package com.simplilearn.capstone2.util;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

import java.util.function.Function;

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

import org.springframework.stereotype.Component;

import io.jsonwebtoken.Claims;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

@Component

public class JwtUtil {

private static final String SECRET\_KEY = "my\_movie\_plan";

private static final int TOKEN\_VALIDITY = 3600 \* 5;

public String getUserNameFromToken(String token) {

return getClaimFromToken(token,Claims::getSubject);

}

private <T> T getClaimFromToken(String token,Function<Claims, T> claimResolver) {

final Claims claims = getAllClaimsFromToken(token);

return claimResolver.apply(claims);

}

private Claims getAllClaimsFromToken(String token) {

return Jwts.parser().setSigningKey(SECRET\_KEY).parseClaimsJws(token).getBody();

}

public boolean ValidateToken(String token,UserDetails userDetails) {

String userName = getUserNameFromToken(token);

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

}

private boolean isTokenExpired(String token) {

final Date expDate = getExpirationDateFromToken(token);

return expDate.before(new Date());

}

private Date getExpirationDateFromToken(String token) {

return getClaimFromToken(token,Claims::getExpiration);

}

public String generateToken(UserDetails userDetails) {

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

return Jwts.builder()

.setClaims(claims)

.setSubject(userDetails.getUsername())

.setIssuedAt(new Date(System.currentTimeMillis()))

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

.signWith(SignatureAlgorithm.HS512, SECRET\_KEY)

.compact();

}

}

/\*

\* secret key has to be pass from this

\* return Jwts.parser().setSigningKey(null)

\* that can secret can be hard coded...fro better security it can be stored in db.

\*

\* getClaimFromToken--> higher order function of java as take a function as argument

\* all the snippet is the part the functional programming

\*

\* return expDate.before(new Date());

\* If expiration date is not before the current date this will false and the token is not expired in that case

\*/