**Exercise 2 : JWT Secure App**

**Project Structure:**

jwt-secure-app/

├── pom.xml

├── src/

│ └── main/

│ ├── java/

│ │ └── com/

│ │ └── example/

│ │ └── demo/

│ │ ├── DemoApplication.java # Main class

│ │ ├── JwtConfig.java # Reads secret from application.yml

│ │ ├── JwtTokenProvider.java # Creates & validates JWT

│ │ ├── JwtTokenFilter.java # Auth filter that validates token on each request

│ │ ├── SecurityConfig.java # Registers JwtTokenFilter in security chain

│ │ ├── SecureController.java # Protected /secure endpoint

│ │ └── TokenController.java # Public /token endpoint to generate JWT

│ └── resources/

│ └── application.yml # Holds JWT secret & server port

**DemoApplication.java**

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class DemoApplication {

    public static void main(String[] args) {

        SpringApplication.run(DemoApplication.class, args);

    }

}

**JwtConfig.java**

package com.example.demo;

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

import org.springframework.context.annotation.Configuration;

@Configuration

public class JwtConfig {

    @Value("${spring.security.jwt.secret}")

    private String secret;

    public String getSecret() {

        return secret;

    }

}

**JwtTokenFilter.java**

package com.example.demo;

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

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

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

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

import org.springframework.web.filter.OncePerRequestFilter;

import javax.servlet.FilterChain;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.IOException;

public class JwtTokenFilter extends OncePerRequestFilter {

    @Autowired

    private JwtTokenProvider jwtTokenProvider;

    @Override

    protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response,

                                    FilterChain filterChain) throws ServletException, IOException {

        String token = resolveToken(request);

        if (token != null && jwtTokenProvider.validateToken(token)) {

            String username = jwtTokenProvider.getUsername(token);

            UsernamePasswordAuthenticationToken auth =

                new UsernamePasswordAuthenticationToken(username, null, null);

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

            SecurityContextHolder.getContext().setAuthentication(auth);

        }

        filterChain.doFilter(request, response);

    }

    private String resolveToken(HttpServletRequest request) {

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

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

            return bearerToken.substring(7);

        }

        return null;

    }

}

**JwtTokenProvider.java**

package com.example.demo;

import io.jsonwebtoken.\*;

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

import org.springframework.stereotype.Component;

import java.util.Date;

@Component

public class JwtTokenProvider {

    @Autowired

    private JwtConfig jwtConfig;

    public String createToken(String username) {

        Claims claims = Jwts.claims().setSubject(username);

        Date now = new Date();

        Date validity = new Date(now.getTime() + 3600000); // 1 hour

        return Jwts.builder()

                .setClaims(claims)

                .setIssuedAt(now)

                .setExpiration(validity)

                .signWith(SignatureAlgorithm.HS256, jwtConfig.getSecret())

                .compact();

    }

    public boolean validateToken(String token) {

        try {

            Jwts.parser().setSigningKey(jwtConfig.getSecret()).parseClaimsJws(token);

            return true;

        } catch (JwtException | IllegalArgumentException e) {

            return false;

        }

    }

    public String getUsername(String token) {

        return Jwts.parser()

                .setSigningKey(jwtConfig.getSecret())

                .parseClaimsJws(token)

                .getBody()

                .getSubject();

    }

}

**SecureController.java**

package com.example.demo;

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

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

@RestController

public class SecureController {

    @GetMapping("/secure")

    public String secure() {

        return "This is a secure endpoint";

    }

}

**SecurityConfig.java**

package com.example.demo;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

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

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

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

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

@Configuration

@EnableWebSecurity

public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Bean

    public JwtTokenFilter jwtTokenFilter() {

        return new JwtTokenFilter();

    }

    @Override

    protected void configure(HttpSecurity http) throws Exception {

        http

            .authorizeRequests()

                .anyRequest().authenticated()

                .and()

            .addFilterBefore(jwtTokenFilter(), UsernamePasswordAuthenticationFilter.class);

    }

}

**TokenController.java**

package com.example.demo;

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

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

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

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

@RestController

public class TokenController {

    @Autowired

    private JwtTokenProvider jwtTokenProvider;

    @GetMapping("/token")

    public String getToken(@RequestParam String username) {

        return jwtTokenProvider.createToken(username);

    }

}

**Application.yml**

server:

  port: 8083  # You can change this if 8080 is busy

spring:

  security:

    jwt:

      secret: mySecretKey1234567890  # Secret key used to sign and validate JWT

**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>2.7.18</version>

<relativePath/>

</parent>

<groupId>com.example</groupId>

<artifactId>jwt-secure-app</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>jwt-secure-app</name>

<properties>

<java.version>11</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>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.1</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

<configuration>

<mainClass>com.example.demo.DemoApplication</mainClass>

</configuration>

</plugin>

</plugins>

</build>

</project>

**Output:**

