**6.5 - Create authentication service that returns JWT**

**Objective:**

To implement a simple JWT Authentication Service in Spring Boot. The service accepts user credentials using HTTP Basic Authentication and returns a JWT token as response.

When the following CURL command is executed:

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

The response is:

{

"token": "eyJhbGciOiJIUzI1NiJ9..."

}

Technologies used in Creating authentication service that returns JWT.

* Spring Boot 3.x
* Spring Security
* Java JWT
* Maven
* Java 17+

**Structure of project:**

com.cognizant.jwt

├── config

│ └── SecurityConfig.java

├── controller

│ └── AuthController.java

├── util

│ └── JwtUtil.java

├── model

│ └── AuthRequest.java

└── SpringJwtApplication.java

Steps for creating authentication:

**Step1:** Setup Spring Boot project.

1. Create a Spring Boot project with dependencies:

* spring-boot-starter-security
* jjwt
* spring-boot-starter-web

1. Add dependency in pom.xml

|  |
| --- |
| 1. <dependency> 2. <groupId>io.jsonwebtoken</groupId> 3. <artifactId>jjwt</artifactId> 4. <version>0.9.1</version> 5. </dependency> |

**Step2:** Create a class that generate JWT tokens.

i.e., JwtUtil.java

|  |
| --- |
| package com.cognizant.jwt.util;  import io.jsonwebtoken.Jwts;  import io.jsonwebtoken.SignatureAlgorithm;  import org.springframework.stereotype.Component;  import java.util.Date;  @Component  public class JwtUtil {  private static final String SECRET\_KEY = "mySecretKey";  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 expiry  .signWith(SignatureAlgorithm.HS256, SECRET\_KEY)  .compact();  }  } |

**Step3**: Create a controller to handle/ authenticate endpoint.

i.e., Authcontroller.java

|  |
| --- |
| package com.cognizant.jwt.controller;  import com.cognizant.jwt.util.JwtUtil;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.http.ResponseEntity;  import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  import org.springframework.web.bind.annotation.\*;  import java.util.Base64;  @RestController  public class AuthController {  @Autowired  private JwtUtil jwtUtil;  @GetMapping("/authenticate")  public ResponseEntity<?> authenticate(@RequestHeader("Authorization") String authHeader) {  // Decode Basic Auth credentials  String base64Credentials = authHeader.substring("Basic ".length()).trim();  byte[] credDecoded = Base64.getDecoder().decode(base64Credentials);  String credentials = new String(credDecoded);  final String[] values = credentials.split(":", 2);  String username = values[0];  String password = values[1];  // Validate credentials (hardcoded for demo)  if ("user".equals(username) && "pwd".equals(password)) {  String token = jwtUtil.generateToken(username);  return ResponseEntity.ok().body("{\"token\":\"" + token + "\"}");  } else {  return ResponseEntity.status(401).body("{\"error\":\"Invalid credentials\"}");  }  }  } |

**Step4:** Configure Spring Security.

i.e., SecurityConfig.java

|  |
| --- |
| package com.cognizant.jwt.config;  import org.springframework.context.annotation.Bean;  import org.springframework.context.annotation.Configuration;  import org.springframework.security.config.annotation.web.builders.HttpSecurity;  import org.springframework.security.web.SecurityFilterChain;  @Configuration  public class SecurityConfig {  @Bean  public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {  http.csrf().disable()  .authorizeHttpRequests()  .requestMatchers("/authenticate").permitAll()  .anyRequest().authenticated()  .and()  .httpBasic();  return http.build();  }  } |

**Step5**: Start Spring Boot App

i.e., run the SpringJWTApplication.java as Spring Boot App.

|  |
| --- |
| package com.cognizant.jwt;  import org.springframework.boot.SpringApplication;  import org.springframework.boot.autoconfigure.SpringBootApplication;  @SpringBootApplication  public class SpringJwtApplication {  public static void main(String[] args) {  SpringApplication.run(SpringJwtApplication.class, args);  }  } |

**Expected output:**

. \_\_\_\_ \_ \_\_ \_ \_

/\\ / \_\_\_'\_ \_\_ \_ \_(\_)\_ \_\_ \_\_ \_ \ \ \ \

( ( )\\_\_\_ | '\_ | '\_| | '\_ \/ \_` | \ \ \ \

\\/ \_\_\_)| |\_)| | | | | || (\_| | ) ) ) )

' |\_\_\_\_| .\_\_|\_| |\_|\_| |\_\\_\_, | / / / /

=========|\_|==============|\_\_\_/=/\_/\_/\_/

[32m :: Spring Boot :: [39m [2m (v3.2.5)[0;39m

[2m2025-07-14T01:08:51.656+05:30[0;39m [32m INFO[0;39m [35m10596[0;39m [2m---[0;39m [2m[spring\_JWT] [ main][0;39m [2m[0;39m[36mcom.cognizant.jwt.SpringJwtApplication [0;39m [2m:[0;39m Starting SpringJwtApplication using Java 21.0.7 with PID 10596 (C:\Users\HP\Documents\workspace-spring-tools-for-eclipse-4.31.0.RELEASE\spring\_JWT\target\classes started by HP in C:\Users\HP\Documents\workspace-spring-tools-for-eclipse-4.31.0.RELEASE\spring\_JWT)

[2m2025-07-14T01:08:51.660+05:30[0;39m [32m INFO[0;39m [35m10596[0;39m [2m---[0;39m [2m[spring\_JWT] [ main][0;39m [2m[0;39m[36mcom.cognizant.jwt.SpringJwtApplication [0;39m [2m:[0;39m No active profile set, falling back to 1 default profile: "default"

[2m2025-07-14T01:08:52.569+05:30[0;39m [32m INFO[0;39m [35m10596[0;39m [2m---[0;39m [2m[spring\_JWT] [ main][0;39m [2m[0;39m[36mo.s.b.w.embedded.tomcat.TomcatWebServer [0;39m [2m:[0;39m Tomcat initialized with port 8096 (http)

[2m2025-07-14T01:08:52.580+05:30[0;39m [32m INFO[0;39m [35m10596[0;39m [2m---[0;39m [2m[spring\_JWT] [ main][0;39m [2m[0;39m[36mo.apache.catalina.core.StandardService [0;39m [2m:[0;39m Starting service [Tomcat]

[2m2025-07-14T01:08:52.581+05:30[0;39m [32m INFO[0;39m [35m10596[0;39m [2m---[0;39m [2m[spring\_JWT] [ main][0;39m [2m[0;39m[36mo.apache.catalina.core.StandardEngine [0;39m [2m:[0;39m Starting Servlet engine: [Apache Tomcat/10.1.20]

[2m2025-07-14T01:08:52.640+05:30[0;39m [32m INFO[0;39m [35m10596[0;39m [2m---[0;39m [2m[spring\_JWT] [ main][0;39m [2m[0;39m[36mo.a.c.c.C.[Tomcat].[localhost].[/] [0;39m [2m:[0;39m Initializing Spring embedded WebApplicationContext

[2m2025-07-14T01:08:52.640+05:30[0;39m [32m INFO[0;39m [35m10596[0;39m [2m---[0;39m [2m[spring\_JWT] [ main][0;39m [2m[0;39m[36mw.s.c.ServletWebServerApplicationContext[0;39m [2m:[0;39m Root WebApplicationContext: initialization completed in 921 ms

[2m2025-07-14T01:08:52.761+05:30[0;39m [33m WARN[0;39m [35m10596[0;39m [2m---[0;39m [2m[spring\_JWT] [ main][0;39m [2m[0;39m[36m.s.s.UserDetailsServiceAutoConfiguration[0;39m [2m:[0;39m

Using generated security password: 2f4d2d55-617d-475e-9b30-289fa44ca5b7

This generated password is for development use only. Your security configuration must be updated before running your application in production.

[2m2025-07-14T01:08:52.960+05:30[0;39m [32m INFO[0;39m [35m10596[0;39m [2m---[0;39m [2m[spring\_JWT] [ main][0;39m [2m[0;39m[36mo.s.s.web.DefaultSecurityFilterChain [0;39m [2m:[0;39m Will secure any request with [org.springframework.security.web.session.DisableEncodeUrlFilter@64c781a9, org.springframework.security.web.context.request.async.WebAsyncManagerIntegrationFilter@71ed560f, org.springframework.security.web.context.SecurityContextHolderFilter@c386958, org.springframework.security.web.header.HeaderWriterFilter@633fd91, org.springframework.web.filter.CorsFilter@5b88b8e, org.springframework.security.web.authentication.logout.LogoutFilter@76a14c8d, org.springframework.security.web.savedrequest.RequestCacheAwareFilter@44d64d4e, org.springframework.security.web.servletapi.SecurityContextHolderAwareRequestFilter@1dd74143, org.springframework.security.web.authentication.AnonymousAuthenticationFilter@58253c57, org.springframework.security.web.access.ExceptionTranslationFilter@7c781c42, org.springframework.security.web.access.intercept.AuthorizationFilter@5c9934da]

[2m2025-07-14T01:08:53.145+05:30[0;39m [32m INFO[0;39m [35m10596[0;39m [2m---[0;39m [2m[spring\_JWT] [ main][0;39m [2m[0;39m[36mo.s.b.w.embedded.tomcat.TomcatWebServer [0;39m [2m:[0;39m Tomcat started on port 8096 (http) with context path ''

[2m2025-07-14T01:08:53.153+05:30[0;39m [32m INFO[0;39m [35m10596[0;39m [2m---[0;39m [2m[spring\_JWT] [ main][0;39m [2m[0;39m[36mcom.cognizant.jwt.SpringJwtApplication [0;39m [2m:[0;39m Started SpringJwtApplication in 1.846 seconds (process running for 2.474)

[2m2025-07-14T01:09:24.104+05:30[0;39m [32m INFO[0;39m [35m10596[0;39m [2m---[0;39m [2m[spring\_JWT] [nio-8096-exec-3][0;39m [2m[0;39m[36mo.a.c.c.C.[Tomcat].[localhost].[/] [0;39m [2m:[0;39m Initializing Spring DispatcherServlet 'dispatcherServlet'

[2m2025-07-14T01:09:24.104+05:30[0;39m [32m INFO[0;39m [35m10596[0;39m [2m---[0;39m [2m[spring\_JWT] [nio-8096-exec-3][0;39m [2m[0;39m[36mo.s.web.servlet.DispatcherServlet [0;39m [2m:[0;39m Initializing Servlet 'dispatcherServlet'

[2m2025-07-14T01:09:24.105+05:30[0;39m [32m INFO[0;39m [35m10596[0;39m [2m---[0;39m [2m[spring\_JWT] [nio-8096-exec-3][0;39m [2m[0;39m[36mo.s.web.servlet.DispatcherServlet [0;39m [2m:[0;39m Completed initialization in 1 ms

**Step6:** Use postman.

Create new file and make GET request.

Enter the below URL:

i.e., <http://localhost:8096/authenticate>

In Authorization,

Make type as Basic Auth.

Enter credentials username = user,

Password = pwd

Then the response in the body is,

