Create authentication service that returns JWT

As part of first step of JWT process, the user credentials needs to be sent to authentication service request that generates and returns the JWT.

Ideally when the below curl command is executed that calls the new authentication service, the token should be responded. Kindly note that the credentials are passed using -u option.

Request

curl -s -u user:pwd http://localhost:8090/authenticate

Response

{"token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNTcwMzc5NDc0LCJleHAiOjE1NzAzODA2NzR9.t3LRvlCV-hwKfoqZYlaVQqEUiBloWcWn0ft3tgv0dL0"}

This can be incorporated as three major steps:

- **Create authentication controller and configure it in SecurityConfig**
- **Read Authorization header and decode the username and password**
- **Generate token based on the user retrieved in the previous step**

Let incorporate the above as separate hands on exercises.

AuthenticationController.java

Create a controller to handle /authenticate:

```
@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 authentication");
    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 authentication");
    return map;
  }

  private String getUser(String authHeader) {
    LOGGER.info("Extracting user from header");
    String encodedCredentials = authHeader.substring("Basic ".length());
    byte[] decodedBytes = Base64.getDecoder().decode(encodedCredentials);
    String decodedString = new String(decodedBytes, StandardCharsets.UTF_8);
    LOGGER.debug("Decoded credentials: {}", decodedString);
    return decodedString.split(":")[0]; // returns "user"
  }

  private String generateJwt(String user) {
    LOGGER.info("Generating JWT");
    JwtBuilder builder = Jwts.builder()
```

```
            .setSubject(user)
            .setIssuedAt(new Date())
            .setExpiration(new Date(System.currentTimeMillis() + 20 * 60 * 1000)) // 20 mins
            .signWith(SignatureAlgorithm.HS256, "secretkey");

        return builder.compact();
    }
}
```

SecurityConfig.java

```
@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");
    }

    @Override
    protected void configure(HttpSecurity httpSecurity) throws Exception {
        httpSecurity.csrf().disable().httpBasic().and()
            .authorizeRequests()
        .antMatchers("/countries").hasRole("USER")

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

        .anyRequest().authenticated();

  }


  @Bean

  public PasswordEncoder passwordEncoder() {

    LOGGER.info("Creating password encoder");

    return new BCryptPasswordEncoder();

  }

}
```

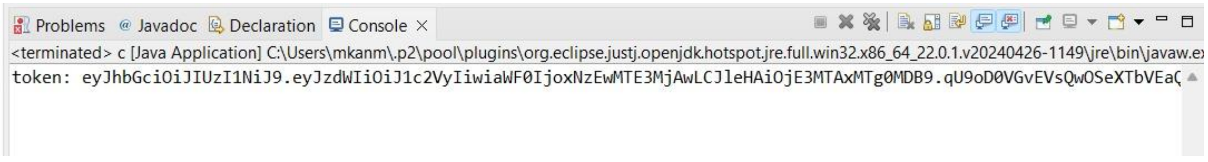
Add Dependency for JWT (in pom.xml)

```
<dependency>

  <groupId>io.jsonwebtoken</groupId>

  <artifactId>jjwt</artifactId>

  <version>0.9.0</version>

</dependency>
```

Generate Token:

```
curl -s -u user:pwd http://localhost:8090/authenticate
```

Output:



```
<terminated> c [Java Application] C:\Users\mkanm\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.1.v20240426-1149\jre\bin\javaw.ex
token: eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNzEwMTE3MjAwLCJleHAiOjE3MTAxMTg0MDB9.qU9oD0VGvEVsQwOSeXTbVEaC
```