

db.js

const mongoose = require("mongoose");

const asynchandler = require("express-async-handler");

require("dotenv").config(); // .env 파일 사용하기 위해

const connectDb = asynchandler(async () => {

const connect = await mongoose.connect(process.env.DB\_CONNECT);

console.log(`DB connected: ${connect.connection.host}`);

});

module.exports = connectDb;

Article.js

const mongoose = require('mongoose');

// 스키마 정의

const ArticleSchema = new mongoose.Schema({

// 게시물 제목

title: {

type: String,

required: true,

},

// 게시물 내용

body: {

type: String,

required: true,

},

filename: {

type: String,

required: true,

},

path: {

type: String,

require: true,

},

// 게시물 작성일

createdAt: {

type: Date,

default: Date.now(),

},

});

const Article = mongoose.model('Article', ArticleSchema);

module.exports = Article;

News.js

const mongoose = require("mongoose");

// 스키마 정의

const commentSchema = new mongoose.Schema({

// 게시물 제목

nickname: {

type: String,

required: true,

},

// 게시물 내용

body: {

type: String,

required: true,

},

// 게시물 작성일

createdAt: {

type: Date,

default: Date.now,

},

});

// 모델 생성

const Comment = mongoose.model('Comment', commentSchema);

module.exports = Comment;

Order.js

const mongoose = require('mongoose');

const orderdataSchema = new mongoose.Schema(

{

devicename: {

type: String,

},

casewhat: {

type: String,

},

sangtae: {

type: String,

},

name: {

type: String,

},

email: {

type: String,

},

phone: {

type: String,

},

},

{

timestamps: true,

}

);

const Contact = mongoose.model('Orderdata', orderdataSchema);

module.exports = Contact;

User.js

const mongoose = require("mongoose");

// Mongo 스키마 가져오기

const Schema = mongoose.Schema;

// 사용자 스키마 생성

const UserSchema = new Schema({

username: {

type: String,

required: true,

unique: true, // 중복X

},

password: {

type: String,

required: true,

},

});

// Users 모델을 생성하고 내보내기

module.exports = mongoose.model("Userdata", UserSchema);

admin.js

const express = require("express");

const router = express.Router();

const adminLayout = "layouts/admin";

const adminLayout2 = "layouts/admin-nologout";

const asyncHandler = require("express-async-handler");

const bcrypt = require("bcrypt");

const User = require("../models/User");

const Comment = require("../models/News");

const jwt = require("jsonwebtoken");

const JWT\_SECRET = process.env.JWT\_SECRET;

// 이미지 업로드 관련

// const multer = require('multer');

// const Image = require('../models/Article');

// const path = require('path');

// const fs = require('fs');

// Check Login

const checkLogin = async (req, res, next) => {

const token = req.cookies.token; // 쿠키 정보 가져오기

// 토큰이 없다면

if (!token) {

return res.redirect("/admin");

}

// 토큰이 있다면 토큰을 확인하고 사용자 정보를 요청에 추가

try {

const decoded = jwt.verify(token, JWT\_SECRET);

req.userId = decoded.userId; // res.userId가 아닌 req.userId에 저장

next();

} catch (error) {

return res.redirect("/admin");

}

};

// GET /logout

// Admin Logout

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

res.clearCookie("token");

res.redirect("/"); // 루트 페이지로 이동

});

// GET /admin

// 관리자 페이지

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

const locals = {

title: "관리자 페이지",

};

res.render("admin/index", { locals, layout: adminLayout2 });

});

// GET /register

// 회원가입 페이지

router.get(

"/register",

asyncHandler(async (req, res) => {

res.render("admin/register", { layout: adminLayout2 });

})

);

// POST /admin

// 관리자 페이지 로그인

router.post(

"/admin",

asyncHandler(async (req, res) => {

const { username, password } = req.body; // req.body에서 username과 password를 가져옴

// 사용자 이름으로 사용자 찾기

const user = await User.findOne({ username });

// 일치하는 사용자가 없으면 에러 메시지 출력

if (!user) {

return res.status(401).json({ message: "일치하는 사용자가 없습니다." });

}

const isValidPassword = await bcrypt.compare(password, user.password);

// 비밀번호가 일치하지 않으면 에러 메시지 출력

if (!isValidPassword) {

return res.status(401).json({ message: "비밀번호가 일치하지 않습니다." });

}

// JWT 토큰 생성

const token = jwt.sign({ id: user.\_id }, JWT\_SECRET);

// JWT 토큰을 쿠키에 저장

res.cookie("token", token, {

httpOnly: true,

});

// 로그인에 성공하면 전체 게시물 목록 페이지로 이동

res.redirect("/allPosts");

})

);

// GET /allPosts

// 전체 게시물 목록 페이지

// router.get(

// "/allPosts",

// checkLogin,

// asyncHandler(async (req, res) => {

// const locals = {

// title: "전체 게시물",

// };

// const data = await Comment.find().sort({ updatedAt: "desc", createdAt: "desc" });

// res.render("admin/allPosts", {

// locals,

// data,

// layout: adminLayout,

// });

// })

// );

router.get(

"/allPosts",

checkLogin,

asyncHandler(async (req, res) => {

const locals = {

title: "최신 테크 뉴스",

};

const comments = await Comment.find();

res.render("admin/allPosts", { comments: comments, layout: adminLayout });

})

);

router.delete(

"/:id",

checkLogin,

asyncHandler(async (req, res) => {

await Comment.deleteOne({\_id: req.params.id});

res.redirect('allPosts');

})

);

// 업로드 폴더 확인 후 생성

// const uploadDir = path.join(\_\_dirname, 'uploads');

// if (!fs.existsSync(uploadDir)) {

// fs.mkdirSync(uploadDir);

// }

// // Multer 저장 설정 (디스크 저장 방식)

// const storage = multer.diskStorage({

// destination: function (req, file, cb) { // callback

// cb(null, uploadDir);

// },

// filename: function(req, file, cb) {

// cb(null, Date.now() + path.extname(file.originalname));

// // 파일 이름을 현재 시간과 확장사로 지정

// }

// });

// // Multer 미들웨어 설정

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

// // 업로드된 파일을 정적 파일로 제공

// router.use('/allPosts', express.static(uploadDir));

// // 파일 업로드 폼을 EJS 템플릿으로 렌더링

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

// // views/index.ejs 파일을 렌더링

// res.render('addImage');

// });

// // 파일 업로드를 처리하는 라우터

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

// if (!req.file) {

// return res.status(400).send('파일이 업로드 되지 않았습니다.');

// }

// const { title, body } = req.body;

// console.log('파일 정보:', req.file);

// console.log('제목:', title);

// console.log('내용:', body);

// // MongoDB

// const newImage = new Image({

// title: title,

// body: body,

// filename: req.file.filename,

// path: path.join('uploads', req.file.filename),

// });

// newImage.save()

// .then(() => {

// console.log('파일 업로드 완료');

// // EJS에 파일과 제목, 내용을 모두 전달

// res.render('admin/allPosts', {

// uploadedFile: {

// title,

// body,

// filename: req.file.filename,

// path: req.file.path

// }

// });

// })

// .catch((err) => {

// console.error('오류:', err);

// res.status(500).send('파일 업로드 오류');

// });

// });

/\*\*

\* DELETE /delete/:id

\* Admin - Delete Post

\*/

router.delete(

"/delete/:id",

checkLogin,

asyncHandler(async (req, res) => {

await Post.deleteOne({\_id: req.params.id});

res.redirect('/allPosts');

})

);

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

res.render('admin/register', {layout: adminLayout2});

})

router.post(

'/register',

asyncHandler(async (req, res) => {

const {username, password, password2} = req.body;

if(password === password2) {

// bcrypt를 사용해 비밀번호를 암호화

const hashedPassword = await bcrypt.hash(password, 10);

console.log(hashedPassword);

// 사용자 이름과 암호화된 비밀번호를 사용해서 새 사용자를 생성

const user = await User.create({ username, password: hashedPassword});

// 성공메시지 출력

res.status(201).json({message: "Register successful", user})

}

else {

res.send("Register Failed")

}

})

)

module.exports = router;

main.js

const express = require("express");

const router = express.Router();

const mainLayout = "layouts/main";

const Comment = require("../models/News");

const Contact = require('../models/Order');

const Article = require('../models/Article');

// const Image = require("../models/Images");

const asynchandler = require("express-async-handler");

// const path = require('path');

// const fs = require('fs');

// const multer = require('multer');

// "/home" 라우트

router.get(

["/", "/order"],

asynchandler(async (req, res) => {

const locals = {

title: "Order",

};

const contacts = await Contact.find();

res.render("order", { contacts: contacts, layout: mainLayout });

})

);

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

res.render('newOrder', {layout: mainLayout});

})

router.post(

"/newOrder",

asynchandler(async (req, res) => {

const { devicename, casewhat, sangtae, name, phone, email } = req.body;

if(!devicename || !casewhat || !sangtae || !name || !phone || !email) {

return res.status(400).send('필수값이 입력되지 않았습니다.');

}

const contact = await Contact.create({

devicename,

casewhat,

sangtae,

name,

phone,

email,

});

// res.status(201).send("사용자 데이터 생성됨");

res.redirect("/order"); //mid add

})

)

router.get(

"/order/:id",

asynchandler(async (req, res) => {

const locals = {

title: "주문수정"

};

const contact = await Contact.findById(req.params.id);

//mid add

// res.status(200).send(contact);

res.render('update', { contact: contact, layout: mainLayout });

})

)

router.put(

"/order/:id",

asynchandler(async (req, res) => {

const locals = {

title: "주문수정"

};

const id = req.params.id;

const { devicename, casewhat, sangtae, name, phone, email } = req.body;

const contact = await Contact.findById(id);

if(!contact) {

res.status(404);

throw new Error('사용자 데이터 찾을 수 없음');

}

// 수정

contact.devicename = devicename;

contact.casewhat = casewhat;

contact.sangtae = sangtae;

contact.name = name;

contact.email = email;

contact.phone = phone;

// 저장

contact.save();

//mid add

// res.status(200).json(contact);

res.redirect("/order");

})

)

router.delete(

"/delete/:id",

asynchandler(async (req, res) => {

await Contact.deleteOne({\_id: req.params.id});

res.redirect('/order');

})

);

router.get(

"/news",

asynchandler(async (req, res) => {

const locals = {

title: "최신 테크 뉴스",

};

const comments = await Comment.find().sort({ updatedAt: "desc", createdAt: "desc" });

const articles = await Article.find().sort({ updatedAt: "desc", createdAt: "desc" });

res.render("news", { comments: comments, articles: articles, layout: mainLayout });

})

);

router.post(

"/news",

asynchandler(async (req, res) => {

const { nickname, body, createdAt } = req.body;

if (!nickname || !body) {

return res.status(400).send('필수값이 입력되지 않았습니다.');

}

const comment = await Comment.create({

nickname,

body,

createdAt,

});

res.redirect("/news");

})

);

// router.get(

// "/news",

// asynchandler(async (req, res) => {

// const locals = {

// title: "최신 테크 뉴스",

// };

// const articles = await Article.find();

// res.render("news", { article: article, layout: mainLayout });

// })

// );

module.exports = router;

allPosts.ejs

<div class="clear"></div>

<main>

<div id="sec1\_bg">

<div class="admin-layout">

<!-- 뉴스 업데이트 섹션 -->

<section class="news-update">

<h2>뉴스 업데이트</h2>

<!-- 업로드 폼 -->

<form action="/allPosts" method="post" enctype="multipart/form-data">

<label for="file">파일선택:</label>

<input type="file" id="file" name="file" required />

<label for="title">제목:</label>

<input type="text" id="title" name="title" placeholder="제목">

<label for="body">내용:</label>

<textarea name="body" id="body" placeholder="내용" rows="10"></textarea>

<button type="submit">업로드</button>

</form>

<br>

<!-- 업로드된 파일 정보가 있을 경우 표시 -->

<% if (typeof uploadedFile !== 'undefined') { %>

<h2>업로드된 파일:</h2>

<p>제목: <%= uploadedFile.title %></p>

<p>내용: <%= uploadedFile.body %></p>

<p>파일명: <%= uploadedFile.filename %></p>

<p>파일 경로: <%= uploadedFile.path %></p>

<!-- 이미지 미리보기 -->

<h3>업로드된 이미지:</h3>

<img src="/uploads/<%= uploadedFile.filename %>" alt="Uploaded Image" style="max-width: 500px;">

<% } %>

</section>

<!-- 댓글 관리 섹션 -->

<section class="comment-management">

<h2>댓글관리</h2>

<% comments.forEach(comment => { %>

<div class="comment">

<div class="comment-content">

<p><strong><%= comment.nickname %></strong></p>

<p><%= comment.body %></p>

<form action="/<%= comment.\_id %>?\_method=DELETE" method="POST" style="display: inline-block; width: 100px;">

<input type="submit" value="삭제" class="btn-delree btn">

</form>

</div>

</div>

<% }); %>

</section>

</div>

</div>

</main>

admin > index.ejs

<div class="clear"></div>

<!-- MAIN -->

<main>

<div class="home-container">

<h1>어드민 로그인</h1>

<form action="/admin" method="POST" class="login">

<label for="username"><b>Username</b></label>

<input type="text" placeholder="어드민 아이디" name="username" id="username"><br>

<label for="password"><b>Password</b></label>

<input type="password" placeholder="비밀번호" name="password" id="password">

<br>

<button type="submit">로그인</button>

</form>

<p>ADMIN ONLY ~UNIFIX</p>

</div>

</main>

<!-- /MAIN -->

admin > register.ejs

<div class="clear"></div>

<!-- MAIN -->

<main>

<div class="home-container">

<h1>Register</h1>

<form action="/register" method="POST" class="register">

<label for="username"><b>아이디</b></label>

<input type="text" placeholder="아이디" name="username" id="username">

<label for="password"><b>비밀번호</b></label>

<input type="password" placeholder="비밀번호" name="password" id="password">

<label for="password2"><b>비밀번호 확인</b></label>

<input type="password" placeholder="비밀번호 확인" name="password2" id="password2">

<input type="submit" value="등록" class="register-btn">

</form>

<p>!!!ADMIN ONLY</p>

</div>

</main>

<!-- /MAIN -->

layouts > admin-nologout.ejs

<!DOCTYPE html>

<html lang="ko">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>주문조회</title>

<link rel="stylesheet" href="/css/style.css">

<link rel="stylesheet" as="style" crossorigin href="https://cdn.jsdelivr.net/gh/orioncactus/pretendard@v1.3.9/dist/web/static/pretendard.min.css" />

</head>

<body>

<!-- HEADER -->

<header>

<nav>

<div id="nav\_logo\_white">

<a href="/"><img src="/img/unifix\_logo.png" alt="nav\_logo" height="25"></a>

</div>

<ul class="nav\_menu">

<span><b>ADMIN ONLY!!!</b></span>

<!-- <li><a href="/order">최신 뉴스 수정</a></li> -->

<!-- <li><a href="/news">최신 테크 뉴스</a></li>

<li><a href="/mypage">내 정보</a></li> -->

</ul>

</nav>

</header>

<!-- /HEADER -->

<main>

<%- body %>

</main>

</body>

</html>

layouts > admin.ejs

<!DOCTYPE html>

<html lang="ko">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>주문조회</title>

<link rel="stylesheet" href="/css/style.css">

<link rel="stylesheet" as="style" crossorigin href="https://cdn.jsdelivr.net/gh/orioncactus/pretendard@v1.3.9/dist/web/static/pretendard.min.css" />

</head>

<body>

<!-- HEADER -->

<header>

<nav>

<div id="nav\_logo\_white">

<a href="/"><img src="/img/unifix\_logo.png" alt="nav\_logo" height="25"></a>

</div>

<ul class="nav\_menu">

<span><b>ADMIN ONLY!!!</b></span>

<li><a href="/order">최신 뉴스 수정</a></li>

<li><a href="/logout">어드민 로그아웃</a></li>

<!-- <li><a href="/news">최신 테크 뉴스</a></li>

<li><a href="/mypage">내 정보</a></li> -->

</ul>

</nav>

</header>

<!-- /HEADER -->

<main>

<%- body %>

</main>

</body>

</html>

layouts > main.ejs  
  
<!DOCTYPE html>

<html lang="ko">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>주문조회</title>

<link rel="stylesheet" href="/css/style.css">

<link rel="stylesheet" as="style" crossorigin href="https://cdn.jsdelivr.net/gh/orioncactus/pretendard@v1.3.9/dist/web/static/pretendard.min.css" />

</head>

<body>

<!-- HEADER -->

<header>

<nav>

<div id="nav\_logo\_white">

<a href="/"><img src="/img/unifix\_logo.png" alt="nav\_logo" height="25"></a>

</div>

<ul class="nav\_menu">

<li><a href="/order">주문(수리)</a></li>

<li><a href="/news">최신 테크 뉴스</a></li>

</ul>

</nav>

</header>

<!-- /HEADER -->

<main>

<%- body %>

</main>

</body>

</html>

newOrder.ejs

<div id="sec1\_bg">

<section id="sector\_neworder">

<h1 class="title">새 주문서 작성</h1>

<form action="" method="POST" id="addOrder">

<p class="subtitle">기기 정보</p>

<label for="deviceName">수리맡길 기기 이름을 적어주세요.</label><br>

<input type="text" class="orderInput" placeholder="ex. 삼성 노트북" name="devicename"><br><br>

<label for="">케이스를 선택해 주세요.</label><br>

<select name="casewhat" class="orderInput">

<option value="고장수리">고장수리</option>

<option value="업그레이드(추가)">업그레이드(추가)</option>

<option value="유지보수(청소)">유지보수(청소)</option>

</select> <br><br>

<label for="sangtae">현재 상태를 간략하게 적어주세요.</label><br>

<input type="text" class="orderInput" placeholder="ex. 전원이 안 켜져요" name="sangtae"><br><br>

<br><br>

<p class="subtitle">주문자 정보</p>

<label for="name">주문자명을 입력해 주세요.</label><br>

<input type="text" class="orderInput" placeholder="ex. 홍길동" name="name"><br><br>

<label for="name">이메일 주소를 입력해 주세요.</label><br>

<input type="email" class="orderInput" placeholder="example@aaa.bbb" name="email"><br><br>

<label for="phone">전화번호를 입력해 주세요.</label><br>

<input type="number" class="orderInput" placeholder="01000000000" name="phone"><br><br>

<input type="submit" value="제출" class="orderSubmit">

<a href="/contacts" class="btn back">주문 페이지로 돌아가기</a>

</form>

</section>

</div>

news.ejs

<div class="news-container">

<!-- 최신 뉴스 섹션 -->

<div class="news">

<div class="title">최신 테크 뉴스</div>

<% articles.forEach(item => { %>

<!-- 이미지 자리 -->

<img src="/uploads/<%= item.filename %>" alt="뉴스 이미지" class="news-image">

<h2><%= item.title %></h2>

<div>

<%= item.body %>

</div>

<% }); %>

</div>

<!-- 유저 댓글 섹션 -->

<div class="comments">

<div class="title">유저 댓글</div>

<form class="form-container" action="/news" method="POST">

<input type="text" name="nickname" placeholder="닉네임" required>

<textarea name="body" rows="3" placeholder="내용" required></textarea>

<button type="submit">등록</button>

</form>

<!-- 댓글 리스트 -->

<% comments.forEach(comment => { %>

<div class="comment">

<b><%= comment.nickname %></b> <br>

<big><%= comment.body %></big> <br><br>

<%= comment.createdAt %>

</div>

<% }); %>

<!-- <div class="comment">홍길동: 댓글 내용입니다.</div>

<div class="comment">홍길동: 댓글 내용입니다.</div>

<div class="comment">홍길동: 댓글 내용입니다.</div> -->

</div>

</div>

order.ejs

<div id="sec1\_bg">

<a href="/newOrder" class="btn new">새 주문 요청</a>

<% contacts.forEach(contact => { %>

<section id="sector1">

<div id="title\_box">

<h1><%= contact.name %>님,<br><%= contact.devicename %> 관련 주문 요청입니다.</h1>

</div>

<article id="status\_box">

<div id="flexbox">

<div class="status one">

<div id="flexbox2">

<div class="left\_status one">

<p class="current\_status">케이스: <%= contact.casewhat %><p>

<p class="main\_title">기기명</p>

<span class="subtitle"><%= contact.devicename %></span>

<p class="main\_title">상태</p>

<span class="subtitle"><%= contact.sangtae %></span>

<p class="main\_title">주문자 정보</p>

<span class="subtitle">

<%= contact.name %> <br>

<%= contact.phone %> <br>

<%= contact.email %>

</span>

</div>

</div>

</table>

</div>

<div class="status two">

<a href="/order/<%= contact.\_id %>" class="btn update">주문 수정</a>

<form action="/delete/<%= contact.\_id %>?\_method=DELETE" method="POST" style="display: inline-block;">

<input type="submit" value="삭제" class="btn-delree btn">

</form>

</div>

</div>

</article>

</section>

</div>

<% }); %>

update.ejs

<!-- MAIN -->

<main>

<div id="sec1\_bg">

<section id="sector\_neworder">

<h1 class="title">주문 수정</h1>

<form action="/order/<%= contact.\_id %>?\_method=PUT" method="POST" id="addOrder">

<p class="subtitle">기기 정보</p>

<label for="deviceName">수리맡길 기기 이름을 적어주세요.</label><br>

<input type="text" class="orderInput" name="devicename" value="<%= contact.devicename %>"><br><br>

<label for="">케이스를 선택해 주세요.</label><br>

<select name="casewhat" class="orderInput" value="<%= contact.casewhat %>">

<option value="고장수리">고장수리</option>

<option value="업그레이드(추가)">업그레이드(추가)</option>

<option value="유지보수(청소)">유지보수(청소)</option>

</select> <br><br>

<label for="sangtae">현재 상태를 간략하게 적어주세요.</label><br>

<input type="text" class="orderInput" name="sangtae" value="<%= contact.sangtae %>"><br><br>

<br><br>

<p class="subtitle">주문자 정보</p>

<label for="name">주문자명을 입력해 주세요.</label><br>

<input type="text" class="orderInput" name="name" value="<%= contact.name %>"><br><br>

<label for="name">이메일 주소를 입력해 주세요.</label><br>

<input type="email" class="orderInput" name="email" value="<%= contact.email %>"><br><br>

<label for="phone">전화번호를 입력해 주세요.</label><br>

<input type="number" class="orderInput" name="phone" value="<%= contact.phone %>"><br><br>

<input type="submit" value="제출" class="orderSubmit">

<a href="/order" class="btn back">주문 페이지로 돌아가기</a>

</form>

</section>

</div>

app.js

require('dotenv').config({ path: './path/to/.env' });

const express = require("express");

const expressEjsLayouts = require("express-ejs-layouts");

const connectDb = require("./config/db"); // DB 연결 함수 가져오기

const cookieParser = require("cookie-parser"); // 쿠키 파서 가져오기

const methodOverride = require("method-override")

// const mainLayout = "../views/layouts/main.ejs";

// 이미지 업로드 관련

const multer = require('multer');

const Image = require('./models/Article');

const path = require('path');

const fs = require('fs');

const app = express();

const port = process.env.PORT || 3001 // .env에 포트가 없으면 3001번 포트로 설정

connectDb(); // DB 연결 함수 실행

app.use(express.json()); // JSON 요청 본문 파싱

app.use(express.urlencoded({extended: true})); // URL-encoded 요청 본문 파싱

app.use(methodOverride("\_method"));

app.set('layout', 'layouts/admin');

app.use(expressEjsLayouts); // ejs 레이아웃 사용

app.use(cookieParser()); // 쿠키 파서 사용

app.set("view engine", "ejs"); // ejs 사용

app.set("views", path.join(\_\_dirname, "/views")); // views 폴더 설정

app.use(express.static(\_\_dirname + "/public")); // public 폴더 설정

app.use("/", require("./routes/main")); // 루트 경로로 들어오면 main.js로 이동

// app.use("/news", require("./routes/main"));

app.use("/", require("./routes/admin")); // /admin 경로로 들어오면 admin.js로 이동

// 업로드 폴더 확인 후 생성

const uploadDir = path.join(\_\_dirname, 'uploads');

if (!fs.existsSync(uploadDir)) {

fs.mkdirSync(uploadDir);

}

// Multer 저장 설정 (디스크 저장 방식)

const storage = multer.diskStorage({

destination: function (req, file, cb) { // callback

cb(null, uploadDir);

},

filename: function(req, file, cb) {

cb(null, Date.now() + path.extname(file.originalname));

// 파일 이름을 현재 시간과 확장사로 지정

}

});

// Multer 미들웨어 설정

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

// 업로드된 파일을 정적 파일로 제공

app.use('/uploads', express.static(uploadDir));

// 파일 업로드 폼을 EJS 템플릿으로 렌더링

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

// views/index.ejs 파일을 렌더링

res.render('admin/allPosts');

});

// 파일 업로드를 처리하는 라우터

app.post('/allPosts', upload.single('file'), (req, res) => {

if (!req.file) {

return res.status(400).send('파일이 업로드 되지 않았습니다.');

}

const { title, body } = req.body;

console.log('파일 정보:', req.file);

console.log('제목:', title);

console.log('내용:', body);

// MongoDB

const newImage = new Image({

title: title,

body: body,

filename: req.file.filename,

path: path.join('uploads', req.file.filename),

});

newImage.save()

.then(() => {

console.log('파일 업로드 완료');

const comments = []; // 실제로는 DB에서 댓글을 가져오는 코드가 필요합니다.

// EJS에 파일과 제목, 내용을 모두 전달

res.render('admin/allPosts', {

uploadedFile: {

title,

body,

filename: req.file.filename,

path: req.file.path

},

comments: comments // 댓글 데이터 전달

});

})

.catch((err) => {

console.error('오류:', err);

res.status(500).send('파일 업로드 오류');

});

});

app.listen(port, () => {

console.log(`Server is running on http://localhost:${port}`);

});