JwtAuthorizationFilter.java:

package com.cognizant.springlearn.security;

import java.io.IOException;

import java.util.ArrayList;

import javax.servlet.FilterChain;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import io.jsonwebtoken.Claims;

import io.jsonwebtoken.Jws;

import io.jsonwebtoken.JwtException;

import io.jsonwebtoken.Jwts;

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;

public class JwtAuthorizationFilter extends BasicAuthenticationFilter {

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

public JwtAuthorizationFilter(AuthenticationManager authenticationManager) {

super(authenticationManager);

LOGGER.info("Start JwtAuthorizationFilter Constructor");

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

}

@Override

protected void doFilterInternal(HttpServletRequest req, HttpServletResponse res,

FilterChain chain) throws IOException, ServletException {

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

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("End doFilterInternal()");

}

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("JWT Subject (User): {}", user);

if (user != null) {

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

}

} catch (JwtException ex) {

LOGGER.error("JWT parsing error: {}", ex.getMessage());

return null;

}

}

return null;

}

}

SecurityConfig.java:

@Override

protected void configure(HttpSecurity httpSecurity) throws Exception {

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

.authorizeRequests()

.antMatchers("/authenticate").hasAnyRole("USER", "ADMIN")

.anyRequest().authenticated()

.and()

.addFilter(new JwtAuthorizationFilter(authenticationManager()));

}