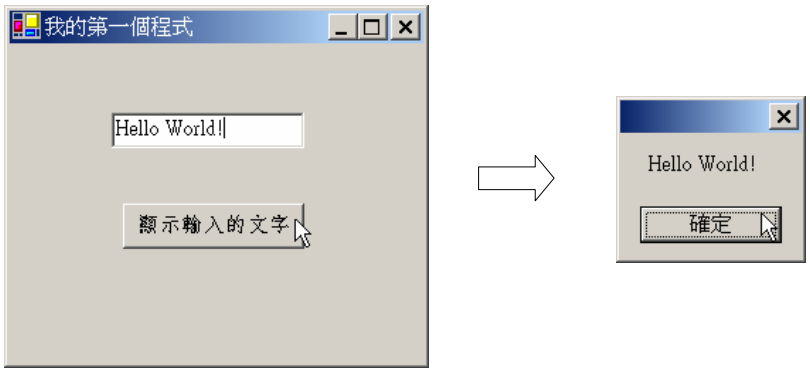
# C#.NET 程式設計

第三章　開始撰寫 Visual C#.NET 程式

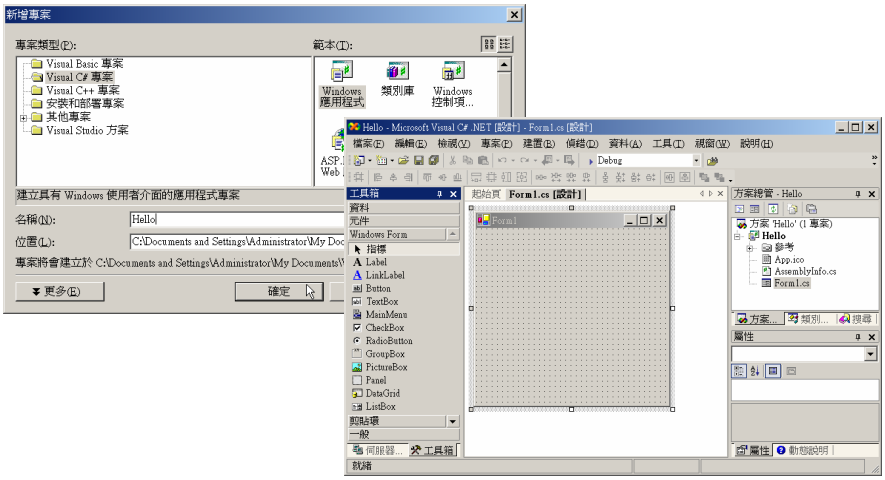
## 3-1　Visual C#.NET 程式設計流程

1. 分析問題與需求
2. 設計演算法與執行流程
3. 在表單上放置控制項並調整屬性
4. 爲控制項撰寫程式碼
5. 偵錯與測試

# 3-2 您的第一個 Visual C#.NET 程式

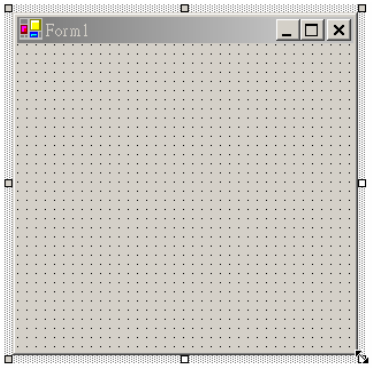



# 3-2-1 新增專案

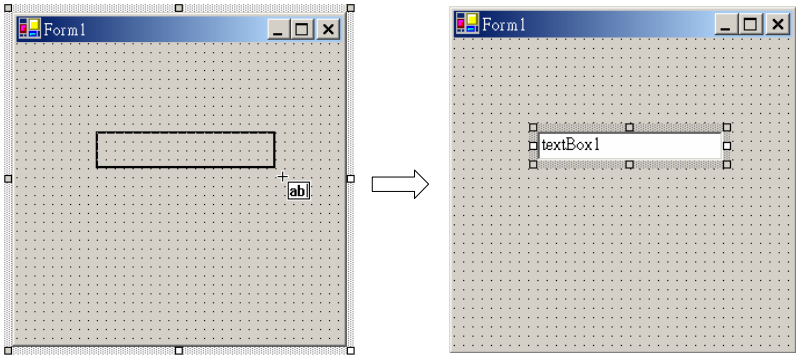


# 3-2-2 在表單上放置控制項

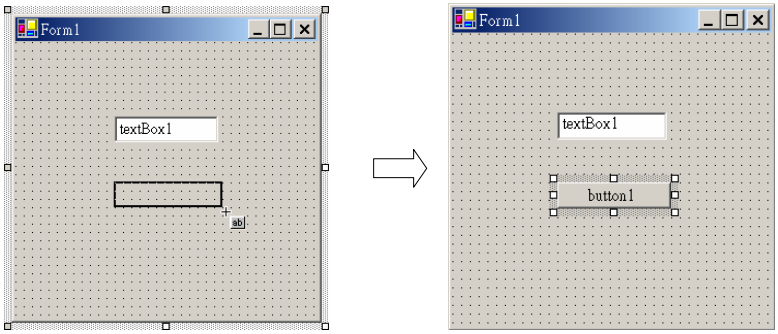
- 調整表單大小




- 插入文字方塊控制項

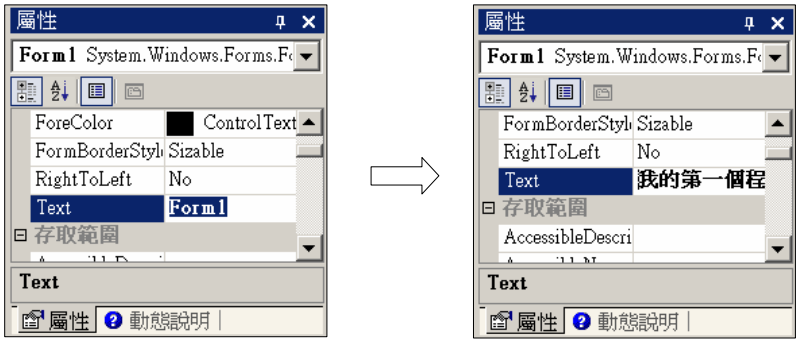


- 插入按鈕控制項

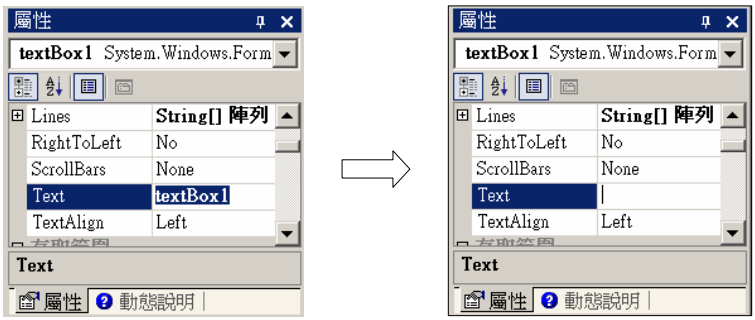



## 3-2-3 設定表單與控制項的屬性

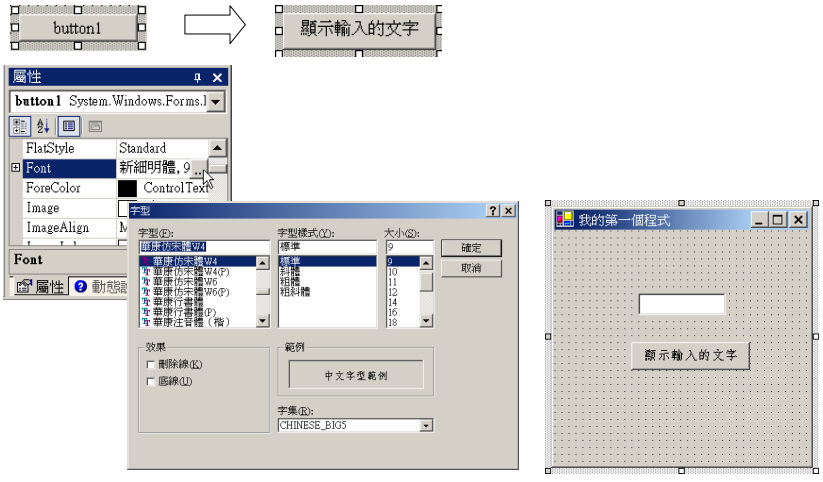
- 設定表單的屬性



- 設定文字方塊的屬性

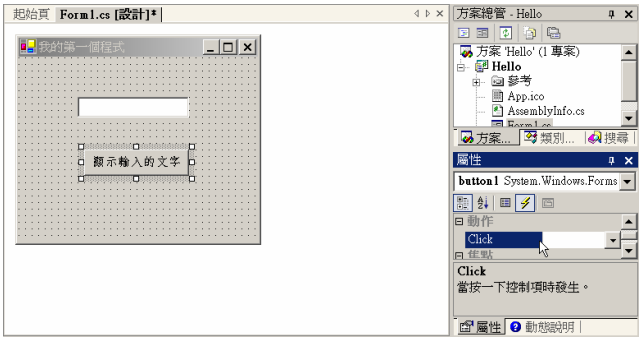



- 設定按鈕的屬性

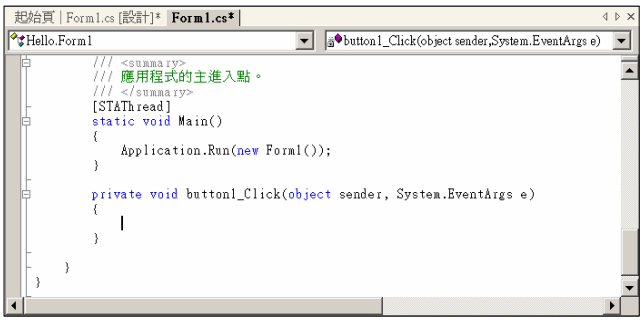


# 3-2-4 撰寫程式碼

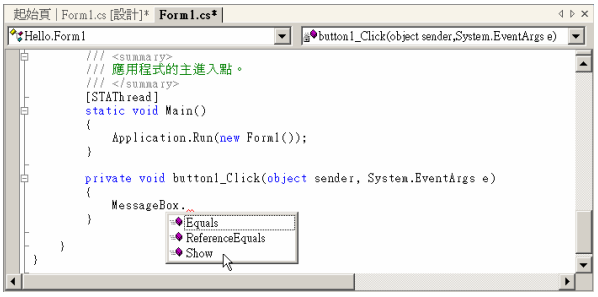
1. 首先，選取要撰寫事件處理程序的控制項，此處爲 [顯示輸入的文字] 按鈕，接著，點取屬性視窗的 [事件] 按鈕，然後在要處理的事件名稱按兩下，此處是在Click事件名稱按兩下。




2. Visual Studio.NET 自動產生如下圖的程式碼，要注意的是名稱爲 “button1_Click” 的方法，當使用者按一下 button1 按鈕時，系統會產生一個 Click 事件，進而呼叫這個方法去做處理。

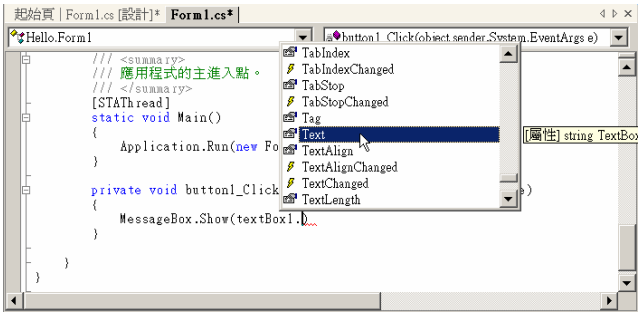


3. 將插入點移到 “button1_Click” 方法裡面，然後開始輸入程式碼，請注意英文字母的大小寫，當您輸入到 MessageBox. 時，螢幕上會自動出現一個下拉式清單，裡面列出了 MessageBox 類別的所有方法，這是 Visual Studio.NET 的 “IntelliSense” 功能，目的是讓程式開發人員不用牢記一堆方法或屬性的拼法，您可以自己輸入 Show，也可以從下拉式清單中找到 Show 方法，然後按兩下，Show 就會出現在程式碼內。




4. 繼續輸入到括號 () 時，螢幕上會自動出現 Show() 方法的語法，說明這個方法的參數就是要顯示在對話方塊內的文字。

5. 繼續輸入 Show() 方法的參數，這是使用者在文字方塊內所輸入的文字，同樣的，在輸入到 textBox1. 時，螢幕上會自動出現一個下拉式清單，裡面列出了 TextBox 控制項的所有屬性，在此，我們是選取 Text 屬性，然後於右括號的後面輸入分號，做爲這個敘述的結尾。




## 3-2-5 開始執行程式

- 我們的第一個 Visual C#.NET 程式終於寫好了，趕快來執行吧！請按 [F5] 鍵或點取標準工具列的 [開始] 按鈕，Visual C#.NET 會先進行編譯，相關資訊顯示在 [輸出] 視窗，確定沒有錯誤之後便會出現如下的執行結果，至於編譯完畢的可執行檔則會放在該專案資料夾內的 \bin\Debug 子資料夾

## 3-2-6　儲存檔案、專案與方案

- 如果您只是要儲存目前正在編輯的檔案，可以點取標準工具列的 [儲存檔案] 按鈕，或從 [檔案] 功能表中選取 [儲存xxx]，xxx爲檔案的名稱。
- 如果您要將目前正在編輯的檔案以其它名稱儲存，可以從 [檔案] 功能表中選取 [另存xxx爲]，xxx爲檔案的名稱。
- 如果您要儲存專案，可以在方案總管內找到這個專案，按一下滑鼠右鍵，然後選取 [儲存xxx]，xxx爲專案的名稱。
- 如果您要儲存方案，可以在方案總管內找到這個方案，按一下滑鼠右鍵，然後選取 [儲存xxx.sln]，xxx.sln爲方案的名稱。
- 如果您要儲存全部的檔案、專案及方案，可以點取標準工具列的 [全部儲存] 按鈕，或從 [檔案] 功能表中選取 [全部儲存]。

## 3-2-7　關閉檔案、專案與方案

- 如果您只是要關閉 Windows Form 設計工具或目前正在編輯的檔案，可以點取 Windows Form 設計工具或程式碼視窗右上角的 [關閉] 按鈕。
- 如果您要關閉整個方案，可以從 [檔案] 功能表中選取 [關閉方案]，此時若有尚未儲存的檔案，螢幕上會顯示對話方塊詢問您是否加以儲存。

## 3-2-8 開啓檔案、專案與方案

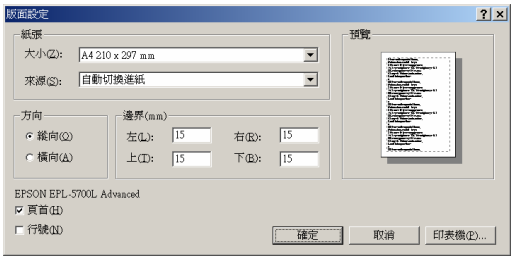
- 如果您要開啓的檔案屬於目前開啓的方案，可以在方案總管內找到這個檔案，然後按兩下；如果您要開啓的檔案不屬於目前開啓的方案，或目前並沒有開啓任何方案，可以選取 [檔案] \ [開啓] \ [檔案]，然後在 [開啓檔案] 對話方塊中選擇所要開啓的檔案。
- 如果您要開啓專案，可以選取 [檔案] \ [開啓] \ [專案]，然後在 [開啓專案] 對話方塊中選擇所要開啓的專案。
- 如果您要開啓方案，可以選取 [檔案] \ [開啓方案]，然後在 [開啓方案] 對話方塊中選擇所要開啓的方案。
- Visual Studio.NET 還提供了一個更快速的方式來開啓最近使用過的檔案或專案，就是選取 [檔案] \ [最近使用過的檔案] 或 [檔案] \ [最近使用過的專案]，然後從子功能表中選取檔案或專案的名稱。

## 3-2-9 刪除檔案、專案與方案

- 如果您要刪除檔案，可以在方案總管內找到這個檔案，然後在檔案的名稱按一下滑鼠右鍵，從快顯功能表中選取 [刪除]。
- 如果您要將專案自方案中移除，可以在方案總管內找到這個專案，然後在專案的名稱按一下滑鼠右鍵，從快顯功能表中選取 [移除]，不過，這個專案依然存在於磁碟，您可以再度將它加入方案 (在方案的名稱按一下滑鼠右鍵，從快顯功能表中選取 [加入] \ [新增專案])，若要從磁碟刪除這個專案，可以找到它在磁碟的位置，然後刪除其資料夾。
- 如果您要刪除方案，可以找到它在磁碟的位置，然後刪除其資料夾。

## 3-2-10 列印

- 當您要列印程式碼時，請先開啓程式碼視窗，然後選取 [檔案] \ [列印]，在 [列印] 對話方塊中選擇印表機、列印範圍與列印份數。
- 如果您要設定列印時的紙張、來源、邊界、橫向、縱向、列印頁首、行號，可以選取 [檔案] \ [版面設定]，然後在 [版面設定] 對話方塊中做設定。




## 3-3 程式碼剖析

- 我們以 Hello 專案的 Form1.cs 爲例：

```
01:using System;
02:using System.Drawing;
03:using System.Collections;
04:using System.ComponentModel;
05:using System.Windows.Forms;
06:using System.Data;
07:
08:namespace Hello
09:{
10:  /// <summary>
11:  /// Form1的摘要描述。
12:  /// </summary>
```

```
13:  public class Form1 : System.Windows.Forms.Form
14:  {
15:    private System.Windows.Forms.TextBox textBox1;
16:    private System.Windows.Forms.Button button1;
17:    /// <summary>
18:    /// 設計工具所需的變數。
19:    /// </summary>
20:    private System.ComponentModel.Container components = null;
21:
22:    public Form1()
23:    {
24:      //
25:      // Windows Form設計工具支援的必要項
26:      //
```

```
27:      InitializeComponent();
28:      //
29:      // TODO:在 InitializeComponent呼叫之後加入任何建構函式程式碼
30:      //
31:    }
32:
33:    /// <summary>
34:    /// 清除任何使用中的資源。
35:    /// </summary>
36:    protected override void Dispose( bool disposing )
37:    {
38:      if( disposing )
39:      {
40:        if (components != null)
```

```
41:       {
42:         components.Dispose();
43:       }
44:     }
45:     base.Dispose( disposing );
46:   }
47:
48:   /// <summary>
49:   /// 應用程式的主進入點。
50:   /// </summary>
51:   [STAThread]
52:   static void Main()
53:   {
54:     Application.Run(new Form1());
55:   }
56:
```

```
57:   private void button1_Click(object sender, System.EventArgs e)
58:   {
59:     MessageBox.Show(textBox1.Text);
60:   }
61: }
62:}
```

## 3-4 程式碼撰寫慣例

- 在開始撰寫 C# 程式之前，我們先來瞭解一下程式碼撰寫慣例，例如程式結構、變數、程序、類別的命名規則、程式碼註解、格式、縮排等，雖然不是硬性規定，但遵循這些慣例卻可以提高程式的可讀性，讓程式更容易偵錯與維護。

## 3-4-1 C# 程式結構

- using 陳述式
- namespace 陳述式
- class 陳述式
- Main() 程序
- public static int Main(string[] args)

## 3-4-2　C# 命名規則

- 識別字 (名稱) 必須以英文字母、中文字或底線 (_) 開頭，雖然識別字裡面能夠包含數字，但不能夠以數字開頭。
- 識別字可以包含 Unicode 字元，Unicode 字元的格式爲 /uXXXX，其中 XXXX 是該 Unicode 字元的十六進位表示法，例如 /u005fUserNam e就是等於 _UserName，因爲 /u005f 所代表的 Unicode 字元正是底線 (_)。
- 每個單字的開頭建議以大寫字母表示，例如 UserName、MyFirstForm。
- 變數名稱的開頭建議以型別簡寫表示，例如 intMySalary、strMyName。

- 方法名稱的開頭建議以動詞表示，例如 InitializeComponent、CloseDialog。
- 類別或屬性名稱的開頭建議以名詞表示，例如 EmployeeRecord、CarData。
- 介面名稱的開頭建議以大寫字母I表示，例如 IComponent。
- 事件處理程序名稱的結尾建議以 EventHandler 表示，例如 MouseEventHandler。
- 對於經常使用的名稱，可以使用合理的簡寫，例如以 XML 來代替 eXtensible Markup Language。
- 避免在內部範圍使用與外部範圍相同的名稱，以免存取錯誤。

## 3-4-3 Visual C#.NET 程式碼註解

- 適當地加上註解可以提高程式的可讀性，C# 提供了兩種註解符號，其中 // 爲單行註解，/* */ 爲多行註解

## 3-5 如何使用 MessageBox.Show() 方法？

- 在對話方塊內顯示參數 *str* 所指定的字串，傳回值爲 DialogResult 列舉。

  public static DialogResult Show(str);
- 在參數 IWin32Window 指定的物件前面顯示對話方塊，而對話方塊內的字串則爲參數 str，傳回值爲 DialogResult 列舉。

  public static DialogResult Show(*IWin32Window*, *str*);
- 在對話方塊內顯示參數 *str1* 所指定的字串，而對話方塊的標題文字則爲參數 *str2*，傳回值爲 DialogResult 列舉。

  public static DialogResult Show(str1, str2);

- 在參數 *IWin32Window* 指定的物件前面顯示對話方塊，而對話方塊內的字串及標題文字則分別爲參數 *str1*、*str2*，傳回值爲 DialogResult 列舉。

  public static DialogResult Show(IWin32Window, str1, str2);
- 在對話方塊內顯示參數 *str1* 所指定的字串及參數 *buttons* 所指定的按鈕，而對話方塊的標題文字則爲參數 *str2*，傳回值爲 DialogResult 列舉。

  public static DialogResult Show(str1, str2, buttons);
- 在參數 IWin32Window 指定的物件前面顯示對話方塊，而對話方塊內的字串、標題文字、按鈕則分別爲參數 str1、str2、buttons，傳回值爲 DialogResult 列舉。

  public static DialogResult Show(IWin32Window, str1, str2, buttons);

- 在對話方塊內顯示參數 str1 所指定的字串、參數 buttons 所指定的按鈕及參數 icon 所指定的圖示，而對話方塊的標題文字則爲參數 str2，傳回值爲 DialogResult 列舉。

  public static DialogResult Show(str1, str2, buttons, icon);
- 在參數 IWin32Window 指定的物件前面顯示對話方塊，而對話方塊內的字串、標題文字、按鈕、圖示則分別爲參數 str1、str2、buttons、icon，傳回值爲 DialogResult 列舉。

  public static DialogResult Show(IWin32Window, str1, str2, buttons, icon);

- 在對話方塊內顯示參數str1所指定的字串、參數buttons所指定的按鈕、參數icon所指定的圖示及參數DefaultButton所指定的預設按鈕，而對話方塊的標題文字則爲參數str2，傳回值爲DialogResult列舉。

  public static DialogResult Show(str1, str2, buttons, icon, DefaultButton);

- 在參數 *IWin32Window* 指定的物件前面顯示對話方塊，而對話方塊內的字串、標題文字、按鈕、圖示、預設按鈕則分別爲參數 *str1*、*str2*、*buttons*、*icon*、*DefaultButton*，傳回值爲 DialogResult 列舉。

  public static DialogResult Show(IWin32Window, str1, str2, buttons, icon, DefaultButton);

- 在對話方塊內顯示參數 *str1* 所指定的字串、參數 *buttons* 所指定的按鈕、參數 *icon* 所指定的圖示、參數 *DefaultButton* 所指定的預設按鈕及參數 *options* 所指定的選項，而對話方塊的標題文字則爲參數 *str2*，傳回值爲 DialogResult 列舉。

  public static DialogResult Show(*str1*, *str2*, *buttons*, *icon*, *DefaultButton*, *options*);

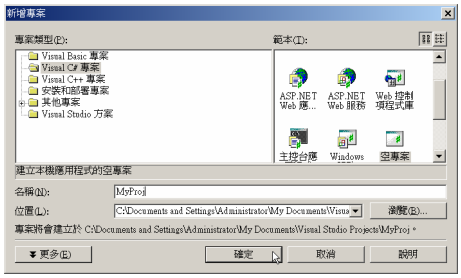
- 在參數 *IWin32Window* 指定的物件前面顯示對話方塊，而對話方塊內的字串、標題文字、按鈕、圖示、預設按鈕、選項則分別爲參數 *str1*、*str2*、*buttons*、*icon*、*DefaultButton*、*options*，傳回值爲 DialogResult 列舉。

  public static DialogResult Show(*IWin32Window*, *str1*, *str2*, *buttons*, *icon*, *DefaultButton, options*);

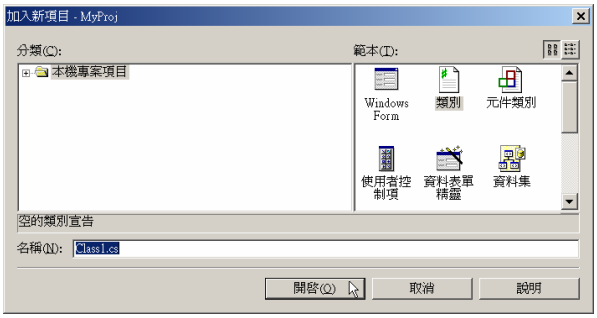
## 3-6 如何撰寫沒有表單的應用程式？

1. 啓動Visual Studio.NET。
2. 點取 [起始頁] 的 [新增專案] 按鈕，或從功能表列選取 [檔案] \ [新增] \ [專案]。

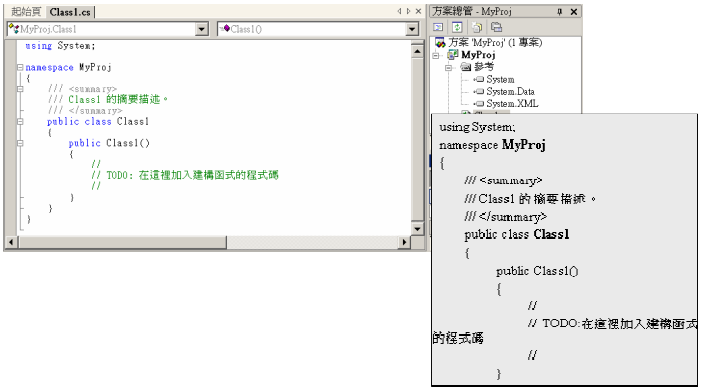
3. 首先，在 [專案類型] 欄位選擇 [Visual C# 專案]；接著，在 [範本] 欄位選擇 [空專案]；繼續，在 [名稱] 欄位輸入專案的名稱，預設爲 Project1，在此我們輸入 MyProj，這個專案存放在磁碟的路徑將爲 \My Documents\Visual Studio Projects\MyProj，同時專案的名稱就是應用程式預設的命名空間；最後，按下 [確定]。



4. 由於目前爲空專案，所以我們要加入類別，請從功能表列選取 [專案] \ [加入類別]，然後在如下對話方塊中輸入類別的名稱，再按 [開啓]。




5. 新增的類別內自動出現如下程式碼，由於沒有指定命名空間及類別的名稱，所以預設的命名空間及類別分別爲專案的名稱和類別檔案的名稱。



6. 我們可以加入自己的程式碼，例如撰寫一個Main() 程序，輸入完畢之後按下 [F5] 鍵編譯並執行程式：


## 3-7 如何使用主控台輸入/輸出？

- 主控台輸入 / 輸出指的是從標準輸入 (鍵盤) 讀取使用者輸入的字串，以及將執行結果或錯誤訊息顯示在標準輸出 (命令提示字元模式)。
- 我們可以使用 System 命名空間的 Console 類別的 Read()、ReadLine() 方法，分別從標準輸入讀取一個字元和一行資料，以及使用 System 命名空間的 Console 類別的 Write()、WriteLine() 方法，分別在標準輸出顯示一個字元和一行資料，例如：

```
char Data1 = System.Convert.ToChar(System.Console.Read());
Console.Write(Data1);
```

## 3-8 如何在命令列編譯 C# 程式？

- 如果您沒有安裝 Visual Studio.NET，那麼您可以執行隨書光碟的 \dotNetFramework\dotnetfx.exe 檔案來安裝 Microsoft .NET Framework，安裝完畢之後就可以在類似 C:\WINNT\Microsoft.NET\Framework\v1.0.3705 的資料夾內找到 C# 的編譯器 csc.exe。

1. 開啓記事本編輯如下程式碼，然後存檔爲 Test.cs，記得副檔名要變更爲 .cs，而不是預設的 .txt，這個程式的作用是顯示 “Hello World!” 字串。

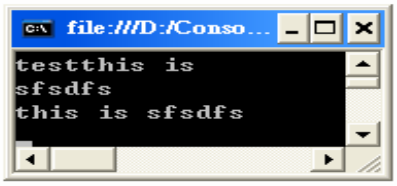
```
using System;
// 匯入System命名空間 ， 直接呼叫Console類別的WriteLine() 方法
class MyClass {
  public static void Main() {
     Console.WriteLine("Hello World!");
  }
}
```

2. 爲了方便起見，我們直接將這個程式存放在和編譯器相同的資料夾內，接下來，我們要進行編譯，請選按 [開始] \ [程式集] \ [附屬應用程式] \ [命令提示字元]，然後於命令提示字元下將目錄切換到編譯器所在的資料夾，例如輸入 cd C:\WINNT\Microsoft.NET\Framework\v1.0.3705，再按 [Enter]。

3. 在命令列輸入 csc Test.cs，然後按 [Enter]，開始將 C# 程式編譯成 .exe 執行檔，若發現錯誤，則會在螢幕上顯示訊息，屆時再根據錯誤訊息做修正即可。順利編譯完畢之後，可以在相同目錄下找到同名的 .exe 執行檔，例如 Test.exe，直接在命令列輸入此執行檔的名稱，然後按 [Enter]，就會在螢幕上顯示 “Hello World!” 字串。

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

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("test");
            Console.WriteLine("this is");
            //char i = System.Convert.ToChar(System.Console.Read());
            //Console.WriteLine("this is "+i);
            string j="jk" ;
            j=Console.ReadLine();
            Console.WriteLine("this is " + j);
            Console.ReadLine();
        }
    }
}
```

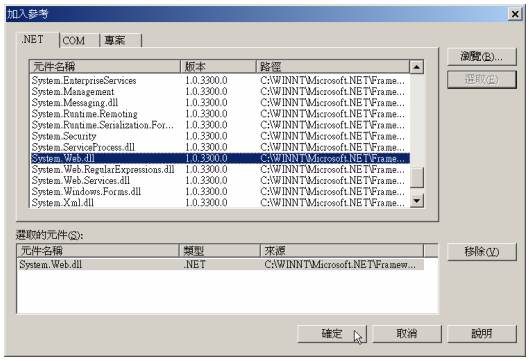



## 3-9 小錦囊 (五) 如何使用命令列參數？

- C# 提供了兩種可以接受命令列參數的 Main() 程序，其形式如下：
  - public static void Main(string[] args)
  - public static int Main(string[] args)

# 3-10 小錦囊 (六) 如何匯入命名空間與設定別名？

- 在方案總管內的「參考」項目按一下滑鼠右鍵，然後從快顯功能表中選取 [加入參考]，就可以在如下對話方塊中新增參考：




- 在方案總管內的專案名稱按一下滑鼠右鍵，然後選取 [屬性]，就可以在如下對話方塊看到專案所匯入的命名空間清單：