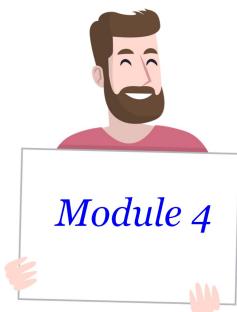




使用 Express.js



designed by freepik

Estimated time:

50 min.

資訊工業策進會 Institute for Information Industry

【Key Points】：

學習目標

- 4-1: 安裝 Express
- 4-2: 建立主要執行檔 index.js
- 4-3: 設定網站根目錄 routes



4-1

 財團法人資訊工業策進會
INSTITUTE FOR INFORMATION INDUSTRY

1. 這是本模組的學習目標
2. 在4-1會介紹如何安裝Express
3. 在4-2會如何建立主要執行檔
4. 在4-3會教如何啟動服務並監聽
5. 後面會有Lab步驟，讓各位依照步驟練習

【Key Points】：

1. 在4-1會介紹如何安裝Express
2. 在4-2會如何建立主要執行檔
3. 在4-3會教如何啟動服務並監聽

4-1：安裝 Express

- 關於 Express
- 事前準備
- 創建專案目錄並初始化
- 如何安裝 Express



designed by freepik



designed by freepik

4-2

 財團法人資訊工業策進會
INSTITUTE FOR INFORMATION INDUSTRY

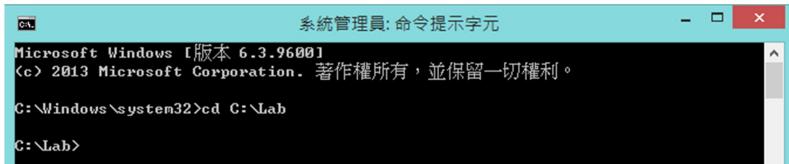
1. 這裡我們會介紹甚麼是 Express
2. 安裝前要做甚麼準備
3. 如何創建專案目錄
4. 如何初始化
5. 接著才會在該專案目錄下安裝Express。

【Key Points】：

1. 這裡我們會介紹甚麼是 Express
2. 安裝前要做甚麼準備
3. 如何創建專案目錄並初始化與安裝Express。

關於 Express

- Express 是何物？
- 安裝 Express 的事前準備
- 建立並切換到工作目錄



4-3

 財團法人資訊工業策進會
INSTITUTE FOR INFORMATION INDUSTRY

1. 安裝前，先了解一下什麼是 Express (官網：<https://expressjs.com/zh-tw/>) 吧！Express 可以說是 Node.js 中最流行的 Web 開發框架/模組，不但有著簡潔靈活的特性、還有豐富完整的 API 文件(<https://expressjs.com/zh-tw/4x/api.html>)，讓您能快速地搭建一個網站。
2. 安裝 Express 前需要做甚麼準備呢？前面提到，Express是Node.js中的模組，故理所當然必須先有Node.js的環境。若尚未安裝 Node.js(<https://nodejs.org/en/>)，請先依照 Module 1 的步驟進行安裝。
3. 在C槽底下新建一資料夾，命名為“Lab”，開啟「命令提示字元」，輸入cd C:\Lab，切換到工作目錄。這工作目錄主要是用來裝之後要建立的專案目錄，因為之後實際開發可能會有很多專案目錄。
4. 命令提示字元的多種開啟方式：
 - ✓ 開始>執行>輸入cmd>按確認鍵
 - ✓ 按Win + X，當出現功能表後，選擇「命令提示字元」或「命令提示字元(系統管理員)」在管理員模式下打開。
 - ✓ 按Win + S，輸入cmd，對結果按右鍵以系統管理員身分執行。
 - ✓ 按Shift + 滑鼠對某資料夾按右鍵，出現功能表後，選擇「在此處開啟命令視窗」。

【Key Points】：

1. 介紹 Express
2. 事前準備
3. 老師可以示範命令提示字元的多種開啟方式

創建 Node 專案目錄

- 輸入：`mkdir myapp` 建立名為 `myapp` 的資料夾作為專案名稱。
- 輸入：`cd myapp` 切換到該目錄



4-4

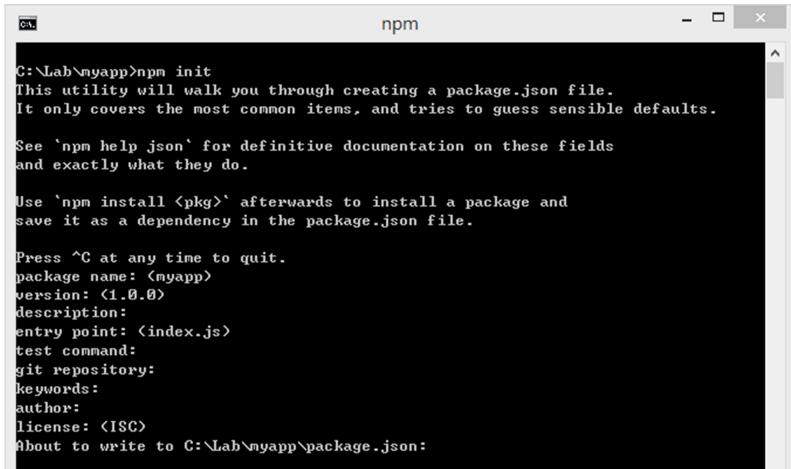
1. 讓我們從頭開始創建一個 Node 專案。
2. 在桌面建立一個叫做 `myapp` 的資料夾作為專案名稱(可改為你想要的名字)。
3. 輸入：`mkdir myapp` 建立名為 `myapp` 的資料夾作為專案名稱。
4. 輸入：`cd myapp` 切換到該目錄
5. 常用cmd指令整理：
查詢目錄 (dir)、建立目錄 (md, mkdir)、變更目錄 (cd, chdir)、刪除目錄 (rd, rmdir)、檔案重新命名 (ren, rename)

【Key Points】：

1. 創建Node專案
2. 創建與切換目錄
3. 老師可以實際示範cmd的各種指令使用方式，但要注意這些指令跟Mac系統有些差異。

專案初始化

- 輸入：**npm init** 將該資料夾轉成一個 **node** 專案。
- 若無特殊需求，可一路按確認鍵到底。



```
C:\Lab\myapp>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See 'npm help json' for definitive documentation on these fields
and exactly what they do.

Use 'npm install <pkg>' afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: <myapp>
version: <1.0.0>
description:
entry point: <index.js>
test command:
git repository:
keywords:
author:
license: <ISC>
About to write to C:\Lab\myapp\package.json:
```

4-5

1. 建立完後在資料夾中運行 **npm init** 將該資料夾轉成一個 **node** 專案。
2. 若無特殊需求，可一路按確認鍵到底。
3. 上述設定會儲存在 **package.json** 這個檔案中，
4. 使用者可以定義應用名稱 (**name**)、應用描述 (**description**)、關鍵字 (**keywords**)、版本號 (**version**)、應用配置 (**config**)、主頁 (**homepage**)、作者(**author**)、版本庫 (**repository**)、bug的提交地址 (**bugs**)、授權方式(**licenses**)... 等。
5. 重點項目是程式進入點與依賴模組

【Key Points】：

1. 使用**npm init**
2. 環境參數設定
3. 老師可以帶著學生看一下**package.json** 這個檔案，說明每個項目可能需填的內容

產生 package.json

- 輸入：**yes** 按下確認鍵，將會產生 package.json 。

```
entry point: <index.js>
test command:
git repository:
keywords:
author:
license: <ISC>
About to write to C:\Lab\myapp\package.json:

{
  "name": "myapp",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}

Is this ok? <yes> yes
```

4-6

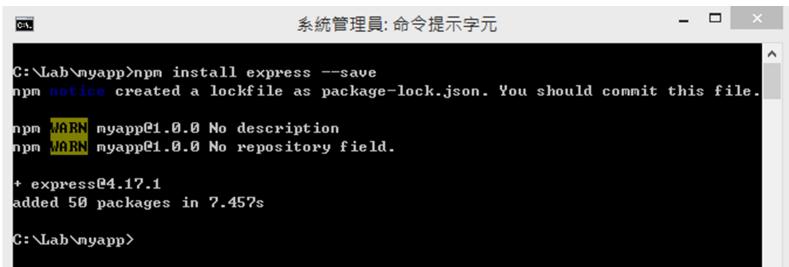
1. 輸入：**yes** 按下確認鍵，將會產生 package.json 。
2. 前面的設定會儲存在 package.json 這個檔案中，
3. 使用者可以定義應用名稱 (name)、應用描述 (description)、關鍵字 (keywords)、版本號 (version)、應用配置 (config)、主頁 (homepage)、作者(author)、版本庫 (repository)、bug 的提交地址 (bugs)、授權方式(licenses)... 等。
4. 重點項目是程式進入點與依賴模組
5. 可以手動修改package.json

【Key Points】：

1. 也可以不輸入**yes**，直接按確認鍵，預設是**yes**
2. 預設的**entry point**名稱為**index.js**，故沒改的話，之後我們建立的js檔案就必須要取為**index.js**。
3. 如果要修改，也可以直接開啟**package.json**進行修改。

開始安裝Express

- 輸入：`npm install express --save` 安裝Express。
- 看到以下畫面，就代表安裝成功。



```
C:\Lab\myapp>npm install express --save
npm notice created a lockfile as package-lock.json. You should commit this file.

npm WARN myapp@1.0.0 No description
npm WARN myapp@1.0.0 No repository field.

+ express@4.17.1
added 50 packages in 7.457s

C:\Lab\myapp>
```

4-7

1. 輸入：`npm install express --save` 安裝Express。
2. 「`--save`」會將這個 package 設定為本次專案的依賴，
3. 「Dependency」(依賴模組)是一個非常重要的概念，專案轉移的時候很好用
4. 依賴會新增至 `package.json` 檔中的 `dependencies` 清單。
5. 看到如圖畫面，就代表安裝成功。

【Key Points】：

1. 說明加上「`--save`」的用意。
2. `--save` 的意思是將這個 package 設定為本次專案的依賴，
3. 新增至 `package.json` 檔中的 `dependencies` 清單。

4-2: 建立主要執行檔 index.js

- 檔名非 index.js 不可？
- 該如何編輯？
- 編輯的語法？



designed by freepik



designed by freepik

4-8

 財團法人資訊工業策進會
INSTITUTE FOR INFORMATION INDUSTRY

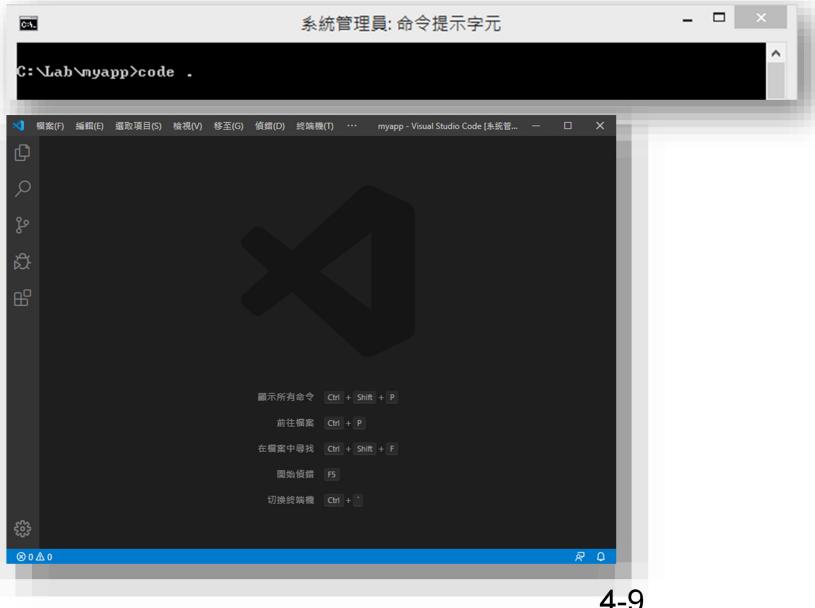
1. 為何檔名要叫做 index.js 呢？
2. 如果前面有專心聽課的話，應該知道在 package.json 中我們有特別提到預設的 entry point 為 index.js，所以在沒有修改的情況下，我們必須取作 index.js。
3. 這邊我們是使用 VS Code 進行示範，當然您也可使用其他程式碼編輯器進行程式撰寫
4. 常見編輯器例如Vim、Emacs、Atom、Sublime等。
5. 由副檔名可以猜到，js 就是 JavaScript 的縮寫，所以編輯的語法就是 JavaScript。

【Key Points】：

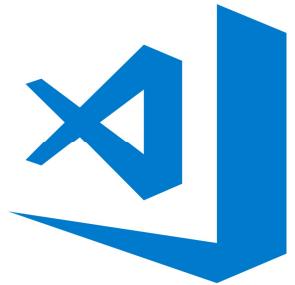
1. 取為 index.js 的理由
2. 常見編輯器
3. js 就是 JavaScript 的縮寫

開啟VS Code

- 在命令提示字元中輸入：**code .** 開啟 VS Code 編輯器。



4-9



III 財團法人資訊工業策進會
INSTITUTE FOR INFORMATION INDUSTRY

- 輸入：**code .** 開啟 VS Code 編輯器。
- 注意 **code** 和 **.** 中間要有空格
- Visual Studio Code(簡稱VS Code) 是一套微軟釋出跨平台 (Windows、Mac、Linux) 的免費開源程式碼編輯器。
- VS Code是用 Electron 框架開發的，支援海量的程式語言，並還有許多強大的功能和第三方套件可以使用。
- 你可以在 VS Code 官網找到 Windows、Mac、Linux 的下載點，下載後即可安裝。
下載點：<https://azure.microsoft.com/zh-tw/products/visual-studio-code/>

【Key Points】：

- 提醒學生注意 **code** 和 **.** 中間要有空格
- 在VS Code中按下 **Ctrl + `**，就可以開啟終端機使用。
- VS Code下載點

建立index.js

- 點選新增檔案，輸入 index.js，按下確認鍵。

The screenshot shows the VS Code interface. On the left, the file tree displays a new file named 'index.js' under the 'MYAPP' folder. On the right, the code editor shows the 'package.json' file with the following content:

```
1 "name": "myapp",
2 "version": "1.0.0",
3 "description": "",
4 "main": "index.js", 主程式檔名
5 "scripts": {
6   "test": "echo \\\"Error: no test specified\\\" && exit 1"
7 },
8 "author": "",
9 "license": "ISC",
10 "dependencies": {
11   "express": "Express 版本^4.17.1"
12 }
13 }
14
15 }
```

4-10

- 在VS Code中點選新增檔案按鈕
- 輸入index.js，新增名為 index.js 的檔案
- 如果要取其他名稱，注意要在package.json中的main修改。
- package.json中可看到前面所安裝的Express版本
- 因為前面是一路確認鍵下去，所以部分欄位是空的，可以手動補上

【Key Points】：

- 新增檔案 index.js
- 可試著改為其他名稱
- 試著修改package.json

編輯index.js內容

- 在 index.js 中輸入以下程式碼並儲存

```
var express = require('express');
var app = express();
app.listen(3000); //監聽 3000 port
```



4-11

 財團法人資訊工業策進會
INSTITUTE FOR INFORMATION INDUSTRY

1. 開啟 index.js
2. 在 index.js 中輸入上述程式碼
3. 記得點選儲存，或Ctrl + S
4. 引入Express用require
5. 監聽連接埠用listen

【Key Points】：

1. Port 範圍從 0 到 65535，共有 65536 個、有些port已經有特定用途，須避免使用。
2. 3000 是我們測試時常用的一個 Port，當然你也可以選擇其他編號。
3. 注意避免使用已知的 Port，否則會發生搶 Port 的情況，造成程式或服務無法順利執行。

關於 Port

- 何謂 Port ?
- 一定要使用 3000 port ?
- 使用任何 port 都可以嗎 ?
- 常用的port有哪些 ?



4-12  財團法人資訊工業策進會
INSTITUTE FOR INFORMATION INDUSTRY

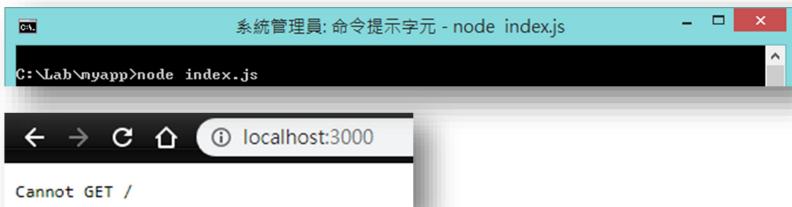
1. 通訊埠（英語：port），又稱為連接埠、協定埠（protocol port），就像是電腦的門，一部電腦的 Port 編號從 0 到 65535 。
2. 3000 是我們測試時常用的一個 Port，當然你也可使用其他，例如 4000, 5000, 8080 等。
3. 要避免其他程式或服務已經使用的 Port，否則會發生搶 Port 的情況，造成程式或服務無法順利執行。
4. 常見的 Port 可參考 維基百科：
<https://zh.wikipedia.org/wiki/TCP/UDP%E7%AB%AF%E5%8F%A3%E5%88%97%E8%A1%A8>
5. 必知的常用 port
 - ✓ 21 : FTP (File Transfer Protocol , 檔傳輸協定) 服務。
 - ✓ 23 : Telnet (遠端登錄) 服務。
 - ✓ 80 : HTTP (HyperText Transport Protocol , 超文本傳輸協定) 服務
 - ✓ 443 : HTTPS 服務，是提供加密和安全傳輸的HTTP。

【Key Points】：

1. 網址沒特別寫出port的話，預設就是80。使用其他port就必須使用冒號特別指定。
2. 要避免其他程式或服務已經使用的 Port，否則會發生搶 Port 的情況。
3. 必知的常用 port 。

啟動服務

- 回到「命令提示字元」
- 輸入 **node index.js** 啟動服務監聽。
- 打開瀏覽器、在 URL 上輸入 **localhost:3000** 可打開網頁。



4-13

1. 回到「命令提示字元」
2. 輸入 **node index.js** 啟動服務監聽。
3. 打開瀏覽器
4. 在 URL 上輸入 **localhost:3000**
5. 打開網頁會發現**Cannot GET**字樣，這是正常的，因為前面的程式中並無定義根目錄與回傳的內容。

【Key Points】：

1. 「命令提示字元」輸入 **node index.js** 啟動服務監聽
2. 打開瀏覽器，在 URL 上輸入 **localhost:3000**
3. 打開網頁會發現**Cannot GET**字樣，目前是正常的

修改index.js

- 加點東西，方便確認服務有啟動，網頁有內容
- 回到 **index.js**，修改為如下程式碼，並記得儲存

```
var express = require('express');
var app = express();
// app.listen(3000); //監聽 3000 port

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

app.listen(3000, function () {
  console.log('Example app listening on port 30
00!');
});
```



4-14

 財團法人資訊工業策進會
INSTITUTE FOR INFORMATION INDUSTRY

1. 開啟index.js
2. 加點東西，方便確認服務有無啟動，網頁有無內容
3. 修改程式碼如上，並記得儲存
4. `app.get`這段是設定根目錄與回傳的內容
5. `app.listen`這段第二參數加入了`function`，是可以在伺服器端印出執行中的字樣，確認服務已經執行

【Key Points】：

1. 開啟index.js
2. 複製貼上上述程式碼
3. 記得儲存，並嘗試理解程式碼意義

重啟服務

- 如何關閉之前已啟動的服務？
- 再次啟動服務監聽。
- 可在畫面上看到方才程式碼中加入的提示文字。

The screenshot displays two windows side-by-side. On the left is a terminal window titled '系統管理員: 命令提示字元 - node index.js'. It shows the command 'node index.js' being run, followed by '^C' to stop it, and then the command 'node index.js' again. The output 'Example app listening on port 3000!' is highlighted with a red box. On the right is a code editor window titled 'JS index.js'. It contains the following code:

```
1 var express = require('express');
2 var app = express();
3
4 app.get('/', function (req, res) {
5   res.send('Hello World!');
6 });
7
8 app.listen(3000, function () {
9   console.log('Example app listening on port 3000!');
10});
```

The line 'console.log('Example app listening on port 3000!');' is also highlighted with a red box.

4-15

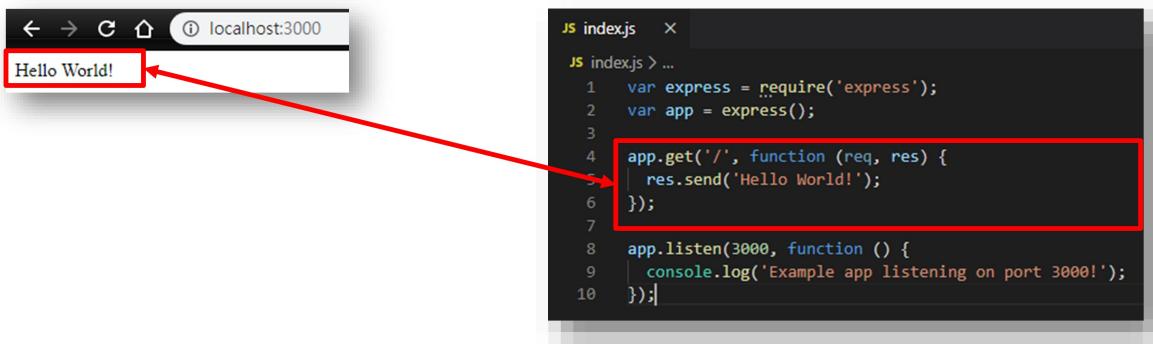
1. 回到「命令提示字元」，
2. 鍵盤按下「Ctrl + C」，將先前的服務先關閉。
3. 輸入 node index.js 再次啟動服務監聽。
4. 可在畫面上看到方才程式碼中加入的提示文字。
5. 這內容只會顯示在伺服器端，客戶端是看不見的

【Key Points】：

1. 應用程式會啟動伺服器，並在埠 3000 監聽連線。
2. 鍵盤按下「Ctrl + C」，將先前的服務先關閉
3. 提醒學生若關閉命令提示字元視窗，服務也會跟著停止，網頁就無法開啟。

檢視網頁

- 打開瀏覽器，在網址列上輸入 **localhost:3000** 可打開網頁。
- 可在網頁看到方才程式碼中加入的 **Hello World!** 字樣。



4-16

 財團法人資訊工業策進會
INSTITUTE FOR INFORMATION INDUSTRY

- 打開瀏覽器
- 在網址列上輸入 **localhost:3000** 可打開網頁。
- 可在網頁看到方才程式碼中加入的 **Hello World!** 字樣。
- req** 是 Request 物件，存放這此請求的所有資訊
- res** 是 Response 物件，用來回應該請求

【Key Points】：

- 老師可以稍微帶一下這段代碼的意思，之後路由也是從這邊下手處理
- req** 是 Request 物件，存放這此請求的所有資訊
- res** 是 Response 物件，用來回應該請求

4-3: 設定網站根目錄 routes

- 網站根目錄
- 使用**express.Router()**
- 自訂 404 頁面



designed by freepik



designed by freepik

4-17

 財團法人資訊工業策進會
INSTITUTE FOR INFORMATION INDUSTRY

1. 前面我們嘗試修改了伺服器端回傳顯示的內容，在網頁上也忠實呈現。
2. 不過一個網站不會只有一個頁面，會有多個頁面，例如首頁、關於頁...等
3. 伺服器端提供的API也不會只有一種，例如帳號登入api，清單api...等
4. 這時候我們就會需要用到路由(Router)。
5. 我們甚至可以自訂404頁面。

【Key Points】：

1. 路由使用時機
2. 根目錄設定
3. 自訂404頁面

網站根目錄

- 斜線是根目錄



```
JS index.js  X
JS index.js > ...
1 var express = require('express');
2 var app = express();
3
4 app.get('/', function (req, res) {
5   res.send('Hello World!');
6 });
7
8 app.listen(3000, function () {
9   console.log('Example app listening on port 3000!');
10});|
```

4-18

1. 還記得前面這段程式碼嗎？
2. 我們將目光放到其中的 `app.get('/')`
3. 裏頭那個斜線，其實就是預設的根目錄。
4. 後面要調整路由也是從這邊下手。
5. 而`res.send()`就是要回傳的內容。

【Key Points】：

1. 網站根目錄
2. 斜線是預設根目錄
3. 路由由此修改

使用express.Router()

- 將程式改寫如下

```
var express = require('express');
var app = express();

var page = express.Router();

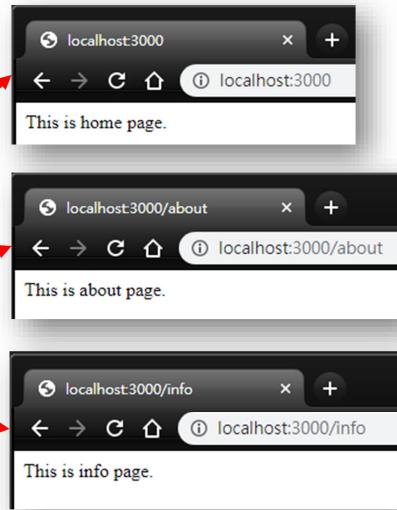
//Pages
//首頁
page.get('/', function (req, res) {
  res.send('This is home page.');
});

//關於頁
page.get('/about', function (req, res) {
  res.send('This is about page.');
});

//資訊頁
page.get('/info', function (req, res) {
  res.send('This is info page.');
});

app.use('/', page);

app.listen(3000, function () {
  console.log('Example app listening on port 3000!');
});
```



4-19

- 假設我們總共要有三個頁面，分別是首頁、關於頁、資訊頁。
- 這裡我們使用**express.Router()**來撰寫。
- 請將程式碼修改如上。
- 重啟服務，並直接在瀏覽器上輸入頁面網址
- 修改斜線後面的參數，就可以得到不同內容。

【Key Points】：

- 使用**express.Router()**設定路由
- 注意最後要用**app.use('/router')**來指定路由位置
- 老師可以示範看看，若不使用**express.Router()**，而直接使用**app.get()**的方式該如何寫。

API Router

- 新增API服務

The image shows a code editor on the left and two browser windows on the right. The code editor contains an Express.js application with two router definitions: 'User API' and 'City API'. The 'User API' section is highlighted with a red box, and the 'City API' section is also highlighted with a red box. Red arrows point from these highlighted sections to their corresponding JSON responses in the browser windows. The top browser window shows the response for the '/getUser' endpoint, and the bottom browser window shows the response for the '/getCity' endpoint, both displaying the expected JSON objects.

```
var api = express.Router();
//APIs
//User API
api.get('/getUser', function (req, res) {
  res.json({
    type: "api",
    content: "This is User API"
  });
}

//City API
api.get('/getCity', function (req, res) {
  res.json({
    type: "api",
    content: "This is City API"
  });
}

app.use('/', api);
```

4-20



1. 假設伺服器要提供API服務
2. 為了方便管理，我們可以再新增另一個express.Router()。
3. 請將上面程式碼加入，重啟服務。
4. 直接在瀏覽器上輸入API網址，就可以得到回傳的JSON值。
5. 使用res.json(obj)，可以回傳JSON格式的字串。

【Key Points】：

- 1.新增另一個express.Router() 方便區隔頁面與API服務。
- 2.使用res.json(obj)，可以回傳JSON格式的字串。
- 3.老師可以示範看看，若不使用express.Router()，而直接使用app.get()的方式該如何寫。

API URL調整

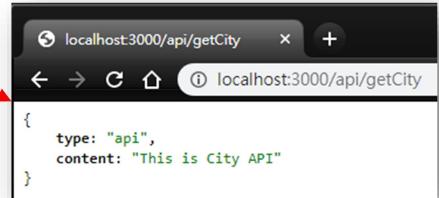
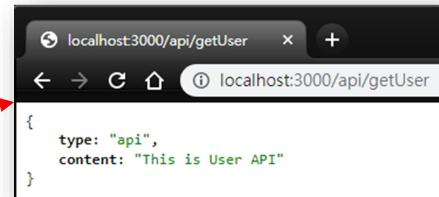
- 新增API服務

```
var api = express.Router();
//APIs
//User API
api.get('/getUser', function (req, res) {
  res.json({
    type: "api",
    content: "This is User API"
  });
}

//City API
api.get('/getCity', function (req, res) {
  res.json({
    type: "api",
    content: "This is City API"
  });
}

app.use('/api', api);
```

請注意網址的變化



4-21

III 財團法人資訊工業策進會
INSTITUTE FOR INFORMATION INDUSTRY

- 細心的同學應該會發現，前面的API網址，看起來跟頁面網址很像。
- 如果我們想要在前面再多一層api字樣，要如何做呢？
- 也就是我的網址想要變成localhost:3000/api/getCity
- 重點在最後一行app.use('/', api);
- 改為app.use('/api', api); 即可。

【Key Points】：

- Router根目錄
- 修改Router
- 老師可以示範看看，若不使用express.Router()，而直接使用app.get()的方式該如何寫。

自訂 404 頁面

- 預設找不到網頁的畫面



- 自訂畫面



```
//自訂404
app.use(function (req, res) {
  res.send('This is my Custom 404 Page');
});
```

4-22

1. 前面我們制訂了三個頁面和二個API的路由，但如果使用者將網址打錯，就會出現預設找不到網頁的畫面。
2. 我們可以加上這段程式碼，就可以自訂想傳回的內容。
3. 當然我們可以做一個自訂的404網頁來顯示。
4. 注意app.use()內現在只有一個function，而沒有帶路由了。
5. res.send()這段就是要顯示的內容，如果改用res.sendFile()就可以指定要顯示的html檔案。

【Key Points】：

- 1.自訂404頁面的方法，請注意app.use()內現在只有一個function，而沒有帶路由了。
- 2.改用res.sendFile()就可以指定要顯示的html檔案
- 3.老師可以預先準備一個html檔案做示範

Summary 〈 精華回顧 〉

- 安裝 Express 指令是 `npm install express --save`
- `--save` 會將該模組新增至依賴清單。
- 使用 `npm install` 可自動安裝 `package.json` 檔中的所有依賴模組。
- 連接埠 (port)，編號從 0 到 65535，HTTP 是 80，HTTPS 是 443。
- 初始化指令是 `npm init`
- 啟動服務監聽指令是 `node index.js`
- 關閉命令提示字元視窗或按下 「`Ctrl + C`」，可將先前的服務先關閉。
- Cmd 指令：建立目錄(`mkdir`)、刪除目錄(`rmdir`)、切換目錄(`cd`)
- 路由可使用 `express.Router()` 進行管理



4-23

 財團法人資訊工業策進會
INSTITUTE FOR INFORMATION INDUSTRY

1. 安裝 Express 的指令
2. `--save` 參數的意義
3. 通常在備份或移動專案程式碼，不會將肥大的模組檔案(`node_modules` 資料夾)一同打包，所以會將所依賴的模組記錄在 `package.json` 中。
4. 在新環境要重新建立服務時，只要使用 `npm install`，就可以將所有依賴模組下載重新安裝。
5. 已知常用 `port` 務必記住，並且避免衝突使用
6. 關閉服務的方法

【Key Points】：

1. 安裝 Express 的指令
2. 老師可以再解釋一下儲存依賴的原因
3. 已知常用 `port` 提醒

線上程式題

- **題目一：如何引用 Express 模組？**

我們想利用 Express 模組建立Web服務，該安裝與引用哪一個模組？

- **題目二：如何建立不同頁面？**

我們想利用 Express 模組建立不同頁面，該如何實作呢？

- **題目三：如何回傳json字串？**

我們想利用 Express 模組建立Web服務，若要做一個簡易 API，該如何回傳 JSON資料呢？

4-24



題目一：如何引用 Express 模組？

內容說明：

我們想利用 Express 模組建立Web服務，該安裝與引用哪一個模組？

題目二：如何建立不同頁面？

內容說明：

我們想利用 Express 模組建立不同頁面，該如何實作呢？

題目三：如何回傳json字串？

內容說明：

我們想利用 Express 模組建立Web服務，若要做一個簡易 API，該如何回傳JSON資料呢？

請閱讀教學系統的程式與說明，然後，到「// 作答區」填入答案。

答案有區分大寫小寫。

【Key Points】：

Lab01: 安裝Express模組

Lab02: 建立主要執行檔 index.js

Lab03: 簡單API製作

課後練習題(Lab)

- **Lab01: 安裝Express模組**
 - 先確認是否已經安裝 Node.js 環境
 - npm init 將該資料夾轉成一個 node 專案
 - npm install express --save 安裝Express
- **Lab02: 建立主要執行檔 index.js**
 - require('express')
 - app.listen(4000, ...)
- **Lab03: 簡單API製作**
 - var router = express.Router();
 - router.get('/data', ...)

Estimated time:

20 min.

4-25



情節描述:

本Lab 是假設在已經安裝Node.js環境的Windows 電腦下，如何安裝Express模組，建置並成功啟動一簡單API服務。

預設目標:

成功安裝Express，最終可完成一簡單API。

三段練習的重點描述:

Lab01: 安裝Express模組

先確認是否已經安裝 Node.js 環境
npm init 將該資料夾轉成一個 node 專案
npm install express --save 安裝Express

Lab02: 建立主要執行檔 index.js

require('express')
app.listen(4000, ...)

Lab03: 簡單API製作

var router = express.Router();
router.get('/data', ...)

【Key Points】：

- npm install express
- require('express')
- express.Router();

範例程式使用說明

- 範例程式資料夾: Module_04_example
- 使用步驟:
 1. 安裝 Node.js
 2. 安裝 Visual Studio Code
 3. 以 Visual Studio Code 開啟本模組的範例資料夾
 4. 在 Visual Studio Code 按下「Ctrl + 滑鼠右鍵」開啟 terminal 終端機視窗。
 5. 在終端機視窗，輸入下列指令，安裝必要的模組套件:
`npm install express`
 6. 在終端機視窗，輸入 `node index.js`
 7. 啟動瀏覽器，連接 `http://localhost:4000`

4-26



使用步驟:

1. 安裝 Node.js
<https://nodejs.org/en/>
2. 安裝 Visual Studio Code
<https://code.visualstudio.com/>
3. 以 Visual Studio Code 開啟範例資料夾
在檔案總管，滑鼠右鍵點按「本範例資料夾」，從快捷功能表點選「以Code開啟」
或者，啟動 Visual Studio Code 之後，功能表 File | Open Folder，選擇「本範例資料夾」
4. 在 Visual Studio Code 按下「Ctrl + 滑鼠右鍵」開啟 terminal 終端機視窗。
5. 在終端機視窗，輸入下列指令，安裝必要的模組套件:
`npm install express`
6. 在終端機視窗，輸入 `node index.js`
7. 啟動瀏覽器，連接 `http://localhost:4000`

【Key Points】：

程式執行環境 Node.js

程式開發編輯環境: Visual Studio Code

在 Visual Studio Code 按下「Ctrl + 滑鼠右鍵」開啟 terminal 終端機視窗，輸入：「node 主程式.js」