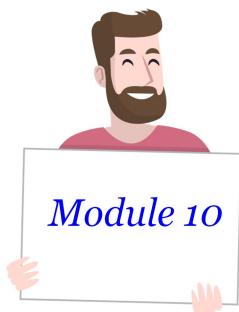




路由模組化



designed by freepik

Estimated time:

50 min.

III 資訊工業策進會 Institute for Information Industry

【Key Points】：

學習目標

- **10-1: 以函式方式模組**
- **10-2: 使用 Routers**
- **10-3: 將 Routers 當 middlewares 使用**



10-1

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

本篇將介紹如何將路由模組化，並且學習如何使用 Express.js 的 Router，還有 middleware 的基本概念：

- 如何把函式當作 router 使用
- 路由函式的語法規範
- 應用多重路由函式
- 甚麼是 Routers
- 路由模組語法
- 如何解讀路由路徑
- 何謂 middleware

【Key Points】：

10-1: 以函式方式模組

10-2: 使用 Routers

10-3: 將 Routers 當 middlewares 使用

10-1:以函式方式模組

- 如何把函式當作 router 使用
- 路由函式的語法規範
- 應用多重路由函式
- 情境範例



designed by freepik



designed by freepik

10-2

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

在這個章節，將說明

- 如何把函式當作 router 使用
- 透過使用陣列，路由就可以可以分段處理資料
- 還可以依照不同情況給予個別的處理
- 使用函式，會讓整個 router 更有脈絡
- 路由函式的語法規範
- 應用多重路由函式
- 情境範例

【Key Points】：

如何把函式當作 router 使用

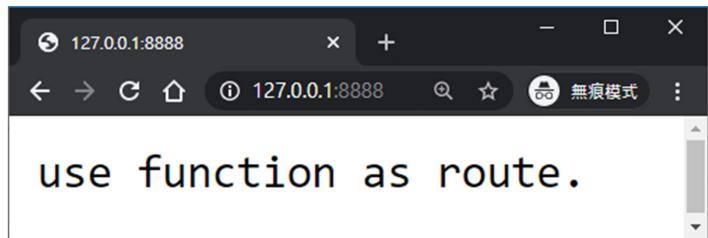
路由函式的語法規範

應用多重路由函式

如何把函式當作 router 使用

- 建立一個名為 func 的函式
- 該函式需要兩個參數
 - req 代表 request
 - res 代表 response

```
1 const express = require('express');
2 const app = express();
3
4 function func(req, res) {
5   res.end('use function as route.');
6 }
7
8 app.get('/', func);
9
10 app.listen(8888);
```



10-3

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

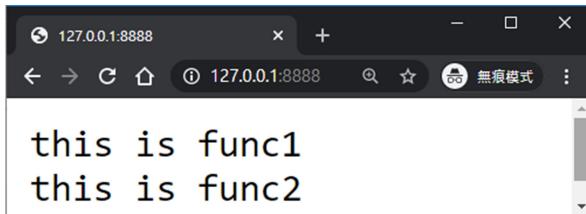
- 之前都是直接把函式直接寫在 app.get 的第二個參數，但其實是可以拿出來另外處理的
- 當程式碼越來越龐大時，將函式獨立出來，做出更相關的命名，會更好維護
- 方便未來重複的程式碼互相引用
- req, res 只是習慣的命名，學員可以自行替換成喜歡的變數名
- 目前只是小範例，下一頁會介紹完整用法

【Key Points】：

之前都是直接把函式直接寫在 app.get 的第二個參數，但其實是可以拿出來另外處理
將函式獨立出來，做出更相關的命名，會更好維護
方便未來重複的程式碼互相引用

路由函式的語法規範

- 函式其實可以有第三個參數 `next`
- `app.get('/', [func1, func2])`



```
1 const express = require('express');
2 const app = express();
3
4 function func1(req, res, next) {
5   res.write(`this is func1\n`);
6   next();
7 }
8
9 function func2(req, res, next) {
10   res.end(`this is func2\n`);
11 }
12
13 app.get('/', [func1, func2]);
14
15 app.listen(8888);
```

10-4

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

- 路由函式有第三個參數 `next`
- `app.get` 第二個參數可以是陣列 (array)
- 會有連續執行的效果
- `\n` 會被當作換行
- 重要：如果不是最後一個函式，一定要在最後用 `next()` 才會繼續執行，或是在前面就 `res.end()`，也會就此結束

【Key Points】：

路由函式有第三個參數 `next`

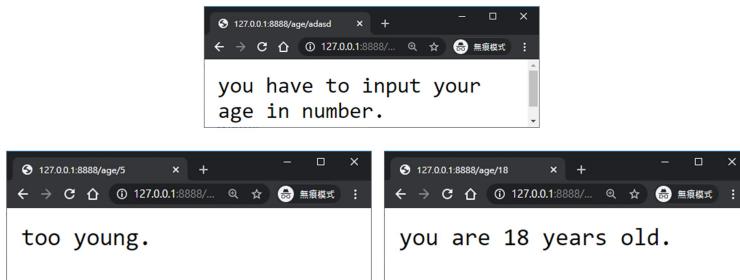
`app.get` 第二個參數可以是陣列 (array)

如果不是最後一個函式，一定要在最後用 `next()` 才會繼續執行

應用多重路由函式

- 分別瀏覽

- <http://127.0.0.1:8888/age/one>
- <http://127.0.0.1:8888/age/5>
- <http://127.0.0.1:8888/age/18>



```
4  function notNumber(req, res, next) {
5    if (isNaN(parseInt(req.params.age)))
6      |   res.end(`you have to input your age in number.`);
7    |   next();
8  }
9
10 function younger(req, res, next) {
11   if (req.params.age < 18)
12     |   res.end(`too young.`);
13   |   next();
14 }
15
16 function func(req, res) {
17   res.end(`you are ${req.params.age} years old.`);
18 }
19
20 app.get('/age/:age', [notNumber, younger, func]);
```

10-5

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

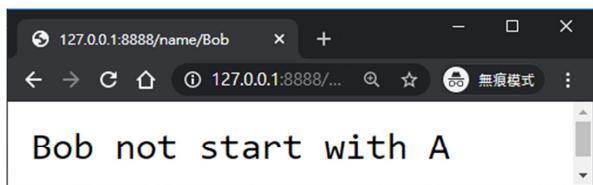
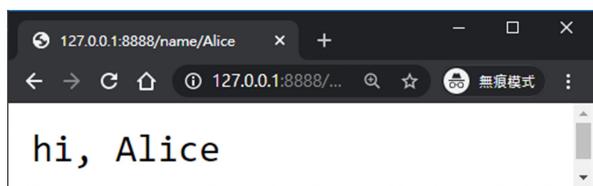
- app.get() 的第二個參數可以是陣列，透過陣列，路由就可以分段處理資料
- 還可以依照不同情況給予個別的處理
- parseInt 可以將字串轉為 Number，可以搭配 isNaN 檢查輸入是否為數字
- NaN 是 not a number 的縮寫，代表“不是數字”
- 使用函式，會讓整個 router 更有脈絡

【Key Points】：

透過使用陣列，路由就可以分段處理資料
還可以依照不同情況給予個別的處理
使用函式，會讓整個 router 更有脈絡

情境範例

- 檢查輸入的名字是否為 A 開頭



```
1 const express = require('express');
2 const app = express();
3
4 function prev(req, res, next) {
5   if (req.params.name.startsWith('A') == false)
6     res.end(` ${req.params.name} not start with A`);
7   next();
8 }
9
10 function func(req, res) {
11   res.end(` hi, ${req.params.name}`);
12 }
13
14 app.get('/name/:name', [prev, func]);
15
16 app.listen(8888);
```

10-6

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

- <http://127.0.0.1:8888/name/Alice> 看到訊息 hi Alice
- <http://127.0.0.1:8888/name/Bob> 看到訊息 Bob not start with A
- 使用 `startsWith` 判斷字串開頭
- 可以簡單的在前面就處理好資訊
- 後面的 `function` 會更加簡潔

【Key Points】：

<http://127.0.0.1:8888/name/Alice> 與 <http://127.0.0.1:8888/name/Bob> 的結果不同
簡單的在前面就處理好資訊
後面的 `function` 會更加簡潔

10-2: 使用 Routers

- 甚麼是 Routers
- 語法
- 如何解讀
- 情境範例



designed by freepik



designed by freepik

10-7

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

在這個章節，將說明使用 Routers，並且用範例解釋語法：

1. 先定義路由模組檔
2. module.export 用來指定匯出的是什麼
3. 主程式以 require() 匯入: const birds = require("./birds");
4. 透過 app.use() 掛入 Router
5. 如何解讀路由路徑

【Key Points】：

先定義路由模組檔

主程式以 require() 匯入: const birds = require("./birds");

透過 app.use() 掛入 Router

甚麼是 Routers

- Routers 是 Express.js 內建的函式庫
- 主要將路由模組化
- 在未來更好將功能區分，並且分類
- 使用方式 `app.use('/somepath', routers)`
- 更可以分在不同檔案中

10-8



- Routers 是 express 的內建函式庫
- 在透過 Routers 後，程式碼會更加模組化
- 更好區分每個不同的功能
- 使用方式主要用 `app.use`，在下一頁會詳細介紹
- 路由處理可以寫在不同檔案中

【Key Points】：

在透過 Routers 後，程式碼會更加模組化
更好區分每個不同的功能
路由處理可以寫在不同檔案

路由模組語法

- 瀏覽器開啟 127.0.0.1:8888/birds

```
JS index.js  X  JS birds.js
C: > Users > tyze > Desktop > Module10 > JS index.js >
1 const express = require('express');
2 const app = express();
3 const birds = require('./birds');
4
5 app.use('/birds', birds);
6
7 app.listen(8888);
```

```
JS index.js  X  JS birds.js  X
C: > Users > tyze > Desktop > Module10 > JS birds.js > ...
1 var express = require('express');
2 var router = express.Router();
3
4 router.get('/', function(req, res) {
5   res.send('Birds home page');
6 });
7
8 router.get('/about', function(req, res) {
9   res.send('About birds');
10 });
11
12 module.exports = router;
```

127.0.0.1:8888/birds
Birds home page

10-9



- 先定義路由模組檔 (本例的 `birds.js`)
- 再以 `app.use()` 掛入，`/birds` 右側的路徑由 `birds.js` 裡的路由程式分派請求到適當的函式
- 連結 `http://127.0.0.1:8888/birds/about` 試試看
- 請注意：`birds.js` 和 `index.js` 在同個路徑下
- 如果不想在同個路徑下，請修改 `index.js` 第三行的路徑至相對應檔案位置
- 在透過 `Routers` 輔助後，`index.js` 主程式碼部分變得相當簡潔

【Key Points】：

請注意：`birds.js` 和 `index.js` 在同個路徑下
先定義路由模組檔
再以 `app.use()` 掛入

如何解讀路由路徑

- 路由模組檔匯出路由:
module.exports = XXXX
- 主程式引用模組:
birds = require("./birds");
- **app.use('/birds', birds)**
是將**birds** 右側的路徑
交由 **birds** 路由處理
- 例如: **/birds/about**,
about 交給路由模組的
app.get('/about', ...) 負責

```
2 var router = express.Router();
3
4 router.get('/', function(req, res) {
5   | res.send('Birds home page');
6 });
7
8 router.get('/about', function(req, res) {
9   | res.send('About birds');
10 });
11
12 module.exports = router;
```

```
JS index.js  X JS birds.js
C: > Users > tyze > Desktop > Module10 > JS index.js >
1  const express = require('express');
2  const app = express();
3  const birds = require('./birds');
4
5  app.use('/birds', birds);
6
7  app.listen(8888);
```

10-10

- **module.export** 用來指定匯出的是什麼
- 主程式以 **require()** 引用模組: **const birds = require("./birds");**
- 透過 **app.use()** 掛入 Router 時，路徑不會如同 Router 自己內部設定，也就是說：不是根目錄，而是看 **app.use** 時，掛在哪一個指定的路徑
- 例如: **/birds/about**, **about** 交給路由模組的 **app.get('/about', ...)** 負責
- 此時瀏覽器開啟 **127.0.0.1:8888/** 不會顯示東西
- 應該用 **127.0.0.1:8888/birds** 才會到達 **birds**路由的第四行 /
- **127.0.0.1:8888/about** 同理是無法連結的，應該要用 **127.0.0.1:8888/birds/about**

【Key Points】：

module.export 用來指定匯出的是什麼

主程式以 **require()** 匯入: **const birds = require("./birds");**

透過 **app.use()** 掛入 Router

情境範例

- 使用 Router 簡化主程式吧！
- 分別用瀏覽器開啟
 - <http://127.0.0.1:8888/birds>
 - <http://127.0.0.1:8888/dogs>

```
JS index.js X JS birds.js JS dogs.js
C: > Users > tyze > Desktop > Module10 > JS index.js > ...
1 const express = require('express');
2 const app = express();
3 const routers1 = require('../birds');
4 const routers2 = require('../dogs');
5
6 app.use('/birds', routers1);
7 app.use('/dogs', routers2);
8
9 app.listen(8888);
```

```
JS index.js JS birds.js JS dogs.js X
C: > Users > tyze > Desktop > Module10 > JS dogs.js > ...
1 var express = require('express');
2 var router = express.Router();
3
4 router.get('/', function(req, res) {
5 |   res.send('Dogs home page');
6 });
7
8 module.exports = router;

JS index.js JS birds.js X JS dogs.js
C: > Users > tyze > Desktop > Module10 > JS birds.js > ...
1 var express = require('express');
2 var router = express.Router();
3
4 router.get('/', function(req, res) {
5 |   res.send('Birds home page');
6 });
7
8 module.exports = router;
```

10-11



- 建立兩個 Router 模組檔
- 一個是給 Dogs 用，一個是給 Birds 用
- 之後只要在各別的檔案中撰寫功能，就做好功能的分類
- 範例比較簡化，在實務上會感受較深
- Router 實際也可以放在同個 JS 檔案下，只是想告訴同學可以分開撰寫，程式的可攜性會變得更好

【Key Points】：

建立兩個 Router 模組檔

分別掛入 /birds 與 /dogs

之後只要在各別的檔案中撰寫功能，就做好功能的分類

10-3: 將 Routers 當 middlewares 使用

- 何謂 middleware
- 情境範例 – 後臺記錄
- 情境範例 – 密碼驗證
- 情境範例 – 資料前處理



designed by freepik



designed by freepik

10-12

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

在這個章節，將介紹 middleware 的概念及用法，內容包括：

- Middleware 翻譯為中介軟體
- 主要指的是“系統層”與“應用層”之間的程式
- 簡單地說，每次的 Request, 中介軟體都會介入處理
- Express 以 app.use() 方法，掛入中介軟體
- 在網路應用中，常用來做 Routing，或是一般的登入驗證，或兩者一起實作。

【Key Points】：

何謂 middleware

每次的 Request, 中介軟體都會介入處理

Express 以 app.use() 方法，掛入中介軟體

何謂 middleware

- Middleware 中介軟體
- 系統層與應用層之間的程式
- 簡單地說，每次的 Request, 中介軟體都會介入處理
- 在網路應用程式中，middleware 先前一層就將資訊處理好
- 應用程式都可共用 middleware，使用相同資源

10-13



- Middleware 翻譯為中介軟體
- 主要指的是“系統層”與“應用層”之間的程式
- 在網路應用中，常用來做 Routing
- 或是一般的登入驗證也可以一起實現
- 簡單地說，每次的 Request, 中介軟體都會介入處理
- Express 以 app.use() 方法，掛入中介軟體
- 詳細的使用方式，可以參考 <https://expressjs.com/zh-tw/guide/using-middleware.html>

【Key Points】：

Middleware 翻譯為中介軟體

簡單地說，每次的 Request, 中介軟體都會介入處理

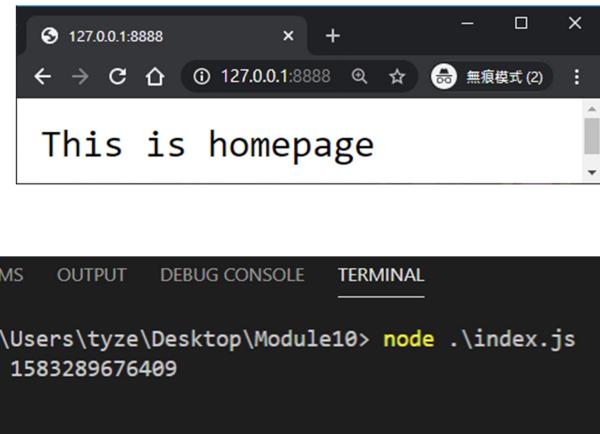
Express 以 app.use() 方法，掛入中介軟體

情境範例 – 後臺記錄

- 可以在後台顯示瀏覽網頁的時間

— 127.0.0.1:8888/

```
1 const express = require('express');
2 const app = express();
3
4 app.use(function (req, res, next) {
5   console.log(`Time: ${Date.now()}`);
6   next();
7 });
8
9 app.get('/', function (req, res) {
10   res.end(`This is homepage`);
11 })
12
13 app.listen(8888);
```



10-14



- 第 4~7 行就是 middleware 的程式
- 在處理 9~11 這項 GET 的程式之前，會先經過一層層的 middleware
- 在後面的範例會持續增加 middleware，就會更理解它的用處
- Date.now() 是這部電腦上目前的日期和時間，以本地時間表示。
- 如同一開始介紹的，每個middleware結束都應該要有 next()

【Key Points】：

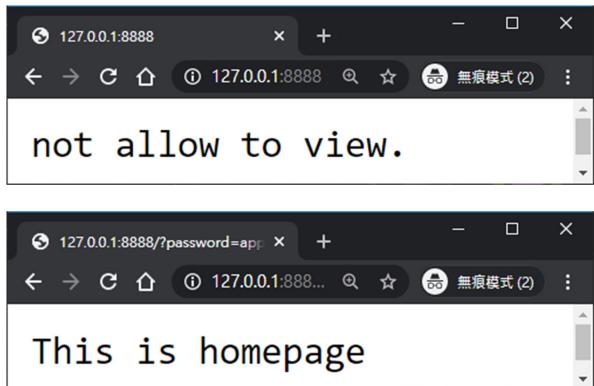
app.use() 掛入 middleware 程式

在處理 GET 的程式之前，會先經過一層層的 middleware

每個middleware結束都應該要有 next()

情境範例 – 密碼驗證

- 當沒有密碼時，就不會看到首頁
 - <http://127.0.0.1:8888/>
 - <http://127.0.0.1:8888/?password=apple>



```
1 const express = require('express');
2 const app = express();
3
4 app.use(function (req, res, next) {
5   console.log(`Time: ${Date.now()}`);
6   next();
7 });
8
9 app.use(function (req, res, next) {
10   if (req.query.password !== "apple")
11     res.end(`not allow to view.`);
12   next();
13 });
14
15 app.get('/', function (req, res) {
16   res.end(`This is homepage`);
17 });
18
19 app.listen(8888);
```

10-15

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

- GET method 可以用 Query String 傳遞參數
- 在程式中使用 req.query 把該 Query String 的內容取出（如程式第 10）
- 瀏覽兩個網址，都會有時間記錄在後台
- 但只有 Query String 具有 password 的時候，才會執行到 15 行
- 沒有密碼時，就不會看到首頁

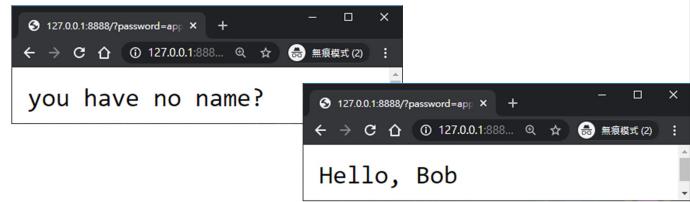
【Key Points】：

瀏覽兩個網址，都會有時間記錄在後台
使用 req.query 把該 Query String 的內容取出
但只有 Query String 具有 password 的時候，才會執行到 15 行

情境範例 – 資料前處理

- 沒有輸入名稱時，將看到不同頁面

- <http://127.0.0.1:8888/?password=apple>
- <http://127.0.0.1:8888/?password=apple&name=Bob>



- 執行順序

- 後臺紀錄時間 (4~7)
- 檢查有沒有密碼 (9~13)
- 檢查有沒有名字 (15~19)
- 顯示首頁 (21~23)

```
9  app.use(function (req, res, next) {  
10     if (req.query.password === "apple")  
11         res.end(`not allow to view.`);  
12     next();  
13 };  
14  
15  app.use(function (req, res, next) {  
16     if (req.query.name === undefined)  
17         res.end(`you have no name?`);  
18     next();  
19 };  
20  
21  app.get('/', function (req, res) {  
22     res.end(`Hello, ${req.query.name}`);  
23 }  
24
```

10-16

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

1. GET method 以 Query String 傳遞參數時，透過 & 可傳遞多項參數
2. 當有加上 name 時，就會顯示 hello XXXX
3. 這也是很多登入驗證的實現方式
4. 將原有的程式碼，加入15~19行，就可以實現功能
5. 程式碼執行順序：紀錄時間、檢查密碼、檢查名字、顯示首頁
6. 可以嘗試把 15~19拿掉，只輸入密碼
<http://127.0.0.1:8888/?password=apple?password=apple> 看看會發生甚麼事情 (Hello, undefined)

【Key Points】：

- 留意執行順序:
1. 後臺紀錄時間 (4~7)
 2. 檢查有沒有密碼 (9~13)
 3. 檢查有沒有名字 (15~19)
 4. 顯示首頁 (21~23)

Summary 〈 精華回顧 〉

- 透過 Router 的概念簡化主程式
- app.get 函式參數可以寫成函式名稱
- 路由函式可以有第三個參數，next
- 可以連續用好幾個函式，依序執行
- Routers 是 Express.js 原生的功能
- 可以將檔案區分開來，未來更好管理
- 用 module.exports = router 匯出路由
- 再用 require 把寫好的 Router 匯入



10-17

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

回顧一下前面的內容

學會如何透過 Router 的概念簡化主程式，
並且知道可以將函式放在 app.get 的第二個參數。

函式可以有第三個參數，next，
可以連續用好幾個函式，依序執行。

Routers 是 Express.js 原生的功能，
最主要是可以將檔案區分開來，在未來專案龐大時更好管理。

最後，
用 module.exports = router 可以匯出路由，
再用 require 把寫好的 Router 匯入。

【Key Points】：

透過 Router 的概念簡化主程式
路由函式可以有第三個參數，next

用 module.exports = router 可以匯出路由,再用 require 把寫好的 Router 匯入

線上程式題

- **10-1 建立路由模組檔案**

請問，如何建立路由模組檔案？

- **10-2 如何掛載路由模組**

`myRoute.js`是一個路由模組，如何掛載進來使用呢？

- **10-3 路由函式、路由模組混用之時的執行順序**

請寫一支程式，實驗一下：路由函式、路由模組混用之時的執行順序。

10-18



題目名稱: 10-1 建立路由模組檔案

內容說明:

請問，如何建立路由模組檔案？

題目名稱: 10-2 如何掛載路由模組

內容說明:

`myRoute.js`是一個路由模組，如何掛載進來使用呢？

題目名稱: 10-3 路由函式、路由模組混用之時的執行順序

內容說明:

請寫一支程式，實驗一下：路由函式、路由模組混用之時的執行順序。

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】：

10-1 建立路由模組檔案

10-2 如何掛載路由模組

10-3 路由函式、路由模組混用之時的執行順序

課後練習題(Lab)

- **情節描述:**

本次練習假設您已經安裝 Node.js 環境於 Windows 電腦平台，並且悉知如何安裝 Express 模組套件。我們將透過已學會的 Routers 處理方式，做更進階的運用。

- **預設目標:**

學會使用模組化路由。

- **Lab01: 開發路由模組**

- **Lab02: 保留路由使用紀錄**

Estimated time:

20 minutes

- 完成後的程式與檔案，請參考 Example 資料夾的內容。

10-19



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

【情節描述】

本次練習假設您已經安裝 Node.js 環境於 Windows 電腦平台，並且悉知如何安裝 Express 模組套件。我們將透過已學會的 Routers 處理方式，做更進階的運用。

【預設目標】

學會使用模組化路由。

關鍵程式:

```
const myRoute = require('./myRoute');
app.use(function (req, res, next) {
  console.log("Time:" + Date());
  next();
});
app.use("/apple", function (req, res, next) {
  console.log("有人在用 apple 的模組");
  next();
});
app.use("/apple", myRoute);
```

【Key Points】:

Lab01: 開發路由模組

Lab02: 保留路由使用紀錄

app.use("/apple", myRoute);

範例程式使用說明

- 範例程式資料夾: Module_10_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://127.0.0.1:8888/apple/today`

10-20



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://127.0.0.1:8888/apple/today`

【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」