



檔案上傳



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

- 16-1: 安裝 multer
- 16-2: 設定上傳資料夾
- 16-3: 過濾上傳的檔案



16-1

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

我們即將一起學習：

- 安裝 multer 套件
- 透過yarn安裝 multer
- multer 設定: diskStorage
- multer設定: 限定上傳的檔案大小
- 自定義儲存模式
- 上傳文件表單
- 後端處理上傳文件
- 在前端過濾檔案類型
- 在伺服端過濾檔案
- 顯示檔案資訊

【Key Points】：

- 16-1: 安裝 multer
- 16-2: 設定上傳資料夾
- 16-3: 過濾上傳的檔案

16-1: 安裝 multer

- 安裝 multer 套件
- 透過yarn安裝 multer
- multer 設定: diskStorage
- multer 設定: 限定上傳的檔案大小



designed by freepik



designed by freepik

16-2

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

- 以 npm 安裝 multer
- multer只支援content-type:multipart/form-data。
- form表單務必記得要填 enctype="multipart/form-data"。
- diskStorage()方法可以傳入一個物件參數，destination屬性可以指定存放的資料夾。
- diskStorage()方法可以傳入一個物件參數，filename屬性可以變更檔案名稱。
- 限定上傳檔案的大小，有助於資訊安全

【Key Points】：

以 npm 安裝 multer

multer只支援content-type:multipart/form-data

限定上傳檔案的大小，有助於資訊安全

安裝 multer 套件

- 輸入：`npm install multer --save`
- multer 跟前面提過的 body-parser 一樣都是屬於中介軟體，不過不同的是 multer 只處理類型為 multipart/form-data 的資料
- 發送 POST 表單預設是用 application/x-www-urlencoded 類型編碼送出，但如果遇到要發送二進制文件時，效率就會很差。
- 於是，需要將表單的 enctype 屬性設定成 multipart/form-data，而且，multer 套件也只處理這種編碼。

16-3



1. 輸入: `npm install multer --save`
2. multer 跟前面提過的 body-parser 一樣都是屬於中介軟體，不過不同的是 multer 只處理類型為 multipart/form-data 的資料
3. 剛剛我們發送 POST 表單是用 application/x-www-urlencoded 類型編碼送出
4. 發送二進制文件時如果用 UTF-8重新編碼的話，效率就會很差
5. 這時候就需要 multipart/form-data 的

HTML表單中的enctype屬性值有三種類型

- application/x-www-urlencoded
- multipart/form-data
- text/plain

rfc1867 在 Content-Type 的類型擴充了 multipart/form-data 用來支持向server發送二進制數據

【Key Points】：

安裝 multer 套件: `npm install multer --save`

multer 屬於中介軟體，只處理類型為 multipart/form-data 的資料

表單的enctype屬性要設定為"multipart/form-data"

透過yarn安裝 multer

1. 請先到 <https://classic.yarnpkg.com/zh-Hant/>
2. 請點選「安裝YARN」，下載並且安裝
3. 在終端機視窗輸入
yarn add multer
透過yarn安裝樣板引擎



16-4

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

1. 請先到 <https://classic.yarnpkg.com/zh-Hant/>
2. 請點選「安裝YARN」。
3. 網站會依據你的作業系統自動提示可以裝的版本。
4. 下載並且安裝
5. 在 Visual Studio Code 按下「Ctrl + 滾動」開啟 terminal 終端機視窗
6. 在專案根目錄底下，透過 yarn 安裝 multer，指令: `yarn add multer`

【Key Points】：

安裝YARN

在 Visual Studio Code 按下「Ctrl + 滾動」開啟 terminal 終端機視窗
輸入 `yarn add multer`，透過yarn安裝 multer

multer 設定: diskStorage

- 以 `multer.diskStorage({ ... })` 建構 storage
- `destination`屬性可以指定存放的資料夾
- `filename`屬性用來
變更檔案名稱

```
const multer = require('multer');
const storage = multer.diskStorage([
  destination: function (req, file, cb) {
    cb(null, './uploads')
  },
  filename: function (req, file, cb) {
    cb(null, file.originalname)
  }
])

const upload = multer({ storage: storage })

const app = express();

app.post('/profile', upload.single('uploadedFile'), function (req, res, next) {
  console.log('[req.file]', req.file)
  res.send('OK')
})
```

16-5



- 如果不是用 `diskStorage`，將無法變更儲存的檔名。
- `diskStorage()`方法可以傳入一個物件參數，其 `destination` 屬性可以指定存放的資料夾。
- 以本例來說，指定目前資料夾之下的「`uploads`」子資料夾。「`uploads`」需要系統管理者手動建立。
- `diskStorage()`方法可以傳入一個物件參數，其 `filename` 屬性可以變更檔案名稱。舉例來說，若改成「`Date.now() + '-' + file.originalname`」，是將檔案名稱保存成「時間戳記-原始檔名.`xxx`」格式。
- `originalName` 為檔案在用戶端的原始檔名。

【Key Points】：

`destination` 屬性可以指定存放的資料夾
`filename` 屬性可以變更檔案名稱放的資料夾
`originalName` 為檔案在用戶端的原始檔名

multer設定：限定上傳的檔案大小

- 呼叫 multer 時，可透過 limits，限定上傳的檔案大小與數量。
- 以本例來說，**fileSize** 設定為 **1 * 1024 * 1024**，表示單一檔案大小的上限是**1MB**。
- 壓低各項限定值，有助於降低被惡意上傳的風險
- 有助於降低網路阻斷攻擊(DoS)的成本

```
const maxSize = 1 * 1024 * 1024; // 1MB
var upload = multer({
  storage: storage,
  limits: { fileSize: maxSize }
})
```

16-6



- 針對單一上傳的檔案，設定大小的限制
- 也可以設定上傳一次上傳的數量
- 降低被惡意上傳的風險
- 抵禦網路阻斷攻擊(DoS)
- 詳組的設定名稱與預設值，請參考：<https://github.com/expressjs/multer#limits>

【Key Points】：

針對單一上傳的檔案，設定大小的限制
降低被惡意上傳的風險
抵禦網路阻斷攻擊(DoS)

16-2：設定上傳資料夾

- 自定義儲存模式
- 上傳文件表單
- 後端處理上傳文件



designed by freepik



designed by freepik

16-7

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

接下來，我們要一起學習：

- multer 提供了 storage 這個參數來讓我們對檔案儲存的路徑、檔名進行自定義
- storage 可以設置儲存路徑(destination)跟檔名(filename)
- 上傳文件表單請留意 表單 <form> 屬性 enctype 改成 multipart/form-data，以及文件選擇器的語法: <input type="file" name="myfile">
- 後端處理上傳文件
- 實際經歷整個上傳文件的流程

【Key Points】：

multer 可自訂檔案儲存的路徑、檔名

上傳文件表單請留意 表單 <form> 屬性 enctype 改成 multipart/form-data

文件選擇器的語法: <input type="file" name="myfile">

自定義儲存模式

- 預設的儲存模式不見得是我們想要的形式，multer 提供了 **storage** 參數來讓我們配置儲存設定。
- **storage** 可以設置儲存路徑(**destination**)跟檔名(**filename**)
- 下面範例，把檔案名稱保存成 時間戳-原始檔名.xxx 格式

```
//自定義 storage
var myStorage = multer.diskStorage({
  destination: function (req, file, cb) {
    cb(null, "upload"); // 保存的路徑 (需先自己創建)
  },
  filename: function (req, file, cb) {
    cb(null, Date.now() + '-' + file.originalname); // 自定義檔案名稱
  }
});

var upload = multer({storage: myStorage}); // 設置 storage
```

16-8



- 預設的儲存模式不見得是我們想要的形式，multer 提供了 **storage** 這個參數來讓我們對檔案儲存的路徑、檔名進行自定義。
- **storage** 可以設置儲存路徑(**destination**)跟檔名(**filename**)
 - **destination**：設定資源的儲存路徑。注意，如果沒有這個配置項，預設會儲存在 `/tmp/uploads` 下。此外，路徑需要自己建立。
 - **filename**：設定資源儲存在本地的檔名。
- 範例就是把檔案名稱保存成「時間戳記-原始檔名.xxx」格式
- 同學可以試試看其他的命名規則
- 如果我們想要回傳錯誤訊息可以用 `cb(null, false, {message: '具體錯誤'});` 的方式設置

【Key Points】：

multer 提供了 **storage** 這個參數來讓我們對檔案儲存的路徑、檔名進行自定義

storage 可以設置儲存路徑(**destination**)跟檔名(**filename**)

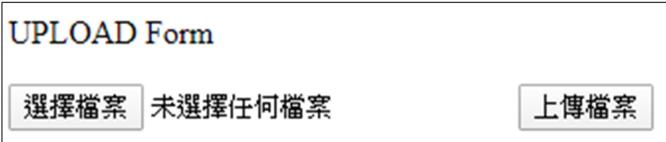
範例就是把檔案名稱保存成 時間戳記-原始檔名.xxx 得格式保存

上傳文件表單

- 回到 index.js 上，我們加入一個上傳文件的表單

```
<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>
```

- 在 <form> 標籤中加入 enctype 屬性，並且把值改成 **multipart/form-data**
- 刷新瀏覽器，可以看到表單已經加進去了



16-9



表單 <form> 屬性 enctype 主要有三個值可選，這個屬性管理的是表單的 MIME 編碼：

- application/x-www-form-urlencoded (預設值)
 - multipart/form-data
 - text/plain
-
- 因為我們要實作文件上傳，因此這邊要把 enctype 改成 multipart/form-data
 - 在 post 表單裡面新增一個文件選擇器
 - 文件選擇器的語法: <input type="file" name="myfile">
 - type 固定是"file"，name可以自己命名，以本例來說: "myfile"

【Key Points】：

表單 <form> 屬性 enctype 改成 multipart/form-data

文件選擇器的語法: <input type="file" name="myfile">

type 固定是"file"，name可以自己命名，以本例來說: "myfile"

後端處理上傳文件

- 接著在 app.js 中加入 multer 並且新增對應路由。

```
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);
```

16-10



- 如果還沒安裝 multer 請先安裝 (npm install multer –save)
- 接著在 app.js 中引用(require) multer 並且新增對應路由
- multer 用於處理檔案上傳到伺服器上，因此需先設置保存路徑
multer({ dest: 'upload/' })
- 接著在 post 路由中加進 upload.single('myfile')，這裡的 myfile 為剛剛我們HTML上面設的檔案選擇器名稱
- 因為只有上傳單一文件 所以用 single，假如要多文件的話要設定成 array

【Key Points】：

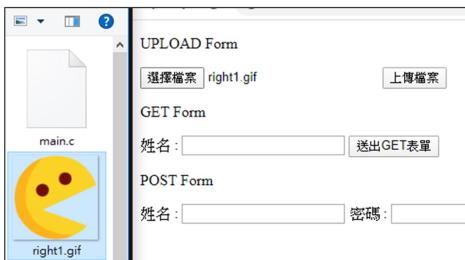
npm install multer

設置保存路徑 multer({ dest: 'upload/' })

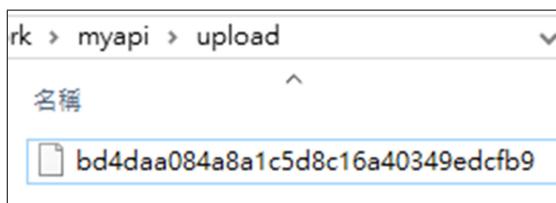
在 post 路由中加進 upload.single('myfile')，這裡的 myfile 為HTML設的檔案選擇器名稱

上傳文件

- 在瀏覽器上點擊“選擇檔案”按鈕，選取一個文件後上傳。



- 在專案目錄下，找出“upload”資料夾，裡面便會有我們剛剛上傳的檔案



16-11

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

- 當改寫完 app.js 後我們重新運行伺服器
- 之後在瀏覽器上刷新頁面
- 點擊“選擇檔案”按鈕，選取一個文件後上傳
- 回到專案目錄，找到“upload”資料夾，裡面便會有我們剛剛上傳的檔案
- 檔案名稱保存成「時間戳記-原始檔名.xxx」格式

【Key Points】：

重新運行伺服器，瀏覽器上刷新頁面

選取一個文件後上傳

專案目錄下的“upload”資料夾，內含我們剛剛上傳的檔案

16-3：過濾上傳的檔案

- 在前端過濾檔案類型
- 在前端過濾檔案類型（實例）
- 在伺服端過濾檔案
- 顯示檔案資訊



designed by freepik



designed by freepik

16-12

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

- 同學們應該會發現剛剛上傳的檔案名稱已經被 hash 過而且無法明確從檔名判斷原檔案類型
- 這是因為 multer 可以讓我們自定義檔案上傳路徑、名稱、型別。
- 而剛剛的範例只有修改上傳路徑，其他選項維持預設沒做任何更動，因此才會有這個問題。
- 接下來我們會教學如何定義 multer 得上傳參數。
- 還有如何透過判斷MIME類型來過濾文件

【Key Points】：

剛剛上傳的檔案名稱已經被 hash 過而且無法明確從檔名判斷原檔案類型
multer 可以讓我們自定義檔案上傳路徑、名稱、型別
如何透過判斷MIME類型來過濾文件

在前端過濾檔案類型

- 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。
- 前端過濾的方式我們可以利用 `<input>` 控件裡面的 `accept` 屬性
- 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">`
- 如果想要全部的圖檔MIME類型
`<input type="file" name="myfile" accept="image/*">`
- 更多的 MIME Type 請看 <https://reurl.cc/mnd8AA>

16-13



1. 過濾檔案類型的方法主要可以分為在前端或後端處理
2. 前端過濾的方式我們可以利用 `<input>` 控件裡面的 `accept` 屬性
3. 如要限制只要讀取 PNG、GIF、BMP 類型的圖片可以在 `input` 裡面加上
 - `accept=".png,.gif,.bmp"`
4. MIME Type 請看 <https://reurl.cc/mnd8AA>
5. 當然在前端過濾是比較有安全疑慮的 (因為過濾規則放在別人的瀏覽器上而非我們自己的伺服器)

【Key Points】：

前端過濾的方式我們可以利用 `<input>` 控件裡面的 `accept` 屬性

`accept=".png,.gif,.bmp"`

讓同學試試看前端過濾是否有方法可以繞過規則 (提示：用瀏覽器開發人員介面 (F12))

在前端過濾檔案類型（實例）

- **<form>** 表單的設定：

以 post 的方式，向同一伺服器的 /upload_file 路由傳送表單資料
表單資料的編碼方式為：multipart/form-data

- **name = "myfile"** 的控制項：

設定 accept = "image/*" 表示只接受上傳圖形類的檔案

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

16-14



關於表單與控制項的設定重點：

method="post"
action="/upload_file"
enctype="multipart/form-data"

以 post 的方式
向同一伺服器的 /upload_file 路由傳送表單資料
表單資料的編碼方式為：multipart/form-data

name = "myfile" 的 <input>
name = "myfile" 的 <input>

accept = "image/*"，表示只接受上傳圖形類的檔案
accept = "accept=".png,.gif"，
表示只接受上傳 *.png 或 *.gif 的檔案

【Key Points】：

表單的 enctype="multipart/form-data"
<input type="file"> 的控制項，以 accept 設定過濾條件
accept = "image/*"，表示只接受上傳圖形類的檔案

在伺服端過濾檔案

- 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)。
- 因此我們可以利用 multer 的 fileFilter callback 去寫規則來自定義過濾方案，這個方法屬於在後端 (Server) 上多做一次檢查，相對安全。

```
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)  
  }  
});
```

16-15



- 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)
- 後端上可以利用 multer 的 fileFilter callback 去寫規則來自定義過濾方案
- 我們在 multer 宣告時多加入一個 fileFilter 的引數，設置一個 callback function
- 在裡面我們去判斷當前檔案的 MIME 型態是否是我們想要的文件類型
- 當不符合類型時回傳錯誤訊息，符合就放行繼續保存文件

當檔案的 mimetype 不為指定類型時會返回一組錯誤訊息到瀏覽器上，這邊返回的類型可以自己決定，要返回 HTTP 狀態也可以，例如 403.3 伺服器禁止寫入，或是返回一個錯誤頁面...等

【Key Points】：

前端過濾的方式可能因為請求參數被惡意修改而變得不安全 (過濾失敗)
後端上可以利用 multer 的 fileFilter callback 去寫規則來自定義過濾方案
讓同學試看看上傳的時候可不可以上傳非 GIF 類型的檔案。

顯示檔案資訊

- multer 把檔案的資訊跟儲存位置放在 `req.file` 裡面，上傳後可以讀取

```
app.post('/upload_file', upload.single('myfile'), function(req, res){  
    var file = req.file;  
    console.log('檔案類型 : %s', file.mimetype);  
    console.log('原始檔名 : %s', file.originalname);  
    console.log('檔案大小 : %s', file.size);  
    console.log('檔案存放路徑 : %s', file.path);  
    res.send("上傳成功" + file.path);  
});
```

- 在瀏覽器上重新上傳檔案，可以看到終端機上印出相關的資訊

```
檔案類型 : image/gif  
原始檔名 : right0.gif  
檔案大小 : 40169  
檔案存放路徑 : upload\cad43519f6dbe7b078fc6cac99128e8f
```

16-16



- multer 把檔案的資訊跟儲存位置放在 `req.file` 裡面
- 我們可以把它從 `request.file` 中抓出來，其中有些屬性值得我們看看
- 在 `upload_file` 裡面用 `console.log` 把各個屬性印出來
- 瀏覽器上重新上傳檔案，可以看到終端機上印出相關的資訊
- 觀察幾項重要資訊: `file.mimetype`, `file.originalname`, `file.size`, `file.path`

【Key Points】：

multer 把檔案的資訊跟儲存位置放在 `req.file` 裡面

幾項重要資訊: `file.mimetype`, `file.originalname`, `file.size`, `file.path`

讓同學們觀察不同類型的文件它們的 `mimetype` 是否有什麼不同

Summary < 精華回顧 >

- HTML 的 `<form>` 控件可以用 `action` 屬性控制目的路徑；`method` 控制請求方法。
- 送出表單的按鈕 `type` 需設為 `submit` 類型。
- GET 表單會把內容當成 `Query String` 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。
- POST 表單預設是用 `application/x-www-urlencoded` 格式
- 當表單需要發送二進制文件時需要使用 `multipart/form-data` 格式。
- multer 提供 `storage` 參數，設置儲存路徑(`destination`)跟檔名(`filename`)。
- 檔案過濾建議用 multer 的 `fileFilter callback` 去定義規則，會比前端判斷安全。



16-17

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

- 回顧一下上面我們教過的一些重點
1. HTML 的 `<form>` 控件可以用 `action` 屬性控制目的路徑；`method` 控制請求方法
 2. 送出表單的按鈕 `type` 需設為 `submit` 類型
 3. GET 表單會把內容當成 `Query String` 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
 4. POST 表單預設是用 `application/x-www-urlencoded` 格式編碼資料
 5. 當表單需要發送二進制文件時需要使用 `multipart/form-data` 格式
 - 二進制文件時如果用 UTF-8 重新編碼的話，效率就會很差
 6. multer 提供 `storage` 參數可以設置儲存路徑(`destination`)跟檔名(`filename`)
 - `destination`：設定資源的儲存路徑。注意，如果沒有這個配置項，預設會儲存在 `/tmp/uploads` 下。此外，路徑需要自己建立。
 - `filename`：設定資源儲存在本地的檔名。
 7. 檔案過濾建議用 multer 的 `fileFilter callback` 去定義規則，會比前端判斷安全

【Key Points】：

當表單需要發送二進制文件時需要使用 `multipart/form-data` 格式
multer 提供 `storage` 參數可以設置儲存路徑(`destination`)跟檔名(`filename`)
檔案過濾建議用 multer 的 `fileFilter callback` 去定義規則，會比前端判斷安全

線上程式題

- 16-1 如何限定只能上傳 PDF 檔

```
<input type="file" name="uploadedFile" multiple/>
```

- 請修改上述 HTML，使其只允許上傳pdf檔。

- 16-2 將上傳的檔案置於指定的位置

請修改程式碼，使其能夠將檔案存放在專案根目錄files資料夾底下，並且，檔名修改成「原始檔名-{當下ISO時間}」。

- 16-3 限定上傳的檔案數量

如何限定上傳的檔案數量？

16-18



題目名稱: 16-1 如何限定只能上傳 PDF 檔

內容說明:

```
<input type="file" name="uploadedFile" multiple/>
```

請修改上述 HTML，使其只允許上傳pdf檔。

題目名稱: 16-2 將上傳的檔案置於指定的位置

內容說明:

請修改程式碼，使其能夠將檔案存放在專案根目錄files資料夾底下，並且，檔名修改成「原始檔名-{當下ISO時間}」。

題目名稱: 16-3 限定上傳的檔案數量

內容說明:

如何限定上傳的檔案數量？

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】：

如何限定只能上傳 PDF 檔

將上傳的檔案置於指定的位置

限定上傳的檔案數量

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，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。

- **Lab01: 安裝/確認Node.js執行環境**

- **Lab02: 撰寫主程式與表單**

- **Lab03: 過濾上傳檔案**

Estimated time:
20 minutes

16-19



【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。

Lab01: 安裝/確認Node.js執行環境

Lab02: 撰寫主程式與上傳表單

Lab03: 過濾上傳檔案

完成後的程式與檔案，請參考 Example 資料夾的內容。

【Key Points】：

Lab01: 安裝/確認Node.js執行環境

Lab02: 撰寫主程式與上傳表單

Lab03: 過濾上傳檔案

範例程式使用說明

- 範例程式資料夾: Module_16_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 body-parser multer`
 6. 在終端機視窗，輸入 `node app.js`
 7. 啟動瀏覽器，連接 `http://localhost:3000` (測試時，僅限上傳 gif 圖檔)

16-20



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 body-parser multer`
6. 在終端機視窗，輸入 `node app.js`
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 + 滑鼠右鍵」開啟 terminal 終端機視窗，輸入：「node 主程式.js」