**9장. 익스프레스로 SNS 서비스 만들기**

1. 프로젝트 구조 갖추기

nodebird라는 폴더 생성

* package.json 생성

{

"name": "nodebird",

"version": "0.0.1",

"description": "익스프레스로 만드는 SNS 서비스",

"main": "app.js",

"scripts": {

"start": "nodemon app"

},

"author": "ZeroCho",

"license": "MIT"

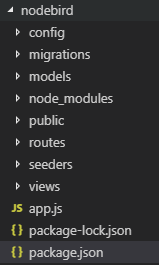
}

C:\materK1m\newmimac\test\_kes\nodebird>npm i -g sequelize-cli

C:\materK1m\newmimac\test\_kes\nodebird>npm i sequelize mysql2

C:\materK1m\newmimac\test\_kes\nodebird>sequelize init

**[nodebird 폴더 구조]**



C:\materK1m\newmimac\test\_kes\nodebird>npm i express cookie-parser express-session morgan connect-flash pug

C:\materK1m\newmimac\test\_kes\nodebird>npm i -g nodemon

C:\materK1m\newmimac\test\_kes\nodebird>npm i -D nodemon

(비밀키 모아두는 기능)

C:\materK1m\newmimac\test\_kes\nodebird>npm i dotenv

* .env

COOKIE\_SECRET=nodebirdsecret

* app.js

const express = require('express');

const cookieParser = require('cookie-parser');

const morgan = require('morgan');

const path = require('path');

const session = require('express-session');

const flash = require('connect-flash');

require('dotenv').config();

const pageRouter = require('./routes/page');

const app = express();

app.set('views', path.join(\_\_dirname, 'views'));

app.set('view engine', 'pug');

app.set('port', process.env.PORT || 8001);

app.use(morgan('dev'));

app.use(express.static(path.join(\_\_dirname, 'public')));

app.use(express.json());

app.use(express.urlencoded({ extended: false }));

app.use(cookieParser(process.env.COOKIE\_SECRET));

app.use(session({

resave: false,

saveUninitialized: false,

secret: process.env.COOKIE\_SECRET,

cookie: {

httpOnly: true,

secure: false,

},

}));

app.use(flash());

app.use('/', pageRouter);

app.use((req, res, next) => {

const err = new Error('Not Found');

err.status = 404;

next(err);

});

app.use((err, req, res, next) => {

res.locals.message = err.message;

res.locals.error = req.app.get('env') === 'development' ? err : {};

res.status(err.status || 500);

res.render('error');

});

app.listen(app.get('port'), () => {

console.log(app.get('port'), '번 포트에서 대기중');

});

* view 폴더
* layout.pug

doctype

html

head

meta(charset='UTF-8')

title= title

meta(name='viewport' content='width=device-width, user-scalable=no')

meta(http-equiv='X-UA-Compatible' content='IE=edge')

link(rel='stylesheet' href='/main.css')

body

.container

.profile-wrap

.profile

if user && user.id

.user-name= '안녕하세요! ' + user.nick + '님'

.half

div 팔로잉

.count.following-count= user.Followings && user.Followings.length || 0

.half

div 팔로워

.count.follower-count= user.Followers && user.Followers.length || 0

input#my-id(type='hidden' value=user.id)

a#my-profile.btn(href='/profile') 내 프로필

a#logout.btn(href='/auth/logout') 로그아웃

else

form#login-form(action='/auth/login' method='post')

.input-group

label(for='email') 이메일

input#email(type='email' name='email' required autofocus)

.input-group

label(for='password') 비밀번호

input#password(type='password' name='password' required)

if loginError

.error-message= loginError

a#join.btn(href='/join') 회원가입

button#login.btn(type='submit') 로그인

a#kakao.btn(href='/auth/kakao') 카카오톡

footer

| Made by&nbsp;

a(href='https://www.zerocho.com' target='\_blank') ZeroCho

block content

* main.pug

extends layout

block content

.timeline

if user

div

form#twit-form(action='/post' method='post' enctype='multipart/form-data')

.input-group

textarea#twit(name='content' maxlength=140)

.img-preview

img#img-preview(src='' style='display: none;' width='250' alt='미리보기')

input#img-url(type='hidden' name='url')

div

label#img-label(for='img') 사진 업로드

input#img(type='file' accept='image/\*')

button#twit-btn.btn(type='submit') 짹짹

.twits

form#hashtag-form(action='/post/hashtag')

input(type='text' name='hashtag' placeholder='태그 검색')

button.btn 검색

for twit in twits

.twit

input.twit-user-id(type='hidden' value=twit.user.id)

input.twit-id(type='hidden' value=twit.id)

.twit-author= twit.user.nick

-const follow = user && user.Followings.map(f => f.id).includes(twit.user.id);

if user && user.id !== twit.user.id && !follow

button.twit-follow 팔로우하기

.twit-content= twit.content

if twit.img

.twit-img

img(src=twit.img alt='섬네일')

script.

if (document.getElementById('img')) {

document.getElementById('img').addEventListener('change', function (e) {

var formData = new FormData();

console.log(this, this.files);

formData.append('img', this.files[0]);

var xhr = new XMLHttpRequest();

xhr.onload = function () {

if (xhr.status === 200) {

var url = JSON.parse(xhr.responseText).url;

document.getElementById('img-url').value = url;

document.getElementById('img-preview').src = url;

document.getElementById('img-preview').style.display = 'inline';

} else {

console.error(xhr.responseText);

}

};

xhr.open('POST', '/post/img');

xhr.send(formData);

});

}

document.querySelectorAll('.twit-follow').forEach(function (tag) {

tag.addEventListener('click', function () {

var isLoggedIn = document.querySelector('#my-id');

if (isLoggedIn) {

var userId = tag.parentNode.querySelector('.twit-user-id').value;

var myId = isLoggedIn.value;

if (userId !== myId) {

if (confirm('팔로잉하시겠습니까?')) {

var xhr = new XMLHttpRequest();

xhr.onload = function () {

if (xhr.status === 200) {

location.reload();

} else {

console.error(xhr.responseText);

}

};

xhr.open('POST', '/user/' + userId + '/follow');

xhr.send();

}

}

}

});

});

* profile.pug

extends layout

block content

.timeline

.followings.half

h2 팔로잉 목록

if user.Followings

for following in user.Followings

div= following.nick

.followers.half

h2 팔로워 목록

if user.Followers

for follower in user.Followers

div= follower.nick

* join.pug

extends layout

block content

.timeline

form#join-form(action='/auth/join' method='post')

.input-group

label(for='join-email') 이메일

input#join-email(type='email' name='email')

.input-group

label(for='join-nick') 닉네임

input#join-nick(type='text' name='nick')

.input-group

label(for='join-password') 비밀번호

input#join-password(type='password' name='password')

if joinError

.error-message= joinError

button#join-btn.btn(type='submit') 회원가입

* error.pug

extends layout

block content

h1= message

h2= error.status

pre #{error.stack}

* routes/page.js

const express = require('express');

const router = express.Router();

router.get('/profile', (req, res) => {

res.render('profile', { title: '내 정보 - NodeBird', user: null });

});

router.get('/join', (req, res) => {

res.render('join', {

title: '회원가입 - NodeBird',

user: null,

joinError: req.flash('joinError'),

});

});

router.get('/', (req, res, next) => {

res.render('main', {

title: 'NodeBird',

twits: [],

user: null,

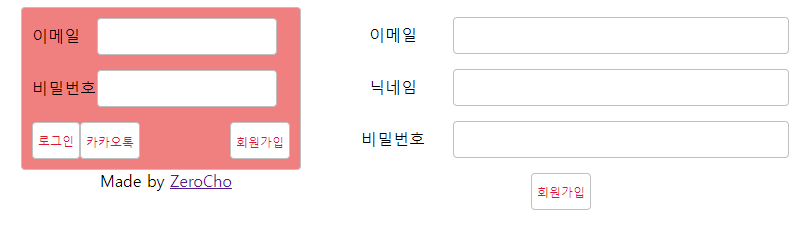
loginError: req.flash('loginError'),

});

});

module.exports = router;

* npm start



2. 데이터베이스 세팅하기

* models/user.js

module.exports = (sequelize, DataTypes) => (

sequelize.define('user', {

email: {

type: DataTypes.STRING(40),

allowNull: true,

unique: true,

},

nick: {

type: DataTypes.STRING(15),

allowNull: false,

},

password: {

type: DataTypes.STRING(100),

allowNull: true,

},

provider: {

type: DataTypes.STRING(10),

allowNull: false,

defaultValue: 'local',

},

snsId: {

type: DataTypes.STRING(30),

allowNull: true,

},

}, {

timestamps: true,

paranoid: true,

})

);

* models/post.js

module.exports = (sequelize, DataTypes) => (

sequelize.define('post', {

content: {

type: DataTypes.STRING(140),

allowNull: false,

},

img: {

type: DataTypes.STRING(200),

allowNull: true,

},

}, {

timestamps: true,

paranoid: true,

})

);

* models/hashtag.js

module.exports = (sequelize, DataTypes) => (

sequelize.define('hashtag', {

title: {

type: DataTypes.STRING(15),

allowNull: false,

unique: true,

},

}, {

timestamps: true,

paranoid: true,

})

);

* config/config.json

{

"development": {

"username": "root",

"password": "pw123",

"database": "nodebird",

"host": "127.0.0.1",

"dialect": "mysql"

},

…

}

C:\materK1m\newmimac\test\_kes\nodebird>sequelize db:create

DB를 자동으로 생성해줌

* app.js

const pageRouter = require('./routes/page');

const { sequelize } = require('./models');

const app = express();

sequelize.sync();

app.set('views', path.join(\_\_dirname, 'views'));

3. PassPort 모듈로 로그인 구현하기

C:\materK1m\newmimac\test\_kes\nodebird>npm i passport passport-local passport-kakao bcrypt

* app.js

const flash = require('connect-flash');

const passport = require('passport');

require('dotenv').config();

const pageRouter = require('./routes/page');

const { sequelize } = require('./models');

const passportConfig = require('./passport');

const app = express();

sequelize.sync();

passportConfig(passport);

…

app.use(session({

resave: false,

saveUninitialized: false,

secret: process.env.COOKIE\_SECRET,

cookie: {

httpOnly: true,

secure: false,

},

}));

app.use(flash());

app.use(passport.initialize());

app.use(passport.session());

…

* passport/index.js

const local = require('./localStrategy');

const kakao = require('./kakaoStrategy');

const { User } = require('../models');

module.exports = (passport) => {

passport.serializeUser((user, done) => {

done(null, user.id);

});

passport.deserializeUser((id, done) => {

User.find({ where: { id } })

.then(user => done(null, user))

.catch(err => done(err));

});

local(passport);

kakao(passport);

};

1. 로그인 요청이 들어옴

2. passport.authenticate 메서드 호출

3. 로그인 전략 수행

4. 로그인 성공 시 사용자 정보 객체와 함께 req.login 호출

5. req.login 메서드가 passport.serializeUser 호출

6. req.session에 사용자 아이디만 저장

7. 로그인 완료

로그인 후,

1. 모든 요청에 passport.session() 미들웨어가 passport.deserializeUser 메서드 호출

2. req.session에 저장된 아이디로 데이터베이스에서 사용자 조회

3. 조회된 사용자 정보를 req.user에 저장

4. 라우터에서 req.user 객체 사용 가능

3. 로컬 로그인 구현하기

* routes/middlewares.js

exports.isLoggedIn = (req, res, next) => {

if (req.isAuthenticated()) {

next();

} else {

res.status(403).send('로그인 필요');

}

};

exports.isNotLoggedIn = (req, res, next) => {

if (!req.isAuthenticated()) {

next();

} else {

res.redirect('/');

}

};

* routes/page.js

const express = require('express');

const { isLoggedIn, isNotLoggedIn } = require('./middlewares');

const router = express.Router();

router.get('/profile', isLoggedIn, (req, res) => {

res.render('profile', { title: '내 정보 - NodeBird', user: req.user });

});

router.get('/join', isNotLoggedIn, (req, res) => {

res.render('join', {

title: '회원가입 - NodeBird',

user: req.user,

joinError: req.flash('joinError'),

});

});

router.get('/', (req, res, next) => {

res.render('main', {

title: 'NodeBird',

twits: [],

user: req.user,

loginError: req.flash('loginError'),

});

});

module.exports = router;

* routes/auth.js

const express = require('express');

const passport = require('passport');

const bcrypt = require('bcrypt');

const { isLoggedIn, isNotLoggedIn } = require('./middlewares');

const { User } = require('../models');

const router = express.Router();

router.post('/join', isNotLoggedIn, async (req, res, next) => {

const { email, nick, password } = req.body;

try {

const exUser = await User.find({ where: { email } });

if (exUser) {

req.flash('joinError', '이미 가입된 이메일입니다.');

return res.redirect('/join');

}

const hash = await bcrypt.hash(password, 12);

await User.create({

email,

nick,

password: hash,

});

return res.redirect('/');

} catch (error) {

console.error(error);

return next(error);

}

});

router.post('/login', isNotLoggedIn, (req, res, next) => {

passport.authenticate('local', (authError, user, info) => {

if (authError) {

console.error(authError);

return next(authError);

}

if (!user) {

req.flash('loginError', info.message);

return res.redirect('/');

}

return req.login(user, (loginError) => {

if (loginError) {

console.error(loginError);

return next(loginError);

}

return res.redirect('/');

});

})(req, res, next); // 미들웨어 내의 미들웨어에는 (req, res, next)를 붙입니다.

});

router.get('/logout', isLoggedIn, (req, res) => {

req.logout();

req.session.destroy();

res.redirect('/');

});

module.exports = router;

* passport/localStrategy.js

const LocalStrategy = require('passport-local').Strategy;

const bcrypt = require('bcrypt');

const { User } = require('../models');

module.exports = (passport) => {

passport.use(new LocalStrategy({

usernameField: 'email',

passwordField: 'password',

}, async (email, password, done) => {

try {

const exUser = await User.find({ where: { email } });

if (exUser) {

const result = await bcrypt.compare(password, exUser.password);

if (result) {

done(null, exUser);

} else {

done(null, false, { message: '비밀번호가 일치하지 않습니다.' });

}

} else {

done(null, false, { message: '가입되지 않은 회원입니다.' });

}

} catch (error) {

console.error(error);

done(error);

}

}));

};

3.2. 카카오 로그인 구현하기

* passport/kakaoStrategy.js

const KakaoStrategy = require('passport-kakao').Strategy;

const { User } = require('../models');

module.exports = (passport) => {

passport.use(new KakaoStrategy({

clientID: process.env.KAKAO\_ID,

callbackURL: '/auth/kakao/callback',

}, async (accessToken, refreshToken, profile, done) => {

try {

const exUser = await User.find({ where: { snsId: profile.id, provider: 'kakao' } });

if (exUser) {

done(null, exUser);

} else {

const newUser = await User.create({

email: profile.\_json && profile.\_json.kaccount\_email,

nick: profile.displayName,

snsId: profile.id,

provider: 'kakao',

});

done(null, newUser);

}

} catch (error) {

console.error(error);

done(error);

}

}));

};

* routes/auth.js

….

router.get('/kakao', passport.authenticate('kakao'));

router.get('/kakao/callback', passport.authenticate('kakao', {

failureRedirect: '/',

}), (req, res) => {

res.redirect('/');

});

…

* app.js

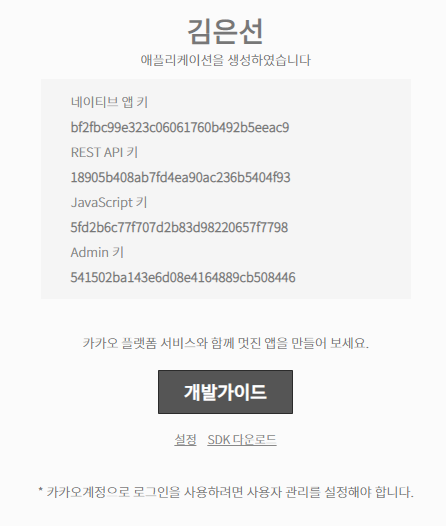
const pageRouter = require('./routes/page');

const authRouter = require('./routes/auth');

app.use('/', pageRouter);

app.use('/', authRouter);

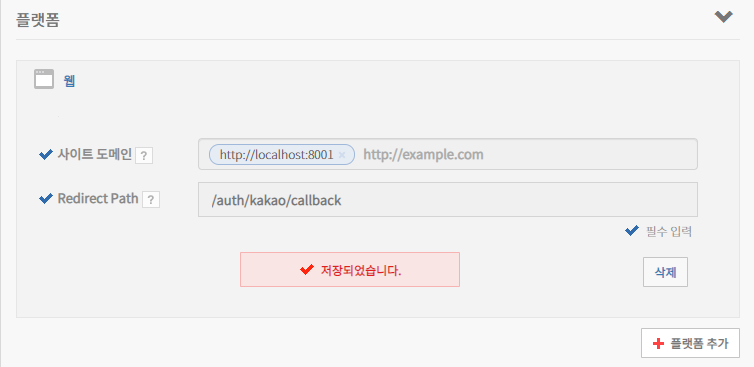
카카오 개발자 가입 : <https://developers.kakao.com/>



* .env에 카카오 REST API 키 삽입

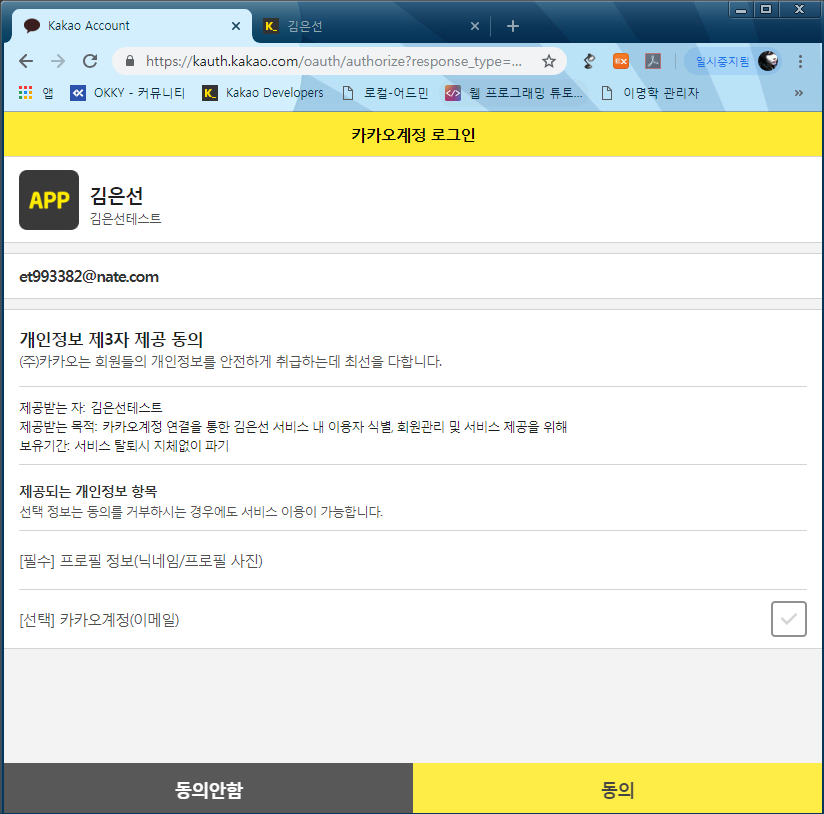
COOKIE\_SECRET=nodebirdsecret

KAKAO\_ID=18905b408ab7fd4ea90ac236b5404f93





npm start





4. Multi 모듈로 이미지 업로드 구현하기

C:\materK1m\newmimac\test\_kes\nodebird2>npm i multer

* routes/post.js

const express = require('express');

const multer = require('multer');

const path = require('path');

const fs = require('fs');

const { Post, Hashtag, User } = require('../models');

const { isLoggedIn } = require('./middlewares');

const router = express.Router();

fs.readdir('uploads', (error) => {

if (error) {

console.error('uploads 폴더가 없어 uploads 폴더를 생성합니다.');

fs.mkdirSync('uploads');

}

});

const upload = multer({

storage: multer.diskStorage({

destination(req, file, cb) {

cb(null, 'uploads/');

},

filename(req, file, cb) {

const ext = path.extname(file.originalname);

cb(null, path.basename(file.originalname, ext) + new Date().valueOf() + ext);

},

}),

limits: { fileSize: 5 \* 1024 \* 1024 },

});

router.post('/img', isLoggedIn, upload.single('img'), (req, res) => {

console.log(req.file);

res.json({ url: `/img/${req.file.filename}` });

});

const upload2 = multer();

router.post('/', isLoggedIn, upload2.none(), async (req, res, next) => {

try {

const post = await Post.create({

content: req.body.content,

img: req.body.url,

userId: req.user.id,

});

const hashtags = req.body.content.match(/#[^\s]\*/g);

if (hashtags) {

const result = await Promise.all(hashtags.map(tag => Hashtag.findOrCreate({

where: { title: tag.slice(1).toLowerCase() },

})));

await post.addHashtags(result.map(r => r[0]));

}

res.redirect('/');

} catch (error) {

console.error(error);

next(error);

}

});

module.exports = router;

* routes/page.js

const express = require('express');

const { isLoggedIn, isNotLoggedIn } = require('./middlewares');

const { Post, User } = require('../models');

const router = express.Router();

router.get('/profile', isLoggedIn, (req, res) => {

res.render('profile', { title: '내 정보 - NodeBird', user: req.user });

});

router.get('/join', isNotLoggedIn, (req, res) => {

res.render('join', {

title: '회원가입 - NodeBird',

user: req.user,

joinError: req.flash('joinError'),

});

});

router.get('/', (req, res, next) => {

Post.findAll({

include: {

model: User,

attributes: ['id', 'nick'],

},

order: [['createdAt', 'DESC']],

})

.then((posts) => {

res.render('main', {

title: 'NodeBird',

twits: posts,

user: req.user,

loginError: req.flash('loginError'),

});

})

.catch((error) => {

console.error(error);

next(error);

});

});

module.exports = router;

5. 프로젝트 마무리 하기

* routes/post.js

const express = require('express');

const multer = require('multer');

const path = require('path');

const fs = require('fs');

const { Post, Hashtag, User } = require('../models');

const { isLoggedIn } = require('./middlewares');

const router = express.Router();

fs.readdir('uploads', (error) => {

if (error) {

console.error('uploads 폴더가 없어 uploads 폴더를 생성합니다.');

fs.mkdirSync('uploads');

}

});

const upload = multer({

storage: multer.diskStorage({

destination(req, file, cb) {

cb(null, 'uploads/');

},

filename(req, file, cb) {

const ext = path.extname(file.originalname);

cb(null, path.basename(file.originalname, ext) + new Date().valueOf() + ext);

},

}),

limits: { fileSize: 5 \* 1024 \* 1024 },

});

router.post('/img', isLoggedIn, upload.single('img'), (req, res) => {

console.log(req.file);

res.json({ url: `/img/${req.file.filename}` });

});

const upload2 = multer();

router.post('/', isLoggedIn, upload2.none(), async (req, res, next) => {

try {

const post = await Post.create({

content: req.body.content,

img: req.body.url,

userId: req.user.id,

});

const hashtags = req.body.content.match(/#[^\s]\*/g);

if (hashtags) {

const result = await Promise.all(hashtags.map(tag => Hashtag.findOrCreate({

where: { title: tag.slice(1).toLowerCase() },

})));

await post.addHashtags(result.map(r => r[0]));

}

res.redirect('/');

} catch (error) {

console.error(error);

next(error);

}

});

router.get('/hashtag', async (req, res, next) => {

const query = req.query.hashtag;

if (!query) {

return res.redirect('/');

}

try {

const hashtag = await Hashtag.find({ where: { title: query } });

let posts = [];

if (hashtag) {

posts = await hashtag.getPosts({ include: [{ model: User }] });

}

return res.render('main', {

title: `${query} | NodeBird`,

user: req.user,

twits: posts,

});

} catch (error) {

console.error(error);

return next(error);

}

});

module.exports = router;

* routes/user

const express = require('express');

const { isLoggedIn } = require('./middlewares');

const { User } = require('../models');

const router = express.Router();

router.post('/:id/follow', isLoggedIn, async (req, res, next) => {

try {

const user = await User.find({ where: { id: req.user.id } });

await user.addFollowing(parseInt(req.params.id, 10));

res.send('success');

} catch (error) {

console.error(error);

next(error);

}

});

module.exports = router;

* passport/index.js

const local = require('./localStrategy');

const kakao = require('./kakaoStrategy');

const { User } = require('../models');

module.exports = (passport) => {

passport.serializeUser((user, done) => {

done(null, user.id);

});

passport.deserializeUser((id, done) => {

User.find({

where: { id },

include: [{

model: User,

attributes: ['id', 'nick'],

as: 'Followers',

}, {

model: User,

attributes: ['id', 'nick'],

as: 'Followings',

}],

})

.then(user => done(null, user))

.catch(err => done(err));

});

local(passport);

kakao(passport);

};

* app.js

const express = require('express');

const cookieParser = require('cookie-parser');

const morgan = require('morgan');

const path = require('path');

const session = require('express-session');

const flash = require('connect-flash');

const passport = require('passport');

require('dotenv').config();

const pageRouter = require('./routes/page');

const authRouter = require('./routes/auth');

const postRouter = require('./routes/post');

const userRouter = require('./routes/user');

const { sequelize } = require('./models');

const passportConfig = require('./passport');

const app = express();

sequelize.sync();

passportConfig(passport);

app.set('views', path.join(\_\_dirname, 'views'));

app.set('view engine', 'pug');

app.set('port', process.env.PORT || 8001);

app.use(morgan('dev'));

app.use(express.static(path.join(\_\_dirname, 'public')));

app.use('/img', express.static(path.join(\_\_dirname, 'uploads')));

app.use(express.json());

app.use(express.urlencoded({ extended: false }));

app.use(cookieParser(process.env.COOKIE\_SECRET));

app.use(session({

resave: false,

saveUninitialized: false,

secret: process.env.COOKIE\_SECRET,

cookie: {

httpOnly: true,

secure: false,

},

}));

app.use(flash());

app.use(passport.initialize());

app.use(passport.session());

app.use('/', pageRouter);

app.use('/auth', authRouter);

app.use('/post', postRouter);

app.use('/user', userRouter);

app.use((req, res, next) => {

const err = new Error('Not Found');

err.status = 404;

next(err);

});

app.use((err, req, res, next) => {

res.locals.message = err.message;

res.locals.error = req.app.get('env') === 'development' ? err : {};

res.status(err.status || 500);

res.render('error');

});

app.listen(app.get('port'), () => {

console.log(app.get('port'), '번 포트에서 대기중');

});