



取得POST參數



designed by freepik

Estimated time:

50 min.

 資訊工業策進會 Institute for Information Industry

【Key Points】：

學習目標

- 15-1: 使用 **body-parser** 套件
- 15-2: 全域的使用方式
- 15-3: Postman 測試



15-1

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

我們即將一起學習：

- HTTP如何運作
- 常見的 HTTP 請求
- 安裝 body-parser 套件
- 使用 body-parser
- 以 app.locals 全域存取資料
- 當次請求的各函式以 res.locals 分享資料
- 以 app.set() 存入資料
- 以 app.get() 讀出資料
- 安裝 Postman
- 發送 GET 請求
- 發送 POST 請求

【Key Points】：

15-1: 使用 body-parser 套件

15-2: 全域的使用方式

15-3: Postman 測試

15-1: 使用 body-parser 套件

- HTTP如何運作
- 常見的 HTTP 請求
- 安裝 body-parser 套件
- 使用 body-parser



designed by freepik



designed by freepik

15-2



- 之前的課程，只能解析帶在 URL 上面的資料
- 不過，有些資料並不適合顯示在網址上面，例如：密碼
- 接下來，我們會學習如何使用 body-parser 套件來解析來自 POST 請求中包在 Body 的參數
- 並且銜接本模組稍後的 Postman
- 如果同學有興趣還可以用前面教過的 jquery 的 post() 方法來發送請求測試。

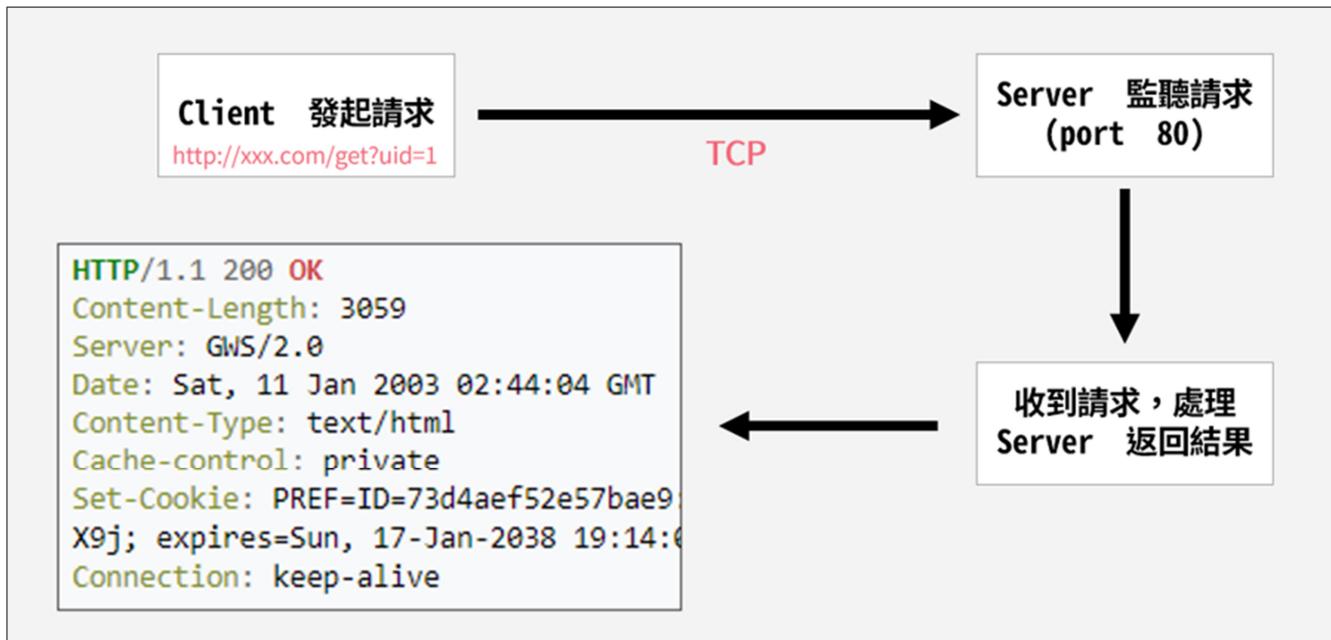
【Key Points】：

有些資料並不適合顯示在網址上面

如何使用 body-parser 套件來解析來自 POST 請求中包在 Body 的參數

銜接本模組稍後的 Postman

HTTP如何運作



15-3

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

1. 通常每個連線都是由HTTP用戶端發起一個請求
2. 建立一個到伺服器指定埠（預設是80埠）的TCP連接。
3. HTTP伺服器則在那個埠監聽用戶端的請求。
4. 一旦收到請求，伺服器會向用戶端返回一個狀態，比如“**HTTP/1.1 200 OK**”
5. 還有要返回的內容，如請求的檔案、錯誤訊息、或者其它訊息。

【Key Points】：

HTTP伺服器監聽用戶端的請求

由HTTP用戶端發起一個請求

一旦收到請求，伺服器會向用戶端返回一個狀態，比如“**HTTP/1.1 200 OK**”

常見的 HTTP 請求

GET

語義:
獲取、查詢、檢索

描述:
給我「<http://xx.com/Anna>」這個資源

大部分上網的請求都是GET，
他會把表單的參數置於URI上。
<http://xx.com/hello?name=Anna>

POST

語義:
提交、發布

描述:
提交一份資料表單
新增 or 建立資源(上傳檔案)

僅次於GET，最常見的用法，
主要參數置於Message body上。
Demo <http://live.ccu.edu.tw/admin/>

15-4



•回到 Client發起請求的部分，由上上一篇投影片有提到HTTP有幾種請求方法，而我們取其中最常見的四種：GET、POST、PUT、DELETE做介紹。

- GET:

GET是最常見的HTTP方法，用途是Client向伺服器請求某目標的資源，然後Server會回傳該資源的 representation，比如說網頁(HTML)。

因此 GET是資訊檢索最主要的機制，也是效能優化的重點(對搜尋引擎尤其重要)。

GET請求時如果有需提供參數的話，會以 key-value的方式置於 URI query的部分。也就是說你的參數值會出現在瀏覽器網址列的上頭，而且是明碼的！

- POST:

POST是僅次於GET常見的用法，跟GET不同的是它會把參數置放在Message body中，由於不會赤裸裸地出現在網址列上頭，因此是相對安全的。

但其實還是不安全，如果沒有使用HTTPS，Message body依然可以被中間人(intermediaries)輕易的察看到、甚至修改內容。

* GET、POST的用法與更詳細的介紹請查看 <https://notfalse.net/44/http-get-vs-post>

【Key Points】：

GET是最常見的HTTP方法，用途是Client向伺服器請求某目標的資源

POST跟GET不同的是它會把參數置放在Message body中

如果沒有使用HTTPS，Message body依然可以被中間人(intermediaries)輕易的察看到、甚至修改內容

安裝 body-parser 套件

- 通常在安裝 Express.js 時就會一起安裝了，如果沒有安裝的話請輸入：`npm install body-parser --save`
- 安裝完可以輸入：`npm list -s --depth=1` 檢查有沒有成功

```
C:\Work\myapi>npm list -s --depth=1
myapi@1.0.0 C:\Work\myapi
`-- express@4.17.1
  +-- accepts@1.3.7
  +-- array-flatten@1.1.1
  +- body-parser@1.19.0
    +-- content-disposition@0.5.3
    +-- content-type@1.0.4
    +-- cookie@0.4.0
    +-- cookie-signature@1.0.6
      +-- parseSignedCookie@1.0.0
```

15-5



- Express 在Node.js中是個跟 http 對應的第三方核心模組，用於處理http請求。
- 其中有個好用的套件(在Express 4.0之後獨立出來)，就是 body-parser
- body-parser屬於中介軟體(middleware)，它會攔截和解析所有的請求，協助解析用戶端POST過來的資料。
- Express 在安裝時若已自動安裝的話就可以不用在裝獨立套件
- 如果引用過程有出錯在另外安裝即可，安裝指令: `npm install body-parser --save`

【Key Points】：

Express 用於處理 http 請求

body-parser 屬於中介軟體(middleware)可以攔截和解析所有的請求

安裝指令: `npm install body-parser --save`

使用 body-parser

- 安裝完成後我們可以使用下面方式啟用 body-parser 的功能。

```
const bodyParser = require('body-parser');
const express = require('express');
const app = express();
app.use(bodyParser.json()); //處理 JSON 資料
```

- body-parser 套件主要可以解析 4 種方式的 POST 帶過來的參數。
 - bodyParser.json() → 處理 JSON 資料 (上面的範例)
 - bodyParser.urlencoded() → 處理 UTF-8 編碼的資料
 - bodyParser.raw() → 處理 Buffer 流資料
 - bodyParser.text() → 處理文字資料

15-6



body-parser 功能放在我們叫做 **middleware** 的中介層上，

使得每個路由被拜訪前，都會經過這個 **middleware**，由這個 **middleware** 先進行參數上的解析，最後在拜訪我們的路由。

換句話說就是攔截和解析所有的請求的參數。

而它解析的處理方式依照內容有不同的方法，分別是

- bodyParser.json() → 處理 JSON 資料 (上面的範例)
- bodyParser.urlencoded() → 處理 UTF-8 編碼的資料
- bodyParser.raw() → 處理 Buffer 流資料
- bodyParser.text() → 處理文字資料

在範例中我們從 app.use() 方法中呼叫 bodyParser.json()，這邊沒有另外設定路由，表示會對該 app 的所有的請求做 JSON Parser 的處理。

【Key Points】：

body-parser 功能放在我們叫做 **middleware** 的中介層上

bodyParser.json() 與 bodyParser.urlencoded() 這兩個常用

app.use()

使用 body-parser

- 剛剛使用 `app.use()` 方式呼叫的話對所有的請求資料做處理(全域)。
- 如果我們想要不同請求間個別處理的話就得用express的 `post` 或 `get` 去呼叫，這麼一來便會是區域性。

```
const bodyParser = require('body-parser');const express = require('express');
const app = express();
var jsonParser = bodyParser.json() // 解析 JSON 資料
var urlencodedParser = bodyParser.urlencoded() //解析 Form Data

app.post('/json', jsonParser, function (req, res) {
    console.log(req.body); // 可以從 req.body 中抓取解析的資料
    res.send('Form Data = ' + req.body)
})
app.post('/form', urlencodedParser, function (req, res) {
    console.log(req.body);
    res.send('JSON Data = ' + req.body)
})
app.listen(3000);
```

進會 INSTITUTE FOR INFORMATION INDUSTRY

- 我們把 `json` 跟 `form` 資料請求分別用兩個 POST 路由分開處理。
- 因為資料的性質不同，所以得宣告不同的 `bodyParser` 來使用。
- 一旦收到對應路由的請求後就會觸發不同的 Parser 來解析
- 一旦解析完成我們就可以在後續的程式中從 `req.body` 中取得解析完的 `object`
- 使用 `app.use()` 方式呼叫的話對所有的請求資料做處理(全域)。
- 如果我們想要不同請求間個別處理的話就得用express的 `post` 或`get` 去呼叫，這麼一來便會是區域性。

【Key Points】：

使用 `app.use()` 方式呼叫的話對所有的請求資料做處理(全域)

想要不同請求間個別處理的話就得用express的 `post` 或`get` 去呼叫，這麼一來便會是區域性

本例，`json` 跟 `form` 資料請求分別用兩個 POST 路由分開處理

15-2：全域的使用方式

- 以 `app.locals` 全域存取資料
- 當次請求的各函式以 `res.locals` 分享資料
- 以 `app.set()` 存入資料
- 以 `app.get()` 讀出資料



designed by freepik



designed by freepik

15-8

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

- 全域變數能少用就少用，因為有幾個缺點：記憶體會使用較多、變數汙染、變數值難管理。
- express的app instance有一個屬性：`locals`。由於app是全域變數；所以 `app.locals` 也就可以全域使用，生命週期是整個伺服器運作期間都會存在。
- express 提供 `res.locals` 讓開發者能夠使用該屬性存放需要的值，不過生命週期僅止於該次請求期間而已。
- `app.set()`可以傳入兩個參數，第一個是key，第二個是value。
- `app.get()`可以透過 key 取得之前存入的 value。

【Key Points】：

express 提供 `res.locals` 讓開發者能夠使用該屬性存放需要的值。

`app.set()`可以傳入兩個參數，第一個是key，第二個是value。

`app.get()`可以透過 key 取得之前存入的 value。

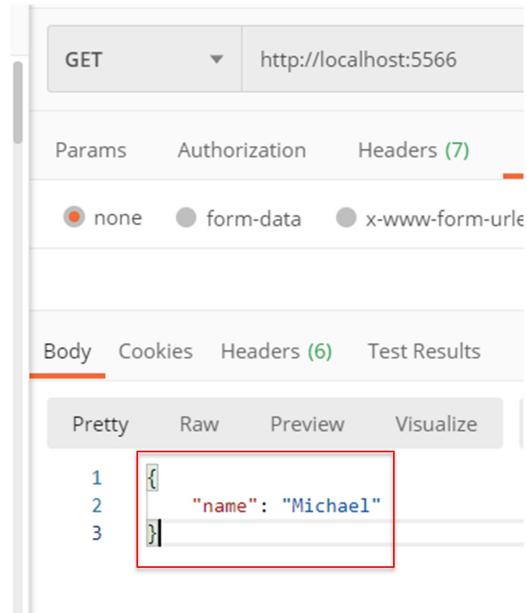
以 app.locals 全域存取資料

```
p.js > ...
const express = require('express')
const bodyParser = require('body-parser')
const app = express()

app.use(function (req, res,next) {
  app.locals='Michael'
  next()
})

app.get('/',function (req, res) {
  res.json({name:app.locals})
})

module.exports=app
```



15-9



- 全域變數能少用就少用，因為有幾個缺點：記憶體會使用較多、變數汙染、變數值難管理。
- Express 的 app instance 有一個屬性：locals。由於 app 是全域變數；所以 app.locals 也就可以全域使用。
- express 提供 app.locals 讓開發者能夠使用該屬性存放需要的值。
- app.locals 生命週期是整個伺服器運作期間都會存在。
- 直接指定 app.locals 就可以給值與取值。

【Key Points】：

app.locals 也可以全域使用

app.locals 生命週期是整個伺服器運作期間都會存在

全域變數能少用就少用

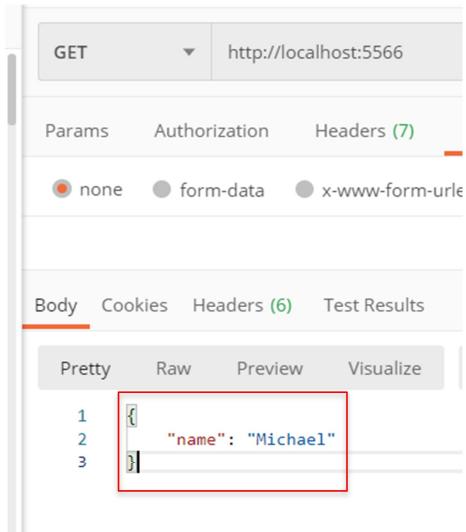
當次請求的各函式以 `res.locals` 分享資料

```
s > JS index.js > ...
const express = require('express');
const router = express.Router();
/* GET home page. */

router.use((req,res,next)=>{
  res.locals='Michael'
  next();
})

router.get('/', function(req, res, next) {
  res.json({name:res.locals});
});

module.exports = router;
```



15-10

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

- Router 的 instance 有一個屬性: `locals`。`res.locals`，只能在當次請求使用。
- 如果擔心 `app.locals` 會有全域汙染的副作用，可以考慮使用 `res.locals`，變數汙染的風險較低些（雖然可能性還是存在）。
- `express` 提供 `res.locals` 讓開發者能夠使用該屬性存放需要的值。
- `res.locals` 的生命週期是當次請求期間。
- 直接指定 `res.locals` 就可以給值與取值。

【Key Points】：

`express` 提供 `res.locals` 讓開發者能夠使用該屬性存放需要的值
`res.locals` 的生命週期是當次請求期間
直接指定 `res.locals` 就可以給值與取值

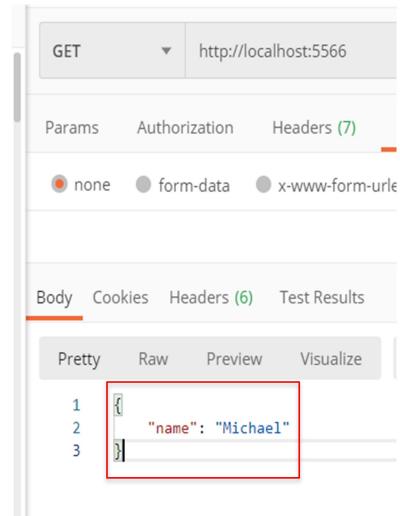
以 app.set() 存入資料

```
const express = require('express')
const app = express()

app.set('name', 'Michael');

app.get('/', (req, res, next) => {
  res.json({name: app.get('name')})
})

module.exports = app
```



15-11

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

- app有一個set()方法可以存放key與value。
- app.set()可以傳入兩個參數，第一個是key，第二個是value。
- es6語法中有新的數據結構Map，類似字典，app.set()跟Map的語法一樣都是透過set()方法設定key value。
- app.set()的value還可以放array、object、false、null、空字串等。
- app.set()的value不建議放undefined，因為一但是回傳JSON或有對app.get()做JSON處理，將會直接刪除該Object的key value。

【Key Points】：

app有一個set()方法可以存放key與value

app.set()可以傳入兩個參數，第一個是key，第二個是value

app.set()的value還可以放array、object、false、null、空字串等

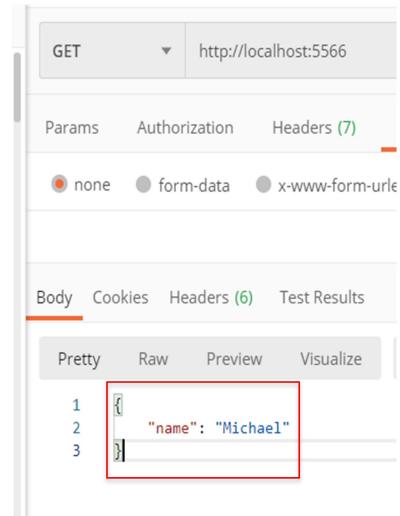
以 app.get() 讀出資料

```
const express = require('express')
const app = express()

app.set('name', 'Michael')

app.get('/', (req, res, next) => {
  res.json({name: app.get('name')})
})

module.exports = app
```



15-12

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

- `app.get()` 在早先前介紹他可以當作router handler function。
- 但其實`app.get()`可以透過key取得value。
- `app.get()` 可以只傳入一個參數，將字典的key帶入當參數。
- es6語法中有新的數據結構Map，類似字典，`app.get()`跟Map的語法一樣都是透過`get()`方法取得 value。
- `app.get()`會依據參數，回傳該物件相對應的值。

【Key Points】：

`app.get()` 在早先前介紹他可以當作router handler function

其實`app.get()`可以透過key取得value

`app.get()` 可以只傳入一個參數，將字典的key帶入當參數；依據參數，回傳該物件相對應的值

15-3: Postman 測試

- 安裝 Postman
- 發送 GET 請求
- 發送 POST 請求



designed by freepik



designed by freepik

15-13

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

剛剛我們寫好了我們的 API Server，裡面有兩個 POST 方法等著我們去調用，
不過在沒有寫前端網頁的情況下我們是否可以用其他工具來發送 HTTP 請求來做測試呢??

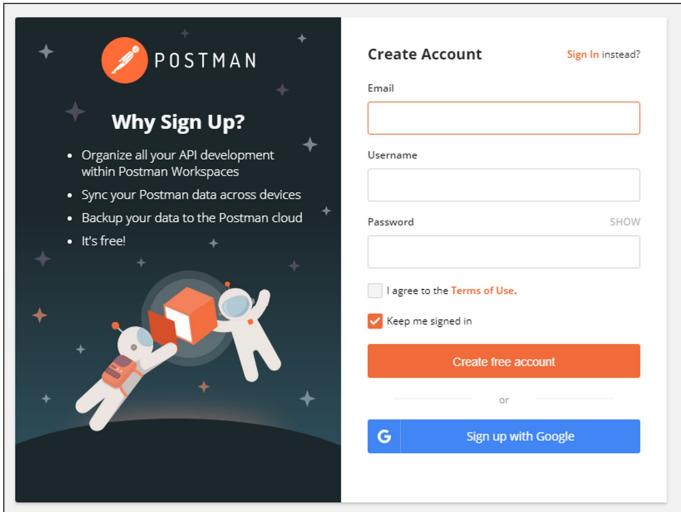
- 這邊就一定要來介紹這個測試 API 的神器 - Postman !
- 本知識點會介紹 Postman 的基本用法
- 實作發送 GET
- 實作發送 POST
- 除了 Postman 之外還有 RESTer、Firebug... 等工具

【Key Points】：

測試 API 的神器 - Postman
介紹 Postman 的基本用法
實作發送 GET 與 POST

安裝 Postman

- 到 <https://www.postman.com/downloads/> 下載桌面版的 Postman
- 下載完後點擊 exe 執行檔進行安裝，如果看到以下畫面表示完成



15-14



- Postman 是一個可以模擬 HTTP Request 的工具，其中包含常見的 HTTP 的請求方式
例如：GET、POST、PUT、DELETE
- 而它的主要功能就是能夠快速的測試你的 API 是否能夠正常的請求資料，並得到正確的請求結果。
- 除了快速測試的功能以外，Postman 還擁有非常容易使用的介面，以及 Collection 的功能，我們會先介紹 Request 的功能，並實際操作一次 GET 與 Post 這兩個 Method。
- Postman 有 Chrome 線上應用程式的版本也有桌面版本
- 這邊以桌面版本為主，老師希望可以跨平台教學的話可以請同學安裝 Chrome APP 版

桌面版：

<https://www.postman.com/downloads/>

Chrome 版：

<https://chrome.google.com/webstore/detail/postman/fhbjgbiflinjbdggehcddcbncddomop?hl=zh-TW>

【Key Points】：

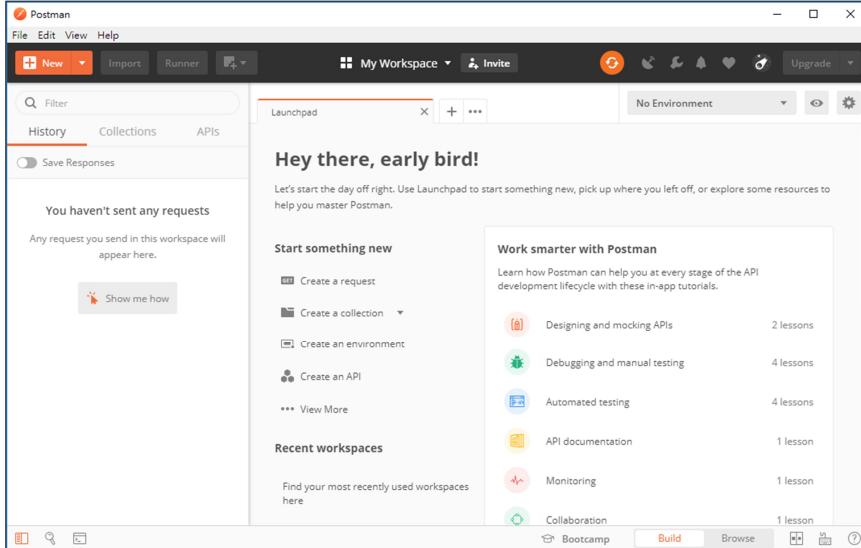
Postman 是一個可以模擬 HTTP Request 的工具

常見的 HTTP 的請求方式: GET、POST、PUT、DELETE

Postman 有 Chrome 線上應用程式的版本也有桌面版本，本課程以桌面版本為準

安裝 Postman

- Postman 使用需要註冊帳號，同學們可以使用Google帳號快速登入，或是用 Email 辦理，出現以下畫面就是登入成功了。



15-15

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

- Postman 使用需要註冊帳號
- 同學們可以使用Google帳號快速登入，或是用 Email 辦理
- 出現圖中的畫面就是登入成功了

- Postman 可以把請求資料、歷史數據...等跟帳號一同綁定，如果要再多平台或裝置間測試很方便
- Postman 有協作功能，也可以把人拉進同個工作群組

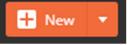
【Key Points】：

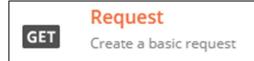
Postman 使用需要註冊帳號

使用Google帳號快速登入，或是用 Email 辦理

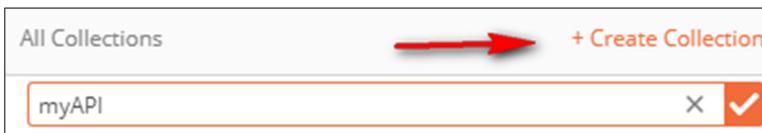
Postman 可以把請求資料、歷史數據...等跟帳號一同綁定

發送 GET 請求

- 點擊左上角的  按鈕，選擇新增一個 Request
- 在表單中填入這個請求的名稱，命名好懂就好



- 接著我們在下方新增一個目錄來存放這個Request，然後保存。



15-16



- 新增一個請求的方法就是按左上角的 New 鈕
- 選擇要新增的 Request 選項
- 接著需要輸入一些說明文字，這邊依照個人喜好設定即可
- 最後需要創建一個資料夾來保存這次的 HTTP Request
- 目錄的部分我們先暫時用 myAPI 即可，如果之後有團隊合作的需求會建議統一命名

【Key Points】：

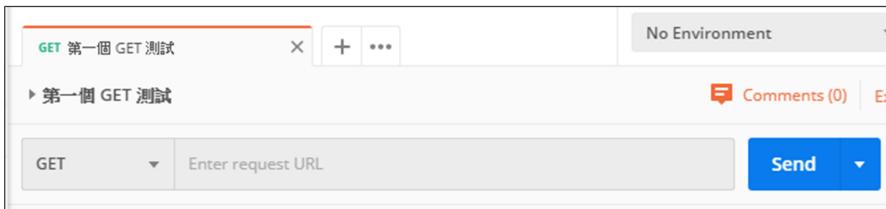
新增一個請求的方法就是按左上角的 New 鈕

選擇要新增的 Request 選項

接著需要輸入一些說明文字

發送 GET 請求

- 看到這個分頁畫面就是成功了



- 回到 app.js 我們加入一個 GET 請求的測試路由

```
app.get('/test', function (req, res) {  
    res.send(' query string = ' + req.query.name);  
});
```

- 記得要在終端機上執行 node app.js 把 server 運行起來

15-17

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

- 如果有出現一個剛剛命名的分頁代表新增成功了
- 回到 app.js 我們加入一個 GET 請求的測試路由
- 記得要到終端機上重新執行 node app.js 把 server 運行起來
- 在URL中加入 name=XXX 的查詢字串
- 成功的話就會在瀏覽器上顯示出來

【Key Points】：

回到 app.js 我們加入一個 GET 請求的測試路由
記得要到終端機上重新執行 node app.js 把 server 運行起來
在URL中加入 name=XXX 的查詢字串

發送 GET 請求

- 回到 Postman 上面，在 URL 框中輸入 <http://127.0.0.1:3000/test>



- 按下 Send 之後應該可以看到下方有 Response Body 的內容

The screenshot shows the Postman interface after sending a GET request to 'http://127.0.0.1:3000/test'. The status bar at the top indicates a successful 200 OK response with a time of 38ms and a size of 229 B. The response body is displayed below, showing the single line of text 'query_string = undefined'.

15-18

- 到 Postman 上在 URL 框中輸入 <http://127.0.0.1:3000/test>
- 按下 Send 之後把該請求送到Server
- 在 Postman 中我們可以很輕易地發送請求的 header 跟參數內容
- 也可以看到 Server 回應的狀態；內容、Cookies、Headers 等等資訊
- 這次執行結果會顯示 undefined 的原因是因為我們沒有在GET請求中加入參數

【Key Points】：

到 Postman 上在 URL 框中輸入 <http://127.0.0.1:3000/test>

按下 Send 之後把該請求送到Server

在 Postman 中我們可以很輕易地發送請求的 header 跟參數內容

發送 GET 請求

- 注意到剛剛的請求結果為 **query string = undefined**，因為我們還沒設置 **Query String**。Postman 可以很簡單的新增參數欄位

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> name	你的名子	
Key	Value	Description

- 把 **KEY** 跟 **VALUE** 打上即可加在 **URL** 上面。再次執行結果

1 query string = 你的名子

15-19

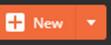
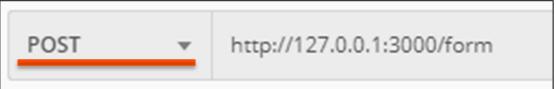


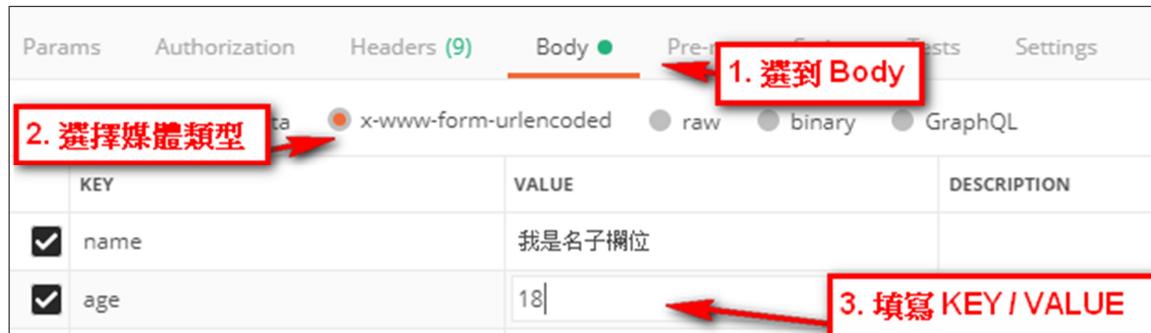
- 請求結果為 **query string = undefined**，因為我們還沒設置 **Query String**
- Postman 可以很簡單的新增參數欄位，切換到下面 **Params** 分頁下，可以直接再 **KEY/VALUE** 欄位加上數值
- 在左邊也可以輸入方便自己查看的描述欄位 (不會帶入 URL)
- 如果設置上去的話可以觀察到上方的 **URL** 欄位也跟著在變動
- 再次 **Send** 出去可以看到 **Response** 回來的內容已經是我們剛剛填入的值了，成功!

【Key Points】：

切換到下面 **Params** 分頁下，可以直接再 **KEY/VALUE** 欄位加上數值
如果設置上去的話可以觀察到上方的 **URL** 欄位也跟著在變動
再次 **Send** 出去可以看到 **Response** 回來的內容已經是我們剛剛填入的值了，成功!

發送 POST 請求

- 一樣點擊左上角的  按鈕，我們新增一個 POST 請求
- 把左邊請求方法改成 POST，填上路徑 
- 參數的部分選到 Body 分頁，類型選擇 x-www-form-urlencoded



1. 選到 Body

2. 選擇媒體類型

3. 填寫 KEY / VALUE

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> name	我是名子欄位	
<input checked="" type="checkbox"/> age	18	

15-20

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

- 跟GET一樣新增一個請求(按左上角的 New 鈕)
- 選擇要新增的 Request 選項
- 接著我們需要把預設的 GET 改成 POST 請求
- 參數的部分選到 Body 分頁，類型選擇 x-www-form-urlencoded
- 之所以要選擇 x-www-form-urlencoded 類型是因為 form 預設是用 UTF-8 的 urlencoded 格式編碼

【Key Points】：

跟GET一樣新增一個請求(按左上角的 New 鈕)

把預設的 GET 改成 POST 請求

類型選擇 x-www-form-urlencoded

發送 POST 請求

- 可以發現回應的結果已經被解析成 Object 了
- 回到終端機上可以看到剛剛傳進來的資料被解析成 Object 並且打印出來，代表 `bodyParser.urlencoded()` 的解析方法成功！

```
Form Data = [object Object]
```

```
C:\Work\myapi>node app.js
body-parser deprecated undefined extended: provide ext
:6:35
{ name: '我是名子欄位', age: '18' }
-
```

15-21



1. 再次發送請求後可以發現我們送出去的表單已經被正確解析成 object 了
2. 如果這邊希望可以顯示內容的話，可以用 `JSON.stringify(obj)`，將 obj 序列化成 JSON 格式的字串
3. 後台的在終端機上會看到 Object 的內容被印了出來
4. form 其實還可以選擇其他格式，有興趣的同學可以先查詢
5. 我們下一節課也會帶著大家實做上傳文件功能的表單

【Key Points】：

再次發送請求後可以發現我們送出去的表單已經被正確解析成 object 了
用 `JSON.stringify(obj)` 可將 obj 序列化成 JSON 格式的字串
後台的在終端機上會看到 Object 的內容被印了出來

發送 POST 請求

- 接著我們試試看 JSON 請求的部分，假如我們只把 URL 改成 /



- 執行 Send 之後，回到終端機上會發現沒有任何資料，這是因為 jsonParse 無法解析 form 的內容格式

```
C:\Work\myapi>node app.js
body-parser deprecated undefined extend
:6:35
{}
```

15-22

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

- 再來我們試試看JSON API
- 首先把URL 上的 /form 改成 /json
- 接著執行 Send 把 表單送出
- 執行 Send 之後，回到終端機上會發現沒有任何資料
- 這是因為我們前端維持用 x-www-form-urlencoded 的模式去送，後端的 jsonParse 無法解析 form 的內容格式

【Key Points】：

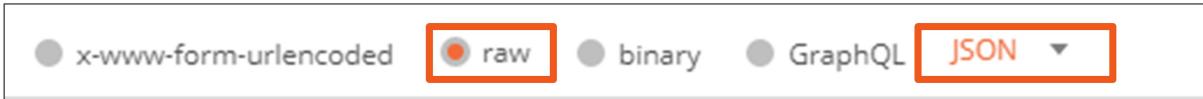
試試看JSON API

URL 上的 /form 改成 /json

前端維持用 x-www-form-urlencoded 的模式去送，後端的 jsonParse 無法解析 form 的內容格式

發送 POST 請求

- 如果要正確發送 JSON 媒體的話需要改用 raw 類型，並且在右邊選單中選擇媒體語言為 JSON 格式。



- 接著下面輸入正確的 JSON 內容

```
1 ~ {  
2   "name": "JSON 名子",  
3   "age": 18  
4 }  
5
```

- 再次送出就可以看到正確解析的結果了而且 age 正確被解析成整數類型了！

```
C:\Work\myapi>node app.js  
body-parser deprecated undefined  
:6:35  
{}  
{ name: 'JSON 名子', age: 18 }
```

15-23

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

- 因此如果要正確發送 JSON 媒體的話需要改用 raw 類型，並且在右邊選單中選擇媒體語言為 JSON 格式
- 注意這邊媒體格式一定要選對 JSON，否則還是會解析錯誤
- 另外Postman 會提醒你輸入的 JSON 格式是否正確
 - 旁邊還有 Beautify 的按鈕可以把內容縮排美化，請同學可以多多善用。
- JSON 的好處是可以傳遞字串以外的類型例如，Bool、Int、Array、Object ... 等等
- 因此在前後端都是用 JavaScript 開發得情況下會建議同學資料傳遞格式用 JSON 比較適合。

【Key Points】：

如果要正確發送 JSON 媒體的話需要改用 raw 類型，在右邊選單中選擇媒體語言為 JSON 格式
媒體格式一定要選對 JSON，否則還是會解析錯誤
前後端都是用 JavaScript 開發得情況下會建議同學資料傳遞格式用 JSON 比較適合

Summary 〈 精華回顧 〉

- HTTP如何運作
- 使用 **body-parser** 套件
- 了解 **app.locals**和**res.locals**用法
- 以 **app.set()**存入資料
- 以 **app.get()**讀出資料
- 以 Postman 測試 Web 程式



15-24

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

- 了解 HTTP 的運作方式
- **app.locals**也就可以全域使用，**app.locals**生命週期是整個伺服器運作期間都會存在
- **express** 提供 **res.locals** 讓開發者能夠使用該屬性存放需要的值，**res.locals**的生命週期是當次請求期間
- 以 **app.set()**存入資料
- 以 **app.get()**讀出資料
- Postman 是一個可以發出 HTTP 請求的測試工具。
- POST 常用於新增資料，利用 HTTP 文件的 **body** 區，攜帶想要新增的資料。

【Key Points】：

使用 **body-parser** 套件

存取全域資料

以 Postman 測試 Web 程式

線上程式題

- **15-1 如何使用 body-parser 套件模組**

請修改程式碼，使其能夠正常console.log()出req.body。

```
const bodyParser=require('body-parser');
app.use(bodyParser.jsonParser());
app.use((req,res,next)=>{
    console.log(req.body);
    next();
});
```

- **15-2 透過 app.set() 儲存 POST 的資料**

請問，Web API 的程式該怎麼透過 app.set()將 {"name":"Jack"} 存起來。

- **15-3 如何讀出之前 app.set() 存入的資料**

如何讀出之前 app.set() 存入的資料？

15-25



15-1 如何使用 body-parser 套件模組

請修改程式碼，使其能夠正常console.log()出req.body。

```
const bodyParser=require('body-parser');
app.use(bodyParser.jsonParser());
app.use((req,res,next)=>{
    console.log(req.body);
    next();
});
```

15-2 透過 app.set() 儲存 POST 的資料

請問，Web API 的程式該怎麼透過 app.set()將 {"name":"Jack"} 存起來。

15-3 如何讀出之前 app.set() 存入的資料

如何讀出之前 app.set() 存入的資料？

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】：

15-1 如何使用 body-parser 套件模組

15-2 透過 app.set() 儲存 POST 的資料

15-3 如何讀出之前 app.set() 存入的資料

課後練習題(Lab)

- **情節描述:**
網站已十分普及，後端API也朝向Restful API發展了數年，本次會注重在Restful API的POST練習，並且以電商平台產品查詢為例。
- **預設目標:**
 - 理解app.locals
 - 理解POST req.body
 - 理解Postman測試
- **Lab01:理解app.locals**
- **Lab02:理解POST req.body**
- **Lab03:理解Postman測試**

Estimated time:
20 minutes

15-26



【情節描述】

網站已十分普及，後端API也朝向Restful API發展了數年，本次會注重在Restful API的POST練習，並且以電商平台產品查詢為例。

【預設目標】

- 理解app.locals
- 理解POST req.body
- 理解Postman測試

Lab01:理解app.locals

Lab02:理解POST req.body

Lab03:理解Postman測試

完成後的程式與檔案，請參考 Example 資料夾的內容。

【Key Points】：

Lab01:理解app.locals

Lab02:理解POST req.body

Lab03:理解Postman測試

範例程式使用說明

- 範例程式資料夾: Module_15_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」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
 6. 在終端機視窗，輸入 npm start
 7. 啟動瀏覽器，連接 http://localhost:3000

15-27



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」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
6. 在終端機視窗，輸入 npm start
7. 啟動瀏覽器，連接 http://localhost:3000

【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」開啟 terminal 終端機視窗，輸入：「npm start」