2010-01-25

[入門][Regex] Regular Expression 詳論

ASP.NET 的初學者，一旦學到了 Validator 控制項，就會踫到 Regular Expression 這個主題。Regular Expression 指的是一種檢驗文數字的表示式，早在 ASP.NET 出現之前就已存在，而且也不是 ASP.NET 所專有。大多數主流程式語言（包括 Java、PHP 等）都可以透過 Regular Expression 來驗證輸入...

ASP.NET 的初學者, 一旦學到了 Validator 控制項, 就會踫到 Regular Expression 這個主題。Regular Expression 指的是一種檢驗文數字的表示式, 早在 ASP.NET 出現之前就已存在, 而且也不是 ASP.NET 所專有。大多數主流程式語言 (包括 Java、JavaScript、Ruby、PHP 等) 都可以透過 Regular Expression 來驗證輸入。

不過, 除非你早就了解 Regular Expression 是什麼, 否則當你一開始遇上它時, 可能會被搞得一頭霧水。

Regular Expression 到底是什麼?

簡單的說, 所謂的 Regular Expression (正規表示式, 經常被簡寫為 Regex 或是 RegExp) 的最主要目的, 在於使用一組特定的表示式, 來驗證一段字串是否符合這個樣式 (Pattern)。舉例來說, 當你希望使用者輸入他的 Email 位址時, 你要使用什麼方法讓程式來判斷使用者確實輸入正確的格式, 而沒有輸入亂七八糟的東西呢? 又例如, 如果你需要使用者輸入他的電話號碼, 你又如何確定他輸入了正確的樣式 (像 02-12345678), 而不是隨心所欲的亂敲一通?

當然, 你不能期望 Regular Expression 能幫你篩選類似故意假造或亂寫的電子郵件或電話號碼, 你只能使用它來防止使用者因為沒有依循你所設定的「樣式」來輸入資料。以電話號碼為例, 如果你規定輸入的樣式是 02-12345678 或 07-1234567, 那麼以下的數字樣式都是錯的：

- 0212345678 (少了一橫槓)
- 12345678 (少了區域碼)
- (02)12345678 (使用括弧而非橫槓)
- 02-1234-5678 (多了一個橫槓)
- +886-2-1234-5678 (並非你指定的格式)
- 206-1234-5678 (非本國電話格式)

如果你要撰寫程式來過濾以上的問題, 恐怕要下不少判斷式, 而且不見得能把所有的錯誤情況全部掌握。但是如果使用 Regular Expression , 再配合 Validators, 通常都可以輕鬆的解決。

曾經有人告訴我, 輸入樣式不對, 有什麼關係？反正這些電話號碼都是給人看的, 電腦不需要理解, 人能理解就好啦！

我的回答是「話是沒錯, 但是你確定你的電腦系統永遠都不需要能夠正確的判斷這些電話號碼嗎？」如果你的資料庫中每一筆電話格式都是正確的、一致的, 那麼我只要寫幾行程式, 甚至不用寫程式, 就可以做到以下幾件事情：

- 把這些電話與它的區域碼分離出來。如此也可以據以判斷使用者的區域分佈。
- 能使用更有效的方法, 以程式判斷使用者是不是輸入了重複的號碼。

- 可以將號碼進行正確的排序。
- 未來如果遭遇任何電話系統的更改 (例如將所有人的電話號碼從七碼改成八碼, 或在前面加上 2; 這不是沒發生過), 都有辦法寫程式對整個資料庫內的所有號碼進行變更, 無需使用人工作業一筆一筆過濾處理。
- 同樣的, 未來可以視需要, 將全部號碼進行變更, 例如將 02 改成 +886-2。
- 方便以程式方法進行其它處理。

這些事情都是幾乎不需要代價的舉手之勞, 唯一的前題是, 你只需要把 Validator 放在 TextBox 後面, 並餵給它正確的 Regular Expression 就行了。為求萬全, 我們最好在伺服器端同樣使用 Regular Expression 再多做一次驗證。

Regex 不只使用在 ASP.NET 的 RegularExpressionValidator 控制項而已; 事實上它可以廣泛地運用在許多其它方面。把 Regex 學會、學好, 對你有很大的幫助。

Regex 所使用的符號彙整

那麼 Regular Expression 是如何表示的呢？我們先來看看 Regex 的 Notation 列表：

記號	說明
字元	代表該字元, 例如輸入 a 就代表那個地方應該出現 a 這個字元。
^	限制字串必須出現於行首(但是在某些語言中會受單行/多行模式所影響), 例如 ^a 表這串字必須以 a 開頭; 如果 a 出現在其它地方, 都不算數。
$	限制字串必須出現於行末(但是在某些語言中會受單行/多行模式所影響), 例如 a$ 表這串字必須以 a 結尾; 如果 a 出現在其它地方, 都不算數。請特別注意 ^ 和 $ 的使用時機; 使用 [0-9]+ 和使用 ^[0-9]+$ 可能檢測出極為不同的結果。
\A	限制字串必須出現於整個字串的最前面, 這不受多行模式所影響
\z	限制字串必須出現於整個字串的最後面, 這不受多行模式所影響
\	將特殊字元還原成字面意義的字元, 我們說這種動作叫做「逸出」(Escape), 例如 \(代表 (這個符號, \\ 代表 \ 這個符號; 這種表示法適用於 (,), [,] 等等在 Regex 式子中具有特殊意義的字元。
^	某字元以外的任何字元, 必須包在中括號裡面。例如 [^a] 表示 a 除外的任何字元或符號, [^a\t] 表示 a 和 tab 除外的任何字元或符號。
-	字元集合中可使用 - 來指定字元的區間, 必須包在中括號裡面。例如 [a-z] 表示從 a 到 z 的英文小寫字元, [1-3] 表示從 1 到 3 這三個數字之一。
+	其前的字元或字元集合出現一次或一次以上, 例如 a+。
?	其前的字元或字元集合可出現一次或不出現, 例如 a?。
*	其前的字元或字元集合可出現任何次數或不出現, 例如 a*。
(...)	用以括住一群字元,且將之視成一個集合, 通常用來集合表示多個檢核式。

{n}	重複 n 次。
{n,m}	重複 n 到 m 次。
{n,}	至少重複 n 次。
[]	其中之一字元可出現可不出現, 例如 [abc] 表示不論出現 a 或 b 或 c 都算符合。
[-]	.Net 獨有的集合內排除表示式, 例如 [[0-9]-[135]] 表示在 0 ~ 9 這十個字元中排除 1, 3 與 5 這三個字元
|	代表「或」, 例如 (Sun|Mon|Tue|Wed|Thu|Fri|Sat), (日|一|二|三|四|五|六); 必須以左右括號括住。
. (句點符號)	代表除了換行符號 (\n) 以外的任一字元。如果要包括換行符號, 請使用 [\s\S]。
\w (\W)	代表任何英文(以外的) 字元 - 請注意, 數字字元也被承認。
\s (\S)	代表空白 (以外的) 字元。
\d (\D)	代表數字 (以外的) 字元, 在 .Net 中也包括中文的全形數字 (其它語言如 Ruby 則不一定)。如果你只想檢測 0~9 這十個字元, 你應該使用 [0-9] 而不是 \d (\D)。不過為了節省篇幅, 本文中一律使用 \d(\D)。
\b (\B)	代表位於文字邊界的 (以外的) 字元, 例如 \bA 可以檢核出 AB, A\b 可以檢核出 BA, \bAA\b 可以檢核出 AA。
\a	代表 Bell 字元。可以以 \u0007 取代。
\v	代表 Vertical Tab 字元。可以以 \u000B 取代。
\b	代表倒退字元 (Backspace); 不要跟上面的 \b 搞混。雖然這個字元很少用到, 但如果真的要使用的話, 建議以 [\b] 取代(用中括號包住), 或者以 \u0008 取代。
\r	代表換行字元 (或稱 CR, Carriage Return)。
\n	代表換行字元 (或稱 LF, Line Feed; 通常和 \r 一同出現, 所以一般以 \r\n 代表換行, 但根據我的測試, 無論使用 \r 或 \n 或 \r\n 都會得到相同的結果, 但唯獨不能寫成 \n\r, 但建議使用 \r?\n)。
\t	代表 TAB 字元 (或稱 HT, Horizontal Tab)。
\(	代表左括號。
\)	代表右括號。
\nn 或 \nnn	代表以八進位方式表示的字元。例如 \101 等於英文字母 a。
\xnn	以十六進位字元碼代表某個字元; 例如 [\x21-\x7E] 可代表所有看得到的字元 ([\x20-\x7E] 則包括空白字元)。不過注意 \x 之後要使用兩個數字, 不足兩個數字者請補 0, 例如 \x01
\unnnn	代表以四個數字的 Unicode 所表示的字元。例如 \u6587 等於中文的「文」。在 .Net 下可以透過 Encoding.Unicode.GetBytes 方法將中文文

	字轉成 Unicode。
\1, \2...	(Backreference Constructs) 表示出現過的群組; 例如 "(\d)(\D)" 樣式中有兩個群組, 若使用 "(\d)(\D)\1" 可檢出 "2A2"; 若使用 "(\d)(\D)\2+" 則可檢出 "2AA"; 餘此類推。請注意, 反向參考所參考的是實際的字元, 而不是樣式本身。所以第一個樣式無法檢出 "2A3", 第二個無法檢出 "2AB"。
\k<name>	同上, 但適用於命名的群組; 例如 "(?<Digit>\d)(?<NonDigit>\D)\k<Digit>" 亦可檢出 "2A2"。同樣的, 它反向參考的是實際字元而不是樣式本身。
\p{Lu} (\P{Lu})	檢出大寫(非大寫)的字母, 例如 (?-i:\p{Lu}) 可檢出字串中所有大寫字母, 而 (?-i:\P{Lu}) 可檢出所有非大寫 (包括數字、空白等) 的字母。
\p{IsCJKUnifiedIdeographs}	檢出中文文字。而且, 雖然代表字中有 CJK, 其實無法檢出日文或韓文。
$n	不使用於 Regex 樣式中, 而是用於字串的取代 (Regex.Replace() 方法)。$1 代表第一個 Matched Groups, $2 代表第二個, 依此類推。
${name}	不使用於 Regex 樣式中, 而是用於字串的取代 (Regex.Replace() 方法)。name 代表具名群組的群組名稱。

註: 各種語言或有不同的語法或習慣 (大致上都是一樣的, 彼此間差異很小); 在本文中, 除非特別指出, 否則均採 .Net (特別是 C#) 語法。

至於更詳細的用法, 你可以使用 Google 搜尋以找到更多的相關資料, 或者直接參考 MSDN 上的說明。

在正式解說 Regular Expression 之前, 我要先介紹 Regex 的工具。因為 Regex 有點複雜, 你再聰明, 恐怕也不能光是看就把它學好。你必須重複的練習, 直到你對它完全熟悉為止。

若想節省時間, 你可以直接在 RegExLib.com 作練習。先進入網頁, 把畫面往下捲一點, 在 Source 方塊中打進測試文字(例如 abc), 然後在稍下方的 Pattern 方塊中打進你自己定的 Regex 樣式(例如 \w{3}), 按 Submit 按鈕, 再稍等一下子, 在畫面的最下面就會出現 Match 或是 No match, 表示正確或是錯誤。

此外, Rubular 也是不錯的線上測試網站。這個網站可以把你的輸入資料、樣式和結果存成一個連結, 以方便你放在部落格中, 或者與其他人分享。

Expresso 也是一套既強大、知名又好用的 Regex 測試工具, 不過它是單機版, 需要安裝。

當然, 最方便的測試工具就是 Visual Studio (2012 或以後版本) 了! 雖然它沒有上述工具提供的那麼多特殊功能, 但是它讓你能在撰寫程式的同時測試你的 Regex 樣式, 也可以拿來修改你的程式。它的搜尋功能 (Ctrl-F) 可以讓你使用 Regex 樣式來搜尋文字, 它的取代功能 (Ctrl-H) 可以輸入 $0, $1, $2 等樣式, 所以事實上是很強大的。

除了 Visual Studio, Sublime Text 3 也是我個人蠻推薦的一套文字編輯工具。它以 Regex 樣式做搜尋和取代的功能和 Visual Studio 幾乎一模一樣 (連快速鍵都一樣), 但是 Sublime Text 允許你搜尋/取代換行字元 (按下 Ctrl-Enter 即可輸入, 或者在 Regex 樣式裡使用 "\r\n"), 而且輸入框還更大一點。所以如果你需要以 Regex 樣式做較大量文字的取代工作時, 我推薦 Sublime Text 3。每次我想把部落格中一大堆亂七八糟的 span 區段快速清除的時候, 我找不到比它更好用的工具了。它也有 Mac 和 Linux 版本。

Sublime Text 也支援一些其它工具不見得提供的特殊功能。例如你可以按下 Ctrl-H 以進入搜尋與取代工具列, 假設你在 "Find What:" 文字框中輸入 "[a-z]+" (不含引號), 它會找到所有小寫的英文字 (記得先按下左下角的「.*」按鈕以啟動 Regex 搜尋); 然後在 "Replace With:" 文字框中輸入 "\u$0", 然後按下 Replace All 按鈕, 那麼它會把所有搜尋到的文字改成首字大寫。同理, 它支援下列幾種功能:

1. \U - 全部改成大寫 (abc 變成 ABC)
2. \L - 全部改成小寫 (ABC 變成 abc)
3. \u - 改成首字大寫 (abc 變成 Abc)
4. \l - 改成首字小寫 (ABC 變成 aBC)

我不確定除了 Sublime Text 之外還有哪些編輯程式支援這功能, 但是我確定 Visual Studio 不支援。

不過, 最快而且方便的方式, 就是使用在本文下一段裡嵌入的那個 jsFiddle 程式, 你可以在閱讀本文的同時進行練習。因為在本文載入後, 那個 jsFidele 也已經載入, 它的 Regex 執行又是在你的機器上執行, 所以速度很快。如果你想要在閱讀本文時練習 Regex 的通用樣式, 使用這個 jsFiddle 是最適合的。

自己撰寫測試程式

在 RegExLib 或 Rubular 等線上檢測網站中進行測試固然是個方便的做法, 但畢竟那是別人的網站, 每次進行測試都要浪費一點點時間。為什麼我們不能自己寫一個專門用來測試的網頁或應用程式?

以下就是一個僅用來進行 Regex 樣式測試的 ASP.NET 網頁:

.aspx

```
<%@ Page ... ValidateRequest="false" %>
...
<p>
    Input:
    <asp:TextBox ID="txtInput" runat="server" />
</p>
<p>
    Pattern:
    <asp:TextBox ID="txtPattern" runat="server" />
</p>
<p>
    <asp:Button ID="Button1" runat="server" Text="Submit" onclick="Button1_Click" />
</p>
<p>
    <asp:Label ID="lb" runat="server"></asp:Label>
</p>
```

.aspx.cs

```
using System.Text.RegularExpressions;
```

```
...
protected void Button1_Click(object sender, EventArgs e)
{
    string pattern = txtPattern.Text;
    string input = txtInput.Text;

    RegexOptions opt = new RegexOptions();
    opt = RegexOptions.IgnoreCase | RegexOptions.Multiline;
    Regex reg;

    try
    {
        reg = new Regex(pattern, opt);
        lb.Text = "Found "+ reg.Matches(input).Count.ToString() + " match(es).<hr />"
        for (int i = 0; i < reg.Matches(input).Count ; i++)
        {
            Match match = reg.Matches(input)[i];
            lb.Text += "Match[" + i.ToString() + "] as the following -<b r />";
            for (int j = 0; j < match.Groups.Count; j++)
            {
                lb.Text += "Group[" + j.ToString() + "] = " +
                           match.Groups[j].ToString() + "<b r />";
            }
            lb.Text += "<hr />";
        }
    }
    catch (Exception ex)
    {
        lb.Text = "Error: " + ex.Message;
    }
}
```

在此網頁中我們可以輸入受測字串以及樣式, 按下 SUBMIT 按鈕即可觀察其結果了。至於程式的邏輯, 我在下文中會陸續解釋。

提醒一下, 如果你使用 ASP.NET 4.0 以上, 你還必須在 web.config 中加上一行

<httpRuntime requestValidationMode="2.0" />

否則你可能無法順利輸入樣式。細節可參考「在 VS2010 Web 專案下遭遇 potentially dangerous 錯誤的問題」這篇文章。

但是如果你真的很急的話, 那麼可以直接使用我寫的一個 Regex 樣式檢測工具 (以下均簡稱為「樣式檢測工具」)。

以下是一張使用範例圖:

但是容我強調一下, 上面這個工具使用的是你的瀏覽器所提供的 JavaScript 的 Regex 引擎! 換句話說, 本文中講述的是 C# 中的 Regex, 和上述工具有小部份的語法及功能差異。雖然兩者在核心功能上是差不多的, 但是並不是百分之百相容。上面介紹過的 Rubular (使用 Ruby 的 Regex 引擎) 和 Sublime Text (使用 Python 的 Regex 引擎) 都有類似的情況, 大家在使用時請記得這一點。

暖身練習

現在, 我們拿前面所舉的台灣電話號碼的格式為例, 來做一下練習。請在樣式檢測工具中的 Pattern 方塊中輸入以下文字：

```
0\d{1,2}-\d{6,8}
```

你可以隨時回頭參考最上方的 Regex Notation 列表。不過我還是簡單的解說一下上面這個式子:

1. 開頭的 0, 就是固定的 0
2. \d 表示一個數字 (Digital), 而 {1,2} 表示 1 到 2 個
3. - (減號) 代表一個減號
4. \d{6,8} 表示 6 到 8 個數字

換句話說, 上述的式子就是要拿來驗證一串數字是否符合像 02-12345678 這樣的格式。

接著, 依序將下面的字串輸入 Input 方塊：

- 02-12345678
- 07-1234567
- 035-123456
- 0212345678
- 12345678
- (02)12345678
- 02-1234-5678
- +886-2-1234-5678
- 02-12345a

輸入後, 馬上可觀察到結果。

我們可以發現, 只有前面三個受測字串是 Match 的, 後面則通通是 No match 的。你不妨再多做幾個測試, 才能證明 0\d{1,2}-\d{6,8} 這個樣式是確實可用的。不過, 除了 0\d{1,2}-\d{6,8} 之外, 有沒有更好的樣式呢？你可以再回頭參考一下我所列的 Notation 列表, 再好好想一想。

然而, 如果你把樣式改成 0\w{1,2}-\w{6,8} 的話, 你會發現那幾個電話號碼也可以檢測出來。你再參考一下 Notation 列表, 想想為什麼。但是這麼一來, 像 0A-abcdEFGH 這種字串也能通過檢測了。所以把 \d 改成 \w 是不行的。

此外, 請注意如果你在輸入字串中不小心打入全形數字的話 (例如 02-１２３４５６７８; 使用者很容易發生這種錯誤), 你在上面的樣式檢測工具中會檢測不出來, 但是你在 .Net 中 (包括在 Visual Studio 的 IDE 裡) 卻能檢測出來。之所以會發生這種情況, 主要是因為我寫的樣式檢測工具是採用 JavaScript 引擎, 和 .Net 引擎在 Unicode 的支援情況不一樣的關係。換句話說, 這種情況會造成網頁前端 (使用 JavaScript 的 Regular Expression 引擎) 和後端 (使用 .Net 的 Regular Expression 引擎) 在驗證時出現不一致的現象。因此, 請特別留意, 當你要驗證數字時, 最好使用 "[0-9]" 來代表而不是 "\d", 即使你看到許多文章 (包括本文) 中多半會使用 "\d" 做範例。

接著, 我交給你一個習題, 那就是請你自己練習製作一個行動電話專用的樣式。這個應該很簡單。

如果行動電話號碼能用的樣式也想出來之後, 你不妨再想想, 有沒有辦法製作一個樣式, 是既能驗證普通電話, 又能驗證行動電話的？這並不難, 難的是你能不能運用手頭上的工具, 自己把它做出來。

最後, 如果你時常為某種特定用途的樣式想到頭都快破了, 那麼我要告訴你一個好消息, 就是有時候你不用想得那麼辛苦, 因為有人已經都幫你想好了。你可以在 RegExLib.com 網站上, 找到許多現成的樣式來用, 如此將能你節省非常多的時間。不過雖然這是個好消息, 我還是要稍為潑你一下冷水。在那個網站上能找到的樣式, 不見得通通符合我們在台灣的情況。所以自己要有製作 Regex 樣式的能力, 這才是最重要的。

設計 Regex 樣式的思考方向

你應該怎樣思考 Regex 的設計方式? 你不會永遠只需要驗證像 02-12345678 這麼簡單的字串。面對較困難 (例如前面舉的習題, 即同時驗證市話和手機號碼) 和真的困難 (例如驗證 HTML 標籤屬性) 的樣式, 你到底應該如何進行思考?

以下我列出三個設計 Regex 樣式的基本原則, 做為你的思考出發點:

1. 某些字一定會出現 (Must appear)
2. 某些字一定不會出現 (Must not appear)
3. 某些字可能出現、可能不出現, 以及出現幾次 (May appear, and how many times)

若以括號對 (pair) 為例, 當這種情況出現時, 我們通常會以 "\([^)]+\)" 來取出像 "(ABC)" 這樣的受測字串。左邊的左括號和右邊的右括號是 Must appear, 而中間的部份則一定不會有右括號出現 (Must not appear), 我們再以 "+" 取出至少有一個在內的字串 (May appear, 出現一次以上)。如果你希望連 "()" 這樣的字串都檢測出來, 那麼你應該把 "+" 改成 "*" (出現零次以上)。

再以前面的本國電話號碼為例。最前面的 0 是一定會出現的。跟在 0 後面的一或兩個數字也是一定會出現的。然後一個減號也是一定會出現的。

接著, 後面跟著六到八個數字, 也是一定會出現的 (否則那就不是合理的電話號碼)。

最後, 就得到我們的 "0\d{1,2}-\d{6,8}" 這個樣式。

但是事情不可能那麼簡單。當你繼續思考下去, 你永遠會發現更多改善的空間。所以, 如果你希望你設計的樣式朝完美的方向改善, 你有兩件事要做:

1. 盡量收集對的受測字串 (正面表列)
2. 盡量收集錯的受測字串 (負面表列)

就像我在上一段中列出的九個受測字串; 事實上你應該列出更多的受測字串, 直到你覺得確實足夠為止。然後, 我們說這些受測字串就是這次 Regex Pattern 的 Scope。如果你不先訂出 Scope, 那麼即使一個真正簡單的 Regex 都可以搞到相當發散而難以控制, 何況是複雜的 Regex。想像一下, 光是一個電話號碼格式, 如果任由無止盡的發想, 你的樣式將必需支援多少需求:

1. 如果使用者輸入全形數字怎麼辦?
2. 如果使用者的輸入夾雜著一部份全形數字怎麼辦?
3. 如果使用者把 l (小寫的 L) 當做 1 (或者 O 當作 0) 來輸入怎麼辦?
4. 如果使用者輸入英文的數字 (例如 One) 怎麼辦?
5. 如果使用者輸入羅馬數字 (例如 I, II, III, IV, V, VI...) 和國字大寫 (例如三五七捌玖) 怎麼辦?
6. 如果使用者的電話在外國怎麼辦?
7. ...

我曾經在國外的論壇與人爭辯, 因為對方堅持說非得要使用 [0-9] 來取代 \d 不可。但是真是如此嗎? 在 Regex 中許多語言為什麼讓 \d 支援所謂 Culture-Variant 的數字呢? 不正是因為有那個必要嗎? 就像支援 Unicode 的 Regex 引擎 (例如 .Net) 可以辨認出全形的中文符號以做為邊界區隔。我們不能說那是錯的。

你的樣式應該怎麼寫, 端看你允許或不允許哪些輸入而定。你大可以允許使用者輸入全形或者大寫數字, 因為或許你的程式會自動把全形或大寫數字轉換成半形數字。所以 Regex 樣式沒有「最完美」的, 一切都要看你支援的需求是什麼而訂。

此外, 當你制訂好初步的樣式之後, 你可以再慢慢予以改善。例如, 再仔細想想, 在前面的樣式中, 開頭的 "0\d{1,2}-" 是最好的寫法嗎? 事實上, 如果我們把全國的電話區碼列出來看, 我們並看不到有 "01-" 這種開頭的數字, 所以我們大可以把它改成 "0[2-8]{1,2}-"。只不過, 如此一來, 使用者仍然可能輸入像 "022-" 這種錯誤的區碼。

還有, 我們要不要順便也支援一下馬祖 (0836) 和烏坵 (0826) 的電話格式呢?

如果再仔細想想, 其實全國的電話區碼都是固定的, 所以我們何不乾脆把它改成 "(02|03|037|04|049|05|06|07|08|082|0826|0836|089)-" 算了。如此一來, 使用者幾乎沒有輸入錯誤的可能。

到這裡為止, 我們已經把樣式改成 "(02|03|037|04|049|05|06|07|08|082|0826|0836|089)-[0-9]{5,8}", 這已經跟一開始的樣式有頗大差異了。原則上, 你的樣式應該長什麼樣子, 端看你希望接受哪些電話號碼格式、不接受哪些格式, 並沒有所謂最好的樣式存在。把你的樣本設計好, 再來設計你的樣式, 這才是最佳的策略。

.Net下的正規表示式

在 .Net 中正規表示式的相關功能都在 Regex 命名空間之下。請看以下的簡單範例:

```
string patHtml = @"<([\s\S]+)?>";
Regex regHtml = new Regex(patHtml );
Match mHtml = regHtml.Match(txtUserInput.Text);
if (mHtml.Success)
   lbMessage.Tex = "請勿輸入 HTML 標籤!";
```

以上這個簡單的程式可以檢測使用者是否企圖輸入疑似 HTML 標籤並予以警告。

以下這個程式可以直接將所有疑似 HTML 標籤全部消去:

```
string patHtml = @"<([\s\S]+)?>";
txtInput.Text = Regex.Replace(txtInput.Text, patHtml, string.Empty, RegexOptions.Multiline);
```

以下這個程式則是將所有輸入的疑似 HTML 列出來(請輸入幾個 HTML 標籤, 同時在 HTML 標籤之間插入一些普通的文字):

```
string patHtml = @"(<[\s\S]+?>)";
Regex regHtml = new Regex(patHtml);
foreach (Match m in regHtml.Matches(txtInput.Text))
   lbMessage.Text += m.Groups[0].Value.Replace("<", "&lt;").Replace(">","&gt;") + ", ";
```

在下面程式中, 我們使用 Regex 的 Match 方法取出使用 Regex 找出的符合字樣。Match.Groups[0] 可以取出整個樣式 (在上例中就是整個 HTML 標籤 (包括左右兩個角括號)); 如果在 Pattern 中我們使用了小括號括住部份樣式, 則這個部份樣式就叫做一個群組 (Group)。我們可以以 Match.Groups[1]、Match.Groups[2]、Match.Groups[3] 逐項取出個別值。這是非常好用的功能; 即使並未找到符合的字串, 它只會傳回空字串, 而不會引發超出索引界限的例外, 所以可以大膽的使用。我們不妨來試試:

```
string patHtml = @"<([\s|\S]+)?>";
Regex regHtml = new Regex(patHtml);
foreach (Match m in regHtml.Matches(txtInput.Text))
   lbMessage.Text += m.Groups[1].Value.Replace("<", "&lt;").Replace(">","&gt;") + ", ";
```

請注意我在這個程式中所做的小修改, 再比對其輸出結果, 你就可以了解這個程式所代表的意義。

標示並取出特定的群組

此外, .Net 提供了一個很方便的功能, 就是可以為特定的群組取名, 以利後續操作。以上面的程式為例, 我們如果使用 Match.Groups[*n*] 來取出符合的字串, 那麼每次我們變更了正規表示式的樣式, 這個號碼就可能隨之異動。如果我們能將它取名, 那麼不管我們怎麼變動表示式, 都不至於引用到錯誤的符合字串。

要使用這個功能, 我們可以將表示式標示為如下的樣子:

(?<*name*>*pattern*)

我把上一個範例稍加修改如下:

```
string patHtml = @"<(?<Tag>[\s\S]+)?>";
Regex regHtml = new Regex(patHtml);
foreach (Match m in regHtml.Matches(txtInput.Text))
    lbMessage.Text += m.Groups["Tag"].Value.Replace("<", "&lt;").Replace(">","&gt;") + ", ";
```

將正規表示式的片段予以命名之後, 我們就可以使用名稱來取得符合字串, 如上例所示。

額外說明一下。我在這個例子中使用 "<([\s\S]+)?>" 來檢核 HTML 標籤, 這是為了舉例方便而已。實際上, 一般用來檢核真正的 HTML 標籤的做法並不是這樣的。嚴謹的做法, 你除了找出對應的角括號樣式之外, 你應該再把內容送到某個 parser 去解析, 而這個 parser 應該維持一份符合 W3C 標準的 HTML 標籤字典以判斷該標籤是否合法; 接下來再解析出標籤的其它屬性。在上述範例中, 你固然可以檢核出所有 HTML 標籤, 但是它並無法判斷像 <ENTER> 或 <1> 這種看似 HTML 標籤但實際上並不是的文字。

關於 Regex 的群組檢測 (亦即在括號中以問號開頭的表示法), 還有許多其它進階的功能, 詳細說明可參考 MSDN。對於複雜的情況, 我們免不了要用到這些進階功能; 所以當你能夠充分領略 Regex 初步功能之後, 一定會用得上這些進階功能。不過, 如果你還沒有 Regex 的基礎, 這些進階功能只會加深你的負擔, 所以請量力而為。

由於這個部份過於複雜而不適合放在入門文章中, 我已經把它另外放到「[Regex] 進階群組建構」一文中, 有興趣者請自行參考。

簡單範例

以下我把幾個我認為有用的範例貼在下面; 讀者可以直接拿去使用, 或者作為參考 -

1. 排除所有數字與符號, 只允許英文字母以及中文: "**([^\x00-\x40\x5B-\x60\x7B-\x7F])+**"
2. 本國身分證字號格式: "**[a-zA-Z][0-9]{9}**"
3. 合法的 HTML 標籤: "**<\s*(?<tag>\w+)\s*(?<attributes>[^\n\r/>]+)?\s*/?\s*>**"

強密碼驗證範例

經我多方查探, 從網路上過濾了許多可供參考的範例之後, 我認為能作為驗證強密碼(Strong Password)的最佳 Regular Expression 如下：

(?=^.{7,}$)(?=.*[a-zA-Z])(?=.*[~!@#$%^&*()_+}{":;'?/>.<,])(?!.*\s).*$

在上面的正規式子中, 你至少必須輸入一個特殊符號。它大致上已經符合 ASP.NET 2.0 中預設限制密碼的樣式。

如果你還想要強迫密碼中使用數字並限制須有大小寫英文字, 則可以使用以下的式子：

(?=^.{7,}$)(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z])(?=.*[~!@#$%^&*()_+}{":;'?/>.<,])(?!.*\s).*$

以上兩個式子都會限制密碼的最短長度為 7; 如果你希望最短長度不是 7, 請自己修改。

檢測 HTML 標籤

接著, 我們使用現實的例子來示範 Regex 的建構方式。以下是我們使用的被檢驗文字：

```
<html>
< html>
<html >
< html >

<html abc="abc">
< html abc="abc" cba="cba >
<html abc="abc">
<  html abc="abc" cba="cba  >

</html>
< /html>
</ html >
```

```
< /html >
< / html>

</html abc="abc">
</ html abc="abc">
</ html abc="abc" >
< /html abc="abc" cba="cba >
</html abc="abc" cba="cba >
<  / html abc="abc" cba="cba  >

< / html abc="abc" cba="cba >>>>>>>>>>>>>>
<<<<>>>>
>>>><<<<>>>><<<>>>><<
```

1. 樣式： <\s*([^<^>]+)\s*>

可以檢出所有的字串, 但 <<<>>> 則不會被檢出。

2. 樣式： <\s*(?<Tag>/*\s*[^<^>^\s]+)(?<Para>(\s*[^<^>^\s]+)*)\s*>

可以檢出所有的字串, 而且定義出 Tag 與 Para 兩種 Backreferences。

3. 樣式： **((<!--(?<Remark>([\s\S]*))->)|(<\s*(?<Tag>/*\s*[^<^>^\s]+)(?<Att>(\s*[^<^>^\s]+)*)\s*>))**

我的最後版本, 已經可以把程式裡的註解取出來/拿掉。

你可以建立更多的測試資料, 看看這個樣式有沒有錯誤。我自己已經測試過 RegExLib.com 的所有號稱可以檢測出 HTML Tag 的 Pattern, 都沒辦法從我的測試資料中檢測出全部, 或是無法排除像 <<<>>> 這種錯誤的 Tag。我的第二和第三個 Pattern 則可以正確執行。

檢測 HTML 標籤內的屬性

當你使用以上的方法檢測出 HTML Tag 之後, 你可以根據擷取出來的資料做進一步的分析。在這裡, 我要以 img 控制項做例子, 將文件中所有找到的 Tag, 若它的名字是 img, 則取出它的 src 元素。

樣式如下： **(?ix:src=((["'\s]+(?<Src>([^<>"']+))["'\s]+)))**

在這裡, 我特別加上了容錯的功能。如果使用者輸入的是像 <img src="http://abc.com/abc.jpg"> 或 <img src='http://abc.com/abc.jpg'> 的合法格式, 當然可以正確的擷取出來, 但如果像 <img src=http://abc.com/abc.jpg>, 也可以擷取出來。

我的測試樣式如下：

```
<img SRC=     "   abc0" cba="cba" >
<img aaa=ccc src=   abc1   >
< img src=     "   abc2" cba="cba >
```

```
<img src= ' a  bc3' >
< img src="  abc4  " cba="cba  >
```

其中 abc0, abc1, abc2, a bc3 和 abc4 都應該能被檢測出來。

前端驗證的技巧與注意要點

後端的主流語言 (C#、VB、Java、PHP等等) 固然都已提共 Regex 功能, 其實在前端的 JavaScript 程式中也同樣提供 Regex 檢核。在 JavaScript 中使用 Regex 非常的簡單, 基本上就是套用 string.**match()** 函式即可(有興趣者可以參考 w3school.com 上面的介紹)。

以下我舉個實用的範例。例如, 如果我在 ASP.NET 網頁中放置一個只允許輸入數字的文字方塊, 我可以這樣寫:

```
<asp:TextBox ID="txtInput" runat="server"
    onkeyup="if (value.match(/^\d+$/)==null) value='';"
    onblur="if (value.match(/^\d+$/)==null) value='';" />
```

在這裡請特別注意, 在 JavaScript 中 Regex 的 pattern 是以斜線括住來表示, 也就是如上例中的 /^\d+$/ 以 // 括住的部份。我使用了 onkeyup 事件讓使用者每打一個字就檢查一遍。但為什麼要在 onblur 事件中又做一次同樣的事情呢? 這是因為使用者如果硬是以複製的方法貼一個非數字的字串進來的話, 那麼當 onblur 發生時, 還是可以進行驗證。或許你又要問了, 如果按 Ctrl-V 的話, 這個動作也會引發 onkeyup 事件不是嗎? 是這樣沒錯, 但萬一人家使用滑鼠右鍵選單中的「貼上」呢? 這個世界上有不少人是十分堅持複製貼上一定只能使用滑鼠來進行的。所以既然要防呆, 就給它防得徹底一點吧!

如果你個人是堅持絕對不能在 onkeyup 和 onblur 裡面撰寫相同程式碼的話 (最近感覺到世界上有很多人有很多奇怪的堅持, 我不做評論), 那麼要防止使用者使用貼上功能的一個方法, 就是下一個 **onpaste("return false;");** 就可以了。但若依我個人的習慣, 我寧願保留使用者使用剪貼簿的權力。

除了 string.match() 函式之外, JavaScript 的 string.replace() 函式也支援 Regex。以下我花一點篇輻介紹一下。

本函式的語法如下: `newstring = str.replace(regexp|substr, newSubStr|function[, ` Non-standard `flags]);`

關於本函式的用法, 由於上述網站 (即 developer.mozilla.org) 已經解釋得相當清楚, 所以我就不重覆介紹了。我只把該函式把 Regex 代入 function 的部份拿出來說明。

JavaScript 可以隱含迴圈, 我想熟悉 JavaScript 的人都知道。而 replace 函式中的第二個參數可以使用 function, 讓你可以把 regex 檢核出來的結果丟進去當作參數以進行後續的處理。請注意這就是一個隱含迴圈 (檢核出來幾個結果就執行幾次)。我們來看上述網站中最後一個例子:

```
function f2c(x)
{
  function convert(matched, p1, offset, str)
  {
    return ((p1-32) * 5/9) + "C";
  }
```

```
  var s = String(x);
  var test = /(\d+(?:\.\d*)?)F\b/g;
  return s.replace(test, convert);
}
```

比較值得注意的一點, 在於 function 的參數部份。在上述程式中, 你看到 function 的參數是 (str, p1, offset, s), 依次代表 matched substring (符合的子字串)、參數 1、位移數字和整個字串。其實你可以只賦予 str 和 p1 兩個參數, 亦即將程式中 convert(str, p1, offset, s) 這一段改成 convert(str, p1)。此外, 如果你的 Regex 中會回傳兩個結果, 那麼你必須把參數改為像 convert(str, p1, p2, offset, str) 或者 convert(str, p1, p2) 的形式。前述網站中有舉一個例子, 假設你的 Regex pattern 是 /(\a+)(\b+)/, 那麼 p1 會接受符合 \a+ 的結果, p2 會接受符合 \b+ 的結果。不過這麼一來, 如果你的程式中需要 offset 和 str 這兩個參數, 那麼更改 pattern 時要注意參數的排列, 以免取到錯誤的結果。我會建議你採用前述網站的命名規則 (即 p1, p2, offset, str 等) 以避免混淆, 也才不會不小心取到 undefined。

我把這段程式略做修改後放上 jsFiddle, 各位可以試著執行看看:

我最後想再強調一下, 不管你在前端做了任何預防性的檢查 (無論是使用 Validator 或者自己寫 JavaScript), 在後端最好也能再做一次檢查, 才能確保萬無一失。為什麼要這麼麻煩呢? 最主要的原因在於安全上的考量。例如, 你可能已經為所有輸入項目都加上了 Regex 驗證, 以預防像 SQL Injection 和 XSS 等攻擊手法, 但是請不要忽略了一點, 萬一客戶端禁止 JavaScript 執行, 你到底該不該信任這名客戶的輸入呢? 此外, 高竿的駭客可以冒充瀏覽器, 直接向伺服器進行 request, 如此一來, 你的所有前端程式都自動繳械了, 請問你又該怎麼處置這種情況呢?

所以, 除非你非常確定上述情況絕對不會發生 (例如你的網頁只有少數可信賴的人會上來操作), 否則你最好能在伺服器端再重新驗證所有的輸入 (或者予以編碼)。

不過另一種情況又來了。在少數情況下, .Net 和 JavaScript 對於 Regex 的支援並不完全一致, 尤其在一些進階應用的部份 (例如我在「[Regex] 進階群組建構」裡面介紹的那些技巧)。因此, 如果你要在前端程式中進行 Regex 驗證 (尤其是採用複雜樣式的時候), 你最好能事先進行詳細的測試, 如此才能確保在前端及後端都能獲得相同的結果。

HTML5 的支援

在 HTML5 中增加了使用 Regex 樣式進行驗證的功能, 這使得我們可以不必再使用 JavaScript 做驗證 (不過如上提到的, 後端的重複驗證仍然不能省)。它的語法相當簡單:

```
<input type="text" pattern="..." title="..." />
```

如上, 將 Regex 樣式寫在 pattern 中, 把警告訊息寫在 title 中就可以了。如果使用者的輸入不符合樣式, 那麼他填寫的內容就無法送出。

不過, 這個文字框必須包在 form 區段裡面。否則驗證並不會發生。

有興趣的朋友可以參考我製作的示範 fiddle:

請特別留意, HTML5 的 Regex 驗證雖然方便, 它只提供非常輕量級的 Regex 功能, 而且其能力只限於「驗證」。如果你需要的是較複雜的 Regex 功能, 使用較有力的語言 (例如 C# 或者 JavaScript) 仍然是有必要的。