**Exercise 12: Online Bookstore - Securing RESTful Endpoints with Spring Security**

**Business Scenario:**

Secure your bookstore's RESTful endpoints using Spring Security with JWT-based authentication.

**Instructions:**

1. **Add Spring Security:**
   * Integrate Spring Security into your project.
2. **JWT Authentication:**
   * Implement JWT-based authentication and authorization.
3. **CORS Handling:**
   * Configure CORS to handle cross-origin requests.

**pom.xml**

<dependencies>

<!-- Spring Security -->

<dependency>

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

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

</dependency>

<!-- JWT Support -->

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.1</version>

</dependency>

<!-- Web dependencies -->

<dependency>

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

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

</dependency>

</dependencies>

**SecurityConfig.java**

package com.example.bookstoreapi.config;

import com.example.bookstoreapi.security.JwtAuthenticationEntryPoint;

import com.example.bookstoreapi.security.JwtRequestFilter;

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.builders.AuthenticationManagerBuilder;

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

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

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

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

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

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

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

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

@Configuration

@EnableWebSecurity

public class SecurityConfig extends WebSecurityConfigurerAdapter {

@Autowired

private JwtAuthenticationEntryPoint jwtAuthenticationEntryPoint;

@Autowired

private UserDetailsService jwtUserDetailsService;

@Autowired

private JwtRequestFilter jwtRequestFilter;

@Autowired

public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {

auth.userDetailsService(jwtUserDetailsService).passwordEncoder(passwordEncoder());

}

@Bean

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

@Bean

@Override

public AuthenticationManager authenticationManagerBean() throws Exception {

return super.authenticationManagerBean();

}

@Override

protected void configure(HttpSecurity httpSecurity) throws Exception {

httpSecurity.csrf().disable()

.authorizeRequests().antMatchers("/authenticate", "/register").permitAll().

anyRequest().authenticated().and().

exceptionHandling().authenticationEntryPoint(jwtAuthenticationEntryPoint).and().sessionManagement()

.sessionCreationPolicy(SessionCreationPolicy.STATELESS);

httpSecurity.addFilterBefore(jwtRequestFilter, UsernamePasswordAuthenticationFilter.class);

}

@Override

public void configure(WebSecurity web) throws Exception {

web.ignoring().antMatchers("/h2-console/\*\*");

}

}

**JwtTokenUtil.java**

package com.example.bookstoreapi.security;

import io.jsonwebtoken.Claims;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

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

import org.springframework.stereotype.Component;

import java.io.Serializable;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

import java.util.function.Function;

@Component

public class JwtTokenUtil implements Serializable {

private static final long serialVersionUID = -2550185165626007488L;

public static final long JWT\_TOKEN\_VALIDITY = 5 \* 60 \* 60;

private String secret = "jwtSecretKey";

public String getUsernameFromToken(String token) {

return getClaimFromToken(token, Claims::getSubject);

}

public Date getExpirationDateFromToken(String token) {

return getClaimFromToken(token, Claims::getExpiration);

}

public <T> T getClaimFromToken(String token, Function<Claims, T> claimsResolver) {

final Claims claims = getAllClaimsFromToken(token);

return claimsResolver.apply(claims);

}

private Claims getAllClaimsFromToken(String token) {

return Jwts.parser().setSigningKey(secret).parseClaimsJws(token).getBody();

}

private Boolean isTokenExpired(String token) {

final Date expiration = getExpirationDateFromToken(token);

return expiration.before(new Date());

}

public String generateToken(UserDetails userDetails) {

Map<String, Object> claims = new HashMap<>();

return doGenerateToken(claims, userDetails.getUsername());

}

private String doGenerateToken(Map<String, Object> claims, String subject) {

return Jwts.builder().setClaims(claims).setSubject(subject).setIssuedAt(new Date(System.currentTimeMillis()))

.setExpiration(new Date(System.currentTimeMillis() + JWT\_TOKEN\_VALIDITY \* 1000))

.signWith(SignatureAlgorithm.HS512, secret).compact();

}

public Boolean validateToken(String token, UserDetails userDetails) {

final String username = getUsernameFromToken(token);

return (username.equals(userDetails.getUsername()) && !isTokenExpired(token));

}

}

**JwtAuthenticationEntryPoint.java**

package com.example.bookstoreapi.security;

import org.springframework.security.core.AuthenticationException;

import org.springframework.security.web.AuthenticationEntryPoint;

import org.springframework.stereotype.Component;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.IOException;

@Component

public class JwtAuthenticationEntryPoint implements AuthenticationEntryPoint {

@Override

public void commence(HttpServletRequest request, HttpServletResponse response,

AuthenticationException authException) throws IOException, ServletException {

response.sendError(HttpServletResponse.SC\_UNAUTHORIZED, "Unauthorized");

}

}

**JwtRequestFilter.java**

package com.example.bookstoreapi.security;

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

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

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

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

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

import org.springframework.stereotype.Component;

import org.springframework.web.filter.OncePerRequestFilter;

import io.jsonwebtoken.ExpiredJwtException;

import javax.servlet.FilterChain;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.IOException;

@Component

public class JwtRequestFilter extends OncePerRequestFilter {

@Autowired

private UserDetailsService jwtUserDetailsService;

@Autowired

private JwtTokenUtil jwtTokenUtil;

@Override

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

throws ServletException, IOException {

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

String username = null;

String jwtToken = null;

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

jwtToken = requestTokenHeader.substring(7);

try {

username = jwtTokenUtil.getUsernameFromToken(jwtToken);

} catch (IllegalArgumentException e) {

System.out.println("Unable to get JWT Token");

} catch (ExpiredJwtException e) {

System.out.println("JWT Token has expired");

}

} else {

logger.warn("JWT Token does not begin with Bearer String");

}

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

UserDetails userDetails = this.jwtUserDetailsService.loadUserByUsername(username);

if (jwtTokenUtil.validateToken(jwtToken, userDetails)) {

UsernamePasswordAuthenticationToken usernamePasswordAuthenticationToken = new UsernamePasswordAuthenticationToken(

userDetails, null, userDetails.getAuthorities());

usernamePasswordAuthenticationToken

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

SecurityContextHolder.getContext().setAuthentication(usernamePasswordAuthenticationToken);

}

}

chain.doFilter(request, response);

}

}

**JwtAuthenticationController.java**

package com.example.bookstoreapi.controller;

import com.example.bookstoreapi.model.JwtRequest;

import com.example.bookstoreapi.model.JwtResponse;

import com.example.bookstoreapi.security.JwtTokenUtil;

import com.example.bookstoreapi.service.JwtUserDetailsService;

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

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

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

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

@RestController

@CrossOrigin

public class JwtAuthenticationController {

@Autowired

private AuthenticationManager authenticationManager;

@Autowired

private JwtTokenUtil jwtTokenUtil;

@Autowired

private JwtUserDetailsService jwtUserDetailsService;

@PostMapping("/authenticate")

public JwtResponse authenticate(@RequestBody JwtRequest authenticationRequest) throws Exception {

authenticate(authenticationRequest.getUsername(), authenticationRequest.getPassword());

final UserDetails userDetails = jwtUserDetailsService

.loadUserByUsername(authenticationRequest.getUsername());

final String token = jwtTokenUtil.generateToken(userDetails);

return new JwtResponse(token);

}

private void authenticate(String username, String password) throws Exception {

try {

authenticationManager.authenticate(new UsernamePasswordAuthenticationToken(username, password));

} catch (Exception e) {

throw new Exception("Invalid username or password", e);

}

}

}