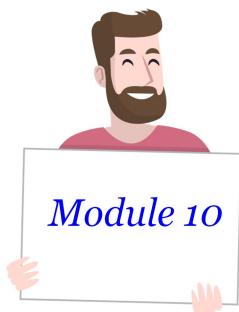




設定routes



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

- 10-1: 網站根目錄
- 10-2: 注意路由順序
- 10-3: 自訂 404 頁面



10-1

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

我們即將一起學習：

- 絕對路徑與相對路徑
- 程式讀取路徑
- public、routes、views 資料夾
- path.resolve、path.join
- Middleware 中介軟體
- Router 路由程式設計
- Route 路由順序
- Router 層之進階寫法
- HTTP 回應狀態碼
- 以 res.status() 設定狀態碼
- 以 res.render() 動態產生網頁到用戶端
- 以 res.json() 輸出 JSON 格式的結果

【Key Points】：

- 10-1: 網站根目錄
- 10-2: 注意路由順序
- 10-3: 自訂 404 頁面

10-1：網站根目錄

- 絶對路徑與相對路徑
- 程式讀取路徑
- **public**、**routes**、**views**資料夾
- **path.resolve**、**path.join**



designed by freepik



designed by freepik

10-2

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

- 絶對路徑表示會以主機根目錄為起點開始，顯示每層目錄直到該執行檔案。
- 相對路徑指的是以該執行檔案為起點，標記其相對位置。
- 在Node.js專案中常使用相對路徑引用模組。
- `__dirname`、`dir`是`directory`的意思，會回傳如`C:\Users\{username}\OneDrive\文件\Module_10_`設定`routes\lab_01`
- `__filename`，會回傳如`C:\Users\{username}\OneDrive\文件\Module_10_`設定`routes\lab_01\show_filename.js`

【Key Points】：

絕對路徑表示會以主機根目錄為起點開始，顯示每層目錄直到該執行檔案

相對路徑指的是以該執行檔案為起點，取相對位置

`__dirname`、`dir`是`directory`的意思

絕對路徑與相對路徑

- 絶對路徑:以根目錄為起點，顯示每層目錄直到檔案
例如:

/c/Users/{username}/OneDrive/文件/Module_10_設定routes

- 相對路徑:以該檔案為起點，標記出相對位置
例如: ../文件

- ./ 符號 表示同層目錄
- ../ 符號 表示上層目錄
- ~ 符號 在Linux 系統，代表用戶根目錄

10-3



1. 絶對路徑表示會以根目錄為起點開始，顯示每層目錄直到該檔案。
2. 相對路徑指的是以該檔案為起點，標記出相對位置。
3. 相對路徑以 ./ 表示同層目錄。
4. 相對路徑以 ../ 表示上層目錄。
5. 在Linux 系統中以~符號代表用戶根目錄。

【Key Points】：

絕對路徑:以根目錄為起點，顯示每層目錄直到檔案

相對路徑:以該檔案為起點，標記出相對位置

相對路徑以./ 表示同層目錄

程式讀取路徑

- **__dirname:**
執行檔案的目錄絕對路徑
- **__filename:**
執行檔案的檔案絕對路徑
- **process.cwd:**
command line執行檔案的目錄絕對路徑
- 引用檔案請盡量使用相對路徑，避免因主機資料夾結構不同，發生搬遷程式碼後導致引用路徑錯誤。

10-4



- **__dirname:**回傳執行檔案的目錄絕對路徑。dir是directory的意思。
會回傳如C:\Users\{username}\OneDrive\文件\Module_10_設定routes\lab_01
- **__filename:**回傳執行檔案的檔案絕對路徑。
__filename，會回傳如 C:\Users\{username}\OneDrive\文件\Module_10_設定routes\lab_01\show_filename.js
- **process.cwd**，會回傳如 C:\Users\{username}\OneDrive\文件\Module_10_設定routes\lab_01
- 引用檔案請盡量使用相對路徑，避免因主機資料夾結構不同，發生搬遷程式碼後導致引用路徑錯誤。
- 在Node.js專案中常使用相對路徑引用模組。

【Key Points】：

__dirname:回傳執行檔案的目錄絕對路徑
__filename:回傳執行檔案的檔案絕對路徑
引用檔案請盡量使用相對路徑

public、routes、views資料夾

- 一般express專案主要會有這三個資料夾。
- **Public** 主要放置靜態檔案如css、images以及js
- **Routes** 主要放置網站router function
- **Views** 主要放置要給前端的視圖(Views)
- `app.set("views", path.join(__dirname, 'views'))`
指定views資料夾為視圖(view)所在地
- `app.set("view engine", "ejs");`
指定 ejs 為樣版引擎

10-5



- 使用 express-generator 會直接看到套件產生這三個資料夾: public、routes、views。
- `express.static()` 可以將 public 設定為放置靜態檔案的資料夾，存放並提供給瀏覽器例如 css、images、js 等類型檔案。
- 在 routes 資料夾，可以撰寫一些路由分派邏輯。
- `app.set("views", path.join(__dirname, 'views'))` 可以指定views資料夾為視圖(view)所在地，未來 `res.render()` 時，就將views資料夾的視圖渲染(動態輸出)給瀏覽器。
- `app.set("view engine", "ejs");` 的作用是: 指定 ejs 為樣版引擎。常見的view 引擎有ejs、pug 等等。

【Key Points】：

Public 主要放置靜態檔案如css、images以及js

Routes 主要放置網站router function

Views 主要放置要給前端的視圖(Views)

path.resolve、path.join

- **path.join()**，會把函式裡所有path參數字串串連起來並回傳
- **path.resolve()**
函式的路徑解析會由第一個參數開始解析最後再回傳，可以理解成
函式中的所有參數，都依序執行一次 cd 指令，最後，再回傳 pwd
顯示的絕對路徑

The screenshot shows a terminal window with the following details:

- File tree on the left:
 - Welcome
 - X JS app.js
 - ✓ TMP_03
 - ✓ a \ ab
 - ✓ bc01
 - ✓ bc02 \ cd01
 - JS app.js
- Code editor on the right:

```
1 const path=require('path');
2 console.log(path.join('/a','/b'));
3 console.log(path.resolve('a/ab','bc01','..../bc02','cd01'));
```

10-6

- **path.join()**，會把函式裡所有path參數字串串連起來並回傳，例如 `path.join('/a', '/b') // 回傳 /a/b`
- **path.resolve()**，函式的路徑解析會由第一個參數開始解析最後再回傳，可以理解成：函式中的所有參數，都依序執行一次 cd 指令，最後，再回傳 pwd 顯示的絕對路徑。
- 例如：`path.resolve('a/ab', 'bc01', '..../bc02','cd01')`。回傳 `a\ab\bc02\cd01`，可以看成：
cd /a/ab
cd /bc01
cd ../bc02
cd /cd01
pwd
顯示回傳

C:\Users\{username}\OneDrive\文件\Module_10_設定routes\tmp_03\a\ab\bc02\cd01

- 在express專案裡，常使用**path.join()**來綁定網站目錄裡的資料夾路徑相對關係。
- **path.resolve()**要注意每一層資料夾的相對位置。

【Key Points】：

path.join()
path.resolve()
path.resolve()要注意每一層資料夾的相對位置

10-2: 注意路由順序

- **Middleware** 中介軟體
- **Router** 路由程式設計
- **Route** 路由順序
- **Router** 層之進階寫法



designed by freepik



designed by freepik

10-7

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

- `app.use()` 意味著express會將該方法所運行的函式，當作一個application級別的middleware，所有請求都會經過它。
- `next()` 非常重要，如果沒有在middleware function裡給定`next()`，程式就不會執行下一個middleware function。
- `route.use()` 屬於Route級別的middleware，意即所有訪問該Route的請求都會經過該middleware function。
- router的callback function都是`req`、`res`、`next`這三個參數，分別代表Request物件、Response物件、`next()`。
- MVC是一套成熟的開發架構，分層管理，Model負責資料層、Controller負責程式流程、邏輯，View負責資料呈現。

【Key Points】：

`app.use()` 意味著express會將該方法當成middleware，所有請求都會經過它
`router`的callback function都是`req`、`res`、`next`這三個參數
`next()` 非常重要，如果沒有呼叫`next()`，程式就不會執行下一個middleware function

Middleware 中介軟體

- **use()**
- **next()**

```
const logInMiddleware = (req,res,next) =>{
  console.log(`log:[${req.method}]${req.url} -- ${new Date().toISOString()}`)
  //log:[GET]/ -- 2020-03-09T12:19:47.552Z
  next();
}
app.use(logInMiddleware)
```

- **next(error)**
- **Application級別的middleware**
- **Route級別的middleware**

```
const exceptionShowUp=(req,res,next)=>{
  setTimeout(()=>{
    next(new Error('exception'))
  },3000)
}
app.use(exceptionShowUp)
app.use((err,req,res,next)=>{
  console.error(err.message)//exception
  next()
})
```

10-8



- `app.use()`意味著express會將該方法所運行的函式，當作一個application級別的middleware，所有請求都會經過它。
- `next()`非常重要，如果沒有在middleware function裡給定`next()`，程式就不會執行下一個middleware function。
- callback function裏頭的error無法直接被外層的try catch捕捉，但可以透過`next(error)`傳遞到下一個middleware function做處理。
- `app.use()`屬於Application 級別的middleware，意即所有請求都會經過該middleware function。
- `route.use()`屬於Route級別的middleware，意即所有訪問該Route的請求都會經過該middleware function。

【Key Points】：

`app.use()`意味著express會將該方法當成middleware，所有請求都會經過它
`route.use()`屬於Route級別的middleware。

`next()`非常重要，如果沒有呼叫`next()`，程式就不會執行下一個middleware function

Router路由程式設計

- **router.get()**
- **router.get("/")**
- **router.get("/", callback)**
- **router的callback函式都是req、res、next這三個參數，分別代表：**
 - Request物件
 - Response物件
 - next()

```
const express = require('express');
const router = express.Router();

/* GET home page. */
router.get('/', function(req, res, next) {
  res.render('index', { title: 'Express' });
});

module.exports = router;
```

```
app.use('/', indexRouter);
app.use('/users', usersRouter);
```

10-9



1. router.get()，意即該Route的HTTP method為GET，主要的HTTP method 有GET、PUT、PATCH、DELETE。
2. router.get("/", callback)，代表定義了一個root route，可以自定義route。
3. router的callback function 都是req、res、next這三個參數，分別代表Request物件、Response物件、next()。
4. router function可以不用寫next()，因為處理完請求就可以直接回傳，程式不需要往下一個middleware function前進。
5. app.use("/users", usersRouter)，代表所有 /users 請求都會進入到 usersRouter 。

【Key Points】：

router.get("/", callback)

router的callback function 都是req、res、next這三個參數

app.use("/users", usersRouter)，代表所有 /users 請求都會進入到 usersRouter

Route 路由順序

- `app.get('hello-world')` 與 `router.get('/hello')` 都可以重複宣告，但要記得呼叫`next()`，程式執行才會進入到下一個
- 程式執行到同樣的path會依序由上而下執行
- `app.use()`會依序加載

```
app.get('/hello-world',(req,res,next)=>{
  console.log('hello-world first');
  next();
});
app.get('/hello-world',(req,res,next)=>{
  console.log('hello-world second');
  next();
});
```

```
router.get('/hello', function(req, res, next) {
  console.log('hello first');
  next();
});
router.get('/hello', function(req, res, next) {
  console.log('hello second');
  next();
});
```

10-10



- `app.get('hello-world')` 與 `router.get('/hello')` 可以重複宣告，但要記得呼叫`next()`，程式執行才會進入到下一個function。
- 程式執行到同樣的path會依序由上而下執行。
- `app.get()`與`router.get()`差別在於: `router.get()`還需要透過`app.use()`呼叫，才會執行到該`router.get()`。
- `app.use()`會依序加載。
- 實務上建議route path 獨立不重複，如果route path會重複，可透過HTTP method做區隔，請參考Restful API設計。例如:
`app.get("/member/login", ...)` → 登入表單
`app.post("/member/login", ...)` → 接收資料，進行登入檢查

【Key Points】：

程式執行到同樣的path會依序由上而下執行。

`app.use()`會依序加載 function

可以重複宣告，但要記得呼叫`next()`，程式執行才會進入到下一個function

Router層之進階寫法

- **controller function** 只是將`router.get()`的**callback function**拆分出來
- 可讀性較高
- **router層定義好route path**
- **controller層負責程式流程**

```
const express = require('express');
const router = express.Router();

const firstController=(req,res,next)=>{
    res.send('hello first');
}
const secondController=(req,res,next)=>{
    res.send('hello second');
}

router.get('/hello-first',firstController);
router.get('/hello-second',secondController);

module.exports = router;
```

10-11



- controller function只是將`router.get()`的**callback function**拆分出來。
- 直接在**router function**裡寫程式邏輯會使程式可讀性降低。
- **router層定義好route path**即可，**controller層負責程式流程、邏輯**。
- 可以將**controller**獨立出去放到統一的**Controller**資料夾，分資料夾管理。
- **MVC**是一套成熟的開發架構，分層管理，**Model**負責資料層、**Controller**負責程式流程、邏輯，**View**負責資料呈現。

【Key Points】：

controller function只是將`router.get()`的**callback function**拆分出來
router層定義好route path即可，**controller層負責程式流程、邏輯**
MVC = **Model**負責資料層、**Controller**負責程式流程、邏輯，**View**負責資料呈現

10-3：自訂 404 頁面

- HTTP 回應狀態碼
- 以 `res.status()` 設定狀態碼
- 以 `res.render()` 動態產生網頁到用戶端
- 以 `res.json()` 輸出 JSON 格式的結果



designed by freepik



designed by freepik

10-12

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

- 404，請求網頁不存在或資源不存在。
- `res.status()` 可以傳一個數字型別或數字的字串型別參數，作為HTTP status code回傳給用戶端。
- `res.send()` 回傳內容就能設定為404 Not Found。
- `res.render()`可以傳出一個ejs檔並渲染到用戶端
- `res.json()` 可以傳遞一個物件。

【Key Points】：

404，請求網頁不存在或資源不存在

`res.status()` 可以傳出 HTTP status code 紿用戶端

`res.render()` 會將一個ejs檔渲染(動態輸出)到用戶端

HTTP 回應狀態碼

- **2xx** 表示運作正常
- **4xx** 這類的錯誤，
用戶端的責任佔比多一些
- **5xx** 通常是伺服端出了狀況



10-13

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

- 200，請求正常被伺服器處理並回傳。
- 301，請求跳轉至其他url path。
- 401，用戶端的使用者身分無法被伺服器驗證通過，予以拒絕請求。
- 403，用戶端的權限不足以取得伺服器端資源，予以拒絕請求。
- 404，請求網頁不存在或資源不存在。
- 500，伺服器端發生錯誤

【Key Points】：

2xx 表示正常

4xx 這類的錯誤，用戶端的責任佔多一些

5xx 通常是伺服端出了狀況

以 res.status() 設定狀態碼

- **res.status()**
傳出 HTTP status code 給用戶端
- 也可以使用 **res.statusCode**
- 預設回傳200 HTTP status code

```
const express = require('express');
const router = express.Router();

router.get('/response-200-status',(req,res,next)=>{
  res.status(200);
  res.send('OK');
});

router.get('/response-401-status',(req,res,next)=>{
  res.statusCode=401;
  res.send('Authentication fails');
  next();
});

router.get('/response-500-status',(req,res,next)=>{
  try {
    throw new Error('exception');
  } catch (e) {
    console.error(e.message);
    res.status('500');
    res.send('Server error');
  }
});
```

10-14



- **res.status()**可以傳一個數字型別或數字的字串型別參數，作為HTTP status code回傳給用戶端。
- 另外，也可以使用**res.statusCode**，給定一個數字於**res**物件的**statusCode**屬性，作為回傳的HTTP status code。
- 之所以有**res.status()**與**res.statusCode**這兩種寫法，是因為**express**新舊版本寫法不同，但新版仍向前支援。
- 500 status code常被用作表達伺服器發生錯誤，可以選擇寫在**catch function**或者**error handler function**。
- **res.status()**不需要在每個**router function**都呼叫使用，沒使用預設是回傳200 HTTP status code。

【Key Points】：

res.status() 傳出 HTTP status code 給用戶端
也可以使用 res.statusCode
預設是回傳200 HTTP status code

以res.render()動態產生網頁到用戶端

- `res.status(404);`
- `res.render("404page")`
將一個 ejs 檔
渲染 (動態輸出) 到用戶端

```
app.use('/', indexRouter);
app.use('/users', usersRouter);

app.use(function(req, res, next) {
  res.status(404);
  res.render('404page');
  next();
});
```



10-15

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

- 請求進入到 express server後，如果請求都無法匹配到 route path，就會進入到下一個 middleware function 。
- `res.status(404); res.send("404 Not Found")`回傳內容就能設定為 404 Not Found 。
- 也能透過 `res.render()` 回傳一個網頁 。
- `res.render()` 會將一個ejs檔渲染(動態輸出)到用戶端
- 一般網站，404回傳多半是回傳網頁而且只是一行字。網頁才是較好的呈現，`res.send()` 較少使用 。

【Key Points】：

`res.status(404); res.send("404 Not Found")`
`res.render()` 會將一個ejs檔渲染(動態輸出)到用戶端
一般網站，404回傳多半是回傳網頁

以 res.json() 輸出 JSON 格式的結果

- **res.json()** 可以傳遞物件或字串
- **res.json()** 預設**header**的 **content-type** 屬性值為 **application/json**

```
app.use('/', indexRouter);
app.use('/users', usersRouter);

app.use(function(req, res, next) {
  res.status(404);
  res.json('Page Not Found');
  next();
});
```

```
app.use('/', indexRouter);
app.use('/users', usersRouter);

app.use(function(req, res, next) {
  res.status(404);
  res.json({error: 'Page Not Found'});
  next();
});
```

```
app.use('/', indexRouter);
app.use('/users', usersRouter);

app.use(function(req, res, next) {
  res.status(404);
  res.setHeader('content-type','application/json')
  res.send({error: 'Page Not Found'});
  next();
});
```

10-16



- JSON格式已經是網站前後端溝通很常見的格式。
- **res.send()** 也可以回傳JSON字串，只需要設定回傳的 **header** 的 **content-type** 屬性值為 **application/json**。
- **res.json()** 可以傳遞字串當參數。
- **res.json()** 也可以傳遞一個物件。
- **res.json()** 預設**header**的 **content-type** 屬性值為**application/json**。

【Key Points】：

res.send() 回傳JSON字串時，要一併設定 **content-type** 屬性值為 **application/json**

res.json() 可以傳遞物件或字串

res.json() 預設**header**的 **content-type** 屬性值為**application/json**

精彩回顧

- 絕對路徑與相對路徑
- `public`、`routes`、`views`資料夾
- `path.resolve`、`path.join`
- **Middleware** 中介軟體
- **Router** 路由程式設計
- **Route** 路由順序與進階寫法
- 以 `res.status()` 設定狀態碼
- 以 `res.render()` 動態產生網頁到用戶端



10-17

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

- 絶對路徑表示會以主機根目錄為起點開始，顯示每層目錄直到該執行檔案。
- 相對路徑指的是以該執行檔案為起點，顯示其相對位置。
- `app.use()` 意味著 `express` 會將該方法所運行的函式，當作一個 `application` 級別的 `middleware`，所有請求都會經過它。
- `router.get()`，意即該 `Route` 的 `HTTP method` 為 `GET`，主要的 `HTTP method` 有 `GET`、`PUT`、`PATCH`、`DELETE`。
- `res.render()` 可以傳遞一個 `ejs` 檔名，作為 `view page`。

【Key Points】：

絕對路徑與相對路徑

Router 路由程式設計

以 `res.status()` 設定狀態碼、以 `res.render()` 動態產生網頁到用戶端

線上程式題

- **10-1 設定根路由接受GET方法**

請修改以下程式碼，調整HTTP method為GET。

```
router.delete('/', function(req, res, next) {  
    res.render('index', { title: 'Express' });  
});
```

- **10-2 指定路由處理結果的MIME型態**

請修改 GET /name router，繼續使用res.send()，但使其回傳JSON

- **10-3 指定 HTTP status**

修改 GET /name router，res.send() 換成 res.json()，並且改變回傳的HTTP status為204

10-18



10-1 設定根路由接受GET方法

請修改以下程式碼，調整HTTP method為GET。

```
router.delete('/', function(req, res, next) {  
    res.render('index', { title: 'Express' });  
});
```

10-2 指定路由處理結果的MIME型態

請修改以下GET /name router，繼續使用res.send()，但使其回傳JSON {"name":"Michael"}。

```
router.get('/name', function(req, res, next) {  
    res.send('name:Michael')  
});
```

10-3 指定 HTTP status

修改以下GET /name router，res.send() 換成 res.json()，並且改變回傳的HTTP status為204

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】：

10-1 設定根路由接受GET方法

10-2 指定路由處理結果的MIME型態

10-3 指定 HTTP status

課後練習題(Lab)

- **情節描述:**
透過使用Express.js建構一個網站，具有簡單的頁面，可以在瀏覽器上瀏覽。
具體要做到 [GET] <http://localhost:3000/firstday/lab>，
網頁回傳Welcome to Lab，如[GET]任意url即回傳404 NOT FOUND。
- **預設目標:**
 - 理解相對路徑
 - 理解Route設定
 - 理解404請求處理
- **Lab01:理解相對路徑**
- **Lab02:理解Route設定**
- **Lab03:理解404請求處理**

Estimated time:
20 minutes

10-19



【情節描述】

透過使用Express.js建構一個網站，具有簡單的頁面，可以在瀏覽器上瀏覽。具體要做到 [GET] <http://localhost:3000/firstday/lab>，網頁回傳Welcome to Lab，如[GET]任意url即回傳404 NOT FOUND。

【預設目標】

- 理解相對路徑
- 理解Route設定
- 理解404請求處理

Lab01:理解相對路徑

Lab02:理解Route設定

Lab03:理解404請求處理

完成後的程式與檔案，請參考 Example 資料夾的內容。

【Key Points】：

Lab01:理解相對路徑

Lab02:理解Route設定

Lab03:理解404請求處理

範例程式使用說明

- 範例程式資料夾: Module_10_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
 6. 在終端機視窗，輸入 npm start
 7. 啟動瀏覽器，連接 http://localhost:3000

10-20



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
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 + 滑鼠右鍵」開啟 terminal 終端機視窗，輸入：「node 主程式.js」