第 17 章　系統通知區的程式

簡介：

本單元將建立一個沒有典型視窗只以一個圖示顯示於系統通知區的程式，也就是類似 MSN 或防毒軟體一般具有背景常駐程式外觀的軟體。基本上是使用 NotifyIcon 物件，以一個系統通知區圖示代替主表單，搭配跳出式選單操作。製作的兩個主要功能是產生類似便利貼的備忘貼紙，以及擷取目前螢幕的影像存為影像檔案。除了系統通知區程式的特殊性之外，這也是我們第一次嘗試製作的多表單程式專案。換言之，連表單在此單元也成為可以動態產生的物件了！

17-1 建立一個沒有主視窗的程式

[初識 NotifyIcon 物件]

請開啓新專案，加入一個工具箱中「通用控制項」分類的 NotifyIcon 物件，設定其 Icon 屬性為某個 Icon 圖示檔案，譬如 。立即執行程式就可以看到除了正常的表單之外，在螢幕右下角的系統通知區會出現這個圖示，畫面大致如下：

[隱藏主表單的方法]

但是它依然有正常視窗程式的外觀，也就是一個典型的表單(視窗)物件。接下來我們要完全隱藏這個主表單。首先你會發現 Form1 並沒有 Visible 的屬性，不能直接設定 Visible=False 來隱藏它！正確的作法是先將表單的 WindowState 屬性設為 Minimized，此時表單會縮小到工作列(Taskbar)，但這還不算隱藏，使用者仍然可以點選它再度開啟為正常表單。所以必須進一步將表單的 ShowInTaskbar 屬性設定為 False！就是讓它連在工作列(Taskbar)都不會顯示，最後整個程式的對外介面就只剩下右下方系統通知區的一個圖示了！

17-2 建立功能選單

[建立貼紙表單 Form2]

系統通知區程式的啟動通常必須依靠點選滑鼠右鍵出現的跳出式選單，本階段設計目標是在跳出式選單中可以選擇產生新的貼紙或者結束程式。所謂「新的貼紙」其實是一個「新的表單」，所以請先在專案功能表中選擇「專案」→「加入 Windows Form」，可以產生一個預設稱為 Form2 的新表單，我們預備稍後將它作為貼紙表單的範本。

[為 NotifyIcon 加入跳出式選單]

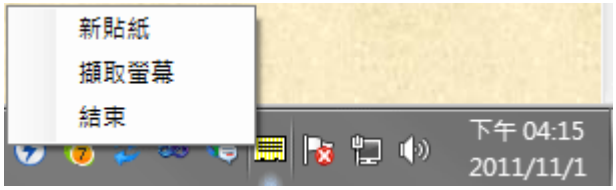
接著請切換回到 Form1 表單，至工具箱的功能表與工具列分類中選擇加入一個

contextMenuStrip1(跳出式選單)；接著在 NotifyIcon 物件的 ContextMenuStrip 屬性將此選單(ContextMenuStrip1)選入；選單中請建立「新貼紙」、「擷取螢幕」與「結束」三個項目，並先寫入其中兩個功能程式碼如下圖：


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

private void 新貼紙ToolStripMenuItem_Click(object  sender, EventArgs e)
{
    Form2 f =new Form2();
    f.Show();
}
```

[用 new Form2 製造新表單]

程式碼中的 Application.Exit()表示讓整個應用程式(Application)結束(Exit)；新貼紙部分則是宣告並建置(使用 new 關鍵字)一個名為 f 的 Form2 物件，並顯示它(Show)。在此 Form2 是個「類別」的名稱，f 則是依據 Form2 定義為模型產生的實體，如果這個副程式執行多次就可以產生多個實體，也就是很多張新貼紙了！作好上述程式與設定，執行程式，在系統通知區圖示上按下右鍵就可以看到選單，畫面如下：



試試看是否可以選擇『新貼紙』新增多個 Form2 表單，以及結束程式。

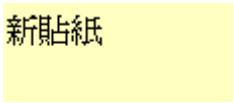
17-3 貼紙表單的設計

[讓 Form2 外觀像貼紙]

貼紙表單的主要功能是要可以寫字，還要可以拖曳以及縮放大小，以便使用者安排它到桌面的任何適當位置。如果只是這樣，我們在表單中放一個 TextBox，讓它看起來像個記事本的視窗就可以了！但是比較有趣的部分是我們想將它作成一張「貼紙」的外觀，讓使用者認為它不是一個普通的視窗。

首先我們將 Form2 表單的 FormBorderStyle 屬性改為 None，讓視窗邊框消失；再加入

一個 TextBox 物件將其 Multiline 屬性設為 True(變成可多行編輯)，Dock = Fill (填滿表單)，還可以給個適當的背景色。這時候表單看起來就真的像一張貼紙了！執行的時候也確實可以寫字。最後還必須將表單的「ShowInTaskbar」屬性設為 False，如此新貼紙表單產生時就不會在工作列顯示，讓使用者更不覺得它是一個視窗了！做好的貼紙外觀寫上幾個字可能會是這樣：



[讓貼紙可以被拖曳]

接下來要讓這個表單可以被拖曳，請在 Form2 表單中先建立 textBox1 的 MouseDown 與 MouseMove 事件副程式框架，並寫程式如下：(請注意是在「Form2」寫程式，不是 Form1 哦！)

```
int mdx, mdy;
private void textBox1_MouseDown(object  sender, MouseEventArgs  e)
{
    mdx = e.X;
    mdy = e.Y;
}

private void textBox1_MouseMove(object  sender, MouseEventArgs  e)
{
    if(e.Button == MouseButtons.Left)
    {
        this.Left += e.X - mdx;
        this.Top += e.Y - mdy;
    }
}
```

其中 mdx 與 mdy 全域變數紀錄開始拖曳(MouseDown)時的起點座標，在滑鼠拖曳(MouseMove)時本表單(this)的位置就會跟著移動，類似程式在之前多個單元中都有學過，應該不難。比較特別的是這個拖曳事件的 MouseDown 與 MouseMove 事件是屬於 textBox1，因為實際被點選的物件確實是 textBox1，但是我們希望移動的物件卻是整張表單"this"(在此是指 Form2 而非 Form1 哦！)，所以 MouseMove 事件中最後被移動的是表單"this"，只是 textBox1 事實上在表單裡面，所以移動表單也等於是移動 textBox1 了。

17-4 貼紙表單的大小調整

[製作調整大小的熱點]

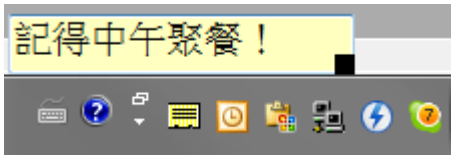
除了拖曳之外，貼紙本身的大小也要可以調整，最簡單的設計方式就是模仿小畫家中圖紙大小調整的介面，在圖紙角落設計一個小黑方塊(熱點)，拖曳這個方塊時圖紙大小也跟著改變。這個方塊的作法可以使用一個標籤(lable1)，將：AutoSize 屬性設為 False；背景色(BackColor)設為黑色；文字(Text)刪除；長寬設為 10 個像素點；游標(Cursor)設為

SizeNWSE；位置則置於表單的最右下方。如下圖：

其中游標的選擇是一個左上(NW)到右下(SE)的雙箭頭，讓滑鼠一移到這個物件上時使用者就知道可以左右或上下拖曳了。接著對此物件(label1)寫拖曳的程式如下。試試看！應該可以改變貼紙的大小了！

```
private void label1_MouseDown(object  sender, MouseEventArgs e)
{
    mdx = e.X;
    mdy = e.Y;
}

private void label1_MouseMove(object  sender, MouseEventArgs e)
{
    if(e.Button == MouseButtons.Left)
    {
        label1.Left += e.X - mdx;
        label1.Top += e.Y - mdy;
        this.Width = label1.Right;
        this.Height = label1.Bottom;
    }
}
```

執行以上程式，增加新貼紙，記入一則提示：「記得中午聚餐！」，放到桌面右下方，畫面概略如下圖：



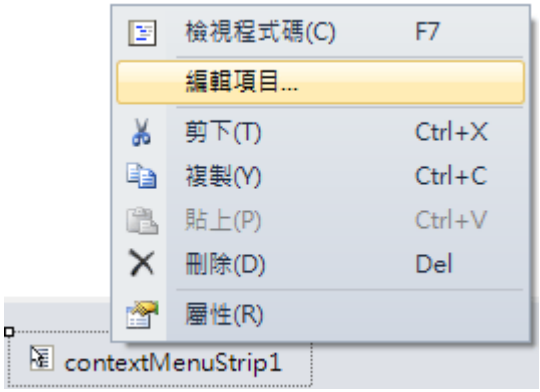
17-5 貼紙顏色、文字格式與刪除貼紙

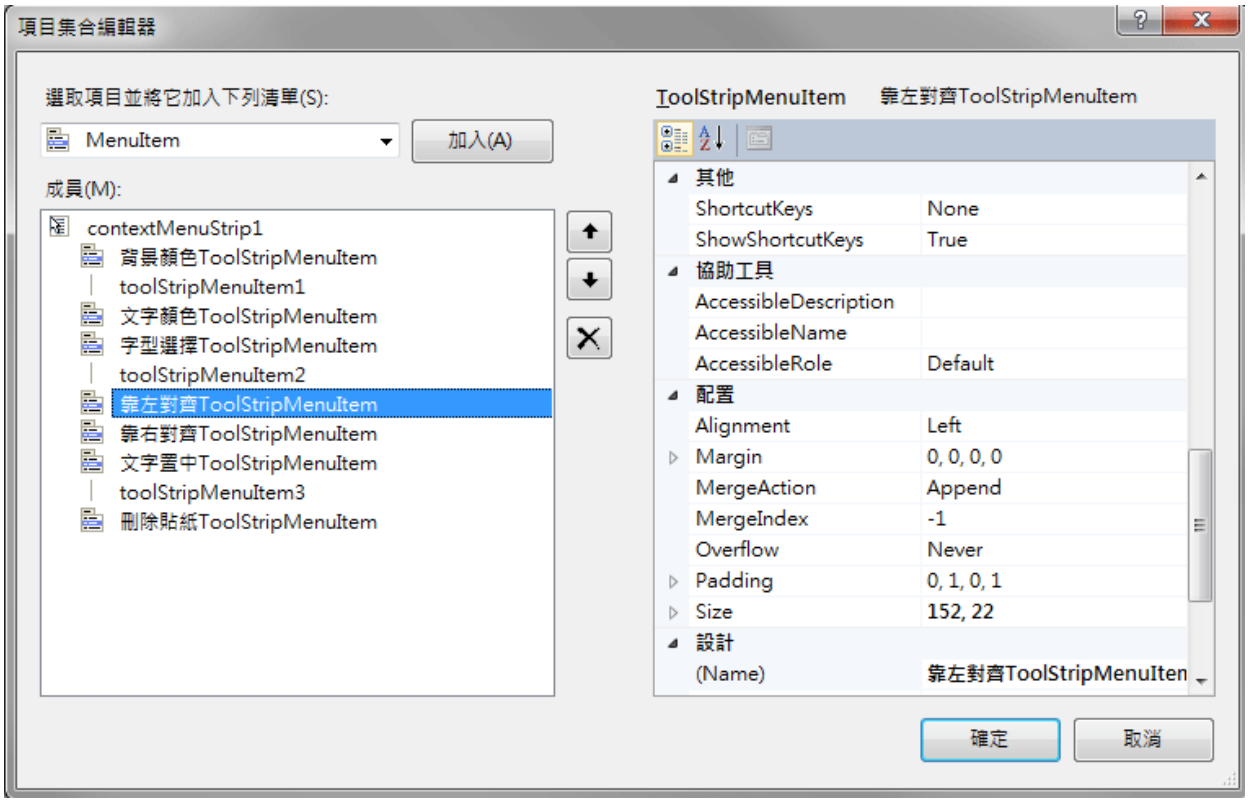
[設計跳出式選單]

除了拖曳與縮放貼紙，我們常常也希望貼紙可以變色、字體可以改變大小與顏色、文字靠左、靠右或置中等等，最重要的是必須可以在不停止整個程式的前提下「刪除」某一張貼紙。這些瑣碎的功能可以集中設計到一個貼紙專用的跳出式選單來完成。請先切換到代表貼紙的 Form2 表單，先叫用一個顏色對話方塊 colorDialog1 準備作為顏色選擇之用，再叫用一個 fontDialog1 作為字型選擇之用，再加入一個 contextMenuStrip1，並在其中設

計下列項目：

編輯項目時除了直接寫入功能項目(MenuItem)之外也可以如上圖選擇加入格線(Separator)，甚至下拉式選單(ComboBox)或文字方塊(TextBox)，要加入格線時可以直接在輸入窗格內寫『-』號按下輸入鍵即可。此例中我們就加入了幾條格線，以區分功能類別。除此之外，也可以對著 contextMenuStrip1 的標題按下滑鼠右鍵選擇「編輯項目」，會出現如下介面，可以更加方便全面的調整所有功能項目的相對位置與內容：





[改變字型與顏色的程式]

上述功能表(contextMenuStrip1)必須依附於 textBox1 物件，也就是 textBox1 的 ContextMenuStrip 屬性須設定為 contextMenuStrip1。使用者在貼紙上按下滑鼠右鍵就可以看到這些選項。設定背景顏色與文字顏色的程式是先呼叫選色對話方塊，接著將使用者選的顏色填入 textBox1 的 BackColor(背景色)或 ForeColor(字的顏色)；字型選擇程式也類似，呼叫字型選擇方塊(fontDialog1)。如果使用者未選擇顏色字型，直接按取消時，就會設定為 colorDialog1 與 fontDialog1 的預設顏色與字型。

```
private void 背景顏色ToolStripMenuItem_Click(object  sender, EventArgs e)
{
    colorDialog1.ShowDialog();
    textBox1.BackColor  = colorDialog1.Color;
}

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

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

[文字對齊的程式]

文字對齊部分則是直接設定 textBox1 的 TextAlign 屬性，預設常數中就有靠左右對齊與置中等選項！

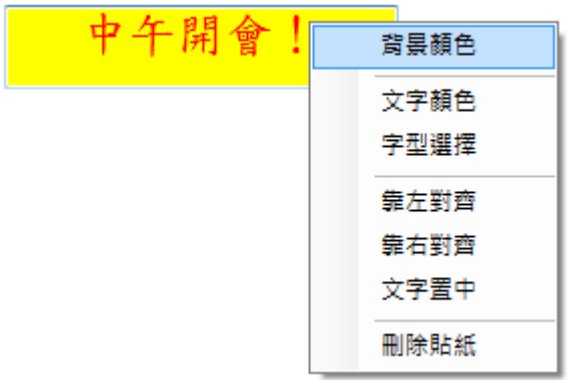
```
private void 靠左對齊ToolStripMenuItem_Click(object  sender, EventArgs e)
{
    textBox1.TextAlign  = HorizontalAlignment.Left;
}

private void 靠右對齊ToolStripMenuItem_Click(object  sender, EventArgs e)
{
    textBox1.TextAlign  = HorizontalAlignment.Right;
}

private void 置中ToolStripMenuItem_Click(object  sender, EventArgs e)
{
    textBox1.TextAlign  = HorizontalAlignment.Center;
}
```

[刪除貼紙的程式]

「刪除」在此使用 this,Dispose()或 this.Close()感覺效果都一樣，但是使用 Dispose 明確的將表單自記體中刪除較佳，Close 意謂關閉，與 Visible=False 意義相似，可能會增加記憶體無謂的負擔。請注意 this 在此是指由新貼紙功能產生的其中一個「Form2」表單實體，而不是「Form1」主表單！所以不論是 Dispose 或 Close 都與程式主體的 Form1 或其他的貼紙(Form2 實體)無關。

```
private void 刪除貼紙ToolStripMenuItem_Click(object  sender, EventArgs e)
{
    this.Dispose();
}
```

程式完成後執行畫面範例如下：



17-6 螢幕擷取程式

[Print Scr 按鍵的程式版]

很多人都會碰到想將螢幕畫面抓下來的時候，也有很多人知道按下鍵盤的 PrintScr 鍵

(螢幕列印)就可以將螢幕影像拷貝(到 Windows 系統剪貼簿)，之後打開小畫家貼上圖片存檔就搞定了！但是這種動作可不可以自己寫程式執行呢？就是不必這麼麻煩去中斷工作打開小畫家另存新檔，只要點一點滑鼠或按個鍵盤圖片就直接變成一個檔案？連續按還可以連續抓圖存檔不會重複！其實有這種軟體的，有些還免費，但是核心技術其實很簡單，我們自己做就可以了！

請按下之前在 Form1 設計的跳出式選單中的『擷取螢幕』功能，先寫程式如下：

```
private void 擷取螢幕ToolStripMenuItem_Click(object  sender, EventArgs e)
{
    int W = Screen.PrimaryScreen.Bounds.Width;
    int H = Screen.PrimaryScreen.Bounds.Height;
    Bitmap bmp = new Bitmap(W, H);
    Graphics g = Graphics.FromImage(bmp);
    Size S = new Size(W, H);
    g.CopyFromScreen(0,  0, 0, 0, S);
    bmp.Save(@"D:\tmp.jpg",  System.Drawing.Imaging.ImageFormat.Jpeg);
}
```

測試一下是不是你按下擷取螢幕之後，到 D 磁碟就可以看到一個 tmp.jpg 檔案，內容就是你剛剛執行程式時的螢幕畫面？解釋一下程式碼的作用：首先是取得目前螢幕的寬(W)與高(H)，方法是讀取螢幕(Screen)物件的邊界(Bounds)屬性中的寬(Width)與高(Height)；接著準備好一個影像(Bitmap)物件 bmp，寬高與螢幕相同，再宣告一個此 bmp 影像的繪圖物件 g，重點就是使用此繪圖物件 g 去拷貝螢幕(CopyFromScreen)，螢幕影像就複製到 bmp 了！

[逸出字元(\)與檔案路徑名稱的衝突處理]

接著是存檔案的動作，比較特別的是@這個記號，這在本書我的瀏覽器單元 2-4 節中解釋過，原因是在 C 語言中"\"這個字碼稱為逸出(Escape)字元，不加上@程式會認為它有特殊的意義，譬如"\n"代表換行等等，加上@就是告訴 C#不要想太多，此處的"\"只是磁碟目錄名稱的前置符號而已！所以最後一行程式的意義是：將 bmp 存到"D:\tmp.jpg"這個檔案，格式使用 Jpeg，與一般常用的 jpg 意義相同。事實上 JPEG 是制定此格式的一個協會名稱的縮寫，會變成 jpg 是為了適應 DOS 時代的檔名規則，規定副檔名最多三個字元的限制，現在你用兩種寫法都可以的，意義完全相同！

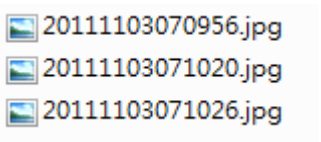
17-7 自動產生不重複的檔名

[用時間當作檔名]

上面的程式確實只要滑鼠點選就可以自動產生螢幕擷取檔案，但是有個小問題：重複擷取時會將 tmp.jpg 覆蓋，如果希望連續取得多張畫面時就不適用了！最簡單的解決方式就是讓存檔時間變成檔名的一部分，請將上面的 bmp.Save 存檔程式變成以下幾行：

```
DateTime D = DateTime.Now;
String fn = @"D:\"
    + D.ToString("yyyy")//年
    + D.ToString("MM")//月
    + D.ToString("dd")//日
    + D.ToString("HH")//時(24小p時制)
    + D.ToString("mm")//分
    + D.ToString("ss")//秒
    + ".jpg";
bmp.Save(fn, System.Drawing.Imaging.ImageFormat.Jpeg);
```

上面程式先宣告一個日期時間(DateTime)物件 D 代表現在(Now)的時間，接著建立檔名字串 fn，除了目錄與 jpg 副檔名和原來一樣之外，主檔名就用年月日時分秒表示。困難一點的是:如果直接用 D.ToString()的方法產生完整的日期時間字串大概像這樣:『2011/11/6 上午 08:00』，這不是一個合法的檔名，可能是其中有冒號或斜線等字元吧？所以必須麻煩一點，年月日時分秒一項一項的處理，剛好也是一個完整的格式化時間函數的展示！如 D.ToString("MM")代表兩個字元的月表示法，如六月會變成"06"，D.ToString("mm")則代表分(minute)等等。最終連續存檔實結果大致是這樣的：



[桌面的磁碟路徑寫法]

但是如果我們想將擷取的圖檔放在桌面，而不是 D 磁碟怎麼辦？『桌面』或者『我的文件』等特殊的資料夾就不像"D:\ABC\XYZ"這種目錄這麼明確了！因為系統內可能有多個使用者，各自有不同的目錄擺設他們自己桌面的檔案。這對 C#程式來說有一點刁難，因為在 VB 語言中有 My 捷徑，桌面目錄這樣寫就可以了：

My.Computer.FileSystem.SpecialDirectories.Desktop

但是既然 VB 與 C#是用同一個基底函式庫(.NET Framework)，VB 能 C#也一定能！C#的桌面目錄應該是這樣寫的(VB 應該也通用)：

Environment.GetFolderPath(Environment.SpecialFolder.Destkop))

Evivironment 是環境物件的意思，這個物件就包含了許多與作業系統環境相關的方法與屬性，上述程式就是使用 GetFolderPath(取得目錄路徑)方法取得系統(Environment)特殊目錄(SpecialFolder)桌面(Destkop)的實體路徑。所以如果你將程式改成這樣就可以將擷取畫面放在桌面了！

```
private void 擷取螢幕ToolStripMenuItem_Click(object  sender, EventArgs e)
{
    int W = Screen.PrimaryScreen.Bounds.Width;
    int H = Screen.PrimaryScreen.Bounds.Height;
    Bitmap bmp = new Bitmap(W, H);
    Graphics g = Graphics.FromImage(bmp);
    Size S = new Size(W, H);
    g.CopyFromScreen(0,  0, 0, 0, S);
    DateTime D = DateTime.Now;
    String desktop = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);
    String fn = desktop + "\\" //桌面目錄路徑
        + D.ToString("yyyy")//年
        + D.ToString("MM")//月
        + D.ToString("dd")//日
        + D.ToString("HH")//時(24小時制)
        + D.ToString("mm")//分
        + D.ToString("ss")//秒
        + ".jpg";
    bmp.Save(fn, System.Drawing.Imaging.ImageFormat.Jpeg);
}
```

[處理逸出字元(\)的另一個方法]

問題又來了！為什麼 desktop 的後面要加上"\\"呢？其實又是逸出字元的問題，上面程式取得的 desktop，如果在我的電腦系統裡面直接寫出來其實是這樣的：

C:\\Users\\YCC\\Desktop

所有原來在 Windows 作業系統下的"\"字元都會被轉換成"\\"，這也是告訴 C#這不是逸出字元的一個方式，當然在此字串之後，要加上檔名之前，也必須自己寫程式加上一個"\\"記號了！由此可知，我們之前用@告訴 C#忽略"/"為逸出字元的方法，其實也可以用"\\"取代"\"達到一樣的效果。有點麻煩，但是誰說寫專業程式沒代價呢？好家在，現在你可以像專家一樣，不必將程式死死的只能放在 C 或 D 槽某目錄了。

17-8 進階挑戰

一、如何產生月曆功能的貼紙？

提示：建立另一個表單貼上月曆物件 monthCalendar，如貼紙般叫用即可。

二、如何製作一個鬧鐘或碼表物件？

提示：建立新表單以 Timer 物件設計倒數計時等功能即可。

三、如何將其他遊戲程式加入？

提示:可使用功能表加入現有項目之功能匯入其他程式的表單,但需先改名以免衝突。

課後閱讀

多表單的程式

[為何需要多表單？]

這個單元除了製作出不像視窗的貼紙視窗之外，最特別的是在一個專案中我們製作了不只一個，而是「兩個」表單！雖然多數時候我們學寫程式好像使用一個表單就夠了，但是實用的系統中，一個軟體擁有多達數十個表單才是正常的事情！譬如以一個公司的資料管理系統來說：可能需要顯示員工資料、薪資資料、貨物資料、客戶資料、交易明細，還有各種需要列印的報(告)表，如果要將所有功能都集中在一個視窗處理不是不行，但是就必須在同一表單上安排幾百(上千)個物件，而且要隨著處理的工作不同，隱藏大部分的物件，只顯示目前要用的一部分。

這些都需要很費事的安排物件位置，還必須寫程式控制誰該上場誰該退場，加上每種作業本身也都需要很多資料處理的程式碼，結果就是你的表單東西多到如同垃圾堆！程式碼也多到幾千行像一本小說，想找到某個副程式都很困難！也因為所有物件與變數都擠在同一程式單元之內，資料衝突的情況也非常容易發生。此時多表單的架構就是你的救命恩人了！

多表單的共用變數

[沒有模組怎麼辦？]

一旦用了多表單可能會產生一個新的需求，就是表單之間的資料溝通與交換。譬如某人在 Form1 登入身分為 A 先生之後，程式執行中如果用到 Form2 時，程式如何知道他仍然是 A 先生？必須有個變數層級是每個表單都看得到的！在 VB 程式中可以建立一個模組(Module)將共用的變數在那邊以 Public 宣告即可，但是 C#並沒有模組這種東西，雖然這是個比較進階的議題，但是說實話如果表單之間都不能溝通，學習多表單程式和多個單一表單程式又有何不同？

[C#的靜態類別]

答案是在 C#中必須使用靜態類別(Static Class)來建立專案層級的變數，請打開本單元專案，選取功能表的專案→加入類別，會產生一個 Class1 的檔案，程式碼如下：

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CsPostIt
{
    class Class1
    {
    }
}
```

請稍微修改一下 class Class1 的標題與內容變成：

```
public static class Class1
{
    public string S = "Hello!";
}
```

此時變數 s 就是一個所有表單都看得見的共用變數了！你可以在 Form1_Load 與 Form2_Load 事件都寫下面的程式測試一下！證明 Form1 與 Form2 都可以看到這個變數的，甚至也可以修改它！

```
private void Form1_Load(object sender, EventArgs e)
{
    MessageBox.Show(Class1.S);
    Class1.S = "Hi!";
    MessageBox.Show(Class1.S);
}
```

[public 與 private 的差異]

這裡出現的重要關鍵字有兩個！一是 public 表示公開公用的意思，專案中各個表單都可以使用；相對的，就是 private 表示只在本類別內有用的意思。另一個關鍵字是 static，表示本類別不需要(事實上也不允許)建立實體，要用時直接呼叫類別名稱就好了！如上面寫的 Class1.S，如果你寫成 Class1 C = new Class1(); 反而是錯的！

C#號稱是相當完美的物件導向語言，有關物件導向設計的各種功能(介面、實作、繼承、多載或覆寫等等)是許多 C#語言書介紹的重頭戲，但是不好意思，作者才疏學淺，雖然已經作了不少實務案例，仍然覺得多數艱深的類別操作技巧其實極少使用，就偷懶不作深入介紹了！但是靜態類別是打通多表單任督二脈的重要關鍵技術，則是不可不知的一招！

原來程式可以管到表單之外欸！

[程式可以讀取系統資訊]

本單元另一個驚喜應該是擷取螢幕的程式，特別之處是原來視窗程式可以取得表單視窗以外的東西！整個螢幕也有一個 Screen 物件可以用！還有 Environment 可以取得系統的其他資訊咧！好好找一找，連電腦記憶體、CPU 型號與數量也都是可以讀取的。

[不能改變記憶體內容]

在以往的 DOS 時代，個別程式連作業系統的時間，甚至別的程式使用的記憶體都可以直接讀寫修改！這樣其實很危險，所以常常會造成程式與作業系統之間工作的衝突而當機！當然也不容易安全的同時執行多個程式而不互相干擾。所以到了 Windows 作業系統的時代，個別程式的權限縮小了，要使用記憶體都得向作業系統程式『申請』！不能直接讀寫記憶體這件事情讓很多早期習慣用 C 語言的人非常不爽！因為讓 C 語言具有最高效能的指標變數技巧至此全都白學了？

[可以啟動其他程式]

『侵入』作業系統或其他程式的管區，當然是不妥的，如果你可以做到就算駭客了！但是現在的程式基本上仍然可以自由地取得所有的系統資訊，也可以和其他的程式之間進行『通訊』，就是交換資訊的！甚至可以呼叫啟動其他的程式！試試看在擷取螢幕程式的最後加上一行：

System.Diagnostics.Process.Start(fn);

你會發現擷取的圖檔(fn)立即被系統預設開啟 jpg 圖檔的程式(如小畫家等等)開啟給你看！效果與你直接到檔案總管點擊該檔案開圖一樣！本書實質內容的量或許不如一般程式語言工具書，但是很希望藉著一些小的範例隨時提醒您：程式設計無遠弗屆的可能性！