Spring Boot –JWT Integration

Update the pom.xnl file with below dependency.

```
<dependency>
   <groupId>io.jsonwebtoken</groupId>
   <artifactId>jjwt</artifactId>
  <version>0.6.0  </version>
</dependency>
```

Steps involved in this approach

1. Create JWT Token for combining both username and password.
2.Include this generated token in each and every request headers.
3.In the target micro service once you reach the request headers data decode the jwt token.
4. Compare the Logged in user details against decoded JWT token values.
5. On successful validation of these credentials provide access to resource.

Step1: JWT token generation.

```
public static String generateToken
             (String signingKey, String username, String password) {
       long nowMillis = System.currentTimeMillis();
        JwtBuilder builder = Jwts.builder()
                 .setIssuedAt(new Date(nowMillis))
                 .setSubject(String.valueOf(username))
                        .setIssuer(signingKey)
                        .claim("password", password)
                        .signWith(SignatureAlgorithm.HS256, signingKey);
                        return builder.compact();

    }
```

Step 2: Once JWT Token generated include this token in headers.

```
this.authenticate = function(username, password, callback) {
             var user = {
                    "username" : username,
                    "password" : password };
var responsePromise = $http({
                    url : "http://localhost:8086/customerService/login",
                    method : "POST",
                    data : user,
                    headers : {
                           'Content-Type' : 'application/json',
                           'jwtToken' : jwttoken }
             });
responsePromise.success(function(data, status, headers, config) {
                    callback(data); });
responsePromise.error(function(data, status, headers, config) {
alert("AJAX failed! because no webservice is attached yet");
             });

       }
```

Step3: In the target Micro Service (rest Controller) decode the jwt token as mentioned below.

```
public static String getSubject(HttpServletResponse httpServletResponse,
String token, String signingKey) {
Jws<Claims> claims = Jwts.parser().setSigningKey(signingKey).parseClaimsJws(token);
             String username = claims.getBody().getSubject();
             LOGGER.info("Subject :::"+username);
             return username;
       }

public static String getPassword(HttpServletResponse httpServletResponse,String
token, String signingKey) {
Jws<Claims> claims = Jwts.parser().setSigningKey(signingKey).parseClaimsJws(token);
             Object password = claims.getBody().get("password");
             LOGGER.info("password :::"+password);
             return password.toString();
       }
```

Step 4 : Validate the decoded JWT tokenvalues on successful validate provide the access to resource.

```
  String jwtUsername = JwtUtil.getSubject(httpServletResponse, listObj, signingKey);
  String jwtPassword = JwtUtil.getPassword(httpServletResponse, listObj, signingKey);
       if (userCredentials != null)
             userCredjson = new JSONObject(userCredentials.toString());
             if (userCredjson != null)
               username = userCredjson.optString("username");
             if (username != null)
               customerObj = customerService.getCustomerDetails(username);
                    if (customerObj != null) {
       // Comapare the object from database and de copuled details from JWT Token
details ..
       if ((jwtPassword.equals(customerObj.getPassword())) &&
             jwtUsername.equals(customerObj.getUsername())) {
             validation = true;

             Step 5 : Provide the access to resource
             LOGGER.info("JWT Token Validataion Sucessful in customer Service.");
             } else {
             LOGGER.info("JWT Token Validataion failed in customer Service.");
             validation = false;
             Step 5 : Won't  Provide the access to resource     }
          }
```