目　錄

1 REST 與 RESTful API

1.1 REST 導論

- REST(具象狀態傳輸/Representational State Transfer）是一組可實現有效率、可信任、可擴展的分散式系統的軟體架構設計規範(design constraints)或設計風格。
- 簡單講，REST 是一種讓 Client/Server 兩方機器利用具象(representations)透過 HTTP 協定去轉換資源狀態的方法。
- 目前應用較普遍的 Web 服務實作方案中(例如 REST、SOAP、XML-RPC 等)，因為 REST 在使用上更加簡潔，越來越多的 Web 服務開始採用 REST 模式來設計和實現。例如 Amazon.com、雅虎等公司所提供的一些 Web 服務皆採用了 REST。

1.2 何謂 RESTful API

- 一個軟體系統如果能維持 REST 規範，它就是 RESTful。
- 同樣地，一個 Web API 如果能維持 REST 規範，它就是 RESTful API。
- RESTful API 提供的服務可讓各種軟體(如 Web 應用程式、桌面應用程式等)或各類裝置(如手機、物聯網裝置等)呼叫存取。例如：
  - ✓ 使用者以 GET 請求方法 + /api/employees/ 可攫取伺服端的所有員工資料
  - ✓ 使用者以 GET 請求方法 + /api/employees/1 可攫取伺服端的 ID=1 的員工資料
  - ✓ 使用者以 POST 請求方法 + /api/employees/ 可在伺服端新增一個員工資料
  - ✓ 使用者以 PUT 請求方法 + /api/employees/1 可修改伺服端的 ID=1 的員工資料
  - ✓ 使用者以 DELETE 請求方法 + /api/employees/1 可刪除伺服端的 ID=1 的員工資料

1.3 實作 RESTful API 的關鍵技術

1.3.1 使用 HTTP 通訊協定的「Request URI」來識別資源

- 「全球資訊網」(World Wide Web)的資料通信是以 HTTP 通訊協定為基礎。
- HTTP 最初的設計目的是為了提供一種發佈和接收特定資源(例如 HTML 網頁)的方法，其中特定的資源是以「統一資源識別碼」(Uniform Resource Identifier / URI)來標識的。

1.3.2 使用 HTTP 通訊協定的「Request methods」來傳輸(轉換)資源的狀態

- HTTP/1.1 是現今最被廣泛使用的 HTTP 的版本。它是「全球資訊網協會」(World Wide Web Consortium / W3C）和「網際網路工程任務組」(Internet Engineering Task Force / IETF）於 1999 年 6 月共同協調所發佈的版本。
- HTTP/1.1 中共定義了九種請求方法(也可稱為 HTTP 動詞)，讓你可以使用不同的方式操作指定的資源，以實現資源狀態傳輸(轉換)的需求，就好像我們對資料庫進行 CRUD (create / read / update / delete)操作後，改變了資料庫的狀態一般。
- 常用的 HTTP 請求方法計有：
  - ✓ GET
    - ✧ GET 方法請求展示指定資源。
    - ✧ 使用 GET 的請求只應用於取得資料。
  - ✓ POST
    - ✧ POST 方法用於提交指定資源的實體，通常會改變伺服器的狀態。
  - ✓ PUT
    - ✧ PUT 方法會取代指定資源的內容。
  - ✓ DELETE
    - ✧ DELETE 方法會刪除指定資源。
  - ✓ PATCH
    - ✧ PATCH 方法會對指定資源進行部份修改。

1.3.3 使用「資料交換格式」來表達資源 (資源具象化)

1.3.3.1 XML 資料格式

- W3C 於 1998 年發佈 XML(Extensible Markup Language)為其推薦標準(W3C Recommendation)。XML 其實是 SGML(Standard Generalized Markup Language)的子集(subset)。
  - ✓ 1978 年 ANSI 發佈 SGML。
  - ✓ 1986 年 ISO 將 SGML 納為其標準。
- XML 是一種類似 HTML 的標記語言，但是 XML 沒有使用預先定義的標記。因此，您可以根據自己的設計需求定義專屬於您的標記。
- XML 的基本格式是標準化的，所以如果您跨系統或平台共享或傳輸 XML，無論是在本地還是在網際網路上，接收方仍然可以根據標準化的 XML 語法法解析資料。
- XML 文件的結構
  - ✓ XML 文件以根元素(root element)為分水嶺，可以分成序言區(prolog)和主體區(body)兩大部份。
  - ✓ 序言區的組成包括 XML 宣告式(XML declaration)、處置指令(Processing instruction)、DOCTYPE 宣告式、以及註解等。各項目可視需要選用：
    - ✧ XML 宣告式
      `<?xml version="1.0" encoding="UTF-8"?>`
    - ✧ 處置指令
      `<?xml-stylesheet type="text/css" href="emps.css"?>`
    - ✧ DOCTYPE 宣告式
      `<!DOCTYPE emps PUBLIC "-//III//DTD EMPS 1.0//EN" "emps.dtd">`
    - ✧ 註解
      `<!-- Comment -->`
  - ✓ XML 文件的內容是放在主體區，其核心語法規則如下：
    - ✧ 必須有一個且只能有一個根元素(root element)。
    - ✧ 標籤成對出現，即一個開口標籤(Opening tag)必須對應一個閉合標籤(Closing tag)。
    - ✧ 元素不能互層。
    - ✧ 元素(Element)與屬性(Attribute)名稱的大小寫有差別。
    - ✧ 屬性值必須用一對相同的引號圍起來。
    - ✧ XML 定義了 5 個"predefined entities"：(使用方式：&entityName;)

Name	Character	Unicode code point (decimal)	Standard	Name
quot	"	U+0022 (34)	XML 1.0	quotation mark
amp	&	U+0026 (38)	XML 1.0	ampersand
apos	'	U+0027 (39)	XML 1.0	apostrophe
lt	<	U+003C (60)	XML 1.0	less-than sign
gt	>	U+003E (62)	XML 1.0	greater-than sign

  - ✓ 格式良好(well-formed)的 XML 文件範例：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--message的內容可有subject,body -->
<message lang="en">
    <subject>
        Greeting
    </subject>
    <body>Welcome to XML...</body>
</message>
```

1.3.3.2 json 資料格式

- JSON(JavaScript Object Notation)為將結構化資料(structured data)呈現為 JavaScript 物件的標準格式(ECMA-404 / The JSON Data Interchange Standard)，常用於網站上的資料呈現、傳輸(例如將資料從伺服器送至用戶端，以利顯示網頁)。
- JSON 是一個輕量型的資料交換格式(data-interchange format)，也是一個依照 JavaScript 物件語法所設計的資料格式，經 Douglas Crockford 推廣普及。
- 雖然 JSON 是以 JavaScript 語法為基礎，但可獨立使用，且許多程式開發環境皆可讀取(剖析)並產生 JSON。
- JSON 的官方 MIME 類型是 application/json，副檔名是.json。
- JSON 支援的資料型別有：
  - ✓ Number
  - ✓ String
    - ✧ 須以**雙引號**包圍
  - ✓ Boolean
  - ✓ Array
    - ✧ 須使用 JavaScript 的 Array literal 語法表示
  - ✓ Object
    - ✧ 須使用 JavaScript 的 Object literal 語法表示
    - ✧ 屬性名稱須以**雙引號**包圍
  - ✓ null
- JSON 格式範例：

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "height_cm": 197.6,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [],
  "spouse": null
}
```

1.3.4 使用 HTTP 通訊協定的「狀態碼」來回應資源操作的結果

- HTTP 回應的第一行是狀態行，依次是目前 HTTP 版本，3 位數字組成的狀態碼，以及描述狀態的短語，彼此由空格分隔。
- 狀態碼的第一個數字代表回應的類型：
  1. 資訊回應 (Informational responses, 100 – 199)

- ✧ 102 Processing
  此狀態碼表明伺服器收到並處理請求中，但目前未有回應。

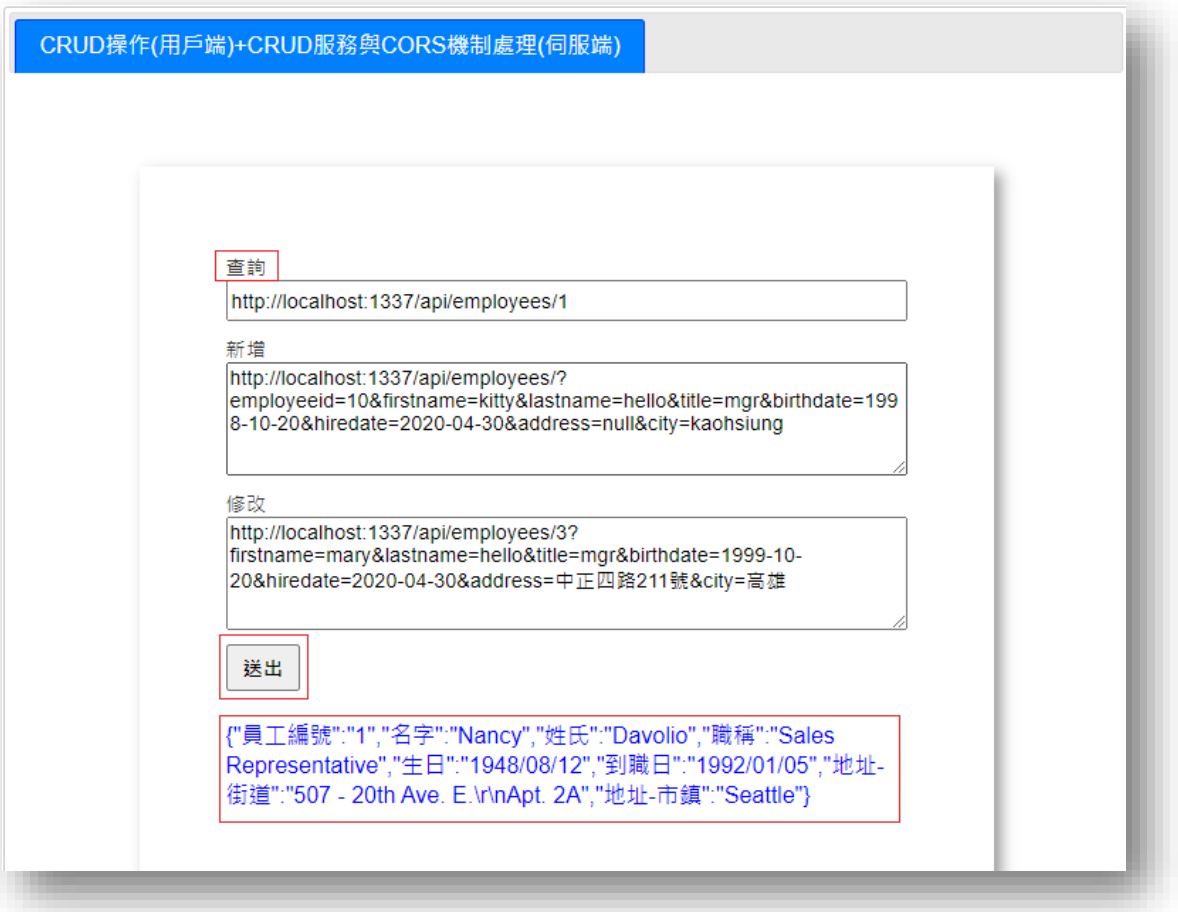
2. 成功回應 (Successful responses, 200 – 299)
   - ✧ 200 OK
     請求成功。
   - ✧ 201 Created
     請求成功且新的資源成功被創建，通常用於 POST 或一些 PUT 請求後的回應。
   - ✧ 202 Accepted
     此請求已經被接受但尚未處理。此狀態為非承諾性，代表 HTTP 無法在之後傳送一個非同步的回應告知請求的處理結果。
3. 重新導向 (Redirects, 300 – 399)
   - ✧ 301 Moved Permanently
     資源已移至它處。
   - ✧ 302 Found
     資源的網址暫時變更。(通常搭配 Location 標頭: <重新導向網址>)
   - ✧ 304 Not Modified
     請求的資源並未修改。
4. 用戶端錯誤 (Client errors, 400 – 499)
   - ✧ 400 Bad Request
     此回應意味伺服器因為收到無效語法，而無法理解請求。
   - ✧ 403 Forbidden
     用戶端並無存取權限，所以伺服器拒絕給予應有的回應。
   - ✧ 404 Not Found
     伺服器找不到請求的資源。
   - ✧ 405 Method Not Allowed
     伺服器理解此請求方法，但它被禁用或不可用。
5. 伺服器端錯誤 (Server errors, 500 – 599)
   - ✧ 500 Internal Server Error
     伺服器端發生未知或無法處理的錯誤。
   - ✧ 501 Not Implemented
     用戶端的請求目前未支援(將來有可能支援)。
   - ✧ 503 Service Unavailable
     暫停服務。

2 RESTful API 實作

2.1 CRUD 操作(用戶端)

2.1.1 Client 範例程式 T02-1.html

website3\T02-1.html




```
<!DOCTYPE html>
<html>
<head>
    <title>RESTful API 實作</title>
    <link rel="icon" href="./favicon.ico" />
    <link href="stylesheets/jquery-ui/base/jquery-ui.min.css" rel="stylesheet" />
    <link href="stylesheets/myStyleSheet.css" rel="stylesheet" />
    <script src="javascripts/jquery-3.6.0.min.js"></script>
    <script src="javascripts/jquery-ui.min.js"></script>
    <script type="text/javascript">
        $(function () {
            $("#tabs").tabs();
        });
    </script>
</head>
<body>
```

```
<div id="tabs">
    <ul>
        <li><a href="#p1"> CRUD操作(用戶端&伺服端)+CORS機制處理(伺服端)</a></li>
    </ul>
    <div id="p1">
        <div class="main-box large-padding size-500x500 left-align">
            <form id="form1" class="main-form">
                <div class="form-group">
                    <label for="text1-1">查詢</label>
                    <input class="form-control" type="text" id="text1-1" title="GET"
                           value="http://localhost:1337/api/employees/3" />
                </div>
                <div class="form-group">
                    <label for="text1-2">新增</label>
                    <textarea class="form-control height-80px" id="text1-2" title="POST">
                        http://localhost:1337/api/employees/?employeeid=10&firstname=kitty&l
                        astname=hello&title=mgr&birthdate=1998-10-20&hiredate=2020-04-
                        30&address=null&city=Kaohsiung
                    </textarea>
                </div>
                <div class="form-group">
                    <label for="text1-3">修改</label>
                    <textarea class="form-control height-80px" id="text1-3" title="PUT">
                       http://localhost:1337/api/employees/3?firstname=mary&lastname=hello&tit
                       le=mgr&birthdate=1999-10-20&hiredate=2020-04-30&address=中正四路
                       211號&city=高雄
                    </textarea>
                </div>
                <div class="form-group">
                    <input type="submit" value="送出" />
                </div>
            </form>
            <div id="msg1" class="message-box"></div>
            <script>
                var httpMethod, resourceUrl;
                //動態變更httpMethod,resourceUrl之值
                $("#p1 :text, #p1 textarea").on("focus input", function () {
                    httpMethod = $(this).attr("title");
                    resourceUrl = $(this).val() ;
                });
                $("#p1 :text:first").focus();
                $("#form1").submit(function () {
                    $.ajax({
                        method: httpMethod,
                        url: resourceUrl,
                        dataType:"json",
                        success: function (data) {
                            $("#msg1").html(JSON.stringify(data));
                            //查詢/api/employees/3後  data如: {"員工編號":"3","名字":"Janet",
```

```
                                    "姓氏":"Leverling","職稱":"Sales Representative",
                                    "生日":"1963-08-30","到職日":"1992-04-01","地址-街道":"722 Moss Bay
                                    Blvd.","地址-市鎮":"Kirkland"}
                                //新增/api/employees/後   data如: {"insert-status": "success"}
                                //修改/api/employees/3後  data如  {"update-status": "success"}
                            },
                            error: function (jqXHR, textStatus, errorThrown) {
                                $("#msg1").html(
                                    `<span class="color-red">textStatus=${textStatus};
                                    errorThrown=${errorThrown}</span>`);
                            }
                        });
                        return false;
                    });
                </script>
            </div>
        </div>
    </div>
</body>
</html>
```

2.2 CRUD 服務與 CORS 機制處理(伺服端)

2.2.1 「Node.js+Express.js」Web 應用程式開發入門

(請參考：9 附錄：「Node.js+Express.js」Web 應用程式入門)

2.2.2 CORS 機制特別說明

- 瀏覽器因為安全性的考量，一般會採取 CORS(跨來源資源共享/Cross-Origin Resource Sharing)政策
- CORS 政策簡單說就是同源政策(Same-origin policy)。
  - ✓ 即如果您目前的網頁去存取與該網頁不同網站(包括連接埠或 http/https 不同)的資源時，原則上您的瀏覽器會送出這個請求(Request)，但是會把對應的回應(Response)擋下來，不讓你的 JavaScript 存取並且傳回錯誤訊息。誤訊息如下：

    Access to XMLHttpRequest at 'http://localhost:1337/api/employees/3' from origin 'http://localhost:63175' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.

    GET http://localhost:1337/api/employees/3 net::ERR_FAILED 200 (jquery-3.6.0.min.js:2)
  - ✓ 如果您想開啟跨來源 HTTP 請求的話，網頁伺服器必須在回應(Response)裡加上 Access-Control-Allow-Origin 標頭(Header)。例如：

    Access-Control-Allow-Origin: *

    或

    Access-Control-Allow-Origin: http://localhost:63025
  - ✓ 當瀏覽器收到網頁伺服器的回應(Response)之後，會先檢查 Access-Control-Allow-Origin 標頭的內容，如果裡面包含現在這個發起請求的 Origin 的話，就會允許其 JavaScript 程式順利接收到此回應。
- 在應用了 Express.js 框架的 Web 應用程式中，可以使用 Express.js 官方自己推出的 cors 套件來設定 CORS 。CORS 套件的使用方法：
  - ✓ 安裝 cors 套件

```
npm install cors
```

✓ 在應用程式中匯入、使用 cors 套件

```
var cors = require('cors');
var express = require("express");
var app = express();
app.use( cors() );
//cors()回傳一個中介軟體函式，該函式會處理跨來源的 HTTP 請求，在回應的訊息中預設添加下列標頭：
//Access-Control-Allow-Origin: *
//Access-Control-Allow-Methods: GET, HEAD, PUT, PATCH, POST, DELETE
```

■ 另外瀏覽器在用戶端發送具特殊 HTTP 方法(即 GET、POST、 HEAD 以外的方法，例如 PUT 方法)的跨來源的請求時，會進行下列的特殊處理：

✓ 先送出一個 HTTP 方法為 OPTIONS 的請求(即預檢請求/Preflight Requset)給這個跨來源的伺服端。

✧ HTTP 請求訊息如下：

```
OPTIONS
/api/employees/3?firstname=mary&lastname=hello&title=mgr&birthdate=1999-10-20&hiredate=2020-04-30&(略) HTTP/1.1
Host: localhost:1337
Connection: keep-alive
Accept: */*
Access-Control-Request-Method: PUT
Origin: http://localhost:63175
(略)
```

✓ 然後等候並接收到伺服端的回應訊息後，檢視其內的「Access-Control-Allow-Methods」的標頭值，以確定這個跨來源的伺服端是否支援這個特殊的 HTTP 方法？若是，方正式送出該請求。

✧ HTTP 回應訊息如下：

```
HTTP/1.1 204 No Content
X-Powered-By: Express
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET,HEAD,PUT,PATCH,POST,DELETE
Content-Length: 0
Date: Sun, 07 Nov 2021 09:35:00 GMT
Connection: keep-alive
Keep-Alive: timeout=5
```

2.2.3 Server 範例程式 server.js

node_server > NodejsWebApp1 專案 > server.js

```
//1. 匯入Express套件
var express = require("express");
//2. Creates an Express application.
var app = express();
//4. 設定伺服器擬監聽的連線埠號
app.listen(1337, function () { console.log("Server listening on port 1337"); });
//5. 記得以UTF-5之字元編碼存檔
//-------------------------------------------------------------------------//
```

```
//6. 匯入cors套件以支援CORS政策
var cors = require('cors');
app.use(cors());
//cors()回傳一個中介軟體函式，該函式會處理跨來源的HTTP請求，在回應的訊息中預設添加下列標頭：
//Access-Control-Allow-Origin: *
//Access-Control-Allow-Methods: GET, HEAD, PUT, PATCH, POST, DELETE
//------------------------------------------------------------------------------//

//3. 設定路由(Routing)
//------------------------------------------------------------------------------//
app.get('/', function (req, res, next) {
    res.send('<h1 style="text-align:center;"> Express Web Server 運作中...</h1>');
    next();
});
//查詢所有員工
app.get('/api/employees/', function (req, res, next) {
    res.send(employees);
});
//查詢指定員工
app.get('/api/employees/:id', function (req, res, next) {
    var emp = employees.find(function (value, index) {
        return value.員工編號 == req.params.id;
    });
    //arr.find(callback) 方法會對每個元素執行一次 callback 函式，直到找到一個讓 callback 函式回傳 true 的元素。
    //當元素被找到的時候，find 會立刻回傳該元素，否則 find 會回傳 undefined。
    if (emp == undefined)
        emp = null;
    res.writeHead(200, { 'content-type': 'application/json;charset=utf-8' });
    res.end(JSON.stringify(emp));
});
//新增
app.post('/api/employees', function (req, res, next) {
    try {
        let emp = {
            "員工編號": req.query.employeeid,
            "名字": req.query.firstname,
            "姓氏": req.query.lastname,
            "職稱": req.query.title,
            "生日": req.query.birthdate,
            "到職日": req.query.hiredate,
            "地址-街道": req.query.address,
            "地址-市鎮": req.query.city
        }
        employees.push(emp);
        res.send({ "insert-status": "success" });
    }
    catch (ex) {
        res.send({ "insert-status": "fail" });
    }
```

```
});
//修改
app.put('/api/employees/:id', function (req, res, next) {
    try {
        var emp = employees.find(function (value, index) {
            return value.員工編號  == req.params.id;
        });
        if (emp) {
            emp["員工編號"] = req.params.id;
            emp["名字"] = req.query.firstname;
            emp["姓氏"] = req.query.lastname;
            emp["職稱"] = req.query.title;
            emp["生日"] = req.query.birthdate;
            emp["到職日"] = req.query.hiredate;
            emp["地址-街道"] = req.query.address;
            emp["地址-市鎮"] = req.query.city;
            res.send({ "update-status": "success" });
        }
        else
            res.send({ "update-status": "fail" });
    }
    catch (ex) {
        res.send({ "update-status": "fail" });
    }
});
//-------------------------------------------------------------------------------/
//定義資料來源
let employees =
    [
        {
            "員工編號": "1", "名字": "Nancy", "姓氏": "Davolio", "職稱": "Sales Representative",
            "生日": "1948-12-08", "到職日": "1992-05-01",
             "地址-街道": "507 - 20th Ave. E. Apt. 2A", "地址-市鎮": "Seattle"
        },
        {
            "員工編號": "2", "名字": "Andrew", "姓氏": "Fuller", "職稱": "Vice President, Sales",
            "生日": "1952-02-19", "到職日": "1992-08-14",
            "地址-街道": "908 W. Capital Way", "地址-市鎮": "Tacoma"
        },
        (略)
    ];
//-------------------------------------------------------------------------------/
```

3 使用 XMLHttpRequest 物件實作 AJAX 技術

3.1 AJAX 技術特別說明

- AJAX 即「Asynchronous JavaScript and XML」(非同步的 JavaScript 與 XML 技術)，指的是一套綜合了多項技術的瀏覽器端網頁開發技術。
- AJAX 應用程式可以僅向網頁伺服器傳送並取回必須的資料，並在瀏覽器端使用 JavaScript 處理來自伺服器的回應。
  - ✓ 因在網頁伺服器和瀏覽器之間交換的資料大量減少，網頁伺服器的回應就更快速了。
  - ✓ 同時很多處理工作可以在發出請求的瀏覽器端機器上完成，因此網頁伺服器的負荷也可以減少。
- 非同步請求相較於同步請求的優點
  - ✓ 瀏覽器發送同步請求(Synchronous Request)時，必須等候網頁伺服器的回應(Response)完成之後才能繼續網頁程式的執行，如此，瀏覽器視窗的畫面有時可能會出現凍結(freezing)現象，因而造成不及時響應(unresponsive)的使用者經驗。
  - ✓ 利用 JavaScript API 發送非同步請求(Asynchronous Request)則可解決上述問題。
  - ✓ 非同步應用程式的效能通常比同步應用程式的效能高。
- 非同步應用程式的開發也有一些問題必須面對，例如：
  - ✓ 非同步應用程式的開發相對比較複雜。
  - ✓ 非同步應用程式對於網頁伺服器回應資料的處理相對也比較複雜。

3.2 使用 XMLHttpRequest 發送同步的請求

- XMLHttpRequest 相關 API
  - ✓ XMLHttpRequest() 建構函式
    - ✧ The constructor initializes an XMLHttpRequest. It must be called before any other method calls.
  - ✓ XMLHttpRequest.open(method, url[, async]) 方法
    - ✧ The XMLHttpRequest method open() initializes a newly-created request, or re-initializes an existing one.
    - ✧ Parameters
      - ◆ method
        The HTTP request method to use, such as "GET", "POST", "PUT", "DELETE", etc.
      - ◆ url
        A DOMString representing the URL to send the request to.
      - ◆ async Optional
        - ● An optional Boolean parameter, defaulting to true, indicating whether or not to perform the operation asynchronously.
        - ● If this value is false, the send() method does not return until the response is received.
        - ● If true, notification of a completed transaction is provided using event listeners.
  - ✓ XMLHttpRequest.send(body) 方法
    - ✧ The XMLHttpRequest method send() sends the request to the server.
    - ✧ If the request is asynchronous (which is the default), this method returns as soon as the request is sent and the result is delivered using events.
    - ✧ If the request is synchronous, this method doesn't return until the response has arrived.
    - ✧ Parameters
      - ◆ body Optional
        A body of data to be sent in the XHR request.
        If no value is specified for the body, a default value of null is used.
    - ✧ Return value

- undefined.

✓ XMLHttpRequest.status 屬性
- ✧ Read only
- ✧ 回傳一個無符號短整數（unsigned short）表示已發送請求之回應的狀態。

✓ XMLHttpRequest.statusText 屬性
- ✧ The read-only XMLHttpRequest.statusText property returns a DOMString containing the response's status message as returned by the HTTP server.

✓ XMLHttpRequest.responseText 屬性
- ✧ The read-only XMLHttpRequest property responseText returns the text received from a server following a request being sent.
- ✧ Value
  - A DOMString which contains either the textual data received using the XMLHttpRequest or null if the request failed or "" if the request has not yet been sent by calling send().
- ✧ While handling an asynchronous request, the value of responseText always has the current content received from the server, even if it's incomplete because the data has not been completely received yet.
- ✧ You know the entire content has been received when the value of readyState becomes XMLHttpRequest.DONE (4), and status becomes 200 ("OK").

3.2.1 範例程式 T03-1.html (I)


```
<!DOCTYPE html>
<html>
<head>
    <title>使用XMLHttpRequest物件實作AJAX技術</title>
    <link rel="icon" href="./favicon.ico" />
    <link href="stylesheets/jquery-ui/base/jquery-ui.min.css" rel="stylesheet" />
    <link href="stylesheets/myStyleSheet.css" rel="stylesheet" />
    <style>
    </style>
```

```
        <script src="javascripts/jquery-3.6.0.min.js"></script>
        <script src="javascripts/jquery-ui.min.js"></script>
        <script type="text/javascript">
            $(function () {
                $("#tabs").tabs();
            });
        </script>
    </head>
    <body>
        <div id="tabs">
            <ul>
                <li><a href="#p1">使用XMLHttpRequest發送同步的請求</a></li>
                <li><a href="#p2">使用XMLHttpRequest發送非同步的GET請求</a></li>
                <li><a href="#p3">使用XMLHttpRequest發送非同步的POST請求</a></li>
            </ul>
            <div id="p1">
                <div class="main-box large-padding">
                    <p>
                        <a id="a1" href="data/Emp.json" class="a-button">
                                測試 XMLHttpRequest 同步GET...</a>
                    </p>
                    <div class="message-box color-444 height-60percent" id="div1"></div>
                    <script>
                        $("#a1").click(function () {
                            var xhr = new XMLHttpRequest();
                            xhr.open("get", "data/Emp.json", false); //false表示同步請求。
                            xhr.send(null);
                            if (xhr.status === 200) /Tthe HTTP status is 200(ok)
                                $("#div1").html(`來自伺服器的回應:<br/> ${xhr.responseText}`);
                            else
                                $("#div1").html(`<span class="color-red">
                                        來自伺服器的回應:${xhr.statusText}</span>`);
                            return false;
                        });
                    </script>
                </div>
            </div>
            <div id="p2">
                (略)
            </div>
            <div id="p3">
                (略)
            </div>
        </div>
    </body>
    </html>
```

3.2.2 範例 JSON 文件 data/Emp.json

```
[
  {
    "EmpId": "E001",
    "Name": "Hello",
    "Age": 18
  },
  {
    "EmpId": "E002",
    "Name": "Kitty",
    "Age": 28
  },
  {
    "EmpId": "E003",
    "Name": "Gorge",
    "Age": 38
  },
  {
    "EmpId": "E004",
    "Name": "Mary",
    "Age": 48
  }
]
```


3.3 使用 XMLHttpRequest 發送非同步的 GET 請求

- XMLHttpRequest 相關 API
  - ✓ XMLHttpRequest.onreadystatechange 屬性
    - ✧ An EventHandler that is called whenever the readyState attribute changes.
  - ✓ XMLHttpRequest.readyState 屬性
    - ✧ Returns an unsigned short, the state of the request.

Value	State	Description
0	UNSENT	Client has been created. open() not called yet.
1	OPENED	open() has been called.
2	HEADERS_RECEIVED	send() has been called, and headers and status are available.
3	LOADING	Downloading; responseText holds partial data.
4	DONE	The operation is complete.

  - ✓ XMLHttpRequest.onload = callback; 屬性 (新 API)
    - ✧ The XMLHttpRequestEventTarget.onload is the function called when an XMLHttpRequest transaction completes successfully.
    - ✧ Values
      - ◆ callback is the function to be executed when the request completes successfully.

3.3.1 範例程式 T03-1.html (II)


```
<!DOCTYPE html>
<html>
<head>
    <title>使用XMLHttpRequest物件實作AJAX技術</title>
    <link rel="icon" href="./favicon.ico" />
    <link href="stylesheets/jquery-ui/base/jquery-ui.min.css" rel="stylesheet" />
    <link href="stylesheets/myStyleSheet.css" rel="stylesheet" />
    <style>
    </style>
    <script src="javascripts/jquery-3.6.0.min.js"></script>
    <script src="javascripts/jquery-ui.min.js"></script>
    <script type="text/javascript">
        $(function () {
            $("#tabs").tabs();
        });
    </script>
</head>
<body>
    <div id="tabs">
        <ul>
            <li><a href="#p1">使用XMLHttpRequest發送同步的請求</a></li>
            <li><a href="#p2">使用XMLHttpRequest發送非同步的GET請求</a></li>
            <li><a href="#p3">使用XMLHttpRequest發送非同步的POST請求</a></li>
        </ul>
        <div id="p1">
            (略)
        </div>
```

```
            <div id="p2">
                <div class="main-box large-padding">
                    <p>
                        <a id="a2" href="data/Emp.json" class="a-button">
                                測試 XMLHttpRequest 非同步GET...</a>
                    </p>
                    <div class="message-box color-444 height-60percent" id="div2"></div>
                    <script>
                        $("#a2").click(function () {
                            var xhr = new XMLHttpRequest();
                            xhr.open("get", "data/Emp.json", true); //true表示非同步請求。
                            xhr. onreadystatechange = function () {
                                if (xhr.readyState === 4) { //The operation is complete.
                                    if (xhr.status === 200) {//The HTTP status is 200(ok)
                                        $("#div2").html(
                                            `來自伺服器的回應:<br/> ${xhr.responseText}`);
                                    } else {
                                        $("#div2").html(`<span class="color-red">
                                            來自伺服器的回應:${xhr.statusText}</span>`);
                                    }
                                }
                            };
                            xhr.send(null);
                            return false;
                        });
                    </script>
                </div>
            </div>
            <div id="p3">
                (略)
            </div>
        </div>
    </body>
    </html>
```

3.4 使用 XMLHttpRequest 發送非同步的 POST 請求.

- XMLHttpRequest 相關 API
  - ✓ XMLHttpRequest.onload = callback; 屬性 (取代 onreadystate 屬性)
    - ✧ The XMLHttpRequestEventTarget.onload is the function called when an XMLHttpRequest transaction completes successfully.
    - ✧ Values
      - ◆ callback is the function to be executed when the request completes successfully.
  - ✓ XMLHttpRequest.setRequestHeader(header, value) 屬性
    - ✧ 設定 HTTP 請求標頭（request header）值。
    - ✧ 回傳 undefined。
    - ✧ setRequestHeader() 可被呼叫的時間點必須於 open()之後、在 send()之前。
- FormData([form element])建構子函式
  - ✓ FormData 物件可為表單資料中的欄位/值建立相對應的的鍵/值對（key/value）集合，之後便可使用 XMLHttpRequest.send()方法來送出資料。

- ✓ FormData 物件與 Content-Type 請求標頭之間的關係：
  - ✧ 當 xhr.open("post",..)時，xhr.send(FormData 物件)會主動設定 Content-Type 標頭之值為 multipart/form-data
  - ✧ 當 xhr.open("get",..)時，xhr.send(FormData 物件)不會送出表單資料且不設定 Content-Type 標頭。

■ 表單資料(Form Data)進行 URL encoding (Percent-encoding)的規則

- ✓ The alphanumeric characters "a" through "z", "A" through "Z" and "0" through "9" remain the same.
- ✓ The special characters ".", "*", "_", and "-" remain the same.
- ✓ The space character " " is converted into a plus sign "+". (註: %20 亦可)
- ✓ All other characters are unsafe and are first converted into one or more bytes using some encoding scheme. Then each byte is represented by the 3-character string "%xy", where xy is the two-digit hexadecimal representation of the byte.
- ✓ 例如：

  <input type='text' name='username' value='小甜甜 A9.*_-' />

  經過 URL encoding (字元編碼：utf-8)之後產生之表單資料如下：

  username=%E5%B0%8F%E7%94%9C%E7%94%9C+A9.*_-

3.4.1 範例程式 T03-1.html (III)


```
<!DOCTYPE html>
<html>
<head>
    <title>使用XMLHttpRequest物件實作AJAX技術</title>
    <link rel="icon" href="./favicon.ico" />
    <link href="stylesheets/jquery-ui/base/jquery-ui.min.css" rel="stylesheet" />
    <link href="stylesheets/myStyleSheet.css" rel="stylesheet" />
    <style>
    </style>
```

```
    <script src="javascripts/jquery-3.6.0.min.js"></script>
    <script src="javascripts/jquery-ui.min.js"></script>
    <script type="text/javascript">
        $(function () {
            $("#tabs").tabs();
        });
    </script>
</head>
<body>
    <div id="tabs">
        <ul>
            <li><a href="#p1">使用XMLHttpRequest發送同步的請求</a></li>
            <li><a href="#p2">使用XMLHttpRequest發送非同步的GET請求</a></li>
            <li><a href="#p3">使用XMLHttpRequest發送非同步的POST請求</a></li>
        </ul>
        <div id="p1">
            (略)
        </div>
        <div id="p2">
            (略)
        </div>
        <div id="p3">
            <div class="main-box large-padding size-500x350 left-align">
                <div id="header">
                    <div id="logo">
                        <a href="#">
                            <img src="images/greefies/logo.gif" />
                        </a>
                    </div>
                </div>
                <form id="form1" class="main-form">
                    <div class="form-group">
                        <label for="username">使用者名稱</label>
                        <input class="form-control" type="text" name="username" value="小倩"
                            id="username" placeholder="請輸入使用者名稱" />
                    </div>
                    <div class="form-group">
                        <label for="password">密碼</label>
                        <input class="form-control" type="password" name="password"
                            value="采臣"    id="password" placeholder="請輸入密碼" />
                    </div>
                    <div class="form-group">
                        <input type="submit" value="送出" /> <div id="msg1"
                            class="message-box inline"></div>
                    </div>
                </form>
                <div id="footer">
```

```
                    <div>
                        <a href="#">
                            <img src="images/greefies/footer_logo.gif" />
                        </a>
                    </div>
                </div>
                <script>
                    $("#form1").submit(function () {
                        var xhr = new XMLHttpRequest();
                        xhr.open("post", "_AjaxServer01.aspx", true); //true表示非同步請求。
                        xhr.onload = function () {//The transaction completes successfully
                            if (xhr.status === 200) {
                                if (xhr.responseText == 'pass')
                                    $("#msg1").html(`<span>您已通過驗證!</span>`);
                                else
                                    $("#msg1").html(
                                        `<span class="color-red">您未通過驗證!</span>`);
                            } else {
                                $("#msg1").html(
                                        `<span class="color-red">${xhr.statusText}</span>`);
                            }
                        };
                        //準備表單資料
                        //方法一
                        xhr.setRequestHeader("Content-Type",
                           "application/x-www-form-urlencoded;charset=UTF-8") //預設
                                                    Content-Type: text/plain;charset=UTF-8
                        var formData = $(this).serialize(); //formData如:
                           username=%E5%B0%8F%E5%80%A9&password=%E9%87%87%E8%87%A3
                        //方法二
                        //var formData = new FormData(this); //此招勿設定Content-Type標頭
                        xhr.send(formData);
                        return false;
                    });
                </script>
            </div>
        </div>
    </div>
</body>
</html>
```

3.4.2 範例伺服端程式_AjaxServer01.aspx

```
<%@ Page Language="C#" ContentType="text/html" %>
<%
    if (Request["username"] == "小倩" && Request["password"] == "采臣")
        Response.Write("pass");
    else
        Response.Write("fail");
%>
```

4 XMLHttpRequest 進階應用

4.1 透過 XMLHttpRequest 物件接收 XML 格式的資料

- 透過 XMLHttpRequest 取得一個遠端的 XML 文件內容時，responseXML 屬性(property)將會是一個由 XML 文件解析而來的 Document 物件。
- DOM API
  - ✓ Node
    - ✧ Node is an interface from which various types of DOM API objects inherit, allowing those types to be treated similarly; for example, inheriting the same set of methods, or being testable in the same way.
    - ✧ nodeName 屬性
      - ◆ The nodeName read-only property returns the name of the current Node as a string.

Interface	nodeName value
Attr	The value of Attr.name
Comment	"#comment"
Document	"#document"
Element	The value of Element.tagName
Text	"#text"

- jQuery.find(selector) Returns: jQuery
  - ✓ Get the descendants of each element in the current set of matched elements, filtered by a selector, jQuery object, or element.
  - ✓ Parameters
    - ✧ selector：A string containing a selector expression to match elements against.
  - ✓ Given a jQuery object that represents a set of DOM elements, the .find() method allows us to search through the descendants of these elements in the DOM tree and construct a new jQuery object from the matching elements.
- jQuery.children([selector]) Returns: jQuery
  - ✓ Get the children of each element in the set of matched elements, optionally filtered by a selector.
  - ✓ Parameters
    - ✧ selector：A string containing a selector expression to match elements against.
  - ✓ Given a jQuery object that represents a set of DOM elements, the .children() method allows us to search through the children of these elements in the DOM tree and construct a new jQuery object from the matching elements.
  - ✓ The .children() method differs from .find() in that .children() only travels a single level down the DOM tree while .find() can traverse down multiple levels to select descendant elements (grandchildren, etc.) as well.
  - ✓ Note also that like most jQuery methods, .children() does not return text nodes; to get all children including text and comment nodes, use .contents().

4.1.1 範例程式 T04-1.html(I)


```
<!DOCTYPE html>
<html>
<head>
	<title>XMLHttpRequest進階應用</title>
	<link rel="icon" href="./favicon.ico" />
	<link href="stylesheets/jquery-ui/base/jquery-ui.min.css" rel="stylesheet" />
	<link href="stylesheets/myStyleSheet.css" rel="stylesheet" />
	<script src="javascripts/jquery-3.6.0.min.js"></script>
	<script src="javascripts/jquery-ui.min.js"></script>
	<script type="text/javascript">
		$(function () {
			$("#tabs").tabs();
		});
	</script>
</head>
<body>
	<div id="tabs">
		<ul>
			<li><a href="#p1">透過XMLHttpRequest物件接收XML格式的資料</a></li>
			<li><a href="#p2">透過XMLHttpRequest物件接收JSON格式的資料</a></li>
		</ul>
		<div id="p1">
			<div class="main-box large-padding">
				<p>
					<a id="a1" href="data/PurchaseOrder.xml" class="a-button">
						測試  XMLHttpRequest物件接收XML格式的資料...</a>
				</p>
				<div class="message-box color-444 height-60percent" id="div1"></div>
				<script>
					$("#a1").click(function () {
```

```
                var xhr = new XMLHttpRequest();
                xhr.open("get", $(this).attr("href"), true); //true表示非同步請求。
                xhr.onload = function () { //The transaction completes successfully
                    if (xhr.status === 200) {
                        var xmlDoc = xhr.responseXML;
                        var items = $(xmlDoc).find("Item");
                        //<Item PartNumber="872-AA">
                        //      <ProductName>Lawnmower</ProductName>
                        //      <Quantity>1</Quantity>
                        //      <USPrice>148.95</USPrice>
                        //      <Comment>Confirm this is electric</Comment>
                        //</Item>
                        //<Item PartNumber="926-AA">
                        //      <ProductName>Baby Monitor</ProductName>
                        //      <Quantity>2</Quantity>
                        //      <USPrice>39.98</USPrice>
                        //      <ShipDate>1999-05-21</ShipDate>
                        //</Item>
                        $.each(items, function (index, value) { //此處value為<Item>
                            $("#div1").append(
                            `<div>PartNumber: ${$(value).attr("PartNumber")}<div>`);
                            $.each($(value).children(), function (i, v) {//此處v為
                                                        <ProductName>或<Quantity>
                                                        或<USPrice>或<Comment>
                                $("#div1").append(
                                        `<div class="color-blue">
                                        ${v.nodeName}=${$(v).text()}</div>`)
                            });
                        });
                    } else {
                        $("#div1").html(
                                `<span class="color-red">
                                來自伺服器的回應:${xhr.statusText}</span>`);
                    }
                };
                xhr.send(null);

                return false;
            });
        </script>
    </div>
</div>
<div id="p2">
 (略)
</div>
</div>
</body>
</html>
```

4.1.2 範例 XML 文件 data/PurchaseOrder.xml

data/PurchaseOrder.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<PurchaseOrder PurchaseOrderNumber="99503" OrderDate="1999-10-20">
  <Address Type="Shipping">
    <Name>Ellen Adams</Name>
    <Street>123 Maple Street</Street>
    <City>Mill Valley</City>
    <State>CA</State>
    <Zip>10999</Zip>
    <Country>USA</Country>
  </Address>
  <Address Type="Billing">
    <Name>Tai Yee</Name>
    <Street>8 Oak Avenue</Street>
    <City>Old Town</City>
    <State>PA</State>
    <Zip>95819</Zip>
    <Country>USA</Country>
  </Address>
  <DeliveryNotes>Please leave packages in shed by driveway.</DeliveryNotes>
  <Items>
    <Item PartNumber="872-AA">
      <ProductName>Lawnmower</ProductName>
      <Quantity>1</Quantity>
      <USPrice>148.95</USPrice>
      <Comment>Confirm this is electric</Comment>
    </Item>
    <Item PartNumber="926-AA">
      <ProductName>Baby Monitor</ProductName>
      <Quantity>2</Quantity>
      <USPrice>39.98</USPrice>
      <ShipDate>1999-05-21</ShipDate>
    </Item>
  </Items>
</PurchaseOrder>
```

4.2 透過 XMLHttpRequest 物件接收 JSON 格式的資料

- JSON(JavaScript Object Notation)是一種輕量級的資料交換語言，以文字為基礎，透過 XMLHttpRequest 物件的.responseText 屬性可以接收 JSON 格式的資料。
- JavaScript API
  - ✓ JSON.parse(text)
    - ✧ JSON.parse()方法把會把一個 JSON 字串轉換成 JavaScript 的數值或是物件。
    - ✧ 參數 text：要解析成 JSON 的字串。
    - ✧ 回傳值：從給定的 JSON text 回傳對應的 Object。
    - ✧ Throws：如果解析的字串不是合法的 JSON 格式會丟出一個 SyntaxError 例外

4.2.1 範例程式 T04-1.html(II)


```
<!DOCTYPE html>
<html>
<head>
    <title>XMLHttpRequest進階應用</title>
    <link rel="icon" href="./favicon.ico" />
    <link href="stylesheets/jquery-ui/base/jquery-ui.min.css" rel="stylesheet" />
    <link href="stylesheets/myStyleSheet.css" rel="stylesheet" />
    <script src="javascripts/jquery-3.6.0.min.js"></script>
    <script src="javascripts/jquery-ui.min.js"></script>
    <script type="text/javascript">
        $(function () {
            $("#tabs").tabs();
        });
    </script>
</head>
<body>
    <div id="tabs">
        <ul>
            <li><a href="#p1">透過XMLHttpRequest物件接收XML格式的資料</a></li>
            <li><a href="#p2">透過XMLHttpRequest物件接收JSON格式的資料</a></li>
        </ul>
        <div id="p1">
            (略)
        </div>
        <div id="p2">
            <div class="main-box large-padding left-align">
                <p>
                    <a id="a2" href="data/Emp.json" class="a-button">
                        測試 XMLHttpRequest物件接收JSON格式的資料...</a>
```

```
                    </p>
                    <div class="message-box color-444 large-padding" id="div2"></div>
                    <script>
                        $("#a2").click(function () {
                            var xhr = new XMLHttpRequest();
                            xhr.open("get", $(this).attr("href"), true); //true表示非同步請求。
                            xhr.onload = function () {    //The transaction completes successfully
                                if (xhr.status === 200) {
                                    var text = xhr.responseText;
                                    var array = JSON.parse(text);
                                    var s = "";
                                    $.each(array, function (index, value) {
                                        s += `<div>員工編號: ${value.EmpId}
                                            姓名: ${value.Name}  年齡: ${value.Age}歲<div>`;
                                    });
                                    $("#div2").html(s);
                                } else {
                                    $("#div2").html(`<span class="color-red">
                                            來自伺服器的回應:${xhr.statusText}</span>`);
                                }
                            };
                            xhr.send(null);
                            return false;
                        });
                    </script>
                </div>
            </div>
        </div>
    </body>
    </html>
```

5 使用「Promise」執行非同步操作

5.1 比較「回呼函式」vs「Promise」應用於非同步操作的用法

5.1.1 回呼函式導論

- 回呼函式(callback function)是指能藉由參數(argument)傳遞到另一個函式的函式。它會在外部函式內(inside the outer function)被呼叫、以完成某些事情。
- 回呼函式常用來延續非同步操作(asynchronous operation)完成後的程式執行：這就叫做非同步回呼(asynchronous callbacks)。

5.1.2 ES6 Promise 導論

- Promise 是一個表示非同步操作(asynchronous operation)的最終完成或失敗的物件。
- 基本上，一個 Promise 是一個被回傳的物件(a returned object)，該物件之後會被附加 Callback 函式，以取代傳統的傳遞 Callback 到一個函式的作法。
- 舉例來說，下方的範例若用舊方式應該會有兩個 Callback，並根據成功或失敗來決定使用哪個：

```
function successCallback(result) {
  console.log("It succeeded with " + result);
}
function failureCallback(error) {
  console.log("It failed with " + error);
}
doSomething(successCallback, failureCallback);
```

- 而新作法會回傳一個 Promise，這樣你就可以附加 Callback：

```
let promise = doSomething();
promise.then(successCallback, failureCallback);
```

再簡單點：

```
doSomething().then(successCallback, failureCallback);
```

我們稱之為**非同步函式呼叫**(asynchronous function call)。

- 這個做法有許多好處，例如：
  - ✓ 用 .then() 附加的 Callback 一定會被呼叫(甚至在非同步運算結束後才附加的也一樣)。
  - ✓ 透過多次呼叫.then()的方式可以附加多個 Callback。
- 基本上，每個 Promise 代表著 Promise 鏈(Promise chain)中前一個非同步函式的完成。
- Promise 的實作
  - ✓ new Promise(/* executor */ function(resolve, reject) { ... });
    - ✧ 參數
      - ◆ executor 為一個依序接收兩個參數的函式：resolve 及 reject（實現及拒絕回呼函式）。
      - ◆ executor 函式會在 Promise 建構函式回傳 Promise 物件前被執行。
      - ◆ resolve 與 reject 函式會在被個別呼叫時，個別執行之。通常 executor 函式會發起一些非同步操作。接著，成功完成後執行 resolve 以完成 promise；或如果有錯誤，執行 rejects。
      - ◆ 如果 executor 函式在執行中拋出錯誤，promise 會被拒絕(rejected)，回傳值也將被忽略。
      - ◆ 一個 Promise 物件處於以下幾種狀態：
        A. 擱置（pending）：初始狀態，不是 fulfilled 與 rejected。
        B. 實現（fulfilled）：表示操作成功地完成。

C. 拒絕（rejected）：表示操作失敗了。

- ◆ 若一個 promise 已被實現或拒絕，繫結（attached）於它的處理函式將立即被呼叫。

✓ Promise.then()

✧ 回傳值

The then() method returns a Promise.

✧ 參數

It takes up to two arguments: callback functions for the success and failure cases of the Promise.

```
p.then(onFulfilled [, onRejected]);
p.then(value => { /* fulfillment */ }, reason => { /* rejection*/ });
```

- ◆ onFulfilled
  - ● A Function called if the Promise is fulfilled.
  - ● This function has one argument, the fulfillment value.
- ◆ onRejected
  - ● A Function called if the Promise is rejected.
  - ● This function has one argument, the rejection reason.

Return value

- ● Once a Promise is fulfilled or rejected, the respective handler function (onFulfilled or onRejected) will be called asynchronously.
- ● The behaviour of the handler function follows a specific set of rules. If a handler function:
  - A. returns a value, the promise returned by then gets resolved with the returned value as its value.
  - B. doesn't return anything, the promise returned by then gets resolved with an undefined value.
  - C. throws an error, the promise returned by then gets rejected with the thrown error as its value.
  - D. returns an already fulfilled promise, the promise returned by then gets fulfilled with that promise's value as its value.
  - E. returns an already rejected promise, the promise returned by then gets rejected with that promise's value as its value.

5.1.3 範例程式 T05-1.html (I)


```
<!DOCTYPE html>
<html>
<head>
    <title>使用「Promise」執行非同步操作</title>
    <link rel="icon" href="./favicon.ico" />
    <link href="stylesheets/jquery-ui/base/jquery-ui.min.css" rel="stylesheet" />
    <link href="stylesheets/myStyleSheet.css" rel="stylesheet" />
    <script src="javascripts/jquery-3.6.0.min.js"></script>
    <script src="javascripts/jquery-ui.min.js"></script>
    <script type="text/javascript">
        var jqDiv1, jqDiv2;
        $(function () {
            $("#tabs").tabs();
            jqDiv1 = $("#div1");
            jqDiv2 = $("#div2");
        });
        function getConnection(cb) {
            setTimeout(function () {
                var connection = {};
                jqDiv1.append("<div>1. connection已取得!</div>");
                cb(connection);
            }, 1000);
            jqDiv1.append("<div>1. connection取得中...</div>");
        }
        function executeQuery(connection, cb) {
            setTimeout(function () {
                var resultSet = '[ { "empid": 1, "name": "小倩" },{ "empid": 2, "name": "采臣" } ]';
                jqDiv1.append("<div>2. ResultSet已取得!</div>");
                cb(resultSet);
            }, 1000);
```

```
            jqDiv1.append("<div>2. ResultSet取得中...</div>");
        }
        function processResultSet(resultSet, cb) {
            setTimeout(function () {
                var rows = JSON.parse(resultSet);
                jqDiv1.append("<div>3. resultSet轉換處理已完成!</div>");
                cb(rows);
            }, 1000);
            jqDiv1.append("<div>3. resultSet轉換處理中...</div>");
        }
        //-----------------------------------------------------//
        function getConnection2() {
            return new Promise(function (resolve, reject) {
                setTimeout(function () {
                    var connection = {};
                    jqDiv2.append("<div>1. connection已取得!</div>");
                    resolve(connection);
                }, 1000);
                jqDiv2.append("<div>1. connection取得中...</div>");
            });
        }
        function executeQuery2(connection) {
            return new Promise(function (resolve, reject) {
                setTimeout(function () {
                    var resultSet = '[ { "empid": 1, "name": "米妮" },
                                      { "empid": 2, "name": "米奇" } ]';
                    jqDiv2.append("<div>2. ResultSet已取得!</div>");
                    resolve(resultSet);
                }, 1000);
                jqDiv2.append("<div>2. ResultSet取得中...</div>");
            });
        }
        function processResultSet2(resultSet) {
            return new Promise(function (resolve, reject) {
                setTimeout(function () {
                    var rows = JSON.parse(resultSet);
                    jqDiv2.append("<div>3. resultSet轉換處理已完成!</div>");
                    resolve(rows);
                }, 1000);
                jqDiv2.append("<div>3. resultSet轉換處理中...</div>");
            });
        }
    </script>
</head>
<body>
    <div id="tabs">
        <ul>
            <li><a href="#p1">比較「回呼函式」vs「Promise」應用於非同步操作的用法</a>
            </li>
```

```
        <li><a href="#p2">完整的「Promise」使用範例</a></li>
    </ul>
    <div id="p1">
        <div style="width:80%; margin:auto;">
            <div class="main-box size-500x350" style="display:inline-block; width:45%;">
                <p>
                    <a id="a1" href="#" class="a-button">
                        測試「回呼函式」應用於非同步操作...</a>
                </p>
                <div class="message-box height-60percent" id="div1"></div>
                <script>
                    $("#a1").click(function () {
                        getConnection(function (conn) {
                            executeQuery(conn, function (rs) {
                                processResultSet(rs, function (rows) {
                                    rows.forEach(function (value, index) {
                                        jqDiv1.append(
                                          `<div style="color:green;">
                                          empid=${value.empid} name=${value.name}
                                          </div>`);
                                    });
                                });
                            });
                        });
                        return false;
                    });
                </script>
            </div>
            <div class="main-box size-500x350" style="display: inline-block; width: 45%;">
                <p>
                    <a id="a2" href="#" class="a-button">
                        測試「Promise」應用於非同步操作...</a>
                </p>
                <div class="message-box height-60percent" id="div2"></div>
                <script>
                    $("#a2").click(function () {
                        getConnection2()
                            .then(function (conn) {
                                return executeQuery2(conn);
                            })
                            .then(function (rs) {
                                return processResultSet2(rs);
                            })
                            .then(function (rows) {
                                rows.forEach(function (value, index) {
                                    jqDiv2.append(`<div style="color:green;">
                                      empid=${value.empid} name=${value.name}</div>`);
                                });
                            });
```

```
                    return false;
                });
            </script>
        </div>
    </div>
</div>
<div id="p2">
    (略)
</div>
</div>
</body>
</html>
```

5.2 完整的「Promise」使用範例

5.2.1 範例程式 T05-1.html (II)


```
<!DOCTYPE html>
<html>
<head>
    <title>使用「Promise」執行非同步操作</title>
    <link rel="icon" href="./favicon.ico" />
    <link href="stylesheets/jquery-ui/base/jquery-ui.min.css" rel="stylesheet" />
    <link href="stylesheets/myStyleSheet.css" rel="stylesheet" />
    <script src="javascripts/jquery-3.6.0.min.js"></script>
    <script src="javascripts/jquery-ui.min.js"></script>
    <script type="text/javascript">
        var jqDiv1, jqDiv2;
        $(function () {
            $("#tabs").tabs();
            (略)
        });
```

```
            (略)
        </script>
</head>
<body>
        <div id="tabs">
            <ul>
                <li><a href="#p1">比較「回呼函式」vs「Promise」應用於非同步操作的用法</a>
                </li>
                <li><a href="#p2">完整的「Promise」使用範例</a></li>
            </ul>
            <div id="p1">
                (略)
            </div>
            <div id="p2">
                <div class="main-box">
                    <p>
                        <a id="a3" href="#" class="a-button">測試完整的「Promise」使用範例...</a>
                    </p>
                    <div class="message-box height-60percent" id="div3"></div>
                    <script>
                        function sendAjaxRequest(url, username, password) {
                            return new Promise(
                                function (resolve, reject) { //executor
                                    //非同步操作
                                    $.ajax({
                                        method: 'post',
                                        url: url,
                                        data: { username, password },
                                        success: function (data) {
                                            resolve(data);
                                        },
                                        error: function (jqXHR, textStatus, errorThrown) {
                                            reject(`${textStatus} - ${errorThrown}`);
                                        }
                                    });
                                });
                        }
                        //-----------------------------------------------------//
                        $("#a3").click(function () {
                            sendAjaxRequest('_AjaxServer02.aspx', "hello", "kitty")
                                .then(
                                    function (data) {
                                        try {
                                            //解析伺服端回應的JSON字串
                                            var obj = JSON.parse(data);
                                            return obj.status;
                                        }
                                        catch (e) {
                                            throw `伺服端回應的JSON字串格式有誤<br/>
```

```
                                        ${e.toString()}`;
                                }
                            },
                            function (err) {
                                $("#div3").append(`<div style="color:red">
                                                    錯誤狀況1: ${err}</div>`);
                            }
                        )
                        .then(
                            function (status) {
                                if (status == 'pass')
                                    $("#div3").append(`<div style="color:green">
                                                恭喜您！您已通過身分驗證。</div>`);
                                else if (status == 'fail')
                                    $("#div3").append(`<div style="color:blue">
                                                很抱歉！您未通過身分驗證。</div>`);
                                else
                                    throw `伺服端回應的status資料有誤
                                            (status=${status})`;
                            }
                        )
                        .catch(function (err) {
                            $("#div3").append(`<div style="color:red">
                                                    錯誤狀況2: ${err}</div>`);
                        }); //catch(failureCallback) 是 then(null, failureCallback) 的簡寫
                    return false;
                });
            </script>
        </div>
    </div>
</div>
</body>
</html>
```

5.2.2 範例伺服端程式_AjaxServer02.aspx

```
<%@ Page Language="C#" ContentType="text/html" %>
<%
    if (Request["username"] == "hello" && Request["password"] == "kitty")
        Response.Write("{\"status\": \"pass\"}");
    else
        Response.Write("{\"status\": \"fail\"}");
%>
```

6 Async function

6.1 ES7 Async function

- async function 的宣告用來定義一個非同步函式(asynchronous function)。
  - ✓ 非同步函式是一個 AsyncFunction 物件。
  - ✓ 當 async function 被呼叫時，它會回傳一個 Promise。
    - ✧ 如果該 async function 回傳了一個值，Promise 的狀態將為一個帶有該回傳值的 resolved。
    - ✧ 如果 async function 拋出例外或某個值，Promise 的狀態將為一個帶有被拋出值的 rejected。
  - ✓ async function 內部可以使用 await 表達式，它會暫停(pause)此 async function 的執行，並且等待傳遞至表達式的 Promise 的解析，解析完之後會回傳解析值，並繼續此 async function 的執行。
- async function 的語法

```
async function name([param[, param[, ... param]]]) {
   statements
}
```

  - ✓ 參數
    - ✧ name
      The function's name.
    - ✧ param
      The name of an argument to be passed to the function.
    - ✧ statements
      The statements comprising the body of the function.
  - ✓ 回傳值
    A Promise which will be resolved with the value returned by the async function, or rejected with an exception uncaught within the async function.

6.1.1 範例程式 T06-1.html (I)


```
<!DOCTYPE html>
<html>
```

```
<head>
    <title> Async function </title>
    <link rel="icon" href="./favicon.ico" />
    <link href="stylesheets/jquery-ui/base/jquery-ui.min.css" rel="stylesheet" />
    <link href="stylesheets/myStyleSheet.css" rel="stylesheet" />
    <script src="javascripts/jquery-3.6.0.min.js"></script>
    <script src="javascripts/jquery-ui.min.js"></script>
    <script type="text/javascript">
        $(function () {
            $("#tabs").tabs();
        });
    </script>
</head>
<body>
    <div id="tabs">
        <ul>
            <li><a href="#p1">Async function</a></li>
            <li><a href="#p2">Async function expression</a></li>
        </ul>
        <div id="p1">
            <div class="main-box large-padding">
                <p>
                    <a id="a1" href="#" class="a-button">測試Async Function...</a>
                </p>
                <div class="message-box height-60percent" id="div1"></div>
                <script>
                    function sendAjaxRequest(url, username, password) {
                        return new Promise(
                            function (resolve, reject) { //executor
                                //非同步操作
                                $.ajax({
                                    method: 'get',
                                    url: url,
                                    data: { username, password },
                                    success: function (data) {
                                        resolve(data);
                                    },
                                    error: function (jqXHR, textStatus, errorThrown) {
                                        reject(`${textStatus} - ${errorThrown}`);
                                    }
                                });
                            });
                    }

                    async function myAsyncFunc() {
                        try {
                            var result = await sendAjaxRequest("_AjaxServer02.aspx",
                                                               "hello", "kitty");
                            $("#div1").append(`<div>1.${result}</div>`);
                            result = await sendAjaxRequest("data/Emp.json");
```

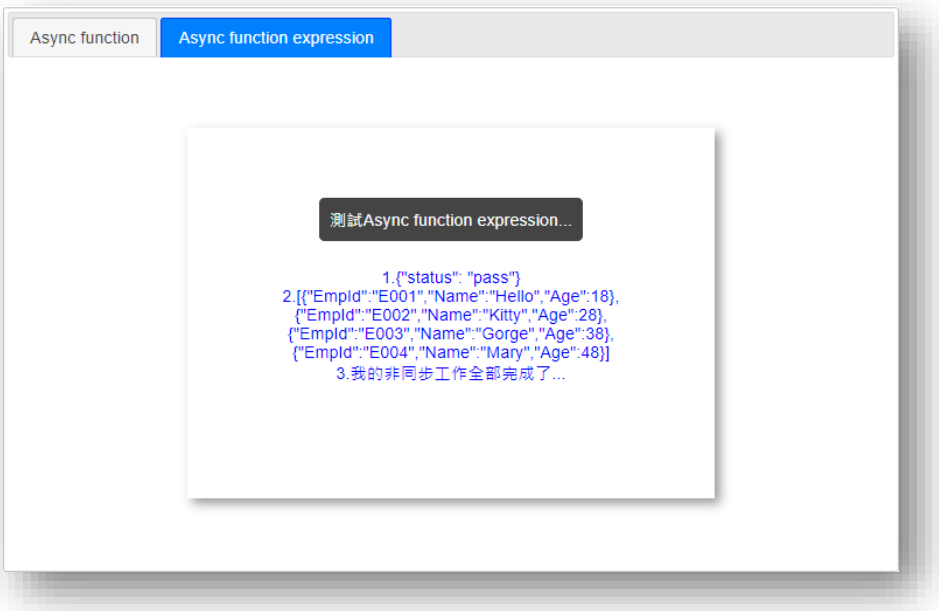
```
                $("#div1").append(`<div>2.${JSON.stringify(result)}</div>`);
            }
            catch (err) {
                $("#div1").append(`<div style="color:red">${err}</div>`);
            }
            return "3.我的非同步工作全部完成了...";
        }

        $("#a1").click(function () {
            myAsyncFunc()
                .then(function (data) {
                    $("#div1").append(`<div>${data}</div>`);
                });
            return false;
        });
    </script>
</div>
</div>
<div id="p2">
    (略)
</div>
</div>
</body>
</html>
```

6.2 ES7 Async function expression

- 你也可以使用 async function expression 來定義一個非同步函式。

6.2.1 範例程式 T06-1.html (II)




```
<!DOCTYPE html>
<html>
```

```
<head>
    <title>Async function</title>
    <link rel="icon" href="./favicon.ico" />
    <link href="stylesheets/jquery-ui/base/jquery-ui.min.css" rel="stylesheet" />
    <link href="stylesheets/myStyleSheet.css" rel="stylesheet" />
    <script src="javascripts/jquery-3.6.0.min.js"></script>
    <script src="javascripts/jquery-ui.min.js"></script>
    <script type="text/javascript">
        $(function () {
            $("#tabs").tabs();
        });
    </script>
</head>
<body>
    <div id="tabs">
        <ul>
            <li><a href="#p1">Async function</a></li>
            <li><a href="#p2">Async function expression</a></li>
        </ul>
        <div id="p1">
         (略)
        </div>
        <div id="p2">
            <div class="main-box large-padding">
                <p>
                    <a id="a2" href="#" class="a-button">測試Async function expression...</a>
                </p>
                <div class="message-box height-60percent" id="div2"></div>
                <script>
                    $("#a2").click(function () {
                        (async function () {
                            try {
                                var result = await sendAjaxRequest("_AjaxServer02.aspx",
                                                                   "hello", "kitty");
                                $("#div2").append(`<div>1.${result}</div>`);

                                result = await sendAjaxRequest("data/Emp.json");
                                $("#div2").append(`<div>2.${JSON.stringify(result)}</div>`);
                            }
                            catch (err) {
                                $("#div2").append(`<div style="color:red">${err}</div>`);
                            }
                            return "3.我的非同步工作全部完成了...";
                        })()
                            .then(function (data) {
                                $("#div2").append(`<div>${data}</div>`);
                            });
                        return false;
                    });
```

```
                </script>
            </div>
        </div>
    </div>
</body>
</html>
```

7 Fetch API

7.1 開始使用 fetch()

- Fetch API 提供了一個能獲取包含跨網路資源在內的資源介面。它有點像我們所熟悉的 XMLHttpRequest，但這個新的 API 提供了更強更彈性的功能，使開發者能夠用更簡潔的語法取得非同步資料。
- fetch(resource [, init])
  - ✓ The fetch() method starts the process of fetching a resource from the network, returning a promise which is fulfilled once the response is available.
  - ✓ Parameters
    - ✧ resource
      - ◆ This defines the resource that you wish to fetch. This can either be:
        - A USVString containing the direct URL of the resource you want to fetch.
        - A Request object.
          - ➢ USVString corresponds to the set of all possible sequences of unicode scalar values.
          - ➢ USVString maps to a String when returned in JavaScript
    - ✧ init Optional
      - ◆ An object containing any custom settings that you want to apply to the request. The possible options are:
        - method
          The request method, e.g., GET, POST.
        - headers
          Any headers you want to add to your request, contained within a Headers object or an object literal with ByteString values.
        - body
          Any body that you want to add to your request: this can be a Blob, BufferSource, FormData, URLSearchParams, USVString, or ReadableStream object.
          Note that a request using the GET or HEAD method cannot have a body.
  - ✓ Return value
    - ✧ A Promise that resolves to a Response object.
- Response
  - ✓ The Response interface of the Fetch API represents the response to a request.
  - ✓ Response implements Body, so it also has the following methods available to it:
    - ✧ Body.arrayBuffer()
      Takes a Response stream and reads it to completion. It returns a promise that resolves with an ArrayBuffer.
    - ✧ Body.blob()
      Takes a Response stream and reads it to completion. It returns a promise that resolves with a Blob.
    - ✧ Body.formData()
      Takes a Response stream and reads it to completion. It returns a promise that resolves with a FormData object.
    - ✧ Body.json()
      Takes a Response stream and reads it to completion. It returns a promise that resolves with the result of parsing the body text as JSON.
    - ✧ Body.text()
      Takes a Response stream and reads it to completion. It returns a promise that resolves with

a USVString (text).

7.1.1 範例程式 T07-1.html (I)


```
<!DOCTYPE html>
<html>
<head>
    <title> Fetch API</title>
    <link rel="icon" href="./favicon.ico" />
    <link href="stylesheets/jquery-ui/base/jquery-ui.min.css" rel="stylesheet" />
    <link href="stylesheets/myStyleSheet.css" rel="stylesheet" />
    <style>
    </style>
    <script src="javascripts/jquery-3.6.0.min.js"></script>
    <script src="javascripts/jquery-ui.min.js"></script>
    <script type="text/javascript">
        $(function () {
            $("#tabs").tabs();
        });
    </script>
</head>
<body>
    <div id="tabs">
        <ul>
            <li><a href="#p1">開始使用fetch()</a></li>
            <li><a href="#p2">使用fetch接收json格式資料</a></li>
            <li><a href="#p3">使用fetch上傳圖檔</a></li>
        </ul>
        <div id="p1">
            <div class="main-box large-padding">
                <form action="" method="get" id="form1">
                    <label>
                        姓名 <input type="text" id="text1" name="username" size="20" value="采臣" />
                    </label>
                    <input type="submit" value="送出" />
                </form>
```

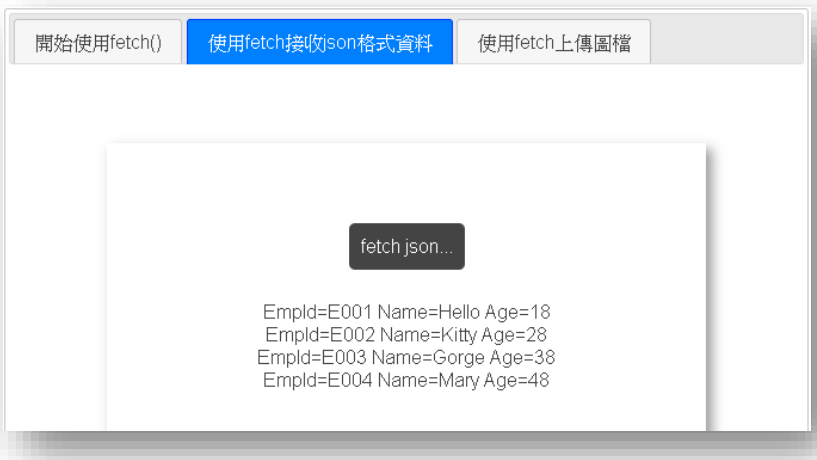
```
                    <div class="message-box" id="div1"></div>
                    <script>
                        $("#form1").submit(function () {
                            fetch('_AjaxServer03.aspx', {
                                method: 'POST', //POST HTTP method才可設定body;又POST必須大寫
                                body: $(this).serialize(),
                                 //$(this).serialize()之值如: username=%E9%87%87%E8%87%A3
                                 //亦可如body: 'username=采臣&password=123',
                                headers: {
                                    'Content-Type': 'application/x-www-form-urlencoded; charset=utf-8'
                                      //當使用者設定了body為query string時，Content-Type會自動被
                                      設為text/plain;charset=UTF-8
                                }
                            })
                                .then(function (response) {
                                    return response.text();
                                })
                                .then(function (text) {
                                    $("#div1").html(`伺服器已回應: ${text}`);
                                });
                            return false;
                        });
                    </script>
                </div>
            </div>
            <div id="p2">
                (略)
            </div>
            <div id="p3">
                (略)
                  </div>
            </div>
        </div>
    </body>
    </html>
```

7.1.2 範例伺服端程式_AjaxServer03.aspx

```
<%@ Page Language="C#" ContentType="text/html" %>
<%
    System.Threading.Thread.Sleep(1000);
    if(Request["username"] != null)
        Response.Write("歡迎" + Request["username"] + "光臨我們網站..." );
    else
        Response.Write("歡迎光臨我們網站..." );
%>
```

7.2 使用 fetch 接收 json 格式資料

7.2.1 範例程式 T07-1.html (II)




```
<!DOCTYPE html>
<html>
<head>
    <title> Fetch API</title>
    <link rel="icon" href="./favicon.ico" />
    <link href="stylesheets/jquery-ui/base/jquery-ui.min.css" rel="stylesheet" />
    <link href="stylesheets/myStyleSheet.css" rel="stylesheet" />
    <style>
    </style>
    <script src="javascripts/jquery-3.6.0.min.js"></script>
    <script src="javascripts/jquery-ui.min.js"></script>
    <script type="text/javascript">
        $(function () {
            $("#tabs").tabs();
        });
    </script>
</head>
<body>
    <div id="tabs">
        <ul>
            <li><a href="#p1">開始使用fetch()</a></li>
            <li><a href="#p2">使用fetch接收json格式資料</a></li>
            <li><a href="#p3">使用fetch上傳圖檔</a></li>
        </ul>
        <div id="p1">
            (略)
        </div>
        <div id="p2">
            <div class="main-box large-padding">
                <p>
                    <a id="a1" href="data/Emp.json" class="a-button">fetch json...</a>
                </p>
```

```
                    <div class="message-box color-444" id="div2"></div>
                    <script>
                        $("#a1").click(function () {
                            fetch($(this).attr("href"))
                                .then(function (response) {
                                    return response.json();
                                })
                                .then(function (array) {
                                    $.each(array, function (index, value) {
                                        $("#div2").append(`EmpId=${value.EmpId}
                                            Name=${value.Name} Age=${value.Age}<br/>`);
                                    });
                                });
                            return false;
                        });
                    </script>
                </div>
            </div>
            <div id="p3">
                (略)
            </div>
        </div>
    </body>
    </html>
```

7.3 使用 fetch 上傳圖檔

7.3.1 範例程式 T07-3.html (III)

```
<!DOCTYPE html>
<html>
<head>
    <title> Fetch API</title>
    <link rel="icon" href="./favicon.ico" />
    <link href="stylesheets/jquery-ui/base/jquery-ui.min.css" rel="stylesheet" />
    <link href="stylesheets/myStyleSheet.css" rel="stylesheet" />
    <style>
    </style>
    <script src="javascripts/jquery-3.6.0.min.js"></script>
    <script src="javascripts/jquery-ui.min.js"></script>
    <script type="text/javascript">
        $(function () {
            $("#tabs").tabs();
        });
    </script>
</head>
<body>
    <div id="tabs">
        <ul>
            <li><a href="#p1">開始使用fetch()</a></li>
            <li><a href="#p2">使用fetch接收json格式資料</a></li>
            <li><a href="#p3">使用fetch上傳圖檔</a></li>
        </ul>
        <div id="p1">
            (略)
        </div>
        <div id="p2">
            (略)
        </div>
        <div id="p3">
            <div class="main-box large-padding size-500x350 left-align">
                <div id="header">
                    <div id="logo">
                        <a href="#">
                            <img src="images/greefies/logo.gif" />
                        </a>
                    </div>
                </div>
                <form method="post" action="_UploadServer.aspx"
                        enctype="multipart/form-data" id="form2" class="main-form">
                    <div class="form-group">
                        <label for="username">使用者名稱</label>
                        <input class="form-control" type="text" name="username"
                            id="username" placeholder="請輸入使用者名稱" value="采臣" />
                    </div>
                    <div class="form-group">
                        <label for="upload">上傳照片 </label>
                        <input class="form-control upload" type="file"
```

```
                    name="upload" id="upload" accept="image/*" />
            <div class="preview" id="preview">
                <div class="table">
                    <div class="td"></div>
                </div>
            </div>
        </div>
        <div class="form-group">
            <input type="submit" value="送出" /> <div id="msg1"
                                                    class="message-box inline"></div>
        </div>
        <div id="msg1"></div>
    </form>
    <div id="footer">
        <div>
            <a href="#">
                <img src="images/greefies/footer_logo.gif" />
            </a>
        </div>
    </div>
    <script>
        //預覽上傳照片(時機：選檔變更事件)
        $("#upload").change(function () {
            $("#preview .table .td").html("");//先清除前次預覽的<img>
            previewImg(this.files);
        });
        function previewImg(files) {
            if (files.length == 0)
                return;
            var file = files[0];
            var fr = new FileReader();
            //註冊：選檔被讀取完成後之事件處理器
            fr.onload =
                function () {
                    var img = $("<img>").attr({ src: fr.result });
                    /*    fr.result: The file's contents.  內容如下:
                     ...
                    */
                    $("#preview .table .td").html(img);
                };
            fr.readAsDataURL(file);
        }
        //使用AJAX模式上傳檔案
        $("#form2").submit(function () {
            var formData = new FormData(this)
            fetch($(this).attr("action"), {
                method: 'POST',
                body: formData,
                headers: {
```

```
                        //勿設定: 'Content-Type': 'multipart/form-data'
                        //由fetch()  自動設定如: Content-Type: multipart/form-data;
                              boundary=----WebKitFormBoundaryFCv84fGlEKy9TkrT
                    }
                })
                    .then(function (response) {
                        return response.text();
                    })
                    .then(function (text) {
                        $("#msg1").html(text);
                        setTimeout('$("#msg1").html("");',3000);
                    })
                return false;
            });
        </script>
    </div>
</div>
</div>
</body>
</html>
```

7.3.2 範例伺服端程式_UploadServer.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" ValidateRequest="false" %>
<script runat="server">
    private string galleryPath = "Uploads/";
    private void Page_Load(System.Object sender, System.EventArgs e)
    {
        String username = Request["username"];
        username = username == null || username == "" ? "閣下" : Request["username"];
        HttpPostedFile sourceFile = Request.Files["upload"];
        String message;
        if (sourceFile != null)//Request含upload欄位
        {
            string fileName = System.IO.Path.GetFileName(sourceFile.FileName);
            if(fileName=="")
                message = "Oops:  您忘記上傳圖檔了...";
            else
            {
                sourceFile.SaveAs(Server.MapPath(galleryPath + fileName));
                message = "感謝" + username + ":  您的檔案(" + fileName + ")上傳成功...";
            }
            Response.Write(message);
        }
    }
</script>
```

8 jQuery 提供的 AJAX 方法

8.1 $.get() 與 $.post()

- jQuery.get(url [, data] [, success] [, dataType]) Returns: jqXHR
  - ✓ Load data from the server using a HTTP GET request.
  - ✓ Parameters
    - ✧ url
      - ◆ Type: String
      - ◆ A string containing the URL to which the request is sent.
    - ✧ data
      - ◆ Type: PlainObject or String
      - ◆ A plain object or string that is sent to the server with the request.
    - ✧ success
      - ◆ Type: Function(PlainObject data, String textStatus, jqXHR jqXHR)
      - ◆ A callback function that is executed if the request succeeds. Required if dataType is provided, but you can use null or jQuery.noop as a placeholder.
    - ✧ dataType
      - ◆ Type: String
      - ◆ The type of data expected from the server. Default: Intelligent Guess (xml, json, script, text, html).
- jQuery.get([settings])
  - ✓ Parameters
    - ✧ settings
      - ◆ Type: PlainObject
      - ◆ A set of key/value pairs that configure the AJAX request.
      - ◆ All properties except for url are optional.
- jQuery.post(url [, data] [, success] [, dataType]) Returns: jqXHR
  - ✓ Load data from the server using a HTTP POST request.
  - ✓ Parameters
    - ✧ url
      - ◆ Type: String
      - ◆ A string containing the URL to which the request is sent.
    - ✧ data
      - ◆ Type: PlainObject or String
      - ◆ A plain object or string that is sent to the server with the request.
    - ✧ success
      - ◆ Type: Function(PlainObject data, String textStatus, jqXHR jqXHR)
      - ◆ A callback function that is executed if the request succeeds. Required if dataType is provided, but can be null in that case.
    - ✧ dataType
      - ◆ Type: String
      - ◆ The type of data expected from the server. Default: Intelligent Guess (xml, json, script, text, html).
- jQuery.post([settings])
  - ✓ Parameters
    - ✧ settings
      - ◆ Type: PlainObject
      - ◆ A set of key/value pairs that configure the AJAX request. All properties except for url are optional.

8.1.1 範例程式 T08-1.html (I)


```
<!DOCTYPE html>
<html>
<head>
    <title>jQuery提供的AJAX方法</title>
    <link rel="icon" href="./favicon.ico" />
    <link href="stylesheets/jquery-ui/base/jquery-ui.min.css" rel="stylesheet" />
    <link href="stylesheets/myStyleSheet.css" rel="stylesheet" />
    <style>
        #p1 .form-wrapper {
            width: 80%;
            margin: 10px auto;
        }
        #p2 #headlines {
            height: 250px;
            overflow: auto;
            background-color: rgba(110,138,195, 0.1);
        }
        #p3 ul {
            width: 50%;
            margin: auto;
            text-align: left;
        }
    </style>
    <script src="javascripts/jquery-3.6.0.min.js"></script>
    <script src="javascripts/jquery-ui.min.js"></script>
    <script type="text/javascript">
        $(function () {
            $("#tabs").tabs();
        });
    </script>
```

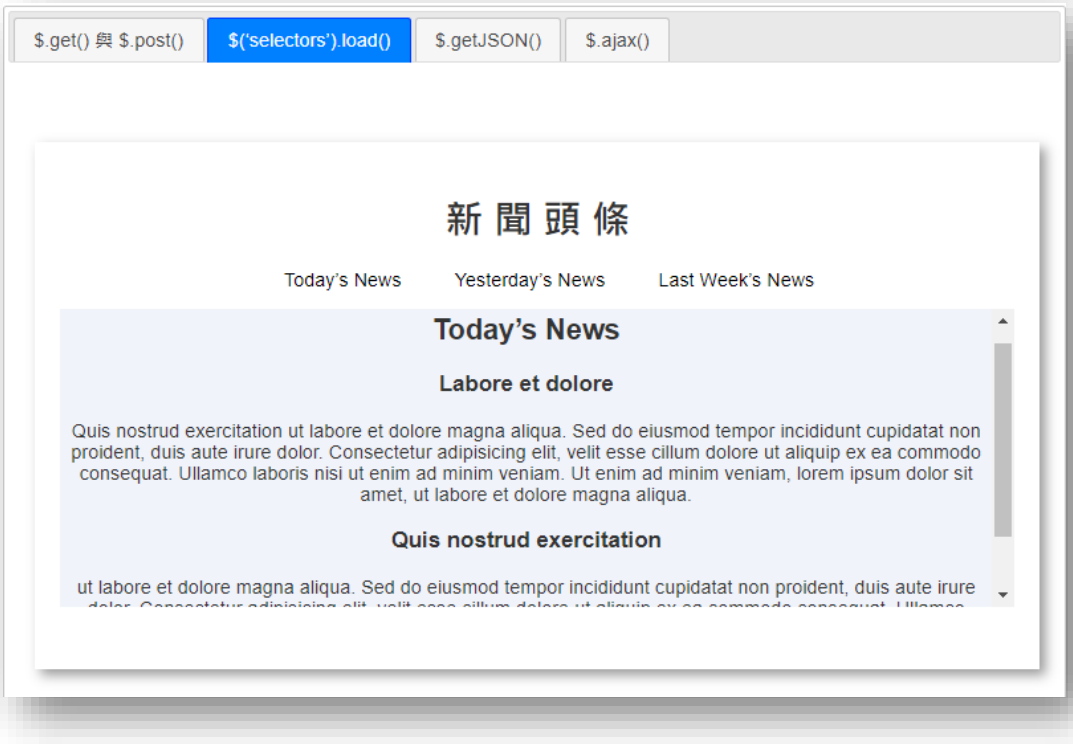
```
</head>
<body>
    <div id="tabs">
        <ul>
            <li><a href="#p1">$.get() 與 $.post()</a></li>
            <li><a href="#p2">$('selectors').load()</a></li>
            <li><a href="#p3">$.getJSON()</a></li>
            <li><a href="#p4">$.ajax()</a></li>
        </ul>
        <div id="p1">
            <div class="main-box">
                <h1 class="xx-large-font">Login</h1>
                <div class="form-wrapper">
                    <form method="get" action="_AjaxServer06-11.aspx"
                                                        id="form1" class="table">
                        <p class="table-row">
                            <label for="username" class="table-cell right-align">
                                使用者名稱</label>
                            <input type="text" name="username" id="username"
                                                class="table-cell" value="小倩" />
                        </p>
                        <p class="table-row">
                            <label for="password" class="table-cell right-align">密碼</label>
                            <input type="text" name="password" id="password"
                                                class="table-cell" value="采臣" />
                        </p>
                        <p class="table-caption">
                            <input type="submit" id="submit1" />
                        </p>
                    </form>
                    <div class="message-box" id="div1"></div>
                </div>
                <script>
                    $('#p1 #form1').submit(function () {
                        $.get('_AjaxServer01.aspx', $(this).serialize(),
                            function (data) {
                                if (data == 'pass') {
                                    $('#div1').html(
                                        '<span>恭喜您! 您已通過了身分驗證...</span>');
                                } else {
                                    $('#div1').html('<span class="color-red">
                                        您的登入資訊不正確，請再試一次...<span>');
                                }
                            }
                        );
                        return false;
                    });
                </script>
```

```
            </div>
        </div>
        <div id="p2">
            (略)
        </div>
        <div id="p3">
            (略)
        </div>
        <div id="p4">
            (略)
        </div>
    </div>
</body>
</html>
```

8.2 $('selectors').load()

- .load(url [, data] [, complete]) Returns: jQuery
  - ✓ Load data from the server and place the returned HTML into the matched elements.
  - ✓ Parameters
    - ✧ url
      - ◆ Type: String
      - ◆ A string containing the URL to which the request is sent.
    - ✧ data
      - ◆ Type: PlainObject or String
      - ◆ A plain object or string that is sent to the server with the request.
      - ◆ The POST method is used if data is provided as an object; otherwise, GET is assumed.
    - ✧ complete
      - ◆ Type: Function(String responseText, String textStatus, jqXHR jqXHR)
      - ◆ A callback function that is executed when the request completes.

8.2.1 範例程式 T08-1.html (II)



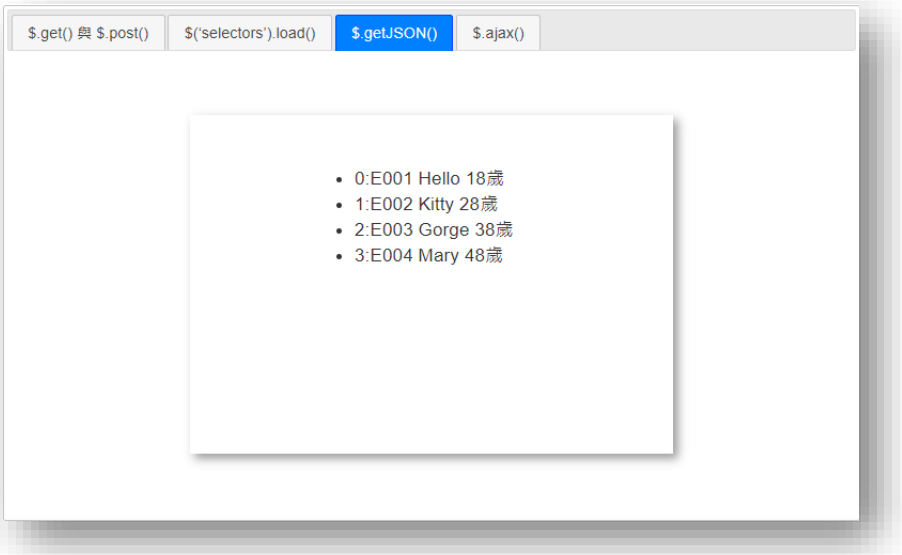

```
<!DOCTYPE html>
<html>
(略)
<body>
    <div id="tabs">
        <ul>
            <li><a href="#p1">$.get() 與 $.post()</a></li>
            <li><a href="#p2">$('selectors').load()</a></li>
            <li><a href="#p3">$.getJSON()</a></li>
            <li><a href="#p4">$.ajax()</a></li>
        </ul>
        <div id="p1">
            (略)
        </div>
        <div id="p2">
            <div class="main-box size-800x400">
                <h1 class="xx-large-font">新 聞 頭 條</h1>
                <ul id="newslinks">
                    <li><a href="data/today.html">Today&#8217;s News</a></li>
                    <li><a href="data/yesterday.html">Yesterday&#8217;s News</a></li>
                    <li><a href="data/lastweek.html">Last Week&#8217;s News</a></li>
                </ul>
                <div id="headlines"></div>
                <script>
                    $('#p2 #newslinks a').click(function () {
                        var url = $(this).attr('href');
                        $('#headlines').load(url + ' #newsItem');
```

```
                    return false;
                });
            </script>
        </div>
    </div>
    <div id="p3">
        (略)
    </div>
    <div id="p4">
        (略)
    </div>
</div>
</body>
</html>
```

8.3 $.getJSON()

- jQuery.getJSON(url [, data] [, success]) Returns: jqXHR
  - ✓ Load JSON-encoded data from the server using a GET HTTP request.
  - ✓ Parameters
    - ✧ url
      - ◆ Type: String
      - ◆ A string containing the URL to which the request is sent.
    - ✧ data
      - ◆ Type: PlainObject or String
      - ◆ A plain object or string that is sent to the server with the request.
    - ✧ success
      - ◆ Type: Function(PlainObject data, String textStatus, jqXHR jqXHR)
        - ● The success callback is passed the returned data, which is typically a JavaScript object or array as defined by the JSON structure and parsed using the $.parseJSON()
      - ◆ A callback function that is executed if the request succeeds.

8.3.1 範例程式 T08-1.html (III)



```
<!DOCTYPE html>
<html>
  (略)
<body>
    <div id="tabs">
        <ul>
            <li><a href="#p1">$.get() 與 $.post()</a></li>
            <li><a href="#p2">$('selectors').load()</a></li>
            <li><a href="#p3">$.getJSON()</a></li>
            <li><a href="#p4">$.ajax()</a></li>
        </ul>
        <div id="p1">
            (略)
        </div>
        <div id="p2">
            (略)
        </div>
        <div id="p3">
            <div class="main-box large-font large-padding">
                <ul></ul>
                <script>
                    $.getJSON("data/Emp.json", function (data) {
                        $.each(data, function (index, value) {
                            $("#p3 ul").append(
                                `<li>${index}:${value.EmpId} ${value.Name} ${value.Age}歲</li>`);
                        });
                    });
                </script>
            </div>
        </div>
        <div id="p4">
            (略)
        </div>
    </div>
</body>
</html>
```

8.4 $.ajax()

- jQuery.ajax([settings])
  - ✓ Perform an asynchronous HTTP (AJAX) request.
  - ✓ Returns jqXHR.
    - ✧ The jQuery XMLHttpRequest (jqXHR) object returned by $.ajax() as of jQuery 1.5 is a superset of the browser's native XMLHttpRequest object.
  - ✓ settings
    - ✧ Type: PlainObject
    - ✧ A set of key/value pairs that configure the AJAX request.
    - ✧ All settings are optional. A default can be set for any option with $.ajaxSetup().
    - ✧ 常用屬性如下：
      - ◆ method (default: 'GET')

- Type: String
- The HTTP method/verb to use for the request (e.g. "POST", "GET", "PUT").

◆ type (default: 'GET')
- Type: String
- An alias for method. You should use type if you're using versions of jQuery prior to 1.9.0.

◆ url (default: The current page)
- Type: String
- A string containing the URL to which the request is sent.

◆ data
- Type: PlainObject or String
- Data to be sent to the server.
  - It is converted to a query string, if not already a string.
  - It's appended to the url for GET-requests.
  - Object must be Key/Value pairs.

◆ processData (default: true)
- Type: Boolean
- By default, data passed in to the data option as an object (technically, anything other than a string) will be processed and transformed into a query string, fitting to the default content-type "application/x-www-form-urlencoded".
- If you want to send a DOMDocument, or other non-processed data (例如 new FormData()), set this option to false.

◆ contentType (default: 'application/x-www-form-urlencoded; charset=UTF-8')
- Type: Boolean or String
- When sending data to the server, use this content type.
- If you explicitly pass in a content-type to $.ajax(), then it is always sent to the server (even if no data is sent).
- As of jQuery 1.6 you can pass false to tell jQuery to not set any content type header.

◆ dataType (default: Intelligent Guess (xml, json, script, or html))
- Type: String
- The type of data that you're expecting back from the server.
- If none is specified, jQuery will try to infer it based on the MIME type of the response (an XML MIME type will yield XML, JSON will yield a JavaScript object, script will execute the script, and anything else will be returned as a string).
- The available types are:
  - "xml": Returns a XML document that can be processed via jQuery.
  - "html": Returns HTML as plain text; included script tags are evaluated when inserted in the DOM.
  - "script": Evaluates the response as JavaScript and returns it as plain text.
  - "json": Evaluates the response as JSON and returns a JavaScript object.
  - "text": A plain text string.

◆ async (default: true)
- Type: Boolean
- By default, all requests are sent asynchronously. If you need synchronous requests, set this option to false.

◆ beforeSend
- Type: Function(jqXHR jqXHR, PlainObject settings)

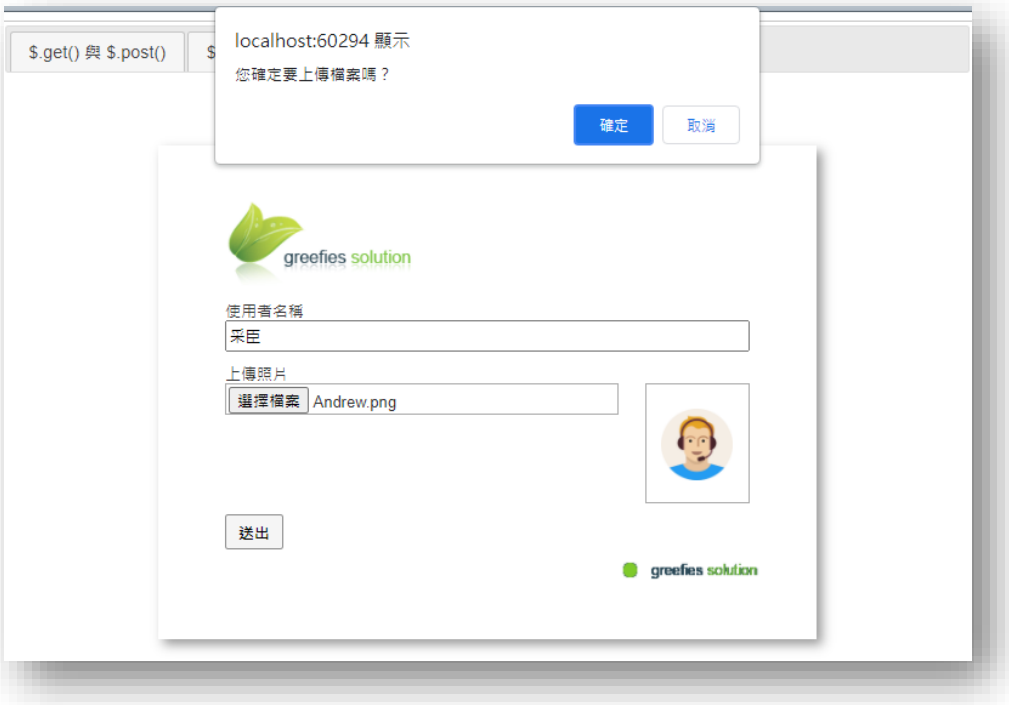
- A pre-request callback function that can be used to modify the jqXHR object before it is sent. Use this to set custom headers, etc.
- The jqXHR and settings objects are passed as arguments.
- This is an AJAX Event.
- Returning false in the beforeSend function will cancel the request.

◆ success
- Type: Function(Anything data, String textStatus, jqXHR jqXHR)
- A function to be called if the request succeeds.
- The function gets passed three arguments:
  - The data returned from the server, formatted according to the dataType parameter or the dataFilter callback function, if specified;
  - a string describing the status; and
  - the jqXHR object.
- This is an AJAX Event.

◆ error
- Type: Function(jqXHR jqXHR, String textStatus, String errorThrown)
- A function to be called if the request fails.
- The function receives three arguments:
  - The jqXHR object,
  - a string describing the type of error that occurred. (Possible values for the second argument (besides null) are "timeout", "error", "abort", and "parsererror") and
  - an optional exception object, if one occurred. (When an HTTP error occurs, errorThrown receives the textual portion of the HTTP status, such as "Not Found" or "Internal Server Error.")
- This is an AJAX Event..

◆ complete
- Type: Function(jqXHR jqXHR, String textStatus)
- A function to be called when the request finishes (after success and error callbacks are executed).
- The function gets passed two arguments:
  - The jqXHR object and
  - a string categorizing the status of the request ("success", "notmodified", "nocontent", "error", "timeout", "abort", or "parsererror").
- This is an AJAX Event.

8.4.1 範例程式 T08-1.html(IV)




```
<!DOCTYPE html>
<html>
  (略)
<body>
    <div id="tabs">
        <ul>
            <li><a href="#p1">$.get() 與 $.post()</a></li>
            <li><a href="#p2">$('selectors').load()</a></li>
            <li><a href="#p3">$.getJSON()</a></li>
            <li><a href="#p4">$.ajax()</a></li>
        </ul>
        <div id="p1">
            (略)
        </div>
        <div id="p2">
            (略)
        </div>
        <div id="p3">
            (略)
        </div>
        <div id="p4">
            <div class="main-box large-padding size-500x350 left-align">
                <div id="header">
                    <div id="logo">
                        <a href="#">
                            <img src="images/greefies/logo.gif" />
                        </a>
                    </div>
                </div>
```

```
<form method="post" action="_UploadServer.aspx" enctype="multipart/form-data"
        id="form2" class="main-form">
    <div class="form-group">
        <label for="username">使用者名稱</label>
        <input class="form-control" type="text" name="username"
                id="username" placeholder="請輸入使用者名稱" value="采臣" />
    </div>
    <div class="form-group">
        <label for="upload">上傳照片 </label>
        <input class="form-control upload" type="file"
                name="upload" id="upload" accept="image/*" />
        <div class="preview" id="preview">
            <div class="table">
                <div class="td"></div>
            </div>
        </div>
    </div>
    <div class="form-group">
        <input type="submit" value="送出" /> <div id="msg1"
                                        class="message-box inline"></div>
    </div>
</form>
<div id="footer">
    <div>
        <a href="#">
            <img src="images/greefies/footer_logo.gif" />
        </a>
    </div>
</div>
<script>
    //預覽上傳照片(時機：選檔變更事件)
    $("#upload").change(function () {
        $("#preview .table .td").html("");//先清除前次預覽的<img>
        previewImg(this.files);
    });
    function previewImg(files) {
        if (files.length == 0)
            return;
        var file = files[0];
        var fr = new FileReader();
        //註冊：選檔被讀取完成後之事件處理器
        fr.onload =
            function () {
                var img = $("<img>").attr({ src: fr.result });
                /*    fr.result: The file's contents.  內容如下:
                 ...
                */
                $("#preview .table .td").html(img);
            };
```

```
                fr.readAsDataURL(file);
            }
            //使用Ajax模式上傳檔案
            $("#form2").submit(function () {
                var formData = new FormData(this) //只適用於post方法
                $.ajax({
                    //async: true,
                    method: 'post',
                    url: $(this).attr('action'),
                    data: formData,
                    //contentType:  預設'application/x-www-form-urlencoded; charset=UTF-8',
                    contentType: false, //required for「 formData = new FormData(this) 」
                    processData: false, //required for「 formData = new FormData(this) 」
                    beforeSend: function () {
                        var ans = confirm("您確定要上傳檔案嗎？");
                        if (ans)
                            return true;
                        else
                            return false;
                    },
                    success: function (data) {
                        $("#msg1").html(data);
                    },
                    error: function (jqXHR, textStatus, errorThrown) {
                        $('#msg1').html(`<span class="color-red">
                       textStatus=${textStatus} errorThrown=${errorThrown}<span>`);
                    },
                    complete: function () {
                        setTimeout("$('#msg1').html('')", 2000);
                    }
                });
                return false;
            });
        </script>
    </div>
</div>
</div>
</body>
</html>
```

9 附錄：「Node.js+Express.js」Web 應用程式開發入門

9.1 Node.js 簡介

- Node.js 是一個功能強大的 JavaScript-based 框架(framework)
- Node.js 是以 Google Chrome 的 JavaScript V8 Engine 為基礎所開發出來的 JavaScript 程式執行環境(Runtime Environment)
- Node.js 很適合用來開發 Web 應用程式，採用的知名企業如：Netflix、Uber、PayPal、eBay、LinkedIn、NASA、Mozilla、Walmart 等

9.2 Node.js 應用程式開發「先要軟體」的安裝與測試

9.2.1 安裝 Node.js Runtime 軟體

- 下載網址: https://nodejs.org/zh-tw/download/
- 下載檔案: node-v##.#.#-x64.msi (Windows Installer / LTS (Long Term Support) version / 64-bit) 並執行之
- Node.js 軟體預設安裝路徑： C:\Program Files\nodejs
- Node.js 安裝狀況測試：
  - ✓ 開啟「命令提示字元」
  - ✓ 執行 node -v 或 node --version
    - ✧ 若正常顯示 Node.js 之版本(例如 v16.15.1)，即表示成功安裝了 Node.js 軟體
    - ✧ 若非，即表示 Node.js 軟體未成功安裝

9.2.2 測試 Node.js 應用程式的執行

- 開啟「命令提示字元」
- 變更工作目錄至 JavaScript 檔所在目錄，例如：

```
cd C:\_restful\_misc
```

- 使用 node.exe 執行 Node.js 應用程式，例如:

```
node app.js
```

app.js

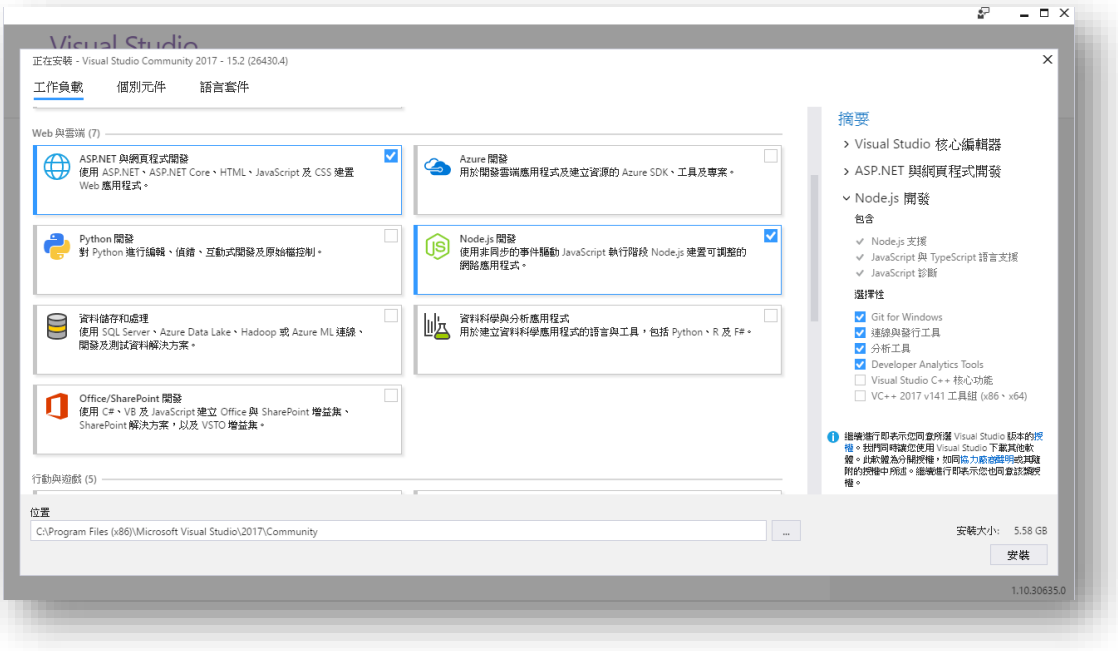
```
var greeting = "Hello World...";
console.log(greeting);
```

- 執行後所得之輸出結果如下:

```
C:\_restful\_misc>node app.js
Hello World...
```

9.3 使用 Visual Studio IDE 進行 Node.js Web 應用程式的開發

9.3.1 安裝「Node.js 開發」工作負載



9.3.2 建立一個「Node.js Web 應用程式」專案

9.3.2.1 建立專案的基本步驟

- 啟動「方案總管」
  - ✓ 在「解決方案」按下滑鼠右鍵
  - ✓ 點擊「加入」 > 「新增專案」
  - ✓ 點選「空白的 Node.js Web 應用程式」
- Visual Studio 會自動建立一個 JavaScript 檔案(server.js)

```
'use strict';
var http = require('http');
var port = process.env.PORT || 1337;

http.createServer(function (req, res) {
    res.writeHead(200, { 'Content-Type': 'text/plain' });
    res.end('Hello World\n');
}).listen(port);
```

9.3.2.2 Node.js 模組與套件的安裝與匯入

- Node.js 模組(Modules)相當於一個 JavaScript 的函式庫(Function Libraries)；套件則是一個或多個模組的群組。
- 一個 Node.js 模組對應一個.js 檔案；每個模組會有自己的執行環境(Context)。
- Node.js 包含三種模組
  - ✓ 核心模組(Core Modules)
    - ◆ Node.js 安裝時會內建「核心模組」(例如：http)，我們在開發應用程式時可以直接載入而不需要另外安裝。
    - ◆ 應用程式匯入「核心模組」的方式： require('http');
  - ✓ 協力廠商模組(Third Party Modules)

- ◆ 「協力廠商模組」一般使用「Node 套件管理員」(Node Package Manager/NPM) CLI(Command Line Interface)工具來下載安裝，例如：

```
cd C:\_restful\_misc
npm install express
```

- ◆ NPM 會將模組安裝在**當前工作目錄**的「node_modules」子目錄裡，並且自動在**當前工作目錄**中產生一個 package.json 檔案：

```
{
  "dependencies": {
    "express": "^4.17.1"
  }
}
```

- ◆ 應用程式匯入「協力廠商模組」的方式，例如： require('express');

✓ 本地模組(Local Modules)

- ◆ 「本地模組」是指儲存於本地端檔案系統的應用程式自訂模組。
- ◆ 應用程式匯入「本地模組」的方式，例如： require('./my-module.js');

9.3.2.3 Node.js 常用的 API 簡介

■ 'use strict'

✓ 意義與用途:

- ◆ Defines that JavaScript code should be executed in "strict mode"(嚴格模式).
- ◆ 「嚴格模式」要求程式碼需講求安全性

✓ 對程式的影響，例如：

- ◆ 變數一定要經過 var 或 let 宣告

```
"use strict";
x = 3.14;                // This will cause an error
```

- ◆ 函數的參數名稱不可重複

```
"use strict";
function x(p1, p1) {};   // This will cause an error
```

- ◆ 變數名稱不可使用 arguments

```
"use strict";
var arguments = 3.14;    // This will cause an error
```

■ process 物件

✓ 導論

- ◆ The process object is a global that provides information about, and control over, the current Node.js process.
- ◆ As a global, it is always available to Node.js applications without using require().

- ✓ process.env 屬性
  - ◆ The process.env property returns an object containing the user environment.
  - ◆ 在 Windows 設定環境變數後隨即啟動 Node.js 應用程式的方法如下：
    - ✧ *開啟「命令提示字元」視窗*
    - ✧ *執行指令：set Port=1337 & node server.js*

■ http.createServer([requestListener])方法
- ✓ 參數：requestListener <Function(request <http.IncomingMessage>, response <http.ServerResponse>)>
- ✓ 回傳：<http.Server>
- ✓ 說明：
  - ◆ Returns a new instance of http.Server.
  - ◆ The requestListener is a function which is automatically added to the 'request' event.

■ response.writeHead(statusCode[, statusMessage][, headers])
- ✓ 參數：
  - ◆ statusCode <number> (例如 200 或 404)
  - ◆ statusMessage <string> (例如 OK 或 Not Found)
  - ◆ headers <Object> (例如 { 'Content-Type': 'text/plain'})
- ✓ 說明：
  - ◆ Sends a response header to the request.
  - ◆ The status code is a 3-digit HTTP status code, like 404.
  - ◆ The last argument, headers, are the response headers.

■ response.write(chunk)
- ✓ 參數：
  - ◆ chunk <string>
- ✓ 回傳: <boolean>
- ✓ 說明:
  - ◆ This sends a chunk of the response body. This method may be called multiple times to provide successive parts of the body.
  - ◆ Returns true if the entire data was flushed successfully to the kernel buffer. Returns false if all or part of the data was queued in user memory. 'drain' will be emitted when the buffer is free again

■ response.end([data])
- ✓ 參數:
  - ◆ data <string>
- ✓ 說明:
  - ◆ This method signals to the server that all of the response headers and body have been sent; that server should consider this message complete. The method, response.end(), MUST be called on each response.
  - ◆ If data is specified, it is equivalent to calling response.write(data) followed by response.end().

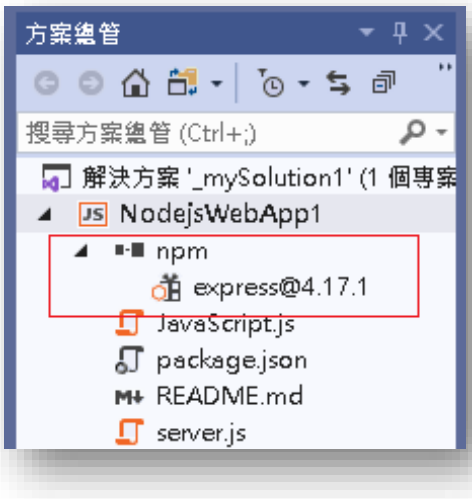
■ <http.server>.listen([port] [, callback]) 方法
- ✓ 參數:
  - ◆ port <number>
  - ◆ callback <Function>
- ✓ 回傳: <net.Server>
- ✓ 說明:
  - ◆ Starts the HTTP server listening for connections on the given port.
  - ◆ This function is asynchronous.
  - ◆ When the server starts listening, the 'listening' event will be emitted. The last

parameter callback will be added as a listener for the 'listening' event.

9.3.3 使用「Express.js Web Framework」快速建置 Web 應用程式

9.3.3.1 安裝 Express 套件

- 啟動「方案總管」視窗
- 在目標專案之「npm」項目上按下滑鼠右鍵
- 點擊「安裝新的 npm 套件」選項
  - ✓ 在「搜尋套件」輸入欄裡輸入: express
  - ✓ 點選「express #.#.# (最新版本) 」項目，然後點擊「安裝套件」按鈕



9.3.3.2 加入並編輯主程式

test.js

```
//1. 匯入express套件
var express = require("express");
//2. Creates an Express application.
var app = express();
//3. 設定路由(Routing)
app.get("/", function (req, res) {
      res.send("Hello World 哈囉世界...");
});
//app.get(path, callback)：
//設定「“GET”HTTP方法 +“根”路由路徑」的請求由指定的callback函式來處理。
//Express稱上述callback函式為「中介軟體函式(Middleware Functions)」。

//4. 設定伺服器擬監聽的連線埠號
app.listen(1337, function () { console.log("Server listening on port 1337"); });
//5. 記得以 UTF-5 之字元編碼存檔
```

9.3.3.3 啟動應用程式

- 在「方案總管」以滑鼠右鍵將主程式「設定為 Node.js 啟動檔案」。
- 按下「網頁伺服器(Google Chrome)」按鈕(F5)。
  - ✓ 主程式(Web Server)被啟動。
  - ✓ 瀏覽器跟著被啟動並發送 HTTP 請求。
  - ✓ 瀏覽器等候伺服端的回應，並在接收到回應後將之處理。處理後的結果如下：

9.3.4 Express 常用的 API 簡介

9.3.4.1 express()

- 說明:
  - ✓ Creates an Express application.
  - ✓ The express() function is a top-level function exported by the express module.
  - ✓ 例如：

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

9.3.4.2 app.get(path, callback [, callback ...])

- 說明:
  - ✓ Routes HTTP GET requests to the specified path with the specified callback functions.
- 參數:

Argument	Description
path	・ The path for which the middleware function is invoked; can be any of: ✓ A string representing a path. ✓ 其他省略
callback	・ Callback functions; can be: ✓ A middleware function. ✓ A series of middleware functions (separated by commas). ✓ An array of middleware functions. ✓ A combination of all of the above.

- 中介軟體函式(Middleware Functions)的三個參數補充：

```
function(req,res,next){ }
```

- ✓ req <Reuest>
  - ◆ The req object represents the HTTP request and has properties for the request query string, body, HTTP headers, and so on.
  - ◆ The req object is an enhanced version of Node's own request object and supports all built-in fields and methods.
- ✓ res <Response>
  - ◆ The res object represents the HTTP response that an Express app sends when it gets an HTTP request.
  - ◆ The res object is an enhanced version of Node's own response object and supports all built-in fields and methods.
- ✓ next <Funciton>
  - ◆ It is called to pass control to the next middleware function.

9.3.4.3 res.send([body])

- ■ 說明:
  - ✓ Sends the HTTP response (and ends the request).
- ■ 參數:
  - ✓ When the parameter is a String,
    - ◆ the method sets the **Content-Type** to "text/html; charset=utf-8"
  - ✓ When the parameter is an Array or Object, Express responds with the JSON representation:

```
res.send({ user: 'tobi' });
res.send([1,2,3]);
```

    - ◆ the method sets the **Content-Type** to "application/json; charset=utf-8"

9.3.4.4 app.listen(port [, callback])

- ■ 說明:
  - ✓ Binds and listens for connections on the specified port.
  - ✓ This method is identical to Node's http.Server.listen().

9.3.4.5 app.use([path,] callback [, callback...])

- ■ 說明:
  - ✓ Mounts(裝載) the specified middleware function or functions at the specified path: the middleware function is executed when the base of the requested path matches path.
  - ✓ A route will match any path that follows its path immediately with a "/". For example: app.use('/apple', ...) will match "/apple", "/apple/images", "/apple/images/news", and so on.
  - ✓ Since path defaults to "/", middleware mounted without a path will be executed for every request to the app. For example, this middleware function will be executed for every request to the app:

```
app.use(function (req, res, next) {
  console.log('Time: %d', Date.now());
  //Date.now() 方法回傳自 1970/01/01 00:00:00 UTC 起經過的毫秒數。
  next();
});
```

- ■ 參數:
  - ✓ 請參考 app.get(path, callback [, callback...])。

10 附錄：實務練習–自動完成功能

10.1 使用 jQuery UI 的自動完成功能

10.1.1 使用要點

- 使用方法： $("input[type=text]").autocomplete({options})
- 常用的 options 屬性有 source、delay、minLength 等。
  - ✓ source: 設定自動完成之資料來源之 URL 字串
    - ✧ 如"_AjaxServer04-1.aspx"
    - ✧ 當使用者按下按鍵時，jQuery UI 會送出具 term 請求參數的 GET 請求，如: _AjaxServer04-1.aspx?term=使用者輸入之文字
    - ✧ 伺服器須回應 JSON(字串陣列)資料
  - ✓ delay: 設定按下按鍵至啟動查詢之間的延遲時間
    - ✧ 單位: 毫秒
    - ✧ 預設:300
  - ✓ minLength: 設定啟動查詢所需輸入的最小字元數
    - ✧ 預設:1

10.1.2 範例程 09-1.html (I)


```
<!DOCTYPE html>
<html>
<head>
    <title>實務練習 – 自動完成功能</title>
    <link rel="icon" href="./favicon.ico" />
    <link href="stylesheets/jquery-ui/base/jquery-ui.min.css" rel="stylesheet" />
    <link href="stylesheets/myStyleSheet.css" rel="stylesheet" />
    <script src="javascripts/jquery-3.6.0.min.js"></script>
    <script src="javascripts/jquery-ui.min.js"></script>
    <style>
        .ui-autocomplete-loading {
```

```
                background: white url("images/ui-anim_basic_16x16.gif") right center no-repeat;
            }
            (略)
        </style>
        <script type="text/javascript">
            $(function () {
                $("#tabs").tabs();
            });
        </script>
</head>
<body>
        <div id="tabs">
            <ul>
                <li><a href="#p1">使用jQuery UI的自動完成功能</a></li>
                <li><a href="#p2">自行開發自動完成功能</a></li>
            </ul>
            <div id="p1">
                <div class="main-box large-padding left-align">
                    <div id="header">
                        <div id="logo">
                            <a href="#">
                                <img src="images/greefies/logo.gif" />
                            </a>
                        </div>
                    </div>
                    <form id="form1" class="main-form">
                        <div class="form-group">
                            <label for="username">使用者名稱</label>
                            <input class="form-control" type="text" name="username"
                                id="username" placeholder="請輸入使用者名稱" />
                        </div>
                        <div class="form-group">
                            <input type="submit" value="送出" /> <div id="msg1"
                                                                    class="message-box inline"></div>
                        </div>
                    </form>
                    <div id="footer">
                        <div>
                            <a href="#">
                                <img src="images/greefies/footer_logo.gif" />
                            </a>
                        </div>
                    </div>
                    <script>
                        $("#username").autocomplete({
                            source: "_AjaxServer04-1.aspx"
                            //當使用者按下按鍵時，jQuery UI會送出具term請求參數的GET請求，
                                如: _AjaxServer04-1.aspx?term=使用者輸入之文字
                        });
```

```
                $("#form1").submit(function () {
                    var formData = $(this).serialize(); // new FormData(this)
                    $.ajax({
                        method: 'POST',
                        url: "_AjaxServer04-2.aspx",
                        data: formData,
                        success: function (data) {
                            $("#msg1").html(data);
                            setTimeout('$("#msg1").html("");', 3000);
                        },
                        error: function (jqXHR, textStatus, errorThrown) {
                            $("#msg1").html(
                                `<span class="color-red">
                                    伺服器回應: ${jqXHR.status} ${textStatus}
                                  ${errorThrown}</span>`);
                            setTimeout('$("#msg1").html("");', 3000);
                        }
                    });
                    return false;
                });
            </script>
        </div>
    </div>
    <div id="p2">
        (略)
    </div>
</div>
</body>
</html>
```

10.1.3 範例程式 _AjaxServer04-1.aspx

```
<%@ Page Language="C#" ContentType="text/html" %>
<script runat="server">
    static List<String[]> employees = new List<String[]> {
                new string[] { "1", "Nancy", "Davolio", "Sales Representative", "1948-12-08",
                               "1992-05-01", "507 - 20th Ave. E. Apt. 2A", "Seattle" },
                new string[] { "2", "Andrew", "Fuller", "Vice President, Sales", "1952-02-19",
                               "1992-08-14", "908 W. Capital Way", "Tacoma" },
                new string[]{ "3", "Janet", "Leverling", "Sales Representative", "1963-08-30",
                               "1992-04-01", "722 Moss Bay Blvd.", "Kirkland" },
                (略),
                new string[] { "8", "Laura", "Callahan", "Inside Sales Coordinator",
                               "2015-03-19", "1994-03-05", "4726 - 11th Ave. N.E.", "Seattle" },
                new string[] { "9", "anne", "Dodsworth", "Sales Representative", "2015-03-04",
                               "1994-11-15", "7 Houndstooth Rd.", "London" }
        };
    private void Page_Load(Object sender, EventArgs e)
    {
        IEnumerable<string> empNames =
```

```
            from emp in employees
            where emp[1].ToUpper().Contains(Request["term"].ToUpper())
            select emp[1];
        string jsonStr = "[";
        int count = empNames.Count();
        int index = 0;
        foreach (var empName in empNames)
        {
            jsonStr += "\"" +empName+ "\"";
            if (index++ < count - 1)
                jsonStr += ",";
        }
        jsonStr += "]";
        System.Threading.Thread.Sleep(2000);
        Response.Write(jsonStr);
    }
</script>
```

10.1.4 範例程式 _AjaxServer04-2.aspx

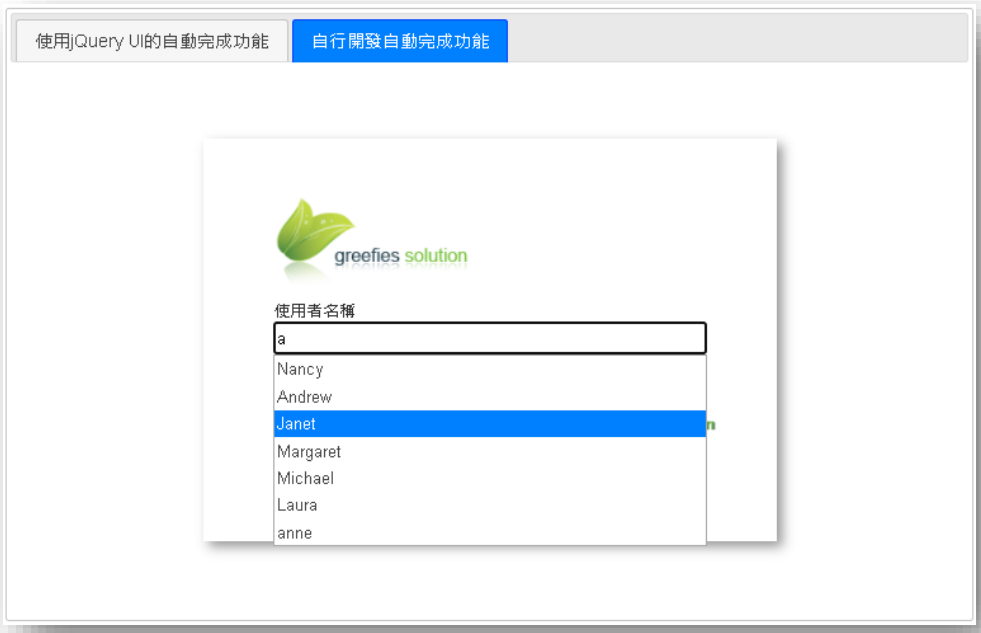
```
<%@ Page Language="C#" ContentType="text/html" %>
<%
    if (Request["username"] != null && Request["username"] != "")
        Response.Write("歡迎"+Request["username"]+ "光臨本網站...");
    else
        Response.StatusCode = 400; //Bad Request
%>
```

10.2 自行開發自動完成功能

10.2.1 自動完成功能的運作原理

- Client 端程式：
  - ✓ 在 input 元素的 input 事件上註冊事件處理器：每當使用者鍵入一個字元時就發送 AJAX 請求給網頁伺服器。
  - ✓ AJAX 請求可以使用 HTTP GET 或 POST 等方法。
  - ✓ 使用者在 input 元素所鍵入的文字，以 AJAX 請求之 Query String 將之傳送給網頁伺服器。例如：
    - ✧ _AjaxServer04-1.aspx?term=使用者在 input 元素所鍵入之文字
    - ✧ 一些知名網站首頁的搜尋皆提供自動完成功能，例如：Google、Facebook、Yahoo 等
  - ✓ 網頁伺服器回應後，將回應的 JSON 資料(字串陣列)予以解析，然後呈現在目前的網頁上。
- Server 端程式：
  - ✓ 網頁伺服器在接收到瀏覽器端所發送的請求後，解析 Query String，求得使用者所鍵入的文字(關鍵字)，然後據以尋找相關資源/資料。
  - ✓ 尋得的相關資源/資料以 JSON 格式包裝後回應給 Client 端的瀏覽器。

10.2.2 範例程式 T09-1.html (II)




```
<!DOCTYPE html>
<html>
<head>
    <title>實務練習 – 自動完成功能</title>
    <link rel="icon" href="./favicon.ico" />
    <link href="stylesheets/jquery-ui/base/jquery-ui.min.css" rel="stylesheet" />
    <link href="stylesheets/myStyleSheet.css" rel="stylesheet" />
    <script src="javascripts/jquery-3.6.0.min.js"></script>
    <script src="javascripts/jquery-ui.min.js"></script>
    <style>
        .ui-autocomplete-loading {
            background: white url("images/ui-anim_basic_16x16.gif") right center no-repeat;
        }
        .auto-complete-item:hover {
            background: #007fff;
            color: #fff;
        }
        #auto-complete-container {
            position: absolute;
            border: 1px solid #aaa;
            box-sizing: border-box;
            background: white;
            cursor: pointer;
            color: #444;
            display: none;
        }
    </style>
    <script type="text/javascript">
        $(function () {
```

```
                $("#tabs").tabs();
            });
        </script>
</head>
<body>
    <div id="tabs">
        <ul>
            <li><a href="#p1">使用jQuery UI的自動完成功能</a></li>
            <li><a href="#p2">自行開發自動完成功能</a></li>
        </ul>
        <div id="p1">
            (略)
        </div>
        <div id="p2">
            <div class="main-box large-padding left-align">
                <div id="header">
                    <div id="logo">
                        <a href="#">
                            <img src="images/greefies/logo.gif" />
                        </a>
                    </div>
                </div>
                <form method="post" id="form2" class="main-form"    autocomplete="off">
                    <div class="form-group">
                        <label for="username">使用者名稱</label>
                        <input class="form-control" type="text" name="username"
                                id="username2" placeholder="請輸入使用者名稱" />
                    </div>
                    <div id="auto-complete-container"></div>
                    <div class="form-group">
                        <input type="submit" value="送出" /> <div id="msg2"
                                                                class="message-box inline"></div>
                    </div>
                </form>
                <div id="footer">
                    <div>
                        <a href="#">
                            <img src="images/greefies/footer_logo.gif" />
                        </a>
                    </div>
                </div>
                <script>
                    var myInput = $("#username2");
                    var autoCompleteContainer = $("#auto-complete-container");
                    autoCompleteContainer.css({ width: myInput.outerWidth(false) });
                    //點擊.auto-complete-item以外的區域->將#auto-complete-container隱藏
                    $("body").click(function (event) {
                        // event.target指被點擊處的最內層的元素
                        if ($(event.target).is(".auto-complete-item") == false)
```

```
            autoCompleteContainer.hide();
});
//註冊input元素(id=username)的input事件處理器
myInput.on("input", function () {
    $.ajax({
        method: 'POST',
        url: "_AjaxServer04-1.aspx",
        data: { "term": $(this).val() },
        dataType: 'json',
        beforeSend: function () {
            myInput.addClass("ui-autocomplete-loading");
        },
        success: function (data) {
                //data相當於JSON.parse(XMLHttpRequest.responseText)
                所得之字串陣列如:["Margaret", "Michael"]
            console.log(data)
            $(".auto-complete-item", autoCompleteContainer).remove();
            $.each(data, function (index, value) {
                var myDiv = $(`<div class="auto-complete-item"
                                    style="padding:3px;">${value}</div>`);
                myDiv.click(function () {
                    myInput.val($(this).text());
                    autoCompleteContainer.hide();
                });
                autoCompleteContainer.append(myDiv);
            });
            autoCompleteContainer.show();
            autoCompleteContainer.position({
                of: myInput, //定位被對準的元素
                my: "left top", //定位時本身元素的 x y 位置
                at: "left bottom" //定位時被對準的元素的 x y 位置
            });
        },
        complete: function () {
            myInput.removeClass("ui-autocomplete-loading");
        }
    });
});
$("#form2").submit(function () {
    var formData = $(this).serialize(); // new FormData(this)
    $.ajax({
        method: 'POST',
        url: "_AjaxServer04-2.aspx",
        data: formData,
        success: function (data) {
            $("#msg2").html(data);
            setTimeout('$("#msg2").html("");', 3000);
        },
        error: function (jqXHR, textStatus, errorThrown) {
```

```
                        $("#msg2").html(
                            `<span class="color-red">
                                伺服器回應${jqXHR.status} ${textStatus}
                            ${errorThrown}</span>`);
                        setTimeout('$("#msg2").html("");', 3000);
                    }
                });
                return false;
            });
        </script>
    </div>
</div>
</div>
</body>
</html>
```