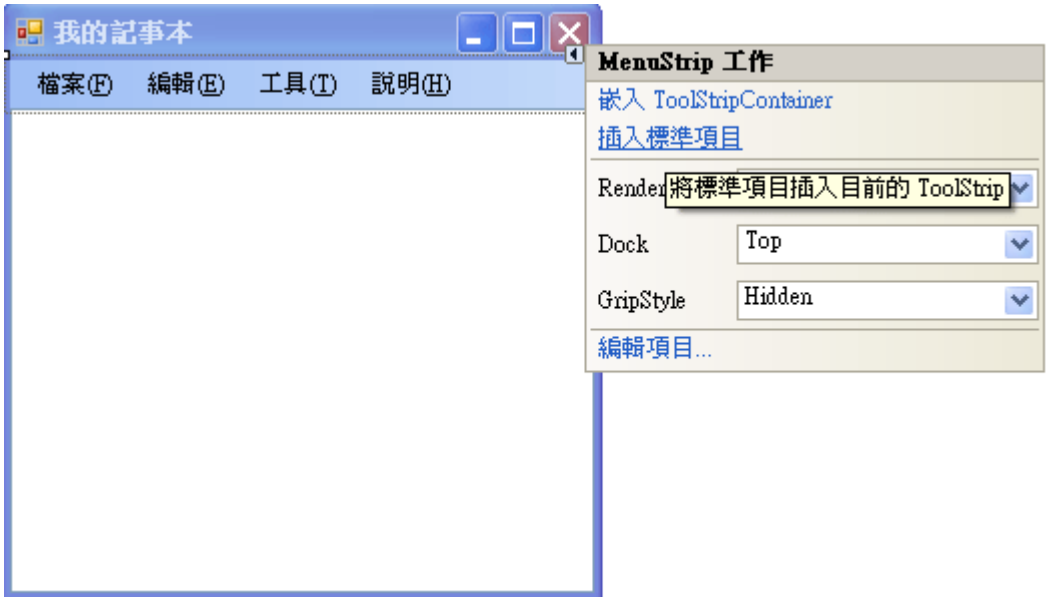
第 4 章　簡易記事本

簡介：

這一章要製作一個簡易的記事本，過程中會介紹功能表的建立與編輯、檔案的存取、對話方塊的設定與使用，以及一些文字資料，也就是字串的處理技巧。你會發現充分的運用現有的工具物件與預設功能，要寫出一個有「標準軟體」外觀的程式真的不難！

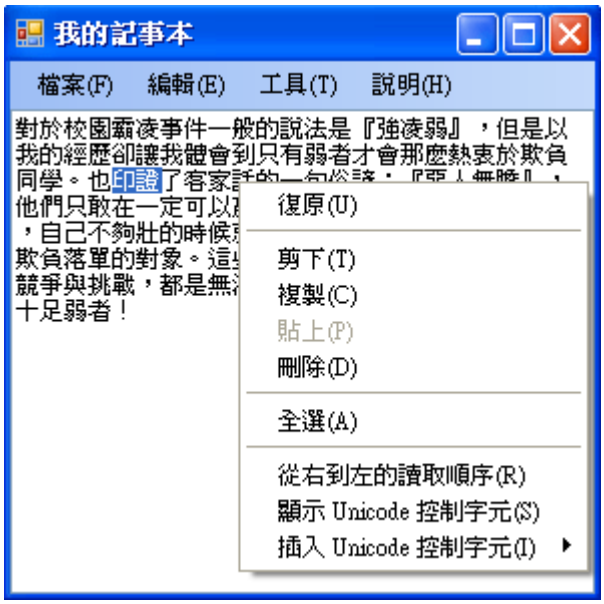
4-1 建立編輯區及功能表

一個記事本程式的基本外觀就是一個視窗之內有功能表，以及可以編輯文字的區域。首先請在工具箱找到「功能表與工具列」分類的 MenuStrip(主功能表)物件加入表單，並點選主功能表右上方的三角形顯示 MenuStrip 工作視窗，點選「插入標準項目」，會看到立即出現相當完整的預設功能表選項(如下圖)。接著再到工具箱加入一個 TextBox，將它的 Dock 屬性設定為 Fill(填滿可用區域)。



但是你會發現它仍然只有一行！也沒有真的填滿視窗空間，原因是有個 Multiline(多行)屬性預設值是"False"("非"多行！就是限制為單行囉！)，必須將它改為 True，才會看到正確的畫面。此外，請將它的 ScrollBars(捲軸)屬性設為 Both，就是水平與垂直向都要可以捲動，這樣檔案太大時才能看得完整。當然表單的標題(Text)，甚至圖示(Icon)也最好都改一下囉！

此時我們還沒寫任何程式，但是你試著啟動程式，會發現功能表已經可以下拉與選擇(但沒反應)，文字方塊當然也有輸入文字的功能，甚至按右鍵還有預設的跳出式選單，讓你可以執行剪下或複製文字到別處，或從別處複製一段文字進來貼上等等的基本操作。如下圖：



4-2 開啟檔案

打開多數的軟體通常最先執行的是開啟舊檔，點選之後會出現一個選擇檔案的視窗，它也是工具箱裡面的物件哦！它位於「對話方塊」分類下，名為 OpenFileDialog。請先選取這個物件(OpenFileDialog1)進入表單，但它不是預設會顯示於表單上的物件，設計時只會顯示於表單下方的灰色區塊之內。需要設定的屬性只有一個就是 Filter，它可以限定(或稱「過濾」)特定的檔案，譬如我們的記事本其實只能處理純文字檔案，也就是副檔名為 txt 的檔案，所以要在 Filter 屬性填入「純文字檔案|*.txt」。其中的垂直線是一般鍵盤上的「Shift+"\"」產生的字碼。在此符號之前的文字是顯示給使用者看的(可以任意更改)，直線之後的「*.txt」則是告知電腦：本對話方塊只要顯示副檔名為 txt 的檔案。當然，如果你不定義 Filter 屬性，所有檔案就會通通出現，這時如果使用者選擇將非文字檔(譬如圖檔)載入本程式的文字方塊，程式就會當掉！

接著請到你設定的主功能表的「檔案」項目下雙擊「開啟」功能，進入事件副程式寫程式如下：

```
Private Sub 開啟OToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles 開啟OToolStripMenuItem.Click
    If OpenFileDialog1.ShowDialog <> Windows.Forms.DialogResult.Cancel Then
        TextBox1.Text = My.Computer.FileSystem.ReadAllText(OpenFileDialog1.FileName, _
                                                           System.Text.Encoding.Default)
    End If
End Sub
```

這段程式碼的功能是使用者點選開啟後會出現一個選檔案視窗，除了目錄一定會有之外，只會出現副檔名為 txt 的純文字檔案；如果使用者後悔不想開檔了，可能會按下「取消」(Cancel)鍵，此時程式就會關閉選檔視窗，不再繼續開檔的動作。事實上只要直接寫：

```
OpenFileDialog1.ShowDialog
```

就是開啟選檔視窗的指令了！它會開啟視窗，當使用者選好檔案之後將檔名路徑存到它的 FileName 屬性之中，接下來就可以據此執行開檔案的動作了！如果使用者按下「取消」按鍵，檔名(FileName)應該是不具意義的預設值「OpenFileDialog1」，當後面的程式碼嘗試開啟不存在的檔案時，程式就會當掉了！但是我們如何知道使用者是否有按下「取消」呢？關鍵在於 OpenFileDialog1.ShowDialog 的回傳值，我們用：

```
If OpenFileDialog1.ShowDialog <> Windows.Forms.DialogResult.Cancel Then
```

這一行程式來判斷取消鍵是否被按下，不必擔心 Cancel 前面的一大串文字，寫程式中它會自動出現讓你選擇，你只要記得關鍵字「Cancel」即可。此行的意思是：「如果 Cancel 鍵「沒有」(<>表示不等於)被按下(回傳值不等於 Cancel)就繼續以下程式…」

```
TextBox1.Text = My.Computer.FileSystem.ReadAllText(OpenFileDialog1.FileName, _
                                                   System.Text.Encoding.Default)
```

上面的程式碼是開啟檔案的動作，完整的意義意是：在我的(My)電腦(Computer)的檔案系統(FileSystem)中讀取(Read)檔案名為 OpenFileDialog1.FileName(選取的)檔案中的所有文字(AllText)，需使用系統(System)預設(Default)的文字編碼(Encoding)方式，並將讀取的文字放到 TextBox1 的 Text 屬性之中」

這種 My…的寫法看起來有點冗長卻是 VB 專屬的好用特色之一！它被稱為「My 捷徑」，是 VB 將常用的系統函式庫功能集結作一個較小的分類體系，這樣比起到完整的系統程式庫選程式方便很多，而且它使用簡易英文的思考邏輯分類，譬如要處理「檔案」就可以試著到「我的・電腦・檔案系統」項目下去找功能。事實上你還可以自己修改這個 My 體系適應你的使用習慣，在此就不再詳述了！

4-3 儲存檔案與另存新檔

開檔時要用到"**Open**"FileDialog 物件，存檔時理所當然就必須用到"**Save**"FileDialog 了！確實有這個東西，也請你在工具箱對話方塊類別中找到它並加入表單。當我們開始構思有關存檔功能時會有個小困擾：儲存檔案與另存新檔都是存檔，但是一個是使用"原來"的名稱 OpenFileDialog1.FileName 存，一個是使用新選定的名稱 SaveFileDialog1.FileName，最麻煩的是如果是新增的檔案並沒有名稱，此時又必須與另存新檔程序一樣，要存檔時又如何知道目前檔案是開啟的舊檔或新增的檔案呢？

首先請先建立好最單純的狀況「另存新檔」的程式碼如下：

```
Private Sub 另存新檔AToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles 另存新檔AToolStripMenuItem.Click
        If SaveFileDialog1.ShowDialog <> Windows.Forms.DialogResult.Cancel Then
            My.Computer.FileSystem.WriteAllText(SaveFileDialog1.FileName, TextBox1.Text, False)
        End If
End Sub
```

如同開檔時一樣，先要確定使用者「沒有」按下取消鍵才要進行存檔。存檔的指令類似開檔也是從 My 捷徑起始，將 Read 改為 Write(寫入)，參數有三個，包括：要寫入的檔名，

要寫入的資料(TextBox1.Text)，以及是不是要接續(Append)於原檔案內容之後？我們在作的是整個檔案的編輯，當然是選「不要」(False)接續，也就是整個舊檔案要被覆蓋掉的意思。

有了「另存新檔」的基礎，接下來我們使用 OpenFileDialog1.FileName 屬性的有無來判斷目前使用的是新增檔案，還是已經開啟的舊檔，如果是新增的！OpenFileDialog1 應該沒有檔名，不過請注意到這個物件預設是有一個檔名(FileName)叫「OpenFileDialog1」的！這個檔名當然毫無意義，也無法開啟，請先將它刪除。

接著請建立新增與儲存的程式碼如下：

```
Private Sub 新增NToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles 新增NToolStripMenuItem.Click
    OpenFileDialog1.FileName = Nothing
    TextBox1.Clear()
End Sub

Private Sub 儲存SToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles 儲存SToolStripMenuItem.Click
    If OpenFileDialog1.FileName = Nothing Then
        If SaveFileDialog1.ShowDialog <> Windows.Forms.DialogResult.Cancel Then
            My.Computer.FileSystem.WriteAllText(SaveFileDialog1.FileName, TextBox1.Text, False)
        End If
    Else
        My.Computer.FileSystem.WriteAllText(OpenFileDialog1.FileName, TextBox1.Text, False)
    End If
End Sub
```

在「新增」的功能中我們先將 OpenFileDialog1 的 FileName 屬性設為 Nothing 相當於清除資料，接著也將文字編輯區 TextBox1 的內容清除(Clear)，如果你寫成 TextBox1.Text="" 或者 TextBox1.Text=Nothing 效果也是相同的！接下來的儲存功能中首先就是檢查 OpenFileDialog1 的 FileName 屬性是否存在？如果沒有，表示是新檔案，處理方式與另存新檔完全相同，否則(Else)就直接使用 OpenFileDialog1 的 FileName 作存檔案的動作，相當於回存到之前開啟的舊檔案。

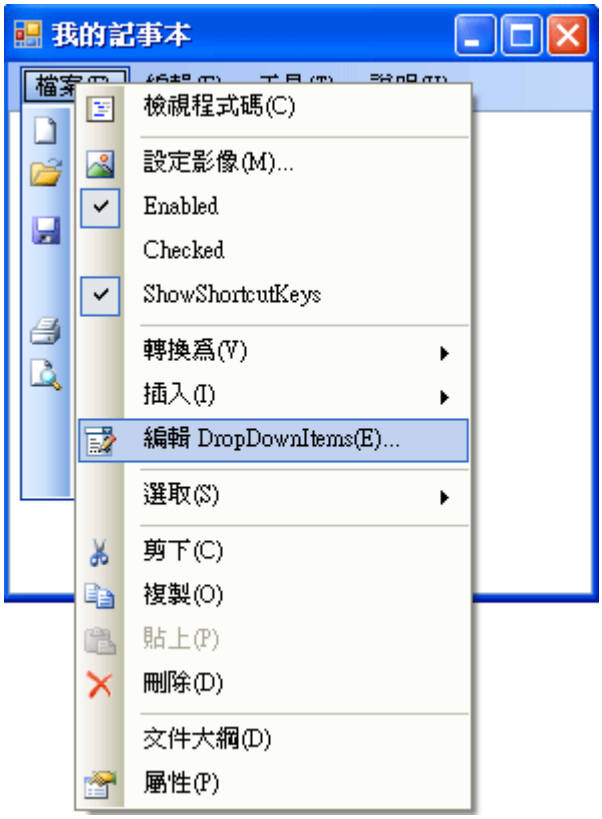
在此順便交代一下結束功能的程式就是這樣：

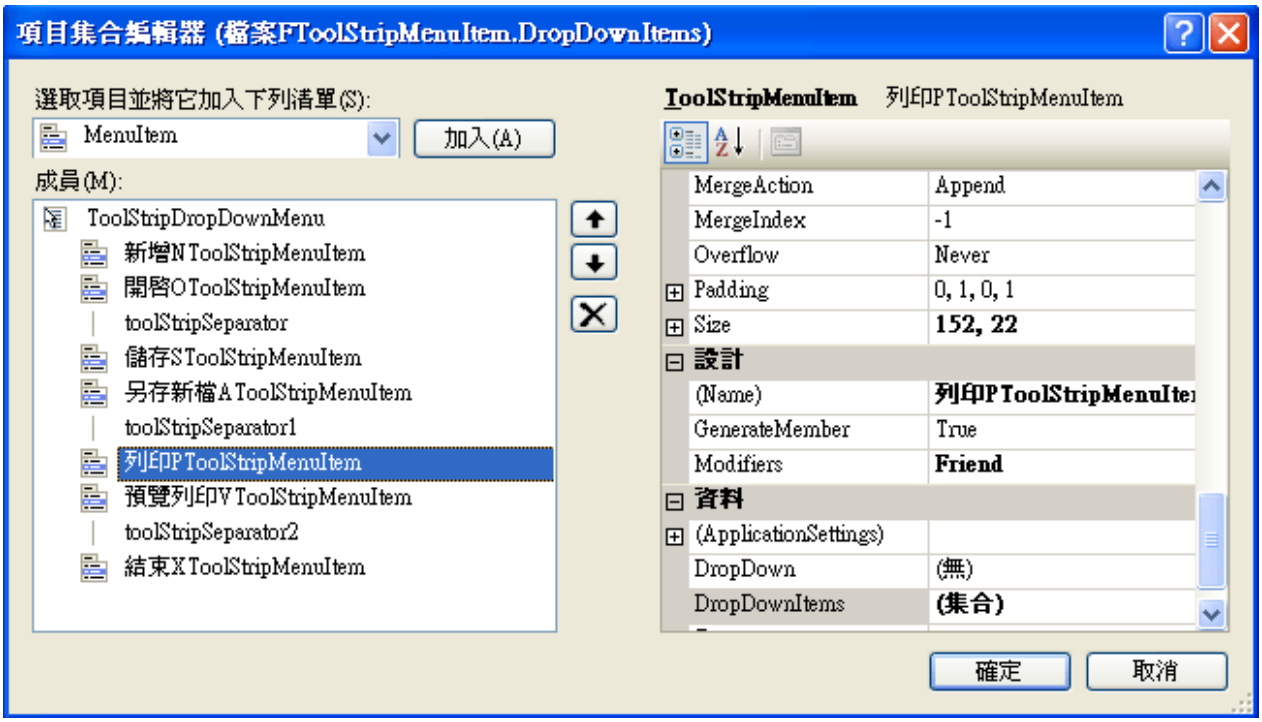
```
Private Sub 結束XToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles 結束XToolStripMenuItem.Click
    Application.Exit()
End Sub
```

有些程式範例中結束程式會寫成 Me.Close，意思是關閉本表單，在此例中效果一樣，都可以完全結束程式，但是兩者涵義稍有不同。Application 是指整個專案程式，Me 是指本表單，而一個專案程式可能有多個表單，關閉其中一個未必關閉得了其他的表單。但是如果專案只有一個表單，或者你關閉的是啟動時第一個出現的「主表單」，還是可以完全關閉程式。建議初學者建立正確觀念，不要將 Me.Close(關閉表單)視為結束程式的意思比較好！當然 VB 的舊版還有更簡單的結束程式指令，直接寫 End，在此也是可以使用的！

4-4 編輯功能表

預設的功能表總不能照單全收，如果需要更改時應該如何操作？首先本單元不準備作列印的功能，請將滑鼠移到功能表的「檔案」位置按下右鍵出現選單如下時選擇「編輯 DropDownItem...」，會出現一個「項目集合編輯器」視窗。





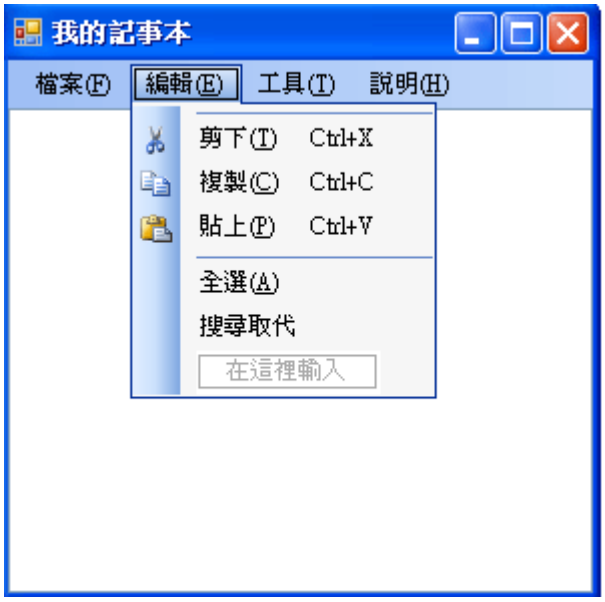
在上面的編輯視窗中可以選取任何一個項目作細節屬性的編輯，如顯示文字的 Text 屬性，也可以刪除(叉叉按鈕)或移動項目的上下位置(上下箭頭按鈕)，使用加入按鍵還可以新

增項目。當然不用這個編輯器，直接在 MenuStrip 物件上也可以作編輯動作，但以使用此編輯器較為方便，功能也比較完整。

在此，請使用這種方式將列印相關的兩項功能刪除，其中顯示為「toolStripSeperator」的項目是格線，也可以同時刪除。在此順便也將「編輯」功能中的「復原」與「取消復原」刪除並加入一個「搜尋取代」的項目；再將「工具」選項中的「自訂」改為「字型」,「選項」改為「顏色」；最後將「說明」項目刪除。這些保留或新增的項目就是本單元接下來會教大家製作的功能。

4-5 編輯功能

如上節指示編輯好的「編輯」功能表內容大概是這樣的：



請分別雙擊「剪下」、「複製」、「貼上」與「全選」項目寫程式如下：

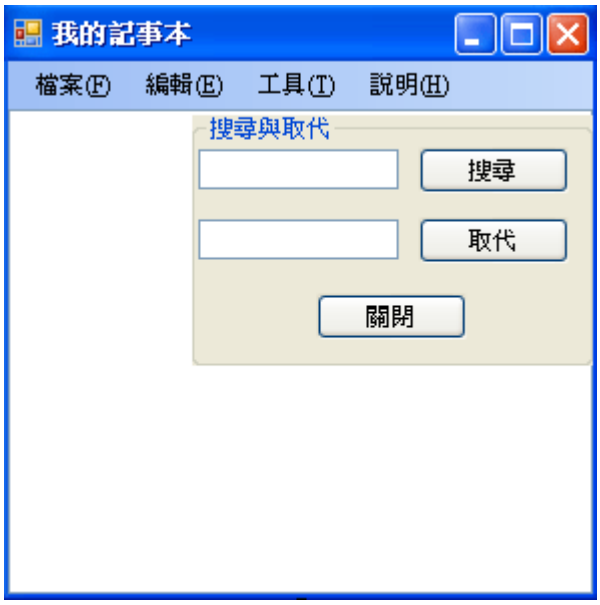
```
Private Sub 剪下TToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles 剪下TToolStripMenuItem.Click
    TextBox1.Cut()
End Sub

Private Sub 複製CToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles 複製CToolStripMenuItem.Click
    TextBox1.Copy()
End Sub

Private Sub 貼上PToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles 貼上PToolStripMenuItem.Click
    TextBox1.Paste()
End Sub

Private Sub 全選AToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles 全選AToolStripMenuItem.Click
    TextBox1.SelectAll()
End Sub
```

非常簡單，都是一個指令就完成一個操作，因為它們都是 TextBox 物件原本就有的功能！這些功能其實可有可無，因為即使不寫這些程式，使用 TextBox 預設的按右鍵出現的跳出式選單也一樣可以做到這些事情。真正需要一點技術的是「搜尋取代」功能。請先建立下面的物件組合：叫用一個工具箱中「容器」類的 GroupBox 物件，再到這容器內建立兩個 TextBox 以及三個按鍵，並改變 GroupBox 的 Text 為「搜尋與取代」，佈置如下圖：



先將上面的 GrouBox1 的 Visible 屬性設為 False 意思是程式預設它是隱藏的，當使用者按功能表的「搜尋取代」時才會出現，按關閉鍵時則須再度隱藏，所以這兩個事件副程式內容如下：

```
Private Sub 搜尋取代ToolStripMenuItem1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles 搜尋取代ToolStripMenuItem1.Click
    GroupBox1.Visible = True
End Sub

Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _
    Handles Button3.Click
        GroupBox1.Visible = False
End Sub
```

請先測試一下這個 GroupBox1 是不是可以正確的開關！如果沒問題，請寫「搜尋」按

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles Button1.Click
    Dim P As Integer
    If TextBox1.SelectionLength > 0 Then
        P = TextBox1.Text.IndexOf(TextBox2.Text, TextBox1.SelectionStart + 1)
    Else
        P = TextBox1.Text.IndexOf(TextBox2.Text, TextBox1.SelectionStart)
    End If
    If P < 0 Then
        MsgBox("未發現搜尋字串！")
    Else
        TextBox1.SelectionStart = P
        TextBox1.SelectionLength = TextBox2.TextLength
        TextBox1.Select()
    End If
End Sub
```

這段程式的核心程式碼是 TextBox1.Text.IndexOf(TextBox2.Text…，意義是在 A 字串中搜尋 B 字串時的語法就是 A.IndexOf(B, StartIndex)，所謂的 Index 是索引，字串中的第一個字索引值是 0，第二個字是 1，依此類推！IndexOf 表示 B 字串在 A 字串中的索引位置，如果搜尋沒有發現符合的狀況就會回傳 IndexOf = -1！IndexOf 內的第二個參數 StartIndex 是指在 A 字串中開始搜尋的「索引起點」，如果你不指定這個值也是可以的，那麼它永遠會從字串的開始(索引 0)處開始搜尋，這樣你就永遠只能搜尋到「第一個」符合條件的目標，無法持續找下一個符合條件的目標了！

上面的程式一開始檢查目前的 TextBox1 中有沒有選取文字？(SelectionLength>0)如果有的話表示已經有找到目標，接下來必須搜尋下一個目標，那麼索引值就必須是目前選取起點(就是編輯區的文字游標位置)的下一個字，否則會再度搜到同一個字串！反之，如果目前沒有選取任何文字(SelectionLength=0)應該是剛開始搜尋，那就直接使用目前游標的位置(SelectionStart)開始搜尋，有可能第一個字就找到目標了！

接下來依據回傳值 P，也就是目標字串的索引位置作出回應。如果找不到時 P= -1！就讓它出現訊息(MsgBox)顯示未找到字串！否則就是找到了，先將選取範圍的起點(SelectonStart)設定為 P，再將目標字串(TextBox2.Text)的長度設為選取資料的長度(SelectionLength)，最後執行選取動作(Select)此時就會看到被搜出的字串被選取出來而呈現反白了！

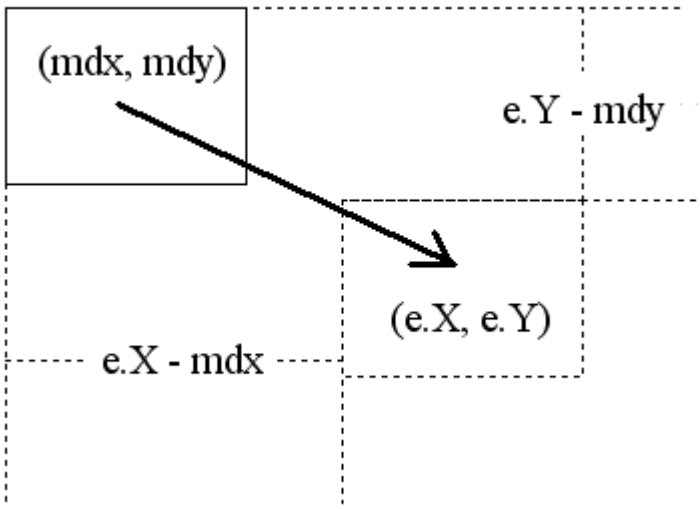
接下來我們可以依據 TextBox3 的內容來執行取代，請寫「取代」按鍵的程式如下：

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles Button2.Click
    TextBox1.SelectedText = TextBox3.Text
End Sub
```

就是將 TextBox1 中被選取的文字(SelectedText)用 TextBox3 的內容(Text)取代。

4-6 拖曳視窗

至此如果還有一點不滿意的可能是那個搜尋取代的小框框(GroupBox1)無法移動，常常會遮住想看的文字。我們可以使用 GrouBox1 的 MouseDown 與 MouseMove 事件製作一個拖曳程式即可。物件移動座標的示意圖如下：



程式寫法如下：

```
Dim mdx As Integer, mdy As Integer
Private Sub GroupBox1_MouseDown(ByVal sender As Object, ByVal e As System.Windows.Forms.MouseEventArgs) _
   Handles GroupBox1.MouseDown
    mdx = e.X
    mdy = e.Y
End Sub

Private Sub GroupBox1_MouseMove(ByVal sender As Object, ByVal e As System.Windows.Forms.MouseEventArgs) _
   Handles GroupBox1.MouseMove
       If e.Button = Windows.Forms.MouseButtons.Left Then
           GroupBox1.Left += e.X - mdx
           GroupBox1.Top += e.Y - mdy
       End If
End Sub
```

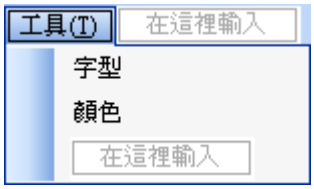
首先是宣告拖曳起點的座標變數 mdx 與 mdy，這兩個變數必須宣告在事件副程式之外，可以視為公告，程式設計的術語叫作「全域變數」就是整個程式中的任一副程式都可以辨識並使用這個變數的意思，因為我們必須在之後的兩個事件副程式中都用到它，所以這樣宣告是必須的。接下來在 MouseDown 事件，就是滑鼠按下(準備開始拖曳)的時候紀錄起點的 X 與 Y 座標。所謂的 e 是事件副程式夾帶的參數集合，e.X 表示滑鼠的 X 座標，e.Y 是 Y 座標，每個事件副程式都會有這個顯示為 e 的集合，但是內容依據事件性質而有不同。譬如上一單元製作的鍵盤事件中就不會有滑鼠座標值。

實際拖曳的程式寫在 MouseMove(滑鼠移動)事件中，先檢查滑鼠左鍵是否按下，其中的 e.Button 指滑鼠鍵，MouseButtons.Left 當然就是指滑鼠左鍵了！如果左鍵是按住的，那麼就該移動物件了！程式事實上是改變物件(GroupBox1)的 Left(X 方向)與 Top(Y 方向)屬性，每次的 X 移動量是終點的 e.X 減去起點的 mdx，Y 也比照辦理。試試看執行程式，現

在搜尋視窗就可以移動了！

4-7 字型與顏色的選擇

記事本程式其實並不能將文字的字型與顏色存入檔案，但是在編輯過程中選擇字型、大小，甚至顏色，還是很有用的！雖然正版的記事本沒有顏色選項，本單元卻決定製作，主要目的是介紹顏色選擇的物件。首先你應該如前面提到的，將功能表的「工具」選項內的項目更改成這個樣子：



接著請再到工具箱的「對話方塊」類別中找到 FontDialog 與 ColorDialog 物件加入表單，它們是選擇字型與顏色的工具，用法與前面的 OpenFileDialog 相似，請在功能表的「字型」與「顏色」功能的點選事件中寫程式如下：

```
Private Sub 字型CToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles 字型CToolStripMenuItem.Click
    FontDialog1.ShowDialog()
    TextBox1.Font = FontDialog1.Font
End Sub

Private Sub 顏色OToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles 顏色OToolStripMenuItem.Click
    ColorDialog1.ShowDialog()
    TextBox1.ForeColor = ColorDialog1.Color
End Sub
```

上面程式碼的意義是先呼叫開啟對話方塊(ShowDialog)，使用者選擇(或取消選擇)後將選定的字型與顏色賦予 TextBox1 的字型與顏色即可。其中 ForeColor 指字的顏色，Fore 是「前景」的意思，就是字的顏色，與它相對的就是 BackColor(背景顏色)了。值得注意的是我們並沒有如之前的 OpenFile 與 SaveFile 的 Dialog 一般，費事的處理使用者按下 Cancel(取消)時的狀況，原因是不正確的檔名確實會讓程式當掉，但是對於 Font 與 Color 而言，按下取消時仍有正確可用的「預設」字型與顏色，就讓它們重複套用一次程式也不會當掉，沒有關係的。

4.8、進階挑戰

一、如何建立功能表的快捷鍵？(難度：低)

提示：設定功能表編輯器中的 ShortCutKeys。

二、如何作出列印相關功能？(難度：中)

提示：請參考工具箱中「所有 Windows Form」類別中以 Print 開頭的相關物件。

三、如何作出復原與取消復原的動作？(難度：高)

提示：可以在 TextChanged 事件中紀錄每次文字變化的內容到字串陣列。

課後閱讀

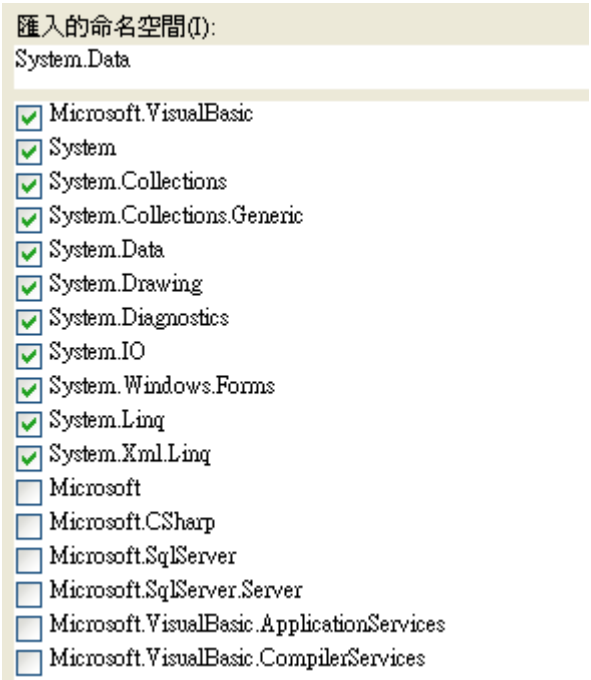
從 My 捷徑到.NET Framework

本單元碰到一個比較特別的東西，就是我們用 My.Computer.FileSystem.ReadAllText 這樣的指令去讀取文字檔。這是非常人性化的設計，也是 VB 程式語言所獨有的！我們可以依據常識的邏輯尋找到龐大程式庫裡面的特定程式來使用，你甚至可以用：

My.Computer.Network.DownloadFile(網址)

這樣的指令去下載某個網址的檔案。但是這背後代表的意義是甚麼？很值得我們深入了解一下！如同之前說的：現代的程式其實是由很多物件所組成，這些「物件」嚴格說都是功能單純的小程式。工具箱裡面的物件只是其中極小的一部分！對於微軟公司的程式軟體來說，為了方便管理與使用這些既有的龐大程式資源，他們將所有程式如圖書館的書籍一樣作階層式的編碼分類，編好的東西就稱為「.NET Framwork Libray」(函式庫)，我們使用的 Visual Studio 2008 軟體背後支援的函式庫就是「NET Framwork 3.5」版。

當我們開啟新的程式專案時，Visual Studio 會預設載入一些基本常用的物件與函式，我們可以到方案總管的 My Project 專案屬性中選擇「參考」頁，可以在視窗下方看到這樣的畫面：



所謂的命名空間(Name Space)就是函式庫裡面的分類名稱，上面「匯入的命名空間」已經勾選的項目就是已經預設匯入，寫程式時隨時可以使用的程式資源(包括工具箱物件在內)。在底下還有極長的命名空間選項可以依你的需要匯入，譬如寫資料庫或網路程式時就會需要額外匯入一些額外的命名空間。因為這個程式庫極為龐大，當然是不宜全部匯入的，不然記憶體就可能不夠用了！

一般初學者甚至老手都不可能弄清楚這些龐大的程式庫分類，而且某一個常用的功能可能為了遵守分類的邏輯而被分到很難找到的階層目錄的下層角落裡。因此 VB 程式語言的設計者想到了使用 My 捷徑的方法，就是將整個函式庫中常用的功能用一個比較簡單的體系重新分類，而且是以使用者需求的角度去分，還讓使用者可以去修改這個分類！這就是 My 捷徑了！之後的單元我們會一再的用到 My 捷徑去指定使用函式庫裡的程式，包括呼叫資源檔案與播放音效等等。

總之，必須知道的概念是：程式可以使用的資源幾乎是無限的，預設專案只會載入其中的一部分，而不管函式的命名空間有沒有被載入，只要你的程式碼可以打出正確的命名空間(NameSpace，也就是函式庫路徑)，任何程式資源都是可以使用的！My 捷徑則是整個.NET Framework 函式庫架構的精簡版！讓我們可以更容易地找到要用的程式功能。