



# 新增資料表單



designed by freepik

Estimated time:

50 min.

III 資訊工業策進會 Institute for Information Industry

【Key Points】：

## 學習目標

- 14-1: 使用AJAX傳送表單資料
- 14-2: 資料寫入資料表
- 14-3: 處理回應



14-1

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

我們即將一起學習：

- AJAX觀念
- `$.ajax()` 使用說明
- JSON 介紹
- 建立HTML表單
- 設定 `$.ajax({ POST參數設定 })`
- 使用body-parser接收表單資料
- db.js · 我們的mysql封裝模組
- 新增資料到資料表
- 根據 MySQL 伺服器的回應，我們的後端程式得以告知前端是否新增成功
- 傳回給 Client 端的資料格式統一成: `{ errno: 1, data: '資料' , message: '訊息' }`
- 情境範例 – 新增藥局口罩庫存資訊

【Key Points】：

`$.ajax()` 使用說明

建立HTML表單

新增資料到資料表

## 14-1：使用AJAX傳送表單資料

- 關於AJAX、\$.ajax() 使用說明
- JSON 介紹
- 建立HTML表單
- 設定 \$.ajax({ POST參數設定 })



designed by freepik



designed by freepik

14-2

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

- 建立專案
- 匯入口罩庫存範例資料
- 關於AJAX
- \$.ajax() 使用說明
- JSON 介紹
- 建立HTML表單
- 設定 \$.ajax({ POST參數設定 })

【Key Points】：

\$.ajax() 使用說明

建立HTML表單

設定 \$.ajax({ POST參數設定 })

## 建立專案

1. 在「命令提示字元」，輸入：「**mkdir form**」 建立資料夾，當做我們專案的工作目錄。
2. 繼續輸入：「**cd form**」 切換目錄
3. 輸入：「**npm init -f**」 初始化專案
4. 用 **VS Code** 編輯我們的專案資料夾
5. 在 **VS Code** 按下「**Ctrl + 撇號**」開啟 **terminal** 終端機視窗。
6. 在終端機視窗，安裝套件(四項):  
**npm install express mysql body-parser ejs**
7. 在 **VS Code**，新增 **index.js** (本專案的主程式)
8. 啟動 **XAMPP Control Pane**，點按 **Apache** 與 **MySQL** 的「**Start**」按鈕，啟動兩套伺服器

14-3



1. 在「命令提示字元」，輸入：「**mkdir render\_page**」 建立資料夾，當做我們專案的工作目錄。
2. 繼續輸入：「**cd render\_page**」 切換到該目錄
3. 輸入：「**npm init -f**」 初始化專案
4. 用 **VS Code** 編輯我們的專案資料夾
5. 在 **VS Code** 按下「**Ctrl + 撇號**」開啟 **terminal** 終端機視窗。
6. 在終端機視窗，安裝套件: **npm install express mysql**
7. 在 **VS Code**，新增 **index.js** (本專案的主程式)
8. 啟動 **XAMPP Control Pane** (在檔案總管點兩下c:\xampp\xampp-control.exe )
9. 點按 **Apache** 與 **MySQL** 的「**Start**」按鈕，啟動兩套伺服器

### 【Key Points】：

初始化專案

安裝套件: **npm install express mysql**

啟動 **Apache** 與 **MySQL**

## 匯入口罩庫存範例資料

1. 在檔案總管點兩下c:\xampp\xampp-control.exe，啟動 XAMPP Control Panel
2. 點按 Apache 與MySQL的「Start」按鈕，啟動兩套伺服器
3. 點按MySQL那列的「Admin」按鈕，啟動phpMyAdmin 管理程式
4. 切換到 SQL 頁籤
5. 複製貼入 mask.sql 內容到SQL 頁籤，然後按下「執行」按鈕。  
(mask.sql位於本模組 example 資料夾)

14-4



如果你尚未進行過前一個模組的練習，請先按下列步驟建立 Mask 口罩庫存範例資料庫：

1. 在檔案總管點兩下c:\xampp\xampp-control.exe，啟動 XAMPP Control Panel
2. 點按 Apache 與MySQL的「Start」按鈕，啟動兩套伺服器
3. 點按MySQL那列的「Admin」按鈕，啟動phpMyAdmin 管理程式
4. 切換到 SQL 頁籤
5. 複製貼入 mask.sql 內容到SQL 頁籤，然後按下「執行」按鈕。

Note: mask.sql位於本模組 example 資料夾。

### 【Key Points】：

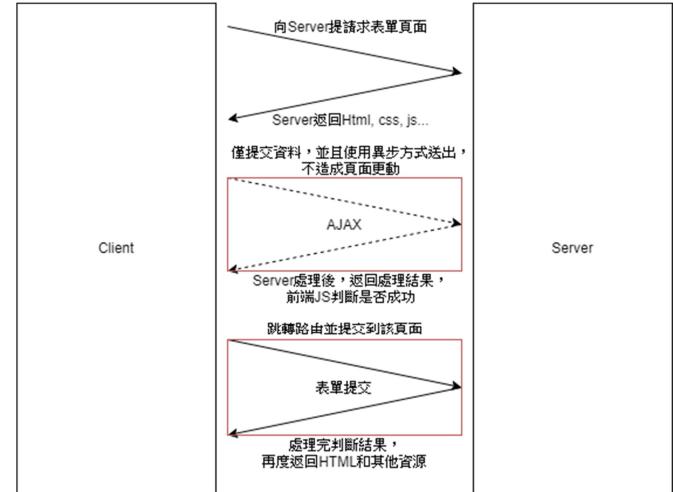
啟動 XAMPP Control Panel

切換到 SQL 頁籤

執行 mask.sql 內的程式

# 關於AJAX

- 什麼是 AJAX?
  - Asynchronous JavaScript and XML
  - 非同步 JavaScript 和 XML
- 為什麼要使用 AJAX?
  - 省去後端一再渲染整個頁面
  - 增加使用者體驗
  - 反覆使用單頁面資源



14-5

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

- AJAX是非同步 JavaScript 和 XML 的簡稱
  - 使用 AJAX，我們可以省去頁面跳轉等待的空白時間，省去Server渲染頁面的過程
  - 因為是異步的，所以大幅度提升了使用者體驗
1. 假設我們要到一個有表單的頁面，於是一開始向Server請求，Server當然也會回傳HTML，css...等檔案
  2. 但是當我們提交資料時，要傳送請求到另外一個路由，我們就必須跳轉到那個路由連帶資料傳遞進去，
  3. 可是程式碼處理邏輯的時候不會有頁面顯示，而是一個空白的頁面讓使用者看著，等待執行完顯示結果，或是頁面跳轉
  4. 這樣反覆頁面跳轉也是會讓使用者感受到不好的體驗
  5. 於是有了 AJAX 的出現，解決了這一部分的問題。使用異步的方式，用JavaScript傳送請求，如果執行完成，只要返回一個結果的訊息就好，不用看到頁面跳來跳去

## 【Key Points】：

AJAX是非同步 JavaScript 和 XML 的簡稱  
省去重覆地在後端渲染頁面過程  
增加使用者體驗

# \$.ajax() 使用說明

- **使用方式**
  - 原生JS的new XMLHttpRequest()
  - 引入jQuery函式庫，\$.ajax({})
- **重要參數 (以jQuery.ajax()為例)**
  - **url:** "連線目的地的網路路徑/路由"
  - **type:** 請求方法 ( POST、GET、PUT等等 )
  - **data:** 傳遞的資料，格式要看Client/Server商議的規格
  - **contentType:** 傳遞請求的編碼形式，預設是 "application/x-www-form-urlencoded"，本課程將以 "application/json;" 為主

```
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
  if (this.readyState === 4 && this.status === 200) {
    // do something
  }
};
xhttp.open("GET", "url", true);
xhttp.send();
```

```
$.ajax({
  url: "url",
  type: 'GET',
  data: {},
  success: function(res) {
    //do something
  }
});
```

14-6



1. 使用AJAX有兩種方式：一種是直接使用原生的 XMLHttpRequest()物件，另外則是透過(例如)jQuery函式庫整理好的 \$.ajax({})
2. 由於使用原生方法太過複雜，本課程將由jQuery輔助來實作AJAX
3. 那使用AJAX有幾個重要的參數，URL是我們要請求的路由，type則是請求方法，預設為GET，亦可請求POST、PUT、DELETE等等。
4. Data就看你有沒有需要。通常get方法資料就在URL中了，也不用特別傳遞；但是POST如果要提交表單，則一定要使用。
5. ContentType是我們傳遞請求的編碼形式，一般都是x-www-form-urlencoded，但是如果我們要用程式傳遞資料的話，使用"application/json;charset=utf-8"會比較建議。

## 【Key Points】：

本課程將由jQuery輔助來實作AJAX

AJAX有幾個重要的參數: url, type, data, contentType

本課程的 ContentType 使用"application/json;charset=utf-8"

# JSON 介紹

- 前後端常用的傳遞資料格式
- 容易閱讀與編寫
- 相對於XML，結構簡單，ex. {name: 'name'}
- 可放入String · Int · Array · Object · Boolean · Null 多種型態
- 大多數程式語言皆能處理 JSON

```
{  
    name: "Andy",  
    age: 18,  
    smoke: false,  
    hobby: [ "swiming", "hiking" ]  
}
```

14-7



- JSON 是個以純文字為基底，儲存和傳送簡單結構資料
- 前後端溝通相當常用的資料格式
- 可以儲存的資料有字串, 數字, 陣列, 物件, 布林, 空值
- 也可以透過物件或陣列來傳送較複雜的資料
- 大多數程式語言皆可以閱讀JSON

請留意下列符號的作用：

[ ]	一對中括號表示陣列
{ }	一對大括號表示物件
,	多個項目的分隔字元
:	分隔屬性與屬性值
"	文字資料要用雙引號包起來

## 【Key Points】：

JSON 是個以純文字為基底，儲存和傳送簡單結構資料  
前後端溝通相當常用的資料格式  
留意七個符號的作用

# 建立HTML表單

- 在 views 資料夾，新增 index.ejs
- 加入 <form></form> 表單
- 加入各個資料輸入控制項
- 從官網 CDN 引用 jQuery 函式館

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>新增表單</title>
</head>
<body>
    <form id="form">
        藥局名稱: <input type="text" name="name"><br>
        電話: <input type="text" name="phone"><br>
        地址: <input type="text" name="address"><br>
        成人口罩庫存: <input type="text" name="adult_mask"><br>
        兒童口罩庫存: <input type="text" name="child_mask"><br>
        <button type="button" id="submit">新增</button>
    </form>
    <script
        src="https://code.jquery.com/jquery-3.4.1.min.js"
        integrity="sha256-CSXorXvZcTkaix6Yvo6EppqpmmF0yJpgAHRrMj4="
        crossorigin="anonymous"></script>
</body>
</html>
```

14-8



1. 在 views 資料夾，新增 index.ejs
2. 加入 <form></form> 表單
3. 參照資料庫的資料表結構，加入各個資料輸入控制項到表單之中
4. 加入 type="submit" 的按鈕
5. 從官網 <https://code.jquery.com/jquery-3.4.1.min.js> CDN 引用 jQuery

## 【Key Points】：

在 views 資料夾，新增 index.ejs

加入 <form></form> 表單，內含各個資料輸入控制項

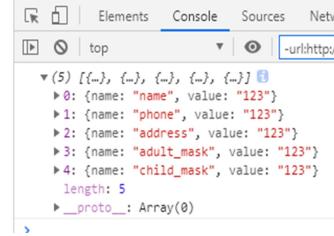
從官網 <https://code.jquery.com/jquery-3.4.1.min.js> CDN 引用 jQuery

# 建立HTML表單

- `$('#form').serializeArray()` 獲取欄位值

```
<script>
  //監聽click事件
  $('#submit').on('click', function() {
    //整理表單資料到變數
    console.log($('#form').serializeArray())
  })
</script>
```

藥局名稱: 123  
電話: 123  
地址: 123  
成人口罩庫存: 123  
孩童口罩庫存: 123  
新增



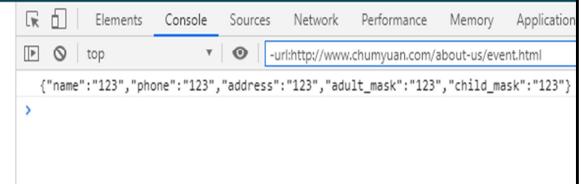
- 將所有值轉成JSON格式

```
<script>
  //監聽click事件
  $('#submit').on('click', function() {
    //整理表單資料到變數
    var data = $('#form').serializeArray()
    JSONData = serializeToJson(data)
    console.log(JSONData)
  })

  function serializeToJson(data) {
    var values = {};
    for(index in data){
      values[data[index].name] = data[index].value;
    }
    return JSON.stringify(values)
  }

</script>
```

藥局名稱: 123  
電話: 123  
地址: 123  
成人口罩庫存: 123  
孩童口罩庫存: 123  
新增



14-9



1. 抓取 `id="sumbit"` 的元素來監聽點擊事件，截取出 `id="#form"` 的表單序列化後的資料
2. 序列化資料後皆為 `[ {name: value}, {name: value}, ... ]` 格式，資料格式不是非常的明確。
3. 我們撰寫一個 `serializeToJson()` 的函式，將每個「`name` 配對一個 `value`」轉成各個屬性。  
例如: `[ {name: "name", value: "Lin"}, {name: "phone", value: "0963123456"} ]`  
轉成: `{ name: "Lin", phone: "0963123456" }`
4. 最後，再用 `JSON.stringify()` 字串化
5. `JSON.stringify()` 字串化的這筆資料，就可以當作 `data` 傳遞給後端

## 【Key Points】：

抓取 `id="sumbit"` 的元素來監聽點擊事件

撰寫一個 `serializeToJson()` 的函式，轉換資料成我們希望的格式

`JSON.stringify()` 字串化資料

# 設定 \$.ajax({ POST參數設定 })

- 撰寫AJAX請求

- url使用 /add
- type使用 POST
- contentType使用 json格式
- success 返回200(OK)
- error返回失敗報告

```
//監聽click事件
$('#submit').on('click', function() {
    //整理表單資料到變數
    var data = $('#form').serializeArray()
    JSONData = serializeToJSON(data)

    //ajax請求
    $.ajax({
        url: "/add",
        type: "POST",
        contentType: "application/json; charset=utf-8",
        data: JSONData,
        success: function(res) {
            var res = JSON.parse(res)
            //後端會封裝一個response class，預先寫好執行No.
            if(res errno === 1) {
                alert("新增成功!")
            } else if(res errno === 0) {
                alert("新增失敗!")
            }
        },
        error: function() {
            alert("系統錯誤!")
        }
    })
    //序列化轉JSON
    function serializeToJSON(data) {
        var values = {};
        for(index in data){
            values[data[index].name] = data[index].value;
        }
        return JSON.stringify(values)
    }
})
```

14-10

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

1. 將資料處理成JSON後，我們開始撰寫 \$.ajax({ 通訊參數設定 })
2. 在url 使用 /add路由 (下一個知識點馬上就會講到)
3. method: 使用POST方法
4. contentType 定義JSON utf-8格式
5. 如果請求返回200 OK，AJAX則會處理succes這個function，我們依照返回的errno來判斷是否新增成功
6. 如果返回404之類的錯誤訊息，則會處理error這個function

## 【Key Points】：

url 使用 /add路由 (下一個知識點馬上就會講到)

method: 使用POST方法

contentType 定義JSON utf-8格式

## 14-2：資料寫入資料表

- 使用body-parser接收表單資料
- db.js · 我們的mysql封裝模組
- 新增資料到資料表



designed by freepik



designed by freepik

14-11

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

在這個章節，將說明如何接收Client傳過來的資料，並且把資料新增資料庫內：

- 使用body-parser接收表單資料
- body-parser 模組，會將收到的 POST 的資料存放在 req.body
- 封裝mysql到 db.js 模組，讓後來的我們在寫資料庫程式的時後，更加方便
- 新增資料到資料表
- 定義SQL語句，「？」參數化查詢，為避免SQL Injection的一種方式
- 填寫表單，點按「送出」按鈕
- 啟動phpMyAdmin 管理程式，看看是否真的有我們新增的內容

### 【Key Points】：

使用body-parser接收表單資料

新增資料到資料表

啟動phpMyAdmin 管理程式，看看是否真的有我們新增的內容

# 使用body-parser接收表單資料

- 使用body-parser
  - express引入使用json
- 新增POST路由 /add
  - req.body中獲取POST資料
  - 打印資料
- 測試功能
  - node index.js
  - 送出表單

```
var express = require('express');
var bodyParser = require('body-parser');
var app = express();
//解析json資料
app.use(bodyParser.json());
```

```
app.post('/add', function(req, res){
  var body = req.body
  console.log(body)
})
```

藥局名稱: 1231  
電話: 123  
地址: 123  
成人口罩庫存: 123  
孩童口罩庫存: 123  
新增

```
C:\Users\Administrator\Desktop\14_1ng\form>node index.js
{ name: '1231',
  phone: '123',
  address: '123',
  adult_mask: '123',
  child_mask: '123' }
```

14-12



1. 之前我們在 Module 11. 的時候，有使用過body-parser，但是我們使用的時候是用urldecode 方式
2. 這次我們將 body-parser 改成以 json 格式來接收資料
3. 新增 /add的POST路由。
4. body-parser 模組，會將收到的 POST 的資料存放在 req.body，所以，我們先打印看看傳過來了什麼資料
5. 在 VS Code 按下「Ctrl + 撇號」開啟 terminal 終端機視窗，輸入: node index.js
6. 以瀏覽器連接 http://localhost:3000，填寫表單，點按「送出」按鈕。
7. 查看終端機視窗顯示的資料

## 【Key Points】：

以 body-parser 模組接收 json 格式的資料

body-parser 模組，會將收到的 POST 的資料存放在 req.body

檢視用戶端 POST 過來的資料

# db.js，我們的mysql封裝模組

- 新增db.js

- 引入mysql

- 連線

- 設定好配置
    - 連線

- 查詢語句

- SqI語句
    - 需要執行的data
    - 回調結果

- 關閉連線

```
var mysql = require('mysql');

exports.exec = (sql,data,callback) => {
    const connection = mysql.createConnection({
        host:'localhost',
        user:'root',
        password:'root',
        database:'mask'
    });
    connection.connect();

    connection.query(sql,data,function(error,results,fields){
        if(error) {
            console.log(error)
        };
        callback(results, fields);
    })
    connection.end();
}
```

14-13



- 封裝mysql到模組的原因之一：把連線程式碼寫在index.js，會造成裏頭太過雜亂
- 原因之二：封裝好的功能，使用後mysql才會連線，可以避免一開始啟動就連線，造成不必要的消耗
- 因為匯出的功能只有一個，所以直接用 exports.function 的方式輸出
- 函式有三個參數要傳入
  - sql是sqI語句
  - data則是要傳入以陣列形式的值
  - callback是程式碼執行完，可以讓引用的人自行撰寫程式，並且執行的功能
- mysql的基本操作皆封裝在模組裡頭，包括：設定配置、連線設定、執行SQL語句、關閉連線等等。

## 【Key Points】：

封裝mysql到模組

匯出的功能只有一個，所以直接用 exports.function 的方式輸出

函式有三個參數要傳入：sql, data, callback

# 新增資料到資料表

- **require** 我們自己寫 db.js

```
var db = require('./db')
```

- 使用方式

- 定義SQL語法，「？」參數化查詢，為避免SQL Injection的一種方式
- data為一陣列，需要照著欄位順序來擺放
- db.exec類似於connection.query

```
app.post('/add', function(req, res){  
    var body = req.body  
    var sql = `INSERT INTO inventory(name, phone, address, adult_mask, child_mask) VALUES(?, ?, ?, ?, ?);`  
    var data = [body.name, body.phone, body.address, parseInt(body.adult_mask), parseInt(body.child_mask)]  
    db.exec(sql, data, function(results, fields) {  
        console.log(results)  
    })  
})
```

14-14



## 使用方式

1. 引入我們自己撰寫的db.js

```
var db = require('./db')
```

2. 定義SQL語句，「？」參數化查詢，為避免SQL Injection的一種方式

```
INSERT INTO inventory(name, phone, address, adult_mask, child_mask) VALUES(?, ?, ?, ?, ?);
```

3. 傳入data，如果沒有使用到「？」，可為空陣列 []

```
[body.name, body.phone, body.address, parseInt(body.adult_mask), parseInt(body.child_mask)]
```

4. 執行，在callback內查看結果

```
db.exec(sql, data, function(results, fields) {  
    console.log(results)  
})
```

5. Callback內亦可自行撰寫其他程式

## 【Key Points】：

引入我們自己撰寫的db.js

定義SQL語句，「？」參數化查詢，為避免SQL Injection的一種方式

傳入data 並在callback內查看結果

# 新增資料到資料表

- 測試功能

- node index.js

- 送出表單

- insertId:29

- phpMyAdmin查看

The screenshot illustrates the workflow for inserting data into a MySQL database:

- Node.js Output:** Shows the command "node index.js" running in a terminal, with the response indicating a successful insertion: "insertId: 29".
- Form Screenshot:** A screenshot of a web form with fields for 藥局名稱 (藥局名称), 電話 (Phone), 地址 (Address), 成人口罩庫存 (Adult Mask Stock), and 孩童口罩庫存 (Child Mask Stock). A "新增" (Add) button is visible.
- phpMyAdmin Screenshot:** Shows the results of a SQL query: "SELECT \* FROM `inventory` WHERE id = 29". The result table displays one row with id 29 and values 123 for all other columns.

14-15



1. 在 VS Code 按下「Ctrl + 撇號」開啟 terminal 終端機視窗，輸入: node index.js
2. 以瀏覽器連接 <http://localhost:3000>，填寫表單，點按「送出」按鈕。
3. 查看終端機視窗顯示的資料  
其中的 insertId，是可以讓我們判斷是否新增成功的重要依據
4. 在檔案總管點兩下 c:\xampp\xampp-control.exe，啟動 XAMPP Control Panel
5. 點按 MySQL 那列的「Admin」按鈕，啟動 phpMyAdmin 管理程式
6. 先點一下 Mask(切換到 Mask 資料庫)，再點按 SQL 頁籤。
7. 輸入並執行 SELECT \* FROM `inventory`，看看是否真的有我們新增的內容。

【Key Points】：

node index.js 啟動伺服器

填寫表單，點按「送出」按鈕

以 phpMyAdmin 確認是否真的有我們新增的內容

## 14-3 處理回應

- 建立 Success 與 Error 類別
- 使用 Success 與 Error 類別
- 後端程式告知前端是否新增成功
- 情境範例 – 新增藥局口罩庫存資訊



designed by freepik



designed by freepik

14-16

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

在這個章節，將把新增後的消息，回傳回去給 Client。但是我們必須定義好統一的回傳格式，Client 端才好寫程式判斷。

- 建立 Success 與 Error 類別
- 傳回給 Client 端的資料格式統一成: { errno: 1, data: '資料' , message: '訊息' }
- errno: 1為執行成功，errno: 0為執行失敗
- 根據 MySQL 伺服器的回應，我們的後端程式得以告知前端是否新增成功
- 情境範例 – 新增藥局口罩庫存資訊

### 【Key Points】：

建立 Success 與 Error 類別

傳回給 Client 端的資料格式統一成: { errno: 1, data: '資料' , message: '訊息' }

errno: 1為執行成功，errno: 0為執行失敗

# 建立 Success 與 Error 類別

- 新增一個程式檔案: response.js
- 新增兩個類別: Success 和 Error
- 統一返回 errno, message, data
- 判斷 data 是否為 string
  - 是，將它變成 message
  - 否，則放入data
- errno
  - 1: 表示成功
  - 0: 表示失敗

```
class Success {  
    constructor(data, message) {  
        if(typeof data === 'string') {  
            this.message = data  
            data = null  
            message = null  
        }  
        if(data) {  
            this.data = data  
        }  
        if(message) {  
            this.message = message  
        }  
        this(errno = 1)  
    }  
}  
  
class Error {  
    constructor(data, message) {  
        if(typeof data === 'string') {  
            this.message = data  
            data = null  
            message = null  
        }  
        if(data) {  
            this.data = data  
        }  
        if(message) {  
            this.message = message  
        }  
        this(errno = 0)  
    }  
}  
  
module.exports = {  
    Success,  
    Error  
}
```

14-17



- 專案下新增一個程式檔案: response.js
- 內含兩個class，分別為Success 與 Error
- 傳回給 Client 端的資料格式是: { errno: 1, data: '資料' , message: '訊息' }
- 利用建構函式判斷傳入值的型別。因為有可能只傳訊息不傳資料，或是只傳資料不傳訊息。所以統一第一個傳遞參數為data，第二個為message。判斷data是否為string，如果data是string，代表只想傳遞訊息，沒有要傳遞data，那就設定this.message = data，然後將data = null, message = null
- 反之，如果data是陣列或是值，那麼就會直接賦予this.data = data，如果有訊息就this.message = message
- 然後我們設定 errno: 1為執行成功，errno: 0為執行失敗

## 【Key Points】：

新增一個程式檔案: response.js, 內含兩個class，分別為Success 與 Error  
傳回給 Client 端的資料格式是: { errno: 1, data: '資料' , message: '訊息' }  
errno: 1為執行成功，errno: 0為執行失敗

# 使用Success 與 Error 類別

- 呼叫 require( ) 引用 response.js

- 新增兩個GET路由

- /success

- /error

- res.end( JSON.stringify(new Success()) )

- 測試功能

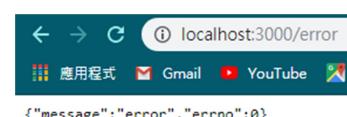
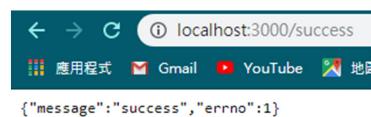
1. node index.js

2. 訪問 <http://localhost:3000/success>

3. 訪問 <http://localhost:3000/error>

```
var { Success, Error } = require('./response')
```

```
app.get('/success', function(req, res){  
    res.end(  
        JSON.stringify(new Success('success'))  
    )  
})  
  
app.get('/error', function(req, res){  
    res.end(  
        JSON.stringify(new Error('error'))  
    )  
})
```



14-18

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

- 引用 Success 與 Error 這兩個類別:

```
var { Success, Error } = require('./response')
```

- 傳遞訊息，因為它是class，要先new 初始化建構函式才能執行

```
res.end(JSON.stringify(new Success('success')))
```

1. 建立兩GET 路由 /success /error

2. 分別回傳成功和失敗

3. 在 VS Code 按下「Ctrl + 撇號」開啟 terminal 終端機視窗，輸入: node index.js

4. 查看 <http://localhost:3000/success>

5. 查看 <http://localhost:3000/error>

【Key Points】：

建立兩GET 路由 /success /error，分別回傳成功和失敗

```
var { Success, Error } = require('./response')
```

```
res.end(JSON.stringify(new Success('success')))
```

# 後端程式告知前端是否新增成功

- 以 `insertId` 判斷是否新增成功
  - 有，返回成功訊息
  - 沒有，返回失敗訊息

```
app.post('/add', function(req, res){  
    var body = req.body  
    var sql = `INSERT INTO inventory(name, phone, address, adult_mask, child_mask) VALUES(?, ?, ?, ?, ?);`  
    var data = [body.name, body.phone, body.address, parseInt(body.adult_mask), parseInt(body.child_mask)]  
    db.exec(sql, data, function(results, fields) {  
        if(results.insertId){  
            res.end(  
                JSON.stringify(new Success('insert success'))  
            )  
        } else {  
            res.end(  
                JSON.stringify(new Error('insert failed'))  
            )  
        }  
    })  
})
```

14-19



1. 送出 `INSERT INTO` 指令給 MySQL 資料伺服器
2. 新增資料後，MySQL 會傳回新增結果
3. 以 `results.insertId` 來判斷
4. 如果有值，表示新增資料成功，傳出 `success` 給瀏覽器
5. 反之，傳出 `error` 給瀏覽器

## 【Key Points】：

送出 `INSERT INTO` 指令給 MySQL 資料伺服器  
以 `results.insertId` 來判斷  
如果有值，表示新增資料成功，傳出 `success` 給瀏覽器

# 情境範例 – 新增藥局口罩庫存資訊

- 情境範例

- node index.js
- 前往 <http://localhost:3000>
- 鍵盤 F12 滑鼠點擊 Network
- 提交表單
- 查看返回訊息

```
success: function(res) {
  var res = JSON.parse(res)
  //後端會封裝一個response class，預先寫好執行No.
  if(res errno === 1) {
    alert("新增成功!")
  } else if(res errno === 0) {
    alert("新增失敗!")
  }
},
error: function() {
  alert("系統錯誤!")
}
```

Name	Headers	Preview	Response	Timing
localhost				
jquery-3.4.1.min.js				
add			▼ {message: "insert success", errno: 1} message: "insert success"	



14-20

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

1. 在 VS Code 按下「Ctrl + 撇號」開啟 terminal 終端機視窗，輸入: node index.js
2. 用瀏覽器連接 <http://localhost:3000>
3. 在瀏覽器，按下「Ctrl + Shift + I」開啟 Chrome 開發人員工具，切換到「Network」頁籤，稍後在此觀察請求與回應的通訊情況
4. 填寫資料，送出表單
5. 查看返回訊息

## 【Key Points】：

在 VS Code 按下「Ctrl + 撇號」開啟 terminal 終端機視窗，輸入: node index.js

用瀏覽器連接 <http://localhost:3000>

填寫資料，送出表單

# Summary 〈 精華回顧 〉

- 關於 `$.ajax()`
- JSON格式
- body-parser解析JSON方式
- 封裝mysql
- SQL語句使用「？」，將傳遞值進行轉譯，可以防止 SQL Injection
- 統一返回格式
- 接收返回訊息，並且alert出來



14-21

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

回顧一下稍早之前的內容：

一開始，我們介紹了jQuery.ajax() 的使用方式以及相關參數

之後，介紹 JSON 這種傳遞資訊的資料格式

再來，更改我們用過的body-parser，把原本的urlencode改成json

另外，我們也封裝mysql到 db.js 模組，讓後來的我們在寫資料庫程式的時後，更加方便

SQL語句使用「？」，將傳遞值進行轉譯，可以防止 SQL Injection

建立responses，統一返回格式，方便瀏覽器端的 JavaScript 判斷及使用資料。

接收到Client端的請求後，我們用 errno 來告訴Client端，這趟的資料處理是否成功。

## 【Key Points】：

jQuery.ajax() 的使用方式以及相關參數

用 body-parser 解讀 JSON 資料

封裝 mysql到 db.js 模組，讓後來的我們在寫資料庫程式的時後，更加方便

# 線上程式題

- **14-1 如何以 \$.ajax() 來 POST 資料**

如何以 jQuery 的 \$.ajax() 來 POST 資料到伺服器？

- **14-2 使用AJAX傳送JSON資料**

有一段程式的內容如下：

```
$.ajax({  
    //...  
})
```

如何使用JSON的方式傳遞請求？

- **14-3 處理並回應 POST 的結果**

依據我們課堂看過的 Success 範例，下列程式，請問返回的格式？

```
res.end(  
    JSON.stringify(new Success('insert success'))  
)
```

14-22



題目名稱: 14-1

如何以 \$.ajax() 來 POST 資料如何以 jQuery 的 \$.ajax() 來 POST 資料到伺服器？

題目名稱: 14-2 使用AJAX傳送JSON資料

有一段程式的內容如下：

```
$.ajax({  
    //...  
})
```

如何使用JSON的方式傳遞請求？

題目名稱: 14-3 處理並回應 POST 的結果

依據我們課堂看過的 Success 範例，下列程式，請問返回的格式？

```
res.end(  
    JSON.stringify(new Success('insert success'))  
)
```

【Key Points】：

14-1 如何以 \$.ajax() 來 POST 資料

14-2 使用AJAX傳送JSON資料

14-3 處理並回應 POST 的結果

# 課後練習題(Lab)

- **情節描述:**

本練習假設Node.js已經安裝於Windows作業系統並且已安裝好XAMPP，也假設您學過如何在前端使用AJAX技術傳遞表單的內容到後端Web API 服務。

- **預設目標:**

- 將表單資料轉換成JSON
- 解析回應類別並探討其原理與應用

- **Lab01: 表單資料轉換JSON**

Estimated time:

30 minutes

- **Lab02: 解析回應類別並探討其原理與應用**

14-23



## 【預設目標】

- 將表單資料轉換成JSON
- 解析回應類別並探討其原理與應用

Lab01: 表單資料轉換JSON

Lab02: 解析回應類別並探討其原理與應用

完成後的程式與檔案，請參考 Example 資料夾的內容。

關鍵程式:

```
$.ajax({  
    url: "/add",  
    type: "POST",  
    contentType: "application/json; charset=utf-8",  
    data: JSONData,  
    ...  
})  
JSON.stringify(values)
```

## 【Key Points】:

Lab01: 表單資料轉換JSON

Lab02: 解析回應類別並探討其原理與應用

\$.ajax()

# 範例程式使用說明

- 範例程式資料夾: Module\_14\_example
- 使用步驟:
  1. 安裝 Node.js
  2. 安裝 Visual Studio Code
  3. 安裝 XAMPP, 匯入 mask 口罩庫存資料庫
  4. 以 Visual Studio Code 開啟本模組的範例資料夾
  5. 在 Visual Studio Code 按下「Ctrl + 滑鼠」開啟 terminal 終端機視窗。
  6. 在終端機視窗，輸入下列指令，安裝必要的模組套件:  
npm install
  7. 在終端機視窗，輸入 node index.js
  8. 啟動瀏覽器，連接 http://localhost:3000

14-24



使用步驟:

1. 安裝 Node.js (<https://nodejs.org/en/>)
2. 安裝 Visual Studio Code (<https://code.visualstudio.com/>)
3. 安裝 XAMPP ([https://www.apachefriends.org/zh\\_tw/index.html](https://www.apachefriends.org/zh_tw/index.html))
4. 在檔案總管點兩下c:\xampp\xampp-control.exe，啟動 XAMPP Control Panel
5. 點按 Apache 與MySQL的「Start」按鈕，啟動兩套伺服器
6. 點按MySQL那列的「Admin」按鈕，啟動phpMyAdmin 管理程式，切換到 SQL 頁籤
7. 複製貼入 mask.sql 內容到SQL 頁籤，然後按下「執行」按鈕。
8. 以 Visual Studio Code 開啟範例資料夾  
在檔案總管，滑鼠右鍵點按「本範例資料夾」，從快捷功能表點選「以Code開啟」  
或者，啟動 Visual Studio Code 之後，功能表 File | Open Folder，選擇「本範例資料夾」
9. 在 Visual Studio Code 按下「Ctrl + 滑鼠」開啟 terminal 終端機視窗。
10. 在終端機視窗，輸入下列指令，安裝必要的模組套件:  
npm install
11. 在終端機視窗，輸入 node index.js
12. 啟動瀏覽器，連接 http://localhost:3000

【Key Points】：

程式執行環境 Node.js

程式開發編輯環境: Visual Studio Code

在 Visual Studio Code 按下「Ctrl + 滑鼠」開啟 terminal 終端機視窗，輸入：「node 主程式.js」