第 4 章　我的記事本

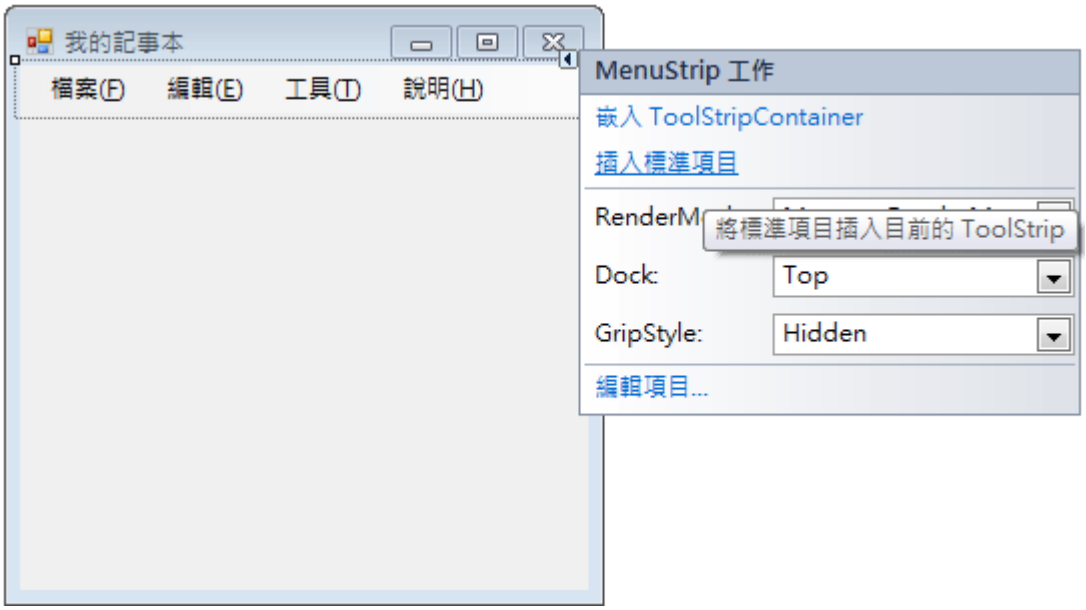
簡介：

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

[使用功能表樣板]

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)。



[設定填滿表單的文字盒]

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

[開啟檔案的對話方塊]

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。請先選取這個物件進入表單(出現 openFileDialog1)，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。

首先需要設定的屬性是 Filter，它可以限定(亦即「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」。中間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。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，程式就會當掉啦！其次此物件還有個預設的檔名(Filename)是 openFileDialog1，事實上當然沒有這個檔案，最好將它刪除，否則如果忘了選檔就按下確定鍵，程式用此檔名開檔也是會當掉的！(找不到檔案)

[匯入檔案輸出入處理的函式庫]

有一點奇怪的是：現在的 C#視窗程式預設功能中是不處理檔案的！所以沒有預先匯入相關函式庫，也許是現在的多數軟體都用資料庫處理資料，直接處理檔案的比較少吧？反而是視窗專案預設就可以處理資料庫程式。但是接下來我們要嘗試讀進文字檔，所以你必須先到程式碼頁面最前面許多匯入(using)函式庫(命名空間)的地方加上一行，「using

System.IO」改完大概是這個樣子(如下最後一行)：

```
.....
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
```

請注意！匯入的名稱應該是大寫的 IO，而不是阿拉伯數字的 10，同時間注意到有一行是匯入 Data，那就是資料庫相關函式庫了！其他值得注意的是 Drawing 表示繪圖函式庫，Text 表是文字處理函式庫，裡面就有文字編碼等函式，需要處理非英文的資料時就很重要。如果你不作這個匯入函式庫的動作，又堅持要使用相關函數，如上一單元讀取網頁原始碼的 File 函式，也可以在函式前面直接加上 System.IO(.File)。

[開啟檔案的程式]

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 void 開啟OToolStripMenuItem_Click(object  sender, EventArgs e)
{
    if (openFileDialog1.ShowDialog( )  == DialogResult.OK){
       textBox1.Text  = File.ReadAllText(openFileDialog1.FileName,Encoding.Default);
    }
}
```

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(openFileDialog1)，其中除了目錄一定會顯示之外，只會出現副檔名為 txt 的純文字檔案；這是因為我們剛剛設定的 Filter 屬性「純文字檔案|*.txt」產生的效果。如果使用者正確選擇檔案之後按下確定鍵，就會合乎(openFileDialog1.ShowDialog() == DialogResult.OK)的條件，進入後續大括號內的程式繼續執行；相對的，如果按「取消」鍵，不選檔案了，大括號內的程式就不會執行。

請注意到上述程式中間的兩個連續等號是 C#語言中表示兩者相等的關係運算子，如果左右兩邊相等傳回 True(是)，不等時傳回 False(否)。順便介紹不等於的關係運算子是「!=」，大於與小於就是「>」與「<」，「>=」是大於等於，「<=」是小於等於。這應該不難理解，但是初學者常犯的錯誤是將 >= 或 != 當作兩個字，中間加上一個空格，這樣 C#反而會看不懂而產生錯誤！

openFileDialog1.ShowDialog 是開啟選檔視窗的指令，它會開啟選檔視窗，當使用者選好檔案之後將檔名路徑會存到 openFileDialog1 的 FileName 屬性之中，接下來就可以據此執行開檔案的動作。

```
textBox1.Text = File.ReadAllText(openFileDialog1.FileName,Encoding.Default);
```

上面的程式碼是開啟檔案的動作，完整的意義是：「使用檔案(File)功能讀取所有的文字(ReadAllText)，從 openFileDialog1.FileName(選取的)檔案中讀取，而且需使用系統預設(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」

[File 物件的功能]

File 是一個可以處理很多檔案功能的靜態類別，主要包括取得檔案的資訊，讀與寫文字格式的檔案等等。類似第一個單元用過的 Math 物件，它不需要宣告就可以執行一些功能，但是它屬於 System.IO 命名空間，前面如果未宣告 using System.IO 是無法使用的。多數 C#語言書籍介紹檔案讀寫時會使用資料流物件如 FileStream，StreamReader，甚至 BinaryReader 等等複雜的語法。但是目前多數較複雜的資料處理多使用資料庫技術，比較需要學習的是 SQL 語法，資料流的程式技術相對較少用到，我們暫時無須為了簡單的文字檔讀寫去學習相當複雜的資料流處理技術，學會 File 就很夠用了！

[if 的語法結構]

在此必須回頭介紹一下 if ( ) { }，這個語法，這是 C#語言最基本的流程選擇結構，在小括號內必須寫條件式，表示符合此條件時須執行後方大括號內的程式碼。初學者在此常犯的錯誤是在小括號之後加上分號，雖然實際寫程式時為了閱讀清晰起見小括號與大括號之間多半會分行，但是在語法上這個指令(條件)並沒有結束，所以不能加上分號，必須小心！相對的，大括號內部的程式可能有很多行，則是每個指令都需要在末端加上分號的！

寫 C 語言程式最麻煩的地方就是括號很多，且必須完全對稱，因此對於初學者最佳的建議是每次必須用到有括號的語法結構時，請先將左右括號都一起先打好，再回頭在括號內填寫程式碼，以免忘了打右括號而出錯，當你的括號層次多時真的很難辨認括號應該打哪裡？所以要用 if 時就請先一次打好：**if ( ) { }**就對了！

4-3 儲存檔案與另存新檔

[儲存檔案的對話方塊]

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(存檔)！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。同時也讓此物件的 Filter 與 openFileDialog1 一樣可以過濾文字檔(Filter=純文字檔案|*.txt)。在此 Filter 的最大作用是當使用者鍵入輸出檔名時，不必打「.txt」的副檔名，程式會自動加上。如果沒有定義 Filter，又沒有自己鍵入副檔名，輸出檔案就會沒有副檔名，變成不知道要用甚麼程式開啟的不明檔案格式了！首先請先建立好最單純的狀況「另存新檔」的程式碼如下：

```
private void 另存新檔AToolStripMenuItem_Click(object  sender, EventArgs e)
{
    if(saveFileDialog1.ShowDialog()  == DialogResult.OK){
        File.WriteAllText(saveFileDialog1.FileName,  textBox1.Text, Encoding.Default);
    }
}
```

[儲存與另存新檔的差別]

上面程式碼會強制我們選個新檔名存檔，但是當我們繼續構思**儲存**功能時會有個小困擾就是：**儲存**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？

結論是：使用者按儲存鍵時必須先檢查 openFileDialog1.FilenName 是不是空的！如果是空字串表示之前使用者並未選檔開啟或者選了檔案最終卻按取消，這時必須使用與另存新檔一樣的程式碼，否則(else)就必須以 openFileDialog1.FilenName 檔名存回原來的檔案。這個較複雜的選擇結構框架是這樣的：if ( ) { } eles {}，完整程式碼如下：

```
private void 儲存SToolStripMenuItem_Click(object  sender, EventArgs e)
{
    if (openFileDialog1.FileName  == "")
    {
       if (saveFileDialog1.ShowDialog()  == DialogResult.OK)
       {
          File.WriteAllText(saveFileDialog1.FileName,  textBox1.Text, Encoding.Default);
       }
    }
    else
    {
       File.WriteAllText(openFileDialog1.FileName,  textBox1.Text, Encoding.Default);
    }
}
```

不過請注意到 openFileDialog1 物件預設是有一個檔名(FileName)叫「openFileDialog1」的！如前所述，這個檔名毫無意義，必須先將它刪除，否則上面的程式碼看到『有』檔名，結果會將新檔案自動存成 openFileDialog1.txt 也很怪的！

接著建立新增功能的程式碼如下：

```
private void 新增NToolStripMenuItem_Click(object  sender, EventArgs e)
{
    openFileDialog1.FileName  = "";
    textBox1.Clear();
}
```

在「新增」的功能中我們先將 openFileDialog1 的 FileName 屬性設為""，相當於清除檔名，接著也將文字編輯區(textBox1)的內容清除(Clear)。這樣稍後此檔案儲存時會被視為未

命名的新檔案，循著另存新檔的方式處理。

[結束程式的程式碼]

此外，結束功能的程式也相當簡單，如下所示：

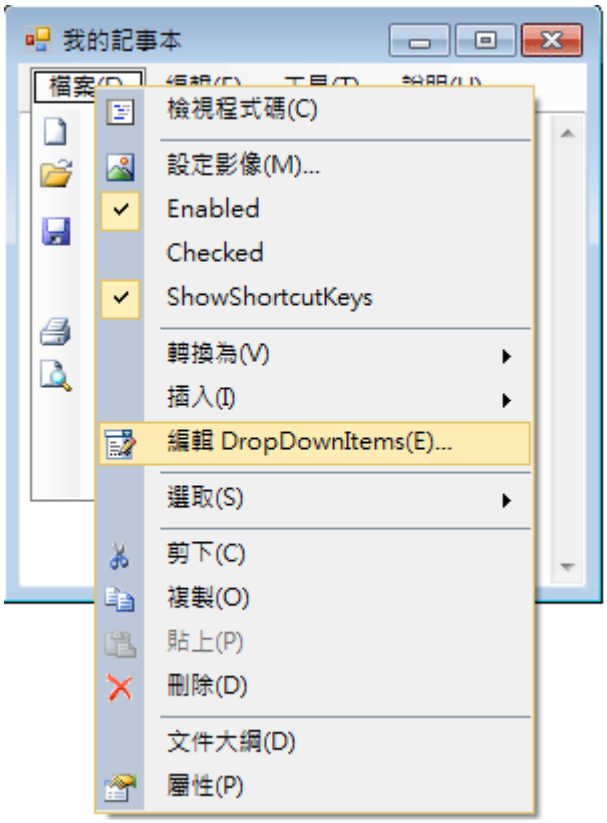
```
private void 結束XToolStripMenuItem_Click(object  sender, EventArgs e)
{
    Application.Exit();
}
```

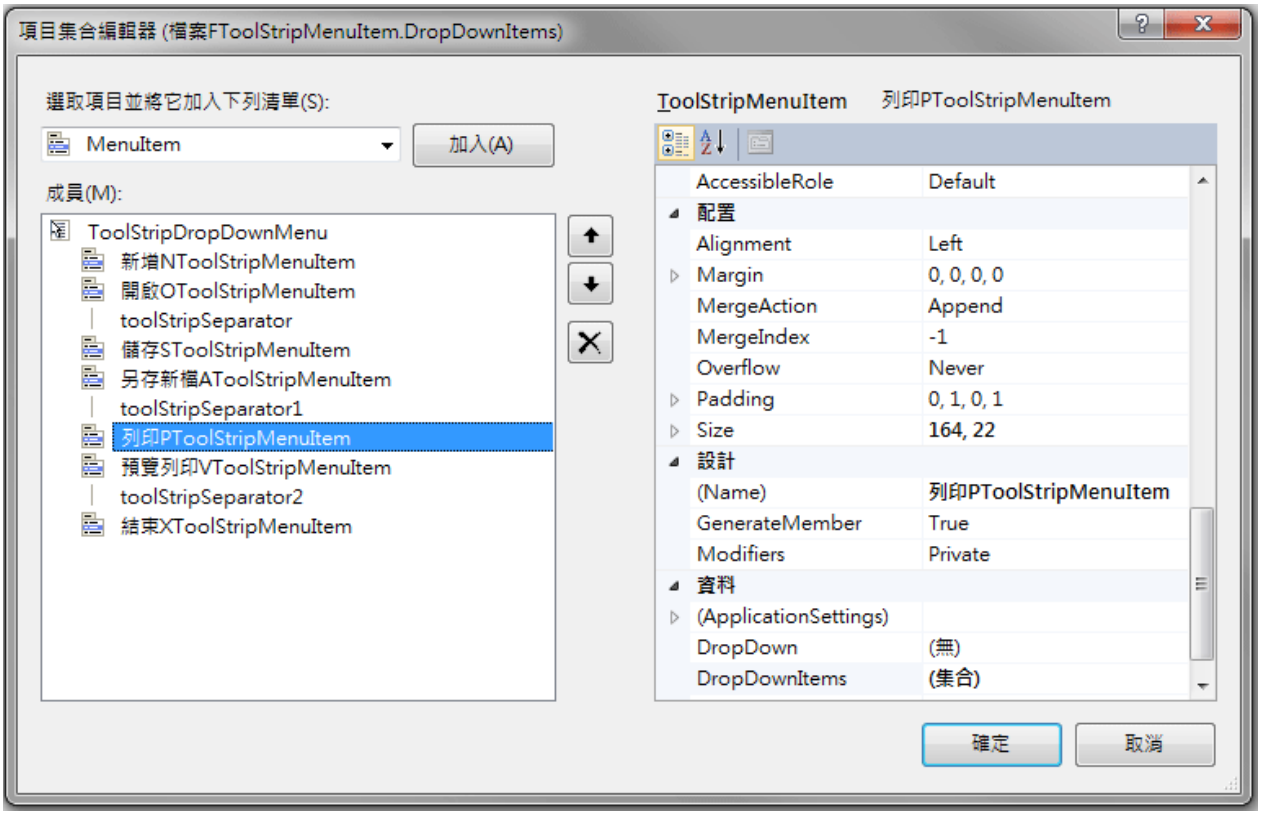
其中 Application 的意義是指此『應用程式』，Exit 就是離開或結束的意思。如果我們寫成 this.Close()，意思則是關閉(Close)本表單(this)，在此例中效果是一樣的，都可以完全結束程式。但是兩者涵義稍有不同。Application 是指整個專案程式，this 是指本表單，而一個專案程式可能有多個表單，關閉其中一個未必關閉得了其他的表單。但是如果專案只有一個表單，或者你關閉的是啟動時第一個出現的「主表單」，還是可以完全關閉程式的。在此建議初學者應建立正確的觀念，使用 Application.Exit 比較正確！

4-4 編輯功能表

[使用功能表編輯器]

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？首先你可以直接在選單項目下的空白處新增項目，也可以對現有項目按右鍵在出現的選單中刪除它們。但更好的方式是使用編輯器。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「toolStripSeparator」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；加入項目時如果用編輯器加入預設名稱(Name)會是「toolStripMenuItem1」，這樣不好辨識，你可以將它和文字屬性 Text 都改成「搜尋取代」！用中文命名可以嗎？目前的 C#內建使用 Unicode 編碼，用日文都可以的！

接著再將「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

[呼叫 TextBox 物件的編輯功能]

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 void 剪下TToolStripMenuItem_Click(object  sender, EventArgs e)
{
    textBox1.Cut();
}

private void 複製CToolStripMenuItem_Click(object  sender, EventArgs e)
{
    textBox1.Copy();
}

private void 貼上PToolStripMenuItem_Click(object  sender, EventArgs e)
{
    textBox1.Paste();
}

private void 全選AToolStripMenuItem_Click(object  sender, EventArgs e)
{
    textBox1.SelectAll();
}
```

非常簡單，都是一個指令就完成一個操作，因為它們都是 TextBox 物件原本就有的內建功能！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。

[搜尋與取代的操作介面]

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 Panel 物件，再到這容器內建立兩個 TextBox 以及三個按鍵，佈置如下圖：

先將上面的 panel1 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 void 搜尋取代_Click(object  sender, EventArgs e)
{
    panel1.Visible = true;
}

private void button3_Click(object  sender, EventArgs e)
{
    panel1.Visible = false;
}
```

請先測試一下這個 panel1 是不是可以正確的開關！如果沒問題，請寫「搜尋」按鍵的程式如下：

[搜尋的程式碼]

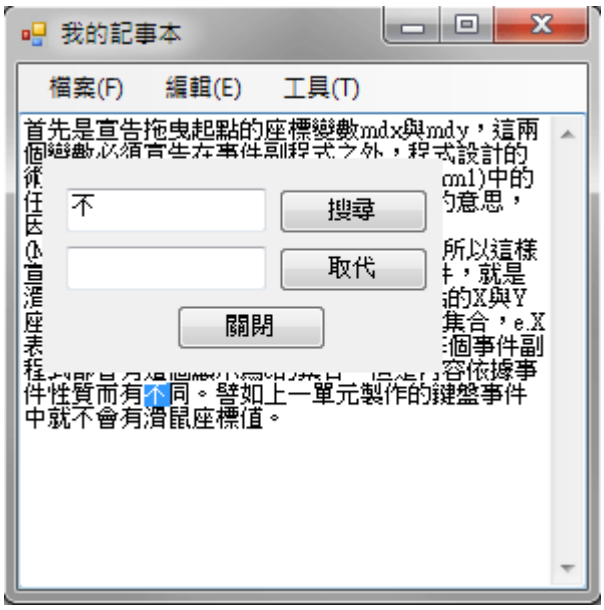
```
private void button1_Click_1(object  sender, EventArgs e)
{
    int P;
    if (textBox1.SelectionLength  > 0)
    {
        P = textBox1.Text.IndexOf(textBox2.Text,  textBox1.SelectionStart  + 1);
    }
    else
    {
        P = textBox1.Text.IndexOf(textBox2.Text,  textBox1.SelectionStart);
    }           if (P < 0)
    {
        MessageBox.Show("未發現搜尋字串！");
    }
    else
    {
        textBox1.SelectionStart  = P;
        textBox1.SelectionLength  = textBox2.TextLength;
        textBox1.Select();
    }
}
```

這段程式首先宣告一個指標變數 P，代表被搜到字串的起始位置。在 A 字串(textBox1.Text)中搜尋 B 字串(textBox2.Text)的語法是 A.IndexOf(B, StartIndex)，所謂的 Index 是索引，每個字串中的第一個字索引值是 0，第二個字是 1，依此類推！IndexOf 這個函式會回應 B 字串在 A 字串中起始字元的索引位置，如果沒有發現符合的字串就會回傳-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最開始(索引 0)處開始搜尋，這樣你就永遠只能搜尋到字串中「第一個」符合條件的目標，無法繼續找下一個符合條件的目標，也就是一般軟體中常見的『下一筆』了！

上面的程式一開始檢查目前的 textBox1.Text 中有沒有已經被選取的文字？(SelectionLength>0)如果有的話表示之前已經找到目標字串，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(SelectionStart，也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！

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(MessageBox.Show)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！操作畫面如下：



[取代的程式碼]

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：

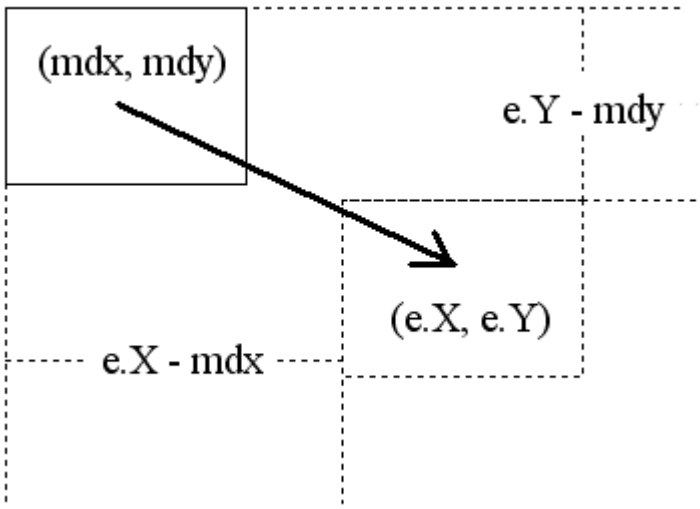
```
private void button2_Click(object  sender, EventArgs e)
{
    textBox1.SelectedText  =  textBox3.Text;
}
```

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

[拖曳物件的程式]

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 panel1 無法移動，常常會遮住想看的文字。我們可以使用 panel1 的 MouseDown 與 MouseMove 事件製作一個拖曳程式即可。用滑鼠拖曳物件的座標示意圖如下，其中的 mdx 與 mdy 代表滑鼠壓下左鍵時的像素點座標，e.X 與 e.Y 代表滑鼠移動的新位置座標。



首先請到設計頁面點選 panel1，再到它的屬性視窗點選黃色閃電標記到事件選單選擇 MouseDown 與 MouseMove 產生程式框架後寫程式如下：

```
int mdx, mdy;
private void panel1_MouseDown(object  sender, MouseEventArgs e)
{
    mdx = e.X;
    mdy = e.Y;
}

private void panel1_MouseMove(object  sender, MouseEventArgs e)
{
    if (e.Button == System.Windows.Forms.MouseButtons.Left)
    {
        panel1.Left += e.X - mdx;
        panel1.Top += e.Y - mdy;
    }
}
```

程式碼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，程式設計的術語叫作「全域變數」，就是整個程式(Form1)中的任何一個副程式都可以辨識並使用這種變數的意思。因為我們必須在之後的兩個事件副程式(MouseDown 與 MouseMove)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)的時候記錄起點的 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 是滑鼠 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(e.X & e.Y)。

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(panel1)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，現在的搜

尋視窗可以被拖動了！

[可以簡化的名稱]

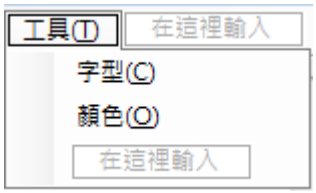
或許程式碼中的 System.Windows.Forms.MouseButtons.Left 會有點嚇到初學者，需要背這麼長的名稱嗎？其實不必的！首先只要你之前的程式沒打錯就會有自動提示，而且如果你省略 System.Windows.Forms，只寫 MouseButtons.Left 也可以的！System.Windows.Forms 代表的是函式庫項目或叫命名空間(Namespace)，你可以看到程式碼頁面最前面已經有以下程式，就表示該函式庫(命名空間)已經匯入，不寫全名，C#也可以看懂的！

```
using System.Windows.Forms;
```

4-7 字型與顏色的選擇

[使用字型與顏色對話方塊]

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，本單元卻要製作看看，主要目的是藉此介紹一下顏色選擇的物件。首先你應該如前面提到的，先將功能表的「工具」選項內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：

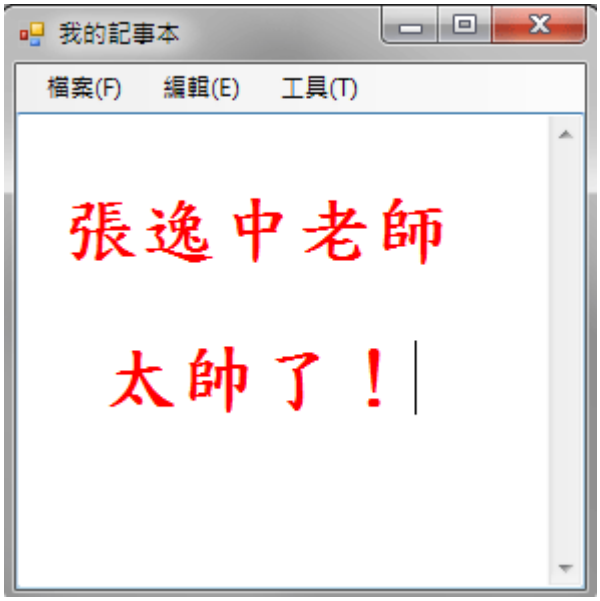
```
private void 字型CToolStripMenuItem_Click(object  sender, EventArgs e)
{
    fontDialog1.ShowDialog();
    textBox1.Font  = fontDialog1.Font;
}

private void 顏色OToolStripMenuItem_Click(object  sender, EventArgs e)
{
    colorDialog1.ShowDialog();
    textBox1.ForeColor  = colorDialog1.Color;
}
```

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，費事的處理使用者是否按下確定(OK)或取消(Cancel)的狀況，原因是之前不正確的檔名確實可能讓程式當

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設設的字型或顏色，程式也不會當掉，所以沒有關係的。試試看，你的文字可能會變成這樣：



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

淺談.NET Framework Library

本單元中我們第一次碰觸到了程式碼頁面中的 using 指令區，如同之前談過的：現代的程式其實是由很多物件所組成，這些「物件」嚴格說都是功能較單純的許多小程式。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)，我們目前使用的 C# 2010 軟體背後支援的函式庫就是「NET Framwork 4.0」版。

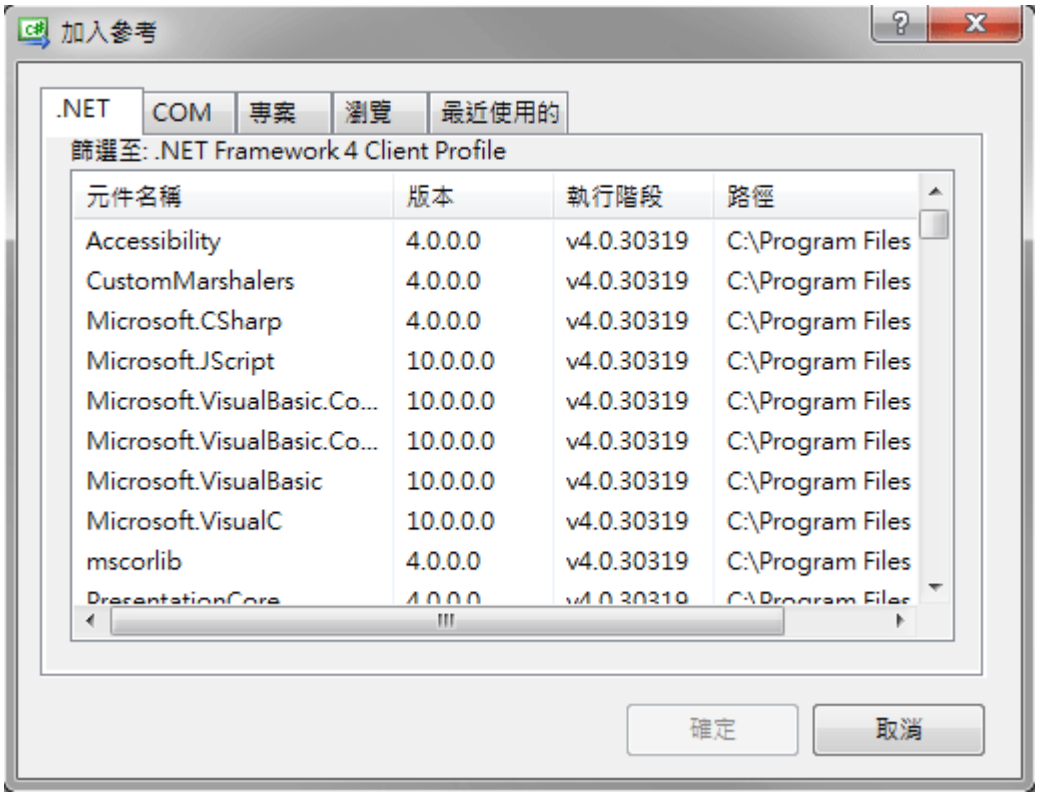
當我們開啟新的程式專案時，Visual Studio 會預設載入一些基本常用的物件與函式，就是程式碼頁面最上面的許多 using 項目，如：

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
……
```

這些程式碼的意義是本表單，也就是本檔案，可以使用這些引用的程式集，標準名稱為『命名空間』(NameSpace)。如果你用到某些功能不在這些程式集合之中，你編寫程式時就不會自動出現提示，程式執行時也會說看不懂這個指令！當然你還是可以使用全名呼叫它，如本單元中用到的 File 物件，如果不匯入 System.IO 就必須寫成：System.IO.File。

[加入參考]

另一種匯入命名空間的方式是找到方案總管視窗中的的『**參考**』項目，按右鍵選擇「加入參考」；或者到主功能表的「專案」項目選擇「加入參考」都可以看到如下的視窗：



在此就可以選擇加入參考的命名空間或程式庫，它甚至可以超出.NET Framework 的範圍，譬如『COM』選項是指按照微軟較舊的 COM 格製作的程式庫，到瀏覽或專案的位置還可以選擇加入任何符合規格的自製函式庫。

[讓 MSDN 幫你學習]

一般初學者甚至老手都不可能完全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。在 VB 程式語言中有 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！但是目前 C#還沒有這種機制，我們要使用各種特別功能時通常必須仰賴一些既有範例與線上說明，此時最好的朋友可能是 MSDN(Microsoft Developer Network)，這是微軟公司為程式開發者建立的龐大的網頁化的說明文件，對於不認識的物件或文字，選取之後按下 F1 按鍵就可以開始進行學習之旅了，或者你到網路瀏覽器搜尋 MSDN 也可以直接找到這個龐大的網路資源。

總之，在此必須知道的概念是：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)，任何資源都是可以使用的！我們不可能也不必認識所有的資源，但是必須知道一些找尋資源的方法，網路是最方便的途徑，但是專業文件對初學者來說並不容易理解，因此找到好的書本參考，認真上課勤問老師還是初學者最佳的學習策略。