NODE.JS 網站設計

林新德

shinder.lin@gmail.com

參考專案：https://bitbucket.org/lsd0125/mfee36-node.git

1.1 什麼是 NODE.JS

- 2009年 **Ryan Dahl** 使用 **Chrome** 的 **JavaScript** 引擎（代號：**V8**），包裝成 **JavaScript** 執行環境（**Runtime**）**Node.js**。
- 可以在瀏覽器以外執行 **JavaScript**（像 **Python** 或 **Ruby**），讀寫檔案、寫服務程式、做資料庫連線等。
- 官網：https://nodejs.org/
- 安裝：至官網下載安裝檔。
- 安裝後，開啟命令提示列（**command prompt**、**terminal**）。
- 查看版本：> `node --version`
- 查看 **npm** 版本：> `npm -v`

1.2 建立專案

- 查看工作目錄內容 Windows：> `dir`
- 查看工作目錄內容 Mac：$ `ls -al`
- 建立資料夾：> `mkdir 資料夾名稱`
- 切換資料夾：> `cd 資料夾名稱`
- 到檔案管理員，點擊上方路徑，可以考備完整路徑。
- 使用 npm 建立專案套件管理檔案 package.json：
- > `npm init -y`
- 從 github 或 bitbucket 網站 clone 下來的專案，可以下式安裝 package.json 裡記錄的模組：
- > `npm install`

1.3 使用 ES6 (ES 2015)

- 全域安裝 **es-checker** 模組（套件）：
- > `npm install -g es-checker`
- > `sudo npm install -g es-checker` # mac 上全域安裝需要權限
- 測試環境（測試用，通常只使用一次）：> `es-checker`

- 查看所有全域套件：> `npm ls -g`
- 套件官網：**https://www.npmjs.org/**

1.4 箭頭函式

1. 專案目錄內建立 **src/** 資料夾。
2. 建立 **src/func.js** 內容如右。

```
const f1 = a=> a*a;
const f2 = ()=>{
    let sum = 0;
    for(let i=1; i<=10; i++) {
        sum += i;
    }
    return sum;
}
console.log(f1(6));
console.log(f2());
```

執行方式：> **node ./src/func.js**

1.5 CommonJS 的模組引入和匯出

```
// src/person.js
class Person {
    constructor(name='noname', age=20) {
        this.name = name;
        this.age = age;
    }
    toJSON(){
        const obj = {
            name: this.name,
            age: this.age,
        };
        return JSON.stringify(obj);
    }
}
module.exports = Person; // node 匯出類別
```

```
// src/person_test.js
const Person = require('./person');
const p1 = new Person('Bill', 26);
const p2 = new Person;
console.log(p1.toJSON());
console.log(p2.toJSON());
```

執行方式：> **node src/person_test.js**

CJS 和 MJS（ESM）

- **CJS** 為 **CommonJS**，在預設的 **Nodejs** 專案為 **CJS**，使用 **require()** 和 **module.exports** 進行模組的匯入和匯出。
- https://nodejs.org/docs/latest-v14.x/api/modules.html
- **MJS** 為使用 **ES6 module**，可以使用 **import** 和 **export** 關鍵字。
- 從 **Node12** 開始，加入的 **MJS** **實驗性質**的功能。
- 在 **package.json** 中加入屬性 "type": "module" 可以使整個專案為 **MJS**。
- 也可以使用**.mjs** 副檔名，指示檔案為 **ES6 modules**
- https://nodejs.org/docs/latest-v14.x/api/esm.html

ESM

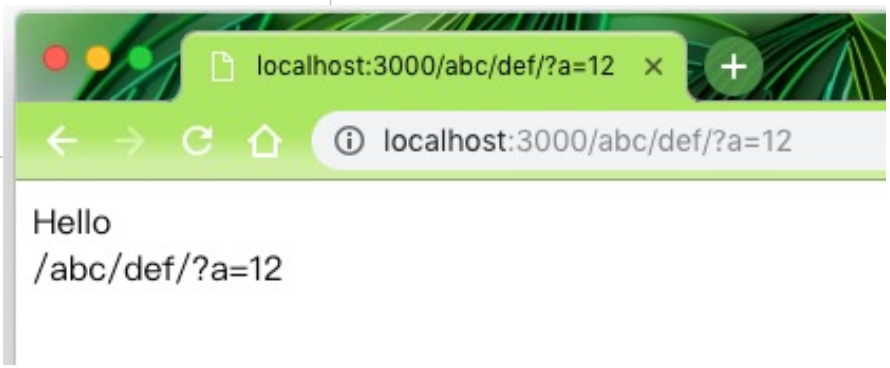
```
// func03.mjs
export const f1 = (a) => a * a;
const f3 = (a) => a * a * a;
export default f3;
```

```
// test-func03.mjs
import f3, {f1 as newName} from './func03.mjs';

console.log(newName(6));
console.log(f3(5));
```

1.6 簡易 WEB SERVER

```
// src/http_server.js
const http = require('http');
const server = http.createServer((req, res)=>{
    res.writeHead(200, {
        'Content-Type': 'text/html'
    });
    res.end(`<h2>hola 123</h2>
        <p>${req.url}</p>
    `);
});
server.listen(3000);
```



說明文件 https://nodejs.org/en/docs/

執行方式：> node src/http_server.js

Ctrl-C 停止 server

1.7 安裝 NODEMON 開發測試

- nodemon 會監看專案裡的檔案，有任何檔案變更，會重新啟動。
- 全域安裝 nodemon
- > `npm i -g nodemon`
- $ `sudo npm i -g nodemon`
- nodemon 的功能：專案中相關檔案修改時，會重新啟動程式（Server 程式）。

- 其它執行管理器：PM2 (http://pm2.keymetrics.io/)

1.8 讀寫檔案

```
// 注意非同步問題 (這是個錯誤的作法)
const http = require('http'),
        fs = require('fs');
http.createServer((request, response)=>{
    fs.writeFile(__dirname+'/header01.json', JSON.stringify(request.headers), error=>{
        if(error) return console.log(error);
        console.log('HTTP檔頭儲存');
    });
    fs.readFile(__dirname+'/data01.html', (error, data)=>{
       if(error) {
            response.writeHead(500, {'Content-Type': 'text/plain'});
            response.end('500 - data01.html not found');
        } else {
            response.writeHead(200, {'Content-Type': 'text/html'});
            response.end(data);
        }
    });
}).listen(3000);
```

執行方式：> **nodemon src/http_server2.js**

```
const http = require('node:http');
const fs = require('node:fs/promises');

const server = http.createServer(async (req, res)=>{
  res.writeHead(200, {
    'Content-Type': 'text/plain; charset=UTF-8'
  });
  const str = JSON.stringify(req.headers, null, 4);

  try {
    // __dirname: 目前 js 所在的資料夾
    await fs.writeFile(__dirname + '/headers01.txt', str);
  }catch(ex){
    return res.end('寫檔發生錯誤: ' + ex.toString());
  }
res.end(str);
});
server.listen(3000);
```

1.9 process.env 和 process.argv

- **process** 代表整個 **node** 執行的行程。
- **process.env** 可以取得作業系統的環境變數。
- **process.argv** 執行程式時，取得命令列參數。

process01.js

```
process.env.MY_PARAM = 'HELLO ENV';  // 直接設定
console.log('*** process.env ***', process.env);
console.log('*** process.args ***', process.argv);
```

```
// 執行
$ node src/process01.js aaa bbb --c
```

- 安裝 **dotenv** 套件，以載入 **.env** 檔案裡的設定。
- > **npm i dotenv**
- **.env** 檔不應該加入 **git**（版本控制）。
- **.env** 可以放在專案以外的路徑。

```
// require('dotenv').config(); // 使用專案的 .env

require('dotenv').config({
    path: '/Users/shinder/WebstormProjects/.env'
});
console.log(process.env.MY_USER);
```

2.1 安裝 EXPRESS

- 專案安裝 Express
- > `npm install --save express`
- > `npm i express`
- 查看 package.json 內容

- 建立主程式：index.js（沒有限定檔名）

2.2 EXPRESS 初體驗

```
// 1. 引入 express
const express = require('express');

// 2. 建立 web server 物件
const app = express();

// 3. 路由
app.get('/', function (req, res) {
    res.send('Hello World!');
});

// 4. Server 偵聽
app.listen(3000, function () {
    console.log('啟動 server 偵聽埠號 3000');
});
```

執行方式：> nodemon index.js

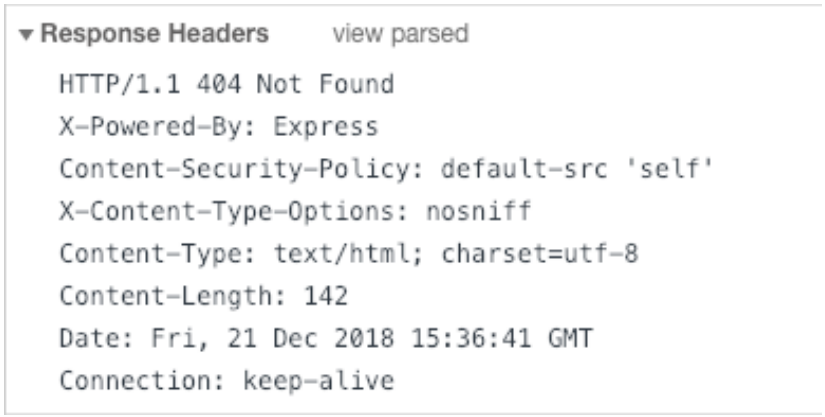
測試
http://localhost:3000/

2.3 自訂404頁面

```
// *** 此段放在所有路由設定的後面 ***

app.use((req, res) => {
    res.type('text/plain');
    res.status(404);
    res.send('404 - 找不到網頁');
});
```

測試
http://localhost:3000/abc




```
app.use((req, res) => {
  res.type('text/html').status(404).send(`<h1>找不到頁面</h1>`);
});
```

2.4 使用靜態內容的資料夾

- 專案內建立資料夾 **public/**
- 在裡面放 **a.html**
- 將下列程式，放在所有路由設定的前面

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

或

```
app.use(express.static(__dirname + '/public'));
```

- 使用瀏覽器查看 http://localhost:3000/a.html
- 靜態內容的資料夾可以設定多個，但「**請注意順序**」

2.5 使用 jQuery 和 Bootstrap

- 使用 **npm** 安裝 **jQuery** 和 **Bootstrap**，並設定靜態資料夾。

```
app.get(express.static('public'));
app.get(express.static('node_modules/bootstrap/dist'));
app.get(express.static('node_modules/jquery/dist'));
```

- 在 **a.html** 裡引入所需的 **css** 檔和 **js** 檔：

```
<link rel="stylesheet" href="/css/bootstrap.min.css" />
<script src="/jquery.min.js"></script>
<script src="/js/bootstrap.bundle.min.js"></script>
```

- 在 **a.html** 裡，放入 **bootstrap** 官網的 **navbar** 範例。

3.1 樣版引擎 EJS

- 官網：**https://ejs.co/**
- **ejs (Embedded JavaScript templating)** 套件位址：
- https://www.npmjs.com/package/ejs

- 使用樣版引擎的優點：可以把「呈現」和「邏輯處理」分開，易於管理。
- 安裝：> `npm i ejs`
- 並在專案中建立 **/views** 資料夾，做為存放樣版檔案的位置

3.2 設定 EJS

```
// 註冊樣版引擎
app.set('view engine', 'ejs');

// 設定views路徑（選擇性設定）
// app.set('views', __dirname + '/views');
```

3.3 EJS TAGS

說明網址：**https://ejs.co/#docs**

<%	'Scriptlet' tag, for control-flow, no output
<%_	'Whitespace Slurping' Scriptlet tag, strips all whitespace before it
<%=	Outputs the value into the template (HTML escaped)
<%-	Outputs the unescaped value into the template
<%#	Comment tag, no execution, no output
<%%	Outputs a literal '<%'
%>	Plain ending tag
-%>	Trim-mode ('newline slurp') tag, trims following newline
_%>	'Whitespace Slurping' ending tag, removes all whitespace after it

3.4 測試 EJS

修改 src/index.js ：

```
app.get('/', function (req, res) {
    res.render('home', {name: 'Shinder'});
});
```

views/home.ejs 的內容：

```
<h2><%= name %></h2>
```

3.5 使用 JSON 資料檔

```
[
  {
    "name": "Bill",
    "age": 28,
    "id": "A001"
  },
  {
    "name": "Peter",
    "age": 32,
    "id": "A002"
  },
  {
    "name": "Carl",
    "age": 29,
    "id": "A003"
  }
]
```

```
// index.js 加入路由
app.get('/sales', function (req, res) {
    const data = require(__dirname + '/data/sales');
    res.render('sales', {
        sales: data
    });
});
```

3.6 以表格呈現 JSON 裡的資料

views/sales.ejs

```
<table class="table table-striped">
    <thead>
    <tr>
        <th scope="col">員工編號</th>
        <th scope="col">姓名</th>
        <th scope="col">年齡</th>
    </tr>
    </thead>
    <tbody>
     <% for(let s of sales){ %>
         <tr>
             <td><%= s.id %></td>
             <td><%= s.name %></td>
             <td><%= s.age %></td>
         </tr>
     <% } %>
    </tbody>
</table>
```

3.7 使用 EJS 的 include()

建立 **views/parts/navbar.ejs** 檔，並將 **navbar** 的部份放入。

再修改 **sales.ejs**：

```
<%- include('parts/html-head') %>
<%- include('parts/navbar') %>
<div class="container">
    <table class="table table-striped">
       ...
```

<%- 用來避免 **HTML** 跳脫

路徑為相對路徑

.ejs 副檔名可以省略

4.1 取得 queryString 資料

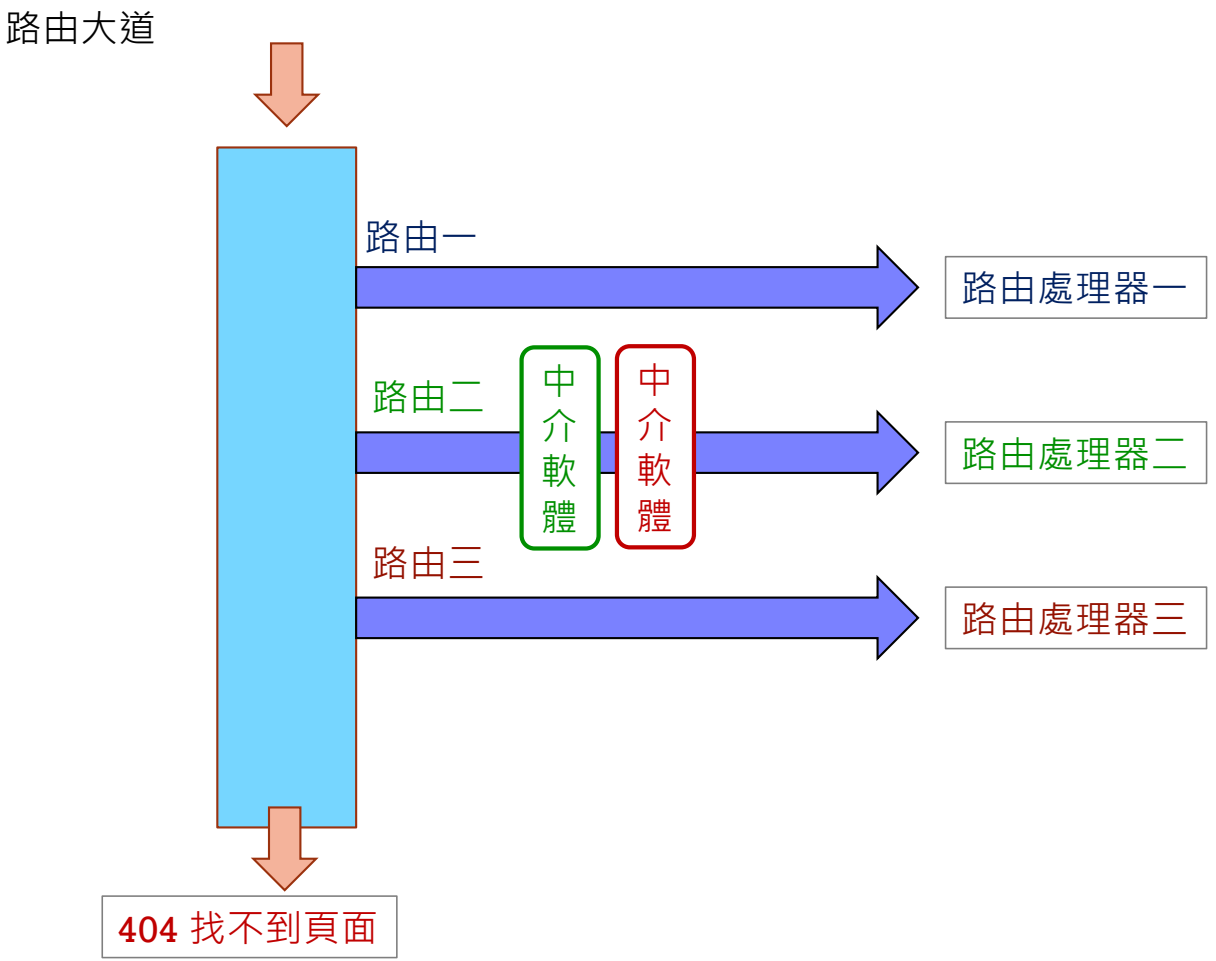
可以透過 req.query.名稱 取得，例如：req.query.a

```
http://localhost:3000/try-qs?a=1&b=3
http://localhost:3000/try-qs?a[]=2&a[]=bill
http://localhost:3000/try-qs?a=2&a=bill
http://localhost:3000/try-qs?a[age]=20&a[name]=bill
```

4.2 取得 POST 資料

- 使用 **express** 物件的 **body-parser** 功能

```
// 取得 urlencoded parser, 不使用 qs lib, 而使用內建的 querystring lib
const urlencodedParser = express.urlencoded({extended: false});
app.get('/try-post-form', (req, res) => {
    res.render('try-post-form');
});
// 把 urlencodedParser 當 middleware
app.post('/try-post-form', urlencodedParser, (req, res) => {
    res.render('try-post-form', req.body);
});
```


路由大道
路由一
路由處理器一
路由二
中介軟體
中介軟體
路由處理器二
路由三
路由處理器三
404 找不到頁面

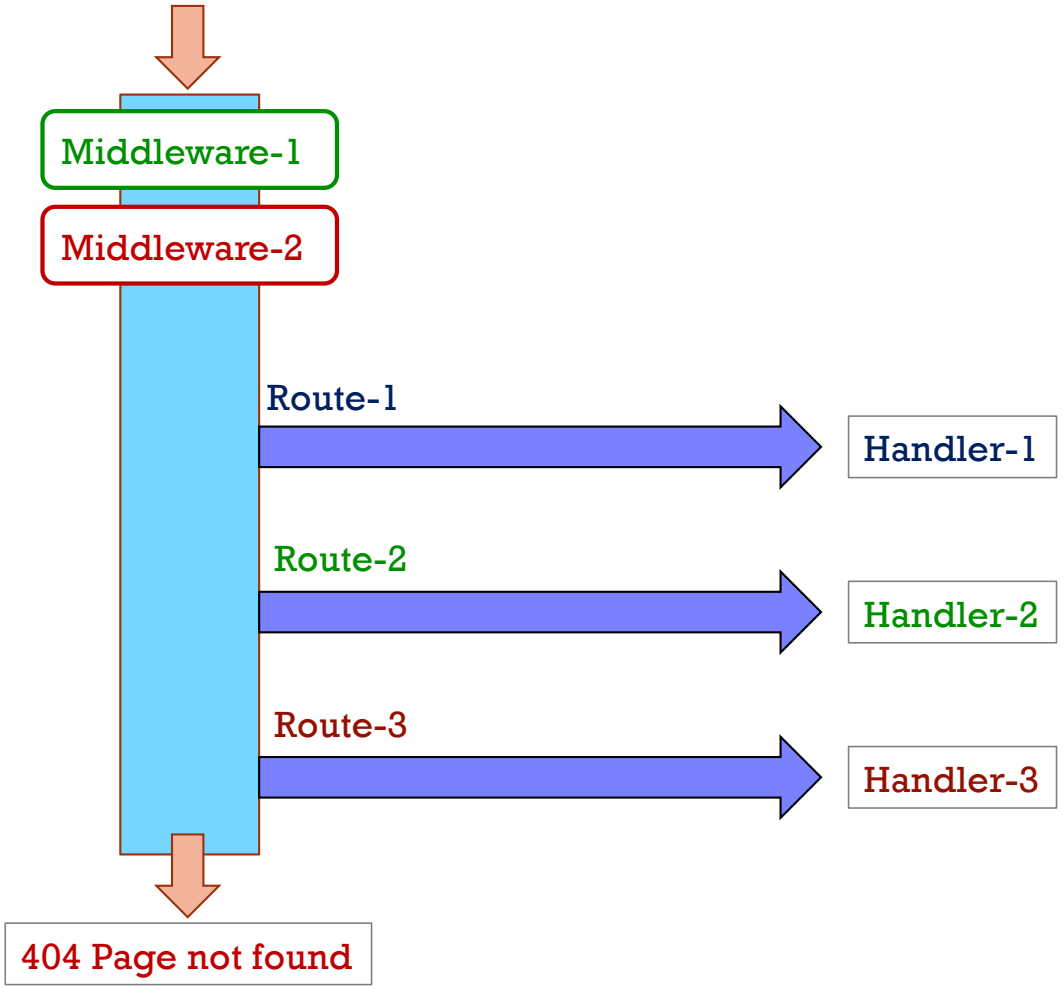
```
<%# views/try-post-form.ejs %>
<div class="col-lg-6">
    <form method="post" enctype="application/x-www-form-urlencoded">
        <div class="form-group">
            <label for="email">Email</label>
            <input type="email" class="form-control" name="email">
            <% if(typeof email !== 'undefined'){ %>
                <small>上次輸入: <%= email %></small>
            <% } %>
        </div>
        <div class="form-group">
            <label for="password">Password</label>
            <input type="text" class="form-control" name="password">
            <% if(locals.password){ %>
                <small>上次輸入: <%= password %></small>
            <% } %>
        </div>
        <button type="submit" class="btn btn-primary">Submit</button>
    </form>
</div>
```

4.3 Top-level Middleware

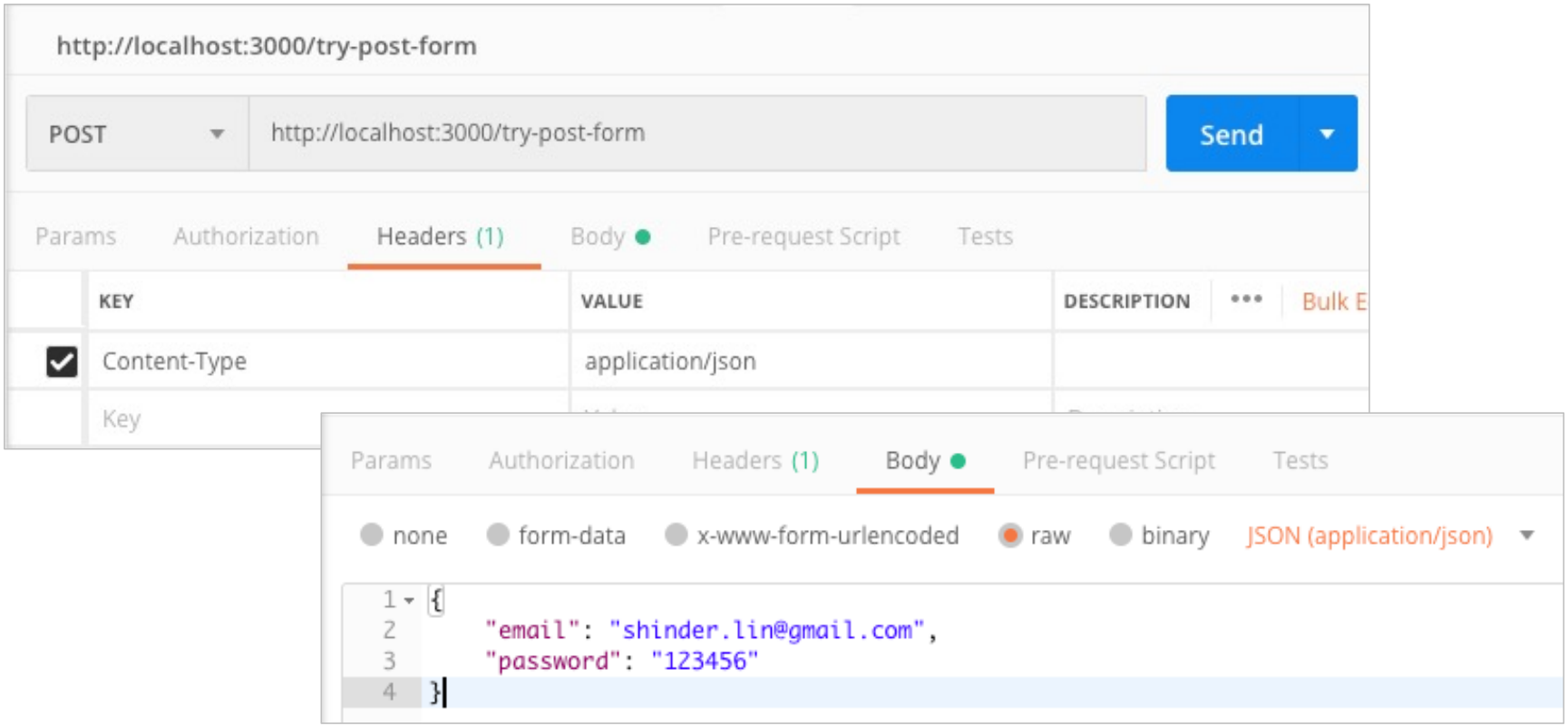
- 將 **body-parser** 設定成頂層 **middleware**，放在所有路由之前。
- 其包含兩種解析功能： **urlencoded** 和 **json** 。

```
// parse application/x-www-form-urlencoded
app.use(express.urlencoded({ extended: false }));

// parse application/json
app.use(express.json());
```


Middleware-1
Middleware-2
Route-1
Handler-1
Route-2
Handler-2
Route-3
Handler-3
404 Page not found

- 使用 **postman** 測試 **json** 格式。



- 使用 **VSCode** 外掛 **REST Client** 測試後端功能。
- **REST Client** 使用 ***.rest** 文字檔，做為測試設定。

```
GET HTTP://localhost:3000/


### 分隔線
POST HTTP://localhost:3000/try-post
Content-Type: application/json

{
    "name": "shinder"
}
```

4.4 使用 Multer 處理檔案上傳

- 使用 multer
- 安裝: > `npm i multer`
- 說明可參考 multer 的npmjs 主頁
- https://www.npmjs.com/package/multer
- 建立 tmp_uploads 做為檔案上傳的暫存資料夾（名稱可自訂）
- 建立 public/img 做為存放圖檔的資料夾（名稱可自訂）

```
<%# views/try-upload.ejs %>
<% if(locals.result){ %>
    <div class="card" style="width: 18rem;">
        <img src="<%= avatar %>" class="card-img-top" alt="">
        <div class="card-body">
            <h5 class="card-title"><%= name %></h5>
        </div>
    </div>
<% } %>
<div class="col-lg-6">
    <form method="post" enctype="multipart/form-data">
        <div class="form-group">
            <label>姓名</label>
            <input type="text" class="form-control" name="name">
        </div>
        <div class="form-group">
            <label>大頭貼</label>
            <input type="file" class="form-control" name="avatar">
        </div>
        <button type="submit" class="btn btn-primary">送出</button>
    </form>
</div>
```

```
const multer = require('multer');
const upload = multer({dest:'tmp_uploads/'}); // 設定上傳暫存目錄
const fs = require('fs'); // 處理檔案的核心套件
app.get('/try-upload', (req, res)=>{
    res.render('try-upload');
});
app.post('/try-upload', upload.single('avatar'), (req, res)=>{
    console.log(req.file); // 查看裡面的屬性
    if(req.file && req.file.originalname){
        // 判斷是否為圖檔
        if(/\.(jpg|jpeg|png|gif)$/i.test(req.file.originalname)){
            // 將檔案搬至公開的資料夾
            fs.rename(req.file.path, './public/img/' + req.file.originalname, error=>{});
        } else {
            fs.unlink(req.file.path, error=>{}); // 刪除暫存檔
        }
    }
    res.render('try-upload', {
        result: true,
        name: req.body.name,
        avatar: '/img/' + req.file.originalname
    });
});
```

```
console.log(req.file); //結果
```


```
{
  "fieldname": "avatar",
  "originalname": "test00.png",
  "encoding": "7bit",
  "mimetype": "image/png",
  "destination": "tmp_uploads/",
  "filename": "25e3c4de203391f7dc8bfce9360002b0",
  "path": "tmp_uploads/25e3c4de203391f7dc8bfce9360002b0",
  "size": 21634
}
```


- 4.4.1 multer 使用 storage 和 fileFilter
- upload-module.js

```
const multer = require('multer');
const {v4: uuidv4} = require('uuid');

const extMap = {
    'image/jpeg': '.jpg',
    'image/png':  '.png',
    'image/gif':  '.gif',
};
```

```
const storage = multer.diskStorage({
    destination : (req, file, cb)=>{
        cb(null, __dirname + '/public/img-uploads')
    },
    filename: (req, file, cb)=>{
        let ext = extMap[file.mimetype];
        cb(null, uuidv4() + ext)
    }
});
const fileFilter = (req, file, cb)=>{
    cb(null, !!extMap[file.mimetype]);
};

const upload = multer({storage, fileFilter});
module.exports = upload;
```

4.5 Router 處理

- 使用變數代稱設定路由
- 使用 **regular expression** 設定路由
- 路由模組化

- **4.5.1** 使用變數代稱設定路由
- **:** 冒號之後為代稱名
- **?** 為選擇性的
- ***** 為 **wildcard**

```
app.get('/my-params1/:action/:id', (req, res)=>{
    res.json(req.params);
});

app.get('/my-params2/:action?/:id?', (req, res)=>{
    res.json(req.params);
});

app.get('/my-params3/*/*?', (req, res)=>{
    res.json(req.params);
});
```

- **4.5.2 使用 regular expression 設定路由**

```
app.get(/^\/hi\/?/, (req, res)=>{
    let result = {
        url : req.url
    };
    result.split = req.url.split('/');
    res.json(result);
});
```

測試以下的 url:
http://localhost:3000/hi
http://localhost:3000/hi/
http://localhost:3000/hi/123
http://localhost:3000/hi123

```
// 手機號碼
app.get(/^\/09\d{2}\-?\d{3}\-?\d{3}$/, (req, res)=>{
    let u = req.url.slice(1);
    u = u.split('?')[0];
    u = u.split('-').join('');
    res.send(u);
});
```

- **4.5.3 路由模組化 (方式一)**

```
// routes/admin1.js
module.exports = app => {
    app.get('/admin1/:p1?/:p2?', (req, res)=>{
        res.json(req.params);
    });
};
```

```
// 在 index.js 內加入
const admin1 = require(__dirname + '/routes/admin1');
admin1(app);
```

- **4.5.4** 路由模組化 (方式二)

```
// routes/admin2.js
const express = require('express');
const router = express.Router();
router.get('/admin2/:p1?/:p2?', (req, res)=>{
    res.json(req.params);
});
module.exports = router;
```

```
// 在 index.js 內加入
const admin2Router = require(__dirname + '/routes/admin2');
app.use(admin2Router); //當成 middleware 使用
```

- **4.5.5** 路由模組化 (方式三)

```
// 在 index.js 內加入
const admin3Router = require(__dirname + '/routes/admin3');
app.use('/admin3', admin3Router); // module 裡面的路徑用相對路徑
```

```
// admins/admin3.js
const express = require('express');
const router = express.Router();
router.route('/member/edit/:id')
    .all((req, res, next)=>{
        // 找到該會員資料
        res.locals.memberData = {
            name: 'shinder',
            id: 'A002'
        };
        next();
    })
    .get((req, res)=>{
        const obj = {
            baseUrl: req.baseUrl, // 查看基底 url
            url: req.url,
            data: res.locals.memberData
        };
        res.send('get edit:' + JSON.stringify(obj));
    })
    .post((req, res)=>{
        res.send('post edit:' + JSON.stringify(res.locals.memberData));
    });
module.exports = router;
```

5.1 Session

- 若 Client 的瀏覽器停在某個網頁，使用者可能某些原因久久未再拜訪該網站，或者根本就已離開該站。此時會依 Session 的存活時間，決定 Session 是否有效。
- Server 是以 Client 最後一次拜訪開始重新計時的，若 Client 在 Session 存活時間內，持續訪問該站，Session 就會一直有效。
- 利用 Cookie 存放「Session ID」，在 Client 第一次拜訪時將 Session ID 存入 Cookie。
- 有了 Session ID 之後，Server 會在主機（記憶體、檔案或資料庫）為每個 Session ID 建立一個對應的 Session 物件，資料就存在 Session 物件裡。

- 安裝 express-session
- > `npm i express-session`

- 範例：顯示頁面刷新次數

```
const session = require('express-session');
app.use(session({
    // 新用戶沒有使用到 session 物件時不會建立 session 和發送 cookie
    saveUninitialized: false,
    resave: false, // 沒變更內容是否強制回存
    secret: '加密用的字串',
    cookie: {
        maxAge: 1200000, // 20分鐘，單位毫秒
    }
}));
```

```
app.get('/try-session', (req, res)=>{
    req.session.my_var = req.session.my_var || 0; // 預設為 0
    req.session.my_var++;
    res.json({
        my_var: req.session.my_var,
        session: req.session
    });
});
```

- 登入表單

```
<%# views/login.ejs %>
<% if(locals.flashMsg) { %>
    <div class="alert alert-danger" role="alert">
        <%= flashMsg %>
    </div>
<% } %>
<div class="row">
    <div class="col-md-6">
        <% if(locals.isLogined) { %>
            <div class="alert alert-primary" role="alert">
                <%= loginUser %> 您好 <a href="/logout">登出</a>
            </div>
        <% } else { %>
            <form method="post">
                <div class="form-group">
                    <label for="user">帳號</label>
                    <input type="text" class="form-control" name="user">
                </div>
                <div class="form-group">
                    <label for="password">密碼</label>
                    <input type="text" class="form-control" name="password">
                </div>
                <button type="submit" class="btn btn-primary">登入</button>
            </form>
        <% } %>
    </div>
</div>
```

- 登入的表單：**src/index.js**

```
app.get('/login', (req, res)=>{
    let data = {
        flashMsg: req.session.flashMsg || '', // 快閃訊息
        loginUser: req.session.loginUser
    };
    delete req.session.flashMsg; // 移除快閃訊息
    res.render('login', data);
});
```

- 登出表單：**index.js**

```
app.get('/logout', (req, res)=>{
    delete req.session.loginUser;
    res.redirect('/login');
});
```

```
app.post('/login', (req, res)=>{
    // SELECT * FROM `members` WHERE `email`=? AND `password`=SHA1(?)
    const list = {
      shin: '1234',
      der: '1234',
    };

    if(
        req.body.user &&
        list[req.body.user] &&
        list[req.body.user]===req.body.password
    ){
        req.session.loginUser = req.body.user;
    } else {
        req.session.flashMsg = '密碼錯誤';
    }
    res.redirect('/login');
});
```

5.2 時間格式

- 使用 moment.js
- 官網： https://momentjs.com
- 說明文件：https://momentjs.com/docs
- 安裝： > `npm i moment`

- 若需要時區的功能，請使用 moment-timezone
- 安裝： > `npm i moment-timezone`

時間格式化輸出

```
// index.js
const moment = require('moment-timezone');

app.get('/try-moment', (req, res)=>{
    const fm = 'YYYY-MM-DD HH:mm:ss';
    const mo1 = moment(req.session.cookie.expires);
    const mo2 = moment(new Date());

    res.json({
        'local-mo1': mo1.format(fm),
        'local-mo2': mo2.format(fm),
        'london-mo1': mo1.tz('Europe/London').format(fm),
        'london-mo2': mo2.tz('Europe/London').format(fm),
    });
});
```

6.1 連線MySQL

- 預先安裝 MySQL 資料庫管理系統
- 可安裝 MAMP（Apache, MySQL, PHP）開發環境
- 使用 node 的 mysql2 套件連線
- 安裝：> `npm i mysql2`

- 在 **test** 資料庫，建立資料表輸入資料

```
CREATE TABLE `address_book` (
  `sid` int(11) NOT NULL,
  `name` varchar(255) NOT NULL,
  `email` varchar(255) NOT NULL,
  `mobile` varchar(255) NOT NULL,
  `birthday` date NOT NULL,
  `address` varchar(255) NOT NULL,
  `created_at` datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `address_book`
(`sid`, `name`, `email`, `mobile`, `birthday`, `address`, `created_at`) VALUES
(1, '李小明', 'ming01@gmail.com', '0918555666', '1995-10-02', '台南市', '2020-03-24 09:30:37'),
(2, '李小明2', 'ming01@gmail.com', '0918555666', '1995-10-02', '台南市', '2020-03-24 09:30:37'),
(3, '李小明3', 'ming01@gmail.com', '0918555666', '1995-10-02', '台南市', '2020-03-24 09:30:37');

ALTER TABLE `address_book`
  ADD PRIMARY KEY (`sid`);
ALTER TABLE `address_book`
  MODIFY `sid` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=5;
```

- 呈現的頁面

```
<%# views/address-book/list.ejs %>
<% for(let item of rows){ %>
<tr>
    <td>
        <a href="/address-book/edit/<%= item.sid %>">
            <i class="fas fa-edit"></i>
        </a>
    </td>
    <td><%= item.sid %></td>
    <td><%= item.name %></td>
    <td><%= item.email %></td>
    <td><%= item.mobile %></td>
    <td><%= item.birthday %></td>
    <td><%= item.address %></td>
    <td>
        <a class="del-a-tag" href="/address-book/delete/<%= item.sid %>"
           data-sid="<%= item.sid %>">
            <i class="fas fa-trash-alt"></i>
        </a>
    </td>
</tr>
<% } %>
```

- 連線模組

```
const mysql = require("mysql2/promise");

const {DB_HOST, DB_USER, DB_PASS, DB_NAME } = process.env;

console.log({DB_HOST, DB_USER, DB_PASS, DB_NAME });

const pool = mysql.createPool({
  host: DB_HOST,
  user: DB_USER,
  password: DB_PASS,
  database: DB_NAME,

  waitForConnections: true,
  connectionLimit: 3,  // 最大連線數
  queueLimit: 0,
})
module.exports = pool;
```

- 讀取資料

```
const t_sql = `SELECT COUNT(1) totalRows FROM address_book ${where}`;
const [[{totalRows}]] = await db.query(t_sql);
let totalPages = 0;
let rows = [];
if(totalRows){
  totalPages = Math.ceil(totalRows/perPage);
  if(page > totalPages) {
    output.redirect = req.baseUrl + '?page=' + totalPages;
    return output;
  };
  const sql = `SELECT * FROM address_book ${where} LIMIT ${perPage*(page-1)}, ${perPage}`;
  [rows] = await db.query(sql);

}
output = {...output, totalRows, perPage, totalPages, page, rows, keyword};
```

6.2 將 Session 資料存入 MySQL

- 安裝 express-mysql-session 套件

```
const session = require('express-session');
const MysqlStore = require('express-mysql-session')(session);
const db = require(__dirname + '/db_connect2');
const sessionStore = new MysqlStore({}, db);

app.use(session({
    saveUninitialized: false,
    resave: false,
    secret: '加密的文字',
    store: sessionStore,
    cookie: {
        maxAge: 1200000
    }
}));
```

6.3 新增資料

```
<%# views/address-book/add.ejs 中的表單（部份） %>
<h5 class="card-title">新增通訊錄資料</h5>
<form name="form1" onsubmit="return formCheck()">
    <div class="form-group">
        <label for="name">姓名</label>
        <input type="text" class="form-control" id="name" name="name">
    </div>
    <div class="form-group">
        <label for="email">電子郵箱</label>
        <input type="text" class="form-control" id="email" name="email">
    </div>
    <div class="form-group">
        <label for="mobile">手機</label>
        <input type="text" class="form-control" id="mobile" name="mobile">
    </div>
        ...
```

```
// views/address-book/add.ejs 中的發 AJAX 片段
if(isPass){
    const fd = new FormData(document.form1);
    fetch('/address-book/add', {
        method: 'POST',
        body: fd
    })
        .then(r=>r.json())
        .then(obj=>{
            if(obj.success){
                infoBar.removeClass('alert-danger')
                    .addClass('alert-success')
                    .text('新增成功');
            } else {
                infoBar.removeClass('alert-success')
                    .addClass('alert-danger')
                    .text(obj.error);
            }
            infoBar.show();
        });
}
```

```
// 後端路由
router.get('/add',(req, res)=>{
    res.render('address-book/add');
});
// upload.none() 用來解析 multipart/form-data 格式的 middleware
router.post('/add', upload.none(), (req, res)=>{
    const output = {
        success: false,
        error: '',
    };
    req.body.created_at = new Date();
    const sql = "INSERT INTO `address_book` SET ?";
    db.query(sql, [req.body])
        .then(([results])=>{
            output.results = results;
            if(results.affectedRows===1){
                output.success = true;
            }
            res.json(output);
        })
        .catch(ex=>{
            console.log('ex:', ex);
        })
});
```

6.4 完成 CRUD

- 試著完成修改資料及刪除資料的功能
- 依相同的方式完成管理者資料的新增
- 參考 https://bitbucket.org/lsd0125/aien06-nodejs/src/master/src/proj.js

- 管理者登入功能
- 登入後才能編輯員工資料

7.1 使用 CORS

- 跨來源資源共用（Cross-Origin Resource Sharing (CORS)）是一種使用額外 HTTP 標頭令目前瀏覽網站的使用者代理取得存取其他來源（網域）伺服器特定資源權限的機制。
- https://developer.mozilla.org/zh-TW/docs/Web/HTTP/CORS
- 當使用fetch() 或傳統 AJAX 跨源（cross origin）去呼叫 API 時，需要對方主機允許。
- npmjs主頁：https://www.npmjs.com/package/cors
- 安裝：> `npm i cors`

- 一般的使用方式（不需使用**cookies** 和**session**時）

```
const cors = require('cors');
app.use(cors());
```

- 需要使用**cookies** 和**session**時（使用白名單）

```
const cors = require('cors');
var whitelist = ['http://localhost:8080', undefined, 'http://localhost:3000'];
var corsOptions = {
    credentials: true,
    origin: function (origin, callback) {
        console.log('origin: '+origin);
        if (whitelist.indexOf(origin) !== -1) {
            callback(null, true)
        } else {
            callback(new Error('Not allowed by CORS'))
        }
    }
};
app.use(cors(corsOptions));
```

- 準備後端服務

```
app.post('/try-session2', (req, res)=>{
    req.session.views = req.session.views || 0;
    req.session.views ++;
    res.json({
        views: req.session.views
    })
});
```

- 前端 JS

```
<!-- public/cors-session.html -->
<script>
    function doPost(){
        var data = {
            name: document.form1.name.value,
            age: document.form1.age.value
        };
        fetch('//localhost:3000/try-session2', {
            method: 'POST',
            body: JSON.stringify(data),
            credentials: 'include', // cross origin 傳送 cookie
            headers: {
                'Content-Type': 'application/json'
            }
        })
            .then(response=>{
                return response.json();
            })
            .then(obj=>{
                $('#output').val( JSON.stringify(obj))
            });
    }
</script>
```

加油！