**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 Boot Starter for Web -->

<dependency>

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

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

</dependency>

<!-- Spring Boot DevTools for live reload -->

<dependency>

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

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

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<!-- Spring Boot Starter Test (JUnit + MockMvc) -->

<dependency>

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

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

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-tx</artifactId>

</dependency>

<dependency>

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

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

</dependency>

</dependencies>

<build>

<plugins>

<!-- Spring Boot Maven Plugin -->

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

**SecurityConfig.java**

package com.cognizant.spring\_learn.security;

import org.springframework.context.annotation.Configuration;

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

import org.springframework.security.web.SecurityFilterChain;

import org.springframework.context.annotation.Bean;

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

*@Configuration*

*@EnableWebSecurity*

public class SecurityConfig {

*@Bean*

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

.authorizeHttpRequests(authz -> authz

.anyRequest().authenticated()

)

.httpBasic()

.and()

.csrf().disable();

return http.build();

}

}

**SpringLearnApplication.java**

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

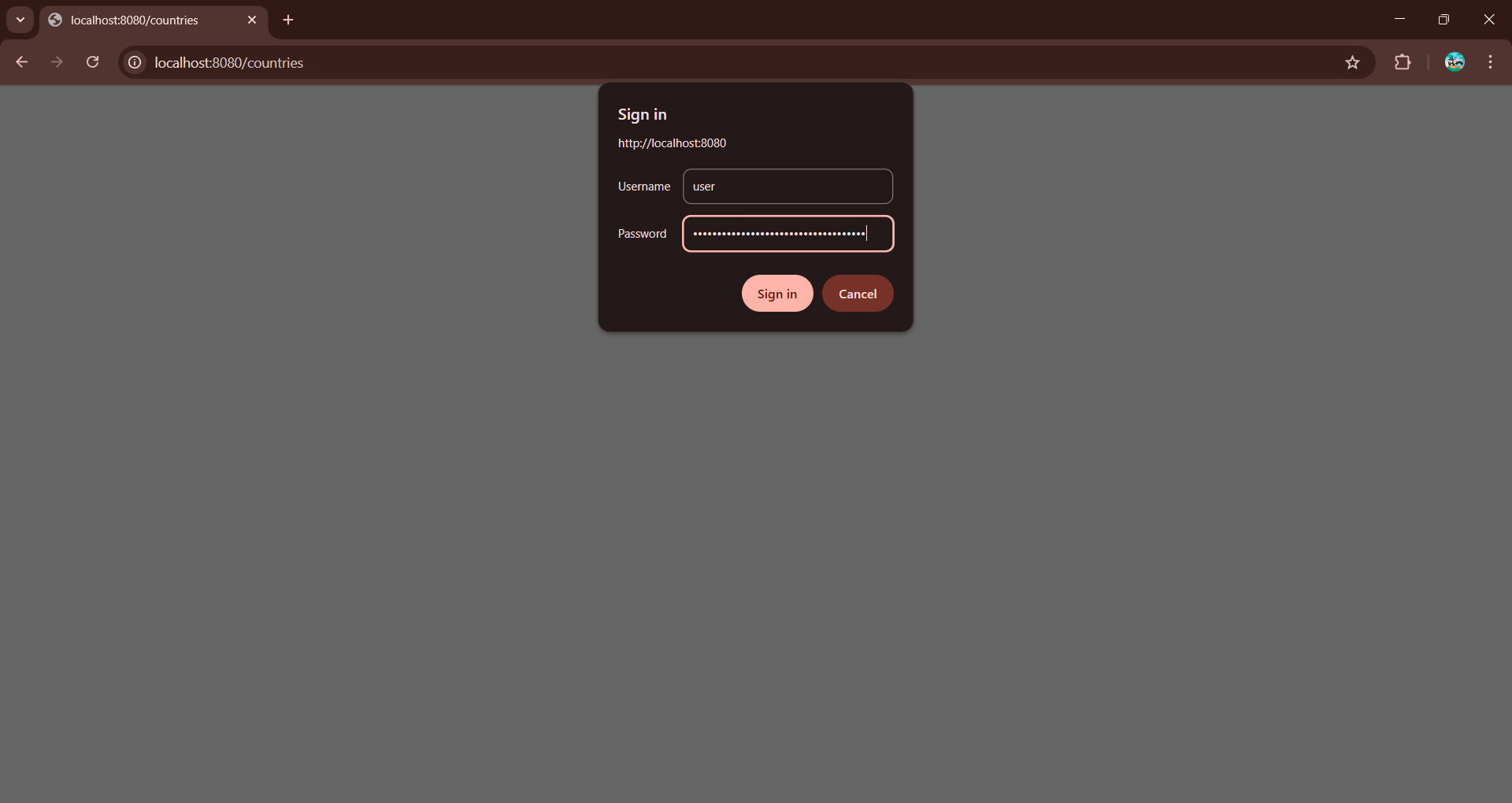
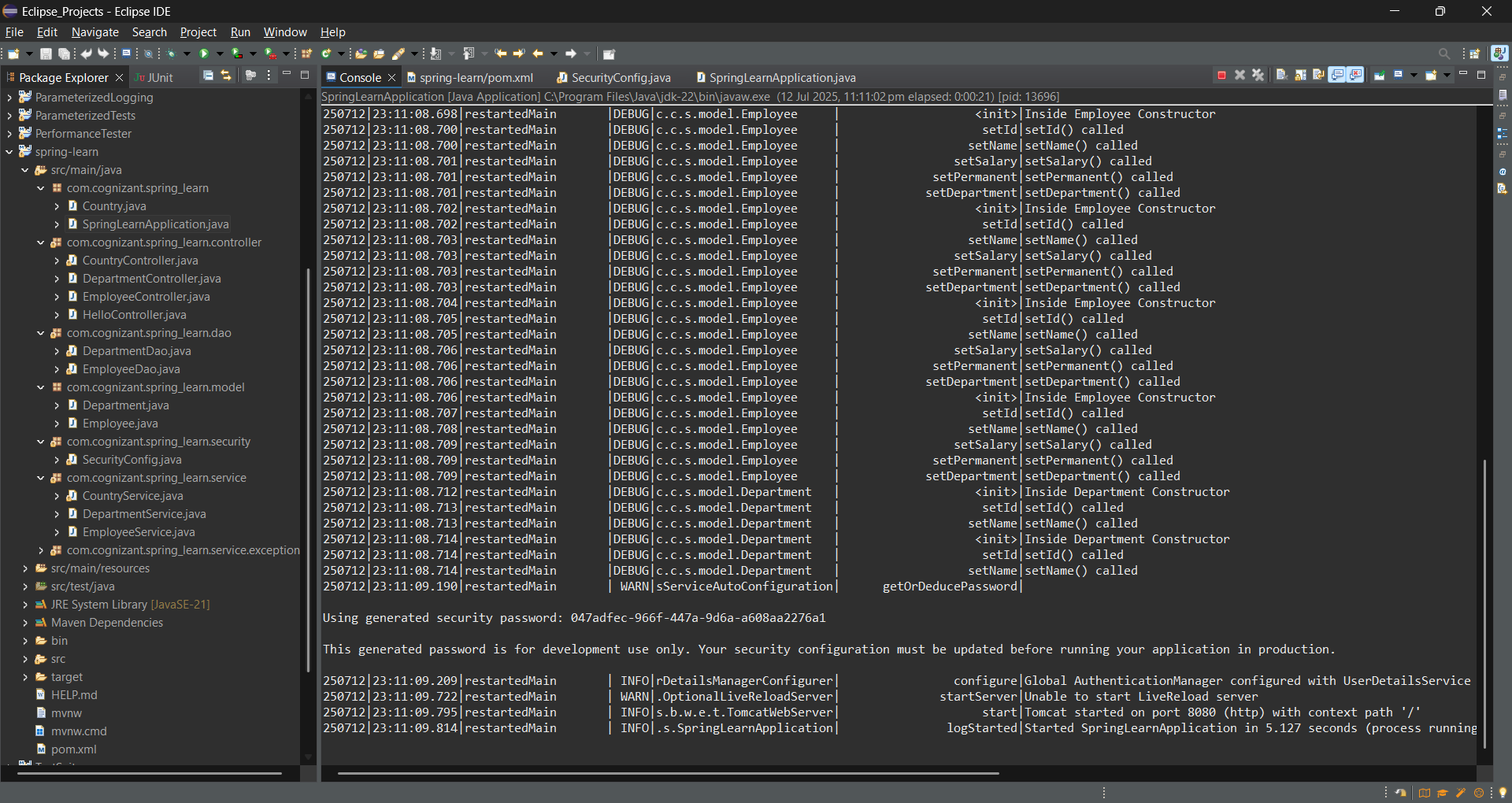
public static void main(String[] args) {

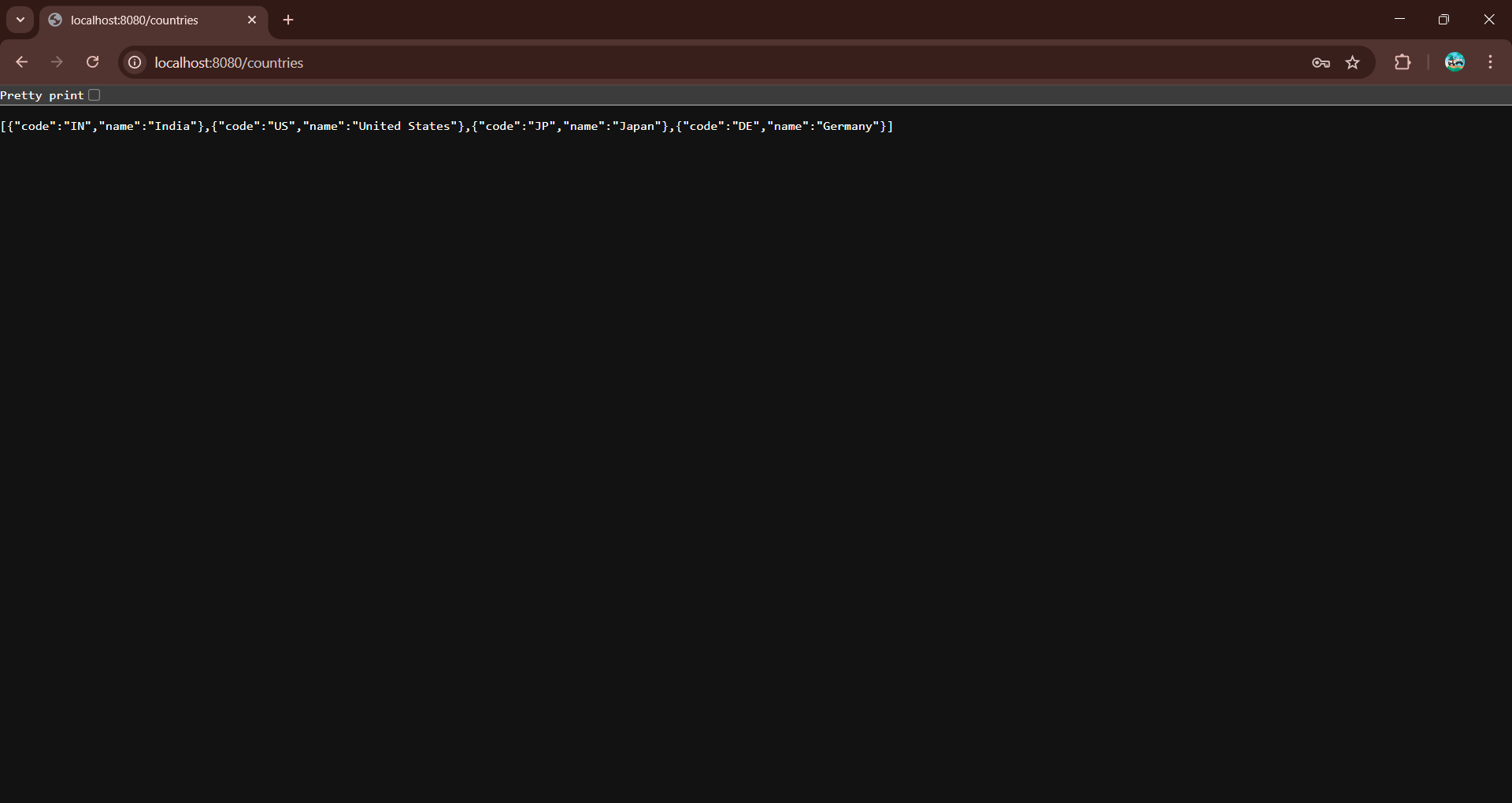
SpringApplication.run(SpringLearnApplication.class, args);

}

}

**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>2.7.18</version>

<relativePath/>

</parent>

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<properties>

<java.version>11</java.version> <!-- Use Java 11 for compatibility -->

</properties>

<dependencies>

<!-- Spring Boot Web -->

<dependency>

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

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

</dependency>

<!-- Spring Boot DevTools -->

<dependency>

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

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

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<!-- Spring Security -->

<dependency>

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

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

</dependency>

<!-- Transaction Support -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-tx</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>

**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.config.annotation.authentication.builders.AuthenticationManagerBuilder;

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

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

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

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

*@Configuration*

*@EnableWebSecurity*

public class SecurityConfig extends WebSecurityConfigurerAdapter {

*@Override*

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.inMemoryAuthentication()

.withUser("admin").password(passwordEncoder().encode("pwd")).roles("ADMIN")

.and()

.withUser("user").password(passwordEncoder().encode("pwd")).roles("USER");

}

*@Bean*

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

*@Override*

protected void configure(HttpSecurity httpSecurity) throws Exception {

httpSecurity.csrf().disable().httpBasic().and()

.authorizeRequests()

.antMatchers("/countries").hasRole("USER")

.anyRequest().authenticated();

}

}

SpringLearnApplication.java

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@SpringBootApplication*

public class SpringLearnApplication {

public static void main(String[] args) {

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

}

}

**CountryController.java**

package com.cognizant.spring\_learn.controller;

import java.util.List;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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

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

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

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

import com.cognizant.spring\_learn.Country;

import com.cognizant.spring\_learn.dao.CountryDao;

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

import com.cognizant.spring\_learn.service.exception.CountryNotFoundException;

*@RestController*

public class CountryController {

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

*@Autowired*

private CountryService countryService;

*@Autowired*

private CountryDao countryDao;

*@GetMapping*("/countries")

public List<Country> getAllCountries() {

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

List<Country> countries = countryDao.getAllCountries(); // loaded from XML

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

return countries;

}

*@GetMapping*("/countries/{code}")

public Country getCountry(*@PathVariable* String code) throws CountryNotFoundException {

***LOGGER***.info("Start getCountry: code={}", code);

Country country = countryService.getCountry(code);

***LOGGER***.info("End getCountry");

return country;

}

}

**CountryDao.java**

package com.cognizant.spring\_learn.dao;

import com.cognizant.spring\_learn.Country;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.stereotype.Repository;

import java.util.List;

*@Repository*

public class CountryDao {

private static List<Country> *countryList*;

static {

ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");

*countryList* = (List<Country>) context.getBean("countryList");

}

public List<Country> getAllCountries() {

return *countryList*;

}

public Country getCountry(String code) {

return *countryList*.stream()

.filter(c -> c.getCode().equalsIgnoreCase(code))

.findFirst()

.orElse(null);

}

}

**Country.xml**

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

<beans xmlns="http://www.springframework.org/schema/beans"

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

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="in" class="com.cognizant.spring\_learn.Country">

<property name="code" value="IN"/>

<property name="name" value="India"/>

</bean>

<bean id="us" class="com.cognizant.spring\_learn.Country">

<property name="code" value="US"/>

<property name="name" value="United States"/>

</bean>

<bean id="jp" class="com.cognizant.spring\_learn.Country">

<property name="code" value="JP"/>

<property name="name" value="Japan"/>

</bean>

<bean id="de" class="com.cognizant.spring\_learn.Country">

<property name="code" value="DE"/>

<property name="name" value="Germany"/>

</bean>

<bean id="countryList" class="java.util.ArrayList">

<constructor-arg>

<list>

<ref bean="in"/>

<ref bean="us"/>

<ref bean="jp"/>

<ref bean="de"/>

</list>

</constructor-arg>

</bean>

</beans>

**Country.java**

package com.cognizant.spring\_learn;

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;

}

}

**CountryService.java**

package com.cognizant.spring\_learn.service;

import com.cognizant.spring\_learn.Country;

import com.cognizant.spring\_learn.dao.CountryDao;

import com.cognizant.spring\_learn.service.exception.CountryNotFoundException;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import java.util.List;

*@Service*

public class CountryService {

*@Autowired*

private CountryDao countryDao;

*@Transactional*

public List<Country> getAllCountries() {

return countryDao.getAllCountries();

}

*@Transactional*

public Country getCountry(String code) throws CountryNotFoundException {

Country country = countryDao.getCountry(code);

if (country == null) {

throw new CountryNotFoundException("Country not found");

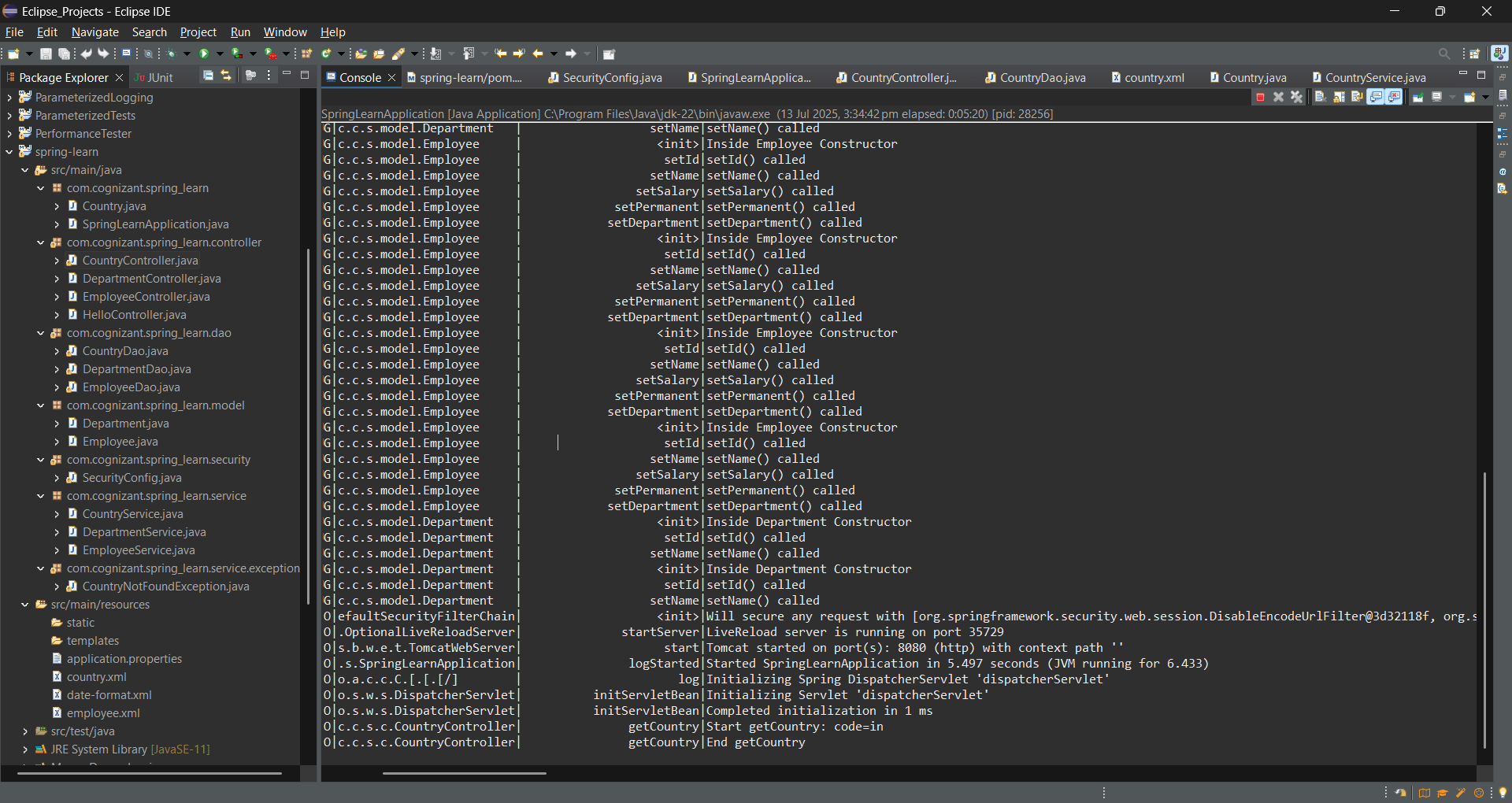
}

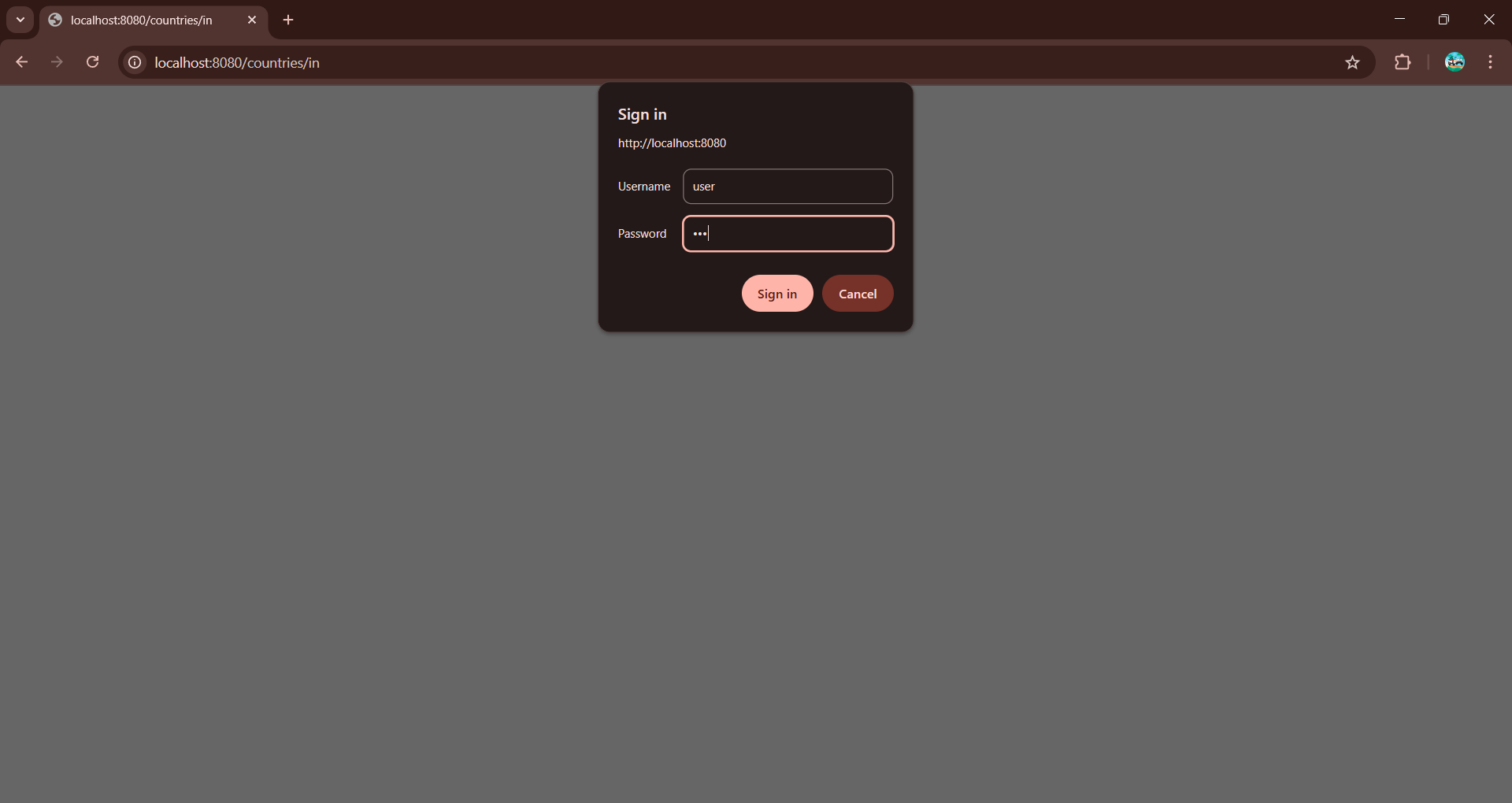
return country;

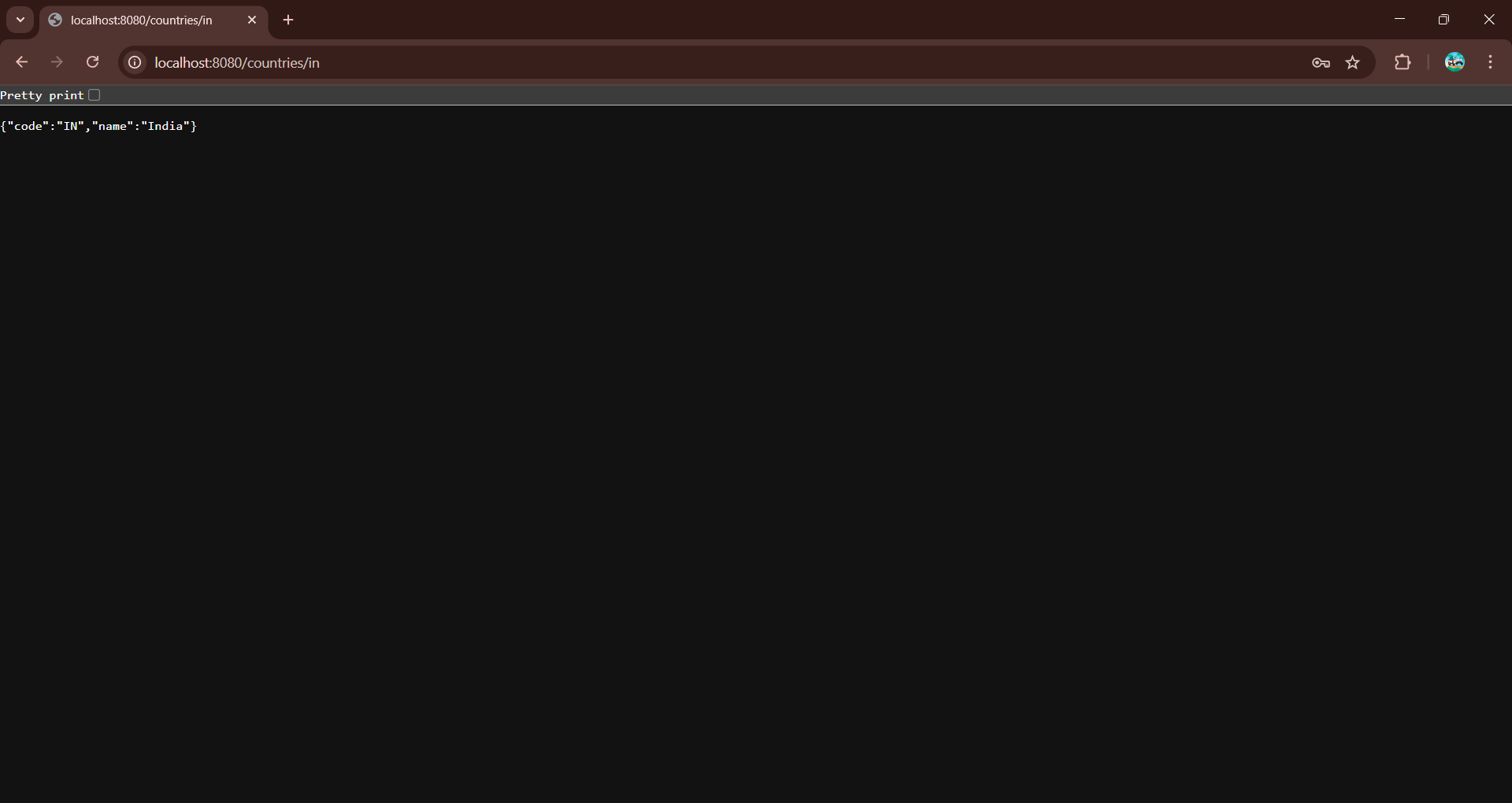
}

}

**Output:**







**Understanding JWT**   
  
**What is JWT?**

* JWT stands for JSON Web Token
* Internet standard ([**IETF 7519 Link**](https://tools.ietf.org/html/rfc7519)) for creating [JSON](https://en.wikipedia.org/wiki/JSON)-based [access tokens](https://en.wikipedia.org/wiki/Access_token)
* JWT can be typically used to pass identity of authenticated users and [service provider](https://en.wikipedia.org/wiki/Service_provider),

**JWT Process Flow (**[**diagram**](https://miro.medium.com/max/1600/0*13yKRyewaI1sLFSz.)**link)**

* Clien t sends username and password to server
* Servers validates credentials, creates token (JWT) and reponds it back
* Client attaches the token in the subsequent requests to server
* Server validates the token (JWT) on each client request

**Structure of JSON Web Token**

* Reference: https://en.wikipedia.org/wiki/JSON\_Web\_Token#Structure
* Header: Contains the encryption algorithm
* Payload: Contains application specific data. Usually this contains the user id and role.
* Signature: Computed based on the formula defined using header and payload

**Exercise to check how JWT token is created**

* Open link https://en.wikipedia.org/wiki/JSON\_Web\_Token#Structure in browser
* Open link https://jwt.io/ in another browser tab and scroll down to the Encoded, Decoded section
* Copy and paste the header content from wikipedia article and paste it in header section of https://jwt.io
* Copy and paste the payload content from wikipedia article and paste it in payload section of https://jwt.io
* Type "secretkey" in the textbox within Verify Signature section
* Check if the token generated in the Encoded section of https://jwt.io matches with the generated token displayed in the Structure section of wikipedia article

**Create authentication service that returns JWT** 

**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>2.7.18</version>

<relativePath/>

</parent>

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-learn</name>

<description>Spring Learn Project with JWT</description>

<properties>

<java.version>11</java.version>

</properties>

<dependencies>

<!-- Spring Boot Web -->

<dependency>

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

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

</dependency>

<!-- Spring Boot Security -->

<dependency>

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

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

</dependency>

<!-- Spring Boot DevTools -->

<dependency>

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

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

<scope>runtime</scope>

<optional>true</optional>

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

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

<!-- Spring TX -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-tx</artifactId>

</dependency>

<!-- JAXB (Optional, used only if needed by other libs, safe to keep) -->

<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>

<!-- 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>

**SpringLearnApplication.java**

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@SpringBootApplication*

public class SpringLearnApplication {

public static void main(String[] args) {

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

}

}

**AuthenticationController.java**

package com.cognizant.spring\_learn.controller;

import java.util.Base64;

import javax.servlet.http.HttpServletRequest;

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

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

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

import com.cognizant.spring\_learn.util.JwtUtil;

*@RestController*

public class AuthenticationController {

*@Autowired*

private JwtUtil jwtUtil;

*@RequestMapping*("/authenticate")

public String generateToken(HttpServletRequest request) {

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

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

throw new RuntimeException("Missing or invalid Authorization header");

}

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

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

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

String username = credentials[0];

String password = credentials[1];

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

return "{\"token\":\"" + jwtUtil.generateToken(username) + "\"}";

} else {

throw new RuntimeException("Invalid credentials");

}

}

}

**SecurityConfig.java**

package com.cognizant.spring\_learn.security;

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

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.config.annotation.web.configuration.EnableWebSecurity;

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

import org.springframework.security.web.SecurityFilterChain;

import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

import com.cognizant.spring\_learn.filter.JwtRequestFilter;

*@Configuration*

*@EnableWebSecurity*

public class SecurityConfig {

*@Autowired*

private JwtRequestFilter jwtRequestFilter;

*@Bean*

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http.csrf().disable()

.authorizeRequests()

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

.anyRequest().authenticated()

.and()

.sessionManagement().sessionCreationPolicy(*SessionCreationPolicy*.*STATELESS*);

http.addFilterBefore(jwtRequestFilter, UsernamePasswordAuthenticationFilter.class);

return http.build();

}

*@Bean*

public AuthenticationManager authenticationManager(AuthenticationConfiguration configuration) throws Exception {

return configuration.getAuthenticationManager();

}

}

**JwtUtil.java**

package com.cognizant.spring\_learn.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

import java.security.Key;

import java.util.Date;

import org.springframework.stereotype.Component;

*@Component*

public class JwtUtil {

private static final Key ***key*** = Keys.*secretKeyFor*(*SignatureAlgorithm*.***HS256***);

private static final long ***EXPIRATION\_TIME*** = 10 \* 60 \* 1000;

public String generateToken(String username) {

return Jwts.*builder*()

.setSubject(username)

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

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

.signWith(***key***)

.compact();

}

public String extractUsername(String token) {

return Jwts.*parserBuilder*()

.setSigningKey(***key***)

.build()

.parseClaimsJws(token)

.getBody()

.getSubject();

}

public boolean validateToken(String token, String username) {

String extractedUsername = extractUsername(token);

return (username.equals(extractedUsername) && !isTokenExpired(token));

}

private boolean isTokenExpired(String token) {

Date expiration = Jwts.*parserBuilder*()

.setSigningKey(***key***)

.build()

.parseClaimsJws(token)

.getBody()

.getExpiration();

return expiration.before(new Date());

}

}

**JwtRequestFilter.java**

package com.cognizant.spring\_learn.filter;

import java.io.IOException;

import javax.servlet.FilterChain;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

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

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;

import org.springframework.stereotype.Component;

import org.springframework.web.filter.OncePerRequestFilter;

import com.cognizant.spring\_learn.util.JwtUtil;

*@Component*

public class JwtRequestFilter extends OncePerRequestFilter {

*@Autowired*

private JwtUtil jwtUtil;

*@Override*

protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain chain)

throws ServletException, IOException {

final String authorizationHeader = request.getHeader("Authorization");

String username = null;

String jwt = null;

if (authorizationHeader != null && authorizationHeader.startsWith("Bearer ")) {

jwt = authorizationHeader.substring(7);

username = jwtUtil.extractUsername(jwt);

}

if (username != null && SecurityContextHolder.*getContext*().getAuthentication() == null) {

if (jwtUtil.validateToken(jwt, username)) {

UsernamePasswordAuthenticationToken authToken =

new UsernamePasswordAuthenticationToken(username, null, new java.util.ArrayList<>());

authToken.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));

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

}

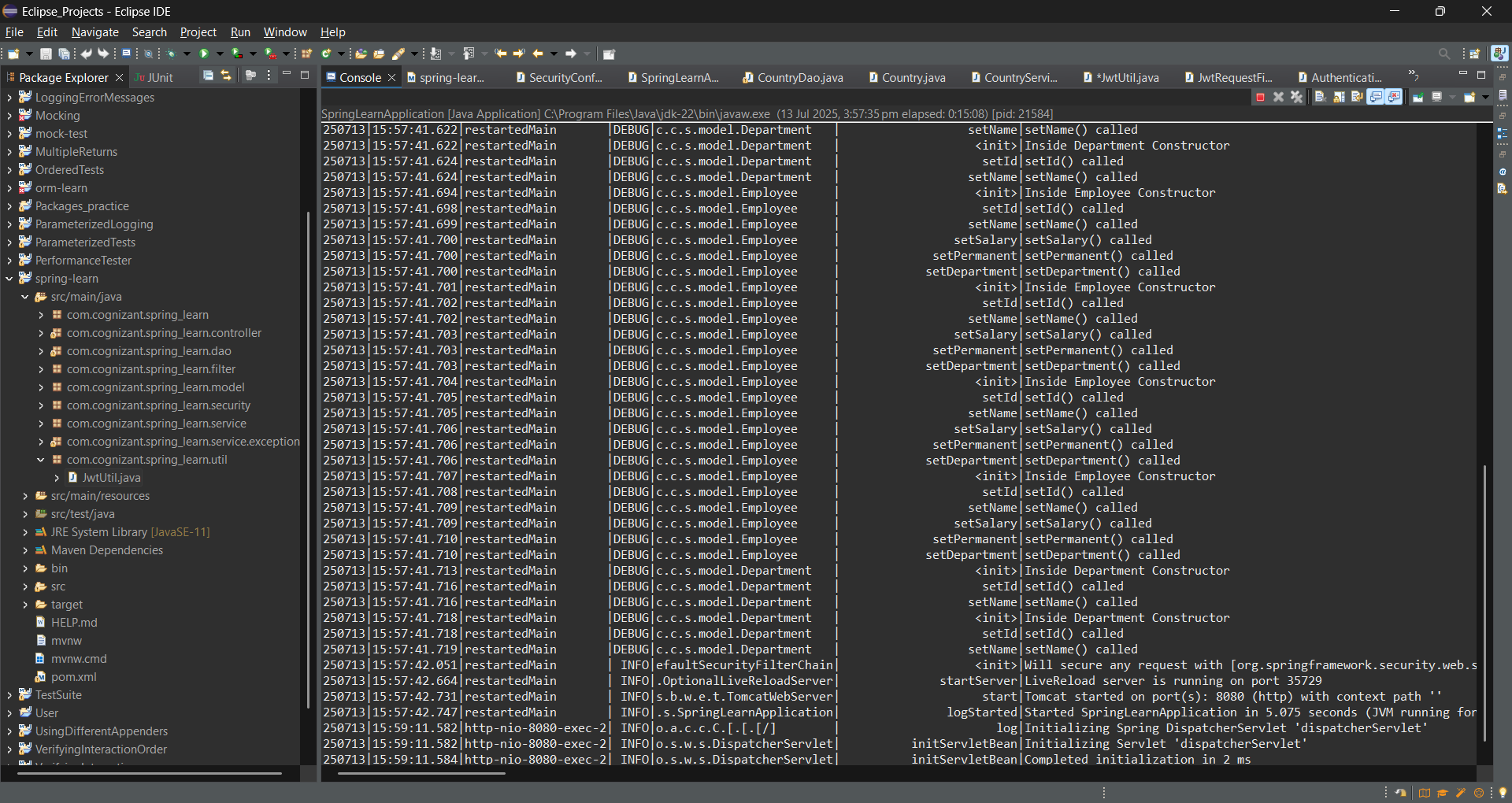
}

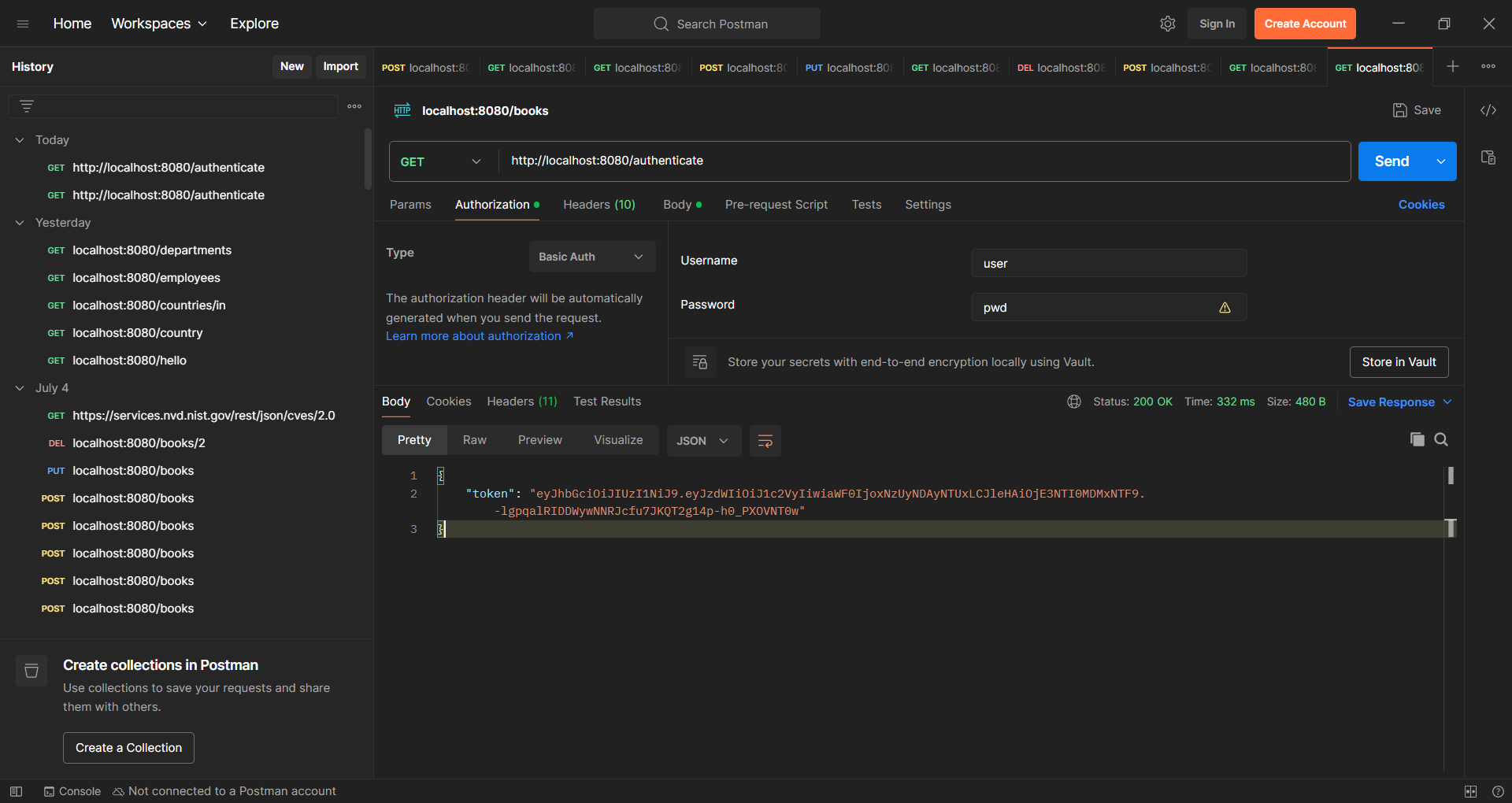
chain.doFilter(request, response);

}

}

**Output:**

****

****