6. Create authentication service that returns JWT

**Request**

curl -s -u user:pwd <http://localhost:8090/authenticate>

**Response**

{"token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNTcwMzc5NDc0LCJle HAiOjE1NzAzODA2NzR9.t3LRvlCV-hwKfoqZYlaVQqEUiBloWcWn0ft3tgv0dL0"}

**Request**

curl -s -u user:pwd <http://localhost:8090/authenticate>

**Response**

{"token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNTcwMzc5NDc 0LCJleHAiOjE1NzAzODA2NzR9.t3LRvlCV-

hwKfoqZYlaVQqEUiBloWcWn0ft3tgv0dL0"}

This can be incorporated as three major steps:

* Create authentication controller and configure it in SecurityConfig
* Read Authorization header and decode the username and password
* Generate token based on the user retrieved in the previous step

**Code snippets:**

**jwtDemoApplication.java:**

package com.example.jwtDemo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class JwtDemoApplication {

public static void main(String[] args) { SpringApplication.*run*(JwtDemoApplication.class, args);

}

}

**AuthenticationController.java:**

package com.example.jwtDemo.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.GetMapping; import org.springframework.web.bind.annotation.RequestHeader; import org.springframework.web.bind.annotation.RestController;

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);

@GetMapping("/authenticate")

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

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

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

Map<String, String> map = new HashMap<>(); String user = getUser(authHeader); *LOGGER*.debug("Extracted user: {}", user); String token = generateJwt(user);

map.put("token", token); *LOGGER*.info("End /authenticate"); return map;

}

private String getUser(String authHeader) {

String encoded = authHeader.substring("Basic ".length()); byte[] decoded = Base64.*getDecoder*().decode(encoded); String[] parts = new String(decoded).split(":");

return parts[0]; // return username

}

private String generateJwt(String user) { JwtBuilder builder = Jwts.*builder*(); builder.setSubject(user); builder.setIssuedAt(new Date());

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

1000));

builder.signWith(SignatureAlgorithm.*HS256*, "secretkey"); return builder.compact();

}

}

**CountryController.java:**

package com.example.jwtDemo.Controller;

import org.springframework.web.bind.annotation.\*; import java.util.List;

@RestController

public class CountryController {

@GetMapping("/countries")

public List<String> getCountries() {

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

}

}

**SecurityConfig.java:**

package com.example.jwtDemo.Security;

import com.example.jwtDemo.Security.JwtAuthorizationFilter; import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration; import org.springframework.security.authentication.\*;

import org.springframework.security.authentication.dao.DaoAuthenticationProvider; import org.springframework.security.config.Customizer;

import org.springframework.security.config.annotation.web.builders.HttpSecurity; import org.springframework.security.core.userdetails.\*;

import org.springframework.security.crypto.password.\*;

import org.springframework.security.provisioning.InMemoryUserDetailsManager; import org.springframework.security.web.SecurityFilterChain;

import org.springframework.security.web.authentication.UsernamePasswordAuthenticatio nFilter;

import java.util.List;

@Configuration

public class SecurityConfig {

@Bean

public UserDetailsService userDetailsService() { UserDetails user = User.*withUsername*("user")

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

.roles("USER")

.build();

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

.password(passwordEncoder().encode("admin"))

.roles("ADMIN")

.build();

return new InMemoryUserDetailsManager(user, admin);

}

@Bean

public AuthenticationManager authenticationManager(UserDetailsService userDetailsService) {

DaoAuthenticationProvider provider = new DaoAuthenticationProvider(); provider.setUserDetailsService(userDetailsService); provider.setPasswordEncoder(passwordEncoder());

return new ProviderManager(List.*of*(provider));

}

@Bean

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

return http

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

.httpBasic(Customizer.*withDefaults*())

.authorizeHttpRequests(auth -> auth

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

"ADMIN")

.anyRequest().authenticated()

)

.addFilterBefore(new JwtAuthorizationFilter(authManager), UsernamePasswordAuthenticationFilter.class)

.build();

}

@Bean

public PasswordEncoder passwordEncoder() {

return NoOpPasswordEncoder.*getInstance*(); // For development only

}

}

**JwtAuthorizationFilter.java:**

package com.example.jwtDemo.Security;

import io.jsonwebtoken.\*; import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.security.authentication.\*;

import org.springframework.security.core.context.SecurityContextHolder; import org.springframework.security.web.authentication.www.BasicAuthenticationFilter

;

import jakarta.servlet.\*; import jakarta.servlet.http.\*; import java.io.IOException; import java.util.\*;

public class JwtAuthorizationFilter extends BasicAuthenticationFilter { private static final Logger *LOGGER* =

LoggerFactory.*getLogger*(JwtAuthorizationFilter.class);

public JwtAuthorizationFilter(AuthenticationManager authManager) { super(authManager);

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

}

@Override

protected void doFilterInternal(HttpServletRequest req, HttpServletResponse res,

FilterChain chain) throws IOException,

ServletException {

*LOGGER*.info("Filtering...");

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

if (header == null || !header.startsWith("Bearer ")) { chain.doFilter(req, res);

return;

}

UsernamePasswordAuthenticationToken auth = getAuthentication(req); SecurityContextHolder.*getContext*().setAuthentication(auth); chain.doFilter(req, res);

}

private UsernamePasswordAuthenticationToken getAuthentication(HttpServletRequest req) {

String token = req.getHeader("Authorization"); if (token != null) {

try {

Jws<Claims> parsedToken = Jwts.*parser*()

.setSigningKey("secretkey")

.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", e);

}

}

return null;

}

}

**Pom.xml:**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns[="http://maven.apache.org/POM/4.0.0"](http://maven.apache.org/POM/4.0.0) xmlns:xsi[="http://www.w3.org/2001/XMLSchema-instance"](http://www.w3.org/2001/XMLSchema-instance)

xsi:schemaLocation[="http://maven.apache.org/POM/4.0.0](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.example</groupId>

<artifactId>jwtDemo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>jwtDemo</name>

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

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

<java.version>21</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>javax.xml.bind</groupId>

<artifactId>jaxb-api</artifactId>

<version>2.3.1</version>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.0</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

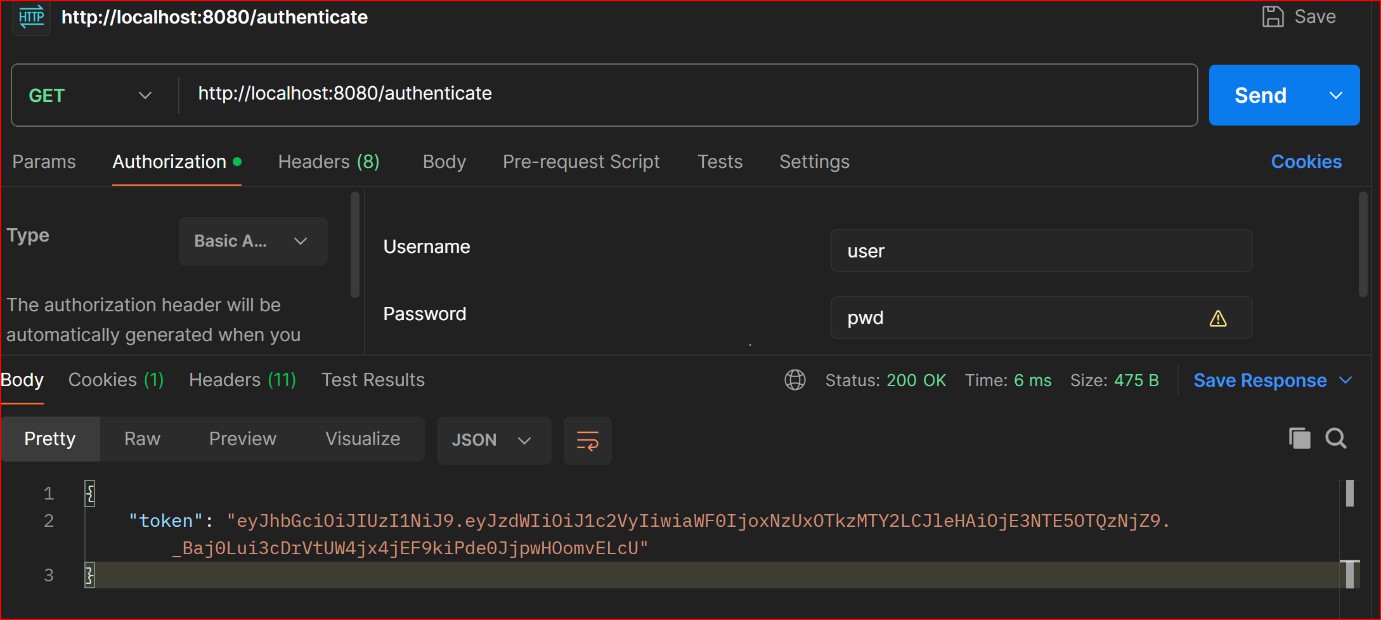
</plugin>

</plugins>

</build>

</project>

**Output:**

****

