<생활코딩 Node.js – MySQL 요약>

기술설계팀 35기 장진희

1. 필요한 배경지식
2. 생활코딩 Node.js, MySQL수업

\*필요한 코드는 https://opentutorials.org/module/3560/21175에서 다운

1. Git bash, atom, cmd 등등 사용법

\*강의 영상에서는 visual studio code를 사용하고 있음

1. 사전 준비
2. 다운받은 코드를 atom에서 열고 example.sql의 코드를 cmd창을 통에 mySQL에 입력

\*cmd창에서 C:\Bitnami\wampstack-7.2.20-1\mysql\bin까지 접속한 후 mysql명령을 실행해야 함

1. Git bash에서 npm install 실행

\* C:\Users\betty\Downloads\node.js-mysql-1\node.js-mysql-1까지 접속한 후 실행해야 함

1. Node-modules에 Sanitize-html이 추가되었으면 성공
2. Pm2를 이용하여(자세한 방법은 node.js요약 참고) pm2 start main.js –watch 시행
3. Local:3000 에서 페이지가 정상적으로 작동하는지 확인
4. npm install –save mysql 시행
5. package.json의 dependencieds에 “mysql” : “^2.15.0”이 추가되었는지 확인
6. node.js에 mysql.js라는 파일 만들어 아래의 코드 붙여넣기

var mysql = require('mysql');

var connection = mysql.createConnection({

host : 'localhost',

user : 'root',

password : '111111', //본인이 정한 비밀번호로 변경

database : 'opentutorials'

});

connection.connect();

connection.query('SELECT \* FROM topic', function (error, results, fields) {

if (error) {

console.log(error);

}

console.log(results);

});

connection.end();

1. pm2 log 시행
2. 1차 코드
3. Main.js

var http = require('http');

var fs = require('fs');

var url = require('url');

var qs = require('querystring');

var template = require('./lib/template.js');

var path = require('path');

var sanitizeHtml = require('sanitize-html');

var mysql = require('mysql'); //mysql 모듈 가져오기

var db = mysql.createConnection({

host : 'localhost',

user : 'root',

password : 'wlsgml214', //본인이 설정한 비밀번호로 변경

database : 'opentutorials' //연결하려는 database의 이름

});

db.connect(); //mysql에 접속

var app = http.createServer(function(request,response){

var \_url = request.url;

var queryData = url.parse(\_url, true).query;

var pathname = url.parse(\_url, true).pathname;

if(pathname === '/'){ //홈화면

if(queryData.id === undefined){ //홈 화면일 때, SQL문을 실행(READ)

db.query(`SELECT \* FROM topic`, function (error, topics) {

var title = 'Welcome';

var description = 'Hello, Node.js';

var list = template.list(topics); //template.js 에 있는 list함수 호출

var html = template.HTML(title, list, //template.js 에 있는 html함수 호출

`<h2>${title}</h2>${description}`, //title, list, 본문내용을 argument로 전달

`<a href="/create">create</a>` //본문내용에 생성버튼(create) 설정(링크 이용)

);

response.writeHead(200);

response.end(html); //html 변수 호출

});

} else {

db.query(`SELECT \* FROM topic`, function (error, topics) { //에러처리

if(error){

throw error;

}

db.query(`SELECT \* FROM topic WHERE id =?`,[queryData.id],function(error2,topic){

//id 값에 맞는 페이지로 들어가기

//? [ ]를 사용하여 보안성을 높임(자바스크립트 형태로 들어온 입력내용 처리)

if(error2){ //에러처리2

throw error2;

}

var title = topic[0].title; //topic변수는 배열의 형태이므로 topic[0]으로 표현해야 함

var description = topic[0].description;

var list = template.list(topics);

var html = template.HTML(title, list,

`<h2>${title}</h2>${description}`,

` <a href="/create">create</a> //생성(create)링크 생성

<a href="/update?id=${queryData.id}">update</a> //수정(update)링크 생성

<form action="delete\_process" method="post"> //삭제(delete)버튼 생성

//delete는 항상 링크의 형태가 아닌 post의 형태로 데이터를 전달해야함

<input type="hidden" name="id" value="${queryData.id}">

//삭제할 내용을 hidden의 형태로 전달

<input type="submit" value="delete">

</form>`

);

response.writeHead(200);

response.end(html);

})

});

}

} else if(pathname === '/create'){ //생성(create)화면

db.query(`SELECT \* FROM topic`, function (error, topics) {

var title = 'Create';

var list = template.list(topics);

var html = template.HTML(title, list,`

<form action="/create\_process" method="post">

//생성처리(create\_process) 시행

<p><input type="text" name="title" placeholder="title"></p>

//생성할 글의 제목칸

<p> //문단 설정

<textarea name="description" placeholder="description"></textarea>

//생성할 글의 내용칸

</p>

<p>

<input type="submit">

</p>

</form>

`,

`<a href="/create">create</a>`

);

response.writeHead(200);

response.end(html);

});

} else if(pathname === '/create\_process'){ //생성처리(create\_process)

var body = ''; //본문 내용의 초기값

request.on('data', function(data){ //data를 argument로 받은 함수

body = body + data; //입력된 내용(data)을 초기값에 추가하기

});

request.on('end', function(){

var post = qs.parse(body);

db.query(`INSERT INTO topic (title,description,created,author\_id)

VALUES(?,?,NOW(),?)`, //NOW()는 MySQL의 함수

[post.title,post.description,1],

function(error,result){

if(error){

throw error; //에러처리

}

response.writeHead(302, {Location: `/?id=${result.insertId}`});

//result.insertId는 입력된 글의 id값을 찾기

response.end();

}

)

});

} else if(pathname === '/update'){

db.query(`SELECT \* FROM topic`, function (error, topics) {

if(error){

throw error; //에러처리

}

db.query(`SELECT \* FROM topic WHERE id =?`,[queryData.id],function(error2,topic){

if(error2){

throw error2; //에러처리2

}

var title = 'Update';

var list = template.list(topics);

var html = template.HTML(topic[0].title, list,`

<form action="/update\_process" method="post"> //수정처리(create\_process) 시행

<input type="hidden" name="id" value="${topic[0].id}">

<p>

<input type="text" name="title" placeholder="title" value="${topic[0].title}">

</p>

<p>

<textarea name="description" placeholder="description">${topic[0].description}</textarea>

</p>

<p>

<input type="submit">

</p>

</form>

`,

`<a href="/create">create</a> <a href="/update?id=${topic[0].id}">update</a>`

);

response.writeHead(200);

response.end(html);

});

});

} else if(pathname === '/update\_process'){ //수정처리(update\_process)

var body = '';

request.on('data', function(data){

body = body + data;

});

request.on('end', function(){

var post = qs.parse(body);

db.query('UPDATE topic SET title=?,description=?,author\_id=1 WHERE id=?',

[post.title,post.description,post.id],function(error,result){

response.writeHead(302, {Location: `/?id=${post.id}`}); //id값에 맞는 글을 수정

response.end();

})

});

} else if(pathname === '/delete\_process'){ //삭제처리(delete\_process)

var body = '';

request.on('data', function(data){

body = body + data;

});

request.on('end', function(){

var post = qs.parse(body);

db.query(`DELETE FROM topic WHERE id = ?`,[post.id],function(error,result){

if(error){

throw error;

}

response.writeHead(302, {Location: `/`});

response.end();

});

});

} else { //파일을 찾지 못했을 때 오류처리

response.writeHead(404);

response.end('Not found');

}

});

app.listen(3000); //3000번 port일 때 시행

1. 최종 코드

\*1차 코드와 같은 부분은 설명을 생략

1. Main.js

var http = require('http');

var url = require('url');

var topic = require('./lib/topic');

var author = require('./lib/author');

//topic.js와 author.js에 필요한 내용들을 써두고 불러옴으로서 main.js를 단순화시킴

var app = http.createServer(function(request,response){

var \_url = request.url;

var queryData = url.parse(\_url, true).query;

var pathname = url.parse(\_url, true).pathname;

//각 페이지에 맞는 함수를 호출하기만 하는 형태

if(pathname === '/'){ //홈페이지

if(queryData.id === undefined){

topic.home(request, response);

} else {

topic.page(request, response);

}

} else if(pathname === '/create'){ //생성(create)페이지

topic.create(request, response);

} else if(pathname === '/create\_process'){ //생성처리(create\_process)

topic.create\_process(request, response);

} else if(pathname === '/update'){ //수정(update)페이지

topic.update(request, response);

} else if(pathname === '/update\_process'){ //수정처리(update\_process)

topic.update\_process(request, response);

} else if(pathname === '/delete\_process'){ //삭제처리(delete\_process)

topic.delete\_process(request, response);

} else if(pathname === '/author'){ //작가 홈페이지

author.home(request, response);

} else if(pathname === '/author/create\_process'){ //작가 생성(create)페이지

author.create\_process(request, response);

} else if(pathname === '/author/update'){ //작가 수정(update)페이지

author.update(request, response);

} else if(pathname === '/author/update\_process'){ //작가 수정처리(update\_process)

author.update\_process(request, response);

} else if(pathname === '/author/delete\_process'){ //작가 삭제처리(delete\_process)

author.delete\_process(request, response);

} else {

response.writeHead(404);

response.end('Not found');

}

});

app.listen(3000);

1. Template.js

module.exports = { //하나의 모듈을 다른 파일에서도 불러올 수 있도록 함

HTML:function(title, list, body, control){

//html함수 : title,list,body,control을 argument로 받음

return `

<!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} //내용을 쿼리스트링의 형식으로 삽입

${control}

${body}

</body>

</html>

`;

},list:function(topics){ //list 함수 : 제목을 argument로 받음

var list = '<ul>'; //list 초기값

var i = 0;

while(i < topics.length){ //받은 제목을 list의 형태로 합침

list = list + `<li><a href="/?id=${topics[i].id}">${topics[i].title}</a></li>`;

i = i + 1;

}

list = list+'</ul>';

return list;

},authorSelect:function(authors, author\_id){

//authorSelect함수 : author, author\_id를 arguments로 받음

var tag = ''; //tag초기값

var i = 0;

while(i<authors.length){

var selected = '';

if(authors[i].id === author\_id){

selected = ' selected';

}

tag +=`<option value='${authors[i].id}'${selected}>${authors[i].name}</option>`;

//tag에 찾은(selected된) 작가의 id와 이름을 담음

i++; //리스트를 돌면서 일치하는 작가를 찾음

}

return `

<select name = "author">

${tag} // tag를 return

</select>

`

},authorTable:function(authors){ //authorTable함수 : authors를 argument로 받음

var tag = '<table>'; //tag초기값, 표(table)의 형태

var i = 0;

while(i<authors.length){ //모든 작가들을 while문을 돌며 list처럼 합침

tag +=`

<tr>

<td>${authors[i].name}</td>

<td>${authors[i].profile}</td>

<td><a href = "/author/update?id=${authors[i].id}">update</a></td>

//update버튼

<td>

<form action="/author/delete\_process" method="post"> //delete버튼

<input type="hidden" name="id" value="${authors[i].id}">

<input type="submit" value="delete">

</form>

</td>

</tr>`

i++;

}

tag +='</table>';

return tag;

}

}

1. Topic.js

//본문 내용을 불러오는 모듈

var db = require('./db.js');

var template = require('./template.js');

var url = require('url');

var qs = require('querystring');

exports.home = function(request, response){

//다수의 모듈들을 다른 파일에서도 불러올 수 있도록 함(exports)

db.query(`SELECT \* FROM topic`, function (error, topics) {

var title = 'Welcome';

var description = 'Hello, Node.js';

var list = template.list(topics);

var html = template.HTML(title, list,

`<h2>${title}</h2>${description}`,

`<a href="/create">create</a>`

);

response.writeHead(200);

response.end(html);

});

}

exports.page = function(request, response){

var \_url = request.url;

var queryData = url.parse(\_url, true).query;

db.query(`SELECT \* FROM topic`, function (error, topics) {

if(error){

throw error;

}

db.query(`SELECT \* FROM topic LEFT JOIN author ON topic.author\_id=author.id WHERE topic.id =?`,[queryData.id],function(error2,topic){

if(error2){

throw error2;

}

console.log(topic);

var title = topic[0].title;

var description = topic[0].description;

var list = template.list(topics);

var html = template.HTML(title, list,

`<h2>${title}</h2>${description}<p>by ${topic[0].name}</p>`,

` <a href="/create">create</a>

<a href="/update?id=${queryData.id}">update</a>

<form action="delete\_process" method="post">

<input type="hidden" name="id" value="${queryData.id}">

<input type="submit" value="delete">

</form>`

);

response.writeHead(200);

response.end(html);

})

});

}

exports.create = function(request,response){

db.query(`SELECT \* FROM topic`, function (error, topics) {

db.query(`SELECT \* FROM author`, function (error2, authors){

var title = 'Create';

var list = template.list(topics);

var html = template.HTML(title, list,`

<form action="/create\_process" method="post">

<p><input type="text" name="title" placeholder="title"></p>

<p>

<textarea name="description" placeholder="description"></textarea>

</p>

<p>

${template.authorSelect(authors)}

</p>

<p>

<input type="submit">

</p>

</form>

`,

`<a href="/create">create</a>`

);

response.writeHead(200);

response.end(html);

});

});

}

exports.create\_process = function(request,response){

var body = '';

request.on('data', function(data){

body = body + data;

});

request.on('end', function(){

var post = qs.parse(body);

db.query(`INSERT INTO topic (title,description,created,author\_id)

VALUES(?,?,NOW(),?)`,

[post.title,post.description,post.author],

function(error,result){

if(error){

throw error;

}

response.writeHead(302, {Location: `/?id=${result.insertId}`});

response.end();

}

)

});

}

exports.update = function(request,response){

var \_url = request.url;

var queryData = url.parse(\_url, true).query;

db.query(`SELECT \* FROM topic`, function (error, topics) {

if(error){

throw error;

}

db.query(`SELECT \* FROM topic WHERE id =?`,[queryData.id],function(error2,topic){

if(error2){

throw error2;

}

db.query(`SELECT \* FROM author`, function (error3, authors){

var title = 'Update';

var list = template.list(topics);

var html = template.HTML(topic[0].title, list,`

<form action="/update\_process" method="post">

<input type="hidden" name="id" value="${topic[0].id}">

<p>

<input type="text" name="title" placeholder="title" value="${topic[0].title}">

</p>

<p>

<textarea name="description" placeholder="description">

${topic[0].description}

</textarea>

</p>

<p>

${template.authorSelect(authors, topic[0].author\_id)}

</p>

<p>

<input type="submit">

</p>

</form>

`,

`<a href="/create">create</a>

<a href="/update?id=${topic[0].id}">update</a>`

);

response.writeHead(200);

response.end(html);

});

});

});

}

exports.update\_process = function(request,response){

var body = '';

request.on('data', function(data){

body = body + data;

});

request.on('end', function(){

var post = qs.parse(body);

db.query('UPDATE topic SET title=?,description=?,author\_id=? WHERE id=?',

[post.title,post.description,post.author,post.id],function(error,result){

response.writeHead(302, {Location: `/?id=${post.id}`});

response.end();

})

});

}

exports.delete = function(request,response){

var body = '';

request.on('data', function(data){

body = body + data;

});

request.on('end', function(){

var post = qs.parse(body);

db.query(`DELETE FROM topic WHERE id = ?`,[post.id],function(error,result){

if(error){

throw error;

}

response.writeHead(302, {Location: `/`});

response.end();

});

});

}

1. Author.js

//작가관련 내용을 불러오는 모듈

var db = require('./db');

var template = require('./template.js');

var qs = require('querystring');

var url = require('url');

exports.home = function(request, response){

//홈페이지에 생성(create)페이지가 합쳐진 형태

db.query(`SELECT \* FROM topic`, function(error,topics){

db.query(`SELECT \* FROM author`, function(error2,authors){

var title = 'author';

var list = template.list(topics);

var html = template.HTML(title, list,`

${template.authorTable(authors)}

<style> //표(table)의 css코드

table{

border-collapse: collapse;

}

td{

border:1px solid black; //검정으로 테두리 채우기

}

</style>

<form action="/author/create\_process" method="post">

<p>

<input type="text" name="name" placeholder="name">

</p>

<p>

<textarea name="profile" placeholder="description"></textarea>

</p>

<p>

<input type="submit" value="create">

</p>

</form>`, ``

);

response.writeHead(200);

response.end(html);

});

});

}

exports.create\_process = function(request, response){

var body = '';

request.on('data', function(data){

body = body + data;

});

request.on('end', function(){

var post = qs.parse(body);

db.query(`

INSERT INTO author (name, profile) VALUES(?, ?)`, [post.name, post.profile],

function(error, result){

if(error){

throw error;

}

response.writeHead(302, {Location: `/author`});

response.end();

}

)

});

}

exports.update = function(request, response){

db.query(`SELECT \* FROM topic`, function(error,topics){

db.query(`SELECT \* FROM author`, function(error2,authors){

var \_url = request.url;

var queryData = url.parse(\_url, true).query;

db.query(`SELECT \* FROM author WHERE id=?`,[queryData.id], function(error3,author){

var title = 'author';

var list = template.list(topics);

var html = template.HTML(title, list,`

${template.authorTable(authors)}

<style>

table{

border-collapse: collapse;

}

td{

border:1px solid black;

}

</style>

<form action="/author/update\_process" method="post">

<p>

<input type="hidden" name="id" value="${queryData.id}">

</p>

<p>

<input type="text" name="name" value="${author[0].name}" placeholder="name">

</p>

<p>

<textarea name="profile" placeholder="description">

${author[0].profile}

</textarea>

</p>

<p>

<input type="submit" value="update">

</p>

</form>

`, ``

);

response.writeHead(200);

response.end(html);

});

});

});

}

exports.update\_process = function(request, response){

var body = '';

request.on('data', function(data){

body = body + data;

});

request.on('end', function(){

var post = qs.parse(body);

db.query(`

UPDATE author SET name=?, profile=? WHERE id=?`, [post.name, post.profile, post.id], function(error, result){

if(error){

throw error;

}

response.writeHead(302, {Location: `/author`});

response.end();

}

)

});

}

exports.delete\_process = function(request, response){

var body = '';

request.on('data', function(data){

body = body + data;

});

request.on('end', function(){

var post = qs.parse(body);

db.query(`DELETE FROM topic WHERE author\_id=?`,[post.id],

function(error1, result1){

if(error1){

throw error1;

}

db.query(`

DELETE FROM author WHERE id=?`, [post.id],

function(error2, result2){

if(error2){

throw error2;

}

response.writeHead(302, {Location: `/author`});

response.end();

}

)

}

);

});

}

1. 더 알아보기
2. 검색기능

<form method=”get”> / SELECT \* FROM topic WHERE title = “keyword”

1. 색인기능(index)
2. 정렬기능

SELECT \* FROM topic ORDER BY id DESC

1. 페이지기능

SELECT \* FROM topic LIMIT 0 OFFSET 20

1. NOSQL(Not Only SQL)