**Exercise 3: Using JSON Web Tokens (JWT) for Secure Communication**

**Application.yml:**

spring:

security:

jwt:

secret: mysecretkey123456

**JwtDemoApplication.java:**

**package** com.example.jwt\_demo;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

**public** **class** JwtDemoApplication {

**public** **static** **void** main(String[] args) {

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

}

}

**JwtConfig.java:**

**package** com.example.jwt\_demo.config;

**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;

}

}

**SecurityConfig.java:**

**package** com.example.jwt\_demo.config;

**import** com.example.jwt\_demo.security.JwtTokenFilter;

**import** org.springframework.context.annotation.Bean;

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

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

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

**import** org.springframework.security.web.SecurityFilterChain;

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

@EnableWebSecurity

**public** **class** SecurityConfig {

@Bean

**public** JwtTokenFilter jwtTokenFilter() {

**return** **new** JwtTokenFilter();

}

@Bean

**public** SecurityFilterChain filterChain(HttpSecurity http) **throws** Exception {

http.csrf().disable()

.sessionManagement().sessionCreationPolicy(SessionCreationPolicy.***STATELESS***) // no sessions

.and()

.authorizeHttpRequests(auth -> auth

.requestMatchers("/login").permitAll() // allow login without token

.anyRequest().authenticated()

)

.addFilterBefore(jwtTokenFilter(), UsernamePasswordAuthenticationFilter.**class**);

**return** http.build();

}

}

**AuthController.java:**

**package** com.example.jwt\_demo.controller;

**import** com.example.jwt\_demo.security.JwtTokenProvider;

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

**import** org.springframework.http.ResponseEntity;

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

@RestController

**public** **class** AuthController {

@Autowired

**private** JwtTokenProvider jwtTokenProvider;

@PostMapping("/login")

**public** ResponseEntity<?> login(@RequestParam String username, @RequestParam String password) {

// In real app: validate username and password from DB or other source

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

String token = jwtTokenProvider.createToken(username);

**return** ResponseEntity.*ok*("Bearer " + token);

}

**return** ResponseEntity.*status*(401).body("Invalid credentials");

}

}

**HelloController.java:**

**package** com.example.jwt\_demo.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, secured world!";

}

}

**JwtTokenFilter.java:**

**package** com.example.jwt\_demo.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.security.web.authentication.WebAuthenticationDetailsSource;

**import** org.springframework.web.filter.OncePerRequestFilter;

**import** java.io.IOException;

**import** java.util.Collections;

**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)) {

String username = jwtTokenProvider.getUsername(token);

UsernamePasswordAuthenticationToken auth = **new** UsernamePasswordAuthenticationToken(

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

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

SecurityContextHolder.getContext().setAuthentication(auth);

}

filterChain.doFilter(request, response);

}

**private** String resolveToken(HttpServletRequest request) {

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

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

**return** bearer.substring(7);

}

**return** **null**;

}

}

**JwtTokenProvider.java:**

**package** com.example.jwt\_demo.security;

**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** com.example.jwt\_demo.config.JwtConfig;

**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 expiry

**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** (Exception e) {

**return** **false**;

}

}

**public** String getUsername(String token) {

**return** Jwts.*parser*()

.setSigningKey(jwtConfig.getSecret())

.parseClaimsJws(token)

.getBody()

.getSubject();

}

}

**OUTPUT:**





