**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.

**jwtUtil.java**

package com.cognizant.spring\_learn.util;  
  
import io.jsonwebtoken.\*;  
import io.jsonwebtoken.security.Keys;  
import org.springframework.stereotype.Component;  
  
import java.security.Key;  
import java.util.Date;  
  
@Component  
public class JwtUtil {  
  
 private static final String *SECRET\_KEY* = "mysecretkeymysecretkeymysecretkeymysecretkey"; // 256-bit key  
 private static final long *EXPIRATION\_TIME* = 1000 \* 60 \* 10; // 10 mins  
  
 private final Key key = Keys.*hmacShaKeyFor*(*SECRET\_KEY*.getBytes());  
  
 public String generateToken(String username) {  
 return Jwts.*builder*()  
 .setSubject(username)  
 .setIssuedAt(new Date())  
 .setExpiration(new Date(System.*currentTimeMillis*() + *EXPIRATION\_TIME*))  
 .signWith(key, SignatureAlgorithm.*HS256*)  
 .compact();  
 }  
}

**SecurityConfig.java**

package com.cognizant.spring\_learn.config;  
  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.web.SecurityFilterChain;  
  
import org.springframework.context.annotation.Bean;  
  
@Configuration  
public class SecurityConfig {  
  
 @Bean  
 public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {  
 http.csrf(csrf -> csrf.disable())  
 .authorizeHttpRequests(auth -> auth.anyRequest().permitAll());  
 return http.build();  
 }  
}

**AuthController.java**

package com.cognizant.spring\_learn.controller;  
  
import com.cognizant.spring\_learn.util.JwtUtil;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.Base64;  
  
@RestController  
public class AuthController {  
  
 @Autowired  
 private JwtUtil jwtUtil;  
  
 @GetMapping("/authenticate")  
 public TokenResponse authenticate(@RequestHeader("Authorization") String authHeader) {  
 String[] credentials = extractCredentials(authHeader);  
 String username = credentials[0];  
 String password = credentials[1];  
  
 // Mocked auth logic  
 if ("user".equals(username) && "pwd".equals(password)) {  
 String token = jwtUtil.generateToken(username);  
 return new TokenResponse(token);  
 } else {  
 throw new RuntimeException("Invalid credentials");  
 }  
 }  
  
 private String[] extractCredentials(String authHeader) {  
 String base64Creds = authHeader.substring("Basic ".length());  
 byte[] decodedBytes = Base64.*getDecoder*().decode(base64Creds);  
 return new String(decodedBytes).split(":", 2);  
 }  
  
 public static class TokenResponse {  
 private String token;  
  
 public TokenResponse(String token) {  
 this.token = token;  
 }  
  
 public String getToken() {  
 return token;  
 }  
  
 public void setToken(String token) {  
 this.token = token;  
 }  
 }  
}

**Output:**

