데이터베이스설계 (ICE4016)

실습 6주차

MySQL Express 연동

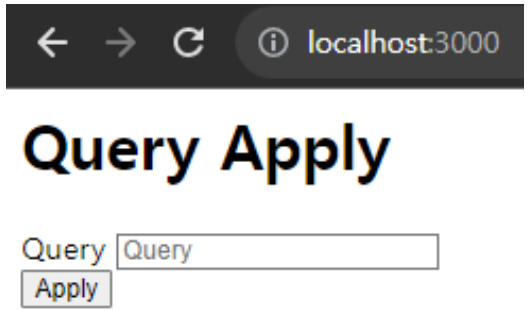
Prof. Wonik Choi

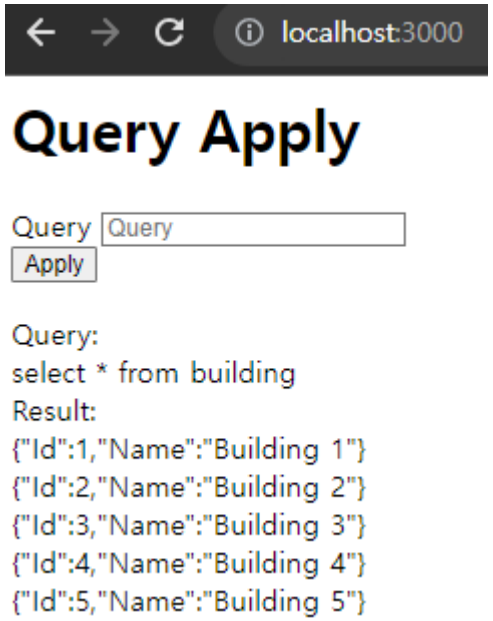
데이터 인텔리전스 연구실
Data Intelligence Laboratory

인하대학교
INHA UNIVERSITY

Week 6 practice goal

- 웹에서 쿼리를 보내고, 결과값을 출력





Week 6 practice

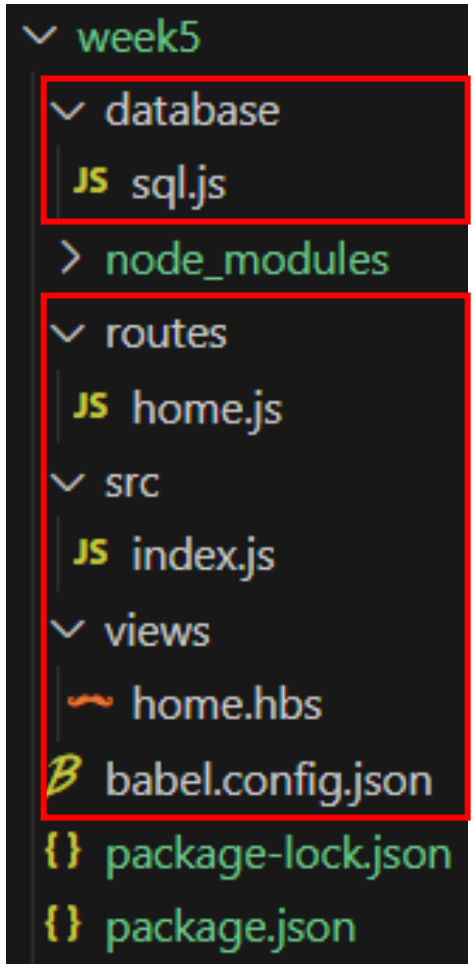
- 프로젝트 생성 및 환경 세팅
  - 환경 세팅을 위한 모듈 설치
    - npm init
    - npm install express mysql2 body-parser nodemon morgan dotenv
    - npm install @babel/node @babel/core @babel/preset-env
    - npm link hbs

```
PS C:\Users\leesw\OneDrive\바탕 화면\db\Database\week5_test> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (week5_test)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to C:\Users\leesw\OneDrive\바탕 화면\db\Database\week5_test\package.json:

{
  "name": "week5_test",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}


Is this OK? (yes) yes
```

```
PS C:\Users\leesw\OneDrive\바탕 화면\db\Database\week5_test> npm install express mysql2 body-parser nodemon morgan dotenv

added 110 packages, and audited 111 packages in 4s

12 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\leesw\OneDrive\바탕 화면\db\Database\week5_test> npm install @babel/node @babel/core @babel/preset-env

added 247 packages, and audited 358 packages in 25s

64 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\leesw\OneDrive\바탕 화면\db\Database\week5_test> npm link hbs

added 1 package, and audited 360 packages in 1s

64 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Week 6 practice

○ 프로젝트 생성 및 환경 세팅



Database > week5 > babel.config.json > ...

```
{
    "presets": ["@babel/preset-env"]
}
```

Database > week5 > package.json > ...

```
{
  "name": "week5",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  ▷ Debug
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "nodemon --exec babel-node ./src/index.js"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "@babel/core": "^7.22.20",
    "@babel/node": "^7.22.19",
    "@babel/preset-env": "^7.22.20",
    "body-parser": "^1.20.2",
    "dotenv": "^16.3.1",
    "express": "^4.18.2",
    "morgan": "^1.10.0",
    "mysql2": "^3.6.1",
    "nodemon": "^3.0.1"
  }
}
```

src/index.js

- router
  - 웹 애플리케이션에서 URL 경로에 따라 요청(requests), 응답(responses)을 처리
  - 웹 애플리케이션에서 여러 URL 경로에 대해 다른 동작을 정의하고자 할 때 router를 사용
  - URL 경로와 그에 대응하는 처리 로직을 매핑하며, 요청이 들어올 때 해당 경로에 대응하는 핸들러 함수를 실행

Database > week5 > src > JS index.js > ...

```
1   import express from 'express';
2   import logger from 'morgan';
3   import path from 'path';
4
5   import homeRouter from '../routes/home';
6
7   const PORT = 3000;
8
9   const app = express(); // Create an 'app' object using the 'express' class
10
11  app.use(express.static(path.join(__dirname, '/src')));
12  // Serve static files from the '/src' directory
13  app.use(express.urlencoded({ extended: false }))
14  // Use URL encoding with 'extended' set to false
15  app.use(express.json()); // Parse incoming data as JSON
16
17  app.set('views', path.join(__dirname, '../views'));
18  // Set the 'views' directory for template rendering
19  app.set('view engine', 'hbs'); // Use the 'hbs' view engine
20
21  app.use(logger('dev'));
22  // Use the 'dev' logger for logging HTTP requests
23
24  app.use('/', homeRouter);
25  // Use the 'homeRouter' for handling routes at the root path
26
27  app.listen(PORT, () => {
28      console.log(`Server is running at http://localhost:${PORT}`);
29      // Start the server and log the listening URL
30  });
```

routes/home.js

- GET method
  - GET method는 서버에서 리소스(데이터)를 요청할 때 사용
  - 정보를 조회, 요청한 리소스를 읽기 위해 사용함
  - GET 요청은 URL에 데이터를 포함시켜 보내며, 이 데이터는 주로 쿼리 문자열(query string)의 형태로 전달
  - 보안적으로 민감한 데이터를 전송하기에는 적합하지 않음

- POST method
  - POST 메서드는 서버로 데이터를 제출하기 위해 사용
  - 사용자가 양식(form)을 작성하고 제출하면, 양식의 데이터가 서버로 전송됨
  - POST 요청은 HTTP 요청 body에 데이터를 포함시켜 보냄
  - GET과 달리 POST 요청은 브라우저의 캐시에 저장되지 않으며, 데이터를 보내는 것이므로 길이 제한이 없음
  - POST 요청은 보안적으로 민감한 데이터를 안전하게 전송할 수 있음
  - 예를 들어, 로그인 정보와 같은 비밀번호를 전송할 때 POST를 사용

Database > week5 > routes > JS home.js > ...

```
1  import express from 'express';
2  import { ApplyQuery } from '../database/sql';
3
4  const router = express.Router();
5
6  // Differentiating between GET and POST methods
7  router.get('/', (_req, res) => {
8      res.render('home', { data: [] });
9      // Render the 'home' template with an empty data field
10 })
11
12 router.post('/', async (req, res) => {
13     const vars = req.body; // Extract request body variables
14     const data = {
15         Query: vars.Query
16     };
17     console.log('data\n', data.Query); // Log the received query data
18     let all_data = []; // Initialize an array to store query results
19
20     try {
21         const result = await ApplyQuery.applyquery(data.Query);
22         // Execute the query and await the result
23         console.log('result\n', result); // Log the query result
24
25         all_data.push('Query:')
26         all_data.push(data.Query)
27         all_data.push('Result:')
28         for (let i = 0; i < result.length; i++) {
29             all_data.push(JSON.stringify(result[i]));
30         }   // Store each result as a JSON string in the 'all_data' array
31         console.log('all_data\n', all_data); // Log the accumulated data
32     }
33     catch (error) { // Handle query errors
34         console.error('Error:', error.message);
35         all_data.push(`${data.Query} is not a query, or there is an error.`);
36         all_data.push('Please check.');
37     }
38
39     res.render('home', { data: all_data });
40     // Render the 'home' template with the collected data
41 })
42
43 module.exports = router;
```

views/home.hbs

- {{#each}}
  {{this}}
  {{/each}}
  - Array의 요소를 순서대로 처리

Database > week5 > views > home.hbs > ...

```
1   <h1>Query Apply</h1> <!-- Title for the page -->
2
3   <form name="employee" method="post" action="/">
4   <!-- Form for submitting a POST request to the root path -->
5       <div>
6           <label for="Query">Query</label>
7           <!-- Label for the input field -->
8           <input id="Query" name="Query" type="text" required placeholder="Query" />
9           <!-- Input field for entering a query, which is required -->
10      </div>
11      <div>
12          <input type="submit" value="Apply" />
13          <!-- Submit button with the label "Apply" -->
14      </div>
15  </form>
16
17  <div>
18      {{#each data}}  <!-- Loop through the 'data' array and display each item -->
19      {{this}}<br>    <!-- Display the current item followed by a line break -->
20      {{/each}}
21  </div>
```

database/sql.js

- Asynchronous Programming
  - 여러 작업을 동시에 수행하거나, 작업이 완료되기를 기다리지 않고도 다음 작업을 시작할 수 있는 프로그래밍 방식
  - 시간이 오래 걸리는 작업, 네트워크 요청, 파일 읽기/쓰기, 데이터베이스 쿼리 등을 효과적으로 다룰 수 있음

- Promise
  - JavaScript에서 비동기 작업을 다루는 방법 중 하나로, 작업의 상태를 나타내는 객체
  - Promise를 사용하면 비동기 작업을 더 간결하고 가독성 있게 다룰 수 있음

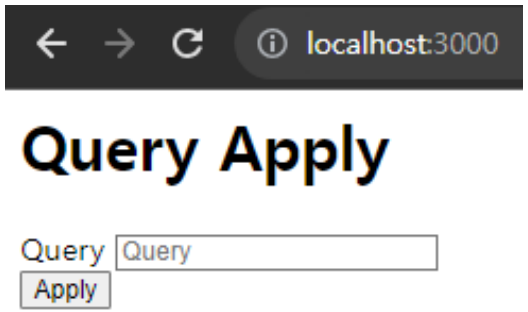
- async/await
  - JavaScript에서 비동기 코드를 작성할 때 사용
  - async 함수 내에서 await를 사용하여 비동기 작업의 완료를 기다릴 수 있고, 코드를 동기식처럼 작성할 수 있음

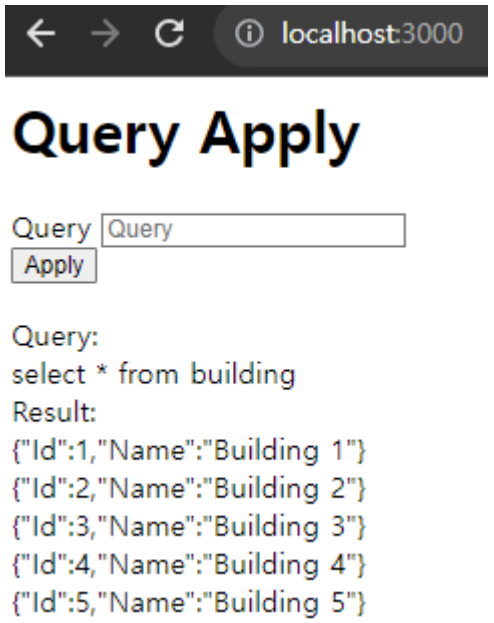
Database > week5 > database > JS sql.js > ...

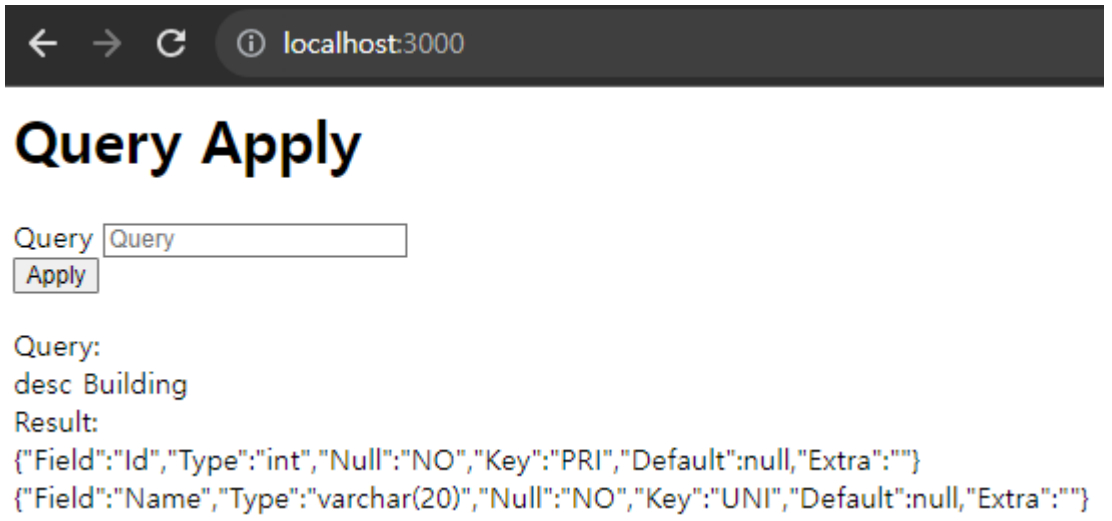
```
1   import mysql from 'mysql2';
2
3   require("dotenv").config();
4
5   // Create a MySQL connection pool with configuration
6   const pool = mysql.createPool({
7       host: 'localhost',          // Hostname of the MySQL server
8       port: 3306,                 // Port to connect to MySQL
9       user: 'root',               // MySQL username
10      password: '',       // MySQL password
11      database: 'inha_week5',     // Name of the database to use
12  });
13
14  // Create a promise-based version of the connection pool
15  const promisePool = pool.promise();
16
17  // Export an object containing a method for running queries
18  export const ApplyQuery = {
19      applyquery: async (Query) => {
20          const sql = Query;  // Store the SQL query provided as a parameter
21          const [result] = await promisePool.query(sql);
22          // Execute the SQL query and await the result
23          return result;  // Return the query result
24      },
25  };
```

Week 6 practice

- Week5 디렉토리에서 'npm run start'
  - http://localhost:3000/ 또는 http://127.0.0.1:3000/
  - Ipconfig(window), ifconfig(linux) 명령어로 ip 확인 가능
  - Ip주소:3000 으로 외부 접속 가능 (ex: 199.199.199.199:3000)







Week 6 practice Assignment

- STEP 1 : 5주차 실습에서 만든 Inha 데이터베이스 확장
  - Inha 데이터베이스에 Class, Club, Employee 테이블 추가
  - 추가된 테이블을 참고하여 기존 DB의 관계(Relation)를 자유롭게 재구성
  - 각 테이블에 5개 이상의 데이터가 있어야 함

Table Name	Attribute	Attribute	Attribute	Attribute	Attribute	Attribute	Attribute
Class	Id	Name	Professor	Number of participants			
Club	Id	Name					
Employee	Id	Name	Position				

- STEP 2 : 확장한 Inha 데이터베이스를 Express와 연동
  - DESC, SELECT 명령어로 확장된 Inha 데이터베이스 확인
- STEP 3 : Express, Inha 데이터베이스 구현 관련 보고서 작성
  - 캡처한 사진들을 포함 할 것
  - 코드 제출 필요 없음