**1. 문제정의**

Express, 몽고db의 account, post를 컬렉션을 만들도록 한다.

Account 컬렉션에는 userid, userpw, friends 필드를 넣어서 로그인과 친구 목록 추가에 사용할 수 있도록 한다. Post는 글 제목, 내용, 글쓴이 등을 넣어서 제작한다.

가독성과 동적 콘텐츠 처리를 쉽게 하기 위해서 pug 템플릿을 사용한다.

라우터를 분리하여 각각의 모듈이 특정 경로 또는 서비스에 대한 요청을 처리하도록 한다

**2. 문제해결방안**

**서버 접속 ip 주소 : 52.79.253.84**

**프로젝트 구조:**

**-** node\_modules/

- public/

- css / (style.css 파일)

- images/ (사진)

- uploads/ (업로드 된 사진)

- routes/

- app.js

- blog.js

- content.js

- delete.js

- edit.js

- enter.js

- f\_blog.js

- f\_delete.js

- friends\_list.js

- home.js

- login.js

- logout.js

- mypet.js

- signup.js

- submit.js

- views/

-content.pug

- edit.pug

- enter.pug

- form.pug

- friends.pug

- login.pug

- menu.pug

- myblog.pug

- home.pug

- submit.pug

- pet.pug

- table.pug

- server..js

- package.json

- package-lock.json

**몽고bd 구조**

Blog:

-account/ userid, userpw, name, address, phone, email, friends

-post/ title, content, date

**1.** **server.js**

: Express 애플리케이션 설정. Express를 사용하여 MongoDB와 MySQL 데이터베이스에 접속한다.

데이터를 추가, 조회, 삭제하는 웹 애플리케이션을 구현하고 ejs 템플릿을 사용한다.

**1.1 MongoDB 및 Express 앱 설정:**

`mongodb` 모듈을 사용하여 MongoDB에 연결

Express 설정

Express 앱의 필요한 라이브러리들을 불러오고 설정

View 엔진으로 pug 설정

**1.2 라우터를 가지고 오기:**

require 함수를 사용하여 파일 경로를 통해 각각의 라우터 모듈을 가져오기. homeRouter, signupRouter, blogRouter, loginRouter 등 변수들은 각각 라우터 객체.

**1.3 세션 사용**

세션 암호에 사용할 secretKey 생성

const crypto = require('crypto');

const secretKey = crypto.randomBytes(32).toString('hex');

세션 사용

app.use(session({

secret: secretKey, #암호로 사용

resave: false,

saveUninitialized: true,

}));

**1.4 각 라우터 객체를 등록.**

경로를 지정하고 해당 경로를 처리를 담당할 라우터 객체 전달

**2. ‘/’ 페이지. home.js, home.pug**

: ‘/’ 경로에 사용될 홈 페이지 라우터. 초기 페이지를 설정한다.

핫 토픽으로 동물들 사진을 보여준다.

전체 post 데이터를 서버에서 가져와 table 형태로 보여준다. 글의 제목을 누르면 content/id 페이지로 가서 상세 글을 볼 수 있다. 글쓴이를 누르면 해당 글쓴이의 블로그로 이동할 수 있도록 했다.

로그인을 하면 최근 추가한 친구들의 post 3개를 보여준다. 만약 로그인을 안했다면 ‘로그인을 하시오’ 라는 메시지를 출력

**2.1 routes/home.js**

1.1.1express 모듈 가져오기. express 라우터 객체를 생성

const express = require('express');

const router = express.Router();

2.1.2 ‘/’ 경로에 대한 get 요청 핸들러 정의. 비동기적인 작업을 수행하기 위해 ‘async’키워드를 사용했다.

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

//server.js에서 MongoDB 데이터베이스 연결 위해 생성한 mydb 변수 얻기

const mydb = req.app.locals.mydb;

//로그인을 이미 했을 때 보여줄 메시지 저장

const infomessage = req.flash('info');

//로그인 안 했을 때 친구들의 post를 못 본다는 메시지 저장 변수

let msg;

//현재 로그인 유저 친구 목록 저장 배열 변수

let friends = [];

let count = 0; //for문에서 사용할 변수

const user = req.session.user;

try

if 로그아웃 상태 (!user)

msg = “로그인 해서 친구 post 확인!”

else

// 몽고DB에서 userid 찾아서 result 객체에 저장

result = await mydb.collection('account').findOne({ userid: req.session.user.userid });

if result (찾은 경우)

friends = result.friends || []; // friends가 null이면 빈 배열로 초기화

else friends = [] 빈배열로 초기화

'post' 컬렉션에서 데이터 가져오기

const posts = await mydb.collection('post').find().toArray();

// 렌더링 시 데이터 전달

res.render('home', { infomessage, user, msg, friends, data: posts, count })

2.1.3 라우터 객체를 모듈로 내보내기

module.exports = router;

**2.2 views/home.pug**

**Body**

이미 로그인된 경우

If (또 로그인 페이지 or 회원가입 들어감) infomessage가 있으면

Script.

alert('#{infomessage}'); 메시지 alert로 출력

strong 핫토픽 > 멸종 위기 동물

.images-container

Img, figcaption을 사용해 여러 동물 사진들을 출력

//post 데이터를 보여주는 테이블

.table\_form

table.table

thead

th 글 제목, 글쓴이 작성일 출력

tbody

//전체 post중 5개의 post만 보여준다.

each item in data.slice(0,5)

tr

//해당 글 페이지로 이동

td: a(href=`/content/${item.\_id}`) #{item.title}

//해당 글쓴이의 블로그로 이동

td: a(href=`/f\_blog/${item.author}`) #{item.author}

td= item.date //글 작성일

//최근 친구들의 post 보여주는 테이블

If msg //로그아웃 상태인 경우

Strong= msg 메시지 출력

Else

//친구가 빈 문자열, 친구가 없는 경우

if (friends.length === 0)

strong 아직 친구가 없습니다 친구를 추가하세요

else

//친구들 post를 데이블로 출력

Table thead tr th 글 제목, 글쓴이, 작성일 출력

Tbody

each name in friends.reverse() //최신부터 출력하기 위해 거꾸로. 각 친구들 이름 name

- count = 0;

each item in data //전체 post 데이터 item으로

친구가 쓴 게시물인 경우

if item.author == name && count < 3

tr

td: a(href=`/content/${item.\_id}`) #{item.title}

td: a(href=`/f\_blog/${item.author}`) #{item.author}

td= item.date

- count++; //3개 까지만 출력

**3. ‘/signup’ 페이지. 회원가입 페이지 signup.js, form.pug.**

**3.1 routes/signup.js**

3.1.1 라우터 및 암호 모듈 로드:

express 모듈과 sha256 암호 모듈을 로드

router 객체를 생성하여 라우팅을 처리.

3.1.2 라우터 및 암호 모듈 로드, formatPhone()함수 정의

전화번호 가공 함수 formatPhone 함수는 전화번호를 특정 형식으로 가공하는 역할.

3.1.3 ‘/’ 경로에 대한 get 요청 핸들러 정의.

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

if 로그인 상태

req.flash('info', '이미 로그인 되었습니다.'); flash 메시지 설정

res.redirect('/'); 메인페이지로 이동

else res.render(‘form’) ‘form’ 템플릿(회원가입)을 랜더링

3.1.4 ‘/’ 경로에 대한 post 요청 핸들러 정의(회원 가입 버튼 클릭)

router.post('/', (req, res) =>

const mydb = req.app.locals.mydb //mydb 가져오기

요청으로부터 데이터 추출 name, add, (생략), userid, userpw 변수에 저장

const userid = req.body.userid;

const userpw = req.body.userpw;

//account에 정보 추가

mydb.collection("account").insertOne({

userid : userid,

userpw : sha(userpw), //비밀번호 암호화

name : name, address : add, phone : phone, email : email,

friends : [],

})

.then((result) => 성공

/table 템플릿 렌더링, 추출한 데이터를 전달해서 사용자 정보를 table로 보여줌

res.render('table', {name, add, phone, email, userid, userpw});

3.1.5 라우터 객체를 모듈로 내보내기 module.exports = router;

**3.2 views/form.pug, table.pug**

**3.2.1 form.pug**

**Body**

이미 로그인했는데

If (회원가입 들어감) infomessage가 있으면

Script.

alert('#{infomessage}'); 메시지 alert로 출력

회원가입 form

form(action="/signup" method="post") 폼이 제출될 때 전송될 url 설정

각 이름(name=’name’), 번호 .. 아이디(name=’userid’), 비밀번호(name=’userpw’)란이 있다.

가입 버튼

**3.2.2 table.pug**

**Body**

회원 가입 완료 메시지 출력

P=name+ ‘님의 회원정보’

입력한 정보를 토대로 table을 출력해서 보여준다.

tbody

tr

//이름 변수

td(rowspan = "5")= name

td Phone … (생략)

**4. ‘/login’ 페이지. 로그인 페이지 login.js, login.pug.**

**4.1 routes/login.js**

4.1.1 모듈 및 패키지 로드: express: Express 프레임워크를 사용하기 위한 모듈

express-flash: Flash 메시지를 사용하기 위한 모듈 (로그인 했는데 로그인)

sha256: 비밀번호를 해싱하기 위한 SHA-256 해시 함수를 사용하기 위한 모듈.(회원가입에서 암호화해서)

4.1.2 ‘/’ 경로에 대한 get 요청 핸들러 정의.

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

로그인 상태

if(req.session.user)

console.log('세션유지');

req.flash('info', '이미 로그인 되었습니다.'); flash 메시지 설정

res.redirect('/'); //메인페이지로 이동

else res.render('login'); //로그인 안되있으면 login 랜더링

4.1.3 ‘/’ 경로에 대한 post 요청 핸들러 정의.

router.post('/', (req, res) =>

const mydb = req.app.locals.mydb; app.locals에서 mydb 가져오기

MongoDB의 account 컬렉션에서 입력받은 아이디(userid)로 사용자를 찾기

mydb.collection('account').findOne({userid : req.body.userid})

.then((result) =>)

입력된 비밀번호를 해싱하여 비교

If 비밀번호일치 result.userpw == sha(req.body.userpw) 일치하면

//req.session.user에 전달된 req.body(userid,userpw)를 저장

req.session.user = req.body

res.redirect('/') 메인 페이지로 리다이렉션

Else //비밀번호가 일치하지 않으면 에러메시지

4.1.4 라우터 객체를 모듈로 내보내기 module.exports = router;

**4.2 views/login.pug**

Body

//로그인을 할 form

form(action="/login" method="post") 입력한 정보를 POST로 "/login" 경로로 전송

id 입력 input(name = userid) 태그

pw 입력 input(name = userpw) 태그

button. 제출버튼

**5. ‘/logout’ 기능. logout.js, menu.pug. : 로그아웃 링크를 네비게이션에 추가. 로그인을 한 상태에서만 보일 수 있도록 한다.**

**5.1 logout.js**

5.1.1 express모듈, 라우터 객체 const router = express.Router() 생성

5.1.2 ‘/’ 경로에 대한 get 요청 핸들러 정의

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

req.session.destroy(); 세션 파괴

res.redirect('/'); 다시 메인페이지로 이동

5.1.3 라우터 객체를 모듈로 내보내기 module.exports = router;

**5.1 menujs**

Nav ~ 생략

form.d-flex()

if user 로그인을 한 경우

a(href="/logout") Logout 링크 추가

**6. ‘/blog’ 페이지. 나의 블로그, 친구 블로그 blog.js, f\_blog.js, myblog.pug.**

**-Myblog.pug에서 글 목록 table에서 내가 쓴 글이면 삭제 번튼이 출력되게 한다.**

**-로그인한 상태이고 내 블로그인 경우 글쓰기 버튼이 나오게 한다.**

**-로그인한 상태이고 내 친구목록에 없는 경우일 때 친구추가 버튼이 출력하도록 한다.**

**6.1 routes/blog.js**

6.1.1 express모듈, 라우터 객체 const router = express.Router() 생성

6.1.2 ‘/’ 경로에 대한 get 요청 핸들러 정의

//비동기 async 키워드는 해당 함수가 비동기 함수.

await mydb.collection('account').findOne({ userid: req.session.user.userid }) 와 같이 몽고DB에서 데이터를 조회하는 부분에서는 데이터베이스 작업이 완료될 때까지 대기하며, 그동안 다른 작업을 처리 가능.

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

로그인을 안했으면 blog를 보여주지 않느다.

if(!req.session.user || req.session.user === undefined){

res.redirect('/login'); 로그인 페이지로 이동

return; // 함수 종료

}

const user = req.session.user; user객체

const author = req.session.user.userid 해당 유저 아이디

const mydb = req.app.locals.mydb; app.locals 부터 mydb 가져오기

let friends = []; 친구의 이름을 저장할 배열

try

몽고DB에서 현재 사용자의 계정 정보 추출

accountResult = await mydb.collection('account').findOne({ userid: req.session.user.userid });

friends = accountResult.friends || [];

'post' 컬렉션에서 게시물 데이터 추출

const posts = await mydb.collection('post').find().toArray()

myblog템플릿 렌더링. 사용자정보, 친구목록, 블로그 주인의 아이디, 게시물 데이터 전달

res.render('myblog', { user, friends, author, data: posts });

catch(error) {오류 메시지}

6.1.3 라우터 객체를 모듈로 내보내기 module.exports = router;

**6.2 rountes/f\_blog.js. 글 목록에서 작성자 이름을 누르면 해당 글쓴이의 블로그로 갈 수 있도록한다. f\_blog/글쓴이 -> 시멘틱 url 방식의 매개변수 처리.**

**로그인을 안 했을 경우 f\_blog/글쓴이 페이지에서 글쓰기, 삭제 버튼X,**

**현재 친구 목록에 있지 않은 경우 친구 추가 버튼이 보일 수 있게 한다**

**‘/enter’:** txt 파일을 선택하여 업로드 후, 문서의 내용을 컨텐츠에 포함가능

6.2.1 express모듈, const ObjId = require('mongodb').ObjectId 라우터 객체 const router = express.Router() 생성

6.2.2 ‘f\_blog/:author’ 경로에 대한 get 요청 핸들러 정의

//비동기, 시멘틱 url 방식

router.get('/:author', async (req, res) =>

로그인X인 경우, 글쓰기, 삭제 등 버튼이 안 보이게 하기 위해서 user을 null로 처리

if(!(req.session.user))

user = null;

else {user = req.session.user; 현재 user객체}

let myfriends = [] 로그인 유저 친구 저장 변수

const author = req.params.author 글쓴이 아이디 변수

let friends = [] 글쓴이의 친구 저장 변수

const mydb = req.app.locals.mydb;

let userResult = ""

try

if (user != null) 로그인한 경우

현재 로그인한 사용자의 정보 및 친구 데이터 로드

userResult = await mydb.collection('account').findOne({ userid: user.userid });

myfriends = userResult.friends || [];

글쓴이(블로그 주인f\_blog)의 정보 및 친구 데이터 로드

const accountResult = await mydb.collection('account').findOne({ userid: author });

friends = accountResult.friends || [];

글 목록 db에서 가져오기

const posts = await mydb.collection('post').find().toArray();

'myblog' 템플릿을 렌더링, 각 데이터 전달

res.render('myblog', { myfriends, user, userResult, friends, author, data: posts });

catch(error) {오류메시지}

6.2.3 라우터 객체를 모듈로 내보내기 module.exports = router;

**6.3 myblog.pug**

6.3.1

Body

if user !=null && user.userid == author 내 블로그인 경우

h1.display-4.mb-4 My Blog

else

h1.display-4.mb-4= author + ' 님 블로그'

//친구 목록을 보여주는 영역

.mb-3.text-center

strong 친구 목록

if (friends.length === 0) 친구가 빈 문자열, 친구가 없는 경우

strong 아직 친구가 없습니다

else

table.table.table-sm

thead

tr

th 아이디

tbody

each name in friends 친구 목록에서 친구 이름 출력

tr

td= name

//글 목록 영역

.table (~생략)

tbody

each item in data

post에서 해당 글쓴이가 쓴 경우만

if(author == item.author)

tr

td: a(href=`/content/${item.\_id}`) #{item.title} //링크로 상세 글로 이동

td= item.date

해당 유저인 경우만 삭제 버튼

if user != null && user.userid == author

td: button.delete.btn.btn-outline-danger(**data-id=item.\_id**) 삭제

//글쓰기 버튼

해당 유저인 경우만 글쓰기 가능

if user != null && user.userid == author

button.write.btn.btn-primary(type="submit") 글쓰기

//친구추가 버튼

본인이 아니고 친구목록에 없는 경우

if user != null && user.userid != author && !(myfriends.includes(author))

td: button.add.btn.btn-outline-danger(**data-id=userResult.\_id, data-new=author**) 친구추가

script(src="https://code.jquery.com/jquery-3.6.4.min.js")

script.

//글쓰기 버튼 클릭 함수 정의

$('.write').click(function(e)

$.ajax

type: 'get', //요청 방식

url: '/enter', //요청 경로

.done(function(result)

글을 추가하는 /enter로 이동

location.href = '/enter';

.fail(function(xhr, textStatus, errorThrown) 실패

//친구 추가 버튼 클릭 함수 정의

$('.add').click(function(e)

let sid = e.target.dataset.id 이벤트대상의 data-id의 내용 저장

let new\_f = e.target.dataset.new; 이벤트대상의 data-new 내용 저장

let item = $(this);

$.ajax

type: 'post', 요청 방식

url: '/add\_friend', 요청 경로

data: {\_id: sid, new\_friend: new\_f} 전송데이터 /add\_friend에 전송됨

.done(function(result) 성공

location.reload();

.fail(function(xhr, textStatus, errorThrown) 실패

//데이터 삭제 버튼 클릭 함수 정의

$('.delete').click(function(e)

let sid = e.target.dataset.id 이벤트대상의 data-id의 내용 저장

let item = $(this);

$.ajax

type: 'post', 요청 방식

url: '/delete', 요청 경로

data: {\_id: sid } 전송데이터 /delete에 전송됨

.done(function(result) 성공

item.parent('td').parent('tr').remove(); 해당 행 제거

location.reload(); 리로드

.fail(function(xhr, textStatus, errorThrown) 실패

**7. ‘/add\_friend’(친구추가), ‘/delete'(post 데이터 삭제), ‘/enter’(글쓰기)**

**7.1 ‘/add\_friend’(친구추가)**

7.1.1 const ObjId = require('mongodb').ObjectId, express 모듈, 라우터 객체

7.1.2 ‘/add\_friend’ 경로에 대한 post 요청 핸들러 정의(친구추가 버튼 클릭)

router.post("/", (req, res) =>

전송된 id와 친구이름이 req.body안에 있다.

req.body.\_id = new ObjId(req.body.\_id) //id로

updateOne()으로 account 친구목록 변경

$push 연산자: friends 배열에 req.body.new\_friend 값을 추가

mydb.collection('account')

.updateOne({ \_id: req.body.\_id }, {$push: { friends: req.body.new\_friend } })

.then((result) => {

res.redirect('/');

.catch((err) => 에러메시지

7.1.3 라우터 객체를 모듈로 내보내기 module.exports = router;

**7.2 ‘/delete’ (post 삭제)**

7.2.1 const ObjId = require('mongodb').ObjectId, express 모듈, 라우터 객체

7.2.2 ‘/delete’ 경로에 대한 post 요청 핸들러 정의(삭제 버튼 클릭)

router.post("/", (req, res) =>

전송된 \_id가 req.body안에 있다.

req.body.\_id = new ObjId(req.body.\_id) //id로

deleteOne()으로 post 컬렉션에서 해당 데이터를 삭제

mydb.collection('post')

.deleteOne(req.body)

.then((result) => {

res.redirect('/');

.catch((err) => 에러메시지

7.2.3 라우터 객체를 모듈로 내보내기 module.exports = router;

**7.3 ‘/enter’ (글쓰기)**

7.3.1 모듈 및 라우터 생성, 저장 경로 설정 및 변수

const express = require('express');

const multer = require('multer');

const fs = require('fs');

const path = require('path');

const storage = multer.diskStorage({/\* ... \*/});

const upload = multer({ storage: storage });

const router = express.Router();

let documentContent = '';

저장될 경로 설정

const storage = multer.diskStorage({

//destination 함수: 파일이 저장될 디렉토리 설정

destination(req, file, done) {

done(null, './public/uploads/');

},

파일 이름 설정

filename(req, file, done) {

파일 확장자 추출

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

파일 이름 설정 (원래 파일이름 + 현재 시간 + 확장자)

done(null, path.basename(file.originalname, ext) + Date.now() + ext);

},

});

const upload = multer({ storage: storage });

let documentContent = ''; //첨부된 문서의 내용 저장

7.3.2 라우터 핸들러 정의

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

res.render('enter');

7.3.3 글 저장 post 핸들러

router.post('/save', (req, res) =>

let text\_content; //첨부 문서 내용 저장

// 호출되는 시간 기록

let now = new Date();

userid = req.session.user.userid; //로그인 userid

파일 첨부 내용이 있다면

if(documentContent){

text\_content = documentContent; //text\_content로 복사

text\_content = text\_content+ '<br>' + req.body.content; 컨텐츠 내용과 붙이기

MongoDB의 post 컬렉션에 새로운 글을 추가

` mydb.collection('post').insertOne(

{title : req.body.title, content : text\_content, author : userid, date : now.toLocaleDateString('KR')})

.then(result =>

console.log(result);

res.redirect('/blog');

7.3.4 문서 첨부 post 핸들러

router.post('/text', upload.single('document'), function(req, res){

console.log(req.file);

// 업로드된 파일이 있다면

if (req.file)

이미지 경로 저장

const filePath = req.file ? 'public/uploads/' + req.file.filename : null

업로드된 파일의 내용은 documentContent에 저장

documentContent = fs.readFileSync(filePath, 'utf-8');

else

파일이 제대로 업로드 되지 않은 경우

**8. content.js: 게시글 링크 클릭 시 전체 내용 보여주기**

**content/id -> 시멘틱 url 방식의 매개변수 처리.**

8.1 const ObjId = require('mongodb').ObjectId, const express = require('express'), const router = express.Router();

8.2 ‘contnet/id’ get 요청 핸들러 정의

router.get("/:id", function (req, res)

let new\_id = new ObjId(req.params.id);

let userid; //userid 저장 변수

const user = req.session.user user객체

로그인X 경우

if(!(req.session.user))

userid = ""; 빈 것으로 초기화

else

사용자 아이디 저장 변수

userid = req.session.user.userid;

몽고db에서 id필드가 값은 것을 찾기

mydb.collection('post').findOne({ \_id: new\_id })

.then(result =>

console.log(result);

res.render('content', { user, userid, data : result }); 넘겨주기

.catch( 오류 )

8.3 라우터 객체를 모듈로 내보내기 module.exports = router;

**9. ‘/friends\_list’: 친구목록 페이지, 친구 삭제, 친구 추가 버튼구현. friends\_list.js, friends.pug, f\_delete.js**

**로그인한 유저의 친구 목록을 table로 보여준다. 친구 목록 table에 친구 삭제 버튼을 출력하여 해당 id의 친구를 제거할 수 있도록 구현**

**Account 컬렉션에서 아직 친구목록에 들어가지 않은 사람들을 보여주고 친구추가 버튼을 출력한다. 친구추가 버튼 구현은 앞과 동일**

**9.1 friends\_list.js**

9.1.1 const express = require('express');

라우터 객체

const router = express.Router();

9.1.2‘/friends\_list’ 요청에 대한 핸들러 정의

//비동기

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

로그인이 안되어 있는 경우

if(!req.session.user || req.session.user === undefined)

로그인 페이지로 이동

res.redirect('/login');

return; // 함수 종료

let friends = []; 친구목록 저장 배열 변수

//user객체

const user = req.session.user;

//해당 유저 아이디

const userid = req.session.user.userid;

try {

내 아이디로 account에서 찾은 결과

const userResult = await mydb.collection('account').findOne({ userid: user.userid});

해당 유저 아이디로 찾기

const account = await mydb.collection('account').findOne({ userid: req.session.user.userid });

friends = account.friends || []; 친구 목록 저장

account 데이터 가지고 오기

const accountResult = await mydb.collection('account').find().toArray();

console.log(accountResult);

템플릿을 렌더링

res.render('friends', { user, friends, userResult, userid, data: accountResult });

} catch (error) {에러처리}

9.1.2 라우터 객체를 모듈로 내보내기 module.exports = router;

**9.2 friends.pug**

Body

p=userid+ '님의 친구리스트'

친구 리스트 테이블

table.table

thead

tr

th(scope="col") 아이디

th 블로그

tbody

친구 목록 friends 출력

each name in friends

tr

td= name

td.blog\_link: a(href=href=`/f\_blog/${name}`) #{name}님의 blog

친구삭제 버튼

td: button.f\_delete(**data-id=userResult.\_id, data-new=name**) 친구 삭제

친구 추가 리스트 테이블

table.table

thead

tr

th(scope="col") 아이디

th 블로그

tbody

each item in data

tr

내 account가 아니고 친구목록에 없는 경우

if userid != item.userid && !(friends.includes(item.userid))

td= item.userid

td.blog\_link: a(href=href=`/f\_blog/${name}`) #{name}님의 blog

친구추가 버튼

td: button.add.(**data-id=userResult.\_id, data-new=item.userid**) 친구추가

script(src="https://code.jquery.com/jquery-3.6.4.min.js")

script.

//친구 삭제 버튼 클릭 함수 정의

$(' f\_delete ').click(function(e)

let sid = e.target.dataset.id 이벤트대상의 data-id의 내용 저장

let f\_delete = e.target.dataset.new; 이벤트대상의 data-new 내용 저장

let item = $(this);

$.ajax

type: 'post', 요청 방식

url: '/delete\_friend', 요청 경로

data: {\_id: sid, delete\_friend: f\_delete} 전송데이터 '/delete\_friend'에 전송

.done(function(result) 성공

location.reload();

.fail(function(xhr, textStatus, errorThrown) 실패

//친구 추가 버튼 클릭 함수 정의

$('.add').click(function(e)

let sid = e.target.dataset.id 이벤트대상의 data-id의 내용 저장

let new\_f = e.target.dataset.new; 이벤트대상의 data-new 내용 저장

let item = $(this);

$.ajax

type: 'post', 요청 방식

url: '/add\_friend', 요청 경로

data: {\_id: sid, new\_friend: new\_f} 전송데이터 /add\_friend에 전송됨

.done(function(result) 성공

location.reload();

.fail(function(xhr, textStatus, errorThrown) 실패

**9.3 f\_delete.js (**/delete\_friend' post 요청을 처리**)**

9.3.1 router.post("/", function (req, res)

req.body.\_id = new ObjId(req.body.\_id);

const mydb = req.app.locals.mydb;

account 컬렉션에 해당id의 값을 업데이트

mydb.collection('account')

$pull 연산자: friends 배열에 req.body.new\_friend 값을 추가

.updateOne({ \_id: req.body.\_id }, {$pull: { friends: req.body.delete\_friend } })

.then((result) =>

console.log('친구 삭제');

res.redirect('/');

.catch(에러처리)

9.3.2 라우터 객체를 모듈로 내보내기 module.exports = router;

**10 edit.js, edit.pug: ‘content/id’ 경로 페이지에서 본인이 쓴 글이면 수정하기 링크가 나오도록 해서 글을 수정할 수 있도록 한다.**

**10.1 edit.js**

10.1.1 router.get("/:id", function (req, res)

let userid = req.session.user.userid;

let new\_id = new ObjId(req.params.id);

몽고db에서 변경할 해당 id 찾기

mydb.collection('post').findOne({ \_id: new\_id })

.then(result => {

res.render('edit.pug', { user, data : result }); // 넘겨주기

.catch(에러처리)

10.1.2 router.post('/', function(req, res) 수정 버튼을 누르고 post 처리

해당 id의 값을 req.body로 받은 값으로 변경한다.

mydb.collection('post').updateOne({\_id: new\_id},

{$set: {title : req.body.title, content : req.body.content, author : userid, date : req.body.someDate}})

.then(result => {

res.redirect('/blog');

10.1.3 라우터 객체를 모듈로 내보내기 module.exports = router;

**10.2 edit.pug**

Body

h1 글 수정하기

수정 form, 사용자가 입력한 데이터를 /edit 경로로 POST 방식으로 전송

form(action="/edit", method="post")

id 정보를 보내야 수정 가능. 보안상 id 정보를 숨기기 위해서 style을 넣는다.

input(type="text", name="id", class="form-control", value=data.\_id, style="display: none;")

input(name="title", value=data.title) 수정 전 데이터를 value로 넣는다.

textarea(name="content")= data.content 이전 데이터

input(type="date", value=data.date) 이전 데이터

수정 버튼 button(type=”submit”)수정

**11. routes/submit.js** :

**사용자가 이미지 파일을 첨부할 수 있도록 한다.**

‘/submit’ 경로에 사용될 라우터 객체. ‘/submit’ 경로에 대한 get, post 요청을 처리. express를 사용하여 이미지를 업로드하고, 업로드된 이미지와 함께 제목(캡션)을 받아 배열에 추가한 뒤, 웹페이지를 리다이렉트하는 라우터.

11-1. express 및 관련 모듈 설정

11-2. 업로드 된 이미지 저장 경로 및 이름 설정. Multer 미들웨어 초기화, 인스턴스 생성

이미지 저장될 디렉토리 설정

const storage = multer.diskStorage({

destination(req, file, done) {

done(null, './public/uploads/'); public 폴더의 uploads 폴더에 저장

},

파일 이름 설정(원래 파일이름 + 현재 시간)

Filename(req, file, done) {

const ext = path.extname(file.originalname) 파일 확장자 추출

파일 이름 설정 (원래 파일이름 + 현재 시간 + 확장자)

done(null, path.basename(file.originalname, ext) + Date.now() + ext);

},

});

const upload = multer({

storage: storage,

});

11-3. 업로드 된 파일의 경로와 제목을 저장할 빈 데이터 배열 data 생성

let data = []; // 파일 경로, 제목을 저장

11-4. ‘/’ 경로에 대한 get 요청 핸들러 정의. ‘/’ 경로에 대한 get 요청이 들어왔을 때 data를 전달, ‘submit’ 템플릿을 렌더링 하여 응답.

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

//get 요청시 submit 템플릿 랜더링, data 전달

res.render('submit', { data });

});

11-5. ‘/’ 경로에 대한 post 요청 핸들러 정의. ‘/’ 경로에 대한 post 요청이 들어왔을 때 요청(req)으로부터 이미지 캡션 제목, 이미지 경로 추출. 추출한 결과를 data 변수에 추가. 변형된 페이지를 보여주기 위해서 submit 페이지 리다이렉트

'/’ 경로에 대한 post 요청 핸들러 정의.

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

console.log(req.file, req.body.title);

요청으로부터 이미지 캡션(title) 추출

const title = req.body.title;

이미지 경로 추출.

file이 업로드 되면 /uploads/ + 파일이름(+확장자)를 imagePath 변수에 저장

const imagePath = req.file ? '/uploads/' + req.file.filename : null;

if(title && imagePath) {

data 배열에 추가 data.push({title, image: imagePath})

}

submit 페이지 리다이렉트. res.redirect('/submit')

});

11-6. 라우터 객체를 모듈로 내보내기

module.exports = router;

**12. routes/mypet.js**

: ‘/mypet’ 경로에 사용될 라우터 객체. '/mypet' 경로에 대한 GET 요청이 들어왔을 때 'pet' 템플릿을 렌더링하여 응답하는 역할

12-1. express 모듈 가져오기

const express = require('express');

12-2 express 라우터 객체를 생성

const router = express.Router();

12-3 ‘/’ 경로에 대한 get 요청 핸들러 정의. ‘/’ 경로에 대한 get 요청이 들어왔을 때 ‘pet’ 템플릿을 렌더링하여 응답

‘/’ 경로에 대한 get 요청이 들어왔을 때

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

‘pet’ 템플릿을 렌더링. res.render('pet');

});

12-4 라우터 객체를 모듈로 내보내기

module.exports = router;

**3. 소스코드**

-add.js

const ObjId = require('mongodb').ObjectId;

//express 모듈

const express = require('express');

//라우터 객체

const router = express.Router();

router.post("/", function (req, res) {

req.body.\_id = new ObjId(req.body.\_id);

const mydb = req.app.locals.mydb;

//$push 연산자: friends 배열에 req.body.new\_friend 값을 추가

mydb.collection('account')

.updateOne({ \_id: req.body.\_id }, {$push: { friends: req.body.new\_friend } })

.then((result) => {

console.log('친구 추가');

res.redirect('/');

})

.catch((err) => {

console.log(err);

})

});

//모듈로 내보내기

module.exports = router;

**-blog.js**

//express 모듈

const express = require('express');

//라우터 객체

const router = express.Router();

// ‘/’ 경로에 대한 get 요청 핸들러 정의

//비동기

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

//로그인을 안했으면

if(!req.session.user || req.session.user === undefined){

//로그인 페이지로 이동

res.redirect('/login');

return; // 함수 종료

}

//user객체

const user = req.session.user;

console.log(user.userid)

//해당 유저 아이디

const author = req.session.user.userid;

const mydb = req.app.locals.mydb;

let friends = [];

try {

const accountResult = await mydb.collection('account').findOne({ userid: req.session.user.userid });

friends = accountResult.friends || [];

const posts = await mydb.collection('post').find().toArray();

console.log(posts);

//'myblog' 템플릿을 렌더링

res.render('myblog', { user, friends, author, data: posts });

} catch (error) {

console.error('데이터를 가져오는 중 오류 발생:', error);

res.status(500).send('서버 오류');

}

});

//모듈로 내보내기

module.exports = router;

-content.js

const ObjId = require('mongodb').ObjectId;

//express 모듈

const express = require('express');

//라우터 객체

const router = express.Router();

router.get("/:id", function (req, res) {

const mydb = req.app.locals.mydb;

console.log(req.params.id);

let new\_id = new ObjId(req.params.id);

let userid;

//user객체

const user = req.session.user;

//로그인X

if(!(req.session.user)){

userid = "";

}

else{

//사용자 아이디 저장 변수

userid = req.session.user.userid;

}

if(userid === undefined){

userid = '';

}

mydb.collection('post').findOne({ \_id: new\_id }) //몽고db에서 id찾기

.then(result => {

console.log(result); // 넘겨주기

res.render('content', { user, userid, data : result });

}).catch(err =>{

console.log(err)

res.status(500).send();

});

});

//라우터 객체를 모듈로 내보내기

module.exports = router;

-delete.js

const ObjId = require('mongodb').ObjectId;

//express 모듈

const express = require('express');

//라우터 객체

const router = express.Router();

router.post("/", function (req, res) {

console.log(req.body);

req.body.\_id = new ObjId(req.body.\_id);

const mydb = req.app.locals.mydb;

//$pull 연산자: friends 배열에 req.body.new\_friend 값을 추가

mydb.collection('post').deleteOne(req.body)

.then((result) => {

console.log('친구 삭제');

res.redirect('/');

})

.catch((err) => {

console.log(err);

})

});

//모듈로 내보내기

module.exports = router;

**-edit.js**

const ObjId = require('mongodb').ObjectId;

//express 모듈

const express = require('express');

//라우터 객체

const router = express.Router();

router.get("/:id", function (req, res) {

const mydb = req.app.locals.mydb;

console.log(req.params.id);

let new\_id = new ObjId(req.params.id);

//user객체

const user = req.session.user;

mydb.collection('post').findOne({ \_id: new\_id }) //몽고db에서 해당 id 객체 찾기

.then(result => {

console.log(result);

res.render('edit.pug', { user, data : result }); // 넘겨주기

}).catch(err =>{

console.log(err)

res.status(500).send();

});

});

router.post('/', function(req, res){

const mydb = req.app.locals.mydb;

console.log(req.body.title);

console.log(req.body.content);

let userid = req.session.user.userid;

let new\_id = new ObjId(req.body.id); //edit.ejs의 43번쨰 줄에서 id로 넘겨받음

mydb.collection('post').updateOne({\_id: new\_id},

{$set: {title : req.body.title, content : req.body.content, author : userid, date : req.body.someDate}})

.then(result => {

console.log('데이터 수정 성공');

res.redirect('/blog');

});

});

//모듈로 내보내기

module.exports = router;

-enter.js

//express 모듈

const express = require('express');

const multer = require('multer');

const fs = require('fs');

const path = require('path');

//저장될 경로 설정

const storage = multer.diskStorage({

//destination 함수: 파일이 저장될 디렉토리 설정

destination(req, file, done) {

done(null, './public/uploads/');

},

//파일 이름 설정

filename(req, file, done) {

//파일 확장자 추출

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

// 파일 이름 설정 (원래 파일이름 + 현재 시간 + 확장자)

done(null, path.basename(file.originalname, ext) + Date.now() + ext);

},

});

const upload = multer({ storage: storage });

//라우터 객체

const router = express.Router();

//첨부된 문서의 내용 저장

let documentContent = '';

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

res.render('enter');

});

router.post('/save', function(req, res){

const mydb = req.app.locals.mydb;

let text\_content; //첨부 문서 내용 저장

// 호출되는 시간 기록

let now = new Date();

userid = req.session.user.userid;

if(documentContent){

text\_content = documentContent;

}

text\_content = text\_content+ '<br>' + req.body.content;

console.log(req.body.title);

console.log(req.body.content);

mydb.collection('post').insertOne(

{title : req.body.title, content : text\_content, author : userid, date : now.toLocaleDateString('KR')})

.then(result => {

console.log(result);

res.redirect('/blog');

});

});

router.post('/text', upload.single('document'), function(req, res){

console.log(req.file);

// 업로드된 파일이 있는지 확인

if (req.file) {

//이미지 경로

const filePath = req.file ? 'public/uploads/' + req.file.filename : null;

console.log(filePath);

documentContent = fs.readFileSync(filePath, 'utf-8');

console.log('File content:', documentContent);

} else {

// 파일이 제대로 업로드되지 않은 경우

console.log('Error: No file uploaded or file content is undefined.');

res.status(400).send('Error: No file uploaded or file content is undefined.');

}

});

//모듈로 내보내기

module.exports = router;

-f\_blog.js

const ObjId = require('mongodb').ObjectId;

//express 모듈

const express = require('express');

//라우터 객체

const router = express.Router();

// ‘/’ 경로에 대한 get 요청 핸들러 정의

//비동기

router.get('/:author', async (req, res) =>{

console.log(req.params.author);

let user;

//로그인X

if(!(req.session.user)){

user = null;

}

else{

//현재 user객체

user = req.session.user;

}

//내 친구 저장 변수

let myfriends = [];

//해당 유저(글쓴) 아이디

const author = req.params.author;

//글쓴이의 친구 저장 변수

let friends = [];

const mydb = req.app.locals.mydb;

let userResult = "";

try {

if(user != null){

//내 아이디의 객체 찾기

userResult = await mydb.collection('account').findOne({ userid: user.userid});

//내 친구 데이터 저장

myfriends = userResult.friends || [];

}

//글쓴이 아이디로 찾기

const accountResult = await mydb.collection('account').findOne({ userid: author});

//글쓴이의 친구 데이터 저장

friends = accountResult.friends || [];

const posts = await mydb.collection('post').find().toArray();

console.log(posts);

//'blog' 템플릿을 렌더링

res.render('myblog', { myfriends, user, userResult, friends, author, data: posts });

} catch (error) {

console.error('데이터를 가져오는 중 오류 발생:', error);

res.status(500).send('서버 오류');

}

});

//모듈로 내보내기

module.exports = router;

-f\_delete.js

const ObjId = require('mongodb').ObjectId;

//express 모듈

const express = require('express');

//라우터 객체

const router = express.Router();

router.post("/", function (req, res) {

console.log(req.body);

req.body.\_id = new ObjId(req.body.\_id);

const mydb = req.app.locals.mydb;

//$pull 연산자: friends 배열에 req.body.new\_friend 값을 추가

mydb.collection('account')

.updateOne({ \_id: req.body.\_id }, {$pull: { friends: req.body.delete\_friend } })

.then((result) => {

console.log('친구 삭제');

res.redirect('/');

})

.catch((err) => {

console.log(err);

})

});

//모듈로 내보내기

module.exports = router;

**-friends\_list.js**

//express 모듈

const express = require('express');

//라우터 객체

const router = express.Router();

//비동기

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

//로그인이 안되어 있는 경우

if(!req.session.user || req.session.user === undefined){

//로그인 페이지로 이동

res.redirect('/login');

return; // 함수 종료

}

const mydb = req.app.locals.mydb;

let friends = [];

//user객체

const user = req.session.user;

//해당 유저 아이디

const userid = req.session.user.userid;

try {

//내 아이디로 찾기

const userResult = await mydb.collection('account').findOne({ userid: user.userid});

//해당 유저 아이디

const account = await mydb.collection('account').findOne({ userid: req.session.user.userid });

friends = account.friends || [];

const accountResult = await mydb.collection('account').find().toArray();

console.log(accountResult);

//템플릿을 렌더링

res.render('friends', { user, friends, userResult, userid, data: accountResult });

} catch (error) {

console.error('데이터를 가져오는 중 오류 발생:', error);

res.status(500).send('서버 오류');

}

});

//모듈로 내보내기

module.exports = router;

**-home.js**

//express 모듈

const express = require('express');

//라우터 객체

const router = express.Router();

//비동기

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

const mydb = req.app.locals.mydb;

//메시지가 저장

const infomessage = req.flash('info');

let msg;

let friends = [];

let count = 0;

const user = req.session.user;

try {

// 로그아웃 상태

if (!user) {

msg = "로그아웃 상태입니다. \n로그인 하여 친구의 post를 확인하세요!";

} else {

// 몽고DB에서 userid 찾기

const result = await mydb.collection('account').findOne({ userid: req.session.user.userid });

//찾은 경우

if (result) {

friends = result.friends || []; // friends가 null이면 빈 배열로 초기화

} else {

friends = [];

}

}

// 'post' 컬렉션에서 데이터 가져오기

const posts = await mydb.collection('post').find().toArray();

// 렌더링 시 데이터 전달

res.render('home', { infomessage, user, msg, friends, data: posts, count });

} catch (error) {

console.error('데이터를 가져오는 중 오류 발생:', error);

res.status(500).send('서버 오류');

}

});

//모듈로 내보내기

module.exports = router;

**-login.js**

//express 모듈

const express = require('express');

//flash

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

//암호

const sha = require('sha256');

//라우터 객체

const router = express.Router();

// ‘/’ 경로에 대한 get 요청 핸들러 정의

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

console.log(req.session);

//req.sesson.user 데이터가 존재한다면

//로그인 상태

if(req.session.user){

console.log('세션유지');

//flash 메시지 설정

req.flash('info', '이미 로그인 되었습니다.');

//메인페이지로 이동

res.redirect('/');

}

else{

//아니면 로그인 페이지 렌더링

res.render('login');

}

});

//‘/’ 경로에 대한 post 요청 핸들러 정의

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

const mydb = req.app.locals.mydb; // app.locals에서 mydb 가져오기

console.log("아이디 : " + req.body.userid);

console.log("비밀번호 : " + req.body.userpw);

mydb.collection('account').findOne({userid : req.body.userid}) //몽고db에서 userid찾기

.then((result) => { //있으면

//비밀번호 일치 확인(암호화 했기 때문에 고려해서 비교)

if (result.userpw == sha(req.body.userpw)){

//계정정보를 req.sesson.user라는 이름의 세션에 추가

req.session.user = req.body;

console.log("새로운 로그인");

res.redirect('/');

}

else{

res.send('비밀번호가 틀렸습니다.')

}

})

});

//모듈로 내보내기

module.exports = router;

**-logout.js**

//express 모듈

const express = require('express');

//라우터 객체

const router = express.Router();

// ‘/’ 경로에 대한 get 요청 핸들러 정의

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

console.log('로그아웃');

req.session.destroy();

res.redirect('/');

});

//모듈로 내보내기

module.exports = router;

-mypet.js

//express 모듈

const express = require('express');

//라우터 객체

const router = express.Router();

// ‘/’ 경로에 대한 get 요청 핸들러 정의

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

//user객체

const user = req.session.user;

//get 요청시 ‘pet’ 템플릿을 렌더링

res.render('pet', {user});

});

//라우터 객체를 모듈로 내보내기

module.exports = router;

-signup.js

//express 모듈

const express = require('express');

//암호

const sha = require('sha256');

//라우터객체

const router = express.Router();

// - 기호를 전화번호 형식에 맞게 추가하는 함수

function formatPhone(num) {

// 숫자를 제외한 모든 문자 제거

num = num.replace(/[^0-9]/g, '');

// - 추가

num = num.replace(/^(\d{2,3})(\d{3,4})(\d{4})$/, `$1-$2-$3`);

return num;

}

// ‘/’ 경로에 대한 get 요청 핸들러 정의

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

//로그인 상태

if(req.session.user){

console.log('세션유지');

//flash 메시지 설정

req.flash('info', '이미 로그인 되었습니다.');

//메인페이지로 이동

res.redirect('/');

}

//로그인X

else{

//get 요청시 ‘form’ 템플릿(회원가입)을 렌더링

res.render('form');

}

});

//‘/’ 경로에 대한 post 요청 핸들러 정의

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

// app.locals에서 mydb 가져오기

const mydb = req.app.locals.mydb;

//요청으로부터 데이터 추출

const name = req.body.name;

const add = req.body.add;

const phone = formatPhone(req.body.phone); //전화번호 가공

const email = req.body.email;

const userid = req.body.userid;

const userpw = req.body.userpw;

//account에 정보 추가

mydb.collection("account").insertOne({

userid : userid,

userpw : sha(userpw), //비밀번호 암호화

name : name,

address : add,

phone : phone,

email : email,

friends : [],

})

.then((result) => {

console.log('회원가입 성공')

})

//table 템플릿 렌더링, 추출한 데이터를 전달

res.render('table', {name, add, phone, email, userid, userpw}); //pug 템플릿 파일을 사용

});

//라우터 객체를 모듈로 내보내기

module.exports = router;

**-submit.js**

const express = require('express');

const multer = require('multer');

const router = express.Router();

const path = require('path');

//이미지 저장될 경로 설정

const storage = multer.diskStorage({

//destination 함수: 이미지가 저장될 디렉토리 설정

destination(req, file, done) {

done(null, './public/uploads/'); // routes 폴더에서 올라가서 public/uploads/로 이동

},

//파일 이름 설정

filename(req, file, done) {

//파일 확장자 추출

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

// 파일 이름 설정 (원래 파일이름 + 현재 시간 + 확장자)

done(null, path.basename(file.originalname, ext) + Date.now() + ext);

},

});

//multer 미들웨어 설정, 생성

const upload = multer({

storage: storage,

});

//사진과 캡션 제목을 넣을 배열

let data = [];

// '/'경로에 대한 get 요청 핸들러

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

//user객체

const user = req.session.user;

//get 요청시 submit 템플릿 랜더링

res.render('submit', { user, data });

});

// '/’ 경로에 대한 post 요청 핸들러 정의.

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

console.log(req.file, req.body.title);

//이미지 캡션

const title = req.body.title;

//이미지 경로

const imagePath = req.file ? '/uploads/' + req.file.filename : null;

//data 배열에 추가

if(title && imagePath) {

data.push({title, image: imagePath})

}

//submit 페이지 리다이렉트

res.redirect('/submit')

});

//라우터 객체를 모듈로 내보내기

module.exports = router;

**-server.js**

const mongoclient = require("mongodb").MongoClient;

const ObjId = require('mongodb').ObjectId;

const url ='mongodb+srv://udmin:1213@cluster0.kdtozsc.mongodb.net/?retryWrites=true&w=majority' ;

let mydb;

mongoclient

.connect(url)

.then((client) => {

mydb = client.db('myblog');

app.locals.mydb = mydb; // app.locals에 저장

app.listen(3000, function () {

console.log("포트 3000으로 서버 대기중 ... ");

});

})

.catch((err) => {

console.log(err);

});

//express, 파일 처리를 위한 미들웨어..

const express = require('express');

const path = require('path');

const multer = require('multer');

const bodyParser = require('body-parser');

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

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

//secretKey 생성

const crypto = require('crypto');

const secretKey = crypto.randomBytes(32).toString('hex');

//라우터 모듈 import

const homeRouter = require('./routes/home');

const signupRouter = require('./routes/signup');

const blogRouter = require('./routes/blog');

const loginRouter = require('./routes/login');

const enterRouter = require('./routes/enter');

const editRouter = require('./routes/edit');

const conRouter = require('./routes/content');

const f\_blogRouter = require('./routes/f\_blog');

const friendRouter = require('./routes/friends\_list');

const logoutRouter = require('./routes/logout');

const addRouter = require('./routes/add');

const f\_dRouter = require('./routes/f\_delete');

const d\_Router = require('./routes/delete');

const subRouter = require('./routes/submit');

const mypetRouter = require('./routes/mypet');

//express 애플리케이션 생성

const app = express();

//pug 템플릿 엔진 설정

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

// 정적 파일 제공

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

app.use(flash());

app.use(express.json());

app.use(bodyParser.urlencoded({ extended: true }));

app.use(session({

secret: secretKey,

resave: false,

saveUninitialized: true,

}));

//각 라우터 등록

app.use('/', homeRouter);

app.use('/signup', signupRouter);

app.use('/blog', blogRouter)

app.use('/login', loginRouter)

app.use('/enter', enterRouter);

app.use("/edit", editRouter);

app.use("/content", conRouter);

app.use("/f\_blog", f\_blogRouter);

app.use("/friends\_list", friendRouter);

app.use("/logout", logoutRouter);

app.use("/add\_friend", addRouter);

app.use("/delete\_friend", f\_dRouter);

app.use("/delete", d\_Router);

app.use('/submit', subRouter)

app.use('/mypet', mypetRouter);

**-content.pug**

doctype html

html(lang="en")

head

meta(charset="UTF-8")

meta(name="viewport", content="width=device-width, initial-scale=1.0")

title Document

include menu.pug

link(rel='stylesheet', href='/css/style1.css')

link(rel="preconnect" href="https://fonts.googleapis.com")

link(rel="preconnect" href="https://fonts.gstatic.com" crossorigin)

link(href="https://fonts.googleapis.com/css2?family=Black+Han+Sans&display=swap" rel="stylesheet")

link(rel="preconnect" href="https://fonts.googleapis.com")

link(rel="preconnect" href="https://fonts.gstatic.com" crossorigin)

link(href="https://fonts.googleapis.com/css2?family=Orbit&display=swap" rel="stylesheet")

//부트스트랩 사용

link(href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css", rel="stylesheet", integrity="sha384-4bw+/aepP/YC94hEpVNVgiZdgIC5+VKNBQNGCHeKRQN+PtmoHDEXuppvnDJzQIu9", crossorigin="anonymous")

script(src="https://code.jquery.com/jquery-3.6.4.min.js")

script(src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js")

body

.container.mt-4

.card(style="width: 18rem;")

.card-body

h5.card-title= data.title

h6.card-subtitle.mb-2.text-body-secondary= data.date

p.card-text= data.content

//글쓴이만 수정가능!

if (data.author == userid)

a.card-link(href=`/edit/${data.\_id}`) 수정하기

**-edit.pug**

doctype html

html(lang="en")

head

meta(charset="UTF-8")

meta(name="viewport", content="width=device-width, initial-scale=1.0")

title Document

include menu.pug

link(rel='stylesheet', href='/css/style1.css')

link(rel="preconnect" href="https://fonts.googleapis.com")

link(rel="preconnect" href="https://fonts.gstatic.com" crossorigin)

link(href="https://fonts.googleapis.com/css2?family=Black+Han+Sans&display=swap" rel="stylesheet")

link(rel="preconnect" href="https://fonts.googleapis.com")

link(rel="preconnect" href="https://fonts.gstatic.com" crossorigin)

link(href="https://fonts.googleapis.com/css2?family=Orbit&display=swap" rel="stylesheet")

//부트스트랩 사용

link(href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css", rel="stylesheet", integrity="sha384-4bw+/aepP/YC94hEpVNVgiZdgIC5+VKNBQNGCHeKRQN+PtmoHDEXuppvnDJzQIu9", crossorigin="anonymous")

script(src="https://code.jquery.com/jquery-3.6.4.min.js")

script(src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js")

body

h1 글 수정하기

.container.mt-4

form(action="/edit", method="post")

div.form-group

label(for="id") 제목

// 수정이기 때문에 이전의(원래) 값이 들어가 있어야 됨

// id 정보를 넘겨줘야 한다. -> id 정보를 숨겨놔야됨. 보안으로 style을 넣는다.

input(type="text", name="id", class="form-control", value=data.\_id, style="display: none;")

input(type="text", name="title", class="form-control", value=data.title)

p

div.form-group

label(for="content") 내용

textarea(name="content", rows='10', class="form-control")= data.content

p

div.form-group

label(for="someDate") 작성일

input(type="date", class="form-control", name="someDate", value=data.date)

p

button(type="submit", class="btn btn-warning", style="float:right") 수정

-enter.pug

doctype html

html(lang="en")

head

meta(charset="UTF-8")

meta(name="viewport", content="width=device-width, initial-scale=1.0")

title Document

include menu.pug

link(rel='stylesheet', href='/css/style1.css')

link(rel="preconnect" href="https://fonts.googleapis.com")

link(rel="preconnect" href="https://fonts.gstatic.com" crossorigin)

link(href="https://fonts.googleapis.com/css2?family=Black+Han+Sans&display=swap" rel="stylesheet")

link(rel="preconnect" href="https://fonts.googleapis.com")

link(rel="preconnect" href="https://fonts.gstatic.com" crossorigin)

link(href="https://fonts.googleapis.com/css2?family=Orbit&display=swap" rel="stylesheet")

//부트스트랩 사용

link(href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css", rel="stylesheet", integrity="sha384-4bw+/aepP/YC94hEpVNVgiZdgIC5+VKNBQNGCHeKRQN+PtmoHDEXuppvnDJzQIu9", crossorigin="anonymous")

script(src="https://code.jquery.com/jquery-3.6.4.min.js")

script(src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js")

body

div.titlename.text-center

h1 글 작성하기

.container.mt-4

form(action="/enter/save", method="post")

div.form-group

label(for="id") 제목

input(type="text", name="title", class="form-control")

p

div.form-group

label(for="content") 내용

textarea(name="content", rows='10', class="form-control")

p

button(type="submit", class="btn btn-warning", style="float:right") 작성

.container.mt-3

form(action="/enter/text", method="post", enctype = "multipart/form-data")

//upload.single('document')과 name 일치

input(type="file", name="document", accept=".txt" class="btn btn-outline-primary")

button(type="submit", class="btn btn-outline-primary", style="float:right") 첨부

**-form.pug**

//form.pug : 회원가입 폼

doctype html

html(lang="en")

head

meta(charset="UTF-8")

meta(name="viewport", content="width=device-width, initial-scale=1.0")

title form

include menu.pug

link(rel='stylesheet', href='/css/style2.css')

link(rel="preconnect" href="https://fonts.googleapis.com")

link(rel="preconnect" href="https://fonts.gstatic.com" crossorigin)

link(href="https://fonts.googleapis.com/css2?family=Black+Han+Sans&display=swap" rel="stylesheet")

link(rel="preconnect" href="https://fonts.googleapis.com")

link(rel="preconnect" href="https://fonts.gstatic.com" crossorigin)

link(href="https://fonts.googleapis.com/css2?family=Orbit&display=swap" rel="stylesheet")

//부트스트랩 사용

link(href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css", rel="stylesheet", integrity="sha384-4bw+/aepP/YC94hEpVNVgiZdgIC5+VKNBQNGCHeKRQN+PtmoHDEXuppvnDJzQIu9", crossorigin="anonymous")

script(src="https://code.jquery.com/jquery-3.6.4.min.js")

script(src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js")

body

//이미 로그인됨

//메시지가 존재하는 경우에만 출력하고 alert를 사용하여 메시지를 띄움

if infomessage && Object.keys(infomessage).length > 0

script.

alert('#{infomessage}');

//설명

div.titlename

h1.display-4.mb-4 회원가입을 위해 <br>

| 정보를 입력해 주세요

//form 작성 부분

div.form

form(action="/signup" method="post")

legend.info show your info

.mb-3

//이름

label(for="name").form-label Name

input.form-control(type="text", id="name", name="name" required)

.mb-3

//아이디

label(for="name").form-label Id

input.form-control(type="text", id="userid", name="userid" required)

.mb-3

//비밀번호

label(for="name").form-label Password

input.form-control(type="text", id="userpw", name="userpw" required)

.mb-3

//주소

label(for="add").form-label Address

input.form-control(type="text", id="add", name="add" required)

.mb-3

//전화번호

label(for="phone").form-label Phone

input.form-control(type="text", id="phone", name="phone" required)

.mb-3

//이메일

label(for="email").form-label Email

input.form-control(type="email", id="email" name="email" required)

//개인정보 수집

.mb-3.form-check

input.form-check-input(type="checkbox", id="Check" required)

label.form-check-label(for="Check") 이용약관 개인정보 수집 및 정보이용에 동의합니다.

//확인 메시지

.mb-3.form-check

input.form-check-input(type="checkbox", id="Check" required)

label.form-check-label(for="Check") 이름과 전화번호를 정확하게 적었는지 꼭 확인해 주세요!

//제출 버튼

button.btn.btn-primary(type="submit") 가입

**-friends.pug**

doctype html

html(lang="en")

head

meta(charset="UTF-8")

meta(name="viewport", content="width=device-width, initial-scale=1.0")

title Document

include menu.pug

link(rel='stylesheet', href='/css/style5.css')

link(rel="preconnect" href="https://fonts.googleapis.com")

link(rel="preconnect" href="https://fonts.gstatic.com" crossorigin)

link(href="https://fonts.googleapis.com/css2?family=Black+Han+Sans&display=swap" rel="stylesheet")

link(rel="preconnect" href="https://fonts.googleapis.com")

link(rel="preconnect" href="https://fonts.gstatic.com" crossorigin)

link(href="https://fonts.googleapis.com/css2?family=Orbit&display=swap" rel="stylesheet")

//부트스트랩 사용

link(href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css", rel="stylesheet", integrity="sha384-4bw+/aepP/YC94hEpVNVgiZdgIC5+VKNBQNGCHeKRQN+PtmoHDEXuppvnDJzQIu9", crossorigin="anonymous")

script(src="https://code.jquery.com/jquery-3.6.4.min.js")

script(src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js")

body

//제목

div.titlename

p 친구 리스트

hr

div.background-container

//친구

div.information

p=userid+ '님의 친구리스트'

//친구 리스트 테이블

.table\_form

table.table

thead

tr

th(scope="col") 아이디

th 블로그

tbody

each name in friends

tr

td= name

td.blog\_link: a(href=href=`/f\_blog/${name}`) #{name}님의 blog

//친구삭제 버튼

td: button.f\_delete.btn.btn-outline-danger(data-id=userResult.\_id, data-new=name) 친구 삭제

hr

div.background-container

//친구

div.information

p=userid+ '추가하고 싶으면 해라'

//친구 리스트 테이블

.table\_form

table.table

thead

tr

th(scope="col") 아이디

th 블로그

tbody

each item in data

tr

//내 account가 아니고 친구목록에 없는 경우

if userid != item.userid && !(friends.includes(item.userid))

td= item.userid

td.blog\_link: a(href=href=`/f\_blog/${name}`) #{name}님의 blog

//친구추가 버튼

td: button.add.btn.btn-outline-danger(data-id=userResult.\_id, data-new=item.userid) 친구추가

script(src="https://code.jquery.com/jquery-3.6.4.min.js")

script.

//버튼 클릭 함수 정의

$('.add').click(function(e) {

let sid = e.target.dataset.id;

let new\_f = e.target.dataset.new;

let item = $(this);

$.ajax({

type: 'post',

url: '/add\_friend',

data: {\_id: sid, new\_friend: new\_f}

}).done(function(result) {

// 응답에 성공했을 때

location.reload();

}).fail(function(xhr, textStatus, errorThrown) {

// 응답에 실패했을 때

console.log('친구 추가 실패');

console.log(xhr, textStatus, errorThrown);

});

});

//버튼 클릭 함수 정의

$('.f\_delete').click(function(e) {

console.log('삭제');

let sid = e.target.dataset.id;

let f\_delete = e.target.dataset.new;

let item = $(this);

$.ajax({

type: 'post',

url: '/delete\_friend',

data: {\_id: sid, delete\_friend: f\_delete}

}).done(function(result) {

// 응답에 성공했을 때

location.reload();

}).fail(function(xhr, textStatus, errorThrown) {

// 응답에 실패했을 때

console.log('친구 삭제 실패');

console.log(xhr, textStatus, errorThrown);

});

});

**-home.pug**

//home.pug : 멸종 위기종 설명

doctype html

html(lang="en")

head

meta(charset="UTF-8")

meta(name="viewport", content="width=device-width, initial-scale=1.0")

title home

include menu.pug

link(rel='stylesheet', href='/css/style6.css')

link(rel="preconnect" href="https://fonts.googleapis.com")

link(rel="preconnect" href="https://fonts.gstatic.com" crossorigin)

link(href="https://fonts.googleapis.com/css2?family=Black+Han+Sans&display=swap" rel="stylesheet")

link(rel="preconnect" href="https://fonts.googleapis.com")

link(rel="preconnect" href="https://fonts.gstatic.com" crossorigin)

link(href="https://fonts.googleapis.com/css2?family=Orbit&display=swap" rel="stylesheet")

//부트스트랩 사용

link(href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css", rel="stylesheet", integrity="sha384-4bw+/aepP/YC94hEpVNVgiZdgIC5+VKNBQNGCHeKRQN+PtmoHDEXuppvnDJzQIu9", crossorigin="anonymous")

script(src="https://code.jquery.com/jquery-3.6.4.min.js")

script(src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js")

body

//이미 로그인됨

//메시지가 존재하는 경우에만 출력하고 alert를 사용하여 메시지를 띄움

if infomessage && Object.keys(infomessage).length > 0

script.

alert('#{infomessage}');

//시작 페이지

.mb-4

div.titlename.text-center

h1.display-4.mb-4 BLOG

.info

.mb-3

strong 핫토픽 > 멸종 위기 동물

hr

.images-container

figure

img(src="/images/시베리아흰두루미.jpg", alt="시베리아흰두루미 사진")

figcaption 시베리아흰두루미

figure

img(src="/images/스라소니.jpg", alt="스라소니 사진")

figcaption 스라소니

figure

img(src="/images/여우.jpg", alt="여우 사진")

figcaption 여우

figure

img(src="/images/삵.jpg", alt="삵 사진")

figcaption 삵

.total\_post

.mb-3

strong post

//post 테이블

.table\_form

table.table

thead

tr

th 글 제목

th 글쓴이

th 작성일

tbody

//전체 post중 5개의 post만 보여준다.

each item in data.slice(0,5)

tr

//해당 글 페이지로 이동

td: a(href=`/content/${item.\_id}`) #{item.title}

//해당 글쓴이의 블로그로 이동

td: a(href=`/f\_blog/${item.author}`) #{item.author}

td= item.date

.friends\_post

.mb-3

strong 친구들의 post

hr

//로그아웃 상태

if msg

strong= msg

else

//친구가 빈 문자열, 친구가 없는 경우

if (friends.length === 0)

strong 아직 친구가 없습니다 친구를 추가하세요

else

.table\_form

table.table

thead

tr

.mb-6

th 글 제목

th 글쓴이

th 작성일

tbody

//최신부터 출력하기 위해 거꾸로

each name in friends.reverse()

- count = 0;

each item in data

if item.author === name && count < 3

tr

td: a(href=`/content/${item.\_id}`) #{item.title}

td: a(href=`/f\_blog/${item.author}`) #{item.author}

td= item.date

- count++;

**-login.pug**

//form.pug : 회원가입 폼

doctype html

html(lang="en")

head

meta(charset="UTF-8")

meta(name="viewport", content="width=device-width, initial-scale=1.0")

title form

include menu.pug

link(rel='stylesheet', href='/css/style2.css')

link(rel="preconnect" href="https://fonts.googleapis.com")

link(rel="preconnect" href="https://fonts.gstatic.com" crossorigin)

link(href="https://fonts.googleapis.com/css2?family=Black+Han+Sans&display=swap" rel="stylesheet")

link(rel="preconnect" href="https://fonts.googleapis.com")

link(rel="preconnect" href="https://fonts.gstatic.com" crossorigin)

link(href="https://fonts.googleapis.com/css2?family=Orbit&display=swap" rel="stylesheet")

//부트스트랩 사용

link(href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css", rel="stylesheet", integrity="sha384-4bw+/aepP/YC94hEpVNVgiZdgIC5+VKNBQNGCHeKRQN+PtmoHDEXuppvnDJzQIu9", crossorigin="anonymous")

script(src="https://code.jquery.com/jquery-3.6.4.min.js")

script(src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js")

body

//설명

div.titlename

h1.display-4.mb-4 로그인을 위해 <br>

| 정보를 입력해 주세요

div.form

form(action="/login" method="post")

legend.info show your info

.mb-3

//아이디

label(for="userid").form-label Id

input.form-control(type="text", id="userid", name="userid" required)

.mb-3

//비밀번호 type password로 수정

label(for="userpw").form-label Password

input.form-control(type="password", id="userpw", name="userpw" required)

//제출 버튼

button.btn.btn-primary(type="submit") Submit

**-menu.pug**

// 네비게이션 부분

nav.navbar.navbar-expand-lg.bg-body-tertiary

.container-fluid

a.navbar-brand(href='/') Menu

button.navbar-toggler(type='button', data-bs-toggle='collapse', data-bs-target='#navbarSupportedContent', aria-controls='navbarSupportedContent', aria-expanded='false', aria-label='Toggle navigation')

span.navbar-toggler-icon

.collapse.navbar-collapse#navbarSupportedContent

ul.navbar-nav.me-auto.mb-2.mb-lg-0

li.nav-item

// 첫 번째 home 페이지. active: 현재 페이지, 활성 링크 표시

a.nav-link.active(aria-current='page', href='/') Home

li.nav-item

//login 페이지 active: 현재 페이지, 활성 링크 표시

a.nav-link.active(aria-current='page', href='/login') Login

li.nav-item

// form 페이지

a.nav-link(href='/signup') Sign up

li.nav-item

// myblog 페이지

a.nav-link(href='/blog') My blog

li.nav-item.dropdown

a.nav-link.dropdown-toggle(href='#', role='button', data-bs-toggle='dropdown', aria-expanded='false') Something else here

ul.dropdown-menu

li

// 친구추가

a.dropdown-item(href='/friends\_list') 친구목록

li

// 멸종 위기종 추가하는 페이지

a.dropdown-item(href='/submit') 멸종위기종 추가

li

// 메뉴 내에서 메뉴 항목 간에 수평 구분선을 생성

hr.dropdown-divider

li

// 강아지 소개 페이지

a.dropdown-item(href='/mypet') 반려동물

form.d-flex()

if user

a(href="/logout") Logout

**-myblog.pug**

doctype html

html(lang="en")

head

meta(charset="UTF-8")

meta(name="viewport", content="width=device-width, initial-scale=1.0")

title myblog

include menu.pug

link(rel='stylesheet', href='/css/style1.css')

link(rel="preconnect" href="https://fonts.googleapis.com")

link(rel="preconnect" href="https://fonts.gstatic.com" crossorigin)

link(href="https://fonts.googleapis.com/css2?family=Black+Han+Sans&display=swap" rel="stylesheet")

link(rel="preconnect" href="https://fonts.googleapis.com")

link(rel="preconnect" href="https://fonts.gstatic.com" crossorigin)

link(href="https://fonts.googleapis.com/css2?family=Orbit&display=swap" rel="stylesheet")

//부트스트랩 사용

link(href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css", rel="stylesheet", integrity="sha384-4bw+/aepP/YC94hEpVNVgiZdgIC5+VKNBQNGCHeKRQN+PtmoHDEXuppvnDJzQIu9", crossorigin="anonymous")

script(src="https://code.jquery.com/jquery-3.6.4.min.js")

script(src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js")

body

div.titlename.text-center

//내 블로그인 경우

if user !=null && user.userid == author

h1.display-4.mb-4 My Blog

else

h1.display-4.mb-4= author + ' 님 블로그'

.row

.col-md-12

.row

.col-md-3

.mb-3.text-center

strong 친구 목록

//친구가 빈 문자열, 친구가 없는 경우

if (friends.length === 0)

strong 아직 친구가 없습니다

else

table.table.table-sm

thead

tr

th 아이디

tbody

each name in friends

tr

td= name

.col-md-8

.jumbotron

h2 글 목록

.table\_form

table.table

thead

tr

th(scope="col") 글 제목

th(scope="col") 작성일

tbody

each item in data

//post에서 해당 글쓴이가 쓴 경우만

if(author == item.author)

tr

td: a(href=`/content/${item.\_id}`) #{item.title}

td= item.date

//해당 유저인 경우만 삭제

if user != null && user.userid == author

//삭제 버튼

td: button.delete.btn.btn-outline-danger(data-id=item.\_id) 삭제

//해당 유저인 경우만 글쓰기 가능

if user != null && user.userid == author

button.write.btn.btn-primary(type="submit") 글쓰기

//아닌 경우 중 친구목록에 없는 경우

if user != null && user.userid != author && !(myfriends.includes(author))

//친구추가 버튼

td: button.add.btn.btn-outline-danger(data-id=userResult.\_id, data-new=author) 친구추가

ol-md-2i

script(src="https://code.jquery.com/jquery-3.6.4.min.js")

script.

//글쓰기 버튼 클릭 함수 정의

$('.write').click(function(e) {

console.log('추가!');

$.ajax({

type: 'get',

url: '/enter',

}).done(function(result) {

location.href = '/enter';

}).fail(function(xhr, textStatus, errorThrown) {

// 응답에 실패했을 때

console.log('글쓰기 실패');

console.log(xhr, textStatus, errorThrown);

});

});

//친구추가 버튼 클릭 함수 정의

$('.add').click(function(e) {

console.log('추가!');

let sid = e.target.dataset.id;

let new\_f = e.target.dataset.new;

let item = $(this);

$.ajax({

type: 'post',

url: '/add\_friend',

data: {\_id: sid, new\_friend: new\_f}

}).done(function(result) {

// 응답에 성공했을 때

location.reload();

}).fail(function(xhr, textStatus, errorThrown) {

// 응답에 실패했을 때

console.log('친구 추가 실패');

console.log(xhr, textStatus, errorThrown);

});

});

//삭제 버튼 클릭 함수 정의

$('.delete').click(function(e) {

console.log('추가!');

let sid = e.target.dataset.id;

let item = $(this);

$.ajax({

type: 'post',

url: '/delete',

data: {\_id: sid}

}).done(function(result) {

// 응답에 성공했을 때

item.parent('td').parent('tr').remove();

location.reload();

}).fail(function(xhr, textStatus, errorThrown) {

// 응답에 실패했을 때

console.log('삭제 실패');

console.log(xhr, textStatus, errorThrown);

});

});

**-pet.pug**

//pet.pug : 반려동물을 소개

doctype html

html(lang="en")

head

meta(charset="UTF-8")

meta(name="viewport", content="width=device-width, initial-scale=1.0")

title My Pet

include menu.pug

link(rel='stylesheet', href='/css/style3.css')

link(rel="preconnect" href="https://fonts.googleapis.com")

link(rel="preconnect" href="https://fonts.gstatic.com" crossorigin)

link(href="https://fonts.googleapis.com/css2?family=Black+Han+Sans&display=swap" rel="stylesheet")

link(rel="preconnect" href="https://fonts.googleapis.com")

link(rel="preconnect" href="https://fonts.gstatic.com" crossorigin)

link(href="https://fonts.googleapis.com/css2?family=Orbit&display=swap" rel="stylesheet")

//부트스트랩 사용

link(href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css", rel="stylesheet", integrity="sha384-4bw+/aepP/YC94hEpVNVgiZdgIC5+VKNBQNGCHeKRQN+PtmoHDEXuppvnDJzQIu9", crossorigin="anonymous")

script(src="https://code.jquery.com/jquery-3.6.4.min.js")

script(src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js")

body

//강아지

div.titlename.text-center

h1.display-4.mb-4 My Dog

//강아지 프로필

div.pet

//강아지 사진

img.profile(src="/images/dog1.jpg", alt="강아지 사진1")

//강아지 설명 테이블

div.info\_table

table.table

tbody

tr.name

th 이름

td <strong>정수동

tr

th 종

td 장모 치와와

tr

th 생년월일

td 2018/09/01

tr

th 성별

td 남

tr

th 성격

td 소심

//강아지 특징 설명

div.summary

.mb-3

strong 개요

hr

p.text

| 털이 긴 장모 치와와이다. 코가 좁고 길쭉하며, 꼬리는 북실하고 끝은 흰색이며 짧다. <br>

| 주둥이는 뾰족하고 흰색, 등의 전면에는 황갈색 반점이 있다. <br>

| 털이 길어 너무 많이 빠지기 떄문에 주기적인 관리가 필수이다. <br>

| 사료를 주식으로 먹지만 간식을 더 좋아하고 대부분의 음식을 선호한다. <br>

| 특히 사과를 좋아함. 소리에 민감하게 반응하고 산책을 좋아한다. <br>

//강아지 사진

div.pic

.mb-3

div.pic\_name

strong 사진

hr

.images-container

img(src="/images/dog2.jpg", alt="강아지 사진2")

img(src="/images/dog3.jpg", alt="강아지 사진3")

img(src="/images/dog4.jpg", alt="강아지 사진4")

**-submit.pug**

//submit.pug : 멸종 위기종 사진과 이름을 추가

doctype html

html(lang="en")

head

meta(charset="UTF-8")

meta(name="viewport", content="width=device-width, initial-scale=1.0")

title 멸종위기동물 추가하기

include menu.pug

link(rel='stylesheet', href='/css/style4.css')

link(rel="preconnect" href="https://fonts.googleapis.com")

link(rel="preconnect" href="https://fonts.gstatic.com" crossorigin)

link(href="https://fonts.googleapis.com/css2?family=Black+Han+Sans&display=swap" rel="stylesheet")

link(rel="preconnect" href="https://fonts.googleapis.com")

link(rel="preconnect" href="https://fonts.gstatic.com" crossorigin)

link(href="https://fonts.googleapis.com/css2?family=Orbit&display=swap" rel="stylesheet")

//부트스트랩 사용

link(href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css", rel="stylesheet", integrity="sha384-4bw+/aepP/YC94hEpVNVgiZdgIC5+VKNBQNGCHeKRQN+PtmoHDEXuppvnDJzQIu9", crossorigin="anonymous")

script(src="https://code.jquery.com/jquery-3.6.4.min.js")

script(src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js")

body

//멸종 위기 동물 리스트

div.info

.mb-3

strong 멸종 위기동물

hr

.images-container

figure

img(src="/images/시베리아흰두루미.jpg", alt="시베리아흰두루미 사진")

figcaption 시베리아흰두루미

figure

img(src="/images/스라소니.jpg", alt="스라소니 사진")

figcaption 스라소니

figure

img(src="/images/여우.jpg", alt="여우 사진")

figcaption 여우

figure

img(src="/images/삵.jpg", alt="삵 사진")

figcaption 삵

each item in data

figure

img(src= item.image, alt="업로드한 사진")

figcaption= item.title

hr

//멸종 위기 동물을 추가하는 폼

div.form

form(action="/submit" method="post" enctype="multipart/form-data")

legend.add 멸종 위기동물 추가하기

.mb-4

//제목(이름)

label(for="title").form-label Title

input.form-control(type="text", id="title", name="title" required)

.mb-4

//사진

label(for="image").form-label Image

input.form-control(type="file", id="image", name="image" required)

button(type="submit") 업로드

**-table.pug**

//table.pug : 회원 가입 후 작성한 회원 정보 테이블

doctype html

html(lang="en")

head

meta(charset="UTF-8")

meta(name="viewport", content="width=device-width, initial-scale=1.0")

title table

include menu.pug

link(rel='stylesheet', href='/css/style5.css')

link(rel="preconnect" href="https://fonts.googleapis.com")

link(rel="preconnect" href="https://fonts.gstatic.com" crossorigin)

link(href="https://fonts.googleapis.com/css2?family=Black+Han+Sans&display=swap" rel="stylesheet")

link(rel="preconnect" href="https://fonts.googleapis.com")

link(rel="preconnect" href="https://fonts.gstatic.com" crossorigin)

link(href="https://fonts.googleapis.com/css2?family=Orbit&display=swap" rel="stylesheet")

//부트스트랩 사용

link(href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css", rel="stylesheet", integrity="sha384-4bw+/aepP/YC94hEpVNVgiZdgIC5+VKNBQNGCHeKRQN+PtmoHDEXuppvnDJzQIu9", crossorigin="anonymous")

script(src="https://code.jquery.com/jquery-3.6.4.min.js")

script(src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js")

body

//제목

div.titlename

p 회원가입이 완료되었습니다.

hr

div.background-container

//이름

div.information

//이름 변수

p=name+ '님의 회원정보'

//회원 정보 테이블

.table\_form

table.table

thead

tr

th(scope="col") 이름

th(scope="col") 정보

th(scope="col") 값

tbody

tr

//이름 변수

td(rowspan = "5")= name

td Phone

//전화번호 변수

td= phone

tr

td Address

//주소 변수

td= add

tr

//이메일 변수

td Email

td= email

tr

//id 변수

td Id

td= userid

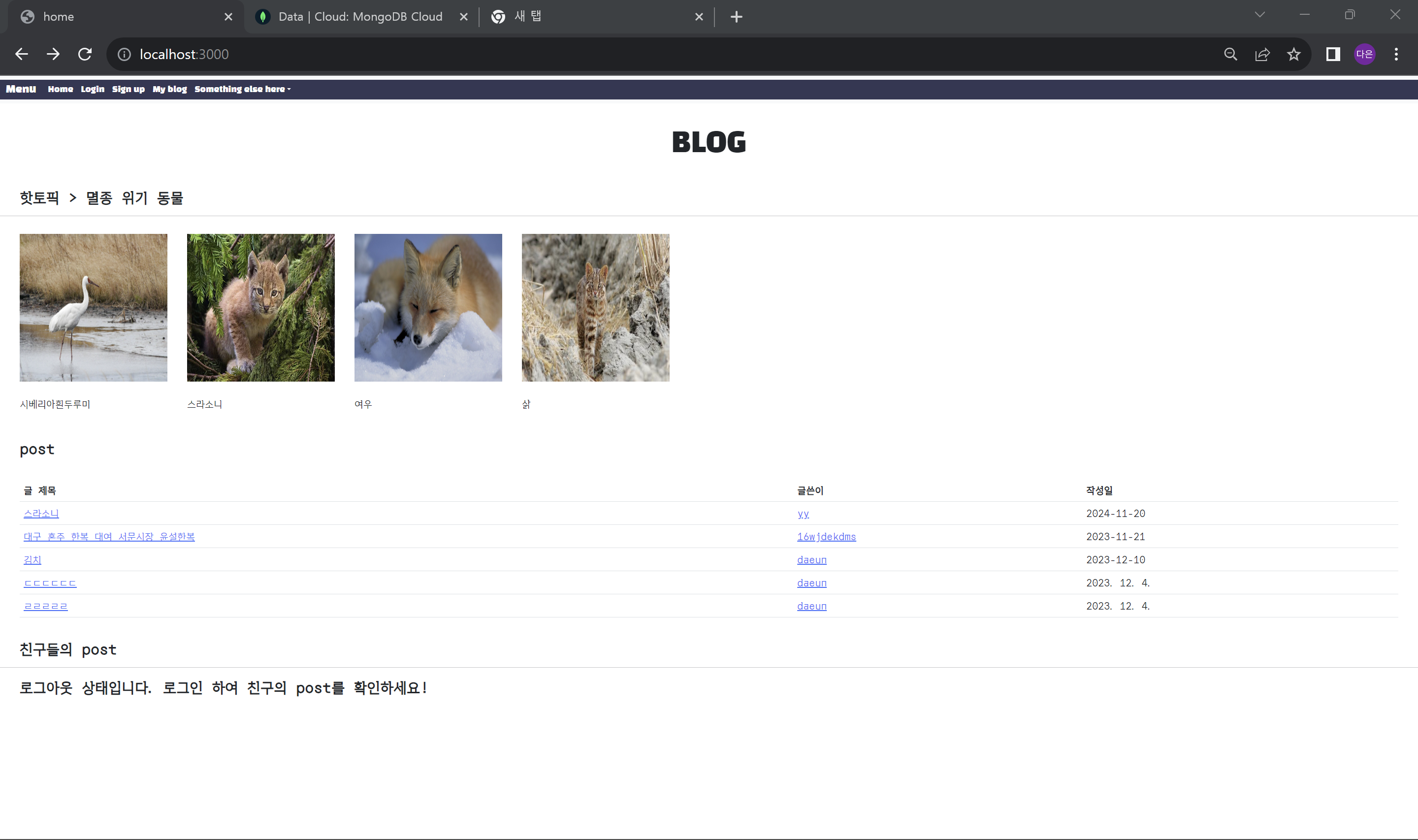
tr

//password 변수

td Password

td= userpw

**4. 실행화면**



텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

자동 생성된 설명