

Application.properties : can be configured with n2-databse/ mysql

server.port=9090

#spring.h2.console.enabled=true

#spring.h2.console.settings.trace=false

#spring.h2.console.settings.web-allow-others=false

#spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

#spring.datasource.url = jdbc:mysql://localhost:3306/ig

#spring.datasource.username=root

#spring.datasource.password=root

#spring.jpa.properties.hibernate.dialect= org.hibernate.dialect.MySQL8Dialect

# Hibernate ddlauto (create, create-drop, validate, update)

#spring.jpa.hibernate.ddl-auto = update

#spring.jpa.show-sql=true

Pom.xml

<dependency>

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

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<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>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>8.0.28</version>

</dependency>-->

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.1</version>

</dependency>

<dependency>

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

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

<scope>test</scope>

<exclusions>

<exclusion>

<groupId>org.junit.vintage</groupId>

<artifactId>junit-vintage-engine</artifactId>

</exclusion>

</exclusions>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

<dependency>

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

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

</dependency>

package com.ig.springsecurityjwt.api;

import java.util.List;

import java.util.stream.Collectors;

import java.util.stream.Stream;

import javax.annotation.PostConstruct;

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

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.autoconfigure.domain.EntityScan;

import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

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

import com.ig.springsecurityjwt.api.entity.User;

import com.ig.springsecurityjwt.api.repository.UserRepository;

/\*

\* Run in postman, post method, http://localhost:9090/authenticate

\* you will get jwt token, copy and paste in https://jwt.io/ to

\* decode the token

\*

\*/

@SpringBootApplication

public class SpringSecurityJwtExample2Application {

@Autowired

private UserRepository userRepository;

@PostConstruct

public void initUsers() {

List<User> users = Stream.of(

new User(101,"asr","asr@123","asr@gmail.com"),

new User(102,"user1","pwd1","user1@gmail.com"),

new User(103,"user2","pwd2","user2@gmail.com"),

new User(104,"user3","pwd3","user3@gmail.com")

).collect(Collectors.toList());

userRepository.saveAll(users);

}

public static void main(String[] args) {

SpringApplication.run(SpringSecurityJwtExample2Application.class, args);

}

}

package com.ig.springsecurityjwt.api.config;

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.BeanIds;

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.configuration.EnableWebSecurity;

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

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

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

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

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

import com.ig.springsecurityjwt.api.filter.JwtFilter;

import com.ig.springsecurityjwt.api.service.CustomUserDetailsService;

@Configuration

@EnableWebSecurity

public class SecurityConfig extends WebSecurityConfigurerAdapter{

@Autowired

private CustomUserDetailsService userDetailsService;

//Register JwtFilter in SecurityConfig class

@Autowired

JwtFilter jwtFilter;

@Override

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.userDetailsService(userDetailsService);

}

@Bean

public PasswordEncoder passwordEncoder() {

return NoOpPasswordEncoder.getInstance();

}

@Bean(name = BeanIds.AUTHENTICATION\_MANAGER)

@Override

public AuthenticationManager authenticationManagerBean() throws Exception {

return super.authenticationManagerBean();

}

@Override

protected void configure(HttpSecurity http) throws Exception {

http.csrf().disable().authorizeRequests()

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

.anyRequest().authenticated()

.and().exceptionHandling()

.and().sessionManagement()

.sessionCreationPolicy(SessionCreationPolicy.STATELESS);

http.addFilterBefore(jwtFilter, UsernamePasswordAuthenticationFilter.class);

}

/\*

public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {

auth.inMemoryAuthentication().withUser("admin").password("admin").roles("ADMIN");

}

\*/

}

**package** com.ig.springsecurityjwt.api.controller;

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

**import** org.springframework.security.authentication.AuthenticationManager;

**import** org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

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

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

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

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

**import** com.ig.springsecurityjwt.api.entity.AuthRequest;

**import** com.ig.springsecurityjwt.api.util.JwtUtil;

@RestController

**public** **class** HelloController {

@Autowired

**private** JwtUtil jwtUtil;

@Autowired

**private** AuthenticationManager authenticationManager;

/\*

\* copy the jwt token, in postman headers, paste preceding with

\* word, bearer and run http://localhost:9090/,

\* gives 403 error. So we need to add an filter to

\* inform spring application to authenticate the request with

\* the specified jwt token in the header

\* Ex. Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyMiIsImV4cCI6MTU5NjE1NTU2OSwiaWF0IjoxNTk2MTE5NTY5fQ.xET\_GLLlIPQIoCyUFSDJd662vAAAoyVXNh\_GflaJgE0

You shoukd get Welcome to yamaha message

\*/

/\*

\* Now uncheck the Authorization headr and again send in postman,

\* we will get 403 error which means our get request is following

\* stateless authentication and also the jwt token is neither stored in server

\*

\*/

@GetMapping("/")

**public** String hello() {

**return** "Welcome to Secured SpringBoot Application using JWT";

}

/\*

\* http://localhost:9090/authenticate

\* In the body type the following json object

\* {

"userName": "user2",

"password": "pwd2"

}

\* In the headers section of postman, uncheck Authorization,

\* you will get JWT token

\*

\*

\*

\*/

/\*

\* Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyMiIsImV4cCI6MTY5MzI1MDY1NiwiaWF0IjoxNjkzMjE0NjU2fQ.Ins9ydPOjIc5mqwNt41IqMf0oCy1bW0-G7ccb6EXMDc

\*/

@PostMapping("/authenticate")

**public** String generateToken(@RequestBody AuthRequest authRequest) **throws** Exception{

**try** {

authenticationManager.authenticate(**new** UsernamePasswordAuthenticationToken(authRequest.getUserName(),authRequest.getPassword()));

}**catch**(Exception e) {

**throw** **new** Exception("Invalid credentials");

}

**return** jwtUtil.generateToken(authRequest.getUserName());

}

}

package com.ig.springsecurityjwt.api.entity;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@AllArgsConstructor

@NoArgsConstructor

public class AuthRequest {

private String userName;

private String password;

}

package com.ig.springsecurityjwt.api.entity;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.Table;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@AllArgsConstructor

@NoArgsConstructor

@Entity

@Table(name = "USER\_TBL")

public class User {

@Id

private Integer id;

private String userName;

private String password;

private String email;

}

package com.ig.springsecurityjwt.api.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.core.userdetails.UserDetails;

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

import org.springframework.stereotype.Component;

import org.springframework.web.filter.OncePerRequestFilter;

import com.ig.springsecurityjwt.api.service.CustomUserDetailsService;

import com.ig.springsecurityjwt.api.util.JwtUtil;

@Component

public class JwtFilter extends OncePerRequestFilter{

@Autowired

private JwtUtil jwtUtil;

@Autowired

private CustomUserDetailsService service;

@Override

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

throws ServletException, IOException {

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

String token=null;

String userName=null;

//Ex. Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyMSIsImV4cCI6MTU5NjE0ODk3NCwiaWF0IjoxNTk2MTEyOTc0fQ.kZK6iSzga5c-JsMiJ3TSVrDwDx\_nLuhQwBvsjWqIfw0

if(authorizationHeader != null &&

authorizationHeader.startsWith("Bearer ")) {

token = authorizationHeader.substring(7);

userName=jwtUtil.extractUsername(token);

}

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

UserDetails userDetails=service.loadUserByUsername(userName);

if(jwtUtil.validateToken(token, userDetails)) {

UsernamePasswordAuthenticationToken usernamePasswordAuthenticationToken =

new UsernamePasswordAuthenticationToken(userDetails,null,userDetails.getAuthorities());

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

SecurityContextHolder.getContext().setAuthentication(usernamePasswordAuthenticationToken);

}

}

filterChain.doFilter(request, response);

}

}

package com.ig.springsecurityjwt.api.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.ig.springsecurityjwt.api.entity.User;

public interface UserRepository extends JpaRepository<User,Integer> {

User findByUserName(String username);

}

package com.ig.springsecurityjwt.api.service;

import java.util.ArrayList;

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

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

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

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

import org.springframework.stereotype.Service;

import com.ig.springsecurityjwt.api.entity.User;

import com.ig.springsecurityjwt.api.repository.UserRepository;

@Service

public class CustomUserDetailsService implements UserDetailsService{

@Autowired

private UserRepository repository;

@Override

public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {

User user= repository.findByUserName(username);

return new org.springframework.security.core.userdetails.User(user.getUserName(),

user.getPassword(), new ArrayList<>());

}

}

package com.ig.springsecurityjwt.api.util;

import io.jsonwebtoken.Claims;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

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

import org.springframework.stereotype.Service;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

import java.util.function.Function;

@Service

public class JwtUtil {

private String secret = "secret";

public String extractUsername(String token) {

return extractClaim(token, Claims::getSubject);

}

public Date extractExpiration(String token) {

return extractClaim(token, Claims::getExpiration);

}

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

final Claims claims = extractAllClaims(token);

return claimsResolver.apply(claims);

}

private Claims extractAllClaims(String token) {

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

}

private Boolean isTokenExpired(String token) {

return extractExpiration(token).before(new Date());

}

public String generateToken(String username) {

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

return createToken(claims, username);

}

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

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

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

.signWith(SignatureAlgorithm.HS256, secret).compact();

}

public Boolean validateToken(String token, UserDetails userDetails) {

final String username = extractUsername(token);

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

}

}