1. **Securing RESTful Web Services with Spring Security1**

**pom.xml:**

<dependency>

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

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

</dependency>

**CountryController.java:**

**package** com.cognizant.spring\_learn.controller;

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

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

**import** java.util.List;

**import** com.cognizant.spring\_learn.model.Country;

@RestController

**public** **class** CountryController {

@GetMapping("/countries")

**public** List<Country> getCountries() {

**return** List.*of*(

**new** Country("IN", "India"),

**new** Country("US", "United States"),

**new** Country("JP", "Japan")

);

}

}

**SecurityConfig.java:**

package com.cognizant.spring\_learn.security;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

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

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

import org.springframework.security.web.SecurityFilterChain;

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

@Configuration

@EnableWebSecurity

public class SecurityConfig {

@Autowired

private CustomAuthEntryPoint customAuthEntryPoint;

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

.authorizeHttpRequests()

.anyRequest().authenticated()

.and()

.httpBasic()

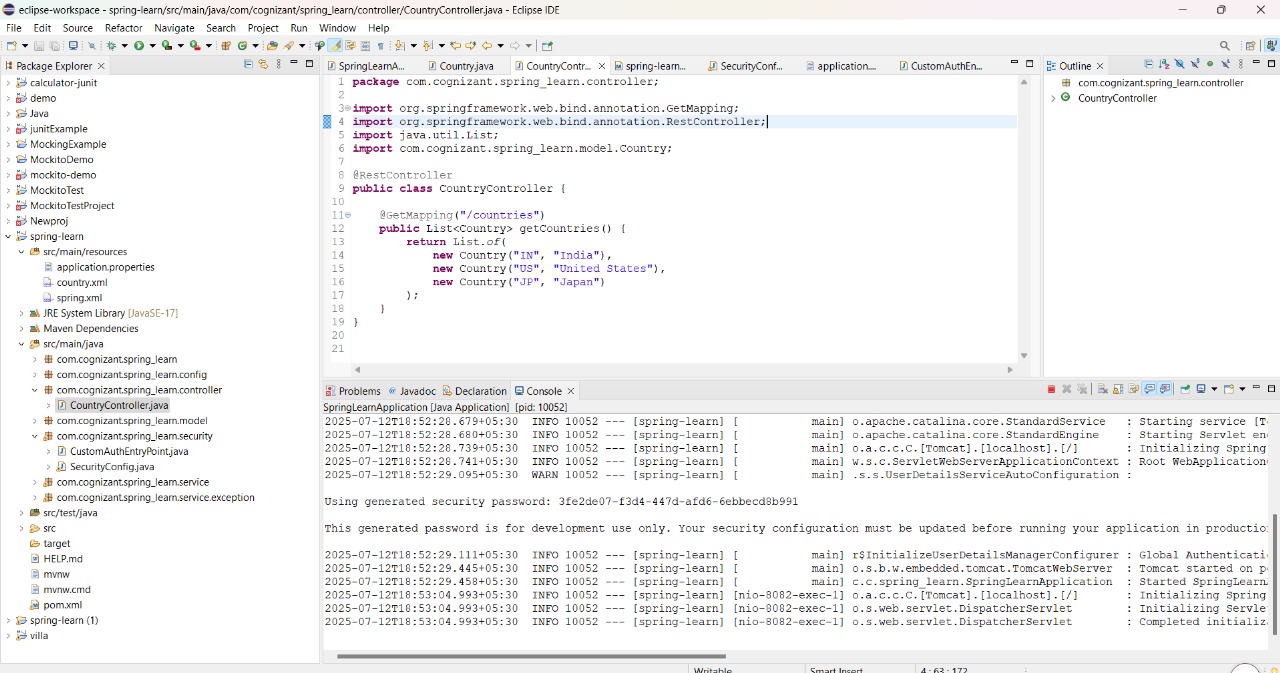
.authenticationEntryPoint(customAuthEntryPoint); // 👈 custom error handler

return http.build();

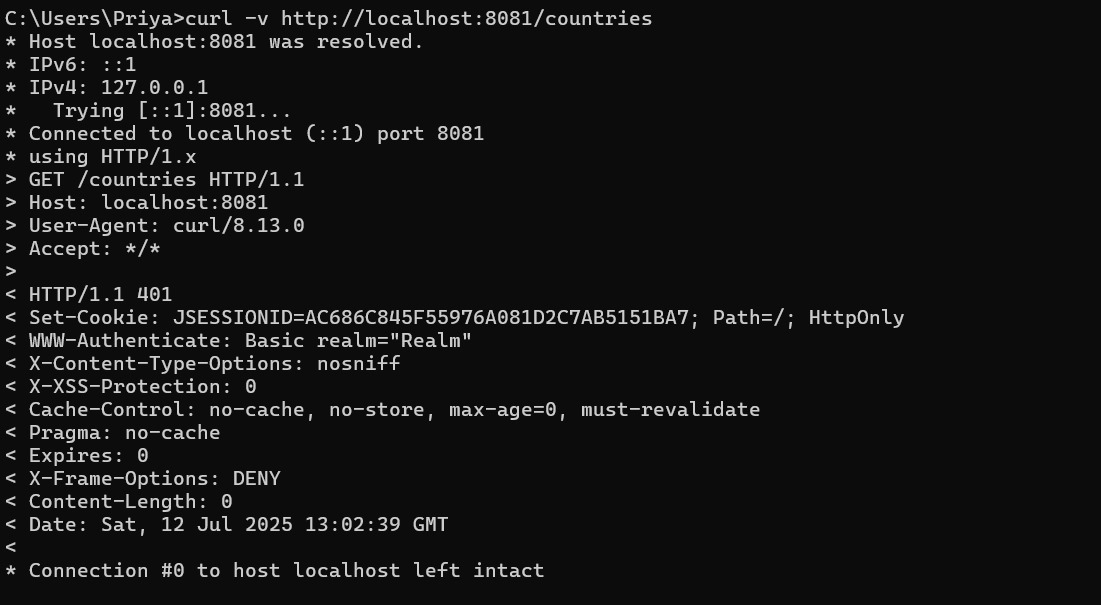
}

}

**OUTPUT:**

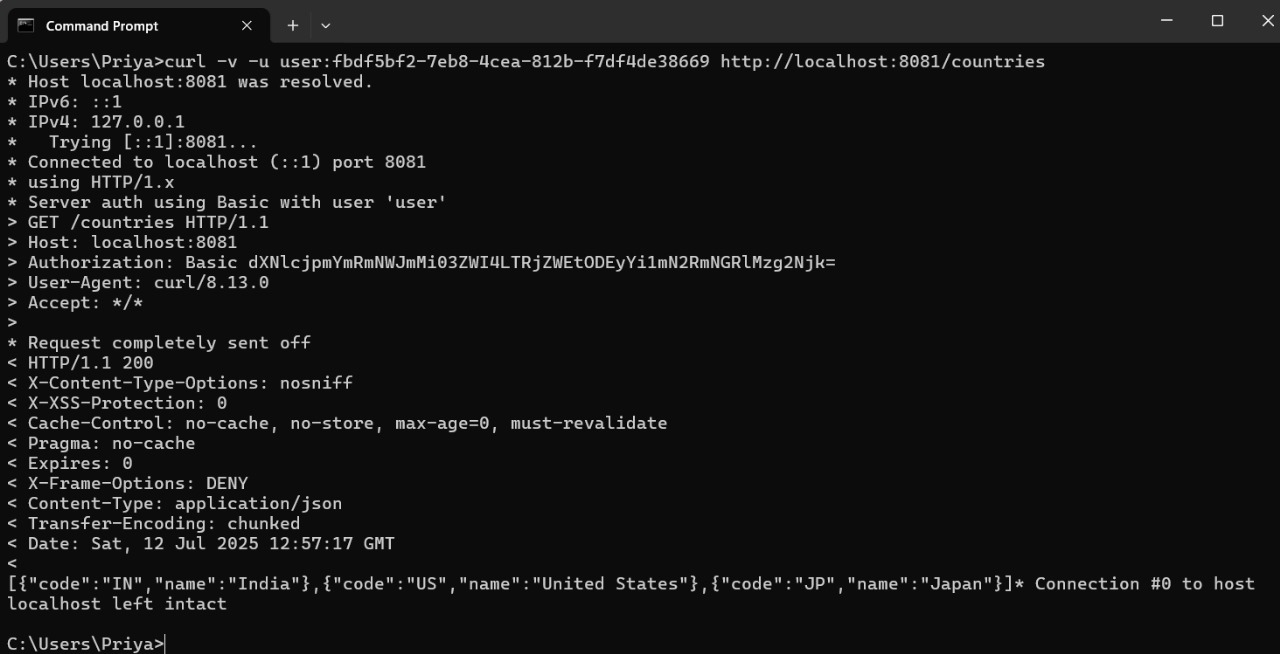


**Test the REST API Without Authentication:**

curl -v http://localhost:8081/countries 

**Test the API WITH Authentication:**

curl -v -u user:fbdf5bf2-7eb8-4cea-812b-f7df4de38669 http://localhost:8081/countries



**Creating users and roles in Spring Security**   
  
The earlier hands on demonstrated securing all URLs of the application with a common password. But it is not user and role specific.  
  
Let us create two new in memory users with names 'admin' and 'user'. The password for both the users will be 'pwd'.  
  
Let us define the rule that getting all countries can be accessed only 'user'.  
  
**SecurityConfig.java:**

package com.cognizant.spring\_learn.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.configuration.EnableWebSecurity;

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

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

import org.springframework.security.web.SecurityFilterChain;

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

@Configuration

@EnableWebSecurity

public class SecurityConfig {

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

@Bean

public PasswordEncoder passwordEncoder() {

LOGGER.info("PasswordEncoder Bean created");

return new BCryptPasswordEncoder();

}

@Bean

public AuthenticationManager authenticationManager(HttpSecurity http, PasswordEncoder passwordEncoder) throws Exception {

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

authBuilder.inMemoryAuthentication()

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

.and()

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

return authBuilder.build();

}

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

.csrf().disable()

.httpBasic()

.and()

.authorizeHttpRequests()

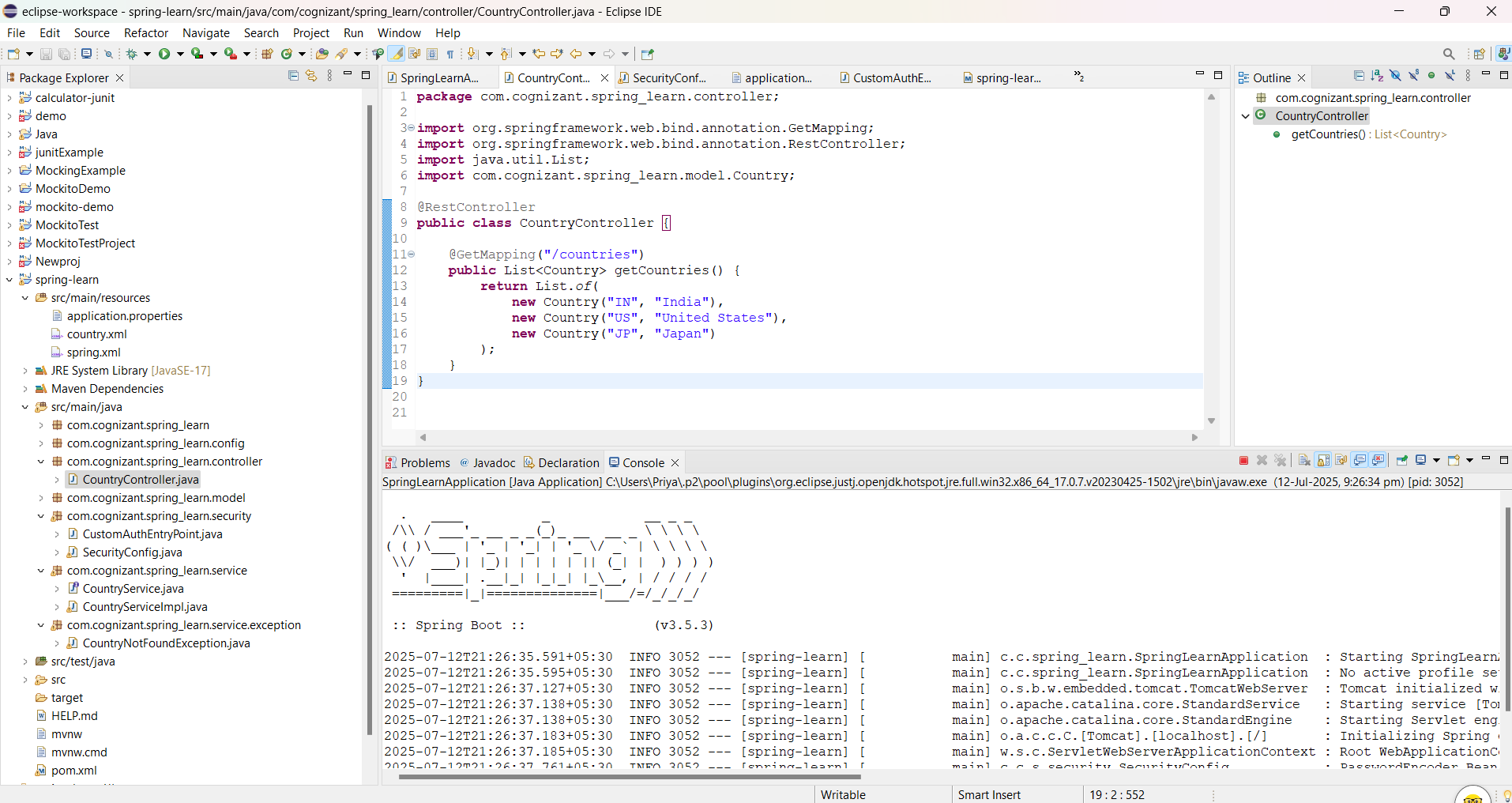
.antMatchers("/countries").hasRole("USER");

return http.build();

}

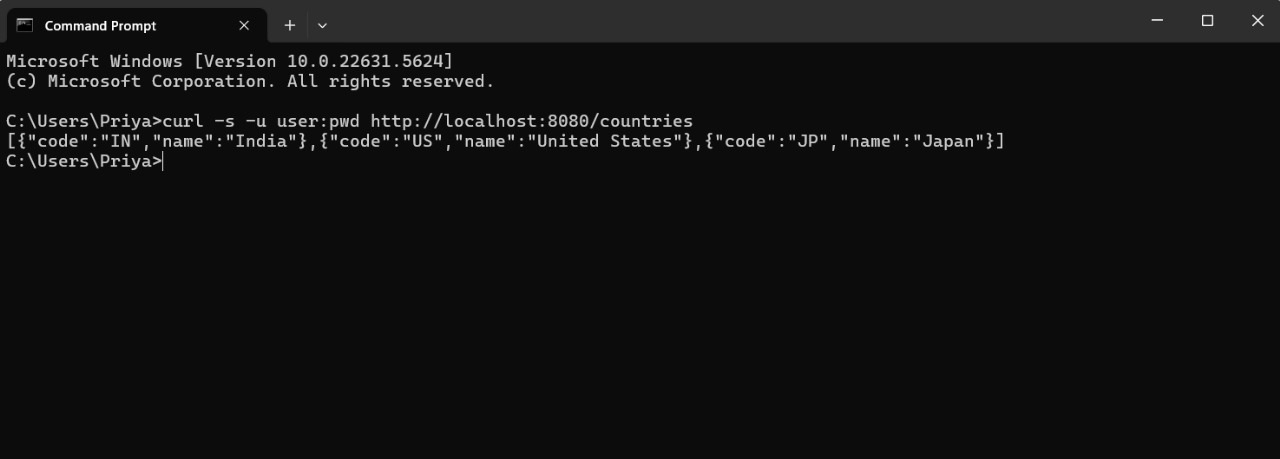
}

**OUTPUT:**



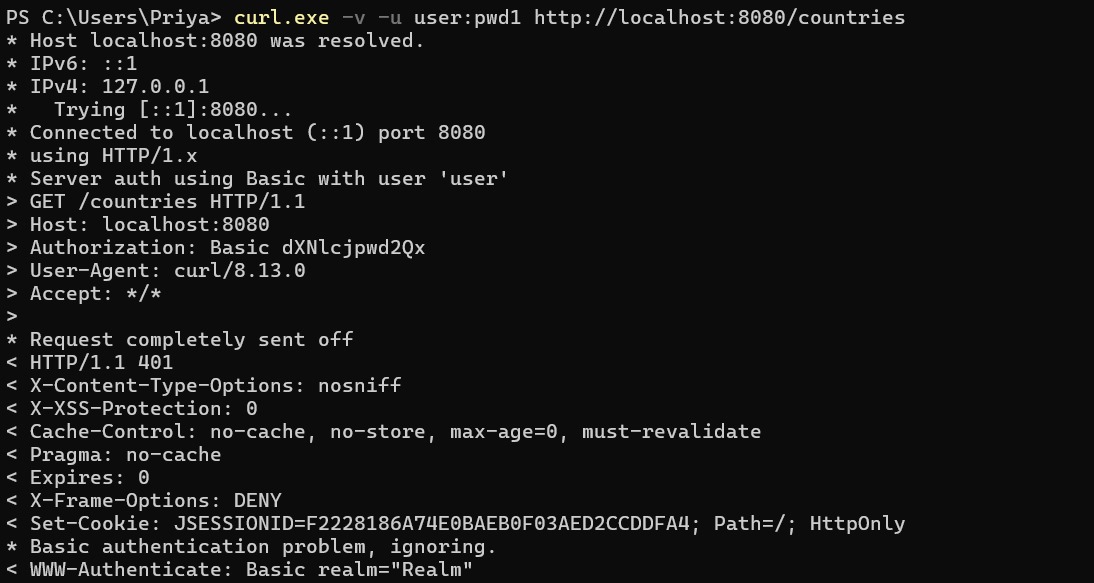
**Successful call with correct role:**

curl -s -u user:pwd <http://localhost:8080/countries>



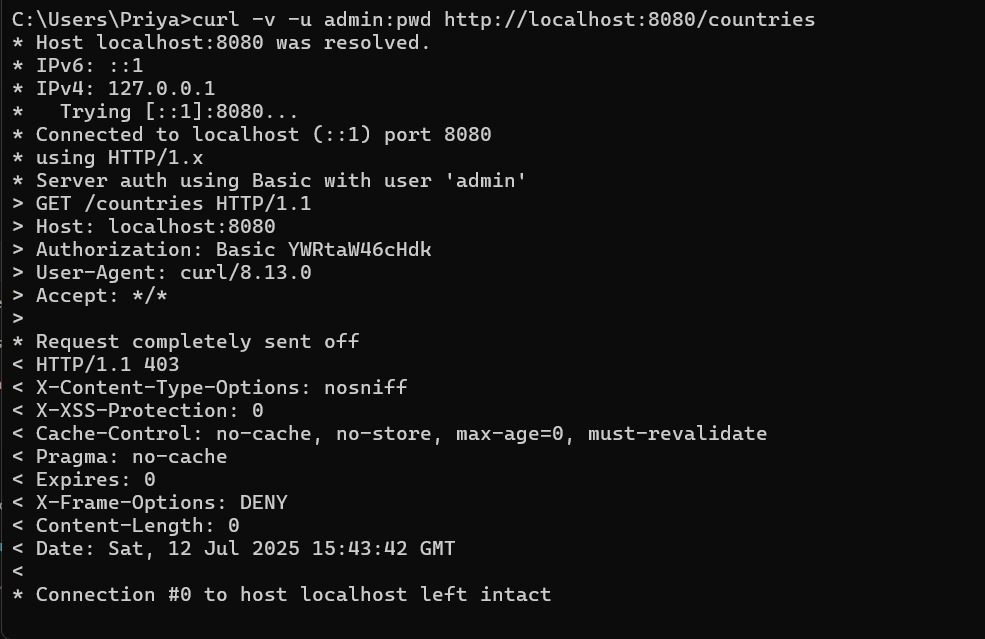
**Wrong password:**

curl.exe -v -u user:pwd1 <http://localhost:8080/countries>



Correct user but wrong role (admin has no access to /countries):

curl -v -u admin:pwd <http://localhost:8080/countries>



**3.Create authentication service that returns JWT** 

**Pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<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 https://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.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-learn</name>

<description>Spring Learn Project with JWT</description>

<properties>

<java.version>11</java.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>

<!-- Spring Boot DevTools -->

<dependency>

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

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<!-- JWT Dependencies -->

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-api</artifactId>

<version>0.11.5</version>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-impl</artifactId>

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-jackson</artifactId> <!-- or jjwt-gson if preferred -->

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

<!-- Spring TX -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-tx</artifactId>

</dependency>

<!-- JAXB (Optional, used only if needed by other libs, safe to keep) -->

<dependency>

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

<artifactId>jaxb-api</artifactId>

<version>2.3.1</version>

</dependency>

<dependency>

<groupId>org.glassfish.jaxb</groupId>

<artifactId>jaxb-runtime</artifactId>

<version>2.3.1</version>

</dependency>

<!-- Testing -->

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

**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);

}

}

**AuthenticationController.java**

package com.cognizant.spring\_learn.controller;

import java.util.Base64;

import javax.servlet.http.HttpServletRequest;

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

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

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

import com.cognizant.spring\_learn.util.JwtUtil;

*@RestController*

public class AuthenticationController {

*@Autowired*

private JwtUtil jwtUtil;

*@RequestMapping*("/authenticate")

public String generateToken(HttpServletRequest request) {

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

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

throw new RuntimeException("Missing or invalid Authorization header");

}

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

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

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

String username = credentials[0];

String password = credentials[1];

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

return "{\"token\":\"" + jwtUtil.generateToken(username) + "\"}";

} else {

throw new RuntimeException("Invalid credentials");

}

}

}

**SecurityConfig.java**

package com.cognizant.spring\_learn.security;

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

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.configuration.EnableWebSecurity;

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

import org.springframework.security.web.SecurityFilterChain;

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

import com.cognizant.spring\_learn.filter.JwtRequestFilter;

*@Configuration*

*@EnableWebSecurity*

public class SecurityConfig {

*@Autowired*

private JwtRequestFilter jwtRequestFilter;

*@Bean*

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http.csrf().disable()

.authorizeRequests()

.antMatchers("/authenticate").permitAll()

.anyRequest().authenticated()

.and()

.sessionManagement().sessionCreationPolicy(*SessionCreationPolicy*.*STATELESS*);

http.addFilterBefore(jwtRequestFilter, UsernamePasswordAuthenticationFilter.class);

return http.build();

}

*@Bean*

public AuthenticationManager authenticationManager(AuthenticationConfiguration configuration) throws Exception {

return configuration.getAuthenticationManager();

}

}

**JwtUtil.java**

package com.cognizant.spring\_learn.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

import java.security.Key;

import java.util.Date;

import org.springframework.stereotype.Component;

*@Component*

public class JwtUtil {

private static final Key ***key*** = Keys.*secretKeyFor*(*SignatureAlgorithm*.***HS256***);

private static final long ***EXPIRATION\_TIME*** = 10 \* 60 \* 1000;

public String generateToken(String username) {

return Jwts.*builder*()

.setSubject(username)

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

.setExpiration(new Date(System.*currentTimeMillis*() + ***EXPIRATION\_TIME***))

.signWith(***key***)

.compact();

}

public String extractUsername(String token) {

return Jwts.*parserBuilder*()

.setSigningKey(***key***)

.build()

.parseClaimsJws(token)

.getBody()

.getSubject();

}

public boolean validateToken(String token, String username) {

String extractedUsername = extractUsername(token);

return (username.equals(extractedUsername) && !isTokenExpired(token));

}

private boolean isTokenExpired(String token) {

Date expiration = Jwts.*parserBuilder*()

.setSigningKey(***key***)

.build()

.parseClaimsJws(token)

.getBody()

.getExpiration();

return expiration.before(new Date());

}

}

**JwtRequestFilter.java**

package com.cognizant.spring\_learn.filter;

import java.io.IOException;

import javax.servlet.FilterChain;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

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.stereotype.Component;

import org.springframework.web.filter.OncePerRequestFilter;

import com.cognizant.spring\_learn.util.JwtUtil;

*@Component*

public class JwtRequestFilter extends OncePerRequestFilter {

*@Autowired*

private JwtUtil jwtUtil;

*@Override*

protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain chain)

throws ServletException, IOException {

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

String username = null;

String jwt = null;

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

jwt = authorizationHeader.substring(7);

username = jwtUtil.extractUsername(jwt);

}

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

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

UsernamePasswordAuthenticationToken authToken =

new UsernamePasswordAuthenticationToken(username, null, new java.util.ArrayList<>());

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

SecurityContextHolder.*getContext*().setAuthentication(authToken);

}

}

chain.doFilter(request, response);

}

}

**OUTPUT:**

