

Module 7-1 –解析 QueryString

情節描述:

本 Lab 是假設在已經安裝 Node.js 環境的 Windows 電腦下，使用其內建的 `querystring` API 來解析網址中的 Query String。

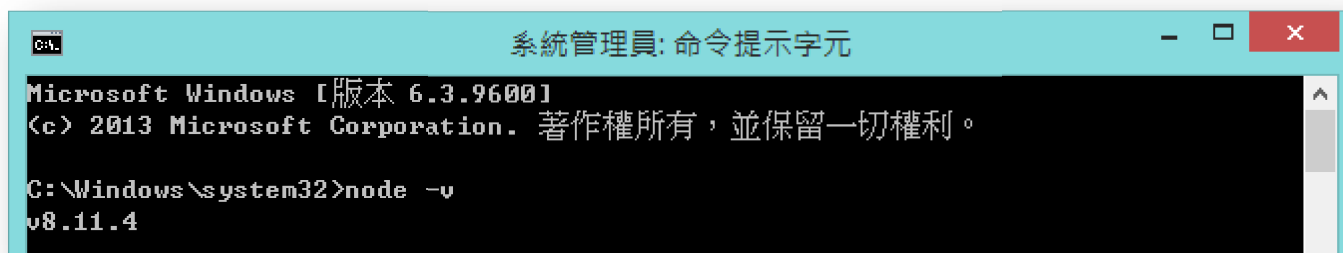
預設目標:

使用 `querystring` 中的 `parse`、`stringify`、`escape`、`unescape` 等方法，最終可完成一簡單的成績單功能。

估計時間: 15 分鐘

★★★★ Lab01: 安裝 Node.js 環境 ★★★★★

1. 先確認是否已經安裝 Node.js 環境，可在命令提示字元中輸入「`node -v`」，若能顯示版本，代表已經擁有 Node.js 環境。



```
Microsoft Windows [版本 6.3.9600]
(c) 2013 Microsoft Corporation. 著作權所有，並保留一切權利。

C:\Windows\system32>node -v
v8.11.4
```

2. 若沒顯示版本號或顯示無 `node` 指令可用，代表沒安裝過 Node.js，或者環境變數設置有問題。
3. 請直接到官網(<https://nodejs.org/en/>)下載最新版本依照 Module 1 的說明進行安裝。理論上新版本會自動設定好環境變數。
4. 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。
5. Lab 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許多專案目錄。
6. 在「命令提示字元」輸入：`mkdir myapp`，建立名為 `myapp` 的資料夾作為專案名稱。
7. 輸入：`cd myapp` 切換到該目錄

8. 輸入：`npm init` 將該資料夾轉成一個 `node` 專案。
9. 若無特殊需求，可一路按確認鍵到底。
10. 最後輸入：`yes` 按下確認鍵，將會產生 `package.json`。
11. 輸入：`npm install express --save` 安裝 `Express`。

★★★★ Lab02: 建立主要執行檔 `app.js`★★★★

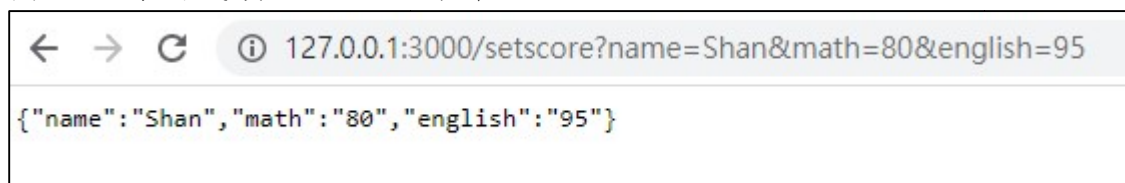
1. 在命令提示字元中輸入：`code`。開啟 `VS Code` 編輯器。
2. 點選新增檔案，輸入 `app.js`，按下確認鍵。
3. 在 `app.js` 中輸入以下程式碼並儲存

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

app.get('/setscore', function (req, res) {
  var qstr = req.url.replace("/setscore?", "");
  var myScore = querystring.parse(qstr);
  console.log("原始 Query = \n" + qstr + "\n解析 Object = ");
  console.log(myScore);
  res.send(myScore);
});

app.listen(3000);
```

4. 到「命令提示字元」輸入 `node app.js` 執行 `js`。
5. 打開瀏覽器，在網址列上輸入
「`http://127.0.0.1:3000/setscore?name=Shan&math=80&english=95`」打開網頁，可看到成績的 `JSON` 字串。



6. 回到終端機上可以看到印出的原始 `Query String` 跟解析過的成績物件。

```
C:\Work\myapi>node app.js
原始 Query =
name=Shan&math=80&english=95
解析 Object =
[Object: null prototype] { name: 'Shan', math: '80', english: '95' }
```

★★★ Lab03: 測試 stringify ★★★

1. 修改 app.js 程式碼如下，把 myScore 變數宣告成全域變數，加入 /getfix 跟 /show：

```
var querystring = require('querystring');
const express = require('express');
const app = express();
var myScore;

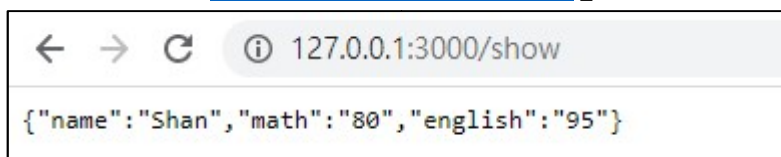
app.get('/getfix', function (req, res) {
  myScore.math = 100;
  myScore.english = 100;
  var fixQS = querystring.stringify(myScore);
  res.send(fixQS);
});

app.get('/show', function (req, res) {
  res.send(myScore);
});

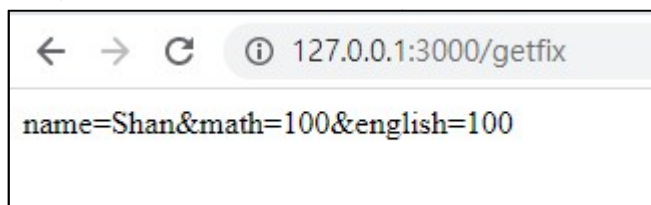
app.get('/setscore', function (req, res) {
  var qstr = req.url.replace("/setscore?", "");
  myScore = querystring.parse(qstr);
  console.log("原始 Query = \n" + qstr + "\n解析 Object = ");
  console.log(myScore);
  res.send(myScore);
});

app.listen(3000);
```

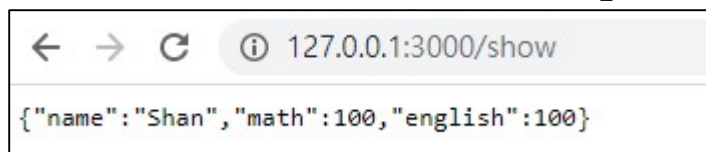
2. 回到「命令提示字元」，鍵盤按下「Ctrl + C」，將先前的服務先關閉。
3. 輸入 node app.js 再次啟動 js。
4. 回到瀏覽器，在網址列上一樣先輸入剛剛的
「<http://127.0.0.1:3000/setscore?name=Shan&math=80&english=95>」把成績加進程式里。
5. 接著輸入「<http://127.0.0.1:3000/show>」，可以看到當前的成績。



6. 接著輸入「<http://127.0.0.1:3000/getfix>」，可以使用 `querystring.stringify()` 修改當前輸入的成績，範例上是把數學跟英文都改 100 分，並把 object 序列化回查詢字串顯示在網頁上。



7. 當再次執行「<http://127.0.0.1:3000/show>」，可以看到修改後的成績。



Module 7-2 – 使用 body-parser + Postman 測試

情節描述:

本 Lab 是假設在已經安裝 Node.js 環境的 Windows 電腦下，另外安裝 Express.js 與 body-parser 套件來解析 POST Method 的 body 內容，最後用 Postman 這套 API 測試工具進行測試。

預設目標:

撰寫 GET、POST 請求類型的 API 各一個，當前端 post JSON 資料時需要被 body-parser 正確解析成物件。

最後用 Postman 來模擬前端的 API 呼叫。

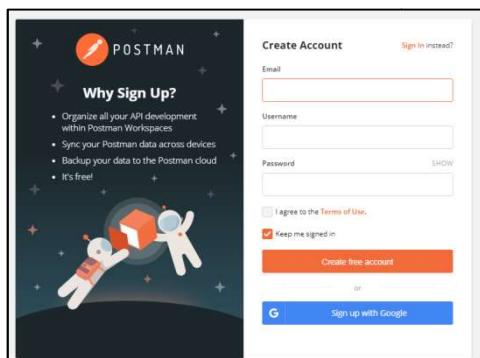
估計時間: 25 分鐘

★★★★ Lab01: 安裝 body-parser、Postman 環境 ★★★★★

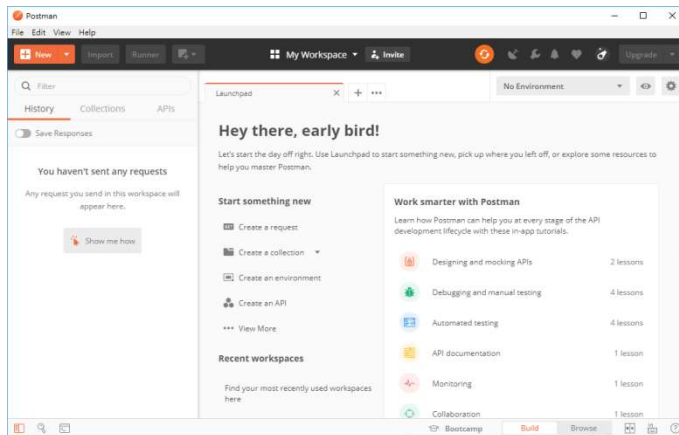
1. body-parser 通常在安裝 Express.js 時就會一起安裝了，如果沒有安裝的話請輸入：「npm install body-parser --save」。
2. 可以用「npm list -s --depth=1」，檢查套件是否有安裝成功。

```
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
```

3. 到 <https://www.postman.com/downloads/> 下載桌面版的 Postman。
4. 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。



5. Postman 使用需要註冊帳號，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。



★★★ Lab02: 建立主要執行檔 app.js + Postman 測試 ★★★

1. 在命令提示字元中輸入：code . 開啟 VS Code 編輯器。
2. 點選新增檔案，輸入 app.js，按下確認鍵。
3. 在 app.js 中輸入以下程式碼並儲存

```
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('/form', urlencodedParser, function (req, res) {
  console.log(req.body);
  res.send('Form Data = ' + req.body)
})

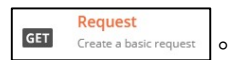
app.post('/json', jsonParser, function (req, res) {
  console.log(req.body);
  res.send('JSON Data = ' + req.body)
})

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

app.listen(3000);
```

4. 到「命令提示字元」輸入 `node app.js` 執行 `js`。

5. 打開 Postman，點擊左上角的  按鈕，選擇新增 Request



6. 在表單中填入這個請求的名稱，命名好懂就好。



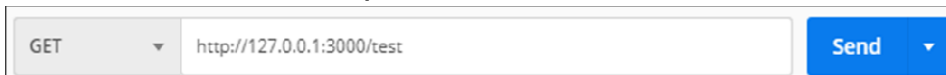
7. 接著在下方新增一個目錄來存放這個 Request，然後保存。



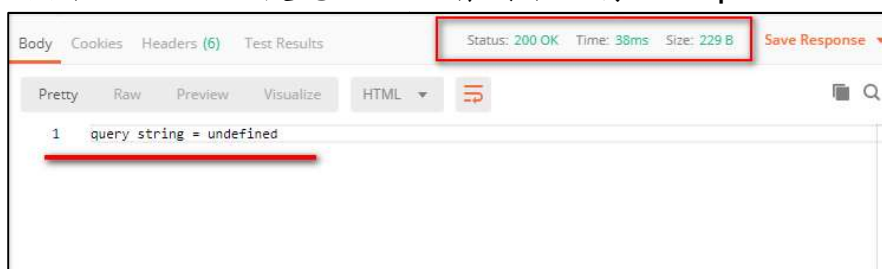
8. 看到這個分頁畫面就是成功了。



9. 在 URL 框中輸入 `http://127.0.0.1:3000/test`。



10. 按下 **Send** 之後應該可以看到下方有 **Response Body** 的內容。




11. 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。



12. 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
---	---------------------

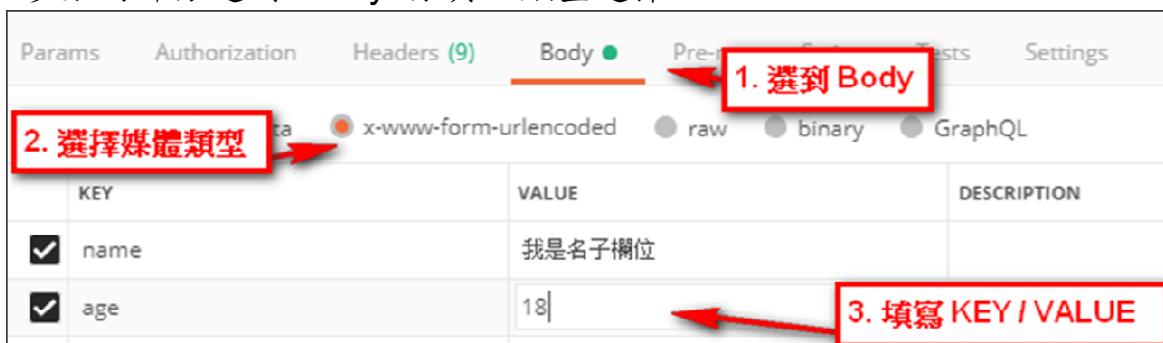
13. 接下來我們要測試 POST 請求。

14. 一樣點擊左上角的  按鈕，我們新增一個 POST 請求。

15. 把左邊請求方法改成 POST，填上路徑。

POST	http://127.0.0.1:3000/form
------	----------------------------

16. 參數的部分選到 Body 分頁，類型選擇 x-www-form-urlencoded



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

17. 可以發現回應的結果已經被解析成 Object 了。

Form Data = [object Object]

18. 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！

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

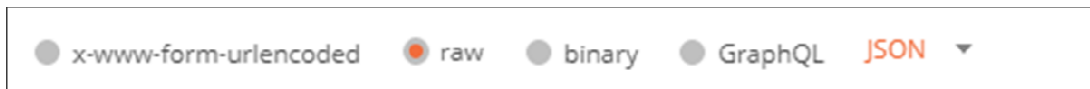
19. 接著我們試試看 JSON 請求的部分，假如我們只把 URL 改成 /json

http://127.0.0.1:3000/json	Send
----------------------------	------

20. 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。

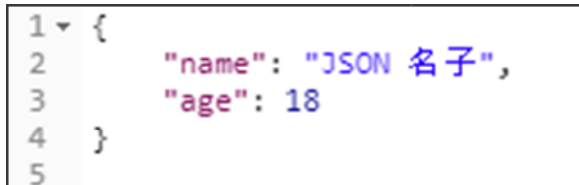
```
C:\Work\myapi>node app.js
body-parser deprecated undefined extend
:6:35
{ }
```


21. 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。



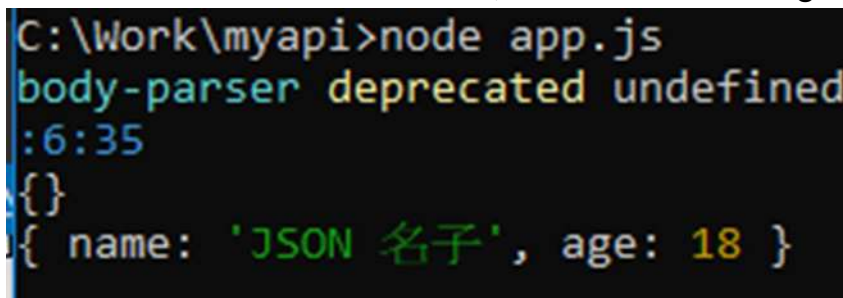
☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▼

22. 接著下面輸入正確的 JSON 內容。



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

23. 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 名字', age: 18 }  
  }  
}
```