

Module 8-1 – 表單發送資料

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 Express.js 搭建網頁應用程式，並且撰寫前端頁面向後端伺服器發送表單請求。

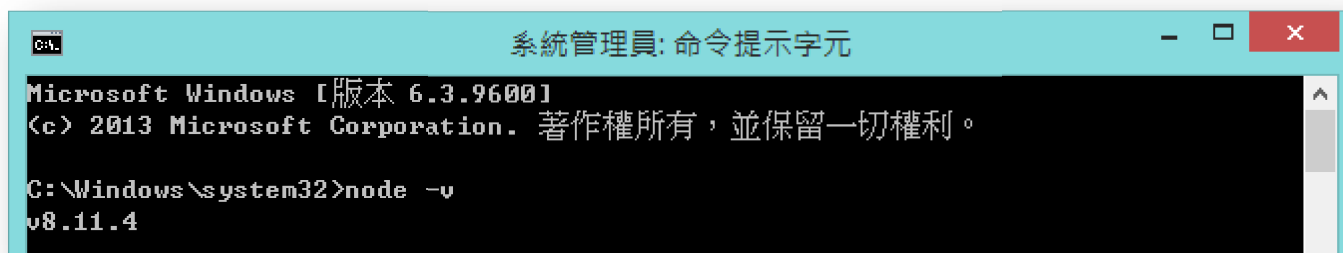
預設目標:

使用 HTML 的 `<input>` 控件完成一份表單，並且在後端應用上接收請求並解析資料。

估計時間: 20 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。



```
系統管理員: 命令提示字元
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

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

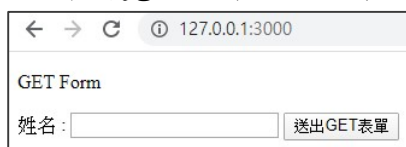
app.get("/", function(req, res) {
  res.sendFile(__dirname + '/index.html', function(err) {
    if (err) res.send(404);
  });
});

app.listen(3000);
```

4. 接著打開 `VS code`，專案目錄下新增 `index.html` 文件，並且打開編輯輸入：

```
<!DOCTYPE html>
<html>
<body>
  <p> GET Form </p>
  <form action="getdata" method="GET">
    姓名: <input name="name" type="text">
    <input type="submit" value="送出 GET 表單">
  </form>
</body>
</html>
```

5. 到「命令提示字元」輸入 `node app.js` 執行 `js`。
6. 打開瀏覽器，在網址列上輸入「<http://127.0.0.1:3000>」打開網頁，可表單畫面。



7. 剛剛我們在 `action` 跟 `method` 分別填上 `'getdata'` 跟 `'GET'`，代表這份表單會用 `GET` 請求送到 `/getdata` 路由下，因此我們的 `app.js` 需要加入對應的程式碼'

```
app.get('/getdata', function (req, res) {  
  console.log(req.query);  
  res.send('收到的資料 = ' + JSON.stringify(req.query))  
})
```

8. 重新執行 `app.js`，在瀏覽器上表單填入姓名後點擊按鈕送出
9. 可以發現 `URL` 被依照欄位內容格式化對應的 `Query String` 了

<div>GET Form</div> <div>姓名: <input type="text" value="珊"/></div> <div>送出GET表單</div>	<div>← → ↻ 127.0.0.1:3000/getdata?name=珊</div> <div>收到的資料 = {"name": "珊"}</div>
--	---

★★★ Lab03: 發送 POST 表單 ★★★

1. 因為 GET 表單會把內容當成查詢字串送出去(明碼顯示在網址列上)，因此不適合傳送密碼等敏感資訊。
2. 修改 index.html 程式碼如下，把加入一個請求類型為 POST 的表單：

```
<p> POST Form </p>
<form action="/postdata" method="POST">
  姓名：<input name="name" type="text">
  密碼：<input name="pwd" type="password">
  <input type="submit" value="送出POST表單">
</form>
```

3. 刷新瀏覽器可以看到一個新的表單

POST Form

姓名： 密碼：

4. 回到 app.js 加入對應的 /postdata 事件。

```
const bodyParser = require('body-parser');
var urlencodedParser = bodyParser.urlencoded() //解析 Form Data

app.post('/postdata', urlencodedParser, function (req, res) {
  console.log(req.body);
  res.send('收到的資料 = ' + JSON.stringify(req.body))
})
```

5. 輸入 node app.js 再次啟動 js。
6. 回到瀏覽器，在網址列上一樣先輸入剛剛的「<http://127.0.0.1:3000>」。
7. 在瀏覽器上填入 POST 表單，並且點擊送出。

POST Form

姓名： 密碼：

8. 可以發現資料不會出現在網址列上面了，而且後端也能被正確解析成 Object。

← → ↻ ⓘ 127.0.0.1:3000/postdata

收到的資料 = {"name":"我的名子","pwd":"mytestpassword"}

```
body-parser deprecated undefined extended: p
:6:35
{ name: '我的名子', pwd: 'mytestpassword' }
```

Module 8-2 – 使用 multer 上傳文件 + 過濾檔案

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 Express.js 搭建網頁應用程式，並且撰寫前端頁面向後端伺服器發送檔案上傳表單請求，。

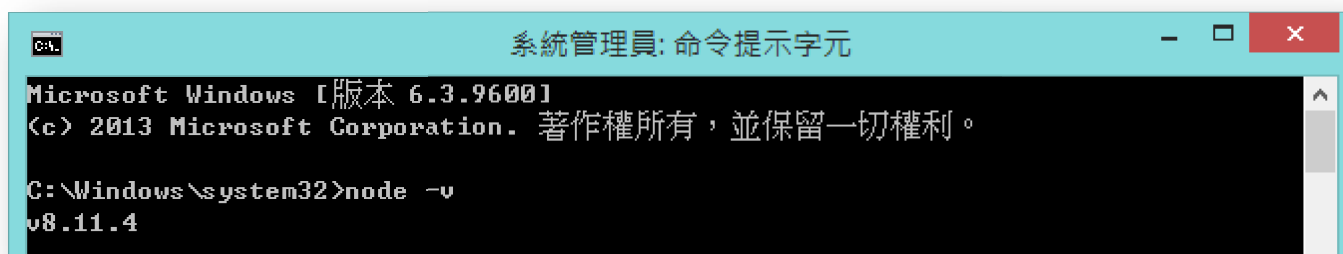
預設目標:

藉助 express、multer 實現檔案上傳功能，並且實作前端與後端形式的文件過濾功能。

估計時間: 20 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。



```
系統管理員: 命令提示字元
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 。
12. 輸入：npm install multer --save 安裝 Multer 。

★★★★ Lab02: 撰寫主程式與表單★★★★

1. 在命令提示字元中輸入：code . 開啟 VS Code 編輯器。
2. 在 index.html 中輸入以下程式碼並儲存

```
<p> UPLOAD Form</p>
<form action="/upload_file" method="post" enctype="multipart/form-data">
  <input type="file" name="myfile">
  <input type="submit" value="上傳檔案">
</form>
```

3. 記得要在 <form> 標籤中加入 enctype 屬性，並且把值改成 multipart/form-data
4. 在 app.js 中輸入以下程式碼並儲存

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

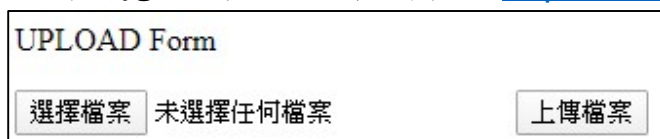
var upload = multer({ dest: 'upload/' }); // 設置檔案存放的路徑

app.post('/upload_file', upload.single('myfile'), function(req, res){
  res.send("上傳成功");
});

app.get("/", function(req, res) {
  res.sendFile(__dirname + '/index.html', function(err) {
    if (err) res.send(404);
  });
});

app.listen(3000);
```

5. 在專案資料夾，新建一個子資料夾，名稱：「upload」
6. 到「命令提示字元」輸入 node app.js 執行 js 。
7. 打開瀏覽器，在網址列上輸入「<http://127.0.0.1:3000>」打開網頁，可以看到表單。

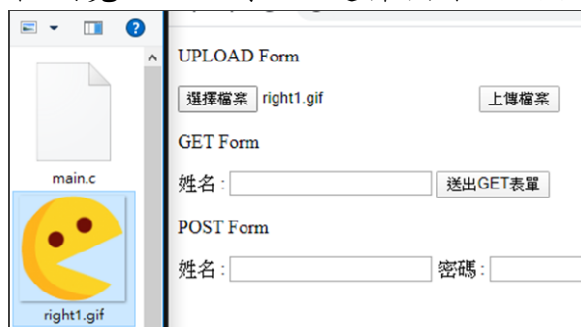


UPLOAD Form

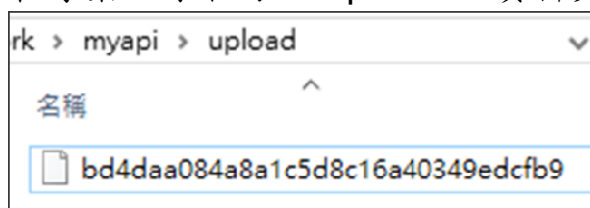
選擇檔案 未選擇任何檔案

上傳檔案

8. 在瀏覽器上點擊 “選擇檔案” 按鈕，選取一個文件後上傳。



9. 在專案目錄下的 “upload” 資料夾，裡面便會有我們剛剛上傳的檔案



★★★ Lab03: 過濾上傳檔案 ★★★

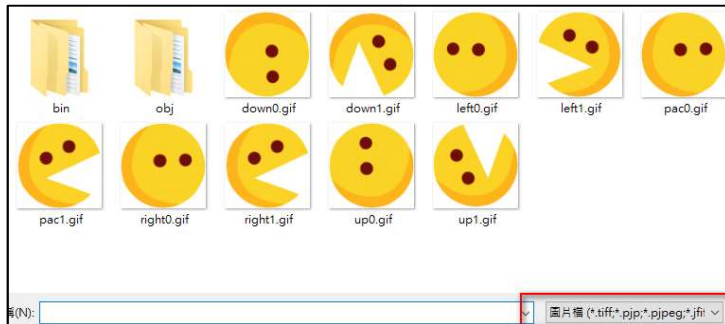
1. 過濾檔案類型的方法主要可以分為在前端 (HTML、JS) 上處理，或是後端伺服器收到文件後再處理。
2. 前端過濾的方式我們可以利用 HTML `<input>` 控件裡面的 `accept` 屬性。
3. 如果要限制只要讀取 PNG、GIF、BMP 類型的圖片只需加上

```
<input type="file" name="myfile" accept=".png,.gif,.bmp">
```

4. 如果想要全部的圖檔 MIME 類型。

```
<input type="file" name="myfile" accept="image/*">
```

5. 更多的 MIME Type 請看 <https://reurl.cc/mnd8AA>
6. 回到瀏覽器，在網址列上一樣先輸入剛剛的「<http://127.0.0.1:3000>」。
7. 點擊上傳按鈕，可以發現被限制只顯示圖片類型的文件了。



8. 前端過濾的方式因為是在 Client 端做判斷，可能因為請求參數被惡意修改而變得不安全 (過濾失敗)。
9. 我們可以利用 `multer` 的 `fileFilter` callback 去寫規則來自定義過濾方案。
10. 回到 `app.js` 上，編輯 `multer` 項目。

```
var upload = multer({
  storage: myStorage, // 設置 storage
  fileFilter: function (req, file, cb) { // 檔案過濾
    if (file.mimetype !== 'image/gif') { // 檢查 MIME 類型是否不為 image/gif
      return cb(new Error('Wrong file type'));
    }
    cb(null, true)
  }
});
```

11. 重新執行 `app.js`，試看看上傳的時候可不可以上傳非 GIF 類型的檔案。
12. 試看看修改上傳失敗時返回的畫面 (403 page)