Node -> node.js교과서 p169~p216

-요청과 응답

파일경로: node/ch4/server1.js

const http = require('http');

http.createServer((req,res)=>{// request는 요청에대한 정보를 response는 응답에관한 정보를나타내며 요청이들어올때마다 해당인수 객체를가져와 콜백함수가 진행된다

//res객체는 writeHead write end 세가지 메서드가있다

res.writeHead(200,{'Content-Type':'text/html; charset=utf-8'});//응답에대한 정보를 기록하는 메서드 매개변수 첫번째 200 은 성공적인요청임을

res.write('<h1>Hello Node!</h1>');//첫번째 인수는 클라이언트로 보낼 데이터, 버퍼를 보낼수도있으며 여러번 호출하여 데이터를 여러개 보내도된다

res.end('<p>Hello Server!</p>');//응답을 종료하는 메서드 만약인수가 있다면 그데이터도 클라이언트로 보내고 응답을 종료한다

}).listen(8080,()=>{//listen메서드를 붙여 클라이언트에 공개할 포트번호와 포트 연결 완료후 실행될 콜백함수를 넣는다 즉 서버연결

console.log('8080번 포트에서 서버 대기 중입니다');//실행될 콜백내용

})

-‘listening’이벤트를 따로만들어 콜백함수 지정하기

파일경로: node/ch4/server1-1.js

const { error } = require('console');

const http = require('http');

const server =http.createServer((req,res)=>{//만들어질서버의 객체를 server에담는다

res.writeHead(200,{'Content-Type':'text/html; charset=utf-8'});//응답에대한 정보를 기록하는 메서드 매개변수 첫번째 200 은 성공적인요청임을

res.write('<h1>Hello Node!</h1>');//첫번째 인수는 클라이언트로 보낼 데이터, 버퍼를 보낼수도있으며 여러번 호출하여 데이터를 여러개 보내도된다

res.end('<p>Hello Server!</p>');//응답을 종료하는 메서드 만약인수가 있다면 그데이터도 클라이언트로 보내고 응답을 종료한다

});

server.listen(8080);

server.on('listening',()=>{//server객체에 이벤트 'listening'을 달아 따로 콜백함수를 지정하였다

console.log('8080 서버가 대기중 입니다');

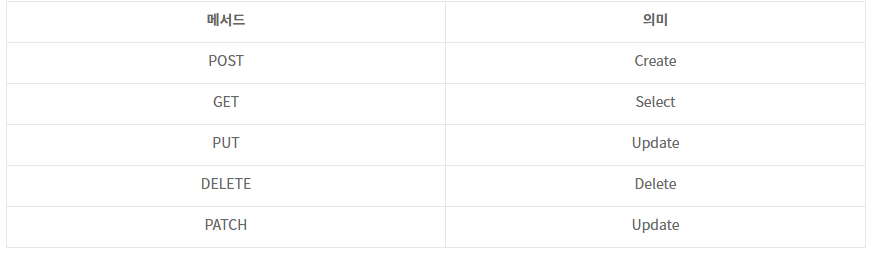
})

server.on('error',(error)=>{//error이벤트를 달아 발생한 error를 콘솔로 찍어준다

console.log(error);

})

-REST(REpresentational State Transfer): 서버의 자원을 정의하고 자원에 대한 주소를 지정하는 방법을 가리킨다 /user라면 사용자 정보에 관련된 자원을요청하는것,/post라면 게시글에 관련된 자원을 요청하는것



파일경로: node/ch4/restFront.js

async function getUser() {

try {

const res = await axios.get('/users');//await 키워드를 사용하여 axios.get()요청의 응답을 기다린다

//이때 await은 비동기함수 내에서만 사용할수있으며 해당함수가 완료될때까자지 코드실행을 일시 중단

const users = res.data;//res.data는 axios.get()로 서버로부터 받아온 응답데이터를 가리킨다

const list = document.getElementById('list');

/\* document는 현재 열려있는 웹 페이지의 문서 객체를 나타냅니다. getElementById() 메소드는 인수로

 전달된 id를 가진 HTML 요소를 찾아서 반환합니다. 반환된 요소는 JavaScript 객체로, 해당 요소의

속성과 메소드를 사용하여 요소를 조작할 수 있습니다.\*/

list.innerHTML = ''; //list요소 내부 HTML을 빈문자열로 설정

Object.keys(users).map(function (key) {

/\*Object.keys(users)는 users객체의 키들을 배열로 반환하며

반환되배열을 map()메소드의 첫번째 인수로 전달즉 콜백함수의 인자 key는 배열의 현재요소

즉 users객체의 각키를 의미하게된다 이하{}는 각키에대한 콜백함수 \*/

const userDiv = document.createElement('div');//document 현재열려있는 문서객체에

//createElement('div')인수에해당하는 새로운 HTML요소를 생성한다

const span = document.createElement('span');

span.textContent = users[key];

//요소.textContent 해당요소의 텍스트 내용을 users객체의 key에 해당하는 값을 설정

const edit = document.createElement('button');

edit.textContent = '수정';

edit.addEventListener('click', async () => {

const name = prompt('바꿀이름을 입력하세요');

if (!name) {

return alert("이름을 반드시 입력해야합니다");

}

try {

await axios.put('/user/' + key, { name });

// /user/경로에 key값을 덧붙여 put요청을 내보내고 요청 본문은 {name}객체이다

getUser()

//await 으로 성공적으로 처리되면 getUser()함수를 호출하여 사용자목록을 다시 불러온다

} catch (err) {

console.error(err);

}

})

const remove = document.createElement('button');

remove.textContent = '삭제';

remove.addEventListener('click', async () => {

try {

await axios.delete('/user/' + key);

getUser();

} catch (err) {

console.error(err);

}

});

userDiv.appendChild(span);

userDiv.appendChild(edit);

userDiv.appendChild(remove);

list.appendChild(userDiv);

console.log(res.data);

});

}catch(err){

console.error(err);

}

}

window.onload = getUser;

document.getElementById('form').addEventListener('submit',async (e)=>{

e.preventDefault();//submit의 기본동작인페이지를 새로고침하는것을 취소

const name = e.target.username.value;//e.target은 이벤트가발생한 HTML요소중 id가 username인 값에접근

if(!name){

return alert("이름을 입력하세요");

}

try{

await axios.post('/user',{name});//사용자를 등록하기위해 axios.post를하여 HTTP POST요청을 보내는것으로

//요청 URL은 '/user'이다 이때 {name}이 요청본문으로 전송된다

//이때 axios.post의 문법으로 데이터를{}안에 넣어 보내야한다

getUser();

}catch(err){

console.error(err);

}

e.target.username.value='';//서버에 응답을 받은후 사용자가 입력한 값을 초기화 하기위해 사용

});

파일경로 : node/ch4/restServer.js

const http = require('http');

const fs = require('fs').promises;

const path = require('path');

const users={};

http.createServer(async (req, res) => {

try {

console.log(req.method, req.url);

/\*req.method는 HTTP요청 메서드를 나타내며 req.url은 클라이언트가 요청한 URL을 나타낸다 \*/

if (req.method === 'GET') {

if (req.url === '/') {//HTTP GET 요청이 /경로에 들어온경우

const data = await fs.readFile('./restFront.html');

console.log(path.join(\_\_dirname, 'restFront.html'));

//현재실행중인 스크립트 파일의 디렉토리 경로인 \_\_dirname과 'restFront.html'을join시킨 path

//를 얻어 fs.readFile()메서드로 파일을 비동기 적으로 읽어온다

//읽어온 파일을 data변수에 할당한다

res.writeHead(200, { 'Content-Type': 'text/html; charset=utf-8' });

return res.end(data);//클라이언트에게 data(HTML문서를담고있는)를 보낸다

} else if (req.url === '/about') {

const data = await fs.readFile(path.join(\_\_dirname, 'about.html'));

res.writeHead(200, { 'Content-Type': 'text/html; charset=utf-8' });

return res.end(data);

} else if(req.url ==='/users'){

res.writeHead(200, { 'Content-Type': 'text/html; charset=utf-8' });

return res.end(JSON.stringify(users));

//JSON.stringify(객체)는 객체를 JSON문자열로 반환하는 메소드

//문자열로 변환하는이유는 HTTP응답에서 데이터는 문자열 형태로 전송 되기때문에

}

try {//주소가 /도 /about도 /users도 아니면

const data = await fs.readFile(path.join(\_\_dirname, req.url));

return res.end(data);

} catch (err) {

console.error(err);

}

}else if(req.method === 'POST'){//사용자 등록

if(req.url === '/user'){

let body = '';

req.on('data',(data)=>{

body += data;

});

return req.on('end',()=>{

console.log('POST 본문(body)',body);

const{name}= JSON.parse(body);

const id = Date.now();//id와

users[id]= name;

res.writeHead(201, { 'Content-Type': 'text/html; charset=utf-8' });

res.end('등록성공');

});

}

}else if(req.method ==='PUT'){//사용자 id 수정

if(req.url.startsWith('/user/')){

const key = req.url.split('/')[2];

let body = '';

req.on('data',(data)=>{

body += data;

});

return req.on('end',()=>{

console.log('PUT 본문(body):',body);

users[key] = JSON.parse(body).name;

res.writeHead(200, { 'Content-Type': 'text/html; charset=utf-8' });

return res.end(JSON.stringify(users));

})

}

}else if(req.method ==='DELETE'){//사용자의 id제거

if(req.url.startsWith('/user/')){

const key = req.url.split('/')[2];

delete users[key];

res.writeHead(200, { 'Content-Type': 'text/html; charset=utf-8' });

return res.end(JSON.stringify(users));

}

}

res.writeHead(404);

return res.end('NOT FOUND');

}catch(err){

console.error(err);

res.writeHead(500,{'Content-Type': 'text/html; charset=utf-8'});

res.end(err.message);

}

}).listen(8082,()=>{

console.log('8082포트에서 서버 대기중입니다');

});