

林新德

[shinder.lin@gmail.com](mailto:shinder.lin@gmail.com)

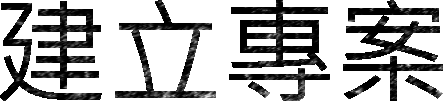
參考專案：https://bitbucket.org/lsd0125/mfee05-nodejs.git

# 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

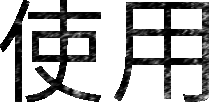
 

# 查看工作目錄內容Windows：> **dir**

* 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。

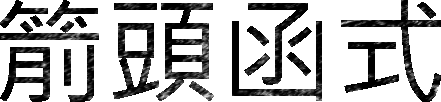
* 使用 npm 建立專案套件管理檔案 package.json：
* > **npm init -y**
* 從 github 或 bitbucket 網站 clone 下來的專案，可以下式安裝 package.json 裡記錄的模組：
* > **npm install**

# 全域安裝 es-checker 模組（套件）：

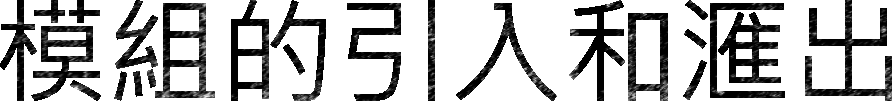
* > **npm install -g es-checker**
* > **sudo npm install -g es-checker** # mac 上全域安裝需要權限
* 測試環境（測試用，通常只使用一次）： > **es-checker**
* 查看所有全域套件：> **npm ls –g**

# 套件官網：https://[www.npmjs.org/](http://www.npmjs.org/)

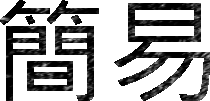
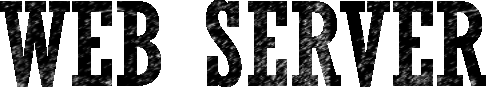
 

|  |  |
| --- | --- |
| 1. 專案目錄內建立 **src/** 資料夾。 2. 建立 **src/func.js** 內容如右。 | **let** func = a => a\*a; // 單 行  **let** func2 = () => {  **let** r = 0;  **for**(**let** i=1; i<=10; i++){ r += i;  }  **return** r;  };  console.log( func(5) ); console.log( func2() ); |
|  |

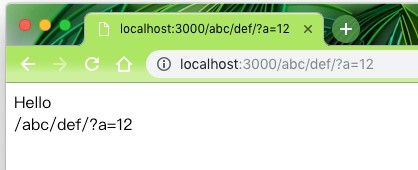
執行方式：> **node ./src/func.js**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| // src/person.js |  |  |  | // src/person\_test.js |
| **class** Person { |  |  |  | **const** Person = require('./person'); |
| constructor(name=‘noname’, | age=20) | { |  | **const** p1 = **new** Person('Bill', 25); |
| **this**.name = name; |  |  |  | **const** p2 = **new** Person; |
| **this**.age = age; |  |  |  | console.log(p1.toJSON()); |
| } |  |  |  | console.log(p2.toJSON()); |
| toJSON() { |  |  |  |  |
| **const** obj = {  name: **this**.name, age: **this**.age  };  **return** JSON.stringify(obj);  }  }  module.exports = Person; // node 匯出類別 | | |  | 執行方式：> **node src/person\_test.js** |

// http\_server.js



**const** http = require('http');

**const** server = http.createServer((request, response)=>{ response.writeHead(200, {

'Content-Type': 'text/html'

});

});

response.end(`<div>Hello<br>

${request.url}

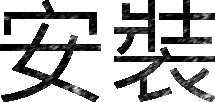
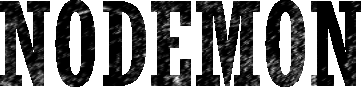
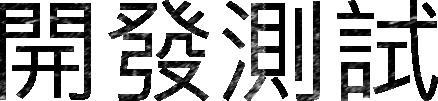
</div>`);

server.listen(3000);

說明文件 https://nodejs.org/en/docs/

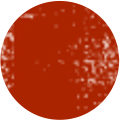
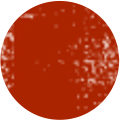
執行方式：> **node src/http\_server.js**

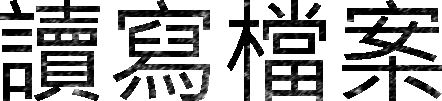
Ctrl-C 停止 server

# 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 server。
* 其它執行管理器：PM2 (<http://pm2.keymetrics.io/)>



// http\_server2.js

**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** {

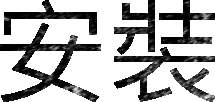
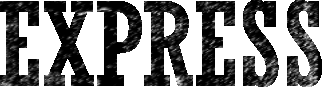
response.writeHead(200, {'Content-Type': 'text/html'}); response.end(data);

}

});

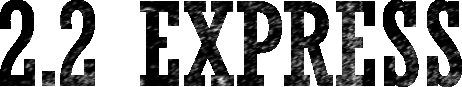
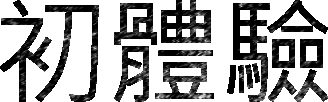
}).listen(3000);

執行方式：> **nodemon src/http\_server2.js**

# 專案安裝 Express

* > **npm install --save express**
* > **npm i express**
* 查看 package.json 內容
* 建立主檔：index.js （沒有限定檔名）

# 執行方式：> nodemon src/index.js

// 1. 引 入 express

**var** express = require('express');

// 2. 建 立

**var** app =

web

server 物 件

express();

// 3. 路 由

app.get('/', **function**(request, response){

response.send('Hello World!');

});

// 4. Server 偵 聽

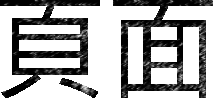
app.listen(3000, **function**(){

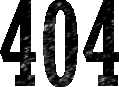
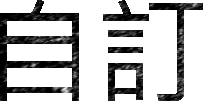
console.log('啟動 server 偵聽埠號 3000');

});

測試

http://localhost:3000/



測試

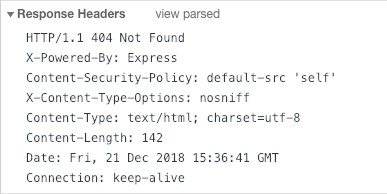
// **\*\*\*** 此段放在所有路由設定的後面 **\*\*\***

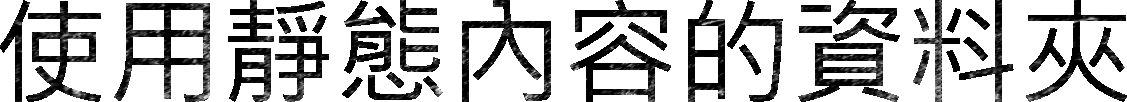
app.use((req, res)=>{ res.type('text/plain'); res.status(404);

res.send('404 - 找不到網頁');

});

http://localhost:3000/abc



# 專案內建立資料夾 public/

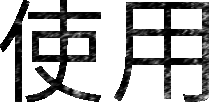
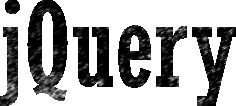
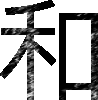
* 在裡面放 a.html
* 將下列程式，放在所有路由設定的前面

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」

* jQuery 官 網 <http://jquery.com/>
* 下載 jquery-3.4.1.js，放入 public/js/ 裡。
* Bootstrap 官網 https://getbootstrap.com/
* 下載編譯好的檔案，放入 public/bootstrap/
* 在 a.html 裡引入所需的 css 檔和 js 檔，順序為：

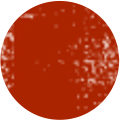
<link rel="stylesheet" href=”**/**bootstrap/css/bootstrap.css">

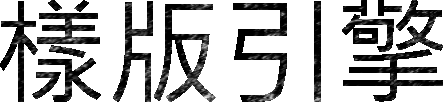
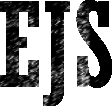
<script src="**/**js/jquery-3.4.1.js"></script>

<script src="**/**bootstrap/js/bootstrap.js"></script>

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

* https://getbootstrap.com/docs/4.3/components/navbar/

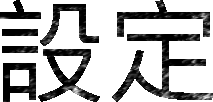
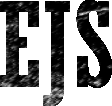


# 官網：https://ejs.co/

* ejs (Embedded JavaScript templating) 套件位址：
* https://[www.npmjs.com/package/ejs](http://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

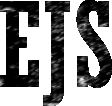
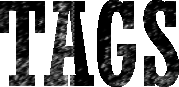
  

// 註冊樣版引擎

app.set('view engine', 'ejs’);

// 設定views路徑 (選擇性設定)

// app.set('views', \_\_dirname + '/../views');

說明網址：https://ejs.co/#docs

## <%

<%\_

## <%=

**<%-**

<%#

<%%

## %>

-%>

\_%>

'Scriptlet' tag, for control-flow, no output

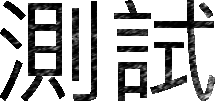
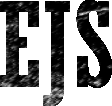
‘Whitespace Slurping’ Scriptlet tag, strips all whitespace before it

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

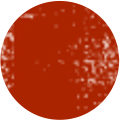
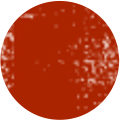
修改 src/index.js ：

app.get('/', function (request, response) { response.render('home', {name: 'Shinder'});

});

views/home.ejs 的內容：

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



// src/index.js 加入路由

app.get('/sales', function (request, response) {

const data = require(\_\_dirname + '/../data/sales'); response.render('sales', {

sales: data

});

});

[

{

"name": "Bill", "age": 28,

"id": "A001"

},

{

"name": "Peter", "age": 32,

"id": "A002"

},

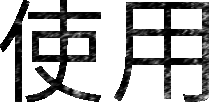
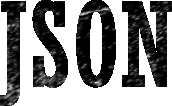
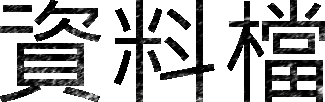
{

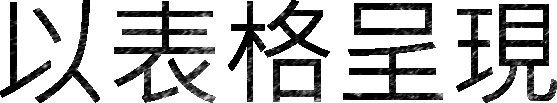
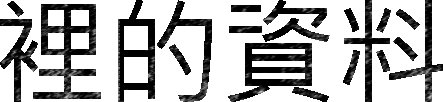
"name": "Carl", "age": 29,

"id": "A003"

}

]

<table class="table">

<thead>

<tr>

<th scope="col">#</th>

<th scope="col">Name</th>

<th scope="col">Age</th>

</tr>

</thead>

<tbody>

<% for(let i=0;

<tr>

<td><%=

<td><%=

<td><%=

</tr>

<% } %>

</tbody>

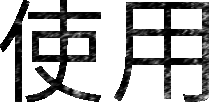
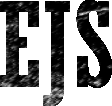
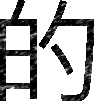
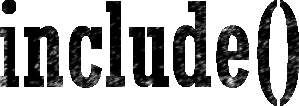
</table>

i<sales.length; i++) { %>

sales[i].id %></td>

sales[i].name %></td> sales[i].age %></td>

views/sales.ejs

建立 views/parts/navbar.ejs 檔，並將 navbar 的部份放入。再修改 sales.ejs：

.ejs 副檔名可以省略

路徑為相對路徑

<%- 用來避免 HTML 跳脫

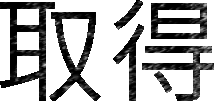
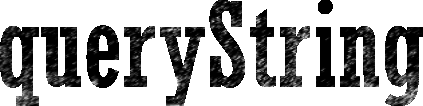
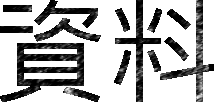
<body>

<%- include('parts/navbar') %>

<div class="container">

<table class="table">

...

// src/index.js 加入路由

// 引入核心套件 url

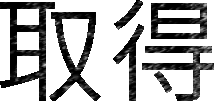
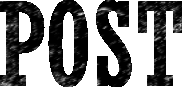
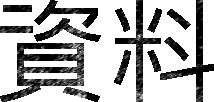
const url = require('url');

app.get(**'/try-qs'**, (req, res)=>{

**const** urlParts = url.parse(req.**url**, **true**); console.log(urlParts); // 在 server 端 查 看res.json(urlParts);

});

http://localhost:3000/try-qs?a=1&b=3

# 使用 body-parser

* 安裝: > **npm i body-parser**

// 引入套件

const bodyParser = require('body-parser');

// 取得 urlencoded parser, 不使用 qs lib, 而使用內建的 querystring lib

const urlencodedParser = bodyParser.urlencoded({extended: false});

app.get('/try-post-form', (req, res) => { res.render('try-post-form');

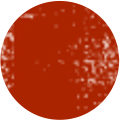
});

// 把 urlencodedParser 當 middleware

app.post('/try-post-form', urlencodedParser, (req, res) => {

res.render('try-post-form', req.body);

});



<%# 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>

<% } %>

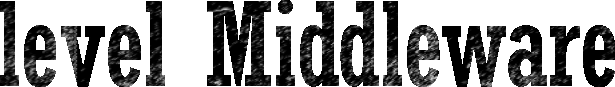
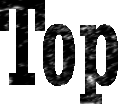
</div>

<button type="submit" class="btn btn-primary">Submit</button>

</form>

</div>





# 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 。

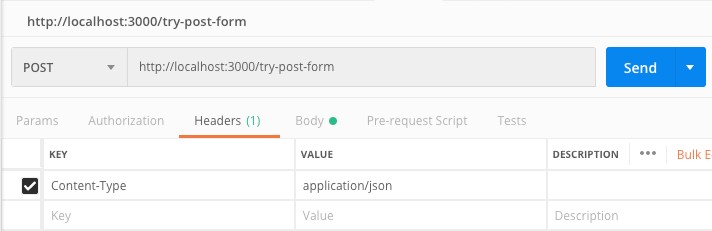
// 引入套件

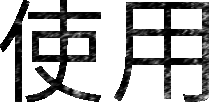
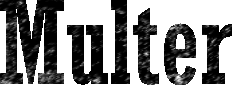
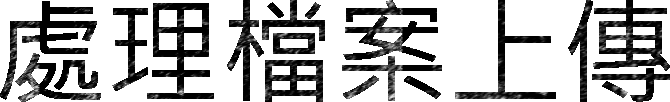
**const** bodyParser = require('body-parser');

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

// parse application/json app.use(bodyParser.json());

* 使用 postman 測試 json 格式。



# 使用 multer

* + 安裝: > **npm i multer**

# 說明可參考 multer 的npmjs 主頁

* + https://[www.npmjs.com/package/multer](http://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

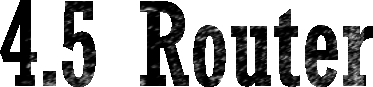
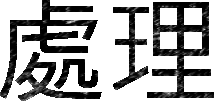
{

"fieldname": "avatar", "originalname": "test00.png", "encoding": "7bit", "mimetype": "image/png", "destination": "tmp\_uploads/",

"filename": "25e3c4de203391f7dc8bfce9360002b0",

"path": "tmp\_uploads/25e3c4de203391f7dc8bfce9360002b0", "size": 21634

}

# 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

測 試 以 下 的 url: http://localhost:3000/hi

app.get(/^\/hi\/?/, (req, res)=>{

**let** result = { url : req.url

};

result.split = req.url.split('/'); res.json(result);

});

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)

// src/admins/admin1.js module.exports = app => {

app.get(**'/admin1/:p1?/:p2?'**, (req, res)=>{ res.json(req.params);

});

};

// 在 src/index.js 內加入

const admin1 = require(\_\_dirname + '/admins/admin1');

admin1(app);

* 4.5.4 路由模組化 (方式二)

// src/admins/admin2.js

**const** express = require(**'express'**);

**const** router = express.Router(); router.get(**'/admin2/:p1?/:p2?'**, (req, res)=>{

res.json(req.params);

});

module.**exports** = router;

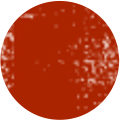
// 在 src/index.js 內加入

const admin2Router = require(\_\_dirname + '/admins/admin2'); app.use(admin2Router); //當成 middleware 使用

* 4.5.5 路由模組化 (方式三)

// 在 src/index.js 內加入

const admin3Router = require(\_\_dirname + '/admins/admin3'); app.use('/admin3', admin3Router); // module 裡面的路徑用相對路徑



// src/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));

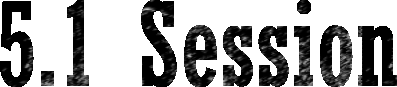
})

.post((req, res)=>{

res.send('post edit:' + JSON.stringify(res.locals.memberData));

});

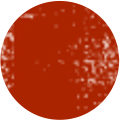
module.exports = router;



* + 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**++;

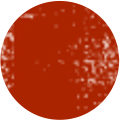
res.json({

**my\_var**: req.**session**.**my\_var**,

**session**: req.**session**

});

});



|  |  |  |
| --- | --- | --- |
| <%# views/login.ejs %>  <% if(locals.flashMsg) { %>  <div class="alert alert-danger" role="alert"> | |  |
| <%= flashMsg %>  </div> | * 登入表單 |  |
| <% } %>  <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> | |  |

|  |  |  |
| --- | --- | --- |
|  | |  |
| app.get(**'/login'**, (req, res)=>{  **let** data = { | * 登入的表單：src/inde | x.js |
| **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：src/index.js

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

});

app.post(**'/login'**, (req, res)=>{

**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**;

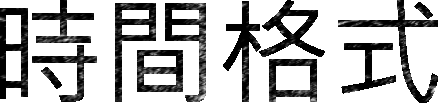
} **else** {

req.**session**.**flashMsg** = **'**密碼錯誤**'**;

}

res.**redirect**(**'/login'**);

});

# 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

// src/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),

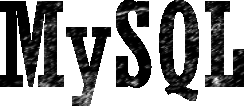
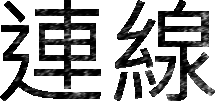
**'london-mo1'**: mo1.tz(**'Europe/London'**).format(fm),

**'london-mo2'**: mo2.tz(**'Europe/London'**).format(fm),

});

});





# 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 mysql 套件連線
  + 安裝：> **npm i mysql**

# 在 test 資料庫，建立資料表輸入資料

**CREATE TABLE** IF **NOT EXISTS** `sales` (

`sid` **int**(11) **NOT NULL**,

`sales\_id` **varchar**(255) **NOT NULL**,

`name` **varchar**(255) **NOT NULL**,

`birthday` **date NOT NULL**

) ENGINE=InnoDB AUTO\_INCREMENT=4 **DEFAULT** CHARSET=utf8;

**INSERT INTO** `sales` (`sid`, `sales\_id`, `name`, `birthday`) **VALUES**

|  |  |  |
| --- | --- | --- |
| (1, | 'A001', '李小明', | '1990-09-13'), |
| (2, | 'B002', '陳小華', | '1989-01-10'), |
| (3, | 'A002', '吳小安', | '1990-02-20'); |

**ALTER TABLE** `sales`

**ADD PRIMARY KEY** (`sid`),

**ADD UNIQUE KEY** `sales\_id` (`sales\_id`);

**ALTER TABLE** `sales`

MODIFY `sid` **int**(11) **NOT NULL** AUTO\_INCREMENT,AUTO\_INCREMENT=4;

# 呈現的頁面

<%# views/sales3.ejs %>

<div class="row">

<div class="col-md-6">

<table class="table table-striped">

<thead>

<tr>

<th scope="col">Id</th>

<th scope="col">Name</th>

<th scope="col">Birthday</th>

</tr>

</thead>

<tbody>

<% for(let v of sales) { %>

<tr>

<td><%= v.sales\_id %></td>

<td><%= v.name %></td>

<td><%= v.birthday2 %></td>

</tr>

<% } %>

</tbody>

</table>

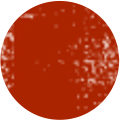
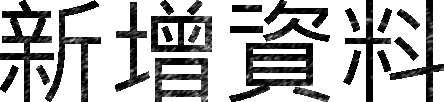
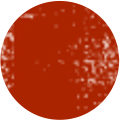
</div>

</div>

|  |  |  |
| --- | --- | --- |
| // src/index.js 加 入 | |  |
| var mysql = require('mysql');  var db = mysql.createConnection({ | * 連線並讀取資料 | 表 |
| host: 'localhost',  user: 'root', password: 'root', database: 'test'  });  db.connect(); // 連 線  app.get('/sales3', (req, res)=>{  var sql = "SELECT \* FROM `sales`"; db.query(sql, (error, results, fields)=>{  if(error) throw error;  console.log(results, fields); for(let v of results){  v.birthday2 = moment(v.birthday).format('YYYY-MM-DD');  }  res.render('sales3', { sales: results  });  });  }); | |  |

* 測試結果 http://localhost:3000/sales3





<%# views/sales3-add.ejs %>

<% if(locals.message) { %>

<div class="alert alert-<%= message.type %>" role="alert">

<%= message.text %>

</div>

<% } %>

<div class="row">

<div class="col-md-6">

<form method="post">

<div class="form-group">

<label for="sales\_id">員工編號</label>

<input type="text" class="form-control" name="sales\_id" value="<%= locals.body ? body.sales\_id : '' %>">

</div>

<div class="form-group">

<label for="name">姓名</label>

<input type="text" class="form-control" name="name" value="<%= locals.body ? body.name : '' %>">

</div>

<div class="form-group">

<label for="birthday">生日</label>

<input type="text" class="form-control" name="birthday" value="<%= locals.body ? body.birthday : '' %>">

</div>

<button type="submit" class="btn btn-primary">新增</button>

</form>

</div>

</div>

// src/index.js 內 容

app.get('/sales3/add', (req, res)=>{

res.render('sales3-add');

});

app.post('/sales3/add', (req, res)=>{

const data = { success: false, message: {type: 'danger', text:''} }; data.body = req.body; // 取得所有 post 資料

if(!req.body.sales\_id || !req.body.name || !req.body.birthday){ data.message.text = '資料不齊全';

return res.render('sales3-add', data);

}

// \*\*\* 內容見下一頁 \*\*\*

});

//接續上一頁內容

let sql = "INSERT INTO sales(sales\_id, `name`, birthday) VALUES

db.query( sql,

[req.body.sales\_id, req.body.name, req.body.birthday], (error, results, fields)=>{

if(error) throw error; console.log(results); if(results.affectedRows===1){

data.success = true;

(?,

?, ?);";

data.message.type

data.message.text

} else {

data.message.text

}

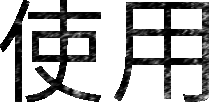
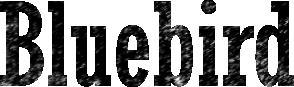
= 'primary';

= '新增完成';

= '資料沒有新增';

res.render('sales3-add', data);

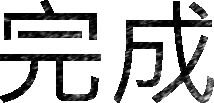
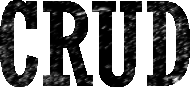
});

# 讓 non-blocking IO 變成 Promise

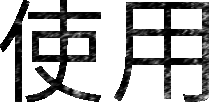
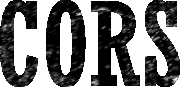
* + 可以確保完成某 query 之後再進行另一個 query
  + 官網： <http://bluebirdjs.com/>
  + 安裝：> **npm i bluebird**

|  |  |  |
| --- | --- | --- |
| // src/index.js 新增內容 | |  |
| var bluebird = require('bluebird');  bluebird.promisifyAll(db); // db 為 mysql 連線物件 | 測試 bluebi | rd |
| app.get('/try-bluebird', (req, res)=>{ const output = [];  db.**queryAsync**("SELECT \* FROM sales WHERE sid=?", [3])  .then( results=> { output.push(results[0]);  return db.queryAsync("SELECT \* FROM sales WHERE sid=?", [2]);  })  .then(results=>{ output.push(results[0]); res.json(output);  })  .catch((error)=>{  console.log('\*\*\* sql error \*\*\*: ', error);  });  }); | |  |

# 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

# 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](http://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>

* 測試方式：
* 需要預先安裝 serve 套件： $ npm i –g serve
* 啟開一個 terminal 視窗，在 public 路徑，啟動 serve



$ **cd public**

$ **serve**