

10주차

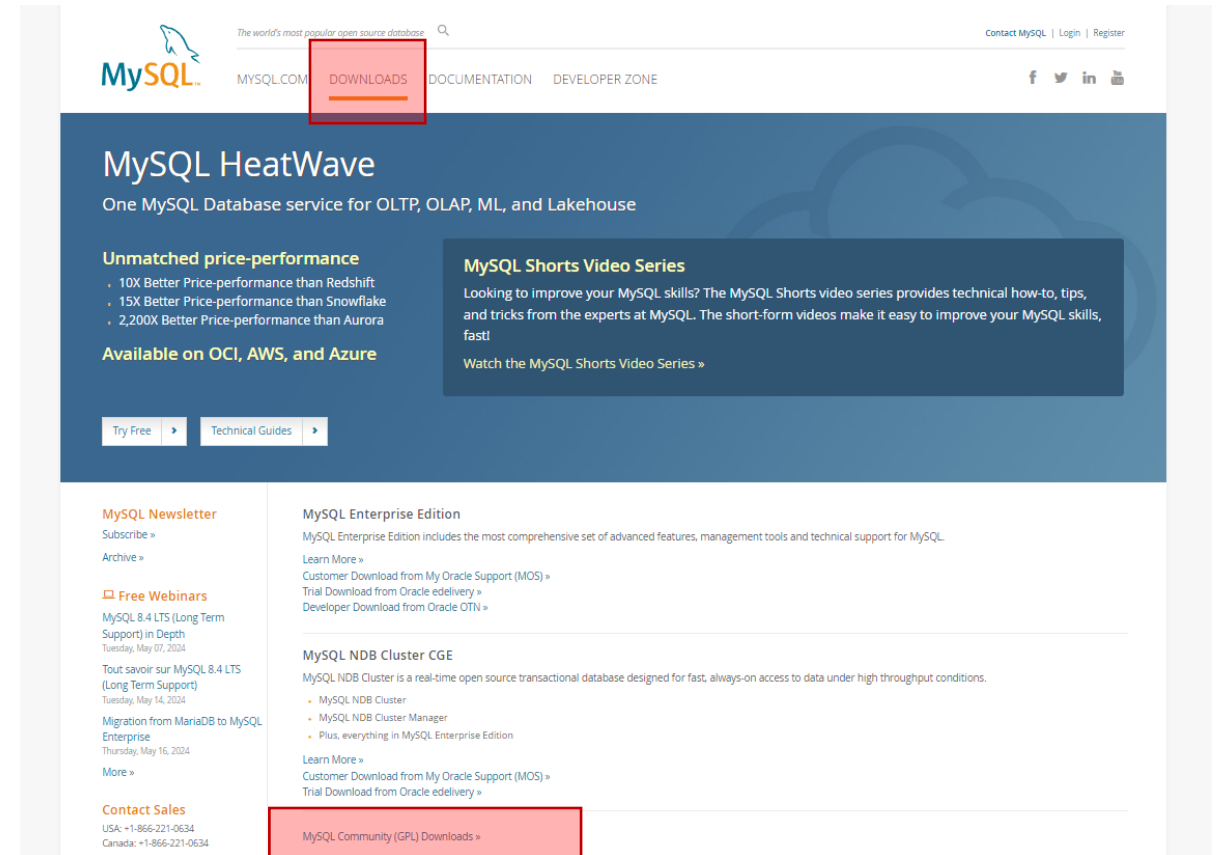
소프트웨어 시스템 설계 및 개발

2024.1학기

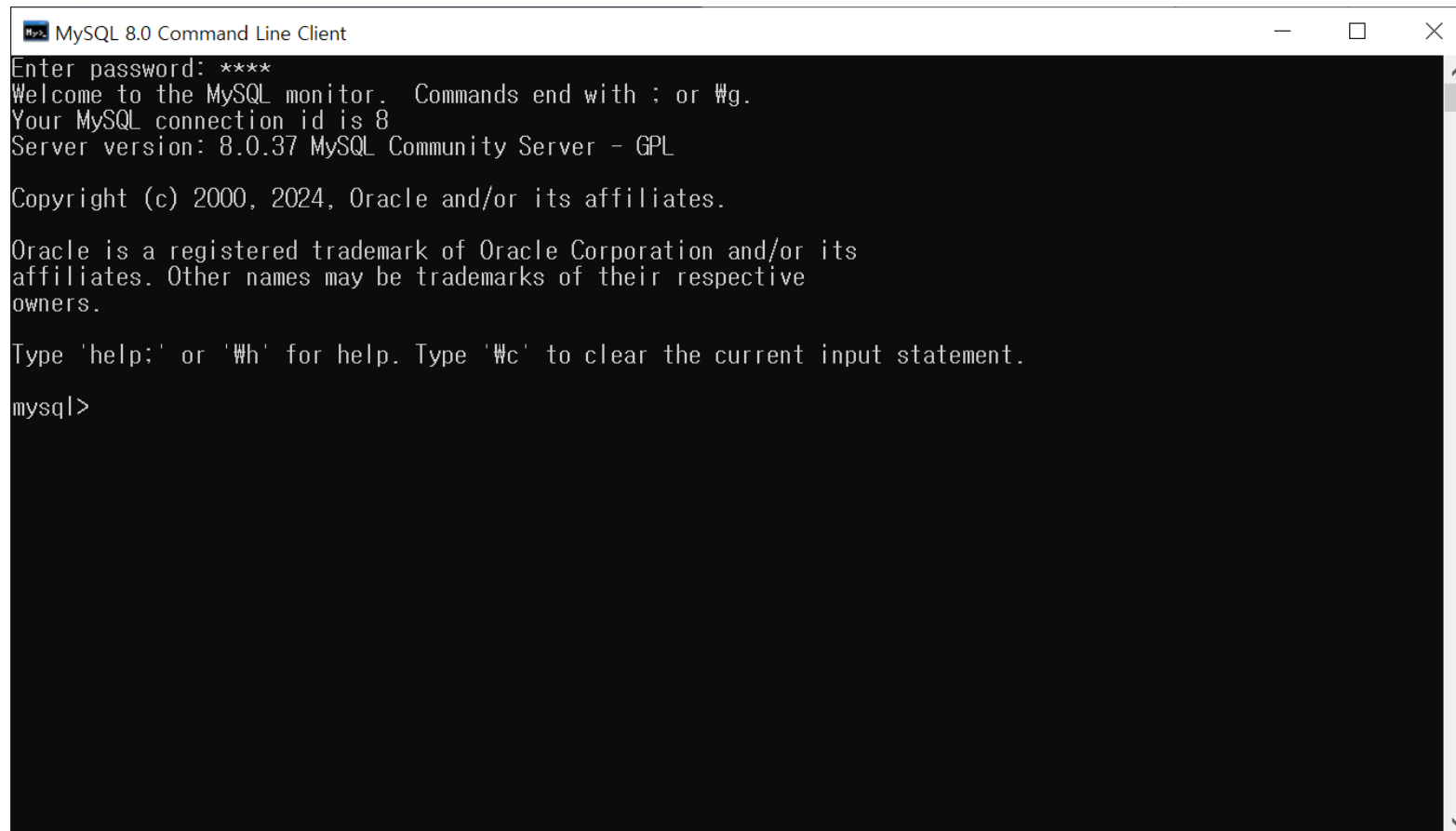
CONTENTS

1. DB(MySQL, MongoDB)
2. 아키텍처(express-generator 활용)
3. 실습

- 대표적인 오픈소스 데이터베이스
- 관계형 데이터베이스(RDBMS)에서는
오라클DB 다음으로 높은 점유율을 가지고 있음
- 유료 버전과 무료 버전이 있으며,
무료는 GPL 라이선스를 따름



- MySQL 설치 후 'MySQL Command Line Client' 실행 (for windows)



```
MySQL 8.0 Command Line Client
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.37 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

• MySQL 기본 명령어 (1/2)

내용	예시 코드
데이터베이스 생성	<code>CREATE DATABASE example_db;</code>
테이블 생성	<code>CREATE TABLE users (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100), email VARCHAR(100));</code>
데이터 삽입	<code>INSERT INTO users (name, email) VALUES ('Jonggyu', 'pjk5401@dau.ac.kr');</code>
데이터 조회	<code>SELECT * FROM users;</code>
데이터 조회(조건)	<code>SELECT * FROM users WHERE name = 'Jonggyu';</code>

- MySQL 기본 명령어 (2/2)

내용	예시 코드
데이터 업데이트	UPDATE users SET email = 'pjk5401@gmail.com' WHERE name = 'jonggyu';
데이터 삭제	DELETE FROM users WHERE name = 'jonggyu';
테이블 수정	ALTER TABLE users ADD COLUMN age INT;
테이블 삭제	DROP TABLE users;
데이터베이스 삭제	DROP DATABASE example_db;

예제 - MySQL

- NodeJS <-> MySQL 연결 예제

<< MySQL CLI에서 >>

1. Users DB 생성

- CREATE DATABASE user_db;

2. Users Table 생성

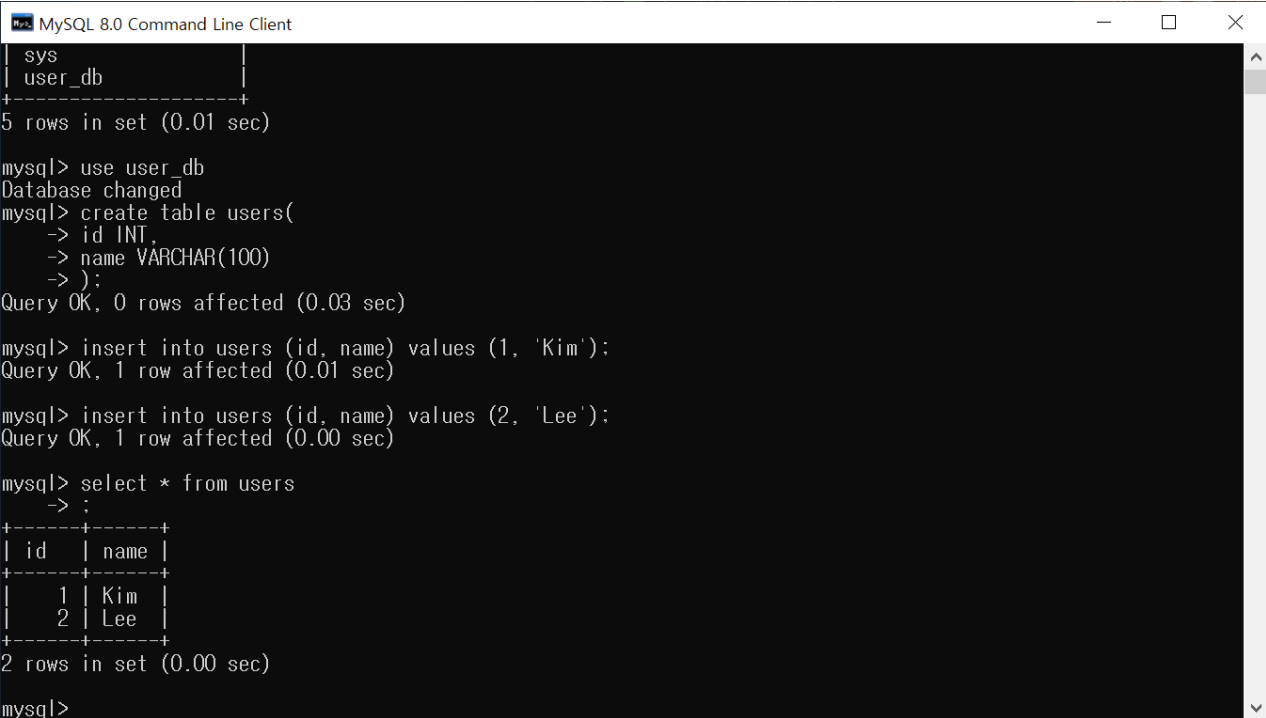
- CREATE TABLE users (
 id INT,
 name VARCHAR(100)
);

3. User Data 입력

- INSERT INTO users (id, name) VALUES (1, 'Kim');
- INSERT INTO users (id, name) VALUES (2, 'Lee');

4. User Data 조회

- SELECT * FROM users;



```
MySQL 8.0 Command Line Client
| sys |
| user_db |
+-----+
5 rows in set (0.01 sec)

mysql> use user_db
Database changed
mysql> create table users(
    -> id INT,
    -> name VARCHAR(100)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> insert into users (id, name) values (1, 'Kim');
Query OK, 1 row affected (0.01 sec)

mysql> insert into users (id, name) values (2, 'Lee');
Query OK, 1 row affected (0.00 sec)

mysql> select * from users
    -> ;
+----+-----+
| id | name |
+----+-----+
| 1  | Kim  |
| 2  | Lee  |
+----+-----+
2 rows in set (0.00 sec)

mysql>
```

예제 - MySQL

- NodeJS <-> MySQL 연결 예제
- << 7주차 app.js에서 변경(일부) >>

```
// npm install express
const express = require('express');
const app = express();
const port = 3000;

// npm install mysql
var mysql = require('mysql');
var db = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '1234',
  database: 'user_db'
});
db.connect((err) => {
  if (err) {
    throw err;
  }
  console.log('Connected to database');
});

app.use(express.json()); // JSON 파싱 미들웨어

// 초기 사용자 데이터
// let users = [
//   { id: 1, name: 'Kim' },
//   { id: 2, name: 'Lee' },
//   { id: 3, name: 'Park' }];
```

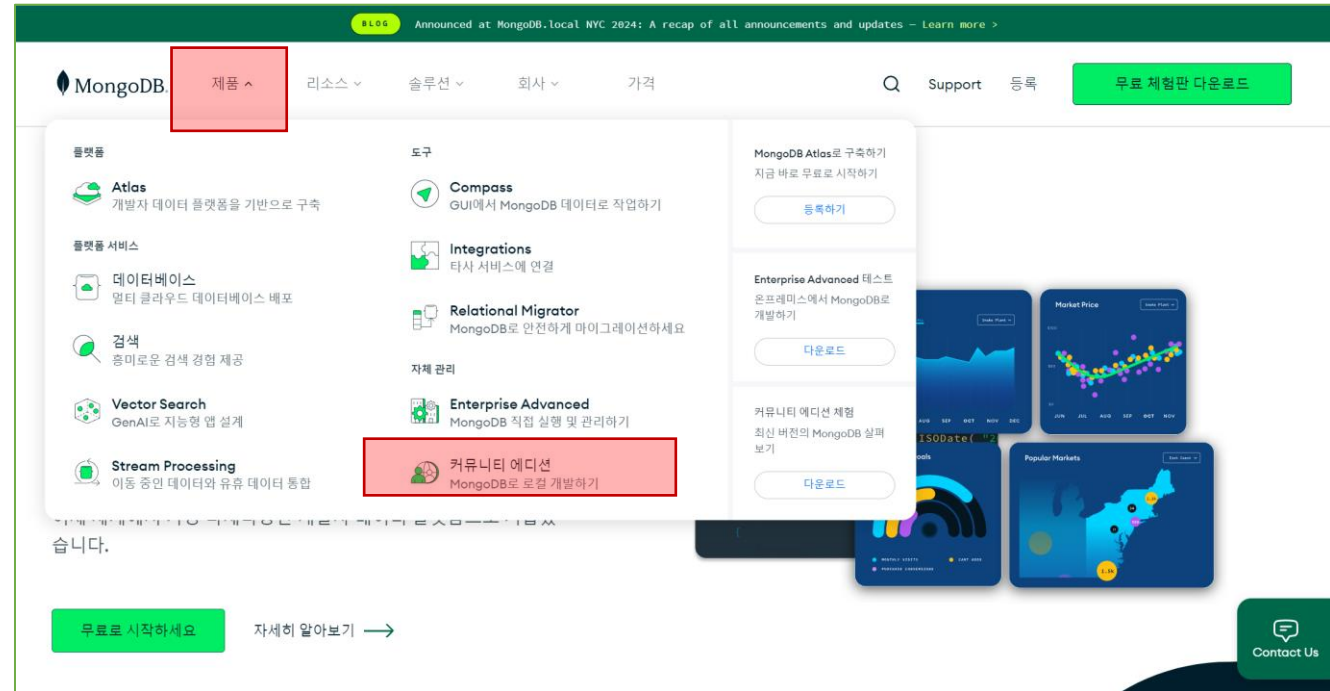
```
// 모든 사용자 정보 조회
app.get('/users', (req, res) => {
  db.query('select * from users', (err, result) => {
    if (err){
      console.log(err);
    }
    console.log(result);
    res.status(200).json(result);
  });
});

// 특정 사용자 정보 조회, :id는 변수처럼 사용한다는 의미
// 해당 위치에 id 값인 1, 2, 3등을 넣으면 됨 /users/1, /users/2 이런 식
app.get('/users/:id', (req, res) => {
  db.query(`select * from users where id = ${req.params.id}`, (err, result) => {
    if (err){
      console.log(err);
    }
    console.log(result);
    res.status(200).json(result);
  });
});

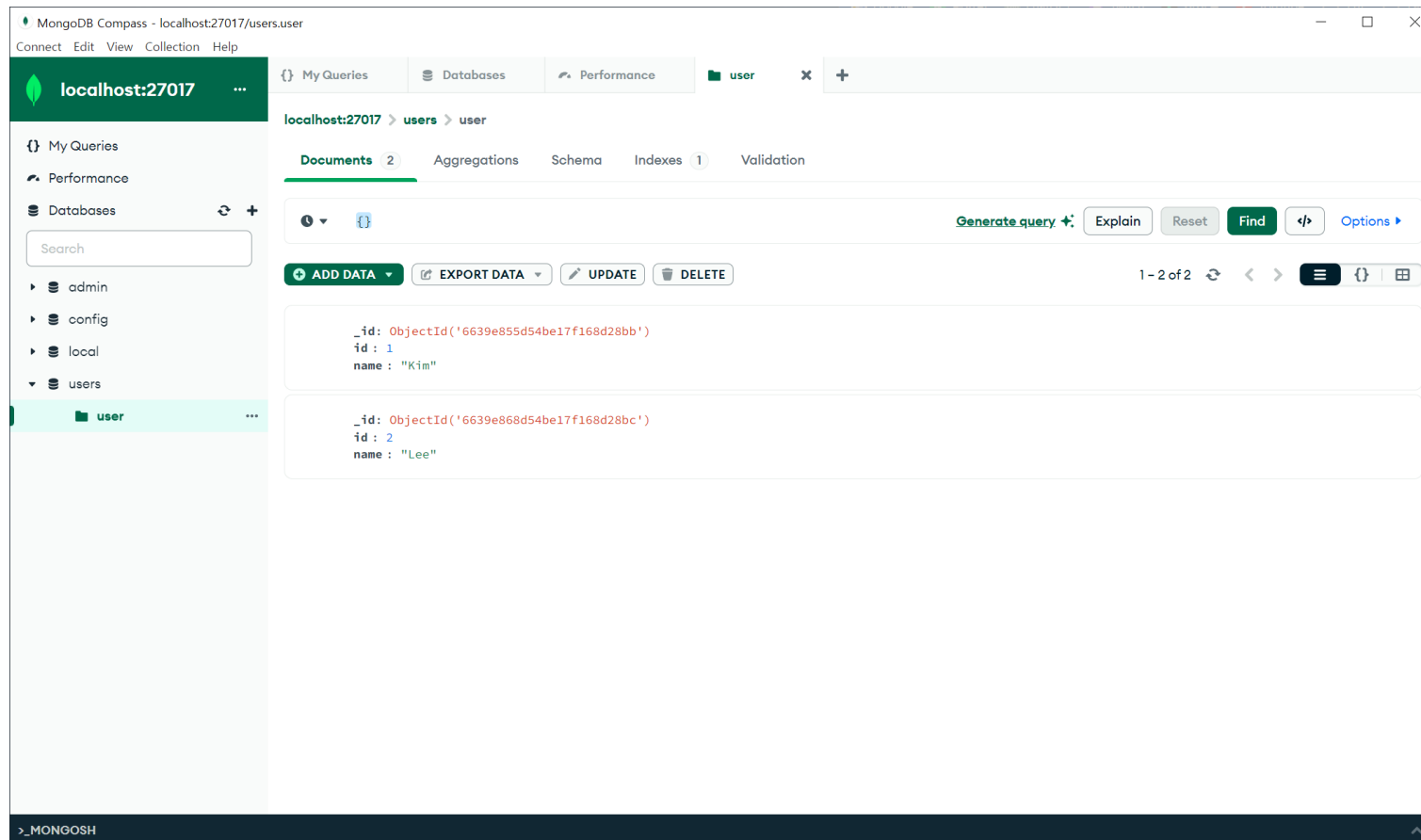
// 사용자 생성
app.post('/users', (req, res) => {
  const user = {
    id: req.body.id,
    name: req.body.name
  };
  db.query(`insert into users(id, name) values(${user.id}, '${user.name}')`, (err, result) => {
    if (err){
      console.log(err);
    }
    console.log(result);
    res.status(200).json(result);
  });
});
```


MongoDB

- 대표적인 NoSQL 데이터베이스
- NoSQL은 SQL은 못 쓴다는 게 아니라, SQL 외의 방법으로도 쓸 수 있다는 뜻 (Not only SQL)
- ACID*를 포기하면서 얻은 성능
- ACID : 데이터베이스 트랜잭션이 안전하게 수행되는 것을 보장하는 성질
 - 원자성(Atomicity)
 - 정합성(Consistency)
 - 독립성(Isolation)
 - 지속성(Durability)
- 각 데이터베이스 시스템들은 강점과 약점이 서로 다르므로, 프로젝트에 맞는 DB를 사용해야함



- complete 옵션으로 설치 후 MongoDB Compass 실행
- MongoDB에서는 Table을 Collection으로 표현



• NodeJS <-> MongoDB 연결 예제

<< 7주차 app.js에서 변경(일부) >>

```
// npm install express mongodb
const express = require('express');
const {MongoClient} = require('mongodb');
// Express 설정
const app = express();
const port = 3000;
// MongoDB 데이터베이스 연결 설정
const uri = "mongodb://localhost:27017";
const client = new MongoClient(uri, { useNewUrlParser: true, useUnifiedTopology: true });

async function connectToMongoDB() {
  try {
    await client.connect();
    console.log('Connected to MongoDB');
    return client.db('users'); // 데이터베이스 이름 설정
  } catch (err) {
    console.error(err);
    process.exit(1); // 연결 실패 시 서버 종료
  }
}

// MongoDB 데이터베이스 연결
const db = connectToMongoDB();
// JSON 파싱 미들웨어
app.use(express.json());
```

```
// 모든 사용자 정보 조회
app.get('/users', async (req, res) => {
  try {
    const database = await db;
    const users = database.collection('user');
    const query = {};
    const results = await users.find(query).toArray();
    res.status(200).json(results);
  } catch (err) {
    console.error(err);
    res.status(500).json({ error: "Database query error" });
  }
});

// 사용자 생성
app.post('/users', async (req, res) => {
  const database = await db;
  const users = database.collection('user');

  try {
    const result = await users.insertOne(req.body);
    res.status(201).json(result);
  } catch (err) {
    console.error(err);
    res.status(500).json({ error: "Error inserting data" });
  }
});
```

- 소프트웨어 아키텍처 패턴
 - 소프트웨어 설계에서 공통적인 문제를 해결하기 위해 재사용 가능한 솔루션을 제공
 - 소프트웨어의 기본 구조를 구성하며 컴포넌트의 배치, 각 컴포넌트의 역할 그리고 각 컴포넌트 간의 상호작용 방식을 정의함
- MVC (Model-View-Controller) : 데이터, 사용자 인터페이스, 비즈니스 로직을 구분하여 각각의 개발과 유지보수를 용이하게 하며 코드의 재사용성과 확장성을 향상시킴

구분	역할	기능
Model	<ul style="list-style-type: none">• 모델은 애플리케이션의 데이터와 비즈니스 로직을 캡슐화• 애플리케이션에서 사용되는 데이터의 구조를 정의하고, 데이터를 처리하는 로직(예: 계산, 조건 검사)을 포함	<ul style="list-style-type: none">• 모델은 데이터 변경 사항을 컨트롤러에게 통지하고, 필요에 따라 데이터를 데이터베이스나 다른 저장소와 동기화함
View	<ul style="list-style-type: none">• 뷰는 사용자에게 보이는 인터페이스를 담당• 사용자가 볼 수 있는 모든 UI 요소(예: 버튼, 입력 상자, 레이아웃)를 렌더링하고, 사용자의 입력을 받음	<ul style="list-style-type: none">• 뷰는 모델로부터 데이터를 받아 사용자에게 표시하며, 사용자의 입력을 컨트롤러로 전달• 뷰는 보통 템플릿과 데이터를 결합하여 최종 사용자 인터페이스를 생성
Controller	<ul style="list-style-type: none">• 컨트롤러는 사용자의 입력을 처리하고, 그 입력에 따라 모델과 뷰를 업데이트• 컨트롤러는 사용자의 액션을 해석하고, 그에 따라 모델을 조작하거나 뷰를 변경할 수 있는 명령을 내림	<ul style="list-style-type: none">• 컨트롤러는 사용자의 요청을 받아 처리 결과에 따라 모델을 업데이트하거나, 특정 뷰를 선택하여 응답으로 제공

- Express의 유틸리티 도구
- Express 프로젝트를 시작할 때, 템플릿을 제공해주며 개발자가 직접 수정하여 사용할 수 있음

```
$ npm install -g express-generator

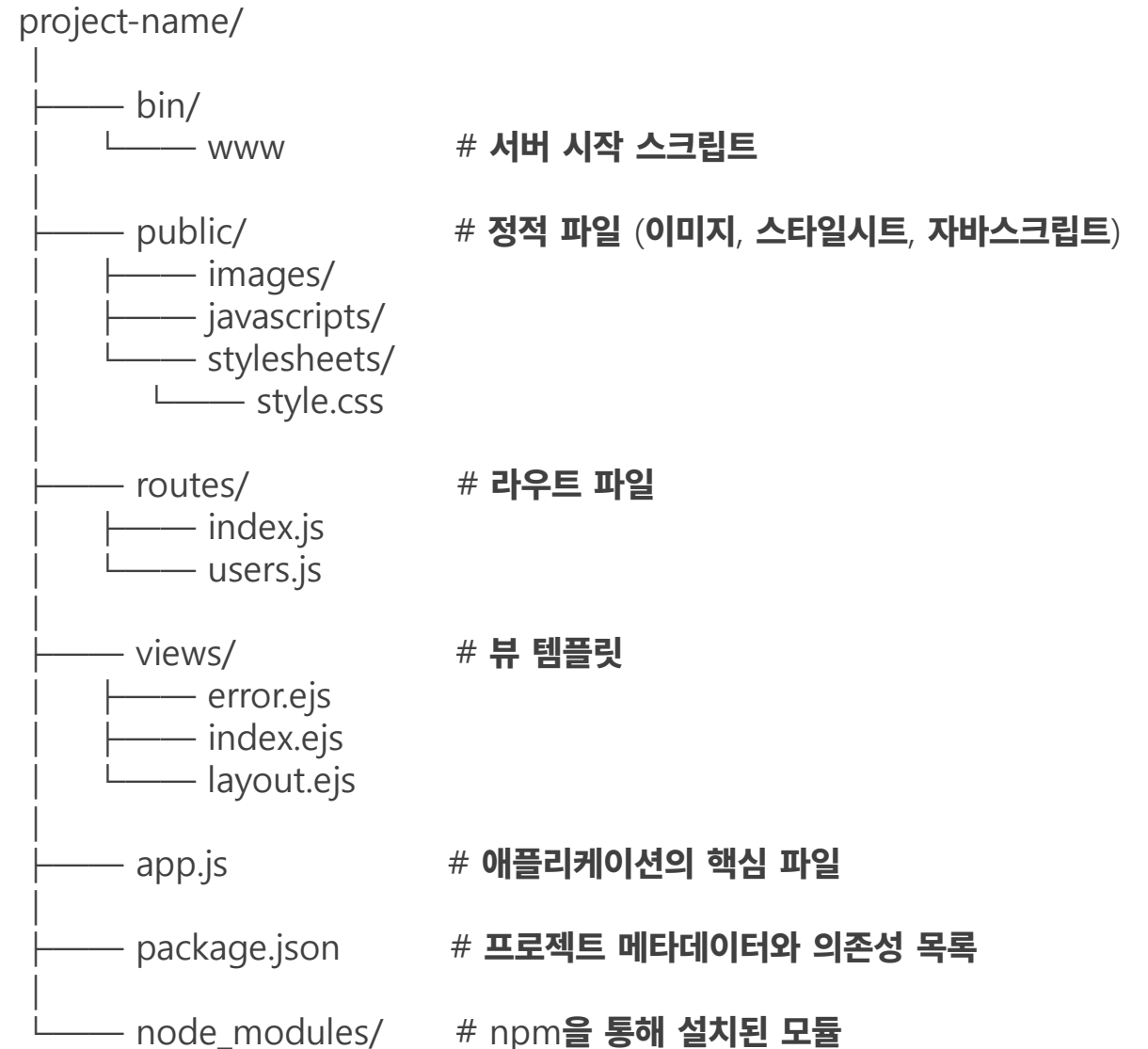
// 프로젝트를 만들 경로로 이동해서
$ express --view=ejs projectName

// 프로젝트 생성 후 이동해서 npm 설치
$ cd projectName
$ npm install

// 프로젝트 시작 시
$ npm start
```

express-generator

- **generator**로 생성 후 다음과 같은 디렉터리 구조가 생성됨
(※이 구조가 **MVC** 구조는 아님)
- **bin** : 바이너리, 주로 실행파일을 의미
- **public** : 클라이언트 측에서 실행되는 정적 리소스 파일들
- **routes** : 핵심 로직들 (컨트롤러 역할)
- **views** : 사용자에게 보여지는 부분 (뷰 역할)



.gitignore

- git에서 무시하는 파일 목록들
- node_modules와 같이 npm으로 설치한 파일들은 git에서 공유하면 안 됨
- package.json에는 npm으로 설치한 목록들이 있어 이것만 공유함
(그래서 git clone을 한 후, npm install을 한번 해줌)
- <https://github.com/github/gitignore> 위 레포에서 여러가지 개발 언어에 대한 ignore를 참고할 수 있음
- / 아래에 .gitignore 파일을 두면 됨

```
project-name/
├── bin/
│   └── www
│
├── public/
│   ├── images/
│   ├── javascripts/
│   ├── stylesheets/
│   └── style.css
│
├── routes/
│   ├── index.js
│   └── users.js
│
├── views/
│   ├── error.ejs
│   ├── index.ejs
│   └── layout.ejs
│
├── app.js
├── package.json
└── node_modules/
```

서버 시작 스크립트

정적 파일 (이미지, 스타일시트, 자바스크립트)

라우트 파일

뷰 템플릿

애플리케이션의 핵심 파일

프로젝트 메타데이터와 의존성 목록

npm을 통해 설치된 모듈

- express에서 사용할 수 있는 뷰 템플릿 엔진
- 서버 사이드에서 템플릿을 사용해 뷰를 만들고 사용자에게 전송하는 방법
- 일반적인 html과 동일하며, 동적인 부분은 <%= %>로 감싸서 처리함
- <%= %>에서 사용하는 변수는, render() 에서 직접 호출하여 보내줌

```
var express = require('express');
var router = express.Router();

/* GET home page. */
router.get('/', function(req, res, next) {
  res.render('index', { title: 'Express' });
});

module.exports = router;
```

index.js

```
<!DOCTYPE html>
<html>
  <head>
    <title><%= title %></title>
    <link rel='stylesheet' href='/stylesheets/style.css' />
  </head>
  <body>
    <h1><%= title %></h1>
    <p>Welcome to <%= title %></p>
  </body>
</html>
```

index.ejs

- 지금까지 배운 것들을 종합하여 간단한 회원가입/로그인 사이트를 만들어보세요.
 - express-generator
 - REST API
 - db 등...