**JWT-Handons**

**Create authentication controller and configure it in SecurityConfig**

**SpringLearnApplication.java**

package com.cognizant.spring\_learn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

private static final Logger LOGGER = LoggerFactory.getLogger(SpringLearnApplication.class);

public static void main(String[] args) {

LOGGER.info("=== SpringLearnApplication Started ===");

SpringApplication.run(SpringLearnApplication.class, args);

LOGGER.info("=== SpringLearnApplication Ended ===");

}

}

**Pom.xml**

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.1</version>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

**JwtUtil.java**

package com.cognizant.spring\_learn.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import org.springframework.stereotype.Component;

import java.util.Date;

@Component

public class JwtUtil {

private final String secret = "mySecretKey";

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

public String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date(System.currentTimeMillis()))

.setExpiration(new Date(System.currentTimeMillis() + expiration))

.signWith(SignatureAlgorithm.HS256, secret)

.compact();

}

}

**AuthController.java**

package com.cognizant.spring\_learn.controller;

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

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

import org.springframework.http.ResponseEntity;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.Authentication;

import org.springframework.security.core.AuthenticationException;

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

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

import java.util.Base64;

@RestController

public class AuthController {

@Autowired

private JwtUtil jwtUtil;

@GetMapping("/authenticate")

public ResponseEntity<?> authenticate(@RequestHeader("Authorization") String authHeader) {

try {

// Remove Basic prefix

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

String decoded = new String(Base64.getDecoder().decode(encodedCredentials));

String[] parts = decoded.split(":");

String username = parts[0];

String password = parts[1];

// Dummy check (In real-world, check from DB)

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

String token = jwtUtil.generateToken(username);

return ResponseEntity.ok().body("{\"token\": \"" + token + "\"}");

} else {

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

}

} catch (Exception e) {

return ResponseEntity.status(400).body("Bad Request: " + e.getMessage());

}

}

}

**SecurityConfig.java**

package com.cognizant.spring\_learn.config;

import org.springframework.context.annotation.Configuration;

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

import org.springframework.security.web.SecurityFilterChain;

import org.springframework.context.annotation.Bean;

@Configuration

public class SecurityConfig {

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

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

.authorizeHttpRequests(auth -> auth

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

.anyRequest().authenticated()

)

.httpBasic(); // Needed for Basic Auth

return http.build();

}

}





