Seoul woman up

스타트업 개발자와 함께 공부하는

Node.js

08. ORM

강의 내용은 강사가 별도로 명시하지 않는 한 비공개로 간주합니다. 녹음이나 사진 촬영를 허락하지 않으며 콘텐츠를 블로그, SNS 등에 게시하거나 공개적으로 공유하지 마세요.

콘텐츠 공유 가능 여부에 대해 궁금한 점이 있는 경우 강사에게 문의하시기 바랍니다.

목차

Seoul woman up

ORM

ORM

등장 배경

- ORM은 Object-Relational Mapping 약자
- 전통적으로 SQL 쿼리를 통해 데이터베이스에서 데이터를 가져오고 애플리케이션에서 사용
- SQL 쿼리와 프로그래밍 언어 코드 간의 불일치

ORM

소개

- ORM은 객체 지향 프로그래밍 언어에서 데이터베이스를 쉽게 사용할 수 있게 함
- 데이터베이스의 테이블을 애플리케이션의 클래스로 매핑
- 테이블 행을 클래스 인스턴스로 매핑
- SQL 쿼리를 직접 작성 하는 대신, 클래스와 객체로 데이터베이스와 상호작용

ORM

소개

❑ 장점

1. 생산성 증가
2. 유지보수 용이
3. 데이터베이스 독립성
4. 보안

❑ 단점

1. 성능 저하
2. 복잡한 쿼리 처리 어려움
3. 러닝 커브

ORM

종류

- Python : Django ORM, SQLAlchemy
- Javascript/Node.js : Sequelize, TypeORM, Prisma
- Java : Hibernate, JPA
- Go : GoORM
- Ruby : ActiveRecord

Seoul woman up

Sequelize ORM

Sequelize ORM

소개

- Node.js용 ORM 라이브러리
- 다양한 SQL 데이터베이스(MySQL, PostgreSQL, SQLite, MariaDB 등)
- 객체 지향 프로그래밍 모델을 사용하여 데이터베이스와 상호작용

Sequelize ORM

주요기능

- 모델 정의 : 테이블 == 모델
- 쿼리 빌딩 : `find, create, update, destroy`
- 관계 설정 : 1:1, 1:N, JOIN 쿼리 지원
- 트랜젝션 지원
- 마이그레이션 : 스키마 버전 관리 지원

Sequelize ORM

프로젝트 생성

```
npm init -y
npm i sequelize@6.37.6 sqlite3
```

- [C]-[nodejs]-[project]-[08]-[ch08_01]

Sequelize ORM

기본 사용법

```
// 설정
// ch08_01.js

1 const { Sequelize, Model, DataTypes }
2     = require('sequelize');
3 const sequelize = new Sequelize({
4     dialect: 'sqlite',
5     storage: 'sample.sqlite'
6 });
7
8 console.log(sequelize);
```

```
// 모델 정의
// ch08_02.js

10 const User = sequelize.define('User', {
11     username: {
12         type: DataTypes.STRING,
13         allowNull: false,
14     },
15     email: {
16         type: DataTypes.STRING,
17         allowNull: false
18     }
19 }, {freezeTableName : true});
```

Sequelize ORM

기본 사용법

```
// 동기화
// ch08_01.js
19 (async () => {
20
21     await sequelize.sync({force: true});
22
```

```
// 데이터 조작(생성)
23     const user = await User.create({
24         username: 'ethan_lee',
25         email : 'ethan.lee@gmail.com'
26     });
27     console.log(`user created => ${user}`);
28
```

Sequelize ORM

기본 사용법

// 조회

```
28
29      const users = await User.findAll();
30      console.log(`user findall => ${users}`);
31
```

// 수정

```
32      await User.update({
33          'email':'ethan.lee@naver.com'
34      },{
35          where : {
36              username : 'ethan_lee'
37          }
38      });
39      const updated_user = await User.findOne({
40          where : {username : 'ethan_lee'}
41      });
42      console.log(`user updated_user => ${updated_user}`);
```

Sequelize ORM

기본 사용법

// 삭제

```
44      await User.destroy({
45          where : {
46              username: 'ethan_lee'
47          }
48      });
49      const deleted_user = await User.findOne({
50          where : {username : 'ethan_lee'}
51      });
52      console.log(`user deleted_user => ${deleted_user}`);
53
54  })();
```

// 관계 설정

```
19  const Post = sequelize.define('Post', {
20      title: {
21        type: DataTypes.STRING,
22        allowNull: false
23      },
24      content: {
25        type: DataTypes.TEXT,
26        allowNull: false
27      }
28    });
29
30    User.hasMany(Post); // 1:N 관계 설정
31    Post.belongsTo(User); // N:1 관계 설정
```

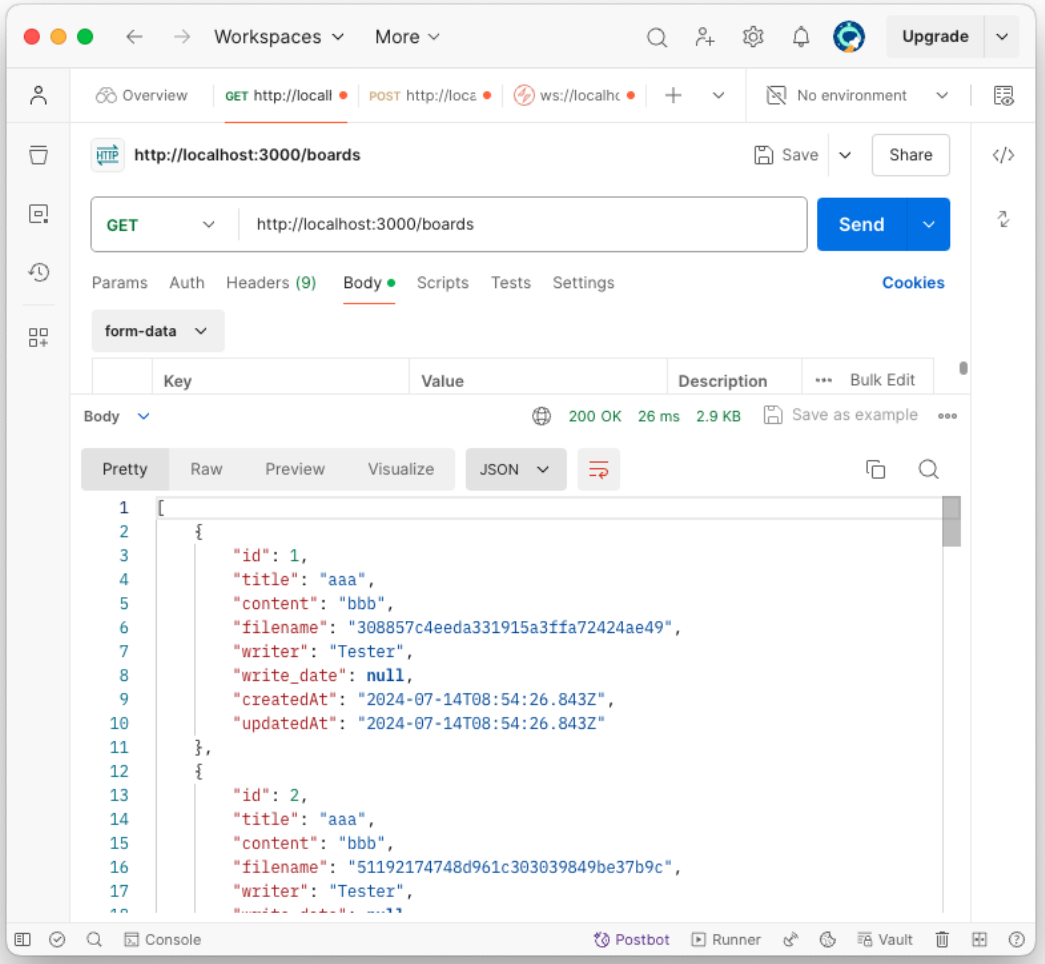
Seoul woman up

첨부파일 게시판 API

첨부파일 게시판 API

소개

- 첨부파일이 저장되는 게시판
- RESTful API 로만 서비스
- Postman 파일 등록 테스트



첨부파일 게시판 API

프로젝트 생성

```
npm init -y
npm i express multer nodemon sequelize
sequelize-cli sqlite3
npx sequelize init
```

- [C]-[nodejs]-[project]-[08]-[ch08_02]

첨부파일 게시판 API

config 및 모델

// config/config.json

```
{
  "development": {
    "dialect": "sqlite",
    "storage": "./board.sqlite3"
  },
  "test": {
    "dialect": "sqlite",
    "storage": "./board.sqlite3"
  },
  "production": {
    "dialect": "sqlite",
    "storage": "./board.sqlite3"
  }
}
```

// models/board.js

```
module.exports = (sequelize, DataTypes) => {
    const Board = sequelize.define('Board', {
        id: {
            type: DataTypes.INTEGER,
            allowNull: false,
            primaryKey: true,
            autoIncrement: true,
        },
        title: DataTypes.STRING,
        content: DataTypes.STRING,
        filename: {
            type: DataTypes.STRING,
            allowNull: true,
        },
        writer: DataTypes.STRING,
        write_date: DataTypes.DATE,
    }, {
        tableName: 'board',
        underscore: true,
    });
    return Board;
};
```

첨부파일 게시판 API

파일 업로드 설정

```
1  const express = require('express');
2  const fs = require('fs');
3  const path = require('path');
4  const models = require('./models');
5  const multer = require('multer');
6
7  const app = express();
8
9  app.use(express.json());
10 app.use(express.urlencoded({extended: true}));
11 app.use('/downloads',
12     express.static(path.join(__dirname, 'public/uploads')));
13 const upload_dir = `public/uploads`
14
15 const storage = multer.diskStorage({
16     destination: `./${upload_dir}`,
17     filename: function (req, file, cb) {
18         cb(null, path.parse(file.originalname).name + '-' +
19             Date.now() + path.extname(file.originalname));
20     }
21 });
22
23
24 const upload = multer({storage: storage});
```

첨부파일 게시판 API

목록

```
26 app.get('/boards', async (req, res) => {
27     const boards = await models.Board.findAll();
28     console.log(`All boards : ${boards}`);
29     res.json(boards);
30 });
```

첨부파일 게시판 API

목록

```
32 app.post('/boards', upload.single('file'), async (req, res) => {
33     console.log(req.body);
34     const {title, content} = req.body;
35     let filename = req.file ? req.file.filename : null;
36     filename = `/downloads/${filename}`;
37
38     const board = await models.Board.create({
39         title: title,
40         content: content,
41         filename: filename,
42         writer: 'Tester',
43         wirte_date: Date.now()
44     });
45     res.status(201).json(board);
46 });
```

첨부파일 게시판 API

상세

```
68 app.get('/boards/:id', async (req, res) => {
69     const id = req.params.id;
70     const board = await Board.findByPk(req.params.id);
71     if (board) {
72         res.json(board);
73     }else{
74         res.status(404),send('Board not found')
75     }
76 })
```

첨부파일 게시판 API

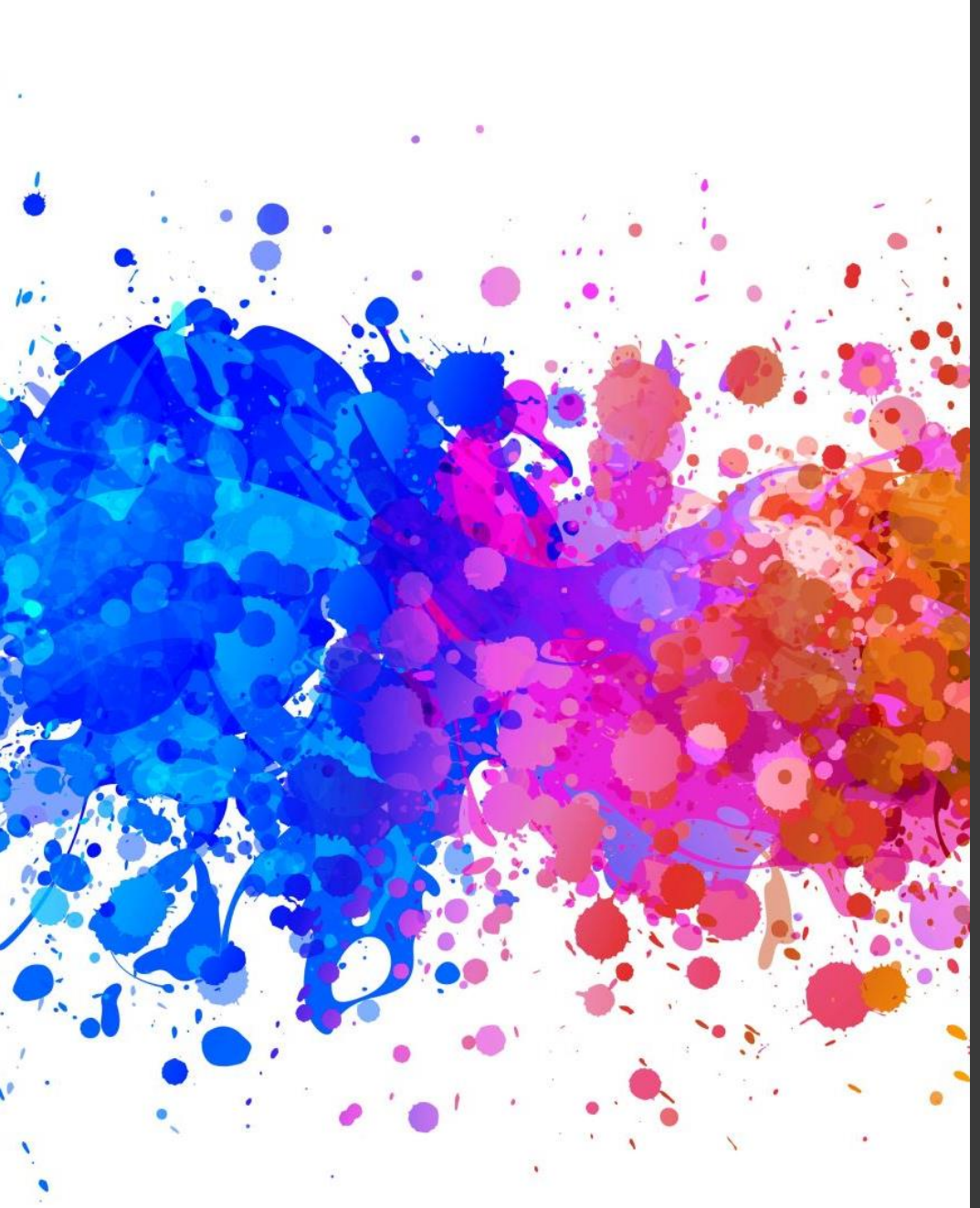
수정

```
48 app.put('/boards/:id', upload.single('file'),  async (req, res) => {
49     const id = req.params.id;
50     const {title, content} = req.body;
51     let filename = req.file ? req.file.filename : null;
52     filename = `/downloads/${filename}`;
53 
54     const board = await models.findByPk(req.params.id)
55     if(board) {
56         board.title = title;
57         board.content = content;
58         if(filename) {
59             board.filename = filename;
60         }
61         await board.save();
62         res.json(board);
63     }else {
64         res.status(404).send('Board not found')
65     }
66 });
```

첨부파일 게시판 API

삭제

```
78 app.delete('/boards/:id', async (req, res) => {
79     const result = await models.Board.destroy({
80         where : { id: req.params.id }
81     });
82
83     if(result) {
84         res.status(204).send();
85     }else {
86         res.status(404).send('Board not found');
87     }
88 });
```



Seoul woman up

multipart/form-data

multipart/form-data

소개

- HTTP POST 요청을 사용
- 클라이언트는 파일 업로드를 위해 Content-Type 헤더에
- multipart/form-data 설정
- 파일 및 기타 폼을 데이터 요청 본문에 포함

multipart/form-data

본문 형식

```
------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="field1"

value1
------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="file"; filename="example.txt"
Content-Type: text/plain

(File contents here)
------WebKitFormBoundary7MA4YWxkTrZu0gW--
```

Seoul woman up

백문이 불여일타