Pom.xml:

<dependencies>

<!-- Web & Security -->

<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>

<!-- JSON Web Token (JJWT) -->

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.1</version>

</dependency>

</dependencies>

Application.yml:

server:

port: 8080

spring:

security:

jwt:

secret: my-secret-key-1234567890

JwtConfig.java:

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

import org.springframework.context.annotation.Configuration;

@Configuration

public class JwtConfig {

@Value("${spring.security.jwt.secret}")

private String secret;

public String getSecret() {

return secret;

}

}

**JwtTokenProvider.java**

import io.jsonwebtoken.\*;

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

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.Authentication;

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

import org.springframework.stereotype.Component;

import java.util.Collections;

import java.util.Date;

@Component

public class JwtTokenProvider {

@Autowired

private JwtConfig jwtConfig;

public String createToken(String username) {

Claims claims = Jwts.claims().setSubject(username);

Date now = new Date();

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

return Jwts.builder()

.setClaims(claims)

.setIssuedAt(now)

.setExpiration(expiry)

.signWith(SignatureAlgorithm.HS256, jwtConfig.getSecret())

.compact();

}

public boolean validateToken(String token) {

try {

Jwts.parser().setSigningKey(jwtConfig.getSecret()).parseClaimsJws(token);

return true;

} catch (JwtException | IllegalArgumentException e) {

return false;

}

}

public Authentication getAuthentication(String token) {

String username = Jwts.parser()

.setSigningKey(jwtConfig.getSecret())

.parseClaimsJws(token)

.getBody()

.getSubject();

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

}

}

JwtTokenFilter.java:

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.core.Authentication;

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

import org.springframework.web.filter.OncePerRequestFilter;

import java.io.IOException;

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)) {

Authentication auth = jwtTokenProvider.getAuthentication(token);

SecurityContextHolder.getContext().setAuthentication(auth);

}

filterChain.doFilter(request, response);

}

private String resolveToken(HttpServletRequest request) {

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

return (bearerToken != null && bearerToken.startsWith("Bearer "))

? bearerToken.substring(7)

: null;

}

}

SecurityConfig.java:

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.web.SecurityFilterChain;

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

import org.springframework.security.config.http.SessionCreationPolicy;

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

@Configuration

@EnableWebSecurity

public class SecurityConfig {

@Autowired

private JwtTokenFilter jwtTokenFilter;

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

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

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

.authorizeHttpRequests(auth -> auth

.requestMatchers("/auth/\*\*").permitAll()

.anyRequest().authenticated()

)

.addFilterBefore(jwtTokenFilter, UsernamePasswordAuthenticationFilter.class);

return http.build();

}

}

AuthController.java:

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

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

@RestController

@RequestMapping("/auth")

public class AuthController {

@Autowired

private JwtTokenProvider jwtTokenProvider;

@PostMapping("/login")

public String login(@RequestParam String username) {

// Simulate authentication — in real cases, verify username/password

return jwtTokenProvider.createToken(username);

}

}

SecureController.java:

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

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

@RestController

public class SecureController {

@GetMapping("/secure")

public String securedData() {

return This is a protected endpoint. You are authenticated with JWT.";

}

}