**Securing RESTful Web Services with Spring Security**  
  
Follow steps below to secure all web services using Spring Security:

* Open spring-learn project in Eclipse
* Include spring security related libraries by adding the below dependency in pom.xml

<dependency>

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

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

</dependency>

* Rebuild the project in command line using mvn clean package command (ensure to include proxy details in mvn command).
* To ensure the new libraries are enabled in Eclipse, right click the project and select Maven > Update Project
* Create a new package 'com.cognizant.spring-learn.security'
* Create a new class SecurityConfig in the new package created above which extends from WebSecurityConfigurerAdapter
* Include annotations @Configuration and @EnableWebSecurity at class level
* Import appropriate classes using Ctrl + Shift + O
* Start the application and check the logs and test the REST service. Refer command below:

curl -s http://localhost:8090/countries

* The following error message is the expected response:

{"timestamp":"2019-10-05T09:24:33.794+0000","status":401,"error":"Unauthorized","message":"Unauthorized","path":"/countries"}

* The inclusion of @EnableWebSecurity has restricted access to all the web services with a common password.
* Refer the logs to find out the password generated. Now execute the invocation of the service with password as specified below, which should get the list of countries. include the password from the log file after user:.

curl -s -v -u user:d27321a9-0751-4f59-8fc6-f8633847a9b8 http://localhost:8090/countries

* Find below a sample response for the above command:

[{"code":"US","name":"United States"},{"code":"DE","name":"Germany"},{"code":"IN","name":"India"},{"code":"JP","name":"Japan"}]\* timeout on name lookup is not supported

\*   Trying ::1...

\* TCP\_NODELAY set

\* Connected to localhost (::1) port 8090 (#0)

\* Server auth using Basic with user 'user'

> GET /countries HTTP/1.1

> Host: localhost:8090

> Authorization: Basic dXNlcjpkMjczMjFhOS0wNzUxLTRmNTktOGZjNi1mODYzMzg0N2E5Yjg=

> User-Agent: curl/7.55.0

> Accept: \*/\*

>

< HTTP/1.1 200

< Set-Cookie: JSESSIONID=C0C907417A21BBCA9F30BEEA4B512AEE; Path=/; HttpOnly

< X-Content-Type-Options: nosniff

< X-XSS-Protection: 1; mode=block

< Cache-Control: no-cache, no-store, max-age=0, must-revalidate

< Pragma: no-cache

< Expires: 0

< X-Frame-Options: DENY

< Content-Type: application/json;charset=UTF-8

< Transfer-Encoding: chunked

< Date: Sat, 05 Oct 2019 09:36:34 GMT

<

{ [133 bytes data]

\* Connection #0 to host localhost left intact

* First line contains the country list responded successfully.
* The above result contains the request header and response header.
* The request lines starts with > and reponse lines starts with <
* Notice the Authorization header in the HTTP Request
* This denotes that it uses basic HTTP Authorisation. Whatever following Basic is Base64 encoding of the password that was supplied in the command line.

**Program:**

**CountryController.java**

package com.cognizant.spring\_learn.controller;

import java.util.List;

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

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

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

import com.cognizant.spring\_learn.model.Country;

import com.cognizant.spring\_learn.service.CountryService;

@RestController

public class CountryController {

@Autowired

private CountryService countryService;

@GetMapping("/countries")

public List<Country> getAllCountries() {

return countryService.getAllCountries();

}

}

**CountryService.java**

package com.cognizant.spring\_learn.service;

import java.util.ArrayList;

import java.util.List;

import org.springframework.stereotype.Service;

import com.cognizant.spring\_learn.model.Country;

@Service

public class CountryService {

public List<Country> getAllCountries() {

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

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

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

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

return countryList;

}

}

**Country.java**

package com.cognizant.spring\_learn.model;

public class Country {

private String code;

private String name;

public Country() {}

public Country(String code, String name) {

this.code = code;

this.name = name;

}

public String getCode() {

return code;

}

public void setCode(String code) {

this.code = code;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

**application.properties**

spring.application.name=spring-learn

logging.level.org.springframework=info

logging.level.com.cognizant.spring\_learn=debug

logging.pattern.console=%d{yyMMdd}|%d{HH:mm:ss.SSS}|%-20.20thread|%5p|%-25.25logger{25}|%25M|%m%n

server.port=8086

spring.security.user.name=admin

spring.security.user.password=admin123

**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>1.0.0</version>

<packaging>jar</packaging>

<parent>

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

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

<version>3.1.5</version>

<relativePath/> <!-- look up parent from Maven Central -->

</parent>

<properties>

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

</properties>

<dependencies>

<!-- Web for REST APIs -->

<dependency>

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

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

</dependency>

<!-- Logging (already included in web starter, just explicit) -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

<dependency>

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

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

</dependency>

<!-- Optional: Spring DevTools for hot reload (remove if error continues) -->

<!--

<dependency>

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

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

<scope>runtime</scope>

</dependency>

-->

</dependencies>

<build>

<plugins>

<!-- Use Spring Boot Maven Plugin -->

<plugin>

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

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

<version>3.1.5</version>

</plugin>

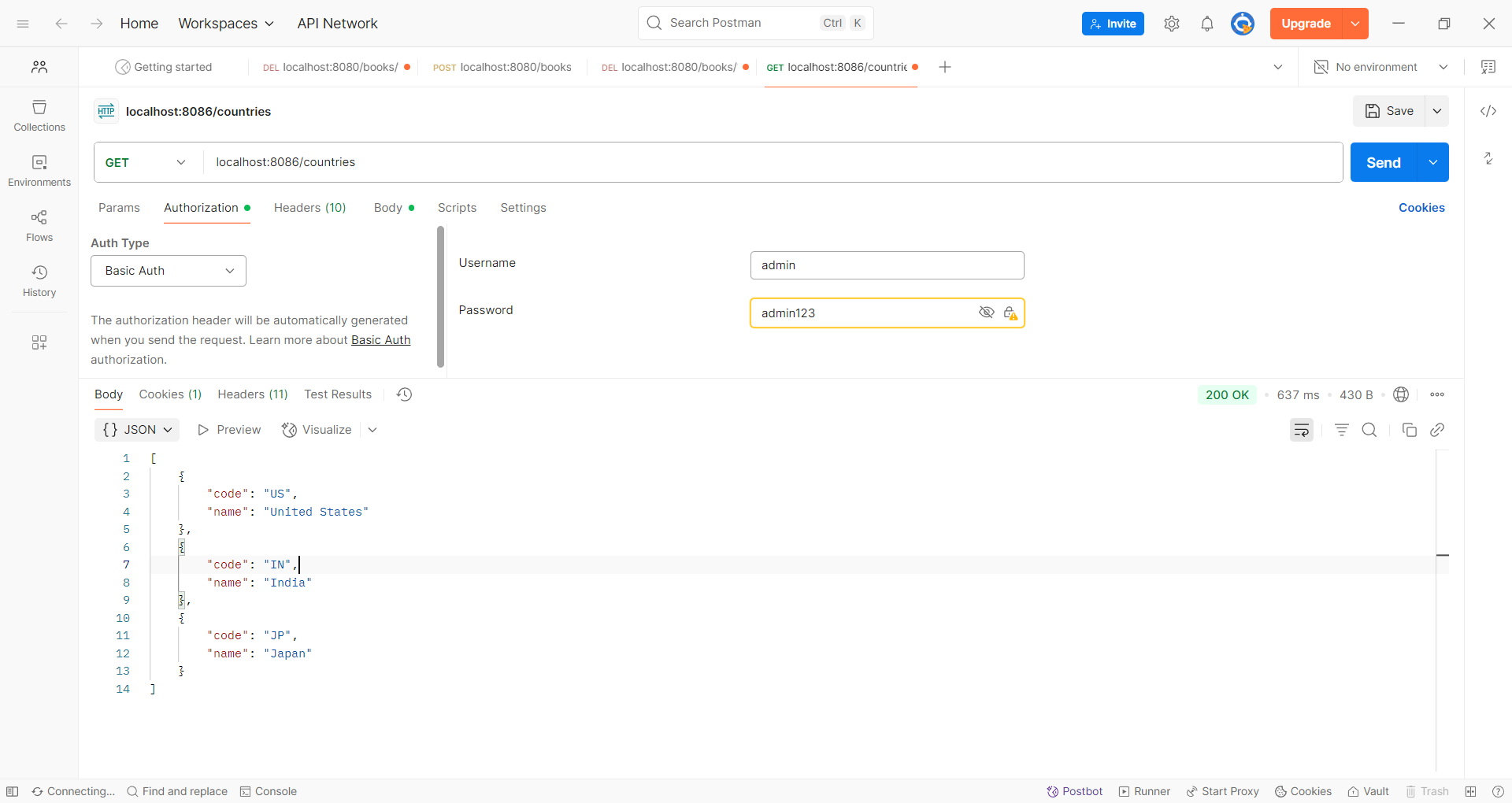
</plugins>

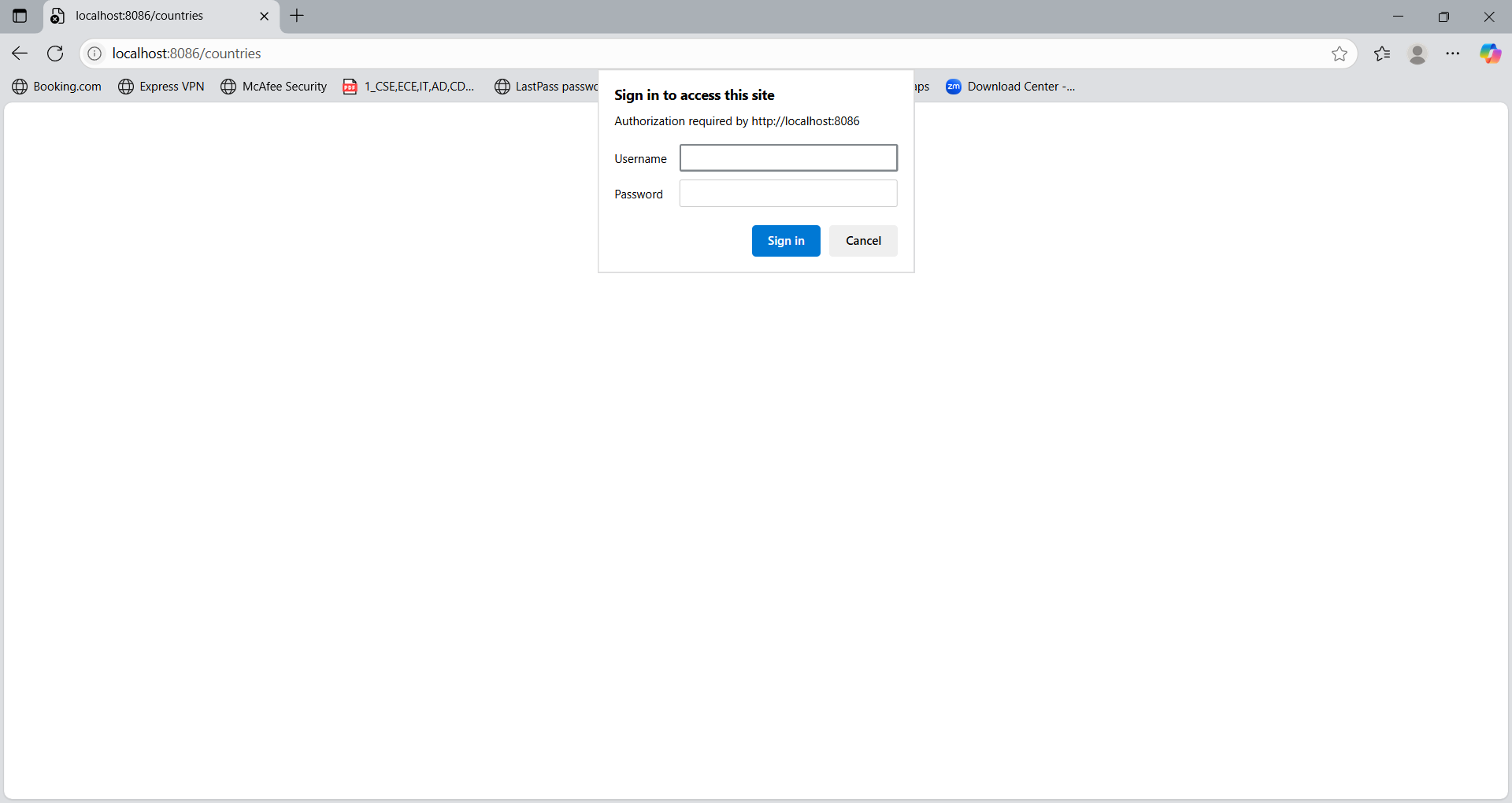
</build>

</project>

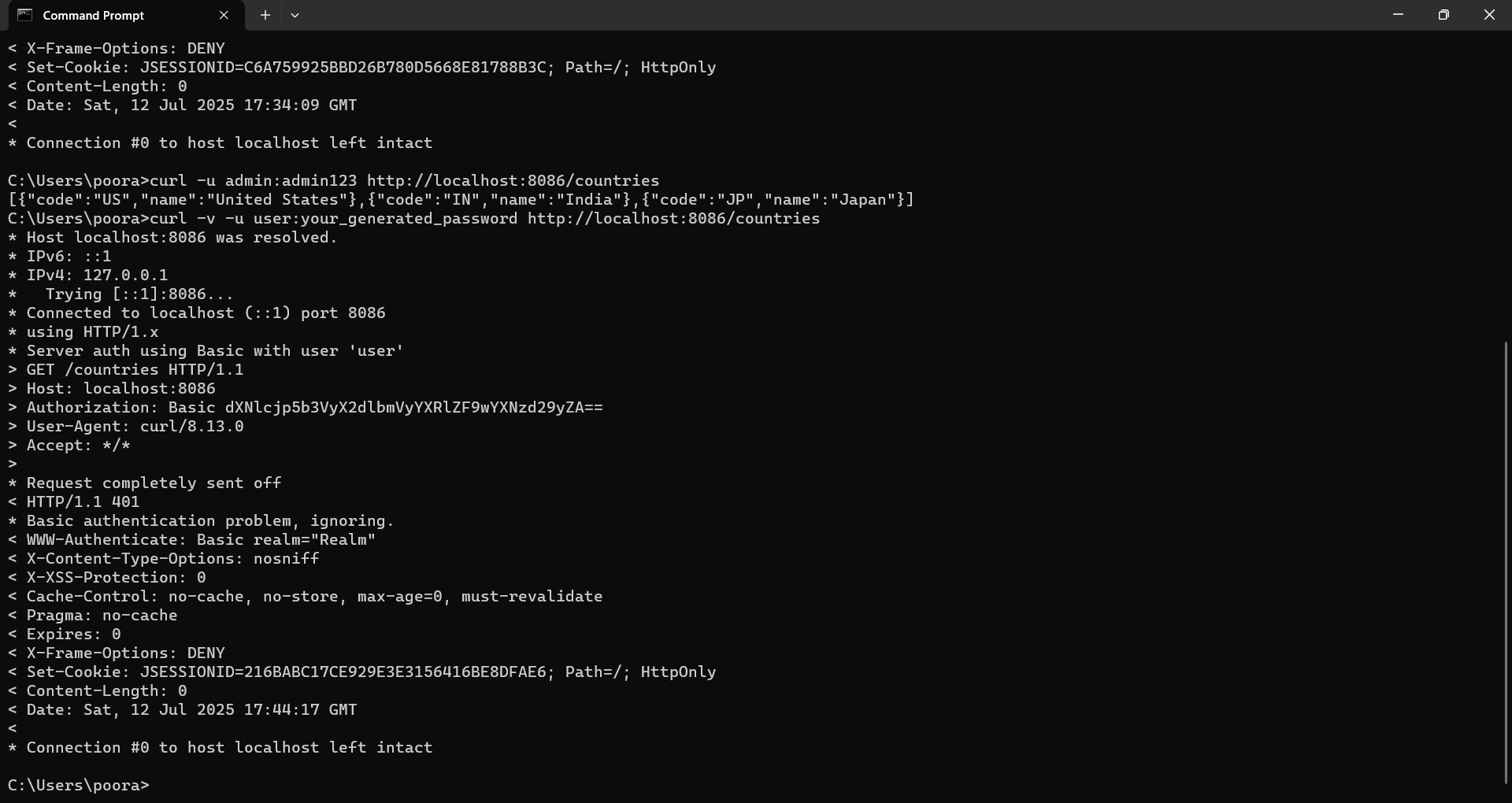
**Output:**

**With Authentication**





**In Verbose mode**



**Create authentication service that returns JWT   
Program:**

**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.example.jwt</groupId>

<artifactId>jwt-auth-service</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>jwt-auth-service</name>

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

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

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

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<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> <!-- or jjwt-gson if preferred -->

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

<dependency>

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

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

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<configuration>

<annotationProcessorPaths>

<path>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</path>

</annotationProcessorPaths>

</configuration>

</plugin>

<plugin>

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

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

<configuration>

<excludes>

<exclude>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</exclude>

</excludes>

</configuration>

</plugin>

</plugins>

</build>

</project**>**

**AuthController.java**

package com.example.jwt.jwt\_auth\_service.controller;

import com.example.jwt.jwt\_auth\_service.service.JwtUtil;

import org.springframework.http.ResponseEntity;

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

import jakarta.servlet.http.HttpServletRequest;

import java.util.Base64;

@RestController

public class AuthController {

private final JwtUtil jwtUtil;

public AuthController(JwtUtil jwtUtil) {

this.jwtUtil = jwtUtil;

}

@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];

// Simple in-memory check

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.example.jwt.jwt\_auth\_service.security;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.Customizer;

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

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

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

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

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

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.security.web.SecurityFilterChain;

@Configuration

public class SecurityConfig {

// Security filter chain for HTTP requests

@Bean

public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

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

.authorizeHttpRequests(auth -> auth

.requestMatchers("/authenticate").permitAll() // allow unauthenticated access

.anyRequest().authenticated()

)

.httpBasic(Customizer.*withDefaults*()); // enable Basic Auth

return http.build();

}

// In-memory user details

@Bean

public UserDetailsService userDetailsService() {

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

.password("pwd")

.roles("USER")

.build();

return new InMemoryUserDetailsManager(user);

}

// Password encoder

@Bean

public PasswordEncoder passwordEncoder() {

return NoOpPasswordEncoder.getInstance(); // Plaintext password (for demo only)

}

}

**JwtUtil.java**

package com.example.jwt.jwt\_auth\_service.service;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.security.Keys;

import io.jsonwebtoken.SignatureAlgorithm;

import org.springframework.stereotype.Service;

import java.util.Date;

import java.security.Key;

@Service

public class JwtUtil {

private final Key SECRET\_KEY = Keys.*hmacShaKeyFor*("mysecretkey123456789012345678901234".getBytes()); // 32+ chars

public String generateToken(String username) {

return Jwts.*builder*()

.setSubject(username)

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

.setExpiration(new Date(System.*currentTimeMillis*() + 1000 \* 60 \* 10)) // 10 mins

.signWith(SECRET\_KEY, SignatureAlgorithm.*HS256*)

.compact();

}

}

**JwtAuthServiceApplication.java**

package com.example.jwt.jwt\_auth\_service;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class JwtAuthServiceApplication {

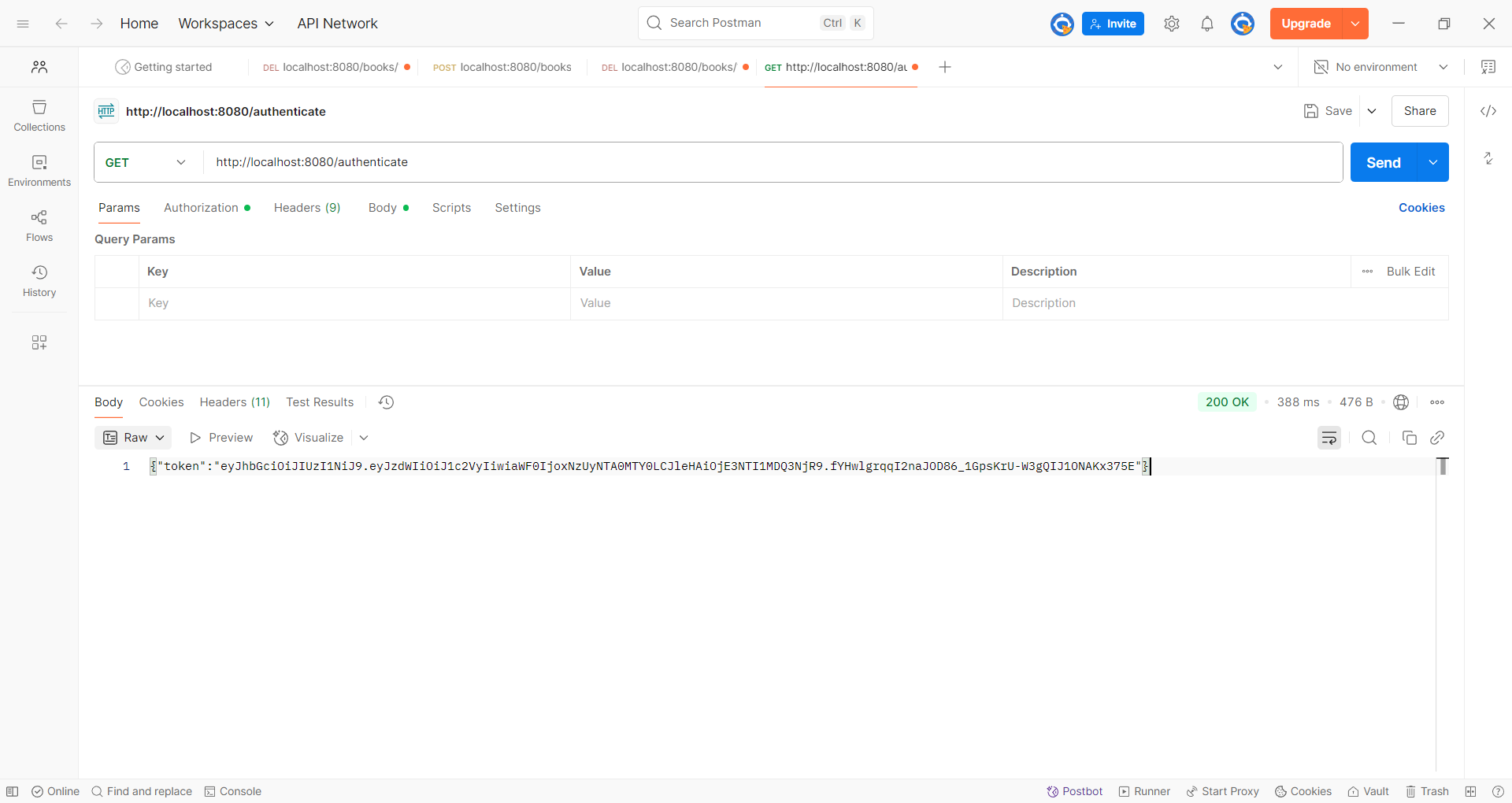
public static void main(String[] args) {

SpringApplication.*run*(JwtAuthServiceApplication.class, args);

}

}

**Output:**

****