# Module 14 – 新增資料表單

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:

1. 將表單資料轉換成 JSON
2. 解析回應類別並探討其原理與應用

估計時間: 30 分鐘

## ★★★ Lab01: 表單資料轉換 JSON ★★★

1. 以下是個 HTML 表單，請閱讀程式，並且繼續完成 serializeToJSON() 的程式。

```
<form id="form">
    名稱: <input type="text" name="name"><br>
    電話: <input type="text" name="phone"><br>
    <button type="button" id="submit">新增</button>
</form>
<script
src="https://code.jquery.com/jquery-3.4.1.min.js"
integrity="sha256-CSXorXvZcTkaix6Yvo6HppcZGetbYMGWSFlBw8HfCJo="
crossorigin="anonymous"></script>
<script>
    $('#submit').on('click', function() {
        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)
                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!")
            }
        })
    })
    function serializeToJSON(data) {
        var values = {};
        //
          //
          //
          //
          //
        //
        return ???
    }
</script>
```

2. 答案:

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

3. 學會 JSON 格式轉換非常重要，之後課程皆會用到此程式碼

## ★★★ Lab02: 解析回應類別並探討其原理與應用 ★★★

### 1. 查看 response.js 撰寫方式

```
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
}
```

2. class 是程式設計中的一個非常常看到的撰寫方式，目的在於將功能打包，讓大家在不同的程式碼中，方便引用，且擁有一致的流程。增加系統可維護性。
3. 新增 test.js，使用 new Success( 'message' )，打印在 cmd 看看

```
const { Success } = require('./response');
console.log(new Success('insert success'));
```

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

```
C:\Users\Administrator\Desktop\14_ing-1\form>node test.js
Success { message: 'insert success', errno: 1 }
```

5. 可以發現 class 裡面，使用 this 宣告的值(class 中的變數)皆會以 Object 被打印出來
6. 所以，我們可以自己定義要返回的值，只要在 class 中宣告好變數即可，每次輸入對應的值，則會返回相同格式
7. 請問如果要設定格式為
   {status: (200/404/403), data: (array/object), message: (string)}，
   要如何撰寫 class?

8. 答案:

```
class Status {
    constructor(status, data, message) {
         this.status = parseInt(status)
        if(typeof data === 'string') {
            this.message = data
            data = nulls
            message = null
        }
        if(data) { this.data = data }
        if(message) { this.message = message }
    }
}
```