# Create authentication service that returns JWT

Program structure:



SecurityConfig.java

package com.cognizant.springlearn.config;

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.configuration.AuthenticationConfiguration;

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

import org.springframework.security.config.annotation.web.configurers.AbstractHttpConfigurer;

import org.springframework.security.config.http.SessionCreationPolicy;

import org.springframework.security.web.SecurityFilterChain;

import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

import com.cognizant.springlearn.filter.JwtAuthenticationFilter;

@Configuration

public class SecurityConfig {

    private final JwtAuthenticationFilter jwtAuthFilter;

    public SecurityConfig(JwtAuthenticationFilter jwtAuthFilter) {

        this.jwtAuthFilter = jwtAuthFilter;

    }

    @Bean

    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

        http.csrf(AbstractHttpConfigurer::disable)

            .authorizeHttpRequests(auth -> auth

                .requestMatchers("/", "/authenticate").permitAll()  // ⬅️ Permit access to root and authenticate

                .anyRequest().authenticated()                      // ⬅️ Everything else requires JWT

            )

            .sessionManagement(session -> session

                .sessionCreationPolicy(SessionCreationPolicy.STATELESS)

            )

            .addFilterBefore(jwtAuthFilter, UsernamePasswordAuthenticationFilter.class);

        return http.build();

    }

    @Bean

    public AuthenticationManager authenticationManager(AuthenticationConfiguration config) throws Exception {

        return config.getAuthenticationManager();

    }

}

AuthController.java

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.util.JwtUtil;

import org.springframework.beans.factory.annotation.Autowired;

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

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

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

import java.util.Base64;

import java.util.HashMap;

import java.util.Map;

@RestController

public class AuthController {

    @Autowired

    private JwtUtil jwtUtil;

    @RequestMapping("/authenticate")

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

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

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

        String credentials = new String(credDecoded);

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

        String username = values[0];

        String password = values[1];

        // Basic validation (for demo)

        if ("user".equals(username) && "pwd".equals(password)) {

            String token = jwtUtil.generateToken(username);

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

            tokenMap.put("token", token);

            return tokenMap;

        } else {

            throw new RuntimeException("Invalid credentials");

        }

    }

}

HelloController.java

package com.cognizant.springlearn.controller;

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

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

@RestController

public class HelloController {

    @GetMapping("/")

    public String hello() {

        return "Spring Boot is running!";

    }

}

JwtAuthenticationFilter.java

package com.cognizant.springlearn.filter;

import jakarta.servlet.FilterChain;

import jakarta.servlet.ServletException;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

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

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

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

import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;

import org.springframework.stereotype.Component;

import org.springframework.web.filter.OncePerRequestFilter;

import com.cognizant.springlearn.util.JwtUtil;

import java.io.IOException;

@Component

public class JwtAuthenticationFilter extends OncePerRequestFilter {

    private final JwtUtil jwtUtil;

    public JwtAuthenticationFilter(JwtUtil jwtUtil) {

        this.jwtUtil = jwtUtil;

    }

    @Override

    protected void doFilterInternal(HttpServletRequest request,

                                    HttpServletResponse response,

                                    FilterChain filterChain) throws ServletException, IOException {

        final String authHeader = request.getHeader("Authorization");

        String username = null;

        String jwt = null;

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

            jwt = authHeader.substring(7);

            username = jwtUtil.extractUsername(jwt);

        }

        if (username != null && SecurityContextHolder.getContext().getAuthentication() == null) {

            // For demo, we just create a simple user. In real apps, load user details from DB.

            User userDetails = new User(username, "", java.util.Collections.emptyList());

            if (jwtUtil.validateToken(jwt, userDetails)) {

                UsernamePasswordAuthenticationToken authToken =

                        new UsernamePasswordAuthenticationToken(userDetails, null, userDetails.getAuthorities());

                authToken.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));

                SecurityContextHolder.getContext().setAuthentication(authToken);

            }

        }

        filterChain.doFilter(request, response);

    }

}

User.java

package com.cognizant.springlearn.model;

public class User {

    private String username;

    private String password;

    public User() {}

    public User(String username, String password) {

        this.username = username;

        this.password = password;

    }

    public String getUsername() {

        return username;

    }

    public void setUsername(String username) {

        this.username = username;

    }

    public String getPassword() {

        return password;

    }

    public void setPassword(String password) {

        this.password = password;

    }

}

JwtUtil.java

package com.cognizant.springlearn.util;

import io.jsonwebtoken.Claims;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

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

import org.springframework.stereotype.Component;

import java.util.Date;

import java.util.function.Function;

@Component

public class JwtUtil {

    private final String secretKey = "mysecretkey"; // Use a secure key in production

    public String generateToken(String username) {

        return Jwts.builder()

                .setSubject(username)

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

                .setExpiration(new Date(System.currentTimeMillis() + 1000 \* 60 \* 60 \* 10)) // 10 hours

                .signWith(SignatureAlgorithm.HS256, secretKey)

                .compact();

    }

    public String extractUsername(String token) {

        return extractClaim(token, Claims::getSubject);

    }

    public Date extractExpiration(String token) {

        return extractClaim(token, Claims::getExpiration);

    }

    public <T> T extractClaim(String token, Function<Claims, T> claimsResolver) {

        final Claims claims = extractAllClaims(token);

        return claimsResolver.apply(claims);

    }

    private Claims extractAllClaims(String token) {

        return Jwts.parser()

                .setSigningKey(secretKey)

                .parseClaimsJws(token)

                .getBody();

    }

    private Boolean isTokenExpired(String token) {

        return extractExpiration(token).before(new Date());

    }

    public Boolean validateToken(String token, UserDetails userDetails) {

        final String username = extractUsername(token);

        return (username.equals(userDetails.getUsername()) && !isTokenExpired(token));

    }

}

SpringBootJwtAuthApplication.java

package com.cognizant.springlearn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringBootJwtAuthApplication {

    public static void main(String[] args) {

        SpringApplication.run(SpringBootJwtAuthApplication.class, args);

    }

}

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>

    <groupId>com.cognizant</groupId>

    <artifactId>spring-boot-jwt-auth</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <packaging>jar</packaging>

    <name>spring-boot-jwt-auth</name>

    <description>Spring Boot JWT Authentication Example</description>

    <parent>

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

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

        <version>3.2.5</version>

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

    </parent>

    <properties>

        <java.version>17</java.version>

        <jjwt.version>0.9.1</jjwt.version>

    </properties>

    <dependencies>

        <!-- Spring Boot Web -->

        <dependency>

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

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

        </dependency>

        <!-- Spring Boot Security -->

        <dependency>

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

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

        </dependency>

        <!-- JJWT (Java JWT) -->

        <dependency>

            <groupId>io.jsonwebtoken</groupId>

            <artifactId>jjwt</artifactId>

            <version>${jjwt.version}</version>

        </dependency>

        <!-- Jakarta Servlet API (for filters, if needed) -->

        <dependency>

            <groupId>jakarta.servlet</groupId>

            <artifactId>jakarta.servlet-api</artifactId>

            <scope>provided</scope>

        </dependency>

        <!-- Spring Boot Test (optional, for testing) -->

        <dependency>

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

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

            <scope>test</scope>

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <!-- Spring Boot Maven Plugin -->

            <plugin>

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

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

                <version>3.2.5</version>

            </plugin>

        </plugins>

    </build>

</project>

Output:





