

CHAPTER

4

瀏覽器端程式庫

- Microsoft AJAX Library
- \$get 方法
- \$addHandler 方法
- \$find 方法
- \$addHandlers 方法
- 類別設計
- 建立繼承類別
- 建立介面類別
- Sys.UI 命名空間
- DOMElement 類別
- DOMEvent 類別

4.1 Microsoft AJAX Library

對於非 ASP.NET 2.0 系統的其他平台網頁設計者而言，並不能使用 ASP.NET AJAX Extensions 1.0 套件所帶來的方便性，於是微軟開發了 AJAX Library 套件來協助非 ASP.NET 2.0 平台的使用者建立豐富的 AJAX 網頁。Microsoft AJAX Library 是一組可在瀏覽器端獨立運作的 JavaScript 函式庫，包含網頁核心服務以及提供瀏覽器的 XML 遠端資料存取能力，廣泛支援包括微軟 Internet Explore、Mozilla、Firefox 與 Apple Safari 等瀏覽器。

如果已經安裝 ASP.NET AJAX Extensions v1.0 者不需要再下載 Microsoft AJAX Library，其他平台使用者請參考 3.1.1 節下載此套件。雖然使用 ASP.NET 2.0 系統者較少利用 AJAX Library 來建立 AJAX 網頁，但了解 AJAX Library 設計過程對 AJAX 運作有幫助。若是讀者目前對於使用 AJAX Library 建立 AJAX 網頁缺乏興趣，也可以暫時跳過此章，等將來需要用到時再來閱讀。

4.1.1 簡介

AJAX Library 是專為 AJAX 功能提供的 Script，它是由標準 Javascript 發展而來，擴充了 Javascript 的功能，可以支援目前使用的多種瀏覽器，也相容於大部分的伺服器。

Javascript 本身已有物件導向的設計，但相關的支援並不完整，AJAX Library 為 Javascript 提供更多的物件導向擴充，例如：繼承、介面、DOM 文件操作等，讓網頁設計者能用最簡單的方法，撰寫與伺服器互動的 Javascript 程式，讓非使用 ASP.NET 者也能快速的建立 AJAX 網頁程式。

由於 AJAX Library 主要針對非 ASP.NET 使用者設計，而非 ASP.NET 使用者無法叫用 ASP.NET 所提供的伺服器控制項，因此 AJAX Library 提供了一組用戶端控制項，讓網頁設計者可以在不利用伺服器端技術下，用這一組控制項模擬出伺服器控制項功能，甚至可以讓瀏覽器端直接存取伺服器的資料。

4.1.2 命名空間

在 ASP.NET 中，大部分的功能都使用類別庫的方式提供給使用者，而使用者要利用這些功能時，則透過命名空間 (NameSpace) 來取用。AJAX Library 也不例外，它將各種功能分類置於數

個命名空間，要使用該命名空間包含的功能時，需先呼叫該命名空間 (語法為：using 命名空間名稱)。

AJAX Library 對於原本就慣於使用 ASP.NET 者而言，學習上非常容易，因為兩者的觀念和用法都是一致的，在 ASP.NET 中的基礎命名空間為「System」，而在 AJAX Library 中的基礎命名空間為「Sys」，其餘的階層架構、分類方式等皆大致相同。AJAX Library 包含的命名空間簡述於下：

- Sys：最基本的命名空間，提供基礎的通用類別。
- Sys.Net：用於管理 ASP.NET AJAX 瀏覽器端應用程式與 Web 伺服器之間的服務溝通功能類別。
- Sys.Serialization：提供 ASP.NET AJAX 應用程式資料序列化的類別功能。
- Sys.Services：提供 ASP.NET AJAX 瀏覽器端應用程式透過 Script 存取 ASP.NET 服務所需的類別功能，包含 Authentication 及 Profile 服務。
- Sys.WebForms：提供與局部功能有關的類別功能。
- Sys.UI：提供與視覺化介面功能有關的類別功能。

4.2 全域命名空間

AJAX Library 還有一個全域命名空間 (Global Namespace) ，這是 AJAX Library 最常使用的部分。全域命名空間提供延伸 Javascript 功能的型別和成員，讓網頁設計者可以透過這些物件導向模組簡化 Javascript 程式的撰寫。

4.2.1 方法

全域命名空間提供了數個實用的「方法」讓設計者呼叫使用，可用來存取 AJAX Library 中相關的 API 函式。這些方法都是以「\$」符號做為起始字元，而且由於它們是位於全域命名空間，因此通常不需要透過物件實體即能使用。

AJAX Library 的全域命名空間所提供的方法為：

- \$get：由網頁元素的 ID 來取得網頁元素的方法，相當於 Javascript 中的 getElementById 執行的結果。
- \$create：建立一個的物件。
- \$find：尋找網頁上特定的元素。
- \$addHandler：為網頁元素新增一個事件。
- \$addHandlers：一次為網頁元素新增多個事件。
- \$removeHandler：為網頁元素移除一個事件。
- \$clearHandlers：為網頁元素移除所有個事件。

下面針對每個方法逐一詳細介紹其使用方式。

4.2.2 \$get 方法

網頁的每一個元素通常會對應一個 HTML 標籤，當 Javascript 要針對網頁元素進行各種操作時，必須先取得該網頁元素的實體參照。網頁元素的實體參照要由其名稱 (ID) 才能取得，而 \$get 方法可讓設計者輕鬆取得網頁元素的實體參照。

\$get 方法的語法為：

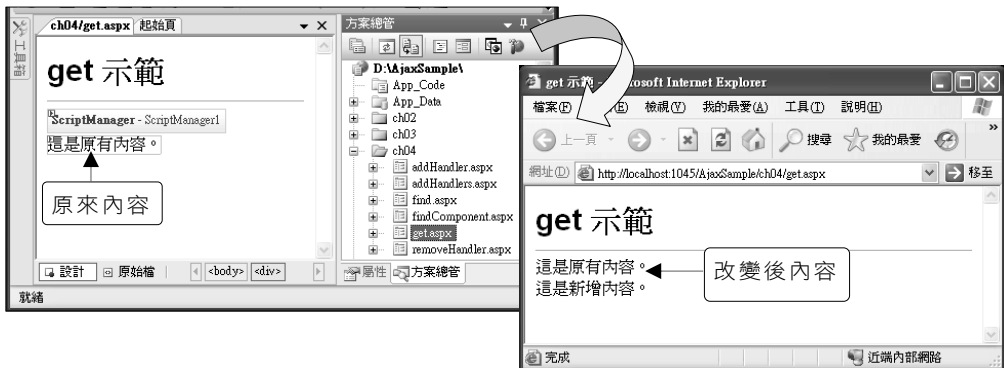
```
var 變數名稱 = $get(網頁元素名稱);
```

例如：網頁上有一個按鈕，其名為 **Button1** (`<input id="Button1" type="button" />`)，在 Javascript 中要以 `g_Button` 變數名稱取得此按鈕的參照，其語法為：

```
var g_Button = $get("Button1");
```

然後在網頁的其他地方就可用 `g_Button` 變數來操作此控制項。

下面的範例中示範如何使用 \$get 方法取得一個 **Label** 網頁元素，然後改變此網頁元素的內容。進入 VWDE，開啟 `<ch04>` 資料夾中的 `<get.aspx>` 網頁檔案，在 [設計] 視窗可見到 **Label** 控制項的內容為「這是原有內容」。按 **[Ctrl]+[F5]** 鍵執行網頁，於瀏覽器中見到 **Label** 控制項的內容除原有內容外，還增加了「這是新增內容」。



切換到 [原始檔] 視窗查看程式碼：

```
6 <head id="Head1" runat="server">
7 <title>get 示範</title>
8 <script language="JavaScript">
9     var g_Label;
10    function pageLoad()
11    {
12        g_Label = $get('Label1');
13        g_Label.innerHTML += "<br />這是新增內容。";
14    }
```

挑戰 Microsoft ASP.NET AJAX 1.0

```
15     </script>
16 </head>
17 <body>
18     <form id="form1" runat="server">
19         <DIV style="DISPLAY: inline; FONT-WEIGHT: bold; FONT-SIZE:
20 30px; FONT-FAMILY: Arial, Verdana">
21             get 示範</DIV>
22             <HR width="100%" SIZE="1">
23             <div>
24                 <asp:ScriptManager ID="ScriptManager1" runat="server">
25                 </asp:ScriptManager>
26                 <asp:Label ID="Label1" runat="server">這是原有內容。</asp:Label>
27             </div>
28     </form>
29 </body>
30 </html>
```

請注意：本書的主體就是非同步更新 (AJAX) 網頁設計，所以本書中所有範例網頁都具備非同步更新功能，由本章的範例開始，不再於範例中加入照片來檢視非同步更新效果。23 及 24 列的 **ScriptManager** 控制項是非同步更新網頁必定具備的元件，幾乎在每個網頁程式中都會出現，以後的程式說明也不再強調此控制項。

本網頁除了標題外，主要是 26 列的 **Label** 控制項，名稱為「Label1」，預設的內容為「這是原有內容。」，所以在 [設計] 視窗見到網頁內容即為此。

8 到 15 列的 Javascript 程式是網頁主要部分，12 列程式使用 \$get 方法取得 **Label1** 控制項實體並設定給 g_Label 變數。13 列程式利用 g_Label 變數來改變 **Label1** 控制項的內容：**Label** 控制項的內容儲存於 **innerHTML** 屬性中，此列程式是在原有內容中加入「
這是新增內容。」，所以會換行後再顯示「這是新增內容。」。因為 **pageLoad()** 函式會在網頁載入時就執行，所以會在網頁開啟就看到 **Label1** 控制項更改後的內容。

請讀者留意 26 列的 **Label** 控制項是伺服器控制項，而 8 到 15 列的 Javascript 程式則是瀏覽器端執行的程式，但此 Javascript 程式卻可以改變伺服器控制項的內容，而且是以非同步更新的方式更新 (此範例沒有看到網頁變動，下一小節範例就可看出其非同步更新的效果)，這是後端 ASP.NET AJAX 內建的 XMLHttpRequest 所達成。

4.2.3 \$addHandler 方法

如果要使網頁產生互動的效果，勢必要使用網頁元素的事件才能產生互動效果，否則僅是靜態顯示而已。\$addHandler 方法可為取得的網頁元素實體參照建立事件與其對應的執行函式。

\$addHandler 方法的語法為：

```
$addHandler(網頁元素實體參照, 事件名稱, 執行函式);
```

例如：網頁上的 **Button1** 按鈕的參照為 `g_Button` 變數，現在要為其新增一個 **Click** 事件，且按下該按鈕後執行 `onButton1` 函式，語法為：

```
$addHandler(g_Button, 'Click', onButton1);
```

現在改寫 4.2.2 的範例，加入 \$addHandler 方法使範例具有互動功能。開啟 <ch04> 資料夾中的 <addHandler.aspx> 網頁檔案，按 **[Ctrl]+[F5]** 鍵執行網頁，於瀏覽器中可見到 **Label** 控制項的內容為「這是原有內容」，按下 **[新增內容]** 鈕後，會在原有內容的下一列增加「這是新增內容」。



切換到 **[原始檔]** 視窗查看程式碼：

```
6 <head id="Head1" runat="server">
7 <title>addHandler 示範</title>
8 <script language="JavaScript">
9     var g_Label;
10    function pageLoad()
11    {
12        g_Label = $get('Label1');
```

挑戰 Microsoft ASP.NET AJAX 1.0

```
13             $addHandler ($get ('Button1'), 'click', onAddText);
14         }
15
16         function onAddText()
17         {
18             g_Label.innerHTML += "<br />這是新增內容。";
19         }
20     </script>
21 </head>
22 <body>
23     <form id="form1" runat="server">
24         .....
31     <asp:Label ID="Label1" runat="server">這是原有內容。</asp:Label><br
/><br />
32         <input id="Button1" type="button" value="新增內容" />
33     </form>
```

- 32 新建立一個名稱為 Button1 的按鈕，做為與使用者互動之用。
- 9 建立一個公用變數 g_Label。因為下面兩個函式都要使用此變數，所以放在函式外部成為公用變數。
- 10-14 網頁載入時就取得 Label1 網頁元素及設定按鈕事件。
- 12 取得 Label 網頁元素並設定給 g_Label 公用變數。
- 13 使用 \$addHandler 方法設定 Button1 按鈕的 **Click** 事件的執行函式為 onAddText，此函式在 16 到 20 列程式。
- 16-19 **onAddText** 函式程式碼。
- 18 在原有的 Label1 控制項內容中加入換行符號及「這是新增內容。」文字。

\$addHandler 方法常置於 **pageLoad** 事件函式中，如此可在網頁開啟時就完成控制項的事件設定，使該事件一直處於可用狀態。

4.2.4 AJAX 網頁中的按鈕

讀者是否注意到 4.2.3 節的範例中，**Label** 控制項是使用伺服器控制項，但按鈕則不是使用伺服器控制項 (asp:Button)，而是用一般 HTML 標籤，為什麼？按鈕是否可使用伺服器控制項？要解決這個問題，可修改 4.2.3 節的範例程式：將 32 列改為

```
<asp:Button ID="Button1" runat="server" Text="新增內容" />
```

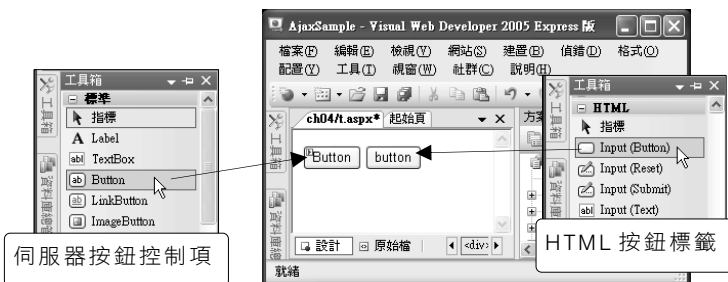
按 [Ctrl]+[F5] 鍵執行網頁，按下 [新增內容] 鈕後，請讀者仔細觀察，「這是新增內容」會顯示後立刻消失，瀏覽器下方出現網頁重整的圖示。



問題在於按鈕的伺服器控制項會自動 **PostBack**：當按下 [新增內容] 鈕後會執行自行建立的 **onAddText** 函式，所以讀者會見到「這是新增內容」的顯示；但隨即進行網頁重新載入，因此又回到網頁的初始狀態（只顯示「這是原有內容。」），如果使用者沒有仔細觀察網頁，會認為網頁沒有變動。

讀者在設計 AJAX 網頁時，要特別注意：按鈕不可使用伺服器控制項的按鈕（其程式碼為 `asp:Button`），要使用 HTML 標籤的按鈕（其程式碼為 `input type="button"`），除非您是刻意要使網頁進行 **PostBack** 動作。

使用 VWDE 設計網頁時，控制項的建立通常是使用拖曳的方式，伺服器按鈕控制項位於工具箱的「標準」群組，HTML 標籤的按鈕則位於工具箱的「HTML」群組。



4.2.5 \$find 方法

\$find 方法的功能與用法皆和 \$get 方法類似，主要被用來搜尋網頁中某個網頁元素是否存在。

\$find 方法的語法為：

```
var 變數名稱 = $find(網頁元素名稱, 父標籤名稱);
```

父標籤名稱即為該網頁元素的上一層標籤，是可有可無的參數。

例如：網頁上有一個按鈕，其名為 **Button1**，在 Javascript 中要以 f_Button 變數名稱取得此按鈕的參照，其語法為：

```
var f_Button = $find("Button1", this);  
或  
var f_Button = $find("Button1");
```

4.2.3 節的範例，可將 12 列程式改為

```
g_Label = $find('Label1',this);
```

其結果與原來完全相同，此修改完成的網頁檔案是位於 <ch04> 資料夾中的 < find.aspx> 檔，讀者可自行開啟執行。

\$find 方法的最主要功能是搜尋網頁中的特定元素是否存在，如果不存在會傳回 null 值。下面範例中，只有 **Label1** 控制項存在，**Label2** 控制項並不存在，我們用 \$find 方法檢查並顯示結果。下面範例實作利用 \$find 方法檢查控制項是否存在的方法： <ch04> 資料夾中的 < findComponent.aspx> 網頁檔案，按 **[Ctrl]+[F5]** 鍵執行網頁，按下 **[檢查 Label1 控制項]** 鈕後，按鈕上方會出現「Label1 控制項已經存在！」訊息；按下 **[檢查 Label2 控制項]** 鈕後，按鈕上方會出現「Label2 控制項不存在！」訊息。



切換到 [原始檔] 視窗查看程式碼：

```

6 <head id="Head1" runat="server">
7   <title>find 示範二</title>
8   <script language="JavaScript">
9     var f1, f2;
10    function pageLoad()
11    {
12      f1=$find('Label1', this);
13      f2=$find('Label2', this);
14      $addHandler($get('Button1'),'click',onCheckLabel1);
15      $addHandler($get('Button2'),'click',onCheckLabel2);
16    }
17
18    function onCheckLabel1()
19    {
20      if (f1==null)
21        Label1.innerHTML = 'Label1 控制項不存在！';
22      else
23        Label1.innerHTML = 'Label1 控制項已經存在！';
24    }
25
26    function onCheckLabel2()
27    {
28      if (f2==null)
29        Label1.innerHTML = 'Label2 控制項不存在！';
30      else

```

```
31         Label1.innerHTML = 'Label2 控制項已經存在!';
32     }
33
34     </script>
35 </head>
36 <body>
37     <form id="form1" runat="server">
38         .....
45         <asp:Label ID="Label1" runat="server">本控制項名為 Label1。
</asp:Label><br /><br />
46         <input id="Button1" type="button" value="檢查 Label1 控制項" />
47         <input id="Button2" type="button" value="檢查 Label2 控制項" />
48     </form>
```

- 45 建立 **Label1** 控制項，**Label2** 控制項不存在。
- 46-47 建立兩個檢查控制項按鈕。
- 9 設定 `f1` 及 `f2` 為公用變數。
- 10-16 網頁開啟時建立檢查按鈕的執行函式事件。
- 12 尋找 **Label1** 控制項。
- 13 尋找 **Label2** 控制項。
- 14 設定 **Button1** 按鈕 (檢查 **Label1** 控制項) **Click** 事件的處理函式為 `onCheckLabel1`。
- 15 設定 **Button2** 按鈕 (檢查 **Label2** 控制項) **Click** 事件的處理函式為 `onCheckLabel2`。
- 18-24 檢查 **Label1** 控制項是否存在的函式。
- 20-21 `f1` 為檢查 **Label1** 控制項後的傳回值，如果其值為 `null`，表示 **Label1** 控制項不存在，然後顯示「**Label1** 控制項不存在」的訊息。
- 22-23 如果 `f1` 的值不為 `null`，表示 **Label1** 控制項存在，於是顯示「**Label1** 控制項已經存在」的訊息。
- 26-32 檢查 **Label2** 控制項是否存在的函式，其運作模式與 `onCheckLabel1` 函式完全相同，不再重複。

4.2.6 \$removeHandler 方法

`$removeHandler` 方法剛好執行 `$addHandler` 方法相反的動作，可將 `$addHandler` 方法所加入的事件移除。`$removeHandler` 方法的語法及使用方式都與 `$addHandler` 方法相同，讀者可輕易使用。

\$removeHandler 方法的語法為：

```
$removeHandler(網頁元素實體參照, 事件名稱, 執行函式);
```

例如：網頁上的 Button1 按鈕的參照為 g_Button 變數，之前已用 addHandler 方法設定其 Click 事件的執行函式為 onButton1 函式，使用 removeHandler 方法將此 Click 事件移除的語法為：

```
$removeHandler(g_Button, 'Click', onButton1);
```

有了 addHandler 及 removeHandler 方法，網頁設計者可以在任意時間對網頁元素加入執行事件，也可以隨時讓這些事件失去作用，將使網頁設計的彈性大為增加。

下面的範例，[切換顯示] 按鈕可以切換文字的顯示模式，即原來有顯示文字，按鈕後文字會消失；若原來沒有顯示文字，按鈕後會顯示文字。另外設計了兩個按鈕，一個會使 [切換顯示] 按鈕發生效用，另一個會使按鈕失去效用。開啟 <ch04> 資料夾中的 <removeHandler.aspx> 網頁檔案，按 [Ctrl]+[F5] 鍵執行網頁，於瀏覽器中按下 [切換顯示] 鈕並沒有作用。



先按 [使上面按鈕有效] 鈕，再按 [切換顯示] 鈕則按鈕上方的文字會消失，再按一次 [切換顯示] 鈕則文字會再出現，表示 [切換顯示] 鈕的確有作用。



挑戰 Microsoft ASP.NET AJAX 1.0

按 [使上面按鈕無效] 鈕，再按 [切換顯示] 鈕發現網沒有任何變化，再按幾次依然如此，表示 [切換顯示] 鈕已經失去作用。



切換到 [原始檔] 視窗查看程式碼：

```
6 <head id="Head1" runat="server">
7   <title>removeHandler 示範</title>
8   <script language="JavaScript">
9     var g_Label;
10    function pageLoad()
11    {
12      g_Label = $get('Label1');
13      $addHandler($get('Button1'),'click',onButtonOK);
14      $addHandler($get('Button2'),'click',onButtonCancel);
15      $get('Button2').disabled=true;
16    }
17
18    function onButtonOK()
19    {
20      $addHandler($get('btnToggle'),'click',onShowToggle);
21      $get('Button1').disabled=true;
22      $get('Button2').disabled=false;
23    }
24
25    function onButtonCancel()
26    {
27      $removeHandler($get('btnToggle'),'click',onShowToggle);
28      $get('Button1').disabled=false;
```

```

29         $get('Button2').disabled=true;
30     }
31
32     function onShowToggle()
33     {
34         if(g_Label.innerHTML=="")
35             g_Label.innerHTML="按下面按鈕可切換顯示狀態。";
36         else
37             g_Label.innerHTML="";
38     }
39     </script>
40 </head>
41 <body>
42     <form id="form1" runat="server">
43     .....
44
45     <asp:Label ID="Label1" runat="server">按下面按鈕可切換顯示狀態。
46 </asp:Label><br /><br />
47     <input id="btnToggle" type="button" value="切換顯示" />
48         <HR width="100%" SIZE="1">
49     <input id="Button1" type="button" value="使上面按鈕有效" />
50     <input id="Button2" type="button" value="使上面按鈕無效" />
51 </form>

```

- 51 建立 [切換顯示] 按鈕。
- 53-54 建立使 [切換顯示] 按鈕有效及失效的兩個按鈕。
- 10-16 網頁開啟時建立使按鈕生效及失效的執行函式事件。
- 12 取得 Label1 控制項實體。
- 13 設定 Button1 按鈕 (使按鈕生效) Click 事件的處理函式為 onButtonOK。
- 14 設定 Button2 按鈕 (使按鈕無效) Click 事件的處理函式為 onButtonCancel。
- 15 網頁開啟時設定 Button2 按鈕無法使用。
- 18-23 使按鈕生效的函式。
- 20 利用 \$addHandler 方法設定 btnToggle 按鈕的 Click 事件為 onShowToggle 函式，也就是按下 btnToggle 按鈕會執行 onShowToggle 函式，所以使 btnToggle 按鈕有作用。
- 21-22 設定 Button1 按鈕無法使用而 Button2 按鈕可以使用。
- 25-30 使按鈕失效的函式。
- 27 利用 \$removeHandler 方法將 btnToggle 按鈕的 Click 事件移除，所以 btnToggle 按鈕不再有作用。
- 28-29 設定 Button1 按鈕可以使用而 Button2 按鈕無法使用。