**SPRING LEARN JWT**

**PROJECT STRUCTURE**

spring-learn/

├── pom.xml

├── src/

│ └── main/

│ ├── java/

│ │ └── com/

│ │ └── cognizant/

│ │ └── springlearn/

│ │ ├── SpringLearnApplication.java ← MAIN class

│ │ ├── controller/

│ │ │ ├── AuthenticationController.java ← `/authenticate` endpoint

│ │ │ └── CountryController.java ← `/countries` endpoint

│ │ └── security/

│ │ ├── SecurityConfig.java ← Spring Security setup

│ │ └── JwtAuthorizationFilter.java ← JWT validation logic

│ └── resources/

│ ├── application.properties ← sets port = 8090

└── target/

**AuthenticationController.java**

package com.cognizant.springlearn.controller;

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;

import io.jsonwebtoken.JwtBuilder;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

@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);

        String token = generateJwt(user);

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

        map.put("token", token);

        LOGGER.info("End authenticate");

        return map;

    }

    private String getUser(String authHeader) {

        String encodedCredentials = authHeader.replace("Basic ", "");

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

        String decodedString = new String(decodedBytes);

        return decodedString.split(":")[0];

    }

    private String generateJwt(String user) {

        JwtBuilder builder = Jwts.builder();

        builder.setSubject(user);

        builder.setIssuedAt(new Date());

        builder.setExpiration(new Date((new Date()).getTime() + 1200000));

        builder.signWith(SignatureAlgorithm.HS256, "secretkey");

        return builder.compact();

    }

}

**JwtAuthorizationFilter.java**

package com.cognizant.springlearn.security;

import java.io.IOException;

import java.util.ArrayList;

import javax.servlet.FilterChain;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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.Claims;

import io.jsonwebtoken.Jws;

import io.jsonwebtoken.JwtException;

import io.jsonwebtoken.Jwts;

public class JwtAuthorizationFilter extends BasicAuthenticationFilter {

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

    public JwtAuthorizationFilter(AuthenticationManager authenticationManager) {

        super(authenticationManager);

    }

    @Override

    protected void doFilterInternal(HttpServletRequest req, HttpServletResponse res,

                                    FilterChain chain) throws IOException, ServletException {

        LOGGER.info("Start");

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

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

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

            chain.doFilter(req, res);

            return;

        }

        UsernamePasswordAuthenticationToken authentication = getAuthentication(req);

        SecurityContextHolder.getContext().setAuthentication(authentication);

        chain.doFilter(req, res);

        LOGGER.info("End");

    }

    private UsernamePasswordAuthenticationToken getAuthentication(HttpServletRequest request) {

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

        if (token != null) {

            try {

                Jws<Claims> jws = Jwts.parser()

                        .setSigningKey("secretkey")

                        .parseClaimsJws(token.replace("Bearer ", ""));

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

                if (user != null) {

                    return new UsernamePasswordAuthenticationToken(user, null, new ArrayList<>());

                }

            } catch (JwtException ex) {

                return null;

            }

        }

        return null;

    }

}

**SecurityConfig.java**

package com.cognizant.springlearn.security;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

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

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

@Configuration

@EnableWebSecurity

public class SecurityConfig extends WebSecurityConfigurerAdapter {

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

    @Override

    protected void configure(AuthenticationManagerBuilder auth) throws Exception {

        auth.inMemoryAuthentication()

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

            .and()

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

    }

    @Bean

    public PasswordEncoder passwordEncoder() {

        return new BCryptPasswordEncoder();

    }

    @Override

    protected void configure(HttpSecurity httpSecurity) throws Exception {

        httpSecurity.csrf().disable().httpBasic().and()

            .authorizeRequests()

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

            .anyRequest().authenticated()

            .and()

            .addFilter(new JwtAuthorizationFilter(authenticationManager()));

    }

}

**SpringLearnApplication.java**

package com.cognizant.springlearn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

    public static void main(String[] args) {

        SpringApplication.run(SpringLearnApplication.class, args);

    }

}

**Application.properties**

**server.port=8090**

**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-learn</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <packaging>jar</packaging>

    <name>spring-learn</name>

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

    <parent>

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

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

        <version>2.7.2</version>

        <relativePath/>

    </parent>

    <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</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:**





