**Create authentication service that returns JWT**

**CODE :**

**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/maven-4.0.0.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.authexample</groupId>  
 <artifactId>springsecurityjwt</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>springsecurityjwt</name>  
 <description>Demo project for Spring Boot JWT Authentication</description>  
  
 <properties>  
 <java.version>17</java.version>  
 </properties>  
  
 <dependencies>  
 <!-- Spring Boot Web -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 <!-- Spring Boot Security -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-security</artifactId>  
 </dependency>  
  
 <!-- JWT dependencies (latest recommended 0.11.5) -->  
 <dependency>  
 <groupId>io.jsonwebtoken</groupId>  
 <artifactId>jjwt-api</artifactId>  
 <version>0.11.5</version>  
 </dependency>  
 <dependency>  
 <groupId>io.jsonwebtoken</groupId>  
 <artifactId>jjwt-impl</artifactId>  
 <version>0.11.5</version>  
 <scope>runtime</scope>  
 </dependency>  
 <dependency>  
 <groupId>io.jsonwebtoken</groupId>  
 <artifactId>jjwt-jackson</artifactId> <!-- or jjwt-gson if you prefer -->  
 <version>0.11.5</version>  
 <scope>runtime</scope>  
 </dependency>  
  
 <!-- Jakarta Servlet API (provided scope for server) -->  
 <dependency>  
 <groupId>jakarta.servlet</groupId>  
 <artifactId>jakarta.servlet-api</artifactId>  
 <version>6.0.0</version>  
 <scope>provided</scope>  
 </dependency>  
  
 <!-- JAXB API and runtime for XML binding -->  
 <dependency>  
 <groupId>jakarta.xml.bind</groupId>  
 <artifactId>jakarta.xml.bind-api</artifactId>  
 <version>4.0.0</version>  
 </dependency>  
 <dependency>  
 <groupId>org.glassfish.jaxb</groupId>  
 <artifactId>jaxb-runtime</artifactId>  
 <version>4.0.0</version>  
 </dependency>  
  
 <!-- Spring Boot DevTools for hot reload (optional) -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-devtools</artifactId>  
 <scope>runtime</scope>  
 <optional>true</optional>  
 </dependency>  
  
 <!-- Test dependencies -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.security</groupId>  
 <artifactId>spring-security-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <!-- Spring Boot Maven Plugin -->  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 <version>3.5.3</version>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

**application.properties**

spring.application.name=springsecurityjwt  
server.port=8090

**SecurityConfig.java**

package com.authexample.springsecurityjwt.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.web.builders.HttpSecurity;  
import org.springframework.security.core.userdetails.User;  
import org.springframework.security.core.userdetails.UserDetailsService;  
import org.springframework.security.crypto.password.NoOpPasswordEncoder;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import org.springframework.security.provisioning.InMemoryUserDetailsManager;  
import org.springframework.security.web.SecurityFilterChain;  
  
@Configuration  
public class SecurityConfig {  
  
 @Bean  
 public UserDetailsService userDetailsService() {  
 var user = User.*withUsername*("user")  
 .password("pwd")  
 .roles("USER")  
 .build();  
 return new InMemoryUserDetailsManager(user);  
 }  
  
 @Bean  
 public PasswordEncoder passwordEncoder() {  
 return NoOpPasswordEncoder.*getInstance*(); // for simplicity only  
 }  
  
 @Bean  
 public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {  
 http.csrf(csrf -> csrf.disable())  
 .authorizeHttpRequests(auth -> auth  
 .requestMatchers("/authenticate").authenticated()  
 .anyRequest().permitAll())  
 .httpBasic(httpBasic -> {});  
  
 return http.build();  
 }  
  
 @Bean  
 public AuthenticationManager authenticationManager(AuthenticationConfiguration config) throws Exception {  
 return config.getAuthenticationManager();  
 }  
}

**AuthController.java**

package com.authexample.springsecurityjwt.controller;  
  
import com.authexample.springsecurityjwt.service.JwtUtil;  
import com.authexample.springsecurityjwt.model.AuthResponse;  
import org.springframework.web.bind.annotation.\*;  
  
import jakarta.servlet.http.HttpServletRequest;  
import java.util.Base64;  
  
@RestController  
public class AuthController {  
  
 @GetMapping("/authenticate")  
 public AuthResponse authenticate(HttpServletRequest request) {  
 String authHeader = request.getHeader("Authorization");  
  
 if (authHeader == null || !authHeader.startsWith("Basic ")) {  
 throw new RuntimeException("Missing or invalid Authorization header");  
 }  
  
 String base64Credentials = authHeader.substring("Basic ".length()).trim();  
 byte[] credDecoded = Base64.*getDecoder*().decode(base64Credentials);  
 String[] credentials = new String(credDecoded).split(":", 2);  
  
 String username = credentials[0];  
  
 String token = JwtUtil.*generateToken*(username);  
 return new AuthResponse(token);  
 }  
}

**AuthResponse.java**

package com.authexample.springsecurityjwt.model;  
  
public class AuthResponse {  
 private String token;  
  
 public AuthResponse(String token) {  
 this.token = token;  
 }  
  
 public String getToken() {  
 return token;  
 }  
}

**JwtUtil.java**

package com.authexample.springsecurityjwt.service;  
  
import io.jsonwebtoken.Jwts;  
import io.jsonwebtoken.security.Keys;  
  
import javax.crypto.SecretKey;  
import java.nio.charset.StandardCharsets;  
import java.util.Date;  
  
public class JwtUtil {  
  
 private static final String *SECRET\_KEY* = "mysecretkeymysecretkeymysecretkey12"; // at least 256-bit key (32 chars)  
  
 // Generate SecretKey from plain text secret  
 private static SecretKey getSigningKey() {  
 byte[] keyBytes = *SECRET\_KEY*.getBytes(StandardCharsets.*UTF\_8*);  
 return Keys.*hmacShaKeyFor*(keyBytes);  
 }  
  
 public static String generateToken(String username) {  
 return Jwts.*builder*()  
 .setSubject(username)  
 .setIssuedAt(new Date())  
 .setExpiration(new Date(System.*currentTimeMillis*() + 1000 \* 60 \* 60)) // 1 hour  
 .signWith(*getSigningKey*()) // <-- updated method  
 .compact();  
 }  
}

**SpringsecurityjwtApplication.java**

package com.authexample.springsecurityjwt;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class SpringsecurityjwtApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(SpringsecurityjwtApplication.class, args);  
 }  
  
}

**OUTPUT :**





