Name : Arshiya Tabassum A

Superset ID : 6424209

**WEEK 4**

**Securing RESTful Web Services with Spring Security**

**Pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<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>

<parent>

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

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

<version>3.5.3</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-learn</name>

<description>Demo project for Spring Boot</description>

<properties>

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

</properties>

<dependencies>

<!-- Spring Web -->

<dependency>

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

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

</dependency>

<!-- Spring Validation -->

<dependency>

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

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

</dependency>

<!-- Spring Security -->

<dependency>

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

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

</dependency>

<!-- DevTools for hot reload -->

<dependency>

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

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

<scope>runtime</scope>

<optional>true</optional>

</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>

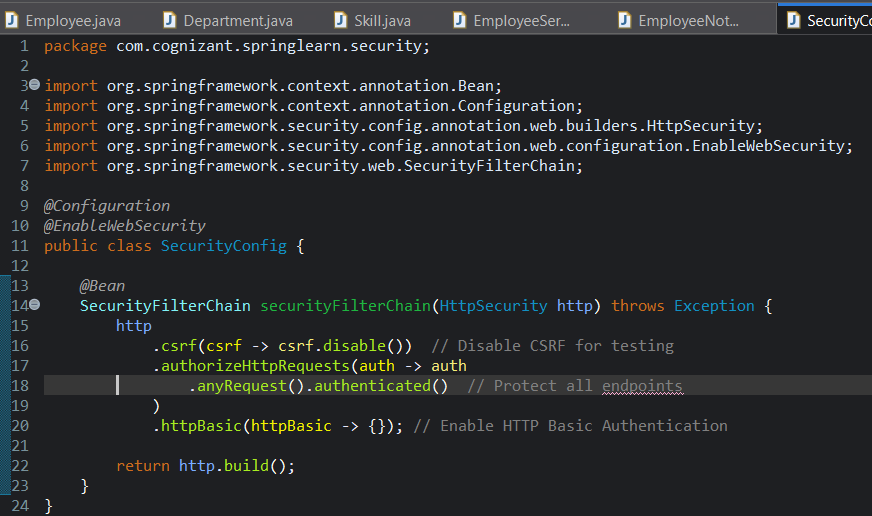
</plugin>

</plugins>

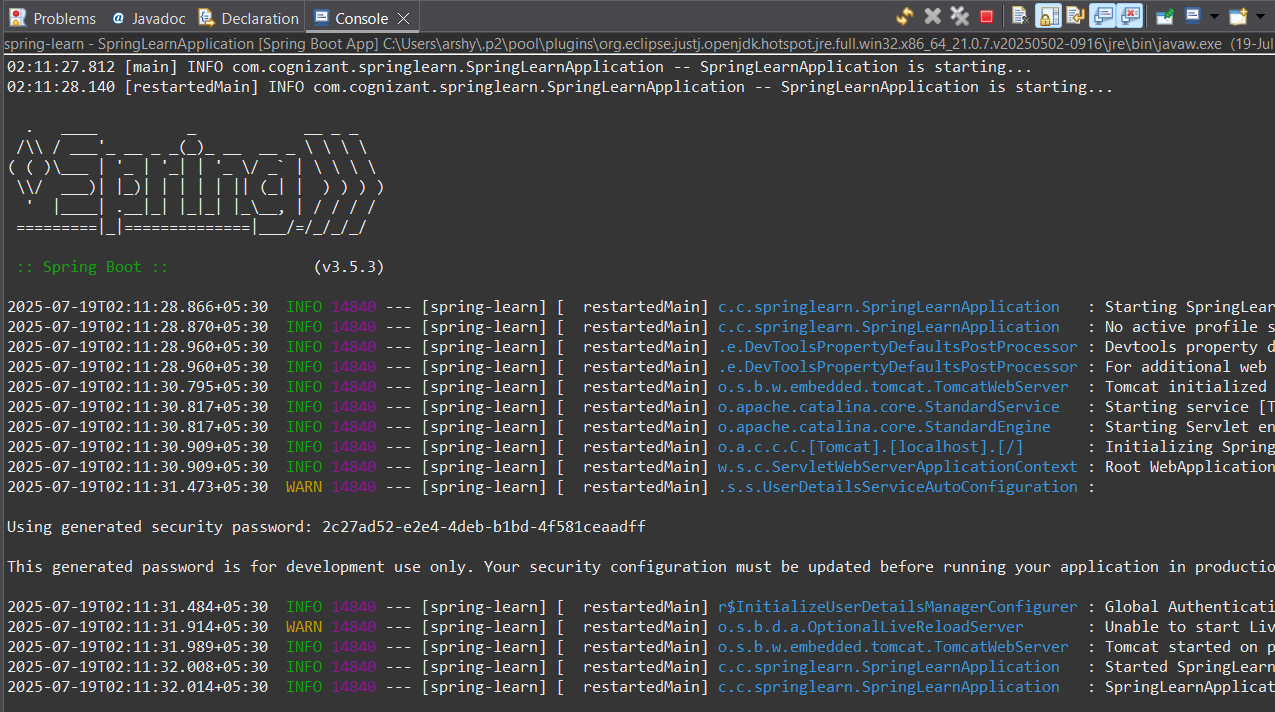
</build>

</project>

**SecurityConfig.java**

****

**Output :**

****

**Creating users and roles in Spring Security**

**Pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<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>

<parent>

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

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

<version>3.5.3</version>

<relativePath/>

</parent>

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-learn</name>

<description>Demo project for Spring Boot</description>

<properties>

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

</properties>

<dependencies>

<!-- Spring Web -->

<dependency>

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

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

</dependency>

<!-- Spring Validation -->

<dependency>

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

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

</dependency>

<!-- Spring Security -->

<dependency>

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

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

</dependency>

<!-- Jakarta Servlet API (needed for some servlet containers) -->

<dependency>

<groupId>jakarta.servlet</groupId>

<artifactId>jakarta.servlet-api</artifactId>

<scope>provided</scope>

</dependency>

<!-- DevTools for hot reload -->

<dependency>

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

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

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<!-- Testing -->

<dependency>

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

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

<scope>test</scope>

</dependency>

<!-- Spring Security Test -->

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-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>

**SecurityConfig.java**

package com.cognizant.springlearn.security;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

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

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

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

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

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

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

import org.springframework.security.web.SecurityFilterChain;

*@Configuration*

*@EnableWebSecurity*

class SecurityConfig {

// Step 1: Create in-memory users

*@Bean*

InMemoryUserDetailsManager userDetailsService(PasswordEncoder passwordEncoder) {

UserDetails admin = User.*withUsername*("admin")

.password(passwordEncoder.encode("pwd"))

.roles("ADMIN")

.build();

UserDetails user = User.*withUsername*("user")

.password(passwordEncoder.encode("pwd"))

.roles("USER")

.build();

return new InMemoryUserDetailsManager(admin, user);

}

// Step 2: Password encoder (BCrypt)

*@Bean*

PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

// Step 3: Configure role-based security with new Lambda DSL

*@Bean*

SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

http

.csrf(csrf -> csrf.disable()) // Disable CSRF for testing

.authorizeHttpRequests(auth -> auth

.requestMatchers("/countries").hasRole("USER") // Only USER can access /countries

.anyRequest().authenticated()

)

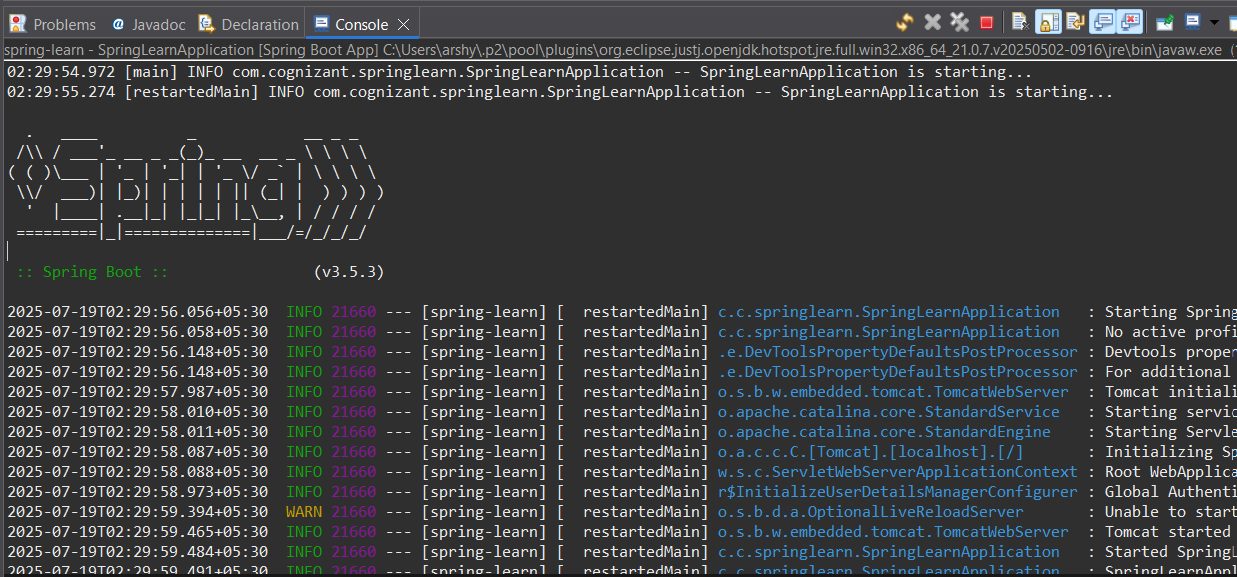
.httpBasic(basic -> {}); // Replaces deprecated httpBasic()

return http.build();

}

}

**Output :**

****

**Limitations of this security approach**

**CountryController.java**

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.model.Country;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import jakarta.validation.Valid;

import java.util.ArrayList;

import java.util.List;

*@RestController*

*@RequestMapping*("/countries")

public class CountryController {

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

/\*\*

\* Handles GET requests to fetch a list of countries.

\* Accessible only by users with role "USER".

\*/

*@GetMapping*

public List<Country> getAllCountries() {

*LOGGER*.info("Start - Inside getAllCountries()");

List<Country> countries = new ArrayList<>();

countries.add(new Country("US", "United States"));

countries.add(new Country("DE", "Germany"));

countries.add(new Country("IN", "India"));

countries.add(new Country("JP", "Japan"));

*LOGGER*.info("End - getAllCountries()");

return countries;

}

/\*\*

\* Handles POST requests to add a new Country.

\* Validation of 'code' and 'name' is automatically handled by @Valid.

\*/

*@PostMapping*

public Country addCountry(*@RequestBody* *@Valid* Country country) {

*LOGGER*.info("Start - Inside addCountry()");

*LOGGER*.info("Country received: Code={}, Name={}", country.getCode(), country.getName());

*LOGGER*.info("End - addCountry()");

return country; // Echo the validated country back to client

}

}

**SecurityConfig.java**

package com.cognizant.springlearn.security;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

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

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

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

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

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

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

import org.springframework.security.web.SecurityFilterChain;

*@Configuration*

*@EnableWebSecurity*

public class SecurityConfig { // Make class public

// Step 1: Define in-memory users

*@Bean*

public InMemoryUserDetailsManager userDetailsService(PasswordEncoder passwordEncoder) {

UserDetails admin = User.*withUsername*("admin")

.password(passwordEncoder.encode("pwd"))

.roles("ADMIN")

.build();

UserDetails user = User.*withUsername*("user")

.password(passwordEncoder.encode("pwd"))

.roles("USER")

.build();

return new InMemoryUserDetailsManager(admin, user);

}

// Step 2: Password encoder (BCrypt for encoding passwords)

*@Bean*

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

// Step 3: Secure endpoints using role-based access

*@Bean*

public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

http

.csrf(csrf -> csrf.disable()) // Disable CSRF for testing

.authorizeHttpRequests(auth -> auth

.requestMatchers("/countries").hasRole("USER") // Only USER role can access

.anyRequest().authenticated() // All others need authentication

)

.httpBasic(httpBasic -> {}); // Enable Basic Auth using new DSL

return http.build();

}

}

**Pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<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>

<parent>

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

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

<version>3.5.3</version>

<relativePath/>

</parent>

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-learn</name>

<description>Demo project for Spring Boot</description>

<properties>

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

</properties>

<dependencies>

<!-- Spring Web -->

<dependency>

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

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

</dependency>

<!-- Spring Validation -->

<dependency>

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

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

</dependency>

<!-- Spring Security -->

<dependency>

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

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

</dependency>

<!-- Jakarta Servlet API (needed for some servlet containers) -->

<dependency>

<groupId>jakarta.servlet</groupId>

<artifactId>jakarta.servlet-api</artifactId>

<scope>provided</scope>

</dependency>

<!-- DevTools for hot reload -->

<dependency>

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

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

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<!-- Testing -->

<dependency>

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

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

<scope>test</scope>

</dependency>

<!-- Spring Security Test -->

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

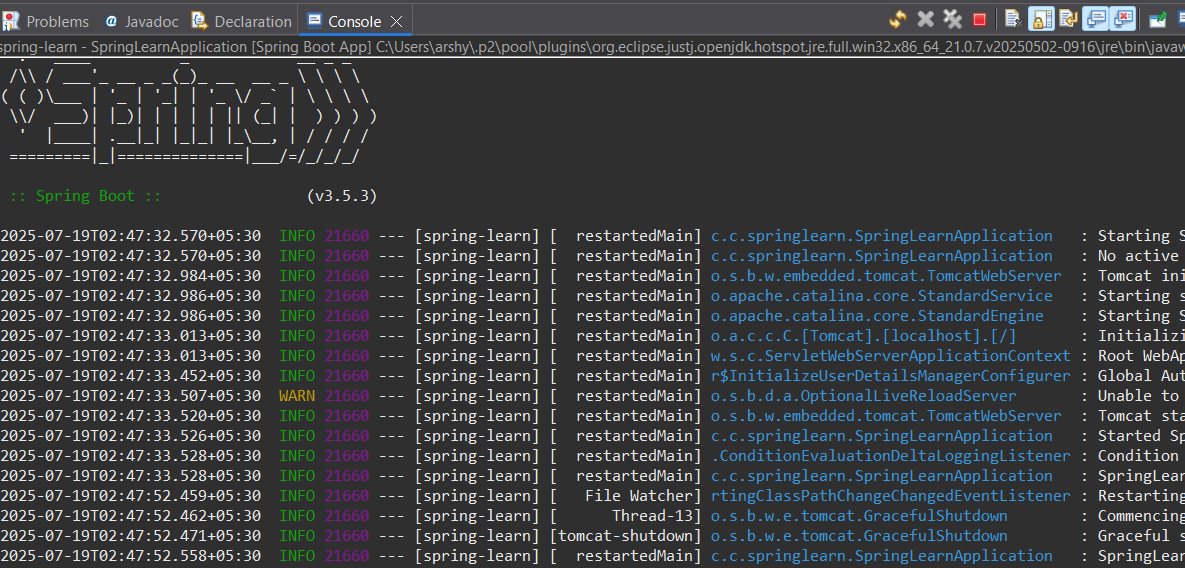
</plugin>

</plugins>

</build>

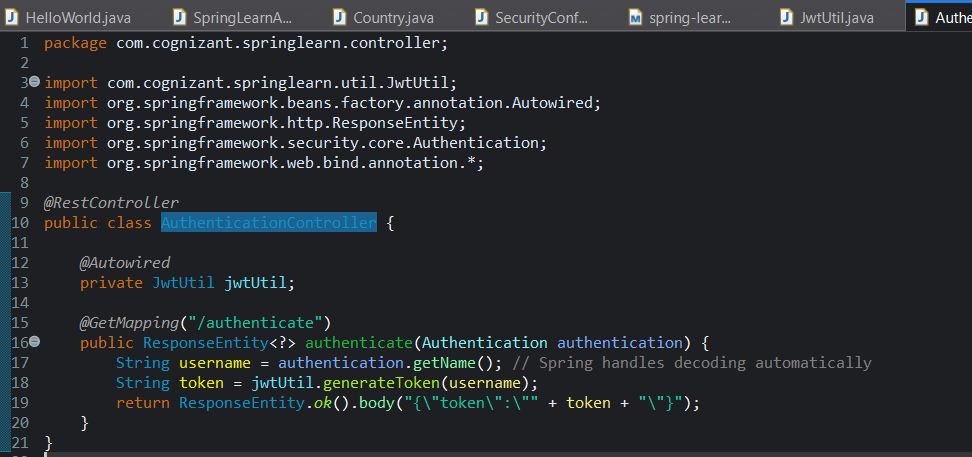
</project>

**Output :**

****

****

**Create authentication service that returns JWT   
  
AuthenticationController.java**

****

**SecurityConfig.java**

package com.cognizant.springlearn.security;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

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

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

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

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

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

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

import org.springframework.security.web.SecurityFilterChain;

*@Configuration*

*@EnableWebSecurity*

public class SecurityConfig {

// Step 1: Define in-memory users (no "public" keyword)

*@Bean*

InMemoryUserDetailsManager userDetailsService(PasswordEncoder passwordEncoder) {

UserDetails admin = User.*withUsername*("admin")

.password(passwordEncoder.encode("pwd"))

.roles("ADMIN")

.build();

UserDetails user = User.*withUsername*("user")

.password(passwordEncoder.encode("pwd"))

.roles("USER")

.build();

return new InMemoryUserDetailsManager(admin, user);

}

// Step 2: Password encoder (also remove "public")

*@Bean*

PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

// Step 3: Secure endpoints (also remove "public")

*@Bean*

SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

http

.csrf(csrf -> csrf.disable()) // Disable CSRF for APIs

.authorizeHttpRequests(auth -> auth

.requestMatchers("/authenticate").permitAll() // Open for JWT token generation

.requestMatchers("/countries").hasRole("USER") // Only USER role can access

.anyRequest().authenticated()

)

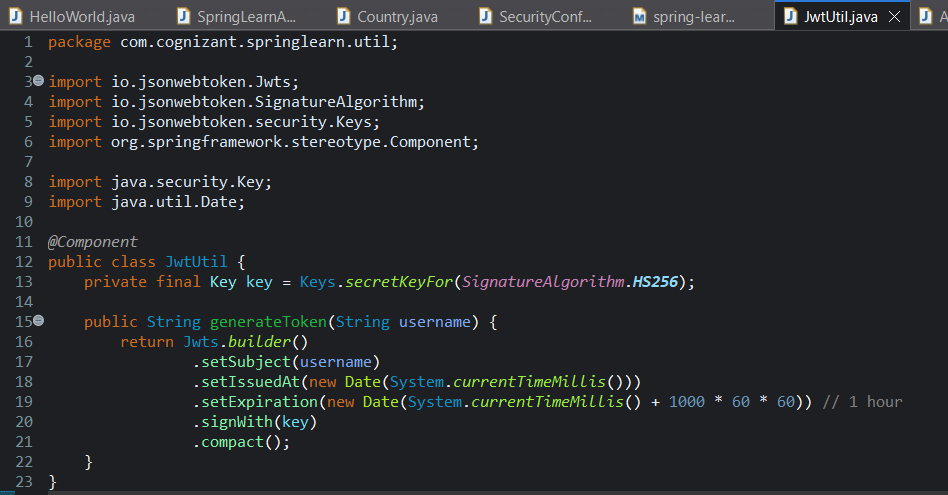
.httpBasic(httpBasic -> {}); // Use Basic Auth (for now)

return http.build();

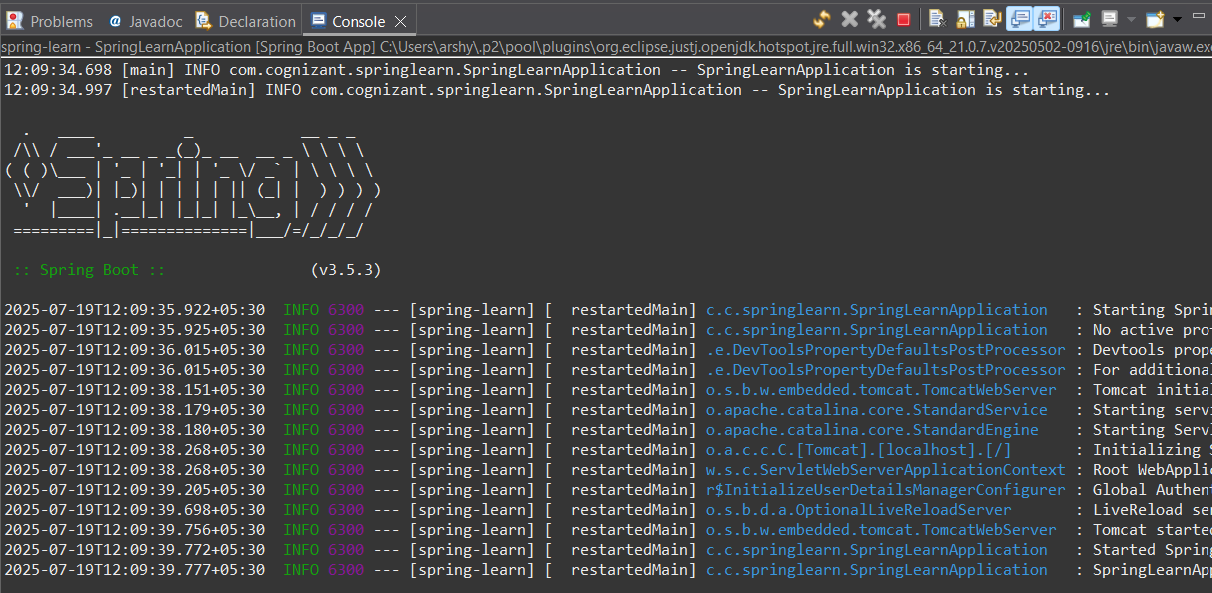
}

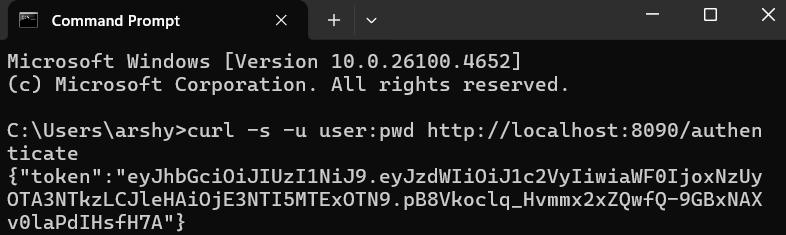
}

**JWTUtil.java**

****

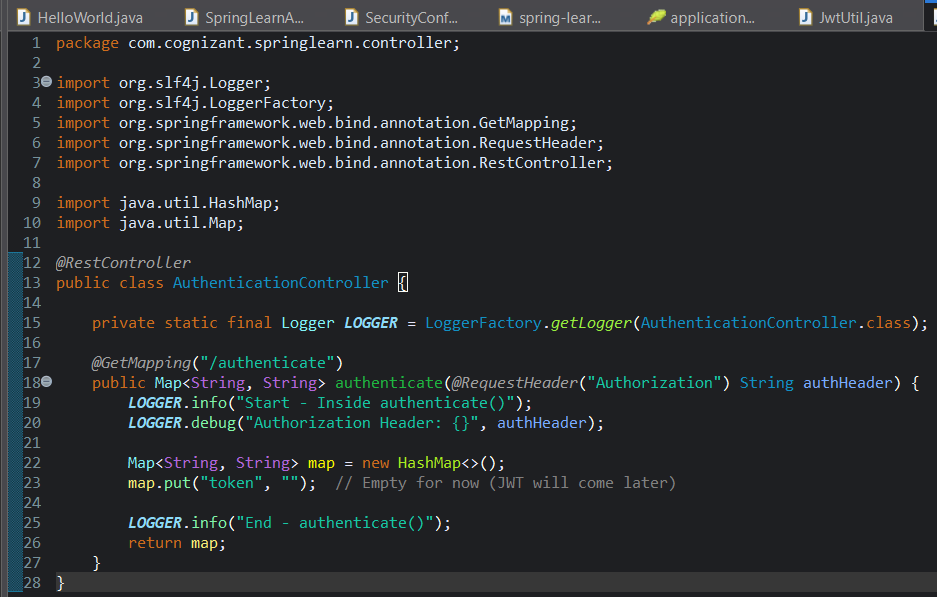
**Output :**

****

****

**Create authentication controller and configure it in SecurityConfig**

**AuthenticationController.java**

****

**SecurityConfig.java**

package com.cognizant.springlearn.security;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

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

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

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

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

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

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

import org.springframework.security.web.SecurityFilterChain;

*@Configuration*

*@EnableWebSecurity*

public class SecurityConfig {

// Step 1: In-memory users

*@Bean*

InMemoryUserDetailsManager userDetailsService(PasswordEncoder passwordEncoder) {

UserDetails admin = User.*withUsername*("admin")

.password(passwordEncoder.encode("pwd"))

.roles("ADMIN")

.build();

UserDetails user = User.*withUsername*("user")

.password(passwordEncoder.encode("pwd"))

.roles("USER")

.build();

return new InMemoryUserDetailsManager(admin, user);

}

// Step 2: Password encoder (BCrypt)

*@Bean*

PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

// Step 3: Security rules for endpoints

*@Bean*

SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

http

.csrf(csrf -> csrf.disable()) // Disable CSRF for APIs

.authorizeHttpRequests(auth -> auth

.requestMatchers("/authenticate").permitAll() // Open for token generation

.requestMatchers("/countries").hasRole("USER") // Only USER can access /countries

.anyRequest().authenticated() // Everything else requires login

)

.httpBasic(httpBasic -> {}); // Basic Authentication

return http.build();

}

}

**Pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<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>

<parent>

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

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

<version>3.5.3</version>

<relativePath/>

</parent>

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-learn</name>

<description>Demo project for Spring Boot</description>

<properties>

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

</properties>

<dependencies>

<!-- Spring Web -->

<dependency>

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

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

</dependency>

<!-- Spring Validation -->

<dependency>

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

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

</dependency>

<!-- Spring Security -->

<dependency>

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

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

</dependency>

<!-- JWT (JSON Web Token) -->

<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>

<!-- DevTools for hot reload -->

<dependency>

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

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

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<!-- Testing -->

<dependency>

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

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

<scope>test</scope>

</dependency>

<!-- Spring Security Test -->

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

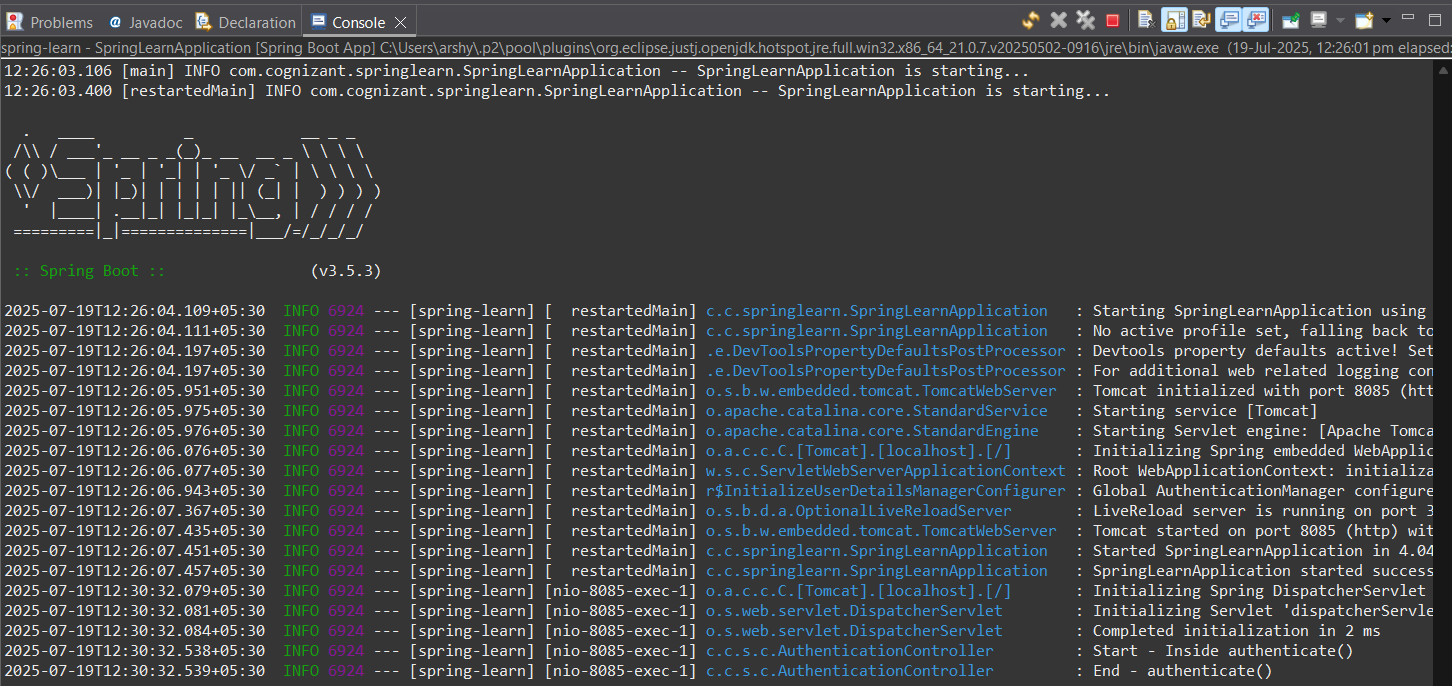
</plugin>

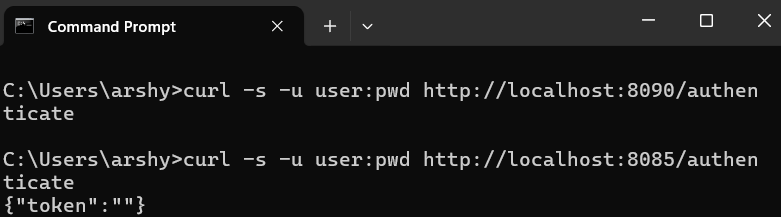
</plugins>

</build>

</project>

**Output :**

****

****

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

**AuthenticationController.java**

package com.cognizant.springlearn.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.web.bind.annotation.RequestHeader;

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

import java.util.Base64;

import java.util.HashMap;

import java.util.Map;

*@RestController*

public class AuthenticationController {

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

*@GetMapping*("/authenticate")

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

***LOGGER***.info("Start - Inside authenticate()");

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

String username = getUser(authHeader); // Extract the username

***LOGGER***.debug("Authenticated user: {}", username);

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

map.put("token", ""); // Token generation will be done in next step

***LOGGER***.info("End - authenticate()");

return map;

}

private String getUser(String authHeader) {

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

// Remove "Basic " prefix (6 characters)

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

// Decode from Base64

byte[] decodedBytes = Base64.*getDecoder*().decode(base64Credentials);

String decodedString = new String(decodedBytes); // "user:pwd"

String username = decodedString.split(":")[0]; // Extract username only

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

return username;

}

}

**SecurityConfig.java**

package com.cognizant.springlearn.security;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

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

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

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

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

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

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

import org.springframework.security.web.SecurityFilterChain;

*@Configuration*

*@EnableWebSecurity*

public class SecurityConfig {

// Step 1: In-memory users

*@Bean*

InMemoryUserDetailsManager userDetailsService(PasswordEncoder passwordEncoder) {

UserDetails admin = User.*withUsername*("admin")

.password(passwordEncoder.encode("pwd"))

.roles("ADMIN")

.build();

UserDetails user = User.*withUsername*("user")

.password(passwordEncoder.encode("pwd"))

.roles("USER")

.build();

return new InMemoryUserDetailsManager(admin, user);

}

// Step 2: Password encoder (BCrypt)

*@Bean*

PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

// Step 3: Security rules for endpoints

*@Bean*

SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

http

.csrf(csrf -> csrf.disable()) // Disable CSRF for APIs

.authorizeHttpRequests(auth -> auth

.requestMatchers("/authenticate").permitAll() // Open for token generation

.requestMatchers("/countries").hasRole("USER") // Only USER can access /countries

.anyRequest().authenticated() // Everything else requires login

)

.httpBasic(httpBasic -> {}); // Basic Authentication

return http.build();

}

}

**Pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<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>

<parent>

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

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

<version>3.5.3</version>

<relativePath/>

</parent>

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-learn</name>

<description>Demo project for Spring Boot</description>

<properties>

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

</properties>

<dependencies>

<!-- Spring Web -->

<dependency>

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

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

</dependency>

<!-- Spring Validation -->

<dependency>

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

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

</dependency>

<!-- Spring Security -->

<dependency>

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

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

</dependency>

<!-- JWT (JSON Web Token) -->

<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>

<!-- DevTools for hot reload -->

<dependency>

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

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

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<!-- Testing -->

<dependency>

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

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

<scope>test</scope>

</dependency>

<!-- Spring Security Test -->

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

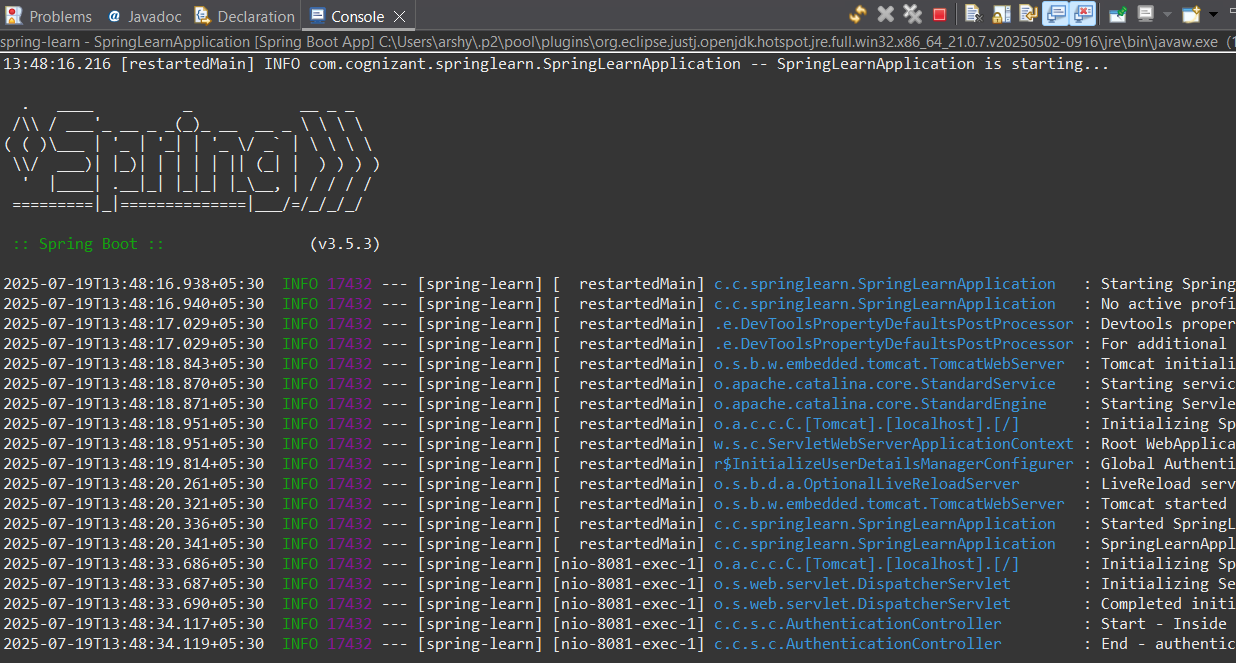
</plugin>

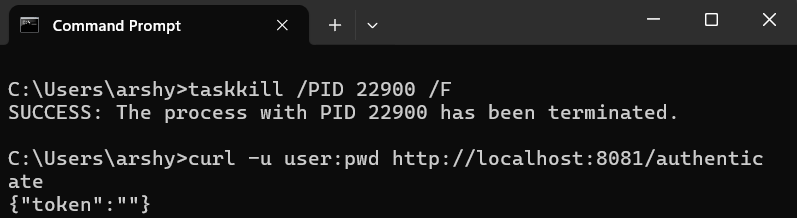
</plugins>

</build>

</project>

**Output :**

****

****

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

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.util.JwtUtil;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.web.bind.annotation.RequestHeader;

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

import java.util.Base64;

import java.util.HashMap;

import java.util.Map;

*@RestController*

public class AuthenticationController {

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

*@GetMapping*("/authenticate")

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

***LOGGER***.info("Start - Inside authenticate()");

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

String username = getUser(authHeader); // Extract the username

***LOGGER***.debug("Authenticated user: {}", username);

// Generate JWT Token for the user

String token = generateJwt(username);

// Prepare response

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

map.put("token", token);

***LOGGER***.info("End - authenticate()");

return map;

}

// Decode the Basic Auth header to extract username

private String getUser(String authHeader) {

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

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

byte[] decodedBytes = Base64.*getDecoder*().decode(base64Credentials);

String decodedString = new String(decodedBytes); // "user:pwd"

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

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

return username;

}

// Generate JWT token for the given user

private String generateJwt(String user) {

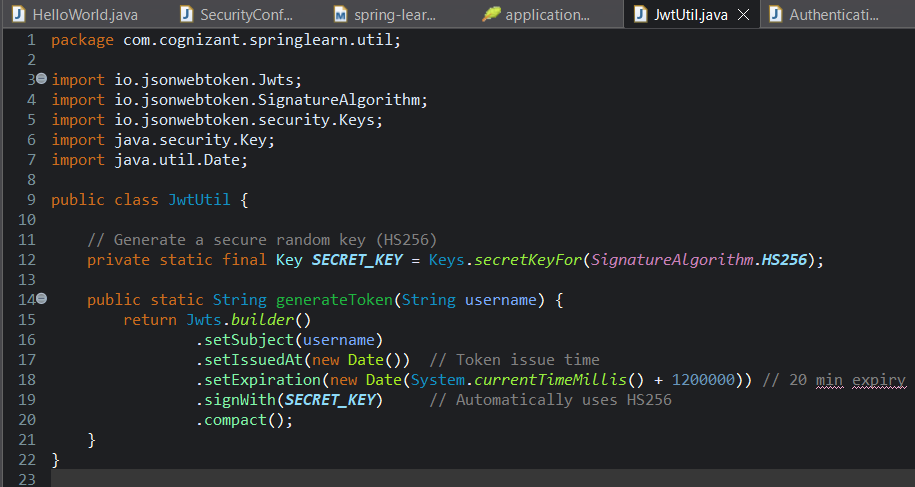
***LOGGER***.debug("Generating JWT for user: {}", user);

return JwtUtil.*generateToken*(user);

}

}

**JwtUtil.java**

****

**Pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<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>

<parent>

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

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

<version>3.5.3</version>

<relativePath/>

</parent>

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-learn</name>

<description>Demo project for Spring Boot</description>

<properties>

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

</properties>

<dependencies>

<!-- Spring Web -->

<dependency>

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

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

</dependency>

<!-- Spring Validation -->

<dependency>

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

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

</dependency>

<!-- Spring Security -->

<dependency>

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

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

</dependency>

<!-- JWT (JSON Web Token) -->

<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>

<!-- DevTools for hot reload -->

<dependency>

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

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

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<!-- Testing -->

<dependency>

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

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

<scope>test</scope>

</dependency>

<!-- Spring Security Test -->

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

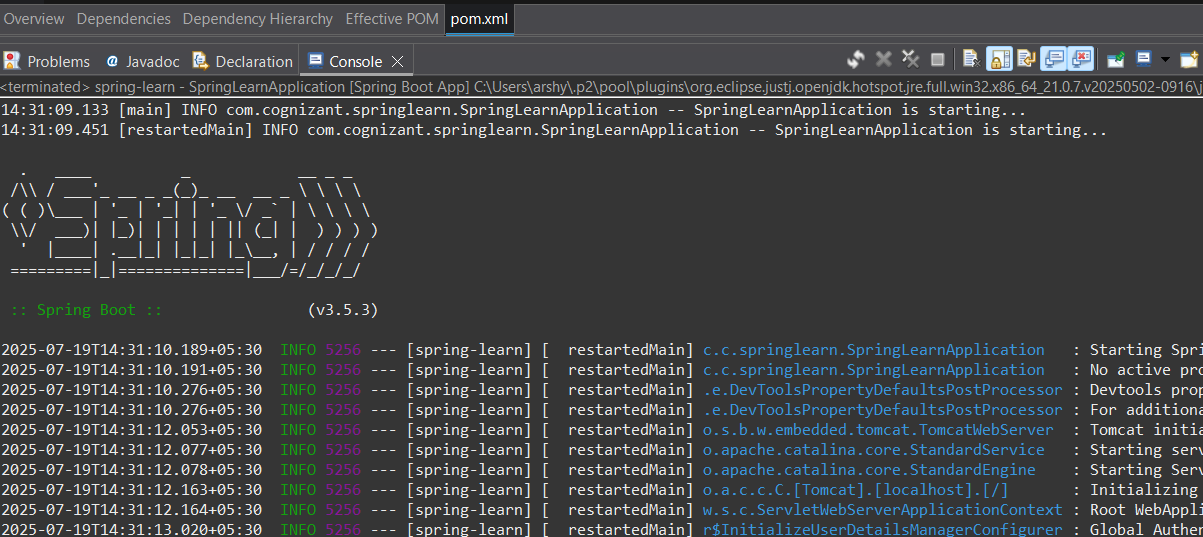
</plugin>

</plugins>

</build>

</project>

**Output :**

****

**Authorize based on JWT   
  
AuthenticationController.java**

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.util.JwtUtil;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.web.bind.annotation.RequestHeader;

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

import java.util.Base64;

import java.util.HashMap;

import java.util.Map;

*@RestController*

public class AuthenticationController {

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

*@GetMapping*("/authenticate")

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

***LOGGER***.info("Start - Inside authenticate()");

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

String username = getUser(authHeader); // Extract the username

***LOGGER***.debug("Authenticated user: {}", username);

// Generate JWT Token for the user

String token = generateJwt(username);

// Prepare response

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

map.put("token", token);

***LOGGER***.info("End - authenticate()");

return map;

}

// Decode the Basic Auth header to extract username

private String getUser(String authHeader) {

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

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

byte[] decodedBytes = Base64.*getDecoder*().decode(base64Credentials);

String decodedString = new String(decodedBytes); // "user:pwd"

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

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

return username;

}

// Generate JWT token for the given user

private String generateJwt(String user) {

***LOGGER***.debug("Generating JWT for user: {}", user);

return JwtUtil.*generateToken*(user);

}

}

**JwtAuthorizationfilter.java**

package com.cognizant.springlearn.security;

import io.jsonwebtoken.Claims;

import io.jsonwebtoken.Jws;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.JwtException;

import io.jsonwebtoken.security.Keys;

import jakarta.servlet.FilterChain;

import jakarta.servlet.ServletException;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

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.context.SecurityContextHolder;

import org.springframework.security.web.authentication.www.BasicAuthenticationFilter;

import java.io.IOException;

import java.nio.charset.StandardCharsets;

import java.security.Key;

import java.util.ArrayList;

public class JwtAuthorizationFilter extends BasicAuthenticationFilter {

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

private static final String ***SECRET*** = "secretkey"; // For demo, use env variable in production

public JwtAuthorizationFilter(AuthenticationManager authenticationManager) {

super(authenticationManager);

***LOGGER***.info("JwtAuthorizationFilter initialized");

}

*@Override*

protected void doFilterInternal(HttpServletRequest req, HttpServletResponse res, FilterChain chain)

throws IOException, ServletException {

***LOGGER***.info("JWT Filter: Start");

String header = req.getHeader("Authorization");

***LOGGER***.debug("Authorization Header: {}", header);

if (header == null || !header.startsWith("Bearer ")) {

chain.doFilter(req, res);

return;

}

UsernamePasswordAuthenticationToken authentication = getAuthentication(req);

SecurityContextHolder.*getContext*().setAuthentication(authentication);

chain.doFilter(req, res);

***LOGGER***.info("JWT Filter: End");

}

private UsernamePasswordAuthenticationToken getAuthentication(HttpServletRequest request) {

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

if (token != null) {

try {

// Convert secret string to a proper signing key

Key key = Keys.*hmacShaKeyFor*(***SECRET***.getBytes(StandardCharsets.***UTF\_8***));

Jws<Claims> jws = Jwts.*parserBuilder*()

.setSigningKey(key) // Secure key, replaces deprecated method

.build()

.parseClaimsJws(token.replace("Bearer ", ""));

String user = jws.getBody().getSubject();

***LOGGER***.debug("JWT Subject (user): {}", user);

if (user != null) {

return new UsernamePasswordAuthenticationToken(user, null, new ArrayList<>());

}

} catch (JwtException e) {

***LOGGER***.error("Invalid JWT: {}", e.getMessage());

}

}

return null;

}

}

**Pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<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>

<parent>

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

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

<version>3.5.3</version>

<relativePath/>

</parent>

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-learn</name>

<description>Demo project for Spring Boot</description>

<properties>

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

</properties>

<dependencies>

<!-- Spring Web -->

<dependency>

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

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

</dependency>

<!-- Spring Validation -->

<dependency>

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

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

</dependency>

<!-- Spring Security -->

<dependency>

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

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

</dependency>

<!-- JWT (JSON Web Token) -->

<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>

<!-- DevTools for hot reload -->

<dependency>

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

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

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<!-- Testing -->

<dependency>

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

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

<scope>test</scope>

</dependency>

<!-- Spring Security Test -->

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

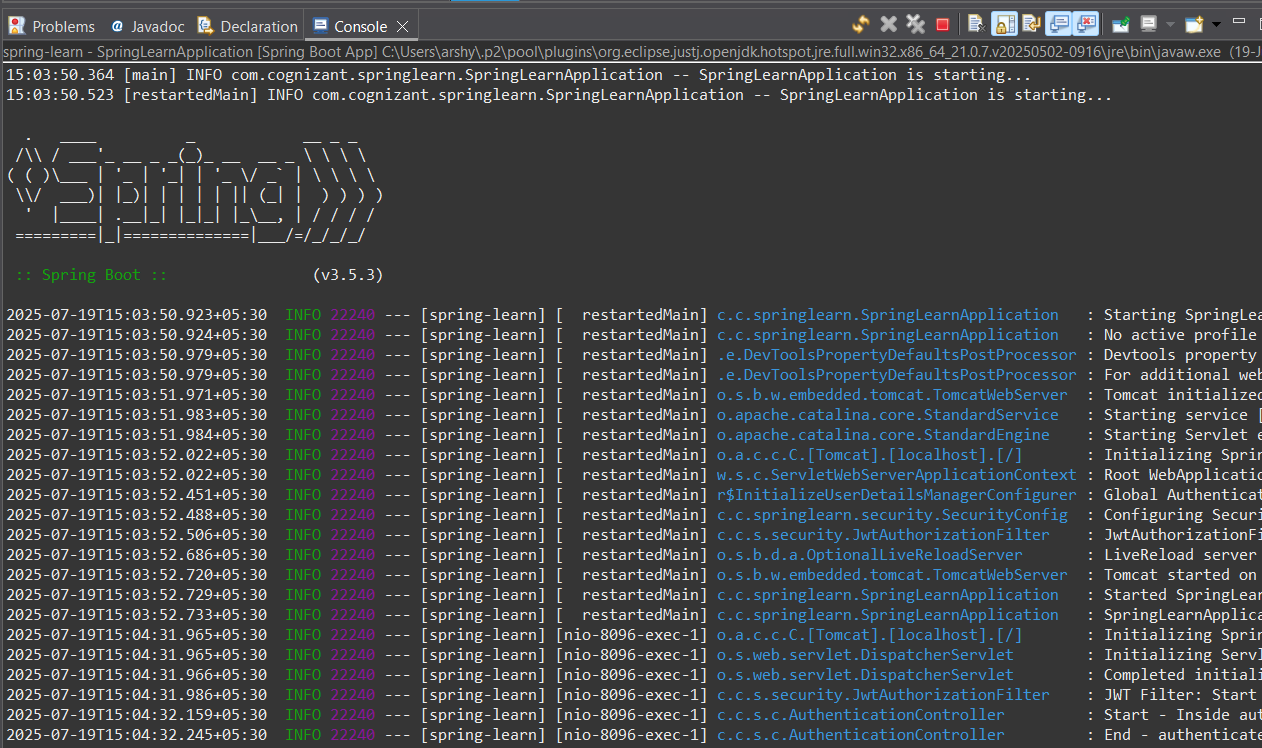
</plugin>

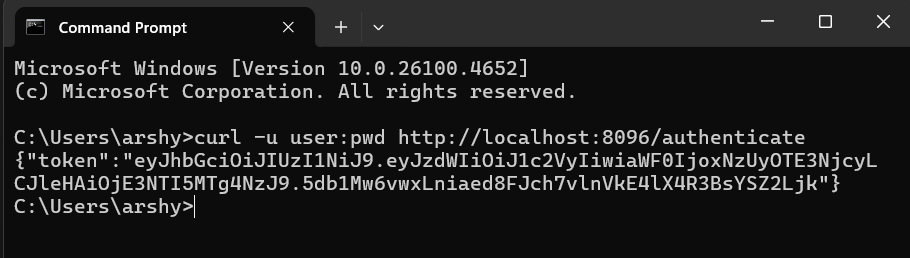
</plugins>

</build>

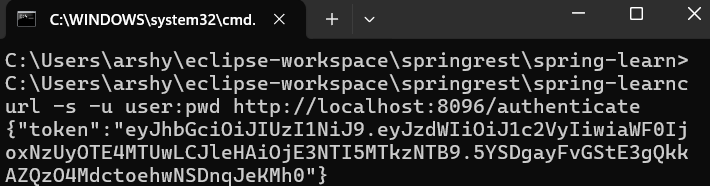
</project>

**Output :**

****

****

**Test**

****