## **JWT HANDSON**

**Create authentication service that returns JWT**

## **✅ Prerequisites**

* Spring Boot project with:
  + Spring Web
  + Spring Security
  + jjwt (JWT library)
* Port: 8090
* Java 8+

Add this dependency in pom.xml (for JWT):

<dependency>  
 <groupId>io.jsonwebtoken</groupId>  
 <artifactId>jjwt</artifactId>  
 <version>0.9.1</version>  
</dependency>

## **Hand-on 1: Create Authentication Controller**

### **Steps:**

1. **Create a controller class: AuthenticationController**

@RestController  
public class AuthenticationController {  
  
 private static final Logger LOGGER = LoggerFactory.getLogger(AuthenticationController.class);  
  
 @Autowired  
 private JwtUtil jwtUtil;  
  
 @RequestMapping("/authenticate")  
 public ResponseEntity<?> authenticate(@RequestHeader("Authorization") String authHeader) {  
 LOGGER.info("START authenticate");  
  
 // Logic will go in next hands-on  
 return ResponseEntity.ok(new JwtResponse("dummy-token"));  
  
 }  
}

1. **Create DTO JwtResponse**

public class JwtResponse {  
 private String token;  
  
 public JwtResponse(String token) {  
 this.token = token;  
 }  
  
 public String getToken() {  
 return token;  
 }  
}

## **Hand-on 2: Read and Decode Authorization Header**

### **Steps:**

1. In AuthenticationController, modify the /authenticate method:

@RequestMapping("/authenticate")  
public ResponseEntity<?> authenticate(@RequestHeader("Authorization") String authHeader) {  
 LOGGER.info("START authenticate");  
  
 // Header looks like: "Basic dXNlcjpwd2Q=" (user:pwd base64)  
 String base64Credentials = authHeader.substring("Basic".length()).trim();  
 byte[] decodedBytes = Base64.getDecoder().decode(base64Credentials);  
 String credentials = new String(decodedBytes);  
 String[] values = credentials.split(":", 2);  
 String username = values[0];  
 String password = values[1];  
  
 LOGGER.debug("Username: {}", username);  
 LOGGER.debug("Password: {}", password);  
  
 // In real scenarios, verify user from DB/service  
 if ("user".equals(username) && "pwd".equals(password)) {  
 String token = jwtUtil.generateToken(username);  
 return ResponseEntity.ok(new JwtResponse(token));  
 } else {  
 return ResponseEntity.status(HttpStatus.UNAUTHORIZED).build();  
 }  
}

## **Hand-on 3: Generate JWT Token**

### **Steps:**

1. **Create a Utility Class JwtUtil**

@Component  
public class JwtUtil {  
 private String secretKey = "secret";  
  
 public String generateToken(String username) {  
 return Jwts.builder()  
 .setSubject(username)  
 .setIssuedAt(new Date(System.currentTimeMillis()))  
 .setExpiration(new Date(System.currentTimeMillis() + 1000 \* 60 \* 60)) // 1 hour  
 .signWith(SignatureAlgorithm.HS256, secretKey)  
 .compact();  
 }  
}

1. ✅ Now, calling this cURL should return JWT:

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

**Expected Response:**

json

CopyEdit

{"token":"<JWT\_TOKEN\_STRING>"}

## **Security Configuration (SecurityConfig.java)**

Make sure Spring Security allows /authenticate:

@EnableWebSecurity  
public class SecurityConfig extends WebSecurityConfigurerAdapter {  
 @Override  
 protected void configure(HttpSecurity http) throws Exception {  
 http.csrf().disable()  
 .authorizeRequests()  
 .antMatchers("/authenticate").permitAll()  
 .anyRequest().authenticated()  
 .and().httpBasic();  
 }

}

**Mohammed Ayaan**

**Supersetid – 6409789**