아바라 한 잔 주세요☕

2022년 8월 3주차

목차

- 2022-08-08 (Mon) : 캡스톤디자인 README.md 작성
- 2022-08-09 (Tue) : HCS DB 구축
- 2022-08-10 (Wed) : HCS PR & 라즈베리파이 OS 설치
- 2022-08-11 (Thu) : swagger 사용법 학습 & VS Code 단축키 정리
- 2022-08-12 (Fri) : swagger 사용법 학습 -2 & swagger ui 출력

Sequelize 프로젝트 작성

Sequelize 프로젝트 구성

models > index.js > ...

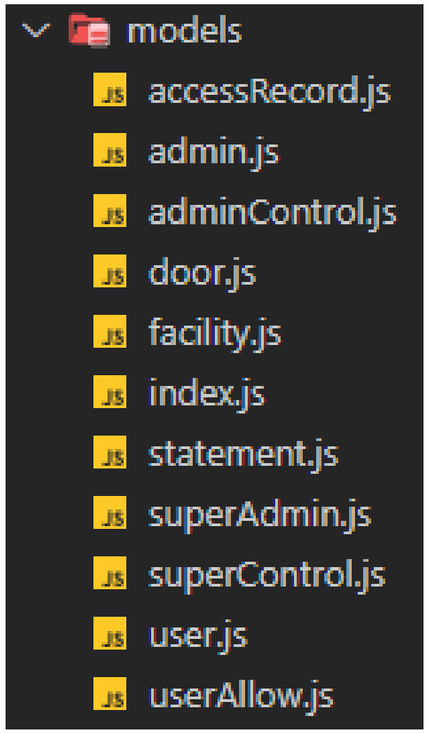
```
1   const Sequelize = require('sequelize');
2   const env = process.env.NODE_ENV || 'development'; // 지정된 환경변수가 없으면 'development'로 지정
3
4   // config/config.json 파일에 있는 설정값들을 불러온다.
5   // config객체의 env변수(development)키 의 객체값들을 불러온다.
6   // 즉, 데이터베이스 설정을 불러온다고 말할 수 있다.
7   const config = require("../config/config.json")[env]
8
9   const db = {};
10
11  // new Sequelize를 통해 MySQL 연결 객체를 생성한다.
12  const sequelize = new Sequelize(config.database, config.username, config.password, config)
13
14  // 연결객체를 나중에 재사용하기 위해 db.sequelize에 넣어둔다.
15  db.sequelize = sequelize;
16
17  // 모듈로 꺼낸다.
18  module.exports = db;
19
```

models/index.js 세팅

sequelize-cli가 기본적으로 생성하는 코드를 그대로 사용하면 에러가 발생하고,

불필요한 부분이 많으므로 다 지워주고 다음과 같이 세팅한다.

Sequelize 프로젝트 구성



models/모델.js 파일 작성

DB를 구성할 테이블을 하나의 js파일로 만들어 모듈화 시킨다.

Sequelize 프로젝트 구성

```
class statement extends Sequelize.Model {

    static init(sequelize) {
        return super.init({
                staId: {
                    type: Sequelize.UUID,
                    allowNull: false,
                    primaryKey: true,
                },
                staName: {
                    type: Sequelize.STRING(45),
                    allowNull: false,
                },
            },{
                sequelize,
                timestamps: false,
                underscored: false,
                modelName: 'statement',
                tableName: 'statement',
                paranoid: false,
                charset: 'utf8',
                collate: 'utf8_general_ci',
            });
    }
```

모델 파일 작성 요령

init 메서드의 첫 번째 필드에는 테이블의 컬럼을 작성하고, 타입, NULL 허용 여부,
기본값 등의 설정을 한다.
두 번째 필드는 테이블 속성에 대한 설정값을 작성한다.

Sequelize 프로젝트 구성

```
    static associate(db) {
        db.statement.belongsTo(db.facility, {
            foreignKey: 'facId',
            targetKey: 'facId',
            onDelete: 'cascade',
            onUpdate: 'cascade',
        });
        db.statement.hasMany(db.door, {
            foreignKey: 'staId',
            sourceKey: 'staId',
            onDelete: 'cascade',
            onUpdate: 'cascade',
        });
    }
};

module.exports = statement;
```

모델 파일 작성 요령

assciate 메서드에는 다른 테이블과의 관계를 설정한다.
1:1, 1:N, N:M 등 관계에 따라
belongsTo, hasMany, hasOne을 선택하여 작성

Sequelize 프로젝트 구성

```
const sequelize = new Sequelize(config.database, config.username, config.password, config)

// 연결객체를 나중에 재사용하기 위해 db.sequelize에 넣어둔다.
db.sequelize = sequelize;

// 모델 클래스를 넣음.
db.facility     = facility;
db.statement    = statement;
db.door         = door;
db.user         = user;
db.accessRecord = accessRecord;
db.superAdmin   = superAdmin;
db.superControl = superControl;
db.admin        = admin;
db.adminControl = adminControl;
db.userAllow    = userAllow;

// 모델과 테이블 종합적인 연결이 설정된다.
facility.init(sequelize);
statement.init(sequelize);
door.init(sequelize);
user.init(sequelize);
accessRecord.init(sequelize);
superAdmin.init(sequelize);
superControl.init(sequelize);
admin.init(sequelize);
adminControl.init(sequelize);
userAllow.init(sequelize);

// db객체 안에 있는 모델들 간의 관계가 설정된다.
facility.associate(db);
statement.associate(db);
door.associate(db);
user.associate(db);
accessRecord.associate(db);
superAdmin.associate(db);
superControl.associate(db);
admin.associate(db);
adminControl.associate(db);
userAllow.associate(db);

// 모듈로 꺼낸다.
module.exports = db;
```

models/index.js에 모델 추가

index.js에 모델을 추가하여 모델.init을 수행하면 테이블이 생성되고
모델.associate를 수행하면 테이블 간의 관계, 즉 테이블의 외래키가 생성된다.

Sequelize 프로젝트 구성

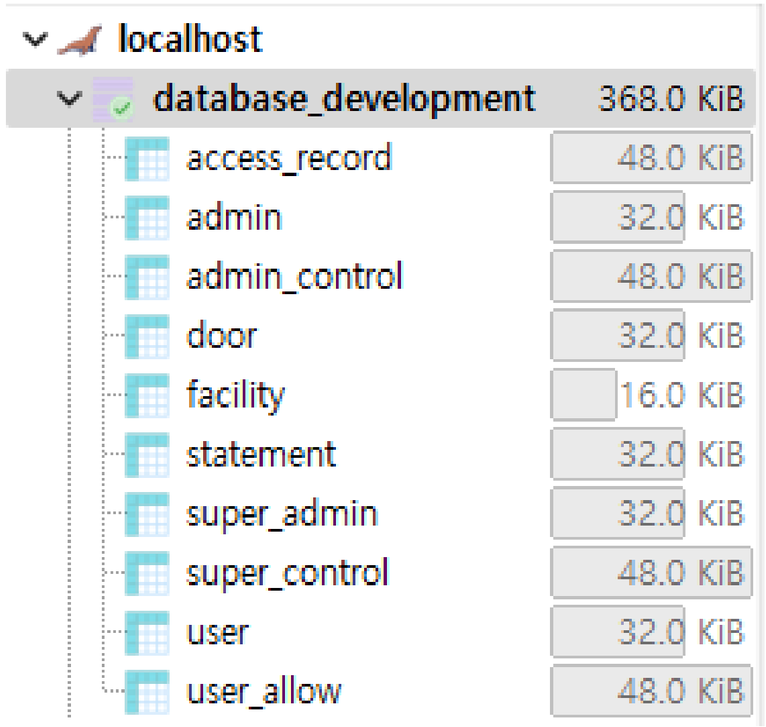
```
sequelize
    //? force: true 옵션은 모델 수정 시 db에 반영
    //? 테이블 삭제 후 다시 생성하기 때문에 데이터가 삭제됨
    //? alter: true 옵션을 통해 기존 데이터를 유지하면서 테이블 업데이트 가능
    //? 단, 필드를 새로 추가할때 필드가 NULL값을 허용하지 않을 시 오류 발생하므로 대처 필요
    .sync({ force: false })
    .then(() => {
        console.log('데이터베이스 연결 성공');
    }).catch((err) => {
        console.error(err);
    });
```

src/app.js에서 DB 모듈 초기화

전체 서버 시퀀스가 실행되는 app.js 에서 sequelize.sync를 수행하면
실제로 DB 테이블이 생성되는 것을 확인 할 수 있다.

생성된 DB 테이블

```
Executing (default): CREATE TABLE IF NOT EXISTS `admin` (`adminId` CHAR(36) BINARY NOT NULL , `adminName` VARCHAR(45) NOT NULL, `position` VARCHAR(45) NOT NULL, `adminLoginId` VARCHAR(45) NOT NULL UNIQUE, `adminLoginPw` VARCHAR(45) NO
T NULL, `phoneNum` VARCHAR(45) NOT NULL, PRIMARY KEY (`adminId`)) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE utf8_general_ci;
Executing (default): SHOW INDEX FROM `admin` FROM `database_development`
Executing (default): SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_NAME = 'admin_control' AND TABLE_SCHEMA = 'database_development'
Executing (default): CREATE TABLE IF NOT EXISTS `admin_control` (`controlId` CHAR(36) BINARY NOT NULL , `doorId` CHAR(36) BINARY, `adminId` CHAR(36) BINARY, PRIMARY KEY (`controlId`), FOREIGN KEY (`doorId`) REFERENCES `door` (`doorId`
) ON DELETE CASCADE ON UPDATE CASCADE, FOREIGN KEY (`adminId`) REFERENCES `admin` (`adminId`) ON DELETE CASCADE ON UPDATE CASCADE) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE utf8_general_ci;
Executing (default): SHOW INDEX FROM `admin_control` FROM `database_development`
Executing (default): SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_NAME = 'user_allow' AND TABLE_SCHEMA = 'database_development'
Executing (default): CREATE TABLE IF NOT EXISTS `user_allow` (`allowId` CHAR(36) BINARY NOT NULL , `doorId` CHAR(36) BINARY, `userId` CHAR(36) BINARY, PRIMARY KEY (`allowId`), FOREIGN KEY (`doorId`) REFERENCES `door` (`doorId`) ON DEL
ETE CASCADE ON UPDATE CASCADE, FOREIGN KEY (`userId`) REFERENCES `user` (`userId`) ON DELETE CASCADE ON UPDATE CASCADE) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE utf8_general_ci;
Executing (default): SHOW INDEX FROM `user_allow` FROM `database_development`
데이터베이스 연결 성공
```




```
USE database_development;
DESC statement;
```

Field	Type	Null	Key	Default	Extra
staId	char(36)	NO	PRI	(NULL)	
staName	varchar(45)	NO		(NULL)	
facId	char(36)	YES	MUL	(NULL)	

Swagger 사용법

Swagger란?

API 명세서 작성을 대신 해주는 UI라이브러리

API 명세서를 개발자가 직접 작성하면, 개발 내용이 수정되었을 때 일일히
API 명세서를 수정해주어야한다.

Swagger는 개발자가 프로젝트에 정의한 API 파일을 바탕으로 API를 UI형태로
제공하는 라이브러리이다.

Swagger 설치

```
npm i swagger-jsdoc swagger-ui-express --save-dev
```

Swagger 패키지 설치

터미널에 다음 명령어를 입력한다.

Swagger 설치

```
const swaggerUi = require('swagger-ui-express');
const swaggereJsdoc = require('swagger-jsdoc');

const options = {
    swaggerDefinition: {
        info: {
            title: 'Test API',
            version: '1.0.0',
            description: 'Test API with express',
        },
        host: 'localhost:3300',
        basePath: '/'
    },
    apis: ['./api/*.js', './swagger/*']
};

const specs = swaggereJsdoc(options);

module.exports = {
    swaggerUi,
    specs
};
```

src/swagger.js 파일 생성

swagger 모듈을 불러오고, 옵션을 설정하는 swagger.js 파일을 생성한다.

Swagger 설치

```
const swaggerUi = require('swagger-ui-express');
const swaggereJsdoc = require('swagger-jsdoc');

const options = {
    swaggerDefinition: {
        info: {
            title: 'Test API',
            version: '1.0.0',
            description: 'Test API with express',
        },
        host: 'localhost:3300',
        basePath: '/'
    },
    apis: ['./api/*.js', './swagger/*']
};

const specs = swaggereJsdoc(options);

module.exports = {
    swaggerUi,
    specs
};
```

src/swagger.js 파일 생성

swagger 모듈을 불러오고, 옵션을 설정하는 swagger.js 파일을 생성한다.

Swagger 설치

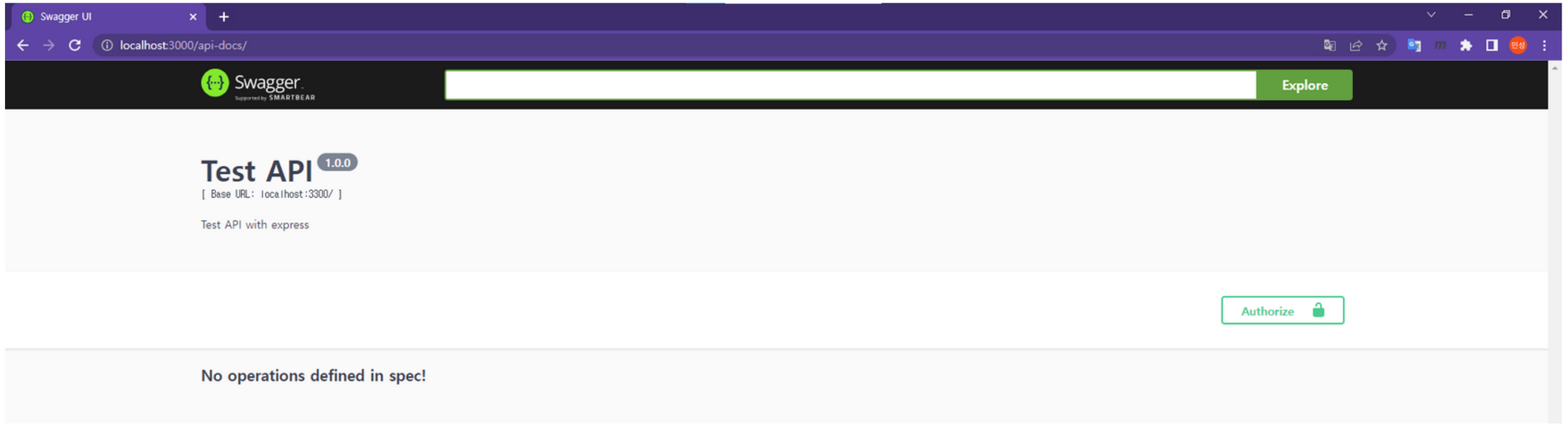
```
const { swaggerUi, specs } = require('./swagger');
```

```
app.use('api-docs', swaggerUi.serve, swaggerUi.setup(specs));
```

src/app.js에 등록

app.js에 swagger.js를 등록하고 미들웨어를 추가해준다.
이때 API Route 미들웨어 코드보다 위에 추가해주도록 한다.

Swagger 설치



localhost:3000/api-docs/ 접속

반드시 localhost:서버포트번호/api-docs/로 접속해야 swagger ui 페이지에
정상 접속 되므로 왜 접속이 안되는지 헤메지 말자.

2022년 8월 3주차 정리

학습내용 : Sequelize, swagger, Markdown

DB구축, swagger 사용환경 구축, README 작성

다음 학습 내용 : API 작성, 웹 통신

웹 API 작성, React.js <-> Node.js 간 통신

미비한 점 : README 작성, Swagger 적용

실제 API 작성필요, 깃헙 모든 레포지토리 README 정리 필요

😍Thanks for Watching!!😍

참고자료

Sequelize

Dev Scroll | ORM- -시퀄라이즈-모델-정의하기
https://inpa.tistory.com/entry/ORM- -시퀄라이즈-모델-정의하기

Swagger

pageseo | Node.js + Swagger, 어렵지 않게 사용하기
https://gngsn.tistory.com/69

Markdown

https://m.blog.naver.com/jooeun0502/221956294941
https://inasie.github.io/it일반/마크다운-표-만들기/

무료 자원

Canva 프레젠테이션에
다음 콘텐츠를 사용하세요.