Seoul
Software
ACademy

MVC with MySQL

- 실제 데이터 베이스 연결하기 -

SeSAC 도봉 1기
웹 풀스택 과정

프로젝트 생성 실습

실습. mvc_mysql/ 프로젝트 생성

MVC 패턴으로 프로젝트 생성!!

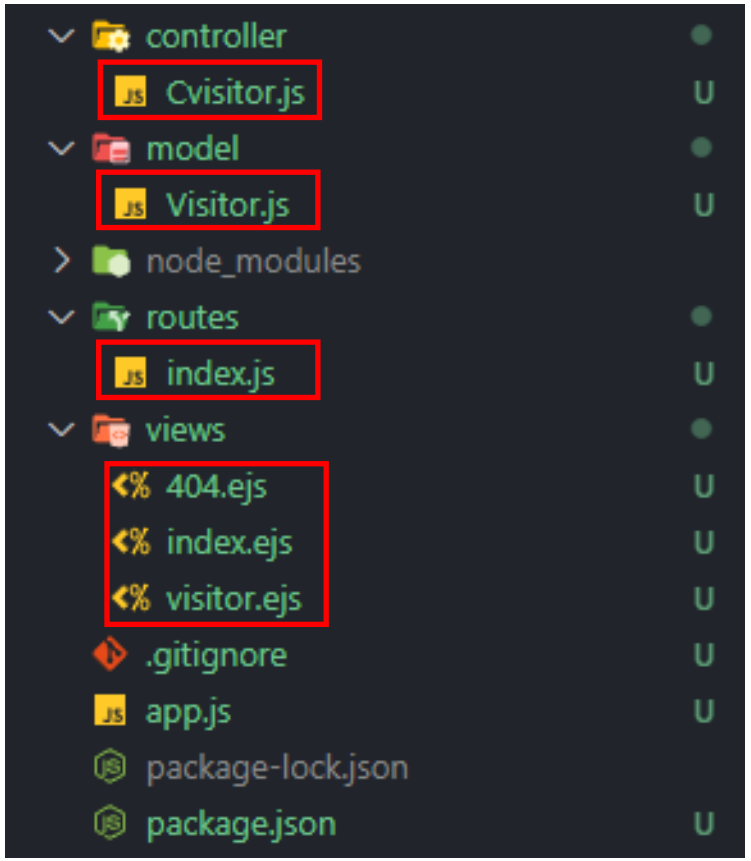
```
mkdir mvc_mysql # 폴더 생성
cd mvc_mysql # 폴더 이동

npm init -y # 프로젝트 시작 명령어 (-y 옵션: package.json 기본 값으로 생성)
# package.json에서 "main" 값을 index.js 에서 app.js로 변경 (진입점 파일명)

npm install express ejs # express와 ejs 패키지 설치

# app.js 파일 생성
# mvc 패턴에 따른 폴더 및 파일 생성
# .gitignore 파일 생성
```

실습. mvc_mysql/ 프로젝트 생성

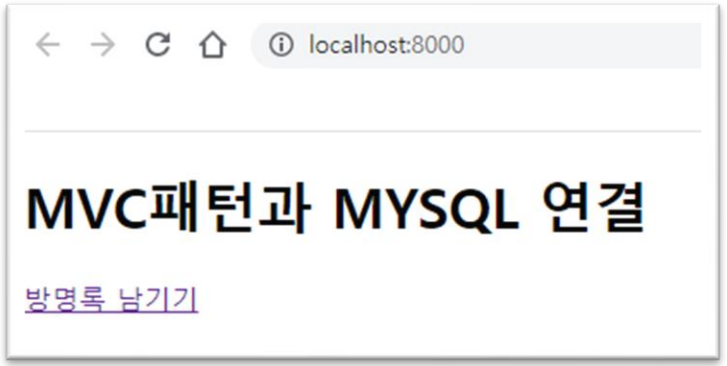


프로젝트 폴더 구조

- view와 model 연결하는 부분
- 데이터 처리하는 부분
- 경로 설정하는 부분
- UI 관련 처리

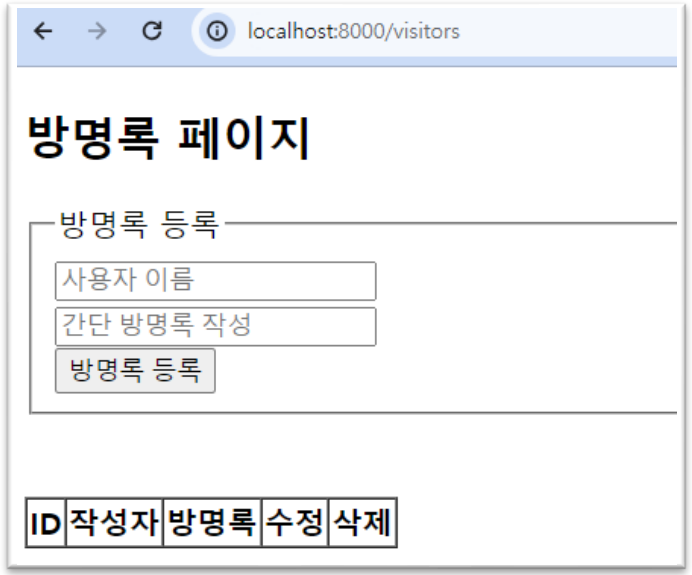
실습. mvc_mysql/ 프로젝트 생성

GET /

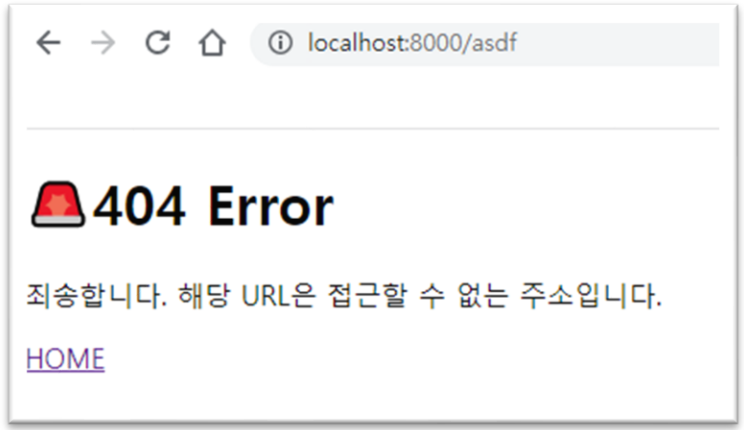


views/**index**.ejs

GET /visitors



views/**visitors**.ejs



views/**404**.ejs

실습. mvc_mysql/ 프로젝트 생성

```
const express = require('express');
const app = express();
const PORT = 8000;

app.set('view engine', 'ejs');
app.use('/views', express.static(__dirname + '/views'));
app.use('/static', express.static(__dirname + '/static'));
app.use(express.urlencoded({ extended: true }));
app.use(express.json());

// [라우터 분리]
const indexRouter = require('./routes'); // index는 생략 가능!
app.use('/', indexRouter); // localhost:PORT/ 경로를 기본으로 ./routes/index.js 파일에 선언한 대로 동작

// [404 error] 맨 마지막 라우트로 선언 -> 나머지 코드 무시되기 때문!!
app.get('*', (req, res) => {
  res.render('404');
});

app.listen(PORT, () => {
  console.log(`http://localhost:${PORT}`);
});
```

app.js

실습. mvc_mysql/ 프로젝트 생성

```
const express = require('express');
const controller = require('../controller/Cvisitor');
const router = express.Router();

// 기본주소: localhost:PORT

// GET / => localhost:PORT/
router.get('/', controller.main);

// GET /visitor => localhost:PORT/visitor
router.get('/visitor', controller.getVisitors);

module.exports = router;
```

routes/index.js

실습. mvc_mysql/ 프로젝트 생성

```
const Visitor = require('../model/Visitor');

// GET / => localhost:PORT/
exports.main = (req, res) => {
  res.render('index');
};

// GET /visitor => localhost:PORT/visitor
exports.getVisitors = (req, res) => {
  console.log(Visitor.getVisitors());
  res.render('visitor', { data: Visitor.getVisitors() });
};
```

controller/Cvisitor.js

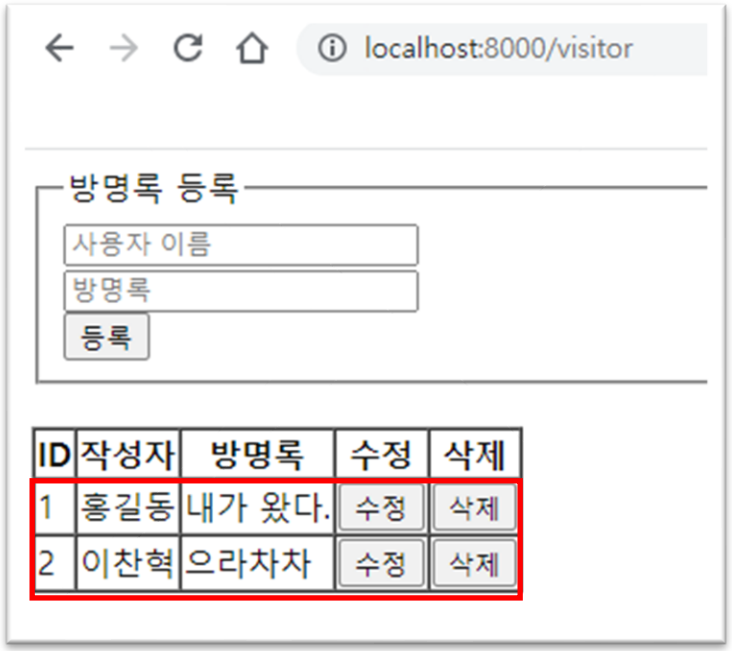
실습. mvc_mysql/ 프로젝트 생성

```
// (임시) DB로부터 방명록 데이터를 받아옴
exports.getVisitors = () => {
  return [
    { id: 1, name: '홍길동', comment: '내가 왔다.' },
    { id: 2, name: '이찬혁', comment: '으라차차' },
  ];
};
```

model/Visitor.js

❌MySQL(DB) 연결 전 상태

실습. mvc_mysql/ 프로젝트 생성




```ejs
<table border="1" cellspacing="0">
  <thead>
    <tr>
      <th>ID</th>
      <th>작성자</th>
      <th>방명록</th>
      <th>수정</th>
      <th>삭제</th>
    </tr>
  </thead>
  <tbody>
    <!-- data: db에서 가지고 오는 데이터 => 새로고침해도 데이터 남아있음 -->
    <% for (let i = 0; i < data.length; i++) { %>
    <tr id="tr_<%= data[i].id %>">
      <td><%= data[i].id %></td>
      <td><%= data[i].name %></td>
      <td><%= data[i].comment %></td>
      <td><button type="button">수정</button></td>
      <td><button type="button">삭제</button></td>
    </tr>
    <% } %>
  </tbody>
</table>
```

views/**visitor**.ejs (일부)

데이터베이스
테이블 생성

실습. visitor 테이블 생성하기

```
mysql> DESC visitor;
+---------+-------------+------+-----+---------+----------------+
| Field   | Type        | Null | Key | Default | Extra          |
+---------+-------------+------+-----+---------+----------------+
| id      | int         | NO   | PRI | NULL    | auto_increment |
| name    | varchar(10) | NO   |     | NULL    |                |
| comment | mediumtext  | YES  |     | NULL    |                |
+---------+-------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM visitor;
+----+--------+------------+
| id | name   | comment    |
+----+--------+------------+
|  1 | 홍길동 | 내가 왔다. |
|  2 | 이찬혁 | 으라차차   |
+----+--------+------------+
2 rows in set (0.00 sec)
```

```
// (임시) DB로부터 방명록 데이터를 받아옴
exports.getVisitors = () => {
  return [
    { id: 1, name: '홍길동', comment: '내가 왔다.' },
    { id: 2, name: '이찬혁', comment: '으라차차' },
  ];
};
```

model/Visitor.js 에서 임시로 작성한 데이터를

실제 visitor 테이블에 저장해보자!!

프로젝트에 MySQL 연결하기

Todo List

1. 작성 후 “**등록**” 을 누르면 **DB 에 저장**된다. (Create)
2. “**등록**” 된 내용이 아래 Table 에 **바로 보인다**. (Read)
3. “**수정**” 을 누르면 **방명록 수정**이 가능하다. (Update)
4. “**삭제**” 를 누르면 **방명록이 삭제**된다. (Delete)

Node.js – MySQL 연결

- mysql 패키지를 설치

```
$ npm install mysql
```

- package.json 에서 설치 확인

```
"dependencies": {
  "ejs": "^3.1.8",
  "express": "^4.18.2",
  "mysql": "^2.18.1"
}
```

model/Visitor.js

- mysql 패키지를 설치 후 이용

```
const mysql = require('mysql');

const conn = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '1234',
  database: 'kdt',
});
```

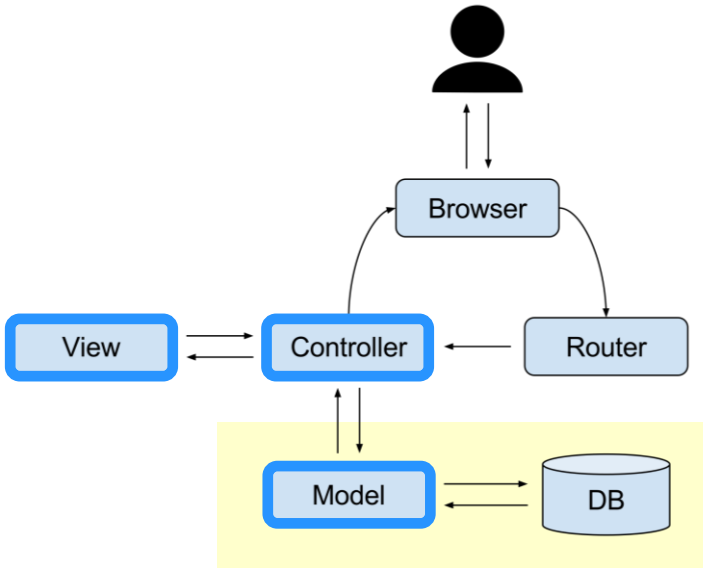
DB가 설치된 호스트 IP 주소

DB 접속 유저이름

DB 접속 비밀번호

DB 이름

model/Visitor.js



model/Visitor.js

```
exports.getVisitors = (cb) => {
  conn.query(`SELECT * FROM visitor`, (err, rows) => {
    if (err) {
      throw err;
    }

    console.log('Visitor.js: ', rows);
    cb(rows);
  });
};
```

model/Visitor.js – **getVisitors()**: 전체 방명록 목록을 가져옴

Node.js mysql 연결

```
Error: ER_NOT_SUPPORTED_AUTH_MODE: Client does not support authentication protocol requested by server; consider upgrading MySQL client
    at Handshake.Sequence._packetToError (C:\Users\Lily\github\KDT\KDT-2nd\node_modules\mysql\lib\protocol\sequences\Sequence.js:47:14)
    at Handshake.ErrorPacket (C:\Users\Lily\github\KDT\KDT-2nd\node_modules\mysql\lib\protocol\sequences\Handshake.js:123:18)
    at Protocol._parsePacket (C:\Users\Lily\github\KDT\KDT-2nd\node_modules\mysql\lib\protocol\Protocol.js:291:23)
```

- 외부에서 최상위 **root 계정으로의 비밀번호 접근을 허용하지 않는다!**
- 즉, **새로운 사용자(user 계정)**를 만들고 그 사용자로 접근을 해야 한다.

```
const mysql = require('mysql');

const conn = mysql.createConnection({
  host: 'localhost',
  user: 'user',
  password: '1234',
  database: 'kdt',
});
```

model/Visitor.js

Node.js mysql 연결

```sql
-- MySQL 사용자 추가하기
CREATE USER 'user'@'%' IDENTIFIED BY '비밀번호';
-- user 계정에 DB 권한 부여 (모든 DB에 접근 가능하도록)
GRANT ALL PRIVILEGES ON *.* TO 'user'@'%' WITH GRANT OPTION;
-- 현재 사용중인 MySQL 캐시를 지우고 새로운 설정 적용
FLUSH PRIVILEGES;

-- MySQL 비번 변경하고 싶다면?
ALTER USER 'user'@'%' IDENTIFIED WITH mysql_native_password BY '비밀번호';
```

model/Visitor.js

```js
const mysql = require('mysql');

const conn = mysql.createConnection({
  host: 'localhost',
  user: 'user',
  password: '1234',
  database: 'kdt',
});
```

Node.js mysql 연결

```
exports.getVisitors = (cb) => {
  conn.query(`SELECT * FROM visitor`, (err, rows) => {
    if (err) {
      throw err;
    }

    console.log('Visitor.js: ', rows);
    cb(rows);
  });
};
```

model/Visitor.js

```
Visitor.js:  [
  RowDataPacket { id: 1, name: '홍길동', comment: '내가 왔다.' },
  RowDataPacket { id: 2, name: '이찬혁', comment: '으라차차' }
]
```

model/Visitor.js

```
exports.getVisitors = (cb) => {
  conn.query(`SELECT * FROM visitor`, (err, rows) => {
    if (err) {
      throw err;
    }

    console.log('Visitor.js: ', rows);
    cb(rows);
  });
};
```

model/Visitor.js

controller/Cvisitor.js

```
exports.getVisitors = (cb) => {
  conn.query(`SELECT * FROM visitor`, (err, rows) => {
    if (err) {
      throw err;
    }

    console.log('Visitor.js: ', rows);
    cb(rows);
  });
};
```

model/Visitor.js

controller/Cvisitor.js

```
exports.getVisitors = (req, res) => {
  Visitor.getVisitors((result) => {
    console.log('Cvisitor.js: ', result);
    res.render('visitor', { data: result });
  });
};
```

라우터 정리

1. 작성 후 “**등록**” 을 누르면 **DB 에 저장**된다. (Create)
2. “**등록**” 된 내용이 아래 Table 에 **바로 보인다**. (Read)
3. “**수정**” 을 누르면 **방명록 수정**이 가능하다. (Update)
4. “**삭제**” 를 누르면 **방명록이 삭제**된다. (Delete)

GET `/` : 메인 페이지 보이기 (index.ejs)

GET `/visitors` : 방명록 전체 보이기 (visitor.ejs)

GET `/visitor/:id` : 방명록 하나 조회

POST `/visitor` : 방명록 하나 추가

PATCH `/visitor` : 방명록 하나 수정

DELETE `/visitor` : 방명록 하나 삭제

실습. Node.js – MySQL

- **회원가입, 로그인, 회원정보 수정, 회원 탈퇴**를 **DB와 연동**하여 구현
- 이 때, CRUD 모두 사용
  - CREATE (INSERT) : 회원 가입
  - READ (SELECT) : 로그인 (회원 검색)
  - UPDATE (UPDATE) : 회원 정보 수정
  - DELETE (DELETE) : 회원 탈퇴 (삭제)
- (기능 구현 완성 시) CSS로 디자인 적용
  - css 파일은 static 폴더에 생성하기
  - reset.css, nomalize.css, bootstrap, google font 등 활용 해보기

실습. Node.js – MySQL

- user 테이블 구조

```
mysql> DESC user;
+--------+-------------+------+-----+---------+----------------+
| Field  | Type        | Null | Key | Default | Extra          |
+--------+-------------+------+-----+---------+----------------+
| id     | int         | NO   | PRI | NULL    | auto_increment |
| userid | varchar(20) | NO   |     | NULL    |                |
| name   | varchar(10) | NO   |     | NULL    |                |
| pw     | varchar(20) | NO   |     | NULL    |                |
+--------+-------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```