

2024.04.09

Express-generator 프로젝트 구조

- bin/www : 포트 번호 등과 같은 웹 서버를 구축하는데에 필요한 설정 데이터가 정의되어 있는 파일
⇒ .env 파일과 같이 설정 값을 가지고 에러 처리, 기타 추가 설정을 해주는 파일
- node_modules : Node.js , Express에 필요한 모듈들이 설치되는 폴더
- public : images, javascripts, stylesheets → 정적 파일(ex. 로고, 회사 소개 페이지 ...)파일

cf. 동적 : 사람마다 다른 데이터

- routes : 각 경로를 담당하는 모듈들이 들어있는 폴더
= 라우팅 로직을 구현하는 모듈들 : 클라이언트에서 어떤 요청으로 주냐에 따라 어떤 로직을 수행할 지 파일별로 분할 해서 관리하는 정도

(cf. 자바의 controller 역할)

- views : 클라이언트에게 html코드로 화면을 보내는 파일

app.js 서버의 시작점 ⇒ URL에 따라서 라우팅을 해줌.

- package.json : 이 프로젝트에 설치된 모듈 이름, 버전 등등 정보들이 작성되어 있는 파일

프로젝트 시작 과정

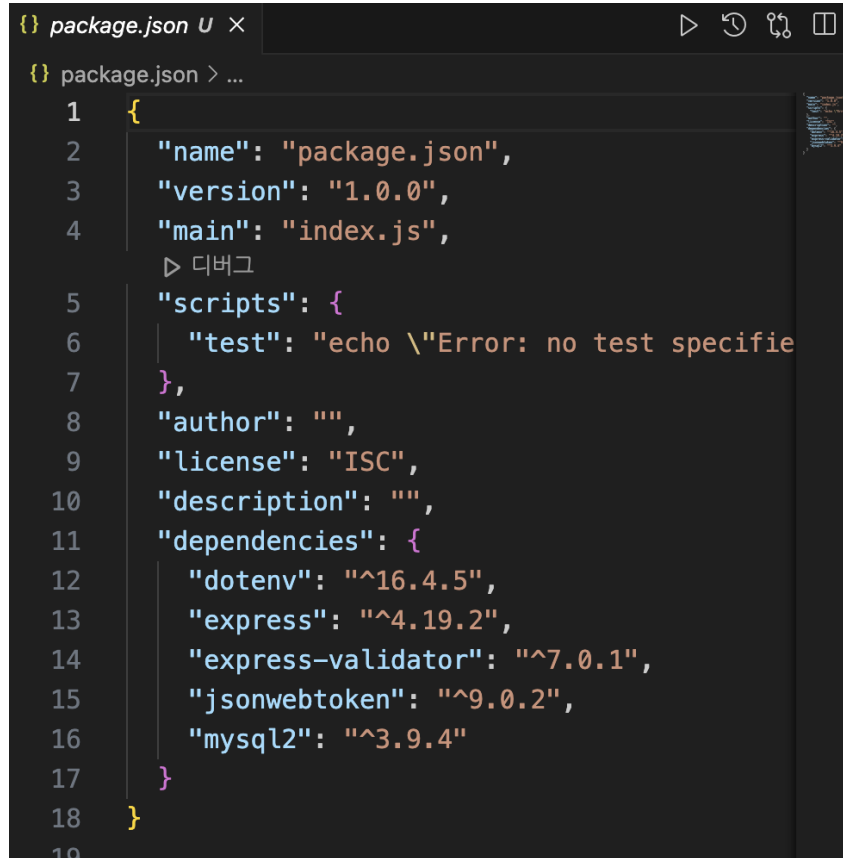
```
npm i express // package.json 파일 설치되지 않음
```

```
git init // 새로운 프로젝트 초기화 시키는 명령어
```

```
//package.json파일 생성 > 정보 입력
```

```
npm i express // package.json 파일에 express설치된 것을 확인할 수 있을
```

```
npm i dotenv express-validator jsonwebtoken mysql2
```

A screenshot of a code editor window showing the contents of a file named 'package.json'. The editor has a dark theme. The file content is a JSON object with the following properties: 'name' is 'package.json', 'version' is '1.0.0', 'main' is 'index.js', 'scripts' has a 'test' property with the value 'echo \"Error: no test specified\"', 'author' is an empty string, 'license' is 'ISC', 'description' is an empty string, and 'dependencies' is an object with 'dotenv' (^16.4.5), 'express' (^4.19.2), 'express-validator' (^7.0.1), 'jsonwebtoken' (^9.0.2), and 'mysql2' (^3.9.4). The editor shows line numbers from 1 to 19 on the left margin. There are some UI elements at the top like a search icon and a refresh icon.

```
{  
  "name": "package.json",  
  "version": "1.0.0",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \"Error: no test specified\"",  
  },  
  "author": "",  
  "license": "ISC",  
  "description": "",  
  "dependencies": {  
    "dotenv": "^16.4.5",  
    "express": "^4.19.2",  
    "express-validator": "^7.0.1",  
    "jsonwebtoken": "^9.0.2",  
    "mysql2": "^3.9.4"  
  }  
}
```

app.js 파일 생성 및 모듈 불러오기

app.js파일

```
const express = require('express');  
const app = express();
```

```
const dotenv = require('dotenv');
dotenv.config();

app.listen(process.env.PORT);
```

.env파일

```
PORT = 9999
```

프로젝트 구현 시작

1단계

app.js파일

```
const express = require('express');
const app = express();

const dotenv = require('dotenv');
dotenv.config();

app.listen(process.env.PORT);

const userRouter = require("./routes/users");
app.use("/", userRouter)
```

users 파일

```
const express = require("express");
const router = express.Router();

router.use(express.json()); //json 형식으로 들어올 것이다.
//회원가입
```

```

router.post('/join', (req, res) => {
  res.json('회원가입')
});
//로그인
router.post('/login' , (req, res) => {
  res.json('로그인')
});
//비밀번호 초기화 요청
router.post('/reset', (req, res) => {
  res.json('초기화 요청')
});
//비밀번호 초기화
router.put('/reset', (req, res) => {
  res.json('초기화')
});

module.exports = router;

```

2단계

app.js

```

const express = require('express');
const app = express();

const dotenv = require('dotenv');
dotenv.config();

app.listen(process.env.PORT);

const userRouter = require("./routes/users");
const bookRouter = require("./routes/books");
const likeRouter = require("./routes/likes");
const cartRouter = require("./routes/carts");

```

```
const orderRouter = require("./routes/orders");

app.use("/users", userRouter)
app.use("/books", bookRouter)
app.use("/likes", likeRouter)
app.use("/carts", cartRouter)
app.use("/orders", orderRouter)
```

users.js

```
const express = require("express");
const router = express.Router();

router.use(express.json()); //json 형식으로 들어올 것이다.
//회원가입
router.post('/join', (req, res) => {
    res.json('회원가입')
});
//로그인
router.post('/login', (req, res) => {
    res.json('로그인')
});
//비밀번호 초기화 요청
router.post('/reset', (req, res) => {
    res.json('초기화 요청')
});
//비밀번호 초기화
router.put('/reset', (req, res) => {
    res.json('초기화')
});

module.exports = router;
```

books.js

```

const express = require("express");
const router = express.Router();

router.use(express.json()); //json 형식으로 들어올 것이다.
//전체 도서 조회
router.get('/', (req, res) => {
    res.json('전체 도서 조회')
});
//개별 도서 조회
router.get('/:id' , (req, res) => {
    res.json('개별 도서 조회')
});
//카테고리별 도서목록 조회
router.get('/', (req, res) => {
    res.json('카테고리별 도서목록 조회')
});

module.exports = router;

```

likes.js

```

const express = require("express");
const router = express.Router();

router.use(express.json()); //json 형식으로 들어올 것이다.
//좋아요 추가
router.post('/:id', (req, res) => {
    res.json('좋아요 추가')
});
//좋아요 삭제
router.post('/:id' , (req, res) => {
    res.json('좋아요 삭제')
});

```

```
module.exports = router;
```

orders.js

```
const express = require("express");
const router = express.Router();

router.use(express.json()); //json 형식으로 들어올 것이다.
//주문하기
router.post('/', (req, res) => {
    res.json('주문하기')
});
//주문 목록 조회
router.get('/', (req, res) => {
    res.json('주문 목록 조회')
});
//주문 상세 상품 조회
router.get('/:id', (req, res) => {
    res.json('주문 상세 상품 조회')
});
//장바구니에서 선택한 주문 예상 상품 목록 조회
// router.get('/carts', (req, res) => {
//     res.json('장바구니 도서 삭제')
// });

module.exports = router;
```

carts.js

```
const express = require("express");
const router = express.Router();

router.use(express.json()); //json 형식으로 들어올 것이다.
//장바구니 담기
```

```
router.post('/', (req, res) => {
  res.json('장바구니 담기')
});
//장바구니 조회
router.get('/', (req, res) => {
  res.json('장바구니 조회')
});
//장바구니 도서 삭제
router.delete('/:id', (req, res) => {
  res.json('장바구니 도서 삭제')
});
//장바구니에서 선택한 주문 예상 상품 목록 조회
// router.get('/carts', (req, res) => {
//   res.json('장바구니 도서 삭제')
// });

module.exports = router;
```