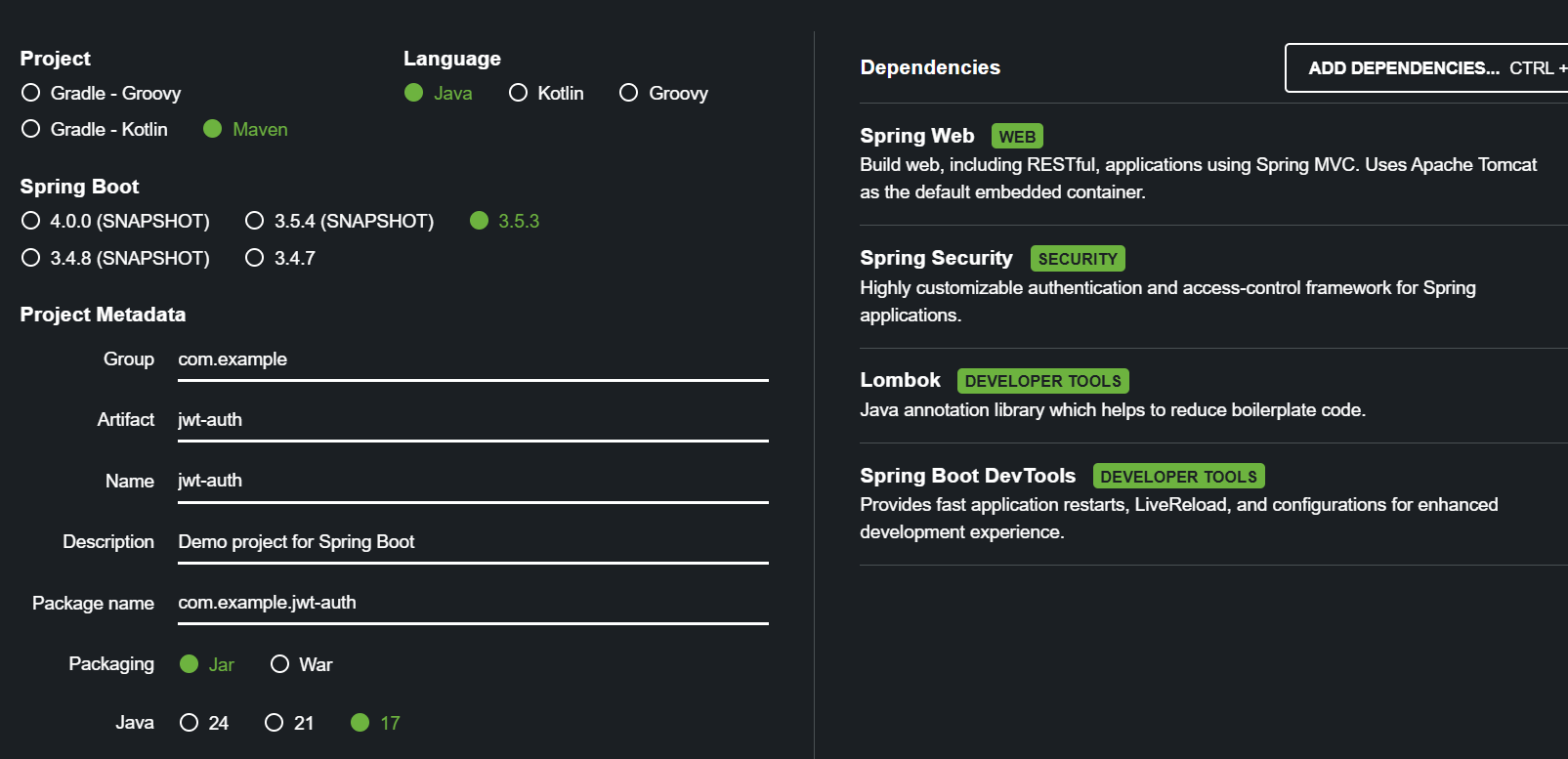
**WEEK 4 -** **Spring REST using Spring Boot 3**

**5. JWT-handson**

**Create authentication service that returns JWT**

PROGRAM:

**In Spring Initializer**



**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>jwt-auth</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>jwt-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>17</java.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<!-- Spring Boot Starter Web -->

<dependency>

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

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

</dependency>

<!-- Spring Boot Security -->

<dependency>

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

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

</dependency>

<!-- JWT -->

<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> <!-- or jjwt-gson -->

<version>0.11.5</version>

<scope>runtime</scope>

</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.projectlombok</groupId>

<artifactId>lombok</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>

<dependency>

<groupId>jakarta.servlet</groupId>

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

<scope>provided</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

<configuration>

<annotationProcessorPaths>

<path>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</path>

</annotationProcessorPaths>

</configuration>

</plugin>

<plugin>

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

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

<configuration>

<excludes>

<exclude>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</exclude>

</excludes>

</configuration>

</plugin>

</plugins>

</build>

</project>

**AuthenticationController.java**

package com.example.jwt\_auth.controller;

import jakarta.servlet.http.HttpServletRequest;

import org.springframework.http.ResponseEntity;

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

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

import java.util.Base64;

import java.util.Date;

import java.security.Key;

*@RestController*

public class AuthenticationController {

private static final Key ***SECRET\_KEY*** = Keys.*hmacShaKeyFor*("mysecretkeymysecretkeymysecretkey12".getBytes());

*@GetMapping*("/authenticate")

public ResponseEntity<?> authenticate(HttpServletRequest request) {

System.***out***.println("authenticate endpoint hit");

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

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

return ResponseEntity.*status*(401).body("Missing or invalid Authorization header");

}

try {

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

byte[] decodedBytes = Base64.*getDecoder*().decode(base64Credentials);

String decodedCredentials = new String(decodedBytes);

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

String username = credentials[0];

String password = credentials[1];

if ("user".equals(username) && "pwd".equals(password)) {

String token = Jwts.*builder*()

.setSubject(username)

.setIssuedAt(new Date())

.setExpiration(new Date(System.*currentTimeMillis*() + 1000 \* 60 \* 2)) .signWith(***SECRET\_KEY***, *SignatureAlgorithm*.***HS256***)

.compact();

return ResponseEntity.*ok*("{\"token\":\"" + token + "\"}");

}

return ResponseEntity.*status*(403).body("Invalid credentials");

} catch (Exception e) {

e.printStackTrace();

return ResponseEntity.*status*(500).body("Error: " + e.getMessage());

}

}

}

**SecurityConfig.java**

package com.example.jwt\_auth.config;

import com.example.jwt\_auth.filter.JwtAuthFilter;

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.config.http.SessionCreationPolicy;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

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

import org.springframework.security.web.SecurityFilterChain;

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

*@Configuration*

public class SecurityConfig {

private final JwtAuthFilter jwtAuthFilter;

public SecurityConfig(JwtAuthFilter jwtAuthFilter) {

this.jwtAuthFilter = jwtAuthFilter;

}

*@Bean*

public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

http

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

.sessionManagement(session -> session.sessionCreationPolicy(*SessionCreationPolicy*.***STATELESS***))

.addFilterBefore(jwtAuthFilter, UsernamePasswordAuthenticationFilter.class);

return http.build();

}

*@Bean*

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

*@Bean*

public AuthenticationManager authenticationManager(AuthenticationConfiguration config) throws Exception {

return config.getAuthenticationManager();

}

}

**JwtAuthFilter.java**

package com.example.jwt\_auth.filter;

import io.jsonwebtoken.\*;

import io.jsonwebtoken.security.Keys;

import jakarta.servlet.FilterChain;

import jakarta.servlet.ServletException;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

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

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

import org.springframework.stereotype.Component;

import org.springframework.web.filter.OncePerRequestFilter;

import java.io.IOException;

import java.nio.charset.StandardCharsets;

import java.util.Collections;

*@Component*

public class JwtAuthFilter extends OncePerRequestFilter {

private static final String ***SECRET\_KEY*** = "mysecretkeymysecretkey123456"; // 32+ characters

*@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 token = authHeader.substring(7);

try {

Claims claims = Jwts

.*parserBuilder*()

.setSigningKey(Keys.*hmacShaKeyFor*(***SECRET\_KEY***.getBytes(StandardCharsets.***UTF\_8***)))

.build()

.parseClaimsJws(token)

.getBody();

String username = claims.getSubject();

if (username != null) {

UsernamePasswordAuthenticationToken auth =

new UsernamePasswordAuthenticationToken(username, null, Collections.*emptyList*());

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

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

}

} catch (JwtException e) {

response.setStatus(HttpServletResponse.***SC\_UNAUTHORIZED***);

return;

}

filterChain.doFilter(request, response);

}

}

**JwtAuthApplication.java**

package com.example.jwt\_auth;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@SpringBootApplication*

public class JwtAuthApplication {

public static void main(String[] args) {

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

}

}

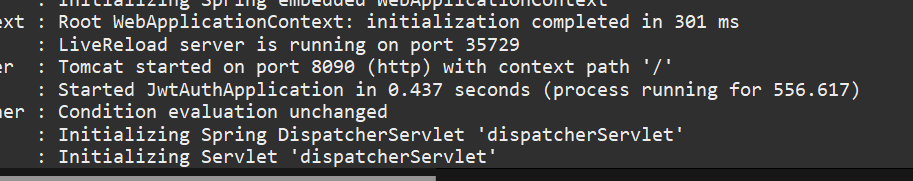
**application.properties**

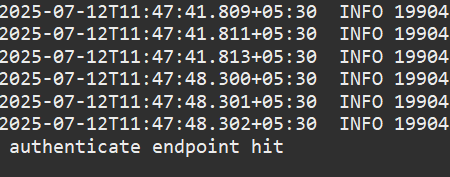
spring.application.name=jwt-auth

server.port=8090

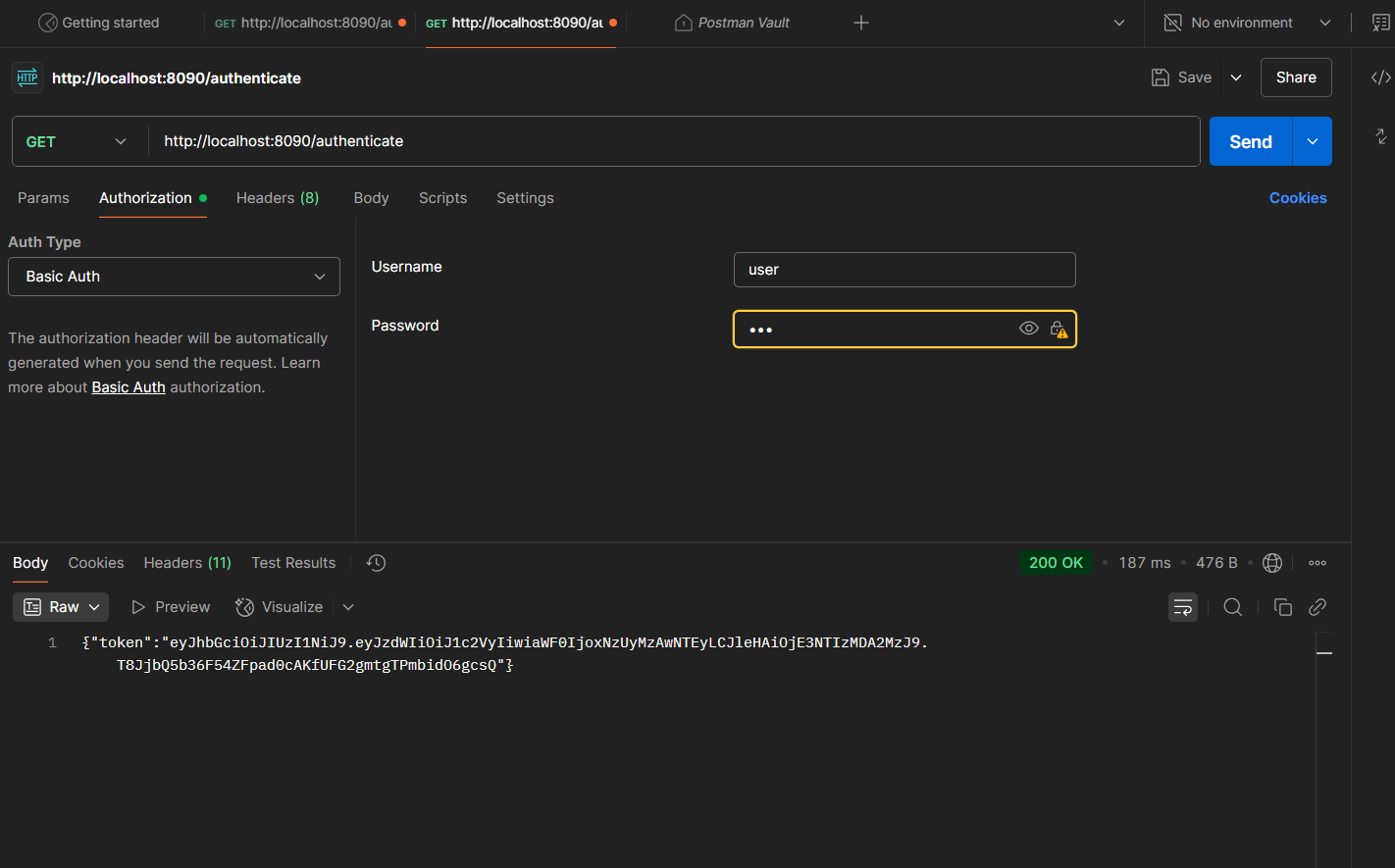
OUTPUT:

**In eclipse console:**

****

****

**In postman:**

****