**Week 4 : JSON Web Tokens(JWT)**

**Create authentication service that returns JWT**

**Code :**

**SecurityConfig.java**

package com.cognizant.jwt\_authentication.config;

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

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

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

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

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

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

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

import org.springframework.security.web.SecurityFilterChain;

*@Configuration*

public class SecurityConfig {

*@Bean*

public AuthenticationManager authenticationManager(AuthenticationConfiguration config) throws Exception {

return config.getAuthenticationManager();

}

*@Bean*

public PasswordEncoder passwordEncoder() {

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

}

*@Bean*

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

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

.authorizeHttpRequests(auth -> auth

.requestMatchers("/authenticate").permitAll()

.anyRequest().authenticated())

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

return http.build();

}

}

**AuthController.java**

package com.cognizant.jwt\_authentication.controller;

import com.cognizant.jwt\_authentication.model.\*;

import com.cognizant.jwt\_authentication.util.JwtUtil;

import com.cognizant.jwt\_authentication.service.JwtUserDetailsService;

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

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

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

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

*@RestController*

public class AuthController {

*@Autowired*

private AuthenticationManager authManager;

*@Autowired*

private JwtUserDetailsService userDetailsService;

*@Autowired*

private JwtUtil jwtUtil;

*@PostMapping*("/authenticate")

public AuthenticationResponse generateToken(*@RequestBody* AuthenticationRequest request) throws Exception {

try {

authManager.authenticate(

new UsernamePasswordAuthenticationToken(request.getUsername(), request.getPassword()));

} catch (BadCredentialsException e) {

throw new Exception("Invalid username or password", e);

}

UserDetails userDetails = userDetailsService.loadUserByUsername(request.getUsername());

String token = jwtUtil.generateToken(userDetails);

return new AuthenticationResponse(token);

}

}

**AuthenticationRequest.java**

package com.cognizant.jwt\_authentication.model;

public class AuthenticationRequest {

private String username;

private String password;

public AuthenticationRequest() {}

public AuthenticationRequest(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; }

}

**AuthenticationResponse.java**

package com.cognizant.jwt\_authentication.model;

public class AuthenticationResponse {

private String jwt;

public AuthenticationResponse() {}

public AuthenticationResponse(String jwt) {

this.jwt = jwt;

}

public String getJwt() { return jwt; }

public void setJwt(String jwt) { this.jwt = jwt; }

}

**UserModel.java**

package com.cognizant.jwt\_authentication.model;

public class UserModel {

private String username;

private String password;

public UserModel() {}

public UserModel(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; }

}

**JwtUserDetailsService.java**

package com.cognizant.jwt\_authentication.service;

import org.springframework.security.core.userdetails.\*;

import org.springframework.stereotype.Service;

import java.util.\*;

import org.springframework.security.core.authority.SimpleGrantedAuthority;

*@Service*

public class JwtUserDetailsService implements UserDetailsService {

*@Override*

public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {

if ("user1".equals(username)) {

return new User("user1", "password1", new ArrayList<>());

} else {

throw new UsernameNotFoundException("User not found with username: " + username);

}

}

}

**JwtUtil.java**

package com.cognizant.jwt\_authentication.util;

import io.jsonwebtoken.\*;

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

import org.springframework.stereotype.Component;

import java.util.\*;

import java.util.function.Function;

*@Component*

public class JwtUtil {

private String SECRET\_KEY = "secret";

public String extractUsername(String token) {

return extractClaim(token, Claims::getSubject);

}

public Date extractExpiration(String token) {

return extractClaim(token, Claims::getExpiration);

}

public <T> T extractClaim(String token, Function<Claims, T> claimsResolver) {

return claimsResolver.apply(Jwts.parser().setSigningKey(SECRET\_KEY).parseClaimsJws(token).getBody());

}

public String generateToken(UserDetails userDetails) {

return Jwts.builder()

.setSubject(userDetails.getUsername())

.setIssuedAt(new Date())

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

.signWith(SignatureAlgorithm.HS256, SECRET\_KEY)

.compact();

}

public boolean validateToken(String token, UserDetails userDetails) {

return extractUsername(token).equals(userDetails.getUsername()) && !isTokenExpired(token);

}

private boolean isTokenExpired(String token) {

return extractExpiration(token).before(new Date());

}

}

**application.properties**

spring.application.name=jwt\_authentication

server.port=8085

**JwtAuthenticationApplication.java**

package com.cognizant.jwt\_authentication;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@SpringBootApplication*

public class JwtAuthenticationApplication {

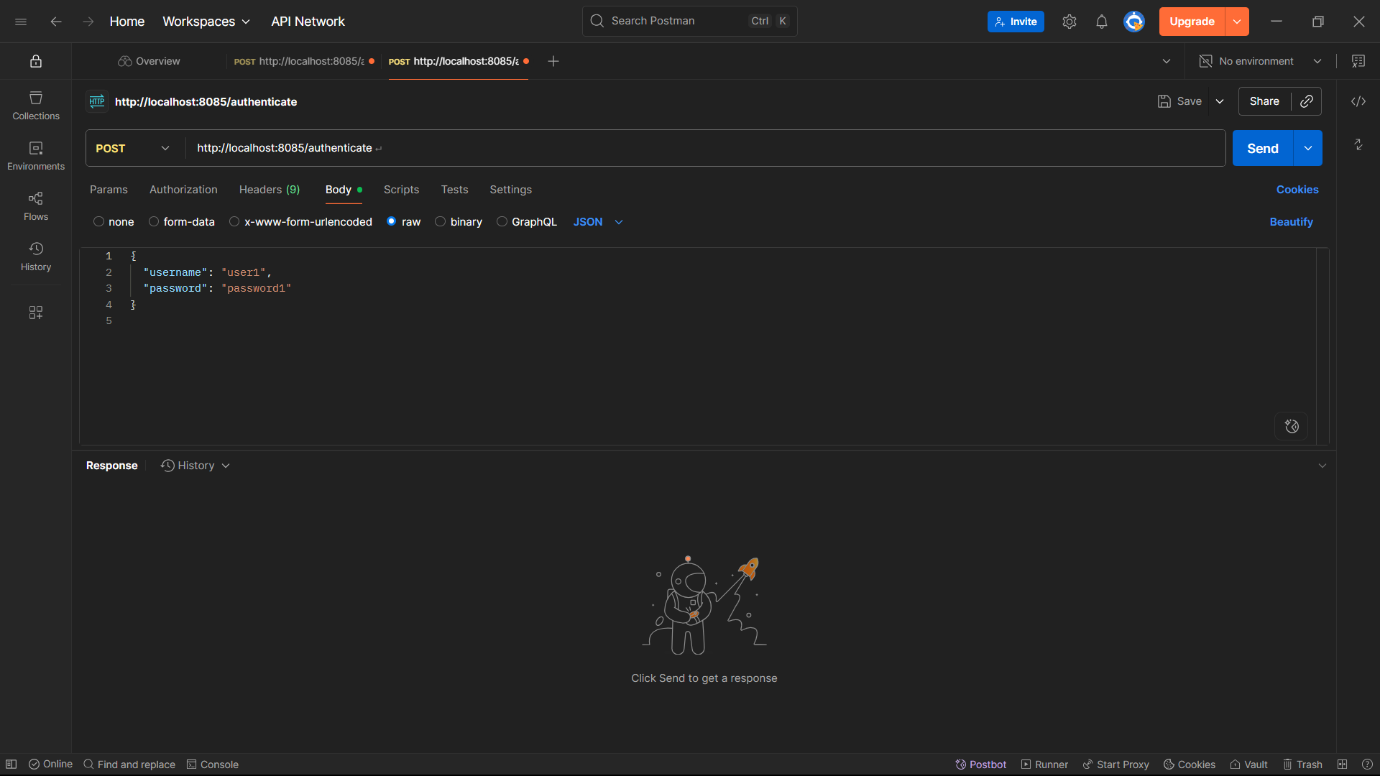
public static void main(String[] args) {

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

}

}

**OUTPUT:**

****