**Create authentication controller and configure it in SecurityConfig** 

Directory:

jwt-auth-spring/

├── pom.xml

├── src/

│ └── main/

│ ├── java/

│ │ └── com/

│ │ └── example/

│ │ └── jwtauth/

│ │ ├── JwtAuthApplication.java

│ │ ├── config/

│ │ │ └── SecurityConfig.java

│ │ └── controller/

│ │ └── AuthenticationController.java

│ │ └── security/

│ │ └── SecurtiyConfig.java

│ │ └── util/

│ │ └── JwtUtil.java

│ └── resources/

│ └── application.properties

**Code for JwtAuthApplication.java:**

package com.example.jwtauth;

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);

    }

}

**Code for SecurityConfig.java:**

package com.example.jwtauth.config;

import org.springframework.context.annotation.\*;

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

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

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

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

import org.springframework.security.web.SecurityFilterChain;

@Configuration

public class SecurityConfig {

    @Bean

    public AuthenticationManager authManager(HttpSecurity http) throws Exception {

        return http.getSharedObject(AuthenticationManagerBuilder.class)

            .inMemoryAuthentication()

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

            .and()

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

            .and()

            .and().build();

    }

    @Bean

    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

        http.csrf().disable()

            .authorizeHttpRequests()

            .requestMatchers("/countries").hasRole("USER")

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

            .anyRequest().authenticated()

            .and().httpBasic();

        return http.build();

    }

    @Bean

    public static NoOpPasswordEncoder passwordEncoder() {

        return (NoOpPasswordEncoder) NoOpPasswordEncoder.getInstance(); // For demo only

    }

}

**Code for AuthenticationController.java:**

package com.example.jwtauth.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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<>();

        map.put("token", "");

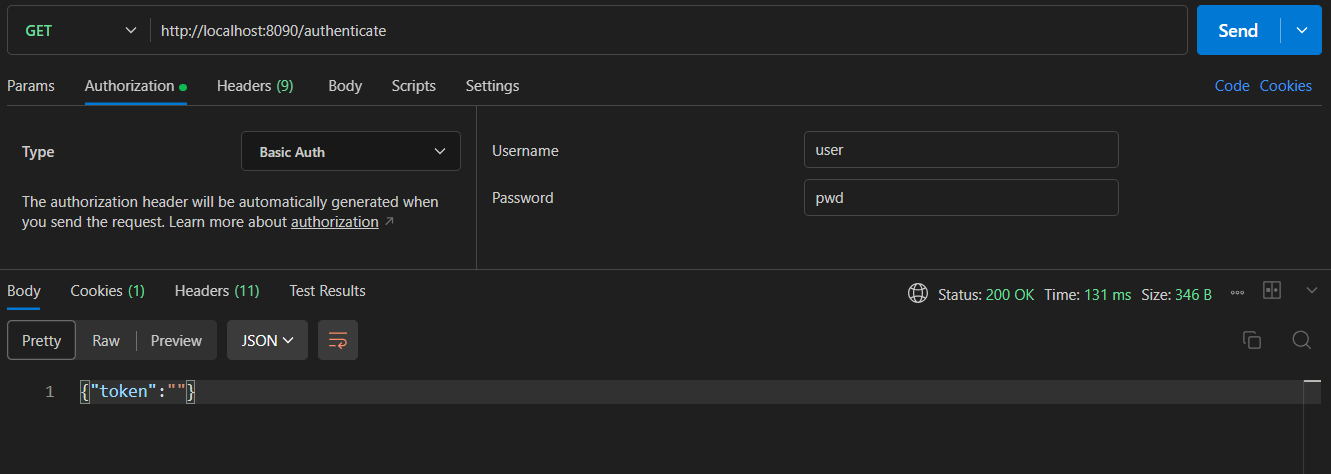
        LOGGER.info("END - /authenticate");

        return map;

    }

}

**Output:**



**Read Authorization header and decode the username and password**

**Code for JwtAuthApplication.java:**

package com.example.jwtauth;

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);

    }

}

**Code for SecurityConfig.java:**

package com.example.jwtauth.config;

import org.springframework.context.annotation.\*;

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

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

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

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

import org.springframework.security.web.SecurityFilterChain;

@Configuration

public class SecurityConfig {

    @Bean

    public AuthenticationManager authManager(HttpSecurity http) throws Exception {

        return http.getSharedObject(AuthenticationManagerBuilder.class)

            .inMemoryAuthentication()

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

            .and()

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

            .and()

            .and().build();

    }

    @Bean

    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

        http.csrf().disable()

            .authorizeHttpRequests()

            .requestMatchers("/countries").hasRole("USER")

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

            .anyRequest().authenticated()

            .and().httpBasic();

        return http.build();

    }

    @Bean

    public static NoOpPasswordEncoder passwordEncoder() {

        return (NoOpPasswordEncoder) NoOpPasswordEncoder.getInstance(); // For demo only

    }

}

**Code for AuthenticationController.java:**

package com.example.jwtauth.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import java.util.Base64;

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);

        String user = getUser(authHeader);

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

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

        map.put("token", "");

        LOGGER.info("END - /authenticate");

        return map;

    }

    private String getUser(String authHeader) {

        LOGGER.debug("Entered getUser() with authHeader: {}", authHeader);

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

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

            LOGGER.debug("Base64 Encoded Credentials: {}", encodedCredentials);

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

            String decoded = new String(decodedBytes);

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

            int colonIndex = decoded.indexOf(":");

            if (colonIndex != -1) {

                return decoded.substring(0, colonIndex);

            } else {

                LOGGER.warn("Invalid Basic Auth format: No colon found");

            }

        } else {

            LOGGER.warn("Invalid or missing Authorization header");

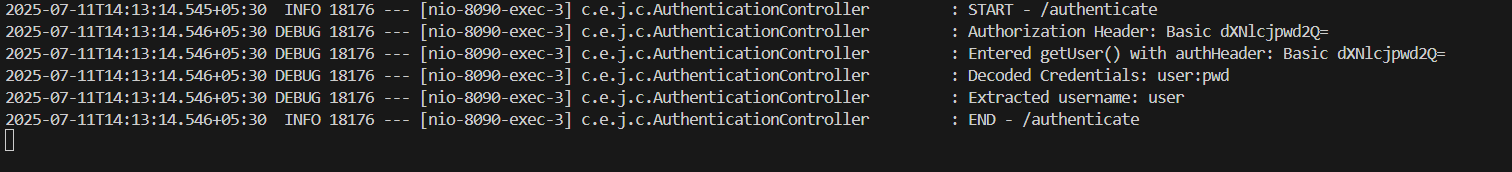
        }

        return "unknown";

    }

}

**Output:**

****

**Generate token based on the user**

**Code for JwtAuthApplication.java:**

package com.example.jwtauth;

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);

    }

}

**Code for SecurityConfig.java:**

package com.example.jwtauth.config;

import org.springframework.context.annotation.\*;

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

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

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

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

import org.springframework.security.web.SecurityFilterChain;

@Configuration

public class SecurityConfig {

    @Bean

    public AuthenticationManager authManager(HttpSecurity http) throws Exception {

        return http.getSharedObject(AuthenticationManagerBuilder.class)

            .inMemoryAuthentication()

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

            .and()

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

            .and()

            .and().build();

    }

    @Bean

    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

        http.csrf().disable()

            .authorizeHttpRequests()

            .requestMatchers("/countries").hasRole("USER")

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

            .anyRequest().authenticated()

            .and().httpBasic();

        return http.build();

    }

    @Bean

    public static NoOpPasswordEncoder passwordEncoder() {

        return (NoOpPasswordEncoder) NoOpPasswordEncoder.getInstance(); // For demo only

    }

}

**Code for AuthenticationController.java:**

package com.example.jwtauth.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.\*;

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 Key SECRET\_KEY = Keys.secretKeyFor(SignatureAlgorithm.HS256);

    @GetMapping("/authenticate")

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

        LOGGER.info("START - /authenticate");

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

        String username = getUser(authHeader);

        LOGGER.debug("Extracted username: {}", username);

        String token = generateToken(username);

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

        map.put("token", token);

        LOGGER.info("END - /authenticate");

        return map;

    }

    private String getUser(String authHeader) {

        LOGGER.debug("Entered getUser() with authHeader: {}", authHeader);

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

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

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

            String decoded = new String(decodedBytes);

            int colonIndex = decoded.indexOf(":");

            if (colonIndex != -1) {

                return decoded.substring(0, colonIndex);

            } else {

                LOGGER.warn("Invalid Basic Auth format: No colon found");

            }

        } else {

            LOGGER.warn("Invalid or missing Authorization header");

        }

        return "unknown";

    }

    private String generateToken(String username) {

        Date now = new Date();

        Date expiry = new Date(now.getTime() + 1000 \* 60 \* 60);

        return Jwts.builder()

                .setSubject(username)

                .setIssuedAt(now)

                .setExpiration(expiry)

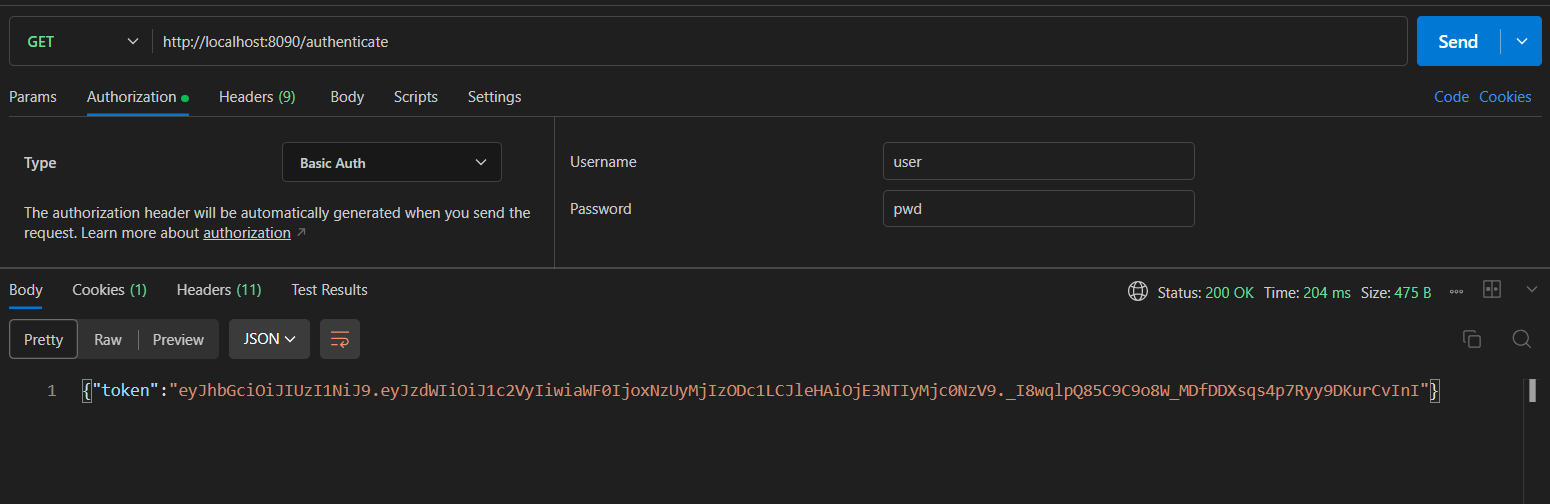
                .signWith(SECRET\_KEY)

                .compact();

    }

}

**Output:**

****

**Authorize based on JWT**

**Code for JwtAuthApplication.java:**

package com.example.jwtauth;

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);

    }

}

**Code for SecurityConfig.java:**

package com.example.jwtauth.config;

import com.example.jwtauth.security.JwtAuthorizationFilter;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

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

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

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

import org.springframework.security.web.SecurityFilterChain;

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

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

@Configuration

public class SecurityConfig {

    @Bean

    public AuthenticationManager authManager(HttpSecurity http) throws Exception {

        return http.getSharedObject(AuthenticationManagerBuilder.class)

            .inMemoryAuthentication()

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

            .and()

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

            .and()

            .and().build();

    }

    @Bean

    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

        http.csrf().disable()

            .authorizeHttpRequests()

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

            .anyRequest().authenticated()

            .and()

            .addFilterBefore(new JwtAuthorizationFilter(authManager(http)), BasicAuthenticationFilter.class);

        return http.build();

    }

    @Bean

    public static NoOpPasswordEncoder passwordEncoder() {

        return (NoOpPasswordEncoder) NoOpPasswordEncoder.getInstance();

    }

}

}

**Code for AuthenticationController.java:**

package com.example.jwtauth.controller;

import com.example.jwtauth.util.JwtUtil;

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

import org.springframework.security.core.Authentication;

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

@RestController

public class AuthenticationController {

    private final AuthenticationManager authenticationManager;

    public AuthenticationController(AuthenticationManager authenticationManager) {

        this.authenticationManager = authenticationManager;

    }

    @GetMapping("/authenticate")

    public String authenticate(@RequestParam String username, @RequestParam String password) {

        Authentication auth = authenticationManager.authenticate(

                new UsernamePasswordAuthenticationToken(username, password));

        return JwtUtil.generateToken(username);

    }

}

**Code for JwtAuthorizationFilter.java:**

package com.example.jwtauth.security;

import com.example.jwtauth.util.JwtUtil;

import io.jsonwebtoken.\*;

import org.slf4j.\*;

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.Collections;

public class JwtAuthorizationFilter extends BasicAuthenticationFilter {

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

    public JwtAuthorizationFilter(AuthenticationManager authenticationManager) {

        super(authenticationManager);

        LOGGER.info("JWT Filter Initialized");

    }

    @Override

    protected void doFilterInternal(HttpServletRequest req, HttpServletResponse res,

                                    FilterChain chain) throws IOException, ServletException {

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

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

            chain.doFilter(req, res);

            return;

        }

        UsernamePasswordAuthenticationToken authentication = getAuthentication(header);

        if (authentication != null) {

            SecurityContextHolder.getContext().setAuthentication(authentication);

        }

        chain.doFilter(req, res);

    }

    private UsernamePasswordAuthenticationToken getAuthentication(String header) {

        try {

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

            Jws<Claims> jws = Jwts.parserBuilder()

                    .setSigningKey(JwtUtil.SECRET\_KEY)

                    .build()

                    .parseClaimsJws(token);

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

            if (user != null) {

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

            }

        } catch (JwtException e) {

            LOGGER.warn("Invalid JWT: {}", e.getMessage());

        }

        return null;

    }

}

**Code for JwtUtil.java:**

package com.example.jwtauth.util;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

import java.security.Key;

import java.util.Date;

public class JwtUtil {

    public static final Key SECRET\_KEY = Keys.secretKeyFor(SignatureAlgorithm.HS256);

    public static String generateToken(String username) {

        return io.jsonwebtoken.Jwts.builder()

                .setSubject(username)

                .setIssuedAt(new Date())

                .setExpiration(new Date(System.currentTimeMillis() + 3600000))

                .signWith(SECRET\_KEY)

                .compact();

    }

}

**Output:**

