



使用 session



designed by freepik

Estimated time:

50 min.

 資訊工業策進會 Institute for Information Industry

【Key Points】：

學習目標

- 11-1: 安裝設定 express-session
- 11-2: 使用 session
- 11-3: 簡單的登入功能



11-1

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

本篇將介紹什麼是session以及如何運用。具體的內容，分點條列如下：

- Cookie與session
- Session儲存方式
- express-session主要設定介紹
- 存取 session 的值
- 建立登入表單
- 實作登入流程

【Key Points】：

Cookie與session

存取 session 的值

建立登入表單，實作登入流程

11-1：安裝設定 express-session

- Cookie與Session
- Session儲存方式
- express-session主要設定介紹



designed by freepik



designed by freepik

11-2

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

這裡我們會介紹

- 什麼是 Session
- 什麼是 Cookie
- 創建專案資料夾
- 安裝express-session
- 如何設定express-session

【Key Points】：

Cookie與Session

Session儲存方式

express-session主要設定介紹

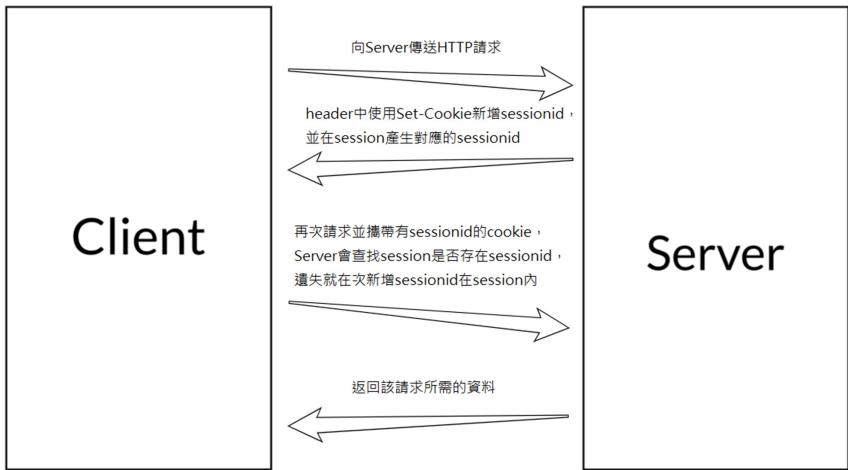
Cookie與session

- 什麼是cookie

- 儲存於Client的資料
- 後續請求皆會帶上這段資料

- 什麼是session

- 儲存於Server的資料
- Client 端只存SessionID



11-3

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

- cookie 是伺服器傳給瀏覽器的一段數據(通過 Set-Cookie header)
- 瀏覽器會保存這段數據一段時間，此後，每次訪問該伺服器都會自動帶上這段數據。
- 服務端將 SessionID (隨機數) 通過 Set-Cookie 發給客戶端
- 客戶端訪問伺服器時，都會自動帶上這段SessionID，服務端讀取 SessionID 來驗證
- 以 SessionID 為線索，服務端保存了所有與該 SessionID 有關的內容 (採 key-value 的 hash table 形式)

【Key Points】：

cookie 是伺服器傳給瀏覽器的一段數據

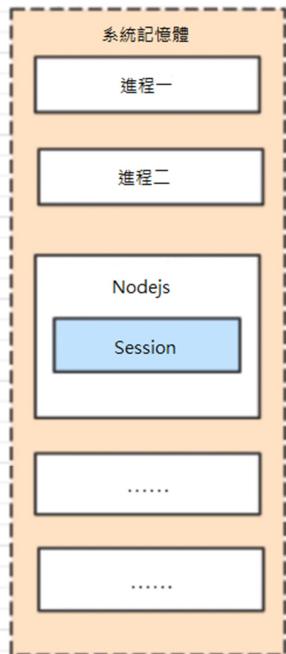
瀏覽器會保存這段數據一段時間，此後每次訪問該伺服器都會自動帶上這段數據

服務端將 SessionID (隨機數) 發給客戶端，通過 SessionID 可以獲取對應的用戶隱私信息

Session儲存方式

- **session儲存方式**

- Node.js 記憶體
- Redis
 - 分散式的key-value資料庫
- File
 - 暫存檔形式
- DB
 - MySQL、SQL Server



11-4

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

- 預設使用Node.js行程(process)中的記憶體來存 Session 資訊，這麼做的缺點有兩個：
 - 過多的session會造成node記憶體不足的窘境
 - 如果未來使用pm2或是集成類的服務(docker)會造成各行程(process)存放的session值不一致
- 使用專門儲存值在記憶體的Redis來存放session，可以保存的值有字符串(String), 雜湊(Hash), 列表(list), 集合(sets) 和有序集合(sorted sets)，為存放session最佳方式。
- 使用暫存檔來存放，會消耗I/O效能，且造成硬碟負擔
- 使用資料庫，相對於redis，訪問速度較慢
- file和redis這兩種作法，將會在課後練習(Lab)實際演練

【Key Points】：

session儲存方式: 記憶體、Redis、File、DB

過多的session會造成node記憶體不足的窘境

使用專門儲存值在記憶體的Redis來存放session，為存放session最佳方式

創立專案資料夾

- 建立資料夾: **mkdir session**
- 進入**session** 資料夾: **cd session**
- 初始化專案: **npm init -f**
- **npm install express express-session**
- 新增 **index.js**

```
C:\Users\Administrator\Desktop\教材>mkdir session  
C:\Users\Administrator\Desktop\教材>cd session  
C:\Users\Administrator\Desktop\教材\session>npm init
```

```
C:\Users\Administrator\Desktop\教材\session>npm install express express-session  
npm notice created a lockfile as package-lock.json. You should commit this file.  
npm WARN session@1.0.0 No description  
npm WARN session@1.0.0 No repository field.  
+ express-session@1.17.0  
+ express@4.17.1  
added 56 packages from 38 contributors and audited 137 packages in 1.759s  
found 0 vulnerabilities
```

11-5



1. mkdir session 建立資料夾 session
2. 進入**session** 資料夾
3. 初始化專案
4. 安裝 **express**和**express-seseion**
5. 最後用VSCode或其他IDE建立**index.js**

【Key Points】：

初始化專案 **npm init -f**

安裝 **express**和**express-seseion**

最後用VSCode建立**index.js**

express-session主要設定介紹

- **secret**: 型態為字串，作為Server端生成session的簽章
- **name**: 儲存在Client端的key的名稱，預設為connect.sid
- **resave**: 每次請求時，是否對session進行修改並覆寫
- **saveUninitialized**: 是否保存未初始化session
- **store**: session的儲存方式，預設為記憶體
- **Cookie**: {
 - path**: cookie存放在Client端的路徑，預設為/
 - httpOnly**: 是否只能由Server更改Cookie
 - secure**: 是否只在HTTPS協定中使用Cookie
 - maxAge**: 設定cookie存活時間，以毫秒為單位}

11-6



1. Secret: 型態為字串，作為Server端生成session的簽章。
2. Name: 儲存在Client端的key的名稱，預設為 connect.sid
3. Resave: 每次請求時，是否對session進行修改並覆寫
4. SaveUninitialized: 當首次沒有session的Client訪問時，是否要先保存session
5. Store: session的儲存方式，預設為記憶體
6. Cookie配置
 - path: 儲存Cookie的路徑
 - httpOnly: 是否只在HTTPS協定中使用Cookie, 能避免Cookie在Client直接被修改
 - secure 如果是使用HTTP的話，不會返回cookie，直到使用HTTPS
 - maxAge: 設定cookie存活時間，以毫秒為單位

【Key Points】:

Name 即是 Cookie中sessionid的key
httpOnly能避免Cookie在Client直接被修改
maxAge: session存活時間

11-2: 使用 session

- express-session 設定
- 存取 session 的值
- 情境範例



designed by freepik



designed by freepik

11-7

 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：

- 引用 express 模組
- 引用 express-session 模組
- 設定 express-session
- app.use(session(...)) 掛載 express-session
- 存取 session 的值
- 寫入資料到 Session
- 從 Session 讀出資料
- 透過情境範例說明系統如何運作

【Key Points】：

設定 express-session

app.use(session(...)) 掛載 express-session

存取 Session 的資料

express-session 設定

- 引入 express express-session
- 引入 express
- 設定 session 參數選項
- 以 app.use() 掛入 express-session 中介程式

```
var express = require('express');
var session = require('express-session');
var app = express();
app.use(session({
    secret: 'asdoijisjdfiogjfdiogjdfoi',
    resave: true,
    saveUninitialized: true,
    cookie: {
        path: '/',
        httpOnly: true,
        secure: false,
        maxAge: 5*1000
    }
}))
```

11-8



引用 express 模組:

```
var express = require("express");
var app = express();
```

引用 express-session 模組:

```
var session = require("express-session");
```

呼叫 session() 函式時，傳入「選項物件」。關於選項物件，請參閱前一頁「express-session 主要設定介紹」。

最後，將 session() 函式傳回的結果，帶入 app.use() :

```
app.use( session( ... ) ); // 載入 Middleware
```

【Key Points】：

引用 express-session 模組

呼叫 session() 函式時，傳入「選項物件」

最後，將 session() 函式傳回的結果，帶入 app.use()

存取 session 的值

- 寫入資料到 Session:

```
req.session.Key = value;
```

例如: `req.session.userId = "007";`

- 從 Session 讀出資料:

```
var userId = req.session.userId;
```

- 如何確認 Session 內有我們要的資料:

```
if (req.session.userId) {  
    var userId = req.session.userId;  
    // ....  
}
```

11-9



- Session 是以 `key: value` 的形式儲存
- 存取的方式是使用 `req.session.key`，而非 `res.session.key`
- 寫入資料到 Session
`req.session.Key = value;`
例如: `req.session.userId = "007";`
- Key 的名稱可以自行定義。
- 在讀取 key 之前，若是不曾先設定過內容，因為該 key 沒有值，會返回 `undefined`。
- `Undefined` 在 if 會被判定成 `false`，因此，我們可以使用 if 判斷某個 Key 是否存在，如果有的話，才好讀取該 key 對應到的內容。

【Key Points】：

session是以key: value的形式儲存

存取方式是使用 `req.session.key`，而不是 `res.session.key`

使用 if 判斷是否某個 key 是否存在

存取Session值的程式示範

- 建立兩個路由: / 和 /name，皆為GET方法
- 判斷在req.session是否有 views 這個值，決定輸出不同的內容
- /name 這個路由再示範一次如何寫入與讀出 req.session.name 的值

```
app.get('/', function(req, res){  
    res.setHeader('Content-Type', 'text/html; charset=utf-8')  
    //req.session.views 為自己定義的值  
    if (req.session.views) {  
        req.session.views++  
        res.write('<p>瀏覽次數: ' + req.session.views + '</p>')  
        res.write('<p>Cookie ID: ' + req.session.id + '</p>')  
        res.write('<p>Cookie過期時間: ' + (req.session.cookie.maxAge / 1000) + '秒</p>')  
        res.end()  
    } else {  
        req.session.views = 1  
        res.end('您是第一次訪問！')  
    }  
})  
app.get('/name', function(req, res){  
    req.session.name = 'session中的name'  
    res.setHeader('Content-Type', 'text/html; charset=utf-8')  
    res.write('<p>name: ' + req.session.name + '</p>')  
    res.end()  
})  
app.listen(3000);
```

11-10



- 建立兩個路由: / 和 /name，皆為GET方法
- 先說 / 這個路由。一開始，判斷在req.session是否有 views 這個值，
- 如果沒有session.views，則輸出「您是第一次訪問」到瀏覽器，並且設定 session.views 的值為壹。
- 如果有session.views，則遞增 session.views 的值並輸出「瀏覽次數」到瀏覽器
- COOKIE過期後，session.id 也會不一樣
- /name 這個路由再示範一次如何寫入與讀出 req.session.name 的值。

【Key Points】：

如何判斷在 req.session 是否有 views 這個值

重複使用 session.views 的值

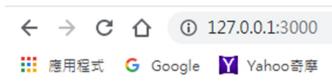
COOKIE過期後，session.id 也會不一樣

情境範例

- node index.js 啟動 Web Server，開啟瀏覽器訪問：

- <http://127.0.0.1:3000>

- <http://127.0.0.1:3000/name>



您是第一次訪問！

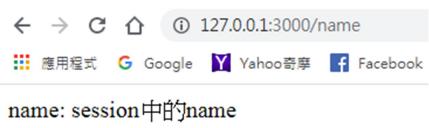


瀏覽次數: 2

Cookie ID: E8431a2GvqHwF9KQFYBAMWJX0I_53yj1

Cookie過期時間: 5秒

C:\Users\Administrator\Desktop\教材\session>node index.js



瀏覽次數: 7

Cookie ID: FlGgyhnFkzafopLcZZAnRx02XPO19ZUA

Cookie過期時間: 5秒

11-11

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

1. 在終端機輸入指令: node index.js 啟動 Web Server
2. 開啟瀏覽器訪問<http://127.0.0.1:3000/>，重新整理多訪問幾次看看。
3. 可以看到Client和Server的訊息皆有在期限內被記錄
4. 多次訪問<http://127.0.0.1:3000/name>
5. 查看 session 中 name 儲存什麼字串

【Key Points】：

開啟瀏覽器訪問<http://127.0.0.1:3000/>

重新整理多訪問幾次看看

查看session中name儲存什麼字串

11-3：簡單的登入功能

- 安裝body-parser
- 撰寫登入邏輯
- 建立登入表單
- 登入功能測試



designed by freepik



designed by freepik

11-12

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

接下來，將介紹：

- 一個叫body-parser的模組，專門用來解析POST傳來的內容
- 以 POST 傳送而來的表單資料，各欄位的內容都登錄在req.body 名下。
- 撰寫登入邏輯
- 建立登入表單
- 登入流程
- 登入功能測試

【Key Points】：

以 POST 傳送而來的表單資料，各欄位的內容都登錄在req.body 名下
建立登入表單
登入流程程式設計與測試

安裝body-parser

- 安裝步驟
 - npm install body-parser
 - 變更index.js原先配置
 - 以 app.use() 掛入 body-parser
- 以 POST 傳送而來的表單資料，各欄位的內容都登錄在req.body
- 使用方式
 - req.body.變數名稱

```
C:\Users\Administrator\Desktop\11_1ing\session>npm install body-parser
npm WARN session@1.0.0 No description
npm WARN session@1.0.0 No repository field.

+ body-parser@1.19.0
updated 1 package and audited 170 packages in 2.734s
found 0 vulnerabilities
```

```
var express = require('express');
var session = require('express-session');
var bodyParser = require('body-parser');
var app = express();
//用來解析POST資料的設定
app.use(bodyParser.urlencoded({ extended: false }));
```

11-13



- 因為提交表單內含帳號密碼敏感資訊，不建議使用GET讓值暴露在URL之中，建議選擇POST。
- 引入一個叫body-parser的模組，專門用來解析POST傳來的內容。
- 以 app.use() 掛入 body-parser
- 以 POST 傳送而來的表單資料，各欄位的內容都登錄在req.body 名下。
- 舉例來說，表單有一個文字輸入方塊 name="account"，表單POST到伺服端後，以 req.body.account 可讀出該欄資料。

【Key Points】：

因為提交表單保有帳號密碼敏感資訊，不建議使用GET讓值暴露在URL之中
引入一個叫body-parser的模組，專門用來解析POST傳來的內容
想要查看該變數是否有值，可以使用req.body.XXX來做判斷

撰寫登入邏輯

- 兩個GET路由 /form 和 /login
 - Get /form 路由傳回表單給瀏覽器
 - Get /login 判斷是否登入
未登入，由 /form 路由傳回表單
已登入，傳送「你已經登入囉」給瀏覽器
- 一個POST路由 /login
 - 判斷登入狀態
 - 處理登入邏輯

```
app.get('/login', function(req, res){  
    //判斷是否已經登入  
    if(req.session.user) {  
        res.setHeader('Content-Type', 'text/html; charset=utf-8')  
        res.write('<p>' + req.session.user.account + '你已經登入囉!</p>')  
        res.write('<p>登入時間:' + req.session.user.logged_in_at + '</p>')  
        res.end()  
    } else {  
        res.redirect('/form')  
    }  
})  
app.post('/login', function(req, res){  
    //判斷是否已經登入  
    if(req.session.user) {  
        res.write('<p>' + req.session.user.account + '你已經登入囉!</p>')  
        res.write('<p>登入時間:' + req.session.user.logged_in_at + '</p>')  
    } else {  
        //判斷是否有傳遞值  
        if(req.body.account == undefined || req.body.password == undefined) {  
            res.write('<p>沒有傳遞值，無法登入哦!</p>')  
        } else {  
            //預設帳號密碼  
            const account = 'user'  
            const password = 'user'  
            //處理請求  
            if(req.body.account === account && req.body.password === password) {  
                res.write('<p>登入成功!</p>')  
                //將登入狀態寫入 session  
                req.session.user = {  
                    'account': req.body.account,  
                    'logged_in_at': new Date(Date.now())  
                }  
            } else {  
                res.write('<p>登入失敗，帳號或密碼錯誤!</p>')  
            }  
        }  
    }  
    res.end()  
})
```

11-14



Get /form

傳回表單給瀏覽器。關於表單如何製作，次頁會有說明。

GET /login

1. 一開始，先判斷該用戶是否已登入。
2. 如果已登入，傳送「你已經登入囉」給瀏覽器。
3. 沒有尚未登入，重導到 /form 路由。

POST /login

1. 一開始先確認是否有登入
2. 如果已登入，傳送「上次登入時間」給瀏覽器。
3. 沒有尚未登入
 1. 檢查用戶端是否有提交帳號、密碼資料
 2. 檢查req.body.account和req.body.password的內容是否正確

<Note>

- 目前還沒有建立User資料表，所以我先把帳號密碼寫死在程式裡頭(不建議)進行比對
- 要求瀏覽重導的程式寫法是: res.redirect("路由路徑");

【Key Points】：

經由 /form 路由，取得登入表單

表單POST送出account與password兩個欄位

用req.body.account和req.body.password讀出資料並且判斷是否登入成功

建立登入表單

- 建立index.html
- <form method="get" action="/login">
- 內含
account 與
password 欄位
- 一個button，
type為 submit

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>登入</title>
</head>
<body>
    <!-- <form action="/login" method="GET"> -->

    <form action="/login" method="POST">
        帳號: <input type="text" name="account">
        密碼: <input type="password" name="password">
        <button type="submit">送出</button>
    </form>

</body>
</html>
```

11-15



1. 使用 VS Code 建立新檔案，檔名: index.html
 2. 一開始，可以先打「html」，照理說，會有自動完成提示，選擇html:5，VS Code 會建立一個基本架構的HTML文件
 3. 再來，加入form元素，針對<form>開始標籤，設定兩個重要的設定: action="" 為我們提交表單後要送去的路由，method="" 則為方法，常用的有GET、POST。本次以POST示範。
 4. 再來，建立兩個input控制項，其中之一是 name="account" type="text" 的文字輸入方塊，
 5. 另一個是 name="password" type="password" 的密碼輸入欄位，
 6. 最後，置入一個按鈕，type = "submit"
- <Note>name的設定值若是XXX，後端的程式從 req.body.XXX 讀資料。

【Key Points】：

<form method="post" action="/login">
內含 account 與 password 欄位
name的設定值若是XXX，後端的程式從 req.body.XXX 讀資料。

登入功能測試

- 訪問 /login，被跳轉到 /form
- 進入 /form

— <http://127.0.0.1:3000/form>

— 假設輸入錯的帳號密碼：登入流



帳號: 密碼: 送出



11-16

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

- 如果我們直接訪問 /login，會有兩種情況：
- 在已經登入的情況下，畫面會出現「上次登入時間」
- 如果未登入的情況下，則跳轉到 /form
- 訪問 /form，瀏覽器會收到空白表單
- 帳號密碼分別輸入 user、user 進行測試。(可以先輸入錯誤的帳號密碼，判斷邏輯有沒有寫錯)

【Key Points】：

訪問 /form，瀏覽器會收到空白表單

帳號密碼分別輸入 user、user

再訪問 /login 看看

登入功能測試

- 若我們輸入正確的帳號密碼，提交後則會顯示登入成功
- 登入成功後，如果重新整理的話，則出現「上次登入時間」
- cookie有時效限制，閒置時間若超過時限，登入資訊將會作廢
(變回「未登入」狀態)



11-17

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

- 接續上一張投影片，
- 若我們輸入正確的帳號密碼，提交後則會顯示登入成功
- 登入成功後，如果重新整理的話，則出現「上次登入時間」
- 因為cookie有時效限制，閒置時間若超過時限，登入資訊將會作廢
- 重新整理網頁，可重算閒置時間

【Key Points】：

若我們輸入正確的帳號密碼，提交後則會顯示登入成功
登入成功後，如果重新整理的話，則出現「上次登入時間」
cookie有時效限制，閒置時間若超過時限，登入資訊將會作廢

Summary〈精華回顧〉

- 關於**session**的原理和儲存方式
- 在**express-session**中配置**session**的設定
- 用**session**建立簡易的計數器，查看訪問次數
- 用**session**建立一個簡單登入邏輯判斷
- 建立登入表單



11-18

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

回顧一下稍早之前的內容：

- 我們了解了**session**的運作模式，並且介紹四種不同儲存**session**的方式
- 在**express-session**中簡單介紹**session**的配置和功能，並可以依照自己需求更改
- 利用**session**尚未過期的情況下，記錄Client訪問次數
- 建立登入表單
- 配合表單提交來實作一個簡單的登入系統

【Key Points】：

session的運作模式

express-session中簡單介紹**session**的配置和功能

配合表單提交來實作一個簡單的登入系統

線上程式題

- **11-1 如何參用express-session模組？**

想要參用express-session模組，程式該怎麼寫？

- **11-2 設定session的各項參數？**

- 將session的過期時間加長到7天
- 使用HTTPS才傳遞cookie
- 並且節省資源不重複儲存session

- **11-3 如何在session中賦予value變數值？**

如何在session中賦予value變數值？

11-19



題目名稱: 11-1 如何參用express-session模組？

想要參用express-session模組，程式該怎麼寫？

題目名稱: 11-2 設定session的各項參數？

程式修改要求事項:

- 將session的過期時間加長到7天
- 使用HTTPS才傳遞cookie
- 並且節省資源不重複儲存session

題目名稱: 11-3 如何在session中賦予value變數值？

有一段程式的內容如下：

```
app.get('/', function(req, res){
```

...

})

如何在session中賦予value變數值？

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】：

11-1 如何參用express-session模組？

11-2 設定session的各項參數？

11-3 如何在session中賦予value變數值？

課後練習題(Lab)

- 情節描述:
- 本練習假設已安裝 Node.js 環境於 Windows 作業系統，並且知悉如何安裝 Express。透過已學會 Express-session 處理方式，做更進階的運用
- 預設目標:
- 學會使用file 與 redis這兩種session儲存方式。
- Lab01: 使用file儲存session
- Lab02: 使用redis儲存session
- 完成後的程式與檔案，請參考 Example 資料夾的內容。

Estimated time:
20 minutes

11-20



【情節描述】

本練習假設已安裝 Node.js 環境於 Windows 作業系統，並且知悉如何安裝 Express。透過已學會 Express-session 處理方式，做更進階的運用

【預設目標】

學會使用file 與 redis這兩種session儲存方式。

關鍵程式:

```
app.use(session({  
  ...  
  store: new FileStore({  
    //指定file儲存位置  
    path: __dirname+'/sessions'  
  }),  
  ...  
}));
```

```
store: new RedisStore({client: rClient}),
```

【Key Points】:

```
store: new FileStore()
```

```
store: new RedisStore({client: rClient})
```

完成後的程式與檔案，請參考 Example 資料夾的內容

範例程式使用說明

- 範例程式資料夾: Module_11_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」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 node index.js
 7. 啟動瀏覽器，連接 http://localhost:3000

11-21



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」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 node index.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」開啟 terminal 終端機視窗，輸入：「node 主程式.js」