SuperSet ID : 6421064

**Securing RESTful Web Services with Spring Security**

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.config.annotation.web.configuration.EnableWebSecurity;

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

@Configuration

@EnableWebSecurity

**public** **class** SecurityConfig **extends** ~~WebSecurityConfigurerAdapter~~ {

@Override

**protected** **void** configure(HttpSecurity http) **throws** Exception {

http

.authorizeRequests()

.antMatchers("/countries").permitAll()

.anyRequest().authenticated()

.and()

.httpBasic();

}

}

Output:

[moon\_nika@archlinux ~]$ curl -v -u user:cbf5d05f-fbd3-4fd5-8e31-f7e253a7ac6e http://localhost:8081/country

\* Host localhost:8081 was resolved.

\* IPv6: ::1

\* IPv4: 127.0.0.1

\* Trying [::1]:8081...

\* Connected to localhost (::1) port 8081

\* using HTTP/1.x

\* Server auth using Basic with user 'user'

> GET /country HTTP/1.1

> Host: localhost:8081

> Authorization: Basic dXNlcjpjYmY1ZDA1Zi1mYmQzLTRmZDUtOGUzMS1mN2UyNTNhN2FjNmU=

> User-Agent: curl/8.15.0

> Accept: \*/\*

>

\* Request completely sent off

< HTTP/1.1 200

< Set-Cookie: JSESSIONID=322249A84F65D727623151930918E7BC; 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

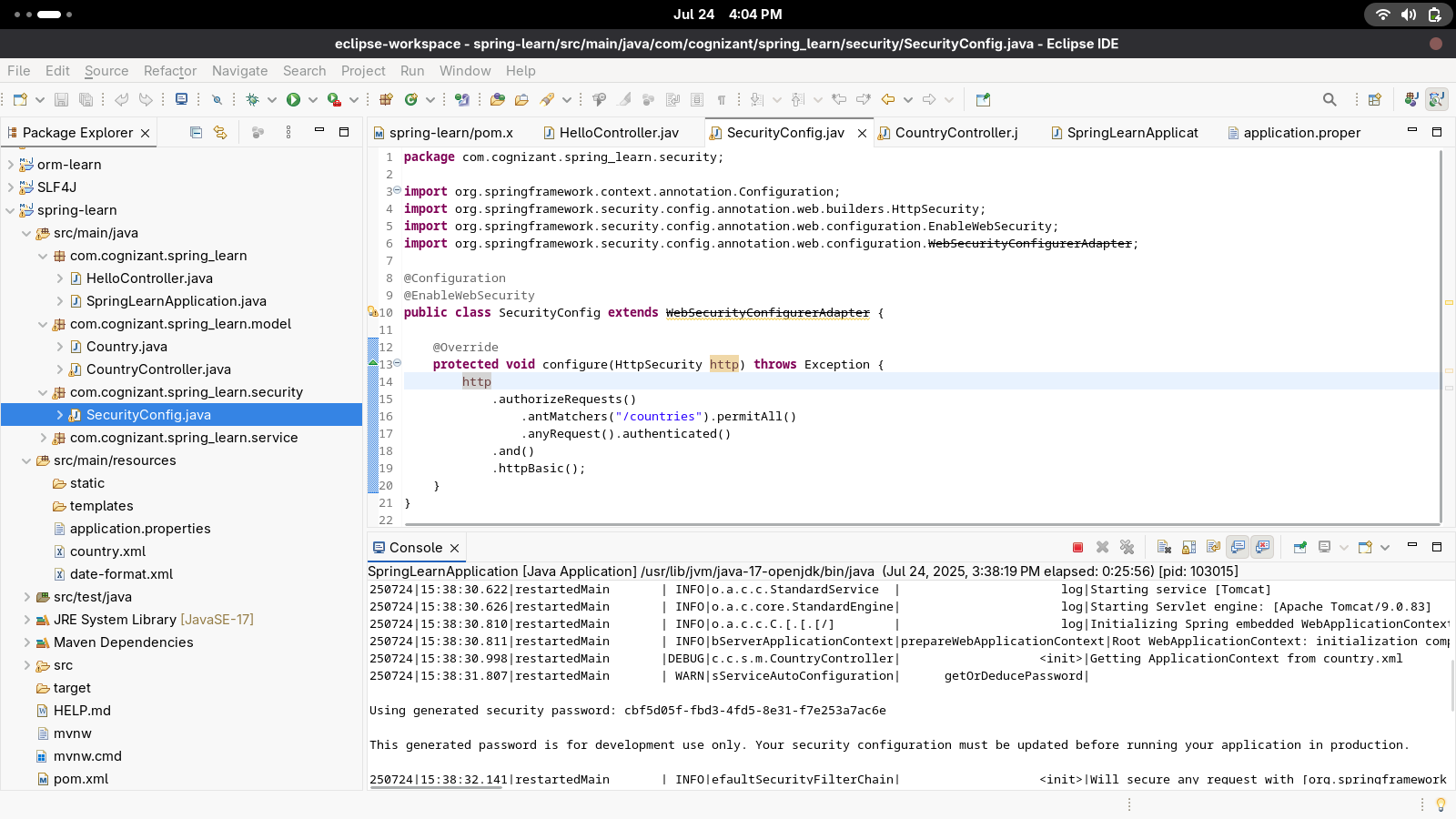
< Transfer-Encoding: chunked

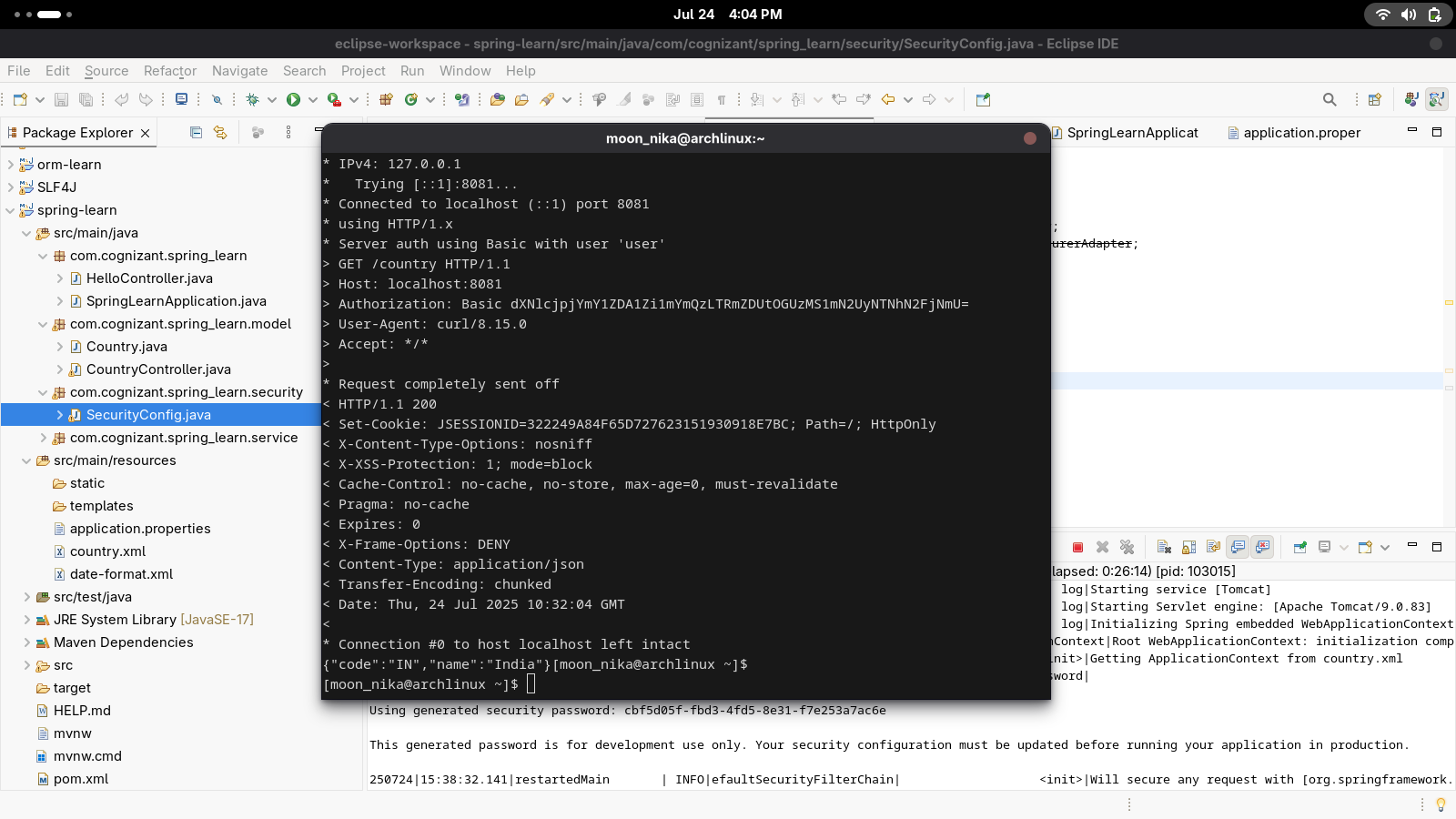
< Date: Thu, 24 Jul 2025 10:32:04 GMT

<

\* Connection #0 to host localhost left intact

{"code":"IN","name":"India"}





**Creating users and roles in Spring Security**

Securityconfig.java

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

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**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.authentication.builders.AuthenticationManagerBuilder;

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

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

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

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

@Configuration

@EnableWebSecurity

**public** **class** SecurityConfig **extends** ~~WebSecurityConfigurerAdapter~~ {

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

@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() {

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

**return** **new** BCryptPasswordEncoder();

}

@Override

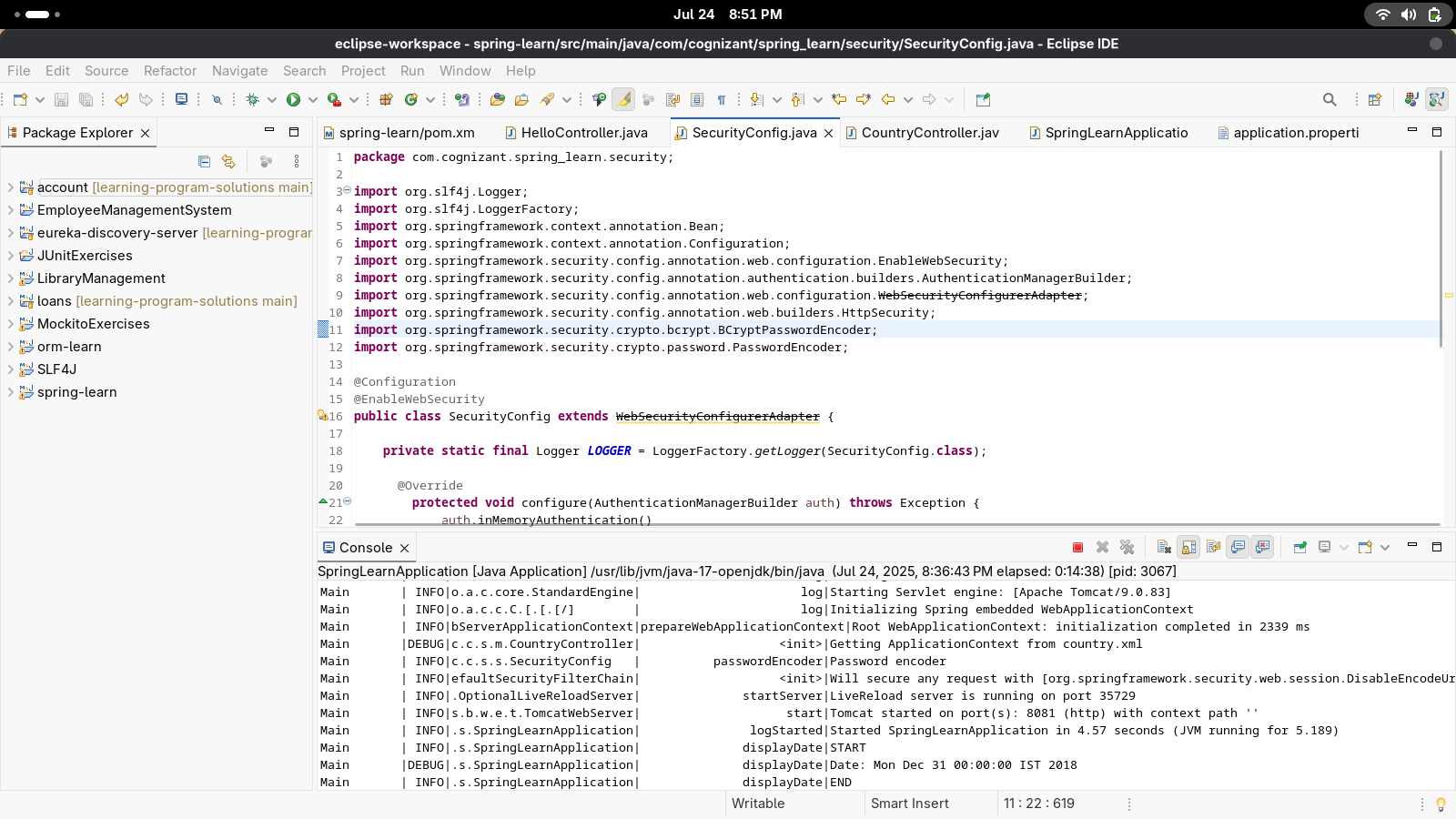
**protected** **void** configure(HttpSecurity httpSecurity) **throws** Exception {

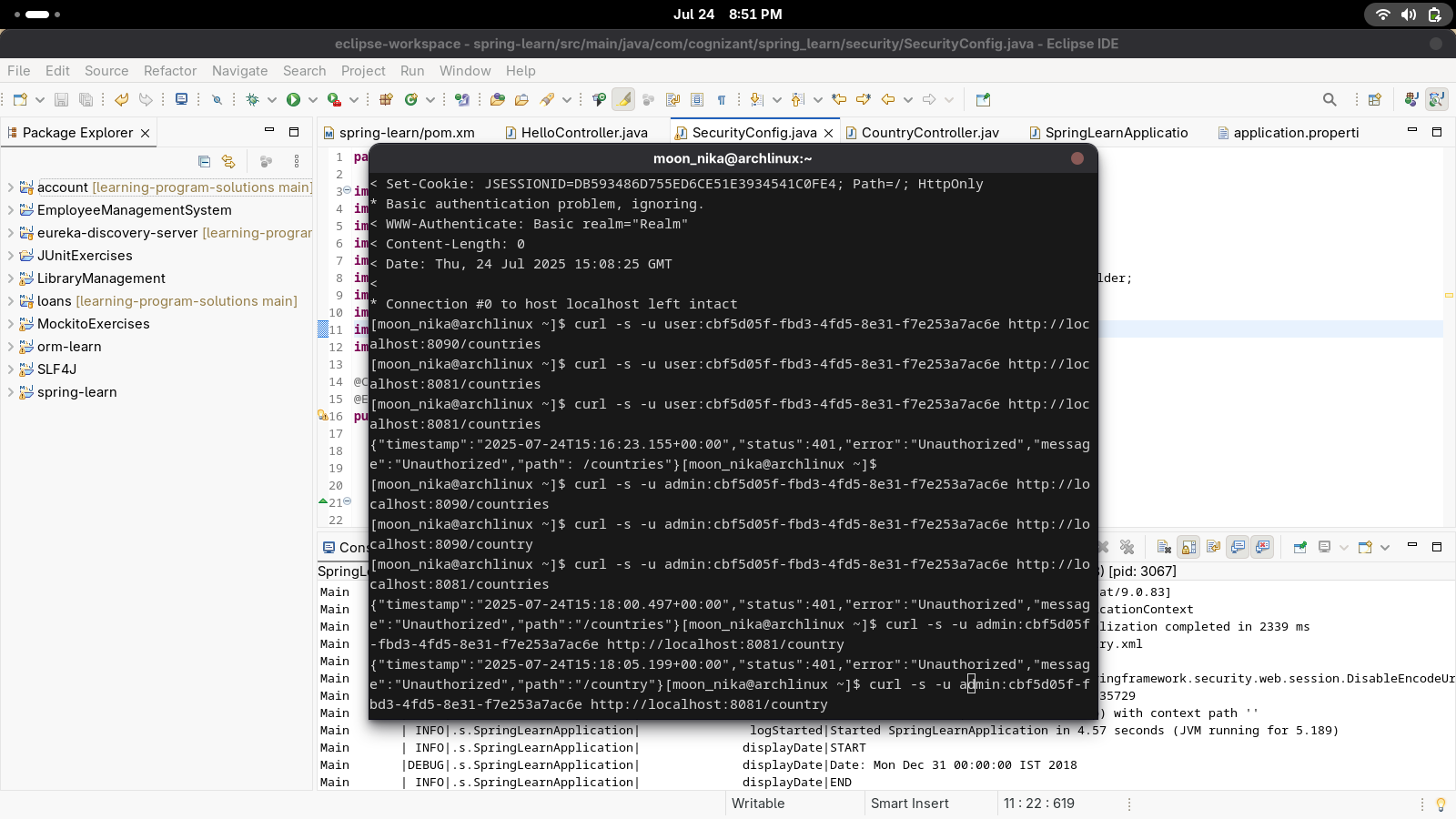
httpSecurity.csrf().disable().httpBasic().and()

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

}

}





**Create authentication service that returns JWT**

Create authentication controller and configure it in SecurityConfig

**package** com.cognizant.spring\_learn.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.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");

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

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

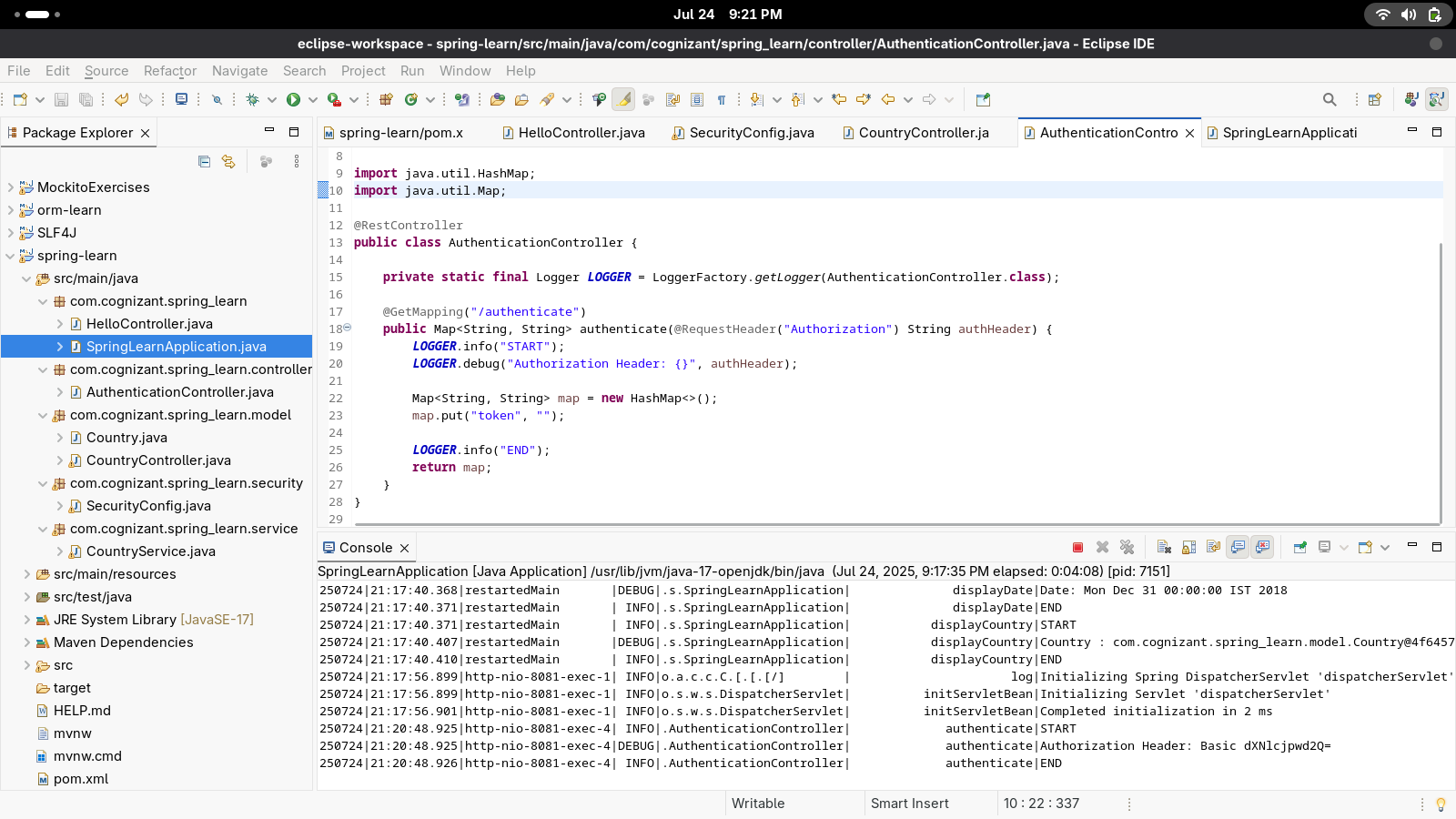
map.put("token", "");

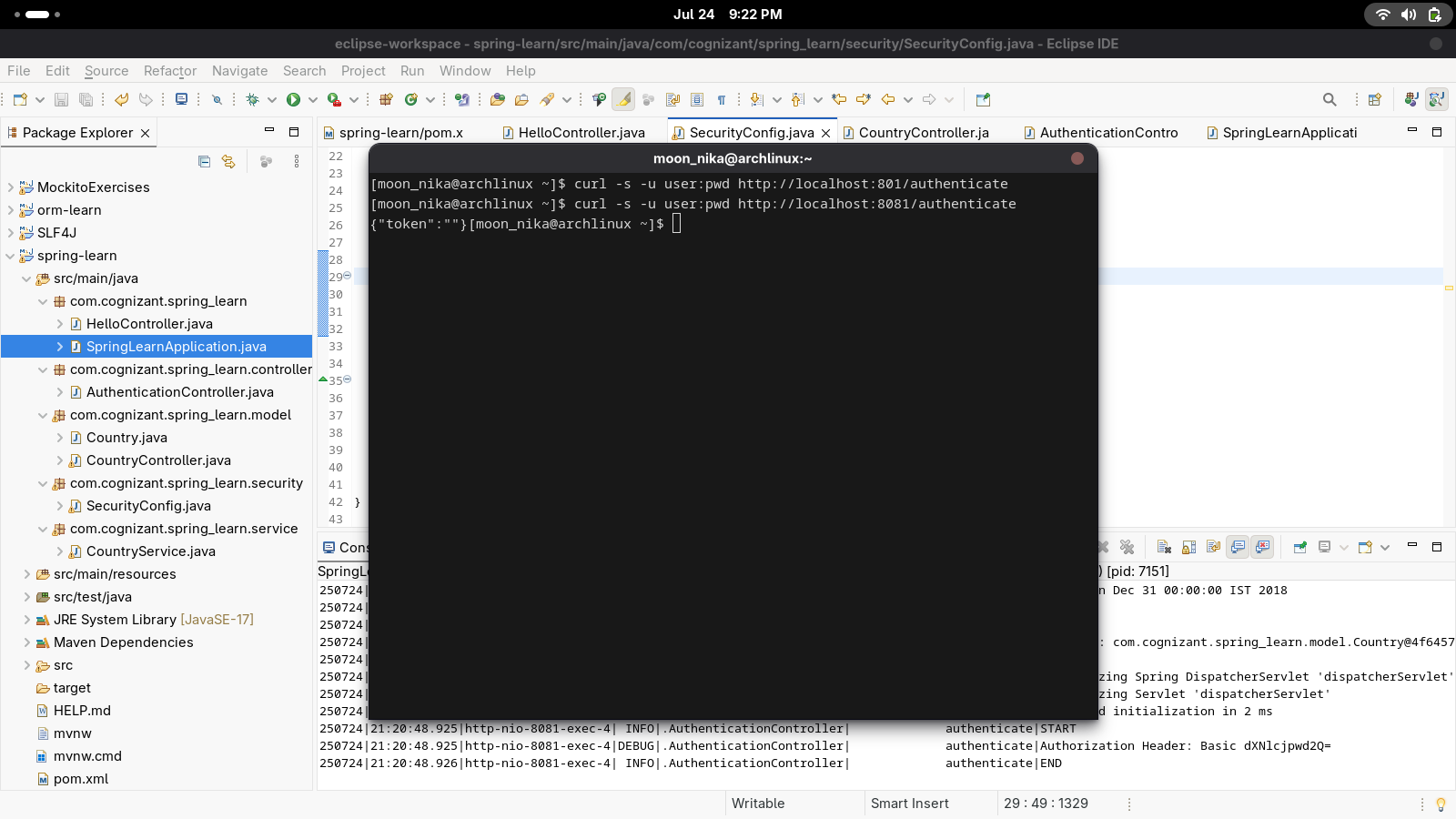
***LOGGER***.info("END");

**return** map;

}

}





Read Authorization header and decode the username and password

**package** com.cognizant.spring\_learn.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");

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

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

map.put("token", "");

***LOGGER***.info("END");

**return** map;

}

**private** String getUser(String authHeader) {

***LOGGER***.debug("Inside getUser method");

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

***LOGGER***.debug("Encoded credentials: {}", encodedCredentials);

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

String decodedString = **new** String(decodedBytes);

***LOGGER***.debug("Decoded string: {}", decodedString);

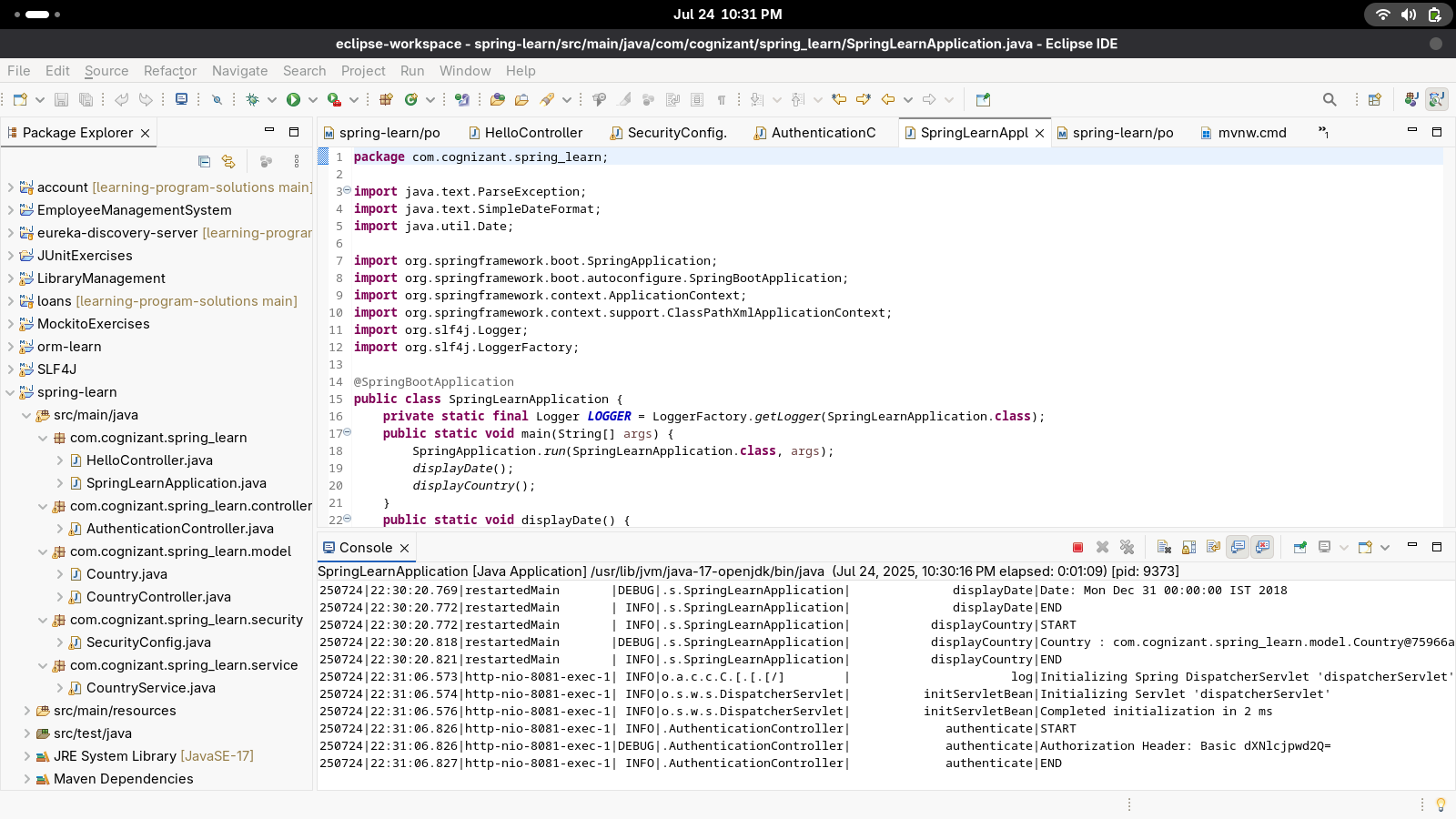
String username = decodedString.split(":")[0];

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

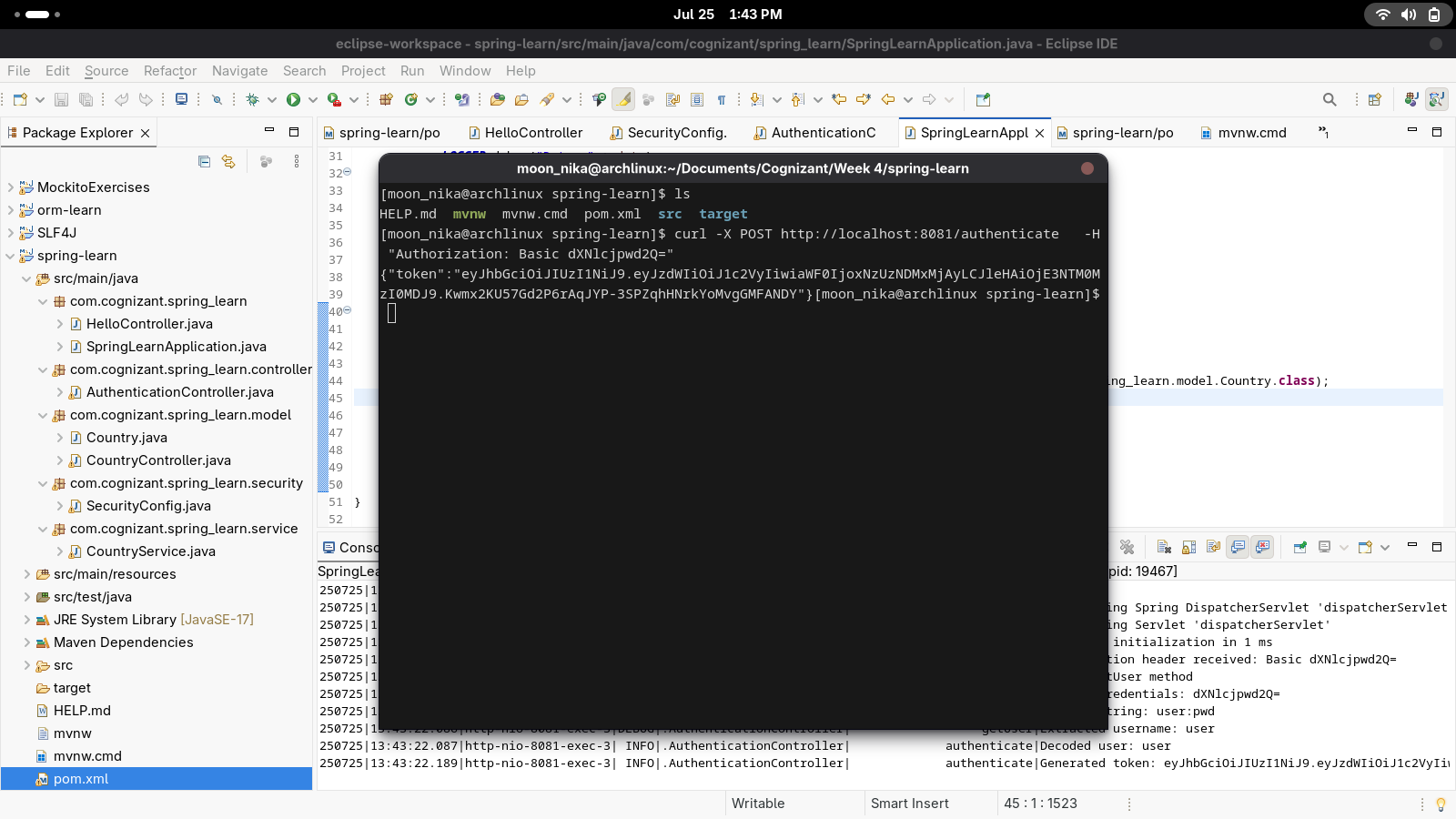
**return** username;

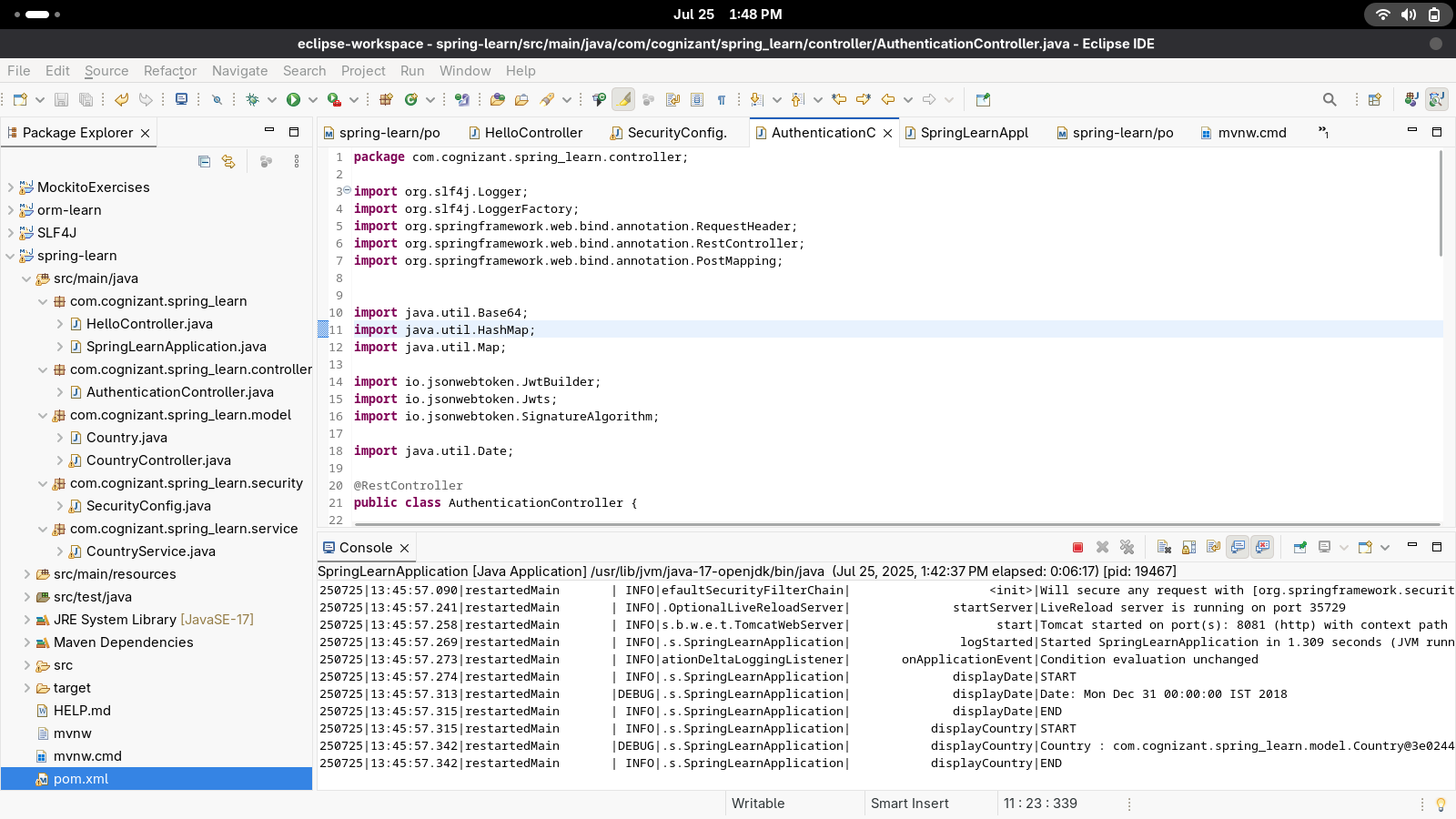
}

}



**Generate token based on the user**



****

**Authorize based on JWT**

JwtAuthorizationFilter.java

package com.cognizant.spring\_learn.security;

import javax.servlet.FilterChain;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.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 com.cognizant.spring\_learn.controller.AuthenticationController;

import java.io.IOException;

import java.util.ArrayList;

import io.jsonwebtoken.Claims;

import io.jsonwebtoken.Jws;

import io.jsonwebtoken.JwtException;

import io.jsonwebtoken.Jwts;

public class JwtAuthorizationFilter extends BasicAuthenticationFilter {

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

public JwtAuthorizationFilter(AuthenticationManager authenticationManager) {

super(authenticationManager);

LOGGER.info("Start");

LOGGER.debug("{}: ", authenticationManager);

}

@Override

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

throws IOException, ServletException {

LOGGER.info("Start");

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

LOGGER.debug("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("End");

}

private UsernamePasswordAuthenticationToken getAuthentication(HttpServletRequest request) {

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

if (token != null) {

try {

Jws<Claims> jws = Jwts.parser()

.setSigningKey("secretkey")

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

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

LOGGER.debug("User: {}", user);

if (user != null) {

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

}

} catch (JwtException ex) {

LOGGER.error("Invalid JWT", ex);

return null;

}

}

return null;

}

}

