**Create authentication service that returns JWT** 

**Pom.xml**

<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>

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<parent>

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

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

<version>2.7.18</version>

</parent>

<dependencies>

<!-- Spring Web -->

<dependency>

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

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

</dependency>

<!-- Spring Security -->

<dependency>

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

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

</dependency>

<!-- JWT -->

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.0</version>

</dependency>

<!-- Spring Context (for applicationContext.xml) -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

</dependency>

<!-- Logging -->

<dependency>

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

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

</dependency>

<!-- Testing -->

<dependency>

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

<artifactId>spring-boot-starter-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>

**AuthenticationController.java:**

package com.cognizant.spring\_learn.controller;

import java.util.Base64;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RequestHeader;

import org.springframework.web.bind.annotation.RestController;

import io.jsonwebtoken.JwtBuilder;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

@RestController

public class AuthenticationController {

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

@GetMapping("/authenticate")

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

LOGGER.info("Start authenticate");

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

String user = getUser(authHeader);

String token = generateJwt(user);

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

map.put("token", token);

LOGGER.info("End authenticate");

return map;

}

private String getUser(String authHeader) {

LOGGER.info("Start getUser");

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

byte[] decoded = Base64.getDecoder().decode(encoded);

String decodedString = new String(decoded);

String user = decodedString.split(":")[0];

LOGGER.debug("Decoded user: {}", user);

LOGGER.info("End getUser");

return user;

}

private String generateJwt(String user) {

LOGGER.info("Start generateJwt");

JwtBuilder builder = Jwts.builder();

builder.setSubject(user);

builder.setIssuedAt(new Date());

builder.setExpiration(new Date(System.currentTimeMillis() + 20 \* 60 \* 1000)); // 20 min

builder.signWith(SignatureAlgorithm.HS256, "secretkey");

String token = builder.compact();

LOGGER.info("End generateJwt");

return token;

}

}

**JwtAuthorizationFilter.java:**

package com.cognizant.spring\_learn.security;

import java.io.IOException;

import java.util.ArrayList;

import javax.servlet.FilterChain;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import io.jsonwebtoken.Claims;

import io.jsonwebtoken.Jws;

import io.jsonwebtoken.JwtException;

import io.jsonwebtoken.Jwts;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

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

import org.springframework.security.web.authentication.www.BasicAuthenticationFilter;

public class JwtAuthorizationFilter extends BasicAuthenticationFilter {

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

public JwtAuthorizationFilter(AuthenticationManager authenticationManager) {

super(authenticationManager);

}

@Override

protected void doFilterInternal(HttpServletRequest req,

HttpServletResponse res,

FilterChain chain)

throws IOException, ServletException {

LOGGER.info("Start doFilterInternal");

String header = req.getHeader("Authorization");

LOGGER.debug("Header: {}", header);

if (header == null || !header.startsWith("Bearer ")) {

chain.doFilter(req, res);

return;

}

UsernamePasswordAuthenticationToken authentication = getAuthentication(req);

SecurityContextHolder.getContext().setAuthentication(authentication);

chain.doFilter(req, res);

LOGGER.info("End doFilterInternal");

}

private UsernamePasswordAuthenticationToken getAuthentication(HttpServletRequest request) {

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

if (token != null) {

try {

Jws<Claims> jws = Jwts.parser()

.setSigningKey("secretkey")

.parseClaimsJws(token.replace("Bearer ", ""));

String user = jws.getBody().getSubject();

LOGGER.debug("JWT user: {}", user);

if (user != null) {

return new UsernamePasswordAuthenticationToken(user, null, new ArrayList<>());

}

} catch (JwtException e) {

LOGGER.error("JWT Error", e);

return null;

}

}

return null;

}

}

**SecurityConfig.java**

package com.cognizant.spring\_learn.security;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

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

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

import org.springframework.security.authentication.AuthenticationManager;

@Configuration

public class SecurityConfig extends WebSecurityConfigurerAdapter {

@Override

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.inMemoryAuthentication()

.withUser("user").password("{noop}pwd").roles("USER")

.and()

.withUser("admin").password("{noop}pwd").roles("ADMIN");

}

@Override

protected void configure(HttpSecurity http) throws Exception {

http.csrf().disable().httpBasic()

.and().authorizeRequests()

.antMatchers("/authenticate").hasAnyRole("USER", "ADMIN")

.anyRequest().authenticated()

.and()

.addFilter(new JwtAuthorizationFilter(authenticationManager()));

}

@Bean

public AuthenticationManager customAuthenticationManager() throws Exception {

return authenticationManager();

}

}

OUTPUT:

A screenshot of a computer

AI-generated content may be incorrect.