**6) 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

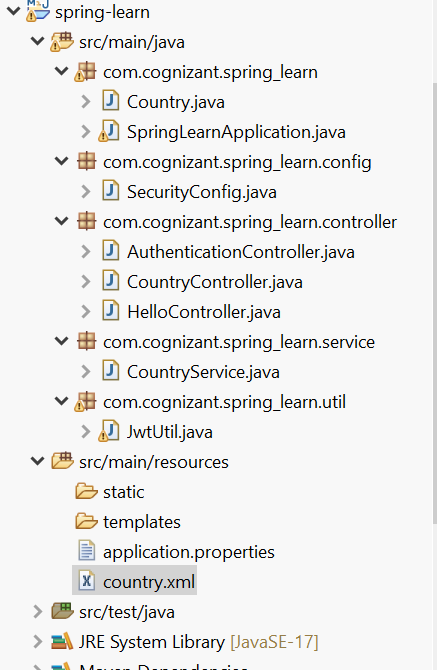
**Response**

{"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.

**Project Structure:  
**

**Code:**

**SecurityConfig.java:**

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

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

**import** org.springframework.security.config.annotation.method.configuration.EnableMethodSecurity;

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

**import** org.springframework.security.core.userdetails.\*;

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

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

**import** org.springframework.security.provisioning.InMemoryUserDetailsManager;

**import** org.springframework.security.web.SecurityFilterChain;

**import** org.springframework.security.config.Customizer;

@Configuration

@EnableMethodSecurity

**public** **class** SecurityConfig {

@Bean

**public** PasswordEncoder passwordEncoder() {

**return** **new** BCryptPasswordEncoder();

}

@Bean

**public** UserDetailsService userDetailsService(PasswordEncoder enc) {

UserDetails user = User.*withUsername*("user")

.password(enc.encode("pwd"))

.roles("USER")

.build();

**return** **new** InMemoryUserDetailsManager(user);

}

@Bean

**public** SecurityFilterChain filterChain(HttpSecurity http) **throws** Exception {

http.csrf(csrf -> csrf.disable())

.authorizeHttpRequests(auth -> auth

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

.anyRequest().authenticated())

.httpBasic(Customizer.*withDefaults*());

**return** http.build();

}

}

**JwtUtil.java:**

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

**import** java.util.Date;

**import** javax.crypto.SecretKey;

**import** org.springframework.beans.factory.annotation.Value;

**import** org.springframework.stereotype.Component;

**import** io.jsonwebtoken.Jwts;

**import** io.jsonwebtoken.~~SignatureAlgorithm~~;

**import** io.jsonwebtoken.security.Keys;

@Component

**public** **class** JwtUtil {

@Value("${jwt.secret}")

**private** String secret;

@Value("${jwt.expirationMs}")

**private** **long** expirationMs;

**public** String generateToken(String username) {

Date now = **new** Date();

Date expiry = **new** Date(now.getTime() + expirationMs);

SecretKey key = Keys.*hmacShaKeyFor*(secret.getBytes());

**return** Jwts.*builder*()

.setSubject(username)

.setIssuedAt(now)

.setExpiration(expiry)

.signWith(key, SignatureAlgorithm.~~HS256~~)

.compact();

}

}

**AuthenticationController.java:**

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

**import** java.security.Principal;

**import** java.util.Map;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** org.springframework.http.HttpStatus;

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

**import** org.springframework.web.server.ResponseStatusException;

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

@RestController

**public** **class** AuthenticationController {

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

**private** **final** JwtUtil jwtUtil;

**public** AuthenticationController(JwtUtil jwtUtil) {

**this**.jwtUtil = jwtUtil;

}

@GetMapping("/authenticate")

**public** Map<String, String> authenticate(Principal principal) {

***LOGGER***.debug("START authenticate()");

**if** (principal == **null**) {

**throw** **new** ResponseStatusException(HttpStatus.***UNAUTHORIZED***, "No credentials");

}

String token = jwtUtil.generateToken(principal.getName());

***LOGGER***.debug("Token generated for {}", principal.getName());

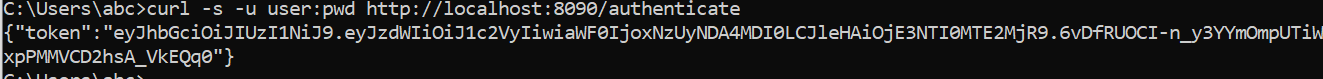
***LOGGER***.debug("END authenticate()");

**return** Map.*of*("token", token);

}

}

**Output:**

****