

Hello,

KDT 웹 개발자 양성 프로젝트

5기!

with





Express 기본!

- 기본 세팅



Express 기본 구조 만들기!

- 프로젝트 용 폴더 생성

```
✓ EXPRESS  
  .gitattributes
```

- Npm init -y

- 명령어를 통해서 node package manager 기본 세팅 완료 하기!

```
✓ EXPRESS  
  .gitattributes  
  {} package.json      U
```



깃 이그노어 추가하기!

- Package 파일들일 설치 될 node_modules 폴더는 미리 .gitignore 에 추가하기!



- Npm init -y
 - 명령어를 통해서 node package manager 기본 세팅 완료 하기!

필요 모듈 설치 하기!



- 배포 시 필요한 패키지 / npm install -save 옵션(npm i -S)
 - Express
 - cors
 - Ejs

```
lhs@DESKTOP-86MUCGC MINGW64 /d/git/express (main)
$ npm i -S express cors ejs

added 75 packages, and audited 76 packages in 4s

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

found 0 vulnerabilities
```

필요 모듈 설치 하기!



- 개발 시에만 필요한 패키지 / `npm install --save-dev` 옵션(`npm i -D`)
 - Prettier
 - Eslint / `eslint-config-airbnb-base` / `eslint-plugin-import`

```
lhs@DESKTOP-86MUCGC MINGW64 /d/git/express (main)
$ npm i -D prettier eslint eslint-config-airbnb-base eslint-plugin-import

added 158 packages, and audited 234 packages in 10s

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

found 0 vulnerabilities
```

Prettier / Eslint 설정 파일 가져오기!



- 기존에 세팅 해놓은 파일을 가지고 와서 바로 적용 합시다!
- 프로젝트 폴더 최 상위에
- .pritterrc / .eslintrc.js 파일 복사 붙여 넣기
- .vscode 폴더도 복사 붙여 넣기!
- 설치 커맨드를 저장한 파일을 만들어도 편합니다~!

≡ set-command.txt

```
1  npm i -S express cors ejs
2  npm i -D prettier eslint eslint-config-airbnb-base eslint-plugin-import
3  |
```



Express 기본!

- 서버 만들기!

서버 실행 코드 작성!



- App.js 파일을 생성하고 기본 서버를 생성하는 코드를 작성해 봅시다!

```
const express = require('express');
const cors = require('cors');

const app = express();
const PORT = 4000;

app.use(cors());

app.listen(PORT, () => {
  console.log(`서버는 ${PORT}번 포트에서 실행 중입니다!`);
});
```

서버 실행 하기!



- nodemon app.js 로 서버 실행하기!

```
lhs@DESKTOP-86MUCGC MINGW64 /d/git/express (main)
$ nodemon app.js
[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
```

```
New version of nodemon available!
Current Version: 2.0.20
Latest Version: 2.0.21
```

서버는 4000번 포트에서 실행 중입니다!

서버에 주소 요청을 받아줄 미들웨어 설정!



- 힘들게 찾아온 요청을 받아 줄 미들웨어를 만들어 줍시다!

```
const express = require('express');  
const cors = require('cors');
```

```
const app = express();  
const PORT = 4000;
```

```
app.use(cors());
```

```
app.get('/', (req, res) => {  
  res.send('어서와 Express 는 처음이지!~');  
});
```

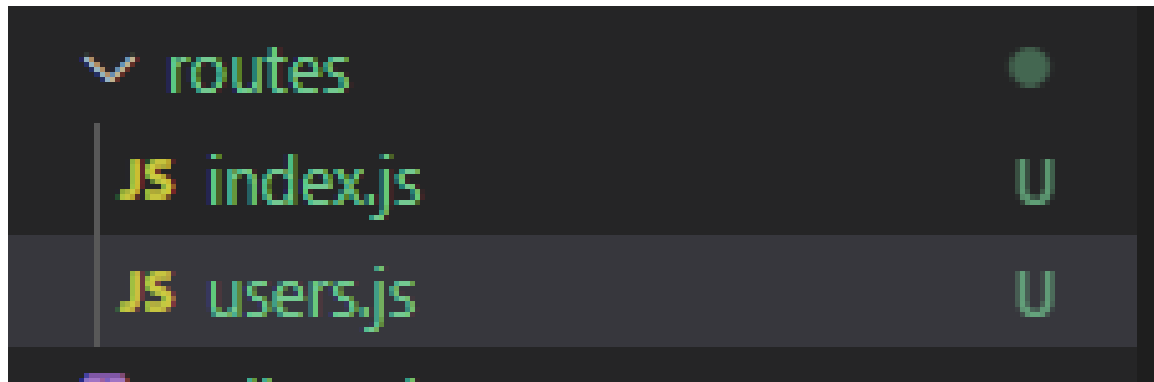
```
app.listen(PORT, () => {  
  console.log(`서버는 ${PORT}번 포트에서 실행 중입니다!`);  
});
```

미들웨어

주소별로 업무를 나누자! 라우팅!



- Routes 폴더에 메인 라우터를 위한 index.js 파일과 회원 관련 기능을 위한 users.js 파일 생성



메인 라우터 작업!



- 가장 기본이 되는 요청 (<http://localhost:4000/>) 을 처리하는 index.js 작업

```
const express = require('express');  
const router = express.Router();  
  
router.get('/', (req, res) => {  
  res.send('여기는 메인 라우터 입니다!');  
});  
  
module.exports = router;
```

Express Router 를 호출하고
Router 변수에 넣어주기!

<http://localhost:4000/>
요청에 대한 처리!

외부(app.js 서버 코드)에서 사용이
가능하도록 모듈로 빼주기!

메인 라우터를 서버에 적용!



- mainRouter 역할을 할, index.js 모듈을 불러오고 서버에 추가해 줍시다!

```
const express = require('express');
const cors = require('cors');

const app = express();
const PORT = 4000;

app.use(cors());

const mainRouter = require('./routes');

app.use('/', mainRouter);

app.listen(PORT, () => {
  console.log(`서버는 ${PORT}번 포트에서 실행 중입니다!`);
});
```

Index.js 파일을 모듈로 불러오기
Index.js 는 생략 가능!

서버 주소인
http://localhost:4000 이후에
들어오는 주소가 '/' 로 들어오면

메인 라우터로 요청을 보내도록
설정해주기!



server

<http://localhost:4000/>

<http://localhost:4000/users>

<http://localhost:4000/>



server

<http://localhost:4000/>

<http://localhost:4000/users>

`http://localhost:4000/users`



server

`http://localhost:4000/`

`http://localhost:4000/users`

뷰 엔진 적용 하기!



- 백엔드 서버에서 데이터를 바로 받아서 그릴 수 있는 뷰 엔진을 적용

```
const express = require('express');
const cors = require('cors');
const app = express();
const PORT = 4000;

app.use(cors());
app.set('view engine', 'ejs');

const mainRouter = require('./routes');
const userRouter = require('./routes/users');
app.use('/', mainRouter);
app.use('/users', userRouter);

app.listen(PORT, () => {
  console.log(`서버는 ${PORT}번 포트에서 실행 중입니다!`);
});
```

뷰엔진을 ejs 로 세팅한다는 코드 추가

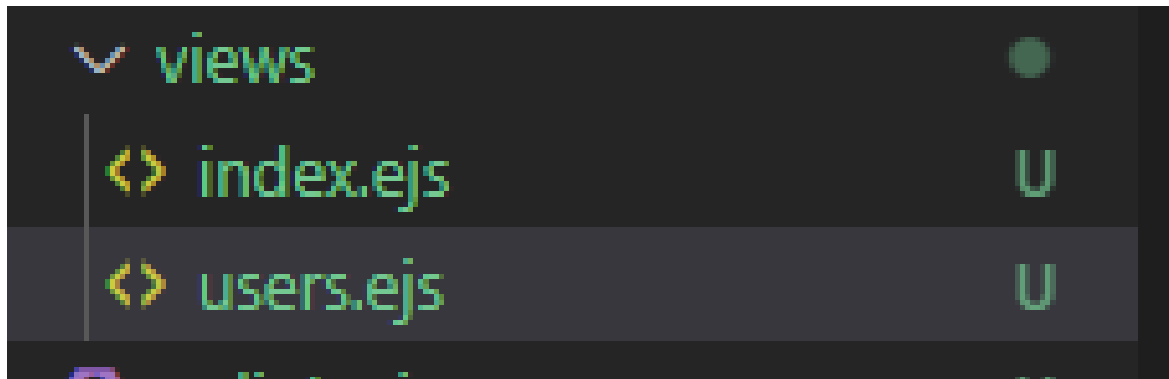
해당 코드를 추가하면 Express 가
알아서 프로젝트 폴더의 views 폴더를
인식하게 됩니다!

그리고 ejs 파일을 찾아가죠!

뷰 폴더 생성 & 뷰 파일 설정!



- View 파일이 저장되는 views 폴더 생성하기
- 메인 페이지를 위한 index.ejs 파일과 회원 서비스를 위한 users.ejs 파일 생성 하기!



뷰 파일에 데이터 전달하기!



- 뷰 파일에 백엔드 데이터를 전달하려면, `res.render` 메소드의 두 번째 인자로 객체를 전달하고 해당 객체에 데이터를 넣어 주면 됩니다!

```
const express = require('express');  
const router = express.Router();  
  
router.get('/', (req, res) => {  
  res.render('index', { msg: '이 데이터는 백엔드가 보냈어요!' });  
});  
  
module.exports = router;
```

Routes/index.js

`Res.render('뷰파일명', {데이터})`
을 통해서 백엔드의 데이터를
뷰파일에 전달 가능!

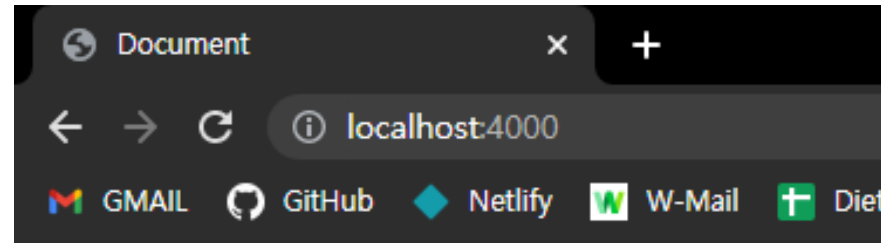
뷰 파일에서 백엔드 데이터 사용하기!



- 뷰 파일에서는 ejs 문법 `<% %>` or `<%= %>` 을 사용하여 데이터 출력이 가능합니다!

```
res.render('index', { msg: '이 데이터는 백엔드가 보냈어요!' });
```

```
<body>  
  <h1>메인 페이지 입니다!</h1>  
  <p><%= msg %></p>  
</body>
```



메인 페이지 입니다!

이 데이터는 백엔드가 보냈어요!



스태틱 폴더 설정 하기!

- `express.static('지정할 폴더명')` 을 통해 지정 가능!
- `App.use` 를 사용하여 스태틱 폴더 사용을 서버에 알려주기!

```
app.use(cors());  
app.set('view engine', 'ejs');  
app.use(express.static('public'));
```

- 스태틱 폴더는 프로젝트 폴더 최상위에 존재해야 합니다!



```
<!DOCTYPE html>
<html lang="en">

<head>
  <link rel="stylesheet" href="/css/style.css">
  <script defer src="/js/main.js"></script>
</head>

<body>
  <h1>메인 페이지 입니다!</h1>
  <p><%= msg %></p>
  
</body>

</html>
```

localhost:4000 내용:

!

확인

메인 페이지 입니다!

이 데이터는 백엔드가 보냈어요!





게시판
만들기!!



Board.js 로 라우팅!

- 먼저 board.js 파일을 만들고 express 모듈 추가 등의 기본 코드만을 추가하고 모듈화 작업

```
const express = require('express');  
  
const router = express.Router();  
  
module.exports = router;
```

- App.js 에서 해당 모듈 불러오고 주소 라우팅 설정

```
const boardRouter = require('./routes/board');  
app.use('/board', boardRouter);
```

```
router.get('/', (req, res) => {
  // 글 전체 목록이 보이는 페이지
});

router.get('/write', (req, res) => {
  // 글 쓰기 모드로 이동
});

router.post('/write', (req, res) => {
  // 글 추가
});

router.get('/modify/:title', (req, res) => {
  // 글 수정 모드로 이동
});

router.post('/modify/:title', (req, res) => {
  // 글 수정
});

router.delete('/delete/:title', (req, res) => {
  // 글 삭제
});
```





전체 목록 보여주기



전체 목록 보여주기

- 글 전체 목록을 데이터에 담아서 board.ejs 파일 그려주기
- `res.render(뷰파일명, 데이터);` 사용
- Ejs 코드에서 사용한 변수명과 일치하는지 체크 필요

```
router.get('/', (req, res) => {  
  const articleCounts = ARTICLE.length;  
  res.render('board', { ARTICLE, articleCounts });  
});
```



글 쓰기 모드로 이동



글 쓰기 모드로 이동

- Board.ejs 뷰에서 글쓰기 버튼을 클릭 → </board/write> 라는 주소로 이동
- 해당 요청이 들어오면 글 쓰기 페이지 board_write.ejs 를 그려주기

```
<div class="board_write">
  <span>현재 등록 글 : &nbsp;&nbsp;&nbsp;<%= articleCounts %></span>
  <a class="btn red" href="/board/write">글쓰기</a>
</div>
```

- 데이터 전달 필요 X

```
router.get('/write', (req, res) => {
  res.render('board_write');
});
```



글 추가 기능



글 추가 기능(프론트)

- 글쓰기 모드에서 입력 받은 title, content 를 새로운 글로 추가하기
- 아무것도 안 쓸 경우의 예외를 처리하기 위해 필수 입력 지정(required)
- 주소는 </board/write>, 메소드는 POST 로 전달

```
<form action="/board/write" method="POST" class="board_form">
  <div class="form_title">
    <h3>제목</h3>
    <input type="text" name="title" required />
  </div>
  <div class="form_content">
    <h3>내용</h3>
    <textarea name="content" id="content" cols="30" rows="10" required></textarea>
  </div>
  <button type="submit">글 작성하기</button>
</form>
```




```
router.post('/', (req, res) => {  
  if (req.body.title && req.body.content) {  
    const newArticle = {  
      title: req.body.title,  
      content: req.body.content,  
    };  
    ARTICLE.push(newArticle);  
    res.redirect('/board');  
  } else {  
    const err = new Error('요청 이상');  
    err.statusCode = 404;  
    throw err;  
  }  
});
```



글 수정 모드로 이동



글 수정 모드로 이동

- 뷰에서 수정 버튼을 클릭 → </board/modify/:title> 라는 주소로 이동
- title 파라미터는 ejs 에서 직접 입력하여 전달

```
<a class="btn orange" href="board/modify/<%= ARTICLE[i].title %>">수정</a>
```

- 데이터 전달 필요 O, ARTICLE 배열에서 제목이 같은 글을 찾아 전달

```
router.get('/modify/:title', (req, res) => {  
  const arrIndex = ARTICLE.findIndex(  
    (article) => article.title === req.params.title  
  );  
  const selectedArticle = ARTICLE[arrIndex];  
  res.render('board_modify', { selectedArticle });  
});
```



글 수정 모드 뷰 페이지

- 전달 받은 데이터(selectedArticle)를 `<input>` `<textarea>` 값으로 전달
- 데이터 전송은 수정할 글의 title 을 파라미터로 담아서
`board/modify/:title` 로 전달 / 메소드는 **POST**

```
<form action="/board/modify/<%=selectedArticle.title%>" method="POST" class="board_form">
  <div class="form_title">
    <h3>제목</h3>
    <input type="text" name="title" value="<%= selectedArticle.title %>" required />
  </div>
  <div class="form_content">
    <h3>내용</h3>
    <textarea name="content" id="content" cols="30" rows="10" required>
      <%= selectedArticle.content %></textarea>
    </div>
  <button type="submit">글 수정하기</button>
</form>
```



글 수정 기능



```
router.post('/modify/:title', (req, res) => {
  if (req.body.title && req.body.content) {
    const arrIndex = ARTICLE.findIndex(
      (article) => article.title === req.params.title
    );
    if (arrIndex !== -1) {
      ARTICLE[arrIndex].title = req.body.title;
      ARTICLE[arrIndex].content = req.body.content;
      res.redirect('/board');
    } else {
      const err = new Error('해당 제목의 글이 없습니다. ');
      err.statusCode = 404;
      throw err;
    }
  } else {
    const err = new Error('요청 쿼리 이상');
    err.statusCode = 404;
    throw err;
  }
});
```



글 삭제 기능



글 삭제 기능(프론트)

- 삭제는 **DELETE** 요청을 해야 합니다! → 그런데 지금 우리는 A 태그에서만 요청을 보낼 수 있습니다!
- 따라서, 이전에 배운 JS의 fetch 를 사용합니다!

```
<a class="btn blue" href="#" onclick="deleteArticle('<%= ARTICLE[i].title %>')">삭제</a>
```




글 삭제 기능(프론트)

- Fetch 를 통해 삭제 요청을 보냅니다
- 그리고 요청이 완료 되면 페이지를 다시 읽어 들어서 삭제 사항이 반영 되도록 합니다.

```
function deleteArticle(title) {  
  fetch(`board/delete/${title}`, {  
    method: 'delete',  
    headers: {  
      'Content-type': 'application/json'  
    },  
  }).then((res) => {  
    location.href = '/board';  
  })  
}
```



```
router.delete('/delete/:title', (req, res) => {
  const arrIndex = ARTICLE.findIndex(
    (article) => article.title === req.params.title
  );
  if (arrIndex !== -1) {
    ARTICLE.splice(arrIndex, 1);
    res.send('삭제 완료!');
  } else {
    const err = new Error('해당 제목을 가진 글이 없습니다. ');
    err.statusCode = 404;
    throw err;
  }
});
```



지금 시작합니다





DataBase



데이터 들의 집합



DBMS

(DataBase Management System)



DBMS

- 데이터 베이스를 관리하고 운영하는 SW
- 다양한 형태, 서비스가 존재 합니다!





SQL

(Structured Query Language)

SQL



- 구조가 있는 질문용 언어라는 뜻
- SELECT, INSERT, UPDATE, DELETE 같은 언어를 통해 데이터 베이스의 데이터를 다루는 언어
- 다수의 DBMS가 SQL 방식을 따르기 때문에 SQL 을 배우면, 많은 DBMS를 비교적 빠르게 습득이 가능
- MySQL, SQLite, ORACLE 등이 SQL 구문을 사용



DB의 종류



관계형(SQL)

VS

비관계형(NoSQL)



관계형 DB



관계형, Relational DBMS(RDBMS)

- RDBMS 는 SQL 을 사용하는 DB 입니다
- 키와 값의 관계를 테이블화 시킨 원칙을 토대로 DB 를 구성합니다

아이디	이름	연락처
flower	화사	010-1111-1111
finetree	솔라	010-2222-2222
moon	문별	010-3333-3333
whee	휘인	010-4444-4444



관계형, Relational DBMS(RDBMS)

- 즉, RDBMS 는 테이블로 구성이 됩니다.
- 먼저 테이블이 구성되고 테이블의 구조에 맞추어 데이터가 들어가기 때문에 DB를 구성하기 전에 스키마라 불리는 DB의 구조, 관계, 제약 사항에 대한 정의가 필요합니다
- 사실 예전엔 컴퓨터가 느리고 용량이 작아서 이런 구조가 아니면 속도 측면과 용량면에서 답이 없었습니다 ☺



관계형, Relational DBMS(RDBMS)

- 장점

- 구조화가 명확하게 되어 있어서 예외가 없음
- 데이터 입, 출력 속도가 매우 빠릅니다
- 신뢰성이 매우 높음

- 단점

- DB의 구조 변경이 매우 어려움 → 빅데이터 등에는 사용이 어려움(새로운 키가 추가 되면 전체 스키마 변경이 필요)



비관계형 DB

비관계형, Non Relational DBMS(NoSQL)



- SQL 을 사용하지 않는 모든 DB를 통칭합니다
 - ex) 한국어 vs 비한국어(= 영어, 프랑스어, 스페인어 등등)
- 대표적으로 문서형, 그래프형, 키밸류형, 와이드 컬럼형 등이 있습니다!
- 약간 컴퓨터 언어와 일맥 상통하는 부분으로 특정 목적에 맞는 DB가 존재 합니다 (ex. Html → 웹 개발 / 파이썬 → 인공지능 / Swift → 앱)

비관계형, Non Relational DBMS(NoSQL)



- 장점

- 보통 대용량 데이터 처리에 효율적
- DB의 구조 변경이 쉽고, 확장성이 뛰어남
- 복잡한 데이터 구조의 표현이 가능

- 단점

- 데이터 자체가 크면 전체 데이터를 일부 읽어서 처리해야 하므로 데이터가 크면 속도가 저하되는 문제 발생

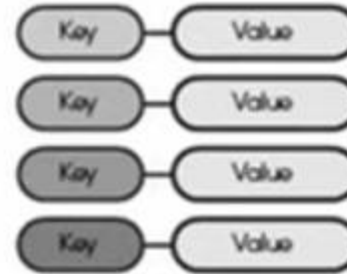
Document



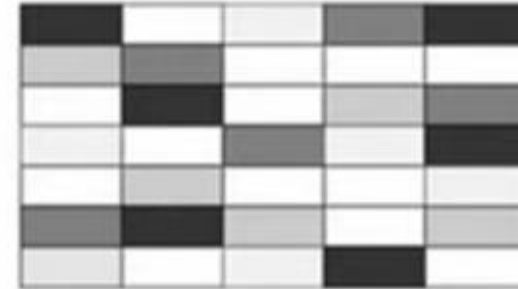
Graph



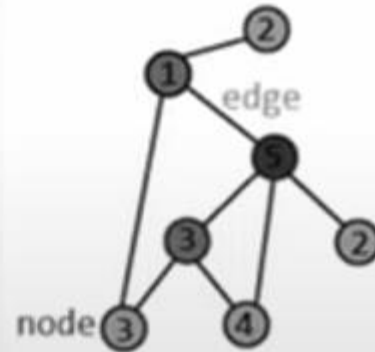
Key-Value



Wide-Column



```
{
  "user": {
    "id": "143",
    "name": "improgrammer",
    "city": "New York"
  }
}
```





1	Fruit	A Foo	B Baz	
2	City	E DC	D PLA	G FLD
3	State	A NZ	C CL	





https://youtu.be/Q_9cFgzZr8Q



개발자	오라클
발표일	1995년 5월 23일
라이선스	GPL v2 (커뮤니티 에디션) 상용 라이선스 (표준, 엔터프라이즈, 클러스터)
링크	 



https://www.w3schools.com/sql/trysql.asp?filename=trysql_select_all

SQL Statement:

```
SELECT * FROM Customers;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Click "**Run SQL**" to execute the SQL statement above.

W3Schools has created an SQL database in your browser.

The menu to the right displays the database, and will reflect any changes.

Feel free to experiment with any SQL statement.

You can restore the database at any time.

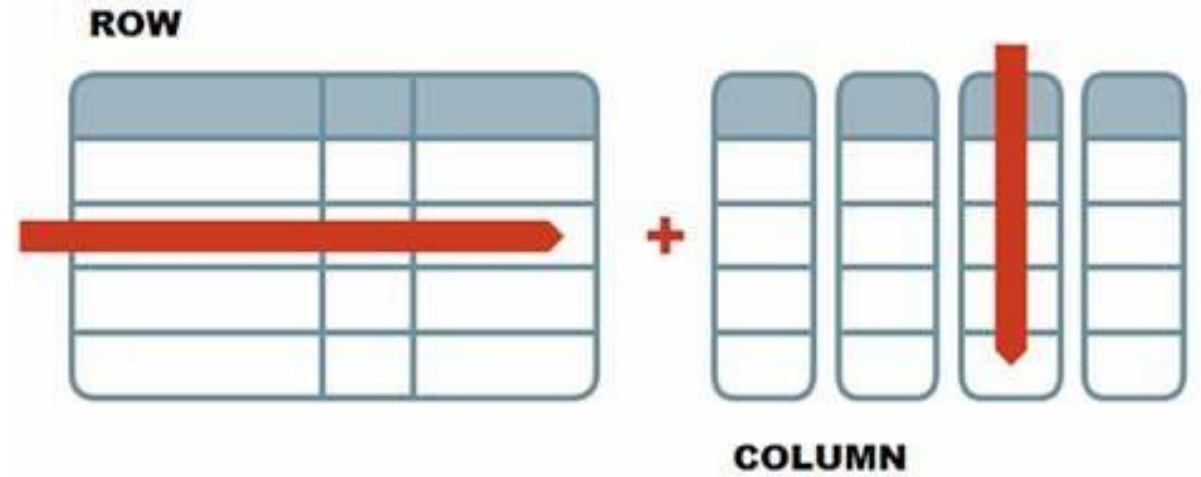




Diagram illustrating a table structure with labels and arrows:

- 행(row)**: Points to the rows of the table.
- 열 이름**: Points to the header row.
- 열(column)**: Points to the columns of the table.

아이디	이름	연락처
flower	화사	010-1111-1111
finetree	솔라	010-2222-2222
moon	문별	010-3333-3333
whee	휘인	010-4444-4444





SELECT



SELECT, DB 에서 원하는 데이터 가져오기

- 갓춰진 DB 에서 데이터를 뽑아 올 때 사용하는 구문 입니다!
- 일단
- `SELECT * FROM Customer;`

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden
6	Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	68306	Germany
7	Blondel père et fils	Frédérique Citeaux	24, place Kléber	Strasbourg	67000	France
8	Bólido Comidas preparadas	Martín Sommer	C/ Araquil, 67	Madrid	28023	Spain
9	Bon app'	Laurence Lebihans	12, rue des Bouchers	Marseille	13008	France
10	Bottom-Dollar Marketse	Elizabeth Lincoln	23 Tsawassen Blvd.	Tsawassen	T2F 8M4	Canada
11	B's Beverages	Victoria Ashworth	Fauntleroy Circus	London	EC2 5NT	UK
12	Cactus Comidas para llevar	Patricio Simpson	Cerrito 333	Buenos Aires	1010	Argentina
13	Centro comercial Moctezuma	Francisco Chang	Sierras de Granada 9993	México D.F.	05022	Mexico
14	Chop-suey Chinese	Yang Wang	Hauptstr. 29	Bern	3012	Switzerland
15	Comércio Mineiro	Pedro Afonso	Av. dos Lusíadas, 23	São Paulo	05432-043	Brazil



원하는 컬럼의 값만 가지고 올 때?

- 모든 컬럼 값이 필요 한게 아니라면!?
- `SELECT 컬럼명 FROM table`
- `SELECT City, Country FROM Customers`

SQL Statement:



```
SELECT City, Country FROM Customers;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 91

City	Country
Berlin	Germany
México D.F.	Mexico
México D.F.	Mexico
London	UK
Luleå	Sweden
Mannheim	Germany
Strasbourg	France

DB에는 영향 없이 원하는 데이터 추가 할 때!



- 필요한 값을 임의로 추가해서 가져오고 싶을 땐?
- `SELECT` 컬럼명, 원하는 데이터 `FROM` table
- `SELECT` City, 1, '원하는 문자열', NULL `FROM` Customers

SQL Statement:

SELECT City, 1, '원하는 문자열', NULL FROM Customers

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 91

1	City	'원하는 문자열'	NULL
1	Berlin	원하는 문자열	null
1	México D.F.	원하는 문자열	null
1	México D.F.	원하는 문자열	null
1	London	원하는 문자열	null
1	Luleå	원하는 문자열	null
1	Mannheim	원하는 문자열	null
1	Strasbourg	원하는 문자열	null

원하는 조건을 충족하는 Row 만 가져 올 때!



- 원하는 조건을 충족 하는 값을 찾고 싶을 땐? **WHERE**
- **SELECT * FROM table WHERE 조건**
- **SELECT * FROM OrderDetails WHERE Quantity > 5;**

SQL Statement:

SELECT * FROM OrderDetails WHERE Quantity > 5;

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 476

OrderDetailID	OrderID	ProductID	Quantity
1	10248	11	12
2	10248	42	10
4	10249	14	9
5	10249	51	40
6	10250	41	10
7	10250	51	35
8	10250	65	15
9	10251	22	6



Row 를 정렬해서 가져 올 때!

- 선택한 값들을 정렬해서 가지고 올 때는? ORDER BY
- SELECT * FROM Customers ORDER BY ContactName;
- SELECT * FROM Customers ORDER BY ContactName DESC;
- SELECT * FROM Customers ORDER BY CustomerName ASC, ContactName DESC;

SQL Statement:

```
SELECT * FROM Customers ORDER BY ContactName;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 91

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
69	Romero y tomillo	Alejandra Camino	Gran Vía, 1	Madrid	28001	Spain
52	Morgenstern Gesundkost	Alexander Feuer	Heerstr. 22	Leipzig	04179	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
81	Tradição Hipermercados	Anabela Domingues	Av. Inês de Castro, 414	São Paulo	05634-030	Brazil
31	Gourmet Lanchonetes	André Fonseca	Av. Brasil, 442	Campinas	04876-786	Brazil
19	Eastern Connection	Ann Devon	35 King George	London	WX3 6FW	UK
41	La maison d'Asie	Annette Roulet	1 rue Alsace-Lorraine	Toulouse	31000	France
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
21	Familia Arquibaldo	Aria Cruz	Rua Orós, 92	São Paulo	05442-030	Brazil
75	Split Rail Beer & Ale	Art Braunschweiger	P.O. Box 555	Lander	82520	USA

SQL Statement:

```
SELECT * FROM Customers ORDER BY ContactName DESC;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 91

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland
54	Océano Atlántico Ltda.	Yvonne Moncada	Ing. Gustavo Moncada 8585 Piso 20-A	Buenos Aires	1010	Argentina
42	Laughing Bacchus Wine Cellars	Yoshi Tannamuri	1900 Oak St.	Vancouver	V3F 2K1	Canada
36	Hungry Coyote Import Store	Yoshi Latimer	City Center Plaza 516 Main St.	Elgin	97827	USA



Row 의 개수를 지정 하고 싶을 때!

- 가지고 오는 ROW의 수를 지정하고 싶을 때? **LIMIT**
- **SELECT * FROM Customers LIMIT 10;**
- **SELECT * FROM Customers LIMIT 30, 10;**
 - 원하는 Row 순번, 자르는 개수

SQL Statement:

```
SELECT * FROM Customers LIMIT 10;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 10

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden
6	Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	68306	Germany
7	Blondel père et fils	Frédérique Citeaux	24, place Kléber	Strasbourg	67000	France
8	Bólide Comidas preparadas	Martín Sommer	C/ Araquil, 67	Madrid	28023	Spain
9	Bon app'	Laurence Lebihans	12, rue des Bouchers	Marseille	13008	France
10	Bottom-Dollar Marketse	Elizabeth Lincoln	23 Tsawassen Blvd.	Tsawassen	T2F 8M4	Canada

Column 명을 변경해서 가지고 오고 싶을 때?



- Column 명을 변경해서 가지고 올 때? **AS**
- **SELECT** CustomerId **AS** id, CustomerName **AS** name **FROM**
Customers;



SQL Statement:

```
SELECT CustomerId AS id, CustomerName AS name FROM Customers;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 91

id	name
1	Alfreds Futterkiste
2	Ana Trujillo Emparedados y helados
3	Antonio Moreno Taquería
4	Around the Horn
5	Berglunds snabbköp
6	Blauer See Delikatessen



테이블 합치기!

JOIN

여러 테이블의 값을 하나로 합치고 싶다면!?



- 여러 테이블을 하나로 붙이고 싶을 땐? JOIN
- `SELECT * FROM Products P JOIN Categories C ON P.CategoryID = C.CategoryID;`

SQL Statement:

SELECT * FROM Products P JOIN Categories C ON P.CategoryID = P.CategoryID;

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 616

ProductID	ProductName	SupplierID	CategoryID	Unit	Price	CategoryName	Description
1	Chais	1	1	10 boxes x 20 bags	18	Beverages	Soft drinks, coffees, teas, beers, and ales
1	Chais	1	2	10 boxes x 20 bags	18	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings
1	Chais	1	3	10 boxes x 20 bags	18	Confections	Desserts, candies, and sweet breads
1	Chais	1	4	10 boxes x 20 bags	18	Dairy Products	Cheeses
1	Chais	1	5	10 boxes x 20 bags	18	Grains/Cereals	Breads, crackers, pasta, and cereal
1	Chais	1	6	10 boxes x 20 bags	18	Meat/Poultry	Prepared meats
1	Chais	1	7	10 boxes x 20 bags	18	Produce	Dried fruit and bean curd
1	Chais	1	8	10 boxes x 20 bags	18	Seafood	Seaweed and fish
2	Chang	1	1	24 - 12 oz bottles	19	Beverages	Soft drinks, coffees, teas, beers, and ales
2	Chang	1	2	24 - 12 oz bottles	18	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings





Local 에서
연습 시작!

MySQL 설치하기!



- Windows

- <https://velog.io/@joajoa/MySQL-%EB%8B%A4%EC%9A%B4%EB%A1%9C%EB%93%9C-%EB%B0%8F-%EC%84%A4%EC%B9%98-%EB%B0%A9%EB%B2%95>

- Mac

- <https://velog.io/@kms9887/mysql-%EB%8B%A4%EC%9A%B4%EB%A1%9C%EB%93%9C-workbench>

이제 Local 에 Mysql DB 를 구축해 봅시다!



- 터미널 실행!

- Mysql -V

```
C:\Users\wtetz>mysql -V  
mysql Ver 8.0.28 for Win64 on x86_64 (MySQL Community Server - GPL)
```

- 이게 안 뜨면 시스템 환경 변수 설정
 - <https://dog-developers.tistory.com/21>

이제 Local 에 Mysql DB 를 구축해 봅시다!



- `mysql -u root -p`
- 설정한 비밀번호를 치고 들어가기

```
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 53  
Server version: 8.0.28 MySQL Community Server - GPL  
  
Copyright (c) 2000, 2022, 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> _
```

이제 Local 에 Mysql DB 를 구축해 봅시다!



- show databases;

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mydb      |
| mysql     |
| performance_schema |
| sakila    |
| sys       |
| world     |
+-----+
7 rows in set (0.00 sec)
```

이제 Local 에 Mysql DB 를 구축해 봅시다!



- use sakila;
- show tables;

```
mysql> use sakila;
Database changed
mysql> show tables;
+-----+
| Tables_in_sakila |
+-----+
| actor              |
| actor_info         |
| address            |
| category           |
| city               |
| country            |
| customer           |
| customer_list      |
| film               |
| film_actor         |
| film_category      |
| film_list          |
| film_text          |
| inventory          |
| language           |
| nicer_but_slower_film_list |
| payment            |
| rental             |
| sales_by_film_category |
| sales_by_store     |
+-----+
```



여기서 부터는 자유롭게 Query 를 사용!

- 다만 아무래도 CLI 는 불편하기 때문에 GUI 를 사용하는 편이 좋겠죠?
- 명령어가 익숙하면 상관이 없지만, 그렇지 않다면 아무래도 Workbench 를 사용하는 편이 좋습니다!
- MySQL Workbench 실행



MySQL Workbench

Workbench 사용!



- 먼저 schema 선택 하기

The screenshot shows the MySQL Workbench interface. On the left, the 'Navigator' pane is open, displaying a tree view of database objects. The 'SCHEMAS' section is expanded, showing a list of databases. The 'answer' database is selected. Below the Navigator, the 'Administration' and 'Schemas' tabs are visible, with 'Schemas' being the active tab. A red arrow points to the 'answer' table in the Schemas list.

Query 1

```
SELECT * FROM mydb.answer
```

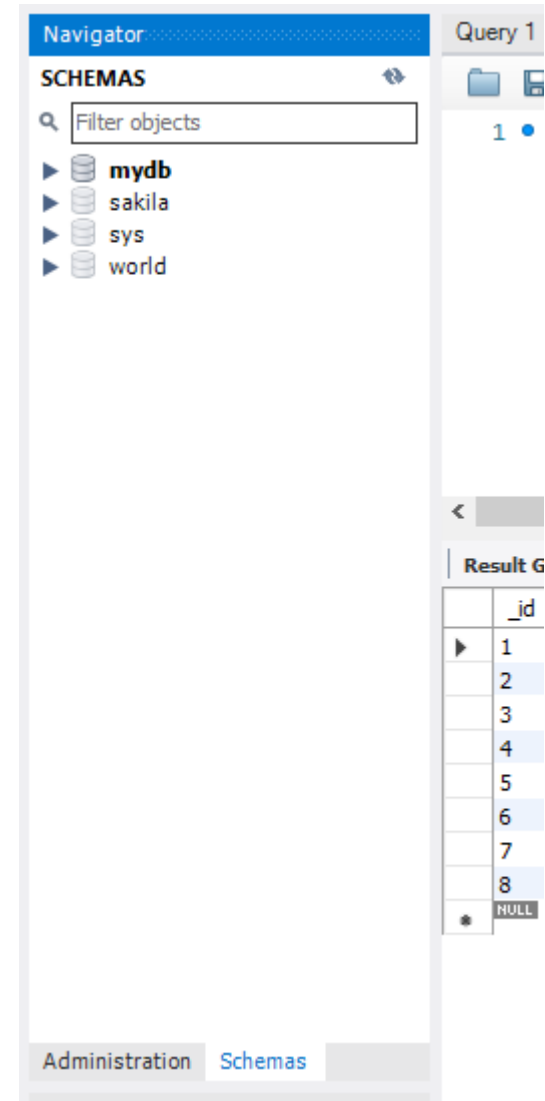
_id	question_id	answer_text
1	0	그런 모임을 왜 이제서
2	0	1년 전에 알려줬어도
3	1	원소리여, 그냥 하던
4	1	오호? 그런게 있어? 들
5	2	무슨 버그가 발생한 거
6	2	아... 내일도 야근 각
7	3	일단 빠르게 개발 완
8	3	그거 내일 아침에 와
*	NULL	NULL

Table: **answer**

Columns: **_id** int AI PK

Schema

- 만들어진 DB 를 확인 할 수 있습니다!



Schema

- 각각의 DB 에 있는 테이블과 데이터 확인 가능

Query 1 question answer answer visitor answer

Limit to 1000 rows

1 • `SELECT * FROM sakila.actor;`

Filter objects

mydb

sakila

Tables

actor

Select Rows - Limit 1000

Table Inspector

Copy to Clipboard

Table Data Export Wizard

Table Data Import Wizard

Send to SQL Editor

Create Table...

Create Table Like...

Alter Table...

Table Maintenance...

Drop Table...

Truncate Table...

Search Table Data...

Refresh All

Filter Rows: Edit:

	first_name	last_name	last_update
1	ENVELOPE	GUINNESS	2006-02-15 04:34:33
2	WICK	WAHLBERG	2006-02-15 04:34:33
3	D	CHASE	2006-02-15 04:34:33
4	ENNIFER	DAVIS	2006-02-15 04:34:33
5	JOHNNY	LOLLOBRIGIDA	2006-02-15 04:34:33
6	ETTE	NICHOLSON	2006-02-15 04:34:33
7	GRACE	MOSTEL	2006-02-15 04:34:33
8	MATTHEW	JOHANSSON	2006-02-15 04:34:33
9	OE	SWANK	2006-02-15 04:34:33
10	CHRISTIAN	GABLE	2006-02-15 04:34:33
11	ZERO	CAGE	2006-02-15 04:34:33
12	KARL	BERRY	2006-02-15 04:34:33
13	UMA	WOOD	2006-02-15 04:34:33
14	VIVIEN	BERGEN	2006-02-15 04:34:33
15	CUBA	OLIVIER	2006-02-15 04:34:33
16	FRED	COSTNER	2006-02-15 04:34:33
17	HELEN	VOIGHT	2006-02-15 04:34:33
18	DAN	TORN	2006-02-15 04:34:33
19	BOB	FAWCETT	2006-02-15 04:34:33
20	LUCILLE	TRACY	2006-02-15 04:34:33
21	KIRSTEN	PALTROW	2006-02-15 04:34:33
22	ELVIS	MARX	2006-02-15 04:34:33
23	SANDRA	KILMER	2006-02-15 04:34:33

Administration Schemas

Information

Table: actor

Columns:

actor_id	smallint UNSIGNED AUTO INCREMENTAL PRIMARY KEY
first_name	varchar(45)
last_name	varchar(45)
last_update	timestamp

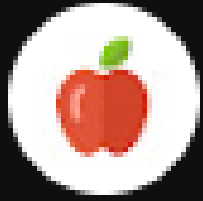


정규화



정규화?

- DB 설계에 있어서 중복을 최소화 하기 위해 데이터를 구조화 하는 과정
- 크고 조직화 되지 않은 테이블 → 작고, 잘 조직된 테이블로 변경
- 이렇게 작성을 해야만, 데이터 추가 및 삭제 시에 이상 현상(Abnormal)을 예방 할 수 있습니다!



코딩애플
구독자 14.9만명

<https://www.youtube.com/watch?v=Y1FbowQRcml>



제 1 정규형(1NF)

- 하나의 컬럼은 반드시 하나의 속성만을 가져야 하는 법칙

회원번호	회원이름	프로그램
101	강호동	스쿼시초급
102	손흥민	헬스
103	김민수	헬스, <u>골프초급</u>

회원번호	회원이름	프로그램
101	강호동	스쿼시초급
102	손흥민	헬스
103	김민수	헬스
103	김민수	골프초급



제 2 정규형(2NF)

- 모든 컬럼에 대한 부분 종속이 없어야 한다 → 현 테이블의 주제(?)와 필요 없는 친구는 빼준다

회원번호	회원이름	프로그램	가격	납부여부
101	강호동	스쿼시초급	5000	0
102	손흥민	헬스	6000	1
103	김민수	헬스	6000	1
103	김민수	골프초급	8000	0

수강등록현황 table				프로그램 table	
회원번호	회원이름	프로그램	납부여부	프로그램	가격
101	강호동	스쿼시초급	0	스쿼시초급	5000
102	손흥민	헬스	1	헬스	6000
103	김민수	헬스	1	골프초급	8000
103	김민수	골프초급	0		



제 3 정규형(3NF)

- 이행 종속성($A = B$, $B = C$ 여서 $A = C$ 인 경우)이 없어야 한다
- 일반 컬럼에만 종속된 컬럼은 다른 테이블로 빼기

프로그램	가격	강사	출신대학
스쿼시	5000	김을용	서울대
헬스	6000	박덕팔	연세대
골프	8000	이상구	고려대
골프중급	9000	이상구	고려대
개인피티	6000	박덕팔	연세대

프로그램	가격	강사	강사	출신대학
스쿼시	5000	김을용	김을용	서울대
헬스	6000	박덕팔	박덕팔	연세대
골프	8000	이상구	이상구	고려대
골프중급	9000	이상구	이상구	고려대
개인피티	6000	박덕팔	박덕팔	고려대



Foreign Key

(외래 키)



외래 키(Foreign Key)

- 정규화를 하게 되면 테이블은 최소한의 단위로 쪼개지게 됩니다!
- 하지만 데이터를 불러 들일 때에는 한꺼번에 많은 값을 가지는 테이블을 JOIN 해서 가지고 와야 하는 경우가 많습니다.
- 앞서 배운 JOIN 을 사용하여 테이블을 합치게 되는데요. 보통 기준이 되는 값을 통해서 테이블을 합쳐 주게 됩니다! → 이 때 기준이 되는 값(서로 공유하고 있는 값)을 외래 키라고 합니다!



기본키(Primary Key)

외래키(Foreign Key)

기본키(Primary Key)

선수번호	이름	팀 코드	포지션	등번호	키
1	김남일	K03	DF	33	177
2	박지성	K07	MF	7	178
3	이영표	K02	MF	22	176

〈선수 테이블〉

팀 코드	팀 명	연고지
K03	스틸러스	포항
K07	드래곤즈	전남
K02	블루윙즈	수원

〈구단 테이블〉

선수번호	이름	팀 코드	포지션	등번호	키	팀 명	연고지
1	김남일	K03	DF	33	177	스틸러스	포항
2	박지성	K07	MF	7	178	드래곤즈	전남
3	이영표	K02	MF	22	176	블루윙즈	수원



수업을 위한 DB 만들기

DB 생성!



- `CREATE SCHEMA `mydb` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;`

```
Query 1 x
1 CREATE SCHEMA `myDB` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;
2
```

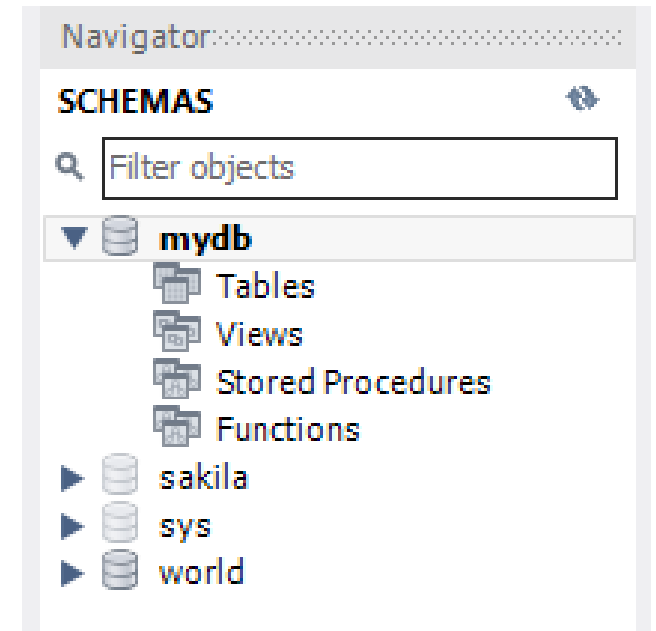




TABLE 생성!



Table 이름 짓기 규칙! (필수 X)

- 대문자 사용하기!
- 단어와 단어 사이는 _ 로 구분
- 누구나 컬럼명만 봐도 해당 컬럼이 어떤 데이터인지 알 수 있게 이름 짓기
- Primary Key 역할을 하는 단어가 최우선으로 온다
- Primary Key 컬럼은 뒤에 _PK 를 추가
- Foreign Key 컬럼은 뒤에 _FK 를 추가



3. 테이블 생성시 제약 넣기

```
CREATE TABLE people (  
  person_id INT AUTO_INCREMENT PRIMARY KEY,  
  person_name VARCHAR(10) NOT NULL,  
  nickname VARCHAR(10) UNIQUE NOT NULL,  
  age TINYINT UNSIGNED,  
  is_married TINYINT DEFAULT 0  
);
```

제약	설명
AUTO_INCREMENT	새 행 생성시마다 자동으로 1씩 증가
PRIMARY KEY	중복 입력 불가, NULL(빈 값) 불가
UNIQUE	중복 입력 불가
NOT NULL	NULL(빈 값) 입력 불가
UNSIGNED	(숫자일시) 양수만 가능
DEFAULT	값 입력이 없을 시 기본값

DB 의 TABLE 생성!



```
Query 1 x
[Icons: Folder, Save, Run, Undo, Redo, Stop, Refresh, Limit to 1000 rows, Star, Erase, Zoom In, Zoom Out, Full Screen, Exit]
1 CREATE TABLE user (
2     `ID_PK` INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
3     `NAME` VARCHAR(100) NOT NULL,
4     `EMAIL` VARCHAR(100) NOT NULL UNIQUE,
5     `PASSWORD` VARCHAR(100) NOT NULL,
6     `ADDRESS` VARCHAR(100) NOT NULL,
7     `AGE` TINYINT UNSIGNED,
8     `MEMBERSHIP` TINYINT DEFAULT 0,
9     `REGISTER_DATE` DATETIME DEFAULT CURRENT_TIMESTAMP,
10    `UPDATE_DATE` DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
11 );
12
```



DATA 삽입하기



생성한 TABLE 에 DATA 삽입!

```
INSERT INTO user (NAME, EMAIL, PASSWORD, ADDRESS, AGE)  
VALUES ('이효석', 'tetz@spreatics', '1324', '서울 서대문구', 38);
```

[illegible]

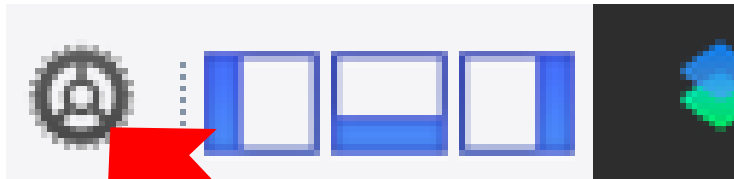


DATA 수정 및 삭제

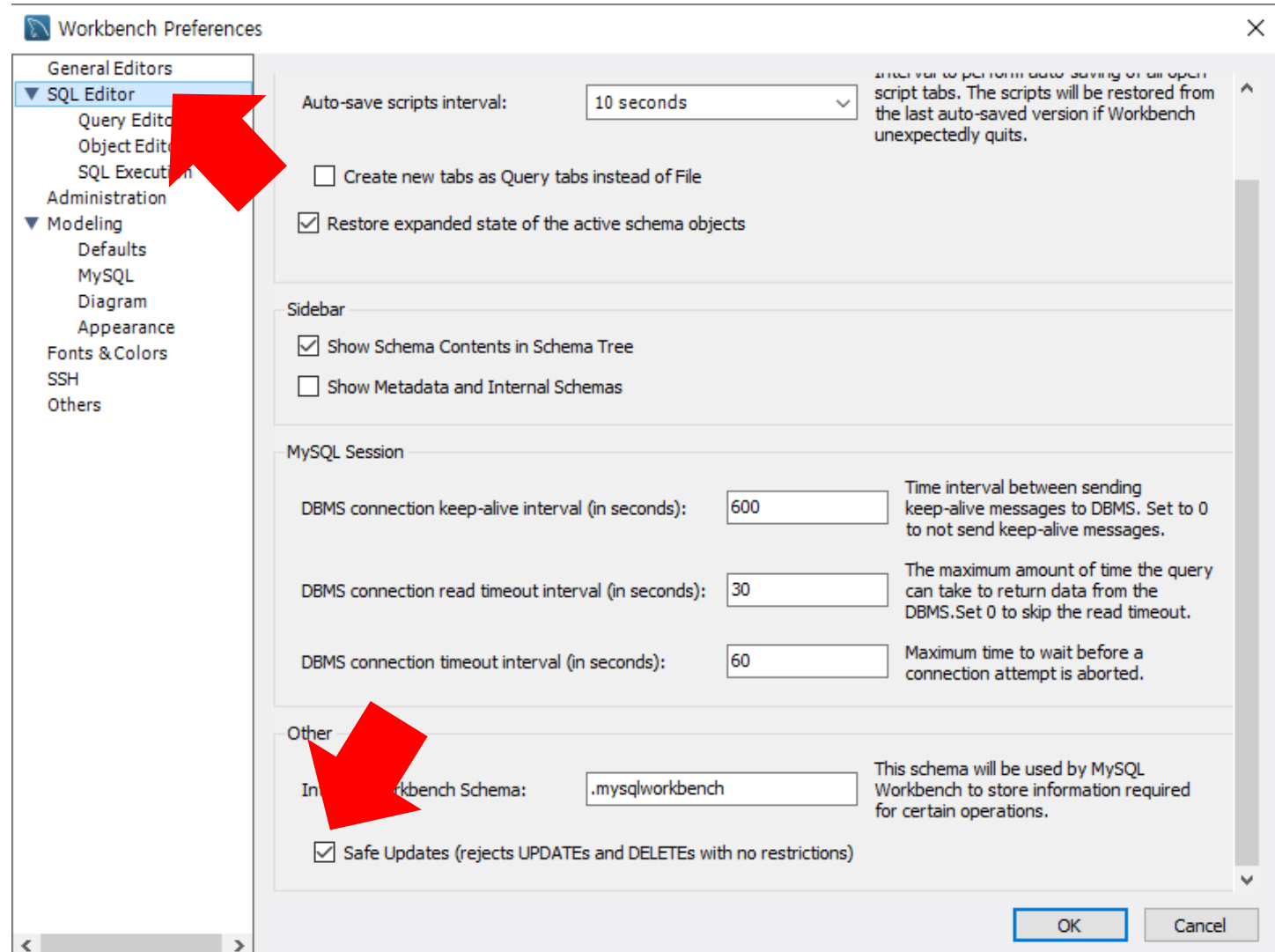


먼저 설정 변경이 필요합니다!

- 환경 설정 > SQL Editor



- 체크 박스를 해제





DATA 삭제



- ```
17 ● DELETE FROM user WHERE ID_PK = 3;
```

[illegible]



# DATA 수정



- ```
18 • UPDATE user SET AGE = AGE + 1 WHERE ID_PK = 1;
19
```

[illegible]



Table 수정 및 삭제하기



ALTER TABLE - 테이블 변경

```
-- 테이블명 변경
ALTER TABLE people RENAME TO friends,
-- 컬럼 자료형 변경
CHANGE COLUMN person_id person_id TINYINT,
-- 컬럼명 변경
CHANGE COLUMN person_name person_nickname VARCHAR(10),
-- 컬럼 삭제
DROP COLUMN birthday,
-- 컬럼 추가
ADD COLUMN is_married TINYINT AFTER age;
```

DROP TABLE - 테이블 삭제

```
DROP TABLE friends;
```



WorkBench 로
수행하기!



TABLE 생성!



Navigator

SCHEMAS

Filter objects

- mydb
 - Tables
 - answer
 - explanation
 - new_table
 - Columns
 - Indexes
 - Foreign Keys
 - Triggers
 - question
 - timestamps
 - visitor
 - Views
 - Stored Procedures
 - Functions
- sakila
- sys
- world

Administration Schemas

Information

Schema: mydb

question answer answer visitor answer actor new_table timestamps new_table - Table x



Table Name: new_table

Schema: mydb

Charset/Collation: Default Charset

Default Collation

Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Column Name:

Data Type:

Charset/Collation:

Default:

Comments:

Storage:

☐ Virtual

☐ Stored

☐ Primary Key

☐ Not Null

☐ Unique

☐ Binary

☐ Unsigned

☐ Zero Fill

☐ Auto Increment

☐ Generated

Columns

Indexes

Foreign Keys

Triggers

Partitioning

Options

Apply

Revert





TABLE 수정!



Navigator

SCHEMAS

Filter objects

mydb

Tables

- answer
- explain
- new_t
- Co
- In
- For
- Tri
- questi
- timest
- visitor
- Views
- Stored Pro
- Functions

sakila

sys

world

Administration

Sch

Information

question

answer

ansi

Table Name

Charset/Collatio

Comments:

Select Rows - Limit 1000

Table Inspector

Copy to Clipboard

Table Data Export Wizard

Table Data Import Wizard

Send to SQL Editor

Create Table...

Create Table Like...

Alter Table...

Table Maintenance...

Drop Table...

Truncate Table...

Search Table Data...

Refresh All



Data 수정 및 삭제



Result Grid									
Filter Rows: <input type="text"/>									
Edit:									
Export/Import:									
Wrap Cell Content: <input type="checkbox"/>									
	ID_PK	NAME	EMAIL	PASSWORD	ADDRESS	AGE	MEMBERSHIP	REGISTER_DATE	UPDATE_DATE
	1	이호	tetz@spreatics	1324	서울 서대문구	39	0	2022-11-23 17:03:19	2022-11-23 17:07:25
	5	이호석	tett@spreatics	1324	서울 서대문구	38	0	2022-11-23 17:05:23	2022-11-23 17:05:23
	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Result Grid									
Filter Rows: <input type="text"/>									
Edit:									
Export/Import:									
Wrap Cell Content: <input type="checkbox"/>									
	ID_PK	NAME	EMAIL	PASSWORD	ADDRESS	AGE	MEMBERSHIP	REGISTER_DATE	UPDATE_DATE
	1	이호	tetz@spreatics	1324	서울 서대문구	39	0	2022-11-23 17:03:19	2022-11-23 17:07:25
	5	이호석	tett@spreatics	1324	서울 서대문구	38	0	2022-11-23 17:05:23	2022-11-23 17:05:23
	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Result Grid

user 7 x

Apply

Revert



Review SQL Script

Apply SQL Script

Review the SQL Script to be Applied on the Database

```
1 UPDATE `mydb`.`user` SET `NAME` = '이효' WHERE (`ID_PK` = '1');  
2
```



Back

Apply

Cancel



Review SQL Script

Apply SQL Script

Applying SQL script to the database

The following tasks will now be executed. Please monitor the execution.
Press Show Logs to see the execution logs.

☒ Execute SQL Statements

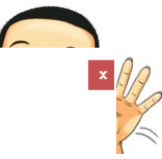
SQL script was successfully applied to the database.

Show Logs

Back

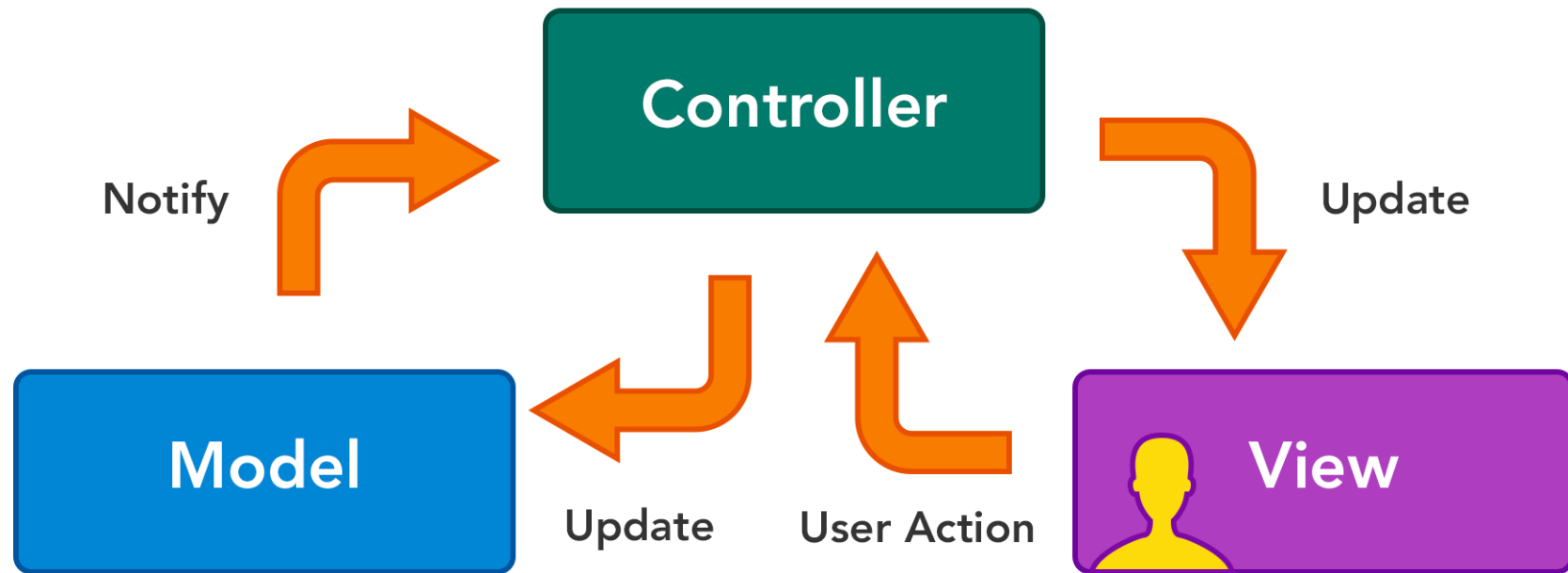
Finish

Cancel





MVC 패턴





MVC?

- Model View Controller 의 약자입니다!
- 웹 설계에 대한 구조입니다!
- 웹 설계를 3가지 단계로 구분하여 구조적 장점을 가지기 위한 수단 입니다



Model

- 어플리케이션의 데이터를 처리하는 역할을 합니다
- 사용자가 볼 수 없는 곳에서 DB로부터 데이터를 읽고, 삽입하고, 수정하고, 삭제하는 역할을 합니다
- Controller 와 소통하며, View 와는 소통하지 않습니다
- 우리로 따지면 DB에서 데이터를 읽고, 삽입하고, 수정하는 역할을 합니다
→ 단, 이 모든 기능을 하나의 파일에 묶어서 처리하는 방식이 되겠죠!



Controller

- Model 과 View 의 상호작용을 컨트롤 합니다
- Model 로 부터 전달 받은 데이터를 가공하여 View 에게 전달합니다
- 또한 View 부터 들어온 사용자 요청을 Model 에 전달 합니다!
- 사용자 요청은 전부 Controller 가 처리 합니다!
 - View 에서 삭제 버튼을 누름 → 해당 내용을 Controller 에 전달 → Controller 는 해당 내용을 Model 이 알아 들을 수 있는 언어로 변경하여 Model 에 전달 → DB에서 해당 내용 삭제 → 해당 내용은 다시 Controller 를 통해 View 전달 → 사용자가 보는 화면이 변경



View

- Model 부터 받은 정보를 Controller 가 받아서 전달하면 그려주는 역할을 합니다
- 사용자가 보는 화면을 담당
- 사용자의 요청을 Controller 로 전달 합니다



MVC의 장단점

- 장점

- 역할이 명확히 구분 되어 있어서 유지 보수가 쉽다
- 기능 추가 및 확장, 유지 보수가 쉽다
- 프론트 백이 서로 독립적으로 개발 이 가능

- 단점

- MVC 구조에 대한 이해를 바탕으로 설계가 필요하다
- MVC 를 명확하게 나누기가 어려운 부분이 있어서 구조 상으로 복잡해 지는 경우가 많다



MVC의 현재

- 장점

- 역할이 명확히 구분 되어 있어서 유지 보수가 쉽다
- 기능 추가 및 확장이 쉽다
- 서로간의 협업에 좋다

- 단점

- MVC 구조에 대한 이해를 바탕으로 설계가 필요하다
- MVC 를 명확하게 나누기가 어려운 부분이 있어서 구조 상으로 복잡해 지는 경우가 많다



A screenshot from a live performance by the K-pop group Boys' Generation. A member is shown from the chest up, wearing a shiny gold jacket over a gold shirt. He is holding a microphone in his left hand and has his right hand near his ear. The background is a bright blue stage light. In the top left corner, there is a logo for 'The Crown' (트로니) and a text box that reads '박상철 X 벨리댄스 <무조건>'. In the top right corner, the 'MBN' logo is visible. At the bottom, there is a large stylized text overlay '무조건~ 무조건이야~' and a news ticker at the very bottom with the text '보이스퀸 공연 안내' and 'MBN 당신이 바로! (보이스퀸) 전국투어 콘서트가 전'.



DB 통신용 서버 구축하기



DB 통신을 하는 Back 서버를 구축

- 이제 DB 통신을 하는 Backend 서버를 구축하고 해당 서버로 부터 데이터를 받아 봅시다!
- 백엔드 폴더에 DB를 컨트롤 하는 controllers 폴더 생성

```
> controllers  
> node_modules  
> public  
> routes  
> views  
⊙ .eslintrc.js  
◆ .gitattributes
```

MySQL 모듈 설치



- 먼저 모듈부터 설치 합시다!
- Npm i mysql



MySQL 서버 통신용 모듈 작성

- dbConnect.js 라는 DB 통신용 모듈을 작성해 봅시다!

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

const connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '비밀번호',
  port: '3306',
  database: 'mydb',
});
```

```
connection.connect();
```

```
module.exports = connection;
```

controllers/dbConnect.js



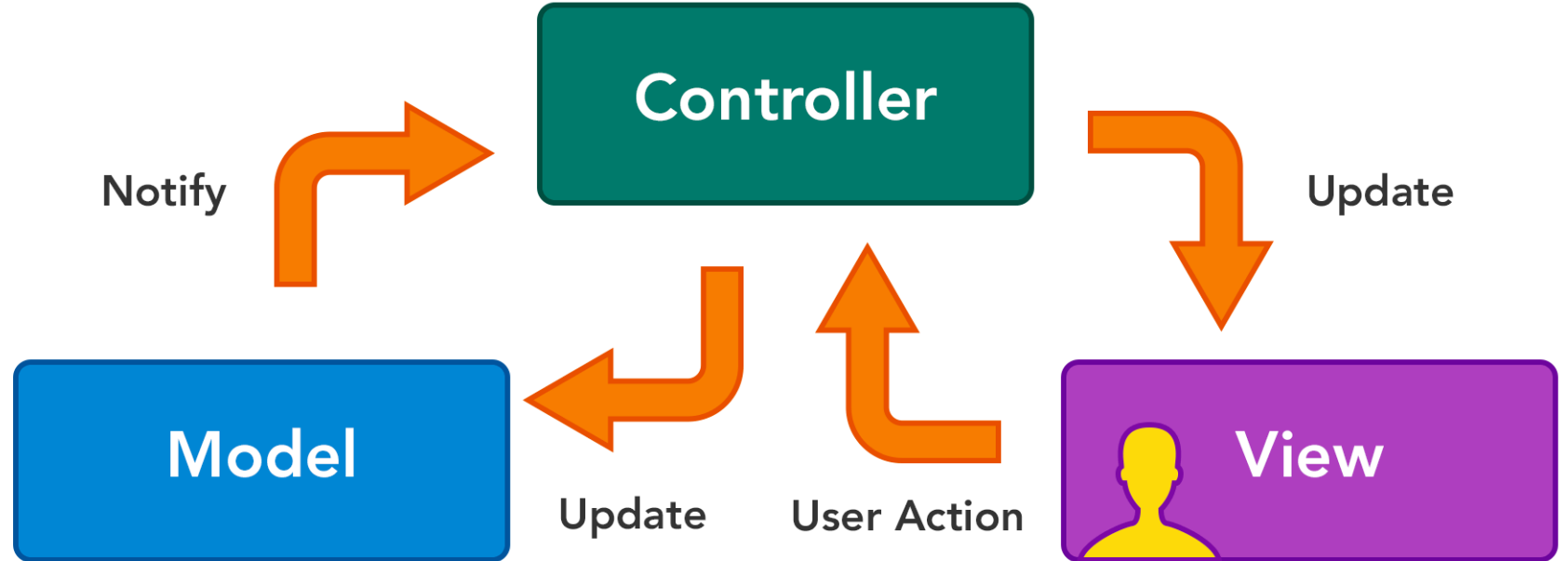
DB 통신용

컨트롤러 작성



DB 통신 컨트롤러 작성

- DB와 통신을 해서 우리가 원하는 데이터를 만들어 주는 Controller 를 작성해 봅시다





DB 통신 컨트롤러 작성

- Constrollers 폴더에 userController.js 파일 생성



```
const connection = require('./dbConnect');

const db = {
  getUsers: (cb) => {
    connection.query('SELECT * FROM mydb.user;', (err, data) => {
      if (err) throw err;
      console.log(data);
      cb(data);
    });
  },
};
```

```
module.exports = db;
```

contorollers/dataController.js



DB 통신용 라우터 생성



DB 통신 라우터 생성

- Routes 폴더에 data.js 생성

```
const express = require('express');
const db = require('../controllers/userController');

const router = express.Router();

router.get('/', (req, res) => {
  db.getUsers((data) => {
    res.send(data);
  });
});

module.exports = router;
```

routes/data.js



메인 서버에 라우터 등록

```
const indexRouter = require('./routes');
const userRouter = require('./routes/users');
const postRouter = require('./routes/posts');
const boardRouter = require('./routes/board');
const dataRouter = require('./routes/data');

app.set('view engine', 'ejs');

app.use('/', indexRouter);
app.use('/users', userRouter);
app.use('/posts', postRouter);
app.use('/board', boardRouter);
app.use('/data', dataRouter);
```

App.js



POSTMAN 으로

테스트!

localhost:4000/data



GET



localhost:4000/data

```
1  [
2  {
3      "ID_PK": 1,
4      "NAME": "이호",
5      "EMAIL": "tetz@spreatics",
6      "PASSWORD": "1324",
7      "ADDRESS": "서울 서대문구",
8      "AGE": 39,
9      "MEMBERSHIP": 0,
10     "REGISTER_DATE": "2022-11-23T08:03:19.000Z",
11     "UPDATE_DATE": "2022-11-23T08:09:42.000Z"
12 },
13 {
14     "ID_PK": 5,
15     "NAME": "이호석",
16     "EMAIL": "tett@spreatics",
17     "PASSWORD": "1324",
18     "ADDRESS": "서울 서대문구",
19     "AGE": 38,
20     "MEMBERSHIP": 0,
21     "REGISTER_DATE": "2022-11-23T08:05:23.000Z",
22     "UPDATE_DATE": "2022-11-23T08:05:23.000Z"
23 }
```




통신 에러 시 대처 방법



```
D:\git\developer-mbti\node_modules\mysql\lib\protocol\Parser.js:437
  throw err; // Rethrow non-MySQL errors
  ^
```

```
Error: ER_NOT_SUPPORTED_AUTH_MODE: Client does not support authentication protocol requested by server;
consider upgrading MySQL client
    at Handshake.Sequence._packetToError (D:\git\developer-mbti\node_modules\mysql\lib\protocol\sequences\Sequence.js:47:14)
    at Handshake.ErrorPacket (D:\git\developer-mbti\node_modules\mysql\lib\protocol\sequences\Handshake.js:123:18)
    at Protocol.parsePacket (D:\git\developer-mbti\node_modules\mysql\lib\protocol\Protocol.js:281:23)
```

<https://1mini2.tistory.com/88>

3



조운호

2018.07.13 오후 4:33

<https://stackoverflow.com/questions/50093144/mysql-8-0-client-does-not-support-authentication-protocol-requested-by-server>

ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY '사용할패스워드'

뭔가 이렇게 하니까 됐어요~~

3 • ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY '1234';



게시판에 DB 적용하기



게시판 서비스를 위한 TABLE 생성

```
1 • ○ CREATE TABLE board (  
2     `ID_PK` INT NOT NULL PRIMARY KEY AUTO_INCREMENT,  
3     `TITLE` VARCHAR(100) NOT NULL,  
4     `CONTENT` VARCHAR(300) NOT NULL,  
5     `REGISTER_DATE` DATETIME DEFAULT CURRENT_TIMESTAMP,  
6     `UPDATE_DATE` DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP  
7 ) ;
```



테스트를 위한 기본 데이터 삽입

- 9 • `INSERT INTO board (TITLE, CONTENT) VALUES ('제목1', '테스트 컨텐츠 입니다!');`
- 10 • `INSERT INTO board (TITLE, CONTENT) VALUES ('제목1', '테스트 컨텐츠 입니다!');`

Result Grid

Filter Rows:

Edit:

Export/Import:

	ID_PK	TITLE	CONTENT	REGISTER_DATE	UPDATE_DATE
▶	1	제목1	테스트 컨텐츠 입니다!	2022-11-24 14:27:39	2022-11-24 14:27:39
	2	제목2	테스트 컨텐츠 입니다!	2022-11-24 14:27:41	2022-11-24 14:27:41
✱	NULL	NULL	NULL	NULL	NULL



DB 버전

게시판 작업 시작!



게시판 용 컨트롤러 작성

- Controllers 폴더에 boardController.js 작성

```
const connection = require('./dbConnect');

const db = {
  getAllArticles: (cb) => {
    connection.query('SELECT * FROM mydb.board;', (err, data) => {
      if (err) throw err;
      console.log(data);
      cb(data);
    });
  },
};
```

```
module.exports = db;
```

Controllers/boardController.js



dbBoard.js 라우터 만들기

- routes 폴더에 dbBoard.js 라우터 작성

```
const express = require('express');
const db = require('../controllers/boardController');

const router = express.Router();

router.get('/getAll', (req, res) => {
  db.getAllArticles((data) => {
    res.send(data);
  });
});

module.exports = router;
```

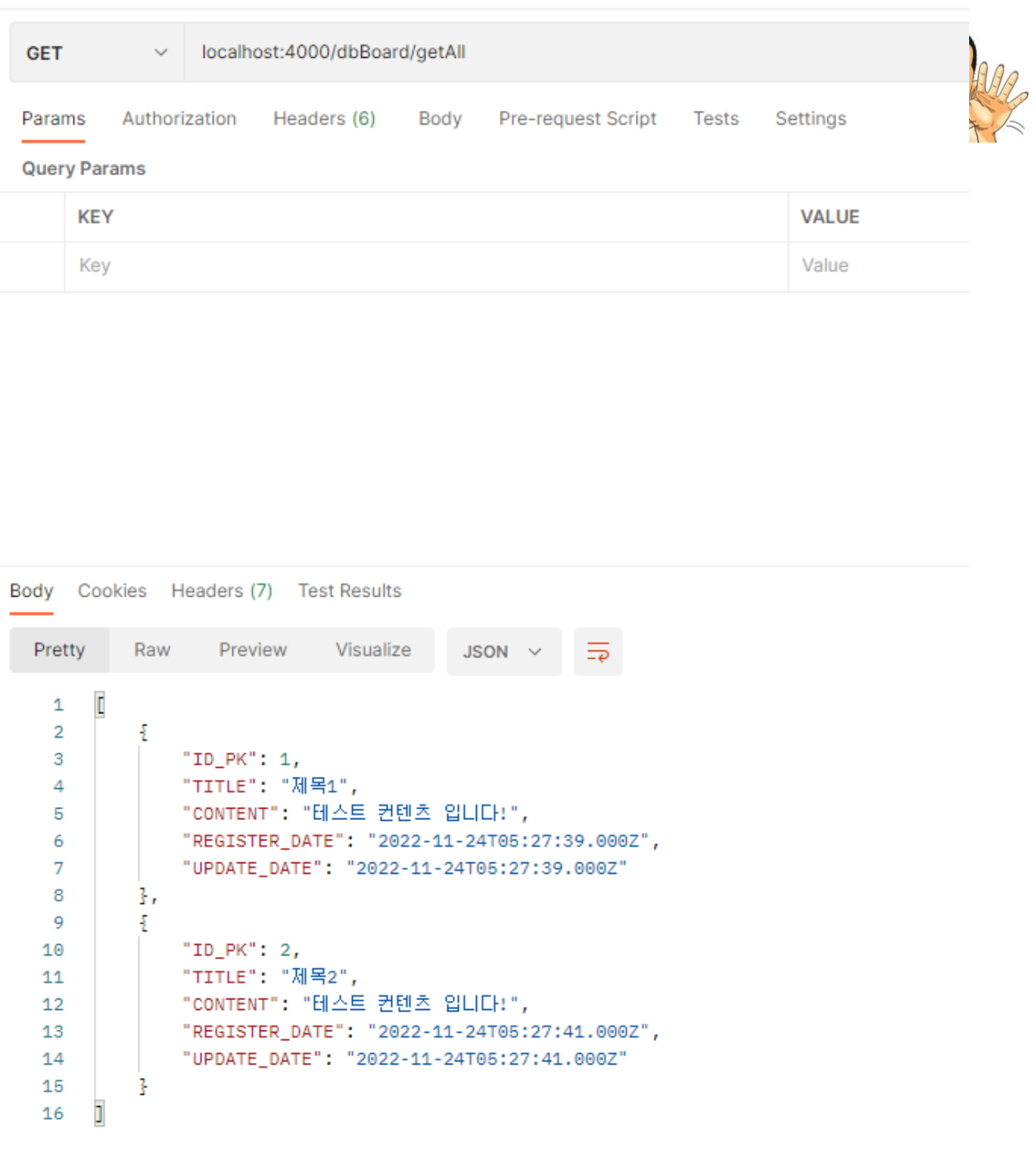
routes/dbBoard.js



메인 서버에 라우터 등록

```
const mainRouter = require('./routes');  
const boardRouter = require('./routes/board');  
const dbBoardRouter = require('./routes/dbBoard');  
  
app.use('/', indexRouter);  
app.use('/board', boardRouter);  
app.use('/dbBoard', dbBoardRouter);
```

포스트 맨으로 테스트!



GET localhost:4000/dbBoard/getAll

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE
Key	Value

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   {
3     "ID_PK": 1,
4     "TITLE": "제목1",
5     "CONTENT": "테스트 컨텐츠 입니다!",
6     "REGISTER_DATE": "2022-11-24T05:27:39.000Z",
7     "UPDATE_DATE": "2022-11-24T05:27:39.000Z"
8   },
9   {
10    "ID_PK": 2,
11    "TITLE": "제목2",
12    "CONTENT": "테스트 컨텐츠 입니다!",
13    "REGISTER_DATE": "2022-11-24T05:27:41.000Z",
14    "UPDATE_DATE": "2022-11-24T05:27:41.000Z"
15  }
16 }
```



dbBoard 용 EJS 파일 생성!

- 기존 board.ejs 파일을 카피해서 쓰기 → db_board.ejs 생성
- 글쓰기 모드로 이동 href 수정

```
<span>현재 등록 글 : &nbsp; <%= articleCounts %></span>  
<a class="btn red" href="/dbBoard/write">글쓰기</a>
```



DB에서 데이터를
받아서 EJS에 출력



기본 페이지 작업!

- MySQL 로 부터 데이터를 받아서 EJS 에 전달하는 라우터 작성

```
router.get('/', (req, res) => {  
  db.getAllArticles((data) => {  
    const ARTICLE = data;  
    const articleCounts = ARTICLE.length;  
    res.render('dbBoard', { ARTICLE, articleCounts });  
  });  
});
```

routes/dbBoard.js



Tetz Board

현재 등록 글 : 2

글쓰기

수정

삭제

수정

삭제





DB로 부터 받아오는 데이터 찍어보기!

- DB에서 데이터가 잘 들어오는지 확인해 봅시다!

```
router.get('/', (req, res) => {  
  db.getAllArticles((data) => {  
    const ARTICLE = data;  
    const articleCounts = ARTICLE.length;  
    console.log(ARTICLE);  
    res.render('dbBoard', { ARTICLE, articleCounts });  
  });  
});
```

routes/dbBoard.js



DB로 부터 받아오는 데이터 찍어보기!

- 아하! 키가 컬럼 명을 그대로 받아오기 때문에 대문자로 변경이 되었군요!

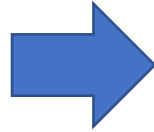
```
[
  RowDataPacket {
    ID_PK: 1,
    TITLE: '제목1',
    CONTENT: '테스트 콘텐츠 입니다!',
    REGISTER_DATE: 2022-11-24T05:27:39.000Z,
    UPDATE_DATE: 2022-11-24T05:27:39.000Z
  },
  RowDataPacket {
    ID_PK: 2,
    TITLE: '제목2',
    CONTENT: '테스트 콘텐츠 입니다!',
    REGISTER_DATE: 2022-11-24T05:27:41.000Z,
    UPDATE_DATE: 2022-11-24T05:27:41.000Z
  }
]
```



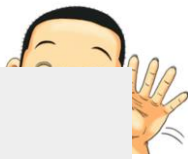
EJS 파일 변경

- EJS 파일을 변경!

```
<div class="title">
  <%= ARTICLE[i].title %>
</div>
<div class="content">
  <%= ARTICLE[i].content %>
</div>
```



```
<div class="title">
  <%= ARTICLE[i].TITLE %>
</div>
<div class="content">
  <%= ARTICLE[i].CONTENT %>
</div>
```



Tetz Board

현재 등록 글 : 2

글쓰기

제목1

테스트 콘텐츠 입니다!

수정

삭제

제목2

테스트 콘텐츠 입니다!

수정

삭제





글 쓰기 기능

구현



글 쓰기용 EJS 파일 생성!

- 기존 board_write.ejs 파일을 재사용 하기! → dbBoard_write.ejs
- Form 태그의 action 주소를 dbBoard 용으로 변경!

```
<form action="/dbBoard/write" method="POST" class="board_form">
  <div class="form_title">
    <h3>제목</h3>
    <input type="text" name="title" required />
```



글 쓰기 모드로 이동하는 주소 라우팅

- GET 방식 dbBoard/write 라는 주소 요청이 들어오면 글 쓰기 페이지로 이동하는 라우터 만들기!

```
router.get('/write', (req, res) => {  
  res.render('dbBoard_write');  
});
```

routes/dbBoard.js



Form 태그 데이터 받기



Action 으로 보낸 주소 라우터 만들기

- 해당 주소 값으로 form 태그 데이터가 잘 들어오는지 확인

```
router.post('/write', (req, res) => {  
  console.log(req.body);  
});
```

routes/dbBoard.js

서버는 4000번에서 실행 중입니다!

```
[Object: null prototype] { title: '11', content: '11' }
```

새로운 글을 DB에 등록하는 컨트롤러 만들기!



- 데이터는 잘 들어오므로 해당 데이터를 DB에 등록하는 컨트롤러 작업!

```
writeArticle: (newArticle, cb) => {  
  connection.query(  
    `INSERT INTO mydb.board (TITLE, CONTENT) VALUES ('${newArticle.title}', '${newArticle.content}')`,  
    (err, data) => {  
      if (err) throw err;  
      cb(data);  
    }  
  );  
},
```

Controllers/boardContoroller.js



컨트롤러를 사용하도록 라우터 수정

```
router.post('/write', (req, res) => {  
  if (req.body.title && req.body.content) {  
    db.writeArticle(req.body, (data) => {  
      console.log(data);  
      if (data.protocol41) {  
        res.redirect('/dbBoard');  
      } else {  
        const err = new Error('글 쓰기 실패');  
        throw err;  
      }  
    });  
  } else {  
    const err = new Error('글 제목 또는 내용이 없습니다!');  
    throw err;  
  }  
});
```

routes/dbBoard.js



제목2

테스트 콘텐츠 입니다!

수정

삭제

aa

aa

수정

삭제

CC

cc

수정

삭제



글 수정하기 기능

구현



글 수정 용 EJS 파일 생성!

- 기존 board_modify.ejs 파일을 재사용! → dbBoard_modify.ejs
- Form 태그의 action 주소를 dbBoard 용으로 변경!
- title 과 content 도 대문자로 변경



```
<form action="/dbBoard/modify/<%= selectedArticle.ID_PK %>" method="POST" class="board_form">
  <div class="form_title">
    <h3>제목</h3>
    <input type="text" name="title" value="<%= selectedArticle.TITLE %>" required />
  </div>
  <div class="form_content">
    <h3>내용</h3>
    <textarea name="content" id="content" cols="30" rows="10" required><%= selectedArticle.CONTENT %>
    </textarea>
  </div>
  <button type="submit">글 수정하기</button>
</form>
```



글 수정 모드로 이동하는 주소 라우팅

- GET 방식 dbBoard/modify 라는 주소 요청이 들어오면 글 쓰기 페이지로 이동하는 라우터 만들기!

```
router.get('/modify/:id', (req, res) => {  
  res.render('dbBoard_modify');  
});
```

routes/dbBoard.js



글 수정 모드로 이동하는 주소 라우팅

- 하지만 아직은 작동을 안 할 겁니다!
- 특정 글의 내용을 전달 해야 하는데, 아직 DB 내에서 특정 게시글을 찾아 주는 컨트롤러 기능이 없습니다!
- 그럼 만들러 가시죠!

ID_PK 값으로 특정 글을 찾는 컨트롤러 만들기



```
getArticle: (id, cb) => {  
  connection.query(  
    `SELECT * FROM mydb.board WHERE ID_PK = ${id};`,  
    (err, data) => {  
      if (err) throw err;  
      cb(data);  
    }  
  );  
},
```

Controllers/boardContoroller.js



글 수정 모드로 이동 라우터 변경!

```
router.get('/modify/:id', (req, res) => {  
  db.getArticle(req.params.id, (data) => {  
    if (data.length > 0) {  
      res.render('dbBoard_modify', { selectedArticle: data[0] });  
    }  
  });  
});  
});
```

routes/dbBoard.js



Write Mode

제목

제목2

내용

테스트 콘텐츠 입니다!

글 수정하기

글 수정 페이지에서 전달 된 값으로 DB 수정!



- ID_PK 값을 받아서 해당 ID를 가지는 DB를 UPDATE 해주면 됩니다!
- 컨트롤러 작업!

```
modifyArticle: (id, modifyArticle, cb) => {  
  connection.query(  
    `UPDATE mydb1.board SET TITLE = '${modifyArticle.title}', CONTENT =  
    '${modifyArticle.content}' WHERE ID_PK = ${id};`,  
    (err, data) => {  
      if (err) throw err;  
      cb(data);  
    })  
  );  
},
```

Controllers/boardContoroller.js



글을 수정하는 라우터 생성!

```
router.post('/modify/:id', (req, res) => {  
  if (req.body.title && req.body.content) {  
    db.modifyArticle(req.params.id, req.body, (data) => {  
      if (data.protocol41) {  
        res.redirect('/dbBoard');  
      } else {  
        const err = new Error('글 수정 실패');  
        throw err;  
      }  
    });  
  } else {  
    const err = new Error('글 제목 또는 내용이 없습니다!');  
    throw err;  
  }  
});
```

routes/dbBoard.js





실습, 삭제 기능 구현하기!

- 삭제 버튼을 누르면 삭제 처리하시면 됩니다!
- 기존 처럼 제목을 이용해서 지우는 것이 아니라 ID_PK 를 사용해서 삭제 처리 해주세요! 😊

