



# Routes處理



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

- 9-1: 使用變數代稱設定路由
- 9-2: 使用 wildcard
- 9-3: 使用 Regular Expression 設定路由



9-1

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

我們即將一起學習：

- 為何需要變數代稱
- 變數代稱的語法
- 甚麼是 wildcard
- Wildcard語法解析
- 何謂 Regular Expression
- 使用 regex101

【Key Points】：

使用變數代稱設定路由

使用 wildcard

使用 Regular Expression 設定路由

## 9-1：使用變數代稱設定路由

- 為何需要變數代稱
- 變數代稱的語法
- 如何解讀
- 其他範例



designed by freepik



designed by freepik

9-2

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

- 本知識點的內容涵蓋：
- 變數路由的原理
- 為何需要變數代稱
- 使用 wildcard 寫 route 的語法
- 以 req.params 解讀內容
- 以情境範例進行示範

### 【Key Points】：

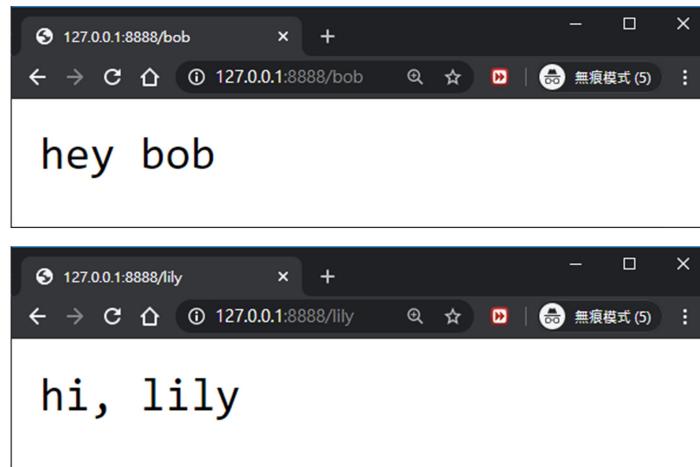
變數路由的原理

使用 wildcard 寫 route 的語法

以 req.params 解讀內容

# 為何需要變數代稱

- 依據用戶端送來的網址內容，分別對應不同的內容或功能
- 變數代稱是網址路徑的一部分



9-3

- 可依據用戶端送來的網址內容，分別對應不同的內容或功能。
- 例如我們平常常用的短網址，就利用亂碼來分配到不同網頁。
- 也可直接當作網頁的秘密通行證。
- 變數代稱是網址路徑的一部分。Express 不需要透過 GET 的查詢參數或 POST 來的各欄位內容，從網址可直接取得用戶端送來的資訊。
- 下方是兩個效果圖，若使用 bob 就會看到 hey bob；若輸入 lily，則看到 hi, lily。

## 【Key Points】：

依據用戶端送來的網址內容，分別對應不同的內容或功能

例如短網址

變數代稱是網址路徑的一部分

# 變數代稱的語法

- 使用冒號和變數名稱

`app.get("/:name", ...)`

- 瀏覽器分別開啟

— `localhost:8888/bob`

— `localhost:8888/lily`

```
index.js > ...
1  const express = require('express');
2  const app = express();
3
4  app.get('/:name', function (req, res) {
5      let name = req.params.name;
6      if (name === `bob`)
7          res.end(`hey bob`);
8      else
9          res.end(`hi, ${name}`);
10 });
11
12 app.listen(8888);
```

9-4



- `localhost` 代表本機 ip 位置 (127.0.0.1)。
- 冒號後面的字是埠號，請避免使用數字以外的符號，否則會導致解析錯誤。
- 使用 `req.params.XXXX` 取出值。舉例來說，如果網址用到 `:name`，程式使用 `req.params.name` 取出值；如果改用到 `:who`，則使用 `req.params.who` 取出值。
- 若與其他 route 衝突，有優先順序之分。
- 不必額外安裝模組套件，屬於 `express` 的原生功能。

## 【Key Points】：

`localhost` 代表本機

`req.params.XXXX` 取出值

務必指出冒號的作用，並且提示冒號後面的名稱讓學員有印象，後來的程式要對應這個名稱。

# 如何解讀

- 以 `app.get("/:name", ...)` 為例，  
在瀏覽器輸入：`http://localhost/bob`  
**Express** 會將網址「/」後面的文字，代入到變數 `name`
- 後端的程式，透過  
`req.params.name` 取出文字內容

```
index.js > ...
1  const express = require('express');
2  const app = express();
3
4  app.get('/:name', function (req, res) {
5      let name = req.params.name;
6      if (name === 'bob')
7          res.end(`hey bob`);
8      else
9          res.end(`hi, ${name}`);
10 });
11
12 app.listen(8888);
```

9-5



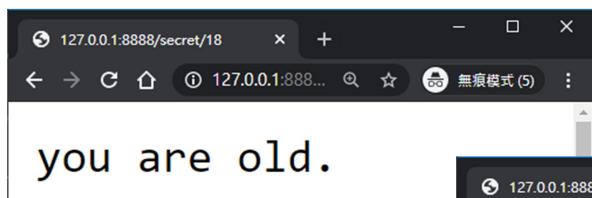
- Name 這個單字不是固定的，可以自行替換。舉例來說，如果網址用到 `:name`，程式使用 `req.params.name` 取出值；如果改用到 `:who`，則使用 `req.params.who` 取出值。
- 若與其他 route 衝突，有優先順序之分。
- 推薦使用 `let` 語法
- 使用三個等於(`==`)，表示資料型態相同而且內容相等
- 使用 `if else` 來分別顯示不同畫面
- Port 若是衝突，請修改範例12行的 8888，改成其他數字

## 【Key Points】：

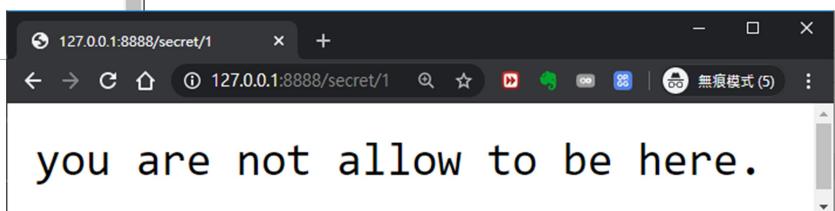
- 請一定要提到 `req.params` 後面為什麼接 `name`
- `name` 可以換成 `who`，但若是如此，之前的程式就必須改成 `:who` 才對得上
- Port 若是衝突，請修改範例12行的 8888，改成其他數字

## 其他範例

- 做出年齡判斷
- Age  $\geq 18$  看到訊息
- Age  $< 18$  無法存取



```
1 const express = require('express');
2 const app = express();
3
4 app.get('/secret/:age', function (req, res) {
5   let age = req.params.age;
6   if (age >= 18)
7     res.end(`you are old.`);
8   else
9     res.end(`you are not allow to be here.`);
10 });
11
12 app.listen(8888);
```



9-6

- 以範例程式來說，在瀏覽器輸入 `http://127.0.0.1:8888/secret/18`，會看到「you are old」。
- 當輸入  $\geq 18$  時，就會看到「you are old」訊息。
- 簡單地說，以 JavaScript 的 if 敘述比對大小。
- 網址在主目錄之後的路徑，例如本例的 secret 都可以換
- 應用範圍很廣泛，例如 PChome 用這招來瀏覽不同商品 (<https://24h.pchome.com.tw/prod/DGBQ4G-A9009VS6O>)

### 【Key Points】：

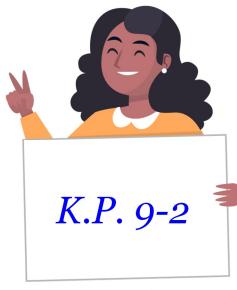
瀏覽器輸入 `http://127.0.0.1:8888/secret/18`

請導覽一下 if 那邊的程式

應用範圍很廣泛

## 9-2：使用 wildcard

- 甚麼是 wildcard
- 語法
- 如何解讀
- 情境範例



designed by freepik



designed by freepik

9-7

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

- 三個路由 /Apple 、 /Alice 、 /Amanda ，都是 A 開頭
- 在設計程式時，可以考慮寫成 app.get("/A\*", ...) ，利用 wildcard 讓程式更精簡優雅
- 通配符是一段「吻合配對」的翻譯語法
- 用一條規則匹配多個字串
- 不過，語法有些「眉角」，請留意：
  - ? 代表問號的前一個字 可有可無
  - \* 代表任意數量的字元
  - + 代表一個或一個以上

【Key Points】：

甚麼是 wildcard

Wildcard語法解析

以情境範例進行說明

# 甚麼是 wildcard

- **wildcard** 通配符，用來「吻合配對」
- 以一條規則通吃多個字串  
例如 ab?c 可以符合 abc 、 ac
- 功能完整富有彈性，另有很多種符號可以使用: + ? ( ) \*

9-8



- 通配符是一段「吻合配對」的翻譯語法
- 一條規則匹配多個字串
- 語法類似 Regular Expression
- 功能完整富有彈性，另有很多種符號可以使用: + ? ( ) \*
- 這部份很容易寫錯，要多加留意

## 【Key Points】：

不妨反過來說明，如果沒有 wildcard 功能，會有多麻煩。

一條規則匹配多個字串

提醒學員留意 \* ? + 的差別

# Wildcard語法

- 使用 ab?c 採用 wildcard 寫 route
- 瀏覽器分別開啟
  - localhost:8888/ac
  - localhost:8888/abc
- 發現都對應到同個 route
- 上下兩個效果相等，更簡潔

```
1 const express = require('express');
2 const app = express();
3
4 app.get('/ab?c', function (req, res) {
5   |   res.end(`Hello world`);
6 });
7
8 app.listen(8888);
```

```
1 const express = require('express');
2 const app = express();
3
4 app.get('/ac', function (req, res) {
5   |   res.end(`Hello world`);
6 });
7
8 app.get('/abc', function (req, res) {
9   |   res.end(`Hello world`);
10 });
11
12 app.listen(8888);|
```

9-9



- 將 wildcard 寫在 route 的第一個參數，當作配對依據
- 將 server 跑起來後，用瀏覽器分別開啟兩段網址
- 根據 wildcard 的規則 ac, abc 皆符合 ab?c，所以會對到同一個 route
- 只要寫一個 route 就能同時符合多個網址
- 精簡程式碼、減少重複的 code

## 【Key Points】：

使用wildcard如何讓程式更精簡優雅

ac, abc 皆符合 ab?c，所以會對到同一個 route

以情境範例進行說明

# Wildcard語法解析

- ? 代表問號的前一個字，該字可有可無
- ac, abc 都符合 ab?c 的規則
- \* 代表任意數量的字元
- + 代表一個或一個以上

```
1 const express = require('express');
2 const app = express();
3
4 app.get('/ab?c', function (req, res) {
5   |   res.end(`Hello world`);
6 });
7
8 app.listen(8888);
```

```
1 const express = require('express');
2 const app = express();
3
4 app.get('/ac', function (req, res) {
5   |   res.end(`Hello world`);
6 });
7
8 app.get('/abc', function (req, res) {
9   |   res.end(`Hello world`);
10 });
11
12 app.listen(8888);|
```

9-10



- ? 代表問號的前一個字 可有可無
- 在和 ab?c 比對後 ac、abc 都符合，所以才會在同個 route
- 以 /ab+cd 為例，abcd、abbcd、abbcd 都符合
- 以 /ab\*cd 為例，abcd、abxyzcd、ab1234cd 都符合
- 寫這部分程式必須多加留意，很容易匹配出現問題
- 更完整的說明，請參考 <https://expressjs.com/zh-tw/guide/routing.html>

## 【Key Points】：

- ? 代表問號的前一個字 可有可無
- \* 代表任意數量的字元
- + 代表一個或一個以上

## 情境範例

- 使用 wildcard 只用一個 route 讓 Apple, Alice, Amanda 都符合

```
JS index.js > ...
1 const express = require('express');
2 const app = express();
3
4 app.get('/A*', function (req, res) {
5   res.end(`name start with A.`);
6 })
```

9-11



- Apple、Alice、Amanda，都是 A 開頭
- 因為字數比較多變化，所以使用星號，以便符合全部A開頭的文字
- 使用 A\* 即可讓Apple、Alice、Amanda都被包含
- 可能會有奇怪的人名也符合條件，例如 Alic168。更精細的作法，本模組的下一個知識點馬上就會教到。
- 請同學另外想想有沒有其他適合使用Wildcard的例子

### 【Key Points】：

- 因為字數比較多變化，所以使用星號，以便符合全部A開頭的文字
- `app.get("/A*", ...)`
- 請同學另外想想有沒有其他適合使用Wildcard的例子，例如產品料號

## 9-3: 使用 Regular Expression 設定路由

- 何謂 Regular Expression
- 讓 wildcard 更進階
- 使用 regex101
- 情境範例



designed by freepik



designed by freepik

9-12

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

本知識點的主要內容如下：

- 簡介 Regular Expression
- Regular Expression 如何讓 wildcard 更進階
- 說明與應用 Regular Expression 語法
- 使用 regex101 工具程式
- 透過情境範例進行說明

### 【Key Points】：

多配合實例比較容易讓學員了解 Regular Expression

操作使用 regex101 工具程式

不妨分階段示範，例如 E-mail，先從 \w+@\W+ 開始，再慢慢演化成完整的版本

# 何謂 Regular Expression

- Regular Expression 正則表達式 / 正規表達式
- 有更多樣化的符號，可以匹配比 wildcard 更精細
- 應用層面非常廣泛
- 其他介紹 (MDN) :
- [https://developer.mozilla.org/zh-TW/docs/Web/JavaScript/Guide/Regular\\_Expressions](https://developer.mozilla.org/zh-TW/docs/Web/JavaScript/Guide/Regular_Expressions)

9-13



- Regular Expression 翻譯為正則表達式、正規表達式
- Regular Expression 定義更多符號與規則，使得匹配文字可以做到更精細的地步
- 一般在做使用者輸入資料驗證時，都會使用到
- 例如驗證手機號碼、時間格式、電話號碼等等
- 更完整詳細的用法，請參考 MDN，裡面有所有 JavaScript regex 的用法 (Google 搜尋: mdn regex)

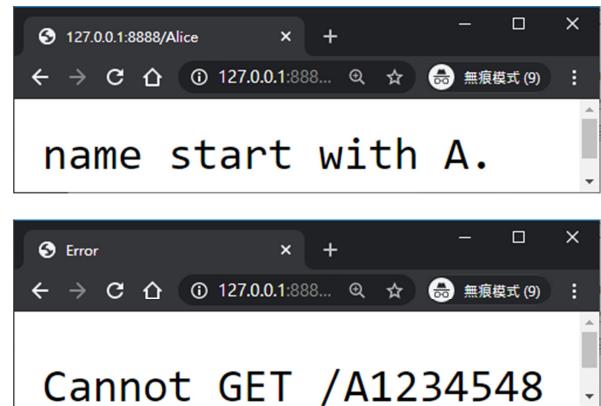
## 【Key Points】:

Regular Expression 的中文名詞翻譯並不統一  
匹配文字可以做到更精細的地步  
應用層面非常廣泛

# 讓 wildcard 更進階

- 設計程式成: `app.get("/A*", ...)`，不失為精簡優雅的寫法
- 但「`A*`」會使 `A123, ALONE` 都符合，如果只要字母呢？
- 請試試功能更強大的：  
**Regular Expression**

```
1 const express = require('express');
2 const app = express();
3
4 app.get(/A[a-z]+/, function (req, res) {
5   res.end(`name start with A.`);
6 });
7
8 app.listen(8888);
```



9-14

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

- 三個路由 `/Apple`、`/Alice`、`/Amanda`，都是 A 開頭
- 在設計程式時，可以考慮寫成 `app.get("/A*", ...)`，利用 wildcard 讓程式更精簡優雅
- 但是，因為「`A*`」的範圍很寬，可能會有奇怪的人名也符合條件，例如 `Alic168`
- 改用 `/A[a-z]+` 這樣的 Regular Expression 之後，可完全符合「英文名字」的規則（A開頭，後面接續小寫的英文字母）
- 當我們連進 `/A1234548` 之後，因為不符規格（因為只允許`[a-z]`）而不會對應到我們的 route
- 按照 Regular Expression 的語法，在中括號內註記可以接受的字元範圍，例如: `[0-9]`
- 數字可以寫成「`\d`」，英數字可以寫成「`\w`」
- 有關於數量符號：
  - `?` 零個或一個
  - `+` 一個以上
  - `*` 零個以上
  - `{n, m}` `n`至`m`個，例如: `\d{3,5}` 表示三個至五個數字

## 【Key Points】：

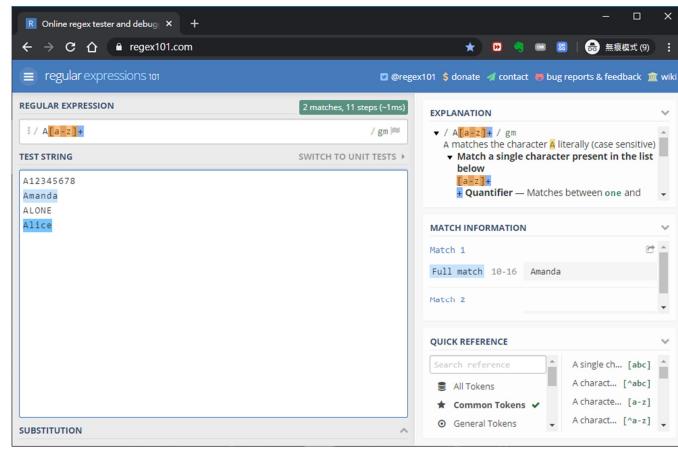
指出 `A*` 的缺點

針對 `/A[a-z]+`，請說明一下 `[a-z]` 的意思

因為只允許`[a-z]`，所以 `/A1234548` 不符規格

# 使用 regex101

- Regex101 的網址: <https://regex101.com/>
- Regular Expression 語法複雜，很容易出錯，用 regex101 幫助我們撰寫與測試
- 透過 regex101 先做測試，可以確保正確性，節省許多程式除錯時間



9-15

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

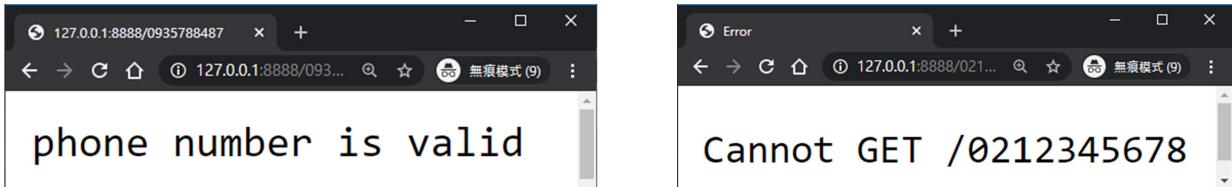
- Regular Expression 語法十分多樣複雜
- 直接在 JavaScript 撰寫 Regular Expression 時，出現問題時，並不容易除錯
- Regex101 是一個協助我們撰寫與測試 Regular Expression 的網路工具程式
- 透過 regex101 先做測試，可以確保正確性，節省許多程式除錯時間
- Regex101 的網址: <https://regex101.com/>

## 【Key Points】：

Regex101 是一支工具程式  
請示範一下 regex101 的用法  
透過 regex101 可節省開發除錯時間

## 情境範例

- 使用 Regular Expression 檢查是否是手機號碼



```
1 const express = require('express');
2 const app = express();
3
4 app.get(/09[0-9]{8}/, function (req, res) {
5   res.end(`phone number is valid`);
6 });
7
8 app.listen(8888);
```

9-16

- 最常使用在配對電話號碼
- 若要配對手機號碼，開頭必須是 09，其餘為”八個數字”
- 使用 [0-9] 配對數字，或是可以使用 \d 也相同
- {8} 裡面的8代表重複八次; {3, 5} 則是3到5個
- 這樣就可以判斷台灣的手機號碼了

【Key Points】：

[0-9] 可用 \d 代替  
{8} 表示固定8個  
{3, 5} 則是3到5個

# Summary 〈精華回顧〉

- 用不同的匹配方式精簡程式碼
- 透過冒號 ( :) 可以將網址的區段文字存成變數
- 使用 wildcard 匹配更多文字，易學易用，功能簡單
  - 疑問號 (?) 代表「可有可無」
  - 米字號 (\*) 代表「任何文字」
- Regular Expression 有更精細地規則
  - [a-z] 代表所有的「小寫英文字元」
  - [0-9] 代表所有的「數字」
  - 加號 (+) 代表「一個或以上」



9-17

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

回顧一下我們剛學到的內容：

透過冒號 ( :) 可以將網址的區段文字存成變數

使用 wildcard 匹配更多文字

問號 (?) 代表「可有可無」

米字號 (\*) 代表「任何文字」

Regular Expression 有更精細地規則

[a-z] 代表所有的「小寫英文字元」

[0-9] 代表所有的「數字」

加號 (+) 代表「一個或以上」

【Key Points】：

使用 wildcard 易學易用，功能簡單

Regular Expression 有更精細地規則

留意各個符號的作用

# 程式練習題

- **題目1: 如何使用變數代稱設定路由**

針對使用者可能輸入一些類似但內容不盡相同的網址，例如：

`http://www.youComp.com.tw/welcome/Chien`

`http://www.youComp.com.tw/welcome/Wang`

`http://www.youComp.com.tw/welcome/Lin`

- **題目2: 承上，如何讀取變數代稱的內容**

- **題目3: Regular Expression 練習**

9-18



## 題目：如何使用變數代稱設定路由

內容說明：

針對使用者可能輸入一些類似但內容不盡相同的網址，例如：

`http://www.youComp.com.tw/welcome/Chien`

`http://www.youComp.com.tw/welcome/Wang`

`http://www.youComp.com.tw/welcome/Lin`

如何以變數代稱設定路由？

## 題目：如何讀取變數代稱的內容

內容說明：

有一段程式的內容如下：

```
app.get("/welcome/:name", function (req, res) {  
    // ...  
});
```

如何讀取變數代稱 `:name` 的內容

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】：

使用變數代稱

如何讀取變數代稱的內容

練習Regular Expression

# 課後練習題 (Lab)

- **情節描述:**  
本 Lab 是假設在已經安裝 Node.js 環境的 Windows 電腦下，您已學過如何安裝 Express.js。透過剛才學到的 route 處理方式，依照規則分配「HTTP請求」到不同的 route 上。
- **預設目標:**  
在一支程式中，做出兩個不同的頁面。
- **Lab01: 早安您好**  
透過剛才學到的 route 處理方式，依照規則分配「HTTP請求」到不同的 route 上。
- **Lab02: Email 檢查**  
以Regular Express判斷路由

Estimated time:  
**20 min.**

9-19



## 【情節描述】

本 Lab 是假設在已經安裝 Node.js 環境的 Windows 電腦下，您已學過如何安裝 Express.js。透過剛才學到的 route 處理方式，依照規則分配「HTTP請求」到不同的 route 上。

## 【預設目標】

在一支程式中，做出兩個不同的頁面。

Lab01: 早安您好

Lab02: 以Regular Express判斷路由

關鍵程式:

```
app.get("/welcome/:name", function (req, res) {  
    let name = req.params.name;  
    res.end(`Welcome ${name}, have a good day.`);  
});  
app.get(/[a-z0-9]+@[a-z0-9]+\.[a-z0-9]+/, function (req, res) {  
    res.end(`your email is correct.`);  
});
```

## 【Key Points】:

依照規則分配「HTTP請求」到不同的 route

```
app.get("/welcome/:name", function () ...);  
app.get(/[a-z0-9]+@[a-z0-9]+\.[a-z0-9]+/, function () ...)
```

# 範例程式使用說明

- 範例程式資料夾: Module\_09\_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/welcome/人名`

9-20



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://localhost:4000/welcome/人名`

【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