**6 Create authentication service that returns JWT**

**JetAuthApplication.java**

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@SpringBootApplication*

public class JwtAuthApplication {

public static void main(String[] args) {

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

System.*out*.println("JWT Authentication Server started at http://localhost:8090");

}

}

**SecurityConfig.java**

package com.cognizant.spring\_learn.config;

import com.cognizant.spring\_learn.filter.JwtAuthorizationFilter;

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.builders.AuthenticationManagerBuilder;

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

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

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

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

import org.springframework.security.web.SecurityFilterChain;

*@Configuration*

*@EnableWebSecurity*

public class SecurityConfig {

*@Bean*

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

*@Bean*

public AuthenticationManager authManager(HttpSecurity http, PasswordEncoder passwordEncoder) throws Exception {

return http.getSharedObject(AuthenticationManagerBuilder.class)

.inMemoryAuthentication()

.withUser("admin").password(passwordEncoder.encode("pwd")).roles("ADMIN")

.and()

.withUser("user").password(passwordEncoder.encode("pwd")).roles("USER")

.and()

.passwordEncoder(passwordEncoder)

.and()

.build();

}

*@Bean*

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

http

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

.authorizeHttpRequests(auth -> auth

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

.anyRequest().authenticated()

)

.addFilter(new JwtAuthorizationFilter(authManager))

.httpBasic();

return http.build();

}

}

**JwtUtil.java**

package com.cognizant.spring\_learn.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

import java.security.Key;

import java.util.Date;

public class JwtUtil {

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

private static final long ***EXPIRATION\_TIME*** = 1000 \* 60 \* 60;

public static String generateToken(String username) {

return Jwts.*builder*()

.setSubject(username)

.setIssuedAt(new Date(System.*currentTimeMillis*()))

.setExpiration(new Date(System.*currentTimeMillis*() + ***EXPIRATION\_TIME***))

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

.compact();

}

}

**JwtFilter.java**

package com.cognizant.spring\_learn.security;

import jakarta.servlet.FilterChain;

import jakarta.servlet.ServletException;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import io.jsonwebtoken.Jws;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.Claims;

import io.jsonwebtoken.JwtException;

import io.jsonwebtoken.security.Keys;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

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

import org.springframework.web.filter.OncePerRequestFilter;

import javax.crypto.SecretKey;

import java.io.IOException;

import java.util.Collections;

public class JwtFilter extends OncePerRequestFilter {

private static final SecretKey ***key*** = Keys.*hmacShaKeyFor*("secretsecretsecretsecretsecret123456".getBytes());

*@Override*

protected void doFilterInternal(HttpServletRequest request,

HttpServletResponse response,

FilterChain filterChain) throws ServletException, IOException {

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

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

filterChain.doFilter(request, response);

return;

}

String token = header.substring(7); // Remove "Bearer "

try {

Jws<Claims> claimsJws = Jwts.*parserBuilder*()

.setSigningKey(***key***)

.build()

.parseClaimsJws(token);

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

UsernamePasswordAuthenticationToken auth = new UsernamePasswordAuthenticationToken(

user, null, Collections.*emptyList*());

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

} catch (JwtException e) {

response.setStatus(HttpServletResponse.***SC\_UNAUTHORIZED***);

return;

}

filterChain.doFilter(request, response);

}

}

**AuthController.java**

package com.cognizant.spring\_learn.controller;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

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

import java.util.Base64;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

import jakarta.servlet.http.HttpServletRequest;

import javax.crypto.SecretKey;

*@RestController*

public class AuthController {

private static final SecretKey ***key*** = Keys.*hmacShaKeyFor*("secretsecretsecretsecretsecret123456".getBytes());

private static final long ***EXPIRATION\_TIME*** = 1000 \* 60 \* 60;

*@GetMapping*("/authenticate")

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

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

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

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

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

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

if (values.length == 2 && values[0].equals("user") && values[1].equals("pwd")) {

String token = Jwts.*builder*()

.setSubject("user")

.setIssuedAt(new Date())

.setExpiration(new Date(System.*currentTimeMillis*() + ***EXPIRATION\_TIME***))

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

.compact();

response.put("token", token);

return response;

}

}

response.put("token", "invalid");

return response;

}

*@GetMapping*("/hello")

public String hello() {

return "Hello, secured world!";

}

}

**JwtAuthorizationFilter.java**

package com.cognizant.spring\_learn.filter;

import jakarta.servlet.FilterChain;

import jakarta.servlet.ServletException;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

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.\*;

import io.jsonwebtoken.security.Keys;

import javax.crypto.SecretKey;

import java.io.IOException;

import java.util.Collections;

public class JwtAuthorizationFilter extends BasicAuthenticationFilter {

private static final SecretKey ***key*** = Keys.*hmacShaKeyFor*("secretsecretsecretsecretsecret123456".getBytes());

public JwtAuthorizationFilter(AuthenticationManager authenticationManager) {

super(authenticationManager);

}

*@Override*

protected void doFilterInternal(HttpServletRequest request,

HttpServletResponse response,

FilterChain chain)

throws IOException, ServletException {

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

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

chain.doFilter(request, response);

return;

}

UsernamePasswordAuthenticationToken authentication = getAuthentication(header);

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

chain.doFilter(request, response);

}

private UsernamePasswordAuthenticationToken getAuthentication(String tokenHeader) {

String token = tokenHeader.replace("Bearer ", "");

try {

String user = Jwts.*parserBuilder*()

.setSigningKey(***key***)

.build()

.parseClaimsJws(token)

.getBody()

.getSubject();

if (user != null) {

return new UsernamePasswordAuthenticationToken(user, null, Collections.*emptyList*());

}

} catch (JwtException e) {

return null;

}

return null;

}

}

**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 http://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>3.2.4</version>

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

</parent>

<groupId>com.cognizant.spring\_learn</groupId>

<artifactId>jwt-auth</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>jwt-auth</name>

<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-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

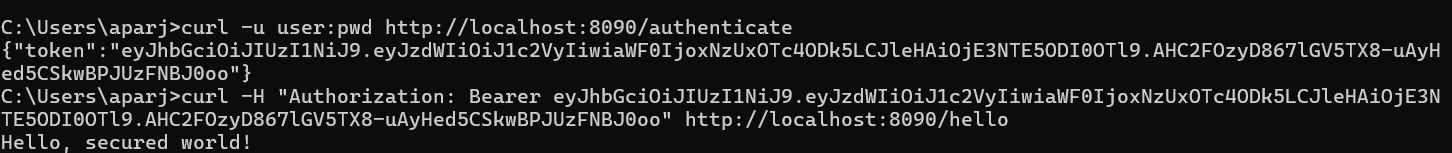
</plugin>

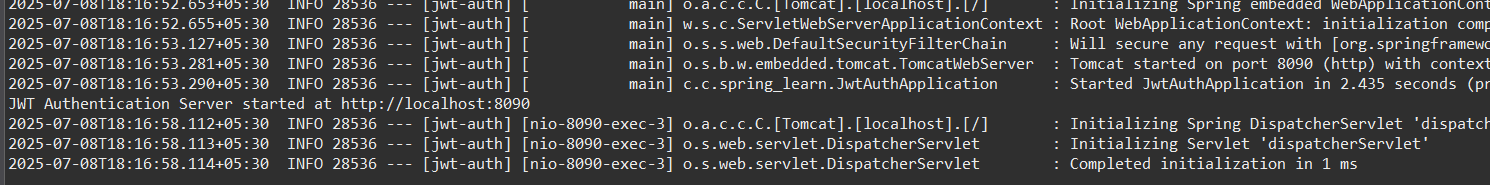
</plugins>

</build>

</project>

**OUTPUT:**

****

****