**JWT**

**HandsOn**

Create authentication service that returns JWT

**DemoApplication.java**

**package** com.example.demo;

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

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

@SpringBootApplication

**public** **class** DemoApplication {

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

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

}

}

**JwtAuthApplication.java:**

package com.example.demo;

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);

}

}

SecurityConfig.java

**package** com.example.demo.config;

**import** com.example.demo.filter.JwtAuthorizationFilter;

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

**import** org.springframework.security.authentication.AuthenticationManager;

**import**org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

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

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

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

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

@Configuration

@EnableWebSecurity

**public** **class** SecurityConfig **extends** WebSecurityConfigurerAdapter

@Override

**protected** **void** configure(AuthenticationManagerBuilder auth) **throws** Exception {

auth.inMemoryAuthentication()

.withUser("admin").password(passwordEncoder().encode("pwd")).roles("ADMIN")

.and()

.withUser("user").password(passwordEncoder().encode("pwd")).roles("USER");

}

@Override

**protected** **void** configure(HttpSecurity http) **throws** Exception {

http.csrf().disable().httpBasic()

.and()

.authorizeRequests()

.antMatchers("/authenticate").hasAnyRole("USER", "ADMIN")

.anyRequest().authenticated()

.and()

.addFilter(**new** JwtAuthorizationFilter(authenticationManager()));

}

@Bean

**public** PasswordEncoder passwordEncoder() {

**return** **new** BCryptPasswordEncoder();

}

}

**AuthenticationController.java**

package com.example.demo.controller;

import io.jsonwebtoken.\*;

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

import java.util.\*;

@RestController

public class AuthenticationController {

@GetMapping("/authenticate")

public Map<String, String> authenticate(@RequestHeader("Authorization") String authHeader) {

String user = getUser(authHeader);

String token = generateJwt(user);

Map<String, String> response = new HashMap<>();

response.put("token", token);

return response;

}

private String getUser(String authHeader) {

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

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

String decoded = new String(decodedBytes);

return decoded.split(":")[0];

}

private String generateJwt(String user) {

return Jwts.*builder*()

.setSubject(user)

.setIssuedAt(new Date())

.setExpiration(new Date(System.*currentTimeMillis*() + 1200000)) // 20 mins

.signWith(SignatureAlgorithm.*HS256*, "secretkey")

.compact();

}

}

CountryController.java

package com.example.demo.controller;

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

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

import java.util.Arrays;

import java.util.List;

@RestController

public class CountryController {

@GetMapping("/countries")

public List<String> getCountries() {

return Arrays.*asList*("India", "USA", "Germany");

}

}

**JwtAuthorizationFilter.java**

package com.example.demo.filter;

import io.jsonwebtoken.\*;

import org.springframework.security.authentication.\*;

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

import org.springframework.security.web.authentication.www.BasicAuthenticationFilter;

import javax.servlet.\*;

import javax.servlet.http.\*;

import java.io.IOException;

import java.util.\*;

public class JwtAuthorizationFilter extends BasicAuthenticationFilter {

public JwtAuthorizationFilter(AuthenticationManager authenticationManager) {

super(authenticationManager);

}

@Override

protected void doFilterInternal(HttpServletRequest request,

HttpServletResponse response,

FilterChain chain)

throws IOException, ServletException {

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

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

chain.doFilter(request, response);

return;

}

UsernamePasswordAuthenticationToken authentication = getAuthentication(request);

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

chain.doFilter(request, response);

}

private UsernamePasswordAuthenticationToken getAuthentication(HttpServletRequest request) {

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

if (token != null) {

try {

Claims claims = Jwts.*parser*()

.setSigningKey("secretkey")

.parseClaimsJws(token.replace("Bearer ", ""))

.getBody();

String user = claims.getSubject();

if (user != null) {

return new UsernamePasswordAuthenticationToken(user, null, new ArrayList<>());

}

} catch (JwtException e) {

return null;

}

}

return null;

}

}

**OUTPUT:**

