**JWT - Handson**

**1. Create authentication service that returns JWT**  
  
Step 1: Add Dependencies in **pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

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

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

<version>3.5.3</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-learn</name>

<description>Demo project for Spring Boot</description> <url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/> </scm>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

<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>

</dependencies>

<build>

<plugins>

<plugin>

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

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

Step 2: **application.properties**

spring.application.name=spring\_learn

server.port=4234

logging.level.root=DEBUG

Step 3: Create a **JwtUtil.java** class in **spring-learn/src/main/java/com/cognizant/spring\_learn/security**

package com.cognizant.spring\_learn.util;

import java.util.Date;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import org.springframework.stereotype.Component;

*@Component*

public class JwtUtil {

private final String SECRET\_KEY = "mysecretkey";

public String generateToken(String username) {

return Jwts.*builder*()

.setSubject(username)

.setIssuedAt(new Date())

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

.signWith(*SignatureAlgorithm*.***HS256***, SECRET\_KEY)

.compact();

}

}

Step 4: Create an **AuthenticationController.java** in **com.cognizant.spring\_learn.controller**

package com.cognizant.spring\_learn.controller;

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

import org.springframework.http.ResponseEntity;

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

import jakarta.servlet.http.HttpServletRequest;

import java.util.Base64;

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

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

*@RestController*

public class AuthenticationController {

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

*@Autowired*

private JwtUtil jwtUtil;

*@RequestMapping*("/authenticate")

public ResponseEntity<?> generateToken(HttpServletRequest request) {

***LOGGER***.info("START - Authentication");

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

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

String base64Credentials = authHeader.substring("Basic ".length());

byte[] decodedBytes = Base64.*getDecoder*().decode(base64Credentials);

String credentials = new String(decodedBytes);

String[] values = credentials.split(":", 2);

String username = values[0];

String password = values[1];

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

String token = jwtUtil.generateToken(username);

***LOGGER***.info("END - Authentication");

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

}

}

return ResponseEntity.*status*(401).body("Unauthorized");

}

}

Step 5: Create a **SecurityConfig.java**

package com.cognizant.spring\_learn.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;

import org.springframework.security.config.annotation.method.configuration.EnableMethodSecurity;

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

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

import org.springframework.security.web.SecurityFilterChain;

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

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

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

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

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

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

*@Configuration*

*@EnableWebSecurity*

*@EnableMethodSecurity*

public class SecurityConfig {

*@SuppressWarnings*("removal")

*@Bean*

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

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

.authorizeHttpRequests(auth -> auth

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

.anyRequest().authenticated()

)

.httpBasic()

.and()

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

return http.build();

}

*@Bean*

public UserDetailsService userDetailsService(PasswordEncoder encoder) {

return new InMemoryUserDetailsManager(

User.*withUsername*("user")

.password(encoder.encode("pwd"))

.roles("USER")

.build()

);

}

*@Bean*

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

*@Bean*

public AuthenticationManager authenticationManager(AuthenticationConfiguration authConfig) throws Exception {

return authConfig.getAuthenticationManager();

}

}

**Output:**

