**HandsOn : REST - Get country based on country code**

The objective of this HandsOn is to create an authentication service that receives username and password via HTTP Basic Auth, validates them, and returns a signed JWT token to the client.

**Steps :**

1. **Project Setup –**

I created a new Spring Boot project named –

* **Group ID** : com.cognizant
* **Artifact ID** : jwt-auth-service
* **Dependencies** : spring web, spring security, sprint boot web developer.

1. **Added dependencies in *pom.xml* .**

Following dependencies are added to support JWT creation, security, and REST API development –

<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>org.springframework.boot</groupId>

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

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-test</artifactId>

<scope>test</scope>

</dependency>

<!-- JWT -->

<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>

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

</dependencies>

1. **A class *SecurityConfig.java* is created to configure Spring Security .**

* package com.cognizant.jwt\_auth\_service.config;
* import org.springframework.context.annotation.Bean;
* import org.springframework.context.annotation.Configuration;
* import org.springframework.security.web.SecurityFilterChain;
* import org.springframework.security.config.annotation.web.builders.HttpSecurity;
* import org.springframework.security.core.userdetails.User;
* import org.springframework.security.core.userdetails.UserDetails;
* import org.springframework.security.provisioning.InMemoryUserDetailsManager;
* import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
* *@Configuration*
* *@EnableWebSecurity*
* public class SecurityConfig {
* *@Bean*
* public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
* http
* .~~csrf~~().disable()
* .~~authorizeHttpRequests~~()
* .requestMatchers("/authenticate").permitAll()
* .anyRequest().authenticated()
* .~~and~~()
* .~~httpBasic~~();
* return http.build();
* }
* *@Bean*
* public InMemoryUserDetailsManager userDetailsService() {
* UserDetails user = User
* .*withUsername*("user")
* .password("{noop}pwd")
* .roles("USER")
* .build();
* return new InMemoryUserDetailsManager(user);
* }
* }

1. **A class *Authorization Controller* is created.**

* package com.cognizant.jwt\_auth\_service.controller;
* import io.jsonwebtoken.Jwts;
* import io.jsonwebtoken.SignatureAlgorithm;
* import org.springframework.http.ResponseEntity;
* import org.springframework.web.bind.annotation.\*;
* import io.jsonwebtoken.security.Keys;
* import java.util.Date;
* import java.util.Base64;
* import java.util.HashMap;
* import java.util.Map;
* import jakarta.servlet.http.HttpServletRequest;
* *@RestController*
* public class AuthController {
* private static final String ***SECRET\_KEY*** = "f53b7c88f4e9caa8f53b7c88f4e9caa8";
* private static final long ***EXPIRATION\_TIME*** = 10 \* 60 \* 1000; // 10 minutes
* *@GetMapping*("/authenticate")
* public ResponseEntity<?> authenticate(HttpServletRequest request) {
* String header = request.getHeader("Authorization");
* if (header == null || !header.startsWith("Basic ")) {
* return ResponseEntity.*status*(401).body("Missing or invalid Authorization header.");
* }
* String base64Credentials = header.substring("Basic ".length());
* byte[] credDecoded = Base64.*getDecoder*().decode(base64Credentials);
* String credentials = new String(credDecoded);
* String[] values = credentials.split(":", 2);
* String username = values[0];
* String password = values[1];
* if (!"user".equals(username) || !"pwd".equals(password)) {
* return ResponseEntity.*status*(401).body("Invalid username or password.");
* }
* String token = Jwts.*builder*()
* .setSubject(username)
* .setIssuedAt(new Date())
* .setExpiration(new Date(System.*currentTimeMillis*() + ***EXPIRATION\_TIME***))
* .signWith(Keys.*hmacShaKeyFor*(***SECRET\_KEY***.getBytes()), *SignatureAlgorithm*.***HS256***)
* .compact();
* Map<String, String> response = new HashMap<>();
* response.put("token", token);
* return ResponseEntity.*ok*(response);
* }
* }

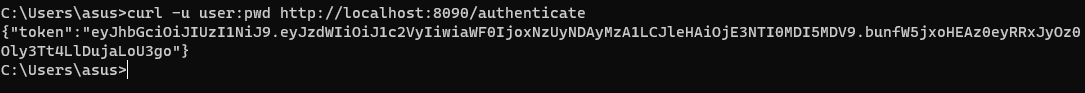
1. **Run the application *JwtAuthServiceApplication* as Java Application-**

We get console output as –

A screenshot of a computer

AI-generated content may be incorrect.

1. **Testing with cURL –**

****

1. **Output Token Structure –**

**A screenshot of a computer

AI-generated content may be incorrect.**