

# Web Tutorial

2024년 2학기	
제출일	2024. 09.30.
과목명	데이터베이스및데이터시각화
담당교수	이기훈

학과	컴퓨터정보공학부
학번	2020202031
이름	김재현

KWANGWOON UNIVERSITY

## 1. 튜토리얼 모든 과정 설명 및 캡처

Discover TypeScript in Node.js →

# Run JavaScript Everywhere

Node.js® is a free, open-source, cross-platform JavaScript runtime environment that lets developers create servers, web apps, command line tools and scripts.

**Download Node.js (LTS)** 

Downloads Node.js **v20.17.0**<sup>1</sup> with long-term support.  
Node.js can also be installed via [package managers](#).

Want new features sooner? Get **Node.js v22.9.0**<sup>1</sup> instead.

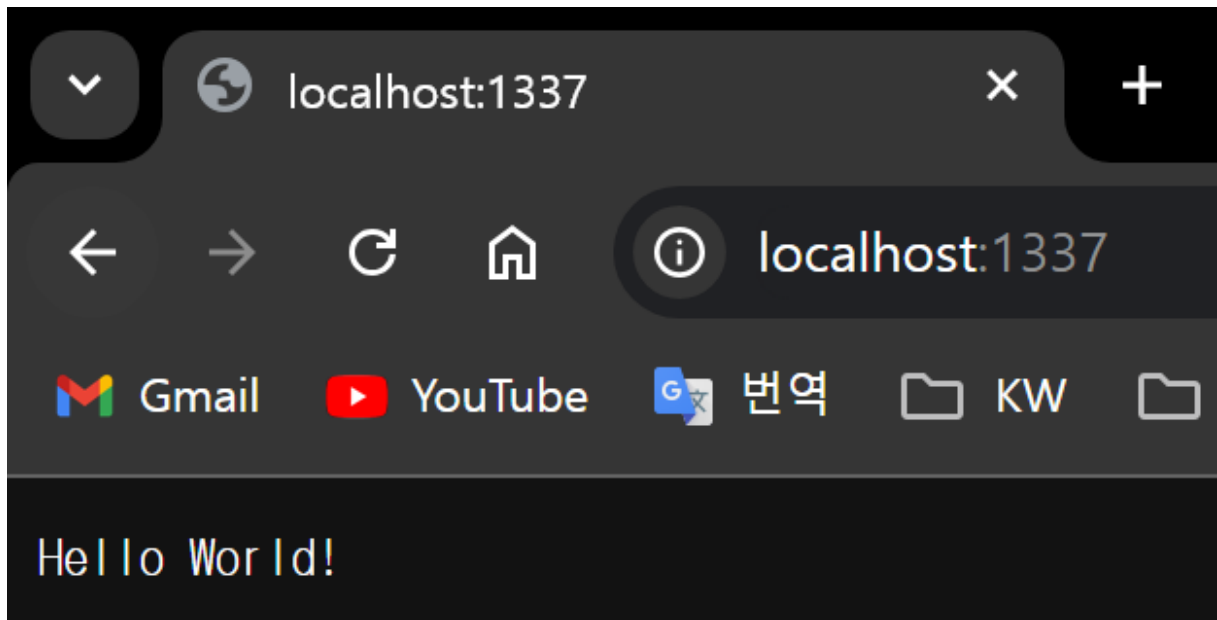
node.js를 다운로드 후 설치한다.

```
C: > Users > kk200 > helloNode > JS hello.js > ...  
1  var http = require('http');  
2  http.createServer(function (req, res){  
3    res.writeHead(200, {'Content-Type': 'text/plain'});  
4    res.end('Hello World!\n');  
5  }).listen(1337, '127.0.0.1');  
6  console.log('Server running at http://127.0.0.1:1337/');
```

helloNode 폴더를 만들고 hello.js파일을 위와 같이 입력한다.

```
C:\Users\kk200\helloNode>node hello.js  
Server running at http://127.0.0.1:1337/  
|
```

서버를 실행한다.



잘 실행됨을 확인할 수 있다.

```
C:\Users\kk200\myNode>npm install -g express-generator
```

express를 설치한다.

```
C:\Users\kk200\myNode>express example1
```

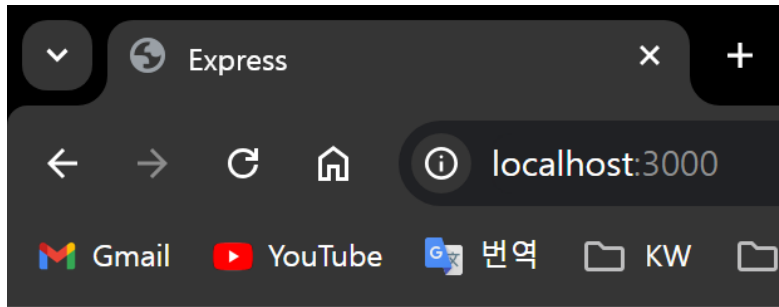
```
C:\Users\kk200\myNode>cd example1
```

```
C:\Users\kk200\myNode\example1>npm install
```

install dependency

```
C:\Users\kk200\myNode\example1>npm start
```

서버 실행



# Express

Welcome to Express

잘 실행됨을 확인할 수 있다.

```
C:\Users\kk200\myNode>express --ejs joinForm
```

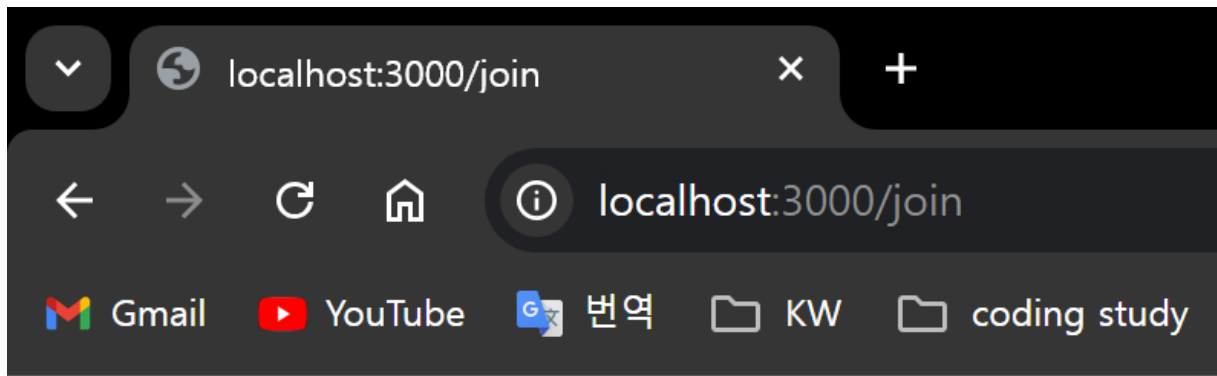
프로젝트 생성

```
C:\Users\kk200\myNode>cd joinForm  
C:\Users\kk200\myNode\joinForm>npm install
```

모듈설치

```
C:\Users\kk200\myNode\joinForm>npm install --save multer
```

multer 설치

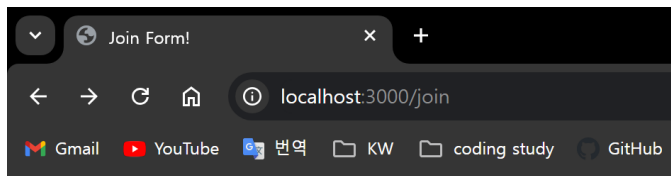


## joinForm.js check

잘 실행 됨을 확인할 수 있다.



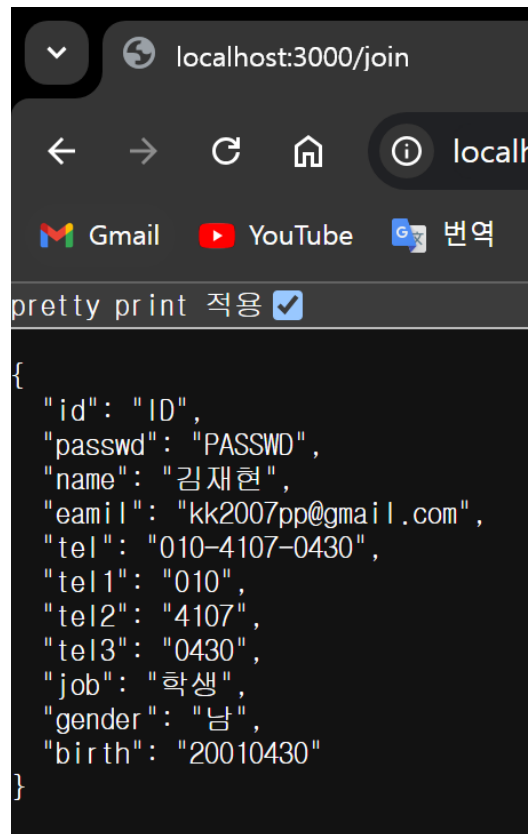
jQuery 다운로드



## Join Form!

아이디	<input type="text" value="ID"/>
비밀번호	<input type="password" value="....."/>
이름	<input type="text" value="김재현"/>
이메일	<input type="text" value="kk2007pp@gmail.com"/>
전화번호	<input type="text" value="010"/> <input type="text" value="4107"/> <input type="text" value="0430"/>
주소	<input type="text" value="광운대학교"/>
직업	<input type="text" value="학생"/>
성별	<input checked="" type="radio"/> 남 <input type="radio"/> 여
생일	<input type="text" value="20010430"/> *YYYYMMDD
<input type="button" value="가입"/> <input type="button" value="취소"/>	

잘 실행됨을 확인할 수 있다.



```
C:\Users\kk200\myNode\joinForm>npm install -g nodemon  
  
added 29 packages in 2s  
  
4 packages are looking for funding  
run `npm fund` for details
```

소스코드를 수정할 때 마다 서버를 재시작하는 것을 너무 비효율적이다.  
따라서 수정된 소스가 콘솔창에 나타나도록 nodemon을 설치한다.

```
joinForm > {} package.json > {} scripts > start
1  {
2    "name": "joinform",
3    "version": "0.0.0",
4    "private": true,
5    "scripts": {
6      "start": "nodemon ./bin/www"
7    },
8    "dependencies": {
9      "cookie-parser": "~1.4.4",
10     "debug": "~2.6.9",
11     "ejs": "~2.6.1",
12     "express": "~4.16.1",
13     "http-errors": "~1.6.3",
14     "morgan": "~1.9.1",
15     "multer": "^1.4.5-lts.1"
16   }
17 }
18
```


package.json 수정

## MySQL Community Downloads

MySQL Installer

**General Availability (GA) Releases**Archives

### MySQL Installer 8.0.39

 **Note:** MySQL 8.0 is the final series with MySQL Installer. As of MySQL 8.1, use a MySQL product's MSI or Zip archive for installation. MySQL Server 8.1 and higher also bundle MySQL Configurator, a tool that helps configure MySQL Server.

Select Version:  
8.0.39

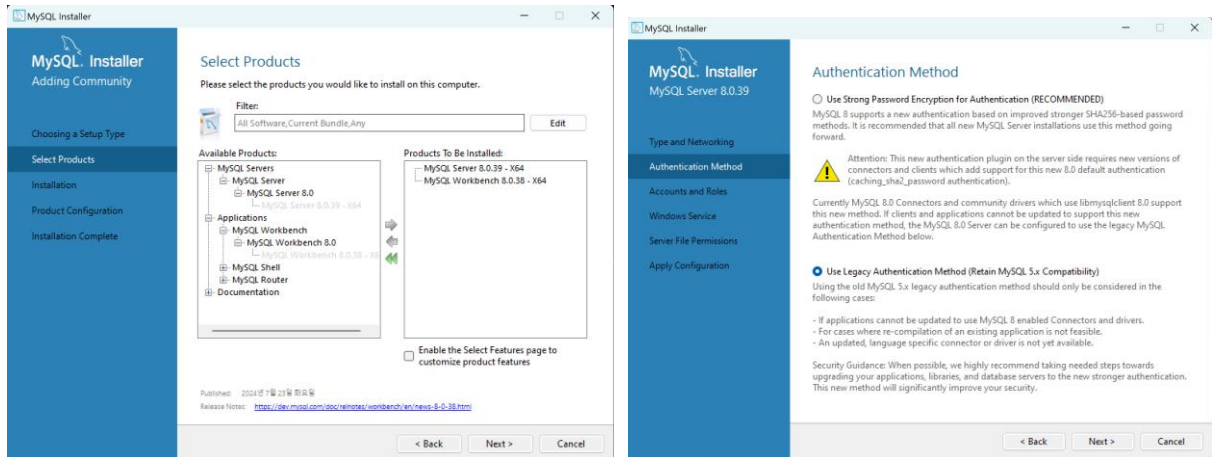
Select Operating System:  
Microsoft Windows

<b>Windows (x86, 32-bit), MSI Installer</b> (mysql-installer-web-community-8.0.39.0.msi)	8.0.39	2.1M	<a href="#">Download</a>
<b>Windows (x86, 32-bit), MSI Installer</b> (mysql-installer-community-8.0.39.0.msi)	8.0.39	303.6M	<a href="#">Download</a>

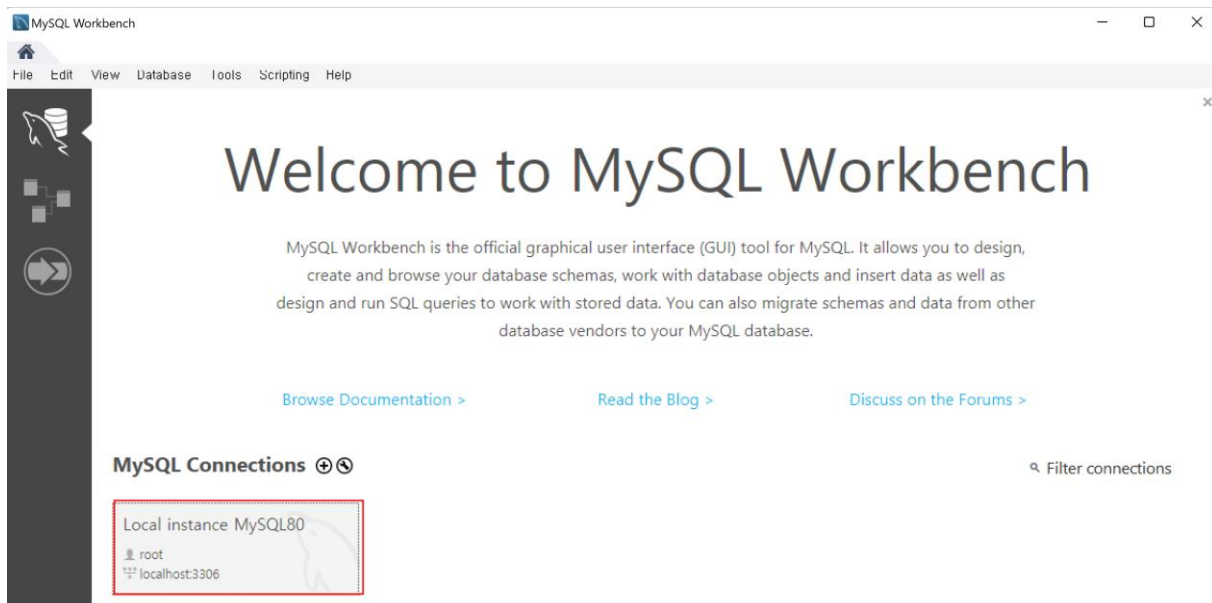
MD5: d8499da0b2c4b5dfa81a5c5185af9238 | [Signature](#)

MD5: 353c5e5ab9350d0e9ddcb42264229b5d | [Signature](#)

MySQL 다운로드

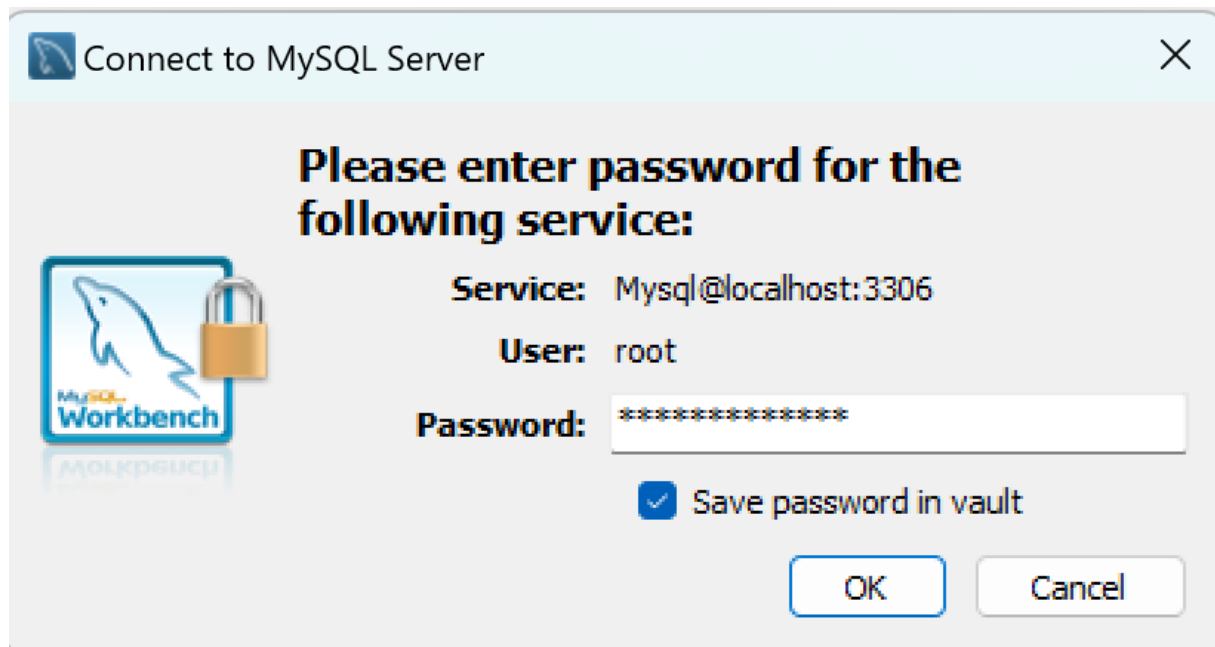


MySQL 설치

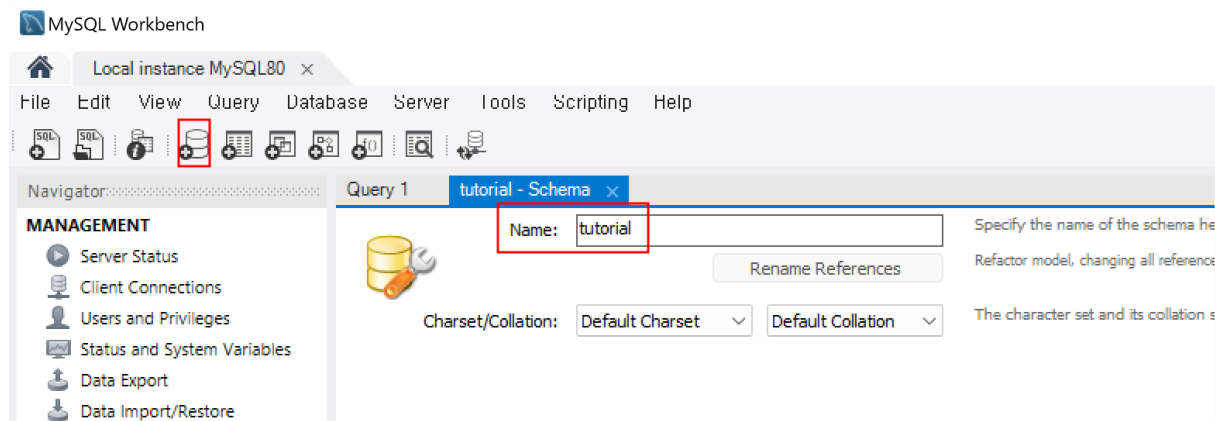


더블클릭

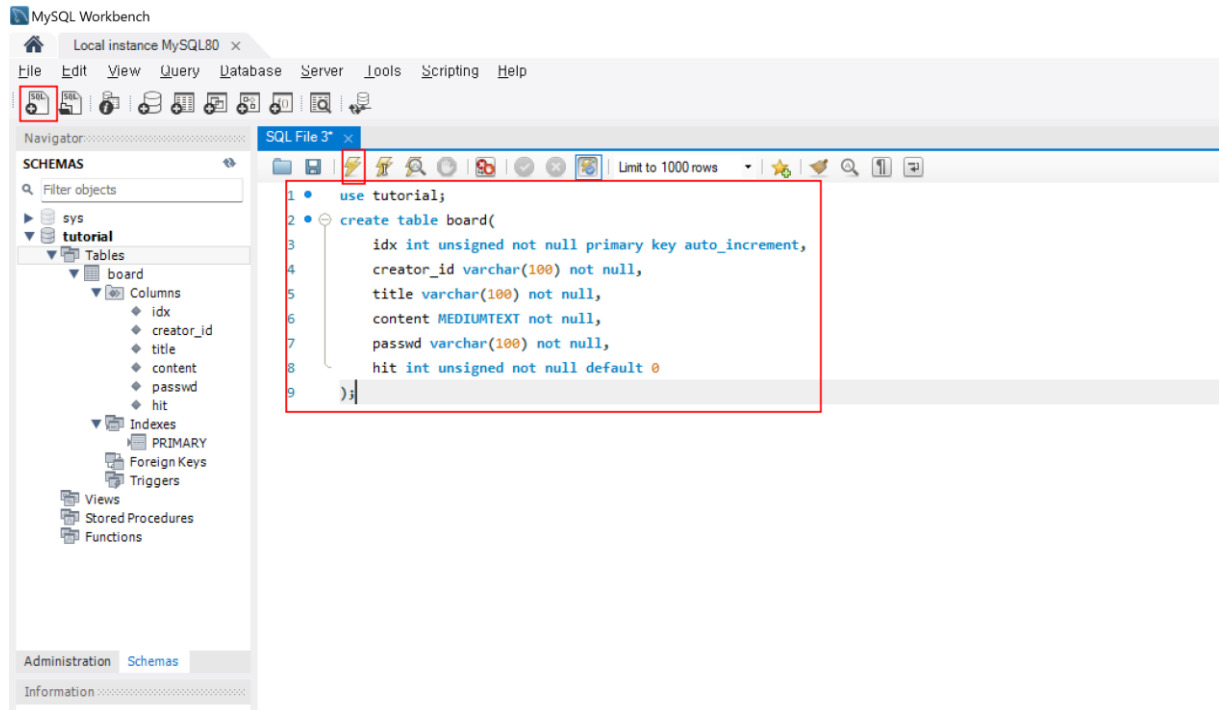




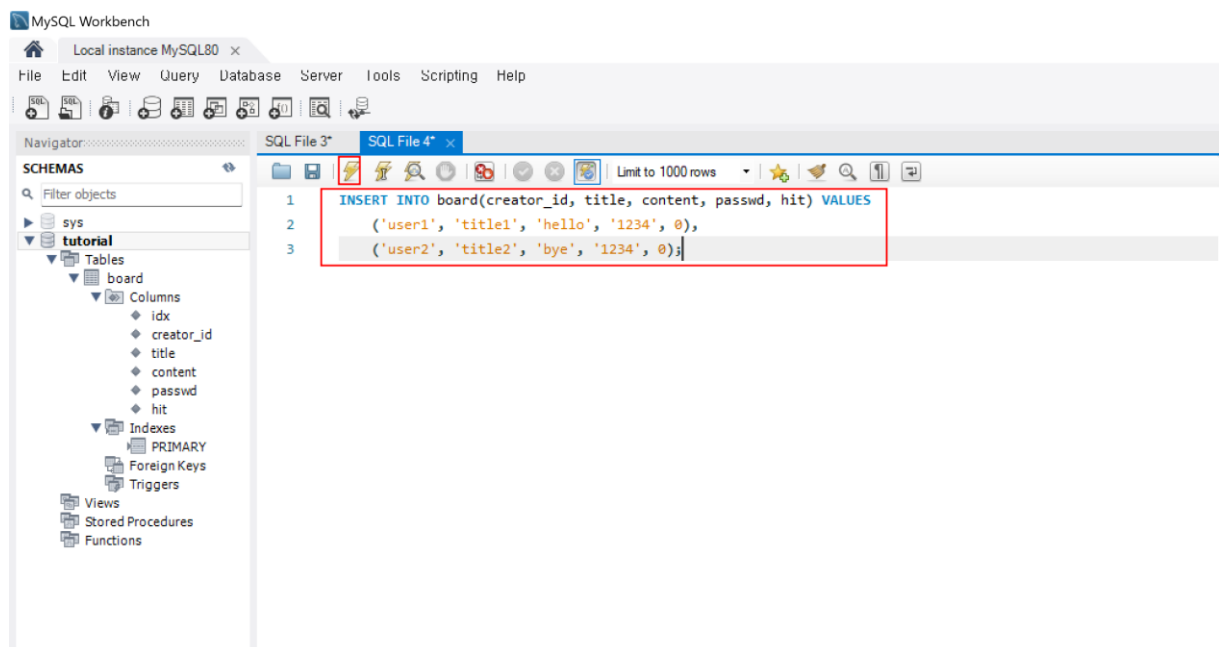
비밀번호 입력



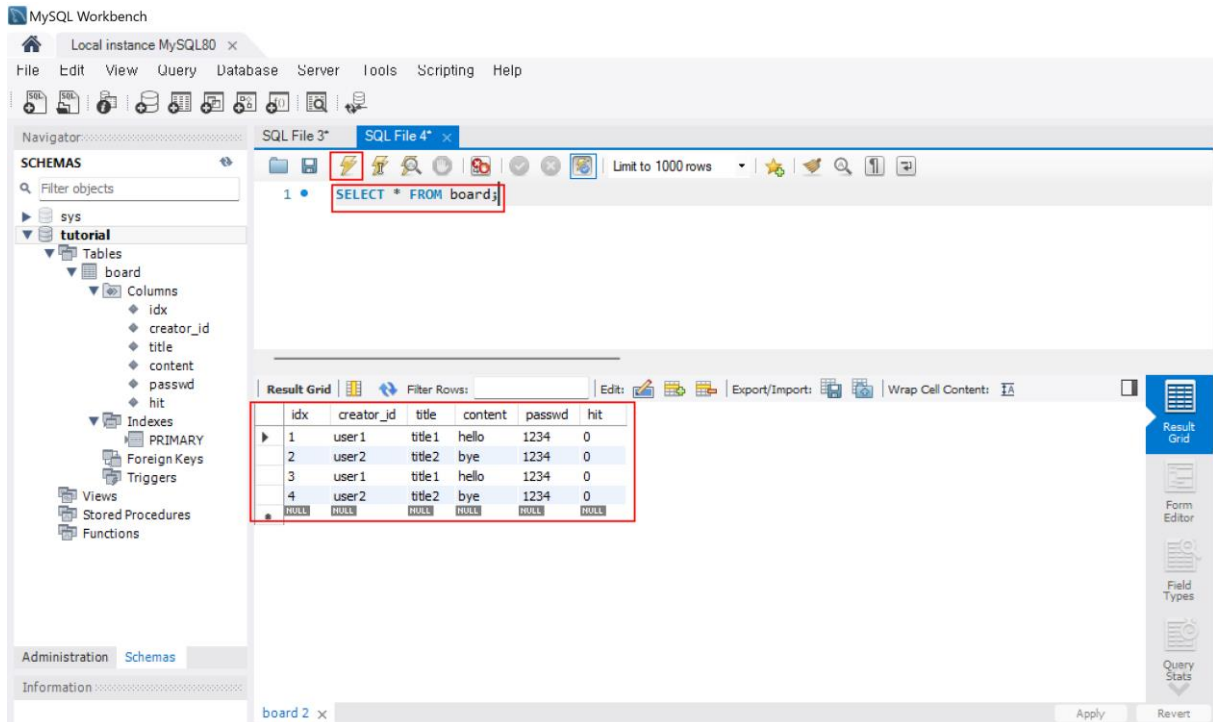
tutorial 이름으로 new schema 생성



new SQL을 create하고 코드를 입력하고 번개 아이콘을 클릭해줍니다.  
생성된 테이블을 확인합니다.



test data를 입력 후 번개 아이콘을 클릭합니다. database에 test data가 삽입됩니다.



다음과 같이 코드를 재입력하고 번개 아이콘을 클릭하면 표가 나타납니다.

이전 단계에서 테스트 데이터 삽입을 두 번 실행하여 총 4개의 데이터가 삽입됐습니다.

```
C:\Users\kk200\myNode\joinForm>npm install mysql
```

```
added 4 packages, and audited 76 packages in 3s
```

```
1 package is looking for funding
  run `npm fund` for details
```

```
7 vulnerabilities (2 moderate, 4 high, 1 critical)
```

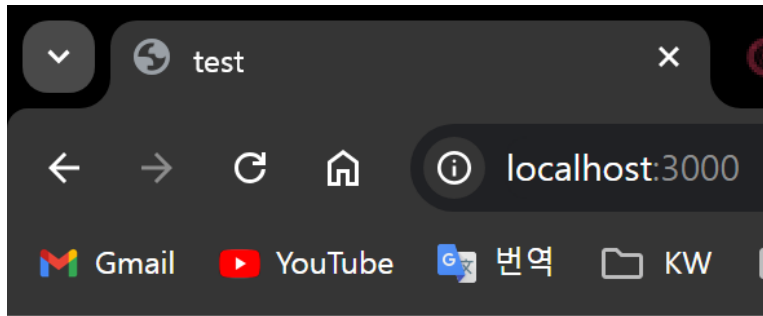
```
To address issues that do not require attention, run:
  npm audit fix
```

```
To address all issues (including breaking changes), run:
  npm audit fix --force
```

```
Run `npm audit` for details.
```

```
C:\Users\kk200\myNode\joinForm>
```

Node.js에 mysql을 설치합니다.



# test

Welcome to test

1

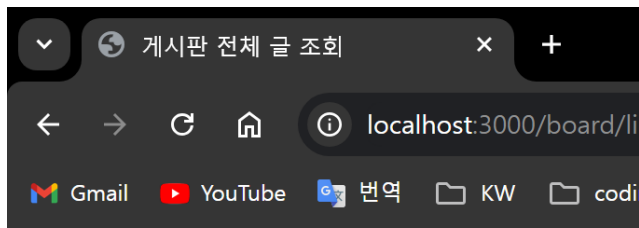
user1

title1

hello

데이터베이스가 잘 연동됐음을 확인

# 게시판 구축 시작



## 게시판 전체 글 조회

[글쓰기로 이동](#)

번호	작성자	제목	조회수
1	user1	title1	0
2	user2	title2	0
3	user1	title1	0
4	user2	title2	0

잘 실행됨을 알 수 있다.

## 게시판 전체 글 조회

[글쓰기로 이동](#)

번호	작성자	제목	조회수
1	user1	title1	0
2	user2	title2	0
3	user1	title1	0
4	user2	title2	0

## 게시판 글 쓰기

작성자	<input type="text" value="김재현"/>
제목	<input type="text" value="글쓰기 테스트"/>
내용	<div>글쓰기 테스트</div>
패스워드	<input type="password" value="...."/>
<input type="button" value="글쓰기"/>	

글쓰기가 제대로 수행됨을 확인

list.ejs파일에 글 조회 링크 추가

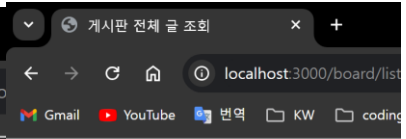
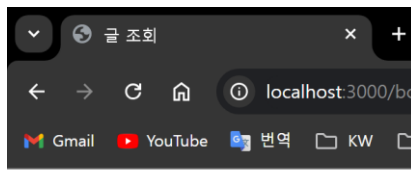
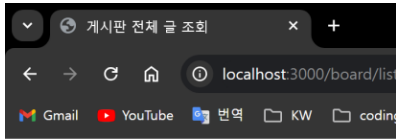
## 게시판 글 쓰기

작성자	<input type="text"/>
제목	<input type="text"/>
내용	<div></div>
패스워드	<input type="password"/>
<input type="button" value="글쓰기"/>	

## 게시판 전체 글 조회

[글쓰기로 이동](#)

번호	작성자	제목	조회수
1	user1	title1	0
2	user2	title2	0
3	user1	title1	0
4	user2	title2	0
5	김재현	글쓰기 테스트	0
6	김재현	글쓰기 테스트	0



## 게시판 전체 글 조회

[글쓰기로 이동](#)

번호	작성자	제목	조회수
1	user1	<a href="#">title1</a>	0
2	user2	<a href="#">title2</a>	0
3	user1	<a href="#">title1</a>	0
4	user2	<a href="#">title2</a>	0
5	김재현	<a href="#">글쓰기 테스트</a>	0
6	김재현	<a href="#">글쓰기 테스트</a>	0

코드가 잘 적용됐는지 확인

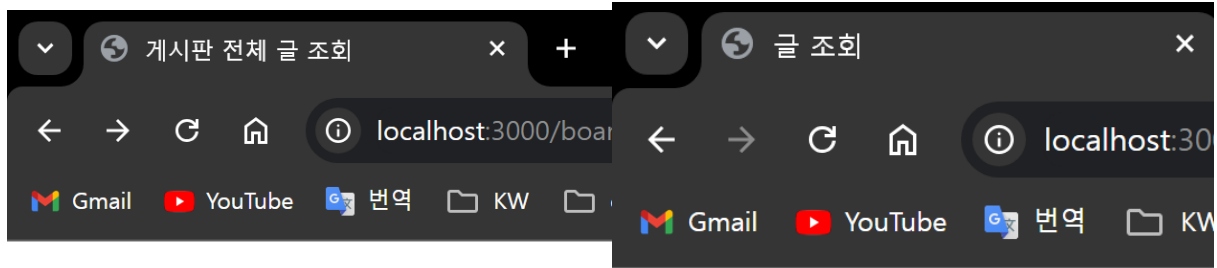
## 글 조회

작성자	user2
제목	title2
내용	bye
조회수	0
글 수정	<a href="#">리스트로 돌아가기</a>

## 게시판 전체 글 조회

[글쓰기로 이동](#)

번호	작성자	제목	조회수
1	user1	<a href="#">title1</a>	0
2	user2	<a href="#">title2</a>	0
3	user1	<a href="#">title1</a>	0
4	user2	<a href="#">title2</a>	0
5	김재현	<a href="#">글쓰기 테스트</a>	0
6	김재현	<a href="#">글쓰기 테스트</a>	0



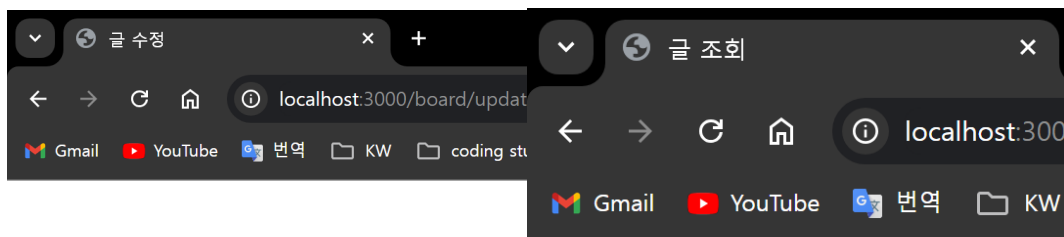
## 게시판 전체 글 조회

[글쓰기로 이동](#)

번호	작성자	제목	조회수
1	user1	<a href="#">title1</a>	0
2	user2	<a href="#">title2</a>	0
3	user1	<a href="#">title1</a>	0
4	user2	<a href="#">title2</a>	0
5	김재현	<a href="#">글쓰기 테스트</a>	0
6	김재현	<a href="#">글쓰기 테스트</a>	0

## 글 조회

작성자	김재현
제목	글쓰기 테스트
내용	글쓰기 테스트
조회수	0
글 수정	<a href="#">리스트로 돌아가기</a>



## 글 수정

작성자	김재현
제목	글쓰기 테스트
내용	수정 수정
패스워드	....
글쓰기	

## 글 조회

작성자	김재현
제목	글쓰기 테스트
내용	수정 수정
조회수	0
글 수정	<a href="#">리스트로 돌아가기</a>

글 수정이 잘 수행됨을 알 수 있습니다.



## MySQL을 cmd창으로 실행해보기

```
명령 프롬프트 - mysql -u root X + v
Microsoft Windows [Version 10.0.22631.4169]
(c) Microsoft Corporation. All rights reserved.

C:\Users\kk200>cd \Program Files\MySQL\MySQL Server 8.0\bin
C:\Program Files\MySQL\MySQL Server 8.0\bin>mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 14
Server version: 8.0.39 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> |
```

cd 명령어를 통해 WProgram FilesWMySQLWMySQL Server 8.0Wbin 폴더로 이동합니다.  
mysql -u root -p 명령어를 통해 데이터베이스에 접근합니다.

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
| tutorial |
+-----+
5 rows in set (0.00 sec)
```

show databases 명령어를 통해 database를 확인합니다.

```
mysql> create database basic;  
Query OK, 1 row affected (0.02 sec)
```

```
mysql> show databases;  
+-----+  
| Database |  
+-----+  
| basic    |  
| information_schema |  
| mysql    |  
| performance_schema |  
| sys      |  
| tutorial  |  
+-----+  
6 rows in set (0.00 sec)
```

basic 이라는 이름의 database를 생성하고 확인합니다.

```
mysql> use basic;  
Database changed
```

basic database에 접속합니다.

```
mysql> create table sample(
    -> item varchar(20),
    -> price int);
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> show tables
    -> ;
+-----+
| Tables_in_basic |
+-----+
| sample          |
+-----+
1 row in set (0.00 sec)
```

```
mysql> |
```

sample이라는 이름의 table을 생성하고 확인합니다.

```
mysql> desc sample;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| item  | varchar(20)   | YES  |     | NULL    |       |
| price | int           | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

```
mysql> |
```

table의 속성을 확인합니다.

```
mysql> insert into sample(item, price) value('사과', 2000);
Query OK, 1 row affected (0.02 sec)
```

```
mysql> |
```

```
mysql> select * from sample;
+-----+-----+
| item | price |
+-----+-----+
| 사과 | 2000 |
+-----+-----+
1 row in set (0.00 sec)

mysql> select item from sample;
+-----+
| item |
+-----+
| 사과 |
+-----+
1 row in set (0.00 sec)
```

sample table의 data를 출력합니다.

```
mysql> select * from sample where item='사과';
+-----+-----+
| item | price |
+-----+-----+
| 사과 | 2000 |
+-----+-----+
1 row in set (0.01 sec)

mysql> select * from sample where item='포도';
Empty set (0.00 sec)
```

item이 사과인 data를 출력합니다.

item이 포도인 data는 존재하지 않습니다.

```
mysql> select * from sample where item='사과';
+-----+-----+
| item | price |
+-----+-----+
| 사과 | 2000  |
+-----+-----+
1 row in set (0.00 sec)

mysql> update sample set price=3000 where item='사과';
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from sample where item='사과';
+-----+-----+
| item | price |
+-----+-----+
| 사과 | 3000  |
+-----+-----+
1 row in set (0.00 sec)
```

item이 '사과'인 data의 price를 3000으로 변경합니다.

```
mysql> select * from sample;
+-----+-----+
| item | price |
+-----+-----+
| 사과 | 3000  |
| 포도 | 4000  |
+-----+-----+
2 rows in set (0.00 sec)

mysql> delete from sample where item='사과';
Query OK, 1 row affected (0.02 sec)

mysql> select * from sample;
+-----+-----+
| item | price |
+-----+-----+
| 포도 | 4000  |
+-----+-----+
1 row in set (0.00 sec)
```

item이 사과인 data를 삭제한 것을 확인할 수 있습니다.

```
mysql> show databases;
+-----+
| Database |
+-----+
| basic    |
| information_schema |
| mysql    |
| performance_schema |
| sys      |
| tutorial |
+-----+
6 rows in set (0.00 sec)

mysql> drop database basic
-> ;
Query OK, 1 row affected (0.05 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql            |
| performance_schema |
| sys              |
| tutorial         |
+-----+
5 rows in set (0.00 sec)
```

실습을 위해 생성한 basic database를 삭제합니다.

## 2. 게시물 삭제 기능과 이미지 업로드 및 보여주기 구현 과정 설명 및 캡처

```
JS fileController.js U X
controllers > JS fileController.js > ...
1  var multer = require('multer');
2  const fs = require('fs');
3  var path = require('path');
4
5  const storage = multer.diskStorage({
6    destination: function (req, file, cb) {
7      cb(null, 'public/images/');
8    },
9    filename: function (req, file, cb) {
10      cb(null, Date.now() + path.extname(file.originalname)); // 파일 이름 설정
11    }
12  });
13
14  exports.upload = multer({ storage: storage }).single('image');
15
16
17  // 이미지 삭제 함수
18  exports.deleteImage = (imagePath) => {
19    // 파일 삭제
20    fs.unlink(imagePath, (err) => {
21      if (err) {
22        // 파일이 존재하지 않거나 다른 에러가 발생한 경우
23        if (err.code === 'ENOENT') {
24          console.log('파일이 존재하지 않습니다:', imagePath);
25        } else {
26          console.error('이미지 삭제 중 오류 발생:', err);
27        }
28      } else {
29        console.log('이미지가 성공적으로 삭제되었습니다:', imagePath);
30      }
31    });
32  }
33
34  |
```

이미지 업로드와 삭제에 관한 함수들은 fileController.js 파일에 작성해두었습니다.

storage는 파일 저장 위치, filename은 파일이 저장명입니다.

이미지는 public/images/에 저장합니다.

storage를 multer에 전달하여 image라는 name tag를 가진 req의 파일을 하나 업로드합니다.

deleteImage는 이미지경로를 인자로하여 해당 이미지를 삭제하는 함수입니다.



```
JS writeController.js M X JS writeModel.js M
controllers > JS writeController.js > writeData > fileController.upload() callback
1 var writeModel = require('../models/writeModel');
2 var fileController = require('../controllers/fileController');
3 var express = require('express');
4 var multer = require('multer');
5 var path = require('path');
6
7 exports.writeForm = (req, res) => {
8   res.render('write', {title: "게시판 글 쓰기"});
9 };
10 exports.writeData = (req, res) => {
11   fileController.upload(req, res, function (err) {
12     if (err instanceof multer.MulterError) {
13       // Multer 관련 오류 처리
14       return res.status(500).send('파일 업로드 실패: ' + err.message);
15     } else if (err) {
16       // 기타 오류 처리
17       return res.status(500).send('오류 발생: ' + err.message);
18     }
19
20     // 파일 업로드가 성공적으로 처리된 경우
21     var creator_id = req.body.creator_id;
22     var title = req.body.title;
23     var content = req.body.content;
24     var passwd = req.body.passwd;
25     var image = req.file ? `/images/${req.file.filename}` : null; // 업로드된 이미지 경로 (없으면 null)
26
27     // 데이터베이스에 저장할 데이터 배열
28     var datas = [creator_id, title, content, passwd, image];
29
30     // 데이터베이스에 데이터 삽입
31     writeModel.insertData(datas, () => {
32       res.redirect('/board');
33     });
34   });
35 };

```

writeData에서 fileController.upload를 통해 req object를 가공하고 id, title, content, passwd, filename등을 writeModel의 insertData로 넘겨줍니다.

```
models > JS writeModel.js > <unknown>
1 var mysql = require('mysql');
2 var connection = mysql.createConnection({
3   connectionLimit: 5,
4   host: 'localhost',
5   user: 'root',
6   password: 'guswo3733^SQL',
7   database: 'tutorial'
8 });
9
10 module.exports = {insertData(datas, callback){
11   var sql = "INSERT INTO board(creator_id, title, content, passwd, image) VALUES(?,?,?,?,?)";
12   connection.query(sql, datas, function(err, rows){
13     if(err) console.error("err: " + err);
14     console.log("rows : " + JSON.stringify(rows));
15     callback();
16   });
17 }}
```

넘겨받은 datas를 통해 데이터베이스에 정보를 저장합니다.  
image는 서버에 저장하고 데이터베이스에는 그 이미지의 경로를 저장해줍니다.

```
JS updateController.js M X JS board.js M X JS updateModel.js M
routes > JS board.js > ...
1 var express = require('express');
2 var router = express.Router();
3 var fileController = require('../controllers/fileController');
4 var listController = require('../controllers/listController');
5 var writeController = require('../controllers/writeController');
6 var readController = require('../controllers/readController');
7 var updateController = require('../controllers/updateController');
8 var deleteController = require('../controllers/deleteController');
9 const multer = require('multer');
10
11 router.get('/', listController.getListFirst);
12 router.get('/list/:idx', listController.getList);
13 router.get('/write', writeController.writeForm);
14 router.post('/write', writeController.writeData);
15 router.get('/read/:idx', readController.readData);
16 router.get('/update', updateController.updateForm);
17 router.post('/update', fileController.upload, (req, res, next) => updateController.updateData(req, res));
18 router.get('/delete', deleteController.deleteForm);
19 router.post('/delete', deleteController.deleteData);
20
21 module.exports = router;
```

업로드 또한 fileController.upload를 통해 이미지를 업로드 하고 updateData로 정보를 넘겨줍니다.

```
JS updateController.js M X JS updateModel.js M
controllers > JS updateController.js > updateData > updateData > [0] datas
1 var updateModel = require('../models/updateModel');
2 var express = require('express');
3 var multer = require('multer');
4 var path = require('path');
5 var url = require('url');
6
7 exports.updateForm=(req, res, next)=>{
8   var queryData = url.parse(req.url, true).query;
9   var idx = queryData.idx;
10  updateModel.getData(idx, (rows)=>{
11    console.log('update에서 1개 글 조회 결과 확인 : ', rows);
12    res.render('update', {title: "글 수정", row: rows[0]});
13  });
14 };
15
16
17 exports.updateData = (req, res)=>{
18   console.log("update req: " + req);
19   var idx = req.body.idx;
20   var creator_id = req.body.creator_id;
21   var title = req.body.title;
22   var content = req.body.content;
23   var passwd = req.body.passwd;
24   var image = req.file ? '/images/' + (req.file.filename) : null; // 업로드된 이미지 경로 (없으면 null)
25   var datas = [[creator_id, title, content, image, idx, passwd]];
26   console.log("data : ", datas);
27   console.log("req.body : ", JSON.stringify(req.body));
28   updateModel.updateData(datas, (result)=>{
29     if(result.affectedRows == 0){
30       res.send("<script>alert('패스워드가 일치하지 않거나, 잘못된 요청으로 인해 변경되지 않았습니다. ');history.back();</script>");
31     }
32     else{
33       res.redirect('/board/read/' + idx);
34     }
35   });
36 };
```

writeData와 비슷하게 idx, id, title, content, passwd, filename 등을 받아서 updateModel의 updateData로 넘겨줍니다.

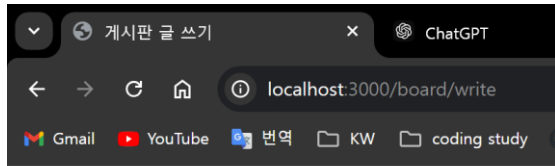
updateData에서 result인자를 callback 함수에 전달해주면 result를 통해 database가 수정됐는지 여부를 판단하여, 수정됐다면 board/read로 경로를 변경하고 수정되지 않았다면 알림을 띄웁니다.

```
JS updateModel.js M X
models > JS updateModel.js > updateData > updateData
1 var mysql = require('mysql');
2 var fileController = require('../controllers/fileController')
3
4 var connection = mysql.createConnection({
5   connectionLimit: 5,
6   host: 'localhost',
7   user: 'root',
8   password: 'guswo3733^SQL',
9   database: 'tutorial'
10 });
11
12 exports.getData = (idx, callback)=>{
13   connection.query('SELECT idx, creator_id, title, content, image FROM board WHERE idx=?;', idx, (err, rows, fields)=>{
14     if(err) throw err;
15     callback(rows);
16   });
17 }
18
19 exports.updateData=(datas, callback)=>{
20   // 사진 수정 X
21   if (datas[3] === null){
22     new_datas = [datas[0], datas[1], datas[2], datas[4], datas[5]]
23     var sql = "UPDATE board SET creator_id=?, title=?, content=? WHERE idx=? AND passwd=?;";
24     connection.query(sql, new_datas, function(err, result){
25       if(err) console.error("글 수정 중 에러 발생 err: " + err);
26       callback(result);
27     });
28   }
29   // 사진 수정 d
30   else{
31     connection.query('SELECT image From board where idx = ?', datas[4], (err, rows)=>{
32       var sql = "UPDATE board SET creator_id=?, title=?, content=?, image=? WHERE idx=? AND passwd=?;";
33       connection.query(sql, datas, function(err, result){
34         if(err) console.error("글 수정 중 에러 발생 err: " + err);
35         else if(result.affectedRows) fileController.deleteImage("public" + rows[0].image);
36         callback(result);
37       });
38     });
39   }
}
```

updateData를 보면,  
새로운 사진이 입력되지 않았다면 이미지를 제외한 나머지 정보들만 update하도록 하고, 새로운 사진이 입력됐다면 기존의 이미지를 삭제하고 데이터베이스에 새로운 이미지 경로를 update해줍니다.

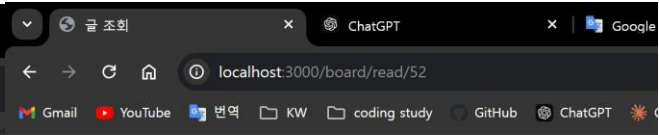
```
JS readController.js JS readModel.js M X
models > JS readModel.js > <unknown>
1 var mysql = require('mysql');
2 var connection = mysql.createConnection({
3   connectionLimit: 5,
4   host: 'localhost',
5   user: 'root',
6   password: 'guswo3733^SQL',
7   database: 'tutorial'
8 });
9
10 module.exports={getData(idx, callback){
11   var sql = "SELECT idx, creator_id, title, content, hit, image FROM board WHERE idx=?;";
12   connection.query(sql, idx, (err, row, fields)=>{
13     if(err) throw err;
14     callback(row);
15   });
16 }}
```

readModel의 getData에도 image의 경로를 추가해줌으로써 이미지를 조회할 수 있도록 했습니다.

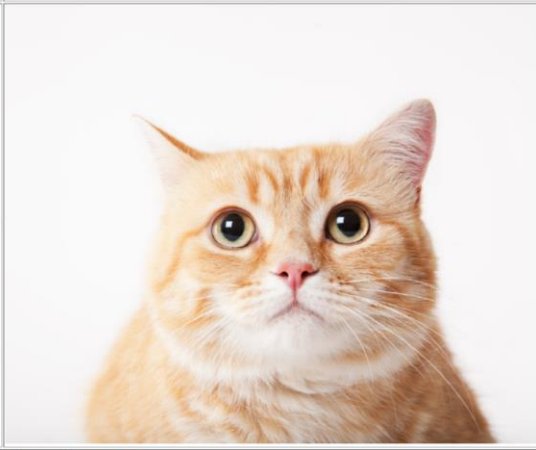


## 게시판 글 쓰기

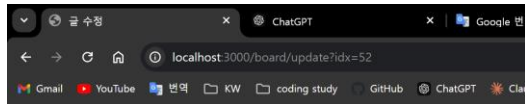
작성자	김재현
제목	image upload
내용	cat
이미지	파일 선택 2023071701753_0.jpg
패스워드	****
글쓰기	



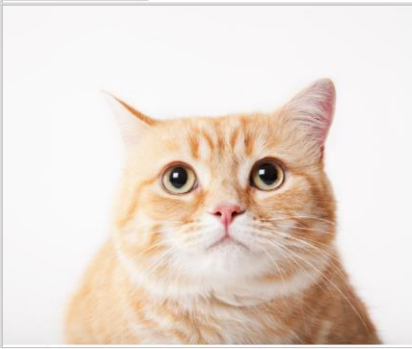
## 글 조회

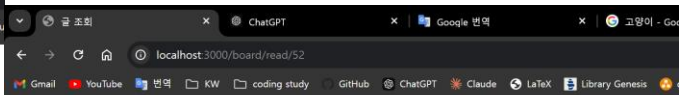
작성자	김재현
제목	image upload
내용	cat
조회수	0
이미지	
<a href="#">글 수정</a> <a href="#">글 삭제</a> <a href="#">리스트로 돌아가기</a>	

새로운 글을 올리고 조회하는 화면입니다.




## 글 수정

작성자	김재현
제목	image modifying
내용	another cat
이미지	
이미지 변경	파일 선택 NIS2023071...3626_web.jpg
패스워드	****
글쓰기	



## 글 조회

작성자	김재현
제목	image modifying
내용	another cat
조회수	0
이미지	
<a href="#">글 수정</a> <a href="#">글 삭제</a> <a href="#">리스트로 돌아가기</a>	

올린 글을 수정하고 조회하는 화면입니다.

```

JS deleteController.js U X JS deleteModel.js U
controllers > JS deleteController.js > deleteData > deleteModel.deleteData() callback
1 var deleteModel = require('../models/deleteModel');
2 var express = require('express');
3 var url = require('url');
4
5 exports.deleteForm = (req, res)=>{
6   var queryData = url.parse(req.url, true).query;
7   var idx = queryData.idx;
8   res.render('delete', {title: "글 삭제", idx: idx});
9 };
10 exports.deleteData=(req, res)=>{
11   var idx = req.body.idx;
12   var passwd = req.body.passwd;
13   deleteModel.deleteData(idx, passwd, (result)=>{
14     if (result.affectedRows)
15       res.redirect('/board');
16     else
17       res.send("<script>alert('패스워드가 일치하지 않거나, 잘못된 요청으로 인해 삭제되지 않았습니다.');"</script>");
18   });
19 };
20 };

```

/board/read/ 페이지에서 삭제 버튼을 누르면 deleteController의 deleteForm이 호출됩니다. 비밀번호를 입력하는 /board/delete/ 페이지로 이동합니다.  
비밀번호를 입력하고 확인 버튼을 누르면 deleteData가 호출되고, idx와 passwd를 deleteModel의 deleteData로 전달합니다.  
만일 passwd가 틀려서 삭제가 진행되지 않는다면 알람을 띄우고, 삭제가 진행됐다면 /board로 이동합니다.

```

JS deleteController.js U X JS deleteModel.js U X
models > JS deleteModel.js > deleteData > connection.query('SELECT image From board where idx = ?') callback > connection.query('D
1 var mysql = require('mysql');
2 var fileController = require('../controllers/fileController')
3
4 var connection = mysql.createConnection({
5   connectionLimit: 5,
6   host: 'localhost',
7   user: 'root',
8   password: 'guswo3733^SQL',
9   database: 'tutorial'
10 });
11
12 module.exports={deleteData(idx, passwd, callback){
13   connection.query('SELECT image From board where idx = ?', idx, (err, rows)=>{
14     connection.query('DELETE FROM board where idx=? AND passwd=?', [idx, passwd], (err, result, fields)=>{
15       if(err) throw err;
16       else if(result.affectedRows) fileController.deleteImage("public" + rows[0].image);
17       callback(result);
18     });
19   });
20 };
21
22 }}

```

idx가 일치하는 데이터의 이미지경로를 통해 서버에 저장된 이미지를 삭제하고, 데이터베이스에서 해당 튜플을 삭제합니다.

## 게시판 전체 글 조회


[글쓰기로 이동](#)

번호	작성자	제목	조회수
52	김재현	<a href="#">image modifying</a>	0

### 글 조회

작성자	김재현
제목	image modifying
내용	another cat
조회수	0

이미지



글 수정 글 삭제 [리스트로 돌아가기](#)

## 글 삭제

패스워드	<input type="password"/>
확인	<input type="button" value="확인"/>

[리스트로 돌아가기](#)

## 게시판 전체 글 조회

[글쓰기로 이동](#)

번호	작성자	제목	조회수
----	-----	----	-----

게시판의 글을 삭제하는 과정입니다.

## 게시판 전체 글 조회

[글쓰기로 이동](#)

번호	작성자	제목	조회수
53	김재현	<a href="#">2020202031</a>	0

게시글에 학번 이름 작성했습니다.

### 3. 고찰

update 함수에서 먼저 data를 가공하고 이미지 삭제를 진행한 후 이미지 추가를 하려고 했는데, req에 값들이 undefined 상태인 것을 확인했습니다.

여기서 multipart/form-data 형태의 파일을 포함하는 request는 multer를 사용해야 정보가 가공되는 것을 알 수 있었습니다.

그래서 먼저 multer().upload()를 통해 파일을 업로드 한 후 이미지를 삭제하는 절차를 밟았더니 의도대로 코드가 실행되는 것을 경험했습니다.