package.json

 "scripts": {

  "start":"node index.js"

*/\*터미널에 작성한 명령어를 여기에 작성할 수 있음 달라진 점은 계속해서 node index.js라고 하지 않아도 되고 npm start라고 명령어를 주어도 괜찮음.*

*\*/*

“ "dependencies": // 프로그램이 실행되기위해 필요한 것.

2.4-

브라우져와 웹사이트는 Post Request로 정보를 전달한다. 영상에 코멘트를 달면 post 웅앵이 실행하게 됨. 즉 우리가 적합한 http method를 사용함.

Get request 🡺URL에 접근

URL을 쓴다(youtube.com)🡺브라우저가 Get method를 실행한다 🡺 브라우저가 URL의 페이지를 읽어온다.

웹사이트에 Login 한다. 🡺브라우저가 POST method를 실행한다 🡺 브라우저가 웹사이트에 정보를 전달한다.

서버가 응답하게 다른 사람이 접속할 수 있도록 하는 것.

const express = require('express'); *// node module을 어딘가에 가지고 옴  express를 내 파일들 속에서 찾으려고 함.  아님 없으면 node-modul에서 찾으려고 한다.*

const app = express();

const PORT=4000;

function handleListening(){

    console.log(`Listening on: http://localhost:${PORT}`); *// '와 `는 다르다!*

}

function handleHome(req,res){*//request obj response obj 누가 홈페이지를 접속하고 어떤 종류의 데이터가  페이지로 전송됐는지.. 그럴 때 req를 사용.. 아이디와 패스워드를 전송한다면 req에서 정보를 얻을 수 있다.*

    console.log(req);

    res.send("hello baby");

}

function handleProfile(req,res){

    res.send("You are on my profile");

}

app.get("/",handleHome); *//누군가가 app에 접속하게 되면 handlehome을 부름*

app.get("/profile",handleProfile);

app.listen(PORT,handleListening); *// 4000개의 포트를 들으라!! app아 !!*

2.5

Babel ? 최신의 JS코드를 예전 JS코드로 전환해준다.

const something =(a,b)=>a\*b --🡪 var something = function something(a,b){return a\*b}

npm install @babel/node로 설치 해준다.

또한 많은 stage를 가지고 있음.

stage 0는 완전 실험적인 걸 얻게 될 것임.

우리가 사용할 것은 -env

<https://babeljs.io/docs/en/babel-preset-env>

babel을 적용하고 싶으면 package.json 파일에서 “start” : “babel-node index.js” 라고 바꿔준다.

그럼 babel 코드로 바뀌게 된다.

nodeman을 설치한 이유- 새로고침하고 저장하고 멈췄다가 다시 켜고.. 등등의 처리과정을 거치지 않아도 되기 때문.

2.6

middleware 란? 어려운 express 내용 처리가 끝날 때까지 기다리는 것.

어떻게 연결? 웹사이트 연결 -> route를 봄 아 !!! home으로 연결하는 구나!! -> handlehome을 실행 -> betweenHome이 middleware가 된다.

app.get("/",betweenHome,handleHome);

유저와 마지막 응답 사이에 무언ㄴ가 있어야 하는데 이를 middleware라고 함.

어떻게 middleware를 만든는지? 걍 중간에 함수 끼어놓아도 ㄱㅊ

그렇게 만들고 구글 크롬으로부터 온 요청을 계속 처리할 지에 대해 우리가 권한을 줘야함.

이는 어디에 사용되냐면 사용자의 로그인 현황을 알 수 있고 파일을 전송할 때 중간에서 가로챌 수 있다.

middleware는 오직 route에만 사용됨

app.get("/",handleHome); *//누군가가 app에 접속하게 되면 handlehome을 부름*

app.get("/profile",handleProfile);

app.listen(PORT,handleListening); *// 4000개의 포트를 들으라!! app아 !!*

*}\*/*

app.use(betweenHome);

app.get("/",handleHome); *//누군가가 app에 접속하게 되면 handlehome을 부름*

app.get("/profile",handleProfile);

app.listen(PORT,handleListening);

*import* express *from* "express"; *// const express = require('express');를 최신버전으로 바꿀 수 있다.*

*import* "core-js";

*// node module을 어딘가에 가지고 옴  express를 내 파일들 속에서 찾으려고 함.  아님 없으면 node-modul에서 찾으려고 한다.*

const app = express();

const PORT=4000;

const handleListening=()=> console.log(`Listening on: http://localhost:${PORT}`); *// '와 `는 다르다!*

*//섹시한 방향으로 function을 사용하지않고 const를 이용해서 바꿀 수 있다.*

const handleHome=(req,res)=>res.send("hello baby");

const handleProfile =(req,res)=> res.send("You are on my profile");

const betweenHome=(req,res,next)=>{

    console.log("between");

    next();

};

*/\*function handleProfile(req,res){*

*res.send("You are on my profile");*

*}\*/*

app.use(betweenHome);

app.get("/",handleHome); *//누군가가 app에 접속하게 되면 handlehome을 부름*

app.get("/profile",handleProfile);

app.listen(PORT,handleListening); *// 4000개의 포트를 들으라!! app아 !!*

2.7

추가적인 middleware

morgan – logging 정보를 줌 (어떤 일이 일어났는지 기록하는 것)

app.use(morgan("tiny"));

profile에 들어가면 로깅한 기록이 뜨게된다.

helmet- node.js 앱의 ㅂ보안을 안전하게 만들어 줌.

이 밖에도 많은 것들이 있다…

그런데 우리가 middle ware를 사용할 때 연결을 끊을 수 있다.

cookie parser- cookie에 유저 정보를 저장. cookie를 전달 받아서 사용할 수 있도록 만들어주는 미들웨어

body parser- 누가 form을 끝까지 다 채우면 form은 서버에 의해서 받아져야만 함. ex) 사람의 비디오를 업로드할 때 제목이나 댓글 정보를 전달할 때 업로드

이름과 비밀번호를 작성하면 받아야한다. 우리는 req ob에 접근할 수 있기를 원한다. 그러기 위해서 body parser을 이용.

const handleHome=(req,res)=>res.send("hello baby");

const handleProfile =(req,res)=> res.send("You are on my profile");

app.use(cookieparser());

app.use(bodyparser.json());

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

app.use(morgan("div"));

app.use(helmet());

app.get("/",handleHome); *//누군가가 app에 접속하게 되면 handlehome을 부름*

app.get("/profile",handleProfile);

app.listen(PORT,handleListening);

*// 4000개의 포트를 들으라!! app아 !!*

2.8

*export* *default* app;

다른 파일에서도 사용 할 수 있도록 함.

router은 route들의 복잡함을 쪼개주는데 사용할 수 있다.

userRouter.get("/",(req,res) => res.send('user index'))

userRouter.get("/edit",(req,res) => res.send('user edit'))

userRouter.get("/password",(req,res) => res.send('user password'))

이렇게 쓰기 가능

/user/edit/. …..

이처럼 router는 작은 파일로 나눠서 사용 할 수있다.

app.js

*import* { userRouter } *from* "./router";

app.use("/user",userRouter); *//누가 user경로로 접속하면 userRouter을 전체 다 사용하겠다는 의미이다.*

*export* *default* app;

router.js

*import* express *from* "express";

*export* const userRouter = express.Router(); *//이 녀석을 내보낼 수 있다.ㄴ*

userRouter.get("/",(req,res) => res.send('user index'));

userRouter.get("/edit",(req,res) => res.send('user edit'));

userRouter.get("/password",(req,res) => res.send('user password'));

2.9

mvc는 model :데이터 view: 데이터가 어떻게 생겼는지 control: 데이터를 찾는 함수 을 의미

각각 router의 의미가 담기게 분리하였다. /join 을 하려면 userRouter.js에서 만들어야한다. 카테고리칼 작업을 하는 것.

*import* express *from* "express"; *// const express = require('express');를 최신버전으로 바꿀 수 있다.*

*import* "core-js";

*import* morgan *from* "morgan";

*import* helmet *from* "helmet";

*import* cookieparser *from* "cookie-parser";

*import* bodyparser *from* "body-parser";

*import* userRouter *from* "./routers/userRouter";

*import* videoRouter *from* "./routers/videoRouter";

*import* globalRouter *from* "./routers/globalRouter";

*// node module을 어딘가에 가지고 옴  express를 내 파일들 속에서 찾으려고 함.  아님 없으면 node-modul에서 찾으려고 한다.*

const app = express();

const PORT=4000;

app.use(cookieparser());

app.use(bodyparser.json());

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

app.use(morgan("div"));

app.use(helmet());

app.use("/",globalRouter); *// /join , /login /serch*

app.use("/user",userRouter); *//누가 user경로로 접속하면 userRouter을 전체 다 사용하겠다는 의미이다.*

app.use("/video",videoRouter);

*export* *default* app;

*import* express *from* "express";

 const globalRouter = express.Router(); *//이 녀석을 내보낼 수 있다.ㄴ*

*export* *default* globalRouter; *// what export ???  파일로 export한다는 뜻이다.*

2.10

/user/edit/password

이런식으로 url의 구조를 다 외우게 하지 못하도록 작업을 할 것이다.

그래서 따로 분리해 놓게 코드를 작성했따.

*//전역적으로 선언*

const Home = "/";

const JOIN="/join";

const LOGIN="/login";

const LOGOUT="/logout";

const SEARCH="/search";

*//users*

const USERS="/users";

const USERS\_DETAIL="/:id"; *// 만약에 유저라면 아이디를 부여한다. : <- 이 부호는 값이 변한다는 것을 뜻 한다.*

*//users/1*

const EDIT\_PROFILE="/edit-profile";

const CHANGE\_PASSWORD="/change-password";

*//videos*

const VIDEOS="/videos";

const UPLOAD="/upload";

const VIDEO\_DETAIL="/:id";

const EDIT\_VIDEO="/:id/edit";

const DELETE\_VIDEO="/:id/delete";

*//videos/1/edit*

const routes={

    home:Home,

    join:JOIN,

    login:LOGIN,

    logout:LOGOUT,

    search:SEARCH,

    users:USERS,

    usersdetail:USERS\_DETAIL,

    editprofile:EDIT\_PROFILE,

    changepassword:CHANGE\_PASSWORD,

    videos:VIDEOS,

    upload:UPLOAD,

    videosdetail:VIDEO\_DETAIL,

    editvideo:EDIT\_VIDEO,

    deletevideo:DELETE\_VIDEO

};

*export* *default* routes; *// what export ???  파일로 export한다는 뜻이다.*

2.11

routes를 컨트롤 할 수 있는 것이 필요하다.

wetube에서 video, 비디오를 업로드할 users 그리고 routes를 한 번 보자.

컨트롤러는 어떤 일이 어떻게 발생하는지에 관한 로직이다.

*import* express *from* "express";

*import* {home,search} *from* "../controllers/videoController"

*import* {login,logout,Join} *from* "../controllers/userController"

*import* routes *from* '../routes';

*import* routers *from* "../routes";

const globalRouter = express.Router(); *//이 녀석을 내보낼 수 있다.ㄴ*

globalRouter.get(routes.home,home); *//export해서 이렇게 다른 파일 변수를 사용할 수 있따.*

globalRouter.get(routes.join,Join);

globalRouter.get(routes.login,login);

globalRouter.get(routes.logout,logout);

globalRouter.get(routes.search,search);

*export* *default* globalRouter; *// what export ???  파일로 export한다는 뜻이다.*

*export* const home = (req,res) => res.send("home");

*export* const search=(req,res)=>res.send("search");

=> 이 화살표의 의미는 무엇인가?

암시적 리턴이라는 것이 있는데

function alalalal(){return true}

lalalala = () => true 이를 암시적 리턴이라고 함.

여기서 화살표를 적어주면 암시적 성격을 잃게되어서

lalalala = () => {return true } 를 적어줘야 한다.

2.13

V(데이터가 어떻게 생겼는지)에 해당하는 것을 보기위해 pug를 다운

html을 더 이쁘게 만들어 준다.

pug는 템플릿 언어이다. express의 view engine이다.

*export* const home = (req,res) => res.send("home");

send를 render로 바꿔준다.

2.14

pug에 extension을 이용해서 다른 파일 코드를 사용할 수 있도록 확장해준다.

extends layouts/main.pug

덮어쓰기하는 작업

2.15

/// 폰트 어썸 , font awesome

link(rel="stylesheet", href="https://use.fontawesome.com/releases/v5.5.0/css/all.css" integrity="sha384-B4dIYHKNBt8Bc12p+WXckhzcICo0wtJAoU8YZTY5qE0Id1GSseTk6S+L3BlXeVIU", crossorigin="anonymous")

footer.footer

    .footer\_\_icons

        i.fab.fa-youtube

    span.footer\_\_text &copy; #{new Date().getFullYear()} Wetube

pug에 js를 넣을 수 있음.

텍스트 사이에 자바스크립트를 추가하고 싶으면 #{} 이렇게 추가하면 된다.

doctype html

html

    head

        link(rel="stylesheet", href="https://use.fontawesome.com/releases/v5.5.0/css/all.css", integrity="sha384-B4dIYHKNBt8Bc12p+WXckhzcICo0wtJAoU8YZTY5qE0Id1GSseTk6S+L3BlXeVIU", crossorigin="anonymous")

        title Wetube

    body

        include ../partials/header

        main

            block content

        include ../partials/footer

header.pug 파일과 footer.pug 파일을 추가해주는 작업을 한다.

2.16

헤더.pug가 라우트 객체에 접근하도록 하고 싶을 때 🡺미들웨어 사용

//복습

app.use((req,res,next) =>{

})

app.use(function(req,res,next){

})

이렇게 함수를 만들 수 있다.

*import* routes *from* "./routes";

*export* const localsMiddleware = (req,res,next)=>{

    res.locals.siteName = "wetube";

    res.locals.routes=routes;

    next(); *//미들웨어가 next에 req를 전달해야 함. 미들웨어가 커넥션과 라우트들 사이에 있으니까 여기서 next를 넣음.*

};

app.use(helmet());

app.set('view engine', "pug");

app.use(cookieparser());

app.use(bodyparser.json());

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

app.use(morgan("div"));

*//local 기능을 사용하면 변수에 접근할 수 있다.*

app.use(localsMiddleware);

app.use(routes.home,globalRouter); *// /join , /login /serch*

app.use(routes.users,userRouter); *//누가 user경로로 접속하면 userRouter을 전체 다 사용하겠다는 의미이다.*

app.use(routes.videos,videoRouter);

next()를 부른 이유는

app.use(routes.home,globalRouter); *// /join , /login /serch*

app.use(routes.users,userRouter); *//누가 user경로로 접속하면 userRouter을 전체 다 사용하겠다는 의미이다.*

app.use(routes.videos,videoRouter);

얘들을 사용하기 위해서다.

2.17

*export* const home = (req,res) => res.render("home",{pageTitle: "Home"});

두번째 인자에 변수를 선언해서

   title #{pageTitle}#{siteName}

main.pug에 변수를 이용해서 추가할 수 있따.

2.18

        form(action="get")

pug

url에 있는 값을 보기위해 get을 이용함,.

    .header\_\_column

        form(action="get")

            input(type="text", placeholder="Search by term.....")

빈칸을 만들어주는 작업을 한다.

*export* const search=(req,res)=>{

    console.log(req.query);

res.render("search",{pageTitle: "search"});};

req.query를 이용하여 빈칸에 사용자가 적은 문장을 리턴해준다.

*export* const search=(req,res)=>{

    const {

        query: {term: searchingBy}}=req; *//req.query.term term=req.query.term 이라고 하는 것보다 더 좋은 방식이다.*

*//console.log(term);*

res.render("search",{pageTitle: "search", searchingBy});

};

------------------복습 1

처음 node.js를 시작할 때 npm init으로 package.json 파일을 만들어준다.

npm install express로 express를 설치해준다

express?? : JavaScript 생태계에서, 인기있는 프레임워크의 앞글자를 따서 MERN stack으로 흔히 부르곤 합니다. (MongoDB, Express, React, Node)  
이들 중 Express.js는 Node.js 환경에서 웹 어플리케이션 혹은 API를 제작하기 위해 사용되는 인기있는 프레임워크입니다.

node.js를 위한 빠르고 개방적인 간결한 웹 프레임 워크

(그런데 항상 다른 프로젝트를 만들 때 계속 install 해주어야하나?)

const express=require(‘express’) 🡺 express를 node\_modules에 가지고오는 것.

export defult 를 사용하고

import를 사용하려면 babelrc를 먼저 적용해야한다.

그런데 express 옵션 중 use 와 get의 차이점은 무엇인가?

res.send

다음의 라우트 경로는 요청을 /random.text에 일치시킵니다.

app.get('/random.text', function (req, res) {

res.send('random.text');

});

문자열 패턴을 기반으로 하는 라우트 경로의 몇 가지 예는 다음과 같습니다.

다음의 라우트 경로는 acd 및 abcd와 일치합니다.

app.get('/ab?cd', function(req, res) {

res.send('ab?cd');

});

res.render

res.send : 다양한 유형의 응답을 전송합니다.

res.redirect : 브라우저를 리다이렉트 합니다.

res.render : 설정된 템플릿 엔진을 사용해서 views를 렌더링합니다.

res.json : JSON 응답을 전송합니다.

res.end : 응답 프로세스를 종료합니다.

main.pug

doctype html

html

    head

        link(rel="stylesheet", href="https://use.fontawesome.com/releases/v5.5.0/css/all.css", integrity="sha384-B4dIYHKNBt8Bc12p+WXckhzcICo0wtJAoU8YZTY5qE0Id1GSseTk6S+L3BlXeVIU", crossorigin="anonymous")

        title #{pageTitle} | #{siteName}

    body

        include ../partials/header

        main

            block content

        include ../partials/footer

head부분에는 일단 font awesome에서 사용 할 link를 걸어두고, 사이트 이름을 바꿔줄 수 있다.

body부분에는 ../partials/header 파일을 연결하고

main 부분에는 block과 ../partials/footer 파일을 연결했다.

header.pug

header.header

    .header\_\_column

        a(href=routes.home)

            i.fab.fa-youtube

    .header\_\_column

        form(action=routes.search,method="get")

            input(type="text", placeholder="Search by term.....",name="term")

    .header\_\_column

        ul

            li

                a(href=routes.join) Join

            li

                a(href=routes.login) Log In

유튜브 아이콘에 하이퍼링크를 걸어두고

그 다음 줄에 서치하는 칸을 만들어서 그 칸안에 Search by term을 써주었다.

그 다음 줄에는 join log in을 하이퍼 링크로 걸어주고 list를 만들어 주었다.

2.19

*<!-----join화면과 login화면 둘다 로그인 기능을 넣기위해서 만듬-->*

.social-login

    button

        span

            i.fab.fa-github

        |Continue with Github

github, facebook등의 아이콘을 join과 login 화면에 나오도록 하기위해 sociallogin.pug를 따로 만들어 주어서

login.pug과 join.pug에 include 해주었다.

            include partials/socialLogin

로그인 , join 등의 화면에 나오게 pug를 작업해줌.

socialLogin.pug

2.20

/user/edit-profile에 접속해야하는데 /users/:id로 접속한걸로 오해할 수 있음.

그래서

userRouter.get(routes.usersdetail,usersdetail);

userRouter.get(routes.editprofile,editprofile);

userRouter.get(routes.changepassword,changepassword);

이랬던 위치를

userRouter.get(routes.editprofile,editprofile);

userRouter.get(routes.usersdetail,usersdetail);

userRouter.get(routes.changepassword,changepassword);

이렇게 바꿔줌

2.21

/videos/1/edit

하이퍼링크가 걸린 delete video를 자세히 살펴보면 /videos/:id/edit 이라고 되어있다.

:id 를 1로 바꿔야한다. 🡺(이렇게 나오는 이유는 .routes.js에 있는 라우트들은 express와 잘 맞는다. 그래서 이렇게 나옴)

어느 하나는 express에 값을 전달하고 있고, 어느 하나는 브라우저에 바로 값을 전달함.

가짜 데이터베이스를 만드는 이유 == 데이터베이스에서 무엇을 추가해야 할지 설정을 하기 전에 빠르게 수정하고 싶기 때문이다.

*export* const videos =[

    {

        id:3243,

        title: 'Video awesom',

        description: 'This is somthing I love',

        views:24,

        videoFile:"https://archive.org/details/BigBuckBunny\_124"

        creator:{

            id:123123,

            name:"길태연",

            email:"jookinho@naver.com",

        }

    }

]

*import* {videos} *from* "../dp"

*export* const home = (req,res) => {

    res.render("home",{pageTitle: "Home,videos}

});

홈화면에 가면 비디오의 목록을 받아올 것이다.

home.pug

extends layouts/main.pug

block content

    .videos

        each video in videos1

            h1= video.title

그래서 each video in videos1라는 함수를 사용해서

타이틀들을 받아온다.

2.22

html 재활용 하기위해서 바로 html 코드를 작성하는 것이아닌 mixin 함수를 이용.

mixin videoblock()

다른 곳에 사용하고 싶으면

include mixin/videoblock

비디오를 이용하고 싶을 때 main.pug에서 추가

meta(http-equiv="Content-Security-Policy",content="defalut-src \*")

아니면

app.js에서

app.use(helmet({contentSecurityPolicy:false}));

사용

 video.videoblock\_\_thumbnail(src=video.videoFile,controls=true)

이렇게 controls를 true로 하면 비디오를 실제로 볼 수 있다.

2.23

회원가입하고 누를 때 정보 얻기

globalRouter.get(routes.join,getJoin);

globalRouter.post(routes.join,postJoin);

get한 것을 post 할 수 있음.

bodypaser의 역할

console.log(req.body);

console.log 창에서 회원가입의 정보를 얻을 수 있다.

만약에 비밀번호와 verify비밀번호가 다르면 어떤 액션을 취할 것

그러기위해 상태 코드(status code)가 필요한데 인터넷이 서로 어떻게 상호작용하는지 표시하는 것이다.

200이면 ok

404 not found

*export* const postJoin=(req,res)=>{

    const {

        body: {name, email, password, password2}

    }=req;

*if*(password!==password2){

        res.status(400);

        res.render("Join",{pageTitle: "Join"});

    } *else*{

        res.redirect(routes.home);

    }

만약에 패스워드 같으면

다시 홈페이지를 부른다.

2.24

   if user.isAuthenticated //아직 존재하지않는 변수

            li

                a(href=routes.join) Join

            li

                a(href=routes.login) Log In

        else

            li

                a(href=routes.upload) Upload

            li

                a(href=routes.usersdetail) Profile

            li

                a(href=routes.logout) Log out

미들웨어에 정보를 추가해줬다

    res.locals.user={

        isAuthenticated:true,

        id:1

    };

그런데 html은 /:id routes를 이해하지 못해서 id를 반환하지 못한다.

그래서 routes에서 함수를 적용해 주었다.

  usersdetail:(id) =>{

*if*(id){

*return* `/users/#{id}`;

        }*else* {

*return* USERS\_DETAIL;

        }

함수 적용 후 함수를 불러야 하므로 ()를 추가했다.

userRouter.get(routes.usersdetail(),usersdetail);

4.1, 4.2

몽고 db –

database는 sql과 nosql로 나뉜다.

dotenv를 설치한 이유는 내가 어떤 부분을 숨겨놓고 싶을 수 있기 떄문이다.

4.3

dotenv

.env 파일을 만든다.

MONGO\_URL="mongodb://localhost:27017/wetube"

PORT=4000

//gitignore에 .env파일이 ignore되고 있다는 것을 알 수 있다.

init에도 똑같이 해본다.

*mport* dotenv *from* "dotenv";

dotenv.config();

const PORT = process.env.PORT || 4000; *// 대상을 못 찾으면 4000으로*

이렇게 키를 숨길 수 있다.

4.4 mongodb의 장점: 다큐멘트를 줄일 수 있다는 것.

schema로 무엇을 하는지 알고싶으면 schema 사이트에가서 찾을 수 있다.

 mongoose.Schema

*import* mongoose *from* "mongoose";

*// model => document name / schema*

const VideoSchema= new mongoose.Schema({

    fileUrl: { // 비디오는 데이터가 커서 주소로 받아온다.

        type: String,

        required:"File URL is requird"

    },

    title:{

        type:String,

        required:"Title is required"

    },

    description: String,

    views:{ //0 뷰부터시작

        type:Number,

        defalut:0

    },

    createdAt:{ //언제 만들어졌는지 현재 날짜를 받아온다.

        type:Date,

        defalut:Date.now

    }

*//video 링크를 집어넣음 저장하지 않는다. 무거워지니까*

});

4.5

async 나를 기다려주는 무언가.

javascript는 나를 기다리지 않는다. 일단 video를 살펴보고 있으니까 다음 render작업도 같이 할게!!

우리는 js가 ㅜ릴 기다리게 해야한다.

async: js야 이 function의 어떤 부분은 꼭 기다려야해!!

*export* const home = async(req,res) => {

    const videos = *await* Video.find({}); *//await는 다음 과정이 끝날 때까지 잠시 기다려 달라는 의미이다.*

*//이렇게하면 datebase에 있는 모든 video를 가지고 올 수 있다.*

    res.render("home",{pageTitle: "Home",videos1}

*export* const home = async(req,res) => {

*try*{

    const videos = *await* Video.find({}); *//await는 다음 과정이 끝날 때까지 잠시 기다려 달라는 의미이다.*

*//이렇게하면 datebase에 있는 모든 video를 가지고 올 수 있다.*

    res.render("home",{pageTitle: "Home",videos});

    }*catch*(error){

        console.log(error);

        res.render("home",{pageTitle: "Home",videos:[]});

    }

};

에러를 잡기위해 try catch문을 사용한다.

*throw* Error("lalla");

를 이용해서 에러가 뜨면. 콘솔에 뜰 수 있게 할 수 있다.

4.6

2:44 초 3.25와 연결해서하기

middleware multer: function을 집어넣으면 middleware가 file의 url을 반환할 것이다.