**Create Authentication Service That Returns Jwt**

**Spring REST using Spring Boot 3**

Vaishnavi

12/7/2025

This Spring Boot application implements a basic **JWT-based authentication service**. It allows a client to send user credentials via HTTP Basic Auth to an /authenticate endpoint and receive a signed **JSON Web Token (JWT)** in response. The token can later be used to access protected resources securely.

**Objective:**

* To create a secure RESTful /authenticate endpoint that accepts user credentials via HTTP Basic Authentication.
* To validate the credentials and generate a JWT token using the io.jsonwebtoken (JJWT) library.
* To configure Spring Security to permit public access to the authentication endpoint while securing other routes.

**Implementation:**

### Generate Project from Spring Initializr

Go to: [https://start.spring.io](https://start.spring.io" \t "_new)

Add Dependencies:

Spring Web

Spring Boot DevTools.

Click **Generate**, and download the .zip file.

Extract the zip file to a suitable folder.

### Open Project in IntelliJ IDEA

Open IntelliJ IDEA.

Click **File > Open**.

Browse and select the extracted spring-learn folder.

IntelliJ will auto-detect it as a Maven project and build it.

If prompted to import as Maven project, click **Yes** or **Import**.

**JwtAuthApplication.java:**

package com.cognizant.jwt.jwt\_auth;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class JwtAuthApplication {

public static void main(String[] args) {

SpringApplication.run(JwtAuthApplication.class, args);

}

}

**JwtUtil.java:**

package com.cognizant.jwt.jwt\_auth;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import java.util.Date;

public class JwtUtil {

private static final String SECRET\_KEY = "secret-key";

private static final long EXPIRATION\_TIME = 1000 \* 60 \* 60;

public static String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date(System.currentTimeMillis()))

.setExpiration(new Date(System.currentTimeMillis() EXPIRATION\_TIME))

.signWith(SignatureAlgorithm.HS256, SECRET\_KEY)

.compact();

}

}

**AuthenticationController.java:**

package com.cognizant.jwt.jwt\_auth;

import org.springframework.http.ResponseEntity;

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

import javax.servlet.http.HttpServletRequest;

import java.util.Base64;

@RestController

public class AuthenticationController {

@GetMapping("/authenticate")

public ResponseEntity<?> authenticate(HttpServletRequest request) {

String authHeader = request.getHeader("Authorization");

if (authHeader == null || !authHeader.startsWith("Basic ")) {

return ResponseEntity.status(401).body("Missing or invalid Authorization header");

}

String base64Credentials = authHeader.substring("Basic ".length());

String credentials = new String(Base64.getDecoder().decode(base64Credentials));

String[] values = credentials.split(":", 2);

String username = values[0];

String password = values[1];

if ("user".equals(username) && "pwd".equals(password)) {

String token = JwtUtil.generateToken(username);

return ResponseEntity.ok().body("{\"token\":\"" + token + "\"}");

} else {

return ResponseEntity.status(401).body("Invalid credentials");

}

}

}

**SecurityConfig.java:**

package com.cognizant.jwt.jwt\_auth;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.web.SecurityFilterChain;

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

@Configuration

public class SecurityConfig {

@Bean

public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

http

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

.authorizeHttpRequests(auth -> auth

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

.anyRequest().authenticated()

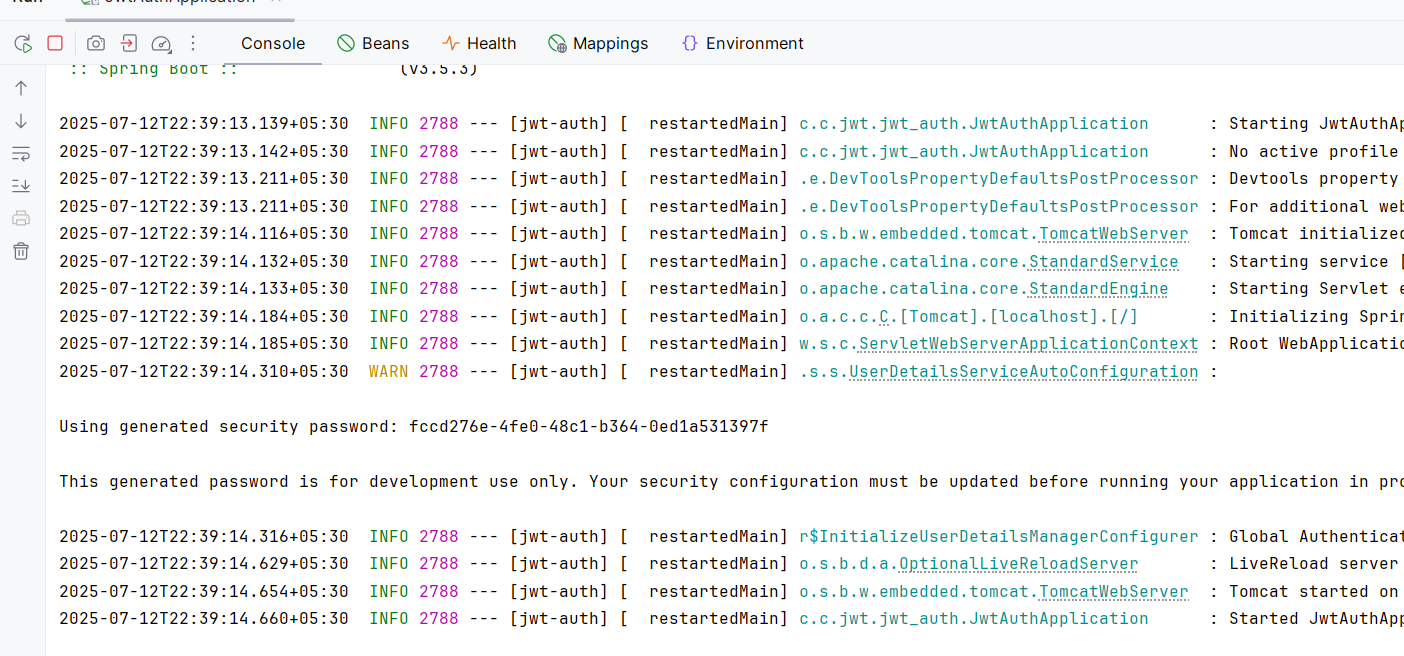
);

return http.build();

}

}

**Output:**

****