**Create authentication service that returns JWTJwtService.java:**

package com.cognizant.springsecurity.service;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

import org.springframework.stereotype.Service;

import java.nio.charset.StandardCharsets;

import java.security.Key;

import java.util.Base64;

import java.util.Date;

*@Service*

public class JwtService {

private static final String ***SECRET*** = "ThisIsASecretKeyUsedForHS256MustBeAtLeast32Characters!";

private static final Key ***SECRET\_KEY*** = Keys.*hmacShaKeyFor*(***SECRET***.getBytes(StandardCharsets.***UTF\_8***));

public String generateTokenFromBasicAuth(String authHeader) {

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

throw new RuntimeException("Missing or invalid Authorization header");

}

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

String credentials = new String(Base64.*getDecoder*().decode(base64Credentials));

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

if (values.length != 2) throw new RuntimeException("Invalid basic auth format");

String username = values[0];

String password = values[1];

if (!username.equals("user") || !password.equals("pwd")) {

throw new RuntimeException("Invalid credentials");

}

return Jwts.*builder*()

.setSubject(username)

.setIssuedAt(new Date())

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

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

.compact();

}}

**SecurityConfig.java:**

package com.cognizant.springsecurity.config;

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.web.SecurityFilterChain;

*@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();

return http.build();

}

}

**AuthController.java:**

package com.cognizant.springsecurity.controller;

import io.jsonwebtoken.JwtBuilder;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import java.util.Base64;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

import io.jsonwebtoken.security.Keys;

import java.nio.charset.StandardCharsets;

import java.security.Key;

*@RestController*

public class AuthController {

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

*@GetMapping*("/authenticate")

public Map<String, String> authenticate(*@RequestHeader*("Authorization") String authHeader) {

***LOGGER***.info("Start of authenticate()");

***LOGGER***.debug("Authorization header: {}", authHeader);

String user = getUser(authHeader);

***LOGGER***.debug("Extracted user from header: {}", user);

String token = generateJwt(user);

***LOGGER***.debug("Generated token: {}", token);

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

map.put("token", token);

***LOGGER***.info("End of authenticate()");

return map;

}

private String getUser(String authHeader) {

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

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

String decodedString = new String(decodedBytes);

return decodedString.split(":")[0];

}

private String generateJwt(String user) {

byte[] keyBytes = "ThisIsASecretKeyThatIsAtLeast32BytesLong!".getBytes(StandardCharsets.***UTF\_8***);

Key key = Keys.*hmacShaKeyFor*(keyBytes);

return Jwts.builder()

.setSubject(user)

.setIssuedAt(new Date())

.setExpiration(new Date(System.*currentTimeMillis*() + 20 \* 60 \* 1000)) // 20 minutes

.signWith(key, SignatureAlgorithm.HS256)

.compact();

}}

**SpringsecurityApplication.java:**

package com.cognizant.springsecurity;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@SpringBootApplication*

public class SpringsecurityApplication {

public static void main(String[] args) {

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

}}

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/>

</parent>

<groupId>com.cognizant</groupId>

<artifactId>springsecurity</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>springsecurity</name>

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

<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-starter-security</artifactId>

</dependency>

<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>

<version>0.11.5</version>

<scope>runtime</scope>

</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>org.springframework.security</groupId>

<artifactId>spring-security-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

