생활코딩

Node.js 노드제이에스 프로그래밍

3주, 4주 수업

2024.09.24

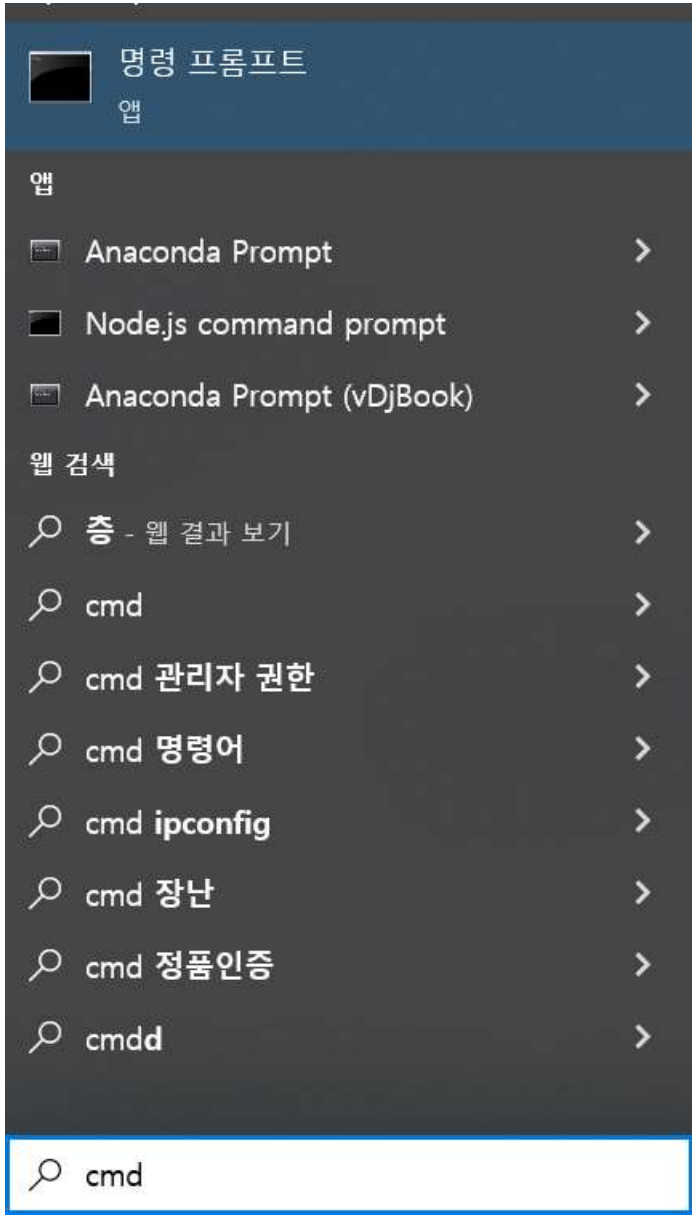
※ 3주차 수업의 범위(웹 DB의 범위)

* mysql에서 데이터 베이스 생성
* 데이터베이스에서 테이블 생성
* node에서 DB로부터 데이터를 가져와서 ejs에 넘겨주는 법
* module화

02 DB 실습 준비

1. mysql 작업 – DB, 테이블생성, 자료 입력

① 명령 프롬프트 들어가기



② 명령 프롬프트 화면에서 mysql.exe 파일이 있는 디렉토리로 들어가기

1. mysql 작업

③ 명령 프롬프트 화면에서 mysql.exe 파일이 있는 디렉토리로 들어가기
– u root : user id인 root로 로그인 하겠다는 의미 , - p : password는 입력 안했기 때문에 질의 함

```
C:₩Program Files₩MySQL₩MySQL Server 8.0₩bin>mysql –u root -p
```

④ password 물어보면 입력하기 : mysql 설치 시 root에 대해 입력했던 password를 입력

```
C:₩Program Files₩MySQL₩MySQL Server 8.0₩bin>mysql –u root –p
Enter password: ****
```

⑤ mysql prompt : GUI인 Workbench를 사용하지 않고도 sql 실행

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

mysql>
```

1. mysql 작업

⑥ db생성하기

```
mysql> create database webdb2024;
```

```
mysql> create database webdb2024;
Query OK, 1 row affected (0.03 sec)

mysql>
```

⑦ 실습을 위한 topic 테이블 생성 예정 ➔ Workbench에서 sqlForTopic.sql 파일 활용해서 생성 10page 참고

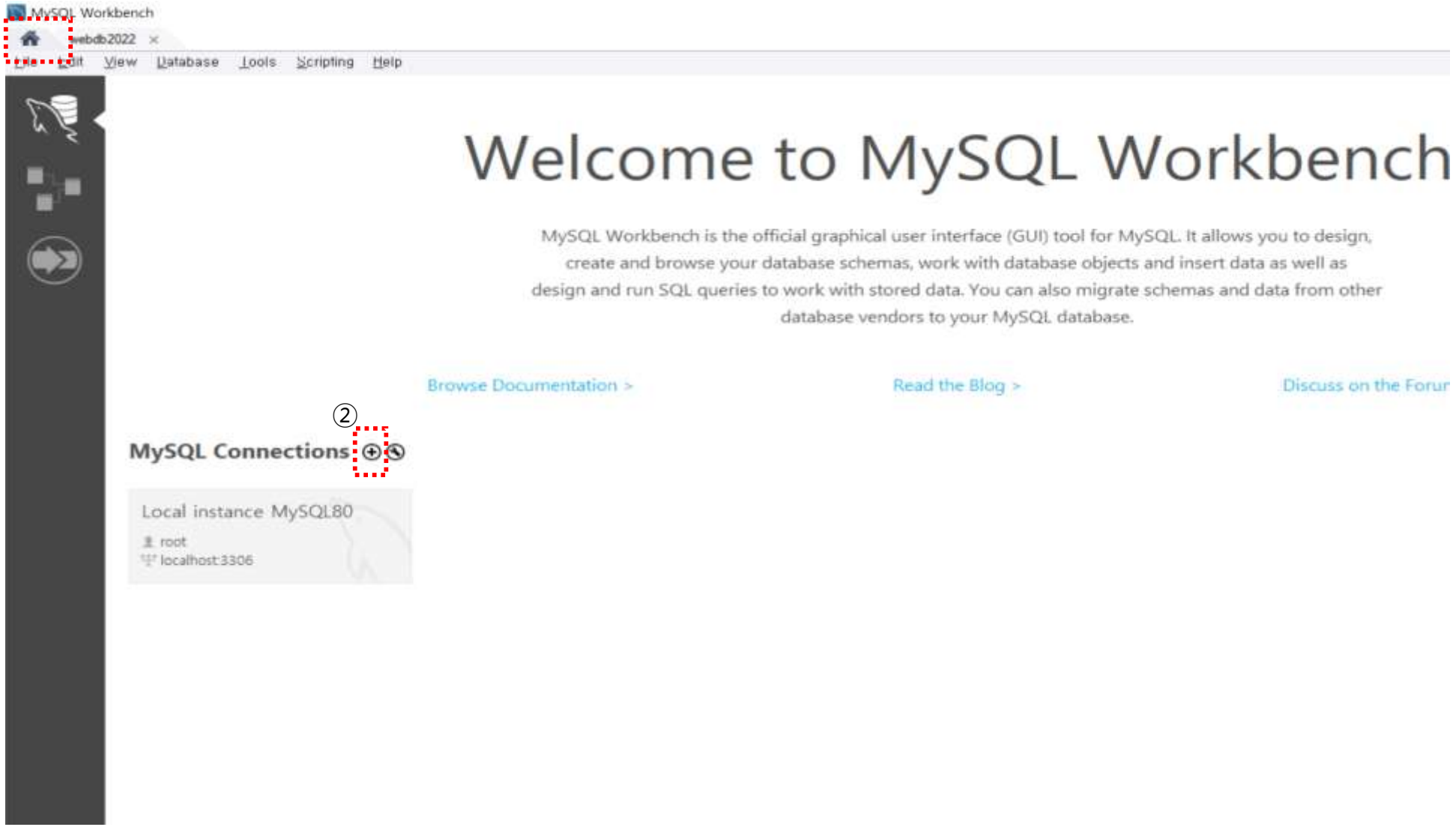
[topic 테이블]

필드명	데이터 타입	null 여부	
id	int	not null	auto_increment
title	varchar(30)	not null	
descrpt	text	not null	
created	datetime	not null	
author_id	int	not null	default

02 DB 실습준비

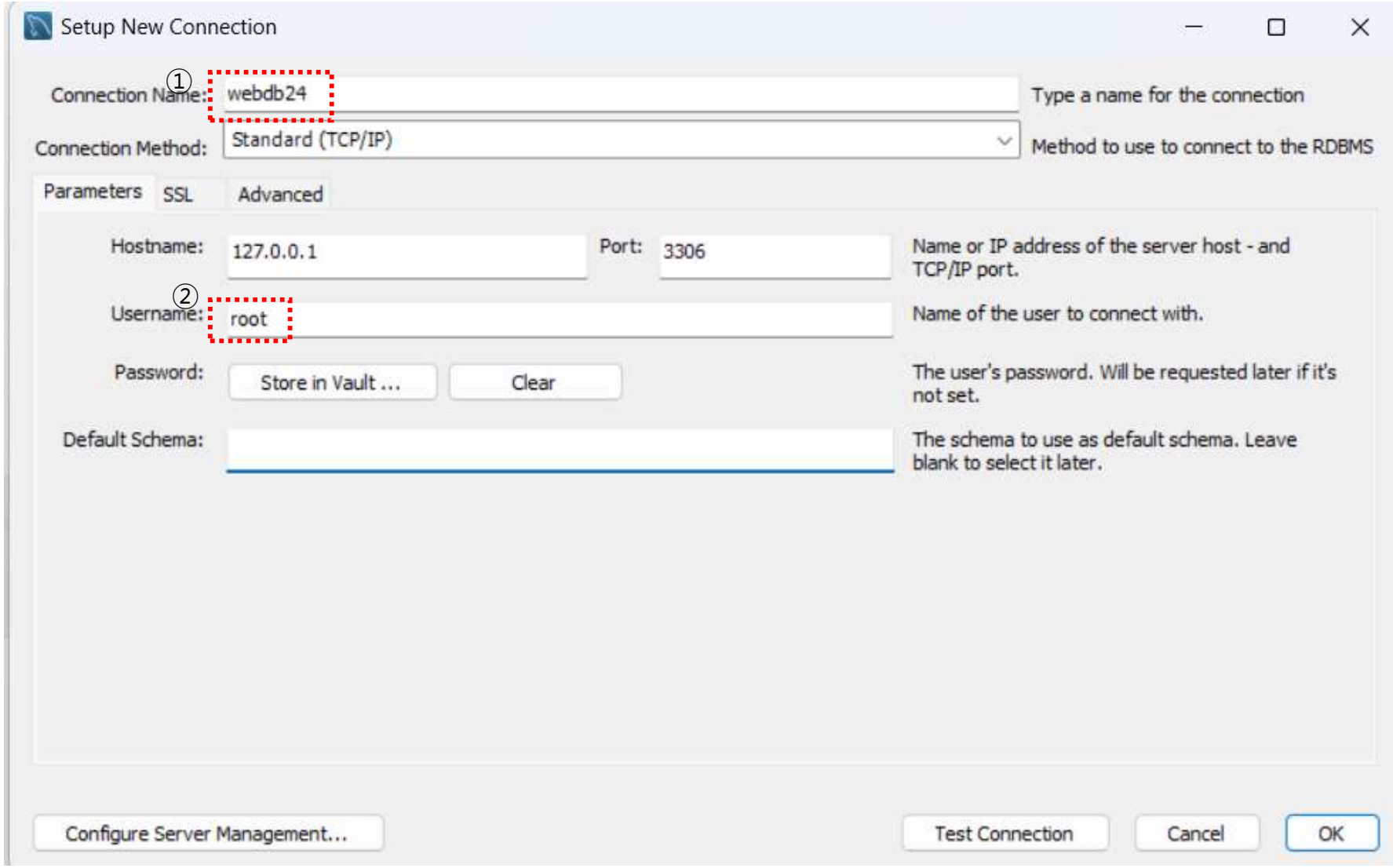
※ mysql Workbench 사용하기 ➔ 새로운 DB 연결자 생성

①

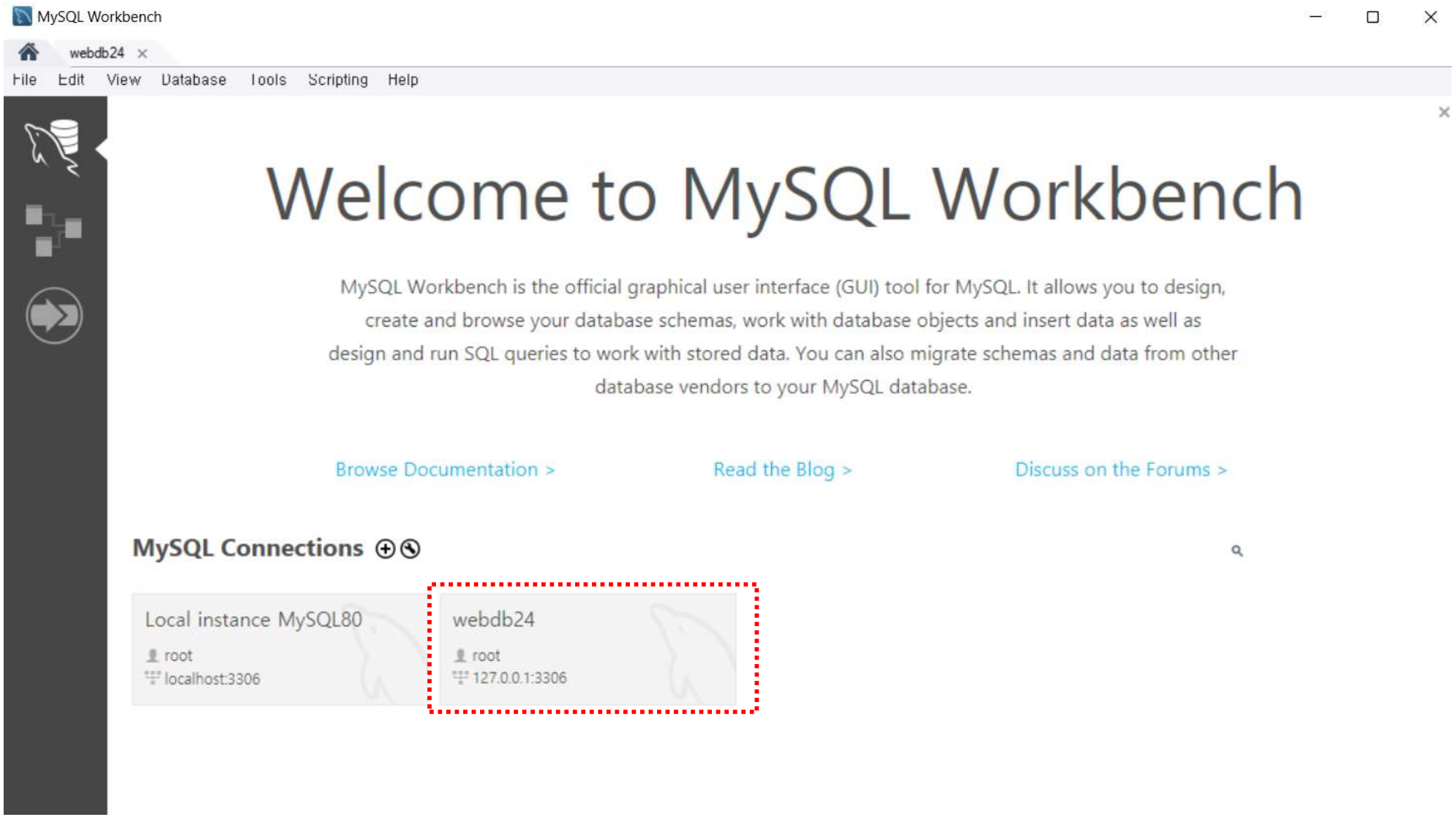


02 DB 실습준비

※ mysql Workbench 사용하기 ➔ 새로운 DB 연결자 생성, 다른 아이디 부여로 DB 접근 제어 가능

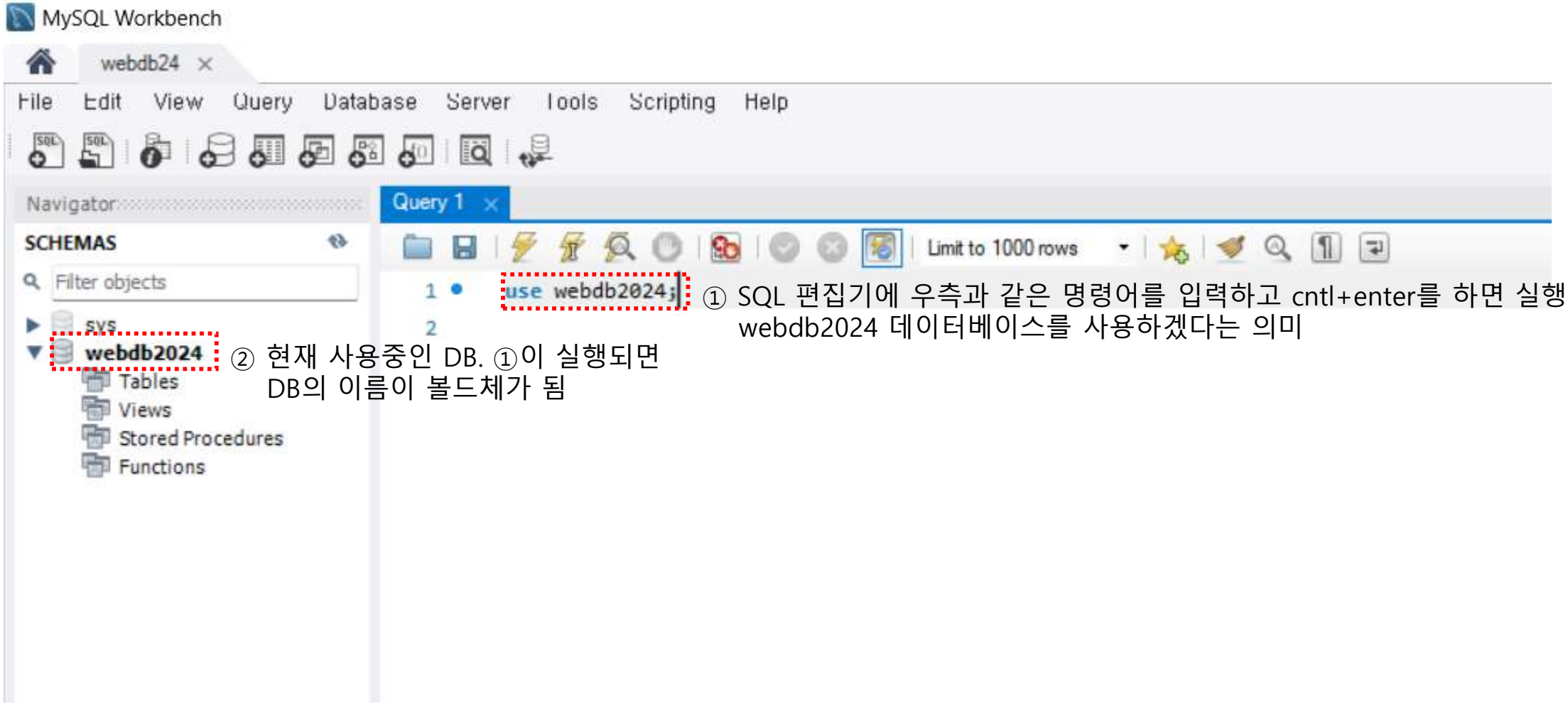


※ mysql Workbench 사용하기



02 DB 실습준비

※ mysql Workbench 사용하기 : root로 로그인 했기 때문에 모든 데이터 베이스가 보임. 콘솔 환경에서 생성한 db webdb2024도 보임



① SQL 편집기에 우측과 같은 명령어를 입력하고 cntl+enter를 하면 실행됨
webdb2024 데이터베이스를 사용하겠다는 의미

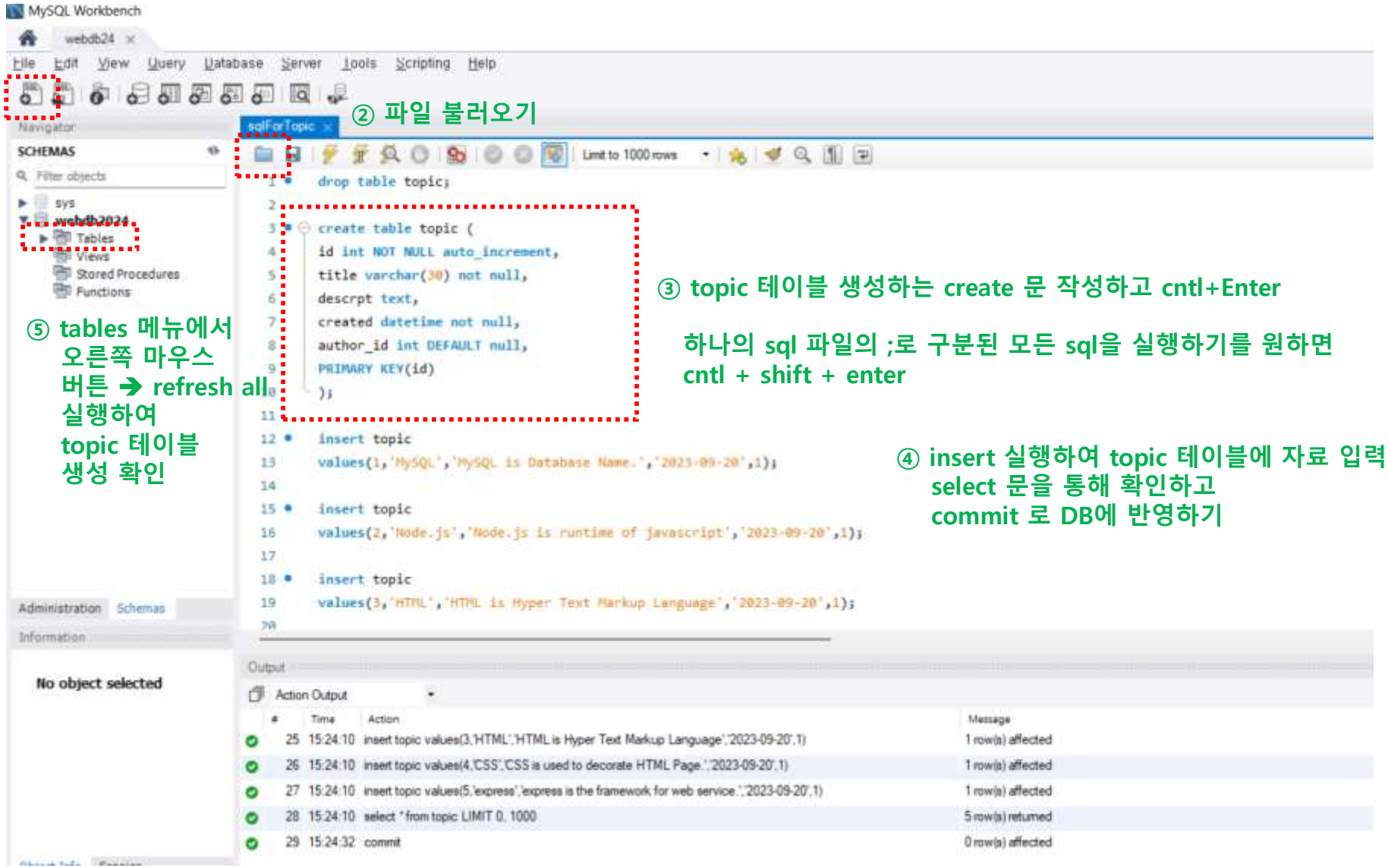
② 현재 사용중인 DB. ①이 실행되면
DB의 이름이 볼드체가 됨

02 DB 실습준비

※ mysql Workbench 사용하기 : sqlForTopic.sql 파일 불러와서 실행시키기

① NEW SQL 편집기 오픈

② 파일 불러오기



⑤ tables 메뉴에서 오른쪽 마우스 버튼 ➔ refresh all 실행하여 topic 테이블 생성 확인

③ topic 테이블 생성하는 create 문 작성하고 cntl+Enter

하나의 sql 파일의 ;로 구분된 모든 sql을 실행하기를 원하면
cntl + shift + enter

④ insert 실행하여 topic 테이블에 자료 입력하고 select 문을 통해 확인하고 commit 로 DB에 반영하기

03 실습준비- 새로운 폴더(홈디렉토리)에서 작업 할때

1. node 작업 ➔ 해당할 경우에만

① 이전 폴더의 package.json과 package-lock.json 파일 새 폴더로 copy

② >npm install 실행 ➔ 이전 폴더에서 사용한 모듈 설치

03 mysql 모듈의 기본 사용법 : mysql과 Nodejs의 연동

2. mysql 모듈 설치

① mysql 모듈 설치

```
C:₩Users₩WBH₩nodejs₩202402>npm install mysql --save
```

② package.json 파일에서 mysql 모듈이 설치되었음을 볼 수 있음

03 mysql 모듈의 기본 사용법 : mysql과 Nodejs의 연동

2. mysql.js 작성

① nodejs 폴더 생성 후 다음과 같이 mysql.js를 작성하여 nodejs폴더에 저장

```
var mysql = require('mysql');
var connection = mysql.createConnection({
    host     : 'localhost',
    user     : 'root',
    password : 'root',
    database : 'webdb2024'
});

connection.connect();

connection.query('SELECT * from topic', (error, results, fields)
=> {
    if (error) {
        console.log(error);
    }
    console.log(results);
});

connection.end();
```

실행결과

```
[
  RowDataPacket {                    results[0]
    id: 1,
    title: 'MySQL',
    description: 'MySQL is...',
    created: 2018-01-01T03:10:11.000Z,
    author_id: 1
  },
  RowDataPacket {                    results[1]
    id: 2,
    title: 'Oracle',
    description: 'Oracle is ...',
    created: 2018-01-03T04:01:10.000Z,
    author_id: 1
  },
  RowDataPacket {                    results[2]
    id: 3,
    title: 'SQL Server',
    description: 'SQL Server is ...',
    created: 2018-01-20T02:01:10.000Z,
    author_id: 2
  },
  RowDataPacket {                    results[3]
    id: 4,
    title: 'PostgreSQL',
    description: 'PostgreSQL is ...',
    created: 2018-01-22T16:03:03.000Z,
    author_id: 3
  },
  RowDataPacket {                    results[4]
    id: 5,                             results[4].id
    title: 'MongoDB',                    results[4].title
    description: 'MongoDB is ...',       results[4].description
    created: 2018-01-30T03:31:03.000Z,
    author_id: 1
  }
]
```

2. mysql.js 작성

WAS 프로그램 or Controller (main.js)

① nodejs와 MySQL 사이에 길을 만들어 준다.

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

```
connection.query();
```

② nodejs가 SQL을 MySQL로 보내면
MySQL은 결과 데이터셋을 nodejs에게 보낸다.

MySQL

2. mysql.js 작성

```
var mysql = require('mysql');
var connection = mysql.createConnection({
    host     : 'localhost',  // 데이터베이스 서버가 있는 주소. Node.js와 같은 서버에 있으므로 localhost
    user     : 'nodejs',     // 데이터베이스에 접근하기 위한 ID
    password : ‘nodejs',     // 데이터베이스에 접근하기 위한 Password
    database : 'webdb2023‘   // 접근하고자 하는 데이터베이스 이름
});

connection.connect();        // 위의 정보를 가지고 DB와 연결

connection.query('SELECT * from topic', (error, results, fields) => {    // ①
    if (error) {
        console.log(error);  // 에러가 발생했을 때 넘겨받는 에러 메시지
    }
    console.log(results);    // results에는 DB에서 하나의 레코드가 객체로 저장, 객체들이 배열로 저장되어 있음
    console.log(fields);     // 콜백함수 세번째 인자에는 필드들의 자세한 정보가 저장
});

connection.end();
```

① query메소드

첫번째 인자 : SQL 문장

실행 실패 - 콘솔에 오류 발생, 실행 성공 – 두번째 인자인 콜백 함수 실행

두번째 인자 : SQL이 실행 완료되면 수행하는 콜백함수 . SQL문의 결과를 콜백함수의 두번째 인자로 넘겨줌.

03 mysql 모듈의 기본 사용법 : mysql과 Nodejs의 연동

3. 오류 발생시

```
  code: 'ER_NOT_SUPPORTED_AUTH_MODE',
  errno: 1251,
  sqlMessage: 'Client does not support authentication protocol requested by server; consider upgrading MySQL client',
  sqlState: '08004',
  fatal: true
}
```

① 오류 발생 이유 2가지

첫째. 클라이언트 프로그램에서 mysql 패스워드 plugin인 "caching_sha2_password " 를 소화하지 못해서 발생

둘째. host가 localhost로 되어 있어 웹 서버와 같은 외부 접근이 허용되지 않음

② mysql로 들어가서 다음 두 명령어 실행

```
mysql> use mysql;
```

mysql 기본 DB인 mysql을 사용

```
mysql> select host, user, plugin from user;
```

Mysql에 있는 user 테이블 select

```
mysql> select user,host, plugin from user;
+------------------+-----------+-----------------------+
| user             | host      | plugin                |
+------------------+-----------+-----------------------+
| mysql.infoschema | localhost | caching_sha2_password |
| mysql.session    | localhost | caching_sha2_password |
| mysql.sys        | localhost | caching_sha2_password |
| root             | localhost | caching_sha2_password |
+------------------+-----------+-----------------------+
4 rows in set (0.00 sec)
```

모든 user들은 localhost 즉 내부 접속만 허용하겠다는 의미
%를 이용해 웹서버와 같은 외부 접근도 허용해 주어야 함.

3. 오류 발생시

③ 첫번째 방법으로 오류 해결 : plugin을 수정

```
mysql> alter user 'root'@'localhost'  identified with mysql_native_password by 'root';
Query OK, 0 rows affected (0.02 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.01 sec)
```

④ 새로운 id 생성하여 plugin도 수정하고 host도 수정

- root를 통상적으로 사용하지 않으므로 새로운 id 생성 : nodejs 암호를 갖는 nodejs id를 생성하고 모든 host가 접근 가능

```
mysql> create user 'nodejs'@'%' identified by 'nodejs';
Query OK, 0 rows affected (0.04 sec)
mysql> select user, host, plugin from user;
```

```
mysql> select user,host, plugin from user;
+------------------+-----------+-----------------------+
| user             | host      | plugin                |
+------------------+-----------+-----------------------+
| nodejs           | %         | caching_sha2_password |
| mysql.infoschema | localhost | caching_sha2_password |
| mysql.session    | localhost | caching_sha2_password |
| mysql.sys        | localhost | caching_sha2_password |
| root             | localhost | mysql_native_password |
+------------------+-----------+-----------------------+
```

03 mysql 모듈의 기본 사용법 : mysql과 Nodejs의 연동

- nodejs에 webdb2024에 대한 모든 권한을 부여하고 반영

```
mysql> grant all privileges on webdb2024.*  TO 'nodejs'@'%';
Query OK, 0 rows affected (0.01 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.01 sec)
```

- nodejs에 대해 plugin을 수정

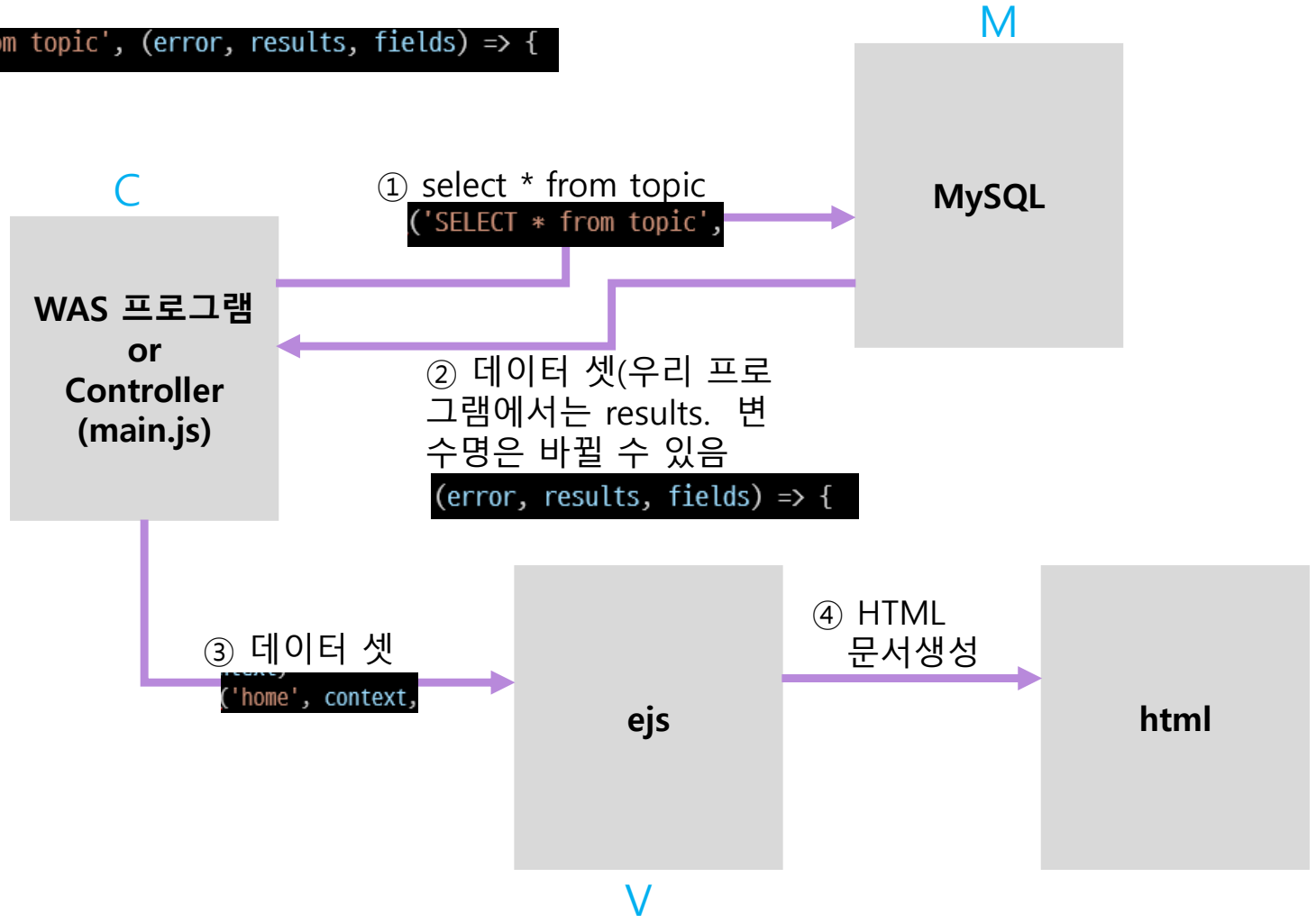
```
mysql> alter user 'nodejs'@'%' identified with mysql_native_password by 'nodejs';
Query OK, 0 rows affected (0.02 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.01 sec)
```

※ mysql Workbench에서도 nodejs로 다시 들어간다.

04 mysql 모듈을 이용한 홈페이지 구현 – topic 테이블의 CRUD

1. topic 테이블 **Read** ➔ **topic** 테이블을 데이터 베이스에서 읽어서 **HTML**에 그 내용을 보내줌

```
connection.query('SELECT * from topic', (error, results, fields) => {
```



04 mysql 모듈을 이용한 홈페이지 구현 – topic 테이블의 R

2. main.js

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

app.set('views',__dirname+'/views');
app.set('view engine','ejs');

var mysql = require('mysql');
var connection =
mysql.createConnection({
  host : 'localhost',
  user : 'nodejs',
  password : 'nodejs',
  database : 'webdb2023'
})
connection.connect();
```

```
app.get('/',(req,res)=>{

  connection.query('SELECT * FROM topic', (error,results)=>{
    var context = {list:results[0],
                   title:'Welcome'};
    console.log(context)
    res.render('home', context, (err,html)=>{
      res.end(html)  })
  });
  connection.end();

})

app.get('/favicon.ico',(req, res)=>res.writeHead(404));
app.listen(3000, () => console.log('Example app listening on port 3000')) ;
```

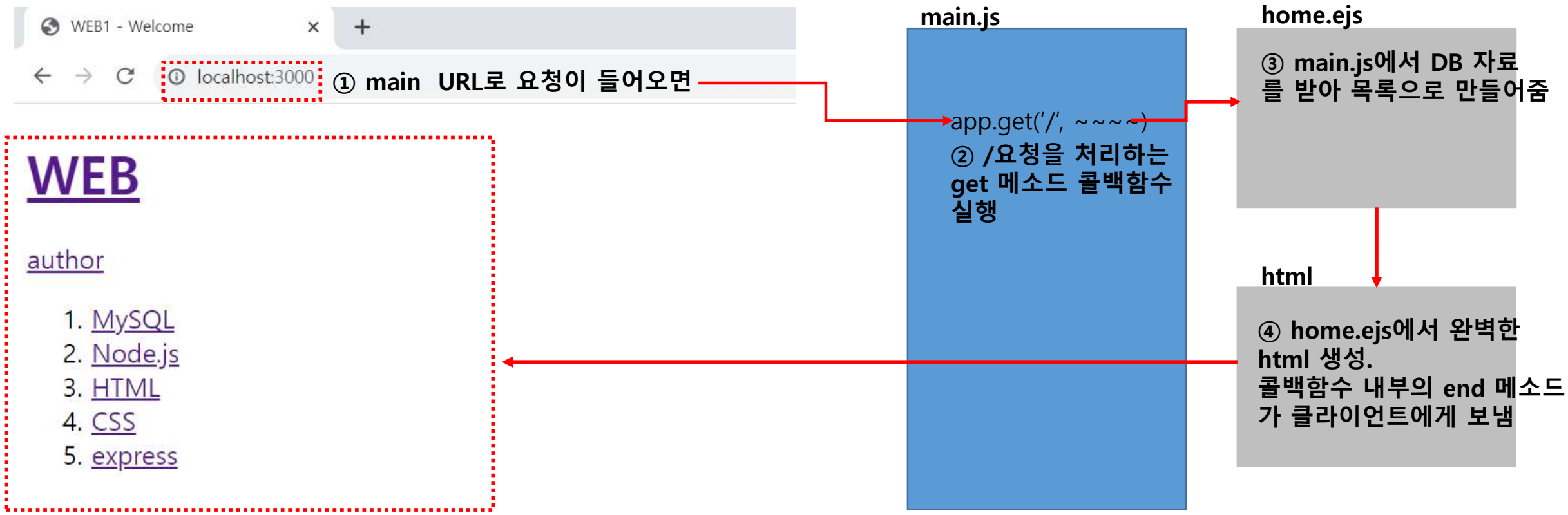
2. home.ejs

```
<!doctype html>
<html>
    <head>
        <title>WEB1 - <%= title %></title>
        <meta charset="utf-8">
    </head>
    <body>
        <h1><a href="/">WEB</a></h1>
        <a href="/author">author</a>
        <%= list %>

    </body>
</html>
```

화면에 object라고 만 출력 ➔ results[0]는 객체이기 때문

04 mysql 모듈을 이용한 홈페이지 구현 – topic 테이블의 R

3. topic 제목으로 목록 만들기



04 mysql 모듈을 이용한 홈페이지 구현 – topic 테이블의 R

3. topic 제목으로 목록 만들기 - ejs 파일에서 목록 만들기

main.js

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

app.set('views',__dirname+'/views');
app.set('view engine','ejs');

var mysql = require('mysql');
var connection =
mysql.createConnection({
  host : 'localhost',
  user : 'nodejs',
  password : 'nodejs',
  database : 'webdb2023'
})
connection.connect();
```

```
app.get('/',(req,res)=>{

  connection.query('SELECT * FROM topic', (error,results)=>{
    var context = {list:results,
                   title:'Welcome'};
    console.log(context)
    res.render('home', context, (err,html)=>{
      res.end(html)  })
  });
  connection.end();

})

app.get('/favicon.ico',(req, res)=>res.writeHead(404));
app.listen(3000, () => console.log('Example app listening on port 3000')) ;
```

04 mysql 모듈을 이용한 홈페이지 구현 – topic 테이블의 R

3. topic 제목으로 목록 만들기 - ejs 파일에서 목록 만들기

home.ejs

```
<!doctype html>
<html>
    <head>
        <title>WEB1 - <%= title %></title>
        <meta charset="utf-8">
    </head>
    <body>
        <h1><a href="/">WEB</a></h1>
        <a href="/author">author</a>
        <ol type="1">
        <% var i = 0;
           while( i < list.length)
           {
        %>
                <li><a  href="#" ><%= list[i].title %></a>
        <%
                i += 1;
           }
        %>

        </ol>
    </body>
</html>
```

```
req.app.render('home', context, (err,html)=>{
  res.end(html)  })
```

html

```
<!doctype html>
<html>
    <head>
        <title>WEB1 - Welcome</title>
        <meta charset="utf-8">
    </head>
    <body>
        <h1><a href="/">WEB</a></h1>
        <a href="/author">author</a>
        <ol type="1">

                <li><a  href="#" >MySQL</a>

                <li><a  href="#" >Node.js</a>

                <li><a  href="#" >HTML</a>

                <li><a  href="#" >CSS</a>

                <li><a  href="#" >express</a>
        </ol>
    </body>
</html>
```

3. topic 제목으로 목록 만들기 - ejs 파일에서 목록 만들기 – ejs 설명

home.ejs

```
<!doctype html>
<html>
    <head>
        <title>WEB1 - <%= title %></title>
        <meta charset="utf-8">
    </head>
    <body>
        <h1><a href="/">WEB</a></h1>
        <a href="/author">author</a>
        <ol type="1">
        <% var i = 0; ①
           while( i < list.length) ②
           {
        %> ③
                <li><a  href="#" ><%= list[i].title %></a> ④
        <%
                i += 1;
           }
        %>

        </ol>
    </body>
</html>
```

① js 문법을 html 내에서 사용할 때는 <% 시작해서 %> 종료
template 태그는 html의 어느 위치에 와도 된다.
js 문법에 따라 변수 i 를 선언하고 0으로 초기화

② main.js로부터 받은 list객체에는 DB의 데이터 row 들이 저장되어 있고 row들의 개수가 length 멤버변수에 저장되어 있다
js 문법의 while 문 시작

③ 다음에 html 문이 나와야 하기 때문에 template 언어를 종료해야 한다. 그래서 %>로 종료

④ 목록을 만드는 li 태그. li 태그가 topic 테이블의 row 개수 만큼 생성
<%= %>는 main.js에서 받은 변수의 내용을 출력한다.
list[i]는 i번째 row이고 .tiltle은 title 속성(컬럼 또는 필드)을 의미.

04 mysql 모듈을 이용한 홈페이지 구현 – topic 테이블의 R

3. topic 제목으로 목록 만들기 - ejs 파일에서 목록 만들기 – ejs 설명

home.ejs

```
<!doctype html>
<html>
    <head>
        <title>WEB1 - <%= title %></title>
        <meta charset="utf-8">
    </head>
    <body>
        <h1><a href="/">WEB</a></h1>
        <a href="/author">author</a>
        <ol type="1">
        <% var i = 0;
           while( i < list.length)
           {
        %>
                <li><a  href="#" ><%= list[i].title %></a>
        <%  ⑤
                i += 1; ⑥
           }
        %>  ⑦

        </ol>
    </body>
</html>
```

⑤ html 문장 뒤에 js 문법이 필요하여 <%로 시작

⑥ while 문에서 반복 종료를 위해 사용되는 변수 i의 값을 1 증가한다.
while 문의 끝을 의미하는 } 이 필요하다

⑦ template 언어의 끝을 알리는 %> 태그

※ template 언어와 html 문장은 섞어서 사용 가능

04 mysql 모듈을 이용한 홈페이지 구현 – topic 테이블의 R

3. topic 제목으로 목록 만들기 - node 파일에서 목록 만들기

home.ejs

```
<!doctype html>
<html>
    <head>
        <title>WEB1 - <%= title %></title>
        <meta charset="utf-8">
    </head>
    <body>
        <h1><a href="/">WEB</a></h1>
        <a href="/author">author</a>

        <%- list %> ①

    </body>
</html>
```

① <%- %>

main.js에서 넘어오는 list 변수의 내용에 html 이 포함되어 있고 그 html을 문자열이 아닌 html 문장으로 번역하여 출력

04 mysql 모듈을 이용한 홈페이지 구현 – topic 테이블의 R

3. topic 제목으로 목록 만들기 - node 파일에서 목록 만들기

main.js의 '/' 요청에 대한 콜백함수 수정

```
app.get('/',(req,res)=>{

  connection.query('SELECT * FROM topic', (error,results)=>{
    /* 여기서부터 추가된 내용 */
    var lists = '<ol type="1">'; ①
    var i = 0;  ②
    while(i < results.length) {
        lists = lists + `<li><a href="#">${results[i].title}</a></li>`; ③
        i = i + 1; ④
    }
    lists = lists+'</ol>';⑤
    /* 추가된 내용 끝 */
    var context = {list:lists, // results가 아니라 lists를 넘겨줌 ⑥
                   title:'Welcome'};
    console.log(context)
    req.app.render('home', context, (err,html)=>{
      res.end(html)  })
  });
  connection.end();

})
```

① lists 변수를 선언하고 목록 생성에 필요한 <ol> 태그를 저장한다.

② while 종료에 필요한 변수 i 선언 및 초기화

③ lists 변수에 <li> 태그를 topic 테이블의 row의 개수만큼 추가한다. ${}는 백틱 문자열 내부에 변수 사용
results[i]는 topic 테이블의 i번째 row. .tiltle은 title 속성(컬럼 또는 필드)을 의미.

④ while 문에서 반복 종료를 위해 사용되는 변수 i의 값을 1 증가한다.

⑤ ol 종료 태그를 추가

⑥ results 대신 html 문장인 lists를 넘겨줌

04 mysql 모듈을 이용한 홈페이지 구현 – topic 테이블의 R

4. 모듈화 하기

01-02-func.js

```
const { odd, even } = require('./01-02-var'); //reuqire 함수 : 외부 모듈을 객체로 반환
const mtest = require('./01-02-var');
function checkOddOrEven(num){
    if (num%2) {
        return odd;
    }
    return even;
}

console.log(checkOddOrEven(5));

function checkOddOrEven2(num){
    if (num%2) {
        return mtest.odd;
    }
    return mtest.even;
}

console.log(checkOddOrEven2(8));
```

01-02-var.js

```
const odd = '홀수입니다';
const even = '짝수입니다';

module.exports = {   //module 객체의 exports 속성에 외부에 사용가능하게 할
    odd,             //변수, 또는 함수를 객체 형태로 전달
    even,
};
```

모듈 : 특정한 기능을 하는 함수나 변수들의 집합

04 mysql 모듈을 이용한 홈페이지 구현 – topic 테이블의 R

4. 모듈화 하기

① home 디렉토리에 lib 폴더를 생성한다.

② 데이터 베이스 부분을 모듈로 옮긴다. main.js에서는 db모듈을 import한다.

./lib/db.js

```
var mysql = require('mysql');
var db =
mysql.createConnection({
  host : 'localhost',
  user : 'nodejs',
  password : 'nodejs',
  database : 'webdb2023'
})
db.connect();
module.exports = db;
```

main.js

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

app.set('views',__dirname+'/views');
app.set('view engine','ejs');

var db = require('./lib/db');  // DB 관련 line들을 삭제하고 db모듈을 import 하는 코드추가
```

04 mysql 모듈을 이용한 홈페이지 구현 – topic 테이블의 R

4. 모듈화 하기

③ topic.js 파일을 생성한다.

④ app.get('/',~~~) 의 콜백함수의 내용을 모듈로 옮긴다. 모듈에 옮길 때는 객체화 한다.

main.js

```
app.get('/',(req,res)=>{

  db.query('SELECT * FROM topic',
(error,results)=>{  //connection이었던 변수를 db로 수정
    /* 여기서부터 추가된 내용 */
    var lists = '<ol type="1">';
    var i = 0;
    while(i < results.length) {
        lists = lists + `<li><a
href="#">${results[i].title}</a></li>`;
        i = i + 1;
    }
    lists = lists+'</ol> ';
    /* 추가된 내용 끝 */
    var context = {list:lists, // results가 아니라 lists를 넘겨줌
                   title:'Welcome-db 모듈 생성'};
    console.log(context)
    res.render('home', context, (err,html)=>{
      res.end(html)  })
  });
  db.end();

})
```

이부분의 내용을 멤버 함수화 하기 위해 모듈로 옮긴다. 이 부분에는 모듈의 객체에 있는 메소드를 호출하는 코드를 추가한다.

```
var topic = require('./lib/topic');
// topic 모듈을 추가한다.

app.get('/',(req,res)=>{

  topic.home(req,res)

})
```

4. 모듈화 하기

⑤ 모듈 내부 형태

1. 객체변수를 먼저 정의하고 module.exports에 객체변수 저장

```
var obj = {
    멤버변수 : 값 ,
    멤버함수 : (  ) => {  },
}

module.exports = obj;
```

2. 객체를 module.exports에 바로 저장

```
module.exports = {
    멤버변수 : 값 ,
    멤버함수 : (  ) => {  },
};
```

04 mysql 모듈을 이용한 홈페이지 구현 – topic 테이블의 R

4. 모듈화 하기

⑥ module.exports에 객체를 바로 저장하는 방법으로 topic 모듈 작성

./lib/topic.js

```
const db = require('./db');

module.exports = {
    home : (req,res) => {

        db.query('SELECT * FROM topic', (error,results)=>{  //connection이었던 변수를 db로 수정
            /* 여기서부터 추가된 내용 */
            var lists = '<ol type="1">';
            var i = 0;
            while(i < results.length) {
                lists = lists + `<li><a href="#">${results[i].title}</a></li>`;
                i = i + 1;
            }
            lists = lists+'</ol> ';
            /* 추가된 내용 끝 */
            var context = {list:lists, // results가 아니라 lists를 넘겨줌
                           title:'welcome-db 모듈 생성'};
            console.log(context)
            res.render('home', context, (err,html)=>{
              res.end(html)  })
          });
        db.end();
        }
}
```

04 mysql 모듈을 이용한 홈페이지 구현 – topic 테이블의 R

4. 모듈화 하기

⑥ module.exports에 객체를 바로 저장하는 방법으로 topic 모듈 작성

main.js

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

app.set('views',__dirname+'/views');
app.set('view engine','ejs');

var db = require('./lib/db');  // DB 관련 line들을 삭제하고 db모듈을 import 하는 코드추가
var topic = require('./lib/topic');  // topic 모듈을 추가한다.

app.get('/',(req,res)=>{
  topic.home(req,res);
})

app.get('/favicon.ico',(req, res)=>res.writeHead(404));
app.listen(3000, () => console.log('Example app listening on port 3000')) ;
```

1. link 만들기 – 세부 내용 descrpt 필드를 보여주는 기능

main.js

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

app.set('views',__dirname+'/views');
app.set('view engine','ejs');

var db = require('./lib/db');  // DB 관련 line들을 삭제하고 db모듈을 import 하는 코드추가
var topic = require('./lib/topic');  // topic 모듈을 추가한다.

app.get('/',(req,res)=>{
  topic.home(req,res);
})

app.get('/:id',(req,res)=>{
  topic.page(req,res);
})


app.get('/favicon.ico',(req, res)=>res.writeHead(404));
app.listen(3000, () => console.log('Example app listening on port 3000')) ;
```