



使用 JSON 資料檔



designed by freepik

Estimated time:

50 min.

資訊工業策進會 Institute for Information Industry

【Key Points】：

學習目標

- 13-1: 引入 JSON 檔
- 13-2: 以表格呈現內容
- 13-3: 使用 EJS include()



13-1

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

我們即將一起學習：

- JSON 檔案與 JS 檔案的差異
- 使用 require() 引入 JSON 檔
- 使用 fs.readFile() 讀取 JSON 檔
- 使用 fs.readFileSync() 讀取 JSON 檔
- HTML表格的語法
- 以 res.render()渲染視圖(view)
- 將資料代入表格的標籤之中
- 由程式構成表格內容，整批代入視圖
- 主板頁面觀念
- 標題及導覽列
- 頁面內容
- 網頁頁尾

【Key Points】：

- 13-1: 引入 JSON 檔
- 13-2: 以表格呈現內容
- 13-3: 使用 EJS include()

13-1：引入 JSON 檔

- JSON 檔案與 JS 檔案的差異
- 使用 `require()` 引入 JSON 檔
- 使用 `fs.readFile()` 讀取 JSON 檔
- 使用 `fs.readFileSync()` 讀取 JSON 檔



designed by freepik



designed by freepik

13-2

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

1. JSON 檔案不能寫註解，JS 檔案可以寫註解。
2. JSON 檔案不能有方法，JS 檔案裡的模組可以有方法。
3. Require不只可以引用模組也可以引用JSON 檔案。
4. 引用JSON模組後，宣告的引用變數值就會是一個JSON物件。
5. 引用JSON模組後，會被npm cache，所以更動JSON內容將不會改變其引用值。

【Key Points】：

使用 `require()` 引入 JSON 檔
使用 `fs.readFile()` 讀取 JSON 檔
使用 `fs.readFileSync()` 讀取 JSON 檔

JSON 檔案與 JS 檔案的差異

- JSON 檔案不能寫註解，JS 檔案可以寫註解。
- JSON 檔案不能有方法，JS 檔案裡的模組可以有方法。
- JSON 檔案內容格式須符合 JSON 格式，
JS 檔案只需要是物件格式即可。
- JSON 檔案不需要 export 接口，JS 檔案需要有export 接口。
- JSON 檔案比較像檔案，JS 檔案比較像模組。

13-3



JSON 檔案與 JS 檔案有幾個差異：

- JSON 檔案不能寫註解，JS 檔案可以寫註解。
- JSON 檔案不能有方法，JS 檔案裡的模組可以有方法。
- JSON 檔案內容格式須符合 JSON 格式，JS 檔案只需要是物件格式即可。
- JSON 檔案不需要 export 接口，JS 檔案需要有export 接口。
- JSON 檔案比較像檔案，JS 檔案比較像程式模組。

【Key Points】：

JSON 檔案不能有方法，JS 檔案裡的模組可以有方法

JSON 檔案比較像檔案，JS 檔案比較像程式模組

JSON 檔案不需要 export 接口，JS 檔案需要有export 接口

使用 require() 引入 JSON 檔

- **Require**不只可以引用模組
也可以引用**JSON** 檔案
- **Require**引用**JSON**的語法跟
引用模組一樣
- 指定檔名時，連同副檔名也
要註記清楚：

```
let m = require("foo.json");
```

(JS 檔案可以忽略副檔名)

The screenshot shows a code editor with a file tree on the left and a code editor on the right. The file tree includes files like app.js, students.json, and package-lock.json. The code editor displays a JSON object:

```
[{"total":2, "students":[{"name":"Michael"}, {"name":"Mark"}]}]
```

Below the code editor, the text "JSON Object file" is displayed.

```
const students=require('./students.json');  
console.log(students);  
//{ total: 2, students: [ { name: 'Michael' }, { name: 'Mark' } ] }
```

13-4



- **Require**是常見的引用模組方式。
- **Require**不只可以引用模組也可以引用**JSON** 檔案。
- **Require**引用**JSON**的語法跟引用模組一樣，但在指定檔名時，連同副檔名要註記清楚。
- 引用**JSON**模組後，宣告的引用變數值就會是一個**JSON**物件。
- 引用**JSON**模組後，會被 **npm cache**，所以更動**JSON**內容將不會改變其引用值。

【Key Points】：

Require不只可以引用模組也可以引用**JSON** 檔案

Require引用**JSON**的語法跟引用模組一樣

指定檔名時，連同副檔名也要註記清楚

使用 `fs.readFile()` 讀取 JSON 檔

- `fs.readFile()` 的 path 請注意需填入完整副檔名
- 注意字元編碼，請正確選擇檔案內容的編碼方式，例如：`utf-8`。
- **fs.readFile()** 讀到的內容都是字串
- 需要透過 `JSON.parse()` 將字串轉換成物件

```
const fs=require('fs');
fs.readFile('../students.json',{encoding:'utf-8'},(e,data)=>{
  if(e){
    console.error(e)
  }
  console.log(typeof data) // string
  const students=JSON.parse(data)
  console.log(students)
  //{ total: 2, students: [ { name: 'Michael' }, { name: 'Mark' } ] }
});
```

13-5



- 可以使用 `fs` 讀取 JSON 檔案。
- `fs.readFile()` 的 path 請注意需填入完整副檔名。
- 注意字元編碼，請正確選擇檔案內容的編碼方式，例如：`utf-8`。
- `fs.readFile()` 讀取到的內容都是字串，也就是讀取到的是 JSON 格式的字串。
- 因此，需要透過 `JSON.parse()` 將字串轉換成 JSON 物件。

【Key Points】：

`fs.readFile()` 的 path 請注意需填入完整副檔名

`fs.readFile()` 讀取到的內容都是字串

透過 `JSON.parse()` 將字串轉換成 JSON 物件

使用 `fs.readFileSync()` 讀取 JSON 檔

- `fs.readFileSync()` 採用同步的方式讀取檔案內容。
- `fs.readFileSync()` 的 `path` 請注意需填入完整副檔名
- 注意字元編碼，請正確選擇檔案內容的編碼方式，例如：`utf-8`。
- `fs.readFileSync()` 讀到的內容都是字串
- 需要透過 `JSON.parse()` 將字串轉換成物件

```
try {
  const fs=require('fs');
  const data= fs.readFileSync('./students.json',{encoding:'utf-8'});
  console.log(typeof data) // string
  const students=JSON.parse(data);
  console.log(students);
  //{ total: 2, students: [ { name: 'Michael' }, { name: 'Mark' } ] }
} catch (e) {
  console.error(e);
};
```

13-6



- 可以使用`fs`讀取 JSON 檔案。
- `fs.readFileSync()` 採用同步的方式讀取檔案內容，資料讀完後，傳出資料內容，才會執行 `fs.readFileSync()` 的下一行程式。
- `fs.readFileSync()` 的`path`要注意填入完整副檔名。
- 注意編碼請選擇檔案的編碼方式。
- `fs.readFileSync()` 讀取到的內容都是字串，也就是讀取到的是JSON格式的字串。
- 需要透過`JSON.parse()`轉換成JSON 物件。

【Key Points】：

`fs.readFileSync()` 採用同步的方式讀取檔案內容

`fs.readFileSync()` 讀取到的內容都是字串

需要透過`JSON.parse()`轉換成JSON 物件

13-2：以表格呈現內容

- HTML表格的語法
- 以 `res.render()`渲染視圖(view)
- 將資料代入表格的標籤之中
- 由程式構成表格內容，整批代入視圖



designed by freepik



designed by freepik

13-7

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

- HTML 5是透過`<table></table>`標籤顯示網頁表格。
- `<thead></thead>`可以定義表格表頭。
- 表格內容主要放在`<tbody></tbody>`裡，透過`<tr></tr>` or `<td></td>`設計表格內容。
- `res.render()`第一個參數為字串型態，可以指定view 視圖檔案，第二個參數為物件型態，將作為資料傳遞至view。
- `res.render()`中的物件，key會對應 `<%={Object.key} %>`。

【Key Points】：

HTML表格的語法

以 `res.render()`渲染視圖(view)

將資料代入表格的標籤之中

HTML表格的語法

- 表格以「橫列直欄」二維的方式呈現資料
- HTML 5 透過 `<table></table>` 處理網頁表格
- `<thead></thead>` 收納表格的欄位標題
- 表格內容主要放在 `<tbody><tbody>` 裡
- `<tbody>` 內含一個以上的 `<tr>`，有多少個`<tr></tr>`就代表表格有幾個橫列
- `<td></td>` 放入 `<tr></tr>` 之內，有多少個`<td></td>`在一個橫列，就代表該列有幾欄

```
<body> 前端網頁部分
<table>
  <thead>
    <th>first</th>
    <th>second</th>
  </thead>
  <tbody>
    <tr>
      <td>1</td>
      <td>2</td>
    </tr>
    <tr>
      <td>3</td>
      <td>4</td>
    </tr>
  </tbody>
</table>
</body>
```

13-8



- 表格以「橫列直欄」二維的方式呈現資料。
- HTML 5 透過 `<table></table>` 元素處理網頁表格。
- `<thead></thead>` 定義表格的表頭，通常在這裡放入表格的欄位標題。
- 表格內容主要放在 `<tbody><tbody>` 裡，`<tbody>` 內含一個以上的 `<tr>`。
- 有多少個`<tr></tr>`就代表表格有幾個橫列。
- `<td></td>` 放入 `<tr></tr>` 之內，有多少個`<td></td>`在一個橫列，就代表該列有幾欄。

【Key Points】：

HTML 5 透過 `<table></table>` 處理網頁表格

`<tbody>` 內含一個以上的 `<tr>`

`<td></td>` 放入 `<tr></tr>` 之內

以 res.render() 渲染視圖(view)

- 渲染(render) 的作用是將視圖(view)內的HTML，結合資料，動態產生 HTML 內容，然後將產生好的內容傳給用戶端。
- res.render() 第一個參數為字串型態，用來指定視圖(view)的檔名
- res.render() 第二個參數為物件型態，用來將資料傳遞至視圖(view)
- 舉例來說：
app.render("index", { hello: "Hello! World" });
index.ejs 內就可以用 <p><%= hello %></p>
輸出 hello屬性的屬性值，最後的結果就是
「<p>Hello! World</p>」

13-9



- res.render() 第一個參數為字串型態，用來指定視圖(view)的檔名
- res.render() 第二個參數為物件型態，用來將資料傳遞至視圖(view)。
- 傳遞至視圖(view)的物件，其屬性名稱會對應到視圖(view) 中的 <%=名稱%>。
- 舉例來說: app.render("index", { hello: "Hello! World" });
index.ejs 內就可以用 <%= hello %> 輸出 hello屬性的屬性值，也就是 「Hello! World」
- 以 <%= %> 輸出時，會轉譯HTML符號，
會轉譯為

- <%- %> 則輸出「非轉譯的內容」(
不會轉譯為
)

【Key Points】：

res.render() 第一個參數為字串型態，用來指定視圖(view)的檔名
res.render() 第二個參數為物件型態，用來將資料傳遞至視圖(view)
傳遞至視圖(view)的物件，其屬性名稱會對應到視圖(view) 中的 <%=名稱%>

將資料代入表格的標籤之中

- 透過 ejs 渲染HTML，可以從後端傳遞資料至前端
- 各個物件屬性，依照名稱一一將其內容嵌在表格之中

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

/* GET home page */
router.get('/', function(req, res, next) {
  res.render('index', {
    firstHead: 'first',
    secondHead: 'second',
    firstRowFirstTd: '1',
    firstRowSecTd: '2',
    secRowFirstTd: '3',
    secRowSecTd: '4'
  });
}

module.exports = router;
```

後端Router部分

```
<!DOCTYPE html>
<html>
<head>
  <link rel='stylesheet' href='/stylesheets/style.css' />
</head>
<body>
<table>
<thead>
  <th><%= firstHead %></th>
  <th><%= secondHead %></th>
</thead>
<tbody>
<tr>
  <td><%= firstRowFirstTd %></td>
  <td><%= firstRowSecTd %></td>
</tr>
<tr>
  <td><%= secRowFirstTd %></td>
  <td><%= secRowSecTd %></td>
</tr>
</tbody>
</table>
</body>
</html>
```

前端網頁部分

13-10



- 透過 ejs 渲染HTML，可以從後端傳遞資料至前端。
- 可以看出HTML 仍然保持原有的格式，ejs頂多是增加了它自己的語法。
- 由於只是傳遞資料不傳遞HTML tag，所以採用<%= %>。
- res.render() 的 Object key 會對應到<%= {Object.key} %>
- 如果 <%= {Object.key} %> 無法與 res.render() 中的物件key匹配，會因為出錯而回傳 500 錯誤碼。

<Note> 此處的範例旨在入門階段示範語法，如果有大量的資料一再地要嵌入 <tr>，通常會以迴圈進行。

【Key Points】：

透過 ejs 渲染HTML，可以從後端傳遞資料至前端
各個物件屬性，依照名稱一一將其內容嵌在表格之中
如果有大量的資料一再地要嵌入 <tr>，通常會以迴圈進行

由程式構成表格內容，整批代入視圖

1. 先用程式構成表格標籤與內容

2. 再用

`res.render("viewName", { table: htmlTable })`

帶入表格給視圖(view)

3. 視圖(view) 透過

`<%- table %>`

渲染出 HTML 內容

The screenshot shows a code editor with a file tree on the left and a code editor on the right. The file tree includes files like index.js, routes, views, error.ejs, index.ejs, users.js, and app.js. The code editor displays a portion of index.js with a red box highlighting the section where a table is generated programmatically.

```
1 const express = require('express');
2 const router = express.Router();
3
4 /* GET home page. */
5 router.get('/', function(req, res, next) {
6   res.render('index', {
7     table: [
8       {
9         first: 'John',
10        second: 'Doe'
11      },
12      {
13        first: 'Jane',
14        second: 'Doe'
15      },
16      {
17        first: 'Mike',
18        second: 'Smith'
19      },
20      {
21        first: 'Sarah',
22        second: 'Johnson'
23      }
24    ]
25  });
26
27
28  module.exports = router;
29 }
```

13-11



- 由程式構成表格內容，整批代入視圖。
- 先用程式構成表格標籤與內容，例如: `let htmlTable = "<table><tr>...</tr>...</table>"`
- 再用 `res.render("viewName", { table: htmlTable })` 帶入表格給視圖(view)
- 視圖(view) 透過`<%- table %>` 渲染 HTML 內容。
- 因為 `table` 的內容含有標籤，請留意要使用 `<%- %>` 輸出，以保留小於大於符號。
- 這種做法漸漸不流行，現在講求前端後端各司其職。全部由程式處理畫面與資料，在套用美術樣式與後續改版時，比較不便於團隊分工。

【Key Points】：

先用程式構成表格標籤與內容，例如: `let htmlTable = "<table><tr>...</tr>...</table>"`

再用 `res.render("viewName", { table: htmlTable })` 帶入表格給視圖(view)

視圖(view) 透過`<%- table %>` 渲染 HTML 內容

13-3：使用 EJS include()

- 主板頁面觀念
- 標題及導覽列
- 頁面內容
- 網頁頁尾



designed by freepik



designed by freepik

13-12

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

- 如果有設計主板頁面就可以更改需要更改的地方即可。
- <% include {eJS 檔案 name} %>可以將其他ejs網頁的內容含入進當前網頁中。
- <% include {eJS 檔案 name} %> include可以傳入相對位置的路徑。
- <% include {eJS 檔案 name} %> include也可以傳入絕對位置的路徑。
- 透過切分網頁與善用透過 <% include %>就可以把一個複雜網頁切分乾淨且重複使用主板頁面。

【Key Points】：

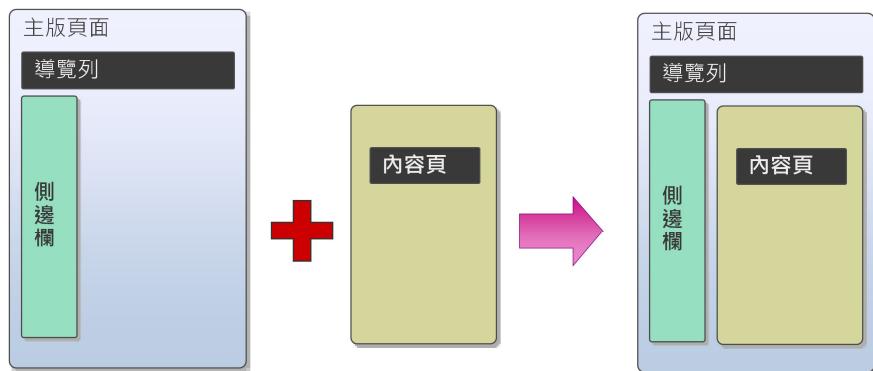
主板頁面觀念

<% include {eJS 檔案 name} %>

標題及導覽列、頁面內容、網頁頁尾

主板頁面觀念

- 固定的內容，放在主板頁面
- 變動的內容，放在內容頁
- 將主板頁面的內容，嵌進來，合成完整的一頁



13-13

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

- 一般的內容網頁常具有標題、內容、頁尾。
- 頁尾的內容在網站內的每個網頁都是一樣，例如「聯絡我們」等資訊。
- 於是，可以考慮將「頁尾的內容」獨立成一個檔案，然後，再嵌進各網頁之中。
- 如果有設計主板頁面就可以更改需要更改的地方即可。
- 固定的內容，放在主板頁面。
- 變動的內容，放在內容頁。
- 將主板頁面的內容，嵌進來，合成完整的一頁。

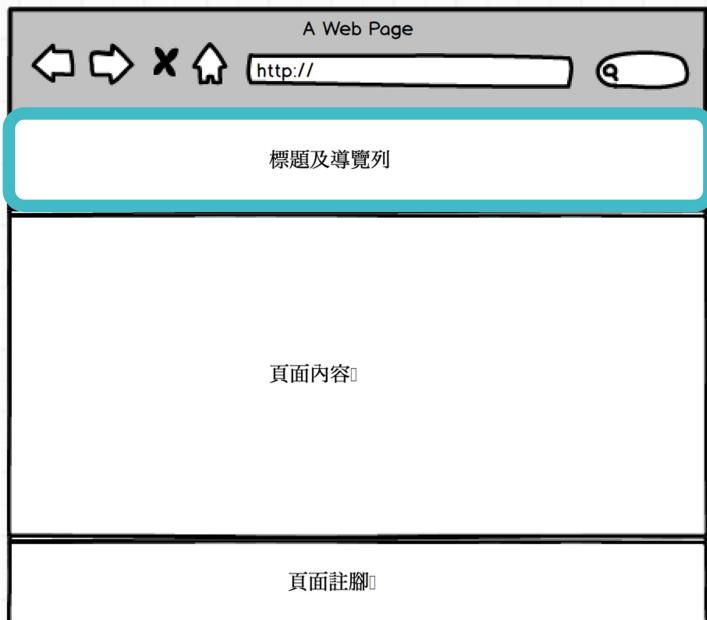
【Key Points】：

固定的內容，放在主板頁面

變動的內容，放在內容頁

將主板頁面的內容，嵌進來，合成完整的一頁

標題及導覽列



- 標題及導覽列的位置？
- 標題及導覽列包含什麼資訊？
- 以 <%- include() %> 嵌入內容到主畫面

13-14

- header 儲存頁面標題等資訊
- header 內的 nav 導覽列，包含頁面間的超連結設定
- 各頁的導覽列如果相同或近似，可另存內容成例如 header.ejs
- 以 <%- include() %> 嵌入內容到主畫面，例如：

```
-<%- include('header') %>
HTML code
<%- include('footer') %>
```

-Header.esj 的內容亦包括 meta 標籤、JavaScript 程式庫連結與 CSS 連結

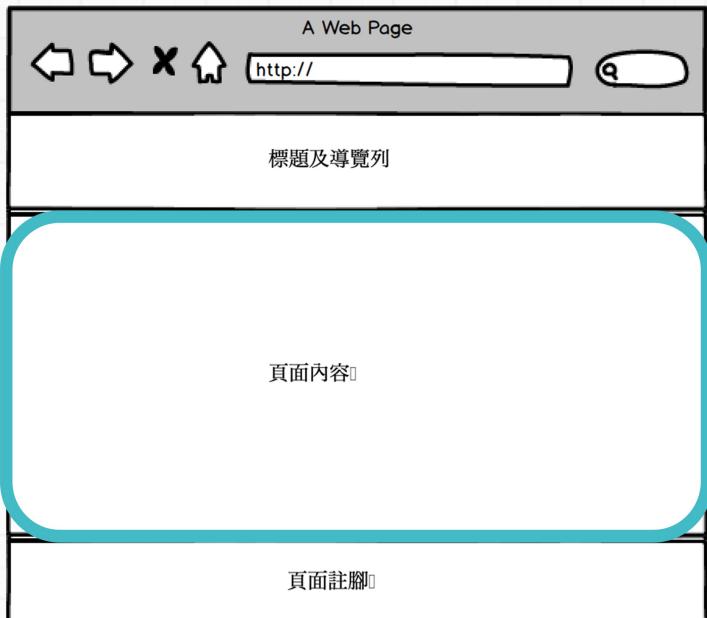
【Key Points】：

header 儲存頁面標題等資訊

header 內的 nav 導覽列，包含頁面間的超連結設定

以 <%- include() %> 嵌入內容到主畫面

頁面內容



- 頁面主要呈現的內容
- 程式自資料庫提取資料後，以 `res.render()` 將資料傳入 EJS 引擎，再以 `<%= %>` 輸出
- 頁面可以使用 `if, for` 進行流程控制：
`<% if () { %> ... <% } %>`
`<% for () { %> ... <% } %>`

13-15



- 頁面主要呈現的內容
- 通常包括文字圖片影音各資訊
- 典型的作法：程式自資料庫提取資料後，以 `res.render()` 將資料傳入 EJS 引擎，再以 `<%= %>` 嵌入 HTML 文件
- 圖片及影音的網址，也可從資料庫中提取。
- 頁面可以使用 `if, for` 進行流程控制：
`<% if () { %> ... <% } %>`
`<% for () { %> ... <% } %>`

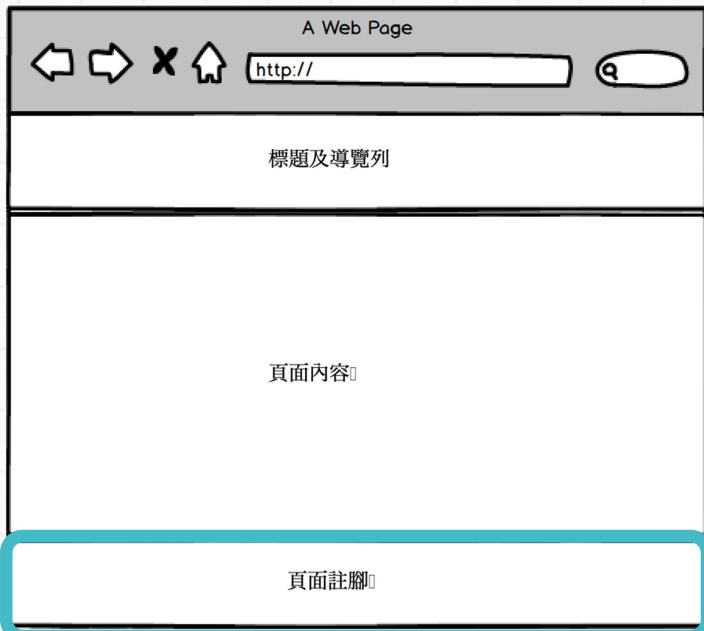
【Key Points】：

頁面主要呈現的內容

程式自資料庫提取資料，以 `res.render()` 將資料傳入 EJS 引擎，再以 `<%= %>` 嵌入 HTML 文件

頁面可以使用 `if, for` 進行流程控制

網頁頁尾



- 通常置於網頁的最下方
- 網頁的註腳包含哪些資訊？
- 內容通常重複使用，建議以
`<%- include("footer") %>` 嵌進來

13-16

- 通常是在頁面最下方
- 提供各頁面快速連結（例如連絡資訊）
- 包括版權資訊、隱私權宣告、公開政策等內容或連結。
- 內容通常重複使用
- 建議以 `<%- include("footer") %>` 嵌進來

【Key Points】：

通常是在頁面最下方

內容通常重複使用

建議以 `<%- include("footer") %>` 嵌進來

Summary〈精華回顧〉

- 了解JSON檔的引入方式。
- 了解後端傳ejs 資料傳遞方式。
- 了解`<%= %>`的用法。
- 了解`<%- %>`的用法。
- 了解`<% include %>`的用法。



13-17

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

- Require不只可以引用模組，也可以引用JSON 檔案。
- `res.render()`第一個參數為字串型態，用來指定 view 視圖檔案名稱；第二個參數為物件型態，用來將資料傳遞給視圖(view)。
- `<%= {Object.key} =>`是ejs的語法，可以將資料渲染至HTML畫面。
- `<%- %>`與 `<%= %>`差別在`<%- ->`可以渲染HTML 語法，`<%= %>` 則會進行 HTML 特殊字元轉譯。
- `<% include {eJS 檔案 name} %>`可以將其他 ejs 網頁的內容嵌進當前網頁。

【Key Points】：

Require不只可以引用模組，也可以引用JSON 檔案

`<%- %>`與 `<%= %>`的差別

`<% include {eJS 檔案 name} %>`可以將其他 ejs 網頁的內容嵌進當前網頁

線上程式題

- **13-1 如何 require 引用 JSON 檔**

請修改以下程式碼，使其能夠正常引用JSON檔。

```
const teacher=require('teacher');
```

- **13-2 如何傳遞資料到視圖(View)**

請修改以下程式碼，使其能夠正常 res.render()。

```
router.get('/', function(req, res, next) {  
    res.render('index',[{count:100}]);  
});
```

- **13-3 如何含入(嵌入)其他的 view page**

請修改程式碼，使其能夠正常含入(嵌入)其他view page。

13-18



13-1 如何 require 引用 JSON 檔

請修改以下程式碼，使其能夠正常引用JSON檔。

```
const teacher=require('teacher');
```

13-2 如何傳遞資料到視圖(View)

請修改以下程式碼，使其能夠正常 res.render()。

```
router.get('/', function(req, res, next) {  
    res.render('index',[{count:100}]);  
});
```

13-3 如何含入(嵌入)其他的 view page

請修改程式碼，使其能夠正常含入(嵌入)其他view page。

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】：

13-1 如何 require 引用 JSON 檔

13-2 如何傳遞資料到視圖(View)

13-3 如何含入(嵌入)其他的 view page

課後練習題(Lab)

- **情節描述:**

JSON格式是常見的資料格式，請使用express專案，並且在其根目錄引用JSON檔，透過res.render()渲染JSON資料至HTML表格中。

- **預設目標:**

- 理解JSON引用
- 理解res.render()
- 理解ejs語法

- **Lab01:理解JSON引用**

Estimated time:

20 minutes

- **Lab02:理解res.render()**

- **Lab03:理解ejs語法**

13-19



【情節描述】

JSON格式是常見的資料格式，請使用express專案，並且在其根目錄引用JSON檔，透過res.render()渲染JSON資料至HTML表格中。

【預設目標】

- 理解JSON引用
- 理解res.render()
- 理解ejs語法

Lab01:理解JSON引用

Lab02:理解res.render()

Lab03:理解ejs語法

完成後的程式與檔案，請參考 Example 資料夾的內容。

【Key Points】：

Lab01:理解JSON引用

Lab02:理解res.render()

Lab03:理解ejs語法

範例程式使用說明

- 範例程式資料夾: Module_13_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:3001

13-20



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:3001

【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: 「npm start」