**Create authentication service that returns JWT**

**Create authentication controller and configure it in SecurityConfig**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.  
  
**package** com.cognizant.spring\_learn.controller;

**import** io.jsonwebtoken.Jwts;

**import** io.jsonwebtoken.SignatureAlgorithm;

**import** io.jsonwebtoken.security.Keys;

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

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

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** java.security.Key;

**import** java.util.Date;

@RestController

**public** **class** AuthenticationController {

**private** **static** **final** Logger ***LOGGER*** = LoggerFactory.*getLogger*(AuthenticationController.**class**);

// Generate secure secret key (256-bit) for HMAC-SHA256

**private** **static** **final** Key ***SECRET\_KEY*** = Keys.secretKeyFor(SignatureAlgorithm.***HS256***);

@GetMapping("/authenticate")

**public** String authenticate() {

***LOGGER***.info("START - /authenticate");

String jwt = Jwts.*builder*()

.setSubject("user") // your username

.setIssuedAt(**new** Date())

.setExpiration(**new** Date(System.*currentTimeMillis*() + 86400000)) // 1 day

.signWith(***SECRET\_KEY***)

.compact();

***LOGGER***.info("END - /authenticate");

**return** "{\"token\": \"" + jwt + "\"}";

}

}

**SecurityConfig.java**

**package** com.cognizant.spring\_learn.security;

**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()

.authorizeRequests()

.antMatchers("/authenticate").permitAll() // allow this endpoint

.anyRequest().authenticated(); // all others require login

**return** http.build();

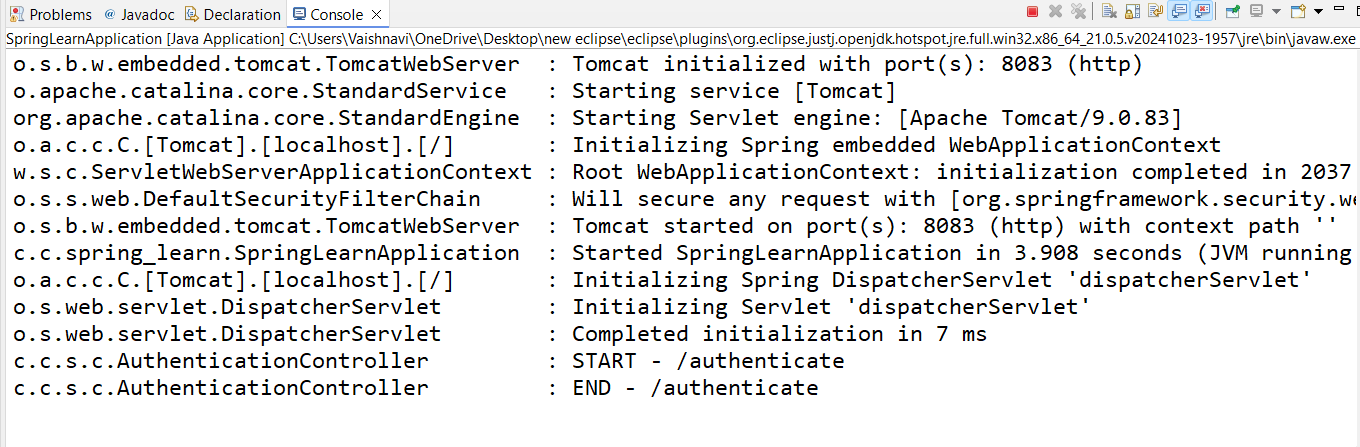
}

}

**JSON Response:**

****

**Logs (in console):**

****

**Read Authorization header and decode the username and password**

**Authenticationcontroller.java**

package com.cognizant.spring\_learn.controller;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import java.util.Base64;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

import java.security.Key;

@RestController

public class AuthenticationController {

private static final Logger *logger* = LoggerFactory.*getLogger*(AuthenticationController.class);

private static final Key *SECRET\_KEY* = Keys.*secretKeyFor*(SignatureAlgorithm.*HS256*);

@GetMapping("/authenticate")

public Map<String, String> authenticate(@RequestHeader("Authorization") String authHeader) {

*logger*.info("START - /authenticate");

*logger*.debug("Authorization Header: {}", authHeader);

String user = getUser(authHeader);

*logger*.debug("Decoded Username: {}", user);

String jwt = Jwts.*builder*()

.setSubject(user)

.setIssuedAt(new Date())

.setExpiration(new Date(System.*currentTimeMillis*() + 86400000)) // 1 day

.signWith(*SECRET\_KEY*)

.compact();

Map<String, String> map = new HashMap<>();

map.put("token", jwt);

*logger*.info("END - /authenticate");

return map;

}

private String getUser(String authHeader) {

*logger*.debug("START - getUser()");

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

byte[] decodedBytes = Base64.*getDecoder*().decode(encodedCredentials);

String decoded = new String(decodedBytes);

*logger*.debug("Decoded credentials: {}", decoded);

String username = decoded.split(":")[0];

*logger*.debug("Extracted username: {}", username);

return username;

}

}

**Pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>spring-learn</name>

<description>Spring Learn Project</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.7.18</version>

<relativePath/>

</parent>

<properties>

<java.version>21</java.version>

</properties>

<dependencies>

<!-- Spring Boot Starters -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

<!-- JWT Dependencies -->

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-api</artifactId>

<version>0.11.5</version>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-impl</artifactId>

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-jackson</artifactId>

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

<!-- Logging -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

</dependency>

<!-- Testing -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

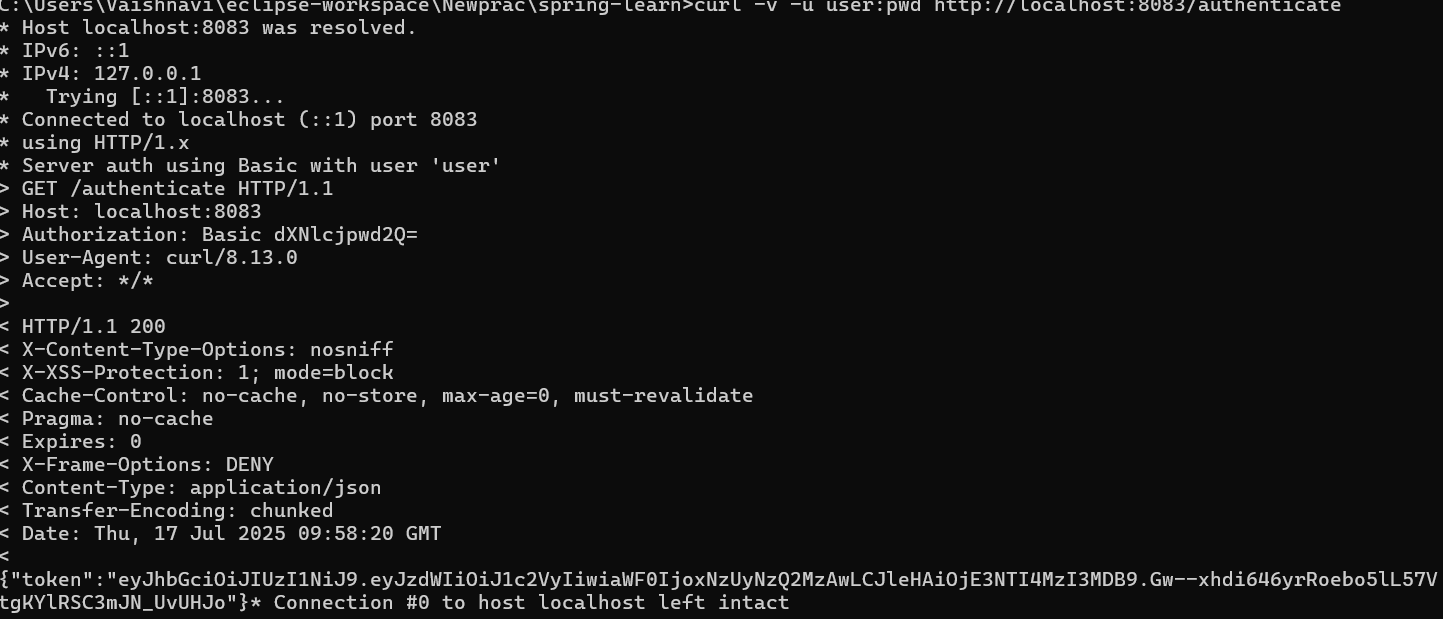
</build>

</project>

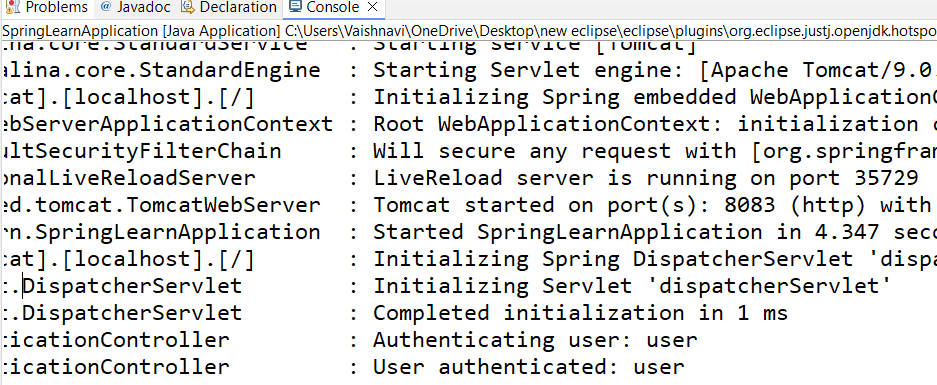
**Curl Command Used:**

curl -v -u user:pwd <http://localhost:8083/authenticate>

**Response:**

****

**Console log:**

****

**Generate token based on the user  
Securityconfig.java**

package com.cognizant.spring\_learn.security;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;

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

import org.springframework.security.web.SecurityFilterChain;

import org.springframework.security.core.userdetails.User;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.security.config.http.SessionCreationPolicy;

@Configuration

public class SecurityConfig {

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

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

.authorizeRequests(auth -> auth

.antMatchers("/auth/authenticate").permitAll()

.anyRequest().authenticated()

)

.sessionManagement(sess -> sess.sessionCreationPolicy(SessionCreationPolicy.*STATELESS*));

return http.build();

}

@Bean

public AuthenticationManager authenticationManager(AuthenticationConfiguration config) throws Exception {

return config.getAuthenticationManager();

}

@Bean

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

@Bean

public UserDetailsService userDetailsService() {

String encodedPwd = passwordEncoder().encode("pwd");

System.*out*.println("Encoded password: " + encodedPwd); // <-- add this

UserDetails user = User.*builder*()

.username("user")

.password(encodedPwd)

.roles("USER")

.build();

return new InMemoryUserDetailsManager(user);

}

}

**Authenticationcontroller.java**

**package** com.cognizant.spring\_learn.controller;

**import** io.jsonwebtoken.JwtBuilder;

**import** io.jsonwebtoken.Jwts;

**import** io.jsonwebtoken.SignatureAlgorithm;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** org.springframework.security.authentication.AuthenticationManager;

**import** org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

**import** org.springframework.security.core.userdetails.UserDetails;

**import** org.springframework.security.core.userdetails.UserDetailsService;

**import** org.springframework.beans.factory.annotation.Autowired;

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

**import** java.util.Date;

**import** java.util.HashMap;

**import** java.util.Map;

@RestController

@RequestMapping("/auth")

**public** **class** AuthenticationController {

**private** **static** **final** Logger ***LOGGER*** = LoggerFactory.*getLogger*(AuthenticationController.**class**);

@Autowired

**private** AuthenticationManager authenticationManager;

@Autowired

**private** UserDetailsService userDetailsService;

@PostMapping("/authenticate")

**public** Map<String, String> authenticate(@RequestBody Map<String, String> requestMap) {

String username = requestMap.get("username");

String password = requestMap.get("password");

***LOGGER***.info("Authenticating user: {}", username);

authenticationManager.authenticate(**new** UsernamePasswordAuthenticationToken(username, password));

***LOGGER***.info("User authenticated: {}", username);

String token = generateJwt(username);

Map<String, String> response = **new** HashMap<>();

response.put("token", token);

***LOGGER***.info("Token generated: {}", token);

**return** response;

}

// Token Generator

**private** String generateJwt(String user) {

JwtBuilder builder = Jwts.*builder*();

builder.setSubject(user);

builder.setIssuedAt(**new** Date());

builder.setExpiration(**new** Date((**new** Date()).getTime() + 1200000)); // 20 minutes

builder.signWith(SignatureAlgorithm.***HS256***, "secretkey");

**return** builder.compact();

}

}

**Pom.xml:**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>spring-learn</name>

<description>Spring Boot project with JWT</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.7.18</version>

<relativePath/>

</parent>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<!-- Spring Boot Starter -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter</artifactId>

</dependency>

<!-- Spring Web -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<!-- Spring Security -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

<!-- JWT -->

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.0</version>

</dependency>

<!-- Devtools (Optional, for hot reload) -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

</dependency>

<!-- Logging -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-logging</artifactId>

</dependency>

<!-- Testing (Optional) -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>javax.xml.bind</groupId>

<artifactId>jaxb-api</artifactId>

<version>2.3.1</version>

</dependency>

<dependency>

<groupId>org.glassfish.jaxb</groupId>

<artifactId>jaxb-runtime</artifactId>

<version>2.3.1</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

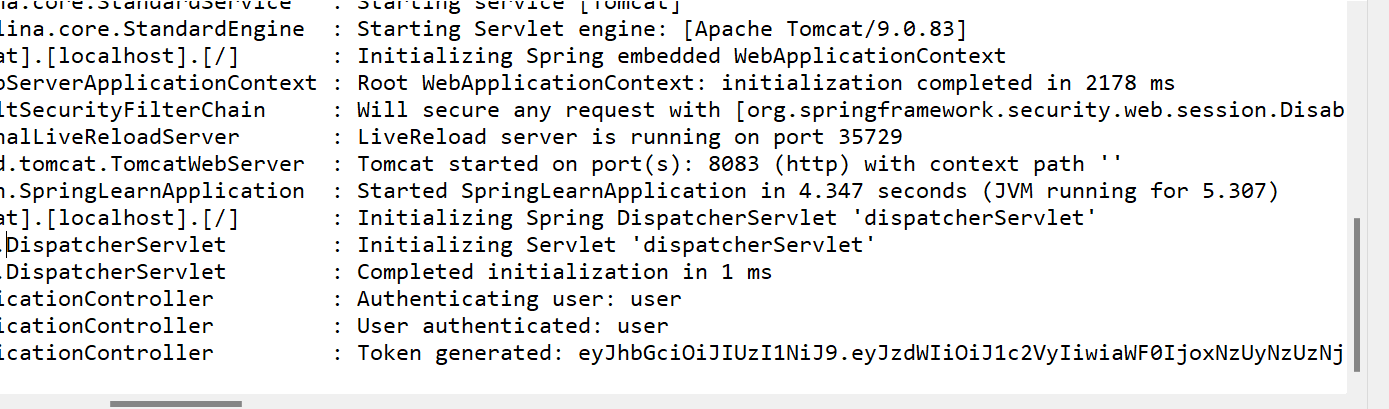
</plugin>

</plugins>

</build>

</project>

**CONSOLE:**



**TERMINAL:**

