package com.simplilearn.capstone2.service;

import java.util.HashSet;

import java.util.Set;

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

import org.springframework.context.annotation.Lazy;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.BadCredentialsException;

import org.springframework.security.authentication.DisabledException;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.authority.SimpleGrantedAuthority;

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.simplilearn.capstone2.dao.UserDao;

import com.simplilearn.capstone2.entity.JwtRequest;

import com.simplilearn.capstone2.entity.JwtResponse;

import com.simplilearn.capstone2.entity.User;

import com.simplilearn.capstone2.util.JwtUtil;

@Service

public class JwtService implements UserDetailsService {

@Autowired

private UserDao userdao;

//@Lazy

@Autowired

private JwtUtil jwtutil;

@Autowired

private AuthenticationManager authenticationmanager;

public JwtResponse createJwtToken(JwtRequest jwtRequest) throws Exception {

String userName = jwtRequest.getUserName();

String userPassword = jwtRequest.getUserPassword();

authenticate(userName, userPassword);

final UserDetails userDetails = loadUserByUsername(userName);

String newGeneratedToken = jwtutil.generateToken(userDetails);

User user= userdao.findById(userName).get();

return new JwtResponse(user, newGeneratedToken);

}

@Override

public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {

User user = userdao.findById(username).get();

if (user != null) {

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

user.getUserpassword(),

getAuthorities(user));

}

else

{

throw new UsernameNotFoundException("username is not valid");

}

}

private Set getAuthorities(User user) {

Set authorities = new HashSet();

user.getRole().forEach(role -> {

authorities.add(new SimpleGrantedAuthority("ROLE\_" + role.getRoleName()));

});

return authorities;

}

private void authenticate(String userName, String userPassword) throws Exception {

try {

authenticationmanager.authenticate(new UsernamePasswordAuthenticationToken(userName, userPassword));

} catch (DisabledException e) {

throw new Exception("user is disabled");

} catch (BadCredentialsException e) {

throw new Exception("Bad credentials from user");

}

}

}