**[7장. MYSQL]**

mysql 설치(생략) 후,

**6. 시퀄라이즈 사용하기**

$ **cd test\_kes**

Administrator@KESNOT MINGW64 /c/materK1m/newmimac/test\_kes (master)

$ **express learn-sequelize --view=pug**

create : learn-sequelize\

create : learn-sequelize\public\

create : learn-sequelize\public\javascripts\

create : learn-sequelize\public\images\

create : learn-sequelize\public\stylesheets\

create : learn-sequelize\public\stylesheets\style.css

create : learn-sequelize\routes\

create : learn-sequelize\routes\index.js

create : learn-sequelize\routes\users.js

create : learn-sequelize\views\

create : learn-sequelize\views\error.pug

create : learn-sequelize\views\index.pug

create : learn-sequelize\views\layout.pug

create : learn-sequelize\app.js

create : learn-sequelize\package.json

create : learn-sequelize\bin\

create : learn-sequelize\bin\www

change directory:

$ cd learn-sequelize

install dependencies:

$ npm install

run the app:

$ DEBUG=learn-sequelize:\* npm start

Administrator@KESNOT MINGW64 /c/materK1m/newmimac/test\_kes (master)

$ **cd learn-sequelize**

Administrator@KESNOT MINGW64 /c/materK1m/newmimac/test\_kes/learn-sequelize (master)

$ **npm i**

npm notice created a lockfile as package-lock.json. You should commit this file.

added 118 packages from 174 contributors and audited 247 packages in 10.254s

found 1 low severity vulnerability

run `npm audit fix` to fix them, or `npm audit` for details

Administrator@KESNOT MINGW64 /c/materK1m/newmimac/test\_kes/learn-sequelize (master)

$ **npm i sequelize mysql2**

+ mysql2@1.6.5

+ sequelize@5.1.1

added 33 packages from 102 contributors and audited 287 packages in 7.84s

found 1 low severity vulnerability

run `npm audit fix` to fix them, or `npm audit` for details

Administrator@KESNOT MINGW64 /c/materK1m/newmimac/test\_kes/learn-sequelize (master)

$

Administrator@KESNOT MINGW64 /c/materK1m/newmimac/test\_kes/learn-sequelize (master)

$ **npm i -g sequelize-cli**

C:\Users\Administrator\AppData\Roaming\npm\sequelize -> C:\Users\Administrator\AppData\Roaming\npm\node\_modules\sequelize-cli\lib\sequelize

+ sequelize-cli@5.4.0

added 106 packages from 47 contributors in 9.086s

Administrator@KESNOT MINGW64 /c/materK1m/newmimac/test\_kes/learn-sequelize (master)

$ **sequelize init**

Sequelize CLI [Node: 11.7.0, CLI: 5.4.0, ORM: 5.1.1]

Created "config\config.json"

Successfully created models folder at "C:\materK1m\newmimac\test\_kes\learn-sequelize\models".

Successfully created migrations folder at "C:\materK1m\newmimac\test\_kes\learn-sequelize\migrations".

Successfully created seeders folder at "C:\materK1m\newmimac\test\_kes\learn-sequelize\seeders".

Administrator@KESNOT MINGW64 /c/materK1m/newmimac/test\_kes/learn-sequelize (master)

**config, models, migrations, seeders 폴더가 생성됩니다.**

* **model/index.js**

'use strict';

const fs = require('fs');

const path = require('path');

const Sequelize = require('sequelize');

const basename = path.basename(\_\_filename);

const env = process.env.NODE\_ENV || 'development';

const config = require(\_\_dirname + '/../config/config.json')[env];

const db = {};

let sequelize;

if (config.use\_env\_variable) {

sequelize = new Sequelize(process.env[config.use\_env\_variable], config);

} else {

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

}

fs

.readdirSync(\_\_dirname)

.filter(file => {

return (file.indexOf('.') !== 0) && (file !== basename) && (file.slice(-3) === '.js');

})

.forEach(file => {

const model = sequelize['import'](path.join(\_\_dirname, file));

db[model.name] = model;

});

Object.keys(db).forEach(modelName => {

if (db[modelName].associate) {

db[modelName].associate(db);

}

});

db.sequelize = sequelize;

db.Sequelize = Sequelize;

module.exports = db;

필요 없는 소스 제거

const path = require('path');

const Sequelize = require('sequelize');

const env = process.env.NODE\_ENV || 'development';

const config = require(\_\_dirname + '/../config/config.json')[env];

const db = {};

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

db.sequelize = sequelize;

db.Sequelize = Sequelize;

module.exports = db;

* app.js

mysql 연동

var createError = require('http-errors');

var express = require('express');

var path = require('path');

var cookieParser = require('cookie-parser');

var logger = require('morgan');

var indexRouter = require('./routes/index');

var usersRouter = require('./routes/users');

**var sequelize = require('./**models').sequelize;

var app = express();

**sequelize.sync();**

* models/user.js

모델 정의하기

module.exports = (sequelize, DataTypes) => {

return sequelize.define('user', {

name : {

type : DataTypes.STRING(20),

allowNull : false,

unique : true,

},

age : {

type : DataTypes.INTEGER.UNSIGNED,

allowNull : false,

},

married : {

type : DataTypes.BOOLEAN,

allowNull : false,

},

comment : {

type : DataTypes.TEXT,

allowNull : true,

},

create\_at : {

type : DataTypes.DATE,

allowNull : false,

defaultValue : sequelize.literal('now()'),

},

}, {

timestamp : false,

});

}

‘

* models/comment.js

module.exports = (sequelize, DataTypes) => {

return sequelize.define('commit', {

comment : {

type : DataTypes.STRING(100),

allowNull : false,

},

create\_at : {

type : DataTypes.DATE,

allowNull : true,

defaultValue : sequelize.literal('now()'),

},

},{

timestamp : false,

});

};

* models/index.js

연결 하기

const path = require('path');

const Sequelize = require('sequelize');

const env = process.env.NODE\_ENV || 'development';

const config = require(\_\_dirname + '/../config/config.json')[env];

const db = {};

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

db.sequelize = sequelize;

db.Sequelize = Sequelize;

**db.User = require('./user')(sequelize, Sequelize);**

**db.Comment = require('./comment')(sequelize, Sequelize);**

module.exports = db;

* config/config.json

{

"development": {

"username": "root",

"password": "pw123",

"database": "nodejs",

"host": "127.0.0.1",

"dialect": "mysql",

"operatorsAliases" : false

},

"test": {

"username": "root",

"password": null,

"database": "database\_test",

"host": "127.0.0.1",

"dialect": "mysql"

},

"production": {

"username": "root",

"password": null,

"database": "database\_production",

"host": "127.0.0.1",

"dialect": "mysql"

}

}

* models/index.js

1:N 관계 정의 하기

db.User = require('./user')(sequelize, Sequelize);

db.Comment = require('./comment')(sequelize, Sequelize);

**db.User.hasMany(db.Comment, {foreignKey : 'commenter', sourceKey : 'id'});**

**db.Comment.belongsTo(db.User, {foreignKey : 'commenter', targetKey : 'id'});**

C:\materK1m\newmimac\test\_kes\learn-sequelize>npm start

> learn-sequelize@0.0.0 start C:\materK1m\newmimac\test\_kes\learn-sequelize

> node ./bin/www

(sequelize) Warning: A boolean value was passed to options.operatorsAliases. This is a no-op with v5 and should be removed.

Executing (default): CREATE TABLE IF NOT EXISTS `users` (`id` INTEGER NOT NULL auto\_increment , `name` VARCHAR(20) NOT NULL UNIQUE, `age` INTEGER UNSIGNED NOT NULL, `married` TINYINT(1) NOT NULL, `comment` TEXT, `create\_at` DATETIME NOT NULL DEFAULT now(), `createdAt` DATETIME NOT NULL, `updatedAt` DATETIME NOT NULL, PRIMARY KEY (`id`)) ENGINE=InnoDB;

Executing (default): SHOW INDEX FROM `users` FROM `nodejs`

Executing (default): CREATE TABLE IF NOT EXISTS `commits` (`id` INTEGER NOT NULL auto\_increment , `comment` VARCHAR(100) NOT NULL, `create\_at` DATETIME DEFAULT now(), `createdAt` DATETIME NOT NULL, `updatedAt` DATETIME NOT NULL, `commenter` INTEGER, PRIMARY KEY (`id`), FOREIGN KEY (`commenter`) REFERENCES `users` (`id`) ON DELETE SET NULL ON UPDATE CASCADE) ENGINE=InnoDB;

Executing (default): SHOW INDEX FROM `commits` FROM `nodejs`

* 관계 정의하기

1:N 관계 정의

db.User.hasMany(db.Comment, {foreignKey : 'commenter', sourceKey : 'id'});

db.Comment.belongsTo(db.User, {foreignKey : 'commenter', targetKey : 'id'});

1:1 관계 정의

db.User.hasOne(db.Info, {foreignKey : ‘user\_id’, sourceKey : 'id'});

db.Info.belongsTo(db.User, {foreignKey : ‘user\_id’, targetKey : 'id'});

N:M 관계 정의

db.Post.belonsToMany(db.Hashtag, { through : ‘PostHashtag’ });

db.Hashtag.belonsToMany(db.Post, { through : ‘PostHashtag’ });

N:M 경우 몇가지 메소드

1. async/await 코드

- 해시태그를 찾으면 그 해시태그에서 바로 getPost 메서드를 사용할 수 있다. get+모델이름의 복수형

async (req, res, next) => {

const tag = await Hashtag.find({where : {title : ‘노드’}});

const posts = await tag.getPosts();

};

2. 프로미스 형식

HashTag.find({ Where : {title : ‘노드’}})

.then(tag => tag.getPosts())

.then(posts => console.log(posts));

3. PostHashtag 모델에 postId가 3이고 hashtagId가 1인 로우 생성

async (req, res, next) => {

const tag = await Hashtag.find({where : {title : ‘노드’}});

await tag.setPost(3);

};

4. 쿼리문 생성

sql : INSERT INTO nodejs.users (name, age, married, comment) VALUES (‘zero’, 24, 0, ‘자기소개1’);

const {User} = require(‘../models’)

User.create({

name : ‘zero’,

age : 24,

married : false,

comment : ‘자기소개1’,

});

sql : SELECT \* FROM nodejs.users;

User.findAll({});

sql : SELECT \* FROM nodejs.users LIMIT 1;

User.find({});

sql : SELECT name, married FROM nodejs.users;

User.findAll({

attributes : [‘name’, ‘married’],

});

sql : SELECT name, age FROM nodejs.users WHERE married = 1 AND age > 30;

const { User, Sequelize : {Op}} = require(‘../modls’);

User.findAll({

attributes : [name, age],

where : {

married : 1,

age : { [Op.gt] : 30},

},

});

Op.gt(초과), Op.gte(이상), Op.lt(미만), Op.lte(이하), Op.ne(같지 않음), Op.or(또는), Op.in(배열 요소 중 하나), Op.notIn(배열 요소와 모두 다름) 등

sql : SELECT id, name FROM users WHERE married = 0 OR age > 30;

const { User, Sequelize : {Op}} = require(‘../modls’);

User.findAll({

attributes : [name, age],

where : {

[Op.or] : [{married :0}, {age:{[Op.gt]:30}}],

},

});

sql : SELECT id, name FROM users ORDER BY age DESC;

User.findAll({

attributes : [‘id’, ‘name’],

order : [[‘age’, ‘DESC’]],

});

sql : SELECT id, name FROM users ORDER BY age DESC LIMIT 1;

User.findAll({

attributes : [‘id’, ‘name’],

order : [‘age’, ‘DESC’],

limit :1,

});

sql : SELECT id, name FROM users ORDER BY age DESC LIMIT 1; OFFSET 1

User.findAll({

attributes : [‘id’, ‘name’],

order : [‘age’, ‘DESC’],

limit :1,

offset : 1,

});

sql : UPDATE nodejs.users SET comment=’바꿀 내용’ WHERE id = 2;

User.update({

comment : ‘바꿀 내용’,

}, {

where : {id : 2},

});

sql : DELETE FROM nodejs.users WHERE ID = 2;

User.destory({

where : {id :2},

});