0. 서버의 요청은 4개 방식이 있음

1. 읽기(GET)
2. 쓰기(POST)
3. 수정(PUT)
4. 삭제(DELETE)

1. API는 = 웹서버와 고객간의 소통방법 = 어떻게 서버랑 통신할까?

REST API : detail?no=22를 detail/22로

2. JavaScript : HTML에 종속된 동적언어(브라우저마다 JavaScript 해석엔진이 포함됨

그중 크롬의 V8엔진을 node.js로 release 하여 브라우저 내에서 말고도 다른 환경에서도 JavaScript를 실행할 수 있게 함.

3. Node.js 특징

(1) NonpBlocking i/o : client의 요청을 다 받아 처리속도가 빠른 요청부터 응답. 자바의 경우 멀티쓰레딩이 되어 괜찮

(2) 코드가 쉬워 프로토타입을 만들기 좋다.

(3) 이미지 처리나 통계 분석 등의 서버 프로그램은 좋은 라이브러리가 없음.

4. Node.js 환경 setting up

(1) Node.js 설치(https://nodejs.org)

(2) Node.js 설치 확인 : >node -v

(3) IDE 설치 : vscode 다운로드 및 설치

(4) vscode를 소스폴더(작업폴더)로 열기

(5) express 라이브러리 설치하기

ⓐ npm init - 어떤 라이브러리를 설치했는지 알 수 있게 하기 위함(package.json에 있음)

디폴트로 entry point 는 index.js가 됨

ⓑ npm install express : express 라이브러리 설치 (npm은 라이브러리 설치를 도와주는 도구)

(mac의 경우 설치가 안 되면 : yarn설치및 재시작 후 yarn add express)

ⓒ index.js 코드를 입력 한 후 node index.js로 서버 구동후 브라우저로 확인

const express = require('express'); // express라이브러리 첨부

const app = express(); // express라이브러리를 이용해서 객체 생성

// express 라이브러리를 이용하여 서버를 띄우기 위한 기본 셋팅

// listen(서버를 띄울 포트번호, 띄운 후 실행할 코드)

app.listen(8090, () => {

  console.log('listening on 8090');

});

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

  res.send('<h2>펫 용품 쇼핑 페이지입니다</h2>');

});// app.get('경로', function(request, response){})

app.get('/beauty', function(req, res){

  res.send('<h2>뷰티 용품 쇼핑 페이지입니다</h2>');

});

app.use(express.static('public'));

app.get('/', function(req, res){

  res.sendFile(\_\_dirname + '/index.html');

});

ⓓ npm install -g nodemon ; 소스 수정시 자동으로 서버 시작

(yarn add global nodemon)

nodemon index.js가 에러나면 powershell을 관리자권한으로 실행해서 executionpolicy 실행결과가 Restricted이면 허가한 거만 실행해줌. 그래서 set-executionpolicy unrestricted 실행(Y)

ⓔ bootstrap의 Include Bootstrap’s CSS and JS 및 navBar, Jumbotron 이용하여 index.html을 작성

<!doctype html>

<html lang="en">

  <head>

    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <title>ToDo App</title>

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-KK94CHFLLe+nY2dmCWGMq91rCGa5gtU4mk92HdvYe+M/SXH301p5ILy+dN9+nJOZ" crossorigin="anonymous">

    <link href="/css/style.css" rel="stylesheet">

  </head>

  <body>

    <nav class="navbar navbar-expand-lg bg-body-tertiary">

      <div class="container-fluid">

        <a class="navbar-brand" href="/">ToDo App</a>

        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNavAltMarkup" aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-label="Toggle navigation">

          <span class="navbar-toggler-icon"></span>

        </button>

        <div class="collapse navbar-collapse" id="navbarNavAltMarkup">

          <div class="navbar-nav">

            <a class="nav-link active" aria-current="page" href="/">Home</a>

            <a class="nav-link" href="/write">Write</a>

            <a class="nav-link" href="/list">List</a>

          </div>

        </div>

      </div>

    </nav>

    <div class="container-fluid py-5">

      <h1 class="display-5 fw-bold">ToDo App에 와주셔서 감사합니다</h1>

    </div>

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" integrity="sha384-ENjdO4Dr2bkBIFxQpeoTz1HIcje39Wm4jDKdf19U8gI4ddQ3GYNS7NTKfAdVQSZe" crossorigin="anonymous"></script>

  </body>

</html>

ⓕ bootstrap의 Include Bootstrap’s CSS and JS 및 navBar, Jumbotron 이용하여 write.html을 작성

<!doctype html>

<html lang="en">

  <head>

    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <title>ToDo App</title>

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-KK94CHFLLe+nY2dmCWGMq91rCGa5gtU4mk92HdvYe+M/SXH301p5ILy+dN9+nJOZ" crossorigin="anonymous">

    <link href="/css/style.css" rel="stylesheet">

  </head>

  <body>

    <nav class="navbar navbar-expand-lg bg-body-tertiary">

      <div class="container-fluid">

        <a class="navbar-brand" href="/">ToDo App</a>

        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNavAltMarkup" aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-label="Toggle navigation">

          <span class="navbar-toggler-icon"></span>

        </button>

        <div class="collapse navbar-collapse" id="navbarNavAltMarkup">

          <div class="navbar-nav">

            <a class="nav-link active" aria-current="page" href="/">Home</a>

            <a class="nav-link" href="/write">Write</a>

            <a class="nav-link" href="/list">List</a>

          </div>

        </div>

      </div>

    </nav>

    <div class="container mt-5">

      <h1>To Do 등록 페이지</h1>

      <form  action="/write" method="post">

        <div class="mb-3">

          <label for="exampleFormControlInput1" class="form-label">할일</label>

          <input type="text" class="form-control" id="form-exampleFormControlInput1" name="title">

        </div>

        <div class="mb-3">

          <label for="exampleFormControlInput2" class="form-label">날짜</label>

          <input type="date" class="form-control" id="exampleFormControlInput2" name="date">

        </div>

        <div class="col-auto">

          <button type="submit" class="btn btn-danger mb-3">등록</button>

        </div>

      </form>

    </div>

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" integrity="sha384-ENjdO4Dr2bkBIFxQpeoTz1HIcje39Wm4jDKdf19U8gI4ddQ3GYNS7NTKfAdVQSZe" crossorigin="anonymous"></script>

  </body>

</html>

index.js에 다음 추가후 <http://localhost:8090/write>실행

app.get('/write', function(req, res){

  res.sendFile(\_\_dirname + '/write.html');

});

ⓖ style.css

body{background-color: lightyellow;}

(6) app.use(express.urlencoded({extended : true})); 을 이용해서 넘어온 파라미터 접근하기

const { response } = require('express');

const express = require('express'); // express라이브러리 첨부

const app = express(); // express라이브러리를 이용해서 객체 생성

// express 라이브러리를 이용하여 서버를 띄우기 위한 기본 셋팅

// listen(서버를 띄울 포트번호, 띄운 후 실행할 코드)

app.listen(8090, () => {

  console.log('listening on 8090');

});

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

  res.send('<h2>펫 용품 쇼핑 페이지입니다</h2>');

});// app.get('경로', function(request, response){})

app.get('/beauty', function(req, res){

  res.send('<h2>뷰티 용품 쇼핑 페이지입니다</h2>');

});

app.use(express.static('public'));

app.get('/', function(req, res){

  res.sendFile(\_\_dirname + '/index.html');

});

app.get('/write', function(req, res){

  res.sendFile(\_\_dirname + '/write.html');

});

app.use(express.urlencoded({extended : true})); // 2021년 이후 express 라이브러리는 body-parser 내포

app.post('/write', function(req, res){

  console.log(req.body.title);

  console.log(req.body.date);

  console.log(req.body);

});

5. MongoDB Atlas 사용하기

(1) Mogodb atlas 회원가입 후 클러스터 생성(512MB 무료 호스팅받음)

(2) npm install mongodb@3.6 후 소스 아래와 같이 변경 후 실행(nodemon index.js)

/\*app.listen(8090, () => {

  console.log('listening on 8090');

});

\*/

const MongoClient = require('mongodb').MongoClient;

const uri = "mongodb+srv://spacenyi:vZteS7kkCLDI31ej@cluster0.utcsmgl.mongodb.net/?retryWrites=true&w=majority";

var db;

MongoClient.connect(uri, function(err, client){

  // MongoDB 연결하면 할 일

  if(err){return console.log(err);}

  db = client.db('toDoApp2nd');

  app.listen(8090, () => {

    console.log('listening on 8090');

  });

});

6. Cluster0에 Collections 탭에 [Add My Own Data]

DataBase Name : todoapp

Collection Name : post

const { response } = require('express');

const express = require('express'); // express라이브러리 첨부

const app = express(); // express라이브러리를 이용해서 객체 생성

// express 라이브러리를 이용하여 서버를 띄우기 위한 기본 셋팅

// listen(서버를 띄울 포트번호, 띄운 후 실행할 코드)

/\*app.listen(8090, () => {

  console.log('listening on 8090');

});

\*/

const MongoClient = require('mongodb').MongoClient;

const uri = "mongodb+srv://spacenyi:vZteS7kkCLDI31ej@cluster0.utcsmgl.mongodb.net/?retryWrites=true&w=majority";

var db;

MongoClient.connect(uri, function(err, client){

  // MongoDB 연결하면 할 일

  if(err){return console.log(err);}

  db = client.db('toDoApp');

  /\*db.collection('post').insertOne({\_id : 1, 이름:'홍길동',나이:30}, function(err1, result){

    console.log('저장완료');

  });\*/

  app.listen(8090, () => {

    console.log('listening on 8090');

  });

});

app.use(express.static('public'));

app.get('/', function(req, res){

  res.sendFile(\_\_dirname + '/index.html');

});

app.get('/write', function(req, res){

  res.sendFile(\_\_dirname + '/write.html');

});

app.use(express.urlencoded({extended : true})); // 2021년 이후 express 라이브러리는 body-parser 내포

app.post('/write', function(req, res){

  console.log(req.body.title);

  console.log(req.body.date);

  console.log(req.body);

  db.collection('post').insertOne(req.body, function(err, result){

    if(err) { return console.log(err);}

    console.log('새 post 등록 완료');

  });

});

7. 상세보기를 위해서 sequence 역할을 할 Collection counter({name :’현재시퀀스', post\_sq:0}) 입력 후 소스

const { response } = require('express');

const express = require('express'); // express라이브러리 첨부

const app = express(); // express라이브러리를 이용해서 객체 생성

const MongoClient = require('mongodb').MongoClient;

const uri = "mongodb+srv://spacenyi:vZteS7kkCLDI31ej@cluster0.utcsmgl.mongodb.net/?retryWrites=true&w=majority";

var db;

MongoClient.connect(uri, function(err, client){

  // MongoDB 연결하면 할 일

  if(err){return console.log(err);}

  db = client.db('toDoApp2nd');

  /\*db.collection('post').insertOne({\_id : 1, 이름:'홍길동',나이:30}, function(err1, result){

    console.log('저장완료');

  });\*/

  app.listen(8090, () => {

    console.log('listening on 8090');

  });

});

app.use(express.static('public'));

app.get('/', function(req, res){

  res.sendFile(\_\_dirname + '/index.html');

});

app.get('/write', function(req, res){

  res.sendFile(\_\_dirname + '/write.html');

});

app.use(express.urlencoded({extended : true})); // 2021년 이후 express 라이브러리는 body-parser 내포

app.post('/write', function(req, res){

  db.collection('counter').findOne({name:'현재시퀀스'}, function(err, result){

    // console.log(result);

    var post\_sq = result.post\_sq;

    db.collection('post').insertOne({\_id:post\_sq+1, title:req.body.title, date:req.body.date}, function(err, result){

      db.collection('counter').updateOne({name : '현재시퀀스'}, {$inc:{post\_sq:1}}, function(err, result){

        if(err) {return console.log(err);}

        console.log({\_id:post\_sq+1, title:req.body.title, date:req.body.date});

        console.log('저장완료');

        res.sendFile(\_\_dirname + '/write.html');

      });

    });

  });

});

8. EJS 라이브러리(HTML을 쓰기 쉽게 도와주는 템플릿 엔진)를 이용하여 /list 추가

(1) npm install ejs

(2) view 엔진으로 ejs를 쓴다고 index.js에 등록 후 /list 추가

// view engine을 ejs로 쓴다고 등록

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

app.get('/list', function(req, res){

  // 실제 DB에 저장된 데이터(post라는 DB안의 post collection 안의 모든 데이터)로 예쁘게 꾸며진 ejs를 보여줌

  db.collection('post').find().toArray(function(err, result){

    console.log(result);

    res.render("list.ejs", {posts : result});

  }); // 모든 데이터 가져오기

});

**[ views/list.ejs ]**

<!doctype html>

<html lang="en">

  <head>

    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <title>ToDo App</title>

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-KK94CHFLLe+nY2dmCWGMq91rCGa5gtU4mk92HdvYe+M/SXH301p5ILy+dN9+nJOZ" crossorigin="anonymous">

    <link href="/css/style.css" rel="stylesheet">

  </head>

  <body>

    <%- include('nav.html') %>

    <div class="container mt-5">

      <h2>서버에서 가져온 할일 리스트</h2>

        <% for(var i=0 ; i< posts.length ; i++) {%>

          <div class="container mt-4">

            <ul class="list-group">

              <li class="list-group-item">

                <p><%=posts[i].\_id %>번째.</p>

                <h4 class="detail" data-id="<%=posts[i].\_id %>">할일 제목 : <%=posts[i].title %></h4>

                <p class="detail" data-id="<%=posts[i].\_id %>">할일 마감날짜 : <%=posts[i].date %></p>

                <button class="btn btn-warning delete" data-id="<%=posts[i].\_id %>">삭제</button>

              </li>

            </ul>

          </div>

        <% } %>

    </div>

    <script src="https://code.jquery.com/jquery-3.6.4.min.js"></script>

    <script>

      $(document).ready(function(){

        $('.detail').click(function(e){

          var \_id = $(this).attr('data-id');

          // alert(\_id);

          location.href = '/detail/'+\_id;

        });

        $('button.delete').click(function(e){

          var \_id = e.target.dataset.id;

          var div = $(this).parent('li').parent('ul').parent('div');

          $.ajax({

            method : 'DELETE',

            url : '/delete',

            data : {\_id:\_id},

            success : function(result){

              alert(result.msg);

              div.fadeOut();

            }

          });

        });

      });

    </script>

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" integrity="sha384-ENjdO4Dr2bkBIFxQpeoTz1HIcje39Wm4jDKdf19U8gI4ddQ3GYNS7NTKfAdVQSZe" crossorigin="anonymous"></script>

  </body>

</html>

**[ views/nav.html ]**

<nav class="navbar navbar-expand-lg bg-body-tertiary">

  <div class="container-fluid">

    <a class="navbar-brand" href="/">ToDo App</a>

    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNavAltMarkup" aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-label="Toggle navigation">

      <span class="navbar-toggler-icon"></span>

    </button>

    <div class="collapse navbar-collapse" id="navbarNavAltMarkup">

      <div class="navbar-nav">

        <a class="nav-link active" aria-current="page" href="/">Home</a>

        <a class="nav-link" href="/write">Write</a>

        <a class="nav-link" href="/list">List</a>

      </div>

    </div>

  </div>

</nav>

9. /delete, /detail/:id, /update 추가

app.delete('/delete', function(req, res){

  console.log(req.body);

  // req.body에 담겨온 게시물번호를 가진 글을 db에서 찾아서 삭제해 주세요

  db.collection('post').deleteOne({\_id:Number(req.body.\_id)}, function(err, result){

    if(err) { return console.log(err); }

    console.log('삭제완료');

    res.status(200).send({msg : '삭제 성공했습니다'});

  });

});

app.get('/detail/:id', function(req, res){ // REST API 방식으로 파라미터 전달

  db.collection('post').findOne({\_id : parseInt(req.params.id)}, function(err, result){

    if(err) {

      return console.log(err);

    }

    console.log(result);

    res.render('detail.ejs', {post:result})

  });

});

app.get('/update/:id', function(req, res){ // REST API 방식으로 파라미터 전달

  db.collection('post').findOne({\_id : parseInt(req.params.id)}, function(err, result){

    if(err) {

      return console.log(err);

    }

    console.log(result);

    res.render('update.ejs', {post:result})

  });

});

app.post('/update', function(req, res){

  console.log(req.body);

  db.collection('post').updateOne({\_id:Number(req.body.\_id)},

                      {$set : {title : req.body.title,

                                date : req.body.date}}, function(err, result){

    if(err) {

      return console.log(err);

    }

    console.log('수정 완료');

    res.redirect('/detail/'+req.body.\_id);

  });

});