**6.Create authentication service that returns JWT**

**JwtUtil.java**

package com.cognizant.spring\_learn.service;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

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

import org.springframework.stereotype.Service;

import java.util.Date;

@Service

public class JwtUtil {

private final String SECRET\_KEY = "mysecretkey";

private final long EXPIRATION\_TIME = 1000 \* 60 \* 60; // 1 hour

public String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date())

.setExpiration(new Date(System.currentTimeMillis() + EXPIRATION\_TIME))

.signWith(SignatureAlgorithm.HS256, SECRET\_KEY)

.compact();

}

public String extractUsername(String token) {

return Jwts.parser()

.setSigningKey(SECRET\_KEY)

.parseClaimsJws(token)

.getBody()

.getSubject();

}

public boolean validateToken(String token, UserDetails userDetails) {

final String username = extractUsername(token);

return username.equals(userDetails.getUsername());

}

}

AuthenticationController.java

package com.cognizant.spring\_learn.controller;

import com.cognizant.spring\_learn.service.JwtUtil;

import org.springframework.http.ResponseEntity;

import org.springframework.security.core.Authentication;

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

import java.util.HashMap;

import java.util.Map;

@RestController

public class AuthenticationController {

private final JwtUtil jwtUtil;

public AuthenticationController(JwtUtil jwtUtil) {

this.jwtUtil = jwtUtil;

}

@GetMapping("/authenticate")

public ResponseEntity<?> authenticate(Authentication authentication) {

String username = authentication.getName(); // From Basic Auth

String token = jwtUtil.generateToken(username);

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

response.put("token", token);

return ResponseEntity.ok(response);

}

**JwtRequestFilter.java**

package com.cognizant.spring\_learn.filter;

import com.cognizant.spring\_learn.service.JwtUtil;

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.\*;

import org.springframework.stereotype.Component;

import org.springframework.web.filter.OncePerRequestFilter;

import java.io.IOException;

@Component

public class JwtRequestFilter extends OncePerRequestFilter {

@Autowired

private JwtUtil jwtUtil;

@Autowired

private UserDetailsService userDetailsService;

@Override

protected void doFilterInternal(HttpServletRequest request,

HttpServletResponse response,

FilterChain chain)

throws ServletException, IOException {

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

String username = null;

String jwt = null;

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

jwt = authHeader.substring(7);

try {

username = jwtUtil.extractUsername(jwt);

} catch (Exception e) {

System.out.println("Invalid JWT: " + e.getMessage());

}

}

if (username != null && SecurityContextHolder.getContext().getAuthentication() == null) {

UserDetails userDetails = userDetailsService.loadUserByUsername(username);

if (jwtUtil.validateToken(jwt, userDetails)) {

UsernamePasswordAuthenticationToken token =

new UsernamePasswordAuthenticationToken(userDetails, null, userDetails.getAuthorities());

SecurityContextHolder.getContext().setAuthentication(token);

}

}

chain.doFilter(request, response);

}

}

**SecurityConfig.java**

package com.cognizant.spring\_learn.config;

import com.cognizant.spring\_learn.filter.JwtRequestFilter;

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.core.userdetails.\*;

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

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

import org.springframework.security.web.SecurityFilterChain;

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

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

@Configuration

public class SecurityConfig {

@Autowired

private JwtRequestFilter jwtRequestFilter;

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http.csrf().disable()

.authorizeHttpRequests()

.requestMatchers("/authenticate").authenticated()

.anyRequest().authenticated()

.and()

.httpBasic();

http.addFilterBefore(jwtRequestFilter, UsernamePasswordAuthenticationFilter.class);

return http.build();

}

@Bean

public UserDetailsService userDetailsService() {

UserDetails user = User.withUsername("user")

.password("pwd")

.roles("USER")

.build();

return new InMemoryUserDetailsManager(user);

}

@Bean

public PasswordEncoder passwordEncoder() {

return NoOpPasswordEncoder.getInstance(); // For demo only

}

}

**HelloController.java**

package com.cognizant.spring\_learn.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, secure world!";

}

}

**Output:**

