**Create authentication service that returns JWT:**

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>

<packaging>jar</packaging>

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

<description>JWT Authentication Service</description>

<parent>

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

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

<version>2.7.17</version> <!-- Choose a version that works with your Java version -->

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

</parent>

<properties>

<java.version>11</java.version> <!-- Adjust based on your setup -->

</properties>

<dependencies>

<!-- Web -->

<dependency>

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

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

</dependency>

<!-- Security -->

<dependency>

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

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

</dependency>

<!-- JWT -->

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.1</version>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Spring Boot Maven Plugin -->

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

JwtUtil.java:

package com.example.jwtauth.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import org.springframework.stereotype.Component;

import java.util.Date;

@Component

public class JwtUtil {

private final String SECRET\_KEY = "secret";

public String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date())

.setExpiration(new Date(System.currentTimeMillis() + 1000 \* 60 \* 10)) // 10 min expiry

.signWith(SignatureAlgorithm.HS256, SECRET\_KEY)

.compact();

}

}

SecurityConfig.java:

package com.example.jwtauth.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.web.SecurityFilterChain;

@Configuration

public class SecurityConfig {

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http.csrf().disable()

.authorizeRequests()

.anyRequest().permitAll(); // allow all requests

return http.build();

}

}

AuthController.java:

package com.example.jwtauth.controller;

import com.example.jwtauth.model.AuthResponse;

import com.example.jwtauth.util.JwtUtil;

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

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

import javax.servlet.http.HttpServletRequest;

import java.util.Base64;

@RestController

public class AuthController {

@Autowired

private JwtUtil jwtUtil;

@GetMapping("/authenticate")

public AuthResponse authenticate(HttpServletRequest request) {

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

if (header != null && header.startsWith("Basic ")) {

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

String credentials = new String(Base64.getDecoder().decode(base64Credentials));

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

String username = userDetails[0];

String password = userDetails[1];

// For simplicity, hardcoded validation

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

String token = jwtUtil.generateToken(username);

return new AuthResponse(token);

}

}

throw new RuntimeException("Invalid Credentials");

}

}

AuthResponse.java:

package com.example.jwtauth.model;

public class AuthResponse {

private String token;

public AuthResponse(String token) {

this.token = token;

}

public String getToken() {

return token;

}

public void setToken(String token) {

this.token = token;

}

}

JwtAuthenticationFilter.java:

package com.example.jwtauth.filter;

import io.jsonwebtoken.Claims;

import io.jsonwebtoken.Jwts;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

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

import org.springframework.web.filter.OncePerRequestFilter;

import javax.servlet.FilterChain;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.util.Collections;

public class JwtAuthenticationFilter extends OncePerRequestFilter {

private final String SECRET\_KEY = "secret"; // same as used in JwtUtil

*@*Override

protected void doFilterInternal(HttpServletRequest request,

HttpServletResponse response,

FilterChain filterChain)

throws ServletException, IOException {

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

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

String token = header.substring(7);

try {

Claims claims = Jwts.parser()

.setSigningKey(SECRET\_KEY)

.parseClaimsJws(token)

.getBody();

String username = claims.getSubject();

if (username != null) {

UsernamePasswordAuthenticationToken auth =

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

SecurityContextHolder.getContext().setAuthentication(auth);

}

} catch (Exception e) {

throw new RuntimeException("Invalid JWT Token");

}

}

filterChain.doFilter(request, response);

}

}

AuthRequest.java:

package com.example.jwtauth.model;

public class AuthRequest {

private String username;

private String password;

public AuthRequest() {}

public AuthRequest(String username, String password) {

this.username = username;

this.password = password;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

}

application.properties:

server.port=8090

Tomcat started on port(s): 8090 (http) with context path ''

Started JwtAuthServiceApplication in 2.345 seconds (JVM running for 3.123)