**Create authentication service that returns JWT**

**Solution:**

**Step 1:** Create Spring Boot Project

**Step 2:** Add JWT 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/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.cognizant.jwt</groupId>

<artifactId>jwt-auth-service</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>jwt-auth-service</name>

<description>Demo project for JWT</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-security</artifactId>

</dependency>

<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>org.springframework.security</groupId>

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

<scope>test</scope>

</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>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

**Step 3:** Configure Spring Security (SecurityConfig.java)

package com.cognizant.jwt.security;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.AuthenticationManagerResolver;

import org.springframework.security.config.Customizer;

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.core.userdetails.User;

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

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

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

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

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.security.web.SecurityFilterChain;

@Configuration

@EnableWebSecurity

@EnableMethodSecurity

public class SecurityConfig {

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

@Bean

public UserDetailsService userDetailsService(PasswordEncoder passwordEncoder) {

UserDetails admin = User.*withUsername*("admin")

.password(passwordEncoder.encode("pwd"))

.roles("ADMIN")

.build();

UserDetails user = User.*withUsername*("user")

.password(passwordEncoder.encode("pwd"))

.roles("USER")

.build();

return new InMemoryUserDetailsManager(admin, user);

}

@Bean

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

@Bean

public AuthenticationManager authenticationManager(AuthenticationConfiguration config) throws Exception {

return config.getAuthenticationManager();

}

@Bean

public SecurityFilterChain filterChain(HttpSecurity http, AuthenticationManager authManager) throws Exception {

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

.authorizeHttpRequests(auth -> auth

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

.anyRequest().authenticated()

)

.httpBasic(Customizer.*withDefaults*())

.addFilter(new JwtAuthorizationFilter(authManager));

return http.build();

}

}

**Step 4:** Create /authenticate Endpoint (AuthenticationController.java)

package com.cognizant.jwt.controller;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

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 java.security.Key;

import java.util.Base64;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

@RestController

public class AuthenticationController {

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

private static final String *SECRET* = "mysecretkeymysecretkeymysecretkey1234";

private static final Key *key* = Keys.*hmacShaKeyFor*(*SECRET*.getBytes());

@GetMapping("/authenticate")

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

*LOGGER*.info("Start Authentication");

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

String user = extractUser(authHeader);

String token = generateJwt(user);

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

response.put("token", token);

return response;

}

private String extractUser(String authHeader) {

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

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

String credentials = new String(credDecoded);

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

}

private String generateJwt(String user) {

return Jwts.*builder*()

.setSubject(user)

.setIssuedAt(new Date())

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

.signWith(*key*, SignatureAlgorithm.*HS256*)

.compact();

}

}

**Step 5:** Create JWT Filter (JwtAuthorizationFilter.java)

package com.cognizant.jwt.security;

import java.io.IOException;

import java.util.ArrayList;

import jakarta.servlet.FilterChain;

import jakarta.servlet.ServletException;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

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;

import io.jsonwebtoken.Claims;

import io.jsonwebtoken.Jws;

import io.jsonwebtoken.JwtException;

import io.jsonwebtoken.Jwts;

public class JwtAuthorizationFilter extends BasicAuthenticationFilter {

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

private static final String *SECRET* = "mysecretkeymysecretkeymysecretkey1234"; // Same secret as in controller

public JwtAuthorizationFilter(AuthenticationManager authenticationManager) {

super(authenticationManager);

*LOGGER*.info("JwtAuthorizationFilter initialized");

}

@Override

protected void doFilterInternal(HttpServletRequest request,

HttpServletResponse response,

FilterChain chain)

throws IOException, ServletException {

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

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

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

chain.doFilter(request, response);

return;

}

UsernamePasswordAuthenticationToken authentication = getAuthentication(request);

SecurityContextHolder.*getContext*().setAuthentication(authentication);

chain.doFilter(request, response);

}

private UsernamePasswordAuthenticationToken getAuthentication(HttpServletRequest request) {

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

if (token != null) {

try {

Jws<Claims> parsedToken = Jwts.parser()

.setSigningKey(*SECRET*.getBytes())

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

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

if (user != null) {

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

}

} catch (JwtException e) {

*LOGGER*.error("Invalid JWT token", e);

}

}

return null;

}

}

**Step 6:** Create Secured Endpoint (CountryController.java)

package com.cognizant.jwt.controller;

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

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

import java.util.List;

@RestController

public class CountryController {

@GetMapping("/countries")

public List<String> getCountries() {

return List.*of*("India", "US", "Germany", "Japan");

}

}

**Step 7:** Test the application’s end point.

**Output:**



