**Spring Boot JWT Authentication**

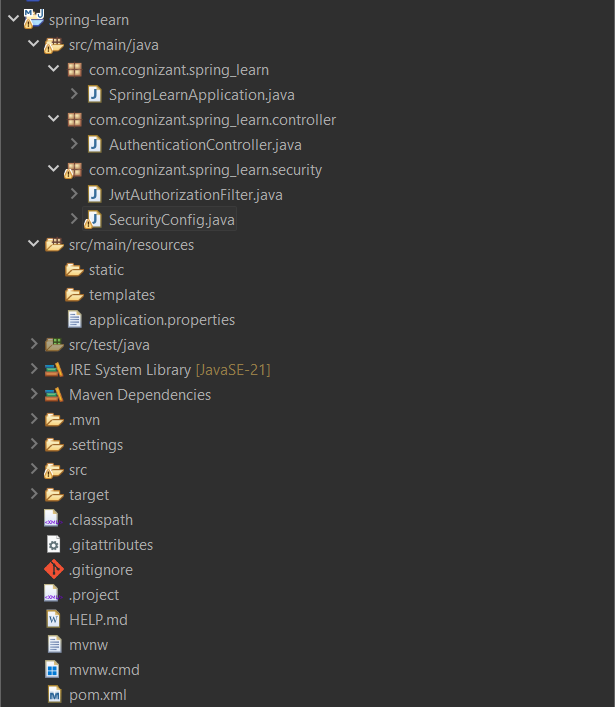
**Task Objective:**

Build an authentication service using Spring Boot + Spring Security + JWT that:

* Accepts username/password via Basic Auth
* Returns a signed JWT token
* Protects other REST endpoints using the JWT token

**📁 Project Setup**

**Project Name**: spring-learn



**⚡ Dependencies in pom.xml**

<dependencies>

<!-- Spring Boot Web -->

<dependency>

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

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

</dependency>

<!-- Spring Security -->

<dependency>

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

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

</dependency>

<!-- JWT -->

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.0</version>

</dependency>

<!-- Fix for Java 11+ (JAXB) -->

<dependency>

<groupId>javax.xml.bind</groupId>

<artifactId>jaxb-api</artifactId>

<version>2.3.1</version>

</dependency>

</dependencies>

**⚖️ Application Configuration**

**File**: application.properties

server.port=8080

**🚀 Spring Boot Entry Point**

**File**: SpringLearnApplication.java

package com.cognizant.spring\_learn;

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);

}

}

**🛡️ Security Configuration**

**File**: SecurityConfig.java

package com.cognizant.spring\_learn.security;

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.configuration.EnableWebSecurity;

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

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

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

import org.springframework.security.web.SecurityFilterChain;

@Configuration

@EnableWebSecurity

public class SecurityConfig {

@Bean

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

@Bean

public AuthenticationManager authManager(HttpSecurity http, PasswordEncoder encoder) throws Exception {

AuthenticationManagerBuilder auth = http.getSharedObject(AuthenticationManagerBuilder.class);

auth.inMemoryAuthentication()

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

.and()

.withUser("admin").password(encoder.encode("pwd")).roles("ADMIN");

return auth.build();

}

@Bean

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

http.csrf().disable()

.authorizeHttpRequests()

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

.anyRequest().authenticated()

.and()

.authenticationManager(authManager)

.addFilter(new JwtAuthorizationFilter(authManager));

return http.build();

}

}

**🔑 Authentication Controller**

**File**: AuthenticationController.java

package com.cognizant.spring\_learn.controller;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.JwtBuilder;

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

import java.util.Base64;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

@RestController

public class AuthenticationController {

@GetMapping("/authenticate")

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

String user = getUser(authHeader);

String token = generateJwt(user);

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

map.put("token", token);

return map;

}

private String getUser(String authHeader) {

String encoded = authHeader.substring(6);

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

return new String(decodedBytes).split(":" )[0];

}

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();

}

}

**🔒 JWT Filter**

**File**: JwtAuthorizationFilter.java

package com.cognizant.spring\_learn.security;

import io.jsonwebtoken.\*;

import jakarta.servlet.FilterChain;

import jakarta.servlet.ServletException;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

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 java.io.IOException;

import java.util.ArrayList;

public class JwtAuthorizationFilter extends BasicAuthenticationFilter {

public JwtAuthorizationFilter(AuthenticationManager authManager) {

super(authManager);

}

@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(req);

SecurityContextHolder.getContext().setAuthentication(authentication);

chain.doFilter(req, res);

}

private UsernamePasswordAuthenticationToken getAuthentication(HttpServletRequest request) {

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

if (token != null) {

try {

String user = Jwts.parser()

.setSigningKey("secretkey")

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

.getBody()

.getSubject();

if (user != null) {

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

}

} catch (Exception e) {

return null;

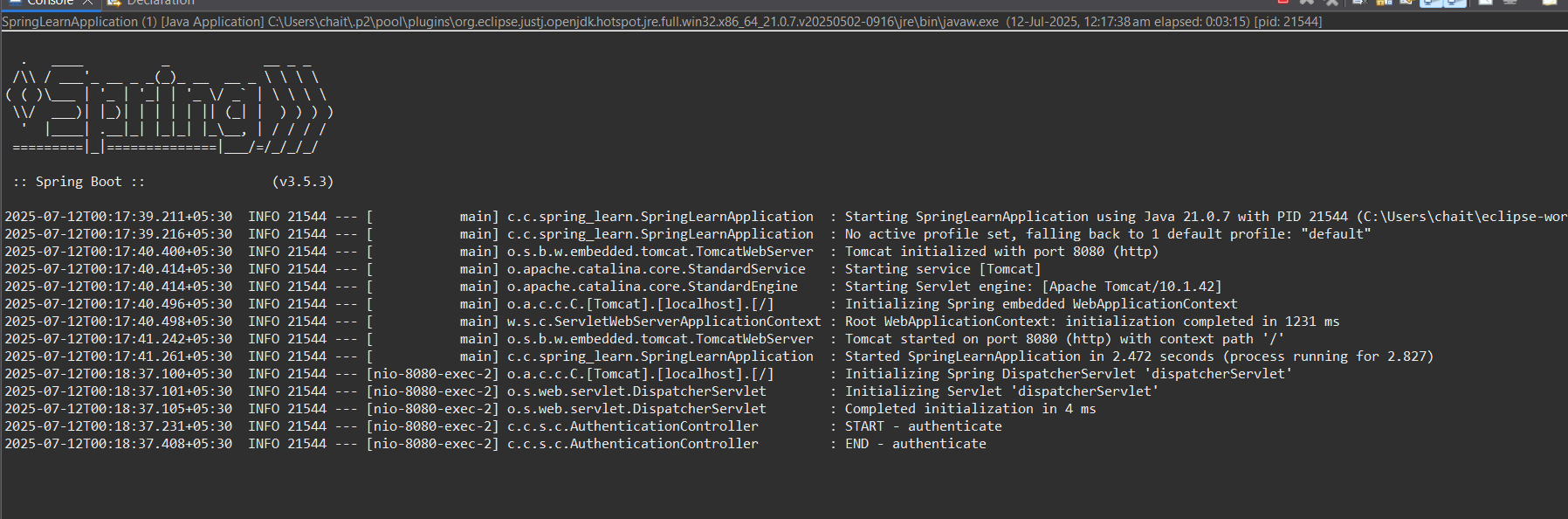
}

}

return null;

}

}

Console : 

**🔍 Testing /authenticate Endpoint (Postman)**

* Method: **GET**
* URL: http://localhost:8080/authenticate
* Authorization: Basic Auth
  + Username: user
  + Password: pwd

